

AFRL-IF-RS-TR-2002-221
Final Technical Report
October 2002



JOINT FORCE AIR COMPONENT COMMANDER (JFACC) PROJECT

Washington University

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. J110

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

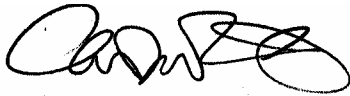
The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

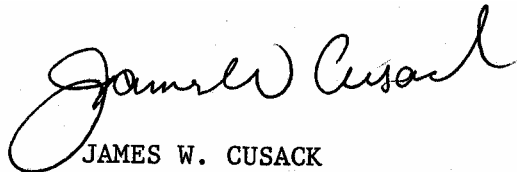
AFRL-IF-RS-TR-2002-221 has been reviewed and is approved for publication.

APPROVED:



CARL A. DEFRANCO, Jr.
Project Engineer

FOR THE DIRECTOR:



JAMES W. CUSACK
Chief, Information Systems Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

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 Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE Oct 02	3. REPORT TYPE AND DATES COVERED Final Sep 99 – Jun 01	
4. TITLE AND SUBTITLE JOINT FORCE AIR COMPONENT COMMANDER (JFACC) PROJECT			5. FUNDING NUMBERS C - F30602-99-2-0551 PE - 63760E PR - J110 TA - 00 WU - 01	
6. AUTHOR(S) Hiro Mukai				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Dept of Systems Science and Mathematics Washington University Campus Box 1040 St Louis, MO 63130			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency 3701 North Fairfax Drive Arlington, VA 22203-1714			10. SPONSORING / MONITORING AGENCY REPORT NUMBER AFRL/IFSA 525 Brooks Rd Rome, NY 13441-4505 AFRL-IF-RS-TR-2002-221	
11. SUPPLEMENTARY NOTES AFRL Project Engineer: Carl DeFranco, IFSA, 315-330-3096, defrancoc@rl.af.mil				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 Words) Application of algorithms based on advanced mathematical & engineering disciplines has the potential of creating an interdisciplinary field of Command & Control. Careful application of the ideas of differential games combined with stochastic processes gives an effective means of countering enemy action in a timely manner and achieving specified goals. This work describes the results of experiments examining the combination of feedback control and game theory, particularly continuous feedback of the Nash solution to a zero-sum game. The work includes a model of air mission dynamics, and applies nonlinear continuous-time deterministic differential equations using an iterative method based on successive local linear or quadratic approximations to reduce complexity. Additional work includes results of Kalman filtering used to estimate enemy state from partial measurements.				
14. SUBJECT TERMS JFACC, command and control, C2 experimentation			15. NUMBER OF PAGES 373	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

Contents

Executive Summary	1
Experiment 1: Combat Modeling and Validation	1
Experiment 2: Controller Performance Comparison with Other Controllers	2
Experiment 3: Controller Performance under Noise in the State Observation	2
Experiment 4: Controller Performance under Parameter Variations	2
Experiment 5: Controller Computational Complexity	3
Experiment 6: Controller with a Kalman Filter for Estimation	3
Experiment 7: Controller Applied to a More Realistic Plant	3
Experiment 8: All Quadratic Method for Nash Computation	3
Experiment 9: Detector Performance under Noise	4
Experiment 10: Detector Performance under Parameter Variations	4
Experiment 11: Method of Characteristics	4
Experiment 12: Game Flow Model	5
Experiment 13: Discrete Platform Dynamics	5
Experiment 14: Non-linear Detector for the Fully Non-linear Model	6
Experiment 15: Comparison with Honeywell's Results	6
Experiment 16: Controller Computational Complexity: Correction	6
Experiment 17: Controller with a Kalman Filter for Estimation	6
Experiment 18: Method of Characteristics: Addendum	7
Experiment 19: New Game Flow Models	7
1 Combat Modeling and Validation	9
1.1 Executive Summary	9
1.2 Purpose of the Experiment	10
1.3 Hypothesis to Prove or Disprove	10
1.4 Introduction	10
1.4.1 Notation	11
1.5 Random Variables	11
1.6 Uncoordinated Target Selection	12
1.6.1 Defining the Markov Chain	13
1.7 Coordinated Target Selection	14
1.7.1 Case 1 : Independent Round Arrival, No Wrap-Around	14
1.7.2 Case 2 : Independent Round Arrival, With Wrap-Around	15
1.7.3 Defining the Markov Chain	16
1.8 Evolution of Expected Values	16
1.8.1 Uncoordinated Target Selection	16
1.8.2 Coordinated Target Selection	17
1.8.3 Summary of ODE's	18
1.9 Weapons Expenditure	18
1.9.1 Uncoordinated Target Selection	19
1.9.2 Coordinated Target Selection	19

1.10	Experiment Results and Analysis	19
1.11	Conclusions and Recommendations	24
1.12	Appendix: Mission Dynamics Continuous-Time Model 3.0	39
1.12.1	Modeling Assumptions	39
1.12.2	State Equations	40
	Bibliography	41
2	Controller: Comparison with Other Controllers	43
2.1	Executive Summary	43
2.2	Experiment Scope	43
2.3	Experiment Results	44
2.3.1	Scenario One: Cross	44
2.3.2	Scenario Two: Joust	52
2.4	Conclusions	61
2.5	References	63
2.6	Appendix	63
2.6.1	Scenario File for Cross	63
2.6.2	Scenario File for Joust	66
3	Controller Performance under Noise	69
3.1	Executive Summary	69
3.2	Purpose of the Experiment	69
3.3	Hypothesis to Prove or Disprove	69
3.4	Experimental Setup	69
3.5	Experimental Results	74
3.6	Conclusions and Recommendations	82
4	Controller Performance under Parameter Variations	83
4.1	Executive Summary	83
4.2	Purpose of the Experiment	83
4.3	Hypothesis to Prove or Disprove	83
4.4	Experimental Setup	83
4.5	Experimental Results	83
4.5.1	Weight Mismatches	84
4.5.2	Experiments with Weight Mismatches	84
4.6	Conclusions and Recommendations	97
5	Controller Computational Complexity	99
5.1	Executive Summary	99
5.2	Introduction	99
5.3	Experiment 5.1	100
5.3.1	One vs. One	100
5.3.2	Multi-units Case	103
5.3.3	Multi-units Case And Computational Complexity	107
5.4	Experiment 5.2	109
5.5	Conclusions	109
6	Controller with a Kalman Filter	111
6.1	Executive Summary	111
6.2	Purpose of the Experiment	111
6.3	Hypothesis to Prove or Disprove	111
6.4	Experiment Setup and Experiment Design	111
6.5	Experiment Results and Analysis	117

6.6	Conclusions and Recommendations	120
	Bibliography	123
7	Controller Applied to a More Realistic Plant	125
7.1	Executive Summary	125
7.2	Purpose of the Experiment	125
7.3	Hypothesis to Prove or Disprove	126
7.4	Experiment Setup	126
7.5	Experiment Results	127
7.6	Analysis	128
7.7	Conclusions and Recommendations	128
	Bibliography	137
8	All Quadratic Method for Nash Computation	139
8.1	Executive Summary	139
8.2	Purpose of the Experiment	139
8.3	Hypotheses to Prove or Disprove	140
8.4	Experiment Setup	140
8.4.1	Problem and Nash Solutions	140
8.4.2	Sequential Quadratic-Quadratic Method	142
8.4.3	Riccati Equation Method	144
8.4.4	SQQM Iterative Algorithm for Game Solution	146
8.5	Experiment Results and Analysis	149
8.6	Conclusions and Recommendations	152
	Bibliography	153
9	Detector Performance under Noise	155
9.1	Executive Summary	155
9.2	Purpose of the Experiment	155
9.3	Hypothesis to Prove or Disprove	156
9.4	Experiment Setup	157
9.5	Example of experiment	161
9.6	Results of the Experiments	165
9.7	Conclusions and Recommendations	165
	Bibliography	175
10	Detector Performance under Parameter Variations	177
10.1	Executive Summary	177
10.2	Purpose of the Experiment	177
10.3	Hypothesis to Prove or Disprove	178
10.4	Experiment setup	179
10.5	Example of experiment	183
10.6	Results of the experiments	186
10.7	Conclusions and Recommendations	186
	Bibliography	193
11	Method of Characteristics	195
11.1	Executive Summary	195
11.2	Purpose of the Experiment	195
11.3	Hypothesis to Prove or Disprove	195
11.4	Experiment Setup	195
11.5	Experiment Results	196
11.5.1	Joust	196

11.5.2 Cross	209
11.6 Analysis	221
11.7 Conclusions and Recommendations	221
Bibliography	223
12 Game Flow Model	225
12.1 Executive Summary	225
12.2 Purpose of the Experiment	225
12.3 Hypothesis to Prove or Disprove	226
12.4 Experiment Setup	226
12.5 Experiment Results	230
12.6 Analysis	235
12.7 Conclusions and Recommendations	235
13 Discrete Platform Dynamics	237
13.1 Executive Summary	237
13.2 Introduction	238
13.3 Hypothesis to Prove	238
13.4 Stochastic Discrete Model Description	238
13.5 Experiment and Methods	239
13.6 Conclusion	239
14 Non-linear Detector for Non-Linear Model	255
14.1 Executive Summary	255
14.2 Purpose of the Experiment	255
14.3 Hypothesis to Prove or Disprove	256
14.4 Experiment Setup	257
14.5 Example of Experiment	259
14.6 Results of the Experiments	262
14.7 Conclusions and Recommendations	266
Bibliography	271
15 Comparison with Honeywell's Results	273
15.1 Executive Summary	273
15.2 Introduction	273
15.3 Experiment Setup	274
15.4 Experiment Results and Analysis	276
15.5 Conclusions and Recommendations	283
16 Controller Computational Complexity: Correction	287
16.1 Executive Summary	287
16.2 Introduction	287
16.3 Experiment 5.1	288
16.3.1 One vs. One	288
16.3.2 Multi-units Case	288
16.3.3 Multi-units Case And Computational Complexity	288
16.4 Experiment 5.2	292
16.5 Conclusions	292

17 Controller with a Kalman Filter for Estimation	293
17.1 Executive Summary	293
17.2 Introduction	293
17.3 Kalman filters for systems with unknown inputs	295
17.4 Experiment scope and setup	300
17.5 Experiment Results and Analysis	302
17.6 Conclusions	320
18 Method of Characteristics: Addendum	323
18.1 Executive Summary	323
18.2 Purpose of the Experiment	323
18.3 Hypothesis	323
18.4 Methods	323
18.5 Experiment Scope	323
18.6 Experiment Results	324
18.7 Analysis	324
18.8 Conclusion	324
19 New Game Flow Models	329
19.1 Executive Summary	329
19.2 Introduction	329
19.3 Mathematical Model	330
19.4 Solution of the System Equations	331
19.5 Differential Game	332
19.6 Solution of the Differential Game	332
19.7 Experimental Results	333
19.7.1 Experiment 1	333
19.7.2 Experiment 2	341
19.8 Conclusions	349

List of Figures

1.1	Probability distribution for Uncoordinated Target Selection for Scenario A	21
1.2	Probability distribution for Coordinated Target Selection for Scenario A	21
1.3	Probability distribution for Uncoordinated Target Selection for Scenario B	22
1.4	Probability distribution for Coordinated Target Selection for Scenario B	22
1.5	Probability distribution for Uncoordinated Target Selection for Scenario C	23
1.6	Probability distribution for Coordinated Target Selection for Scenario C	23
1.7	Evolution of the expected values, Experiment 1.1	26
1.8	Evolution of the expected values, Experiment 1.2	27
1.9	Evolution of the expected values, Experiment 1.3	28
1.10	Evolution of the expected values, Experiment 1.4	29
1.11	Evolution of the expected values, Experiment 1.5	30
1.12	Evolution of the expected values, Experiment 1.6	31
1.13	Evolution of the expected values, Experiment 1.7	32
1.14	Evolution of the expected values, Experiment 1.8	33
1.15	Evolution of the expected values, Experiment 1.9	34
1.16	Evolution of the expected values, Experiment 1.10	35
1.17	Evolution of the expected values, Experiment 1.11	36
1.18	Evolution of the expected values, Experiment 1.12	37
1.19	Comparison of Expected Values for Experiment 1.8	38
2.1	Cross 1: Trajectories	44
2.2	Cross 1: Firing Intensities	45
2.3	Cross 1: Velocities	45
2.4	Cross 1: Number of Platforms	46
2.5	Cross 2: Trajectories	46
2.6	Cross 2: Firing Intensities	47
2.7	Cross 2: Velocities	47
2.8	Cross 2: Number of Platforms	48
2.9	Cross 3: Trajectories	48
2.10	Cross 3: Firing Intensities	49
2.11	Cross 3: Velocities	49
2.12	Cross 3: Number of Platforms	50
2.13	Cross 4: Trajectories	50
2.14	Cross 4: Firing Intensities	51
2.15	Cross 4: Velocities	52
2.16	Cross 4: Number of Platforms	52
2.17	Joust 1: Trajectories	53
2.18	Joust 1: Firing Intensities	53
2.19	Joust 1: Velocities	54
2.20	Joust 1: Number of Platforms	54
2.21	Joust 2: Trajectories	55
2.22	Joust 2: Firing Intensities	55

2.23	Joust 2: Velocities	56
2.24	Joust 2: Number of Platforms	56
2.25	Joust 3: Trajectories	57
2.26	Joust 3: Firing Intensities	57
2.27	Joust 3: Velocities	58
2.28	Joust 3: Number of Platforms	58
2.29	Joust 4: Trajectories	59
2.30	Joust 4: Firing Intensities	59
2.31	Joust 4: Velocities	60
2.32	Joust 4: Number of Platforms	60
3.1	Controller-Plant-Controller Setup	70
3.2	Conceptual Representation of the Controller	72
3.3	The Scenario without any Noise	75
3.4	Average Values of the States and Controls over 100 Sample Paths for the Noise Amplitude 1%	76
3.5	Average Values of the States and Controls over 100 Sample Paths for the Noise Amplitude 10%	76
3.6	Average Values of the States and Controls over 100 Sample Paths for the Noise Amplitude 90%	77
3.7	Average Values of the States and Controls over 100 Sample Paths for the Noise Amplitude 130%	77
3.8	A Sample Path (Noise Amplitude 1%)	78
3.9	A Sample Path (Noise Amplitude 10%)	78
3.10	A Sample Path (Noise Amplitude 90%)	79
3.11	A Sample Path (Noise Amplitude 90%)	79
3.12	A Sample Path (Noise Amplitude 130%)	80
3.13	A Sample Path (Noise Amplitude 130%)	80
3.14	A Sample Path (Noise Amplitude 130%)	81
4.1	Case 1: Red's Weight is 20	85
4.2	Case 2: Red's Weight is 40	85
4.3	Case 3: Red's Weight is 60	86
4.4	Both sides exactly know the probability of kill 0.3	88
4.5	Red Underestimates Blue (Red thinks $\text{pkillBlue}=0.1$) $\text{pkill}:0.3$	88
4.6	Red Overestimates Blue (Red thinks $\text{pkillBlue}=0.8$) $\text{pkill}:0.3$	89
4.7	Blue Underestimates Red (Blue thinks $\text{pkillRed}=0.1$) $\text{pkill}:0.3$	89
4.8	Blue Overestimates Red (Blue thinks $\text{pkillRed}=0.8$) $\text{pkill}:0.3$	90
4.9	Both Sides Underestimate (Red thinks $\text{pkillBlue}=0.1$ similarly for Blue) $\text{pkill}:0.3$	90
4.10	Red Underestimates Blue (Red thinks $\text{pkillBlue}=0.1$) and Blue overestimates Red (Blue thinks $\text{pkillRed}=0.8$) $\text{pkill}:0.3$	91
4.11	Blue Underestimates Red (Blue thinks $\text{pkillRed}=0.1$) and Red overestimates Blue (Red thinks $\text{pkillBlue}=0.8$) $\text{pkill}:0.3$	91
4.12	Both Sides Overestimate (Red thinks $\text{pkillBlue}=0.8$ similarly for Blue) $\text{pkill}:0.3$	92
4.13	Both sides exactly know the probability of kill 0.8	93
4.14	Red Underestimates Blue (Red thinks $\text{pkillBlue}=0.1$) $\text{pkill}:0.8$	93
4.15	Red Underestimates Blue (Red thinks $\text{pkillBlue}=0.4$) $\text{pkill}:0.8$	94
4.16	Blue Underestimates Red (Blue thinks $\text{pkillRed}=0.1$) $\text{pkill}:0.8$	94
4.17	Blue Underestimates Red (Blue thinks $\text{pkillRed}=0.4$) $\text{pkill}:0.8$	95
4.18	Both Sides Underestimate (Red thinks $\text{pkillBlue}=0.1$ similarly for Blue) $\text{pkill}:0.8$	95
4.19	Both Sides Underestimate (Red thinks $\text{pkillBlue}=0.4$ similarly for Blue) $\text{pkill}:0.8$	96
4.20	Both Sides Underestimate (Red thinks $\text{pkillBlue}=0.4$ Blue thinks $\text{pkillRed}=0.1$) $\text{pkill}:0.8$	96
4.21	Both Sides Underestimate (Red thinks $\text{pkillBlue}=0.1$ Blue thinks $\text{pkillRed}=0.4$) $\text{pkill}:0.8$	97

5.1	Initial Trajectories In One vs. One	101
5.2	Control Update In One vs. One	101
5.3	Nash Trajectories In One vs. One	102
5.4	Nash Firing Intensities In One vs. One	102
5.5	Nash Number Of Platforms In One vs. One	103
5.6	Initial Trajectories In Three vs. Three	104
5.7	Control Update In Three vs. Three	105
5.8	Nash Trajectories In Three vs. Three	105
5.9	Nash Firing Intensities In Three vs. Three	106
5.10	Nash Number Of Platforms In Three vs. Three	107
5.11	The Computational Time Changes As The Number Of Units Is Increased	108
5.12	The Number Of Iterations Changes As The Number Of Units Is Increased	108
5.13	The Computational Time Changes As The Mission Duration Is Increased	109
6.1	Block diagram of the Extended Kalman Filter	113
6.2	The flowchart of the estimation algorithm	115
6.3	Closed-loop EKF combined with the game theoretic controller	116
6.4	Blue states, and red states (observed (solid) and estimated (dotted)), no noise.	118
6.5	Blue states, and red states (observed (solid) and estimated (dotted)), maximum noise.	119
6.6	Trajectories of units	120
6.7	Speed controls	121
6.8	Fire intensities	121
6.9	Number of platforms	122
6.10	Weapons per platform	122
7.1	Simulink Implementation of EPMDM	127
7.2	Strategy A (<i>cross</i>): Blue Trajectory	131
7.3	Strategy A (<i>cross</i>): Red Trajectory	131
7.4	Strategy A (<i>cross</i>): Number of Platforms	131
7.5	Strategy A (<i>cross</i>): Fire Intesity	131
7.6	Strategy A (<i>cross</i>): Speed Control	132
7.7	Strategy A (<i>cross</i>): Weapons Expenditures	132
7.8	Strategy B (<i>cross</i>): Blue Trajectory	132
7.9	Strategy B (<i>cross</i>): Red Trajectory	132
7.10	Strategy B (<i>cross</i>): Number of Platforms	133
7.11	Strategy B (<i>cross</i>): Fire Intesity	133
7.12	Strategy B (<i>cross</i>): Speed Control	133
7.13	Strategy B (<i>cross</i>): Weapons Expenditures	133
7.14	Strategy A (<i>joust</i>): Blue Trajectory	134
7.15	Strategy A (<i>joust</i>): Red Trajectory	134
7.16	Strategy A (<i>joust</i>): Number of Platforms	134
7.17	Strategy A (<i>joust</i>): Fire Intesity	134
7.18	Strategy A (<i>joust</i>): Speed Control	135
7.19	Strategy A (<i>joust</i>): Weapons Expenditures	135
7.20	Strategy B (<i>joust</i>): Blue Trajectory	135
7.21	Strategy B (<i>joust</i>): Red Trajectory	135
7.22	Strategy B (<i>joust</i>): Number of Platforms	136
7.23	Strategy B (<i>joust</i>): Fire Intesity	136
7.24	Strategy B (<i>joust</i>): Speed Control	136
7.25	Strategy B (<i>joust</i>): Weapons Expenditures	136
8.1	Convergence of SQQM and SLQM (Model 2, 1 unit vs. 1 unit).	149
8.2	Convergence of SQQM and SLQM (Model 3, 1 unit vs. 1 unit).	150

8.3	Convergence of SQQM and SLQM (Model 2, 5 units vs. 5 units).	151
8.4	Convergence of SQQM and SLQM (Model 3, 5 units vs. 5 units).	151
8.5	Convergence of SQQM, SLQM and SLQM-SQQM (Model 2, 5 units vs. 5 units).	152
9.1	Engagement enemy actions (to be detected).	158
9.2	Time history of the number of platforms of the four units. The two top plots represent the number of platforms of the Red units, the two bottom plots represent the number of platforms of the Blue units.	159
9.3	Noisy outputs. Noise-corrupted measured observations (top: number of platforms of the first Blue unit, bottom: number of platforms of the second Blue unit).	160
9.4	Block diagram describing the experiment.	161
9.5	Response to Action 1 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 15% of the energy of the noise-free output.	162
9.6	Response to Action 2 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 15% of the energy of the noise-free output.	163
9.7	State evolution of the game-theoretic (top) and Kalman-like filter (bottom).	164
9.8	Response to Action 1 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 25% of the energy of the noise-free output.	165
9.9	Response to Action 2 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 25% of the energy of the noise-free output.	167
9.10	Response to Action 1 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 35% of the energy of the noise-free output.	167
9.11	Response to Action 2 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 35% of the energy of the noise-free output.	168
9.12	Response to Action 1 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 55% of the energy of the noise-free output.	168
9.13	Response to Action 2 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 55% of the energy of the noise-free output.	169
9.14	Response to Action 1 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 80% of the energy of the noise-free output.	169
9.15	Response to Action 2 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 80% of the energy of the noise-free output.	170
9.16	Response to Action 1 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 110% of the energy of the noise-free output.	170
9.17	Response to Action 2 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 110% of the energy of the noise-free output.	171
9.18	Responses of the game-theoretic (top) and Kalman-like filter (bottom) to Action 1. The six responses correspond to six different random choices of the seed of the noise signals. The energy of the noise signals is equal to 15% of the output signal energy.	171
9.19	Responses of the game-theoretic (top) and Kalman-like filter (bottom) to Action 2. The six responses correspond to six different random choices of the seed of the noise signals. The energy of the noise signals is equal to 15% of the output signal energy.	172
9.20	Responses of the game-theoretic (top) and Kalman-like filter (bottom) to Action 1. The six responses correspond to six different random choices of the seed of the noise signals. The energy of the noise signals is equal to 25% of the output signal energy.	172
9.21	Responses of the game-theoretic (top) and Kalman-like filter (bottom) to Action 2. The six responses correspond to six different random choices of the seed of the noise signals. The energy of the noise signals is equal to 25% of the output signal energy.	173
9.22	Responses of the game-theoretic (top) and Kalman-like filter (bottom) to Action 1. The six responses correspond to six different random choices of the seed of the noise signals. The energy of the noise signals is equal to 35% of the output signal energy.	173

9.23	Responses of the game-theoretic (top) and Kalman-like filter (bottom) to Action 2. The six responses correspond to six different random choices of the seed of the noise signals. The energy of the noise signals is equal to 35% of the output signal energy.	174
10.1	Engagement enemy actions (to be detected).	180
10.2	Time history of the number of platforms of the four units. The two top plots represent the number of platforms of the Red units, the two bottom plots represent the number of platforms of the Blue units.	181
10.3	Noisy outputs. Noise-corrupted measured observations (top: number of platforms of the first Blue unit; bottom: number of platforms of the second Blue unit).	182
10.4	Block diagram describing the experiment.	183
10.5	Response to Action 1 of the detection filter in the presence of uncertainty in the parameter α_1 . The actual value of α_1 varies from 110% (top graph) to 150% (bottom graph) of the nominal value.	184
10.6	Response to Action 2 of the detection filter in the presence of uncertainty in the parameter α_1 . The actual value of α_1 varies from 110% (top graph) to 150% (bottom graph) of the nominal value.	185
10.7	Response to Action 1 of the detection filter in the presence of uncertainty in the parameter α_2 . The actual value of α_2 varies from 110% (top graph) to 150% (bottom graph) of the nominal value.	186
10.8	Response to Action 2 of the detection filter in the presence of uncertainty in the parameter α_2 . The actual value of α_2 varies from 110% (top graph) to 150% (bottom graph) of the nominal value.	187
10.9	Response to Action 1 of the detection filter in the presence of uncertainty in the parameter β_{11} . The actual value of β_{11} varies from 110% (top graph) to 150% (bottom graph) of the nominal value.	187
10.10	Response to Action 2 of the detection filter in the presence of uncertainty in the parameter β_{11} . The actual value of β_{11} varies from 110% (top graph) to 150% (bottom graph) of the nominal value.	188
10.11	Response to Action 1 of the detection filter in the presence of uncertainty in the parameter β_{12} . The actual value of β_{12} varies from 110% (top graph) to 150% (bottom graph) of the nominal value.	188
10.12	Response to Action 2 of the detection filter in the presence of uncertainty in the parameter β_{12} . The actual value of β_{12} varies from 110% (top graph) to 150% (bottom graph) of the nominal value.	189
10.13	Response to Action 1 of the detection filter in the presence of uncertainty in the parameter β_{21} . The actual value of β_{12} varies from 110% (top graph) to 150% (bottom graph) of the nominal value.	189
10.14	Response to Action 2 of the detection filter in the presence of uncertainty in the parameter β_{12} . The actual value of β_{12} varies from 110% (top graph) to 150% (bottom graph) of the nominal value.	190
10.15	Response to Action 1 of the detection filter in the presence of uncertainty in the parameter β_{22} . The actual value of β_{22} varies from 110% (top graph) to 150% (bottom graph) of the nominal value.	190
10.16	Response to Action 2 of the detection filter in the presence of uncertainty in the parameter β_{22} . The actual value of β_{22} varies from 110% (top graph) to 150% (bottom graph) of the nominal value.	191
11.1	Trajectories	197
11.2	Engagement Intensities and Number of Platforms	198
11.3	Velocities	199
11.4	Trajectories	200
11.5	Engagement Intensities and Number of Platforms	201

11.6	Velocities	202
11.7	Trajectories	203
11.8	Engagement Intensities and Number of Platforms	204
11.9	Velocities	205
11.10	Trajectories	206
11.11	Engagement Intensities and Number of Platforms	207
11.12	Velocities	208
11.13	Trajectories	209
11.14	Engagement Intensities and Number of Platforms	210
11.15	Velocities	211
11.16	Trajectories	212
11.17	Engagement Intensities and Number of Platforms	213
11.18	Velocities	214
11.19	Trajectories	215
11.20	Engagement Intensities and Number of Platforms	216
11.21	Velocities	217
11.22	Trajectories	218
11.23	Engagement Intensities and Number of Platforms	219
11.24	Velocities	220
12.1	Map of the game area: light colored cells indicate smooth area. The white contour line marks the boundary between the two different regions.	227
12.2	Efficiency of attack for the blue force.	228
12.3	Efficiency of attack for the red force.	229
12.4	(a) Running cost associated with instant values of the strength concentration in the cells. (b) Terminal cost associated with final values of the strength concentration in the cells. Lighter color indicates higher value.	229
12.5	Convergence of SLQ algorithm	230
12.6	Initial Strength Distribution for Blue force. Arrows indicate magnitude of the velocity components across the boundaries.	231
12.7	Initial Strength Distribution for Red force. Arrows indicate magnitude of the velocity components across the boundaries.	232
12.8	Final Strength Distribution for Blue force. Arrows indicate magnitude of the velocity components across the boundaries.	233
12.9	Final Strength Distribution for Red force. Arrows indicate magnitude of the velocity components across the boundaries.	234
13.1	Comparison of MDCM (- -) and MDCM-SD (-) for Game Trajectories (One sample run).	242
13.2	Comparison of MDCM (- -) and MDCM-SD (-) for the Number of Platforms (One sample run).	242
13.3	Comparison of MDCM (- -) and MDCM-SD (-) for Fire Intensities (One sample run).	243
13.4	Comparison of MDCM (- -) and MDCM-SD (-) for Weapons Expenditures (One sample run).	243
13.5	Comparison of MDCM (- -) and MDCM-SD (-) for Speed (One sample run).	244
13.6	Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Game Trajectories ($\Delta t = 0.001$).	245
13.7	Comparison of MDCM (- -) and Averaged MDCM-SD (-) for the Number of Platforms ($\Delta t = 0.001$).	246
13.8	Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Fire Intensities ($\Delta t = 0.001$).	246
13.9	Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Weapons Expenditures ($\Delta t = 0.001$).	247
13.10	Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Speed ($\Delta t = 0.001$).	247
13.11	Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Game Trajectories ($\Delta t = 0.01$).	248

13.12	Comparison of MDCM (- -) and Averaged MDCM-SD (-) for the Number of Platforms ($\Delta t = 0.01$).	249
13.13	Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Fire Intensities ($\Delta t = 0.01$).	249
13.14	Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Weapons Expenditures ($\Delta t = 0.01$).	250
13.15	Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Speed ($\Delta t = 0.01$).	250
13.16	Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Game Trajectories ($\Delta t = 0.1$).	251
13.17	Comparison of MDCM (- -) and Averaged MDCM-SD (-) for the Number of Platforms ($\Delta t = 0.1$).	251
13.18	Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Fire Intensities ($\Delta t = 0.1$).	252
13.19	Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Weapons Expenditures ($\Delta t = 0.1$).	252
13.20	Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Speed ($\Delta t = 0.1$).	253
14.1	Geographical areas in which Red and Blue units can evolve.	258
14.2	Trajectories for the Red and Blue units in Experiments 1 and 2.	258
14.3	Engagement actions of the Red units versus the Blue units in Experiment 1.	259
14.4	Evolution in time of the number of platforms in the Blue and Red Units in Experiment 1.	260
14.5	Action 1 and Performance Signal 1 in the Experiment 1.	260
14.6	Action 2 and Performance Signal 2 in the Experiment 1.	261
14.7	Internal state of the detection filter in Experiment 1.	261
14.8	Actions 1 and 2 and Performance Signal 1 for the Linear Detection Filter in Experiment 1.	262
14.9	Actions 1 and 2 and Performance Signal 2 for the Linear Detection Filter in Experiment 1.	263
14.10	Actions 1 and 2 of the Red Units in Experiment 2.	263
14.11	Evolution in time of the number of platforms of the Blue Units in Experiment 1 and Experiment 2.	264
14.12	Action and Performance Signal 1 in Experiment 2.	264
14.13	Action and Performance Signal 2 in Experiment 2.	265
14.14	Actions 1 and 2 and Performance Signal 1 of the linear filter in Experiment 2.	265
14.15	Actions 1 and 2 and Performance Signal 2 of the linear filter in Experiment 2.	266
14.16	Trajectories of the Red and Blue Units in Experiment 3.	267
14.17	Evolution in time for the number of platforms of the Blue Units in Experiment 1 and Experiment 3.	267
14.18	Action 1 and Performance Signal 1 in Experiment 3.	268
14.19	Action 2 and Performance Signal 2 in Experiment 3.	268
14.20	Actions 1 and 2 and Performance Signal 1 of the linear filter in Experiment 3.	269
14.21	Actions 1 and 2 and Performance Signal 2 of the linear filter in Experiment 3.	269
15.1	Flowchart for interaction of software components	275
15.2	Initial positions of the units.	276
15.3	Control update in Example 1	278
15.4	Cost function value for Example 1	278
15.5	Trajectories of units, Example 1, Sortie 1.	279
15.6	Trajectories of units, Example 1, Sortie 2.	279
15.7	Trajectories of units, Example 1, Sortie 3.	280
15.8	Control update in Example 2	280
15.9	Cost function value for Example 2	281
15.10	Trajectories of units, Example 2, Sortie 1.	281
15.11	Trajectories of units, Example 2, Sortie 2.	282
15.12	Trajectories of units, Example 2, Sortie 3.	282
16.1	Initial Trajectories for Three vs. Three	289
16.2	Convergence for Control Updates for Three vs. Three	289

16.3 Nash Trajectories for Three vs. Three	290
16.4 Nash Firing Intensities for Three vs. Three	290
16.5 Nash Number Of Platforms for Three vs. Three	291
16.6 The Computational Time Changes As The Number Of Units Is Increased	291
17.1 Block diagram of the extended Kalman filter	298
17.2 The flow chart of the estimation algorithm.	299
17.3 The closed-loop game theoretic controller combined with the Kalman filter.	301
17.4 Observed Trajectories of Units	303
17.5 Observed Numbers of Platforms	303
17.6 Weapons per platform	304
17.7 Speed Controls	304
17.8 Fire Intensities	305
17.9 Blue states for high sensor noise	307
17.10 Red states for high sensor noise	307
17.11 Red inputs for high sensor noise	308
17.12 Blue states for high process noise	309
17.13 Red states for high process noise	309
17.14 Red inputs for high process noise	310
17.15 Blue states for high process and sensor noises	311
17.16 Red states for high process and sensor noises	311
17.17 Red inputs for high process and sensor noise	312
17.18 Trajectories of Units	313
17.19 Number of Platforms	313
17.20 Weapons per platform	314
17.21 Speed Controls	314
17.22 Fire Intensities	315
17.23 Blue states for high sensor noise for unit1	316
17.24 Blue states for high sensor noise for unit2	316
17.25 Red states for high sensor noise for unit1	317
17.26 Red states for high sensor noise for unit2	317
17.27 Red states for high sensor noise for unit3	318
17.28 Red inputs for high sensor noise for unit1	318
17.29 Red inputs for high sensor noise for unit2	319
17.30 Red inputs for high sensor noise for unit3	319
18.1 Nash Trajectories for Three Units vs. Three Units	324
18.2 Nash Engagement Intensities and Number of Platforms	325
19.1 Cell numbering scheme.	331
19.2 Game Board (clockwise starting from top left figure: terminal payoff associated with final values of the strength concentration in the cells; running cost associated with instantaneous values of the strength concentration in the cells; local attrition; and running cost on velocity). Darker shade indicates higher value.	334
19.3 Initial strength distribution of the blue force.	335
19.4 Initial strength distribution of the red force.	335
19.5 Efficiency of attack for the blue and red forces.	336
19.6 Initial Nash Strength Distribution for Blue force. Arrows indicate magnitudes of the velocity components across the boundaries.	338
19.7 Initial Nash Strength Distribution for Red force. Arrows indicate magnitudes of the velocity components across the boundaries.	338
19.8 Final Nash Strength Distribution for Blue force. Arrows indicate magnitudes of the velocity components across the boundaries.	339

19.9	Final Nash Strength Distribution for Red force. Arrows indicate magnitudes of the velocity components across the boundaries.	339
19.10	Final Nash Strength Distribution for Blue force.	340
19.11	Final Nash Strength Distribution for Red force.	341
19.12	Game Board (clockwise starting from top left figure: terminal payoff associated with final values of the strength concentration in the cells; running cost associated with instant values of the strength concentration in the cells; local attrition; and running cost on velocity). Darker shade indicates higher value.	342
19.13	Initial strength distribution of the blue force.	343
19.14	Initial strength distribution of the red force.	343
19.15	Efficiency of attack for the blue and red forces.	344
19.16	Convergence of the SLQ algorithm.	345
19.17	Initial Nash Strength Distribution for Blue force. Arrows indicate magnitudes of the velocity components across the boundaries.	346
19.18	Initial Strength Distribution for Red force. Arrows indicate magnitudes of the velocity components across the boundaries.	346
19.19	Final Nash Strength Distribution for Blue force. Arrows indicate magnitudes of the velocity components across the boundaries.	347
19.20	Final Nash Strength Distribution for Red force. Arrows indicate magnitudes of the velocity components across the boundaries.	347
19.21	Final Nash Strength Distribution for Blue force.	348
19.22	Final Nash Strength Distribution for Red force.	348
19.23	Progression of the instantaneous Nash Strength Distributions for Blue and Red forces. Arrows indicate magnitudes of the velocity components across the boundaries.	349

List of Tables

1.1	The Approximate Evolution of Expected Number of Platforms	18
1.2	Scenarios Used For Experiments	19
1.3	L_2 -Norm of Covariances	24
1.4	L_2 -Norm of Error Between Actual and Approximate Expected Values	24
2.1	List of Scenarios	44
2.2	Scenario One – Cross (Parameters Value)	61
2.3	Scenario Two – Joust (Parameters)	61
2.4	Experiment Results for Scenario one – Parameter Values	62
2.5	Experiment Results for Scenario One – Cost Components	62
2.6	Experiment Results for Scenario Two – Parameter Values	62
2.7	Experiment Results for Scenario Two – Cost Components	62
3.1	The Noise Levels for the Experiments	74
4.1	Different Experimental Setup for Weight Mismatches, Weights on Final Number of Red Platforms	84
4.2	Parameter Mismatches for the Experiments	87
5.1	Data For One vs. One	100
5.2	Data For Three vs. Three	103
5.3	Weights In Cost Function For Three vs. Three	104
7.1	Scenario Description	128
7.2	Cost Components of Objective Function Using <i>cross</i> Scenario	129
7.3	Total Costs of Objective Function using <i>cross</i> Scenario	129
7.4	Cost Components of Objective Function Using <i>joust</i> Scenario	129
7.5	Total Costs of Objective Function Using <i>joust</i> Scenario	129
12.1	Payoff function value for the initial guessed solution	230
12.2	Payoff function value for the Nash equilibrium solution	231
13.1	Cross 11 Scenario description	239
13.2	Summary of Results for One Sample Run with $\Delta t = 0.1$	244
13.3	Summary of Results Averaged over 100 Sample Runs for $\Delta t = 0.001$	245
13.4	Summary of Results Averaged over 100 Sample Runs for $\Delta t = 0.01$	248
13.5	Summary of Results Averaged over 100 Sample Runs for $\Delta t = 0.1$	253
15.1	Probability of Kill Values	276
15.2	Weights in the Cost Function	277
15.3	Example 1: Probability of Success and Remaining Number of Platforms	283
15.4	Example 2: Probability of Success and Remaining Number of Platforms	283

16.1 Data for Three vs. Three	288
19.1 Payoff function value for the initial solution estimate	337
19.2 Payoff function value for the Nash equilibrium solution	337
19.3 Payoff function value for the initial solution estimate	345
19.4 Payoff function value for the Nash equilibrium solution	349

Acknowledgments and Disclaimer

We would like to thank the following people from other agencies: JFACC Program Manager at the DARPA Agency: Major Sharon Heise, Ph. D.; Government Agents from the Air Force Research Laboratory at Rome: Drs. Timothy Busch and Carl DeFranco; Members from Emergent Information Technologies: Drs. Steve Morse and Mike Ownby.

We also would like to thank people from our office: Ms. Annette Crain and Ms. Kim Kalter.

Acknowledgments: "Effort sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-99-2-0551. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon."

Disclaimer: "The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the defense Advanced Research Projects Agency (DARPA), the Air Force Research Laboratory, or the U.S. Government."

Project Roster

Project Principal Investigator

Prof. Hiro Mukai

Co-Principal Investigators

Dean Christopher I. Byrnes

Prof. Alberto Isidori

Faculty Associates

Prof. I. Norman Katz

Prof. Heinz M. Schättler

Prof. Liyi Dai

Senior Research Associate

Dr. Akio Tanikawa

Postdoctoral Research Associates

Dr. Alpay Özcan

Dr. Claudio De Persis

Dr. Fikret Çalıřkan

Dr. Guan Jun Wang

Dr. Ilker Tunay

Dr. Hong Gao

Dr. Mike Malisoff

Dr. Mingjun Zhang

Dr. Rafael de la Guardia

Dr. Raffaella De Santis

Dr. Ruisheng Li

Dr. Yuichi Sawada

Graduate Research Assistants

Ms. Chun-hua Lan

Mr. Evan Maki

Mr. Jenner Joseph

Mr. Long Yang

Mr. Paolo Rinaldi

Mr. Min Xu

Undergraduate Research Assistants

Mr. Justin Goodwin

Ms. Shirley Birman

List of Contributors

Chap.	Software and Experiments Interim Technical Report	Writer Submitted on Feb. 28, 2001	Editor
1	Tunay, Goodwin	Tunay	Schättler
2	de la Guardia, Zhang, Tunay	de la Guardia, Zhang, Wang	Katz
3	Özcan, Wang, Tunay	Özcan, Tanikawa	Schättler
4	Özcan, Tunay	Özcan, Tanikawa	Schättler
5	Wang, Gao, Rinaldi, Tunay	Wang	Schättler
6	Çalışkan, Lan, Tunay	Çalışkan	Katz
7	Tunay, Goodwin	Tunay	Katz
8	Rinaldi	Rinaldi, Tanikawa	Mukai
9	De Persis	De Persis	Isidori
10	De Persis	De Persis	Isidori
11	Zhang	Tanikawa, Zhang	Katz
12	de la Guardia, Sawada	de la Guardia	Mukai
	Addendum	Submitted on June 30, 2001	
13	Goodwin, Maki	Goodwin, Maki	Katz
14	De Persis	De Persis	Isidori
15	Tunay	Tunay	Mukai
16	Rinaldi,	Rinaldi	Mukai
17	Caliskan, Lan	Caliskan	Katz
18	Xu	Xu	Katz
19	de la Guardia	de la Guardia	Mukai

Executive Summary

Experiments 1 through 12 were reported as the Interim Technical Report on February 28, 2001. Experiments 13 through 19 were added on June 28, 2001.

Experiment 1: Combat Modeling and Validation

The purpose of this experiment is to validate the low-order ordinary differential equation (ODE) models, which are derived to approximate the evolution of expected values in a more realistic hybrid-stochastic model, called the Probabilistic Mission Dynamics Model (PMDM), under different assumptions for target acquisition and target selection coordination.

The hypothesis is that the evolution of the expected values of the Markov chain (MC) mission dynamics can be approximated by a low-order ordinary differential equation (ODE) model, for a time period of sufficient duration, when the control signals are generated in an open-loop setting.

This study first identifies four different sets of assumptions about target acquisition and target selection coordination, which will be abbreviated as:

MARI (Acquisition Rate Independent) Uncoordinated target selection, independent target acquisition,

MARD (Acquisition Rate Dependent) Uncoordinated target selection, linear target acquisition,

MNWA (No Wrap-around) Coordinated target selection, without wrap-around,

MWWA (With Wrap-around) Coordinated target selection, with wrap-around.

Then, continuous-transition Markov chain (MC) models are developed under these assumptions. Using the MC models, the probability distributions for the number of platforms and their exact expected values η^B, η^R are calculated as a function of time. Next, approximate ODE models of the evolution of the expected values are derived.

The trajectories of the ODE models ($\hat{\eta}^B, \hat{\eta}^R$) are compared with the exact outcomes (η^B, η^R) in twelve experiments, summarized in the table below, which indicates the initial number of platforms and the probability of kill for the Blue and Red units.

Scenarios Used In the Experiments	
Scenario Types; $N^B = 8, P_k^B = 0.8$	
A	$N^B = N^R, P_k^B = P_k^R$
B	$N^B = 2N^R, P_k^B = P_k^R$
C	$N^B = N^R, P_k^B = 2P_k^R$

The approximation quality of the ODE models is compared in the following table using the L_2 norm:

$L_2[0, 10]$ -Norm of Error Between Actual and Approximate Expected Values

Exp. #	1	2	3	4	5	6	7	8	9	10	11	12
	MARI			MARD			MNWA			MWWA		
$\ \eta^B - \hat{\eta}^B\ $	0.14	0.37	0.01	0.06	0.03	0.02	1.35	5.80	0.02	1.53	4.74	0.16
$\ \eta^R - \hat{\eta}^R\ $	0.14	1.35	0.25	0.06	0.06	0.04	1.35	5.80	0.04	1.53	0.06	0.58
$\ \eta^B - \hat{\eta}^B + \eta^R - \hat{\eta}^R\ $	0.28	1.72	0.26	0.12	0.09	0.06	2.70	11.60	0.06	3.06	4.80	0.74

It was observed that the ODE models were good approximations under the uncoordinated target selection assumption, and were found to be sufficient to represent the attrition dynamics in a differential game setup in this case. The discrepancy between the MC and ODE models increase as the engagement proceeds, which should be expected. Under the coordinated target selection assumption, the ODE approximations were worse. This can be partially explained by the fact that coordination implies firing in rounds, and therefore platform loss is more discrete in nature.

Experiment 2: Controller Performance Comparison with Other Controllers

This is experiment for hypothesis two. Both the plant and internal models are the same, i.e., the Mission Dynamics Continuous-time Model (MDCM). There is no noise added to the state variables when constructing the observed state variables (the output variables). The control actions of the Blue and Red teams are generated by one of the following strategies: the proposed game theoretic algorithm, a simple heuristic stochastic strategy (e.g. a movement bias is given toward targets), a simple heuristic deterministic strategy, and a human planner.

The strategy adopted by Blue and Red is optimal with sense of a Nash equilibrium with respect to the value function; that is, it maximizes the value function with respect to Red and minimizes it with respect to Blue.

Experiment 3: Controller Performance under Noise in the State Observation

We performed a series of experiments to evaluate the effectiveness of the current differential game technology as a means of countering enemy actions under idealized situations with perfect information about the enemy initial conditions and objectives, but with noisy measurements of the enemy state. Our main findings are that while average values look good, individual sample paths might be quite surprising. One can conclude that the game theoretic controller CPC (Controller-Plant-Controller) is sensitive to observation noise. The first step to remedy the noise problem is to implement proper filters in the controllers.

Experiment 4: Controller Performance under Parameter Variations

The purpose is to test how the Controller-Plant-Controller setup (CPC) will react to parameter mismatches between the battlefield and the sides as well as to parameter mismatches between the sides. Assuming that both sides have chosen the game theory as the intelligence behind their controllers, systematic tests have been performed to investigate its sensitivity, i.e., how strongly the proposed game-theoretic controller reacts to changes in the parameters. The important conclusion to draw from these experiments is that even a single parameter can have important effects on the outcome of the battle. It is therefore very important to be able to estimate the enemy parameters in order to succeed in the battle simulation.

Experiment 5: Controller Computational Complexity

The *purpose* of experiment 5 is to test *Hypothesis 5*: The computational complexity of the differential game technology based controller, combined with an extended Kalman filter or a nonlinear observer, increases quadratically as a function of the number of units and linearly as a function of the mission duration.

A number of experiments have been performed to test that Hypothesis.

In the set of experiments, both the plant and internal models are the same, given by MDCM. In a first set of experiments we increase the number of units in the scenario while the mission objectives and duration are kept constant. In a second set, the mission duration is increased, while the mission objectives and the number of units are kept constant. The computation time and the number of iterations required for the computation of the control law to converge were recorded in both cases.

Our main *conclusions* are that the computational time required to reach the convergence criterion depends on many factors, such as the units categories, the number of units, initial trajectories, weights in the cost function, step size in our numerical procedure and the manner of engagements as well as initial positions and target locations. Similarly the number of iterations required to reach a convergence criterion depends on the same factors. From our experimental results, major factors which affect the computational time are the number of units and mission duration. As expected from theoretical considerations the computational time of the controller increased quadratically as a function of the number of units. We also saw that it increased linearly as a function of the mission duration, while the number of iterations remained relatively constant as a function of the number of units.

Experiment 6: Controller with a Kalman Filter for Estimation

In this chapter, we present an algorithm based on the Extended Kalman Filter (EKF) for state estimation when enemy inputs are unavailable. We show the overall structure of the estimation scheme through a block diagram. We present the implementation of the algorithm for the air operation theater through a flowchart. We also present the results of simulation experiments.

Experiment 7: Controller Applied to a More Realistic Plant

The purpose of this experiment is to observe the effect of the discrepancy between internal and plant models in a closed-loop setting. The internal model is a reduced-order ODE model, called the Mission Dynamics Continuous-time Model (MDCM 3.0), and the plant model is a full order ODE model, abbreviated as EPMDM, which exactly describes the evolution of expected values in PMDM.

The hypothesis is that the current differential game technology would provide an effective means of countering the enemy actions, who may be either following the Nash solution or using some simple heuristic strategy, when noise-free state measurements are available, in spite of the mismatch between the plant and the internal models.

It is concluded that approximating the plant model with a lower order internal model does not cause a significant difference in game results, as long as the engagement terminates before one side is completely wiped-off.

Experiment 8: All Quadratic Method for Nash Computation

The purpose of Experiment 8 is to develop, implement and test the Sequential Quadratic-Quadratic Method (SQQM) for Differential Games, with two hypotheses of interest. The first hypothesis tests whether the Nash solution computed through the Sequential Quadratic-Quadratic Method is identical

to the one found using the Sequential Linear-Quadratic Algorithm (SLQM); the second hypothesis tests whether there is an improvement in convergence time. The issue of speed can become of great importance in real time applications; moreover, due to the presence of nonlinearity and constraints, a different approach serves the purpose of validating previous results. The algorithm is based on an iterative method for computing a Nash solution to a zero-sum differential game with a system of nonlinear differential equations.

Several experiments on different scenarios, based on both Model 2 and Model 3, have shown the convergence of the outputs of the SQQM and SLQM algorithms to the same solution. So the first hypothesis of Experiment 8 is proven true. As for the second hypothesis, namely an improvement in convergence speed, the conclusion is that the SQQM alone proves to be fast in simple scenarios; if, however, the starting trajectory and costate estimates are too far from the optimal solution, the SLQM may be used at first, and then switch to the SQQM once the solution estimate is closer to the optimal solution. In more complex cases, it is thus advantageous to blend the linear-quadratic algorithm and the quadratic-quadratic algorithm, taking advantage of both the superior stability of the SLQM and the superior speed of the SQQM.

Experiment 9: Detector Performance under Noise

In this Chapter we report the experiments performed to test the effectiveness of a *newly designed* “game-theoretic-optimal” detection filter in handling noise-corrupted observations of the battlefield. The basic purpose of the detection filter is to reveal the occurrence of an “engagement action” from enemy units by monitoring only variables associated with the friendly units. The game-theoretic approach to the design of the filter makes it possible to attenuate the effects of measurement noises, but not the effects of the action to be detected. The outcome of the experiments shows very clearly that the game-theoretic filter is very effective under different situations of noise and compares very favorably with a filter designed on the basis of classical state-estimation methods.

Experiment 10: Detector Performance under Parameter Variations

This Chapter describes the experiment results regarding the game-theoretic detection filter under parametric uncertainty. The exact values of the parameters in the mathematical model of the battlefield are not known, and only a nominal value is available. The filter, whose objective is to reveal the occurrence of an “engagement action” from enemy units, is designed on the basis of the nominal value. This set of experiments shows that the game-theoretic detection filter, although proven to be effective in the selective attenuation of measurement noise, is relatively sensitive to the uncertainty in the parameters.

Experiment 11: Method of Characteristics

The purpose is to verify that the solution computed by the Sequential Linear-Quadratic Method (SLQM) is the same as the Nash solution computed by the Method of Characteristics. We verified that the solutions computed by the Sequential Linear-Quadratic Method (SLQM) are the same as the Nash solutions computed by the Method of Characteristics under several scenarios. Also, systematic tests have been performed to study robustness under two ways of enforcing constraints: penalties and explicit enforcement. Specifically, weights for velocities, engagement intensities, final numbers of platforms and targets, as well as maximum rated speeds have been varied. The results show that the trajectories are quite similar in shape.

Experiment 12: Game Flow Model

The purpose of this experiment was to validate the Game Flow approach. Validation is meant in the sense that the game theoretic solution engine (i.e., the *Sequential Linear – Quadratic* algorithm), acting on the Game Flow model, converges to a Nash solution that generally improves the value of the payoff function.

The Game Flow model simulates a two-force game where the assets of each force, say the blue or red forces, are distributed over a large geographical area.

In this experiment, the game area was a square divided into 64 square cells. At the start of the game, the two forces were spread uniformly over the entire game area, but the total strength of the blue force was only two thirds the total strength of the red force. To counter this mismatch, the attack range of the blue force was larger, and the cost of movement for the blue force was lower than that of the red force.

The goal of each force was to reach the end of the game with a minimum loss of their own strength, while inflicting maximum damage to the opposing force. Also, each force assigned more value (larger weight) to the cells located in the middle of the game area than to the cells located near the boundaries, so higher score might be earned by finishing the game with heavier strength concentration in more valuable cells. Finally, movement of assets across the game area was penalized, so economy of movement was also reflected in the final score of each force.

The game was carried out for a specified amount of time, with the phases of the game, i.e., asset movement and attack, evolving uninterrupted for the duration of the game.

The SLQ algorithm was used to find a Nash equilibrium solution for the game. In this experiment, the solver was stopped after 10 iterations, when the error (i.e., the norm of the velocity updates) was approximately one percent of the original error. At this error level, further iterations had an insignificant effect on the solution.

Experimental results show that the Nash equilibrium solution found by the SLQ algorithm, greatly improved the performance of the two forces with respect to the value of the payoff function selected for this experiment.

Qualitatively speaking, we can say that, in this scenario, the superiority of the blue force in the attack range, and its lower cost on movement prevailed, allowing the blue force to keep the red force out of the most valuable cells in the middle of the game area.

Experiment 13: Discrete Platform Dynamics

In any type of battlefield, the loss of platforms is usually a stochastic discrete event over time. In JFACC simulations, however, the number of platforms has been modeled as a real number representing its probabilistic expectation. In other words, our game-theoretic controller based on an expected-value model needs to be tested on a more realistic plant, in which the numbers of platforms are integers. Our approach was to first develop a model such that the dynamics of the number of platforms is a stochastic discrete-event equation, i.e., in our new stochastic discrete-event model, the number of platforms in each unit is an integer. Hence, the number of platforms changes from 10 to 9 at one point and then on to eight platforms later based on the probability of kill. Moreover, the numbers of platforms vary differently for different runs due to random number generators, which control the time when an actual kill occurs. Using this new model, we conducted multiple runs and took an average. This average was then compared against the results based on the expected-value model. We concluded that our game-theoretical controller (based on the simpler expected-value model) performed just as well when tested on this more realistic stochastic discrete-event plant model as when tested on the expected-value plant model.

Experiment 14: Non-linear Detector for the Fully Non-linear Model

In Chapters 9 and 10 we have reported the results of experiments performed to test the effectiveness of a “game-theoretic-optimal” detection filter to process noise-corrupted observations of the battlefield. In those series of experiments, a bilinear approximation of the non-linear model of the battlefield was considered and the filter was designed accordingly. When the fully non-linear model of the battlefield is considered, a different (non-linear) detection filter must be designed. The purpose of this Chapter is to present the experimental results concerning the non-linear filter and to compare them with those obtained by using the detection filter designed on the basis of the bilinear model of the battlefield. For the sake of simplicity, the case of noise-free measurements will be considered in this series of experiments.

Experiment 15: Comparison with Honeywell’s Results

A comparison of the platform loss and probability of success values is made between Washington University and Honeywell results on two example missions, each consisting of three sorties. The results are similar in the first example. Due to a change in the initial number of Red fighters and their probability of kill, the outcome of the second example is drastically different. It has also been observed that the selection of weights in the cost function may affect the unit trajectories and platform loss significantly.

Despite running our Sequential Linear-Quadratic Method for 50 iterations or more, convergence to a possible Nash solution was not achieved in either example, although the obtained unit trajectories and platform loss numbers were reasonable, given the mission objectives.

Experiment 16: Controller Computational Complexity: Correction

The *purpose* of Experiment 16 is to correct an error present in the subprogram that evaluates the Jacobian of the model MDCM. This error would have affected the results in cases in which multiple units are deployed against multiple units and some units are not fired upon. This error affects only one such case in the Interim Report (experiments 1 through 12), that is experiment 5.3.2. Therefore, a corrected version of the subprogram for computing the Jacobian has been developed, and corrected computational results are reported in this chapter. Even with this change we can draw the same conclusions as in Experiment 5; namely, the computational time is a quadratic function of the number of units.

Experiment 17: Controller with a Kalman Filter for Estimation

In this chapter, we present how an algorithm based on the Extended Kalman Filter (EKF) for state estimation is used in a differential game, which models the air operations of two opposing forces. We show the overall structure of the game in a block diagram. We present the implementation of the algorithm in a flowchart. We also present simulation results.

In an air operation game, it is reasonable to assume that one does not get direct information about his enemy’s input. In this paper, we present an approach for estimating the states of the friendly as well as enemy forces and compare their respective simulation results. The Kalman filter due to Darouach et al. treats the enemy inputs as part of the extended state and obtains an estimate of both the state of the two forces and the input of the enemy. But their filter is designed for linear time-invariant systems. Hence, we present an extension of their filter to a nonlinear time-variant system.

The extended Kalman filter algorithm presented in this report is capable of estimating the states of both forces in the presence of process noise as well as sensor noise. We note that the estimates of the enemy inputs are too noisy to be directly useful. However, our game-theoretic controller requires only an

estimate of the enemy state and it does not require any estimates of the enemy input. We thus observed the game-theoretic controller remained effective when the extended Kalman filter is introduced in the loop.

Experiment 18: Method of Characteristics: Addendum

The purpose of Experiment 11 was to verify that the solution computed by the Sequential Linear-Quadratic Method (SLQM) was the same as the Nash solution computed by the Method of Characteristics. We verified that the solutions computed by the Sequential Linear-Quadratic Method (SLQM) were indeed the same as the Nash solutions computed by the Method of Characteristics under several scenarios. However, the experiments in Chapter 11 all involved one Blue unit against one Red unit. In Experiment 18, we extend the results in Experiment 11 to a scenario of multi-units against multi-units. Specifically, Experiment 18 tests the Method of Characteristics for the case of three blue units against three red units.

Experiment 19: New Game Flow Models

Military operations can be viewed as a hierarchical structure in which actions are taken by individual units at a low level, based on strategies developed by planners at a high level. In this experiment we consider the situation in which two forces, say the blue and red forces, control a large number of units distributed over a large geographical area. We develop a tool that is useful to high-level planners in simulating and computing the optimal strategy for the two forces. We also report the results of our numerical experiments.

The geographical area in our model is represented by an abstract game board that is divided into cells so that the strength concentration of the blue (resp. red) force in a cell is defined as the number of blue (resp. red) units contained in the cell divided by the area of the cell. The game is concurrent in the sense that both the blue and red forces can move some or all of their respective units simultaneously and continuously during the game.

We formulated the military operation control problem as a differential game over the abstract game board. The differential game consists of a quadratic payoff function and a set of ordinary differential equations describing the system dynamics of the unit distribution over the discretized geographical area (the abstract game board).

In order to solve such a geographically distributed differential game, we developed a computer method for finding a local Nash solution to the adversarial game. The optimum strategy for each team is found using the iterative algorithm called Sequential Linear-Quadratic Method. Experimental results are also presented that demonstrate the validity of this concept.

Chapter 1

Experiment 1: Combat Modeling and Validation

1.1 Executive Summary

The purpose of this experiment is to validate the low-order ordinary differential equation (ODE) models, which are derived to approximate the evolution of expected values in a more realistic hybrid-stochastic model, called the Probabilistic Mission Dynamics Model (PMDM), under different assumptions for target acquisition and target selection coordination.

The hypothesis is that the evolution of the expected values of the Markov chain (MC) mission dynamics can be approximated by a low-order ordinary differential equation (ODE) model, for a time period of sufficient duration, when the control signals are generated in an open-loop setting.

This study first identifies four different sets of assumptions about target acquisition and target selection coordination, which will be abbreviated as:

MARI (Acquisition Rate Independent) Uncoordinated target selection, independent target acquisition,

MARD (Acquisition Rate Dependent) Uncoordinated target selection, linear target acquisition,

MNWA (No Wrap-around) Coordinated target selection, without wrap-around,

MWWA (With Wrap-around) Coordinated target selection, with wrap-around.

Then, continuous-transition Markov chain (MC) models are developed under these assumptions. Using the MC models, the probability distributions for the number of platforms and their exact expected values η^B, η^R are calculated as a function of time. Next, approximate ODE models of the evolution of the expected values are derived.

The trajectories of the ODE models ($\hat{\eta}^B, \hat{\eta}^R$) are compared with the exact outcomes (η^B, η^R) in twelve experiments, summarized in the table below, which indicates the initial number of platforms and the probability of kill for the Blue and Red units.

Scenarios Used In the Experiments	
Scenario Types; $N^B = 8, P_k^B = 0.8$	
A	$N^B = N^R, P_k^B = P_k^R$
B	$N^B = 2N^R, P_k^B = P_k^R$
C	$N^B = N^R, P_k^B = 2P_k^R$

The approximation quality of the ODE models is compared in the following table using the L_2 norm:

$L_2[0, 10]$ -Norm of Error Between Actual and Approximate Expected Values

Exp. #	1	2	3	4	5	6	7	8	9	10	11	12
	MARI			MARD			MNWA			MWWA		
$\ \eta^B - \hat{\eta}^B\ $	0.14	0.37	0.01	0.06	0.03	0.02	1.35	5.80	0.02	1.53	4.74	0.16
$\ \eta^R - \hat{\eta}^R\ $	0.14	1.35	0.25	0.06	0.06	0.04	1.35	5.80	0.04	1.53	0.06	0.58
$\ \eta^B - \hat{\eta}^B + \eta^R - \hat{\eta}^R\ $	0.28	1.72	0.26	0.12	0.09	0.06	2.70	11.60	0.06	3.06	4.80	0.74

It was observed that the ODE models were good approximations under the uncoordinated target selection assumption, and were found to be sufficient to represent the attrition dynamics in a differential game setup in this case. The discrepancy between the MC and ODE models increase as the engagement proceeds, which should be expected. Under the coordinated target selection assumption, the ODE approximations were worse. This can be partially explained by the fact that coordination implies firing in rounds, and therefore platform loss is more discrete in nature.

1.2 Purpose of the Experiment

The purpose of this experiment is to validate the low-order ordinary differential equation (ODE) models, which are derived to approximate the evolution of expected values in a more realistic hybrid-stochastic model, called the Probabilistic Mission Dynamics Model (PMDM), under different assumptions for target acquisition and target selection coordination.

1.3 Hypothesis to Prove or Disprove

The hypothesis is that the evolution of the expected values of the Markov chain (MC) mission dynamics can be approximated by a low-order ordinary differential equation (ODE) model, for a time period of sufficient duration, when the control signals are generated in an open-loop setting. For the closed-loop situation, see Chapter 7.

1.4 Introduction

We consider a geographical area, a theater of air operations, in which two forces oppose each other and try to accomplish their respective mutually conflicting air missions. For example, two forces may be operating in an area, in which the ground force of one side tries to invade the other side while the air force of the other side tries to stop the invasion.

Attrition dynamics are inherently probabilistic, yet most attrition models approximate the dynamics with a deterministic system. This chapter shows the predictive capabilities of a probabilistic model using Markov Chains, and illustrates different methods to approximate the probabilistic model with a deterministic model.

In this chapter we study the combat between one Blue unit and one Red unit. Modeling multiple units is more involved and may be the subject of future work. (A multi-unit model, based on heuristic arguments, is included as an appendix to this chapter.) Each unit is homogeneous, that is, each unit consists of only one type of platform equipped with the same type of weapons. Platforms can be SEADS, ground troops, bombers, interceptors, etc. In the scenario we would have data consisting of such ideas as the probability of destroying a targeted platform, firing intensities and speed controls. Firing intensity is a control which the mission commander (controller) may change, in a dynamic game situation.

The model outcome is mainly concerned with the expected values of the number platforms for each unit at the end of a mission. We have presented approximations to these expected values that are deterministic ODE's. Modeling has two different cases: uncoordinated target selection (any platform that acquires a target may fire a weapon) and coordinated target selection (the unit commander controls

how the platforms target). We will see that some of these approximations are very good, and some are not.

The chapter is organized as follows: a presentation of the general background to the model, a derivation of the model for uncoordinated target selection, a derivation of the model for coordinated target selection, the evolution of the expected number of platforms for both uncoordinated and coordinated cases with their approximations, numerical experiments, and some ideas of what to do next. An abridged version has been published as [1].

1.4.1 Notation

We will denote the quantities which belong to the forces Blue and Red with superscripts $.^B$ and $.^R$. If the symbols in an expression do not have superscripts, this is intended to mean that the expression is valid for either force.

The assumptions which appear to be reasonable for the situation we are modeling will be preceded by **M**... Simplifying assumptions, which are introduced for the sake of mathematical convenience, but may not always hold, will be preceded with **S**... Postulates, which we believe to reflect the “nature of things”, and should be satisfied with rare exceptions, will be preceded with **P**...

1.5 Random Variables

The position of the units are ξ^B and ξ^R , which are deterministic quantities. On the other hand, the number of platforms will depend on chance occurrences. To capture this, define the random variables,

$$\begin{aligned} X^B(t) &\stackrel{\text{def}}{=} \text{number of platforms in the Blue unit at time } t, \\ X^R(t) &\stackrel{\text{def}}{=} \text{number of platforms in the Red unit at time } t. \end{aligned}$$

The initial values are known to be $X^B(0) = N^B$, $X^R(0) = N^R$. At any given time, the commanders (or controllers) can observe only the expected values, denoted by

$$\eta^B(t) \stackrel{\text{def}}{=} E[X^B(t)], \eta^R(t) \stackrel{\text{def}}{=} E[X^R(t)], \eta \stackrel{\text{def}}{=} [\eta^B, \eta^R]^T.$$

Each unit has a fire intensity control $\nu \in [0, 1]$. This can be interpreted as the frequency of firing a weapon per target acquisition (i.e., the probability of firing, given a target is acquired).

After acquiring a target, a platform fires a salvo of s weapons (with probability ν). This will decrease the weapons load of this platform. Define the random variables,

$$\begin{aligned} W^B(t) &\stackrel{\text{def}}{=} \text{number of salvo loads per platform in the Blue unit at time } t, \\ W^R(t) &\stackrel{\text{def}}{=} \text{number of salvo loads per platform in the Red unit at time } t. \end{aligned}$$

Again, at any given time, the commanders (or controllers) can observe only the expected values,

$$\zeta^B(t) \stackrel{\text{def}}{=} E[W^B(t)], \zeta^R(t) \stackrel{\text{def}}{=} E[W^R(t)], \zeta \stackrel{\text{def}}{=} [\zeta^B, \zeta^R]^T.$$

Therefore the fire intensity controls are deterministic quantities, $\nu = \nu(t, \xi, \eta, \zeta)$.

Consider the Blue unit firing on the Red unit. Let the probability of kill for each weapon, given it is fired, be $P\{\text{wkill}^B|\text{fired}^B\}$. The probability of killing the target, with a salvo of s^B weapons, given that they are fired simultaneously, is

$$P_k^B = P\{\text{kill}^B|\text{fired}^B\} = \begin{cases} [1 - (1 - P\{\text{wkill}^B|\text{fired}^B\})^{s^B}] & \text{if } W^B(t) > 0, \\ 0 & \text{if } W^B(t) = 0. \end{cases} \quad (1.1)$$

1.6 Uncoordinated Target Selection

Consider two (homogeneous) units of the opposing forces engaged in a battle, in which platforms of both sides are shooting at each other simultaneously. During this engagement, each platform searches for an enemy platform (in the sky, on the ground, etc.). When a platform is located in space, identified to be an enemy and the weapon system is locked on to this platform, a target is said to be acquired. Target acquisition is a stochastic process, in which events occurring on disjoint intervals of time are assumed to be independent.

The following assumptions appear to be reasonable for air-to-air or air-to-ground combat with modern, electronically guided weapon systems:

MNCO: Friendly platforms do not communicate for target selection (uncoordinated selection). The exception is when a platform, which has depleted its supply of weapons, locates and identifies a target. In that case, this platform will relay this information to a friendly platform in the same unit which has weapons.

MPKD: The probability of killing the target depends on the distance between the units.

MNWT: The time it takes for missiles, bombs, and other weapons to reach the target is negligible.

From **MPKD**, (1.1) will depend on the distance between units, with a function $\psi : \mathbb{R} \rightarrow [0, 1]$ which depends on the positions of each unit, $\psi^B(\|\xi^B - \xi^R\|)$. Therefore the probability of a platform killing its target, given it is assigned to a target, is

$$\begin{aligned} P_k^B \psi^B \nu^B &\sim \text{probability of a blue platform killing a red platform with one salvo,} \\ P_k^R \psi^R \nu^R &\sim \text{probability of a red platform killing a blue platform with one salvo.} \end{aligned}$$

There are two different situations for acquisition rate and self-attribution:

MARI: The target acquisition rate does not depend on the number of enemy platforms or their distribution in space. (Search devices are advanced enough that they will locate enemy platforms efficiently even when they are distributed sparsely.)

MNSA: Self-attribution or equipment breakdowns are negligible.

and

MARD: The target acquisition rate does depend on the number of enemy platforms.

MWSA: Self-attribution or equipment breakdowns are not negligible.

To satisfy both **MARI** and **MARD**, considering a Blue platform, define a function, $\sigma : \mathbb{N} \rightarrow \mathbb{R}$, as

$$\sigma^B(m) = \begin{cases} 1 & m \geq \sigma_0^B, \\ \frac{m}{\sigma_0^B} & m < \sigma_0^B \end{cases} \quad (1.2)$$

where σ_0^B is the value at which the Blue platform's acquisition rate saturates. If $\sigma_0^B = 1$, we have an acquisition rate that is independent of the number of enemy platforms (**MARI**), otherwise there is linear dependence on the number of enemy platforms (**MARD**), until saturation is reached.

Now, consider the Blue unit firing at the Red unit. Let α^B be the maximum rate a Blue platform acquires a Red platform as a target. Given $X^B(t) = n$ and $X^R(t) = m$, from **MPKD** with Equations (1.1) and (1.2), the loss rate of Red platforms due to one Blue platform's fire is

$$\sigma^B(m) \lambda^R, \text{ with } \lambda^R \stackrel{\text{def}}{=} P_k^B \psi^B(\|\xi^B - \xi^R\|) \alpha^B \nu^B.$$

From **MNCO**, the loss rate for the Red unit due to all platforms of the Blue unit is $\sigma^B(m) \lambda^R(t) n$. The loss rate for the Blue unit is $\sigma^R(n) \lambda^B(t) m$, from symmetry.

Next, when self-attribution is not negligible, **MWSA**, we may define an independent process describing the self-attribution rate for a platform, β^B for the Blue unit platforms and β^R for the Red unit platforms. Then the self-attribution rate due to all platforms in the Blue unit is $n\beta^B$, and the rate due to all platforms in the Red unit is $m\beta^R$. When self-attribution is negligible, **MNSA**, then $\beta = 0$.

1.6.1 Defining the Markov Chain

Under the previous assumptions, we can postulate a two-dimensional non-homogeneous continuous-time Markov Chain for the platform dynamics, with state space $\{0, \dots, N^B\} \times \{0, \dots, N^R\}$, by specifying the state transition probabilities from time t to time $t + h$, where h is small:

PDI: The number of losses in disjoint intervals are independent.

PKB: $P\{\text{exactly one Blue killed}\} =$

$$P\{X^B(t+h) = n-1, X^R(t+h) = m \mid X^B(t) = n, X^R(t) = m\} = (m\sigma^R(n)\lambda^B(t) + n\beta^B)h + o(h)$$

PKR: $P\{\text{exactly one Red killed}\} =$

$$P\{X^B(t+h) = n, X^R(t+h) = m-1 \mid X^B(t) = n, X^R(t) = m\} = (n\sigma^B(m)\lambda^R(t) + m\beta^R)h + o(h)$$

PMD: $P\{\text{two or more deaths}\} =$

$$\begin{cases} o(h) & n+m \geq 2 \\ 0 & \text{otherwise} \end{cases}$$

PRE: $P\{\text{resurrection}\} =$

$$P\{X^B(t+h) > n \cup X^R(t+h) > m \mid X^B(t) = n, X^R(t) = m\} = 0$$

PND: It follows that $P\{\text{no death}\} =$

$$P\{X^B(t+h) = n, X^R(t+h) = m \mid X^B(t) = n, X^R(t) = m\} = 1 - (m\sigma^R(n)\lambda^B(t) + n\beta^B + n\sigma^B(m)\lambda^R(t) + m\beta^R)h + o(h).$$

Using the standard notation

$$\Pi_{n,m}(t) \stackrel{\text{def}}{=} P\{X^B(t) = n, X^R(t) = m\}, \quad (1.3)$$

the evolution of state probabilities are described by

$$\begin{aligned} \frac{d}{dt}\Pi_{n,m}(t) = & - (m\sigma^R(n)\lambda^B + n\beta^B + n\sigma^B(m)\lambda^R + m\beta^R)\Pi_{n,m} \\ & + (m\sigma^R(n+1)\lambda^B + (n+1)\beta^B)\Pi_{n+1,m} \\ & + (n\sigma^B(m+1)\lambda^R + (m+1)\beta^R)\Pi_{n,m+1} \end{aligned}$$

where the time arguments have been dropped for brevity. If we stack all components of $\Pi_{n,m}$ into a row vector Π , with $(N^B + 1)(N^R + 1)$ elements, the above differential equation (1.4) can be written

$$\frac{d}{dt}\Pi(t) = \Pi(t)Q(t) \quad (1.4)$$

where Q is called the transition rate matrix or the infinitesimal generator of the process.

1.7 Coordinated Target Selection

In this case, one can no longer talk about a target acquisition process. In order to decide which enemy platforms will be a target, the unit commander will need to know the number and type of platforms in the enemy unit. Target selection coordination implies less independence for each platform. Now the platform commanders are only executing strict orders, or what their training dictates, when their unit engages an enemy unit.

Coordination also implies that target selection takes place in rounds, from the unit commander's perspective.

MWCO At the beginning of each round the unit commander assigns each of his platforms an enemy platform as a target. During the round firing takes place, and platforms may be killed on both sides. Then the unit commander assigns targets for the next round.

There are two different situations:

MRTI: The inter-arrival time between rounds (round length) is independent of the number of platforms, and independent of the previous round length. Both sides fire simultaneously.

MRTK: A new round begins only when a friendly or enemy platform is killed.

MRTI is appropriate for artillery duels, for bombers versus ground troops or bombers versus air defense. **MRTK** may be appropriate for air-to-air combat. In both cases, the round length is the same for both Blue and Red. We focus only on **MRTI** in this report.

There are two situations for target selection: with wrap-around and without wrap-around;

MNWA: (No wrap around) At the beginning of a round, each platform is assigned to a unique target. If there are more platforms than targets, the remaining platforms do not participate in the firing (although they may be targeted by the enemy).

MWWA: (With wrap around) At the beginning of a round, each platform is assigned to a unique target, until all platforms have assignments. If there are more platforms than targets, the end of the target list wraps around to the beginning.

As in the uncoordinated case we have,

$$\begin{aligned} P_k^B \psi^B \nu^B &\sim \text{probability of a blue platform killing a red platform with one salvo,} \\ P_k^R \psi^R \nu^R &\sim \text{probability of a red platform killing a blue platform with one salvo.} \end{aligned}$$

The time between rounds is needed for the platforms to reload their weapons, for bombers to turn back, for determination of losses and kills, and for decision making for the assignments. From **MRTI**, arrival of rounds is a Poisson process with parameter ρ (which may depend on the unit category).

1.7.1 Case 1 : Independent Round Arrival, No Wrap-Around

Suppose a round occurs in $(t, t + h]$. Given $X^B(t) = n$ and $X^R(t) = m$, there will be $\min(n, m)$ target assignments on each side, and $\min(n, m)\nu$ shots will be taken. Consider the number of losses on the Red side. Since each round of shots taken is independent, this can be regarded as a Bernoulli trial with probability of success, $P_k^B \psi^B \nu^B$. Thus the probability that Red will lose k platforms, given the round started at time t , is

$$\begin{aligned} G^R(n, m, k, t) &\stackrel{\text{def}}{=} P\{X^R(t+h) = m - k \mid X^B(t) = n, X^R(t) = m, \text{ one round in } (t, t+h]\} \\ &= \binom{\min(n, m)}{k} (P_k^B \psi^B \nu^B)^k (1 - P_k^B \psi^B \nu^B)^{\min(n, m) - k}. \end{aligned} \quad (1.5)$$

Note that this is valid only when $k \leq \min(n, m)$, so one should define

$$\binom{a}{b} \stackrel{\text{def}}{=} \begin{cases} \frac{a!}{b!(a-b)!} & a \geq b, \\ 0 & a < b. \end{cases}$$

Similarly for Blue losing l platforms, given the round started at time t ,

$$\begin{aligned} G^B(n, m, l, t) &\stackrel{\text{def}}{=} P\{X^B(t+h) = n-l \mid X^B(t) = n, X^R(t) = m, \text{ one round in } (t, t+h)\} \\ &= \binom{\min(n, m)}{l} (P_k^R \psi^R \nu^R)^l (1 - P_k^R \psi^R \nu^R)^{\min(n, m)-l}. \end{aligned} \quad (1.6)$$

Since both sides fire simultaneously, the number of losses on one side is independent of the number of losses on the other side. Thus for $(l, k) \neq (0, 0)$

$$P\{X^B(t+h) = n-l, X^R(t+h) = m-k \mid X^B(t) = n, X^R(t) = m, \text{ one round in } (t, t+h)\} = G^B(n, m, l, t) G^R(n, m, k, t).$$

1.7.2 Case 2 : Independent Round Arrival, With Wrap-Around

Suppose a round occurs in $(t, t+h]$. Given $X^B(t) = n$ and $X^R(t) = m$. The minimum number of platforms assigned to fire at a Red platform by the Blue unit is

$$f^B = \begin{cases} \lfloor \frac{n}{m} \rfloor, & m > 0, \\ 0, & m = 0, \end{cases}$$

where $\lfloor \bullet \rfloor$ is the truncation function. Out of the n Blue platforms, mf^B of them will be assigned regularly, and

$$n \bmod m = n - mf^B$$

of them will be assigned as extras (the wrap-around). Then $n - mf^B$ Red platforms will receive $f^B + 1$ shots and $m(f^B + 1) - n$ Red platforms will receive f^B shots, given that all Blue platforms choose to fire.

The number of Red losses, given a round in $(t, t+h]$, is the sum of Red losses in the group which receive f^B shots, plus the Red losses in the group which receive $f^B + 1$ shots. The ' f^B ' group has $m(f^B + 1) - n \stackrel{\text{def}}{=} b_0$ Bernoulli trials, each with probability of success

$$B_0 \stackrel{\text{def}}{=} 1 - (1 - P_k^B \psi^B \nu^B)^{f^B}.$$

The ' $f^B + 1$ ' group has $n - mf^B \stackrel{\text{def}}{=} b_1$ Bernoulli trials, each with probability of success

$$B_1 \stackrel{\text{def}}{=} 1 - (1 - P_k^B \psi^B \nu^B)^{f^B+1}$$

Thus the probability that Red will lose k platforms given the round started at time t is,

$$G^R(n, m, k, t) \stackrel{\text{def}}{=} P\{X^R(t+h) = m-k \mid X^B(t) = n, X^R(t) = m, \text{ one round in } (t, t+h)\} = \begin{cases} \sum_{q=0}^k \binom{b_0}{q} (B_0)^q (1 - B_0)^{b_0-q} \binom{b_1}{k-q} (B_1)^{k-q} (1 - B_1)^{b_1-(k-q)} & f^B \neq 0 \\ \binom{n}{k} (B_1)^k (1 - B_1)^{n-k} & f^B = 0 \end{cases} \quad (1.7)$$

A similar argument holds for Blue to find $G^B(n, m, l, t)$. Since both sides fire simultaneously, the number of losses on one side is independent of the number of losses on the other side. Thus with $(l, k) \neq (0, 0)$,

$$P\{X^B(t+h) = n-l, X^R(t+h) = m-k \mid X^B(t) = n, X^R(t) = m, \text{ one round in } (t, t+h)\} = G^B(n, m, l, t) G^R(n, m, k, t).$$

1.7.3 Defining the Markov Chain

First set $F(l, k, n, m, t) = G^B(n, m, l, t)G^R(n, m, k, t)$, where $G^B(n, m, l, t)$ and $G^R(n, m, k, t)$ are defined in Equations (1.5), (1.6) and (1.7) for both cases. Under the previous assumptions, we can postulate a two-dimensional non-homogeneous continuous-time Markov chain for the platform dynamics, with state space $\{0, \dots, N^B\} \times \{0, \dots, N^R\}$, by specifying the state transition probabilities from time t to time $t+h$, where h is small:

PDI: The number of losses in disjoint intervals are independent.

PKBR: $(l, k) \neq (0, 0)$

$$P\{X^B(t+h) = n-l, X^R(t+h) = m-k \mid X^B(t) = n, X^R(t) = m\} = F(l, k, n, m, t)(\rho h + o(h))$$

PND: $(l, k) = (0, 0)$

$$\begin{aligned} P\{X^B(t+h) = n, X^R(t+h) = m \mid X^B(t) = n, X^R(t) = m\} = \\ (1 - \rho h + o(h)) + F(0, 0, n, m, t)(\rho h + o(h)) \\ + P\{\text{losing } l \text{ and } k \text{ in } 2 \text{ or more rounds}\} \underbrace{P\{2 \text{ or more rounds occur}\}}_{o(h)} \end{aligned}$$

Using the standard notation (1.3), the evolution of the state probabilities are described by

$$\begin{aligned} \dot{\Pi}_{n,m}(t) &= (F(0, 0, n, m, t) - 1)\rho\Pi_{n,m} \\ &+ \sum_{(l,k) \neq (0,0)} F(l, k, n+l, m+k, t)\rho\Pi_{n+l, m+k} \end{aligned} \quad (1.8)$$

where the time arguments have been dropped for brevity. If we stack all components of $\Pi_{n,m}$ into a row vector Π , with $(N^B + 1)(N^R + 1)$ elements, we have Equation (1.4).

1.8 Evolution of Expected Values

1.8.1 Uncoordinated Target Selection

Independent Target Acquisition (MNCO, MARI)

With independent target acquisition, we have $\sigma = 1$. Under this assumption we may write the evolution of the expected number of Blue platforms as

$$\begin{aligned} \dot{\eta}^B(t) &= \frac{d}{dt} \sum_{n=0}^{N^B} \sum_{m=0}^{N^R} n\Pi_{n,m}(t) \\ &= -\lambda^B(t)\eta^R(t) + \lambda^B(t) \sum_{m=1}^{N^R} m\Pi_{0,m}(t). \end{aligned} \quad (1.9)$$

Similarly, the evolution of the expected number of Red platforms is

$$\begin{aligned} \dot{\eta}^R(t) &= \frac{d}{dt} \sum_{n=0}^{N^B} \sum_{m=0}^{N^R} m\Pi_{n,m}(t) \\ &= -\lambda^R(t)\eta^B(t) + \lambda^R(t) \sum_{n=1}^{N^B} n\Pi_{n,0}(t). \end{aligned} \quad (1.10)$$

A way of developing an approximation of the expected values is to drop the extra terms in (1.9) and (1.10). This will give us a deterministic ODE approximating the expected value of platforms,

$$\dot{\hat{\eta}}^B = -\hat{\eta}^R P_k^R \psi^R \nu^R \alpha^R, \quad (1.11)$$

$$\dot{\hat{\eta}}^R = -\hat{\eta}^B P_k^B \psi^B \nu^B \alpha^B. \quad (1.12)$$

Dependent Target Acquisition (MNCO, MARD)

Consider a short interval, $[t, t + \Delta t]$. We then may assume that the number of Blue losses in the interval is approximated by

$$m\sigma^R(n)\lambda^B \approx \hat{\eta}^R \sigma^R(\hat{\eta}^B) \lambda^B \Delta t = \hat{\eta}^R \hat{\eta}^B \frac{1}{\sigma_0^R} \lambda^B \Delta t.$$

Then we may approximate

$$\hat{\eta}^B(t + \Delta t) = \hat{\eta}^B(t) - (\hat{\eta}^R \hat{\eta}^B \frac{1}{\sigma_0^R} \lambda^B \Delta t),$$

and by taking the proper limit as $\Delta t \rightarrow 0$, we get

$$\dot{\hat{\eta}}^B = -\hat{\eta}^B \hat{\eta}^R \frac{\alpha^R}{\sigma_0^R} P_k^R \psi^R \nu^R. \quad (1.13)$$

Similarly we have

$$\dot{\hat{\eta}}^R = -\hat{\eta}^R \hat{\eta}^B \frac{\alpha^B}{\sigma_0^B} P_k^B \psi^B \nu^B. \quad (1.14)$$

1.8.2 Coordinated Target Selection

No Wrap-Around (MWCO, MRTI, MNWA)

In each round, both sides have $\min(\eta^B, \eta^R)$ expected shots, so we would expect Red to lose

$$\min(\eta^B, \eta^R) P_k^B \psi^B \nu^B,$$

and expect Blue to lose

$$\min(\eta^B, \eta^R) P_k^R \psi^R \nu^R,$$

in each round. In a short interval, Δt , the expected number of rounds is $\rho \Delta t$. Assuming that during these rounds η^B and η^R do not change much (this assumption will not necessarily hold), we may approximate

$$\hat{\eta}^B(t + \Delta t) = \hat{\eta}^B(t) - \rho \Delta t \min(\hat{\eta}^B, \hat{\eta}^R) P_k^R \psi^R \nu^R.$$

Then by taking the proper limit as $\Delta t \rightarrow 0$, we obtain the evolution equation for the expected number of Blue platforms,

$$\dot{\hat{\eta}}^B(t) = -\rho \min(\hat{\eta}^B, \hat{\eta}^R) P_k^R \psi^R \nu^R.$$

If it is desired to replace $\min(\eta^B, \eta^R)$ by a smooth function, one possibility is to use

$$\frac{\min(\hat{\eta}^B, \hat{\eta}^R)}{\hat{\eta}^B} = \min\left(\frac{\hat{\eta}^R}{\hat{\eta}^B}, 1\right) \approx 1 - e^{-2\frac{\hat{\eta}^R}{\hat{\eta}^B}}.$$

Then we obtain

$$\dot{\hat{\eta}}^B(t) = -\rho \hat{\eta}^B \left[1 - e^{-2\frac{\hat{\eta}^R}{\hat{\eta}^B}} \right] P_k^R \psi^R \nu^R. \quad (1.15)$$

Similarly, by symmetry, we have

$$\dot{\hat{\eta}}^R(t) = -\rho \hat{\eta}^R \left[1 - e^{-2\frac{\hat{\eta}^B}{\hat{\eta}^R}} \right] P_k^B \psi^B \nu^B. \quad (1.16)$$

However, it was observed that this replacement results in poor approximations when one force has numerical superiority.

With Wrap-Around (MNCO, MRTI, MWVA)

Consider the Blue unit firing on the Red unit. The average number of shots Red will receive per platform is $\frac{n}{m}$. In each round, the expected Red loss is the sum of the expected losses in the ' f^B ' and ' $f^B + 1$ ' groups, which is

$$[m(f^B + 1) - n][1 - (1 - P_k^B \psi^B \nu^B)^{f^B}] + [n - mf^B][1 - (1 - P_k^B \psi^B \nu^B)^{f^B+1}]$$

Approximating f^B by $\frac{n}{m}$ (the average number of shots Red will receive), the above equation simplifies to $m[1 - (1 - P_k^B \psi^B \nu^B)^{\frac{n}{m}}]$. Following the same logic as in the first case, we obtain

$$\dot{\hat{\eta}}^R(t) = -\rho \hat{\eta}^R \left[1 - (1 - P_k^B \psi^B \nu^B)^{\frac{n}{\hat{\eta}^R}} \right]. \quad (1.17)$$

Similarly, by symmetry, we have

$$\dot{\hat{\eta}}^B(t) = -\rho \hat{\eta}^B \left[1 - (1 - P_k^R \psi^R \nu^R)^{\frac{n}{\hat{\eta}^B}} \right]. \quad (1.18)$$

1.8.3 Summary of ODE's

Table 1.1 summarizes the ODE's for approximating the expected value of platforms for each model.

Table 1.1: The Approximate Evolution of Expected Number of Platforms

Case	Approximate Expected Value ODE's
Uncoordinated(MARI)	$\dot{\hat{\eta}}^B = -\hat{\eta}^R P_k^R \psi^R \nu^R \alpha^R$
Uncoordinated(MARD)	$\dot{\hat{\eta}}^B = -\hat{\eta}^B \hat{\eta}^R \frac{\alpha^R}{\sigma_0^R} P_k^R \psi^R \nu^R$
Coordinated(MNVA)	$\dot{\hat{\eta}}^B = -\rho \hat{\eta}^B \min(\hat{\eta}^B, \hat{\eta}^R) P_k^R \psi^R \nu^R$
Coordinated(MWVA)	$\dot{\hat{\eta}}^B = -\rho \hat{\eta}^B \left[1 - (1 - P_k^R \psi^R \nu^R)^{\frac{n}{\hat{\eta}^B}} \right]$

1.9 Weapons Expenditure

When a platform with a large number of weapons is shot down, the salvo loads per platform $W(t)$ will decrease faster, compared to the case when a platform with less weapons is shot down. Therefore, for an exact prediction of the probability distribution of $W(t)$ one needs to develop a Markov chain model for the number of weapons in each platform individually, and then calculate the distribution of $W(t)$ from these. This is not only computationally very costly, but also only marginally useful for the overall project objective of developing a game theoretic controller. This is because $W(t)$ does not enter the cost function, and it becomes significant, per (1.1), only when a unit depletes all of its weapons. Considering this, we will take a shortcut by introducing the following simplifying assumption:

SWES: Since the target acquisition or round arrival rate, the fire intensity controls and the salvo sizes are the same for all platforms in a unit, all platforms will be assumed to have the same number of weapons on board at any given time.

Because of **SWES**, when a platform is killed, $W(t)$ does not change. The salvo loads per platform of the unit obey, for small Δt

$$W(t + h) - W(t) = \text{number of salvos fired in } [t, t + \Delta t]. \quad (1.19)$$

1.9.1 Uncoordinated Target Selection

For both the independent target acquisition case (**MNCO,MARI**), and the linearly dependent case (**MNCO,MARD**), the firing rate for a Blue platform will be $\sigma^B(X^R)\alpha^B\nu^B$. The expected number of salvos fired per Blue platform in $[t, t + \Delta t]$ will be $\sigma^B(\eta^R)\alpha^B\nu^B\Delta t$. Taking expected values in (1.19), dividing by Δt and taking the limit $\Delta t \rightarrow 0$ yields

$$\dot{\zeta}^B(t) = -\sigma^B(\eta^R(t))\alpha^B\nu^B(t). \quad (1.20)$$

A similar equation holds for the Red unit.

1.9.2 Coordinated Target Selection

For the case without wrap-around (**MWCO,MRTI,MNWA**), the assumption **SWES** may not hold for a few rounds. In the long run, if the platforms which are selected to fire on the enemy are rotated regularly between rounds, one can still employ this assumption. In that case, the Blue unit will fire, on average,

$$\frac{\min(\eta^B, \eta^R)}{\eta^B} \nu^B = \min\left(\frac{\eta^R}{\eta^B}, 1\right) \nu^B$$

shots per platform, in each round. Then, the weapons dynamics will be

$$\dot{\zeta}^B(t) = -\rho \min\left(\frac{\eta^R(t)}{\eta^B(t)}, 1\right) \nu^B(t). \quad (1.21)$$

For the case with wrap-around (**MWCO,MRTI,MWWA**), the **SWES** is more reasonable. Since all Blue platforms will have target assignments, the expected number of shots per platform in the unit is ν^B , per round. Therefore the weapons dynamics are

$$\dot{\zeta}^B(t) = -\rho \nu^B(t). \quad (1.22)$$

1.10 Experiment Results and Analysis

Table 1.2: Scenarios Used For Experiments

Scenario Types; $N^B = 8, P_k^B = 0.8$	
A	$N^B = N^R, P_k^B = P_k^R$
B	$N^B = 2N^R, P_k^B = P_k^R$
C	$N^B = N^R, P_k^B = 2P_k^R$

To compare each of the models we simulate each of the above scenarios. We note that each model shows the principle of force concentration. That is,

- When both forces have equal loss rates and equal number of platforms, the dynamics for each force are exactly equal,
- When one force dominates in numbers over the other force, the dominating force has fewer casualties.

There are three types of scenarios that are summarized in Table 1.2. For both uncoordinated and coordinated target selection, we use $\psi = 1$ (the units do not move) and $\nu = 1$ (the units always fire at an acquired target). Also, for **MNCO** we have $\alpha = 0.1$ and for **MWCO** we have $\rho = 0.5$. Under the

MARD assumption for uncoordinated target selection, the acquisition rate has a linear dependence on the number of enemy platforms. That is,

$$\sigma_0^B = N^R \text{ and } \sigma_0^R = N^B.$$

Also, we assume that there are enough weapons on each platform to last through the simulation. We do not worry about weapon expenditure for these experiments.

To compare the models, we must first understand how each model behaves in a realistic battlefield (with our given scenario).

Uncoordinated (MARI) Independent acquisition rate implies $\sigma = 1$. Whenever a platform acquires a target, the platform will fire one salvo at the target with a probability of kill, P_k . Whenever a target is destroyed, the platform will reacquire a new target. This will go on until the simulation ends, or until the platform is destroyed.

Uncoordinated (MARD) The dynamics are the same as **MARI** except for how a platform acquires a target. That is, $\sigma^B(m) = \frac{m}{N^R}$ and $\sigma^R(n) = \frac{n}{N^B}$, where m is the number of Red platforms and n is the number of Blue platforms at some time t . When targets are sparse, the acquisition rate is smaller.

Coordinated (MNWA) At the beginning of each round, the maximum number of platforms targeted by the enemy is equal to the number of platforms in the smallest unit. For our scenarios, there will be $\min(N^B, N^R)$ Red targets for Blue and $\min(N^B, N^R)$ Blue targets for Red when the simulation begins. When a round begins, both sides fire simultaneously.

Coordinated (MWWA) For the simulation we have targeting as defined in Section 1.7.2. The commander will assign unique targets for each his platforms until all targets have been targeted, then the remaining platforms will start targeting from the beginning of the list of targets until all platforms have an enemy target. When a round begins, both sides fire simultaneously.

Figures 1.1, 1.3 and 1.5 give the plots of the probability distribution of certain simulation times for Uncoordinated Target Selection. In each scenario we can see that the probability distribution for **MARD** cases show lower covariances over the simulation then the **MARI** cases. This is expected since the independent acquisition rate case allows for more acquisitions per unit time then the dependent case when targets become sparse.

Next we shift our attention to the coordinated target selection models. From Figures 1.2, 1.4 and 1.6 we see less variation in the distribution for **MWWA** cases than in the **MNWA** cases. By understanding the differences between the models we can see why this happens. Since the ‘wrap-around’ case allows for targets to be selected by more than one enemy platform, we can understand that the dominating force will obtain better chances of winning than the weaker force. With ‘no wrap-around’, the force that wins is the force with better probabilities of killing the enemy target.

There are definite differences between the distributions of uncoordinated and coordinated target selection models. In the uncoordinated target selection case, at any given time, only one platform can be destroyed. On the other hand, in the coordinated target selection model, after a round, a whole unit has a non-zero probability of being destroyed.

After seeing how the probability distribution evolves for each model, we need to examine the expected values of the number of platforms for both sides and compare them with the approximated expected values. We want to examine how much of an outlook of the battle dynamics we can see with the approximated expected values. The derivation of the approximate expected values does not guarantee any reliable prediction over a long horizon.

Tables 1.3 and 1.4 summarize the covariances and the approximation error in the expected values using the L_2 norm for the duration of the engagement, for all models and scenarios.

In Figures 1.7-1.12 we have the expected values (with covariances) for the uncoordinated target selection case. There are, of course, small differences in the exact values from **MARI** to **MARD**. Yet we see that the approximation for **MARD** is much better since it has an overall better prediction of

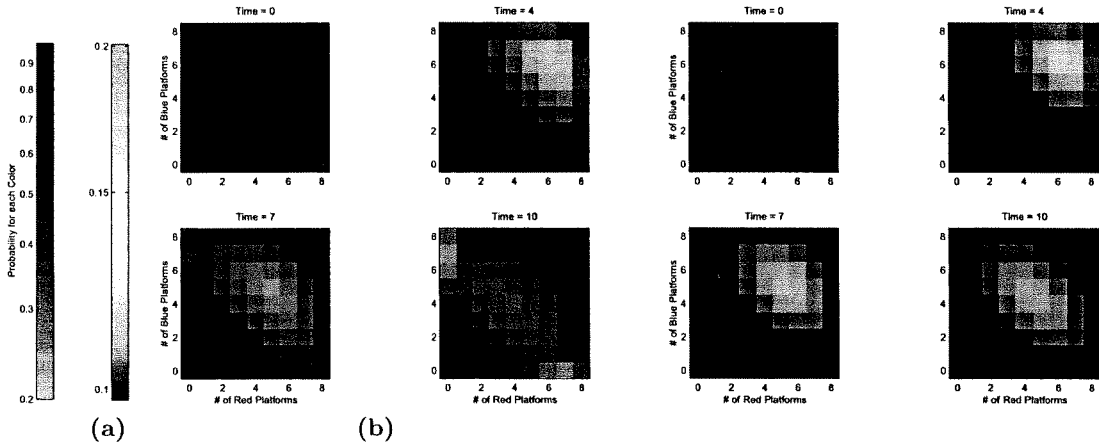


Figure 1.1: Evolution of the probability distribution for Uncoordinated Target Selection for Scenario A((a):MARI and (b):MARD).

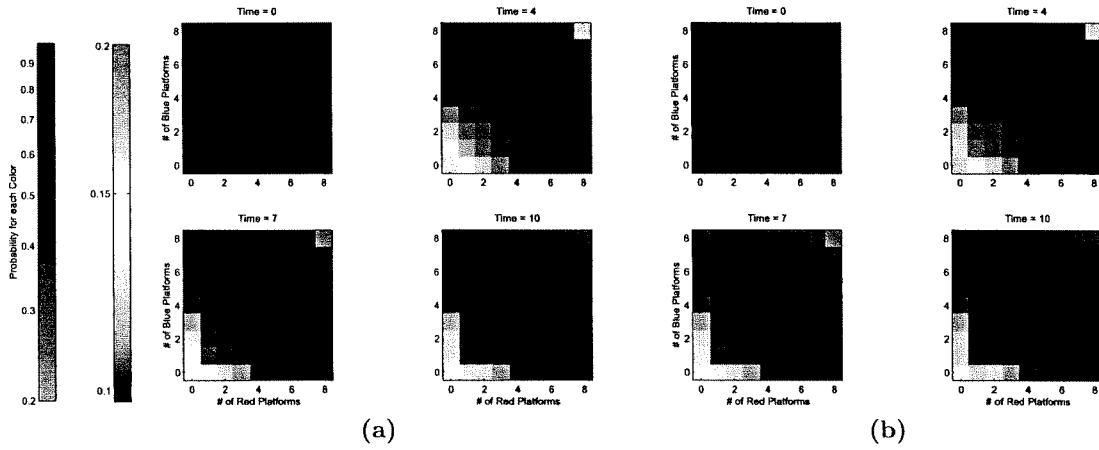


Figure 1.2: Evolution of the probability distribution for Coordinated Target Selection for Scenario A((a):No Wrap-Around and (b):Wrap-Around)

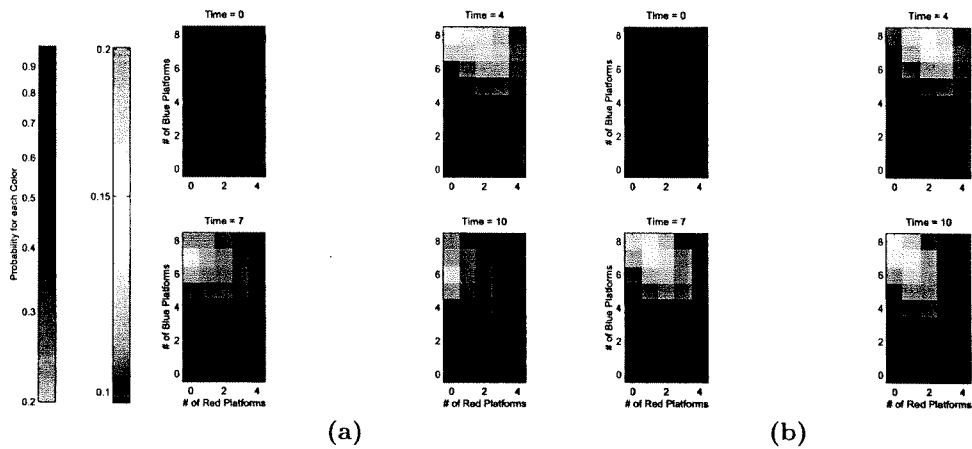


Figure 1.3: Evolution of the probability distribution for Uncoordinated Target Selection for Scenario B((a):MARI and (b):MARD).

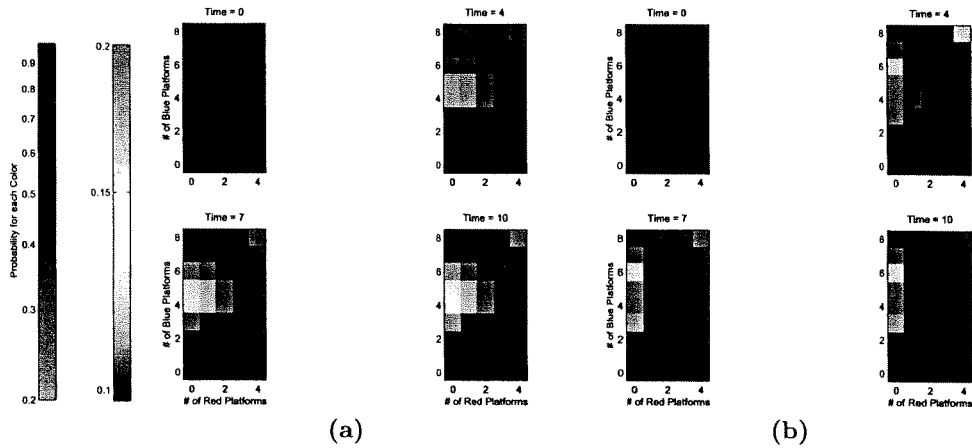


Figure 1.4: Evolution of the probability distribution for Coordinated Target Selection for Scenario B((a):No Wrap-Around and (b):Wrap-Around)

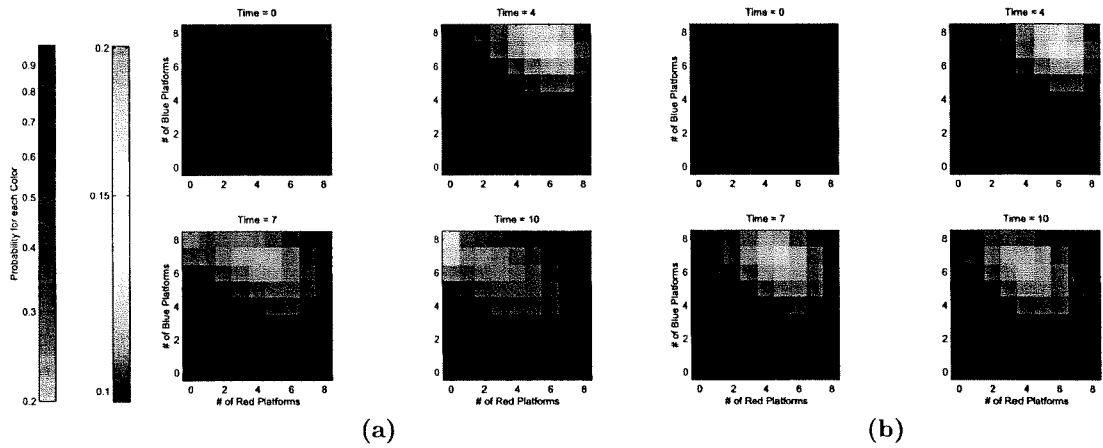


Figure 1.5: Evolution of the probability distribution for Uncoordinated Target Selection for Scenario C((a):MARI and (b):MARD).

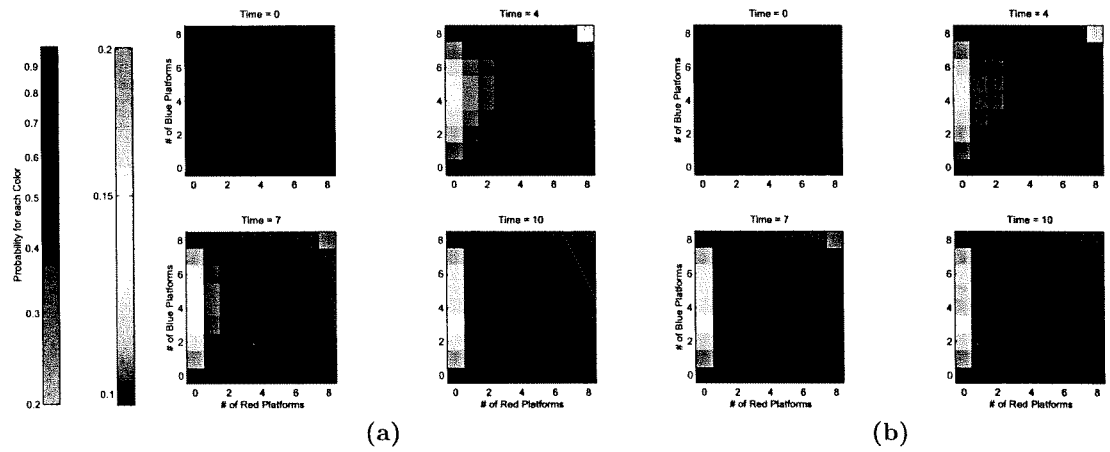


Figure 1.6: Evolution of the probability distribution for Coordinated Target Selection for Scenario C((a):No Wrap-Around and (b):Wrap-Around)

Table 1.3: L_2 -Norm of Covariances

Exp. #	1	2	3	4	5	6	7	8	9	10	11	12
	MARI			MARD			MNWA			MWWA		
Scenario	A	B	C	A	B	C	A	B	C	A	B	C
σ^{B^2}	10.16	4.09	4.90	5.33	3.19	3.46	20.83	8.79	10.54	19.18	5.31	10.16
σ^{R^2}	10.16	3.85	9.81	5.33	2.80	5.26	20.83	8.36	21.67	19.18	5.79	20.12
$\sigma^B \sigma^R$	5.82	1.65	2.93	2.09	1.05	1.30	18.42	7.70	10.15	16.80	4.74	9.28

Table 1.4: L_2 -Norm of Error Between Actual and Approximate Expected Values

Exp. #	1	2	3	4	5	6	7	8	9	10	11	12
	MARI			MARD			MNWA			MWWA		
$\ \eta^B - \hat{\eta}^B\ $	0.14	0.37	0.01	0.06	0.03	0.02	1.35	5.80	0.02	1.53	4.74	0.16
$\ \eta^R - \hat{\eta}^R\ $	0.14	1.35	0.25	0.06	0.06	0.04	1.35	5.80	0.04	1.53	0.06	0.58
$\ \eta^B - \hat{\eta}^B + \eta^R - \hat{\eta}^R\ $	0.28	1.72	0.26	0.12	0.09	0.06	2.70	11.60	0.06	3.06	4.80	0.74

the outcome over the simulation. It should be noted that the approximation is not necessarily always this good, yet over other simulations, the approximation for **MARD** is better than **MARI**. A reason for this can be associated with the acquisition rate. In the **MARI** case, platforms acquire targets at a constant rate independent of the number of enemy platforms. In the **MARD** case, platforms acquire targets at rate dependent on the number of enemy platforms. This means as targets become more sparse, the dynamics of the dependent case will “slow down”. That is, the amount of targets being acquired will decrease as the targets become more sparse. This allows for an approximation to remain good as targets become more sparse.

Lastly, Figures 1.13-1.18 illustrate the expected values (with covariances) for the coordinated target selection case. Here we do not see much difference between the actual expected values, yet one notable difference is in the covariances. The covariances for the ‘wrap-around’ case are smaller at the end of the simulation time than the ‘no wrap-around’ case. We notice through other simulations, that over a significant amount of simulation time the ‘wrap-around’ case has better covariances.

1.11 Conclusions and Recommendations

These results indicate that, the ODE models were good approximations under the uncoordinated target selection assumption, and were found to be sufficient to represent the attrition dynamics in a differential game setup in this case. The discrepancy between the MC and ODE models increase as the engagement proceeds, which should be expected. Under the coordinated target selection assumption, the ODE approximations were worse. This can be partially explained by the fact that coordination implies firing in rounds, and therefore platform loss is more discrete in nature.

An extension of the above modeling procedure to forces with multiple units is not immediate. There are at least two issues that need to be addressed:

- A unit may engage in a firefight with more than one enemy unit at a given time. In this case, the fire intensity command for this unit must be replaced by a vector of commands, with length equal to the number of enemy units. From a modeling perspective this may be acceptable. However, this “proliferation of inputs” is quite undesirable for control design.
- The state-space of the resulting Markov Chain will grow geometrically with the number of units. As a result, verification of approximation of expected values via simulation may become infeasible.

A multi-unit model, based on heuristic arguments, is included as an appendix to this chapter.

As mentioned in Section 1.7, round arrival may depend on platform loss (**MWCO**, **MRTK**). In this case, one needs to consider a random variable for the time it takes one platform to kill its assigned target. In most cases, the distribution of this variable will not be exponential and the resulting stochastic process may not be a Markov Chain. Further investigation of this topic, with several likely distributions for the kill time, would be interesting.

It is also of interest to model attrition when fighting units have different rules about target selection and coordination. For instance, the platforms of one unit may be coordinated (and hence will fire in rounds), while the platforms of the enemy unit are uncoordinated (and hence will fire continuously as targets are acquired).

Another research direction would be to develop better approximations for the expected values by employing model reduction tools, such as singular perturbation analysis or principal component analysis. Whether more sophisticated mathematics will result in better overall predictive capability will, no doubt, depend on the nature of assumptions involved.

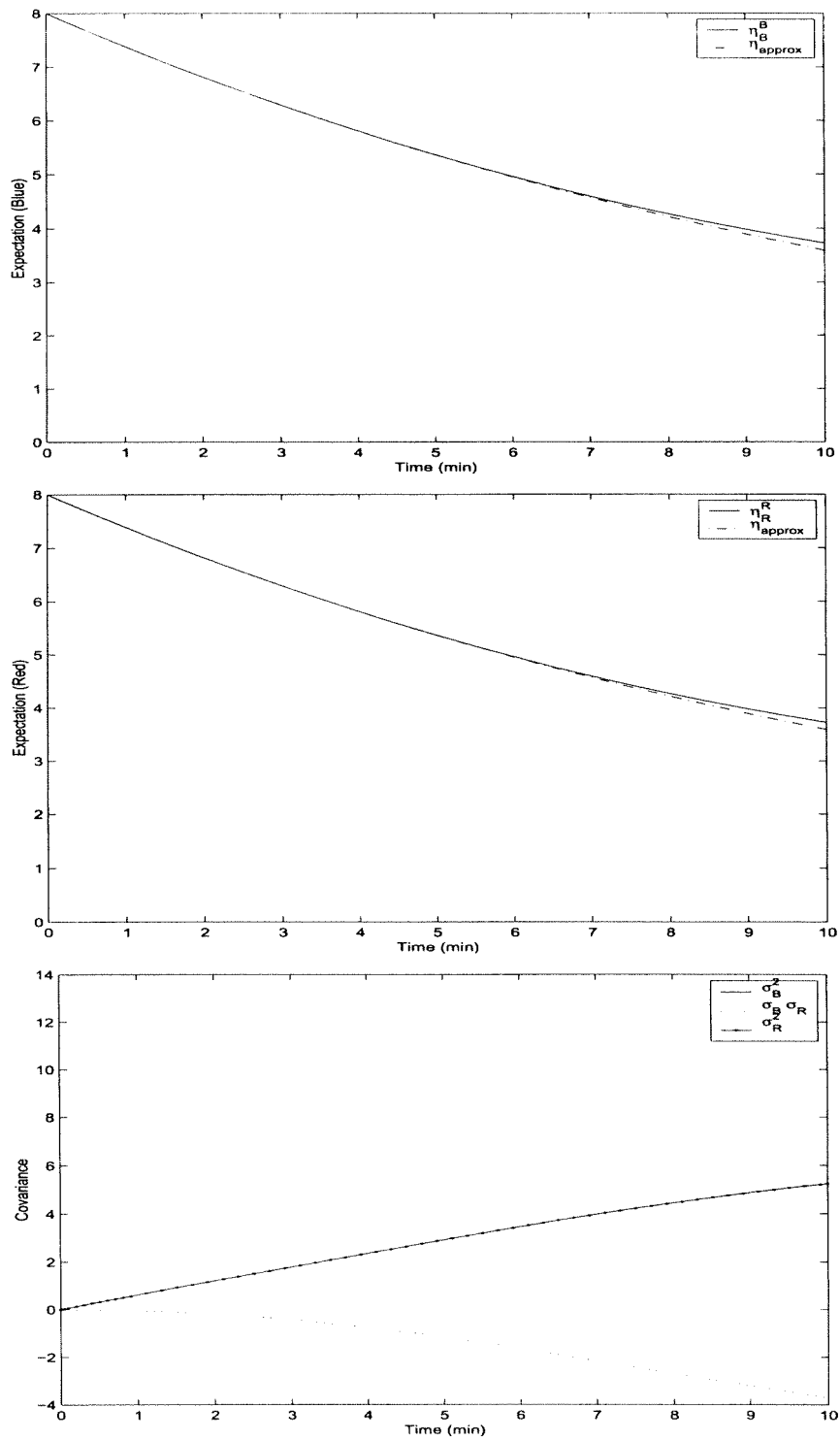


Figure 1.7: Evolution of the expected values (actual and approximated) and covariances for Uncoordinated Target Selection (MARI, Experiment 1.1)

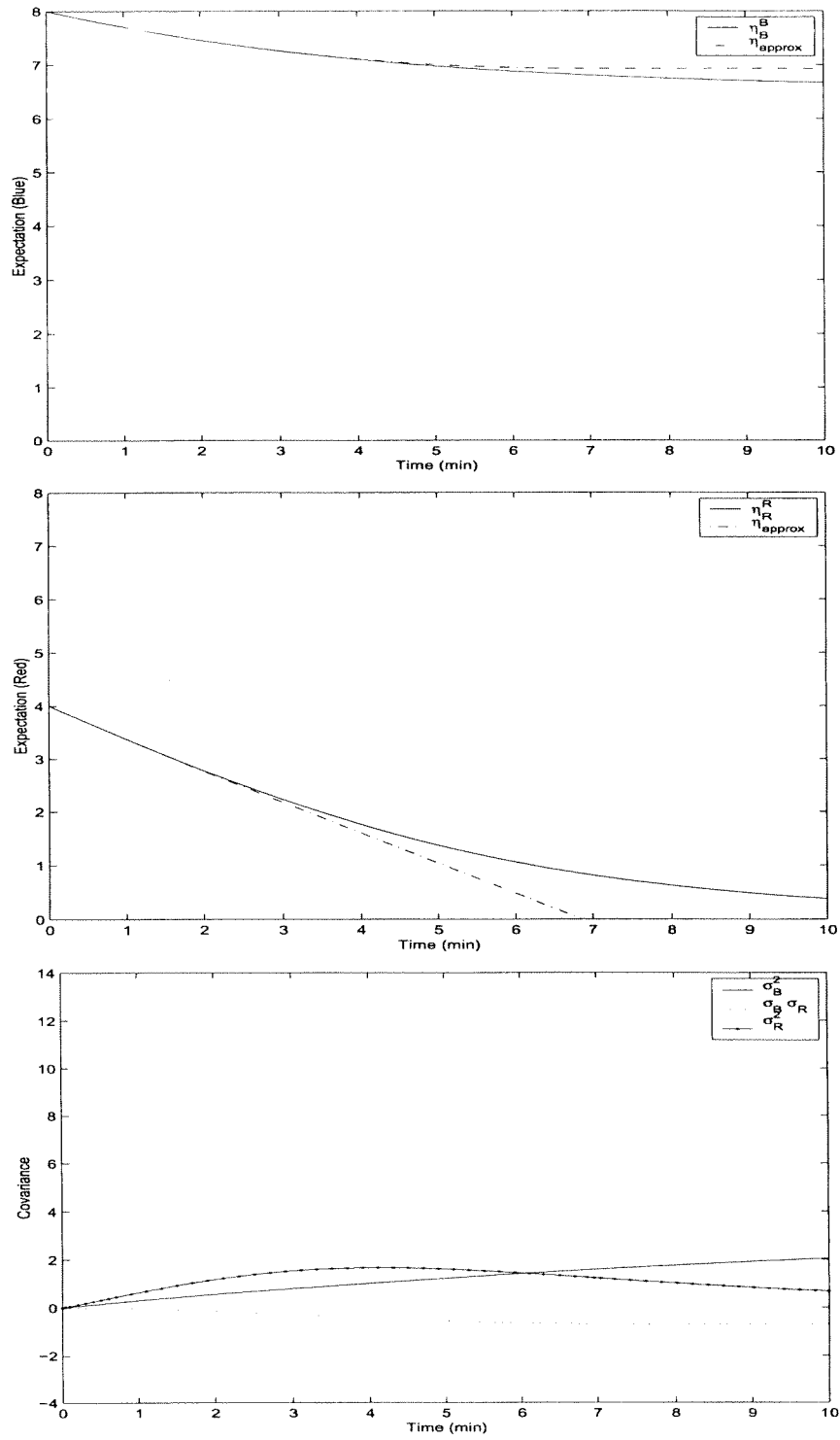


Figure 1.8: Evolution of the expected values (actual and approximated) and covariances for Uncoordinated Target Selection (MARI, Experiment 1.2)

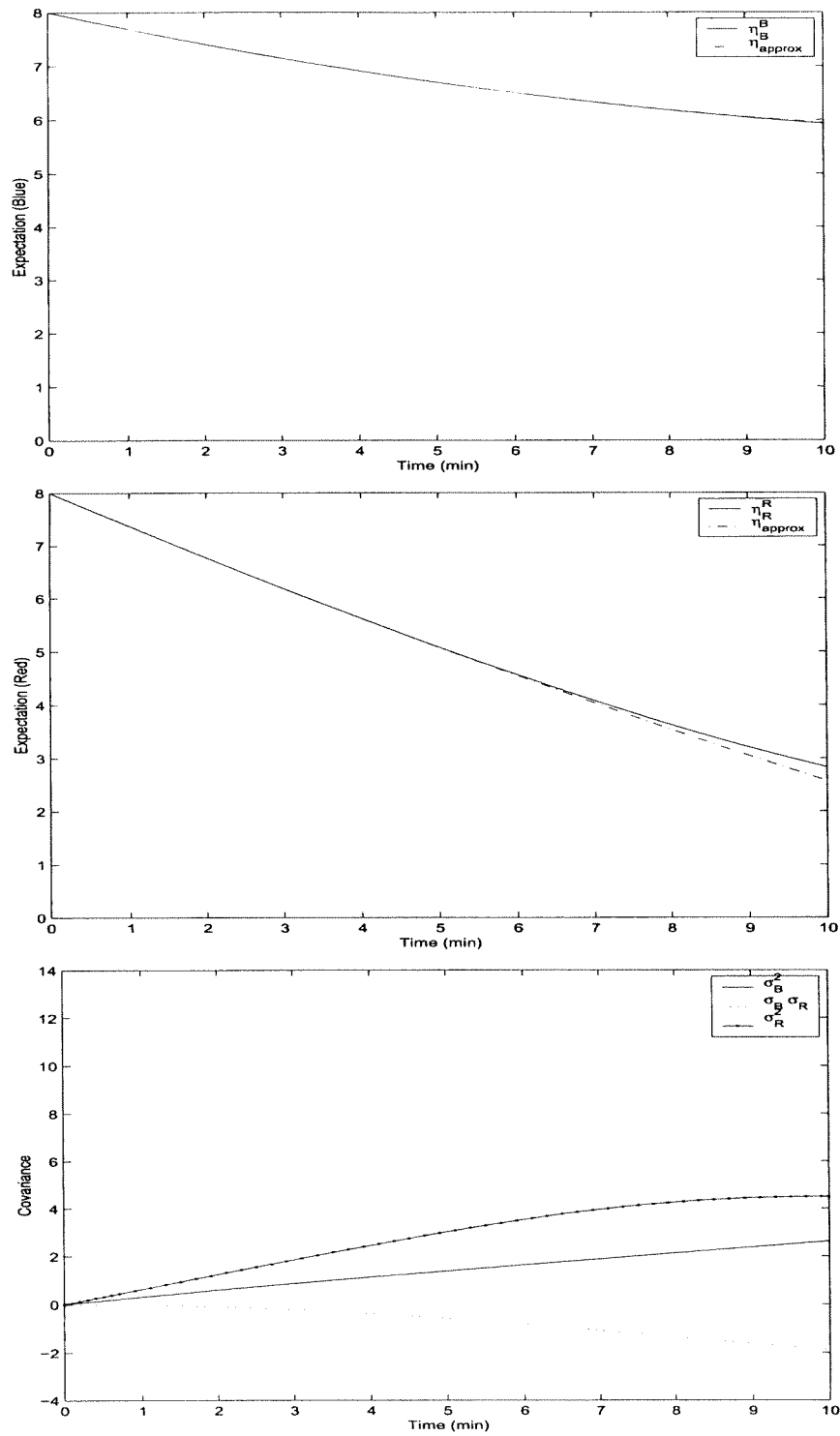


Figure 1.9: Evolution of the expected values (actual and approximated) and covariances for Uncoordinated Target Selection (MARI, Experiment 1.3)

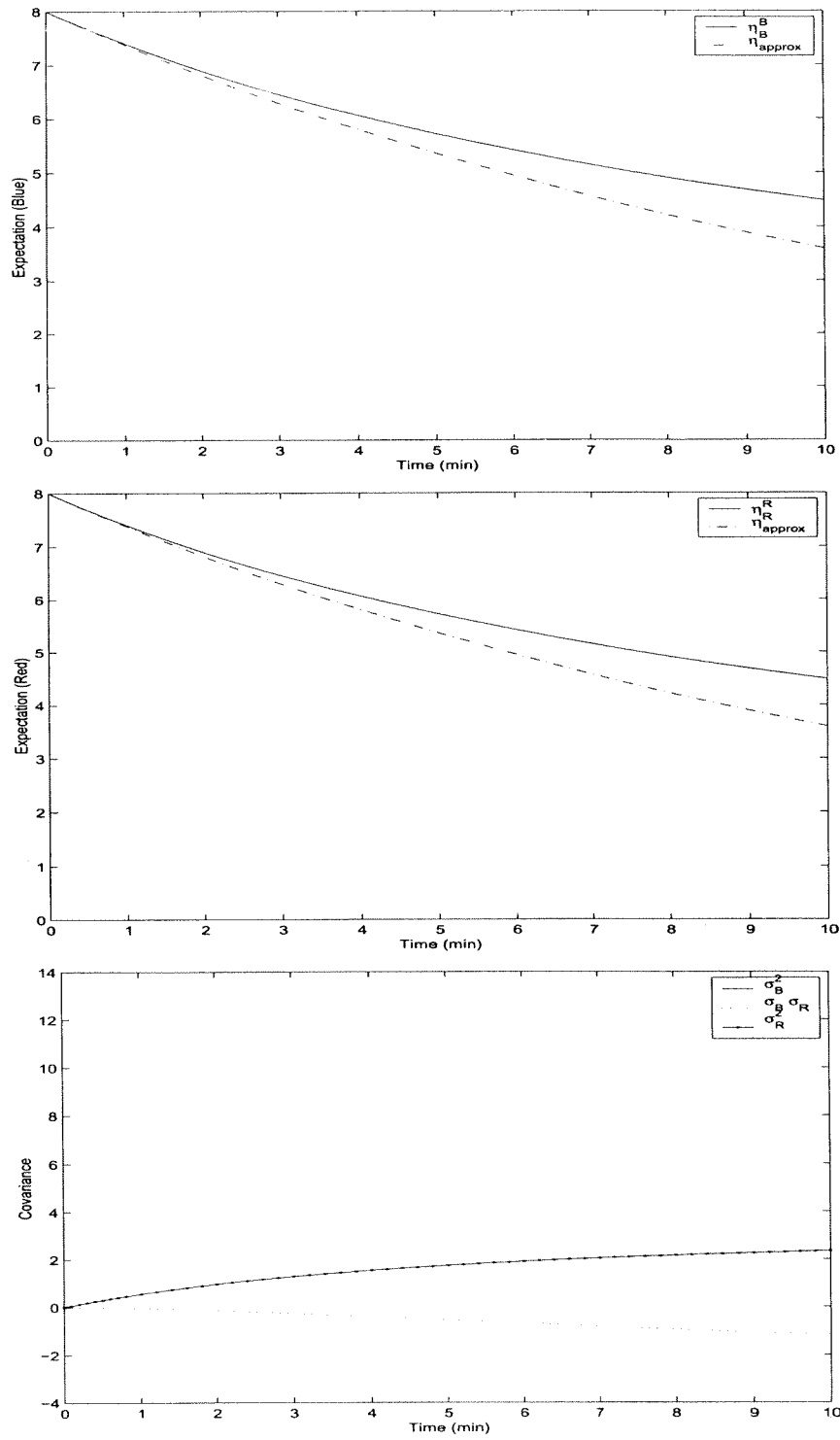


Figure 1.10: Evolution of the expected values (actual and approximated) and covariances for Uncoordinated Target Selection (MARD, Experiment 1.4)

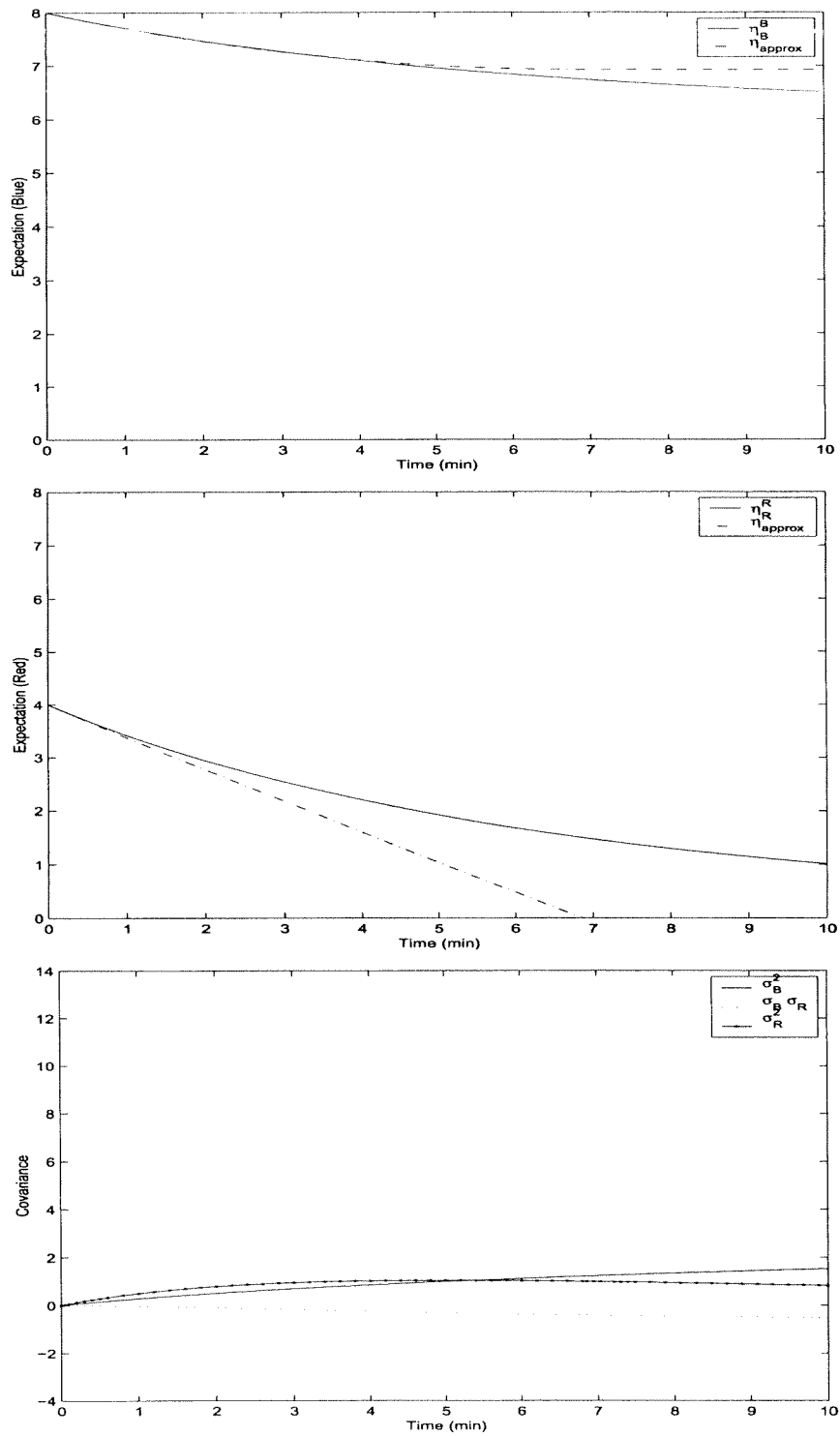


Figure 1.11: Evolution of the expected values (actual and approximated) and covariances for Uncoordinated Target Selection (MARD, Experiment 1.5)

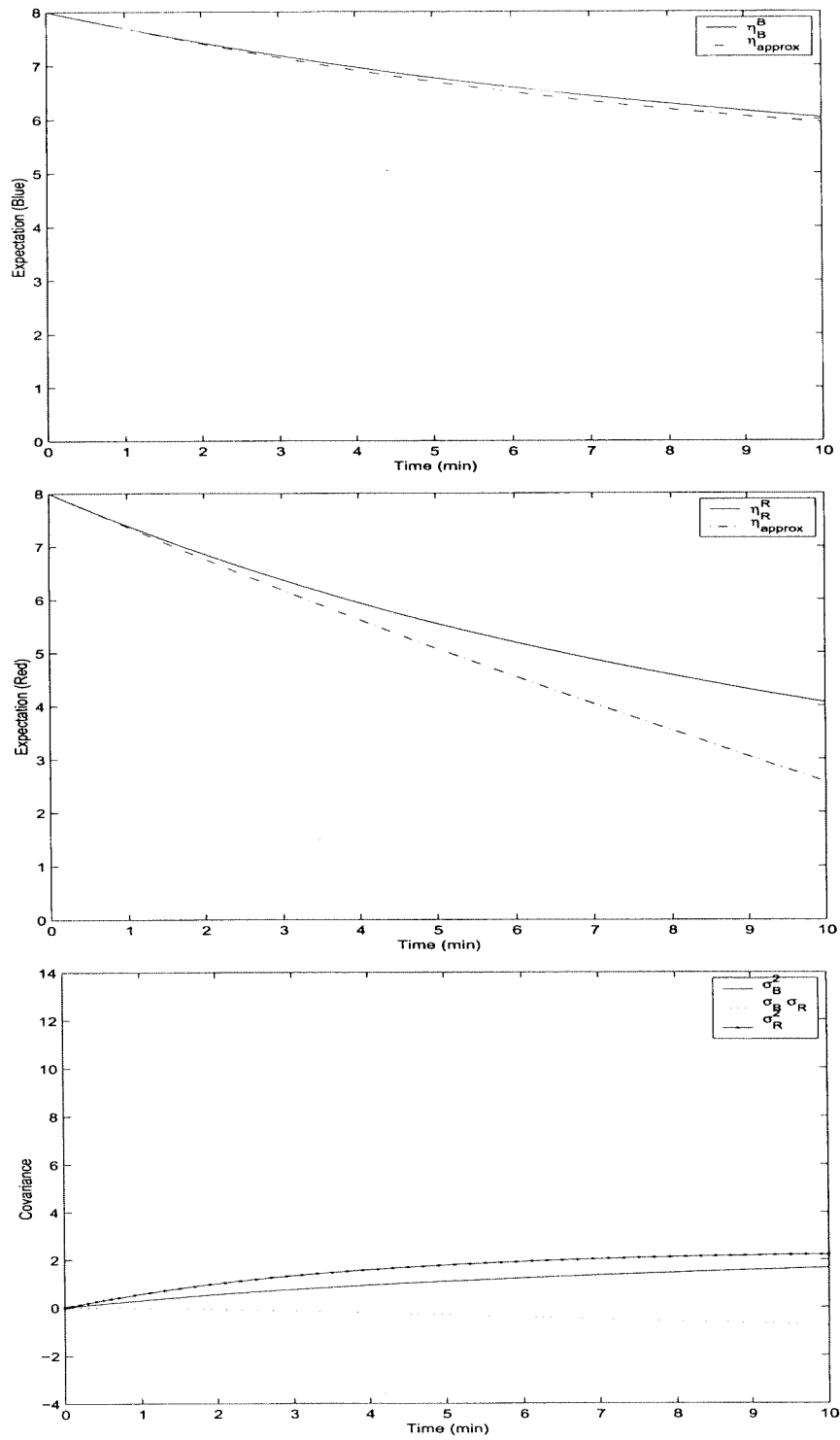


Figure 1.12: Evolution of the expected values (actual and approximated) and covariances for Uncoordinated Target Selection (MARD, Experiment 1.6)

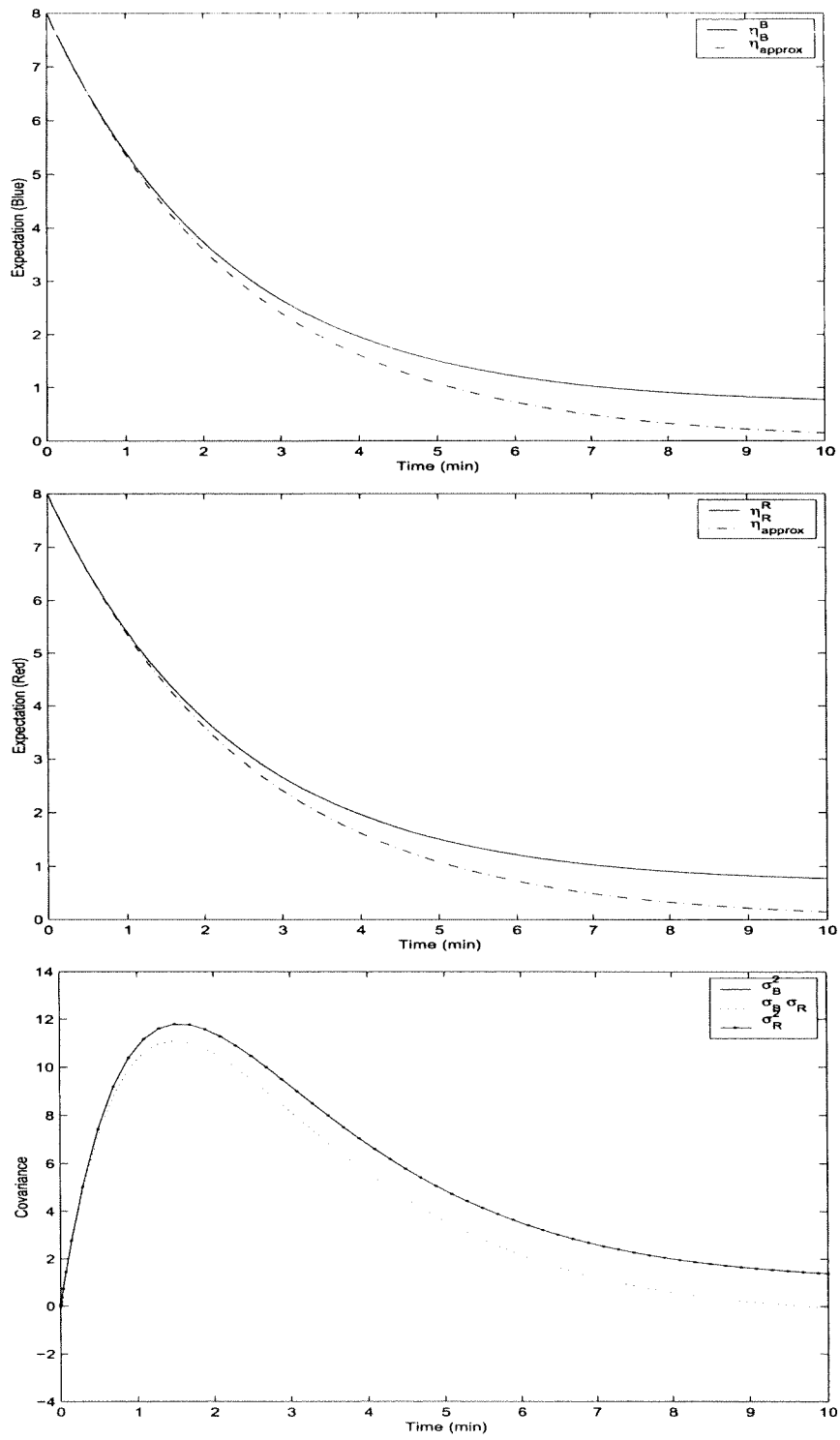


Figure 1.13: Evolution of the expected values (actual and approximated) and covariances for Coordinated Target Selection (MNWA, Experiment 1.7)

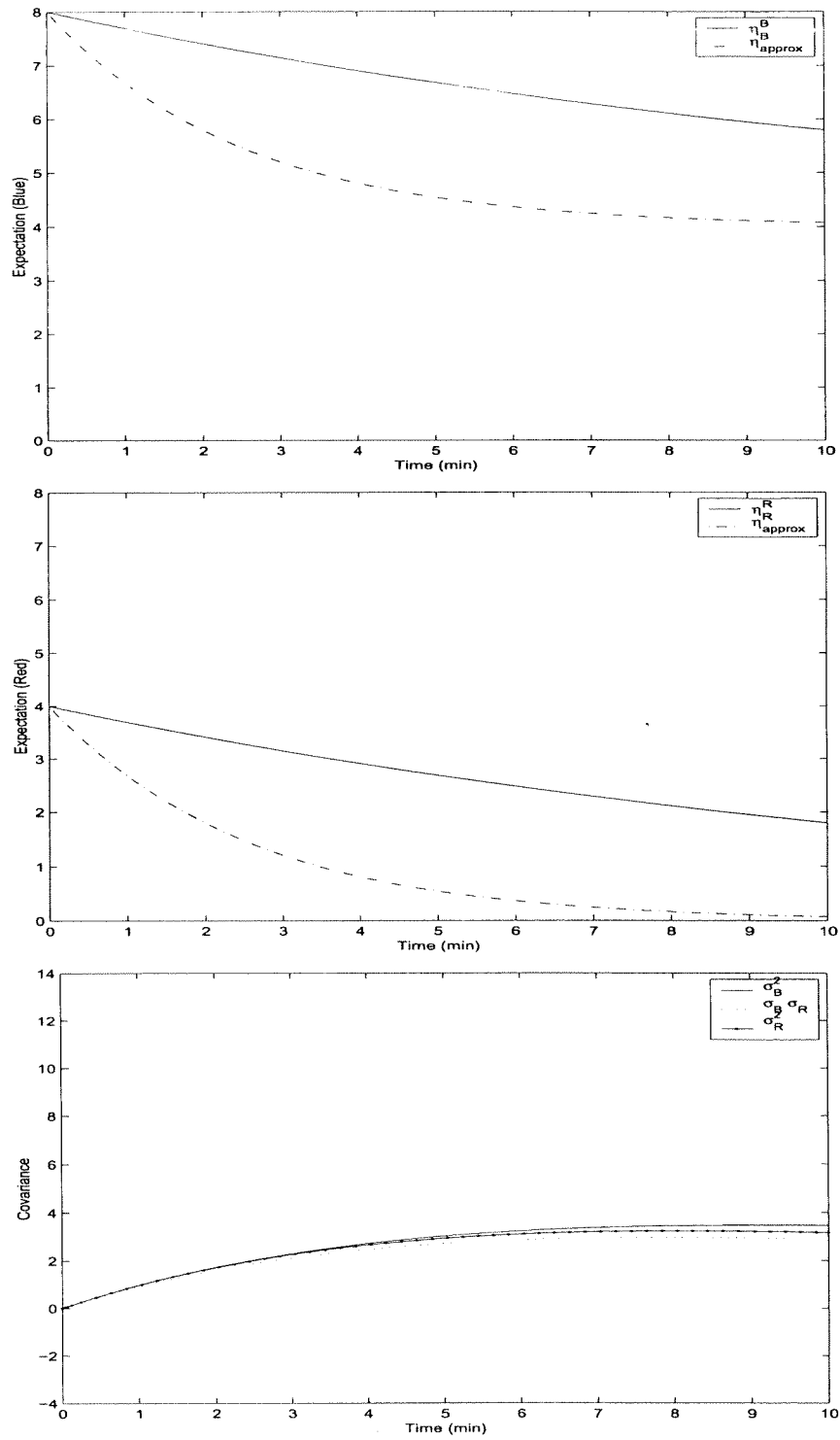


Figure 1.14: Evolution of the expected values (actual and approximated) and covariances for Coordinated Target Selection (MNWA, Experiment 1.8)

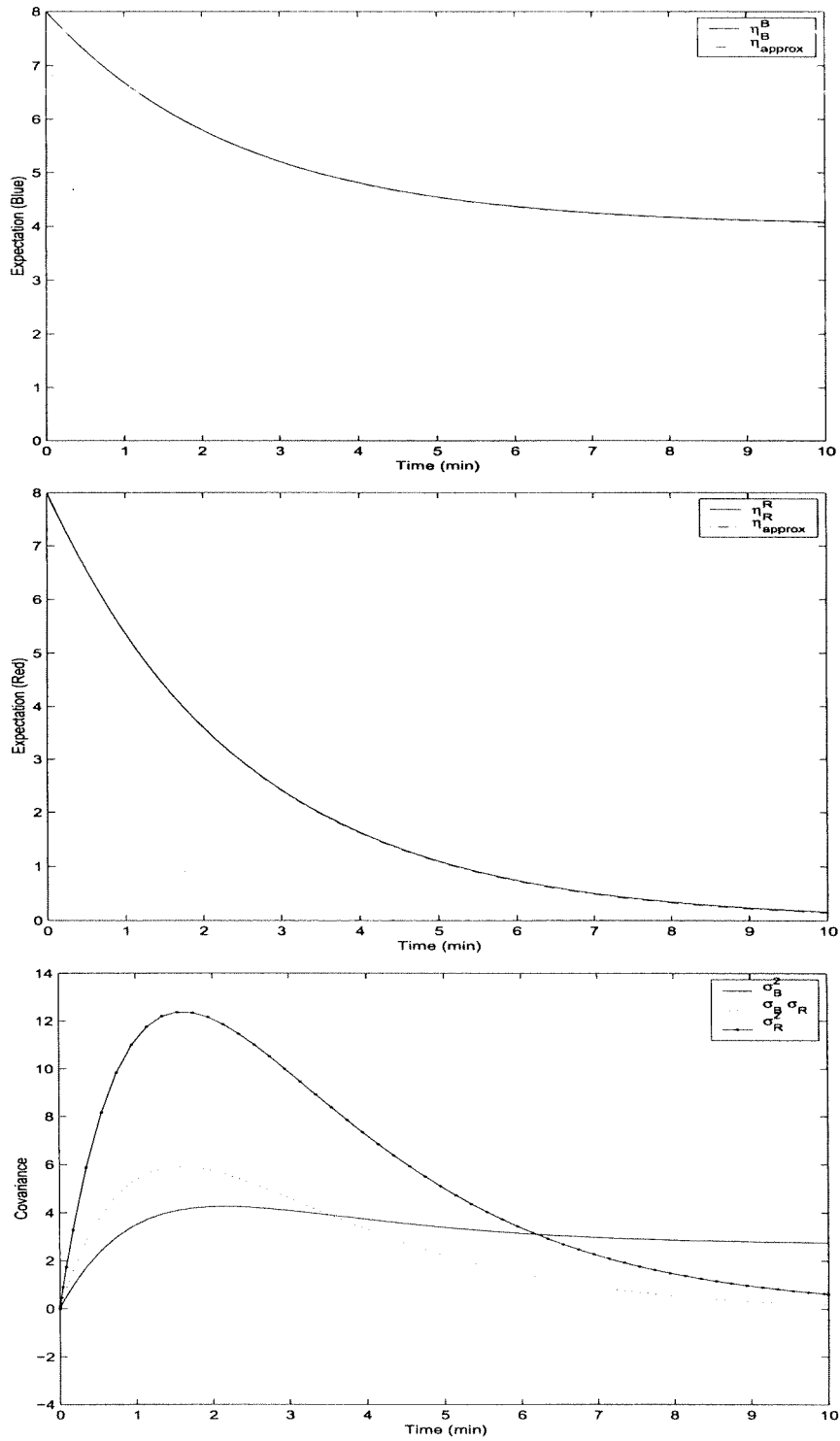


Figure 1.15: Evolution of the expected values (actual and approximated) and covariances for Coordinated Target Selection (MNWA, Experiment 1.9)

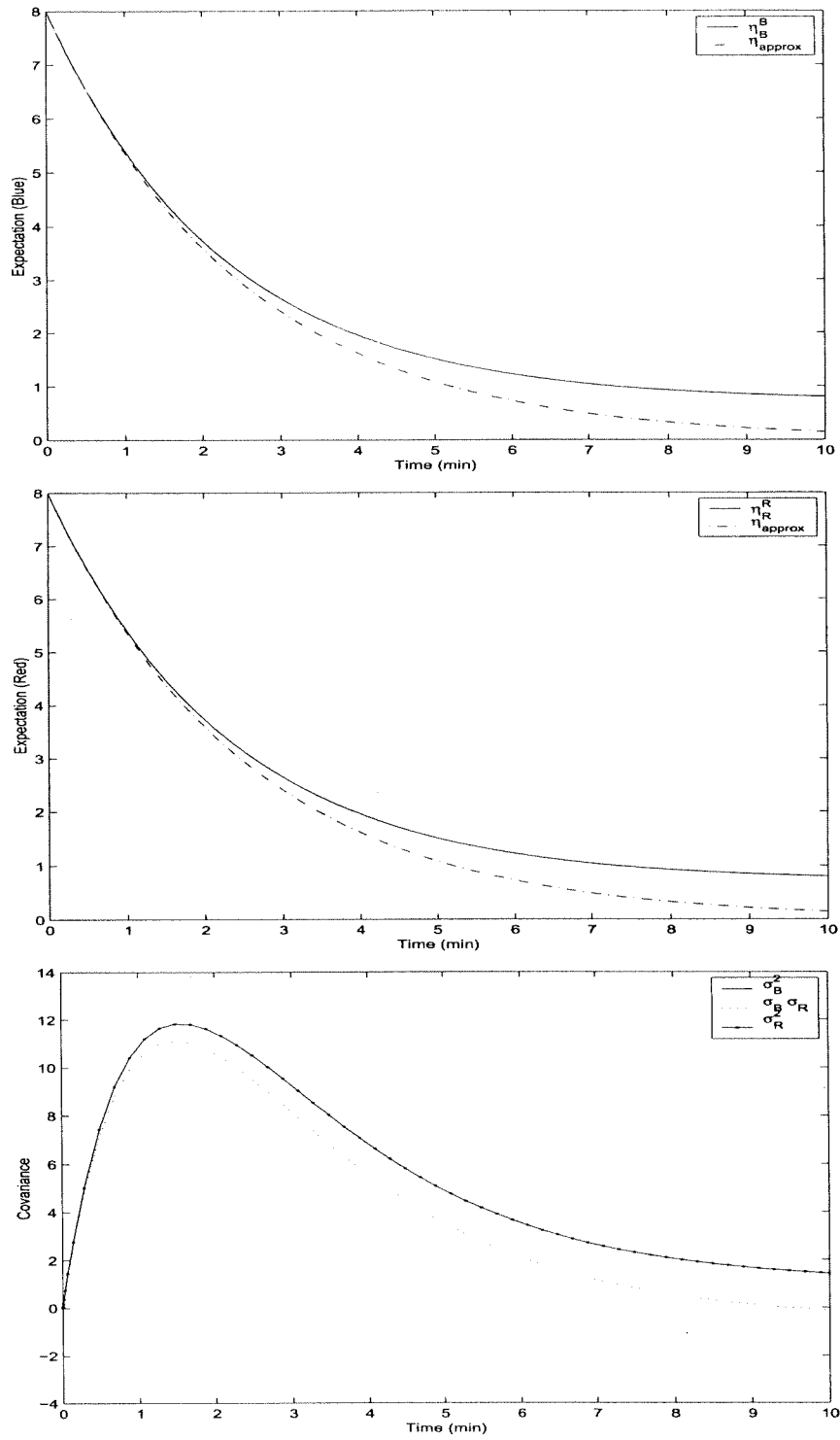


Figure 1.16: Evolution of the expected values (actual and approximated) and covariances for Coordinated Target Selection (MWWA, Experiment 1.10)

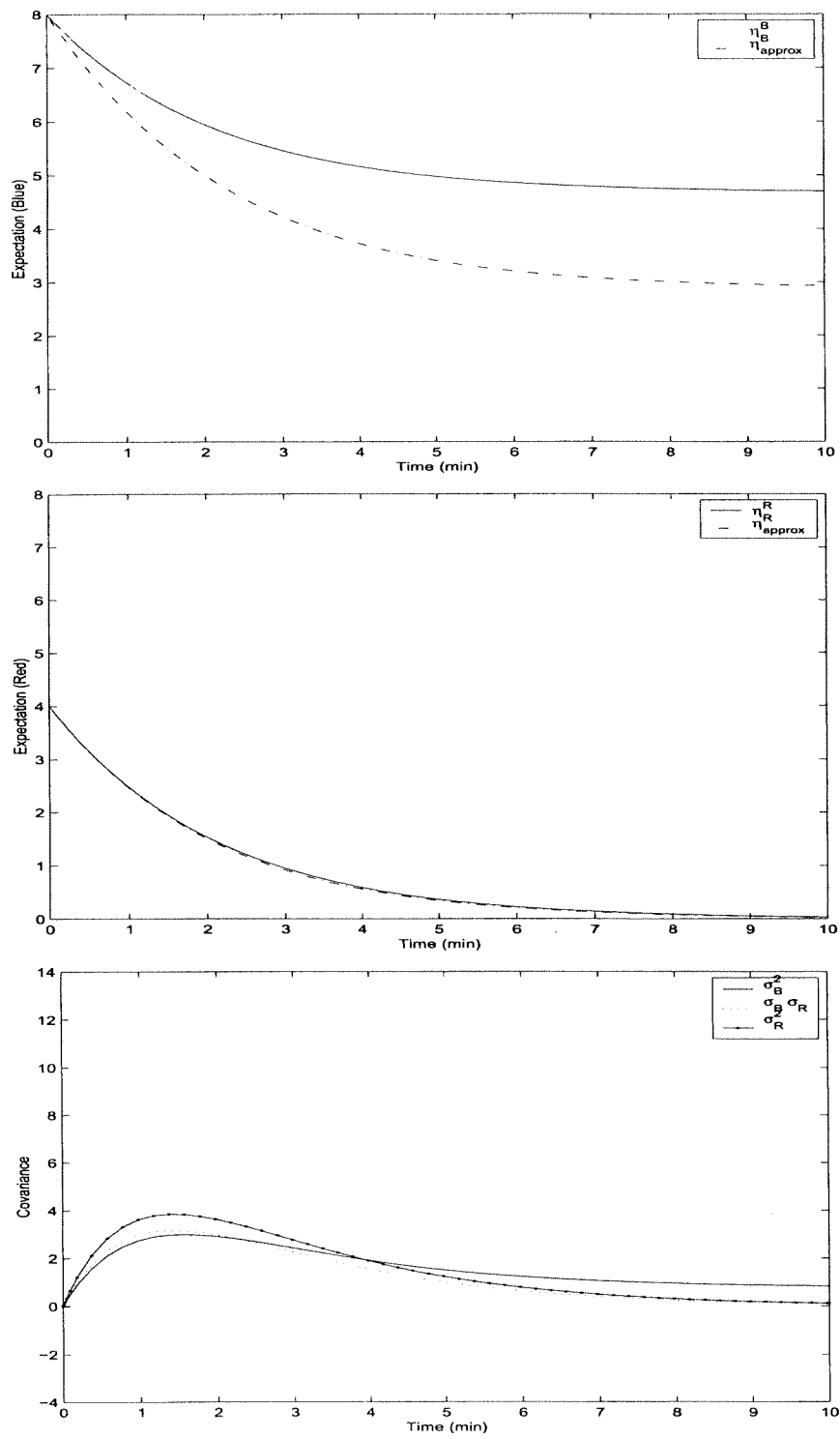


Figure 1.17: Evolution of the expected values (actual and approximated) and covariances for Coordinated Target Selection (MWWA, Experiment 1.11)

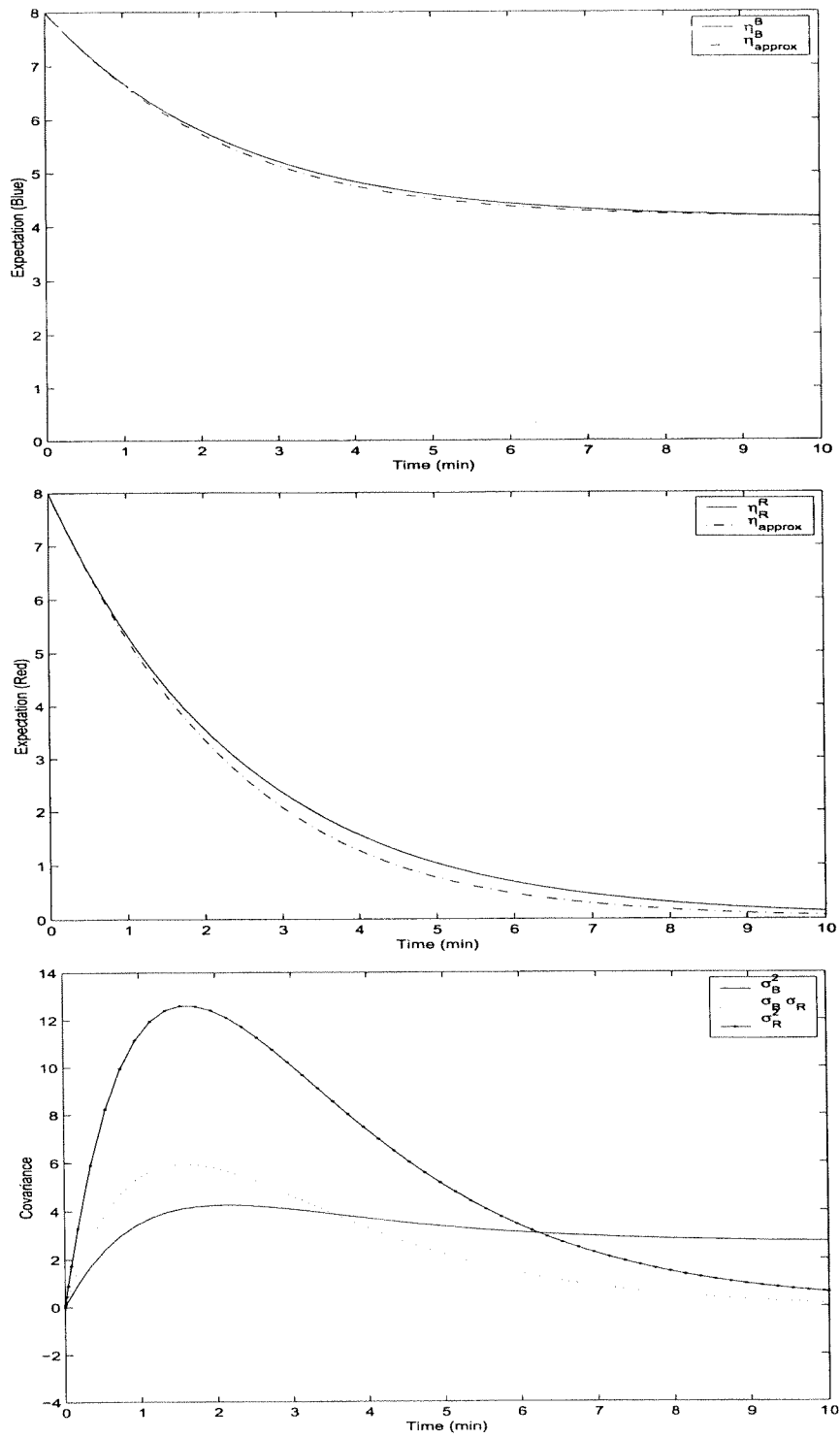


Figure 1.18: Evolution of the expected values (actual and approximated) and covariances for Coordinated Target Selection (MWWA, Experiment 1.12)

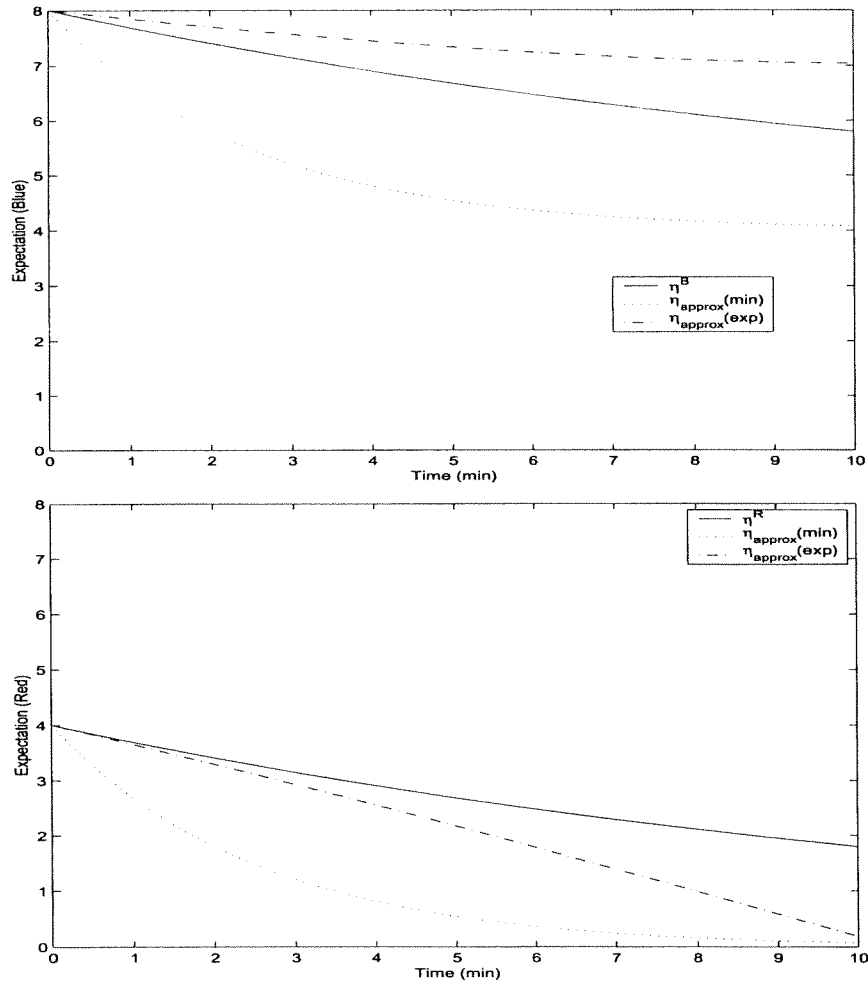


Figure 1.19: Comparison of Expected Values for Experiment 1.8 with approximated ODE's using $\min(\hat{\eta}^B, \hat{\eta}^R)$ and $e^{\frac{\eta^B}{\eta^R}}$.

1.12 Appendix: Mission Dynamics Continuous-Time Model 3.0

1.12.1 Modeling Assumptions

This section describes the Mission Dynamics Continuous-time Model version 3.0, (MDCM 3.0), which is a multi-unit extension of the ODE model which approximates the expected values of the number of platforms, for the uncoordinated target selection, independent acquisition rate case.

As usual, we will denote the quantities which belong to the forces Blue and Red with superscripts B and R . If the symbols in an expression do not have superscripts, this is intended to mean that the expression is valid for either force.

Consider two (homogeneous) units of the opposing forces engaged in a fire fight, in which platforms of both sides are shooting at each other simultaneously. During this engagement, each platform searches for an enemy platform (in the sky, on the ground, etc.). When a platform is located in space, identified to be an enemy and the weapon system is locked on to this platform, a target is said to be acquired.

The state variables for each unit are its position $\xi \in \mathbb{R}^2$, the expected value of the number of platforms in the unit $\eta \geq 0$ and the expected value of the number of salvo loads of weapons per platform $\zeta \geq 0$.

In addition to the speed controls $\mu \in [-1, 1]^2$, each unit has a fire intensity control $\pi \in [0, 1]$. One interpretation is as follows: Suppose an object is located and identified as a target by our detection device. This may be a false detection due to the enemy's electronic warfare or the object may in fact be a decoy. Then π reflects the mission commander's confidence in the detection device under the given circumstances and allows provident use of weapons. In this way, π becomes the frequency of firing a salvo per target acquisition (i.e., the probability of firing, given a target is acquired).

Consider the Blue unit firing on the Red unit. Let the probability of kill for each weapon, given it is fired, be $P\{\text{wkill}^B \mid \text{fired}^B\}$. The probability of killing the target, with a salvo of s^B weapons, given that they are fired simultaneously, is

$$P^B \stackrel{\text{def}}{=} P\{\text{kill}^B \mid \text{fired}^B\} = \begin{cases} \left[1 - (1 - P\{\text{wkill}^B \mid \text{fired}^B\})^{s^B}\right] & \text{if } \zeta^B(t) > 0, \\ 0 & \text{if } \zeta^B(t) = 0. \end{cases} \quad (1.23)$$

The following were assumed to hold when deriving the Markov chain model in [1]:

MNCO (No COordination) Friendly platforms do not communicate for target selection (uncoordinated selection). The exception is when a platform, which has depleted its supply of weapons, locates and identifies a target. In that case, this platform will relay this information to a friendly platform in the same unit which still has weapons.

MARI (Acquisition Rate Independent): Target acquisition rate does not depend on the number of enemy platforms or their distribution in space. (Search devices are advanced enough that they will locate enemy platforms efficiently even when they are distributed sparsely.)

MPKD (Probability of Kill depends on Distance) Probability of killing the target depends on the distance between the units.

MNWT (Negligible Weapon reach Time) The time it takes for a missile, bomb, or other weapon to reach its target is negligible.

MNSA (No Self-Attrition): Self-attrition or equipment breakdowns are negligible.

From **MPKD**, (1.23) will depend on the distance between units, with a function $\psi : \mathbb{R} \rightarrow [0, 1]$ which depends on the positions of each unit, $\psi^B(\|\xi^B - \xi^R\|)$. Let us take this function as $\psi(r) = \exp(-(r/r_0)^2)$, where r_0 is a parameter which depends on the type of units. Therefore the probability of a platform killing its target, given it is assigned to a target, is $P\psi\pi$.

1.12.2 State Equations

Although the models in [1] were derived for a one-on-one engagement, we will assume that extension to multiple unit case can be obtained simply by considering multiple, simultaneous, independent one-on-one engagements. As usual, we will use subscripts to index units. The first subscript indicates the “shooter” and the second one indicates the “shootee”. The maximum speeds for unit i , in both x and y directions is α_i . Denote by ρ_{ij} the rate at which a platform in unit i acquires a platform in unit j as a target. We include the possibility that a unit may fire at more than one enemy unit, and it may be fired upon by more than one enemy unit at the same time. However, these target assignments are fixed during the mission. The indices of the Red units that Blue unit i is shooting at (“the shootees of Blue i ”) are denoted by $f_s^B(i)$. Thus, the firing intensity control for this Blue unit has $|f_s^B(i)|$ components, where $|\cdot|$ denotes the cardinality of a set. The k^{th} component of the firing intensity of Blue unit i , $\pi_{ik}^B \in [0, 1]$, corresponds to its fire against the j^{th} Red unit, in which j is the k^{th} element of $f_s^B(i)$. The indices of the Red units which are shooting at Blue unit i (“the shooters against Blue i ”) are denoted by $f_e^B(i)$.

Here, only the dynamics for Blue unit i will be displayed. The corresponding equations for a Red unit can be obtained using the symmetry between the forces, by interchanging B with R . The motion on the plane is given by

$$\frac{d}{dt} \xi_{xi}^B(t) = \alpha_i^B \mu_{xi}^B(t), \quad (1.24)$$

$$\frac{d}{dt} \xi_{yi}^B(t) = \alpha_i^B \mu_{yi}^B(t). \quad (1.25)$$

The platform loss evolves as

$$\frac{d}{dt} \eta_i^B(t) = - \sum_{j \in f_e^B(i)} \eta_j^R(t) \rho_{ji}^R P_{ji}^R \psi_{ji}^R (\|\xi_i^B(t) - \xi_j^R(t)\|) \pi_{jk}^R(t), \quad (1.26)$$

where i is the k^{th} element of $f_s^R(j)$. The weapon expenditure is given by

$$\frac{d}{dt} \zeta_i^B(t) = - \sum_{k=1}^{|f_s^B(i)|} \text{sign}(\eta_j^R(t)) \rho_{ij}^B \pi_{ik}^B(t), \quad (1.27)$$

where j is the k^{th} element of $f_s^B(i)$.

Bibliography

- [1] Markov Chain Combat Models for Attrition Prediction and Mission Control, İ. Tunay, J. Goodwin, and H. Mukai, presented and accepted for publication in 3rd Int'l Meeting of INFORMS Military Applications Society (3MAS), (San Antonio, Texas), November 2000.

Chapter 2

Experiment 2: Controller Performance Comparison with Other Controllers

2.1 Executive Summary

This is experiment for hypothesis two. Both the plant and internal models are the same, i.e., the Mission Dynamics Continuous-time Model (MDCM). There is no noise added to the state variables when constructing the observed state variables (the output variables). The control actions of the Blue and Red teams are generated by one of the following strategies: the proposed game theoretic algorithm, a simple heuristic stochastic strategy (e.g. a movement bias is given toward targets), a simple heuristic deterministic strategy, and a human planner.

The strategy adopted by Blue and Red is optimal with sense of a Nash equilibrium with respect to the value function; that is, it maximizes the value function with respect to Red and minimizes it with respect to Blue.

2.2 Experiment Scope

We did a series of experiments to evaluate the effectiveness of the current differential game technology as a means of countering the enemy actions under idealized situations with perfect information about enemy states, initial conditions and objectives.

For the experiments we considered two different one vs. one scenarios and four different cases corresponding to each scenario. In all cases the strategy for the blue player was determined by a game theoretic controller, with running cost on the velocity controls and the distance to the target, and terminal cost on the final number of platforms.

Case 1 (for each of the two scenarios) represents the baseline for comparison, as in this case the actions for red were also determined using a game theoretic controller.

For Case 2, an open loop, heuristic-deterministic strategy was adopted for the red player, the idea behind it summarized as follows: reach the target for the red units while avoiding confrontation by taking an indirect route towards the target, such that any possible encounter with the blue units will occur as far as possible from the target of the blue units. This strategy is consistent with the weights on the payoff function selected for the blue player, except that no effort made to minimize the control effort.

For Case 3, a heuristic-stochastic strategy was adopted for the red player. Basically, a direct route was plotted for the red units to follow towards their target, with random shifts in direction, velocity and firing intensity introduced at several intervals along the route.

Finally, for Case 4 a human planner assumed the control of the red units, trying to accomplish the same

Table 2.1: List of Scenarios

	Blue	Red
Case 1 (cross 1, joust 1)	Game theoretic controller	Game theoretic controller
Case 2 (cross 2, joust 2)	Game theoretic controller	Deterministic Heuristic controller
Case 3 (cross 3, joust 3)	Game theoretic controller	Stochastic Heuristic controller
Case 4 (cross 4, joust 4)	Game theoretic controller	Human-being planner

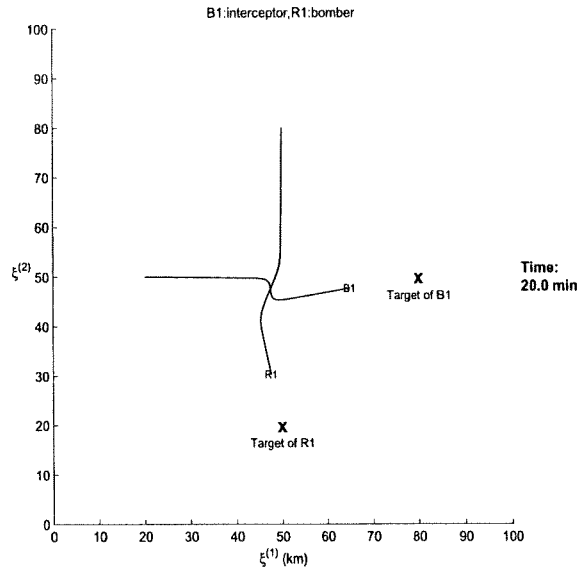


Figure 2.1: Cross 1: Trajectories

objectives as before.

2.3 Experiment Results

2.3.1 Scenario One: Cross

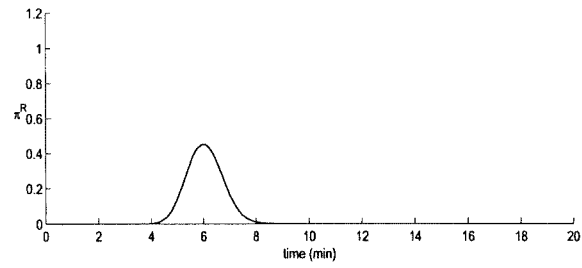
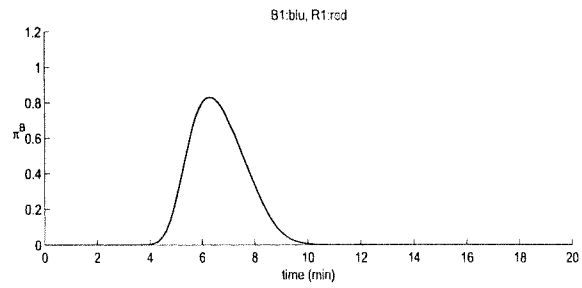


Figure 2.2: Cross 1: Firing Intensities

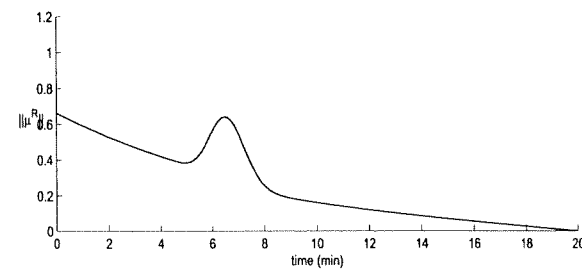
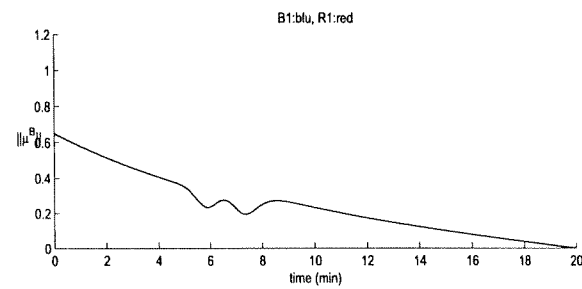


Figure 2.3: Cross 1: Velocities

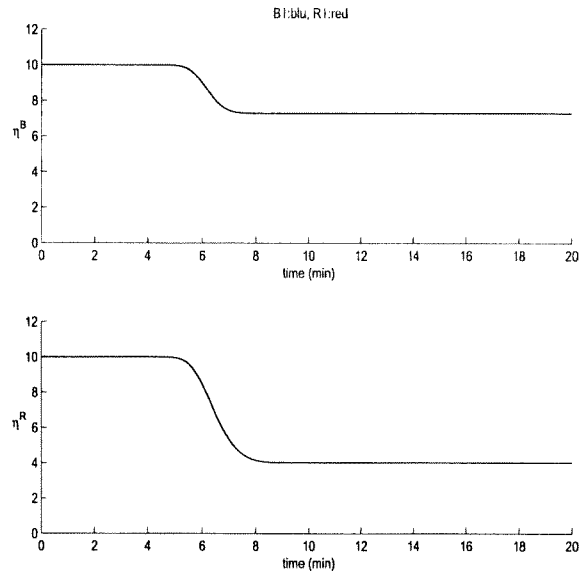


Figure 2.4: Cross 1: Number of Platforms

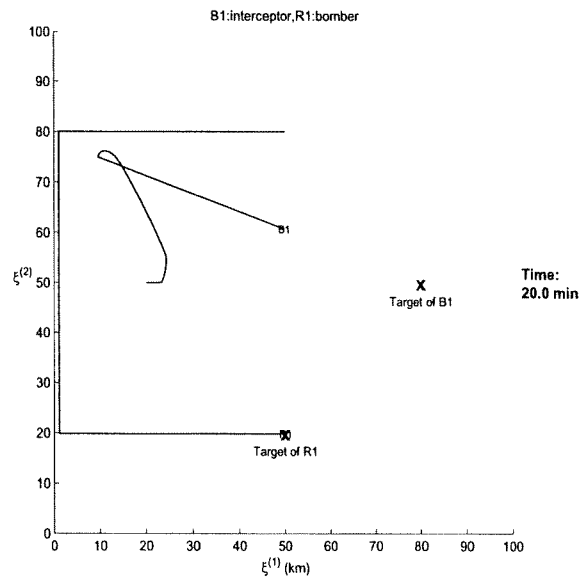


Figure 2.5: Cross 2: Trajectories

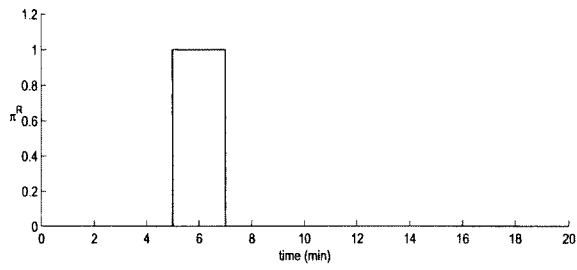
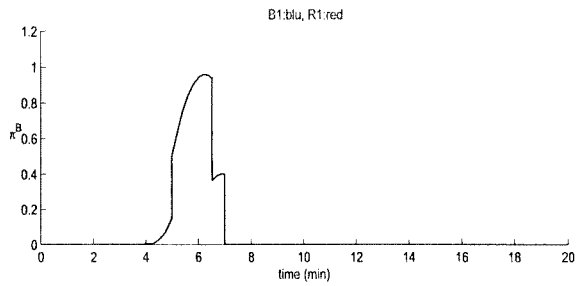


Figure 2.6: Cross 2: Firing Intensities

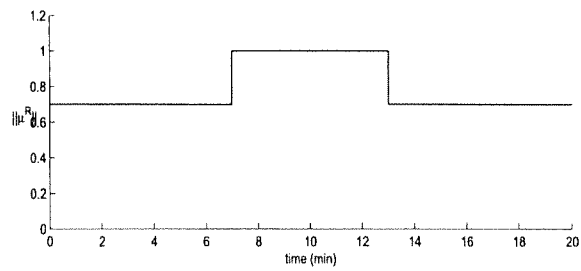
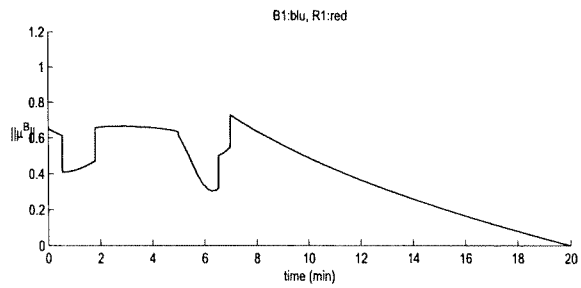


Figure 2.7: Cross 2: Velocities

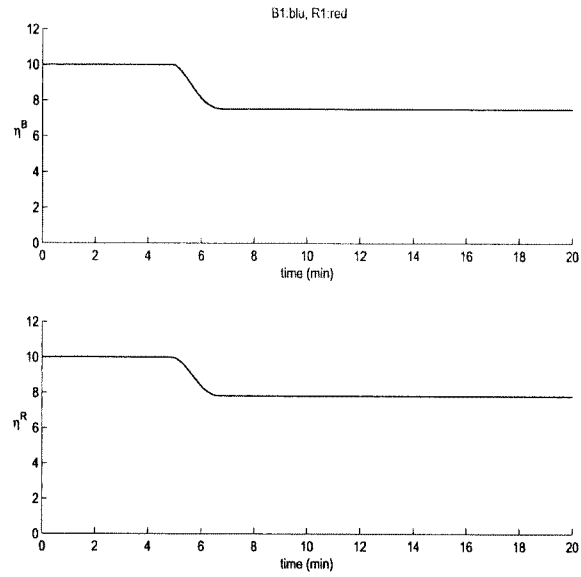


Figure 2.8: Cross 2: Number of Platforms

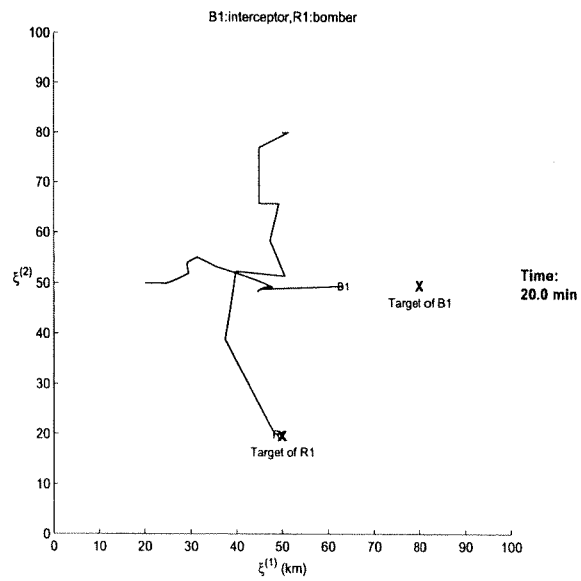


Figure 2.9: Cross 3: Trajectories

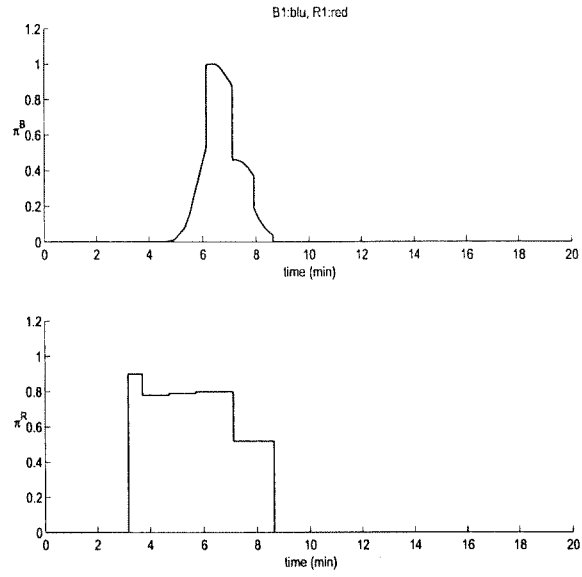


Figure 2.10: Cross 3: Firing Intensities

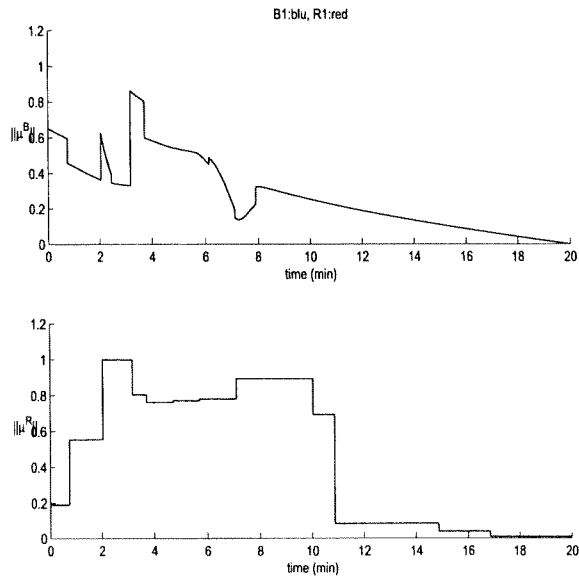


Figure 2.11: Cross 3: Velocities

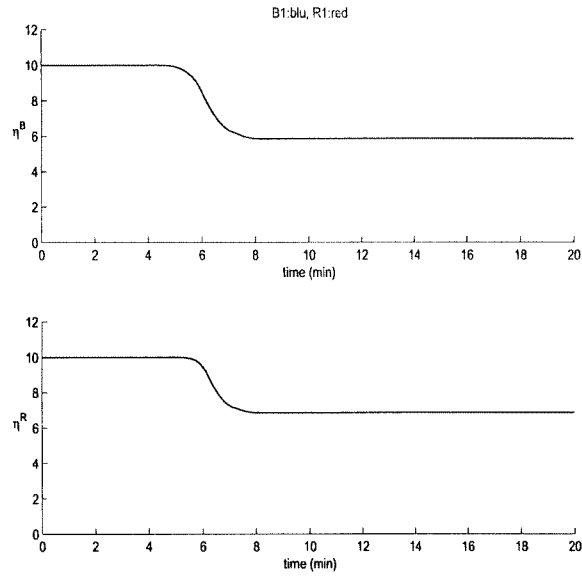


Figure 2.12: Cross 3: Number of Platforms

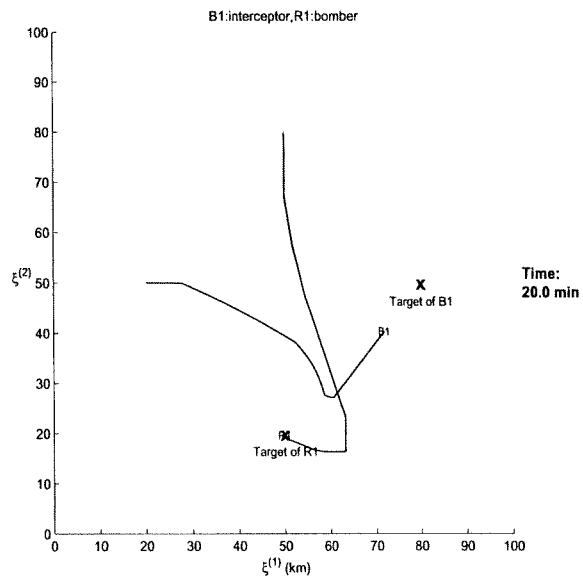


Figure 2.13: Cross 4: Trajectories

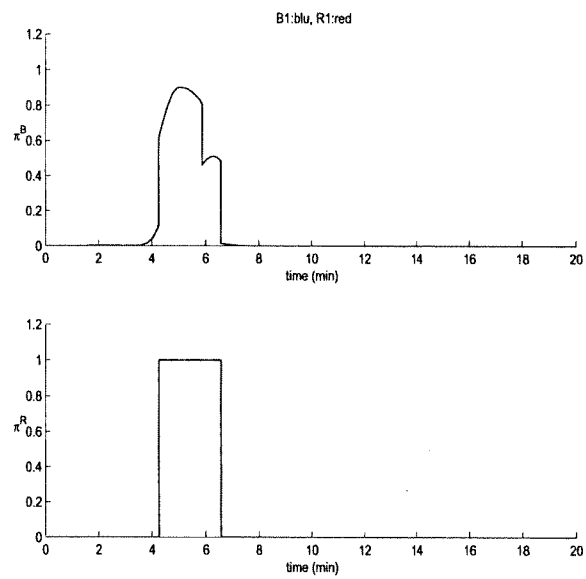


Figure 2.14: Cross 4: Firing Intensities

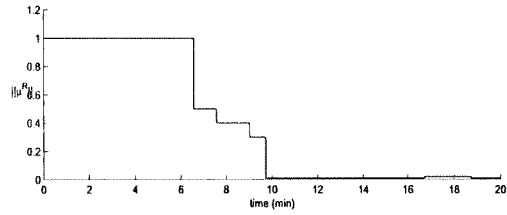
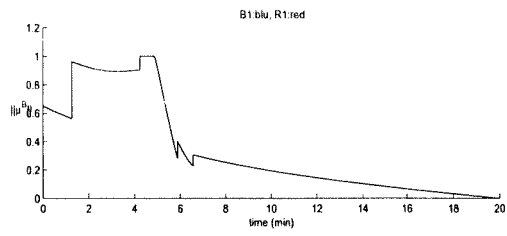


Figure 2.15: Cross 4: Velocities

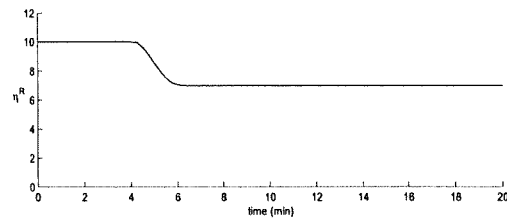
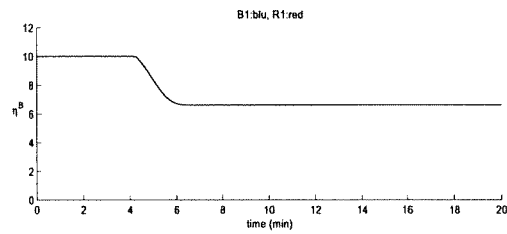


Figure 2.16: Cross 4: Number of Platforms

2.3.2 Scenario Two: Joust

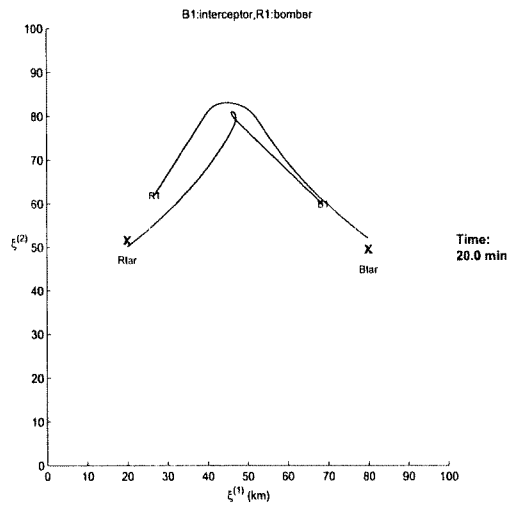


Figure 2.17: Joust 1: Trajectories

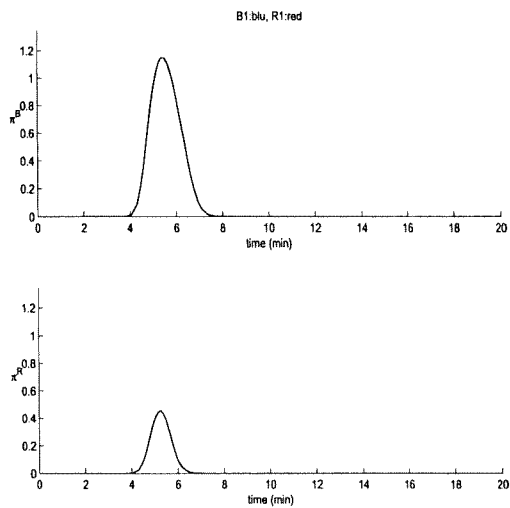


Figure 2.18: Joust 1: Firing Intensities

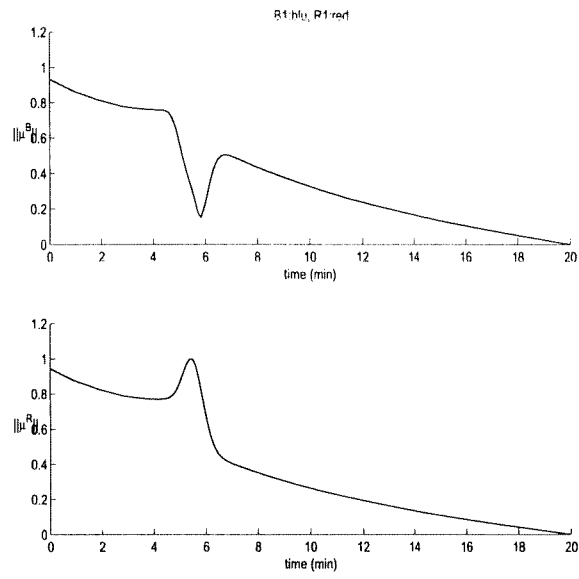


Figure 2.19: Joust 1: Velocities

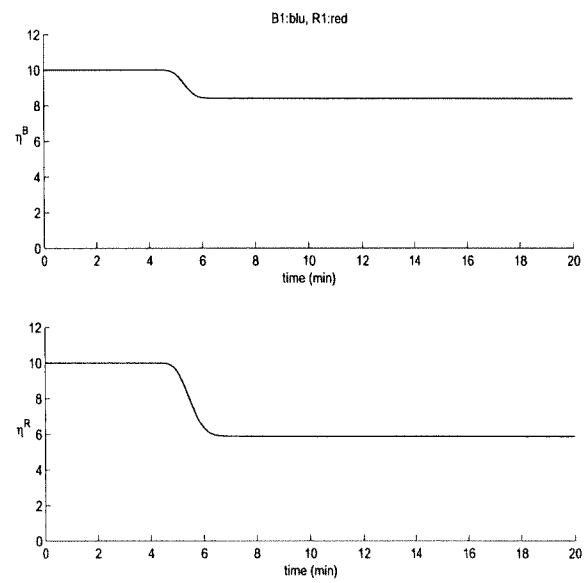


Figure 2.20: Joust 1: Number of Platforms

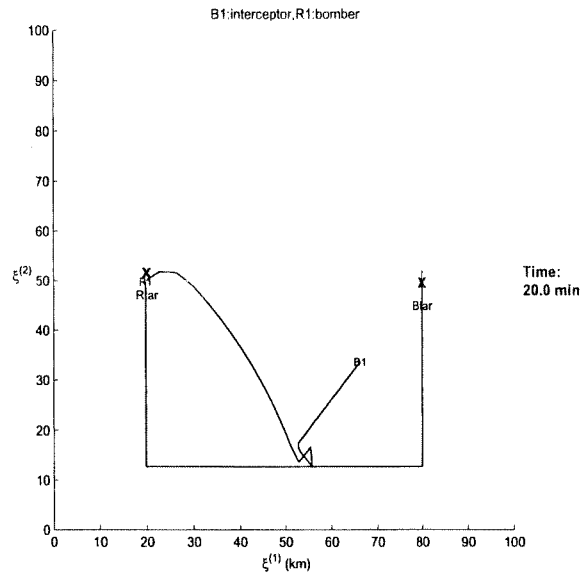


Figure 2.21: Joust 2: Trajectories

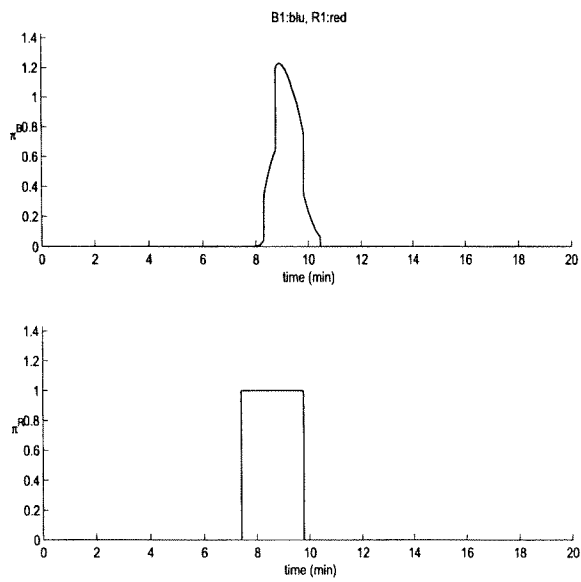


Figure 2.22: Joust 2: Firing Intensities

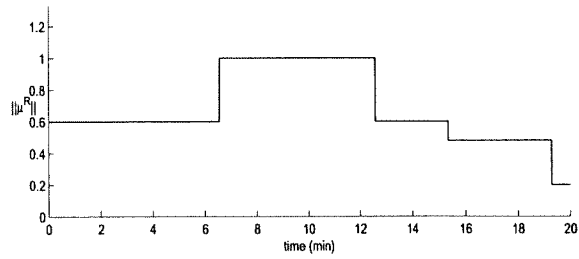
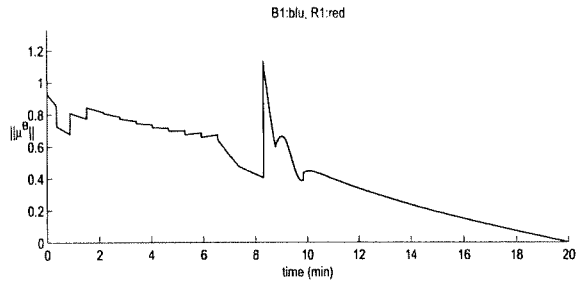


Figure 2.23: Joust 2: Velocities

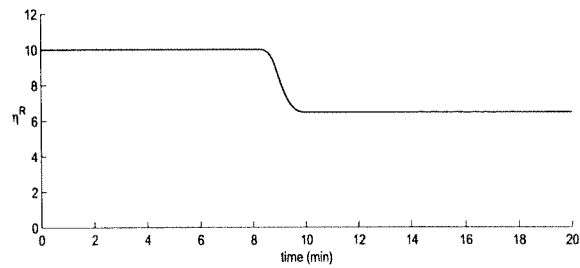
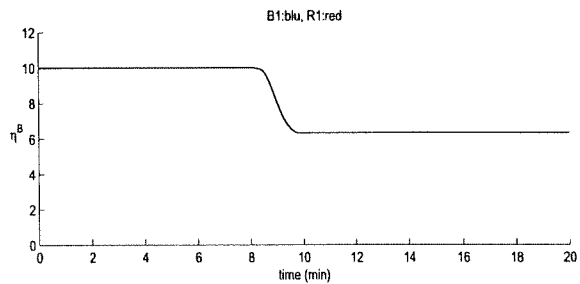


Figure 2.24: Joust 2: Number of Platforms

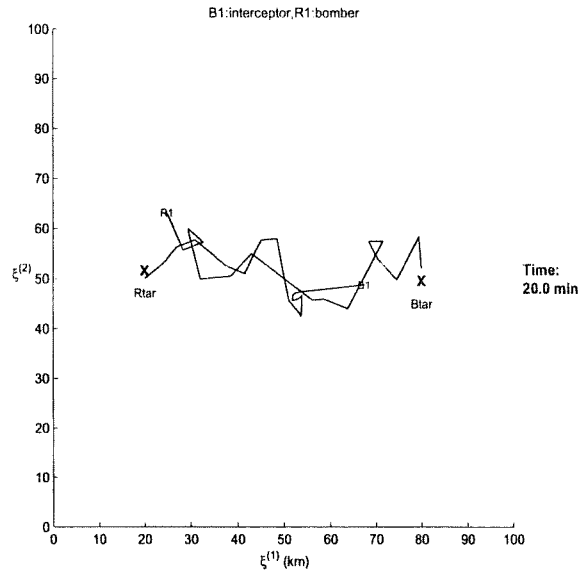


Figure 2.25: Joust 3: Trajectories

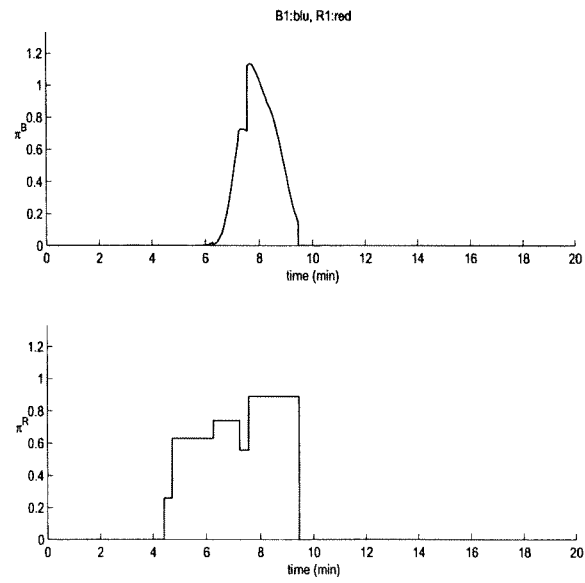


Figure 2.26: Joust 3: Firing Intensities

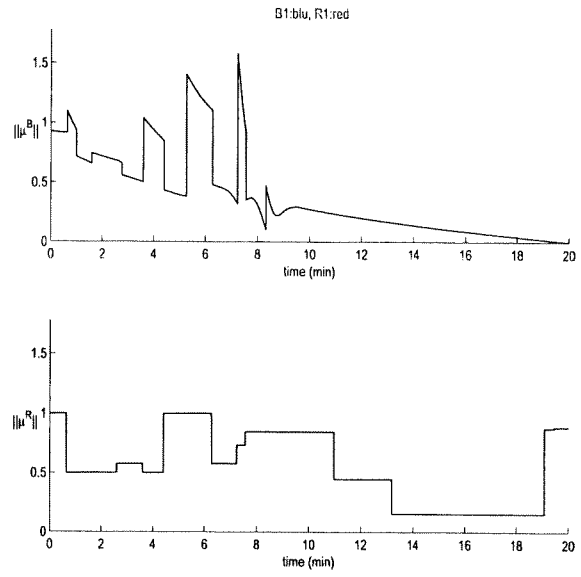


Figure 2.27: Joust 3: Velocities

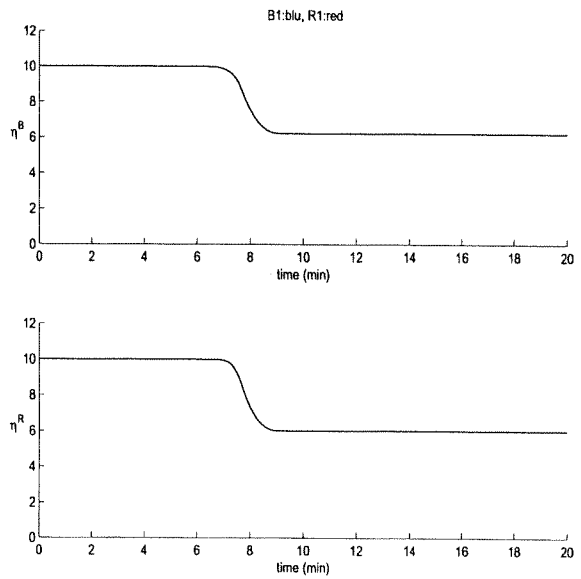


Figure 2.28: Joust 3: Number of Platforms

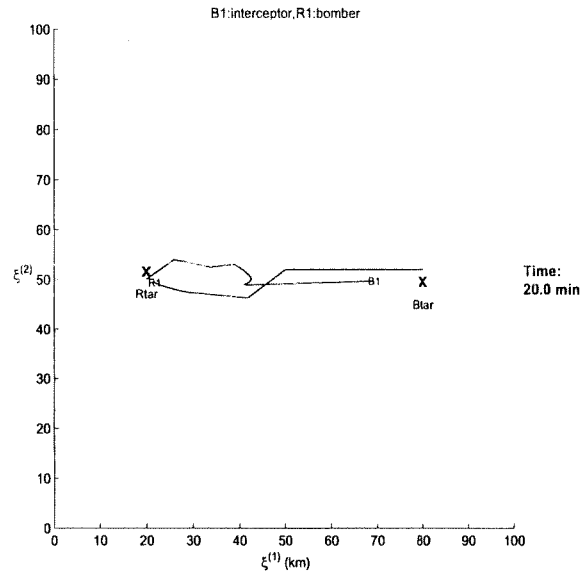


Figure 2.29: Joust 4: Trajectories

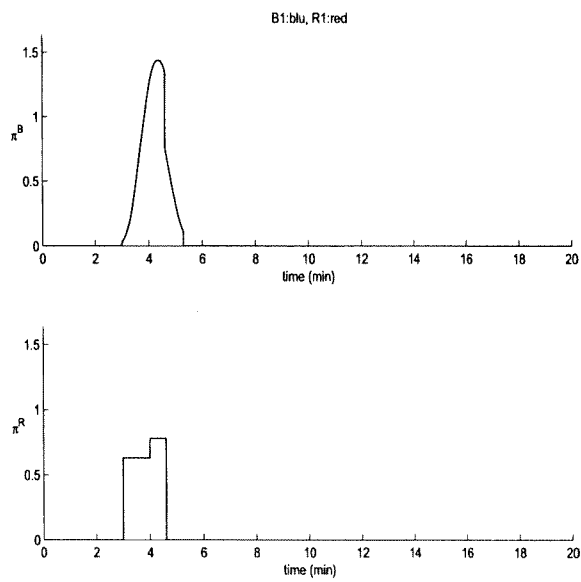


Figure 2.30: Joust 4: Firing Intensities

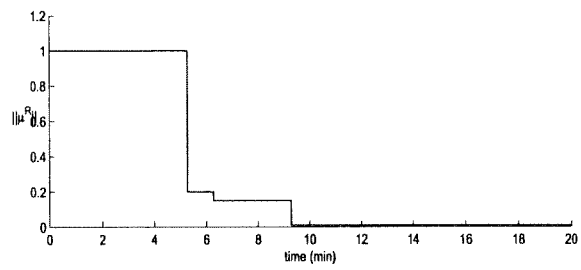
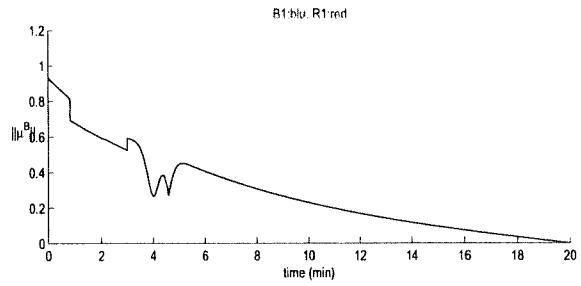


Figure 2.31: Joust 4: Velocities

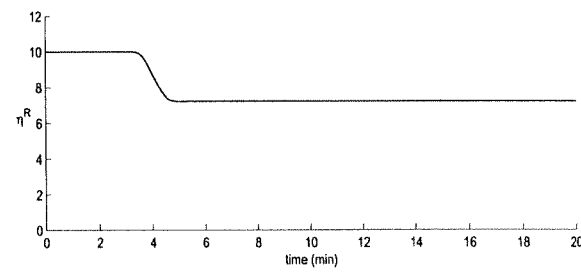
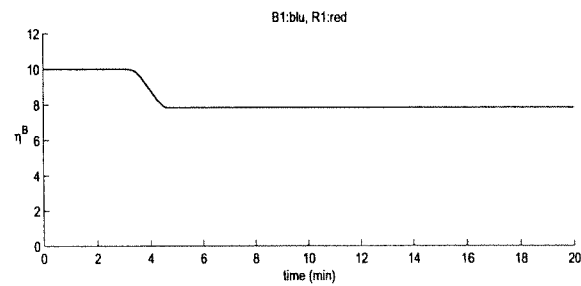


Figure 2.32: Joust 4: Number of Platforms

Table 2.2: Scenario One – Cross (Parameters Value)

	Initial Conditions			Weights		
	Platform	X_0	Y_0	Speed	Distance	Final Platform
Blue	10	80	50	200	0.1	0.2
Red	10	50	20	200	0.1	20

Table 2.3: Scenario Two – Joust (Parameters)

	Initial Conditions			Weights		
	Platform	X_0	Y_0	Speed	Distance	Final Platform
Blue	10	80	50	150	0.1	0.2
Red	10	20	50	150	0.1	20

2.4 Conclusions

Tables two and four show that using different control strategies, other than the one based on differential game theory, it was possible to improve the performance of the red player with respect to the number of platform losses and final distance to the target.

Tables three and five show that the total cost that the red player was trying to maximize, according to the game theoretic controller, was indeed maximum in Case 1, when the game theoretic controller was used to determine the strategy of both players. Furthermore, the total value of the game (red minus blue) was also a maximum for Case 1.

Table 2.4: Experiment Results for Scenario one – Parameter Values

	BLUE					Red				
	Platform	X_f	Y_f	Plat lost	Distance	Platform	X_f	Y_f	Plat lost	Distance
cross1	7.3	64.8	47.5	2.7	15.4	4.0	47.5	30.1	6.0	10.0
cross2	7.5	49.9	60.7	2.5	32.0	7.8	50.1	19.8	2.2	2.0
cross3	5.9	63.4	49.4	4.1	16.6	6.9	49.4	19.9	3.1	6.1
cross4	6.6	71.8	40.3	3.4	12.7	7.0	49.8	19.6	3.0	4.5

Table 2.5: Experiment Results for Scenario One – Cost Components

	BLUE				Red				Game Value
	Speed	Distance	Final Plat	Total	Speed	Distance	Final Plat	Total	Game Value
cross1	819	7927	-11	8735	-898	-2110	323	-2685	6050
cross2	1721	6081	-11	7791	-5346	-7	1219	-5514	2277
cross3	1033	7798	-7	8824	-2976	-2027	940	-4063	4761
cross4	1887	8111	-9	9989	-3078	-2508	975	-4611	5378

Table 2.6: Experiment Results for Scenario Two – Parameter Values

	BLUE					RED				
	Platform	X_f	Y_f	Plat lost	Distance	Plat	X_f	Y_f	Plat lost	Distance
joust1	8.4	68.9	59.9	1.6	14.9	5.9	26.8	61.9	4.1	13.7
joust2	6.3	66.3	33.7	3.7	21.3	6.5	19.8	49.7	3.5	3.6
joust3	6.2	67.5	48.7	3.8	12.6	6.0	24.9	63.1	4.0	14.0
joust4	7.8	69.4	49.7	2.2	10.6	7.2	21.8	49.3	2.8	19.3

Table 2.7: Experiment Results for Scenario Two – Cost Components

	BLUE				Red				Game Value
	Speed	Distance	Final Plat	Total	Speed	Distance	Final Plat	Total	Game Value
joust1	1360	7794	-14	9140	-1453	-2183	693	-2943	6197
joust2	1727	7473	-8	9192	-3273	-757	837	-3192	6000
joust3	1761	8038	-8	9791	-2538	-1936	724	-3750	6041
joust4	885	8068	-12	8941	-1673	-2720	1041	-3352	5589

2.5 References

[1] Sequential Linear Quadratic Method for Differential Games

[2] Mission Dynamics Continuous-time Model, Version 2.55

2.6 Appendix

2.6.1 Scenario File for Cross

```
% This scenario file is for experiment 1. Put different weight on red's terminal
% number of platforms, 7th element in g_Qf = diag([0 0 -0.20 0 0 0 20 0]);
%
%
%
% dR(1)=20 or 40 or 60.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% UNIT PROPERTIES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% numbers of categories
un.NBc=5; un.NRc=5;
% each category is represented by an integer,
% which will used for indexing into parameter matrices
% 1:ground troops 2:air defense 3:bombers
% 4:interceptors 5:SEAD
% numbers of units
NB_u=1; NR_u=1; un.NBu=NB_u; un.NRu=NR_u;
% categories of units for each force
% Blue 1 is a bomber unit, Blue 2 is an interceptor unit
% Red 1 is a ground troop unit, Red 2 is an interceptor unit
un.cB=[4]; un.cR=[4];
% descriptive names of units
un.nameB = {'fighter'}; un.nameR = {'interceptor'};
% Assume that one unit can attack only one enemy unit,
% but one unit can be attacked by multiple units
% In this example, B1 shoots at R1, B2 shoots at R2
%
% R1 shoots at B1, R2 shoots at B2
% matrix form
un.fB=[1]; un.fR=[1];
% from the shooter's perspective:
% Blue unit i shoots at all the Red units with indices un.fsB{i}
un.fsB={1}; un.fsR={1};
% from the shootee's perspective:
% Blue unit i is shoted at by all the Red units with indices
% un.feB{i}
un.feB={1}; un.feR={1};
% state vectors for each force
% xB=[xiB(1,1); xiB(1,2); etaB(1); zetaB(1); xiB(2,1); xiB(2,2);
% etaB(2); zetaB(2)]
% state vector for the whole system
% xsys=[xB; xR]
% initial conditions for states
% [xiB(1,1); xiB(1,2); etaB(1); zetaB(1); xiB(2,1);
% xiB(2,2); etaB(2); zetaB(2)]
```

```

xiB(1,1)=20; xiB(1,2)=50; etaB(1)=10; zetaB(1)=10; xiR(1,1)=50;
xiR(1,2)=80; etaR(1)=10; zetaR(1)=10; xB_init=[xiB(1,:)]';
etaB(1); zetaB(1)]; xR_init=[xiR(1,:)]'; etaR(1); zetaR(1)];
x_init=[xB_init; xR_init];
% control vector for a Blue unit i

% uB_i=[muB_ix; muB_iy; piB_i ]
% control vector for the Blue force
% uB=[uB(1); uB(2); ...; uB_NBu ];
% control vector for the whole system
% usys=[uB ; uR]
% parameters for plant simulation blocks, do not edit
% control constraints
numinputs = 3*(un.NBu+un.NRu); numstates = 4*(un.NBu+un.NRu);
contr_uplim = ones(1,numinputs); contr_lolim = []; for
i=1:un.NBu+un.NRu,
    contr_lolim = [contr_lolim, -1, -1, 0];
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CONSTANT PARAMETERS (WHICH DEPEND ON THE SCENARIO)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% maximum speeds, km/min
pm.alphaB=[0.5; 0; 10; 10; 10]; pm.alphaR=[0.5; 0; 10; 10; 10];
% parameter in probability of engagement function
pm.sigmaB=ones(un.NBc,un.NRc); pm.sigmaR=ones(un.NRc,un.NBc);
% Example: sigma for Blue unit i against Red unit j is
% pm.sigmaB(un.cB(i),un.cR(j))
% modification factor for number of engagements
pm.betaengB=ones(un.NBc,un.NRc); pm.betaengR=ones(un.NRc,un.NBc);
% modification factor for prob kill of a weapon
pm.betawepB=ones(un.NBc,un.NRc); pm.betawepR=ones(un.NRc,un.NBc);
% prob kill of weapon type i against platform type j
pm.pkillB=0.8*ones(un.NBc,un.NRc);
pm.pkillR=0.8*ones(un.NRc,un.NBc);
% salvo size of platform type i shooting at platform type j
pm.salvoB=ones(un.NBc,un.NRc); pm.salvoR=ones(un.NRc,un.NBc);
% parameter of the distance factor (varphi) function
pm.rzeroB=5*ones(un.NBc,un.NRc); pm.rzeroR=5*ones(un.NRc,un.NBc);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TERRAIN INFORMATION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% the rectangular zone for the mission
% coordinates of lower left and upper right corners, in km
% zone_lim= [ xmin xmax;
%             ymin ymax ]
%tr.zone_lim=[ 0 100;
%              0 100];
tr.zone_lim=[[0;0],[100;100]];
% obstacle locations, radii and names
tr.NBo=0; tr.NRo=0;
tr.obsB(1).loc=[0 0]; % x and y, in km
tr.obsB(1).rad=0; tr.obsB(1).name='';
tr.obsR(1).loc=[0 0]; % x and y, in km

```

```

tr.obsR(1).rad=0; tr.obsR(1).name='';
% fixed target locations, sizes and names
tr.NBt=1; tr.NRt=1; qB(1,1)=80; qB(1,2)=50;
tr.tarB(1).loc=[qB(1,1) , qB(1,2)]; % x and y, in km
tr.tarB(1).size=0; % not used in this version
tr.tarB(1).name='B1target'; qR(1,1)=50; qR(1,2)=20;
tr.tarR(1).loc=[qR(1,1) qR(1,2)]; % x and y, in km
tr.tarR(1).size=0; tr.tarR(1).name='R1target';
% initial and final times for mission, unit of time is 1 min
t_initfin=[0 ; 20];
% use all caps for global variables or make them stand out in some way
global NOM_INPUTTRAJ NOM_STATETRAJ NOM_T
load simpnominal1; % load the nominal trajectory
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% weights in the cost function of the nonlinear-quadratic game
% min max J(uB,uR)
% uB uR
%
% J(uB,uR) = (1/2)*
%integral[ti,tf]{x'*Q*x + 2*q'*x + 2*r1'*uB - 2*r2'*uR +
%uB'*R1*uB - uR'*R2*uR }
% + (1/2)*x(tf)'*Qf*x(tf) + qf'*x(tf)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global g_Q g_q g_r1 g_r2 g_R1 g_R2 g_Qf g_qf
% [xiB(1,1); xiB(1,2); etaB(1); zetaB(1); xiB(2,1);
%xiB(2,2); etaB(2); zetaB(2)]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% weights in the cost function of the nonlinear-quadratic game(MDCM2.55)
% min max J(uB,uR)
% uB uR
%
%J(uB,uR)=integral[t0,tf]{sum_{i=1}^{NB_u}aB(i)*(||xiB(i)(t)-
%qB(i)(t)||)^2-bB(i)*(etaB(i)(t))^2
% -sum_{k=1}^{NB_o} pB(i,k)||xiB(i)(t)-eB(k)||^2+{uB(i)(t)}'RB(i)uB(i)(t))
% -sum_{j=1}^{NR_u}aR(j)*(||xiR(j)(t)-qR(j)(t)||)^2-br(j)*(etaR(j)(t))^2
% -sum_{l=1}^{NR_o} pR(j,l)||xiR(j)(t)-eR(l)||^2+{uR(j)(t)}'RR(i)uR(j)(t))
% +sum_{i=1}^{NB_u}cB(i)*(||xiB(i)(tf)-qB(i)(tf)||)^2-dB(i)*(etaB(i)(tf))^2)
% -sum_{i=1}^{NR_u}cR(i)*(||xiR(i)(tf)-qR(i)(tf)||)^2-dR(i)*(etaR(i)(tf))^2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
aB(1)=0.05;%weight on the distance between blue 1 and its target.
bB(1)=0; %weight on the number of blue 1's platform(running cost).
aR(1)=0.05;%weight on the distance between red 1 and its target.
bR(1)=0; %weight on the number of red 1's platform(running cost).
%weights on control command,velocity and firing intensity,for blue 1.
RB_1=[800 0 0 ; 0 800 0 ; 0 0 200];%weights on control command for blue 1.
RR_1=[800 0 0 ; 0 800 0 ; 0 0 200];%weights on control command for red 1.
cB(1)=0;%weight on the distance between blue 1 and its target at final time.
dB(1)=0.1;%weight on the terminal number of blue 1's platforms.
cR(1)=0;
dR(1)=10;% or 4o or 60.
Qvec=2*[aB(1); aB(1); -bB(1); 0; -aR(1); -aR(1); bR(1); 0];
Qvecf=2*[cB(1); cB(1); -dB(1); 0; -cR(1); -cR(1); dR(1); 0 ]; g_Q
= diag(Qvec); g_q =2*[-aB(1)*qB(1,1); -aB(1)*qB(1,2); 0 ; 0;

```

```

aR(1)*qR(1,1); aR(1)*qR(1,2); 0 ; 0 ]; g_r1 = zeros(3*Nb_u,1);
g_R1 = diag([diag([RB_1])]); g_Qf = diag(Qvecf); g_qf =
2*[-cB(1)*qB(1,1); -cB(1)*qB(1,2); 0 ; 0; cR(1)*qR(1,1);
cR(1)*qR(1,2); 0 ; 0];

```

2.6.2 Scenario File for Joust

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% UNIT PROPERTIES
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% numbers of categories
un.NBc=5; un.NRc=5;
% each category is represented by an integer,
% which will used for indexing into parameter matrices
% 1:ground troops 2:air defense 3:bombers
% 4:interceptors 5:SEAD
% numbers of units
un.NBu=1; un.NRu=1;
% categories of units for each force
% Blue 1 is a bomber unit, Blue 2 is an interceptor unit
% Red 1 is a ground troop unit, Red 2 is an interceptor unit
un.cB=[4]; un.cR=[4];
% descriptive names of units
un.nameB = {'interceptor'}; un.nameR = {'bomber'};
% Assume that one unit can attack only one enemy unit,
% but one unit can be attacked by multiple units
% In this example, B1 shoots at R1, B2 shoots at R2
% R1 shoots at B2, R2 shoots at B1
% matrix form
un.fB=[1]; un.fR=[1];
% from the shooter's perspective:
% Blue unit i shoots at all the Red units with indices un.fsB{i}
un.fsB={1 }; un.fsR={1 };
% from the shootee's perspective:
% Blue unit i is shoted at by all the Red units with indices un.feB{i}
un.feB={ 1}; un.feR={1 };
% state vectors for each force
% xB=[xiB_1x; xiB_1y; etaB_1; zetaB_1; xiB_2x; xiB_2y; etaB_2; zetaB_2]
% state vector for the whole system
% xsys=[xB; xR]
% initial conditions for states
% [xiB_1x; xiB_1y; etaB_1; zetaB_1; xiB_2x; xiB_2y; etaB_2; zetaB_2]
xB_init=[20; 50; 10; 10 ]; xR_init=[80; 52;
10; 10 ]; x_init=[xB_init; xR_init];
% control vector for a Blue unit i
% uB_i=[muB_ix; muB_iy; piB_i ]
% control vector for the Blue force
% uB=[uB_1; uB_2; ...; uB_NBu ];
% control vector for the whole system
% usys=[uB ; uR]
numinputs = 3*(un.NBu+un.NRu); numstates = 4*(un.NBu+un.NRu);
contr_uplim = ones(1,numinputs); contr_lolim = []; for
i=1:un.NBu+un.NRu,

```

```

    contr_lolim = [contr_lolim, -1, -1, 0];
end;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% CONSTANT PARAMETERS (WHICH DEPEND ON THE SCENARIO)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% maximum speeds, km/min
pm.alphaB=[0.5; 0; 10; 10; 10]; pm.alphaR=[0.5; 0; 10; 10; 10];
% parameter in probability of engagement function
pm.sigmaB=ones(un.NBc,un.NRc); pm.sigmaR=ones(un.NRc,un.NBc);
%Example:sigma for Blue unit i against Red unit j is
% pm.sigmaB(un.cB(i),un.cR(j))
% modification factor for number of engagements
pm.betaengB=ones(un.NBc,un.NRc); pm.betaengR=ones(un.NRc,un.NBc);
% modification factor for prob kill of a weapon
pm.betawepB=ones(un.NBc,un.NRc); pm.betawepR=ones(un.NRc,un.NBc);
% prob kill of weapon type i against platform type j
pm.pkillB=0.8*ones(un.NBc,un.NRc);
pm.pkillR=0.8*ones(un.NRc,un.NBc);
% salvo size of platform type i shooting at platform type j
pm.salvoB=ones(un.NBc,un.NRc); pm.salvoR=ones(un.NRc,un.NBc);
% parameter of the distance factor (varphi) function
pm.rzeroB=5*ones(un.NBc,un.NRc); pm.rzeroR=5*ones(un.NRc,un.NBc);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% TERRAIN INFORMATION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% the rectangular zone for the mission
% coordinates of lower left and upper right corners, in km
% zone_lim= [ [xmin;ymin] , [xmax;ymax] ]
tr.zone_lim=[[0;0],[100;100]];
% obstacle locations, radii and names
tr.NBo=0; tr.NRo=0;
tr.obsB(1).loc=[0 0];      % x and y, in km
tr.obsB(1).rad=0; tr.obsB(1).name='';
tr.obsR(1).loc=[0 0];      % x and y, in km
tr.obsR(1).rad=0; tr.obsR(1).name='';
% fixed target locations, sizes and names
tr.NBt=1; tr.NRt=1; q1B=80;q2B=50; tr.tarB(1).loc=[q1B q2B];
%tr.tarB(1).loc=[80 50];      % x and y, in km
tr.tarB(1).size=0;          % not used in this version
tr.tarB(1).name='Btarget'; q1R=20; q2R=52;
tr.tarR(1).loc=[q1R q2R];   % x and y, in km
%tr.tarR(1).loc=[20 32];
tr.tarR(1).loc=[q1R q2R];   % x and y, in km
tr.tarR(1).size=0; tr.tarR(1).name='Rtarget';
% initial and final times for mission, unit of time is 1 min
t_initfin=[0 ; 20];
% use all caps for global variables or make them stand out in some way
global NOM_INPUTTRAJ NOM_STATETRAJ NOM_T LASTFB
load joust_nominal; % load the nominal trajectory
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% weights in the cost function of the nonlinear-quadratic game
% min max J(uB,uR)
% uB uR

```

```

%
% J(uB,uR) = (1/2)*
%   integral[ti,tf]{x'*Q*x + 2*q'*x + 2*r1'*uB - 2*r2'*uR
%+ uB'*R1*uB - uR'*R2*uR }
%   + (1/2)*x(tf)'\*Qf*x(tf) + qf'*x(tf)
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
aB=1; aR=1; bB=10; bR=10; global g_Q g_q g_r1 g_r2 g_R1 g_R2 g_Qf

g_qf

%   [xiB_1x; xiB_1y; etaB_1; zetaB_1; xiB_2x; xiB_2y; etaB_2; zetaB_2]
Qvec = 0.1*[1 ;    1 ;    0 ;    0 ;    -1 ;    -1 ;    0 ;
0 ];
%Qvec = [aB;aB;-bB;0;-aR;-aR;bR;0];
g_Q = diag(Qvec);

%g_q = zeros(8,1);

g_q = 0.1*[-80; -50; 0; 0; 20; 52; 0; 0]; g_r1 = zeros(3,1); g_r2
= zeros(3,1); g_R1 = 150*diag([4 4 2]); g_R2 = 150*diag([4 4 2]);
g_Qf = diag([0 0 -0.20 0 0 0 5 0]);
%
%
%
g_qf = zeros(8,1); clear aB aR bB bR;

```

Chapter 3

Experiment 3: Controller Performance under Noise in the State Observation

3.1 Executive Summary

We performed a series of experiments to evaluate the effectiveness of the current differential game technology as a means of countering enemy actions under idealized situations with perfect information about the enemy initial conditions and objectives, but with noisy measurements of the enemy state. Our main findings are that while average values look good, individual sample paths might be quite surprising. One can conclude that the game theoretic controller CPC (Controller-Plant-Controller) is sensitive to observation noise. The first step to remedy the noise problem is to implement proper filters in the controllers.

3.2 Purpose of the Experiment

The purpose of the experiments is to test the behavior of the game theoretic controller CPC (Controller-Plant-Controller) when there is noise added to the measurements of the enemy state.

3.3 Hypothesis to Prove or Disprove

The *current* differential game technology provides an effective means of countering enemy actions under *idealized* situations with *perfect* information about the enemy initial conditions and objectives, but with noisy measurements of the enemy state.

3.4 Experimental Setup

We performed a series of experiments to evaluate the effectiveness of the current differential game technology as a means of countering enemy actions under idealized situations with perfect information about the enemy initial conditions and objectives, but with noisy measurements of the enemy state. Here, both the plant and internal models are the same, i.e., the MDCM. Increasing levels of noise will be added to the state variables when constructing the observed state variables (the output variables). The control actions of the Blue and Red teams are generated by the proposed game theoretic algorithm.

The Sequential Linear Quadratic Method (SLQM), as described in [3] and [4] solves only a mathematical problem. Once the weights and the parameters are set, the method computes a solution to the differential game. This means that all the future actions of both parties are determined. They are

described in terms of functions of time with the domain being the duration of the game. In this setup neither of the parties has any initiative, aside that a third party which has the complete knowledge of all the resources for both sides and the battlefield is doing the computations.

The fact that there is only **one** intelligent entity who does all the computations is quite unnatural since in a war there are (generally) three separate entities: the friendly side, the enemy and the battlefield. The battlefield determines the rules, the dynamics of the war. The information coming from the battlefield is observed by each side with possible addition of noise and corruption and it is processed by the friendly and enemy sides using the intelligence they have.

The natural step in making the experiment setup more realistic is therefore to separate the three entities mentioned above. This results in the Controller-Plant-Controller (CPC) Setup of Figure 3.1.

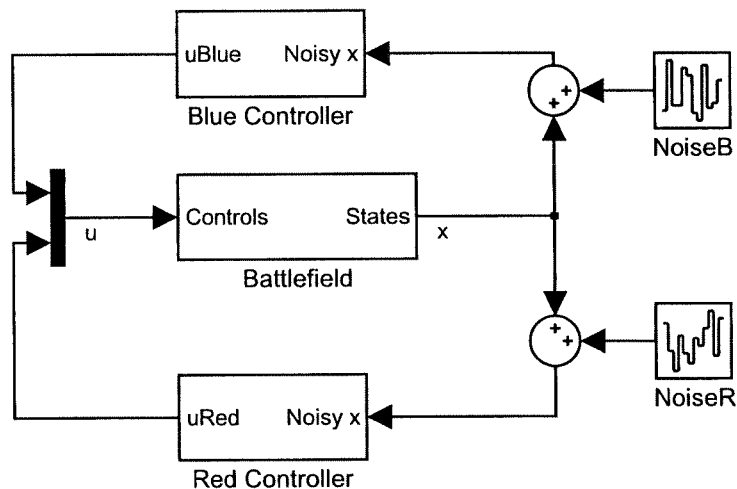


Figure 3.1: Controller-Plant-Controller Setup

The CPC setup realizes the separation of the battlefield from the enemy and friendly sides. In this way noise can be injected separately to the observation channels of each side. More important than that each side will have its own intelligence, the intelligence to compute the control inputs to the battlefield. In our case, based on the development of the game theoretical method, the intelligence for both sides is chosen to be the differential game solver based on SLQM. This means that each of the Red and Blue sides will have their model of the battlefield and the cost function, possibly with different weights, parameter mismatches, parametric uncertainties. Each side will compute the solution of the game they posed (internal to the controller) and this way will prepare their future inputs to the battlefield. It is important to note here that the sides can model the battle as they want, the implementation leaves this option free. The Blue side might be calculating its inputs to the battlefield based on the game controller schemes whereas the red side might be using another scheme like heuristic methods or artificial intelligence. The advantage of the CPC setup comes from its flexibility and modularity. The model for the battlefield is kept separate from the rest, all it needs are the inputs. It does not matter how the inputs are calculated. This also opens the possibility of changing the battlefield (plant) model and keep the same setup.

Another subtle point is about the modeling of the enemy side. For example the Blue side might be using the game technology to compute its inputs to the battlefield. The Blue side has in its internal model the Red side as the opponent in its internal game. The Red side on the other hand does not necessarily have to use the game technology, but instead might be using a different approach, say a heuristic controller. Regardless of this, since the Blue side is sticking to the game technology it would be modeling Red's actions incorporated in its game.

Pushing the content of a controller one step further, since the sides are separated they can independently implement their detection technology or any future new technology in their controllers. For

instance the weight estimator for the enemy actions which looks like a highly challenging task at this point could be implemented in the controllers of both sides in the future. Basically as the enemy actions are not known in the beginning of the battle, the enemy weights in the internal game model are unknown. Starting from a best guess for the enemy weights in the beginning of the battle the weight estimator can update them based on the past enemy actions. This “adaptive” approach is not implemented yet, but is certainly worth investigating in future. It is clear that an adaptation scheme must be carefully invented since the subject is new and not much is known.

Aside all the internals of the controllers it is clear that each side will have better information about itself from the battlefield compared to the information observed about the enemy side. This structure can easily be modeled since the setup allows injection of the noise to the state observation in any desired manner, i.e. any kind of noise can be injected to any desired state.

Implementation of the CPC set-up

Although some of the conceptual entities are not separately programmed it is very important to understand the ideas behind the implementation. The general idea to realize the setup is that each of the Blue and Red commanders has a vision about the inputs they will supply to the battlefield even before the simulation starts. This is done by setting up their own games and computing the solution by using SLQM. The inputs they supply to the battlefield are just the solutions of these separate games. This is in a sense modeling the knowledge prior to the battle. The calculations are carried out in the Game Calculator block in Figure 3.2. The input-state pair coming from the game calculator is then fed to the Storage block of Figure 3.2. It is assumed here that both sides have their intelligence chosen as the game technology, but as described above this is not necessary. All one needs to incorporate any other kind of intelligence in the simulation is just to supply the inputs it computed to the battlefield.

Once the prebattle computations are finished, the solution of the game, the inputs and the corresponding state trajectories are kept in the storage block for each side. In the beginning of the simulation, which can be visualized in Figure 3.1 both sides supply their stored input to the battlefield. In return the states corresponding to the inputs are observed from the battlefield with the addition of noise. This information is collected at the controller and it is filtered. It is compared to the stored value of the states. The storage block keeps mainly the precomputed input-state pair for each side inside the controller. It serves the purpose of sending the precomputed inputs to the battlefield as long as the value of the actual states is close to the predicted ones. This exactly means that as long as the battle takes its course as predicted, the controller just feeds the precomputed inputs. If, however, at any given moment the actual values of the states deviate from the precomputed values, the game calculator is recalled and it computes a new input-state pair using the (possibly) updated weights and sends it to the storage block. The comparison task is achieved by the Decision Block. The new computation starts from the time the deviation occurred and has as its final time the final time of the overall battle simulation. This is by no means a restriction since the time horizon does not have to be constant and can be extended further if necessary.

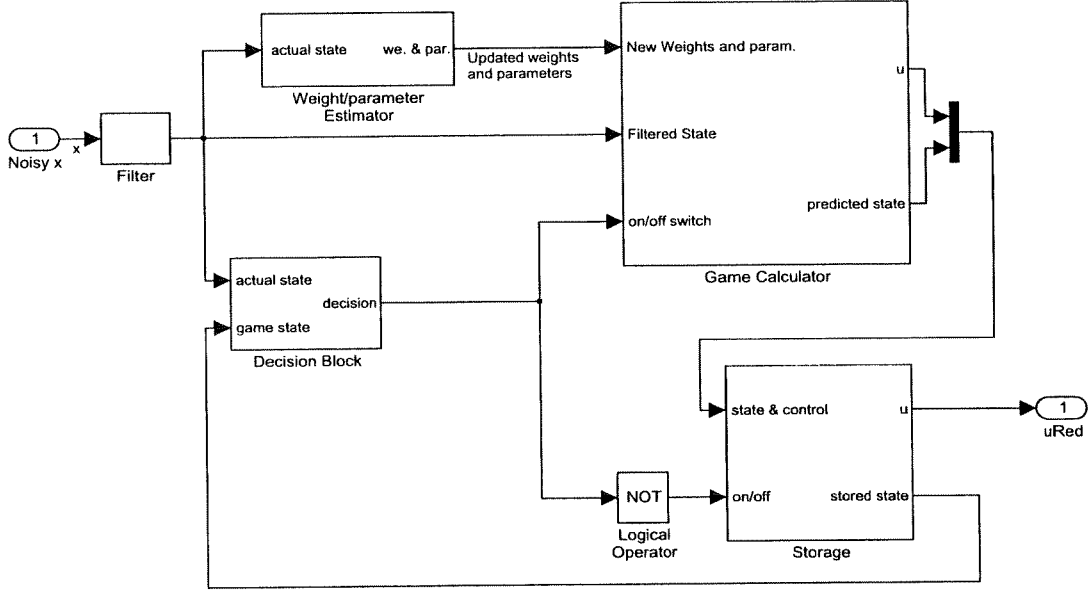


Figure 3.2: Conceptual Representation of the Controller

Game Scenario

The scenario used in all the experiments is the cross11 scenario. In which a Red Bomber is trying to reach its target on the south starting from its base on the north. The Blue unit of interceptors on the other hand starts from its base on west and flies to east to intercept the Red bomber.

The payoff function is given by

$$\begin{aligned}
 J[x; u^B, u^R] = & \frac{1}{2} \int_{t_0}^{t_f} \left[x(t)' Q(t) x(t) + 2x(t)' d(t) \right. \\
 & + u^B(t)' R^B(t) u^B(t) + 2u^B(t)' r^B(t) \\
 & \left. - u^R(t)' R^R(t) u^R(t) - 2u^R(t)' r^R(t) \right] dt \\
 & + \frac{1}{2} x(t_f)' Q_f x(t_f) + x(t_f)' r_f.
 \end{aligned} \tag{3.1}$$

All the parties, Blue side, Red side and the plant have the same model with the same parameters. The parties are using the same intelligence based on the differential game theory. In this experiment the information coming from the battlefield to the parties is noisy. The level of the noise is increased to see how the controllers will react.

As both sides compute their own internal game it does not make sense to assign the same weights to both sides. If all the weights were the same both sides would calculate the same thing twice. Therefore in all the experiments whether noise or parametric mismatch (Experiment 4) is introduced there will always be a weight mismatch between the parties. All other factors will be included in the experiments gradually. The most complicated experiments are the ones with noisy observations where there are weight differences and parameter mismatches (including plant and both controllers internal parameters, see Experiment 4). It is a good way to think of the weight assignments as the decision on the strategy whereas of the parameter mismatches (between plant and the other controller) and the noise as the observation problems.

All the weights and the parameters are the same for both parties except the final cost weights. The running costs for both sides are given as:

$$Q = \begin{bmatrix} 1/10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/10 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1/10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1/10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$d = [-8 \quad -5 \quad 0 \quad 0 \quad 5 \quad 2 \quad 0 \quad 0]$$

$$R^B = \begin{bmatrix} 800 & 0 & 0 \\ 0 & 800 & 0 \\ 0 & 0 & 200 \end{bmatrix} \quad R^R = \begin{bmatrix} 800 & 0 & 0 \\ 0 & 800 & 0 \\ 0 & 0 & 200 \end{bmatrix}$$

$$r^B = [0 \quad 0 \quad 0] \quad r^R = [0 \quad 0 \quad 0]$$

The weights which are different are the final cost matrices:

$$QB_f = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1/5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$QR_f = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1/10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 60 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

As a reminder, the plant model describes the evolution of the states ξ_1, ξ_2 , the horizontal and vertical positions, η , the number of platforms, and ζ , the number of weapons per platform with the inputs μ_1 and μ_2 describing the velocity vector and π the firing intensity respectively.

$$x = [\xi_{b1} \quad \xi_{b2} \quad \eta_b \quad \zeta_b \quad \xi_{r1} \quad \xi_{r2} \quad \eta_r \quad \zeta_r]$$

Table 3.1: The Noise Levels for the Experiments

Experiment	Noise Level
1	1%
2	2%
3	3%
4	4%
5	5%
6	6%
7	7%
8	8%
9	9%
10	10%
11	11%
12	15%
13	20%
14	25%
15	30%
16	35%
17	40%
18	50%
19	60%
20	90%
21	130%

$$u = [\mu_{b1} \quad \mu_{b2} \quad \pi_b \quad \mu_{r1} \quad \mu_{r2} \quad \pi_r] .$$

The final cost is only on the number of platforms with different weights in both parties which have different values for both the blue and the red side's internal games. Looking at the cost matrix for the Blue side it is clear that Blue puts more weight on its final number of platforms as well as on Red's final number of platforms. This can be interpreted as Blue will try to eliminate Red's platforms as much as it can but at the same time will try to preserve its platforms. The Red side on the other hand does not put too much cost on its and Blue's final number of platforms compared to the Blue's weights. This can be interpreted as Red wants to reach its target at the cost of losing its platforms and without too much engaging with blue.

3.5 Experimental Results

At this first step of the experiments the noise is added to only the observation of the enemy states. It is assumed that each side has perfect information about its own states. The noise is white noise with zero mean and is generated by matlab using the random number generator. As each component of the states has different numerical value ranges the noise injected to the same state cannot have uniform amplitude. The amplitude of the injected noise is specified relatively to the initial condition of the state. For instance if the initial horizontal position of the units is 80 and the number of the platforms is 10 with 10% noise injected the maximum value that the noise amplitude can take for the position is 8 and for the number of platforms is 1. Experiments with increasing percentage of noise level are carried out on the scenario. The level of the noise is summarized in Table 3.1.

Figure 3.3 represents the development of the battle with perfect observation. Figures 3.4 – 3.7 show the average value of 100 sample paths computed. The states and the controls are plotted, in addition to that the times when the deviation becomes too much and the game is recomputed is plotted in the Controller Times graph. The value of the controller times function is either zero or one and the average value over the sample paths is presented in the graph. If one reads 0.4 at 6th minute this means that

out of 100 sample paths the deviation has exceeded the threshold and the game calculator was activated 40 times. Comparing the average values for different levels of noise, it is clear that they look similar in general. The interesting part is the controller times function. It is observed from its average value that the controller was called for high level of noise for a small number of sample paths during almost all the battle time.

A better understanding will be reached when certain sample paths are observed for different noise levels.

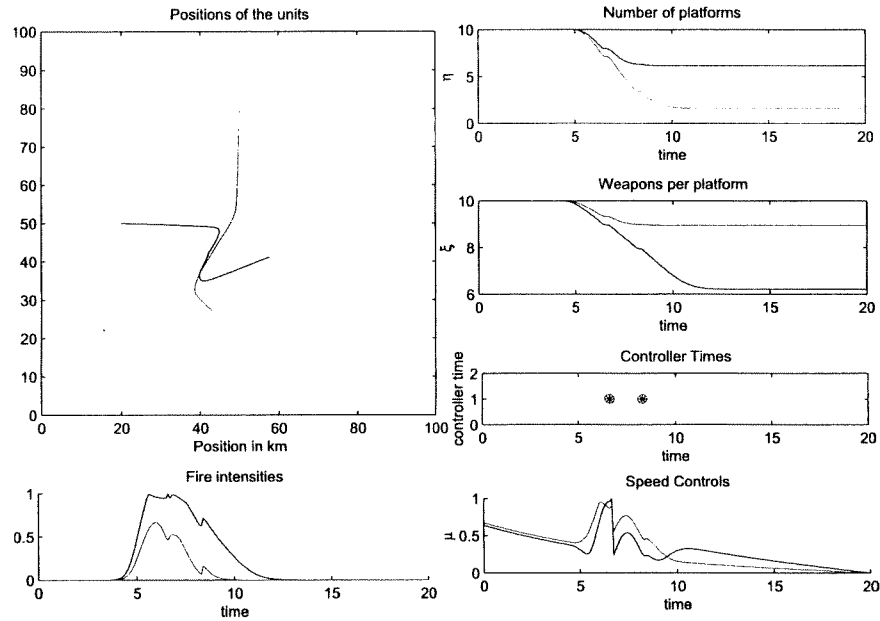


Figure 3.3: The Scenario without any Noise

Figures 3.8–3.14 show a couple of sample paths for different noise levels. When the level of the noise is low there is not a big problem. However as the amplitude of the noise is increased there are problematic sample paths as shown in Figures 3.10, 3.12, 3.13. It can be asserted in a figurative manner that although on the average the results are looking good there is a strong standard deviation.

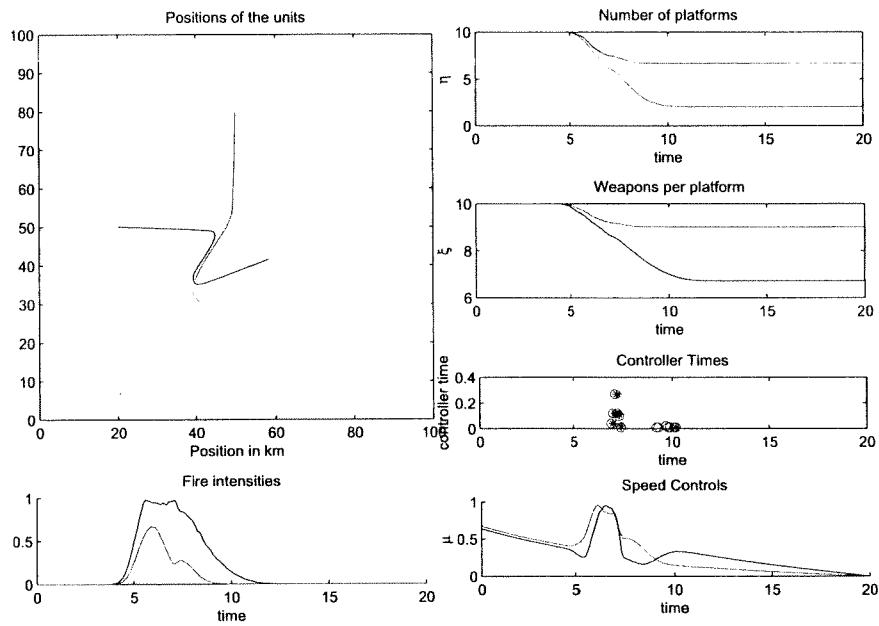


Figure 3.4: Average Values of the States and Controls over 100 Sample Paths for the Noise Amplitude 1%

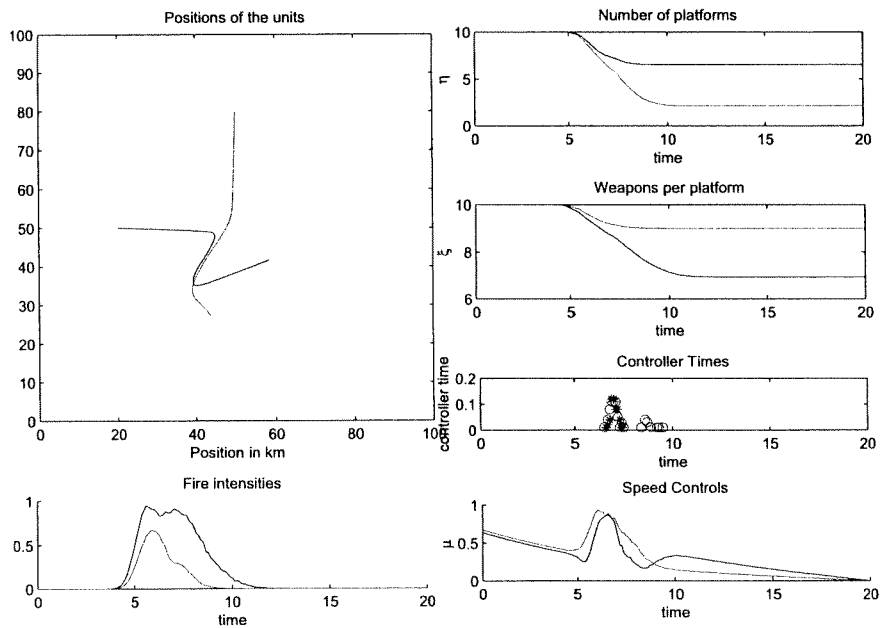


Figure 3.5: Average Values of the States and Controls over 100 Sample Paths for the Noise Amplitude 10%

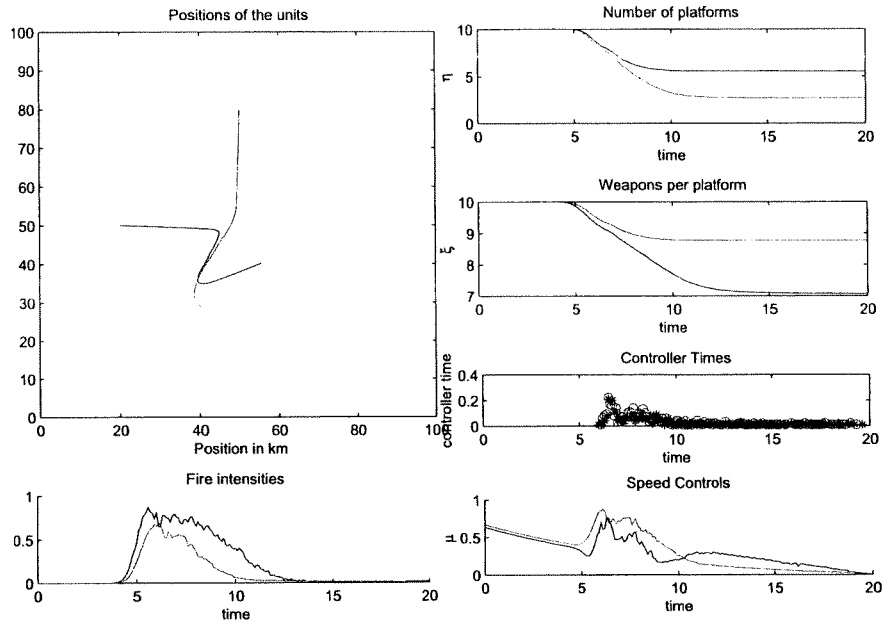


Figure 3.6: Average Values of the States and Controls over 100 Sample Paths for the Noise Amplitude 90%

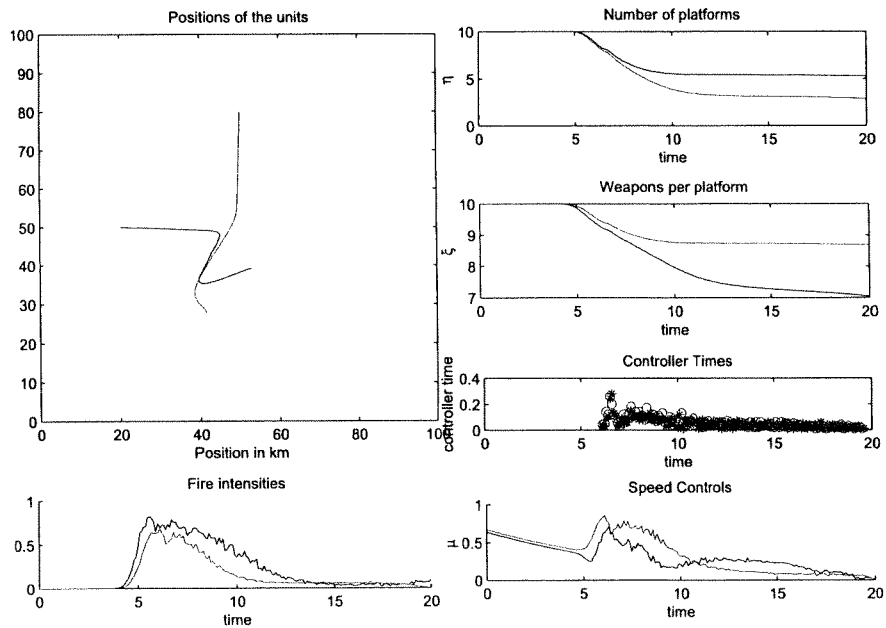


Figure 3.7: Average Values of the States and Controls over 100 Sample Paths for the Noise Amplitude 130%

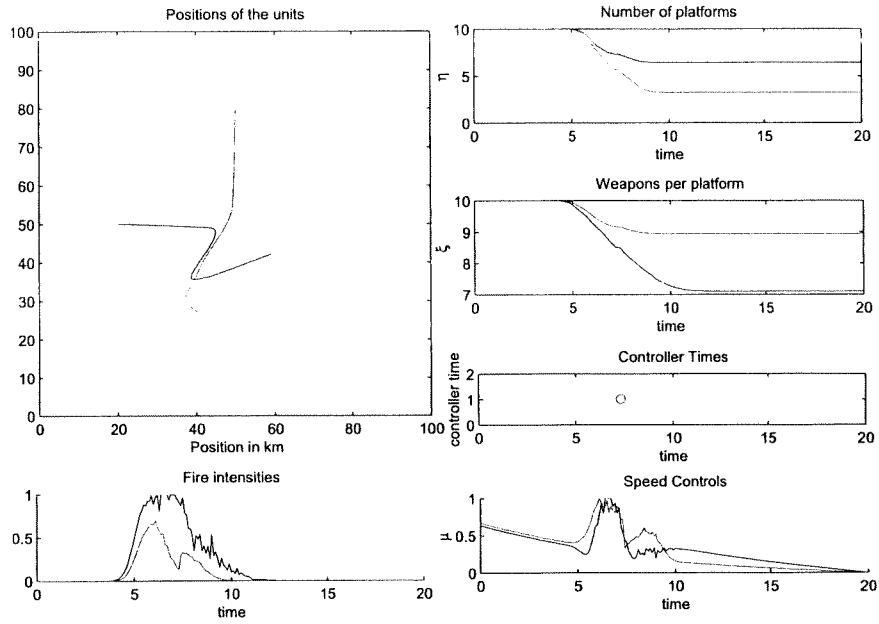


Figure 3.8: A Sample Path (Noise Amplitude 1%)

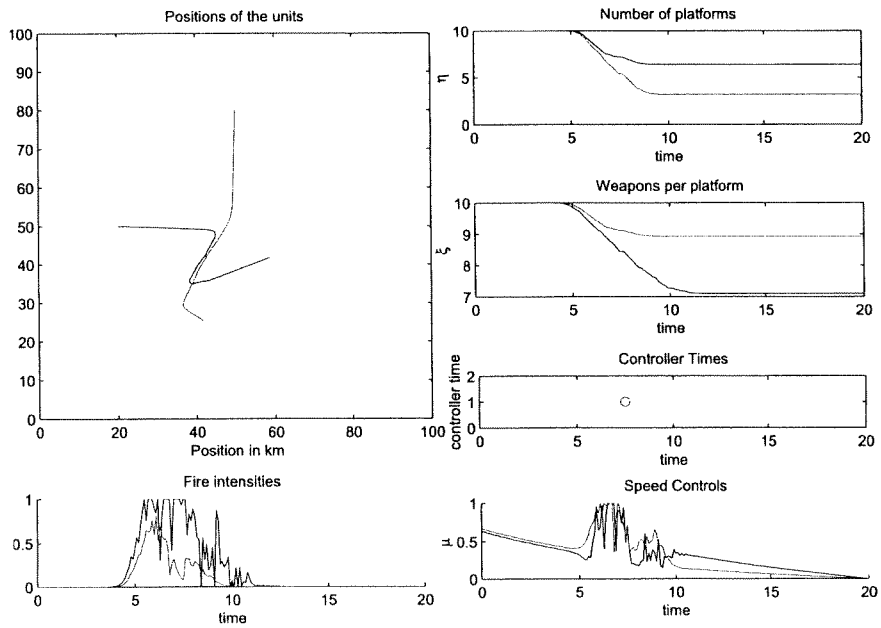


Figure 3.9: A Sample Path (Noise Amplitude 10%)

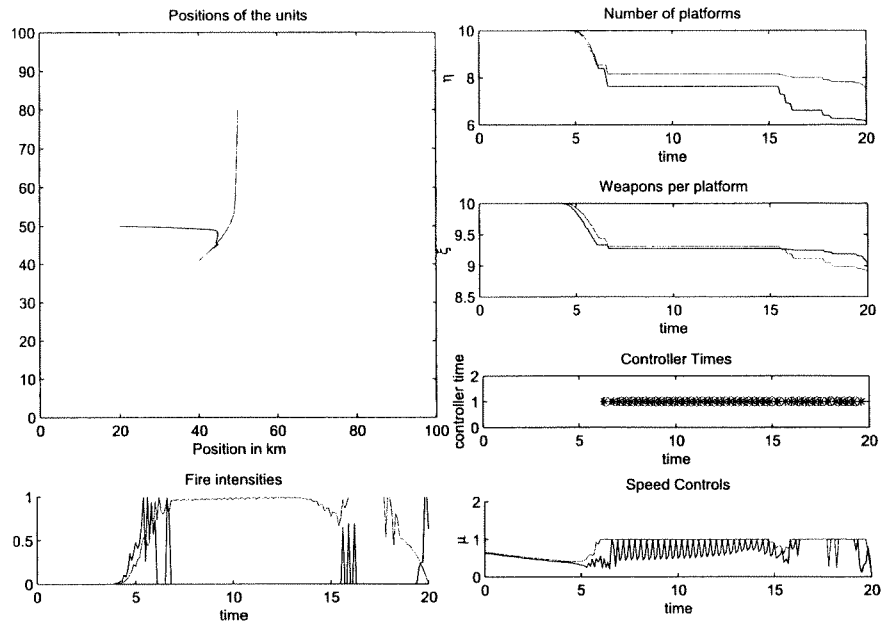


Figure 3.10: A Sample Path (Noise Amplitude 90%)

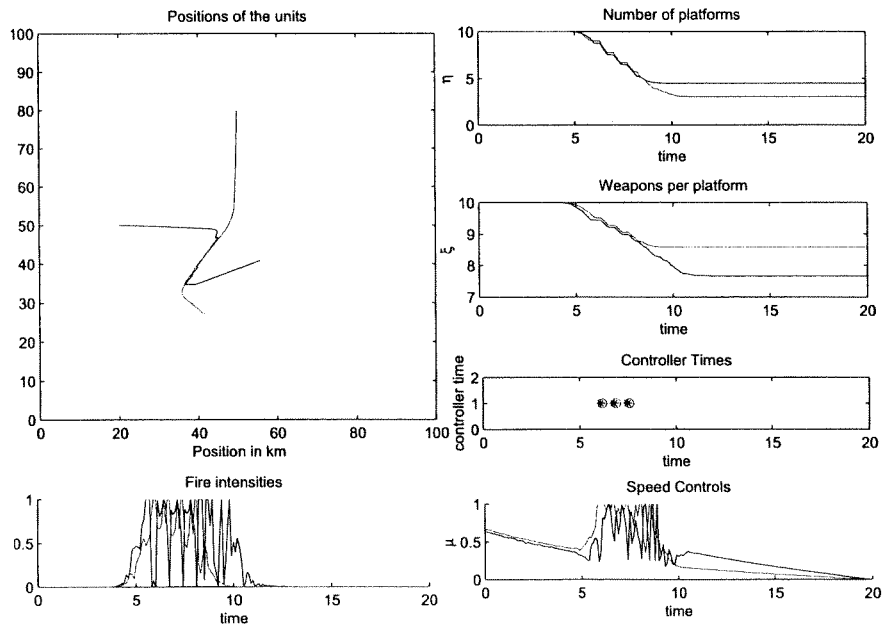


Figure 3.11: A Sample Path (Noise Amplitude 90%)

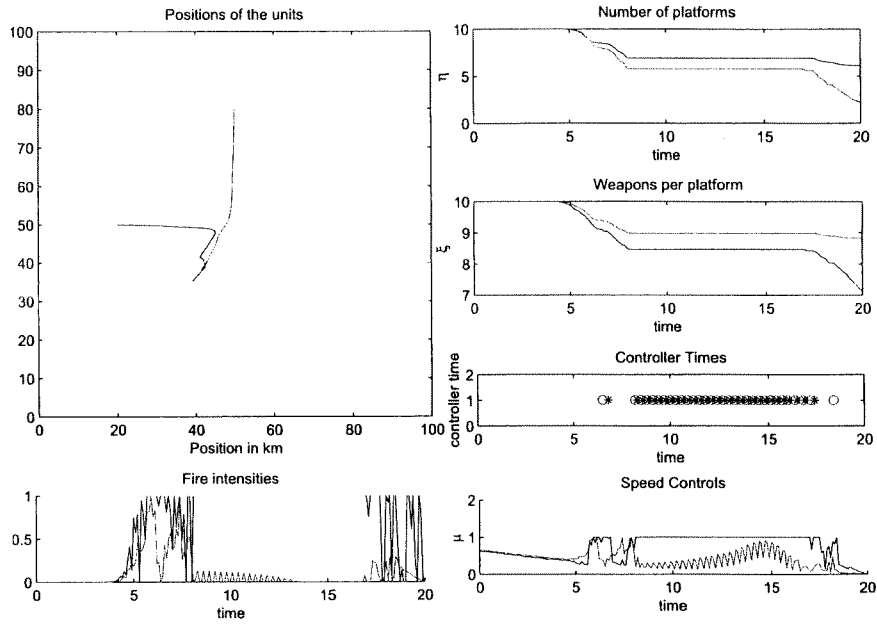


Figure 3.12: A Sample Path (Noise Amplitude 130%)

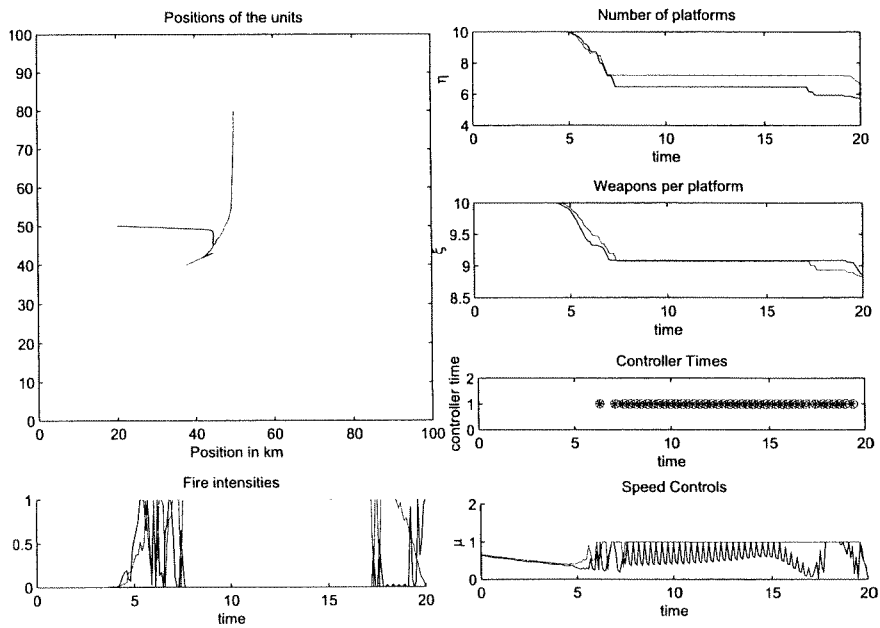


Figure 3.13: A Sample Path (Noise Amplitude 130%)

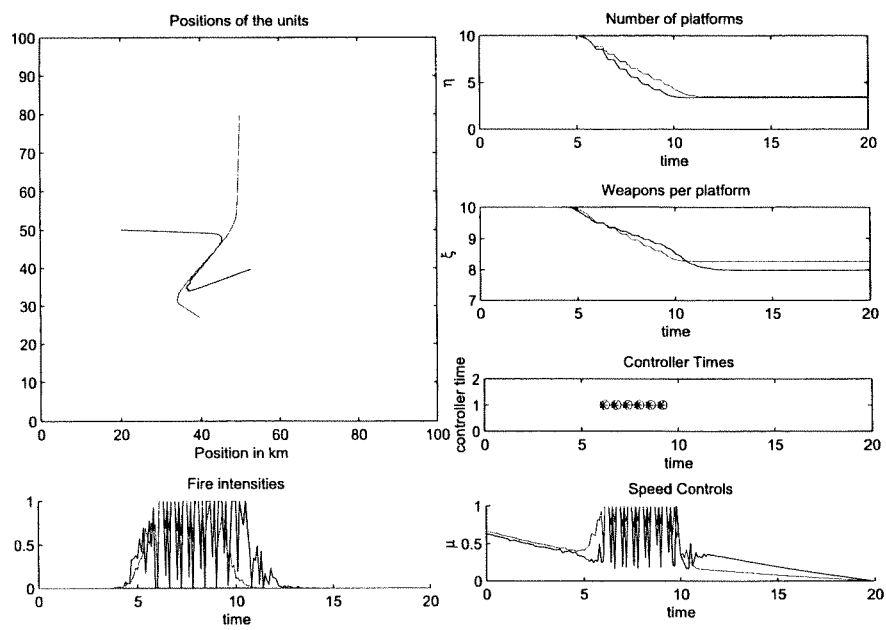


Figure 3.14: A Sample Path (Noise Amplitude 130%)

3.6 Conclusions and Recommendations

Our main findings are that while average values look good, individual sample paths might be quite surprising. One can conclude that the CPC is sensitive to observation noise. The first step to remedy the noise problem is to implement proper filters in the controllers. After this new experiments must be run to test the robustness and the effects of the filters on the observations.

Chapter 4

Experiment 4: Controller Performance under Parameter Variations

4.1 Executive Summary

The purpose is to test how the Controller-Plant-Controller setup (CPC) will react to parameter mismatches between the battlefield and the sides as well as to parameter mismatches between the sides. Assuming that both sides have chosen the game theory as the intelligence behind their controllers, systematic tests have been performed to investigate its sensitivity, i.e., how strongly the proposed game-theoretic controller reacts to changes in the parameters. The important conclusion to draw from these experiments is that even a single parameter can have important effects on the outcome of the battle. It is therefore very important to be able to estimate the enemy parameters in order to succeed in the battle simulation.

4.2 Purpose of the Experiment

The purpose is to test how the Controller-Plant-Controller setup (CPC) will react to parameter mismatches between the battlefield and the sides as well as to parameter mismatches between the sides.

4.3 Hypothesis to Prove or Disprove

The *current* differential game technology is affected by parameter mismatches. The purpose of this experiment is to investigate its sensitivity, i.e., how strongly the proposed game-theoretic controller reacts to changes in the parameters.

4.4 Experimental Setup

The experimental setup is the same as for Experiment 3 and is described in Chapter 3.

4.5 Experimental Results

The purpose of this set of experiments is to test how the Controller-Plant-Controller setup will react to the parameter mismatches between the battlefield and the sides as well the parameter mismatches

Table 4.1: Different Experimental Setup for Weight Mismatches, Weights on Final Number of Red Platforms

Red's weight	Blue has	Blue has	Blue has
20	10	20	40
40	20	40	60
60	40	60	80

between the sides. Assuming that both sides have chosen the game theory as the intelligence behind their controllers, the investigation starts by looking at weight mismatches between the sides.

4.5.1 Weight Mismatches

If both the Red and Blue sides have chosen to use the game technology and their weights in their cost functions would be the same, it would not make any sense to run the simulations using the CPC. In that case there would be a single game and no need for the CPC setup. The difference between the strategies of the sides is implemented by the difference in the internal weights of the controllers. For the following experiments the weight differences are implemented as the final number of platforms, exactly in the same manner as in the case of noisy experiments. There is a constant mismatch between both parties in all the experiments, Blue has 0.2 assigned to its final numbers of platforms while Red assign 0.1 to the Blue's final number of platforms. The weights that are changed on the Blue side, however, are the ones on the final number of platforms of Red as summarized in Table 4.1.

4.5.2 Experiments with Weight Mismatches

In each experiment summarized in Table 4.1 Blue puts its weights below, at the same level, and above Red's weights. This resulted in nine experiments. In Figures 4.1– 4.3 it is clear that Blue chases for a longer period of time the Red side when it has higher weights assigned. It is clear from these experiments that strategies can be assigned by using the weights in the internal games of the controllers.

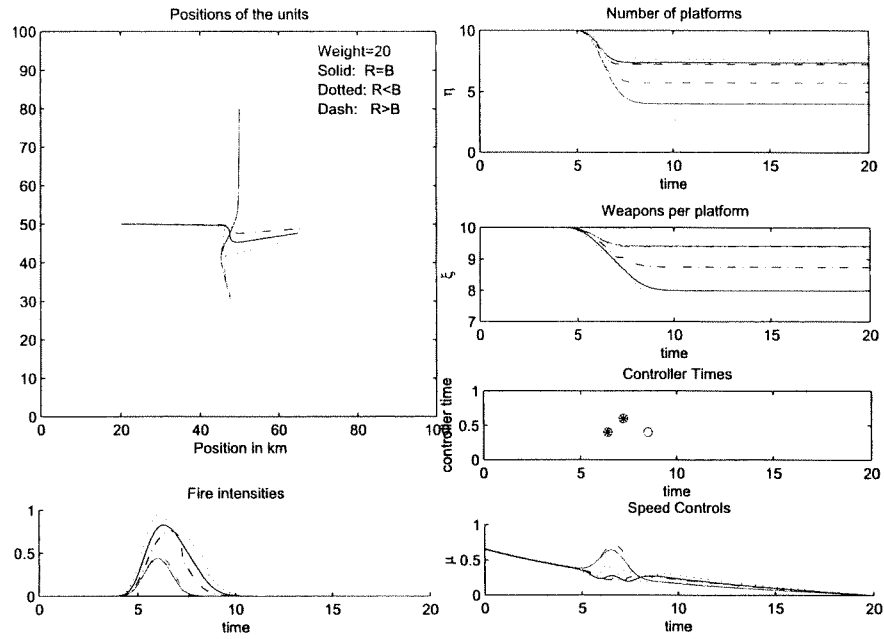


Figure 4.1: Case 1: Red's Weight is 20

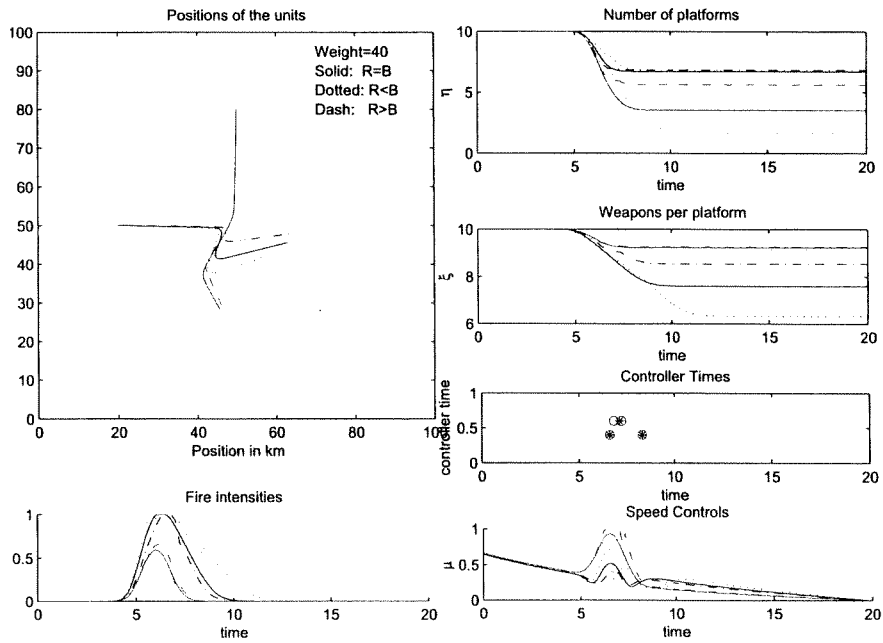


Figure 4.2: Case 2: Red's Weight is 40

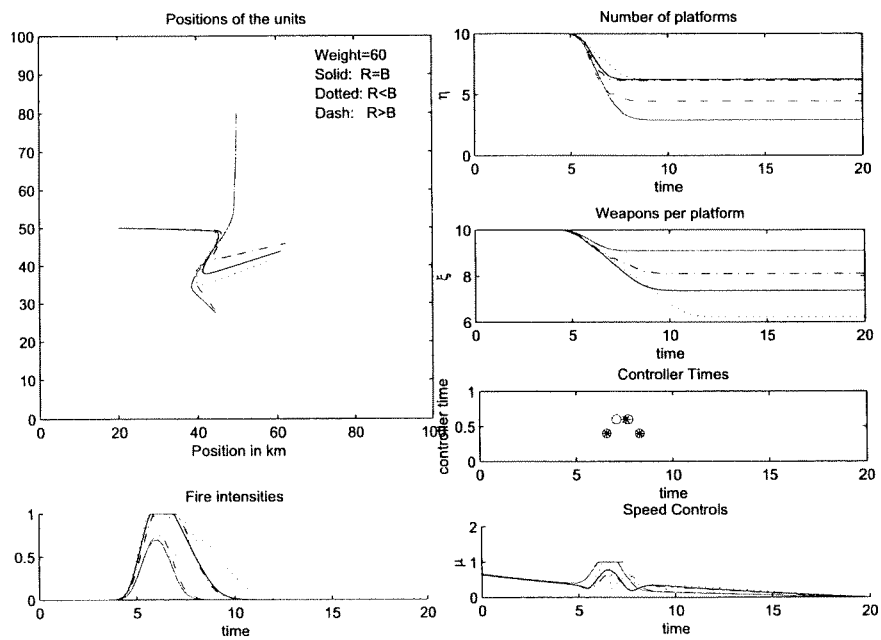


Figure 4.3: Case 3: Red's Weight is 60

Table 4.2. Parameter Mismatches for the Experiments

Case 1	Plant	Blue Side	Red Side
pkill Blue	0.3	0.3	0.1- 0.8
pkill Red	0.3	0.1- 0.8	0.3
Case 2			
pkill Blue	0.8	0.8	0.1- 0.8
pkill Red	0.8	0.1- 0.8	0.8

Differential Game Weights for Parametric Mismatch

The running weights for these experiments are the same as in the noisy experiments. The final cost matrices are given as:

$$QB_f = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1/5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 40 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$QR_f = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1/10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Exactly as it is done to test the Experiment 3 the weights are only on the final number of the platforms for both sides.

Experimental Results for Parametric Mismatch

The experiments can be categorized under two major cases:

- The battlefield dictates low probability of kill
- The battlefield has a high probability of kill

In both cases it is assumed that each side has perfect information about their own probability of kill, but does not exactly know in their internal model the probability kill of the enemy. As summarized in Table 4.2 experimental results are obtained by varying the probability of kill of the enemy side in the internal model of both controllers. The results are shown on the following figures under the low probability of kill and high probability of kill sections.

Low Probability of Kill

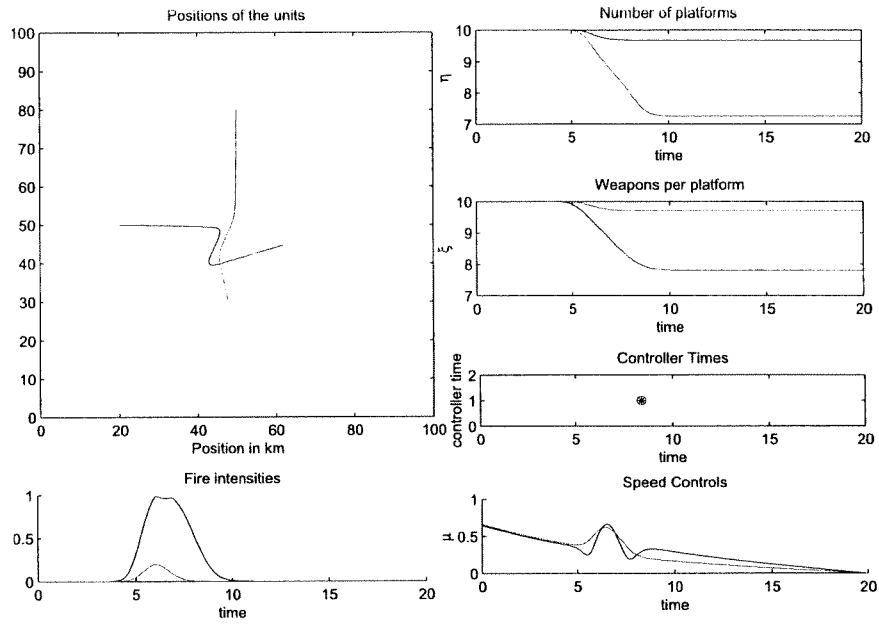


Figure 4.4: Both sides exactly know the probability of kill 0.3

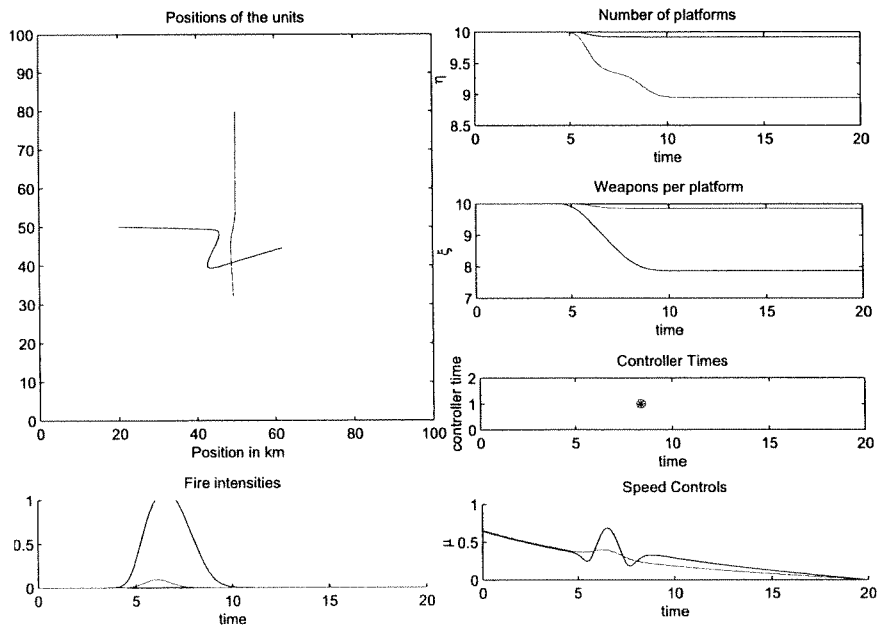


Figure 4.5: Red Underestimates Blue (Red thinks $p_{killBlue}=0.1$) $p_{kill}:0.3$

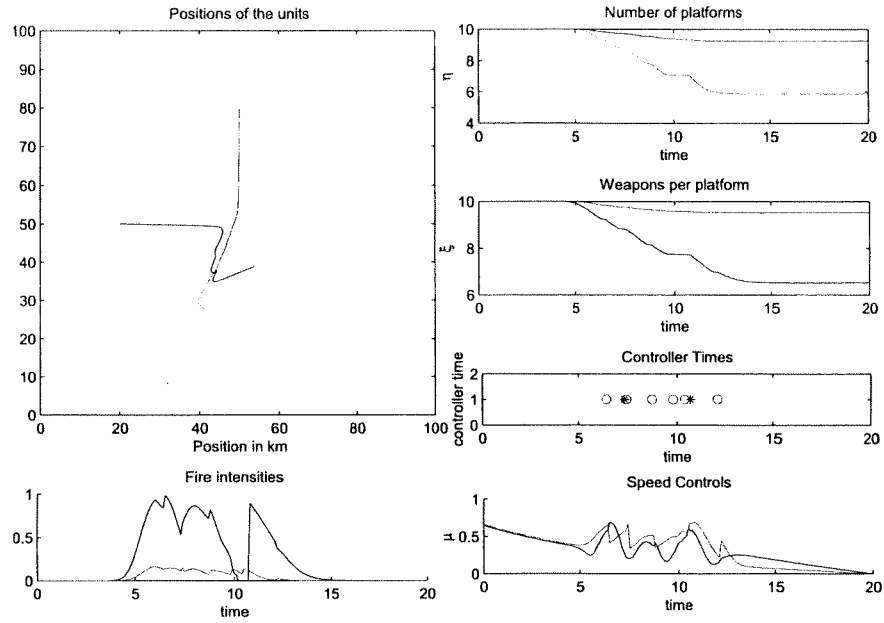


Figure 4.6: Red Overestimates Blue (Red thinks skillBlue=0.8) pkill:0.3

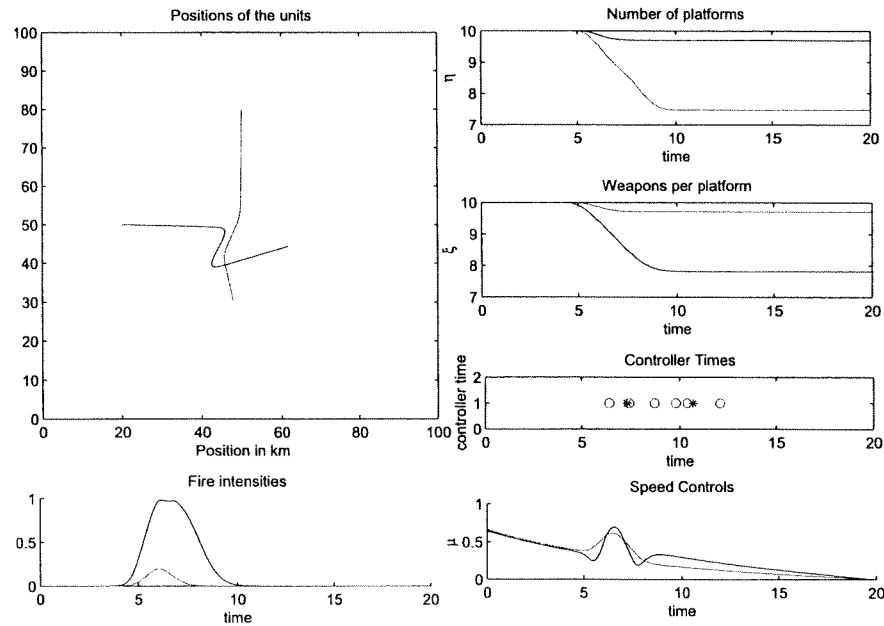


Figure 4.7: Blue Underestimates Red (Blue thinks pkillRed=0.1) pkill:0.3

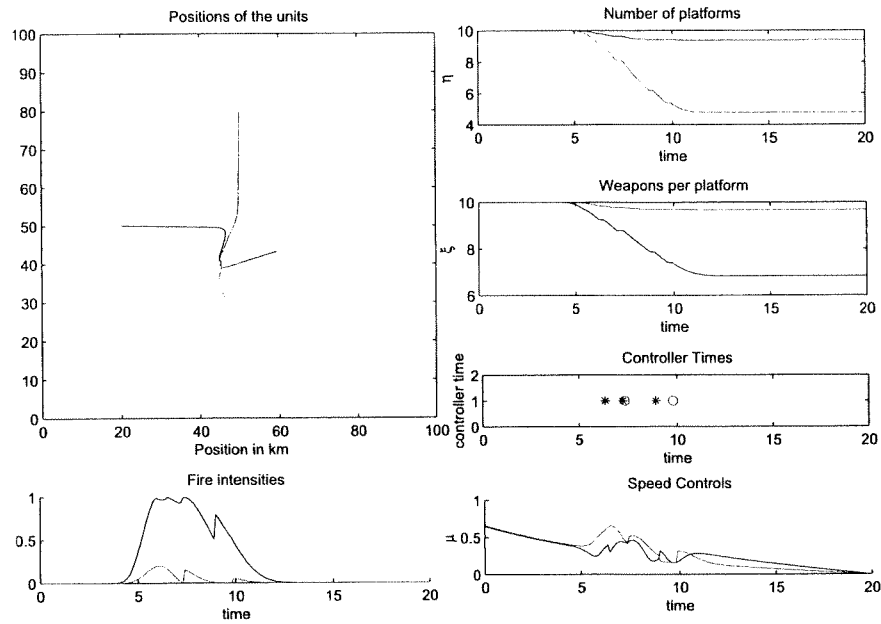


Figure 4.8: Blue Overestimates Red (Blue thinks $p_{killRed}=0.8$) $p_{kill}:0.3$

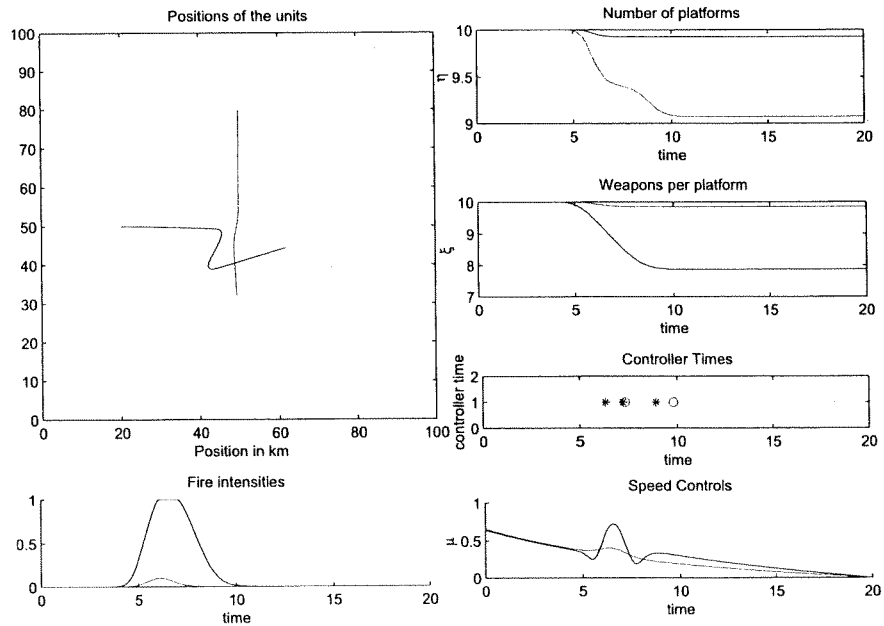


Figure 4.9: Both Sides Underestimate (Red thinks $p_{killBlue}=0.1$ similarly for Blue) $p_{kill}:0.3$

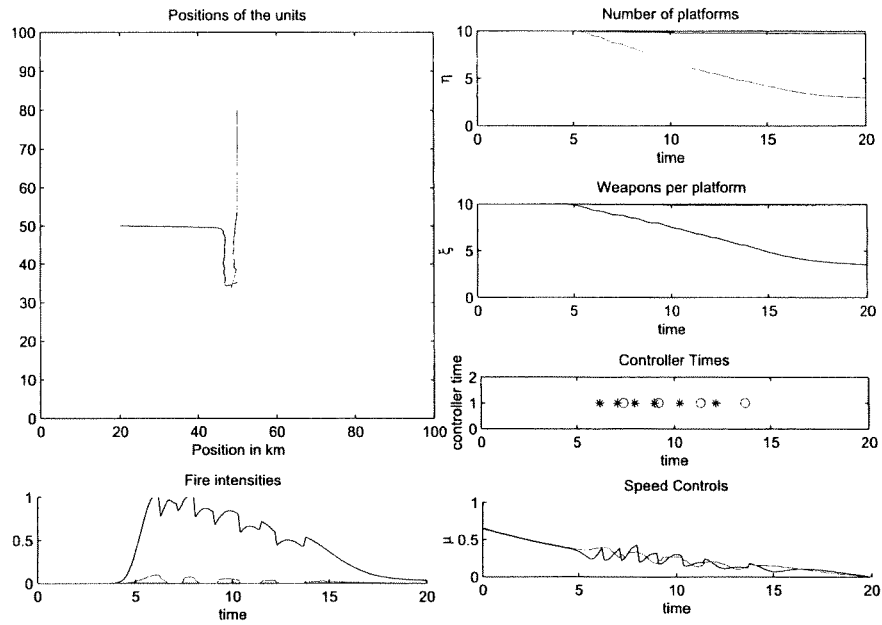


Figure 4.10: Red Underestimates Blue (Red thinks $\text{skillBlue}=0.1$) and Blue overestimates Red (Blue thinks $\text{skillRed}=0.8$) $\text{pkill}:0.3$

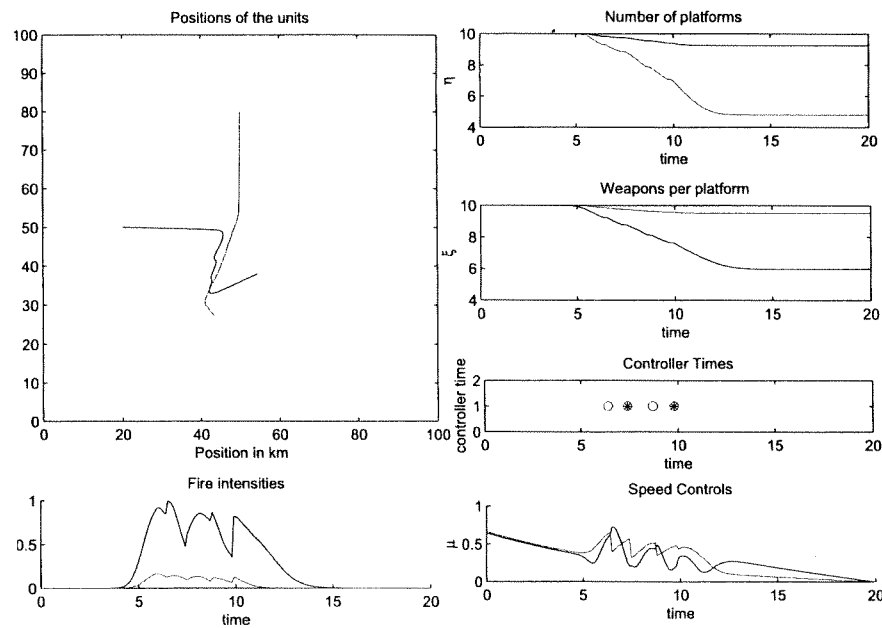


Figure 4.11: Blue Underestimates Red (Blue thinks $\text{skillRed}=0.1$) and Red overestimates Blue (Red thinks $\text{skillBlue}=0.8$) $\text{pkill}:0.3$

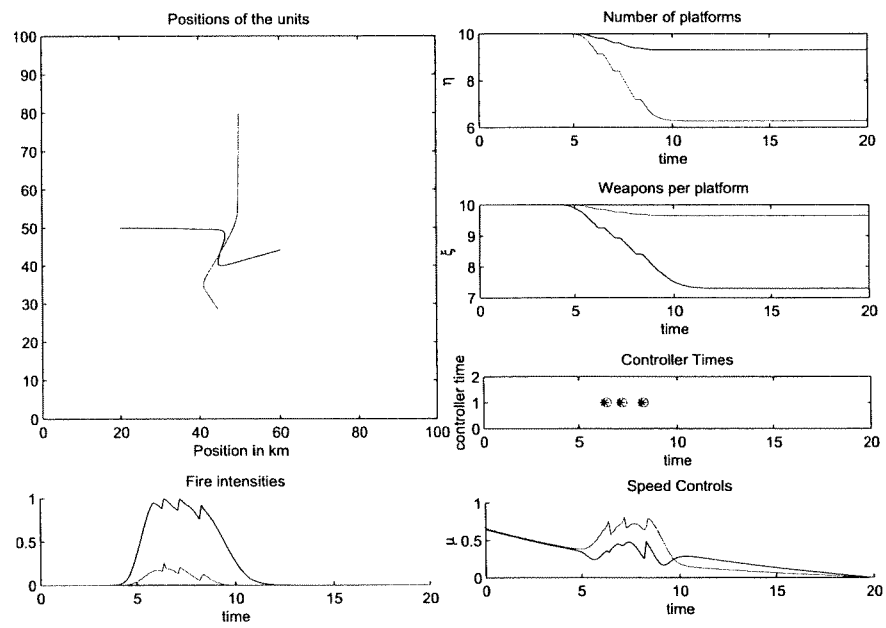


Figure 4.12: Both Sides Overestimate (Red thinks $p_{killBlue}=0.8$ similarly for Blue) $p_{kill}:0.3$

High Probability of Kill

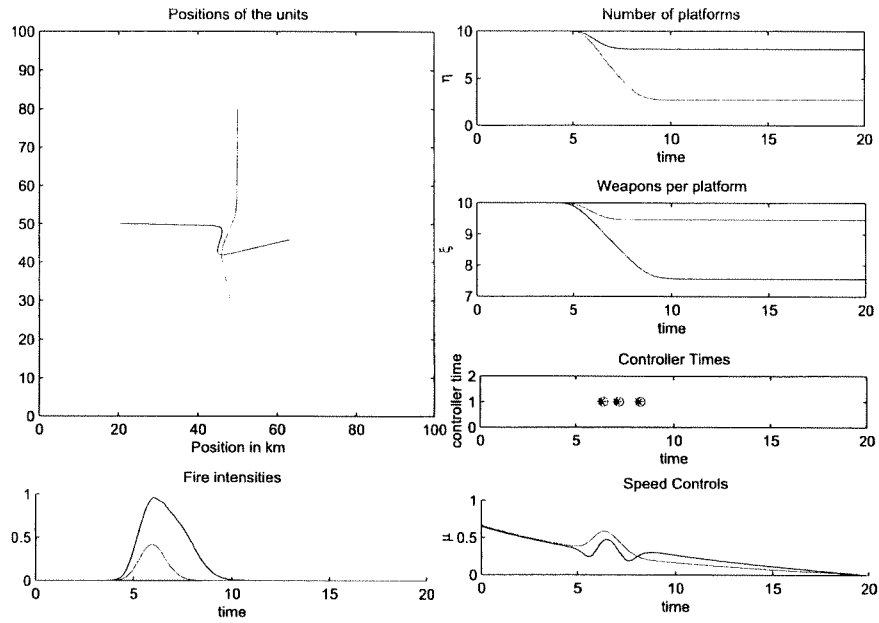


Figure 4.13: Both sides exactly know the probability of kill 0.8

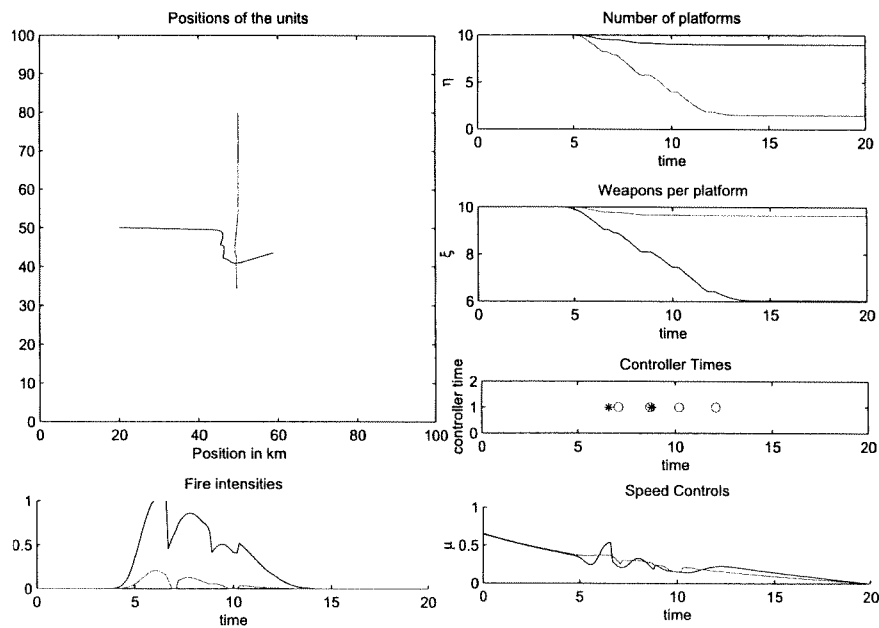


Figure 4.14: Red Underestimates Blue (Red thinks $p_{killBlue} = 0.1$) $p_{kill} = 0.8$

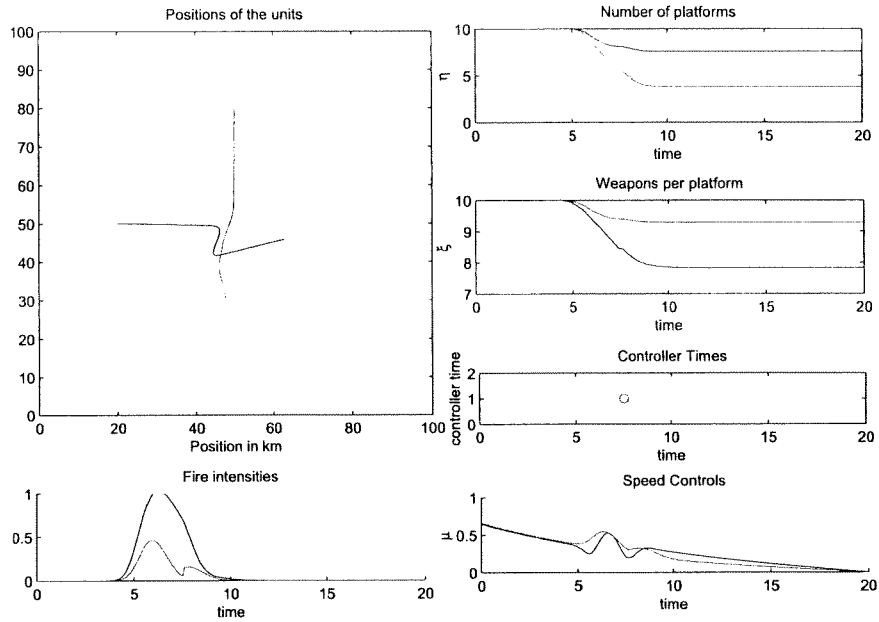


Figure 4.15: Red Underestimates Blue (Red thinks $\text{skillBlue}=0.4$) $\text{pkill}:0.8$

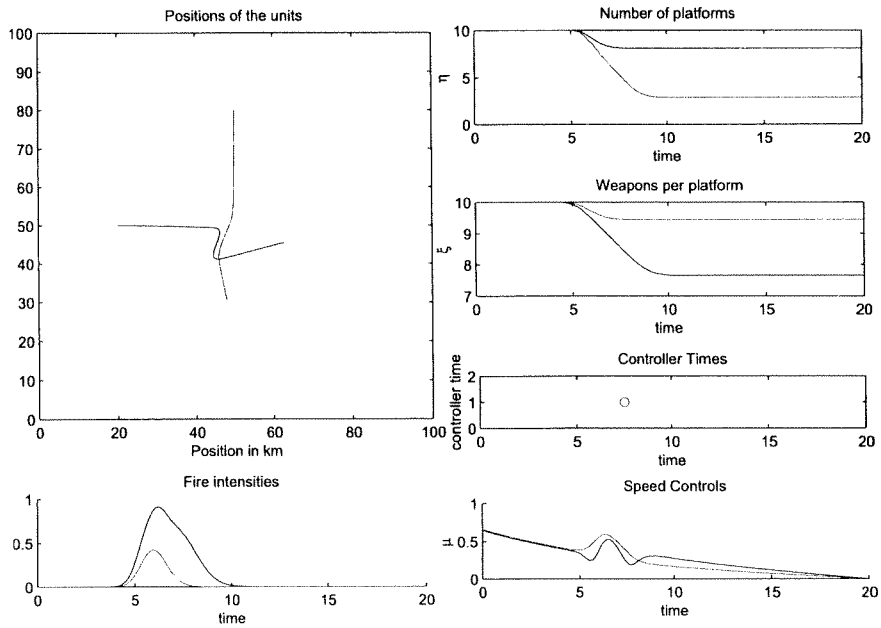


Figure 4.16: Blue Underestimates Red (Blue thinks $\text{skillRed}=0.1$) $\text{pkill}:0.8$

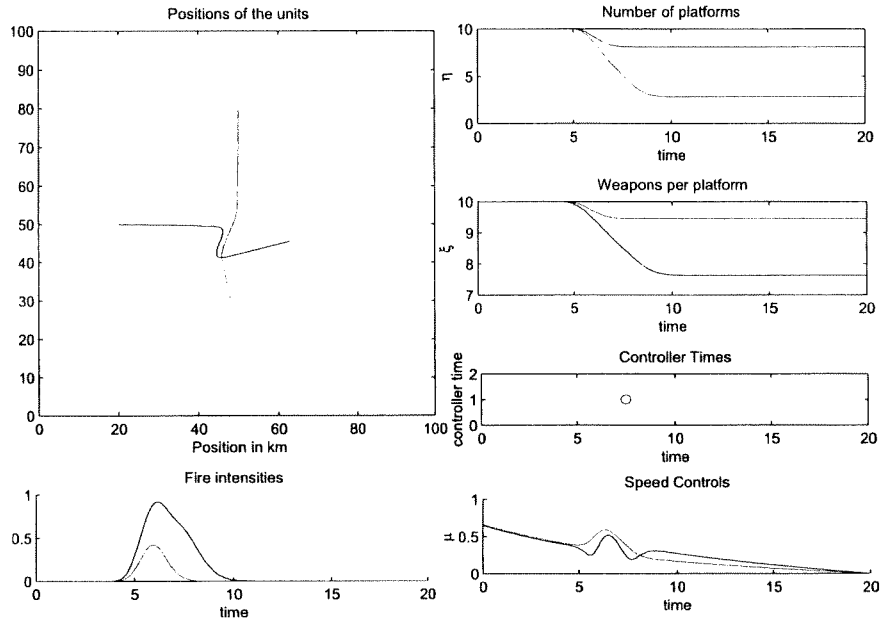


Figure 4.17: Blue Underestimates Red (Blue thinks $\text{pkillRed}=0.4$) $\text{pkill}:0.8$

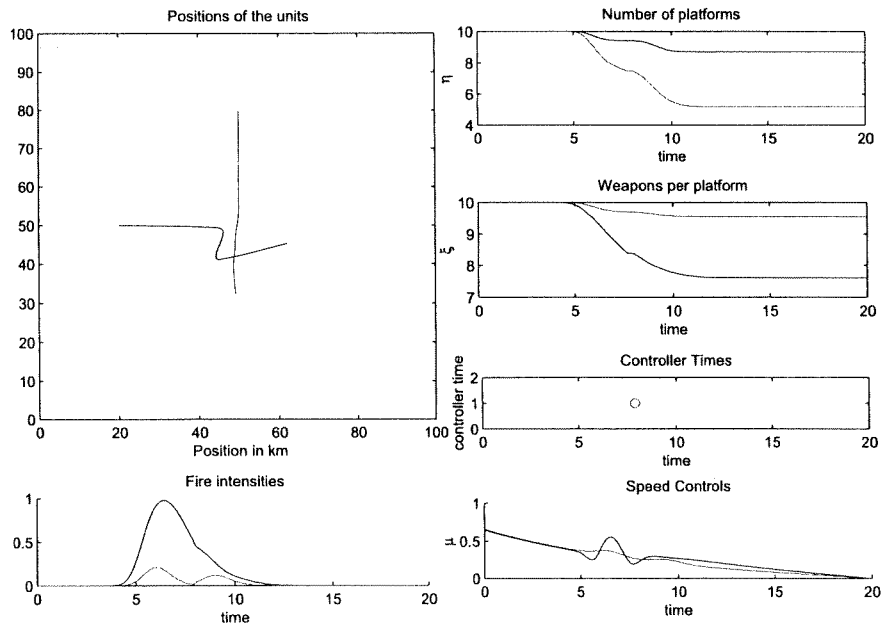


Figure 4.18: Both Sides Underestimate (Red thinks $\text{pkillBlue}=0.1$ similarly for Blue) $\text{pkill}:0.8$

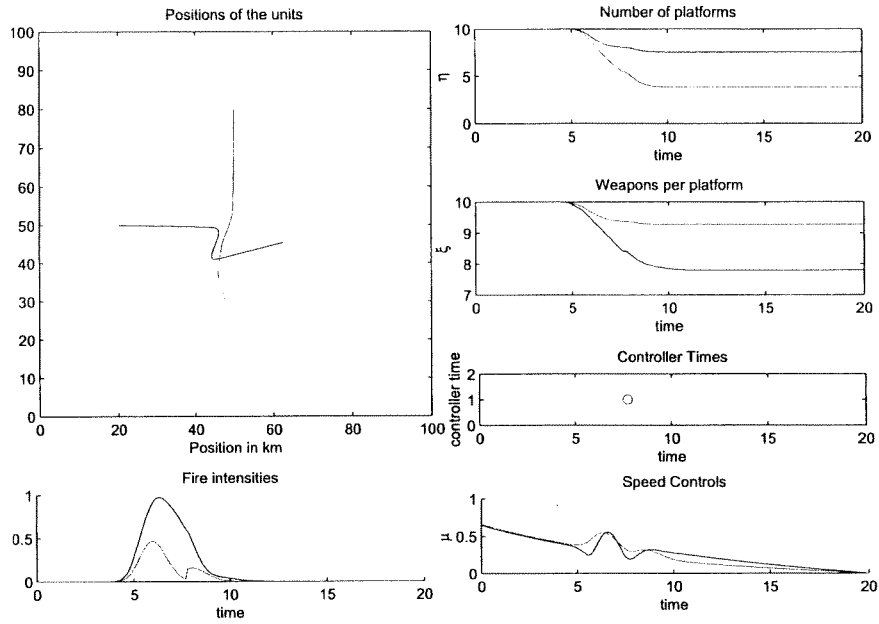


Figure 4.19: Both Sides Underestimate (Red thinks $pkill_{Blue}=0.4$ similarly for Blue) $pkill:0.8$

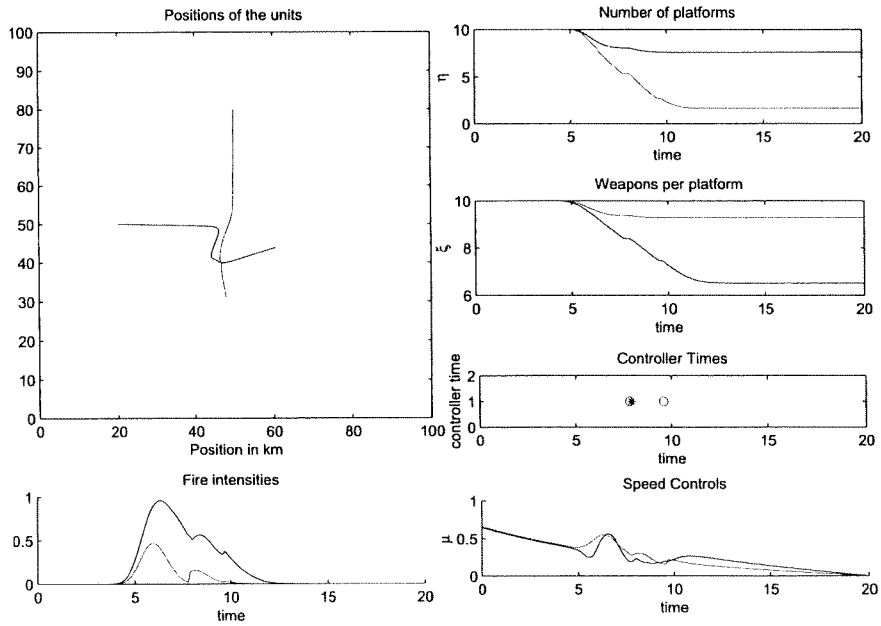


Figure 4.20: Both Sides Underestimate (Red thinks $pkill_{Blue}=0.4$ Blue thinks $pkill_{Red}=0.1$) $pkill:0.8$

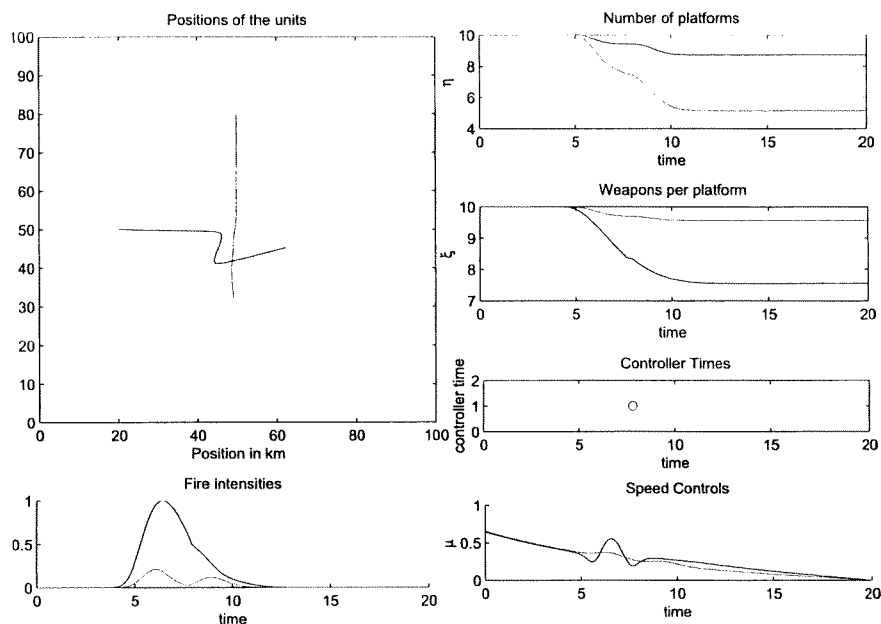


Figure 4.21: Both Sides Underestimate (Red thinks $\text{pkillBlue}=0.1$ Blue thinks $\text{pkillRed}=0.4$) $\text{pkill}=0.8$

4.6 Conclusions and Recommendations

It is clear from Figures 4.4–4.21 that the parameters have a significant impact on the differential game. Whenever the Red side underestimates as in Figure 4.5 and 4.14, it takes less evasive action and directs itself towards the target. On the other hand, whenever there is overestimation for the Red side as in Figure 4.6, more evasive action is taken. Similar remarks follow for the Blue side if the parameters are underestimated by the Blue side as in Figures 4.7, 4.16 and 4.17. Blue starts chasing red earlier than it does in the case of exact estimation or overestimation of Figures 4.4, 4.8 and 4.13. The remaining figures give an idea about the combination of different types of parameter mismatches.

The important conclusion to draw from these experiments is that even a single parameter can have important effects on the outcome of the battle. It is therefore very important to be able to estimate the enemy parameters in order to succeed in the battle simulation. As a future research path the parameter estimation from the enemy actions could be added to the difficult task of weight estimation from the enemy actions.

Chapter 5

Experiment 5: Controller Computational Complexity

5.1 Executive Summary

The *purpose* of experiment 5 is to test *Hypothesis 5*: The computational complexity of the differential game technology based controller, combined with an extended Kalman filter or a nonlinear observer, increases quadratically as a function of the number of units and linearly as a function of the mission duration.

A number of experiments have been performed to test that Hypothesis.

In the set of experiments, both the plant and internal models are the same, given by MDCM. In a first set of experiments we increase the number of units in the scenario while the mission objectives and duration are kept constant. In a second set, the mission duration is increased, while the mission objectives and the number of units are kept constant. The computation time and the number of iterations required for the computation of the control law to converge were recorded in both cases.

Our main *conclusions* are that the computational time required to reach the convergence criterion depends on many factors, such as the units categories, the number of units, initial trajectories, weights in the cost function, step size in our numerical procedure and the manner of engagements as well as initial positions and target locations. Similarly the number of iterations required to reach a convergence criterion depends on the same factors. From our experimental results, major factors which affect the computational time are the number of units and mission duration. As expected from theoretical considerations the computational time of the controller increased quadratically as a function of the number of units. We also saw that it increased linearly as a function of the mission duration, while the number of iterations remained relatively constant as a function of the number of units.

5.2 Introduction

A number of experiments have been performed to test the following **Hypothesis** : The computational complexity of the differential game technology based controller, combined with an extended Kalman filter or a nonlinear observer, increases quadratically as a function of the number of units and linearly as a function of the mission duration.

In these experiments, both the plant and internal models are the same, given by MDCM. In a first set of experiments we increase the number of units in the scenario while the mission objectives and duration are kept constant. In a second set, the mission duration is increased, while the mission objectives and the number of units are kept constant. The computation time and the number of iterations required for the computation of the control law to converge were recorded in both cases.

Table 5.1: Data For One vs. One

		Blue		Red	
Unit categories		fighter		interceptor	
Initial no. of platforms		10		10	
Initial no. of weapons		10		10	
Initial position		(20,50)		(50,82)	
Target location		(80,50)		(50,20)	
Weights in cost function		Run. cost	Ter. cost	Run. cost	Ter. cost
	Dist. to target	0.1	0	0.1	0
	Velocity command	800	0	800	0
	Firing intensities command	200	0	200	0
	No. of platforms	0	0.2	0	20

5.3 Experiment 5.1: The Number Of Units Is Increased With Fixed Mission Duration

In this set of experiments, the mission duration is kept constant at 20 minutes. In the convergence test the control change is set to 0.01 and the step size 0.5 is used.

5 experiments have been done for each of the following cases: 1 vs. 1, 2 vs. 2, 3 vs. 3, 4 vs. 4 and 5 vs. 5. In these 5 experiments for each n vs. n case ($1 \leq n \leq 5$), the units categories, initial conditions, target locations and nominal trajectories as well as the weights in the cost function may vary. The computational time and the number of iterations are recorded for each experiment.

5.3.1 One vs. One

Five different experiments were performed in this case. Here we report on one example of those five in detail. Table 5.1 summarizes the pertinent information for the two opposite forces in that specific example.

Figures 5.1 - 5.5 respectively show the initial trajectories, the control update at each iteration, the Nash solution of trajectories and the corresponding firing intensities as well as number of platforms.

The computational time required to reach the convergence criterion in this experiment is 141.935 seconds, the number of iterations is 13.

In the other four experiments, the computational time required to reach the convergence criterion was around 140 seconds and the number of iterations ranged from 13 to 16.

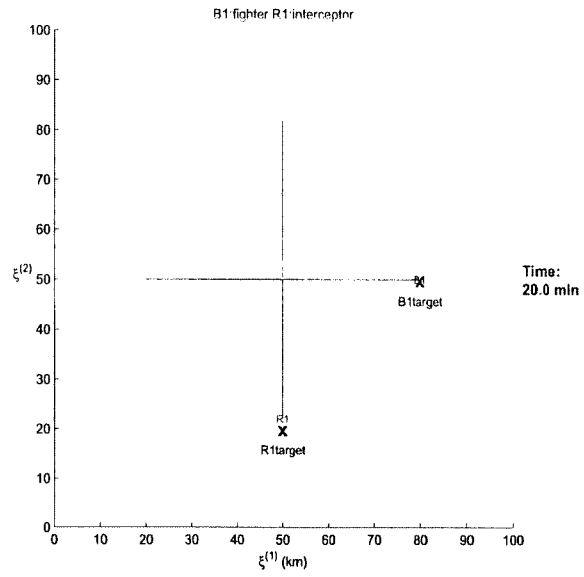


Figure 5.1: Initial Trajectories In One vs. One

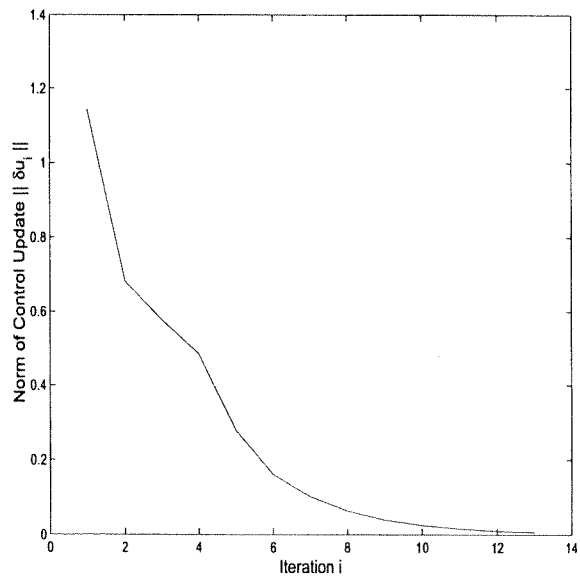


Figure 5.2: Control Update In One vs. One

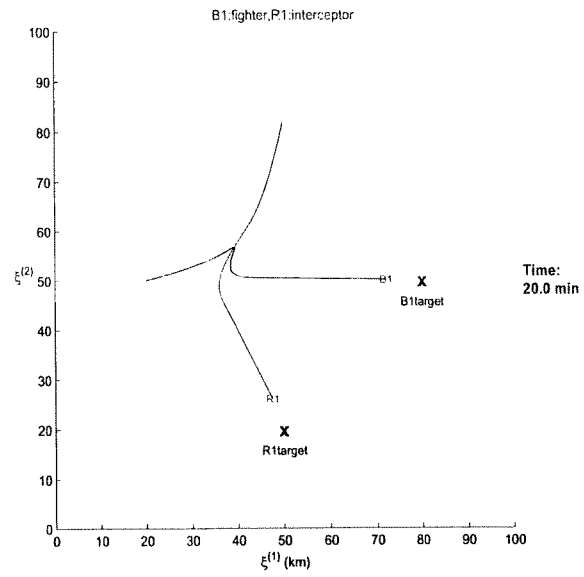


Figure 5.3: Nash Trajectories In One vs. One

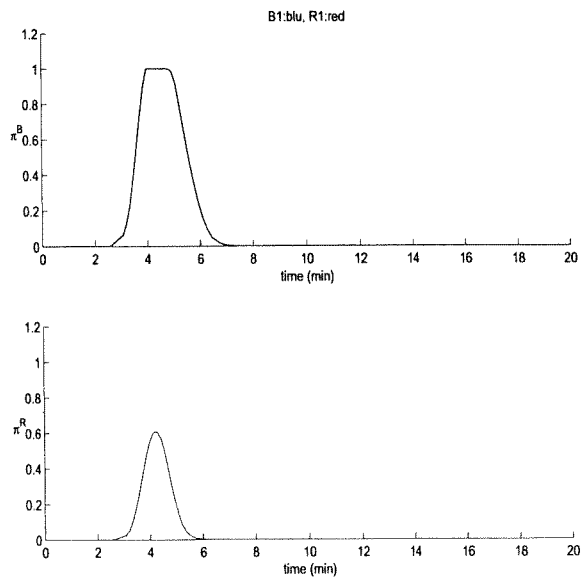


Figure 5.4: Nash Firing Intensities In One vs. One

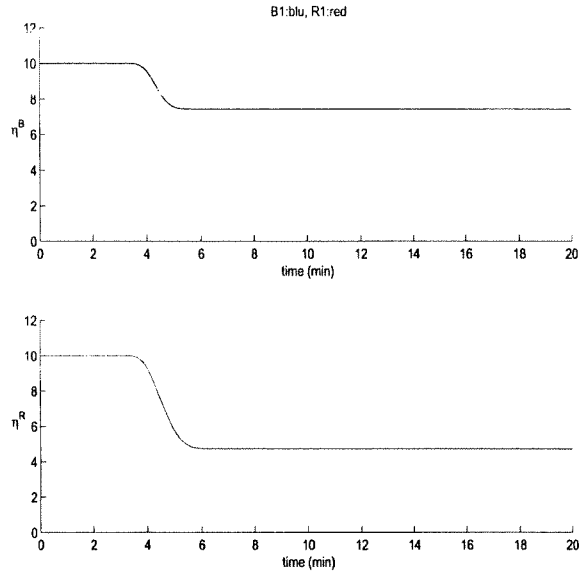


Figure 5.5: Nash Number Of Platforms In One vs. One

Table 5.2: Data For Three vs. Three

	B1	B2	B3	R1	R2	R3
Unit categories	bombers	bombers	grounds	bombers	interceptors	grounds
Initial no. of platforms	10	10	10	10	10	10
Initial no. of weapons	10	10	10	10	10	10
Initial position	(20,53)	(20,50)	(45,47)	(80,53)	(80,50)	(55,47)
Target location	(70,63)	(80,52)	(53,48)	(30,63)	(20,48)	(43,46)

5.3.2 Multi-units Case

We also report on one of five 3 vs. 3 experiments as an example for the multiple units case.

Table 5.2 summarizes the pertinent information for the two opposite forces in that specific example and Table 5.3 shows the weights in the cost function in that example. The manner of engagement in that example is: B1 and B2 are allowed to attack R1 and B3 is allowed to attack R2. R1 and R2 are allowed to attack B2 and R3 is allowed to attack B3.

Figures 5.6 - 5.10 show respectively the initial trajectories, the control update at each iteration, the Nash solution of trajectories and the corresponding firing intensities as well as number of platforms.

The computational time required to reach the convergence criterion in this experiment is 315.214 seconds, the number of iterations is 11.

In other four experiments, the computational time required to reach the convergence criterion was around 320 seconds and the number of iterations ranged from 9 to 14.

Table 5.3: Weights In Cost Function For Three vs. Three

	Running cost						Terminal cost					
	B1	B2	B3	R1	R2	R3	B1	B2	B3	R1	R2	R3
Distance to target	0.05	0.05	0.05	0.05	0.05	0.05						
Velocity command	600	600	600	600	600	600						
Firing intensities command	100	100	100	100	100	100						
No. of platforms							0.05	1	0.5	10	0.5	0.05

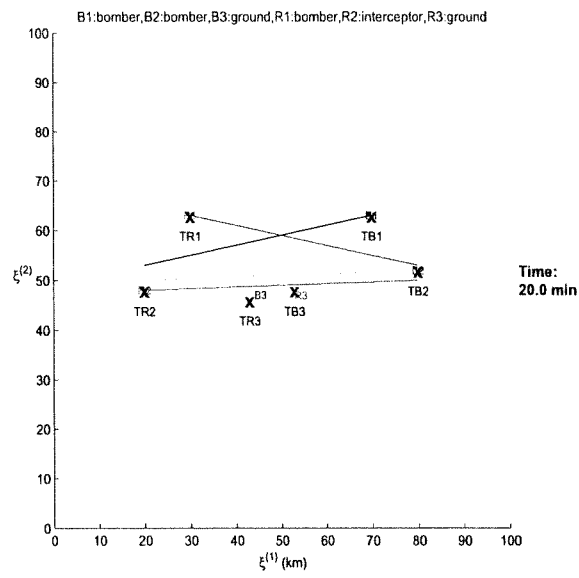


Figure 5.6: Initial Trajectories In Three vs. Three

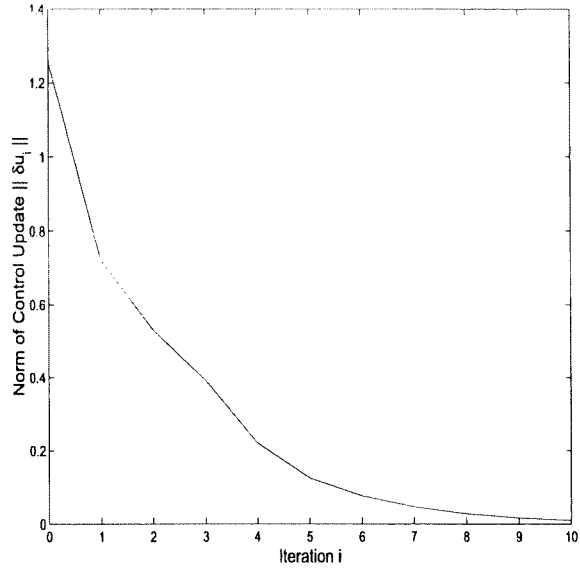


Figure 5.7: Control Update In Three vs. Three

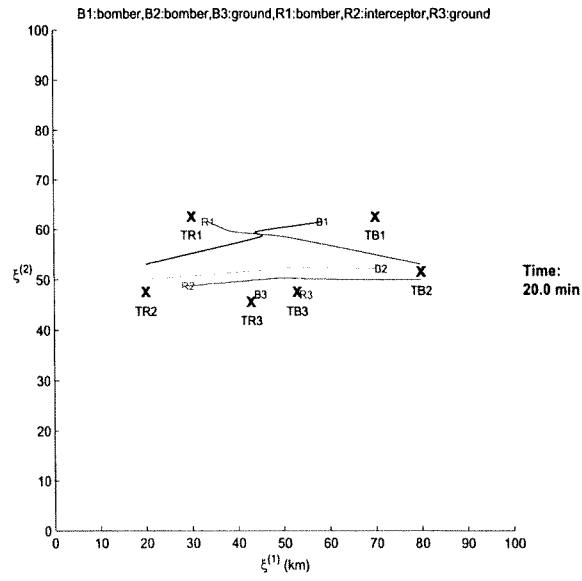


Figure 5.8: Nash Trajectories In Three vs. Three

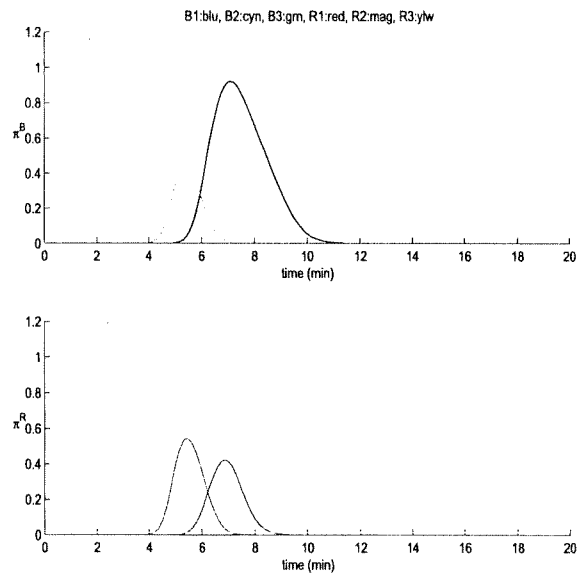


Figure 5.9: Nash Firing Intensities In Three vs. Three

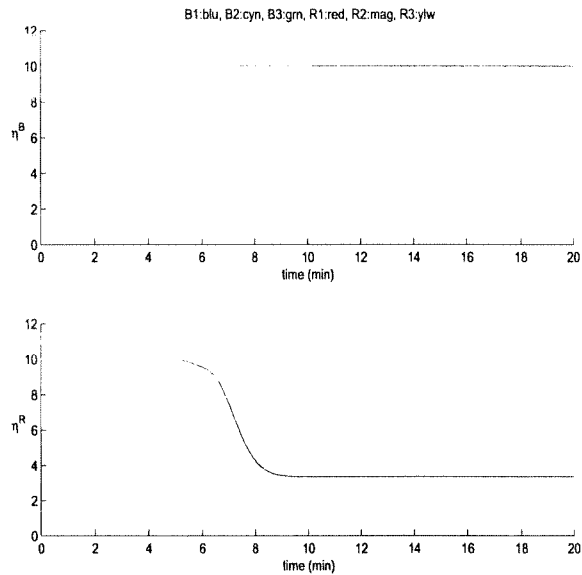


Figure 5.10: Nash Number Of Platforms In Three vs. Three

5.3.3 Multi-units Case And Computational Complexity

Similar experiments were performed in the 2 vs. 2 through 5 vs. 5 case. For a fixed number of units, 5 different experiments have been done. The computational time as well as the number of iterations were recorded in each experiment. Figures 5.11 and 5.12 show how the computational time and number of iterations required to reach a convergence criterion change as the number of units is increased respectively.

While the number of iterations remains close to constant for the 2 vs. 2 to 4 vs. 4 cases, the computational time on the average shows a quadratic growth in these cases. The computational time decreases for the 5 vs. 5 case because the 5 vs. 5 engagement is different and simpler than the one which was used for the other cases which all followed a similar pattern. Clearly the computational time will depend on the degree of interaction and the involvement of the units in battle and the case of 5 vs. 5 verifies an expected decrease in the computation time for a simpler scenario.

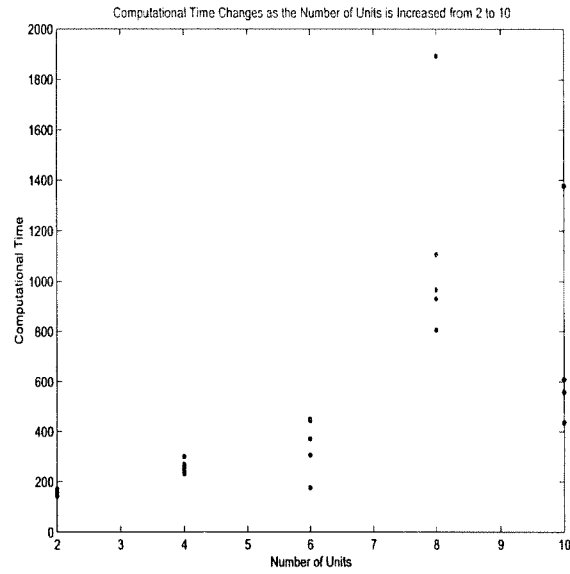


Figure 5.11: The Computational Time Changes As The Number Of Units Is Increased

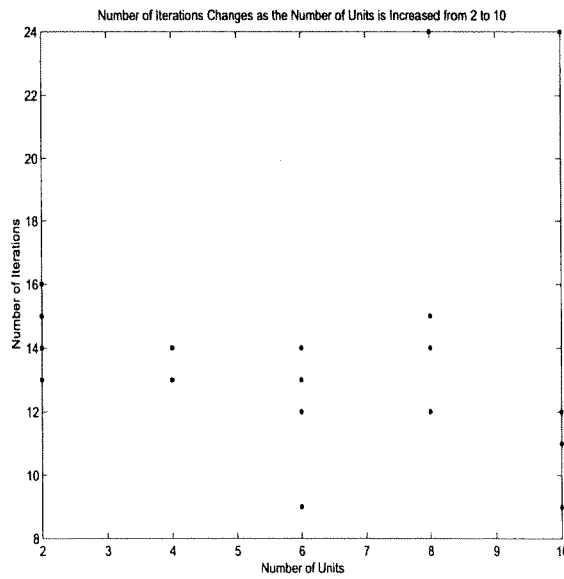


Figure 5.12: The Number Of Iterations Changes As The Number Of Units Is Increased

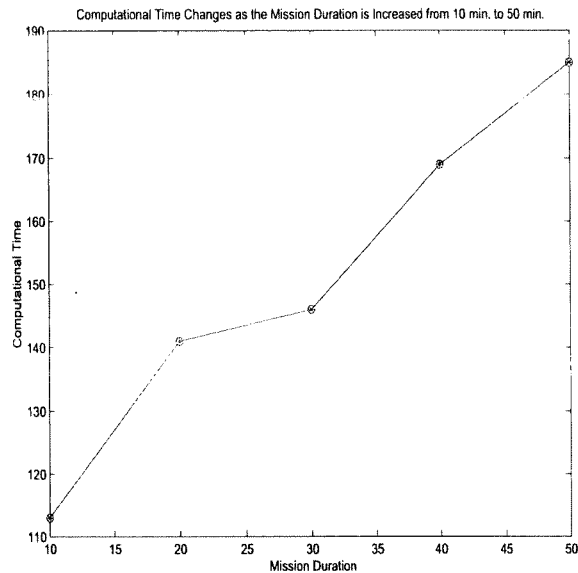


Figure 5.13: The Computational Time Changes As The Mission Duration Is Increased

5.4 Experiment 5.2: The Mission Duration Is Increased While The Number Of Units Is Kept Constant

In this set of experiments, the number of units is taken fixed, as 2. For each experiment, the mission duration varies from 10 to 50 minutes. For the convergence test the control change is set to 0.01 and step size is 0.5. Figure 5.13 records the changes in the computational time as the mission duration is increased. It roughly increases linearly with the duration time.

5.5 Conclusions

The computational time required to reach the convergence criterion depends on many factors, such as the units categories, the number of units, initial trajectories, weights in the cost function, step size in our numerical procedure and the manner of engagements as well as initial positions and target locations. Similarly the number of iterations required to reach a convergence criterion depends on the same factors. From our experimental results, a major factor which affects the computational time is the number of units and mission duration. For our experiments the computational time of the controller increased quadratically as a function of the number of units and linearly as a function of the mission duration, while the number of iterations itself remained relatively constant as a function of the number of units.

Chapter 6

Experiment 6: Controller with a Kalman Filter for Estimation

6.1 Executive Summary

In this chapter, we present an algorithm based on the Extended Kalman Filter (EKF) for state estimation when enemy inputs are unavailable. We show the overall structure of the estimation scheme through a block diagram. We present the implementation of the algorithm for the air operation theater through a flowchart. We also present the results of simulation experiments.

6.2 Purpose of the Experiment

The purpose of the experiment is to show that the *current* differential game technology, combined with an extended Kalman filter provides an effective means of countering the enemy actions under *idealized* situations with *perfect* information about enemy initial conditions and objectives, but with noisy measurements of a subset of the enemy state.

Description: Both the plant and internal models are the same, i.e., the MDCM (Mission Dynamics Continuous Model). Increasing levels of noise will be added to the state variables when constructing the observed state variables (the output variables). Some of the enemy state variables (weapons per platform first, and number of adversary platforms next) will be removed from the set of output variables thus making them unobservable. The control actions of the Blue and Red teams are generated by the proposed game theoretic algorithm.

6.3 Hypothesis to Prove or Disprove

The *current* differential game technology, combined with an extended Kalman filter provides an effective means of countering the enemy actions under *idealized* situations with *perfect* information about enemy initial conditions and objectives, but with noisy measurements of a subset of the enemy state. The algorithm based on EKF adequately estimates the unknown red state in the presence of process and observation noise.

6.4 Experiment Setup and Experiment Design

In this report, we present an algorithm based on the Extended Kalman Filter (EKF) for state estimation when enemy inputs are unavailable. We show the overall structure of the estimation scheme through a

block diagram. We present the implementation of the algorithm for the air operation theatre through a flowchart. We also present simulation results. The theoretical description of the EKF is given in [1] and [2]. The block diagram of the approach is shown in Fig. 6.1.

The inputs to the filter are the output vector y_k of the plant and the input vector u_k^B for the friendly unit, the outputs of the filter are estimates $\hat{x}_{k/k}$ and $\hat{u}_{k/k+1}^R$ of the state vector x_k and enemy input vector u_k^R .

Extended Kalman filter for estimating state x_k and enemy input u_k^R

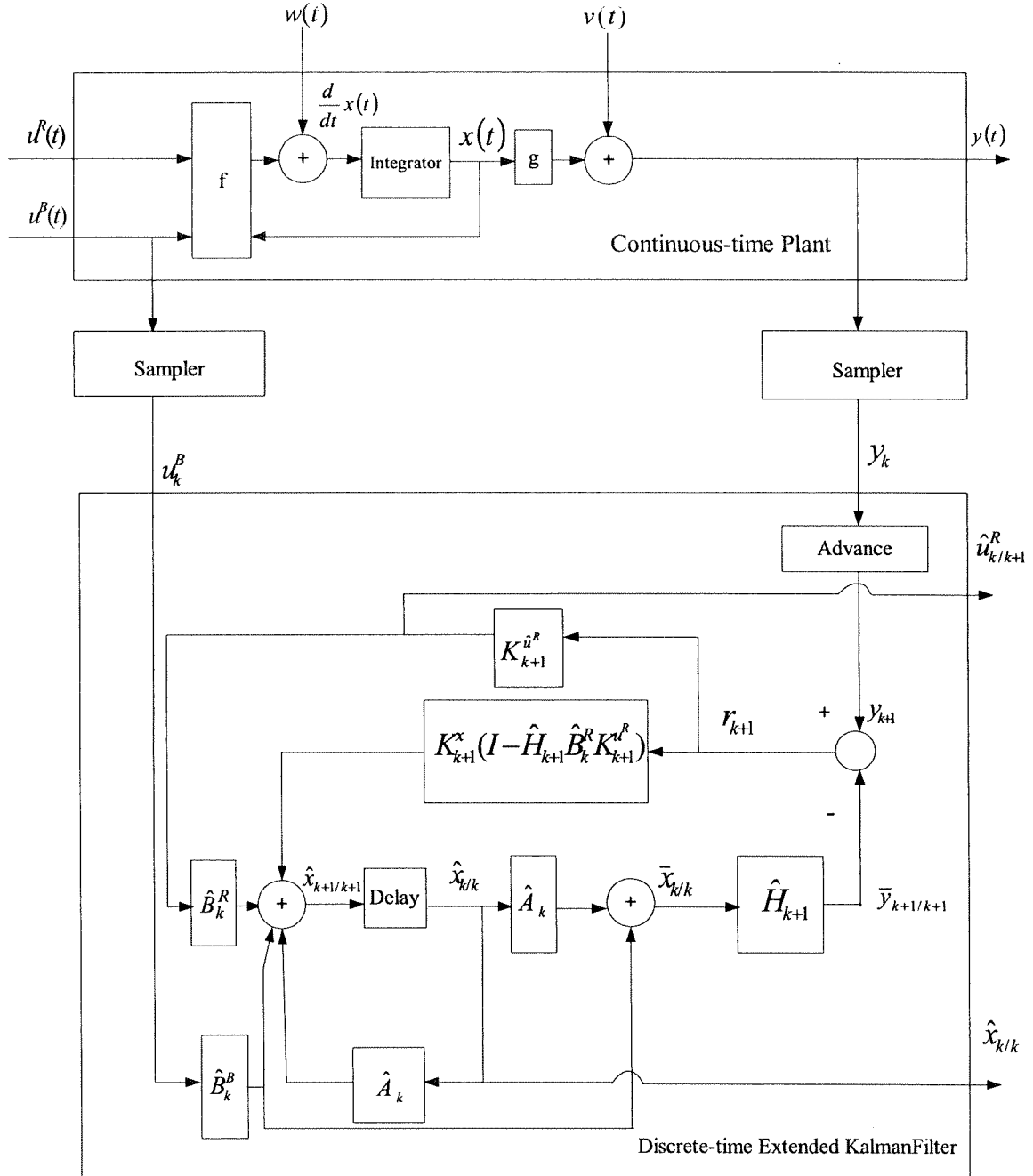
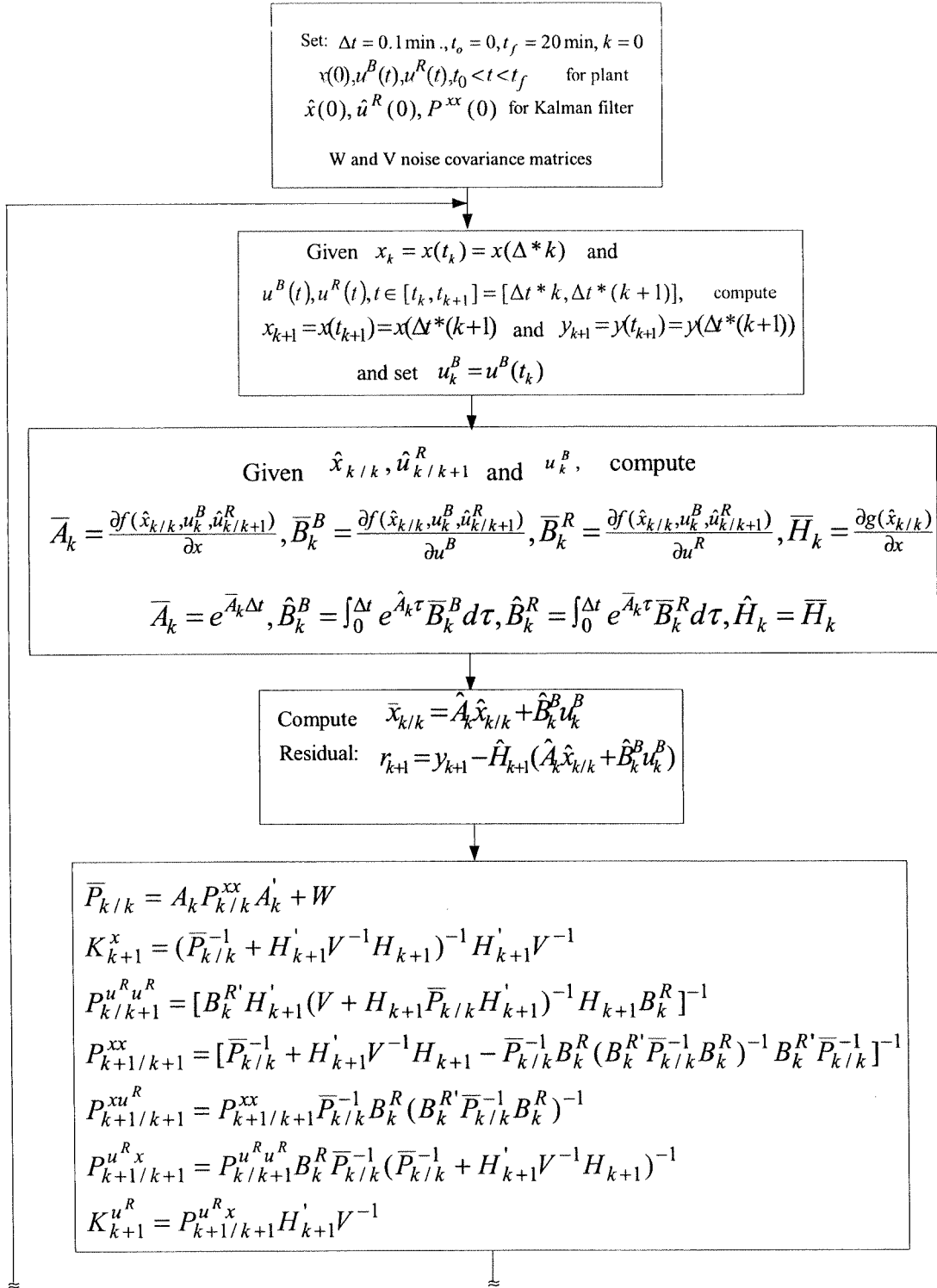


Figure 6.1: Block diagram of the Extended Kalman Filter

The flowchart of algorithm is given in Fig. 6.2.



The EKF has been combined with the Simulink game theoretic controller scheme to test the filter as

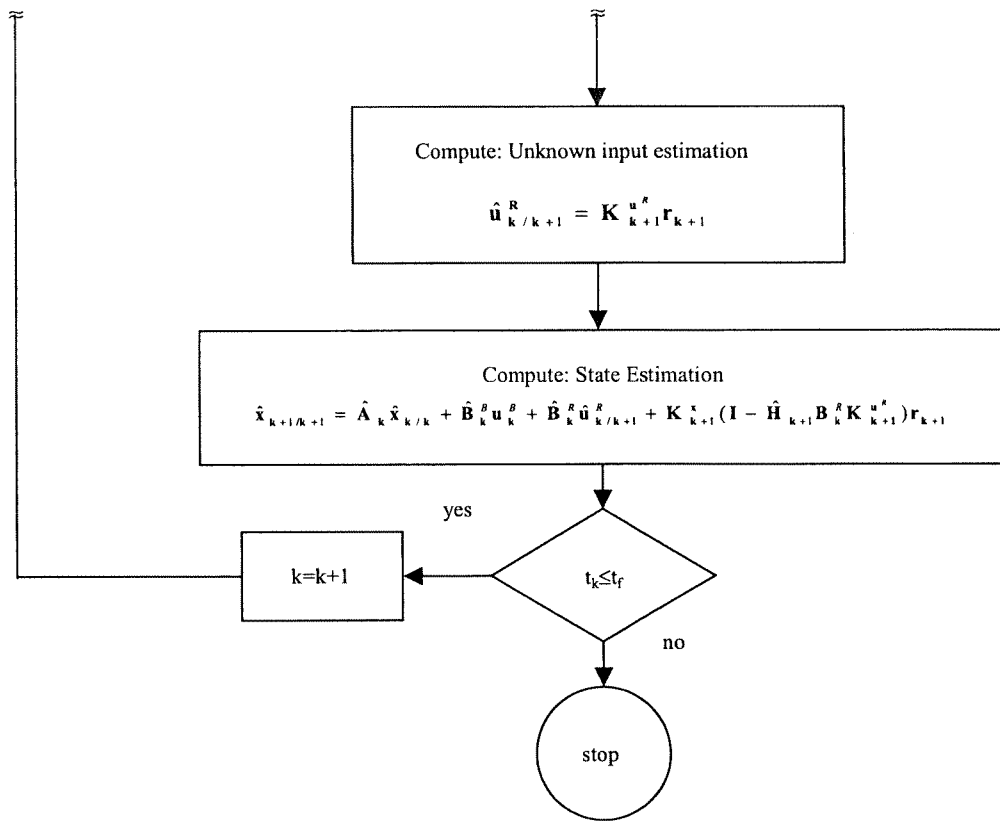


Figure 6.2: The flowchart of the estimation algorithm

a closed-loop device as in Fig. 6.3.

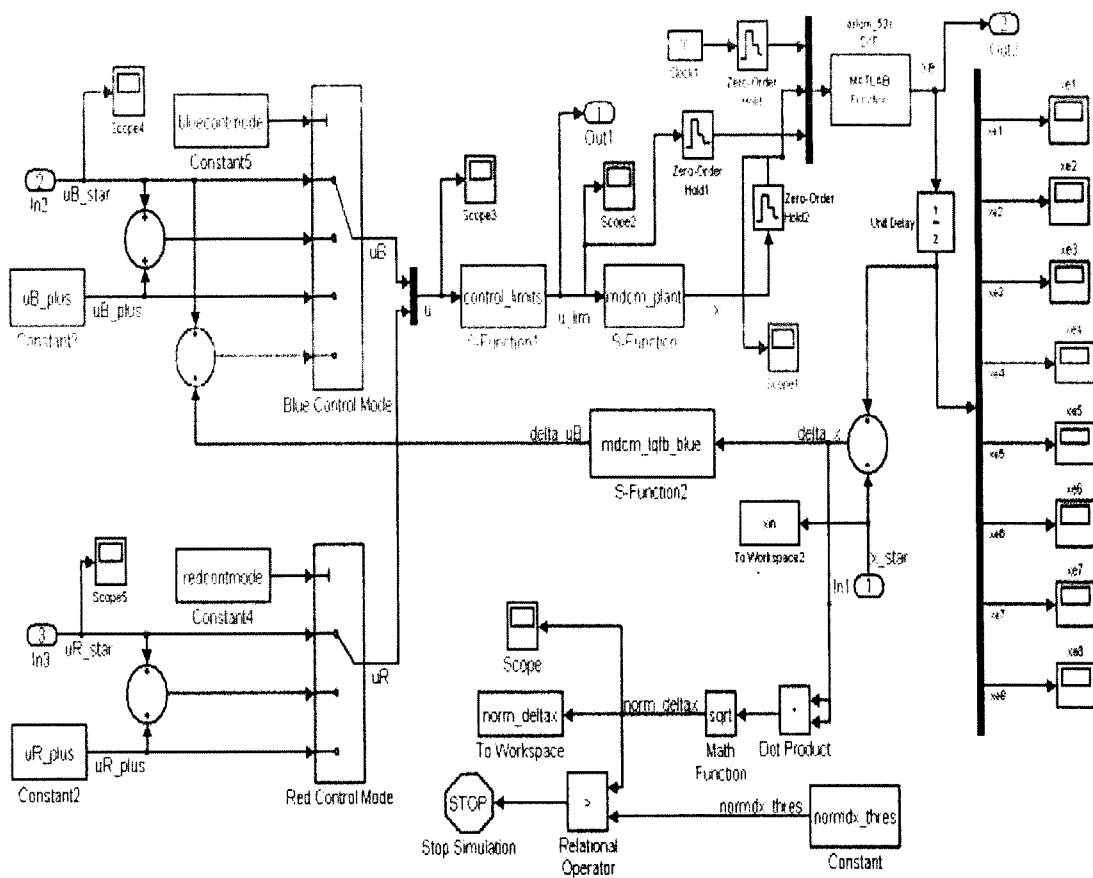


Figure 6.3: Closed-loop EKF combined with the game theoretic controller

6.5 Experiment Results and Analysis

I- NO NOISE

$v=0$

Blue states, and red states (observed (solid) and estimated (dotted)) are presented in Fig. 6.4. respectively,

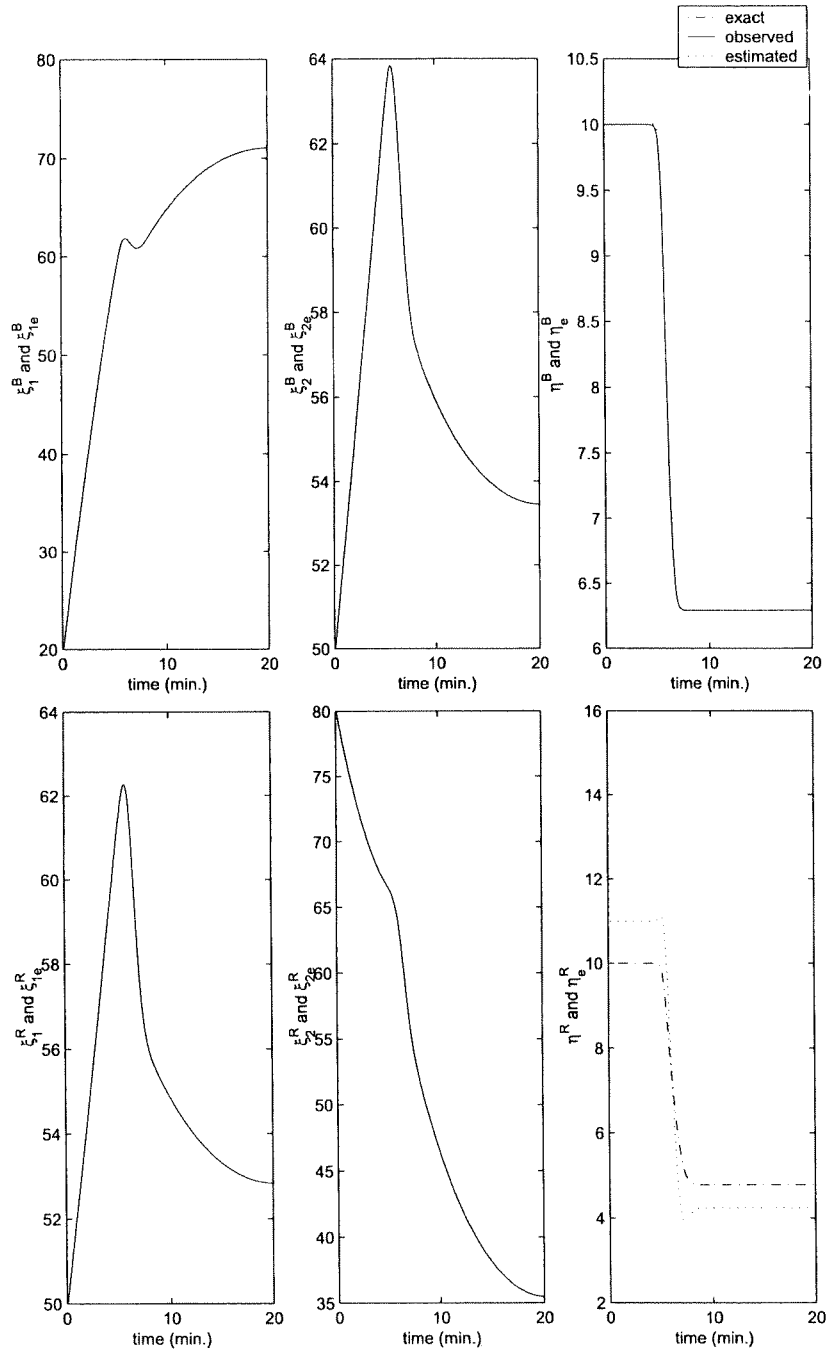


Figure 6.4: Blue states, and red states (observed (solid) and estimated (dotted)), no noise.

II- MAXIMUM SENSOR NOISE

Maximum size of gaussian sensor noises is considered. The maximum size random noise is 1% of the operating point for the blue states, and 5% of the operating point for the red states. Blue states, and red states (observed (solid) and estimated (dotted)) are presented in Fig. 6.5. respectively,

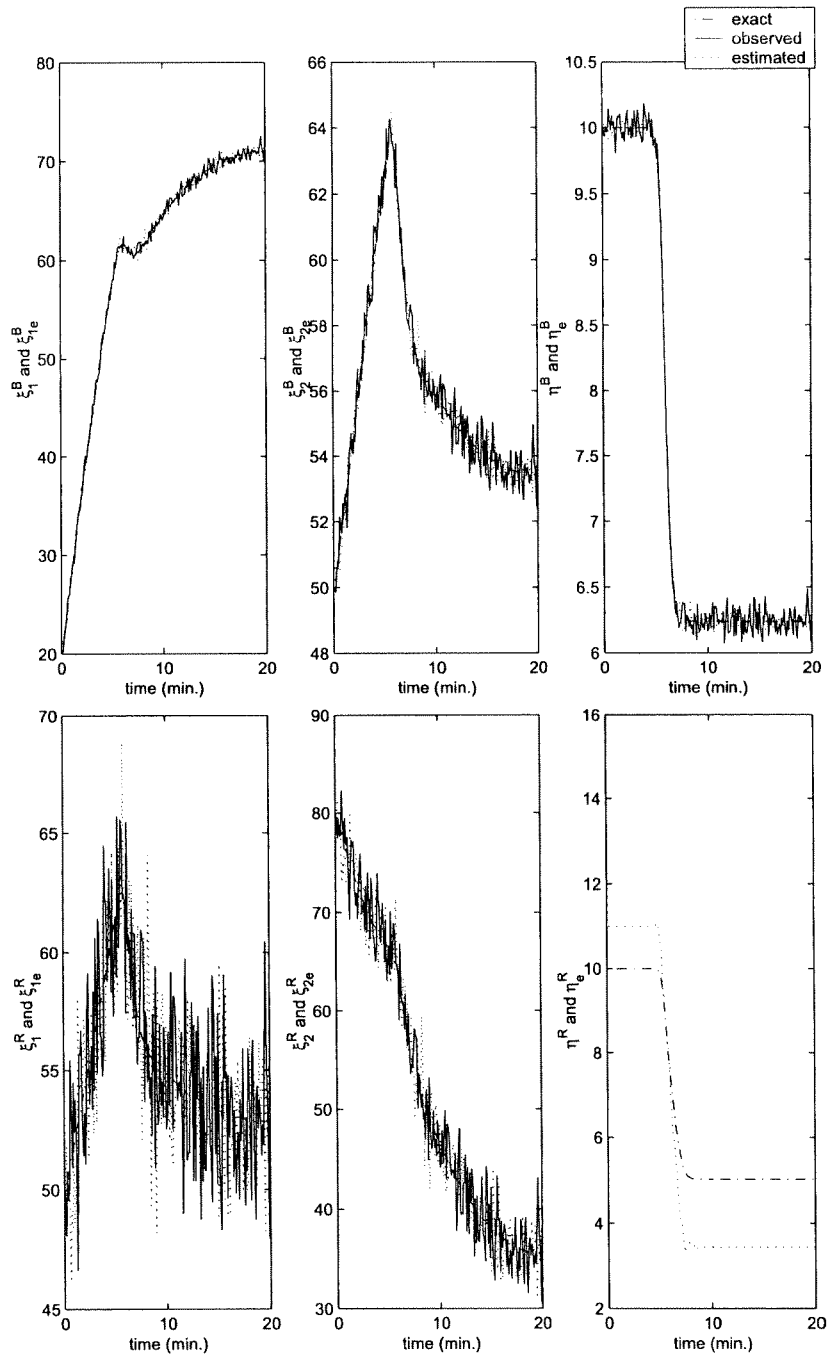


Figure 6.5: Blue states, and red states (observed (solid) and estimated (dotted)), maximum noise.

For a sample game, the results are given below. The enemy control input is given manually. The goal in the game is as follows;

The blue interceptors try to kill as many red bombers as possible and to reach the target, and The red bombers try to preserve their own platforms and to reach the target.

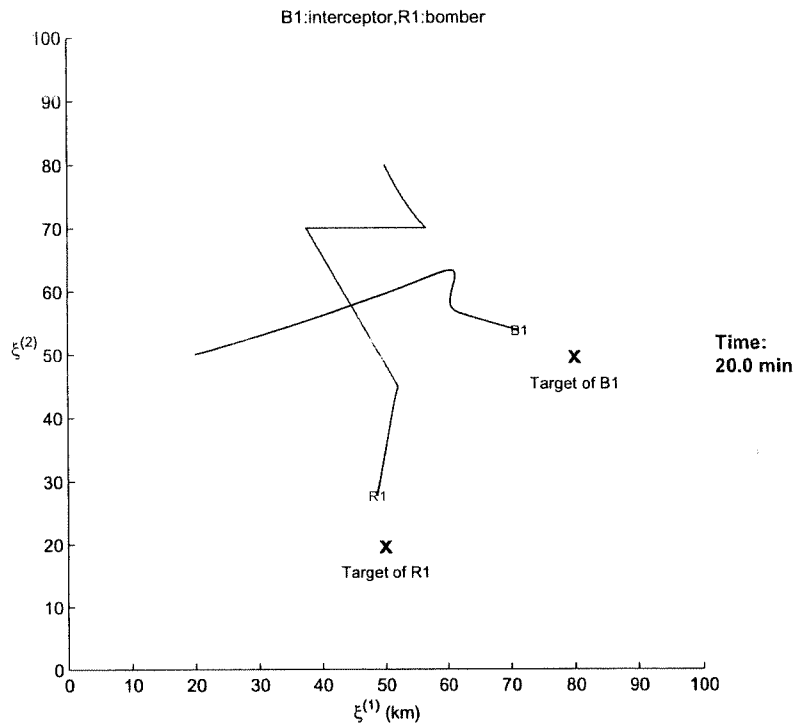


Figure 6.6: Trajectories of units

6.6 Conclusions and Recommendations

The EKF algorithm is capable of estimating the states in the presence of process noise as well as sensor noise in different size. The estimates of the enemy inputs are too noisy to be useful in themselves. Luckily, we only need the states.

The Matlab code for the EKF has also been combined with the Simulink block for the game theoretic controller.

The controller scheme uses the state estimates rather than observed states.

The scheme has successfully been implemented for both open-loop case and closed-loop case.

As η^R is not observed, a small error in the estimation may occur. The estimated enemy inputs (velocities and firing intensity) are only used for the state estimates not for feedback. Therefore, the fluctuations in the input estimations do not cause much error in the estimated states.

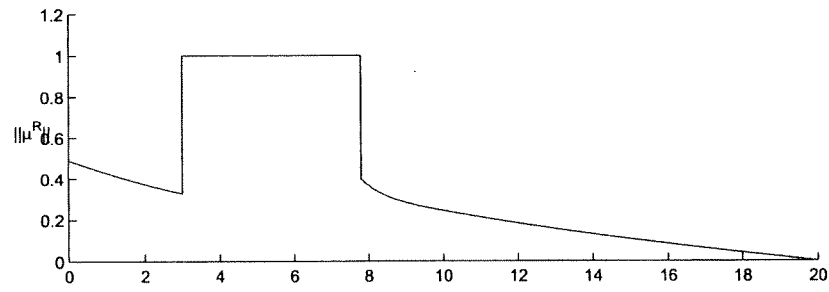
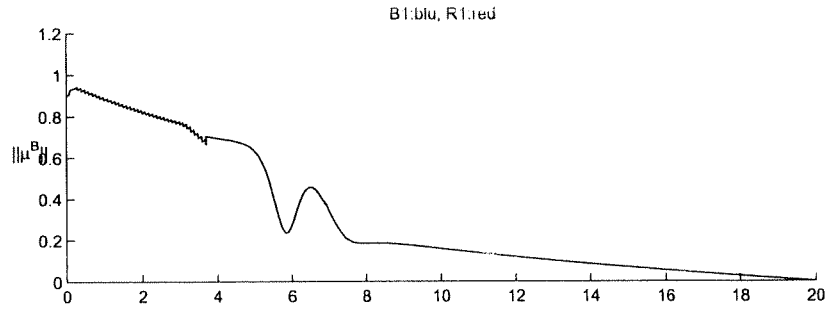


Figure 6.7: Speed controls

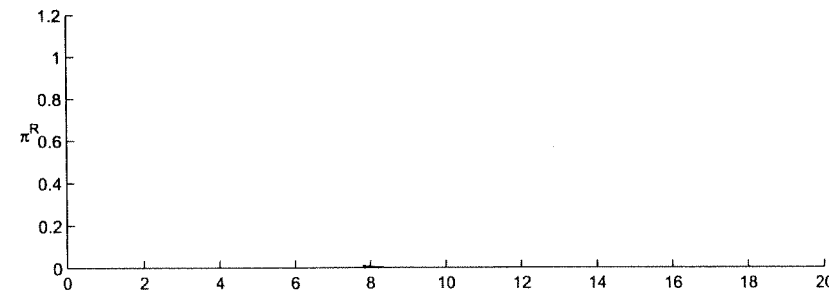
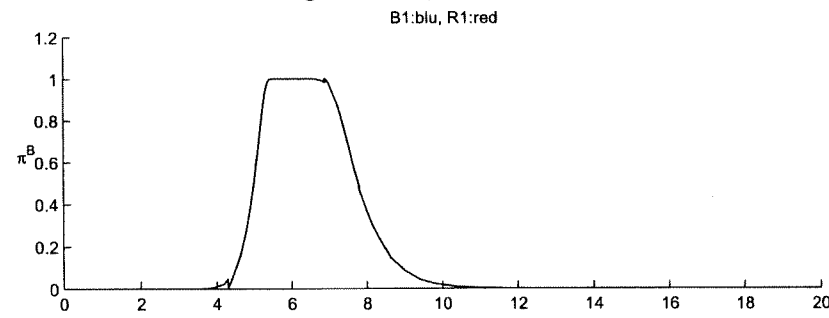


Figure 6.8: Fire intensities

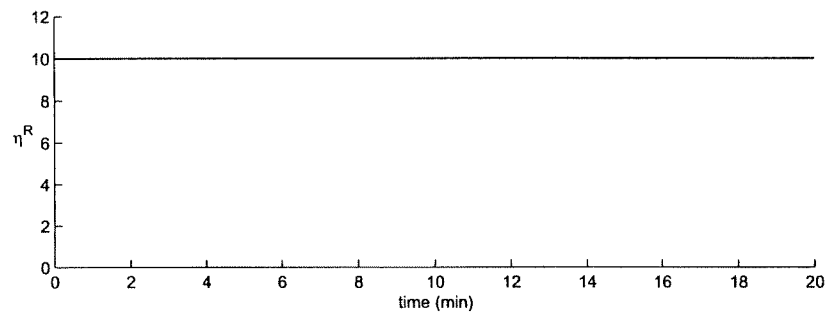
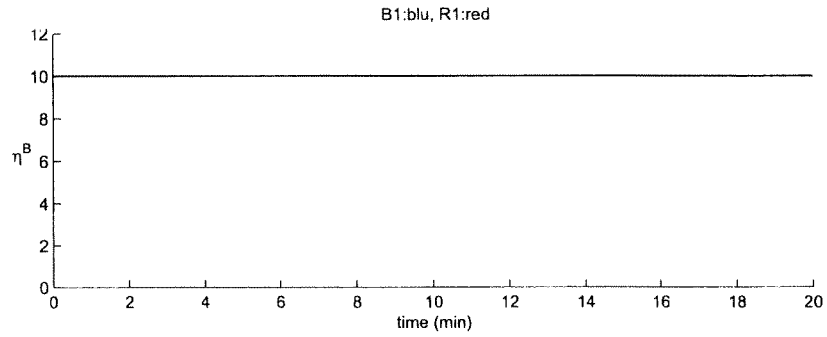


Figure 6.9: Number of platforms

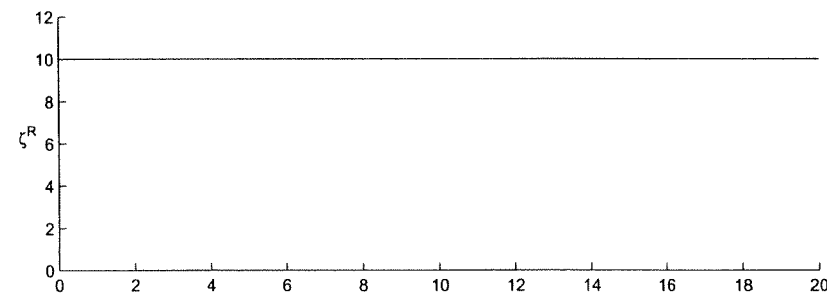
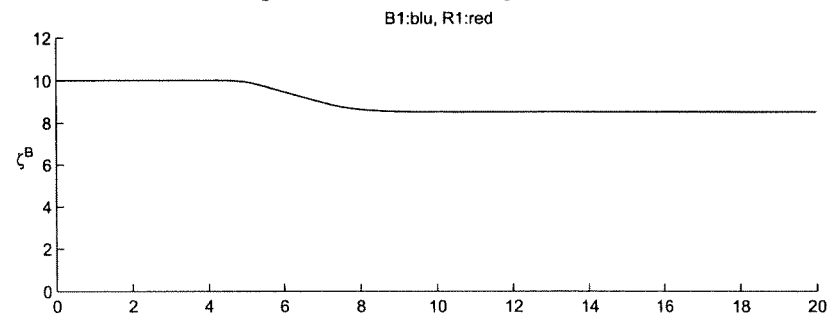


Figure 6.10: Weapons per platform

Bibliography

- [1] Malisoff M. A. and Tanikawa, A. (2000). Kalman filters to estimate states and enemy inputs, written report, Wash. U., St. Louis.
- [2] Darouach M., Zasadanski, M., Onana A.B., and Nowakowski, S. (1995). Kalman filtering with unknown inputs via optimal estimation of singular systems. *Int. J. of Systems Science*, Vol.26, No.10, pp.2015-2028.

Chapter 7

Experiment 7: Controller Applied to a More Realistic Plant

7.1 Executive Summary

The purpose of this experiment is to observe the effect of the discrepancy between internal and plant models in a closed-loop setting. The internal model is a reduced-order ODE model, called the Mission Dynamics Continuous-time Model (MDCM 3.0), and the plant model is a full order ODE model, abbreviated as EPMDM, which exactly describes the evolution of expected values in PMDM.

The hypothesis is that the current differential game technology would provide an effective means of countering the enemy actions, who may be either following the Nash solution or using some simple heuristic strategy, when noise-free state measurements are available, in spite of the mismatch between the plant and the internal models.

It is concluded that approximating the plant model with a lower order internal model does not cause a significant difference in game results, as long as the engagement terminates before one side is completely wiped-off.

7.2 Purpose of the Experiment

The purpose of this experiment is to observe the effect of the discrepancy between internal and plant models in a closed-loop setting. The internal model is a reduced-order ODE model, called the Mission Dynamics Continuous-time Model (MDCM 3.0) (see Chapter 1 and its appendix), and the plant model is a full order ODE model, abbreviated as the EPMDM, which exactly describes the evolution of expected values in the PMDM [2] (see also Chapter 1).

In the Experiment Plan [1] the hypothesis for Experiment 7 is stated as:

The current differential game technology provides an effective means of countering enemy actions under more realistic situations with perfect information about the enemy.

To understand this statement, we remark that the current controller for our system is based on an approximation of the actual model, as described in [2]. When the dynamics of the plant are the same as the internal model used to compute the Nash solution, we know that the current controller is effective. We now test if this controller is still effective when the plant dynamics are more realistic, while the controller is based on approximated dynamics.

7.3 Hypothesis to Prove or Disprove

The hypothesis is that the current differential game technology would provide an effective means of countering the enemy actions, who may be either following the Nash solution or using some simple heuristic strategy, when noise-free state measurements are available, in spite of the mismatch between the plant and the internal models.

7.4 Experiment Setup

To implement more realistic game dynamics, we use the Probabilistic Mission Dynamics Model (PMDM) for uncoordinated target selection as derived in [2]. In [2], the model is derived only for combat between two opposing units, therefore our experiments will only be for two opposing units (Blue and Red). First we summarize the PMDM.

The number of platforms depend on chance occurrences and are defined by the random variables

$$\begin{aligned} X^B(t) &\stackrel{\text{def}}{=} \text{number of platforms in the Blue unit at time } t, \\ X^R(t) &\stackrel{\text{def}}{=} \text{number of platforms in the Red unit at time } t. \end{aligned}$$

The initial values are known to be $X^B(0) = N^B$, $X^R(0) = N^R$. At any given time, the commanders (or controllers) can observe only the expected values, denoted by

$$\eta^B(t) \stackrel{\text{def}}{=} E[X^B(t)] \text{ and } \eta^R(t) \stackrel{\text{def}}{=} E[X^R(t)].$$

Then by using the standard notation

$$\Pi_{n,m}(t) \stackrel{\text{def}}{=} P\{X^B(t) = n, X^R(t) = m\}$$

for a Markov Process, the evolution of state probabilities are described by the differential equation

$$\dot{\Pi}(t) = \Pi(t)Q(t),$$

where we stack all components of $\Pi_{n,m}$ into a vector Π , and Q is called the transition rate matrix of the process. Considering just the Red unit firing on the Blue unit, the loss rate of Blue platforms is defined by

$$\lambda^B(t) \stackrel{\text{def}}{=} \rho^R P_k^R \phi(\|\xi^B - \xi^R\|) \pi^R,$$

where ρ is the acquisition rate, P_k is the probability of kill, ϕ is a function depending on the distance between the units and π is the fire intensity. Then the more realistic plant model, derived in detail in [2], is

$$\frac{d}{dt} \eta^B = -\lambda^B \eta^R + \lambda^B \sum_{m=1}^{N^R} m \Pi_{0,m}, \quad (7.1)$$

$$\frac{d}{dt} \eta^R = -\lambda^R \eta^B + \lambda^R \sum_{m=1}^{N^B} n \Pi_{n,0}. \quad (7.2)$$

We call (7.1) and (7.2) as the evolution of the expected values in the PMDM (EPMDM). The EPMDM is the plant model and is implemented into the game technology software as shown in Figure 7.1, which is the Simulink file that simulates the game dynamics. The only change in the software from the MDCM to the EPMDM is the grey box in Figure 7.1 labeled ‘‘Markov Chain Model’’. This box calculates the summation terms in (7.1) and (7.2).

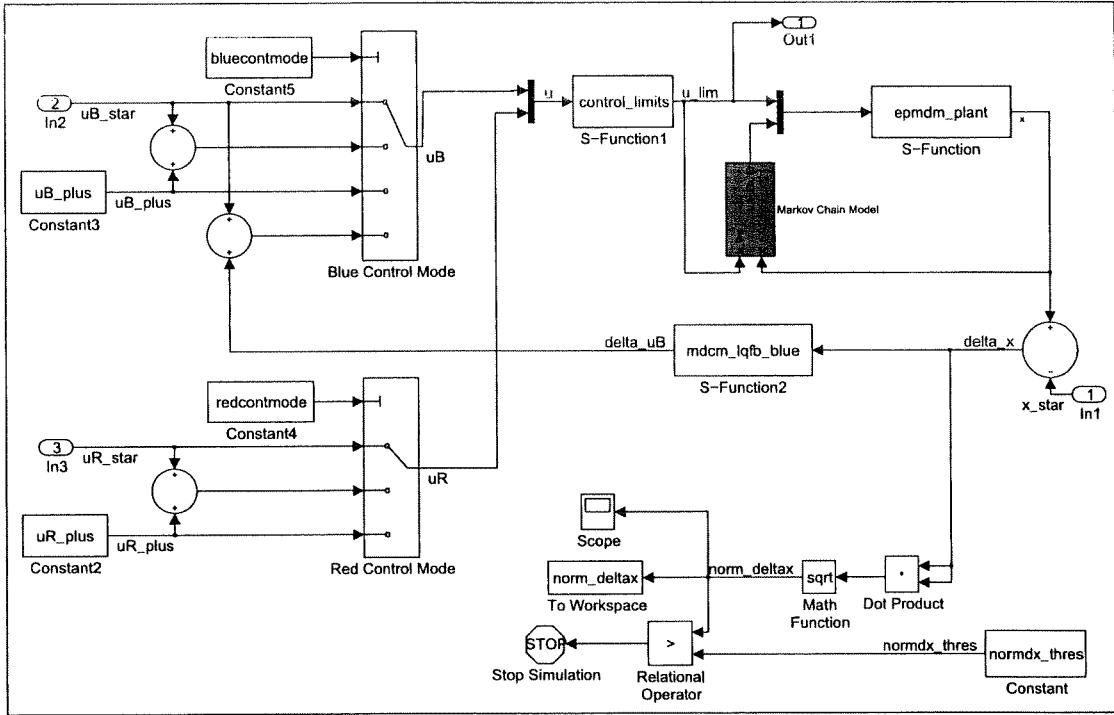


Figure 7.1: Simulink Implementation of EPMDM

Whenever the game needs to recompute the Nash solution, we use the approximated dynamics in MDCM 3.0, in Appendix 1.12

$$\frac{d}{dt}\eta^B = -\lambda^B\eta^R, \quad (7.3)$$

$$\frac{d}{dt}\eta^R = -\lambda^R\eta^B. \quad (7.4)$$

That is, we just drop the summation terms in (7.1) and (7.2). The summation terms correspond to having zero surviving platforms at a certain time t . This probability, for either side, will be small at the beginning of the engagement, but it may grow later. We call (7.3) and (7.4) the internal model, which is just the MDCM.

7.5 Experiment Results

To test the hypothesis, we must compare the game solutions when the plant model is the EPMDM and the MDCM, based on the cost components and the total game cost. If the costs do not differ depending on the plant model we use, then the hypothesis will be verified.

By hypothesis we do not need to add noise to the state variables when constructing observed state variables, so our experiments rely on perfect information. To perform the experiments, the Blue unit always uses the game theoretic algorithm (based on the MDCM), and the Red unit uses the following strategies:

Strategy A: following the Nash solution (based on the MDCM),

Strategy B: a simple heuristic deterministic strategy.

The heuristic strategy is one for which the Red unit takes an assigned path and has an assigned time at which to fire weapons, and Red maintains this course of action no matter what Blue does.

We also use two different scenarios, the *cross* scenario and the *joust* scenario. The *cross* and *joust* scenarios are summarized in Table 7.1. The weights are for the quadratic cost function for the nonlinear game, as described in [3].

Table 7.1: Scenario Description

	<i>cross</i>		<i>joust</i>	
	Blue	Red	Blue	Red
Number of Units	1	1	1	1
Number of Platforms	10	10	10	10
P_k	0.8	0.8	0.8	0.8
ρ	0.5	0.5	0.5	0.5
$\xi^{(1)}(0)(\text{km})$	20.0	50.0	20.0	80.0
$\xi^{(2)}(0)(\text{km})$	50.0	80.0	50.0	52.0
Weight: Distance to Target Cost	0.1	0.1	0.1	0.1
Weight: Running Platform Cost	0.01	3.0	0.2	20.0
Weight: Speed Cost	200.0	200.0	200.0	200.0
Weight: Terminal Platform Cost	0.0	0.0	0.0	0.0
Weight: Terminal Target Cost	0.0	0.0	0.0	0.0
Weight: Terminal Speed Cost	0.0	0.0	0.0	0.0

The figures are organized as follows: Figures 7.2–7.7 are for Strategy A using the *cross* scenario, Figures 7.8–7.13 are for Strategy B using the *cross* scenario, Figures 7.14–7.19 are for Strategy A using the *joust* scenario and last, Figures 7.20–7.25 are for Strategy B using the *joust* scenario. In these four experiments, the figures compare the trajectories computed by the game when the plant model is EPMDM and when the plant model is MDCM. They also compare the game solution for number of platforms, speed controls, fire intensities and weapons expenditures. Note that the compared trajectories are very close, so their plots are indistinguishable in most of the figures.

7.6 Analysis

Examining the figures we can see that the game solutions do not appear to differ significantly. This result is not surprising since the number of platforms on either side decreases only slightly, and we know that MDCM differs noticeably from EPMDM only when one side approaches to zero platforms (as mentioned above and in [2]).

To compare the game solutions with different plant models, we have decided to compare the cost components of the quadratic cost function and the actual game costs. Tables (7.2) and (7.3) summarize these costs for the experiments using the *cross* scenario and Tables (7.4) and (7.5) summarize these costs for the experiments the *joust* scenario. Omitted from the tables is the terminal cost components which are zero. We can clearly see with these results that the difference between using the EPMDM and MDCM for the plant is insignificant.

7.7 Conclusions and Recommendations

With perfect measurement and under closed-loop control, these experiments show that the reduced-order model, MDCM, provides an effective approximation to the more realistic situation, EPMDM. However, we remark that in [2] it was shown that as one side's platforms go to zero, the approximation may not

Table 7.2: Cost Components of Objective Function Using *cross* Scenario

Blue			
Plant Model	Red Control	Running Platform Cost	Running Control Cost
EPMDM	Game	-8172.7	1314.6
MDCM	Game	-8172.6	1316.5
EPMDM	Heuristic	-6181.1	1788.9
MDCM	Heuristic	-6184.0	1789.3
Red			
Plant Model	Red Control	Running Platform Cost	Running Control Cost
EPMDM	Game	-3164.7	852.3
MDCM	Game	-3164.5	853.9
EPMDM	Heuristic	-2088.1	5344.0
MDCM	Heuristic	-2094.0	5344.0

Table 7.3: Total Costs of Objective Function using *cross* Scenario

Plant Model	Red Control	Blue Total Cost	Red Total Cost	Game Cost
EPMDM	Game	-6858.1	2321.4	-4545.7
MDCM	Game	-6586.1	2310.6	-4545.5
EPMDM	Heuristic	-4392.2	-3255.9	-7648.1
MDCM	Heuristic	-4394.7	-3250.0	-7644.7

Table 7.4: Cost Components of Objective Function Using *joust* Scenario

Blue			
Plant Model	Red Control	Running Platform Cost	Running Control Cost
EPMDM	Game	-7703.0	2117.8
MDCM	Game	-7698.9	2123.3
EPMDM	Heuristic	-7141.5	2788.0
MDCM	Heuristic	-7145.5	2787.9
Red			
Plant Model	Red Control	Running Platform Cost	Running Control Cost
EPMDM	Game	1793.3	-2194.7
MDCM	Game	1787.0	-2198.6
EPMDM	Heuristic	4192.8	-5444.0
MDCM	Heuristic	4192.7	-5444.0

Table 7.5: Total Costs of Objective Function Using *joust* Scenario

Plant Model	Red Control	Blue Total Cost	Red Total Cost	Game Cost
EPMDM	Game	-5585.2	-401.4	-5986.6
MDCM	Game	-5575.6	-411.6	-5987.2
EPMDM	Heuristic	-4353.5	-1251.2	-5604.7
MDCM	Heuristic	-4353.6	-1251.3	-5604.9

be as good. That is, if we were to have scenarios for which one side has zero surviving platforms, we may find that the approximation model MDCM may not be as good. Also, the experiments show that approximating the plant model with a lower order internal model does not cause a significant difference in game results, when the adversary uses either a game theoretic controller or a heuristic controller.

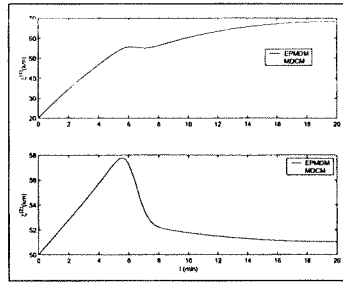


Figure 7.2: Strategy A (*cross*): Blue Trajectory

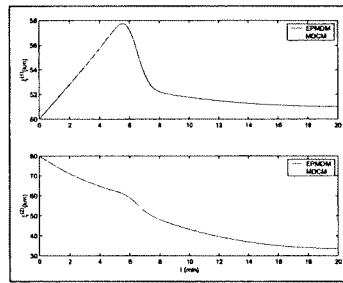


Figure 7.3: Strategy A (*cross*): Red Trajectory

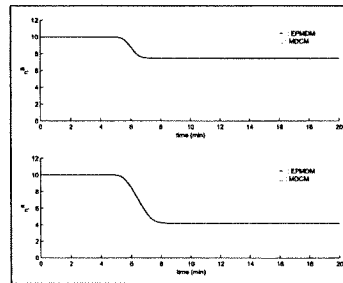


Figure 7.4: Strategy A (*cross*): Number of Platforms

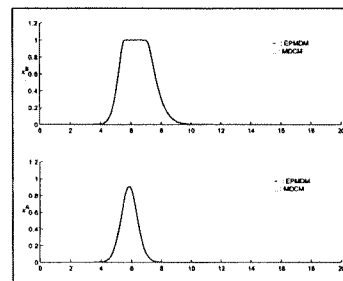


Figure 7.5: Strategy A (*cross*): Fire Intensity

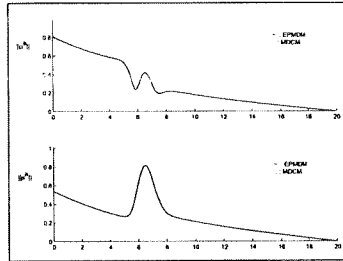


Figure 7.6: Strategy A (*cross*): Speed Control

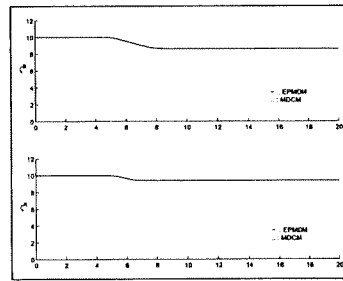


Figure 7.7: Strategy A (*cross*): Weapons Expenditures

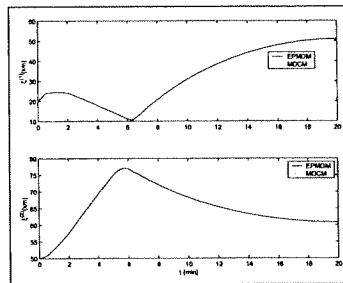


Figure 7.8: Strategy B (*cross*): Blue Trajectory

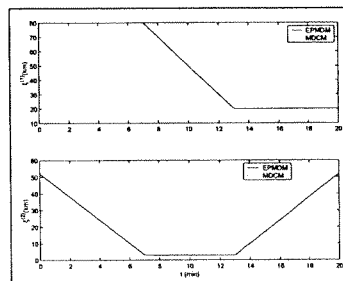


Figure 7.9: Strategy B (*cross*): Red Trajectory

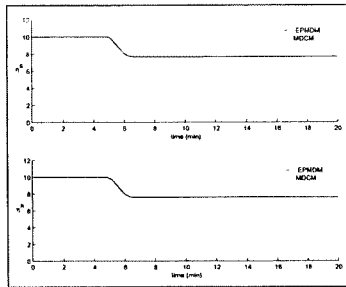


Figure 7.10: Strategy B (*cross*): Number of Platforms

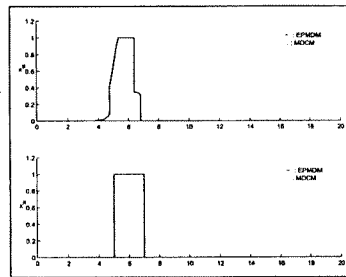


Figure 7.11: Strategy B (*cross*): Fire Intensity

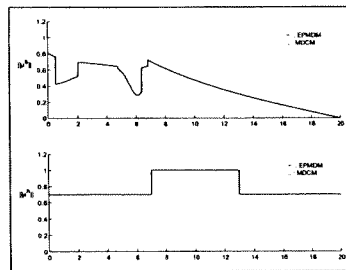


Figure 7.12: Strategy B (*cross*): Speed Control

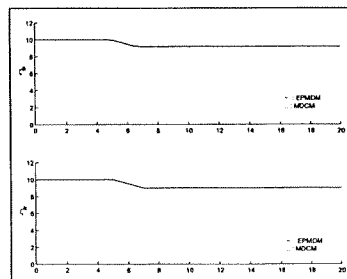


Figure 7.13: Strategy B (*cross*): Weapons Expenditures

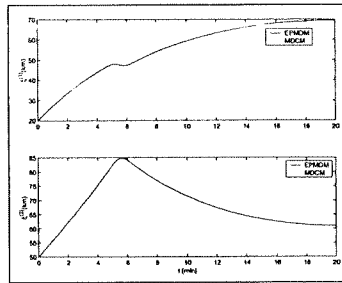


Figure 7.14: Strategy A (*joust*): Blue Trajectory

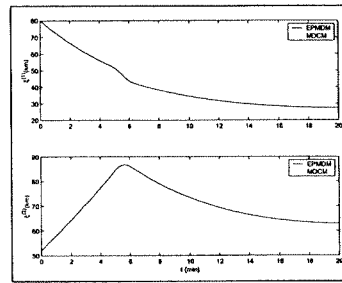


Figure 7.15: Strategy A (*joust*): Red Trajectory

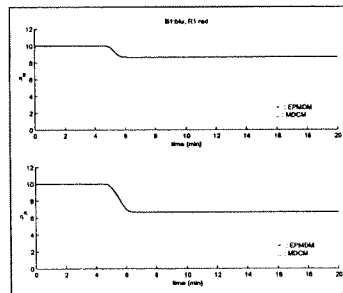


Figure 7.16: Strategy A (*joust*): Number of Platforms

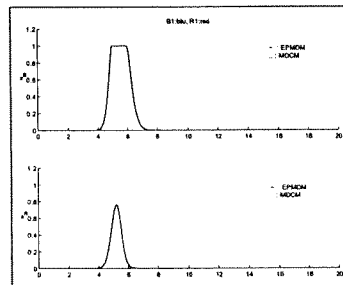


Figure 7.17: Strategy A (*joust*): Fire Intensity

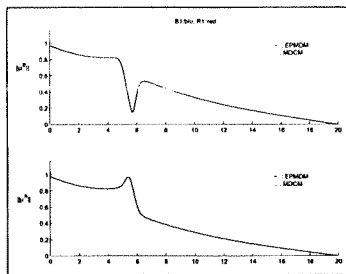


Figure 7.18: Strategy A (*joust*): Speed Control

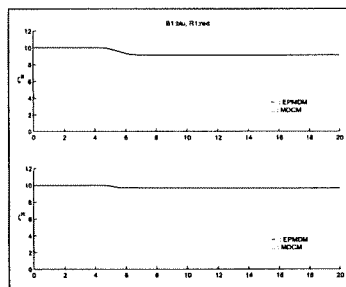


Figure 7.19: Strategy A (*joust*): Weapons Expenditures

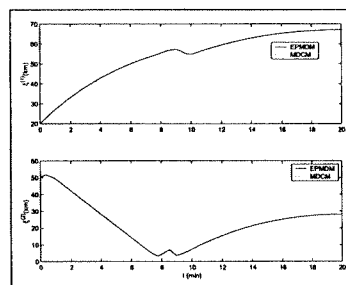


Figure 7.20: Strategy B (*joust*): Blue Trajectory

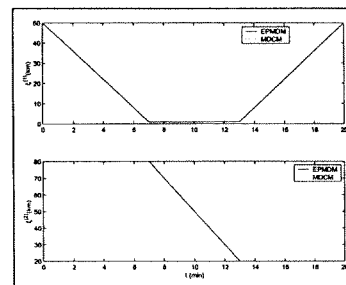


Figure 7.21: Strategy B (*joust*): Red Trajectory

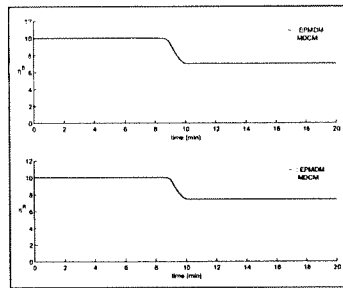


Figure 7.22: Strategy B (*joust*): Number of Platforms

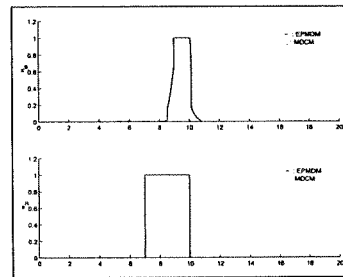


Figure 7.23: Strategy B (*joust*): Fire Intensity

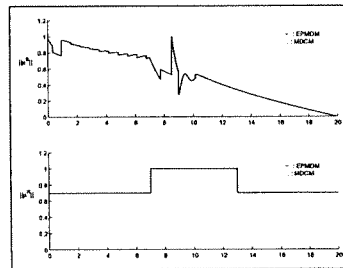


Figure 7.24: Strategy B (*joust*): Speed Control

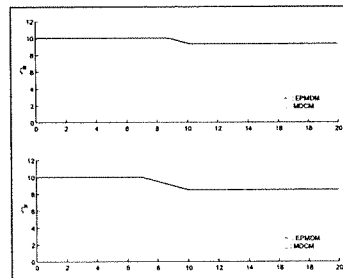


Figure 7.25: Strategy B (*joust*): Weapons Expenditures

Bibliography

- [1] Washington University Experiment Plan version 3.0, January 2001.
- [2] Markov Chain Combat Models for Attrition Prediction and Mission Control, İ. Tunay, J. Goodwin, and H. Mukai, presented and accepted for publication in 3rd Int'l Meeting of INFORMS Military Applications Society (3MAS), (San Antonio, Texas), November 2000.
- [3] Mission Dynamics Continuous-time Model version 2.55, H. Mukai, Y. Sawada, İ. Tunay and P. Girard, April 2000.

Chapter 8

Experiment 8: All Quadratic Method for Nash Computation

8.1 Executive Summary

The purpose of Experiment 8 is to develop, implement and test the Sequential Quadratic-Quadratic Method (SQQM) for Differential Games, with two hypotheses of interest. The first hypothesis tests whether the Nash solution computed through the Sequential Quadratic-Quadratic Method is identical to the one found using the Sequential Linear-Quadratic Algorithm (SLQM); the second hypothesis tests whether there is an improvement in convergence time. The issue of speed can become of great importance in real time applications; moreover, due to the presence of nonlinearity and constraints, a different approach serves the purpose of validating previous results. The algorithm is based on an iterative method for computing a Nash solution to a zero-sum differential game with a system of nonlinear differential equations.

Several experiments on different scenarios, based on both Model 2 and Model 3, have shown the convergence of the outputs of the SQQM and SLQM algorithms to the same solution. So the first hypothesis of Experiment 8 is proven true. As for the second hypothesis, namely an improvement in convergence speed, the conclusion is that the SQQM alone proves to be fast in simple scenarios; if, however, the starting trajectory and costate estimates are too far from the optimal solution, the SLQM may be used at first, and then switch to the SQQM once the solution estimate is closer to the optimal solution. In more complex cases, it is thus advantageous to blend the linear-quadratic algorithm and the quadratic-quadratic algorithm, taking advantage of both the superior stability of the SLQM and the superior speed of the SQQM.

8.2 Purpose of the Experiment

The purpose of Experiment 8 is to develop, implement and test the Sequential Quadratic-Quadratic Method (SQQM) for Differential Games, with the main goal of exploring the possibility of reducing the computational time with respect to the Sequential Linear-Quadratic Algorithm (SLQM).

The algorithm is based on an iterative method for computing a Nash solution to a zero-sum differential game for a system of nonlinear differential equations. Given a solution estimate, a subproblem is defined, which approximates the original problem around the previous solution estimate with a quadratic system dynamics; then, it is replaced with another subproblem which has a quadratic cost and a linear dynamics. Because the latter subproblem has only a linear dynamics, a Riccati equation method can be applied to compute the Nash solution to the subproblem. By adding this Nash solution to the current solution estimate for the original game, a new solution estimate is obtained. Repeating this process, it is possible to successively generate better solution estimates that converge to the Nash solution of the original differential game.

8.3 Hypotheses to Prove or Disprove

There are two hypotheses of interest: the first one tests whether the Nash solution computed through the Sequential Quadratic-Quadratic Method is identical to the one found using the Sequential Linear-Quadratic Algorithm; the second one tests whether there is an improvement in convergence time.

The issue of speed can become of great importance in real time applications; moreover, due to the presence of nonlinearity and constraints, a different approach serves the purpose of validating previous results. The introduction of quadratic terms in the approximation of the plant model along the reference trajectory is taken into account by adjoining it to the cost function expression, by the classical use of an additional costate. On one side, the improved approximation reduces the number of iterations required by the sequential algorithm; on the other side, each iteration requires a longer time to be completed due to the increase in the order of the model involved. As a result, the Quadratic-Quadratic Algorithm may provide a better performance in case the scenario is highly nonlinear. When the nonlinearities do not play a major role, the Linear-Quadratic Algorithm may perform better; this happens, for example, when the initial guess of the costate for the SQQM is far from the optimal one, thus requiring extra time before the quadratic algorithm can actually start converging. Therefore, an algorithm has also been implemented that actually blends both the SQQM and SLQM strategies: after starting with the linear algorithm, a test is routinely made to check whether the quadratic algorithm may take over. This takeover should happen when the linear algorithm generates a solution estimate that it is sufficiently close to the optimal one.

8.4 Experiment Setup

In this section, we report in detail the mathematical formulation of the problem at hand and the implementation of the Sequential Quadratic-Quadratic Method.

8.4.1 Problem and Nash Solutions

Let U denote the set of \mathbb{R}^m -valued continuous functions on $[t_0, t_f]$. Consider a system governed by the ordinary differential equation,

$$\frac{d}{dt}x(t) = f(x(t), u(t)), \quad t \in [t_0, t_f]; \quad x(t_0) = z_0, \quad (8.1)$$

where $f(x, u)$ is an \mathbb{R}^n -valued C^2 -class function on $\mathbb{R}^n \times \mathbb{R}^m$. Given any control $u \in U$ and an initial state $x(t_0) = z_0$, it is assumed that equation (8.1) defines a unique continuously differentiable solution $x(t), t \in [t_0, t_f]$. The solution x is called the *trajectory* of the system produced by control u starting from the initial state z_0 and it is also denoted by $x[u] \in X$, where X is the space of continuously differentiable \mathbb{R}^n -valued functions on $[t_0, t_f]$.

Consider the following game problem. The control function u consists of two parts, u^B and u^R , corresponding to the two forces, the *Blue* and the *Red*: $u = (u^B, u^R)$. As the cost function, consider

$$J(u) = J(u^B, u^R) = \int_{t_0}^{t_f} g(x(t), u(t))dt + g_f(t_f, x(t_f)), \quad (8.2)$$

where g and g_f are general nonquadratic functions and are C^2 -class functions on $[t_0, t_f]$. It is sometimes convenient to consider $J(u)$ as a function of both u and x with an additional constraint (8.1) connecting u and $x = x[u]$, i.e., $J(u) = J[x[u], u]$. The overall game is expressed as the following minimax problem:

$$J^*(t_0, z_0) := \min_{u^B} \max_{u^R} \left\{ J(u^B, u^R) \mid \frac{d}{dt}x(t) = f(x(t), u(t)), \quad x(t_0) = z_0 \right\}, \quad (8.3)$$

where the Red force tries to maximize the cost function $J(u^B, u^R)$ and the Blue force tries to minimize the same cost function $J(u^B, u^R)$.

The control function $u^* = (u^{*B}, u^{*R})$ satisfying the inequalities

$$J(u^{*B}, v^R) \leq J(u^{*B}, u^{*R}) \leq J(v^B, u^{*R}) \quad (8.4)$$

for any $v = (v^B, v^R)$ in a neighborhood of u^* is called a (local) Nash solution to the game problem (8.3). The optimal value $J^*(t_0, z_0) = J(u^*)$ of the cost function $J(u^B, u^R)$ is called the value of the game and it depends on the initial time t_0 and the initial state z_0 .

The proposed iterative process for computing a Nash solution is of the form

$$u_{i+1} = u_i + \alpha_i \delta u_i \quad (8.5)$$

with a step size $\alpha_i \in (0, 1]$. Here, δu_i is a solution to the i -th subproblem which is obtained by applying quadratic (or linear) approximations to g , g_f and f of the original differential game (see Section 3). The following simple proposition suggests that it makes sense to consider the iterative process of the form (8.5) for computing a Nash solution. Here, we consider the simplest iterative process which was proposed in [5] and [6]. The proof of the following proposition can readily be obtained by checking the first order necessary conditions for Nash equilibrium (8.4). In the following, transposition is denoted by a prime and the second-order partial derivatives of the function $g(x, u)$ by $g_{xx}(x, u)$, $g_{xu}(x, u)$, $g_{ux}(x, u)$, and $g_{uu}(x, u)$.

Proposition 1. Suppose that the control $u^* = (u^{*B}, u^{*R})$ is a Nash solution to the problem (8.3). Let $x^* = x[u^*]$ denote its state trajectory. Then the zero solution $(\delta u^B, \delta u^R) = (0, 0)$ is a solution to the subproblem:

$$\begin{aligned} \min_{\delta u^B} \max_{\delta u^R} \left\{ \int_{t_0}^{t_f} \left[g(x^*, u^*) + g_x(x^*, u^*) \delta x + g_u(x^*, u^*) \delta u + \frac{1}{2} \delta x' g_{xx}(x^*, u^*) \delta x \right. \right. \\ \left. \left. + \frac{1}{2} \delta u' g_{ux}(x^*, u^*) \delta x + \frac{1}{2} \delta x' g_{xu}(x^*, u^*) \delta u + \frac{1}{2} \delta u' g_{uu}(x^*, u^*) \delta u \right] dt \right. \\ \left. + g_f(x^*(t_f)) + (g_f)_x(x^*(t_f)) \delta x(t_f) + \frac{1}{2} \delta x(t_f)' (g_f)_{xx}(x^*(t_f)) \delta x(t_f) \right\} \\ \left. \frac{d}{dt} \delta x = f_x(x^*, u^*) \delta x + f_u(x^*, u^*) \delta u, \quad \delta x(t_0) = 0 \right\}, \quad (8.6) \end{aligned}$$

where, in the interest of brevity, the time t is suppressed.

Observe that the cost function in the above subproblem (8.6) is the quadratic approximation of the original cost function in (8.3) around the Nash solution (x^*, u^*) and that the linear differential equation in the above subproblem (8.6) is the linear approximation of the original differential equation in (8.3) around the Nash solution (x^*, u^*) . We now define the Hamiltonian H for the differential game (8.3):

$$H(x(t), u(t), \lambda(t)) = g(x(t), u(t)) + \lambda(t)' f(x(t), u(t)), \quad (8.7)$$

where $\lambda \in X$.

Here, it is assumed for all time t

$$g_{u^B u^R}(x(t), u^B(t), u^R(t)) \equiv 0 \quad \text{and} \quad g_{u^R u^B}(x(t), u^B(t), u^R(t)) \equiv 0 \quad (8.8)$$

and

$$f_{u^B u^R}(x(t), u^B(t), u^R(t)) \equiv 0 \quad \text{and} \quad f_{u^R u^B}(x(t), u^B(t), u^R(t)) \equiv 0. \quad (8.9)$$

Roughly speaking, assumption (8.8) states that there are no cross product terms between Blue and Red controls in the cost. Similarly, assumption (8.9) states that there are no cross product terms between Blue and Red controls in the righthand side of the differential equations. With these assumptions, the following conditions hold for any time t :

$$H_{u^B u^R}(x^*(t), u^{*B}(t), u^{*R}(t), \lambda^*(t)) \equiv 0 \quad \text{and} \quad H_{u^R u^B}(x^*(t), u^{*B}(t), u^{*R}(t), \lambda^*(t)) \equiv 0. \quad (8.10)$$

8.4.2 Sequential Quadratic-Quadratic Method

It is assumed that a solution estimate $u_i = (u_i^B, u_i^R)$ is available and we successively improve it. Let us introduce some notations. Let $x_i = x[u_i]$ be the trajectory of (8.1) corresponding to u_i with $x_i(t_0) = z_0$. Let $\delta u = (\delta u^B, \delta u^R)$ be a small perturbation of u and $x[u_i + \delta u]$ be the trajectory corresponding to control $u_i + \delta u$ with initial condition $x[u_i + \delta u](t_0) = z_0$.

In [5] and [6], we proposed an iterative process, whose i -th step consists of solving the following subproblem, in which the original differential equation (8.1) is linearized around the i -th solution estimate (u_i, x_i) and the original cost function J is approximated by a quadratic function around the i -th solution estimate (u_i, x_i) :

$$\begin{aligned} \min_{\delta u^B} \max_{\delta u^R} & \left\{ \int_{t_0}^{t_f} \left[g(x_i, u_i) + g_x(x_i, u_i)\delta x + g_u(x_i, u_i)\delta u + \frac{1}{2}\delta x' g_{xx}(x_i, u_i)\delta x \right. \right. \\ & + \frac{1}{2}\delta u' g_{ux}(x_i, u_i)\delta x + \frac{1}{2}\delta x' g_{xu}(x_i, u_i)\delta u + \left. \frac{1}{2}\delta u' g_{uu}(x_i, u_i)\delta u \right] dt \\ & + g_f(x_i(t_f)) + (g_f)_x(x_i(t_f))\delta x(t_f) + \frac{1}{2}\delta x(t_f)' (g_f)_{xx}(x_i(t_f))\delta x(t_f) \left. \right\} \\ & \left. \frac{d}{dt}\delta x = f_x(x_i, u_i)\delta x + f_u(x_i, u_i)\delta u, \quad \delta x(t_0) = 0 \right\}, \quad (8.11) \end{aligned}$$

Here, in order to obtain faster convergence than the iterative process based on this *linear*-quadratic approximation, we consider the following *quadratic*-quadratic approximation to the original problem. Namely, at the i -th step, all the functions including the differential equations are approximated by quadratic functions around the i -th solution estimate (u_i, x_i) with $x_i = x[u_i]$:

$$\begin{aligned} \min_{\delta u^B} \max_{\delta u^R} & \left\{ \int_{t_0}^{t_f} \left[g(x_i, u_i) + g_x(x_i, u_i)\delta x + g_u(x_i, u_i)\delta u + \frac{1}{2}\delta x' g_{xx}(x_i, u_i)\delta x \right. \right. \\ & + \frac{1}{2}\delta u' g_{ux}(x_i, u_i)\delta x + \frac{1}{2}\delta x' g_{xu}(x_i, u_i)\delta u + \left. \frac{1}{2}\delta u' g_{uu}(x_i, u_i)\delta u \right] dt \\ & + g_f(x_i(t_f)) + (g_f)_x(x_i(t_f))\delta x(t_f) + \frac{1}{2}\delta x(t_f)' (g_f)_{xx}(x_i(t_f))\delta x(t_f) \left. \right\} \\ & \left. \frac{d}{dt}\delta x^{(j)} = f^{(j)}_x(x_i, u_i)\delta x + f^{(j)}_u(x_i, u_i)\delta u + \frac{1}{2}\tilde{f}^{(j)}(x_i, u_i)[\delta x, \delta u], \quad j = 1, 2, \dots, n, \quad \delta x(t_0) = 0 \right\}, \quad (8.12) \end{aligned}$$

where the second-order terms of the j -th right-hand side $f^{(j)}$ are collected as

$$\begin{aligned} \tilde{f}^{(j)}(x_i; u_i)[\delta x, \delta u] := \\ \delta x' f^{(j)}_{xx}(x_i, u_i)\delta x + \delta u' f^{(j)}_{ux}(x_i, u_i)\delta x + \delta x' f^{(j)}_{xu}(x_i, u_i)\delta u + \delta u' f^{(j)}_{uu}(x_i, u_i)\delta u. \end{aligned}$$

For the sake of notational convenience, the set of n scalar ordinary differential equations in (8.12) will be compactly denoted by the vector differential equation:

$$\frac{d}{dt}\delta x = f_x(x_i, u_i)\delta x + f_u(x_i, u_i)\delta u + \frac{1}{2}F(x_i; u_i)(\delta x, \delta u), \quad \delta x(t_0) = 0. \quad (8.13)$$

Although the quadratic expansions in (8.12) of the original functions, f , g and g_f , around the i -th solution estimate (u_i, x_i) approximate the original problem (8.3) better than the linear-quadratic approximation in (8.11), the i -th subproblem (8.12) obviously is not a linearly constrained quadratic problem. Such nonlinearly constrained problems usually can not be solved easily. So we replace the subproblem (8.12) by a linearly constrained quadratic subproblem, whose solution will be the solution

to the quadratically constrained subproblem (8.12) in the limit as the solution estimates converge to the Nash solution provided a certain parameter (costate $\mu \in X$) is chosen correctly. We thus propose the following linearly constrained subproblem for a given triple (u_i, x_i, μ_i) of control, state and costate with $x_i = x[u_i]$, where μ_i is in the space X of \mathbb{R}^n -valued C^1 -class functions on $[t_0, t_f]$:

$$\begin{aligned} \min_{\delta u^B} \max_{\delta u^R} \left\{ \int_{t_0}^{t_f} \left[g(x_i, u_i) + g_x(x_i, u_i) \delta x + g_u(x_i, u_i) \delta u + \frac{1}{2} \delta x' g_{xx}(x_i, u_i) \delta x \right. \right. \\ \left. \left. + \frac{1}{2} \delta u' g_{ux}(x_i, u_i) \delta x + \frac{1}{2} \delta x' g_{xu}(x_i, u_i) \delta u + \frac{1}{2} \delta u' g_{uu}(x_i, u_i) \delta u \right. \right. \\ \left. \left. + \frac{1}{2} \sum_{j=1}^n \mu_i^{(j)} \bar{f}^{(j)}(x_i, u_i) [\delta x, \delta u] \right] dt \right. \\ \left. + g_f(x_i(t_f)) + (g_f)_x(x_i(t_f)) \delta x(t_f) + \frac{1}{2} \delta x(t_f)' (g_f)_{xx}(x_i(t_f)) \delta x(t_f) \right. \\ \left. \left. \frac{d}{dt} \delta x^{(j)} = f^{(j)}_x(x_i, u_i) \delta x + f^{(j)}_u(x_i, u_i) \delta u, \quad \delta x(t_0) = 0 \right\}. \end{aligned} \quad (8.14)$$

Here, $\mu_i^{(j)}$ denotes the j -th element of an n -dimensional vector-valued function μ_i on $[t_0, t_f]$. In the following, the last term in the integrand of (8.14) will be compactly expressed as

$$\frac{1}{2} \mu_i' F(x_i, u_i) (\delta x, \delta u).$$

Observe that the second-order terms of the quadratic differential equations (8.13) are removed from the differential equations and added to the cost function after taking the inner product with the costate $\mu_i(t)$. Hence, the differential equations in subproblem (8.14) are linear. Because the cost function was quadratic in (8.12), this addition of quadratic terms does not change the quadratic nature of the original cost function in (8.12). The idea of moving the quadratic term of the constraints to the quadratic cost was first proposed by Wilson [9] for optimization in a finite-dimensional context. The convergence analysis of the resulting iterative method is carried out by Robinson [7] and [8] in a finite dimensional context. However, we believe that we are the first to apply the idea to an infinite-dimensional problem in a function space.

As the costate μ_i of the i -th subproblem of (8.14), it is quite natural to choose the costate ν_i corresponding to (u_i, x_i) given by Pontryagin's maximum principle applied to subproblem (8.11). Indeed, following Pontryagin, we define the costate ν_i by

$$\frac{d}{dt} \nu_i(t) = -g_x(x_i(t), u_i(t)) - \sum_{j=1}^m \nu_i^{(j)}(t) f^{(j)}_x(x_i(t), u_i(t)), \quad (8.15)$$

$$\nu_i(t_f) = (g_f)_x(x_i(t_f)). \quad (8.16)$$

After finding the Nash control δu_i for the i -th subproblem (8.14), we update the current control u_i to

$$u_{i+1}(t) = u_i(t) + \alpha_i \delta u_i(t), \quad t \in [t_0, t_f], \quad (8.17)$$

with a step size $\alpha_i \in (0, 1]$. We note that a small step size may be necessary at the beginning for the sake of stability and that the step size of one is recommended towards the end for the sake of fast convergence. We then compute its trajectory

$$x_{i+1}(t) = x[u_{i+1}](t) \quad (8.18)$$

of the original nonlinear differential equation (8.1).

We then update the costate ν_i to ν_{i+1} , which corresponds to the updated costate-state pair (u_{i+1}, x_{i+1}) and which is computed from (8.15)-(8.16) with i replaced by $i + 1$. Finally we can use ν_{i+1} as the next costate μ_{i+1} . However, in order to stabilize this iterative process, we propose to use

$$\mu_{i+1}(t) = \mu_i(t) + \beta_i [\nu_{i+1}(t) - \mu_i(t)], \quad (8.19)$$

with a step size $\beta_i \in (0, 1]$. When the solution estimate (u_i, x_i, μ_i) is far from the local (Nash) optimum (u^*, x^*, λ^*) , we may have to choose relatively small β_i and α_i in order to keep the iterative process stable. However, when the solution estimate (u_i, x_i, μ_i) is close to the local (Nash) optimum (u^*, x^*, λ^*) , we recommend the choice of $\beta_i = 1$ and $\alpha_i = 1$ to make the convergence faster. If, instead of using (8.19), we choose $\mu_i \equiv 0$ for all iterations i , then our iterative process (8.17)-(8.19) reduces to the Sequential Linear-Quadratic Method proposed in [5] and [6].

Note that both subproblems (8.14) and (8.12) reflect the quadratic approximation of the original problem (8.3). However, subproblem (8.14) can be solved more readily (by a Riccati equation method) than subproblem (8.12), because subproblem (8.14) is a *linearly* constrained quadratic problem while (8.12) is a *quadratically* constrained quadratic problem. We expect that the iterative process based on subproblem (8.14) is locally convergent to a local Nash solution and that the convergence is fast, because it is established (Robinson [7] and [8]) for optimization in a finite-dimensional space that the iterative process based on the same idea is locally convergent to a local minimum and that the rate of convergence is quadratic.

8.4.3 Riccati Equation Method

The iterative method described in the previous section requires a technique for solving a linearly constrained quadratic game in order to solve subproblem (8.14). We now present such a technique in this section. The technique consists of solving Riccati differential equations backwards, the linear differential equations forwards and the linear adjoint differential equations backwards. This technique is known as the Riccati equation method.

For the game problem (8.3), the dynamic programming approach requires the *value function*, which is defined by

$$J^*(t, z) = \min_{u^B} \max_{u^R} \left\{ \int_t^{t_f} g(x(\tau), u(\tau), \tau) d\tau + g_f(t_f, x(t_f)) \right. \\ \left. \frac{d}{dt} x(\tau) = f(x(\tau), u(\tau), \tau), \tau \in [t, t_f], x(t) = z \right\}, \quad (8.20)$$

for $t \in [t_0, t_f]$ and $z \in \mathbb{R}^n$. The value function $J^*(t, z)$ satisfies the following boundary condition at time t_f :

$$J^*(t_f, z) = g_f(t_f, z) \quad \text{for any } z \in \mathbb{R}^n. \quad (8.21)$$

Under the assumption of continuous differentiability, a direct application of the principle of optimality to (8.20) yields the so-called Hamilton-Jacobi-Isaacs (HJI) equation,

$$-J_t^*(t, z) = \min_{u^B} \max_{u^R} [J_z^*(t, z) f(z, u, t) + g(z, u, t)], \quad (8.22)$$

which takes (8.21) as a boundary condition. If there exists a function $J^*(t, z)$ satisfying (8.21) and (8.22), then the HJI equation provides a means of obtaining a Nash solution.

We now consider the following affine-quadratic game:

$$\min_{u^B} \max_{u^R} \left\{ J[x; u^B, u^R] \right. \\ \left. \frac{d}{dt} x(t) = A(t)x(t) + B^B(t)u^B(t) + B^R(t)u^R(t) + c(t), \quad x(t_0) = z_0 \right\}, \quad (8.23)$$

where

$$\begin{aligned}
J(u) = \int_{t_0}^{t_f} & \left\{ \frac{1}{2} \begin{pmatrix} x(t)' & u^B(t)' & u^R(t)' \end{pmatrix} \begin{pmatrix} Q(t) & N^B(t) & N^R(t) \\ N^B(t)' & R^B(t) & 0 \\ N^R(t)' & 0 & R^R(t) \end{pmatrix} \begin{pmatrix} x(t) \\ u^B(t) \\ u^R(t) \end{pmatrix} \right. \\
& \left. + \begin{pmatrix} d(t)' & r^B(t)' & r^R(t)' \end{pmatrix} \begin{pmatrix} x(t) \\ u^B(t) \\ u^R(t) \end{pmatrix} \right\} dt + \frac{1}{2} x(t_f)' Q_f x(t_f) + x(t_f)' r_f. \quad (8.24)
\end{aligned}$$

Here, we may suppose that the square matrices $Q(t)$, $R^B(t)$, $R^R(t)$ and Q_f are symmetric. Note that the two control cross product blocks of the quadratic form in the integrand of the cost function (8.24) are identically zero since we assumed (8.8) in order to simplify the minimax problem in the Hamilton-Jacobi-Isaacs (HJI) equation (8.22).

Since one can expect that simple arguments in Anderson and Moore [1] also work for min-max problems, we may assume the value function $J^*(t, z)$ is quadratic in z . Thus, we assume that the value function takes the following form:

$$J^*(t, z) = \frac{1}{2} z' S(t) z + k(t)' z + m(t), \quad (8.25)$$

where $k(t) \in \mathbb{R}^n$, $m(t) \in \mathbb{R}$ and $S(t) \in \mathbb{R}^{n \times n}$ is a symmetric matrix. We may now solve the HJI equation (8.22) explicitly (see Basar and Olsder [2]).

Lemma 2. (Riccati equations) The Hamilton-Jacobi-Isaacs equation (8.22) for the linear-quadratic problem (8.23) has a solution $J^*(t, z)$ of the form (8.25) on $[t_0, t_f] \times \mathbb{R}^n$ if the following system of Riccati equations have a solution (S, k, m) :

$$\begin{aligned}
\frac{d}{dt} S(t) + S(t) A(t) + A(t)' S(t) - \{S(t) B^B(t) + N^B(t)\} R^{B^{-1}}(t) \{B^B(t)' S(t) + N^B(t)'\} \\
- \{S(t) B^R(t) + N^R(t)\} R^{R^{-1}}(t) \{B^R(t)' S(t) + N^R(t)'\} + Q(t) = 0, \quad (8.26)
\end{aligned}$$

$$\begin{aligned}
\frac{d}{dt} k(t) + A(t)' k(t) - \{S(t) B^B(t) + N^B(t)\} R^{B^{-1}}(t) \{B^B(t)' k(t) + r^B(t)\} \\
- \{S(t) B^R(t) + N^R(t)\} R^{R^{-1}}(t) \{B^R(t)' k(t) + r^R(t)\} + S(t) c(t) + d(t) = 0, \quad (8.27)
\end{aligned}$$

$$\begin{aligned}
\frac{d}{dt} m(t) - \frac{1}{2} \{k(t)' B^B(t) + r^B(t)'\} R^{B^{-1}}(t) \{B^B(t)' k(t) + r^B(t)\} \\
- \frac{1}{2} \{k(t)' B^R(t) + r^R(t)'\} R^{R^{-1}}(t) \{B^R(t)' k(t) + r^R(t)\} + k(t)' c(t) = 0, \quad (8.28)
\end{aligned}$$

with the terminal conditions,

$$S(t_f) = Q_f, \quad k(t_f) = r_f, \quad m(t_f) = 0. \quad (8.29)$$

We can obtain the following explicit formula for the Nash control in a state feedback form.

Proposition 3. Suppose that a solution (S, k, m) to the equations (8.26)-(8.28) with (8.29) exists on all of $[t_0, t_f]$. Then a Nash solution u^* to the linear-quadratic differential game (8.23) is found from

$$\begin{aligned}
u^{*B}(t) &= K^B(t) x^*(t) + c^B(t) \\
&\equiv -R^{B^{-1}}(t) \left[\{B^B(t)' S(t) + N^B(t)'\} x^*(t) + B^B(t)' k(t) + r^B(t) \right], \quad (8.30)
\end{aligned}$$

$$\begin{aligned}
u^{*R}(t) &= K^R(t) x^*(t) + c^R(t) \\
&\equiv -R^{R^{-1}}(t) \left[\{B^R(t)' S(t) + N^R(t)'\} x^*(t) + B^R(t)' k(t) + r^R(t) \right], \quad (8.31)
\end{aligned}$$

and the corresponding value is given by

$$J[x^*; u^*] = J^*(t_0, z_0) = \frac{1}{2} z_0' S(t_0) z_0 + k(t_0)' z_0 + m(t_0), \quad (8.32)$$

where $x^* = x[u^*]$ is the state trajectory driven by the Nash control u^* .

The above proposition states that, just like the well known standard form of the Riccati equation method for the regulator problem, we may compute the Nash solution u^* by the following procedure. By substituting (8.30) and (8.31) into the linear ordinary equation in (8.23), we obtain

$$\begin{aligned} \frac{d}{dt} x(t) = & \left[A(t) - B^B(t) R^{B^{-1}}(t) \{ B^B(t)' S(t) + N^B(t)' \} \right. \\ & \left. - B^R(t) R^{R^{-1}}(t) \{ B^R(t)' S(t) + N^R(t)' \} \right] x(t) \\ & + \left[-B^B(t) R^{B^{-1}}(t) \{ B^B(t)' k(t) + r^B(t) \} \right. \\ & \left. - B^R(t) R^{R^{-1}}(t) \{ B^R(t)' k(t) + r^R(t) \} + c(t) \right], \quad x(t_0) = z_0, \end{aligned} \quad (8.33)$$

and can compute its solution x^* from it. Finally, we can compute the optimal control u^* from x^* by (8.30) and (8.31):

$$u^{*B}(t) = -R^{B^{-1}}(t) \left[\{ B^B(t)' S(t) + N^B(t)' \} x^*(t) + B^B(t)' k(t) + r^B(t) \right], \quad (8.34)$$

$$u^{*R}(t) = -R^{R^{-1}}(t) \left[\{ B^R(t)' S(t) + N^R(t)' \} x^*(t) + B^R(t)' k(t) + r^R(t) \right]. \quad (8.35)$$

8.4.4 SQQM Iterative Algorithm for Game Solution

In this section, for nonlinear-quadratic games, we present an iterative algorithm which implements the Sequential Quadratic-Quadratic Method (SQQM) discussed in Section 2 with linear-quadratic solution based on the Riccati equation method in Section 3. Thus, we assume that the cost function $J(u)$ has the form given in (8.24), where the block matrices and component vectors given in the original cost function (8.24) are denoted by barred notations, e.g., respectively by $\bar{Q}(t)$, $\bar{R}^B(t)$, $\bar{R}^R(t)$, $\bar{N}^B(t)$, $\bar{N}^R(t)$, $\bar{d}(t)$, $\bar{r}^B(t)$ and $\bar{r}^R(t)$ to distinguish them from the various matrix-valued functions for the Riccati equations.

Sequential Quadratic-Quadratic Method

Step 0: Select a stopping criterion $\varepsilon > 0$, and an initial control-trajectory-costate triple (u_0, x_0, μ_0) with $x_0 = x[u_0]$, where $\mu_0 \equiv 0$ or μ_0 is determined by

$$\frac{d}{dt} \mu_0(t) = -g_x(x_0(t), u_0(t)) - \sum_{j=1}^m \mu_0^{(j)}(t) f^{(j)}_x(x_0(t), u_0(t)), \quad \mu_0(t_f) = (g_f)_x(x_0(t_f)). \quad (8.36)$$

Set the counter $i = 0$.

Step 1: Set the matrix-valued functions as follows:

$$A(t) = f_x(x_i(t), u_i(t)), \quad B^B(t) = f_{u^B}(x_i(t), u_i(t)), \quad (8.37)$$

$$B^R(t) = f_{u^R}(x_i(t), u_i(t)), \quad c(t) = 0, \quad (8.38)$$

$$d(t) = \bar{d}(t) + \bar{Q}(t)x_i(t) + \bar{N}^B(t)u_i^B(t) + \bar{N}^R(t)u_i^R(t), \quad (8.39)$$

$$r^B(t) = \bar{r}^B(t) + \bar{N}^B(t)x_i(t) + \bar{R}^B(t)u_i^B(t), \quad (8.40)$$

$$r^R(t) = \bar{r}^R(t) + \bar{N}^R(t)x_i(t) + \bar{R}^R(t)u_i^R(t), \quad (8.41)$$

$$Q(t) = \bar{Q}(t) + \mu_i(t)' f_{xx}(x_i(t), u_i(t)), \quad (8.42)$$

$$R^B(t) = \bar{R}^B(t) + \mu_i(t)' f_{u^B u^B}(x_i(t), u_i(t)), \quad (8.43)$$

$$R^R(t) = \bar{R}^R(t) + \mu_i(t)' f_{u^R u^R}(x_i(t), u_i(t)), \quad (8.44)$$

$$N^B(t) = \bar{N}^B(t) + \mu_i(t)' f_{x u^B}(x_i(t), u_i(t)), \quad (8.45)$$

$$N^R(t) = \bar{N}^R(t) + \mu_i(t)' f_{x u^R}(x_i(t), u_i(t)). \quad (8.46)$$

Step 2: Solve the following Riccati equations:

$$\begin{aligned} & \frac{d}{dt}S(t) + S(t)A(t) + A(t)'S(t) \\ & - \{S(t)B^B(t) + N^B(t)\}R^{B^{-1}}(t)\{B^B(t)'S(t) + N^B(t)'\} \\ & - \{S(t)B^R(t) + N^R(t)\}R^{R^{-1}}(t)\{B^R(t)'S(t) + N^R(t)'\} + Q(t) = 0, \end{aligned} \quad (8.47)$$

$$\begin{aligned} & \frac{d}{dt}k(t) + A(t)'k(t) - \{S(t)B^B(t) + N^B(t)\}R^{B^{-1}}(t)\{B^B(t)'k(t) + r^B(t)\} \\ & - \{S(t)B^R(t) + N^R(t)\}R^{R^{-1}}(t)\{B^R(t)'k(t) + r^R(t)\} \\ & + S(t)c(t) + d(t) = 0, \end{aligned} \quad (8.48)$$

$$\begin{aligned} & \frac{d}{dt}m(t) - \frac{1}{2}\{k(t)'B^B(t) + r^B(t)'\}R^{B^{-1}}(t)\{B^B(t)'k(t) + r^B(t)\} \\ & - \frac{1}{2}\{k(t)'B^R(t) + r^R(t)'\}R^{R^{-1}}(t)\{B^R(t)'k(t) + r^R(t)\} \\ & + k(t)'c(t) = 0, \end{aligned} \quad (8.49)$$

backwards from the terminal conditions

$$\begin{aligned} S(t_f) &= \bar{Q}_f, k(t_f) = \bar{r}_f + \bar{Q}_f x_i(t_f), \\ m(t_f) &= 0 \end{aligned} \quad (8.50)$$

and obtain the solution $(S(t), k(t), m(t))$.

Step 3: Solve the linear ordinary differential equation:

$$\begin{aligned} \frac{d}{dt}\delta x_i(t) &= \left[A(t) - B^B(t)R^{B^{-1}}(t)\{B^B(t)'S(t) + N^B(t)'\} \right. \\ & \quad \left. - B^R(t)R^{R^{-1}}(t)\{B^R(t)'S(t) + N^R(t)'\} \right] \delta x_i(t) \\ & + \left[-B^B(t)R^{B^{-1}}(t)\{B^B(t)'k(t) + r^B(t)\} \right. \\ & \quad \left. - B^R(t)R^{R^{-1}}(t)\{B^R(t)'k(t) + r^R(t)\} + c(t) \right], \\ \delta x_i(t_0) &= 0 \end{aligned} \quad (8.51)$$

forwards and obtain the state correction $\delta x_i(t)$.

Step 4: Compute the control correction $\delta u_i(t)$ from the state correction $\delta x_i(t)$ by

$$\begin{aligned}\delta u_i^B(t) &= K^B(t) \delta x_i(t) + c^B(t) \\ &= -R^{B^{-1}}(t) \left[\{B^B(t)' S(t) + N^B(t)'\} \delta x_i(t) + B^B(t)' k(t) + r^B(t) \right]\end{aligned}\quad (8.52)$$

$$\begin{aligned}\delta u_i^R(t) &= K^R(t) \delta x_i(t) + c^R(t) \\ &= -R^{R^{-1}}(t) \left[\{B^R(t)' S(t) + N^R(t)'\} \delta x_i(t) + B^R(t)' k(t) + r^R(t) \right]\end{aligned}\quad (8.53)$$

Step 5: Update the control by

$$u_{i+1}(t) = u_i(t) + \alpha_i \delta u_i(t) \quad (8.54)$$

with a step size $\alpha_i \in (0, 1]$.

Step 6: Compute $x_{i+1} = x[u_{i+1}]$ by solving the original differential equation forwards:

$$\frac{d}{dt} x_{i+1}(t) = f(x_{i+1}(t), u_{i+1}(t)), \quad x_{i+1}(t_0) = z_0. \quad (8.55)$$

Step 7: Compute the new costate ν_{i+1} by solving the adjoint differential equation:

$$\begin{aligned}\frac{d}{dt} \nu_{i+1}(t) &= -g_x(x_{i+1}(t), u_{i+1}(t)) - \sum_{j=1}^m \nu_{i+1}^{(j)}(t) f_x^{(j)}(x_{i+1}(t), u_{i+1}(t)) \\ &= -\bar{d}(t) - \bar{Q}(t) x_{i+1}(t) - \bar{N}^B(t) u_{i+1}^B(t) - \bar{N}^R(t) u_{i+1}^R(t) - \nu_{i+1}(t)' A(t)\end{aligned}\quad (8.56)$$

backwards from the terminal condition $\nu_{i+1}(t_f) = \bar{r}_f + \bar{Q}_f x_{i+1}(t_f)$.

Step 8: Update the costate by

$$\mu_{i+1}(t) = \mu_i(t) + \beta_i [\nu_{i+1}(t) - \mu_i(t)] \quad (8.57)$$

with a step size $\beta_i \in (0, 1]$.

Step 9: If $\|\delta u_i\| \leq \varepsilon$, stop. Otherwise, go to Step 1 with i replaced by $i + 1$.

Remark 4. When the estimate (u_i, x_i, μ_i) is far from a local (Nash) optimum (u^*, x^*, λ^*) , we may choose positive numbers less than one for the step sizes α_i and β_i in order to keep the iterative process stable. However, when the estimate (u_i, x_i, μ_i) is close to a local (Nash) optimum (u^*, x^*, λ^*) , we recommend the choice of $\alpha_i = 1$ and $\beta_i = 1$ to make the iterative process converge faster. Recall that $\delta u^* \equiv 0$ is a necessary condition for the Nash solution u^* . Hence

$$\|\delta u_i\| := \sqrt{(t_f - t_0)^{-1} \int_{t_0}^{t_f} \sum_{1 \leq j \leq m} |\delta u_i^{(j)}(t)|^2 dt} \quad (8.58)$$

or

$$\|\delta u_i\| := \sum_{1 \leq j \leq m} \sup_{t_0 \leq t \leq t_f} |\delta u_i^{(j)}(t)| \quad (8.59)$$

may be used to measure the proximity of the current solution estimate u_i to the Nash solution u^* .

8.5 Experiment Results and Analysis

The cases shown here cover both the scenario for 1 unit vs. 1 unit and the scenario for 5 units vs. 5 units, for both Model 2 and Model 3. The results for 1 unit vs. 1 unit are shown in Figure 8.1 and Figure 8.2. The results for 5 units vs. 5 units are shown in Figure 8.3 and Figure 8.4. Each Blue unit starts with 10 interceptors, and each Red unit starts with 10 bombers. In these experiments¹, each force has two objectives: i) to reach its specified fixed destination target, and ii) to reduce the number of enemy platforms while preserving the number of its own platforms as many as possible.

We performed the experiment starting from an initial solution estimate, whose initial velocity controls consist of a constant control so that its resulting trajectory is the straight line from its initial location to its destination target's location. As an initial firing intensity control for each force, we chose constant functions as well.

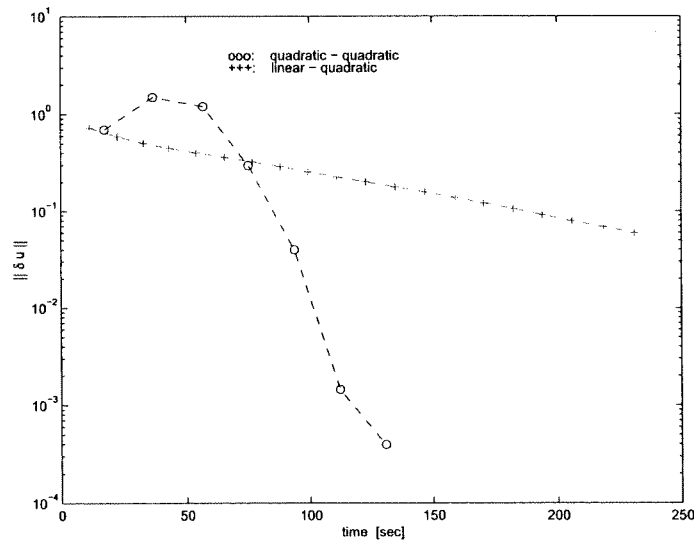


Figure 8.1: Convergence of SQQM and SLQM (Model 2, 1 unit vs. 1 unit).

The actual implementation in Matlab of the Quadratic-Quadratic Algorithm is based on the implementation for the Linear-Quadratic Algorithm. The main difference resides in the introduction of the quadratic term of the plant model, by way of a costate, to the cost function. As the value of the costate is also recomputed at each iteration (by solving a system of ordinary differential equations), the choice of its initial value greatly influences the convergence and speed of the iterative process. As the default, the initial value of the costate is now set to zero; if a better initial value is found by other means (e.g. method of characteristics), a faster convergence may be attained.

The iterations are performed by specifying the following parameters: the maximum number of iterations, the threshold for the norm of the control correction $\|\delta u_i\|$ under which convergence is declared, the step-size α_i for the control refinement and the step-size β_i for the costate refinement. A conservative choice, i.e. smaller values, of the last two parameters, α_i and β_i , is safer for the stability of the sequential iterations, but at the price of a slower convergence speed.

A special remark must be made about numerical precision: as expected from theoretical results, the Quadratic-Quadratic Algorithm is more sensitive to numerical round-offs and approximations than the Linear-Quadratic Algorithm is. Thus, to arrive at a small value of the norm of the control increment $\|\delta u\|$, it might be necessary to increase the relative tolerance and/or the absolute tolerance of the chosen integration algorithms; for this purpose, we made use of integration 'options' (as provided in Matlab) at

¹These scenarios are typical of simulations described in greater detail in other chapters; here our goal is to compare the behavior of the SQQM algorithm with that of the SLQM algorithm.

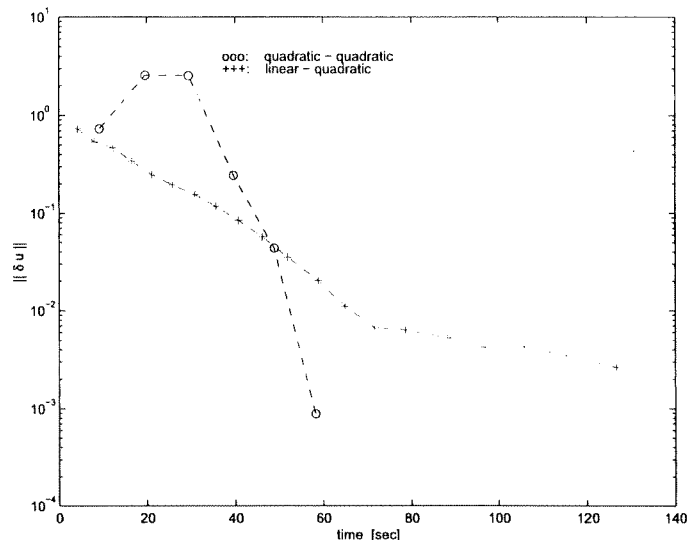


Figure 8.2: Convergence of SQQM and SLQM (Model 3, 1 unit vs. 1 unit).

each call of the integration routines. The current default values have been selected after running several scenarios, obtaining a reasonable compromise between speed and accuracy.

Finally, we mention that the quadratic algorithm numerically requires the Hessian of the plant model. The subprograms for computing the Hessians have been built using the symbolic math tool of the Maple core in Matlab; this procedure for building the subprograms has been automated, so as to be able to switch easily between models (i.e., between Model 2 and Model 3). When the size of the problem increases, the computational work required by the evaluation of the Hessian can considerably increase the computational time.

Several experiments on different scenarios, based on both Model 2 and Model 3, have shown the convergence of the outputs of the SQQM and SLQM algorithms to the same solution. As for the speed, the SQQM has generally proven to be faster in scenarios not too complex, both for Model 2 and Model 3. For example, in Figure 8.1 and Figure 8.2, the norm $\|\delta u\|$ of the control correction for the SLQM is shown as a function of the time needed by the iterative procedure. For the SLQM, the norm reduces slowly but steadily; for the SQQM, instead, it first increases, due perhaps to a poor initial choice of the costate μ_0 , but then decreases much more rapidly than the SLQM. Note that the time needed for the computation of each iteration is almost halved, for both the SLQM and the SQQM, when using Model 3 instead of Model 2, due to a simplification in model structure; this can be seen in all the figures.

For higher order models, the computational time required at each iteration becomes very large, and the SLQM in general performs better, as shown in Figure 8.3 and Figure 8.4 for a scenario of 5 units vs. 5 units. We need to point out the difference in the computational time needed for one iteration between the two methods. The horizontal distance between successive points indicates the time needed to complete one iteration. For example, one iteration of the SQQM takes about 60 seconds, while one iteration of the SLQM takes only 15 seconds in Figure 8.4. Indeed, when the number of units increases, the small number of iterations of the quadratic algorithm cannot compensate for the computational burden of evaluating the Hessian along the estimated trajectory. The choice of the initial value for the costate μ_i in the SQQM is critical, in the sense that the simulation may stop before the costate starts to converge.

In order to surmount this seeming weakness of the SQQM algorithm for complex cases, we devised an alternative method by using both the linear-quadratic and the quadratic-quadratic methods. First, the linear-quadratic algorithm is started and a test is routinely performed in order to monitor if the quadratic-quadratic algorithm can take over, namely when the correction norm at the next estimate computed by the SQQM is smaller than the correction norm at the next estimate computed by the

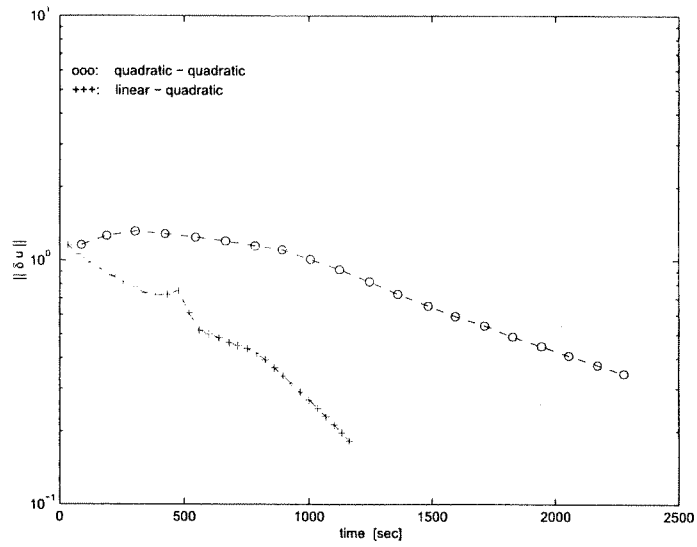


Figure 8.3: Convergence of SQQM and SLQM (Model 2, 5 units vs. 5 units).

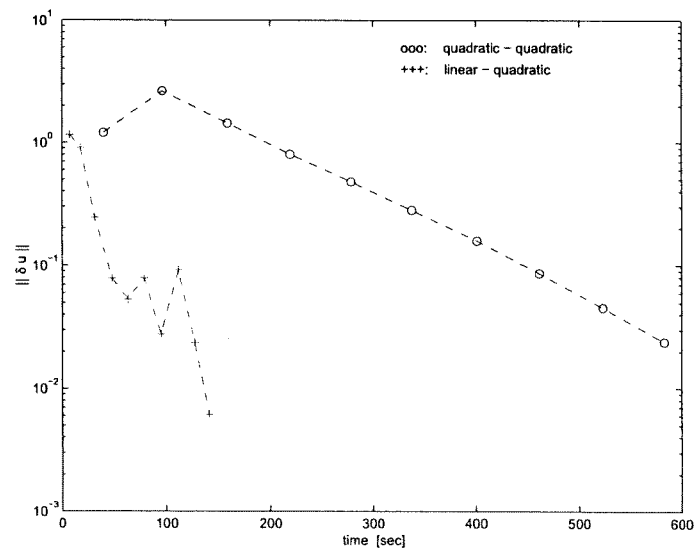


Figure 8.4: Convergence of SQQM and SLQM (Model 3, 5 units vs. 5 units).

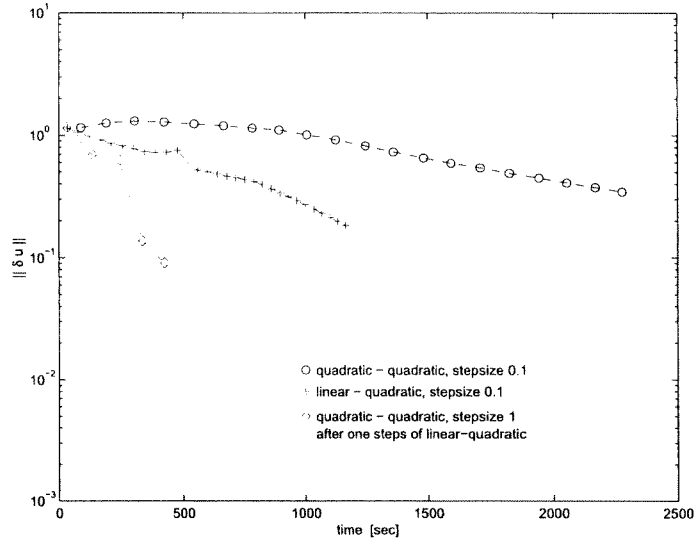


Figure 8.5: Convergence of SQQM, SLQM and SLQM-SQQM (Model 2, 5 units vs. 5 units).

SLQM. Indeed, the linear-quadratic algorithm serves the purpose of reaching an estimate solution which is sufficiently close to the optimal solution, and the purpose of supplying a costate estimate when it is possible to switch to the quadratic-quadratic algorithm. For example, in Figure 8.5, the diamond line shows the performance of the blended method. Indeed the SQQM takes over after just one step of the SLQM, and at that point the value of the step-size for the costate update β_i can be increased to its maximum value of 1, increasing considerably the convergence rate. Indeed, the norm of the correction in the diamond line reduces very quickly to 10^{-1} . Such a blend of the two methods gives the best results: once the SQQM takes over, it converges to the optimal solution more rapidly than the SLQM alone or the SQQM alone. Note that parameters are to be set, such as the step-sizes for control update, costate update and switching conditions.

8.6 Conclusions and Recommendations

The Sequential Quadratic-Quadratic Algorithm converges to the same solution found through the Sequential Linear-Quadratic Algorithm, for all models and scenarios. So the first hypothesis of Experiment 8 is proven true. About the second one, namely an improvement in convergence speed, the conclusion is clear at this point. Indeed, the SQQM alone proves to be faster in simple scenarios; if, however, the starting trajectory and costate estimates are too far from the optimal solution, the SLQM may be used at first, and then switch to the SQQM once the solution estimate is close to the optimal solution. In more complex cases it is thus advantageous to blend the linear-quadratic algorithm and the quadratic-quadratic algorithm, taking advantage of both the superior stability of the SLQM and the superior speed of the SQQM.

Bibliography

- [1] B. Anderson and J. B. Moore, *Optimal Control*, Prentice-Hall, Englewood Cliffs, NJ, 1989.
- [2] T. Basar and G. J. Olsder, *Dynamic Noncooperative Game Theory*, second edition, Academic Press, New York, New York, 1995.
- [3] A. E. Bryson, Jr. and Y. C. Ho, *Applied Optimal Control*, Blaisdell Publishing Company, Waltham, Massachusetts, 1969.
- [4] D. G. Luenberger, *Optimization by Vector Space Method*, John Wiley and Sons, New York, New York, 1969.
- [5] H. Mukai, et al., Sequential linear quadratic method for differential games, in *Proceedings of the 2nd DARPA-JFACC Symposium on Advances in Enterprise Control*, 159-168, Minneapolis, MN, July 2000.
- [6] H. Mukai, et al., Game-theoretic linear-quadratic method for air mission control, in *Proc. 39th IEEE Conf. Decision and Control*, 2574-2580, Sydney, Australia, Dec. 2000.
- [7] S. M. Robinson, A quadratically-convergent algorithm for general nonlinear programming problems, *Math. Programming* **3**, 145–156 (1972).
- [8] S. M. Robinson, Perturbed Kuhn-Tucker points and rates of convergence for a class of nonlinear-programming algorithms, *Math. Programming* **7**, 1–16 (1974).
- [9] R. B. Wilson, A simplicial algorithm for concave programming, Ph.D. Dissertation, Graduate School of Business Administration, Harvard University, Cambridge, Mass., 1963.

Chapter 9

Experiment 9: Detector Performance under Noise

9.1 Executive Summary

In this Chapter we report the experiments performed to test the effectiveness of a *newly designed* “game-theoretic-optimal” detection filter in handling noise-corrupted observations of the battlefield. The basic purpose of the detection filter is to reveal the occurrence of an “engagement action” from enemy units by monitoring only variables associated with the friendly units. The game-theoretic approach to the design of the filter makes it possible to attenuate the effects of measurement noises, but not the effects of the action to be detected. The outcome of the experiments shows very clearly that the game-theoretic filter is very effective under different situations of noise and compares very favorably with a filter designed on the basis of classical state-estimation methods.

9.2 Purpose of the Experiment

This section of the report describes experiments on detection and isolation of multiple enemy actions in a battlefield. Specifically, the basic purpose of this first series of experiments is *to test the effectiveness of a newly-designed* (“game-theoretic”) *filter in handling noise-corrupted observations* of the battlefield.

The mathematical description of the battlefield used here is the one introduced in [1]. We consider the case in which two opposing forces are present in the theater of operations, the Blue force (the “friends”) and the Red force (the “enemies”). Each force consists of two units and each unit consists of a number of platforms whose evolution in time is described by a first order nonlinear differential equation and depends on the “actions” which the opposing units are performing against the unit in question. If any “new” action is performed by any of the opposing units, this affects the evolution of the number of platforms of the other force’s units. Letting η_i^R and η_i^B , with $i = 1, 2$, denote the number of platforms of the i -th Red and – respectively – i -th Blue unit, the model in question is a four-dimensional nonlinear system described by two pairs of equations of the form (cf. [1])

$$\begin{aligned} \frac{d}{dt}\eta_i^R(t) &= -\eta_i^R(t) \\ &\times \left[\alpha^R + \sum_{j=1}^2 \left\{ \left[1 - e^{-\sigma_{ji}^B \left\{ \frac{\eta_j^B(t)\pi_{ji}^B(t)}{\eta_i^R(t)} \right\}} \right] \beta_{eji}^B \left[1 - \{1 - P_{0ji}^B \varphi_{ji}^B(|\tilde{\eta}_i^R(t) - \tilde{\eta}_j^B(t)|)\beta_{kji}^B\}^{n_{ji}^B} \right] \right\} \right], \\ \frac{d}{dt}\eta_i^B(t) &= -\eta_i^B(t) \\ &\times \left[\alpha^B + \sum_{j=1}^2 \left\{ \left[1 - e^{-\sigma_{ji}^R \left\{ \frac{\eta_j^R(t)\pi_{ji}^R(t)}{\eta_i^B(t)} \right\}} \right] \beta_{eji}^R \left[1 - \{1 - P_{0ji}^R \varphi_{ji}^R(|\tilde{\eta}_i^B(t) - \tilde{\eta}_j^R(t)|)\beta_{kji}^R\}^{n_{ji}^R} \right] \right\} \right]. \end{aligned}$$

In these equations, $\pi_{ji}^R(\cdot)$ and $\pi_{ji}^B(\cdot)$ are (independent) input variables representing the “level of engagement” of the j -th Red unit with the i -th Blue unit and – respectively – of the j -th Blue unit with the i -th Red unit. For convenience, we suppose in all our experiments that

$$\pi_{11}^R(t) = \pi_{12}^R(t) =: \pi_1^R(t), \quad \pi_{21}^R(t) = \pi_{22}^R(t) =: \pi_2^R(t).$$

This means, in the terminology of [1], page 2, that we allow the “unique target constraint” to be violated (for the Red units only).

The basic problem addressed in our series of experiments on the design of filters for the detection of enemy actions is the following one: *we monitor only the number of platforms in the two Blue units* (i.e. we measure only the values of the two state variables η_1^B, η_2^B) and *we want to detect the occurrence of an “engagement action” from either one of the two Red units* (i.e. we want to detect when either one of the two input signals $\pi_i^R(\cdot)$ has become nonzero). Implicit in this is the assumption that the two other state variables η_1^R, η_2^R (number of platforms of the two enemy units) as well as all the input variables $\pi_{ij}^B, i, j = 1, 2$, and $\pi_i^R, i = 1, 2$ are not monitored. The purpose of the detection process is precisely the determination of when either π_1^R or π_2^R has become nonzero, without having it directly measured.

9.3 Hypothesis to Prove or Disprove

In the first series of experiments, conceived to test the effectiveness of our “game-theoretic” detection filter, we consider, instead of the full nonlinear model (9.1), a *bilinear approximation*.

This bilinear model has the form (see [1], equation (29))

$$\begin{aligned} \frac{d}{dt} \eta_1^R(t) &= -\alpha^R \eta_1^R(t) - \gamma_{11}^R \eta_1^B(t) \pi_{11}^B(t) - \gamma_{12}^R \eta_2^B(t) \pi_{21}^B(t), \\ \frac{d}{dt} \eta_2^R(t) &= -\alpha^R \eta_2^R(t) - \gamma_{21}^R \eta_1^B(t) \pi_{12}^B(t) - \gamma_{22}^R \eta_2^B(t) \pi_{22}^B(t), \\ \frac{d}{dt} \eta_1^B(t) &= -\alpha^B \eta_1^B(t) - \gamma_{11}^B \eta_1^R(t) \pi_{11}^R(t) - \gamma_{12}^B \eta_2^R(t) \pi_{21}^R(t), \\ \frac{d}{dt} \eta_2^B(t) &= -\alpha^B \eta_2^B(t) - \gamma_{21}^B \eta_1^R(t) \pi_{12}^R(t) - \gamma_{22}^B \eta_2^R(t) \pi_{22}^R(t). \end{aligned} \tag{9.1}$$

The structure of the filter that we use to solve the detection problem described in the previous section is chosen according to a general (differential-geometric) methodology developed in our papers [2], [3]. This filter *receives as inputs* the two observed variables η_1^B, η_2^B and *generates as outputs* two signals r_1, r_2 , called *performance signals* (typically known also with the name of *residuals*), in such a way that $r_i(t)$ is zero if the Red unit i is not engaged with the Blue units at time t (i.e. if $\pi_i^R(t) = 0$), and that $r_i(t)$ is nonzero if the Red unit i is engaged with the Blue units (i.e. if $\pi_i^R(t) \neq 0$). Specifically, this filter is modeled by equations of the form

$$\begin{aligned} \dot{\tilde{\eta}}_1(t) &= -\alpha^B \tilde{\eta}_1(t) + g_1(\eta_1^B(t) - \frac{\gamma_{12}^R}{\gamma_{22}^R} \eta_2^B(t) - \tilde{\eta}_1(t)), \\ \dot{\tilde{\eta}}_2(t) &= -\alpha^B \tilde{\eta}_2(t) + g_2(\eta_2^B(t) - \frac{\gamma_{21}^R}{\gamma_{11}^R} \eta_1^B(t) - \tilde{\eta}_2(t)), \\ r_1(t) &= \eta_1^B(t) - \frac{\gamma_{12}^R}{\gamma_{22}^R} \eta_2^B(t) - \tilde{\eta}_1(t), \\ r_2(t) &= \eta_2^B(t) - \frac{\gamma_{21}^R}{\gamma_{11}^R} \eta_1^B(t) - \tilde{\eta}_2(t). \end{aligned} \tag{9.2}$$

in which the “gain parameters” g_1 and g_2 are to be determined in some “optimal” way.

The basic issue that makes the design of the filter (i.e. of the gain parameters g_1 and g_2) *critical* is the presence of *measurement noises* on the two monitored variables η_1^B, η_2^B (the two inputs to the filter (9.2)). As a matter of fact, standard methods for state estimation from noisy observations (such as those used in the classical Kalman filter) may well reduce the effect of measurement noise on the performance signals r_1, r_2 , but only at the expenses of a *reduction of the sensitivity* of the performance signals to the signals π_1^R, π_2^R that need to be detected. In other words, these methods tend to uniformly filter out the noises affecting the measurements as well as the signals that need to be recognized. This unpleasant circumstance was observed in our first series of experiments, in which a design technique inspired to the theory of Kalman filter was used.

In order to improve the “diagnostic” capability of the detection filter, it is of primary importance to *selectively reduce* the effect of the measurement noise while not attenuating the signal associated with the action to detect. We have achieved this goal by casting the detection problem in a *game-theoretic* framework in which the measurement noise and the event to detect are seen as opposing players (see [4]).

The series of experiments described in this section of the report validates the effectiveness of our “game-theoretic” filter, and demonstrates how this filter compares very favorably with a filter in which the gain parameters are not chosen in a game-theoretical optimal way, but rather on the basis of classical state-estimation methods (in what follows, we will refer to the latter as to a “Kalman-like” filter).

9.4 Experiment Setup

This series of experiments of detection of enemy actions is designed in the following way. The battlefield is modeled as in equation (9.1), whose four states represent the number of platforms of each of the four units involved in the battle. The inputs variables, representing the level of engagement of the battling units, are fixed functions of time. In particular, the two levels of engagement of the Red units 1 and 2 versus the Blue units vary with time as shown in Figure 9.1, where “Action 1” represents the level of engagement of Red unit 1 and “Action 2” represents the level of engagement of Red unit 2. Note that the first action occurs at $t = 30$ units of time, whereas the second action takes place at $t = 50$ units of time and this while the first action is still occurring. The corresponding behavior in time of the number of Red and Blue platforms in each unit is plotted in Figure 9.2. Finally, Figure 9.3 depicts the outcome of the two observations, namely the evolution in time of the number of platforms in the two Blue units, *corrupted by measurement noise* (compare with the two bottom graphs in Figure 9.2, depicting the same quantities without measurement noise).

Figure 9.4 shows a block diagram describing the experiment. The two (noise-corrupted) measured observations are fed into the filter: the Output 1 of this filter is expected to reveal the occurrence of Action 1, while the Output 2 is expected to reveal the occurrence of Action 2. The purpose of the experiment is to compare the effectiveness of the detection process with respect to two different choices of the gain parameters g_1, g_2 in the detection filter (9.2). In the first choice, that leads to what we refer to as a “Kalman-like” filter, each parameter $g_i, i = 1, 2$, is chosen in such a way as to render

$$\sup_{v_1, v_2} \int_0^{t_1} [|r_i|_Q^2 - \gamma^2(|v_1|_{V-1}^2 + |v_2|_{M-1}^2)] dt \leq 0, \text{ for } i = 1, 2, \quad (9.3)$$

where $[0, t_1]$ is the interval of time over which the experiment is performed, $r_i, i = 1, 2$ are the variables as defined in (9.2), $v_i, \text{ for } i = 1, 2$, are the noise signals affecting the measurements of η_i^B , Q, M, V are suitable weighting matrices, and γ is a threshold parameter representing the attenuation of the noise on the residual. In the second choice, that leads to what we refer to as a “game-theoretic” filter, each parameter $g_i, i = 1, 2$, is chosen in such a way as to render

$$\sup_{v_1, v_2} \inf_{\pi_i^R} \int_0^{t_1} [|r_i|_Q^2 + |\pi_i^R|_{N-1} - \gamma^2(|v_1|_{V-1}^2 + |v_2|_{M-1}^2)] dt \leq 0, \text{ for } i = 1, 2, \quad (9.4)$$

in which $\pi_i^R, i = 1, 2$, are the engagement enemy actions (which must be detected), N is a weighting matrix and the other variables and parameters are as in (9.3).

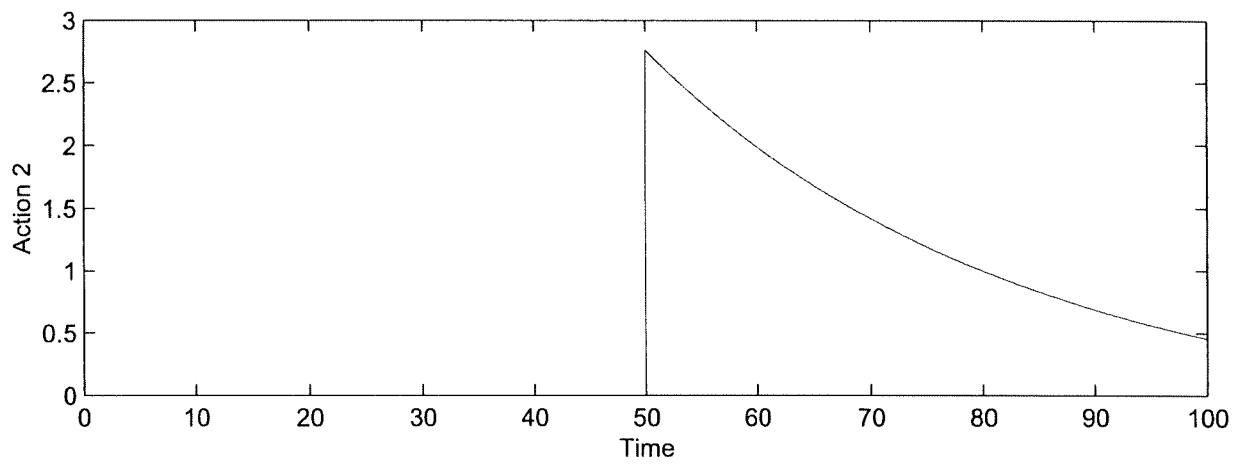
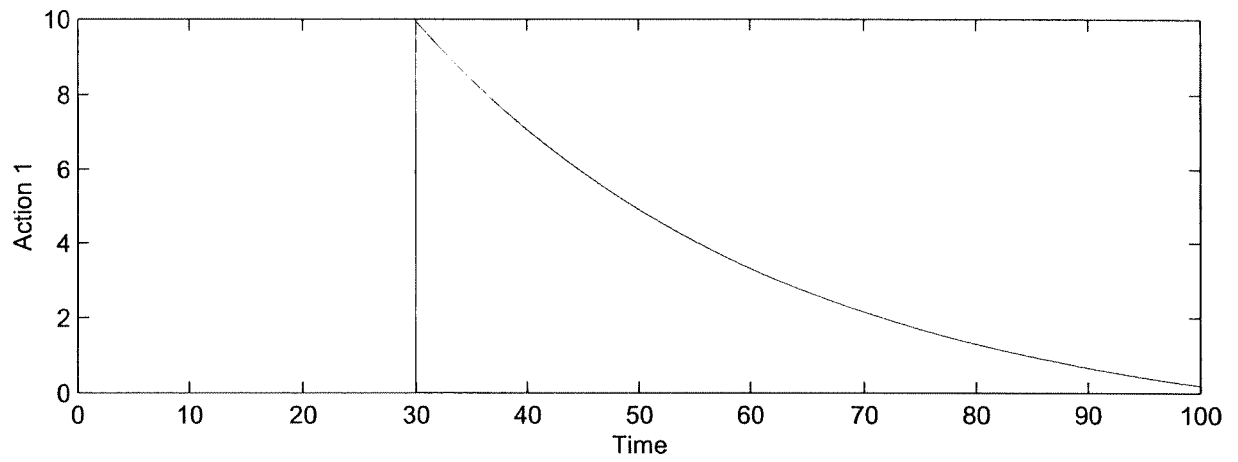


Figure 9.1: Engagement enemy actions (to be detected).

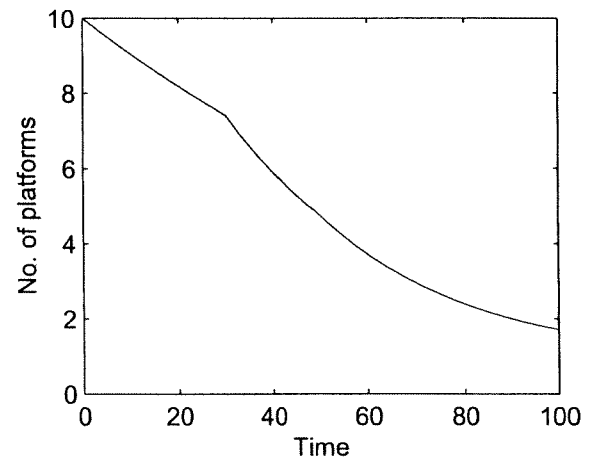
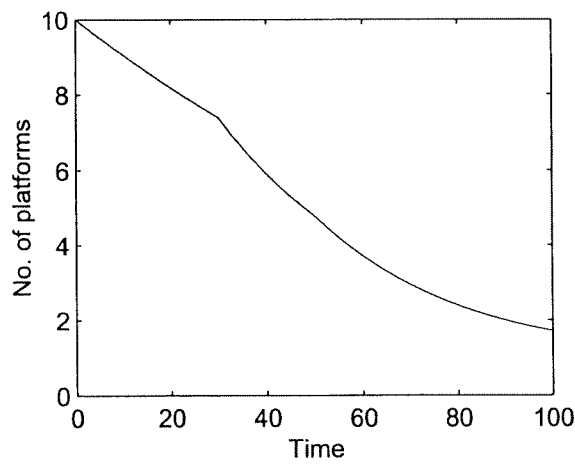
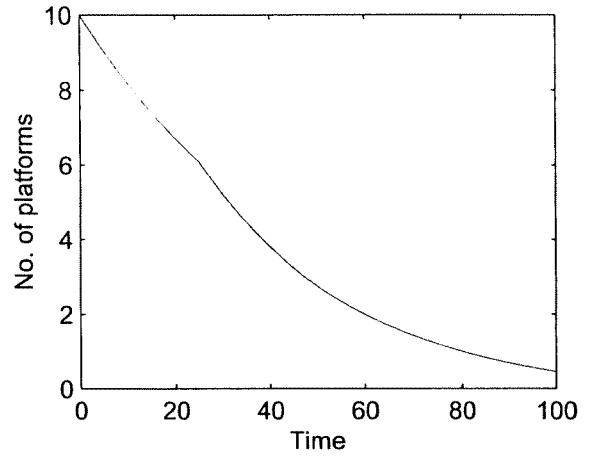
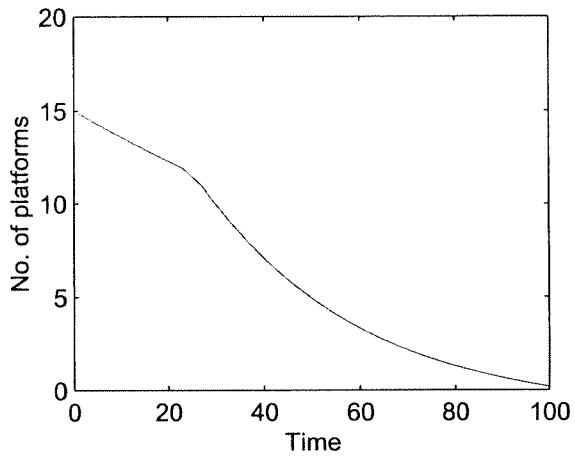


Figure 9.2: Time history of the number of platforms of the four units. The two top plots represent the number of platforms of the Red units, the two bottom plots represent the number of platforms of the Blue units.

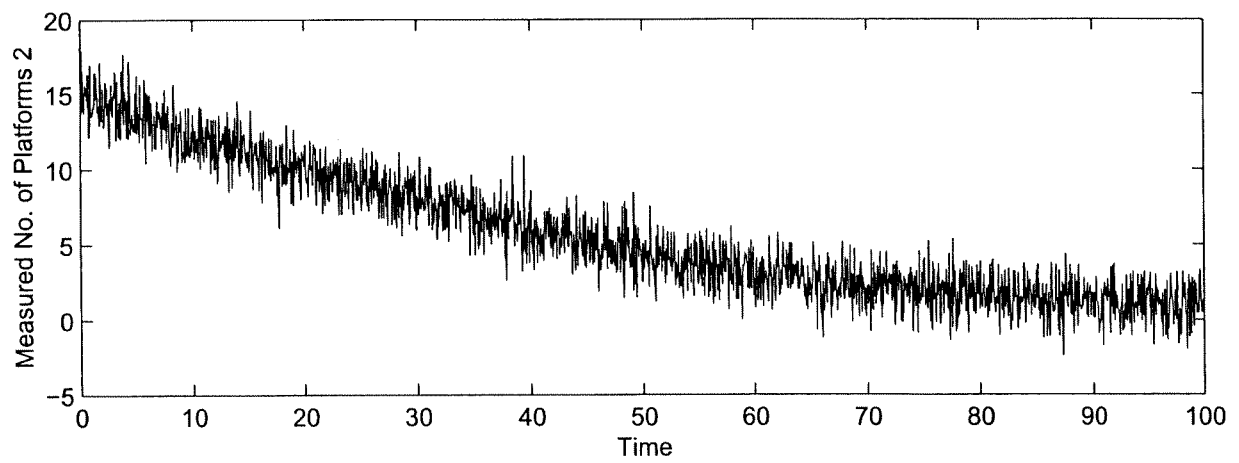
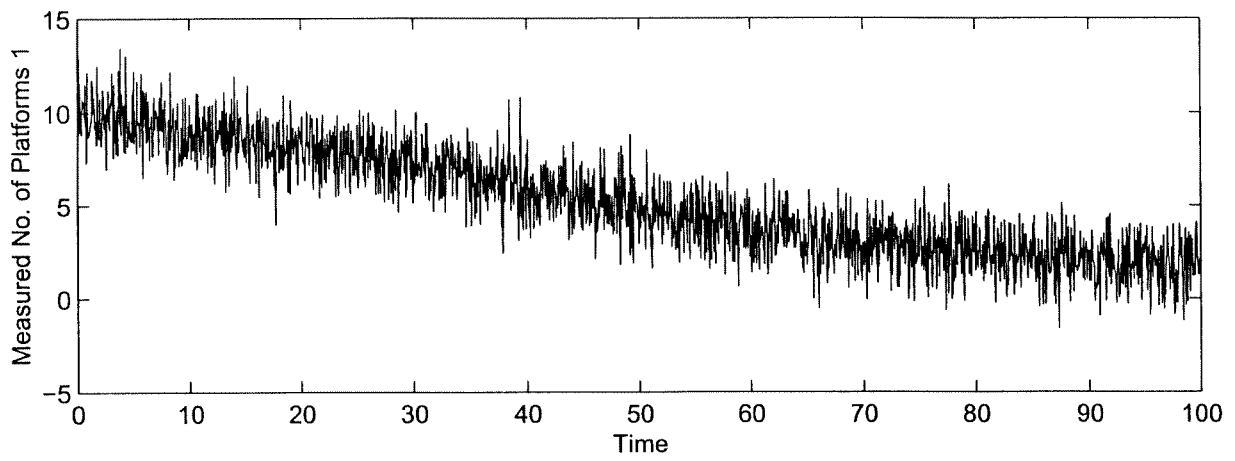


Figure 9.3: Noisy outputs. Noise-corrupted measured observations (top: number of platforms of the first Blue unit, bottom: number of platforms of the second Blue unit).

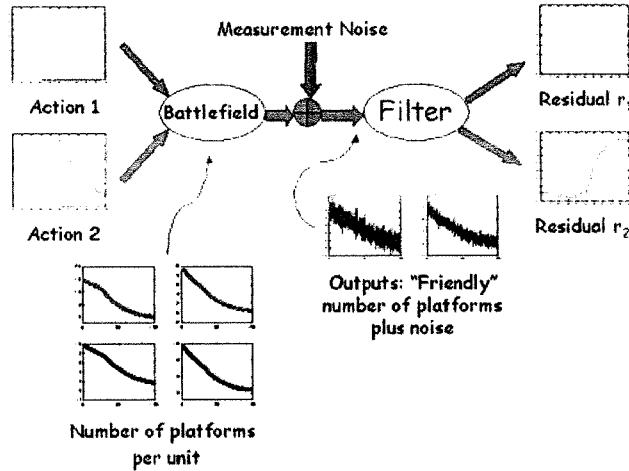


Figure 9.4: Block diagram describing the experiment.

The experimental data that we have obtained show the improved effectiveness of the game-theoretic detection filter with respect to the Kalman-like filter. This is demonstrated by the comparison of the two different time-behaviors of the performance signals generated by the two different filters, in response to the same measured observation corrupted by random noise. The filters are tested with respect to noise with different amounts of energy and, for each fixed amount of noise energy, a certain number of different noise signals are considered.

9.5 Example of experiment

In this experiment we consider a noise whose energy is equal to 15% of the energy of the noise-free output. This amount of noise is already enough to hide, in the observation of the number of platforms of the two friendly units (see Figure 9.3 for a picture of the two noisy observations), any sign of the occurrence of either one of the two engagement actions from the enemy forces. Nevertheless, when the noisy measurements are processed by a detection filter, a good deal of information about the engagement actions which have occurred can be derived.

Figure 9.5 depicts the responses to *Action 1* of the game-theoretic filter (top) and of the Kalman-like filter (bottom). The comparison of the two plots already demonstrates a much better performance of the game-theoretic filter as opposite to that of the Kalman-like filter. Note also that in neither of the two cases the output of the filter is affected by the occurrence of *Action 2*, as it should be. Figure 9.6 depicts the responses to *Action 2* of the game-theoretic filter (top) and of the Kalman-like filter (bottom). The comparison of these latter plots demonstrates more dramatically the efficiency of the game-theoretic filter. As a matter of fact, the presence of noise on the observations is sufficient to completely hide the occurrence of *Action 2* in the output of the Kalman-like filter, while the output of the game-theoretic filter still reveals this occurrence with a good deal of confidence. For the sake of completeness, in Figure 9.7 we show also the time histories of the two state variables of the two filters.

A special feature of the detection process that deserves to be stressed is the complete “independence” or “non-interaction” of the two performance signals. The first one increases its response upon the oc-

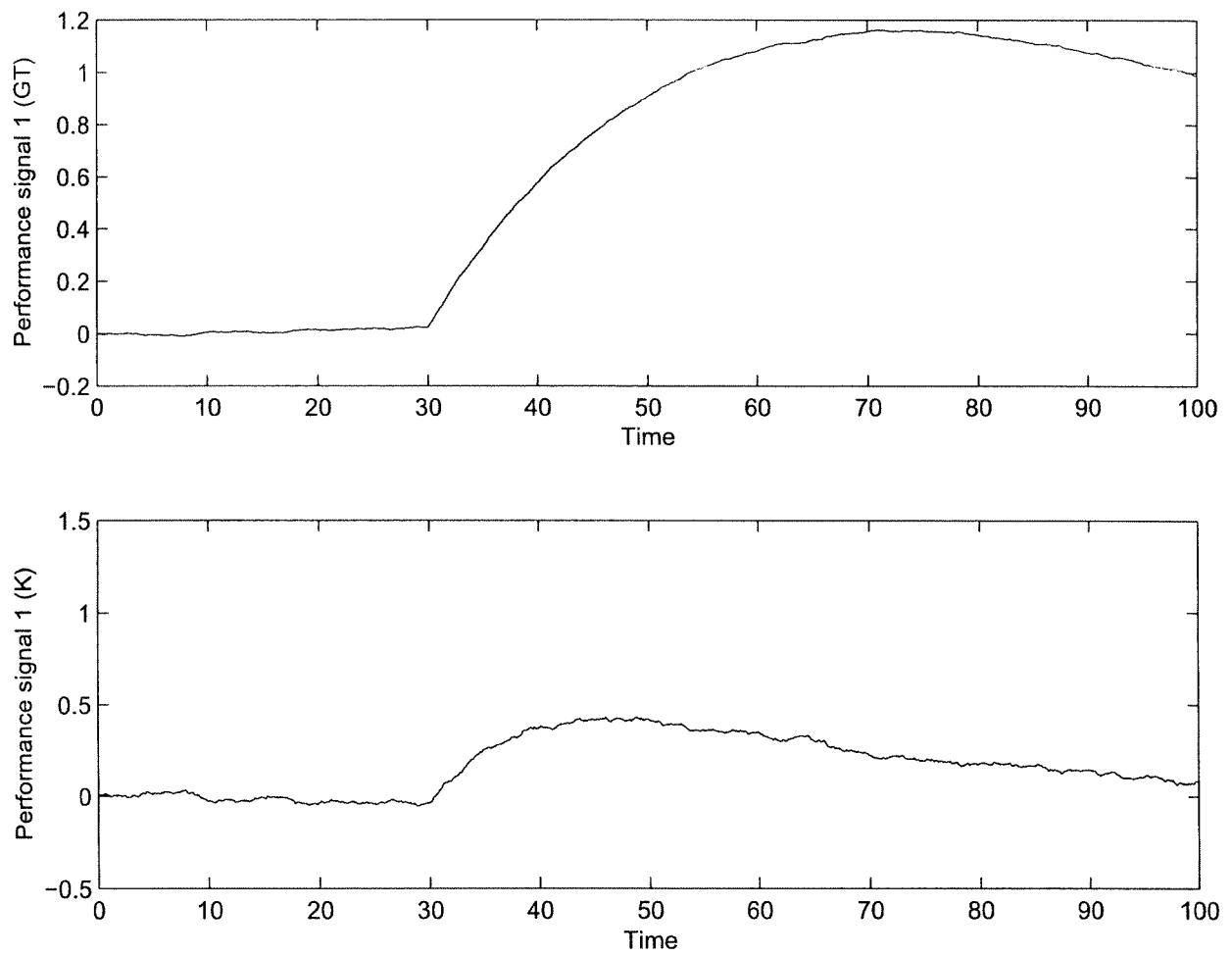


Figure 9.5: Response to Action 1 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 15% of the energy of the noise-free output.

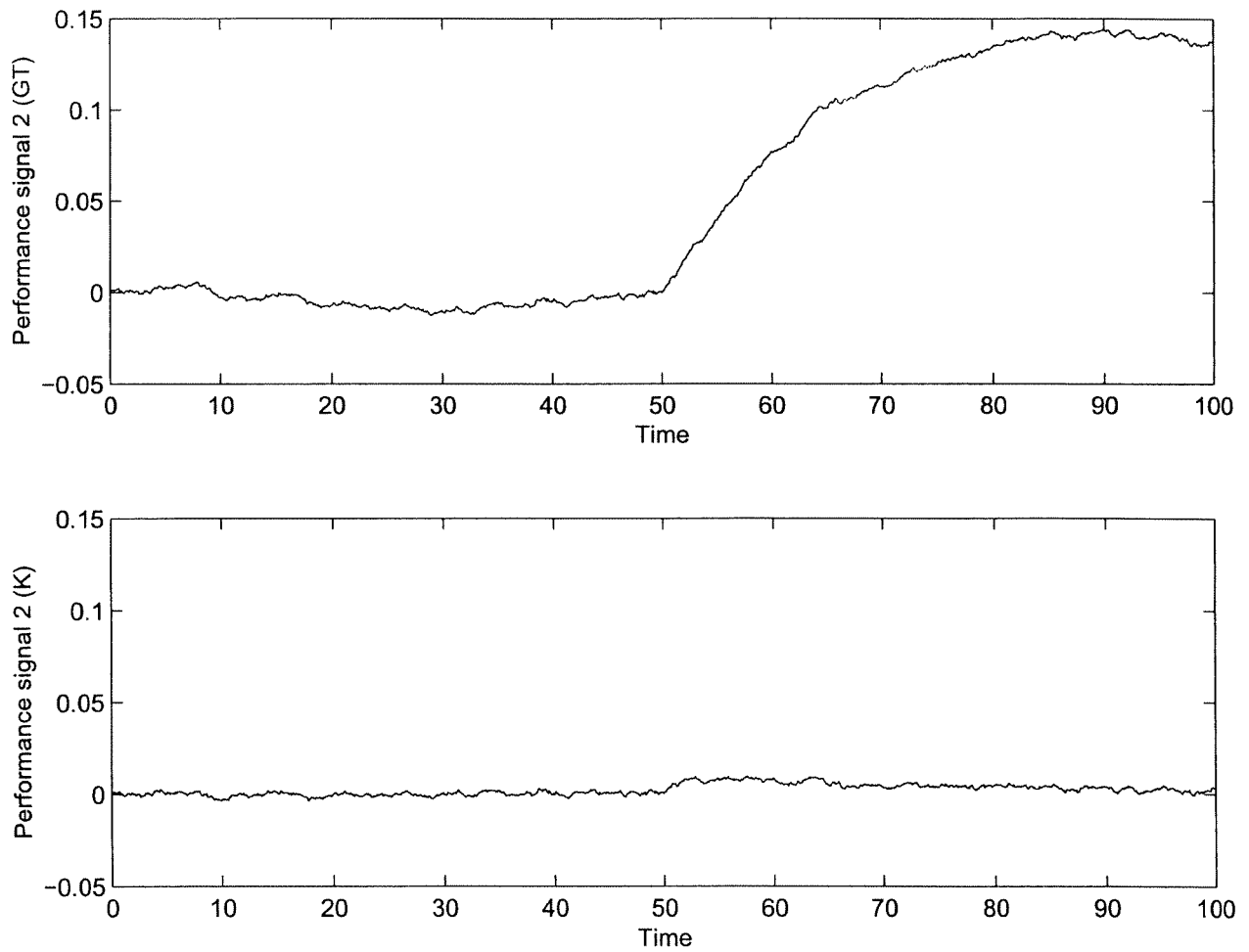


Figure 9.6: Response to Action 2 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 15% of the energy of the noise-free output.

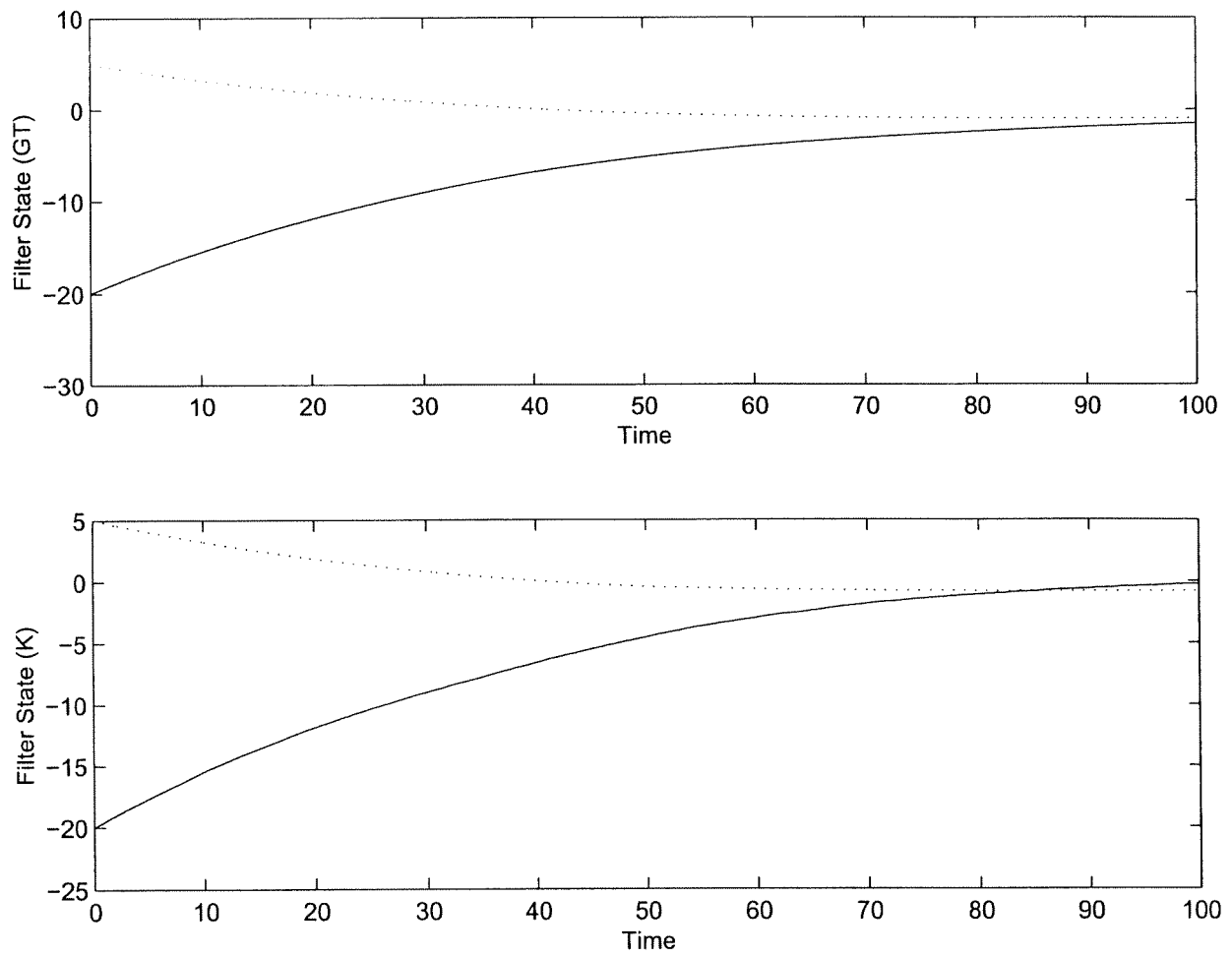


Figure 9.7: State evolution of the game-theoretic (top) and Kalman-like filter (bottom).

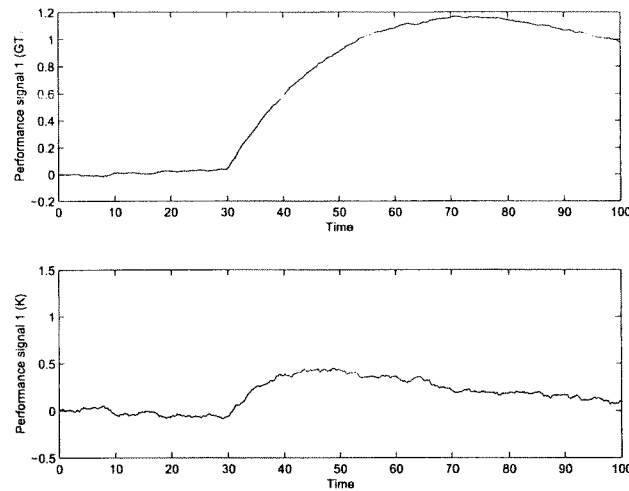


Figure 9.8: Response to Action 1 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 25% of the energy of the noise-free output.

currence of the first action but it is not affected by the occurrence of the second action. The second performance signal remains close to zero when the first action takes place, while it becomes patently nonzero as the second action occurs. This “non-interaction” property is at the basis of the isolation process (that is, to distinguish which action is occurring and when) and is an outcome of the geometric approach to the problem of detection and isolation.

Finally, it should be stressed that the difference between the two performance signals depends not only on the fact that they are responses to different input signals but also on the choice of weights in the cost function which determines the parameters of the filters, which have been differently chosen in the two cases.

9.6 Results of the Experiments

Our experiments are aimed also at testing the behavior of the detection filters to measurement noises of progressively increasing energy. The response of the game-theoretic and Kalman-like filters when the energy of the noise is equal to 15% of the energy of the noise-free measurement has been reported in the previous section (cf. Figures 9.5 and 9.6). Here we consider the responses to the same noise (the seed is the same as before) but with an increased level of energy. In particular, we consider the cases of noise whose energy is equal to 25%, 35%, 55%, 80% and 110% of the energy of the noise-free measurement, and plot the responses of the two filters in Figures 9.8-9.9, 9.10-9.11, 9.12-9.13, 9.14-9.15 and 9.16-9.17, respectively.

In our experiments, we have also tested the behavior of the detection filter under different kinds of noise (randomly assigned seed) and different energy levels of the noise. In particular, for each level of noise (15%, 25%, 35% of the energy of the noise-free measurement signal), we have run three experiments for the game-theoretic filter and three experiments for the Kalman-like filter, each one with a different noise signal. The outcome of these experiments is depicted, for the different levels of noise, in Figures 9.18-9.19, Figures 9.20-9.21 and Figures 9.22-9.23, respectively.

9.7 Conclusions and Recommendations

The game-theoretic detection and isolation filter shows a very good performance under different situation of noise corrupting the measurements. This is particularly true when compared to the behavior of the

Kalman-like detection filter. The main reason for this improvement is in the selective attenuation of the noise accomplished by the game-theoretic filter, which does not reduce the effect of the signal to detect. This selective attenuation of the noise increases the diagnostic capability of the game-theoretic filter even in the presence of high levels of noise energy.

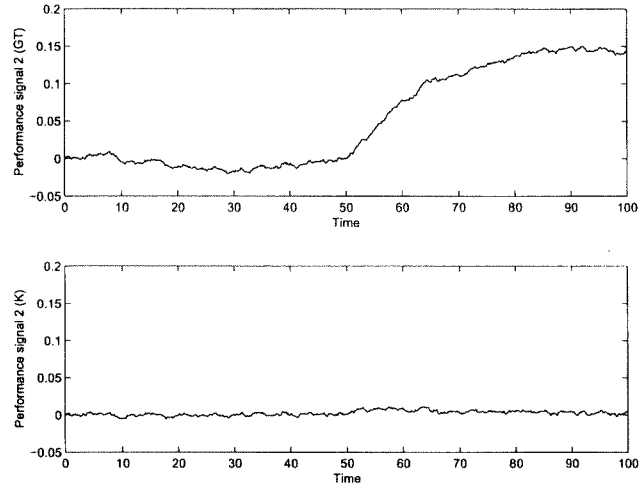


Figure 9.9: Response to Action 2 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 25% of the energy of the noise-free output.

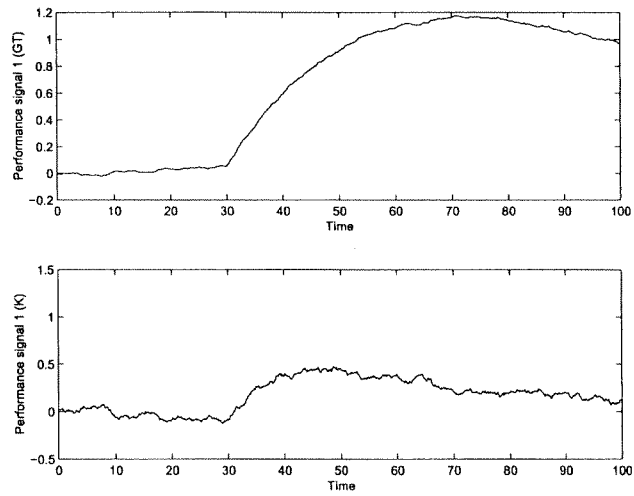


Figure 9.10: Response to Action 1 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 35% of the energy of the noise-free output.

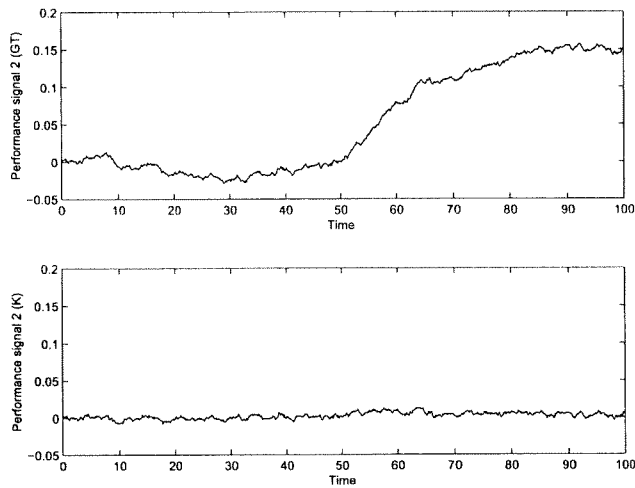


Figure 9.11: Response to Action 2 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 35% of the energy of the noise-free output.

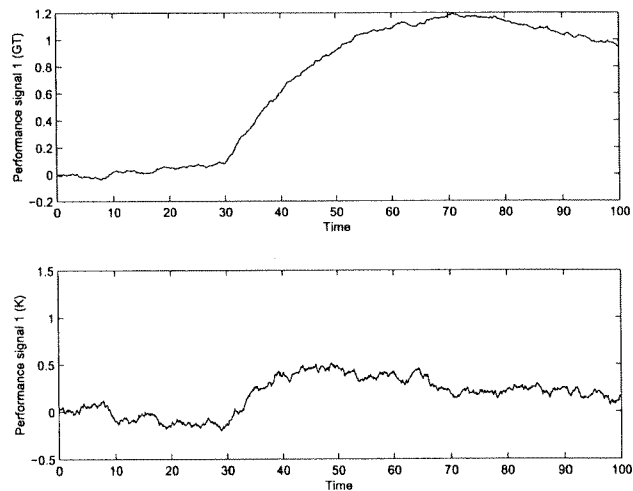


Figure 9.12: Response to Action 1 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 55% of the energy of the noise-free output.

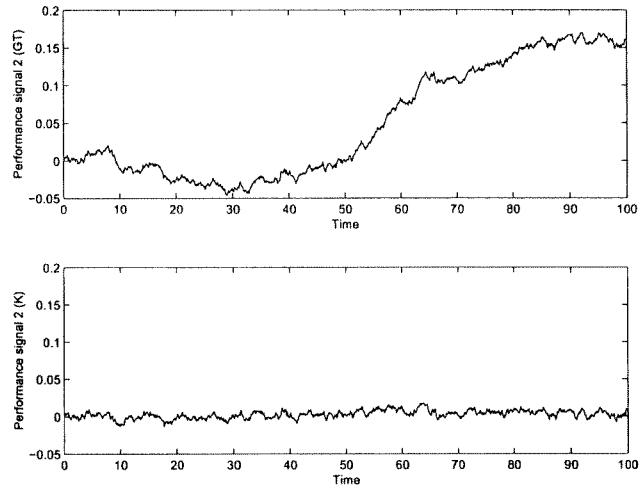


Figure 9.13: Response to Action 2 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 55% of the energy of the noise-free output.

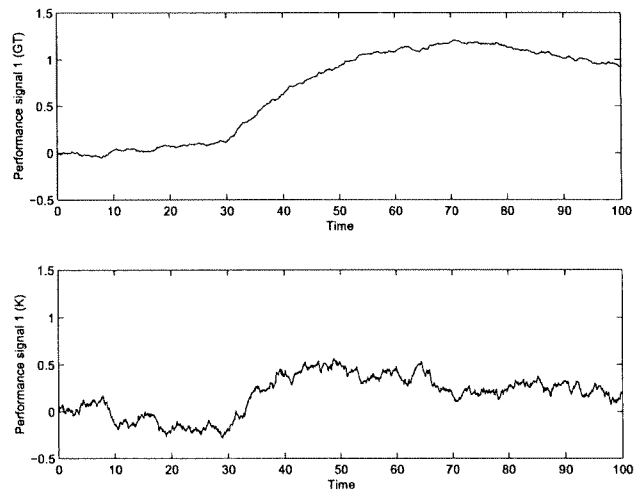


Figure 9.14: Response to Action 1 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 80% of the energy of the noise-free output.

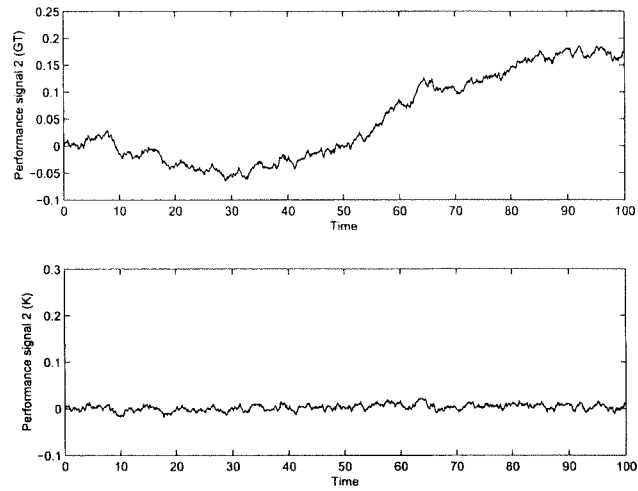


Figure 9.15: Response to Action 2 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 80% of the energy of the noise-free output.

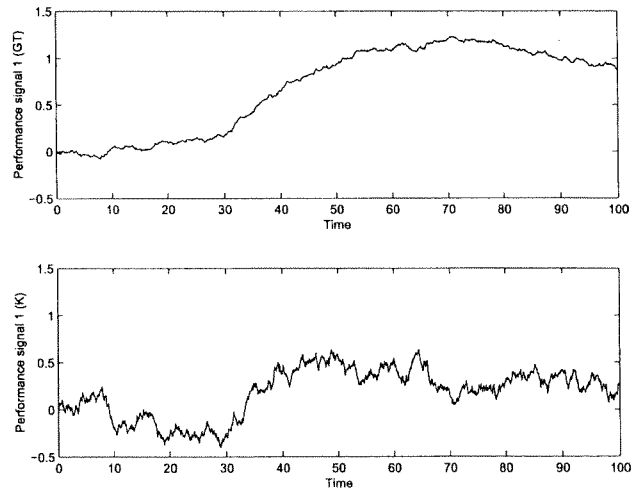


Figure 9.16: Response to Action 1 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 110% of the energy of the noise-free output.

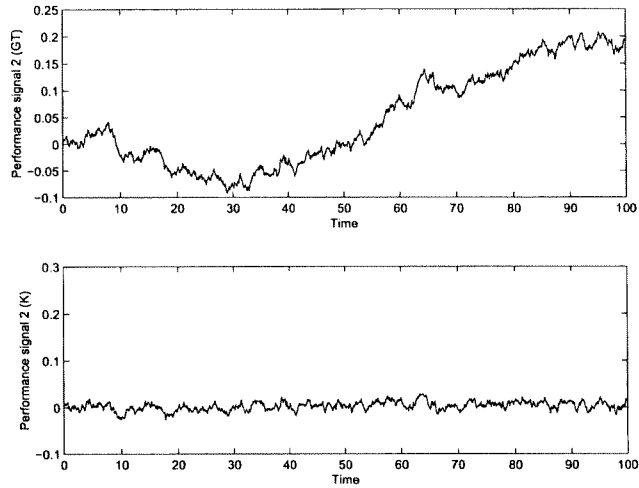


Figure 9.17: Response to Action 2 of the game-theoretic filter (top) and of the Kalman-like filter (bottom) under noise with energy equal to 110% of the energy of the noise-free output.

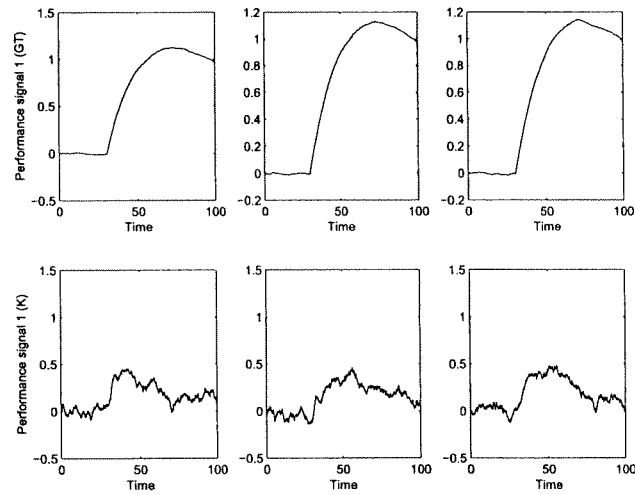


Figure 9.18: Responses of the game-theoretic (top) and Kalman-like filter (bottom) to Action 1. The six responses correspond to six different random choices of the seed of the noise signals. The energy of the noise signals is equal to 15% of the output signal energy.

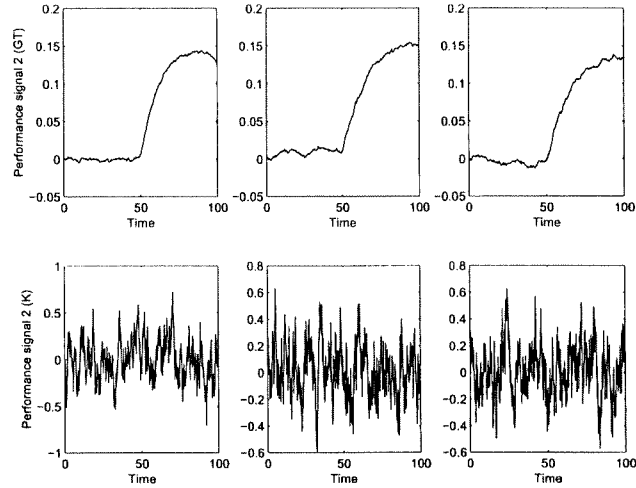


Figure 9.19: Responses of the game-theoretic (top) and Kalman-like filter (bottom) to Action 2. The six responses correspond to six different random choices of the seed of the noise signals. The energy of the noise signals is equal to 15% of the output signal energy.

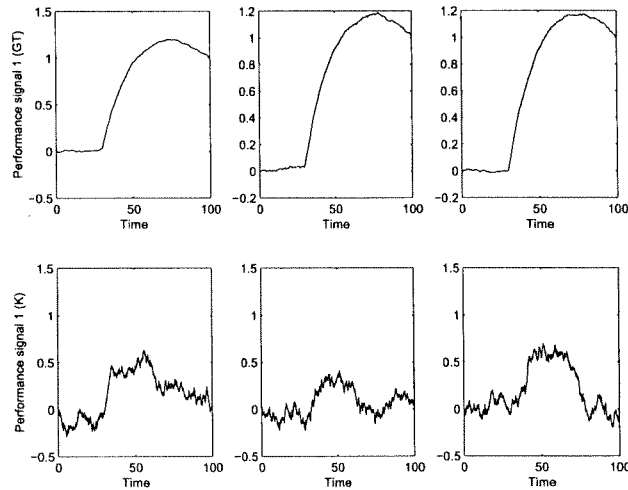


Figure 9.20: Responses of the game-theoretic (top) and Kalman-like filter (bottom) to Action 1. The six responses correspond to six different random choices of the seed of the noise signals. The energy of the noise signals is equal to 25% of the output signal energy.

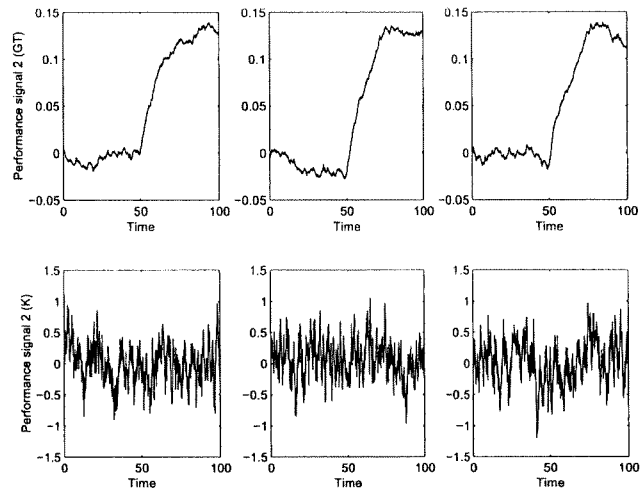


Figure 9.21: Responses of the game-theoretic (top) and Kalman-like filter (bottom) to Action 2. The six responses correspond to six different random choices of the seed of the noise signals. The energy of the noise signals is equal to 25% of the output signal energy.

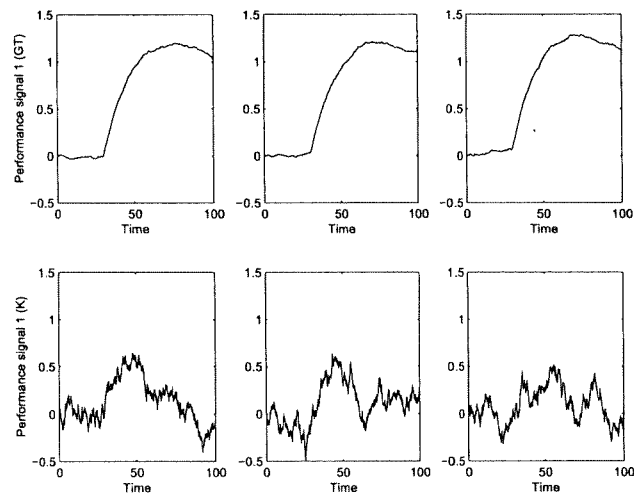


Figure 9.22: Responses of the game-theoretic (top) and Kalman-like filter (bottom) to Action 1. The six responses correspond to six different random choices of the seed of the noise signals. The energy of the noise signals is equal to 35% of the output signal energy.

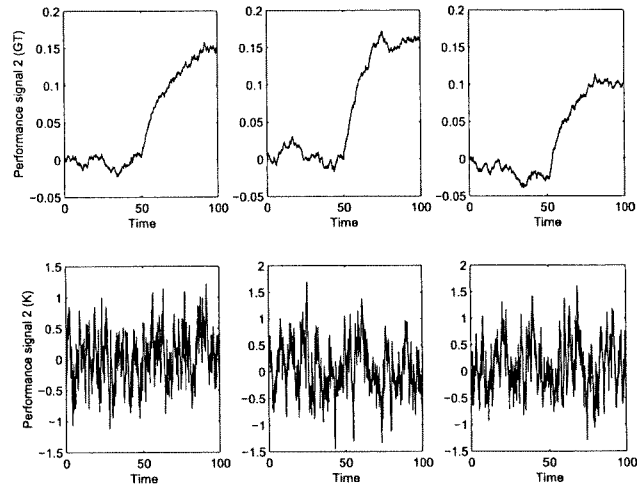


Figure 9.23: Responses of the game-theoretic (top) and Kalman-like filter (bottom) to Action 2. The six responses correspond to six different random choices of the seed of the noise signals. The energy of the noise signals is equal to 35% of the output signal energy.

Bibliography

- [1] Washington University, Mission Dynamics Continuous-Time Model, Version 2.42.
- [2] C. De Persis, A. Isidori, "On the observability codistribution of a nonlinear system," *Systems & Control Letters*, 40, 297-304, 2000.
- [3] C. De Persis, A. Isidori, A geometric approach to nonlinear fault detection and isolation, to appear as a regular paper on *IEEE Transactions on Automatic Control*, Sept. 2001.
- [4] C. De Persis, A. Isidori, "On the design of fault detection filters with game-theoretic-optimal sensitivity", submitted.

Chapter 10

Experiment 10: Detector Performance under Parameter Variations

10.1 Executive Summary

This Chapter describes the experiment results regarding the game-theoretic detection filter under parametric uncertainty. The exact values of the parameters in the mathematical model of the battlefield are not known, and only a nominal value is available. The filter, whose objective is to reveal the occurrence of an “engagement action” from enemy units, is designed on the basis of the nominal value. This set of experiments shows that the game-theoretic detection filter, although proven to be effective in the selective attenuation of measurement noise, is relatively sensitive to the uncertainty in the parameters.

10.2 Purpose of the Experiment

In [5], we tested the effectiveness of the game-theoretic filter in detecting enemy actions in the case in which the measurements coming from the theater of operations were affected by noise. The basic purpose of this second series of experiments is *to assess the robustness* of the “game-theoretic” detection filter (cf. [4]) in the presence of *uncertainty in the parameters* appearing in the mathematical model of the battlefield. Following [5], we summarize in the remainder of this section the problem of detection and isolation of multiple enemy actions in a battlefield.

The mathematical description of the battlefield used here is the one introduced in [1] and already adopted in [5]. We consider the case in which two opposing forces are present in the theater of operations, the Blue force (the “friends”) and the Red force (the “enemies”). Each force consists of two units and each unit consists of a number of platforms whose evolution in time is described by a first order nonlinear differential equation and depends on the “actions” which the opposing units are performing against the unit in question. If any “new” action is performed by any of the opposing units, this affects the evolution of the number of platforms of the other force’s units. Letting η_i^R and η_i^B , with $i = 1, 2$, denote the number of platforms of the i -th Red and – respectively – i -th Blue unit, the model in question is a

four-dimensional nonlinear system described by two pairs of equations of the form (cf. [1])

$$\begin{aligned} \frac{d}{dt}\eta_i^R(t) &= -\eta_i^R(t) \\ &\times \left[\alpha_i^R + \sum_{j=1}^2 \left\{ \left[1 - e^{-\sigma_j^B \left\{ \frac{\eta_j^B(t)\pi_{ji}^B(t)}{\eta_i^R(t)} \right\}} \right] \beta_{ej_i}^B \left[1 - \{1 - P_{0ji}^B \varphi_{ji}^B(|\tilde{\eta}_i^R(t) - \tilde{\eta}_j^B(t)|)\} \beta_{kji}^B \right]^{n_{ji}^B} \right\} \right], \\ \frac{d}{dt}\eta_i^B(t) &= -\eta_i^B(t) \\ &\times \left[\alpha_i^B + \sum_{j=1}^2 \left\{ \left[1 - e^{-\sigma_j^R \left\{ \frac{\eta_j^R(t)\pi_{ji}^R(t)}{\eta_i^B(t)} \right\}} \right] \beta_{ej_i}^R \left[1 - \{1 - P_{0ji}^R \varphi_{ji}^R(|\tilde{\eta}_i^B(t) - \tilde{\eta}_j^R(t)|)\} \beta_{kji}^R \right]^{n_{ji}^R} \right\} \right]. \end{aligned}$$

In these equations, $\pi_{ji}^R(\cdot)$ and $\pi_{ji}^B(\cdot)$ are (independent) input variables representing the “level of engagement” of the j -th Red unit with the i -th Blue unit and – respectively – of the j -th Blue unit with the i -th Red unit. For convenience, we suppose in all our experiments that

$$\pi_{11}^R(t) = \pi_{12}^R(t) =: \pi_1^R(t), \quad \pi_{21}^R(t) = \pi_{22}^R(t) =: \pi_2^R(t).$$

This means, in the terminology of [1], page 2, that we allow the “unique target constraint” to be violated (for the Red units only).

The basic problem addressed in our series of experiments on the design of filters for the detection of enemy actions is the following one: *we monitor only the number of platforms in the two Blue units* (i.e. we measure only the values of the two state variables η_1^B, η_2^B) and *we want to detect the occurrence of an “engagement action” from either one of the two Red units* (i.e. we want to detect when either one of the two input signals $\pi_i^R(\cdot)$ has become nonzero). Implicit in this is the assumption that the two other state variables η_1^R, η_2^R (number of platforms of the two enemy units) as well as all the input variables $\pi_{ij}^B, i, j = 1, 2$, and $\pi_i^R, i = 1, 2$ are not monitored. The purpose of the detection process is precisely the determination of when either π_1^R or π_2^R has become nonzero, without having it directly measured.

10.3 Hypothesis to Prove or Disprove

The experiments of this second series aim to test the performance of the game-theoretic detection filter in the presence of parametric uncertainties and are implemented considering, as in [5], a *bilinear approximation* of (10.1) instead of the full nonlinear model.

This bilinear model has the form (see [1], equation (29))

$$\begin{aligned} \frac{d}{dt}\eta_1^R(t) &= -\alpha_1^R \eta_1^R(t) - \gamma_{11}^R \eta_1^B(t) \pi_{11}^R(t) - \gamma_{12}^R \eta_2^B(t) \pi_{21}^R(t), \\ \frac{d}{dt}\eta_2^R(t) &= -\alpha_2^R \eta_2^R(t) - \gamma_{21}^R \eta_1^B(t) \pi_{12}^R(t) - \gamma_{22}^R \eta_2^B(t) \pi_{22}^R(t), \\ \frac{d}{dt}\eta_1^B(t) &= -\alpha_1^B \eta_1^B(t) - \gamma_{11}^B \eta_1^R(t) \pi_{11}^R(t) - \gamma_{12}^B \eta_2^R(t) \pi_{21}^R(t), \\ \frac{d}{dt}\eta_2^B(t) &= -\alpha_2^B \eta_2^B(t) - \gamma_{21}^B \eta_1^R(t) \pi_{12}^R(t) - \gamma_{22}^B \eta_2^R(t) \pi_{22}^R(t), \end{aligned} \tag{10.1}$$

in which the (actual) values of the parameters appearing in the equations corresponding to the Blue units, namely parameters $\alpha_1^B, \alpha_2^B, \gamma_{11}^B, \gamma_{12}^B, \gamma_{21}^B, \gamma_{22}^B$, are *not* perfectly known and may differ from their nominal values $\bar{\alpha}_1^B, \bar{\alpha}_2^B, \bar{\gamma}_{11}^B, \bar{\gamma}_{12}^B, \bar{\gamma}_{21}^B, \bar{\gamma}_{22}^B$, which are assumed to be *known*. The other parameters $\alpha_1^R, \alpha_2^R, \gamma_{11}^R, \gamma_{12}^R, \gamma_{21}^R, \gamma_{22}^R$ of the model (10.1) *do not affect* the design and the response of the detection filter and consequently their uncertainty does not need to be considered in this series of experiments.

The structure of the filter that we use to solve the detection problem described in the previous section is chosen according to a general (differential-geometric) methodology developed in our papers [2], [3].

This filter *receives as inputs* the two observed variables η_1^B, η_2^B and *generates as outputs* two signals r_1, r_2 , called *performance signals* (typically known also with the name of *residuals*), in such a way that $r_i(t)$ is zero if the Red unit i is not engaged with the Blue units at time t (i.e. if $\pi_i^R(t) = 0$), and that $r_i(t)$ is nonzero if the Red unit i is engaged with the Blue units (i.e. if $\pi_i^R(t) \neq 0$). Specifically, this filter is modeled by equations of the form

$$\begin{aligned}\dot{\tilde{\eta}}_1(t) &= -\bar{\alpha}^B \tilde{\eta}_1(t) + g_1(\eta_1^B(t) - \frac{\tilde{\gamma}_{12}^R}{\tilde{\gamma}_{22}^R} \eta_2^B(t) - \tilde{\eta}_1(t)), \\ \dot{\tilde{\eta}}_2(t) &= -\bar{\alpha}^B \tilde{\eta}_2(t) + g_2(\eta_2^B(t) - \frac{\tilde{\gamma}_{21}^R}{\tilde{\gamma}_{11}^R} \eta_1^B(t) - \tilde{\eta}_2(t)), \\ r_1(t) &= \eta_1^B(t) - \frac{\tilde{\gamma}_{12}^R}{\tilde{\gamma}_{22}^R} \eta_2^B(t) - \tilde{\eta}_1(t), \\ r_2(t) &= \eta_2^B(t) - \frac{\tilde{\gamma}_{21}^R}{\tilde{\gamma}_{11}^R} \eta_1^B(t) - \tilde{\eta}_2(t),\end{aligned}\tag{10.2}$$

in which $\bar{\alpha}^B = \bar{\alpha}_1^B = \bar{\alpha}_2^B$ and the “gain parameters” g_1 and g_2 are to be determined in some “optimal” way.

The basic issue that makes the design of the filter (i.e. of the gain parameters g_1 and g_2) *critical* is the presence of *measurement noises* on the two monitored variables η_1^B, η_2^B (the two inputs to the filter (10.2)). The experiments reported in [5] demonstrate that the performance of a detection filter in which the gain parameters g_1, g_2 are chosen as the result of a “game-theoretic” design is substantially more effective than a detection filter in which the gain parameters are designed through standard methods for state estimation from noisy observations (such as those used in the classical Kalman filter). In this new series of experiments, we want to test how the uncertainty in the parameters of the plant (10.1) affects the behavior of the detection filter in which g_1, g_2 derive from the “game-theoretic” design.

10.4 Experiment setup

This series of experiments of detection of enemy actions is designed in the following way. The battlefield is modeled as in equation (10.1), whose four states represent the number of platforms of each of the four units involved in the battle. The (actual) value of each parameter in (10.1) is unknown. The inputs variables, representing the level of engagement of the battling units, are fixed functions of time. In particular, the two levels of engagement of the Red units 1 and 2 versus the Blue units vary with time as shown in Figure 10.1, where “Action 1” represents the level of engagement of Red unit 1 and “Action 2” represents the level of engagement of Red unit 2. Note that the first action occurs at $t = 30$ units of time, whereas the second action takes place at $t = 50$ units of time and this while the first action is still occurring. The corresponding behavior in time of the number of Red and Blue platforms in each unit is plotted in Figure 10.2. Finally, Figure 10.3 depicts the outcome of the two observations, namely the evolution in time of the number of platforms in the two Blue units, *corrupted by measurement noise* (compare with the two bottom graphs in Figure 10.2, depicting the same quantities without measurement noise).

Figure 10.4 shows a block diagram describing the experiment. The two (noise-corrupted) measured observations are fed into the filter: the Output 1 of this filter is expected to reveal the occurrence of Action 1, while the Output 2 is expected to reveal the occurrence of Action 2. The detection filter considered for the experiments is the one described in (10.2) in which each gain parameter $g_i, i = 1, 2$, is chosen in such a way as to render

$$\sup_{v_1, v_2} \inf_{\pi_i^R} \int_0^{t_1} [|r_i|_Q^2 + |\pi_i^R|_{N-1} - \gamma^2(|v_1|_{V-1}^2 + |v_2|_{M-1}^2)] dt \leq 0, \text{ for } i = 1, 2,\tag{10.3}$$

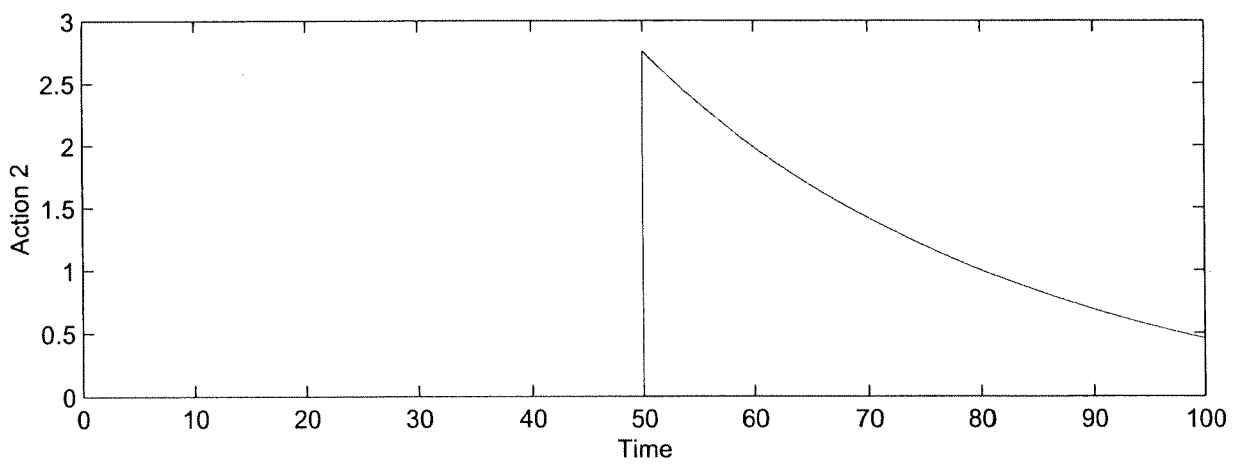
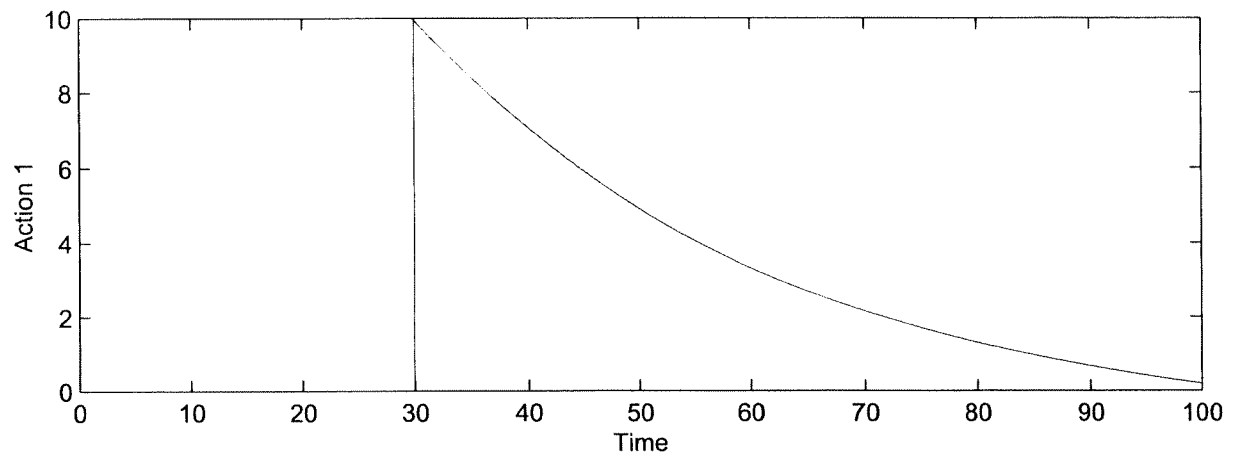


Figure 10.1: Engagement enemy actions (to be detected).

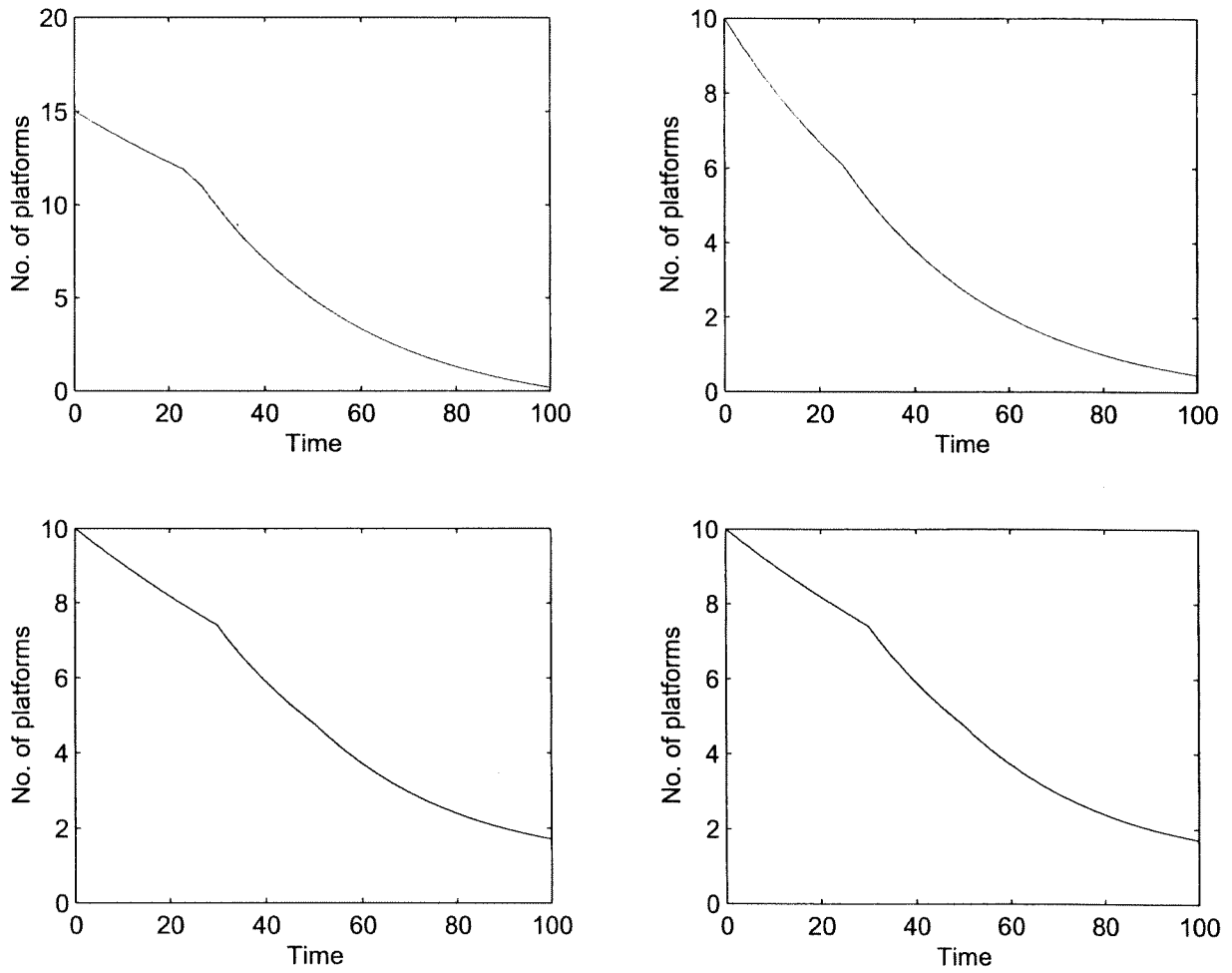


Figure 10.2: Time history of the number of platforms of the four units. The two top plots represent the number of platforms of the Red units, the two bottom plots represent the number of platforms of the Blue units.

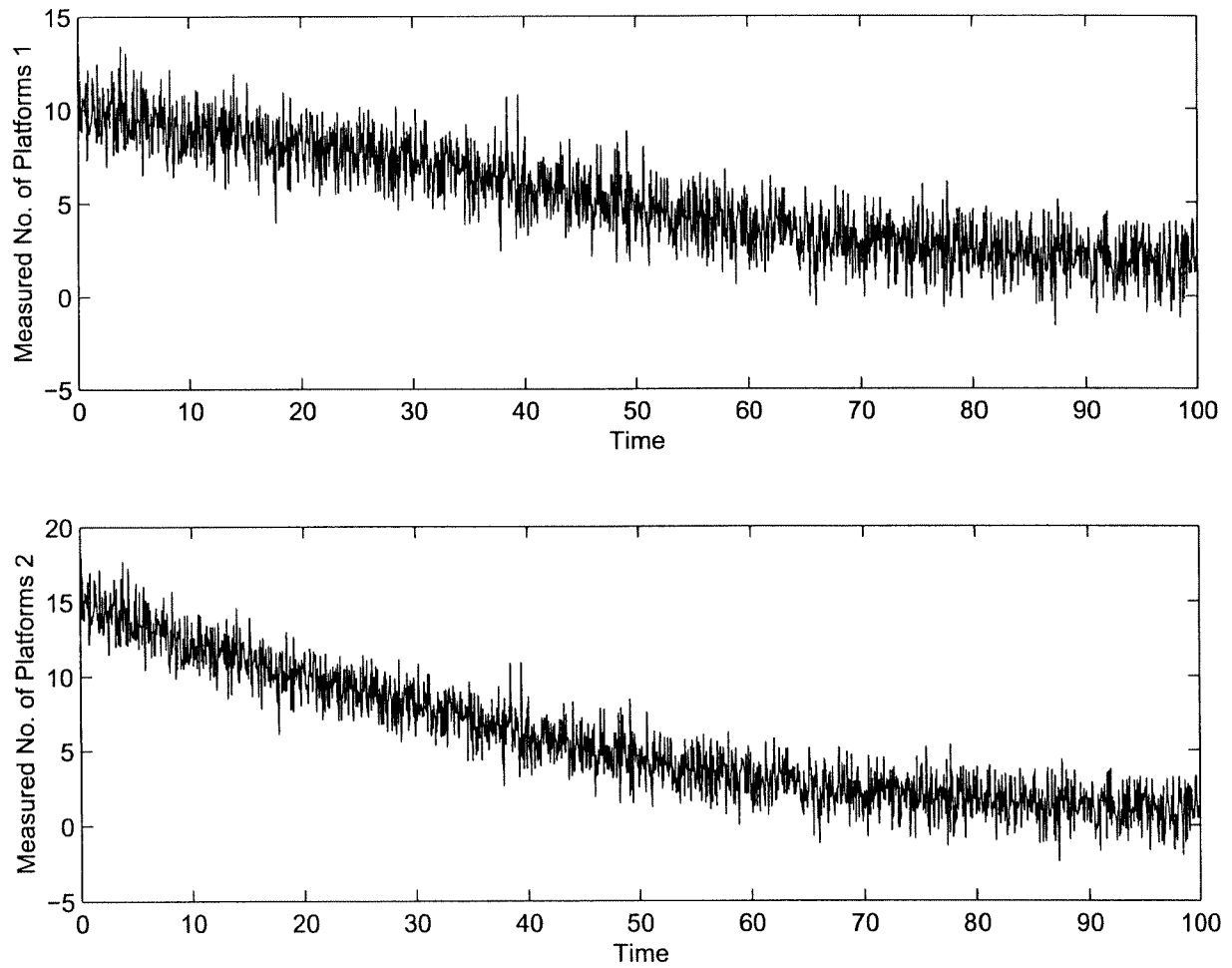


Figure 10.3: Noisy outputs. Noise-corrupted measured observations (top: number of platforms of the first Blue unit; bottom: number of platforms of the second Blue unit).

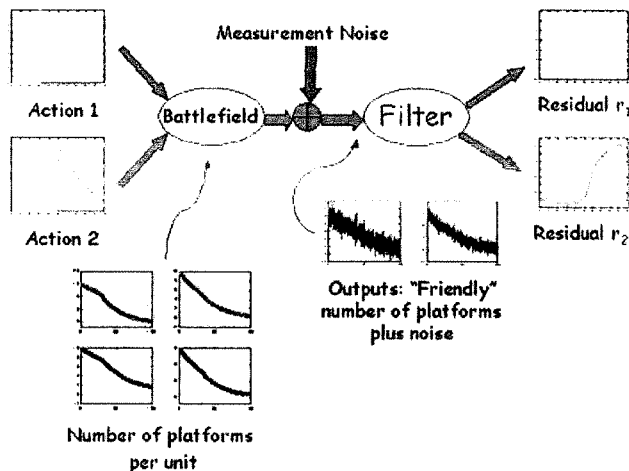


Figure 10.4: Block diagram describing the experiment.

where $[0, t_1]$ is the interval of time over which the experiment is performed, r_i , $i = 1, 2$ are the variables as defined in (10.2), v_i , for $i = 1, 2$, are the noise signals affecting the measurements of η_i^B , π_i^R , $i = 1, 2$, are the engagement enemy actions (which must be detected), Q, M, V, N are suitable weighting matrices, and γ is a threshold parameter representing the desired attenuation of the noise on the residual.

The experiments are performed by letting each of the parameters $\alpha_1^B, \alpha_2^B, \gamma_{11}^B, \gamma_{12}^B, \gamma_{21}^B, \gamma_{22}^B$ in (10.1) to differ – by increasing quantities – from their nominal values $\bar{\alpha}^B, \bar{\alpha}^B, \bar{\gamma}_{11}^B, \bar{\gamma}_{12}^B, \bar{\gamma}_{21}^B, \bar{\gamma}_{22}^B$, which are used in design of the filter (10.2).

10.5 Example of experiment

We illustrate in what follows an example of experiment in which the actual value of the parameter α_1^B is equal to, respectively, 110%, 120%, 130%, 140% and 150% of its nominal value $\bar{\alpha}^B$. The energy of the noise corrupting the measurements of η_i^B , $i = 1, 2$, is equal to 15% of the energy of η_i^B , $i = 1, 2$. Figures 10.5 and 10.6 depict the responses to *Action 1* and, respectively, *Action 2* of the game-theoretic filter.

Figure 10.5 shows how the response to Action 1 tends to evolve away from zero even though Action 1 is not taking place (first 30 units of time). However, the first performance signal is still “sensitive” to the occurrence of Action 1, as it is demonstrated by the “bump” at time $t = 30$ units of time (when Action 1 occurs). In Figure 10.6, a similar outcome is reported. Indeed, the figure shows a nonzero performance signal in absence of Action 2 and a “bump” at time $t = 50$ units of time (when Action 2 occurs). The “bump” becomes less evident as the amount of uncertainty increases, causing the detection of Action 2 to be more difficult. We note, however, that the uncertainty on α_1^B does not influence the “non-interaction” property of the two performance signals. As a matter of fact, the first performance signal is not “sensitive” to Action 2 and the second performance signal is not “sensitive” to Action 1.

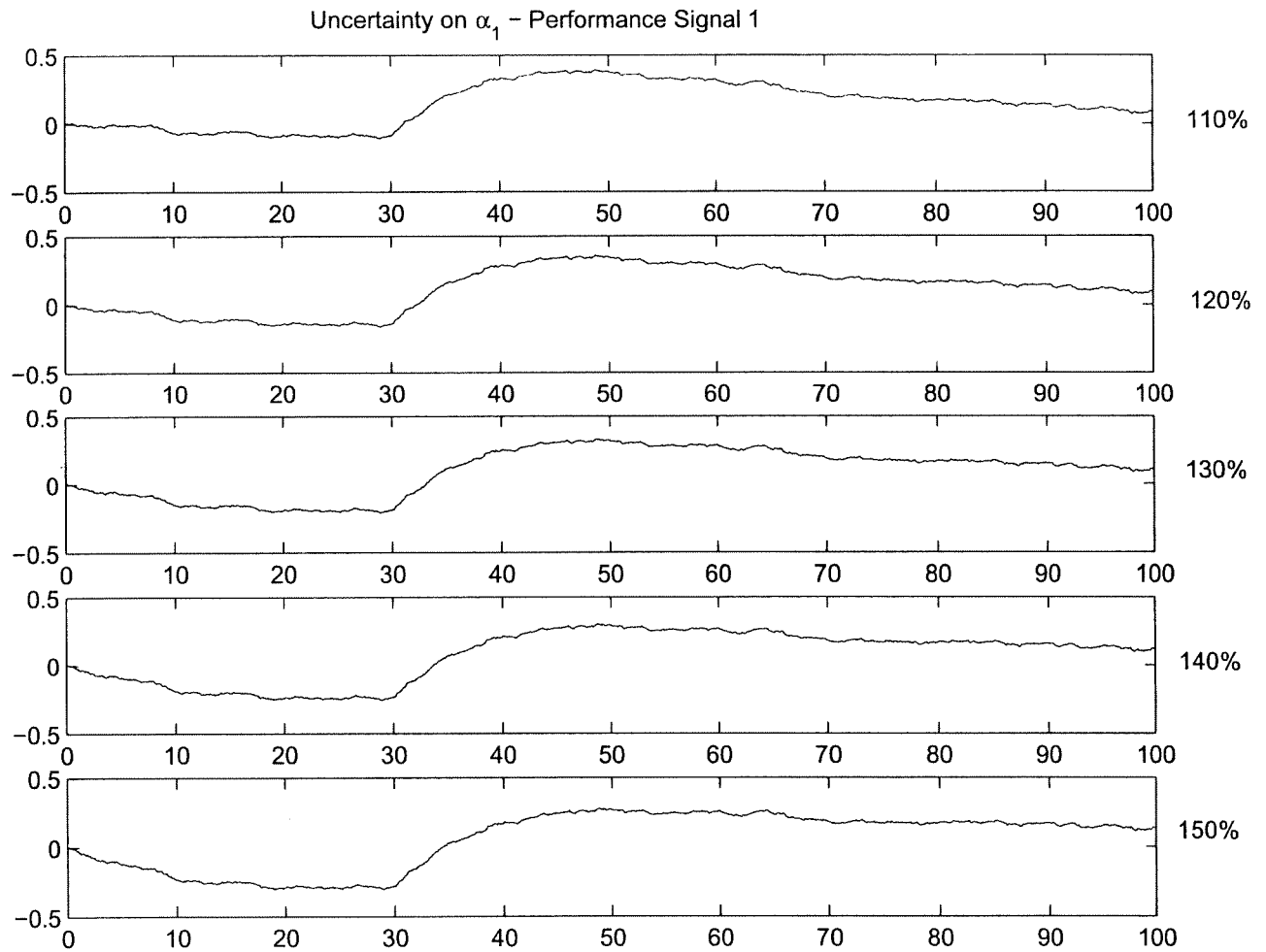


Figure 10.5: Response to Action 1 of the detection filter in the presence of uncertainty in the parameter α_1 . The actual value of α_1 varies from 110% (top graph) to 150% (bottom graph) of the nominal value.

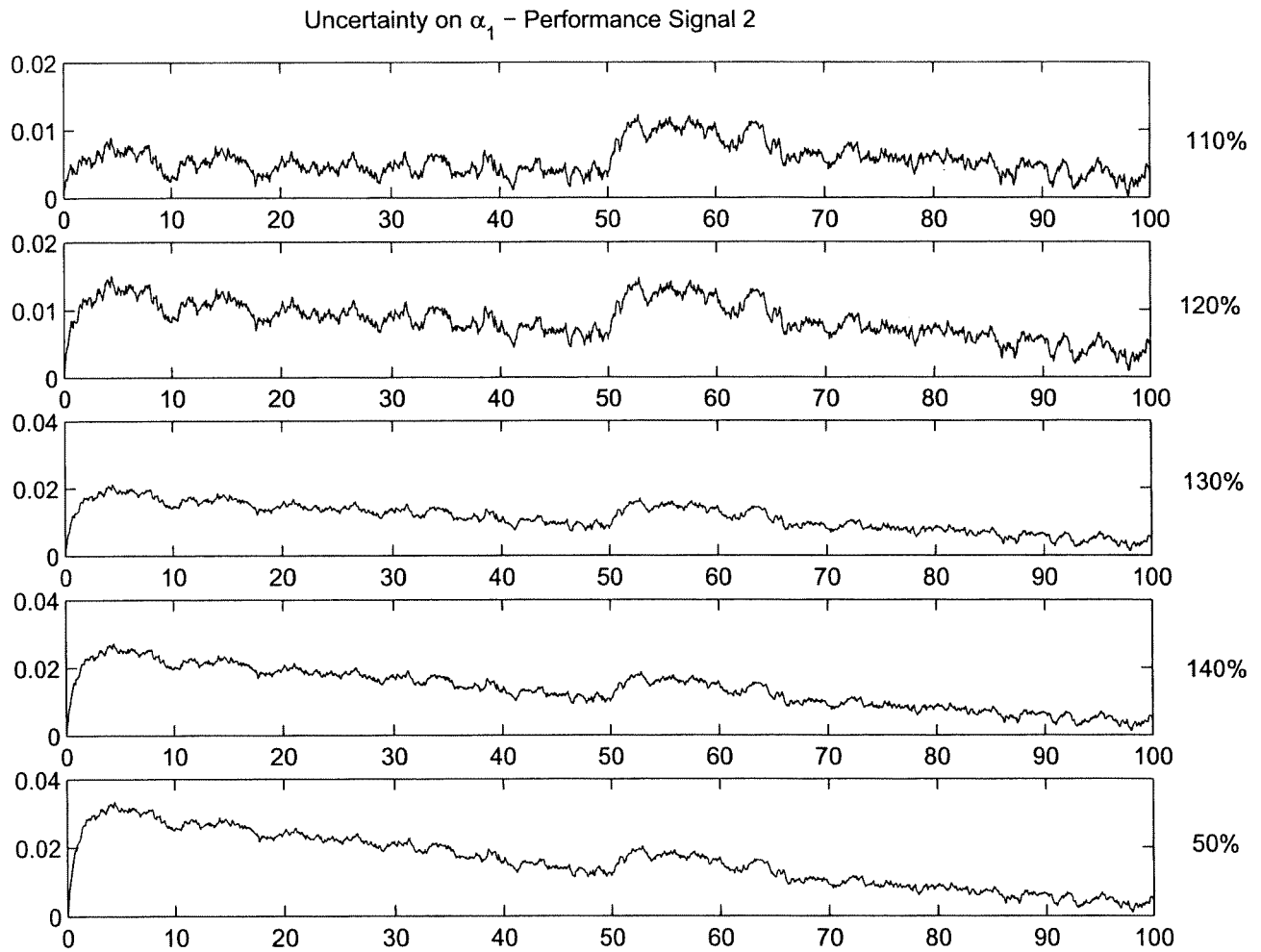


Figure 10.6: Response to Action 2 of the detection filter in the presence of uncertainty in the parameter α_1 . The actual value of α_1 varies from 110% (top graph) to 150% (bottom graph) of the nominal value.

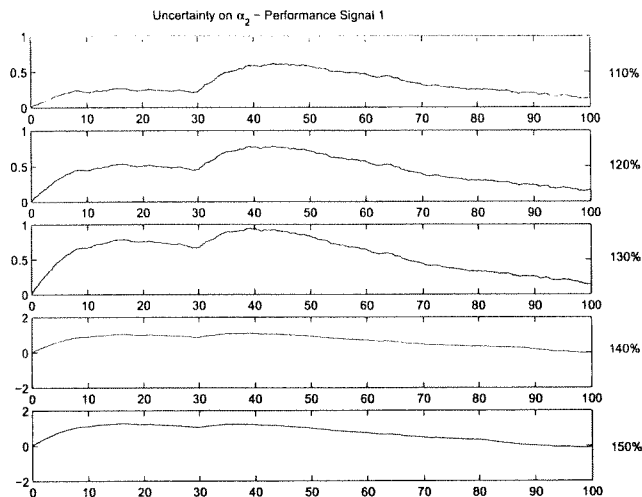


Figure 10.7: Response to Action 1 of the detection filter in the presence of uncertainty in the parameter α_2 . The actual value of α_2 varies from 110% (top graph) to 150% (bottom graph) of the nominal value.

10.6 Results of the experiments

In the remaining experiments of this series, we test the influence of the uncertainty in the parameters $\alpha_2^B, \gamma_{11}^B, \gamma_{12}^B, \gamma_{21}^B, \gamma_{22}^B$. The responses of the filter are given in Figures 10.7 to 10.16. The actual value of each parameter is taken equal to, respectively, 110%, 120%, 130%, 140% and 150% of the corresponding nominal value. We set the value of each parameter in the model equal to one of the actual values and perform a simulation. The noise on the measurements is as specified in the previous section.

From the analysis of the figures, it is possible to observe that three negative phenomena emerge due to parametric uncertainty. The first one is that the performance signals tends to evolve away from zero even when no event is occurring (see Figures 10.7 and 10.8). The second phenomenon is the loss of the “non-interaction” property of the two performance signals. For instance, the second performance signal tends to react to the occurrence of Action 1 (see Figure 10.10 at time $t = 30$ units of time), which may lead to infer erroneously the occurrence of Action 2 when Action 1 is actually taking place. The third phenomenon is a reduction in the capability of the filter to “emphasize” the event signal, as it is evident from the decreased size of the “bumps” in all the performance signals.

10.7 Conclusions and Recommendations

The game-theoretic detection and isolation filter, although proven to be effective in the selective attenuation of measurement noise, has been shown to be relatively sensitive to uncertainty in the parameters of the model which describes the battlefield. In particular, a progressive degradation of the effectiveness of the detection filter has been observed as the difference between actual and nominal values of the parameters increases.

CONCLUSIONS – HYPOTHESIS 10
The data obtained from this set of experiments show a progressive degradation of the filter performance as the difference between actual and nominal values of the parameters increases.

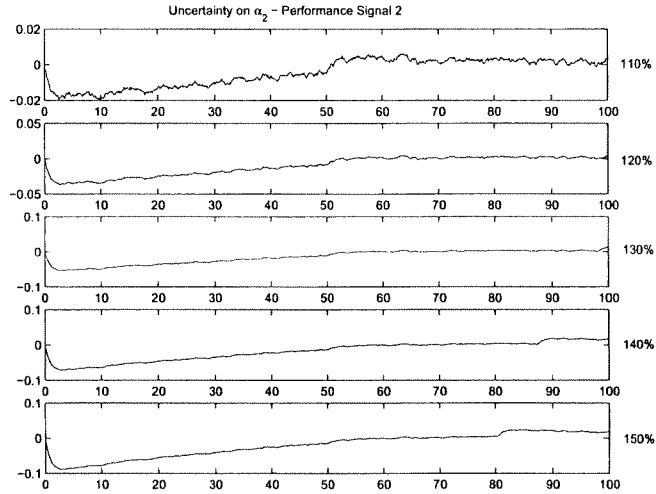


Figure 10.8: Response to Action 2 of the detection filter in the presence of uncertainty in the parameter α_2 . The actual value of α_2 varies from 110% (top graph) to 150% (bottom graph) of the nominal value.

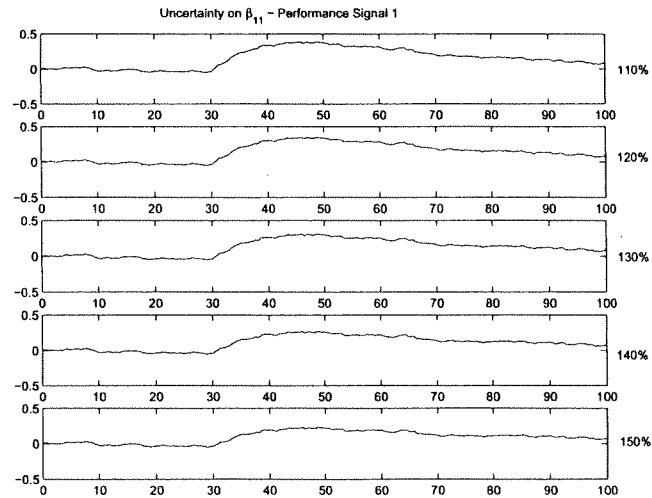


Figure 10.9: Response to Action 1 of the detection filter in the presence of uncertainty in the parameter β_{11} . The actual value of β_{11} varies from 110% (top graph) to 150% (bottom graph) of the nominal value.

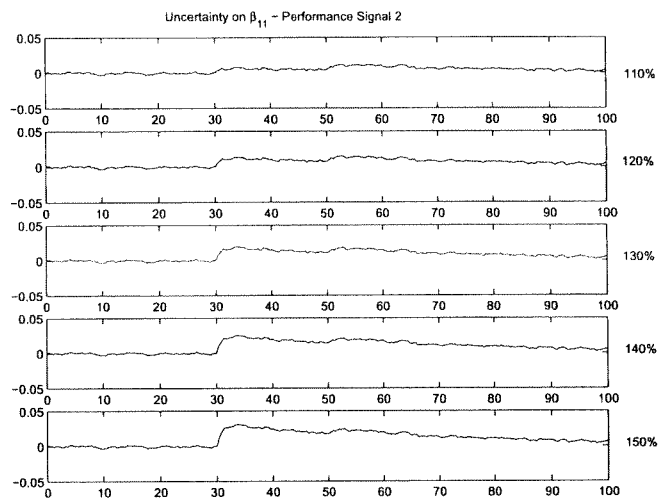


Figure 10.10: Response to Action 2 of the detection filter in the presence of uncertainty in the parameter β_{11} . The actual value of β_{11} varies from 110% (top graph) to 150% (bottom graph) of the nominal value.

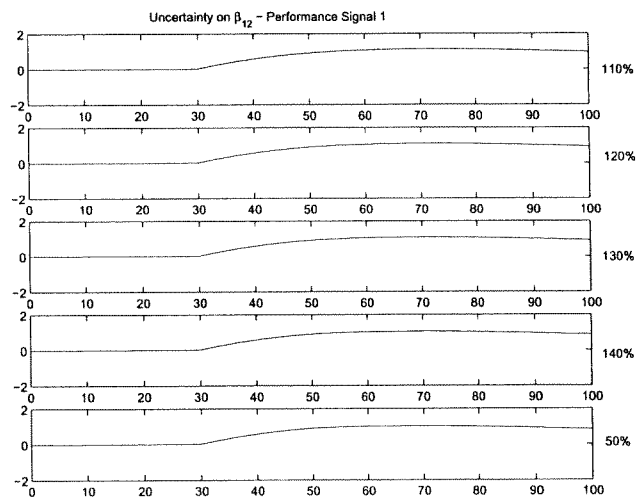


Figure 10.11: Response to Action 1 of the detection filter in the presence of uncertainty in the parameter β_{12} . The actual value of β_{12} varies from 110% (top graph) to 150% (bottom graph) of the nominal value.

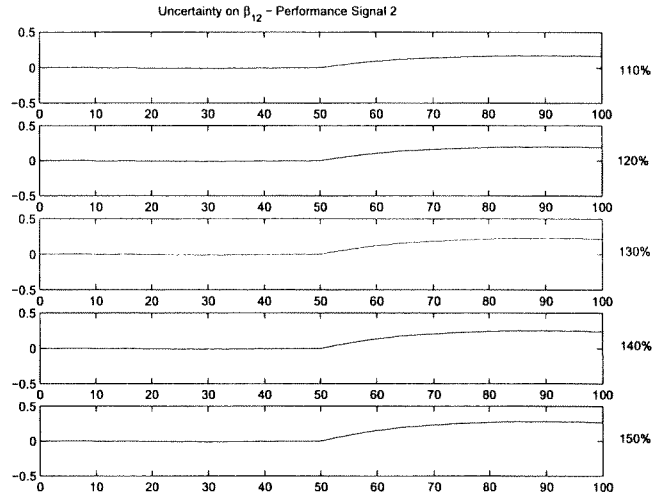


Figure 10.12: Response to Action 2 of the detection filter in the presence of uncertainty in the parameter β_{12} . The actual value of β_{12} varies from 110% (top graph) to 150% (bottom graph) of the nominal value.

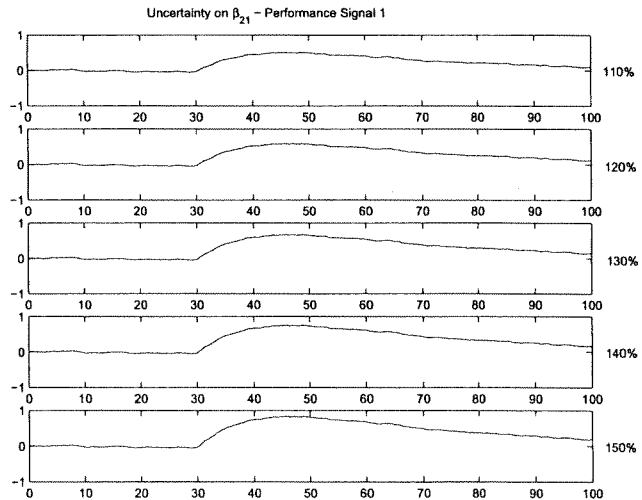


Figure 10.13: Response to Action 1 of the detection filter in the presence of uncertainty in the parameter β_{21} . The actual value of β_{12} varies from 110% (top graph) to 150% (bottom graph) of the nominal value.

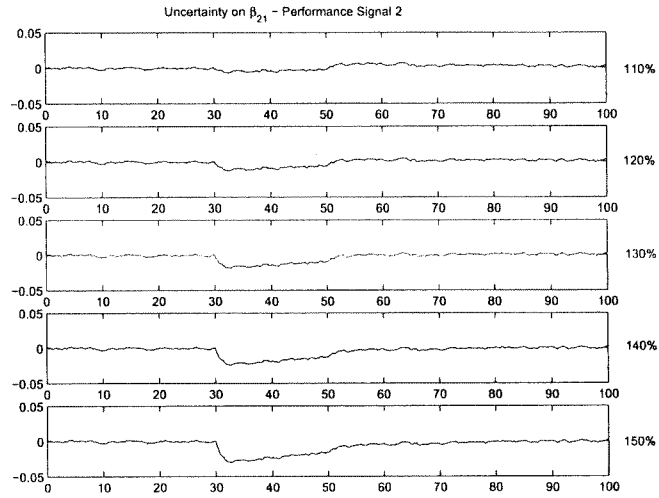


Figure 10.14: Response to Action 2 of the detection filter in the presence of uncertainty in the parameter β_{12} . The actual value of β_{12} varies from 110% (top graph) to 150% (bottom graph) of the nominal value.

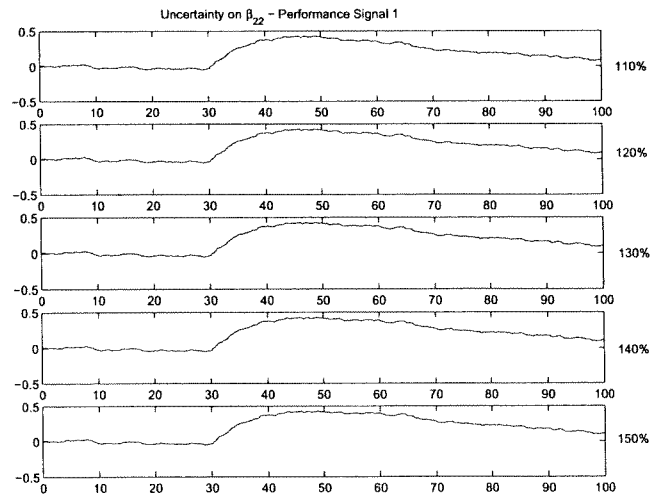


Figure 10.15: Response to Action 1 of the detection filter in the presence of uncertainty in the parameter β_{22} . The actual value of β_{22} varies from 110% (top graph) to 150% (bottom graph) of the nominal value.

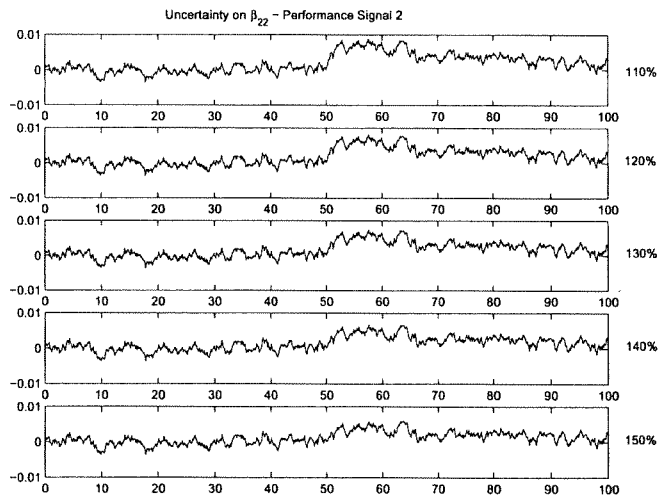


Figure 10.16: Response to Action 2 of the detection filter in the presence of uncertainty in the parameter β_{22} . The actual value of β_{22} varies from 110% (top graph) to 150% (bottom graph) of the nominal value.

Bibliography

- [1] Washington University, Mission Dynamics Continuous-Time Model, Version 2.42.
- [2] C. De Persis, A. Isidori, "On the observability codistribution of a nonlinear system," *Systems & Control Letters*, 40, 297-304, 2000.
- [3] C. De Persis, A. Isidori, A geometric approach to nonlinear fault detection and isolation, to appear as a regular paper on *IEEE Transactions on Automatic Control*, Sept. 2001.
- [4] C. De Persis, A. Isidori, "On the design of fault detection filters with game-theoretic-optimal sensitivity", submitted.
- [5] C. De Persis, A. Isidori, "Experiment 9: Detector Performance under Noise", Chapter 9, Final Report, DARPA JFACC Project, Washington University.

Chapter 11

Experiment 11: Method of Characteristics

11.1 Executive Summary

The purpose is to verify that the solution computed by the Sequential Linear-Quadratic Method (SLQM) is the same as the Nash solution computed by the Method of Characteristics. We verified that the solutions computed by the Sequential Linear-Quadratic Method (SLQM) are the same as the Nash solutions computed by the Method of Characteristics under several scenarios. Also, systematic tests have been performed to study robustness under two ways of enforcing constraints: penalties and explicit enforcement. Specifically, weights for velocities, engagement intensities, final numbers of platforms and targets, as well as maximum rated speeds have been varied. The results show that the trajectories are quite similar in shape.

11.2 Purpose of the Experiment

The purpose is to verify that the solution computed by the Sequential Linear-Quadratic Method (SLQM) is the same as the Nash solution computed by the Method of Characteristics.

11.3 Hypothesis to Prove or Disprove

Both the plant and internal models are the same, i.e., the Mission Dynamics Continuous-time Model (MDCM). The Method of Characteristics, which can be used to determine state and input trajectories of the Nash solution given the final states, verifies that the solution computed by the Sequential Linear-Quadratic Method (SLQM) is indeed the Nash solution.

11.4 Experiment Setup

Two scenarios will be created. The Nash solution of these will be calculated with SLQM, obtaining state and input trajectories. The final state of each solution will be used to solve the two-point boundary value problem (TPBVP) of game theory, by integrating the Hamiltonian system, (which is derived using the method of characteristics for partial differential equations), backwards in time. If the SLQM solution is indeed the Nash solution, it should be the same as the solution of the TPBVP.

11.5 Experiment Results

We consider the simplest case: Each of the Blue and Red forces has one unit. Both the Blue unit (**B1**) and the Red unit (**R1**) have 10 platforms. Since each unit has a 4-dimensional state and a 3-dimensional control input, the entire model has a 8-dimensional state and a 6-dimensional control input. In this experiment, each force has two objectives: i) to reach its specified fixed target; and ii) to reduce the number of enemy platforms while preserving the number of its own.

Numerical simulations have been performed for two different scenarios: “joust” and “cross”. In “joust”, blue and red trajectories tend to be parallel to each other, whereas in “cross”, they intersect each other.

In “joust”, there is a final terminal cost φ on the numbers of platforms in the payoff function. The parameter values are $a^B = a^R = 0.05$, $b^B = b^R = 0.0$, and $p^B = p^R = 0.0$. When control penalties are used, the parameter values are $R_{xy} = R_x^B = R_y^B = R_x^R = R_y^R = 300$, and $R_\pi = R_\pi^B = R_\pi^R = 75$. The initial positions are given by the following coordinates relative to a theater of operations of size 100 by 100: (20, 50) for **B1** and (80, 52) for **R1**. The location of targets are given by the following: (80, 52) for **B1** and (20, 50) for **R1**.

In “cross”, there is no terminal cost φ in the payoff function. The parameter values are $a^B = a^R = 0.05$, $b^B = 0.005$, $b^R = 1.5$, and $p^B = p^R = 0.0$. When control penalties are used the parameter values are $R_{xy} = R_x^B = R_y^B = R_x^R = R_y^R = 400$, and $R_\pi = R_\pi^B = R_\pi^R = 100$. The initial positions are given by the following coordinates relative to a theater of operations of size 100 by 100: (20, 50) for **B1** and (50, 80) for **R1**. The location of targets are given by the following: (80, 50) for **B1** and (50, 20) for **R1**.

For each scenario, three different cases have been tested. In all cases, there are penalties on the velocities. For engagement intensities, we test penalties without explicit constraints, explicit constraints without penalties, both penalties and explicit constraints. The effects of varying penalty parameters for velocities and engagement intensities are also studied.

11.5.1 Joust

Case 1: Penalties only, without explicit constraints on velocities or intensities. (Figures 11.1–11.3)

Case 2: Explicit constraints on intensities without penalties, no constraints on velocities. (Figures 11.4–11.6)

Case 3: Both penalties and explicit constraints on velocities (of type A) and intensities. (Figures 11.7–11.9)

Case 4: Change penalty for case 3. Now $R_{xy} = 240$, and $R_\pi = 55$. (Figures 11.10–11.12)

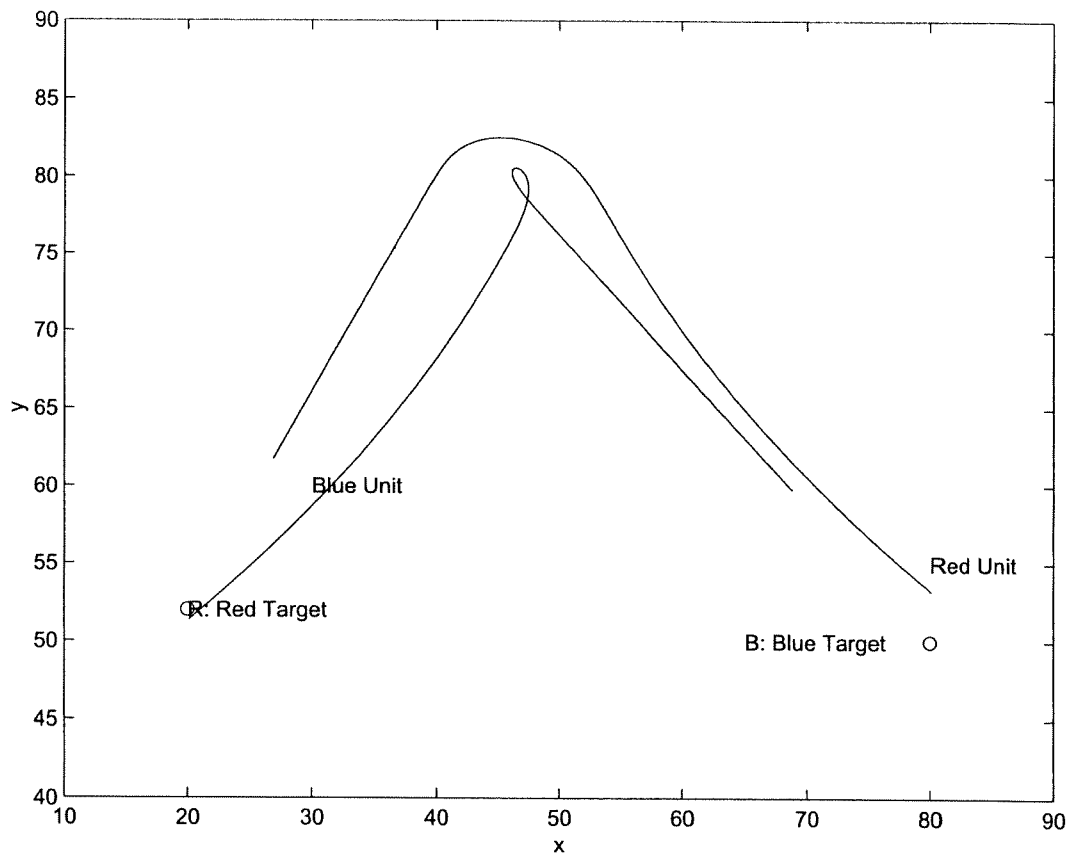


Figure 11.1: Trajectories

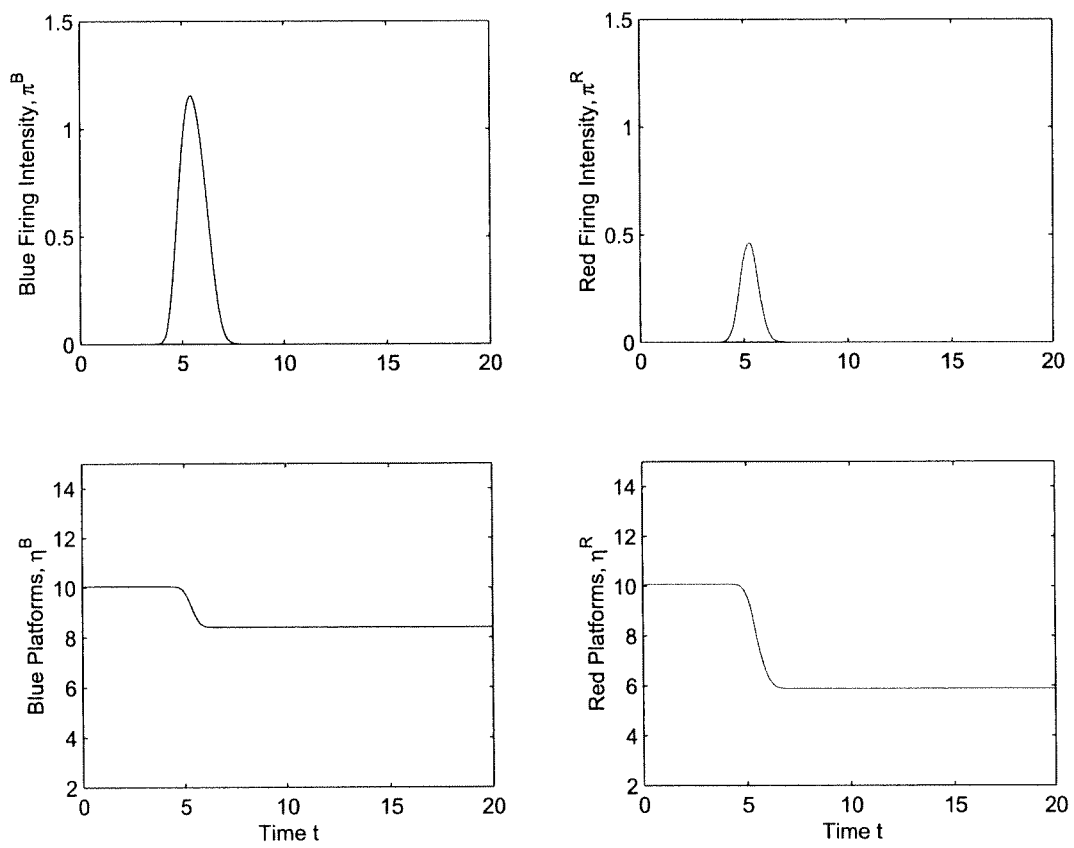


Figure 11.2: Engagement Intensities and Number of Platforms

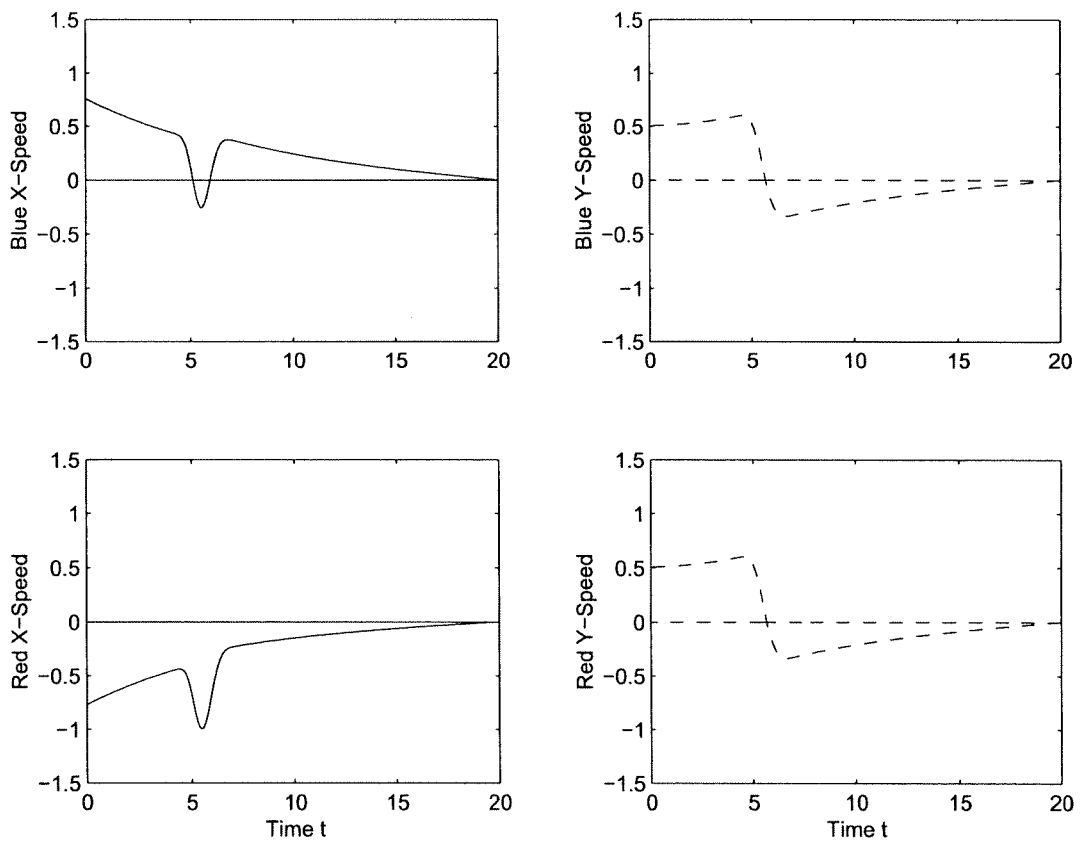


Figure 11.3: Velocities

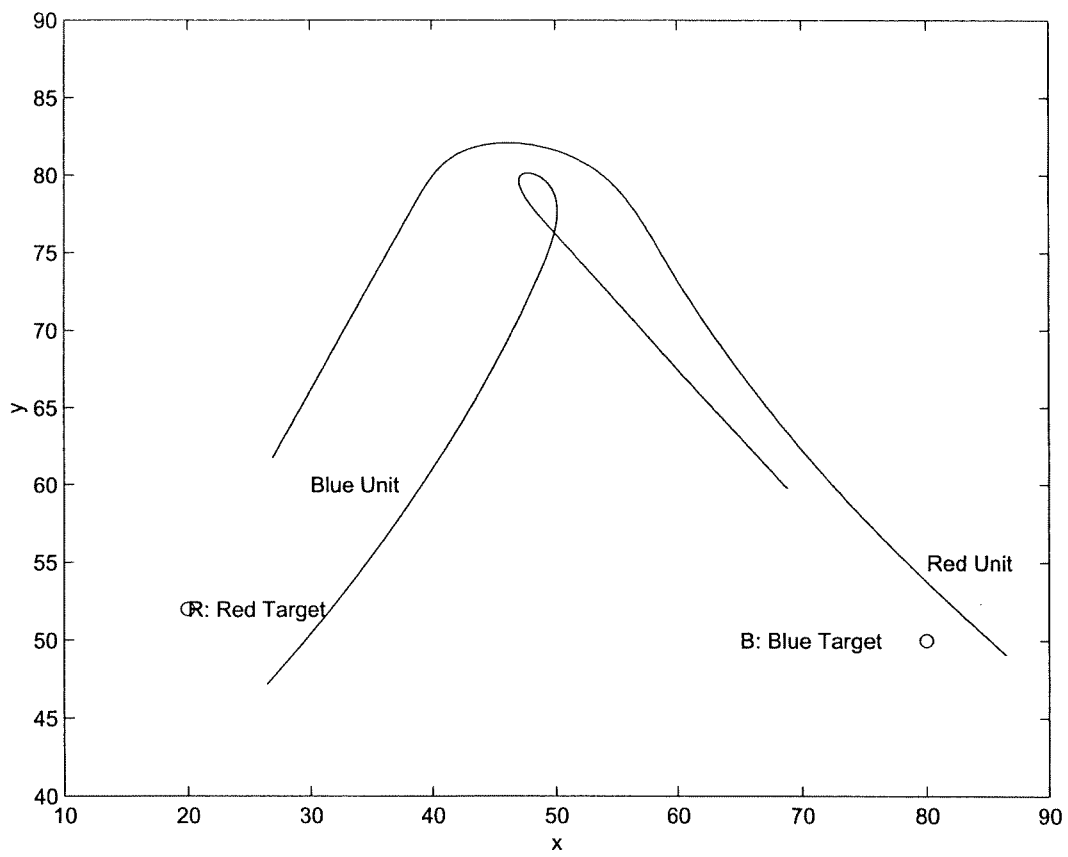


Figure 11.4: Trajectories

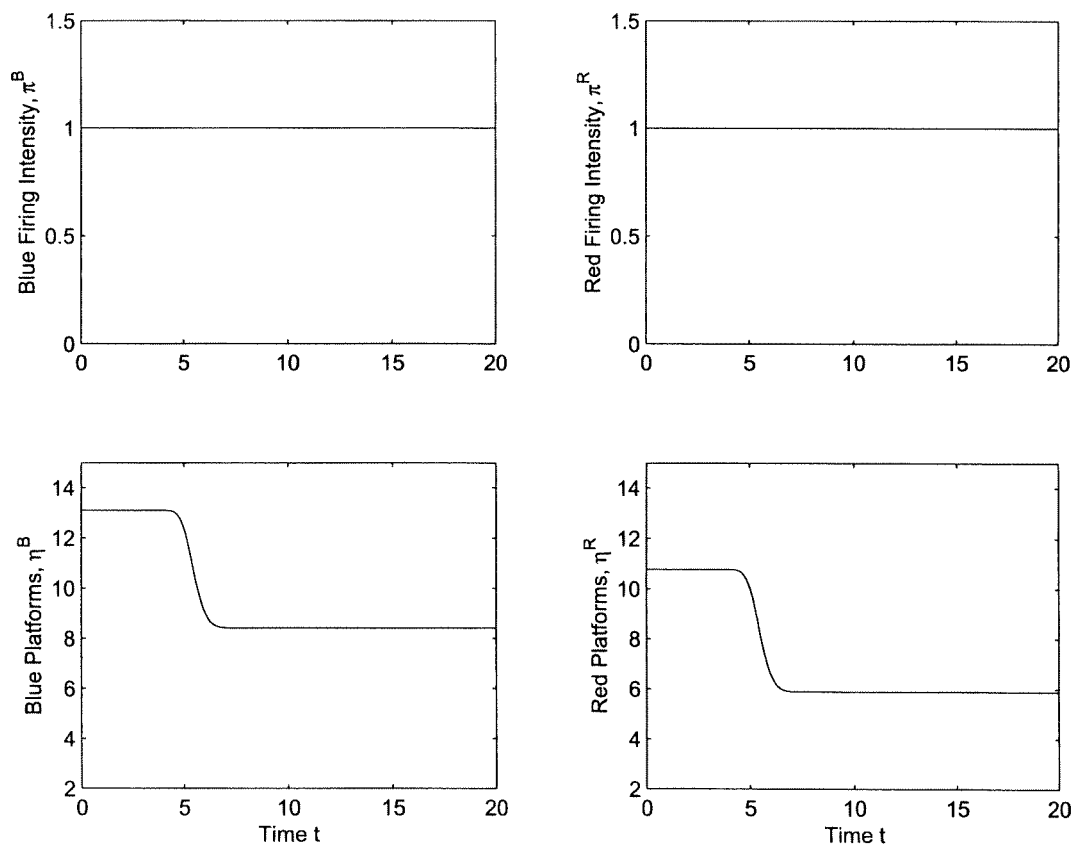


Figure 11.5: Engagement Intensities and Number of Platforms

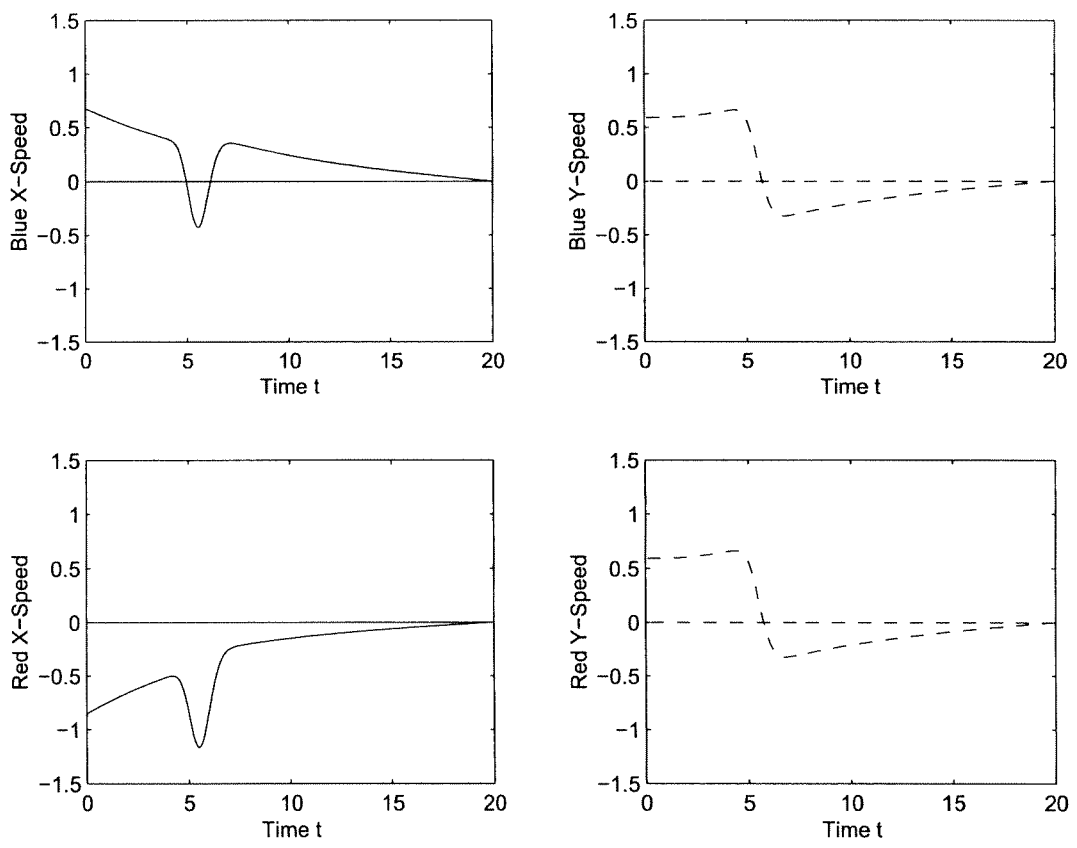


Figure 11.6: Velocities

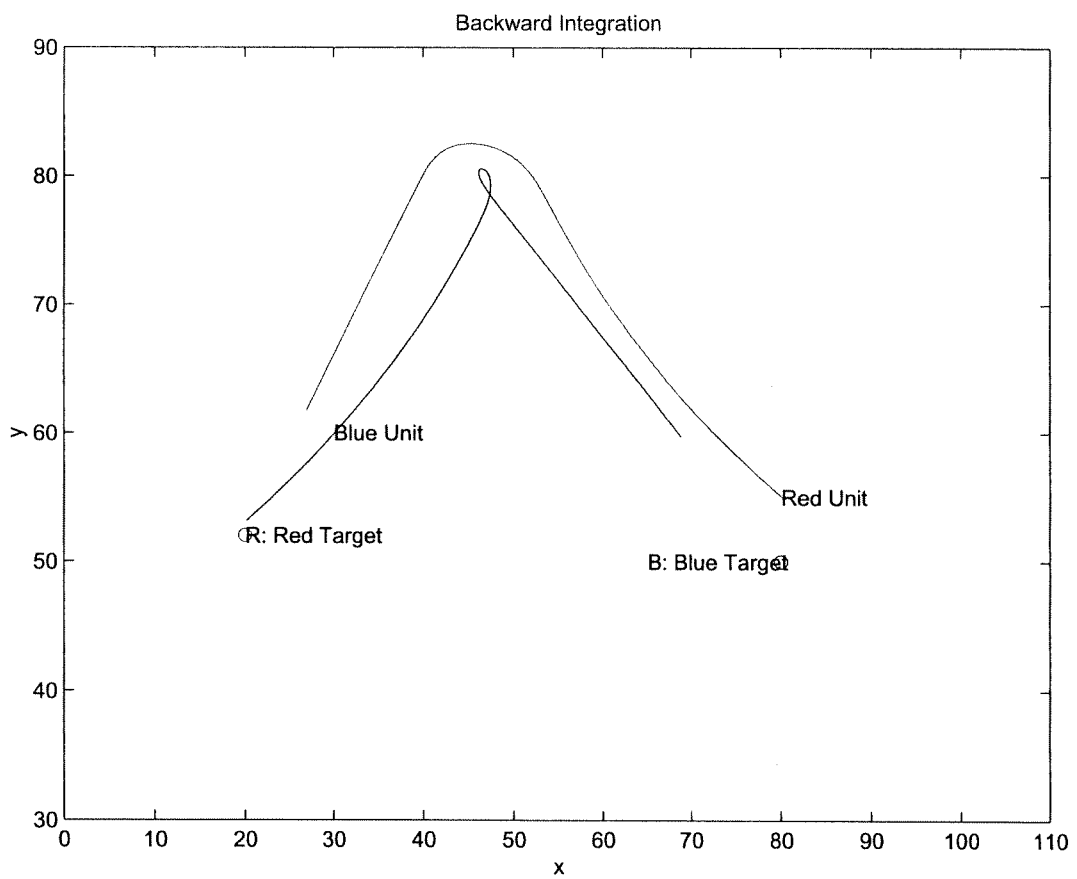


Figure 11.7: Trajectories

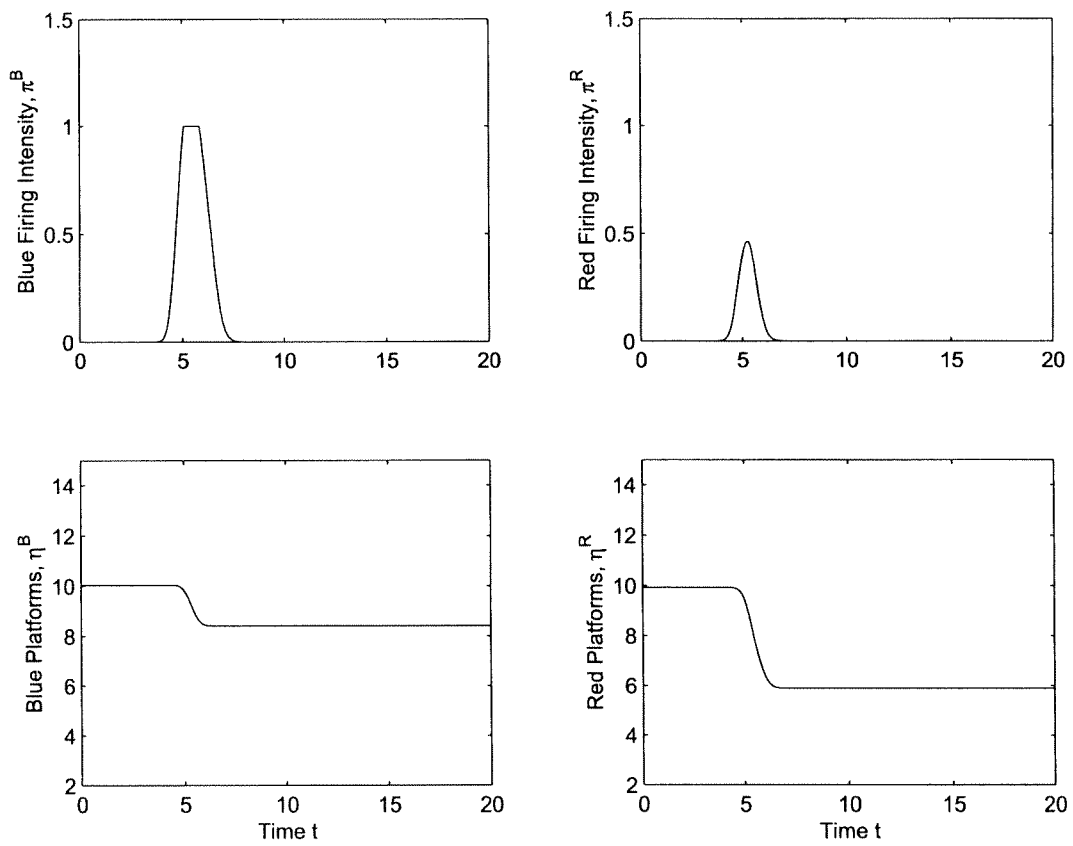


Figure 11.8: Engagement Intensities and Number of Platforms

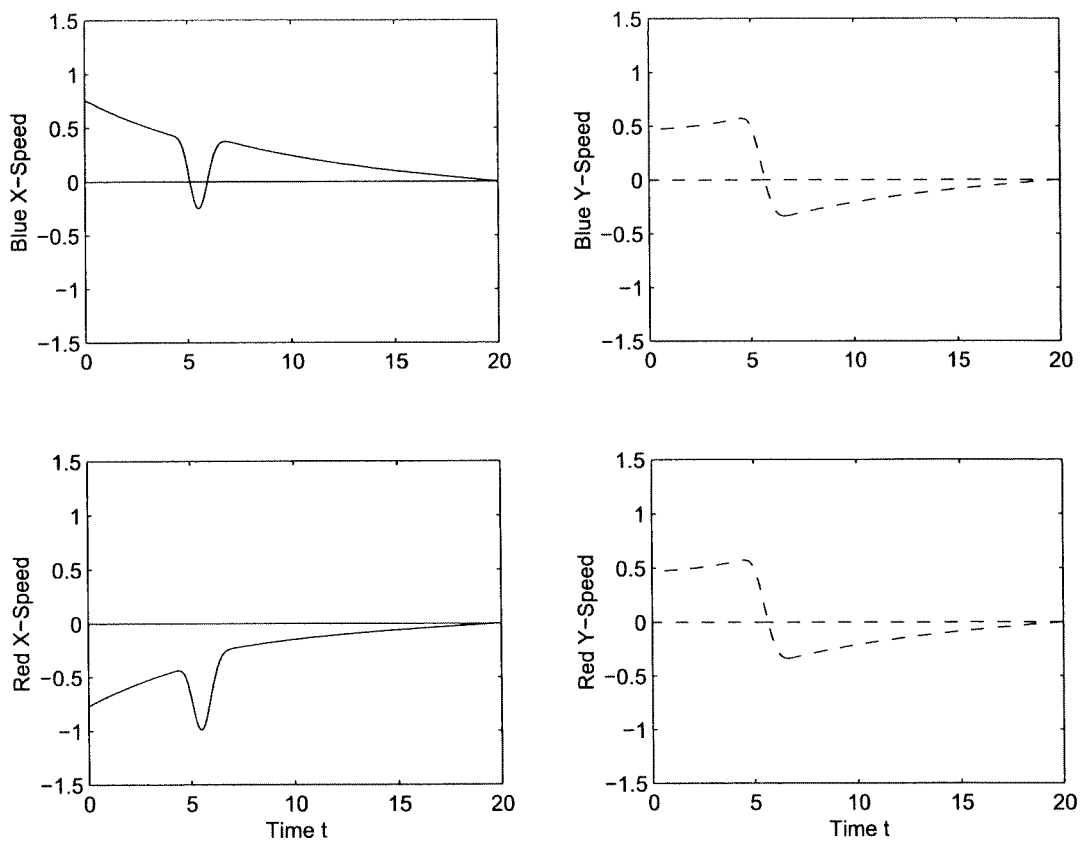


Figure 11.9: Velocities

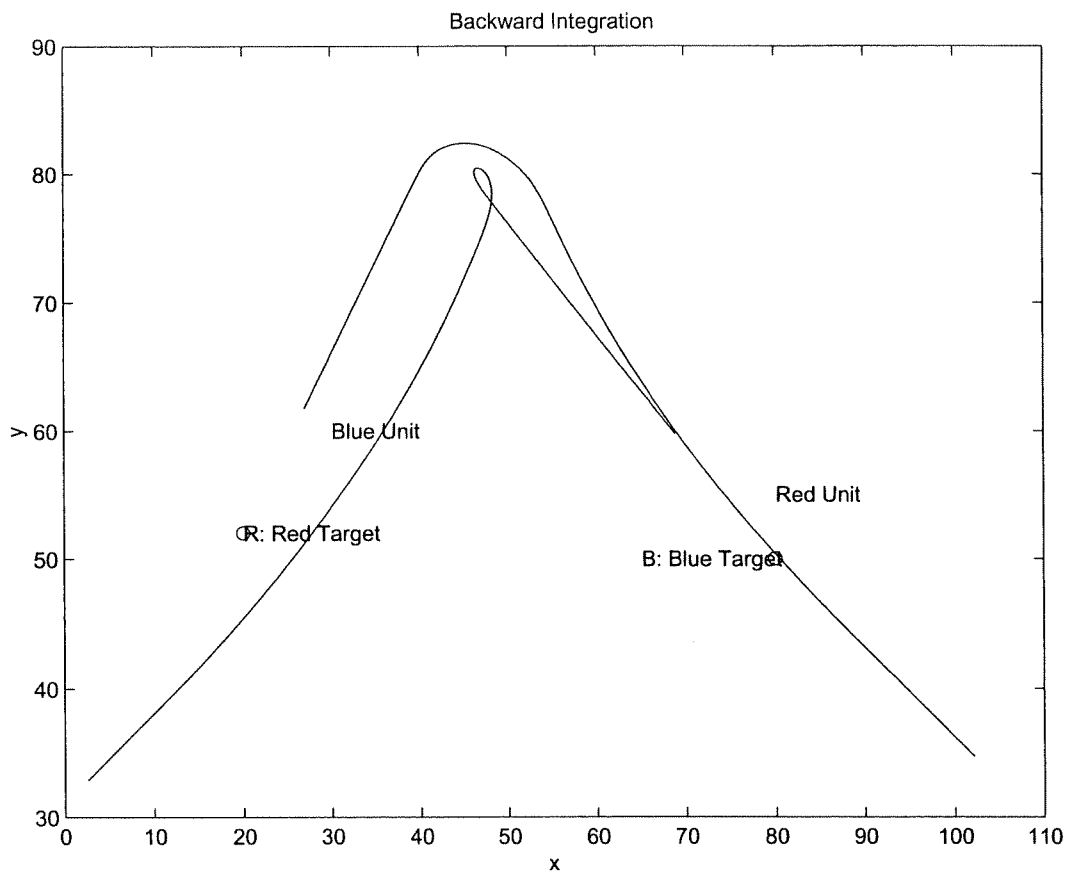


Figure 11.10: Trajectories

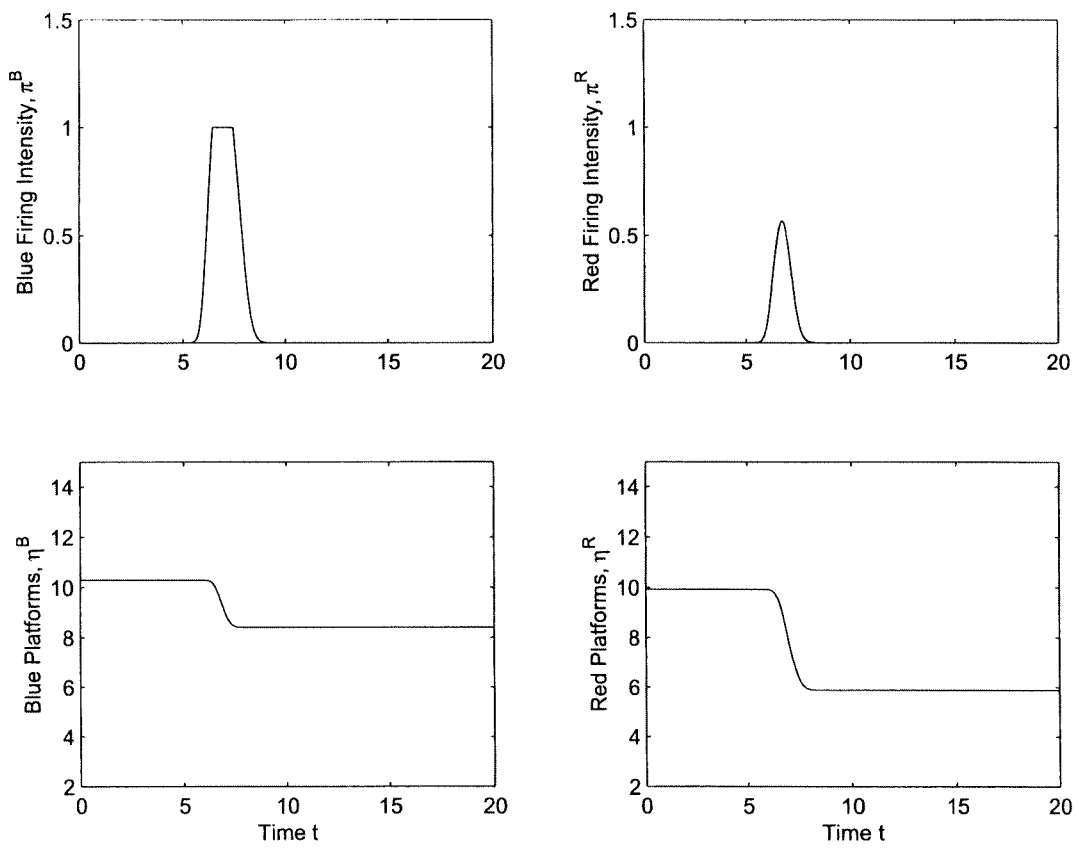


Figure 11.11: Engagement Intensities and Number of Platforms

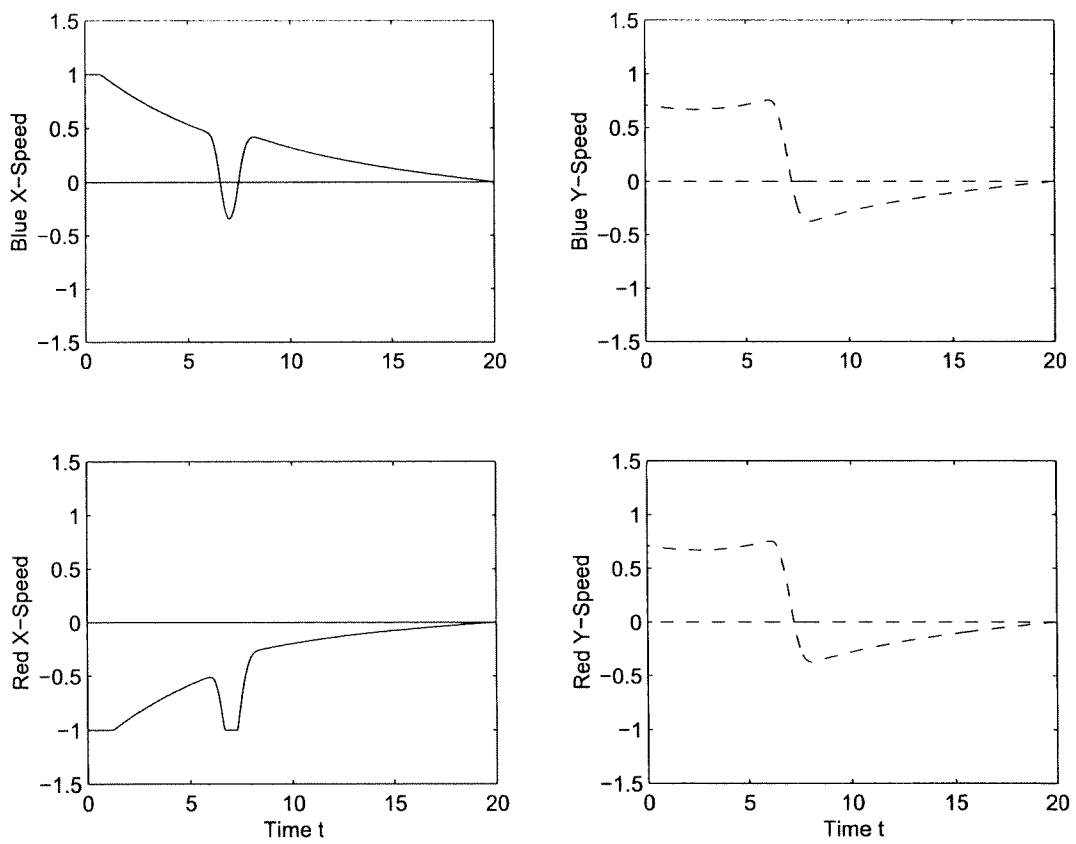


Figure 11.12: Velocities

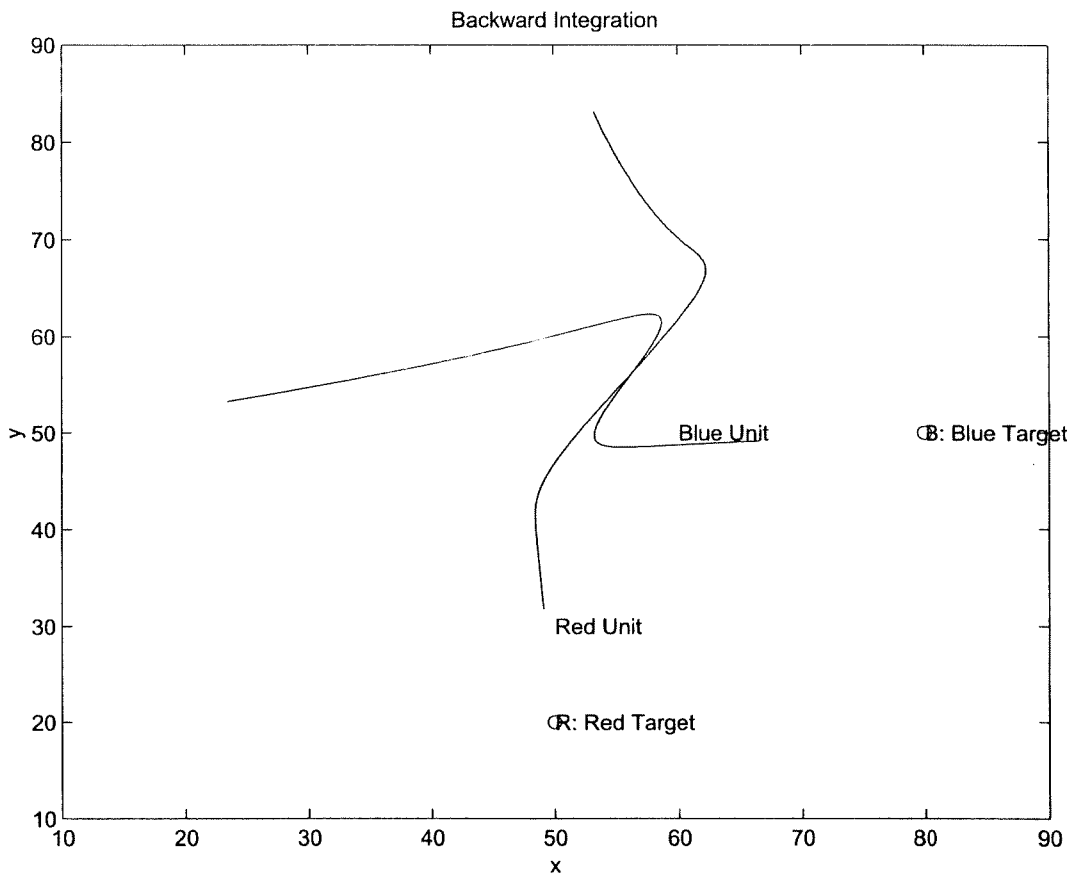


Figure 11.13: Trajectories

11.5.2 Cross

Case 1: Penalties only, without explicit constraints on velocities or intensities. (Figures 11.13–11.15)

Case 2: Explicit constraints on intensities without penalties, no constraints on velocities. (Figures 11.16–11.18)

Case 3: Both penalties and explicit constraints (of type A) on velocities and intensities. (Figures 11.19–11.21)

Case 4: Change penalty for Case 3. Now $R_{xy} = 310$, and $R_{\pi} = 10$. (Figures 11.22–11.24)

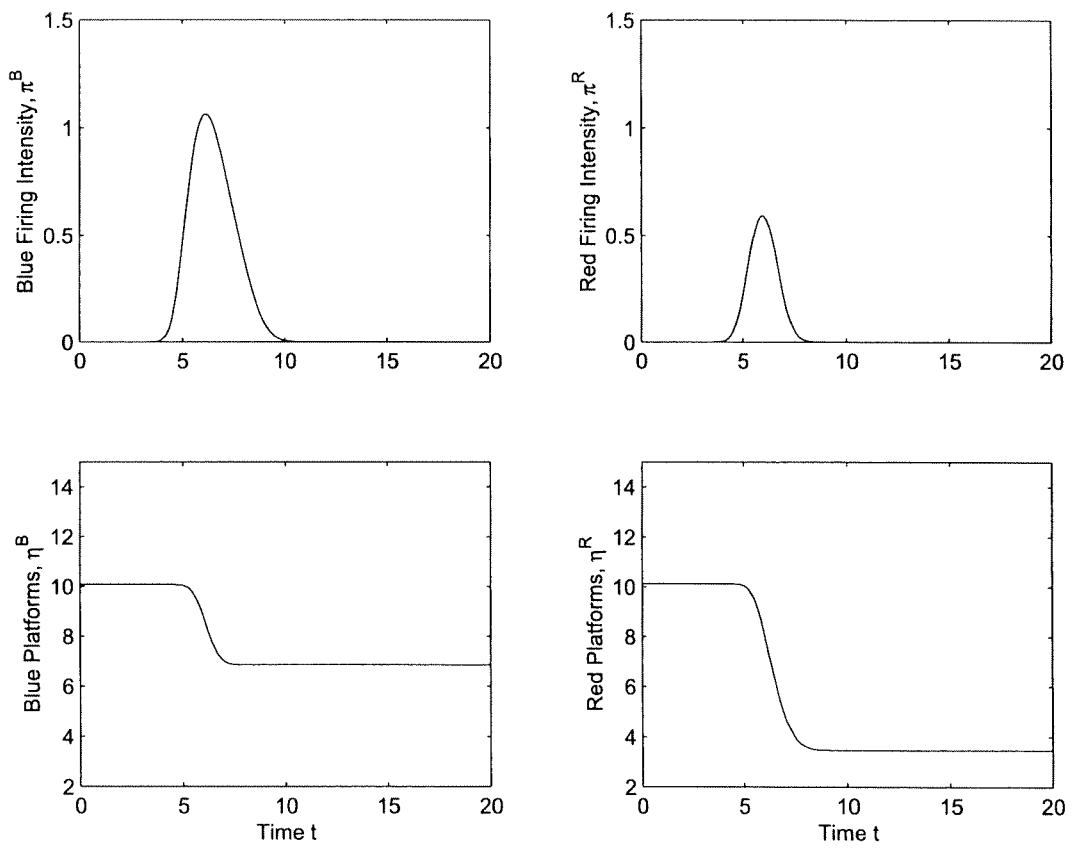


Figure 11.14: Engagement Intensities and Number of Platforms

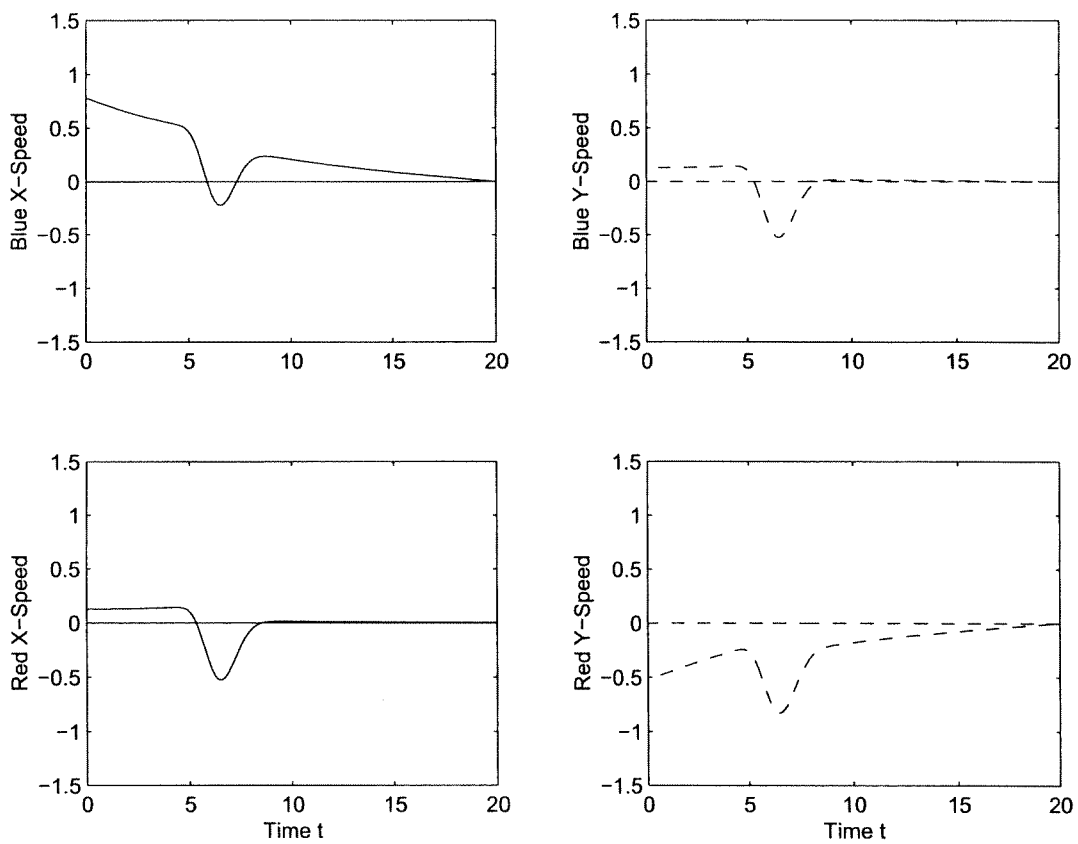


Figure 11.15: Velocities

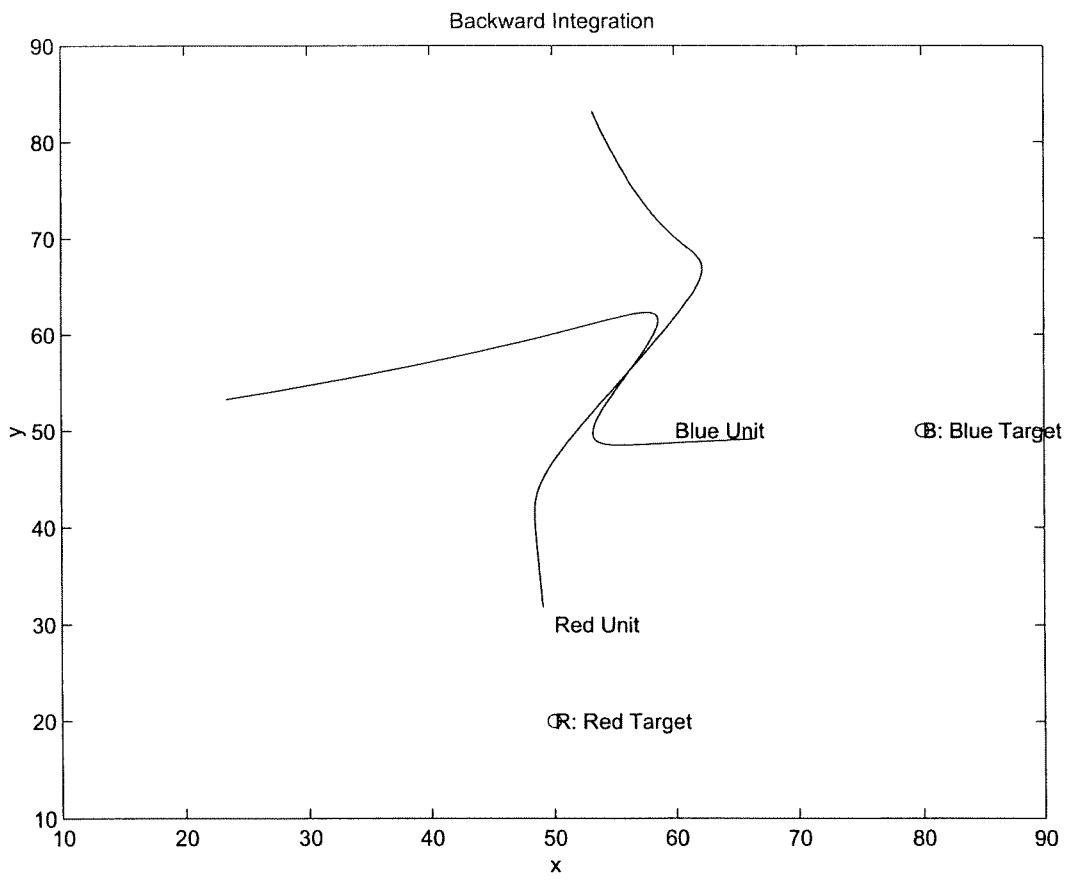


Figure 11.16: Trajectories

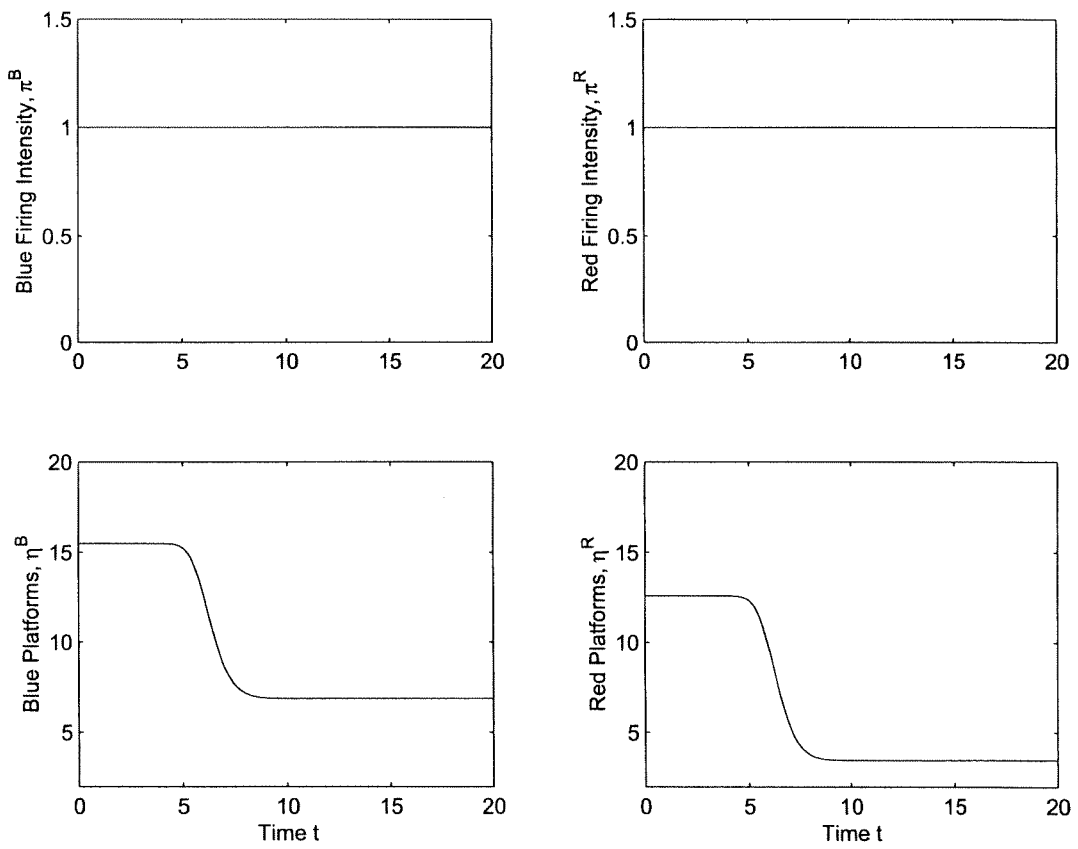


Figure 11.17: Engagement Intensities and Number of Platforms

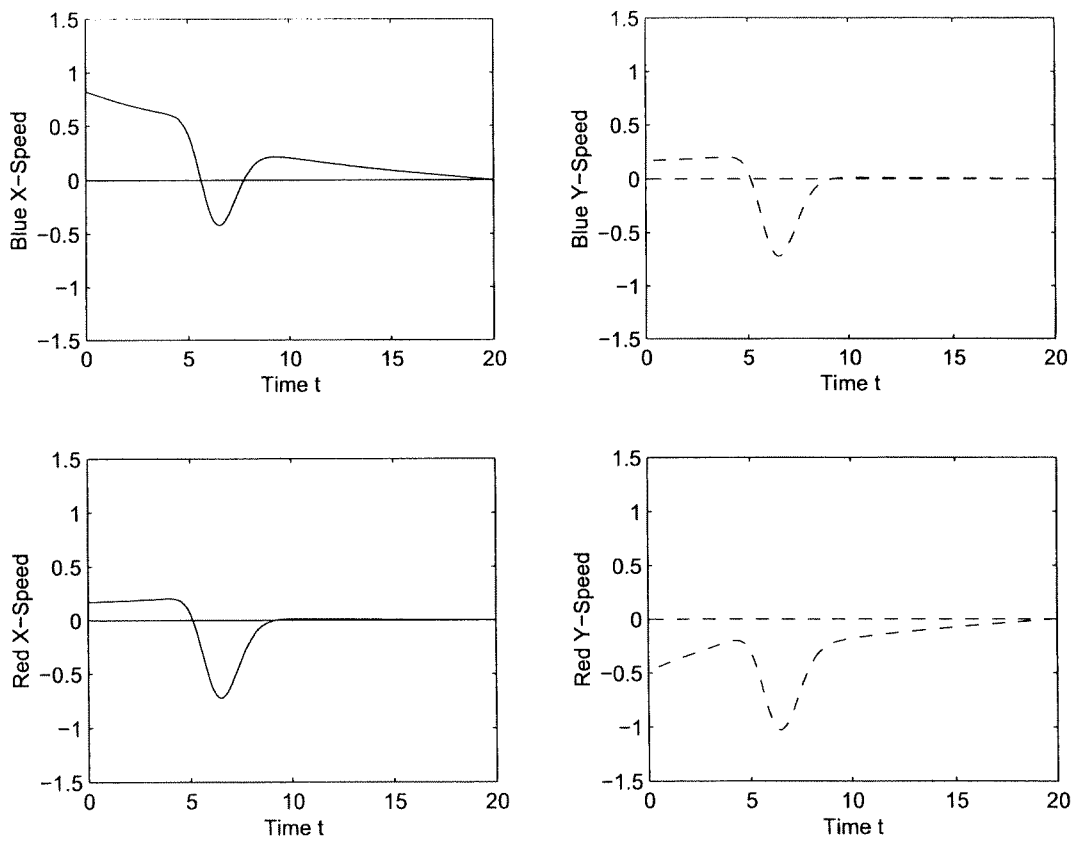


Figure 11.18: Velocities

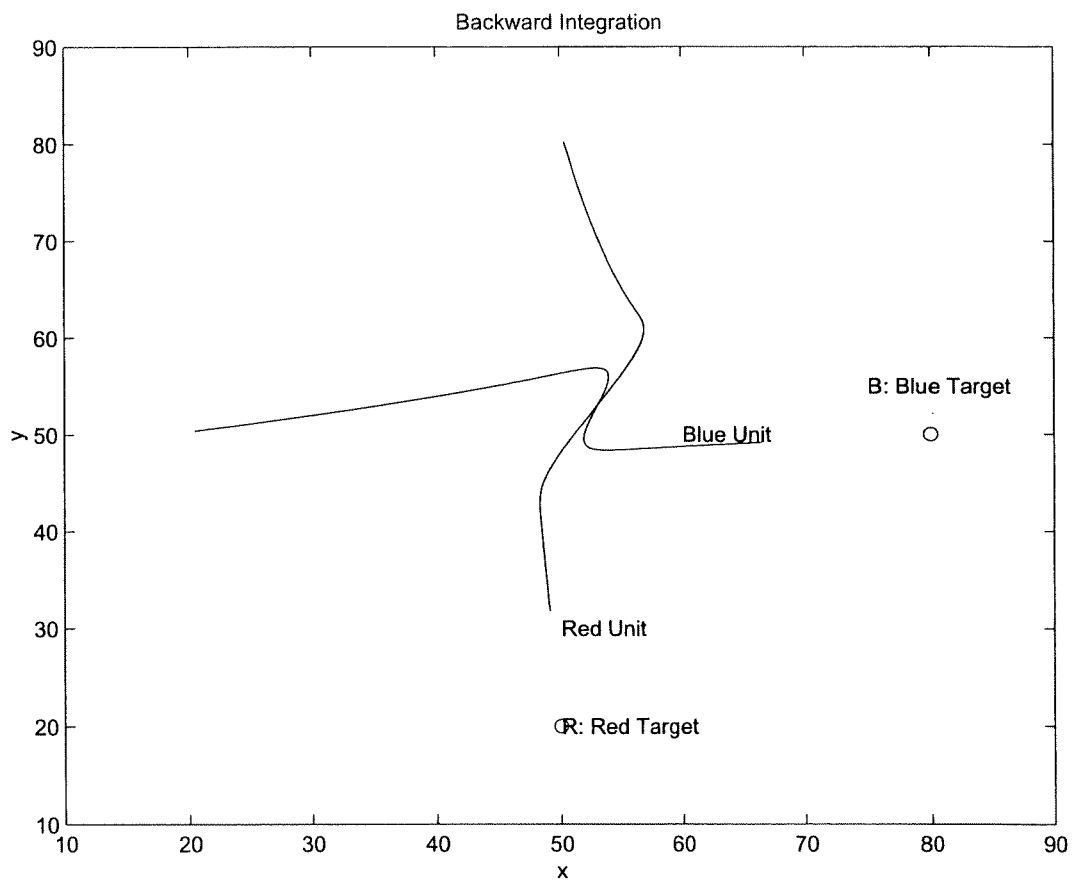


Figure 11.19: Trajectories

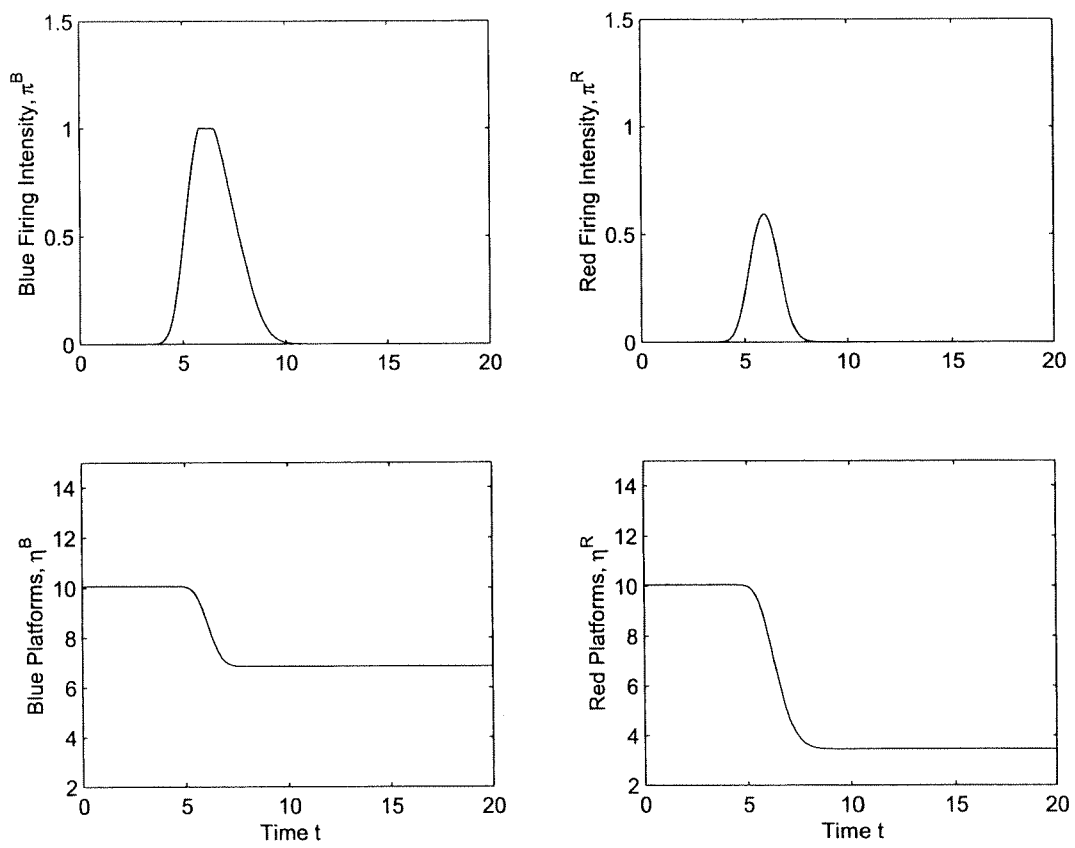


Figure 11.20: Engagement Intensities and Number of Platforms

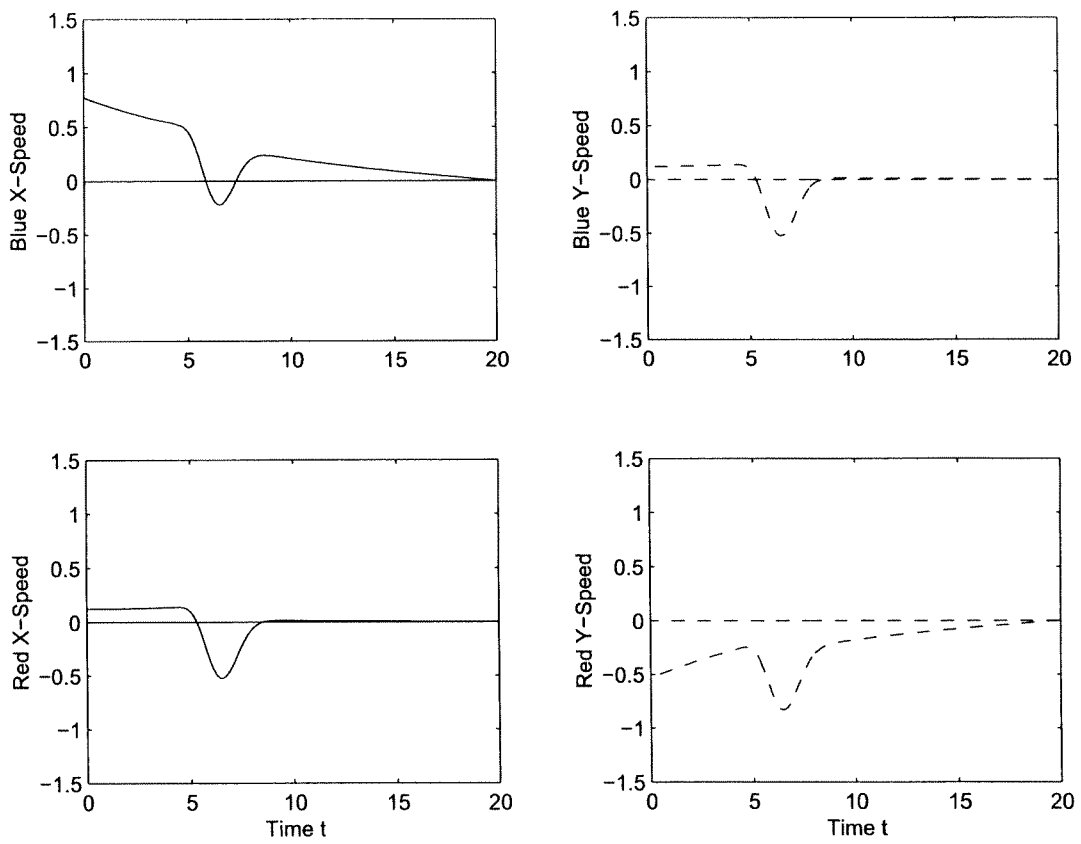


Figure 11.21: Velocities

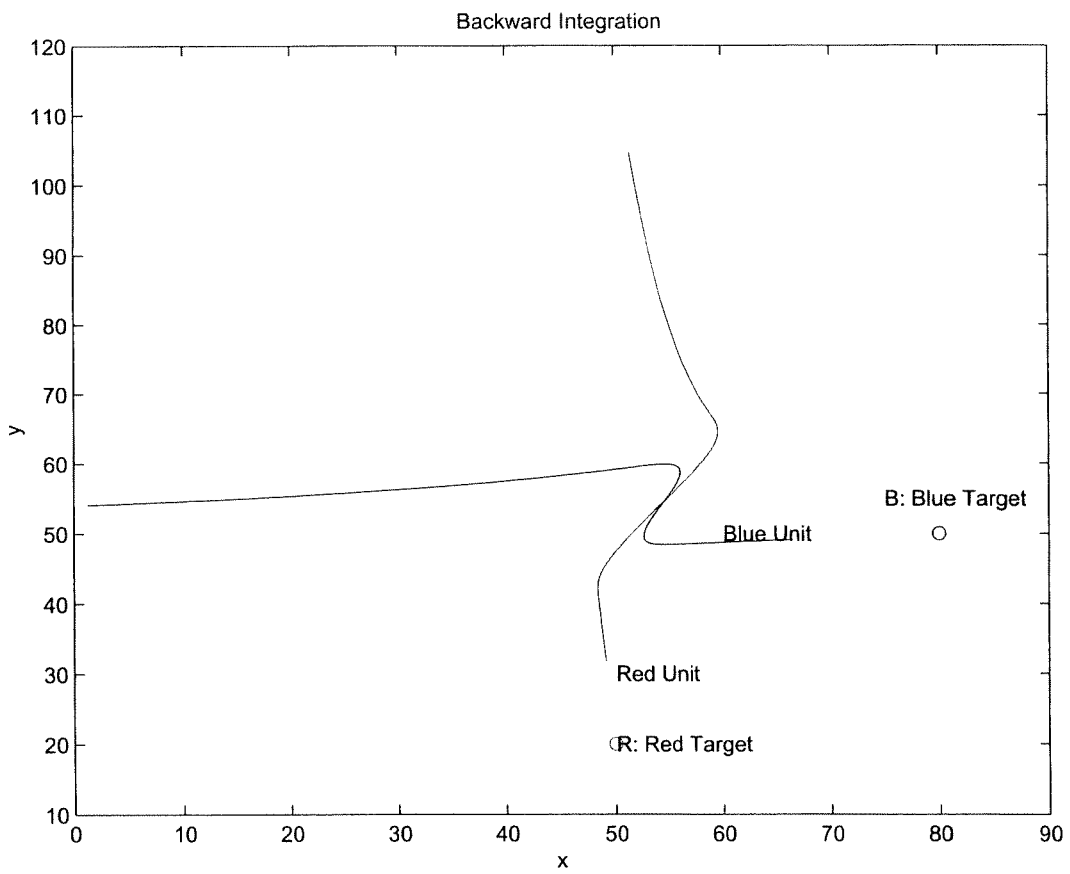


Figure 11.22: Trajectories

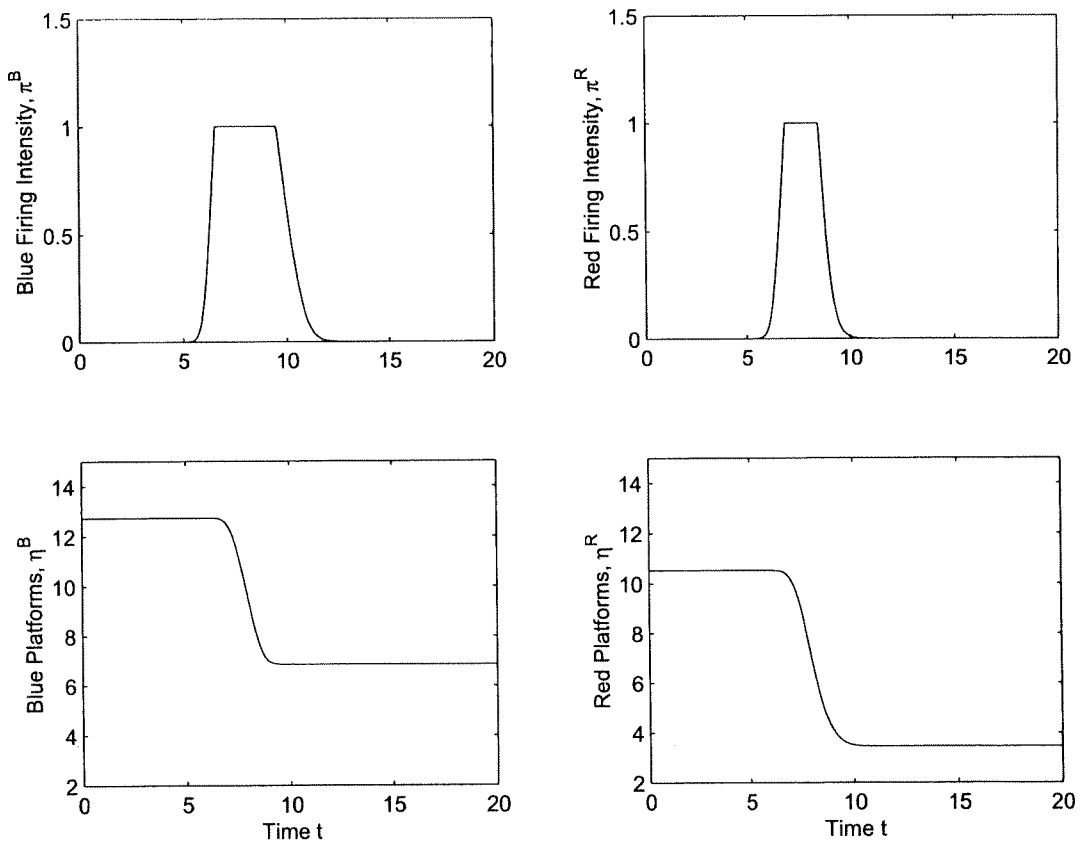


Figure 11.23: Engagement Intensities and Number of Platforms

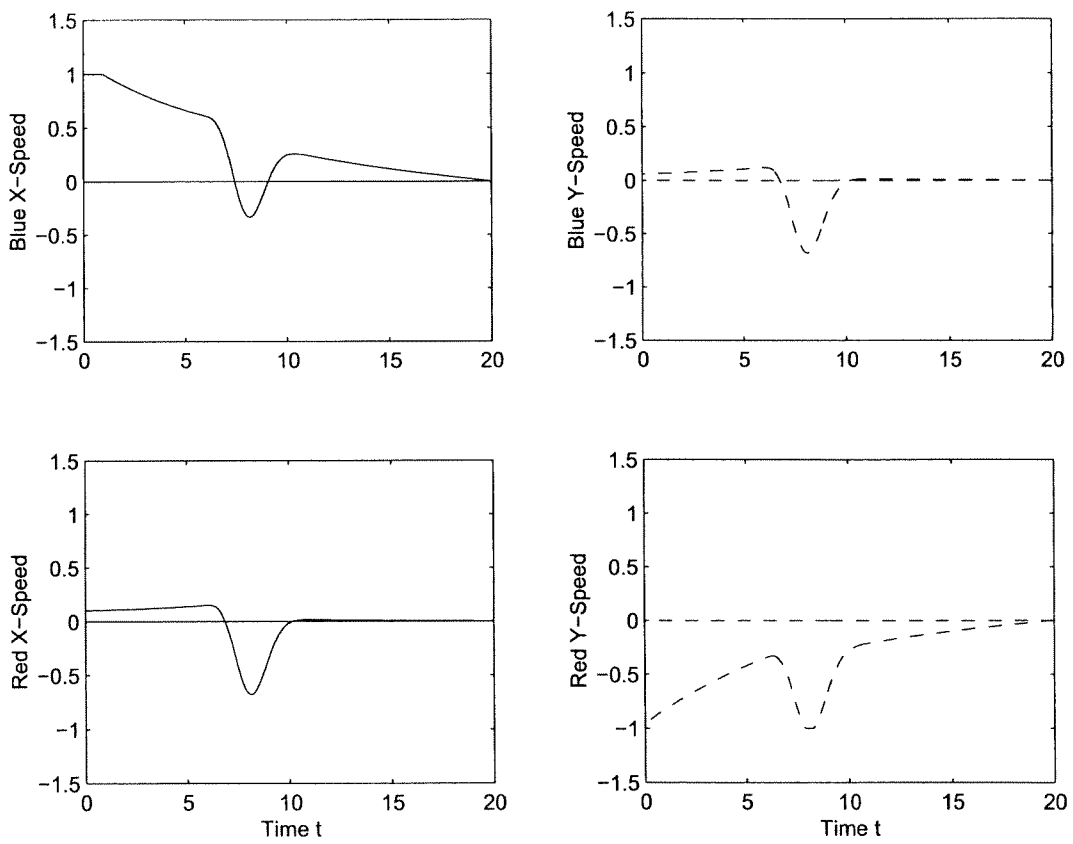


Figure 11.24: Velocities

Remark: Note that in some cases where constraints are enforced, engagement intensities are cut off at their limits (figures 9, 12, 21, 24, and figure 13 for velocities).

11.6 Analysis

Systematic tests have been performed to study two ways of enforcing constraints: penalties and explicit enforcement. Also, systematic tests have been performed to study robustness. Specifically, weights for velocities, engagement intensities, final numbers of platforms and targets, as well as maximum rated speeds have been varied. The results show that the trajectories are quite similar in shape.

11.7 Conclusions and Recommendations

We verified that the solutions computed by the Sequential Linear-Quadratic Method (SLQM) are the same as the Nash solutions computed by the Method of Characteristics under several scenarios.

Bibliography

- [1] H.W. Knobloch, A. Isidori and D. Flockerzi, Topics in Control Theory, DMV Seminar, Band 22, Birkhäuser, Basel, 1993.
- [2] H. Mukai, Y. Sawada, I. Tunay, Washington University JFAAC Team and Paul Girard, SAIC, Mission Dynamics Continuous-time Model (Version 2.55), a working report, Washington University, 2000.
- [3] H. Mukai, et al., Sequential linear quadratic method for differential games, in *Proceedings of the 2nd DARPA-JFACC Symposium on Advances in Enterprise Control*, pp. 159-168, Minneapolis, MN, July 2000.
- [4] H. Mukai, et al., Game-theoretic linear-quadratic method for air mission control, in *Proc. 39th IEEE Conf. Decision and Control*, pp. 2574-2580, Sydney, Australia, Dec. 2000.

Chapter 12

Experiment 12: Game Flow Model

12.1 Executive Summary

The purpose of this experiment was to validate the Game Flow approach. Validation is meant in the sense that the game theoretic solution engine (i.e., the *Sequential Linear–Quadratic* algorithm), acting on the Game Flow model, converges to a Nash solution that generally improves the value of the payoff function.

The Game Flow model simulates a two-force game where the assets of each force, say the blue or red forces, are distributed over a large geographical area.

In this experiment, the game area was a square divided into 64 square cells. At the start of the game, the two forces were spread uniformly over the entire game area, but the total strength of the blue force was only two thirds the total strength of the red force. To counter this mismatch, the attack range of the blue force was larger, and the cost of movement for the blue force was lower than that of the red force.

The goal of each force was to reach the end of the game with a minimum loss of their own strength, while inflicting maximum damage to the opposing force. Also, each force assigned more value (larger weight) to the cells located in the middle of the game area than to the cells located near the boundaries, so higher score might be earned by finishing the game with heavier strength concentration in more valuable cells. Finally, movement of assets across the game area was penalized, so economy of movement was also reflected in the final score of each force.

The game was carried out for a specified amount of time, with the phases of the game, i.e., asset movement and attack, evolving uninterrupted for the duration of the game.

The SLQ algorithm was used to find a Nash equilibrium solution for the game. In this experiment, the solver was stopped after 10 iterations, when the error (i.e., the norm of the velocity updates) was approximately one percent of the original error. At this error level, further iterations had an insignificant effect on the solution.

Experimental results show that the Nash equilibrium solution found by the SLQ algorithm, greatly improved the performance of the two forces with respect to the value of the payoff function selected for this experiment.

Qualitatively speaking, we can say that, in this scenario, the superiority of the blue force in the attack range, and its lower cost on movement prevailed, allowing the blue force to keep the red force out of the most valuable cells in the middle of the game area.

12.2 Purpose of the Experiment

The Game Flow model simulates a two-force game where the assets of each force, say the blue or red force, are distributed over a large area. The game area is divided into cells so that the strength concentration of the blue (resp. red) force in a cell is defined as the amount of blue (resp. red) asset of a single type contained in the cell divided by the area of the cell. The blue and red forces can move their respective

assets continuously during the game, by specifying transport velocities for each pair of contiguous cells, i.e., the rate at which the assets are shifted from one cell to the next.

At the start of a game, the two forces are assigned an initial strength distribution over all the cells in the game area. As the game proceeds, the initial strength distributions evolve in different ways, but the total strength of each force can only decrease due to two types of strength loss mechanisms.

The first type of loss mechanism is characterized by a local attrition parameter associated with each cell. This may represent loss due to mechanical failure or local weather.

The second type of loss mechanism for one force represents attacks from the opposing force. Attacks are carried out continuously and simultaneously by the two forces during a war game. For example, the blue assets contained in one cell at any one time will attack simultaneously all the red assets which are at that time in all the cells within the attack range of the blue force. The actual damage sustained by the red force in each cell will depend on the strength concentration of the blue force in the attacking cell, the strength concentration of the red force in the cell that is being attacked, and on the distance between the two cells.

The game is carried out for a specified amount of time, with the three phases of the game, i.e., asset movement, attrition and attack, evolving uninterrupted for the duration of the game.

The goal of each force is to reach the end of the game with a minimum loss of their own strength, while inflicting maximum damage to the opposing force. Also, each force may assign more value (larger weight) to some of the cells in the game area than to other cells, so a higher score might be earned by finishing the game with heavier strength concentration in more valuable cells. Finally, movement of assets across the game area typically costs valuable energy, so economy of movement is also reflected in the final score of each force.

The purpose of this experiment was to validate the Game Flow approach. Validation is in the sense that the game theoretic solution engine (i.e., the *Sequential Linear-Quadratic* algorithm), acting on the the Game Flow model, converges to a solution that generally improves the value of the game, i.e., the payoff function value.

12.3 Hypothesis to Prove or Disprove

The hypothesis that we tried to prove in this experiment is: The Game Flow model and the SLQ algorithm constitute a feasible tool for solving differential games involving a very large number of opposing units, distributed over a geographical area.

12.4 Experiment Setup

We apply the Game Flow method to find a solution for the game described by the following scenario.

The game area is a square of unit length in each side, and is divided into 64 squares to form an 8×8 grid.

The game area may represent some geographical area where the conflict takes place. It should be expected then, that certain local features of the game area will have an effect on the evolution of the game. For example, the energy that the forces must spend to move their respective assets should vary as they attempt to go across different types of terrain: dessert dunes, marshy land, dense forests, etc. Similarly, different features of the game area might affect the attrition rate sustained by the forces.

In this experiment, the game area consists of two types of terrain: one smooth area, through which movement of assets is relatively easy, surrounded by more difficult terrain. A map of the game area is shown in Figure 12.1, with the smooth area indicated in dark color. Notice that the difficulty in the terrain need not be the same along the horizontal and vertical directions, even locally. In this experiment, for example, it is clear that to go from cell (4,4) to cell (3,5), the path that goes through cell (3,3) is less expensive than the path that goes through cell (3,5).

To simplify visualization of the experiment results, attrition not caused by the enemy is not included in this experiment.

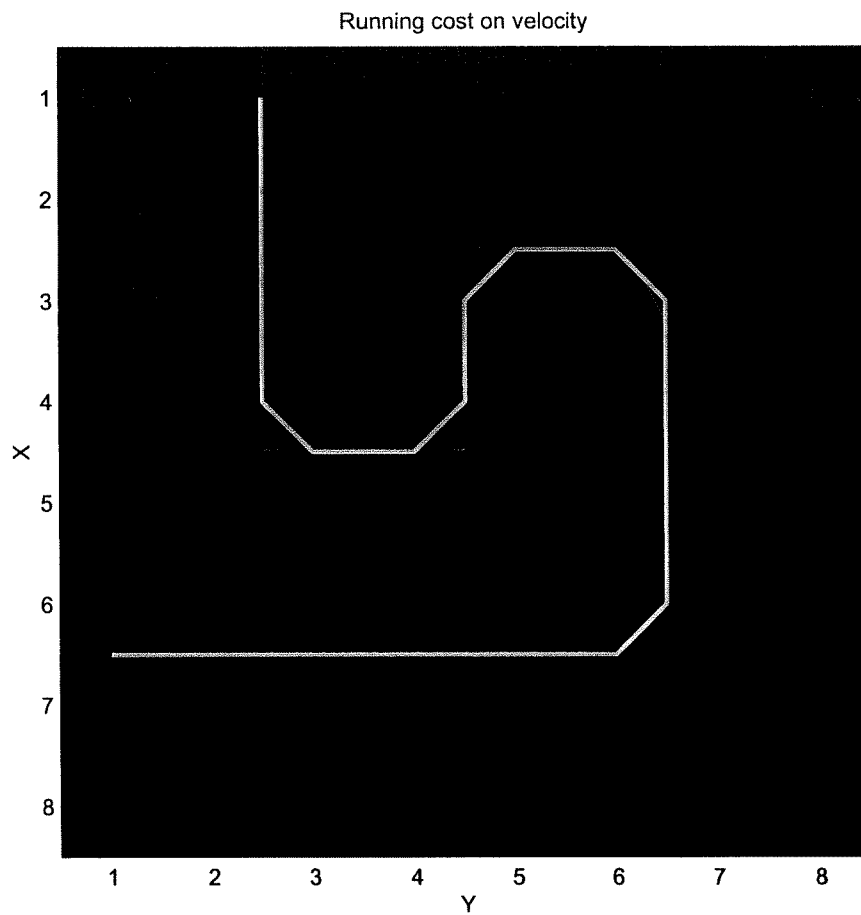


Figure 12.1: Map of the game area: light colored cells indicate smooth area. The white contour line marks the boundary between the two different regions.

The blue force has an initial strength of 128 units spread uniformly on the game area. The red force has an initial strength of 192 units, also spread uniformly on the game area.

We assume that each force has a symmetric attack efficiency function as depicted in Figures 12.2 and 12.3. The Figures show that the red force is slightly more powerful than the blue force at close range. On the other hand, the blue force has a longer range, covering an area of approximately 3×3 cells.

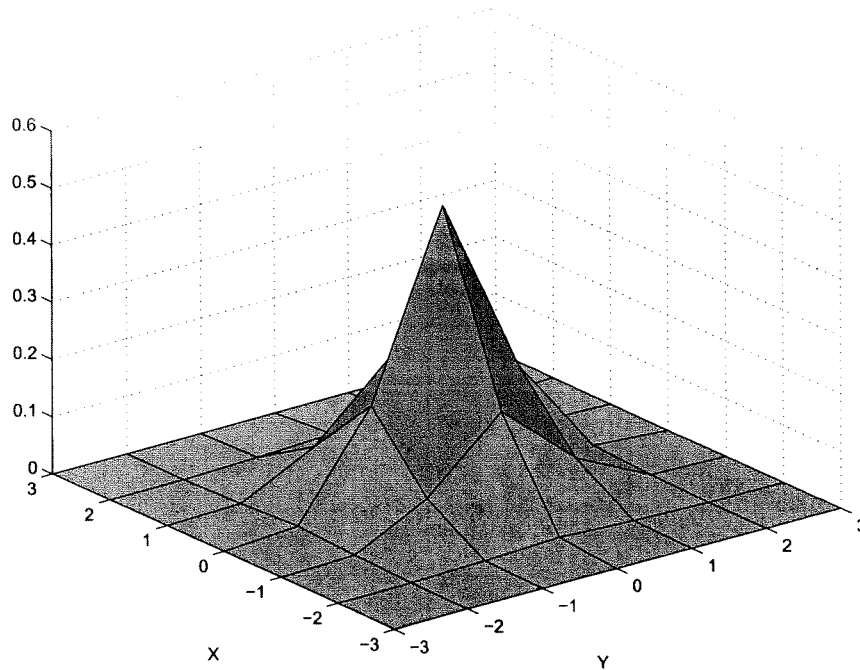


Figure 12.2: Efficiency of attack for the blue force.

The mission for the blue force is defined as: (a) reach the end of the game with as much strength as possible; (b) place as many assets as possible in the four cells located in the middle of the game area; (c) remove the red assets from the four central cells, and block any attempts by the red force to move its own assets into that area; (d) continuously try to minimize the strength of the red force; and (e) minimize the control effort in accomplishing the first four items of this mission statement.

Similarly, the mission for the red force is defined as: (a) reach the end of the game with as much strength as possible; (b) place as many assets as possible in the four cells located in the middle of the game area; (c) remove the blue assets from the four central cells, and block any attempts by the blue force to move its own assets into that area; (d) continuously try to minimize the strength of the blue force; and (e) minimize the control effort in accomplishing the first four items of this mission statement.

While the respective mission statements for the blue and red forces may look the same, each force can assign different priorities (weights) to the different mission tasks. For instance, in this experiment, the cost for the blue force associated with movement of assets over the smooth game area is equivalent to three fourths the corresponding cost for the red force. This means that the blue force has more freedom of movement over the game area than the red force.

As for the values (weights) assigned by each force to the different cells in the game area, the respective mission statements imply that both forces regard the central cells more valuable than the cells located near the edges of the game area. For this experiment, we assume that the two forces assign equal value to each cell, so that a common map of the real estate value is shown for the two forces in Figure 12.4.

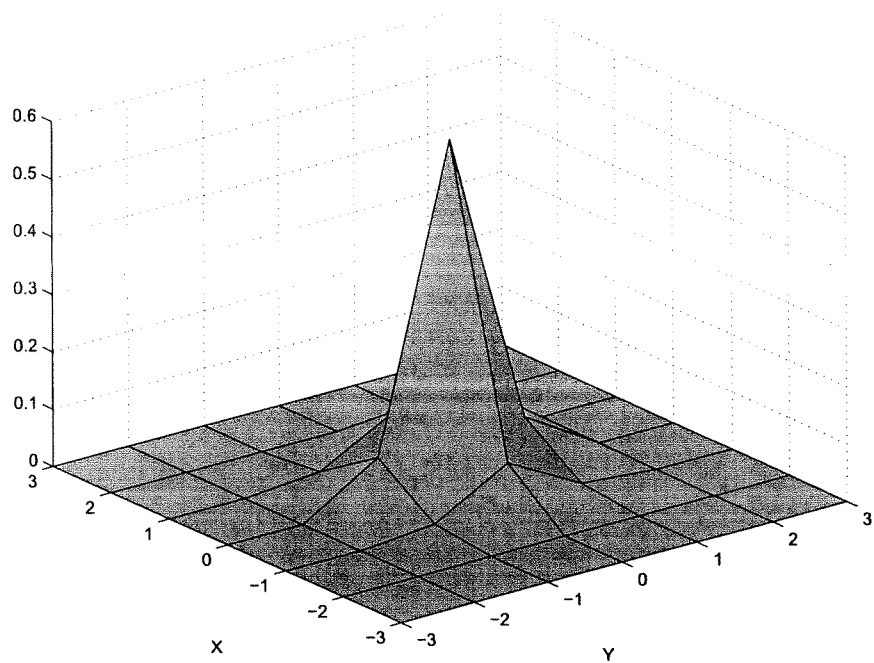


Figure 12.3: Efficiency of attack for the red force.

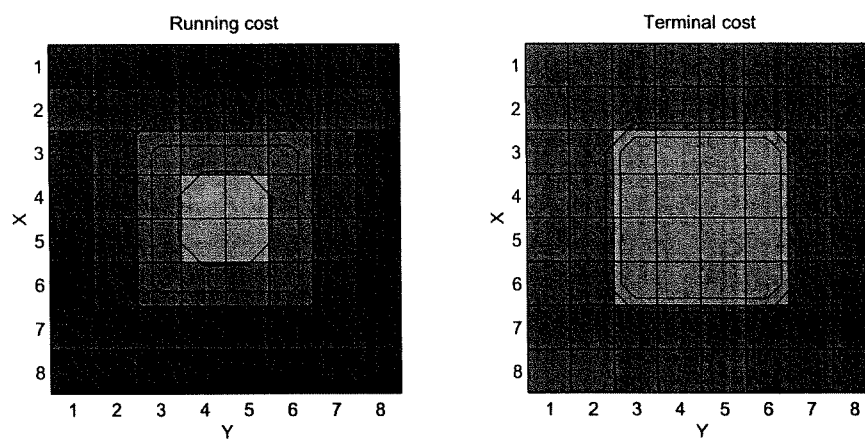


Figure 12.4: (a) Running cost associated with instant values of the strength concentration in the cells. (b) Terminal cost associated with final values of the strength concentration in the cells. Lighter color indicates higher value.

Table 12.1: Payoff function value for the initial guessed solution

Force	Running X-vel.	Running Y-vel.	Running Strength	Terminal Strength	Game Cost
Blue	0	216	-259.04	-469.68	-512.73
Red	-246	0	523.32	961.43	1238.76
Total					726.03

12.5 Experiment Results

The solution technique (SLQ method) is iterative and it improves the current solution estimate at each iteration. Hence, to solve the game, a guess has to be made for the velocity distributions in the horizontal and vertical directions that the forces continuously apply. The initial choice of the velocity distributions affect the rate of convergence in the iterative solution of the game, but in our experiments it did not have significant effects on the final outcome of the game. So, in the experiment we report here, we arbitrarily assigned a uniform velocity distribution parallel to the horizontal direction for the blue force, and a uniform velocity distribution parallel to the vertical direction for the red force.

The value of the game corresponding to the initial non-optimum solution is shown in Table 12.1, broken into the individual cost components.

The SLQ algorithm was used next to find a Nash equilibrium solution for the game. In this experiment, the solver was stopped after 10 iterations, when the error (i.e., the norm of the velocity updates) was approximately one percent of the original error. At this error level, further iterations had an insignificant effect on the solution. Convergence of the algorithm is illustrated in Figure 12.5, which shows the error against iterations.

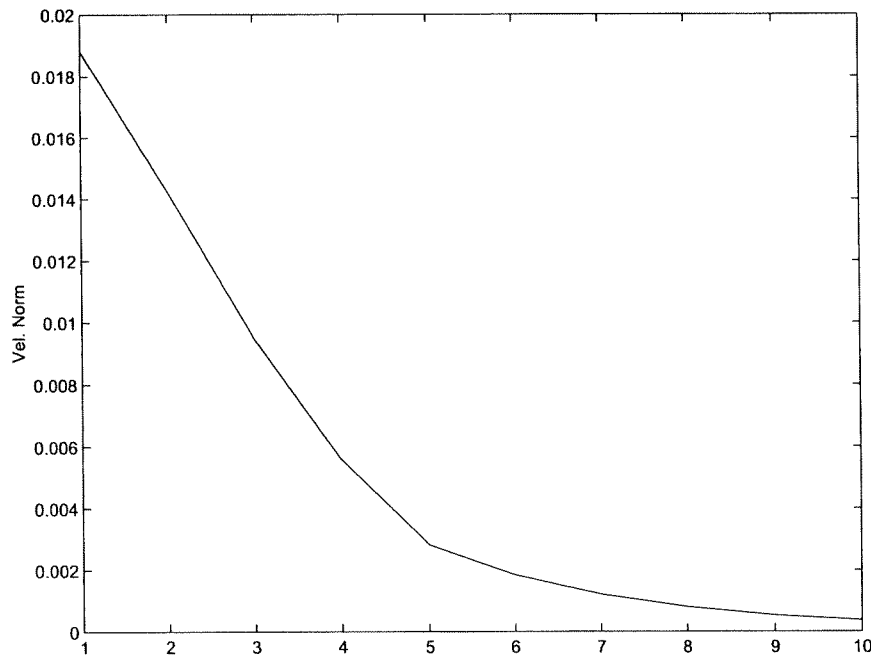


Figure 12.5: Convergence of SLQ algorithm

The value of the game corresponding to the Nash equilibrium solution is shown in Table 12.2, broken

Table 12.2: Payoff function value for the Nash equilibrium solution

Force	Running X-vel.	Running Y-vel.	Running Strength	Terminal Strength	Game Cost
Blue	14.66	10.89	-264.24	-497.18	-735.86
Red	-18.99	-20.06	518.90	1013.80	1493.66
Total					757.80

into the individual cost components.

Clearly, the Nash equilibrium solution found by the SLQ algorithm, greatly improves the performance of the two forces in terms of the value of the payoff function selected for this experiment. Recall that the blue force tries to minimize the total payoff while the red force tries to maximize it.

Figures 12.6 and 12.7 show the initial strength distributions for the blue and red forces. The distributions are identical and uniform, so the shade (color) is uniform over the area. The direction of asset movement across the border between each pair of cells is indicated by an arrow. The size of the arrows indicate the magnitudes of the initial velocity components. The contour lines mark the different regions in the game area as defined by the cost components.

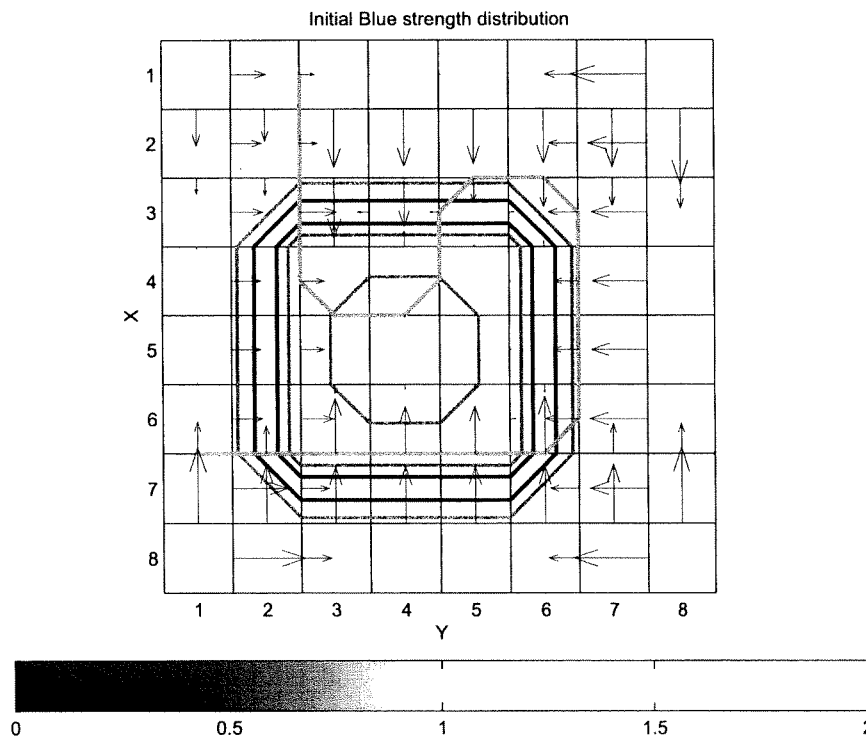


Figure 12.6: Initial Strength Distribution for Blue force. Arrows indicate magnitude of the velocity components across the boundaries.

Figures 12.8 and 12.9 show the final strength distributions for the blue and red forces. The arrows indicate the magnitudes of the velocity components as the respective assets of the two forces reached their final destinations.

The final total strength for the blue force was 20.5, while the final total strength for the red force was

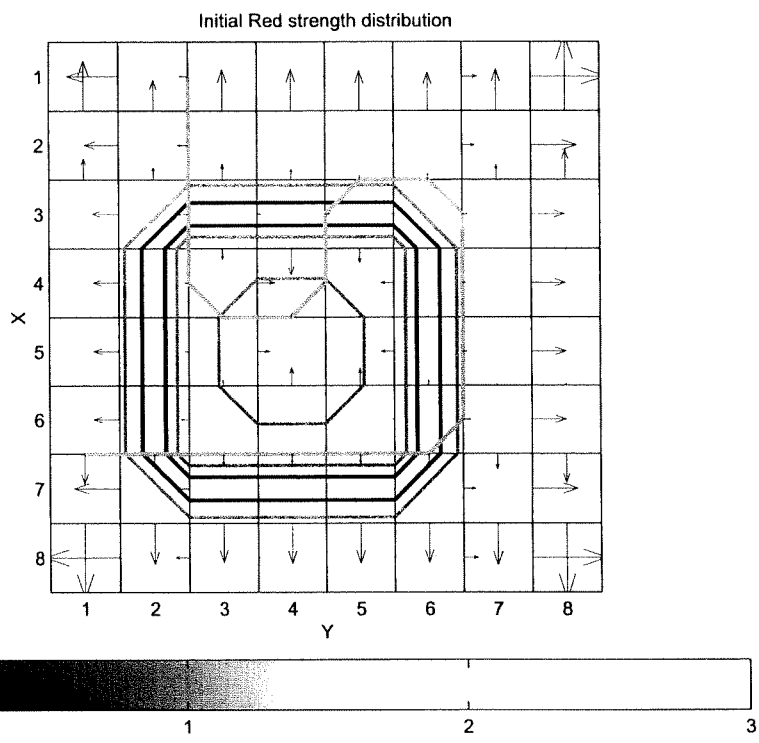


Figure 12.7: Initial Strength Distribution for Red force. Arrows indicate magnitude of the velocity components across the boundaries.

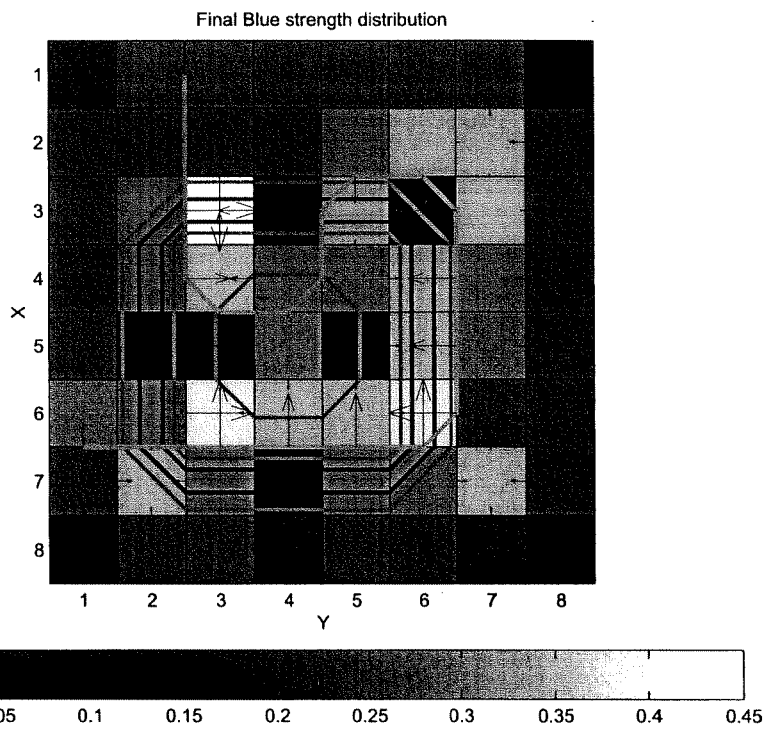


Figure 12.8: Final Strength Distribution for Blue force. Arrows indicate magnitude of the velocity components across the boundaries.

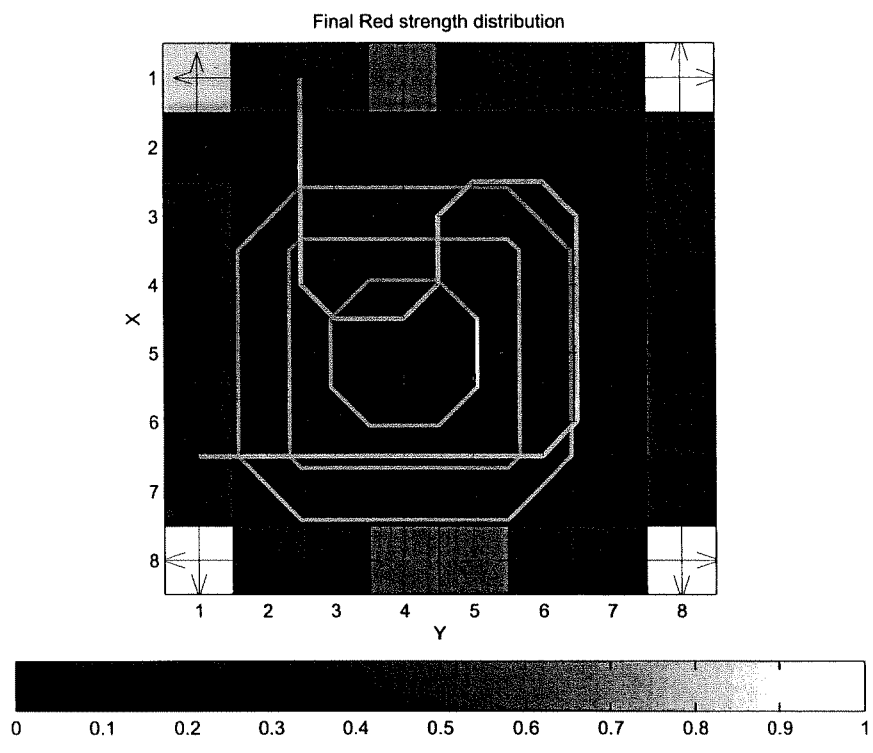


Figure 12.9: Final Strength Distribution for Red force. Arrows indicate magnitude of the velocity components across the boundaries.

29.5. Therefore, the red force conserved only 15.4% of its original strength, while the blue force conserved 16% of its own strength.

12.6 Analysis

Qualitatively speaking, we can say that, in this scenario, the superiority of the blue force in the attack range prevailed, allowing the blue force to keep the red force out of the most valuable cells in the middle of the board. Indeed, the red force was forced to retreat into the four corners of the game area where less valuable cells were to be found. It is also clear that different transportation costs assigned to different regions affected the way in which the blue and red forces adjusted their strength concentrations during the game. This can be seen in the fact that the red's concentration in the (1,1) corner is weaker than the other three corners because the terrain near (1,1) is harder to traverse.

12.7 Conclusions and Recommendations

It was difficult to find a scenario which was both interesting, in the sense that some significant amount of action occurred during the game, and in which a game theoretic solution could be found by the SLQ algorithm. In most cases in which a solution could be found, the velocity terms had to dominate the payoff function so that little movement of assets resulted. This was particularly the case when the two forces had an initial strength distribution concentrated in a small region of the game area. However, it is still not clear at this time what features are most critical in determining whether a given scenario has an SLQ solution or not.

With respect to computational complexity, the Game Flow solution engine executes relatively fast, using a combination of Matlab built-in functions (e.g., ODE solvers) and custom made C++ routines. For example, the CPU time required to run this experiment (8×8 grid) was 113.5 seconds. We also solved the same scenario, but using a 10×10 grid. This represents an increase of 60% in the size of its computational work. The algorithm converged and the results were similar to the ones reported here. The CPU time in this case was 273.6 seconds, which means a 141% increase over the original experiment. These results were obtained running the Game Flow program on a 800 MHz PC with 500 MB RAM. It should be stated, that the actual CPU time required to run different scenarios may vary considerably from these results.

In the current version of the Game Flow model, physical features of the theater itself can affect the dynamics in two ways: i) in the rate of attrition of the human and/or mechanical assets in the field; ii) in the energy cost associated with the movement of assets in the field. The last feature is not implemented directly in the the differential equations of the system. Instead, it is represented in the payoff functions of the two forces. A velocity reduction parameter, similar to an attrition parameter could be implemented in future versions of the Game Flow model.

Another characteristic of the current version of the Game Flow model is that physical features of the theater have no effect on the efficiency of attack. Hence, the enemy cannot hide behind a mountain range, for instance. Also the efficiency of attack functions can have a directionality associated with them before the beginning of a game, but these cannot be rotated or reoriented as the game evolves. This could be an important addition to enhance the strategic capabilities of the Game Flow model. A similar concept regarding shields, that would protect the assets positioned behind them, might conceivably be incorporated as well.

It is hoped that by adding complexity to the model, the class of interesting problems that can be solved with the Game Flow program would be enlarged.

Chapter 13

Experiment 13: Discrete Platform Dynamics

13.1 Executive Summary

In any type of battlefield, the loss of platforms is usually a stochastic discrete event over time. In JFACC simulations, however, the number of platforms has been modeled as a real number representing its probabilistic expectation. In other words, our game-theoretic controller based on an expected-value model needs to be tested on a more realistic plant, in which the numbers of platforms are integers. Our approach was to first develop a model such that the dynamics of the number of platforms is a stochastic discrete-event equation, i.e., in our new stochastic discrete-event model, the number of platforms in each unit is an integer. Hence, the number of platforms changes from 10 to 9 at one point and then on to eight platforms later based on the probability of kill. Moreover, the numbers of platforms vary differently for different runs due to random number generators, which control the time when an actual kill occurs. Using this new model, we conducted multiple runs and took an average. This average was then compared against the results based on the expected-value model. We concluded that our game-theoretical controller (based on the simpler expected-value model) performed just as well when tested on this more realistic stochastic discrete-event plant model as when tested on the expected-value plant model.

13.2 Introduction

In the JFACC simulations, the number of platforms is modeled as a real number representing its probabilistic expectation. Here we will investigate the effect of this assumption on the game-theoretic controller. First, in Section 13.3, we form a hypothesis. Then, in Section 13.4 we describe a stochastic discrete-event model, where the number of platforms in each unit is an integer and varies differently for different runs due to random number generation. Because of the randomness, we needed to make multiple runs and take an average in order to analyze the effectiveness of the game-theoretical controller. This experiment and the methods are fully described in Section 13.5. The average of multiple runs is compared against the expected value results and we form a conclusion in Section 13.6.

13.3 Hypothesis to Prove

The hypothesis is that we will not find any notable differences in the controller performance when the controller based on the expected-value model is applied to the more realistic stochastic discrete-event model.

13.4 Stochastic Discrete Model Description

Note: In the following derivation, we proceed with a generic unit without specifying Blue or Red since the model is identical for both teams. However, in its programming implementation, the equation for the Red and Blue forces are written separately.

As described in the report [1] on the Mission Dynamics Continuous-time Model (MDCM), the dynamics for the number η of platforms are given by

$$\dot{\eta} = -\lambda\eta, \quad (13.1)$$

Here on the Mission Dynamics Continuous-time Model (MDCM) $\dot{\eta} = \frac{d\eta}{dt}$ and λ is defined by

$$\lambda(t) \stackrel{\text{def}}{=} \rho P_k \phi \pi, \quad (13.2)$$

where ρ is the acquisition rate, P_k is the probability of kill, ϕ is a function dependent on the distance between the units, and π is the fire intensity.

To approximate the model, we use the forward difference approximation

$$\frac{\eta(t + \Delta t) - \eta(t)}{\Delta t} \approx \lambda(t)\eta(t).$$

Then the approximate number of platforms lost over a Δt time interval is

$$\eta(t + \Delta t) - \eta(t) \approx \lambda(t)\eta(t)\Delta t.$$

To take advantage of this approximation we must assume that Δt is small, furthermore we will want to choose Δt such that

$$\sup_t \{\lambda(t)\eta(t)\Delta t\} < 1.$$

With this assumption, we view $\lambda(t)\eta(t)\Delta t$ as the probability of losing a platform in the interval $[t, t + \Delta t]$. Now define R as a uniformly distributed random number between zero and one. Then if $R < \lambda(t)\eta(t)\Delta t$, a platform is lost in the interval of Δt . We can now define the stochastic discrete event dynamics for the number of platforms as

$$\eta(t + \Delta t) = \begin{cases} \eta(t) - 1 & \text{if } R < \lambda(t)\eta(t)\Delta t \\ \eta(t) & \text{else} \end{cases} \quad (13.3)$$

where R changes with each update of the discrete system, that is we compute a new random number R at each Δt update.

13.5 Experiment and Methods

To test the new model, we implement the discrete dynamics for the number of platforms into the plant of the MDCM model while the internal model for the controller remains MDCM. We call the new model with discrete dynamics: Mission Dynamics Continuous Model- Stochastic Discrete (MDCM-SD). The scenario used is the *cross11* scenario, which is summarized in Table 13.1. The weights in the table are for the quadratic cost function for the nonlinear game.

	<i>cross11</i>	
	Blue	Red
Number of Units	1.0	1.0
Number of Platforms	10.0	10.0
P_k	0.8	0.8
ρ	0.5	0.5
$\xi^{(1)}(0)(\text{km})$	80.0	50.0
$\xi^{(2)}(0)(\text{km})$	50.0	20.0
Weight: Distance to Target Cost	0.1	0.1
Weight: Running Platform Cost	0.01	3.0
Weight: Speed Cost	200.0	200.0
Weight: Terminal Platform Cost	0.0	0.0
Weight: Terminal Target Cost	0.0	0.0
Weight: Terminal Speed Cost	0.0	0.0

Table 13.1: Cross 11 Scenario description

Next, to show that the plant dynamics are indeed discrete, and the controller acts effectively, we conduct one sample run as illustrated in Figures 13.1-13.5. We can see in Figure 13.2 that the dynamics of MDCM-SD are in fact discrete. Through one run, we see that the Red dynamics for the number of platforms match closely with the MDCM model but the Blue dynamics for the number of platforms do not, and for both Blue and Red, the discrete dynamics lag behind the continuous dynamics. This can be explained by examining how we approximated the continuous model. Since the approximation requires a forward difference approximation, we will have a delay “reaction” of the discrete dynamics. This is because the current update of the discrete equation depends on the previous time (see Equation 13.3). Yet, as we decrease Δt , the lag should become less noticeable.

To understand how well the controller performs on the MDCM-SD model, compared to the MDCM model, we need to run the MDCM-SD simulation multiple times and take an average since the discrete dynamics depend on the generation of random numbers. We have simulated and compared the trajectories of MDCM-SD and MDCM over a hundred sample runs and over a range of update times, $\Delta t = 0.001, 0.01$ and 1.0 min. The comparisons are shown in Figures 13.6-13.20. Notice that, by taking the average over a hundred sample runs, we are producing an approximate continuous dynamic equation to the actual continuous dynamics.

In all the comparisons, and most importantly the dynamics for the number of platforms comparisons, we do not see much dependence on the update times Δt we chose. This is good since smaller update times will require more computation time, although the differences in computation time range on a magnitude of a few minutes.

13.6 Conclusion

We have shown that the game controller performs as well on the more realistic stochastic discrete-event battlefield plant as on the less realistic expected-value battlefield plant. Further experiments, using a less trivial scenario, are currently under way at Washington University and preliminary results have shown

that the controller performance remains unaffected; though we were unable to include a complete analysis at the time this report was written.

Bibliography

- [1] *Mission Dynamics Continuous-time Model: Version 3.0*, Washington University JFACC Team Internal Report, Department of Systems Science and Mathematics, Washington University, 2001.
- [2] *JFACC Experiment Report: Experiment 13, PMDM Model Validation*, Ruisheng Li, Hong Gao and İlker Tunay, Washington University JFACC Team Internal Report, Department of Systems Science and Mathematics, Washington University, 2001.

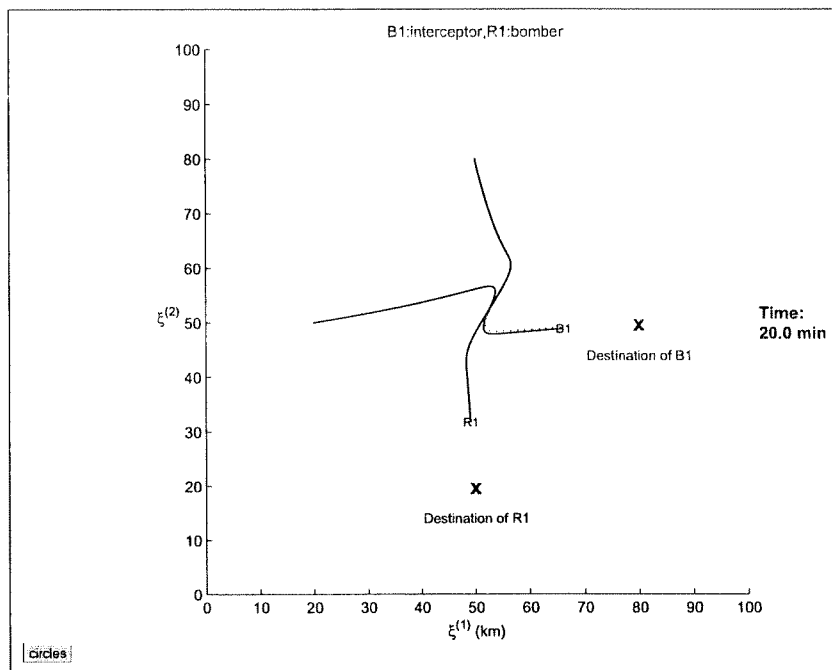


Figure 13.1: Comparison of MDCM (- -) and MDCM-SD (-) for Game Trajectories (One sample run).

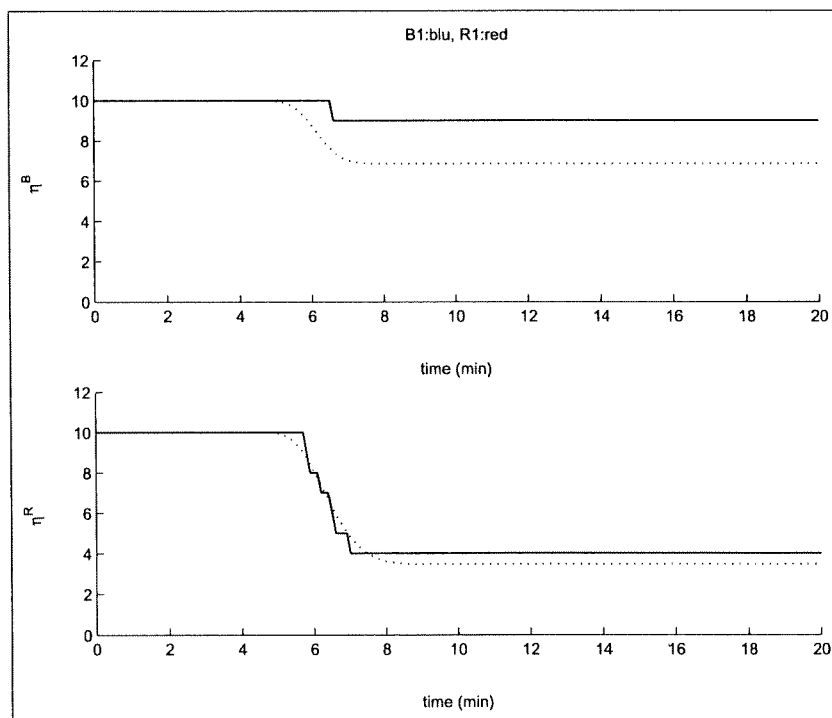


Figure 13.2: Comparison of MDCM (- -) and MDCM-SD (-) for the Number of Platforms (One sample run).

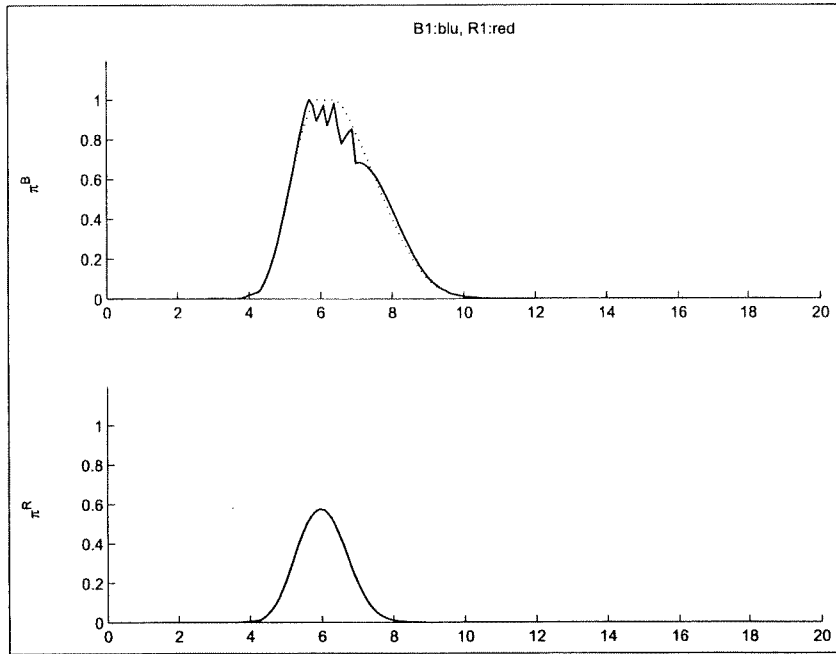


Figure 13.3: Comparison of MDCM (- -) and MDCM-SD (-) for Fire Intensities (One sample run).

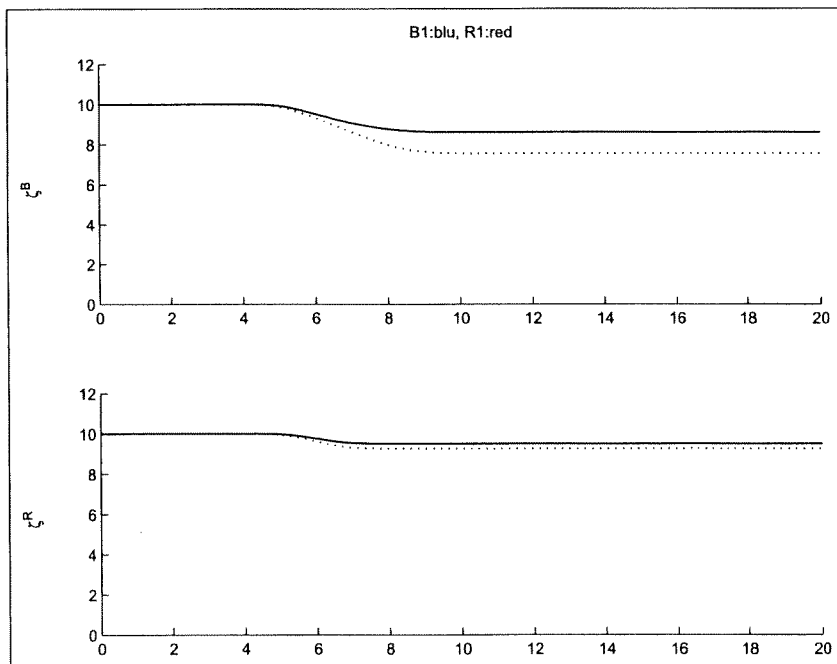


Figure 13.4: Comparison of MDCM (- -) and MDCM-SD (-) for Weapons Expenditures (One sample run).

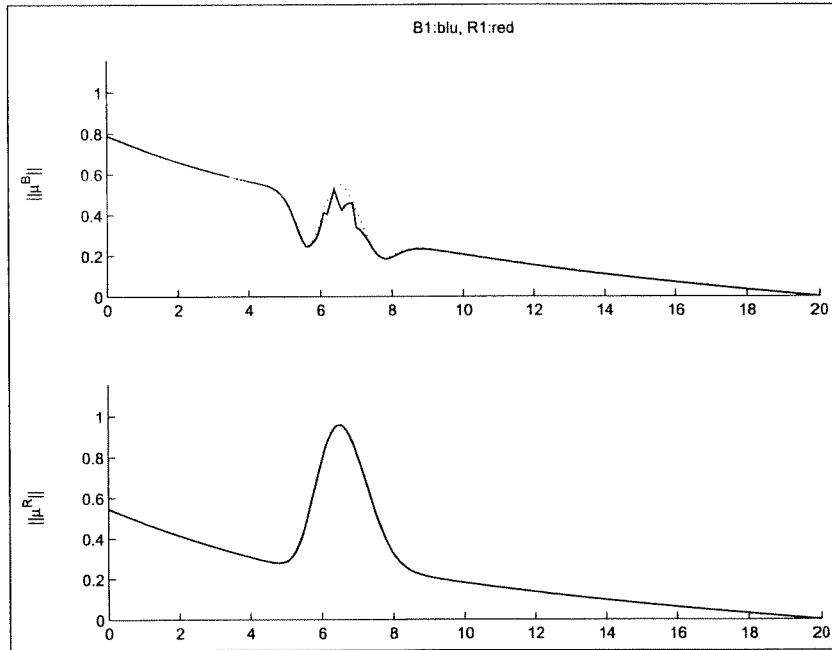


Figure 13.5: Comparison of MDCM (- -) and MDCM-SD (-) for Speed (One sample run).

	MDCM	MDCM-SD
$\eta^B(T)$	7.47	9
$\eta^R(T)$	4.18	4
Game Cost	-4545.5	-4631.0

Table 13.2: Summary of Results for One Sample Run with $\Delta t = 0.1$.

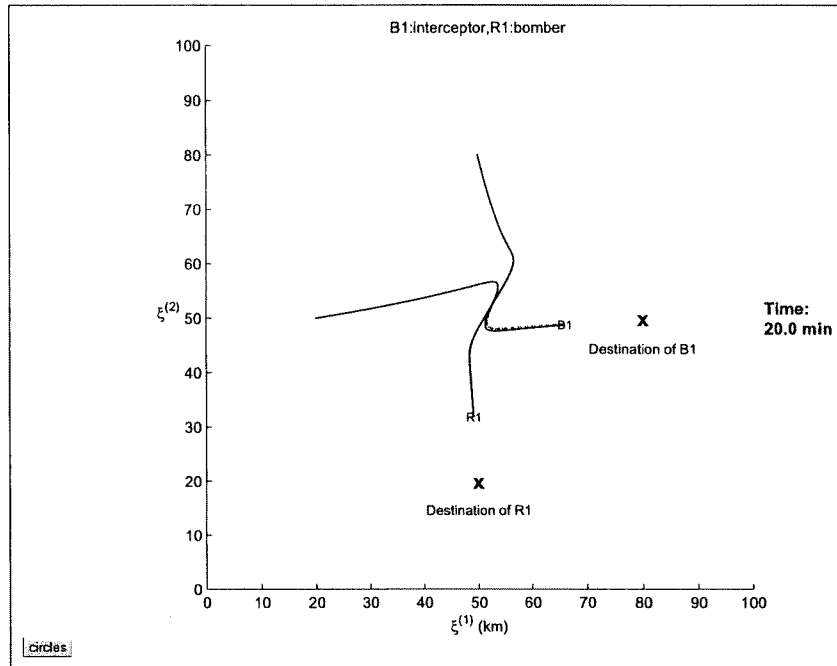


Figure 13.6: Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Game Trajectories ($\Delta t = 0.001$).

	MDCM	MDCM-SD
$\eta^B(T)$	7.47	7.87
$\eta^R(T)$	4.18	4.09
Game Cost	-4545.5	-4507.1

Table 13.3: Summary of Results Averaged over 100 Sample Runs for $\Delta t = 0.001$.

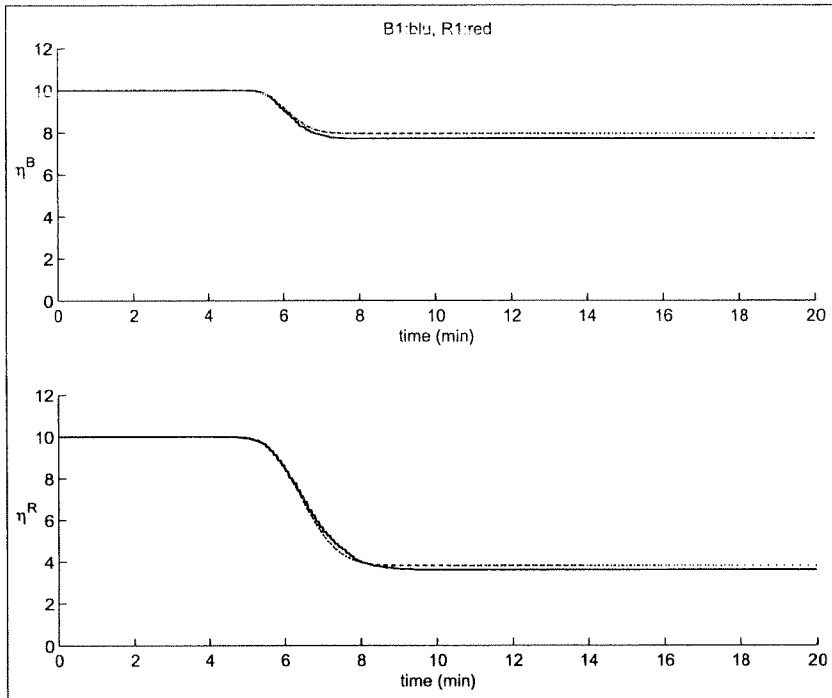


Figure 13.7: Comparison of MDCM (- -) and Averaged MDCM-SD (-) for the Number of Platforms ($\Delta t = 0.001$).

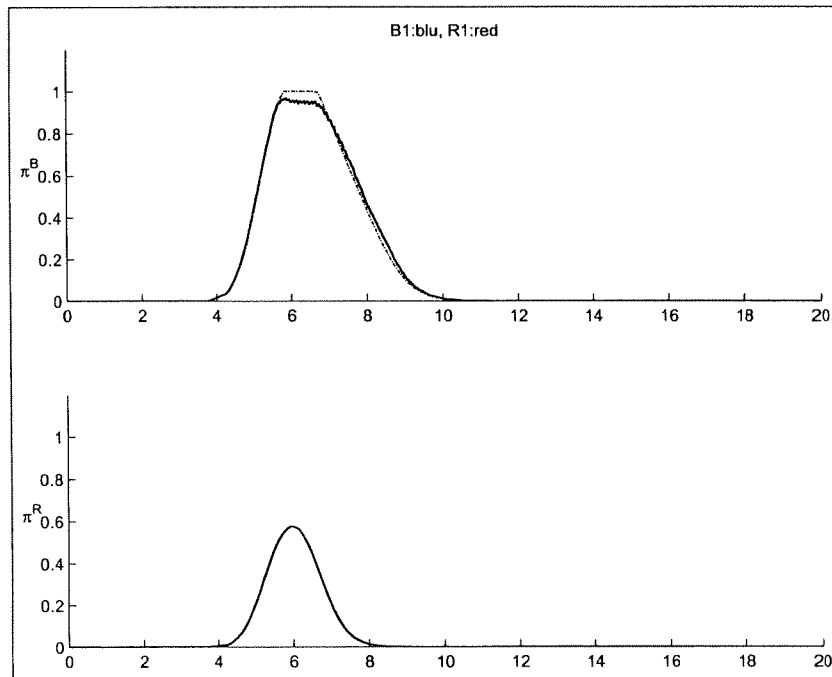


Figure 13.8: Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Fire Intensities ($\Delta t = 0.001$).

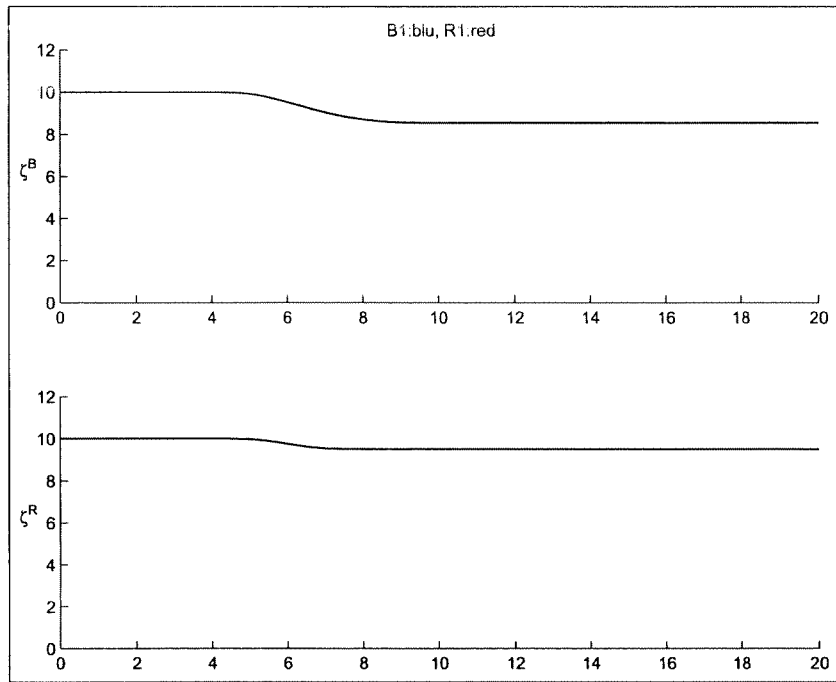


Figure 13.9: Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Weapons Expenditures ($\Delta t = 0.001$).

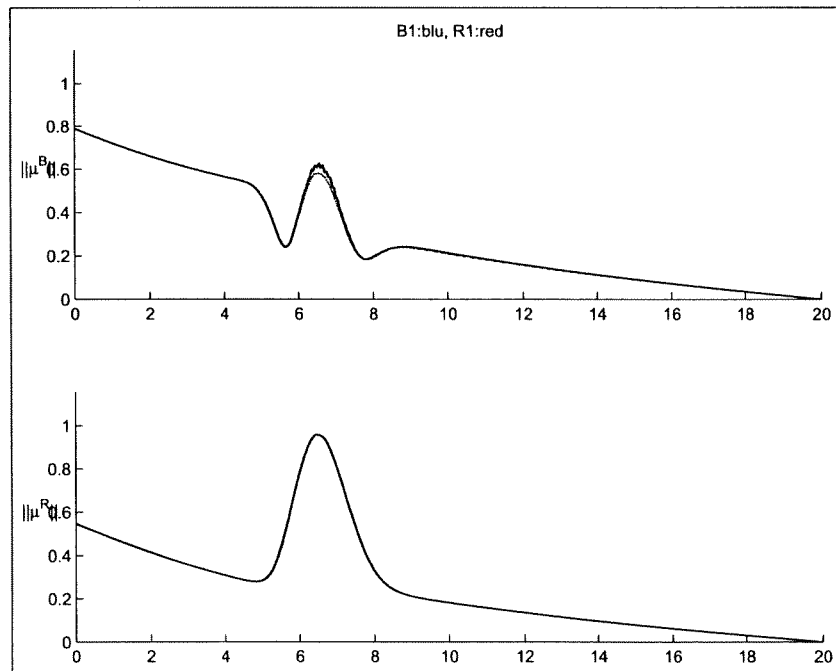


Figure 13.10: Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Speed ($\Delta t = 0.001$).

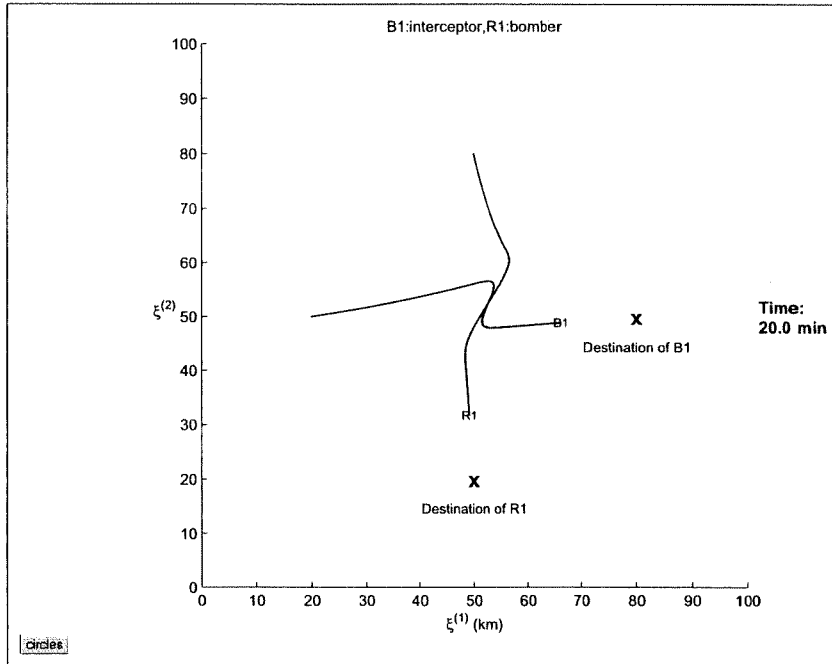


Figure 13.11: Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Game Trajectories ($\Delta t = 0.01$).

	MDCM	MDCM-SD
$\eta^B(T)$	7.47	7.78
$\eta^R(T)$	4.18	4.18
Game Cost	-4545.5	-4499.5

Table 13.4: Summary of Results Averaged over 100 Sample Runs for $\Delta t = 0.01$.

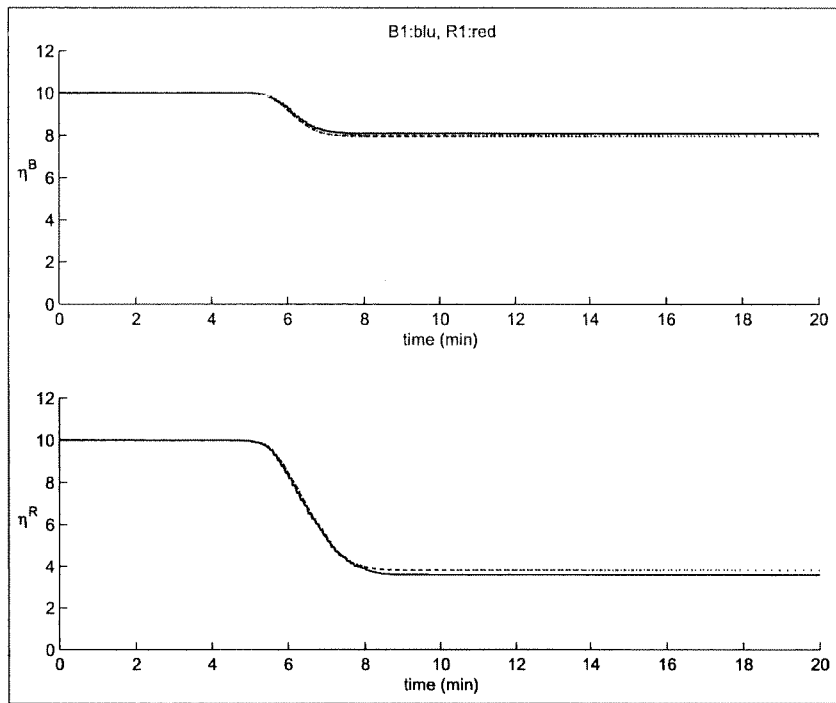


Figure 13.12: Comparison of MDCM (- -) and Averaged MDCM-SD (-) for the Number of Platforms ($\Delta t = 0.01$).

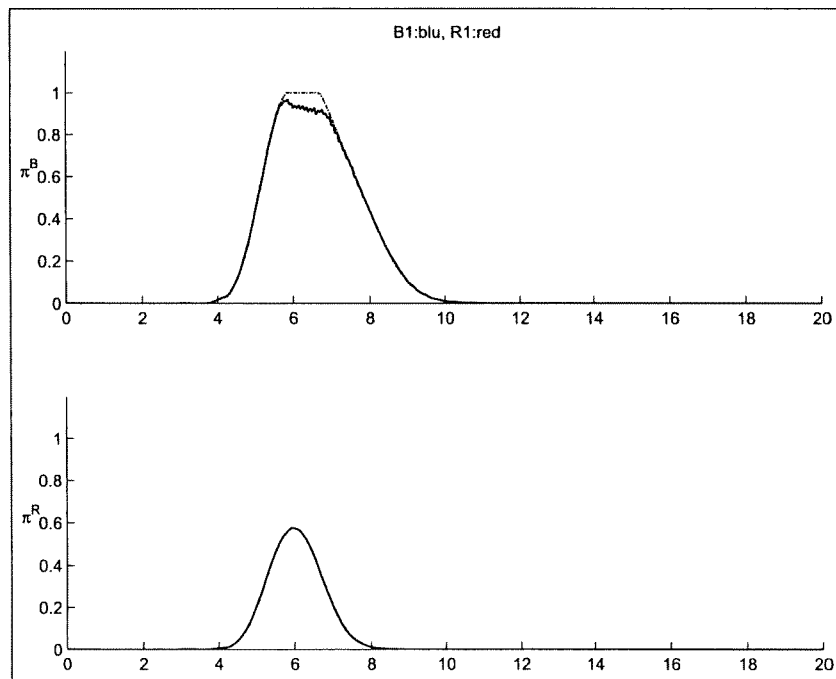


Figure 13.13: Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Fire Intensities ($\Delta t = 0.01$).

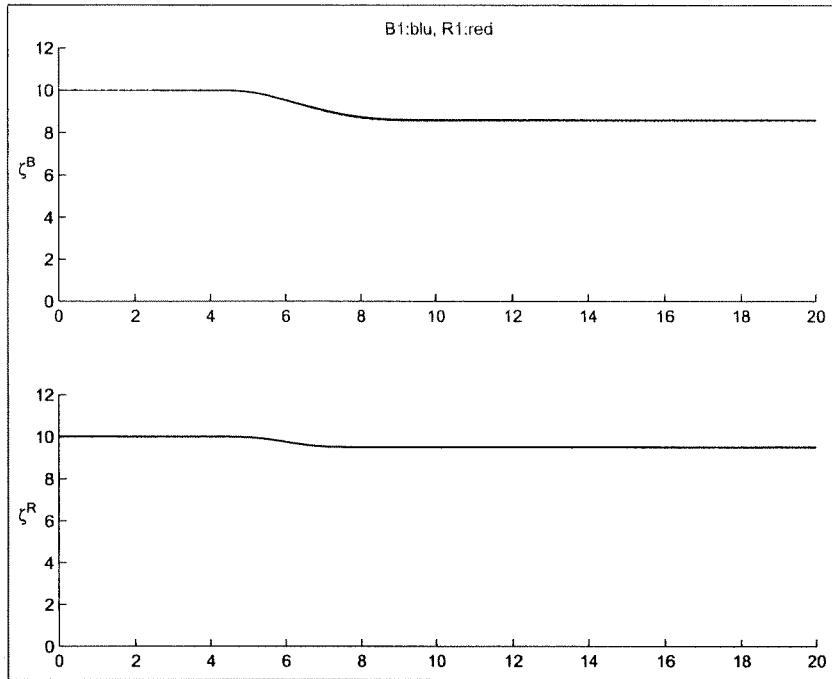


Figure 13.14: Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Weapons Expenditures ($\Delta t = 0.01$).

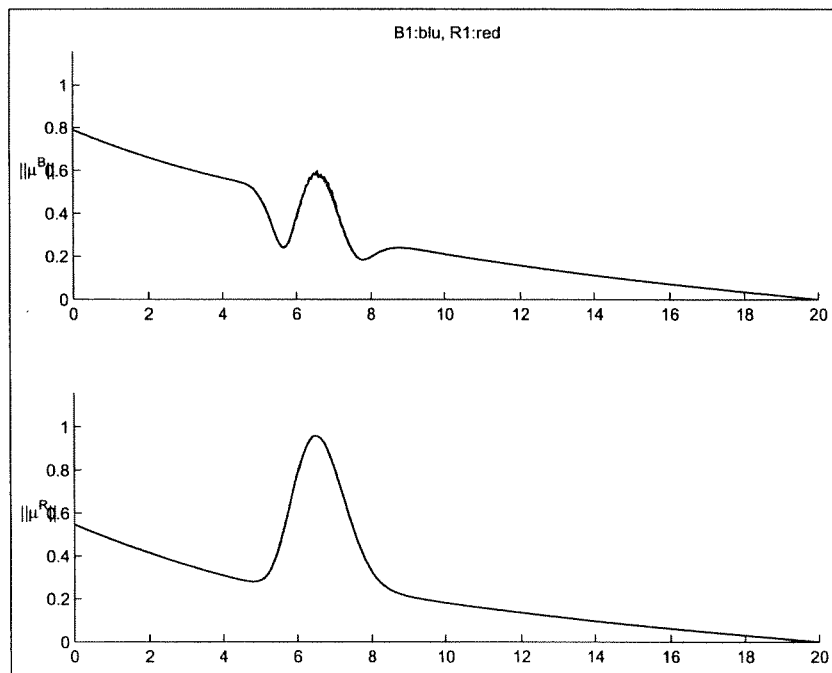


Figure 13.15: Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Speed ($\Delta t = 0.01$).

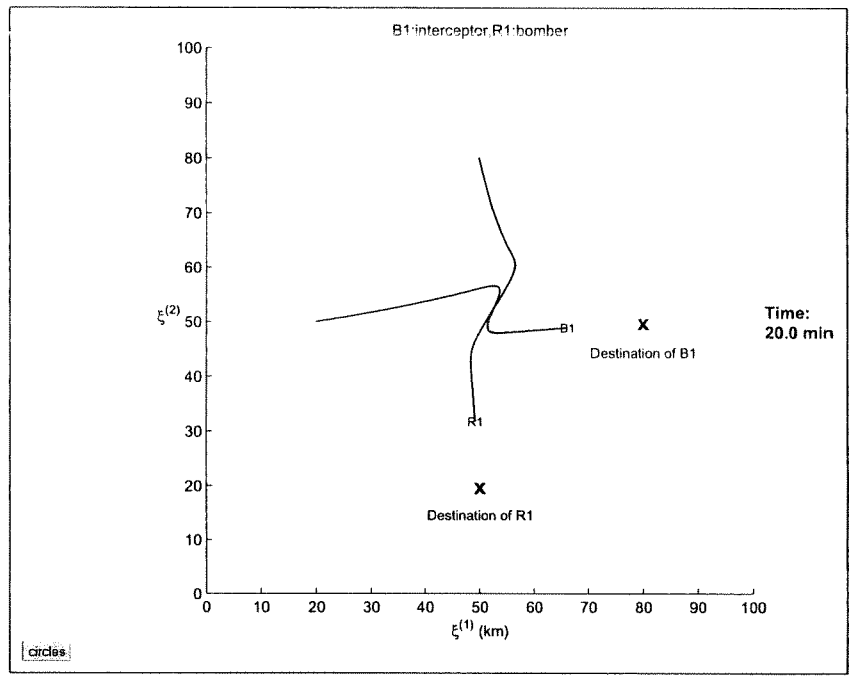


Figure 13.16: Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Game Trajectories ($\Delta t = 0.1$).

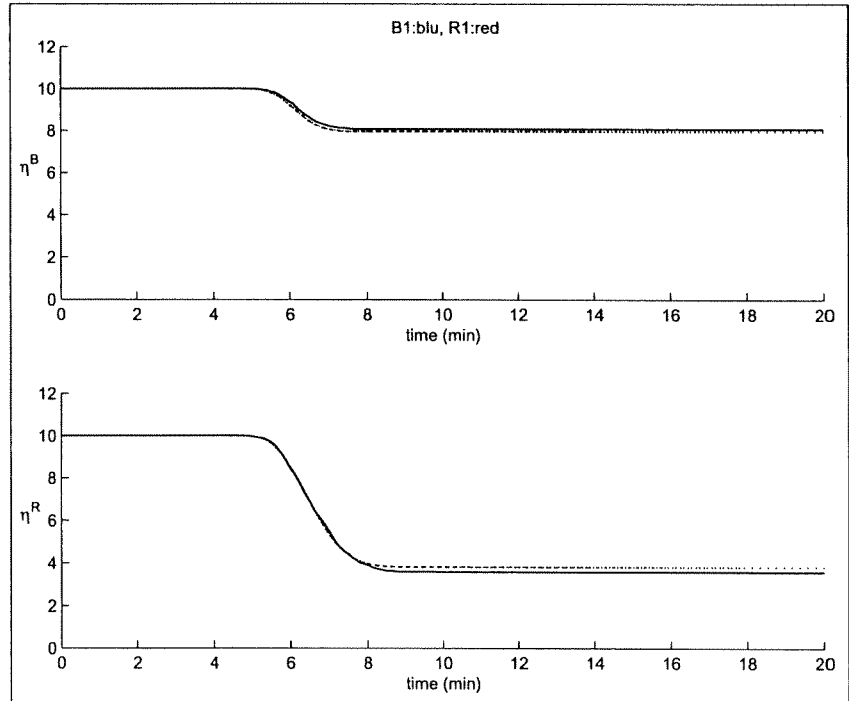


Figure 13.17: Comparison of MDCM (- -) and Averaged MDCM-SD (-) for the Number of Platforms ($\Delta t = 0.1$).

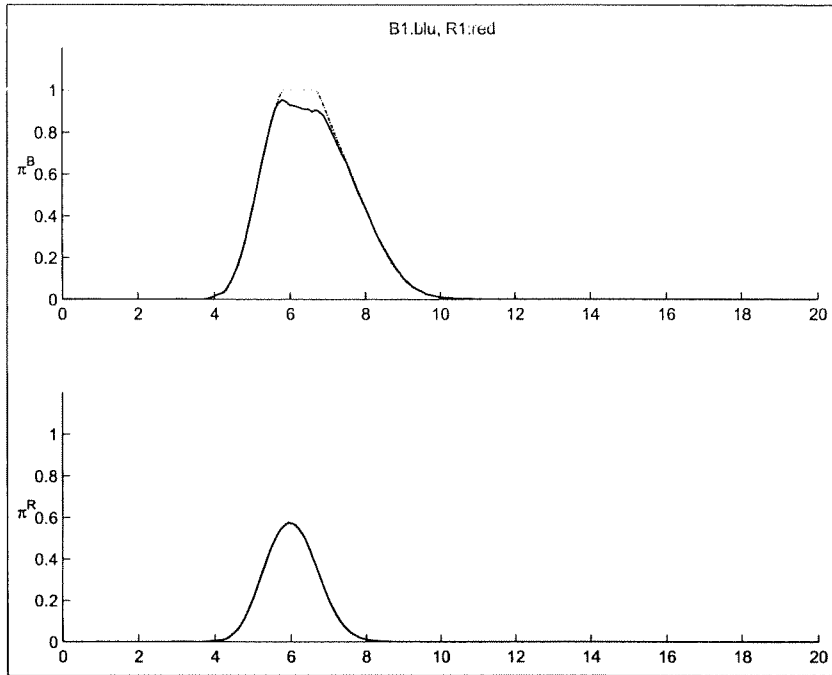


Figure 13.18: Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Fire Intensities ($\Delta t = 0.1$).

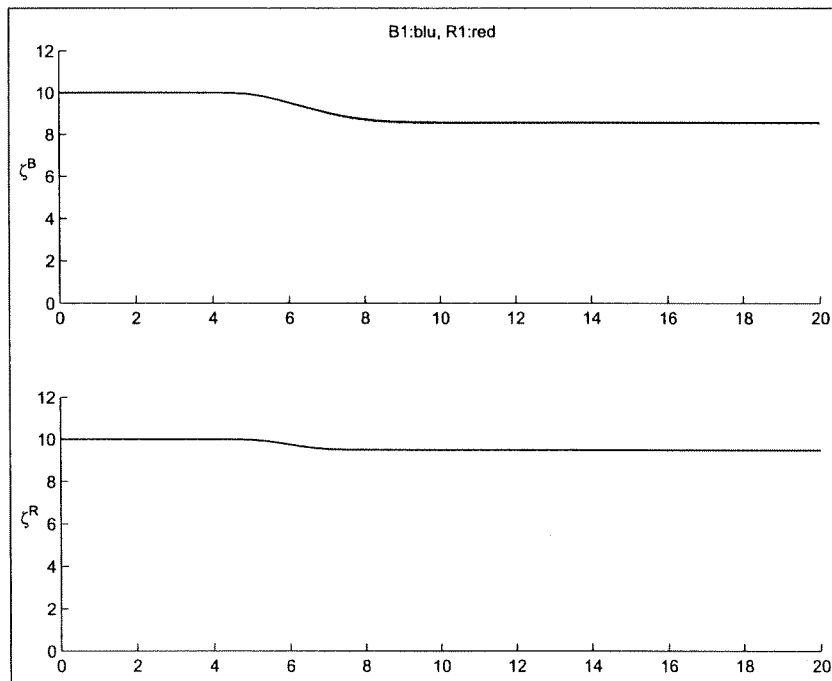


Figure 13.19: Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Weapons Expenditures ($\Delta t = 0.1$).

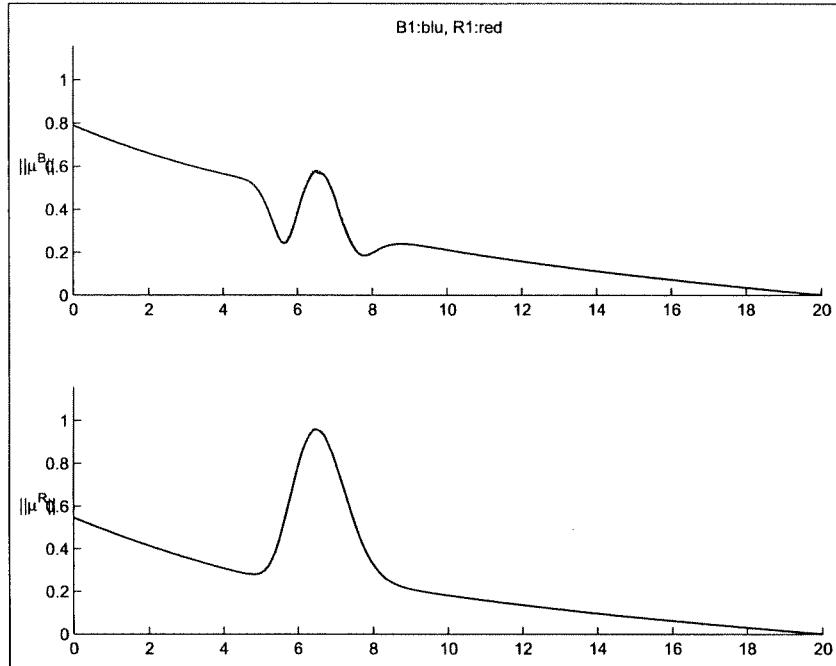


Figure 13.20: Comparison of MDCM (- -) and Averaged MDCM-SD (-) for Speed ($\Delta t = 0.1$).

	MDCM	MDCM-SD
$\eta^B(T)$	7.47	7.85
$\eta^R(T)$	4.18	3.92
Game Cost	-4545.5	-4516.1

Table 13.5: Summary of Results Averaged over 100 Sample Runs for $\Delta t = 0.1$.

Chapter 14

Experiment 14: Non-linear Detector for the Fully Non-Linear Model

14.1 Executive Summary

In Chapters 9 and 10 we have reported the results of experiments performed to test the effectiveness of a “game-theoretic-optimal” detection filter to process noise-corrupted observations of the battlefield. In those series of experiments, a bilinear approximation of the non-linear model of the battlefield was considered and the filter was designed accordingly. When the fully non-linear model of the battlefield is considered, a different (non-linear) detection filter must be designed. The purpose of this Chapter is to present the experimental results concerning the non-linear filter and to compare them with those obtained by using the detection filter designed on the basis of the bilinear model of the battlefield. For the sake of simplicity, the case of noise-free measurements will be considered in this series of experiments.

14.2 Purpose of the Experiment

This section of the report describes experiments on detection and isolation of multiple enemy actions in a battlefield. The mathematical description of the battlefield used here is the one introduced in the Appendix 1.12 of Chapter 1 and there validated under the “uncoordinated target selection, independent target acquisition” assumption. We consider the case in which two opposing forces are present in the theater of operations, the Blue force (the “friends”) and the Red force (the “enemies”). Each force consists of two units and each unit consists of a number of platforms whose evolution in time is described by a first order nonlinear differential equation and depends on the “actions” which the opposing units are performing against the unit in question. If any “new” action is performed by any of the opposing units, this affects the evolution of the number of platforms of the other force’s units. Each unit’s location is represented by a point on the plane and its motion is described by an ordinary differential equation depending on speed control inputs. Denoted with $\xi_i^R = \text{col}(\xi_{xi}^R, \xi_{yi}^R) \in \mathbb{R}^2$ and $\xi_i^B = \text{col}(\xi_{xi}^B, \xi_{yi}^B) \in \mathbb{R}^2$, with $i = 1, 2$, the position vectors of the Red and – respectively – Blue units on the plane, the equations of the motion in question are ¹

$$\begin{aligned}\frac{d}{dt}\xi_{xi}^R(t) &= a_i^R \mu_{xi}^R(t) \\ \frac{d}{dt}\xi_{yi}^R(t) &= a_i^R \mu_{yi}^R(t) \\ \frac{d}{dt}\xi_{xi}^B(t) &= a_i^B \mu_{xi}^B(t) \\ \frac{d}{dt}\xi_{yi}^B(t) &= a_i^B \mu_{yi}^B(t),\end{aligned}\tag{14.1}$$

¹The reader is referred to Chapter 1 for a definition of the parameters appearing in the equations (14.1) and (14.2).

where $\mu_i^R = \text{col}(\mu_{x_i}^R, \mu_{y_i}^R) \in \mathbb{R}^2$ and $\mu_i^B = \text{col}(\mu_{x_i}^B, \mu_{y_i}^B) \in \mathbb{R}^2$ are the speed control inputs of the Red and Blue units, respectively. Letting η_i^R and η_i^B , with $i = 1, 2$, denote the number of platforms of the i -th Red and – respectively – i -th Blue unit, the model of evolution of the number of platforms is a four-dimensional nonlinear system described by two pairs of equations of the form (cf. Chapter 1, formula (1.26))

$$\begin{aligned} \frac{d}{dt}\eta_i^R(t) &= -\eta_i^R(t)(\alpha^B + \sum_{j=1}^2 \eta_j^B(t)\rho_{ji}^B P_{ji}^B \psi_{ji}^B(|\xi_i^R(t) - \xi_j^B(t)|)\pi_{ji}^B(t)) \\ \frac{d}{dt}\eta_i^B(t) &= -\eta_i^B(t)(\alpha^R + \sum_{j=1}^2 \eta_j^R(t)\rho_{ji}^R P_{ji}^R \psi_{ji}^R(|\xi_i^B(t) - \xi_j^R(t)|)\pi_{ji}^R(t)) . \end{aligned} \quad (14.2)$$

In these equations, $\pi_{ji}^R(\cdot)$ and $\pi_{ji}^B(\cdot)$ are (independent) input variables representing the “level of engagement” of the j -th Red unit with the i -th Blue unit and – respectively – of the j -th Blue unit with the i -th Red unit. For convenience, we suppose in all our experiments that

$$\pi_{11}^R(t) = \pi_{12}^R(t) =: \pi_1^R(t), \quad \pi_{21}^R(t) = \pi_{22}^R(t) =: \pi_2^R(t) .$$

This means, in the terminology of [1], page 2, that we allow the “unique target constraint” to be violated (for the Red units only). In spite of what has been assumed for the experiments which have been reported in Chapters 9 and 10, in this series of experiments it will be considered the case in which the effect of the action performed by a unit on the number of platforms of an opposing unit depends on a measure of the distance between the two units through the function ψ . Following Chapter 1, we take as ψ an exponential function which depends on the measure r of the distance between the two units, namely

$$\psi(r) = \exp^{-r/r_0} ,$$

with r_0 a suitable scalar parameter. If ξ_i, ξ_j are the vectors denoting the position of two units, their distance r is chosen equal to the quantity $|\xi_i - \xi_j|_\infty = \max(|\xi_{ix} - \xi_{jx}|, |\xi_{iy} - \xi_{jy}|)$, which is the ∞ -norm of the vector $\xi_i - \xi_j$. For simplicity, we drop henceforth the subscript ∞ to denote such norm.

The basic problem addressed in our series of experiments on the design of filters for the detection of enemy actions is the following one: *we monitor only the position, the number of platforms and speed control inputs of the two Blue units* (i.e. we measure only the values of the four state variables $\xi_1^B, \xi_2^B, \eta_1^B, \eta_2^B$ and of the inputs $\mu_{x_i}^B, \mu_{y_i}^B$) and *we want to detect the occurrence of an “engagement action” from either one of the two Red units* (i.e. we want to detect when either one of the two input signals $\pi_i^R(\cdot)$ has become nonzero). Implicit in this is the assumption that the four other state variables $\xi_1^R, \xi_2^R, \eta_1^R, \eta_2^R$ (number of platforms of the two enemy units) as well as all the input variables $\pi_{ij}^B, i, j = 1, 2, \mu_{x_i}^R, \mu_{y_i}^R$ and $\pi_i^R, i = 1, 2$, are not monitored. The purpose of the detection process is precisely the determination of when either π_1^R or π_2^R has become nonzero, without having it directly measured.

14.3 Hypothesis to Prove or Disprove

Let us consider the systems (14.1) and (14.2), where, for the sake of notational simplicity, the latter is rewritten as

$$\begin{aligned} \frac{d}{dt}\eta_1^R(t) &= -\alpha^B \eta_1^R(t) - \gamma_{11}^B \eta_1^B(t) \psi(|\xi_1^R(t) - \xi_1^B(t)|) \pi_{11}^B(t) - \gamma_{12}^B \eta_2^B(t) \psi(|\xi_1^R(t) - \xi_2^B(t)|) \pi_{12}^B(t) \\ \frac{d}{dt}\eta_2^R(t) &= -\alpha^B \eta_2^R(t) - \gamma_{21}^B \eta_1^B(t) \psi(|\xi_2^R(t) - \xi_1^B(t)|) \pi_{21}^B(t) - \gamma_{22}^B \eta_2^B(t) \psi(|\xi_2^R(t) - \xi_2^B(t)|) \pi_{22}^B(t) \\ \frac{d}{dt}\eta_1^B(t) &= -\alpha^R \eta_1^B(t) - \gamma_{11}^R \eta_1^R(t) \psi(|\xi_1^B(t) - \xi_1^R(t)|) \pi_{11}^R(t) - \gamma_{12}^R \eta_2^R(t) \psi(|\xi_1^B(t) - \xi_2^R(t)|) \pi_{12}^R(t) \\ \frac{d}{dt}\eta_2^B(t) &= -\alpha^R \eta_2^B(t) - \gamma_{21}^R \eta_1^R(t) \psi(|\xi_2^B(t) - \xi_1^R(t)|) \pi_{21}^R(t) - \gamma_{22}^R \eta_2^R(t) \psi(|\xi_2^B(t) - \xi_2^R(t)|) \pi_{22}^R(t) , \end{aligned} \quad (14.3)$$

with $\gamma_{i,j}^R, \gamma_{i,j}^B, i, j = 1, 2$, suitable parameters. As in the equations above the two actions to detect π_1^R and π_2^R appear multiplied by terms which depend on the un-measured functions $\eta_1^R, \eta_2^R, \xi_1^R$ and ξ_2^R , the main

challenge in this problem is to distinguish not only the two actions π_1^R, π_2^R from each other, but also from the “disturbance” action of $\eta_1^R, \eta_2^R, \xi_1^R$ and ξ_2^R . In particular, according to the model in (14.3), the effect of an “engagement action” by a unit on the evolution of the number of platforms of an opposing unit will depend on their positions. Since the number of platforms of the Blue units is processed by the filter, its capability to detect and isolate actions will be related to the location of the units. We fix geographical areas for the Red and the Blue units in the theater of operation, and assume that the units are moving in those zones. In this assumption, we design a detection filter following the methods of [2], [3]. This filter *receives as inputs* the four observed variables $\xi_1^B, \xi_2^B, \eta_1^B, \eta_2^B$ and the measured inputs $\mu_{x_i}^B, \mu_{y_i}^B, i = 1, 2$, and *generates as outputs* two signals r_1, r_2 , called *performance signals* (typically known also with the name of *residuals*), in such a way that $r_i(t)$ is zero if the Red unit i is not engaged with the Blue units at time t (i.e. if $\pi_i^R(t) = 0$), and that $r_i(t)$ is nonzero if the Red unit i is engaged with the Blue units (i.e. if $\pi_i^R(t) \neq 0$), no matter what the locations of the four units in the assigned areas are. Specifically, this filter is modeled by equations of the form

$$\begin{aligned}
\dot{\hat{\eta}}_1(t) &= -\alpha^R \hat{\eta}_1(t) - \frac{\gamma_{12}^R}{\gamma_{22}^R} \frac{d}{dt} \left(\frac{\psi(|\xi_1^B(t)|)}{\psi(|\xi_2^B(t)|)} \right) \eta_2^B(t) + g_1(\eta_1^B(t) - \frac{\gamma_{12}^R}{\gamma_{22}^R} \frac{\psi(|\xi_1^B(t)|)}{\psi(|\xi_2^B(t)|)} \eta_2^B(t) - \hat{\eta}_1(t)), \\
\dot{\hat{\eta}}_2(t) &= -\alpha^R \hat{\eta}_2(t) - \frac{\gamma_{21}^R}{\gamma_{11}^R} \frac{d}{dt} \left(\frac{\psi(|\xi_2^B(t)|)}{\psi(|\xi_1^B(t)|)} \right) \eta_1^B(t) + g_2(\eta_2^B(t) - \frac{\gamma_{21}^R}{\gamma_{11}^R} \frac{\psi(|\xi_2^B(t)|)}{\psi(|\xi_1^B(t)|)} \eta_1^B(t) - \hat{\eta}_2(t)), \\
r_1(t) &= \eta_1^B(t) - \frac{\gamma_{12}^R}{\gamma_{22}^R} \frac{\psi(|\xi_1^B(t)|)}{\psi(|\xi_2^B(t)|)} \eta_2^B(t) - \hat{\eta}_1(t), \\
r_2(t) &= \eta_2^B(t) - \frac{\gamma_{21}^R}{\gamma_{11}^R} \frac{\psi(|\xi_2^B(t)|)}{\psi(|\xi_1^B(t)|)} \eta_1^B(t) - \hat{\eta}_2(t).
\end{aligned} \tag{14.4}$$

These equations contain terms which depend on the positions of the Blue units and on their rate of change, which are quantities available for measurements. In order to compare these equations with those which describe the *linear* filter (9.3) (or 10.3) in Chapters 9 (or 10), one can observe that the second term on the right-hand side of the first two equations in (14.4) is taken identically zero in the linear filter, while the ratio $\psi(|\xi_1^B|)/\psi(|\xi_2^B|)$ (or $\psi(|\xi_2^B|)/\psi(|\xi_1^B|)$) is taken equal to 1. The parameters g_1, g_2 are “gain parameters” to be designed. In the problem considered in Chapter 9 (or 10) the design of g_1 and g_2 was critical in order to obtain a filter which was able to *selectively reduce* the effect of the measurement noise while not attenuating the signal associated with the action to detect. Since we consider the case of noise-free measurements for this series of experiments, it is enough to design g_1 and g_2 so as to guarantee the stability of the filter.

14.4 Experiment Setup

The equations which define filter (14.4) depend on the location of the Blue Units. To test the effectiveness of the detection filter we fix geographical areas in the theater of operations in which the motion of the Blue and Red Units can evolve (see Figure 14.1). We consider the case in which the two Red Units are allowed to move in the red-dashed area in Figure 14.1, whereas the two Blue Units can evolve in the blue-dashed area. We note explicitly that, although the region where the Red units are allowed to move is known, we do *not* know the positions ξ_1^R and ξ_2^R of the two Red units. The four units will evolve along trajectories confined in the areas introduced before and according to the law (14.1). An example of such trajectories is depicted in Figure 14.2. The evolution of the number of platforms is modeled as in equation (14.2). The inputs variables, representing the level of engagement of the battling units, are functions of time which vary with different scenarios. For instance, in the first experiment we consider, the two levels of engagement of the Red units 1 and 2 versus the Blue units vary with time as shown in Figure 14.3, where “Action 1” represents the level of engagement of Red unit 1 and “Action 2” represents the level of engagement of Red unit 2. Note that the first action occurs at $t = 40$ units of time, whereas the second action takes place at $t = 25$ units of time and that the first action takes place while the first action is still

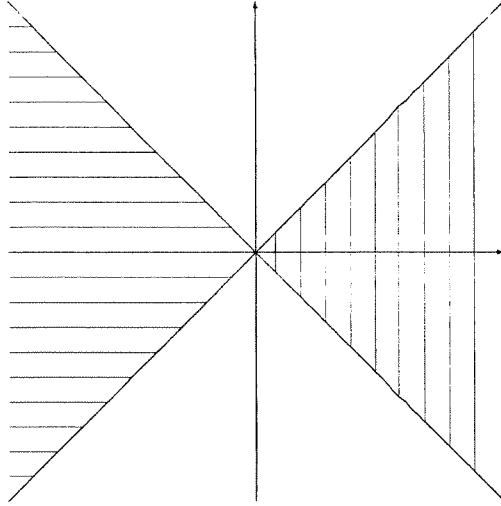


Figure 14.1: Geographical areas in which Red and Blue units can evolve.

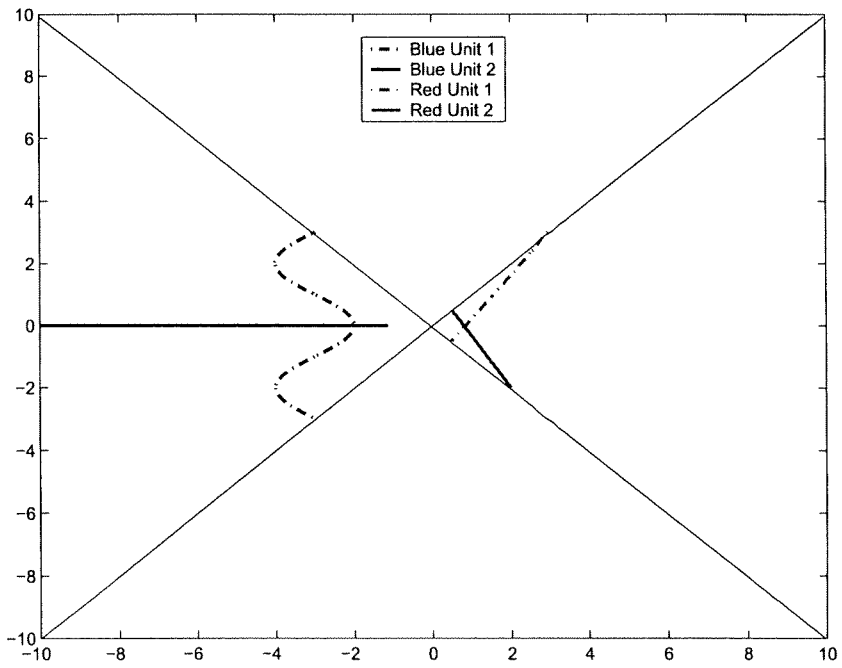


Figure 14.2: Trajectories for the Red and Blue units in Experiments 1 and 2.

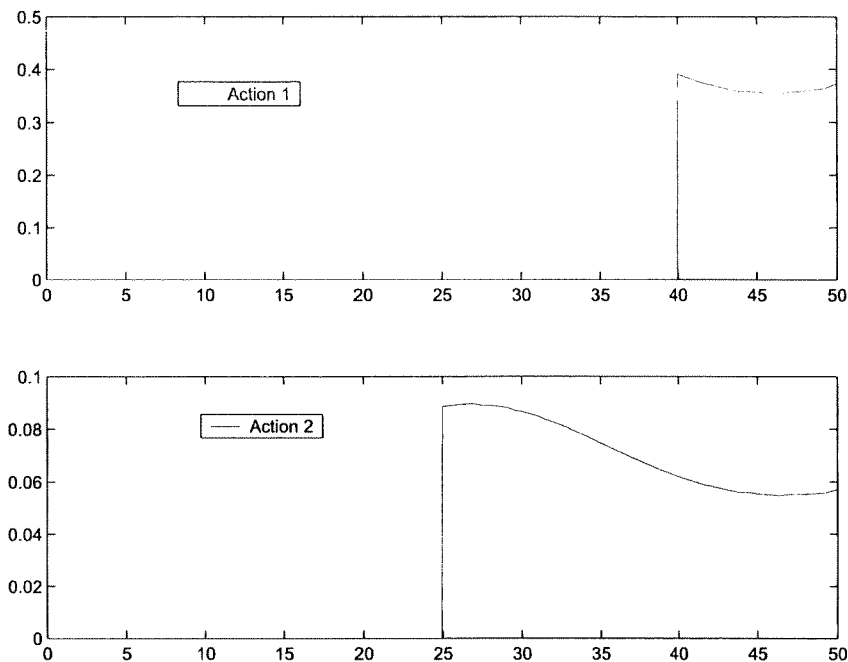


Figure 14.3: Engagement actions of the Red units versus the Blue units in Experiment 1.

occurring. The corresponding evolution in time of the number of Red and Blue platforms for each unit is plotted in Figure 14.4. We remind that the number of platforms of the Blue units is measured and fed into the detection filter, along with the measurement of their positions and speed control inputs. The filter generates the two outputs r_1, r_2 according to the equations (14.4) which must reveal the occurrence of Action 1 and, respectively, Action 2.

14.5 Example of Experiment

For the first experiment (henceforth, referred as “Experiment 1”), we consider the scenario in which the four units present in the battlefield are moving following the trajectories in Figure 14.2. The two Blue units are subject to the engagement actions depicted in Figure 14.3 due to the Red units. The measured number of platforms of the Blue units - which are processed by the detection filter - evolves according to the time behaviour in the plot on the left side of Figure 14.4. Although a change in the profile of the two graphs can be noticed at time $t = 25$ units of time and $t = 40$ units of time - denoting an increased level of engagement due to the action of the Red units - it cannot be inferred from these graphs which one of the two units is actually increasing its level of engagement versus the opposing units. As a matter of fact, only the outputs of the detection filter can clearly reveal the occurrence of the two Red actions, by distinguishing both of them. Figure 14.5 shows the time profile of the action to detect (Action 1) and of the Performance Signal 1 generated by the detector (r_1). It is seen that the performance signal decays to zero after a transient behavior, does not “react” to the occurrence of Action 2, and becomes evidently nonzero only when Action 1 occurs. In this way, it is possible to infer the occurrence of Action 1 without confusing it with the occurrence of Action 2. Analogously, Figure 14.6 shows how the Performance Signal 2 allows to detect the occurrence of Action 2. For the sake of completeness, the evolution of the internal states of the detection filter is reported in Figure 14.7. In order to assess the behaviour of the non-linear filter, the time profiles of the two performance signals generated by the *linear* detection filter are illustrated in the next two figures. Namely, the responses of the Performance Signal 1 and 2

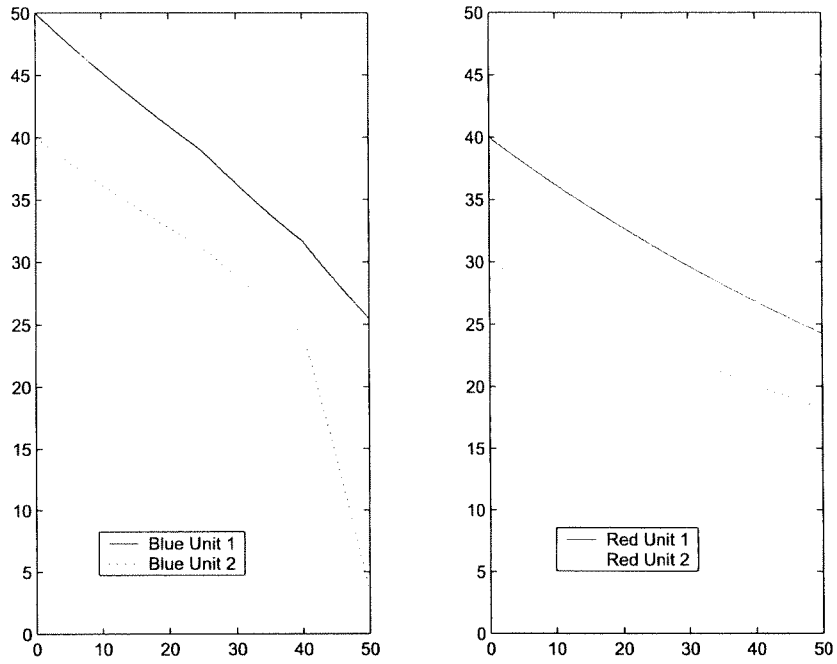


Figure 14.4: Evolution in time of the number of platforms in the Blue and Red Units in Experiment 1.

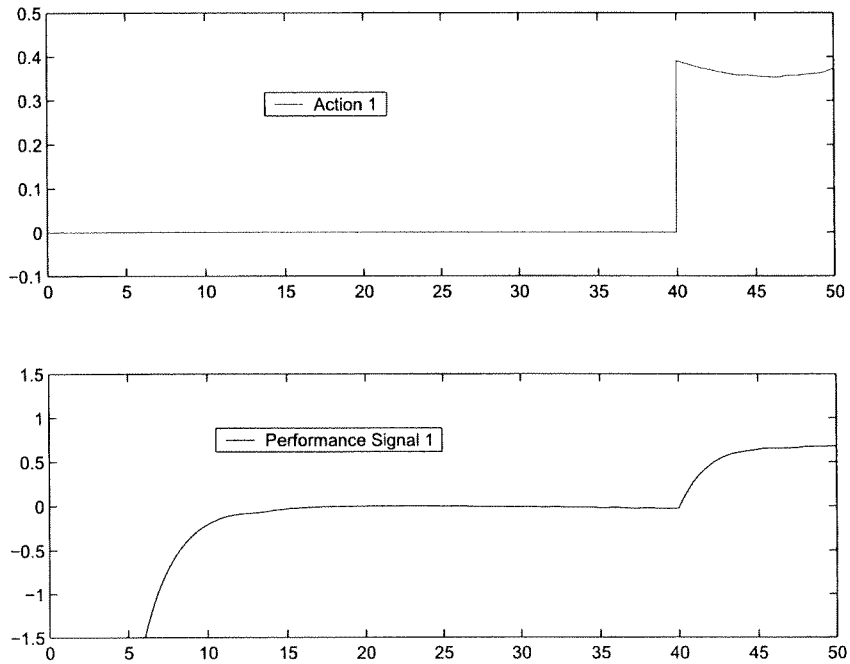


Figure 14.5: Action 1 and Performance Signal 1 in the Experiment 1.

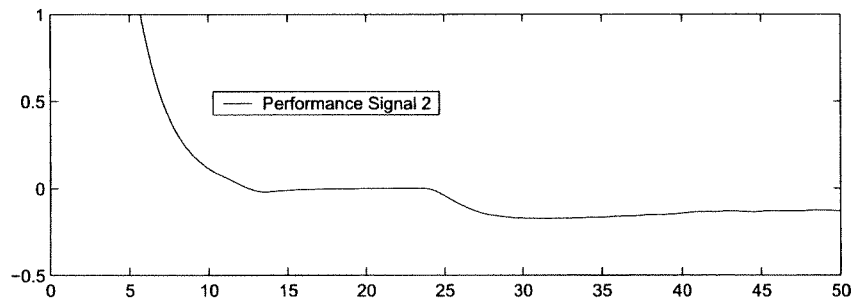
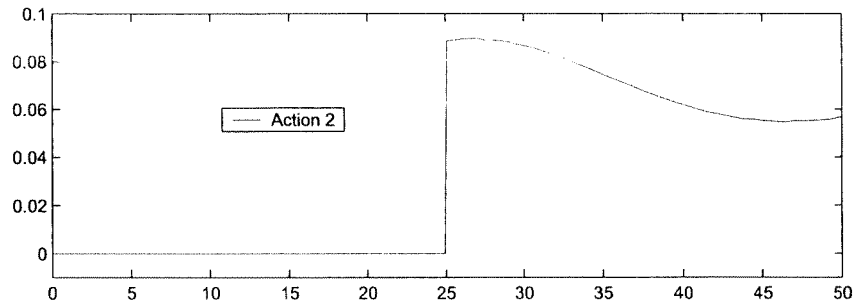


Figure 14.6: Action 2 and Performance Signal 2 in the Experiment 1.

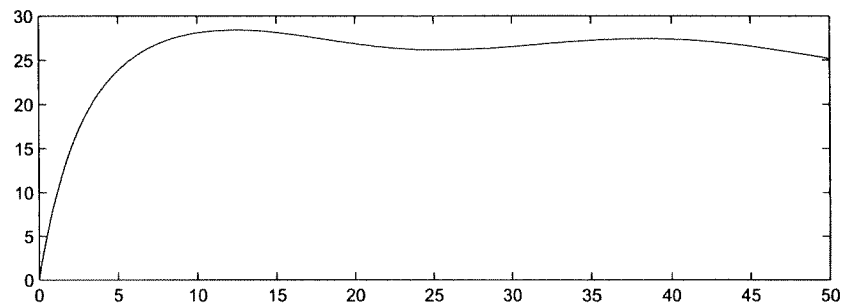
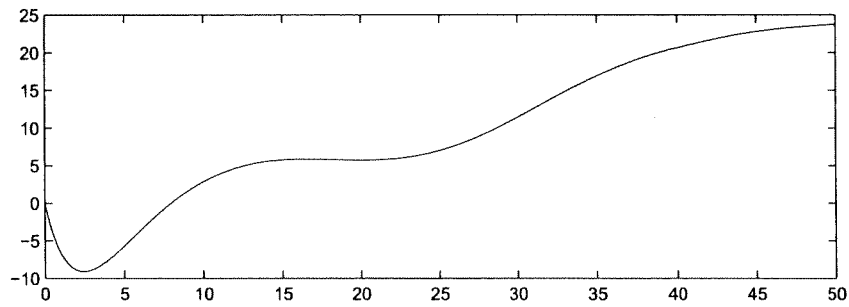


Figure 14.7: Internal state of the detection filter in Experiment 1.

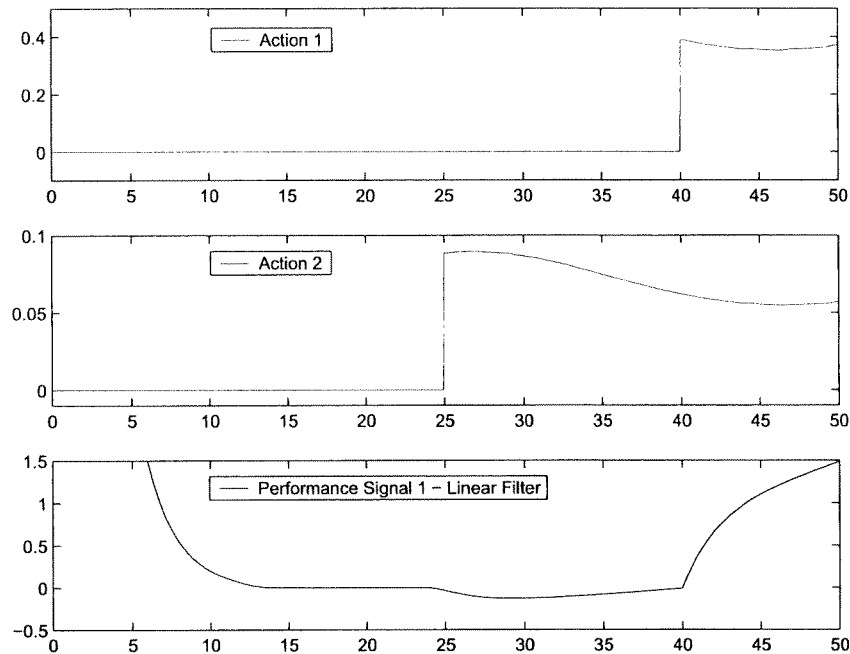


Figure 14.8: Actions 1 and 2 and Performance Signal 1 for the Linear Detection Filter in Experiment 1.

of the linear filter are given in Figure 14.8 and 14.9, respectively. Figure 14.8 evidently shows how the “noninteractive” property of the filter is lost. In fact, the Performance Signal 1 - which virtually should be zero until Action 1 occurs, becomes nonzero in response to the occurrence of Action 2 at time $t = 25$. This is due to the nonlinear terms present in the model (14.2) which are not taken into account by the linear detection filter.

14.6 Results of the Experiments

We report in this section the outcome of experiments performed varying the engagement actions of the two Red Units and the trajectories followed on the plane by the Red and Blue Units. The non-linear model of the battlefield depends on such functions, and so the evolution in time of the number of platforms of the Blue units, which feed the non-linear detection filter. It will be seen that, despite of the changes in the signals which drive the filter, its capabilities of detecting and isolating the two Red unit actions are unaltered.

Consider the following scenario for the second experiment. As in the experiment in the previous Section, the four units present in the battlefield are following the trajectories in Figure 14.2. The actions of the Red Units versus the Blue Units have different occurrence times with respect to the actions considered in the Experiment 1. Action 1 occurs at time $t = 25$ whereas Action 2 occurs at time $t = 40$ (see Figure 14.10). How these different actions change the shape of the time behaviour of the number of platforms of the Blue units is shown in Figure 14.11. The response of the detection filter to the two actions are depicted in Figures 14.12, 14.13. In Figures 14.14, 14.15 are reported the response of the performance signals of the linear filter. As in Experiment 1, even in this case the “non-interaction” property of the filter is lost due to the nonlinear terms. In particular, the Performance Signal 2 erroneously detects Action 2.

The next experiment is aimed to validate the hypothesis that the detector’s performance is not affected by a change in the trajectories of the units in the battlefield. Consider the case in which the four units’

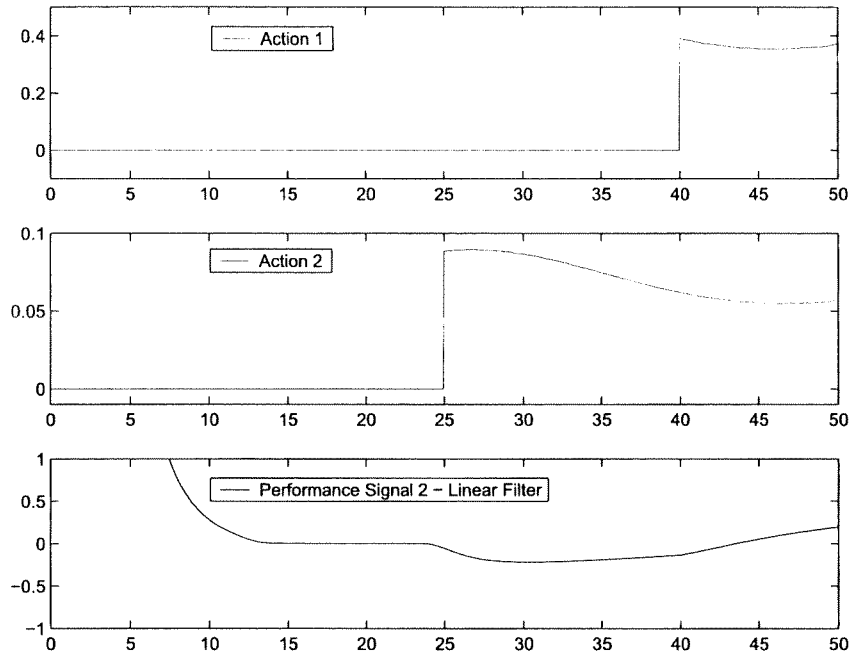


Figure 14.9: Actions 1 and 2 and Performance Signal 2 for the Linear Detection Filter in Experiment 1.

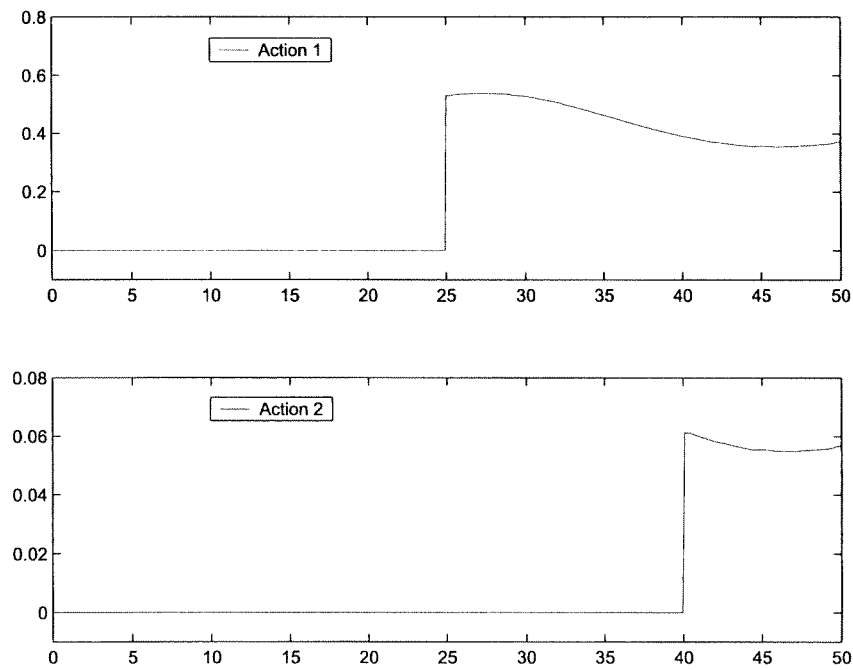


Figure 14.10: Actions 1 and 2 of the Red Units in Experiment 2.

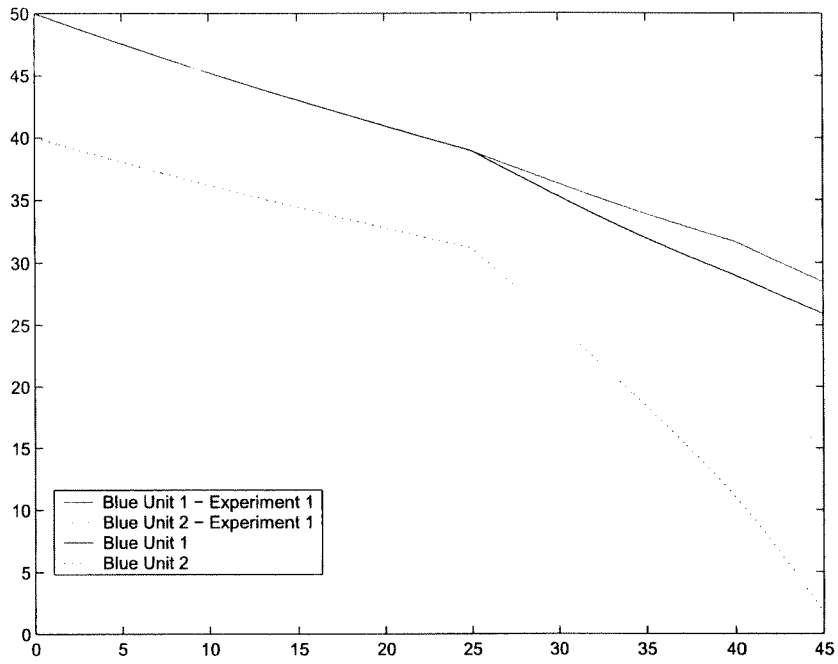


Figure 14.11: Evolution in time of the number of platforms of the Blue Units in Experiment 1 and Experiment 2.

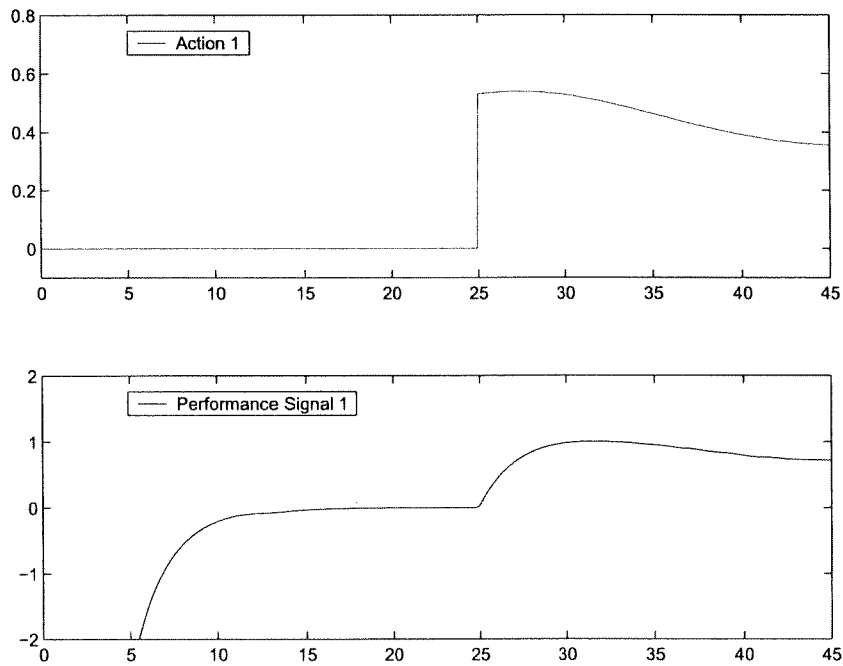


Figure 14.12: Action and Performance Signal 1 in Experiment 2.

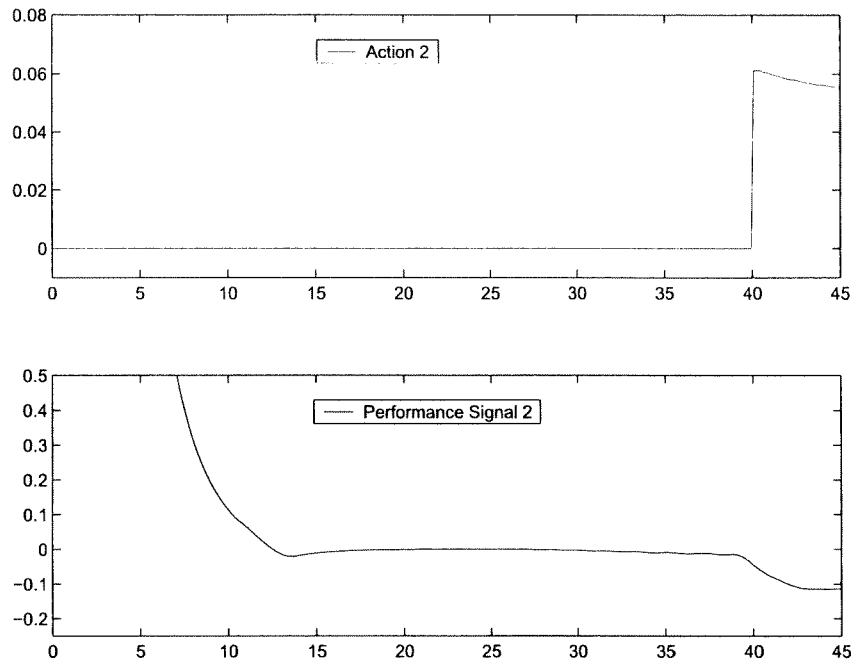


Figure 14.13: Action and Performance Signal 2 in Experiment 2.

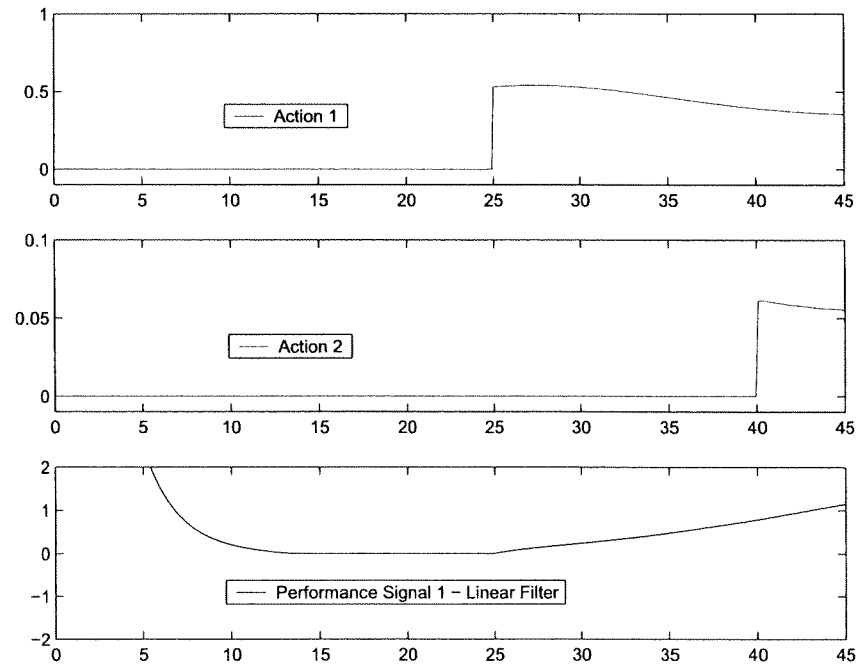


Figure 14.14: Actions 1 and 2 and Performance Signal 1 of the linear filter in Experiment 2.

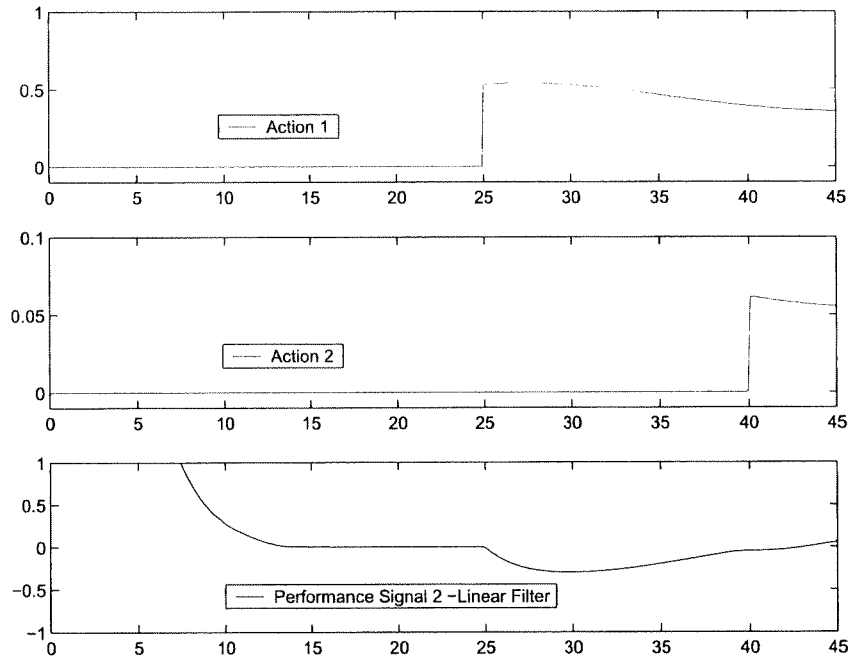


Figure 14.15: Actions 1 and 2 and Performance Signal 2 of the linear filter in Experiment 2.

motion are as those depicted in Figure 14.16. The evolution of the number of platforms of Blue Units can be compared with the same quantities obtained in Experiment 1 in Figure 14.17. The two performance signals generated by the non-linear detection filter are given in Figures 14.18 and 14.19. Figures 14.20 and 14.21 show the performance signals of the linear detection filter.

14.7 Conclusions and Recommendations

The detection and isolation of actions of the “enemy” units versus the “friendly” units in a non-linear model of the battlefield requires the use of a non-linear detection and isolation filter. Since the non-linearities amount to terms which depend on the position of the battling units in the field of operations, the non-linear filter utilizes as inputs the position and the speed control inputs of the friendly units in addition to the number of platforms which is used as input by the linear detection filter. It turns out that the non-linear detection filter detects and isolate concurrent actions by the opposing units whereas the linear filter does not, confounding the two actions to detect. Although not considered in this series of experiments, the case of noisy observations of the number of platforms can be faced similarly to what has been done in Chapters 9 or 10, with an optimal choice of the gain parameters which define the detection filter. This boils down to the solution of a Riccati equation and to a detection filter with more computational complexity.

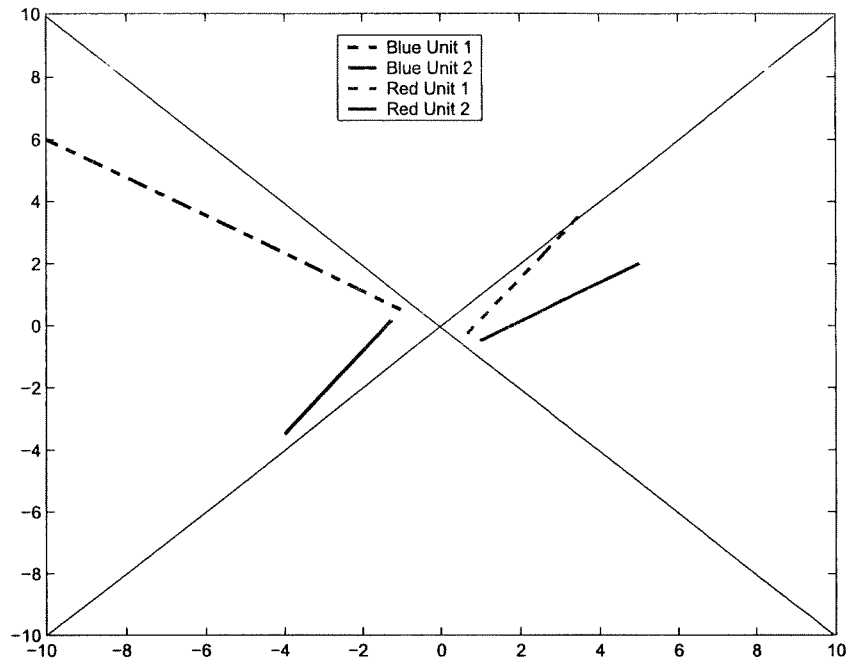


Figure 14.16: Trajectories of the Red and Blue Units in Experiment 3.

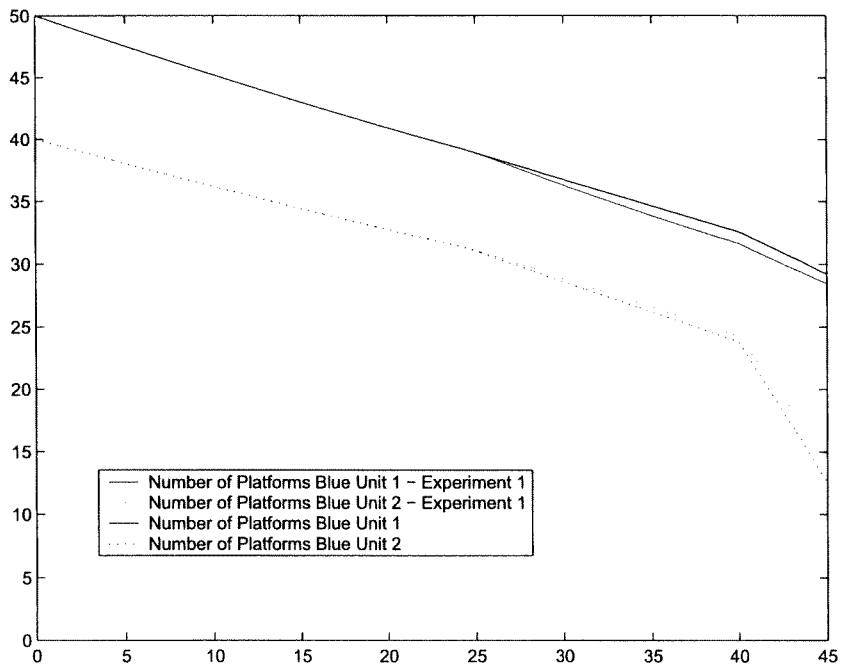


Figure 14.17: Evolution in time for the number of platforms of the Blue Units in Experiment 1 and Experiment 3.

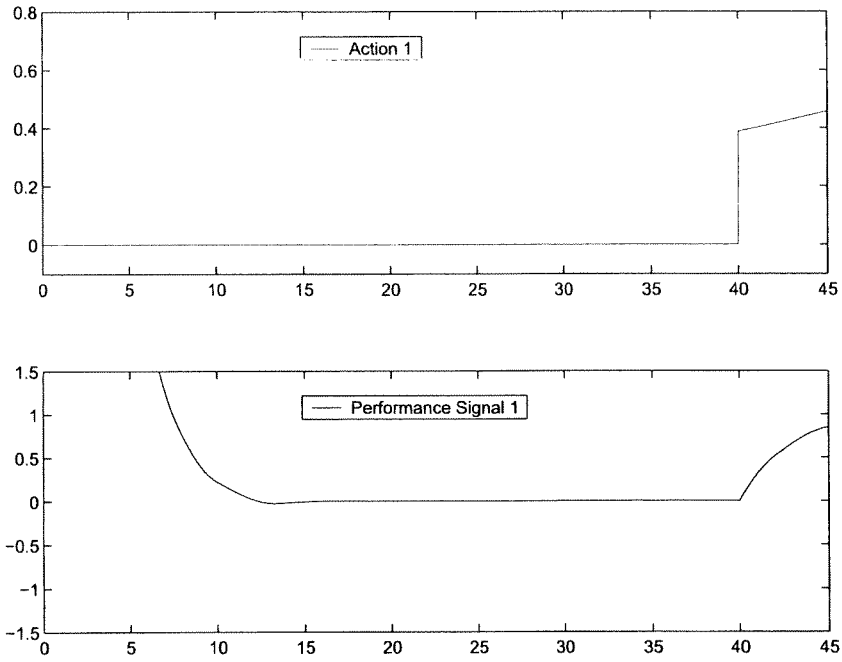


Figure 14.18: Action 1 and Performance Signal 1 in Experiment 3.

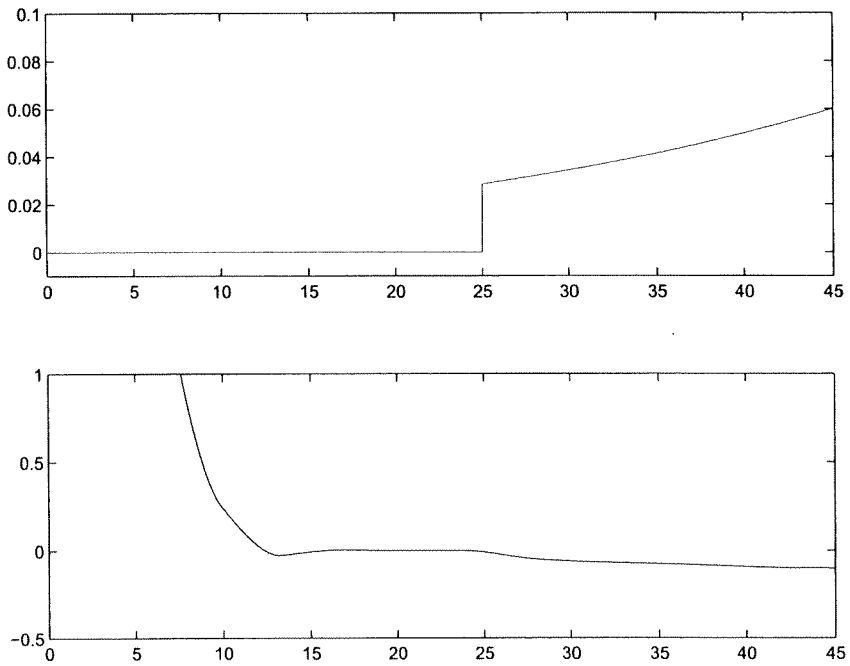


Figure 14.19: Action 2 and Performance Signal 2 in Experiment 3.

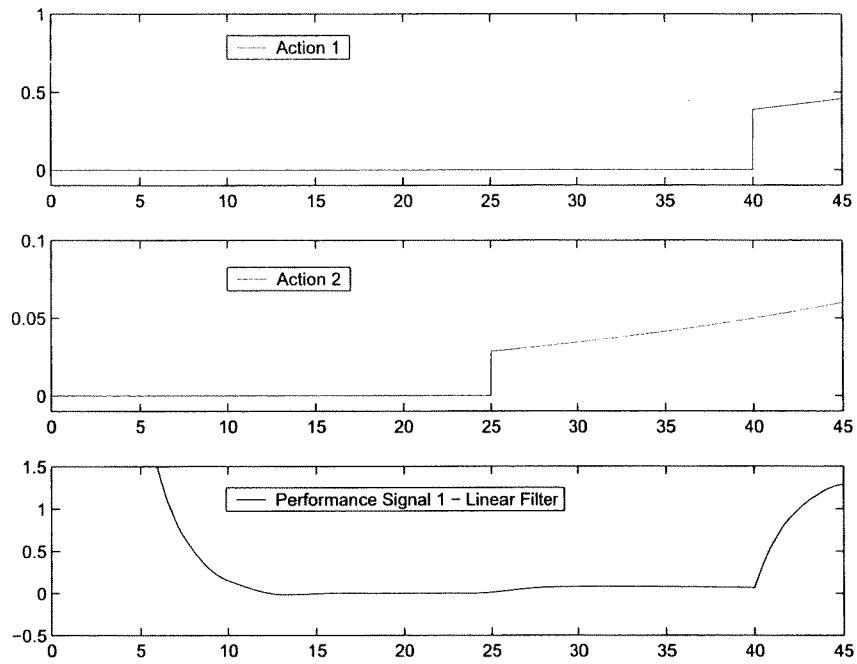


Figure 14.20: Actions 1 and 2 and Performance Signal 1 of the linear filter in Experiment 3.

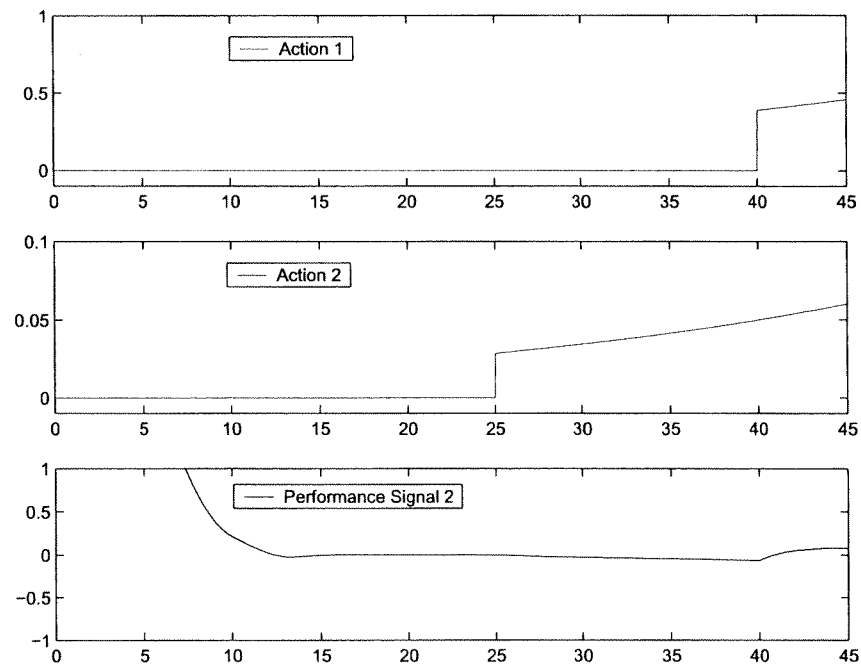


Figure 14.21: Actions 1 and 2 and Performance Signal 2 of the linear filter in Experiment 3.

Bibliography

- [1] H. Mukai et al., Mission Dynamics Continuous-Time Model, Version 2.42, Internal Report, Department of Systems Science and Mathematics, Washington University, 2000.
- [2] C. De Persis, A. Isidori, "On the observability codistribution of a nonlinear system," *Systems & Control Letters*, 40, 297-304, 2000.
- [3] C. De Persis, A. Isidori, A geometric approach to nonlinear fault detection and isolation, to appear as a regular paper on *IEEE Transactions on Automatic Control*, Sept. 2001.

Chapter 15

Experiment 15: Comparison with Honeywell's Results

15.1 Executive Summary

A comparison of the platform loss and probability of success values is made between Washington University and Honeywell results on two example missions, each consisting of three sorties. The results are similar in the first example. Due to a change in the initial number of Red fighters and their probability of kill, the outcome of the second example is drastically different. It has also been observed that the selection of weights in the cost function may affect the unit trajectories and platform loss significantly.

Despite running our Sequential Linear-Quadratic Method for 50 iterations or more, convergence to a possible Nash solution was not achieved in either example, although the obtained unit trajectories and platform loss numbers were reasonable, given the mission objectives.

15.2 Introduction

The purpose of this experiment is to compare the results obtained by the Honeywell Team with those of the Washington University Team, using a common scenario and task description.

The Honeywell approach is based on a discrete-transition Markov chain model of combat between teams of Red and Blue units, and describes platform attrition during a sequence of sorties to accomplish a given mission [1]. In this model, target selection coordination and cooperation between friendly units are explicitly taken into account, but terrain features, initial unit locations and the routes followed by units during a sortie are assumed to be lumped into a single parameter, called the “*lethality*” of a particular type of unit against a particular type of enemy unit. This parameter determines state transitions of the number of platforms during the mission.

Honeywell's Model Predictive Controller [2] is based on a component called the “Initial Deployment Optimizer.” Given the lethality matrix, the number of platforms and decoys in each Red unit, the maximum number of rounds (sorties) in the mission and the desired probability of win, this component calculates the minimum number of Blue platforms to deploy in the first sortie, assuming that all survivors of a sortie will be reassigned to the next one, and no reserves will be called in to join either Red or Blue teams.

The Washington University model starts from similar arguments of target acquisition and target selection coordination, derives a continuous-transition Markov chain for platform attrition, and then approximates the evolution of the expected values of the number of platforms in both Red and Blue teams by a low-order ordinary differential equation. This is combined with unit motion on a two-dimensional battle space and weapon expenditure. In this way, the location and motion of units, the effect of distance on weapon effectiveness and cooperation of friendly units are taken into account. The state transition of the number of platforms is determined by two parameters: the target acquisition rate

and the probability of kill for a particular type of platform against a particular type of enemy platform per firing. As opposed to the aggregate concept of “lethality”, these low-level parameters are determined mainly by the properties of the search devices and the weapon systems, given weather conditions, and thus are independent of the engagement rules, the strength of the teams or the synergy between friendly units.

One component of the Washington University effort is the calculation of the game theoretic optimal control (for both Red and Blue teams) using the Sequential Linear-Quadratic Method (SLQM), given the initial state (number of platforms and weapons in the units, and unit locations), and a cost function which encompasses the trade-off between accomplishing the mission (e.g., reducing the number of enemy platforms), the value of friendly assets, fuel and weapon consumption. The Nash solution (of the zero-sum differential game) computed by this component will depend on how the weights are chosen in the cost function.

It is possible to combine Honeywell’s “Initial Deployment Optimizer” and Washington University’s “Game Theoretic Tactical Solution” components, in an iterative loop for improving lethality estimates. This idea is described in the flowchart in Fig. 15.1.

Before this idea is implemented, it is useful to compare the results of the Honeywell and Washington U. models and controllers. The rest of this report consists of this comparison based on two example scenarios proposed by Honeywell.

15.3 Experiment Setup

The scenarios used for the comparison are best described by the following quotation from [3]:

“Blue is tasked with the objective to destroy a ground target in three missions (sorties) or less. On its way to the target, his strike package will encounter Red’s fighters (...). To lower the loss of his bombers, Blue will provide a few escort fighters to his package. After each mission, the survivors on both sides return to their bases, where they are fully rearmed and then send off again on the next mission. Successful task completion is defined so that both the target must be destroyed within the given deadline and own losses must not exceed a given cap. In particular, destroying the target with own loss exceeding the cap is considered a failure. The task is over whenever the ground target was destroyed (even if it happens in the first or second mission) or three mission have been flown in vain.

Each Blue bomber carries a payload, whose lethality against the Red’s ground target is the first number in the lethality matrix element (1,2). (...) For self-defense, it has cannons whose lethality against the Red fighters is the first number in the lethality matrix element (1,1).

Each Blue fighter is armed with 4 AA missiles, whose lethality against the Red assets are given by the first numbers of the second row elements of the lethality matrix. Note that the fighters has no weapons against the ground target.

Each Red fighter is equipped with 4 AA missiles, whose lethality against Blue bombers and fighters are the second numbers of the first column elements of the lethality matrix.

The Red ground target is passive and cannot shoot back at the Blue package.

Both the Blue and Red fighters fire one missile at a time without target selection coordination with their fellow fighters. (...) Likewise do the bombers. Furthermore, the Blue fighters do not coordinate their target selection with the bombers (The inter-asset coordination.)”

In our experiments, we assume that the Red fighters (**R2**) are targeting Blue bombers (**B1**) only. Blue bombers (**B1**) are dropping bombs on the Red ground target (**R1**) and Blue fighters (**B2**) are firing at the Red fighters (**R2**). This is different than the Honeywell model in which the Blue bombers can also fire at the Red fighters, with low lethality.

The target acquisition rate is set to one for all units, and the lethality numbers in [3] are used as the probability of kill values at optimum distance, as shown in Table 15.1. Note that the probability of kill

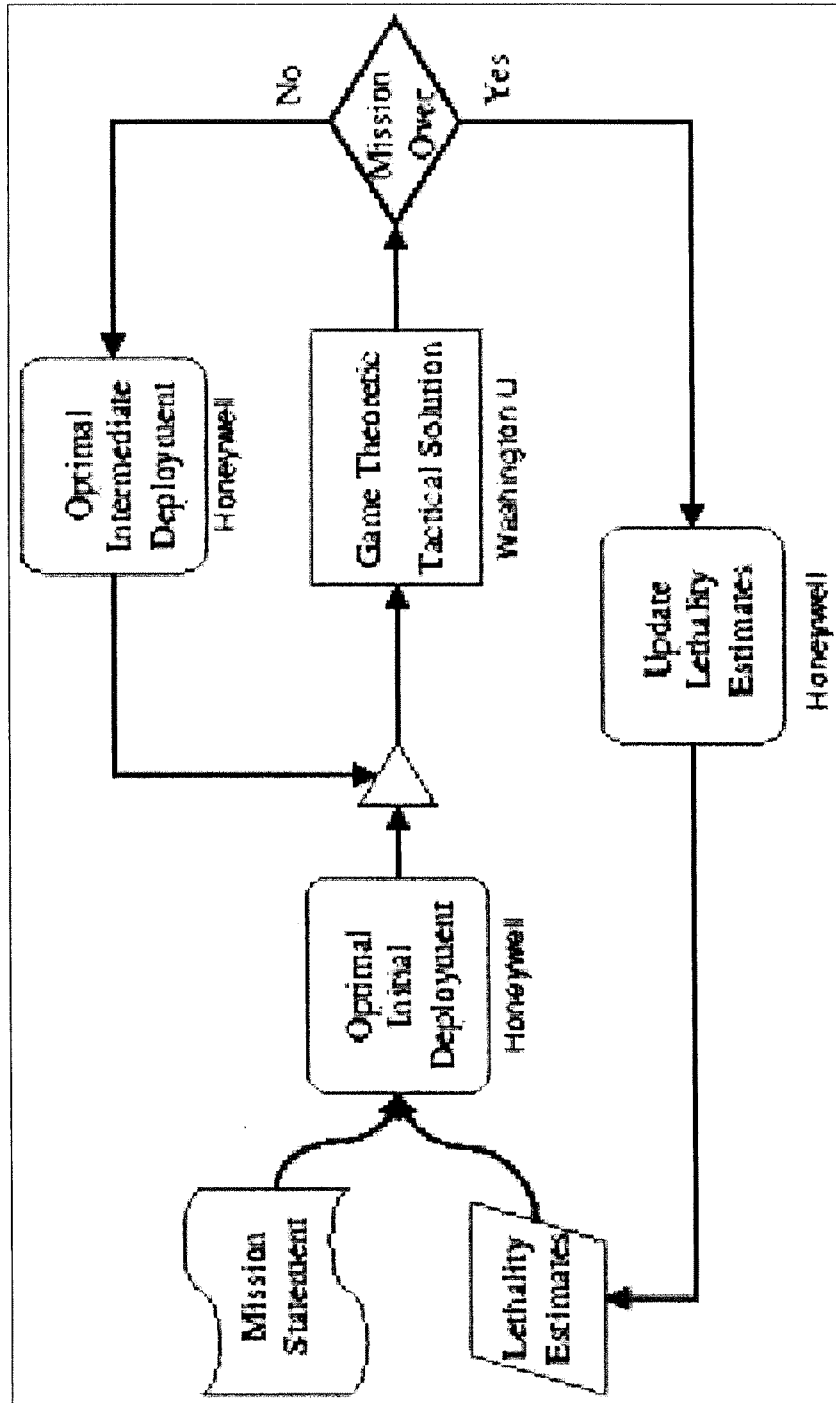


Figure 15.1: Flowchart for interaction of software components

Table 15.1: Probability of Kill Values

	B1 on R1	B2 on R2	R2 on B1
Example 1	0.2	0.3	0.4
Example 2	0.2	0.6	0.6

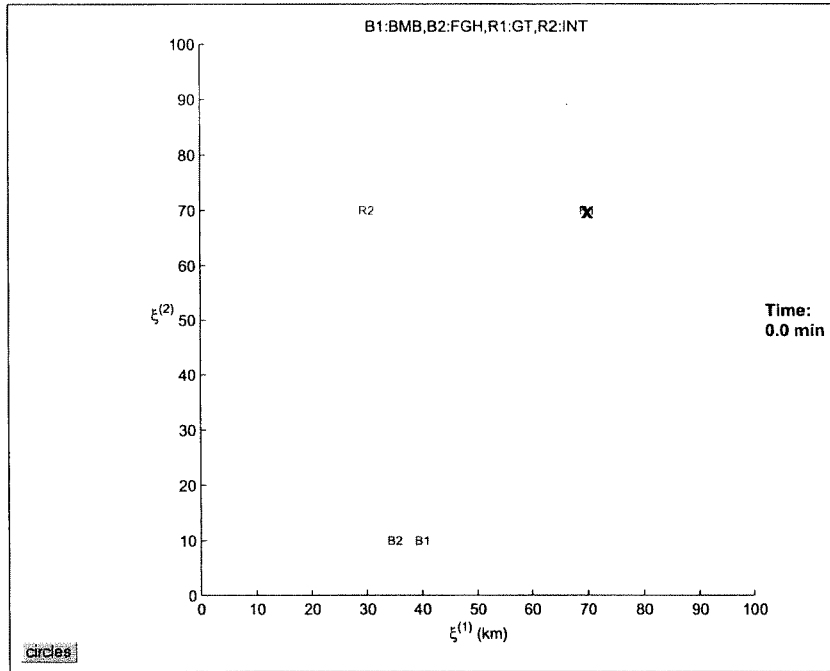


Figure 15.2: Initial positions of the units.

decreases with distance in our model. These are the only parameters that are varied between the two examples.

The initial number of platforms in each unit are those generated by Honeywell's component, as given in [3], except that the Red ground unit is assumed to have 10 platforms initially, instead of a single target. In this way, the expected number of platforms in the Red ground unit at the end of each sortie (loosely) correspond to ten times the probability of failure (one minus prob. success). The initial positions of the units are chosen as shown in Fig. 15.2.

The trajectories and the firing intensities of the units in the Nash solution (in fact, the existence of a Nash solution) are determined by the weights in the cost function. For our experiments, the weights in Table 15.2 are used.

Note that it is possible to obtain very different trajectories and final results by changing the above weights.

15.4 Experiment Results and Analysis

The Sequential Linear-Quadratic Method (SLQM) for the Nash solution computation is terminated in 50 iterations for all sorties. For Example 1, the behavior of the control update (δu) and the cost function,

Table 15.2. Weights in the Cost Function

	B1	B2	R1	R2
distance from destination	1E-3	0	0	0
running cost for platforms	1	1E-4	1	1E-4
final cost for platforms	10	1E-3	10	1E-3
distance from target enemy unit	0	5E-3	0	1E-3
cost on velocity (fuel)	25	25	25	25
cost on firing (weapons)	25	25	25	25

as the iterations progress, are shown in Figs. 15.3 and 15.4 respectively.

It is seen that convergence to a small value of δu is not achieved in 50 iterations. Increasing the number of iterations up to 200, or changing the step size parameter of the algorithm did not improve the convergence. For this reason, it is likely that the results we present below may not correspond to the Nash solution for this problem, although they appear to be reasonable given the mission objectives. In fact, it is not known whether a Nash solution exists for this scenario and the associated cost function. There are many different, equally acceptable, choice of weights corresponding to the same mission statement. However, we do not know a systematic method of determining those weights, and a trial-and-error approach proved to be very time consuming and fruitless for both Examples 1 and 2.

The trajectories obtained by the SLQM algorithm for each sortie of Example 1 are depicted in Figs. 15.5, 15.6 and 15.7.

For Example 2, the behavior of the control update (δu) and the cost function, as the iterations progress, are shown in Figs. 15.8 and 15.9 respectively. It is seen that convergence to a small value of δu is not achieved in 50 iterations. Similar to Example 1, increasing the number of iterations up to 200, or changing the step size parameter of the algorithm did not improve the convergence. For this reason, it is likely that the results we present may not correspond to the Nash solution for this problem, although they appear to be reasonable given the mission objectives.

The trajectories obtained by the SLQM algorithm for each sortie of Example 2 are depicted in Figs. 15.10, 15.11 and 15.12.

The expected platform loss and probability of success results of the two examples are summarized in Tables 15.3 and 15.4. The Honeywell results, taken from [3], are based on the “Bombers First” strategy for the Red fighters and “max loss = 3” assumption. For Washington U. tests, the same strategy for the Red fighters is enforced by assigning the Blue bombers as the sole target of this unit. The probability of success (the probability of destroying the Red ground target) is calculated from the expected number of remaining platforms in the Red ground unit **R1**.

In Example 1, the Honeywell and Washington U. results are comparable. Since the modeling assumptions are similar, the sources of discrepancy are the effect of location of the units in the theater, and the selection of weights in the cost function, which express the relative importance of achieving target destruction versus platform loss, fuel and weapon consumption.

In Example 2, the initial number of Red fighters is increased to 5 and their probability of kill against the Blue bombers is increased to 0.6 from 0.4. Even though the cost function is kept the same as in Example 1, the Blue bomber unit **B1** is more concerned about its own safety and rather reluctant to get closer to its destination, as seen in Figs. 15.10–15.12. Therefore, the probability of success is much lower. The Honeywell results for Example 2 are not available at this time.

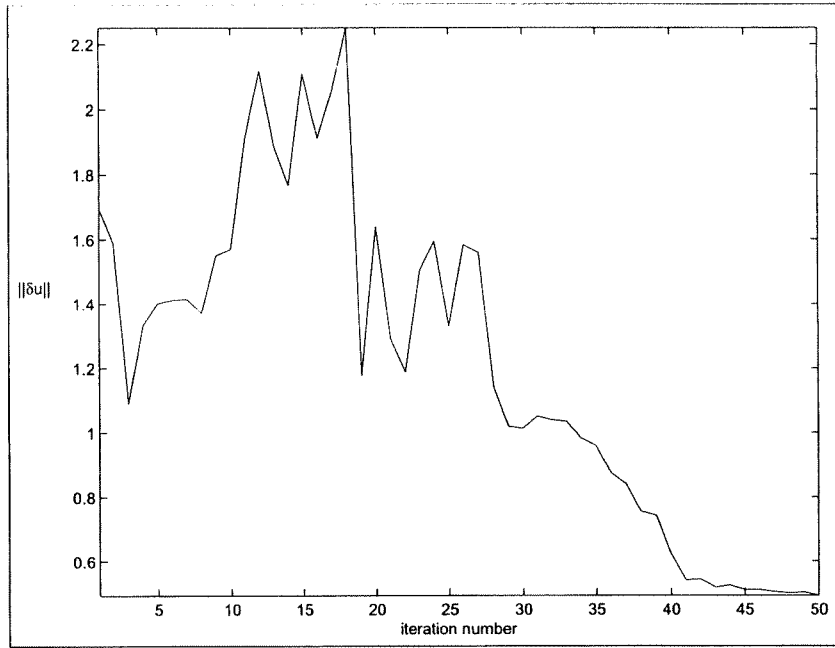


Figure 15.3: Root-mean-square (rms) value of the control update δu vs. iterations for Example 1.

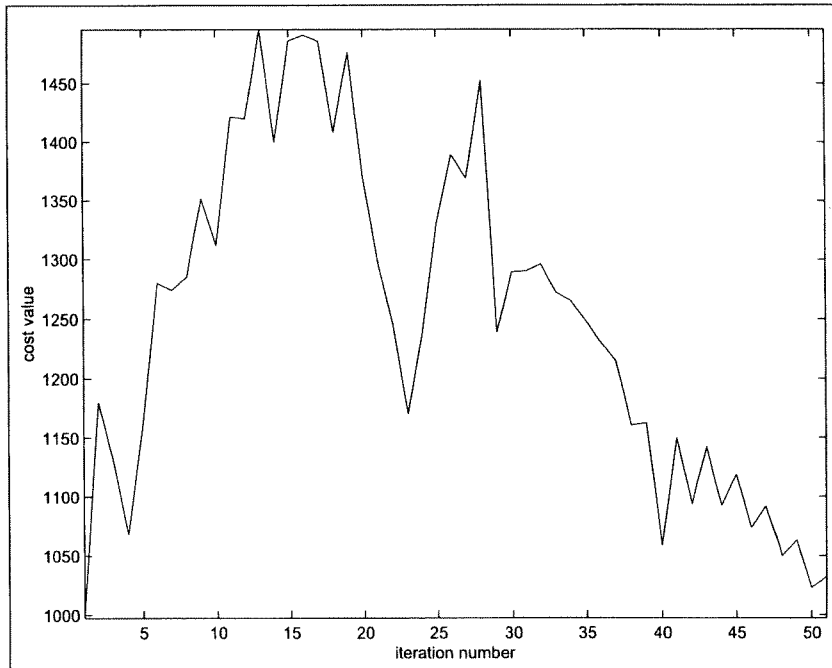


Figure 15.4: Value of the cost function vs. iterations for Example 1.

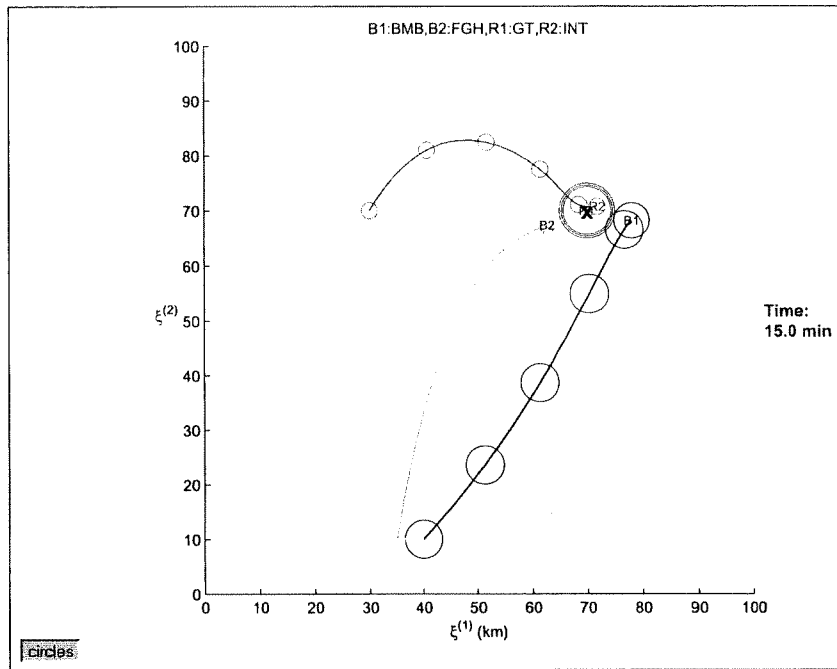


Figure 15.5: Trajectories of units, Example 1, Sortie 1.

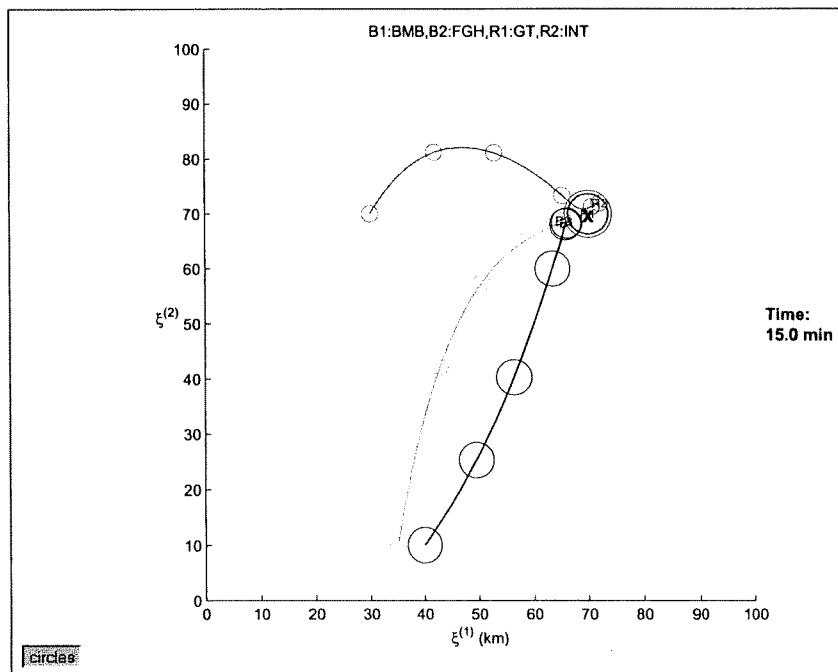


Figure 15.6: Trajectories of units, Example 1, Sortie 2.

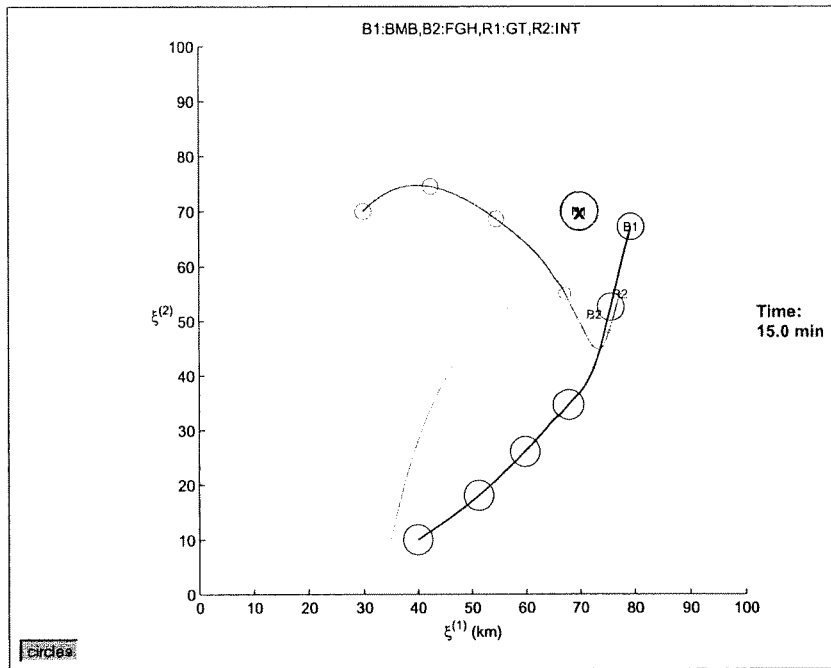


Figure 15.7: Trajectories of units, Example 1, Sortie 3.

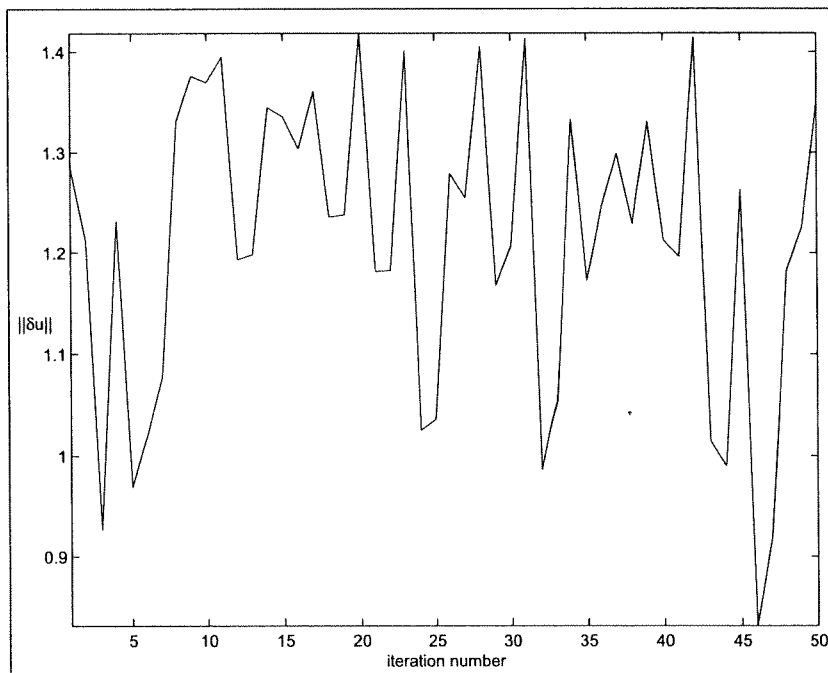


Figure 15.8: Root-mean-square (rms) value of the control update δu vs. iterations for Example 2.

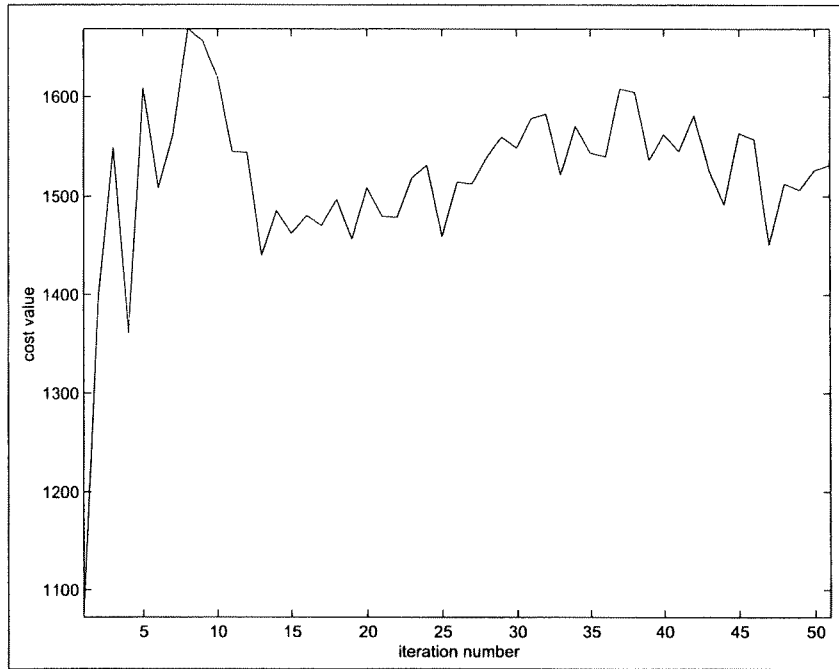


Figure 15.9: Value of the cost function vs. iterations for Example 2.

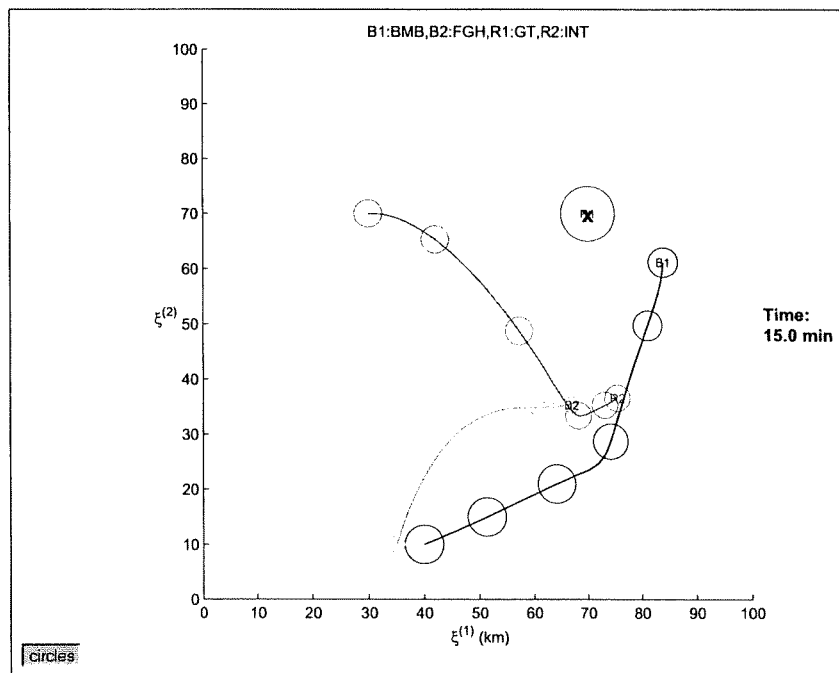


Figure 15.10: Trajectories of units, Example 2, Sortie 1.

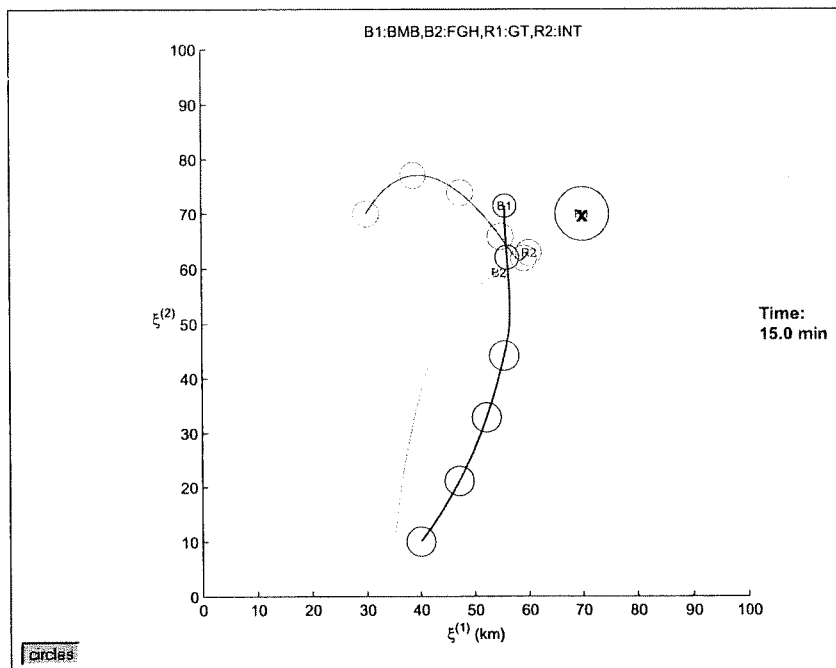


Figure 15.11: Trajectories of units, Example 2, Sortie 2.

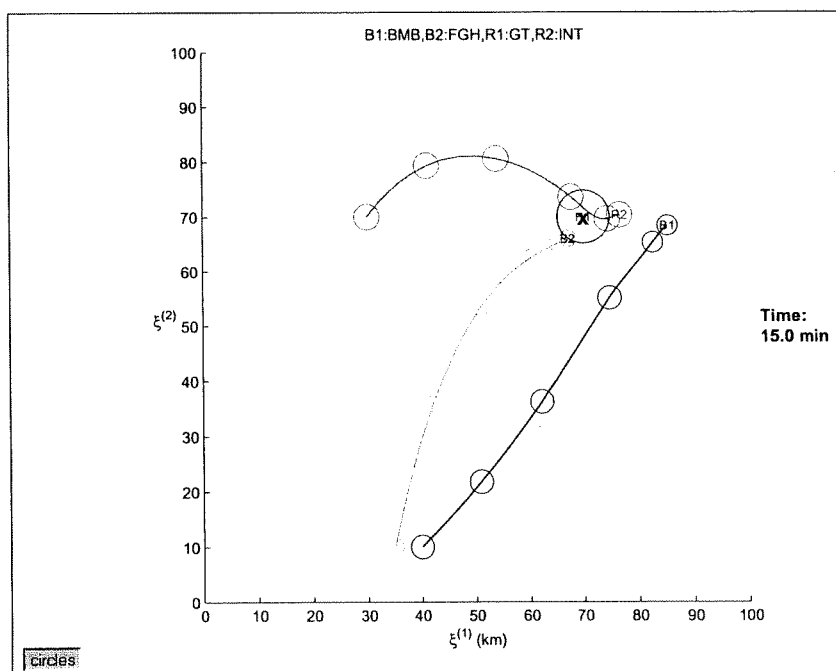


Figure 15.12: Trajectories of units, Example 2, Sortie 3.

Table 15.3: Example 1: Probability of Success and Remaining Number of Platforms After Each Sortie

	Washington U.					Honeywell			
	P{success}	B1	B2	R1	R2	P{success}	B1	B2	R2
Initial		7.00	3.00	10.0	3.00		7.00	3.00	3.00
Sortie 1	0.14	6.40	3.00	8.60	3.00	0.19	4.80	3.00	0.10
Sortie 2	0.29	5.50	3.00	7.10	2.90	0.34	4.74	3.00	0.002
Sortie 3	0.33	4.80	3.00	6.70	0.50	0.46	4.74	3.00	0.00

Table 15.4: Example 2: Probability of Success and Remaining Number of Platforms After Each Sortie

	Washington U.					Honeywell			
	P{success}	B1	B2	R1	R2	P{success}	B1	B2	R2
Initial		7.00	3.00	10.0	5.00		7.00	3.00	5.00
Sortie 1	0.01	5.40	3.00	9.90	4.80				
Sortie 2	0.02	4.30	3.00	9.80	4.70				
Sortie 3	0.04	3.60	3.00	9.60	4.70				

15.5 Conclusions and Recommendations

In this experiment, we have compared the platform loss and probability of success values between Washington U. and Honeywell results on two example missions, each consisting of three sorties. Despite running our SLQM algorithm for 50 iterations, convergence to a small value of the control update (which would indicate a possible Nash solution) was not achieved in either example, although the unit trajectories and platform loss numbers were reasonable, given the mission objectives.

In the first example, our results were similar to Honeywell's. The second example resulted in drastically different attrition numbers, due to an increase in the initial number of Red fighters from 3 to 5 and their probability of kill by 50%. This indicates the sensitivity of our algorithm, and possibly the sensitivity of the Nash solution concept, to initial states. On the other hand, our solution yields optimal (or near-optimal) routes and firing intensity values, given the cost function, which are not part of the Honeywell model.

The Honeywell approach is based on maximizing the probability of success, while our approach tries to find the saddle-point of a cost function. It may be worthwhile to spend some effort to investigate how the selection of weights in the cost function affects the probability of success. Even better would be to devise a "cost translator", which will yield good weight values (for which a Nash solution exists) that can reflect the trade-off between desired probability of success and acceptable platform loss.

Bibliography

- [1] J. Jelinek, *Modeling Complex Battles, Part 1*, Memorandum, Honeywell Technology Center, Minneapolis, MN, September, 2000.
- [2] J. Jelinek and D. Godbole, *Model Predictive Control of Battle Dynamics*, Memorandum, Honeywell Technology Center, Minneapolis, MN, May, 2000.
- [3] J. Jelinek, *Joint Experiment Honeywell, Rockwell, OSU, and WUSTL*, Memorandum, Honeywell Technology Center, Minneapolis, MN, Feb. 12, 2001.

Chapter 16

Experiment 16: Controller Computational Complexity: Correction

16.1 Executive Summary

The *purpose* of Experiment 16 is to correct an error present in the subprogram that evaluates the Jacobian of the model MDCM. This error would have affected the results in cases in which multiple units are deployed against multiple units and some units are not fired upon. This error affects only one such case in the Interim Report (experiments 1 through 12), that is experiment 5.3.2. Therefore, a corrected version of the subprogram for computing the Jacobian has been developed, and corrected computational results are reported in this chapter. Even with this change we can draw the same conclusions as in Experiment 5; namely, the computational time is a quadratic function of the number of units.

16.2 Introduction

The original purpose of Experiment 5 in Chapter 5 of the Interim Report and of this Final Report was to perform a number of experiments to test the following hypothesis: The computational complexity of the differential game technology based controller increases quadratically as a function of the number of units and linearly as a function of the mission duration.

One experiment run reported in Chapter 5 showed that the units not being fired upon did not move, contrary to intuitive expectations. After a careful examination, we found an error in the subprogram that evaluates the Jacobian of the MDCM model (Mission Dynamics Continuous-time Model). This error would have affected the results in cases in which multiple units are deployed against multiple units and some units are not fired upon. There is only one such case in the Interim Report (Experiments 1 through 12), that is experiment 5.3.2, in which two units out of six remain fixed in their initial positions. Therefore, a corrected version of the subprogram for computing the Jacobian has been developed, and corrected computational results are reported in this chapter.

In the original experiments, both the plant and controller models are the same, given by MDCM. In a first set of experiments the number of units in the scenario is increased while the mission objectives and duration are kept constant. In a second set, the mission duration is increased, while the mission objectives and the number of units are kept constant. The computation time and the number of iterations required for the computation of the control law to converge were recorded in both cases.

16.3 Experiment 5.1: The Number Of Units Is Increased While The Mission Duration Is Kept Constant

In this set of experiments, the mission duration is kept constant at 20 minutes.

Five experiments have been conducted for each of the following cases: 1 unit vs. 1 unit, 2 vs. 2, 3 vs. 3, 4 vs. 4 and 5 vs. 5. In these 5 experiments for each n vs. n case ($1 \leq n \leq 5$), the units categories, initial conditions, target locations and nominal trajectories as well as the weights in the cost function may vary. The computational time and the number of iterations are recorded for each experiment.

16.3.1 One vs. One

In this section, the results reported in Chapter 5 were correct and thus there was no need to redo the experiments.

16.3.2 Multi-units Case

An example for multiple units case was reported in Chapter 5 for 3 units vs. 3 units. Here the corrected version of the Jacobian subprogram yields results in which all units now move from their respective starting positions.

Table 16.1 summarizes the pertinent information for the two opposing forces in that specific example. The manner of engagement in that example is: R1 and R2 are programmed to attack B1 and R3 is programmed to attack B3. B1 and B2 are programmed to attack R1 and R2 respectively, and B3 is programmed to attack R3. The choice of the weights is slightly different from those of the Experiment 5.1.

Table 16.1: Data for Three vs. Three

	B1	B2	B3	R1	R2	R3
Unit categories	bombers	bombers	ground	bombers	interceptors	ground
Initial no. of platforms	10	10	10	10	10	10
Initial no. of weapons	10	10	10	10	10	10
Initial position	(20,53)	(20,50)	(45,47)	(80,53)	(80,50)	(55,47)
Target location	(70,63)	(80,52)	(53,48)	(30,63)	(20,48)	(43,46)

Figures 16.1 - 16.2 show respectively the initial state trajectories and the convergence of the control updates $\|\delta u\|$. Figures 16.3 - 16.5 present the Nash solution; specifically Figure 16.3 presents the Nash solution trajectories, Figure 16.4 presents the corresponding firing intensities and Figure 16.5 presents the history of the number of platforms. With a convergence criterion of the norm $\|\delta u\|$ of the control change δu less than 0.01, convergence is attained after 34 iterations. The total simulation time is now 601.13 sec.

The main difference from the previous simulation results reported in Chapter 5 consists of the movement that units B3 and R3 now show, in agreement with the intuitive expectation based on the choice of the weights for the cost function. In general, the corrected version of the Jacobian subprogram makes a difference in scenario files in which some units are not shot at by an enemy unit.

16.3.3 Multi-units Case And Computational Complexity

Due to the correction made for the Jacobian computation, the computational time (601.15 sec.) is slightly longer than what was reported in Chapter 5 for the case of 6 units. However, we may draw the same conclusion as before. This can be verified by analyzing Figure 5.11 of Chapter 5, which is also shown here as Figure 16.6. When the number of units is six, as analyzed here, the computational time is 601.13

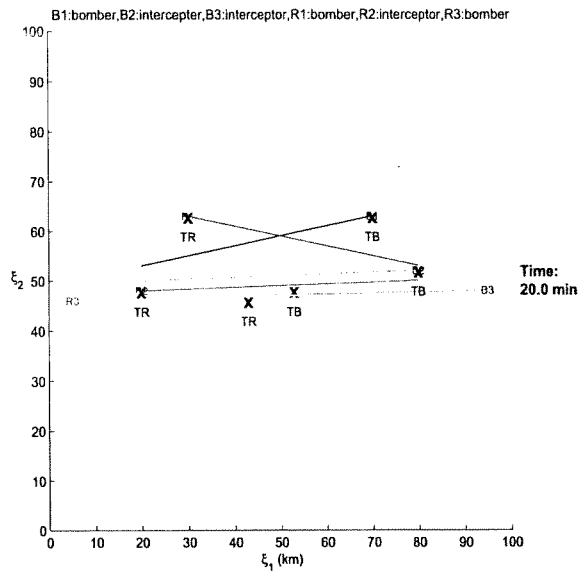


Figure 16.1: Initial Trajectories for Three vs. Three

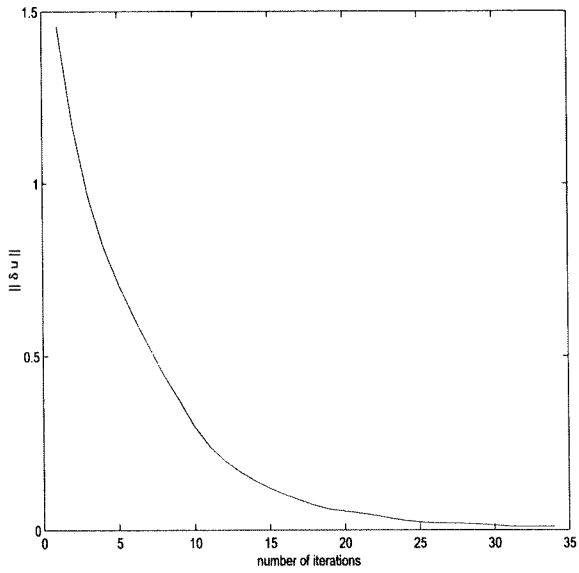


Figure 16.2: Convergence for Control Updates for Three vs. Three

sec. Even with this change we can draw the same conclusions as before. Namely, the computational time is a quadratic function of the number of units.

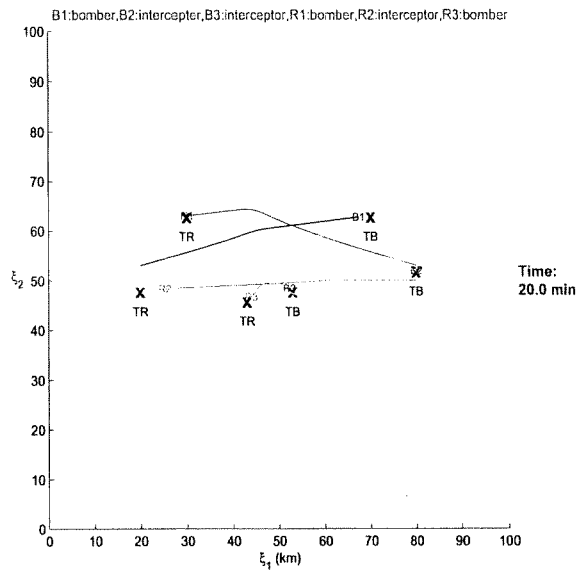


Figure 16.3: Nash Trajectories for Three vs. Three

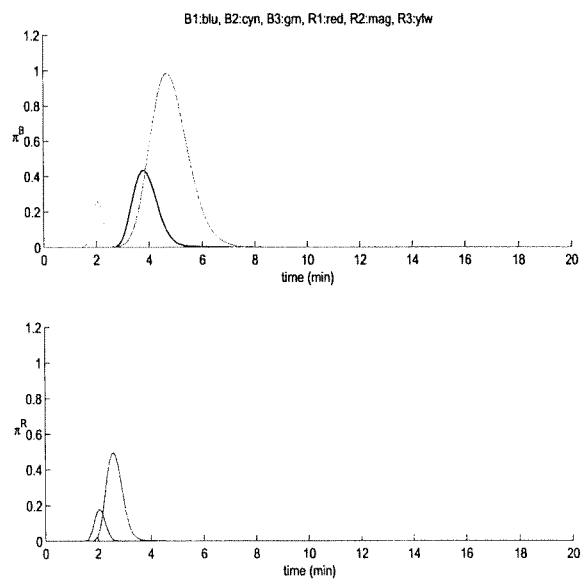


Figure 16.4: Nash Firing Intensities for Three vs. Three

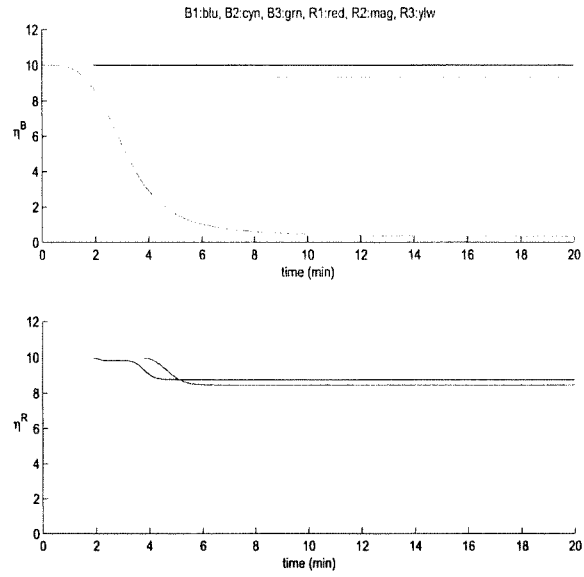


Figure 16.5: Nash Number Of Platforms for Three vs. Three

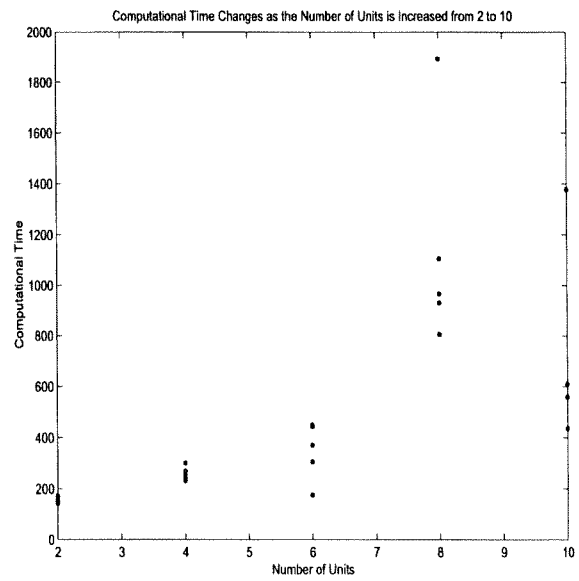


Figure 16.6: The Computational Time Changes As The Number Of Units Is Increased

16.4 Experiment 5.2: The Mission Duration Is Increased While The Number Of Units Is Kept Constant

In this set of experiments, previous results still hold.

16.5 Conclusions

The use of the corrected Jacobian subprogram does not change substantially the main conclusions of Chapter 5. They are: the computational time required to reach the convergence criterion depends on many factors, such as the units categories, the number of units, initial trajectories, weights in the cost function, step size in our numerical procedure and the manner of engagements as well as initial positions and target locations. Similarly the number of iterations required to reach convergence depends on the same factors. In our experimental results, major factors which affect the computational time are the number of units and the mission duration. For our experiments the computational time of the controller increased quadratically as a function of the number of units and linearly as a function of the mission duration, while the number of iterations itself remained relatively constant as a function of the number of units. This last point was a pleasant surprise.

Chapter 17

Experiment 17: Controller with a Kalman Filter for Estimation

17.1 Executive Summary

In this chapter, we present how an algorithm based on the Extended Kalman Filter (EKF) for state estimation is used in a differential game, which models the air operations of two opposing forces. We show the overall structure of the game in a block diagram. We present the implementation of the algorithm in a flowchart. We also present simulation results.

In an air operation game, it is reasonable to assume that one does not get direct information about his enemy's input. In this paper, we present an approach for estimating the states of the friendly as well as enemy forces and compare their respective simulation results. The Kalman filter due to Darouach et al. treats the enemy inputs as part of the extended state and obtains an estimate of both the state of the two forces and the input of the enemy. But their filter is designed for linear time-invariant systems. Hence, we present an extension of their filter to a nonlinear time-variant system.

The extended Kalman filter algorithm presented in this report is capable of estimating the states of both forces in the presence of process noise as well as sensor noise. We note that the estimates of the enemy inputs are too noisy to be directly useful. However, our game-theoretic controller requires only an estimate of the enemy state and it does not require any estimates of the enemy input. We thus observed the game-theoretic controller remained effective when the extended Kalman filter is introduced in the loop.

17.2 Introduction

The purpose of the experiment is to show that the *current* differential game technology, combined with an extended Kalman filter, provides an effective means of countering the enemy actions under *idealized* situations with *perfect* information about enemy initial conditions and objectives, but with noisy measurements of a subset of the enemy state.

Description: Both the plant and internal models are the same, i.e., the MDCM (Mission Dynamics Continuous Model). Increasing levels of noise will be added to the state variables when constructing the observed state variables (the output variables). Some of the enemy state variables (weapons per platform first, and number of adversary platforms next) will be removed from the set of output variables thus making them not directly observable. The control actions of the Blue and Red teams are generated by the proposed game theoretic algorithm.

The *current* differential game technology, combined with an extended Kalman filter (EKF) provides an effective means of countering the enemy actions under *idealized* situations with *perfect* information about enemy initial conditions and objectives, but with noisy measurements of a subset of the enemy state. The algorithm based on EKF adequately estimates the unknown red state in the presence of process and observation noise.

Consider a dynamical system governed by the following equation,

$$\frac{d}{dt}x(t) = f(x(t), u(t), t) + w(t), \quad t \in [t_0, t_f]; \quad x(t_0) = z_0, \quad (17.1)$$

and a observation process given by

$$y(t) = h(x(t), u(t), t) + v(t), \quad t \in [t_0, t_f], \quad (17.2)$$

where the control u is an \mathbb{R}^m -valued function on $[t_0, t_f]$, $f(x, u, t)$ is an \mathbb{R}^n -valued continuously differentiable function on $\mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}$, $h(x, u, t)$ is an \mathbb{R}^p -valued continuously differentiable function on $\mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}$, the initial state z_0 is a Gaussian random variable, and the process noise $w(t)$ and the measurement (sensor) noise $v(t)$ are Gaussian white noise processes. We assume that these random variables and random processes are mutually independent. For any fixed initial state $x(t_0) = z_0$ and any admissible control u of some restricted class U , we assume that, equation (17.1) has a unique solution x . Such a solution x is called the *trajectory* of the system produced by control u and denoted by $x[u]$. Rigorous definition of (17.1) by Ito's stochastic integral and the theory of stochastic differential equations can be found in [1], [3] and [4].

Our dynamical system (17.1)-(17.2) models a game played by opposing military forces in battle through their air operations. The control function u consists of two parts, u^B and u^R , corresponding to the two forces, the *Blue* and the *Red* forces: $u = (u^B, u^R)$. In actual theaters for military air operations, information is sometimes not available and is corrupted by measurement errors and misleading signals from the enemy, even when available. These corruptions are modeled as white noise error processes in (17.1)-(17.2). We investigate the problem of estimating the state of the enemy from noisy signals about the enemy location without knowing the numbers of the enemy platforms (a part of the enemy state) and the control inputs u^R of the enemy. We then propose an extended Kalman filter for the problem and evaluate its effectiveness in the closed-loop of a game-theoretic controller.

In this report, for practical purposes and the flexibility of analysis, we replace the continuous-time model (17.1)-(17.2) by the following discrete-time dynamics and observation process:

$$x_{k+1} = f_k(x_k, u_k) + w_k, \quad k \geq 0; \quad x_0 = z_0, \quad (17.3)$$

$$y_k = h_k(x_k, u_k) + v_k, \quad k \geq 0, \quad (17.4)$$

where $f_k(x, u)$ is an \mathbb{R}^n -valued continuously differentiable function on $\mathbb{R}^n \times \mathbb{R}^m$, $h_k(x, u)$ is an \mathbb{R}^p -valued continuously differentiable function on $\mathbb{R}^n \times \mathbb{R}^m$, the initial state z_0 is a Gaussian random variable, and the process noise w_k and the measurement noise v_k are zero-mean Gaussian white noise sequences uncorrelated with each other and with the initial state z_0 of the system. Their respective covariance matrices are given as

$$E(w w') = W, \quad E(v v') = V, \quad \text{where } W > 0, \quad \text{and } V > 0 \text{ are diagonal.} \quad (17.5)$$

Here, we also assume that each control input u_k consists of two parts, u_k^B and u_k^R , corresponding to the two forces, the *Blue* and the *Red* forces: $u_k = (u_k^B, u_k^R)$. In this report, we take the point of view of the *Blue* force and construct a Kalman filter for the *Blue* force. We suppose that the enemy inputs (i.e., u_k^R) are unknown. Thus, we need filters which do not make use of the enemy inputs u_k^R .

Section 17.3 is devoted to a Kalman filter technique for estimation in systems with unknown inputs. We present a Kalman filter for estimating the enemy state and the enemy inputs for a linear system. This filter was adapted from the result by Darouach et al. [2]. We also propose an extended Kalman filter for a nonlinear system, and then, we apply the extended Kalman filter to a nonlinear, continuous-time, stochastic air operation game. We present the differential game in the battlefield. The experimental scope and setup is given in Section 17.4, and the simulation results and analysis are given in Section 17.5. The chapter concludes with some conclusions.

17.3 Kalman filters for systems with unknown inputs

Linear Kalman filter for estimating states and enemy inputs: We consider the following linear model:

$$x_{k+1} = A_k x_k + B_k^B u_k^B + B_k^R u_k^R + w_k, \quad k \geq 0; \quad x_0 = z_0, \quad (17.6)$$

and

$$y_k = H_k x_k + v_k, \quad k \geq 0. \quad (17.7)$$

Here, for time k , $x_k \in \mathbb{R}^n$ denotes the state vector, $u_k^B \in \mathbb{R}^{m_1}$ denotes the known Blue control input, $u_k^R \in \mathbb{R}^{m_2}$ denotes the unknown Red control input and $y_k \in \mathbb{R}^p$ denotes the output vector. The noise processes w_k and v_k were as given in Section 1 and the matrices A_k, B_k^B, B_k^R and H_k have appropriate dimensions.

We consider the problem of recursive estimation for linear system (17.6)-(17.7) by estimating the state vector x_k and the unknown input u_k^R from Kalman filtering without knowing the value of the enemy input u_k^R . One possible solution is to construct a Kalman filter by defining a new state vector $\mathcal{X}_k = (x_k', u_{k-1}^{R'})'$. This approach was taken and worked out for time-invariant case by Darouach et al. [2] under the time invariant version of the following assumption:

Rank Condition 1

$$\begin{aligned} \text{rank}(H_k) &= p, & \text{rank}(B_k^R) &= m_2, \\ m_2 &\leq p, & \text{and } \text{rank}(H_k B_k^R) &= m_2, \text{ for all } k. \end{aligned}$$

Let $\hat{x}_{k/l}$ denote the estimate of x_k based on the measurements y up to and including time instant l . Other estimates are defined similarly. We extend the Kalman filter in Theorem 3 in [2], which is for a time-invariant system, to the following time-variant version.

Proposition 5. Assume that Rank Condition 1 is satisfied for the linear system (17.6)-(17.7). Then the optimal estimates for the entire state and the unknown Red inputs are obtained by

$$\bar{x}_{k/k} := A_k \hat{x}_{k/k} + B_k^B u_k^B, \quad (17.8)$$

$$\begin{aligned} \hat{x}_{k+1/k+1} &= \bar{x}_{k/k} + B_k^R \hat{u}_{k/k+1}^R \\ &+ K_{k+1}^x \left\{ y_{k+1} - H_{k+1} \left(\bar{x}_{k/k} + B_k^R \hat{u}_{k/k+1}^R \right) \right\}, \end{aligned} \quad (17.9)$$

$$\hat{u}_{k/k+1}^R = K_{k+1}^{u^R} (y_{k+1} - H_{k+1} \bar{x}_{k/k}), \quad (17.10)$$

where the Kalman gain matrices, K_{k+1}^x for the state estimate and $K_{k+1}^{u^R}$ for the unknown input estimate, are obtained respectively by

$$K_{k+1}^x = \left(\bar{P}_{k/k}^{-1} + H_{k+1}' V^{-1} H_{k+1} \right)^{-1} H_{k+1}' V^{-1}, \quad (17.11)$$

$$K_{k+1}^{u^R} = P_{k+1/k+1}^{u^R} H_{k+1}' V^{-1}, \quad (17.12)$$

where

$$\bar{P}_{k/k} = A_k P_{k/k}^x A_k' + W \quad (17.13)$$

and the estimation error covariance matrix

$$P_{k/k} = E \left(\hat{\mathcal{X}}_{k/k} - \mathcal{X}_{k/k} \right) \left(\hat{\mathcal{X}}_{k/k} - \mathcal{X}_{k/k} \right)'$$

with $\mathcal{X}_k = (x'_k, u_{k-1}^R)'$ and $\hat{\mathcal{X}}_k = (\hat{x}'_{k/k}, (\hat{u}_{k-1/k}^R)')$ is partitioned as

$$P_{k/k} = \begin{pmatrix} P_{k/k}^x & P_{k/k}^{xu^R} \\ P_{k/k}^{u^R x} & P_{k-1/k}^{u^R} \end{pmatrix}, \quad (17.14)$$

and each block matrix has the following form:

$$P_{k/k+1}^{u^R} = \left\{ B_k^{R'} H_{k+1}' (V + H_{k+1} \bar{P}_{k/k} H_{k+1}')^{-1} H_{k+1} B_k^R \right\}^{-1}, \quad (17.15)$$

$$P_{k+1/k+1}^{xu^R} = P_{k+1/k+1}^x \bar{P}_{k/k}^{-1} B_k^R \left(B_k^{R'} \bar{P}_{k/k}^{-1} B_k^R \right)^{-1}, \quad (17.16)$$

$$P_{k+1/k+1}^{u^R x} = P_{k/k+1}^{u^R} B_k^{R'} \bar{P}_{k/k}^{-1} \left(\bar{P}_{k/k}^{-1} + H_{k+1}' V^{-1} H_{k+1} \right)^{-1}, \quad (17.17)$$

$$P_{k+1/k+1}^x = \left\{ \bar{P}_{k/k}^{-1} + H_{k+1}' V^{-1} H_{k+1} - \bar{P}_{k/k}^{-1} B_k^R \left(B_k^{R'} \bar{P}_{k/k}^{-1} B_k^R \right)^{-1} B_k^{R'} \bar{P}_{k/k}^{-1} \right\}^{-1}. \quad (17.18)$$

Extended Kalman filter for estimating states and enemy inputs: We assume in this section that the function f_k defined in (17.3) is nonlinear but the function h_k defined in (17.4) is linear, i.e.,

$$y_k = H_k x_k + v_k, \quad k \geq 0. \quad (17.19)$$

Since $f_k(x, u)$ is a continuously differentiable function of (x, u) , we apply a linear approximation to the right hand side of (17.3) around $(\hat{x}_{k/k}, \hat{u}_k)$, where $\hat{u}_k = (u_k^B, \hat{u}_{k/k+1}^R)$. This is a good approximation so long as $\|u\|$ is small. Then, by applying the Kalman filter described in Proposition 5 to this linear (or affine) system and also replacing the state x_k by its estimate $\hat{x}_{k/k}$ and the unknown input u_k^R by its estimate $\hat{u}_{k/k+1}^R$, we obtain the following extended Kalman filter:

$$\bar{x}_{k/k} := A_k \hat{x}_{k/k} + B_k^B u_k^B, \quad (17.20)$$

$$\begin{aligned} \hat{x}_{k+1/k+1} &= \bar{x}_{k/k} + B_k^R \hat{u}_{k/k+1}^R \\ &\quad + K_{k+1}^x \left\{ y_{k+1} - H_{k+1} \left(\bar{x}_{k/k} + B_k^R \hat{u}_{k/k+1}^R \right) \right\}, \end{aligned} \quad (17.21)$$

$$\hat{u}_{k/k+1}^R = K_{k+1}^{u^R} \left(y_{k+1} - H_{k+1} \bar{x}_{k/k} \right), \quad (17.22)$$

where

$$A_k = \frac{\partial f_k}{\partial x}(\hat{x}_{k/k}, \hat{u}_k), \quad (17.23)$$

$$B_k^B = \frac{\partial f_k}{\partial u^B}(\hat{x}_{k/k}, \hat{u}_k), \quad (17.24)$$

$$B_k^R = \frac{\partial f_k}{\partial u^R}(\hat{x}_{k/k}, \hat{u}_k), \quad (17.25)$$

$$(17.26)$$

where the Kalman gain matrices and the estimation error covariance matrix has the same form as in Proposition 5.

Extended Kalman filter algorithm:The block diagram of the extended Kalman filter for estimating state and enemy input, is shown in Fig. 17.1. This diagram also shows the continuous-time plant and how the filter is connected to the plant.

The inputs to the filter are the sampled output vector y_k of the plant and the sampled input vector u_k^B for the friendly Blue unit. The outputs of the filter are respective estimates, $\hat{x}_{k/k}$ and $\hat{u}_{k/k+1}^R$, of the state vector x_k and the enemy input vector u_k^R . The flowchart of the algorithm is given in Fig. 17.2. As the original filter due to Darouach et al. was devised for linear, time-invariant discrete-time plants, we employ samplers and an extension of their Kalman filter to a nonlinear time-varying system. We thus linearized our model around an estimated nominal trajectory and discretized it. We then applied the Kalman filter algorithm to the linearized discretized model.

Extended Kalman filter for estimating state x_k and enemy input u_k^R

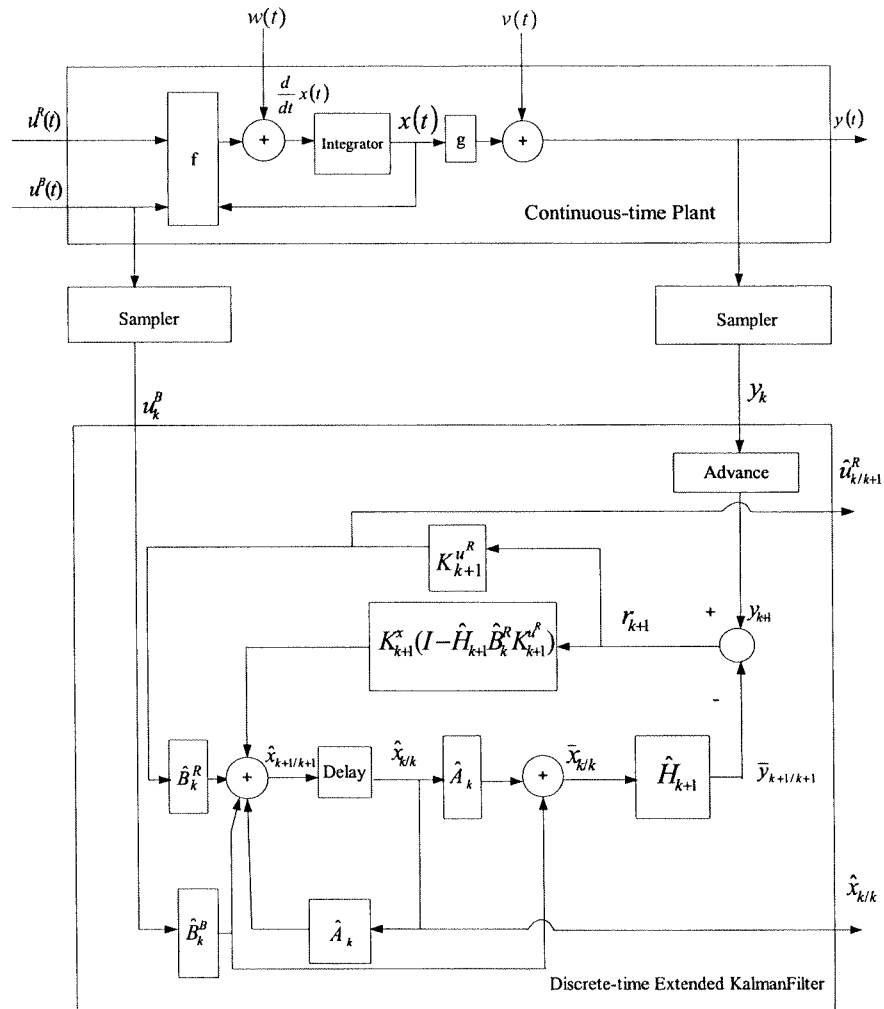


Figure 17.1: Block diagram of the extended Kalman filter

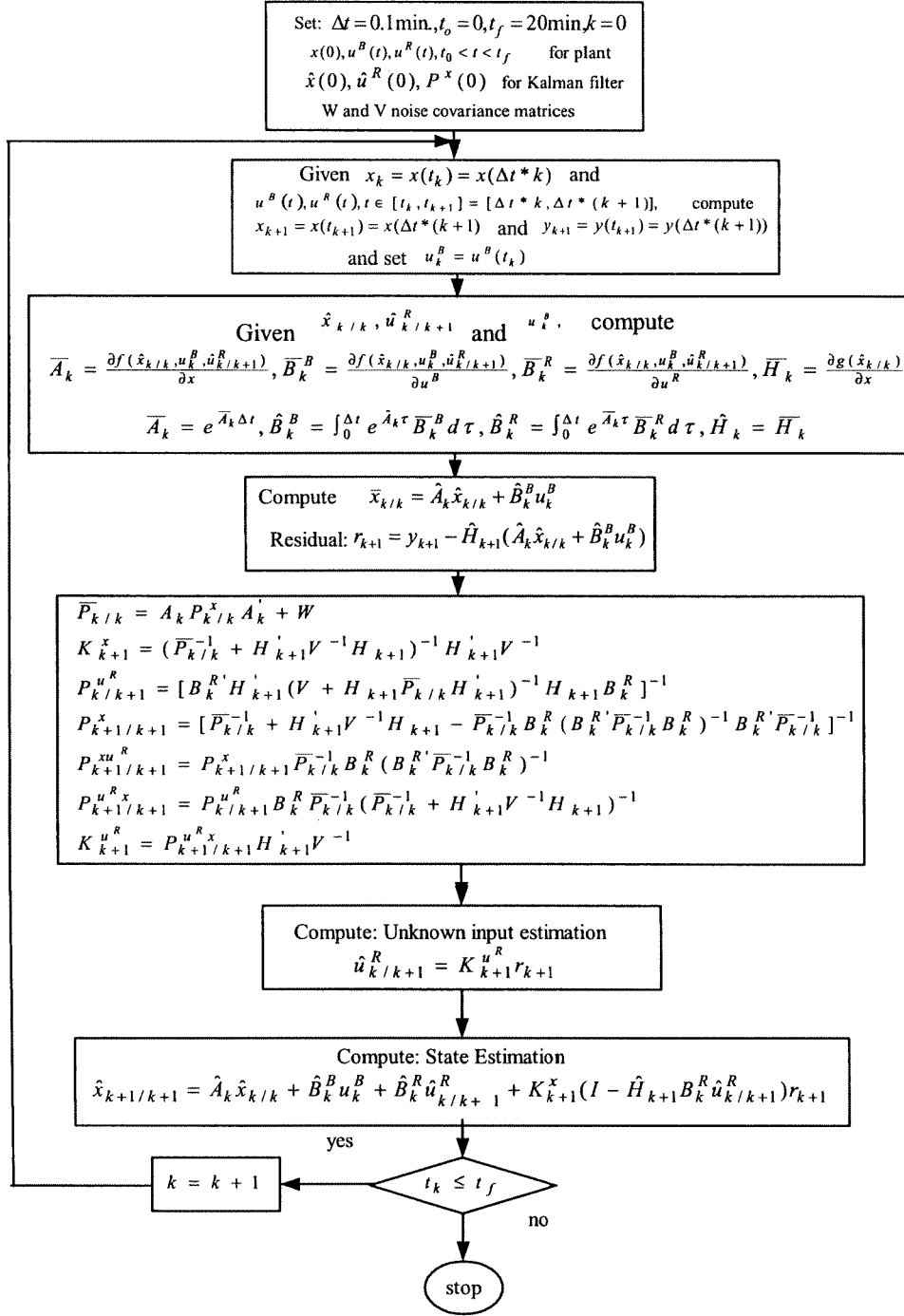


Figure 17.2: The flow chart of the estimation algorithm.

Differential game: The overall game is expressed as the following minmax problem:

$$J^* = \min_{u^B} \max_{u^R} J(u^B, u^R), \quad (17.27)$$

where the Red force tries to maximize the payoff function $J(u^B, u^R)$ and the Blue force tries to minimize the same payoff function. Here the problem is defined over the time interval $[t_0, t_f]$ between the initial time t_0 and the terminal time t_f . The optimal value J^* of the payoff function $J(u^B, u^R)$ is called the value of the game.

We assume that each unit is homogeneous. By this we mean that each bomber unit consists of bombers of the same type, each ground troop unit consists of ground troops of the same type, and so on. In other words, each unit consists of platforms of the same type. The platforms we may consider are bombers, SAM missile launchers, electronic jammers, weasels, fighter-interceptors, personnel carriers and tanks.

We report the simple cases of our differential games in this report: The Blue and Red forces have one, two and four units each depending on the scenario type. Both units start with 10 platforms. For example, in the scenario cross11, the Blue unit (**B1**) starts with 10 interceptors and the Red unit (**R1**) starts with 10 bombers.

The friendly (Blue) unit control is based on the optimum linear feedback of the estimated state around the Nash solution. The adversary (Red) unit control input may be given manually by a human operator. The goal in the game is as follows.

The Blue interceptors try to destroy as many Red bombers as possible and to reach their own respective destinations, and the Red bombers try to preserve their own platforms and to reach their own respective destinations.

If the estimated state vector deviates from the current Nash equilibrium solution, the iteration stops and a new Nash equilibrium solution is recalculated over the remaining time period.

The numerical method for finding the Nash equilibrium solution is an iterative process in which a linear-quadratic approximation of the original game is successively solved using the Riccati equation approach [7].

17.4 Experiment scope and setup

The experimental setup for the game theoretic controller with the Kalman filter is given in Fig. 17.3. The algorithm has been tested for the military air operation model, which is nonlinear and continuous-time. The dynamic model of air operations for the military, and the formulation of the problem of controlling its missions as a differential game are presented in [6]. As the original Kalman filter [2] was devised for linear discrete-time plants, we introduce samplers and its extension to a nonlinear system. Our model is thus linearized around a nominal trajectory and discretized, and then the algorithm is run. We consider the simplest case in this paper: The Blue and Red forces have one unit each. Both units start with 10 platforms. In fact, the Blue unit (**B1**) starts with 10 interceptors and the Red unit (**R1**) starts with 10 bombers.

In all the graphs to follow, the solid line represents the actual values and the dotted line represents the estimated values.

In our simulations, the extended Kalman filter takes as inputs the noisy observations of the position of the Blue force, (ξ_1^B, ξ_2^B) , the number of Blue platforms, η^B , the position of the Red force, (ξ_1^R, ξ_2^R) and Blue input π^B , and it yields as output the estimates for the same variables and, in addition, the number of Red platforms, η^R . We note that the filter gets as input neither the number of Red platforms, η^R nor the enemy input, u^R .

The control inputs for the Blue and Red units are the respective velocities (μ_1^B, μ_2^B) and (μ_1^R, μ_2^R) , and the respective firing intensities (π^B, π^R) .

If the Blue and Red units are not engaged with each other, there is no need to estimate the Red firing intensity π^R , so it is estimated only when the engagement occurs. This was implemented as follows. While the distance between the Blue and Red forces is large, the sensitivity of the number of Blue platforms

η^B to the firing intensity π^R of the Red force is weak and the observability rank condition is almost unsatisfied making the Kalman filter ineffective. Hence, while the distance between the Blue and Red forces is large, we do not estimate the firing intensity π^R of the Red force, but we estimate only the velocity control (μ_1^R, μ_2^R) and the entire state x . In this case, we assume that the Red force would not fire: $\pi^R = 0$.

We tested the following three levels of noise: the high level noise: 1% of the operating value for the blue states, and 5% of the operating value for the red states; the medium size noise: 0.5% of the operating value for the blue states, and 2.5% of the operating value for the red states; the low level noise: 0.25% of the operating value for the blue states, and 1.2% of the operating value for the red states.

GAME-THEORETIC CONTROLLER WITH KALMAN FILTER

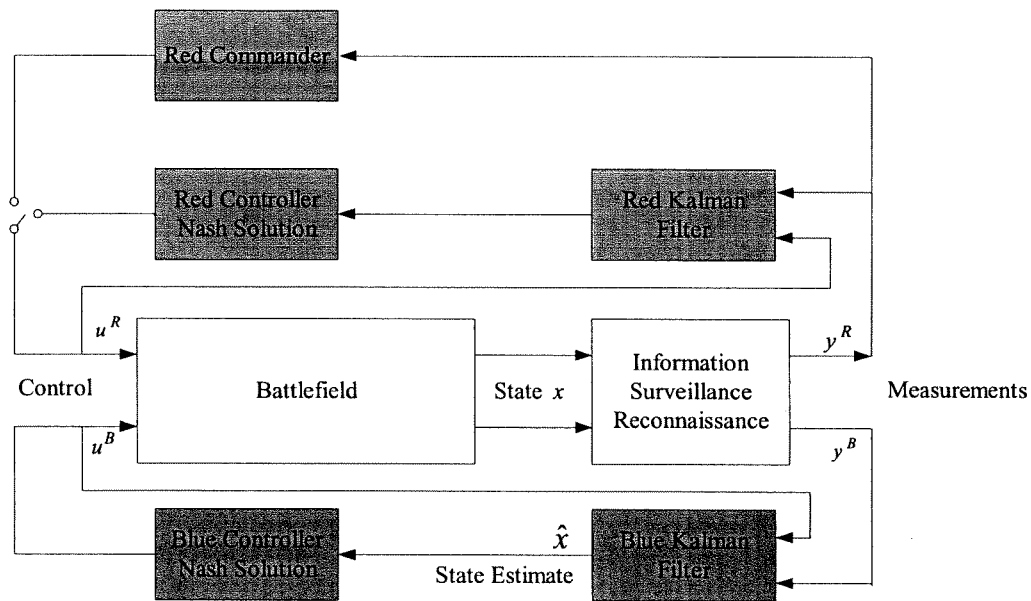


Figure 17.3: The closed-loop game theoretic controller combined with the Kalman filter.

We did experiments for two kinds of scenarios: cross11 and cross23.

We conducted simulations for cross11 for the following 3 cases:

- (1) Observation noise only, i.e., no process noise;
- (2) Process noise only, i.e., no observation noise;
- (3) Both process and observation noise.

For each case, we tested 3 levels of strength: low, medium, and high. In this report, only the results for the high level noise for the above three cases are presented for cross11 scenario.

On the other hand, we conducted simulations for cross23 scenario for only low sensor noise case.

Since the number of platforms for the Red force, η^R , is not observed, a small error in the estimation may occur. The estimated enemy inputs (velocities and firing intensity) are used only for the state estimation but not for feedback control. Therefore, the fluctuations in the input estimations do not cause much error in the overall performance of the controller.

In the following graphs for our simulations, the solid line represents the actual(exact) value and the dotted line represents the estimated value.

17.5 Experiment Results and Analysis

I- Simulation results for the scenario cross11

In this scenario, the Blue force has a bomber and the Red force has an interceptor. For this game, the state variables and the control inputs vary as in Fig. 17.4-17.8. The observed trajectories and the observed numbers of platforms are corrupted by a sensor noise. The scenario is illustrated by these figures. Fig. 17.4 shows the trajectories of the Blue and Red forces. The Blue forces move from west to east and red forces move from north to south. During the mission an engagement occurs. Due to the engagement, both forces lose some number of platforms as seen in Fig. 17.5. Fig. 17.6 and Fig. 17.8 illustrate the weapons used in the mission and the firing intensities, respectively. Firing takes place when both forces meet each other. The speed controls are depicted in Fig. 17.7.

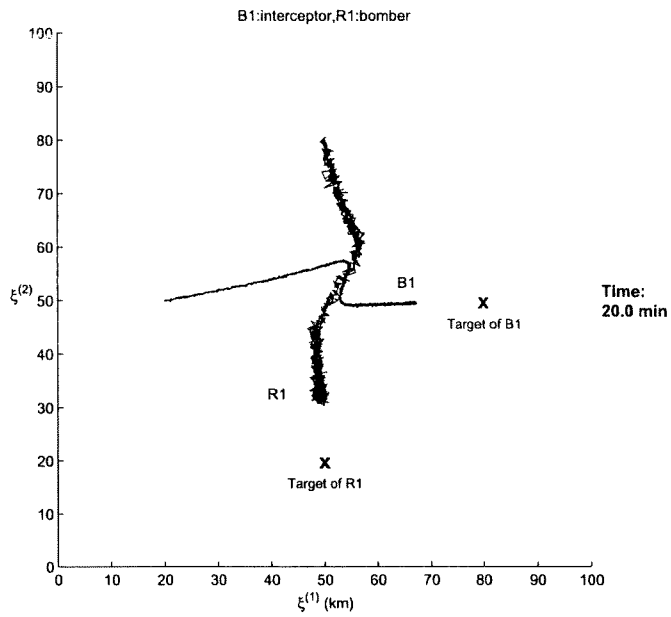


Figure 17.4: Observed Trajectories of Units

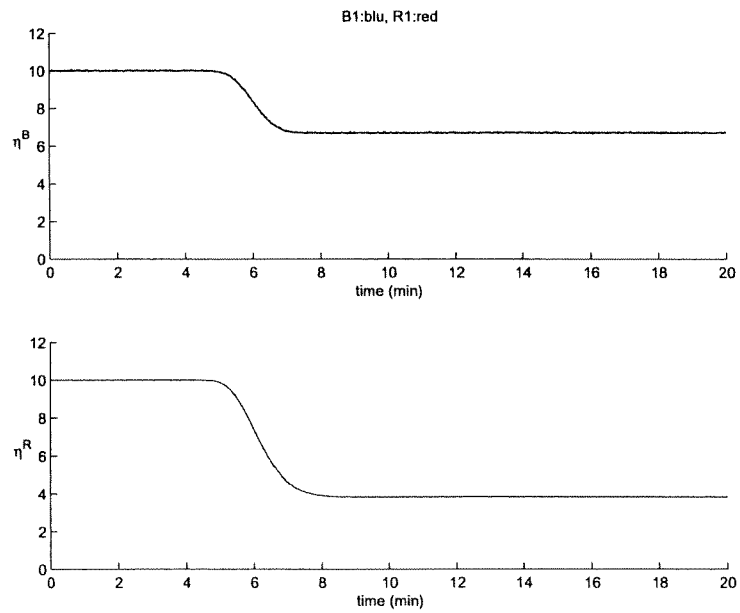


Figure 17.5: Observed Numbers of Platforms

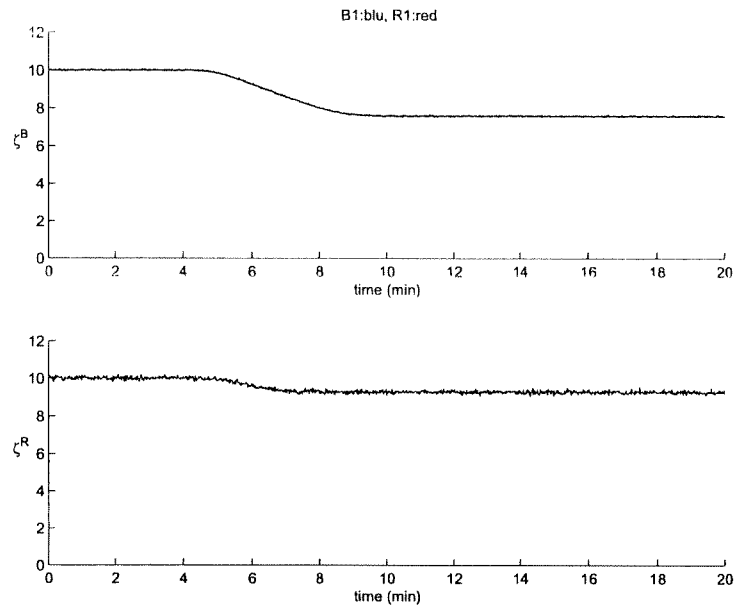


Figure 17.6: Weapons per platform

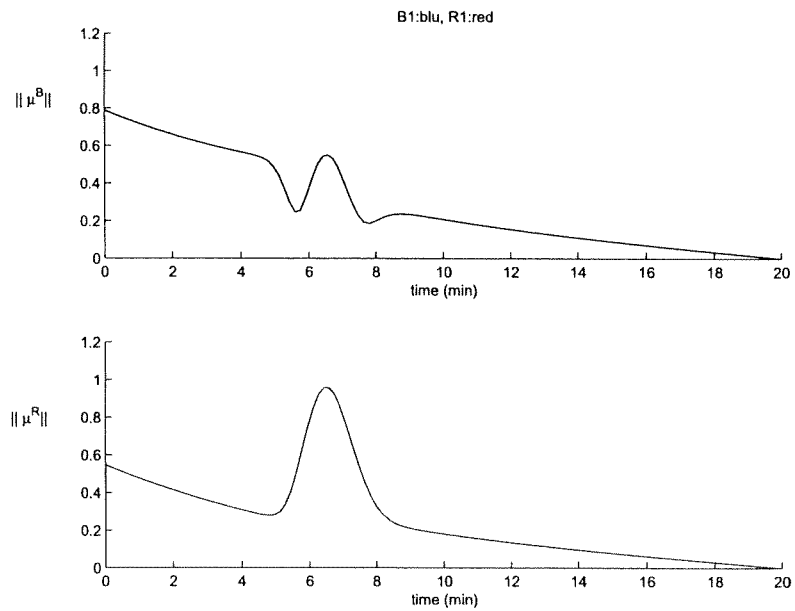


Figure 17.7: Speed Controls

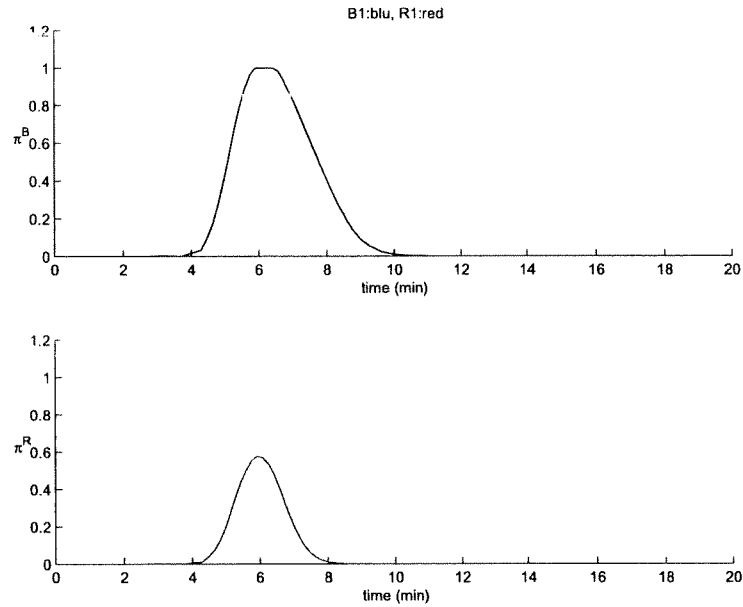


Figure 17.8: Fire Intensities

For high, medium, low sensor noise, Blue state, Red state and Red inputs are presented in Figs. 17.9-17.17. The simulation results show that the error in the Kalman filter estimate is acceptable, though the deviation in the estimates of the Blue number of platforms is considerable.

The first two graphs in each Blue and Red state figures are the exact, observed and estimated values of the x and y positions of the Blue and Red forces. The third graph is the exact, observed and estimated values of the number of platforms. Note that, we do not plot the observed number of red platforms, because it is not observed.

The first two graphs in each red inputs figures are the exact and estimated values of the x and y speeds of the Red force. The third graph is the exact and estimated values of the Red firing intensity. Although the estimates of the speeds and firing intensity are fluctuating highly, their effects on state estimation is small.

Case (1) Observation noise only, no process noise

Process noise: $w = 0$

Covariance for the sensor noise: $V = \text{diag}[(0.5)^2, (0.5)^2, (0.08)^2, (2.5)^2, (2.5)^2]$

Sensor noise is added to the state vector $[\xi_1^B, \xi_2^B, \eta^B, \xi_1^R, \xi_2^R]$.

The Blue states, Red states and enemy inputs (actual and estimated) are presented in Fig. 17.9, Fig. 17.10, and Fig. 17.11, respectively.

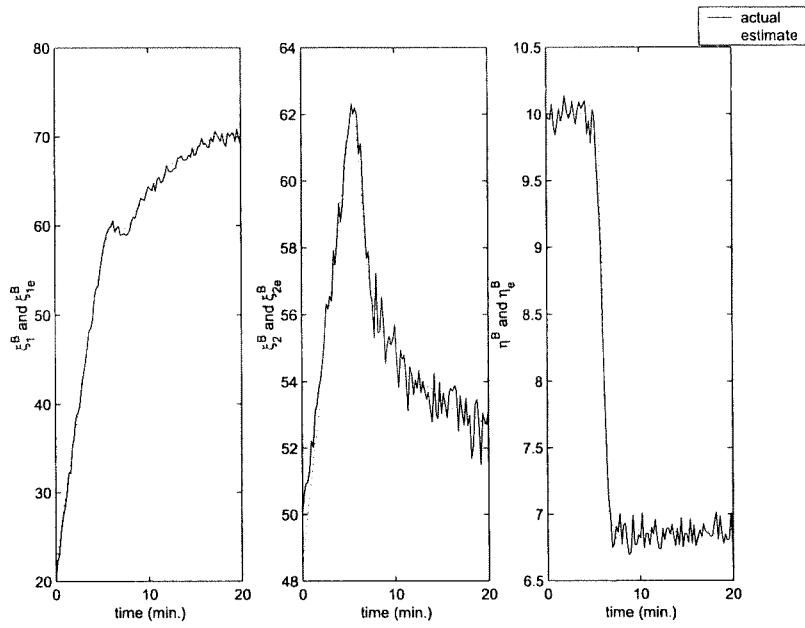


Figure 17.9: Blue states for high sensor noise

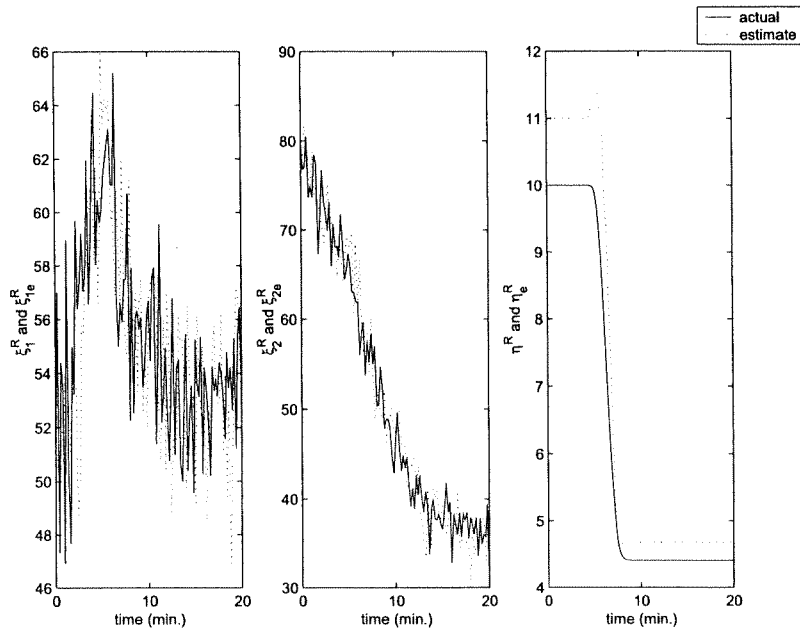


Figure 17.10: Red states for high sensor noise

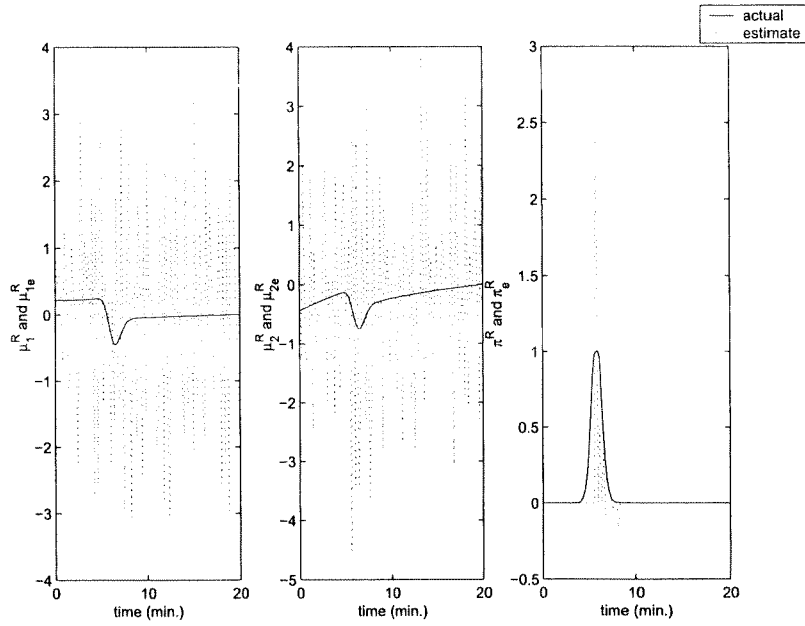


Figure 17.11: Red inputs for high sensor noise

Case (2) Process noise only, no observation noise

Covariance for the process noise: $W = \text{diag}[(0.5)^2, (0.5)^2, (0.08)^2, (2.5)^2, (2.5)^2]$

Process noise is added to the state vector $[\xi_1^B, \xi_2^B, \eta^B, \xi_1^R, \xi_2^R]$

Sensor noise: $v = 0$

The Blue states, Red states and enemy inputs (actual and estimated) are presented in Fig. 17.12, Fig. 17.13, and Fig. 17.14, respectively.

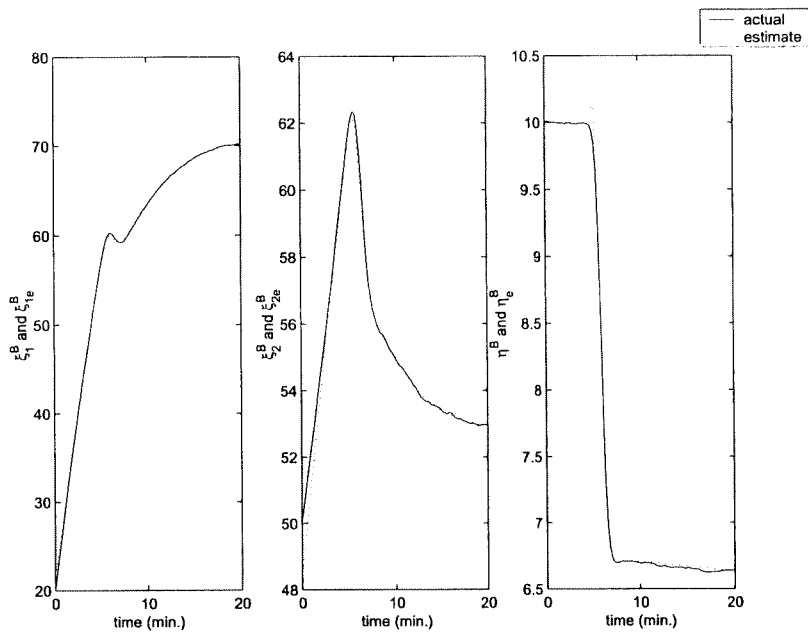


Figure 17.12: Blue states for high process noise

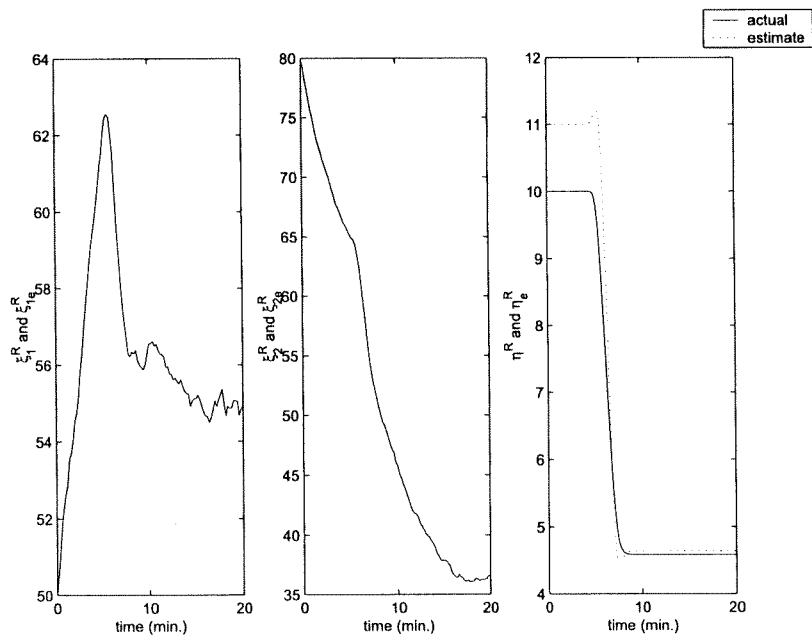


Figure 17.13: Red states for high process noise

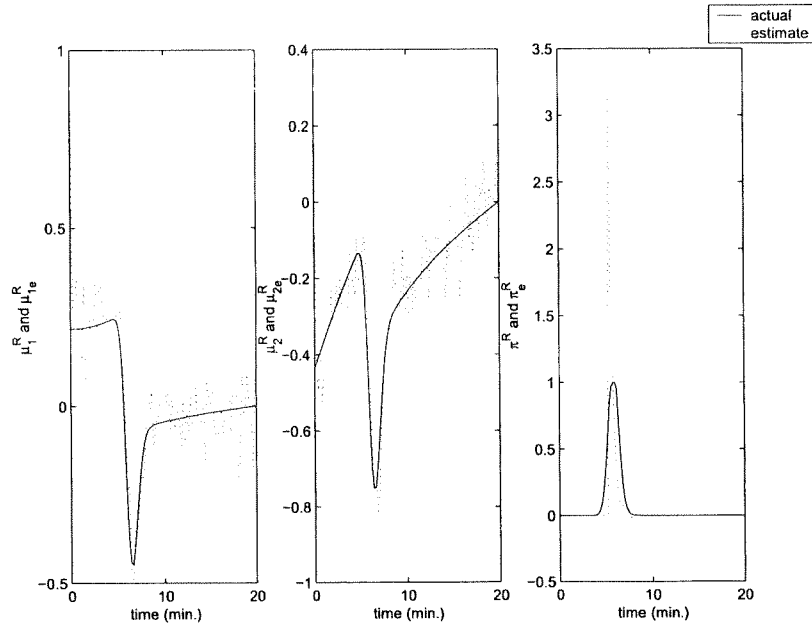


Figure 17.14: Red inputs for high process noise

Case (3) Both observation and process have noise

Covariance for the process noise: $W = \text{diag}[(0.5)^2, (0.5)^2, (0.08)^2, (2.5)^2, (2.5)^2]$

Process noise is added to the state vector $[\xi_1^B, \xi_2^B, \eta^B, \xi_1^R, \xi_2^R]$

Covariance for the sensor noise: $V = \text{diag}[(0.5)^2(0.5)^2(0.08)^2(2.5)^2(2.5)^2]$

Sensor noise is added to the state vector $[\xi_1^B, \xi_2^B, \eta^B, \xi_1^R, \xi_2^R]$

The Blue states, Red states and enemy inputs (actual and estimated) are presented in Fig. 17.15, Fig. 17.16, and Fig. 17.17, respectively.

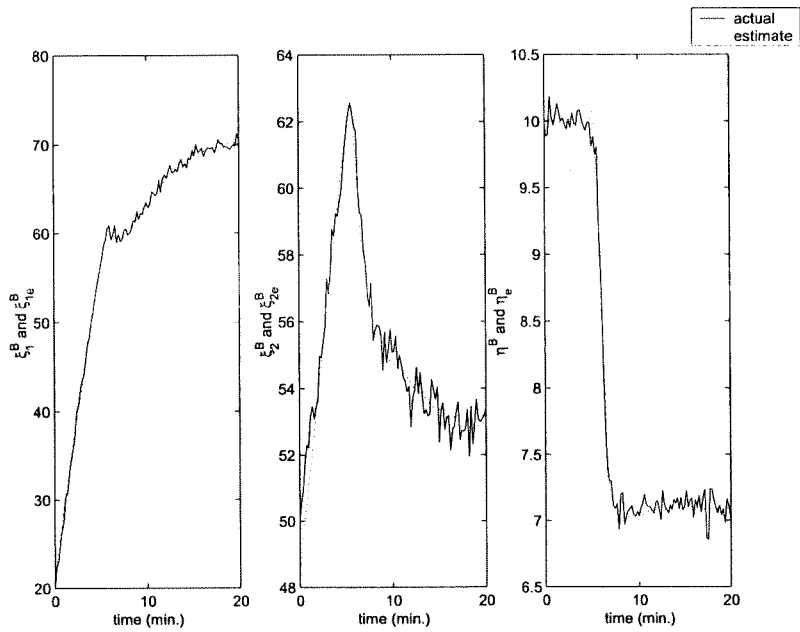


Figure 17.15: Blue states for high process and sensor noises

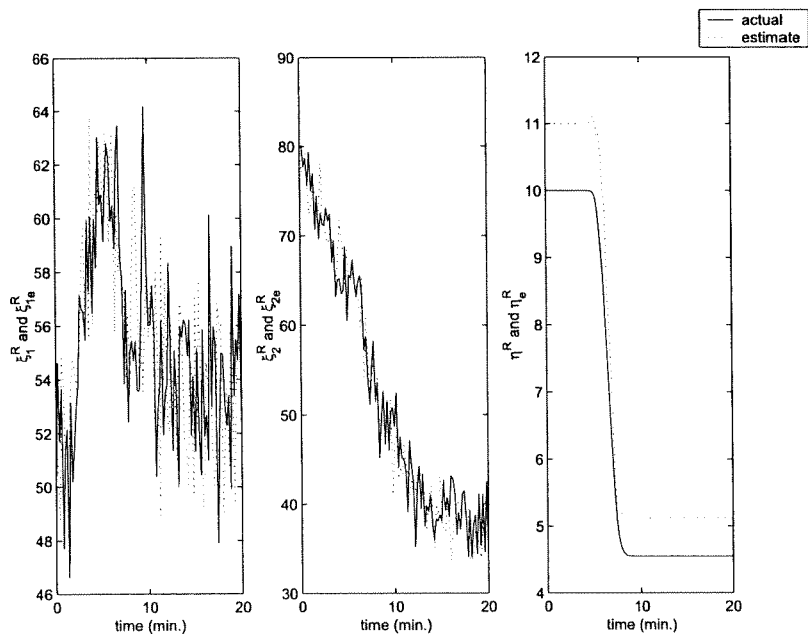


Figure 17.16: Red states for high process and sensor noises

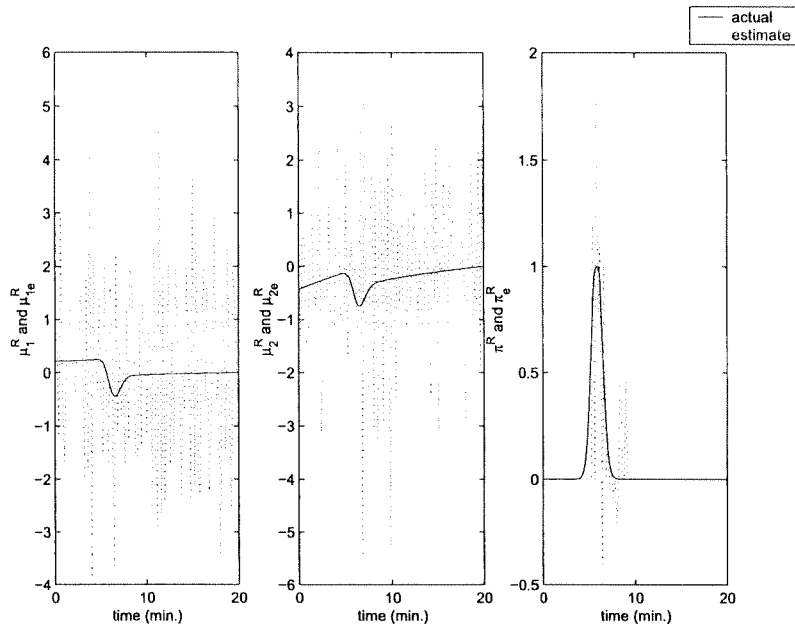


Figure 17.17: Red inputs for high process and sensor noise

II- Simulation results for the scenario cross23

In this scenario, the Blue force has an interceptor and a bomber, and the Red force has two interceptors and a ground troop. Blue interceptor and bomber head to the Red ground troop.

For this game, the state variables and the control inputs vary as in Fig. 17.18-17.22 (Note that instead of the observed results, the exact results are presented here). The scenario is illustrated by these figures. Fig. 17.18 shows the trajectories of the Blue and Red forces. The Blue bomber and Blue interceptor move from west to east towards the Red ground troop. One Red interceptor moves from north to south to its destination and the other Red interceptor moves from south to north and also defends the ground troop. During the mission an engagement occurs. Due to the engagement, both forces lose some number of platforms as seen in Fig. 17.19. Fig. 17.20 and Fig. 17.22 illustrate the weapons used in the mission and the firing intensities, respectively. Firing takes place when both forces meet each other. The speed controls are depicted in Fig. 17.21.

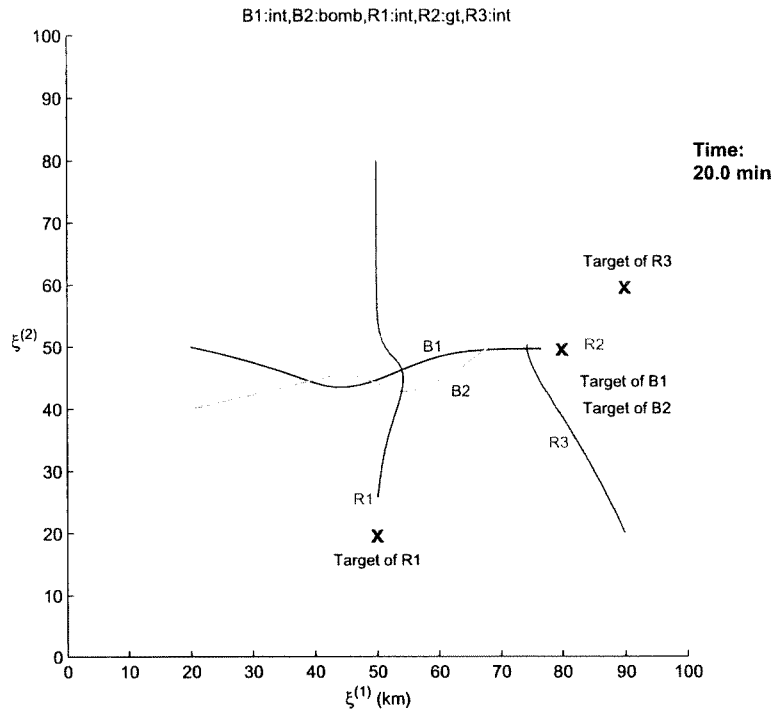


Figure 17.18: Trajectories of Units

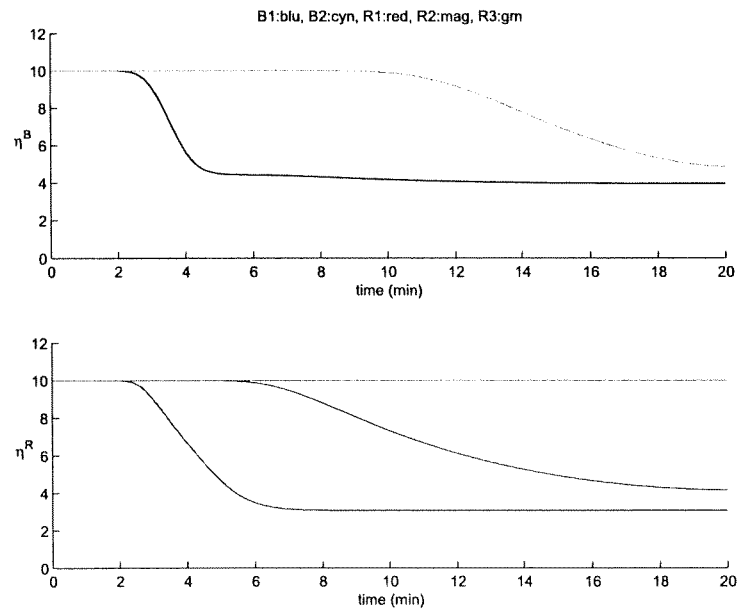


Figure 17.19: Number of Platforms

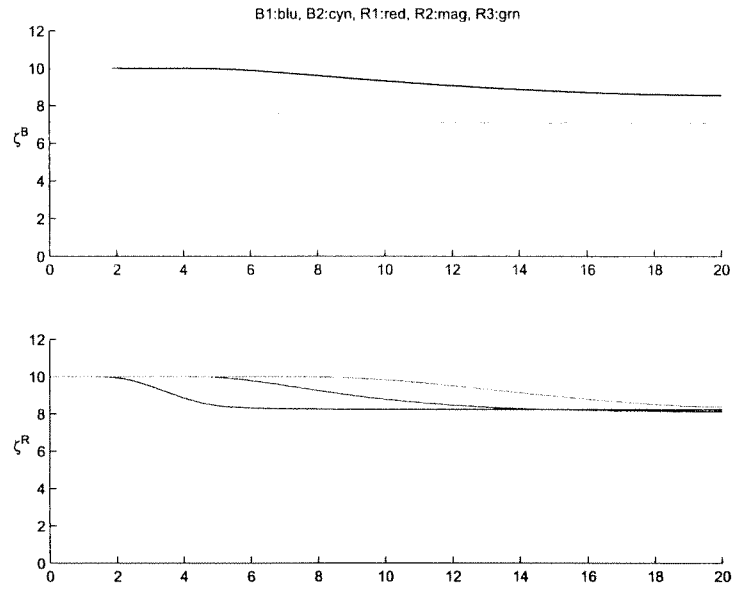


Figure 17.20: Weapons per platform

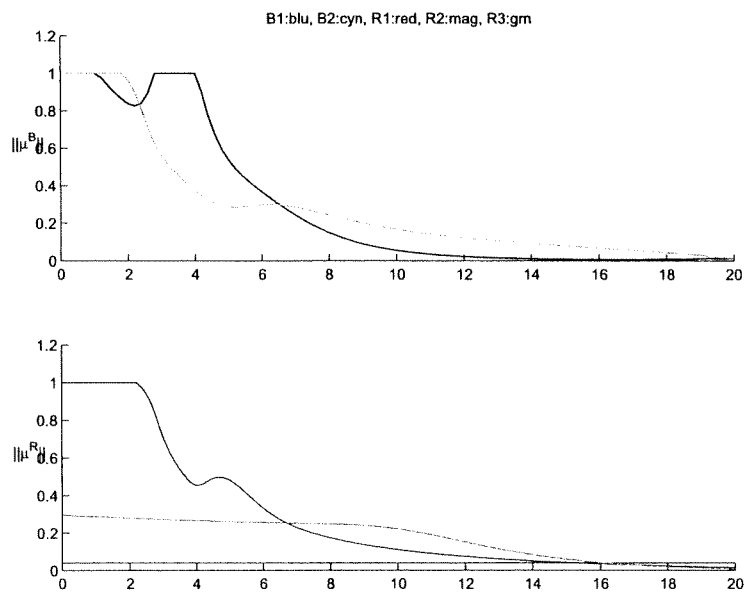


Figure 17.21: Speed Controls

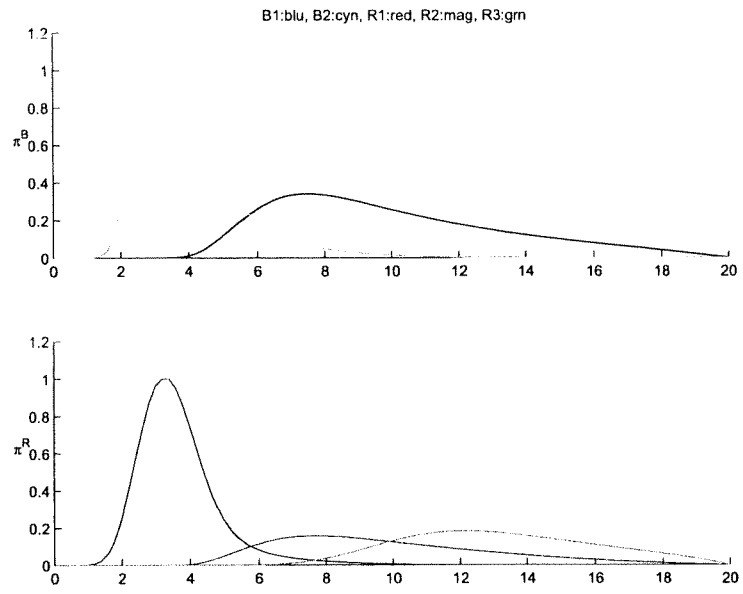


Figure 17.22: Fire Intensities

Low level observation noise only, no process noise

The Blue states, Red states and enemy inputs (exact, observed, and estimated) are presented in Fig. 17.23-17.30, respectively.

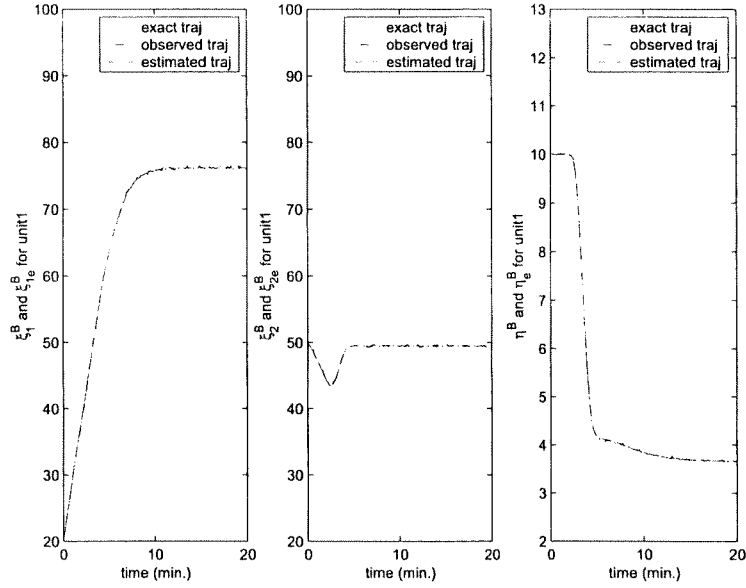


Figure 17.23: Blue states for high sensor noise for unit1

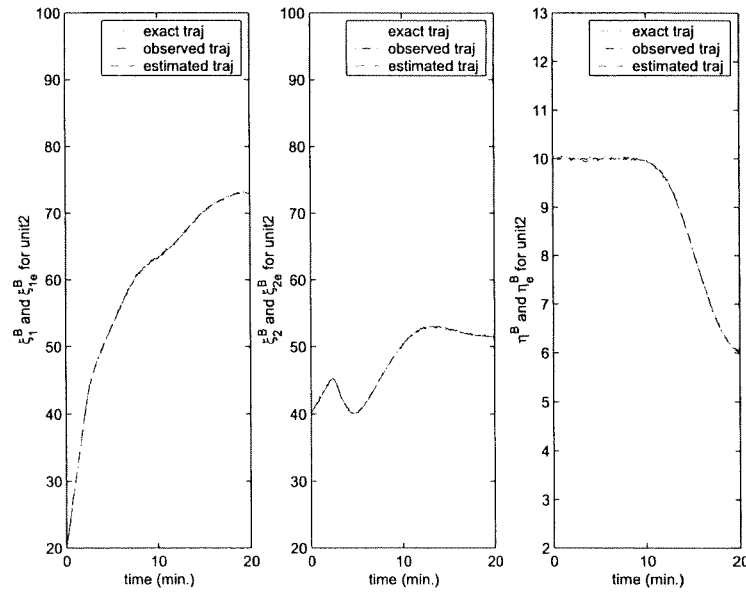


Figure 17.24: Blue states for high sensor noise for unit2

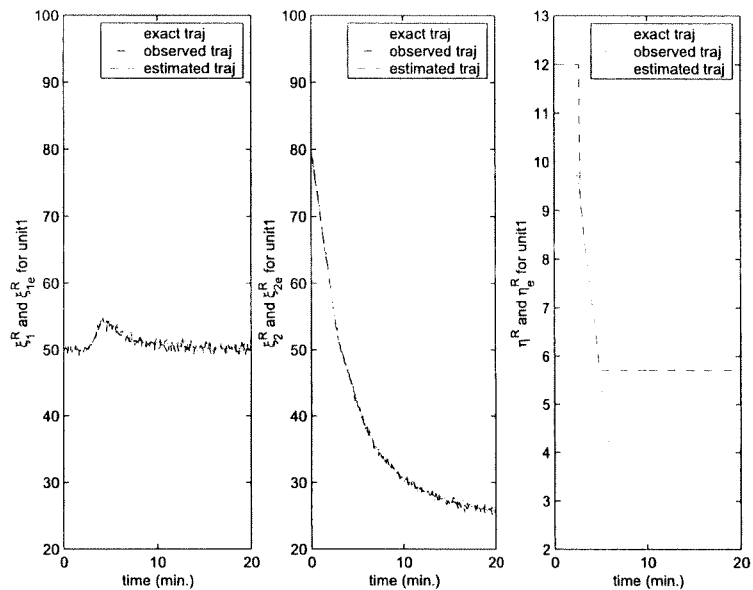


Figure 17.25: Red states for high sensor noise for unit1

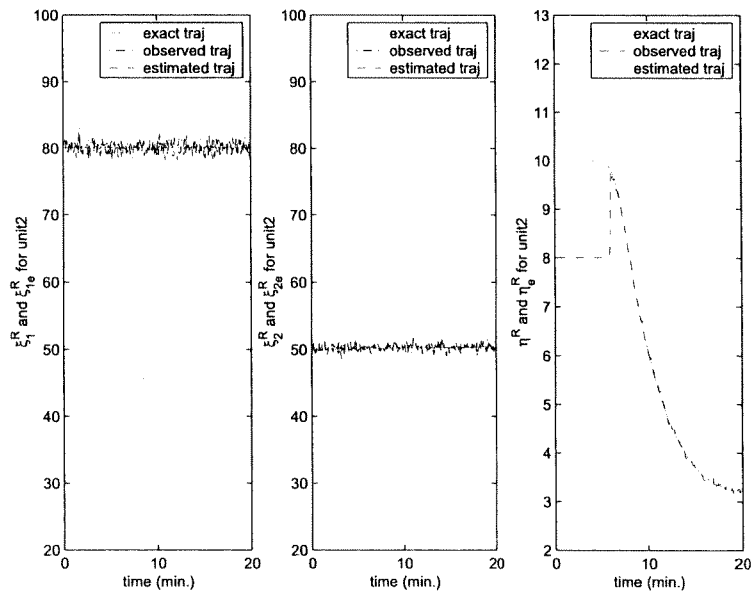


Figure 17.26: Red states for high sensor noise for unit2

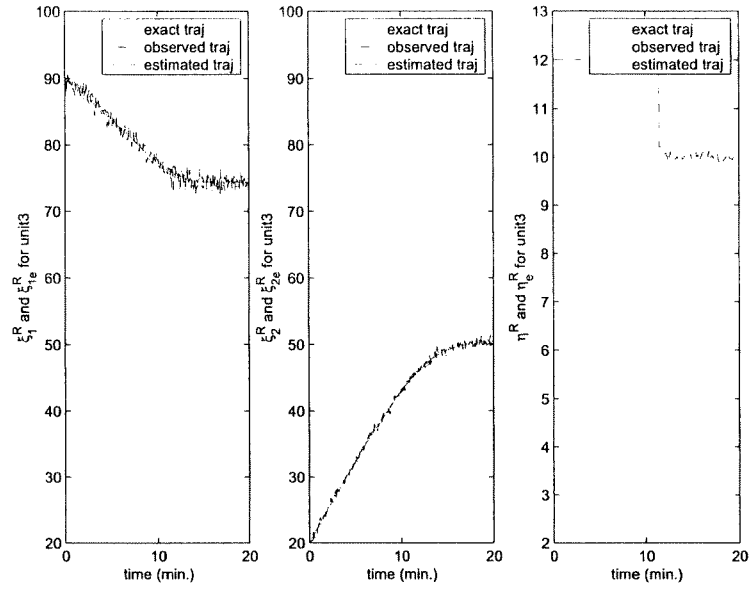


Figure 17.27: Red states for high sensor noise for unit3

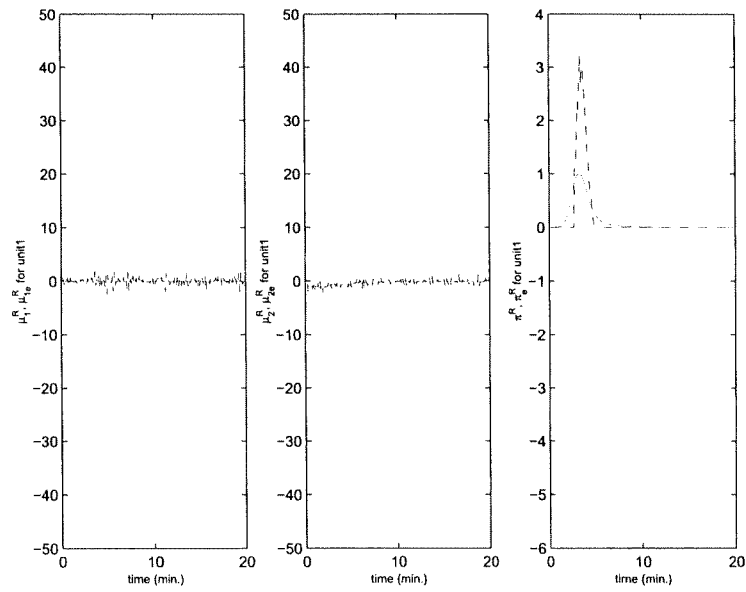


Figure 17.28: Red inputs for high sensor noise for unit1

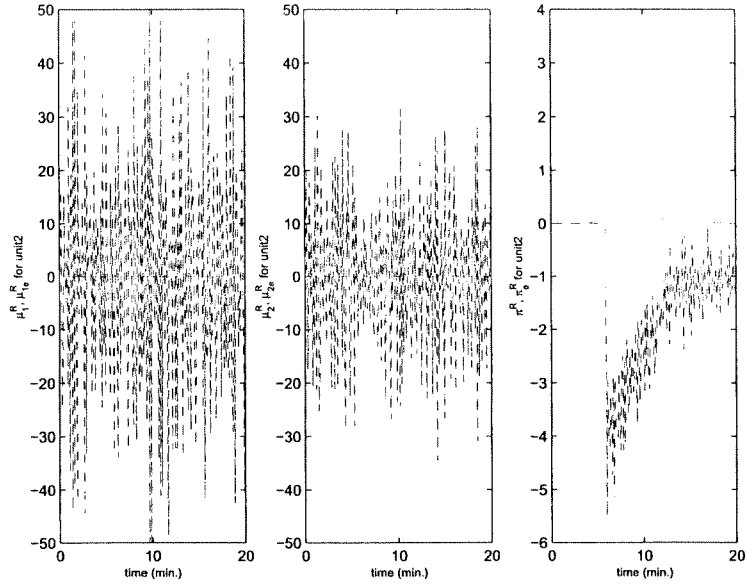


Figure 17.29: Red inputs for high sensor noise for unit2

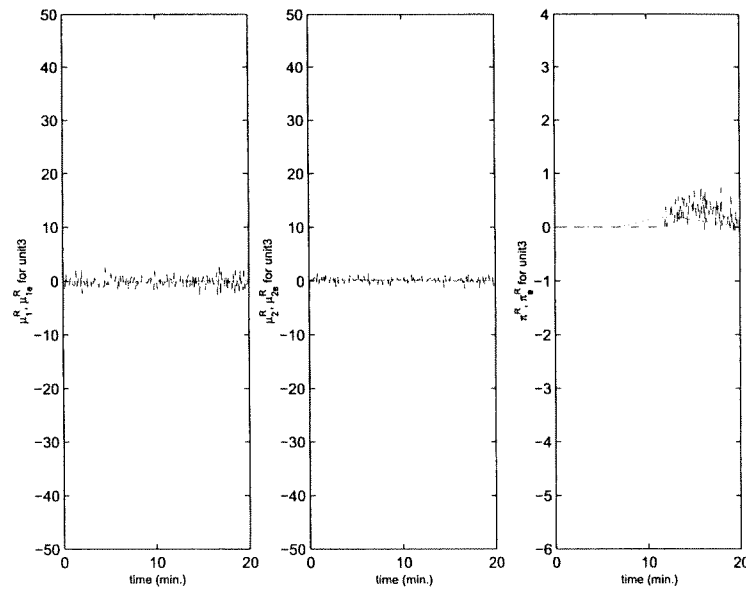


Figure 17.30: Red inputs for high sensor noise for unit3

17.6 Conclusions

In this report, we have presented how the extended Kalman filter algorithm for state estimation is used in a differential game, which models the air operations of two opposing forces. The air operation model is nonlinear and time-varying. As the filter due to Darauch et al. is designed for linear time-invariant systems, we have developed an extension of their filter to a nonlinear time-variant continuous system.

The extended Kalman filter presented in this report is capable of estimating the states in the presence of process noise as well as sensor noise in different magnitudes.

We have observed that the game-theoretic controller remained effective when the extended Kalman filter is introduced in the feedback loop. The closed loop filter performance is verified by the simulations for different scenarios.

Bibliography

- [1] K. J. Astrom, *Introduction to Stochastic Control Theory*, Academic Press, New York, New York, 1970.
- [2] M. Darouach, M. Zasadzinski, A. Bassong Onana and S. Nowakowski, "Kalman filtering with unknown inputs via optimal state estimation of singular systems", *Int. J. Systems Sci.* **26**, 2015-2028 (1995).
- [3] W. H. Fleming and R. W. Rishel, *Deterministic and Stochastic Optimal Control*, Springer-Verlag, New York, New York, 1975.
- [4] B. Oksendal, *Stochastic Differential Equations*, fifth edition, Springer-Verlag, Berlin, 1975.
- [5] R. Nikoukhah, A. S. Willsky and B. C. Levy, "Kalman filtering and Riccati equations for descriptor systems", *IEEE Trans. on Automatic Control* **37**, 1325-1342 (1992).
- [6] H. Mukai, Y. Sawada, I. Tunay and P. Girard, "Mission Dynamics Continuous-time Model", *Working Report, Washington University JFACC Team and SAIC*, Department of Systems Science and Mathematics, Washington University (2000).
- [7] H. Mukai et al., "Game-Theoretic Linear-Quadratic Method for Air Mission Control", Proceedings of the IEEE Conference on Decision and Control, *CDC 2000*, 2574-2580 (2000).

Chapter 18

Experiment 18: Method of Characteristics: Addendum

18.1 Executive Summary

The purpose of Experiment 11 was to verify that the solution computed by the Sequential Linear-Quadratic Method (SLQM) was the same as the Nash solution computed by the Method of Characteristics. We verified that the solutions computed by the Sequential Linear-Quadratic Method (SLQM) were indeed the same as the Nash solutions computed by the Method of Characteristics under several scenarios. However, the experiments in Chapter 11 all involved one Blue unit against one Red unit. In Experiment 18, we extend the results in Experiment 11 to a scenario of multi-units against multi-units. Specifically, Experiment 18 tests the Method of Characteristics for the case of three blue units against three red units.

18.2 Purpose of the Experiment

The *purpose* of Experiment 18 is to test whether the Method of Characteristics works for multi-units against multi-units.

18.3 Hypothesis

The method is successful in the case of multi-units against multi-units.

18.4 Methods

The algorithm is described in paper [1].

18.5 Experiment Scope

In this scenario, three blue units (B1: bomber, B2: interceptor, B3: interceptor) encounter three red units (R1: bomber, R2: interceptor, R3: bomber). The dynamics for the characteristics equations have been computed but are not given here explicitly. The parameter values are $a^{Bi} = a^{Ri} = 0.05$, $b^{Bi} = b^{Ri} = 0.0$, and $p^{Bi} = p^{Ri} = 0.0$. When control penalties are used, the parameter values are $R_x^{Bi} = R_y^{Bi} = R_x^{Ri} = R_y^{Ri} = 150$, and $R_\pi^{Bi} = R_\pi^{Ri} = 2.5$, for $i=1,2,3$.

18.6 Experiment Results

The results of experiment are shown in the following figures. Figure 18.1 shows the Nash trajectory for the Blue and Red units. Figure 18.2 shows the Nash engagement intensities as well as the Nash number of platforms for the both forces.

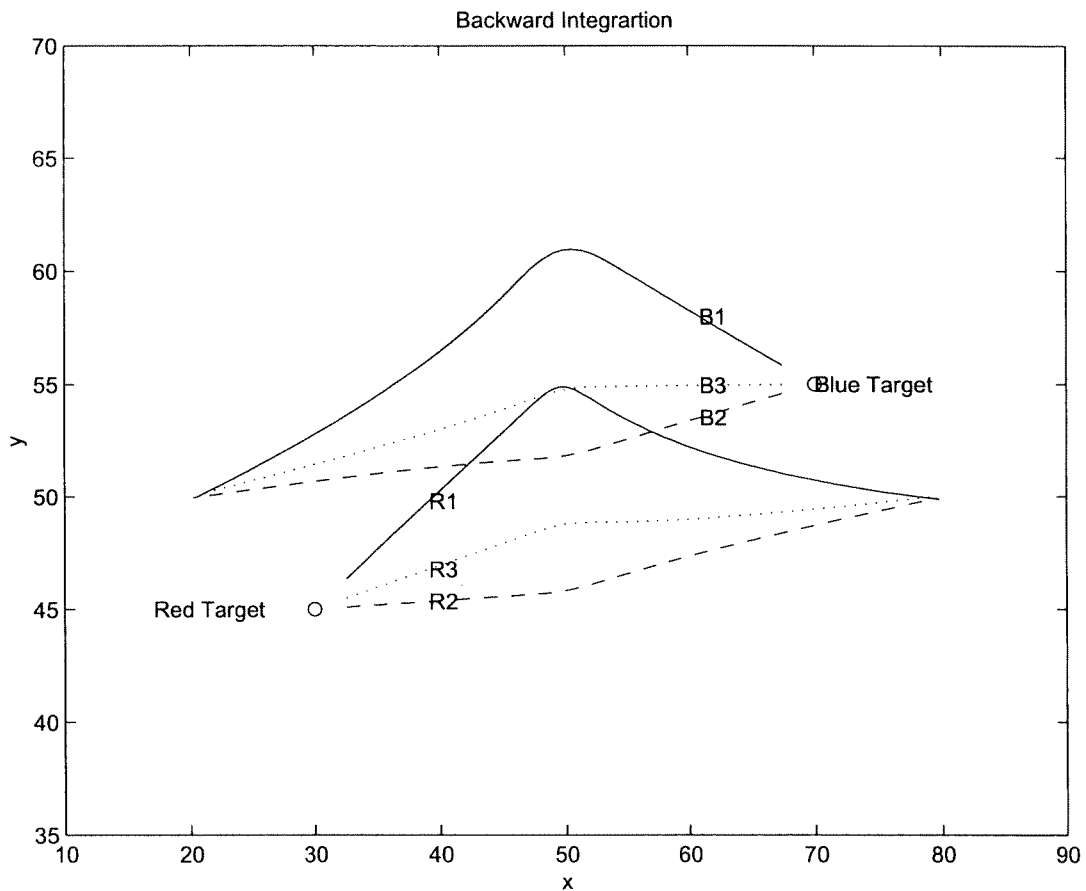


Figure 18.1: Nash Trajectories for Three Units vs. Three Units

18.7 Analysis

The results obtained by the Method of Characteristics for multi-units against multi-units match very closely those obtained by the Sequential Linear-Quadratic Method.

18.8 Conclusion

The Method of Characteristics is a feasible procedure for verifying the results of the Sequential Linear-Quadratic Method.

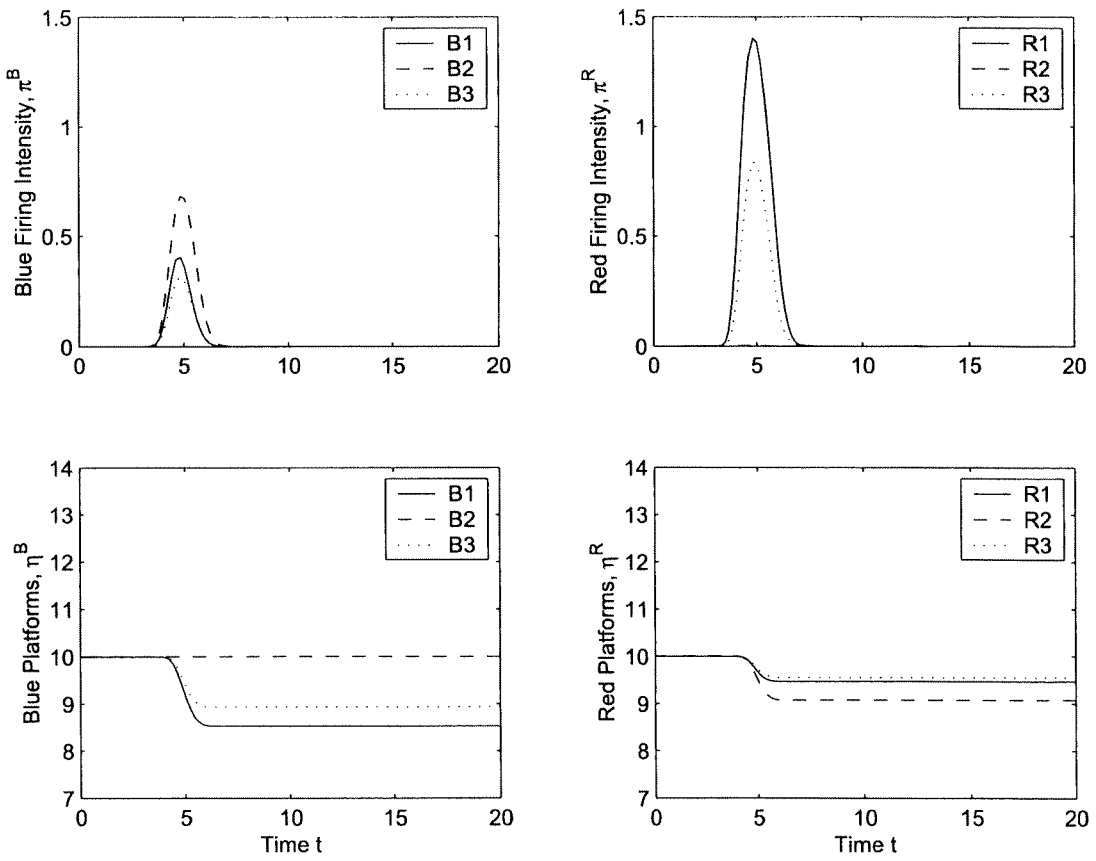


Figure 18.2: Nash Engagement Intensities and Number of Platforms

Bibliography

- [1] I. N. Katz, H. Mukai, H. Schattler, and Mingjun Zhang, *Solution of a differential game formulation of military air operations by the method of characteristics*, Proc. of the 2001 American Control Conference, Washington, D.C., June 2001.

Chapter 19

Experiment 19: New Game Flow Models

19.1 Executive Summary

Military operations can be viewed as a hierarchical structure in which actions are taken by individual units at a low level, based on strategies developed by planners at a high level. In this experiment we consider the situation in which two forces, say the blue and red forces, control a large number of units distributed over a large geographical area. We develop a tool that is useful to high-level planners in simulating and computing the optimal strategy for the two forces. We also report the results of our numerical experiments.

The geographical area in our model is represented by an abstract game board that is divided into cells so that the strength concentration of the blue (resp. red) force in a cell is defined as the number of blue (resp. red) units contained in the cell divided by the area of the cell. The game is concurrent in the sense that both the blue and red forces can move some or all of their respective units simultaneously and continuously during the game.

We formulated the military operation control problem as a differential game over the abstract game board. The differential game consists of a quadratic payoff function and a set of ordinary differential equations describing the system dynamics of the unit distribution over the discretized geographical area (the abstract game board).

In order to solve such a geographically distributed differential game, we developed a computer method for finding a local Nash solution to the adversarial game. The optimum strategy for each team is found using the iterative algorithm called Sequential Linear-Quadratic Method. Experimental results are also presented that demonstrate the validity of this concept.

19.2 Introduction

Military operations, and many types of games, can be characterized by a hierarchical structure in which actions are performed by individual units at the low level, based on strategies developed by planners at the high level [1, 2]. The Game Flow program represents a high-level tool that may be used by planners to simulate, and obtain an optimal strategy for an adversarial game in which two forces, say the blue and red forces, separately control a large number of units distributed over a large geographical area.

The geographical area in our model is represented by an abstract game board that is divided into cells so that the strength concentration of the blue (resp. red) force in a cell is defined as the number of blue (resp. red) units contained in the cell divided by the area of the cell. The game is concurrent in the sense that both the blue and red forces can move some or all of their respective units simultaneously and continuously during the game.

Movement of units is achieved by specifying transport velocities for each pair of contiguous cells, i.e., the rate at which the units are shifted from one cell to the next. At the start of a game, the two forces are assigned an initial strength distribution over the cells in the game board. As the game proceeds, the respective strength distributions evolve in different ways, but the total strength of each force can only decrease due to attrition caused by enemy attacks, mechanical breakdown, etc.

The game is carried out for a specified amount of time, with the three processes of the game, i.e., unit movement, attack and attrition (other than that associated with attack), evolving uninterrupted in parallel for the duration of the game.

The goal of each force is to reach the end of the game with a minimum loss of their own strength, while inflicting maximum damage to the opposing force. Also, each force may assign higher payoff values (larger weights) to some of the cells in the game board than to other cells, so a higher score might be earned by finishing the game with heavier strength concentration in more valuable cells. Finally, movement of units across the game board typically costs valuable energy, so the energy expenditure for movement is also included in the payoff function for each force.

The control strategy of each force is defined as the transport velocities that a force assigns to all the pairs of adjacent cells in the game board during a game. To compute the strategy of each force, we use a game-theoretic solution engine (i.e., the *Sequential Linear-Quadratic* algorithm) [4] that is designed to converge to a Nash solution of the game.

19.3 Mathematical Model

In order to derive equations which model the evolution for the strength distributions of the two forces on the game board, we use an Eulerian approach (see, e.g., [3]).

The game board is divided into a grid with uniform mesh size $\Delta x \times \Delta y$. The evolution of the strength concentration of the blue force in a cell can be computed by balancing the net inflow of blue units through the boundaries of the cell minus the loss of units due to attrition.

The superscript B (resp. R) is affixed to parameters and variables that belong to the blue (resp. red) force. The coordinates of a typical cell in the grid are indicated by the subscripts mn . The strength concentration of the blue force inside the mn -th cell at time t is represented by the symbol ρ_{mn}^B , and corresponds to the number of blue units in the cell divided by the mesh area $\Delta x \Delta y$. The symbol μ_{mn}^B represents the transport velocity in the x direction for the blue units from the mn -th to the $(m+1)n$ -th cell. The symbol ν_{mn}^B represents the transport velocity in the y direction for the blue units from the mn -th to the $m(n+1)$ -th cells. Similarly, the symbols $\mu_{m-1,n}^B$ and $\nu_{m,n-1}^B$ represent the transport velocity for the blue units from the $(m-1)n$ -th and $m(n-1)$ -th cells, respectively, to the mn -th cell. Here the transport velocities, μ and ν , represent distance travelled per length of the cells per unit of time.

The net inflow of blue units into the mn -th cell can be written as

$$\begin{aligned} & - (\rho_{mn}^B H^+[\mu_{mn}^B] + \rho_{m+1,n}^B H^-[\mu_{mn}^B]) \mu_{mn}^B \\ & + (\rho_{m-1,n}^B H^+[\mu_{m-1,n}^B] + \rho_{mn}^B H^-[\mu_{m-1,n}^B]) \mu_{m-1,n}^B \\ & - (\rho_{mn}^B H^+[\nu_{mn}^B] + \rho_{m,n+1}^B H^-[\nu_{mn}^B]) \nu_{mn}^B \\ & + (\rho_{m,n-1}^B H^+[\nu_{m,n-1}^B] + \rho_{mn}^B H^-[\nu_{m,n-1}^B]) \nu_{m,n-1}^B, \end{aligned}$$

where H^+ is the standard Heaviside function, which gives a value of one for a positive argument and zero for a negative argument, and H^- is defined as $H^- \equiv 1 - H^+$. Notice that the strength, ρ , is a positive quantity by definition.

The attrition process due to itself causes a decrease in strength concentration inside the cell, proportional to the instantaneous value of the strength concentration in the cell:

$$\alpha_{mn}^B \rho_{mn}^B,$$

where α_{mn}^B is called the local self-attrition parameter for the blue units in the mn -th cell.

The additional loss of strength due to attack from nearby red units occurs at a rate determined by the surface convolution of an efficiency-of-attack function ϕ_{kl}^R , and the strength of the red units distributed

in the neighborhood:

$$\sum_k \sum_l \phi_{m-k, n-l}^R \rho_{kl}^R \rho_{mn}^B.$$

With the adoption of the efficiency-of-attack function, it will be possible to model the dependence of attack efficiency on the distance between the attacker and the attacked.

Adding the above three terms, we obtain the rate of increase of the strength concentration of the blue force inside the mn -th cell:

$$\begin{aligned} \dot{\rho}_{mn}^B &= -(\rho_{mn}^B H^+[\mu_{mn}^B] + \rho_{m+1, n}^B H^-[\mu_{mn}^B]) \mu_{mn}^B \\ &+ (\rho_{m-1, n}^B H^+[\mu_{m-1, n}^B] + \rho_{mn}^B H^-[\mu_{m-1, n}^B]) \mu_{m-1, n}^B \\ &- (\rho_{mn}^B H^+[\nu_{mn}^B] + \rho_{m, n+1}^B H^-[\nu_{mn}^B]) \nu_{mn}^B \\ &+ (\rho_{m, n-1}^B H^+[\nu_{m, n-1}^B] + \rho_{mn}^B H^-[\nu_{m, n-1}^B]) \nu_{m, n-1}^B \\ &+ \alpha_{mn}^B \rho_{mn}^B \\ &+ \sum_k \sum_l \phi_{m-k, n-l}^R \rho_{kl}^R \rho_{mn}^B. \end{aligned} \quad (19.1)$$

This equation is for the blue force. A similar equation is derived for the red force.

19.4 Solution of the System Equations

In this work we only consider rectangular game boards where there is no flow of units across the boundaries of the board (Neumann boundary conditions). Furthermore, the game board is divided into rectangular cells arranged in a regular grid pattern, and the strength concentrations for the two forces, for each cell, are defined at the center of the cell. Velocities are defined on the boundaries between each pair of adjacent cells. Figure 19.1 illustrates the numbering scheme for the cells in a 4×4 grid.

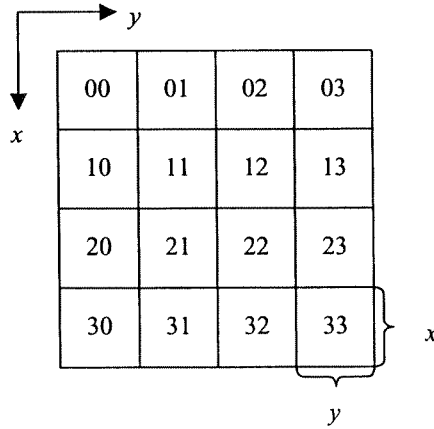


Figure 19.1: Cell numbering scheme.

Since equations similar to equation (19.1) can be formulated for each cell in the game area, we need to solve a system of nonlinear differential equations with the following general form:

$$\begin{aligned} \dot{\rho}^B &= L^B[\rho^B, \rho^R, \mu^B, \mu^R, \nu^B, \nu^R], \\ \dot{\rho}^R &= L^R[\rho^B, \rho^R, \mu^B, \mu^R, \nu^B, \nu^R]. \end{aligned} \quad (19.2)$$

To construct the general form of the system equations, the strength concentrations and velocities are assembled as vectors as follows:

$$\begin{aligned}
\rho^B &= [\rho_{00}^B, \rho_{10}^B, \dots, \rho_{M-1,0}^B, \rho_{01}^B, \dots, \rho_{M-1,N-1}^B]', \\
\rho^R &= [\rho_{00}^R, \rho_{10}^R, \dots, \rho_{M-1,0}^R, \rho_{01}^R, \dots, \rho_{M-1,N-1}^R]', \\
\mu^B &= [\mu_{00}^B, \mu_{10}^B, \dots, \mu_{M-2,0}^B, \mu_{01}^B, \dots, \mu_{M-2,N-1}^B]', \\
\mu^R &= [\mu_{00}^R, \mu_{10}^R, \dots, \mu_{M-2,0}^R, \mu_{01}^R, \dots, \mu_{M-2,N-1}^R]', \\
\nu^B &= [\nu_{00}^B, \nu_{10}^B, \dots, \nu_{M-1,0}^B, \nu_{01}^B, \dots, \nu_{M-1,N-2}^B]', \\
\nu^R &= [\nu_{00}^R, \nu_{10}^R, \dots, \nu_{M-1,0}^R, \nu_{01}^R, \dots, \nu_{M-1,N-2}^R]'.
\end{aligned} \tag{19.3}$$

Notice that the size of the ρ^B, ρ^R vectors is $K = MN$, while the size of the μ^B, μ^R vectors is $L_1 = (M-1)N$ and the size of the ν^B, ν^R vectors is $L_2 = M(N-1)$. The reason for the different vector sizes is that the normal component of the velocities are known to be always zero at the boundaries of the game area, so we don't have to solve for these terms. Thus, for example, the mapping from the cell position index (m,n) to the vector position index k for (ρ^B, ρ^R) is given by $k = M \times n + m$, where k is an integer number between 0 and $M \times N - 1$.

In general, the system of equations (19.2) represents a complicated flow field where the velocities can be arbitrarily chosen by the controller. For example, one cell may act at one moment as a finite source of units and then turn into a sink at the next moment. In our experiments we found good results using a fourth-order Runge-Kutta method with variable step size (Matlab ode solver rk45).

19.5 Differential Game

Given initial strength distributions for the blue and red forces, we wish to generate strategies for the two forces so that, at the end of the game, each force has achieved an optimum balance with respect to its different goals. In other words, we seek a Nash equilibrium solution to a zero-sum differential game defined as follows:

$$J^* = \min_{\mu^B, \nu^B} \max_{\mu^R, \nu^R} J(\mu^B, \mu^R, \nu^B, \nu^R), \tag{19.4}$$

where the blue force tries to minimize the payoff function J while the red force tries to maximize the same payoff function. The Nash equilibrium value J^* of problem (19.4), if it exists, is called the value of the game.

To solve (19.4) we assume that the \mathbb{R}^{2K} -valued variables, $\rho = (\rho^B, \rho^R)$, are continuous functions of time defined on the interval $[t_0, t_f]$. We also assume that the velocities, $\mu = (\mu^B, \mu^R) \in \mathbb{R}^{2L_1}$ and $\nu = (\nu^B, \nu^R) \in \mathbb{R}^{2L_2}$, are continuous functions defined on the same interval.

19.6 Solution of the Differential Game

The numerical method used to seek a local Nash equilibrium solution of the differential game defined by equation (19.4), with a given payoff function subject to the dynamic system defined by (19.2), is the *Sequential Linear-Quadratic* (SLQ) Algorithm [4].

The SLQ algorithm tries to find a local Nash equilibrium solution by an iterative process in which the system differential equation is linearized around the i -th solution state estimate, $\rho_i(\mu_i, \nu_i)$, so that the i -th step consists of solving the following game subproblem:

$$\begin{aligned}
&\min_{\delta\mu^B, \delta\nu^B} \max_{\delta\mu^R, \delta\nu^R} \{ \bar{J}(\rho_i + \delta\rho; \mu_i + \delta\mu, \nu_i + \delta\nu) \} \\
&\delta\dot{\rho}(t) = L_\rho[\rho_i(t), \mu_i(t), \nu_i(t)]\delta\rho(t) \\
&\quad + L_\mu[\rho_i(t), \mu_i(t), \nu_i(t)]\delta\mu(t) \\
&\quad + L_\nu[\rho_i(t), \mu_i(t), \nu_i(t)]\delta\nu(t), t \in [t_0, t_f]; \delta\rho(t_0) = 0
\end{aligned} \tag{19.5}$$

where $\bar{J}(\rho; \mu, \nu) \equiv J(\mu^B, \mu^R, \nu^B, \nu^R)$ is assumed to be a quadratic function. We use $\delta\mu$ and $\delta\nu$ to represent small perturbations, $\mu - \mu_i$ and $\nu - \nu_i$, of the respective controls, μ and ν , from the current estimate, μ_i and ν_i . And $\delta\rho$ represents the corresponding perturbation, $\rho[\mu_i + \delta\mu, \nu_i + \delta\nu] - \rho[\mu_i, \nu_i]$, in the state trajectory.

An approximate solution to the state trajectory perturbation $\delta\rho$ subject to the perturbed inputs, $\delta\mu$ and $\delta\nu$, can be obtained by linearizing the system differential equation around (ρ_i, μ_i, ν_i) :

$$\begin{aligned} \delta\dot{\rho} = & L_\rho[\rho_i^B, \rho_i^R, \mu_i^B, \mu_i^R, \nu_i^B, \nu_i^R] \delta\rho \\ & + L_\mu[\rho_i^B, \rho_i^R, \mu_i^B, \mu_i^R, \nu_i^B, \nu_i^R] \delta\mu \\ & + L_\nu[\rho_i^B, \rho_i^R, \mu_i^B, \mu_i^R, \nu_i^B, \nu_i^R] \delta\nu \end{aligned} \quad (19.6)$$

The current solution estimate (μ_i, ν_i) can then be updated by

$$\begin{aligned} \mu_{i+1} &= \mu_i + s_i \delta\mu_i \\ \nu_{i+1} &= \nu_i + s_i \delta\nu_i \end{aligned} \quad (19.7)$$

with the step size $s_i > 0$, where $(\delta\mu_i, \delta\nu_i)$ denotes the Nash solution to the subgame (19.5).

The original nonlinear system (19.2) is solved next using the inputs (μ_{i+1}, ν_{i+1}) to obtain the new trajectory $\rho_{i+1} = \rho[\mu_{i+1}, \nu_{i+1}]$. Successive iterations proceed in the same way as before, until the norm of the update velocities $(\delta\mu_i, \delta\nu_i)$ reduces to an acceptable level. Recall that when the payoff function is quadratic, the computation of the Nash velocities, $(\delta\mu_i, \delta\nu_i)$, for the above subproblem involves the solution of a system of Riccati differential equations.

19.7 Experimental Results

The following two experiments illustrate the utility of the Game Flow program.

19.7.1 Experiment 1

The game area is a square of unit length in each side, and is divided into 64 squares to form an 8×8 grid.

The game board may represent some geographical area where the conflict takes place. It should be expected then, that certain local features of the game area will have an effect on the evolution of the game. For example, the energy that the forces must spend to move their respective assets should vary as they attempt to go across different types of terrain: dessert dunes, marshy land, dense forests, etc. Similarly, different features of the game area might affect the attrition rate sustained by the forces.

In this experiment, the game area consists of two types of terrain: one smooth area, through which movement of assets is relatively easy, surrounded by more difficult terrain. The type of terrain is reflected on the running cost on velocity as shown in Figure 19.2, where the smooth area is indicated in dark color. Notice that, for example, to go from cell (4,4) to cell (5,5), the path that goes through cell (5,4) is less expensive than the path that goes through cell (4,5).

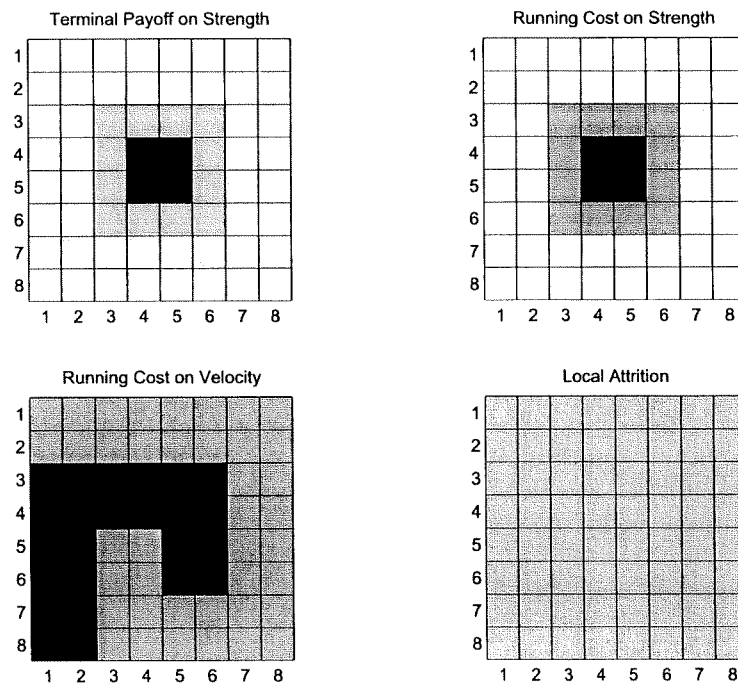


Figure 19.2: Game Board (clockwise starting from top left figure: terminal payoff associated with final values of the strength concentration in the cells; running cost associated with instantaneous values of the strength concentration in the cells; local attrition; and running cost on velocity). Darker shade indicates higher value.

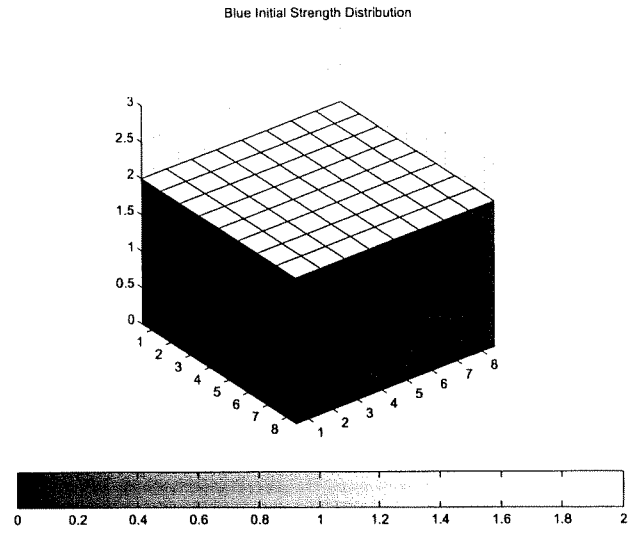


Figure 19.3: Initial strength distribution of the blue force.

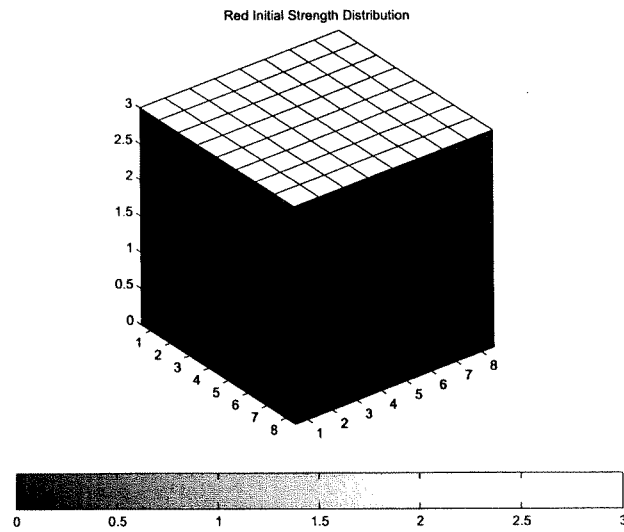


Figure 19.4: Initial strength distribution of the red force.

To simplify visualization of the experiment results, attrition not caused by the enemy is not included in this experiment (note that in Figure 19.2 the local attrition distribution is uniformly zero).

The blue force has an initial strength of 2 units spread uniformly on the game area, as shown in Figure 19.3. The red force has an initial strength of 3 units, also spread uniformly on the game area, as shown in Figure 19.4.

We assume that each force has a symmetric attack efficiency function as depicted in Figure 19.5. The Figure shows that the red force is slightly more powerful than the blue force at close range. On the other hand, the blue force has a longer range.

The objective for the blue (resp. red) force consists of the following five goals: (a) reach the end of the game with as much strength as possible; (b) place as many units as possible in the four cells located in the middle of the game board; (c) remove the red (resp. blue) units from the four central cells, and block any attempts by the red (resp. blue) force to move its own units into that area; (d) continuously try to minimize the strength of the red (resp. blue) force; and (e) minimize the energy expenditure in accomplishing the first four goals of this mission statement.

While the respective mission statements for the blue and red forces may look identical, each force can assign different priorities (weights) to the different goals. For instance, in this experiment, the weight for the blue force associated with movement of units over the smooth terrain of the game board is equivalent to three fourths the corresponding weight for the red force. This means that the blue force has more freedom of movement over the game board than the red force.

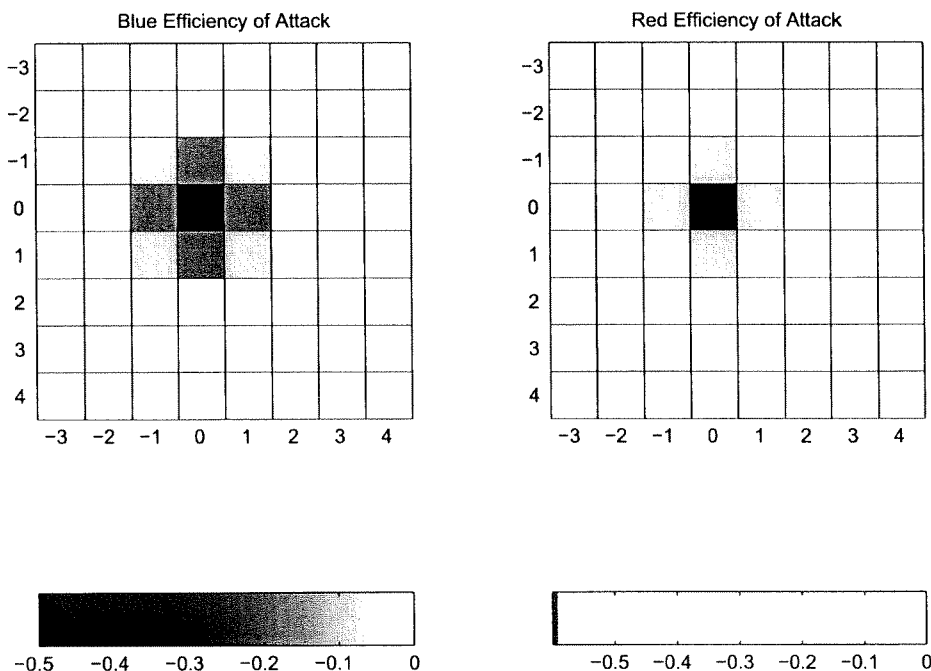


Figure 19.5: Efficiency of attack for the blue and red forces.

The overall objective of the game is defined by the following quadratic payoff function (also see Figure

19.2):

$$\begin{aligned}
J(\mu^B, \mu^R, \nu^B, \nu^R) = & \\
& \int_{t_0}^{t_f} \left\{ \frac{1}{2} \rho^B(t)' [Q^{BB} - Q^{BR}] \rho^B(t) \right. \\
& \quad \left. + \frac{1}{2} \rho^R(t)' [Q^{RB} - Q^{RR}] \rho^R(t) \right. \\
& \quad + \frac{1}{2} \mu^B(t)' [R_1^B] \mu^B(t) + \frac{1}{2} \nu^B(t)' [R_2^B] \nu^B(t) \\
& \quad \left. + \frac{1}{2} \mu^R(t)' [R_1^R] \mu^R(t) + \frac{1}{2} \nu^R(t)' [R_2^R] \nu^R(t) \right\} dt \\
& + \frac{1}{2} \rho^B(t_f)' [Q_f^B] \rho^B(t_f) + \frac{1}{2} \rho^R(t_f)' [Q_f^R] \rho^R(t_f),
\end{aligned} \tag{19.8}$$

where all the weighting matrices are diagonal. The elements of $Q^{BB} \in \mathbb{R}^{K \times K}$, $Q^{BR} \in \mathbb{R}^{K \times K}$, $Q^{RB} \in \mathbb{R}^{K \times K}$ and $Q^{RR} \in \mathbb{R}^{K \times K}$ are chosen in accordance with the goals (b)-(d) of each force as defined above; the elements of $R_1^B \in \mathbb{R}^{L_1 \times L_1}$, $R_1^R \in \mathbb{R}^{L_1 \times L_1}$, $R_2^B \in \mathbb{R}^{L_2 \times L_2}$ and $R_2^R \in \mathbb{R}^{L_2 \times L_2}$ reflect goal (e); and the elements of $Q_f^B \in \mathbb{R}^{K \times K}$ and $Q_f^R \in \mathbb{R}^{K \times K}$ correspond to the terminal cost associated with goal (a).

The solution technique (SLQ method)[4] is iterative and it improves the current solution estimate at each iteration. Hence, to solve the game, an initial guess has to be made for the strategy used by the forces. In this experiment, the initial choice of strategy affected the rate of convergence in the iterative solution of the game, but it did not have significant effects on the final outcome of the game. So, we have arbitrarily assigned a constant velocity distribution such that the units of the blue and red forces converge in the middle of the game board.

The value of the game corresponding to the initial solution estimate is shown in Table 19.1, broken into the individual cost components.

Table 19.1: Payoff function value for the initial solution estimate

Force	J_μ	J_ν	J_ρ	$J_{f\rho}$	J
Blue	17600	18800	-353.4	-1273.6	35534
Red	-20800	-21400	647.5	1287.8	-40264
Total					-730

The SLQ algorithm was used next to find a Nash equilibrium solution for the game. In this experiment, the solver was stopped after 70 iterations, when the error (i.e., the norm $\|(\delta\mu_i, \delta\nu_i)\|$ of the velocity updates) was approximately 0.8 percent of the original error. At this error level, further iterations had no significant effects on the solution.

The value of the game corresponding to the Nash equilibrium solution is shown in Table 19.2, broken into the individual cost components, corresponding to the running costs on velocity and strength, and terminal cost on strength.

Table 19.2: Payoff function value for the Nash equilibrium solution

Force	J_μ	J_ν	J_ρ	$J_{f\rho}$	J
Blue	27.6	23.6	-259.3	-584.1	-792.2
Red	-40.4	-38.3	490.9	1110.0	1522.2
Total					730.0

Clearly, the Nash equilibrium solution found by the SLQ algorithm, greatly improves the performance of the two forces in terms of the value of the payoff function selected for this experiment. Recall that the blue force tries to minimize the total payoff while the red force tries to maximize it.

Figures 19.6 and 19.7 show the initial Nash strength distributions for the blue and red forces. The distributions are identical and uniform, so the shade (color) is uniform over the area. The direction of unit movement across the border between each pair of cells is indicated by an arrow. The size of each arrow indicates the magnitude of an initial velocity component.

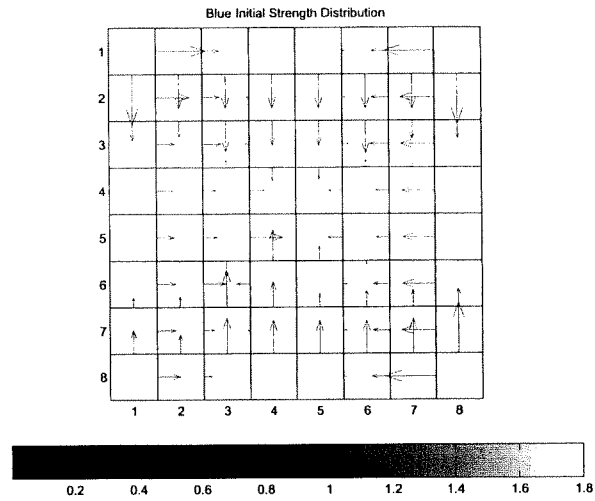


Figure 19.6: Initial Nash Strength Distribution for Blue force. Arrows indicate magnitudes of the velocity components across the boundaries.

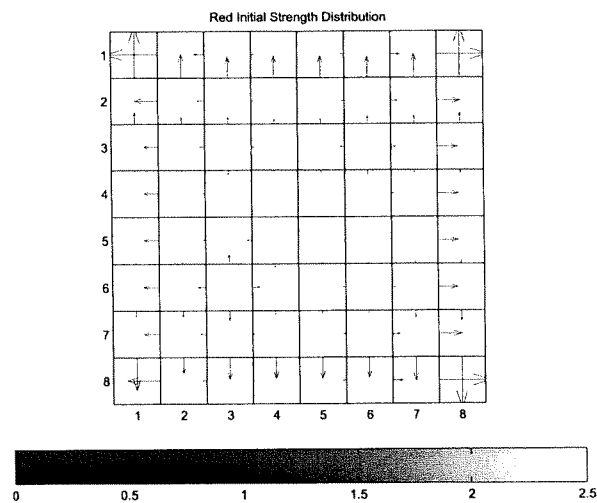


Figure 19.7: Initial Nash Strength Distribution for Red force. Arrows indicate magnitudes of the velocity components across the boundaries.

Figures 19.8 and 19.9 show the final Nash strength distributions for the blue and red forces. The arrows indicate the magnitudes of the velocity components as the respective units of the two forces reached their final destinations.

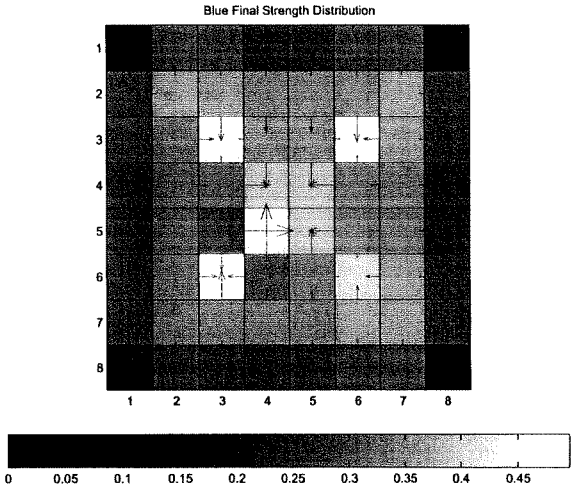


Figure 19.8: Final Nash Strength Distribution for Blue force. Arrows indicate magnitudes of the velocity components across the boundaries.

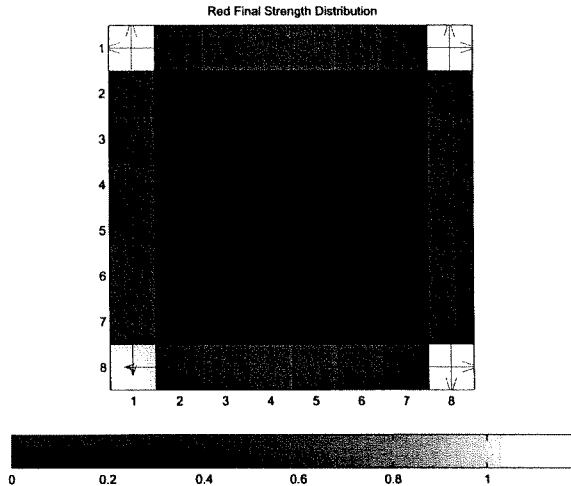


Figure 19.9: Final Nash Strength Distribution for Red force. Arrows indicate magnitudes of the velocity components across the boundaries.

The final total strength for the blue force was 0.32, while the final total strength for the red force was 0.46. Therefore, the red force conserved only 15.4% of its original strength, while the blue force conserved 16.1% of its own strength.

Qualitatively speaking, we can say that, in this scenario, the superiority of the blue force in the attack range prevailed, allowing the blue units to keep the red units out of the most valuable cells in the center of the board. This can be better appreciated in Figures 19.10 and 19.11. Indeed, at the end of the game, the red units were forced to retreat into the four corners of the game board where less valuable cells were to be found. It is also clear that different transportation costs assigned to different regions affected the way in which the blue and red forces adjusted their strength concentrations during the game. This can be seen in the fact that the red's final concentration in the (8,1) corner is weaker than the other three corners because the terrain near (8,1) is harder to traverse.

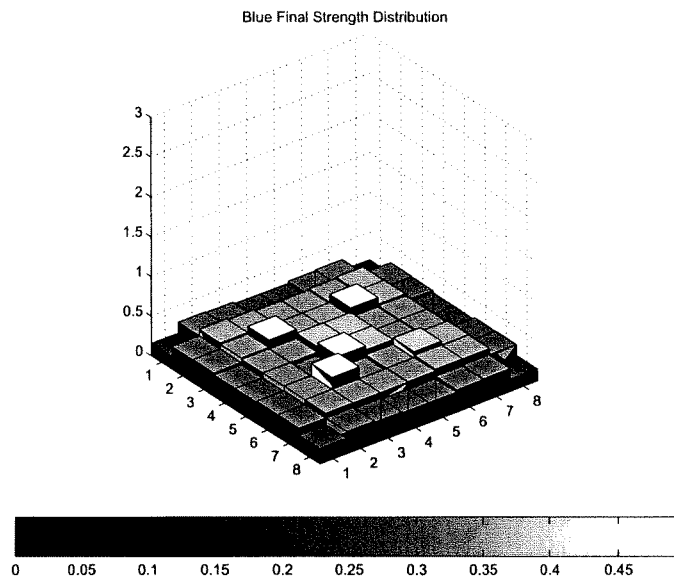


Figure 19.10: Final Nash Strength Distribution for Blue force.

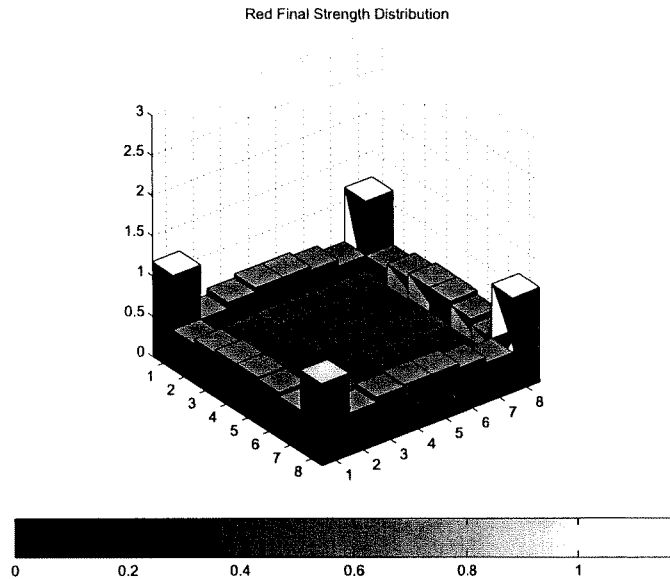


Figure 19.11: Final Nash Strength Distribution for Red force.

19.7.2 Experiment 2

The game board is a square of unit length in each side, and is divided into 36 square cells, which form an 6×6 grid.

We assume that the two forces assign equal value to each cell, so that a common map of the game board showing the real estate value is shown for the two forces in Figure 19.12. With respect to the running cost on velocity, Figure 19.12 shows that the light cells form a path of smooth terrain, through which movement of units is relatively easy, hence units spend relatively low energy when moving through light cells; the dark cells represent more difficult terrain. Figure 19.12 also shows that there is local attrition associated with the terrain. Units that move into high attrition cells will suffer loss of strength even in the absence of enemy attack.

The blue force has an initial strength of $1/6$ units spread uniformly on the six bottom right cells of the game board, as shown in Figure 19.13. The red force has an initial strength of $1/6$ units spread uniformly on the six top left cells of the game board as shown in Figure 19.14.

We assume that each force has a symmetric attack efficiency function as depicted in Figure 19.15. The figure shows that the blue force is more powerful than the red force at close range. On the other hand, the red force has a longer range, covering an area of approximately 3×3 cells.

The objective for the blue force consists of the following three goals: (a) reach the end of the game with as much strength as possible; (b) place as many units as possible in the most valuable cells located in the top right corner of the game board; (c) minimize the energy expenditure in accomplishing the first two goals of this mission statement.

The objective for the red force consists of the following three goals: (a) reach the end of the game with as much strength as possible; (b) block the blue forces as they try to move towards the most valuable cells located in the top right corner of the game board; and (c) minimize the energy expenditure in accomplishing the first two goals of this mission statement.

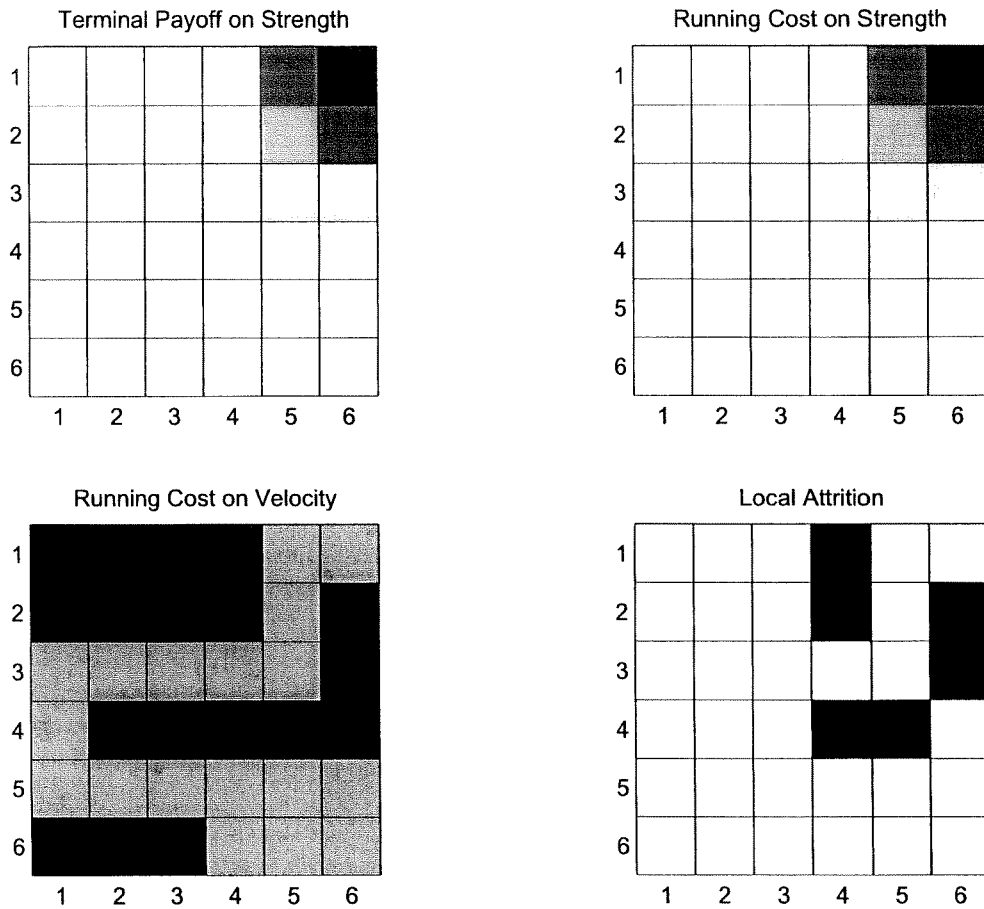


Figure 19.12: Game Board (clockwise starting from top left figure: terminal payoff associated with final values of the strength concentration in the cells; running cost associated with instant values of the strength concentration in the cells; local attrition; and running cost on velocity). Darker shade indicates higher value.

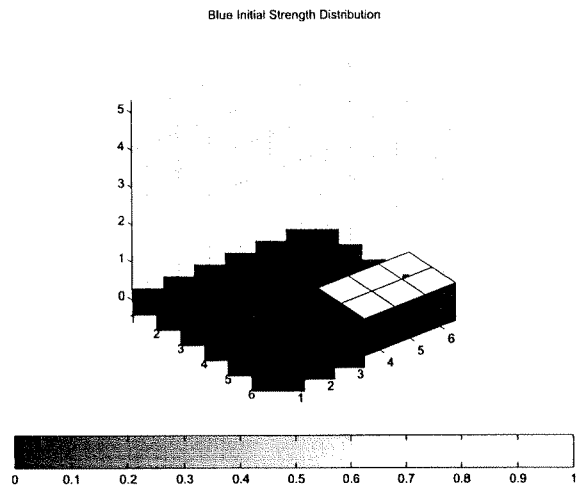


Figure 19.13: Initial strength distribution of the blue force.

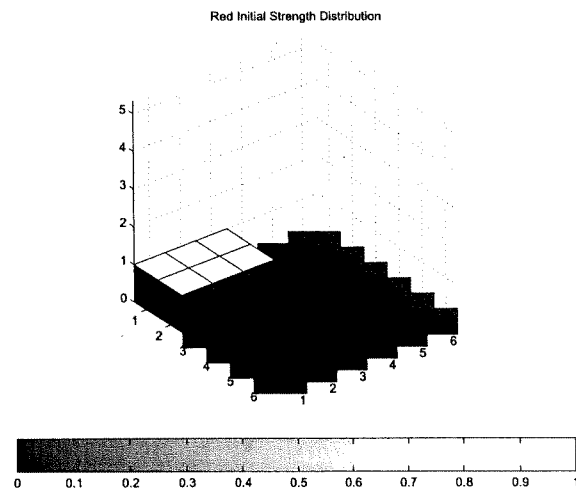


Figure 19.14: Initial strength distribution of the red force.

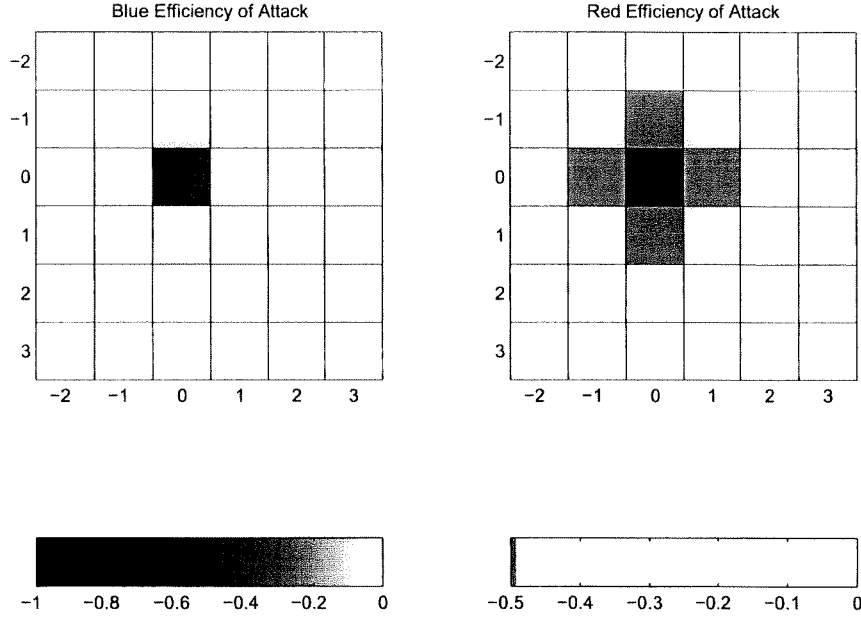


Figure 19.15: Efficiency of attack for the blue and red forces.

The overall objective of the game is defined by the following quadratic payoff function:

$$\begin{aligned}
J(\mu^B, \mu^R, \nu^B, \nu^R) = & \int_{t_0}^{t_f} \left\{ \frac{1}{2} \rho^B(t)' [Q^{BB} - Q^{BR}] \rho^B(t) \right. \\
& + \frac{1}{2} \rho^R(t)' [Q^{RB} - Q^{RR}] \rho^R(t) \\
& + \frac{1}{2} \mu^B(t)' [R_1^B] \mu^B(t) + \frac{1}{2} \nu^B(t)' [R_2^B] \nu^B(t) \\
& + \frac{1}{2} \mu^R(t)' [R_1^R] \mu^R(t) + \frac{1}{2} \nu^R(t)' [R_2^R] \nu^R(t) \Big\} dt \\
& + \frac{1}{2} \rho^B(t_f)' [Q_f^B] \rho^B(t_f) + \frac{1}{2} \rho^R(t_f)' [Q_f^R] \rho^R(t_f),
\end{aligned} \tag{19.9}$$

where all the weighting matrices are diagonal. The elements of $Q^{BB} \in \mathbb{R}^{K \times K}$, $Q^{BR} \in \mathbb{R}^{K \times K}$, $Q^{RB} \in \mathbb{R}^{K \times K}$ and $Q^{RR} \in \mathbb{R}^{K \times K}$ are chosen in accordance with the goals (b)-(d) of each force as defined above; the elements of $R_1^B \in \mathbb{R}^{L_1 \times L_1}$, $R_1^R \in \mathbb{R}^{L_1 \times L_1}$, $R_2^B \in \mathbb{R}^{L_2 \times L_2}$ and $R_2^R \in \mathbb{R}^{L_2 \times L_2}$ reflect goal (e); and the elements of $Q_f^B \in \mathbb{R}^{K \times K}$ and $Q_f^R \in \mathbb{R}^{K \times K}$ correspond to the terminal cost associated with goal (a).

The solution technique (SLQ method)[4] is iterative and it improves the current solution estimate at each iteration. Hence, to solve the game, an initial guess has to be made for the strategy used by the forces. The initial strategy for blue consisted in following the path of smooth terrain at constant velocity throughout the duration of the game. The initial strategy for red consisted on marching one row of cells forward in order to block the smooth path using all its strength. In this particular experiment, the initial choice of strategy was critical in attaining convergence in the iterative solution of the game.

The value of the game corresponding to the initial solution estimate is shown in Table 19.3, broken into the individual cost components.

The SLQ algorithm was used next to find a Nash equilibrium solution for the game. In this experiment, the solver was stopped after 530 iterations. Figure 19.16 shows the evolution of the error (i.e., the norm $\|(\delta\mu_i, \delta\nu_i)\|$ of the velocity updates) as the algorithm converges. Note that the step size had to be reduced from 0.2 to 0.05 after 20 iterations in order to achieve convergence.

Table 19.3: Payoff function value for the initial solution estimate

Force	J_μ	J_L	J_ρ	$J_{f\rho}$	J
Blue	3146080	6350400	-59796	-763535	8673149
Red	-96000	0	16.5	9.8	-95974
Total					8577176

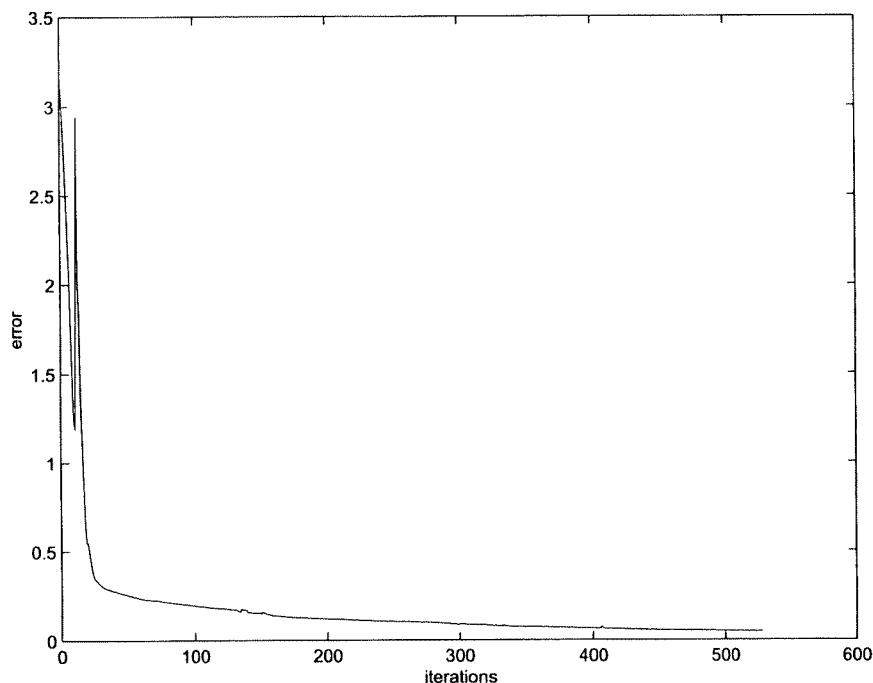


Figure 19.16: Convergence of the SLQ algorithm.

The value of the game corresponding to the Nash equilibrium solution is shown in Table 19.4, broken into the individual cost components, corresponding to the running costs on velocity and strength, and terminal cost on strength.

Clearly, the Nash equilibrium solution found by the SLQ algorithm, greatly improves the performance of the blue force in terms of the value of the payoff function selected. That the Nash equilibrium solution also improves the performance of the red force is not so clear. However, when comparing with the initial solution estimate, one should realize that the reduction observed in the terminal cost associated with the strength of the blue force is entirely due to the more effective deployment of the red force corresponding to the Nash equilibrium found by the SLQ algorithm.

Figures 19.17 and 19.18 show the initial Nash strength distributions for the blue and red forces. The direction of unit movement across the border between each pair of cells is indicated by an arrow. The size of each arrow indicates the magnitude of the initial velocity component.

Figures 19.19 and 19.20 show the final Nash strength distributions for the blue and red forces. The arrows indicate the magnitudes of the velocity components as the respective units of the two forces reached their final destinations.

To aid visualization of the results, Figures 19.21 and 19.22 show a three dimensional view of the final Nash strength distributions for the blue and red forces.

Another view of the results is shown in Figure 19.23. This shows how the respective strength distri-

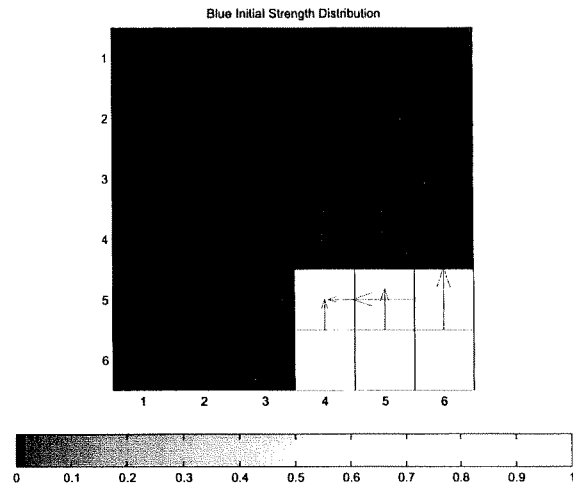


Figure 19.17: Initial Nash Strength Distribution for Blue force. Arrows indicate magnitudes of the velocity components across the boundaries.

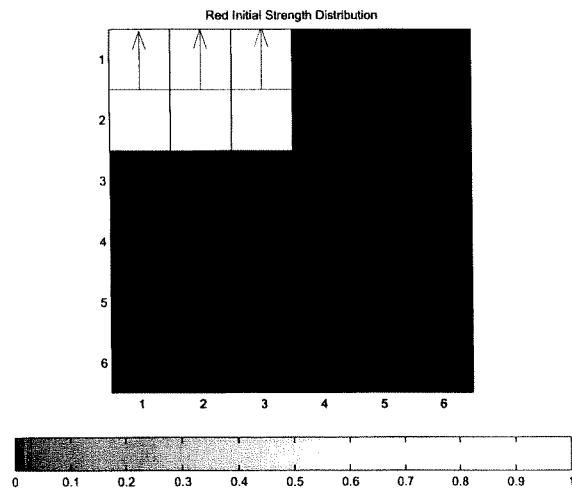


Figure 19.18: Initial Strength Distribution for Red force. Arrows indicate magnitudes of the velocity components across the boundaries.

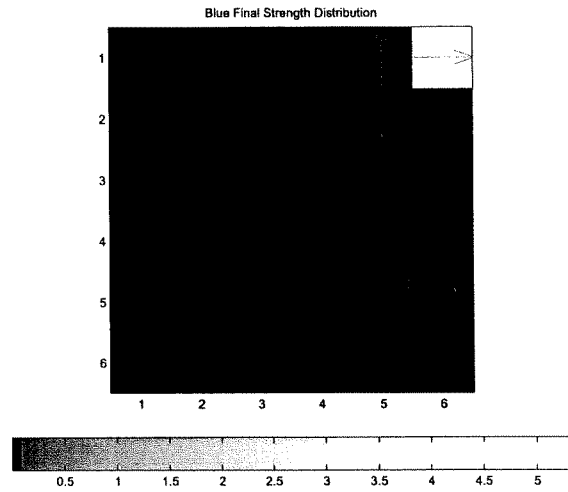


Figure 19.19: Final Nash Strength Distribution for Blue force. Arrows indicate magnitudes of the velocity components across the boundaries.

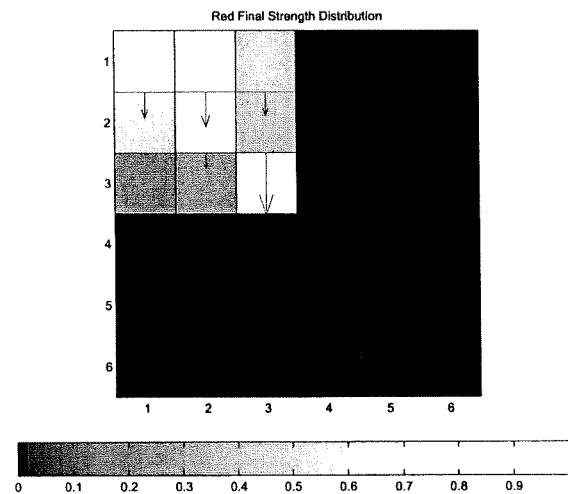


Figure 19.20: Final Nash Strength Distribution for Red force. Arrows indicate magnitudes of the velocity components across the boundaries.

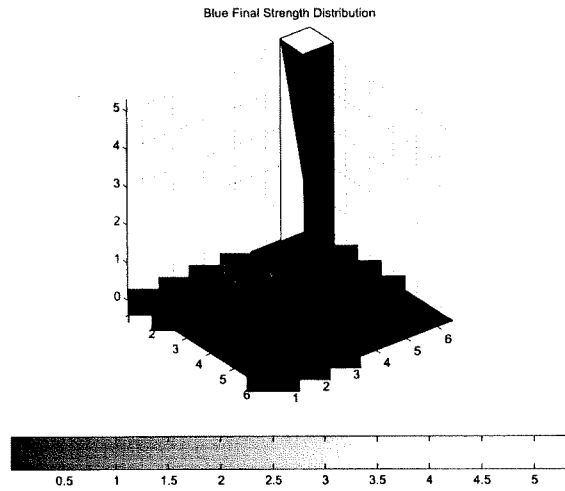


Figure 19.21: Final Nash Strength Distribution for Blue force.

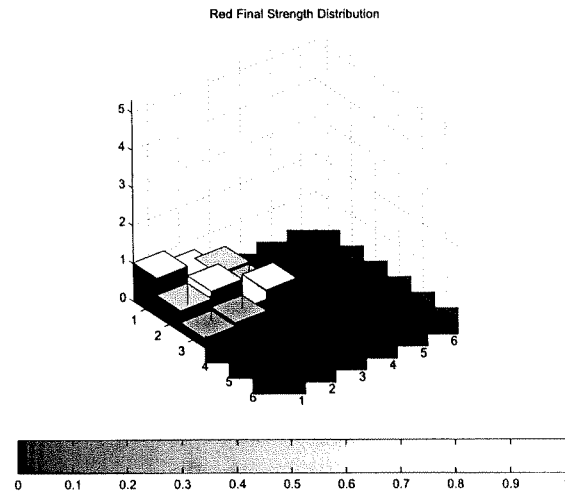


Figure 19.22: Final Nash Strength Distribution for Red force.

Table 19.4: Payoff function value for the Nash equilibrium solution

Force	Running J_μ	Running J_ν	Running J_ρ	Terminal $J_{f\rho}$	Game J
Blue	101086	185527	-13027	-702722	-429135
Red	-103863	0	7.46	2.45	-103853
Total					-532988

butions change over time. The figure shows snapshots of the strength distributions for the blue and red forces at the beginning and the end of the game, and at two intermediate points.

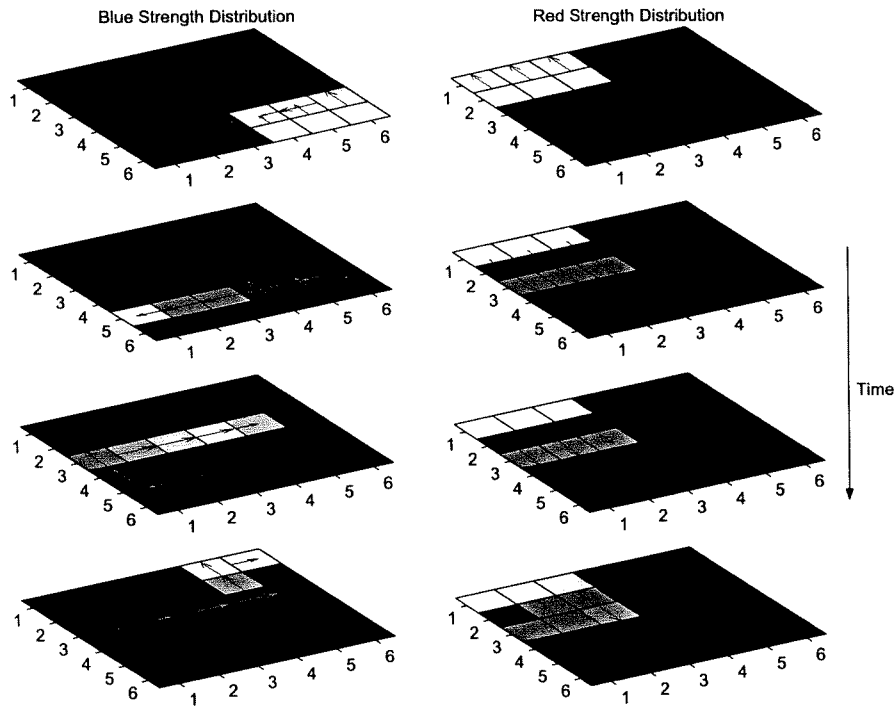


Figure 19.23: Progression of the instantaneous Nash Strength Distributions for Blue and Red forces. Arrows indicate magnitudes of the velocity components across the boundaries.

The results of this experiment are in good agreement with what one might expect given the game scenario. Consider, for example, the strategy selected by the red force. Indeed, if we recall that the red units have an attack range of three cells while the blue units have an attack range of only one cell, one can appreciate the strategy selected by the red force, namely to attack blue from a safe distance with the bulk of its strength, while spending its remaining force to block the passage of the blue force through the smooth pathway.

19.8 Conclusions

It was difficult to find a scenario which was both interesting, in the sense that some significant amount of action occurred during the game, and in which a game theoretic solution could be found by the SLQ algorithm. The main issue was how to select weights appropriately, given a particular scenario. The

choice of the initial guess for the controls was also important in some cases. When the two forces had an initial strength distribution spread uniformly across the whole game area, it was easier to find a solution as compared with the case when the initial strength distributions were concentrated in small regions of the game area. However, it is still not clear at this time what features are most critical in determining whether a given scenario has an SLQ solution or not.

With respect to computational complexity, the Game Flow solution engine executes relatively fast, using a combination of Matlab built-in functions (e.g., ODE solvers) and custom made C++ routines. For example, the CPU time required to run a single iteration in one of the experiments (8×8 grid) was less than ten seconds. Still, it must be mentioned that in some experiments it took hundreds of iterations to attain convergence of the SLQ algorithm. The experimental results in this report were obtained running the Game Flow program on a 800 MHz PC with 500 MB RAM.

In the current version of the Game Flow model, physical features of the theater itself can affect the dynamics in two ways: i) in the rate of attrition of the human and/or mechanical assets in the field; ii) in the energy cost associated with the movements of assets in the field. The last feature is not implemented directly in the the differential equations of the system. Instead, it is represented in the payoff functions of the two forces. A velocity reduction parameter, similar to an attrition parameter could be implemented in future versions of the Game Flow model.

Another characteristic of the current version of the Game Flow model is that physical features of the theater have no effect on the efficiency of attack. Hence, the enemy cannot hide behind a mountain range, for instance. Also the efficiency of attack functions can have a directionality associated with them before the beginning of a game, but these cannot be rotated or reoriented as the game evolves. This could be an important addition to enhance the strategic capabilities of the Game Flow model.

It is hoped that by adding complexity to the model, the class of interesting problems that can be solved with the Game Flow program would be enlarged.

Bibliography

- [1] M.S. Atkin, D.L. Westbrook, P.R. Cohen, *Capture the Flag: Military Simulation Meets Computer Games*, AAAI Spring Symposium on AI and Computer Games, 1999.
- [2] B. Stilman, V. Yakhnis, *Adapting the Linguistic Geometry-Abstract Board Games Approach to the Air Operations*, 2nd AEC Symposium, 219-234, July 2000.
- [3] W.L. Wood, *Introduction to Numerical Methods for Water Resources*, Clarendon Press, 1993.
- [4] H. Mukai, A. Tanikawa, et al., *Game-Theoretic Linear-Quadratic Method for Air Mission Control*, Proceedings of the IEEE Conference on Decision and Control, 2574-2580, December 2000.