



– *Technical Report* –

Registration No.

OPSEC#:8590

Date of Report

04/25/2024

COMPENDIUM OF TECHNICAL NOTES AND PROBLEM SOLUTIONS IN DYNAMICS AND STRUCTURES 2010 – 2023

W. Bylsma
Modeling, Simulation & Software/Analytics/Structures
U.S. Army Combat Capabilities Development Command (DEVCOM)
Ground Vehicle Systems Center (GVSC)
Detroit Arsenal
ATTN: FCDD-GVS-MSP/MS 157
6501 East 11 Mile Road
Warren, Michigan 48397-5000
wesley.w.bylsma.civ@army.mil

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

OPSEC#: 8590

Disclaimer: Reference herein to any specific commercial company, product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the Department of the Army (DoA). The opinions of the authors expressed herein do not necessarily state or reflect those of the United States Government or the DoA, and shall not be used for advertising or product endorsement purposes.

U.S. Army Combat Capabilities Development Command (DEVCOM)
Ground Vehicle Systems Center (GVSC)
Detroit Arsenal
Warren, Michigan 48397-5000

REPORT DOCUMENTATION PAGE			<i>Form Approved</i> <i>OMB No. 0704-0188</i>		
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 25-04-2024		2. REPORT TYPE TECHNICAL		3. DATES COVERED (From - To) 2010-2023	
4. TITLE AND SUBTITLE COMPENDIUM OF TECHNICAL NOTES AND PROBLEM SOLUTIONS IN DYNAMICS AND STRUCTURES 2010 - 2023.		5a. CONTRACT NUMBER			
		5b. GRANT NUMBER			
		5c. PROGRAM ELEMENT NUMBER			
6. AUTHOR(S) WESLEY BYLSMA		5d. PROJECT NUMBER			
		5e. TASK NUMBER			
		5f. WORK UNIT NUMBER			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) ANALYTICS-US ARMY DEVCOM/GVSC STRUCTURES TEAM ATTN: FCDD-GVS-MSP/MS157 6501 E 11 MILE RD WARREN, MI 48397-5000		8. PERFORMING ORGANIZATION REPORT NUMBER OPSEC: 8590			
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)			
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)			
12. DISTRIBUTION / AVAILABILITY STATEMENT DISTRIBUTION STATEMENT A. Approved for public release. OPSEC#: 8590					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT A compendium (a collection of concise but detailed information about a particular subject) of technical notes and problem solutions in Dynamics and Structures (2010 - 2023). Subject matter includes course profile detrending, absorbed power calculation, vehicle cone index calculation, course profile generation, trailer payload limits, sine sweep signal generation, track patter vibration verification, Ramberg-Osgood stress/strain, etc.					
15. SUBJECT TERMS course profile detrending, absorbed power calculation, vehicle cone index calculation, course profile generation, trailer payload limits, sine sweep signal generation, track patter vibration verification, Ramberg-Osgood stress/strain					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT Unclassified	18. NUMBER OF PAGES 443	19a. NAME OF RESPONSIBLE PERSON Wesley Bylsma
a. REPORT Unclassified	b. ABSTRACT Unclassified	c. THIS PAGE Unclassified			19b. TELEPHONE NUMBER (include area code) 571-588-6444

Contents (by Title)

SUMMARY	4
DYNAMICS SOFTWARE 7-BODY EXAMPLE COMPARISON – DADS AND RECURDYN.....	20
EXPONENTIAL COURSE DETRENDING	38
ABSORBED POWER CALCULATION	48
WHEELED VEHICLE ONE-PASS VEHICLE CONE INDEX (VCI) FORMULA	62
POLAR PLOTTING – MULTIPLE PLOTS AND TRANSPARENCY.....	69
ABSORBED POWER FUNCTION USING MATLAB’S ODE45.....	72
DETRENDING COURSE ELEVATION DATA	86
TRAILING ARM SUSPENSION STATIC ANGLE SOLUTION USING MICROSOFT EXCEL....	89
SIMULATING CHOCKS USING 2D ROAD PROFILES.....	93
TRACK LENGTH CALCULATION FROM ROAD WHEEL POSITIONS	98
“FRAME TWISTER” WAVE COURSE GENERATION IN 3D	121
ROAD SURFACE GENERATION IN RDF FORMAT	131
ABSORBED POWER CALCULATION IN THE FREQUENCY DOMAIN	144
GROUND MOBILITY VEHICLE (GMV) WAVENUMBER TERRAINS	157
PROFILE IV COURSE GENERATION	160
RECURDYN HEIGHT RELATIVE TO GROUND DETERMINATION.....	172
TRAILER PAYLOAD POSITION LIMITS	179
ENGLISH UNITS OF LBM VERSUS LBF.....	186
BUSHING RADIAL RATE DECAY MAXIMUM CURVATURE.....	188
BUSHING RADIAL RATE DECAY MAXIMUM CURVATURE - CALCULATING IN EXCEL AND MATLAB.....	192
M2 BRADLEY WITH CONCEPT VEHICLE PROTOTYPING (CVP) EXTERNAL SUSPENSION UNIT (ESU) ANIMATION	199
AUTOMATING THE CREATION OF THE WELD DEFINITION FILE IN FE-SAFE	203
CREATING ABAQUS JOINT/UJOINT CONNECTORS IN HYPERMESH.....	212
GENERATING A TIME HISTORY FROM A FREQUENCY SPECTRUM USING NCODE GLYPHWORKS/VIBESYS	233
GENERATING SINE SWEEP/CHIRP SIGNALS	249
FUNCTIONS FOR GENERATING POWER SPECTRAL DENSITY (PSD) INPUTS FOR FE-SAFE FREQUENCY BASED FATIGUE ANALYSIS	271
ESTIMATING RAYLEIGH DAMPING COEFFICIENTS FOR MODAL DAMPING	302
MODIFIED GOODMAN DIAGRAM FROM ULTIMATE, YIELD, AND ENDURANCE STRENGTH (SU, SY, AND SE)	313
TRACK PATTERN FREQUENCIES.....	329
RANDOM RESPONSE OUTPUT USING TRANSMISSIBILITY.....	336
ABAQUS/STANDARD MODAL ANALYSIS EFFECTIVE MASS AS A PERCENTAGE OF TOTAL EFFECTIVE MASS	344

SENSITIVITY OF FATIGUE LIFE TO CHANGES IN STRESS LEVELS	349
MARGIN OF SAFETY CRITERIA FOR FASTENER PRELOAD/CLAMP FORCE REQUIREMENT	354
PART SELECTION BASED ON YIELD STRENGTH.....	378
ALTAIR HYPERWORKS LICENSE USAGE MONITORING.....	384
POWERSHELL SCRIPT AUTO STARTUP	405
ALTAIR HYPERWORKS LICENSE ACTIVITY MATRIX	408
ABAQUS/STANDARD TO ABAQUS/EXPLICIT CONVERSION ISSUES.....	414
ABAQUS/EXPLICIT HISTORY OUTPUT EXTRACTION USING PYTHON.....	422
RAMBERG-OSGOOD: FROM NEUBER CORRECTION TO PLASTICITY	433

Contents (by Number)

TN 10-0604-01.....	20
TN 10-0729-01.....	38
TN 10-1021-01.....	48
TN 11-0718-01.....	62
TN 12-0727-01.....	69
TN 12-1219-01.....	72
TN 13-0528-01.....	86
TN 13-1113-01.....	89
TN 15-0304-01.....	93
TN 15-0519-01.....	98
TN 15-0709-01.....	121
TN 15-0710-01.....	131
TN 15-0930-01.....	144
TN 15-1021-01.....	157
TN 15-1027-01.....	160
TN 16-0315-01.....	172
TN 16-0419-01.....	179
TN 16-0722-01.....	186
TN 16-0922-01.....	188
TN 16-1207-01.....	192
TN 17-1026-02.....	199
TN 18-0405-01.....	203
TN 18-1129-01.....	212
TN 19-0301-01.....	233
TN 19-0514-01.....	249

TN 19-0801-01.....	271
TN 19-0807-01.....	302
TN 21-0308-01.....	313
TN 21-0317-01.....	329
TN 21-0331-01.....	336
TN 21-0406-01.....	344
TN 21-0407-01.....	349
TN 21-0723-01.....	354
TN 22-0427-01.....	378
TN 22-0524-01.....	384
TN 22-1101-01.....	405
TN 22-1103-01.....	408
TN 23-0516-01.....	414
TN 23-0516-02.....	422
TN 23-0531-01.....	433

SUMMARY

	2010
TN 10-0604-01	DYNAMICS SOFTWARE 7-BODY EXAMPLE COMPARISON - DADS and RECURDYN Describes output comparison between DADS and RecurDyn for a standardized benchmark. [1] Schiehlen (Editor), Multibody Systems Handbook, Springer-Verlag, ISBN 0-387-51946-7. See also https://books.google.com/books/about/Multibody_Systems_Handbook.html?id=UYz_CAAAQBAJ [2024-04-15] [2] Reference Project 10-080 Dynamics Software 7-body Example Comparison - DADS VS RECURDYN [3] Library of Computational Benchmark Problems, IFToMM Technical Committee for Multibody Dynamics (https://www.ifto-mm-multibody.org/benchmark/) See Andrews Squeezer Mechanism. [4] Tagliapietra, et al. MultiBody Systems Benchmark in OpenSIM (MBS-BOS) - User Manual Release 2.0 (Sep 1, 2015). Ch. 3 - MBS Benchmark A03: Andrew's Squeezer Mechanism. (https://simtk.org/docman/view.php/984/1910/MBS-BOSUserGuide.pdf) [5] Schiehlen, Valasek. Virtual Nonlinear Multibody Systems-NATO Advanced Study Institute. Vol 1. Prague, 23 Jun - 3 Jul 2002. DIST A (DTIC AD A409 218). Approved for public release: distribution unlimited. (pp. 144-149)
TN 10-0729-01	EXPONENTIAL COURSE DETRENDING Describes the standard course detrending filter used by WES and APG. [1] Murphy, Newell. A Method for Determining Terrain Surface Roughness. U.S. Army Engineer Waterways Experiment Station, Geotechnical Laboratory. September 1984. [2] Van Deusen, B. D. A Statistical Technique for the Dynamic Analysis of Vehicles Traversing Rough Yielding and Non-yielding Surfaces. NASA CR-659. Prepared by CHRYSLER CORPORATION, Detroit, Mich. for NATIONAL AERONAUTICS AND SPACE ADMINISTRATION, WASHINGTON, D. C., MARCH 1967 [3] Test Operations Procedure (TOP) 01-1-010 Vehicle Test Course Severity (Surface Roughness) (31 Oct 2006). DIST A. Approved for public release: distribution unlimited.
TN 10-1021-01	ABSORBED POWER CALCULATION Describes the calculation of the absorbed power human tolerance criteria in the time (w) and frequency (s) based domains. Also makes comparison to ISO 2631. [1] Test Operations Procedure (TOP) 1-1-14 Ride Dynamics (7 Jan 2005). DIST A (DTIC AD A431 130). Approved for public release: distribution unlimited. [2] Lee and Lins. "Human Vibration Measuring Instrument". Army Tank-Automotive Command Warren, MI. 1973. DIST A (DTIC AD 0785 648). Approved for public release: distribution unlimited. [3] Creighton, D. C. "Revised vehicle dynamics module: user's guide for computer program VEHDYN II". U.S. Army Engineer Waterways Experiment Station, Vicksburg, MS. 1986. TR SL-86-9. DIST C (DTIC AD B102 925). [4] Lins. "Human Vibration Response Measurement". U.S. Army Tank-automotive Command Laboratories. 1972. TR 11551. DIST C (DTIC AD 0901 195). [5] ISO 2631-1:1997(E). "Mechanical vibration and shock – Evaluation of human exposure to whole-body vibration Part 1: General requirements. 2nd Edition-corrected and reprinted 1997-07-15. [6] Murphy and Ahmad. "Comparison of Measures of Vibration Affecting Occupants of Military Vehicles". Waterways Experiment Station (WES), Corps of Engineers, Vicksburg, MS 39180-0631. Final Report, December 1986. TR GL-86-18. DIST A (DTIC AD A178 359). Approved for public release: distribution unlimited.
	2011

TN 11-0718-01	WHEELED VEHICLE ONE-PASS VEHICLE CONE INDEX (VCI) FORMULA Describes Vehicle Cone Index soft-soil performance formula for Annex E to the HMMWV Modernized Expanded Capacity Vehicle (MECV) Performance Specification (program was cancelled). [1] Reference Internal Project 10-038 MECV [2] "Clarification of vehicle cone index with reference to mean maximum pressure", Journal of Terramechanics 43 (2006) 85-96, J. D. Priddy and W. E. Willoughby, (www.elsevier.com/locate/jterra). [3] NATO Reference Mobility Model Ed II, NRMM II User's Guide. Waterways Experiment Station (WES), Corps of Engineers, Vicksburg, MS 39180-0631. Final Report, December 1992. TR GL-92-19. DIST C (DTIC AD B170 301). VPP.FOR (vehicle preprocessor subroutine). [4] Performance Specification for HMMWV Modernized Expanded Capacity Vehicle (MECV) (ver 2.0, 7 Nov 2011). Annex E – Single-Pass Vehicle Cone Index (VCI) Calculation. DIST A. Approved for public release: distribution unlimited. [5] Kennedy, Rush. Trafficability of Soils (Technical memorandum No. 3-240) – Development of Revised Mobility Index Formula for Self-Propelled Wheeled Vehicles in Fine-Grained Soils. US Army Engineer Waterways Experiment Station, Corps of Engineers (March 1968). Appendix A: Vehicle Cone Index Computations. DIST A (DTIC AD 832 912)). Approved for public release: distribution unlimited. [6] Wong. Theory of Ground Vehicles, 3rd Ed. John Wiley & Sons, 2001 (pp. 120-126). [7] Rosco, et. al. Considerations Regarding Vehicle Cone Index for Fine-Grained Soil. Journal of Military Technology, Vol. 1, No. 2, Dec. 2018). [8] Ragheb, et al. Evaluation of Wheeled Vehicle Mobility and Performance Prediction on Deformable Soil. Proceedings of the 13th Int. AMME Conf., 27-29 May, 2008. [9] Bylsma. SIMPLIFIED DYNAMICS AND MOBILITY FACTORS FOR MULTI-DISCIPLINARY OPTIMIZATION OF AN OCCUPANT CENTRIC PLATFORM, TR 22829, April 2012. DIST A (DTIC AD A559 920) Approved for public release: distribution unlimited.
	2012
TN 12-0727-01	POLAR PLOTTING – MULTIPLE PLOTS AND TRANSPARENCY Describes plotting field of view data using MATLAB's polar plot function. [1] MATLAB (www.mathworks.com) [2] Reference Internal Project Misc/2-polarpatch

TN 12-1219-01	ABSORBED POWER FUNCTION USING MATLAB'S ODE45 Describes the calculation of the absorbed power human tolerance criteria using MATLAB's ODE45 function with several examples. [1] ISO/IEC 17025 General requirements for the competence of testing and calibration laboratories (http://en.wikipedia.org/wiki/ISO/IEC_17025) [2] Test Operations Procedure (TOP) 01-1-014A Ride Dynamics and Evaluation of Human Exposure to Whole Body Vibration (3 Apr 2012). DIST A (DTIC AD A559 451). Approved for public release: distribution unlimited. [3] "Overview of Human Vibration Weighting Filters", 5 February 2008. (http://www.ni.com/white-paper/6957/en) [4] Rimell, Mansfield. "Design of Digital Filters for Frequency Weightings Required for Risk Assessments of Workers Exposed to Vibration", Industrial Health 2007, 45, 512–519, April 7, 2007 (http://www.google.com/url?sa=t&rct=j&q=digital+filters+industrial+health+2007&source=web&cd=2&cad=rja&ved=0CDIQFjAB&url=http%3A%2F%2Fwww.researchgate.net%2Fpublication%2F47450362_Design_of_digital_filters_for_frequency_weightings_required_for_risk_assessments_of_workers_exposed_to_vibration%2Ffile%2F4912f50928c8235dad.pdf&ei=pvCHUa6xKlBj4AP954CICQ&usg=AFQjCNE4WYGZQi4zE23jX6AEf0dVN3wZcQ) [5] Svantek Whole-body vibration meter. (http://svantek.com/multichannel-measurements/sv-100-whole-body-vibration-dosimeter-analyser.html#guide) [6] MATLAB (www.mathworks.com) [7] ode45 (http://www.mathworks.com/help/matlab/ref/ode45.html) [8] interp1 (http://www.mathworks.com/help/matlab/ref/interp1.html) [9] Creighton, D. C. "Revised vehicle dynamics module: user's guide for computer program VEHDYN II". U.S. Army Engineer Waterways Experiment Station, Vicksburg, MS. 1986. TR SL-86-9. DIST C (DTIC AD B102 925). [10] Reference Internal Project 12-517 – MATLAB Absorbed Power – ode45 === [11] TN 10-1021-01 Absorbed Power Calculation [12] Lee and Lins. "Human Vibration Measuring Instrument". Army Tank-Automotive Command Warren, MI. 1973. DIST A (DTIC AD 0785 648). Approved for public release: distribution unlimited. (see pg. 207). [13] Ogata. Modern Control Engineering. Prentice-Hall, 1970. (see 14-2 State-Space Representation of Systems, pg. 675-678) [14] Transfer Function to State Space (https://lpsa.swarthmore.edu/Representations/SysRepTransformations/TF2SS.html) [15] Controllable Canonical Form (https://en.wikipedia.org/wiki/State-space_representation) [16] Runge Kutta Methods (https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta_methods) [17] tf2ss (https://www.mathworks.com/help/signal/ref/tf2ss.html) APPENDIX A.1– calcpwrfunc (VehdynII method) APPENDIX A.2 –calcpwrfunc (Ogata method) APPENDIX A.3 –calcpwrfunc (CCF method) APPENDIX A.4 –calcpwrfunc (CCF method/Direct Runge Kutta Integration) APPENDIX A.5 – poles/zeros of transfer function/filter
	2013
TN 13-0528-01	DETRENDING COURSE ELEVATION DATA Describes references for course detrending information. [1] Van Deusen, B.D. "A Statistical Technique for the Dynamic Analysis of Vehicles Traversing Rough Yielding and Non-Yielding Surfaces". NASA Report No. CR-659, March 1967 [2] TN 10-0729-01 Exponential Course Detrending

TN 13-1113-01	TRAILING ARM SUSPENSION STATIC ANGLE SOLUTION USING MICROSOFT EXCEL Describes the calculation of a trailing arm suspension static roadarm angle using Microsoft Excel's Solver Add-in with the Generalized Reduced Gradient (GRG2) nonlinear optimization code. [1] Frontline Systems, Inc., P.O. Box 4288, Incline Village, NV 89450-4288, (775) 831-0300. ("http://www.solver.com", e-mail: info@solver.com).
	2015
TN 15-0304-01	SIMULATING CHOCKS USING 2D ROAD PROFILES Describes the modeling of chocks (as part of a 2d road profile) for transportability rail impact analysis. [1] TN15-0312-01 74 TON M1 TRANSPORTABILITY TIE-DOWN RAIL IMPACT FORCES ANALYSIS (see TR27441 - 2015 TECHNICAL NOTES AND PROBLEM SOLUTIONS IN DYNAMICS. DIST C (DTIC AD 1001 907)). [2] TM 55-2200-001-12 TRANSPORTABILITY GUIDANCE, APPLICATION OF BLOCKING, BRACING, AND TIEDOWN MATERIALS FOR RAIL TRANSPORT. DIST A. Approved for public release: distribution unlimited. Inactive, see (https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=779bca6195ec96a97fc51ccfcb6ca90f82b015dc). [3] TM 55-2350-255-14 TRANSPORTABILITY GUIDANCE FOR TANK COMBAT, FULL-TRACKED, M1-SERIES: 105-MM GUN, M1, 105-MM GUN, IPM1, 120-MM GUN, M1A1, 120-MM GUN, M1A2 (GENERAL ABRAMS). DIST C. (https://www.ldac.army.mil/) [4] Virtual.Lab Motion (http://www.plm.automation.siemens.com/en_us/products/lms/virtual-lab/motion/) [5] Reference Internal Project 14-511 – M1 TRANSPORTABILITY TIEDOWN RAIL IMPACT FORCES [6] TM55-2220-058-14 TRANSPORTABILITY GUIDANCE FOR TRANSPORT OF CARGO ON THE RAILCAR FLAT, 140-TON-CAPACITY. GPO. (https://www.ldac.army.mil/).
TN 15-0519-01	TRACK LENGTH CALCULATION FROM ROAD WHEEL POSITIONS Describes the calculations and functions to determine the track length of a tracked vehicle when it is wrapped around the road wheels, including road wheels inside and outside the track loop. [1] ATAN2 (https://msdn.microsoft.com/en-us/library/system.math.atan2(v=vs.110).aspx) [2] Dot Product (https://en.wikipedia.org/wiki/Dot_product) [3] Cross Product (https://en.wikipedia.org/wiki/Cross_product) [4] MATLAB (http://www.mathworks.com/) [5] Windows PowerShell (http://en.wikipedia.org/wiki/Windows_PowerShell) [6] Reference Internal Project 15-517 TRACK LENGTH CALCULATION FROM ROADWHEEL POSITIONS
TN 15-0709-01	"FRAME TWISTER" WAVE COURSE GENERATION IN 3D Describes the calculation to create a 3D wave "frame twister" course and how to extract 2D profiles to use as inputs for laboratory N-post shaker testing. [1] Test Operations Procedure (TOP) 01-1-011A Vehicle Test Facilities at Aberdeen Test Center and Yuma Test Center (27 Feb 2012). DIST A (DTIC AD A557 002). Approved for public release: distribution unlimited. [2] MATLAB (www.mathworks.com)

TN 15-0710-01	ROAD SURFACE GENERATION IN RDF FORMAT Describes 3D road surface generation in RecurDyn RDF file format. [1] RecurDyn (https://functionbay.com/en) [2] Road Data File (RDF) (https://functionbay.com/documentation/onlinehelp/default.htm#!Documents/rdffileformat.htm) [3] Windows PowerShell (http://en.wikipedia.org/wiki/Windows_PowerShell) [4] Test Operations Procedure (TOP) 01-1-011A Vehicle Test Facilities at Aberdeen Test Center and Yuma Test Center (27 Feb 2012). DIST A (DTIC AD A557 002). Approved for public release: distribution unlimited. [5] TN 15-0709-01 FRAME TWISTER WAVE COURSE GENERATION IN 3D
TN 15-0930-01	ABSORBED POWER CALCULATION IN THE FREQUENCY DOMAIN Describes the calculation of absorbed power in the frequency domain by multiplying the acceleration power spectral density (PSD) by the appropriate weighting factors and provides a comparison to calculation using a time based method. [1] Reference Internal Project 15-018/518 M870 TRAILER PERFORMANCE. [2] Test Operating Procedure (TOP) 01-1-014A Ride Dynamics and Evaluation of Human Exposure to Whole Body Vibration (3 Apr 2012). DIST A (DTIC AD A559 451). Approved for public release: distribution unlimited. [3] Lee, Pradko. Analytical Analysis of Human vibration. Mobility systems Laboratory, U.S. Army Tank-Automotive Command. SAE Paper 680091. Jan 1968. [4] MATLAB. (www.mathworks.com) [5] TN 10-1021-01 ABSORBED POWER CALCULATION [6] TN 12-1219-01 ABSORBED POWER FUNCTION USING MATLAB'S ODE45 [7] Lee and Lins. "Human Vibration Measuring Instrument". Army Tank-Automotive Command Warren, MI. 1973. DIST A (DTIC AD 0785 648). Approved for public release: distribution unlimited. (see pg. 207).
TN 15-1021-01	GROUND MOBILITY VEHICLE (GMV) WAVENUMBER TERRAINS Describes issues with identifying test courses using wave number spectra and RMS. [1] GMV2_Market_Questionnaire-GeneralInfo.docx (557.59 Kb). Ground Mobility Vehicles (GMV)-Market Questionnaire/Request for Information RFI (GMV2). Solicitation Number: W56HZV15RFIGMV2. Agency: Department of the Army. Office: Army Contracting Command. Location: ACC - Warren (ACC-WRN)(DTA) (https://sam.gov/opp/a402d68610234cfc358b09dbae971143/view) [2] Test Operating Procedure (TOP) 1-1-010 Vehicle Test Course Severity (Surface Roughness) (31 Oct 2006). DIST A. Approved for public release: distribution unlimited.
TN 15-1027-01	PROFILE IV COURSE GENERATION Describes the calculations and process to create the man-made APG Profile IV course profile. [1] Test Operations Procedure (TOP) 01-1-011A Vehicle Test Facilities at Aberdeen Test Center and Yuma Test Center (27 Feb 2012). DIST A (DTIC AD A557 002). Approved for public release: distribution unlimited. [2] http://mathworld.wolfram.com/Ellipse.html [3] MATLAB (www.mathworks.com)
	2016

16-0315-01	RECURDYN HEIGHT RELATIVE TO GROUND DETERMINATION Describes the process to obtain marker relative height from a 3D spline surface in RecurDyn. [1] RecurDyn (https://functionbay.com/en,https://functionbay.com/documentation/onlinehelp/default.htm#!Documents/help.htm) [2] SubEntity Spline (https://functionbay.com/documentation/onlinehelp/default.htm#!Documents/steptocreateaspline.htm) [3] Interpolation LINSPL (https://functionbay.com/documentation/onlinehelp/default.htm#!Documents/linspl.htm) [4] MATLAB (www.mathworks.com/) [5] TN 15-0709-01 "FRAME TWISTER" WAVE COURSE GENERATION IN 3D [6] TN 15-0710-01 ROAD SURFACE GENERATION IN RDF FORMAT
TN 16-0419-01	TRAILER PAYLOAD POSITION LIMITS Describes formulas to determine trailer payload placement to meet fifth wheel load and tire capacity limits. [1] PURCHASE DESCRIPTION, M870A4 SEMITRAILER, LOWBED, CONSTRUCTION EQUIPMENT TRANSPORTER (SLCET) 40 TON. ATPD-2395G (05 February 2015). DIST A. Approved for public release: distribution unlimited. (https://sam.gov/opp/192c730bf6926b116d7f6071b64b8802/view) [2] TM 5-2330-325-14&P. OPERATOR'S, UNIT, AND DIRECT SUPPORT MAINTENANCE MANUAL WITH REPAIR PARTS AND SPECIAL TOOLS LIST (RPSTL) FOR TRAILER, MEDIUM HEAVY EQUIPMENT TRANSPORTER (MHET), 40 TON, M870A3 (NSN 2330-01-458-2061) (June 2004). DIST A. Approved for public release: distribution unlimited. Inactive, see (http://www.combatindex.com/store/tech_man/Sample/HET_Trailers/TM_5-2330-325-14_P.pdf) [3] TM 5-2330-325-13&P. OPERATOR AND FIELD MAINTENANCE MANUAL INCLUDING REPAIR PARTS AND SPECIAL TOOLS LIST (RPSTL) FOR SEMITRAILER, MEDIUM HEAVY EQUIPMENT TRANSPORTER (MHET), 40 TON, M870A3 NSN 2330-01-458-2061 (EIC CFJ) (23 June 2014). DIST A. Approved for public release: distribution unlimited. (https://www.ldac.army.mil/). This manual supersedes TM 5-2330-325-14&P dated 30 June 2004, including all changes.
TN 16-0722-01	ENGLISH UNITS OF LBM VERSUS LBF Describes the relationship between units for mass and force in English units. [1] https://en.wikipedia.org/wiki/Pound_(force)
TN 16-0922-01	BUSHING RADIAL RATE DECAY MAXIMUM CURVATURE Describes formulas and calculations needed to determine maximum curvature for bushing radial rate decay data. [1] 832 Elastomer Biaxial Test Frame (https://www.mts.com/en/products/automotive/elastomer-test-systems/elastomer-test-systems) [2] Curvature (https://en.wikipedia.org/wiki/Curvature)
	2017

TN 16-1207-01	BUSHING RADIAL RATE DECAY MAXIMUM CURVATURE - CALCULATING IN EXCEL AND MATLAB Describes formulas, functions, and calculations to determine maximum curvature for bushing radial rate decay with examples in Microsoft Excel and MATLAB. [1] See TN16-0922-01 BUSHING RADIAL RATE DECAY MAXIMUM CURVATURE [2] Microsoft Excel (https://products.office.com/en-us/excel) [2a] LINEST (https://support.microsoft.com/en-us/office/linest-function-84d7d0d9-6e50-4101-977a-fa7abf772b6d) [2b] INDIRECT (https://support.microsoft.com/en-us/office/indirect-function-474b3a3a-8a26-4f44-b491-92b6306fa261) [3] MATLAB (www.mathworks.com) [4] Least Squares Matrix Form (http://global.oup.com/booksites/content/0199268010/samplesec3)
TN 17-1026-02	M2 BRADLEY WITH CONCEPT VEHICLE PROTOTYPING (CVP) EXTERNAL SUSPENSION UNIT (ESU) ANIMATION Describes the actions taken to produce a side by side animation of two vehicle variation to highlight performance differences. [1] RecurDyn (https://functionbay.com/en, https://functionbay.com/documentation/onlinehelp/default.htm#!Documents/help.htm)
	2018
TN 18-0405-01	AUTOMATING THE CREATION OF THE WELD DEFINITION FILE IN FE-SAFE Describes the process and scripts to help automate the creation of weld definition files for fatigue life prediction of welded joints using fe-safe/Verity and Abaqus. [1] Verity in fe-safe User Manual (https://help.3ds.com/2017/English/DSSIMULIA_Established/FesafePdf/FesafeVerity.pdf?contextscope=all&id=c1fd3777556544d8a688b96832ec8750). Requires account at https://help.3ds.com. [2] GNU sed (www.gnu.org/software/sed/) (sed –version: GNU sed version 4.1.5) [3] GNU gawk (www.gnu.org/software/gawk/) (gawk –version: GNU Awk 3.1.8) [4] GNU bash (www.gnu.org/software/bash/) (bash –version: GNU bash, version 3.2.51(1)-release (x86_64-suse-linux-gnu)) [5] SUSE (www.suse.com) (cat /etc/SuSE-release: SUSE Linux Enterprise Server 11 (x86_64) VERSION = 11 PATCHLEVEL = 3) [6] Microsoft Excel (https://products.office.com/en-us/excel)

TN 18-1129-01	CREATING ABAQUS JOINT/UJOINT CONNECTORS IN HYPERMESH Describes how to create clevis joints with spherical bearings in Altair/HyperMesh using Dassault/SIMULIA Abaqus connector elements. [1] Clevis (https://en.wikipedia.org/wiki/Clevis_fastener) [2] Example Cylinder Mounting Accessories (https://www.milwaukeeecylinder.com/downloads/) see “Milwaukee Cylinder Complete Catalog” or “Clevis/Brackets/Pins/Rod Eyes Section”. [3] HyperMesh 2017 (https://altairhyperworks.com/product/hypermesh) [2019-01-07] [4] ABAQUS/Standard 2017 (https://www.3ds.com/products-services/simulia/products/abaqus/) [2019-01-07] Below references require account at https://help.3ds.com [5] Connector Overview (ABAQUS/Standard) (https://help.3ds.com/2017/English/DSSIMULIA_Established/SIMACAEELMRefMap/simaelm-c-connectoroverview.htm) [2019-01-07] [6] Connector Elements (ABAQUS/Standard) (https://help.3ds.com/2017/english/dssimulia_established/simacaeelmrefmap/simaelm-c-connectorelem.htm) [2019-01-07] [7] Connector Type Library (ABAQUS/Standard) (https://help.3ds.com/2017/English/DSSIMULIA_Established/SIMACAEELMRefMap/simaelm-m-ConnectionTypesLibrary-sb.htm) [2019-01-07] [8] JOIN (https://help.3ds.com/2017/English/DSSIMULIA_Established/SIMACAEELMRefMap/simaelm-c-connectiontypedesc-join.htm) [2019-01-07] [9] UNIVERSAL (https://help.3ds.com/2017/English/DSSIMULIA_Established/SIMACAEELMRefMap/simaelm-c-connectiontypedesc-universal.htm) [2019-01-07] [10] UJOINT (https://help.3ds.com/2017/English/DSSIMULIA_Established/SIMACAEELMRefMap/simaelm-c-connectiontypedesc-ujoint.htm) [2019-01-07] [11] General multi-point constraints (ABAQUS/Standard) (https://help.3ds.com/2017/english/DSSIMULIA_Established/SIMACAECSRefMap/simacst-c-mpc.htm) [2019-01-07] [12] Orientations (ABAQUS/Standard) (https://help.3ds.com/2017/english/dssimulia_established/simacaemodrefmap/simamod-c-orientation.htm) [2019-01-07] [13] BUSHING (https://help.3ds.com/2017/English/DSSIMULIA_Established/SIMACAEELMRefMap/simaelm-c-connectiontypedesc-bushing.htm) [2019-01-08] [14] Connector Element Behavior (ABAQUS/Standard) (https://help.3ds.com/2017/English/DSSIMULIA_Established/SIMACAEELMRefMap/simaelm-c-connectorbehavior.htm) [2019-01-08] [15] Connector Motion (ABAQUS/Standard) (https://help.3ds.com/2017/english/DSSIMULIA_Established/SIMACAEKEYRefMap/simakey-r-connectormotion.htm) [2019-01-08] [16] Conventions (ABAQUS/Standard) (https://help.3ds.com/2017/english/DSSIMULIA_Established/SIMACAEMODRefMap/simamod-c-conventions.htm) [2019-01-08] [17] Connector Element Output Variables (ABAQUS/Standard) (https://help.3ds.com/2017/english/DSSIMULIA_Established/SIMACAEOUTRefMap/simaout-c-std-wholeelementvariables.htm) [2019-01-08] Above references require account at https://help.3ds.com [18] Spherical Bearings (https://www.skf.com/us, search for “plain spherical bearing catalog” then select “SKF spherical plain bearings and rod ends” catalog, https://cdn.skfmediahub.skf.com/api/public/0901d19680154a05/pdf_preview_medium/0901d19680154a05_pdf_preview_medium.pdf, pg. 27)
	2019

TN 19-0301-01	GENERATING A TIME HISTORY FROM A FREQUENCY SPECTRUM USING NCODE GLYPHWORKS/VIBESYS Describes the process to generate a time series from a vibration specification defined in the frequency domain as a power spectral density (PSD) using nCode GlyphWorks and VibeSys. [1] International Test Operations Procedure (ITOP) 1-2-601 Laboratory Vibration Schedules (23 April 1998). DIST B (DTIC AD B238 288). (Superseded by ITOP-1-2-601 (DTIC AD B406 927), dated 2009-02-19 which was superseded by Test Operations Procedure (TOP) 1-2-601 Ground Vehicle Laboratory Vibration Schedules, 18 May 2015). The ITOP is considered inactive for U.S. purposes and should not be used by U.S. government personnel and their agents unless specifically called for in a requirement or specification document. [1a] NATO AECTP-400 - Mechanical Environmental Tests, Ed. D V1 (Nov 2019). Default Vibration Severities (Tracked Vehicles) in Annex A-26 (pg.A-36) to Annex A-33 (pg.A-43). Per C-M(2002)60 Information releasable to the Public (it shall carry no markings of any sort) see (https://nso.nato.int/nso/nsdd/main/standards, search for AECTP). [2] nCode GlyphWorks (nCode 2018.1, GlyphWorks V18.1.0 Release Build 281) (https://www.ncode.com/products/ncode-glyphworks-signal-processing-and-durability-analysis) [2019-03-01] now (https://www.hbkworld.com/en/products/software/analysis-simulation/durability/ncode-glyphworks-signal-processing-and-durability-analysis) [3] nCode GlyphWorks VibeSys (https://www.ncode.com/products/ncode-vibesys-acoustic-and-vibration-analysis-software) [2019-03-01] now (https://www.hbkworld.com/en/products/software/analysis-simulation/durability/ncode-vibesys-acoustic-and-vibration-analysis-software) [4] GlyphWorks Worked Examples (GW-WEX, 18.1.01) – pg. 360 “18 Regenerating a Time History from a Frequency Spectrum” [5] MATLAB (https://www.mathworks.com/products/matlab.html) [2019-03-04] [6] Vibration Data (Tom Irvine) (http://www.vibrationdata.com/tutorials_alt/psdsyn.pdf) [2019-03-04]
TN 19-0514-01	GENERATING SINE SWEEP/CHIRP SIGNALS Describes the formulas and calculations to create a sine sweep (chirp) signal with linear and logarithmic swept frequency rates using MATLAB and nCode GlyphWorks. [1] Chirp Function (https://en.wikipedia.org/wiki/Chirp#Definitions) [2] MATLAB (www.mathworks.com) [3] nCode GlyphWorks (www.ncode.com) now (https://www.hbkworld.com/en/products/software/analysis-simulation/durability/ncode-glyphworks-signal-processing-and-durability-analysis) [4] AECTP-400 (Ed. 3, 3 Jan 2006), Allied Environmental Conditions and Tests Publication, Mechanical Environmental Tests. (Released to the Public). Annex B – Tracked Vehicle Vibration – Guidance for Initial Test Severity. Figure B-2. Materiel in Turret Bustle Rack or Installed In Turret, pg. 38. Figure B-4. Light Vehicle – Materiel on Sponson or Installed in Hull Figure, pg. 44 [4a] NATO AECTP-400 - Mechanical Environmental Tests, Ed. D V1 (Nov 2019). Default Vibration Severities (Tracked Vehicles) in Annex A-26 (pg.A-36) to Annex A-33 (pg.A-43). Per C-M(2002)60 Information releasable to the Public (it shall carry no markings of any sort) see (https://nso.nato.int/nso/nsdd/main/standards, search for AECTP). [5] m+p VibControl Mixed Mode Testing (www.mpihome.com) [6] Octave Band (https://en.wikipedia.org/wiki/Octave_band) [7] Octave (https://en.wikipedia.org/wiki/Octave) [8] Integral b^x (http://www.math.com/tables/integrals/more/b%5Ex.htm)

TN 19-0801-01	FUNCTIONS FOR GENERATING POWER SPECTRAL DENSITY (PSD) INPUTS FOR FE-SAFE FREQUENCY BASED FATIGUE ANALYSIS Describes the functions (data reduction, PSD swept narrowband creation) and processes (3 or 1 channel PSD and LDF files) to create PSD data for use with Dassault/fe-safe fatigue analysis software. [1] Reference Internal Project 19-011 STRYKER BATTERY BOX STRUCTURAL ANALYSIS [2] TN19-0514-01 GENERATING SINE SWEEP/CHIRP SIGNALS [3] fe-safe 2017 (https://www.3ds.com/products-services/simulia/products/fe-safe/) [2019-07-31] [4] AECTP-400 (Ed. 3, 3 Jan 2006), Allied Environmental Conditions and Tests Publication, Mechanical Environmental Tests. (Released to the Public). Annex B – Tracked Vehicle Vibration – Guidance for Initial Test Severity. Figure B-2. Materiel in Turret Bustle Rack or Installed In Turret, pg. 38. Figure B-4. Light Vehicle – Materiel on Sponson or Installed in Hull Figure, pg. 44 [4a] NATO AECTP-400 - Mechanical Environmental Tests, Ed. D V1 (Nov 2019). Default Vibration Severities (Tracked Vehicles) in Annex A-26 (pg.A-36) to Annex A-33 (pg.A-43). Per C-M(2002)60 Information releasable to the Public (it shall carry no markings of any sort) see (https://nso.nato.int/nso/nsdd/main/standards, search for AECTP). [5] MATLAB (https://www.mathworks.com/) [5a] 1-D data interpolation (table lookup) interp1 (https://www.mathworks.com/help/matlab/ref/interp1.html) APPENDIX A.1 – DUPLICATE VALUE REMOVAL FUNCTIONS APPENDIX A.2 – PSD SWEEP DATA GENERATION APPENDIX A.3 – PSD AND LDF FILE GENERATION APPENDIX A.4 – EXAMPLE PSD AND LDF FILE FORMATS APPENDIX A.5 – TEST SCRIPT APPENDIX A.6 – MATLAB INTERP1 HELP [5a]
TN 19-0807-01	ESTIMATING RAYLEIGH DAMPING COEFFICIENTS FOR MODAL DAMPING Describes the calculation of Rayleigh damping coefficients for modal damping. [1] Chowdury, Dasgupta. “Computation of Rayleigh Damping Coefficients for Large Systems”. (http://ejge.com/2003/Ppr0318/Ppr0318.pdf) [2019-08-07] [2] Spears, Jensen. “Approach for Selection of Rayleigh Damping Parameters Used for Time History Analysis”. Proceedings of PVP2009, 2009 ASME Pressure Vessels and Piping Division Conference, July 26-30, 2009, Prague, Czech Republic. PVP2009-77257. [3] Ramamurti, Mithun, et. al. “Experimental determination of damping ratios at higher modes for use in modal superposition”. Journal of Engineering and Technology Research Vol. 4(6), pp. 114-128, November 2012. DOI: 10.5897/JETR12.015. [4] Schwarz, Richardson. “Proportional Damping from Experimental Data”. Sound and Vibration, June 2013. (http://www.sandv.com/downloads/1306schw.pdf). [2019-08-07] [5] MATLAB (https://www.mathworks.com/) [2019-08-01] [6] Munson, Kurt. HBM Prencia. “Improving the Accuracy of Dynamic Vibration Fatigue Simulation” (https://www.ncode.com/images/Resources/Proceedings/2016/2016nCodeUGM_ImprovingtheAccuracyofDynamicVibrationFatigueSimulation.pdf) [2019-08-05]
	2021
TN 21-0308-01	MODIFIED GOODMAN DIAGRAM FROM ULTIMATE, YIELD, AND ENDURANCE STRENGTH (SU, SY, AND SE) Describes how to create/plot a modified Goodman infinite fatigue life diagram based on material strength properties. [1] Shigley, et. al. Mechanical Engineering Design, 7th Ed. McGraw-Hill (2004). ISBN 0-07-252036-1. (7-12 Fatigue Failure Criteria for Fluctuating Stress, Figure 7-24, pg. 346) [2] Fatigue limit diagram according to Haigh and Smith (https://www.tec-science.com/material-science/material-testing/fatigue-limit-diagram-according-to-haigh-and-smith-creation/), see also Fatigue Limit Diagram Haigh & Smith at https://www.youtube.com/watch?v=pMubK2DTJR8) [3] PowerShell (https://docs.microsoft.com/en-us/powershell/) [4] MATLAB (https://www.mathworks.com/products/matlab.html) [5] Blodgett, Design of Welded Structures, The James F. Lincoln Arc Welding Foundation (1966). Section 2.9 Designing for Fatigue Loads, pg. 2.9-2 Fig. 3

TN 21-0317-01	TRACK PATTERN FREQUENCIES Describes calculations needed to determine margin of safety for fastener preload which includes joint stiffness and fraction of external load carried by the fastener. [1] MIL-STD-810H ENVIRONMENTAL ENGINEERING CONSIDERATIONS AND LABORATORY TESTS, Method 514.8 Vibration, Annex F, 5.2 Platform Specific Considerations – Road Transport – Tracked Vehicle. DIST A. Approved for public release: distribution unlimited. (https://quicksearch.dla.mil/qsDocDetails.aspx?ident_number=35978) [2] NATO AECP-240 (Ed 1) MECHANICAL CONDITIONS, Edition 1, July 2009. Per C-M(2002)60 Information releasable to the Public (it shall carry no markings of any sort) see (https://nso.nato.int/nso/nsdd/main/standards, search for AECP). [3] nCode 2020 (https://www.ncode.com/products/ncode-glyphworks-signal-processing-and-durability-analysis) now (https://www.hbkworld.com/en/products/software/analysis-simulation/durability/ncode-glyphworks-signal-processing-and-durability-analysis)
TN 21-0331-01	RANDOM RESPONSE OUTPUT USING TRANSMISSIBILITY Describes the verification of calculating a random response using transmissibility obtained from using white noise input. [1] TN18-0801-01 RECOVERABLE ASSET (RA) MODAL AND RANDOM RESPONSE ANALYSIS FOR SQUAD AND CREW CONFIGURATIONS (18-008) (see TR29932 - 2018 Technical Notes and Problem Solutions in Structures. DIST C (DTIC AD 1080 278)). [2] ABAQUS/Standard 2019 (https://www.3ds.com/products-services/simulia/products/abaqus/) [2019-01-07] Below references require account at https://help.3ds.com [2a] Random Response Analysis (https://help.3ds.com/2019x/english/dsdoc/sima3dxanlrefmap/simaanl-c-randomresponse.htm?contextscope=all) [2b] *RANDOM RESPONSE (https://help.3ds.com/2019x/english/dsdoc/sima3dxkeyrefmap/simakey-r-randomresponse.htm?contextscope=all) [2c] *PSD-DEFINITION (https://help.3ds.com/2019x/english/DSDoc/SIMA3DXKEYRefMap/simakey-r-psd-definition.htm?ContextScope=all) Above references require account at https://help.3ds.com [3] White Noise (https://en.wikipedia.org/wiki/White_noise) [4] Transmissibility (https://en.wikipedia.org/wiki/Transmissibility_(vibration))
TN 21-0406-01	ABAQUS/STANDARD MODAL ANALYSIS EFFECTIVE MASS AS A PERCENTAGE OF TOTAL EFFECTIVE MASS Describes the processing of the Abaqus .dat file to obtain percentage of total effective mass for each mode when doing modal analysis. [1] Abaqus/Standard (https://www.3ds.com/products-services/simulia/products/abaqus/abaqusstandard/) Below references require account at https://help.3ds.com [2] About Output (The data file) (https://help.3ds.com/2019/English/DSSIMULIA_Established/SIMACAEOUTRefMap/simaout-c-output.htm?ContextScope=all&id=36fe80847d51418b80d3772f0e69b20a#Pg0) Above references require account at https://help.3ds.com [3] Powershell. (https://docs.microsoft.com/en-us/powershell/)

TN 21-0407-01	SENSITIVITY OF FATIGUE LIFE TO CHANGES IN STRESS LEVELS Describes the analysis of fatigue life sensitivity to stress levels. [1] "What is a SN-Curve?" (https://community.sw.siemens.com/s/article/what-is-a-sn-curve) [2] Shigley, et. al. Mechanical Engineering Design, 7th Ed. McGraw-Hill (2004). ISBN 0-07-252036-1 [a] Common curve fit, pp. 326. [3] Negative Binomial Series (https://mathworld.wolfram.com/NegativeBinomialSeries.html) [4] Binomial Series (https://mathworld.wolfram.com/BinomialSeries.html) [5] Generalized Binomial Theorem (https://en.wikipedia.org/wiki/Binomial_theorem#Newton's_generalized_binomial_theorem) Below references require account at https://help.3ds.com [6] fe-safe Users Guide (2019). Dassault Systemes. (https://help.3ds.com/2019/English/DSSIMULIA_Established/SIMULIA_Established_FrontmatterMap/sim-c-DSDocFesafe.htm?contextscope=all&id=9d30c899d9f8445fa2ee4c92e61118ba). [a] Technical Note TN-001: Approximating materials data (PDF page 435). [7] fe-safe Fatigue Theory Reference (2019). Dassault Systems. (https://help.3ds.com/2019/English/DSSIMULIA_Established/FesafeFatigueTheoryPdf/FesafeFatigueTheory.pdf?contextscope=all&id=0f94ff6de3084b0086f42fa1ff90d767). [a] Figure 3.2 Stress-life relationship, pg. 3-2 (PDF page 64). [b] 3.5 Approximations for other materials, pg. 3-8 (PDF page 70) [c] fatigue strength exponent, b, usually $(-0.12 < b < -0.05)$. pg. 3-9 (PDF page 71)

TN 21-0723-01	MARGIN OF SAFETY CRITERIA FOR FASTENER PRELOAD/CLAMP FORCE REQUIREMENT Describes calculations needed to determine margin of safety for fastener preload which includes joint stiffness and fraction of external load carried by the fastener. [1] Requirements for Threaded Fastening Systems in Spaceflight Hardware. NASA-STD-5020 Rev A (2019-02-11). DIST A. Approved for public release: distribution unlimited. (https://standards.nasa.gov/standard/nasa/nasa-std-5020) [a] access slip individually at each fastener, Eq. 86, pg. 74. [2] Chambers. Preloaded Joint Analysis Methodology for Space Flight Systems. NASA Technical Memorandum 106943 (Dec 1995). (https://ntrs.nasa.gov/citations/19960012183) [3] Barrett. Fastener Design Manual. NASA Reference Publication 1228 (March 1990). (https://ntrs.nasa.gov/citations/19900009424) [4] Ubrako Engineering Guide (https://unbrako.com/images/downloads/engguide.pdf) [a] Joint Diagrams, pp. 56 - 61 [5] Shigley, et. al. Mechanical Engineering Design, 7th Ed. McGraw-Hill (2004). ISBN 0-07-252036-1. [6] Structural Design and Test Factors of Safety for Spaceflight Hardware. NASA-STD-5001B (2016-04-04) (https://standards.nasa.gov/sites/default/files/nasa-std-5001b_w_change_1.pdf) [a] Margin of Safety (MS), pg. 10 [7] NASA Engineering and Safety Center Technical Bulletin No. 12-02 Structural Analyses and Margins of Safety. (https://www.nasa.gov/sites/default/files/files/671748main_NESCTB12-02_StructuralAnalysesandMarginsofSafety.pdf) [8] Raju, Lee, and Mohaghegh: "Negative Stress Margins - Are They Real?", AIAA-2011-1808-588, Paper Presented at the 52nd AIAA SDM conference, Denver, Colorado, April 4-7, 2011. (https://ntrs.nasa.gov/api/citations/20110010003/downloads/20110010003.pdf) [9] ASME B1.1-2019 Unified Inch Screw Threads (Un, UNR, and UNJ Thread Forms) [a] Table 2A - Limits of Size for Standard Series External Thread (UN, UNR, and UNJ) [b] Table 2B - Limits of Size for Standard Series Internal Threads (UN and UNJ) [c] Appendix B - Thread Strength Design Formulas, pg. 150 [10] Machinery's Handbook, 31st Ed. Industrial Press (2020). ISBN 978-0-8311-3631-4 [11] Abbott, Richard. Analysis and Design of Composite and Metallic Flight Vehicle Structures 3rd Edition, 2019 (https://www.abbottaerospace.com/downloads/analysis-and-design/) [12] Bolt Grade Markings and Strength Chart (https://www.boltdepot.com/fastener-information/materials-and-grades/bolt-grade-chart.aspx) [13] Fastenal Torque-Tension Reference Guide (https://www.fastenal.com/content/merch_rules/images/fcom/content-library/Torque-Tension%20Reference%20Guide.pdf) [14] Proof Load Tensile Strength for Grade 2,5, and 8 (https://www.almabolt.com/pages/catalog/bolts/prooffloadtensile.htm) [15] Bolt Clamping Force vs. Tightening Torque for Unlubricated Steel Bolts (https://sabreindustrial.com/content/Tech%20PDFS/003%20Bolt%20Clamp%20Load%20Chart.pdf) [16] Powershell (https://docs.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7) [2020-10-14] [17] .NET Objects List, Dictionary (https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1?view=netcore-3.1, https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.dictionary-2?view=netcore-3.1)
	2022
TN 22-0427-01	PART SELECTION BASED ON YIELD STRENGTH Describes process to reduce number of parts to be analyzed based on material yield strength. [1] Altair HyperView (https://www.altair.com/hyperview/) Below references require account at https://help.3ds.com [2] Dassault Systems SIMULIA Abaqus/Standard (https://www.3ds.com/products-services/simulia/products/abaqus/abaqusstandard/) Above references require account at https://help.3ds.com [3] Powershell (https://docs.microsoft.com/en-us/powershell/)

TN 22-0524-01	ALTAIR HYPERWORKS LICENSE USAGE MONITORING Describes process for monitoring Altair HyperWorks license usage based on the LM-X License Server (includes extracting features and user data using regular expressions, scripts to analyze and plot usage over time) [1] Altair Units 2020 Licensing Guide (almutil) (https://www.altair.com) [2] PowerShell (https://docs.microsoft.com/en-us/powershell/) [3] .NET regular expressions (https://docs.microsoft.com/en-us/dotnet/standard/base-types/regular-expressions) [4] MATLAB (https://www.mathworks.com/products/matlab.html)
TN 22-1101-01	POWERSHELL SCRIPT AUTO STARTUP Describes process for creating auto startup PowerShell scripts. [1] Command Interpreter - CMD.exe (https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/cmd) PowerShell (https://learn.microsoft.com/en-us/powershell/) [2] Powershell Command Line Options (https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_powershell_exe?view=powershell-5.1) [3] PowerShell (https://learn.microsoft.com/en-us/powershell/) [4] PowerShell Jobs (https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_jobs?view=powershell-7.2) [5] TN22-0524-01 ALTAIR HYPERWORKS LICENSE USAGE MONITORING
TN 22-1103-01	ALTAIR HYPERWORKS LICENSE ACTIVITY MATRIX Describes process to determine user license activity (in use for each sample). [1] Altair Units 2020 Licensing Guide (almutil) (https://www.altair.com) [2] TN22-0524-01 ALTAIR HYPERWORKS LICENSE USAGE MONITORING [3] PowerShell (https://learn.microsoft.com/en-us/powershell/)
	2023

TN 23-0516-01	ABAQUS/STANDARD TO ABAQUS/EXPLICIT CONVERSION ISSUES Describes changes needed to convert an Implicit model (Abaqus/Standard) to an Explicit model (Abaqus/Explicit). Below references require account at https://help.3ds.com [1] Dassault Systems SIMULIA Abaqus/Standard (https://www.3ds.com/products-services/simulia/products/abaqus/abaqusstandard/) [2] Dassault Systems SIMULIA Abaqus/Explicit (https://www.3ds.com/products-services/simulia/products/abaqus/abaqusexplicit/) [2a] STEPS (https://help.3ds.com/2022/English/DSSIMULIA_Established/SIMACAEKEYRefMap/simakey-r-dynamic.htm?contextscope=all) [2b] CONTACT PAIR (https://help.3ds.com/2022/English/DSSIMULIA_Established/SIMACAEKEYRefMap/simakey-r-contactpair.htm?contextscope=all&id=3aa9e54191404ba98280b0f90b5842fd) [2c] MASS SCALING (https://help.3ds.com/2022/English/DSSIMULIA_Established/SIMACAEANLRefMap/simaanl-c-massscaling.htm?contextscope=all) [2d] FILTERING OUTPUT (https://help.3ds.com/2022/English/DSSIMULIA_Established/SIMACAEOUTRefMap/simaout-c-dboutput-filter.htm?contextscope=all&id=0d2e45e45fe744379f180d396d2875f6) [2e] BULK VISCOSITY (https://help.3ds.com/2022/english/dssimulia_established/SIMACAEANLRefMap/simaanl-c-expdynamic-bulkviscosity) Above references require account at https://help.3ds.com [3] Implicit and Explicit Analysis in FEA, Part - 01 (Differences & Application), SimTech05. (https://www.youtube.com/watch?v=UMer6RkLpww) [4] Leonhardt. Acceleration Levels of Dropped Objects (Technical Paper 321), Endevco (https://endevco.com/contentStore/mktgContent/endevco/dlm_uploads/2019/02/TP321.pdf) [5] Revisions and Exceptions to SAE ARP 1967B (see SAE ARP1967 Rev B – Aerospace Recommended Practice – Containers, Shipping and Storage, Reusable at https://www.sae.org/standards/content/arp1967b/) (https://sam.gov/opp/08a491daf56f46df9ef0d75eddcaaac0/view) [6] How to Calculate the Velocity of a Pendulum Using the Law of Conservation of Energy Physics Study.com (https://study.com/skill/learn/how-to-calculate-the-velocity-of-a-pendulum-using-the-law-of-conservation-of-energy-explanation.html)
TN 23-0516-02	ABAQUS/EXPLICIT HISTORY OUTPUT EXTRACTION USING PYTHON Describes selection of history output for Abaqus/Explicit output database (.odb) using “odbreport” command and python scripting. [1] Dassault Systems SIMULIA Abaqus/Explicit (https://www.3ds.com/products-services/simulia/products/abaqus/abaqusexplicit/) [2] Python (www.python.org)

TN 23-0531-01	RAMBERG-OSGOOD: FROM NEUBER CORRECTION TO PLASTICITY Describes the Ramberg-Osgood Stress/Strain relationship. [1] Ramberg-Osgood relationship (https://en.wikipedia.org/wiki/Ramberg%E2%80%93Osgood_relationship) [2] Ramberg, Osgood. Description of stress-strain curves by three parameters. NACA-TN-902, 1 Jul 1943 (https://ntrs.nasa.gov/api/citations/19930081614/downloads/19930081614.pdf) [3] Hill, H. Determination of stress-strain relations from "offset" yield strength values. NACA-TN-927, 1 Feb 1944. (https://ntrs.nasa.gov/api/citations/19930081673/downloads/19930081673.pdf) [4] Neuber Plastic Strain Method Theory and Spreadsheet, Abbott Aerospace Canada Ltd. (https://www.abbottaerospace.com/aa-sb-001/14-ultimate-strength-of-metallic-elements/14-1-introduction/14-1-2-neuber-plastic-strain-method/, https://www.abbottaerospace.com/downloads/aa-sm-030/) [5] Neubers Rule (https://community.sw.siemens.com/s/article/Neuber-s-Rule) [6] The Strain Life Approach (https://community.sw.siemens.com/s/article/The-Strain-Life-Approach) [7] The Neuber Correction: A Notch Above the Rest for Linear Stress (https://www.altair.com/newsroom/partner-perspectives/the-neuber-correction-a-notch-above-the-rest-for-linear-stress/) [8] FEMFAT plast-module for calculating elasto-plastic stresses based on a linear elastic FE analysis (https://femfat.magna.com/index.php?id=522) [9] Nonlinear Stresses from Linear FEA (https://www.m4-engineering.com/nonlinear-stresses-from-linear-fea/) [10] Metallic Plasticity Modelling - Abaqus FEM Code (https://www.icemm.com/wp-content/uploads/woocommerce_uploads/ISM2006_Plasticity_Modelling-Abaqus-FEM-Code_v2.pdf) [11] Plasticity Corrections for Elastic Analysis Results: Neuber Method (https://community.ptc.com/sejnu66972/attachments/sejnu66972/cpanalysis/307/2/neuber_method.pdf) [12] Comparison of Static Failure Criteria of Additively Manufactured Inconel 718 Analyzed by means of Strain Energy Density (https://ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/2623223/no.ntnu%3Ainspera%3A2524701.pdf?sequence=1&isAllowed=y) [13] Ramberg-Osgood model (http://www2.me.rochester.edu/courses/ME204/nx_help/index.html#uid:id986788) [14] Numerical Validation and Application of the Neuber-Formula in FEA-Analysis (http://www.iberisa.com/productos/winlife/pdf/Neuber_FEA.pdf) [15] An Estimation of Ramberg-Osgood Constants for Materials with and without Luder's Strain Using Yield and Ultimate Strengths. ICSI 2019 The 3rd International Conference on Structural Integrity, Procedia Structural Integrity 17 (2019) 750-757 (www.elsevier.com/locate/procedia). [16] MATLAB (https://www.mathworks.com/) [17] Python (https://www.python.org) [18] JupyterLab (https://jupyter.org/try-jupyter/lab/) [19] Abaqus/Standard (https://www.3ds.com/products-services/simulia/products/abaqus/abaqusstandard/) Below references require account at https://help.3ds.com [19a] DEFORMATION PLASTICITY (https://help.3ds.com/2023/English/DSSIMULIA_Established/SIMACAEKEYRefMap/simakey-r-deformationplasticity.htm?contextscope=all&id=a27580eddaca46de9de51a65a29a3144) [19b] Inelastic Behavior (https://help.3ds.com/2023/English/DSSIMULIA_Established/SIMACAEMATRefMap/simamat-c-plastic.htm?contextscope=all&id=6c50da490b194d40b71074d25ad8b496) Above references require account at https://help.3ds.com

TN 10-0604-01

DYNAMICS SOFTWARE 7-BODY EXAMPLE COMPARISON – DADS AND RECURDYN

June 4, 2010

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC) Detroit
Arsenal
ATTN: RDTA-RS/MS157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

Recent software and hardware changes caused the Dynamics and Structures Team to evaluate other dynamic simulation software packages against its legacy Dynamic Analysis and Design System (DADS) simulation environment for possible use. RecurDyn (is Computer-Aided Engineering (CAE) software that is focused on Multi-Body Dynamics (MBD) while also offering multi-physics solutions. The core is Multi- Flexible Body Dynamics (MFBD) that combines a multibody dynamics solver and a nonlinear finite element method solver) is one of these software packages and has a history closely related to DADS due to its founder, Dr. Bae, having worked with Dr. Haug at the Department of Mechanical Engineering, Center for Computer Aided Design, The University of Iowa.

PROBLEM SOLUTION:

For output results comparison a standardized problem, found in [1] pp. 10-15, was used. A full description can be found in Appendix A.1 (from [1]).

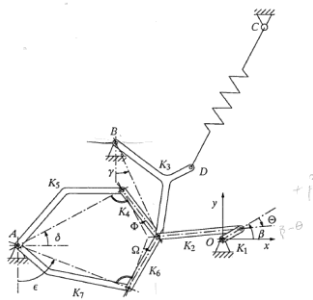
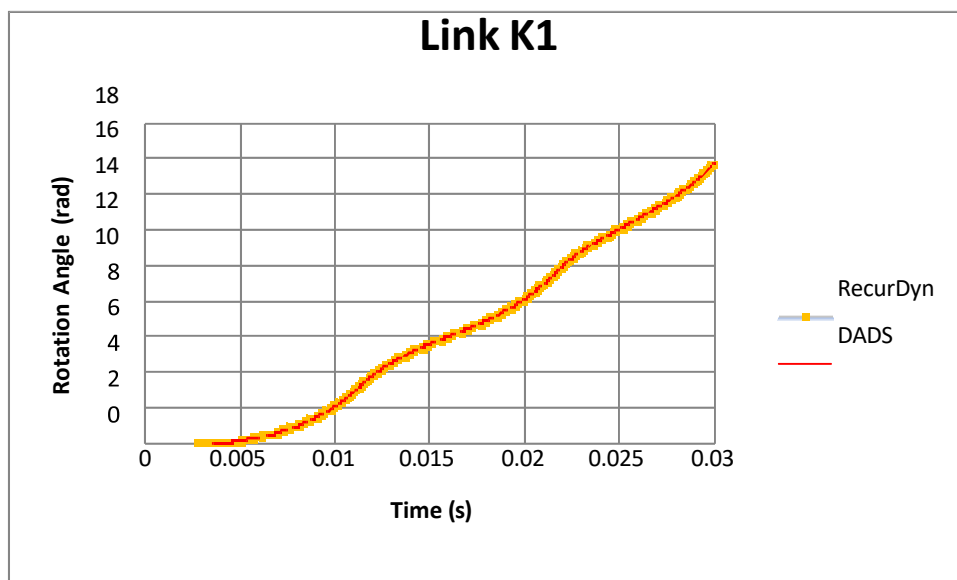


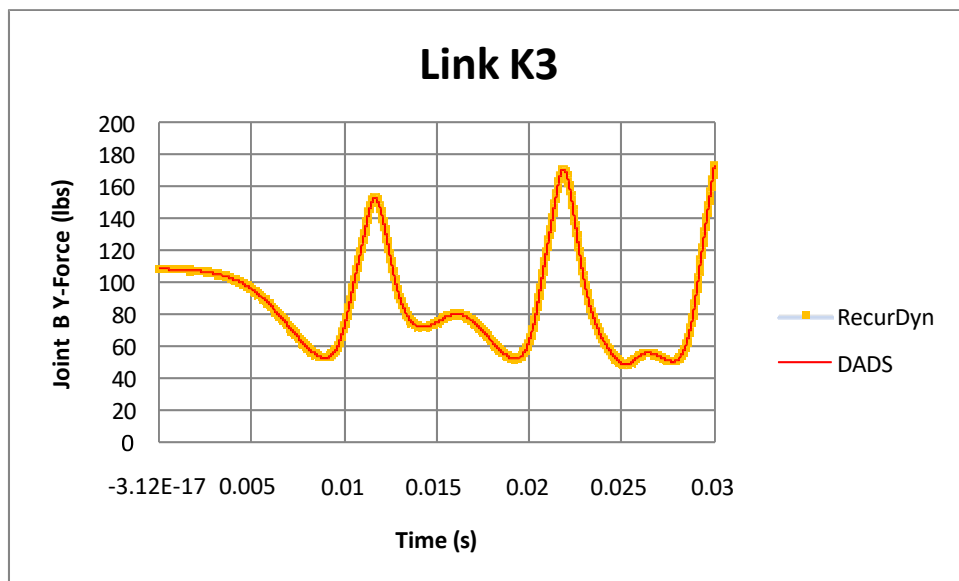
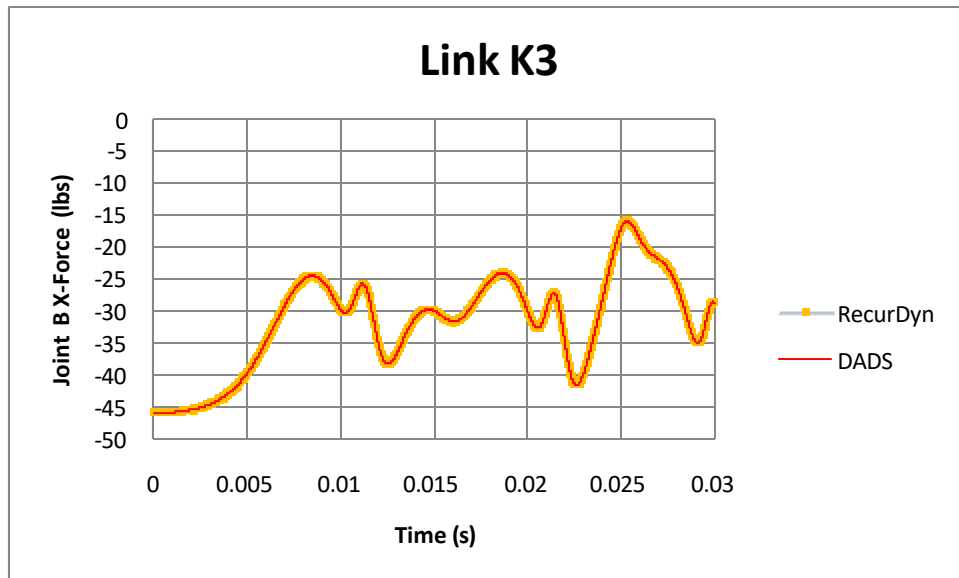
Fig. 1. Seven body mechanism

The problem consists of seven bodies interconnected by joints without friction (see Figure 1 above). A spreadsheet summary of the various positions for center-of-gravity and attachments points is shown below for easy data entry (see Appendix A.1 for original definitions).

[illegible]

A RecurDyn model of Figure 1 was built to compare to a previous DADS model. The RecurDyn Solver File (.RMD) is included in Appendix A.2. Selected output results are shown below.





Analysis of the results use the following MATLAB script to process the output data for DADS and RecurDyn.

```

---BOF---
y=dlmread('linkage_results_dads.dat','\t',5,0); t=y(:,1);
ang=y(:,2);
fm=y(:,8);
fx=y(:,9);
fy=y(:,10);

y1=dlmread('cDD','\t',1,0); t1=y1(:,1);
ang1=y1(:,2);
fx1=y1(:,3);
fy1=y1(:,4);
fm1=y1(:,5);
    
```

```

% for IMGALPH integrator y2=dlmread('cII','',1,0); t2=y1(:,1);
ang2=y1(:,2);
fx2=y1(:,3);
fy2=y1(:,4);
fm2=y1(:,5);

figure(1) clf
plot(t,ang,'r') hold plot(t1,ang1,'k--')
plot(t2,ang2,'g-.')
legend('DADS','DDASSL','RDYN');

figure(2) clf
plot(t,fx,'r') hold on plot(t1,fx1,'k--')
plot(t2,fx2,'g-.')
legend('DADS','DDASSL','RDYN');

figure(3) clf
plot(t,fy,'r') hold on plot(t1,fy1,'k--')
plot(t2,fy2,'g-.')
legend('DADS','DDASSL','RDYN');
---EOF---

```

A comparison of the results shows them to be essentially identical (based on RecurDyn's DDASSL integrator).

REFERENCE

- [1] Schiehlen (Editor), Multibody Systems Handbook, Springer-Verlag, ISBN 0-387-51946-7. See also (https://books.google.com/books/about/Multibody_Systems_Handbook.html?id=UYz_CAAAQBAJ) [2024-04-15]
- [2] Reference Internal Project 10-080 Dynamics Software 7-body Example Comparison - DADS VS RECURDYN
- [3] Library of Computational Benchmark Problems, IFToMM Technical Committee for Multibody Dynamics (<https://www.iftomm-multibody.org/benchmark/>) See Andrews Squeezer Mechanism.
- [4] Tagliapietra, et al. MultiBody Systems Benchmark in OpenSIM (MBS-BOS) - User Manual Release 2.0 (Sep 1, 2015). Ch. 3 - MBS Benchmark A03: Andrew's Squeezer Mechanism. (<https://simtk.org/docman/view.php/984/1910/MBS-BOSUserGuide.pdf>)
- [5] Schiehlen, Valasek. Virtual Nonlinear Multibody Systems-NATO Advanced Study Institute. Vol 1. Prague, 23 Jun - 3 Jul 2002. DIST A (DTIC AD A409 218). Approved for public release: distribution unlimited. (pp. 144-149)

APPENDIX

A.1 – Test Example – Seven Body Mechanism

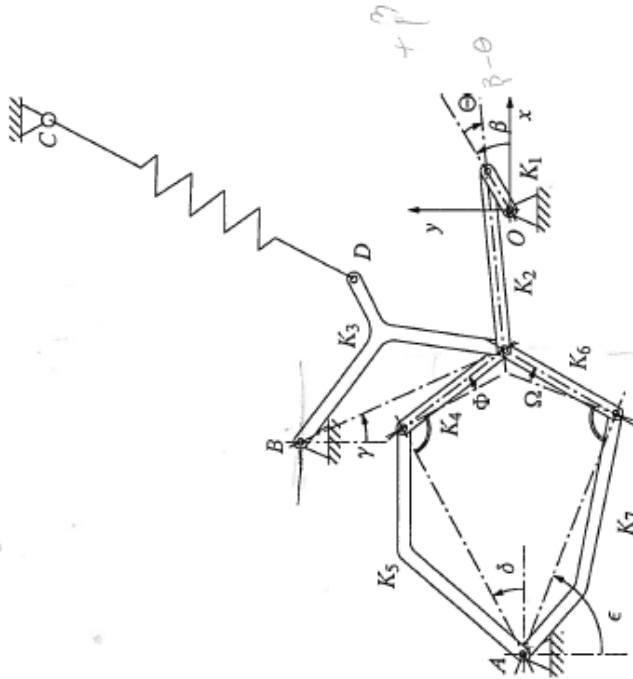


Fig. 1. Seven body mechanism

Test Examples

This chapter presents two mechanical systems which are appropriate as test examples for multibody systems. The plane motion of the mechanism with one degree of freedom was firstly used by Giles [1] and Manning [2] for testing. Later, Homolka [3] and Leister [4] analysed the mechanism using the programs NEWUL and ADAMS, respectively. The topology of the mechanism is characterized by closed kinematic loops. The second example is the spatial motion of a robot with five degrees of freedom. The topology of such a mechanical system shows chain configuration.

Since every multibody formalism presents the time history of system variables in different format, the results of a simulation of motion are often not comparable. In contrary, the visualisation of a simulation by computer graphics animation is a general mean for the system analyst to identify the results easily. For this purpose only an animation of both the examples will be presented. A detailed collection of corresponding simulations is available in Ref. [5].

Description of the Mechanism

The mechanism shown in Fig. 1 consists of 7 bodies interconnected by joints without friction. The cartesian coordinates of the points A, B, C, and all the other parameters were adopted from Giles [1].

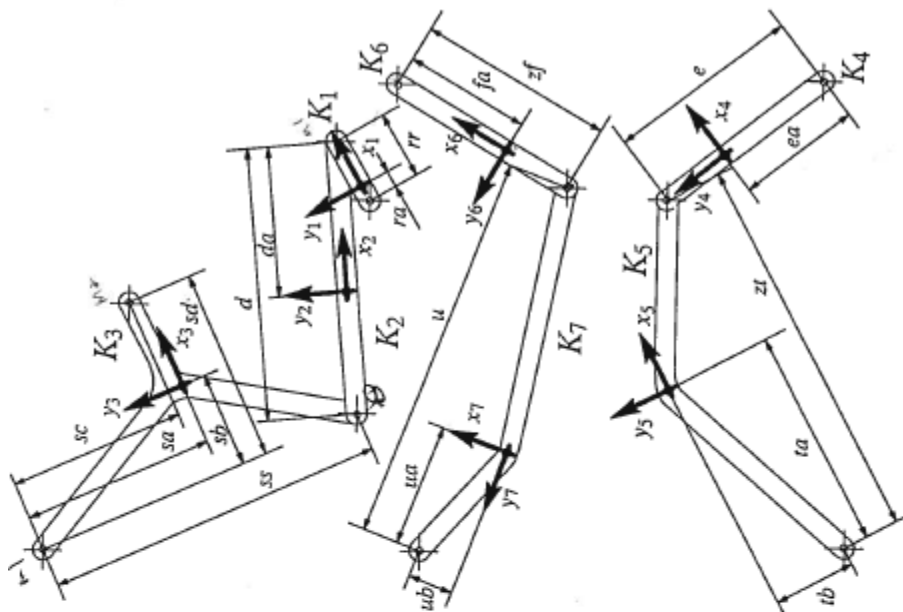


Fig. 2 Geometrical design

The coordinates of the points A, B, C read as

$$\begin{aligned}x_A &= -0.06934 \text{ m}, & y_A &= -0.00227 \text{ m}, \\x_B &= -0.03615 \text{ m}, & y_B &= 0.02273 \text{ m}, \\x_C &= 0.01400 \text{ m}, & y_C &= 0.07200 \text{ m}.\end{aligned}$$

The geometrical and inertia parameters of all bodies are given in Fig. 2 and Table 1. Further, for the spring yields:

$$\begin{aligned}\text{spring coefficient } c_0 &= 4530 \text{ N/m}, \\ \text{unstretched length } l_0 &= 0.07785 \text{ m}.\end{aligned}$$

The body-fixed frames $x_i, y_i, i=1(1)7$, are located in the center of mass of each body. The mechanism is driven by a motor located in point O. The constant drive torque is given by $MOM = 0.033 \text{ Nm}$.

The motion of the mechanism is to be simulated for one revolution. At time zero all the velocities of the mechanism are vanishing and the coordinates satisfying the constraints are given as

$$\begin{aligned}\beta_0 &= -0.0620 \text{ rad}, & \theta_0 &= 0.0000 \text{ rad}, \\ \gamma_0 &= 0.4552 \text{ rad}, & \phi_0 &= 0.2227 \text{ rad}, \\ \delta_0 &= 0.4873 \text{ rad}, & \psi_0 &= 0.2227 \text{ rad}, \\ \epsilon_0 &= 1.2305 \text{ rad}, & & \end{aligned}$$

Each of these coordinates can be chosen as generalized coordinate. An adequate choice is the angle β of the first body.

Fig. 3 presents the animation of motion of the mechanism for an interval of 15 ms with constant steps of 1 ms. Within the interval 0 to 15 ms link 1 completes just one revolution.

14

Table 1 Geometrical and inertia parameters

d	= 0,028	m	sb	= 0,01043	m
da	= 0,0115	m	sc	= 0,018	m
e	= 0,02	m	sd	= 0,02	m
ea	= 0,01421	m	zt	= 0,04	m
zf	= 0,02	m	ta	= 0,02308	m
fa	= 0,01421	m	tb	= 0,00916	m
fr	= 0,007	m	u	= 0,04	m
ra	= 0,00092	m	ua	= 0,01228	m
ss	= 0,035	m	ub	= 0,00449	m
sa	= 0,01874	m			

Link	Mass	Rotational Inertia about centers of mass
	M1 to M7 [kg]	I1 to I7 [kg·m ²]
1	0,04325	2,194 · 10 ⁻⁶
2	0,00365	4,410 · 10 ⁻⁷
3	0,02373	5,255 · 10 ⁻⁶
4	0,00706	5,667 · 10 ⁻⁷
5	0,07050	1,169 · 10 ⁻⁵
6	0,00706	5,667 · 10 ⁻⁷
7	0,05498	1,912 · 10 ⁻⁵

A.2 – RecurDyn Solver File (.RMD)

```

!===== MODEL DATA FILE GENERATED BY RECURDYN MODELER =====
!
!===== Ground =====
!
!
!           RecurDyn_Data_Name='Ground' PART / 1 , GROUND
!           RecurDyn_Data_Name='InertiaMarker' MARKER / 1
, PART = 1
, QP = 0.000000 , 0.000000 , 0.000000
, REULER = 0.000000D , 0.000000D , 0.000000D

!
!           RecurDyn_Data_Name='Xa'
MARKER / 2
, PART = 1
, QP = -0.069340 , -0.002270 , 0.000000
, REULER = 0 , 0 , 0

!
!           RecurDyn_Data_Name='Xb'
MARKER / 3
, PART = 1
, QP = -0.036360 , 0.032730 , 0.000000
, REULER = 0 , 0 , 0

!
!           RecurDyn_Data_Name='Xc'
MARKER / 4
, PART = 1
, QP = 0.014000 , 0.072000 , 0.000000
, REULER = 0 , 0 , 0

!
!           RecurDyn_Data_Name='Marker1' MARKER / 5
, PART = 1
, QP = 0.000000 , 0.000000 , 0.000000
, REULER = 0 , 0 , 0

!
!           RecurDyn_Data_Name='Marker2' MARKER / 6
, PART = 1
, QP = 0.006987 , -0.000434 , 0.000000
, REULER = 0 , 0 , 0

!
!           RecurDyn_Data_Name='Marker3' MARKER / 7
, PART = 1
, QP = -0.036347 , 0.032737 , 0.000000
, REULER = 0 , 0 , 0

!
!           RecurDyn_Data_Name='Marker4' MARKER / 8
, PART = 1
, QP = -0.069340 , -0.002270 , 0.000000
, REULER = 0 , 0 , 0

!
!           RecurDyn_Data_Name='Marker5' MARKER / 9
, PART = 1
, QP = -0.069340 , -0.002270 , 0.000000
, REULER = 0 , 0 , 0

```

```

!                      RecurDyn_Data_Name='Marker6' MARKER / 10
, PART = 1
, QP = 0.014000 , 0.072000 , 0.000000
, REULER = 0 , 0 , 0

!                      RecurDyn_Data_Name='Marker8' MARKER / 11
, PART = 1
, QP = 0.000000 , 0.000000 , 0.000000
, REULER = 0 , 0 , 0

!                      RecurDyn_Data_Name='Marker9' MARKER / 12
, PART = 1
, QP = 0.000000 , 0.000000 , 0.000000
, REULER = 0 , 0 , 0

!                      RecurDyn_Data_Name='Marker7' MARKER / 13
, PART = 1
, QP = 0.014000 , 0.072000 , 0.000000
, REULER = 0 , 0 , 0

!=====
DummyBody@SevenBodyMech2=====
!
!                      RecurDyn_Data_Name='MotherBody'

!===== Body =====
!
!                      RecurDyn_Data_Name='K3'
PART / 3
, MASS = 0.02373
, CM = 15
, IP = 1 , 1 , 5.255e-006
, 0 , 0 , 0
, QG = -0.020960 , 0.001301 , 0.000000
, REULER = 0.4552000203 , 0 , 0

!                      RecurDyn_Data_Name='CM'
MARKER / 15
, PART = 3
, QP = 0.010430 , 0.016260 , 0.000000
, REULER = 0 , 0 , 0

!                      RecurDyn_Data_Name='M1'
MARKER / 16
, PART = 3
, QP = 0.000000 , 0.035000 , 0.000000
, REULER = 0 , 0 , 0

!                      RecurDyn_Data_Name='M2'
MARKER / 17
, PART = 3
, QP = 0.020000 , 0.017000 , 0.000000
, REULER = 0 , 0 , 0

!                      RecurDyn_Data_Name='M3'
MARKER / 18
, PART = 3

```

```

, QP = 0.010430 , 0.017000 , 0.000000
, REULER = 0 , 0 , 0

!                      RecurDyn_Data_Name='zerp' MARKER / 19
, PART = 3
, QP = 0.000000 , 0.000000 , 0.000000
, REULER = 0 , 0 , 0

!                      RecurDyn_Data_Name='Marker1' MARKER / 20
, PART = 3
, QP = 0.000000 , -0.000000 , 0.000000
, REULER = -0.4552000203 , 0 , 0

!                      RecurDyn_Data_Name='Marker2' MARKER / 21
, PART = 3
, QP = 0.024338 , -0.013845 , 0.000000
, REULER = -0.4552000203 , 0 , 0

!                      RecurDyn_Data_Name='Marker3' MARKER / 22
, PART = 3
, QP = 0.000000 , 0.035000 , 0.000000
, REULER = -0.4552000203 , 0 , 0

!                      RecurDyn_Data_Name='Marker4' MARKER / 23
, PART = 3
, QP = 0.020000 , 0.016300 , 0.000050
, REULER = -0.4552000203 , 0 , 0

!                      RecurDyn_Data_Name='Marker5' MARKER / 24
, PART = 3
, QP = 0.020000 , 0.017000 , 0.000000
, REULER = -0.4552000203 , 0 , 0

!===== Body =====
!
!                      RecurDyn_Data_Name='K1'
PART / 4
, MASS = 0.04325
, CM = 25
, IP = 1 , 1 , 2.194e-006
, 0 , 0 , 0
, QG = 0.000000 , 0.000000 , 0.000000
, REULER = -0.06199999424 , 0 , 0

!                      RecurDyn_Data_Name='CM'
MARKER / 25
, PART = 4
, QP = 0.000920 , 0.000000 , 0.000000
, REULER = 0 , 0 , 0

!                      RecurDyn_Data_Name='M1'
MARKER / 26
, PART = 4
, QP = 0.007000 , 0.000000 , 0.000000
, REULER = 0 , 0 , 0

```

```

!                      RecurDyn_Data_Name='Marker1' MARKER / 27
, PART = 4
, QP = 0.000000 , 0.000000 , 0.000000
, REULER = 0.06199999424 , 0 , 0

!                      RecurDyn_Data_Name='Marker2' MARKER / 28
, PART = 4
, QP = 0.007000 , 0.000000 , 0.000000
, REULER = 0.06199999424 , 0 , 0

!                      RecurDyn_Data_Name='Marker4' MARKER / 29
, PART = 4
, QP = 0.000000 , 0.000000 , 0.000000
, REULER = 0.06199999424 , 0 , 0

!                      RecurDyn_Data_Name='Marker5' MARKER / 30
, PART = 4
, QP = 0.000000 , 0.000000 , 0.000000
, REULER = 0 , 0 , 0

!===== Body =====
!
!                      RecurDyn_Data_Name='K2'
PART / 5
, MASS = 0.00365
, CM = 31
, IP = 1 , 1 , 4.41e-007
, 0 , 0 , 0
, QG = 0.000000 , 0.000000 , 0.000000
, REULER = -0.06199999424 , 0 , 0

!                      RecurDyn_Data_Name='CM'
MARKER / 31
, PART = 5
, QP = -0.004500 , 0.000000 , 0.000000
, REULER = 0 , 0 , 0

!                      RecurDyn_Data_Name='M1'
MARKER / 32
, PART = 5
, QP = 0.007000 , 0.000000 , 0.000000
, REULER = 0 , 0 , 0

!                      RecurDyn_Data_Name='zero' MARKER / 33
, PART = 5
, QP = -0.021000 , 0.000000 , 0.000000
, REULER = 0 , 0 , 0

!                      RecurDyn_Data_Name='Marker2' MARKER / 34
, PART = 5
, QP = 0.007000 , 0.000000 , 0.000000
, REULER = 0.06199999424 , 0 , 0

!                      RecurDyn_Data_Name='Marker3' MARKER / 35

```

```

, PART = 5
, QP = -0.021000 , 0.000000 , 0.000000
, REULER = 0.06199999424 , 0 , 0

!
!                      RecurDyn_Data_Name='Marker1' MARKER / 36
, PART = 5
, QP = -0.021000 , 0.000000 , 0.000000
, REULER = 0.06199999424 , 0 , 0

!
!                      RecurDyn_Data_Name='Marker4' MARKER / 37
, PART = 5
, QP = -0.021000 , 0.000000 , 0.000000
, REULER = 0.06199999424 , 0 , 0

!===== Body =====
!
!                      RecurDyn_Data_Name='K4'
PART / 6
, MASS = 0.00706
, CM = 38
, IP = 1 , 1 , 5.667e-007
, 0 , 0 , 0
, QG = -0.020960 , 0.001301 , 0.000000
, REULER = 0.7099999397 , 0 , 0

!
!                      RecurDyn_Data_Name='CM'
MARKER / 38
, PART = 6
, QP = 0.000000 , 0.014210 , 0.000000
, REULER = 0 , 0 , 0

!
!                      RecurDyn_Data_Name='M1'
MARKER / 39
, PART = 6
, QP = 0.000000 , 0.020000 , 0.000000
, REULER = 0 , 0 , 0

!
!                      RecurDyn_Data_Name='zero' MARKER / 40
, PART = 6
, QP = 0.000000 , 0.000000 , 0.000000
, REULER = 0 , 0 , 0

!
!                      RecurDyn_Data_Name='Marker1' MARKER / 41
, PART = 6
, QP = 0.000000 , -0.000000 , 0.000000
, REULER = -0.7099999397 , 0 , 0

!
!                      RecurDyn_Data_Name='Marker2' MARKER / 42
, PART = 6
, QP = 0.000000 , 0.020000 , 0.000000
, REULER = -0.7099999397 , 0 , 0

!===== Body =====
!
!                      RecurDyn_Data_Name='K5'
PART / 7
, MASS = 0.0705

```

```
, CM = 43
, IP = 1 , 1 , 1.169e-005
, 0 , 0 , 0
, QG = -0.069340 , -0.002270 , 0.000000
, REULER = 0.4872999414 , 0 , 0
```

```
!                      RecurDyn_Data_Name='CM'
MARKER / 43
, PART = 7
, QP = 0.023080 , 0.009160 , 0.000000
, REULER = 0 , 0 , 0
```

```
!                      RecurDyn_Data_Name='M1'
MARKER / 44
, PART = 7
, QP = 0.040000 , 0.000000 , 0.000000
, REULER = 0 , 0 , 0
```

```
!                      RecurDyn_Data_Name='cg'
MARKER / 45
, PART = 7
, QP = 0.023080 , 0.009160 , 0.000000
, REULER = 0 , 0 , 0
```

```
!                      RecurDyn_Data_Name='zero' MARKER / 46
, PART = 7
, QP = 0.000000 , 0.000000 , 0.000000
, REULER = 0 , 0 , 0
```

```
!                      RecurDyn_Data_Name='Marker1' MARKER / 47
, PART = 7
, QP = 0.040004 , 0.000008 , 0.000000
, REULER = -0.4872999414 , 0 , 0
```

```
!                      RecurDyn_Data_Name='Marker2' MARKER / 48
, PART = 7
, QP = 0.000000 , 0.000000 , 0.000000
, REULER = -0.4872999414 , 0 , 0
```

```
!===== Body =====
```

```
!
!                      RecurDyn_Data_Name='K6'
PART / 8
, MASS = 0.00706
, CM = 49
, IP = 1 , 1 , 5.667e-007
, 0 , 0 , 0
, QG = -0.031634 , -0.015621 , 0.000000
, REULER = 1.007800059 , 0 , 0
```

```
!                      RecurDyn_Data_Name='CM'
MARKER / 49
, PART = 8
, QP = 0.005790 , 0.000000 , 0.000000
, REULER = 0 , 0 , 0
```

```
!                      RecurDyn_Data_Name='M1'
MARKER / 50
, PART = 8
```

```
, QP = 0.020000 , 0.000000 , 0.000000
, REULER = 0 , 0 , 0
```

```
!                      RecurDyn_Data_Name='zero' MARKER / 51
, PART = 8
, QP = 0.000000 , 0.000000 , 0.000000
, REULER = 0 , 0 , 0
```

```
!                      RecurDyn_Data_Name='Marker1' MARKER / 52
, PART = 8
, QP = 0.020007 , 0.000005 , 0.000000
, REULER = -1.007800059 , 0 , 0
```

```
!                      RecurDyn_Data_Name='Marker2' MARKER / 53
, PART = 8
, QP = 0.000000 , 0.000000 , 0.000000
, REULER = -1.007800059 , 0 , 0
```

```
!===== Body =====
```

```
!
!                      RecurDyn_Data_Name='K7'
PART / 9
, MASS = 0.05498
, CM = 54
, IP = 1 , 1 , 1.912e-005
, 0 , 0 , 0
, QG = -0.031634 , -0.015621 , 0.000000
, REULER = 1.230500058 , 0 , 0
```

```
!                      RecurDyn_Data_Name='CM'
MARKER / 54
, PART = 9
, QP = -0.004490 , 0.027720 , 0.000000
, REULER = 0 , 0 , 0
```

```
!                      RecurDyn_Data_Name='M1'
MARKER / 55
, PART = 9
, QP = 0.000000 , 0.040000 , 0.000000
, REULER = 0 , 0 , 0
```

```
!                      RecurDyn_Data_Name='cg'
MARKER / 56
, PART = 9
, QP = -0.004490 , 0.027720 , 0.000000
, REULER = 0 , 0 , 0
```

```
!                      RecurDyn_Data_Name='zero' MARKER / 57
, PART = 9
, QP = 0.000000 , 0.000000 , 0.000000
, REULER = 0 , 0 , 0
```

```
!                      RecurDyn_Data_Name='Marker1' MARKER / 58
, PART = 9
, QP = 0.000000 , 0.040000 , 0.000000
, REULER = -1.230500058 , 0 , 0
```

```

!                               RecurDyn_Data_Name='Marker2' MARKER / 59
, PART = 9
, QP = 0.000000 , 0.000000 , 0.000000
, REULER = -1.230500058 , 0 , 0

```

```

!===== Joint =====
!
!                               RecurDyn_Data_Name='RevJoint1' JOINT / 1
, I = 27
, J = 5
, Revolute

```

```

!===== Joint =====
!
!                               RecurDyn_Data_Name='RevJoint2' JOINT / 2
, I = 34
, J = 28
, Revolute

```

```

!===== Joint =====
!
!                               RecurDyn_Data_Name='RevJoint3' JOINT / 3
, I = 20
, J = 35
, Revolute

```

```

!===== Joint =====
!
!                               RecurDyn_Data_Name='RevJoint4' JOINT / 4
, I = 41
, J = 36
, Revolute

```

```

!===== Joint =====
!
!                               RecurDyn_Data_Name='RevJoint5' JOINT / 5
, I = 52
, J = 37
, Revolute

```

```

!===== Joint =====
!
!                               RecurDyn_Data_Name='RevJoint6' JOINT / 6
, I = 22
, J = 7
, Revolute

```

```

!===== Joint =====
!
!                               RecurDyn_Data_Name='RevJoint7' JOINT / 7
, I = 42
, J = 47
, Revolute

```



```

!===== Joint =====
!
!               RecurDyn_Data_Name='RevJoint8' JOINT / 8
, I = 48
, J = 8
, Revolute

!===== Joint =====
!
!               RecurDyn_Data_Name='RevJoint9' JOINT / 9
, I = 58
, J = 9
, Revolute

!===== Joint =====
!
!               RecurDyn_Data_Name='RevJoint10' JOINT / 10
, I = 53
, J = 59
, Revolute

!===== Axial Force =====
!
!               RecurDyn_Data_Name='RotationalAxial1' AXIAL_FORCE / 1
, I = 30
, J = 12
, ROTATION
, FUNCTION = 0.033\

!===== Spring/Damper =====
!
!               RecurDyn_Data_Name='Spring1' SPRINGDAMPER / 1
, I = 13
, J = 24
, TRANSLATION
, K = 4530
, C = 0
, FORCE = 0
, LENGTH = 7.785e-002

!===== Expression Request =====
!
!               RecurDyn_Data_Name='ExRq1' REQUEST / 1
, F2 = AZ(25,1)\
, F3 = 0.0\
, F4 = 0.0\
, F6 = 0.0\
, F7 = 0.0\
, F8 = 0.0\

!===== UNIT =====
!
UNITS /
, FORCE = Newton
, MASS = Kilogram
, LENGTH = Meter
, TIME = Second

```

```

!===== GRAVITY =====
!
accgrav / IGRAV = 0
, JGRAV = -9.80665
, KGRAV = 0

!===== OUTPUT =====
! OUTPUT/GRSAVE
, RPLTSAVE = TRUE
, RADSAVE = TRUE
, REQSAVE = TRUE
, OUTSAVE = TRUE
, MSGSAVE = TRUE
, RANSAVE = FALSE
, STRMSG = TRUE
, JOIMSG = TRUE
, CSTMSG = TRUE
, SIMMSG = TRUE
, VIOMSG = TRUE
, REDMSG = TRUE
, IDPMSG = TRUE
, DEPMSG = TRUE
, SYMSG = TRUE
, ITGMSG = TRUE
, STPMSG = TRUE
, STEP_PLOT_FACTOR = 1
, MATCH_ENDTIME = TRUE

!===== INTPAR =====
!
INTPAR / DDASSL
, KMAX = 2
, HMAX = 0.01
, INITIAL_H = 1e-006
, ERROR = 5e-006
, SPARSE = FALSE
, INTERPOLATE_H

!===== EQUILIBRIUM =====
!
EQUILIBRIUM /
, RPLT_INI = TRUE
, ERROR = 0.001
, MAXIT = 100
, INITIALSTEPsize = 1e-005
, HMAX = 1
, STEPsizeFACTOR = 2
, ANIMATIONFLAG = 0
, TYPE = 2

!===== SOLVEROPTION =====
!
SOLVEROPTION /
, TIMEOFFSET = 0.000000
, TOLERANCEIF = 1e-008
, INITIALVELOCITY = RELATIVE
, NUM_THREAD = Auto

!===== INFO =====
!
INFO /
, MODEL_NAME = SevenBodyMech2 END

```

END

TN 10-0729-01

EXPONENTIAL COURSE DETRENDING

July 29, 2010

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC) Detroit Arsenal
ATTN: RDTA-RS/MS157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

The Motion Based Technologies (MBT) team wanted to know how Aberdeen Proving Grounds (APG) detrended their course profiles and if it was using an exponential weighted filter. They wanted to ensure the use of the same detrending process when they processed their own course profiles.

PROBLEM SOLUTION:

Based on a review of Test Operating Procedure (TOP) 1-1-010 Vehicle Test Course Severity (Surface Roughness) it was determined that APG uses the "moving two-sided exponentially weighted average technique" for detrending (see excerpts from TOP 1-1-010 pages 10 and 11 below).

c. The profiles obtained represent values obtained at a nonuniform sampling interval. Linear interpolation techniques are utilized to obtain samples at equal intervals of horizontal distance as required by the spectral transformation. A digital filtering technique is also employed to account for the non-unity transfer function induced by the profilometer geometry. A moving two-sided exponentially weighted average technique (detrending) is utilized to remove long wavelength (low spatial frequency) data created by the integration process used in the above equations. This is described in reference no. 7 and is shown mathematically as:

$$y_n(x) = \frac{\sum \{y(x + l\delta x) + y(x - l\delta x)\}e^{-l\delta x/\gamma}}{2\sum e^{-l\delta x/\gamma}} \quad (\text{sum from } l = 0 \text{ to } E)$$

where:

$y(x)$ = elevation at point x .
 $y_n(x)$ = correction factor.
 l = step number.
 δx = measurement interval (0.25 ft).
 γ = weighting constant.
 E = limit of summation.

$$y_d(x) = y(x) - y_n(x)$$

where:

$y_d(x)$ = Detrended elevation at point x.

d. This filtering process does not discriminate between low frequency data created as an artifact of the integration process (due to offsets in amplifiers and digitizers, etc.) and legitimate long wavelength data such as hills. Thus, the output of the profilometer is useful for describing surface roughness for wavelengths up to approximately 30 m (100 ft) (depending upon the detrending routine coefficients), and is inappropriate for measuring slopes. It is generally accepted among the terrain roughness community that wavelengths beyond 18 m (60 ft) have little effect on vehicle dynamics, and are ignored.

This is the detrending filter used by the U.S. Army Corps of Engineers (USACE) Waterways Experiment Station (WES) mentioned in [1] and adopted from [2]. Appendix A.1 contains excerpts from [2].

MATLAB source code was also provided that implements the “standard” WES exponential weighted detrending filter as shown below.

```
%function [fdet,rms,fmean,xmean]=rmswes(x,N,A,lambda) calculate WES RMS exp.
% weighted average value of terrain
%inputs:
% x - array to filter
% N - number of samples vector eg. [0 1 2 3 4 5]
%      (wes default = 30*1=30 to get 30 ft 60=+/-30)
% A - sample spacing (wes default = 1
% lambda - exp weighting constant (wes default = 10)
%
%returns:
% fdet - detrended terrain
% rms - rms value
% fmean - trend removed
% xmean - mean of fdet
%
% for wes default
% NA = lambda*3 and lambda = 10 implies N=30/A
% N=30/A      NA=30 ft
% lambda = 10
%
function [fdet,rms,fmean,xmean]=rmswes(x,N,A,lambda);
n=length(x);
s = length(N);
e = (n+1) - s;
det1=exp(-N*A/lambda);
d=sum(det1)*2;
fmean=x;
fdet=x;
for i = s:e,
% length(x(i:-1:i-s+1))
% length( x(i:i+s-1))
fmean(i)=sum(x(i:-1:i-s+1).*det1 + x(i:i+s-1).*det1)/d;
end
fdet(s:e)=x(s:e)-fmean(s:e);
xmean=mean(fdet(s:e));
fdet(s:e)=fdet(s:e)-xmean;
%%fdet=fdet-xmean;
```

```
%rms=std(fdet(s:e)) % std(x,0) by N-1 best unbiased estimator
rms=std(fdet(s:e),1); % we divide by N
%rms=sqrt(sum(fdet(s:e).^2)/(length(fdet(s:e)))) % same as std(.,1)
%%%%%%%%%; % what we do
```

REFERENCE

- [1] Murphy, Newell. A Method for Determining Terrain Surface Roughness. U.S. Army Engineer Waterways Experiment Station, Geotechnical Laboratory. September 1984.
- [2] Van Deusen, B. D. A Statistical Technique for the Dynamic Analysis of Vehicles Traversing Rough Yielding and Non-yielding Surfaces. NASA CR-659. Prepared by CHRYSLER CORPORATION, Detroit, Mich. for NATIONAL AERONAUTICS AND SPACE ADMINISTRATION, WASHINGTON, D. C., MARCH 1967.
- [3] Test Operations Procedure (TOP) 01-1-010 Vehicle Test Course Severity (Surface Roughness) (31 Oct 2006). DIST A. Approved for public release: distribution unlimited.

APPENDIX A.1 – Excerpts from NASA CR-659

Table A-1 COEFFICIENTS OF SPECTRAL WINDOWS

Coefficient	Hamming Values	Hanning Values	Akaike Values	Linear Values	Exponential Values
A_0	.54	.50	.42	.11111	.29395
$A_1 = A_{-1}$	.23	.25	.25	.11111	.10531
$A_2 = A_{-2}$	0	0	.04	.11111	.07546
$A_3 = A_{-3}$	0	0	0	.11111	.05407
$A_4 = A_{-4}$	0	0	0	.11111	.03874
$A_5 = A_{-5}$	0	0	0	0	.02776
$A_6 = A_{-6}$	0	0	0	0	.01989
$A_7 = A_{-7}$	0	0	0	0	.01425
$A_8 = A_{-8}$	0	0	0	0	.01212
$A_9 = A_{-9}$	0	0	0	0	.00732
All others	0	0	0	0	0

non-stationary component which will contaminate P.S.D. estimates computed from the data. Two types of common trends found in surface profile traces are shown in Figure A-1. In order to reliably process data it is necessary to eliminate the trends from the data and analyze the results, with the realization that low frequency components have been removed. A numerical filter based on a moving two sided exponentially weighted average was developed to compute the trend for digital data. The data trend $\overline{F(x)}$ at point X as a function of the original data $F(x)$ is given by:

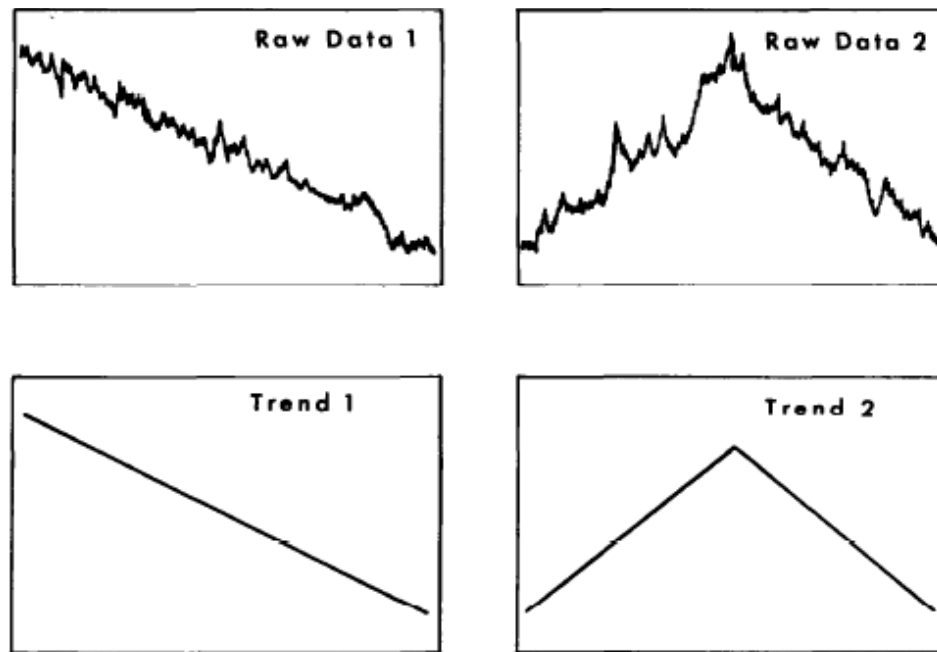


Figure A-1 TWO DATA TRENDS

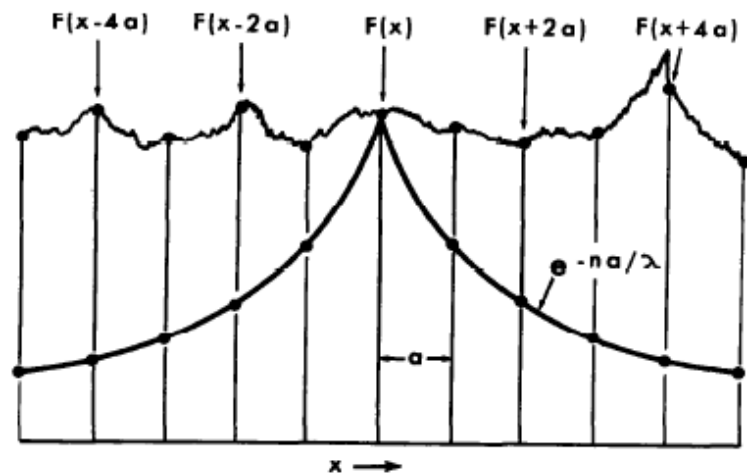


Figure A-2 EXPONENTIAL DETRENDING OF DATA POINTS

$$\overline{F(x)} = \frac{\sum_{n=0}^{\infty} [F(x + na) + F(x - na)] e^{-na/\lambda}}{2 \sum_{n=0}^{\infty} e^{-na/\lambda}} \quad (A-21)$$

Where the notation is explained in Figure A-2.

The detrended or filtered data $F_d(x)$ is then given by the relation:

$$F_d(x) = F(x) - \overline{F(x)} \quad (A-22)$$

This type of filtering is similar to an analog electrical high pass filter with the exception that two sided averaging (both past and future) can be performed on digital data to remove phase shift characteristics. To demonstrate this effect consider Equation A-21 in the limit as $n \rightarrow \infty$ and a becomes a continuous variable.

In this limit, Equation A-21 becomes:

$$\overline{F(x)} = \frac{1}{2\lambda} \int_0^{\infty} [F(x+a) + F(x-a)] e^{-a/\lambda} da \quad (A-23)$$

If $F(x)$ is now considered to be a continuous signal of the form $\sin(2\pi\Omega x)$, then

$F_d(x)$ is:

$$F_d(x) = \frac{\sin(2\pi\Omega x)}{1 + \left(\frac{1}{2\pi\Omega\lambda}\right)^2} \quad (A-24)$$

Thus, the detrending filter effects only amplitude of the input and not phase.

Equation A-24 affords an insight into the interactions and influences between detrending, the weighting constant λ , and frequency. It can be seen that the amplitude of the detrended function approaches that of the original data for large values of $\Omega\lambda$. That is, for high frequencies or for broad based exponentials, the detrended

function is approximately equal to the original function and little attenuation at this frequency is apparent. Conversely, if the product of $\Omega \lambda$ is very small the amplitude of the detrended function approaches zero and this frequency component is highly attenuated.

From Equation A-24 the transfer function as a function of frequency Ω , and weighting constant λ is:

$$\text{Amp. ratio} = \frac{1}{1 + \left(\frac{1}{2\pi\Omega\lambda} \right)^2} \quad (\text{A-25})$$

In practice a finite averaging interval must be used. For this case the transfer function is given by:

$$\text{Amp Ratio} = 1 - \frac{[1 + e^{-n} \{2\pi\Omega\lambda \sin(2\pi n\Omega\lambda) - \cos(2\pi n\Omega\lambda)\}]}{(1 - e^{-n}) [1 + (2\pi\Omega\lambda)^2]} \quad (\text{A-26})$$

The averaging interval is $-n\lambda$ to $n\lambda$, where n is any positive number and λ is the exponential weighting constant. A weighting over 3λ is reasonable for most practical applications.

Since the sample length is finite, some attention must be given to those points for which the averaging interval extends beyond the end points of the data sample. In order to account for this, the data was extended for a length 3λ beyond each of the end-points by mirror reflection. If the original data is defined as $F(x)$ on the interval $(0, L)$, then this reflection is given by:

$$\begin{aligned} F(x - a) &= F(a - x) & \text{For } x - a < 0 \\ F(x + a) &= F(2L - x - a) & \text{For } x + a > L \end{aligned} \quad (\text{A-27})$$

This endpoint reflection is a method of estimating the expected value beyond the sample length. This estimate is exact for an even periodic function but causes a distortion near the endpoints for an odd periodic function. Figure A-3 shows the distortion for a unit amplitude sine wave which has been detrended with a weighting constant equal to one tenth the wave length.

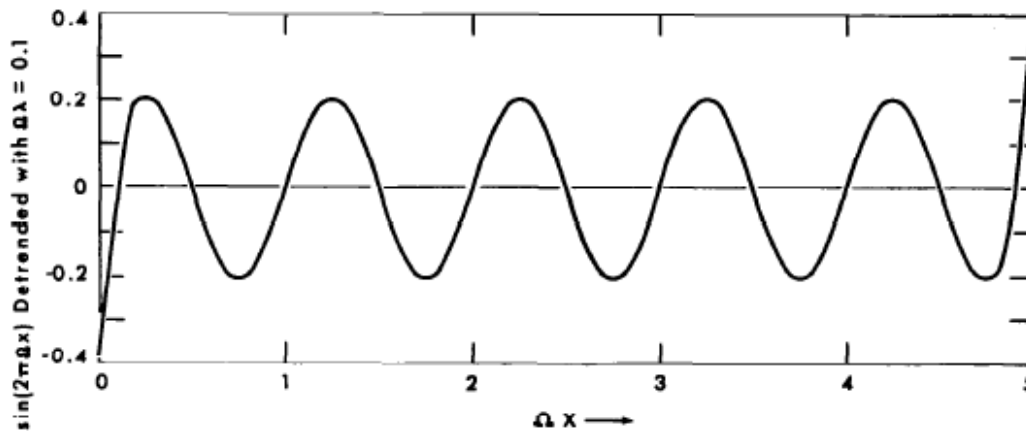


Figure A-3 EFFECT OF END POINT REFLECTION ON DETRENDED ODD FUNCTION

The frequency cut-off or half-power point $\Omega_{1/2}$ is the frequency at which the P.S.D. of the filtered function is one-half of that of the unfiltered function, and is defined as a function of λ by Equation A-28.

$$\Omega_{1/2} = \frac{1}{2\pi\lambda(\sqrt{2}-1)^{1/2}} \quad (\text{A-28})$$

The effect of an exponentially-weighted-average filter on an ideal P.S.D. predicted by Equation A-29 is shown in Figure A-4. If a trend is contained in the data sample it

$$P(\Omega) = C(\Omega)^{-2} \quad (\text{A-29})$$

is quite likely this trend will increase the estimated P.S.D. at each frequency as shown by the line marked Undetrended Estimate in Figure A-4. This, in effect, would raise the estimated value for the surface roughness constant coefficient C. The two finite data trends shown in Figure A-1 have a P.S.D. estimate similar to the ideal P.S.D. If a substantial trend such as either of these exist in the data it might obscure the real data of interest. Thus, before an accurate estimate of surface roughness can be made the data must be detrended. The effect of detrending on the ideal P.S.D. is also shown in Figure A-4. Here it is noted that detrending removes the lower frequencies.

This figure also indicates the importance of selecting an appropriate exponential weighting factor λ . Excessively large weighting factors will fail to remove the trend completely and the resulting estimates of surface roughness will be high. As the weighting factor is reduced the detrended P.S.D. approaches the ideal and reasonable estimates of surface roughness can be made. As the weighting factor is further reduced, higher frequencies are filtered out, and the estimate of surface roughness can be contaminated by statistical errors in resolving higher frequencies from digital data.

A.1.4 Amplitude Probability Distribution

The amplitude probability distribution (A.P.D.) of a continuous random variable $F(x)$ as given by Equation A-30 defines the probability that $F(x)$ will exceed the level Y .

$$\text{A.P.D. (Y)} = \lim_{X \rightarrow \infty} \frac{1}{2X} \int_{-X}^X \delta[F(x) > Y] dx \quad (\text{A-30})$$

where $\delta[F(x) > Y]$ is a delta function defined by Equation A-31.

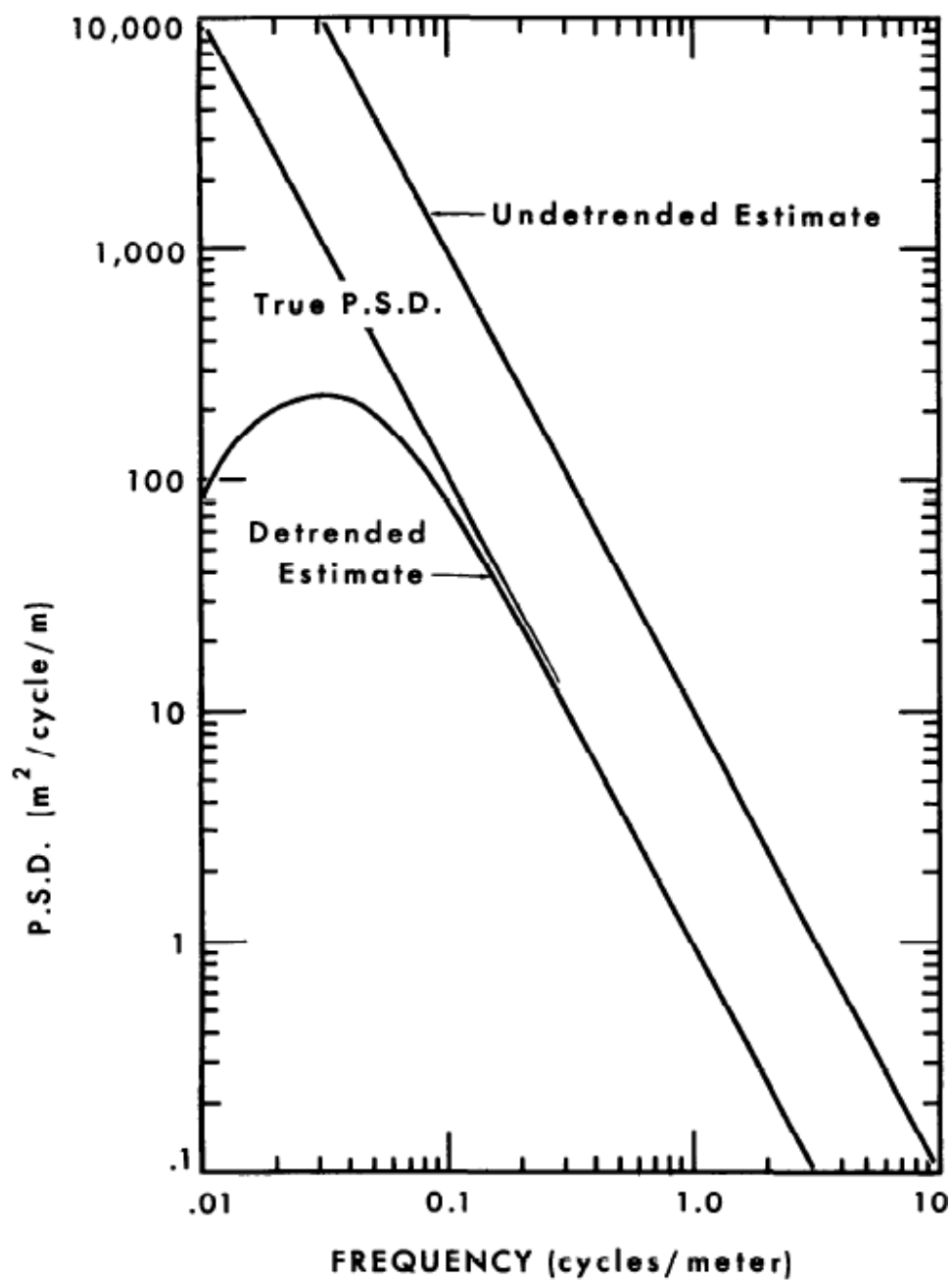


Figure A-4 EFFECT OF EWA FILTER ON P.S.D. ESTIMATE

TN 10-1021-01

ABSORBED POWER CALCULATION

October 21, 2010

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC)
Detroit Arsenal
ATTN: RDTA-RS/MS157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

From the JLTV program there were questions regarding different aspects with "regard[s] to transforming time domain data into frequency domain that can affect...absorbed power calculations." A request was made for a POC at TACOM's modeling and simulation group for assistance with the calculation. A MATLAB script was provided to confirm if the correct absorbed power filter was being used for the accelerations in the vertical direction.

PROBLEM SOLUTION:

Review of [1] and Appendix A.1 define time based filter coefficients for each acceleration power spectral density frequency. Review of [2] and Appendix A.2 thru A.5 shows a simpler filtering technique based on just acceleration. The MATLAB script provided was based on the first method. The response is as follows:

You are using the definition as outlined in Lee & Pradko, "Analytical Analysis of Human Vibration". SAE 680091, 1986. I've attached Test Operations Procedure (TOP) 1-1-014 Ride Dynamics (Approved for Public Release), used at Aberdeen Proving Grounds, that may give you more information. It also references ISO 2631-1, Mechanical Vibration and Shock-Evaluation of Human Exposure to Whole-Body Vibration, whose vertical frequency weightings are close to that for absorbed power.

Usually, a time trace of the acceleration at some point of interest is filtered by the vertical frequency weighting function (convolved in the time domain or multiplied in the frequency s domain), squared, integrated over the time period T, and divided by time period T to get the average (see the formula in TOP 1-1-014 Pg 5---without the square root and where a_w is the weighted acceleration).

Either way you can test your output. After 30 seconds, an 8 ft/sec² sinusoidal acceleration at 4 Hz should give you between 6.2 and 6.21 watts of absorbed power at $t=30$. The standard absorbed power filter requires the input acceleration to be in ft/sec².

```
====MATLAB code defining sinusoidal acceleration input
l=30000;
t=[0:l]*.001;
```

```
u=(8)*sin(2*3.14159*4*t); % 8 ft/sec^2 or 0.25 4Hz g's.
=====
```

Applying the method outlined in Appendix A.2 to the filter definition in Appendix A.1 the magnitude of each are compared in Figures 1 and 2. The s-domain filter is developed through the use of the MATLAB conv() and tf('s') functions.

Example

```
n1 = [0 1 0];
d1 = [1 29.8 1000];
...
n4 = [0 1 5];
d4 = [0 1 6];

n5 = conv(n4,conv(n3,conv(n1,n2)));
d5 = conv(d4,conv(d3,conv(d1,d2)));
sys = tf('s')
sys = tf(15.45*n5,d5)

w = [.1:.1:1000]'; % w in rad/sec
[m,p,w] = bode(sys,[.1:.1:1000]);

for i = 1:length(w)
    mdb(i) = 20*log10(m(1,1,i));
    mag(i) = m(1,1,i);
end
```

The time domain filter is developed based on [1].

Example

```
w = [.1:.1:1000]'; % w in rad/sec
k1 = 1.356;
k0 = 4.3537;

for i = 1:length(w)
    f1 = -0.10245e-9*w(i)^6+0.17583e-5*w(i)^4-0.44601e-2*w(i)^2+1;
    f2 = 0.12882e-7*w(i)^4-0.93394e-4*w(i)^2+0.10543;
    f3 = -0.45416e-9*w(i)^6+0.37667e-5*w(i)^4-0.56104e-2*w(i)^2+1;
    f4 = -0.21179e-11*w(i)^6+0.51728e-7*w(i)^4-0.17947e-3*w(i)^2+0.10543;
    mag(i) = sqrt(k0*k1*(f1*f4-f2*f3)/(f3*f3+w(i)^2*f4*f4));
    mdb(i) = 20*log10(mag(i));
end
```

It should be noted that the s-domain filter implemented in VEHDYN [3] has two slightly different coefficients than used in Appendix A.3---80.0 replaces 105.0 and 100.0 replaces 125.0 in the last numerator and denominator quadratics. This is generally the Army's accepted "absorbed power" filter.

The curve fitting routine outlined in [4] can account for the slight differences between the two filter domains.

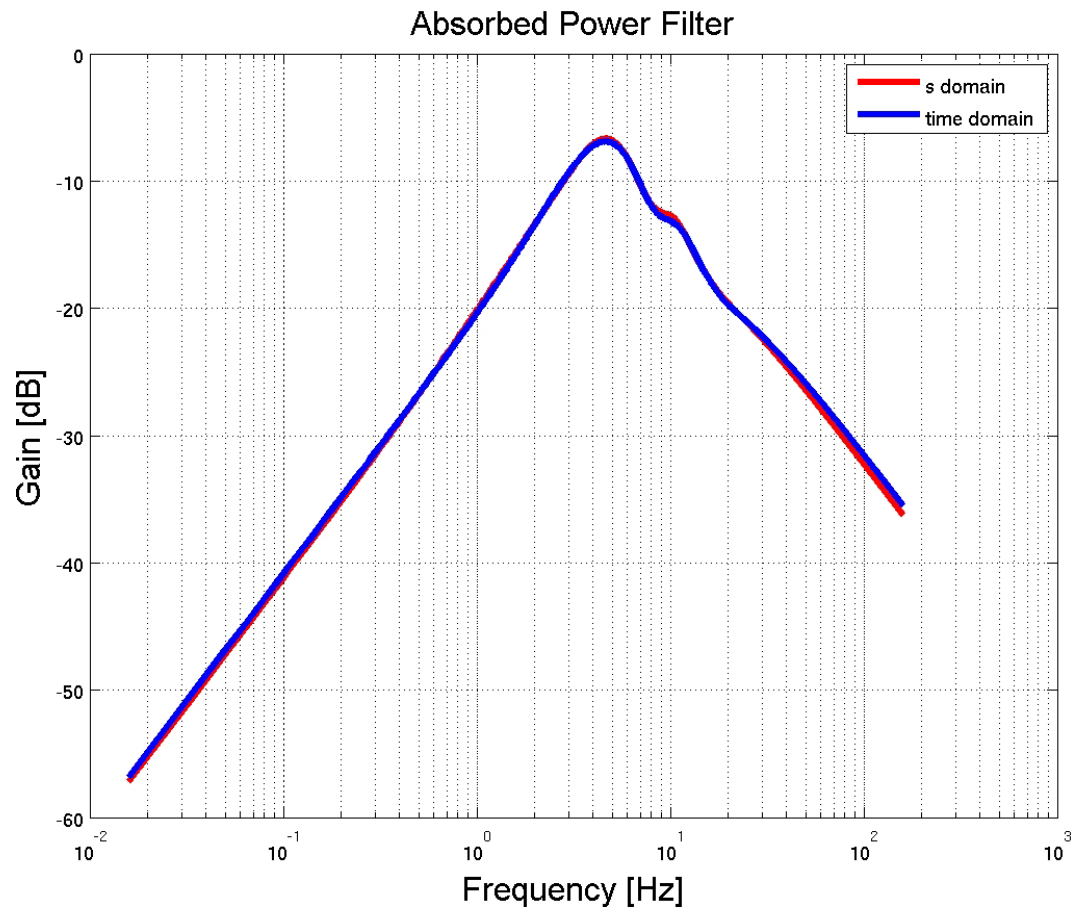


Figure 1 – Vertical Absorbed Power Filter s-domain

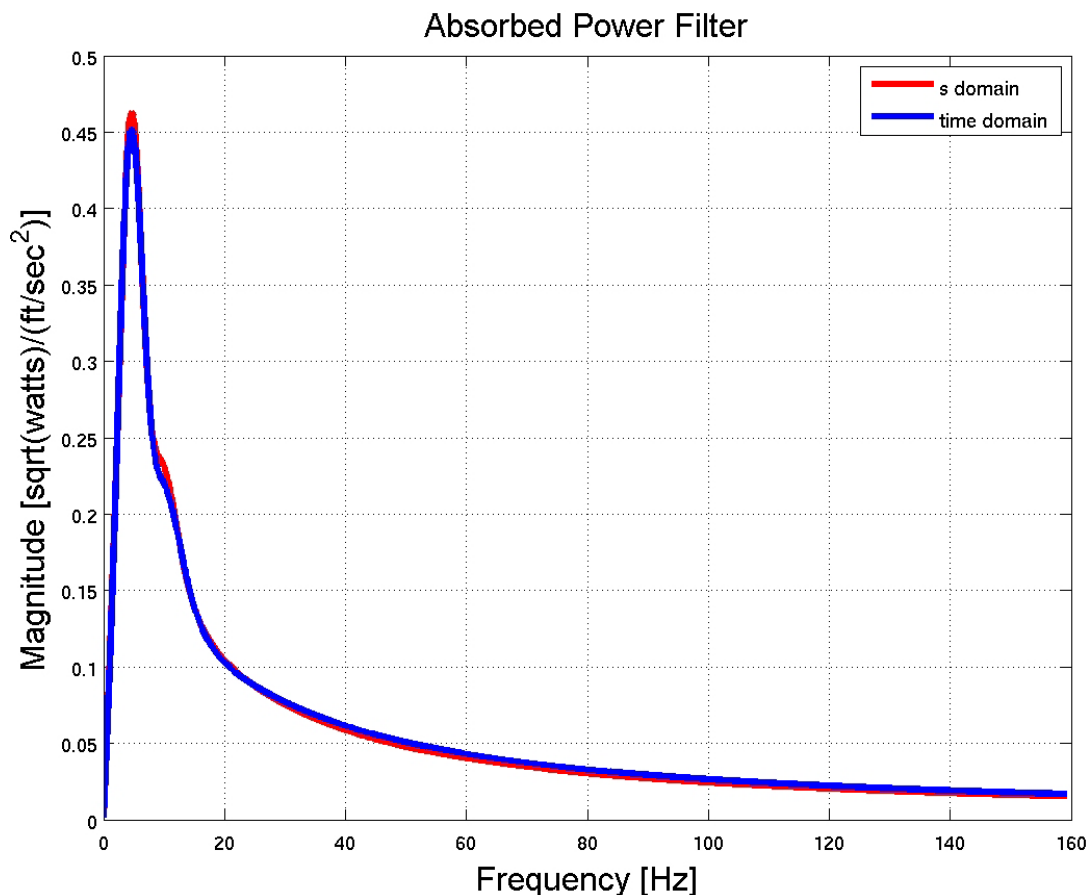


Figure 2 – Vertical Absorbed Power Filter time domain

Review of [5] and [6] and application of the method used above for the time domain filter (but using the vertical (Wk) weightings outlined in [5]) showed greater differences between the vertical frequency weightings of ISO 2631-1 and absorbed power than originally thought.

The ISO (vertical) time domain filter is developed based on [5]. Comparison between the values in Table 3 (Principal frequency weightings in one-third octaves) and the filter defined in Table A.1 of Annex A.1 showed very close agreement---with the filter allowing for finer resolution.

Example

```
%Table 3 Method
isof = [.1, .125, .16, .2, .25, .315, .4, .5, .63, .8, 1, 1.25, 1.6, 2, 2.5, 3.15,
        4, 5, 6.3, 8, 10, 12.5, 16, 20, 25, 31.5, 40, 50, 63, 80, 100, 125, 160,
        200, 250, 315, 400];
isom = [31.2, 48.6, 79, 121, 182, 263, 352, 418, 459, 477, 482, 484, 494, 531, 631,
        804, 967, 1039, 1054, 1036, 988, 902, 768, 636, 513, 405, 314, 246, 186,
        132, 88.7, 54, 28.5, 15.2, 7.9, 3.98, 1.95];
isom = isom/1000;
isomag = interp1(isof,isom,w/(2*pi),'linear');
isodb = 20*log10(isomag);

%Filter Method - Annex A.1, Table A.1
w = [.1:1:1000]'; % w in rad/sec
f1 = 0.4
f2 = 100
f3 = 12.5
f4 = 12.5
```

```

q4 = 0.63
f5 = 2.37
q5 = 0.91
f6 = 3.35
q6 = 0.91

for i = 1:length(w)
    f = w(i)/(2*pi);
    isofh(i) = sqrt(f^4/(f^4+f1^4));
    isofl(i) = sqrt(f2^4/(f^4+f2^4));
    isoft(i) = sqrt((f^2+f3^2)/f3^2)*sqrt((f4^4*q4^2)/(f^4*q4^2+f^2*f4^2*(1-2*q4^2)+f4^4*q4^2));
    isofs(i) = (q6/q5)*sqrt((f^4*q5^2+f^2*f5^2*(1-2*q5^2)+f5^4*q5^2)/(f^4*q6^2+f^2*f6^2*(1-2*q6^2)+f6^4*q6^2));
    isofhlts(i) = 3.28*isofh(i)*isofl(i)*isoft(i)*isofs(i); %m/s^2 to ft/s^2 is 3.28 ft = 1m
end
isodb = 20*log10(isofhlts);

w(find(absmag==max(absmag)))/(2*pi)
w(find(isofhlts==max(isofhlts)))/(2*pi)

```

Figure 3 quantifies the differences with the Absorbed Power filter peak at 4.63 Hz and the ISO filter peak at 6.17 Hz. It should be noted that the acceleration conversion from m/s² (expected by ISO 2631-1) to ft/s² for the comparison in Figure 3 multiplies the ISO filter by a factor of 3.28---significantly increasing the magnitude.

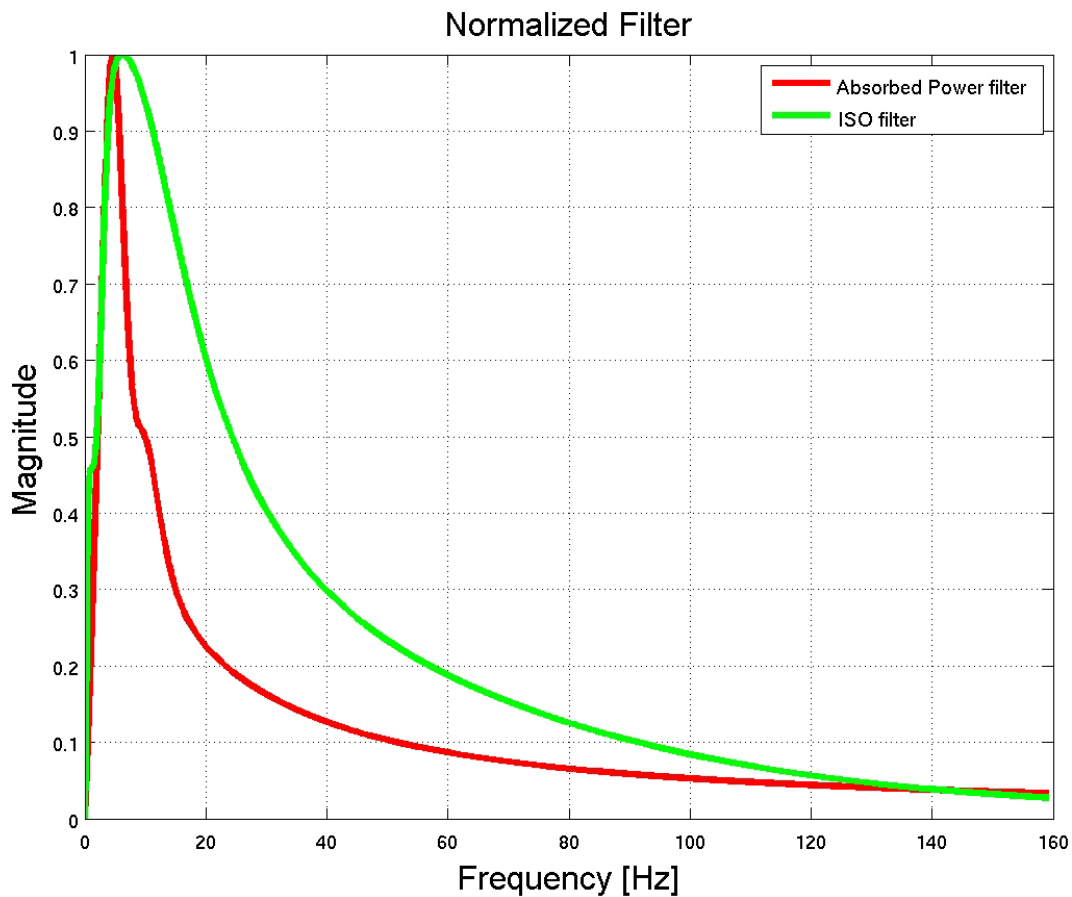


Figure 3 – Normalized Vertical Absorbed Power and ISO Time Domain Filters

The results in Figure 3 differ from those in [6]. The origin is unclear but is expected to be due to a later version of ISO 2631-1 or the conversion factor (3.28) between m/s^2 and ft/s^2 ---as the absorbed power filter assumes the acceleration is in ft/s^2 .

REFERENCE

- [1] Test Operations Procedure (TOP) 1-1-14 Ride Dynamics (7 Jan 2005). DIST A (DTIC AD A431 130). Approved for public release: distribution unlimited.
- [2] Lee and Lins. "Human Vibration Measuring Instrument". Army Tank-Automotive Command Warren, MI. 1973. DIST A (DTIC AD 0785 648). Approved for public release: distribution unlimited.
- [3] Creighton, D. C. "Revised vehicle dynamics module: user's guide for computer program VEHDYN II". U.S. Army Engineer Waterways Experiment Station, Vicksburg, MS. 1986. TR SL-86-9. DIST C (DTIC AD B102 925).
- [4] Lins. "Human Vibration Response Measurement". U.S. Army Tank-automotive Command Laboratories. 1972. TR 11551. DIST C (DTIC AD 0901 195).
- [5] ISO 2631-1:1997(E). "Mechanical vibration and shock – Evaluation of human exposure to whole-body vibration Part 1: General requirements. 2nd Edition-corrected and reprinted 1997-07-15.
- [6] Murphy and Ahmad. "Comparison of Measures of Vibration Affecting Occupants of Military Vehicles". Waterways Experiment Station (WES), Corps of Engineers, Vicksburg, MS 39180-0631. Final Report, December 1986. TR GL-86-18. DIST A (DTIC AD A178 359). Approved for public release: distribution unlimited.

APPENDIX A.1 – TOP 1-1-14 RIDE DYNAMICS

TOP 1-1-014
7 January 2005

4.2 Absorbed Power Technique: Another technique for evaluating human response to vibration or ride quality is the computation of absorbed power. It is a measure of the rate at which energy is absorbed by a human subjected to ride vibration. It is accepted as a measure of human tolerance to vibration for military vehicles negotiating rough terrain. The absorbed power for a given location and axis is computed by multiplying the acceleration power spectral density spectrum by the appropriate transfer function and integrating the resultant spectrum. An advantage of this approach is that average absorbed power is a scalar quantity and can be summed in complex multi-degree of freedom systems to yield a single value describing the total average absorbed power. Typically a standard ride quality test procedure involves determining the speed at which the vertical average absorbed power reaches an upper limit of 6 watts for different types of terrain at selected crew location seats. The terrain is usually characterized by the surface roughness reported as inches or centimeters rms. The vehicle speed as a function of terrain roughness obtained from this process is used as one of the mobility limiting factors in the NATO Reference Mobility Model. Military vehicle specifications frequently include ride quality requirements based on absorbed power (e.g., the vehicle must be capable of producing a ride of 6 watts or less at the driver's seat in the vertical axis for a given speed and surface roughness).

The absorbed power in each axis will be calculated from:

$$P = \sum_{i=1}^n (C_i) A_i^2$$

where:

P = Absorbed power, watts

A_i = rms acceleration in ft/sec² within the i^{th} spectral band

$$C_i = K_1 K_0 (F_1 F_4 - F_2 F_3) / (F_3^2 + W_i^2 F_4^2)$$

W_i = Frequency, radians/second

F and K values = Calculated from Appendix D

The result is a frequency weighting, scaling and integration of the acceleration power spectral density. The weighting functions are shown (by axis) in Figure 5. The factors have been normalized to a value of 1.0 to show frequency without the appropriate scaling.

TOP 1-1-014
7 January 2005

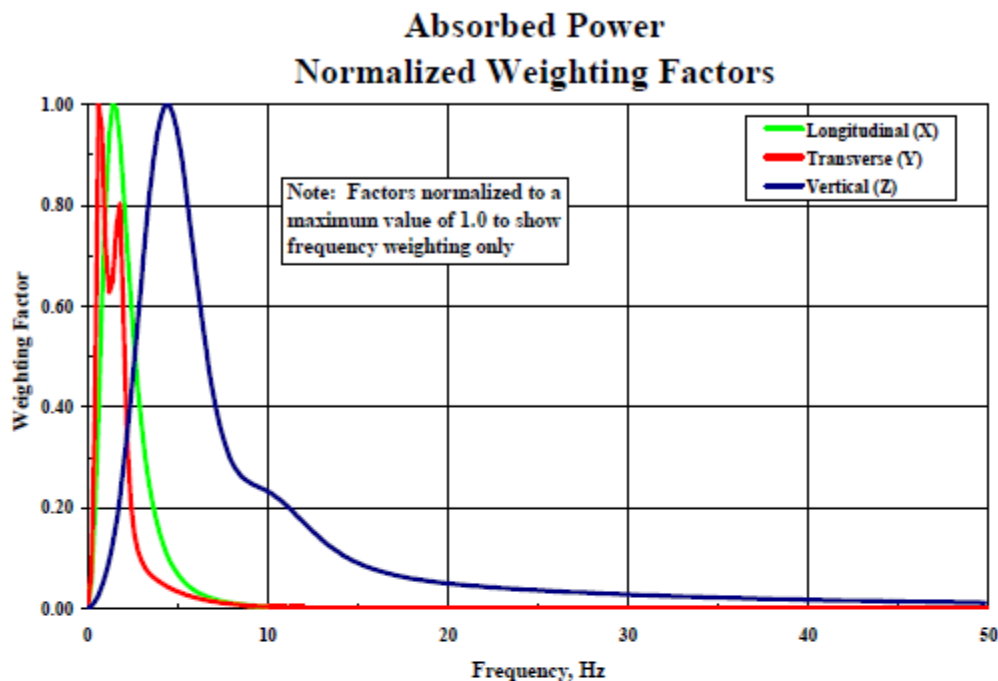


Figure 5. Absorbed Power Normalized Frequency Weighting Factors

For each test course, the absorbed power will be plotted as a function of speed. Specification requirements are generally written in terms of the driver's vertical absorbed power (only), but the following technique will be applied to other locations and axes, if applicable. A non-linear interpolation or a power law ($y = ax^b$, linear fit in a log-log domain,) or n th degree polynomial curve fit will be performed and the speed at which an absorbed power of 6 watts is achieved will be extrapolated from the data set. Absorbed power values of less than 1 watt or greater than 10 watts will not be used in the curve fit process, if possible. The 6-watt speed will then be plotted as a function of terrain roughness rms (a specific value for each test course) to determine a ride quality curve for the vehicle.

4.3 Half-Round Obstacle Technique. The peak acceleration from the vertical accelerometer near the driver's seat will be low-pass filtered at 30 Hz and the resulting peak acceleration will be plotted as a function of speed for each half-round obstacle. The speed at which a peak value of 2.5 g's is reached will be determined by a non-linear interpolation or curve fitting technique as described above.

TOP 1-1-014
7 January 2005

APPENDIX D. ABSORBED POWER CONSTANTS

Longitudinal (X)

K_0	4.3532
K_1	1.356
W_i	$2\pi f_i$
F_1	1.0
F_2	0.219106
F_3	$-0.0185309W_i^2 + 1$
F_4	$-0.00061893W_i^2 + 0.219106$

Note: f_i = Center frequency (Hz) of i^{th} spectral bandTransverse (Y)

K_0	4.353
K_1	1.356
W_i	$2\pi f_i$
F_1	$0.24052124 \times 10^{-3} W_i^4 - 0.066974483 \times 10^{-2} W_i^2 + 1$
F_2	$0.57384538 \times 10^{-5} W_i^4 - 0.50170413 \times 10^{-2} W_i^2 + 0.33092592$
F_3	$-0.14979958 \times 10^{-5} W_i^6 + 0.0010088882 W_i^4 - 0.10108617 W_i^2 + 1$
F_4	$-0.1713749 \times 10^{-7} W_i^6 + 0.53137351 \times 10^{-4} W_i^4 - 0.011096507 W_i^2 + 0.33092592$

Note: f_i = Center frequency (Hz) of i^{th} spectral band

TOP 1-1-014
7 January 2005

Vertical (Z)

K_0	4.3537
K_1	1.356
W_i	$2\pi f_i$
F_1	$-0.10245 \times 10^{-9} W_i^6 + 0.17583 \times 10^{-5} W_i^4 - 0.44601 \times 10^{-2} W_i^2 + 1$
F_2	$0.12882 \times 10^{-7} W_i^4 - 0.93394 \times 10^{-4} W_i^2 + 0.10543$
F_3	$-0.45416 \times 10^{-9} W_i^6 + 0.37667 \times 10^{-5} W_i^4 - 0.56104 \times 10^{-2} W_i^2 + 1$
F_4	$-0.21179 \times 10^{-11} W_i^6 + 0.51728 \times 10^{-7} W_i^4 - 0.17947 \times 10^{-3} W_i^2 + 0.10543$

Note: f_i = Center frequency (Hz) of i^{th} spectral band

APPENDIX A.2 – Lee and Lins pg 207

LEE & LINS

If an acceleration signal is inputted to a filter circuit, the output frequency W_1 is given by:

$$H(jW_1)A(W_1) \quad (12)$$

where $H(jW_1)$ is the weighting function for the filter circuit. Squaring this quantity and equating it to absorbed power at W_1 :

$$H(jW_1) H(-jW_1) A^2(W_1) = K(W_1) A^2(W_1) \quad (13)$$

or:

$$|H(jW_1)|^2 A^2(W_1) = K(W_1) A^2(W_1) \quad (14)$$

Thus:

$$|H(jW_1)| = \sqrt{K(W_1)} \quad (15)$$

Because of the squaring process after filtering, we are interested in only the magnitude of $H(jW_1)$.

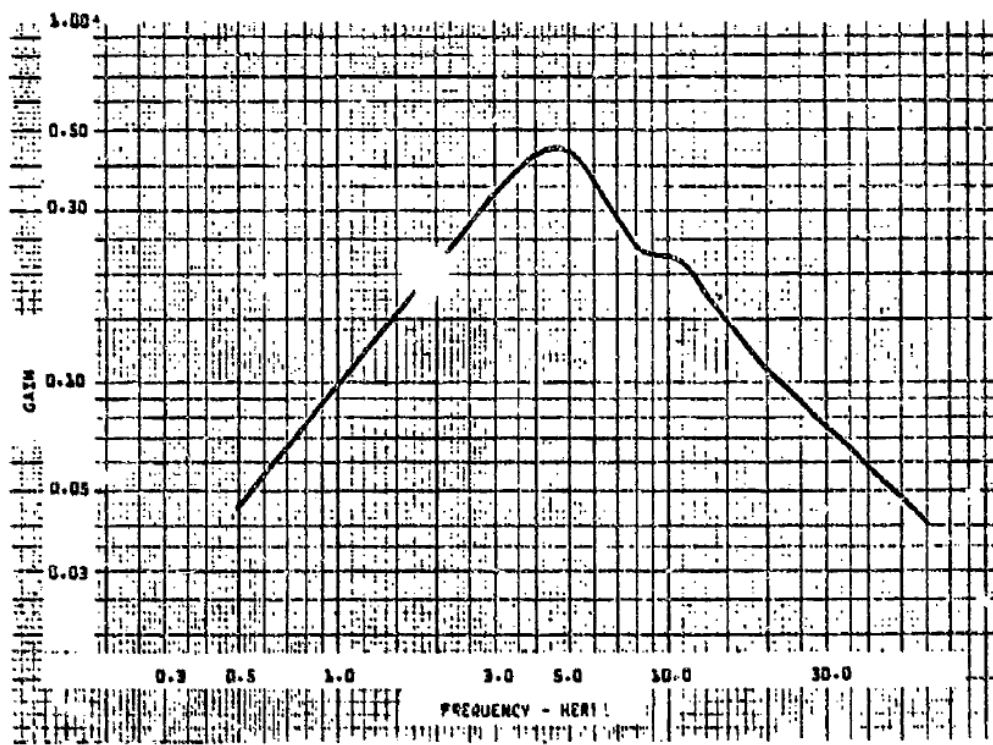
The entire absorbed power computing process is shown in Figure 2 for the three directions of motion. The frequency response was plotted for each motion and polynomials in terms of the complex variable S were fitted to these curves. The weighting function circuits were then synthesized from the derived transfer functions. The frequency response, transfer function in terms of S and the circuit diagram for the weighting functions are given in Figures 3-5 for vertical, longitudinal, and transverse directions of motion respectively. The units on the axis of the frequency response curves is watts/ft/sec². A more detailed description of the development of these functions and circuits is given in reference (4).

FIELD TESTS

Several portable absorbed power measuring devices have been

APPENDIX A.3 – Vertical Absorbed Power Filter (s-domain)

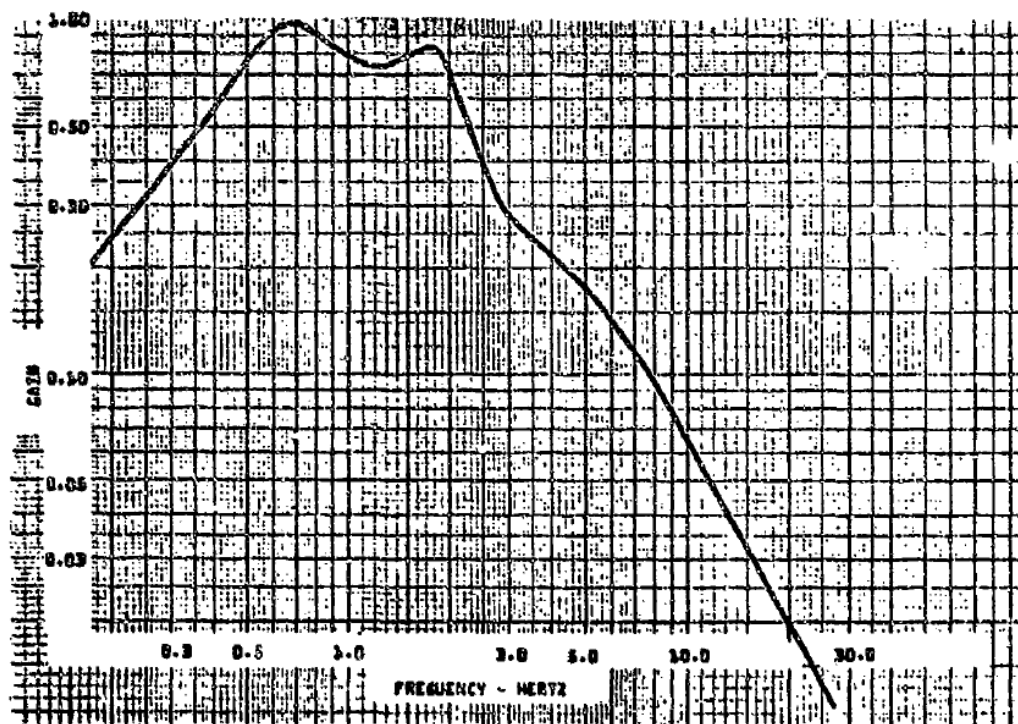
LEE & LINS



$$\frac{15.453 S (S + 5.0) (S^2 + 28.3 S + 2800) (S^2 + 105.0 S + 7570)}{(S + 6.0) (S^2 + 29.8 S + 1000) (S^2 + 39.1 S + 3800) (S^2 + 125.0 S + 5160)}$$

APPENDIX A.4 – Longitudinal Absorbed Power Filter (s-domain)

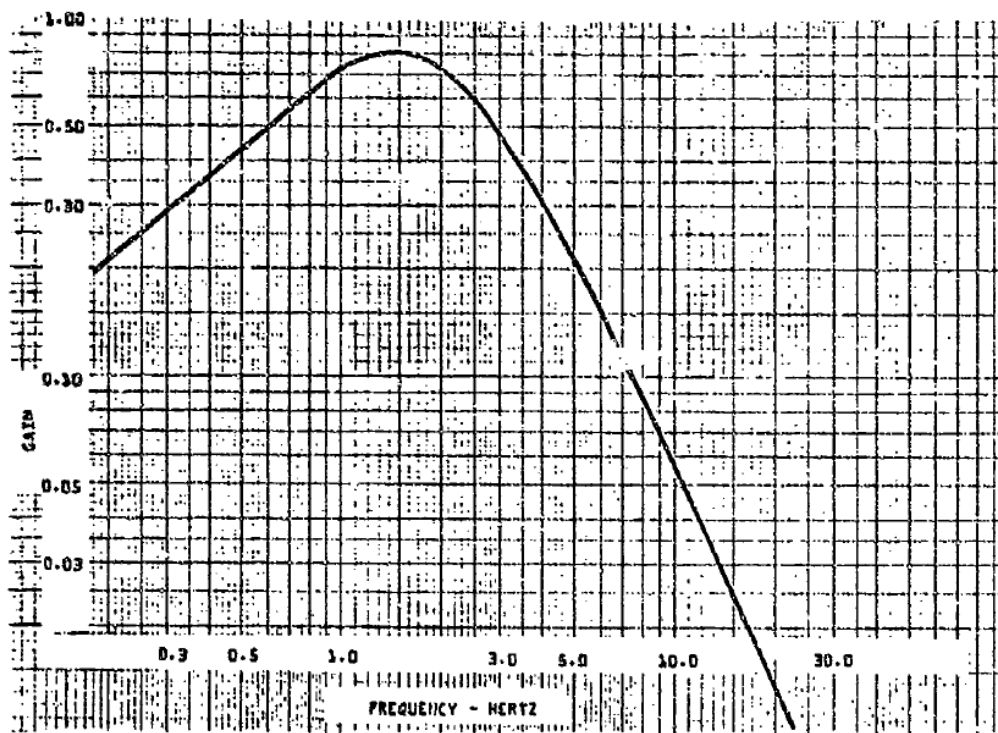
LEE & LINS



$$\frac{478.0 S (S + 130.0) (S^2 + 11.2 S + 60.0) (S^2 + 14.2 S + 260.0)}{(S^2 + 3.33 S + 17.0) (S^2 + 5.5 S + 140.0) (S^2 + 44.0 S + 900.0) (S^2 + 255.0 S + 2500)}$$

APPENDIX A.5 – Transverse Absorbed Power Filter (s-domain)

LEE & LINS



$$\frac{209.0 S (S + 110.0)}{(S^2 + 17.6 S + 125.0) (S^2 + 110.0 S + 1300)}$$

TN 11-0718-01

WHEELED VEHICLE ONE-PASS VEHICLE CONE INDEX (VCI) FORMULA

July 18, 2011

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC)
Detroit Arsenal
ATTN: RDTA-RS/MS157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

Many questions were raised by PM-LTV about the parameters used in the Vehicle Cone Index (VCI) formula provided by ERDC for determining the Modernized Expanded Capacity Vehicle (MECV) soft soil mobility performance (Annex E to the MECV RFP). Clarifications were sought from the Dynamics and Structures Team.

PROBLEM SOLUTION:

Review of the VCI formula in Appendix A.1 indeed revealed a lack of clarity in the formula definitions. Referencing the VPP.FOR subroutine in the NATO Reference Mobility Model (NRMM) helped to clarify its usage but a standard definition, other than in Appendix A.1, was lacking.

Requests to ERDC, the originator of the formula, were not successful for any "official" reference. However, an article, see reference [2], helped explain many of the details of the formula and its parameter requirements. Reference [2] was essentially adopted as the "standard definition" for VCI (wheeled vehicles) since these definitions are much more direct in defining what is really needed to perform an accurate VCI calculation. The previous formula (Appendix A.1) was somewhat ambiguous in this regard.

Annex E (Single-Pass Vehicle Cone Index (VCI1) Calculation) to the HMMWV Modernized Expanded Capacity Vehicle (MECV) Performance Specification was updated from [2] as follows:

Definitions

b	average tire section width (inflated, unloaded) [in.]
CF	clearance factor
CPF	contact pressure factor
d	average tire outside diameter (inflated, unloaded), [in.]
DCF	deflection correction factor
EF	engine factor
GF	grouser factor
h	average tire section height (inflated, unloaded) [in.]
h_c	vehicle minimum clearance height [in.]
MI	mobility index
n	average number of tires per axle
PWR	power to weight ratio [hp/ton]
TEF	traction element factor
TF	transmission factor
VCI_1	one-pass vehicle cone index [psi]
w	average axle loading [lbs.]
WF	weight factor
WLF	wheel load factor
δ	average hard-surface tire deflection [in.]

Vehicle Cone Index (one-pass, wheeled vehicle)

$$VCI_1 = f(MI, DCF)$$

$$where \begin{cases} MI \leq 115 \rightarrow VCI_1 = \left(11.48 + 0.2MI - \frac{39.2}{MI + 3.74} \right) DCF, \\ MI > 115 \rightarrow VCI_1 = (4.1MI^{0.446}) DCF. \end{cases}$$

Mobility Index

$$MI = \left(\frac{(CPF)(WF)}{(TEF)(GF)} + WLF - CF \right) (EF)(TF)$$

Mobility Index Factors

$$CPF = \frac{w}{0.5ndb}$$

$$TEF = \frac{10 + b}{100}$$

$$WLF = \frac{w}{2000}$$

$$CF = \frac{h_c}{10}$$

$GF = 1 + 0.05C_{GF}$, where $C_{GF} = 1$ if tire chains are used or 0 if not

$EF = 1 + 0.05C_{EF}$, where $C_{EF} = 1$ if $PWR < 10 \frac{hp}{ton}$ or 0 if not

$TF = 1 + 0.05C_{TF}$, where $C_{TF} = 1$ if manual transmission or 0 if automatic

$$WF = C_{WF1} \frac{w}{1000} + C_{WF2}$$

$$\text{where } \begin{cases} w < 2000 \text{ lbs.} \rightarrow C_{WF1} = 0.553 \text{ and } C_{WF2} = 0, \\ 2000 \leq w < 13,500 \text{ lbs.} \rightarrow C_{WF1} = 0.033 \text{ and } C_{WF2} = 1.050, \\ 13,500 \leq w < 20,000 \text{ lbs.} \rightarrow C_{WF1} = 0.142 \text{ and } C_{WF2} = -0.420, \\ 20,000 \leq w < 31,500 \text{ lbs.} \rightarrow C_{WF1} = 0.278 \text{ and } C_{WF2} = -3.115, \\ 31,500 \leq w \rightarrow C_{WF1} = 0.836 \text{ and } C_{WF2} = -20.686. \end{cases}$$

Deflection Correction Factor

$$DCF = \left(\frac{0.15}{\delta/h} \right)^{0.25}$$

For multi-pass VCI values consult reference [3].

REFERENCE

- [1] Reference Internal Project 10-038 MECV
- [2] "Clarification of vehicle cone index with reference to mean maximum pressure", Journal of Terramechanics 43 (2006) 85-96, J. D. Priddy and W. E. Willoughby, (www.elsevier.com/locate/jterra).
- [3] NATO Reference Mobility Model Ed II, NRMM II User's Guide. Waterways Experiment Station (WES), Corps of Engineers, Vicksburg, MS 39180-0631. Final Report, December 1992. TR GL-92-19. DIST C (DTIC AD B170 301). VPP.FOR (vehicle preprocessor subroutine).
- [4] Performance Specification for HMMWV Modernized Expanded Capacity Vehicle (MECV) (ver 2.0, 7 Nov 2011). Annex E – Single-Pass Vehicle Cone Index (VCI1) Calculation. DIST A. Approved for public release: distribution unlimited.
- [5] Kennedy, Rush. Trafficability of Soils (Technical memorandum No. 3-240) – Development of Revised Mobility Index Formula for Self-Propelled Wheeled Vehicles in Fine-Grained

Soils. US Army Engineer Waterways Experiment Station, Corps of Engineers (March 1968). Appendix A: Vehicle Cone Index Computations. DIST A (DTIC AD 832 912)). Approved for public release: distribution unlimited.

- [6] Wong. Theory of Ground Vehicles, 3rd Ed. John Wiley & Sons, 2001 (pp. 120-126).
- [7] Rosco, et. al. Considerations Regarding Vehicle Cone Index for Fine-Grained Soil. Journal of Military Technology, Vol. 1, No. 2, Dec. 2018).
- [8] Ragheb, et al. Evaluation of Wheeled Vehicle Mobility and Performance Prediction on Deformable Soil. Proceedings of the 13th Int. AMME Conf., 27-29 May, 2008.
- [9] Bylsma. SIMPLIFIED DYNAMICS AND MOBILITY FACTORS FOR MULTI-DISCIPLINARY OPTIMIZATION OF AN OCCUPANT CENTRIC PLATFORM, TR 22829, April 2012. DIST A (DTIC AD A559 920) Approved for public release: distribution unlimited.

APPENDIX A.1 – VCI Formula (Hand Calculation) – ERDC

One-Pass Vehicle Cone Index (VCI)
Hand Calculation Formulas for Wheeled Vehicles

30-Sep-02

$$\text{Contact Pressure Factor} = \frac{\text{gross vehicle weight (lb)}}{\text{section width (in)} \times \frac{\text{outer diameter (in)}}{2} \times \text{No. of tires}}$$

	Weight Range $\left(\frac{\text{gross vehicle weight (lb)}}{\text{No. of axles}} \right)$	Weight Factor Equations
Weight Factor :	less than 2,000	$Y = 0.553 X$
	2,000 to 13,500	$Y = 0.033 X + 1.050$
	13,501 to 20,000	$Y = 0.142 X - 0.420$
	20,001 to 31,500	$Y = 0.278 X - 3.115$
	greater than 31,500	$Y = 0.836 X - 20.686$

where $X = \frac{\text{gross vehicle weight (kips)}}{\text{No. of axles}}$ and $Y = \text{Weight Factor}$

$$\text{Tire Factor} = \frac{10 + \text{section width (in)}}{100}$$

Grouser Factor :

With tire chains	=	1.05
Without tire chains	=	1.00

$$\text{Wheel Load Factor} = \frac{\text{gross vehicle weight (kips)}}{\text{No. of wheels (duals count as one)}}$$

$$\text{Clearance Factor} = \frac{\text{vehicle ground clearance (in)}}{10}$$

Engine Factor :

engine power to gross vehicle weight ratio < 10 (hp/ton)	=	1.05
engine power to gross vehicle weight ratio ≥ 10 (hp/ton)	=	1.00

Transmission Factor :

manual shift transmission	=	1.05
automatic shift transmission	=	1.00

One-Pass Vehicle Cone Index (VCI)
Hand Calculation Formulas for Wheeled Vehicles

30-Sep-02

$$\text{Mobility Index (MI)} = \left[\frac{\text{Contact Pressure Factor} \times \text{Weight Factor}}{\text{Tire Factor} \times \text{Grouser Factor}} + \frac{\text{Wheel Load Factor} - \text{Clearance Factor}}{\text{Factor}} \right] \times \frac{\text{Engine Factor}}{\text{Factor}} \times \frac{\text{Transmission Factor}}{\text{Factor}}$$

$$\text{MI} \leq 115: \text{VCI}_1 = \left(11.48 + 0.2 \text{MI} - \frac{39.2}{\text{MI} + 3.74} \right) \left(\frac{0.15}{\frac{\text{tire deflection (in)}}{\text{section height (in)}}} \right)^{0.25}$$

Vehicle Cone Index :

$$\text{MI} > 115: \text{VCI}_1 = \left(4.1 \text{MI}^{0.446} \right) \left(\frac{0.15}{\frac{\text{tire deflection (in)}}{\text{section height (in)}}} \right)^{0.25}$$

Pneumatic Tire Terms (see sketch)

Carcass. The portion of a tire outside the lip of the rim flange that is free to deflect.

Diameter (d). The outside diameter, not counting the tread height, of an inflated but unloaded tire.

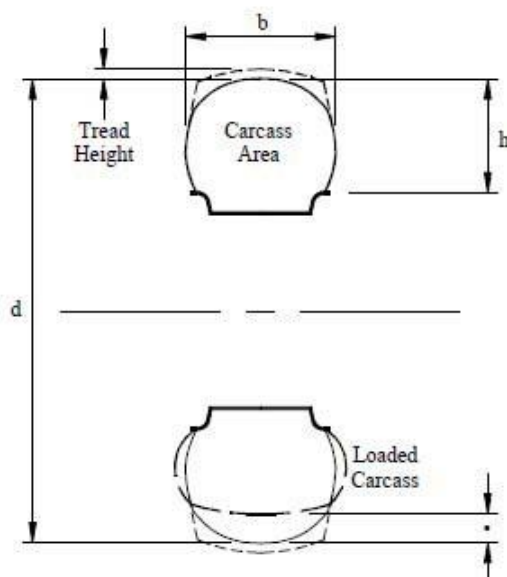
Section height (h). The distance from the lip of the rim flange to the outer tire surface, not counting the tread height, of an inflated but unloaded tire.

Section width (b). The distance between the extreme points on the tire carcass area of an inflated but unloaded tire.

Deflection (•). The distance between the outer surface of a loaded tire and the outer surface of an inflated but unloaded tire taken at the center of the carcass cross section on a non-yielding surface.

Tread height. The thickness of the aggressive features on the outer surface of a tire.

Aspect ratio. The ratio of section height over section width.



TN 12-0727-01

POLAR PLOTTING – MULTIPLE PLOTS AND TRANSPARENCY

July 27, 2012

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC)
Detroit Arsenal
ATTN: RDTA-RS/MS157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

TARDEC's Physical Simulation Team requested assistance with plotting human factors data, field of view from inside a vehicle, using the MATrix LABoratory (MATLAB) from [1]. Multiple plots on the same chart were needed for comparison purposes.

PROBLEM SOLUTION:

The field of view data requires a polar plot for proper representation and interpretation of the data. The **polar** plot command is used to make plots as shown in Figure 1. It requires polar coordinates of angle, in radians, and radius. The **patch** command is used to “patch” or fill in 2-D polygons. It requires x and y data for the current axes. Viewing overlapping patches for comparison purposes requires setting the transparency value as defined in the MATLAB help “Transparency Properties”. Transparency values range between 1 (opaque) and 0 (clear). The **deg2rad** function is used to convert data in units of degrees to units of radians.

The initial plot is setup as follows.

```
% polar plot example with filled patches
clear all
figure(1)
clf

% orig plot (theta[radians],rho)
oxo = deg2rad([0 30 60 60 30 0]);
oyo = [18 18 18 6 6 6];
h = polar(oxo,oyo)
hold on

% use data from polar plot for patch
p = patch(get(h,'XData'),get(h,'YData'),'g')
```

The handle for the polar plot figure is saved and referenced by the **set** command to obtain the plotted data and then is used by the **patch** command to fill in the area outlined by the data (see the green patch in Figure 1).

Next, another patch will be plotted using a separate data set:

```
% plot another patch
ox = deg2rad([0:5:360]);
oy = 10+0*ox;
h1 = polar(ox,oy)

% convert polar data to rectangular coordinates for patch
p1 = patch(oy.*cos(ox),oy.*sin(ox),'r')
```

Note here that since data was not obtained from the polar plot data, the polar coordinates must be converted into rectangular coordinates for the `patch` command. Remember this as

$$x = r \cos(\theta)$$

$$y = r \sin(\theta)$$

and it is shown as the red patch in Figure 1. Using the `patch` command directly another transparent patch will be created in yellow as shown in Figure 1.

```
% plot another patch and set transparency
p2 = patch([-10 10 10 -10],[-10 10 -10 10],'y')
set(p2,'FaceAlpha',0.5)
```

Notice here also that the coordinates of the “butterfly” patch data is given directly in rectangular coordinates and it appears as such on the plot. Internally, the `polar` plot command converts the polar coordinates to rectangular coordinates.

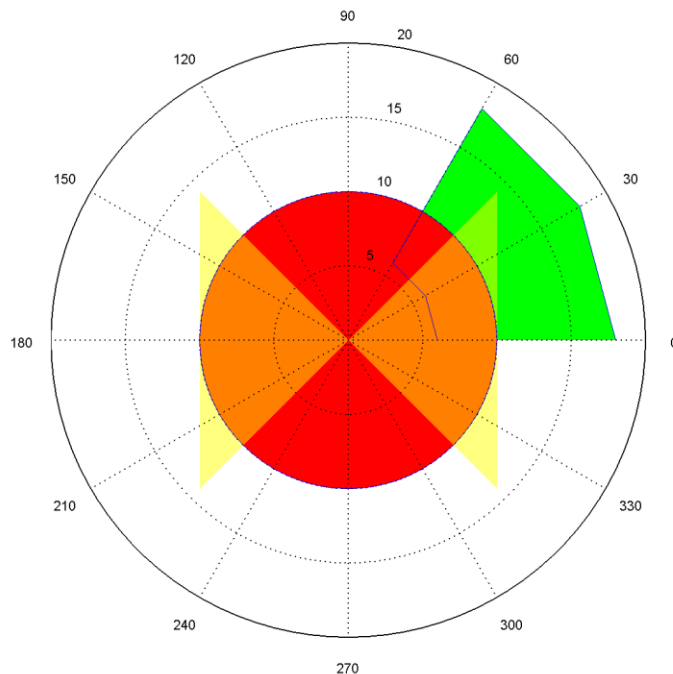


Figure 1 – Polar Plot Example

Portable Network Graphics (PNG) images of the plots can be saved using “`print('-dpng','[filename]')`” command.

REFERENCE

- [1] MATLAB (www.mathworks.com)
- [2] Reference Internal Project Misc/2-polarpatch

TN 12-1219-01

ABSORBED POWER FUNCTION USING MATLAB'S ODE45

December 19, 2013

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC)
Detroit Arsenal
Systems Integration & Engineering
Ground System Engineering & Assurance
ATTN: RDTA-SIE-EA/MS157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

The Motion Based Technologies Team requested assistance with ISO 17025 [1] certification of their seat comfort testing. Specific assistance was requested with the calculation of absorbed power.

PROBLEM SOLUTION:

Review of references [2-5] provides background to the calculation of absorbed power as a human vibration tolerance. A MATLAB [6] based function is provided in Appendix A.1. ("calcpwrfunc") to calculate absorbed using vertical accelerations. The function input is vertical acceleration in g's and the output is instantaneous vertical absorbed power.

Eight equations of state (EOS) are required. The ode45 solver [7], a variable step explicit Runge-Kutta method, is used to duplicate the Runge-Kutta-Gill method used in [9] which is considered the standard for absorbed power calculations. The interp1 function [8] is required to determine the time dependent vertical acceleration in EOS one. An array with time as the first column and vertical acceleration (in g's) as the second is passed into the function to achieve this.

```
% equations of state
eos = @(t,y) [-(29.80*y(1)+15.45*interp1(ver(:,1),ver(:,2),t))*32.2+100*y(2));
-10*(-y(1));
...];
```

The calling sequence for the calcpwrfunc is demonstrated below.

```
% Example function call

% define inputs
vt = t;      % time
v = y(:,3);  % vertical acceleration

% call function
[T Y] = calcpwrfunc([vt,v]);
```

The instantaneous absorbed power is obtained from the eighth EOS. Absorbed power is a time average so EOS #8 must be divided by time in order to achieve the correct result. Assuming stationary properties, the last value is usually defined as the average absorbed power for the input acceleration provided.

```
% Example results
plot(T,Y(:,8)./T,'g'); % average = Y(end,8)/T(end) (assumes stationary properties)
```

Comparison of output results between `calcpwrfunc` type functions from VEHDYNII [9], MATLAB [6], and the C programming language are shown in Figure 1.

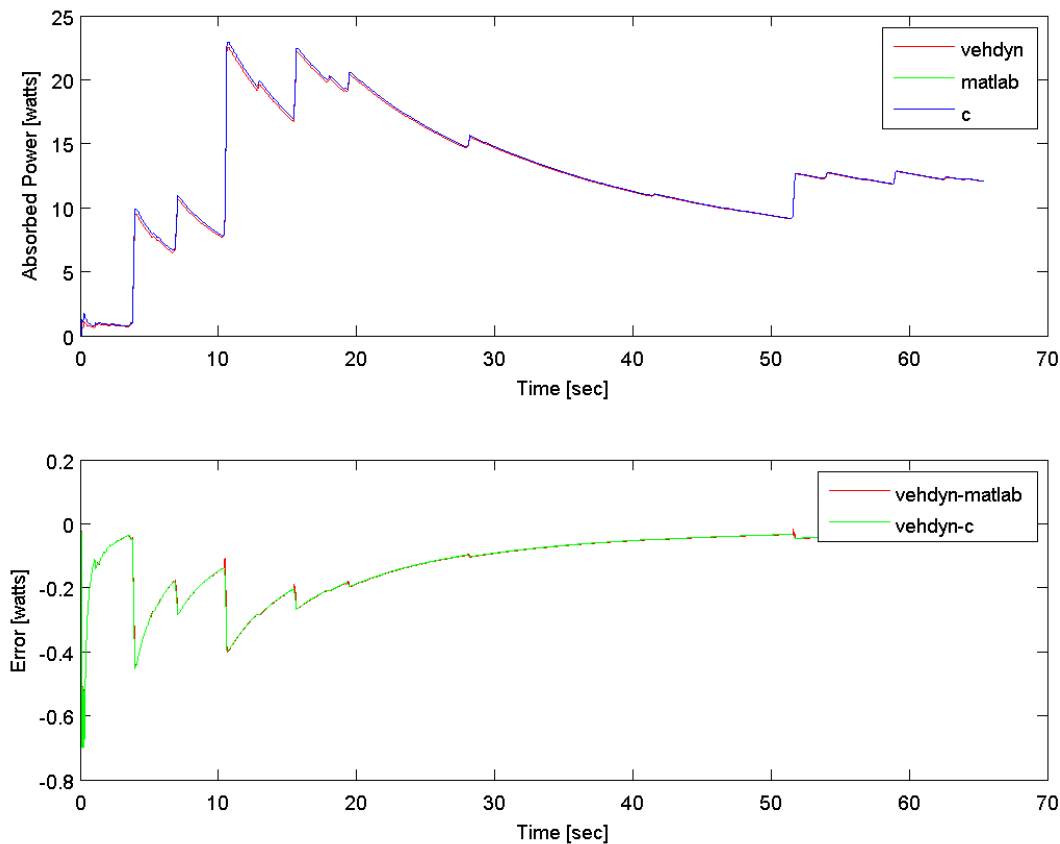


Figure 1 – Absorbed Power Calculation – Multiple Environments

The results indicate very good agreement, particularly as time advances. (Note that input/output from a VEHDYNII run was used to "verify" the accuracy of the absorbed power calculations. The data contained from this run was 1) time (sec), 2) unfiltered driver's acceleration (g's), 3) filtered driver's acceleration (g's), and 4) instantaneous absorbed power. The filtered driver's acceleration is based on a fourth-order/pole Butterworth filter at 30 Hz.)

Another short verification test is---after 30 seconds, an 8 ft/sec² sinusoidal acceleration at 4 Hz should output between 6.2 and 6.21 watts of absorbed power (t=30).

===

The state space definition is not unique. Referencing the vertical transfer function from [12] and Appendix A.3 of TN 10-1021-01 Absorbed Power Calculation [11] we can define several state space representations using the methods outlined in [13] and [14,15] that provide almost identical output as shown in Figure 1 from Appendix A.1. The MATLAB tf2ss (transfer function to state space) function [17] is similar to [15] with the states in reverse order. An example using direct Runge Kutta integration [16] is also included. These MATLAB based function are provided in the Appendix. For clarification, it is repeated that absorbed power is defined as the time average integral of the filtered acceleration squared, where $a(t)$ is acceleration in ft/sec^2 and $h(t)$ is the transfer function defined in the frequency domain in [12].

$$P_{AP} = \lim_{T \rightarrow \infty} \left(\frac{1}{T} \int_0^T [a(t) * h(t)]^2 dt \right)$$

The integral of the output squared (y^2) is defined as equation of state (EOS) #8 in the examples. The time average output above is then `"plot(T, Y(:,8) ./ T)"` for the output from this function (`[T Y] = ode45(eos, vert(:,1), zeros(8,1));`).

Figure 2 and Table 1 show the error of the examples compared to the Benchmark Vehdyn Output.

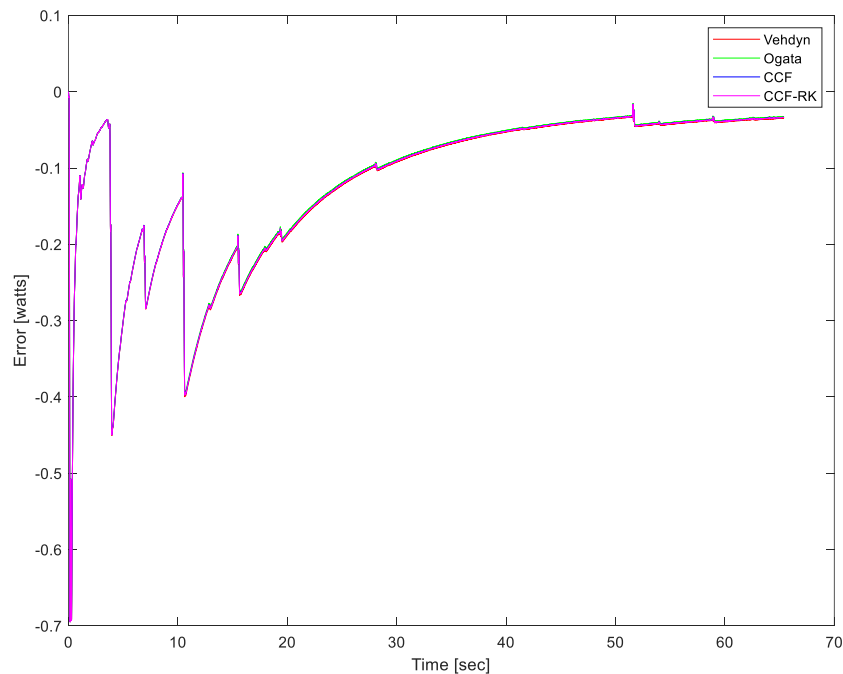


Figure 2 – Example Error Compared to Benchmark Vehdyn Output

Table 1 – Output Error Comparison

time	Benchmark Vehdyn Output	Vehdyn	Ogata	CCF	CCF-RK	Abs Pwr Units
65.404	12.0725	12.10723	12.10512	12.10622	12.10639	[watts]
	error	-0.03473	-0.03262	-0.03372	-0.03389	[watts]

REFERENCE

- [1] ISO/IEC 17025 General requirements for the competence of testing and calibration laboratories (http://en.wikipedia.org/wiki/ISO/IEC_17025)
 - [2] Test Operations Procedure (TOP) 01-1-014A Ride Dynamics and Evaluation of Human Exposure to Whole Body Vibration (3 Apr 2012). DIST A (DTIC AD A559 451). Approved for public release: distribution unlimited.
 - [3] "Overview of Human Vibration Weighting Filters", 5 February 2008. (<http://www.ni.com/white-paper/6957/en>)
 - [4] Rimell, Mansfield. "Design of Digital Filters for Frequency Weightings Required for Risk Assessments of Workers Exposed to Vibration", Industrial Health 2007, 45, 512–519, April 7, 2007 (http://www.google.com/url?sa=t&rct=j&q=digital+filters+industrial+health+2007&source=web&cd=2&cad=rja&ved=0CDIQFjAB&url=http%3A%2F%2Fwww.researchgate.net%2Fpublication%2F47450362_Design_of_digital_filters_for_frequency_weightings_required_for_risk_assessments_of_workers_exposed_to_vibration%2Ffile%2Fpdf%2F50928c8235dad.pdf&ei=pvCHUa6xKlBj4AP954CICQ&usg=AFQjCNE4WyGZQi4zE23jX6AEf0dVN3wZcQ)
 - [5] Svantek Whole-body vibration meter. (<http://svantek.com/multichannel-measurements/sv-100-whole-body-vibration-dosimeter-analyser.html#guide>)
 - [6] MATLAB (www.mathworks.com)
 - [7] ode45 (<http://www.mathworks.com/help/matlab/ref/ode45.html>)
 - [8] interp1 (<http://www.mathworks.com/help/matlab/ref/interp1.html>)
 - [9] Creighton, D. C. "Revised vehicle dynamics module: user's guide for computer program VEHDYN II". U.S. Army Engineer Waterways Experiment Station, Vicksburg, MS. 1986. TR SL-86-9. DIST C (DTIC AD B102 925).
 - [10] Reference Internal Project 12-517 – MATLAB Absorbed Power – ode45
- ===
- [11] TN 10-1021-01 Absorbed Power Calculation
 - [12] Lee and Lins. "Human Vibration Measuring Instrument". Army Tank-Automotive Command Warren, MI. 1973. DIST A (DTIC AD 0785 648). Approved for public release: distribution unlimited. (see pg. 207).
 - [13] Ogata. Modern Control Engineering. Prentice-Hall, 1970. (see 14-2 State-Space Representation of Systems, pg. 675-678)
 - [14] Transfer Function to State Space (<https://lpsa.swarthmore.edu/Representations/SysRepTransformations/TF2SS.html>)
 - [15] Controllable Canonical Form (https://en.wikipedia.org/wiki/State-space_representation)
 - [16] Runge Kutta Methods (https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta_methods)
 - [17] tf2ss (<https://www.mathworks.com/help/signal/ref/tf2ss.html>)

APPENDIX A.1– calcpwrfunc (VehdynII method)

APPENDIX A.2 – calcpwrfunc (Ogata method)

APPENDIX A.3 – calcpwrfunc (CCF method)

APPENDIX A.4 – calcpwrfunc (CCF method/Direct Runge Kutta Integration)

APPENDIX A.5 – poles/zeros of transfer function/filter

APPENDIX A.1 –calcpwrfunc

```

function [T,Y] = calcpwrfunc(vert)
%=====
%
% calcpwrfunc
%
% INPUT: VERTICAL ACCELERATION IN G'S, [time, acceleration]
% Example: [T Y] = calcpwrfunc([vt,v]);
%
% OUTPUT: INSTANTANEOUS VERTICAL ABSORBED POWER
% Example: plot(T,Y(:,8)./T);
%
%=====
%
% THIS PROGRAM COMPUTES THE VERTICAL (Z) AXIS ABSORBED POWER TO A HUMAN.
%
% ADDRESS QUESTIONS TO:
%
% U.S. ARMY RDECOM/TARDEC
% ATTN: RDTA-RS-AN, B215, MS157
% WARREN, MI 48397-5000
%
% REFERENCES:
% [1] Lee and Lins. Human Vibration Measuring Instrument.
% Army Tank-Automotive Command Warren, MI. 1973. DIST A
% (DTIC AD 0785 648). (see pg. 207)
%
%=====
%
% HISTORY:
% 1.0 1994-04-04
% BASE VERSION
%
%=====

% equations of state (EOS)
eos = @(t,y) [-(29.80*y(1)+15.45*interp1(vert(:,1),vert(:,2),t)*32.2+100*y(2));
-10*(-y(1));
-(736.9*(-y(1))+1000*(-(-y(1)+0.108*y(4)))));
-(100*(-y(1))+35.19*y(3)+39.1*y(4));
-(1000*(-(-y(1)+0.108*y(4)))+684.3*(-((-y(1)+0.108*y(4)))+0.25*y(6)))));
-(80*(-(-y(1)+0.108*y(4)))+30.28*y(5)+100*y(6));
-((-((-y(1)+0.108*y(4)))+0.25*y(6)))+6*y(7));
-((-((-y(1)+0.108*y(4)))+0.25*y(6))+y(7))^2];

% use Runge Kutta integration with ode45
[T Y] = ode45(eos,vert(:,1),zeros(8,1));

```

APPENDIX A.2 –calcpwrfunc (Ogata method)

```

function [T,Y] = calcpwrfunc(ver)
%=====
%
% calcpwrfunc
%
% INPUT: VERTICAL ACCELERATION IN G'S, [time, acceleration]
% Example: [T Y] = calcpwrfunc([vt,v]);
%
% OUTPUT: INSTANTANEOUS VERTICAL ABSORBED POWER
% Example: plot(T,Y(:,8)./T);
%
%=====
%
% THIS PROGRAM COMPUTES THE VERTICAL (Z) AXIS ABSORBED POWER TO A HUMAN.
%
% ADDRESS QUESTIONS TO:
%
% U.S. ARMY DEVCOM/GVSC
% ATTN: FCDD-GVS-MSP/MS 157
% WARREN, MI 48397-5000
%
% REFERENCES:
% [1] Lee and Lins. Human Vibration Measuring Instrument.
% Army Tank-Automotive Command Warren, MI. 1973. DIST A
% (DTIC AD 0785 648). (see pg. 207)
% [2] Ogata. Modern Control Engineering. Prentice-Hall, 1970.
% (see 14-2 State-Space Representation of Systems,
% pg. 675-678)
%
%=====
%
% define coefficients for vertical transfer function in [1] based on [2]
% Note that the absorbed power filter assumes the acceleration input is
% in ft/sec^2 so if in g's it must be converted (g*32.2 = ft/sec^2).
%
%==TRANSFER FUNCTION FROM [1]
%nl = [15.45 0];
%nl = [1 0]; % k0=15.45 multiplies the numerator later
%dl = [1 29.8 1000];

n2 = [1 28.3 2800];
%d2 = [1 39.1 3800];

%nl = [1 105 7570]; % orig
%nl = [1 80 7570]; % vehdynii
%dl = [1 125 5180]; % orig
%dl = [1 100 5180]; % vehdynii

n4 = [1 5];
%d4 = [1 6];

n5 = conv(n4,conv(n3,conv(n1,n2)));
d5 = conv(d4,conv(d3,conv(d1,d2)));

% print transfer function
sys = tf(n5,d5)

% number of states
n = length(d5) - 1;

%b = flip(n5(end-[0:n-1]));
%a = flip(d5(end-[0:n-1]));
%b = n5(end-(n-1):end);
%a = d5(end-(n-1):end);
fprintf(['\n', repmat('%12.2f', '1',length(b)-1), '%12.2f\n', '\n'],b);
fprintf(['\n', repmat('%12.2f', '1',length(a)-1), '%12.2f\n', '\n'],a);

%sys =
%
% s^6 + 113.3 s^5 + 1.318e04 s^4 + 501401 s^3 + 2.339e07 s^2 + 1.06e08 s
% -----
% s^7 + 174.9 s^6 + 1.905e04 s^5 + 1.214e06 s^4 + 5.657e07 s^3 + 1.469e09 s^2 + 2.67e10 s + 1.181e11
%
% b=
% a=
% 1.00, 113.30, 13175.50, 501401.00, 23387155.00, 105980000.00, 0.00
% 174.90, 19048.58, 1213971.08, 56568192.40, 1468722994.40, 26698727200.00, 118104000000.00

%--- STATE SPACE REPRESENTATION COEFFICIENTS FROM [2]---%
k0=15.45;

% b7 here is equivalent to b0 in [2] (b0 ==> b7, b1 ==> b6, etc.)
b7=0.0;
b6=1.0;
b5=113.3;
b4=13175.5;
b3=501401.0;
b2=23387155.0;
b1=105980000.0;
b0=0.0;

```

```

% a0 here is equivalent to a7 in [2] (a0 ==> a7, a1 ==> a6, etc.)
a8=0.0;
a7=1.0;
a6=174.9;
a5=19048.58;
a4=1213971.08;
a3=56568192.4;
a2=1468722994.4;
a1=26698727200.0;
a0=118104000000.0;

% Qn here is equivalent to Bn (beta n) in [2]
Q0=b7;
Q1=b6-a6*Q0;
Q2=b5-a6*Q1-a5*Q0;
Q3=b4-a6*Q2-a5*Q1-a4*Q0;
Q4=b3-a6*Q3-a5*Q2-a4*Q1-a3*Q0;
Q5=b2-a6*Q4-a5*Q3-a4*Q2-a3*Q1-a2*Q0;
Q6=b1-a6*Q5-a5*Q4-a4*Q3-a3*Q2-a2*Q1-a1*Q0;
Q7=b0-a6*Q6-a5*Q5-a4*Q4-a3*Q3-a2*Q2-a1*Q1-a0*Q0;

% xdot = Ax + Bu
% y = Cx + Du
%
% where
% A = [ 0 1 0 0 0 0 0;
%       0 0 1 0 0 0 0;
%       0 0 0 1 0 0 0;
%       0 0 0 0 1 0 0;
%       0 0 0 0 0 1 0;
%       0 0 0 0 0 0 1;
%       -a0 -a1 -a2 -a3 -a4 -a5 -a6];
%
% B = k0*[Q1; Q2; Q3; Q4; Q5; Q6; Q7];
%
% C = [ 1 0 0 0 0 0 0];
%
% D = k0*[Q0];
%
% u = interp1(verte(:,1),verte(:,2),t)*32.2;
%
%=====
% equations of state (EOS)
eos = @(t,y) [ y(2) + Q1*k0*interp1(verte(:,1),verte(:,2),t)*32.2;
y(3) + Q2*k0*interp1(verte(:,1),verte(:,2),t)*32.2;
y(4) + Q3*k0*interp1(verte(:,1),verte(:,2),t)*32.2;
y(5) + Q4*k0*interp1(verte(:,1),verte(:,2),t)*32.2;
y(6) + Q5*k0*interp1(verte(:,1),verte(:,2),t)*32.2;
y(7) + Q6*k0*interp1(verte(:,1),verte(:,2),t)*32.2;
-a0*y(1) - a1*y(2) - a2*y(3) - a3*y(4) - a4*y(5) - a5*y(6) - a6*y(7) ...
+ Q7*k0*interp1(verte(:,1),verte(:,2),t)*32.2;
(y(1) + Q0*k0*interp1(verte(:,1),verte(:,2),t)*32.2)^2];

% NOTE: Absorbed power is defined as the time average integral of the
% filtered acceleration squared, so the integral of the square is
% defined as integral(y^2) in EOS #8. The time average output will
% then be plot(T,Y(:,8)./T) for the output from this function;

[T Y] = ode45(eos,verte(:,1),zeros(8,1));

```

APPENDIX A.3 –calcpwrfunc (CCF method)

```

function [T,Y] = calcpwrfunc(ver)
%=====
%
% calcpwrfunc
%
% INPUT: VERTICAL ACCELERATION IN G'S, [time, acceleration]
% Example: [T Y] = calcpwrfunc([vt,v]);
%
% OUTPUT: INSTANTANEOUS VERTICAL ABSORBED POWER
% Example: plot(T,Y(:,8)./T);
%
%=====
%
% THIS PROGRAM COMPUTES THE VERTICAL (Z) AXIS ABSORBED POWER TO A HUMAN.
%
% ADDRESS QUESTIONS TO:
%
% U.S. ARMY DEVCOM/GVSC
% ATTN: FCDD-GVS-MSP/MS 157
% WARREN, MI 48397-5000
%
% REFERENCES:
% [1] Lee and Lins. fHuman Vibration Measuring Instrument.
% Army Tank-Automotive Command Warren, MI. 1973. DIST A
% (DTIC AD 0785 648). (see pg. 207)
% [2] Controllable Canonical Form (https://en.wikipedia.org/wiki/State-space\_representation)
%
%=====
%
% define coefficients for vertical transfer function in [1] based on [2]
% Note that the absorbed power filter assumes the acceleration input is
% in ft/sec^2 so if in g's it must be converted (g*32.2 = ft/sec^2).
%
%===TRANSFER FUNCTION FROM [1]
n1 = [15.45 0];
d1 = [1 29.8 1000];

n2 = [1 28.3 2800];
d2 = [1 39.1 3800];

n3 = [1 105 7570]; % orig
n3 = [1 80 7570]; % vehdynii
d3 = [1 125 5180]; % orig
d3 = [1 100 5180]; % vehdynii

n4 = [1 5];
d4 = [1 6];

n5 = conv(n4,conv(n3,conv(n1,n2)));
d5 = conv(d4,conv(d3,conv(d1,d2)));

% print transfer function
sys = tf(n5,d5)

% number of states
n = length(d5) - 1;

%b = flip(n5(end-[0:n-1]));
%a = flip(d5(end-[0:n-1]));
b = n5(end-(n-1):end);
a = d5(end-(n-1):end);
fprintf(['', repmat('%12.2f', '1,length(b)-1', '%12.2f\n', '')],b);
fprintf(['', repmat('%12.2f', '1,length(a)-1', '%12.2f\n', '')],a);

% original coefficients
% b=      15.45,      2136.74,      216423.60,      8882799.82, 366739044.75, 1637391000.00,      0.00
% a=      199.90,      20921.08,      1373435.58,      61271469.40, 1586573994.40, 27268727200.00, 118104000000.00

%sys =
%
%      15.45 s^6 + 1750 s^5 + 2.036e05 s^4 + 7.747e06 s^3 + 3.613e08 s^2 + 1.637e09 s
% -----
% s^7 + 174.9 s^6 + 1.905e04 s^5 + 1.214e06 s^4 + 5.657e07 s^3 + 1.469e09 s^2 + 2.67e10 s + 1.181e11
%
% b=      15.45,      1750.49,      203561.47,      7746645.45, 361331544.75, 1637391000.00,      0.00
% a=      174.90,      19048.58,      1213971.08,      56568192.40, 1468722994.40, 26698727200.00, 118104000000.00

%--- CCF STATE SPACE REPRESENTATION FROM [2]---%

% xdot = Ax + Bu
% y = Cx + Du
%
% xdot(end) = y^2
%
% where
% A = [ 0 1 0 0 0 0 0;
%       0 0 1 0 0 0 0;
%       0 0 0 1 0 0 0;
%       0 0 0 0 1 0 0;
%       0 0 0 0 0 1 0;
%       0 0 0 0 0 0 1];

```

```

        0 0 0 0 0 0 1;
        flip(-a)];
%
B = [0; 0; 0; 0; 0; 0; 0; 1];
%
C = [ flip(b)];
%
D = [0];
%
u = interp1(ver(:,1),ver(:,2),t)*32.2;
%

%=====

% equations of state (EOS)
% eos = @(t,y) [ y(2);
% ...
% -a7*y(1) - a6*y(2) ... + interp1(ver(:,1),ver(:,2),t)*32.2;
% (-a7*y(1) - a6*y(2) ... + interp1(ver(:,1),ver(:,2),t)*32.2)^2];
% [T Y] = ode45(@(t,x) eos,ver(:,1),zeros(8,1));

% using eos function
[T Y] = ode45(@(t,x) odefcn(t,x,n,A,B,C,ver),ver(:,1),zeros(8,1));

% NOTE: Absorbed power is defined as the time average integral of the
% filtered acceleration squared, so the integral of the square is
% defined as integral(y^2) in EOS #8. The time average output will
% then be plot(T,Y(:,8)./T) for the output from this function;

function dx = odefcn(t,x,n,A,B,C,ver)
    dx = zeros(n+1,1);
    dx = [ A*x(1:n) + B*interp1(ver(:,1),ver(:,2),t)*32.2;
          (C*x(1:n))^2 ];
end

end

```

APPENDIX A.4 –calcpwrfunc (CCF method/Direct Runge Kutta Integration)

```

function [T,Y] = calcpwrfunc(ver)
%=====
%
% calcpwrfunc
%
% INPUT: VERTICAL ACCELERATION IN G'S, [time, acceleration]
% Example: [T Y] = calcpwrfunc([vt,v]);
%
% OUTPUT: INSTANTANEOUS VERTICAL ABSORBED POWER
% Example: plot(T,Y(:,8)./T);
%
%=====
%
% THIS PROGRAM COMPUTES THE VERTICAL (Z) AXIS ABSORBED POWER TO A HUMAN.
%
% ADDRESS QUESTIONS TO:
%
% U.S. ARMY DEVCOM/GVSC
% ATTN: FCDD-GVS-MSP/MS 157
% WARREN, MI 48397-5000
%
% REFERENCES:
% [1] Lee and Lins. Human Vibration Measuring InstrumentY.
% Army Tank-Automotive Command Warren, MI. 1973. DIST A
% (DTIC AD 0785 648). (see pg. 207)
% [2] Controllable Canonical Form (https://en.wikipedia.org/wiki/State-space\_representation)
% [3] Runge Kutta Integration Methods (https://en.wikipedia.org/wiki/Runge%E2%80%93Kutta\_methods)
%
%=====
%
% define coefficients for vertical transfer function in [1] based on [2]
% Note that the absorbed power filter assumes the acceleration input is
% in ft/sec^2 so if in g's it must be converted (g*32.2 = ft/sec^2).
%

%===TRANSFER FUNCTION FROM [1]
n1 = [15.45 0];
d1 = [1 29.8 1000];

n2 = [1 28.3 2800];
d2 = [1 39.1 3800];

n3 = [1 105 7570]; % orig
n3 = [1 80 7570]; % vehdynii
d3 = [1 125 5180]; % orig
d3 = [1 100 5180]; % vehdynii

n4 = [1 5];
d4 = [1 6];

n5 = conv(n4,conv(n3,conv(n1,n2)));
d5 = conv(d4,conv(d3,conv(d1,d2)));

% print transfer function
sys = tf(n5,d5)

% number of states
n = length(d5) - 1;

%b = flip(n5(end-[0:n-1]));
%a = flip(d5(end-[0:n-1]));
b = n5(end-(n-1):end);
a = d5(end-(n-1):end);
fprintf(['', repmat('%12.2f', '1,length(b)-1', '%12.2f\n', '')],b);
fprintf(['', repmat('%12.2f', '1,length(a)-1', '%12.2f\n', '')],a);

% original coefficients
% b= 15.45, 2136.74, 216423.60, 8882799.82, 366739044.75, 1637391000.00, 0.00
% a= 199.90, 20921.08, 1373435.58, 61271469.40, 1586573994.40, 27268727200.00, 118104000000.00

%sys =
%
% 15.45 s^6 + 1750 s^5 + 2.036e05 s^4 + 7.747e06 s^3 + 3.613e08 s^2 + 1.637e09 s
% -----
% s^7 + 174.9 s^6 + 1.905e04 s^5 + 1.214e06 s^4 + 5.657e07 s^3 + 1.469e09 s^2 + 2.67e10 s + 1.181e11
%
% b= 15.45, 1750.49, 203561.47, 7746645.45, 361331544.75, 1637391000.00, 0.00
% a= 174.90, 19048.58, 1213971.08, 56568192.40, 1468722994.40, 26698727200.00, 118104000000.00

%--- CCF STATE SPACE REPRESENTATION FROM [2]---%

% xdot = Ax + Bu
% y = Cx + Du
%
% xdot(end) = y^2
%
% where
A = [ 0 1 0 0 0 0 0;
      0 0 1 0 0 0 0;
      0 0 0 1 0 0 0;
      0 0 0 0 1 0 0;
      0 0 0 0 0 1 0];

```



```

        0 0 0 0 0 1 0;
        0 0 0 0 0 0 1;
        flip(-a)];
%
    B = [0; 0; 0; 0; 0; 0; 0; 1];
%
    C = [ flip(b)];
%
    D = [0];
%
    u = interp1(ver(:,1),ver(:,2),t)*32.2;
%
%=====

% Runge Kutta Integration from [3]

% initialize variables
x = zeros(1,n+1); % last state to integrate y^2 for avg abs pwr
Y = [x]; % state output vector
h = ver(2,1) - ver(1,1); % time increment
t = 0; % initial time
T = [0]; % time output vector

% loop on input (filtered acceleration)
for i = 2:length(ver(:,2)),

    k1 = odefcn(t,x',n,A,B,C,ver);
    k2 = odefcn(t+h/2,x'+h*k1/2,n,A,B,C,ver);
    k3 = odefcn(t+h/2,x'+h*k2/2,n,A,B,C,ver);
    k4 = odefcn(t+h,x'+h*k3,n,A,B,C,ver);

    xd = h/6*(k1' + 2*k2' + 2*k3' + k4');
    x = x + xd;
    t = t + h;

    T = [T;t];
    Y = [Y;x];
    %fprintf('time %12.5f %12.5f\n',t,x(end));
end

% NOTE: Absorbed power is defined as the time average integral of the
% filtered acceleration squared, so the integral of the square is
% defined as integral(y^2) in EOS #8. The time average output will
% then be plot(T,Y(:,8)./T) for the output from this function;

function dx = odefcn(t,x,n,A,B,C,ver)
    dx = zeros(n+1,1);
    dx = [ A*x(1:n) + B*interp1(ver(:,1),ver(:,2),t)*32.2;
          (C*x(1:n))^2 ];
end
end

```

APPENDIX A.5 – poles/zeros of transfer function/filter

```

% zp.m
%%% zeros and poles
clear

%%% vehdyn state space
A = [-29.80 -100 0 0 0 0 0;
     10 0 0 0 0 0;
     -263.1 0 0 108 0 0 0;
     100 0 -35.19 -39.1 0 0 0;
     -315.7 0 0 34.0956 0 171.075 0;
     -80 0 0 8.64 -30.28 -100 0;
     1 0 0 -0.108 0 .25 -6];

B = -32.2*[15.45; 0; 0; 0; 0; 0; 0];
B = [15.45; 0; 0; 0; 0; 0; 0];
C = [1 0 0 -0.108 0 .25 -1];
D = [0];

%%% zeros/poles of transfer function
[num,den] = ss2tf(A,B,C,D,1);
sys = tf(num,den);
xx = zero(sys);
prtc('zeros trf-vehdyn',xx);
xx = pole(sys);
prtc('poles trf-vehdyn',xx);

%%% Lee and Lins transfer function
n1 = [15.45 0];
d1 = [1 29.8 1000];

n2 = [1 28.3 2800];
d2 = [1 39.1 3800];

%n3 = [1 105 7570]; % orig
n3 = [1 80 7570]; % vehdyn
%d3 = [1 125 5180]; % orig
d3 = [1 100 5180]; % vehdyn

n4 = [1 5];
d4 = [1 6];

n5 = conv(n4,conv(n3,conv(n1,n2)));
d5 = conv(d4,conv(d3,conv(d1,d2)));

%%% roots of num/den polynomials
%xx = roots(n5);
%prtc('num roots-lee/lins',xx);
%xx = roots(d5);
%prtc('den roots-lee/lins',xx);

%%% zeros/poles of transfer function
sys = tf(n5,d5);
xx = zero(sys);
prtc('zeros trf-lee/lins',xx);
xx = pole(sys);
prtc('poles trf-lee/lins',xx);

function prtc(s,a)
% print array of roots/zeros/poles and complex conjugate form s^2 + (n1)s + (n2)
% s title
% a array of roots/zeros/poles
fprintf('%s : complex conj form s^2 + (n1)s + (n2)\n',s);
fprintf('%12.2f %12.2f : %12.2f %12.2f %12.2f\n',[real(a), imag(a), ones(length(a),1), -2*real(a), real(a).^2+imag(a).^2]);
end

```

```

>> zp
zeros trf-vehdyn : complex conj form s^2 + (n1)s + (n2)
    0.00      0.00 :      1.00      -0.00      0.00
   -40.00     77.27 :      1.00     80.00     7570.00
   -40.00    -77.27 :      1.00     80.00     7570.00
   -14.15     50.99 :      1.00     28.30     2800.60
   -14.15    -50.99 :      1.00     28.30     2800.60
    -5.00      0.00 :      1.00     10.00      25.00
poles trf-vehdyn : complex conj form s^2 + (n1)s + (n2)
   -50.00     51.77 :      1.00     100.00     5180.15
   -50.00    -51.77 :      1.00     100.00     5180.15
   -19.55     58.47 :      1.00     39.10     3800.52
   -19.55    -58.47 :      1.00     39.10     3800.52
   -14.90     27.89 :      1.00     29.80     1000.00
   -14.90    -27.89 :      1.00     29.80     1000.00
    -6.00      0.00 :      1.00     12.00      36.00
zeros trf-lee/iins : complex conj form s^2 + (n1)s + (n2)
    0.00      0.00 :      1.00      -0.00      0.00
   -40.00     77.27 :      1.00     80.00     7570.00
   -40.00    -77.27 :      1.00     80.00     7570.00
   -14.15     50.99 :      1.00     28.30     2800.00
   -14.15    -50.99 :      1.00     28.30     2800.00
    -5.00      0.00 :      1.00     10.00      25.00
poles trf-lee/iins : complex conj form s^2 + (n1)s + (n2)
   -50.00     51.77 :      1.00     100.00     5180.00
   -50.00    -51.77 :      1.00     100.00     5180.00
   -19.55     58.46 :      1.00     39.10     3800.00
   -19.55    -58.46 :      1.00     39.10     3800.00
   -14.90     27.89 :      1.00     29.80     1000.00
   -14.90    -27.89 :      1.00     29.80     1000.00
    -6.00      0.00 :      1.00     12.00      36.00
>>

```

NOTE: For complex conjugate roots $(s - c)(s - c^*) = s^2 - (c + c^*)s + cc^*$ where $c = a + ib$, $c + c^* = 2a$, and $cc^* = a^2 + b^2$ thereby giving

$$s^2 - (c + c^*)s + cc^* \rightarrow s^2 - (2a)s + (a^2 + b^2).$$

This definition can help verify the filter coefficients for the transfer function defined in [12].

TN 13-0528-01

DETRENDING COURSE ELEVATION DATA

May 28, 2013

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC)
Detroit Arsenal
Systems Integration & Engineering
Ground System Engineering & Assurance
ATTN: RDTA-SIE-EA/MS157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

Track and Suspension requested information on how to calculate terrain roughness (root-mean-square (RMS)) and detrend measured test site courses.

PROBLEM SOLUTION:

There are many ways one can "detrend" terrains, but generally the Army test community uses a moving two-sided exponentially weighted average technique outlined in

Test Operations Procedure (TOP) 1-1-010 Vehicle Test Course Severity, Aberdeen Proving Ground, October 2006. Approved for public release.

Fix, G.A., TARADCOM Signal Analysis Program, September 1978 (DTIC ADA 072 014, TR No. 12412). Approved for public release.

that can be traced back to [1]. See Appendix A.1 for details of TOP 1-1-010.

REFERENCE

- [1] Van Deusen, B.D. "A Statistical Technique for the Dynamic Analysis of Vehicles Traversing Rough Yielding and Non-Yielding Surfaces". NASA Report No. CR-659, March 1967.
- [2] TN 10-0729-01 Exponential Course Detrending

APPENDIX A.1 – Pages 10-11 from Test Operations Procedure (TOP) 1-1-010 Vehicle Test Course Severity, October 2006. Approved for public release.

TOP 1-1-010
31 October 2006

$$y_{mp} = \sum \sin(\theta_i) dl$$

$$x_{mp} = \sum \cos(\theta_i) dl$$

where θ is the pitch angle and dl is the increment of travel along the surface.

The ultrasonic sensors are at a height:

$$y'_L = y_{mp} + w/2 \sin(\phi)$$

$$y'_R = y_{mp} - w/2 \sin(\phi)$$

where $w/2$ is the distance from the midpoint to each sensor and ϕ is the roll angle of the chassis.

Finally, the ultrasonic subsystem component is added to yield

$$y_L = \sum \sin(\theta_i) dl + w/2 \sin(\phi) + U_L$$

$$y_R = \sum \sin(\theta_i) dl - w/2 \sin(\phi) + U_R$$

where U_L and U_R are the distance measured by the left and right ultrasonic sensors, respectively.

c. The profiles obtained represent values obtained at a nonuniform sampling interval. Linear interpolation techniques are utilized to obtain samples at equal intervals of horizontal distance as required by the spectral transformation. A digital filtering technique is also employed to account for the non-unity transfer function induced by the profilometer geometry. A moving two-sided exponentially weighted average technique (detrending) is utilized to remove long wavelength (low spatial frequency) data created by the integration process used in the above equations. This is described in reference no. 7 and is shown mathematically as:

$$y_n(x) = \frac{\sum \{y(x + l\delta x) + y(x - l\delta x)\} e^{-l\delta x/\gamma}}{2 \sum e^{-l\delta x/\gamma}} \quad (\text{sum from } l = 0 \text{ to } E)$$

where:

- $y(x)$ = elevation at point x .
- $y_n(x)$ = correction factor.
- l = step number.
- δx = measurement interval (0.25 ft).
- γ = weighting constant.
- E = limit of summation.

TOP 1-1-010
31 October 2006

$$y_d(x) = y(x) - y_a(x)$$

where:

$y_d(x)$ = Detrended elevation at point x.

d. This filtering process does not discriminate between low frequency data created as an artifact of the integration process (due to offsets in amplifiers and digitizers, etc.) and legitimate long wavelength data such as hills. Thus, the output of the profilometer is useful for describing surface roughness for wavelengths up to approximately 30 m (100 ft) (depending upon the detrending routine coefficients), and is inappropriate for measuring slopes. It is generally accepted among the terrain roughness community that wavelengths beyond 18 m (60 ft) have little effect on vehicle dynamics, and are ignored.

5. WAVE NUMBER SPECTRUM ANALYSIS OF DTC TEST COURSES. The various endurance test courses at APG are regularly measured for their surface roughness using the profilometer system. Before these measurements were made, the profilometer system was calibrated on a fixed course with known characteristics. A single pass of the vehicle along the center of each test course is made monthly to obtain a characteristic wave number spectrum. This single pass is considered sufficient since the wave number spectra curves of several passes across the width of a course have shown reproducibility to within 3 to 5 dB of each other (except for shoulder ruts). The courses measured and the resulting wave number spectra are described in the following paragraphs by test center. These curves are plotted in terms of power per unit spatial frequency versus spatial frequency. Hence, the units will be $\text{ft}^2/\text{cycle per ft} = \text{ft}^3$ for the ordinate and cycles per ft = ft^{-1} for the abscissa. These spectra are general characterizations of the surface roughness of each test course and should be considered typical for that course. Applications that require actual surface roughness at the time of a test (e.g., model validation, ride quality) should conduct a terrain measurement at the time of the test. The wave number spectra for each course are presented in the following paragraphs. Sufficient data exists for most ATC courses such that percentile spectra (10th and 90th) were computed in addition to the average spectrum. The ± 3 dB bands around the average spectrum are also shown for each course. These bands are significant because they correspond to spectral control tolerance bands for closed-loop laboratory vibration tests. Thus, control of test course surface roughness (with some exception on the smoother courses where the excursion above or below the tolerance bands represents a difference of a few hundredths of an inch) is maintained at a level similar to that of a laboratory vibration test. The data for the YTC courses is based on a single sample, and bounds cannot be determined.

TN 13-1113-01

TRAILING ARM SUSPENSION STATIC ANGLE SOLUTION USING MICROSOFT EXCEL

November 13, 2013

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC)
Detroit Arsenal
Systems Integration & Engineering
Ground System Engineering & Assurance
ATTN: RDTA-SIE-EA/MS157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

Given torsion bar index angles and wheel station loads how do you determine the static angle of each trailing arm suspension unit assuming a suspension stiffness with weight and dimensional data provided?

PROBLEM SOLUTION:

A single trailing arm suspension model configuration is shown in Figure 1 (rw=road wheel, ra=road arm).

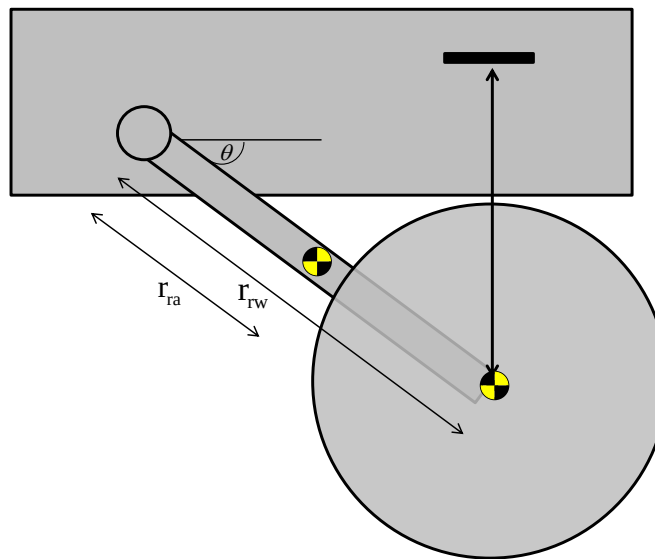


Figure 1 – Trailing Arm Suspension Configuration

A free body diagram of the forces at the torsion bar and road wheel revolute joints are shown in Figures 2 and 3.

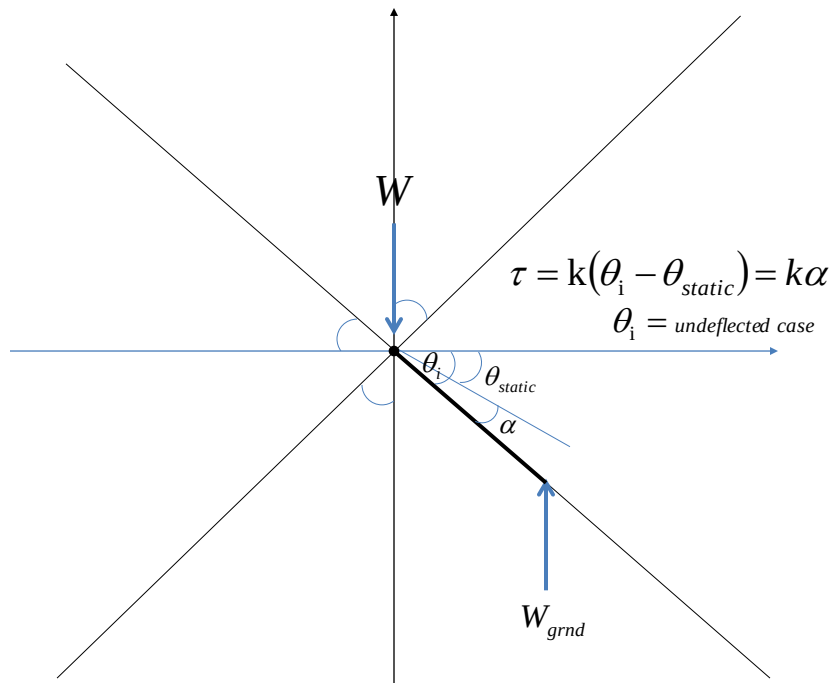


Figure 2 – Torsion Bar Revolute Joint Forces

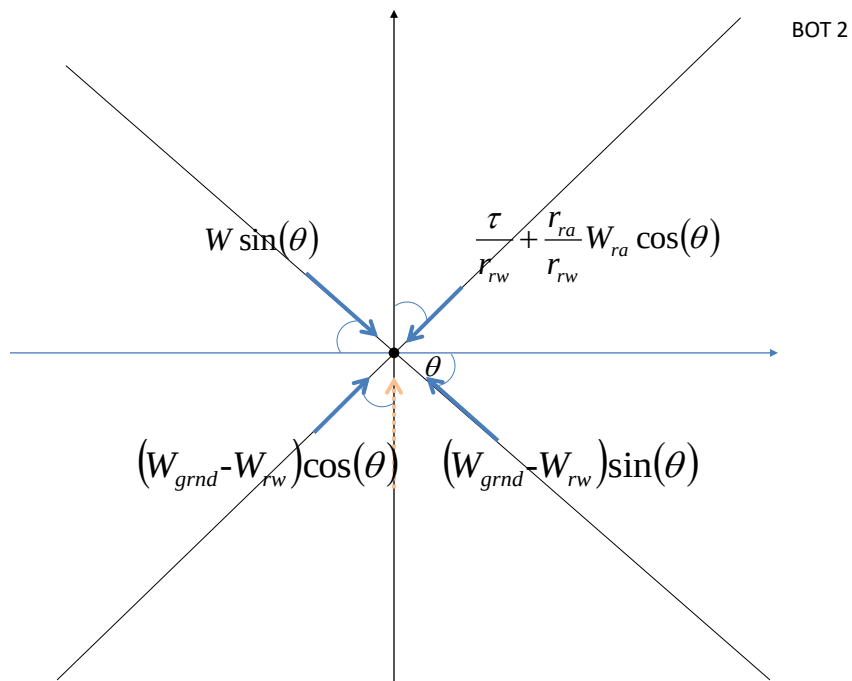


Figure 3 – Road Wheel Revolute Joint Forces (theta = theta static as in Figure 2)

The forces in Figure 3 must sum to zero which gives the nonlinear equation for theta as

$$(W_{grnd} - W_{rw}) \cos(\theta) = \left(\frac{\tau}{r_{rw}} + \frac{r_{ra}}{r_{rw}} W_{ra} \cos(\theta) \right), \tau = f(\theta_i, \theta) \quad (1)$$

then defining the error (which should be zero) as

$$err^2 = \left[(W_{rw} - W_{grnd}) \cos(\theta) + \frac{\tau}{r_{rw}} + \frac{r_{ra}}{r_{rw}} W_{ra} \cos(\theta) \right]^2 \quad (2)$$

the Microsoft Office Excel Solver tool can be used to optimize the error to zero and thus solve for theta. The solver uses the Generalized Reduced Gradient (GRG2) nonlinear optimization code, which was developed by Leon Lasdon, University of Texas at Austin, and Alan Waren, Cleveland State University [1].

Table 1 defines example suspension data used for the optimization to find theta for each roadwheel station. The index angles were obtained from initial baseline static load and static angle data. The static angles here are being calculated to avoid discrepancies between the measured loads and measured static angles. The static load measurements are assumed to be more accurate. Consistency in model behavior is important for consistent predictions that may be used in other models (e.g. load prediction for fatigue analysis). The example static loads defined in Table 1 are from a second configuration (baseline with additional armor weight). The individual squared error terms are summed and that sum value is minimized using the solver (see the cell next to "Total") with (W=weight, R=radius, rw=roadwheel, ra=roadarm, sf=scale factor, k=spring stiffness).

Table 1 – Suspension Data for Optimization

63	F	G	H	I	J	K	L
64	Index	Static Loads		Static Angles		Error	
65	Angles	lbf	lbf	deg	deg	Left	Right
66	deg	Left	Right	Left	Right		
67	39	4800	4600	20.0	21.0	8.35E-08	5.19E-09
68	55	6500	7300	31.4	27.3	1.59E-06	3.90E-07
69	49	6200	6800	25.2	22.2	7.21E-07	2.16E-07
70	41	4000	4500	26.0	23.7	6.69E-06	2.91E-08
71	40	3800	3800	25.8	25.8	6.69E-06	9.95E-08
72	35	2800	2200	24.6	27.2	5.11E-08	1.10E-08
73		Weight	57300			Total	1.66E-05
74							
75	Reference:						
76	Wrw	154	lbf				
77	Wra	54	lbf				
78	Rrw	16.5	in				
79	Rra	10.859	in				
80	sf	1.9					
81	k	228.219139	lbf/deg	= (7154.67lbf-in/deg)/sf/Rrw			

The reference weight and dimensional values were provided or estimated for the vehicle. The torsional stiffness rate (7154.67 lbf-in/deg) was estimated from standard models based on material type (steel, etc.),

diameter, and length, etc. Comparison between measured torque and displacement required the rate to be scaled by (1/1.9) to accurately predict one from the other.

In spreadsheet format equation (2) is defined as

$$=(\$G\$76-G67)*\text{COS}(\text{RADIANS}(I67)) + (\$F67-I67)*\$G\$81 + (\$G\$79/\$G\$78)*\$G\$77*\text{COS}(\text{RADIANS}(I67)))^2 \quad (3)$$

for the first left error with the target cell defined as

$$=\text{SUM}(K67:L72) \quad (4)$$

that is selected in the solver parameter dialog as shown in Figure 4.

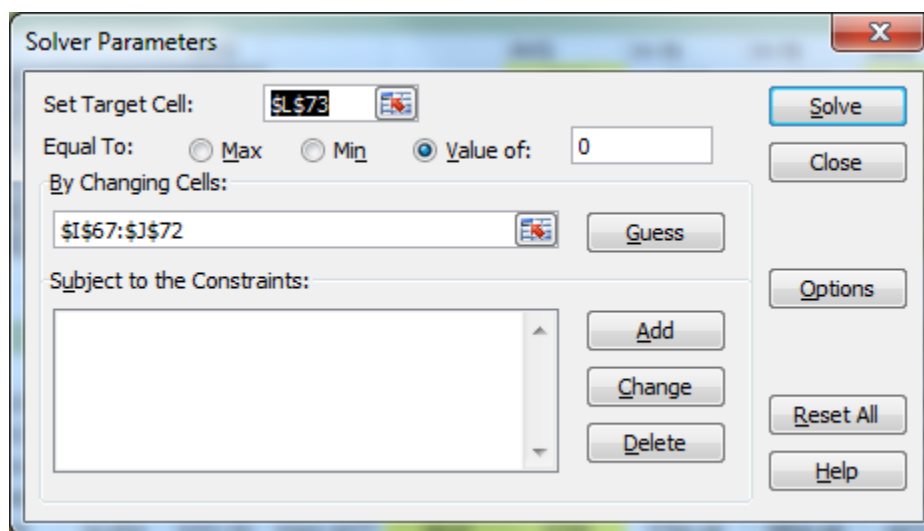


Figure 4 – Microsoft Excel Solver Parameters Dialog

The yellow area in Table 1 are the cells to be changed (theta values) that when the sum of all the squared errors is optimized to zero should result in the correct static angles for each trailing arm suspension corresponding to the static load provided.

Neglecting the weight of the roadarm (Wra with CG at Rra) changes the static angle approximately +/- 0.2 degrees.

REFERENCE

- [1] Frontline Systems, Inc., P.O. Box 4288, Incline Village, NV 89450-4288, (775) 831-0300. ("http://www.solver.com", e-mail: info@solver.com).

TN 15-0304-01

SIMULATING CHOCKS USING 2D ROAD PROFILES

March 4, 2015

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC)
Detroit Arsenal
Systems Integration & Engineering
ATTN: RDTA-SIE/MS157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

Abrams (M1) tank transportability rail impact analysis required the modeling of chocks as part of the dynamics analysis to predict the effects on chain tension [1].

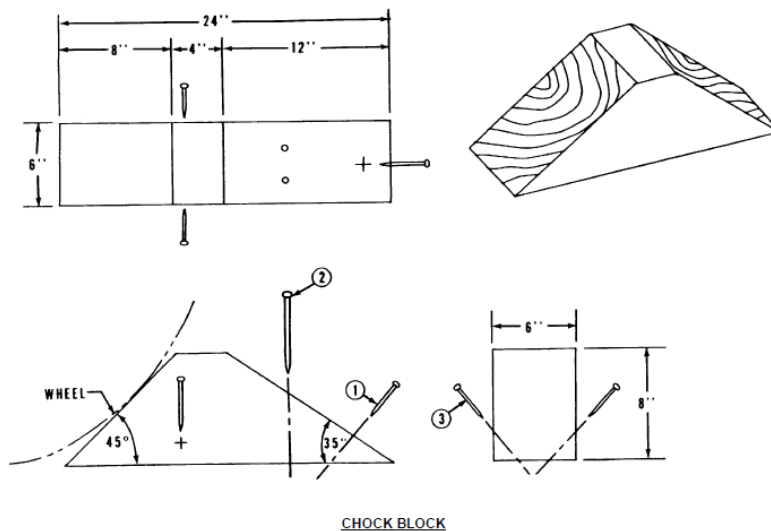


Figure 1 – Chock Block [2, App. O]

PROBLEM SOLUTION:

Chocks were implemented in a multibody dynamics model of the M1 (without a track) as a modification to the 2D road profile with respect to the foremost and rearmost road wheels. The road wheels were modeled with tire elements to link with the road profile using Siemens/LMS Virtual.Lab Motion [4].

Figure 2 – Spline Curve Dialog Box [4]

Road profiles are defined as parameterized spline curves as shown in Figure 2. The profile definition is based on the points defined in Figure 3. Given a wheel center location (x_0, z_0), the radius (R), angle (θ), and height (z_4) the points (x_3, z_3) and (x_4, z_4) need to be defined. The slope at (x_3, z_3) can be used to give (x_4, z_4).

Using polar coordinates

$$x_1 = x_0 + R \cos(90 - \theta)$$

$$z_1 = z_0 - R \sin(90 - \theta)$$

(1).

Using the Law of Sines with $x_2=x_0$ (vertical line)

$$D = \sqrt{(x_2 - x_0)^2 + (z_2 - z_0)^2} = z_2 - z_0$$

$$\frac{D}{\sin(90)} = \frac{D}{1} = \frac{R}{\sin(90 - \theta)}$$

$$z_2 = z_0 - D = z_0 - \left[\frac{R}{\sin(90 - \theta)} \right]$$

(2)

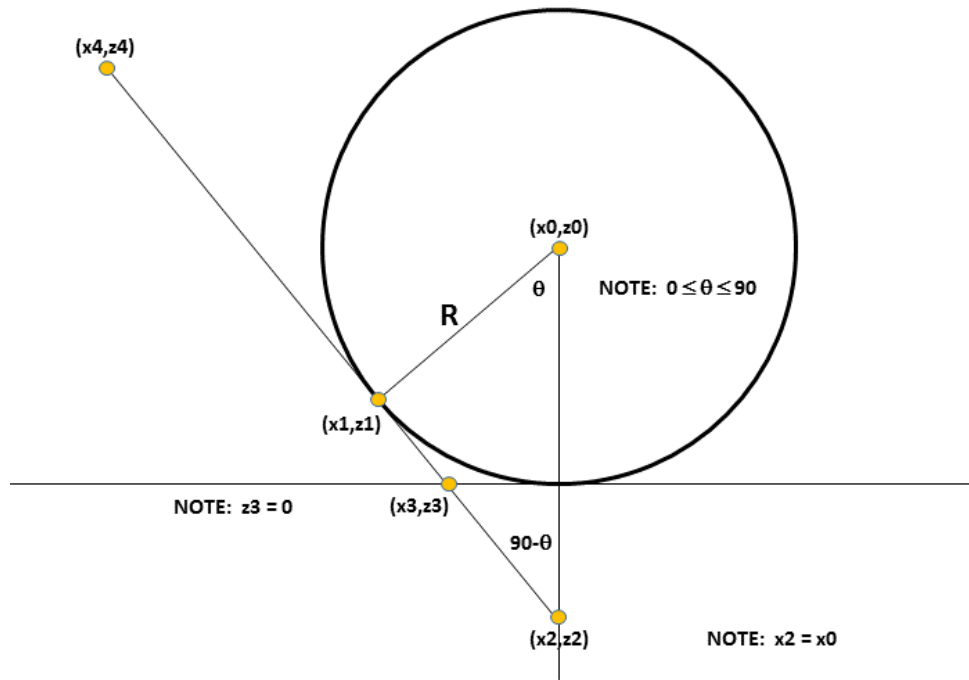


Figure 3 – Spline Curve Definition Points

with the slope defined as

$$Lslope = \frac{z1 - z2}{x1 - x2} = \frac{z1 - z2}{x1 - x0}$$

$$z = z2 + (x - x2)(Lslope) = z2 + (x - x0)(Lslope) \quad (3).$$

When $z=0$ ($z3=0$) then

$$x3 = \frac{-z2}{Lslope} + x0 \quad (4).$$

Once the height $z4$ is specified then

$$\begin{aligned} \frac{z4 - z2}{x4 - x2} &= Lslope \\ x4 &= \frac{z4 - z2}{Lslope} + x2 = \frac{z4 - z2}{Lslope} + x0 \end{aligned} \quad (5).$$

The initial parameters (wheel center location $(x0, z0)$, the radius (R), angle (θ), height ($z4$)) are assumed to be from a static settled state. If the wheel center location is not laterally symmetric in x then separate left and right road profiles will be required. The radius (R) is defined as $R = \text{abs}(z0)$ (settled R), the angle (θ) is set to 45 degrees (or as desired), and the height ($z4$) is set (e.g. 12 inches).

For the rear (1) is modified to

$$x1 = x0 - R \cos(90 - \theta).$$

The profile(s) are then defined as (assuming coordinate +x forward) in Table 1.

Table 1 – 2D Road Profile Definition

X	Z
-1000	rearWHL_z4*chocks_flg
rearWHL_x4	rearWHL_z4*chocks_flg
rearWHL_x3	0
frontWHL_x3	0
frontWHL_x4	frontWHL_z4*chocks_flg
1000	frontWHL_z4*chocks_flg

Figures 4 and 5 show an example of the resulting road profile (angle (θ) = 45 degrees, height (z4) = 12 inches).

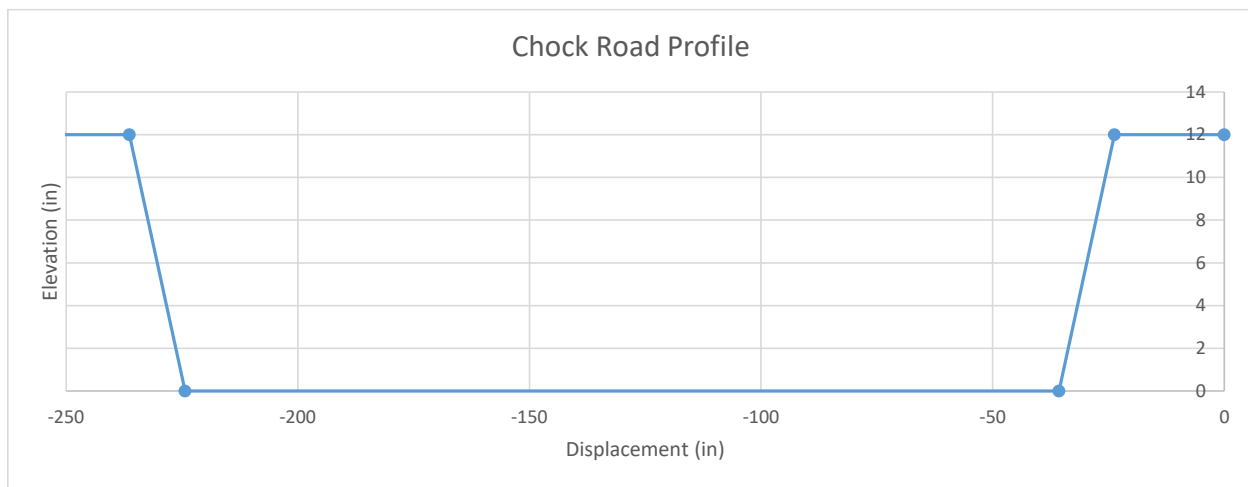


Figure 4 – Road Profile

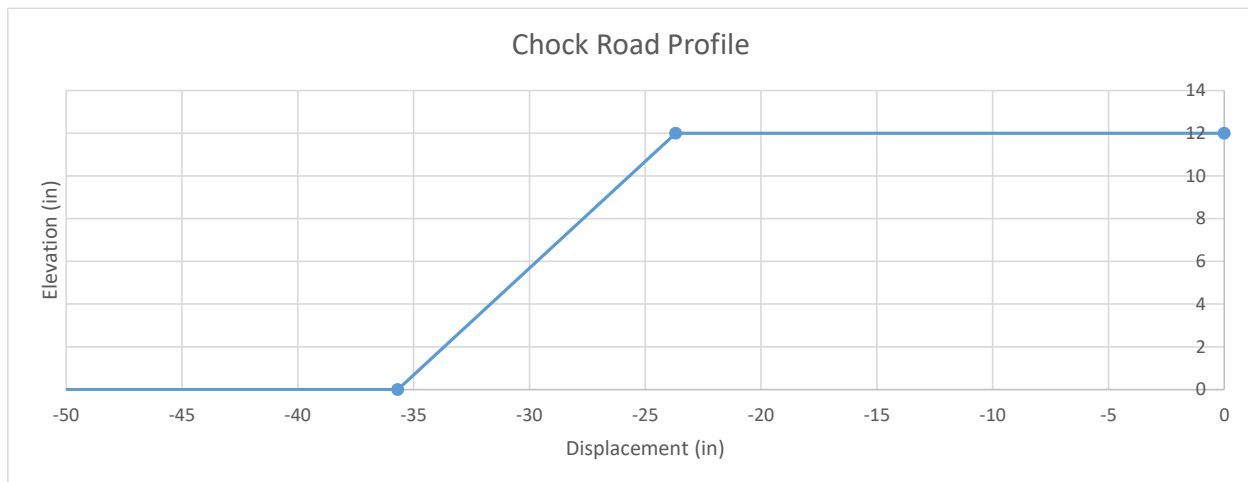


Figure 5 – Zoomed Road Profile

The "chocks_flg" in Table 1 allows for adjustment of the road profile with or without chocks. (0 => flat, 1 => with chocks).

REFERENCE

- [1] TN15-0312-01 74 TON M1 TRANSPORTABILITY TIE-DOWN RAIL IMPACT FORCES ANALYSIS (see TR27441 - 2015 Technical Notes and Problem Solutions in Dynamics. DIST C (DTIC AD 1001 907)).
- [2] TM 55-2200-001-12 TRANSPORTABILITY GUIDANCE, APPLICATION OF BLOCKING, BRACING, AND TIEDOWN MATERIALS FOR RAIL TRANSPORT. DIST A. Approved for public release: distribution unlimited. Inactive, see (<https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=779bca6195ec96a97fc51ccfcb6ca90f82b015dc>)
- [3] TM 55-2350-255-14 TRANSPORTABILITY GUIDANCE FOR TANK COMBAT, FULL-TRACKED, M1-SERIES: 105-MM GUN, M1, 105-MM GUN, IPM1, 120-MM GUN, M1A1, 120-MM GUN, M1A2 (GENERAL ABRAMS). DIST C. (<https://www.ldac.army.mil/>)
- [4] Virtual.Lab Motion (http://www.plm.automation.siemens.com/en_us/products/lms/virtual-lab/motion/)
- [5] Reference Internal Project 14-511 – M1 TRANSPORTABILITY TIEDOWN RAIL IMPACT FORCES
- [6] TM55-2220-058-14 TRANSPORTABILITY GUIDANCE FOR TRANSPORT OF CARGO ON THE RAILCAR FLAT, 140-TON-CAPACITY. GPO. (<https://www.ldac.army.mil/>).

TN 15-0519-01

TRACK LENGTH CALCULATION FROM ROAD WHEEL POSITIONS

May 19, 2015

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC)
Detroit Arsenal
Systems Integration & Engineering
ATTN: RDTA-SIE/MS157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

As a tracked vehicle's suspension articulates what is the track length required to wrap around the road wheels or how "loose" will the track become when the suspension is collapsed for transportability.

PROBLEM SOLUTION:

The track loop is defined in Figure 1. Assuming an x-y plane (x horizontal, y vertical), the road wheels or rollers are numbered consecutively counter-clockwise with their center positions and associated radius. The closed loop nature of the problem requires the calculation of the current indexed road wheel back point/vector with the previous (or last) one. This presents the need for a "back" and "forward" index from

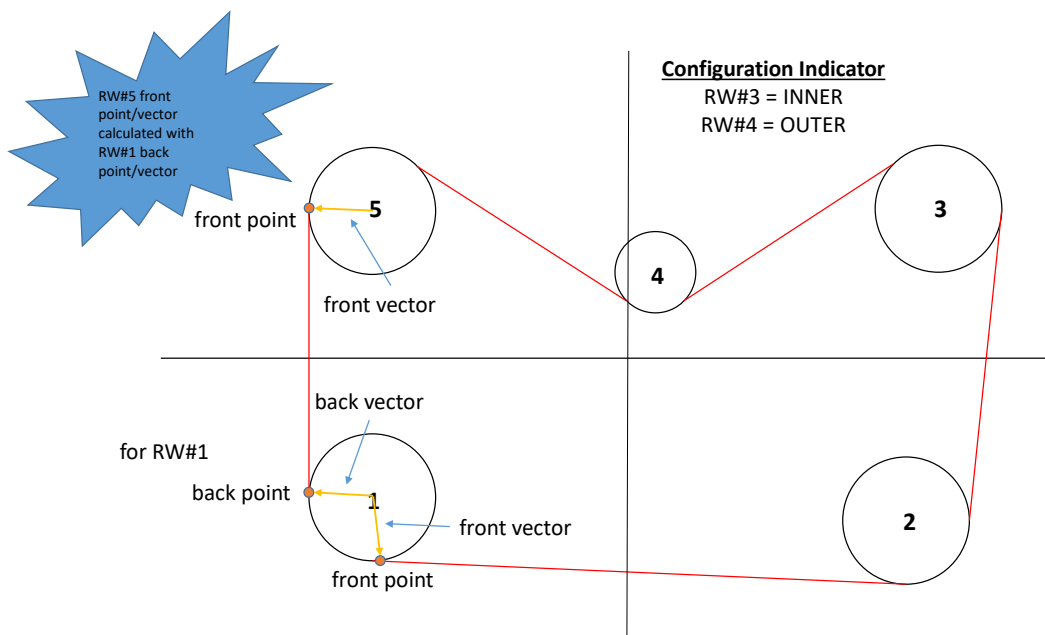


Figure 1 – Track Loop Definitions

the current position. Each road wheel is also classified as either "INNER" or "OUTER" depending on its initial configuration inside or outside of the track loop.

The tangent points on each road wheel are used to calculate the angular distance on the road wheel and the linear distance between them---thus summing these individual path segments will give the total path distance. To understand the calculations the concept of “looking back” is important. In one step, part of the calculations are done on the current road wheel and part on the “backward” or lower index road wheel. At the end of the loop all values for each road wheel will be calculated.

Two configurations are outlined as shown Figures 2 - 3 for different indicator type road wheels. The base angle (the angle from the horizontal to the line between road wheel centers, [1]) and distance between road wheel centers are defined in equations (1) – (4).

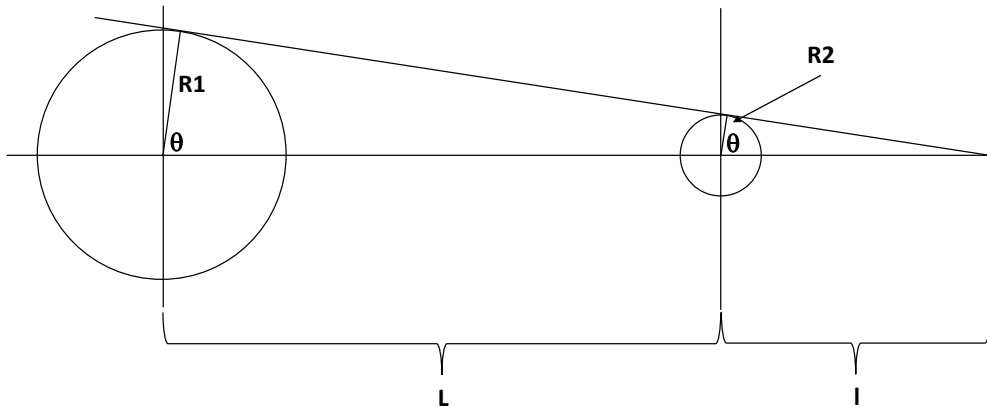


Figure 2 – Same Configuration Indicator Road Wheels ($R1=RW[i]$, $R2=RW[i-1]$)

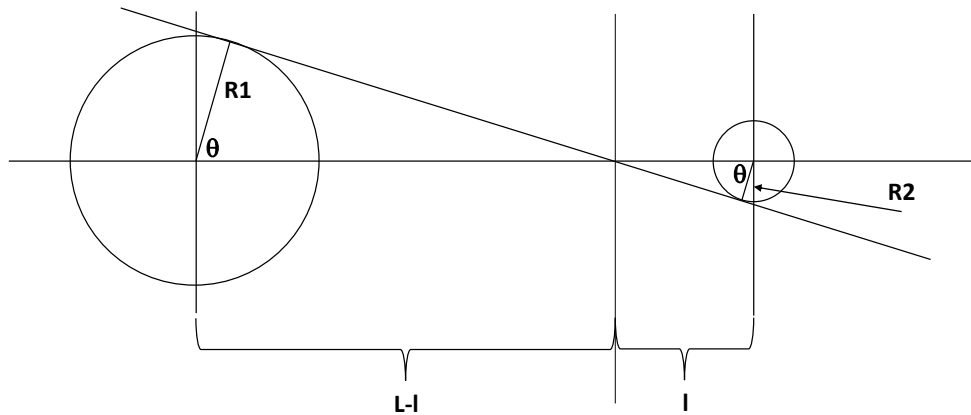


Figure 3 – Opposite Configuration Indicator Road Wheels ($R1=RW[i]$, $R2=RW[i-1]$)

$$rise = y_b - y_f \quad (1)$$

$$run = x_b - x_f \quad (2)$$

$$\theta_b = \tan^{-1} \left(\frac{rise}{run} \right) \quad (3)$$

$$d = \sqrt{(rise)^2 + (run)^2} \quad (4)$$

The points (back (b), for i, and front (f) for i-1) are defined in equations (5) – (6)

$$p_b = p_{cf} + r_f \cos(\theta) \quad (5)$$

$$p_f = p_{cb} + r_b \cos(\theta) \quad (6)$$

where the road wheel center point is added to the radius multiplied by the cosine of the angle. For opposite type road wheels equation (6) is replaced with equation (7) below

$$p_f = p_{cb} + r_b \cos(\theta_2) \quad (7)$$

Table 1 outlines the configurations and associated parameters.

Table 1 – Angle Definitions based on Configuration Indicators and Adjacent Road Wheel Radius		
FRONT	BACK	PARAMETERS
INNER	INNER	$\left\{ \begin{array}{l} r_b = r_f, \quad \theta_0 = \frac{\pi}{2}, \theta = \theta_b + \theta_0 \\ r_b < r_f, \quad l = \frac{r_b d}{r_f - r_b}, \theta_0 = \cos^{-1}\left(\frac{r_f}{d+l}\right), \theta = \theta_b + \theta_0 \\ r_b > r_f, \quad l = \frac{r_f d}{r_b - r_f}, \theta_0 = \cos^{-1}\left(\frac{r_b}{d+l}\right), \theta = \theta_b + \frac{\pi}{2} + \left(\frac{\pi}{2} - \theta_0\right) \end{array} \right\}$
OUTER	OUTER	see above with $\theta = -\theta$
INNER	OUTER	$l = \frac{r_b d}{r_f - r_b}, \theta_0 = \cos^{-1}\left(\frac{r_f}{d-l}\right), \theta = \theta_b + \theta_0, \theta_2 = \theta_b + \pi + \theta_0$
OUTER	INNER	see above with $\theta = \theta_b - \theta_0, \theta_2 = \theta_b + \pi - \theta_0$

Equations (1) – (4) are repeated for the “back” and “front” points for each road wheel to define the track path linear distance. Front and back vectors are defined by subtracting the road wheel center from the front and back points (see Figure 1).

The angular distance is calculated from equation (8) and the normalized front and back vectors, are used with the cross and dot product relationships [2, 3] as defined in Table 2

$$s = r\theta \quad (8)$$

Table 2 – Wrap Angle Definitions	
INDICATOR	WRAP ANGLE
INNER	$\theta_w = \text{sign} \left\{ \sin^{-1} \left(\frac{vb_x vf_y - vf_x vb_y}{vb_d vf_d} \right) \right\} \cos^{-1} \left(\frac{vb_x vf_x + vb_y vf_y}{vb_d vf_d} \right)$
OUTER	$\theta_w = \text{sign} \left\{ \sin^{-1} \left(\frac{vf_x vb_y - vb_x vf_y}{vf_d vb_d} \right) \right\} \cos^{-1} \left(\frac{vb_x vf_x + vb_y vf_y}{vb_d vf_d} \right)$

If the wrap angle becomes negative (more than 180 degrees), a new road wheel center is calculated with an opposite indicator (INNER -> OUTER or OUTER -> INNER) that is tangent to the old road wheel with the new center defined as

$$2r \frac{v_2}{\|v_2\|} \quad (9)$$

from the old center as shown in Figure 4. The indicator is changed, the points and vectors are recalculated, and then new angular and linear distances are calculated to give a more representative measure of a smooth path the track will follow.

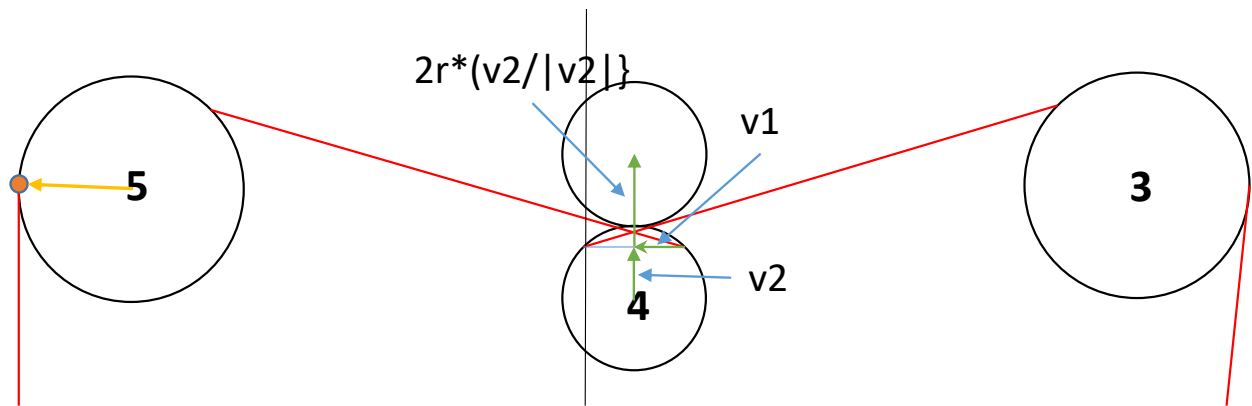


Figure 4 – Road Wheel Indicator Change Center Point Redefinition

Table 3 shows results for several test cases.

Table 3 – Example Results

data0.txt	52.56
data1.txt	40.62
data2.txt	53.43
data3.txt	58.18
data4.txt	523609.50
data5.txt	523613.08

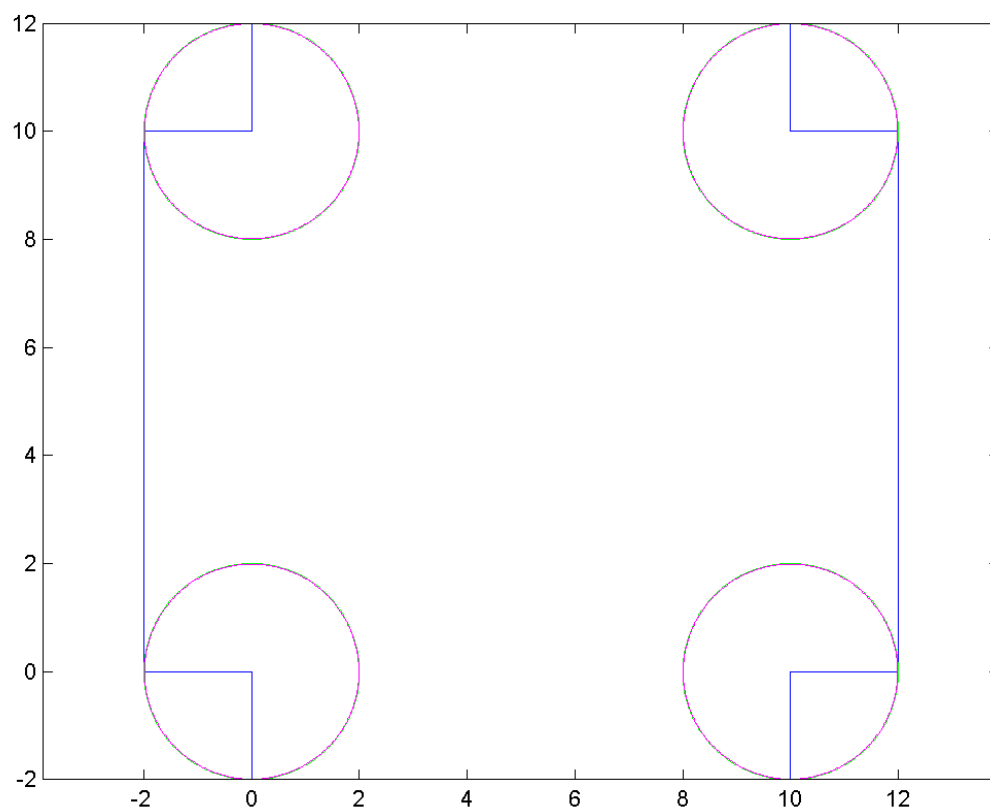
The file format for each file is defined as:

```

===BOF
# Example
# #TITLE file name or title here
# #THICK track thickness (1/2 will be added to radius)
# this is a comment line
# enter road wheels in counter-clockwise
# x y radius indicator(I or O)
# the following Inner RW has a center at (5,10) radius 13.2
# with track thickness of 2.2
#
#TITLE Example
#THICK 2.2
...
5 10 13.2 I
...
# Example
===EOF

```

Several example data files and plots of the path profile are included below.

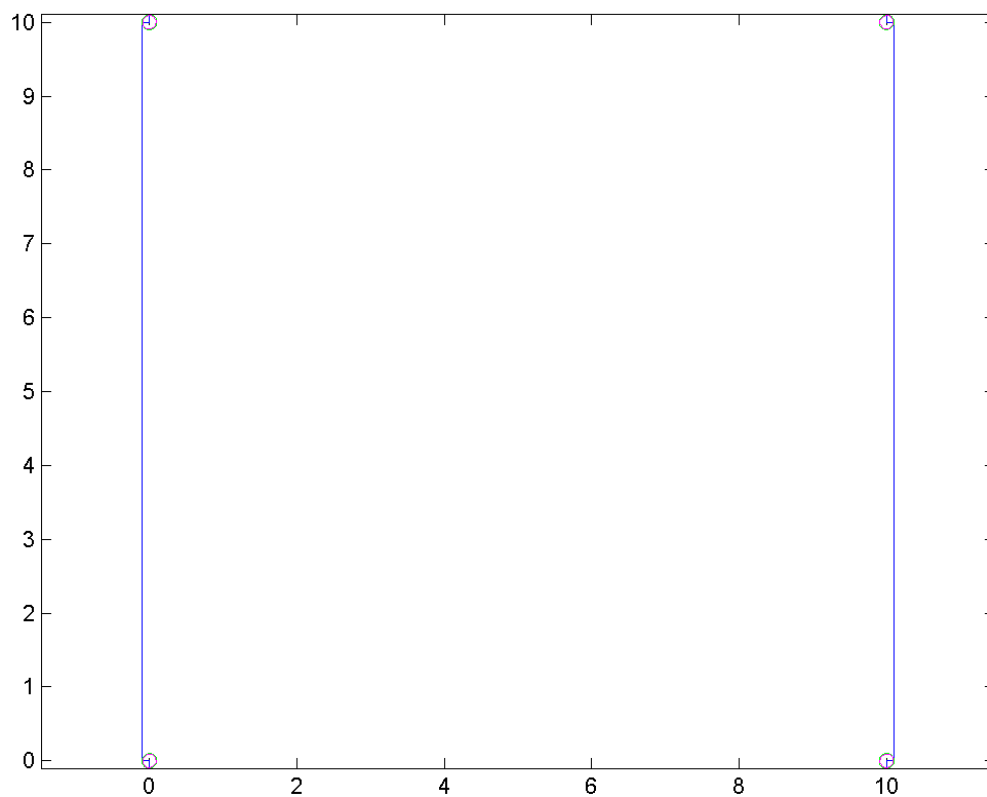


```

===BOF
#TITLE data0.txt
#THICK 0
0 0 2 I
10 0 2 I
#comment line her
10 10 2 I
0 10 2 I
===EOF

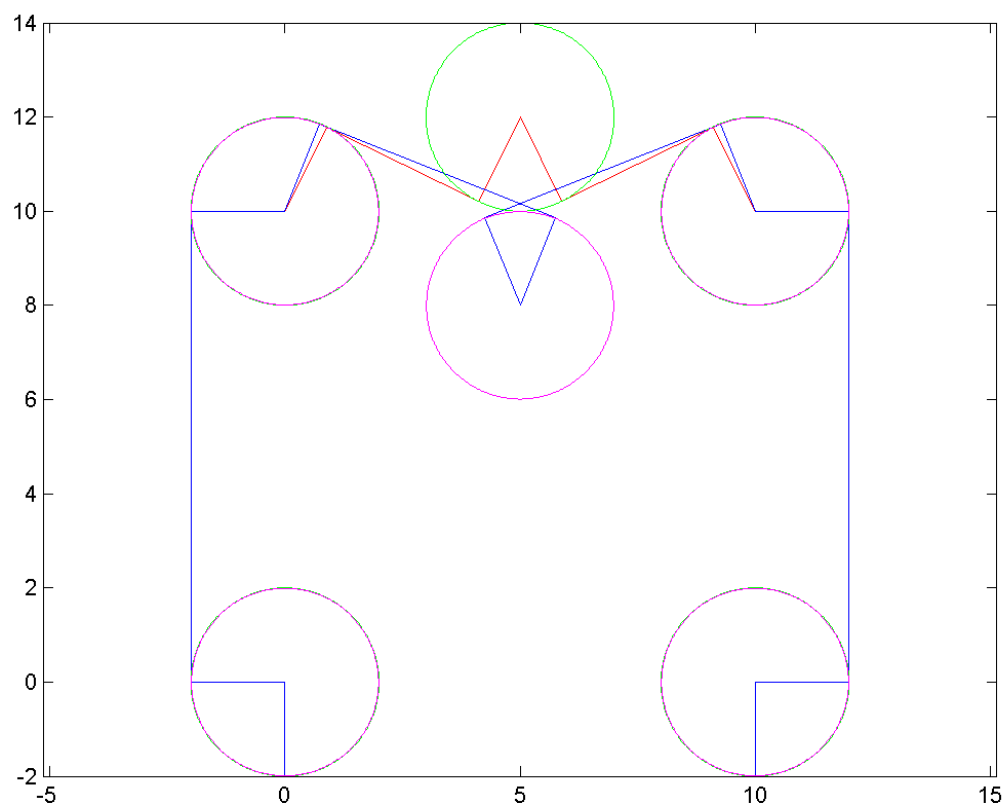
```

Figure 5 – Test Case #0



```
===BOF
#TITLE data1.txt
#THICK 0
0 0 0.1 I
10 0 0.1 I
10 10 0.1 I
0 10 .1 I
===EOF
```

Figure 6 – Test Case #1

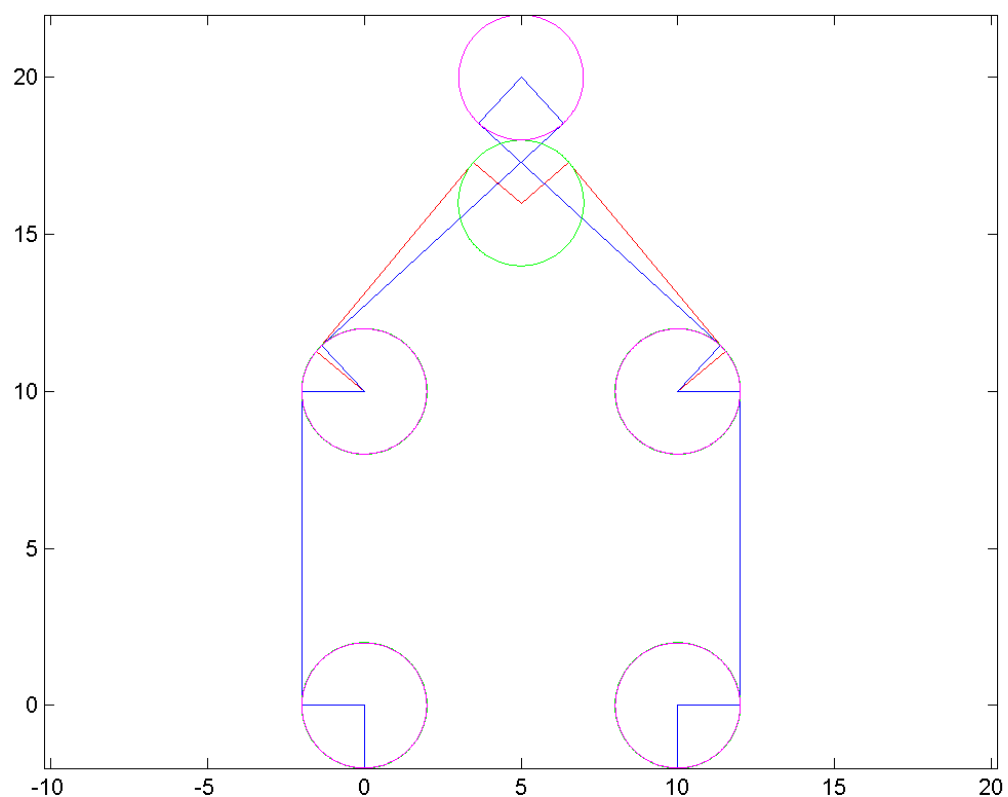


```

===BOF
#TITLE data2.txt
#THICK 0
0 0 2 I
10 0 2 I
10 10 2 I
5 8 2 I
0 10 2 I
===EOF

```

Figure 7 – Test Case #2

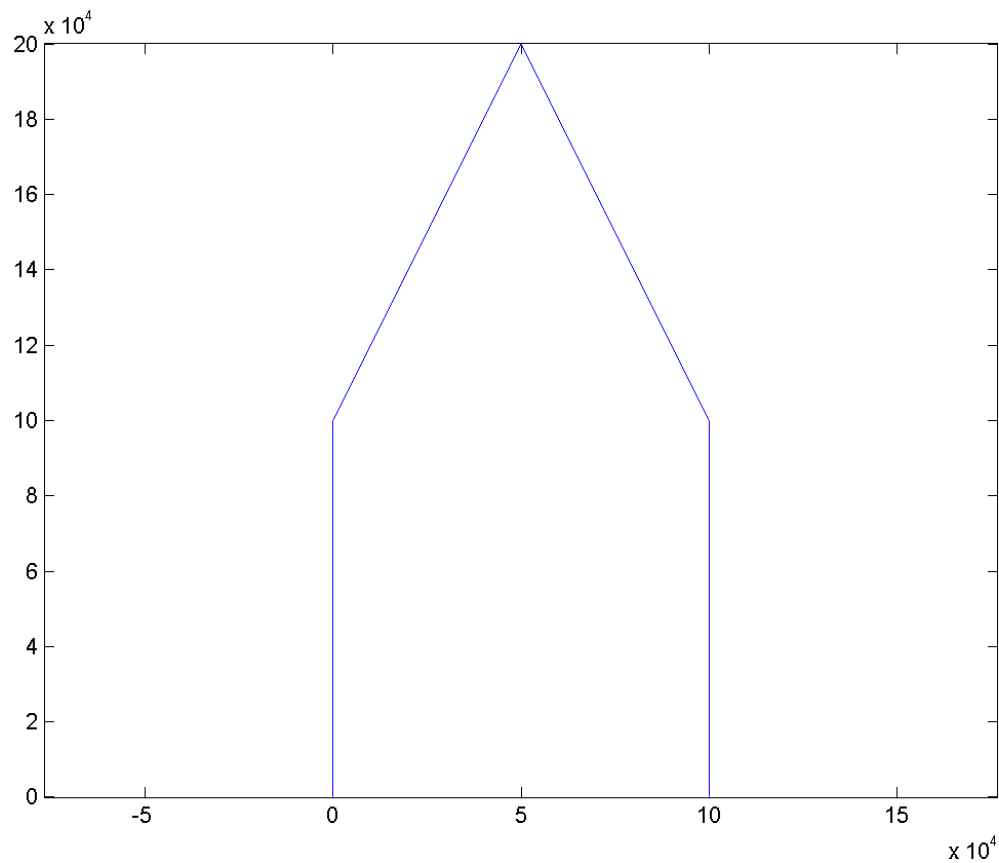


```

===BOF
#TITLE data3.txt
#THICK 0
0 0 2 I
10 0 2 I
10 10 2 I
5 20 2 O
0 10 2 I
===EOF

```

Figure 8 – Test Case #3



```

===BOF
#TITLE data4.txt
#THICK 0
0 0 1 I
100000 0 1 I
100000 100000 1 I
50000 200000 1 O
0 100000 1 I
===EOF

```

Figure 9 – Test Case #4

Figure 10 shows a zoomed view to illustrate how the path is smoothed when the wrap angle is greater than 180 degrees---the indicator type is switched, position changed, and path recalculated. In Figure 10, blue represents the original path and magenta the original road wheel position. In this case (#4), the road wheel is switched from outer to inner. The new path is represented as red and the repositioned road wheel as green. See Appendix A.2 and A.3 for details on how the plots were created using MATLAB [4].

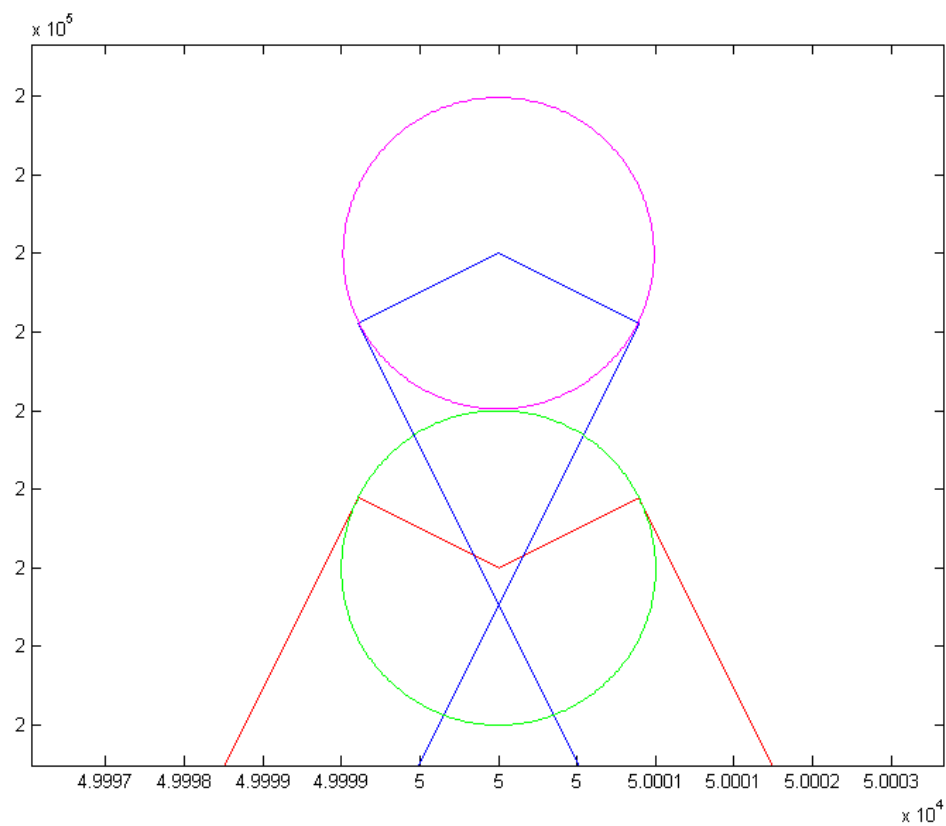
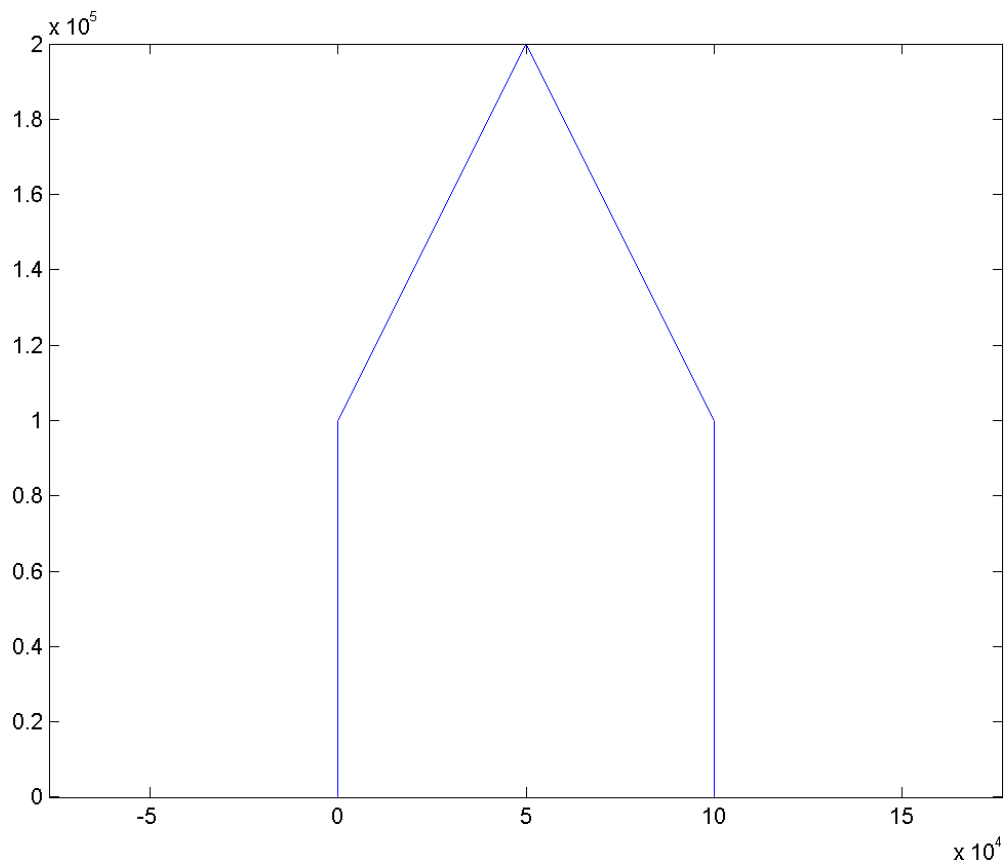


Figure 10 – Zoomed Test Case #4



```

===BOF
#TITLE data5.txt
#THICK 0
0 0 1 I
100000 0 1 I
100000 100000 1 I
50000 200000 1 I
0 100000 1 I
===EOF

```

Figure 11 – Test Case #5

A list of functions in the “trklen.psm1” (Appendix A.1) module are described below:

- `get_h(file)` - read in road wheel data from file
- `prt_rw ()` - print road wheel data (x,y,radius,indicator)
- `prt_pts ()` - print points (front x,y, back x,y)
- `prt_vec ()` - print vectors (front x,y, back x,y)
- `plot_pts ()` - output points for plotting (back,center,front)
- `get_bi (i,l)` - get back index (l-1 or i-1)
- `get_fi (i,l)` - get front index (0 or i+1)
- `get_ptsvec (l)` - calculate tangent points/vectors (front/back)
- `get_wa ()` - calculate wrap angle
- `get_wadis (l, flg)` - calculate wrap angle distance w/wo adjust road wheel
- `sum (flg)` - output sum of path segments w/wo title

Windows PowerShell is used to process the road wheel data and calculate the path length. The calling sequence is described in the following example scripts.

go

```
& "c:\...\powershell\v1.0\powershell.exe" -executionpolicy unrestricted -file ".\go1.ps1"
```

go1

```
remove-item "results.txt"
.\tl data0.txt out0.txt
.\tl data1.txt out1.txt
.\tl data2.txt out2.txt
.\tl data3.txt out3.txt
.\tl data4.txt out4.txt
.\tl data5.txt out5.txt
```

tl

```
#
# Loop length (plane) based on (x,y,r,IN/OUT), CCW loop = pos
#
param($filei,$fileo)
#
# $filei      input file name
# $fileo      output file name
#
import-module .\trklen

#==== initialize variables
# global
$m_thick = 0.0
$global:h=@{}

$global:h["rwx"]=@()
$global:h["rwy"]=@()
$global:h["rwr"]=@()
$global:h["rwi"]=@()
$global:h["title"]=@()

#==== main
$a = get_h $filei
$c = $h["rwx"].length

$global:h["thb"] = new-object "Double[]" $c
$global:h["dis"] = new-object "Double[]" $c
$global:h["pbx"] = new-object "Double[]" $c
$global:h["pby"] = new-object "Double[]" $c
$global:h["pfx"] = new-object "Double[]" $c
$global:h["pfy"] = new-object "Double[]" $c
$global:h["pdis"] = new-object "Double[]" $c
$global:h["vbx"] = new-object "Double[]" $c
$global:h["vby"] = new-object "Double[]" $c
$global:h["vfx"] = new-object "Double[]" $c
$global:h["vfy"] = new-object "Double[]" $c
$global:h["wa"] = new-object "Double[]" $c
$global:h["wdis"] = new-object "Double[]" $c

$global:h["irwx"] = new-object "Double[]" $c
$global:h["irwy"] = new-object "Double[]" $c
$global:h["irwi"] = new-object "String[]" $c
.
.
.
```

tl (cont.)

```

.
.
.
#"====calling get_ptsvec"
(0..($c-1)) | get_ptsvec $c

#"====calling get_wadis false"
(0..($c-1)) | get_wadis $c $false
#prt_rw
#prt_pts
#prt_vec
#prt_h
plot_pts | out-file -encoding ASCII ($fileo+".bef")

#"====calling get_wadis true"
(0..($c-1)) | get_wadis $c $true
#prt_rw
#prt_pts
#prt_vec
#prt_h
plot_pts | out-file -encoding ASCII $fileo
(0..($c-1)) | sum $true | out-file -encoding ASCII -append ("results.txt")
#(0..($c-1)) | sum $false

```

Test cases for the dot and cross product are included in Appendix A.4 and A.5.

REFERENCE

- [1] ATAN2 ([https://msdn.microsoft.com/en-us/library/system.math.atan2\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.math.atan2(v=vs.110).aspx))
- [2] Dot Product (https://en.wikipedia.org/wiki/Dot_product)
- [3] Cross Product (https://en.wikipedia.org/wiki/Cross_product)
- [4] MATLAB (<http://www.mathworks.com/>)
- [5] Windows Powershell (http://en.wikipedia.org/wiki/Windows_PowerShell)
- [6] Reference Internal Project 15-517 TRACK LENGTH CALCULATION FROM ROADWHEEL POSITIONS

APPENDIX A.1 – TRKLEN.PSM1

```

=====
# trklen.psm1
#
# hashtable indexes
#
#   rwx = road wheel x
#   rwy = road wheel y
#   rwr = road wheel radius
#   rwi = road wheel indicator (I=INNER, O=OUTER)
#   thb - theta base/base angle (center of current rw to center of back rw)
#   dis - distance (linear, center of current rw to center of back rw)
#   pbx - back point x
#   pby - back point y
#   pfx - front point x
#   pfy - front point y
#   pdis - path distance (between backpt[i] to frontpt[i-1])
#   vbx - back vector x
#   vby - back vector y
#   vfx - front vector x
#   vfy - front vector y
#   irwx - save original rwx for changed indicator
#   irwy - save original rwy for changed indicator
#   irwi - save original rwi for changed indicator
#   wa - wrap angle
#   wadis - wrap angle distance
#   title - title associated with data file
#
=====
function get_h ($file) {
#
#   Read road wheel data into hashtable (x y radius indicator=I/O)
#
#   Parameters:
#       file = file name to read
#
#   Variables:
#       m_thick = track thickness (assumes global), adds 1/2 to radius
#
$cTITLE = "#TITLE"
$cTHICK = "#THICK"
$cCOMMENT = "#"
$m_thick = 0.0

get-content $file | foreach {
    $tmp = -split $_

    switch ($tmp[0]) {

        $cTITLE {
            #write-host "TITLE:" $tmp[1]
            $h["title"] = $h["title"] + $tmp[1]
            break }

        $cTHICK {
            $m_thick = [double]::Parse($tmp[1])
            #write-host "THICK:" ("0,8:N3)" -f $m_thick
            break }

        {$tmp[0].Substring(0,1) -eq $cCOMMENT} {
            #write-host "COMMENT:" $tmp[1]
            break }

        default {
            #write-host "default"

            $h["rwx"] = $h["rwx"] + [double]::Parse($tmp[0])
            $h["rwy"] = $h["rwy"] + [double]::Parse($tmp[1])
            $h["rwr"] = $h["rwr"] + ([double]::Parse($tmp[2]) + $m_thick/2)
        }
    }
}
}

```

```

        $h["rwi"] = $h["rwi"] + $tmp[3]
    }
}
}
}
#=====
function pow ($v,$p) {
    #
    #   Shorter version of POW function
    #
    #   Parameters:
    #       v = base number
    #       p = exponent
    #
    [Math]::Pow($v,$p)
}
#=====
function prt_h () {
    #
    #   Print hashtable data
    #
    "====prt_h===="
    $h.Keys | foreach {
        "{0,3}. {1}" -f ($_, ($h[$_] -join ", "))
    }
    "====="
}
#=====
function prt_rw () {
    #
    #   Print road wheel data
    #
    "====prt_rw====x,y,r,i"
    for ($i = 0; $i -lt $h["rwx"].count;$i = $i + 1) {
        "{0,8:N3} {1,8:N3} {2,8:N3} {3,8:N3} {4}" -f
        ($i,$h["rwx"][$i],$h["rwy"][$i],$h["rwr"][$i],$h["rwi"][$i] )
    }
    "====="
}
#=====
function prt_pts () {
    #
    #   Print road wheel points (front/back)
    #
    "====prt_pts====f:x,y b:x,y"
    for ($i = 0; $i -lt $h["rwx"].count;$i = $i + 1) {
        "{0,8:N3} {1,8:N3} {2,8:N3} {3,8:N3} {4}" -f
        ($i,$h["pfx"][$i],$h["pfy"][$i],$h["pbx"][$i],$h["pby"][$i] )
    }
    "====="
}
#=====
function prt_vec () {
    #
    #   Print road wheel vectors (front/back)
    #
    "====prt_vec====f:x,y b:x,y"
    for ($i = 0; $i -lt $h["rwx"].count;$i = $i + 1) {
        "{0,8:N3} {1,8:N3} {2,8:N3} {3,8:N3} {4}" -f
        ($i,$h["vfx"][$i],$h["vfy"][$i],$h["vbx"][$i],$h["vby"][$i] )
    }
    "====="
}
#=====
function plot_pts () {
    #
    #   Print track loop points for plotting
    #
    for ($i = 0; $i -lt $h["rwx"].count;$i = $i + 1) {
        "{0,12:F3} {1,12:F3} -1" -f ($h["pbx"][$i],$h["pby"][$i] )
        "{0,12:F3} {1,12:F3} {2,12:F3}" -f ($h["rwx"][$i],$h["rwy"][$i],$h["rwr"][$i])
    }
}

```

```

    "{0,12:F3} {1,12:F3} -1" -f ($h["pfx"][$i],$h["pfy"][$i] )
}
$i = 0
"{0,12:F3} {1,12:F3} -1" -f ($h["pbx"][$i],$h["pby"][$i] )
}
#####
function get_bi ($i,$l) {
#
#   Get back index
#
#   Parameters:
#       i = index number
#       l = total number of road wheels (zero based)
#
if ($i -eq 0) {
    ($l - 1)
} else {
    ($i - 1)
}
}
#####
function get_fi ($i,$l) {
#
#   Get front index
#
#   Parameters:
#       i = index number
#       l = total number of road wheels (zero based)
#
if ($i -eq $l-1) {
    0
} else {
    ($i + 1)
}
}
#####
function get_ptsvec ($l) {
#
#   Calculate tangent points/vectors (front/back)
#
#   Parameters:
#       l = total number of road wheels (zero based)
#
Process {
    $i = $_
    $bi = get_bi $i $l

    $rb = $h["rwr"][$bi]
    $xb = $h["rwx"][$bi]
    $yb = $h["rwy"][$bi]
    $ib = $h["rwi"][$bi]

    $rf = $h["rwr"][$i]
    $xf = $h["rwx"][$i]
    $yf = $h["rwy"][$i]
    $if = $h["rwi"][$i]

    $rise = $yb - $yf
    $run = $xb - $xf
    $thb = [Math]::atan2($rise,$run)
    $h["thb"][$i] = $thb

    $dis = [Math]::Sqrt([Math]::Pow($rise,2)+[Math]::Pow($run,2))
    $h["dis"][$i] = $dis

    $flg2 = $false
    if ($if -eq $ib) {
        #":I SAME"
        # INNER
        #"->INNER"
        if ($rb -eq $rf) {

```

```

        $th0 = [Math]::PI/2
        $th = $thb + $th0
        #"rb = rf,dis,th0,th: $dis, $th0, $th"
    } elseif ($rb -lt $rf) {
        $l = $rb * $dis / ($rf - $rb)
        $th0 = [Math]::acos($rf / ($dis + $l))
        $th = $thb + $th0
        #"rb < rf,dis,l2,th0,th: $dis, $l, $th0, $th"
    } else {
        $l = $rf * $dis / ($rb - $rf)
        $th0 = [Math]::acos($rb / ($dis + $l))
        # $th = $thb + $th0 + [Math]::PI/2
        $th = $thb + [Math]::PI/2 + ([Math]::PI/2 - $th0)
        #"rb > rf,dis,l2,th0,th: $dis, $l, $th0, $th"
    }
    # OUTER
    if ($if -eq "O") {
        # "->OUTER"
        $th = -$th
    }
} else {
    # "I DIFFERENT"
    $flg2 = $true
    # INNER/OUTER
    # "->INNER/OUTER"
    $l = $rb * $dis / ($rf + $rb)
    $th0 = [Math]::acos($rf / ($dis - $l))
    $th = $thb + $th0
    # $th2 = $th - [Math]::PI
    $th2 = $thb + [Math]::PI + $th0
    #"rf,rb,dis,l: $rf, $rb, $dis, $l"
    #"l,th0,th,th2: $l, $th0, $th, $th2"
    # OUTER/INNER
    if ($if -eq "O") {
        # "->OUTER/INNER"
        $th = $thb - $th0
        $th2 = $thb + [Math]::PI - $th0
    }
    #"dis,l,th0,th: $dis, $l, $th0, $th"
}
$pbx = $xf + $rf * [Math]::cos($th)
$pyb = $yf + $rf * [Math]::sin($th)

if ($flg2) {
    $pfx = $xb + $rb * [Math]::cos($th2)
    $pfy = $yb + $rb * [Math]::sin($th2)
} else {
    $pfx = $xb + $rb * [Math]::cos($th)
    $pfy = $yb + $rb * [Math]::sin($th)
}
#write-host ("pts: {0,5:N3} {1,5:N3} {2,5:N3} {3,5:N3} {4,5:N3}" -f
($th*180.0/[Math]::PI) , $pbx, $pyb, $pfx, $pfy )
# back ones
$risew = $pyb - $pfy # $t2[1]-$t1[1] # yb - $y0
$runw = $pbx - $pfx # $t2[0] - $t1[0] # $xb - $x0
$pthdis = [Math]::Sqrt([Math]::Pow($risew,2) + [Math]::Pow($runw,2))
$h["pdis"][$i] = $pthdis

# back
$h["pbx"][$i] = $pbx
$h["pyb"][$i] = $pyb
$h["vbx"][$i] = $pbx - $xf
$h["vby"][$i] = $pyb - $yf

# front
$h["pfx"][$bi] = $pfx
$h["pfy"][$bi] = $pfy
$h["vfx"][$bi] = $pfx - $xb
$h["vfy"][$bi] = $pfy - $yb
}
}

```



```

=====
function get_wa () {
#
# Calculate wrap angle
#
Process {
    $i = $_

    $vbx = $h["vbx"][$i]
    $vby = $h["vby"][$i]
    $vbd = [Math]::Sqrt( [Math]::Pow( $vbx ,2) + [Math]::Pow( $vby ,2))

    $vfx = $h["vfx"][$i]
    $vfy = $h["vfy"][$i]
    $vfd = [Math]::Sqrt( [Math]::Pow( $vfx ,2) + [Math]::Pow( $vfy ,2))

    #write-host ("wa: {0,3:D} {1,5:N3} {2,5:N3} {3,5:N3} {4,5:N3} {5,5:N3} {6,5:N3}" -f ($i
    , $vbx, $vby, $vbd, $vfx, $vfy, $vfd ))
    #write-host ("wa: {0,5:N3} {1,5:N3}" -f (($vbx*$vfx + $vby*$vfy), ($vbd*$vfd)))
    #write-host ("wa: {0,5:N3} {1,5:N3}" -f (($vbx*$vfy - $vfx*$vby), ($vbd*$vfd)))

    $swa = [Math]::acos( ($vbx*$vfx + $vby*$vfy)/($vbd*$vfd) )
    if ($h["rwi"][$i] -eq "I") {
        $swact = [Math]::asin( ($vbx*$vfy - $vfx*$vby)/($vbd*$vfd) )
    } else {
        $swact = [Math]::asin( ($vfx*$vby - $vbx*$vfy)/($vfd*$vbd) )
    }
    # not more than 180 deg apart
    if ($swact -lt 0) {
        $swa = -$swa
    }
    #write-host "wa: $swa"
    $swa
}
}
=====
function get_wadis ($l, $flg) {
#
# Calculate wrap angle distance
#
# Parameters:
# l = total number of road wheels (zero based)
# flg = true/adjust road wheel center for smooth path
#
Process {
    $i = $_
    $bi = get_bi $i $l
    $fi = get_fi $i $l

    $swa = $i | get_wa # $i # | get_wa
    if (($swa -lt 0.0) -and $flg) {
        # "get_wadis"
        $v1x = ($h["pbx"][$i] - $h["pfx"][$i]) / 2 + $h["pfx"][$i]
        $v1y = ($h["pby"][$i] - $h["pfy"][$i]) / 2 + $h["pfy"][$i]
        $v2x = $v1x - $h["rwx"][$i]
        $v2y = $v1y - $h["rwy"][$i]
        $v2l = [Math]::Sqrt( [Math]::Pow($v2x,2) + [Math]::Pow($v2y,2))

        $ncx = 2 * $h["rwr"][$i] * $v2x/$v2l + $h["rwx"][$i]
        $ncy = 2 * $h["rwr"][$i] * $v2y/$v2l + $h["rwy"][$i]
        # $ncx = 2 * $v2x + $h["rwx"][$i]
        # $ncy = 2 * $v2y + $h["rwy"][$i]
        # "wadis: {0,3:D}. {1,5:N3} {2,5:N3} {3,5:N3} {4,5:N3}" -f
        $i, $ncx, $ncy, $h["rwx"][$i], $h["rwy"][$i]

        # save
        $h["irwx"][$i] = $h["rwx"][$i]
        $h["irwy"][$i] = $h["rwy"][$i]
        $h["irwi"][$i] = $h["rwi"][$i]
        $h["rwx"][$i] = $ncx
        $h["rwy"][$i] = $ncy
    }
}
}

```

```

        if ($h["rwi"][$i] -eq "I") {
            $h["rwi"][$i] = "O"
        } else {
            $h["rwi"][$i] = "I"
        }
        #---get_ptsvec from get_wadis"
        $i | get_ptsvec $l
        #---get_ptsvec from get_wadis"
        $fi | get_ptsvec $l
        #---get_wa from get_wadis"
        $i | get_wadis $l $true
        $waa = ($fi,$i,$bi) | get_wa
        $h["wa"][$fi] = $waa[0]
        $h["wa"][$i] = $waa[1]
        $h["wa"][$bi] = $waa[2]

        $h["wdis"][$fi] = $h["rwr"][$fi]*$waa[0]
        $h["wdis"][$i] = $h["rwr"][$i]*$waa[1]
        $h["wdis"][$bi] = $h["rwr"][$bi]*$waa[2]
    } else {
        $h["wa"][$i] = $wa
        $h["wdis"][$i] = $h["rwr"][$i]*$wa
    }
}
}
#=====
function sum ($flg) {
    #
    # Sum angular and linear distances
    #
    # Parameters:
    #   flg = true/print title
    #
    Begin {
        $sum = 0.0
    }
    Process {
        $i = $_ -1
        $sum = $sum + $h["wdis"][$i] + $h["pdis"][$i]
    }
    End {
        if ($flg) {
            $t = "{0}" -f $h["title"]
            ("{0}:{1}" -f $t,$sum)
        } else {
            $sum
        }
    }
}
#=====

export-modulemember -function *

```

APPENDIX A.2 – p.m

```

% p.m - plot path definition points
%
% Point Order:
%   back point
%   center point
%   front point
%
close all

fn = {
'out0.txt';
'out1.txt';
'out2.txt';
'out3.txt';
'out4.txt';
'out5.txt';
}

for i=1:length(fn),
    yname = sprintf('%s',char(fn(i,1)))
    z = dlmread(yname)
    z1 = dlmread([yname '.bef'])

    figure(i)
    clf
    plot(z(:,1),z(:,2),'r')
    hold on
    plot(z1(:,1),z1(:,2),'b')

    indz = find(z(:,3)>0);
    for j=1:length(indz)
        z(indz(j),1), z(indz(j),2), z(indz(j),3)
        circ( z(indz(j),1), z(indz(j),2), z(indz(j),3),'g')
        circ(z1(indz(j),1),z1(indz(j),2),z1(indz(j),3) -0.01,'m')
    end
    axis equal

    oname = sprintf('f%0d',i)
    print('-dpng',oname);
end

```

APPENDIX A.3 – circ.m

```
% circ.m - plot circle
%
% Parameters:
% x = x coordinate of center point
% y = y coordinate of center point
% r = radius
% c = color (e.g. 'g')
%
function circ(x,y,r,c)
inc = 1*pi/180
t=[0:inc:2*pi]';
d=[x+r*cos(t),y+r*sin(t)];
plot(d(:,1),d(:,2),c)
```

APPENDIX A.4 – testdc.m

```

% testdc.m
%
global d
d=zeros(10,4);
v=[1,0,0,1,1,0;
1,0,0,0,1,0;
1,0,0,-1,1,0;
1,0,0,-1,0,0;
1,0,0,-1,-1,0;
1,0,0,0,-1,0;
1,0,0,1,-1,0;
1,0,0,1,0,0];

for z=1:length(v)
    x=v(z,1:3);
    y=v(z,4:6);
    dotcross(x,y,z,d);
end

fprintf('xy cross/dot, yx cross/dot\n');
fprintf('%f %f %f %f\n',[d(:,1)';d(:,2)';d(:,3)';d(:,4)']);

xy cross/dot, yx cross/dot
45.000000 45.000000 -45.000000 45.000000
90.000000 90.000000 -90.000000 90.000000
45.000000 135.000000 -45.000000 135.000000
0.000000 180.000000 -0.000000 180.000000
-45.000000 135.000000 45.000000 135.000000
-90.000000 90.000000 90.000000 90.000000
-45.000000 45.000000 45.000000 45.000000
0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000

```

APPENDIX A.5 – dotcross.m

```

% dotcross.m
%
function dotcross(x,y,z,d)
global d
tmp=asin(cross(x/norm(x,2),y/norm(y,2)))*180/pi
d(z,1) = tmp(3)
d(z,2)=acos(dot(x/norm(x,2),y/norm(y,2)))*180/pi
tmp=asin(cross(y/norm(y,2),x/norm(x,2)))*180/pi
d(z,3) = tmp(3)
d(z,4)=acos(dot(y/norm(y,2),x/norm(x,2)))*180/pi

```

TN 15-0709-01

“FRAME TWISTER” WAVE COURSE GENERATION IN 3D

July 9, 2015

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC)
Detroit Arsenal
Systems Integration & Engineering
ATTN: RDTA-SIE/MS157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

How can one generate the wave course defined in [1] and extract 2D profiles to use as inputs to laboratory N-post shakers.

PROBLEM SOLUTION:

Review of [1] in Appendix A.1 reveals a longitudinal sine wave whose amplitude is scaled linearly across its lateral width

$$wave(x, y) = A \sin(2\pi f x) \quad (1)$$

$$A = -sy \quad (2)$$

with

$$f = \frac{1}{T} \quad (3)$$

and

$$T = 28 \text{ feet}$$

$$y = -10 \text{ to } 10 \text{ feet}$$

$$s = \frac{\text{rise}}{\text{run}} = \frac{4 \text{ feet}}{-10 \text{ to } 10 \text{ feet}} = \frac{1}{5}$$

After converting to inches and choosing suitable increments (usually 3”) longitudinal (x) paths, see equation (1), are generated with changing lateral (y) amplitude, see equation (2), as outlined in Appendix A.2. The data is saved into three separate files---x (vector), y (vector), and z(x,y) (matrix with x as rows and y as columns).

Figure 1 shows a section of the surface using the “mesh” and “interp2” MATLAB [1] commands. Figure 2 shows longitudinal plots for maximum/minimum track width and for +/- half of the track width (red=left track, green=right track). Note the maximum/minimum amplitude of 24” and half cycle of 168” that matches the wave parameters in Appendix A.1. Figure 3 shows all longitudinal cross sections for 24” lateral samples. Figure 4 shows lateral cross sections for the maximum/minimum peaks. Compare the starting amplitude to Figure 1 where red and green are the left and right tracks, respectively. Figure 5

shows multiple lateral cross sections for the longitudinal distance. Note the reduction in vertical distance near the peak as can also be seen in Figure 1.

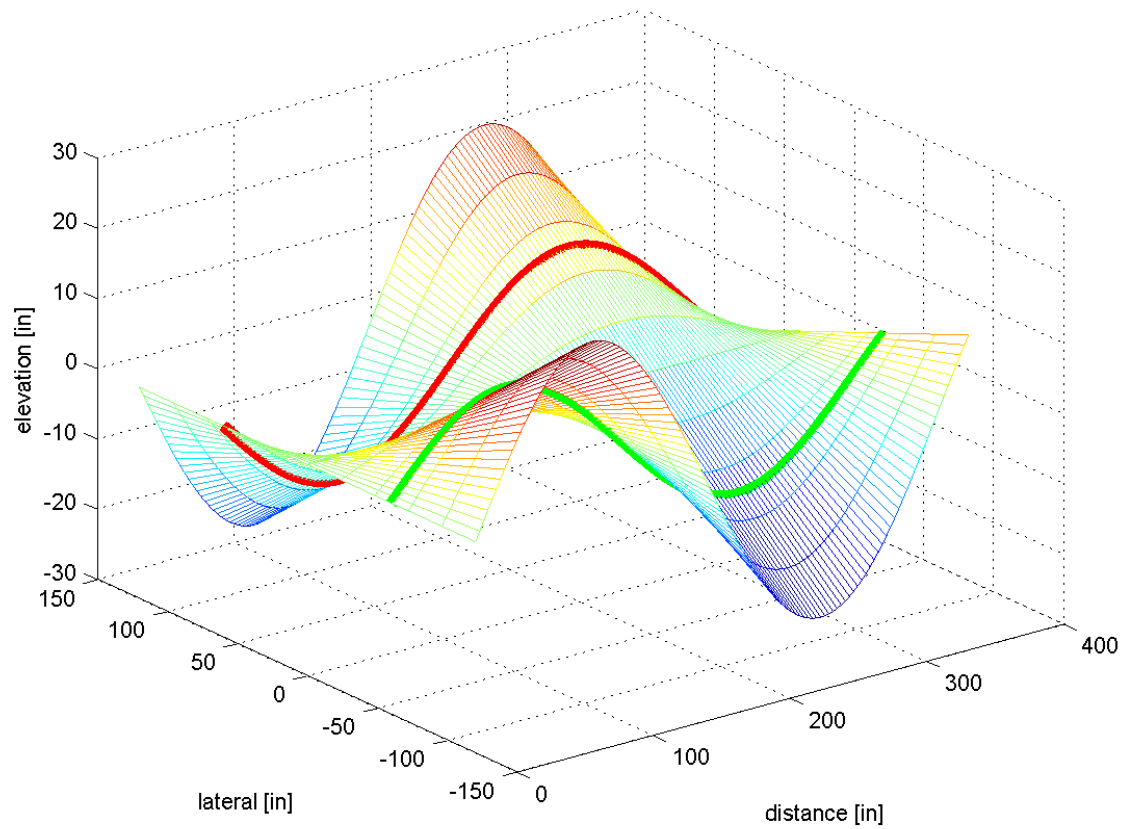


Figure 1 – Wave Course Surface Plot

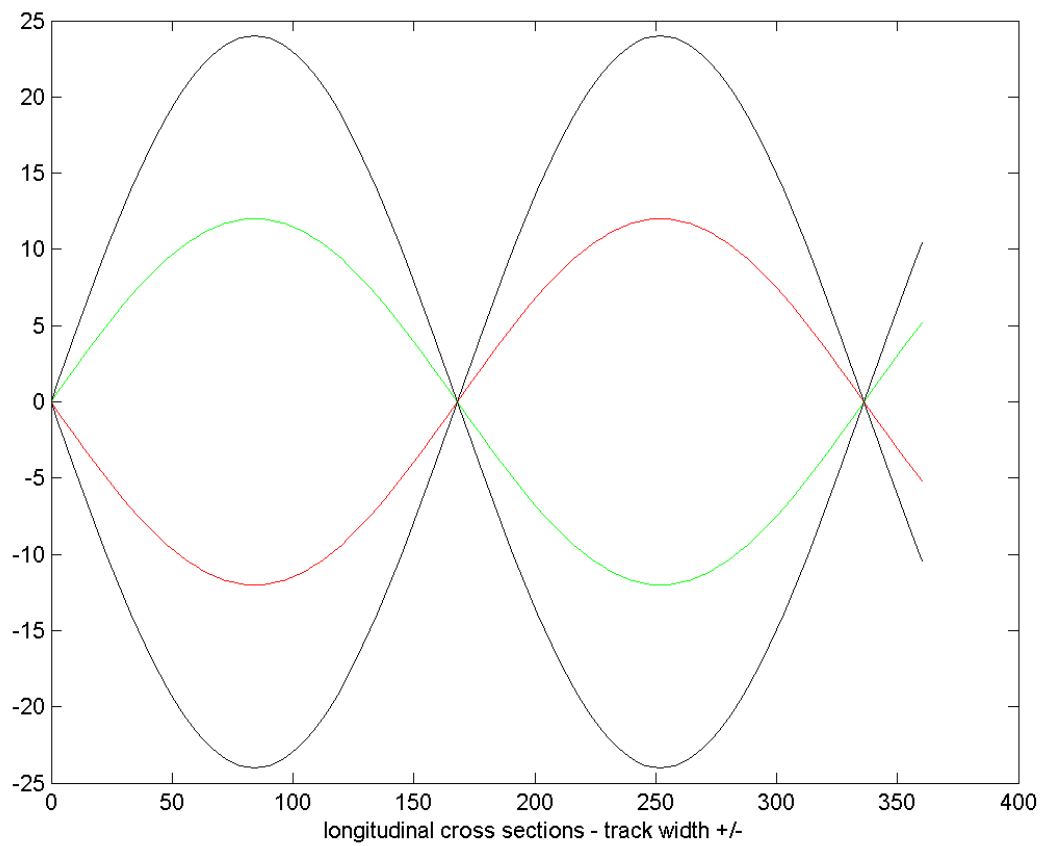


Figure 2 – Longitudinal Cross Sections – Track Width

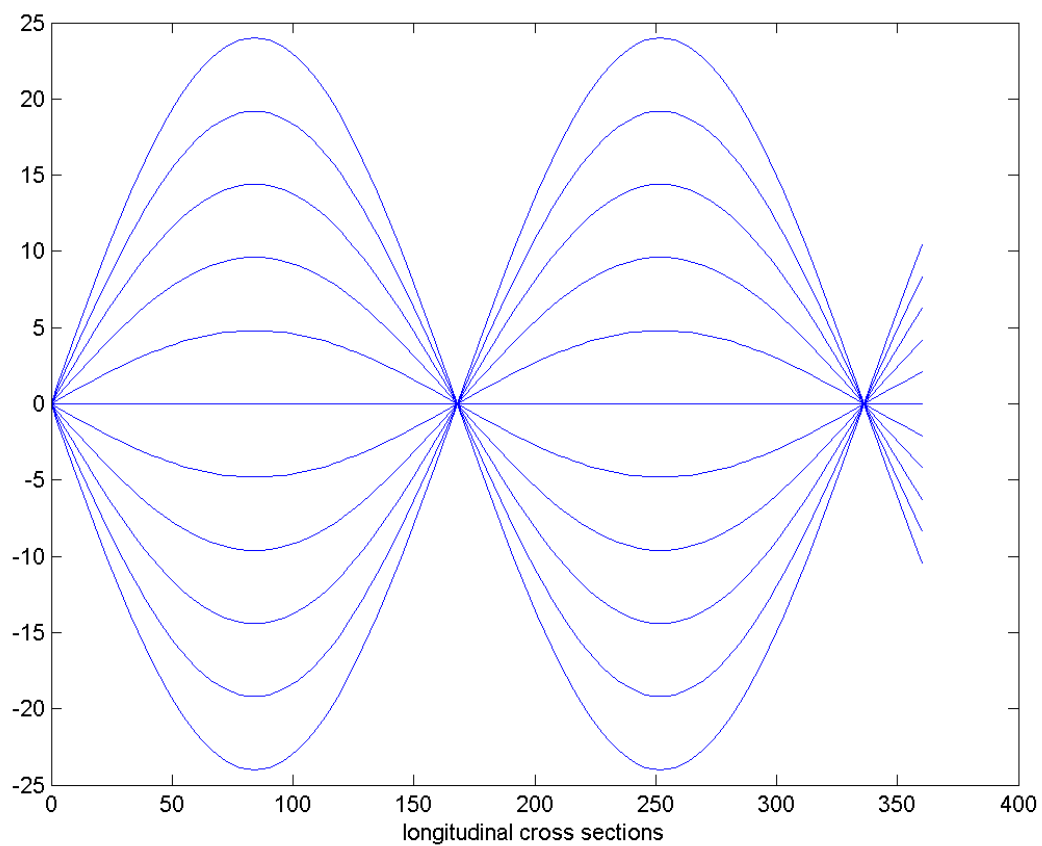


Figure 3 – Longitudinal Cross Sections

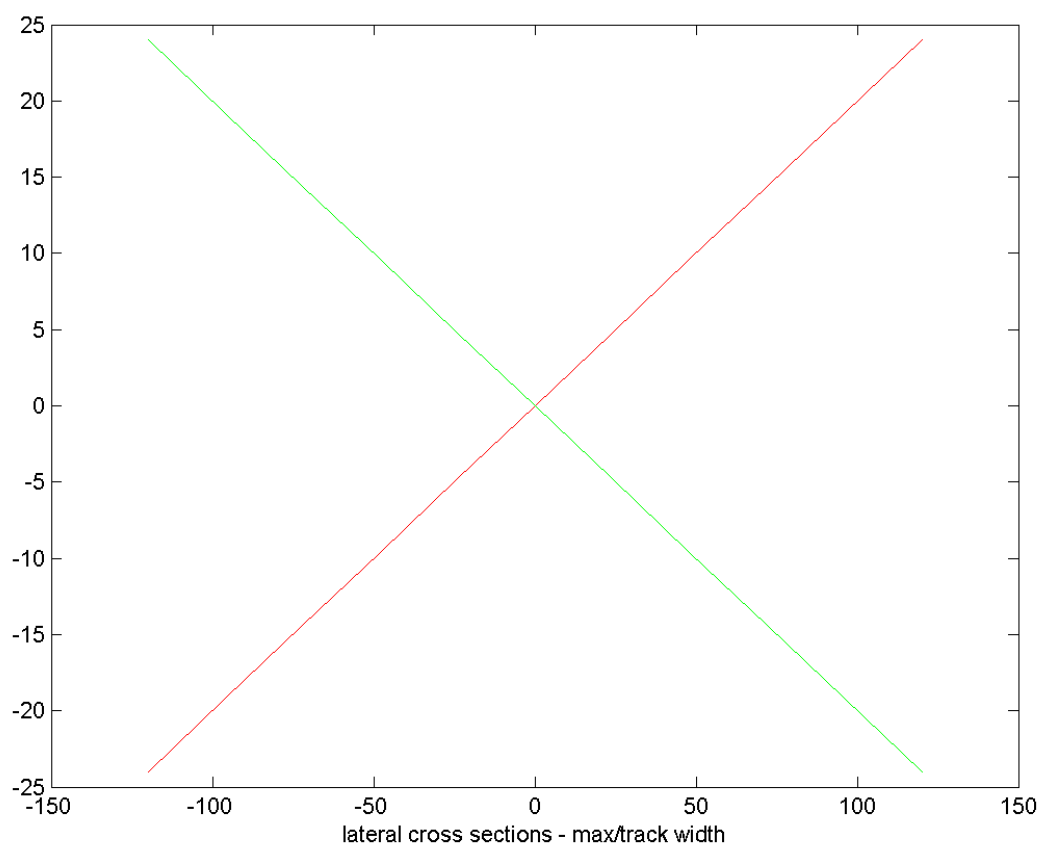


Figure 4 – Lateral Cross Sections – Track Width

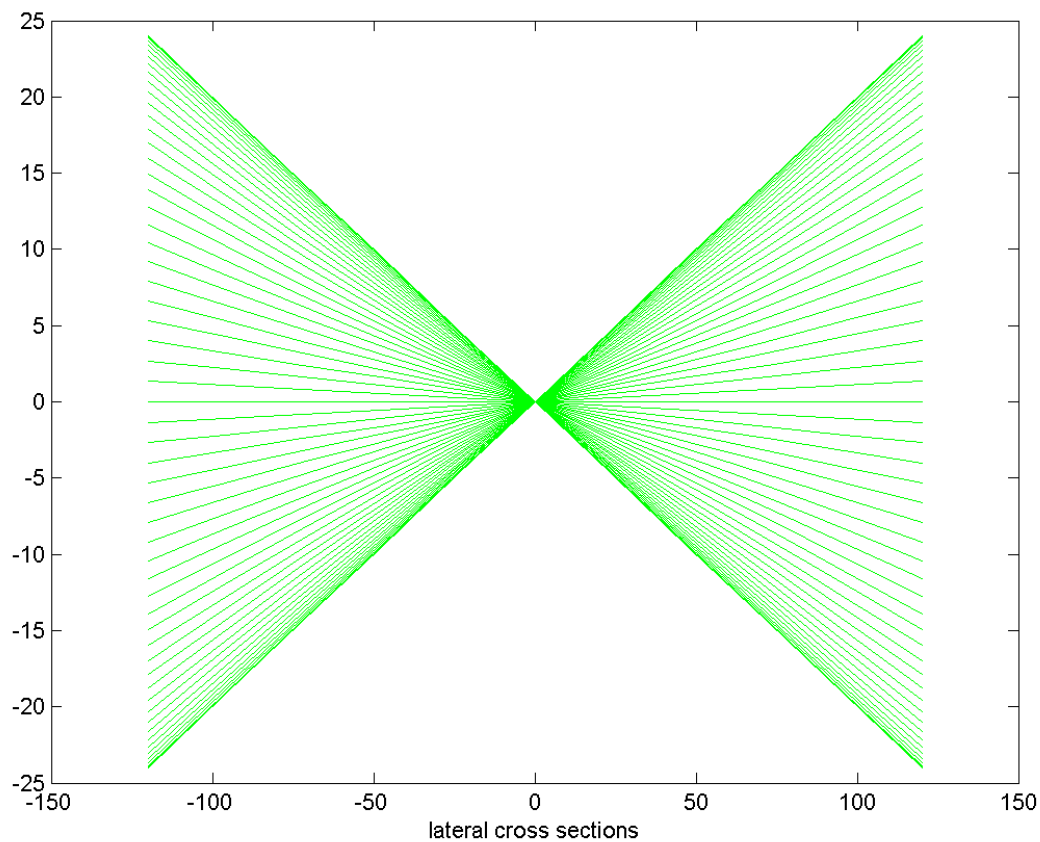


Figure 5 – Lateral Cross Sections

Figures 1 to 5 verify the replication of the wave “frame twister” course as defined in Appendix A.1. The “save” command is used to save the x, y, and z(x,y) data into separate files. Breaking the data up allows easier manipulation for other purposes. MATLAB commands required to replicate the above data and plots are

- figure, clf, mesh, plot3, plot, set, xlabel, ylabel, zlabel, print
- interp2
- find, max, min
- save

Implementation of most of these commands, except for the plotting related functions, can be easily replicated in other languages (.e.g. Windows Powershell, etc.).

Extracting 2D profiles can be accomplished using the “interp2” function as shown in Appendix A.2.

REFERENCE

- [1] Test Operations Procedure (TOP) 01-1-011A Vehicle Test Facilities at Aberdeen Test Center and Yuma Test Center (27 Feb 2012). DIST A (DTIC AD A557 002). Approved for public release: distribution unlimited.
- [2] MATLAB (www.mathworks.com)

APPENDIX A.1

TOP 01-1-011A
27 February 2012

2.6.21 Wave Course.

Also known as the frame twister, this concrete course (Figures 54 and 55) is designed to deflect the opposite wheels of a vehicle in alternately contrary directions. Articulation of the suspension increases with tread width. The course provides a severe test of differentials and universal joints, as well as suspensions. Distortion of vehicle bodies is checked by operating doors, dump bodies, engine hoods, etc., after stopping the vehicle with the suspension at extremes of vertical travel.

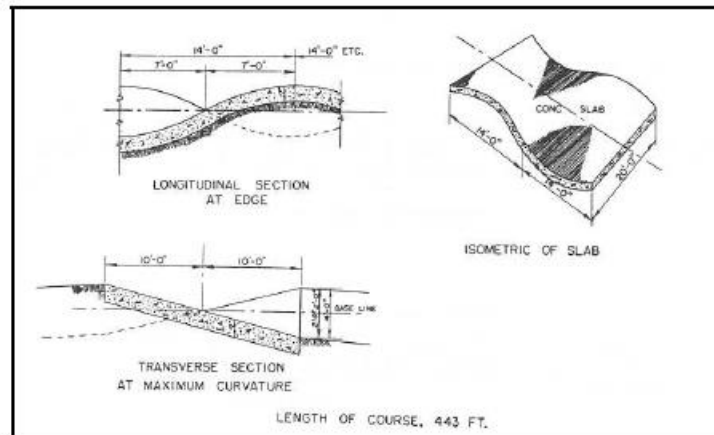


Figure 54. Sections of Wave Course.



Figure 55. Vehicle negotiating the Wave Course.

APPENDIX A.2

```

% w.m - wave
%
% Variables:
% l - length [feet]
% T = period [feet]
% f = frequency [cycles/inches]
% wd = +/- width [feet]
% incl = increment longitudinal [inches]
% incw = increment width [inches]
% x = longitudinal sample points [inches]

close all
clear all

% initialization
g = [];
l = 500; % ft
l = 30; % ft
T = 28; % ft
f = 1/(T*12); % cyc/in
wd = 10; % +/- 10 ft lateral
incl = 3; % in
incw = 3; % in
incw = 24; % in

x = [0:incl:l*12]'
wi = [-wd*12:incw:wd*12];
for y = wi,
    a = -(1/5)*y
    s = a*sin(2*pi*f.*x)
    g = [g;s']; % x across row, y down column
end
wip = wi' % make vector

%===surface
figure(1)
clf
mesh(x,wi,g) % mesh(x,y,z) => x=columns, y=rows, z(y,x)
xlabel('distance [in]')
ylabel('lateral [in]')
zlabel('elevation [in]')
hold on
% example with interp1
%ind1 = interp1(wip,[1:length(wip)]',-wd*12/2,'nearest')
%h = plot3(x,wip(ind1)+0*x,g(ind1,:),'r')

% x=row, y=col
pl = interp2(x,wip,g,x',wd*12/2) % half track width
pr = interp2(x,wip,g,x',-wd*12/2) % half track width
h = plot3(x,wd*12/2+0*x,pl,'r')
set(h(1),'LineWidth',10)
h = plot3(x,-wd*12/2+0*x,pr,'g')
set(h(1),'LineWidth',10)
%print('-dpng','f1')
...

```

```

...
%===longitudinal plot - track width
figure(2)
clf
plot(x,pl,'r')
hold on
plot(x,pr,'g')
plm = interp2(x,wip,g,x',wd*12) % max track width
prm = interp2(x,wip,g,x',-wd*12) % max track width
plot(x,plm,'k')
plot(x,prm,'k')
xlabel(sprintf('longitudinal cross sections - track width +/-%6.1',wd*12/2))
%print('-dpng','f2')

%===longitudinal cross sections
figure(3)
clf
for j = 1:length(wip),
    plot(x,g(j,:))
    hold on
end
xlabel('longitudinal cross sections')
%print('-dpng','f3')

%===lateral cross sections - max/track width
figure(4)
clf
ilmax = find(max(pl)==pl)
ilmin = find(min(pl)==pl)
plot(wip,g(:,ilmax),'r')
hold on
plot(wip,g(:,ilmin),'g')
xlabel('lateral cross sections - max/track width')
%print('-dpng','f4')

%===lateral cross sections
figure(5)
clf
for j = 1:length(x),
    plot(wip,g(:,j),'g')
    hold on
end
xlabel('lateral cross sections')
%print('-dpng','f5')

%===save
save('wavex.txt','x','-ascii','-double'); % column vector
save('wavey.txt','wip','-ascii','-double'); % column vector
save('wavez.txt','g','-ascii','-double'); % matrix (x=rows, y=columns)

```

TN 15-0710-01

ROAD SURFACE GENERATION IN RDF FORMAT

July 10, 2015

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC)
Detroit Arsenal
Systems Integration & Engineering
ATTN: RDTA-SIE/MS157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

How can 3D road surface data be used to create a RecurDyn Road Data File (RDF)?

PROBLEM SOLUTION:

The Road Data File (RDF) format is specified for the RecurDyn multibody dynamics software [1] in its online documentation [2]. Windows PowerShell [3] will be used to process the road data. An example road surface data set is shown below from [4, 5].

wavex.txt

```
0.0000000000000000e+000
3.0000000000000000e+000
6.0000000000000000e+000
9.0000000000000000e+000
...
3.5100000000000000e+002
3.5400000000000000e+002
3.5700000000000000e+002
3.6000000000000000e+002
```

wavey.txt

```
-1.2000000000000000e+002
-9.6000000000000000e+001
-7.2000000000000000e+001
-4.8000000000000000e+001
-2.4000000000000000e+001
0.0000000000000000e+000
2.4000000000000000e+001
4.8000000000000000e+001
7.2000000000000000e+001
9.6000000000000000e+001
1.2000000000000000e+002
```


wavez.txt

```
0.0000000000000000e+000 1.3456907336926027e+000 2.6871474264793882e+000 ...
0.0000000000000000e+000 1.0765525869540824e+000 2.1497179411835110e+000 ...
0.0000000000000000e+000 8.0741444021556164e-001 1.6122884558876329e+000 ...
0.0000000000000000e+000 5.3827629347704120e-001 1.0748589705917555e+000 ...
...
```

The road surface data needed to create the RDF file is specified in the format (4 lines per road surface) described below in “filelist_rdfsrf”.

filelist_rdfsrf

```
# Format:
# line 1: output file name
# line 2: x input data file
# line 3: y input data file
# line 4: z(x,y) input data file
# repeat for each data set
#
wave.rdf
C:\..\surf\wavex.txt
C:\..\surf\wavey.txt
C:\..\surf\wavez.txt
#
...
```

The x and y data are converted to “Double” arrays using “[Double]::Parse(\$val)”. For the z data, each row is split, the values converted to “Double”, and then stored in a “Double[,]” array indexed by z[y,x]. Once this is complete, the file header is output (see Appendix A.1) and the NODES and ELEMENTS sections are processed as described in their respective sections.

Figure 1 details the node and element numbering convention used for three different sizes---gridded data is assumed as well as triangular elements. Note that the actual index numbers are zero based but not the node and element numbering.

NODES Section

The total number of nodes is

$$node\ total = (x.count)(y.count)$$

while each node number is calculated from the x and y index (zero based), i and j respectively.

$$node\ number = j * (x.count) + (i + 1) \quad (1)$$

For each node, its number and coordinate values are specified as

$$node\ number, x[i], y[j], z[j, i]$$

creating one line of output per node.

ELEMENTS Section

The total number of elements is

$$element\ total = rows(x.count - 1)2$$

where

$$rows = y.count - 1.$$

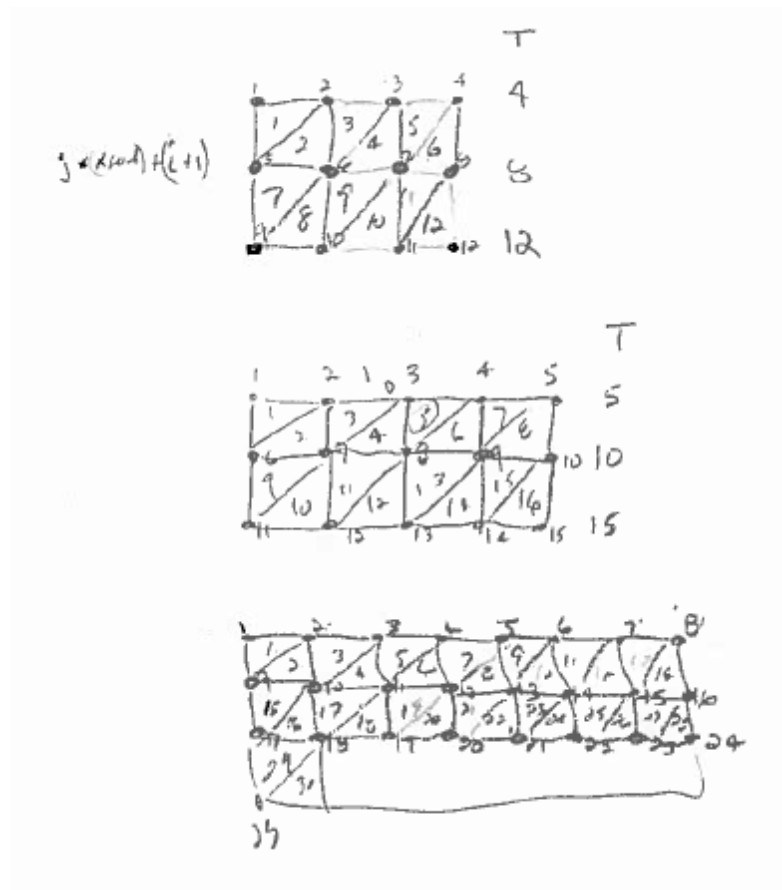


Figure 1 – Node/Element Numbering

The nodes defining each element are related to the element number in the following manner---for each element the “row”, “col”, and “ul” (upper/lower element indicator) values are defined as

$$row = trunc\left(\frac{i - 1}{2(x.count - 1)}\right) + 1 \quad (2)$$

$$ul = i \bmod 2 \quad (3)$$

$$col = i - (row - 1)(x.count - 1)2 \quad (4)$$

where “trunc” indicates the integral part of a specified double-precision floating-point number and is implemented with “[Math]::Truncate()” while “mod” indicates the modulus/remainder and is implemented with the “%” operator in [3]. The “ul” value indicates an upper or lower triangular element in the element square, see Figure 2, between the upper and lower node points. Depending on “ul” the initial “col” value is recalculated as

$$col = \begin{cases} trunc\left(\frac{col - 1}{2}\right) + 1, ul = 1 \text{ (lower element)} \\ trunc\left(\frac{col}{2}\right), ul \neq 1 \text{ (upper element)} \end{cases} \quad (5)$$

with upper/lower triangular elements having the same element column number.

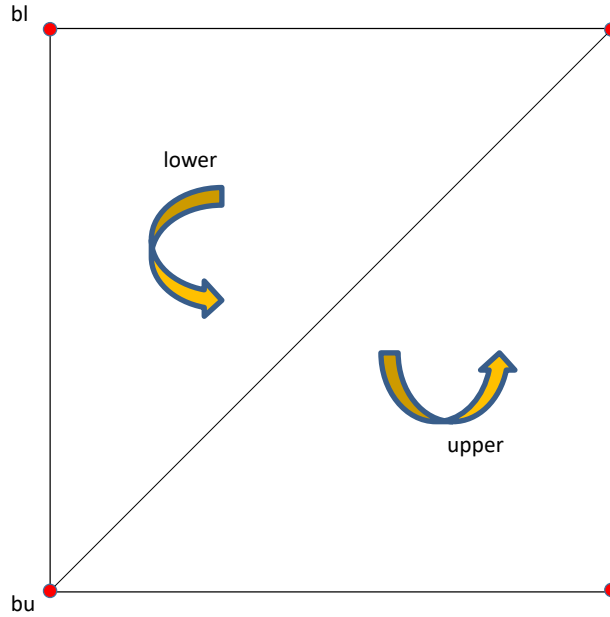


Figure 2 – Element Column and Base Node Definition

Figure 2 shows the relationship between the “bl” and “bu” nodes, element column, and upper and lower triangular elements. In the element node sequence, counter clockwise indicates the top side or “up” surface. These values are calculated as

$$\begin{aligned} bl &= (row - 1)x.count + col \\ bu &= (row)x.count + col \end{aligned} \quad (6)$$

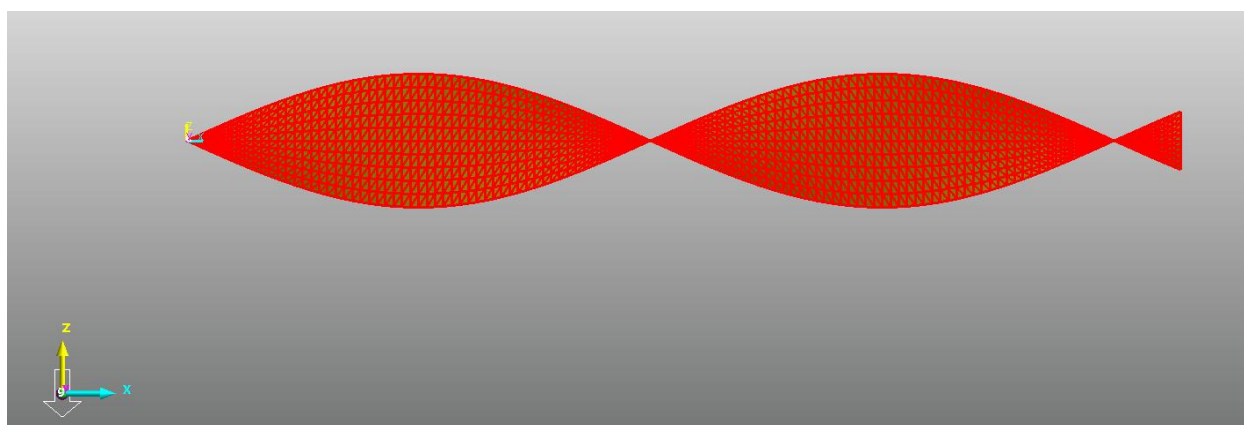
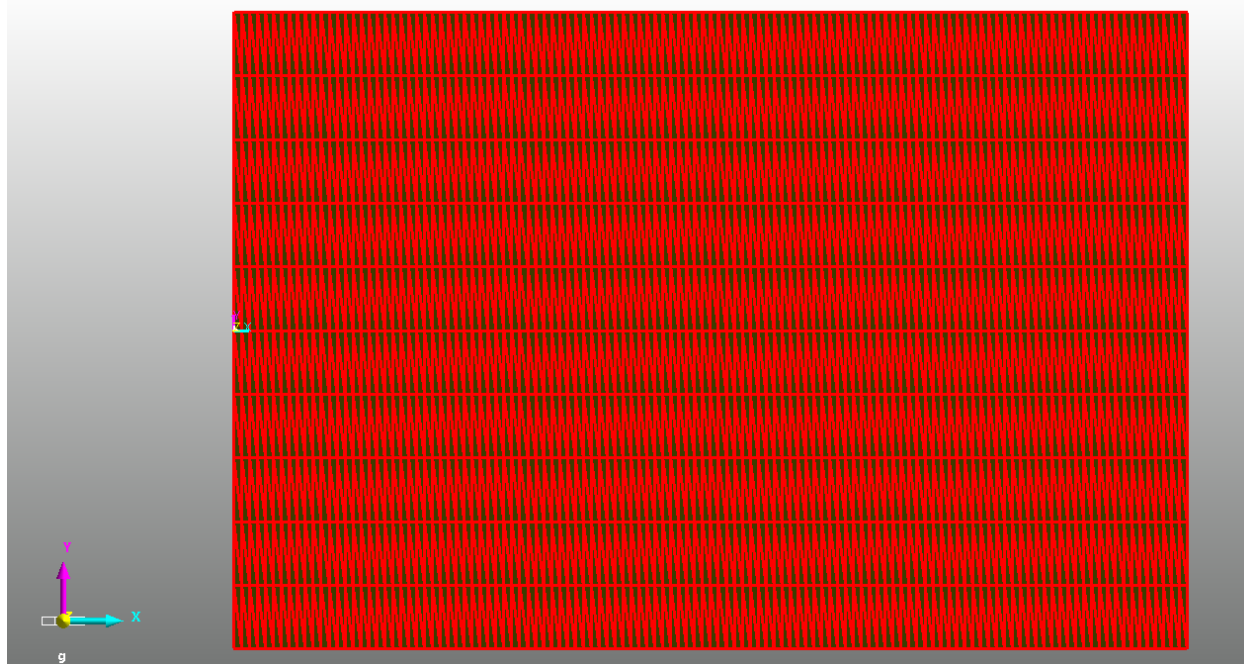
$$element\ node\ sequence = \left\{ \begin{array}{l} \left. \begin{array}{l} n1 = bl \\ n2 = bu \\ n3 = bl + 1 \end{array} \right\}, ul = 1\ (lower\ element) \\ \left. \begin{array}{l} n1 = bu \\ n2 = bu + 1 \\ n3 = bl + 1 \end{array} \right\}, ul \neq 1\ (upper\ element) \end{array} \right\} \quad (7)$$

For each element, its node sequence is specified as

$$n1, n2, n3$$

creating one line of output per element.

Figures 3 – 5 show the top, side, and front views a the example course defined in Appendix A.2 [4] and created in [5] that was imported into RecurDyn [1]. Note the triangular mesh elements.



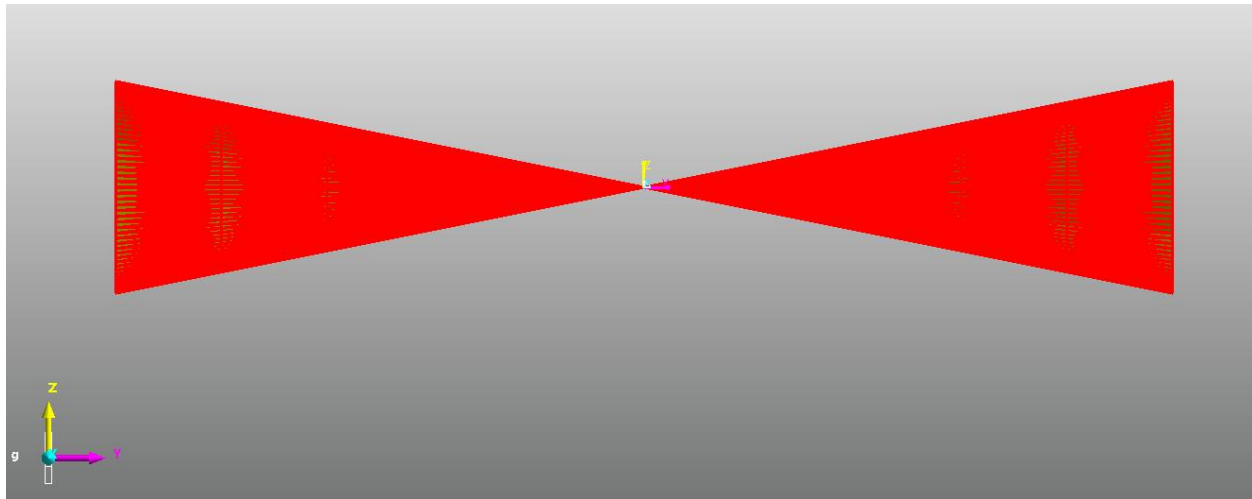


Figure 5 – Front View (Lateral)

Full details of the RDF file creation from 3D road surface data is contained in “rdfs surf.ps1” in Appendix A.3.

REFERENCE

- [1] RecurDyn (<https://functionbay.com/en>)
- [2] Road Data File (RDF)
(<https://functionbay.com/documentation/onlinehelp/default.htm#!Documents/rdffileformat.htm>)
- [3] Windows PowerShell (http://en.wikipedia.org/wiki/Windows_PowerShell)
- [4] Test Operations Procedure (TOP) 01-1-011A Vehicle Test Facilities at Aberdeen Test Center and Yuma Test Center (27 Feb 2012). DIST A (DTIC AD A557 002). Approved for public release: distribution unlimited.
- [5] TN 15-0709-01 FRAME TWISTER WAVE COURSE GENERATION IN 3D

APPENDIX A.1 – Road Data File (RDF) Format

The RDF File Format

Page 1 of 4

The RDF File Format

The format for **Road Data File (RDF)** has two types. One is a **RecurDyn Generated Road File** format, the other is a **user-defined Road file** format.

RecurDyn Generated Road Data File Format

This road data file is automatically generated by **RecurDyn** after simulation, it also can be imported by **Import Road**.

The following table explains the format of the **Road Data** file.

GENDATA	The information about the road in the general road profile format.
X_SCALE	
real	Conversion Factor.
Y_SCALE	
real	Conversion Factor.
Z_SCALE	
real	Conversion Factor.
ORIGIN	Coordinates of the road origin with respect to the global reference frame.
x, y, z	
UP	A vector that indicates the upward side of the road in the global reference frame.
x, y, z	
ORIENTATION	Transformation matrix which is oriented to the road coordinate system with respect to the global reference frame.
x1, y1, z1	
x2, y2, z2	
x3, y3, z3	
NODES	
Number of nodes (N)	Integer number of nodes.
Node_1, x_1, y_1, z_1	Location of Node 1.
Node_2, x_2, y_2, z_2	Location of Node 2.
Node_n, x_n, y_n, z_n	Location of Node n.
ELEMENTS	
Number of elements (Ne)	Integer number of elements.

NodeA_1, NodeB_1, NodeC_1, U0_1, U1_1	Nodes which are consisted of triangle element 1.
NodeA_2, NodeB_2, NodeC_2, U0_2, U1_2	Nodes which are consisted of element 2.
NodeA_n, NodeB_n, NodeC_n, U0_n, U1_n	Nodes which are consisted of element n.

• Example

```

!
!  ROAD DATA FILE  !
!
!
!                               RecurDyn_Data_Name='E:\r_z_z_z_.rdf'
!
METHOD
GENDATA2
!
!
X_SCALE
1.
Y_SCALE
1.
Z_SCALE
1.
!
!Origin is located at the following global coordinates in the data set.
!
ORIGIN
0 0 0
!
!
UP
0, 0, 1
!
ORIENTATION
1 0 0
0 1 0
0 0 1
!
!Coordinates for the node points on road
!
NODES
6
0 0.8 0 0
1 0.7 -0.5 0
2 0 -0.1 0
3 0.1 0.4 0
4 -1.3 0.2 0
5 -1.2 -0.3 0
ELEMENTS
4
0 1 2
2 3 0
4 3 2
2 5 4

```

APPENDIX A.2 – Wave Course

TOP 01-1-011A
27 February 2012

2.6.21 Wave Course.

Also known as the frame twister, this concrete course (Figures 54 and 55) is designed to deflect the opposite wheels of a vehicle in alternately contrary directions. Articulation of the suspension increases with tread width. The course provides a severe test of differentials and universal joints, as well as suspensions. Distortion of vehicle bodies is checked by operating doors, dump bodies, engine hoods, etc., after stopping the vehicle with the suspension at extremes of vertical travel.

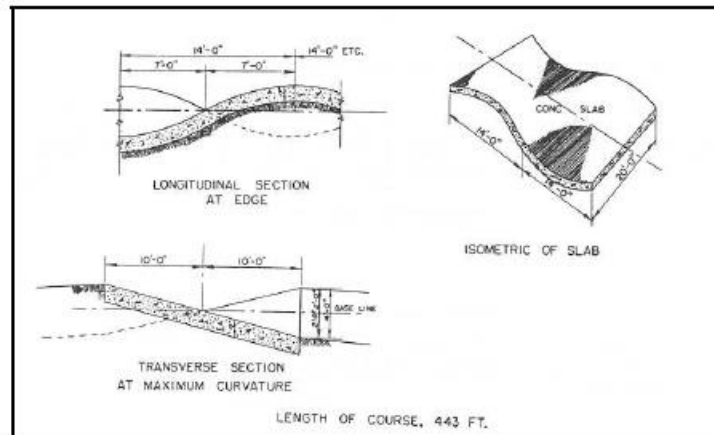


Figure 54. Sections of Wave Course.



Figure 55. Vehicle negotiating the Wave Course.

APPENDIX A.3 – rdfsurf.ps1

```

# rdfsurf.ps1
#
# .\rdfsurf [filelist]
#
param($file)
#
# file = data file list to convert, multiples of 4 lines (output file name, x data, y data
#       z(x,y) data
#
$hdr = @"
! -----
! | RECURDYN ROAD FILE |
! -----
!
!   RecurDyn_Data_Name='wave.rdf'
!
METHOD
GENDATA2
!
!
X_SCALE
1.
Y_SCALE
1.
Z_SCALE
1.
!
!Origin is located at the following global coordinates in the data set.
!
ORIGIN
0 0 0
!
UP
0, 0, 1
!
ORIENTATION
1 0 0
0 1 0
0 0 1
!
!Coordinates for the node points on road
!
"@
...

```

```

$H = @{}
$fd = (get-content $file) | where-object {$_[0] -ne "#"}
$fc = 0
for ($i = 0; $i -lt $fd.count; $i = $i + 4) {
    $h["out"] = $fd[$i]
    $h["inx"] = $fd[$i+1]
    $h["iny"] = $fd[$i+2]
    $h["inz"] = $fd[$i+3]

    $xs = get-content $h["inx"]
    $x = new-object "Double[]" $xs.count
    for ($i = 0; $i -lt $xs.count; $i = $i + 1) {
        $val = $xs[$i]
        $x[$i] = [Double]::Parse($val)
    }

    $ys = get-content $h["iny"]
    $y = new-object "Double[]" $ys.count
    for ($i = 0; $i -lt $ys.count; $i = $i + 1) {
        $val = $ys[$i]
        $y[$i] = [Double]::Parse($val)
    }

    $zs = get-content $h["inz"]
    $z = new-object "Double[,]" $ys.count, $xs.count
    for ($j = 0; $j -lt $ys.count; $j = $j + 1) {
        $l1 = -split $zs[$j]
        if ($l1.count -ne $xs.count) {
            write-host "z-x data not match xc"
            exit
        }
        for ($i = 0; $i -lt $l1.count; $i = $i + 1) {
            $val = [Double]::Parse($l1[$i])
            $z[$j,$i] = $val
        }
        # $host.UI.RawUI.ReadKey()
    }
}
...

```

```

$0 = @()
$0 = $0 + $hdr
$0 = $0 + "NODES"
$0 = $0 + ("0:D)" -f (($x.count)*$y.count ) )

for ($j = 0;$j -lt $y.count; $j = $j + 1) {
  for ($i = 0; $i -lt $x.count; $i = $i + 1) {
    $num = ($j*$x.count)+($i+1)
    $0 = $0 + ("0:D" {1:F8} {2:F8} {3:F8}" -f $num,$x[$i],$y[$j],$z[$j,$i])
  }
}

$0 = $0 + "ELEMENTS"
$rows = $y.count-1
$set = $rows*($x.count-1)*2
$0 = $0 + ($set -f "0:N")
for ($i = 1; $i -le $set; $i = $i + 1) {
  $row = [Math]::Truncate( ($i - 1)/(2*($x.count-1)) ) + 1
  $ul = $i % 2
  $col = $i - ($row-1)*($x.count-1)*2
  if ($ul -eq 1) {
    # lower
    $col = [Math]::Truncate( ($col-1)/2) + 1
  } else {
    # upper
    $col = [Math]::Truncate($col/2)
  }
  $bl = ($row-1)*$x.count + $col
  $bu = ($row)*$x.count + $col
  if ($ul -eq 1) {
    $p1 = $bl
    $p2 = $bu
    $p3 = $bl + 1
  } else {
    $p1 = $bu
    $p2 = $bu + 1
    $p3 = $bl + 1
  }
  $0 = $0 + ($p1 -f "0:N" + " " + ($p2) -f "0:N" + " " + ($p3) -f "0:N")
}
$fo = $h["out"]
$0 | out-file -filepath $fo -encoding ASCII
$fc = $fc + 1
write-host "Files:" $fc

```

TN 15-0930-01

ABSORBED POWER CALCULATION IN THE FREQUENCY DOMAIN

September 30, 2015

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC)
Detroit Arsenal
Systems Integration & Engineering
ATTN: RDTA-SIE/MS157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

A documented method to calculate vertical absorbed power was needed to define performance requirements for the M870A4 Semitrailer. Test Operating Procedure (TOP) 01-1-014A Ride Dynamics and Evaluation of Human Exposure to Whole Body Vibration defined coefficients for this but did not give full implementation details.

PROBLEM SOLUTION:

Review of [2] and [3] led to the development of a script using [4] to calculate absorbed power in the frequency domain---multiplying the acceleration power spectral density by the appropriate weighting factors. The values obtained using this method were then compared to a time based method---filtering the acceleration time history. An outline of the process is presented below with details in Appendix A.3.

- define $y = [\text{time, acceleration in g's}]$
- select $t > 1.0$ to remove initial transients
- plot full and selected signals of y
- filter acceleration ($y(:,2)$) using time based absorbed power filter (standard definition input needs to be in ft/s^2)
- plot instantaneous absorbed power
- define filter for each frequency (see Appendix A.1)
- create PSD of acceleration ($y(:,2)$) (standard definition input needs to be in ft/s^2)
- use interpolation to get coefficients at each frequency
- multiply PSD spectra at that frequency by the associated coefficient
- integrate the weighted PSD over the frequency (dF)
- compare to ending time based absorbed power value

The Simulink Model of the transfer function used in Appendix A.3 to calculate the time domain absorbed power is summarized in Appendix A.4 (see [5,6,7]). Figures 1 to 5 show acceleration and instantaneous absorbed power signals using the time based approach for several test signals.

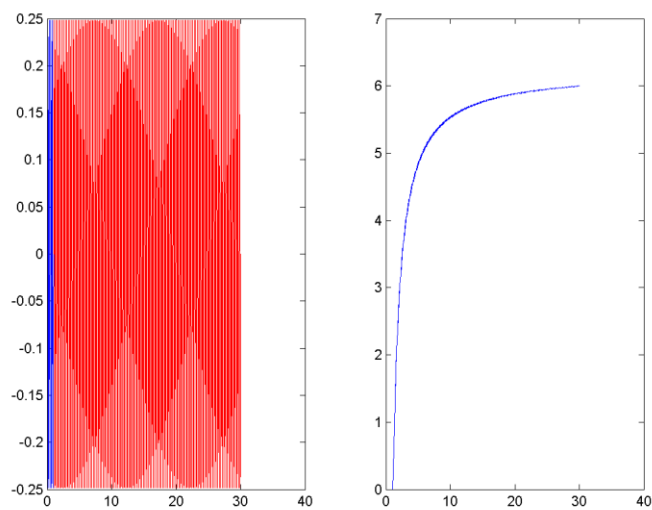


Figure 1 – Sinusoidal Test $((8/32.2)*\sin(2*3.14159*4*t))$; %~0.25 4Hz g's)

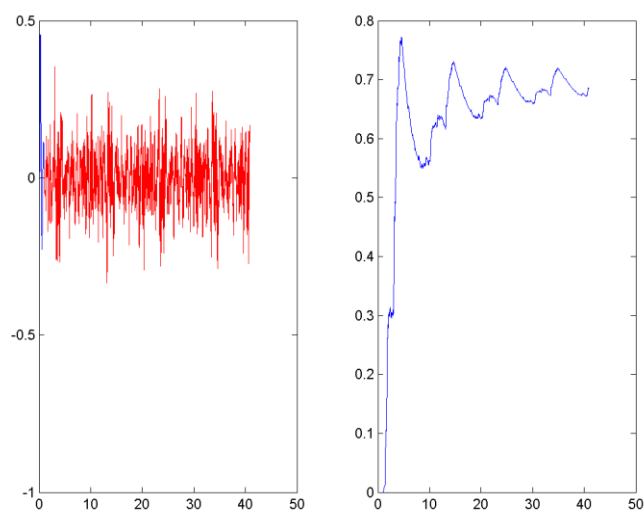


Figure 2 – RMS 0.70 Test ('..\modellib\r2\out_deskr_rms070.txt')

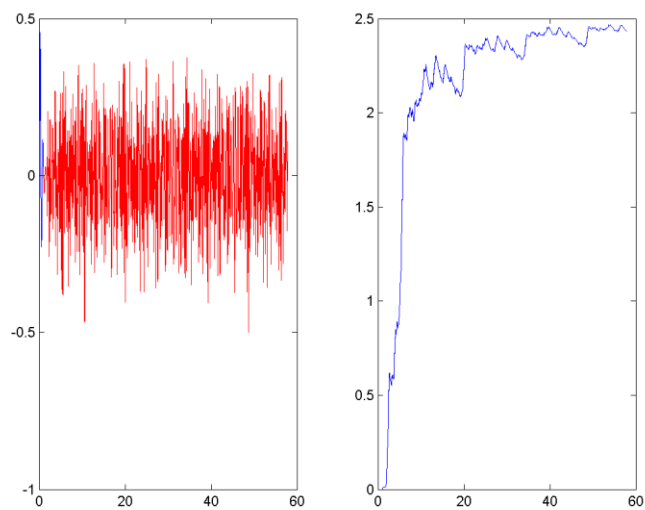


Figure 3 – RMS 1.50 Test ('..\modellib\r2\out_deskr_rms150.txt')

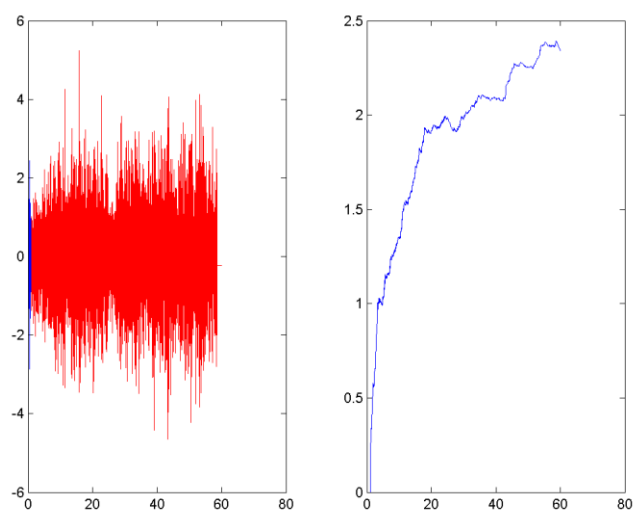


Figure 4 – Belgian Block Full Test ('..\modellib\r2\BB_20mph_full_ext_ASCII.txt')

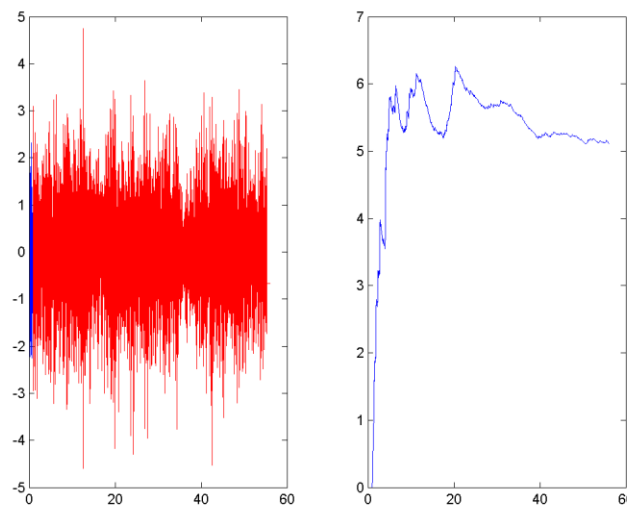


Figure 5 – Belgian Block Empty Test ('..\model\libr2\BB_20mph_empty_ext_ASCII.txt')

Table 1 compares absorbed power output values calculated with both methods and shows the close agreement between them.

Table 1 – Absorbed Power (Watts)	
Time Based	Frequency Based
5.9943	6.0349
0.6856	0.6741
2.4349	2.5010
2.3422	2.5602
5.1166	5.1858

REFERENCE

- [1] Reference Internal Project 15-018/518 M870 TRAILER PERFORMANCE.
- [2] Test Operating Procedure (TOP) 01-1-014A Ride Dynamics and Evaluation of Human Exposure to Whole Body Vibration (3 Apr 2012). DIST A (DTIC AD A559 451). Approved for public release: distribution unlimited.
- [3] Lee, Pradko. Analytical Analysis of Human vibration. Mobility systems Laboratory, U.S. Army Tank-Automotive Command. SAE Paper 680091. Jan 1968.
- [4] MATLAB. (www.mathworks.com)
- [5] TN 10-1021-01 ABSORBED POWER CALCULATION
- [6] TN 12-1219-01 ABSORBED POWER FUNCTION USING MATLAB'S ODE45
- [7] Lee and Lins. "Human Vibration Measuring Instrument". Army Tank-Automotive Command Warren, MI. 1973. DIST A (DTIC AD 0785 648). Approved for public release: distribution unlimited. (see pg. 207).

APPENDIX A.1 – TOP 01-1-014A - Appendix C

TOP 01-1-014A
29 November 2011

dynamics and suspension characteristics, additional data runs can be performed with a single wheel or track path crossing the obstacle while the other wheel or track path remains on level ground.

3. VIBRATION EVALUATION.

a. Vibration evaluation will be performed using two techniques; an ISO technique which deals with health effects of exposure to vibration, and an absorbed power technique which is used to describe speed limiting effects over rough terrain.

b. The principal factors that combine to determine the degree to which human exposure to whole body vibration will be acceptable is described in ISO 2631-1. Four possible effects of vibration include Degraded Health, Comfort, Perception and Motion Sickness. The frequency ranges of these effects are:

(1) 0.5 Hz to 80 Hz for Degraded Health, Comfort and Perception.

(2) 0.1 Hz to 0.5 Hz for Motion Sickness.

c. Unless required by the test plan, only the issue of degraded health will be evaluated. This type of vibration is transmitted to the human body as a whole through the supporting surfaces of the buttocks, back, and feet of a seated person in a moving wheeled or tracked vehicle. Vibration is measured according to the coordinate system originating at a point from which vibration enters the human body. The coordinate system for the alignment of the vibration transducers is shown in Figure 2. The three principal areas of contact for seated persons are: (1) the supporting seat surface, (2) the seat back, and (3) the feet, but only the seat data are of interest for degraded health analysis. Seat data are also the basis for the absorbed power analysis. Vibration transmitted to the body through a non-rigid material, like a seat cushion, is measured with the transducer interposed between the person and the principal contact areas of the surface. This is achieved by mounting the transducers within a ride quality pad as described in Figure 1.

4. ANALYSIS TECHNIQUES.

4.1 Absorbed Power Technique.

a. Another technique for evaluating human response to vibration or ride quality is the computation of absorbed power. It is a measure of the rate at which energy is absorbed by a human subjected to ride vibration. It is accepted as a measure of human tolerance to vibration for military vehicles negotiating rough terrain. The absorbed power for a given location and axis is computed by multiplying the acceleration power spectral density spectrum by the appropriate transfer function and integrating the resultant spectrum. An advantage of this approach is that average absorbed power is a scalar quantity and can be summed in complex multi-degree of freedom systems to yield a single value describing the total average absorbed power. Typically a standard ride quality test procedure involves determining the speed at which the vertical average absorbed power reaches an upper limit of 6 watts for different types of terrain at selected crew

TOP 01-1-014A
29 November 2011

location seats. The terrain is usually characterized by the surface roughness reported as inches or centimeters rms. The vehicle speed as a function of terrain roughness obtained from this process is used as one of the mobility limiting factors in the North Atlantic Treaty Organization (NATO) Reference Mobility Model. Military vehicle specifications frequently include ride quality requirements based on absorbed power (e.g., the vehicle must be capable of producing a ride of 6 watts or less at the driver's seat in the vertical axis for a given speed and surface roughness).

- b. The absorbed power in each axis will be calculated from:

$$P = \sum_{i=1}^n (C_i) A_i^2$$

where:

P = Absorbed power, watts.

A_i = rms acceleration in ft/sec² within the i^{th} spectral band.

$C_i = K_1 K_0 (F_1 F_4 - F_2 F_3) / (F_3^2 + W_i^2 F_4^2)$.

W_i = Frequency, radians/second.

F and K values = Calculated from Appendix C.

- c. The result is a frequency weighting, scaling and integration of the acceleration power spectral density. The weighting functions are shown (by axis) in Figure 3. The factors have been normalized to a value of 1.0 to show frequency without the appropriate scaling.

- d. For each test course, the absorbed power will be plotted as a function of speed. Specification requirements are generally written in terms of the driver's vertical absorbed power (only), but the following technique will be applied to other locations and axes, if applicable. A non-linear interpolation or a power law ($y = ax^b$, linear fit in a log-log domain,) or nth degree polynomial curve fit will be performed and the speed at which an absorbed power of 6 watts is achieved, will be interpolated from the data set. Absorbed power values of less than 1 watt or greater than 10 watts will not be used in the curve fit process, if possible. The 6-watt speed will then be plotted as a function of terrain roughness rms (a specific value for each test course) to determine a ride quality curve for the vehicle.

TOP 01-1-014A
29 November 2011

APPENDIX C. ABSORBED POWER CONSTANTS.

Longitudinal (X)

K_0	4.3532
K_1	1.356
W_i	$2\pi f_i$
F_1	1.0
F_2	0.219106
F_3	$-0.0185309W_i^2 + 1$
F_4	$-0.00061893W_i^2 + 0.219106$

Note: f_i = Center frequency (Hz) of i^{th} spectral band.

Transverse (Y)

K_0	4.353
K_1	1.356
W_i	$2\pi f_i$
F_1	$0.24052124 \times 10^{-3} W_i^4 - 0.066974483 W_i^2 + 1$
F_2	$0.57384538 \times 10^{-5} W_i^4 - 0.50170413 \times 10^{-2} W_i^2 + 0.33092592$
F_3	$-0.14979958 \times 10^{-5} W_i^6 + 0.0010088882 W_i^4 - 0.10108617 W_i^2 + 1$
F_4	$-0.1713749 \times 10^{-7} W_i^6 + 0.53137351 \times 10^{-4} W_i^4 - 0.011096507 W_i^2 + 0.33092592$

Note: f_i = Center frequency (Hz) of i^{th} spectral band.

TOP 01-1-014A
29 November 2011

APPENDIX C. ABSORBED POWER CONSTANTS.

Vertical (Z)

K_0	4.3537
K_1	1.356
W_i	$2\pi f_i$
F_1	$-0.10245 \times 10^{-9} W_i^6 + 0.17583 \times 10^{-5} W_i^4 - 0.44601 \times 10^{-2} W_i^2 + 1$
F_2	$0.12882 \times 10^{-7} W_i^4 - 0.93394 \times 10^{-4} W_i^2 + 0.10543$
F_3	$-0.45416 \times 10^{-9} W_i^6 + 0.37667 \times 10^{-5} W_i^4 - 0.56104 \times 10^{-2} W_i^2 + 1$
F_4	$-0.21179 \times 10^{-11} W_i^6 + 0.51728 \times 10^{-7} W_i^4 - 0.17947 \times 10^{-3} W_i^2 + 0.10543$

Note: f_i = Center frequency (Hz) of i^{th} spectral band

APPENDIX A.2 – NARRATIVE ON HUMAN TOLERANCE CRITERIA

SHOCK AND VIBRATION HUMAN TOLERANCE CRITERIA

Evaluating the mobility of combat systems not only involves vehicle aspects, but personnel as well. The soldier is a key component for the operation, execution, and maintainability of the weapon system. Issues relating to his or her sustained performance capability have a direct bearing on operational effectiveness.

Tolerance criteria for shock and vibration exposure for the individual human operator is critical in determining the overall performance of the combat vehicle system.

The human body is a complex system with varying degrees of physical and behavioral characteristics. The multitude of factors involved limit the reliability of specific test data to general application of tolerance limits for individual subjects.

To illustrate the challenge of such a determination the application of shock and vibration to the body as outlined by Harris (1996) can be applied through a gas, liquid, or solid. The application can be distributed or concentrated and applied in a range of directions to the body from normal to tangential, and in more than one direction. The position of the body as well changes the response of the internal tissues.

Further complications arise from the application of linear modeling techniques. While measurement of shock, the transfer of considerable momentum over a short time period, using the principle of superposition from component response to single frequency sinusoidal forces does not apply for the nonlinear properties encountered.

The effects of shock and vibration can be placed into three broad categories: 1) interference with physical action, 2) mechanical damage, and 3) subjective effects through biological receptor changes.

Physiological changes can occur through neurological and muscular responses. Stress resulting in increased heart rate and blood flow can be detected through blood and urine content.

From a mechanical aspect, the intense exposure to vibrational energy can lead to increased body temperature. Occupants in vehicles being propelled over rough terrain receive repeated random jolting motions. A case has been reported by Harris (1996) of gastrointestinal bleeding in one subject after 15 minutes of exposure to 6 g's.

Subjective limits or perceived discomfort to vibrational forces depends on no less than 7 factors: 1) magnitude, 2) frequency, 3) direction 4) duration, 5) body position, 6) body orientation, 7) point of application, 8) environment in which application occurs.

In an effort to provide a standard criteria for evaluation human toleration to vibrations, while keeping the influence of individual factors in mind, a criteria was developed from statistical analysis to determine the beginning of uncomfortable sensations and tolerable limits affecting task performance.

In the late 1960's Lee and Pradko (1968) developed the ride quality performance criterion known as absorbed power, the time average of frequency weighted acceleration, defined as

$$P_{absorbed} = \frac{1}{T} \int a_w(t)^2 dt$$

The average value for sustained vehicle operation is 6 watts while 15-20 watt levels are sustainable for short periods of time.

Later this work was used in-part in the development of International Standards Organization (ISO) 2631 - Evaluation of human exposure to whole-body vibration for periodic, random and transient vibration.

Both filters show high sensitivity to the natural frequency of the thorax-abdomen in the 3-4 Hertz range. Smith et. al. (1978) provides a comparison between the two with similar results.

References

Harris, C.M., 1996, Shock and Vibration Handbook, 4th Edition, Ch. 22, 44.

ISO 2631/1-1985(E), First edition-1985-05-15 - Evaluation of human exposure to whole-body vibration. International Standards Organization.

Lee, R.A., Pradko, F., 1968, "Analytical Analysis of Human Vibration", SAE Paper No. 680091.

Smith, C.C, McGehee, D.Y., Healey, A.J., 1978, "The Prediction of Passenger Riding Comfort From Acceleration Data", ASME Journal of Dynamic Systems, Measurement, and Control, Vol. 100 (March), pp. 34-41.

APPENDIX A.3 – ABSPWR_dF

```

figure(1)
clf

% Sinusoidal test
l = 30000;
t = [0:l]*.001;
u1 = (8/32.2)*sin(2*3.14159*4*t); %~0.25 4Hz g's

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%CHANGE
% input units = g's
y = [t;u1]';
%y = dlmread('..\modellib\r2\out_deskr_rms070.txt',' ',1,0);
%y = dlmread('..\modellib\r2\out_deskr_rms150.txt',' ',1,0);
%y = dlmread('..\modellib\r2\BB_20mph_full_ext_ASCII.txt');
%y = dlmread('..\modellib\r2\BB_20mph_empty_ext_ASCII.txt');

% Plot Output
y(:,1) = y(:,1) + 0.001;
i = find(y(:,1) > 1.0);
ut = [y(i,1), [y(i,2) y(i,1)]];

subplot(1,2,1), plot(y(:,1), y(:,2),'b');
hold on
subplot(1,2,1), plot(y(i,1), y(i,2),'r');

% Absorbed Power - Time Based
% [T,X,Y] = sim('model',Timespan, Options, UT) (before 2009a)
% options = simset('DstWorkspace','current','SrcWorkspace','current');
[to,xo,yo] = sim('AP',[y(i(1),1),y(i(end),1)],[],ut); %AP, multiplies by ft/s^2 so input in g's

subplot(1,2,2), plot(to,yo);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%CHANGE
%print('-dpng','ff1');
%print('-dpng','ff2');
%print('-dpng','ff3');
%print('-dpng','ff4');
%print('-dpng','ff5');

% Define Frequency Based Filter
w = [.1:.1:1000]'; % rad/sec
w1 = w;
j = i; % save for later

for i = 1:length(w1)
    k1 = 1.356;
    k0 = 4.3537;
    f1 = -0.10245e-9*w1(i)^6+0.17583e-5*w1(i)^4-0.44601e-2*w1(i)^2+1;
    f2 = 0.12882e-7*w1(i)^4-0.93394e-4*w1(i)^2+0.10543;
    f3 = -0.45416e-9*w1(i)^6+0.37667e-5*w1(i)^4-0.56104e-2*w1(i)^2+1;
    f4 = -0.21179e-11*w1(i)^6+0.51728e-7*w1(i)^4-0.17947e-3*w1(i)^2+0.10543;

    % magnitude
    myabs1(i) = sqrt(k0*k1*(f1*f4-f2*f3)/(f3*f3+w1(i)^2*f4*f4));
    myabs1(i) = myabs1(i)^2; % if use with PSD then no sqrt -> sqrt(k)^2 = k
    % mdb1(i)=20*log10(myabs1(i));
end

i=j; % restore

Fs= 1/(y(i(2),1)-y(i(1),1)) % sampling freq

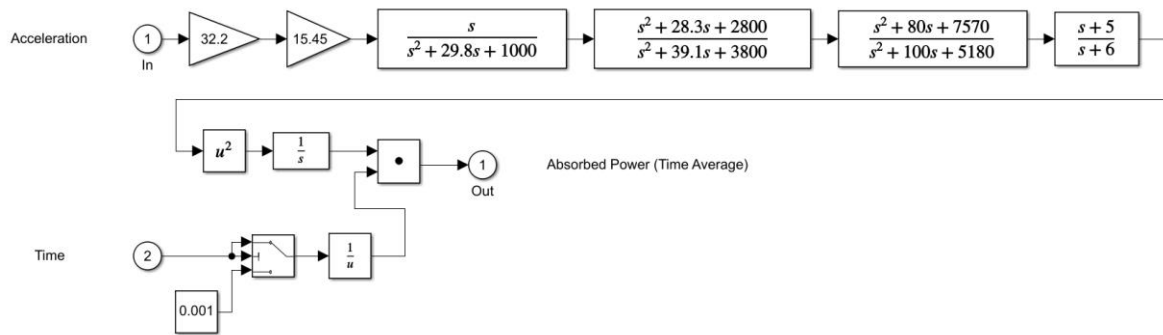
% Alternative to pwelch
%hm = 2^nextpow2( fix( length(y(i,2))/8 ) );
%hs = spectrum.welch('Hamming',max(256, hm+1));
%h1 = psd(hs,32.2*y(i,2),'Fs',Fs); % input units in ft/s^2
%dF = h1.frequencies(2)-h1.frequencies(1)
%fprintf('%f %e\n',[h1.frequencies;h1.data]);
%c=interp1(w/(2*pi),myabs1,h1.frequencies,'linear',0).*h1.data;

[Pxx,F]=pwelch(32.2*y(i,2),[],[],[],Fs);
dF=F(2)-F(1)
c=interp1(w/(2*pi),myabs1,F,'linear',0).*Pxx;
trapz(c)*dF % freq based
yo(end) % time based

```

APPENDIX A.4 – AP.mdl

(after 2009a is AP.slx)



```

<Block BlockType="Inport" Name="In" SID="1">
<P Name="Position">[30, 30, 50, 50]</P>
<P Name="ZOrder">-1</P>
</Block>
<Block BlockType="Inport" Name="In1" SID="2">
<P Name="Position">[30, 185, 50, 205]</P>
<P Name="ZOrder">-2</P>
<P Name="Port">2</P>
</Block>
<Block BlockType="Constant" Name="Constant" SID="3">
<P Name="Position">[60, 220, 90, 250]</P>
<P Name="ZOrder">-3</P>
<P Name="Value">0.001</P>
</Block>
<Block BlockType="DotProduct" Name="Dot Product" SID="4">
<P Name="Position">[205, 112, 235, 143]</P>
<P Name="ZOrder">-4</P>
<P Name="InputSameDT">off</P>
<P Name="OutDataTypeStr">Inherit: Inherit via internal rule</P>
</Block>
<Block BlockType="Gain" Name="Gain" SID="5">
<P Name="Position">[70, 21, 120, 59]</P>
<P Name="ZOrder">-5</P>
<P Name="Gain">32.2</P>
<P Name="SaturateOnIntegerOverflow">on</P>
</Block>
<Block BlockType="Gain" Name="Gain1" SID="6">
<P Name="Position">[140, 20, 185, 60]</P>
<P Name="ZOrder">-6</P>
<P Name="Gain">15.45</P>
<P Name="SaturateOnIntegerOverflow">on</P>
</Block>
<Block BlockType="Integrator" Name="Integrator" SID="7">
<P Name="Ports">[1, 1]</P>
<P Name="Position">[130, 107, 170, 133]</P>
<P Name="ZOrder">-7</P>
</Block>
<Block BlockType="Math" Name="Math Function" SID="8">
<P Name="Ports">[1, 1]</P>
<P Name="Position">[80, 105, 110, 135]</P>
<P Name="ZOrder">-8</P>
<P Name="Operator">square</P>
</Block>
<Block BlockType="Math" Name="Math Function1" SID="9">
<P Name="Ports">[1, 1]</P>
<P Name="Position">[170, 179, 200, 211]</P>
<P Name="ZOrder">-9</P>
<P Name="Operator">reciprocal</P>
</Block>
<Block BlockType="Switch" Name="Switch" SID="10">
<P Name="Position">[115, 180, 145, 210]</P>
<P Name="ZOrder">-10</P>
<P Name="Threshold">.01</P>
</Block>
<Block BlockType="TransferFcn" Name="Transfer Fcn" SID="11">
<P Name="Position">[205, 19, 340, 61]</P>
<P Name="ZOrder">-11</P>

```

```
<P Name="Numerator">[1 0]</P>
<P Name="Denominator">[1 29.8 1000]</P>
</Block>
<Block BlockType="TransferFcn" Name="Transfer Fcn1" SID="12">
<P Name="Position">[360, 19, 515, 61]</P>
<P Name="ZOrder">-12</P>
<P Name="Numerator">[1 28.3 2800]</P>
<P Name="Denominator">[1 39.1 3800]</P>
</Block>
<Block BlockType="TransferFcn" Name="Transfer Fcn2" SID="13">
<P Name="Position">[535, 20, 670, 60]</P>
<P Name="ZOrder">-13</P>
<P Name="Numerator">[1 80 7570]</P>
<P Name="Denominator">[1 100 5180]</P>
</Block>
<Block BlockType="TransferFcn" Name="Transfer Fcn3" SID="14">
<P Name="Position">[690, 21, 750, 59]</P>
<P Name="ZOrder">-14</P>
<P Name="Numerator">[1 5]</P>
<P Name="Denominator">[1 6]</P>
</Block>
<Block BlockType="Outport" Name="Out" SID="15">
<P Name="Position">[270, 120, 290, 140]</P>
<P Name="ZOrder">-15</P>
<P Name="InitialOutput">0</P>
</Block>
```


TN 15-1021-01

GROUND MOBILITY VEHICLE (GMV) WAVENUMBER TERRAINS

October 21, 2015

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
 U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC)
 Detroit Arsenal
 Systems Integration & Engineering
 ATTN: RDTA-SIE/MS157
 6501 East 11 Mile Road
 Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

The question was asked if test data existed that shows how the M998 and M1097 HMMWV's perform over the terrains described in Table 1.

Table 1 – Test Signal (two cycles)

Terrain			Average Speed	Wave Number Spectrum	RMS (root mean square) Roughness
Improved Surfaces	Primary Surfaces	High Quality Paved Road (5%)	55 mph	$G_{xx}(n)=1.4 \times 10^{-8}(n)^{-2.5}$	0.1
		Secondary Pavement (5%)	50 mph	$G_{xx}(n)=1.9 \times 10^{-7}(n)^{-2.5}$	0.2
	Secondary Surfaces	Rough Pavement Degraded (10%)	40 mph	$G_{xx}(n)=8.0 \times 10^{-7}(n)^{-2.5}$	0.3 – 0.5
		Loose Surface (10%)	35 mph	$G_{xx}(n)=3.0 \times 10^{-5}(n)^{-2.0}$	0.6
Un-Improved Surfaces	Trails (35%)		22 mph	$G_{xx}(n)=4.6 \times 10^{-4}(n)^{-1.9}$	1.0 -3.4
	Cross-Country (35%)		17 mph	$G_{xx}(n)=9.2 \times 10^{-4}(n)^{-2.1}$	1.5 – 4.8

PROBLEM SOLUTION:

From review of [1] the following was obtained for Table 1.

The GMV will be a lightweight, high mobility platform that will be transportable by all means including vertical lift (via CH-47 and UH-60) and airdrop.

.

.

.

Mobility: The GMV shall be a four wheel drive vehicle capable of traversing longitudinal grades up to 60%. The GMV shall be capable of negotiating obstacles (e.g. ditches, downed trees, rocks, vertical steps and curbs). The GMV shall be capable of negotiating soft soils (RCI = 22)

at GVW. The GMV shall be capable of meeting the mobility requirements from VCW to GVW on the following terrain profiles: [see Table1].

Referencing the page shown in Appendix A.1 from [2] the following answer was provided.

The single slope wave number spectra is an idealization of the actual course spectra. TOP 01-1-010 does not identify any of the test courses with an idealized wave number spectra while it does provide the individual course spectra. The courses could be matched to a test course based on the RMS value (see Appendix A.1) but that does not guarantee they would have the same wave number spectra---hence the desire to identify courses in this way. It is possible that APG still has actual test data from the M998/M1097 HMMWV era, but highly unlikely for all the courses listed.

REFERENCE

- [1] GMV2_Market_Questionnaire-GeneralInfo.docx (557.59 Kb). Ground Mobility Vehicles (GMV)-Market Questionnaire/Request for Information RFI (GMV2). Solicitation Number: W56HZV15RFIGMV2. Agency: Department of the Army. Office: Army Contracting Command. Location: ACC - Warren (ACC-WRN)(DTA). (<https://sam.gov/opp/a402d68610234cfc358b09dbae971143/view>).
- [2] Test Operating Procedure (TOP) 1-1-010A Vehicle Test Course Severity (Surface Roughness) (31 Oct 2006). DIST A. Approved for public release: distribution unlimited.

APPENDIX A.1 – TOP 1-1-010 – Table 6.

TOP 1-1-010
31 October 2006TABLE 6. TEST COURSES BY TERRAIN TYPE
(BASED ON RMS ROUGHNESS ONLY WHICH DOES
NOT IMPLY EQUIVALENT TEST RESULTS)¹

Terrain Type	rms roughness, in	ATC Courses	Nominal roughness, in	YTC Courses	Nominal roughness, in
Primary	<0.2	Perryman Paved	0.07	Laguna Paved	0.08
Secondary	$\geq 0.2 - \leq 0.6$	Munson Gravel	0.25	KOFA Level Gravel	0.23
		Perryman A	0.33	Laguna Level	0.23
		Perryman 1	0.34	Patton Level	0.36
		Churchville C	0.21	Gravel	
				Patton Hilly Gravel	0.36
				Laguna Level	0.35
				Trails – East	
				Laguna Level	0.39
				Trails – West	
Trails	$> 0.6 - \leq 1.5$			Laguna Hilly Trails	0.40
		Belgian Block	0.82	Patton Hilly Trails	1.15
		Perryman 2	0.90	Trails	
		Churchville B	1.09	Patton Level Trails	0.85
				Mid-East – start	1.02
				Mid-East Section A	1.44
				Mid-East Wash/End	0.70
				Desert March	0.92
Cross Country	>1.5	Perryman 3 Perryman 4	2.86 NA	Mid-East Section B	1.77

¹ Test results are a function of many test course attributes such as longitudinal slope, side slope, mud/dust, soil strength, radius of curvature and surface roughness. The response of a vehicle to surface roughness is dependent upon the shape of the wave number spectrum and the vehicle speed. Therefore, having equivalent rms roughness values does not imply that two test courses will produce equivalent test results.

TN 15-1027-01

PROFILE IV COURSE GENERATION

October 27, 2015

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC)
Detroit Arsenal
Systems Integration & Engineering
ATTN: RDTA-SIE/MS157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

How is Aberdeen Proving Grounds (APG) Profile IV course generated for use in vehicle dynamics simulations.

PROBLEM SOLUTION:

Reference [1] defines the course profile (see Appendix A.1). Two geometries define the elements of the course---an ellipse and a 2-sided half sinusoid.

Ellipse Definition

An ellipse is defined in [2] as

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \quad (1)$$

or

$$\begin{aligned} y &= \frac{b}{a} \sqrt{a^2 - x^2} \\ x &= [-a: a] \end{aligned} \quad (2)$$

where 2b is the length of the minor axis and 2a is the length of the major axis. The associated profile distance associated with the ellipse elevation is shifted from (2) and is defined to be

$$x = [-a: a] + a = [0: 2a] \quad (3)$$

2-Sided Half Sinusoid

The front of the half sinusoid is defined as

$$\begin{aligned} y &= H \sin(2\pi f x) \\ x &= [0: F] \end{aligned} \quad (4)$$

where the frequency is defined as

$$f = \frac{1}{4F} \quad (5)$$

and F is the distance of the front half of the half sinusoid. The rear of the half sinusoid is defined as

$$\begin{aligned} H \cos(2\pi f x) \\ x = [0:R] \end{aligned} \quad (6)$$

where the frequency is defined as

$$f = \frac{1}{4R} \quad (7)$$

and R is the distance of the rear half of the half sinusoid. The associated profile distance associated with the rear half sinusoid is shifted from (6) and is defined to be

$$x = F + [0:R] \quad (8)$$

The outline below describes the MATLAB [3] script used to create the course profile (see Appendix A.2).

- initial data in feet
- define sections (S) (front/rear/height)
- define course (L) from section sequence (negative lengths indicate flat sections)
- create table (T) with starting position (cumulative length) for each section and length
- for each element in L
 - define beginning distance = T(i)
 - define ending distance = T(i) + section length
 - if negative is flat section
 - else if section F treat special
 - define course from section sequence (L1)
 - set ellipse parameters (a,b)
 - for each element in L1
 - if negative is flat section
 - else
 - plot ellipse
 - end
 - end
 - else if section H treat special
 - define course from section sequence (L2)
 - repeat as for section F
 - else 2-sided half sinusoid
 - set F, R, H from S, e.g. $F = S(L(i),2)$
 - for the Front
 - define frequency from F
 - plot using H
 - for the Rear
 - define frequency from R
 - plot using H
 - end
- end
- plot course from accumulated elevation versus distance data
- convert data to inches

- output course elevation versus distance data to a file for later use (apg4_in.txt)

An example of table (T) is shown below:

TLen	Sec	Len
0	A	25
25	Flat	5
30	B	30
60	Flat	10
70	B	30
100	Flat	5
105	C	16
121	D	19
140	D	19
159	E	28
187	Flat	2
189	F	33
222	Flat	1
223	G	37
260	Flat	8
268	H	38
306	Flat	8
314	I	35
349	Flat	2
351	J	17
368	Flat	6
374	K	22
396	Flat	5
401	A	25
426	Flat	10
436	A	25
461		

Figure 1 shows the individual sections in different colors that make of the complete course. Figure 2 shows the complete elevation versus distance profile. This should be the same as in Figure 1 but has removed any overlapping points so that the distance of the profile is monotonically increasing. Figure 3 shows a comparison between the course in feet and scaled to be in inches. Figures 4 to 6 show different views of the course in three dimensions.

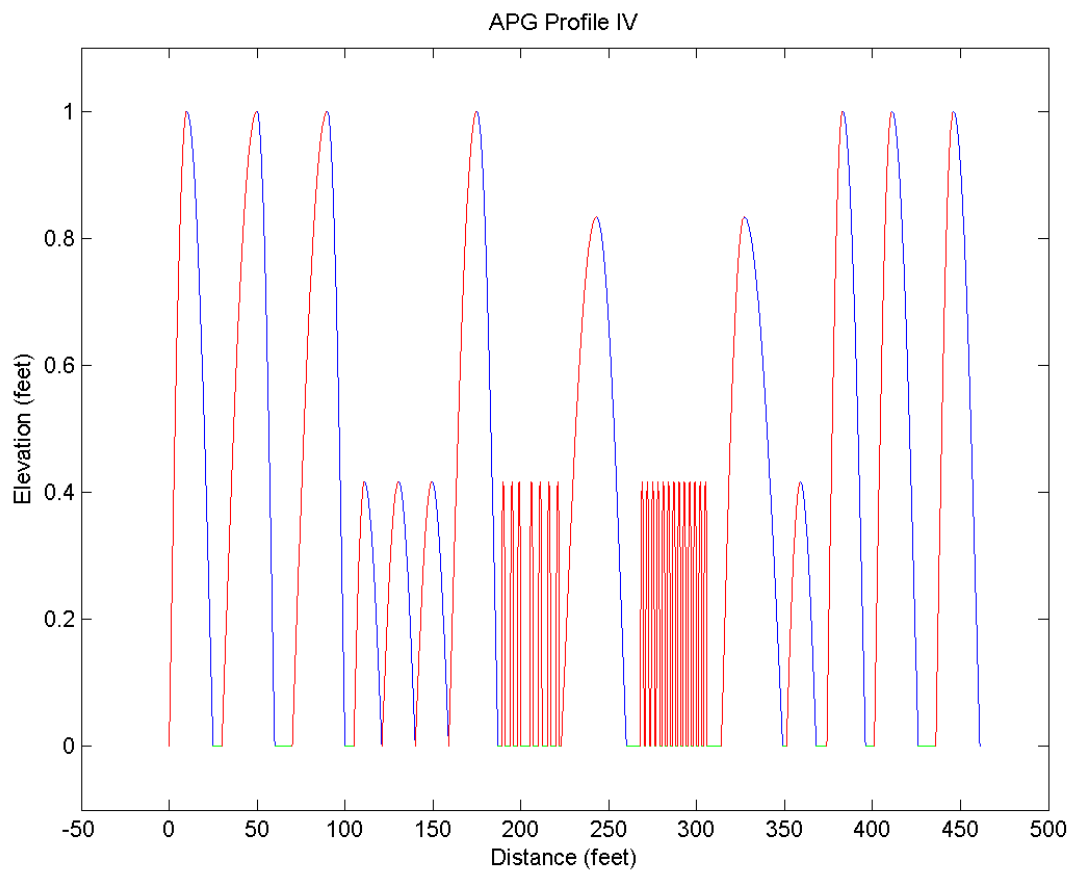


Figure 1 – Course Sections

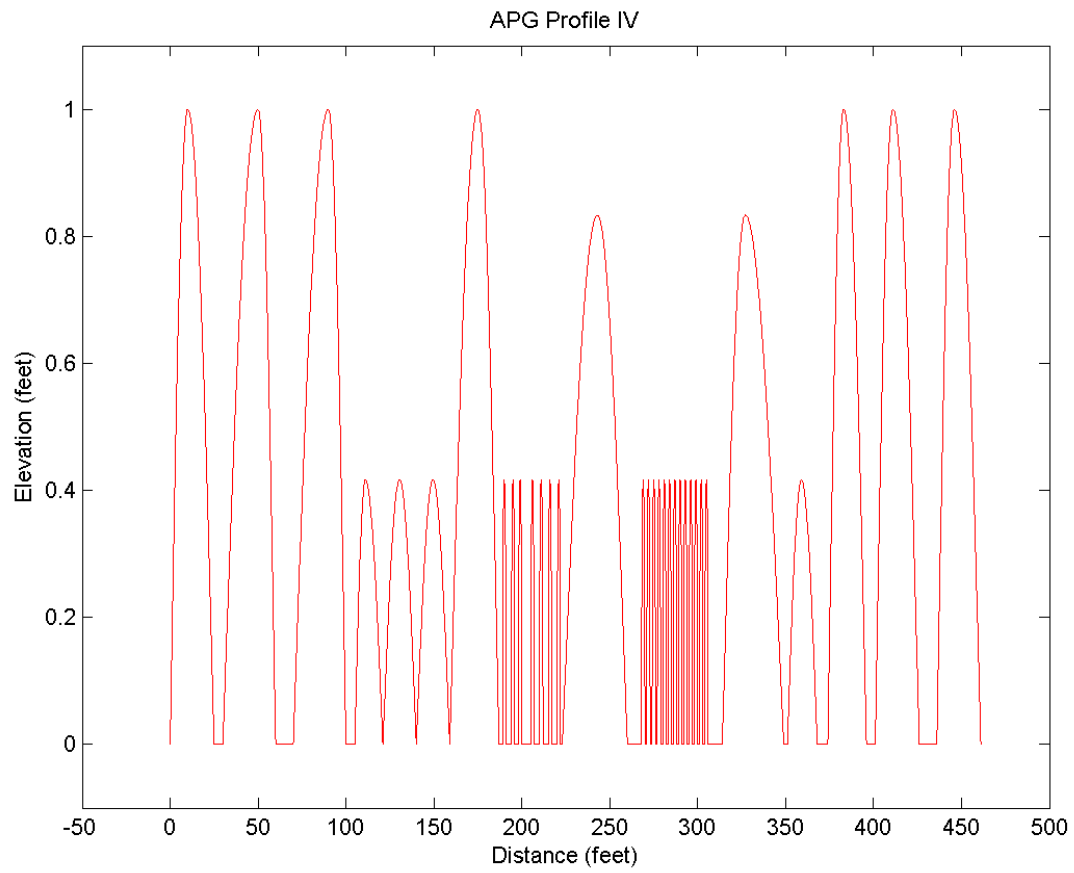


Figure 2 – Complete Course (Elevation vs. Distance)

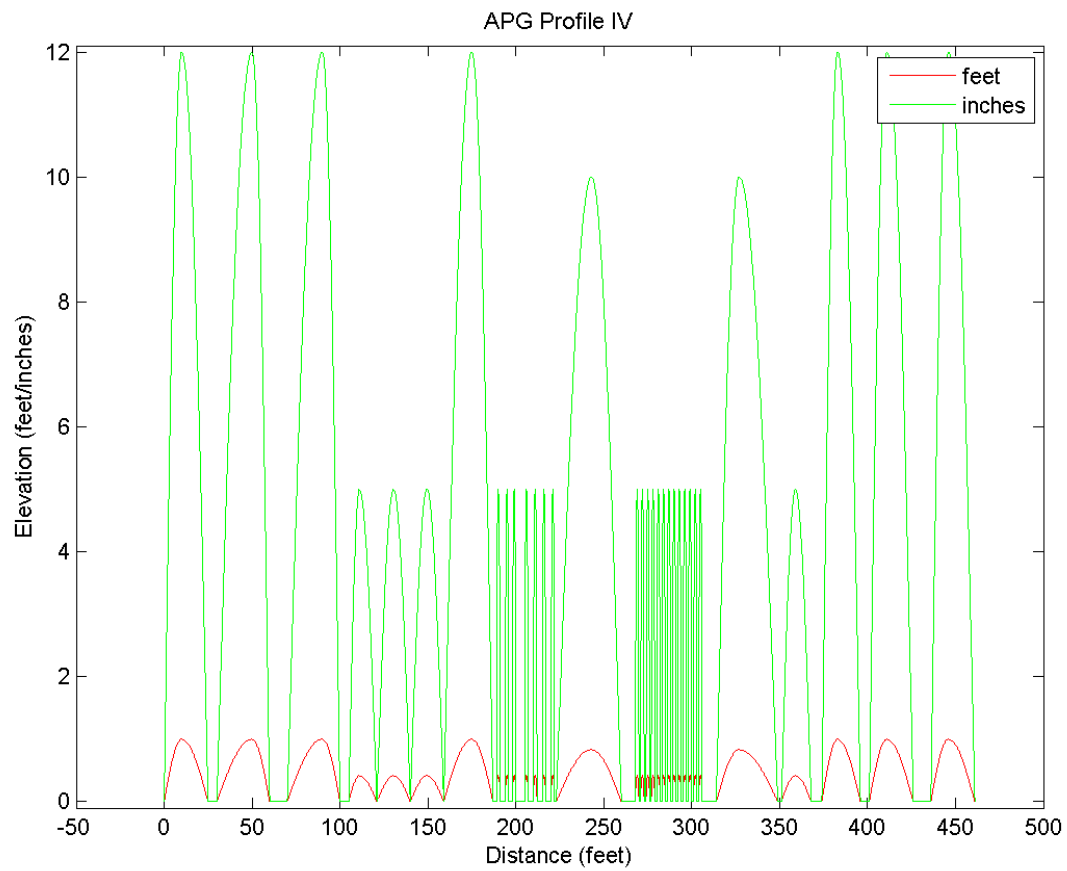


Figure 3 – Course Comparison (feet vs. inches)

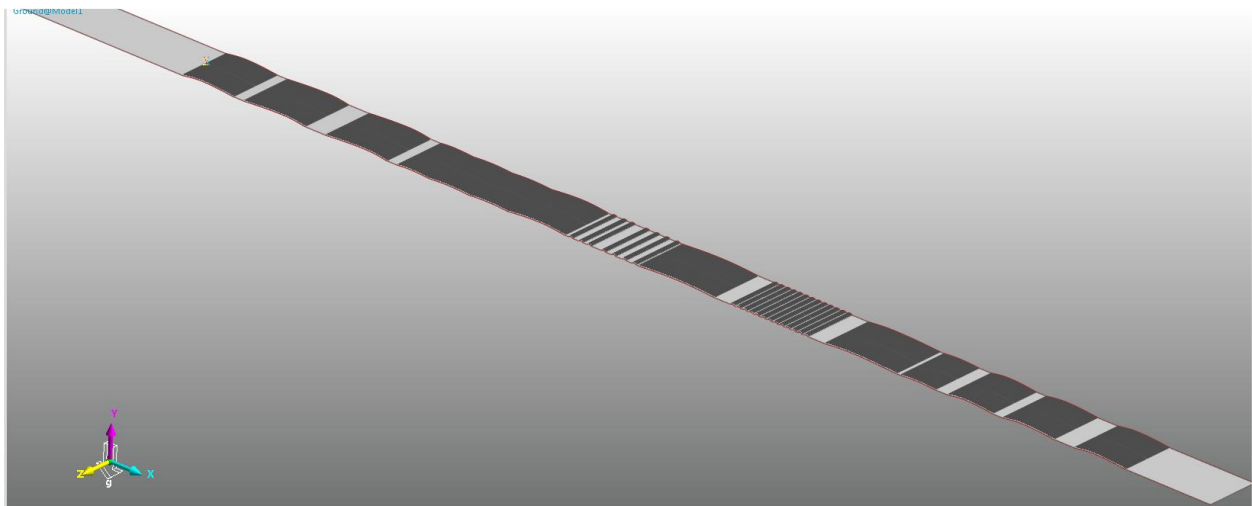


Figure 4 – Course View – Perspective (3D)

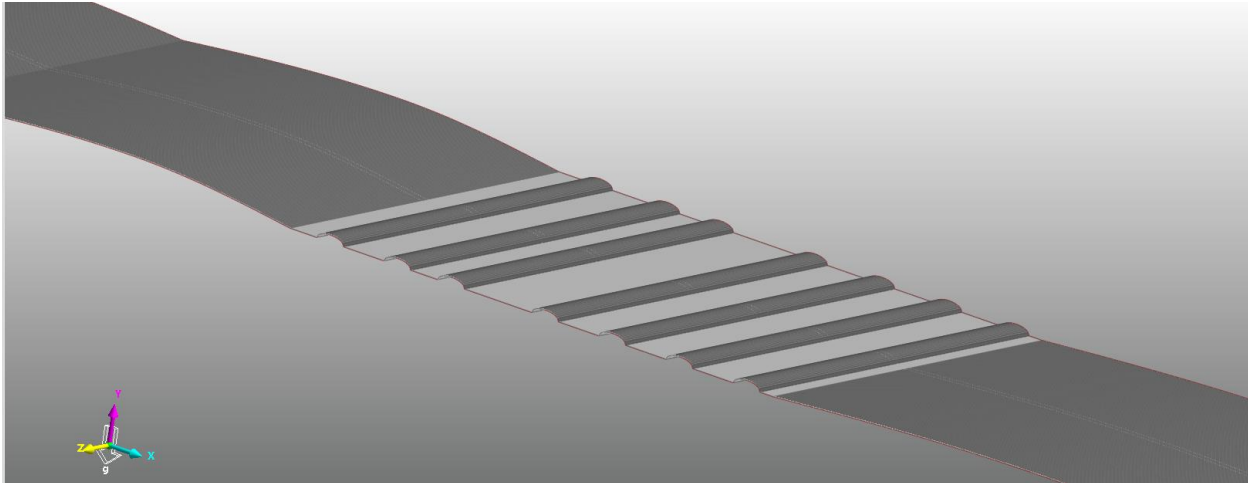


Figure 5 – Course View – Section F

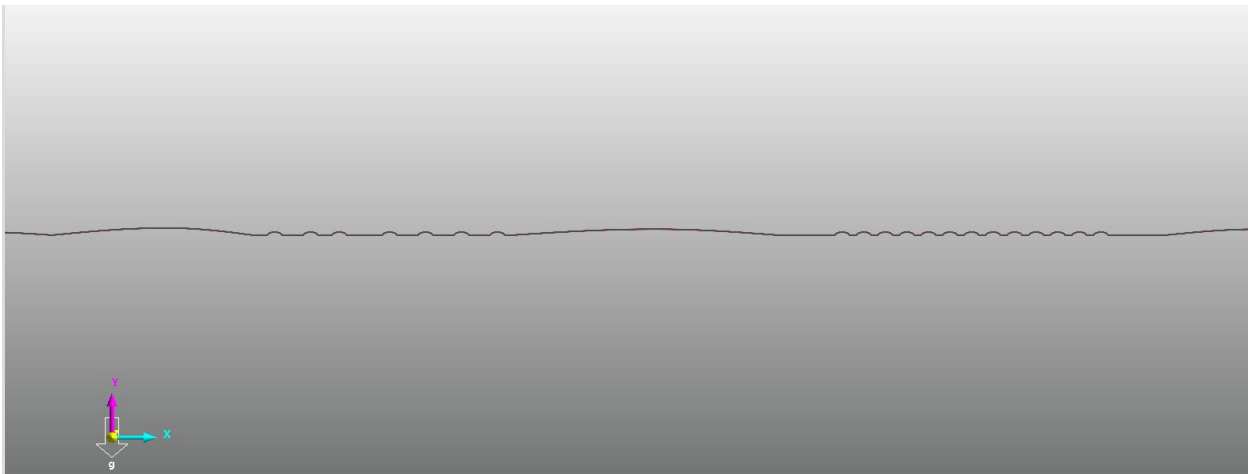


Figure 6 – Course view - Side

The techniques applied here can be used to create other non-standard course profiles in three dimensions.

REFERENCE

- [1] Test Operations Procedure (TOP) 01-1-011A Vehicle Test Facilities at Aberdeen Test Center and Yuma Test Center (27 Feb 2012). DIST A (DTIC AD A557 002). Approved for public release: distribution unlimited.
- [2] <http://mathworld.wolfram.com/Ellipse.html>
- [3] MATLAB (www.mathworks.com)

APPENDIX A.1 – Course Definition

TOP 01-1-011A
27 February 2012

2.6.26 Profile IV Course.

The Profile IV Course consists of a series of eleven different types of bumps (some of which are repeated) distributed over a 142 m (465 ft) long concrete course. The height (displacement) of each of the bumps was computed for each inch of longitudinal travel. The course layout is shown in Figure 64. An M1 tank traversing the MTA Profile IV Course is shown in Figure 65.

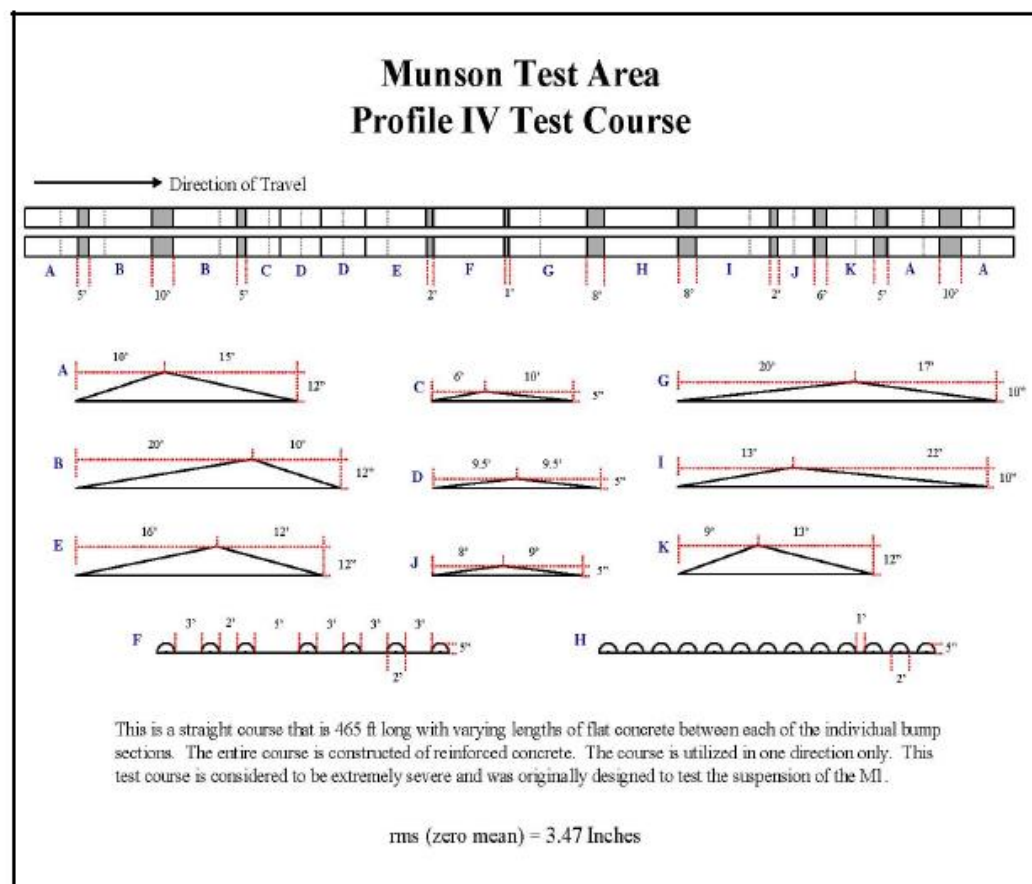


Figure 64. MTA Profile IV Test Course layout.

TOP 01-1-011A
27 February 2012



Figure 65. Abrams tank traversing the MTA Profile IV Test Course.

APPENDIX A.2 – APG IV Profile Creation Script

```

% APG IV Profile Definition (TOP 01-1-011A VEHICLE TEST FACILITIES 27 Feb 12)
%
% Section Names
d={'A','B','C','D','E','F','G','H','I','J','K'};

% Section Definitions
% Length, F(Front), R(Rear), H(Height) - in feet
s=[25,10,15,1;
30,20,10,1;
16,6,10,5/12;
19,9.5,9.5,5/12;
28,16,12,1;
33,0,0,-1;
37,20,17,10/12;
38,0,0,-1;
35,13,22,10/12;
17,8,9,5/12;
22,9,13,1];

% Profile IV Definition
% Index into s, < 0 is length of flat section
l=[1,-5,2,-10,2,-5,3,4,4,5,-2,6,-1,7,-8,8,-8,9,-2,10,-6,11,-5,1,-10,1];

sl = 0; % section position
tbl=[]; % cumulative length

% course description by section
diary out.txt
fprintf('TLen  Sec  Len\n');
for i=1:length(l),
    sn = l(i); % section number

    if (sn < 0)
        nm = 'Flat'; % name
        ln = abs(sn); % length
    else
        nm = d(sn);
        ln = s(sn);
    end

    fprintf('%5.0f %5s %5.0f\n',sl,char(nm),ln);
    tbl=[tbl;[sl]];
    sl = sl + ln;
end
fprintf('%5.0f\n',sl);
diary off

apg4 = [-1,-1]; % Profile IV Distance/Elevation
samp = 0.1; % sample rate

figure(1)
clf

% Generate Course
for i = 1:length(l),
    beg = tbl(i); % beginning distance of section
    %fprintf('%5f\n',beg);
    endl = beg+abs(l(i)); % ending distance of section

    if (l(i) < 0),
        % flat
        h=plot([beg,endl],[0,0],'g');
        %set(h,'LineWidth',3);

    elseif (l(i) == 6)
        % F
        % Section F Definition
        % Index into s, < 0 is length of flat section
        t1=[1,-3,1,-2,1,-5,1,-3,1,-3,1,-3,1];

```

```

s11 = 0; % section position

% Ellipse Definition
be = 5/12; % minor axis = 2b/2
ae = 1; % major axis = 2a/2

for j = 1:length(t1),
    if (t1(j) < 0)
        % flat
        h=plot([beg+s11,beg+s11+abs(t1(j))],[0,0],'g');
        s11 = s11 + abs(t1(j));

    else
        % ellipse
        x = [-ae:samp:ae]; % section distance
        xx = beg+s11+(x+ae);
        if (apg4(end,1) == xx(1)),
            apg4(end,:) = [];
        end
        h=plot(beg+s11+(x+ae), (be/ae)*sqrt(ae^2-x.^2), 'r');
        apg4 = [apg4; [beg+s11+(x+ae)], [(be/ae)*sqrt(ae^2-x.^2)]];
        s11 = s11 + 2;
    end
end

elseif (l(i) == 8)
    % H
    % Section H Definition
    % Index into s, < 0 is length of flat section
    t1=[1,-1, 1,-1, 1,-1, 1,-1, 1,-1, 1,-1, 1,-1, 1,-1, 1,-1, 1];
    s11 = 0; % section position
    be = 5/12; % minor axis = 2b/2
    ae = 1; % major axis = 2a/2

    for j = 1:length(t1),
        if (t1(j) < 0)
            % flat
            h=plot([beg+s11,beg+s11+abs(t1(j))],[0,0],'g');
            s11 = s11 + abs(t1(j));

        else
            % ellipse
            x = [-ae:samp:ae]; % section distance
            xx = beg+s11+(x+ae);
            if (apg4(end,1) == xx(1)),
                apg4(end,:) = [];
            end
            h=plot(beg+s11+(x+ae), (be/ae)*sqrt(ae^2-x.^2), 'r');
            apg4 = [apg4; [beg+s11+(x+ae)], [(be/ae)*sqrt(ae^2-x.^2)]];
            s11 = s11 + 2;
        end
    end

else
    F = s(l(i),2);
    R = s(l(i),3);
    H = s(l(i),4);
    % Front
    T = 4*F; % period for F
    f = 1/T;
    x = [0:samp:F]; % section distance
    xx = beg+x;
    if (apg4(end,1) == xx(1)),
        apg4(end,:) = [];
    end
    h=plot(beg+x, H*sin(2*pi*f*x), 'r');
    apg4 = [apg4; [beg+x], [H*sin(2*pi*f*x)]];
    hold on
    % Rear
    T = 4*R; % period for R
    f = 1/T;

```

```

        x = [0:samp:R]; % section distance
        xx = beg+F+x;
        if (apg4(end,1) == xx(1)),
            apg4(end,:) = [];
        end
        h=plot(beg+F+x,H*cos(2*pi*f*x),'b');
        apg4 = [apg4; [[beg+F+x]', [H*cos(2*pi*f*x)]]'];
    end
end
axis([-50,500,-.1,1.1])
title('APG Profile IV')
ylabel('Elevation (feet)')
xlabel('Distance (feet)')
print('-dpng','f1');

% Plot Complete Profile IV Definition
figure(2)
clf
apg4(1,:)=[];
plot(apg4(:,1),apg4(:,2),'r');
title('APG Profile IV')
ylabel('Elevation (feet)')
xlabel('Distance (feet)')
axis([-50,500,-.1,1.1])
print('-dpng','f2');

% Plot Complete Profile IV Definition in feet and inches
figure(3)
clf
plot(apg4(:,1),apg4(:,2),'r');
hold on
%apg4(:,1) = apg4(:,1)*12 % convert feet to inches
%apg4(:,2) = apg4(:,2)*12 % convert feet to inches
apg4=apg4*12; % convert feet to inches
plot(apg4(:,1)/12,apg4(:,2),'g');
title('APG Profile IV')
legend('feet','inches')
ylabel('Elevation (feet/inches)')
xlabel('Distance (feet)')
axis([-50,500,-.1,12.1])
print('-dpng','f3');

%fid = fopen('apg4.txt','wt');
apg4=[apg4;[apg4(end,1)+500,apg4(end,2)]];
fid = fopen('apg4_in.txt','wt');
fprintf(fid,'%f %f\n',apg4');
fclose(fid);

```

TN 16-0315-01

RECURDYN HEIGHT RELATIVE TO GROUND DETERMINATION

March 15, 2016

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
 U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC)
 Detroit Arsenal
 Systems Integration & Engineering
 ATTN: RDTA-SIE/MS157
 6501 East 11 Mile Road
 Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

In RecurDyn [1], positions of body markers are reported with respect to the global coordinate system (Ground.InertiaMarker). How do you get the marker height relative to the ground (road) surface?

PROBLEM SOLUTION:

The solution requires the determination of the road height at the marker position which can then be subtracted from the marker's global height to get relative height as in (1).

$$h_{relative} = h_{global} - road\ height_{global} \quad (1)$$

This requires a 3D spline of the road surface and the use of an interpolation function, which when combined into an expression will provide the relative height. The following sections describe the process.

Road Surface

A 3D spline subentity can be imported into RecurDyn. Figure 1 shows the format.

	Y1	Y2	Y3	.	.	.	Ym
X1	Z11	Z21	Z31	,	,	,	Zm1
X2	Z12	.					
X3	Z13		.				
.	.			.			
.	.				.		
.	.					.	
Xn	Z1n						Zmn

Figure 1 – 3D Spline Format

Appendix A.1 includes the Spline reference from [2]. Note that the Z values in Figure 1 are called “Y” or “Y1, Y2, etc.” in [2] and are specified for each X value for a given “Z” value (in [2] Y and Z referenced in Figure 1 are switched). This data can be obtained from the road data used to create an “.rdf” file [5,6]. The format of the z data file contains rows as defined in Figure 2.

Z11	Z12	Z13	.	.	.	Z1n
Z21	Z22	Z23	.	.	.	Z2n
Z31	Z32	Z33	.	.	.	Z3n
.						
.						
.						
Zm1	Zm2	Zm3	.	.	.	Zmn

Figure 2 – Z Data Format

The x and y data file format is defined as a single column vector (multiple lines). Assuming the x, y, and z data for the road surface is stored in files as described above, then the following MATLAB [4] script “rdspsurf.m” will create a “.3dm” 3D spline file for importing.

rdspsurf.m

```
% rdspsurf.m - rd spline surface

dir1 = '\data\'; % data directory
prel = 'road'; % prefix

for g=1:25, % road number

    % input name prefix for x,y,z data - road_surf_03_x.txt, etc.
    nm1 = sprintf('_surf_%02d_',g);
    % output name prefix for spline - road_spline_03.3dm
    nm1_o = sprintf('_spline_%02d',g);

    nm1 = [prel nm1 'x.txt'];
    nm2 = [prel nm1 'y.txt'];
    nm3 = [prel nm1 'z.txt'];
    nm1_o = [prel nm1_o '.3dm']; % for RecurDyn 3D spline

    rx = dlmread([dir1 nm1]);
    ry = dlmread([dir1 nm2]);
    rz = dlmread([dir1 nm3]);

    fid = fopen(nm1_o,'wt');
    fprintf(fid,' %20s',' ');
    fprintf(fid,' %20.10f',ry);
    fprintf(fid,'\n');
    for i=1:length(rx),
        fprintf(fid,' %20.10f',[rx(i),rz(:,i)])
        fprintf(fid,'\n');
    end
    fclose(fid)
end
```

Below is an actual file for a 5 foot gap.

===BOF: gap05ftsp.3dm

```
-1000.0000000000    -150.0000000000    -3.0000000000    3.0000000000    150.0000000000
 0.0000000000      0.0000000000      0.0000000000      0.0000000000      0.0000000000
 0.0000000000      0.0000000000      0.0000000000      0.0000000000      0.0000000000
 600.0000000000      0.0000000000      0.0000000000      0.0000000000      0.0000000000
 600.1000000000     -0.3911110000     -0.3911110000     -0.3911110000     -0.3911110000
 600.2000000000     -1.5288890000     -1.5288890000     -1.5288890000     -1.5288890000
 600.3000000000     -3.3600000000     -3.3600000000     -3.3600000000     -3.3600000000
 600.4000000000     -5.8311110000     -5.8311110000     -5.8311110000     -5.8311110000
 600.5000000000     -8.8888890000     -8.8888890000     -8.8888890000     -8.8888890000
 600.6000000000    -12.4800000000    -12.4800000000    -12.4800000000    -12.4800000000
 600.7000000000    -16.5511110000    -16.5511110000    -16.5511110000    -16.5511110000
 600.8000000000    -21.0488890000    -21.0488890000    -21.0488890000    -21.0488890000
 600.9000000000    -25.9200000000    -25.9200000000    -25.9200000000    -25.9200000000
 601.0000000000    -31.1111110000    -31.1111110000    -31.1111110000    -31.1111110000
 601.1000000000    -36.5688890000    -36.5688890000    -36.5688890000    -36.5688890000
 601.2000000000    -42.2400000000    -42.2400000000    -42.2400000000    -42.2400000000
 601.3000000000    -48.0711110000    -48.0711110000    -48.0711110000    -48.0711110000
 601.4000000000    -54.0088890000    -54.0088890000    -54.0088890000    -54.0088890000
```

601.5000000000	-60.0000000000	-60.0000000000	-60.0000000000	-60.0000000000
601.6000000000	-65.9911110000	-65.9911110000	-65.9911110000	-65.9911110000
601.7000000000	-71.9288890000	-71.9288890000	-71.9288890000	-71.9288890000
601.8000000000	-77.7600000000	-77.7600000000	-77.7600000000	-77.7600000000
601.9000000000	-83.4311110000	-83.4311110000	-83.4311110000	-83.4311110000
602.0000000000	-88.8888890000	-88.8888890000	-88.8888890000	-88.8888890000
602.1000000000	-94.0800000000	-94.0800000000	-94.0800000000	-94.0800000000
602.2000000000	-98.9511110000	-98.9511110000	-98.9511110000	-98.9511110000
602.3000000000	-103.4488890000	-103.4488890000	-103.4488890000	-103.4488890000
602.4000000000	-107.5200000000	-107.5200000000	-107.5200000000	-107.5200000000
602.5000000000	-111.1111110000	-111.1111110000	-111.1111110000	-111.1111110000
602.6000000000	-114.1688890000	-114.1688890000	-114.1688890000	-114.1688890000
602.7000000000	-116.6400000000	-116.6400000000	-116.6400000000	-116.6400000000
602.8000000000	-118.4711110000	-118.4711110000	-118.4711110000	-118.4711110000
602.9000000000	-119.6088890000	-119.6088890000	-119.6088890000	-119.6088890000
603.0000000000	-120.0000000000	-120.0000000000	-120.0000000000	-120.0000000000
657.0000000000	-120.0000000000	-120.0000000000	-120.0000000000	-120.0000000000
657.1000000000	-119.6088890000	-119.6088890000	-119.6088890000	-119.6088890000
657.2000000000	-118.4711110000	-118.4711110000	-118.4711110000	-118.4711110000
657.3000000000	-116.6400000000	-116.6400000000	-116.6400000000	-116.6400000000
657.4000000000	-114.1688890000	-114.1688890000	-114.1688890000	-114.1688890000
657.5000000000	-111.1111110000	-111.1111110000	-111.1111110000	-111.1111110000
657.6000000000	-107.5200000000	-107.5200000000	-107.5200000000	-107.5200000000
657.7000000000	-103.4488890000	-103.4488890000	-103.4488890000	-103.4488890000
657.8000000000	-98.9511110000	-98.9511110000	-98.9511110000	-98.9511110000
657.9000000000	-94.0800000000	-94.0800000000	-94.0800000000	-94.0800000000
658.0000000000	-88.8888890000	-88.8888890000	-88.8888890000	-88.8888890000
658.1000000000	-83.4311110000	-83.4311110000	-83.4311110000	-83.4311110000
658.2000000000	-77.7600000000	-77.7600000000	-77.7600000000	-77.7600000000
658.3000000000	-71.9288890000	-71.9288890000	-71.9288890000	-71.9288890000
658.4000000000	-65.9911110000	-65.9911110000	-65.9911110000	-65.9911110000
658.5000000000	-60.0000000000	-60.0000000000	-60.0000000000	-60.0000000000
658.6000000000	-54.0088890000	-54.0088890000	-54.0088890000	-54.0088890000
658.7000000000	-48.0711110000	-48.0711110000	-48.0711110000	-48.0711110000
658.8000000000	-42.2400000000	-42.2400000000	-42.2400000000	-42.2400000000
658.9000000000	-36.5688890000	-36.5688890000	-36.5688890000	-36.5688890000
659.0000000000	-31.1111110000	-31.1111110000	-31.1111110000	-31.1111110000
659.1000000000	-25.9200000000	-25.9200000000	-25.9200000000	-25.9200000000
659.2000000000	-21.0488890000	-21.0488890000	-21.0488890000	-21.0488890000
659.3000000000	-16.5511110000	-16.5511110000	-16.5511110000	-16.5511110000
659.4000000000	-12.4800000000	-12.4800000000	-12.4800000000	-12.4800000000
659.5000000000	-8.8888890000	-8.8888890000	-8.8888890000	-8.8888890000
659.6000000000	-5.8311110000	-5.8311110000	-5.8311110000	-5.8311110000
659.7000000000	-3.3600000000	-3.3600000000	-3.3600000000	-3.3600000000
659.8000000000	-1.5288890000	-1.5288890000	-1.5288890000	-1.5288890000
659.9000000000	-0.3911110000	-0.3911110000	-0.3911110000	-0.3911110000
660.0000000000	0.0000000000	0.0000000000	0.0000000000	0.0000000000
6000.0000000000	0.0000000000	0.0000000000	0.0000000000	0.0000000000

==EOF

Relative Height Expression

After importing the 3D spline discussed above in the “Road Surface” section, define an expression with the first argument as the spline curve and the second argument as the marker of interest. Figure 3 shows an example. The expression

$$DY(2) - LINSPL(DX(2), DZ(2), 1, 0) \quad (2)$$

assumes that the coordinate system is defined as

X = longitudinal (+forward)
Y = vertical (+up)
Z = lateral (+right).

It takes the marker's global height and subtracts the road height, which is found from using the interpolation function as described in Appendix A.2 from [3]. The marker's longitudinal and lateral position is used to find the road height from interpolation on the 3D spline that represents the road surface.

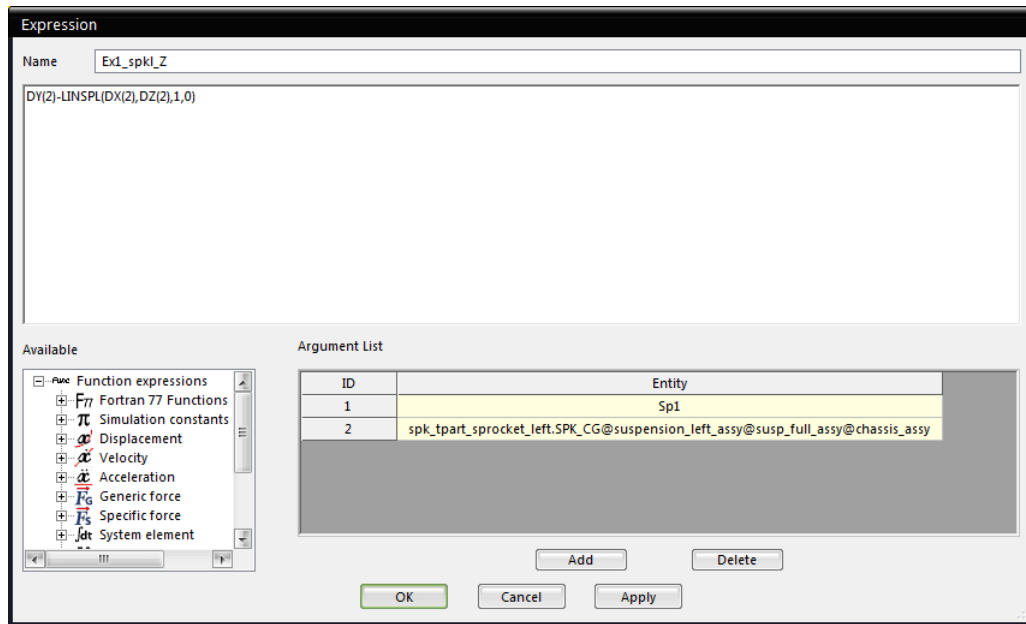


Figure 3 – Relative Height Expression Example

This technique provides the relative marker height as long as a 3D spline of the road surface is available.

REFERENCE

- [1] RecurDyn (<https://functionbay.com/en>,
<https://functionbay.com/documentation/onlinehelp/default.htm#!Documents/help.htm>)
- [2] SubEntity Spline (<https://functionbay.com/documentation/onlinehelp/default.htm#!Documents/steptocreateaspline.htm>)
- [3] Interpolation LINSPL (<https://functionbay.com/documentation/onlinehelp/default.htm#!Documents/linspl.htm>)
- [4] MATLAB (www.mathworks.com/)
- [5] TN 15-0709-01 “FRAME TWISTER” WAVE COURSE GENERATION IN 3D
- [6] TN 15-0710-01 ROAD SURFACE GENERATION IN RDF FORMAT

Appendix A.1 – SubEntity/Spline

Subentity / Spline

Step to create a spline

1. Click the **Spline** icon and then the **Spline List** dialog box appears as follows.

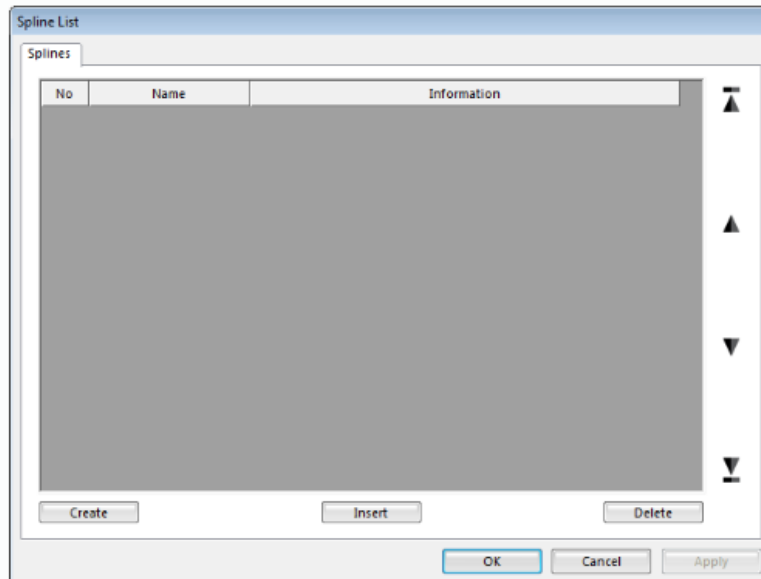


Figure 1 **Spline List** dialog box

- **No:** Shows the number of spline.
- **Name:** Shows the name of spline.
- **Information:** Shows the number of row about the defined data.
- **Create:** Creates a spline.
- **Insert:** Inserts a spline at the current cursor row.
- **Delete:** Deletes a spline at the current cursor row.

2. Click the **Create** or **Insert** button, and then the **Spline** dialog box appears.

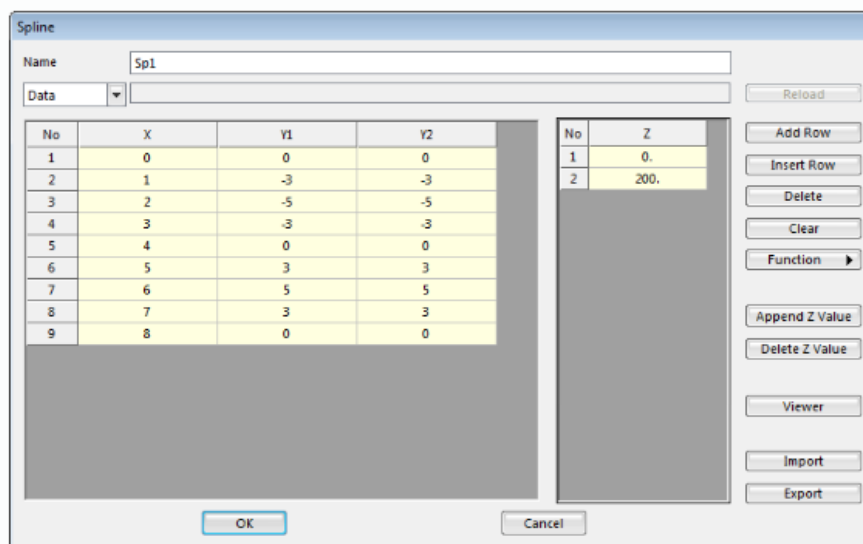


Figure 2 **Spline** dialog box

- **Add Row:** Adds a pair of data
- **Insert Row:** Inserts a pair of data
- **Delete:** Deletes a pair of data
- **Clear:** Deletes all data of all rows.
- **Function:** Creates a **linear line data** by inputting **Slope** and **absolute Value**. If the user inputs 10, the number of data is 11.

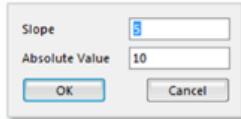


Figure 3 **Function** dialog box

- **Slope:** Input value of slope for linear line.
 - **Absolute Value:** if the user inputs 10, x values are between -10 to 10.
- **Append Z Value:** Append Z data. It is used to create a **3D Spline** and Y data (Y1, Y2... Column) is added automatically.
 - **Delete Z Value:** Deletes Z data.
 - **Draw:** Draw the spline data. Refer to [Using Spline Viewer dialog](#).
 - **Import:** Import the spline data.
 - **Export:** Export the spline data.
 - Supported formats are the **Text** File(*.txt) and the **Matrix Data** File(*.mat).
3. Click the **Add** button.
 4. Specify **Name** and the spline data by **x and y coordinates** in the **Spline** dialog box.
 5. Click the **OK** button.

More:

^ [Using Spline Viewer dialog](#)

Appendix A.2 – Interpolation/LINSPL

Subentity / Expression / Available Functions / Interpolation

LINSPL

The **LINSPL** function uses linear spline interpolation to return the y values for the x variable input through the spline entity.

Format

LINSPL(x, z, Curve name, Order)

Arguments definition

X	An input variable for the LINSPL function Generally, this variable is time or a function that returns a real number.
Z	An input variable for the LINSPL function - The second variable is necessary for three dimensional spline functions. - This variable must be a function that returns a real number. Otherwise, 0 is applied. $y1=f(x,z1), y2=f(x,z2) \quad z1 < z < z2$ $y = (z - z1) * (y2 - y1) / (z2 - z1) + y1$
Curve name	The name or argument number of the spline data defined by the subentity
Order	The interpolation method for the functions (return the value if 0, return calculation for 1st order differential equation if 1) $dy1/dx = df(x,z1)/dx, dy2/dx = df(x,z2)/dx$ $dy/dx = (z - z1) * (dy2/dx - dy1/dx) / (z2 - z1) + dy1/dx$

Formulation

$$\text{LINSPL} = \begin{cases} f(x, \text{spline\_data}), & \text{Order}=0 \\ df(x, \text{spline\_data}) / dx, & \text{Order}=1 \end{cases}$$

Example

LINSPL(time-1,0,1,0) <Argument: (1) Spline1>

No	X	Y1
1	-1.	0.
2	0.	100.
3	1.	100.
4	2.	0.

Figure 1 Spline Data

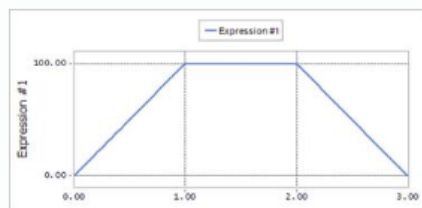


Figure 2 Example using the **LINSPL** function

TN 16-0419-01

TRAILER PAYLOAD POSITION LIMITS

April 19, 2016

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC)
Detroit Arsenal
Systems Integration & Engineering
ATTN: RDTA-SIE/MS157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

What are the restrictions on payload placement given the following requirements from [1]?

3.9.5 Wheels and tires. Tires and wheels shall conform to Federal Motor Vehicles Safety Standards (FMVSS) 571.119 and 571.120. All tire and wheel ratings and dimensions shall conform to the Tire and Rim Association (T&RA) or the European Tire and Rim Technical Organization (ETRTO) Standards Manual at the maximum speed of the M870A4. Tires and wheels rated capacity shall be at least 15% greater than the maximum load imposed on each tire, measured at each wheel, at the ground, with the trailer at GVW. Wheel and tire assemblies currently in the Army system shall be assessed for form, fit, function, and ruled suitable or rejected for use. (Reference TACOM Administrator's Manual (AM) for use with SAE J2014, (R) Pneumatic Tire/Wheel/Runflat Assembly Qualifications for Military Tactical Wheeled Vehicles Appendix A.) Wheel and tire assemblies shall be balanced.

3.11.2 Fifth wheel load. The load on the prime mover's fifth wheel shall not exceed 40,000 pounds (18140 kg) when the M870A4 is at GVW or when carrying any of the equipment identified in Appendix A.

PROBLEM SOLUTION:

From Figure 1 equation (1) defines the basic relationship between the trailer, payload, and system center-of-gravity's (CG's). The variables in Figure 1 are described more fully in Table 1. Equation (1a) shows how the payload position depends on the other variables.

$$TCGX(TM) + XP(PM) = SX(SM) \quad (1)$$

$$XP = \frac{SX(SM) - TCGX(TM)}{PM} \quad (1a).$$

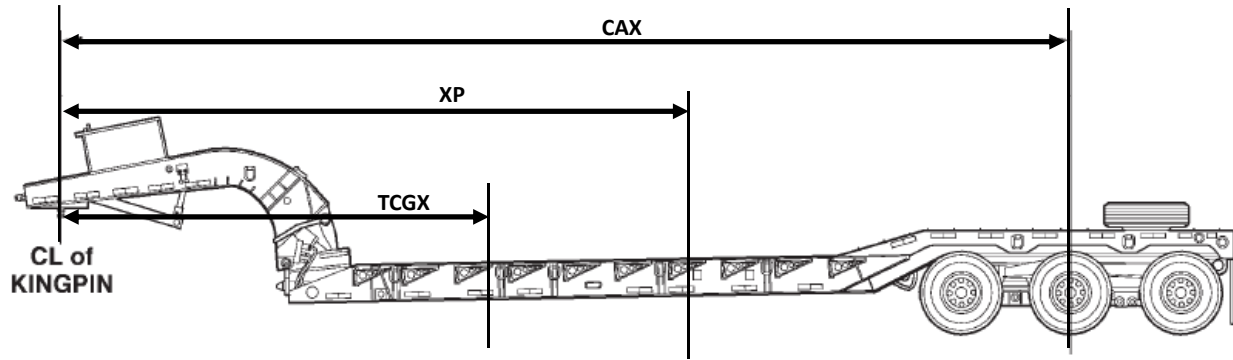


Figure 1 – Semi-trailer Axle and Mass Positions

From Figure 1, again, equation (2) defines the relationship between equivalent moments of the system CG ($SX \cdot SM$) and suspension load ($CAX \cdot SL$). Equation (2a) shows how the system position depends on the other variables.

$$SX(SM) = CAX(SL) \quad (2)$$

$$SX = \frac{CAX(SL)}{SM} \quad (2a).$$

Table 1 illustrates how the payload position limits are calculated for different assumptions. The bordered cells are required inputs. Positions rear of the kingpin (X) are negative. General characteristics are calculated for empty and payload conditions. For the example in Table 1 dual wheels are used. The payload position limits (green cells) are calculated with additional characteristics that include the suspension load, kingpin load, and kingpin % load for the three cases specified and described below. The difference in length between the two extreme payload positions is included last. Example values in Table 1 may not reflect actual trailer characteristics as they are used for illustrational purposes (as can be seen that the last two conditions show a conflict. Reducing the payload will alleviate this condition as is shown in Appendix A.2).

Midpoint

In this case the system CG (SX) is assumed to be at the trailer center of the wheel base

$$SX = \frac{CAX}{2} \quad (3)$$

which from (2a) gives

$$SL = \frac{SM}{2} \quad (3a).$$

Either substitution (equation 3 or equations 2 and 3a) into equation (1a) gives the payload position (XP).

Kingpin

From 3.11.2 in [1] the suspension load (SL) depends on the kingpin load ($KPL=40,000$ lbf) as defined in equation (4)

$$SL = SM - KPL \quad (4)$$

which is combined with equation (2a) and used in equation (1a) to get the payload position (XP).

Tire Rating

From 3.9.5 in [1] the center axle suspension load (SL) is based off of the tire rating plus a 15% safety factor as shown in equation (5). This assumes the tire has the lowest rating in the wheel/axle assembly system.

$$SL = \frac{(tire\ rating\ [lb]) (No. Tires) (No. Axles)}{1.15} \quad (5)$$

Substituting this value into equation (2a) and combining with equation (1a) gives the payload position (XP).

Differences between the effects of equation (4) and (5) are shown as the payload position length. This is the effective “window” in which the payload can be loaded to meet the above requirements. The greater the length, the bigger this “window”, and thus more flexibility is allowed in payload placement.

Page 2-18 from [2] in Appendix A.1 provides a method to calculate the kingpin and axle load for the payload given the empty trailer weight at the front and rear. Using equations (6) – (8) and solving for X (XP in Table 1) gives the same results as for the kingpin and tire rating calculations in Table 1 as described above.

$$EWR = \frac{TCGX(TM)}{CAX} \quad (6)$$

$$EWF = TM - EWR \quad (7)$$

$$Y = CAX - X \quad (8)$$

From the above analysis, payload position limits can be found based on criteria for the maximum kingpin and tire rating (center axle) loads.

Table 1 – Payload Position Limits Example Calculations

Description	Value	Units	Variable	Formula	Notes
Payload Mass	90000.00	lbf	PM		
Start of Deck X from Kingpin	-100.00	in	DS		
Deck Length (flat)	400.00	in	DL		
Center Axle X from Kingpin	-450.00	in	CAX		
Trailer CGX from Kingpin	-320.00	in	TCGX		
Trailer Mass	50.00	lbf-sec ² /in	TM		
Trailer Mass	19304.40	lbf	TM		
No. Axles	3.00		NA		
Tires per Axle	4.00		TPA		dual tires
Tire Rating	6390.00	lbf			
Empty AXLE Load	4575.86	lbf	EA	(TCGX*TM)/CAX/NA	
Empty WHEEL (Dual) Load	2287.93	lbf		EA/2	
Payload X from Kingpin	-300.00	in		DS-DL/2	mid-pt of deck
System Mass (trailer + payload)	109304.40	lbf	SM		
System X from Kingpin	-303.53	in	SX	TM+PM	
GVW AXLE Load	24575.86	lbf	SAL	(SM*SX)/CAX/NA	
GVW WHEEL (Dual) Load	7770.38	lbf		(SAL*SX)/TCGX/NA	
Kingpin Load %	32.55	%		(SM-SAL*NA)/SM*100	
PAYLOAD CHECK					
Deck End	-500.00	in	DE	DS-DL	
Deck Mid	-300.00	in	DM	DS-DL/2	
Center Axle X/2	-225.00	in		CAX/2	
Center Axle X/2 fwd Deck Mid	75.00	in		CAX/2-DM	
X Payload (SX =CAX/2)	-204.62	in	XP_sxmid	((CAX/2)*SM-TCGX*TM)/PM	
Fwd from CAX/2	20.38	in		XP_sxmid-CAX/2	
Fwd from Deck Mid	95.38	in		XP_sxmid-DM	
From Deck Start	104.62	in		DS-XP_sxmid	
Suspension Load (@CAX)	54652.20	lbf	SL	SX*SM/CAX	SX=CAX/2
Kingpin Load	54652.20	lbf	HL	SM-SL	
Kingpin % Load	50.00	%		(HL/SM)*100	
Max Suspension Load @ Kingpin (40K)	40000.00	lbf	KPL		
SX for 40K KP	-285.32	in	SX_kpl	CAX*(SM-KPL)/SM	
X Payload (kp 40k)	-277.88	in	XP_sxkpl	(CAX*(SM-KPL)-TCGX*TM)/PM	
Fwd from Deck Mid	22.12	in		XP_sxkpl-DM	
From Deck Start	177.88	in		DS-XP_sxkpl	
Suspension Load (@CAX)	69304.40	lbf	SL	SX*SM/CAX	SX=SX_kpl
Kingpin Load	40000.00	lbf	HL	SM-SL	
Kingpin % Load	36.60	%		(HL/SM)*100	
Tire Rating (from above)	6390.00	lbf			
Center Axle Load (Tire Rating + 15%)	66678.26	lbf	SL		
X Payload (Tire Rating)	-264.75	in	xs	(CAX*SL-TCGX*TM)/PM	
Fwd from Deck Mid	35.25	in		XP_sxtire-DM	
From Deck Start	164.75	in		DS-XP_sxtire	
Suspension Load (@CAX)	66678.26	lbf			SX=CAX
Kingpin Load	42626.14	lbf	HL	SM-SL	
Kingpin % Load	39.00	%		(HL/SM)*100	
Payload Position Length	-13.13	in	ZN	XP_sxkpl-XP_sxtire	

REFERENCE

- [1] PURCHASE DESCRIPTION, M870A4 SEMITRAILER, LOWBED, CONSTRUCTION EQUIPMENT TRANSPORTER (SLCET) 40 TON. ATPD-2395G. (05 February 2015). DIST A. Approved for public release: distribution unlimited.

- [2] TM 5-2330-325-14&P. OPERATOR'S, UNIT, AND DIRECT SUPPORT MAINTENANCE MANUAL WITH REPAIR PARTS AND SPECIAL TOOLS LIST (RPSTL) FOR TRAILER, MEDIUM HEAVY EQUIPMENT TRANSPORTER (MHET), 40 TON, M870A3 (NSN 2330-01-458-2061) (June 2004). DIST A. Approved for public release: distribution unlimited. Inactive, see (http://www.combatindex.com/store/tech_man/Sample/HET_Trailers/TM_5-2330-325-14_P.pdf)
- [3] TM 5-2330-325-13&P. OPERATOR AND FIELD MAINTENANCE MANUAL INCLUDING REPAIR PARTS AND SPECIAL TOOLS LIST (RPSTL) FOR SEMITRAILER, MEDIUM HEAVY EQUIPMENT TRANSPORTER (MHET), 40 TON, M870A3 NSN 2330-01-458-2061 (EIC CFJ) (23 June 2014). DIST A. Approved for public release: distribution unlimited. (<https://www.ldac.army.mil/>). This manual supersedes TM 5-2330-325-14&P dated 30 June 2004, including all changes.

APPENDIX A.1 – Page from TM 5-2330-325-14&P (see [2])

TM 5-2330-325-14&P

2-10. CALCULATING CENTER OF GRAVITY (Continued).

- c. Use the data and formulas provided below to determine if the trailer is loaded properly and within its limits. Known information for the trailer is provided. Information for the payload will need to be calculated through measurements and from data contained on the payload, vehicle, or equipment.

CG = Center of Gravity

M870A3 Data

CL = Center Line

WB = 448.06 in. (37 ft 4 1/16 in.)

EWf = Trailer Empty Weight at Front

EWf = 7,840 lb (3,556.2 kg)

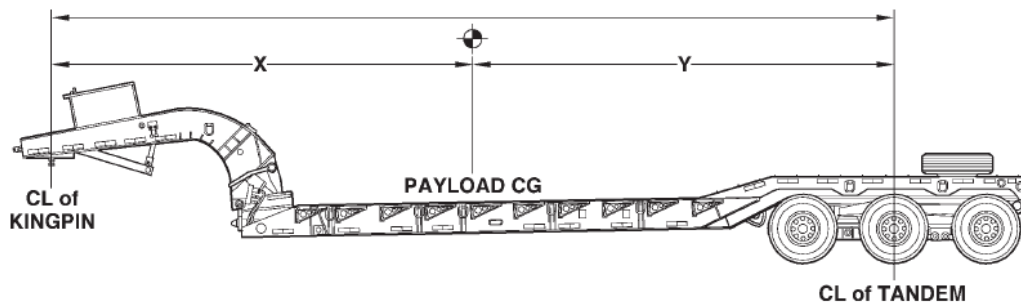
EWr = Trailer Empty Weight at Rear

EWr = 15,320 lb (6,940.0 kg)

WB = Wheelbase

$$\frac{(X) \times \text{Payload Weight}}{\text{WB}} + \text{EWr} = \text{Axle Load}$$

$$\frac{(Y) \times \text{Payload Weight}}{\text{WB}} + \text{EWf} = \text{Kingpin Load}$$



Calculating Trailer Center of Gravity with Payload

APPENDIX A.2 – Table 1 with PM=80,000 lbf

Description	Value	Units	Variable	Formula	Notes
Payload Mass	80000.00	lbf	PM		
Start of Deck X from Kingpin	-100.00	in	DS		
Deck Length (flat)	400.00	in	DL		
Center Axle X from Kingpin	-450.00	in	CAX		
Trailer CGX from Kingpin	-320.00	in	TCGX		
Trailer Mass	50.00	lbf-sec ² /in	TM		
Trailer Mass	19304.40	lbf	TM		
No. Axles	3.00		NA		
Tires per Axle	4.00		TPA		dual tires
Tire Rating	6390.00	lbf			
Empty AXLE Load	4575.86	lbf	EA	(TCGX*TM)/CAX/NA	
Empty WHEEL (Dual) Load	2287.93	lbf		EA/2	
Payload X from Kingpin	-300.00	in		DS-DL/2	mid-pt of deck
System Mass (trailer + payload)	99304.40	lbf	SM		
System X from Kingpin	-303.89	in	SX	TM+PM	
GVW AXLE Load	22353.64	lbf	SAL	(SM*SX)/CAX/NA	
GVW WHEEL (Dual) Load	7076.04	lbf		(SAL*SX)/TCGX/NA	
Kingpin Load %	32.47	%		(SM-SAL*NA)/SM*100	
PAYLOAD CHECK					
Deck End	-500.00	in	DE	DS-DL	
Deck Mid	-300.00	in	DM	DS-DL/2	
Center Axle X/2	-225.00	in		CAX/2	
Center Axle X/2 fwd Deck Mid	75.00	in		CAX/2-DM	
X Payload (SX = CAX/2)	-202.08	in	XP_sxmid	((CAX/2)*SM-TCGX*TM)/PM	
Fwd from CAX/2	22.92	in		XP_sxmid-CAX/2	
Fwd from Deck Mid	97.92	in		XP_sxmid-DM	
From Deck Start	102.08	in		DS-XP_sxmid	
Suspension Load (@CAX)	49652.20	lbf	SL	SX*SM/CAX	SX=CAX/2
Kingpin Load	49652.20	lbf	HL	SM-SL	
Kingpin % Load	50.00	%		(HL/SM)*100	
Max Suspension Load @ Kingpin (40K)	40000.00	lbf	KPL		
SX for 40K KP	-268.74	in	SX_kpl	CAX*(SM-KPL)/SM	
X Payload (kp 40k)	-256.37	in	XP_sxkpl	(CAX*(SM-KPL)-TCGX*TM)/PM	
Fwd from Deck Mid	43.63	in		XP_sxkpl-DM	
From Deck Start	156.37	in		DS-XP_sxkpl	
Suspension Load (@CAX)	59304.40	lbf	SL	SX*SM/CAX	SX=SX_kpl
Kingpin Load	40000.00	lbf	HL	SM-SL	
Kingpin % Load	40.28	%		(HL/SM)*100	
Tire Rating (from above)	6390.00	lbf			
Center Axle Load (Tire Rating + 15%)	66678.26	lbf	SL		
X Payload (Tire Rating)	-297.85	in	xs	(CAX*SL-TCGX*TM)/PM	
Fwd from Deck Mid	2.15	in		XP_sxtire-DM	
From Deck Start	197.85	in		DS-XP_sxtire	
Suspension Load (@CAX)	66678.26	lbf			SX=CAX
Kingpin Load	32626.14	lbf	HL	SM-SL	
Kingpin % Load	32.85	%		(HL/SM)*100	
Payload Position Length	41.48	in	ZN	XP_sxkpl-XP_sxtire	

TN 16-0722-01

ENGLISH UNITS OF LBM VERSUS LBF

July 22, 2016

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC)
Detroit Arsenal
Systems Integration & Engineering
ATTN: RDTA-SIE/MS157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

What is the difference between pounds mass (lbm) and pounds force (lbf) in English units?

PROBLEM SOLUTION:

In English units the mass is defined in terms of the weight in the gravitational system whose units can change. Denoting the following

$$|[\]| = \text{magnitude or value}$$

$$[\] = \text{units}$$

Newton's Second Law of motion states that $\text{force} = \text{mass} * \text{acceleration}$ or

$$F = ma \quad (1).$$

For the earth's gravitational system $a = g$ ($g_{\text{mass}} \rightarrow g_{\text{ft}} = 32.2 \frac{\text{ft}}{\text{s}^2}$ or $g_{\text{in}} = 386.088 \frac{\text{in}}{\text{s}^2}$) hence

$$m = \frac{F_{\text{mass}}}{g_{\text{mass}}} \quad (2).$$

Substituting back into (1) gives

$$F = \left(\frac{F_{\text{mass}}}{g_{\text{mass}}} \right) a \quad (3)$$

or

$$|force|[lbf] = \frac{|mass|[lbm]}{|g|[g]} |a|[a] \quad (4).$$

Depending on the units of g , (ft/s^2 or in/s^2) in (4), the following are definitions

$$1[\text{slug}] = 1 \frac{[\text{lbf}]}{[g_{ft}]} \rightarrow |\text{slug}| = \frac{|\text{mass}|}{|g_{ft}|}$$

$$1[\text{slinch}] = 1 \frac{[\text{lbf}]}{[g_{in}]} \Rightarrow |\text{slinch}| = \frac{|\text{mass}|}{|g_{in}|}.$$

Note that

$$1[\text{slug}] = 1 \frac{[\text{lbf}]}{[g_{ft}] \left(\frac{12[g_{in}]}{[g_{ft}]} \right)} = \frac{1[\text{slinch}]}{12} \rightarrow 12[\text{slug}] = 1[\text{slinch}]$$

If $a = g$ then the numerical value of pounds mass and pounds force are equivalent from (4)

$$|\text{force}|[\text{lbf}] = |\text{mass}|[\text{lbf}]|a|[\text{a}] = |\text{mass}| \frac{[\text{lbf}]}{[g][g]} |a|[\text{a}] = |\text{mass}|[\text{lbf}] \rightarrow |\text{force}| = |\text{mass}| \quad (5).$$

Table 1 - Definitions

[lbm]	g	[slug]	[slinch]
32.2	g_{ft}	1	1/12
386.088	g_{in}	12	1

Table 2 - Examples

[lbm]	a	[lbf]
32.2	g_{ft}	32.2
386.088	g_{in}	386.088
100	$1 \frac{ft}{s^2}$	3.1055
100	g_{ft}	100
100	$1 \frac{in}{s^2}$	0.259
100	g_{in}	100

We conclude with the following important conversions

$$1 \text{ lbm} = 0.45359237 \text{ kg}$$

$$1 \text{ lbf} = 32.174 \text{ lbm} \frac{ft}{s^2}$$

REFERENCE

[1] [https://en.wikipedia.org/wiki/Pound_\(force\)](https://en.wikipedia.org/wiki/Pound_(force))

TN 16-0922-01

BUSHING RADIAL RATE DECAY MAXIMUM CURVATURE

September 22, 2016

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC)
Detroit Arsenal
Systems Integration & Engineering
ATTN: RDTA-SIE/MS157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

TARDEC's Track and Suspension Elastomer Laboratory requested help in finding the "knee in the curve" for MTS-832 [1] testing of bushing radial rate decay data.

PROBLEM SOLUTION:

The terminology "knee in the curve" refers to "the point of maximum curvature". No "knee" is seen in the normal plot in Figure 1 but does stand out in the logarithmic plot in Figure 2.

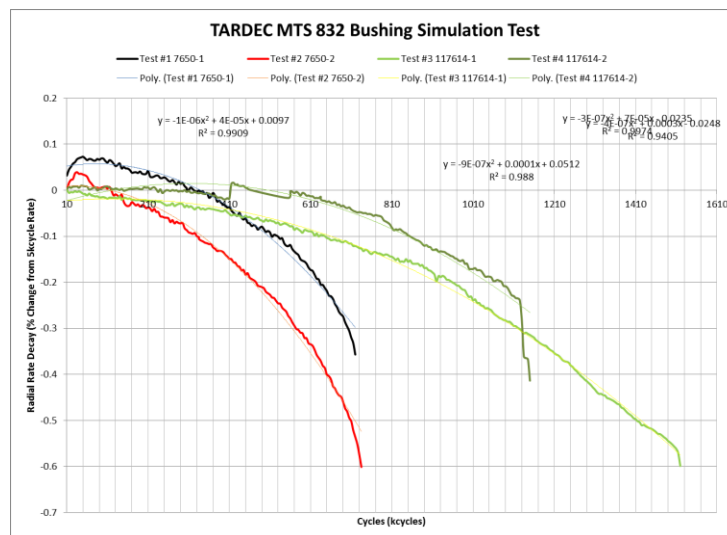


Figure 1 – Normal Plot

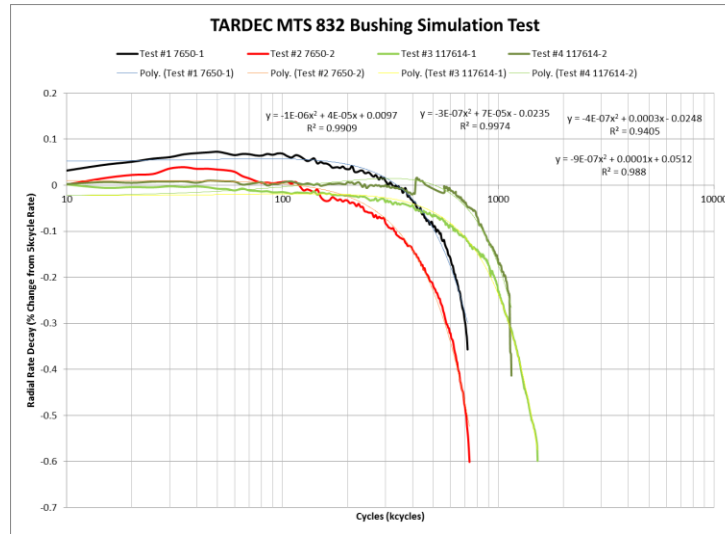


Figure 2 – Logarithmic Plot

From standard calculus and [2] curvature is defined as

$$\kappa = \frac{|y''|}{[1 + (y')^2]^{\frac{3}{2}}} \quad (1).$$

Test data is fitted to a second order polynomial (this can be done using trend lines in Microsoft Excel) of the form

$$y = c_1x^2 + c_2x + c_3 \quad (2).$$

Although the data is defined for x the rate of change for the curvature is with respect to the log (base 10) of x as shown in Figure 2. Using the chain rule from calculus

$$\frac{dy}{d(\log_b x)} = \frac{dy}{du} \quad (3)$$

where

$$u = \log_b x \quad (4)$$

and

$$\frac{d(\log_b x)}{dx} = \frac{du}{dx} = \frac{1}{x \ln b} \quad (5).$$

From (3) and (5)

$$\frac{dy}{d(\log_b x)} = \frac{dy}{du} = \frac{dy}{dx} \frac{dx}{du} = \frac{dy}{dx} x \ln b \quad (6).$$

From (6) and (2)

$$y' = \ln b [2c_1x^2 + c_2x] \quad (7)$$

$$y'' = (\ln b)^2 [4c_1x^2 + c_2x] \quad (8).$$

For each x , values for equations (7) and (8) can be inserted into (1) to find the curvature. The x value giving the maximum is the point of maximum curvature (see Appendix A.1).

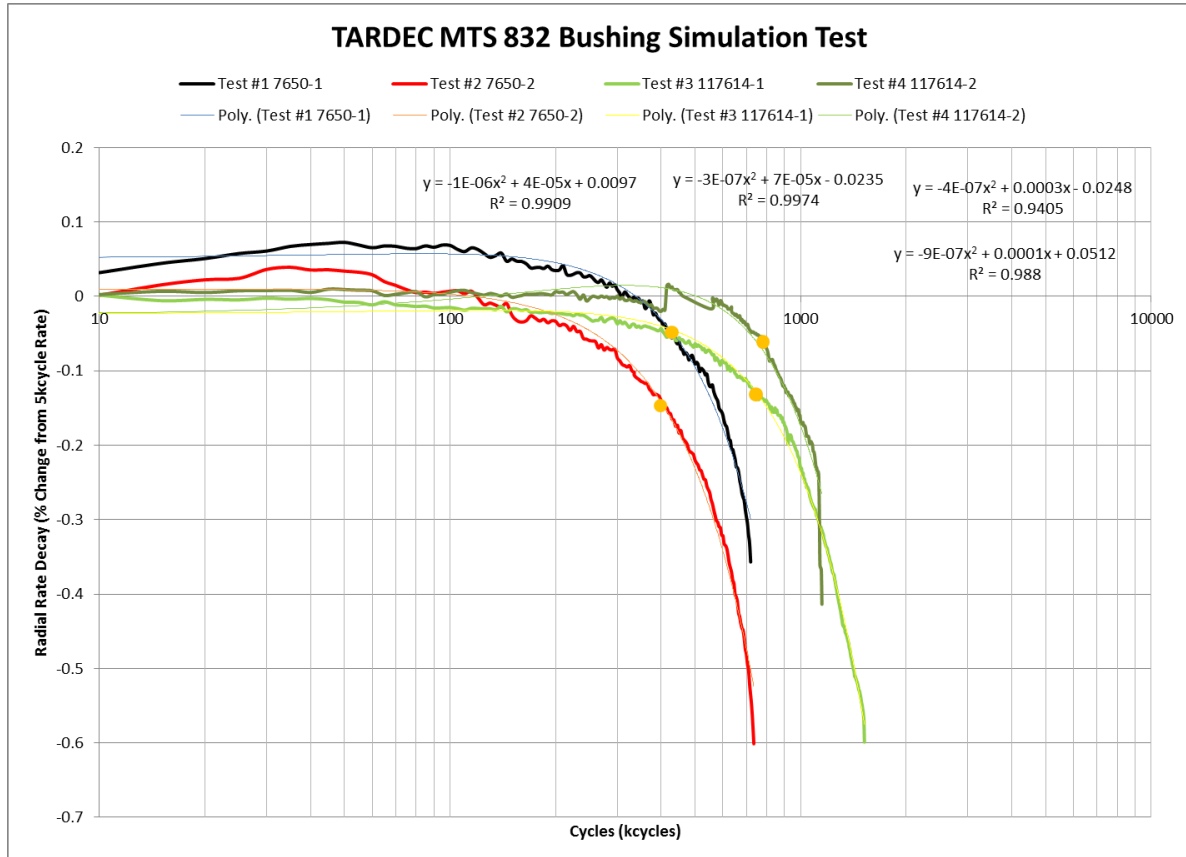


Figure 3 – Points of Maximum Curvature

Results for the maximum curvature of the second order polynomials in Figure 3 are 435, 400, 765, and 790 cycles and are so designated with the large orange circles. This method seems to give consistent and appropriate results.

REFERENCE

- [1] 832 Elastomer Biaxial Test Frame (<https://www.mts.com/en/products/automotive/elastomer-test-systems/elastomer-test-systems>)
- [2] Curvature (<https://en.wikipedia.org/wiki/Curvature>)

APPENDIX A.1 – Example Excel Spreadsheet

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1		1 dark			2 red			3 light green			4 dark green			
2	A	-9.00E-07			-1.00E-06			-3.00E-07			-4.00E-07			
3	B	0.0001			4.00E-05			7.00E-05			0.0003			
4	C	0.512			0.0097			0.0235			-0.0248			
5	X Col	A			H			O			V			
6	Y Col	F			M			T			AA			
7	Start	3			3			3			3			
8	End	148			151			307			217			
9		158			161			317			227			
10	Max			1.900916			1.819052			1.930749			2.532616	
11				435			400			765			790	
12		y'	y''	k	y'	y''	k	y'	y''	k	y'	y''	k	
13	3	0.00E+00	0.00E+00	0	0.00E+00	0.00E+00	0	0.00E+00	0.00E+00	0	0.00E+00	0.00E+00	0	
14	4	1.05E-03	2.17E-03	0.002174	0.00E+00	0.00E+00	0	7.71E-04	1.70E-03	0.001697	3.41E-03	7.74E-03	0.007741	
15	5	1.89E-03	3.39E-03	0.003393	4.61E-04	2.87E-19	2.87E-19	1.47E-03	3.08E-03	0.003075	6.72E-03	1.51E-02	0.015056	
16	6	2.52E-03	3.66E-03	0.003658	3.45E-04	-1.59E-03	0.001591	2.11E-03	4.14E-03	0.004135	9.95E-03	2.19E-02	0.021947	

- 1) create a separate spreadsheet to do the curvature calculations
- 2) define the second order polynomial coefficients from the Excel trend line equations displayed on the chart (here A,B,C is equivalent to c1, c2, c3) for each data set
- 3) define the input data columns (X,Y) and the start and ending rows for each data set (these were obtained from spreadsheet "Sheet1" where the original data is from)
- 4) for each x value create columns for y', y'', and k for each data set. For the first cells of data set 1: y' is $=\text{LN}(10)*(2*\$B\$2*\text{Sheet1}!\text{A3}^2+\$B\$3*\text{Sheet1}!\text{A3})$, y'' is $=\text{LN}(10)^2*(4*\$B\$2*\text{Sheet1}!\text{A3}^2+\$B\$3*\text{Sheet1}!\text{A3})$, and k is $=\text{ABS}(\text{C13})/((1+\text{B13}^2)^(3/2))$
- 5) above the curvature (k) column define the maximum value for k using the MAX function for each data set. For data set 1 the maximum k is $=\text{MAX}(\text{D13:D148})$
- 6) above the k column define the x value or "knee in the curve" associated with the maximum k. For data set 1 the x value is found from the formula $=\text{INDEX}(\text{Sheet1}!\text{A3:A148},\text{MATCH}(\text{D10},\text{D13:D158},0))$. Here the MATCH function finds the index of the maximum value and the INDEX function uses that index to get the associated data value.

TN 16-1207-01

BUSHING RADIAL RATE DECAY MAXIMUM CURVATURE - CALCULATING IN EXCEL AND MATLAB

December 7, 2016

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC)
Detroit Arsenal
Systems Integration & Engineering
ATTN: RDTA-SIE/MS157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

How can the maximum curvature in TN 16-0922-01 BUSHING RADIAL RATE DECAY MAXIMUM CURVATURE be calculated without using Trendlines in Microsoft Excel?

PROBLEM SOLUTION:

Trendlines in [2] are used to provide a second order polynomial equation fit to the data in [1]. The equation can be determined by using the "least squares" method explicitly in Excel [2] and MATLAB [3].

The main equations developed in [1] are outlined below

$$\kappa = \frac{|y''|}{[1 + (y')^2]^{\frac{3}{2}}} \quad (1)$$

where

$$y = c_1x^2 + c_2x + c_3 \quad (2)$$

$$y' = \ln b [2c_1x^2 + c_2x] \quad (3)$$

$$y'' = (\ln b)^2 [4c_1x^2 + c_2x] \quad (4)$$

EXCEL

The spreadsheet shown in Figure 1 illustrates how Excel can be used to directly compute the coefficients for a second order polynomial. Table 1 shows the formulas used to develop the data.

			Worksheet Name	Row Start	X Col	Y Col	Row End				a	b	c	k	x@k	Average for each material type	
Torsional Rate Decay	T	15	Torsional Data	3	A	C	148	'Tor \$A\$' Tor \$C\$	-9.6708E-08	-5.4838E-04	-1.1133E-02	1.017814313	510	488		1	
	T	.5	Torsional Data	3	D	F	151	'Tor \$D\$' Tor \$F\$	-7.6551E-08	-6.2975E-04	1.3130E-02	0.973943787	465			2	
	T	7	Torsional Data	3	G	I	288	'Tor \$G\$' Tor \$I\$	-1.7038E-07	-5.5641E-05	4.5852E-02	1.633558689	885			3	
	T	7	Torsional Data	3	J	L	280	'Tor \$J\$' Tor \$L\$	-2.0227E-07	2.5847E-05	3.5993E-02	1.840161984	900			4	
	T	7	Torsional Data	3	M	O	187	'Tor \$M\$' Tor \$O\$	-5.4767E-07	2.2123E-04	-6.6489E-03	2.191959227	615	800		5	
	T	416	Torsional Data	3	P	R	212	'Tor \$P\$' Tor \$R\$	-3.7476E-07	1.4810E-04	-1.9671E-04	2.097985887	725			6	
	T	7	Torsional Data	3	S	U	266	'Tor \$S\$' Tor \$U\$	-1.7152E-07	-8.9802E-05	2.7403E-02	1.560895955	845			7	
	T	7	Torsional Data	3	V	X	231	'Tor \$V\$' Tor \$X\$	-1.6765E-07	-7.2118E-05	2.5447E-02	1.595891872	875			8	
	T	-1)	Torsional Data	3	Y	AA	307	'Tor \$Y\$' Tor \$AA	-2.3811E-07	1.6503E-04	-4.8595E-02	2.261276425	955	860		9	
	T	-2)	Torsional Data	3	AB	AD	217	'Tor \$AB\$' Tor \$AD	-3.9193E-07	2.4530E-04	-1.0776E-02	2.3621246	765			10	
																	11
Radial Rate Decay	T	15	Radial Data	3	A	C	148	'Rac \$A\$' Rac \$C\$	-9.1525E-07	1.7959E-04	4.5082E-02	2.01520718	450	420		12	
	T	.5	Radial Data	3	D	F	151	'Rac \$D\$' Rac \$F\$	-1.0441E-06	4.3195E-05	8.5329E-03	1.821647624	390			13	
	T	7	Radial Data	3	G	I	288	'Rac \$G\$' Rac \$I\$	-2.8604E-07	-1.0185E-04	1.3816E-01	1.583350752	665			14	
	T	7	Radial Data	3	J	L	280	'Rac \$J\$' Rac \$L\$	-3.0296E-07	5.4461E-05	1.3136E-01	1.892514972	750			15	
	T	416	Radial Data	3	M	O	212	'Rac \$M\$' Rac \$O\$	-3.4017E-07	-7.7194E-05	-1.4916E-02	1.635814036	625	690		16	
	T	7	Radial Data	3	P	R	217	'Rac \$P\$' Rac \$R\$	-5.4844E-08	-3.2574E-04	2.2912E-02	1.070388079	805			17	
	T	7	Radial Data	3	S	U	212	'Rac \$S\$' Rac \$U\$	-3.2334E-07	-8.0233E-05	-4.8685E-02	1.627679735	640			18	
	T	'614-1)	Radial Data	3	V	X	307	'Rac \$V\$' Rac \$X\$	-2.8481E-07	6.9740E-05	-2.2149E-02	1.934650342	785			19	
	T	'614-2)	Radial Data	3	Y	AA	217	'Rac \$Y\$' Rac \$AA	-3.9610E-07	2.4525E-04	-2.2985E-02	2.357970719	760	773		20	
																	21
Benchmark		Benchmark	2	A	B	202	'Ber \$A\$' Ber \$B\$	3.0000E+00	-3.6000E+01	1.0800E+02	1145.209992	6			22		
																23	
																24	
																25	
																26	
																27	
																28	
																29	
																30	
NOTE:	1) Input Data for Columns A-E																31
	2) Copy function from Columns F-N																32
	3) Array expressions require the use of CTRL-SHIFT-ENTER																33
																	34
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R

Figure 1 – Example Spreadsheet for Bushing Radial Rate Decay Maximum Curvature

Table 1 –Spreadsheet Formula Descriptions

COLUMN	FORMULA (Row 2)
G	=Get_Cend (C2, E2)
H	=" " & C2 & " " ! \$ " & E2 & " \$ " & D2
I	=" \$ " & E2 & " \$ " & G2
J	=" " & C2 & " " ! \$ " & F2 & " \$ " & D2
K	=" \$ " & F2 & " \$ " & G2
L	={LINEST (INDIRECT (J2&" ":"&K2) , INDIRECT (H2&" ":"&I2) ^{1,2}) }
M	
N	
O	
P	= {Get_K (L2:N2, INDIRECT (H2&" ":"&I2)) }

Note that Columns (L, M, N) and (O, P) use array expressions. The function Get_Cend is described in Appendix A.1. The LINEST [2a] and INDIRECT [2b] functions determine the polynomial coefficients. In Columns (L, M, N) only first and second powers of x are used, as the zero power (coefficient c_3 in equation (2)) is implicitly defined by LINEST. The spreadsheet denotes coefficient c_1 as a, c_2 as b, etc.

Once the coefficients are found the function Get_K can compute the maximum curvature and its location as described in Appendix A.2

A benchmark test was defined as the parabola

$$y = 3(x - 6)^2 \quad (5)$$

where the maximum curvature should always be located at the root. Cell P23 in Figure 1 agrees with this.

MATLAB

The approach in MATLAB is to use matrices. Here equation (2) is defined as

$$Y_{N \times 1} = X_{N \times k} B_{k \times 1}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \dots \end{bmatrix} = \begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ \dots & \dots & \dots \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$$
(6)

with the error defined as

$$e = Y - XB$$
(7).

Remember that

$$(AB)' = (B'A')$$
(8)

giving the sum of the error squared as

$$e'e = (Y' - B'X')(Y - XB) = Y'Y - 2X'YB + B'X'XB$$
(9).

Finding the coefficients that minimizes equation (9) requires taking the derivative with respect to B and setting it equal to zero as in

$$\frac{d(e'e)}{dB} = -2X'Y + 2X'XB = 0$$
(10).

Take a pause to reflect on the explanation given in Figure 2 [4] regarding matrix order when using derivatives.

Derivation of least squares estimator

The minimum of $S(b)$ is obtained by setting the derivatives of $S(b)$ equal to zero. Note that the function $S(b)$ has scalar values, whereas b is a column vector with k components. So we have k first order derivatives and we will follow the convention to arrange them in a column vector. The second and third terms of the last expression in (3.6) are equal (a 1×1 matrix is always symmetric) and may be replaced by $-2b'X'y$. This is a linear expression in the elements of b and so the vector of derivatives equals $-2X'y$. The last term of (3.6) is a quadratic form in the elements of b . The vector of first order derivatives of this term $b'X'Xb$ can be written as $2X'Xb$. The proof of this result is left as an exercise (see Exercise 3.1). To get the idea we consider the case $k = 2$ and we denote the elements of $X'X$ by c_{ij} , $i, j = 1, 2$, with $c_{12} = c_{21}$. Then $b'X'Xb = c_{11}b_1^2 + c_{22}b_2^2 + 2c_{12}b_1b_2$. The derivative with respect to b_1 is $2c_{11}b_1 + 2c_{12}b_2$ and the derivative with respect to b_2 is $2c_{12}b_1 + 2c_{22}b_2$. When we arrange these two partial derivatives in a 2×1 vector, this can be written as $2X'Xb$. See Appendix A (especially Examples A.10 and A.11 in Section A.7) for further computational details and illustrations.

T

Figure 2 – Derivative of Matrix Expression Order [4]

From equation (10) then

$$X'Y = X'XB$$
(11)

and therefore

$$B = (X'X)^{-1}X'Y$$
(12)

provides the solution for the second order polynomial coefficients in MATLAB as LINEST did in Excel.

Table 2 shows the examples computed in MATLAB for comparison to the Excel results in Figure 1. They are essentially identical except for numerical round-off differences.

Table 2 –MATLAB Results

ROW	max K (Column O)	x@K (Column P)
13	2.0152	450
14	1.8216	390
20	1.9347	785
21	2.358	760

Computation of the maximum curvature (max K) and location continue from equations (3) and (4) defining

$$G_{N \times k} = \begin{bmatrix} 2x_1^2 & x_1 & 0 \\ 2x_2^2 & x_2 & 0 \\ \dots & \dots & \dots \end{bmatrix} \quad (13)$$

$$H_{N \times k} = \begin{bmatrix} 4x_1^2 & x_1 & 0 \\ 4x_2^2 & x_2 & 0 \\ \dots & \dots & \dots \end{bmatrix} \quad (14)$$

therefore

$$y' = \ln(10) GB \quad (15)$$

$$y'' = \ln(10)^2 HB \quad (16)$$

and the matrix form of equation (1) is

$$\kappa = \frac{|\ln(10)^2 HB|}{[1 + (\ln(10) GB)^2]^{\frac{3}{2}}} \quad (17)$$

where curvature is solved for by element-by-element multiplication as shown in Appendix A.3.

REFERENCE

- [1] See TN16-0922-01 BUSHING RADIAL RATE DECAY MAXIMUM CURVATURE
- [2] Microsoft Excel (<https://products.office.com/en-us/excel>)
- [2a] LINEST (<https://support.microsoft.com/en-us/office/linest-function-84d7d0d9-6e50-4101-977a-fa7abf772b6d>)
- [2b] INDIRECT (<https://support.microsoft.com/en-us/office/indirect-function-474b3a3a-8a26-4f44-b491-92b6306fa261>)
- [3] MATLAB (www.mathworks.com)
- [4] Least Squares Matrix Form (<http://global.oup.com/booksites/content/0199268010/samplesec3>)

APPENDIX A.1 – Get_Cend (Excel)

```
Function Get_Cend(n As Range, r As Range)
    'Find end of data in Worksheet Column and return end row number
    '    n = cell for worksheet name
    '    r = cell for column name

    Dim s As String
    Dim ws As Worksheet

    s = "" & n.Value & ""
    Set ws = Worksheets(s)
    'Debug.Print ws.Name

    Get_Cend = ws.Cells(ws.Rows.Count, Range("'" & n.Value & "'!$" & r &
        "$1").Column).End(xlUp).Row
End Function
```


APPENDIX A.2 – Get_K (Excel)

```

Function Get_K(r As Range, x As Range) As Variant
    'Find maximum curvature and location and return maxK, x@maxK
    ' r = range for coefficients a,b,c
    ' x = range for x values (e.g. 'Torsional Data'!$A$3:$A$148)

    Dim k As Double

    'Debug.Print r.Address, x.Address, x.Worksheet.Name
    a = r.Cells(1, 1).Value
    b = r.Cells(1, 2).Value
    'Debug.Print a, b

    kmax = -10000000000000#
    isav = -1
    t1 = Log(10)
    t2 = Log(10) ^ 2

    For i = 1 To x.Rows.Count
        t3 = b * x.Cells(i, 1).Value
        t4 = 2 * a * x.Cells(i, 1).Value ^ 2
        yp = t1 * (t4 + t3)
        ypp = t2 * (2 * t4 + t3)
        k = Abs(ypp) * ((1 + yp ^ 2) ^ (-3 / 2))
        'Debug.Print k
        If (k > kmax) Then
            kmax = k
            isav = i
        End If
    Next
    Get_K = Array(kmax, x.Cells(isav, 1).Value)
    'Debug.Print isav, kmax, x.Cells(isav, 1).Value
    'Debug.Print x.Rows.Count, x.Cells(3, 1).Value
End Function

```

APPENDIX A.3 – ls.m (MATLAB)

```

% find max k (curvature) and location x

y = dlmread('data.txt'); % [x^2, x, c, y]

x = y(:,1:3); % uses only x or x(:,2)
y1 = y(:,4);

b = inv(x'*x)*x'*y1;
%k = abs(y') / [1+(y')^2]^3/2
G = [2*x(:,2).^2 x(:,2) 0.*x(:,2)]; % y' = lnb[2ax^2+bx] = lnb Gb
H = [4*x(:,2).^2 x(:,2) 0.*x(:,2)]; % y'' = (lnb)^2[4ax^2+bx] = (lnb)^2 Hb
yp = log(10)*G*b; % log = ln (natural log)
ypp = log(10)^2*H*b; % log = ln (natural log)
p1 = abs(ypp);
p2 = (1 + (yp).^2).^(-3/2);

k = p1.*p2;
max(k)
i=find(k==max(k));
x(i,2)

```

TN 17-1026-02

M2 BRADLEY WITH CONCEPT VEHICLE PROTOTYPING (CVP) EXTERNAL SUSPENSION UNIT (ESU) ANIMATION

October 26, 2017

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC)
Detroit Arsenal
Systems Integration & Engineering
ATTN: RDTA-SIE/MS157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

An animation was requested showing a side by side comparison of the M2 Bradley---one with the torsion bar suspension and one with an External Suspension Unit (CVP Technology).

PROBLEM SOLUTION:

A M2 Bradley model developed with RecurDyn [1] was updated to include more detailed graphics (e.g. turret, etc.). This model was then imported as a subsystem twice---one named "base" (for the torsion bar suspension) and one named "esu" (for the external suspension units). See Figures 1 and 2. Each subsystem was shifted 120" laterally (base=-120", esu=+120") and the ground deleted from each and moved to the top level. Further modifications to the "esu" subsystem included updating the spring and damping characteristics (RSDA's) for all 6 road wheel stations to reflect the ESU's (spring/damping curves), inactivating the shocks (TSDA's) copied from the base subsystem (ESU's have rotary damping), and adding graphics for the ESU units.

Animations were created over a computer generated 3" RMS course at 25 MPH to show differences in performance (side and front views). A drop simulation from 24" was included to show static performance. Figures 3 to 5 capture snapshots of these animations

Follow-on analysis included increasing the simulation speed to 30 MPH to try and highlight performance differences. At that speed, differences were easily discernable from the animation.

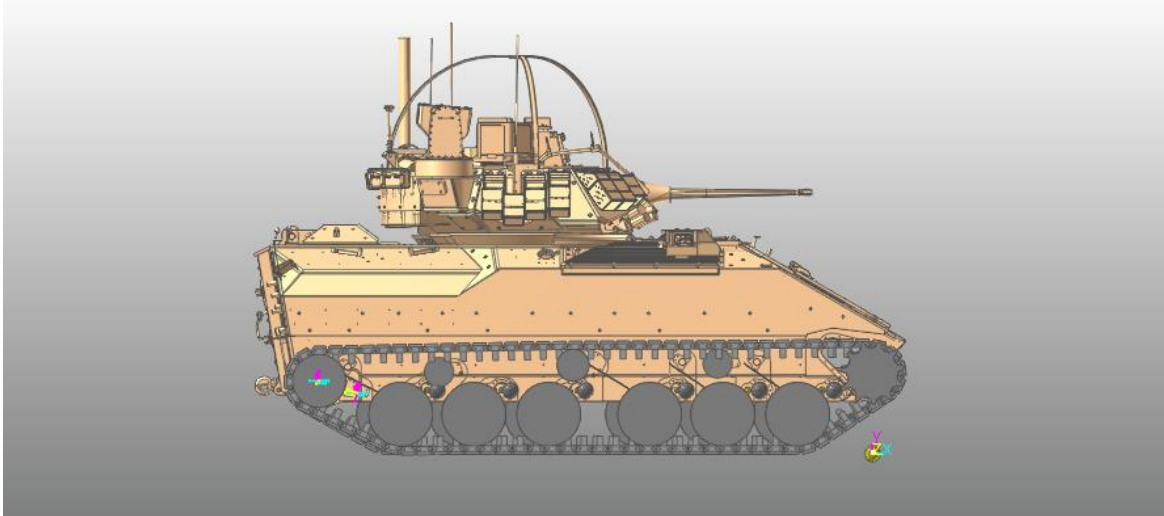


Figure 1 – “base” model

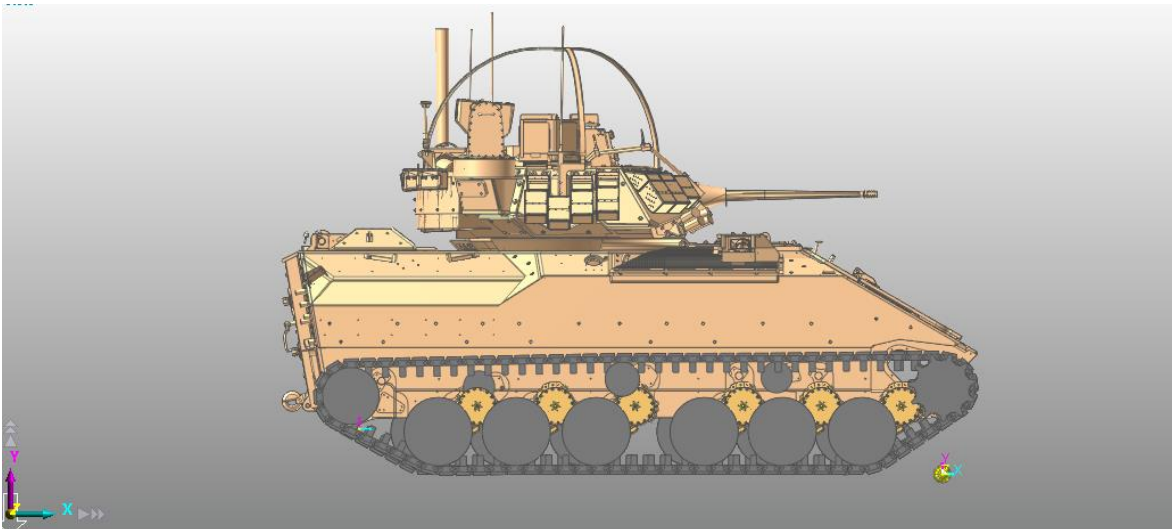


Figure 2 – “esu” model

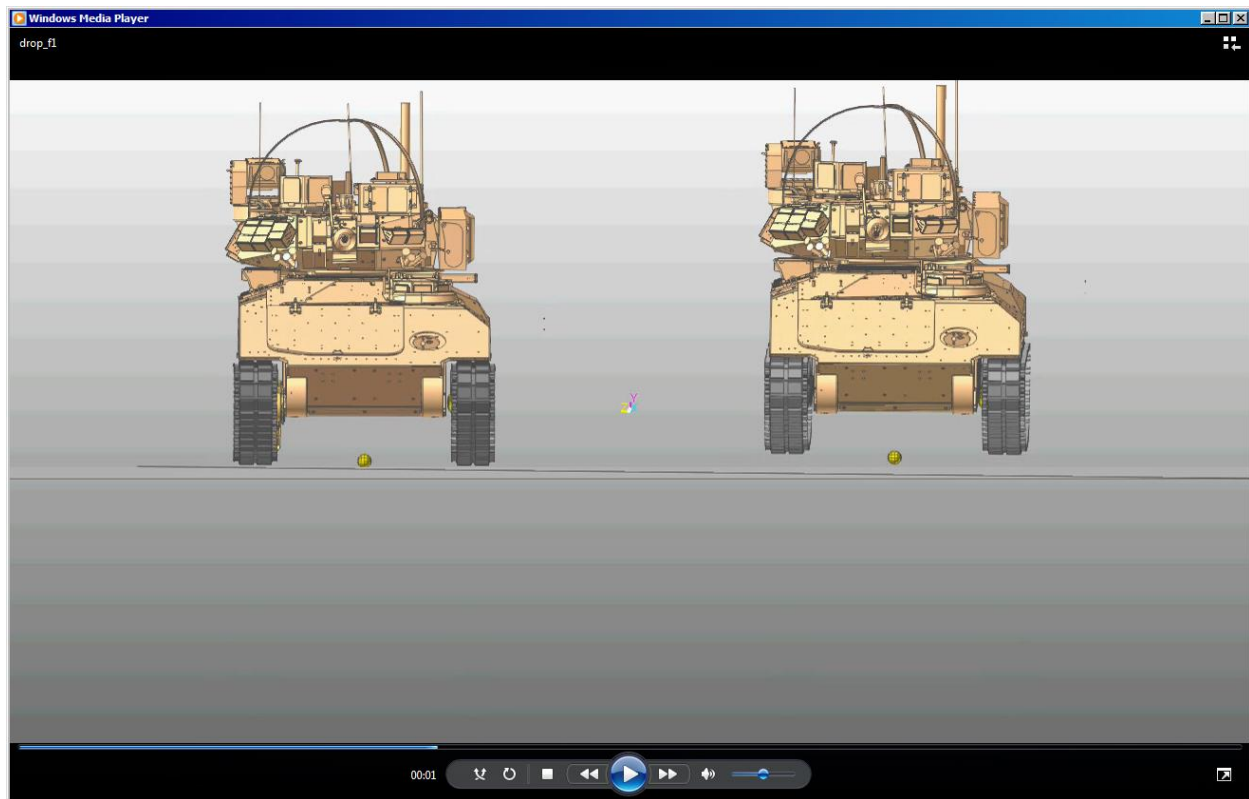


Figure 3 – “drop” animation (24”)

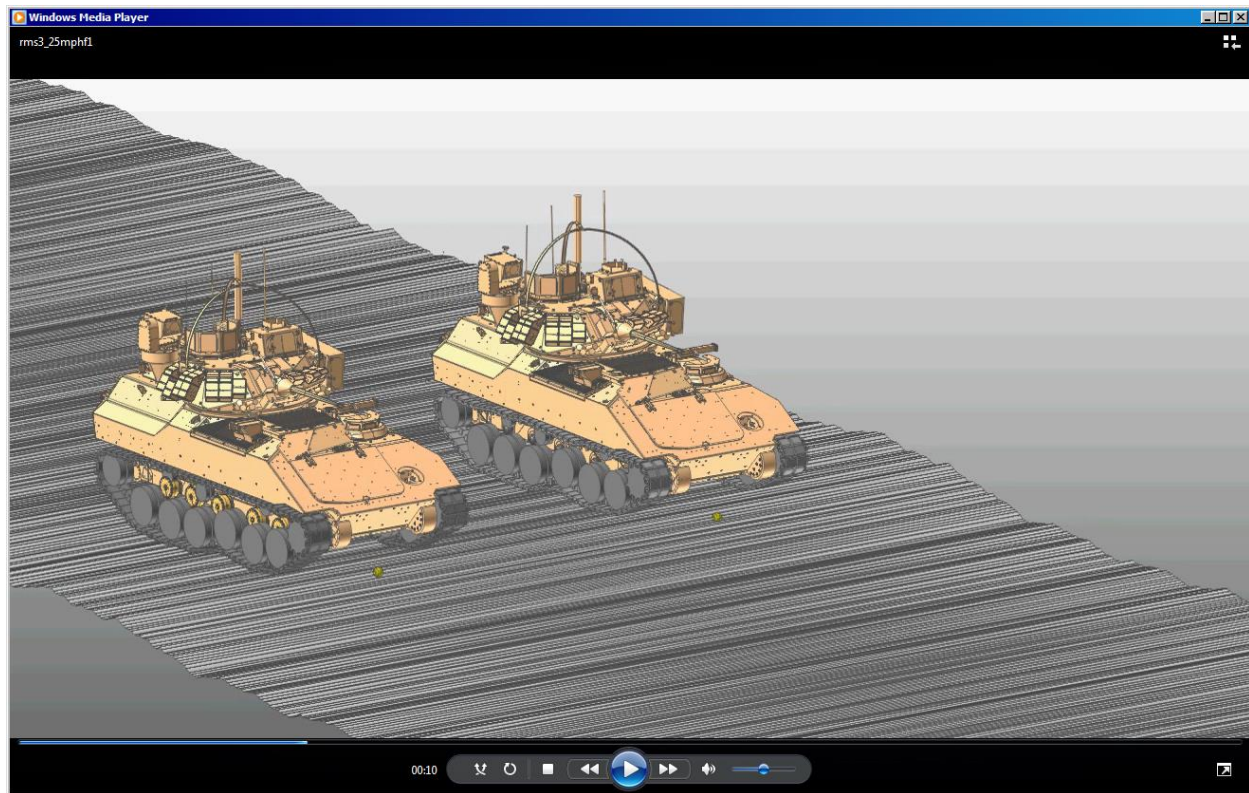


Figure 4 – “rms3” animation (25 MPH)

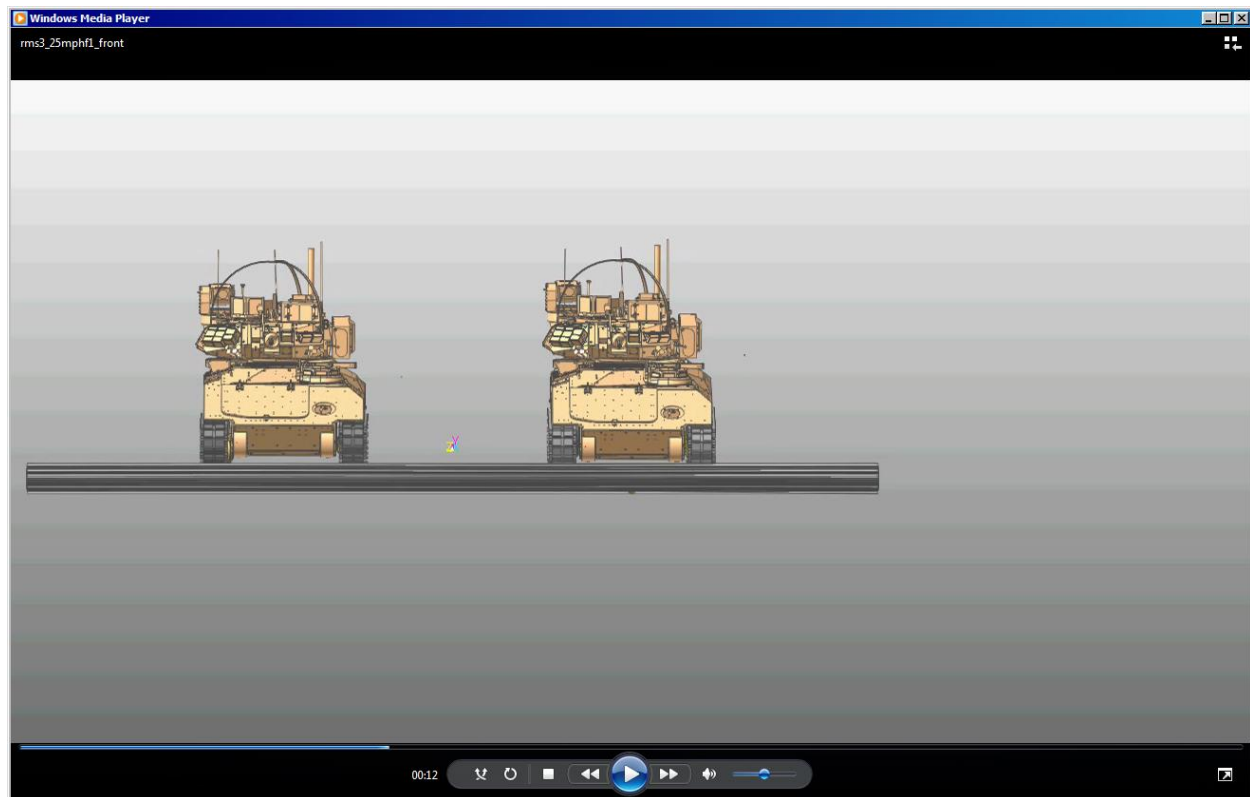


Figure 5 – “rms3front” animation (25 MPH)

In addition to increasing the speed to show pronounced performance differences, the standard following camera of the “base” hull CG was modified so there would be no vertical movement. This was accomplished by adding a dummy body (centered between the two vehicles) that the camera would follow. This dummy body was connected to the ground through an inplane and orientation joint, and then to the “base” hull CG marker with an inline joint. This combination gave the same result as the standard following camera without vertical movement with the “base” hull CG---which helped highlight performance differences in the animation.

REFERENCE

- [1] RecurDyn (<https://functionbay.com/en>, <https://functionbay.com/documentation/onlinehelp/default.htm#!Documents/help.htm>)

TN 18-0405-01

AUTOMATING THE CREATION OF THE WELD DEFINITION FILE IN FE-SAFE

April 5, 2018

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC)
Detroit Arsenal
Systems Integration & Engineering
ATTN: RDTA-SIE/MS157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

A request was made for help in reducing the manual effort it takes to create the weld definition file for Verity (in fe-safe [1]) for a vehicle hull structure.

PROBLEM SOLUTION:

From [1] "1.1 About fe-safe" we have:

fe-safe is a powerful, comprehensive and easy-to-use suite of fatigue analysis software for Finite Element models. It is used alongside commercial FEA software to calculate:

- where fatigue cracks will occur;
- when fatigue cracks will initiate;
- the factors of safety on working stresses (for rapid optimisation);
- the probability of survival at different service lives (the 'warranty claim' curve).

Results are presented as contour plots which can be plotted using standard FE viewers.

and also "1.2 About Verity in fe-safe":

Verity in fe-safe is an add-on module for use with fe-safe fatigue analysis software, that computes equivalent structural stresses based on the Battelle Structural Stress Method, and uses these stresses to calculate fatigue damage and corresponding fatigue lives. Users of Verity in fe-safe are assumed to have a working knowledge of fe-safe, including such techniques as configuring a fatigue analysis and setting properties for different parts of the model, defining the fatigue loading, running an analysis and exporting fatigue results. The use and application of fe-safe is described in the fe-safe User guide, which should be referred to alongside the Verity in fe-safe manual.

with specific reference to "2.1.2 Defining the Welds" and "11.5 Define the Weld" in [1].

A process was created in Linux [5], using sed [2] and bash [4], to efficiently generate multiple weld templates from ABAQUS input file element and node sets for use in predicting fatigue life in fe-safe [1]. First, the portion of the weld definition file that will be repeated for multiple data sets should be extracted from the full file format and field replacement values defined between double equal signs ("==" or some other uncommon set of characters) as shown in Appendix A.1 - "weld_tmpl.wdf". Similarly this should be repeated for the group fields as shown in Appendix A.1 - "weld_tgrp.wdf". Second, a comma delimited file of the weld replacement data must be manually created as shown in Appendix A.1 - "weld_data.csv"

using spreadsheet software [6] or a text editor. Third, the node and element sets from the ABAQUS input file (.inp) must be provided as shown in Appendix A.1 - “sets.inp”.

The automation starts with the process script shown in Appendix A.2 – “go”. The first action is to effectively add a trailing comma to the node and element set name data. The second action combines the named node and element sets each into one line as below:

```
N5000_5004=680795,680794,680793,680792,680791,680790,680789,680788,680787,680786,680785,680784,680783,680782,680781,680780
E5000_5004A=776090,776089,776084,776083,776070,776069,776064,776063,775997,775996,775995,775974,775973,775968,775967
E5000_5004B=676771,676770,676767,676766,676755,676754,676751,676750,676637,676636,676634,676595,676594,676591,676590
.
```

From [2], a summary of the sed commands used in Appendix A.2 – “1-set” to accomplish this is included below:

-n
--quiet
--silent

By default, sed prints out the pattern space at the end of each cycle through the script (see How sed works). These options disable this automatic printing, and sed only produces output when explicitly told to via the p command.

-i[SUFFIX]
--in-place[=SUFFIX]

This option specifies that files are to be edited in-place. GNU sed does this by creating a temporary file and sending output to this file rather than to the standard output.1.

This option implies -s.

When the end of the file is reached, the temporary file is renamed to the output file’s original name. The extension, if supplied, is used to modify the name of the old file before renaming the temporary file, thereby making a backup copy2).

[addr]X
[addr]{ X ; X ; X }
/regexp/X
/regexp/{ X ; X ; X }

Addresses and regular expressions can be used as an if/then conditional: If [addr] matches the current pattern space, execute the command(s). For example: The command **/^#/d** means: if the current pattern matches the regular expression **^#** (a line starting with a hash), then execute the d command: delete the line without printing it, and restart the program cycle immediately.

!

Address negation, use an address with negation (!) to perform substitution only on desired lines.

Matches a sequence of zero or more instances of matches for the preceding regular expression, which must be an ordinary character, a special character preceded by \, a ., a grouped regexp

(see below), or a bracket expression. As a GNU extension, a postfix regular expression can also be followed by `*`; for example, `a**` is equivalent to `a*`. POSIX 1003.1-2001 says that `*` stands for itself when it appears at the start of a regular expression or subexpression, but many nonGNU implementations do not support this and portable scripts should instead use `\*` in these contexts.

^

Matches the null string at beginning of the pattern space, i.e. what appears after the circumflex must appear at the beginning of the pattern space.

In most scripts, pattern space is initialized to the content of each line (see How sed works). So, it is a useful simplification to think of `^#include` as matching only lines where `'#include'` is the first thing on line—if there are spaces before, for example, the match fails. This simplification is valid as long as the original content of pattern space is not modified, for example with an `s` command.

`^` acts as a special character only at the beginning of the regular expression or subexpression (that is, after `\(` or `\|`). Portable scripts should avoid `^` at the beginning of a subexpression, though, as POSIX allows implementations that treat `^` as an ordinary character in that context.

\$

It is the same as `^`, but refers to end of pattern space. `$` also acts as a special character only at the end of the regular expression or subexpression (that is, before `\)` or `\|`), and its use at the end of a subexpression is not portable. `H=` Append a newline to the contents of the hold space, and then append the contents of the pattern space to that of the hold space.

\(regexp\)

Groups the inner regexp as a whole, this is used to:

Apply postfix operators, like `\(abcd\)*`: this will search for zero or more whole sequences of `'abcd'`, while `abcd*` would search for `'abc'` followed by zero or more occurrences of `'d'`. Note that support for `\(abcd\)*` is required by POSIX 1003.1-2001, but many non-GNU implementations do not support it and hence it is not universally portable. Use back references (see below).

\digit

Matches the digit-th `\(...\)` parenthesized subexpression in the regular expression. This is called a back reference. Subexpressions are implicitly numbered by counting occurrences of `\(` left-to-right.

x

Exchange the contents of the hold and pattern spaces

p

Print out the pattern space (to the standard output). This command is usually only used in conjunction with the `-n` command-line option.

d

Delete the pattern space; immediately start next cycle.

\U

Turn the replacement to uppercase until a \L or \E is found,

The third action, uses the defined weld definition templates and replacement values with the defined node and element sets to generate template data for all the welds provided. The generated output is combined with the original weld definition file (XML) header data to make a complete weld definition file.

Appendix A.3 is included to show how this could be accomplished using gawk [3] instead of sed.

REFERENCE

- [1] Verity in fe-safe User Manual (https://help.3ds.com/2017/English/DSSIMULIA_Established/FesafePdf/FesafeVerity.pdf?contextscope=all&id=c1fd3777556544d8a688b96832ec8750). Requires account at <https://help.3ds.com>.
- [2] GNU sed (www.gnu.org/software/sed/) (`sed -version`: GNU sed version 4.1.5)
- [3] GNU gawk (www.gnu.org/software/gawk/) (`gawk -version`: GNU Awk 3.1.8)
- [4] GNU bash (www.gnu.org/software/bash/) (`bash -version`: GNU bash, version 3.2.51(1)-release (x86_64-suse-linux-gnu))
- [5] SUSE (www.suse.com) (`cat /etc/SuSE-release`: SUSE Linux Enterprise Server 11 (x86_64) VERSION = 11 PATCHLEVEL = 3)
- [6] Microsoft Excel (<https://products.office.com/en-us/excel>)

APPENDIX A.1 – Weld Template/Data Files**weld_tmpl.wdf**

```

<weld type="line" name=="NAME==">
  <parameter id="plate thickness">
    <!--Plate Thickness-->
    <value>==TH==</value>
  </parameter>
  <parameter id="crack length">
    <!--Crack Length-->
    <value>==CL==</value>
  </parameter>
  <parameter id="reference length">
    <!--Reference Length-->
    <value>==REFL==</value>
  </parameter>
  <parameter id="weld symmetry">
    <!--Weld symmetry-->
    <value>None</value>
  </parameter>
  <parameter id="rotation into weld space">
    <!--Rotate LCS After Solve-->
    <value>Before</value>
  </parameter>
  <parameter id="smoothing iterations">
    <!--Number of smoothing iterations-->
    <value>0</value>
  </parameter>
  <parameter id="ir function">
    <!--I(r) Function-->
    <value>Structural Joints / Displacement Control</value>
  </parameter>
  <parameter id="element type">
    <!--Element type-->
    <value>Shells</value>
  </parameter>
  <parameter id="weld toe elements">
    <!--Weld reference elements-->
    <value>PART-1-1_==ELNAME==</value>
  </parameter>
  <parameter id="start node">
    <!--Start Node-->
    <value>==ND==</value>
  </parameter>
  <parameter id="start element">
    <!--Start Element-->
    <value>==EL==</value>
  </parameter>
  <parameter id="shell layer">
    <!--Shell layer-->
    <value>Both</value>
  </parameter>
  <parameter id="weld type">
    <!--Weld type-->
    <value>Open</value>
  </parameter>
  <parameter id="mid surface offset">
    <!--Mid surface offset-->
    <value>0</value>
  </parameter>
  <parameter id="rotation of weld space">
    <!--Rotate weld line coord system (deg.)-->
    <value>0</value>
  </parameter>
  <parameter id="failure mode">
    <!--Failure Mode-->
    <value>Unspecified Failure Mode</value>
  </parameter>
  <group type="nodal" id="weld nodes" name="PART-1-1_==NNAME==">==NNUMBERS==</group>
</weld>

```

weld_tgrp.wdf

```
<group type="elemental" name="PART-1-1_==ELNAME==">==ELNUMBERS==</group>
```

weld_data.csv

```
==NAME==,==TH==,==CL==,==REFL==,==ND==,==EL==,==NNAME==,==ELNAME==
5000_5004A,28.575,34.89,87.23,680780,776069,N5000_5004,E5000_5004A
5000_5004B,28.575,34.89,87.23,680780,676754,N5000_5004,E5000_5004B
5000_5004_2A,28.575,34.89,87.23,679730,777729,N5000_5004_2,E5000_5004_2A
5000_5004_2B,28.575,34.89,87.23,679730,679400,N5000_5004_2,E5000_5004_2B
5000_5005A,28.575,62.98,157.46,680765,676684,N5000_5005,E5000_5005A
.
.
.
```

sets.inp

```
*NSET, NSET=n5000_5004
    680795,    680794,    680793,    680792,    680791,    680790,    680789,    680788,
    680787,    680786,    680785,    680784,    680783,    680782,    680781,    680780
*ELSET, ELSET=e5000_5004a
    776090,    776089,    776084,    776083,    776070,    776069,    776064,    776063,
    775997,    775996,    775995,    775974,    775973,    775968,    775967
*ELSET, ELSET=e5000_5004b
    676771,    676770,    676767,    676766,    676755,    676754,    676751,    676750,
.
.
.
```

APPENDIX A.2 – Process Scripts**go**

```
#!/bin/bash

. 0-set ./data/sets.inp > ./data/weld_set.txt

. 1-set ./data/weld_set.txt

# tmp1, data, set, grp_tmp1
. 2-tmpl ./data/weld_tmpl.wdf ./data/weld_data.csv ./data/weld_set.txt ./data/weld_tgrp.wdf
```

0-set

```
#!/bin/bash

sed -e '/^*/{s/[\r\n\t ]//g; s/$/,/}' $1
```

1-set

```
#!/bin/bash

sed -n -i -e '/^*/{x;
    s/[\r\n\t ]//g;
    s/^*.*=\[([^\,]*)\),\[([^\,]*)\]/\U\1=\2/;
    p; d; x}' -e '/^*/!H' -e '${x;
    s/[\r\n\t ]//g;
    s/^*.*=\[([^\,]*)\),\[([^\,]*)\]/\U\1=\2/; p}' $1
```

2-tmpl

```

#!/bin/bash

clear

# input data
ftmp=$1 #'weld_tmpl.txt'          # template
fin=$2  #'weld_data.csv' # data (assumes comma delimited format .csv)
fset=$3 #'weld_set.txt'
fgtmp=$4 #'weld_tgrp.txt'

# temporary files
f1="_tmp"          # original template
f2="_tmp0"         # replaced template
f3="_tmpa"         # appended templates
rm $f3

f1g="_gtmp"        # original template
f3g="_gtmpa"       # appended templates
rm $f3g

echo "Template: $ftmp"
echo "   Data: $fin"

# get header/search data
h=$(head -n 1 $fin)
hl=$(( ${#h} - 1 ))
h=${h:0:$hl}      # correct for end-of-line control characters

# split into array
IFS=' '
hc=(echo "$h" )
unset IFS
hl=$(( ${#hc[@]} - 1 ))

# get replacement data
d=$(cat $fin)
dl=$(( ${#d[@]} - 1 ))

for(( i=1; i <= $dl; i++ )); do
    echo "Processing: $i of $dl"
    cl=$(( ${#d[i]} - 1 ))
    d[i]=${d[i]:0:$cl}    # correct for end-of-line control characters

    # split into array
    IFS=' '
    cells=(echo "${d[i]}" )
    unset IFS

    # create replacement sed commands
    strsed=""
    for(( j=0; j <= ${#hl}; j++ )); do
        #===CHANGE

        if [ "${hc[${j}]}" == "ELNAME" ]; then
            eval=$( sed -n -e "s/^\${cells[${j}]}=\\(.*\\)/\1/p" $fset )
            strsed=$( echo "${strsed} -e s/==ELNUMBERS==/${eval}/g" )
        fi
        if [ "${hc[${j}]}" == "NNAME" ]; then
            nval=$( sed -n -e "s/^\${cells[${j}]}=\\(.*\\)/\1/p" $fset )
            strsed=$( echo "${strsed} -e s/==NNUMBERS==/${nval}/g" )
        fi
        #===
        strsed=$( echo "${strsed} -e s/${hc[${j}]}/${cells[${j}]} /g" )
    done

    # each row start new template, replace all values, concatenate
    cp $ftmp $f1
    cat $f1 | sed $strsed > $f2
    cat $f2 >> $f3

    #grp
    cp $fgtmp $f1g
    sed -i $strsed $f1g
    cat $f1g >> $f3g
done

```

APPENDIX A.3 – Example sed/gawk Comparison

```
#!/bin/bash

#=== for windows, add \ before " (\"), add \ before \ (\\t for \t)

#"===SED===
sed -n -e '/^*/{x;
    s/[\r\n\t ]//g;
    s/^*.*=([^\,]*)\, \(.*$\) /\1=\2/;
    p; d; x}' -e '/^*/!H' -e '${x;
    s/[\r\n\t ]//g;
    s/^*.*=([^\,]*)\, \(.*$\) /\1=\2/; p}' $1 > o_sed

#"===GAWK===
gawk 'BEGIN {
    c="";
}
/^*/ {
    c=gensub(/[\r\n\t ]/ , "", "G", c);
    c=gensub(/^*.*=([^\,]*)\, \(.*$\) /\1=\2/ , "\\1=\2", "G", c);
    print c;
    c=$0;
}
(substr($0,1,1) != "") {
    c=c$0;
}
END {
    c=gensub(/[\r\n\t ]/ , "", "G", c);
    c=gensub(/^*.*=([^\,]*)\, \(.*$\) /\1=\2/ , "\\1=\2", "G", c);
    print c;
}' $1 > o_gawk
```

TN 18-1129-01

CREATING ABAQUS JOINT/UJOINT CONNECTORS IN HYPERMESH

November 29, 2018

W. Bylsma

U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC)
Detroit Arsenal
Systems Integration & Engineering
ATTN: RDTA-SIE/MS157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

Structural analysis of the device in Figure 1 requires connections between the device and its mounting hardware. Consideration must also be given to the proper degrees of freedom (DOF) when an idealization of the joint is made for modeling purposes. How can the clevis connection be modeled in HyperMesh [3] for an ABAQUS [4] solver deck.

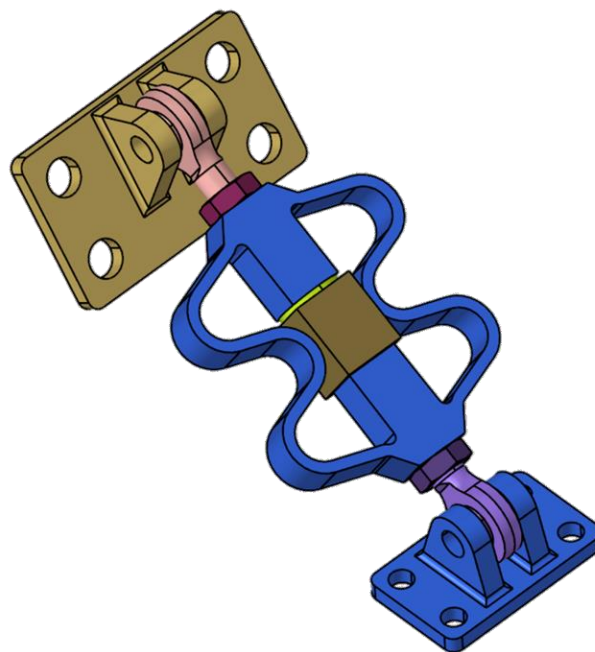


Figure 1 – Energy Absorber and Mounting Hardware

PROBLEM SOLUTION:

The definition of a clevis can be reviewed at [1] and examples (with spherical bearing) from [2] are shown in Figure 2. While the focus here is not on meshing, the mesh for the mounting brackets for this device were created by removing

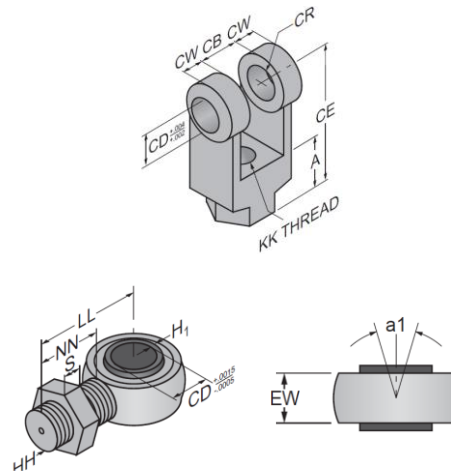


Figure 2 – Cylinder Mounting Accessories [2]

the edge fillets (shown in Figure 1) and using mid-surfaces on the brackets to create shells (where the thickness is defined in the shell property). Simplification was made to the device to allow a reasonable representation as shown in Figure 3. Solid Map meshing was used on this device representation. Notice that the geometry for the clevis was removed as that will be the focus of the connector modeling.

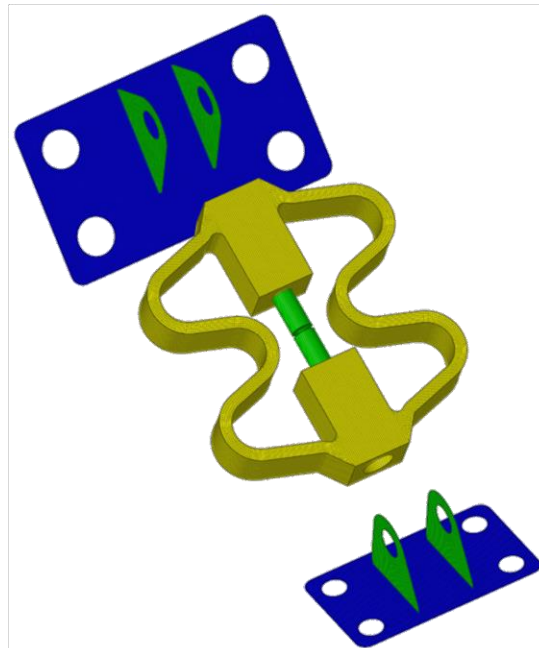


Figure 3 – Meshed Components

Connector elements [6] are used in ABAQUS to represent kinematic constraints [5]. Figure 4 lists the different connector types. Each type represents a different set of conditions on the DOF between nodes, where some require node orientation to be defined.



Connection-type library

This section provides a reference to the connection types available in Abaqus.

In this section:

- ☐ ACCELEROMETER
- ☐ ALIGN
- ☐ AXIAL
- ☐ BEAM
- ☐ BUSHING
- ☐ CARDAN
- ☐ CARTESIAN
- ☐ CONSTANT VELOCITY
- ☐ CVJOINT
- ☐ CYLINDRICAL
- ☐ EULER
- ☐ FLEXION-TORSION
- ☐ FLOW-CONVERTER
- ☐ HINGE
- ☐ JOIN
- ☐ LINK
- ☐ PLANAR
- ☐ PROJECTION CARTESIAN
- ☐ PROJECTION FLEXION-TORSION
- ☐ RADIAL-THRUST
- ☐ RETRACTOR
- ☐ REVOLUTE
- ☐ ROTATION
- ☐ ROTATION-ACCELEROMETER
- ☐ SLIDE-PLANE
- ☐ SLIPRING
- ☐ SLOT
- ☐ TRANSLATOR
- ☐ UJOINT
- ☐ UNIVERSAL
- ☐ WELD

Figure 4 – Connector Element Types [7]

For the clevis connections to the mounting brackets in Figure 3, a JOIN connector element [8] could be used at each end that makes the position of two nodes the same. However, for this connector type there is no rotational kinematic constraint. This would connect the device to the brackets but would allow it to spin freely from the connection points. This idealization of the joint causes unrealistic behavior and potential difficulty for the solver to find a solution. To address this a UJOINT (JOIN + UNIVERSAL [9]) connector element [10] is used at one end of the device connection to the bracket that will allow proper movement but restrict the unconstrained rotation of the device.

NOTE

The different terminology used for HyperMesh and ABAQUS can be confusing. For example, reference to rigid 1D elements in HyperMesh means a BEAM type multi-point constraint [11] in ABAQUS.

Rigid Elements

Rigid elements will be used to create nodes connecting the meshed elements to the connector elements for the joint connections.

NOTE

It is best to create separate components for the rigid elements that reference nodes a and b (the elements connected to the bracket and device, respectively). This way you can more easily select which elements are displayed. Make sure the proper component is the CURRENT component so what is being created will be stored in it. Make the names of the components descriptive (e.g. Jna for JOINT node a, Jnb for JOINT node b, etc.) to help in identifying what they contain.

1) use 1D\rigids to create a rigid element of type BEAM [11] with all DOF constrained using “loops” for the dependent nodes and “calculate node” for the independent node as shown in Figure 5. The calculated node will be considered node a. Figure 6 shows the ABAQUS element types that can be used in HyperMesh to define 1D rigid elements. Repeat this step for the other bracket.

2) use Tool\Translate to translate the calculated node 0.1 units using the face of the device as shown in Figure 7. Make sure to “duplicate” the selected node by selecting nodes again and duplicate. This will be considered node b. See Figure 8 for details. Repeat this step for the other bracket.

NOTE

From Preferences\Graphics the coincident picking option can be selected. In practice it is easier to translate the second node slightly as done in this step.

3) use 1D\rigids to create a rigid element of type BEAM with all DOF constrained using “by path” for the dependent nodes and “select node” for the independent node, connected to the device, as shown in Figure 9. The independent node you select here is the one translated in step 2 (node b). This step requires zooming in and out to identify the proper nodes to select. Repeat this step for the other side of the device.

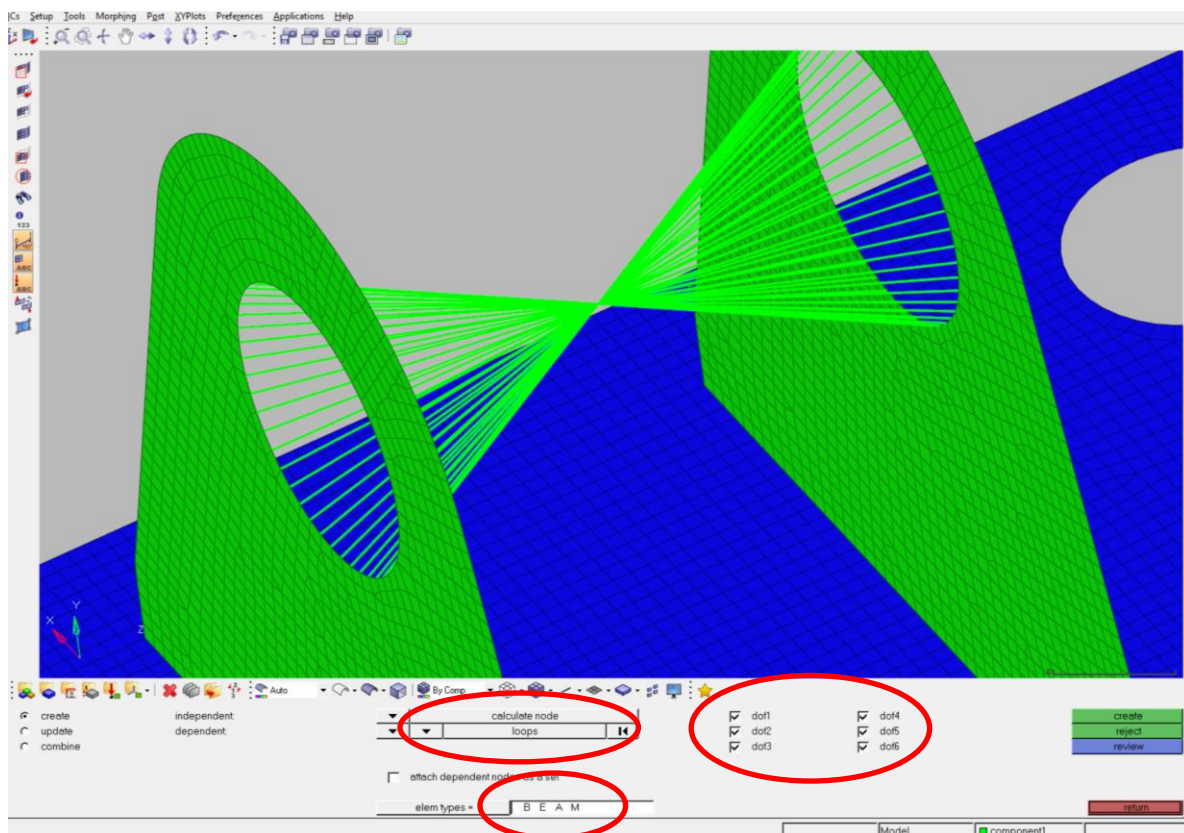


Figure 5 – Rigid Elements - Bracket

MPCs for connections and joints

BEAM

Provide a rigid beam between two nodes to constrain the displacement and rotation at the first node to the displacement and rotation at the second node, corresponding to the presence of a rigid beam between the two nodes.

CYCLSYM^(S)

Constrain nodes to impose cyclic symmetry in a model.

ELBOW^(S)

Constrain two nodes of ELBOW31 or ELBOW32 elements together, where the cross-sectional direction, a_2 , changes (see [Pipes and pipebends with deforming cross-sections: elbow elements](#)).

LINK

Provide a pinned rigid link between two nodes to keep the distance between the two nodes constant. The displacements of the first node are modified to enforce this constraint. The rotations at the nodes, if they exist, are not involved in this constraint.

PIN

Provide a pinned joint between two nodes. This MPC makes the displacements equal but leaves the rotations, if they exist, independent of each other.

REVOLUTE^(S)

Provide a revolute joint.

SLIDER

Keep a node on a straight line defined by two other nodes, but allow the possibility of moving along the line and allow the line to change length.

TIE

Make all active degrees of freedom equal at two nodes.

UNIVERSAL^(S)

Provide a universal joint.

V LOCAL^(S)

Allow the velocity at the constrained node to be expressed in terms of velocity components at the third node defined in a local, body axis system. These local velocity components can be constrained, thus providing prescribed velocity boundary conditions in a rotating, body axis system.

See [About connectors](#) for element-based versions of several of these MPCs for connections and joints.

Figure 6 – Multi-Point Constraint Types [11]

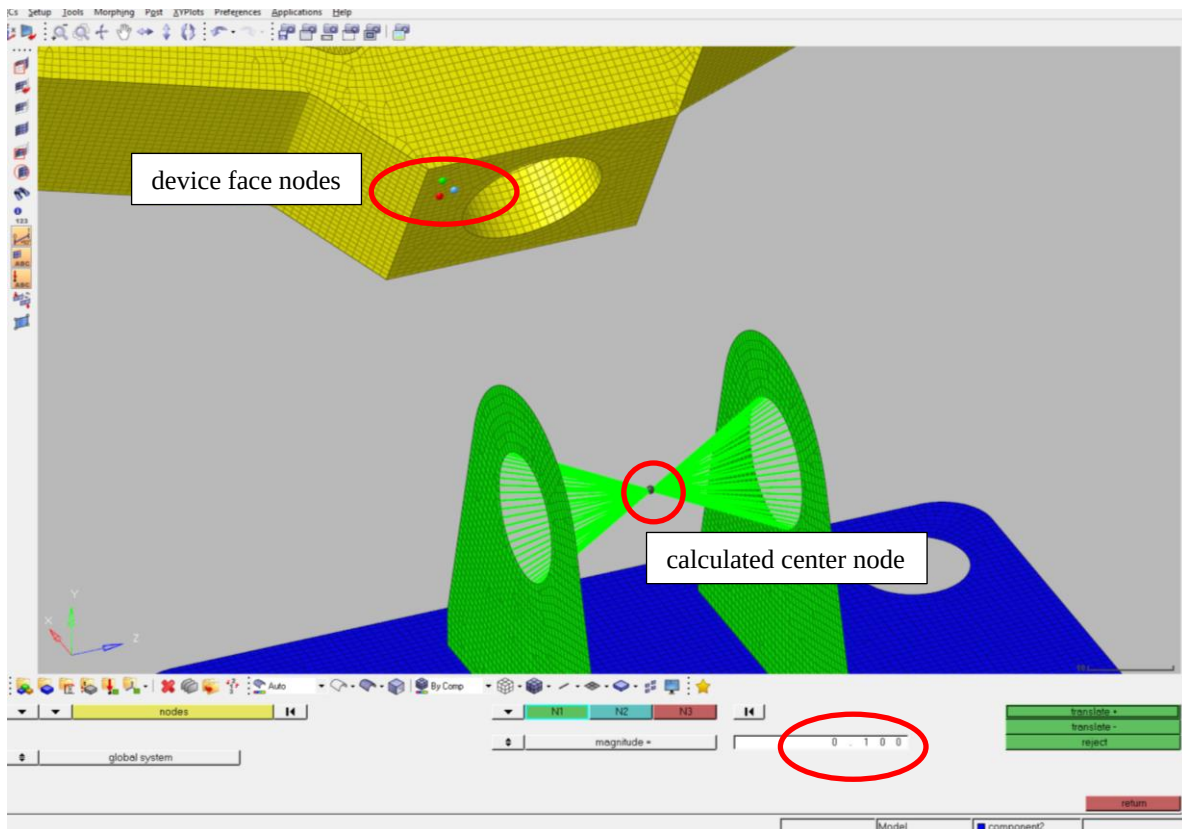


Figure 7 – Translate Node

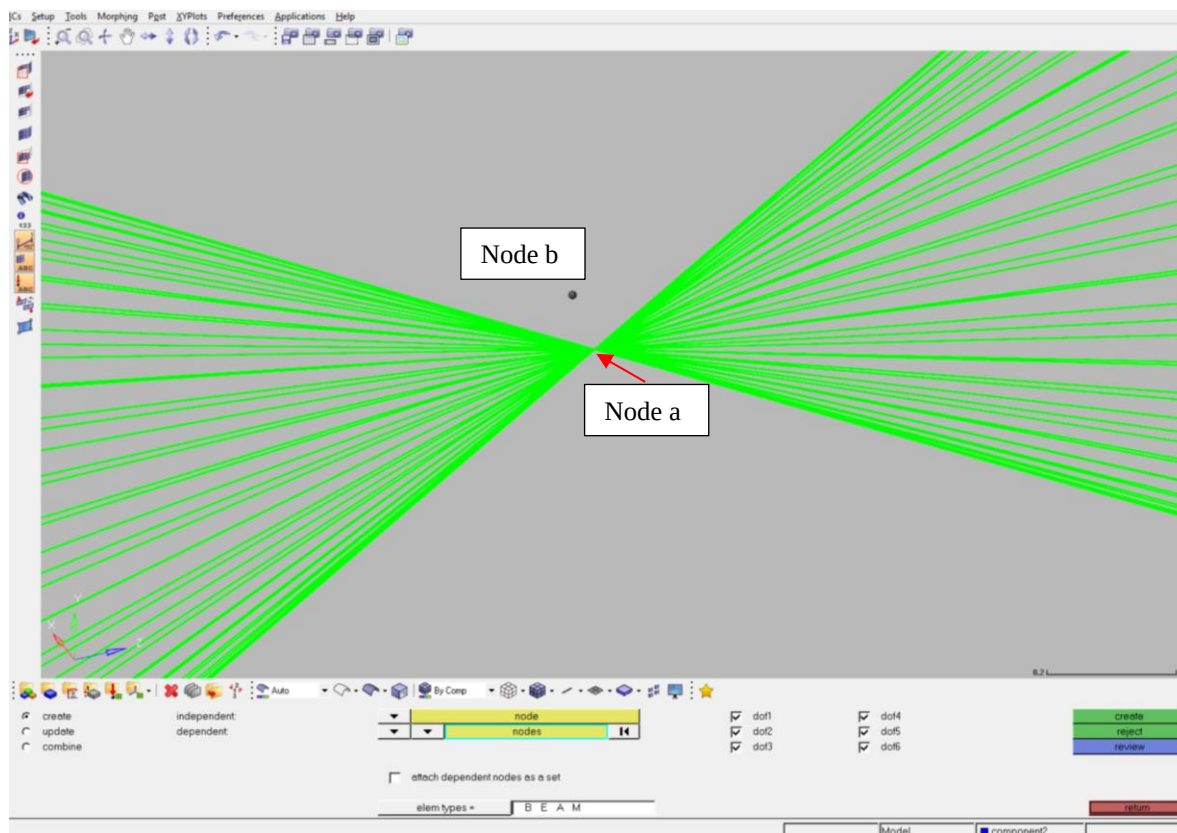


Figure 8 – Node Definitions

Systems

Systems in HyperMesh are used to define orientations [12] in ABAQUS. The upper joint in Figure 3 will be a UJOINT connector element and the lower joint will be a JOIN connector element. The UJOINT connector element requires a system to define orientation.

NOTE

It is best to create separate components for the systems that reference nodes a and b (the elements connected to the bracket and device, respectively). This way you can more easily select which systems are displayed. When creating them make sure the proper system is the CURRENT system so what is being created will be stored in it. Make the names of the systems descriptive (e.g. Sys_na for System Node a, Sys_nb for System Node b, etc.) to help in identifying what they contain.

1) use Geom/distance with the three node option and then “circle center” option to create center nodes in the middle of the bracket holes and device attachment location, as shown in Figure 10a, at each end of the device. These nodes will be used to define the system orientation (turn the rigid components display off, as in Figure 10b, so that selection of the nodes is easier).

2) use 1D\systems with the “create by node reference”, “x-axis”, and “xy-plane” options after selecting node a as the origin node, either one of the bracket center nodes, and the device center node to create a system for node a. See Figure 10b. After changing the current system, repeat these steps using node b instead of node a.

3) use Card Edit with the “systs” type to select and edit each system created. This requires zooming in to make sure the appropriate system is selected since they are very close together as shown in Figure 11a. Make sure “ORIENTATION” is selected and the name is edited appropriately as highlighted in Figure 11b.

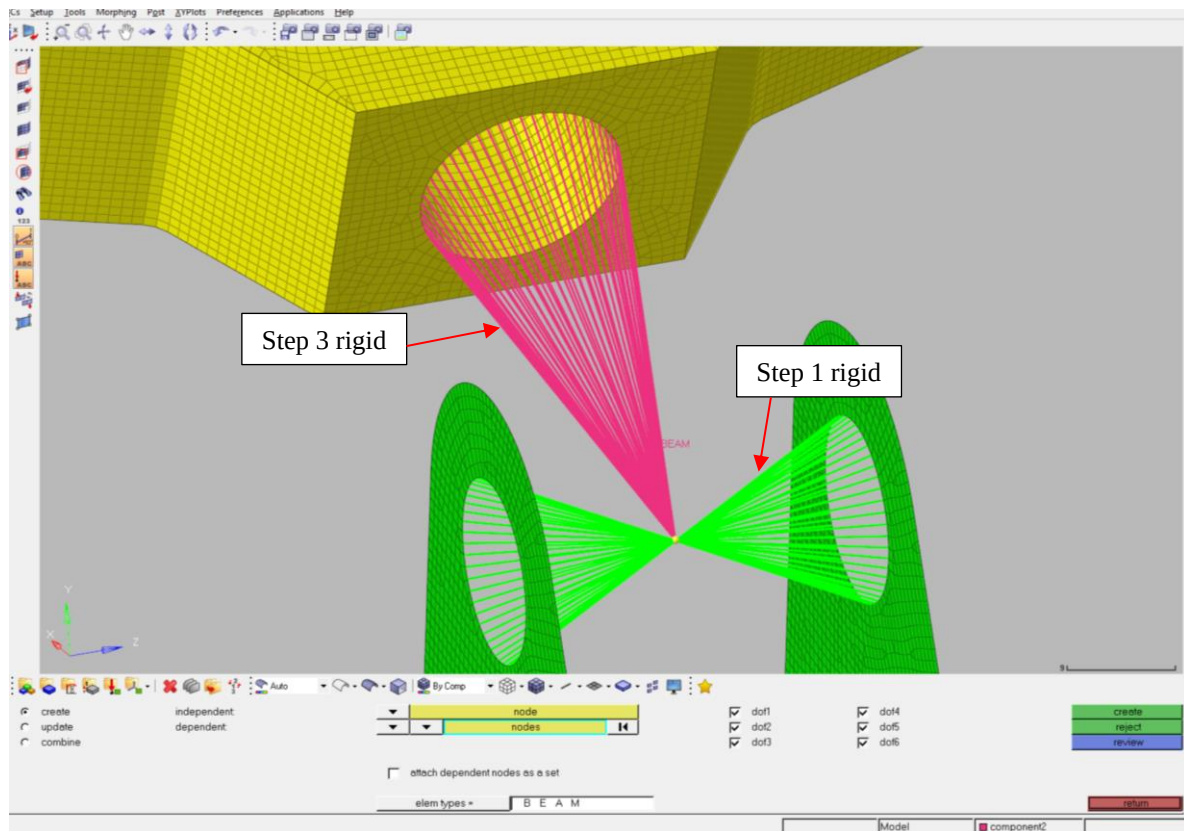


Figure 9 – Rigid Element - Device

Rod Elements

Rod elements in HyperMesh are used to create connector elements in ABAQUS. Each rod element defines the two nodes used in the connector element. Properties assigned to each rod element then define the connector characteristics. For each desired connector element an associated rod element is created. Because connector properties are assigned to individual components, a separate component must be defined for each connector element that has different properties. For UJOINT type connector elements this includes location---as the systems that define the orientations for this connector are location dependent (e.g. a different component for each location would have to be created, UJ_Left1, UJ_Left2, UJ_Right1, etc.). Because the JOIN type connector elements is not location dependent all connectors of that type could be stored in the same component.

NOTE

It is best to create separate components for the rod elements (that reference nodes a and b at either end of the device) to distinguish between which is defined as a JOIN or UJOINT connector. As previously discussed, these two types of connectors are needed to restrict the DOF and prevent the device from spinning freely.

- 1) use 1Drods with element type CONN3D2 [6] to create a connector element for each joint. Select node a first and then node b for consistency. When creating rods make sure the proper component is the CURRENT component so what is being created will be stored in it. See Figure 12.

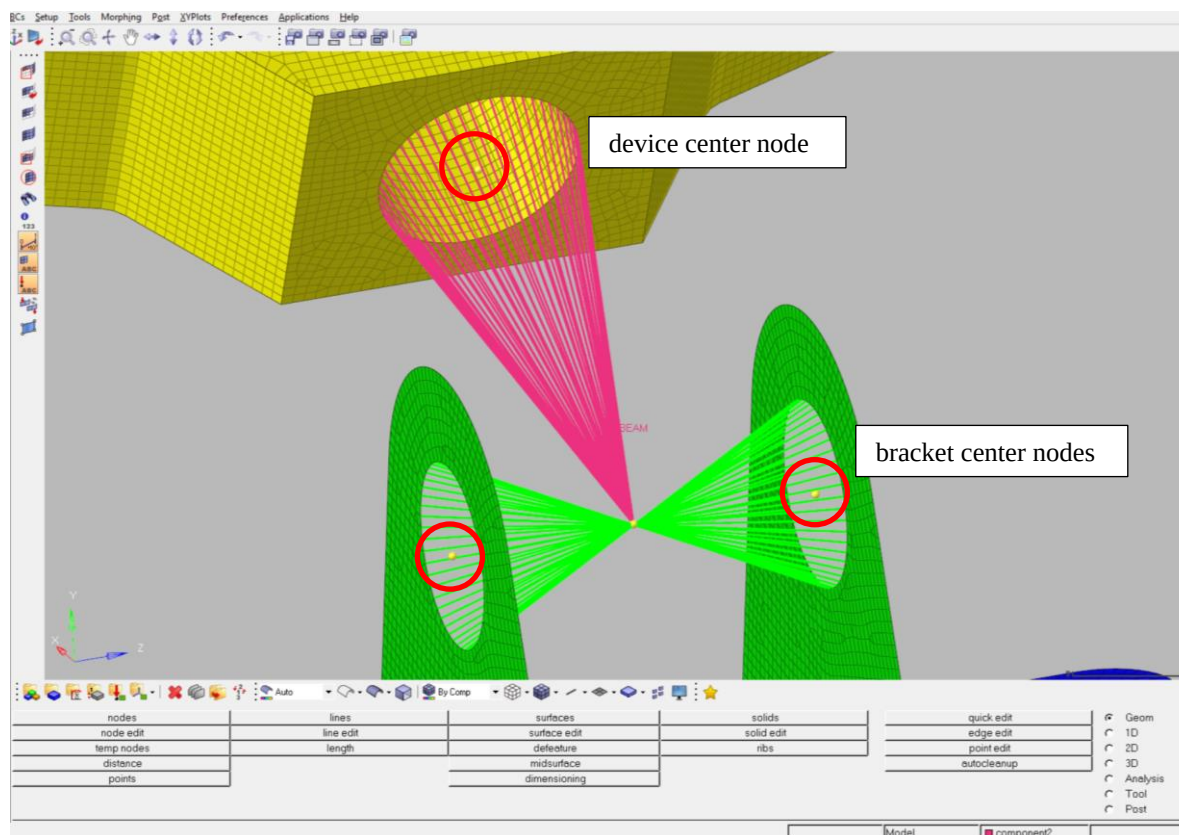


Figure 10a – Nodes for System definition

Properties

Properties in HyperMesh are used to define the type of connector element. The available types are listed in Figure 4. This is accomplished by assigning a property to the component in which the connector resides. The card image must be CONNECTORSECTION as shown in Figure 13 for the JOIN type connector element. For UJOINT type connector elements, orientations are defined for nodes a and b as shown in Figure 14.

NOTE

It is possible to use Card Edit to enter/edit the data as it would appear in the ABAQUS solver input file.

For other connector types additional behaviors are possible by including a material in the property as shown in Figure 15 for the BUSHING type connector element [13]. A brief description is included under the Materials section.

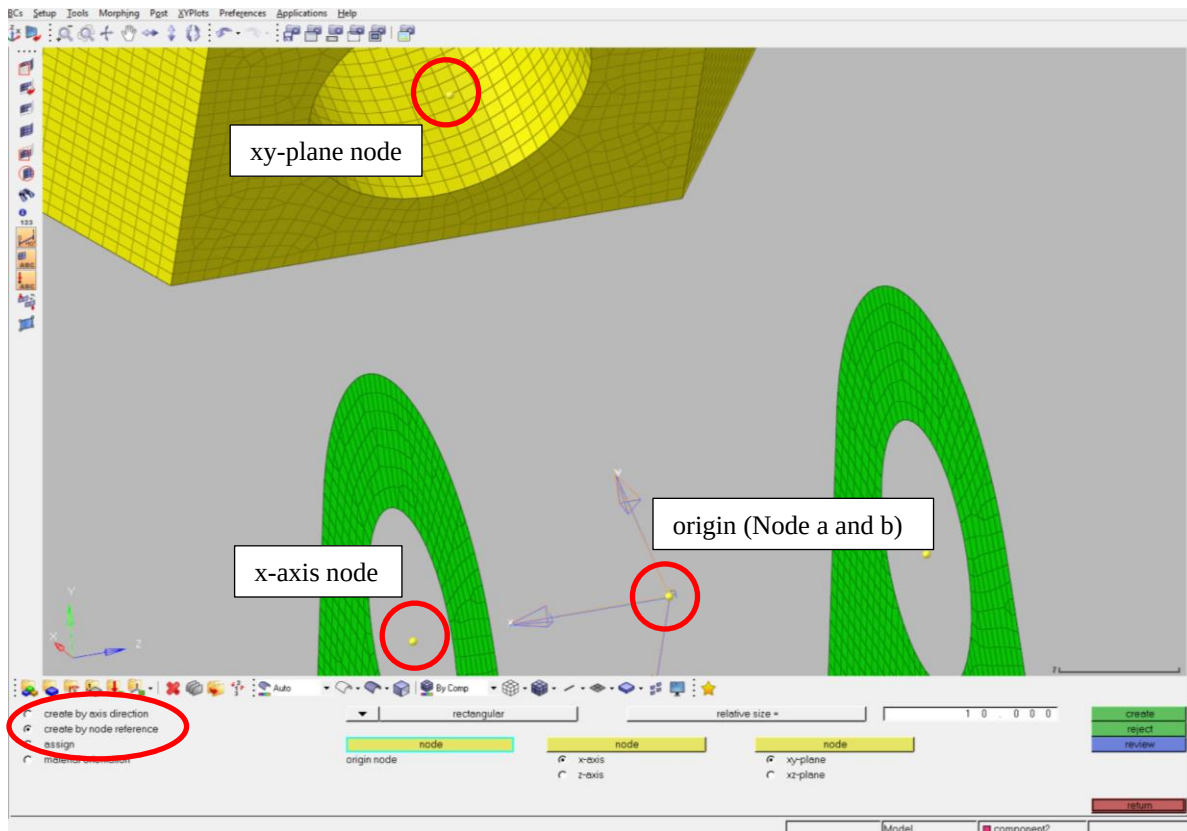


Figure 10b – Systems for Node a and Node b

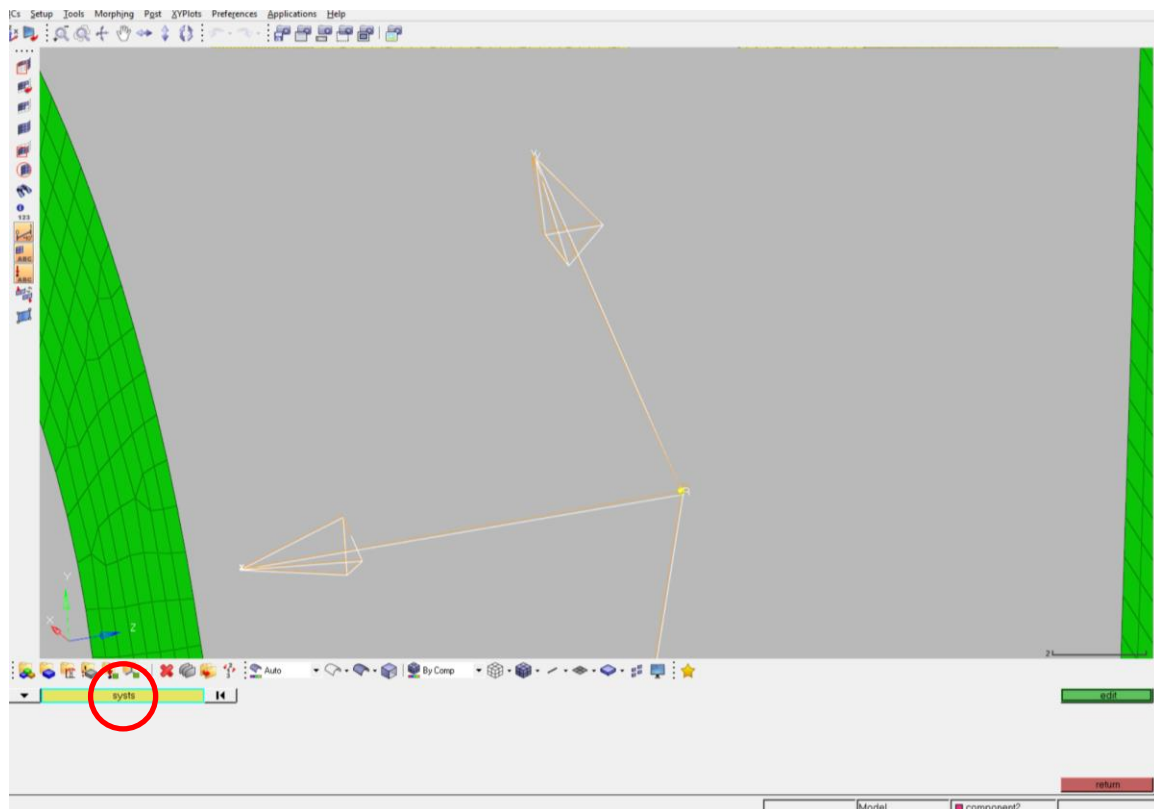


Figure 11a – System for Node a Setup

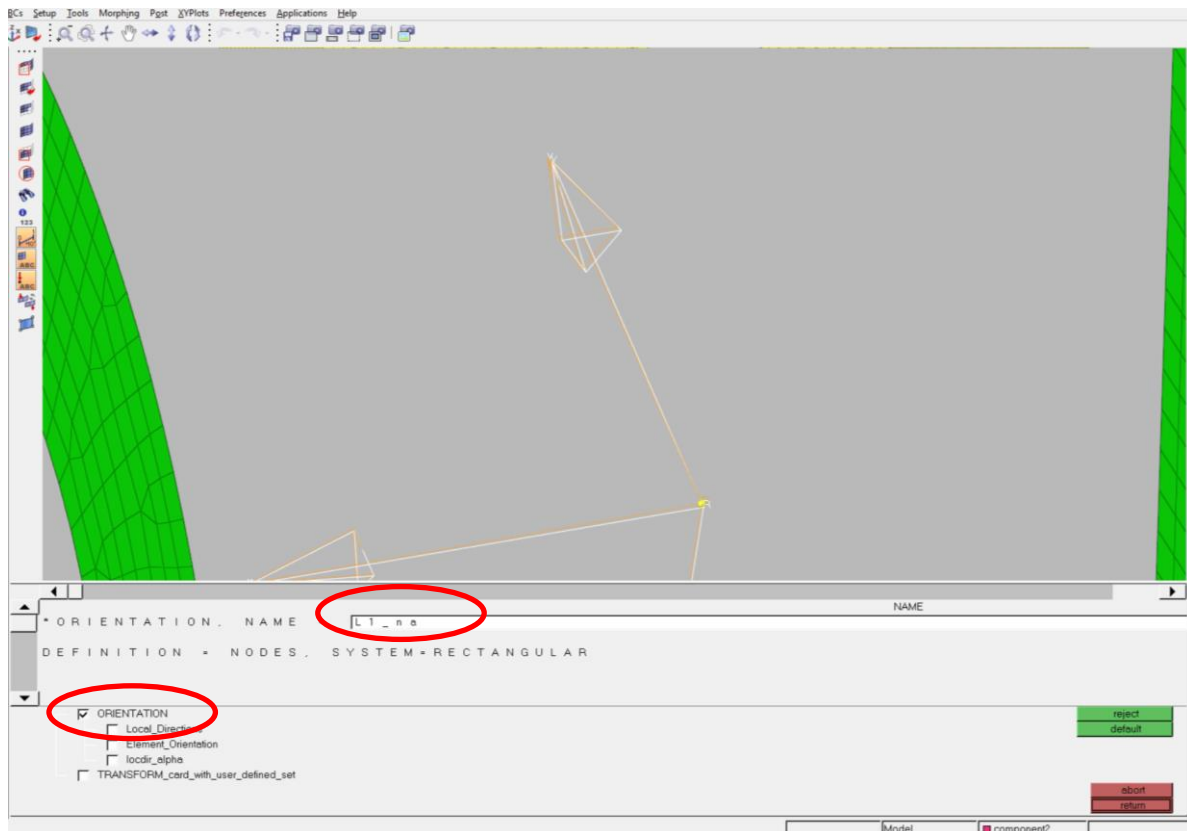


Figure 11b – System for Node a Setup

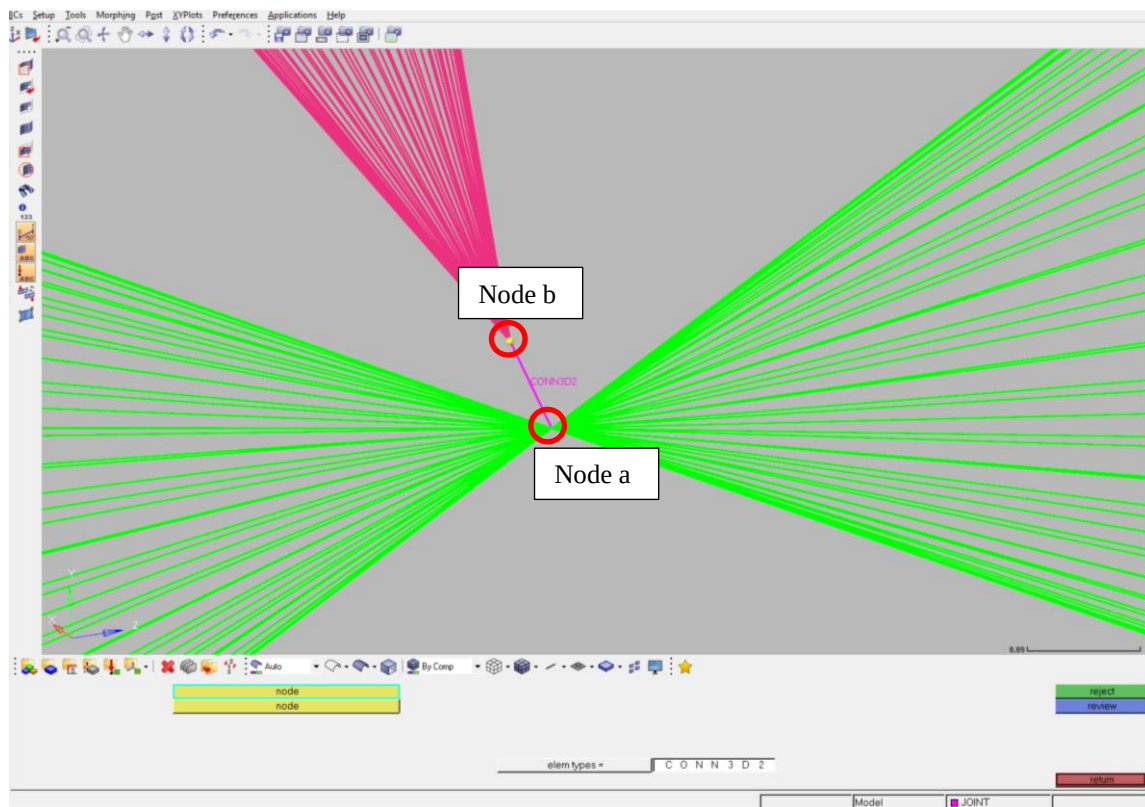


Figure 12 – Rod element creation

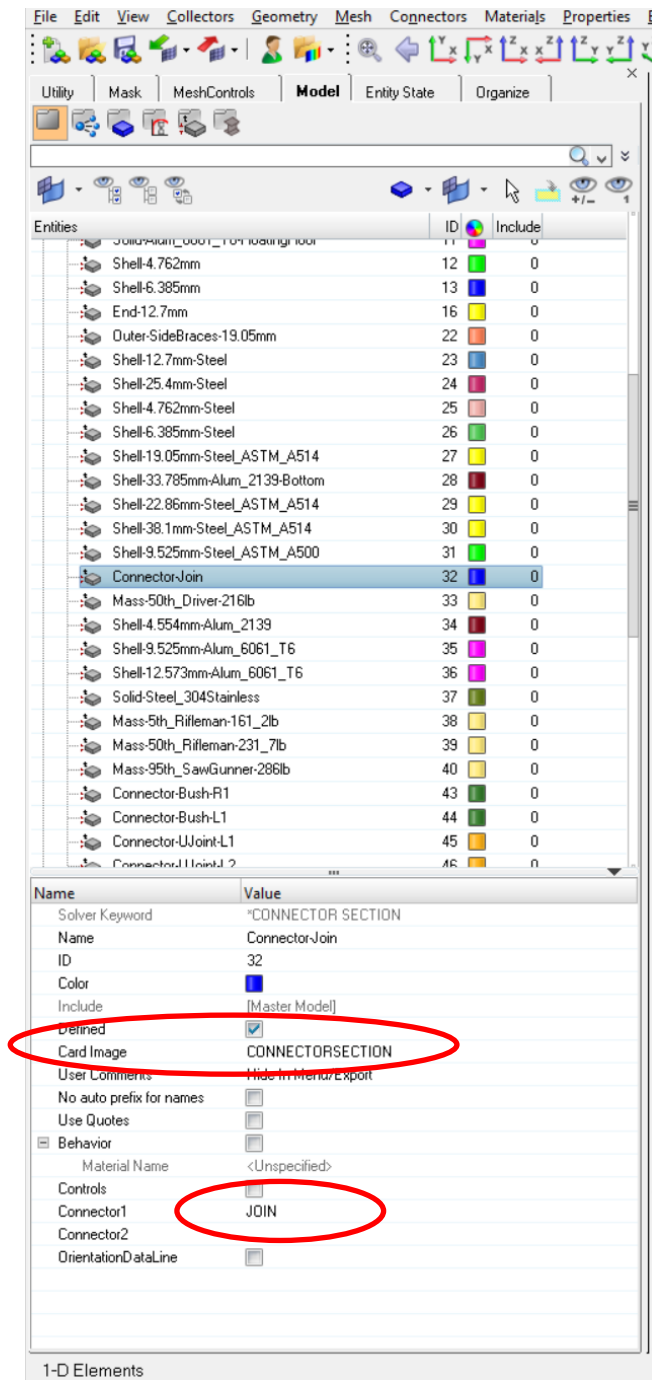


Figure 13 – Connector JOIN type property

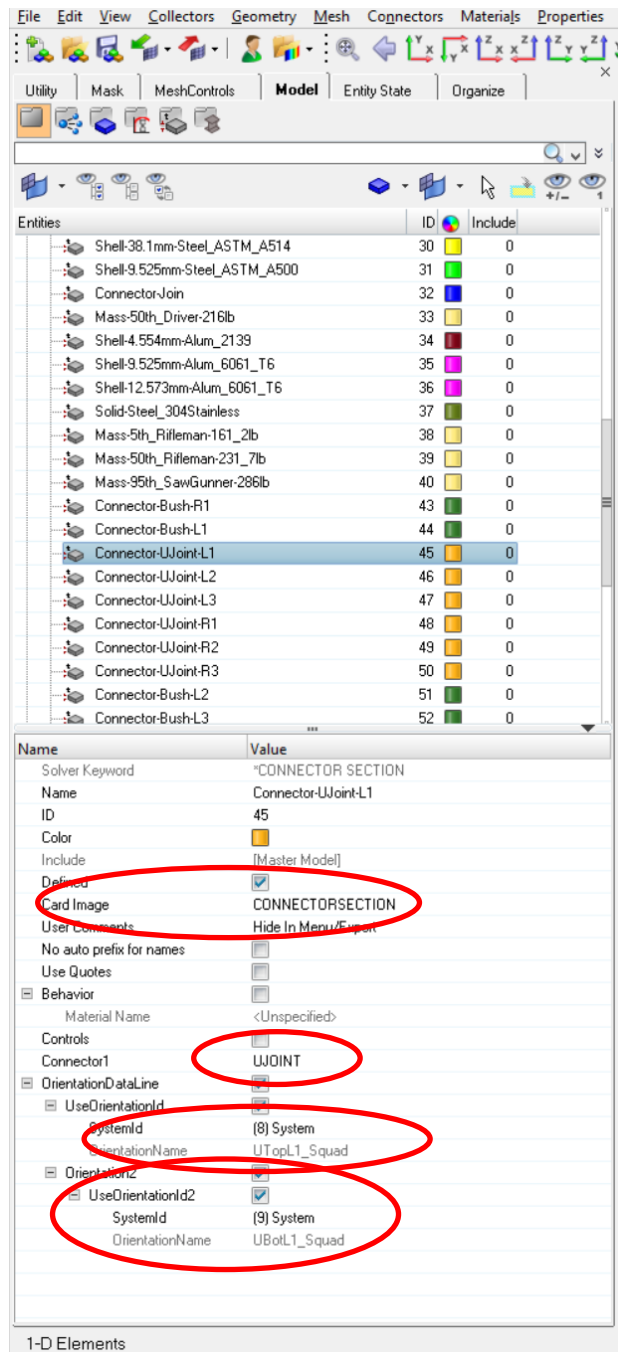


Figure 14 – Connector UJOINT type property

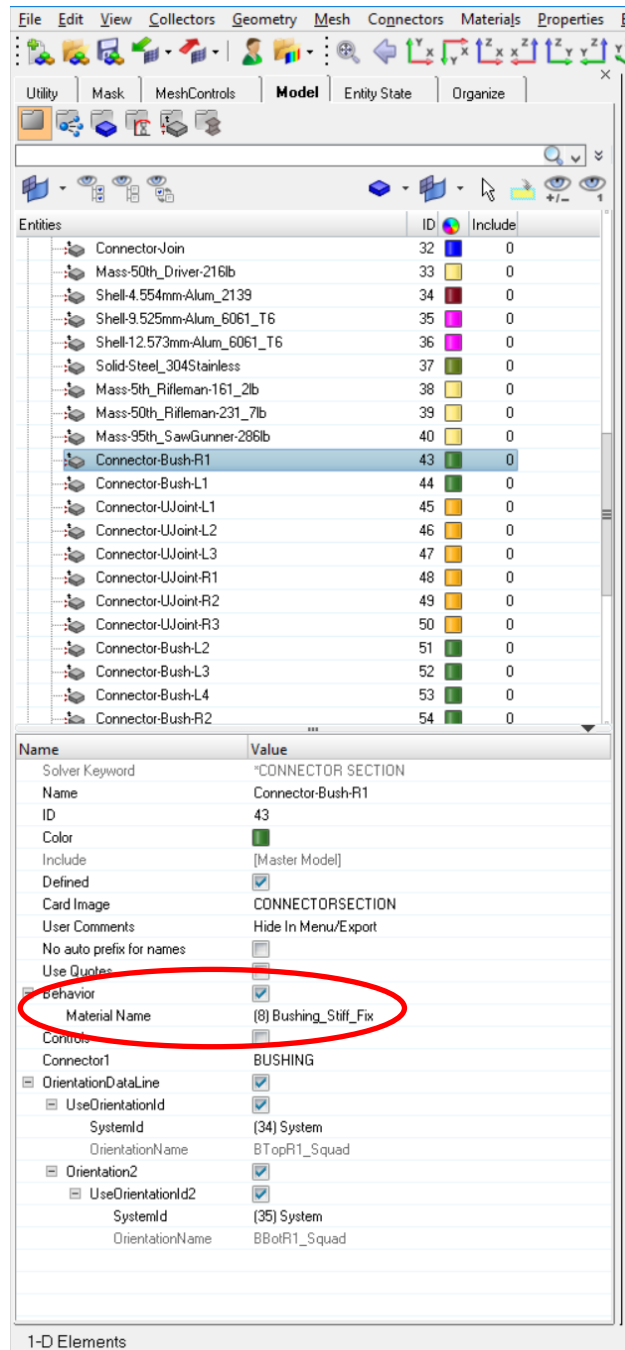


Figure 15 – Connector BUSHING type property

Materials

Materials in HyperMesh are used to define connector element behavior [14]. The card image must be CONNECTOR_BEHAVIOR as shown in Figure 16a for the BUSHING type connector element. These behaviors are defined by assigning a material to the connector property (see Figure 15); assuming the connector element type allows for additional behaviors. Figure 16b shows other behavior that can be defined.

NOTE

The ability to fix a connector element DOF must be done in a separate card image [15]. Based on the orientation of connector node a (here 1=local x direction) the following card image inserted into the ABAQUS/Standard solver input file will fix the specified DOF.

```
*CONNECTOR MOTION, FIXED
[connector element set], [DOF]
```

where [connector element set] is the name of the component in which the connector resides and [DOF] is the available component of the connector to be fixed. See Summary under BUSHING in Appendix A.1 for the available components. From Conventions (ABAQUS/Standard) [16] we have:

Except for axisymmetric elements and electromagnetic elements, the degrees of freedom are always referred to as follows:

- 1 x-displacement
- 2 y-displacement
- 3 z-displacement
- 4 Rotation about the x-axis, in radians
- 5 Rotation about the y-axis, in radians
- 6 Rotation about the z-axis, in radians
- 7 Warping amplitude (for open-section beam elements)
- 8 Pore pressure, hydrostatic fluid pressure, or acoustic pressure
- 9 Electric potential
- 10 Connector material flow (units of length)
- 11 Temperature (or normalized concentration in mass diffusion analysis)
- 12 Second temperature (for shells or beams)
- 13 Third temperature (for shells or beams)
- 14 Etc.

Here the x-, y-, and z-directions coincide with the global X-, Y-, and Z-directions, respectively; however, if a local transformation is defined at a node (see Transformed coordinate systems), they coincide with the local directions defined by the transformation.

A maximum of 20 temperature values (degrees of freedom 11 through 30) can be defined for shell or beam elements in Abaqus/Standard.

REFERENCE

- [1] Clevis (https://en.wikipedia.org/wiki/Clevis_fastener) [2019-01-07]
- [2] Example Cylinder Mounting Accessories (<https://www.milwaukeeecylinder.com/downloads/>) see "Milwaukee Cylinder Complete Catalog" or "Clevis/Brackets/Pins/Rod Eyes Section".
- [3] HyperMesh 2017 (<https://altairhyperworks.com/product/hypermesh>) [2019-01-07]
- [4] ABAQUS/Standard 2017 (<https://www.3ds.com/products-services/simulia/products/abaqus/>) [2019-01-07]

Below references require account at <https://help.3ds.com>

- [5] Connector Overview (ABAQUS/Standard) (https://help.3ds.com/2017/English/DSSIMULIA_Established/SIMACAEELMRefMap/simaelm-c-connectoroverview.htm) [2019-01-07]
- [6] Connector Elements (ABAQUS/Standard) (https://help.3ds.com/2017/english/dssimulia_established/simacaeelmrefmap/simaelm-c-connectorelem.htm) [2019-01-07]
- [7] Connector Type Library (ABAQUS/Standard) (https://help.3ds.com/2017/English/DSSIMULIA_Established/SIMACAEELMRefMap/simaelm-m-ConnectionTypesLibrary-sb.htm) [2019-01-07]

- [8] JOIN
(https://help.3ds.com/2017/English/DSSIMULIA_Established/SIMACAEELMRefMap/simaelm-c-connectiontypedesc-join.htm) [2019-01-07]
- [9] UNIVERSAL
(https://help.3ds.com/2017/English/DSSIMULIA_Established/SIMACAEELMRefMap/simaelm-c-connectiontypedesc-universal.htm) [2019-01-07]
- [10] UJOINT
(https://help.3ds.com/2017/English/DSSIMULIA_Established/SIMACAEELMRefMap/simaelm-c-connectiontypedesc-ujoint.htm) [2019-01-07]
- [11] General multi-point constraints (ABAQUS/Standard)
(https://help.3ds.com/2017/english/DSSIMULIA_Established/SIMACAECSRefMap/simacst-c-mpc.htm) [2019-01-07]
- [12] Orientations (ABAQUS/Standard)
(https://help.3ds.com/2017/english/dssimulia_established/simacaemodrefmap/simamod-c-orientation.htm) [2019-01-07]
- [13] BUSHING
(https://help.3ds.com/2017/English/DSSIMULIA_Established/SIMACAEELMRefMap/simaelm-c-connectiontypedesc-bushing.htm) [2019-01-08]
- [14] Connector Element Behavior (ABAQUS/Standard)
(https://help.3ds.com/2017/English/DSSIMULIA_Established/SIMACAEELMRefMap/simaelm-c-connectorbehavior.htm) [2019-01-08]
- [15] Connector Motion (ABAQUS/Standard)
(https://help.3ds.com/2017/english/DSSIMULIA_Established/SIMACAEKEYRefMap/simakey-r-connectormotion.htm) [2019-01-08]
- [16] Conventions (ABAQUS/Standard)
(https://help.3ds.com/2017/english/DSSIMULIA_Established/SIMACAEMODRefMap/simamod-c-conventions.htm) [2019-01-08]
- [17] Connector Element Output Variables (ABAQUS/Standard)
(https://help.3ds.com/2017/english/DSSIMULIA_Established/SIMACAEOUTRefMap/simaout-c-std-wholeelementvariables.htm) [2019-01-08]

Above references require account at <https://help.3ds.com>

- [18] Spherical Bearings (<https://www.skf.com/us>, search for “plain spherical bearing catalog” then select “SKF spherical plain bearings and rod ends” catalog,
https://cdn.skfmediahub.skf.com/api/public/0901d19680154a05/pdf_preview_medium/0901d19680154a05_pdf_preview_medium.pdf, pg. 27)

APPENDIX A.1 – ABAQUS/Standard CONNECTOR DEFINITION SUMMARY [8, 9, 10, 13]

To view details create an account at <https://help.3ds.com>.

JOIN – (https://help.3ds.com/2017/English/DSSIMULIA_Established/SIMACAEELMRefMap/simaelm-c-connectiontypedesc-join.htm) [2019-01-07]

UNIVERSAL –

(https://help.3ds.com/2017/English/DSSIMULIA_Established/SIMACAEELMRefMap/simaelm-c-connectiontypedesc-universal.htm) [2019-01-07]

UJOINT - (https://help.3ds.com/2017/English/DSSIMULIA_Established/SIMACAEELMRefMap/simaelm-c-connectiontypedesc-ujoint.htm) [2019-01-07]

BUSHING –

(https://help.3ds.com/2017/English/DSSIMULIA_Established/SIMACAEELMRefMap/simaelm-c-connectiontypedesc-bushing.htm) [2019-01-08]

APPENDIX A.2 – ORIENTATIONS [12]

To view details create an account at <https://help.3ds.com>.

Orientations (ABAQUS/Standard) -

(https://help.3ds.com/2017/english/dssimulia_established/simacaemodrefmap/simamod-c-orientation.htm)
[2019-01-07]

- 1) local coordinate systems are defined with points a, b, and c.
- 2) Rectangular: c is at the origin, a is on the x-axis, and b is in the xy plane
- 3) Cylindrical: a and b define the z-axis (x-axis radial, y-axis tangential)
- 4) Spherical: a and b define an axis (x-axis is perpendicular to/radial out at a, y-axis is circumferential about a-b axis at x, z is meridional at x)

APPENDIX A.3 – SPHERICAL BEARINGS [18]

Selection of bearing type

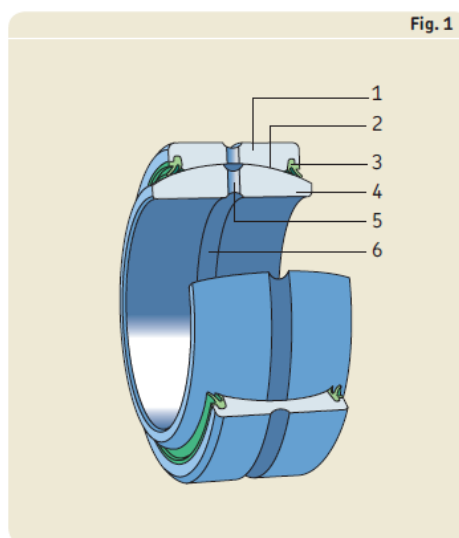
1

Bearing terminology

To better understand frequently used plain bearing and rod end specific terms, definitions are provided in fig. 1 and fig. 2.

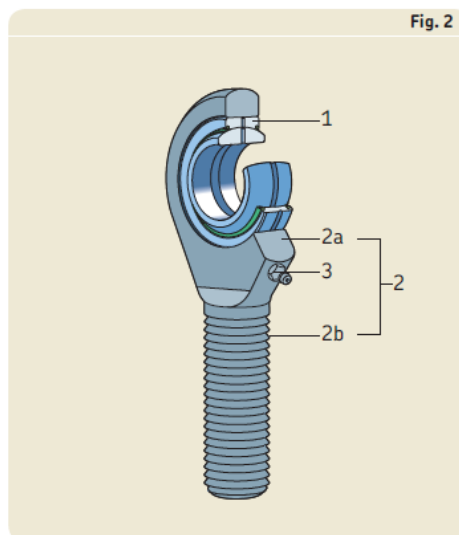
Spherical plain bearing

- 1 Outer ring
- 2 Sliding contact surfaces
- 3 Seal
- 4 Inner ring
- 5 Lubrication hole
- 6 Lubrication groove



Rod end

- 1 Spherical plain bearing
- 2 Rod end
 - 2a Rod end housing
 - 2b Rod end shank, with an external (male) thread. Shanks are also available with an internal (female) thread or with a welding shank.
- 3 Grease fitting



SKF

27

APPENDIX A.4 – ABAQUS CONNECTOR OUTPUT EXAMPLE [17]

Add the yellow highlighted entry to the input file for connector element output.

```

===BOF (.inp file)
...
** defined above
**ELEMENT,TYPE=CONN3D2,ELSET=sphL1
**11622518, 20834838, 17525929
...
** inside step
*OUTPUT, FIELD, NAME = static_out
...
*EL PRINT,ELSET=sphL1
CTF
...
===EOF

```

“.dat” *EL PRINT Output Section

...							
ELEMENT OUTPUT							
THE FOLLOWING TABLE IS PRINTED FOR ELEMENT TYPE CONN3D2 AND ELEMENT SET SPHL1							
ELEMENT	FOOT-	CTF1	CTF2	CTF3	CTM1	CTM2	CTM3
NOTE							
11622518	-247.9	138.8	1.6679E-02	0.000	0.000	0.000	
MAXIMUM	-247.9	138.8	1.6679E-02	0.000	0.000	0.000	
ELEMENT	11622518	11622518	11622518	11622518	11622518	11622518	11622518
MINIMUM	-247.9	138.8	1.6679E-02	0.000	0.000	0.000	
ELEMENT	11622518	11622518	11622518	11622518	11622518	11622518	11622518
...							

TN 19-0301-01

GENERATING A TIME HISTORY FROM A FREQUENCY SPECTRUM USING NCODE GLYPHWORKS/VIBESYS

March 1, 2019

W. Bylsma

U.S. Army Combat Capabilities Development Command (DEVCOM)
Ground Vehicle Systems Center (GVSC)
Systems Integration & Engineering
ATTN: FCDD-GVS-IES/MS 157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

How do you convert a vibration specification, defined in the frequency domain using the power spectral density (PSD) format, into a time series?

PROBLEM SOLUTION:

The solution is based on GlyphWorks Worked Examples [4] and implemented in nCode GlyphWorks [2] using VibeSys [3] glyphs. No theory is presented here, but reference can be made to [4,6]. The implementation of this process uses test phases T05, L06, and V06 from “Table D-11. Narrowband Random-On-Random Vibration Program Data For Installed Equipment On The Crew Compartment Walls Of The M113A1 Armored Personnel Carrier” in ITOP 1-2-601 Laboratory Vibration Schedules [1] to demonstrate

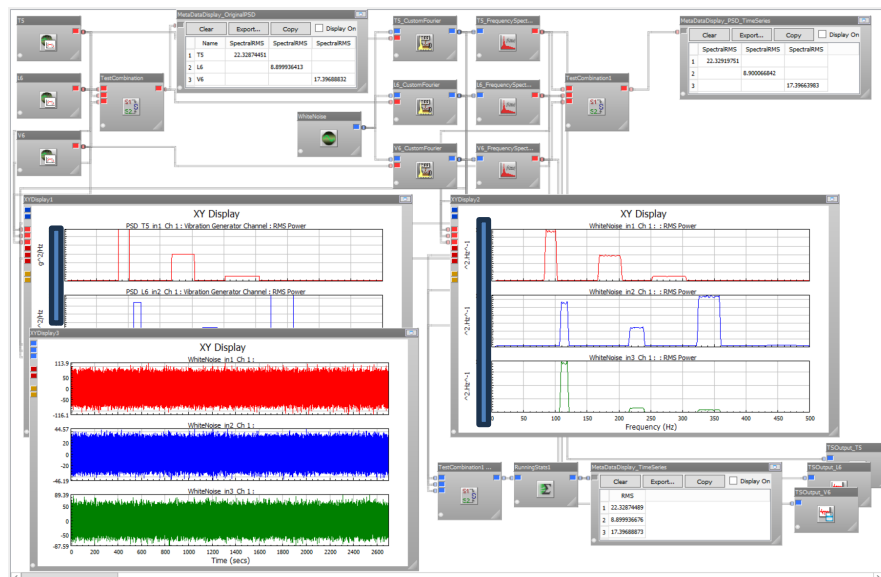


Figure 1 – Overall Process

an example. NOTE: The PSD amplitudes from [1] have been obscured, but this process can be repeated with similar narrow band data in NATO AECTP-400 - Mechanical Environmental Tests, Ed. D V1 (Nov 2019) [1a] which is publicly available. Figure 1 shows the overall process (see also Appendix A.2). During

this time statistics of the signal are taken, in the frequency and time domains, along the way and then compared to ensure equivalency.

Glyphs required from the different functional areas in nCode [2,3] are listed below:

- Input
 - Vibration Generator
 - Time Series Generator
- Function
 - Test Combination
- BasicDSP
 - Frequency Spectrum
 - Running Statistics
- VibeSys
 - Custom Fourier Filter
- Display
 - Meta Data Display
 - XY Display
- Output
 - Time Series Output

Referencing Figure 1, three Vibration Generator glyphs are used to represent the test phases (T05, L06, and V06). Figures 2 – 4 show their definitions.

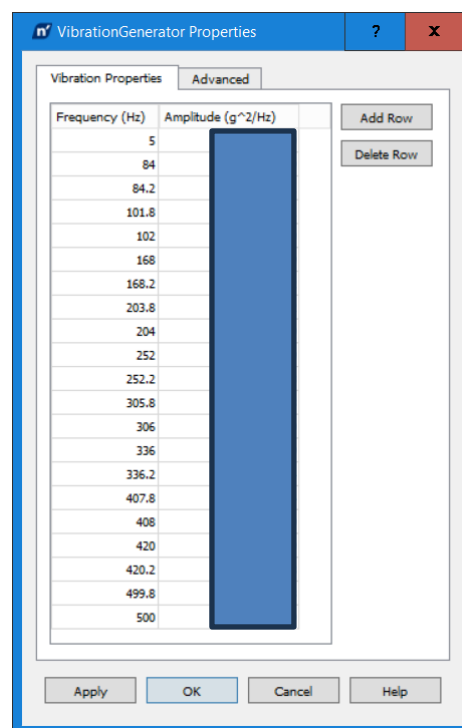


Figure 2 – T5 PSD Definition

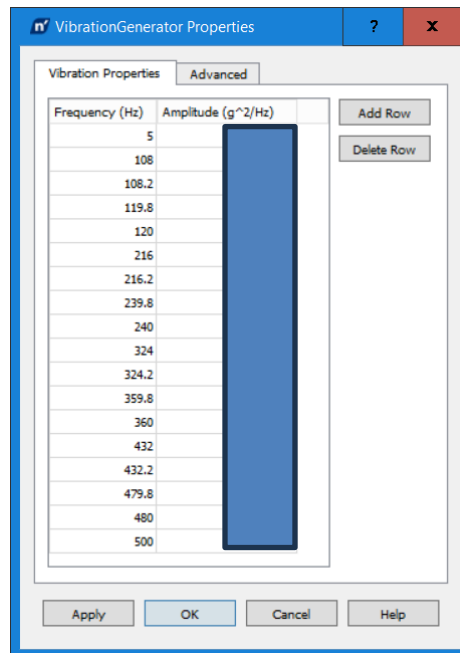


Figure 3 – L6 PSD Definition

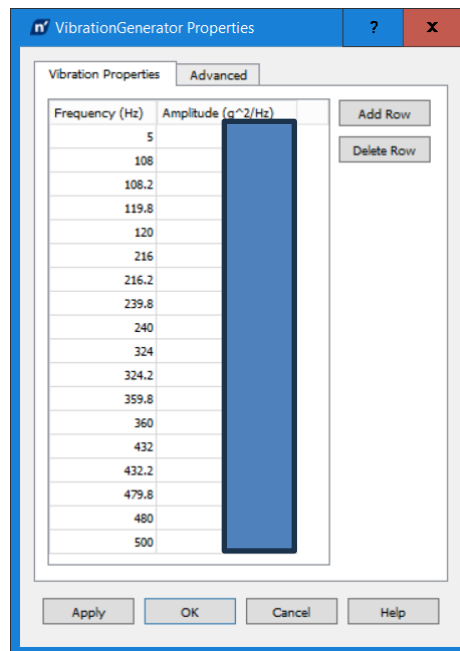


Figure 4 – V6 PSD Definition

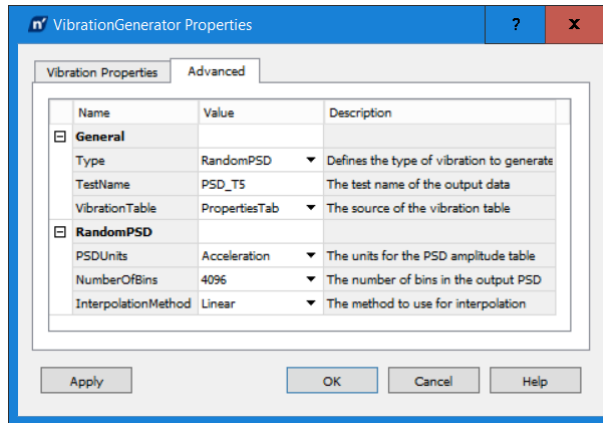


Figure 5 – T5 Vibration Generator Properties

Figure 5 shows the T5 Vibration Generator Properties, but these are duplicated for L06 and V06. Before the three histograms (PSD output) are combined using a Test Combination glyph, which is connected to a Meta Data Display glyph, they are connected to an XY Display glyph to show the individual PSD outputs. See Figures 6 – 8.

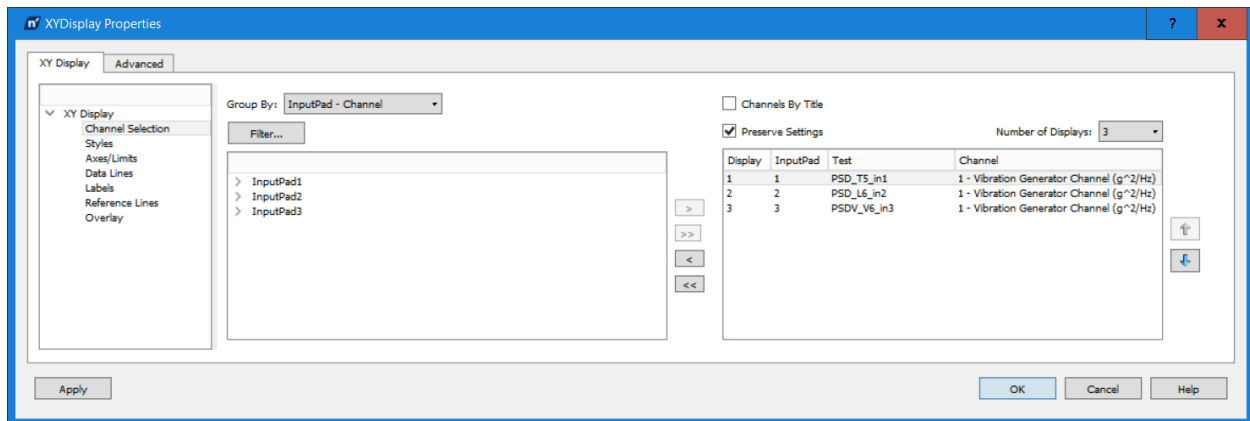


Figure 6a – XY Display Properties - Vibration Generator PSD Output

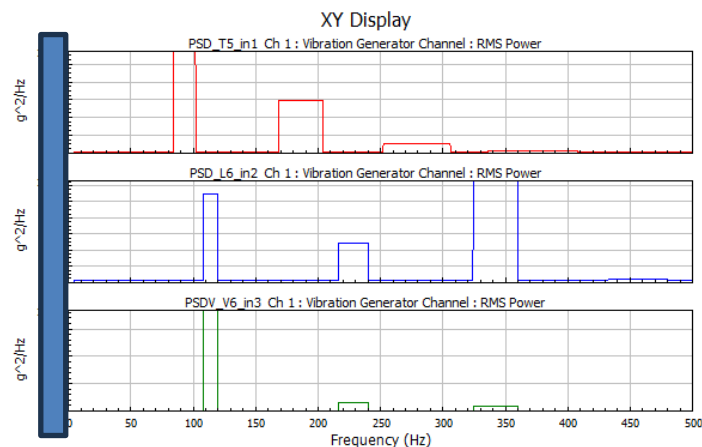


Figure 6b – XY Display Results - Vibration Generator PSD Output

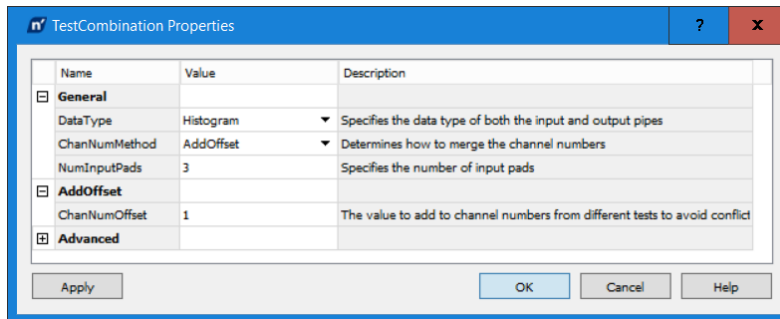


Figure 7 – Test Combination Properties

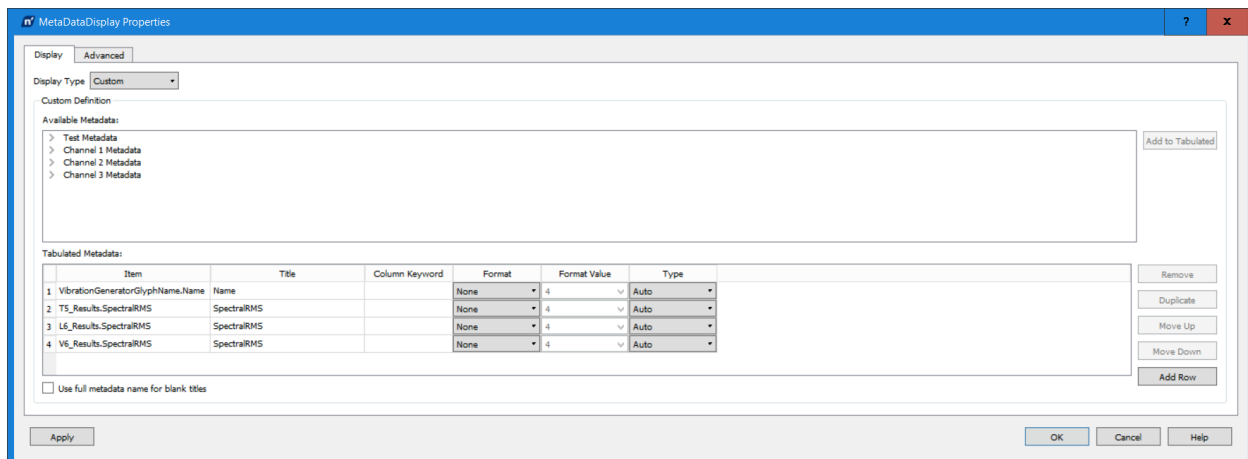


Figure 8a – Meta Data Display Properties - Test Combination of Generated PSD Output

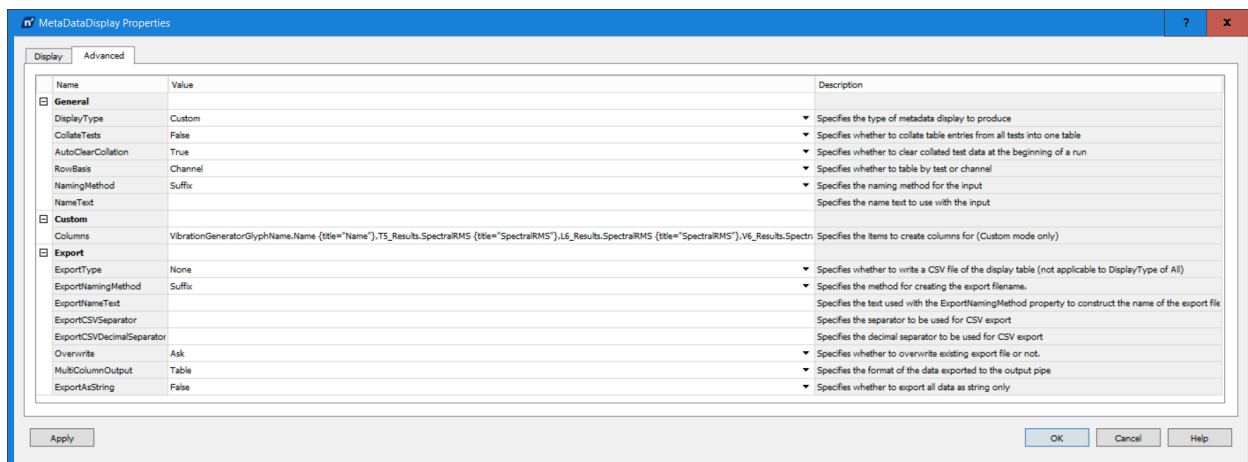


Figure 8b – Meta Data Display Properties - Test Combination of Generated PSD Output

	Name	SpectralRMS	SpectralRMS	SpectralRMS
1	T5	22.32874451		
2	L6		8.899936413	
3	V6			17.39688832

Figure 8c – Meta Data Display Results - Generated PSD Output

A Time Series Generator glyph is used for white noise and connected to each of the three Custom Fourier Filter glyphs whose inputs are from the direct output of each Vibration Generator glyph for T05, L06, and V06, respectively. See Figures 9 and 10.

The screenshot shows the 'TSGenerator Properties' dialog box with the 'General' tab selected. The 'Operation' is set to 'WhiteNoise'. The 'ChannelDetails' tab is also visible, showing 'ChannelNumber' as 1. The 'WhiteNoise' tab is selected, showing 'StandardDeviation' as 1. The 'Advanced' tab is also visible.

Name	Value	Description
General		
Operation	WhiteNoise	Determines the wave type to generate
TestName	WhiteNoise	The test name of the output data
TotalTime	2700	The total time to generate in the output signal, in seconds
SampleRate	5000	The number of data points per second in the output signal
BaseTime	0	The X-axis value of the first data point in the output signal
Median	0	The median of the output signal
ChannelDetails		
ChannelNumber	1	The channel number of the output data
ChannelTitle		The channel title of the output data
XTitle	Time	The X-axis title of the output channel
XUnits	secs	The X-axis units of the output channel
YTitle		The Y-axis title of the output channel
YUnits		The Y-axis units of the output channel
WhiteNoise		
StandardDeviation	1	The standard deviation of the white noise signal
Advanced		

Figure 9 – Time Series Generator Properties - White Noise Input

The screenshot shows the 'CustomFourier Properties' dialog box with the 'General' tab selected. The 'Type' is set to 'HistogramSingleChan'. The 'BufferSize' is set to 4096. The 'RemoveMean' is set to False. The 'HistogramType' is set to PSD.

Name	Value	Description
General		
Type	HistogramSingleChan	Specifies the filter type for processing
BufferSize	4096	Defines the size of the FFT Buffer for processing
RemoveMean	False	Remove the mean
HistogramType	PSD	Specifies the type of the input histogram

Figure 10 – Custom Fourier Filter Properties - Time Series Output

Figure 11 shows the Time Series output from the Custom Fourier Filters connected to an XY Display. These outputs are also connected to a Test Combination glyph, as in Figure 7 but with the DataType as TimeSeries, then to Running Statistics and Meta Data Display glyphs whose properties and output are shown in Figures 12 and 13. The Time Series output is also connected to three Time Series Output glyphs to save the data for further processing. The Time Series glyph properties are shown in Figure 14 with the only difference between each test phase is the NameText changed to reflect the appropriate test.

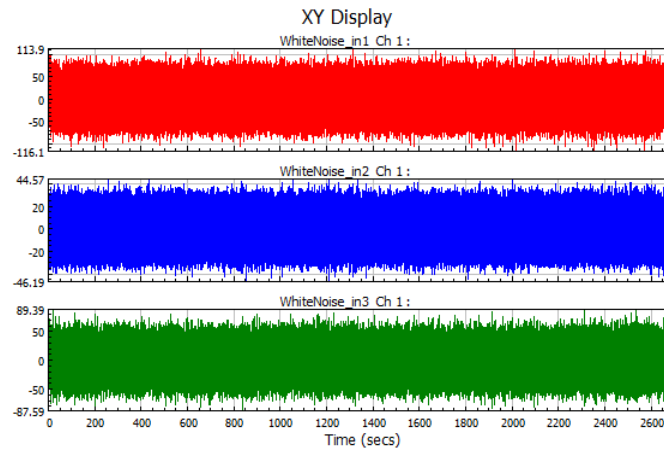


Figure 11 – XY Display - Time Series Output

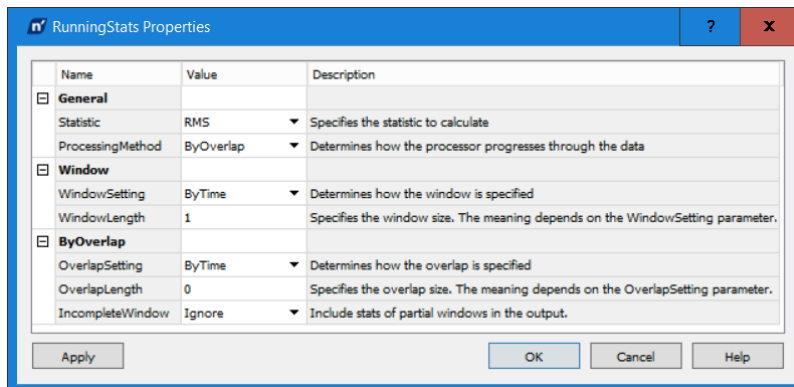


Figure 12 – Running Statistics Properties

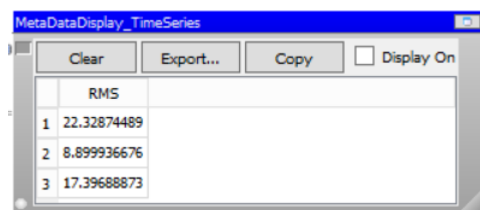


Figure 13 – Meta Data Display - Time Series Output

Name	Value	Description
General		
FileFormat	CSV	Specifies the format of output file to create
MetaDataOutput	SaveWithTest	What to do with the metadata
Overwrite	Ask	Whether to overwrite existing output file(s) or not
AddToFileList	False	Whether to add the newly created output files to the file list
LoadOutputFilesIntoInputGlyph		The names of time series input glyphs in other workspaces to load with the output files
ClearInputGlyphFiles	True	Whether to clear any input glyph file lists before loading with the output files
Naming		
NamingMethod	Suffix	Output naming method
NameText	_out_V6	Text used to form the output test name
SeparatedValue		
WriteMetadataToSV	False	Whether to write the test level metadata to the separated value file or not

Apply OK Cancel Help

Figure 14 – Time Series Output Properties

Each Custom Fourier Filter glyph is connected to an individual Frequency Spectrum glyph. Each of the Frequency Spectrum glyphs is connected to an XY Display to compare the reconstructed frequency spectrum PSD of the time series to the original definition. See Figures 15 and 16.

Name	Value	Description
General		
OutputType	Power	Specifies the type of spectrum to produce
AveragingMethod	Linear	Specifies the way to combine the data from multiple FFT buffers
AutoHistogramLabels	True	Specifies whether to automatically set the axis labels and units text on the output histogram
AnalysisType		
AnalysisType	Standard	Standard or Zoom analysis
Buffer		
WindowType	Hanning	Specifies the window type to apply to each buffer
BufferSize	4096	Specifies the FFT buffer size
BufferOverlap	67	Specifies the overlap on the input data for each FFT analysis
WindowOrder		
WindowPower	1	Specifies the power to apply to a window
AdvancedSettings		

Apply OK Cancel Help

Figure 15 – Frequency Spectrum Properties

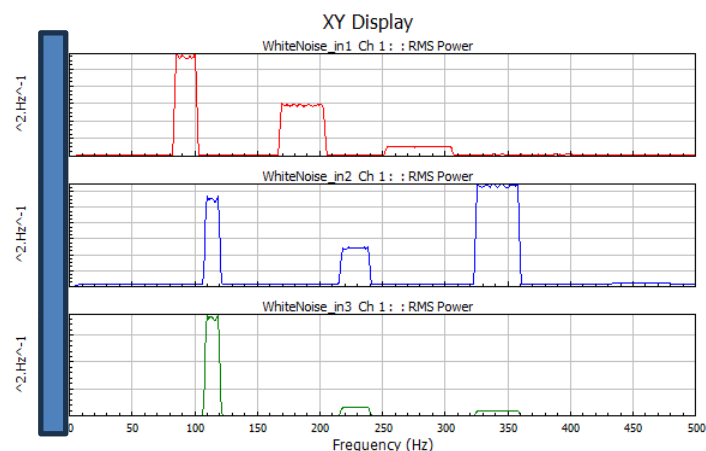


Figure 16 – XY Display – Reconstructed PSD Output

A Test Combination glyph, with the same properties as in Figure 7, connects the reconstructed histograms (PSD output) to a Meta Data Display to compare reconstructed frequency statistics to the original definition. These properties and output are shown in Figures 17a – 17c.

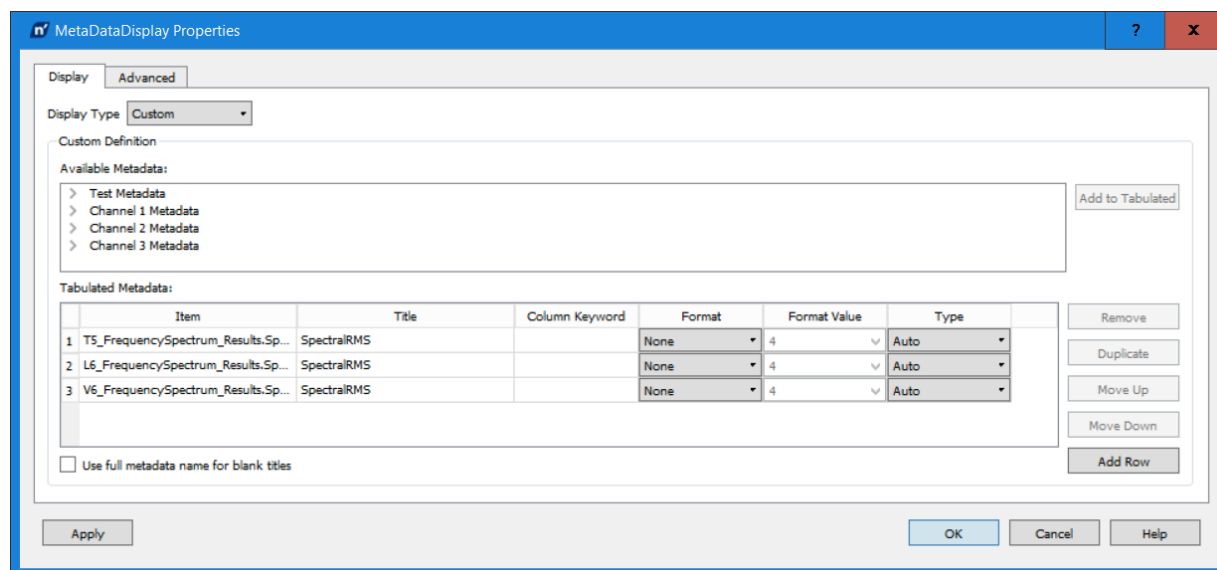


Figure 17a – Meta Data Display Properties - Reconstructed PSD Output

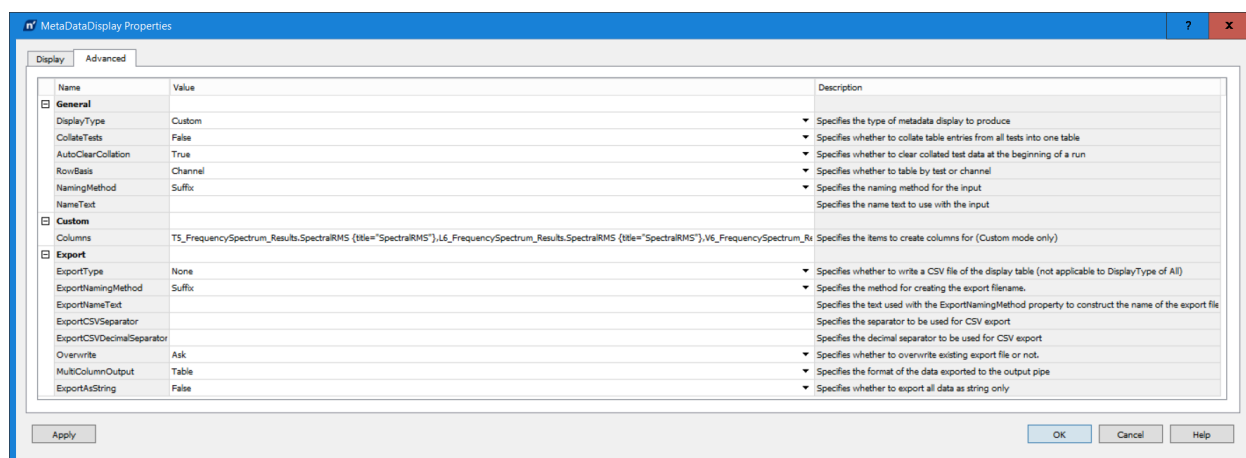


Figure 17b – Meta Data Display Properties - Reconstructed PSD Output

	SpectralRMS	SpectralRMS	SpectralRMS
1	22.32919751		
2		8.900066842	
3			17.39663983

Figure 17c – Meta Data Display - Reconstructed PSD Output

Figure 18 shows the PSD's calculated for each time series using MATLAB [5]. Comparison with Figures 6b and 16 show good agreement.

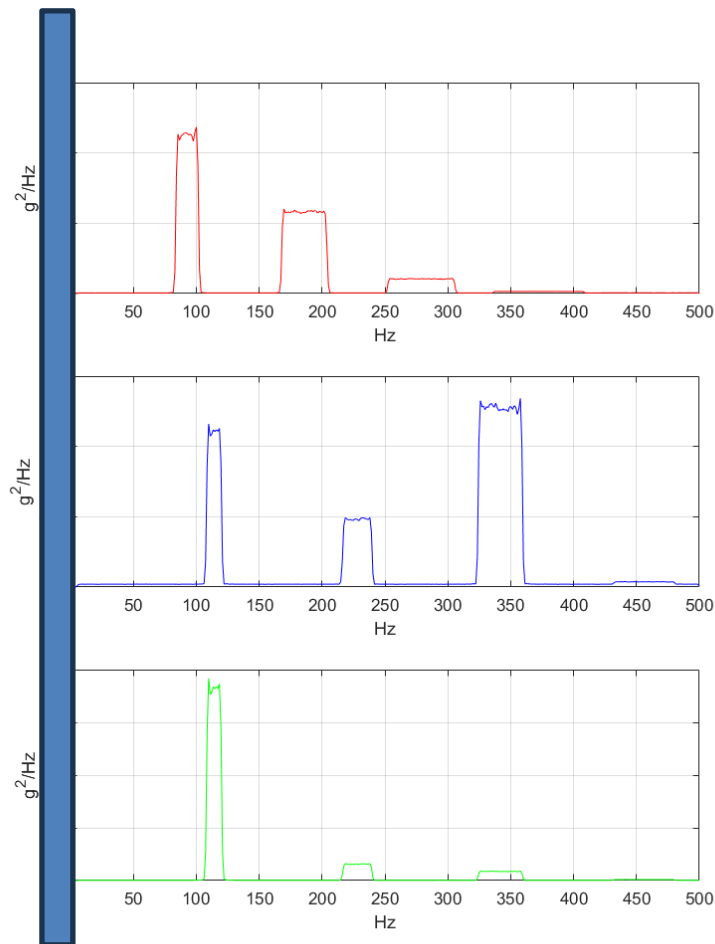


Figure 18 – MATLAB PSD Output

MATLAB RMS Results

```
ans =
    22.3287

ans =
    8.8999

ans =
    17.3969
```

Comparison of Figures 8c, 13, and 17c to the MATLAB RMS Results above (see Appendix A.1) for each of the time series confirms that the process demonstrated here and outlined in [4] preserves the signal statistics in the frequency and time domains. The results indicate agreement to nearly two decimal places. This example demonstrates that the process outlined should provide an accurate representation of a stationary random signal in either the time or frequency domain.

REFERENCE

- [1] International Test Operations Procedure (ITOP) 1-2-601 Laboratory Vibration Schedules (23 April 1998). DIST B (DTIC AD B238 288). (Superseded by ITOP-1-2-601 (DTIC AD B406 927), dated 2009-02-19 which was superseded by Test Operations Procedure (TOP) 1-2-601 Ground Vehicle Laboratory Vibration Schedules, 18 May 2015). The ITOP is considered inactive for U.S. purposes and should not be used by U.S. government personnel and their agents unless specifically called for in a requirement or specification document.

- [1a] NATO AECTP-400 - Mechanical Environmental Tests, Ed. D V1 (Nov 2019). Default Vibration Severities (Tracked Vehicles) in Annex A-26 (pg.A-36) to Annex A-33 (pg.A-43). Per C-M(2002)60 Information releasable to the Public (it shall carry no markings of any sort) see (<https://nso.nato.int/nso/nsdd/main/standards>, search for AECTP).

- [2] nCode GlyphWorks (nCode 2018.1, GlyphWorks V18.1.0 Release Build 281) (<https://www.ncode.com/products/ncode-glyphworks-signal-processing-and-durability-analysis>) [2019-03-01] now (<https://www.hbkworld.com/en/products/software/analysis-simulation/durability/ncode-glyphworks-signal-processing-and-durability-analysis>)

- [3] nCode GlyphWorks VibeSys (<https://www.ncode.com/products/ncode-vibesys-acoustic-and-vibration-analysis-software>) [2019-03-01] now (<https://www.hbkworld.com/en/products/software/analysis-simulation/durability/ncode-vibesys-acoustic-and-vibration-analysis-software>)

- [4] GlyphWorks Worked Examples (GW-WEX, 18.1.01) – pg. 360 “18 Regenerating a Time History from a Frequency Spectrum”

- [5] MATLAB (<https://www.mathworks.com/products/matlab.html>) [2019-03-04]

- [6] Vibration Data (Tom Irvine) (http://www.vibrationdata.com/tutorials_alt/psdsyn.pdf) [2019-03-04]

APPENDIX A.1 – MATLAB SCRIPT TO GENERATE RMS VALUES AND PSD PLOT

```

%T5 L6 V6
yt5=dlmread('WhiteNoise_out_T5.csv',' ',8,1); % not read header, skip row number
yl6=dlmread('WhiteNoise_out_L6.csv',' ',8,1); % not read header, skip row number
yv6=dlmread('WhiteNoise_out_V6.csv',' ',8,1); % not read header, skip row number

diary out.txt
rms(yt5(:,2))
rms(yl6(:,2))
rms(yv6(:,2))
diary off

[Pxxt5,Ft5]=pwelch(yt5(:,2),4096,67,[],5000);
[Pxxl6,F16]=pwelch(yl6(:,2),4096,67,[],5000);
[Pxxv6,Fv6]=pwelch(yv6(:,2),4096,67,[],5000);

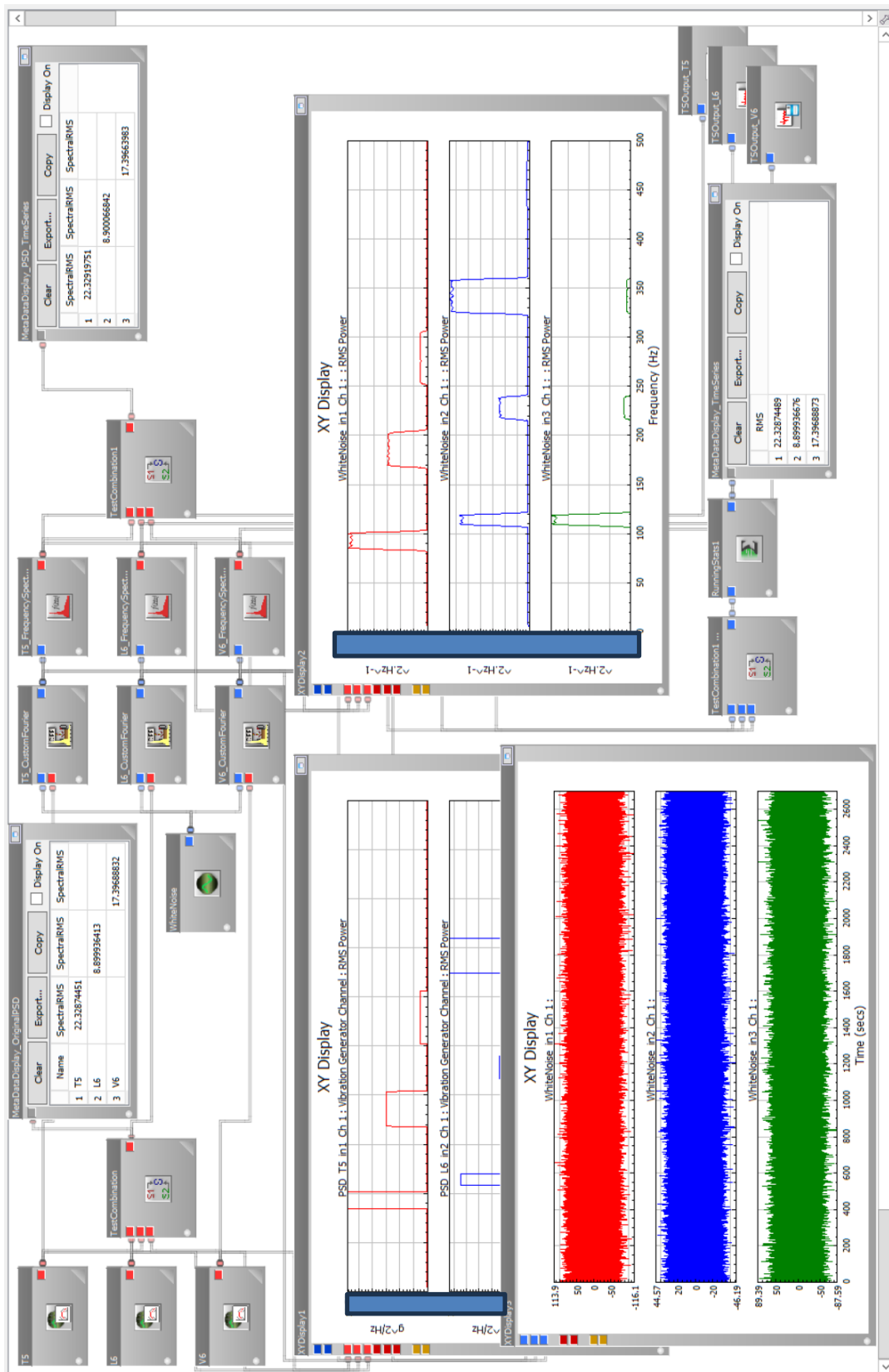
figure(1)
clf
subplot(3,1,1),plot(Ft5,Pxxt5,'r')
subplot(3,1,1),xlim([0,500])
subplot(3,1,1),ylabel('g^2/Hz')
subplot(3,1,1),xlabel('Hz')
subplot(3,1,1),grid on

subplot(3,1,2),plot(F16,Pxxl6,'b')
subplot(3,1,2),xlim([0,500])
subplot(3,1,2),ylabel('g^2/Hz')
subplot(3,1,2),xlabel('Hz')
subplot(3,1,2),grid on

subplot(3,1,3),plot(Fv6,Pxxv6,'g')
subplot(3,1,3),xlim([0,500])
subplot(3,1,3),ylabel('g^2/Hz')
subplot(3,1,3),xlabel('Hz')
subplot(3,1,3),grid on

```

APPENDIX A.2 – OVERALL PROCESS



APPENDIX A.3 – CALCULATION OF RMS FROM PSD

The PSD information in Figures 2 to 4 is summarized below for each case (NB = narrow band, BW = bandwidth). The calculated RMS values agree with Figures 8c and 13 except for numerical round-off error.

	B	C	D	E	F	G
2	T5	Frequency Span (Hz)		BW (Hz)	PSDval (grms ² /Hz)	PSDval*BW
3						
4	Example data					
5	Base	5	800	795	0.0689	54.77
6	NB1	60	70	10	8.643	86.43
7	...					
8						
9	RMSval					22.4150
10						
11	L6	Frequency Span (Hz)		BW (Hz)	PSDval (grms ² /Hz)	PSD*BW
12						
13						
14						
15						
16						
17						
18	RMSval					8.9293
19						
20	V6	Frequency Span (Hz)		BW (Hz)	PSDval (grms ² /Hz)	PSD*BW
21						
22						
23						
24						
25						
26						
27	RMSval					17.5301

The PSD values are defined as

$$PSDval = \frac{(RMSval)^2}{BW} \quad (1)$$

therefore

$$RMSval = \sqrt{(PSDval)BW} \quad (2)$$

Note that in the case of sinusoids the RMS value is defined as

$$RMSval = \sqrt{\frac{(Amplitude)^2}{2}} = \frac{Amplitude}{\sqrt{2}} \quad (3)$$

which when equation (2) is substituted into equation (3) gives

$$Amplitude = \sqrt{2}(RMSval) = \sqrt{2(PSDval)BW} \quad (4)$$

Equation (4) can be used to define the sinusoidal amplitude that is needed to generate a desired PSD level (e.g. for a sine sweep, etc.).

The RMS value for each case is calculated by summing the areas (PSDval*BW) and then taking the square root. The base values are subtracted from the NB PSDvals so that the base area is not duplicated. The formulas below show the necessary computations.

	B	C	D	E	F	G
2	T5	Frequency Span (Hz)		BW (Hz)	PSDval (grms ² /Hz)	PSDval*BW
3			=D3-C3			=F3*E3
4			=D4-C4			=(F4-\$F\$3)*E4
5			=D5-C5			=(F5-\$F\$3)*E5
6			=D6-C6			=(F6-\$F\$3)*E6
7			=D7-C7			=(F7-\$F\$3)*E7
8			=D8-C8			=(F8-\$F\$3)*E8
9					RMSval	=SQRT(SUM(G3:G8))
10						
11	L6	Frequency Span (Hz)		BW (Hz)	PSDval (grms ² /Hz)	PSD*BW
12			=D12-C12			=F12*E12
13			=D13-C13			=(F13-\$F\$12)*E13
14			=D14-C14			=(F14-\$F\$12)*E14
15			=D15-C15			=(F15-\$F\$12)*E15
16			=D16-C16			=(F16-\$F\$12)*E16
17					RMSval	=SQRT(SUM(G12:G17))
18						
19						
20	V6	Frequency Span (Hz)		BW (Hz)	PSDval (grms ² /Hz)	PSD*BW
21			=D21-C21			=F21*E21
22			=D22-C22			=(F22-\$F\$12)*E22
23			=D23-C23			=(F23-\$F\$12)*E23
24			=D24-C24			=(F24-\$F\$12)*E24
25			=D25-C25			=(F25-\$F\$12)*E25
26					RMSval	=SQRT(SUM(G21:G26))
27						

NOTE: Narrow band sinusoidal sweeps can be simulated with rectangular areas, as done here. However, the number of sweeps and other parameters may affect the PSD level. Caution must be taken to ensure that what is desired is properly represented.

TN 19-0514-01

GENERATING SINE SWEEP/CHIRP SIGNALS

May 14, 2019

W. Bylsma

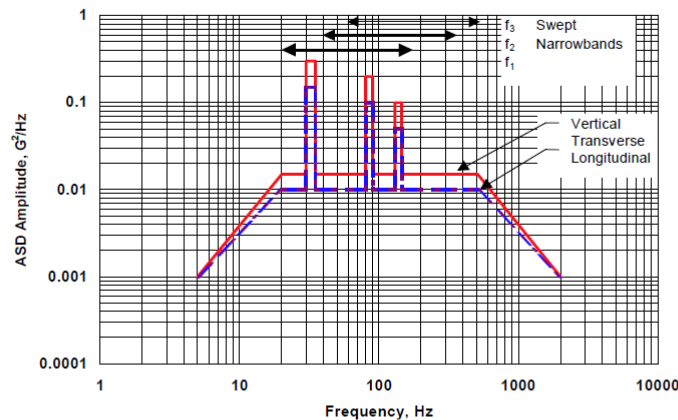
U.S. Army Combat Capabilities Development Command (DEVCOM)
Ground Vehicle Systems Center (GVSC)
Detroit Arsenal
Systems Integration & Engineering
ATTN: FCDD-GVS-IES/MS 157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

Random vibration spectrums are defined in [4] to be used as test inputs for structural response assessments. An example spectrum defined in [4] is shown below.

AECTP 400
(Edition 3)
ANNEX B 401

FIGURE B-2 MATERIEL IN TURRET BUSTLE RACK OR INSTALLED IN TURRET



A descriptive definition is as follows:

2.5.9 Swept Frequency Narrowband Random Vibration on Wideband Random Vibration
Swept frequency narrowband random vibration on wideband random vibration is defined as one or more narrow bands of random vibration swept over a frequency range and superimposed on a background of wideband random vibration. A composite vibration severity of a swept narrowband random vibration superimposed on a wideband random vibration background is defined by the following parameters.

Details of the spectrum breakpoints are shown in the table below.

Figure B-2 Tracked Vehicle Schedule Breakpoints						
Wideband Random Spectrum				Harmonic Swept Narrowbands		
Frequency, Hz	Axis, ASD Amplitude G ² /Hz			Narrowband	f ₁	f ₂
	V	T	L			
5	0.001	0.001	0.001	Bandwidth, Hz	5	10
20	0.015	0.010	0.010	Swept BW, Hz	20 - 170	40 - 340
510	0.015	0.010	0.010	# Sweeps	2	2
2000	0.001	0.001	0.001			
				Axis	ASD Amplitude, G ² /Hz	
				Vertical	0.30	0.20
Wideband Grms	3.56	3.03	3.03	Transverse	0.15	0.10
Total Grms	4.20	3.42	3.42	Longitudinal	0.15	0.10

Although some test equipment can perform continuous narrow band sweeps on wideband random vibration many simulation software products do not have that capability.

Random-on-Random (RoR) mode enables up to 25 independently sweeping or fixed narrowband random signals to be overlaid onto a background random spectrum. RoR is used, for instance, for simulating military tracked vehicles. Each sweeping narrowband may have its own profile (PSD amplitudes) and limits. The narrow bands can sweep at different rates and directions and also can overlap. If required, the user specifies rates at which the narrow bands are switched on/off as well as relative offsets. [5]

A good approximation would be to combine multiple spectrums where the narrow bands (NB) are discretely incremented over the continuous swept frequencies. With finer increments a continuous swept signal is approximated.

How is a sine sweep (or chirp) signal generated so we can define the increments for the swept frequencies?

PROBLEM SOLUTION:

The definition laid out in [1] and Appendix A.1 provide the basis for generating the test signals. From this the defined frequencies can be used to determine the NB center frequency increments. Reference to sine sweep or chirp signals are essentially synonymous. The frequency for both can vary linearly or logarithmically (called exponentially for the chirp function).

A linear rate is referred to in units of [Hz/sec] while a logarithmic rate is referred to in units of [octaves/sec]. Recall that

$$\sin(\theta(t)) = \sin(w(t)t) = \sin(2\pi f(t)t), \text{ where } w(t) = \frac{d\theta(t)}{dt} = 2\pi f(t) \quad (1)$$

where the rates are defined in terms of the instantaneous frequency $f(t)$.

Linear Rate

The instantaneous frequency is defined as

$$f(t) = f_0 + R_{lin}t \quad (2)$$

where the linear rate is

$$R_{lin} = \frac{f(T) - f_0}{T} \left[\frac{Hz}{sec} \right] \quad (3)$$

and f_0 is the initial frequency, T is the total time period. From (2)

$$\begin{aligned} f(0) &= f_0 + R_{lin}0 = f_0 \\ f(T) &= f_0 + R_{lin}T. \end{aligned} \quad (4)$$

Integration of the instantaneous frequency gives the phase as a function of time

$$\theta(t) = \theta_0 + 2\pi \int_0^t (f_0 + R_{lin}\tau) d\tau = \theta_0 + 2\pi \left(f_0\tau \Big|_0^t + \frac{R_{lin}\tau^2}{2} \Big|_0^t \right) = \theta_0 + 2\pi \left(f_0t + \frac{R_{lin}t^2}{2} \right). \quad (5)$$

Logarithmic Rate

Before calculating the logarithmic rate a review of the definition of an octave band is in order---see reference [6]. In short an octave is the doubling or halving of a frequency (see equation 8).

In equations 6 and 7, f_c is the center frequency, f_u is the upper frequency, and f_l is the lower frequency. The frequencies between f_l and f_u form the octave band.

$$f_u = f_c \left(2^{\frac{1}{2}} \right) \quad (6)$$

$$f_l = f_c / \left(2^{\frac{1}{2}} \right) = f_c \left(2^{-\frac{1}{2}} \right) \quad (7)$$

or

$$f_u = f_l(2^1), \quad f_l = f_u(2^{-1}). \quad (8)$$

For a 1/m octave band we have

$$f_u = f_c \left(2^{\frac{1}{2}} \right)^{\frac{1}{m}} = f_c \left(2^{\frac{1}{2m}} \right) \quad (9)$$

and for the nth octave we have

$$f_u = f_c \left(2^{\frac{n}{2m}} \right). \quad (10)$$

From equations 6 to 10 we can conclude that

$$f_l = f_c / \left(2^{\frac{n}{2m}} \right) = f_c \left(2^{-\frac{n}{2m}} \right) \quad (11)$$

and that

$$f_c = \sqrt{f_l f_u}. \quad (12)$$

From [7] the number of octaves is defined as

$$n = \log_2 \left(\frac{f_u}{f_l} \right) \quad (13)$$

(with 1/m octave bands, n represents n/m).

The instantaneous frequency is defined as

$$f(t) = f_0 2^{R_{log} t} \quad (14)$$

where the logarithmic rate is

$$R_{log} = \frac{\log_2 \left(\frac{f_u}{f_l} \right)}{T} \left[\frac{\text{octaves}}{\text{sec}} \right] \quad (15)$$

and T is the total time period. From (14)

$$\begin{aligned} f(0) &= f_0 2^{R_{log} 0} = f_0 \\ f(T) &= f_0 2^{R_{log} T} \end{aligned} \quad (16)$$

Integration of the instantaneous frequency gives the phase as a function of time

$$\theta(t) = \theta_0 + 2\pi \int_0^t (f_0 2^{R_{log} \tau}) d\tau = \theta_0 + 2\pi f_0 \left(\frac{2^{R_{log} t}}{\ln(2^{R_{log}})} \Big|_0^t \right) = \theta_0 + 2\pi f_0 \left(\frac{2^{R_{log} t} - 1}{\ln(2^{R_{log}})} \right). \quad (17)$$

(see [8] and Appendix A.2)

With $k = 2^{R_{log}}$ (therefore $k = \left(\frac{f_u}{f_l} \right)^{\frac{1}{T}}$ from equation 15) the derivation matches that in Appendix A.1.

===

Software platforms [2,3] are used to manually calculate the sine sweep/chirp functions and compare them to “built in” functions provided with each. Appendix A.3 and A.4 show the implementation details.

MATLAB

Figure 1 shows the instantaneous frequency variation between linear and logarithmic sweeps. For the logarithmic sweep, Figures 2 and 3 show the difference between the calculated and “built-in” function. (Note that the function is plotted offset by 0.1 to show both signals.) The difference between the logarithmic sweeps is less than 1e-9 in magnitude. For the linear sweep, Figures 4 and 5 show the difference between the calculated and “built-in” function. The difference between the linear sweeps is essentially zero.

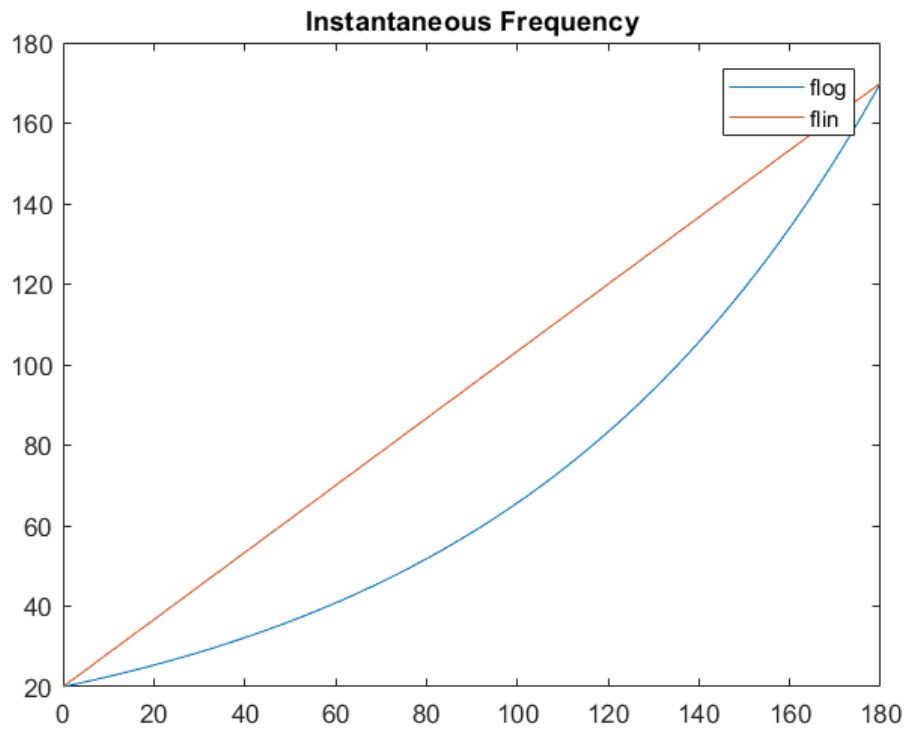


Figure 1 – Linear vs. Logarithmic Frequency Comparison

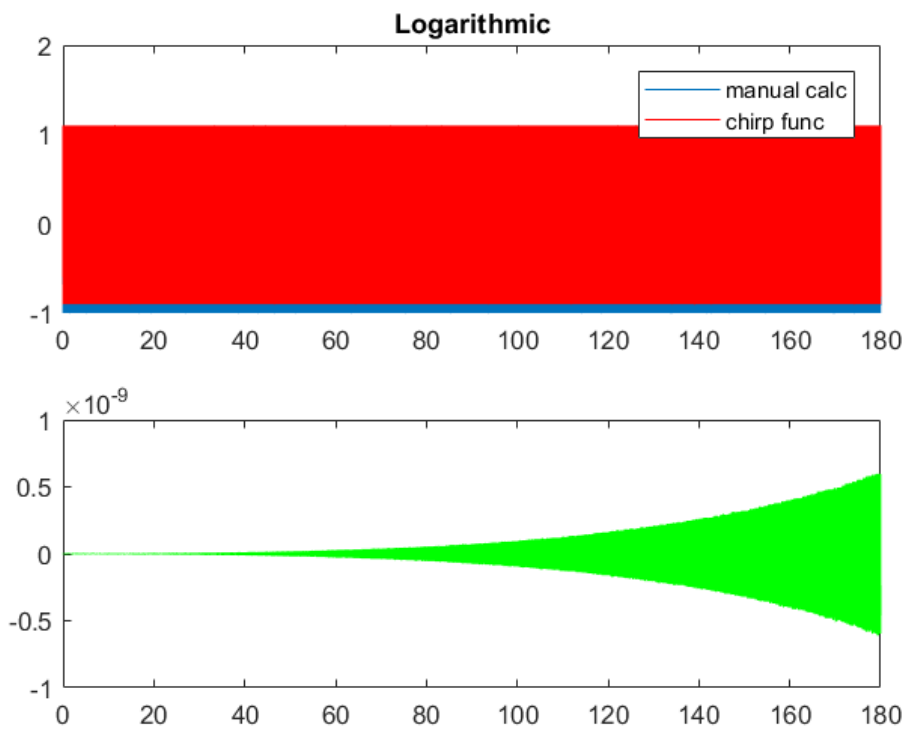


Figure 2 – Logarithmic Calculated vs. Chirp Function (MATLAB) and Difference

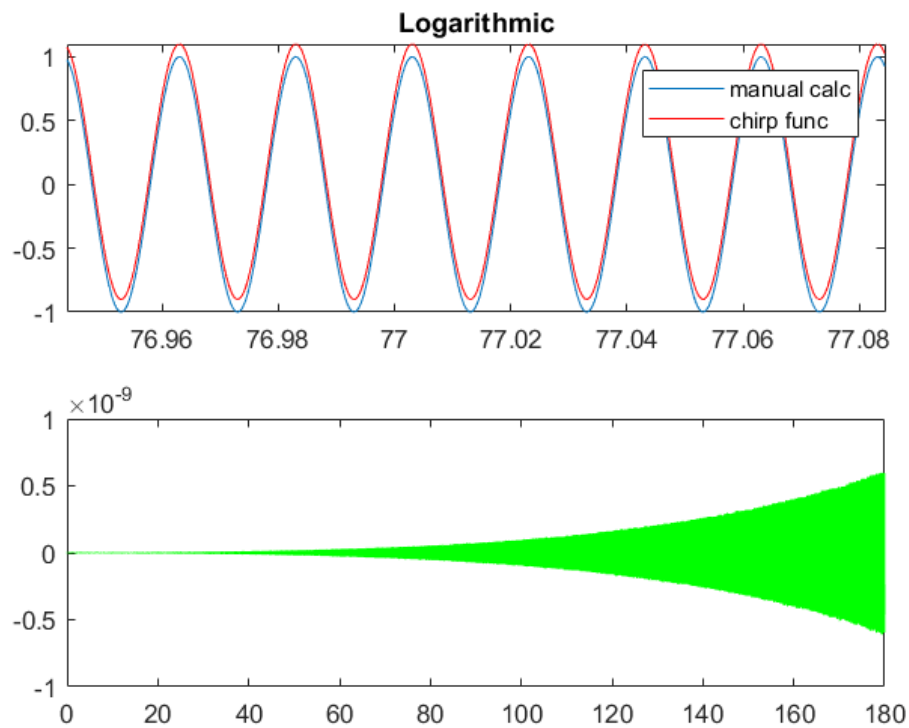


Figure 3 – Zoomed - Logarithmic Calculated vs. Chirp Function (MATLAB) and Difference

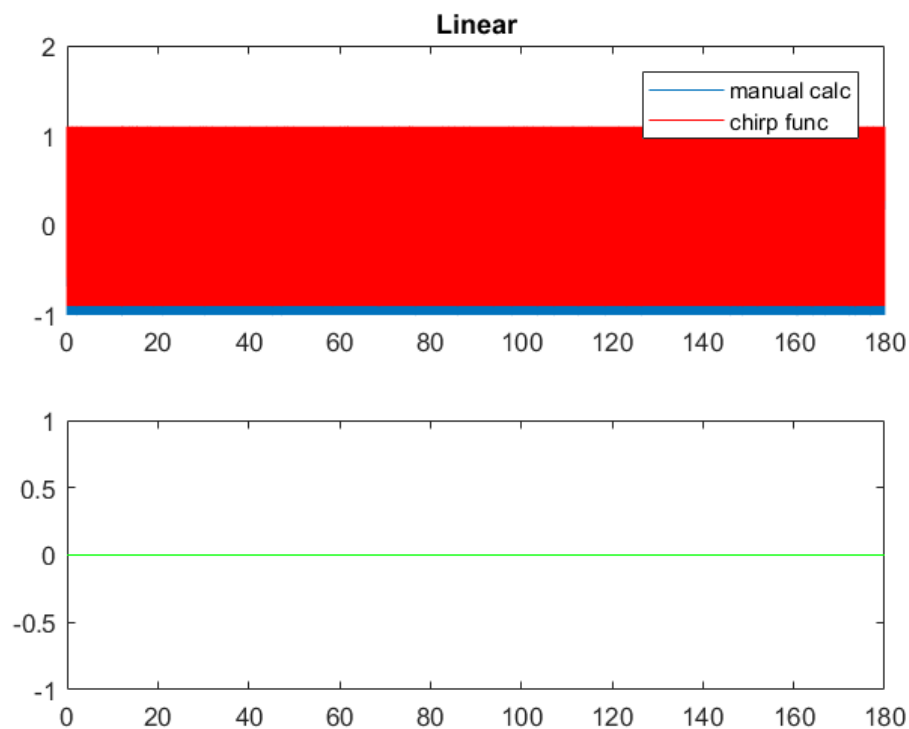


Figure 4 – Linear Calculated vs. Chirp Function (MATLAB) and Difference

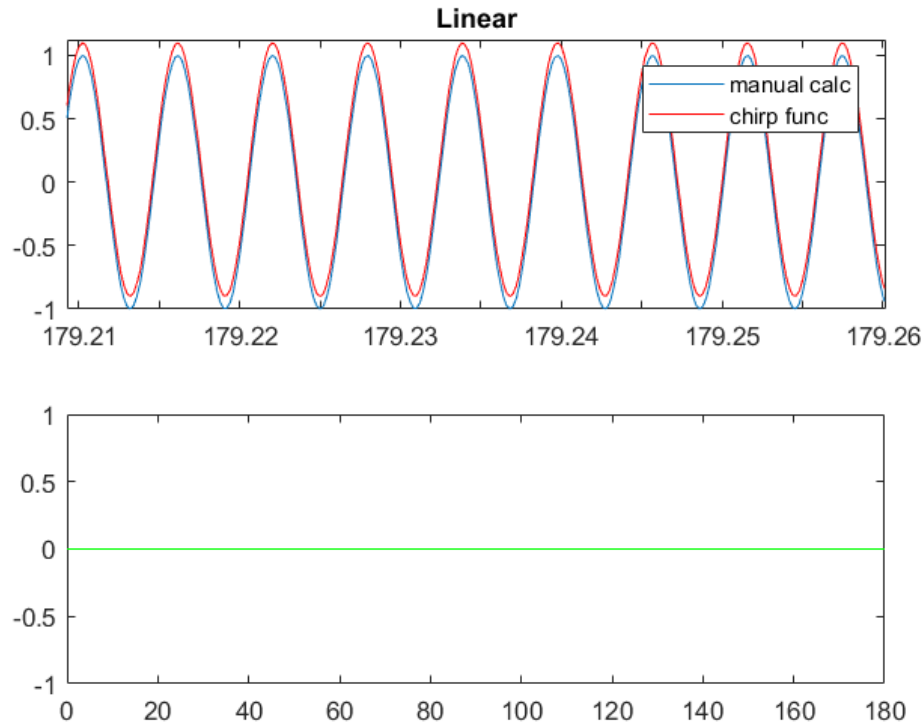


Figure 5 – Zoomed - Linear Calculated vs. Chirp Function (MATLAB) and Difference

nCode GlyphWorks

For the logarithmic sweep, Figure 6 shows the difference between the calculated and “built-in” function. (Note that the function is plotted offset by 0.1 to show both signals.) The difference between the logarithmic sweeps is about 0.04 in magnitude. For the linear sweep, Figure 7 shows the difference between the calculated and “built-in” function. The difference between the linear sweeps is about 0.2 in magnitude. The greater difference between the calculated and “built-in” function is attributed to numerical round off error.

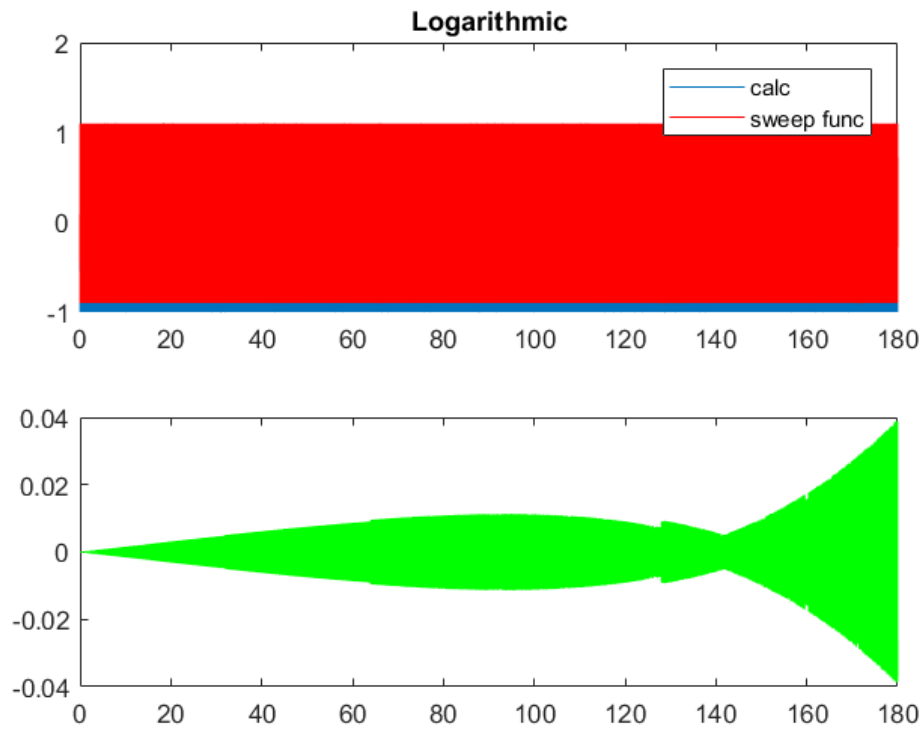


Figure 6 – Logarithmic Calculated vs. SweptSine Function (nCode) and Difference

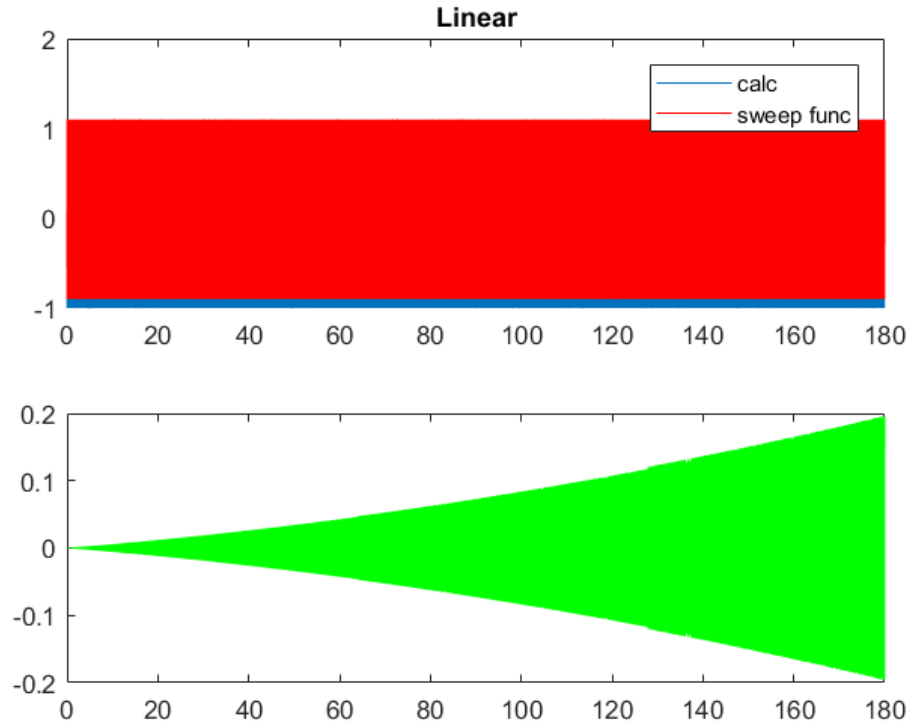


Figure 7 – Linear Calculated vs. SweptSine Function (nCode) and Difference

MATLAB demonstrates closer agreement between the calculated and “built-in” function for sine sweep/chirp functions and is the preferred computational platform.

===

The equations defined in the Linear Rate and Logarithmic Rate sections above will be used to define the discrete frequencies for the swept NB center frequencies. From the example in [4] we obtain information regarding the sweep from the specification below.

The sweep rate should be within the range of one-half to one octave per minute. The minimum 2 sweeps is one sweep up the bandwidth, followed by one sweep down the bandwidth. When the center frequency of the first narrowband, f_1 , is at its lowest frequency, the lower frequency band edge of this narrowband, and the lower band edge of the wideband 0.015 g^2/Hz level (for the vertical axis) coincide at 20 Hz. When the center frequency of the third narrowband, f_3 , is at its highest frequency, the upper band edge of this narrowband, and the upper band edge of the wideband 0.015 g^2/Hz level (for the vertical axis), coincide at 510 Hz

In summary, because the sweep rate is specified in octaves per minute this is a logarithmic sweep. The sweep rate is limited (we choose 0.75 octave/minute), and the center frequency is midway between the lower and upper NB frequencies and is swept over the complete range as specified. The logarithmic rate could be specified as in equation 15 if the total time was provided. Because the rate is specified the total time will be calculated. For the examples we assume the same total time for the linear rate.

The sweep rate is equation 13 divided by the sweep time interval. The number of discrete increments within the band is arbitrary, but the higher the number the closer approximation to a continuous sweep will be achieved.

The octaves/increments, total sweep time (up or down), and sweep time/increments are defined as

$$\frac{\text{octave}}{\text{increments}} = \frac{n}{\text{increments}} \quad (18)$$

$$\text{total sweep time} = \frac{n}{\text{sweep rate}} \quad (19)$$

$$t_{inc} = \frac{\text{total sweep time}}{\text{increments}} = \frac{\left(\frac{n}{\text{sweep rate}}\right)}{\text{increments}} \quad (20)$$

using equation 13. For each increment, the center frequency of each NB is incremented as if the upper frequency, which then becomes the new center frequency as in Table 1.

Table 1 – Swept NB Center Frequencies

	Linear	Logarithmic
Discrete	$f_{c\_new} = f_{c\_current} + R_{lin}(incr + 1)t_{inc}$	$f_{c\_new} = f_{c\_current} 2^{R_{log}(incr+1)t_{inc}}$
Continuous	$f_{c\_new} = f_{c\_current} + R_{lin}t$	$f_{c\_new} = f_{c\_current} 2^{R_{log}t}$

NOTE: increments start at zero

NOTE: the total sweep time includes the initial center frequency (as the increment starts at 0) but not the last center frequency (because we are dividing the BW by the number of increments). To include a t_{inc} time period for the ending center frequency that amount should be added to the total sweep time. These calculations are outlined in Appendix A.5 with examples below.

eg `frq_sweep_lin.m` (prefix = 'b4', sweep_type = 'lin')

```
PARAMETERS:
    prefix = b4 (also dataset name 'ds_x.mat')
    sweep_type = lin

Press any key to continue.

fc start, fc end (all in Hz)
    22.50000      167.50000
    45.00000      335.00000
    67.50000      502.50000

octave/incr, sweep total time (sec), sweep time/incr (sec) [includes initial incr]
    0.14481      231.69314      11.58466
    0.14481      231.69314      11.58466
    0.14481      231.69314      11.58466

Rlin (Hz/sec)
    0.62583
    1.25166
    1.87748

t_inc (sec)
    11.58466
    11.58466
    11.58466

Increments = 20

sweep_type = lin

incr, fc1, fc2, fc3 (all in Hz)
    0      22.50000      45.00000      67.50000
    1      29.75000      59.50000      89.25000
    2      37.00000      74.00000      111.00000
    3      44.25000      88.50000      132.75000
    4      51.50000      103.00000     154.50000
    5      58.75000      117.50000     176.25000
    6      66.00000      132.00000     198.00000
    7      73.25000      146.50000     219.75000
    8      80.50000      161.00000     241.50000
    9      87.75000      175.50000     263.25000
    10     95.00000      190.00000     285.00000
    11     102.25000     204.50000     306.75000
    12     109.50000     219.00000     328.50000
    13     116.75000     233.50000     350.25000
    14     124.00000     248.00000     372.00000
    15     131.25000     262.50000     393.75000
    16     138.50000     277.00000     415.50000
    17     145.75000     291.50000     437.25000
    18     153.00000     306.00000     459.00000
    19     160.25000     320.50000     480.75000
    20     167.50000     335.00000     502.50000
```

eg `frq_sweep_lin.m` (prefix = 'b4', sweep_type = 'log')

```
PARAMETERS:
    prefix = b4 (also dataset name 'ds_x.mat')
    sweep_type = log

Press any key to continue.

fc start, fc end (all in Hz)
    22.50000      167.50000
    45.00000      335.00000
    67.50000      502.50000

octave/incr, sweep total time (sec), sweep time/incr (sec) [includes initial incr]
    0.14481      231.69314      11.58466
    0.14481      231.69314      11.58466
    0.14481      231.69314      11.58466

Rlog (octave/sec)
    0.01250
    0.01250
    0.01250

t_inc (sec)
    11.58466
    11.58466
    11.58466

Increments = 20

sweep_type = log

incr, fc1, fc2, fc3 (all in Hz)
    0      22.50000      45.00000      67.50000
    1      24.87563      49.75127      74.62690
    2      27.50209      55.00419      82.50628
    3      30.40586      60.81173      91.21759
    4      33.61623      67.23246      100.84868
    5      37.16555      74.33110      111.49666
    6      41.08963      82.17926      123.26888
    7      45.42802      90.85604      136.28406
    8      50.22448      100.44896      150.67344
    9      55.52736      111.05473      166.58209
    10     61.39015      122.78029      184.17044
    11     67.87194      135.74389      203.61583
    12     75.03811      150.07622      225.11433
    13     82.96091      165.92182      248.88273
    14     91.72023      183.44046      275.16068
    15     101.40439      202.80877      304.21316
    16     112.11103      224.22207      336.33310
    17     123.94813      247.89625      371.84438
    18     137.03502      274.07005      411.10507
    19     151.50369      303.00737      454.51106
    20     167.50000      335.00000      502.50000
```

REFERENCE

- [1] Chirp Function (<https://en.wikipedia.org/wiki/Chirp#Definitions>)
- [2] MATLAB (www.mathworks.com)
- [3] nCode GlyphWorks (www.ncode.com) now (<https://www.hbkworld.com/en/products/software/analysis-simulation/durability/ncode-glyphworks-signal-processing-and-durability-analysis>)
- [4] AECTP-400 (Ed. 3, 3 Jan 2006), Allied Environmental Conditions and Tests Publication, Mechanical Environmental Tests. (Released to the Public). Annex B – Tracked Vehicle Vibration – Guidance for Initial Test Severity. Figure B-2. Materiel in Turret Bustle Rack or Installed In Turret, pg. 38. Figure B-4. Light Vehicle – Materiel on Sponson or Installed in Hull Figure, pg. 44
- [4a] NATO AECTP-400 - Mechanical Environmental Tests, Ed. D V1 (Nov 2019). Default Vibration Severities (Tracked Vehicles) in Annex A-26 (pg.A-36) to Annex A-33 (pg.A-43). Per C-M(2002)60 Information releasable to the Public (it shall carry no markings of any sort) see (<https://nso.nato.int/nso/nsdd/main/standards>, search for AECTP).
- [5] m+p VibControl Mixed Mode Testing (www.mpihome.com)
- [6] Octave Band (https://en.wikipedia.org/wiki/Octave_band)

- [7] Octave (<https://en.wikipedia.org/wiki/Octave>)
- [8] Integral b^x (<http://www.math.com/tables/integrals/more/b%5Ex.htm>)

APPENDIX A.1 - <https://en.wikipedia.org/wiki/Chirp#Definitions>

Definitions [edit]

If a **waveform** is defined as:

$$x(t) = \sin(\phi(t))$$

then the **instantaneous angular frequency**, ω , is defined as the phase rate as given by the first derivative of phase, with the instantaneous ordinary frequency, f , being its normalized version:

$$\omega(t) = \frac{d\phi(t)}{dt}, \quad f(t) = \frac{\omega(t)}{2\pi}$$

Finally, the **instantaneous angular chirpyness**, γ , is defined to be the second derivative of instantaneous phase or the first derivative of instantaneous angular frequency, with the **instantaneous ordinary chirpyness**, c , being its normalized version:

$$\gamma(t) = \frac{d^2\phi(t)}{dt^2} = \frac{d\omega(t)}{dt}, \quad c(t) = \frac{\gamma(t)}{2\pi} = \frac{df}{dt}$$

Thus chirpyness is the rate of change of the instantaneous frequency.^[2]

Types [edit]

Linear [edit]

In a **linear-frequency chirp** or simply **linear chirp**, the instantaneous frequency $f(t)$ varies exactly linearly with time:

$$f(t) = ct + f_0,$$

where f_0 is the starting frequency (at time $t = 0$), and c is the chirpyness, assumed constant:

$$c = \frac{f_1 - f_0}{T},$$

where f_1 is the final frequency; T is the time it takes to sweep from f_0 to f_1 .

The corresponding time-domain function for the **phase** of any oscillating signal is the integral of the frequency function, as one expects the phase to grow like $\phi(t + \Delta t) \simeq \phi(t) + 2\pi f(t) \Delta t$, i.e., that the derivative of the phase is the angular frequency $\phi'(t) = 2\pi f(t)$.

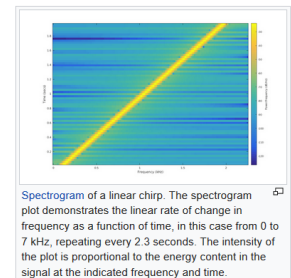
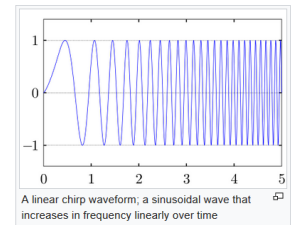
For the linear chirp, this results in:

$$\begin{aligned} \phi(t) &= \phi_0 + 2\pi \int_0^t f(\tau) d\tau \\ &= \phi_0 + 2\pi \int_0^t (c\tau + f_0) d\tau \\ &= \phi_0 + 2\pi \left(\frac{c}{2} t^2 + f_0 t \right), \end{aligned}$$

where ϕ_0 is the initial phase (at time $t = 0$). Thus this is also called a **quadratic-phase signal**.^[3]

The corresponding time-domain function for a **sinusoidal** linear chirp is the sine of the phase in radians:

$$x(t) = \sin \left[\phi_0 + 2\pi \left(\frac{c}{2} t^2 + f_0 t \right) \right]$$



Exponential [[edit](#)]

In a **geometric chirp**, also called an **exponential chirp**, the frequency of the signal varies with a **geometric** relationship over time. In other words, if two points in the waveform are chosen, t_1 and t_2 , and the time interval between them $t_2 - t_1$ is kept constant, the frequency ratio $f(t_2) / f(t_1)$ will also be constant.

In an exponential chirp, the frequency of the signal varies **exponentially** as a function of time:

$$f(t) = f_0 k^t$$

where f_0 is the starting frequency (at $t = 0$), and k is the rate of **exponential change** in frequency. Unlike the linear chirp, which has a constant chirpiness, an exponential chirp has an exponentially increasing frequency rate.

$$k = \left(\frac{f_1}{f_0} \right)^{\frac{1}{\tau}}$$

The corresponding time-domain function for the **phase** of an exponential chirp is the integral of the frequency:

$$\begin{aligned} \phi(t) &= \phi_0 + 2\pi \int_0^t f(\tau) d\tau \\ &= \phi_0 + 2\pi f_0 \int_0^t k^\tau d\tau \\ &= \phi_0 + 2\pi f_0 \left(\frac{k^{(t)} - 1}{\ln(k)} \right) \end{aligned}$$

where ϕ_0 is the initial phase (at $t = 0$).

The corresponding time-domain function for a sinusoidal exponential chirp is the sine of the phase in radians:

$$x(t) = \sin \left[\phi_0 + 2\pi f_0 \left(\frac{k^{(t)} - 1}{\ln(k)} \right) \right]$$

As was the case for the Linear Chirp, the instantaneous frequency of the Exponential Chirp consists of the fundamental frequency $f(t) = f_0 k^t$ accompanied by additional **harmonics**.^{[[citation needed](#)]}

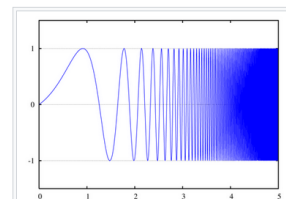
Generation [[edit](#)]

A chirp signal can be generated with **analog circuitry** via a **voltage-controlled oscillator** (VCO), and a linearly or exponentially ramping control **voltage**. It can also be generated **digitally** by a **digital signal processor** (DSP) and **digital to analog converter** (DAC), using a **direct digital synthesizer** (DDS) and by varying the step in the numerically controlled oscillator. It can also be generated by a YIG oscillator.^{[[clarification needed](#)]}

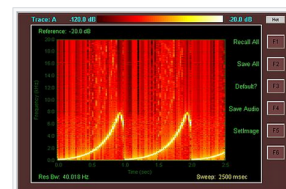
Relation to an impulse signal [[edit](#)]

A chirp signal shares the same spectral content with an **impulse signal**. However, unlike in the impulse signal, spectral components of the chirp signal have different phases,^{[4][5][6]} i.e., their power spectra are alike but the **phase spectra** are distinct. **Dispersion** of a signal propagation medium may result in unintentional conversion of impulse signals into chirps. On the other hand, many practical applications, such as **chirped pulse amplifiers** or echolocation systems,^[6] use chirp signals instead of impulses because of their inherently lower **PAPR**.

Problems playing this file? See [media help](#).



An exponential chirp waveform, a sinusoidal wave that increases in frequency exponentially over time



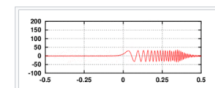
Spectrogram of an exponential chirp. The exponential rate of change of frequency is shown as a function of time, in this case from nearly 0 up to 8 kHz repeating every second. Also visible in this spectrogram is a frequency fallback to 6 kHz after peaking, likely an artifact of the specific method employed to generate the waveform.

Exponential chirp

0:00 ⏮ ⏪ ⏩ ⏭ MENU

Sound example for exponential chirp (five repetitions)

[Problems playing this file? See media help.](#)



APPENDIX A.2 – Integral b^x [8]

Proof: Integral b^x
(Math | Calculus | Integrals | Table Of | b^x)

Discussion of $b^x dx = b^x / \ln(b) + C$

1. Proof

Strategy: Use the $\int e^u du = e^u + C$

since $e^{\ln b} = b$,

$$\int b^x dx = \int [(e^{\ln b})^x] dx$$

$$= \int e^{(\ln b)x} dx$$

set $u = (\ln b)x$
then $du = (\ln b) dx$
substitute...

$$= \int e^u (du / \ln b)$$

$$= (1 / \ln b) \int e^u du$$

solve the integral...

$$= (1 / \ln b) (e^u + C)$$

$$= (1 / \ln b) e^u + C_2 \quad (\text{create new constant})$$

substitute back $u = (\ln b)x$,

$$= (1 / \ln b) e^{(\ln b)x} + C_2$$

$$= (1 / \ln b) (e^{\ln b})^x + C_2$$

$$= (1 / \ln b) b^x + C_2$$

$$= b^x / \ln b + C_2$$

Q.E.D.

See also the proof of $\int e^u du = e^u$. [PROOF](#)

2. You need not memorize this theorem. Derive it each time you use it.

Consider this example: if you have the integral:

$$\int 2^x dx.$$

There is no need to memorize the formula. We will get this integral into the easier form, $\int e^u du$.

This site uses cookies, including third-party cookies, to deliver its services, to personalize ads and to analyze traffic. By continuing to use this site, you agree to its use of cookies. [More info](#)

Got it!

APPENDIX A.3 – MATLAB [2] chirptest.m

```

% MATLAB '9.3.0.713579 (R2017b)'

% chirptest.m

step = 1/5000; % sample rate (sec)
t2 = 180; % total time (sec)
f1 = 20.0; % fmin (Hz)
f2 = 170.0; % fmax (Hz)

t = [0:step:t2]';

%          logarithmic      sine      sweep/exponential      chirp      -      see
% (https://en.wikipedia.org/wiki/Chirp#Definitions)
k = (f2/f1)^(1/t2);
iflog = f1*k.^t; % instantaneous freq (flog = integral of iflog)
flog = f1*(k.^t - 1)/log(k); % log = natural log (ln) in MATLAB
xlog = cos(2*pi.*flog); % MATLAB chirp is based on cosine
ylog = chirp(t,f1,t2,f2,'logarithmic');

% linear sine sweep/linear chirp - see (https://en.wikipedia.org/wiki/Chirp#Definitions)
k = (f2-f1)/t2;
iflin = f1+k*t; % instantaneous freq (flin = integral of iflin)
flin = f1.*t+(k.*t.^2)/2;
xlin = cos(2*pi.*flin); % MATLAB chirp is based on consine
ylin = chirp(t,f1,t2,f2,'linear');

% Results

% frequency log vs lin
figure(1)
clf
plot(t,iflog) % logarithmic
hold on
plot(t,iflin) % linear
legend('flog','flin')
title('Instantaneous Frequency'); % compare frequencies between linear/logarithmic

% compare logarithmic manual calc vs chirp function
figure(2)
clf
subplot(2,1,1),plot(t,xlog)
subplot(2,1,1),hold on
subplot(2,1,1),plot(t,ylog+0.1,'r')
subplot(2,1,1),legend('manual calc','chirp func')
subplot(2,1,1),title('Logarithmic'); % compare manual calculation vs chirp function
(logarithmic)
subplot(2,1,2),plot(t,xlog-ylog,'g'); % difference

% compare linear manual calc vs chirp function
figure(3)
clf
subplot(2,1,1),plot(t,xlin)
subplot(2,1,1),hold on
subplot(2,1,1),plot(t,ylin+0.1,'r')
subplot(2,1,1),legend('manual calc','chirp func')
subplot(2,1,1),title('Linear'); % compare manual calculation vs chirp function (linear)
subplot(2,1,2),plot(t,xlin-ylin,'g'); % difference

%y=dlmread('..\..\work_psd_NB_sweep\sum.out');
%figure(1)
%plot(sum(:,1),'k');
%plot(sum(:,2),'k');
%plot(sum(:,3),'k');

```

APPENDIX A.4 – nCode GlyphWorks [3] Flow

Glyph Outline

TS Generator (SweptSine/Linear)
to XY Display (linear output)
to TS Output (lin_calc)

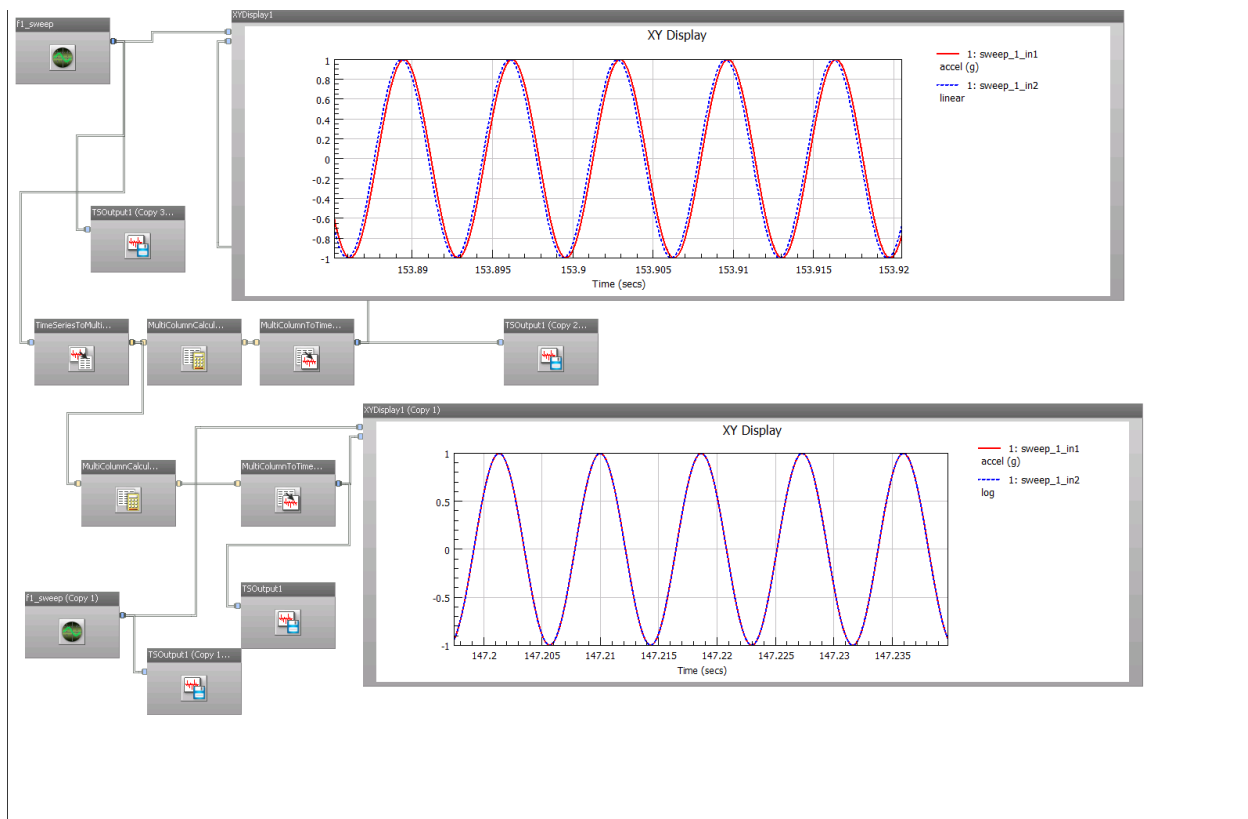
to TimeSeriesToMultiColumn (base calculation of same time column)
to MultiColumnCalculator
($\text{linear} = \sin(2\pi * (20 * \text{Column1} + ((170 - 20) / (2 * 180)) * \text{Column1}^2))$)

to MultiColumnToTimeSeries
to XY Display (linear output above)
to TS Output (lin_func)

to MultiColumnCalculator
($\text{logarithmic} = \sin(2\pi * 20 * 180 * (((170 / 20) ^ (\text{Column1} / 180) - 1) / (\log(170 / 20))))$)

to MultiColumnToTimeSeries
to XY Display (logarithmic output above)
to TS Output (log_calc)

TS Generator (SweptSine/Logarithmic)
to XY Display (logarithmic output above)
to TS Output (log_func)



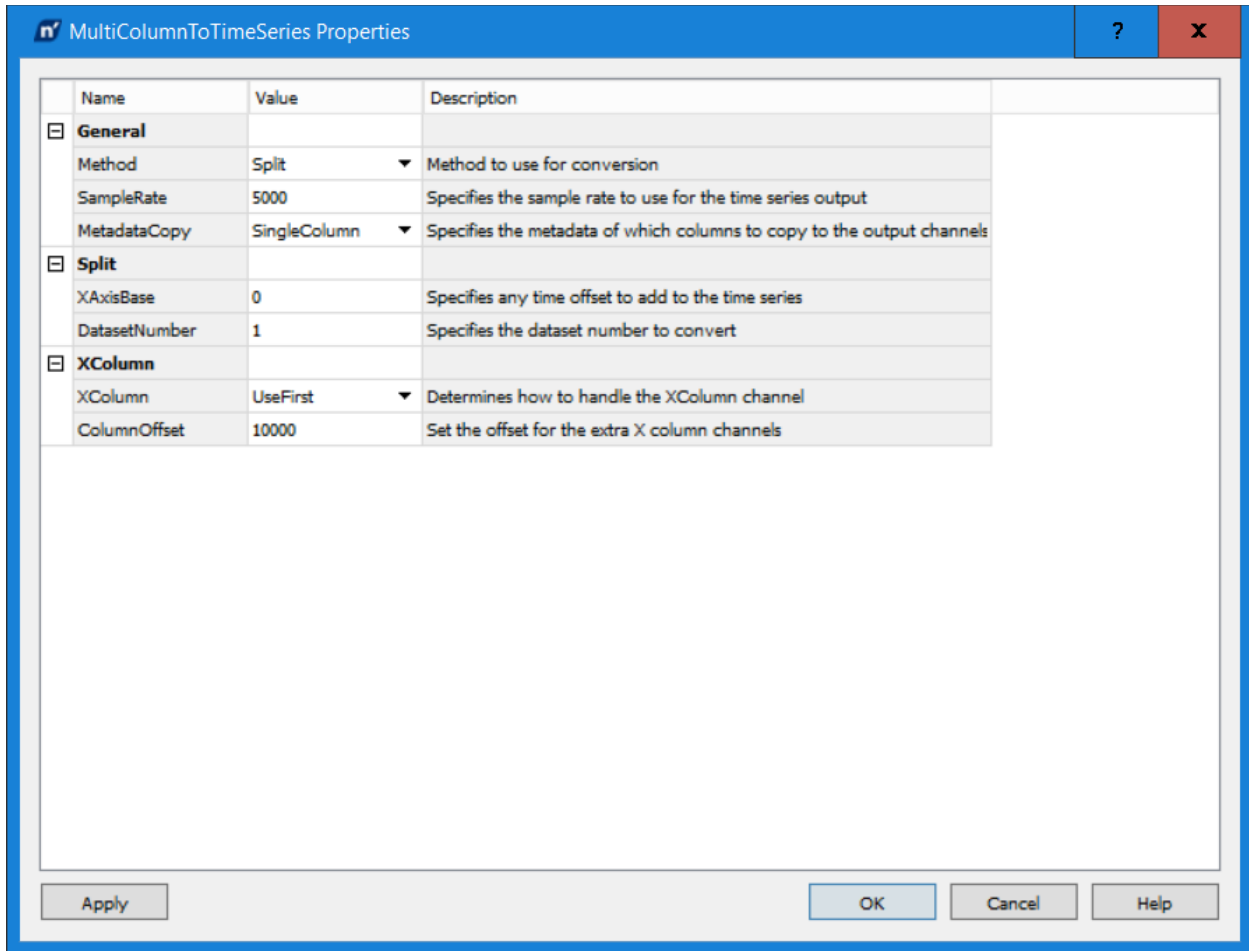
Flow Diagram

TSGenerator Properties ? X

Name	Value	Description
General		
Operation	SweptSine ▼	Determines the wave type to generate
TestName	sweep_1	The test name of the output data
TotalTime	180	The total time to generate in the output signal, in seconds
SampleRate	5000	The number of data points per second in the output signal
BaseTime	0	The X-axis value of the first data point in the output signal
Median	0	The median of the output signal
ChannelDetails		
ChannelNumber	1	The channel number of the output data
ChannelTitle	accel	The channel title of the output data
XTitle	Time	The X-axis title of the output channel
XUnits	secs	The X-axis units of the output channel
YTitle	accel	The Y-axis title of the output channel
YUnits	g	The Y-axis units of the output channel
SweptSine		
StartFrequency	20	The starting frequency of the output wave
StartAmplitude	1	The start amplitude of the output wave
Phase	0	The angle at which to start generating the output wave
EndFrequency	170	The ending frequency of the output swept sine wave
EndAmplitude	1	The end amplitude of the output swept sine wave
SweptSineOperation	Lin ▼	Whether to sweep the frequency linearly or logarithmically
Advanced		
RandomNumberSeed	1	The seed to use to initialise the random number generator

Apply OK Cancel Help

SweptSine/Linear Properties



The image shows a software dialog box titled "MultiColumnToTimeSeries Properties". It features a blue title bar with a question mark icon and a close button (X). The main content area contains a table with three columns: "Name", "Value", and "Description". The table is organized into three sections: "General", "Split", and "XColumn", each with a collapse/expand icon. The "General" section includes "Method" (Split), "SampleRate" (5000), and "MetadataCopy" (SingleColumn). The "Split" section includes "XAxisBase" (0) and "DatasetNumber" (1). The "XColumn" section includes "XColumn" (UseFirst) and "ColumnOffset" (10000). At the bottom of the dialog are four buttons: "Apply", "OK", "Cancel", and "Help".

Name	Value	Description
General		
Method	Split	Method to use for conversion
SampleRate	5000	Specifies the sample rate to use for the time series output
MetadataCopy	SingleColumn	Specifies the metadata of which columns to copy to the output channels
Split		
XAxisBase	0	Specifies any time offset to add to the time series
DatasetNumber	1	Specifies the dataset number to convert
XColumn		
XColumn	UseFirst	Determines how to handle the XColumn channel
ColumnOffset	10000	Set the offset for the extra X column channels

Apply OK Cancel Help

MultiColumnToTimeSeries Properties

TSGenerator Properties ? X

Name	Value	Description
General		
Operation	SweptSine ▼	Determines the wave type to generate
TestName	sweep_1	The test name of the output data
TotalTime	180	The total time to generate in the output signal, in seconds
SampleRate	5000	The number of data points per second in the output signal
BaseTime	0	The X-axis value of the first data point in the output signal
Median	0	The median of the output signal
ChannelDetails		
ChannelNumber	1	The channel number of the output data
ChannelTitle	accel	The channel title of the output data
XTitle	Time	The X-axis title of the output channel
XUnits	secs	The X-axis units of the output channel
YTitle	accel	The Y-axis title of the output channel
YUnits	g	The Y-axis units of the output channel
SweptSine		
StartFrequency	20	The starting frequency of the output wave
StartAmplitude	1	The start amplitude of the output wave
Phase	0	The angle at which to start generating the output wave
EndFrequency	170	The ending frequency of the output swept sine wave
EndAmplitude	1	The end amplitude of the output swept sine wave
SweptSineOperation	Log ▼	Whether to sweep the frequency linearly or logarithmically
Advanced		
RandomNumberSeed	1	The seed to use to initialise the random number generator

Apply OK Cancel Help

SweptSine/Logarithmic Properties

APPENDIX A.5 – MATLAB [2] eg_frq_sweep.m

eg_frq_sweep.m

```

% MATLAB '9.3.0.713579 (R2017b)'

% eg_frq_sweep.m

% check if parameters exist
if exist('prefix') & exist('sweep_type')
    fprintf('PARAMETERS:\n');
    fprintf('    prefix = %s (also dataset name ''ds_x.mat'')\n',prefix);
    fprintf('    sweep_type = %s\n',sweep_type);
    fprintf('\nPress any key to continue.\n');
    fprintf('\n');
    pause
else
    fprintf('Define parameters: prefix='b2/b4'', sweep_type='lin/log' before running.\n');
    stop
end

load( ['ds_' prefix '.mat'] ); % load dataset (see create_ds.m NB1, NB2, NB3)

load('ds_param.mat');

% calc starting/ending center frequencies considering BW and NB levels
fc_lh = [NB1(2) + NB1(1)/2, NB1(3) - NB1(1)/2;
         NB2(2) + NB2(1)/2, NB2(3) - NB2(1)/2;
         NB3(2) + NB3(1)/2, NB3(3) - NB3(1)/2];

% calc octave/increment, sweep time total (sec) up or dn, sweep time/increment (sec)
fc_oct = [log2(fc_lh(1,2)/fc_lh(1,1))/increments, log2(fc_lh(1,2)/fc_lh(1,1))/sweep_rate, ...
          log2(fc_lh(1,2)/fc_lh(1,1))/sweep_rate/increments;
          log2(fc_lh(2,2)/fc_lh(2,1))/increments, log2(fc_lh(2,2)/fc_lh(2,1))/sweep_rate, ...
          log2(fc_lh(2,2)/fc_lh(2,1))/sweep_rate/increments;
          log2(fc_lh(3,2)/fc_lh(3,1))/increments, log2(fc_lh(3,2)/fc_lh(3,1))/sweep_rate, ...
          log2(fc_lh(3,2)/fc_lh(3,1))/sweep_rate/increments];

%Rlog = [log2(fc_lh(1,2)/fc_lh(1,1))/fc_oct(1,2), log2(fc_lh(2,2)/fc_lh(2,1))/fc_oct(2,2), ...
         log2(fc_lh(3,2)/fc_lh(3,1))/fc_oct(3,2)];
Rlog = [sweep_rate, sweep_rate, sweep_rate]; % set as specified speed rate
t_inc = [fc_oct(1,3), fc_oct(2,3), fc_oct(3,3)];
Rlin = [(fc_lh(1,2)-fc_lh(1,1))/fc_oct(1,2), (fc_lh(2,2)-fc_lh(2,1))/fc_oct(2,2), (fc_lh(3,2)-fc_lh(3,1))/fc_oct(3,2)];

% start diary
diary eg1

fprintf('NB, BW, Low, High (all in Hz)\n');
fprintf('%5d %20.5f %20.5f %20.5f\n',1,NB1);
fprintf('%5d %20.5f %20.5f %20.5f\n',2,NB2);
fprintf('%5d %20.5f %20.5f %20.5f\n',3,NB3);
fprintf('\n');

fprintf('fc start, fc end (all in Hz)\n');
fprintf('%20.5f %20.5f\n',fc_lh);
fprintf('\n');

fprintf('octave/incr, sweep total time (sec), sweep time/incr (sec) [includes initial incr]\n');
fprintf('%20.5f %20.5f %20.5f\n',fc_oct);
fprintf('\n');

if strcmp(sweep_type,'log'),
    fprintf('Rlog (octave/sec)\n')
    fprintf('%20.5f\n',Rlog);
    fprintf('\n');
else
    fprintf('Rlin (Hz/sec)\n');
    fprintf('%20.5f\n',Rlin);
    fprintf('\n');
end

fprintf('t_inc (sec)\n');
fprintf('%20.5f\n',t_inc);
fprintf('\n');

fprintf('Increments = %d\n',increments);
fprintf('\n');

fprintf('sweep_type = %s\n',sweep_type);
fprintf('\n');

fc0=[fc_lh(1,1), fc_lh(2,1), fc_lh(3,1)];
fc=fc0;

```

```

% do increments, recalc fc after each
fprintf('incr, fc1, fc2, fc3 (all in Hz)\n');
for j = 0:increments,
    % update fc for next increment
    fprintf('%5d %20.5f %20.5f %20.5f\n',j,fc);

    % update fc for next increment
    if strcmp(sweep_type,'log'),
        fc = fc0.*2.^(Rlog*(j+1).*t_inc);
    else
        fc = fc0+(Rlin*(j+1).*t_inc);
    end
end
diary off

```

create_ds.m

```

% MATLAB '9.3.0.713579 (R2017b)'

clear all
%===DATA ACETP-400 Annex B B-4, 45 min/axis = 1600 km (994 mi) from 4 hr/axis test w/equiv factor
%***CHANGE***
% ===WB
% Frequency (Hz), Amplitude (G^2/Hz)
% vertical
WB_V = [5, 0.001;
20, 0.015;
510, 0.015;
2000, 0.001]; % 3.56 GRMS

% transverse
WB_T = [5, 0.001;
20, 0.010;
510, 0.010;
2000, 0.001]; % 3.03 GRMS

% longitudinal
WB_L = [5, 0.001;
20, 0.010;
510, 0.010;
2000, 0.001]; % 3.03 GRMS

%===NB
% Bandwidth (Hz), Sweep Range (Low, High), Amplitude (G^2/Hz)(vert, transverse, longitudinal)
NB1 = [5, 20, 170, 1.0, 1.0, 1.0]; % WB+NB1 = 5.93 GRMS
NB2 = [10, 40, 340, 1.0, 0.50, 0.50]; % WB+NB2 = 4.79 GRMS
NB3 = [15, 60, 510, 0.5, 0.25, 0.25]; % WB+NB3 = 4.79 GRMS

%=====%
save ds_b4.mat

%=====

clear all
%===PARAMETERS

%set individually
%prefix
%sweep_type

% sweep.m
total_time = 45*60; % seconds
sweep_rate = 0.75/60; % octave/sec, sweep up/dn
increments = 20; % one-way, sweep up/dn
f_step = 0.1;
freq = [5:f_step:2000]'; % Hz
n_channels = 3; % 1=vertical, 2=transverse, 3=longitudinal
bad_val = 99; % replace for log interp
ddir = './dat\'';

exposure_time = 11.033; % secs (see psd1ch.m or psd3ch.m)
olchdir = './dat_psd1\''; % psd1ch.m - 1 channel per file
o3chdir = './dat_psd3\''; % psd3ch.m - 3 channel per file

% show.m - increments, n_channels, ddir
% frqchk.m - increments, n_channels, ddir
% psd1ch.m - increments, n_channels, ddir, exposure_time, olchdir
% psd3ch.m - increments, n_channels, ddir, exposure_time, o3chdir
% ldf1ch.m - (suffix), total_time, exposure_time, olchdir
% ldf3ch.m - total_time, exposure_time, o3chdir
% eg_frq_sweep.m - sweep_rate, increments
%=====%
save ds_param.mat

```

TN 19-0801-01

FUNCTIONS FOR GENERATING POWER SPECTRAL DENSITY (PSD) INPUTS FOR FE-SAFE FREQUENCY BASED FATIGUE ANALYSIS

August 1, 2019

W. Bylsma

U.S. Army Combat Capabilities Development Command (DEVCOM)
Ground Vehicle Systems Center (GVSC)
Detroit Arsenal
Systems Integration & Engineering
ATTN: FCDD-GVS-IES/MS 157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

fe-safe [3] frequency based fatigue analysis requires Power Spectral Density (PSD) inputs. For the vibration specification [4], shown in Figure 1a and 1b, how can a PSD be created for use with fe-safe?

AECTP 400
(Edition 3)
ANNEX B 401
FIGURE B-2 MATERIEL IN TURRET BUSTLE RACK OR INSTALLED IN TURRET

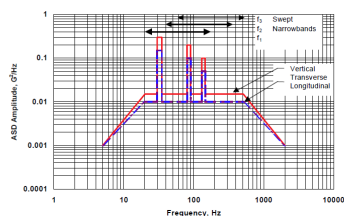


Figure B-2 Tracked Vehicle Schedule Breakpoints									
Wideband Random Spectrum				Harmonic Swept Narrowbands					
Frequency, Hz	Axis, ASD Amplitude G ² /Hz			Narrowband	f ₁	f ₂	f ₃		
	V	T	L	Bandwidth	5	10	15		
5	0.001	0.001	0.001	Sweep BW, Hz	20	40	60		
20	0.015	0.010	0.010	# Sweeps	170	340	510		
510	0.015	0.010	0.010		2	2	2		
2000	0.001	0.001	0.001						
				Axis	ASD Amplitude, G ² /Hz				
Wideband Grms				Vertical	0.30	0.20	0.10		
Total Grms				Transverse	0.15	0.10	0.05		
				Longitudinal	0.15	0.10	0.05		

38

ORIGINAL

(a)

AECTP 400
(Edition 3)
ANNEX B 401
FIGURE B-4 LIGHT VEHICLE - MATERIEL ON SPONSON OR INSTALLED IN HULL

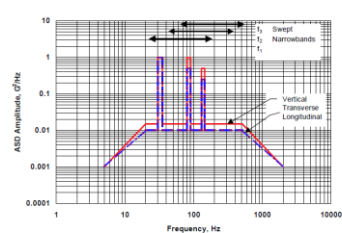


Figure B-4 Tracked Vehicle Schedule Breakpoints									
Wideband Random Spectrum				Harmonic Swept Narrowbands					
Frequency, Hz	Axis, ASD Amplitude, G ² /Hz			Narrowband	f ₁	f ₂	f ₃		
	V	T	L	Bandwidth, Hz	5	10	15		
5	0.001	0.001	0.001	Sweep BW, Hz	20	40	60		
20	0.015	0.010	0.010	# Sweeps	2	2	2		
510	0.015	0.010	0.010						
2000	0.001	0.001	0.001						
				Axis	ASD Amplitude, G ² /Hz				
Wideband Grms				Vertical	1.00	1.00	0.50		
Total Grms				Transverse	1.00	0.50	0.25		
				Longitudinal	1.00	0.50	0.25		

44

ORIGINAL

(b)

Figure 1a – “B2” Vibration Profile (20-60 ton tracked vehicle)
Figure 1b – “B4” Vibration Profile (25 ton “lightweight” tracked vehicle)

PROBLEM SOLUTION:

Review of the vibration profiles shown in Figure 1 from [4] provide their characteristics summarized below:

- Vibration profiles

- swept narrowband random on broadband random
- “b2” = Figure B-2. Materiel in Turret Bustle Rack or Installed in Turret, pg. 38 (20-60 ton tracked vehicle)
- “b4” = Figure B-4. Light Vehicle – Materiel on Sponson or Installed in Hull Figure, pg. 44 (25 ton “lightweight” tracked vehicle)
- (45 min = 1600 km)
- vertical, transverse, and longitudinal (independently)
- minimum 2 sweeps required
- sweep rate = 0.75 octaves/min
- 1 sweep (up or dn) = 2.89 octaves, 4.054 min (20 increments)
- 11.1 sweeps in 45 min

fe-safe uses PSD (.psd) files that are referenced by Load Definition Files (LDF)(.ldf) to determine the loading situation for fatigue analysis. These formats are defined in the User's Guide [3] under “27 Fatigue Analysis Using PSD data\27.3.2 File format” and “13 Defining fatigue loadings\13.9 The Load Definition (LDF) file”, respectively. When the LDF references multiple PSD files the frequency values per “.psd” file must be identical and match those specified in other “.psd” files. Reducing the number of points reduces the file size and the analysis time. From this stand point we investigate functions to eliminate duplicate values using MATLAB [5] to demonstrate the computations. They are listed below and included in Appendix A.1.

- u2x.m - remove more than 2 duplicate values in a vector
- u2xy.m - remove more than 2 duplicate values in vector y (and associated vector x)
- lin2xy.m - keep points of linear segment in vector y (and associated vector x), specify tolerance (if reading this output it should be written with high precision, e.g. %30.20E etc.)

Figure 2 shows the results from using lin2xy.m (results for u2x.m and u2xy.m are included in Appendix A.1).

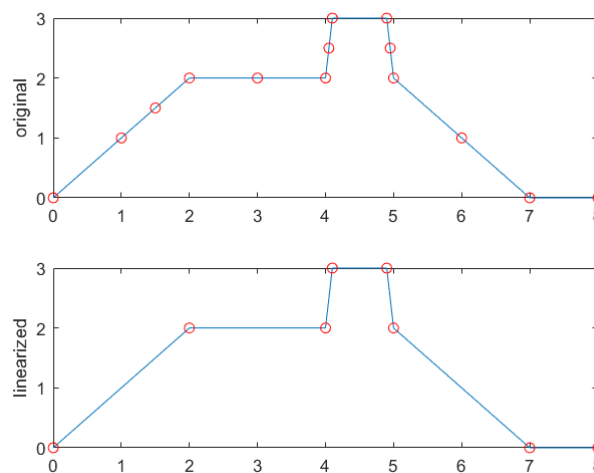


Figure 2 – lin2xy Linearization

The actual creation of the PSD data is accomplished with the scripts listed below and included in Appendix A.2.

- sweep.m - generate wideband (WB) and narrowband (NB) psd data files (vert, trans, long) for "prefix" dataset and parameters see create_ds.m, linear or logarithmic sweep, output .txt files

- show.m - plot psd data files (vert, trans, long) generated by sweep.m, for parameters see create_ds.m
- create_ds.m - create datasets and parameter sets

From the local script directory the following directory structure is defined for the output data,

```

.\ (scripts directory)
.\dat (psd data directory)
    [prefix]_freq_[sweep_type].txt (e.g. b2_freq_log.txt)
    [prefix]_psd_[suffix]_[increment]_[sweep_type].txt (e.g. b4_psd_v_18_lin.txt)

.\dat\dat_psd1 (1 channel psd and ldf files)
    [prefix]_psd_[suffix]_[increment]_[sweep_type].psd (e.g. b4_psd_1_00_log.psd)
    [prefix]_psd_[suffix]_wb_[sweep_type].psd (e.g. b4_psd_v_wb_log.psd)
    [prefix]_psd_[suffix]_[cycle length]_[sweep_type].ldf (e.g. b4_psd_1_22_lin.ldf)

.\dat\dat_psd3 (3 channel psd and ldf files)
    [prefix]_psd3ch_[increment]_[sweep_type].psd (e.g. b2_psd3ch_16_log.psd)
    [prefix]_psd3ch_wb_[sweep_type].psd (e.g. b2_psd3ch_wb_log.psd)
    [prefix]_psd3ch_[cycle length]_[sweep_type].ldf (e.g. b2_psd3ch_22_log.ldf)

```

but can be changed in create_ds.m. The fundamentals of the sweep parameter calculations are defined in [2]. Below an outline of sweep.m is provided:

1. check if 'prefix', 'sweep_type' and 'frqchk' parameters exist. These are required to create the proper psd data.
2. load the data set defined by 'prefix'
3. load the parameters defined in create_ds.m
4. if frq_chk set then, depending upon 'sweep_type", read the linearized frequency data generated by frqchk.m (frqchk.m is discussed below. sweep.m must be run once without frq_chk set).
5. loop for all channels (Vertical, Transverse, and Longitudinal)
 - a. set input WB data for specified channel
 - b. set output file name
 - c. output WB data (interpolate using log then go back so linear on PSD plot as from specification)
 - d. calculate NB data
 - e. loop for 0 to increments, recalc center frequency after each
 - i. output NB data (envelop WB and NB, interpolate using log then go back so linear on PSD plot as from specification)
 - ii. update center frequency for next increment

After running sweep.m once, the frqchk.m script must be run to reduce the number of data points and yet ensure the frequency values per ".psd" file are identical and match those specified in other ".psd" files. This is accomplished by using lin2xy.m on all psd data and then using only unique values. When sweep.m is run with the 'frqchk' flag set, it will read the unique frequencies in the output file created by frqchk.m for all psd data generated. Below is a description of frqchk.m (see Appendix A.2).

- frqchk.m - get all used psd frequency points after log-log linearize (lin2xy.m) and create frequency output file (change frq_chk=1 in sweep.m to use unique frequencies for all psd data)

Since the specification is linear on a log-log scale, the function lin2xy.m reduces duplicate values after converting to a log (base 10) scale and then taking only unique frequencies as described below.

```

...
[freq1, psd1] = lin2xy(log10(y(:,1)), log10(y(:,2)), 1e-6);
freq1 = 10.^freq1;
psd1 = 10.^psd1;
tmp = [tmp, freq1']; % accumulate all points
...
tmp = unique(sort(tmp));

```

The actual creation of the PSD and LDF files is accomplished with the scripts listed below and included in Appendix A.3.

- psd3ch.m - generate fe-safe WB and WB/NB psd files with (vertical, transverse, longitudinal) COMBINED (3 channels) from data created by sweep.m
- psd1ch.m - generate fe-safe WB and WB/NB psd files with (vertical, transverse, longitudinal) as INDIVIDUAL channels from data created by sweep.m
- ldf3ch.m - generate fe-safe LDF file for (vertical, transverse, long) COMBINED (3 channels) (last block uses WB only)
- ldf1ch.m - generate fe-safe LDF file for (vertical, transverse, longitudinal) INDIVIDUAL channels (last block uses WB only)

Example .psd and .ldf files are included in Appendix A.4. For 3 channel files, the order of channels follows the definition in the finite element analysis input file as shown in the example below.

e.g. ABAQUS Steady State Dynamics Input (.inp) File – 3 Channels – Vertical (DOF=3), Transverse (DOF=2), and Longitudinal (DOF=1)

```

...
*STEP, INC = 1000, NLGEOM = NO, PERTURBATION
steady_state_dynamics_v
*STEADY STATE DYNAMICS
0.1, 2000.0, 20, 3
*MODAL DAMPING, MODAL = DIRECT, DEFINITION = FREQUENCY RANGE , FIELD = ALL
0.1, 0.25
2000.0, 0.25
*BASE MOTION, TYPE=ACCELERATION, DOF=3, AMPLITUDE=PSDVAL, SCALE=1.0
*OUTPUT, HISTORY
*MODAL OUTPUT
GU, GPU
*END STEP
*STEP, INC = 1000, NLGEOM = NO, PERTURBATION
steady_state_dynamics_t
*STEADY STATE DYNAMICS
0.1, 2000.0, 20, 3
*MODAL DAMPING, MODAL = DIRECT, DEFINITION = FREQUENCY RANGE , FIELD = ALL
0.1, 0.25
2000.0, 0.25
*BASE MOTION, TYPE=ACCELERATION, DOF=2, AMPLITUDE=PSDVAL, SCALE=1.0
*OUTPUT, HISTORY
*MODAL OUTPUT
GU, GPU
*END STEP
*STEP, INC = 1000, NLGEOM = NO, PERTURBATION
steady_state_dynamics_l
*STEADY STATE DYNAMICS
0.1, 2000.0, 20, 3
*MODAL DAMPING, MODAL = DIRECT, DEFINITION = FREQUENCY RANGE , FIELD = ALL
0.1, 0.25
2000.0, 0.25
*BASE MOTION, TYPE=ACCELERATION, DOF=1, AMPLITUDE=PSDVAL, SCALE=1.0
*OUTPUT, HISTORY
*MODAL OUTPUT
GU, GPU
*END STEP

```

The use of the LDF file is not required. You can select the .psd files from the fe-safe GUI interface.

Many of the values found in create_ds.m are obtained from the vibration specification (e.g. WB and NB channel frequencies and amplitudes, total time each channel represents, and sweep rate). The number of increments (one-way, sweep up or down, twenty (20) in this case) is subjective, but the value should be

large enough to provide a good approximation to a continuous sweep. The exposure time is required for the .psd and .ldf files and can be found from equation 15 [2]

$$T = \frac{\log_2 \left(\frac{f_u}{f_l} \right)}{R_{log}} [sec] = \frac{2.896}{0.75/60} = 231.68 \quad (1)$$

while the increment time from equation 20 [2]

$$t_{inc} = \frac{T}{increments} [sec] = \frac{231.68}{20} = 11.584. \quad (2)$$

Appendix A.5 includes a test script to excise all options. Appendix A.6 includes help for MATLAB's interp1 function.

REFERENCE

- [1] Reference Internal Project 19-011 STRYKER BATTERY BOX STRUCTURAL ANALYSIS
- [2] TN19-0514-01 GENERATING SINE SWEEP/CHIRP SIGNALS
- [3] fe-safe 2017 (<https://www.3ds.com/products-services/simulia/products/fe-safe/>) [2019-07-31]
- [4] AECTP-400 (Ed. 3, 3 Jan 2006), Allied Environmental Conditions and Tests Publication, Mechanical Environmental Tests. (Released to the Public). Annex B – Tracked Vehicle Vibration – Guidance for Initial Test Severity. Figure B-2. Materiel in Turret Bustle Rack or Installed In Turret, pg. 38. Figure B-4. Light Vehicle – Materiel on Sponson or Installed in Hull Figure, pg. 44
- [4a] NATO AECTP-400 - Mechanical Environmental Tests, Ed. D V1 (Nov 2019). Default Vibration Severities (Tracked Vehicles) in Annex A-26 (pg.A-36) to Annex A-33 (pg.A-43). Per C-M(2002)60 Information releasable to the Public (it shall carry no markings of any sort) see (<https://nso.nato.int/nso/nsdd/main/standards>, search for AECTP).
- [5] MATLAB (<https://www.mathworks.com/>)
- [5a] 1-D data interpolation (table lookup) interp1 (<https://www.mathworks.com/help/matlab/ref/interp1.html>)

APPENDIX A.1 – DUPLICATE VALUE REMOVAL FUNCTIONS

APPENDIX A.2 – PSD SWEEP DATA GENERATION

APPENDIX A.3 – PSD AND LDF FILE GENERATION

APPENDIX A.4 – EXAMPLE PSD AND LDF FILE FORMATS

APPENDIX A.5 – TEST SCRIPT

APPENDIX A.6 – MATLAB INTERP1 HELP [5a]

APPENDIX A.1 – DUPLICATE VALUE REMOVAL FUNCTIONS

```

function ux = u2x(x)
% u2x(x) - remove more than 2 duplicate values (x is a col vector)

p = 2;

% parameters
n = length(x); % vector length
val = [x(1)]; % initial value
cnt = 1; % initial count
ux=[val]; % output vector

for i = 2:n
    if x(i) ~= val % if value changes place in output and reset val/cnt
        ux=[ux;x(i)];
        val = x(i);
        cnt = 1;
    else
        if cnt < p % if same value and cnt < p than output and increment cnt
            ux=[ux;val];
        end
        cnt = cnt + 1;
    end
end
end

```

eg u2x

```

ans =
'beg'
ans =
1      2      3      4      5      6      7
ans =
1      2      3      4      5      6      7
ans =
'end'
ans =
'beg'
ans =
Columns 1 through 8
1      1      2      2      3      3      4      4
Columns 9 through 14
5      5      6      6      7      7
ans =
Columns 1 through 8
1      1      2      2      3      3      4      4
Columns 9 through 14
5      5      6      6      7      7
ans =
'end'
ans =
'beg'
ans =
Columns 1 through 8
1      1      1      1      1      2      2      3
Columns 9 through 16
4      4      4      4      4      5      5      5
Column 17
6
ans =
Columns 1 through 8
1      1      2      2      3      4      4      5
Columns 9 through 10
5      6
ans =
'end'
ans =
'beg'
ans =
Columns 1 through 8
1      1      1      2      2      2      3      3
Columns 9 through 16
3      4      4      4      5      5      5      6
Columns 17 through 21
6      6      7      7      7
ans =
Columns 1 through 8
1      1      2      2      3      3      4      4
Columns 9 through 14
5      5      6      6      7      7
ans =
'end'
ans =
'beg'
ans =
Columns 1 through 8
1      1      1      1      1      1      2      2
Columns 9 through 16
2      2      2      2      5      3      3      3
Columns 17 through 24
3      3      4      4      4      5      5      5
Columns 25 through 30
6      6      6      7      7      7
ans =
Columns 1 through 8
1      1      2      2      5      3      3      4
Columns 9 through 15
4      5      5      6      6      7      7
ans =
'end'

```

```

function [ux, uy] = u2xy(x,y)
% u2xy(x,y) - remove more than 2 duplicate y values (x and y are col vector and the same length)

p = 2;

% parameters
n = length(y); % vector length

valy = [y(1)]; % initial value
valx = [x(1)];
cnt = 1; % initial count

uy=[valy]; % output vector
ux=[valx];

for i = 2:n
    if y(i) ~= valy % if value changes place in output and reset val/cnt
        if cnt >= p % output last valx/y
            uy=[uy;valy];
            ux=[ux;valx];
        end
        uy=[uy;y(i)];
        ux=[ux;x(i)];
        valy = y(i);
        valx = x(i);
        cnt = 1;
    else
        valy = y(i);
        valx = x(i);
        cnt = cnt + 1;
    end
end

if cnt >= p % output last valx/y
    uy=[uy;valy];
    ux=[ux;valx];
end

```

eg u2xy

```

ans =
'beg'
ans =
1 1
2 2
3 3
4 4
5 5
6 6
7 7
ans =
1 1
2 2
3 3
4 4
5 5
6 6
7 7
ans =
'end'
ans =
'beg'
ans =
1 1
2 1
3 2
4 2
5 3
6 3
7 4
8 4
9 5
10 5
11 6
12 6
13 7
14 7
ans =
1 1
2 1
3 2
4 2
5 3
6 3
7 4
8 4
9 5
10 5
11 6
12 6
13 7
14 7
ans =
'end'
ans =
'beg'
ans =
1 1
2 1
3 1
4 1
5 1
6 2
7 2
8 3
9 4
10 4
11 4
12 4
13 4
14 5
15 5
16 5
17 6
ans =
1 1
5 1
6 2
7 2
8 3
9 4
13 4
14 5
16 5
17 6
ans =
'end'

```



```

ans =
'beg'
ans =
1 1
2 1
3 1
4 2
5 2
6 2
7 3
8 3
9 3
10 4
11 4
12 4
13 5
14 5
15 5
16 6
17 6
18 6
19 7
20 7
21 7
ans =
1 1
3 1
4 2
6 2
7 3
9 3
10 4
12 4
13 5
15 5
16 6
18 6
19 7
21 7
ans =
'end'
ans =
'beg'
ans =
1 1
2 1
3 1
4 1
5 1
6 1
7 2
8 2
9 2
10 2
11 2
12 2
13 5
14 3
15 3
16 3
17 3
18 3
19 4
20 4
21 4
22 5
23 5
24 5
25 6
26 6
27 6
28 7
29 7
30 7
ans =
1 1
6 1
7 2
12 2
13 5
14 3
18 3
19 4
21 4
22 5
24 5
25 6
27 6
28 7
30 7
ans =
'end'

```

```

function [ux, uy] = lin2xy(x,y,tol)
% lin2xy(x,y,tol) - keep end points of linear (tolerance=tol) segment (x and y are column
    vectors, the same length, and length > 2)

% parameters
%tol = 1e-8;
n = length(y); % vector length

valx = []; % initial value
valy = [];
lastx = y(2);
lasty = x(2);
kx = x(2) - x(1);
ky = y(2) - y(1);
flg = 0;

ux=[x(1)]; % output vector
uy=[y(1)];

for i = 3:n
    kx1 = x(i) - ux(end);
    ky1 = y(i) - uy(end);
    %fprintf('%d %f %f %f %f\n',i,kx,ky,kx1,ky1);

    if (kx == 0 & kx1 == 0)
        %fprintf('linear1\n');
        flg = 1;
    elseif (ky == 0 & ky1 == 0)
        %fprintf('linear2\n');
        flg = 1;
    elseif abs(kx1/kx - ky1/ky) < tol % is linear
        %fprintf('linear3\n');
        flg = 1;
    else
        %fprintf('not linear\n');
        ux=[ux;lastx];
        uy=[uy;lasty];
        % [ux,uy]
        kx = x(i) - lastx;
        ky = y(i) - lasty;
        flg = 0;
    end

    lastx = x(i);
    lasty = y(i);
end

if flg
    ux=[ux;lastx];
    uy=[uy;lasty];
else
    ux=[ux;x(end)];
    uy=[uy;y(end)];
end
end

```

APPENDIX A.2 – PSD SWEEP DATA GENERATION

```
% MATLAB '9.3.0.713579 (R2017b)'

% create_ds.m

clear all
%===DATA ACETP-400 Annex B B-2, 45 min/axis = 1600 km (994 mi) from 4 hr/axis test w/equiv factor
% ===WB
% Frequency (Hz), Amplitude (G^2/Hz)
% vertical
WB_V = [5, 0.001;
20, 0.015;
510, 0.015;
2000, 0.001]; % 3.56 GRMS

% transverse
WB_T = [5, 0.001;
20, 0.010;
510, 0.010;
2000, 0.001]; % 3.03 GRMS

% longitudinal
WB_L = [5, 0.001;
20, 0.010;
510, 0.010;
2000, 0.001]; % 3.03 GRMS

%===NB
% Bandwidth (Hz), Sweep Range (Low, High), Amplitude (G^2/Hz)(vert, transverse, longitudinal)
NB1 = [5, 20, 170, 0.3, 0.15, 0.15]; % WB+NB1 = 4.20 GRMS
NB2 = [10, 40, 340, 0.2, 0.10, 0.10]; % WB+NB2 = 3.42 GRMS
NB3 = [15, 60, 510, 0.1, 0.05, 0.05]; % WB+NB3 = 3.42 GRMS

%%===%%
save ds_b2.mat

%%=====

=

clear all
%===DATA ACETP-400 Annex B B-4, 45 min/axis = 1600 km (994 mi) from 4 hr/axis test w/equiv factor
%***CHANGE***
% ===WB
% Frequency (Hz), Amplitude (G^2/Hz)
% vertical
WB_V = [5, 0.001;
20, 0.015;
510, 0.015;
2000, 0.001]; % 3.56 GRMS

% transverse
WB_T = [5, 0.001;
20, 0.010;
510, 0.010;
2000, 0.001]; % 3.03 GRMS

% longitudinal
WB_L = [5, 0.001;
20, 0.010;
510, 0.010;
2000, 0.001]; % 3.03 GRMS

%===NB
% Bandwidth (Hz), Sweep Range (Low, High), Amplitude (G^2/Hz)(vert, transverse, longitudinal)
NB1 = [5, 20, 170, 1.0, 1.0, 1.0]; % WB+NB1 = 5.93 GRMS
NB2 = [10, 40, 340, 1.0, 0.50, 0.50]; % WB+NB2 = 4.79 GRMS
NB3 = [15, 60, 510, 0.5, 0.25, 0.25]; % WB+NB3 = 4.79 GRMS

%%===%%
```

```

save ds_b4.mat

%%=====

=

clear all
%===PARAMETERS

%=set individually
%prefix
%sweep_type
%frqchk
%suffix

% sweep.m
total_time = 45*60; % seconds
sweep_rate = 0.75/60; % octave/sec, sweep up/dn
increments = 20; % one-way, sweep up/dn
f_step = 0.1;
freq = [5:f_step:2000]'; % Hz
n_channels = 3; % 1=vertical, 2=transverse, 3=longitudinal
bad_val = 99; % replace for log interp
ddir = './dat\';

exposure_time = 11.58; % secs (see psd1ch.m or psd3ch.m)
olchdir = './dat_psd1\'; % psd1ch.m - 1 channel per file
o3chdir = './dat_psd3\'; % psd3ch.m - 3 channel per file

% show.m - increments, n_channels, ddir
% frqchk.m - increments, n_channels, ddir
% psd1ch.m - increments, n_channels, ddir, exposure_time, olchdir
% psd3ch.m - increments, n_channels, ddir, exposure_time, o3chdir
% ldf1ch.m - (suffix), total_time, exposure_time, olchdir
% ldf3ch.m - total_time, exposure_time, o3chdir
% eg_frq_sweep.m - sweep_rate, increments
%%===%%
save ds_param.mat

```

```

% MATLAB '9.3.0.713579 (R2017b)'

% sweep.m

% check if parameters exist
if exist('prefix') & exist('sweep_type') & exist('frqchk')
    fprintf('PARAMETERS:\n');
    fprintf('    prefix = %s (also dataset name ''ds_x.mat'')\n',prefix);
    fprintf('    sweep_type = %s\n',sweep_type);
    fprintf('    frq_chk = %d (see frqchk.m)\n',frq_chk);
    fprintf('\nPress any key to continue.\n');
    pause
else
    fprintf('Define parameters: prefix=''b2/b4'', sweep_type=''lin/log'', frq_chk=0/1 before running.\n');
    stop
end

flg_lin_out = 1; % output linear in log, do not use for fe-safe as requires all same freq
if flg_lin_out
    fprintf('    flg_lin_out = %d\n',flg_lin_out);
    pause;
end

load( ['ds_' prefix '.mat'] ); % load dataset (see create_ds.m, WB_L, WB_T, WB_V, NB1, NB2, NB3)

%===
% octaves: fu = fc * ((2^(1/2))^(1/n))^m, n=1, octave, n=3, 1/3 rd, m = #1,2,3,4....
%           fl = fc * ((2^(-1/2))^(1/n))^-m, n=1, octave, n=3, 1/3 rd, m = #1,2,3,4....
%           m = log(fu/fl) = ln(fu/fl)/ln(2)

load('ds_param.mat');

%%CHANGE
*** see tmp, outf ***

if frq_chk
    if strcmp(sweep_type,'lin')
        freq = dlmread([ddir prefix '_freq_lin.txt']);
    else
        freq = dlmread([ddir prefix '_freq_log.txt']);
    end
end

%===
diary [prefix '_out']
for i = 1:n_channels, % loop 3 for V, T, L
    fprintf('i = %d\n',i);
    if (i == 1),
        tmp = WB_V;
        outf = [prefix '_psd_v'];
    elseif (i == 2),
        tmp = WB_T;
        outf = [prefix '_psd_t'];
    else
        tmp = WB_L;
        outf = [prefix '_psd_l'];
    end

    % WB only
    % interp using log then go back so linear on PSD plot as from spec
    %WB
    yi = 10.^interp1(log10(tmp(:,1)),log10(tmp(:,2)),log10(freq),'linear',0);
    if flg_lin_out
        [freq1,psd1] = lin2xy(log10(freq),log10(yi),1e-8);
        freq1 = 10.^freq1;
        psd1 = 10.^psd1;
    end

    % output WB data
    nm = [ddir outf 'wb_' sweep_type '.txt'];

```

```

fprintf('Writing %s\n',nm);
fid = fopen(nm,'wt');
if flg_lin_out
    fprintf(fid,'%30.20E %30.20E\n',[freq1'; psd1']);
else
    fprintf(fid,'%30.20E %30.20E\n',[freq'; yi']);
end
fclose(fid);
%loglog(freq,yi);
%loglog(freq1,psd1);
%pause;
%

% calc starting/ending center frequencies considering BW and NB levels
fc_lh = [NB1(2) + NB1(1)/2, NB1(3) - NB1(1)/2, NB1(3+i);
         NB2(2) + NB2(1)/2, NB2(3) - NB2(1)/2, NB2(3+i);
         NB3(2) + NB3(1)/2, NB3(3) - NB3(1)/2, NB3(3+i)];

% calc octave/increment, sweep time total (sec) up or dn, sweep time/increment (sec)
fc_oct = [log2(fc_lh(1,2)/fc_lh(1,1))/increments, log2(fc_lh(1,2)/fc_lh(1,1))/sweep_rate,
...
          log2(fc_lh(1,2)/fc_lh(1,1))/sweep_rate/increments;
          log2(fc_lh(2,2)/fc_lh(2,1))/increments, log2(fc_lh(2,2)/fc_lh(2,1))/sweep_rate,
...
          log2(fc_lh(2,2)/fc_lh(2,1))/sweep_rate/increments;
          log2(fc_lh(3,2)/fc_lh(3,1))/increments, log2(fc_lh(3,2)/fc_lh(3,1))/sweep_rate,
...
          log2(fc_lh(3,2)/fc_lh(3,1))/sweep_rate/increments];

%Rlog = [log2(fc_lh(1,2)/fc_lh(1,1))/fc_oct(1,2), log2(fc_lh(2,2)/fc_lh(2,1))/fc_oct(2,2),
log2(fc_lh(3,2)/fc_lh(3,1))/fc_oct(3,2)];
Rlog = [sweep_rate, sweep_rate, sweep_rate]; % set as specified speed rate
%t_inc = [log2(fc_lh(1,2)/fc_lh(1,1))/sweep_rate/(increments),
log2(fc_lh(2,2)/fc_lh(2,1))/sweep_rate/(increments),
log2(fc_lh(3,2)/fc_lh(3,1))/sweep_rate/(increments)];
t_inc = [fc_oct(1,3), fc_oct(2,3), fc_oct(3,3)];
Rlin = [(fc_lh(1,2)-fc_lh(1,1))/fc_oct(1,2), (fc_lh(2,2)-fc_lh(2,1))/fc_oct(2,2),
(fc_lh(3,2)-fc_lh(3,1))/fc_oct(3,2)];

if strcmp(sweep_type,'log'),
    fprintf('Rlog\n');
    Rlog
else
    fprintf('Rlin\n');
    Rlin
end

fprintf('t_inc\n');
t_inc

fprintf('fc start, fc end, level\n');
fc_lh

fprintf('octave/incr, sweep total time (sec), sweep time/incr (sec)\n');
fc_oct

fprintf('fc\n');
fc0=[fc_lh(1,1), fc_lh(2,1), fc_lh(3,1)]
fc=fc0;

% do increments, recalc fc after each
for j = 0:increments,
    fprintf('increment j=%d\n',j);

    % interp using log then go back so linear on PSD plot as from spec
    %NB1
    yil = 10.^interp1(log10([fc(1) - NB1(1)/2:f_step:fc(1) + NB1(1)/2]), ...
        log10([fc(1) - NB1(1)/2:f_step:fc(1) + NB1(1)/2]*0 + fc_lh(1,3)),
    log10(freq),'linear',bad_val);
    ii=find(yil == 10^bad_val);
    yil(ii) = 0;

```

```

%NB2
yi2 = 10.^interp1(log10([fc(2) - NB2(1)/2:f_step:fc(2) + NB2(1)/2]), ...
    log10([fc(2) - NB2(1)/2:f_step:fc(2) + NB2(1)/2]*0 + fc_lh(2,3)),
log10(freq),'linear',bad_val);
ii=find(yi2 == 10^bad_val);
yi2(ii) = 0;

%NB3
yi3 = 10.^interp1(log10([fc(3) - NB3(1)/2:f_step:fc(3) + NB3(1)/2]), ...
    log10([fc(3) - NB3(1)/2:f_step:fc(3) + NB3(1)/2]*0 + fc_lh(3,3)),
log10(freq),'linear',bad_val);
ii=find(yi3 == 10^bad_val);
yi3(ii) = 0;

% envelop results (WB and NB)
psd = max([yi';yi1';yi2';yi3']');
if flg_lin_out
    [freq1,psd1] = lin2xy(log10(freq),log10(psd),1e-8);
    freq1 = 10.^freq1;
    psd1 = 10.^psd1;
end

% output data
nm = [ddir outf sprintf('%02d_',j) sweep_type '.txt'];
fprintf('Writing %s\n',nm);
fid = fopen(nm,'wt');
if flg_lin_out
    fprintf(fid,'%30.20E %30.20E\n',[freq1'; psd1']);
else
    fprintf(fid,'%30.20E %30.20E\n',[freq'; psd']);
end
fclose(fid);

% plot results
figure(1)
clf
if flg_lin_out
    loglog(freq1,psd1)
else
    loglog(freq,psd)
end

% uncomment for test/compare data written
%hold on
%y = dlmread(nm);
%loglog(y(:,1),y(:,2),'r')

xlabel('Frequency (Hz)');
ylabel('PSD');

%pause

% update fc for next increment
%fc = fc.*2.^[fc_oct(1,1), fc_oct(2,1), fc_oct(3,1)];
%fc = fc0.*2.^[fc_oct(1,2)/(increments)*(j+2)*sweep_rate,
fc_oct(2,2)/(increments)*(j+2)*sweep_rate, fc_oct(3,2)/(increments)*(j+2)*sweep_rate];
if strcmp(sweep_type,'log'),
    fc = fc0.*2.^(Rlog*(j+1).*t_inc)
else
    fc = fc0+(Rlin*(j+1).*t_inc)
end
end
end
diary off

```

```

% MATLAB '9.3.0.713579 (R2017b)'

% show.m

% check if parameters exist
if exist('prefix') & exist('sweep_type')
    fprintf('PARAMETERS:\n');
    fprintf('    prefix = %s\n',prefix);
    fprintf('    sweep_type = %s\n',sweep_type);
    fprintf('\nPress any key to continue.\n');
    pause
else
    fprintf('Define parameters: prefix=\'b2/b4\' , sweep_type=\'lin/log\' before running.\n');
    stop
end

load('ds_param.mat');

%===CHANGE===
%*** see infile ***

%===
figure(1);
for i = 1:n_channels, % loop 3 for V, T, L (1=vertical, 2=transverse, 3=longitudinal)
    if (i == 1),
        infile = [prefix '_psd_v_'];
    elseif (i == 2),
        infile = [prefix '_psd_t_'];
    else
        infile = [prefix '_psd_l_'];
    end

    fprintf('Reading %s\n',[infile 'wb_' sweep_type '.txt']);
    y = dlmread([ddir infile 'wb_' sweep_type '.txt']);
    clf
    loglog(y(:,1),y(:,2),'g');
    hold on
    loglog(y(:,1),y(:,2),'or');
    title(replace([infile 'wb_' sweep_type '.txt'],'_',' '));
    pause
end

%===
for j = 0:increments,
    nj = sprintf('%02d',j);
    fprintf('increment j=%d\n',j);

    for i = 1:n_channels, % loop 3 for V, T, L (1=vertical, 2=transverse, 3=longitudinal)
        fprintf('i = %d\n',i);
        if (i == 1),
            infile = [prefix '_psd_v_'];
        elseif (i == 2),
            infile = [prefix '_psd_t_'];
        else
            infile = [prefix '_psd_l_'];
        end

        fprintf('Reading %s\n',[infile nj '_' sweep_type '.txt']);
        y = dlmread([ddir infile nj '_' sweep_type '.txt']);
        clf
        loglog(y(:,1),y(:,2),'g');
        hold on
        loglog(y(:,1),y(:,2),'or');
        title(replace([infile nj '_' sweep_type '.txt'],'_',' '));
        pause
    end
end
end

```



```

% MATLAB '9.3.0.713579 (R2017b)'

% frqchk.m

% check if parameters exist
if exist('prefix') & exist('sweep_type')
    fprintf('PARAMETERS:\n');
    fprintf('    prefix = %s\n',prefix);
    fprintf('    sweep_type = %s\n',sweep_type);
    fprintf('\nPress any key to continue.\n');
    pause
else
    fprintf('Define parameters: prefix='b2/b4'', sweep_type='lin/log'' before running.\n');
    stop
end

load('ds_param.mat');

%===CHANGE===
%*** see infile ***

tmp = [];

%===
figure(1);
for i = 1:n_channels, % loop 3 for V, T, L (1=vertical, 2=transverse, 3=longitudinal)
    if (i == 1),
        infile = [prefix '_psd_v_'];
    elseif (i == 2),
        infile = [prefix '_psd_t_'];
    else
        infile = [prefix '_psd_l_'];
    end

    fprintf('Reading %s\n',[infile 'wb_' sweep_type '.txt']);
    y = dlmread([ddir infile 'wb_' sweep_type '.txt']);

    clf
    loglog(y(:,1),y(:,2),'g');
    hold on
    loglog(y(:,1),y(:,2),'or');
    title(replace([infile 'wb_' sweep_type '.txt'],'_',' '));

    [freq1, psd1] = lin2xy(log10(y(:,1)), log10(y(:,2)), 1e-6);
    freq1 = 10.^freq1;
    psd1 = 10.^psd1;

    loglog(freq1,psd1,'ok','MarkerSize',10);
    tmp = [tmp,freq1'];
    %pause
end

%===
for j = 0:increments,
    nj = sprintf('%02d',j);
    fprintf('increment j=%d\n',j);

    for i = 1:n_channels, % loop 3 for V, T, L (1=vertical, 2=transverse, 3=longitudinal)
        fprintf('i = %d\n',i);
        if (i == 1),
            infile = [prefix '_psd_v_'];
        elseif (i == 2),
            infile = [prefix '_psd_t_'];
        else
            infile = [prefix '_psd_l_'];
        end

        fprintf('Reading %s\n',[infile nj '_' sweep_type '.txt']);
        y = dlmread([ddir infile nj '_' sweep_type '.txt']);

        clf

```

```

        loglog(y(:,1),y(:,2),'g');
        hold on
        loglog(y(:,1),y(:,2),'or');
        title(replace([infile nj '_' sweep_type '.txt'],'_', ' '));

        [freq1, psd1] = lin2xy(log10(y(:,1)), log10(y(:,2)), 1e-6);
        freq1 = 10.^freq1;
        psd1 = 10.^psd1;

        loglog(freq1,psd1,'ok','MarkerSize',10);
        tmp = [tmp,freq1'];
        %pause
    end
end

tmp = unique(sort(tmp));
fid = fopen([ddir prefix '_freq_' sweep_type '.txt'],'wt');
fprintf(fid,'%30.20E\n',tmp);
fclose(fid);

```

APPENDIX A.3 – PSD AND LDF FILE GENERATION

```

% MATLAB '9.3.0.713579 (R2017b)'

% psd3ch.m

% check if parameters exist
if exist('prefix') & exist('sweep_type')
    fprintf('PARAMETERS:\n');
    fprintf('    prefix = %s\n',prefix);
    fprintf('    sweep_type = %s\n',sweep_type);
    fprintf('\nPress any key to continue.\n');
    pause
else
    fprintf('Define parameters: prefix=''b2/b4'', sweep_type=''lin/log'' before running.\n');
    stop
end

load('ds_param.mat');

%===CHANGE===
%*** see infile ***

%===
% output WB file
nm = [o3chdir prefix '_psd3ch_wb_' sweep_type '.psd'];
fid = fopen(nm,'wt');
fprintf(fid,'Number of channels = 3\n\n');
fprintf(fid,'Exposure time = %f\n\n',exposure_time); % secs, assumes all the same
fprintf(fid,'#s\n\n',nm);

for i = 1:n_channels, % loop 3 for V, T, L (1=vertical, 2=transverse, 3=longitudinal)
    if (i == 1),
        infile = [prefix '_psd_v_'];
    elseif (i == 2),
        infile = [prefix '_psd_t_'];
    else
        infile = [prefix '_psd_l_'];
    end

    fprintf('Reading %s\n',[infile 'wb_' sweep_type '.txt']);
    y = dlmread([ddir infile 'wb_' sweep_type '.txt']);
    nm1 = [infile 'wb_' sweep_type '.txt'];
    fprintf(fid,'#s_frequency(Hz) %s_g^2/Hz\n\n',nm1,nm1);
    fprintf(fid,'Writing %s\n',[prefix '_psd3ch_wb_' sweep_type '.psd']);
    fprintf(fid,'%E\t%E\n',[y(:,1)' y(:,2)']);
    fprintf(fid,'\n\n');
end
fclose(fid);

%===
for j = 0:increments,
    nj = sprintf('%02d',j);
    fprintf('increment j=%d\n',j);
    % output file
    nm = [o3chdir prefix '_psd3ch_' nj '_' sweep_type '.psd'];
    fid = fopen(nm,'wt');
    fprintf(fid,'Number of channels = 3\n\n');
    fprintf(fid,'Exposure time = %f\n\n',exposure_time); % secs, assumes all the same
    fprintf(fid,'#s\n\n',nm);

    for i = 1:n_channels, % loop 3 for V, T, L (1=vertical, 2=transverse, 3=longitudinal)
        fprintf('i = %d\n',i);
        if (i == 1),
            infile = [prefix '_psd_v_'];
        elseif (i == 2),
            infile = [prefix '_psd_t_'];
        else
            infile = [prefix '_psd_l_'];
        end
    end
end

```

```

        fprintf('Reading %s\n',[infile nj '_' sweep_type '.txt']);
        y = dlmread([ddir infile nj '_' sweep_type '.txt']);
        nml = [infile nj '_' sweep_type '.txt'];
        fprintf(fid,'#s_frequency(Hz) %s_g^2/Hz\n\n',nml,nml);
        fprintf('Writing %s\n',[prefix '_psd3ch_' nj '_' sweep_type '.psd']);
        fprintf(fid,'%E\t%E\n',[y(:,1)'; y(:,2)']);
        fprintf(fid,'\n\n');
    end
    fclose(fid);
end

```

```

% MATLAB '9.3.0.713579 (R2017b)'

% psd1ch.m

% check if parameters exist
if exist('prefix') & exist('sweep_type')
    fprintf('PARAMETERS:\n');
    fprintf('    prefix = %s\n',prefix);
    fprintf('    sweep_type = %s\n',sweep_type);
    fprintf('\nPress any key to continue.\n');
    pause
else
    fprintf('Define parameters: prefix='b2/b4'', sweep_type='lin/log'' before running.\n');
    stop
end

load('ds_param.mat');

%===CHANGE===
%*** see infile ***

%===
% output WB file
for i = 1:n_channels, % loop 3 for V, T, L (1=vertical, 2=transverse, 3=longitudinal)
    if (i == 1),
        infile = [prefix '_psd_v_'];
    elseif (i == 2),
        infile = [prefix '_psd_t_'];
    else
        infile = [prefix '_psd_l_'];
    end

    nm = [olchdir infile 'wb_' sweep_type '.psd'];
    fid = fopen(nm,'wt');
    fprintf(fid,'Number of channels = 1\n\n'); % ONE CHANNEL PER FILE
    fprintf(fid,'Exposure time = %f\n\n',exposure_time); % secs, assumes all the same
    fprintf(fid,'#%s\n\n',nm);

    fprintf('Reading %s\n',[infile 'wb_' sweep_type '.txt']);
    y = dlmread([ddir infile 'wb_' sweep_type '.txt']);
    nml = [infile 'wb_' sweep_type '.txt'];
    fprintf(fid,'#%s_frequency(Hz) %s_g^2/Hz\n\n',nml,nml);
    fprintf('Writing %s\n',[infile 'wb_' sweep_type '.psd']);
    fprintf(fid,'%E\t%E\n',[y(:,1)'; y(:,2)']);
    fprintf(fid,'\n\n');
    fclose(fid);
end

%===
for j = 0:increments,
    nj = sprintf('%02d',j);
    fprintf('increment j=%d\n',j);
    % output file

    for i = 1:n_channels, % loop 3 for V, T, L (1=vertical, 2=transverse, 3=longitudinal)
        fprintf('i = %d\n',i);
        if (i == 1),
            infile = [prefix '_psd_v_'];
        elseif (i == 2),
            infile = [prefix '_psd_t_'];
        else
            infile = [prefix '_psd_l_'];
        end

        nm = [olchdir infile nj '_' sweep_type '.psd'];
        fid = fopen(nm,'wt');
        fprintf(fid,'Number of channels = 1\n\n'); % ONE CHANNEL PER FILE
        fprintf(fid,'Exposure time = %f\n\n',exposure_time); % secs, assumes all the same
        fprintf(fid,'#%s\n\n',nm);

        fprintf('Reading %s\n',[infile nj '_' sweep_type '.txt']);

```

```

y = dlmread([ddir infile nj '_' sweep_type '.txt']);
nml = [infile nj '_' sweep_type '.txt'];
fprintf(fid, '#s_frequency(Hz) %s_g^2/Hz\n\n', nml, nml);
fprintf('Writing %s\n', [infile nj '_' sweep_type '.psd']);
fprintf(fid, '%E\t%E\n', [y(:,1)'; y(:,2)']);
fprintf(fid, '\n\n');
fclose(fid);
end
end

```

```

% MATLAB '9.3.0.713579 (R2017b)'

% ldf3ch.m

% check if parameters exist
if exist('prefix') & exist('suffix') & exist('sweep_type')
    fprintf('PARAMETERS:\n');
    fprintf('    prefix = %s\n',prefix);
    fprintf('    sweep_type = %s\n',sweep_type);
    fprintf('\nPress any key to continue.\n');
    pause
else
    fprintf('Define parameters: prefix='b2/b4'', sweep_type='lin/log'' before running.\n');
    stop
end

load('ds_param.mat');

up0 = [0:1:20]; % sweep up numbers, first
up = [1:1:20]; % sweep up numbers
dn = [19:-1:0]; % sweep dn numbers
cycle =[up0,dn,up,dn,up,dn,up,dn,up,dn,-1]; % complete cycle numbers

%===CHANGE===
cycle =[up0,-1]; % complete cycle numbers

%===START LDF (PSD - 3 channels per file, see psd3ch.m)
l = length(cycle);
tt = l*exposure_time;

% output file
nm = [o3chdir prefix sprintf('_psd3ch_%02d_%s.ldf',l,sweep_type)];
fid = fopen(nm,'wt');
fprintf(fid,'\n# .ldf created by user\n');
fprintf(fid,'# Number of PSD files: %d\n',l);
fprintf(fid,'#\n\nINIT\nEND\n\n');

for i = 1:l
    bn = sprintf('%02d',i);
    fn = sprintf('%02d',cycle(i));
    fprintf('Writing Block %d\n',i);
    fprintf(fid,'# Block number %d\n',i);

    if (cycle(i) >= 0),
        fprintf(fid,'BLOCK dt=%f\n',exposure_time);
        psdfile = [pwd o3chdir(2:end) prefix '_psd3ch_' fn '_' sweep_type '.psd'];
        fprintf(fid,'psd=%s\n',psdfile);
        fprintf(fid,'END\n\n');
    else
        fprintf(fid,'#BLOCK dt=%f\n',total_time-tt);
        psdfile = [pwd o3chdir(2:end) prefix '_psd3ch_wb_' sweep_type '.psd'];
        fprintf(fid,'#psd=%s\n',psdfile);
        fprintf(fid,'#END\n\n');
    end
end
fclose(fid);

fprintf('\nTotal Time = %f sec (%f min), remaining %f sec (%f min)\n',tt,tt/60,total_time-
    tt,(total_time-tt)/60);
fprintf('\n\nEDIT Uncomment and edit Exposure Time (to %f sec) for last commented
    Block\n',total_time-tt);

```

```

% MATLAB '9.3.0.713579 (R2017b)'

% ldf1ch.m

% check if parameters exist
if exist('prefix') & exist('suffix') & exist('sweep_type')
    fprintf('PARAMETERS:\n');
    fprintf('    prefix = %s\n',prefix);
    fprintf('    suffix = %s\n',suffix);
    fprintf('    sweep_type = %s\n',sweep_type);
    fprintf('\nPress any key to continue.\n');
    pause
else
    fprintf('Define parameters: prefix=''b2/b4'', suffix=''v/t/l'', sweep_type=''lin/log''
    before running.\n');
    stop
end

load('ds_param.mat');

up0 = [0:1:20]; % sweep up numbers, first
up = [1:1:20]; % sweep up numbers
dn = [19:-1:0]; % sweep dn numbers
cycle = [up0,dn,up,dn,up,dn,up,dn,up,dn,-1]; % complete cycle numbers

====CHANGE====
cycle = [up0,-1]; % complete cycle numbers

====START LDF (PSD - 1 channel per file, see psd1ch.m)
l = length(cycle);
tt = l*exposure_time;

% output file
nm = [olchdir prefix sprintf('_psd_%s_%02d_%s.ldf',suffix,l,sweep_type)];
fid = fopen(nm,'wt');
fprintf(fid,'\n# .ldf created by user\n');
fprintf(fid,'# Number of PSD files: %d\n',l);
fprintf(fid,'#\n\nINIT\nEND\n\n');

for i = 1:l
    bn = sprintf('%02d',i);
    fn = sprintf('%02d',cycle(i));
    fprintf('Writing Block %d\n',i);
    fprintf(fid,'# Block number %d\n',i);

    if (cycle(i) >= 0),
        fprintf(fid,'BLOCK dt=%f\n',exposure_time);
        psdfile = [pwd olchdir(2:end) prefix '_psd_' suffix '_' fn '_' sweep_type '.psd'];
        fprintf(fid,'psd=%s\n',psdfile);
        fprintf(fid,'END\n\n');
    else
        fprintf(fid,'#BLOCK dt=%f\n',total_time-tt);
        psdfile = [pwd olchdir(2:end) prefix '_psd_' suffix '_wb_' sweep_type '.psd'];
        fprintf(fid,'#psd=%s\n',psdfile);
        fprintf(fid,'#END\n\n');
    end
end
fclose(fid);

fprintf('\nTotal Time = %f sec (%f min), remaining %f sec (%f min)\n',tt,tt/60,total_time-
tt,(total_time-tt)/60);
fprintf('\n\nEDIT Uncomment and edit Exposure Time (to %f sec) for last commented
Block\n',total_time-tt);

```


APPENDIX A.4 – EXAMPLE PSD AND LDF FILE FORMATS

```

===BOF (1 channel .psd)
Number of channels = 1

Exposure time = 11.58

#.\dat_psd1\b4_psd_v_17_lin.psd

#b4_psd_v_17_lin.txt_frequency(Hz)    b4_psd_v_17_lin.txt_g^2/Hz

5.000000E+00    1.000000E-03
1.990000E+01    1.485384E-02
2.000000E+01    1.500000E-02
2.500000E+01    1.500000E-02
2.510000E+01    1.500000E-02
2.720000E+01    1.500000E-02
2.730000E+01    1.500000E-02
3.220000E+01    1.500000E-02
3.230000E+01    1.500000E-02
...
===EOF

```

```

===BOF (1 channel .ldf)
# .ldf created by user
# Number of PSD files: 22
#

INIT
END

# Block number 1
BLOCK dt=11.58
psd=F:\fe-safe\work_psd_NB_sweep\dat_psd1\b4_psd_v_00_log.psd
END

# Block number 2
BLOCK dt=11.58
psd=F:\fe-safe\work_psd_NB_sweep\dat_psd1\b4_psd_v_01_log.psd
END
.
.
.
(blocks 3 to 20 here)
.
.
.
# Block number 21
BLOCK dt=11.58
psd=F:\fe-safe\work_psd_NB_sweep\dat_psd1\b4_psd_v_20_log.psd
END

# Block number 22
#BLOCK dt=2457.274000
#psd=F:\fe-safe\work_psd_NB_sweep\dat_psd1\b4_psd_v_wb_log.psd
#END
===EOF

```

```

===BOF (3 channel .psd)
Number of channels = 3

Exposure time = 11.58

```

```

#.\dat_psd3\b4_psd3ch_14_log.psd

#b4_psd_v_14_log.txt_frequency(Hz)    b4_psd_v_14_log.txt_g^2/Hz

5.000000E+00    1.000000E-03
1.990000E+01    1.485384E-02
2.000000E+01    1.500000E-02
2.230000E+01    1.500000E-02
2.240000E+01    1.500000E-02
2.500000E+01    1.500000E-02
...
5.101000E+02    1.499417E-02
2.000000E+03    1.000000E-03

#b4_psd_t_14_log.txt_frequency(Hz)    b4_psd_t_14_log.txt_g^2/Hz

5.000000E+00    1.000000E-03
1.990000E+01    9.917089E-03
2.000000E+01    1.000000E-02
2.230000E+01    1.000000E-02
...
2.000000E+03    1.000000E-03

#b4_psd_l_14_log.txt_frequency(Hz)    b4_psd_l_14_log.txt_g^2/Hz

5.000000E+00    1.000000E-03
1.990000E+01    9.917089E-03
...
2.000000E+03    1.000000E-03
===EOF

```

```

===BOF (3 channel .ldf)
# .ldf created by user
# Number of PSD files: 22
#

INIT
END

# Block number 1
BLOCK dt=11.58
psd=F:\fe-safe\work_psd_NB_sweep\dat_psd3\b2_psd3ch_00_log.psd
END

# Block number 2
BLOCK dt=11.58
psd=F:\fe-safe\work_psd_NB_sweep\dat_psd3\b2_psd3ch_01_log.psd
END
.
.
.
(blocks 3 to 20 here)
.
.
.
# Block number 21
BLOCK dt=11.58
psd=F:\fe-safe\work_psd_NB_sweep\dat_psd3\b2_psd3ch_20_log.psd
END

# Block number 22
#BLOCK dt=2457.274000
#psd=F:\fe-safe\work_psd_NB_sweep\dat_psd3\b2_psd3ch_wb_log.psd
#END
===EOF

```

APPENDIX A.5 – TEST SCRIPT

```

% MATLAB '9.3.0.713579 (R2017b)'

% testall.m - exercise all options

clear all

%=== STEP 1 - SWEEP
frq_chk=0
prefix='b2'
sweep_type='lin'
sweep
sweep_type='log'
sweep

prefix='b4'
sweep_type='lin'
sweep
sweep_type='log'
sweep

%stop

%=== STEP 2 - FRQCHK
prefix='b2'
sweep_type='lin'
frqchk
sweep_type='log'
frqchk

prefix='b4'
sweep_type='lin'
frqchk
sweep_type='log'
frqchk

%stop

%=== STEP 3 - SWEEP AFTER FRQCHK
frq_chk=1
prefix='b2'
sweep_type='lin'
sweep
sweep_type='log'
sweep

prefix='b4'
sweep_type='lin'
sweep
sweep_type='log'
sweep

%stop

%=== STEP 3a - SHOW
prefix='b2'
sweep_type='lin'
show
sweep_type='log'
show

prefix='b4'
sweep_type='lin'
show
sweep_type='log'
show

%stop

%=== STEP 4 - PSD3CH

```

```

prefix='b2'
sweep_type='lin'
psd3ch
sweep_type='log'
psd3ch

prefix='b4'
sweep_type='lin'
psd3ch
sweep_type='log'
psd3ch

%stop

%=== STEP 5 - PSD1CH
prefix='b2'
sweep_type='lin'
psd1ch
sweep_type='log'
psd1ch

prefix='b4'
sweep_type='lin'
psd1ch
sweep_type='log'
psd1ch

%stop

%=== STEP 6 - LDF3CH
prefix='b2'
sweep_type='lin'
ldf3ch
sweep_type='log'
ldf3ch

prefix='b4'
sweep_type='lin'
ldf3ch
sweep_type='log'
ldf3ch

%stop

%=== STEP 7 - LDF1CH

prefix='b2'
sweep_type='lin'
suffix='v'
ldf1ch
suffix='t'
ldf1ch
suffix='l'
ldf1ch
sweep_type='log'
suffix='v'
ldf1ch
suffix='t'
ldf1ch
suffix='l'
ldf1ch

prefix='b4'
sweep_type='lin'
suffix='v'
ldf1ch
suffix='t'
ldf1ch
suffix='l'
ldf1ch
sweep_type='log'

```

```
suffix='v'  
ldflch  
suffix='t'  
ldflch  
suffix='l'  
ldflch
```

APPENDIX A.6 – MATLAB INTERP1 HELP [5a]

interp1

1-D data interpolation (table lookup)

R2024a

[collapse all in page](#)**Syntax**

```
vq = interp1(x,v,xq)
vq = interp1(x,v,xq,method)
vq = interp1(x,v,xq,method,extrapolation)

vq = interp1(v,xq)
vq = interp1(v,xq,method)
vq = interp1(v,xq,method,extrapolation)

pp = interp1(x,v,method,'pp')
```

Description

`vq = interp1(x,v,xq)` returns interpolated values of a 1-D function at specific query points using linear interpolation. Vector `x` contains the sample points, and `v` contains the corresponding values, $v(x)$. Vector `xq` contains the coordinates of the query points. [example](#)

If you have multiple sets of data that are sampled at the same point coordinates, then you can pass `v` as an array. Each column of array `v` contains a different set of 1-D sample values.

`vq = interp1(x,v,xq,method)` specifies an alternative interpolation method: 'linear', 'nearest', 'next', 'previous', 'pchip', 'cubic', 'v5cubic', 'makima', or 'spline'. The default method is 'linear'. [example](#)

`vq = interp1(x,v,xq,method,extrapolation)` specifies a strategy for evaluating points that lie outside the domain of `x`. Set extrapolation to 'extrap' when you want to use the method algorithm for extrapolation. Alternatively, you can specify a scalar value, in which case, `interp1` returns that value for all points outside the domain of `x`. [example](#)

`vq = interp1(v,xq)` returns interpolated values and assumes a default set of sample point coordinates. The default points are the sequence of numbers from 1 to `n`, where `n` depends on the shape of `v`. [example](#)

- When `v` is a vector, the default points are `1:length(v)`.
- When `v` is an array, the default points are `1:size(v,1)`.

Use this syntax when you are not concerned about the absolute distances between points.

`vq = interp1(v,xq,method)` specifies any of the alternative interpolation methods and uses the default sample points.

`vq = interp1(v,xq,method,extrapolation)` specifies an extrapolation strategy and uses the default sample points.

`pp = interp1(x,v,method,'pp')` returns the piecewise polynomial form of $v(x)$ using the method algorithm.

TN 19-0807-01

ESTIMATING RAYLEIGH DAMPING COEFFICIENTS FOR MODAL DAMPING

August 7, 2019

W. Bylsma

U.S. Army Combat Capabilities Development Command (DEVCOM)
Ground Vehicle Systems Center (GVSC)
Detroit Arsenal
Systems Integration & Engineering
ATTN: FCDD-GVS-IES/MS 157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

How does one determine Rayleigh damping coefficients for modal damping?

PROBLEM SOLUTION:

From references [1 – 4] we gather information regarding Rayleigh damping. From [1], Rayleigh Damping is defined as

$$[C] = \alpha[M] + \beta[K] \quad (1)$$

where $[C]$ is the damping matrix, $[M]$ is the mass matrix, and $[K]$ is the stiffness matrix---in effect proportional damping. After transformation [1] obtains

$$2\zeta_i\omega_i = \alpha + \beta\omega_i^2 \quad (2)$$

or

$$\zeta_i = \frac{\alpha}{2\omega_i} + \frac{\beta\omega_i}{2}. \quad (3)$$

Since there are two unknown coefficients, α and β , if the damping ratio, ζ , is known at two different frequencies, ω , we can solve for the coefficients. Solving equation (3) at two different values of i results in

$$\beta = \frac{2\zeta_2\omega_2 - 2\zeta_1\omega_1}{\omega_2^2 - \omega_1^2} \quad (4)$$

and

$$\alpha = 2\zeta_i\omega_i - \beta\omega_i^2 \quad (5)$$

from back substitution of beta into equation (2).

Using [5] for computation, Figure 1 demonstrates the inverse proportional part (alpha) and the proportional part (beta) to frequency of the damping ratio in equation (3) (see eg_p1.m in Appendix A.1 for details).

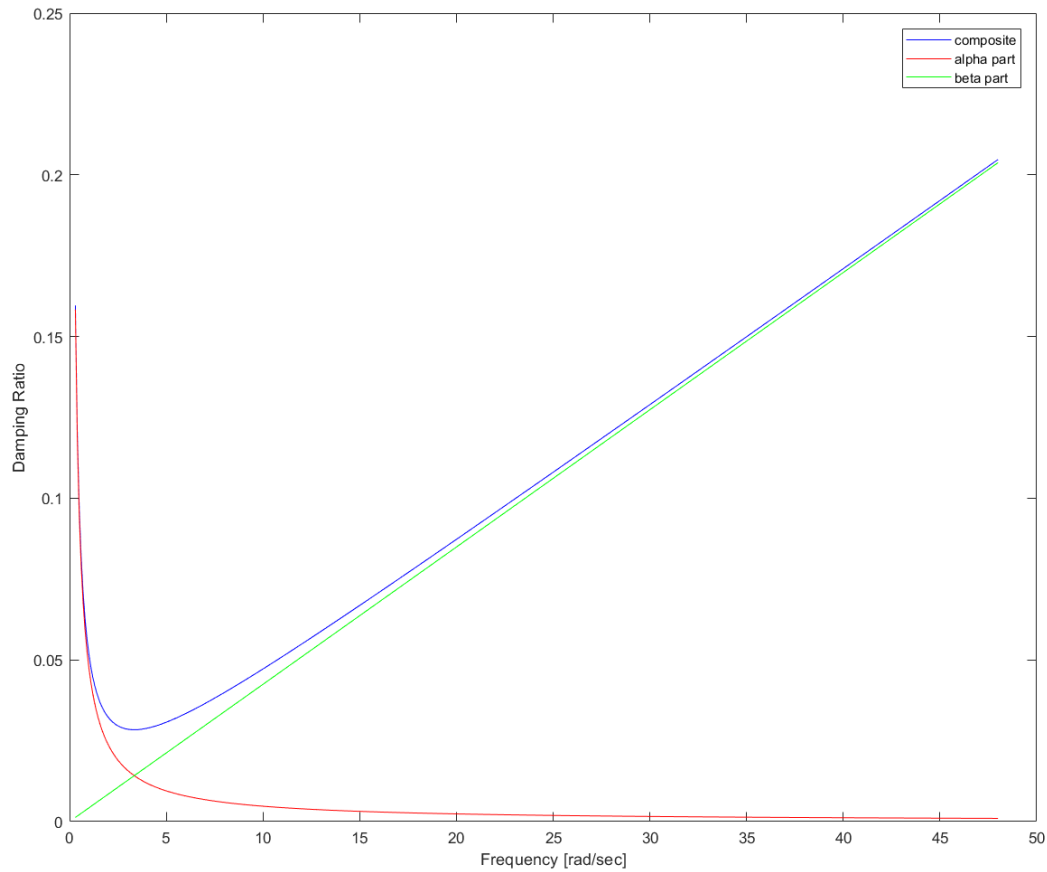


Figure 1 – Rayleigh Damping Parts

The technique to estimate alpha and beta is outlined below and is based on [1]:

1. Define a frequency range that contains the significant modes m (up to 95% of the effective modal mass).
2. Select damping ratios for the first frequency (w_1) and m th mode at frequency (w_2). (e.g. 0.025 at w_1 and 0.1 at w_2).
3. Extrapolate the damping ratio from w_1 and w_2 to the $2.5m$ mode to get a damping ratio at that frequency (w_e).
4. Calculate alpha and beta using the damping ratios chosen in step 2 for w_1 and w_2 . Calculate eta from the alpha and beta just found. This is called the “mid” range eta.
5. Calculate alpha and beta using the damping ratio chosen in step 2 for w_1 and the linearly extrapolated damping ratio in step 3 for w_e . Calculate eta from the alpha and beta just found. This is called the “full” range eta.

6. Calculate alpha and beta as the average of the alpha and beta values found in steps 4 and 5. Calculate eta from the alpha and beta just found. This is called the “avg” range eta.
7. Use the estimates of alpha and beta from step 6 as estimates for the Rayleigh Damping coefficients.

Calculations for alpha and beta can be performed using the eg_rd.m script in Appendix A.1. Example output is shown below and a plot of the damping ratio versus frequency in Figure 2.

eg_rd.m

```

Frequency
Index  [rad/sec]      [Hz]
1      3.0          0.5
2      4.0          0.6
3      7.0          1.1
4      8.0          1.3
5     12.0          1.9
6     20.0          3.2
7     25.0          4.0
8     32.0          5.1
9     38.0          6.0
10    47.0          7.5
11    62.0          9.9
12    75.0         11.9
13   110.0         17.5
14   135.0         21.5
15   140.0         22.3

eta  [rad/sec]      [Hz]
0.0200    3.0        0.5
0.1000   20.0        3.2

alpha      beta
0.030691  0.009923
0.034554  0.009494
0.032622  0.009709

eta
Index  [rad/sec]      linear      mid      full      avg
1      3.0        0.020000    0.020000    0.020000    0.020000
2      4.0        0.024706    0.023683    0.023307    0.023495
3      7.0        0.038824    0.036924    0.035697    0.036310
4      8.0        0.043529    0.041611    0.040136    0.040873
5     12.0        0.062353    0.060818    0.058404    0.059611
6     20.0        0.100000    0.100000    0.095804    0.097902
7     25.0        0.123529    0.124655    0.119367    0.122011
8     32.0        0.156471    0.159252    0.152444    0.155848
9     38.0        0.184706    0.188946    0.180841    0.184894
10    47.0        0.227059    0.233523    0.223477    0.228500
11    62.0        0.297647    0.307869    0.294594    0.301231
12    75.0        0.358824    0.372327    0.356257    0.364292
13   110.0        0.523529    0.545920    0.522329    0.534124
14   135.0        0.641176    0.669935    0.640975    0.655455
15   140.0        0.664706    0.694739    0.664706    0.679722

```

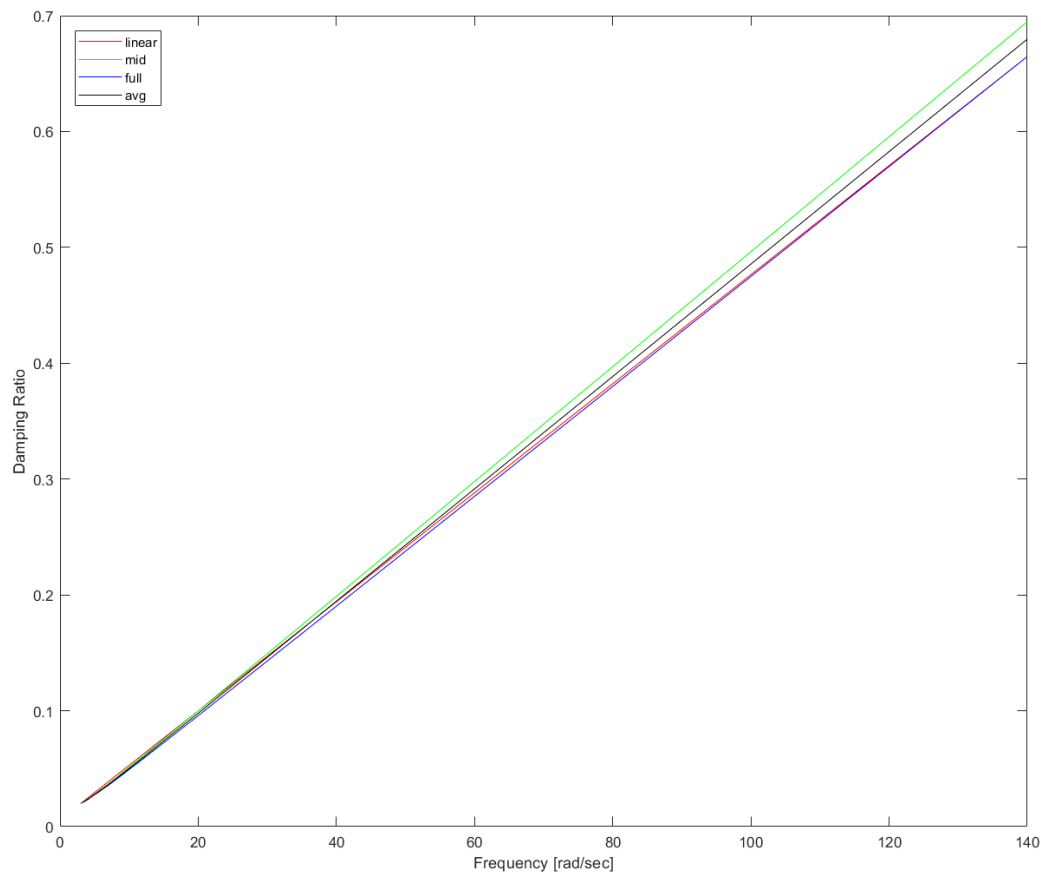


Figure 2 – Example Rayleigh Damping based on Calculated Averaged Alpha and Beta

From [6] it is known that damping has a significant influence on stress which influences fatigue. Caution is in order when approximating damping for the analysis of structures. Best practice is to have some form of reference (e.g. a measurement or test data) on which to base your assumptions.

REFERENCE

- [1] Chowdury, Dasgupta. "Computation of Rayleigh Damping Coefficients for Large Systems". (<http://ejge.com/2003/Ppr0318/Ppr0318.pdf>) [2019-08-07]
- [2] Spears, Jensen. "Approach for Selection of Rayleigh Damping Parameters Used for Time History Analysis". Proceedings of PVP2009, 2009 ASME Pressure Vessels and Piping Division Conference, July 26-30, 2009, Prague, Czech Republic. PVP2009-77257.
- [3] Ramamurti, Mithun, et. al. "Experimental determination of damping ratios at higher modes for use in modal superposition". Journal of Engineering and Technology Research Vol. 4(6), pp. 114-128, November 2012. DOI: 10.5897/JETR12.015.
- [4] Schwarz, Richardson. "Proportional Damping from Experimental Data". Sound and Vibration, June 2013. (<http://www.sandv.com/downloads/1306schw.pdf>). [2019-08-07]
- [5] MATLAB (<https://www.mathworks.com/>) [2019-08-01]

- [6] Munson, Kurt. HBM Prenscia. "Improving the Accuracy of Dynamic Vibration Fatigue Simulation" (https://www.ncode.com/images/Resources/Proceedings/2016/2016nCodeUGM_ImprovingtheAccuracyofDynamicVibrationFatigueSimulation.pdf) [2019-08-05]

APPENDIX A.1 – EXAMPLE SCRIPTS

Files:

eg_p1.m – rayleigh damping, alpha and beta parts

function beta = beta(e1,w1,e2,w2)

function alpha = alpha(e,w,beta)

function eta = eta(a, b, w)

eg_rd.m - rayleigh damping example

```
% MATLAB '9.3.0.713579 (R2017b)'

% eg_p1.m - rayleigh damping, alpha and beta parts

% PARAMETERS
w = [0.3:1:48]'; % rad/sec
b=0.00849; % beta
a=0.095; % alpha

% BEGIN
figure(1)
clf

plot(w,eta(a,b,w),'b'); % composite curve
hold on
plot(w,a./(2*w),'r'); % alpha (inverse proportional to w)
plot(w,b.*w/2,'g'); % beta (proportional to w)

ylabel('Damping Ratio')
xlabel('Frequency [rad/sec]')
legend('composite','alpha part','beta part')
```

```
% MATLAB '9.3.0.713579 (R2017b)'  
function beta = beta(e1,w1,e2,w2)  
% beta (raleigh damping)  
  
beta = 2*(e1*w1-e2*w2)/(w1^2 - w2^2);
```

```
% MATLAB '9.3.0.713579 (R2017b)'  
function alpha = alpha(e,w,beta)  
% alpha (raleigh damping)  
  
alpha = 2*e*w - beta*w^2;
```

```
% MATLAB '9.3.0.713579 (R2017b) '  
function eta = eta(a, b, w)  
% eta (rayleigh damping, damping ratio)  
  
eta = a./(2*w) + b*w/2;
```

```

% MATLAB '9.3.0.713579 (R2017b)'

% eg_rd.m - rayleigh damping

% PARAMETERS
clear all

%w = dlmread('w.dat');
%wn = [1, 20]'; % index of 1st and mid freq

w = [3,4,7,8,12,20,25,32,38,47,62,75,110,135,140]';
wn = [1, 6]'; % index of 1st and mid freq

e = [0.02,0.1]'; % damping = % of critical (w1=first mode, w2=mth mode (95% effective mass),
we=2.5mth mode)

% BEGIN
cnt = [1:length(w)]';
wne = length(w); % index of end freq

% REVIEW DATA
fprintf('%10s\n','Frequency');
fprintf('%10s %10s %10s\n','Index','[rad/sec]','[Hz]');
fprintf('%10d %10.1f %10.1f\n',[cnt';w';w'/(2*pi)]);
fprintf('\n');

fprintf('%10s %10s %10s\n','eta','[rad/sec]','[Hz]');
fprintf('%10.4f %10.1f %10.1f\n',[e';w(wn)';w(wn)/(2*pi)]);
fprintf('\n');

pause;

% EXTRAPOLATE ETA
%el = (( e(2) - e(1) ) / ( w(wn(2)) - w(wn(1)) ) ) .* ( w - w(wn(1)) ) + e(1); % linear interpolation
el = interp1( [w(wn(1)), w(wn(2))],[e(1), e(2)], w, 'linear','extrap'); % linear interpolation
using MATLAB function interp1

%mid
b(1) = beta(e(1),w(wn(1)),e(2),w(wn(2)));
a(1) = alpha(e(2),w(wn(2)),b(1));
et(:,1) = eta(a(1),b(1),w);

%full
b(2) = beta(el(1),w(wn(1)),el(wne),w(wne));
a(2) = alpha(el(wne),w(wne),b(2));
et(:,2) = eta(a(2),b(2),w);

%avg
b(3) = ( b(1) + b(2) ) / 2;
a(3) = ( a(1) + a(2) ) / 2;
et(:,3) = eta(a(3),b(3),w);

% RESULTS
diary out_rd
fprintf('%10s %10s\n','alpha','beta');
fprintf('%10.6f %10.6f\n',[a;b]);
fprintf('\n');

fprintf('%10s\n','eta');
fprintf('%10s %10s %10s %10s %10s\n','Index','[rad/sec]','linear','mid','full','avg');
fprintf('%10d %10.1f %10.6f %10.6f %10.6f %10.6f\n',[cnt';w';el';et]);
fprintf('\n');
diary OFF

% PLOT
figure(1)
clf

plot(w,el,'r'); % linear
hold on
plot(w,eta(a(1),b(1),w),'g'); % mid

```



```
plot(w,eta(a(2),b(2),w),'b'); % full
plot(w,eta(a(3),b(3),w),'k'); % avg
ylabel('Damping Ratio')
xlabel('Frequency [rad/sec]')
legend('linear','mid','full','avg','location','northwest');
```

TN 21-0308-01

MODIFIED GOODMAN DIAGRAM FROM ULTIMATE, YIELD, AND ENDURANCE STRENGTH (SU, SY, AND SE)

March 8, 2021

W. Bylsma

U.S. Army Combat Capabilities Development Command (DEVCOM)
Ground Vehicle Systems Center (GVSC)
Detroit Arsenal
Modeling, Simulation & Software
ATTN: FCDD-GVR-MSS/MS 157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

How can you plot a modified Goodman diagram for an infinite life fatigue check?

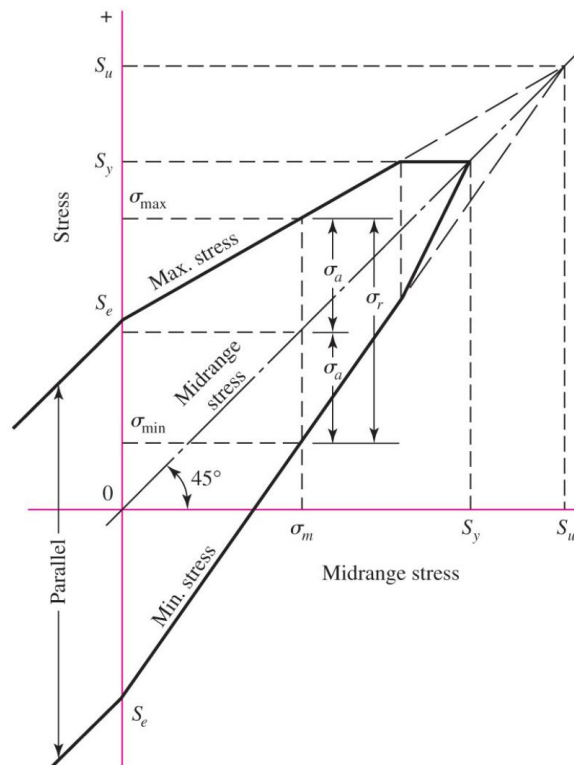


Figure 1 – Modified Goodman Diagram [1]

PROBLEM SOLUTION:

The modified Goodman diagram is defined in [1] and [2]. Powershell [3] is used to create a script that outputs the points to plot the diagram based on inputs for the designated material's ultimate tensile strength

(Stu), tensile yield strength (Sty), and endurance limit (Se). If the ultimate compressive strength (Scu) and compressive yield strength (Scy) is not specified, it is assumed that the compression slope from the endurance limit is one. Output is defined for CSV (point coordinates, .csv) and MATLAB (plot statements, .m) formats. Consistent units are assumed.

From Figure 1, functions for the diagram outline are developed for tension (positive mean stress). They are:

$$\text{upper curve, } yU(x) = \left(\frac{Stu - Se}{Stu} \right) x + Se, \quad x: [0 \text{ to } Stu], y: [Se \text{ to } Stu] \quad (1)$$

$$\text{intersection } (x1, Sty), \quad x1 = \left(\frac{Sty - Se}{Stu - Se} \right) Stu, \quad yU(x) = Sty \quad (2)$$

$$\text{lower curve, } yL(x) = \left(\frac{Stu - (-Se)}{Stu} \right) x - Se, \quad x: [0 \text{ to } Stu], y: [Se \text{ to } Stu] \quad (3)$$

$$\text{intersection } (xp, bL1), \quad bL1 = Sty \left(1 - \frac{(Sty - yL(xp))}{Sty - xp} \right), \quad yL(x): [yU(x) = Sty] \quad (4)$$

$$\begin{aligned} \text{2nd lower curve, } yL1(x) &= \left(\frac{Sty - yL(xp)}{Sty - xp} \right) x + bp, \\ x: [yU(x) = Sty \text{ to } Sty], y: [yL(x: [yU(x) = Sty]) \text{ to } Sty] \end{aligned} \quad (5)$$

For compression (negative mean stress). They are:

$$\text{upper curve (compression), } cU(x) = x + Se, \quad \text{constant slope at } Se \quad (6)$$

$$\text{lower curve (compression), } cL(x) = x - Se, \quad \text{constant slope at } -Se \quad (7)$$

For specified compression values (Scu and Scy), a reflection of the tension curves are used with positive compression values and then negated. Differences between assumed constant or specified compression are

```

Constant
"csu_x = [{0} {1} {2} {3}];" -f 0, -$t_su, -$t_su, 0
"csu_y = [{0} {1} {2} {3}];" -f -$t_su, -$t_su, $t_su, $t_su

Non constant
"csu_x = [{0} {1} {2} {3}];" -f 0, -$c_su, -$c_su, 0
"csu_y = [{0} {1} {2} {3}];" -f -$c_su, -$c_su, $c_su, $c_su

Constant
"csy_x = [{0} {1} {2} {3}];" -f 0, -$t_sy, -$t_sy, 0
"csy_y = [{0} {1} {2} {3}];" -f -$t_sy, -$t_sy, $t_sy, $t_sy

Non constant
"csy_x = [{0} {1} {2} {3}];" -f 0, -$c_sy, -$c_sy, 0
"csy_y = [{0} {1} {2} {3}];" -f -$c_sy, -$c_sy, $c_sy, $c_sy

Constant
"cse_x = [{0} {1} {2} {3} {4}];" -f 0, $cx1_val, -$t_sy, -$t_sy, 0
"cse_y = [{0} {1} {2} {3} {4}];" -f -$se, -$t_sy, -$t_sy, $cy1_val, $se

```

```

Non constant
"cSe_x = [{0} {1} {2} {3} {4}];" -f 0, -$x2_val, -$c_sy, -$x2_val, 0
"cSe_y = [{0} {1} {2} {3} {4}];" -f -$se, -$c_sy, -$c_sy, -(yL1 $x2_val $x2_val $b2_val $c_su $c_sy $se), $se

Constant
"cSm_x = [{0} {1}];" -f 0, -$t_su
"cSm_y = [{0} {1}];" -f 0, -$t_su

Non constant
"cSm_x = [{0} {1}];" -f 0, -$c_su
"cSm_y = [{0} {1}];" -f 0, -$c_su

```

where the points to be plotted are:

```

"plot(tSu_x,tSu_y) "
"hold on"
"plot(cSu_x,cSu_y) "
"plot(tSy_x,tSy_y) "
"plot(cSy_x,cSy_y) "
"plot(tSe_x,tSe_y) "
"plot(cSe_x,cSe_y) "
"plot(tSm_x,tSm_y) "
"plot(cSm_x,cSm_y) "

```

A brief outline of the process follows.

- set tensile parameters
 - \$x1_val = x1 \$t_su \$t_sy \$se
 - \$b1_val = bL1 \$x1_val \$t_su \$t_sy \$se
- if parameter count < \$pc (7 for all), then
 - set constant compression @ Se
 - \$cx1_val = \$se - \$t_sy # x - Se = -Sy
 - \$cy1_val = cU (-\$t_sy) \$se
 - if \$fmt -eq 0, then outcoord_c_const, else outcoord_c_const_MATLAB
 - if \$tbl -eq \$true, then alt_rng_c_const
- else
 - set compression Su, Sy as specified
 - \$x2_val = x1 \$c_su \$c_sy \$se
 - \$b2_val = bL1 \$x2_val \$c_su \$c_sy \$se
 - if (\$fmt -eq 0), then outcoord, else outcoord_MATLAB
 - if \$tbl -eq \$true, then alt_rng

Example output is shown for both the .csv and .m formats.

Example mg.ps1 output (with stress table output, \$tbl = \$true)

Modified Goodman Diagram			
Parameters:			
Material	Steel		
Su_t	400		
Sy_t	250		
Se	100		
fmt	0		
min	mean	max	rng
-250.00	-250.00	-150.00	1
-250.00	-245.00	-145.00	1
-250.00	-240.00	-140.00	1
-250.00	-235.00	-135.00	1
...			
56.25	125.00	193.75	3
62.50	130.00	197.50	3
68.75	135.00	201.25	3
75.00	140.00	205.00	3
...			
230.00	240.00	250.00	4
240.00	245.00	250.00	4
250.00	250.00	250.00	4

Example .csv, CSV output

tSu,0,400
tSu,400,400
tSu,400,-400
tSu,0,-400
cSu,0,-400
cSu,-400,-400
cSu,-400,400
cSu,0,400
tSy,0,250
tSy,250,250
tSy,250,-250
tSy,0,-250
cSy,0,-250
cSy,-250,-250
cSy,-250,250
cSy,0,250
tSe,0,100
tSe,200,250
tSe,250,250
tSe,200,150
tSe,0,-100
cSe,0,-100
cSe,-150,-250
cSe,-250,-250
cSe,-250,-150
cSe,0,100
tSm,0,0
tSm,400,400
cSm,0,0
cSm,-400,-400

Example .m, MATLAB output

```

clf
tSu_x = [0 400 400 0];
tSu_y = [400 400 -400 -400];
cSu_x = [0 -400 -400 0];
cSu_y = [-400 -400 400 400];
tSy_x = [0 250 250 0];
tSy_y = [250 250 -250 -250];
cSy_x = [0 -250 -250 0];
cSy_y = [-250 -250 250 250];
tSe_x = [0 200 250 200 0];
tSe_y = [100 250 250 150 -100];
cSe_x = [0 -150 -250 -250 0];
cSe_y = [-100 -250 -250 -150 100];
tSm_x = [0 400];
tSm_y = [0 400];
cSm_x = [0 -400];
cSm_y = [0 -400];
plot(tSu_x,tSu_y)
hold on
plot(cSu_x,cSu_y)
plot(tSy_x,tSy_y)
plot(cSy_x,cSy_y)
plot(tSe_x,tSe_y)
plot(cSe_x,cSe_y)
plot(tSm_x,tSm_y)
plot(cSm_x,cSm_y)

```

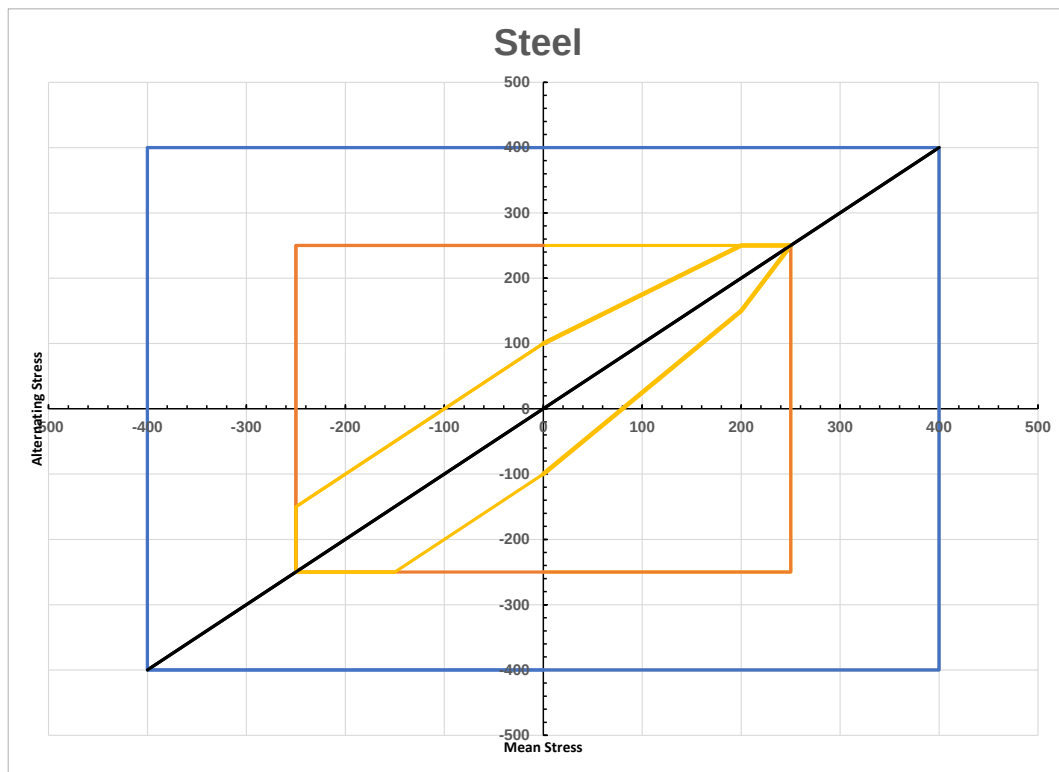


Figure 1 – Constant Compression, CSV format Microsoft Excel Plot

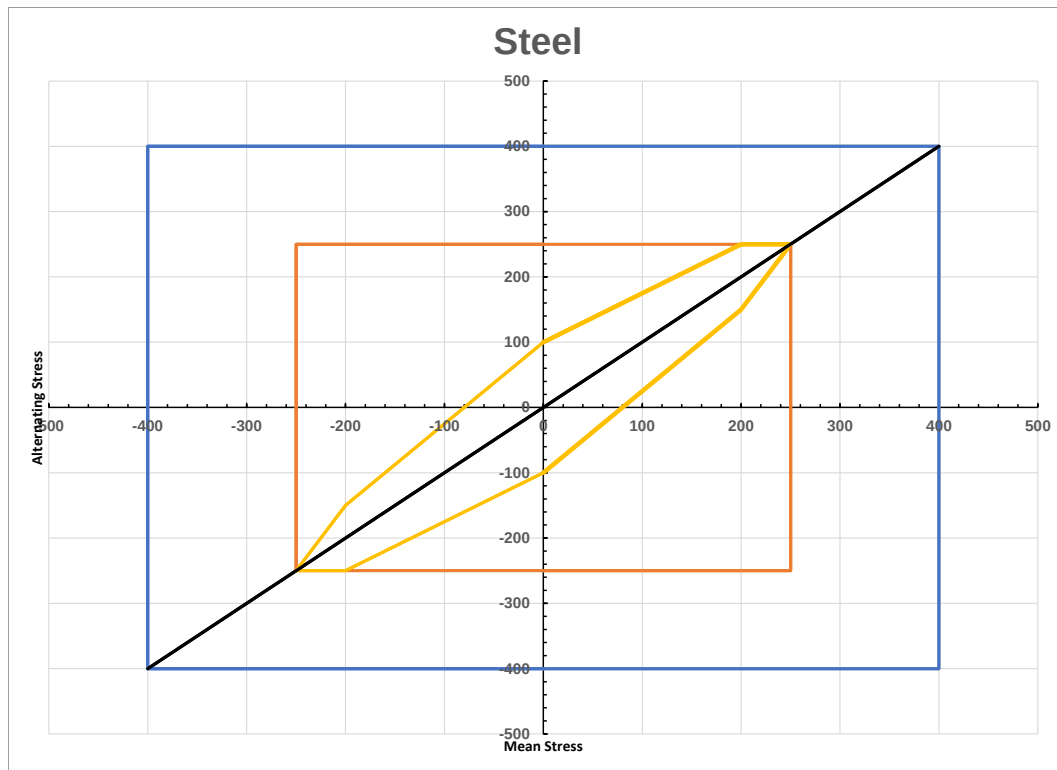


Figure 2 – Specified Compression=Tension, CSV format Microsoft Excel Plot

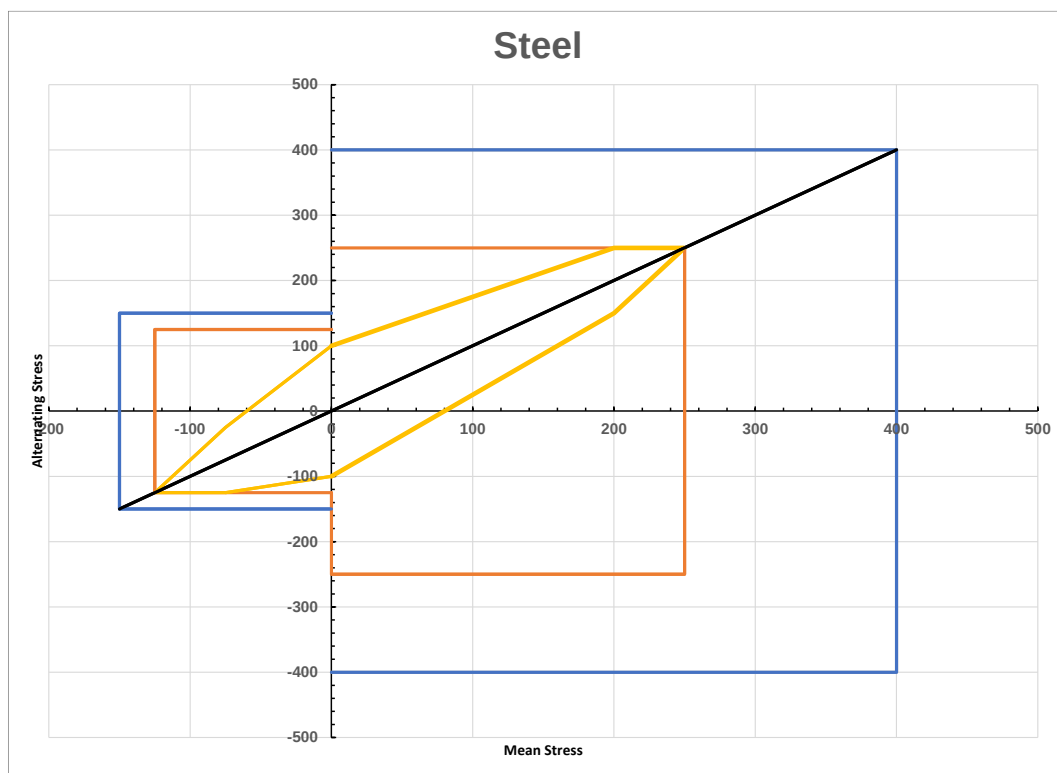


Figure 3 – Specified Compression<Tension, CSV format Microsoft Excel Plot

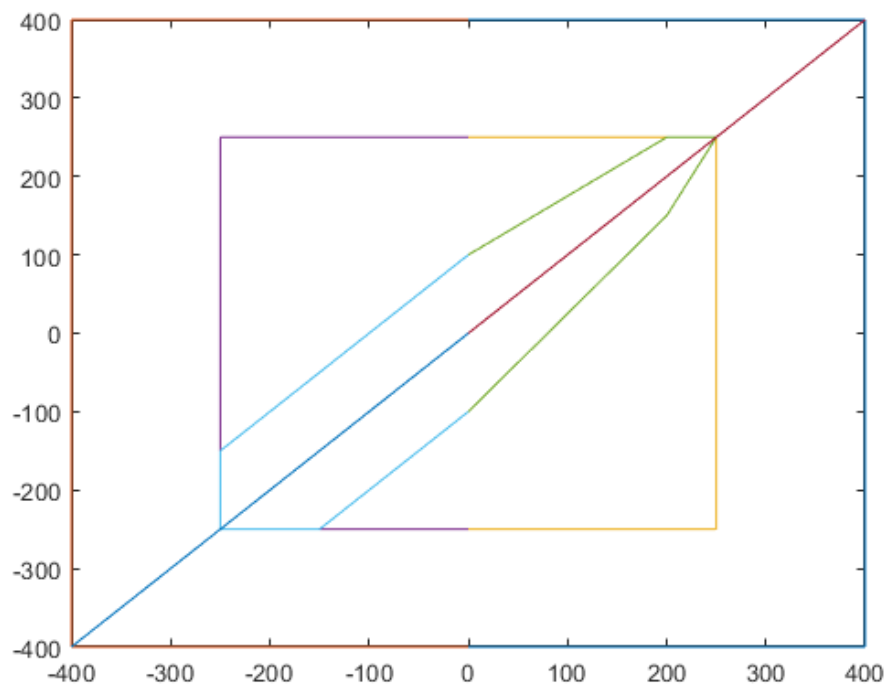


Figure 4 – Constant Compression, MATLAB format MATLAB Plot

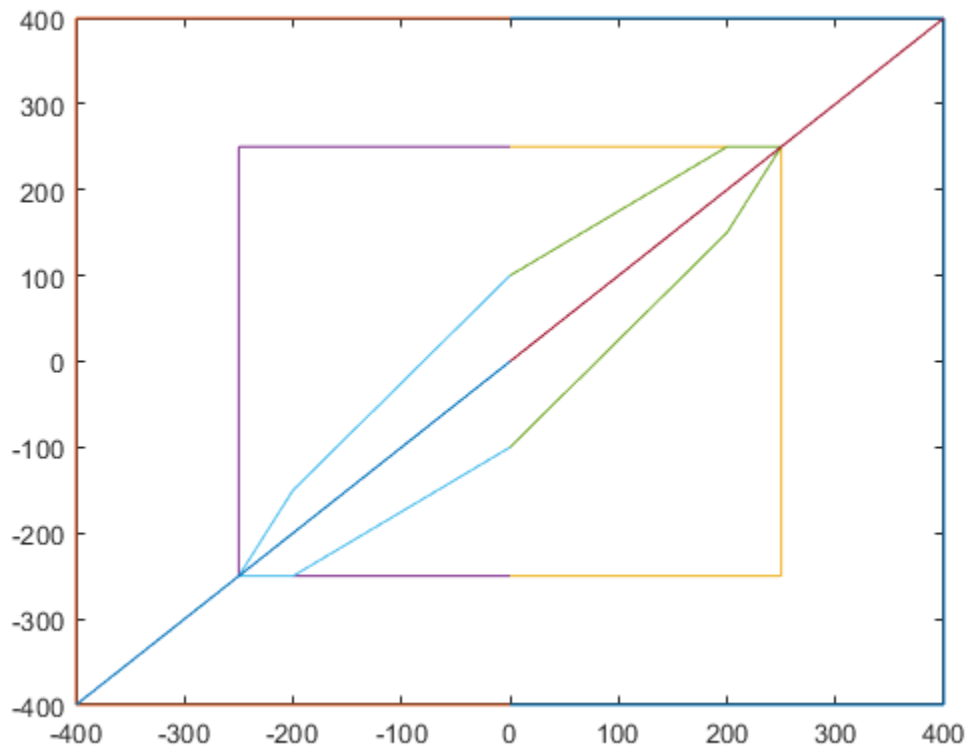


Figure 5 – Specified Compression=Tension, MATLAB format MATLAB Plot

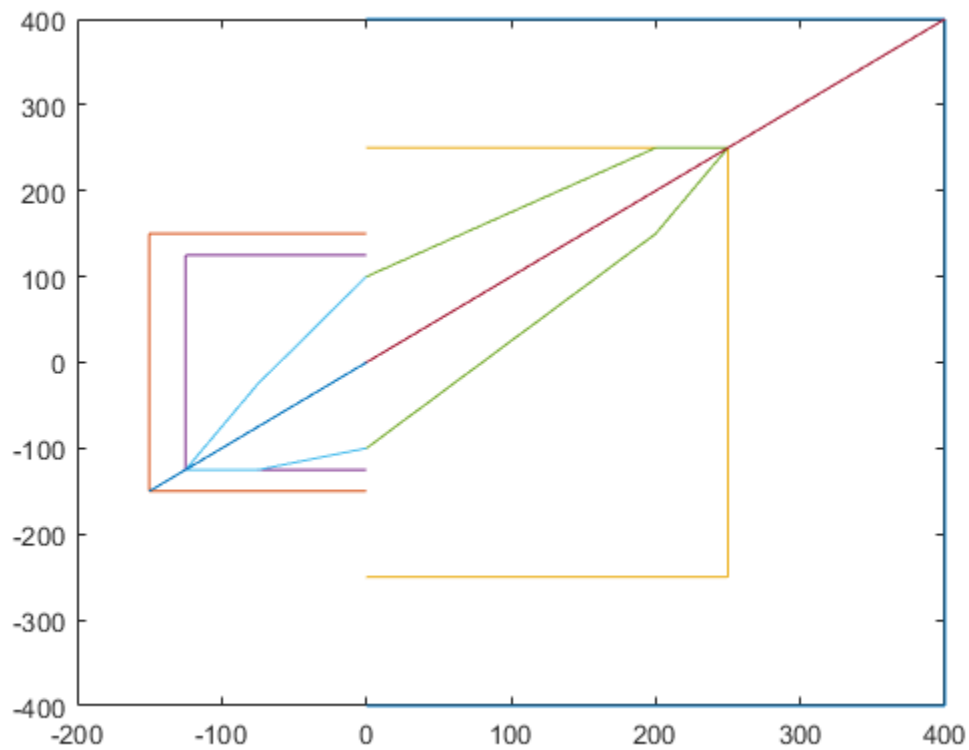


Figure 6 – Specified Compression<Tension, MATLAB format MATLAB Plot

Generally it is assumed that compression is constant from the endurance limit (S_e) to yield (S_{cy}). The option to specify non constant compression gives more flexibility to represent material properties as appropriate.

REFERENCE

- [1] Shigley, et. al. Mechanical Engineering Design, 7th Ed. McGraw-Hill (2004). ISBN 0-07-252036-1. (7-12 Fatigue Failure Criteria for Fluctuating Stress, Figure 7-24, pg. 346)
- [2] Fatigue limit diagram according to Haigh and Smith (<https://www.tec-science.com/material-science/material-testing/fatigue-limit-diagram-according-to-haigh-and-smith-creation/>), see also Fatigue Limit Diagram Haigh & Smith at <https://www.youtube.com/watch?v=pMUbk2DTJR8>)
- [3] Powershell (<https://docs.microsoft.com/en-us/powershell/>)
- [4] MATLAB (<https://www.mathworks.com/products/matlab.html>)
- [5] Blodgett, Design of Welded Structures, The James F. Lincoln Arc Welding Foundation (1966). Section 2.9 Designing for Fatigue Loads, pg. 2.9-2 Fig. 3

APPENDIX

A.1 MODIFIED GOODMAN DIAGRAM POWERSHELL SCRIPT

APPENDIX A.1– MODIFIED GOODMAN DIAGRAM POWERSHELL SCRIPT

```

===BOF (mg.ps1)
param( [string]$m, [double]$t_su, [double]$t_sy, [double]$se, [int]$fmt, [double]$c_su, [double]$c_sy)

<#

mg.ps1 - Modified Goodman Diagram

.\mg [$m] [$t_su] [$t_sy] [$se] [$fmt] [$c_su] [$c_sy]
.\mg "Steel" 400 250 100 0 | out-file -encoding ASCII eg1.csv
.\mg "Steel" 400 250 100 0 400 250 | out-file -encoding ASCII eg2.csv
.\mg "Steel" 400 250 100 0 150 125 | out-file -encoding ASCII eg3.csv
.\mg "Steel" 400 250 100 1 | out-file -encoding ASCII eg4.m
.\mg "Steel" 400 250 100 1 400 250 | out-file -encoding ASCII eg5.m
.\mg "Steel" 400 250 100 1 150 125 | out-file -encoding ASCII eg6.m

INPUTS:
$m                material (e.g. "Steel")
$t_su             ultimate tensile strength
$t_sy             tensile yield strength
$se               endurance limit
$fmt              format (0=CSV, 1=MATLAB)

[optional]
$c_su             ultimate compressive strength (positive value)
$c_sy             compressive yield strength (positive value)

NOTE:  assumes consistent units

RETURNS:
===BOF (.csv, CSV)
tSu,0,400
tSu,400,400
tSu,400,-400
tSu,0,-400
cSu,0,-400
cSu,-400,-400
cSu,-400,400
cSu,0,400

tSy,0,250
tSy,250,250
tSy,250,-250
tSy,0,-250
cSy,0,-250
cSy,-250,-250
cSy,-250,250
cSy,0,250

tSe,0,100
tSe,200,250
tSe,250,250
tSe,200,150
tSe,0,-100
cSe,0,-100
cSe,-150,-250
cSe,-250,-250
cSe,-250,-150
cSe,0,100

tSm,0,0
tSm,400,400
cSm,0,0
cSm,-400,-400
===EOF

===BOF (.m, MATLAB)
clf
tSu_x = [0 400 400 0];
tSu_y = [400 400 -400 -400];
cSu_x = [0 -400 -400 0];
cSu_y = [-400 -400 400 400];
tSy_x = [0 250 250 0];
tSy_y = [250 250 -250 -250];
cSy_x = [0 -250 -250 0];
cSy_y = [-250 -250 250 250];
tSe_x = [0 200 250 200 0];
tSe_y = [100 250 250 150 -100];
cSe_x = [0 -150 -250 -250 0];
cSe_y = [-100 -250 -250 -150 100];
tSm_x = [0 400];
tSm_y = [0 400];
cSm_x = [0 -400];
cSm_y = [0 -400];
plot(tSu_x,tSu_y)
hold on
plot(cSu_x,cSu_y)
plot(tSy_x,tSy_y)
plot(cSy_x,cSy_y)
plot(tSe_x,tSe_y)
plot(cSe_x,cSe_y)
plot(tSm_x,tSm_y)
plot(cSm_x,cSm_y)
===EOF

VARIABLES:
$pc                parameter count = 7 (with compression specified)
$step              stress range table increment units
$tbl               stress range table output flag
$xl_val            x of tension break point where upper curve = Sy (abscissa)
$yl_val            y of tension break point where lower curve = Sy (ordinae)
$cxl_val           x of compression break point where lower curve = Sy (tension, constant)
$cy1_val           y of compression break point where upper curve = Sy (tension, constant)
$x2_val            x of compression break point where lower curve = Sy (tension, specified)
$b2_val            y of compression break point where upper curve = Sy (tension, Specified)

FUNCTIONS:
# tension

```

```

yU          yU(x) upper curve, (x|0 to Su, y|Se to Su)
x1          x where yU(x) = Sy
yL          yL(x) lower curve, (x|0 to Su, y|-Se to Su)
bL1        y of yL(x where yU = Sy)
yL1        yL1(x) 2nd lower curve, (x|x where yU = Sy to Sy, y|yL(x where yU = Sy) to Sy)

# compression
cU          cU(x) compression upper curve, constant slope at Se
cL          cL(x) compression lower curve, constant slope at -Se
NOTE:      uses reflection of tension upper and lower curves for positive compression then negates

# output
outcoord_c_const      diagram coordinates for constant compression
outcoord_c_const_MATLAB      diagram coordinates for constant compression (MATLAB format)
outcoord              diagram coordinates
outcoord_MATLAB        diagram coordinates (MATLAB format)

#>
<#
=====
#>

function yU ( $x, $su, $se ) {
    # yU(x) upper curve, (x|0 to Su, y|Se to Su)
    return ($su - $se)/$su * $x + $se
}

function x1 ($su, $sy, $se) {
    # x1, x where yU(x) = Sy
    return ($sy - $se)/($su - $se) * $su
}

#$x1_val = x1 $t_su $t_sy $se

function yL ( $x, $su, $se ) {
    # yL(x) lower curve, (x|0 to Su, y|-Se to Su)
    return ($su - (-$se))/su * $x - $se
}

function bL1 ($xp, $su, $sy, $se ) {
    # bL1, y of yL(x where yU = Sy)
    return $sy * (1 - ($sy - (yL $xp $su $se))/($sy - $xp))      # solve @(Sy,Sy)
}

#$b1_val = bL1 $x1_val $t_su $t_sy $se

function yL1 ( $x, $xp, $bp, $su, $sy, $se ) {
    # yL1(x) 2nd lower curve, (x|x where yU = Sy to Sy, y|yL(x where yU = Sy) to Sy)
    return ($sy - (yL $xp $su $se))/($sy - $xp) * $x + $bp
}

function cU ( $x, $se ) {
    # cU(x) compression upper curve, constant slope at Se
    return $x + $se
}

function cL ( $x, $se ) {
    # cL(x) compression lower curve, constant slope at -Se
    return $x - $se
}

function outcoord_c_const {
    # diagram coordinates for constant compression

    "{0},{1},{2}" -f "tSu",0,$t_su
    "{0},{1},{2}" -f "tSu",$t_su,$t_su
    "{0},{1},{2}" -f "tSu",$t_su,-$t_su
    "{0},{1},{2}" -f "tSu",0,-$t_su
    "{0},{1},{2}" -f "cSu",0,-$t_su
    "{0},{1},{2}" -f "cSu",- $t_su,-$t_su
    "{0},{1},{2}" -f "cSu",- $t_su,$t_su
    "{0},{1},{2}" -f "cSu",0,$t_su
    " "
    "{0},{1},{2}" -f "tSy",0,$t_sy
    "{0},{1},{2}" -f "tSy",$t_sy,$t_sy
    "{0},{1},{2}" -f "tSy",$t_sy,-$t_sy
    "{0},{1},{2}" -f "tSy",0,-$t_sy
    "{0},{1},{2}" -f "cSy",0,-$t_sy
    "{0},{1},{2}" -f "cSy",- $t_sy,-$t_sy
    "{0},{1},{2}" -f "cSy",- $t_sy,$t_sy
    "{0},{1},{2}" -f "cSy",0,$t_sy
    " "
    "{0},{1},{2}" -f "tSe",0,$se
    "{0},{1},{2}" -f "tSe",$x1_val,$t_sy
    "{0},{1},{2}" -f "tSe",$t_sy,$t_sy
    "{0},{1},{2}" -f "tSe",$x1_val, $x1_val $b1_val $t_su $t_sy $se)
    "{0},{1},{2}" -f "tSe",0,-$se
    "{0},{1},{2}" -f "cSe",0,-$se

```

```

"0),(1),(2)" -f "cSe",$cxl_val,-$t_sy
"0),(1),(2)" -f "cSe",- $t_sy,-$t_sy
"0),(1),(2)" -f "cSe",- $t_sy,$cyl_val
"0),(1),(2)" -f "cSe",0,$sae
"
"0),(1),(2)" -f "tSm",0,0
"0),(1),(2)" -f "tSm",$t_su,$t_su
"0),(1),(2)" -f "cSm",0,0
"0),(1),(2)" -f "cSm",- $t_su,-$t_su
}

function outcoord {

# diagram coordinates

"0),(1),(2)" -f "tSu",0,$t_su
"0),(1),(2)" -f "tSu",$t_su,$t_su
"0),(1),(2)" -f "tSu",$t_su,-$t_su
"0),(1),(2)" -f "tSu",0,-$t_su
"0),(1),(2)" -f "cSu",0,-$c_su
"0),(1),(2)" -f "cSu",- $c_su,-$c_su
"0),(1),(2)" -f "cSu",- $c_su,$c_su
"0),(1),(2)" -f "cSu",0,$c_su
"
"0),(1),(2)" -f "tSy",0,$t_sy
"0),(1),(2)" -f "tSy",$t_sy,$t_sy
"0),(1),(2)" -f "tSy",$t_sy,-$t_sy
"0),(1),(2)" -f "tSy",0,-$t_sy
"0),(1),(2)" -f "cSy",0,-$c_sy
"0),(1),(2)" -f "cSy",- $c_sy,-$c_sy
"0),(1),(2)" -f "cSy",- $c_sy,$c_sy
"0),(1),(2)" -f "cSy",0,$c_sy
"
"0),(1),(2)" -f "tSe",0,$sae
"0),(1),(2)" -f "tSe",$x1_val,$t_sy
"0),(1),(2)" -f "tSe",$t_sy,$t_sy
"0),(1),(2)" -f "tSe",$x1_val,(yL1 $x1_val $x1_val $b1_val $t_su $t_sy $sae)
"0),(1),(2)" -f "tSe",0,-$sae
"0),(1),(2)" -f "cSe",0,-$sae
"0),(1),(2)" -f "cSe",- $x2_val,-$c_sy
"0),(1),(2)" -f "cSe",- $c_sy,-$c_sy
"0),(1),(2)" -f "cSe",- $x2_val,-(yL1 $x2_val $x2_val $b2_val $c_su $c_sy $sae)
"0),(1),(2)" -f "cSe",0,$sae
"
"0),(1),(2)" -f "tSm",0,0
"0),(1),(2)" -f "tSm",$t_su,$t_su
"0),(1),(2)" -f "cSm",0,0
"0),(1),(2)" -f "cSm",- $c_su,-$c_su
}

function outcoord_const_MATLAB {

# diagram coordinates for constant compression (MATLAB)

"clf"

tSu_x = {[0] [1] [2] [3]};" -f 0, $t_su, $t_su, 0
tSu_y = {[0] [1] [2] [3]};" -f $t_su, $t_su, -$t_su, -$t_su
cSu_x = {[0] [1] [2] [3]};" -f 0, -$t_su, -$t_su, 0
cSu_y = {[0] [1] [2] [3]};" -f -$t_su, -$t_su, $t_su, $t_su

tSy_x = {[0] [1] [2] [3]};" -f 0, $t_sy, $t_sy, 0
tSy_y = {[0] [1] [2] [3]};" -f $t_sy, $t_sy, -$t_sy, -$t_sy
cSy_x = {[0] [1] [2] [3]};" -f 0, -$t_sy, -$t_sy, 0
cSy_y = {[0] [1] [2] [3]};" -f -$t_sy, -$t_sy, $t_sy, $t_sy

tSe_x = {[0] [1] [2] [3] [4]};" -f 0, $x1_val, $t_sy, $x1_val, 0
tSe_y = {[0] [1] [2] [3] [4]};" -f $sae, $t_sy, $t_sy, (yL1 $x1_val $x1_val $b1_val $t_su $t_sy $sae), -$sae
cSe_x = {[0] [1] [2] [3] [4]};" -f 0, $cxl_val, -$t_sy, -$t_sy, 0
cSe_y = {[0] [1] [2] [3] [4]};" -f -$sae, -$t_sy, -$t_sy, $cyl_val, $sae

tSm_x = {[0] [1]};" -f 0, $t_su
tSm_y = {[0] [1]};" -f 0, $t_su
cSm_x = {[0] [1]};" -f 0, -$t_su
cSm_y = {[0] [1]};" -f 0, -$t_su

"plot(tSu_x,tSu_y)"
"hold on"
"plot(cSu_x,cSu_y)"
"plot(tSy_x,tSy_y)"
"plot(cSy_x,cSy_y)"
"plot(tSe_x,tSe_y)"
"plot(cSe_x,cSe_y)"
"plot(tSm_x,tSm_y)"
"plot(cSm_x,cSm_y)"
}

function outcoord_MATLAB {

# diagram coordinates (MATLAB)

"clf"

tSu_x = {[0] [1] [2] [3]};" -f 0, $t_su, $t_su, 0
tSu_y = {[0] [1] [2] [3]};" -f $t_su, $t_su, -$t_su, -$t_su
cSu_x = {[0] [1] [2] [3]};" -f 0, -$c_su, -$c_su, 0
cSu_y = {[0] [1] [2] [3]};" -f -$c_su, -$c_su, $c_su, $c_su

tSy_x = {[0] [1] [2] [3]};" -f 0, $t_sy, $t_sy, 0
tSy_y = {[0] [1] [2] [3]};" -f $t_sy, $t_sy, -$t_sy, -$t_sy
cSy_x = {[0] [1] [2] [3]};" -f 0, -$c_sy, -$c_sy, 0
cSy_y = {[0] [1] [2] [3]};" -f -$c_sy, -$c_sy, $c_sy, $c_sy

tSe_x = {[0] [1] [2] [3] [4]};" -f 0, $x1_val, $t_sy, $x1_val, 0
tSe_y = {[0] [1] [2] [3] [4]};" -f $sae, $t_sy, $t_sy, (yL1 $x1_val $x1_val $b1_val $t_su $t_sy $sae), -$sae
cSe_x = {[0] [1] [2] [3] [4]};" -f 0, -$x2_val, -$c_sy, -$x2_val, 0
cSe_y = {[0] [1] [2] [3] [4]};" -f -$sae, -$c_sy, -$c_sy, -(yL1 $x2_val $x2_val $b2_val $c_su $c_sy $sae), $sae

tSm_x = {[0] [1]};" -f 0, $t_su
tSm_y = {[0] [1]};" -f 0, $t_su
cSm_x = {[0] [1]};" -f 0, -$c_su
cSm_y = {[0] [1]};" -f 0, -$c_su

```

```

"plot(tSu_x,tSu_y)"
"hold on"
"plot(cSu_x,cSu_y)"
"plot(tSy_x,tSy_y)"
"plot(cSy_x,cSy_y)"
"plot(tSe_x,tSe_y)"
"plot(cSe_x,cSe_y)"
"plot(tSm_x,tSm_y)"
"plot(cSm_x,cSm_y)"
}

function alt_rng_c_const () {
<#
    # stress range table - constant compression

    VARIABLES:
    $alt          alternating stress
    $lw           lower curve
    $up           upper curve
    $rng          curve range, break points

#>
write-host ("`n`n`n(0,12) {1,12} {2,12} {3,12}" -f "min", "mean", "max", "rng")

# loop through yield strength, compressive to tensile
[double[]]$alt = (-$t_sy..$t_sy)

$alt | where { ($_ - $t_sy) % $step -eq 0 } | foreach {
    if ($_ -le $cx1_val) {
        # range 1, $alt <= $cx1_val
        $lw = -$t_sy
        $up = cU $_ $se
        $rng = 1
    } elseif ($_ -ge $cx1_val -and $_ -le 0) {
        # range 2, $cx1_val <= $alt <= 0
        $lw = CL $_ $se
        $up = CU $_ $se
        $rng = 2
    } elseif ($_ -ge 0 -and $_ -le $x1_val) {
        # range 3, 0 <= $alt <= $x1_val
        $lw = yL $_ $t_su $se
        $up = yU $_ $t_su $se
        $rng = 3
    } elseif ($_ -ge $x1_val) {
        # range 4, $x1_val <= $alt
        $lw = yL1 $_ $x1_val $b1_val $t_su $t_sy $se
        $up = $t_sy
        $rng = 4
    }
    write-host ("{0,12:F2} {1,12:F2} {2,12:F2} {3,12}" -f $lw,$_,$up, $rng)
}
}

function alt_rng () {
<#
    # stress range table - specified compression

    VARIABLES:
    $alt          alternating stress
    $lw           lower curve
    $up           upper curve
    $rng          curve range, break points

#>
write-host ("`n`n`n(0,12) {1,12} {2,12} {3,12}" -f "min", "mean", "max", "rng")

# loop through yield strength, compressive to tensile
[double[]]$alt = (-$c_sy..$t_sy)

$alt | where { ($_ - $c_sy) % $step -eq 0 } | foreach {
    if ($_ -le -$x2_val) {
        # range 1, $alt <= -$x2_val
        $lw = -$c_sy
        $up = -(yL1 (-1*$_) $x2_val $b2_val $c_su $c_sy $se)
        $rng = 1
    } elseif ($_ -ge -$x2_val -and $_ -le 0) {
        # range 2, -$x2_val <= $alt <= 0
        $lw = -(yU (-1*$_) $c_su $se)
        $up = -(yL (-1*$_) $c_su $se)
        $rng = 2
    } elseif ($_ -ge 0 -and $_ -le $x1_val) {
        # range 3, 0 <= $alt <= $x1_val
        $lw = yL $_ $t_su $se
        $up = yU $_ $t_su $se
        $rng = 3
    } elseif ($_ -ge $x1_val) {
        # range 4, $x1_val <= $alt
        $lw = yL1 $_ $x1_val $b1_val $t_su $t_sy $se
        $up = $t_sy
        $rng = 4
    }
}
}

```

321

```

===BOF:eg4.m (constant compression)
clf
tSu_x = [0 400 400 0];
tSu_y = [400 400 -400 -400];
cSu_x = [0 -400 -400 0];
cSu_y = [-400 -400 400 400];
tSy_x = [0 250 250 0];
tSy_y = [250 250 -250 -250];
cSy_x = [0 -250 -250 0];
cSy_y = [-250 -250 250 250];
tSe_x = [0 200 250 200 0];
tSe_y = [100 250 250 150 -100];
cSe_x = [0 -150 -250 -250 0];
cSe_y = [-100 -250 -250 -150 100];
tSm_x = [0 400];
tSm_y = [0 400];
cSm_x = [0 -400];
cSm_y = [0 -400];
plot(tSu_x,tSu_y)
hold on
plot(cSu_x,cSu_y)
plot(tSy_x,tSy_y)
plot(cSy_x,cSy_y)
plot(tSe_x,tSe_y)
plot(cSe_x,cSe_y)
plot(tSm_x,tSm_y)
plot(cSm_x,cSm_y)
===EOF

===BOF:eg5.m (compression reflection of tension)
tSu_x = [0 400 400 0];
tSu_y = [400 400 -400 -400];
cSu_x = [0 -400 -400 0];
cSu_y = [-400 -400 400 400];
tSy_x = [0 250 250 0];
tSy_y = [250 250 -250 -250];
cSy_x = [0 -250 -250 0];
cSy_y = [-250 -250 250 250];
tSe_x = [0 200 250 200 0];
tSe_y = [100 250 250 150 -100];
cSe_x = [0 -200 -250 -200 0];
cSe_y = [-100 -250 -250 -150 100];
tSm_x = [0 400];
tSm_y = [0 400];
cSm_x = [0 -400];
cSm_y = [0 -400];
plot(tSu_x,tSu_y)
hold on
plot(cSu_x,cSu_y)
plot(tSy_x,tSy_y)
plot(cSy_x,cSy_y)
plot(tSe_x,tSe_y)
plot(cSe_x,cSe_y)
plot(tSm_x,tSm_y)
plot(cSm_x,cSm_y)
===EOF

===BOF:eg6.m (specified compression Su, Sy)
clf
tSu_x = [0 400 400 0];
tSu_y = [400 400 -400 -400];
cSu_x = [0 -150 -150 0];
cSu_y = [-150 -150 150 150];
tSy_x = [0 250 250 0];
tSy_y = [250 250 -250 -250];
cSy_x = [0 -125 -125 0];
cSy_y = [-125 -125 125 125];
tSe_x = [0 200 250 200 0];
tSe_y = [100 250 250 150 -100];
cSe_x = [0 -75 -125 -75 0];
cSe_y = [-100 -125 -125 -25 100];
tSm_x = [0 400];
tSm_y = [0 400];
cSm_x = [0 -150];
cSm_y = [0 -150];
plot(tSu_x,tSu_y)
hold on
plot(cSu_x,cSu_y)
plot(tSy_x,tSy_y)
plot(cSy_x,cSy_y)
plot(tSe_x,tSe_y)
plot(cSe_x,cSe_y)
plot(tSm_x,tSm_y)
plot(cSm_x,cSm_y)
===EOF

```


TN 21-0317-01

TRACK PATTERN FREQUENCIES

March 17, 2021

W. Bylsma

U.S. Army Combat Capabilities Development Command (DEVCOM)
Ground Vehicle Systems Center (GVSC)
Detroit Arsenal
Modeling, Simulation & Software
ATTN: FCDD-GVR-MSS/MS 157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

In consideration of a modal analysis on a vehicle system, what are the vibration frequencies generated at specific speeds, known as track pattern? Base excitation should avoid speeds where track pattern frequencies align with modal frequencies to prevent response amplification (resonance).

PROBLEM SOLUTION:

Track pattern is a well-known phenomenon in track vehicles and is documented in MIL-STD-810H [1] and NATO AECTP-240 [2]. Figure 1 provides a succinct description and also summarizes overall grms levels for different terrain severity. Using these references, track pattern excitation frequencies for a range of vehicle speeds were calculated, with references in the Appendix used to determine the track pitch parameter. Figure 2 shows the good correlation between the calculated frequency and the frequency of the maximum PSD value for the turret.

A more graphical view of the test data is quite revealing. Using nCode 2020 [3] and its VibeSys/Frequency Response Analysis (Waterfall Analysis Method, see FrequencyResponse Properties in the Appendix) plots of turret PSD data are shown in Figures 3 to 5 at multiple speeds. Figure 5 indicates that even over cross country terrain, track pattern is a significant contributor to vibration.

In Figure 6, the effects of damping on the natural frequencies is shown. Results from GVSC Structures modal analysis are for undamped frequencies. Consideration should be given to the potential effects damping can have on shifting the natural frequencies.

Based on these findings, the inverse track pattern formula can be used to find an equivalent vehicle speed from a modal frequency to determine what vehicle speeds are less likely to align with modal frequencies and reduce base excitation to prevent response amplification (resonance).

$$velocity[MPH] = \frac{mode\ frequency[Hz] * (track\ pitch\ [inches])}{17.6} \quad (1)$$

In tracked vehicles, the impact of successive track plates on the ground is perceived within the vehicle as narrow band spectral components, which can be severe. These narrow band components are speed dependent and relate to the fundamental track patter frequency and usually several higher harmonics from interaction between the tracks and the road surface. The wide band component is generated by the combined effects of the rolling of the wheels on the tracks, interactions between the track links and the various other sources including engine, gearbox, generators, etc. Peaks in response frequencies corresponding to the vehicle's suspension system can be expected to be low, e.g.: <3 Hz. Relatively broad band peaks in frequencies may also be evident corresponding to structural dynamic modes of the vehicle itself. These modes may lie in the 20 to 100 Hz frequency range. [1][2]

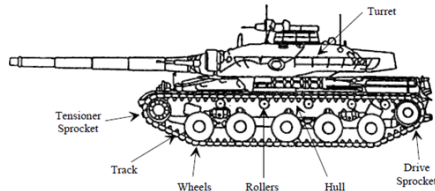


Figure 1 – NATO AECTP-240 and MIL-STD-810H References

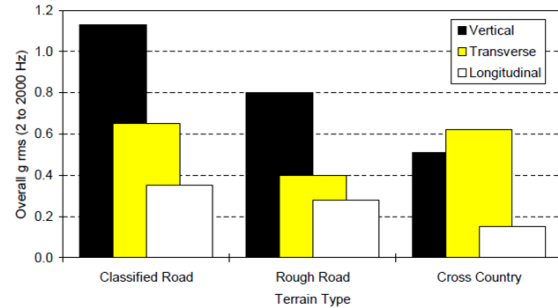
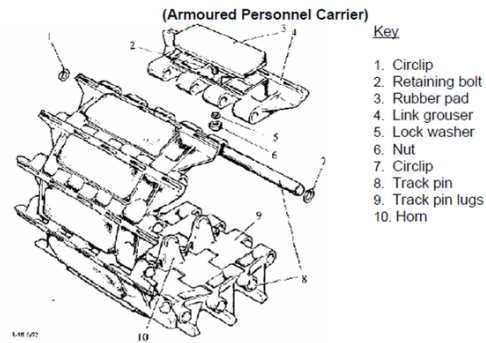


Figure 3: Relative severity of terrain for the hull of a tracked vehicle



NATO AECTP-240
MIL-STD-810H

Calculated track patter frequencies correlate well with the frequency of maximum Power Spectral Density Values from vehicle test data

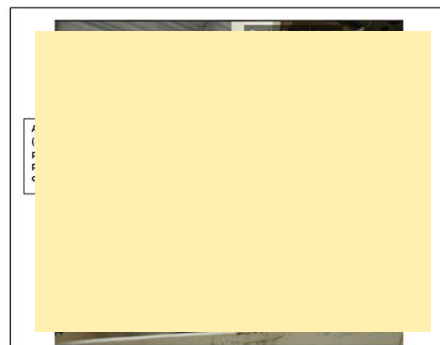


Figure 9. TOW Missile Assembly (TMA) Accelerometer Placements

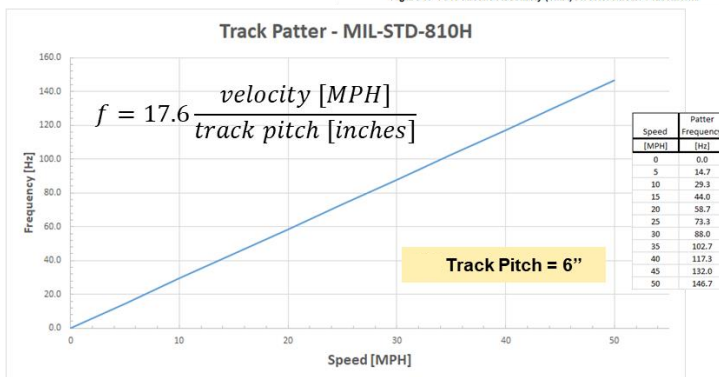


Figure 2 – Calculated Track Patter Frequency vs Test Data for Maximum PSD Frequency

Frequency of Maximum PSD Value
(Channel 1, Turret)

	Speed [MPH]	Patter [Hz]	Channels Direction		
			L	T	V
			2	3	4
Example	15	44.0	44.03	45.01	45.01
	20	58.7	60.67	56.75	57.73
	25	73.3	72.41	70.45	71.43
	30	88.0	87.08	89.04	86.11
	35	102.7	102.74	103.72	100.78
	40	117.3	116.44	115.46	117.42

Track patten and its harmonics can be seen from paved road data where the vehicle accelerated to a top speed of ~43 MPH

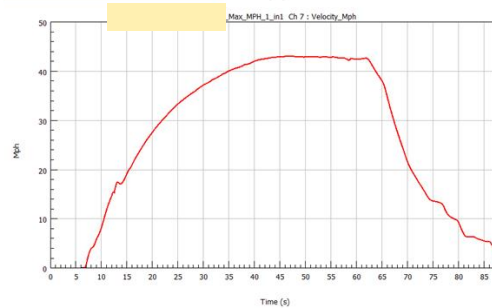
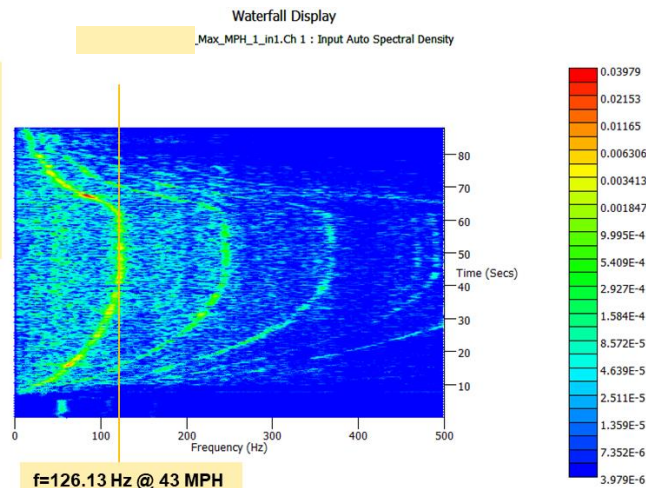


Figure 3 – Track Patter in Paved Test Data (43 MPH)

Track patten can be seen from paved road data at a vehicle speed of ~35 MPH

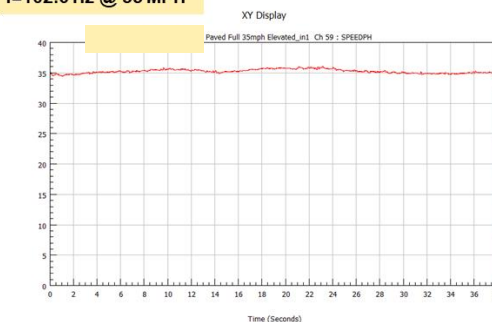
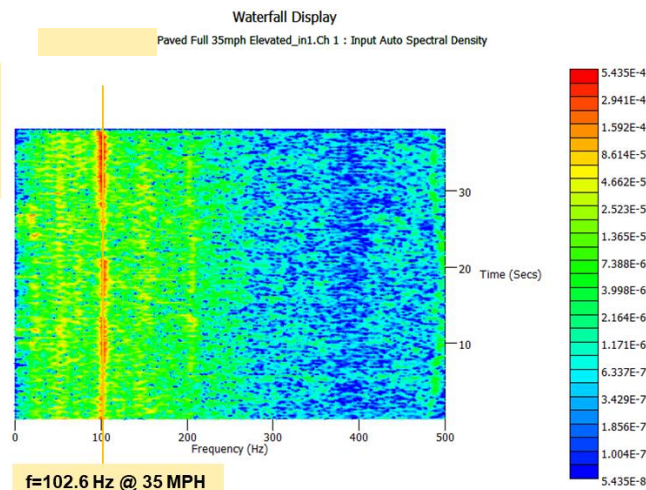


Figure 4 – Track Patter in Paved Test Data (35 MPH)

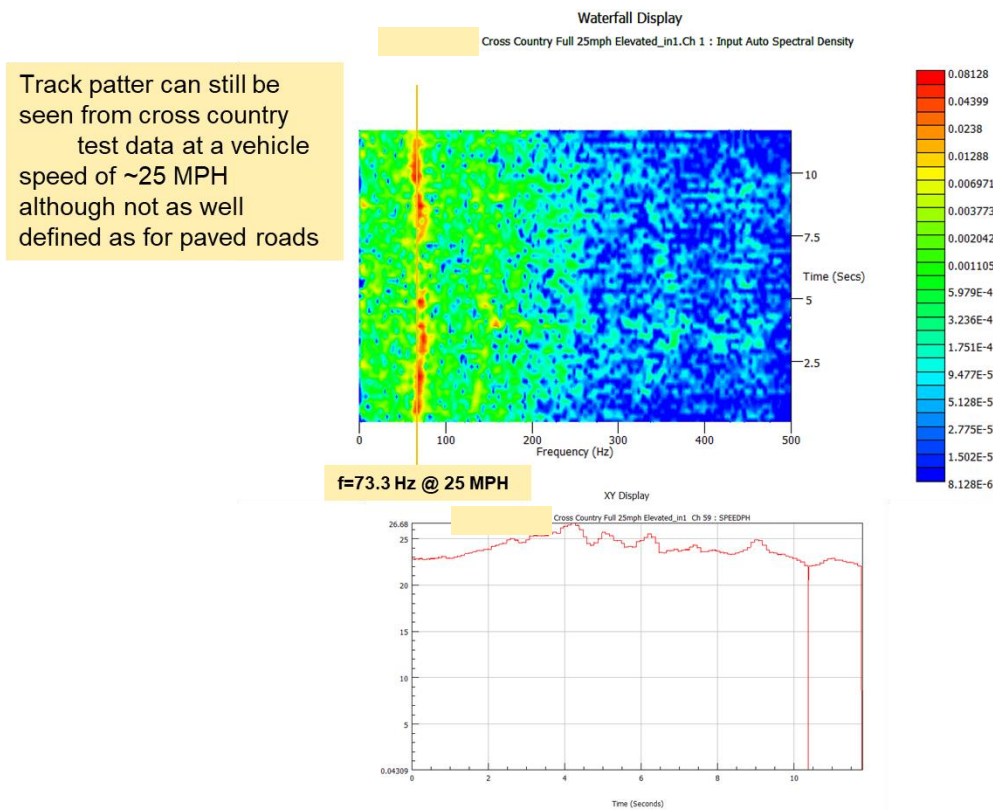


Figure 5 – Track Patter in Cross Country Test Data (25 MPH)

Damped natural frequency depends on the damping ratio (zeta) and is a fraction of the undamped natural frequency. Damping shifts the natural frequencies and can affect the response to track patter.

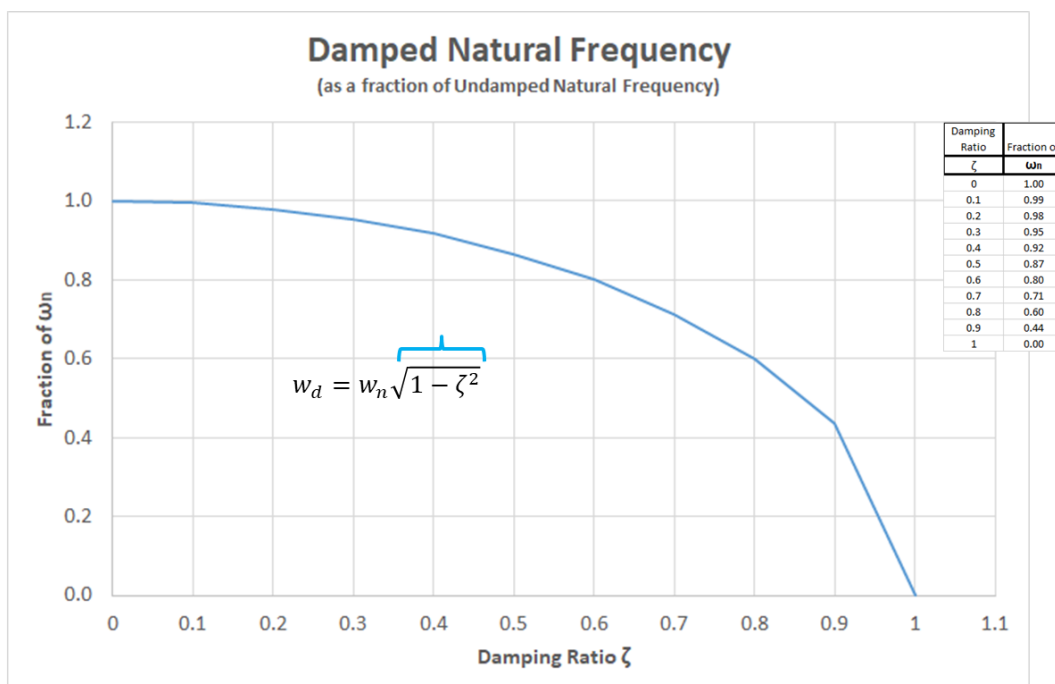


Figure 6 – ω_n vs. Damping Ratio

REFERENCE

- [1] MIL-STD-810H ENVIRONMENTAL ENGINEERING CONSIDERATIONS AND LABORATORY TESTS, Method 514.8 Vibration, Annex F, 5.2 Platform Specific Considerations – Road Transport – Tracked Vehicle. DIST A. Approved for public release: distribution unlimited. (https://quicksearch.dla.mil/qsDocDetails.aspx?ident_number=35978)
- [2] NATO AECTP-240 (Ed 1) MECHANICAL CONDITIONS, Edition 1, July 2009. Per C-M(2002)60 Information releasable to the Public (it shall carry no markings of any sort) see (<https://nso.nato.int/nso/nsdd/main/standards>, search for AECTP).
- [3] nCode 2020 (<https://www.ncode.com/products/ncode-glyphworks-signal-processing-and-durability-analysis>) now (<https://www.hbkworld.com/en/products/software/analysis-simulation/durability/ncode-glyphworks-signal-processing-and-durability-analysis>)

APPENDIX

Track dimension estimates:

Pitch Length = Pin Length (~2.125") + Body Length (~3.875") = ~6".

see DWG 12496800 Track Shoe, Vehicular

see DWG 12496794 Connector, Pin End, Track

see DWG 12496793 Block, Track

 FrequencyResponse Properties ? X

Name	Value	Description
General		
AnalysisMethod	Waterfall	Specifies Analysis Method
OutputType	Amplitude	Specifies the type of spectrum to produce
ZoomTransform	False	Specifies the way to combine the data from multiple FFT buffers
Channels		Specifies the Analysis channel Pairs {Input,Response}
ChannelOffset	1	Specifies the Channel Offset to apply to the groups of results data
AnalysisType	AutoSpectrumInput	Specifies the analysis type to perform
CreateH2Gain	False	Specifies whether H2 Gain results are to be created
ZeroZeroGain	One	Specifies answer for Zero/Zero in the gain file
CreatePhaseData	False	Specifies whether phase results are to be created
HistogramLabels	Minimal	Specifies the type of labelling on the output histograms
AveragingMethod	PeakHold	Specifies the way to combine the data from multiple FFT buffers
Buffer		
WindowType	Hanning	Specifies the window type to apply to each buffer
BufferSize	4096	Specifies the FFT buffer size
BufferOverlap	67	Specifies the overlap on the input data for each FFT analysis
IncrementMethod		
IncrementMethod	Time	Specifies Increment Method
WaterfallTime		
StartTime	#BaseTime#	Specifies the Start Time for the Analysis
EndTime	#BaseTime#+(#NumPoints#-1)/#SampleRate#	Specifies the Start Time for the Analysis
AveragingTime	0.2	Specifies the Averaging Time
SpectrumTimeSpacing	0.2	Specifies the Spectrum Time Spacing
AdvancedSettings		
DataNormalization	Buffer	Specifies the normalization to apply
NoiseFloor	-120	Specifies the value below which a spectral line will be set to zero (
IncludeZeroPaddedBu	True	Specifies whether to include zero padded buffers at the end of the
ENBWCorrection	Auto	Specifies whether to correct for equivalent noise band width

TN 21-0331-01

RANDOM RESPONSE OUTPUT USING TRANSMISSIBILITY

March 31, 2021

W. Bylsma

U.S. Army Combat Capabilities Development Command (DEVCOM)
Ground Vehicle Systems Center (GVSC)
Detroit Arsenal
Modeling, Simulation & Software
ATTN: FCDD-GVR-MSS/MS 157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

Is there a more efficient way to obtain random response results using ABAQUS's [2] random response analysis [2a] method for many inputs (e.g. multiple test data runs at different speeds). An abbreviated description of the random response analysis appears below.

Random response analysis predicts the response of a system that is subjected to a nondeterministic continuous excitation that is expressed in a statistical sense by a cross-spectral density matrix. Since the loading is nondeterministic, it can be characterized only in a statistical sense; Abaqus/Standard assumes that the excitation is stationary and ergodic. These statistical measures are explained in detail in Random response analysis. The random response procedure can, for example, be used to determine the response of an airplane to turbulence, the response of a car to road surface imperfections, the response of a structure to jet noise, or the response of a building to an earthquake.

The random response procedure is based on using a subset of the modes of the system, which must first be extracted by using the eigenfrequency extraction procedure. The modes will include eigenmodes and, if activated in the eigenfrequency extraction step, residual modes. The number of modes extracted must be sufficient to model the dynamic response of the system adequately, which is a matter of judgment on your part. The model can be preloaded prior to the eigenfrequency extraction. Initial stress effects are included in the stiffness used in the eigenfrequency extraction if geometric nonlinearities are included in the general analysis procedure used to apply the preloads (General and perturbation procedures).

The random response of the model is expressed as power spectral density values of nodal and element variables, as well as their root mean square values.---[2a]

PROBLEM SOLUTION:

While completing previous random response analysis [1], it was found to be very beneficial to use white noise [3] (a constant Power Spectral Density (PSD) value of one over all frequencies considered) to determine the random response. This effectively revealed the transmissibility [4] (the ratio of output to input) which provides insight into what frequencies the input is amplified or attenuated at the output---essentially a transfer function revealing the characteristics of the structure.

Instead of running the model for each input to be considered, can the model be ran once using white noise as the input to obtain the transmissibility (for each output node) and then use the transmissibility response to multiply any random input to obtain the random output?. This would reduce the amount of computational time as only simple multiplication of each random input signal would be required and could be accomplished with any calculation platform (e.g. MATLAB, Powershell, Microsoft Excel, etc.)

From [1] we consider the *RANDOM RESPONSE [2b] steps in "t34cpsd.inp".

t34cpsd.inp

```

.
.
.
*BOUNDARY
RB,1,6
*STEP, INC =          10, NAME = static_reg, NLGEOM = YES
static
*STATIC, ALLSDTOL, STABILIZE
0.0          ,1.0          ,1.0000E-05,1.0
*DLOAD, OP=NEW
,GRAV,-9807.0,0,1,0
*OUTPUT, FIELD, NAME = static_out
*NODE OUTPUT
U,RF
*ELEMENT OUTPUT
S
*FILE FORMAT
*END STEP
*STEP, INC =          10, NAME = static_pert, NLGEOM = YES, PERTURBATION
static
*STATIC
*OUTPUT, FIELD, NAME = static_out
*NODE OUTPUT
U,RF
*ELEMENT OUTPUT
S
*END STEP
*STEP, INC =          10, NAME = freq_for_psd_rr-Crew, NLGEOM = YES
freq
*FREQUENCY, EIGENSOLVER = LANCZOS, RESIDUAL MODES
,,500.0
*MODAL FILE
GU, GV, GA
*OUTPUT, FIELD, NAME = freq_out_psd_rr-crew
*NODE OUTPUT, NSET = responses-crew
U
*END STEP
*STEP, INC =          10, NAME = rr_white-Crew, NLGEOM = YES
rr
*RANDOM RESPONSE
0.0          ,250.0          ,500          ,3.0,          0
*CORRELATION, PSD = psd_tbl_white, TYPE = UNCORRELATED, COMPLEX = NO
1
,1.0
*MODAL DAMPING, MODAL = DIRECT, DEFINITION = FREQUENCY RANGE ,FIELD = ALL
125.0,          0.03
*BASE MOTION, DOF=2
*OUTPUT, HISTORY, NAME = rr_out-crew
*NODE OUTPUT, NSET = responses-crew
TA
*END STEP
*STEP, INC =          10, NAME = rr_composite-Crew, NLGEOM = YES
rr
*RANDOM RESPONSE
0.0          ,250.0          ,500          ,3.0,          0
*CORRELATION, PSD = psd_tbl_composite, TYPE = UNCORRELATED, COMPLEX = NO
1
,1.0
*MODAL DAMPING, MODAL = DIRECT, DEFINITION = FREQUENCY RANGE ,FIELD = ALL
125.0,          0.03
*BASE MOTION, DOF=2
*OUTPUT, HISTORY, NAME = rr_out-crew
*NODE OUTPUT, NSET = responses-crew
TA
*END STEP

```

A normal random input is defined as in “psd_tbl_composite”, but for white noise as in “psd_tbl_white” where the values are one for all frequencies.

psd_tbl_composite

```
*PSD-DEFINITION, NAME = psd_tbl_composite, TYPE = BASE, G = 1.0
  0.01752,    0.0,    0.0
  0.1709,     0.0,    0.5
  0.1693,     0.0,    1.0
.
.
.
  0.05193,    0.0,    249.0
  0.07069,    0.0,    249.5
  0.05905,    0.0,    250.0
```

psd_tbl_white

```
*PSD-DEFINITION, NAME = psd_tbl_white, TYPE = BASE, G = 1.0
  1.0,    0.0,    0.0
  1.0,    0.0,    0.5
  1.0,    0.0,    1.0
.
.
.
  1.0,    0.0,    249.0
  1.0,    0.0,    249.5
  1.0,    0.0,    250.0
```

Specific attention should be given to the “*PSD-DEFINITION” [2c] parameters TYPE and G.

G

Set this parameter equal to the reference gravity acceleration; for example, 9.81 m/s². The default is G=1.0. This parameter can be used only with TYPE=BASE.

TYPE

Set TYPE=BASE if this frequency function will be used to define a base motion.

Set TYPE=FORCE (default) if this frequency function is given directly in power units.

Set TYPE=DB if this frequency function is defined in decibel units (see below). This option cannot be used with the USER parameter.

The “G” is a scaling factor for the input units. If “G=1” then the output PSD will be in the input PSD units. If “G=9807.0” (e.g. 9807.0 mm/sec²) for an input PSD in units of g’s (g²/Hz), the output will be scaled by G (e.g. scale the output by 1/9807² to get g²/Hz).

LOCATION		NODE NUMBER	DESCRIPTION
Hull	L1	962128	same as squad
	L2	961332	same as squad
	L3	962763	same as squad
	L4	963559	same as squad
Floor	L1	20818183	
	L2	20799251	
	L3	20882620	
Crew Seat	L1	23050495	50th Driver
	L2	23050496	50th Driver
Crew Seat Bkt	L1	7250921	
	L2	7234731	
RB		15696512	same as squad

Figure 1 – Node List

Node locations from [1] are listed in Figure 1. Two test cases are considered for input at RB (node 15696512) and output at the Floor-L1 (node 20818183) and the Crew Seat-L1 (node 23050495). Figures 2 and 3 show the responses (transmissibility, g^2/Hz) due to white noise input.

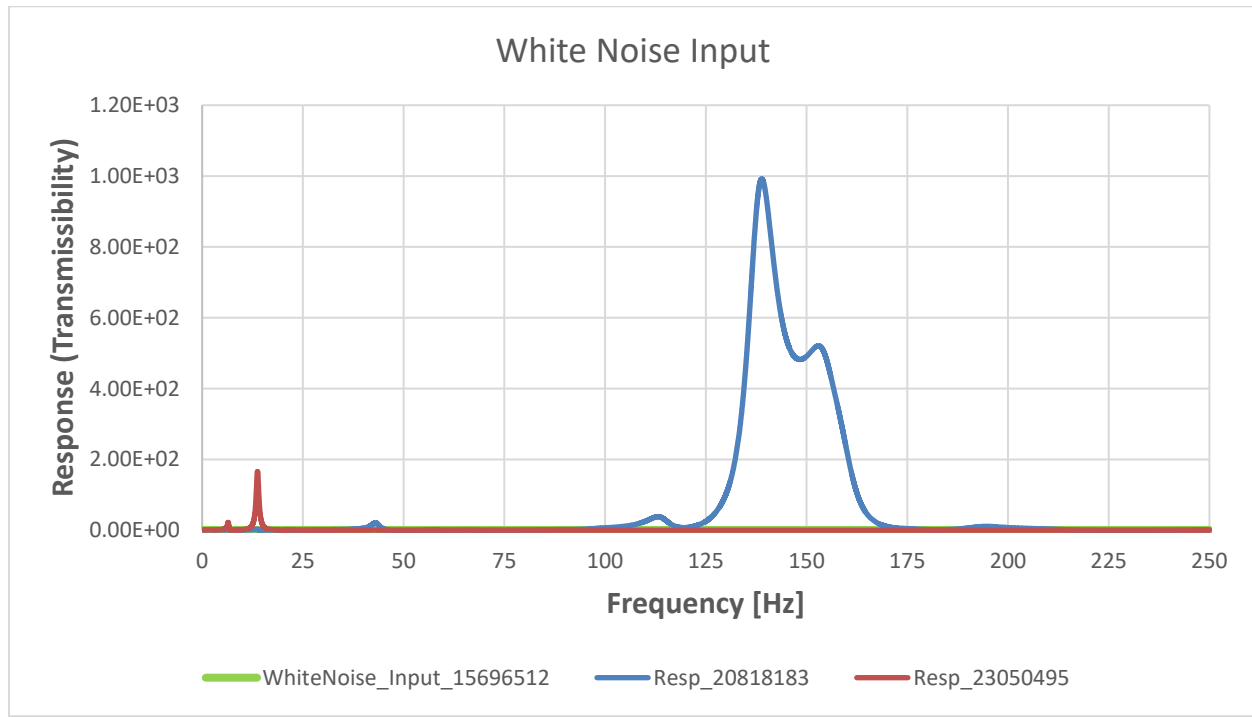


Figure 2 – White Noise Responses

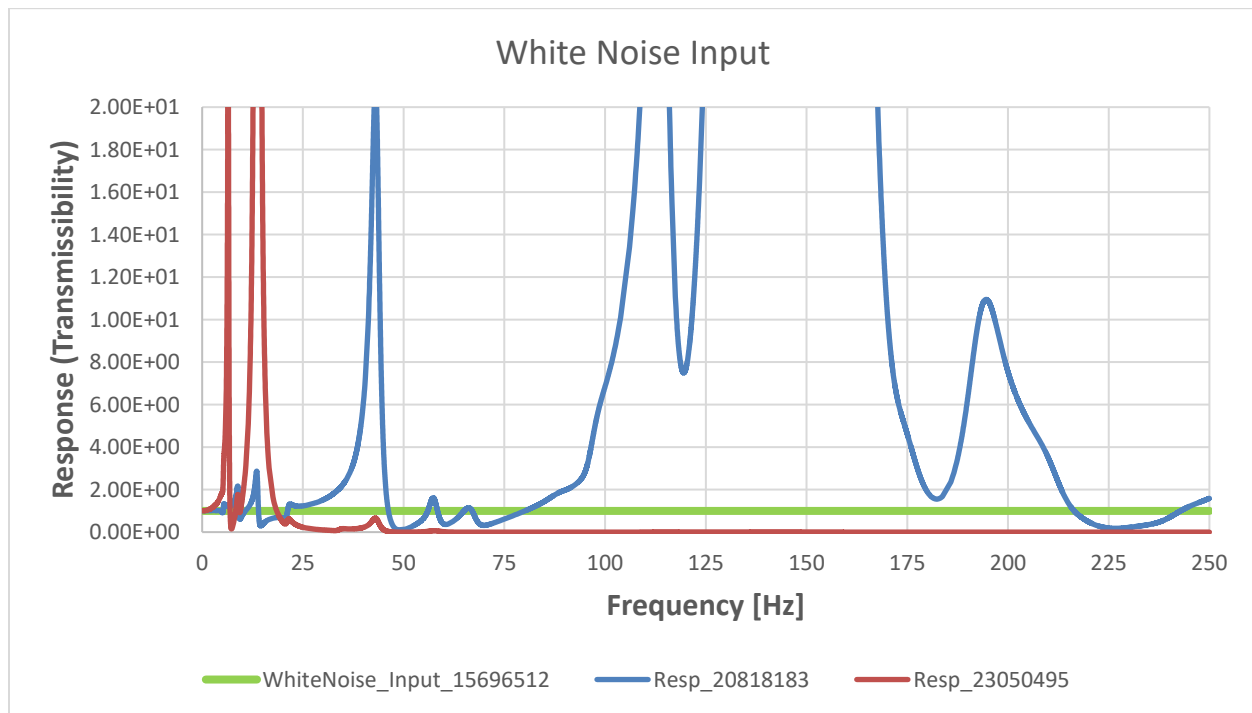


Figure 3 – Zoomed White Noise Responses

Figures 4 and 5 show the responses (g^2/Hz) due to a random input.

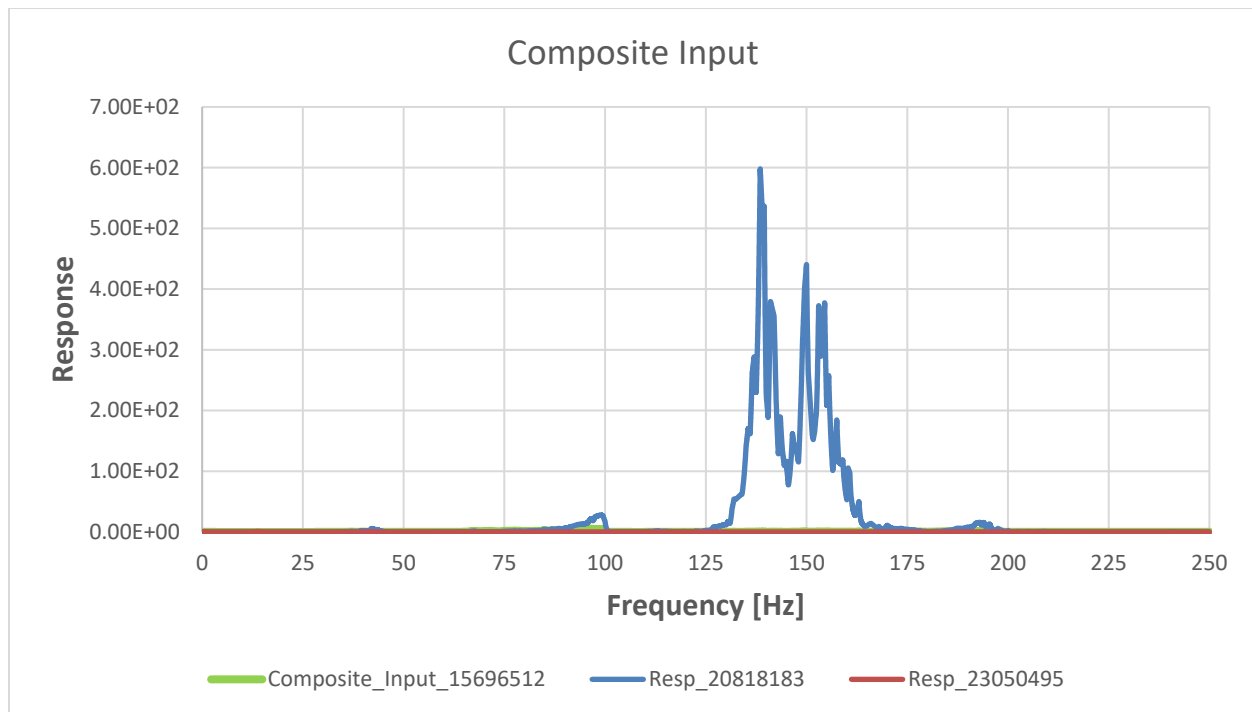


Figure 4 – Composite Responses

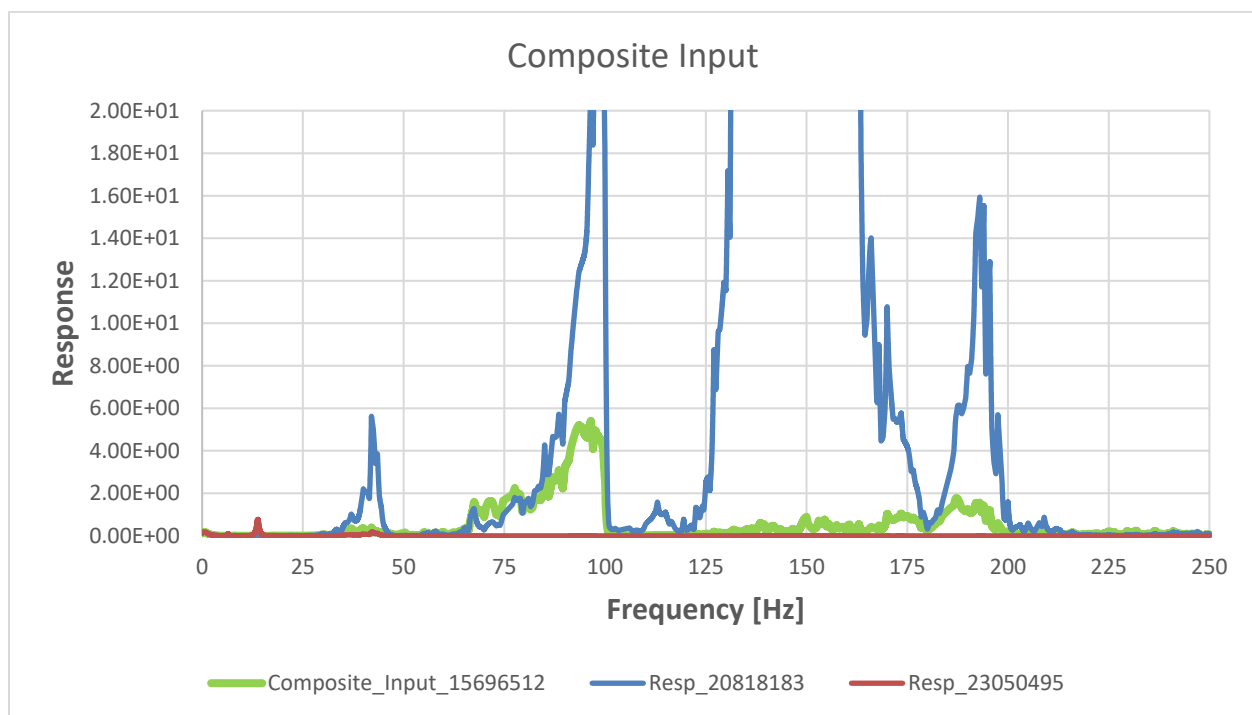


Figure 5 – Zoomed Composite Responses

Figures 6 and 7 show the calculated (transmissibility * random input) responses (g^2/Hz) and the differences from those in Figures 4 and 5.

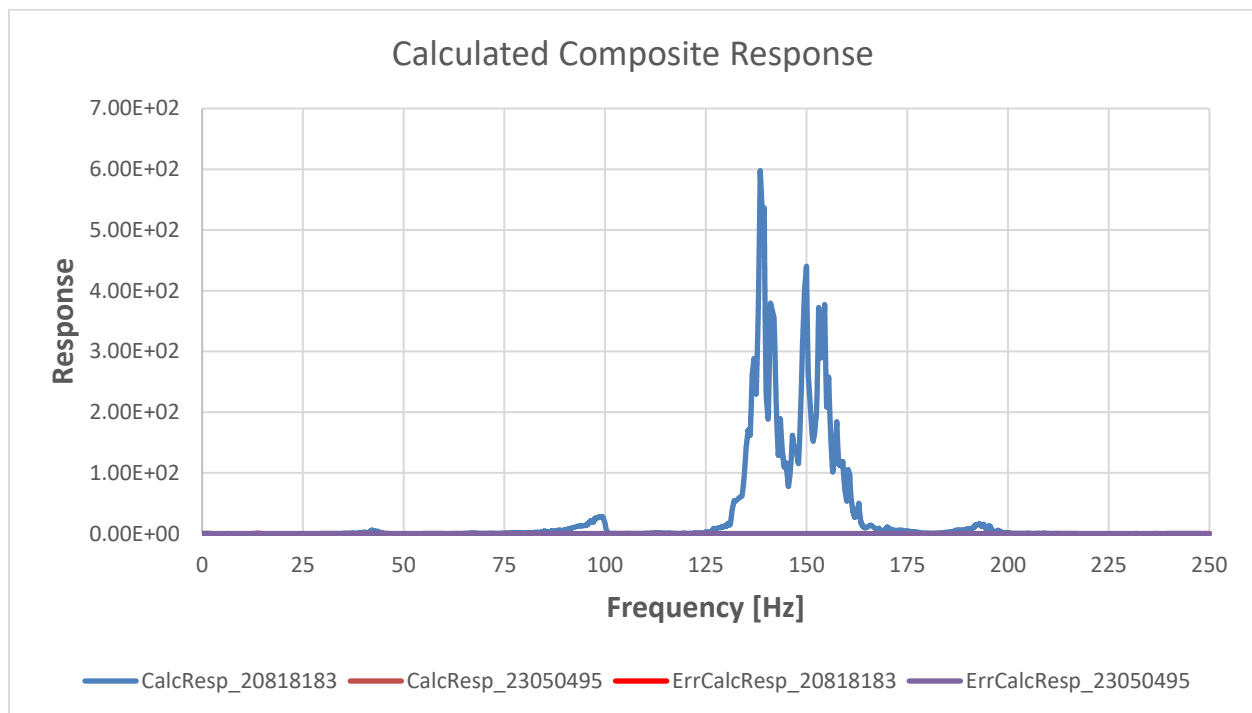


Figure 6 – Calculated Responses and Error

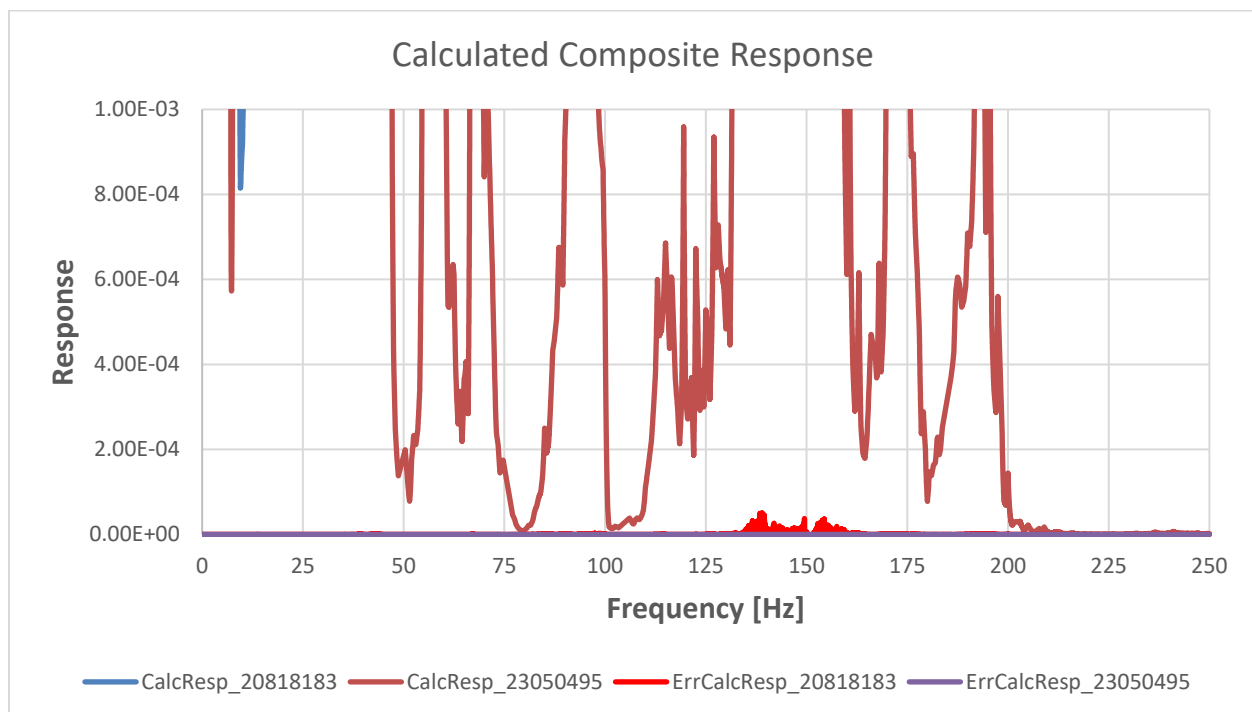


Figure 7 – Zoomed Calculated Responses and Error

Figures 4 and 6 look identical. The maximum error between the response and calculated response was **5.04E-05** for the Floor-L1 (node 20818183) and **6.93E-08** for the Crew Seat-L1 (node 23050495) using Microsoft Excel.

These results indicate the computation of a random output response can be achieved by multiplying the random input by its associated transmissibility PSD.

REFERENCE

- [1] TN18-0801-01 RECOVERABLE ASSET (RA) MODAL AND RANDOM RESPONSE ANALYSIS FOR SQUAD AND CREW CONFIGURATIONS (18-008) (see TR29932 - 2018 Technical Notes and Problem Solutions in Structures. DIST C (DTIC AD 1080 278)).
- [2] ABAQUS/Standard 2019 (<https://www.3ds.com/products-services/simulia/products/abaqus/>) [2019-01-07]

Below references require account at <https://help.3ds.com>

- [2a] Random Response Analysis (<https://help.3ds.com/2019x/english/dsdoc/sima3dxanlrefmap/simaanl-c-randomresponse.htm?contextscope=all>)
- [2b] *RANDOM RESPONSE (<https://help.3ds.com/2019x/english/dsdoc/sima3dxkeyrefmap/simakey-r-randomresponse.htm?contextscope=all>)
- [2c] *PSD-DEFINITION (<https://help.3ds.com/2019x/english/DSDoc/SIMA3DXKEYRefMap/simakey-r-psd-definition.htm?ContextScope=all>)

Above references require account at <https://help.3ds.com>

- [3] White Noise (https://en.wikipedia.org/wiki/White_noise)
- [4] Transmissibility ([https://en.wikipedia.org/wiki/Transmissibility_\(vibration\)](https://en.wikipedia.org/wiki/Transmissibility_(vibration)))

TN 21-0406-01

ABAQUS/STANDARD MODAL ANALYSIS EFFECTIVE MASS AS A PERCENTAGE OF TOTAL EFFECTIVE MASS

April 6, 2021

W. Bylsma

U.S. Army Combat Capabilities Development Command (DEVCOM)
Ground Vehicle Systems Center (GVSC)
Detroit Arsenal
Modeling, Simulation & Software
ATTN: FCDD-GVR-MSS/MS 157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

How do you find the percentage effective mass of the total effective mass in Abaqus/Standard?

PROBLEM SOLUTION:

In modal analysis, it is generally understood that the number of modes used can reasonably represent the structural model if their combined effective mass achieves greater than 90% of the total effective mass. In Abaqus/Standard [1], a “*FREQUENCY” step is used for “eigenvalue extraction to calculate the natural frequencies and corresponding mode shapes of a system.” From [2],

The data file (job-name.dat) is a text file that contains information about the model definition (generated by the analysis input file processor) and, in Abaqus/Standard, tabular output of results. The analysis input file processor information includes the model definition, the history definition, and messages identifying any error and warning conditions that were detected while processing the input data

Example modal result output (eigenvalue output, participation factors, effective mass) is shown at the end of the example Abaqus/Standard “.dat” file output.

A Powershell [3] script, “msum.ps1” (see Appendix), is used to calculate the accumulated percentage of effective mass at each mode by reading the “.dat” file output. The output from processing many “.dat” files can be included in one file by including the “-append” option (e.g. .\msum [file.dat] | out-file -encoding ASCII -append [output.dat]). The “msum.ps1” script relies on the format of the “*FREQUENCY” output in the “.dat” file. It first finds the line with the 'E F F E C T I V E M A S S' pattern and also additional lines below. The number of additional lines is specified, but it is assumed to include the 'TOTAL' line where the effective mass totals are obtained from. For each mode, the percentage of effective mass is calculated and added to the previous sum to get an accumulated percentage of total effective mass. A blank line ends the processing and the accumulated percentages for each mode are output.

Example Abaqus/Standard ".dat" file output

```

.
.
.
      E I G E N V A L U E   O U T P U T

MODE NO      EIGENVALUE      FREQUENCY      GENERALIZED MASS      COMPOSITE MODAL DAMPING
      (RAD/TIME)      (CYCLES/TIME)

      1      -3.53365E+07      0.0000      0.0000      1.0000      0.0000
      2      691.59      26.298      4.1855      1.0000      0.0000
      3      2872.7      53.597      8.5303      1.0000      0.0000
.
.
.
      32      8.05218E+06      2837.6      451.62      1.0000      0.0000
      33      8.90713E+06      2984.5      475.00      1.0000      0.0000
      34      9.42096E+06      3069.4      488.50      1.0000      0.0000

NOTE:
MODE      34 IS A RESIDUAL MODE.

      P A R T I C I P A T I O N   F A C T O R S

MODE NO      X-COMPONENT      Y-COMPONENT      Z-COMPONENT      X-ROTATION      Y-ROTATION      Z-ROTATION

      1      -1.15427E-02      -4.83745E-02      -5.52298E-02      63.131      -13.095      6.1803
      2      -0.22758      -0.43960      -0.32301      720.08      -60.893      -430.44
      3      0.21994      0.17151      -0.32883      400.08      219.04      452.98
.
.
.
      32      -7.74801E-03      -1.32096E-03      -6.10908E-03      5.6069      -7.9998      -5.8912
      33      -8.85974E-03      -1.03029E-03      -2.06247E-02      18.822      -8.3234      -7.5670
      34      -1.55441E-03      2.85869E-03      -1.25373E-02      10.880      1.0011      -2.2760

      E F F E C T I V E   M A S S

MODE NO      X-COMPONENT      Y-COMPONENT      Z-COMPONENT      X-ROTATION      Y-ROTATION      Z-ROTATION

      1      1.33234E-04      2.34009E-03      3.05033E-03      3985.6      171.49      38.196
      2      5.17905E-02      0.19324      0.10433      5.18513E+05      3707.9      1.85276E+05
      3      4.83743E-02      2.94151E-02      0.10813      1.60066E+05      47978.      2.05189E+05
.
.
.
      32      6.00316E-05      1.74495E-06      3.73209E-05      31.437      63.997      34.706
      33      7.84949E-05      1.06150E-06      4.25380E-04      354.25      69.278      57.260
      34      2.41620E-06      8.17209E-06      1.57183E-04      118.38      1.0022      5.1801

TOTAL      0.44765      0.47042      0.44718      8.52443E+05      2.85453E+05      7.10567E+05

      THE ANALYSIS HAS BEEN COMPLETED

      ANALYSIS COMPLETE
      WITH      27 WARNING MESSAGES ON THE DAT FILE
      AND      4 WARNING MESSAGES ON THE MSG FILE
      4 WARNINGS ARE FOR NEGATIVE EIGENVALUES

JOB TIME SUMMARY
USER TIME (SEC)      =      84319.
SYSTEM TIME (SEC)    =      9859.4
TOTAL CPU TIME (SEC) =      94179.
WALLCLOCK TIME (SEC) =      34633

```

MODE NO	PERCENTAGE OF EFFECTIVE MASS (1g, 0 deg elev)						PARTICIPATION FACTORS (1g, 30 deg elev)					
	X	Y	Z	X-ROT	Y-ROT	Z-ROT	X	Y	Z	X-ROT	Y-ROT	Z-ROT
1	0.0603	0.0152	0.0016	0.0026	0.0051	0.0389	-0.1617	0.0811	0.0255	-48.4350	-45.3200	-59.1500
2	0.0673	0.4668	0.2989	0.8384	0.0113	0.0517	-0.0551	-0.4417	-0.3435	83.1400	-9.8970	-1.2380
3	0.3393	0.4881	0.3888	0.8672	0.2485	0.5803	0.3432	0.0960	-0.1889	161.9800	368.4200	586.7300
4	0.3540	0.6058	0.5981	0.8820	0.3861	0.5805	0.0800	0.2256	-0.2882	116.2100	244.8600	12.4880
5	0.5281	0.9465	0.6733	0.8877	0.6618	0.7832	-0.2746	0.3837	-0.1728	-12.0820	-32.5100	-53.3200
6	0.5580	0.9537	0.6893	0.8892	0.6891	0.7944	0.1138	0.0557	-0.0797	37.3980	104.7000	85.1510
7	0.8680	0.9593	0.7714	0.9287	0.9472	0.9361	0.3664	0.0493	0.1806	-99.8100	321.7600	308.7900
8	0.9443	0.9594	0.7715	0.9287	0.9822	0.9729	0.1818	0.0059	-0.0050	0.5747	118.4600	154.8200
9	0.9929	0.9607	0.8978	0.9668	0.9825	0.9928	0.1451	0.0232	-0.2239	186.3200	10.7730	113.9000
10	0.9929	0.9608	0.8993	0.9675	0.9829	0.9929	0.0041	-0.0086	-0.0242	26.2970	-12.7460	6.1713
11	0.9930	0.9608	0.9378	0.9818	0.9897	0.9929	-0.0016	-0.0003	-0.1268	114.1700	-12.1220	-4.3462
12	0.9609	0.9415	0.9825	0.9908	0.9930	0.9930	0.0107	0.0032	0.0258	-24.8030	20.7560	6.8570
13	0.9609	0.9459	0.9840	0.9915	0.9930	0.9930	-0.0006	-0.0002	0.0418	-77.3060	18.2390	-0.8419
14	0.9609	0.9459	0.9840	0.9916	0.9931	0.9931	-0.0095	-0.0010	-0.0010	1.6517	-7.1164	-8.2994
15	0.9609	0.9466	0.9842	0.9917	0.9931	0.9931	-0.0040	0.0028	-0.0170	18.3050	-4.3418	-3.2228
16	0.9610	0.9480	0.9845	0.9919	0.9941	0.9941	0.0318	0.0061	-0.0229	16.9790	9.9988	25.4900
17	0.9610	0.9484	0.9847	0.9920	0.9941	0.9941	-0.0000	-0.0000	-0.0000	4.3807	1.2728	1.2728
18	0.9621	0.9710	0.9910	0.9957	0.9941	0.9941	0.0000	0.0000	0.0000	8.6480	-0.6099	-0.6099
19	0.9732	0.9710	0.9911	0.9958	0.9982	0.9982	0.0100	0.0000	0.0000	5.7081	5.2380	5.2380
20	0.9962	0.9735	0.9774	0.9933	0.9972	0.9983	-0.0000	-0.0000	-0.0000	3.7810	9.4358	9.4358
21	0.9962	0.9768	0.9774	0.9935	0.9972	0.9985	-0.0000	-0.0000	-0.0000	0.0308	12.1070	12.1070
22	0.9963	0.9768	0.9775	0.9935	0.9972	0.9986	0.0000	0.0000	0.0000	2.1052	5.8066	5.8066
23	0.9964	0.9768	0.9775	0.9935	0.9972	0.9987	-0.0072	0.0030	-0.0039	2.3441	0.1588	-5.8899
24	0.9965	0.9769	0.9844	0.9955	0.9980	0.9987	0.0048	0.0057	-0.0521	48.0840	-18.1450	4.5666
25	0.9965	0.9769	0.9845	0.9956	0.9981	0.9987	-0.0010	-0.0002	-0.0078	6.6188	-4.6211	-0.3708
26	0.9973	0.9772	0.9936	0.9981	0.9987	0.9990	-0.0185	-0.0120	0.0602	8.3190	15.8770	-15.1900
27	0.9977	0.9773	0.9936	0.9981	0.9987	0.9993	-0.0134	0.0067	-0.0010	-1.8892	-4.0026	-12.7490
28	0.9991	0.9816	0.9946	0.9990	0.9990	0.9997	-0.0242	-0.0430	-0.0192	28.4090	-10.4540	-6.6890
29	0.9991	0.9823	0.9952	0.9991	0.9992	0.9998	-0.0052	0.0166	-0.0156	8.3153	-8.4254	-7.5567
30	0.9991	0.9826	0.9952	0.9992	0.9992	0.9998	-0.0009	-0.0127	-0.0037	7.5740	0.4023	2.4696
31	0.9996	0.9992	0.9984	0.9994	0.9994	0.9999	0.0144	0.0845	-0.0355	18.6730	-8.4663	9.3185
32	1.0000	0.9999	0.9985	0.9994	0.9997	1.0000	0.0124	0.0175	0.0048	-7.0985	18.2210	5.2434
33	1.0000	0.9999	0.9988	0.9995	0.9997	1.0000	0.0001	0.0041	0.0112	-10.8590	4.9193	-1.0297
34	1.0000	1.0000	0.9989	0.9996	0.9997	1.0000	0.0040	0.0049	0.0059	-5.7156	1.4757	1.7966
35	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0028	0.0045	0.0212	-19.6550	10.3090	1.0227

Figure 1 – Example Usage of Percentage Effective Mass Output

Figure 1 shows how this output can be used (with Microsoft Excel) to determine the minimum number of modes that should be used to represent the response of a structural model.

REFERENCE

- [1] Abaqus/Standard (<https://www.3ds.com/products-services/simulia/products/abaqus/abaqusstandard/>)
- [2] About Output (The data file) (https://help.3ds.com/2019/English/DSSIMULIA_Established/SIMACAEOUTRefMap/simaout-c-output.htm?ContextScope=all&id=36fe80847d51418b80d3772f0e69b20a#Pg0)
- [3] Powershell. (<https://docs.microsoft.com/en-us/powershell/>)

APPENDIX

msum.ps1

```

<#
msum.ps1 - calculate percentage of effective mass for each mode

===BOF - example abaqus/standard .dat *FREQUENCY step output
.
.
.

      E F F E C T I V E   M A S S

MODE NO      X-COMPONENT      Y-COMPONENT      Z-COMPONENT      X-ROTATION      Y-ROTATION      Z-ROTATION
      1      3.26762E-03      3.39755E-03      2.86950E-04      9.2156         378.92         792.43
      2      1.37389E-03      0.19272         0.15535         8.65645E+05    613.92         4226.0
TOTAL        0.45258         0.47148         0.44260         1.00448E+06    4.27782E+05    6.98040E+05
.
.
.
===EOF
#>

param ([string]$fn)                                # .\msum [file.dat] | out-file -encoding ASCII -append [output.dat]

$ptrn1 = 'E F F E C T I V E   M A S S'             # begin pattern
$ptrn2 = 'TOTAL'                                    # total pattern
$nlines = 50                                         # number of lines to search from "E F F E C T I V E   M A S S", make sure
includes "TOTAL" line                               # start line (0 based) after begin pattern
$start_ln = 4

$d = select-string -path $fn -pattern $ptrn1 -context 0,$nlines | foreach {

    $_.line
    $_.context.postcontext
}

write-host $d                                       # output the data
$d | out-file -encoding ASCII ttz
[string] $t = $d | select-string -pattern $ptrn2    # get TOTAL line

write-host $t                                       # output TOTAL line

$ts = ($t.trim()) -split "\s+"                     # split on white space

[double[]] $sum1 = (,0) * $nlines                   # create sum array for each component (X,Y,Z,XR,YR,ZR)
[double[]] $sum2 = (,0) * $nlines
[double[]] $sum3 = (,0) * $nlines
[double[]] $sum4 = (,0) * $nlines
[double[]] $sum5 = (,0) * $nlines
[double[]] $sum6 = (,0) * $nlines

$cnt = 1                                           # index for component sum arrays

for ($i = $start_ln; $i -le $d.length-1; $i = $i + 1) {

    $dt = $d[$i].trim()                             # trim data

    if ($dt.length -eq 0) {                           # a blank line

        $iend = $i - $start_ln                       # number of lines read
        break
    }

    $ds = $dt -split "\s+"                           # split on white space
    $sum1[$cnt] = $sum1[$cnt-1] + $ds[1]/$ts[1]        # accumulate percentage of total
    $sum2[$cnt] = $sum2[$cnt-1] + $ds[2]/$ts[2]
    $sum3[$cnt] = $sum3[$cnt-1] + $ds[3]/$ts[3]
    $sum4[$cnt] = $sum4[$cnt-1] + $ds[4]/$ts[4]
    $sum5[$cnt] = $sum5[$cnt-1] + $ds[5]/$ts[5]
    $sum6[$cnt] = $sum6[$cnt-1] + $ds[6]/$ts[6]
    $cnt = $cnt + 1
}

# output percentage of effective mass for each mode
" `n"
$fn                                                  # output file name

(1..$iend) | foreach {

    ("{0,3:D} {1,8:F4} {2,8:F4} {3,8:F4} {4,8:F4} {5,8:F4} {6,8:F4}" -f $_, $sum1[$_], $sum2[$_], $sum3[$_], $sum4[$_],
    $sum5[$_], $sum6[$_])
}

```

Example msum.ps1 output

r063.dat						
1	0.0003	0.0050	0.0068	0.0047	0.0006	0.0001
2	0.1160	0.4158	0.2401	0.6129	0.0136	0.2608
3	0.2241	0.4783	0.4819	0.8007	0.1817	0.5496
4	0.2332	0.5809	0.6968	0.8300	0.3455	0.5576
5	0.5610	0.9426	0.6976	0.9006	0.6987	0.7970
6	0.8549	0.9621	0.7482	0.9076	0.8793	0.9239
7	0.9661	0.9622	0.7503	0.9090	0.9304	0.9755
8	0.9664	0.9623	0.7978	0.9273	0.9332	0.9756
9	0.9781	0.9645	0.9371	0.9753	0.9658	0.9833
10	0.9880	0.9645	0.9639	0.9862	0.9821	0.9886
11	0.9909	0.9647	0.9740	0.9898	0.9867	0.9904
12	0.9911	0.9647	0.9741	0.9898	0.9869	0.9904
13	0.9911	0.9647	0.9742	0.9899	0.9869	0.9904
14	0.9912	0.9647	0.9742	0.9899	0.9870	0.9905
15	0.9912	0.9649	0.9757	0.9903	0.9870	0.9905
16	0.9914	0.9649	0.9758	0.9903	0.9870	0.9906
17	0.9915	0.9649	0.9764	0.9906	0.9874	0.9906
18	0.9924	0.9758	0.9768	0.9909	0.9876	0.9959
19	0.9945	0.9761	0.9853	0.9949	0.9941	0.9962
20	0.9945	0.9792	0.9853	0.9950	0.9941	0.9964
21	0.9959	0.9792	0.9863	0.9954	0.9954	0.9970
22	0.9961	0.9792	0.9879	0.9961	0.9961	0.9971
23	0.9965	0.9792	0.9899	0.9968	0.9971	0.9973
24	0.9966	0.9792	0.9919	0.9976	0.9977	0.9973
25	0.9966	0.9792	0.9930	0.9981	0.9979	0.9973
26	0.9967	0.9792	0.9930	0.9981	0.9979	0.9973
27	0.9993	0.9792	0.9950	0.9988	0.9991	0.9993
28	0.9993	0.9805	0.9951	0.9990	0.9991	0.9994
29	0.9996	0.9978	0.9980	0.9993	0.9995	0.9997
30	0.9997	0.9990	0.9985	0.9994	0.9995	0.9998
31	0.9997	1.0000	0.9986	0.9994	0.9995	0.9999
32	0.9998	1.0000	0.9987	0.9994	0.9998	0.9999
33	1.0000	1.0000	0.9996	0.9999	1.0000	1.0000
34	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Note the highlighted yellow line indicates that modes 1 to 9 achieve greater than 90% effective mass.

TN 21-0407-01

SENSITIVITY OF FATIGUE LIFE TO CHANGES IN STRESS LEVELS

April 7, 2021

W. Bylsma

U.S. Army Combat Capabilities Development Command (DEVCOM)
Ground Vehicle Systems Center (GVSC)
Detroit Arsenal
Modeling, Simulation & Software
ATTN: FCDD-GVR-MSS/MS 157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

How does a change in stress level affect fatigue life predictions based on an S-N diagram?

PROBLEM SOLUTION:

From [1] a definition is given and described in Figure 1.

A SN-Curve (sometimes written S-N Curve) is a plot of the magnitude of an alternating stress versus the number of cycles to failure for a given material. Typically both the stress and number of cycles are displayed on logarithmic scales.

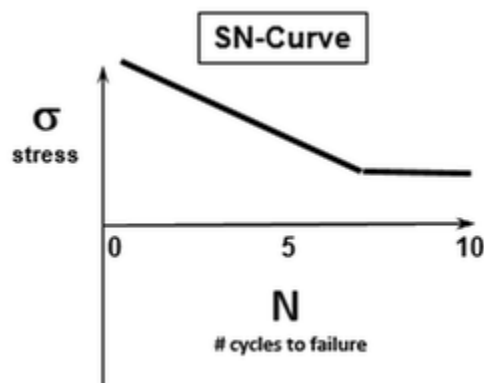


Figure 1 – SN-Curve [1]

From [2a]

$$S_f = aN^b \quad (1a)$$

where

S_f is the fatigue strength (stress level that can withstand N cycles)
 N is the cycles to failure

a, b are coefficients defined by the points ($N=10^3$, S_f) and ($N=10^6$, S_e).

or

$$N = \left(\frac{S_f}{a} \right)^{\frac{1}{b}} \quad (1b)$$

The fatigue strength, S_f , at $N=10^3$ can be denoted as $S_f = f * S_{ut}$, where f is some fractional amount of the ultimate tensile strength, and at $N=10^6$ as S_e , also known as the endurance limit. The coefficients a and b can be found by substituting these points into equation (1a)

$$\begin{aligned} \log S_f &= \log a + b \log 10^3 \\ \log S_e &= \log a + b \log 10^6 \end{aligned}$$

and when solved define

$$b = -\frac{1}{3} \log \frac{S_f}{S_e} \quad (2)$$

$$a = \frac{S_f^2}{S_e} \quad (3)$$

Modifying equation (1b) with changes to the stress level and cycles to failure gives equation (4)

$$N + nN = \left(\frac{S_f + sS_f}{a} \right)^{\frac{1}{b}} \quad (4)$$

where

s is a percentage of S_f

n is a percentage of N

Dividing equation (4) by equation (1b) gives

$$1 + n = (1 + s)^{\frac{1}{b}} \quad (5)$$

and rearranging gives

$$n = (1 + s)^{\frac{1}{b}} - 1 \quad (6)$$

Equation (6) defines the percentage change of the cycles to failure in terms of the percentage change in fatigue strength. Experimentation with approximating the power term using a binomial series truncated to the first order did not provide accurate enough results (see [3], [4], and [5]). See [6] and [7] for details on approximating SN curves for materials.

A plot of equation (6) for various values of b is shown in Figure 2. As the slope increases (b gets larger, $1/b$ gets smaller) the affect or multiplication factor is reduced. With a negative slope (b is negative) a positive percentage change (s) in the stress level gives a negative percentage change (n) in the fatigue life (a reduction). For small slopes (small b), a small change in stress level results in a large change in fatigue life. It should be noted that equation (1a) and (1b) are assumed to be valid only between the points (S_f , at $N=10^3$) and (S_e , at $N=10^6$). A specific example from [2] is included as points 1 and 3 from Table 1.

Table 1 shows values from an SN curve defined in [2]. In the first section, the stress level (S_f) is calculated (yellow) from the fatigue life (N), with the endurance limit (S_e) estimated as $S_f@N=10^6$. In the second section, the fatigue life (N) is calculated (yellow) from the stress level (S_f). The blue and orange regions indicate data points used in section 3. In the third section, the fatigue life ($N*(1+n)$) is calculated based on an adjusted stress level ($S_f*(1+s)$) and the associated percentage fatigue life change (n) is indicated. Points 1 and 3 are shown in Figure 2 for $b=-0.0746$. They indicate that with an initial stress level of 52.48 kips, a 20% and 60% increase in stress level would result in a 91% and 99% reduction in fatigue life, respectively. Point 2, also for $b=-0.0746$ with an initial stress level of 74 kips, indicates that a 10% decrease in stress level would result in a 310% increase in fatigue life.

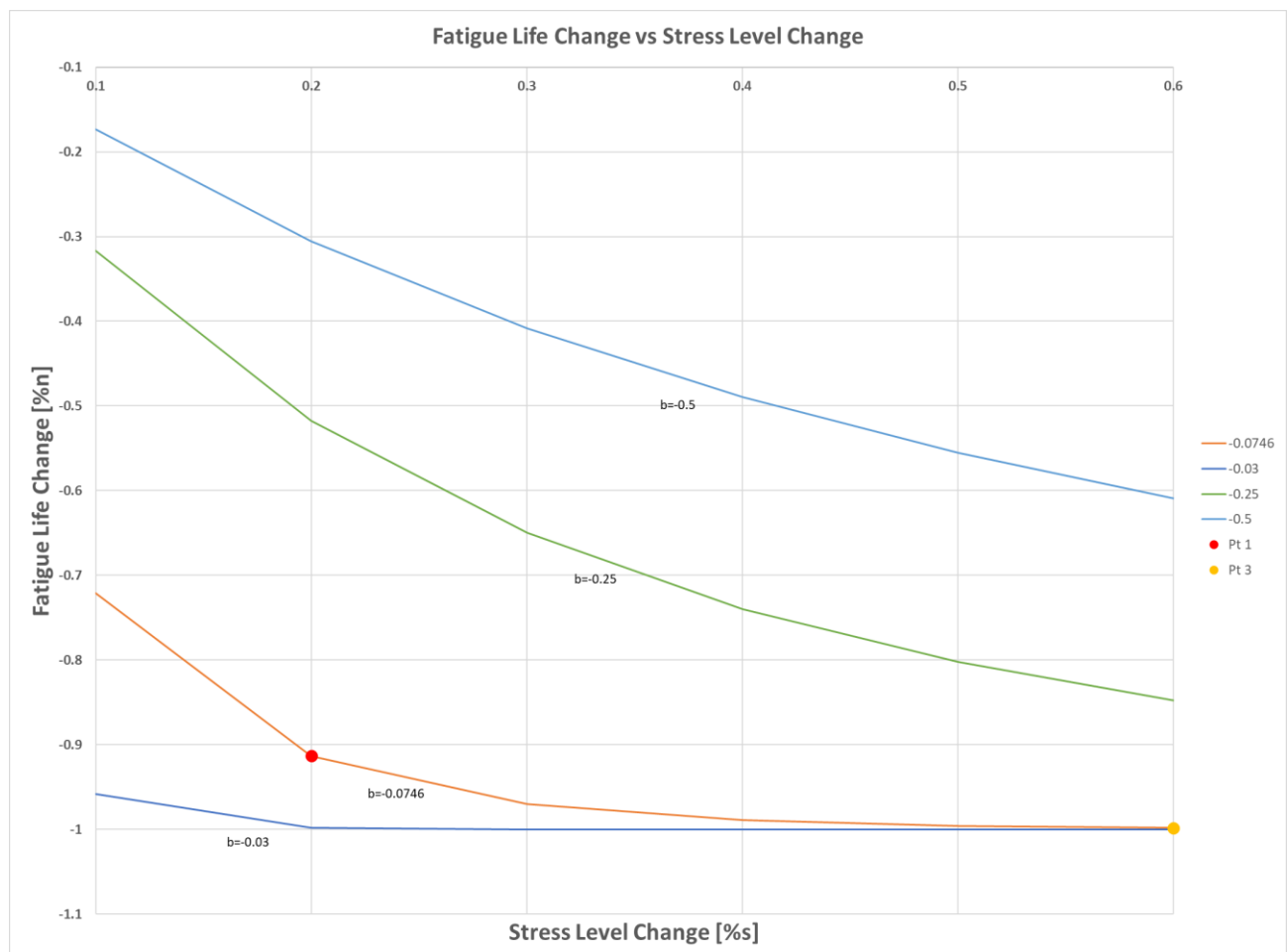


Figure 2 – Percentage Change – Fatigue Life vs. Stress Level

Table 1 – Stress Level and Fatigue Life Calculations

	a	147.1	kpsi			
	b	-0.0746				
1	$S_f = aN^b$					
	N	Sf	Se (Sf @ N=10 ⁶)			
	1,000.0	87.86	52.48	kpsi		
	10,000.0	74.00				
	100,000.0	62.32				
	1,000,000.0	52.48				
2	$N = (S_f/a)^{1/b}$					
	1,000.0	87.86				
	10,000.0	74.00				
	100,000.0	62.32				
	1,000,000.0	52.48				
3			s	n		
	86,813.9	62.98	0.2000	-0.9132		Pt 1
	41,055.5	66.60	-0.1000	3.1055		Pt 2
	1,835.7	83.97	0.6000	-0.9982		Pt 3

The results above indicate the sensitive nature of fatigue life as a function of stress level and SN curve slope (b). Small changes in stress level can result in very large changes in fatigue life. Caution should be used to ensure accurate stress levels when making fatigue life predictions.

REFERENCE

- [1] "What is a SN-Curve?" (<https://community.sw.siemens.com/s/article/what-is-a-sn-curve>)
- [2] Shigley, et. al. Mechanical Engineering Design, 7th Ed. McGraw-Hill (2004). ISBN 0-07-252036-1
[a] Common curve fit, pp. 326.
- [3] Negative Binomial Series (<https://mathworld.wolfram.com/NegativeBinomialSeries.html>)
- [4] Binomial Series (<https://mathworld.wolfram.com/BinomialSeries.html>)
- [5] Generalized Binomial Theorem (https://en.wikipedia.org/wiki/Binomial_theorem#Newton's_generalized_binomial_theorem)

Below references require account at <https://help.3ds.com>

- [6] fe-safe Users Guide (2019). Dassault Systemes. (https://help.3ds.com/2019/English/DSSIMULIA_Established/SIMULIA_Established_FrontmatterMap/sim-c-DSDocFesafe.htm?contextscope=all&id=9d30c899d9f8445fa2ee4c92e61118ba).
[a] Technical Note TN-001: Approximating materials data (PDF page 435).
- [7] fe-safe Fatigue Theory Reference (2019). Dassault Systemes. (https://help.3ds.com/2019/English/DSSIMULIA_Established/FesafeFatigueTheoryPdf/FesafeFatigueTheory.pdf?contextscope=all&id=0f94ff6de3084b0086f42fa1ff90d767).
[a] Figure 3.2 Stress-life relationship, pg. 3-2 (PDF page 64).
[b] 3.5 Approximations for other materials, pg. 3-8 (PDF page 70)
[c] fatigue strength exponent, b, usually $(-0.12 < b < -0.05)$. pg. 3-9 (PDF page 71)

TN 21-0723-01

MARGIN OF SAFETY CRITERIA FOR FASTENER PRELOAD/CLAMP FORCE REQUIREMENT

July 23, 2021

W. Bylsma

U.S. Army Combat Capabilities Development Command (DEVCOM)
Ground Vehicle Systems Center (GVSC)
Detroit Arsenal
Modeling, Simulation & Software
ATTN: FCDD-GVR-MSS/MS 157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

What is the margin of safety for fastener preload and what is the required preload to prevent joint slippage?

PROBLEM SOLUTION:

There are many details to fastener analysis (see references [1] to [4]). If they are assessed individually (see [1]) then to prevent joint slippage requires the condition in (1) to hold

$$(F_{PL} - F_{TL}C)f > F_{SL} \quad (1)$$

where

PL = Preload/Clamp Load

TL = Tensile Load (assumes TL is positive, otherwise see [18])

C = fraction of external load carried by fastener (C=1 for no fastener stiffness consideration)

f = surface friction coefficient

SL = shear load.

Adding a scale factor to F_{PL} to make equality true in (2)

$$(xF_{PL} - F_{TL}C)f = F_{SL} \quad (2)$$

where

x = scale factor (multiplication factor of PL to meet no slippage condition in (1))

then

$$x = \frac{\frac{F_{SL}}{f} + F_{TL}C}{F_{PL}} = \frac{R_{S/P}}{f} + R_{T/P}C \quad (3)$$

with

$R_{S/P}$ = Ratio Shear to Preload
 $R_{T/P}$ = Ratio Tensile to Preload

Then

$$MS = \frac{\text{allowable}}{\text{actual}} - 1 = \frac{F_{PL}}{xF_{PL}} - 1 = \frac{1}{x} - 1 \quad (4)$$

is the preload margin of safety (MS). However, in (4)

$$\frac{1}{x} = \frac{F_{PL}}{\frac{F_{SL}}{f} + F_{TL}C} = \frac{fF_{PL}}{F_{SL} + fCF_{TL}} \quad (5)$$

which makes (4) equivalent to Eq. 86 in [1].

Sensitivity of x to C gives

$$\frac{dx}{dC} = \frac{F_{TL}}{F_{PL}} = R_{T/P} \quad (6)$$

where the change in x due to C can be estimated by

$$\int_{x_1}^{x_2} dx = R_{T/P} \int_{C_1}^{C_2} dC \quad (7)$$

resulting in

$$x_2 = x_1 + (C_2 - C_1)R_{T/P} \quad (8).$$

From (8), x decreases if C_2 is less than C_1 . Similarly for friction,

$$\frac{dx}{df} = \frac{-\frac{F_{SL}}{f^2}}{F_{PL}} = -\frac{R_{S/P}}{f^2} \quad (9)$$

where the change in x due to f can be estimated by

$$\int_{x_1}^{x_2} dx = -R_{S/P} \int_{f_1}^{f_2} \frac{1}{f^2} df \quad (10)$$

resulting in

$$x_2 = x_1 + \left(\frac{1}{f_2} - \frac{1}{f_1}\right)R_{S/P} \quad (11).$$

From (11), x increases if f_2 is less than f_1 , and decreases if greater than f_1 . Combining both effects

$$x_2 = x_1 + (C_2 - C_1)R_{T/P} + \left(\frac{1}{f_2} - \frac{1}{f_1}\right)R_{S/P} \quad (12).$$

Generally, x is more sensitive to the friction coefficient. Table 1 shows examples of calculations using (3), (4), and (12).

Table 1 – Example Preload MS and Scale Factor (x)

FASTENER FORCES						M.S			x			PREDICTED		
MAX TENSILE		MAX COMB. SHEAR			PRELOAD ⁵⁻¹¹	p_l^{12}	p_l^{12}	p_l^{12}	x_1	x_2	x_3	x_2 frm x_1	x_3 frm x_2	x_3 frm x_1
[lbf]	$R_{T/P}$	[N]	[lbf]	$R_{S/P}$	[lbf]	$C=1.0$ $f=0.3$	$C=0.36$ $f=0.3$	$C=0.36$ $f=0.6$				Equ. 12	Equ. 12	Equ. 12
297	0.015	3,894	877	0.043	20,300	+5.30	+5.70	+11.94	0.16	0.15	0.08	0.15	0.08	0.08
4,077	0.201	73,335	16,517	0.814	20,300	-0.66	-0.64	-0.30	2.91	2.78	1.43	2.78	1.43	1.43
4,106	0.202	64,166	14,452	0.712	20,300	-0.61	-0.59	-0.21	2.58	2.45	1.26	2.45	1.26	1.26
2,081	0.103	20,058	4,518	0.223	20,300	+0.18	+0.28	+1.45	0.84	0.78	0.41	0.78	0.41	0.41
725	0.036	14,156	3,188	0.157	20,300	+0.79	+0.86	+2.64	0.56	0.54	0.27	0.54	0.27	0.27
1,326	0.065	23,071	5,196	0.256	20,300	+0.09	+0.14	+1.22	0.92	0.88	0.45	0.88	0.45	0.45
5,268	0.260	16,627	3,745	0.184	20,300	+0.14	+0.41	+1.49	0.87	0.71	0.40	0.71	0.40	0.40
9,838	0.485	38,519	8,675	0.427	20,300	-0.48	-0.37	+0.13	1.91	1.60	0.89	1.60	0.89	0.89
16,000	0.788	73,483	16,550	0.815	20,300	-0.71	-0.67	-0.39	3.51	3.00	1.64	3.00	1.64	1.64
8,088	0.398	35,514	7,999	0.394	20,300	-0.42	-0.31	+0.25	1.71	1.46	0.80	1.46	0.80	0.80

Determination of C depends on the fastener and joint stiffness. Calculations rely mainly on [5] and [2]. The joint stiffness is

$$k = \frac{\pi E d \tan \alpha}{\ln \frac{(2t \tan \alpha + D - d)(D + d)}{(2t \tan \alpha + D + d)(D - d)}} \quad (13)$$

where

E = Modulus of Elasticity of the layer
 d = basic major diameter of fastener
 α = half-apex angle
 t = frustum thickness
 D = frustum starting diameter

and k “must be solved separately for each frustum in the joint. Then individual stiffness’s are assembled to obtain k using Equ. 8-18” [5, pg. 415]. Then

$$\frac{1}{k} = \frac{1}{k_1} + \frac{1}{k_2} + \frac{1}{k_3} + \dots \quad (14).$$

Reference [4] provides other ways to compute the frustum area (see Appendix A.1).

For the fastener stiffness (threaded and unthreaded portions)

$$\frac{1}{k} = \frac{1}{k_1} + \frac{1}{k_2} \text{ or } k = \frac{k_1 k_2}{k_1 + k_2} \quad (15)$$

$$k = \frac{AE}{l} \quad (16)$$

where

k = stiffness
 A = area
 E = modulus of elasticity
 l = length

 k_t = threaded portion
 A_t = tensile-stress area = $\pi(d/2)^2$
 l_t = length of threaded portion of grip
 k_d = unthreaded portion
 A_d = major-diameter area of fastener
 l_d = length of unthreaded portion in grip

From [9] the tensile stress area (A_s) is

$$A_s = 0.7854 \left[D - \left(\frac{0.9743}{1/P} \right) \right]^2 \quad (17)$$

where

D = basic major diameter
 $n=1/p$ = number of threads per inch

Table 2 shows examples of the joint and fastener stiffness calculations using (13) to (17). Here the screw washer thickness has been neglected and the bottom layer is the only threaded portion (so $L_g = h + d/2$, see [5, Table 8-7 Suggested Procedure for finding Fastener Stiffness, pg. 412] for determining the effective grip. If the threaded portion is the same as the length, use a small delta ($L_t = L - 1e-12$) to avoid divide by zero, although the stiffness will be very high but cancel out in the final sum ($1/Inf = 0$).

A Powershell [16] script “frust.ps1” is provided in Appendix A.2 that calculates frustum thickness, diameter, and stiffness parameters for joint stiffness calculation to verify the spreadsheet calculations. It requires data for the layer thickness [in], Modulus of Elasticity [ksi], and joint effective grip [in], fastener basic diameter [in], and “washer” diameter [in]. It returns the frustum thickness layers [in] with diameter [in] and stiffness (including at peak and effective grip) and total stiffness [Mlbf/in].

A Powershell script “bstiff.ps1” is provided in Appendix A.3 that calculates bolt/screw stiffness for threaded/unthreaded portions to verify the spreadsheet calculations. It requires data for the fastener description, basic major diameter [in], threads per inch ($n=1/pitch$) [./in], elasticity of fastener [ksi], fastener length [in], fastener thread length [in], and effective grip [in]. It returns the fastener unthreaded, threaded, and combined stiffness [Mlbf/in].

Example output from “frust.ps1” and “bstiff.ps1” follow Table 2.

Table 2b – Example Joint/Fastener Stiffness Calculations

DESCRIPTION					SYMBOL	UNITS	Test				Test	NOTES	REFERENCE			
JOINT MEMBERS																
BOLT/SCREW						Case		shk_mnt	shk_guard	dbl_roll	dbl_roll_R01	ra_hse	[1] pg. 427	[1] pg. 432		
Description								5/8-11 UNC	5/8-11 UNC	5/8-11 UNC	5/8-11 UNC	5/8-11 UNC	5/8-11 UNC	5/8-11 UNC		
Basic Major Diameter (D)						D	in	0.625	0.625	0.625	0.625	0.625	0.625	0.625		
Threads per Inch						n=1/pitch	/in	11	11	11	11	11	11	11		
Elasticity of Fastener						E	ksi	29000	29000	29000	29000	29000	29000			see [2]
Length						L	in	1.75	3.25	1.75	1.75	1.75	2.25	1		see ASME B1.1, Appendix B [3a]
Length Threads						Lt	in	1.5	1.5	1.5	1.5	1.5	1.5	1.0000E+00	1=1 - 1e-12 to avoid DIV#0	see ASME B1.1, Appendix B [3a]
Length of Unthreaded Portion in Grip						ld	in	0.2500	1.7500	0.2500	0.2500	0.2500	0.7500	0.0000	L - lt	see Shigley [1c]
Major Diameter of Area of Fastener						Ad	in^2	0.3068	0.3068	0.3068	0.3068	0.3068	0.3068	0.3068	Pt*(D/2)^2	
Unthreaded Stiffness						kd	Mlb/in	35.6	5.1	35.6	35.6	35.6	11.9	8.90E+12	(Ad * E) / ld	see Shigley [1a]
Length of Threaded Portion of Grip						lt	in	0.8125	0.5625	0.7125	0.3125	0.7125	0.7500	1.0000	Lg - ld	see Shigley [1c]
Tensile Stress Area (As)						At	in^2	0.2260	0.2260	0.2260	0.2260	0.2260	0.2260	0.2260	f(D,n)	see ASME B1.1, Appendix B [3c]
Threaded Stiffness						kt	Mlb/in	8.07	11.65	9.20	20.97	9.20	8.74	6.55	(At * E) / lt	see Shigley [1a]
FRUSTUM																
Half-Apex Angle						alpha	deg	30.0000	30.0000	30.0000	30.0000	30.0000	30.0000	30.0000		see Shigley [1d]
						tan(alpha)		0.5774	0.5774	0.5774	0.5774	0.5774	0.5774	0.5774		
Thickness 1						ft1	in	0.53125	1.15625	0.48125	0.25000	0.48125	0.75000	0.50000		see Shigley [1d]
Thickness 2						ft2	in	0.21875	0.09375	0.16875	0.03125	0.16875	0.75000	0.18750		
Thickness 3						ft3	in	0.31250	0.75000	0.31250	0.28125	0.31250		0.31250		
Thickness 4						ft4	in		0.31250							
Total Effective Member Thickness							in	1.06250	2.31250	0.96250	0.56250	0.96250	1.50000	1.00000	Lg	
Diameter 1						D1	in	0.93750	0.93750	0.93750	0.93750	0.93750	0.93750	0.93750	1.5*D	see Shigley [1d]
Diameter 2						D2	in	1.29834	2.16437	1.29834	1.22618	1.29834	0.93750	1.29834		
Diameter 3						D3	in	0.93750	1.29834	0.93750	0.93750	0.93750		0.93750		
Diameter 4						D4	in		0.93750							
Modulus of Elasticity 1						fe1	ksi	29700.0	29700.0	27000.0	27000.0	29700.0	14000.0000	30000.0		see [5]
Modulus of Elasticity 2						fe2	ksi	29700.0	29700.0	27000.0	10300.0	29700.0	14000.0000	30000.0		
Modulus of Elasticity 3						fe3	ksi	10300.0	29700.0	10300.0	10300.0			16000.0		
Modulus of Elasticity 4						fe4	ksi		10300.0							
Stiffness 1						k1	Mlb/in	44.59	32.22	42.66	63.14	46.92	17.90	46.46		see Shigley [1e]
Stiffness 2						k2	Mlb/in	172.52	1126.59	194.13	299.50	213.55	17.90	197.58		
Stiffness 3						k3	Mlb/in	20.86	73.98	20.86	22.29	20.86		32.40		
Stiffness 4						k4	Mlb/in		20.86							
STIFFNESS																
Bolt Stiffness						kb	Mlb/in	6.58	3.54	7.31	13.20	7.31	5.03	6.55	1/k = 1/k1 + 1/k2 + ...	see Shigley [1b]
Joint Stiffness						km	Mlb/in	13.13	10.71	13.06	15.62	13.52	8.95	17.41	1/k = 1/k1 + 1/k2 + ...	see Shigley [1e]
Fraction on Bolt						C		0.3337	0.2484	0.3588	0.4580	0.3509	0.3598	0.2735		see Shigley [1f]
Fraction on Members						1-C		0.6663	0.2516	0.6412	0.5420	0.6491	0.6402	0.2765		

Table 2c– Example Joint/Fastener Stiffness Calculations

REFERENCES									
1	Shigley, et. al. Mechanical Engineering Design, 7th Ed. McGraw-Hill (2004). ISBN 0-07-252036-1.								
	> $k = AE/l$, pg. 187								
	> 8-4 Joints-Fastener Stiffness, pg. 410								
	> Lg, Table 8-7, pg. 412								
	> 8-5 Joints-Member Stiffness, pg. 413-415								
	> k, Cone Frusta, pg. 414								
	> $C = kb / (kb + km)$, pg. 421								
2	Steel, ASTM A36 Plate (for reference, http://www.matweb.com)								
	> Modulus of Elasticity 29,000 ksi								
3	ASME B1.1-2019 Unified Inch Screw Threads (UN, UNR, and UNI Thread Forms)								
	> Table 2A - Limits of Size for Standard Series External Thread (UN, UNR, and UNI)								
	> Table 2B - Limits of Size for Standard Series Internal Threads (UN and UNI)								
	> Appendix B - Thread Strength Design Formulas, pg. 150								
4	2.72 Elements of Mechanical Design, MIT OpenCourseWare, Spring 2009								
	> (https://ocw.mit.edu/courses/mechanical-engineering/2-72-elements-of-mechanical-design-spring-2009/lecture-notes/MIT2_72s09_lec10.pdf)								
	> Threads should be at least 1.5 D deep for bolt to reliably hold a load								
5	Material Properties								
	> Aluminum, 5083-H321 (http://www.matweb.com/)								
	> Steel, GR120-95, Quenched and Tempered, ASTM A148 (https://www.sawbrooksteel.com/sawbrook-steel-brochure.pdf)								
	> Steel, AISI 4130, Quenched and Tempered (http://www.matweb.com/)								
	> Steel, AISI 4140, Quenched and Tempered (http://www.matweb.com/)								
	> Steel, ASTM A36 Plate (for reference, http://www.matweb.com)								
6	Ubrako Engineering Guide (http://www.unbrako.com.au/images/downloads/engg_guide.pdf)								
	> Joint Diagrams, pp. 56 - 61								
7	Preloaded Joint Analysis Methodology for Space Flight Systems. NASA Technical Memorandum 106943 (Dec 1995). DIST A (https://ntrs.nasa.gov/citations/19960012183)								

An outline of “frust.ps1” is:

```

loop cases
loop thickness
    if at beginning (index = 0)
        output frustum starting diameter (dwa)
        previous layer frustum diameter = frustum start diameter (dwa)
        calculate layer frustum diameter (diamf dwa ta alpha)
    else
        previous layer frustum diameter = current layer frustum diameter (tdiam)
    sum layer thickness
    if not reached frustum peak (sum < lga/2)
        calculate layer frustum diameter (diamf tdiamprev ta alpha)
        calculate layer stiffness (kstiff Ea ta da tdiamprev alpha)
        sum layer stiffness (sumk = sumk + 1/k)
    else
        calculate thickness delta to end of layer (del = sum - lga/2, used directly for thickness after peak)
        calculate layer frustum diameter (diamf dwa (lga/2 - del) alpha)
    if peak (do once to show peak)
        if at begininng (index = 0)
            calculate layer stiffness (kstiff Ea (lga/2) da dwa alpha)
            sum layer stiffness (sumk = sumk + 1/k)
        else
            calculate layer stiffness (kstiff Ea (lga/2 - (sum - layer thickness)) da tdiamprev alpha)
            sum layer stiffness (sumk = sumk + 1/k)
        if sum greater than grip (sum > grip)
            # reached effective grip (at end)
        else
            calculate stiffness from peak to next layer (kstiff Ea del da tdiam alpha)
            sum layer stiffness (sumk = sumk + 1/k)
        if not at effective grip yet (sum < lga)
            if passed frustum peak (-not flg_pk)
                calculate layer stiffness (kstiff Ea ta da tdiam alpha)
                sum layer stiffness (sumk = sumk + 1/k)
            else
                output layer data
        else
            # reached effective grip (at end)
            calculate thickness to grip (dele = ta - (sum - lga), thickness to lg)
            calculate layer tickness to grip (te = sum - ta + dele, total layer thickness at grip)
            if lg (do once to show grip)
                if peak (do once to show peak)
                    calculate layer stiffness (kstiff Ea (lga - lga/2) da dwa alpha)
                    sum layer stiffness (sumk = sumk + 1/k)
                else
                    calculate layer stiffness (kstiff Ea dele da dwa alpha)
                    sum layer stiffness (sumk = sumk + 1/k)
                # reached effective grip
                set flg_lg = false
            output layer data
            # past frustum peak
            set flg_pk = false
        # thickness loop
        output total stiffness (in series, 1/k = 1/k1 + 1/k2 + ...)
# case loop

```

An outline of “bstiff.ps1” is:

```
loop cases
# unthreaded portion calculations
# threaded portion calculations
# combined stiffness (in series,  $1/k = 1/k1 + 1/k2 + \dots$ )
# output parameters
# output stiffness
# case loop
```


Example (frust.ps1 output)

```

7 Cases

===0:shk_mnt===
  LG 1.06250
  dw 0.93750
  D 1.55093

    SUM          E      THK          D      K
0.00000 <----->
          PK      29700.0          0.53125 0.93750 44.593
          29700.0          0.21875 1.55093 172.523
0.75000 <-----> 29700.0 0.75000 1.29834
1.06250 <-----> LG      10300.0 0.31250 0.93750 20.855
1.68330 <-----> 10300.0 0.93330          13.128

===1:shk_guard===
  LG 2.31250
  dw 0.93750
  D 2.27262

    SUM          E      THK          D      K
0.00000 <----->
          PK      29700.0          1.15625 0.93750 32.222
          29700.0          0.09375 2.27262 1126.589
1.25000 <-----> 29700.0 1.25000 2.16437
2.00000 <-----> 29700.0 0.75000 1.29834 73.977
2.31250 <-----> LG      10300.0 0.31250 0.93750 20.855
2.93300 <-----> 10300.0 0.93300          10.708

===2:dbl_roll===
  LG 0.96250
  dw 0.93750
  D 1.49320

    SUM          E      THK          D      K
0.00000 <----->
          PK      27000.0          0.48125 0.93750 42.657
          27000.0          0.16875 1.49320 194.133
0.65000 <-----> 27000.0 0.65000 1.29834
0.96250 <-----> LG      10300.0 0.31250 0.93750 20.855
1.58330 <-----> 10300.0 0.93330          13.064

===3:dbl_roll_R01===
  LG 0.56250
  dw 0.93750
  D 1.26226

    SUM          E      THK          D      K
0.00000 <----->
          PK      27000.0 0.25000          1.22618 63.142
          10300.0          0.03125 1.26226 299.504
0.56250 <-----> LG      10300.0 0.28125 0.93750 22.294
1.18330 <-----> 10300.0 0.93330          15.618

===4:ra_hse===
  LG 0.96250
  dw 0.93750
  D 1.49320

    SUM          E      THK          D      K
0.00000 <----->
          PK      29700.0          0.48125 0.93750 46.922
          29700.0          0.16875 1.49320 213.547
0.65000 <-----> 29700.0 0.65000 1.29834
0.96250 <-----> LG      10300.0 0.31250 0.93750 20.855
1.83300 <-----> 10300.0 1.18300          13.524

===5:pg427===
  LG 1.50000
  dw 0.93750
  D 1.80353

    SUM          E      THK          D      K
0.00000 <----->
          PK      14000.0          0.75000 0.93750 17.904
          14000.0          0.00000 1.80353 ?
0.75000 <-----> 14000.0 0.75000 1.80353
1.50000 <-----> LG      14000.0 0.75000 0.93750 17.904
1.50000 <-----> 14000.0          8.952

===6:pg432===
  LG 1.00000
  dw 0.93750
  D 1.51485

    SUM          E      THK          D      K
0.00000 <----->
          PK      30000.0          0.50000 0.93750 46.460
          30000.0          0.18750 1.51485 197.580
0.68750 <-----> 30000.0 0.68750 1.29834
1.00000 <-----> LG      16000.0 0.31250 0.93750 32.397
1.31250 <-----> 16000.0 0.62500          17.406

===>Done.

```

Example (bstiff.ps1 output)

```

7 Cases

===0:shk_mnt:5/8-11 UNC===
  D      0.62500
  n      11.00000
  E 29000.00000
  L      1.75000
  Lt     0.81250
  Lg     1.06250

  BOLT    UNTHREAD    THREADED
6.57601   35.58835     8.06654

===1:shk_guard:5/8-11 UNC===
  D      0.62500
  n      11.00000
  E 29000.00000
  L      3.25000
  Lt     0.56250
  Lg     2.31250

  BOLT    UNTHREAD    THREADED
3.53960   5.08405     11.65167

===2:dbl_roll:5/8-11 UNC===
  D      0.62500
  n      11.00000
  E 29000.00000
  L      1.75000
  Lt     0.71250
  Lg     0.96250

  BOLT    UNTHREAD    THREADED
7.30939   35.58835     9.19868

===3:dbl_roll_R01:5/8-11 UNC===
  D      0.62500
  n      11.00000
  E 29000.00000
  L      1.75000
  Lt     0.31250
  Lg     0.56250

  BOLT    UNTHREAD    THREADED
13.19619   35.58835    20.97300

===4:ra_hse:5/8-11 UNC===
  D      0.62500
  n      11.00000
  E 29000.00000
  L      1.75000
  Lt     0.71250
  Lg     0.96250

  BOLT    UNTHREAD    THREADED
7.30939   35.58835     9.19868

===5:pg427:5/8-11 UNC===
  D      0.62500
  n      11.00000
  E 29000.00000
  L      2.25000
  Lt     0.75000
  Lg     1.50000

  BOLT    UNTHREAD    THREADED
5.03195   11.86278     8.73875

===6:pg432:5/8-11 UNC===
  D      0.62500
  n      11.00000
  E 29000.00000
  L      1.00000
  Lt     1.00000
  Lg     1.00000

  BOLT    UNTHREAD    THREADED
6.55406  8897285392994.02000  6.55406

==>Done.

```

REFERENCE

- [1] Requirements for Threaded Fastening Systems in Spaceflight Hardware. NASA-STD-5020 Rev A (2019-02-11). DIST A. Approved for public release: distribution unlimited. (<https://standards.nasa.gov/standard/nasa/nasa-std-5020>)
[a] access slip individually at each fastener, Eq. 86, pg. 74.
- [2] Chambers. Preloaded Joint Analysis Methodology for Space Flight Systems. NASA Technical Memorandum 106943 (Dec 1995). (<https://ntrs.nasa.gov/citations/19960012183>)
- [3] Barrett. Fastener Design Manual. NASA Reference Publication 1228 (March 1990). (<https://ntrs.nasa.gov/citations/19900009424>)
- [4] Ubrako Engineering Guide (<https://unbrako.com/images/downloads/engguide.pdf>)
[1] Joint Diagrams, pp. 56 – 61
- [5] Shigley, et. al. Mechanical Engineering Design, 7th Ed. McGraw-Hill (2004). ISBN 0-07-252036-1.
- [6] Structural Design and Test Factors of Safety for Spaceflight Hardware. NASA-STD-5001B (2016-04-04) (https://standards.nasa.gov/sites/default/files/nasa-std-5001b_w_change_1.pdf)
[a] Margin of Safety (MS), pg. 10
- [7] NASA Engineering and Safety Center Technical Bulletin No. 12-02 Structural Analyses and Margins of Safety. (https://www.nasa.gov/sites/default/files/files/671748main_NESCTB12-02_StructuralAnalysesandMarginsofSafety.pdf)
- [8] Raju, Lee, and Mohaghegh: "Negative Stress Margins - Are They Real?", AIAA-2011-1808-588, Paper Presented at the 52nd AIAA SDM conference, Denver, Colorado, April 4-7, 2011. (<https://ntrs.nasa.gov/api/citations/20110010003/downloads/20110010003.pdf>)
- [9] ASME B1.1-2019 Unified Inch Screw Threads (Un, UNR, and UNJ Thread Forms)
[a] Table 2A - Limits of Size for Standard Series External Thread (UN, UNR, and UNJ)
[b] Table 2B - Limits of Size for Standard Series Internal Threads (UN and UNJ)
[c] Appendix B - Thread Strength Design Formulas, pg. 150
- [10] Machinery's Handbook, 31st Ed. Industrial Press (2020). ISBN 978-0-8311-3631-4
- [11] Abbott, Richard. Analysis and Design of Composite and Metallic Flight Vehicle Structures 3rd Edition, 2019 (<https://www.abbottaerospace.com/downloads/analysis-and-design/>)
- [12] Bolt Grade Markings and Strength Chart (<https://www.boltdepot.com/fastener-information/materials-and-grades/bolt-grade-chart.aspx>)
- [13] Fastenal Torque-Tension Reference Guide (https://www.fastenal.com/content/merch_rules/images/fcom/content-library/Torque-Tension%20Reference%20Guide.pdf)
- [14] Proof Load Tensile Strength for Grade 2,5, and 8 (<https://www.almabolt.com/pages/catalog/bolts/proofloadtensile.htm>)
- [15] Bolt Clamping Force vs. Tightening Torque for Unlubricated Steel Bolts (<https://sabreindustrial.com/content/Tech%20PDFS/003%20Bolt%20Clamp%20Load%20Chart.pdf>)
- [16] Powershell (<https://docs.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7>) [2020-10-14]

- [17] .NET Objects List, Dictionary (<https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.list-1?view=netcore-3.1>, <https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.dictionary-2?view=netcore-3.1>)
- [18] Paule, David. Bolted Joints. Kitplanes Magazine (<https://www.kitplanes.com/stressing-structure-13/>). A more general Margin of Safety for Slippage (not specific to the clamp load) is provided as $MS_{slip} = ((F_{PL} - F_{TL}) * f) / F_{SL} - 1$.

Appendix

A.1 - Unbrako – joint diagrams

A.2 – frust.ps1

A.3 – bstiff.ps1

APPENDIX A.1 – UNBRAKO – JOINT DIAGRAMS



AN EXPLANATION OF JOINT DIAGRAMS

When bolted joints are subjected to external tensile loads, what forces and elastic deformation really exist? The majority of engineers in both the fastener manufacturing and user industries still are uncertain. Several papers, articles, and books, reflecting various stages of research into the problem have been published and the volume of this material is one reason for confusion. The purpose of this article is to clarify the various explanations that have been offered and to state the fundamental concepts which apply to forces and elastic deformations in concentrically loaded joints. The article concludes with general design formulae that take into account variations in tightening, preload loss during service, and the relation between preloads, external loads and bolt loads.

The Joint Diagram

Forces less than proof load cause elastic strains. Conversely, changes in elastic strains produce force variations. For bolted joints this concept is usually demonstrated by joint diagrams.

The most important deformations within a joint are elastic bolt elongation and elastic joint compression in the axial direction. If the bolted joint in Fig. 1 is subjected to the preload F_i the bolt elongates as shown by the line OB in Fig. 2A and the joint compresses as shown by the line OJ. These two lines, representing the spring characteristics of the bolt and joint, are combined into one diagram in Fig. 2B to show total elastic deformation.

If a concentric external load F_e is applied under the bolt head and nut in Fig. 1, the bolt elongates an additional amount while the compressed joint members partially relax. These changes in deformation with external loading are the key to the interaction of forces in bolted joints.

In Fig. 3A the external load F_e is added to the joint diagram F_e is located on the diagram by applying the upper end to an extension of OB and moving it in until the lower end contacts OJ. Since the total amount of elastic deformation (bolt plus joint) remains constant for a given preload, the external load changes the total bolt elongation to $\Delta l_b + \lambda$ and the total joint compression to $\Delta l_j - \lambda$.

In Fig. 3B the external load F_e is divided into an additional bolt load F_{eB} and the joint load F_{eJ} , which unloads the compressed joint members. The maximum bolt load is the sum of the load preload and the additional bolt load:

$$F_{B \max} = F_i + F_{eB}$$

If the external load F_e is an alternating load, F_{eB} is that part of F_e working as an alternating bolt load, as shown in Fig. 3B. This joint diagram also illustrates that the joint absorbs more of the external load than the bolt subjected to an alternating external load.

The importance of adequate preload is shown in Fig. 3C. Comparing Fig. 3B and Fig. 3C, it can be seen that F_{eB} will remain relatively small as long as the preload F_i is greater than F_{eJ} . Fig. 3C represents a joint with insufficient preload. Under this condition, the amount of external load that the joint can absorb is limited, and the excess

load must then be applied to the bolt. If the external load is alternating, the increased stress levels on the bolt produce a greatly shortened fatigue life.

When seating requires a certain minimum force or when transverse loads are to be transformed by friction, the minimum clamping load $F_{J \min}$ is important.

$$F_{J \min} = F_{B \max} - F_e$$

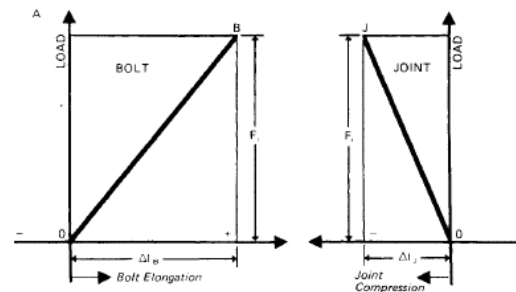


Fig. 1 (above) Joint components

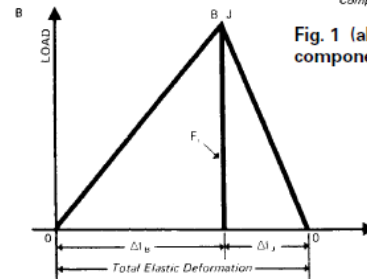


Fig. 2 Joint diagram is obtained by combining load vs. deformation diagrams of bolt and joints.

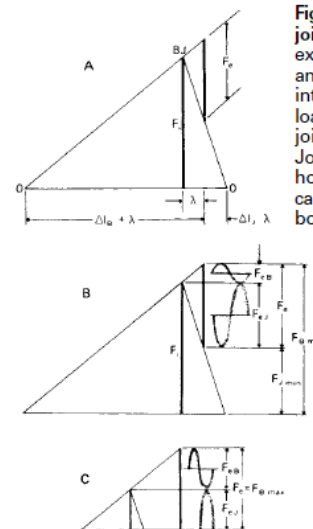


Fig. 3 The complete simple joint diagrams show external load F_e added (A), and external load divided into an additional bolt load F_{eB} and reduction in joint compression F_{eJ} (B). Joint diagram (C) shows how insufficient preload F_i causes excessive additional bolt load F_{eB} .

JOINT DIAGRAMS

Spring Constants

To construct a joint diagram, it is necessary to determine the spring rates of both bolt and joint. In general, spring rate is defined as:

$$K = \frac{F}{\Delta l}$$

From Hook's law:

$$\Delta l = \frac{F}{EA}$$

Therefore:

$$K = \frac{EA}{l}$$

To calculate the spring rate of bolts with different cross sections, the reciprocal spring rates, or compliances, of each section are added:

$$\frac{1}{K_S} = \frac{1}{K_1} + \frac{1}{K_2} + \dots + \frac{1}{K_n}$$

Thus, for the bolt shown in Fig. 4:

$$\frac{1}{K_S} = \frac{1}{E} \left(\frac{0.4d}{A_1} + \frac{l_1}{A_1} + \frac{l_2}{A_2} + \frac{l_3}{A_m} + \frac{0.4d}{A_m} \right)$$

where

d = the minor thread diameter and

A_m = the area of the minor thread diameter

This formula considers the elastic deformation of the head and the engaged thread with a length of $0.4d$ each.

Calculation of the spring rate of the compressed joint members is more difficult because it is not always obvious which parts of the joint are deformed and which are not. In general, the spring rate of a clamped part is:

$$K_J = \frac{EA_S}{l_J}$$

where A_s is the area of a substitute cylinder to be determined.

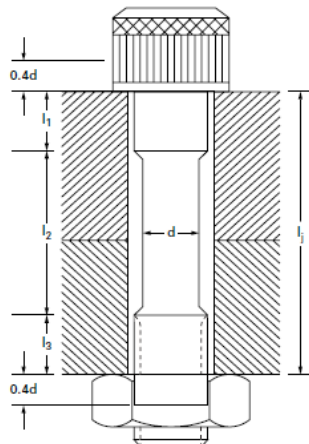


Fig. 4 Analysis of bolt lengths contributing to the bolt spring rate.

When the outside diameter of the joint is smaller than or equal to the bolt head diameter, i.e., as in a thin bushing, the normal cross sectioned area is computed:

$$A_s = \frac{\pi}{4} (D_c^2 - D_h^2)$$

where

D_c = OD of cylinder or bushing and

D_h = hole diameter

When the outside diameter of the joint is larger than head or washer diameter D_h , the stress distribution is in the shape of a barrel, Fig 5. A series of investigations proved that the areas of the following substitute cylinders are close approximations for calculating the spring contents of concentrically loaded joints.

When the joint diameter D_J is greater than D_h but less than $3D_h$:

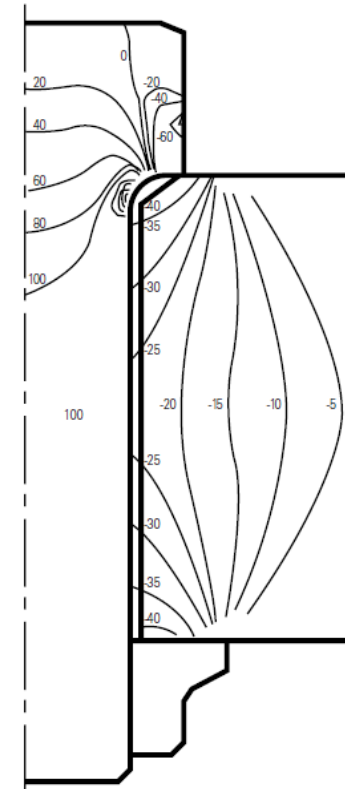


Fig. 5 Lines of equal axial stresses in a bolted joint obtained by the axisymmetric finite element method are shown for a 9/16-18 bolt preloaded to 100 KSI. Positive numbers are tensile stresses in KSI; negative numbers are compressive stresses in KSI.

Unbrako

$$A_e = \frac{\pi}{4} (D_H^2 - D_h^2) + \frac{\pi}{8} \left(\frac{D_J}{D_H} - 1 \right) \left(\frac{D_H l_J}{5} + \frac{l_J^2}{100} \right)$$

When the joint diameter D_J is equal to or greater than $3D_H$:

$$A_e = \frac{\pi}{4} [(D_H + 0.1 l_J)^2 - D_h^2]$$

These formulae have been verified in laboratories by finite element method and by experiments.

Fig. 6 shows joint diagrams for springy bolt and stiff joint and for a stiff bolt and springy joint. These diagrams demonstrate the desirability of designing with springy bolt and a stiff joint to obtain a low additional bolt load F_{eB} and thus a low alternating stress.

The Force Ratio

Due to the geometry of the joint diagram, Fig. 7,

$$F_{eB} = \frac{F_e K_B}{K_B + K_J}$$

$$\text{Defining } \Phi = \frac{K_B}{K_B + K_J}$$

$$F_{eB} = F_e \Phi \text{ and } \Phi, \text{ called the Force Ratio,} = \frac{F_{eB}}{F_e}$$

For complete derivation of Φ , see Fig. 7.

To assure adequate fatigue strength of the selected fastener the fatigue stress amplitude of the bolt resulting from an external load F_e is computed as follows:

$$\sigma_B = \pm \frac{F_{eB}/2}{A_m} \text{ or}$$

$$\sigma_B = \pm \frac{\Phi F_e}{2 A_m}$$

Effect of Loading Planes

The joint diagram in Fig 3, 6 and 7 is applicable only when the external load F_e is applied at the same loading planes as the preloaded F_i under the bolt head and the nut. However, this is a rare case, because the external load usually affects the joint somewhere between the center of the joint and the head and the nut.

When a preloaded joint is subjected to an external load F_e at loading planes 2 and 3 in Fig. 8, F_e relieves the compression load of the joint parts between planes 2 and 3. The remainder of the system, the bolt and the joint parts between planes 1-2 and 3-4, feel additional load due to F_e applied planes 2 and 3, the joint material between planes 2 and 3 is the clamped part and all other joint members, fastener and remaining joint material, are clamping parts. Because of the location of the loading planes, the joint diagram changes from black line to the blue line. Consequently, both the additional bolt load $F_{eB \max}$ decrease significantly when the loading planes of F_e shift from under the bolt head and nut toward the joint center.

Determination of the length of the clamped parts is, however, not that simple. First, it is assumed that the external load is applied at a plane perpendicular to the bolt axis. Second, the distance of the loading planes from each other has to be estimated. This distance may be expressed as the ratio of the length of clamped parts to the total joint length. Fig. 9 shows the effect of two different loading planes on the bolt load, both joints having the same preload F_i and the same external load F_e . The lengths of the clamped parts are estimated to be $0.75l_J$ for joint A, and $0.25l_J$ for joint B.

In general, the external bolt load is somewhere between $F_{eB} = 1\Phi F_e$ for loading planes under head and nut and $F_{eB} = 0\Phi F_e = 0$ when loading planes are in the joint center, as shown in Fig. 10. To consider the loading planes in calculations, the formula:

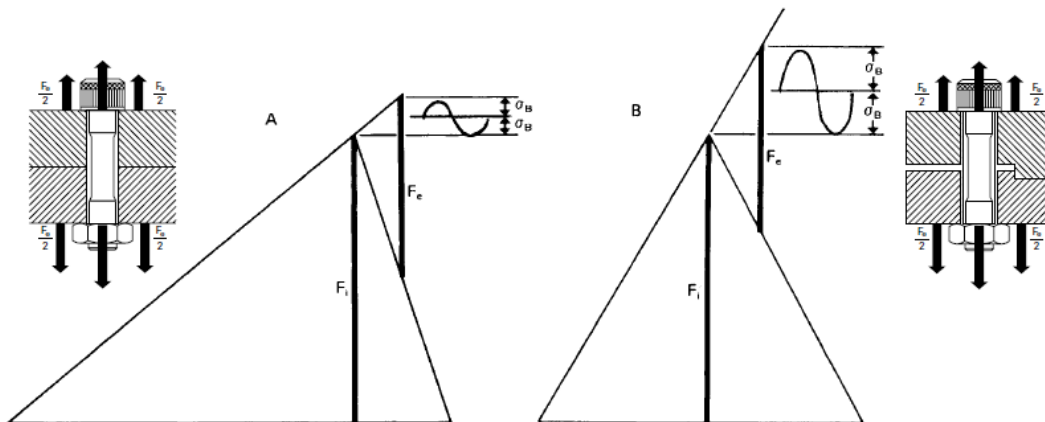


Fig. 6 Joint diagram of a springy bolt in a stiff joint (A), is compared to a diagram of a stiff bolt in a springy joint (B). Preload F_i and external load F_e are the same but diagrams show that alternating bolt stresses are significantly lower with a spring bolt in a stiff joint.

JOINT DIAGRAMS

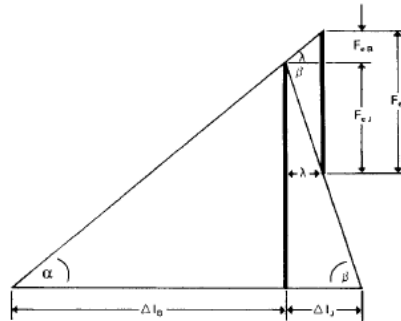


Fig. 7 Analysis of external load F_e and derivation of Force Ratio Φ .

$$\tan \alpha = \frac{F_l}{\Delta l_B} = K_B \text{ and } \tan \beta = \frac{F_l}{\Delta l_J} = K_J$$

$$\lambda = \frac{F_{eB}}{\tan \alpha} = \frac{F_{eJ}}{\tan \beta} = \frac{F_{eB}}{K_B} = \frac{F_{eJ}}{K_J} \text{ or}$$

$$F_{eJ} = \lambda \tan \beta \text{ and } F_{eB} = \lambda \tan \alpha$$

$$\text{Since } F_e = F_{eB} + F_{eJ} \\ F_e = F_{eB} + \lambda \tan \beta$$

$$\text{Substituting } \frac{F_{eB}}{\tan \alpha} \text{ for } \lambda \text{ produces:}$$

$$F_e = F_{eB} + \frac{F_{eB} \tan \beta}{\tan \alpha}$$

$$\text{Multiplying both sides by } \tan \alpha:$$

$$F_e \tan \alpha = F_{eB} (\tan \alpha + \tan \beta) \text{ and}$$

$$F_{eB} = \frac{F_e \tan \alpha}{\tan \alpha + \tan \beta}$$

$$\text{Substituting } K_B \text{ for } \tan \alpha \text{ and } K_J \text{ for } \tan \beta$$

$$F_{eB} = F_e \frac{K_B}{K_B + K_J}$$

$$\text{Defining } \Phi = \frac{K_B}{K_B + K_J}$$

$$F_{eB} = \Phi F_e$$

$$\Phi = \frac{F_{eB}}{F_e} \text{ and it becomes obvious why } \Phi \text{ is called force ratio.}$$

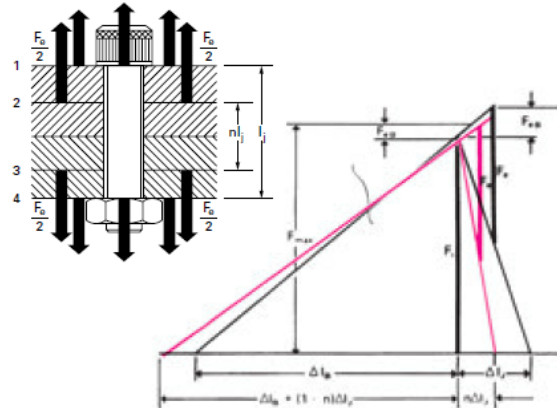


Fig. 8 Joint diagram shows effect of loading planes of F_e on bolt loads F_{eB} and $F_{B \max}$. Black diagram shows F_{eB} and $F_{B \max}$ resulting from F_e applied in planes 1 and 4. Orange diagram shows reduced bolt loads when F_e is applied in planes 2 and 3.

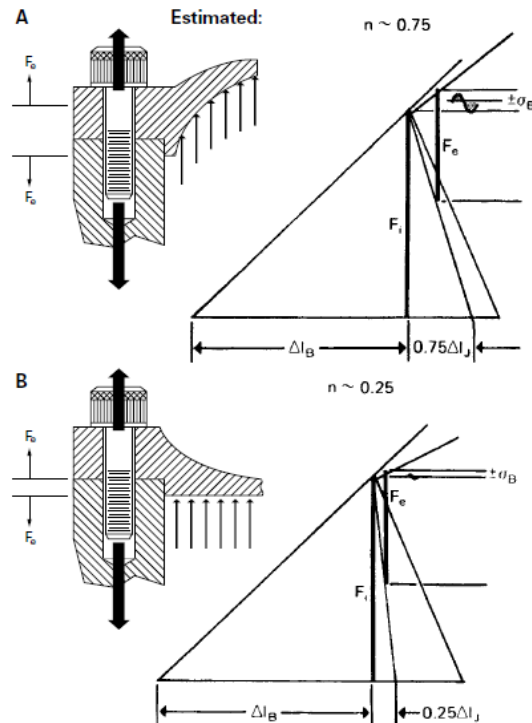


Fig. 9 When external load is applied relatively near bolt head, joint diagram shows resulting alternating stress α_B (A). When same value external load is applied relatively near joint center, lower alternating stress results (B).

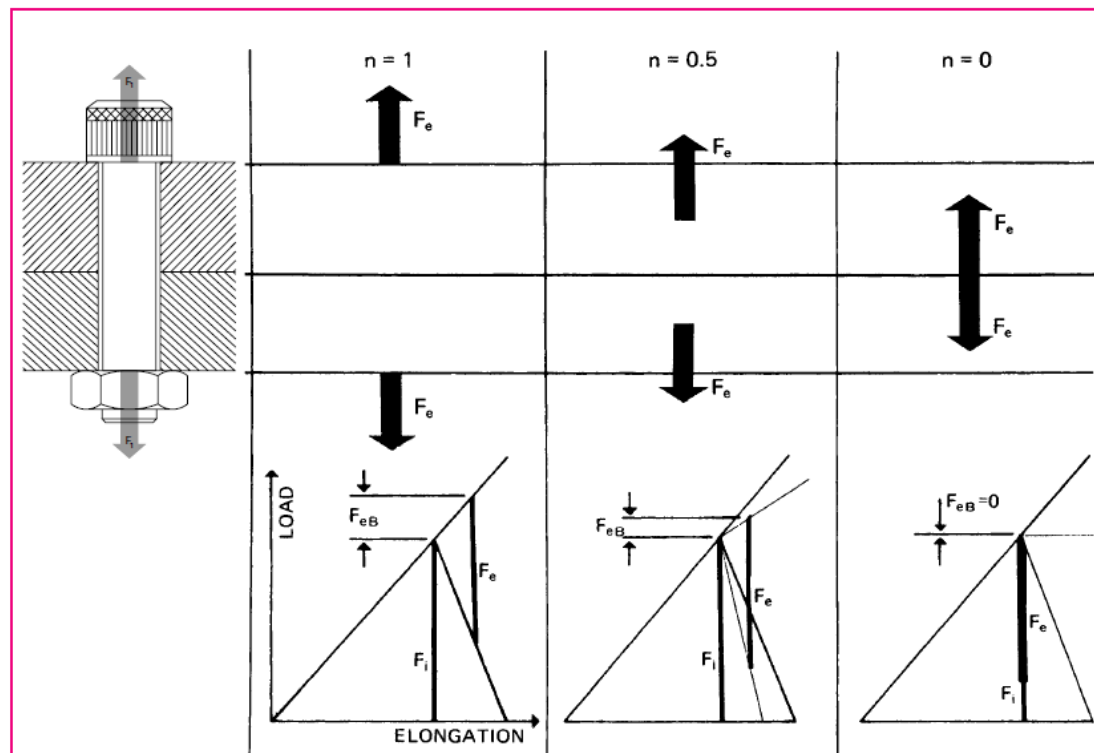


Fig. 10 Force diagrams show the effect of the loading planes of the external load on the bolt load.

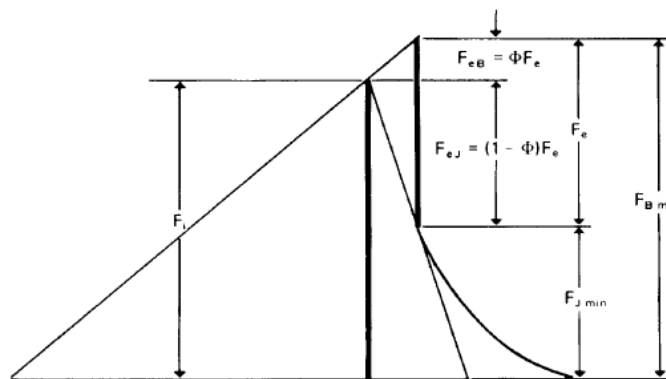


Fig. 11 Modified joint diagram shows nonlinear compression of joint at low preloads.

JOINT DIAGRAMS

$F_{eB} = \Phi F_e$ must be modified to :

$$F_{eB} = n \Phi F_e$$

where n equals the ratio of the length of the clamped parts due to F_e to the joint length l_j . The value of n can range from 1, when F_e is applied under the head and nut, to 0, when F_e is applied at the joint center. Consequently the stress amplitude:

$$\sigma_B = \pm \frac{\Phi F_e}{2 A_m} \text{ becomes}$$

$$\sigma_B = \pm \frac{n \Phi F_e}{2 A_m}$$

General Design Formulae

Hitherto, construction of the joint diagram has assumed linear resilience of both bolt and joint members. However, recent investigations have shown that this assumption is not quite true for compressed parts.

Taking these investigations into account, the joint diagram is modified to Fig. 11. The lower portion of the joint spring rate is nonlinear, and the length of the linear portion depends on the preload level F_i . The higher F_i the longer the linear portion. By choosing a sufficiently high minimum load, $F_{i \min} > 2F_{eB}$, the non-linear range of the joint spring rate is avoided and a linear relationship between F_{eB} and F_e is maintained.

Also from Fig. 11 this formula is derived:

$$F_{i \min} = F_{j \min} + (1 - \Phi) F_e + \Delta F_i$$

where ΔF_i is the amount of preload loss to be expected. For a properly designed joint, a preload loss $\Delta F_i = -(0.005 \text{ to } 0.10) F_i$ should be expected.

The fluctuation in bolt load that results from tightening is expressed by the ratio:

$$a = \frac{F_{i \max}}{F_{i \min}}$$

where a varies between 1.25 and 3.0 depending on the tightening method.

Considering the general design formulae are:

$$F_{i \text{ nom}} = F_{j \min} = (1 - \Phi) F_e$$

$$F_{i \max} = a [F_{j \min} + (1 - \Phi) F_e + \Delta F_i]$$

$$F_{B \max} = a [F_{j \min} + (1 - \Phi) F_e + \Delta F_i] + \Phi F_e$$

Conclusion

The three requirements of concentrically loaded joints that must be met for an integral bolted joint are:

1. The maximum bolt load $F_{B \max}$ must be less than the bolt yield strength.
2. If the external load is alternating, the alternating stress must be less than the bolt endurance limit to avoid fatigue failures.
3. The joint will not lose any preload due to permanent set or vibration greater than the value assumed for ΔF_i .

SYMBOLS

A	Area (in. ²)
A_m	Area of minor thread diameter (in. ²)
A_s	Area of substitute cylinder (in. ²)
A_x	Area of bolt part l_x (in. ²)
d	Diameter of minor thread (in.)
D_c	Outside diameter of bushing (cylinder) (in.)
D_H	Diameter of Bolt head (in.)
D_h	Diameter of hole (in.)
D_j	Diameter of Joint
E	Modulus of Elasticity (psi)
F	Load (lb)
F_e	External load (lb.)
F_{eB}	Additional Bolt Load due to external load (lb)
F_{eJ}	Reduced Joint load due to external load (lb)
F_i	Preload on Bolt and Joint (lb)
ΔF_i	Preload loss (-lb)
$F_{i \min}$	Minimum preload (lb)
$F_{i \max}$	Maximum preload (lb)
$F_{j \text{ nom}}$	Nominal preload (lb)

$F_{B \max}$	Maximum Bolt load (lb)
$F_{j \min}$	Minimum Joint load (lb)
K	Spring rate (lb/in.)
K_B	Spring rate of Bolt (lb/in.)
K_J	Spring rate of Joint (lb/in.)
K_x	Spring rate of Bolt part l_x (lb/in.)
l	Length (in.)
Δl	Change in length (in.)
l_B	Length of Bolt (in.)
Δl_B	Bolt elongation due to F_i (in.)
l_J	Length of Joint (in.)
Δl_J	Joint compression to F_i (in.)
l_x	Length of Bolt part x (in.)
n	$\frac{\text{Length of clamped parts}}{\text{Total Joint Length}}$
α	Tightening factor
Φ	Force ratio
λ	Bolt and Joint elongation due to F_e (in.)
σ_B	Bolt stress amplitude ($\pm$ psi)

APPENDIX A.2 – FRUST.PS1

Powershell Environment (\$PSVersionTable)

```

Name                                     Value
----                                     -
PSVersion                               5.1.18362.1714
PSEdition                               Desktop
PSCompatibleVersions                    {1.0, 2.0, 3.0, 4.0...}
BuildVersion                            10.0.18362.1714
CLRVersion                              4.0.30319.42000
WSManStackVersion                       3.0
PSRemotingProtocolVersion               2.3
SerializationVersion                   1.1.0.1

===BOF
<#

frust.ps1 - calculate frustum thickness, diameter, and stiffness parameters for joint stiffness calculation
           Shigley, Mechanical Engineering Design, 7th Ed., pg. 414

INPUTS:
$case      case name
           Dictionary[string, List[double]]
$data      define case data for
           thickness [in]
           Modulus of Elasticity [ksi]
           effective grip [in]
           basic diameter [in]
           washer diameter [in]

           example:
           $data["shk_mnt_t"] = (.75, .93330)
           $data["shk_mnt_E"] = (29700.0, 10300.0)
           $data["shk_mnt_lg"] = (0.75 + 5/16)
           $data["shk_mnt_d"] = (5/8)
           $data["shk_mnt_dw"] = (1.5*5/8)

RETURNS:
Frustum thickness layers with diameter and stiffness (including at peak and effective grip) and total stiffness

===BOF (example)
===0:shk_mnt===
           LG      1.06250
           dw      0.93750
           D       1.55093

           SUM      E      THK      D      K
0.00000 <----->
           PK      29700.0      0.53125      1.55093      44.593
           29700.0      0.21875      172.523
0.75000 <----->      29700.0      0.75000      1.29834
1.06250 LG      10300.0      0.31250      0.93750      20.855
1.68330 <----->      10300.0      0.93330
           =====
           13.128

...
===EOF

FUNCTIONS:
kstiff      calculate layer stiffness [Mlbf/in]
diamf       calculate layer frustum diameter [in]

VARIABLES:
$vars      data variable names
$fmtl      title format string
$fmtA      entry format string
$alpha     half-apex angle for frustum [deg]
$tanalpha  tangent of $alpha
$lcl       case length
$lcl       case length - 1
$case_no   case number
$ca        case name
$lg        effective grip [in]
$da        basic diameter [in]
$dwa       "washer" diameter [in]
$lg2       effective grip / 2 (frustum peak) [in]
$sum       accumulated/total thickness [in]
$sumk      sum of stiffness (in series, 1/k1 + 1/k2 + ...)
$flg_pk    flag if $sum < $lg2
$flg_lg    flag if $lg reached
$dmx       frustum diameter = $dwa + $lg*Tan($alpha*PI/180) [in]
$ttl       thickness length
$lt        thickness length - 1
$indx      index of thickness loop
$ta        thickness of current layer [in]
$Ea        modulus of elasticity for current layer [ksi]
$diamprev  previous layer frustum diameter [in]
$tdiam     layer frustum diameter [in]
# $dcalc    frustum diameter for current break point (lg2, layer)
$K         layer stiffness [Mlbf/in]
$del       distance to next layer from peak or to $lg [in]
$te        accumulated/total thickness at $lg [in]

#>

<#
FUNCTIONS=====
#>

```

```

function kstiff ($e, $t, $d, $dw, $a) {
<#
    kstiff                calculate frustum layer stiffness
                           Shigley, Mechanical Engineering Design, 7th Ed., pg. 414

    INPUTS:
    $e                    modulus of elasticity [ksi]
    $t                    thickness [in]
    $d                    basic diameter [in]
    $dw                   "washer" diameter [in]
    $a                    half-apex angle [deg]

    RETURNS:
    $k                    layer stiffness [Mlbf/in]

    VARIABLES:
    $tana                 tangent of $a
    $t1 - $t3             temporary variables
#>

    $tana = [Math]::Tan($a * [Math]::PI/180)
    $t1 = [Math]::PI * $e * $d * $tana
    $t2 = (2 * $t * $tana + $dw - $d) * ($dw + $d)
    $t3 = (2 * $t * $tana + $dw + $d) * ($dw - $d)
    $k = ($t1/[Math]::Log($t2/$t3))/1000

    return $k
}

function diamf ($dw, $t, $a) {
<#
    diamf                calculate frustum diameter
                           Shigley, Mechanical Engineering Design, 7th Ed., pg. 414

    INPUTS:
    $dw                   "washer" diameter [in]
    $t                    thickness [in]
    $a                    half-apex angle [deg]

    RETURNS:
    $tdiam               layer diameter [in]

    VARIABLES:
    $tana                 tangent of $a
#>

    $tana = [Math]::Tan($a * [Math]::PI/180)
    $tdiam = $dw + 2 * $t * $tana

    return $tdiam
}

<#
=====
#>

# define parameters
$data = New-Object "System.Collections.Generic.Dictionary[ string], `
[System.Collections.Generic.List[double]] )"
#$data[$ca+"_"+$_] = New-Object System.Collections.Generic.List[double]

$vars = @("t", "E", "lg", "d", "dw")

<#
DATA=BEG=====
#>

<# case data for
    name
    thickness [in]
    Modulus of Elasticity [ksi]
    effective grip [in]
    basic diameter [in]
    washer diameter [in]
#>

$case = @("shk_mnt", "shk_guard", "dbl_roll", "dbl_roll_R01", "ra_hse", "pg427", "pg432", "5layer")

$data["shk_mnt_t"] = (.75, .93330)
$data["shk_mnt_E"] = (29700.0,10300.0)
$data["shk_mnt_lg"] = (0.75 + 5/16)
$data["shk_mnt_d"] = (5/8)
$data["shk_mnt_dw"] = (1.5*5/8)

$data["shk_guard_t"] = (1.25, .75, .93330)
$data["shk_guard_E"] = (29700.0,29700.0,10300.0)
$data["shk_guard_lg"] = (1.25 + 0.75 + 5/16)
$data["shk_guard_d"] = (5/8)
$data["shk_guard_dw"] = (1.5*5/8)

$data["dbl_roll_t"] = (0.65, .93330)
$data["dbl_roll_E"] = (27000.0,10300.0)
$data["dbl_roll_lg"] = (0.65 + 5/16)
$data["dbl_roll_d"] = (5/8)
$data["dbl_roll_dw"] = (1.5*5/8)

$data["dbl_roll_R01_t"] = (0.25, .93330)
$data["dbl_roll_R01_E"] = (27000.0,10300.0)
$data["dbl_roll_R01_lg"] = (0.25 + 5/16)
$data["dbl_roll_R01_d"] = (5/8)
$data["dbl_roll_R01_dw"] = (1.5*5/8)

```

```

$da["ra_hse_t"] = (0.65, 1.1830)
$da["ra_hse_E"] = (29700.0,10300.0)
$da["ra_hse_lg"] = (0.65 + 5/16)
$da["ra_hse_d"] = (5/8)
$da["ra_hse_dw"] = (1.5*5/8)

# from Shigley, 7th Ed.
$da["pg427_t"] = (0.75, 0.75)
$da["pg427_E"] = (14000.0,14000.0)
$da["pg427_lg"] = 1.5
$da["pg427_d"] = (5/8)
$da["pg427_dw"] = (1.5*5/8)

# from Shigley, 7th Ed.
$da["pg432_t"] = ((1/16+ 5/8), (5/8))
$da["pg432_E"] = (30000.0,16000.0)
$da["pg432_lg"] = 1.0
$da["pg432_d"] = (5/8)
$da["pg432_dw"] = (1.5*5/8)

# 5 layers test
$da["5layer_t"] = (1, 1, 1, 1, 1)
#$da["5layer_t"] = (.25, 1.75, 1, 1, 1)
#$da["5layer_t"] = (.1, .1, .1, .1, 4.6)
$da["5layer_E"] = (10, 20, 30, 40, 50)
$da["5layer_lg"] = 1.1
#$da["5layer_lg"] = 4.9
#$da["5layer_lg"] =5
$da["5layer_d"] = (5/8)
$da["5layer_dw"] = (1.5*5/8)

<#
DATA=END=====
#>

# format strings
$fmt1 = "{0,10} {1,10:F5}"
$fmtA = "{0,10:F5} {1,10} {2,10:F1} {3,10:F5} {4,10:F5} {5,10:F5} {6,10:F3}"

# half-apex angle [deg]
$alpha = 30
$stalpha = [Math]::Tan($alpha*[Math]::PI/180)

<#
MAIN=====
#>

clear

# case length
$lcl = $case.length
$lcl = $lcl - 1
("n{0} Cases" -f ($lcl+1) )

# loop cases
(0..$lcl) | foreach {

    $case_no = $_
    $ca = $case[$case_no]

    # get values
    $lga = $da[$ca+"_lg"][0]
    $da = $da[$ca+"_d"][0]
    $dwa = $da[$ca+"_dw"][0]

    # set for new case
    $lga2 = $lga/2.0
    $sum = 0
    $sumk = 0
    $flg_pk = $true    # < $lga2
    $flg_lg = $true    # < $lga

    # output parameters
    ("n===={0}:{1}====" -f $case_no,$ca )
    ($fmt1 -f "LG", $lga)
    ($fmt1 -f "dw", $dwa)
    $Dmax = $dwa + $lga*$stalpha
    ($fmt1 -f "D", $Dmax)
    ($fmt1 -f "", "")

    # output heading
    ($fmtA -f "SUM", "", "E", "THK", "THK", "D", "K")

    # thickness length
    $ltl = $da[$ca+"_t"].count
    $lt = $ltl - 1

    # loop thickness
    (0..$lt) | foreach {

        $indx = $_

        # get values
        $ta = $da[$ca+"_t"][$indx]
        $Ea = $da[$ca+"_E"][$indx]

        # start diameter
        if ($indx -eq 0) {

            ($fmtA -f 0, "<----->", "", "", "", $dwa, "") # zero
            $tdiamprev = $dwa
            $tdiam = diamf $dwa $ta $alpha

        } else {

            $tdiamprev = $tdiam

        }

        # sum thickness
        $sum = $sum + $ta

        # not at frustum peak
        if ($sum -lt $lga2) {

```

```

# diameter at thickness
$tdiam = diamf $tdiamprev $ta $alpha

# layer stiffness
$k = kstiff $Ea $ta $da $tdiamprev $alpha

($fmtA -f $sum, "<----->", $Ea, $ta, "", $tdiam, $k)
$sumk = $sumk + 1/$k

} else {

# get delta and calculate diameter
$del = $sum - $lga2 # used directly for thickness after peak
$tdiam = diamf $dwa ($lga2-$del) $alpha

# do once to show peak
if ($flg_pk) {

    if ($indx -eq 0) {

        # layer stiffness
        $k = kstiff $Ea $lga2 $da $dwa $alpha

        ($fmtA -f "", "PK", $Ea, "", $lga2, $Dmax, $k)
        $sumk = $sumk + 1/$k

    } else {

        # layer stiffness
        $k = kstiff $Ea ($lga2-($sum-$ta)) $da $tdiamprev $alpha

        ($fmtA -f "", "PK", $Ea, "", ($lga2-($sum-$ta)), $Dmax, $k)
        $sumk = $sumk + 1/$k

    }

    if ($sum -ge $lga) {

        # reached effective grip (at end)

    } else {

        # stiffness from peak to next layer
        $k = kstiff $Ea $del $da $tdiam $alpha

        ($fmtA -f "", "", $Ea, "", $del, "", $k)
        $sumk = $sumk + 1/$k

    }

}

if ($sum -lt $lga) {

    # not at effective grip yet
    if (-not $flg_pk) {

        # layer stiffness
        $k = kstiff $Ea $ta $da $tdiam $alpha

        ($fmtA -f $sum, "<----->", $Ea, $ta, "", $tdiam, $k)
        $sumk = $sumk + 1/$k

    } else {

        ($fmtA -f $sum, "<----->", $Ea, $ta, "", $tdiam, "")

    }

} else {

    # reached effective grip (at end)
    $dele = $ta - ($sum - $lga) # thickness to lg
    $te = $sum - $ta + $dele # total thickness @ lg

    if ($flg_lg) {

        if ($flg_pk) {

            # layer stiffness
            $k = kstiff $Ea ($lga-$lga2) $da $dwa $alpha

            ($fmtA -f $te, "LG", $Ea, "", ($lga-$lga2), $dwa, $k)
            $sumk = $sumk + 1/$k

        } else {

            # layer stiffness
            $k = kstiff $Ea $dele $da $dwa $alpha

            ($fmtA -f $te, "LG", $Ea, "", $dele, $dwa, $k)
            $sumk = $sumk + 1/$k

        }

        # reached effective grip
        $flg_lg = $false

    }

    # layer data
    ($fmtA -f $sum, "<----->", $Ea, $ta, "", "", "")

}

# past frustum peak
$flg_pk = $false

}

} # thickness

```

```

# total stiffness (in series, 1/k = 1/k1 + 1/k2 + ...)
($fmtA -f "%", "%", "%", "%", "%", "=====", (1/$sumk))

) # case

("{0}" -f "%n====>Done.")

<#
      1      2      3      4      5      6      7      8      9      1      1      1      1
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
#>

===EOF

```

APPENDIX A.3 – BSTIFF.PS1

Powershell Environment

```

Name                                     Value
----                                     -
PSTVersion                             5.1.18362.1714
PSEdition                              Desktop
PSCompatibleVersions                   {1.0, 2.0, 3.0, 4.0...}
BuildVersion                           10.0.18362.1714
CLRVersion                             4.0.30319.42000
WSManStackVersion                      3.0
PSRemotingProtocolVersion              2.3
SerializationVersion                   1.1.0.1

===BOF
<#

bstiff.ps1 - calculate bolt/screw stiffness for threaded/unthreaded portions
            Shigley, Mechanical Engineering Design, 7th Ed., pg. 410
            ASME B1.1-2019 Unified Inch Screw Threads (Un, UNR, and UNJ Thread Forms)
            > Table 2A - Limits of Size for Standard Series External Thread (UN, UNR, and UNJ)
            > Table 2B - Limits of Size for Standard Series Internal Threads (UN and UNJ)
            > Appendix B - Thread Strength Design Formulas, pg. 150

INPUTS:
$case          case name
$fastdesc      fastener description
$data          Dictionary[string, List[double]]
              define case data for
                0: basic major diameter [in]
                1: threads per inch (n=1/pitch) [./in]
                2: elasticity of fastener [ksi]
                3: fastener length [in]
                4: fastener thread length [in]
                5: effective grip [in]

              example:
              $data["shk_mnt"] = (           (5/8),    11,  29000, 1.75, 1.5, 1.0625)

RETURNS:
fastener unthreaded, threaded, and combined stiffness

===BOF (example)
===0:shk_mnt:5/8-11 UNC===
D      0.62500
n      11.00000
E      29000.00000
L      1.75000
Lt     0.81250
Lg     1.06250

      BOLT  UNTHREAD  THREADED
6.57601  35.58835   8.06654
...
===EOF

VARIABLES:
$vars          data variable names
$fmt1          title format string
$fmtA          entry format string
$lcl           case length
$l             case length - 1
$case_no       case number
$ca            case name
$fastdesc      fastener description
$D             basic major diameter [in]
$n             threads per inch (n=1/pitch) [./in]
$E             elasticity of fastener [ksi]
$L             fastener length [in]
$Lt            fastener thread length [in]
$Lg            effective grip [in]

# unthreaded portion
$l_d           length of unthreaded portion in grip
$A_d           major diameter of area of fastener
$K_d           unthreaded stiffness [Mlbf/in]

# threaded portion
$L_t           length of threaded portion of grip
$A_t           tensile stress area (As)
$K_t           threaded stiffness

# fastener stiffness
$B_s           combined stiffness (in series, 1/k = 1/k1 + 1/k2 + ...)

#>

<#
=====
#>

# define parameters
$data = New-Object "System.Collections.Generic.Dictionary[ [string], `
[System.Collections.Generic.List[double]] ]"
#$data[$ca+"_"+$_] = New-Object System.Collections.Generic.List[double]

$vars = @("D", "n", "E", "L", "Lt", "Lg")

```


===EOF

TN 22-0427-01

PART SELECTION BASED ON YIELD STRENGTH

April 27, 2022

W. Bylsma

U.S. Army Combat Capabilities Development Command (DEVCOM)
Ground Vehicle Systems Center (GVSC)
Detroit Arsenal
Modeling, Simulation & Software
ATTN: FCDD-GVR-MSS/MS 157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

When many parts are involved in a finite element analysis (FEA) how do you reasonably reduce the number of parts to analyze? See Figure 1 for an example.

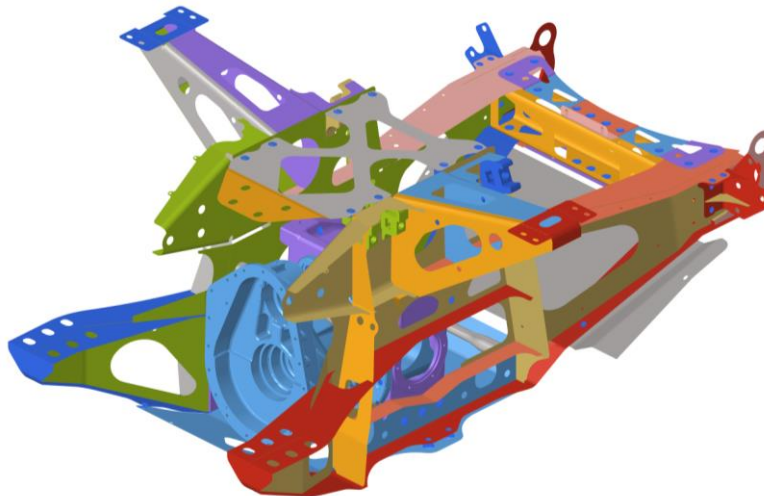


Figure 1 – Complex Frame Model

PROBLEM SOLUTION:

This solution is based on using Altair's HyperView [1] post processing with Dassault's Abaqus/Standard [2] FEA solver, but could be adapted to other software. Powershell [3] is used to create a script ("yield_sel.ps1") that scans element stress values compared to the material yield strength. Any elements with absolute stress values (to account for tension and compression) greater than the yield strength are summed for each part. The stress summary output for each part (if any of its elements exceed the yield strength) are the minimum and maximum stress, count of elements exceeding yield, and the material yield strength. It is assumed that all parts have distinct names and do not begin with "test". Also output in the summary is the overall minimum and maximum stress and related element entries with any material names encountered but not found in the script (see the switch statement).

In HyperView, begin with a Contour plot of the desired elements (e.g. Displayed, Window, etc.) and stress component (Von Mises, Maximum Principal, or Minimum Principal, etc.) with Averaging Method set to

“None” (otherwise HyperView will not show the proper query fields). Make sure the desired STEP frame is selected. From Query (Elements) select the following query fields: Element ID, Component Name, Contour(“...”-Stress components), and Material Name. Apply the selection and Export to a .CSV file. Repeat for different stress components. Following is an example of the HyperView output.

Example HyperView output (for defined stress component Von Mises or Maximum Principal or etc.)

```
Element ID, Component Name, Contour(S-Global-Stress components), Material Name,
16783675, HM_MPC_BEAM, N/A, N/A,
15747611, S28_diff_housing_Al_GrA206_T4_sol, 4.427E+00, Aluminum_GrA206_T4,
15747348, S28_diff_housing_Al_GrA206_T4_sol, 4.585E+00, Aluminum_GrA206_T4,
15747347, S28_diff_housing_Al_GrA206_T4_sol, 4.487E+00, Aluminum_GrA206_T4,
15746797, S28_diff_housing_Al_GrA206_T4_sol, 4.595E+00, Aluminum_GrA206_T4,
15736507, S28_diff_housing_Al_GrA206_T4_sol, 3.709E+00, Aluminum_GrA206_T4,
...
```

From all the output files determine each material name and yield strength. Edit the “yield_sel.ps1” script switch statement to include all material names and yield. Define a “go.ps1” script to process all of the output stress components from the output files (see Appendix A.1 for required parameters).

Example go.ps1

```
#.\yield_sel [input file] [# skip lines] | out-file -encoding ascii [output file]
.\yield_sel .\data_f6\f6_latRH_von_mx.csv 2 | out-file -encoding ascii out_f6_latRH_von_mx # von mises
.\yield_sel .\data_f6\f6_latRH_max_mx.csv 2 | out-file -encoding ascii out_f6_max_mx # max principal
.\yield_sel .\data_f6\f6_latRH_min_mx.csv 2 | out-file -encoding ascii out_f6_min_mx # min principal
```

The output after processing will look similar to the following example.

Example yield_sel.ps1 stress summary output (shows part name, min/max stress, element count, yield)

```
MS-13_brkt_rear_frm_bot_rh_Steel_GrS960QL-6mm      1002.0    1145.0      7    960.0
MS-bot_plate_xm_Steel_514GrB-6_452mm_end          692.2     808.7      9    690.0
MS-13_brkt_rear_frame_bot_lh_Steel_GrS960QL-6mm    964.4    1595.0     19    960.0
MS-top_plate_xm_Steel_514GrB_6_452mm_end          695.8     695.8      1    690.0
MS-17_brkt_lower_ctrl_mntg_rh_Steel_GrS960QL-9mm   1012.0    1212.0      5    960.0
MS-23_skidplate_weldment_Al_B209Gr5052H32-9_525mm  160.4     395.7     154    160.0
MS-17_brkt_lower_ctrl_mntg_lh_Steel_GrS960QL-9mm   961.7    1515.0      9    960.0
MS-skid_plate_Al_Gr5052_H32_4_775mm               160.1     684.2     219    160.0
1595_15353906,MS-13_brkt_rear_frame_bot_lh_Steel_GrS960QL-6mm, 1.595E+03,Steel_EN10025-6_GrS960QL,
-43.55_16507349,S11_block_spring_rh_Steel_A514_GrB,-4.355E+01,Steel_ASTM514_GrB,
===Not Found Property===
N/A
```

Using a text editor or spreadsheet (e.g. Microsoft Excel) review or assembly the stress summary outputs to make a determination on what parts to analyze in detail.

A brief outline of the script process follows.

- Initialize variables
- Skip header lines
- Read file while -not EndOfStream
 - Display count every 5000 entries
 - Split line, parse stress level
 - If material found, set yield, set flag/found, else Not Found
 - Update overall min/max stress and related entries
 - If flag/found, initialize if first, else update count, part min/max stress
- Close file
- Output part stress summary

- Output overall min/max stress
- Output not found materials

A review of parts with low counts may be necessary to determine if they should be included for more detailed analysis. Table 1 shows an example of the final parts selected.

Table 1 – Part Selection Table

LOAD CASE	FAILURE THEORY	PART NAME	MIN [MPa]	MAX [MPa]	COUNT	YIELD [MPa]	REFERENCE
LONG	Von Mises	None					
	Max Principal	None					
	Min Principal	None					
VERT	Von Mises	MS-01_brkt_rearfrm_mntg_lh_Steel_Gr5700MC-6mm	706.1	718.9	3	700	Fixed, Boundary Condition
	"	MS-01_brkt_rearfrm_mntg_rh_Steel_Gr5700MC-6mm	743.1	767.4	3	700	Fixed, Boundary Condition
	"	MS-17_brkt_lower_ctrl_mntg_lh_Steel_Gr5960QL-9mm	989.4	1252	4	960	
	"	S11_block_spring_lh_Steel_A514_GrB	702.6	911.6	3	690	Rigid Tie Connection
	"	S11_block_spring_rh_Steel_A514_GrB	698.8	1153	9	690	Rigid Tie Connection
	Max Principal	MS-17_brkt_lower_ctrl_mntg_lh_Steel_Gr5960QL-9mm	1030	1258	4	960	
	"	S11_block_spring_lh_Steel_A514_GrB	761.4	761.4	1	690	
	Min Principal	S11_block_spring_lh_Steel_A514_GrB	-841.3	-841.3	1	690	
	"	S11_block_spring_rh_Steel_A514_GrB	-1176	-692.6	5	690	
	"						
LAT_LH	Von Mises	MS-13_brkt_rear_frame_bot_lh_Steel_Gr5960QL-6mm	982.9	1508	15	960	
	"	MS-13_brkt_rear_frm_bot_rh_Steel_Gr5960QL-6mm	964.4	1241	11	960	
	"	MS-17_brkt_lower_ctrl_mntg_lh_Steel_Gr5960QL-9mm	994.7	1336	6	960	
	"	MS-17_brkt_lower_ctrl_mntg_rh_Steel_Gr5960QL-9mm	969.2	1111	4	960	
	"	MS-23_skidplate_weldment_Al_B209Gr5052H32-9_525mm	160.5	349.1	186	160	
	"	MS-bot_plate_xm_Steel_514GrB-6_452mm_end	698.1	784.4	7	690	
	"	MS-skid_plate_Al_Gr5052_H32_4_775mm	160.1	850.9	660	160	
	Max Principal	MS-13_brkt_rear_frame_bot_lh_Steel_Gr5960QL-6mm	964.4	1595	19	960	
	"	MS-13_brkt_rear_frm_bot_rh_Steel_Gr5960QL-6mm	1002	1145	7	960	
	"	MS-17_brkt_lower_ctrl_mntg_lh_Steel_Gr5960QL-9mm	961.7	1515	9	960	
	"	MS-17_brkt_lower_ctrl_mntg_rh_Steel_Gr5960QL-9mm	1012	1212	5	960	
	"	MS-23_skidplate_weldment_Al_B209Gr5052H32-9_525mm	160.4	395.7	154	160	
	"	MS-bot_plate_xm_Steel_514GrB-6_452mm_end	692.2	808.7	9	690	
	"	MS-skid_plate_Al_Gr5052_H32_4_775mm	160.1	684.2	219	160	
	"	MS-top_plate_xm_Steel_514GrB-6_452mm_end	695.8	695.8	1	690	
	Min Principal	MS-skid_plate_Al_Gr5052_H32_4_775mm	-183.4	-170.5	2	160	
LAT_RH	Von Mises	MS-13_brkt_rear_frame_bot_lh_Steel_Gr5960QL-6mm	971.1	1247	17	960	
	"	MS-13_brkt_rear_frm_bot_rh_Steel_Gr5960QL-6mm	999.1	1413	11	960	
	"	MS-17_brkt_lower_ctrl_mntg_lh_Steel_Gr5960QL-9mm	972.1	1357	7	960	
	"	MS-17_brkt_lower_ctrl_mntg_rh_Steel_Gr5960QL-9mm	970.2	1091	3	960	
	"	MS-23_skidplate_weldment_Al_B209Gr5052H32-9_525mm	160.2	334	180	160	
	"	MS-bot_plate_xm_Steel_514GrB-6_452mm_end	691.7	825.4	10	690	
	"	MS-skid_plate_Al_Gr5052_H32_4_775mm	160.1	808.1	634	160	
	Max Principal	MS-13_brkt_rear_frame_bot_lh_Steel_Gr5960QL-6mm	981.6	1351	12	960	
	"	MS-13_brkt_rear_frm_bot_rh_Steel_Gr5960QL-6mm	964.9	1551	13	960	
	"	MS-17_brkt_lower_ctrl_mntg_lh_Steel_Gr5960QL-9mm	1012	1493	10	960	
	"	MS-17_brkt_lower_ctrl_mntg_rh_Steel_Gr5960QL-9mm	1030	1227	5	960	
	"	MS-23_skidplate_weldment_Al_B209Gr5052H32-9_525mm	160.1	379.5	108	160	
	"	MS-bot_plate_xm_Steel_514GrB-6_452mm_end	693.1	852.1	11	690	
	"	MS-skid_plate_Al_Gr5052_H32_4_775mm	160.3	746.5	221	160	
	"	MS-top_plate_xm_Steel_514GrB-6_452mm_end	690.5	690.5	1	690	
	Min Principal	MS-skid_plate_Al_Gr5052_H32_4_775mm	-210.2	-167.7	4	160	
NOTE:			Not Selected				

REFERENCE

- [1] Altair HyperView (<https://www.altair.com/hyperview/>)

Below references require account at <https://help.3ds.com>

- [2] Dassault Systems SIMULIA Abaqus/Standard (<https://www.3ds.com/products-services/simulia/products/abaqus/abaqusstandard/>)

Above references require account at <https://help.3ds.com>

- [3] Powershell (<https://docs.microsoft.com/en-us/powershell/>)

APPENDIX

A.1 YIELD_SEL POWERSHELL SCRIPT

APPENDIX A.1- YIELD_SEL SCRIPT

```

===BOF (yield_sel.ps1)
param ([string]$fn, [int]$nskip=0)

<#

Yield_sel.ps1 - find min/max/count of values greater than yield for specific materials and parts
processes output from Altair HyperView (Contour/Query,Averaging Method=None,Export)

.\yield_sel [$fn] [$nskip] | out-file -encoding ascii [output file name]
.\yield_sel .\data_f6\f6_latRH_von_mx.csv 2 | out-file -encoding ascii out_f6_latRH_von_mx
.\yield_sel .\data_f6\f6_latRH_max_mx.csv 2 | out-file -encoding ascii out_f6_latRH_max_mx
.\yield_sel .\data_f6\f6_latRH_min_mx.csv 2 | out-file -encoding ascii out_f6_latRH_min_mx

INPUTS:
$fn                input file name
$nskip             header lines to skip (default = 0, normally 2)

NOTE: define material names and yield (using consistent units) in the switch statement
switch ($l[3]) { "Aluminum_GrA206_T4" { $yld = 205; ...} ...
suggest creating a test file with all the material names with (yield+1)

EXAMPLE INPUT

===BOF
Element ID, Component Name, Contour(S-Global-Stress components), Material Name,
15747611,S28_diff_housing_Al_GrA206_T4_sol, 4.427E+00,Aluminum_GrA206_T4,
15747348,S28_diff_housing_Al_GrA206_T4_sol, 4.585E+00,Aluminum_GrA206_T4,
15747347,S28_diff_housing_Al_GrA206_T4_sol, 4.487E+00,Aluminum_GrA206_T4,
15746797,S28_diff_housing_Al_GrA206_T4_sol, 4.595E+00,Aluminum_GrA206_T4,
15736507,S28_diff_housing_Al_GrA206_T4_sol, 3.709E+00,Aluminum_GrA206_T4,
...
===EOF

===BOF-mattest
#,#,10,#,
-1,test1, 206,Aluminum_GrA206_T4
-1,test2, 161,Aluminum_ASTM_B209_Gr5052_H32
-1,test3, 701,Steel_EN10149-2_GrS700MC
-1,test4, 473,Steel_4130H
-1,test5, 961,Steel_EN10025-6_GrS960QL
-1,test6, 691,Steel_ASTM514_GrH
-1,test7, 249,Steel_ASTM_A1011_CS_TypeB
-1,test8, 691,Steel_ASTM514_GrB
-1,test9, 270,Aluminum_Gr6061_T651
-1,test10, 553,Steel_ASTM656_Gr80
===EOF

===BOF-mattest_neg
#,#,10,#,
-1,test1, -206,Aluminum_GrA206_T4
-1,test2, -161,Aluminum_ASTM_B209_Gr5052_H32
-1,test3, -701,Steel_EN10149-2_GrS700MC
-1,test4, -473,Steel_4130H
-1,test5, -961,Steel_EN10025-6_GrS960QL
-1,test6, -691,Steel_ASTM514_GrH
-1,test7, -249,Steel_ASTM_A1011_CS_TypeB
-1,test8, -691,Steel_ASTM514_GrB
-1,test9, -270,Aluminum_Gr6061_T651
-1,test10, -553,Steel_ASTM656_Gr80
===EOF

EXAMPLE OUTPUT

===BOF
MS-13 brkt_rear_frm_bot_rh Steel GrS960QL-6mm      1002.0  1145.0      7    960.0
MS-bot_plate_xm_Steel_514GrB-6_452mm_end          692.2   808.7      9    690.0
MS-13 brkt_rear_frame_bot_lh Steel GrS960QL-6mm    964.4  1595.0     19    960.0
MS-top_plate_xm_Steel_514GrB_6_452mm_end          695.8   695.8      1    690.0
MS-17 brkt_lower_ctrl_mntg_rh Steel GrS960QL-9mm   1012.0  1212.0      5    960.0
MS-23 skidplate_weldment_Al_B209Gr5052H32-9_525mm 160.4   395.7     154    160.0
MS-17 brkt_lower_ctrl_mntg_lh Steel GrS960QL-9mm   961.7  1515.0      9    960.0
MS-skid_plate_Al_Gr5052_H32_4_775mm              160.1   684.2     219    160.0
1595 15353906,MS-13 brkt_rear_frame_bot_lh Steel GrS960QL-6mm, 1.595E+03,Steel_EN10025-6_GrS960QL,
-43.55 16507349,S11_block_spring_rh_Steel_A514_GrB,-4.355E+01,Steel_ASTM514_GrB,
===Not Found Property===
N/A
===EOF

$fn                file name
VARIABLES:
$sh                hash for part entries greater than ABS(yield) = @(min, max, count, yield)
$hnf              hash for material name not found

$maxall            max stress
$maxall_tag        max stress part entry
$minall            min stress
$minall_tag        min stress part entry

$cnt               total count of part entries
$cntd              display total count after 5000 entries

$stmp_in           absolute path to input file name
$sr_rmd            IO.StreamReader object
$rmdata            part entry line

$l                 array of split line
$l[0]              = element number
$l[1]              = part name
$l[2]              = yield strength
$l[3]              = material name
                  (e.g. 15736507,S28_diff_housing_Al_GrA206_T4_sol, 3.709E+00,Aluminum_GrA206_T4,)
$val               parsed numeric yield value
$flg               found material ($true/$false)

```

```

#>
<#
=====
#>

$H = @{}
$Hnf = @{}

$Smaxall = -1
$Smaxall_tag = ""
$Sminall = 1
$Sminall_tag = ""

$Scnt = 0
$Scntd = 0

$Sstmp_in = (get-location).path + ("\"{0}" -f $fn)
$Ssr_rmd = new-object IO.StreamReader($Sstmp_in)

# skip header lines
while ($Snskip -gt 0) {

    $Srmd_data = $Ssr_rmd.ReadLine()
    write-host "Skipping:",$Srmd_data
    $Snskip = $Snskip - 1

}

# process entries
while (-not $Ssr_rmd.EndOfStream) {

    $Srmd_data = $Ssr_rmd.ReadLine()

    $Scnt = $Scnt + 1
    $Scntd = $Scntd + 1

    # display count after 5000 entries
    if ($Scntd -gt 5000) {

        write-host $Scnt
        $Scntd = 0

    }

    # split line (e.g. 15736507,S28_diff_housing_Al_GrA206_T4_sol, 3.709E+00,Aluminum_GrA206_T4,)
    $S1 = $Srmd_data -split ","
    $Sval = 0
    $Sval = [Double]::Parse($S1[2])

    # find material yield
    $Sflg = $false
    switch ($S1[3]) {

        # defined material and yield
        "Aluminum_GrA206_T4" { $Syl = 205; if ([System.Math]::Abs($Sval) -gt $Syl) {$Sflg = $true};break }
        "Aluminum_ASTM_B209_Gr5052_H32" { $Syl = 160; if ([System.Math]::Abs($Sval) -gt $Syl) {$Sflg = $true};break }
        "Steel_EN10149-2_GrS700MC" { $Syl = 700; if ([System.Math]::Abs($Sval) -gt $Syl) {$Sflg = $true};break }
        "Steel_4130H" { $Syl = 472; if ([System.Math]::Abs($Sval) -gt $Syl) {$Sflg = $true};break }
        "Steel_EN10025-6_GrS960QL" { $Syl = 960; if ([System.Math]::Abs($Sval) -gt $Syl) {$Sflg = $true};break }
        "Steel_ASTM514_GrH" { $Syl = 690; if ([System.Math]::Abs($Sval) -gt $Syl) {$Sflg = $true};break }
        "Steel_ASTM_A1011_CS_TypeB" { $Syl = 248; if ([System.Math]::Abs($Sval) -gt $Syl) {$Sflg = $true};break }
        "Steel_ASTM514_GrB" { $Syl = 690; if ([System.Math]::Abs($Sval) -gt $Syl) {$Sflg = $true};break }
        "Aluminum_Gr6061_T651" { $Syl = 269; if ([System.Math]::Abs($Sval) -gt $Syl) {$Sflg = $true};break }
        "Steel_ASTM656_Gr80" { $Syl = 552; if ([System.Math]::Abs($Sval) -gt $Syl) {$Sflg = $true};break }

        # not found
        default {
            write-host "material not found", $S1[3]
            if (-not ($Hnf.ContainsKey($S1[3]))) {

                $Hnf[$S1[3]] = 0

            }

        }

    }

    # get min/max of all
    if (($S1[1].Substring(0,4) -ne "test") -and ($Smaxall -lt $Sval)) { $Smaxall = $Sval;$Smaxall_tag = $Srmd_data}
    if (($S1[1].Substring(0,4) -ne "test") -and ($Sminall -gt $Sval)) { $Sminall = $Sval;$Sminall_tag = $Srmd_data}

    # process entry (material found)
    if ($Sflg) {

        # if first entry initialize
        if (-not ($H.ContainsKey($S1[1]))) {

            $H[$S1[1]] = @($Sval, $Sval, 1, $Syl)

        } else {

            # cnt
            $H[$S1[1]][2] = $H[$S1[1]][2] + 1

            # max
            if ($Sval -gt $H[$S1[1]][1]) {

                $H[$S1[1]][1] = $Sval

            }

            # min
            if ($Sval -lt $H[$S1[1]][0]) {

                $H[$S1[1]][0] = $Sval

            }

        }

    }

}

```

```

}

# close file
$sr_rmd.Close()
$sr_rmd.Dispose()

# output part results
$х.keys | foreach {

    ("{0,-50} {1,8:F1} {2,8:F1} {3,8:D} {4,8:F1}" -f $_, $h[$_][0], $h[$_][1], $h[$_][2], $h[$_][3])

}

# output overall results
("{0} {1}" -f $maxall,$maxall_tag)
("{0} {1}" -f $minall,$minall_tag)

# output not found materials
if ($hnf.Count -ne 0) {

    "===Not Found Property==="
    $hnf.keys | foreach {

        ("{0}" -f $_)

    }

}

write-host "Done."

<#
=====
#>
<#

      1      2      3      4      5      6      7      8      9      1      1      1      1
1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
#>

===EOF (yield_sel.ps1)

```

TN 22-0524-01

ALTAIR HYPERWORKS LICENSE USAGE MONITORING

May 24, 2022

W. Bylsma

U.S. Army Combat Capabilities Development Command (DEVCOM)
Ground Vehicle Systems Center (GVSC)
Detroit Arsenal
Modeling, Simulation & Software
ATTN: FCDD-GVR-MSS/MS 157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

How do you monitor the license usage for Altair HyperWorks?

PROBLEM SOLUTION:

The Altair Units Licensing Guide [1] specifies the Altair License Manager Utility (**almutil**) that can be called to show the license usage from a PowerShell [2] command prompt as:

```
& "C:\Program Files\Altair\ConnectMe\LicenseUtil\almutil.exe" -licstat -host xxx.xxx.22.18 -port xxx90 -inuse
```

Running the utility with no options will provide a list of available options. For the output see "Example almutil output". Using regular expressions [3] the individual feature data can be extracted, then each feature name, licenses used, total licenses, and denials can be found. This is repeated for the user licenses used and user name. All this information is combined into a single line and output to a log file with a timestamp using "**scan.ps1**". The time stamp (year, month, day, hour, minute, seconds) and features are separated by semicolons (";"). The feature data (name, licenses used, licenses available, denials, users) are separated by commas (",") and the users are separated by ampersands ("&") as:

```
20220516163402;Feature 1,20,100,0,user1&user2&user3;Feature 2,4,10,0,user4&user5;...
```

The "**scan.ps1**" script runs continuously as a PowerShell background job by submitting it with the "**go.ps1**" script (see Example go.ps1). PowerShell jobs can be stopped using the "Stop-Job -Id x" command. To test if the license usage data is being extracted properly use the "**gotest.ps1**" script. If the output from "**almutil**" changes the regular expression searches may need to be modified in "**scan.ps1**". The "**gotest.ps1**" script is "**go.ps1**" script with a sample time of zero, a log file name of "test", and does not run as a background job.

Example almutil output

```

AltairLM End-user Utility v14.5.0 build 208086 (Feb  3 2020)
Copyright (C) 2020 Altair Engineering, Inc. All rights reserved.

+++++
LM-X License Server on xxx90@xxx.xxx.22.18:

Server version: v5.1 Uptime: 11 day(s) 12 hour(s) 23 min(s) 57 sec(s)
-----
Feature: GlobalZoneAM Version: 21.0 Vendor: ALTAIR
Start date: 2021-11-01 Expire date: 2022-10-31
Key type: EXCLUSIVE License sharing: CUSTOM VIRTUAL

42000 of 358000 license(s) used:

21000 license(s) used by k.ca@LPTN0S1N2 [xxx.xxx.113.252]
  Login time: 2022-05-24 04:09   Checkout time: 2022-05-24 04:09
  Shared on custom string: k.ca:LPTN0S1N2

21000 license(s) used by k.ca@LPTN0S1N2 [xxx.xxx.113.252]
  Login time: 2022-05-24 04:11   Checkout time: 2022-05-24 04:11
  Shared on custom string: k.ca:LPTN0S1N2

21000 license(s) used by d.me@LPT6D3B4G3 [xxx.xxx.114.237]
  Login time: 2022-05-24 08:24   Checkout time: 2022-05-24 08:24
  Shared on custom string: d.me:LPT6D3B4G3

21000 license(s) used by d.me@LPT6D3B4G3 [xxx.xxx.114.237]
  Login time: 2022-05-24 08:24   Checkout time: 2022-05-24 08:24
  Shared on custom string: d.me:LPT6D3B4G3

6000 license(s) used by d.me@LPT6D3B4G3 [xxx.xxx.114.237]
  Login time: 2022-05-24 08:24   Checkout time: 2022-05-24 08:24
  Shared on custom string: d.me:LPT6D3B4G3

2000 license(s) used by d.me@LPT6D3B4G3 [xxx.xxx.114.237]
  Login time: 2022-05-24 08:30   Checkout time: 2022-05-24 08:30
  Shared on custom string: d.me:LPT6D3B4G3

0 denial(s) in last 24 hours
-----
Feature: HWHyperGraph Version: 21.0 Vendor: ALTAIR
Start date: 2021-11-01 Expire date: 2022-10-31
Key type: EXCLUSIVE License sharing: CUSTOM VIRTUAL

6000 of 2147483647 license(s) used:

6000 license(s) used by d.me@LPT6D3B4G3 [xxx.xxx.114.237]
  Login time: 2022-05-24 08:24   Checkout time: 2022-05-24 08:24
  Shared on custom string: d.me:LPT6D3B4G3:10568_1263116896_1653395098_xxx963

0 denial(s) in last 24 hours
-----
Feature: HWHyperMesh Version: 21.0 Vendor: ALTAIR
Start date: 2021-11-01 Expire date: 2022-10-31
Key type: EXCLUSIVE License sharing: CUSTOM VIRTUAL

21000 of 2147483647 license(s) used:

21000 license(s) used by k.ca@LPTN0S1N2 [xxx.xxx.113.252]
  Login time: 2022-05-24 04:11   Checkout time: 2022-05-24 04:11
  Shared on custom string: k.ca:LPTN0S1N2:20996_117608464_1653319041_xxx25

0 denial(s) in last 24 hours
-----
Feature: HWHyperView Version: 21.0 Vendor: ALTAIR
Start date: 2021-11-01 Expire date: 2022-10-31
Key type: EXCLUSIVE License sharing: CUSTOM VIRTUAL

42000 of 2147483647 license(s) used:

21000 license(s) used by k.ca@LPTN0S1N2 [xxx.xxx.113.252]
  Login time: 2022-05-24 04:09   Checkout time: 2022-05-24 04:09
  Shared on custom string: k.ca:LPTN0S1N2:21220_198677104_1653313342_xxx253

21000 license(s) used by d.me@LPT6D3B4G3 [xxx.xxx.114.237]
  Login time: 2022-05-24 08:24   Checkout time: 2022-05-24 08:24
  Shared on custom string: d.me:LPT6D3B4G3:10568_1218125856_1653395097_xxx710

0 denial(s) in last 24 hours
-----
Feature: HWMotionView Version: 21.0 Vendor: ALTAIR
Start date: 2021-11-01 Expire date: 2022-10-31
Key type: EXCLUSIVE License sharing: CUSTOM VIRTUAL
.
.
.

```

Example go.ps1

```
<#
    go.ps1 - submit scan as background job
#>

# get current directory
$cwd = Get-Location

# PS> .\scan [almutil] [host ip] [port] [seconds/sample] [logfile]
# PS> .\scan -almutil "c:\Altair\almutil.exe" -hostip xxx.xxx..22.18 -port xxx90 `
#           -st 300 -logfile .\logs\log

# submit job (use stop-job -id x to stop)
start-job -scriptblock {
    Set-Location $args[0]
    .\scan.ps1 -almutil "C:\Program Files\Altair\ConnectMe\LicenseUtil\almutil.exe" `
        -hostip xxx.xxx.22.18 `
        -port xxx90 `
        -st 300 `
        -logfile .\logs\log
} -ArgumentList $cwd
```

The “**get-usage.ps1**” script is used to processes log files to find license usage over time for features and users in terms of total days from the start date, time check sample delta to detect downtime or non-functioning license servers, feature percent of licenses used, and user percent of usage. A separate comma separated file for each feature’s data is created, along with an index of unique features and users in the given log file (see Appendix A.2). These can be imported into Microsoft Excel and analyzed. Also, a table file is created that can be used with MATLAB [4] to plot the usage data (see “**plotusage.m**” in Appendix A.3)

Example get-usage.ps1 call

```
# from the .\logs directory
# PS> ..\get-usage [log file] [usage file]

..\get-usage log_20220519080556 usage
```

Figure 1 shows an example usage plot from “**plotusage.m**” for the Feature, Users, and Sample Delta. Referring to Figure 1, the feature F009 (see the INDEX file) license usage never exceeds 25% of the total number of licenses. For users, the usage is shared between them with multiple feature usage per user being factored in (e.g., if user1 has two F009 features open they both count towards the total number in use). Features with users that have mostly 100% of usage indicates they are the only ones using that feature. A rise in the sample delta minutes indicates a server was down and license usage data was not collected during that time. Even if there are no licenses being used the time check will output a data line at even intervals (e.g., almutil will indicate 0 of x used when not using the -inuse option).

The “**get-feature.ps1**” script scans the license usage output to determine all of the available features and number of licenses for each. It can be run with the “**gofeat.ps1**” script (see Appendix A.4). The suggested directory structure is outlined as

```
.\logs
.\src
.\src\get-features.ps1
.\src\scan.ps1
gotest.ps1
go.ps1
get-usage.ps1
plotusage.m
gofeat.ps1
```

with the suggested order of usage as:

```
. \gotest
. \go
. \get-usage
. \plotusage
. \gofeat
```

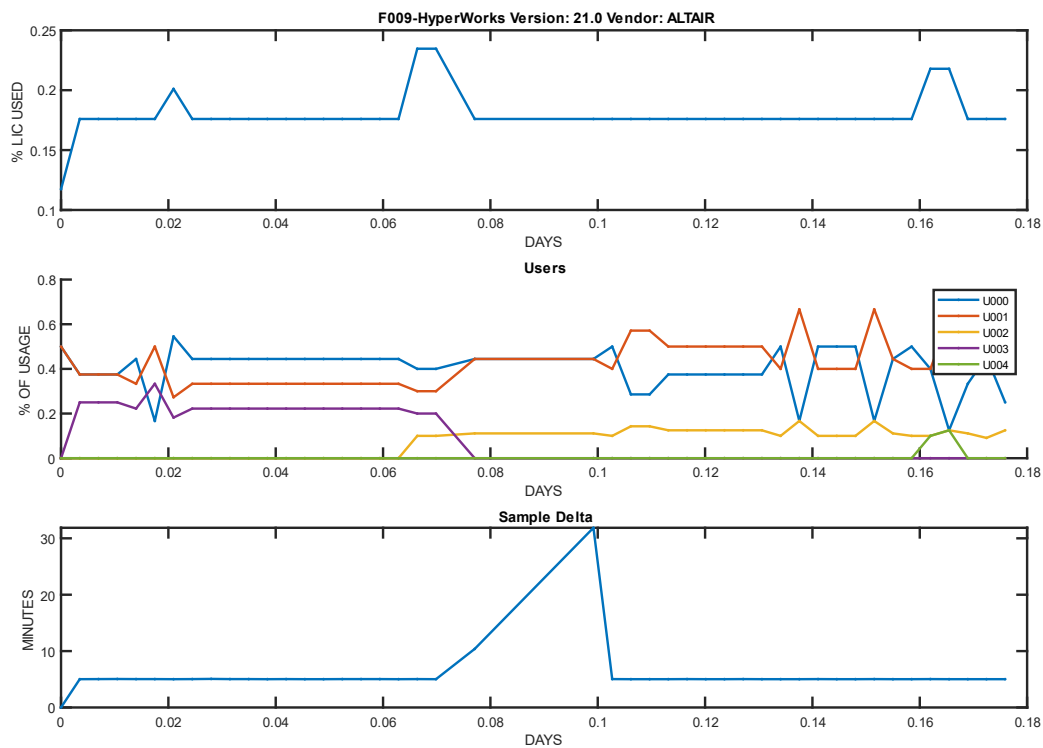


Figure 1 – Usage data for feature F009 (specific to log file)

Modifications can be made to the scripts for customized results.

REFERENCE

- [1] Altair Units 2020 Licensing Guide (almutil) (<https://www.altair.com>)
- [2] PowerShell (<https://docs.microsoft.com/en-us/powershell/>)
- [3] .NET regular expressions (<https://docs.microsoft.com/en-us/dotnet/standard/base-types/regular-expressions>)
- [4] MATLAB (<https://www.mathworks.com/products/matlab.html>)

APPENDIX

- A.1 SCAN.PS1
- A.2 GET-USAGE.PS1
- A.3 PLOTUSAGE.M
- A.4 GET-FEATURES.PS1

APPENDIX A.1– SCAN.PS1

```
<#
.SYNOPSIS
Scans license usage output to determine licenses used (features and users)

.DESCRIPTION
Searches license usage output from Altair's license utility (almutil) using regular expressions
to find license usage for features and users

.INPUTS
Input is the output from "almutil.exe" (its path must be specified). The pipe to "Out-String"
is required to preserve multiline aspects (see $usage variable). The host ip address, port number,
sample time (seconds), and log file must be specified. A sample time of zero will print out
one scan of license usage data and exit.

.OUTPUTS
Output is a single time stamped line containing the current features and users at the time check.
The time stamp (year, month, day, hour, minute, seconds) and features are separated by
semicolons (";"). The feature data (name, licenses used, licenses available, denials, users) is
separated by commas (",") and the users are separated by ampersands ("&").

20220516163402;Feature 1,20,100,0,user1&user2&user3;Feature 2,4,10,0,user4&user5;...

.EXAMPLE

PS> .\scan [almutil] [host ip] [port] [seconds/sample] [log_file]
PS> .\scan -almutil "c:\Altair\almutil.exe" -hostip pxxx.xxx.22.18 -port xxx90 `
    -st 300 -logfile .\logs\log

.NOTES
===REFERENCES===
[1] Altair Units 2020 Licensing Guide (almutil)

===PowerShell/.NET Notes===
# regular expression matches
[System.Text.RegularExpressions.MatchCollection]

#>

param ( [String] $almutil, [String] $hostip, [String] $port, [UInt32] $st, [String] $logfile )

#=====MAIN=====
<#
    Entry Point

    INPUTS:
    $almutil      path to Alair License Utility (almutil.exe)
    $hostip       license server IP address
    $port         license server port number
    $st           sample time (seconds)
    $logfile      logfile name for output

    VARIABLES:
    $log_flg      add time stamp to log file name if $true
    $tm           current date/time
    $ln           usage output line
    $usage        license usage data from almutil.exe

    $srch         feature search regular expression
    $m            regex match collection for $srch search
    $i            loop counter
    $gdata        match group "f" data

    $srch1        feature data search regular expression
    $m1           regex match collection for $srch1 search
    $g1           match group "name" data
    $g2           match group "used" data
    $g3           match group "of" data
    $g4           match group "den" data
    $val          numeric value of match group "used"

    $srch2        user data search regular expression
    $m2           regex match collection for $srch2 search
    $u            user name array (all users), use hashtable (unique users)
    $j            loop counter
    $g5           match group "user_used" data
    $g6           match group "user" data

    $f            test, features
    $n            test, feature index
    $d            test, feature data
    $u            test, user data

#>

$log_flg = $true
write-host "Starting License Usage Scan @:", (Get-Date)

# loop continually
while ($true) {

    # current date/time
    [DateTime] $tm = Get-Date
```

```

$ln = ("0,4:D4){1,2:D2}{2,2:D2}{3,2:D2}{4,2:D2}{5,2:D2}" -f $tm.year,$tm.month, $tm.day, $tm.hour, $tm.minute, $tm.second )

# add time stamp to log file name if $true
if ($log_flg) {
    $logfile = $logfile + "_" + $ln
    $log_flg = $false
}

# usage data
# NOTE:
# 1) needs all features available (do not use "-inuse") so if no features available then no time check, indicates
server down
# 2) requires "Out-String" to include line delimiters "`r`n"
# 3) to simulate input from a file, use "-raw" to include line delimiters
# $usage = get-content atest2 -encoding ascii -raw #-delimiter "`r`n"
[String] $usage = (& $almutil -licstat -host $hostip -port $port | Out-String)

# feature search criteria
<#
    'Feature: (?<f>[\S\s]*?denial\s\ in last)'          "Feature: " followed by named group "f" for
                                                         any one or more characters until
                                                         "denial (s) in last"
#>
$srch = 'Feature: (?<f>[\S\s]*?denial\s\ in last)'

$m = [Regex]::matches($usage, $srch, "Multiline")

# loop features
for ( $i = 0; $i -lt $m.Count; $i = $i + 1 ) {
    $gdata = $m[$i].groups["f"]      # only one group so no need check for Success=$true

    # feature data search criteria
    <#
        '^(?<name>.*?)\r?$){1}[\S\s]*(^(?<used>\d+) of (?<of>\d+) license\s\ used){1}[\S\s]*(^(?<den>\d+) denial)'
        (
            (?<name>.*?)                at beginning of line
            \r?$                        named group "name" for any characters to end of line
            )                           with optional \r
        ){1}                           one time
        [\S\s]*                        zero or more any characters
        (
            (?<used>\d+)                at beginning of line
            of                          named group "used" for one or more decimal digits
            (?<of>\d+)                  " of " characters
            license\s\ used            named group "of" for one or more decimal digits
            )                           " license(s) used" characters
        ){1}                           one time
        [\S\s]*                        zero or more any characters
        (
            (?<den>\d+)                at beginning of line
            denial                     named group "den" for one or more decimal digits
            )                           " denial" characters
        )
    #>
    $srch1 = '^(?<name>.*?)\r?$){1}[\S\s]*(^(?<used>\d+) of (?<of>\d+) license\s\ used){1}[\S\s]*(^(?<den>\d+) denial)'

    $m1 = [regex]::matches($gdata.value, $srch1, "Multiline")

    # get feature data
    if ($m1.Count -gt 0) {
        # only one time so use index 0
        $g1 = $m1[0].groups["name"]
        $g2 = $m1[0].groups["used"]
        $g3 = $m1[0].groups["of"]
        $g4 = $m1[0].groups["den"]

        # convert "used" licenses to numeric
        $val = [UInt32]::Parse($g2.value)

        # usage > 0, get users
        if ($val -gt 0) {
            # user data search criteria
            <#
                '^(?<user_used>\d+) license\s\ used by (?<user>.*?)\r?$)+'
                (
                    (?<user_used>\d+)    at beginning of line
                    license\s\ used by  named group "user_used" for one or more decimal digits
                    (?<user>.*?)         " license(s) used by " characters
                    \r?$                 named group for any characters to end of line
                    )                   with optional \r
                )+                     one or more times
            #>
            $srch2 = '^(?<user_used>\d+) license\s\ used by (?<user>.*?)\r?$)+'

            # since usage > 0, search $gdata again now for user data
            $m2 = [regex]::matches($gdata.value, $srch2, "Multiline")

            # unique users
            #$u = @{}
            # all users

```

```

$u = @()

# loop users
for ($j = 0; $j -lt $m2.Count; $j = $j + 1) {

    $g5 = $m2[$j].groups["user_used"]
    $g6 = $m2[$j].groups["user"]

    # for unique, if user not found, add
    #if (-not ($u.ContainsKey($g6.value))) {
    #    $u[$g6.value] = $g5
    #}
    # for all (for usage count)
    $u = $u + ,($g6.value)

} # loop users

# line data (unique users)
# $ln = [string[]]($ln,("{0},{1},{2},{3},{4}" -f $g1.value, $g2.value, $g3.value, $g4.value, ($u.keys -join "&")))
) -join ";"

# line data (all users)
$ln = [string[]]($ln,("{0},{1},{2},{3},{4}" -f $g1.value, $g2.value, $g3.value, $g4.value, ($u -join "&"))) -
join ";"

} # usage > 0, get users

} # get feature data

} # loop features ($i)

# if license server still running
if ($m.Count -gt 0) {

    # add timecheck
    $ln = $ln + ";TimeCheck,0,0,0,PowerShell"
    $ln | out-file -append -encoding ascii $logfile

}

# wait
Start-Sleep -Seconds $st

# exit if sample time zero ($st = 0)
if ($st -eq 0) {

    # print test data
    ("`n{0}" -f "Test:")
    $f = $ln -split ";"
    $f[0]

    # loop features (skip time stamp, start at index 1)
    (1..($f.length-1)) | foreach {

        $n = $f[$_]
        $d = $f[$n] -split ","
        ("{0,12} of {1,12} with {2,5} denials for {3}, users: {4}" -f $d[1], $d[2], $d[3], $d[0], ($d[4] -split "&").count)
        $u = $d[4] -split "&"

        # loop users
        $u | foreach {

            ("{0}{1}" -f [String]::Empty.PadLeft(30), $_)

        }

    } # test, loop features

    # exit scan
    break

}

}

```

Example scan.ps1 output (one line of log file broken at separators)

```

20220519080556;GlobalZoneAM Version: 21.0 Vendor: ALTAIR,42000,358000,0,
u1@PTHN0S1N2&u1@PTHN0S1N2&u2@PT6D3B4G3&u2@PT6D3B4G3&u2@PT6D3B4G3;
HWHyperGraph Version: 21.0 Vendor: ALTAIR,6000,2147483647,0,
u2@T6D3B4G3;
HWHyperMesh Version: 21.0 Vendor: ALTAIR,42000,2147483647,0,
u1@PTHN0S1N2&u1@PTHN0S1N2;
HWHyperView Version: 21.0 Vendor: ALTAIR,21000,2147483647,0,
u1@PTHN0S1N2;
HWMotionView Version: 21.0 Vendor: ALTAIR,21000,2147483647,0,
u2@PT6D3B4G3;
HWPDataManager Version: 21.0 Vendor: ALTAIR,2000,2147483647,0,
u2@PT6D3B4G3;
HyperWorks Version: 21.0 Vendor: ALTAIR,42000,358000,0,
u1@PTHN0S1N2&u1@PTHN0S1N2&u1@PTHN0S1N2&u2@PT6D3B4G3&u2@DRENLP6D3B4G3&u2@PT6D3B4G3;
TimeCheck,0,0,0,PowerShell
.
.
.

```

Example go.ps1 (to submit scan.ps1)

```

<#
    go.ps1 - submit scan as background job

#>

# get current directory
$cwd = Get-Location

# PS> .\scan [almutil] [host ip] [port] [seconds/sample] [logfile]
# PS> .\scan -almutil "c:\Altair\almutil.exe" -hostip xxx.xxx..22.18 -port xxx90 `
#           -st 300 -logfile .\logs\log

# submit job (use stop-job -id x to stop)
start-job -scriptblock {

    Set-Location $args[0]
    .\src\scan.ps1 -almutil "C:\Program Files\Altair\ConnectMe\LicenseUtil\almutil.exe" `
        -hostip xxx.xxx.22.18 `
        -port xxx90 `
        -st 300 `
        -logfile .\logs\log

} -ArgumentList $cwd

```

Example gotest.ps1 (to test if scan.ps1 regular expressions are extracting data correctly, almutil output may change)

```

<#
    gotest.ps1 - submit scan with sample time as zero (not as background job), modified go.ps1

#>

# PS> .\scan [almutil] [host ip] [port] [seconds/sample] [logfile]
# PS> .\scan -almutil "c:\Altair\almutil.exe" -hostip xxx.xxx..22.18 -port xxx90 `
#           -st 300 -logfile .\logs\log

.\src\scan.ps1 -almutil "C:\Program Files\Altair\ConnectMe\LicenseUtil\almutil.exe" `
    -hostip xxx.xxx.22.18 `
    -port xxx90 `
    -st 0 `
    -logfile .\logs\test

```

Example gotest.ps1 output

```

Test:
20220525073149
21000 of      358000 with      0 denials for GlobalZoneAM Version: 21.0 Vendor: ALTAIR, users: 2
                U3@PTG7TCC93
                U3@PTG7TCC93
21000 of      2147483647 with      0 denials for HWHyperMesh Version: 21.0 Vendor: ALTAIR, users: 1
                U3@PTG7TCC93
21000 of      2147483647 with      0 denials for HWHyperView Version: 21.0 Vendor: ALTAIR, users: 1
                U3@PTG7TCC93
21000 of      358000 with      0 denials for HyperWorks Version: 21.0 Vendor: ALTAIR, users: 2
                U3@PTG7TCC93
                U3@PTG7TCC93
0 of          0 with      0 denials for TimeCheck, users: 1
                PowerShell

```

APPENDIX A.2– GET-USAGE.PS1

```

<#
.SYNOPSIS
Extract license usage information from scan log file (features and users)

.DESCRIPTION
Processes log file to find license usage over time for features and users in terms of total days
from the start date, time check sample delta to detect downtime or non-functioning license server,
feature percent of licenses used, and user percent of usage.

.INPUTS
Input is the output log file from "scan.ps1"

.OUTPUTS
Output is a comma separated data file (for each feature) with time stamp, feature, and user
header data followed by data lines for the days from start date, minutes between sample intervals,
% of licenses used for this feature, and % used for each user. Included is an index file summarizing
abbreviations for each feature and user and table file for reading data in for plotting (see plotusage.m)

.EXAMPLE

PS> .\get-usage [logfile] [usagefile]
PS> .\get-usage .\logs\log_20220519080556 .\usage\usage

.NOTES
===REFERENCES===
[1] see scan.ps1

#>

param ( [String] $logfile, [String] $usagefile )

#=====FUNCTIONS=====
function GetDateTimeFromLine ([String] $l) {
<#

    return date and time from string

    INPUTS:
    $l                time stamp string (e.g. 20220519102322)

    VARIABLES:
    $tm                time stamp, left 14 characters
    $yr                year
    $mt                month
    $dy                day
    $hr                hour
    $mn                minutes
    $sc                seconds

#>

    $tm = $l.SubString(0,14)
    [Int32]$yr = $tm.SubString(0,4)
    [Int32]$mt = $tm.SubString(4,2)
    [Int32]$dy = $tm.SubString(6,2)
    [Int32]$hr = $tm.SubString(8,2)
    [Int32]$mn = $tm.SubString(10,2)
    [Int32]$sc = $tm.SubString(12,2)

    return @($yr, $mt, $dy, $hr, $mn, $sc)
}

function GetFeatureData ([String[]] $fl, [UInt32] $fnum) {
<#

    return feature name, used, total, denied information

    INPUTS:
    $fl                feature list
    $fnum                feature index

    $VARIABLES:
    $f11                feature data array
                        (e.g. HWHyperGraph Version: 21.0 Vendor: ALTAIR,6000,2147483647,0,u1@HH6D&u2@PT6D3B4G)
    $f12                user data array

#>

    $f11 = $fl[$fnum] -split ","
    $f12 = $f11[4] -split "&"

    return @( $f11[0], $f11[1], $f11[2], $f11[3], $f12) # name, used, total, denied
}

#=====MAIN=====
<#
    Entry Point

    INPUTS:
    $logfile            path to log file

```



```

$usagefile          license usage file

VARIABLES:
$f                  feature hash
$u                  users hash
$fau                features and users hash
$d                  license usage data

$l                  line data
$f1                 feature list (separated by ";")

$f_name             feature name
$f_used             feature licenses used
$f_tot              feature total licenses
$f_users            feature users array

$un                 user name
$f_sorted           sorted features
$u_sorted           sorted users
$yr                 year
$mt                 month
$dy                 day
$hr                 hour
$mn                 minutes
$sc                 seconds

$tstamp             time stamp string, 14 characters (e.g. 20220519102322)
$hdt                header, time stamp
$rows               index, rows
$cols               index, cols
$st                 start time

$fnum               feature name index
$f1                 feature name
$hdt                header, data column data
$hdt                header, data column names
$u_idx              user name index array
$unum               user name index
$un                 user name

$dt                 last time step date
$ct                 current date
$tdays              total days (current - start)
$tmmin              total minutes (current - start)

$h_data             current line feature data hashtable
$h2                 line data values
$uval               user count, per user
$utot               user count, total users

#>

write-host "Starting License Usage Extraction..."

$f = @{}
$u = @{}
$fau = @{}

# get license usage data
$d = get-content $logfile -encoding ascii
# to read large files faster
#[String[]] $d1
#$stmp_in = (get-location).path + ("\" + $logfile)
#$sr_rmd = new-object IO.StreamReader($stmp_in)
#while (-not $sr_rmd.EndOfStream) {
#    $rmd_data = $sr_rmd.ReadLine()
#    $d1 = $d1 + , ($rmd_data)
#}

# loop to find unique features and users
$d | foreach {

    $l = $_

    # get features
    $f1 = $l -split ";"

    # loop to find unique features and users
    (1..($f1.length-1)) | foreach {

        ($f_name, $f_used, $f_tot, $f_den, $f_users) = GetFeatureData $f1 $_

        # unique features
        if ( -not ($f.ContainsKey( $f_name )) ) {

            $f[$f_name] = @{}

        }

        # unique users
        $f_users | foreach {

            $un = $_
            # remove IP address (NOTE 1)
            $un = (($un -split " ")[0])

```

```

        if ( -not ($u.ContainsKey($un)) ) {

            $u[$un] = 1

        }

        if ( -not ($f[$f_name].ContainsKey($un)) ) {

            $f[$f_name][$un] = 1

        }

    }

} # end loop features

} # end loop data

# sort features
$f.Remove("TimeCheck")
[string[]]$f_sorted = $f.keys | sort

# sort users
$u.Remove("Powershell")
[string[]]$u_sorted = $u.keys | sort

# start date
($yr, $mt, $dy, $hr, $mn, $sc) = GetDateTimeFromLine $d[0]
$tstamp = $d[0].Substring(0,14)
#$hd_dt = ("START,,{0},{1},{2},{3},{4},{5}`n" -f $yr, $mt, $dy, $hr, $mn, $sc)
$hd_dt = ("START,{0}" -f $tstamp)

# index data
$rows = $d.count
$cols = @{}

===FEATURE USAGE

# start time
$st = New-Object -TypeName DateTime -ArgumentList (GetDateTimeFromLine $tstamp)

# loop features
(0..($f_sorted.length - 1)) | foreach {

    $fnum = $_
    $fn = $f_sorted[$fnum]
    write-host "    $fn"
    $hd0 = (" , ,{0}" -f $fn)
    $hd1 = ("DAYS,TimeCheck,F{0,3:D3}" -f $fnum)
    $cols[$fn] = 3

    $u_indx = @()
    # loop users
    (0..($u_sorted.length - 1)) | foreach {

        $unum = $_
        $un = $u_sorted[$unum]

        if ( $f[ $fn ].ContainsKey( $un ) ) {

            $u_indx = $u_indx + ,($unum)
            $hd0 = $hd0 + (",{0}" -f $un)
            $hd1 = $hd1 + ("U{0,3:D3}" -f $unum)
            $cols[$fn] = $cols[$fn] + 1

        }

    }

}

# output header
$(

    $hd_dt
    $hd0
    $hd1

) | out-file -encoding ASCII ("{0}_F{1,3:D3}_{2}" -f $usagefile, $fnum, $tstamp)

# set delta time
$dt = $st

# loop output usage
$d | foreach {

    $l = $_

    ($yr, $mt, $dy, $hr, $mn, $sc) = GetDateTimeFromLine $l
    $ct = New-Object -TypeName DateTime -ArgumentList @($yr,$mt,$dy,$hr,$mn,$sc)
    $tdys = ($ct - $st).totaldays
    $tmin = ($ct - $dt).totalminutes
    $dt = $ct

    $h_data = @{}
    $fl = $l -split ";"

```

```

(1..($f1.length-1)) | foreach {

    ($f_name, $f_used, $f_tot, $f_den, $f_users) = GetFeatureData $f1 $_
    $h_data[$f_name] = @($f_used, $f_tot, $f_den, $f_users)

}

#write-host "fn:",$fn
if ($h_data.ContainsKey($fn)) {

    $hd2 = ("{0},{1},{2}" -f $tdys, $tmin, ($h_data[$fn][0]/$h_data[$fn][1]) )

    # loop users
    $u_indx | foreach {
        $unum = $_
        $uval = ($h_data[$fn][3] | where {$_ -eq $u_sorted[$unum]}).count
        # remove IP address (see NOTE 1)
        $uval = ($h_data[$fn][3] | where {($_ -split " ")[0] -eq $u_sorted[$unum]}).count
        $utot = ($h_data[$fn][3]).count
        $hd2 = $hd2 + (",{0}" -f ($uval/$utot) )
    }

} else {

    $hd2 = ("{0},{1},{2}" -f $tdys, $tmin, 0)

    # loop users
    $u_indx | foreach {
        $hd2 = $hd2 + ",0"
    }

}

$(

    $hd2

) | out-file -encoding ASCII -append ("{0}_F{1,3:D3}_{2}" -f $usagefile, $fnum, $tstamp)

}

}

===USAGE INDEX
$(

    # start date
    $hd_dt
    ("{0}" -f "")

    # output feature index
    ("FEATURES,,{0}" -f $f.keys.count)
    (0..($f_sorted.length - 1)) | foreach {

        $fnum = $_
        $fn = $f_sorted[$fnum]
        ("F{0,3:D3},{1},{2}" -f $fnum, $fn, $f[$fn].Count)
        #("F{0,3:D3},{1},{2},{3},{4}" -f $fnum, $fn, $f[$fn].Count, $rows, $cols[$fn])

    }

    # output user index
    ("{0}" -f "")
    ("USERS,,{0}" -f $u.keys.count)
    (0..($u_sorted.length-1)) | foreach {

        ("U{0,3:D3},{1}" -f $_, $u_sorted[$_])

    }

) | out-file -encoding ascii ("{0}_INDEX_{1}" -f $usagefile, $tstamp)

===TABLE INDEX
$(

    # headers
    ("KEY,DESC")

    # start date
    $hd_dt

    # name
    ("LOG,{0}" -f $usagefile)

    # output feature index
    (0..($f_sorted.length - 1)) | foreach {

        $fnum = $_
        $fn = $f_sorted[$fnum]
        ("F{0,3:D3},{1}" -f $fnum, $fn)

    }

    # output user index
    (0..($u_sorted.length-1)) | foreach {

```

```

("U{0,3:D3},{1}" -f $_, $u_sorted[$_])
}
) | out-file -encoding ascii ("{0}_TABLE_{1}" -f $usagefile, $tstamp)

```

Example get-usage.ps1 output (feature file, one for each feature)

```

START,20220519080556
, ,HWHyperMesh Version: 21.0 Vendor: ALTAIR,k.ca@PTHN0S1N2,r.re@PT493B4G3
DAYS,TimeCheck,F002,U001,U002
0,0,1.95577740760323E-05,1,0
0.0034837962962963,5.01666666666667,1.95577740760323E-05,1,0
0.0069791666666667,5.03333333333333,1.95577740760323E-05,1,0
0.010486111111111,5.05,1.95577740760323E-05,1,0
0.0139814814814815,5.03333333333333,1.95577740760323E-05,1,0
0.0174768518518519,5.03333333333333,1.95577740760323E-05,1,0
0.0209606481481481,5.01666666666667,1.95577740760323E-05,1,0
0.0244560185185185,5.03333333333333,1.95577740760323E-05,1,0
0.027974537037037,5.06666666666667,1.95577740760323E-05,1,0
0.0314699074074074,5.03333333333333,1.95577740760323E-05,1,0
0.034965277777778,5.03333333333333,1.95577740760323E-05,1,0
0.0384490740740741,5.01666666666667,1.95577740760323E-05,1,0
0.0419444444444444,5.03333333333333,1.95577740760323E-05,1,0
0.0454282407407407,5.01666666666667,1.95577740760323E-05,1,0
0.048912037037037,5.01666666666667,1.95577740760323E-05,1,0
0.0524074074074074,5.03333333333333,1.95577740760323E-05,1,0
0.055902777777778,5.03333333333333,1.95577740760323E-05,1,0
0.0593981481481481,5.03333333333333,1.95577740760323E-05,1,0
0.0628819444444444,5.01666666666667,1.95577740760323E-05,1,0
0.0663773148148148,5.03333333333333,2.93366611140485E-05,0.66666666666667,0.33333333333333
0.069861111111111,5.01666666666667,2.93366611140485E-05,0.66666666666667,0.33333333333333
0.0770717592592593,10.38333333333333,3.91155481520647E-05,0.75,0.25
0.0992013888888889,31.8666666666667,3.91155481520647E-05,0.75,0.25
0.102696759259259,5.03333333333333,3.91155481520647E-05,0.75,0.25
0.106180555555556,5.01666666666667,3.91155481520647E-05,0.75,0.25
0.109664351851852,5.01666666666667,3.91155481520647E-05,0.75,0.25
0.113148148148148,5.01666666666667,3.91155481520647E-05,0.75,0.25
0.116643518518519,5.03333333333333,3.91155481520647E-05,0.75,0.25
0.120127314814815,5.01666666666667,3.91155481520647E-05,0.75,0.25
0.123611111111111,5.01666666666667,3.91155481520647E-05,0.75,0.25
0.127106481481481,5.03333333333333,3.91155481520647E-05,0.75,0.25
0.130590277777778,5.01666666666667,3.91155481520647E-05,0.75,0.25
0.134074074074074,5.01666666666667,3.91155481520647E-05,0.75,0.25
0.13755787037037,5.01666666666667,3.91155481520647E-05,0.75,0.25
0.141053240740741,5.03333333333333,3.91155481520647E-05,0.75,0.25
0.144537037037037,5.01666666666667,3.91155481520647E-05,0.75,0.25
0.148020833333333,5.01666666666667,3.91155481520647E-05,0.75,0.25
0.151516203703704,5.03333333333333,3.91155481520647E-05,0.75,0.25
0.155,5.01666666666667,3.91155481520647E-05,0.75,0.25
0.158483796296296,5.01666666666667,3.91155481520647E-05,0.75,0.25
0.161979166666667,5.03333333333333,3.91155481520647E-05,0.75,0.25
0.165462962962963,5.01666666666667,4.88944351900809E-05,0.8,0.2
0.168946759259259,5.01666666666667,4.88944351900809E-05,0.8,0.2
0.172430555555556,5.01666666666667,4.88944351900809E-05,0.8,0.2
0.175914351851852,5.01666666666667,4.88944351900809E-05,0.8,0.2.

```

Example get-usage.ps1 output (index file)

```

START,20220519080556

FEATURES,,10
F000,GlobalZoneAM Version: 21.0 Vendor: ALTAIR,5
F001,HWHyperGraph Version: 21.0 Vendor: ALTAIR,1
F002,HWHyperMesh Version: 21.0 Vendor: ALTAIR,2
F003,HWHyperView Version: 21.0 Vendor: ALTAIR,3
F004,HWInspire Version: 21.0 Vendor: ALTAIR,1
F005,HWMotionSolve Version: 21.0 Vendor: ALTAIR,1
F006,HWMotionView Version: 21.0 Vendor: ALTAIR,1
F007,HWPDataManager Version: 21.0 Vendor: ALTAIR,1
F008,HWTextView Version: 21.0 Vendor: ALTAIR,1
F009,HyperWorks Version: 21.0 Vendor: ALTAIR,5

USERS,,5
U000,d.me@PT6D3B4G3
U001,k.ca@PTHN0S1N2
U002,r.re@PT493B4G3
U003,w.by@PTG7TCC93 [xxx.xxx.xxx.180]
U004,w.by@PTG7TCC93 [xxx.xxx.xxx.213]

```

Example get-usage.ps1 output (table file)

```
KEY,DESC
START,20220519080556
LOG,log
F000,GlobalZoneAM Version: 21.0 Vendor: ALTAIR
F001,HWHyperGraph Version: 21.0 Vendor: ALTAIR
F002,HWHyperMesh Version: 21.0 Vendor: ALTAIR
F003,HWHyperView Version: 21.0 Vendor: ALTAIR
F004,HWInspire Version: 21.0 Vendor: ALTAIR
F005,HWMotionSolve Version: 21.0 Vendor: ALTAIR
F006,HWMotionView Version: 21.0 Vendor: ALTAIR
F007,HWPDataManager Version: 21.0 Vendor: ALTAIR
F008,HWTextView Version: 21.0 Vendor: ALTAIR
F009,HyperWorks Version: 21.0 Vendor: ALTAIR
U000,d.me@PT6D3B4G3
U001,k.ca@PTHN0S1N2
U002,r.re@PT493B4G3
U003,w.by@PTG7TCC93 [xxx.xxx.xxx.180]
U004,w.by@PTG7TCC93 [xxx.xxx.xxx.213]
```

APPENDIX A.3– PLOTUSAGE.M

```

%
% MATLAB '9.5.0.944444 (R2018b)'
% plotusage.m - plot license usage log data (uses readtable, strcmp, find)
%
% INPUTS:
% fdir          table file directory
% ftbl          table file (see get-usage.ps1)
%
% VARIABLES:
% y             table file data
% ss            START table row number
% ll            LOG table row number
% ii            F (feature) table row numbers (e.g. F002)
% i             loop index, features ii
% n             feature file name (e.g. log_F002_20220519103915)
%
% yy            feature file data
% dd            DAYS column number
% ff            F (feature) column number
% uu            U (user) column numbers
% tt            TimeCheck column number
%
% f             plot figure
% utitle        user title cell array
% lgd           legend name cell array
% lgdc          legend cell array position index
%
% u             loop index, users uu
% j             user table row number
%
%=====MAIN=====

close all
clear

% set the log TABLE file to read
fdir = './logs\'
ftbl = 'log_TABLE_20220519080556'

%=====MAIN

% read TABLE file
y = readtable([fdir ftbl], 'Delimiter',';', 'ReadVariableNames',1);
ss = find(strcmp(y.KEY,"START",5));
ll = find(strcmp(y.KEY,"LOG",3));
ii = find(strcmp(y.KEY,"F",1));

% loop feature files
for i = ii'

    % file name
    n = [y.DESC{ll} '_' y.KEY{i} '_' y.DESC{ss}];

    yy = readtable([fdir n], 'Delimiter',';', 'ReadVariableNames',1, 'HeaderLines',2)

    % find column numbers by name
    dd = find(strcmp(yy.Properties.VariableNames,"DAYS",4))
    ff = find(strcmp(yy.Properties.VariableNames,"F",1))
    uu = find(strcmp(yy.Properties.VariableNames,"U",1))
    tt = find(strcmp(yy.Properties.VariableNames,"TimeCheck",9))

    % plot
    f = figure(i)
    f.Position = [1000 540 1200 800]
    clf

    %===== feature, % lic used
    subplot(3,1,1)
    cla

    plot(yy(:,dd),yy(:,ff),'LineStyle','-','Marker','.')
    title([y.KEY{i} '_' y.DESC{ss}])
    ylabel('% LIC USED')
    xlabel('DAYS')
    axis auto

    %===== users, % of usage
    utitle = {}
    lgd = {}
    lgdc = 1
    axis auto

    subplot(3,1,2)
    cla
    hold on

    % loop users
    for u = uu

        plot(yy(:,dd),yy(:,u),'LineStyle','-','Marker','.')

        % create legend and title

```

```

        lgd(lgdc) = yy.Properties.VariableNames{u}
        j = find(strcmp(y.KEY,yy.Properties.VariableNames{u}))
        utitle(lgdc) = [yy.Properties.VariableNames{u} '-' y.DESC{j}]
        lgdc = lgdc + 1
    end
    title('Users')
    ylabel('% OF USAGE')
    xlabel('DAYS')
    legend(lgd)
    ylim([0 1])
    axis auto

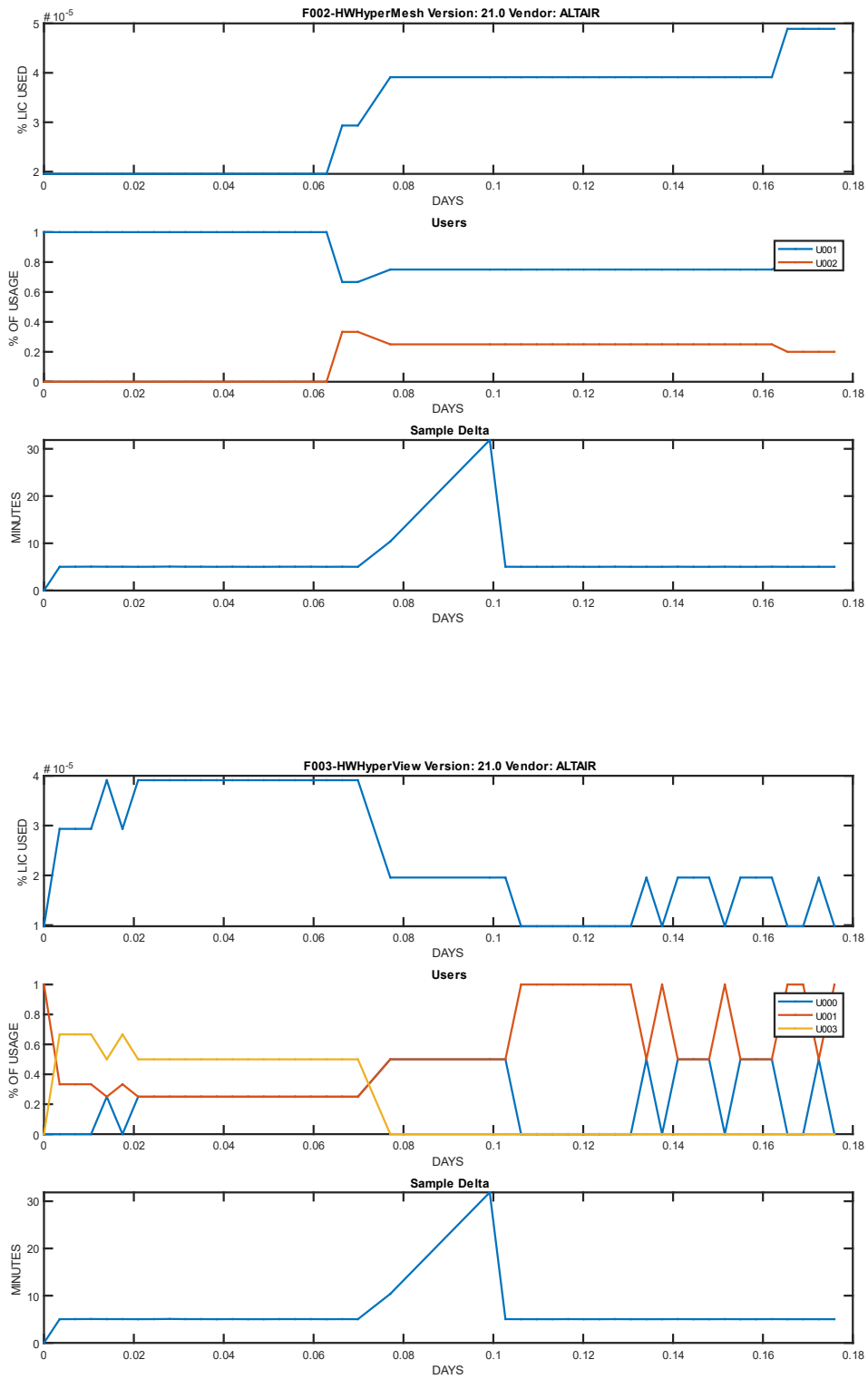
    %===== sample delta
    subplot(3,1,3)
    cla

    plot(yy(:,dd),yy(:,tt),'LineStyle','-','Marker','.')
    title('Sample Delta')
    ylabel('MINUTES')
    xlabel('DAYS')
    axis auto
end

<#
.SYNOPSIS
Extract license

```

Example plotusage.m output (one for each feature)



APPENDIX A.4– GET-FEATURES.PS1

```

<#
.SYNOPSIS
Scans license usage output to determine features (a version of scan.ps1)

.DESCRIPTION
Searches license usage output from Altair's license utility (almutil) using regular expressions
to find license usage for features

.INPUTS
Input is the output from "almutil.exe" (its path must be specified). The pipe to "Out-String"
is required to preserve multiline aspects (see $usage variable). The host ip address and
port number must be specified.

.OUTPUTS
Output is a list of the number of licenses for each feature.

.EXAMPLE

PS> .\get-features [almutil] [host ip] [port]
PS> .\get-features -almutil "c:\Altair\almutil.exe" -hostip xxx.xxx.22.18 -port xxx90

.NOTES
===REFERENCES===
[1] Altair Units 2020 Licensing Guide (almutil)

===PowerShell/.NET Notes===
# regular expression matches
[System.Text.RegularExpressions.MatchCollection]

#>

param ( [String] $almutil, [String] $hostip, [String] $port )

#-----MAIN-----
<#
    Entry Point

    INPUTS:
    $almutil      path to Alair License Utility (almutil.exe)
    $hostip       license server IP address
    $port         license server port number

    VARIABLES:
    $f            feature hashtable

    $srch         feature search regular expression
    $m            regex match collection for $srch search
    $i            loop counter
    $gdata        match group "f" data

    $srchl        feature data search regular expression
    $m1           regex match collection for $srchl search
    $g1           match group "name" data
    $g2           match group "used" data
    $g3           match group "of" data
    $g4           match group "den" data

#>

$f = @{}

# usage data
# NOTE:
# 1) needs all features available (do not use "-inuse") so if no features available then no time check, indicates server
# 2) requires "Out-String" to include line delimiters "r'n"
# 3) to simulate input from a file, use "-raw" to include line delimiters
# $usage = get-content atest2 -encoding ascii -raw #-delimiter "r'n"
[String] $usage = (& $almutil -licstat -host $hostip -port $port | Out-String)

# feature search criteria
<#
    'Feature: (?<f>[\S\s]*?denial\s\ in last)'          "Feature: " followed by named group "f" for
                                                         any one or more characters until
                                                         "denial (s) in last"

#>
$srch = 'Feature: (?<f>[\S\s]*?denial\s\ in last)'

$m = [Regex]::Matches($usage, $srch, "Multiline")

# loop features
for ( $i = 0; $i -lt $m.Count; $i = $i + 1 ) {

    $gdata = $m[$i].Groups["f"]      # only one group so no need check for Success=$true

    # feature data search criteria
    <#
        '^(?<name>.*?)\r?\$}{1}[\S\s]*(^(?<used>\d+) of (?<of>\d+) license\s\ used){1}[\S\s]*(^(?<den>\d+) denial)'

        (^                                     at beginning of line
         (?<name>.*?)                          named group "name" for any characters to end of line
         \r?\$                                with optional \r
    
```

```

) {1}
[\S\s]*
(^
    (?<used>\d+)
    of
    (?<of>\d+)
    license\s\ used
) {1}
[\S\s]*
(^
    (?<den>\d+)
    denial
)

#>
$srchl = '^(?<name>.*?)\r?$){1}[\S\s]*(^(?<used>\d+) of (?<of>\d+) license\s\ used){1}[\S\s]*(^(?<den>\d+) denial)'

$m1 = [regex]::matches($gdata.value, $srchl, "Multiline")

# get feature data
if ($m1.Count -gt 0) {

    # only one time so use index 0
    $g1 = $m1[0].groups["name"]
    $g2 = $m1[0].groups["used"]
    $g3 = $m1[0].groups["of"]
    $g4 = $m1[0].groups["den"]

    # convert "used" licenses to numeric
    #$val = [UInt32]::Parse($g2.value)
    $f[$g1] = @($g2.value, $g3.value, $g4.value)

} # get feature data

} # loop features

===FEATURE USAGE
# output feature index
("{0,12} {1}" -f "FEATURES", $f.keys.count)
$f.keys | sort | foreach {
    ("{0,12} {1}" -f $f[$_][1], $_)
    #("F{0,3:D3},{1},{2},{3},{4}" -f $fnum, $fn, $f[$fn].Count, $rows, $cols[$fn])
}
<#
.SYNOPSIS
Scans license usage output to determine licenses used (features and users)

.DESRIPTION

```

Example get-features.ps1 output

```

FEATURES 281
358000 GlobalZoneAM Version: 21.0 Vendor: ALTAIR
2147483647 HWActivate Version: 21.0 Vendor: ALTAIR
2147483647 HWAcuconsole Version: 21.0 Vendor: ALTAIR
2147483647 HWAcufieldview Version: 21.0 Vendor: ALTAIR
2147483647 HWAcufwh Version: 21.0 Vendor: ALTAIR
2147483647 HWAcumeshsim Version: 21.0 Vendor: ALTAIR
2147483647 HWAcusolve Version: 21.0 Vendor: ALTAIR
2147483647 HWAcutrace Version: 21.0 Vendor: ALTAIR
2147483647 HWAcuview Version: 21.0 Vendor: ALTAIR
2147483647 HWAIFPBS Version: 21.0 Vendor: ALTAIR
2147483647 HWAltairBushingModel Version: 21.0 Vendor: ALTAIR
2147483647 HWAltairManufacturingSolver Version: 21.0 Vendor: ALTAIR
2147483647 HWAltairOneDesktop Version: 21.0 Vendor: ALTAIR
2147483647 HWAltairOneEnterpriseUser Version: 21.0 Vendor: ALTAIR
2147483647 HWAnalyticsPBS Version: 21.0 Vendor: ALTAIR
2147483647 HWAnalyticsUser Version: 21.0 Vendor: ALTAIR
2147483647 HWAUPF Version: 21.0 Vendor: ALTAIR
2147483647 HWBatchMesher Version: 21.0 Vendor: ALTAIR
2147483647 HWClick2CastGUI Version: 21.0 Vendor: ALTAIR
2147483647 HWClick2CastSolver Version: 21.0 Vendor: ALTAIR
.
.
.
2147483647 HWSimLabPrePost Version: 21.0 Vendor: ALTAIR
2147483647 HWSimLabRadiossSolver Version: 21.0 Vendor: ALTAIR
2147483647 HWSimSolid Version: 21.0 Vendor: ALTAIR
2147483647 HWSimSolidAdvanced Version: 21.0 Vendor: ALTAIR
2147483647 HWSimSolidBasic Version: 21.0 Vendor: ALTAIR
2147483647 HWSolidThinking Version: 8.0 Vendor: ALTAIR
2147483647 HWTableView Version: 21.0 Vendor: ALTAIR
2147483647 HWTemplex Version: 21.0 Vendor: ALTAIR
2147483647 HWTextView Version: 21.0 Vendor: ALTAIR
358000 HyperWorks Version: 21.0 Vendor: ALTAIR
2147483647 SolverInApp Version: 21.0 Vendor: ALTAIR

```

Example gofeat.ps1 (to submit go-features.ps1)

```

<#
    gofeat.ps1 - submit scan once to get features
#>

# PS> .\get-features.ps1 [almutil] [host ip] [port]
# PS> .\get-features.ps1 -almutil "c:\Altair\almutil.exe" -hostip xxx.xxx.22.18 -port xxx90

.\src\get-features.ps1 -almutil "C:\Program Files\Altair\ConnectMe\LicenseUtil\almutil.exe" `
    -hostip xxx.xxx.22.18 `
    -port xxx90 `

```

TN 22-1101-01

POWERSHELL SCRIPT AUTO STARTUP

November 1, 2022

W. Bylsma

U.S. Army Combat Capabilities Development Command (DEVCOM)
Ground Vehicle Systems Center (GVSC)
Detroit Arsenal
Modeling, Simulation & Software
ATTN: FCDD-GVR-MSS/MS 157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

How do you get a PowerShell script to automatically startup when you login to a Windows 10 computer?

PROBLEM SOLUTION:

Without Administrator privileges you cannot use triggers to automatically start jobs. However, if you don't mind having a command window open there are other ways.

Using the Command Interpreter

Create the following script as a text file and save it with the ".cmd" suffix.

Example ps_startup.cmd

```
# ps_startup.cmd

# set the start in directory, change to it, and run the Powershell script "go.ps1" using the executionpolicy, noexit, and
# file options
set start_in="C:\...\APPLICATIONS\work"
cd %start_in%
C:\...\PowerShell\v1.0\powershell.exe -executionpolicy unrestricted -noexit -file .\go.ps1
```

Copy this file into the Startup directory and you are all set. This assumes you have a valid "go.ps1" PowerShell script that starts a background job and valid directory paths. To find the Startup directory press {Win key}+R to get the "Run" dialog, then type "shell:startup" and press "Enter" or click "OK" (e.g., C:\Users\...\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup).

Using a Shortcut

Create a shortcut in the Startup directory and edit the following properties (right-click on the shortcut and select Properties/Shortcut) as shown in Figure 1:

TARGET: C:\Users\...\PowerShell\v1.0\powershell.exe -executionpolicy unrestricted -noexit -file .\go.ps1
START IN: C:\Users\...\APPLICATIONS\work
Layout/Window Size/Height: 15

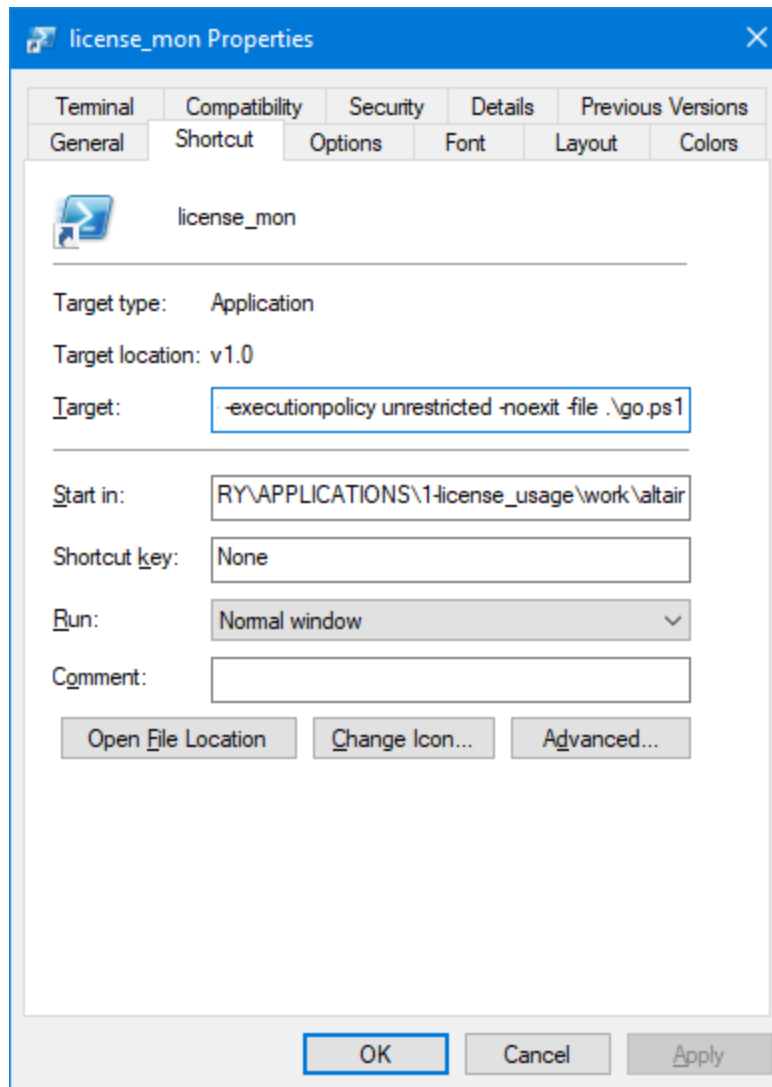


Figure 1 – Shortcut Properties

This also assumes you have a valid “go.ps1” PowerShell script that starts a background job and valid directory paths.

An example “go.ps1” script that starts a background job is shown below.

Example go.ps1

```
<#
    go.ps1 - submit ps1job as background job
#>

# get current directory
$cwd = Get-Location

# PS> .\ps1job [param 1] [param 2] [etc]
# PS> .\ps1job .\logs\logfile

# submit job (use stop-job -id x to stop)
start-job -scriptblock {

    Set-Location $args[0]
    .\ps1job.ps1 .\logs\logfile
} -ArgumentList $cwd
```

The “**ps1job.ps1**” script runs continuously as a PowerShell background job by submitting it with the “**go.ps1**” script (see Example go.ps1). PowerShell jobs can be stopped using the “Stop-Job -Id x” command. You can change the example directory and script names---just be consistent. Modifications can be made to the scripts for customized results.

REFERENCE

- [1] Command Interpreter - CMD.exe (<https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/cmd>) PowerShell (<https://learn.microsoft.com/en-us/powershell/>)
- [2] Powershell Command Line Options (https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_powershell_exe?view=powershell-5.1)
- [3] PowerShell (<https://learn.microsoft.com/en-us/powershell/>)
- [4] PowerShell Jobs (https://learn.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_jobs?view=powershell-7.2)
- [5] TN22-0524-01 ALTAIR HYPERWORKS LICENSE USAGE MONITORING

TN 22-1103-01

ALTAIR HYPERWORKS LICENSE ACTIVITY MATRIX

November 3, 2022

W. Bylsma

U.S. Army Combat Capabilities Development Command (DEVCOM)
Ground Vehicle Systems Center (GVSC)
Detroit Arsenal
Modeling, Simulation & Software
ATTN: FCDD-GVR-MSS/MS 157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

How do you monitor the license activity for Altair HyperWorks?

PROBLEM SOLUTION:

The Altair Units Licensing Guide [1] specifies the Altair License Manager Utility (**almutil**) that can be called to show the license usage from a PowerShell [2] command prompt as:

& "C:\Program Files\Altair\ConnectMe\LicenseUtil\almutil.exe" -licstat -host xxx.xxx.22.18 -port xxx90 -inuse

Running the utility with no options will provide a list of available options. A process was developed in [2] to determine the license usage for each feature and user.

Activity is defined as being in use for each sample. Based on the **"get-usage.ps1"** script in [2] a **"get-activity.ps1"** script was created and is used to process a log file to find activity for features and users in terms of percent in use for the total number of samples. A comma separated matrix file for the activity of each user and feature is created, along with the start date, end date, and number of samples in the given log file (see Appendix A.1). This can be imported into Microsoft Excel and analyzed as shown in Figure 1.

Example get-activity.ps1 call

```
# from the .\logs directory
# PS> ..\get-activity [log file] [usage file]

..\get-activity log_20220519080556 activity
```

Figure 1 – Activity Matrix (specific to log file)

Modifications can be made to the scripts for customized results.

[1] Altair Units 2020 Licensing Guide (almutil) (<https://www.altair.com>)

[3] PowerShell (<https://learn.microsoft.com/en-us/powershell/>)

A.1 GET-ACTIVITY.PS1

APPENDIX A.1– GET-ACTIVITY.PS1

```

<#
.SYNOPSIS
Extract license usage activity information from scan log file (features and users).

.DESCRIPTION
Processes log file to find license usage over time for features and users in terms of activity.
Activity is defined as being in use for each sample.

.INPUTS
Input is the output log file from "scan.ps1"

.OUTPUTS
Output is a comma separated data file matrix for each user (rows) and feature (columns).
The matrix values represent the total amount of samples the user/feature was in use for the
number of samples in the input file.

.EXAMPLE

PS> .\get-activity [logfile] [usagefile]
PS> .\get-activity .\logs\log_20220519080556 .\activity\activity

.NOTES
===REFERENCES===
[1] see scan.ps1
[2] see get-usage.ps1
[3] TypeAccelerators (https://4sysops.com/archives/using-powershell-type-accelerators/
    #listing-all-type-accelerators)
    [PSObject].Assembly.GetType('System.Management.Automation.TypeAccelerators')::Ge

#>

param ( [String] $logfile, [String] $usagefile )

#=====FUNCTIONS=====
function GetFeatureData ([String[]] $fl, [UInt32] $fnum) {
<#

    return feature name, used, total, denied information

20221025072308;
GlobalZoneAM Version: 22.0 Vendor: ALTAIR,126000,252000,30000,m.mull&Jill;
HWHyperGraph Version: 22.0 Vendor: ALTAIR,6000,2147483647,0,Jill;
HWHyperMesh Version: 22.0 Vendor: ALTAIR,63000,2147483647,0,kumar&ven;
HyperWorks Version: 22.0 Vendor: ALTAIR,126000,252000,0,m.mull&Jill;
TimeCheck,0,0,0,PowerShell

    INPUTS:
    $fl          feature list
    $fnum        feature index

    $VARIABLES:
    $fl1         feature data array
                  (e.g. HWHyperGraph Version: 21.0 Vendor: ALTAIR,6000,2147483647,0,u1@HH6D&u2@PT6D3B4G)
    $fl2         user data array

#>

    $fl1 = $fl[ $fnum ] -split ","
    $fl2 = $fl1[4] -split "&"

    return @( $fl1[0], $fl1[1], $fl1[2], $fl1[3], $fl2) # name, used, total, denied
}

function GetUniqueFeaturesUsers ([String[]] $fl, [HashTable] $f, [HashTable] $u) {
<#

    return feature name, used, total, denied information (from part of get-usage.ps1)

    INPUTS:
    $fl          feature list ($l -split ";")
    $f           feature hash
    $u           users hash

    $VARIABLES:
    $f_name      feature name
    $f_used      feature used licenses
    $f_tot       feature total licenses
    $f_den       feature licenses denied
    $f_users     feature users array

    $un         user name

#>

    # loop to find unique features and users
    (1..( $fl.length-1 )) | foreach {

        ($f_name, $f_used, $f_tot, $f_den, $f_users) = GetFeatureData $fl $_

        # unique features
        if ( -not ($f.ContainsKey( $f_name )) ) {

```

```

        $f[ $f_name ] = @{}
    }

    # unique users
    $f_users | foreach {

        $un = $_
        # remove IP address (NOTE 1)
        $un = (($un -split " ")[0])

        if ( -not ($u.ContainsKey( $un )) ) {

            $u[ $un ] = 1

        }

        if ( -not ($f[ $f_name ].ContainsKey( $un )) ) {

            $f[ $f_name ][ $un ] = 1

        }

    }

} # end loop features
}

#=====MAIN=====
<#
    Entry Point

    INPUTS:
    $logfile           path to log file
    $usagefile         license usage file

    VARIABLES:
    $f                 feature hash
    $u                 users hash
    $d                 license usage data

    $l                 line data
    $fl                feature list (separated by ";")

    $f_name            feature name
    $f_used            feature licenses used
    $f_tot             feature total licenses
    $f_users           feature users array

    $un               user name
    $f_sorted          sorted features
    $u_sorted          sorted users

    $ststamp           start time stamp string, 14 characters (e.g. 20220519102322)
    $etstamp           end time stamp string, 14 characters (e.g. 20220519102322)
    $hd_dt             header, time stamp

    $fnum              feature name index
    $fn                feature name
    $unum              user name index

    $actm              activity matrix
    $actm_cnt          activity sample count

    $actm_fpu          activity matrix feature % usage of samples
    $p_used            percentage of samples licenses used

    $hold              temporary hash for unique users

#>

write-host "Starting License Activity Extraction..."

$f = @{}
$u = @{}

# get license usage data
$d = get-content $logfile -encoding ascii

# loop to find unique features and users
$d | foreach {

    $l = $_

    # get features
    $fl = $l -split ";"

    GetUniqueFeaturesUsers $fl $f $u

} # end loop data

# sort features
$f.Remove("TimeCheck")

```

```

[string[]] $f_sorted = $f.keys | sort

# sort users
$u.Remove("Powershell")
[string[]] $u_sorted = $u.keys | sort

# start and end date
$ststamp = $d[0].Substring(0,14)
$etstamp = $d[ ( $d.count - 1 ) ].Substring(0,14)

$hd_dt = ("START,{0}`nEND,{1}" -f $ststamp, $etstamp)

# Initial ACTIVITY MATRIX
$sactm = [Int[],]:new( $u_sorted.Count, $f_sorted.Count)
$sactm_cnt = $d.count
$sactm_fpu = [Double[],]:new( $f_sorted.Count )

# loop to find features and users activity
$d | foreach {

    $l = $_
    #write-host "line:",$l

    # get features
    $fl = $l -split ","

    # loop to find features and users
    (1..( $fl.length-1 )) | foreach {

        ($f_name, $f_used, $f_tot, $f_den, $f_users) = GetFeatureData $fl $_
        $findx = $f_sorted.IndexOf( $f_name )

        if ($findx -ge 0) { # -1 for "TimeCheck"

            $p_used = $f_used/$f_tot
            if ($p_used -gt $sactm_fpu[ $findx ]) {
                $sactm_fpu[ $findx ] = $p_used
            }

            $shold = @{}
            $f_users | foreach {

                $sun = $_
                # remove IP address (NOTE 1)
                $sun = (( $sun -split " " ) [0])

                if ( -not ($shold.ContainsKey( $sun )) ) {

                    # unique user
                    $shold[ $sun ] = 1
                    $uindx = $u_sorted.IndexOf( $sun )
                    $sactm[ $uindx, $findx ] = $sactm[ $uindx, $findx ] + 1

                }

            } # end loop users

        } # end loop feature index check

    } # end loop features

} # end loop data

===ACTIVITY MATRIX
$(

    # start and end date
    $hd_dt

    # samples
    ("SAMPLES,{0}" -f $sactm_cnt)
    ("{0}" -f "")

    $a1 = ","
    $a2 = ","
    # output feature columns
    (0..( $f_sorted.length - 1 )) | foreach {

        $fnum = $_
        $fn = $f_sorted[ $fnum ]
        $a1 = $a1 + ("F{0,3:D3}" -f $fnum)
        $a2 = $a2 + (",{0}" -f $fn)

    }
    ("{0}" -f $a1)
    ("{0}" -f $a2)

    $a3 = ""
    # output user rows
    (0..( $u_sorted.count-1 )) | foreach {

        $sunum = $_
        $a3 = ("U{0,3:D3},{1}" -f $sunum, $u_sorted[ $sunum ])

        (0..( $f_sorted.count-1 )) | foreach {

```

406

```
TART, 20220606064209  
END, 20220928084326  
SAMPLES, 4454
```



```
, , F000, F001, F002, F003, F004, F005, F006, F007, F008, F009, F010, F011, F012, F013, F014, F015, F016, F017, F018, F019, F020, F021, F022, F023, F024, F025  
,, GlobalZoneAM Version: 21.0 Vendor: ALTAIR, GlobalZoneAM Version: 22.0 Vendor: ALTAIR, HWHyperGraph Version: 21.0 Vendor:  
ALTAIR, HWHyperGraph Version: 22.0 Vendor: ALTAIR, HWHyperMesh Version: 21.0 Vendor: ALTAIR, HWHyperMesh Version: 22.0 Vendor:  
ALTAIR, HWHyperStudy Version: 21.0 Vendor: ALTAIR, HWHyperView Version: 21.0 Vendor: ALTAIR, HWHyperView Version: 22.0 Vendor:  
ALTAIR, HWInspire Version: 22.0 Vendor: ALTAIR, HWInspireCastGUI Version: 22.0 Vendor: ALTAIR, HWInspireManufacturingSolver  
Version: 22.0 Vendor: ALTAIR, HWInspireOptistructSolver Version: 22.0 Vendor: ALTAIR, HWLauncher Version: 22.0 Vendor:  
ALTAIR, HWMotionSolve Version: 21.0 Vendor: ALTAIR, HWMotionSolve Version: 22.0 Vendor: ALTAIR, HWMotionEvent Version: 21.0  
Vendor: ALTAIR, HWMotionEvent Version: 22.0 Vendor: ALTAIR, HWOptiStructFEA Version: 21.0 Vendor: ALTAIR, HWOptiStructFEA  
Version: 22.0 Vendor: ALTAIR, HWPDataManager Version: 21.0 Vendor: ALTAIR, HWPDataManager Version: 22.0 Vendor:  
ALTAIR, HWTextView Version: 21.0 Vendor: ALTAIR, HWTextView Version: 22.0 Vendor: ALTAIR, HyperWorks Version: 21.0 Vendor:  
ALTAIR, HyperWorks Version: 22.0 Vendor: ALTAIR  
U000,N012@pc1, 0.0161652447238437, 0.00830713964975303, 0, 0, 0.0161652447238437, 0.00830713964975303, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
0, 0, 0.0161652447238437, 0.00830713964975303  
U001,N001@pc2, 0.0078581050740907, 0, 0, 0.0078581050740907, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.007858105074090  
...
```

TN 23-0516-01

ABAQUS/STANDARD TO ABAQUS/EXPLICIT CONVERSION ISSUES

May 16, 2023

W. Bylsma

U.S. Army Combat Capabilities Development Command (DEVCOM)
Ground Vehicle Systems Center (GVSC)
Detroit Arsenal
Modeling, Simulation & Software
ATTN: FCDD-GVS-MSP/MS 157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

How do you convert an Abaqus/Standard finite element model to Abaqus/Explicit?

PROBLEM SOLUTION:

First, note the difference between how Abaqus/Standard (Implicit) [1] and Abaqus/Explicit [2] solve equations as outlined in Figure 1. Abaqus/Standard assumes things will come to some equilibrium---at which point the solution x depends only on the inverse of the stiffness matrix ($\{x\}=[K]^{-1}\{f\}$), so mass is not important for the final displacement solution). Abaqus/Explicit does not make this assumption and solves for $\{x''\}$ which requires the inverse of the mass matrix ($x''=[M]^{-1}(\{f\} - \text{etc})$). The variable $\{x''\}$ is solved at each time step and then double integrated to get the displacement $\{x\}$. Details can be found in [3].

Implicit v/s Explicit

Overview

► What "Implicit & Explicit" terms indicate ?

If can't Discriminate – Implicit

$$x^y = e^{xy} + x$$

Discriminate – Explicit

$$x = e^{xy}, \ln x = xy, y = \frac{\ln x}{x}$$

► What to do in CAE ?

$$[M]\{x''\} + [C]\{x'\} + [K]\{x\} = \{f\}$$

$$[K]\{x\} = \{f\} - ([M]\{x''\} + [C]\{x'\})$$

$$[K]^{-1}[K]\{x\} = [K]^{-1}(\{f\} - ([M]\{x''\} + [C]\{x'\}))$$

$$\{x\} = [K]^{-1}(\{f\} - ([M]\{x''\} + [C]\{x'\}))$$

$$[M]\{x''\} + [C]\{x'\} + [K]\{x\} = \{f\}$$

$$[M]\{x''\} = \{f\} - ([C]\{x'\} + [K]\{x\})$$

$$[M]^{-1}[M]\{x''\} = [M]^{-1}(\{f\} - ([C]\{x'\} + [K]\{x\}))$$

$$\{x''\} = [M]^{-1}(\{f\} - ([C]\{x'\} + [K]\{x\}))$$

Figure 1 – Standard (Implicit) vs. Explicit Solution Methods[3]

Considerations when changing from Abaqus Standard to Abaqus Explicit are outlined below. These are based on the container example shown in Figure 2 for stacking stress analysis and flatwise drop tests onto a surface.



Figure 2 – Similar container used for Abaqus/Standard Stacking Stress Analysis (<https://www.govdeals.com/asset/519/4964>)

DOUBLE PRECISION

For more accurate results include the double option when running Abaqus/Explicit (e.g. “`abaqus job=[.inp file name] cpus=xx double=explicit`”)

ELEMENTS

Check to make sure the element types are updated correctly.

Table 1 – Abaqus/Explicit Element Types

Type	1 st order	2 nd order
SHELL		
Tri	S3R	SC6R
Quad	S4R	SC8R
SOLID		
Tet	C3D4	C3D10M

Second order elements are more accurate but add computational time. Generally, first order elements are used unless more accurate solutions are required. Tetra meshes create larger sized models than quad meshes, but are sometimes required due to complex geometry. Ideally keep mesh element sizes to greater than 10 mm.

STEPS

Import or create explicit steps. Leave off “FIXED TIME INCREMENTATION” if you want variable automatic time incrementation or use “DIRECT USER CONTROL” to specify a fixed time increment.

Standard

```
*STEP, INC =      1000, NLGEOM = YES
vert-4.5g
*STATIC
, , , 1.0
```

Explicit

```
*STEP, NLGEOM = YES
dyn-1g
*DYNAMIC, EXPLICIT, FIXED TIME INCREMENTATION
, 0.3
```

GROUND

If the static fixed boundary conditions are replaced by ground or another object that will be contacted, then create or import that solid/surface. If using shells for flat ground, adjust the mid-position considering the shell thickness (half on either side) for appropriate contact (see Figure 3). For rigid ground see RIGID BODIES.

RIGID BODIES

If ground or another object is being used for rigid contact, make it a rigid body and fix the reference node degrees of freedom (DOF, see BOUNDARY CONDITIONS)

Explicit

```
*RIGID BODY, REF NODE = 22843718, ELSET = ground, POSITION = INPUT
```

PLASTICITY

Convert materials to linear Elastic and Plastic regions if Deformation Plasticity is used. The plasticity curve starts at zero strain (subtract all absolute values from the yield point strain, use at least 3 points).

Standard

```
*MATERIAL, NAME=Aluminum1_6061-T6_plastic
*DENSITY
2.7000E-09,0.0
*DEFORMATION PLASTICITY
68900.0 , 0.33 , 276.0 , 38.2422 , 0.002 , 0.0
```

Explicit

```
*MATERIAL, NAME=Aluminum1_6061-T6_plastic
*DENSITY
2.7000E-09,0.0
*ELASTIC, TYPE = ISOTROPIC
68900.0 ,0.33 ,0.0
*PLASTIC
276.0 ,0.0 ,0.0
286.0 ,0.0059459 ,0.0
310.0 ,0.1684935 ,0.0
```

CONTACT

Create contact pairs between contact surfaces (e.g. object and ground). Note that the contact pair approach uses the kinematic formulation by default, or you can specify the penalty method as below, whereas the General Contact uses the penalty approach. Issues with self-contact and shell thickness reduction were encountered when trying to use General Contact.

Explicit

```
*CONTACT PAIR, INTERACTION = surf_inter, MECHANICAL CONSTRAINT = PENALTY
block_bot_L1,gnd_block_L1
*CONTACT PAIR, INTERACTION = surf_inter, MECHANICAL CONSTRAINT = PENALTY
block_bot_L2,gnd_block_L2
*CONTACT PAIR, INTERACTION = surf_inter, MECHANICAL CONSTRAINT = PENALTY
block_bot_L3,gnd_block_L3
*CONTACT PAIR, INTERACTION = surf_inter, MECHANICAL CONSTRAINT = PENALTY
block_bot_R1,gnd_block_R1
*CONTACT PAIR, INTERACTION = surf_inter, MECHANICAL CONSTRAINT = PENALTY
block_bot_R2,gnd_block_R2
*CONTACT PAIR, INTERACTION = surf_inter, MECHANICAL CONSTRAINT = PENALTY
block_bot_R3,gnd_block_R3
```

SURFACE INTERACTION

For contacts, a basic surface definition is shown below---many other variations are possible.

Explicit

```
*SURFACE INTERACTION, NAME = surf_inter
*SURFACE BEHAVIOR, PRESSURE-OVERCLOSURE = HARD
*FRICTION
0.3      , 0.0      ,0.0      ,0.0
```

BOUNDARY CONDITIONS

For fixed objects, e.g. ground, constrain the appropriate boundaries or for rigid bodies their reference nodes contained in an NSET (all DOF's 1-6).

Standard

```
*BOUNDARY
19157918,1,3,
...
22769122,1,3,
```

Explicit

```
*BOUNDARY
RF_nodes_gnd,1,6
```

INITIAL CONDITIONS

For a drop test assuming a 304.8 mm (12 inch) drop height, from [4], the initial velocity at impact will be

$$v = \sqrt{2gd} = \sqrt{2 \left(9807 \frac{mm}{s^2} \right) 304.8 mm} = 2,445 \frac{mm}{s}$$

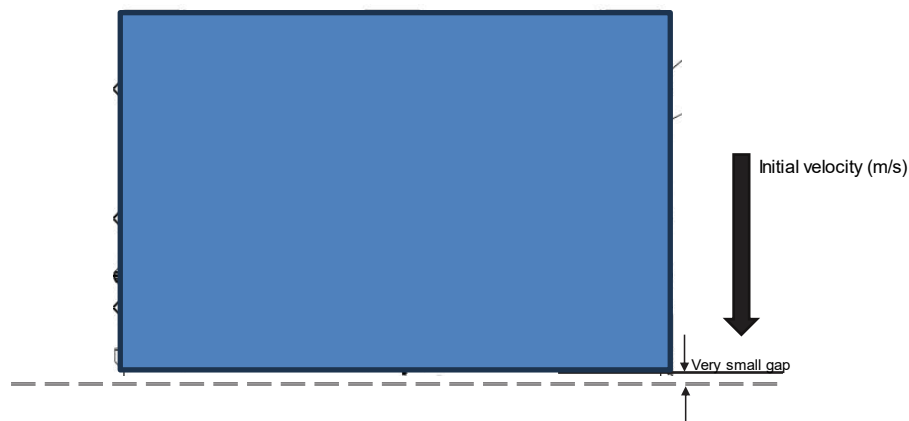


Figure 3 – Flatwise Drop Test

Select and include all nodes in an NSET that have the initial velocity.

Explicit

```
*INITIAL CONDITIONS, TYPE = VELOCITY
init_vel_2450,3,-2450.0
```

BULK VISCOSITY

Bulk viscosity introduces damping associated with volumetric straining. Its purpose is to improve the modeling of high-speed dynamic events (see [2e]).

Explicit

```
*BULK VISCOSITY
*** defaults are 0.06 ,1.2
```

MASS SCALING

Include mass scaling to reduce the required time step (see [2c]). The percent change in mass should be approximately less than 2% of the original mass (change DT to adjust). Removing unnecessary parts with small elements can also help. Variable mass scaling is also possible (see [2c]).

Explicit

```
*FIXED MASS SCALING, DT = 1e-06, TYPE = BELOW MIN
```

OUTPUT FILTERING

In general, the ratio of the filter cut-off frequency, F_c , to the sampling frequency, F_s , should be greater than $1e-3$. This is not always feasible if output space is limited. One has to balance this with the accuracy of the output (e.g., too much high frequency is taken out). The sampling frequency, F_s , must be at least 2x the filter cut-off frequency to avoid aliasing and ideally about 10x. Abaqus's default anti-aliasing filter uses 6x. Here we will use 5x. With a 4-pole Butterworth filter cut-off at 200 Hz

$$\min \text{Output time interval} = \frac{1}{5F_c} = \frac{1}{5(200)} = 0.001 \text{ sec}$$

This will be used in the output request TIME INTERVAL (see OUTPUT). The filter cut-off frequency and minimum output time interval should be recalculated for each model.

Explicit

```
*FILTER , NAME = bw4, TYPE = BUTTERWORTH
200.0 , 4
```

OUTPUT

Reduce the number of output requests, especially FIELD requests, to reduce the output database (.odb) size. Include a filter and corresponding time interval (see OUTPUT FILTERING) on all output requests to avoid anti-aliasing.

Standard

```
*NODE PRINT, NSET = RF_nodes, SUMMARY = YES, TOTALS = YES
*OUTPUT, FIELD
*NODE OUTPUT
U,RF
*ELEMENT OUTPUT
S,CTF,SF,SM
E,EE,IE,PE
*CONTACT OUTPUT
CFORCE,CSTRESS
*PRINT, CONTACT = YES, SOLVE = YES
```

Explicit

```
*OUTPUT, FIELD, FILTER = bw4, TIME INTERVAL = 0.001
*NODE OUTPUT
U
*ELEMENT OUTPUT, DIRECTIONS = YES
PE
*OUTPUT, HISTORY, FILTER = bw4, TIME INTERVAL = 0.001
*NODE OUTPUT, NSET = responses
U,V,A
*PRINT, MASS = YES, ALLKE = YES, CRITICAL ELEMENT = YES, DMASS = YES, ETOTAL = YES
*END STEP
```

Explicit (with comments)

```
*OUTPUT, FIELD, FILTER = bw4, TIME INTERVAL = 0.001
*NODE OUTPUT
U
*ELEMENT OUTPUT, DIRECTIONS = YES
***S,CTF,SF,SM
***E,EE,IE,PE
PE
***CONTACT OUTPUT
***CFORCE,CSTRESS
*OUTPUT, HISTORY, FILTER = bw4, TIME INTERVAL = 0.001
*NODE OUTPUT, NSET = responses
U,V,A
***OUTPUT, HISTORY, VARIABLE = PRESELECT, FILTER = bw4, TIME INTERVAL = 0.001
***INCREMENTATION OUTPUT, VARIABLE = ALL
***DMASS
*PRINT, MASS = YES, ALLKE = YES, CRITICAL ELEMENT = YES, DMASS = YES, ETOTAL = YES
*END STEP
```

Further considerations may be necessary depending on the specific model.

REFERENCE

Below references require account at <https://help.3ds.com>

- [1] Dassault Systems SIMULIA Abaqus/Standard (<https://www.3ds.com/products-services/simulia/products/abaqus/abaqusstandard/>)
- [2] Dassault Systems SIMULIA Abaqus/Explicit (<https://www.3ds.com/products-services/simulia/products/abaqus/abaqusexplicit/>)
- [2a] STEPS (https://help.3ds.com/2022/English/DSSIMULIA_Established/SIMACAEKEYRefMap/simakey-r-dynamic.htm?contextscope=all)
- [2b] CONTACT PAIR (https://help.3ds.com/2022/English/DSSIMULIA_Established/SIMACAEKEYRefMap/simakey-r-contactpair.htm?contextscope=all&id=3aa9e54191404ba98280b0f90b5842fd)
- [2c] MASS SCALING (https://help.3ds.com/2022/English/DSSIMULIA_Established/SIMACAEANLRefMap/simaanl-c-massscaling.htm?contextscope=all)
- [2d] FILTERING OUTPUT (https://help.3ds.com/2022/English/DSSIMULIA_Established/SIMACAEOUTRefMap/simaout-c-dboutput-filter.htm?contextscope=all&id=0d2e45e45fe744379f180d396d2875f6)

- [2e] BULK VISCOSITY (https://help.3ds.com/2022/english/dssimulia_established/SIMACAEANLRefMap/simaanl-c-expdynamic.htm?contextscope=all&id=49240cdba30845de8229253f9dd43787#simaanl-c-expdynamic-bulkviscosity)

Above references require account at <https://help.3ds.com>

- [3] Implicit and Explicit Analysis in FEA, Part - 01 (Differences & Application), SimTech05. (<https://www.youtube.com/watch?v=UMer6RkLpww>)
- [4] Leonhardt. Acceleration Levels of Dropped Objects (Technical Paper 321), Endevco (https://endevco.com/contentStore/mktgContent/endevco/dlm_uploads/2019/02/TP321.pdf)
- [5] Revisions and Exceptions to SAE ARP 1967B (see SAE ARP1967 Rev B – Aerospace Recommended Practice – Containers, Shipping and Storage, Reusable at <https://www.sae.org/standards/content/arp1967b/>) (<https://sam.gov/opp/08a491daf56f46df9ef0d75eddcac0/view>)
- [6] How to Calculate the Velocity of a Pendulum Using the Law of Conservation of Energy | Physics | Study.com (<https://study.com/skill/learn/how-to-calculate-the-velocity-of-a-pendulum-using-the-law-of-conservation-of-energy-explanation.html>)

APPENDIX

A.1 Pendulum Configuration

APPENDIX A.1– Pendulum Configuration

Equating potential and kinetic energy as in [6]

$$mg\Delta h = \frac{1}{2}mv^2$$

and rearranging then gives velocity as in [5]

$$v = \sqrt{2g\Delta h}$$

TN 23-0516-02

ABAQUS/EXPLICIT HISTORY OUTPUT EXTRACTION USING PYTHON

May 16, 2023

W. Bylsma

U.S. Army Combat Capabilities Development Command (DEVCOM)
Ground Vehicle Systems Center (GVSC)
Detroit Arsenal
Modeling, Simulation & Software
ATTN: FCDD-GVS-MSP/MS 157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

How do you extract history output data from the Abaqus/Explicit output database?

PROBLEM SOLUTION:

Abaqus/Explicit is generally run on a High Performance Computing (HPC) system due to the large size of output data generated (> 6 GB). Transferring this data can take significant amounts of time (> 1 hour). However, if post-processing software is not available on the HPC system, or cannot be used efficiently, it may be necessary. A possible solution is to extract history output into an ASCII format and transfer only the output results selected.

The following bash script “odb” can be executed to identify data in the “.odb” file using “odbreport” (See Appendix A.1 for detailed output). From the “odb” output you can identify desired history outputs and modify the following variable values in the python script “xy_csv.py” (see Appendix A.2):

Table 1 – Variable Descriptions and Values

Description	Variable	Value (example)
CSV Output File Name (.csv)	fo_name	'xy'
ODB Database Name	f_odb	'nbd_d02'
Step Name	step	'Step-1'
History Region Name	regi	'Node PART-1-1.19157911'
History Output Name List	vari	['A3_BW4', 'U3_BW4']
Scale Factor for Acceleration	scaleA3	9807.0
Scale Factor Variable Name	scaleNM	vari[0] or 'A3_BW4'

obd

```
#!/usr/bin/bash

# odb - odbreport output
#
# . odb [.odb]
# . odb nbd_02.odb
#

cmd="/p/app/abaqus/2017/Commands/abq2017"
file=$1
$cmd odbreport odb=$file

# $cmd odbreport odb=$file histregion='Node PART-1-1.19157911' history='A3_BW4'
# $cmd odbreport odb=$file histregion='Node PART-1-1.19157911' history='U3_BW4'
```

Once the “xy_csv.py” script variables have been set, execute the bash script “eoho” to extract the history output data to a “.csv” file. You can also capture the standard output using “. eoho > [file name]” (e.g. “. eoho > xy.out”).

eoho

```
#!/usr/bin/bash

# eoho - extract odb history output
#
# . eoho
#

cmd="/p/app/abaqus/2017/Commands/abq2017"
file="xy_csv.py"
$cmd script=$file
```

Example xy.csv

```
time,A3_BW4,U3_BW4
0.0,-0.9999964151881309,0.0
0.0010009966790676117,-0.9999911375484348,-0.028988517820835114
0.00200099335052073,-1.0029831609437136,-0.5045283436775208
0.0030009900219738483,-0.9389412935275824,-2.026224136352539
.
.
.
0.2980000078678131,4.377792170133578,-24.361553192138672
0.2990000247955322,4.453320969715509,-24.65334701538086
0.30000001192092896,4.290335108213521,-24.901695251464844
```

Example xy.out

```
Extracting nbd_d02.odb ==> xy.csv

STEPS:
Step-1

HIST REGIONS:
Node PART-1-1.19157911
.
.
.
time A3 BW4 U3_BW4
0.000000 -0.999996 0.000000
0.001001 -0.999991 -0.028989
0.002001 -1.002983 -0.504528
.
.
.
0.298000 4.377792 -24.361553
0.299000 4.453321 -24.653347
0.300000 4.290335 -24.901695
```

Note that with slight modification the “xy_csv.py” script can be run from Abaqus/CAE (execfile('xy_csv.py')). Abaqus/CAE currently uses Python 2.7. Additional conditions can be added to “xy_csv.py”, similar to the

scaling of the acceleration to units of g's, for other modifications to the history output variables. Care must be taken to properly initialize the variables used and attention to the units is essential.

Example History Output Variable Conditions

```
.  
.   
.   
scaleA3 = 9807.0 # mm/s^2 to g's  
scaleNM = vari[0]  
.   
.   
.   
if (vari[j] == scaleNM):  
    pass  
    # scale acceleration mm/s^2 to g's  
    r.append(ii[1] / scaleA3)  
else:  
    r.append(ii[1])
```

The “xy_csv.py” script uses the “csv” python module for output to the “.csv” format.

REFERENCE

- [1] Dassault Systems SIMULIA Abaqus/Explicit (<https://www.3ds.com/products-services/simulia/products/abaqus/abaqusexplicit/>)
- [2] Python (www.python.org)

APPENDIX

A.1 odb Output

A.2 xy_csv.py

APPENDIX A.1– odb Output

ODB Report: -odb nbd_d02.odb

General ODB information

name : nbd_d02.odb
 analysisTitle :
 description : DDB object
 path : /p/work1/user/drop_fl_2017/nbd_d02/nbd_d02.odb

Job Data

name : nbd_d02.odb
 analysisCode : ABAQUS_EXPLICIT
 precision : DOUBLE_PRECISION
 version : Abaqus/Explicit 3DEXPERIENCE R2017x
 creationTime : Thu May 11 09:08:04 EDT 2023
 modificationTime : Thu May 11 09:08:55 EDT 2023
 machineName :

Section Categories

number of categories = 5
 Section category name 'Mass'
 description : SectionCategory for element type MASS in part PART-1-1

 Section category name 'connector < elset = L3 >'
 description : SectionCategory for element type CONN3D2 in part PART-1-1

 Section category name 'shell < 5 section points > < ALUMINUM_L6061-T6_PLASTIC >'
 description : SectionCategory for element type S4R in part PART-1-1

 Section category name 'shell < 5 section points > < STEEL_A36_PLASTIC >'
 description : SectionCategory for element type S4R in part PART-1-1

 Section category name 'solid < ALUMINUM_L6061-T6_PLASTIC >'
 description : SectionCategory for element type C3D4 in part PART-1-1

 Root assembly name 'Assembly-1'

number of instances = 1

Part instance 'PART-1-1'

embedded space : THREE_D
 type : DEFORMABLE_BODY

number of nodes = 675655

number of element classes = 124

Class	Number of elements
S4R	7869
S3R	287
C3D4	17027
S4R	11911
S3R	18
S4R	20681
S3R	726
C3D4	1158
S4R	4799
S3R	5
CONN3D2	1
S4R	21534
S3R	848
S4R	3743
S3R	2
S4R	15263
S3R	392
C3D4	8116
S4R	2829
S3R	8
S4R	20765
S3R	761

C3D4	1016
S4R	4756
S3R	8
S4R	7720
S3R	273
C3D4	17229
S4R	11599
S3R	23
CONN3D2	1
S4R	21578
S3R	803
S4R	3768
S3R	12
S4R	15297
S3R	371
C3D4	8099
S4R	2815
S3R	12
S4R	22208
S3R	215
S4R	7608
S3R	181
C3D4	17713
S4R	22245
S3R	177
S4R	15657
S3R	96
C3D4	8071
S4R	22220
S3R	202
S4R	8011
S3R	193
C3D4	17212
S4R	22250
S3R	172
S4R	15355
S3R	84
C3D4	8450
CONN3D2	1
S4R	15498
S3R	578
C3D4	7646
S4R	2664
S3R	6
CONN3D2	1
S4R	21550
S3R	832
S4R	3696
S3R	2
S4R	7994
S3R	385
C3D4	16759
S4R	11901
S3R	8
S4R	20389
S3R	833
C3D4	1386
S4R	4777
S3R	6
CONN3D2	1
S4R	15547
S3R	790
C3D4	7339
S4R	2449
S3R	11
CONN3D2	1
S4R	21605
S3R	774
S4R	3855
S3R	4
S4R	20283

S3R	798
C3D4	1556
S4R	4768
S3R	4
S4R	8044
S3R	370
C3D4	16719
S4R	11783
S3R	3
S4R	22228
S3R	195
S4R	22101
S3R	324
S4R	15177
S3R	99
C3D4	8648
S4R	8263
S3R	163
C3D4	16942
MASS	1
S4R	22204
S3R	219
S4R	22074
S3R	352
S4R	15404
S3R	95
C3D4	8378
S4R	8379
S3R	143
C3D4	16825
RNODE3D	16

number of elements = 844273

number of node sets = 5

Name	Size
'INIT_VEL_2450'	582868
'RESPONSES'	5
'RF_NODES'	6
'RF_NODES_BOT'	2295
'RF_NODES_GND'	1

number of element sets = 65

Name	Size
'BLOCKING-AL'	206289
'GROUND'	92245
'L1'	1
'L2'	1
'L3'	1
'MASS_FUPP'	1
'MS-4_826MM_A36'	6565
'MS-6_35MM_A36'	43549
'MS-9_52MM_A36'	4893
'MS-12_7MM_A36'	6448
'MS-BOT-3_416MM'	87287
'MS-BOT-4_554MM'	27196
'MS-BOT-4_762MM'	4714
'MS-BOT-6_35MM'	28448
'MS-BOT-9_525MM'	44634
'MS-BOT-12_7MM'	15387
'MS-BOT-50_8MM'	1595
'MS-TOP_3_416MM'	182748
'MS-TOP_4_762MM'	9812
'MS-TOP_6_35MM'	40656
'MS-TOP_9_525MM'	41371
'MS19_05MM_A36'	429
'R1'	1
'R2'	1
'R3'	1
'_BLOCK_BOT_L1_S1'	655
'_BLOCK_BOT_L1_S2'	15

'_BLOCK_BOT_L1_S3'	15
'_BLOCK_BOT_L1_S4'	19
'_BLOCK_BOT_L2_S1'	597
'_BLOCK_BOT_L2_S2'	11
'_BLOCK_BOT_L2_S3'	9
'_BLOCK_BOT_L2_S4'	15
'_BLOCK_BOT_L3_S1'	651
'_BLOCK_BOT_L3_S2'	18
'_BLOCK_BOT_L3_S3'	19
'_BLOCK_BOT_L3_S4'	14
'_BLOCK_BOT_R1_S1'	655
'_BLOCK_BOT_R1_S2'	15
'_BLOCK_BOT_R1_S3'	17
'_BLOCK_BOT_R1_S4'	17
'_BLOCK_BOT_R2_S1'	585
'_BLOCK_BOT_R2_S2'	19
'_BLOCK_BOT_R2_S3'	13
'_BLOCK_BOT_R2_S4'	15
'_BLOCK_BOT_R3_S1'	647
'_BLOCK_BOT_R3_S2'	24
'_BLOCK_BOT_R3_S3'	18
'_BLOCK_BOT_R3_S4'	15
'_BOTPAD_BLOCK_S1'	7872
'_BOTPAD_BLOCK_S2'	131
'_BOTPAD_BLOCK_S3'	69
'_BOTPAD_BLOCK_S4'	68
'_BOTPAD_SNEG'	3746
'_GND_BLOCK_L1_SPOS'	253
'_GND_BLOCK_L2_SPOS'	222
'_GND_BLOCK_L3_SPOS'	256
'_GND_BLOCK_R1_SPOS'	256
'_GND_BLOCK_R2_SPOS'	221
'_GND_BLOCK_R3_SPOS'	256
'_TOPPAD_BLOCK_S1'	3849
'_TOPPAD_BLOCK_S2'	90
'_TOPPAD_BLOCK_S3'	62
'_TOPPAD_BLOCK_S4'	79
'_TOPPAD_SPOS'	1544

number of surfaces = 0

 Assembly instance 'Assembly-1'
 embedded space : THREE_D
 type : DEFORMABLE_BODY

number of nodes = 0

number of element classes = 0

number of elements = 0

number of node sets = 3

Name	Size	
'_CONSTRAINED_NODES_ALL'	13003	
'_REFERENCE_POINT_PART-1-1_22843718'		1
'_WarnNodeConnZeroMassInVel'	12	

number of element sets = 4

Name	Size
'_WarnElemDistorted'	1723
'_WarnElemFacetThickPt63d-Step1'	6
'_WarnElemWarped'	113
'_Whole_Model_Gravity_Elset'	844267

number of surfaces = 17

Name	Size
'_BLOCK_BOT_L1'	704
'_BLOCK_BOT_L2'	632
'_BLOCK_BOT_L3'	702
'_BLOCK_BOT_R1'	704

'BLOCK_BOT_R2'	632
'BLOCK_BOT_R3'	704
'BOTPAD'	3746
'BOTPAD_BLOCK'	8140
'GND_BLOCK_L1'	254
'GND_BLOCK_L2'	223
'GND_BLOCK_L3'	257
'GND_BLOCK_R1'	257
'GND_BLOCK_R2'	222
'GND_BLOCK_R3'	257
'Rigid Reference Node #22843718'	16
'TOPPAD'	1544
'TOPPAD_BLOCK'	4080

```

number of rigid bodies = 1
Rigid Body #0
  reference node(s):
    22843718 from instance 'PART-1-1'

```

Total number of steps = 1

```

Step name 'Step-1'
  description : dyn-lg
  number      : 1
  domain      : TIME
  procedure   : *DYNAMIC, EXPLICIT
  timePeriod  : 0.300000011920929
  totalTime   : 0.0
  previousStep : Initial
  nlgeom      : Yes

```

```

Total number of frames = 301
  Number of field outputs = 3
    Field name  'PE'
    Field name  'U'
    Field name  'UR'

```

```

Number of history regions = 5
History Region 'Node PART-1-1.19157911'
  Number of history outputs = 18
    History Output 'A1_BW4'
    History Output 'A2_BW4'
    History Output 'A3_BW4'
    History Output 'AR1_BW4'
    History Output 'AR2_BW4'
    History Output 'AR3_BW4'
    History Output 'U1_BW4'
    History Output 'U2_BW4'
    History Output 'U3_BW4'
    History Output 'UR1_BW4'
    History Output 'UR2_BW4'
    History Output 'UR3_BW4'
    History Output 'V1_BW4'
    History Output 'V2_BW4'
    History Output 'V3_BW4'
    History Output 'VR1_BW4'
    History Output 'VR2_BW4'
    History Output 'VR3_BW4'
History Region 'Node PART-1-1.22595336'
  Number of history outputs = 18
    History Output 'A1_BW4'
    History Output 'A2_BW4'
    History Output 'A3_BW4'
    History Output 'AR1_BW4'
    History Output 'AR2_BW4'
    History Output 'AR3_BW4'
    History Output 'U1_BW4'
    History Output 'U2_BW4'
    History Output 'U3_BW4'

```

```

History Output 'UR1_BW4'
History Output 'UR2_BW4'
History Output 'UR3_BW4'
History Output 'V1_BW4'
History Output 'V2_BW4'
History Output 'V3_BW4'
History Output 'VR1_BW4'
History Output 'VR2_BW4'
History Output 'VR3_BW4'
History Region 'Node PART-1-1.22596825'
Number of history outputs = 18
History Output 'A1_BW4'
History Output 'A2_BW4'
History Output 'A3_BW4'
History Output 'AR1_BW4'
History Output 'AR2_BW4'
History Output 'AR3_BW4'
History Output 'U1_BW4'
History Output 'U2_BW4'
History Output 'U3_BW4'
History Output 'UR1_BW4'
History Output 'UR2_BW4'
History Output 'UR3_BW4'
History Output 'V1_BW4'
History Output 'V2_BW4'
History Output 'V3_BW4'
History Output 'VR1_BW4'
History Output 'VR2_BW4'
History Output 'VR3_BW4'
History Region 'Node PART-1-1.22600204'
Number of history outputs = 18
History Output 'A1_BW4'
History Output 'A2_BW4'
History Output 'A3_BW4'
History Output 'AR1_BW4'
History Output 'AR2_BW4'
History Output 'AR3_BW4'
History Output 'U1_BW4'
History Output 'U2_BW4'
History Output 'U3_BW4'
History Output 'UR1_BW4'
History Output 'UR2_BW4'
History Output 'UR3_BW4'
History Output 'V1_BW4'
History Output 'V2_BW4'
History Output 'V3_BW4'
History Output 'VR1_BW4'
History Output 'VR2_BW4'
History Output 'VR3_BW4'
History Region 'Node PART-1-1.22601693'
Number of history outputs = 18
History Output 'A1_BW4'
History Output 'A2_BW4'
History Output 'A3_BW4'
History Output 'AR1_BW4'
History Output 'AR2_BW4'
History Output 'AR3_BW4'
History Output 'U1_BW4'
History Output 'U2_BW4'
History Output 'U3_BW4'
History Output 'UR1_BW4'
History Output 'UR2_BW4'
History Output 'UR3_BW4'
History Output 'V1_BW4'
History Output 'V2_BW4'
History Output 'V3_BW4'
History Output 'VR1_BW4'
History Output 'VR2_BW4'
History Output 'VR3_BW4'

```

End of ODB Report.

APPENDIX A.2– xy_csv.py

```

# Python 2.7
# execfile('xy.py') from CAE

from odbAccess import *
from abaqusConstants import *

import csv

#####CHANGE
# output/odb
fo_name = 'xy' # .csv
f_odb = 'nbd_d02' # .odb

# parameters
step = 'Step-1'
regi = 'Node PART-1-1.19157911'
vari = ['A3_BW4','U3_BW4']
scaleA3 = 9807.0 # mm/s^2 to g's
scaleNM = vari[0]
#####END

print "Extracting %s ==> %s\n" % (f_odb+'.odb', fo_name+'.csv')

# from CAE
# ii=session.odbs.keys()[0]

# open odb
odbfile = openOdb(path=f_odb+'.odb')

print "STEPS:"
for i in odbfile.steps.keys():
    print i
print

print "HIST REGIONS:"
for i in odbfile.steps[step].historyRegions.keys():
    print i
print

print "HIST OUTPUTS:"
for i in odbfile.steps[step].historyRegions[regi].historyOutputs.keys():
    print i
print

# output data
odata1=[]

# loop vari
r = ["time"]
for i in range(0,len(vari)):
    r.append(vari[i])
odata1.append(r)

print " ".join(r)

# loop hist data
for i in range(0,len(odbfile.steps[step].historyRegions[regi].historyOutputs[vari[0]].data)):
    ii = odbfile.steps[step].historyRegions[regi].historyOutputs[vari[0]].data[i]
    r = [ ii[0] ]
    for j in range(0,len(vari)):
        ii = odbfile.steps[step].historyRegions[regi].historyOutputs[vari[j]].data[i]
        if (vari[j] == scaleNM):
            pass
            # scale acceleration mm/s^2 to g's
            r.append(ii[1] / scaleA3)
        else:
            r.append(ii[1])

```

```
odata1.append(r)

print " ".join(["%12.6f" %(i) for i in r])

odbfile.close()

# open .out
with open(fo_name+'.csv','w') as outfile:

    csvwriter = csv.writer(outfile)
    csvwriter.writerows(odata1)
# end with outfile
```

TN 23-0531-01

RAMBERG-OSGOOD: FROM NEUBER CORRECTION TO PLASTICITY

May 31, 2023

W. Bylsma

U.S. Army Combat Capabilities Development Command (DEVCOM)
Ground Vehicle Systems Center (GVSC)
Detroit Arsenal
Modeling, Simulation & Software
ATTN: FCDD-GVS-MSP/MS 157
6501 East 11 Mile Road
Warren, Michigan 48397-5000

PROBLEM DESCRIPTION:

How is the Ramberg-Osgood relationship used for Neuber correction?

PROBLEM SOLUTION:

From using the Neuber correction spreadsheet [4] as shown in Figure 1, we are curious as to how the Ramberg-Osgood relationship is used to estimate plasticity correction into linear results.

Neubers Rule is elaborated on in [5-15]. With some assumptions regarding strain energy ([https://en.wikipedia.org/wiki/ Strain_energy](https://en.wikipedia.org/wiki/Strain_energy)) we have

$$\sigma_e \epsilon_e = \sigma_p \epsilon_p [5]$$

$$\sigma_e \epsilon_e = \sigma_e \left(\frac{\sigma_e}{E} \right) \rightarrow \frac{\sigma_e^2}{E} = \sigma_e \epsilon_e [15]$$

where e represents “elastic” and p represents “plastic”. The Ramberg-Osgood relationship [1] is summarized as

The **Ramberg–Osgood equation** was created to describe the non linear relationship between [stress](#) and [strain](#)—that is, the [stress–strain curve](#)—in materials near their [yield points](#). It is especially applicable to metals that *harden* with plastic deformation (see [work hardening](#)), showing a *smooth* elastic-plastic transition. As it is a [phenomenological model](#), checking the fit of the model with actual experimental data for the particular material of interest is essential.

The original NASA source references are from [2,3]. Figure 2 shows the nonlinear stress-strain relationship from [1].

Combining Neubers Rule with the Ramberg-Osgood equation, Figures 3 and 4 show the plasticity correction curves, without and with annotations respectively, for 4130 Steel. Figure 5 shows actual test data from [3].

The variation from the linear elastic yield point for the Ramberg-Osgood relationship is explored in Figures 6 (varying n) and Figure 7 (varying offset). This demonstrates that nonlinear deformation begins before the elastic yield point.

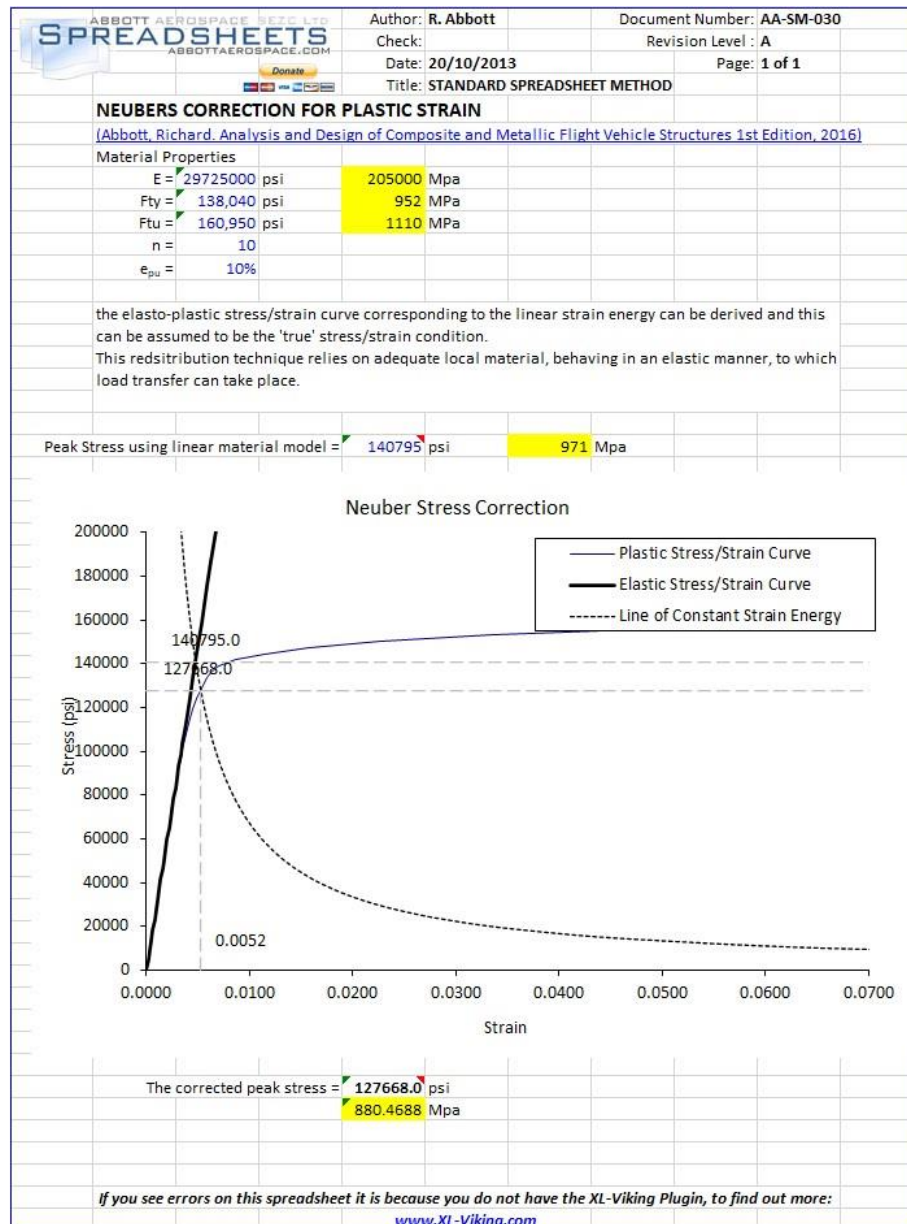


Figure 1 – Neuber Plastic Strain Method Theory and Spreadsheet, Abbott Aerospace Canada Ltd [4]

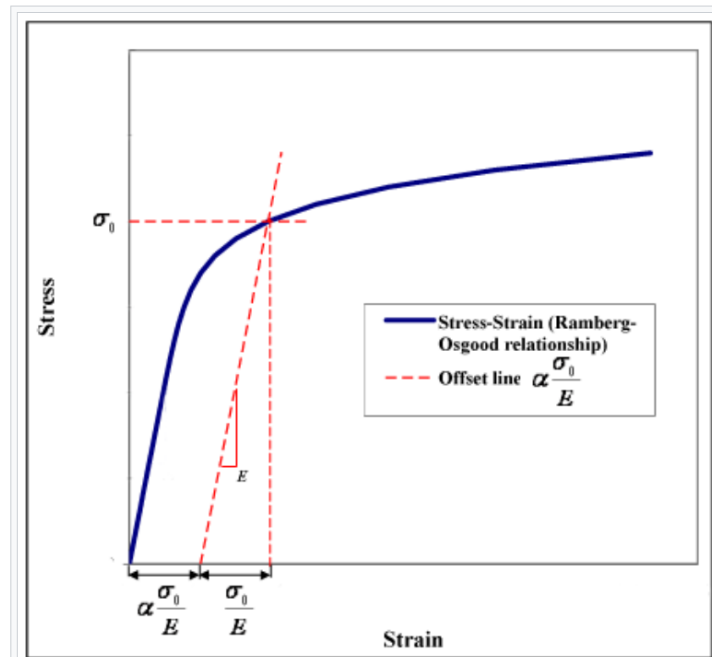


Figure 1: Generic representation of the stress–strain curve by means of the Ramberg–Osgood equation. Strain corresponding to the yield point is the sum of the elastic and plastic components.

Figure 2 – Ramberg-Osgood Stress-Strain relationship [1]

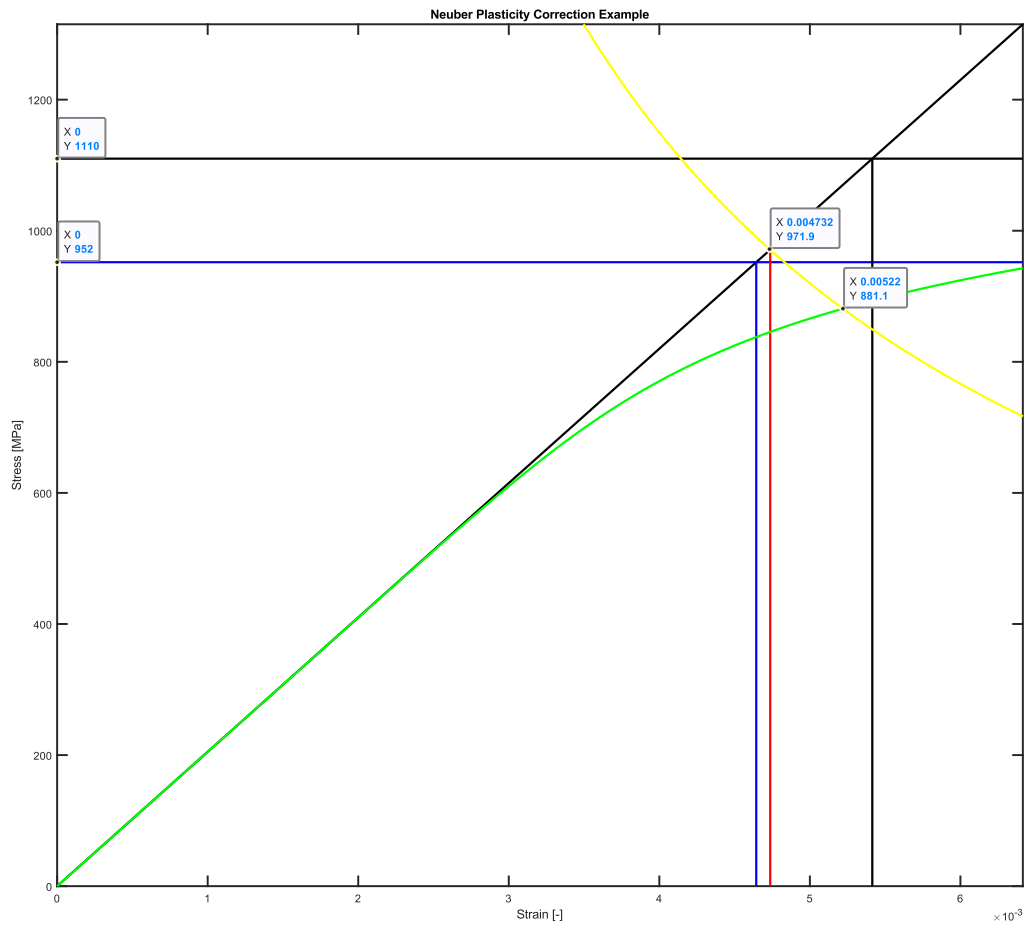


Figure 3 – Neuber Plasticity Correction Example

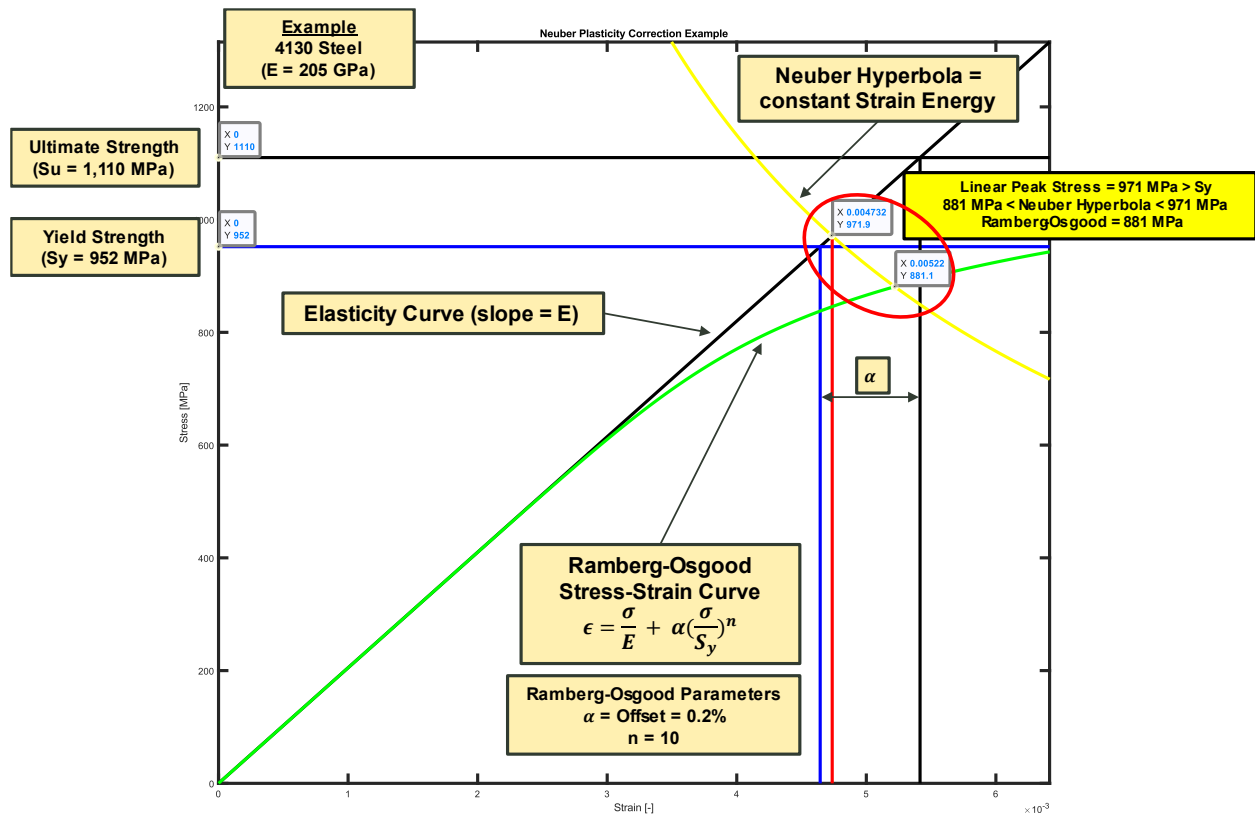


Figure 4 – Neuber Plasticity Correction Example Annotated

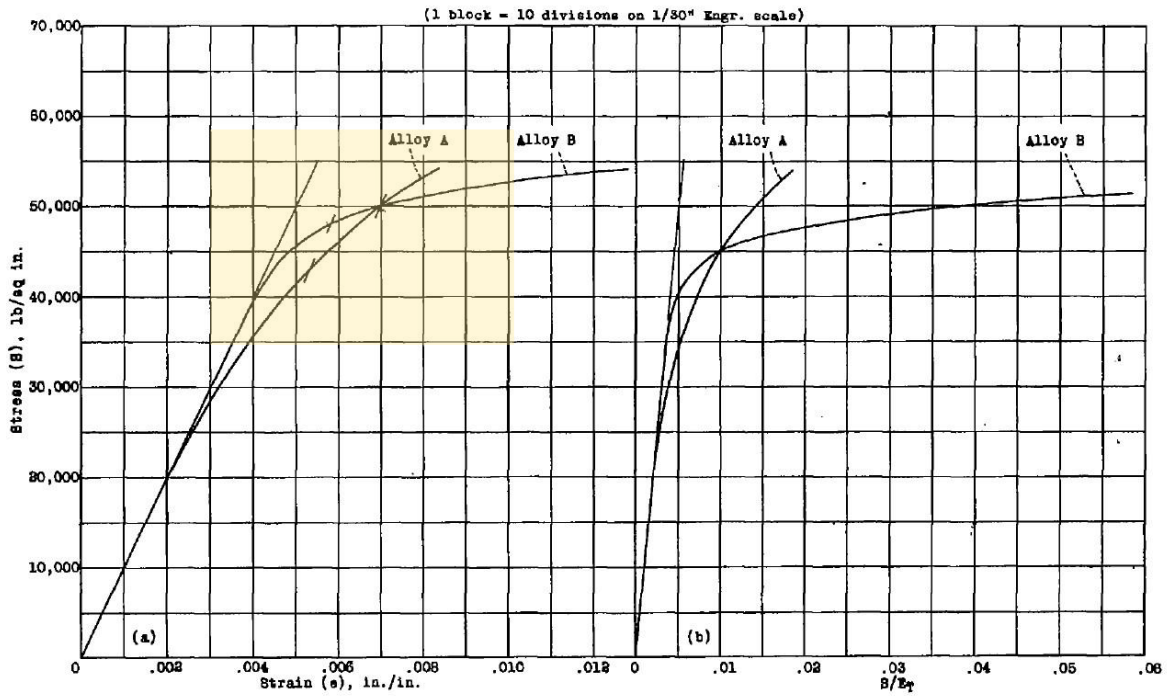


Figure 1.- Stress-strain and stress- S/E_t curves for two alloys having the same modulus and the same yield strength (0.002 offset), but with different stress-strain curves.

WDA Technical Note No. 227

FIG. 1

Figure 5 - Stress-Strain Curve Example [3]

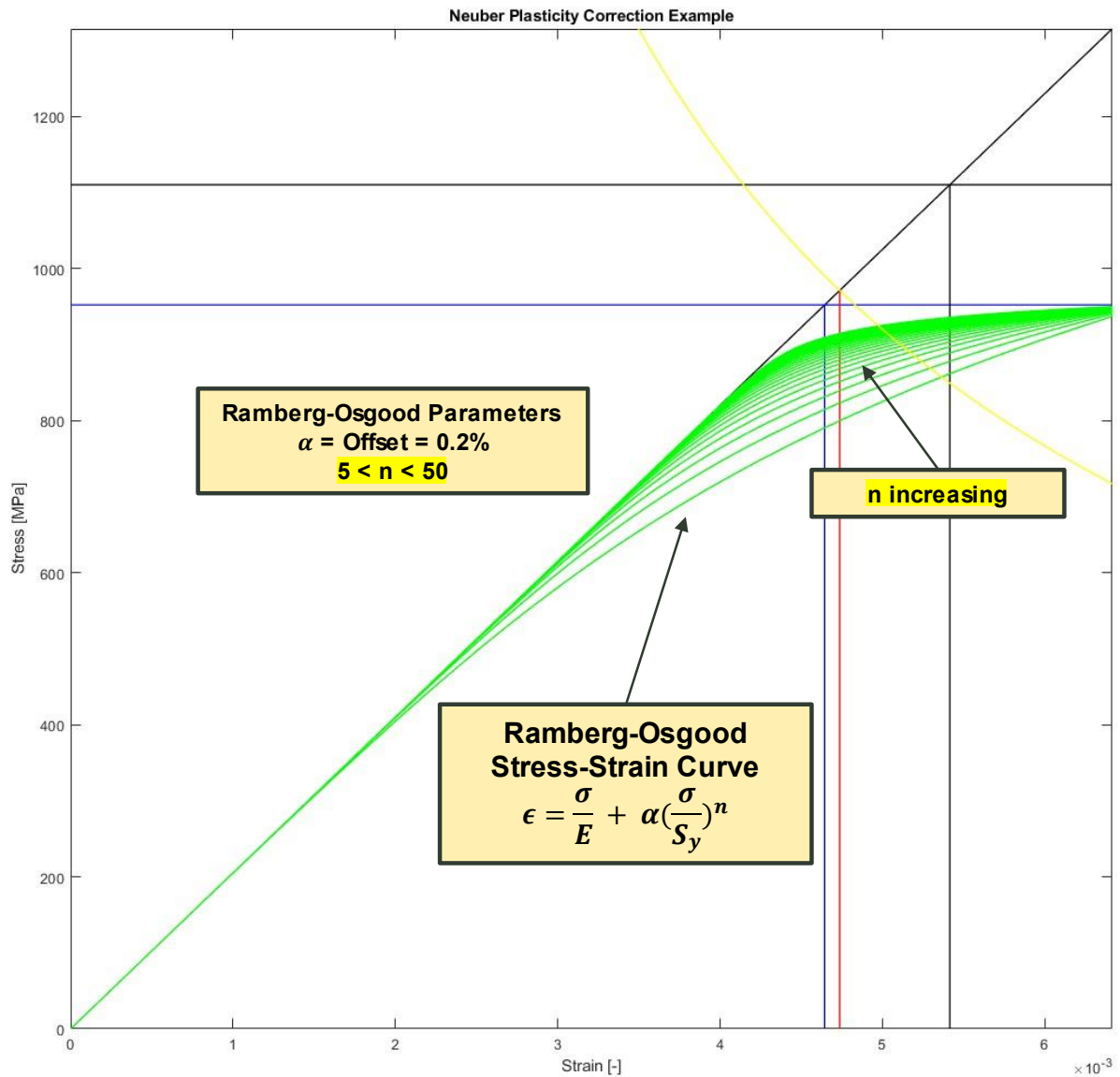


Figure 6 – Ramberg-Osgood Curve (varying n)

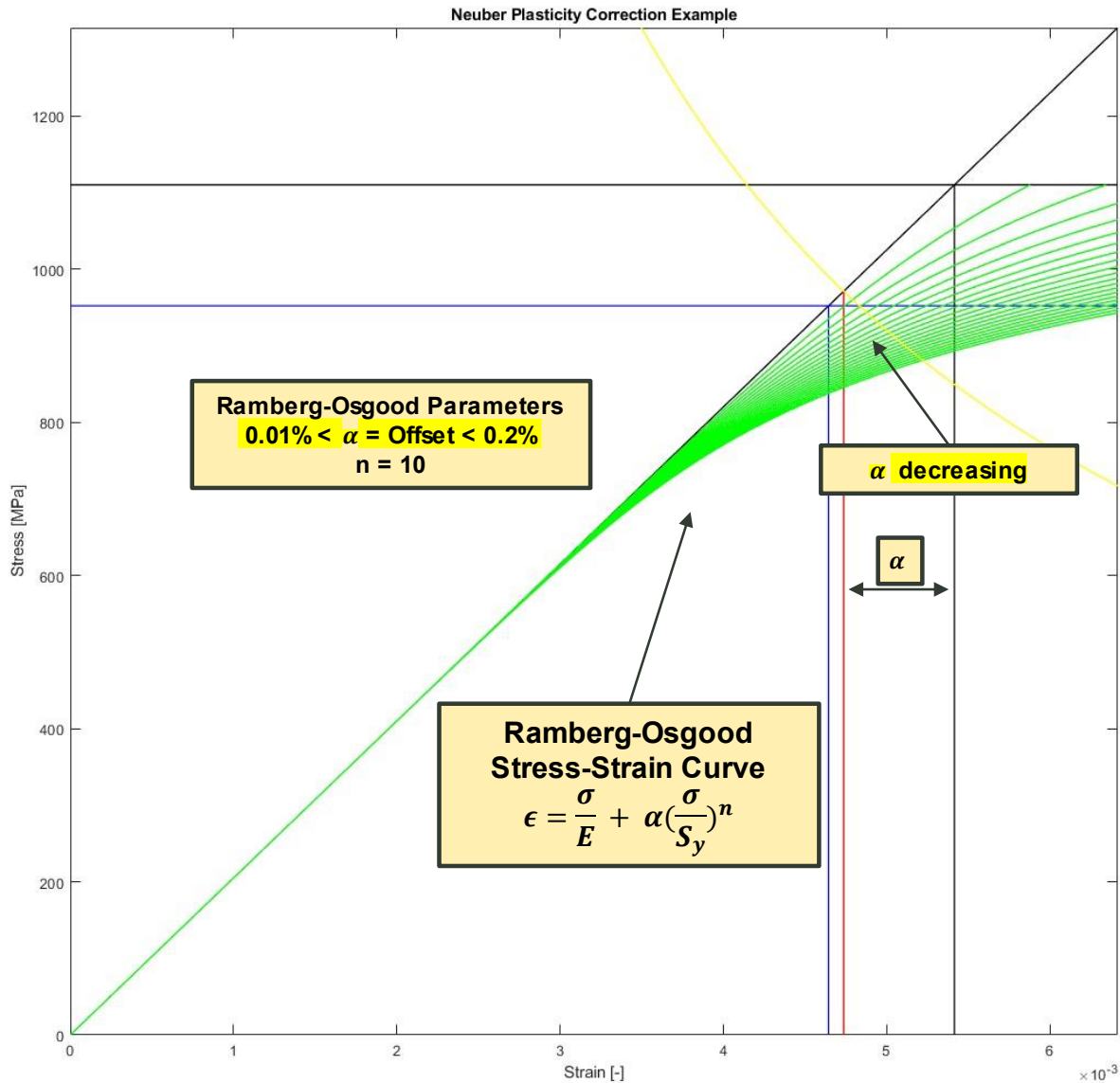


Figure 7 – Ramberg-Osgood Curve (varying offset)

In the example of Figure 1, $n = 10$, but the parameter n can be computed as

$$n = \frac{\log\left(\frac{e_{pu}}{\alpha}\right)}{\log\left(\frac{S_u}{S_y}\right)} [3]$$

where e_{pu} , α , S_u , and S_y represent strain elongation at break, offset, ultimate stress, and yield stress. The offset is assumed to be 0.2% (0.002).

A script (`ramosg.py/ramosg.m`), both in Python [17] and MATLAB [16], was written to perform the Neuber correction for linear results based on the Ramberg-Osgood stress-strain relationship. The scripts are closely related with a few exceptions---Python requires some modules for plotting/computation with function definitions at the beginning while MATLAB requires function definitions at the end. The Python

version ran using JupyterLab in a web browser [18]. An option exists to write out the Ramberg-Osgood curve as described in Appendix A.4.

Parameters for the Ramberg-Osgood curve can also be used to describe material plasticity in Abaqus/Standard [19, 19a] which is based on the Ramberg-Osgood equation.

Table 1 shows the output for both language versions. These are in agreement with the spreadsheet results in Figure 1.

Table 1 – ramosg.py/m Python/MATLAB output

Python	MATLAB
<pre> ===4130 Steel=== Parameters: Offset %= 0.200000 E [MPa]= 205000.000000 Elongation@Brk %= 0.100000 n= 10.000000 Stress [MPa]: Linear Elastic= 971.000000 Yield= 952.000000 Ultimate= 1110.000000 RO Corrected= 881.000000 Strain %: Linear Elastic= 0.473659 RO Yield= 0.664390 RO Ultimate= 1.470195 RO Corrected= 0.521890 </pre>	<pre> ===4130 Steel=== Parameters: Offset %= 0.200000 E [MPa]= 205000.000000 Elongation@Brk %= 0.100000 n= 10.000000 Stress [MPa]: Linear Elastic= 971.000000 Yield= 952.000000 Ultimate= 1110.000000 RO Corrected= 881.000000 Strain %: Linear Elastic= 0.473659 RO Yield= 0.664390 RO Ultimate= 1.470195 RO Corrected= 0.521890 </pre>

REFERENCE

- [1] Ramberg-Osgood relationship (https://en.wikipedia.org/wiki/Ramberg%E2%80%93Osgood_relationship)
- [2] Ramberg, Osgood. Description of stress-strain curves by three parameters. NACA-TN-902, 1 Jul 1943 (<https://ntrs.nasa.gov/api/citations/19930081614/downloads/19930081614.pdf>)
- [3] Hill, H. Determination of stress-strain relations from "offset" yield strength values. NACA-TN-927, 1 Feb 1944. (<https://ntrs.nasa.gov/api/citations/19930081673/downloads/19930081673.pdf>)

- [4] Neuber Plastic Strain Method Theory and Spreadsheet, Abbott Aerospace Canada Ltd.
(<https://www.abbottaerospace.com/aa-sb-001/14-ultimate-strength-of-metallic-elements/14-1-introduction/14-1-2-neuber-plastic-strain-method/>, <https://www.abbottaerospace.com/downloads/aa-sm-030/>)
- [5] Neubers Rule (<https://community.sw.siemens.com/s/article/Neuber-s-Rule>)
- [6] The Strain Life Approach (<https://community.sw.siemens.com/s/article/The-Strain-Life-Approach>)
- [7] The Neuber Correction: A Notch Above the Rest for Linear Stress
(<https://www.altair.com/newsroom/partner-perspectives/the-neuber-correction-a-notch-above-the-rest-for-linear-stress/>)
- [8] FEMFAT plast-module for calculating elasto-plastic stresses based on a linear elastic FE analysis
(<https://femfat.magna.com/index.php?id=522>)
- [9] Nonlinear Stresses from Linear FEA (<https://www.m4-engineering.com/nonlinear-stresses-from-linear-fea/>)
- [10] Metallic Plasticity Modelling - Abaqus FEM Code (https://www.icemm.com/wp-content/uploads/woocommerce_uploads/ISM2006_Plasticity_Modelling-Abaqus-FEM-Code_v2.pdf)
- [11] Plasticity Corrections for Elastic Analysis Results: Neuber Method
(https://community.ptc.com/sejnu66972/attachments/sejnu66972/cpanalysis/307/2/neuber_method.pdf)
- [12] Comparison of Static Failure Criteria of Additively Manufactured Inconel 718 Analyzed by means of Strain Energy Density (<https://ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/2623223/no.ntnu%3Ainspera%3A2524701.pdf?sequence=1&isAllowed=y>)
- [13] Ramberg-Osgood model
(http://www2.me.rochester.edu/courses/ME204/nx_help/index.html#uid:id986788)
- [14] Numerical Validation and Application of the Neuber-Formula in FEA-Analysis
(http://www.iberisa.com/productos/winlife/pdf/Neuber_FEA.pdf)
- [15] An Estimation of Ramberg-Osgood Constants for Materials with and without Luder's Strain Using Yield and Ultimate Strengths. ICSI 2019 The 3rd International Conference on Structural Integrity, Procedia Structural Integrity 17 (2019) 750-757 (www.elsevier.com/locate/procedia).
- [16] MATLAB (<https://www.mathworks.com/>)
- [17] Python (<https://www.python.org>)
- [18] JupyterLab (<https://jupyter.org/try-jupyter/lab/>)

Below references require account at <https://help.3ds.com>

- [19a] DEFORMATION PLASTICITY
([https://help.3ds.com/2023/English/DSSIMULIA_Established/SIMACAEKEYRefMap/simakey-r-deformationplasticity.htm?contextscope=all& \)id=a27580eddaca46de9de51a65a29a3144](https://help.3ds.com/2023/English/DSSIMULIA_Established/SIMACAEKEYRefMap/simakey-r-deformationplasticity.htm?contextscope=all&)id=a27580eddaca46de9de51a65a29a3144)
- [19b] Inelastic Behavior
([https://help.3ds.com/2023/English/DSSIMULIA_Established/SIMACAEMATRefMap/simamat-c-plastic.htm?contextscope=all& id=6c50da490b194d40b71074d25ad8b496 \)](https://help.3ds.com/2023/English/DSSIMULIA_Established/SIMACAEMATRefMap/simamat-c-plastic.htm?contextscope=all& id=6c50da490b194d40b71074d25ad8b496))

Above references require account at <https://help.3ds.com>

APPENDIX

A.1 Ramberg-Osgood Equation

A.2 Python Script – ramosg.py

A.3 MATLAB Script – ramosg.m

A.4 Example Curve Output from ramosg.py/.m

APPENDIX A.1– Ramberg-Osgood Equation [1]

Ramberg–Osgood relationship

 4 languages 
Article [Talk](#)[Read](#) [Edit](#) [View history](#)

From Wikipedia, the free encyclopedia

The **Ramberg–Osgood equation** was created to describe the non linear relationship between [stress](#) and [strain](#)—that is, the [stress–strain curve](#)—in materials near their [yield points](#). It is especially applicable to metals that *harden* with plastic deformation (see [work hardening](#)), showing a *smooth* elastic-plastic transition. As it is a [phenomenological model](#), checking the fit of the model with actual experimental data for the particular material of interest is essential.

In its original form, the equation for strain (deformation) is^[1]

$$\varepsilon = \frac{\sigma}{E} + K \left(\frac{\sigma}{E} \right)^n$$

here

ε is [strain](#),

σ is [stress](#),

E is [Young's modulus](#), and

K and n are constants that depend on the material being considered. In this form K and n are not the same as the constants commonly seen in the [Hollomon equation](#).^[2]

The equation is essentially assuming the elastic strain portion of the stress-strain curve, ε_e , can be modeled with a line, while the plastic portion, ε_p , can be modeling with a power law. The elastic and plastic components are summed to find the total strain.

$$\varepsilon = \varepsilon_e + \varepsilon_p$$

The first term on the right side, σ/E , is equal to the elastic part of the strain, while the second term, $K(\sigma/E)^n$, accounts for the plastic part, the parameters K and n describing the *hardening behavior* of the material. Introducing the *yield strength* of the material, σ_0 , and defining a new parameter, α , related to K as $\alpha = K(\sigma_0/E)^{n-1}$, it is convenient to rewrite the term on the extreme right side as follows:

$$K \left(\frac{\sigma}{E} \right)^n = \alpha \frac{\sigma}{E} \left(\frac{\sigma}{\sigma_0} \right)^{n-1}$$

Replacing in the first expression, the Ramberg–Osgood equation can be written as

$$\varepsilon = \frac{\sigma}{E} + \alpha \frac{\sigma}{E} \left(\frac{\sigma}{\sigma_0} \right)^{n-1}$$

APPENDIX A.2– Python Script – ramosg.py

```

# ramosg.py

# Python 3.x
# Neuber Plasticity Correction

# Revision History
# 2023-06-06

import os
import math
from matplotlib import pyplot as plt
import matplotlib as mpl
import numpy as np

#=====
# functions

def ro(Sp, E, offset, Sy, n):
    """ ro - Ramberg-Osgood stress-strain curve """

    return [x/E + offset*(x/Sy)**n for x in Sp]

#=====
# parameters (SI/MPa)

desc = 'A1011 Steel'

offset = 0.2/100 # percent # % offset, ramberg-osgood parameter
#offset=0.01/100;
#offset = [i/10000 for i in range(1,20+1,1)] # [1e-4:1e-4:2e-3]

E = 200000 # elasticity
Su = 310 # ultimate
Sy = 207 # yield
ef = 25/100 # percent # %elongation at break

#n = 1/0.08364 # ramberg-osgood parameter
#n = range(5,50+1,2) # [5:2:50]
n = math.log(ef/offset)/math.log(Su/Sy)

Se = 400 # linear elastic exceeds Sy

mpl.rcParams['lines.linewidth'] = 2 # linewidth

write_ro = True # write curve to file

#=====
# elastic region

# axis limits
x1 = 0
#x2 = Su/E + 0.001 # strain = 0.001 past ultimate
x2 = Su/E + 0.003 # strain = 0.003 past ultimate
y1 = 0
y2 = x2*E

plt.figure(1)
plt.clf()
plt.plot([x1,x2],[y1,y2],'k') # linear elastic curve
plt.plot([x1,x2],[Su,Su],'k') # ultimate limit
plt.plot([x1,x2],[Sy,Sy],'b') # yield limit
plt.plot([Su/E,Su/E],[0,Su],'k') # ultimate strain
plt.plot([Sy/E,Sy/E],[0,Sy],'b') # yield strain
plt.plot([Se/E,Se/E],[0,Se],'r') # linear elastic stress

#=====
# plastic region

# stress-strain curve
Sp = [x/100 for x in range(0,Su*100+25,25)] # [0:0.25:Su]
ep = ro(Sp, E, offset, Sy, n)
plt.plot(ep,Sp,'g') # Ramberg-Osgood curve

if write_ro:
    # output stress-strain to file (ro_[desc])
    with open('ro_'+desc.replace(" ","_"),'wt') as fid:
        # for x in range(0,len(ep)):
        #     print(f'{ep[x]:f} {Sp[x]:f}',file=fid)

    # include plastic strain relative to yield point
    rng_elas = [i for i in range(0,len(ep)) if Sp[i] < Sy]
    e_Sy = ro([Sy], E, offset, Sy, n)[0]
    for i in range(0,len(ep)):
        if (i <= rng_elas[-1]):

```

```

        if (i == rng_elas[-2]):
            print(f'{ep[i]:f} {Sp[i]:f} ELASTIC {0:f} {0:f}',file=fid)
        elif (i == rng_elas[-1]):
            print(f'{ep[i]:f} {Sp[i]:f} ELASTIC {Sy/E:f} {Sy:f}',file=fid)
        else:
            print(f'{ep[i]:f} {Sp[i]:f} ELASTIC',file=fid)
    else:
        print(f'{ep[i]:f} {Sp[i]:f} PLASTIC {ep[i]-e_Sy:f} {Sp[i]:f}',file=fid)

# Neuber Hyperbola
energy = Se**2/E                                     #  $1/2(Se*Se/E) = 1/2(Sp*ep) = \text{const} \Rightarrow Sp*ep = Se**2/E$ 

s1 = [x*E for x in ep]
e1 = [1e10 if x < 1e-10 else energy/x for x in s1] # e1 = energy./s1;

plt.plot(e1,s1,'y')                                  # Neuber Hyperbola curve
plt.title('Neuber Plasticity Correction Example')
plt.ylabel('Stress [MPa]')
plt.xlabel('Strain [-]')
plt.axis([x1,x2,y1,y2])
#axis([x1,ro(Su, E, offset, Sy, n),y1,y2])

#=====
# find intersection between Ramberg-Osgood and Neuber Hyperbola

# for (e1,s1) need to flip Neuber Hyperbola since defined based on RO curve from Sp=0 to Su
# so e1 is monotonically increasing and replace Inf = energy./s1 since s1 (and ep) start at 0
#e1[0] = 1e10 # np.interp takes care of

# interpolate since values not aligned
ds = [x - np.interp( y, np.flip(e1), np.flip(s1) ) for (x,y) in zip(Sp,ep)]
de = [y - np.interp( x, s1, e1 ) for (x,y) in zip(Sp,ep)]
d = [math.sqrt( x**2 + y**2 ) for (x,y) in zip(ds,de)]
#plt.plot(ep,d,'r-')                                  # distance between, = 0 when cross

ii = d.index( min(d) )
s1 = Sp[ii]                                           # set intersection stress/strain
e1 = ep[ii]

plt.plot(e1,s1, 'ro')                                # Neuber corrected value (1 min value)
plt.plot([e1,e1],[0,s1], 'm')                        # ramberg osgood strain

plt.show()

# search method
## for s in [x/100 for x in range(100*Se,-1,-1): # Se:-0.01:0
##     e1 = s/E + offset*(s/Sy)**n
##     # print(f'{s:f} {energy/e1:f}')
##     if (s < energy/e1):
##         s1 = s
##         break

#=====
# output results
print(f'===(desc)===')
print('    Parameters:');
print(f'        Offset %={offset*100:f}')
print(f'        E [MPa]= {E:f}')
print(f'Elongation@Brk %={ef:f}')
print(f'        n={n:f}')
print('')
print('    Stress [MPa]:')
print(f'    Linear Elastic= {Se:f}')
print(f'        Yield= {Sy:f}')
print(f'        Ultimate= {Su:f}')
print(f'    RO Corrected= {s1:f}')
print('')
print('    Strain %:');
print(f'    Linear Elastic= {Se/E*100:f}')
print(f'    RO Yield= {ro([Sy, E, offset, Sy, n][0]*100:f}')
print(f'    RO Ultimate= {ro([Su, E, offset, Sy, n][0]*100:f}')
print(f'    RO Corrected= {e1*100:f}')

#=====

```

APPENDIX A.3– MATLAB Script – ramosg.m

```

% ramosg.m

% MATLAB '9.5.0.944444 (R2018b)'
% Ramberg-Osgood (RO) Plasticity

% Revision History
% 2023-06-06

clear

%=====
% parameters (SI/MPa)

desc = 'A1011 Steel';

offset = 0.2/100; % percent          % % offset, ramberg-osgood parameter
%offset = 0.01/100;
%offset = [1e-4:1e-4:2e-3];

E = 200000; % elasticity
Su = 310; % ultimate
Sy = 207; % yield
ef = 25/100; % percent %elongation at break

%n = 1/0.08364; % ramberg-osgood parameter
%n = [5:2:50];
n = log(ef/offset)/log(Su/Sy);

Se = 400; % linear elastic exceeds Sy

set(0, 'DefaultLineLineWidth', 2); % linewidth

write_ro = true; % write curve to file

%=====
% elastic region

% axis limits
x1 = 0;
%x2 = Su/E + 0.001; % strain = 0.001 past ultimate
x2 = Su/E + 0.003; % strain = 0.003 past ultimate
y1 = 0;
y2 = x2*E;

figure(1)
clf
plot([x1 x2],[y1,y2],'k') % linear elastic curve
hold on
plot([x1,x2],[Su,Su],'k') % ultimate limit
plot([x1,x2],[Sy,Sy],'b') % yield limit
plot([Su/E,Su/E],[0,Su],'k') % ultimate strain
plot([Sy/E,Sy/E],[0,Sy],'b') % yield strain
plot([Se/E,Se/E],[0,Se],'r') % linear elastic stress

%=====
% plastic region

% stress-strain curve
Sp = [0:0.25:Su]';
ep = ro(Sp, E, offset, Sy, n);
plot(ep,Sp,'g') % Ramberg-Osgood curve

if (write_ro)
    % output stress-strain to file (ro_desc)
    fid = fopen(['ro_' replace(desc, ' ', '_')], 'wt');
    fprintf(fid, '%f %f\n', [ep';Sp]);

    % include plastic strain relative to yield point
    rng_elas = find( Sp < Sy );
    e_Sy = ro(Sy, E, offset, Sy, n);
    for i=1:length(ep)
        if (i <= rng_elas(end))
            if (i == rng_elas(end-1))
                fprintf(fid, '%f %f ELASTIC %f %f\n', ep(i), Sp(i), 0, 0);
            elseif (i == rng_elas(end))
                fprintf(fid, '%f %f ELASTIC %f %f\n', ep(i), Sp(i), Sy/E, Sy);
            else
                fprintf(fid, '%f %f ELASTIC\n', ep(i), Sp(i));
            end
        else
            fprintf(fid, '%f %f PLASTIC %f %f\n', ep(i), Sp(i), ep(i)-e_Sy, Sp(i));
        end
    end
    fclose(fid);
end

% Neuber Hyperbola

```

```

energy = Se^2/E;                                     % 1/2(Se*Se/E) = 1/2(Sp*ep) = const ==> Sp*ep = Se^2/E

s1 = ep*E;
e1 = energy./s1;

plot(e1,s1,'y')                                     % Neuber Hyperbola curve
title('Neuber Plasticity Correction Example')
ylabel('Stress [MPa]')
xlabel('Strain [-]')
axis([x1,x2,y1,y2])
%axis([x1,ro(Su, E, offset, Sy, n),y1,y2])

%=====
% find intersection between Ramberg-Osgood and Neuber Hyperbola

% for (e1,s1) need to flip Neuber Hyperbola since defined based on RO curve from Sp=0 to Su
% so e1 is monotonically increasing and replace Inf = energy./s1 since s1 (and ep) start at 0
e1(1) = 1e10;

% interpolate since values not aligned
ds = Sp - interp1( flip(e1), flip(s1), ep, 'linear' );
de = ep - interp1( s1, e1, Sp, 'linear' );
d = sqrt( ds.^2 + de.^2 );
%plot(ep,d,'r-')                                     % distance between, = 0 when cross

ii = find( d == min(d) );
s1 = Sp(ii(1));                                     % set intersection stress/strain
e1 = ep(ii(1));

plot(e1,s1, 'ro')                                   % Neuber corrected value (1 min value)
plot([e1,e1],[0,s1], 'm')                           % ramberg osgood strain

% search method
%% for s = Se:-0.01:0,
%% e1 = s/E + offset*(s/Sy)^n;
%% % fprintf('%f %f\n',s,energy/e1);
%% if (s < energy/e1),
%%     s1 = s;
%%     break
%% end
%% end

%=====
% output results

fprintf('====s====\n',desc);
fprintf('    Parameters:\n');
fprintf('        Offset %%= %f\n', offset*100);
fprintf('        E [MPa]= %f\n', E);
fprintf('Elongation@Brk %%= %f\n', ef);
fprintf('        n= %f\n', n);
fprintf('\n');
fprintf('    Stress [MPa]:\n');
fprintf('    Linear Elastic= %f\n', Se);
fprintf('        Yield= %f\n', Sy);
fprintf('        Ultimate= %f\n', Su);
fprintf('    RO Corrected= %f\n', s1);
fprintf('\n');
fprintf('    Strain %::\n');
fprintf('    Linear Elastic= %f\n', Se/E*100);
fprintf('    RO Yield= %f\n', ro(Sy, E, offset, Sy, n)*100);
fprintf('    RO Ultimate= %f\n', ro(Su, E, offset, Sy, n)*100);
fprintf('    RO Corrected= %f\n', e1*100);

%=====
% functions

function [ep] = ro(Sp, E, offset, Sy, n)
% ro - Ramberg-Osgood stress-strain curve

    ep = Sp/E + offset.*(Sp/Sy).^n;

end

%=====

```

APPENDIX A.4– Example Curve Output from ramosg.py/m

From the Neuber correction spreadsheet [4] below for A1011 Steel a comparison is shown in Table 2 for both language versions of script ramosg.py/m, similarly to Figure 1.

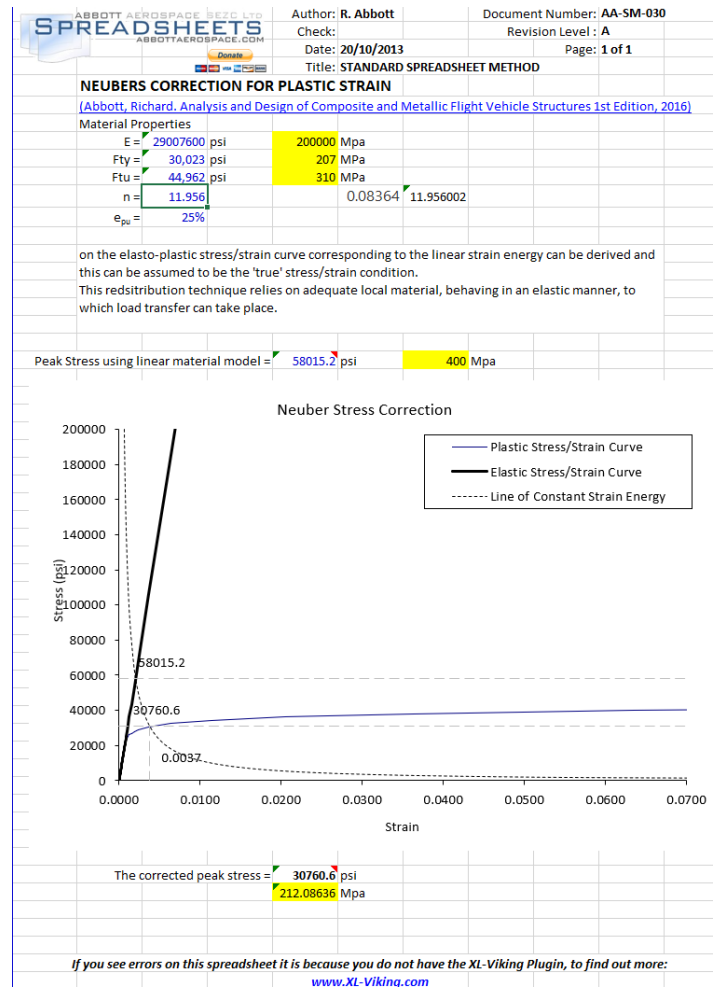
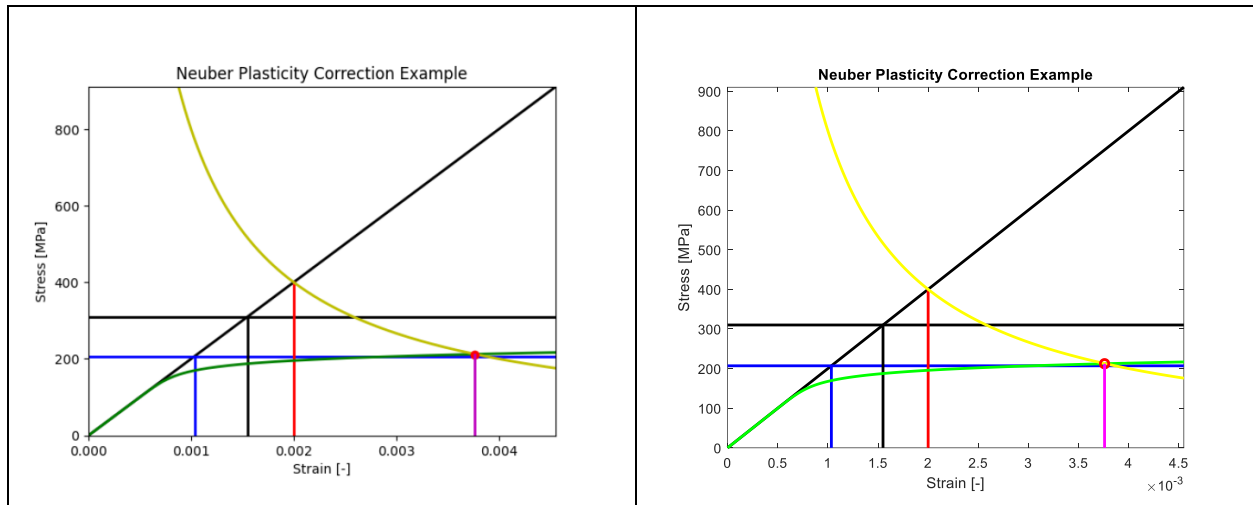


Table 2 – ramosg.py/m Python/MATLAB output

Python	MATLAB
<pre> ===A1011 Steel=== Parameters: Offset %= 0.200000 E [MPa]= 200000.000000 Elongation@Brk %= 0.250000 n= 11.955607 Stress [MPa]: Linear Elastic= 400.000000 Yield= 207.000000 Ultimate= 310.000000 RO Corrected= 212.250000 Strain %: Linear Elastic= 0.200000 RO Yield= 0.303500 RO Ultimate= 25.155000 RO Corrected= 0.375946 </pre>	<pre> ===A1011 Steel=== Parameters: Offset %= 0.200000 E [MPa]= 200000.000000 Elongation@Brk %= 0.250000 n= 11.955607 Stress [MPa]: Linear Elastic= 400.000000 Yield= 207.000000 Ultimate= 310.000000 RO Corrected= 212.250000 Strain %: Linear Elastic= 0.200000 RO Yield= 0.303500 RO Ultimate= 25.155000 RO Corrected= 0.375946 </pre>



If the parameter "write_ro = True" then samples of the Ramberg-Osgood curve will be written to an ASCII file as shown below with columns 1 to 3 being nonlinear Ramberg-Osgood strain, stress (MPa), and region (ELASTIC or PLASTIC)---creating a piecewise linear curve.

In cases where the ELASTIC (linear) and PLASTIC regions are defined separately, as in Explicit codes, columns 4 to 5 can be used. With the ELASTIC region being linear only two points are needed (see yellow region). As many points as are needed can be used to define the PLASTIC region (see green region until end of file). In most cases the PLASTIC strain must begin at zero as noted in [19b]---"Plastic strain values, not total strain values, are used in defining the hardening behavior. Furthermore, the first [PLASTIC region] data pair must correspond with the onset of plasticity (the plastic strain value must be zero in the first pair)." This is accomplished by subtracting the absolute nonlinear Ramberg-Osgood beginning plastic strain (yield point strain) so that the plastic strain begins at the linear yield point as noted below (nonlinear RO yield point strain = 0.003035, linear yield point strain = 0.001035) in the highlighted columns 4 to 5.

```
0.000000 0.000000 ELASTIC
0.000001 0.250000 ELASTIC
0.000003 0.500000 ELASTIC
0.000004 0.750000 ELASTIC
.
.
.
0.002806 205.000000 ELASTIC
0.002833 205.250000 ELASTIC
0.002861 205.500000 ELASTIC
0.002889 205.750000 ELASTIC
0.002917 206.000000 ELASTIC
0.002946 206.250000 ELASTIC
0.002976 206.500000 ELASTIC 0.000000 0.000000
0.003005 206.750000 ELASTIC 0.001035 207.000000
0.003035 207.000000 PLASTIC 0.000000 207.000000
0.003065 207.250000 PLASTIC 0.000030 207.250000
0.003096 207.500000 PLASTIC 0.000061 207.500000
0.003127 207.750000 PLASTIC 0.000092 207.750000
0.003159 208.000000 PLASTIC 0.000124 208.000000
0.003191 208.250000 PLASTIC 0.000156 208.250000
0.003223 208.500000 PLASTIC 0.000188 208.500000
0.003256 208.750000 PLASTIC 0.000221 208.750000
.
.
.
0.232924 308.000000 PLASTIC 0.229889 308.000000
0.235181 308.250000 PLASTIC 0.232146 308.250000
0.237457 308.500000 PLASTIC 0.234422 308.500000
0.239754 308.750000 PLASTIC 0.236719 308.750000
0.242072 309.000000 PLASTIC 0.239037 309.000000
0.244410 309.250000 PLASTIC 0.241375 309.250000
0.246769 309.500000 PLASTIC 0.243734 309.500000
0.249149 309.750000 PLASTIC 0.246114 309.750000
0.251550 310.000000 PLASTIC 0.248515 310.000000
```