



AFRL-RI-RS-TR-2024-024

SYSTEMIC GENERATIVE ENGINEERING

SIEMENS CORPORATION, CORPORATE TECHNOLOGY

FEBRUARY 2024

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (<http://www.dtic.mil>).

AFRL-RI-RS-TR-2024-024 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /

STEVEN L. DRAGER
Work Unit Manager

/ S /

NICK P. KOWALCHUK
RIT Chief Engineer
Computing & Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

1. REPORT DATE		2. REPORT TYPE		3. DATES COVERED	
FEBRUARY 2024		FINAL TECHNICAL REPORT		START DATE SEPTEMBER 2020 END DATE SEPTEMBER 2023	
4. TITLE AND SUBTITLE SYSTEMIC GENERATIVE ENGINEERING					
5a. CONTRACT NUMBER FA8750-20-C-0542		5b. GRANT NUMBER N/A		5c. PROGRAM ELEMENT NUMBER 62303E	
5d. PROJECT NUMBER		5e. TASK NUMBER		5f. WORK UNIT NUMBER R31V	
6. AUTHOR(S) Dr. Arun Ramamurthy					
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) PRIME: Siemens Corporation, Technology: 755 College Road East, Princeton NJ 08540 SUB: University of Pennsylvania: 200 S. 33rd Street University of Pennsylvania, Philadelphia PA 19104 SUB: Massachusetts Institute of Technology: 32 Vassar Street, Cambridge MA 2139					8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RITA DARPA 525 Brooks Road 675 North Randolph St Rome NY 13441-450 Arlington VA 22203-2114			10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/RI & DARPA		11. SPONSOR/MONITOR'S REPORT NUMBER(S) AFRL-RI-RS-TR-2024-024
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for Public Release; Distribution Unlimited. PA#: AFRL-2024-0138 / AFRL-2024-0739 Date Cleared: 27 February 2024					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT This report details research performed under this Defense Advanced Research Projects Agency Symbiotic Design for Cyber-Physical Systems effort titled "Systemic Generative Engineering." The effort focused on technical area 1, aiming to develop algorithms and frameworks enabling the creation of Artificial Intelligence (AI) co-designers. In particular, this effort sought to create a modular, open-source AI toolchain to: (i) construct, (ii) compose, and (iii) explore vast design spaces that arise in cyber-physical systems (CPS). The design space construction AI ingests previous designs and heterogeneous data, using machine learning techniques and knowledge graphs to capture explicit and implicit knowledge (e.g., mappings of requirements to functions, functions to components, components to configurations). The design space construction process accelerates the acquisition and curation of knowledge from existing design corpora; a process that today is mostly manual. The design composition AI generatively synthesizes design alternatives (conventional and novel) and assists the engineers with the automatic generation of evaluation pipelines, allowing engineers to spend their time creating, rather than tweaking tools and models. Finally, the design space exploration AI enables the inverse design of complex CPS driven by a set of target performance metrics and constraints, enabling a capability that is very difficult for experts: finding unintuitive and novel design alternatives that exhibit desired performance and behavior characteristics.					
15. SUBJECT TERMS AI co-design, symbiotic design, cyber physical systems, design space construction, design composition, design space exploration					
16. SECURITY CLASSIFICATION OF:				17. LIMITATION OF ABSTRACT	
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U	SAR		18. NUMBER OF PAGES 183
19a. NAME OF RESPONSIBLE PERSON STEVEN L. DRAGER					19b. PHONE NUMBER (Include area code) N/A

TABLE OF CONTENTS

List of Figures	iv
List of Tables	ix
1.0 SUMMARY	1
1.1 Executive Summary	1
1.2 Impact	2
1.3 Innovations	2
1.4 Project Summary	4
1.5 Document Organization	4
2.0 INTRODUCTION	5
2.1 Evolution of Designs	5
2.2 Complexity in Design of Cyber-Physical Systems	6
2.3 Systemic Generative Engineering	9
2.4 Framework Details	10
3.0 METHODS, ASSUMPTIONS, AND PROCEDURES	14
3.1 Open World Design	14
3.1.1 Background	14
3.1.2 Concept Details	14
3.1.3 Implementation	15
3.1.4 Retrieval System Design	17
3.1.5 Dataset Information	18
3.1.6 Results	18
3.2 Constraint Synthesis	18
3.2.1 Use-case Description	18
3.2.2 Concept Details	20
3.2.3 Results	24
3.3 Design Reuse through Functional Abstractions	25
3.3.1 Background	25
3.3.2 Implementation	26
3.3.3 Design Abstraction	29
3.3.4 Methodology	30
3.4 Design Representation as a Knowledge Graph	32
3.4.1 Background	32
3.4.2 Methodology	33
3.4.3 Results	34
3.5 Design Master Model	35
3.5.1 Background	35
3.5.2 Methodology	36
3.5.3 Results	39
3.6 Design Composition with Conditional Variational Autoencoders	40
3.6.1 Introduction	40

3.6.2	Concept Details	40
3.6.3	Results	42
3.7	Design Completion with Graph Generative Models	43
3.7.1	Background	43
3.7.2	Technical Details	44
3.7.3	Results	46
3.8	Field Prediction with Reduced Order Modeling	48
3.8.1	Background	48
3.8.2	Technical Details	49
3.8.3	Results	52
3.8.4	Summary	57
3.9	Inverse Dynamical Problems with Differentiable Solvers	57
3.9.1	Introduction	57
3.9.2	Methodology	58
3.9.3	Results	59
3.10	Modeling Cross-Domain Interactions	61
3.10.1	Evaluation 1: Dynamic Thermo-mechanical Problem	62
3.10.2	Evaluation 2: Bijective and non-bijective canonical problems	63
3.10.3	Summary	64
3.11	Hybrid Topology Optimization	64
3.11.1	Introduction	64
3.11.2	Methodology	65
3.11.3	Results	68
3.11.4	Observations	71
3.12	Reinforcement Learning for Controller Design	71
3.12.1	Background	71
3.12.2	Methodology	71
3.12.3	Summary	73
3.13	Bayesian Optimization for Spaces with Parametric Discrete Choices	73
3.13.1	Background	73
3.13.2	Methodology	74
3.13.3	Evaluation	75
3.14	Symmetry Constrained Bayesian Optimization	76
3.14.1	Introduction	76
3.14.2	Technical Details	78
3.15	Verification and Assurance of Cyber-Physical Systems	78
3.15.1	Background	78
3.15.2	Methodology	80
3.15.3	Results	82
3.16	Advanced Simulations	85
3.16.1	Computational Design of Soft Matter Computers	85
3.16.2	Real-time Hydrodynamic Simulations with Lattice Boltzmann Method	91
3.16.3	A Six Degree Of Freedom Underwater Vehicle Simulator	97
4.0	RESULTS AND DISCUSSION	106

4.1	Design of an Unmanned Aerial Vehicle	106
4.1.1	Problem Summary	106
	Design Methodology.....	108
4.1.2	Results.....	112
4.2	Design of Arctic Exploration Unmanned Underwater Vehicle	114
4.2.1	Problem Summary	114
4.2.2	Design Methodology.....	116
4.2.3	Results.....	119
4.3	Design of Urban Air Mobility Vehicle	120
4.3.1	Problem Summary	120
4.3.2	Design Methodology.....	121
4.3.3	Results.....	124
4.4	Phase-end Demo 2 – Design of Arctic Exploration Unmanned Underwater Vehicle .	125
4.4.1	Problem Summary	125
4.4.2	Design Methodology.....	126
4.4.3	Results.....	134
4.5	Design of an Unmanned Aerial Vehicle (Phase 2)	136
4.5.1	Problem Summary	136
4.5.2	Design Methodology.....	138
4.5.3	Results.....	139
4.6	Design of an Unmanned Underwater Vehicle (Phase 2).....	140
4.6.1	Problem Summary	140
4.6.2	Design Methodology.....	141
4.6.3	Results.....	145
4.7	Hardware-Software Co-Design of Deep Neural Network Accelerators	147
4.7.1	Introduction.....	147
4.7.2	Methodology Background	149
4.7.3	Design Process	151
4.7.4	Results.....	154
5.0	CONCLUSIONS.....	157
6.0	REFERENCES	159
	APPENDIX A – Publications and Presentations	166
	List of Symbols, Abbreviations, and Acronyms	167

LIST OF FIGURES

Figure 1: The evolution of design complexity in the past few decades.....	5
Figure 2: Challenges in the design of CPS [Adapted from 1]	7
Figure 3: Systemic Generative Engineering – How it works.	10
Figure 4: Modules of the Systemic Generative Engineering framework and their associated interactions.....	11
Figure 5: Open World Design Concept	15
Figure 6: An illustration of the hybrid-electric powertrain of the reference BMW i8 vehicle.....	19
Figure 7: The block definition diagram of the HEV powertrain	19
Figure 8: The ground truth design space constraints for the HEV powertrain	20
Figure 9: An illustrative example of a logic rule that can be used to build a constrained design space.....	21
Figure 10: The process for discovery of logic formulae using template search	23
Figure 11: Comparison of the generated results with the ground truth	24
Figure 12: Pair-wise comparison of generated constraints against the ground truth.....	24
Figure 13: A conceptual overview of design functional reuse system	27
Figure 14: A conceptual illustration of the graph encoder	27
Figure 15: An illustration of the conversion of a SMILE to an undirected graph.....	28
Figure 16: The concept of the AI modeling of function encoder.....	29
Figure 17: (A) Given an initial DSL of lower-level primitives and a corpus of programs written in the initial DSL, (B) Stitch uses a fast and memory efficient corpus-driven top-down synthesis algorithm to construct program abstractions which maximally compress shared structure across the corpus of program trees. (C) Stitch returns a learned library that augments the initial DSL with discovered function abstractions, and automatically rewrites the initial corpus using the learned library functions.	30
Figure 18: Corpus-guided top-down search. (A) Schematic of the top-down search procedure over partial abstractions (shown as function bodies). Partial abstractions can contain holes, or unfinished subtrees, denoted. Complete abstractions do not contain holes. (B) Partial and complete abstractions match at locations in the corpus. The blue partial abstraction can be expanded into the green complete abstraction, so the green abstraction matches in a subset of places that the blue abstraction does.	32
Figure 19: Multi-disciplinary considerations in the aircraft conceptual sizing and synthesis.....	33
Figure 20: An illustration of the different layers of the design hierarchy encoded in a graphical format.....	34
Figure 21: A representation of the design representation as a knowledge graph capturing the various pieces of information that goes into the conceptual sizing of the vehicle based on the Breguet range equation.	35
Figure 22: Engineering design stages from [13].....	36
Figure 23: Core Product Model 2.0 [14, 15].....	37
Figure 24 (a) B-Spline control point grid and (b) T-Spline control point mesh for surfaces	38
Figure 25: Examples of geometry and topology optimization using Isogeometric analysis with T-Splines.....	39

Figure 26: A high-level design completion workflow	40
Figure 27: Process involved in training the CNN-CAE model.....	41
Figure 28: A conceptual illustration of the CNN-VAE network architecture	42
Figure 29: Training loss of the CNN-VAE algorithm	43
Figure 30: A conceptual overview of the design graph completion recommendation framework.....	45
Figure 31: (A) Use of GINE for graph embedding; (B) Prediction of node type based on queued and head node embeddings; (C) Prediction of edge type based on the queued and head node embeddings	46
Figure 32: Plots of percentage of valid and successful graphs generated by the conditional generation framework on the molecule dataset	47
Figure 33: (A) Supplied initial partial molecule design; (B) Final completed molecule; (C) The completion sequence generated by the design graph recommendation algorithm	47
Figure 34. Add-on Workflow to Enable ROM for Design	49
Figure 35. Mapping Method: Mesh Morphing and Snapshot Projections.....	51
Figure 36. ROM for Design. Von Mises Stress MPa (above) and Error (below) on the Clipped Cylindrical PV at an Unseen/Test Design Point. Clipping Plane $y = 0$	53
Figure 37. ROM for Design Accuracy. Relative Von Mises Stress (Nominal Value: 890.653 MPa)	53
Figure 38. 3D Glider Geometry.....	54
Figure 39. ROM for Design. Pressure-Density Ratio (above) and Error (below) at 70% Wingspan at an Unseen/Test Design Point Geometry.	55
Figure 40. ROM for Design Accuracy. Relative Pressure-Density Ratio. 3D Glider.	56
Figure 41. 2D Unsteady Head Equation bc 1. Target Distribution $\mathbf{T} * (\mathbf{D}, \mathbf{t} = \mathbf{tf})$	59
Figure 42. 2D Unsteady Head Equation bc1. Calculated Distribution $\mathbf{T}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{t} = \mathbf{tf})$	60
Figure 43. 2D Unsteady Head Equation bc 2. Target Distribution $\mathbf{T}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{t} = \mathbf{tf})$	60
Figure 44. 2D Unsteady Head Equation bc2. Calculated Distribution $\mathbf{T}(\mathbf{x}_i, \mathbf{y}_i, \mathbf{t} = \mathbf{tf})$	61
Figure 45: Conceptual illustration of the goals of cross-domain modeling.....	62
Figure 46: Performance of the algorithm in the dynamic thermo-mechanical problem.....	62
Figure 47: Inverse inference from the thermal domain	63
Figure 48: Comparison of performance in bijective and non-bijective cases.....	64
Figure 49: Flow of the hybrid TO framework, which can handle, if needed, multiple physics solvers and design-dependent loading. The shared design grid on which the design update is performed allows any number and type of solvers to be integrated for concurrent optimization.	65
Figure 50: Example of the auxiliary field, ϕ_i , used to apply the design-dependent pressure loads for the problem shown in (a). External surfaces are outlines in red. The auxiliary field (d) undergoes the same changes as the design itself (c), but without internal voids.....	67
Figure 51: Illustration of the summation method to calculate design-dependent pressure loads.	68
Figure 52: 3D result (b) for stress-constrained mass minimization, where the external shape is from an imported fairing design (a).....	68
Figure 53: Benchmark results for a bridge design with fixed loads (a-b) and with the proposed design-dependent pressure loading (c-d), with its final auxiliary field (e).	69
Figure 54: Optimized pressure vessels with fixed loads on the fixed outer surface (a-b) and with the proposed design-dependent pressure loading (c-d), with its final auxiliary field (e).	69

Figure 55: Benchmark result of a design with minimized drag at Reynold's number = 10. Streamlines of the fluid flow velocity are shown in black.....	70
Figure 56: A UUV designed under stress-constrained drag minimization at Reynold's number = 10.....	70
Figure 57: Architecture verse configuration design space exploration.	71
Figure 58: Progress of the RL algorithm in the design space exploration.....	72
Figure 59: Maximization on canonical function- several methods.....	74
Figure 60: Minimization on large choice function	75
Figure 61: Minimization on large choice function. Zoom.....	76
Figure 62: Figure depicting different symmetries found in biology; bilateral (Drosophila), radial (actinomorphic flowers) and spherical (coccus bacteria) symmetry [63].....	76
Figure 63: Hexacopter Design. The design space includes the choice of rotors, motors, batteries, and other components. These choices are discrete. Continuous design variables include e.g. the length of the arms and the pitch of the rotors. To find an optimal design, discrete and continuous variables must be optimized simultaneously. This is a mixed-variable high-dimensional optimization problem.	77
Figure 64: Algorithm Bayesian optimization original and symmetry constrained.....	78
Figure 65: A Conceptual Overview of the Proposed Methodology	80
Figure 66: Domain Model of the KG Constructed using the Extracted Information	81
Figure 67: SysML representation of the Landing Gear Model considered in our Experiments...	83
Figure 68: Digital Twin implementation of the system modeling the behavior to external stimulus	83
Figure 69: Example of the constructed knowledge graph capturing and linking assurance information to system models.....	85
Figure 70: Overview of research efforts and relevant physical characteristics at different length scales [84]	86
Figure 71: Overall design system for soft matter computing. Performance requirements are taken into consideration to identify the appropriate polymer chemistry, production methods and conditions, component, and system design through machine learning and multi-scale simulations.	87
Figure 72: A step in the molecular simulation of Poly(1,6-hexamethylene adipate) to determine the ion conductivity.....	89
Figure 73: A schematic of the 18 MOSFET implementation of the full adder	90
Figure 74: Results of the transient analysis illustrating the output voltages measured across the three output pins of the full adder.	91
Figure 75: Illustration of a uniform lattice for LBM. For each lattice node, 9 velocity vectors are considered in D2Q9 model of LBM.	92
Figure 76: Illustration of the simulation domain and setup	95
Figure 77: List of solid shapes considered in this example.	96
Figure 78: Velocity magnitude and vorticity distribution of all the cases at t=45s obtained by LBM.	96
Figure 79: (a). A cylinder-shaped UUV. (b). A glider-type UUV.....	97
Figure 80: A system architecture of an autonomous UUV.....	98
Figure 81: Body reference coordinate system.	99

Figure 82: Various vehicle states during the simulation.....	104
Figure 83: Rudder and stern planes control profile.	104
Figure 84: Step input rudder and stern plane control outputs.....	105
Figure 85: UUV trajectory over time. The blue plane refers to the surface water level, and the dark blue curve is the path trajectory.	105
Figure 86: An illustration of the racetrack which serves as the inspiration for the simulation paths which are used to design the optimal UAV	107
Figure 87: Simulation pipeline used to evaluate the performance of the designs generated by the framework.....	108
Figure 88: A sequential design process optimizing the design topology and control parameters for the UAV	111
Figure 89: Overview of the mission requirement	115
Figure 90: An illustration of the mission profile for which the UUV system has to be designed	116
Figure 91: Free-body diagram guiding the governing equations of motion for the various phases of the mision	117
Figure 92: A conceptual illustration of the sizing environment for the UUV design.....	118
Figure 93: UUV conceptual sizing optimization process	119
Figure 94: Preliminary design optimization process for the sized UUV	119
Figure 95: UAM analysis simulation pipeline developed.....	121
Figure 96: Design process workflow employed to explore the design space for the UAM design challenge	124
Figure 97: An illustration of the mission that the UUV must complete	126
Figure 98: An integrated design pipeline for propeller-based UUV design	127
Figure 99: AI-enhanced design workflow for the design of a propeller-driven UUV.....	130
Figure 100: Integration of AI-based exploration with the UUV design analysis workflow.....	131
Figure 101: Multi-disciplinary feasible design architecture for the multi-disciplinary design of the UUV	133
Figure 102: Individual Disciplinary Feasible MDAO architecture implementation of the UUV design analysis workflow.....	134
Figure 103: An illustration of the flight path for the design of an UAV in the Phase 2 hackathon	136
Figure 104: A conceptual single propeller-motor-battery combination model	138
Figure 105: Actual vs Predicted Thrust produced by a combination at maximum voltage.....	139
Figure 106: Illustration of the mission path around the Queen Elizabeth islands where the OBS payload must be dropped off at (80° 23' 12" N, 85° 06' 46" W) at a depth of 547 m.	141
Figure 107: Design process used in the evaluation of the performance of the UUV design.	142
Figure 108: Battery Pack Sizing Considerations	142
Figure 109: Design process incorporating AI methods to accelerate the simulation of the design.	143
Figure 110: Subspaces considered in the design optimization of the UUV.	144
Figure 111: Illustration of methods and algorithms used in the subspace and collaborative optimization	145

Figure 112: Illustration of the UUV along with the comprised components identifying their positions	146
Figure 113: Designed controller performance illustrating the successful ability to meet all the waypoints and complete the mission with nominal unused energy remaining.....	147
Figure 114: An abstract illustration of the steps involved in the design consideration of DNN hardware accelerator	149
Figure 115: An illustration of the steps involved in the design workflow of 3D-ICs	150
Figure 116: The proposed integrated hardware accelerator design framework.....	151
Figure 117: The M&S steps involved in the multi-fidelity design flow.....	152
Figure 118: Design space representation using ACEL for IC	153
Figure 119: Graph grammar used for design generation.	154
Figure 120: Adder Simulation	155
Figure 121: Counter Simulation.....	155
Figure 122: Design graph for depth 5	156
Figure 123: Design graph for depth 20	156

LIST OF TABLES

Table 1: An enumeration of the AI algorithm developed and evaluated in the program	13
Table 2: The retrieval results from the query “internal combustion engine”	18
Table 3: The retrieval results from the query “pumpjack”	18
Table 4: Performance of pre-trained, un-pretrained and LoRA finetuned models. It is noted that t5-small and t5-base have been finetuned with downstream applications such as summarization and translation while the ones with v1_1 have not gone through any downstream application finetuning.....	84
Table 5: Notations for six degrees of freedom UUV system.....	99
Table 6: Tuned control parameters.	103
Table 7: A summary of the challenge problems considered in the program	106
Table 8: Summary of the flight paths test benches for which the UAV is to be designed for....	108
Table 9: Design space for the UAV system.....	110
Table 10: Seed designs provided by design challenge organizers for the UAV design challenge	113
Table 11: Top-performing synthesized UAV designs using the Systemic Generative Engineering framework	114
Table 12: Summary of the designs generated during the UUV design challenge	120
Table 13: Design space for the UAM design challenge	122
Table 14: Seed UAM designs summary	124
Table 15: Top generated designs using the Systemic Generative Engineering framework	125
Table 16: A summary of the optimal designs discovered by the Systemic Generative Engineering framework	135
Table 17: A comparison of the different MDAO architecture implementations	136
Table 18: Scoring criterion for Phase 2 hackathon	137
Table 19: Seed UAV designs summary	139
Table 20: Top generated designs using the Systemic Generative Engineering framework	140
Table 21: Expert generated best design.	140
Table 22: Characteristics of the designed UUV	146
Table 23: Milestones achieved in Phase 1 and Phase 2 of the program	158

1.0 SUMMARY

This report provides a detailed account of the research performed in the first phase of the Defense Advanced Research Projects Agency (DARPA) – Symbiotic Design for Cyber-Physical Systems (SDCPS) program [1] under the project titled “Systemic Generative Engineering” bearing the contract number FA8750-20-C-0542. The project addresses technical area (TA) 1 of the program, aiming to develop algorithms and frameworks enabling the creation of AI co-designers. The period of performance of the SDCPS program was September 24th, 2020, to September 1st, 2023. Any opinions, findings, conclusions, and recommendations presented in this document are those of the author(s) and do not necessarily reflect the view of DARPA or the US Government.

1.1 Executive Summary

This project, entitled “Systemic Generative Engineering”, aimed to create a modular, open-source artificial intelligence (AI) toolchain to: (i) construct, (ii) compose, and (iii) explore vast design spaces that arise in cyber-physical systems (CPS). The design space construction AI ingest previous designs and heterogeneous data and use machine learning (ML) techniques and knowledge graphs (KG) to capture explicit and implicit knowledge (e.g., mappings of requirements to functions, functions to components, components to configurations). The design space construction process accelerates the acquisition and curation of knowledge from existing design corpora; a process that today is mostly manual. The design composition AI generatively synthesizes design alternatives (conventional and novel) and assists the engineers with the automatic generation of evaluation pipelines. The design composition techniques allow engineers to spend their time creating, rather than tweaking tools and models. Finally, the design space exploration AI enable the inverse design of complex CPS driven by a set of target performance metrics and constraints. This enables a capability that is very difficult for experts: find unintuitive and novel design alternatives that exhibit desired performance and behavior characteristics.

Design automation for models with both cyber and physical components is in the awkward adolescence stage where sufficiently complex component models exist but design exploration and optimization across models does not scale beyond domain-specific point solutions. As a CPS design tool provider, and as a complex CPS manufacturer, Siemens understands the limitations of the state-of-the-art well. In our opinion, the crux of the problem lies on the reliance of human expertise for all decision making in the CPS design process. Currently, design automation tools automate the numerical calculations (i.e., simulation, optimization) and the lifecycle management tasks (i.e., traceability, model transformations). However, human experts are entirely responsible for the construction, composition, and exploration of design spaces. The sheer size of a CPS design makes it impossible for a small group of humans to review the data from existing designs to create new designs. Although the use of AI for design automation has been demonstrated in research prototypes, and some algorithms even implemented in design tools, their scope is limited to a single domain (e.g., mechanical).

1.2 Impact

Systemic Generative Engineering presents a revolutionary approach to systems engineering of CPS and greatly impacts various industries such as automobile and aircraft manufacturers; electrical energy producers and distributors; smart buildings and factories operators; and robot manufactures and operators. Systemic Generative Engineering incorporates ML techniques along with KGs to automate manual decisions made by humans during the CPS engineering design process. As a result of this automation, we demonstrate an improvement of over 10x in the design cycle times for the entire system. Additionally, Systemic Generative Engineering facilitates extraction and reuse of (implicit) design knowledge contained in past designs and models. We demonstrate that using even small quantities of seed designs the developed techniques can significantly improve performances of the generated designs (>10x over manual design processes). Consequently, the improved design cycle time and the improved system (or component) performance will lead to significant cost savings over the course of the design process. Finally, in contrast to existing approaches that rely on formal verification for establishing correctness of the generated design, the Systemic Generative Engineering process leverages the rapid evaluation capabilities via ML models to synthesize designs that are “correct-by-construction”, in addition to providing confidence bounds for the correctness claims, thus drastically reducing the iterative redesign process. This fundamentally alters the paradigm of the engineering design process from trial-and-error slow iterations performed in familiar and known-feasible design space to AI-driven iterations that go beyond known design space.

1.3 Innovations

The Systemic Generative Engineering process introduces several novel techniques that advance the state-of-the-art (SOTA) in various fields related to CPS design. We have demonstrated the following innovations during the application of our framework to the DARPA organized hackathons and challenge problems. While there are several other innovations, we highlight some of the key ones that distinguish the Systemic Generative Engineering framework from, both, other performers and existing SOTA techniques.

- The Systemic Generative Engineering framework leverages knowledge graphs to capture engineering knowledge. The knowledge graph stores implicit and explicit knowledge from past designs and decisions. The knowledge graph provides intrinsic capabilities that permit us to quickly explore and search for required data and discover patterns across knowledge that can even take geometric or topological structures. These can be used to characterize the different aspects of the designs. Further, knowledge graphs provide the ability to reason over specific aspects of the design such as inferring unobserved relationships between the various elements of the design space.
- The Systemic Generative Engineering framework leverages a multi-domain aligned design space on which design space exploration is executed. Overcoming the traditional sequential and iterative nature of the multi-disciplinary design problem, Systemic Generative Engineering enables exploration over all disciplines using inverse design methods that synthesize requirements specific design configurations that are consistent across the different design disciplines considered.
- Systemic Generative Engineering framework leverages probabilistic methods to overcome the need for large training data corpora to develop machine learning models. Furthermore,

the framework has demonstrated the use of physics-informed reduced order modeling techniques that can leverage the underlying physics knowledge to develop fast evaluation surrogates for highly non-linear and complex turbulent flows.

- The Systemic Generative Engineering framework investigates the use of public domain knowledge to automatically expand the design space based on “crowd engineering” capabilities. This ability is realized by automatically identifying functional behaviors of components in the design leveraging natural language processing and graph embedding techniques and augmenting it with the ability to find functional alternatives to the identified functions using public domain knowledge graphs and document embeddings. We call this capability “open world design” as our framework leverages a vast source of public domain knowledge to enhance the design capabilities and realize innovation.
- The Systemic Generative Engineering works with the designers through recommendation and user preference adaptation systems to interactively construct, compose and explore the multi-dimensional and multi-disciplinary design spaces. These recommendation systems leverage adaptive sampling and Bayesian optimization to learn optimal samples based on user defined objectives. This contrasts with the traditional optimization techniques that rely on complete automation of design pipelines and, hence, are unable to incorporate user guidance in design processes. On the other hand, the Systemic Generative Engineering framework can embed user preference in learning design spaces and design constraints in ways where users can identify preferences on the seed designs or users can provide a specification of the constraint forms that are of interest to guide the exploration. We leverage weighted Bayesian posterior estimation and program synthesis techniques that permit our algorithms to adapt to the provided background knowledge and user preferences.
- We demonstrate the ability for the Systemic Generative Engineering framework to explore innovative designs owing to the underlying graph representation of the designs chosen. Systemic Generative Engineering leverages a multi-scale graph representation of the design where system, subsystems and component details are encoded in graphs. This permits us to explore various topologies of designs by modification of graph spaces. We develop efficient path auxiliary sampling techniques to explore the graph space such that Markov Chain Monte Carlo (MCMC) algorithms can minimize the number of rejected samples during the design space exploration process.

1.4 Project Summary

The development of the Systemic Generative Engineering framework has been divided into three categories with each category being addressed in the three phases of the SDCPS program. The three categories are:

- **Exploration (Phase 1):** The goal of the research in this phase of the work is to evaluate the proposed hypothesis that the use of knowledge graphs for design storage, use of machine learning in design space construction, design composition and design space exploration can significantly improve the design cycle times for complex products such as cyber-physical systems. Further, the exploration phase aimed to investigate a multi-disciplinary design problem in which we are considering a minimum of design in two distinct but interrelated disciplines. The outcome of this phase of work is meant to be standalone machine learning algorithms and tools that can be used designer-in-the-loop to execute the novel Systemic Generative Engineering design process.
- **Maturation (Phase 2):** The goal of the research and development in the second phase of the project is to build on the findings in Phase 1 and integrate the promising methods identified. Furthermore, we aim to expand the scope of the research tackling more advanced topics in AI-assisted design of complex CPS while focusing heavily on multi-disciplinary and multi-scale considerations in the design process. This phase of work is aiming to produce AI techniques that can create innovation in designs beyond topological variations in system representations. The outcome of this phase of work in addition to advancing research focused AI methods for design is to develop AI assistants that can support design engineers by learning from experts and guiding novices.

1.5 Document Organization

The rest of the document is organized as follows – the second chapter, Section 2, introduces the work performed in the program providing the background and the motivation for the development of the Systemic Generative Engineering framework. Section 3 introduces the use-cases that have been addressed in the program, highlights the framework developed and provides the technical details of the methods investigated. Next, Section 4, presents detailed results of each of the technical and design challenges considered and highlights the results of the technologies evaluated. Finally, Section 5 concludes the document summarizing the achievements of the team.

2.0 INTRODUCTION

2.1 Evolution of Designs

Over the past few decades, there has been a drastic change to the manner in which complex systems, such as aircrafts, automobiles, gas turbines, etc., are designed. The designs have migrated from being highly mechanical to being multi-disciplinary with a heavy focus on incorporating electrical and cyber considerations. Previously, the mechanical designs consisted of very few interacting components and any interactions were mainly mechanical in nature. The evolution of designs has resulted in the integration of computational, control and software elements in the design, and as a result the products demonstrate highly interrelated subsystems and components that are in themselves complex. These advances and changes to the design process have resulted in the creation of Cyber-Physical Systems, as illustrated in Figure 1.

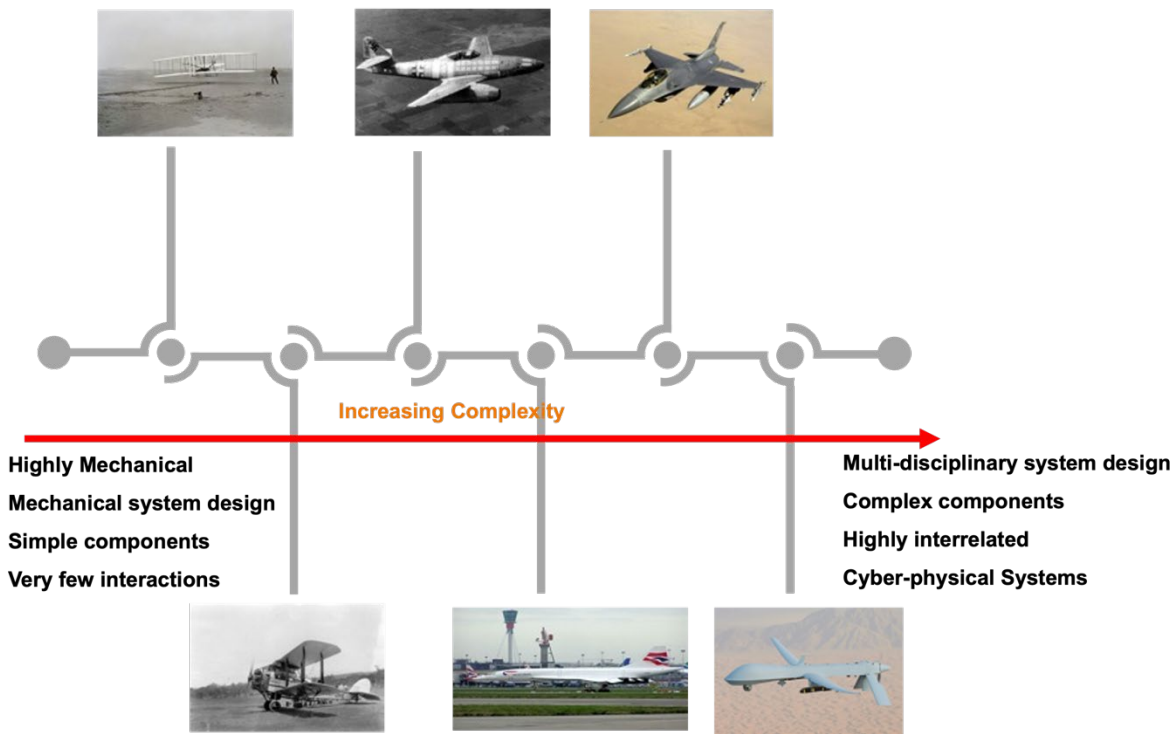


Figure 1: The evolution of design complexity in the past few decades

Researchers have studied the correlation between the complexity of modern designs and characterized them in terms of the number of interactions in their subsystems and components. The study identified that there is a direct correlation between the domain knowledge required to design a product and complexity in the design process (as a function of the number of decisions made by a designer) and the number of interactions between the subsystems and components of the product being designed. The study further identified the relationship between the domain knowledge and expertise required as having a super-linear relationship with the component interactions and that the design complexity grows exponentially with the number of interactions.

Although the complexity of systems and the associated design processes have grown exponentially, the way we design them is still fundamentally the same. The design processes still rely on

traditional systems engineering techniques following the iterative process of requirements elicitation, requirements decomposition, design synthesis, fabrication and finally followed by verification and validation. Thus, the goal of the Systemic Generative Engineering project is to fundamentally alter the paradigm of design of complex systems such that we can overcome the increasing demand on design knowledge and expertise and develop new design processes that can successfully manage the increasing design complexity. The Systemic Generative Engineering project aims to,

Teach machines to design better than humans.

To achieve this vision for AI-assisted design systems, we realize there is a need to,

Teach machines to reason.

It is imperative that AI systems be able to reason about the data in the design process. The purpose of this reasoning must go beyond the capabilities of statistical reasoning, as there is a need to develop systems that can infer reusable elements of knowledge from the available data. The reasoning capabilities System Generative Engineering aims to realize will permit AI design systems to reason about the data to guide decisions in both a symbiotic and an autonomous manner.

Teach machines to imagine.

While reasoning about data permits the creation of new designs, it is highly unlikely that the designs that are created will be innovative. Thus, there is a need for the AI-based design systems to be able to create innovation beyond the obvious that can disrupt existing technologies and thus create innovation. In humans, the innovation and creativity come through the ability for imagination, which the Systemic Generative Engineering design process aims to reproduce in AI systems.

2.2 Complexity in Design of Cyber-Physical Systems

The traditional – largely manual – approach to systems engineering and CPS design is faced with the following challenges:

- **Convergence:** The multi-disciplinary nature of complex systems requires detailed considerations of several disciplines simultaneously. This results in an iterative design process in which the different disciplines are solved separately by different teams in siloed considerations of subsystem and disciplinary design problems. Thus, iterative design practices that require convergence between the different design disciplines are necessary. Unfortunately, this results in very large design cycles to an extent where large complex systems such as aircrafts or gas turbine, take years to realize a single design using the traditional practices.
- **Exploration:** As a by-product of the multi-disciplinary design considerations, there is an explosion in the design space that is to be explored. These design spaces are often a combination of discrete sets ranging in the order of 10^{100} , continuous spaces spanning multiple interrelated disciplines, and unexplorable representations of space such as software programs, hardware deployment models, etc. While traditional approaches rely on human expertise in determining appropriate combinations of design parameters, the number of designs explored are limited to a handful evaluated at "design conditions". Even with modern

automated design space exploration techniques, only a very small subspace of the design space is explored offering little insight into the design performances.

- **Correctness:** Finally, the generated designs offer little in the way of guarantees of performance when deployed. This results in tedious design-build-redesign loops that are not only time consuming, but also expensive. The guarantees provided are often limited to a very small set of operating conditions considered during the design process, resulting in catastrophic failures in the design due to insufficient reliability and safety considerations.

Figure 2 illustrates one such example where all the challenges described above are evident. Consider the design of a military-grade unmanned aerial system (UAS) for the purpose of intelligence, surveillance, and reconnaissance. The design of the UAS relies on reaching optimal performance in several disciplines and involves designing several subsystems. This design problem requires exploration of multiple different disciplines each having its own design space which can both be continuous or discrete or a mix of the two.

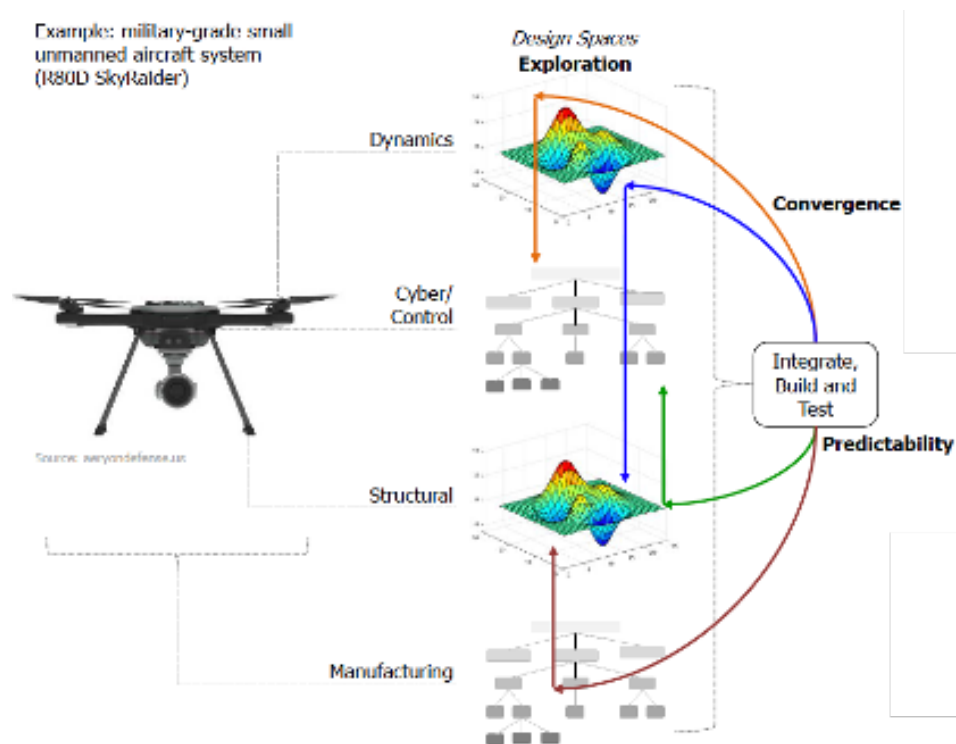


Figure 2: Challenges in the design of CPS [Adapted from 1]

Further, as traditional design processes have large teams working on segments of the design, the individual design teams are siloed with a few contact points where information is exchanged. Thus, there is a need for one of the teams to be able to predict the outcome of another team (for example, the manufacturing team will need to predict the structural design characteristics) to accelerate the design process. There is little room for concurrent design practices in such highly interrelated systems. Finally, as the design processes rely on iterative development of the design through fabrication and testing, there is a need for convergence between the various disciplinary outcomes and the final design performances. Thus, the design process is one of “construct-by-correction” where

the designs themselves are not ready for deployment as soon as they are designed and fabricated. Thus, the design process of complex CPS is limited by the factors of exploration, convergence, and predictability.

While recent developments [2] have attempted to address some of these issues, several of the steps involved in the new processes introduced are extremely manual and require considerable domain expertise. Thus, the challenges presented above can be translated to a set of engineering considerations as follows –

- **Design problem complexity:** A CPS combines the domains of mechanics, electrics and software and therefore has a wide variety of components we can choose from. Unfortunately, that creates a high dimensional design space in the order of 10^{100} which is hard to explore using automated techniques, let alone a human engineer.
- **Design objective variety:** One CPS can have different design objectives e.g. cost, mass, size, performance, or safety. This creates a multi-objective design problem when more than one objective needs to be followed. The multi-disciplinary aspect of the design implies each discipline introducing its own design considerations which are often different from that of the overall system. Translating these disciplinary requirements to that of the system remains an art.
- **Experienced-based design:** Currently good CPS design is mainly created by the expertise of human engineers. This expertise is defined by years-long work in a certain domain space, the knowledge of previous designs and products, cost and time constraints and use and the ability to exchange knowledge with other domains. CPS design combines several domains, each requiring large teams of engineers and domain-specific teams with extensive subject matter expertise.
- **Constraint design creation:** Current CPS designs are defined by the inability to discover innovative and high-performance design solutions due to cost and time constraints. This leads to limited exploration of familiar and known-feasible design spaces and therefore new and innovative designs cannot be discovered.
- **Design tools:** Current design tools for CPS design creation, exploration and optimization do not scale beyond domain-specific point solutions for automotive, aerospace, medical, etc. In addition, a typical CPS design involves hundreds of domain-specific tools.
- **Development Cycles:** Long design cycles due to creation of high-fidelity models and time-consuming simulations lead to trade-offs between model fidelity and performance uncertainties. An additional disadvantage is that redesigns are costly due to disconnected discipline-specific design processes which fail to capture the extremely important cross-domain relationships and interactions.

2.3 Systemic Generative Engineering

To address these design challenges, researchers at Siemens Corporation, Technology, Massachusetts Institute of Technology, and the University of Pennsylvania are developing the Systemic Generative Engineering design process and the associated framework.

Analyzing the term “Systemic Generative Engineering”, we see that it is comprised of –

Systemic – Something that affects the entire process.

Generative Engineering – Something that is engineered by machines for humans.

Thus, Systemic Generative Engineering is defined as –

An AI-based approach to systems engineering that enables the synthesis of context-specific virtually verified CPS

Systemic Generative Engineering is enabled through a set of key technologies, namely –

- **Knowledge graphs** as a storage for explicit and implicit knowledge about the design process.
- **Graphs** as a representation of designs at various levels of detail and graph embeddings as means for operation on graph representations.
- **Logic reasoning** methods that allow incorporation of domain knowledge into AI methods to overcome the dependence on “big data” methods.
- **Probabilistic inference** techniques that permit the assessment of incomplete designs with uncertainty and enables robust design techniques.
- **Multi-disciplinary Analysis and Optimization (MDAO)** that enables acceleration of complex design processes using advanced design automation architectures.
- **Inverse design** that enables the synthesis of requirements-specific designs without the need for lengthy iterative optimization process.

Systemic Generative Engineering builds on the work performed at Siemens Corporation, Technology in the field of Generative Engineering. Figure 3 provides a conceptual illustration of how the Systemic Generative Engineering process is realized. At a high-level, 30,000 feet view, the process is comprised of three (3) steps.

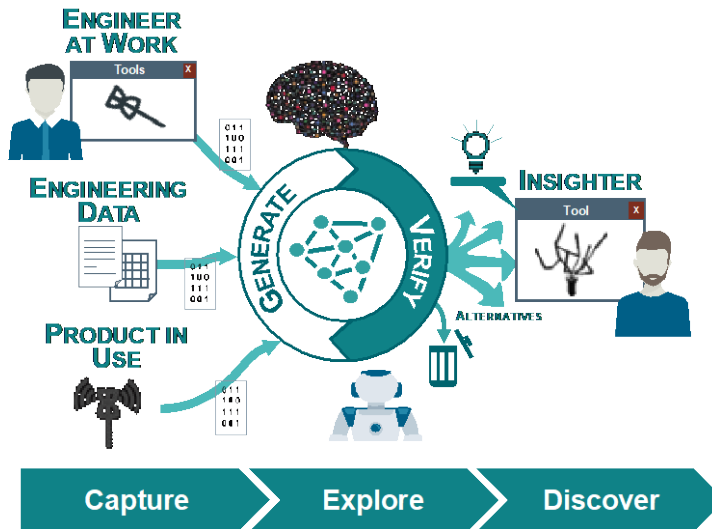


Figure 3: Systemic Generative Engineering – How it works.

A knowledge capture stage where we extract knowledge from various sources including past data, engineering handbooks, engineering processes and even, designers exercising a design process. The knowledge captured is distilled into machine readable formats using a set of domain specific knowledge distillers. These distillers refine knowledge from the various sources identifying the relevant pieces of information for storage and develop a semantically related representation for the information refined.

The process then moves to the exploration stage. Here, the knowledge stored is used to train AI models that can autonomously explore the design spaces by generating various alternatives and continuously validating them for this performance, thus dramatically accelerating the engineering design and decision-making process. The AI models churn through millions of designs in seconds due to the vectorized representations of designs allowing exploration of much larger design spaces, combinations, and topologies that previously imaginable.

Finally, having crunched through a vast set of design alternatives, the process moves to the discovery stage in which the AI models identify a set of design alternatives and provide feedback to designers in design tools that they are familiar with. This process involves the AI agent assisting the engineers during the development process by suggesting alternate approaches to design of the complex CPS and enabling them to effectively work across multiple disciplines.

Mapping to the DARPA SDCPS program, the knowledge capture addresses the challenges of the design space construction, while the challenges in the design composition are addressed both by the knowledge capture and the exploration phases, and finally the challenges faced in the design space exploration are addressed through the goals of the exploration and discover stages.

2.4 Framework Details

Figure 4 illustrates the Systemic Generative Engineering AI workflow including the three TAI sub-modules.

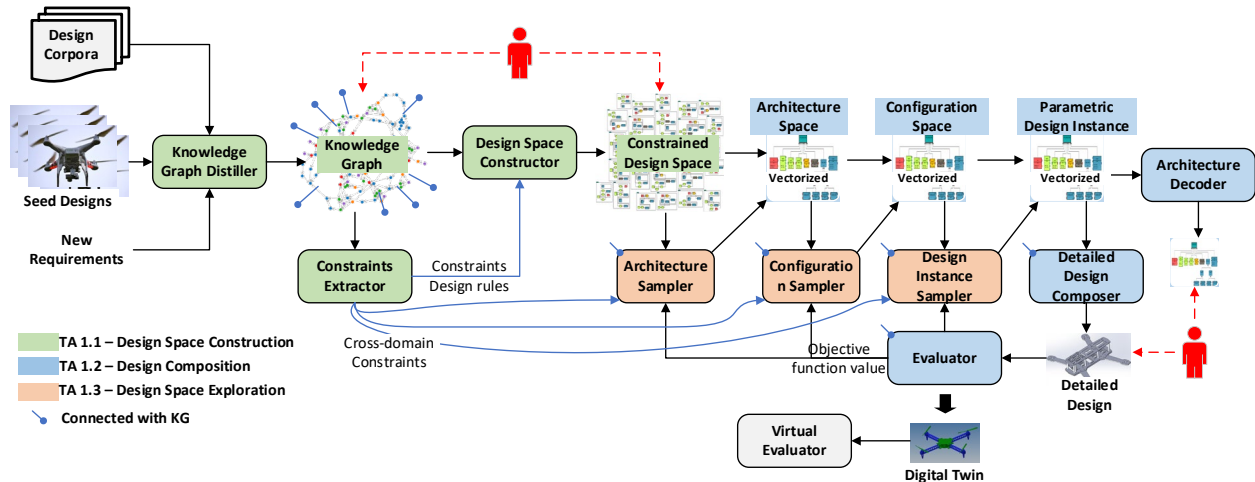


Figure 4: Modules of the Systemic Generative Engineering framework and their associated interactions

The process begins with TA3 performers supplying a set of design corpora. The design corpora can be in the form of component library elements and model library elements. Elements of the component library can be of the form of components models such as computer aided design (CAD), manufacturing, attributed eXtensible Markup Language (XML) etc., and the models in the model library are envisioned to be composed of domain specific models such as Simulink models representing control simulations, OpenFOAM models representing computational fluid dynamics (CFD) simulations, etc. Each of these models are passed through domain specific “distillers” that extract the contents of the models to populate a knowledge graph (represented by a graph of graphs). The user interaction begins with the specification of a set of seed designs and, optionally, a set of requirements. The seed designs are expected to be of the form of Cyber-physical models (CyPhyML) or System Models (SysML). Application specific distiller, e.g., OpenMETA distiller, will populate the KG with information from the seed designs. These seed designs can not only identify new components for the KG but also identify relationships between different elements of the graphs. The Constraint Extractor mines the KG to extract logical and mathematical constraints across multiple fidelities and domains. Through the TA2 interface, a user can curate the generated KG by manually injecting design rules and constraints. The Design Space Constructor maps the requirements to an architecture diagram. Upon applying the extracted constraints, the fully connected architectural diagram results in a set of most likely system architectures leading to generation of Constrained Design Space. Simultaneously, three joint and hierarchical sampling mechanisms – Architecture Sampler, Configuration Sampler, and Design Instance Sampler are used to generate an encoded representation of the Architecture Space, Configuration Space, and Parametric Design Instance. These encodings will be in the form upon which ML algorithms can be readily applied, for e.g., an adjacency matrix or a vector. For each generated sample, the Detailed Design Composer is executed to generate a set of evaluable models that can predict or evaluate the performance (over different domains) of the generated design. The composed models are utilized in an Evaluator to estimate the performance of the design. These performance metrics provide indication of how the distributions on the various samplers, mentioned above, should be updated to continually improve the generated design, in increments, to meet the requirements specified by the designer. The final output of this workflow will be a detailed “correct-by-construction” design of the CPS, represented as a digital twin. If a virtual evaluation framework such as robot operating

system (ROS), Unreal Engine, or Unity 3-dimensional (3D) is provided then the performance of the digital twin can be evaluated against the mission.

The Systemic Generative Engineering framework operates at three levels of granularity of the design. For clarity, we provide their definitions below –

- **Architecture:** A design architecture represents the allocation of functions to logical (i.e., abstract) components and their interconnection (e.g., a battery connected to a motor and a processor). This can be viewed as a product of the conceptual system-level design of the CPS. Here the designers and design teams are identifying the appropriate system topology that should be used to best satisfy the requirements that have been put forth. The process typically involves a functional allocation in which requirements are decomposed into a set of functions and the functions are in turn allocated to logical elements in the design topology.
- **Configuration:** A design configuration allocates the logical components to specific technology components (e.g., commercial-off-the-shelf (COTS) Lithium-Polymer battery). The COTS components are sampled by Systemic Generative Engineering Framework (SGEF) AI from the machine learned design space. This can be viewed as the output of the preliminary design process in the design pipeline. As part of the generation of the configuration, the logical components identified in the system architecture are allocated to physical instances that can be further refined in the downstream design processes. At this stage, the design teams have a good understanding of not only the system level characteristics and behaviors, but also have an inkling of the physical and cyber characteristics of the subsystems and components in the system design which permits initial sizing of the vehicle with multi-disciplinary considerations.
- **Design Instance:** The detailed design instance represents the parameterized design configuration in which the characteristics of the design components are determined (e.g., 1500mAh COTS Lithium-Polymer battery). From the physical design perspective, the detailed design involves exploration in a continuous design space in which specific parameters of the design are explored for. But, on the other hand, detailed cyber designs may involve exploration in more complex spaces such as software implementations and hardware parameters which are typically discrete in nature. In summary, the design instance builds on the configuration and identifies the values of the parameters associated with the design with a very focused consideration of the multi-disciplinary and interrelated nature of the system operation.

Although Figure 4 illustrates these three different design representations as distinct steps in the Systemic Generative Engineering process, it must be noted that this is strictly illustrative. The graph-based and vectorized representations of designs permit us to scale across the different granularities automatically such that a single architecture can be synthesized into several detailed design instances and each design instance would map back to a system architecture automatically. During the first phase of research, development, and implementation of the Systemic Generative Engineering framework, we have explored a variety of techniques. The technical methods developed and demonstrated are listed in Table 1.

Table 1: An enumeration of the AI algorithm developed and evaluated in the program

Topic	AI Techniques Developed
Design Space Construction	Probabilistic synthesis of design spaces from seed design corpus
	User preference embedding in design space synthesis
	Meta-synthesis of design spaces
	Open world design
	Constraint synthesis using program synthesis methods
	Constraint synthesis using dynamic programming
	Constraint synthesis using neural logic programming
	Functional abstraction for design graphs
	Design representation as a program using graph abstraction methods
Design Composition	Conditional Convolutional Variational Autoencoders for design completion
	Conditional Graph Variational Autoencoders for design completion
	Graph Autoregressive models for conditional design completion
	Dynamic Programming-based Design Completion
Design Space Exploration	Hybrid Adaptive Optimization
	Surrogate and Reduced order modeling for Accelerated Simulation
	SAT solver-based Architecture Generation
	Invertible Neural Networks
	Modeling Cross-Domain Interactions
	Advanced Multi-disciplinary Optimization Architectures
	Path Auxiliary Sampling for MCMC
	Hybrid Topology Optimization
	Reinforcement Learning for Controller Design
	Bayesian Optimization for Spaces with Parametric Discrete Choices
	Solver-aided Controller Design

3.0 METHODS, ASSUMPTIONS, AND PROCEDURES

This section introduces all the technical details of the methods explored and available within the Systemic Generative Engineering Framework. The following sections introduce each technology, highlighting the background and motivation, methodologies involved, the algorithm developed, and conclude with results on evaluation test cases. The techniques span that of design space construction, design composition and completion and design space exploration.

3.1 Open World Design

3.1.1 Background

Component-based design approaches struggle with the limitation that novelty is often difficult to attain. This is because disruptions to existing technology requires the introduction of new methods of achieving functional behavior exhibited by existing components. For example, tires on a vehicle achieve two functions:

1. Tires function to realize friction with the traversal surface (e.g., roads, snow, etc.).
2. Tires with the help of propulsion systems help propel the vehicle tangential to the direction of rotation.

When the first function the tire is considered, a functional alternative to it would be a continuous track (e.g., observed in tanks). These components require a modified drive train due to the change to the end propulsion system. While the technology does not introduce a drastic change to the design, the resultant effect of the introduction of the technology (e.g., need for cooling systems, introduction of additional axles, etc.) can result in significant modifications to the system. Furthermore, the resultant control behavior and operational scenario can be quite drastically different given the increased friction.

On the other hand, when considering the second function, a potential alternate technology that can realize similar function is that of magnetic levitation (i.e., maglev). While often used in environments in which the ecosystem can be controlled, there have been recent investigations into the possibility of the use of Maglev technology in other environments as well. The introduction of this technology will cause disruptions to existing designs that are almost always four wheeled vehicles.

3.1.2 Concept Details

Our concept utilizes ideas from the information retrieval domain. Recent developments in information retrieval for the question-answering domain has demonstrated the ability to perform complex reasoning (i.e., multi-hop reasoning) to provide answers to non-obvious questions. Further, algorithms have demonstrated the ability to combine logic operators, symbolic formulae, and natural language in a reasoning process to answer common-sense questions.

While our eventual goal is to achieve the ability to perform common-sense reasoning for the engineering domain, we start small by attempting to reason over a combination of structured and unstructured data to potentially perform hop option generation. We first recognize that a large quantity of public domain data that is usable is in textual form. Thus, our initial investigation focuses on the possibility of reasoning over this combination (like the question and answering (QA) domain).

The process begins with the curation of the dataset. This process can be divided into an offline training step and an online inference step. The offline training step is a training process in which multiple models are trained in an unsupervised manner to discover a joint language. The generated

embeddings are stored in a database, which is used when a designer queries for a new entry to identify similar entries by computing a Euclidean distance between the embeddings.

The realized process is illustrated in Figure 5, where we combine the embeddings of concepts learnt by a language model with those learnt by a graph embedding model that captures the semantic structure of the knowledge graph that links the various concepts together in the model. When attempting to identify alternative components for those identified in the design space, we compare the embeddings of the concept defined in the design space to those that are already constructed. Due to a limitation of time, the embeddings are currently loaded into memory, but the interface between the model and the search engine can be optimized using a vector database (e.g., Elastic [3]).

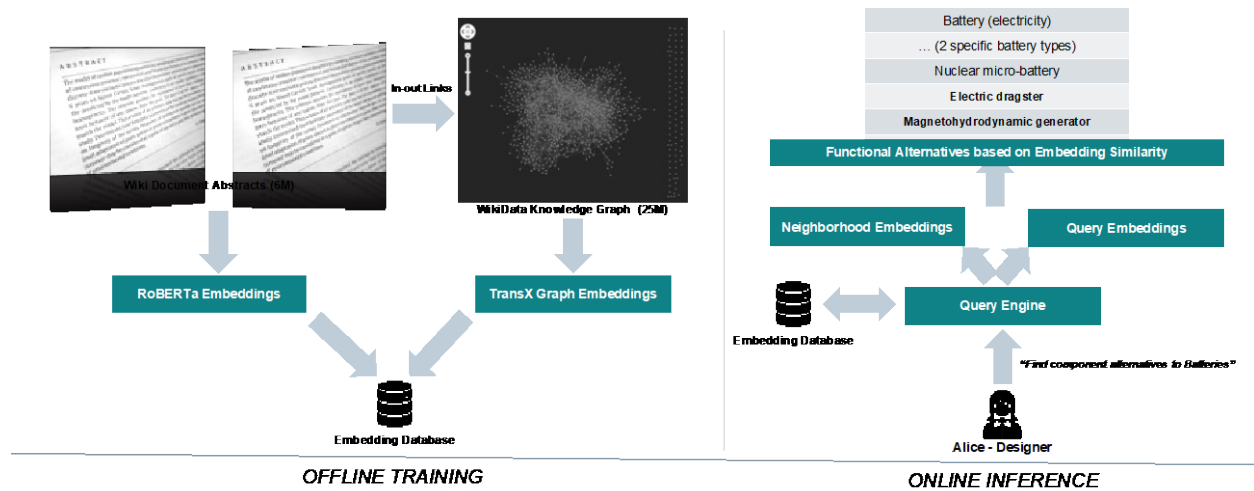


Figure 5: Open World Design Concept

3.1.3 Implementation

Language Model Embedding

We adopt Robustly Optimized Bidirectional Encoder Representations from Transformers (BERT) Pretraining Approach (RoBERTa) [4] embedding to represent the pages and articles (a.k.a “entities”) in the Wikipedia. The RoBERTa is a language model based on BERT language model [5] with modified key hyperparameters, removed option of next-sentence pretraining and much larger mini-batches and learning rates. It also uses Byte-level Byte-Pair-Encoding (BPE) as a tokenizer instead of the WordPiece tokenizer. The RoBERTa model consumes a sentence, paragraph, or the whole article and outputs a vector of numerical embedding to semantically represent the input. To minimize the distraction from lengthy documents and the computational workload, we only take the first paragraphs of the Wikipedia articles, which are presumably considered the abstract of a page or an article, as the input of the RoBERTa models.

In our implementation, we utilize the pretrained RoBERTa model (namely “roberta-base”) to extract the embeddings of each entity. It is important to note that while the “roberta-base” pretrained RoBERTa embedding are trained given the assumption that each input does not exceed the length of 512 tokens, our paragraphs are usually longer than that. To mitigate the issue, we again utilize the pretrained tokenizer for the “roberta-base” RoBERTa model and split the token list into sub-

lists $[X_1, X_2, \dots, X_n]$ whose lengths are shorter than 512. We take the averaged embedding $\bar{Y}$ of all embeddings of the sub-lists:

$$\bar{Y} = \sum_{i=1}^n \mathbf{Y}_i = \sum_{i=1}^n \text{RoBERTa}(X_i), \quad [1]$$

where $\text{RoBERTa}(\cdot)$ denotes the pretrained RoBERTa model, and each embedding $\bar{Y} \in \mathcal{R}^{768}$ because the pretrained “roberta-base” hyperparameter outputs 768 dimensions for each entity.

Graph Embeddings

Overview

The aforementioned RoBERTa language model embedding merely reveals the intrinsic representation of each page and article, while the relation among the articles remains unknown.

Wikipedia comes with a crowd-sourced graph built upon the in-links (hyperlinks that direct into current entity) and out-links (hyperlinks that direct to another entity) and frequently acquires updates with the endless and restless efforts of volunteer editors. The start points of those links are named as “heads”, while the end points of those links – sometimes known as relations – are “tails”. A good graph embedding technique provides good prediction of any one of the head, relation or tail given the other two.

We attempted 4 different graph embedding mechanisms and techniques: *TransE*, *TransH*, *TransR* and *TransD* and we denoted all of them as *TransX*. All *TransX* techniques denote the embedding of the heads $\mathbf{h}$ and the tails $\mathbf{t}$. Those methods rely on a score function $f_r(\mathbf{h}, \mathbf{t})$ which evaluates the quality of connection among heads, tails, and links in the numerical space. The smaller $f_r(\cdot)$ denotes a better connection among the head, tail, and the links. We denote “good” or “positive” connection as $f_r(\cdot)_{pos}$, which are presumably quite small while we denote “bad” or “negative” connection as $f_r(\cdot)_{neg}$ which should be larger than all $f_r(\cdot)_{pos}$ in a *margin*. The *TransX* methods utilizes the same loss function to provide guidance during the training phase:

$$L(h, r, t) = \max(0, f_r(\mathbf{h}, \mathbf{t})_{pos} - f_r(\mathbf{h}, \mathbf{t})_{neg} + \text{margin}) \quad [2]$$

where *margin* is a hyperparameter to denote the differences between good and bad scores of heads, tails, and relations.

TransE

Proposed in [6], *TransE* embeddings attempt to capture the embeddings of heads, tails and relations based on the assumption of

$$\mathbf{h} + \mathbf{r} \approx \mathbf{t}, \quad [3]$$

and the score function is defined as

$$f_r(\mathbf{h}, \mathbf{t}) = ||\mathbf{h} + \mathbf{r} - \mathbf{t}||, \quad [4]$$

where we use L_1 norm here.

TransH

Proposed in [7], *TransH* attempts to discover the connections among heads, tails on a normalized *hyperplane* regulated by the relation.

In this framework, the embedding of the relations $\mathbf{d}_r$ are sampled from a normalized hyperplane which ensures $||\mathbf{d}_r||_2 = 1$. The embeddings of heads and tails are now defined as the projections on the hyperplane.

$$\mathbf{h}_\perp = \mathbf{h} - \mathbf{w}_r^\top \mathbf{h} \mathbf{w}_r, \quad [5]$$

and

$$\mathbf{t}_\perp = \mathbf{t} - \mathbf{w}_r^\top \mathbf{t} \mathbf{w}_r, \quad [6]$$

where $||\mathbf{w}_r||_2 = 1$, and like *TransE*, the embeddings are pursued as:

$$\mathbf{h}_\perp + \mathbf{d}_r \approx \mathbf{t}_\perp, \quad [7]$$

and the score function is defined as an L_2 norm:

$$f_r(\mathbf{h}, \mathbf{t}) = ||\mathbf{h}_\perp + \mathbf{d}_r - \mathbf{t}_\perp||_2^2. \quad [8]$$

TransR

Proposed in [8], *TransR* approach splits the entities (heads and tails) and the relation into two semantic spaces, and the framework attempts to project the entities onto the space where relations reside:

$$\mathbf{h}_r = \mathbf{h} \mathbf{M}_r, \mathbf{t}_r = \mathbf{t} \mathbf{M}_r. \quad [9]$$

The connections of heads, tails and relations are evaluated as:

$$f_r(\mathbf{h}, \mathbf{t}) = ||\mathbf{h}_r + \mathbf{r} - \mathbf{t}_r||_2^2 \quad [10]$$

TransD

Taking a further step, the authors of [9] propose a dynamic projection of embeddings of heads and tails on to two different spaces of relations:

$$\mathbf{h}_\perp = \mathbf{M}_{rh} \mathbf{h}, \mathbf{t}_\perp = \mathbf{M}_{rt} \mathbf{t}, \quad [11]$$

and the score function is defined as:

$$f_r(\mathbf{h}, \mathbf{t}) = ||\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp||_2^2. \quad [12]$$

Selection of the Best Approach

To select the best approach among the 4 *TransX* approaches, we trained 4 different embeddings, these embeddings are randomly initialized with the same random seeds, and we train 2000 epochs on the embeddings using the loss function and Adam optimizer [10].

3.1.4 Retrieval System Design

We assume that the best replacement should be both similar in terms of semantics – which is covered by language model embedding – as well as knowledge structure – which is reflected from the *TransX* embeddings. Given a query q , we concatenate the RoBERTa embedding $\bar{Y}_q$ and the *TranX* embedding e_q as E_q .

We acquire the index j of the candidates with the closest Euclidean distance:

$$j = \arg \min_j ||E_q - E_j||. \quad [13]$$

3.1.5 Dataset Information

We use abstract and link information from DBPedia, a project that dumps Wikipedia and other metadata related to the articles and pages. We extracted and filtered 6 million embeddings of abstracts and 150 million links.

3.1.6 Results

Table 2 and Table 3 demonstrate the top-10 retrieval results based on two sample queries: “internal combustion engine” and “pumpjack”. Besides the queries themselves, the 9 candidates from the retrieval system regarding each query show a reasonable recommendation that may potentially act as or serve in place of the queries.

Table 2: The retrieval results from the query “internal combustion engine”

internal combustion engine		
steam engine	reciprocating engine	turbine blade
axial compressor	microwave electrothermal thruster	stratified charge engine
submersible pump	automotive thermoelectric generator	pulse detonation engine

Table 3: The retrieval results from the query “pumpjack”

pumpjack		
possum belly	mud pump	rig standpipe
rope pump	treadle pump	pressure shale
top kill	tubing hanger	underground pneumatic boring

3.2 Constraint Synthesis

This section discusses the use of knowledge graphs for the purpose of design space construction. In particular, we aim to reason about the structure of the knowledge graph in order to discover patterns that can provide explanations as to the behavior of the design. These explanations can be represented as design space constraints, which can provide a constrained region of space to operate with, potentially improving the performance of the design space exploration. The document considers the system architecture design of the powertrain of a hybrid-electric sports car. Due to the hybrid-electric nature of the design, there is a consideration of both electrical and mechanical components that must be designed, with varying system layouts possible (e.g., series, parallel, etc.).

3.2.1 Use-case Description

We demonstrate the power of knowledge graph on the use-case of the design of the powertrain of a hybrid-electric vehicle (HEV). The use-case under consideration utilizes the Bayerische Motoren Werke (BMW) i8 as its reference model. The setup of the vehicle and its corresponding powertrain is illustrated in Figure 6.



Figure 6: An illustration of the hybrid-electric powertrain of the reference BMW i8 vehicle

Performing a functional abstraction on the reference design and then a physical decomposition of the identified functions, we can identify a set of system components that are suitable for the architectural design. This is one of the functions of the knowledge graph. In the current document, we assume this abstraction and decomposition is already completed and the system definition / formal model is available. Unfortunately, with the abstraction and decomposition process, we do not have any information about the relationships between the different components. For example, information about the compatibility of the different components. Thus, in order to prevent the synthesis of incompatible and unrealistic designs, we aim to learn the constraints from a set of examples of "good" and "bad" designs.

Here the term good implies that the design is a valid design while the term bad implies that the design consists of connections that are invalid. For example, in a bad design, we may have a motor connected to a generator which logically does not make sense.

Based on the functional abstraction and physical decomposition, Figure 7 illustrates the possible set of components that can be loaded onto the HEV powertrain. As indicated in the figure, the HEV powertrain is comprised of an internal combustion engine, a battery, anywhere between one to four motors, an optional generator, an optional gearbox, an optional torque coupler, a set of driven axles which can consist of differential or individual steering mechanisms and finally the chassis.

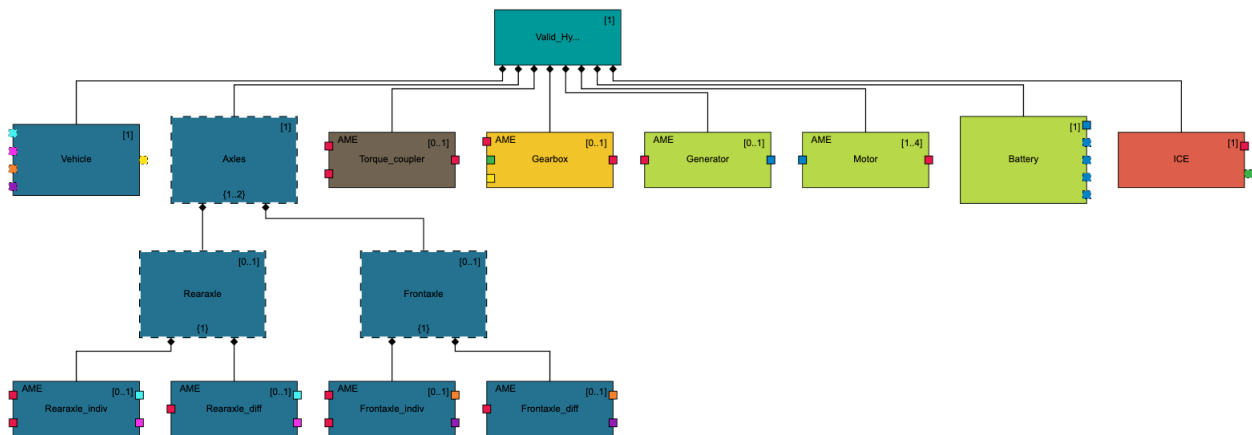


Figure 7: The block definition diagram of the HEV powertrain

As one can imagine, we have a very large number of combinations that can be generated with even this small number of components. It is infeasible to evaluate the various alternatives making it necessary to identify a set of constraints that can significantly reduce the number of possible architecture alternatives. For example, based on knowledge of the components, it is possible to establish a set of constraints that either prevent or enforce connections between the different components. This is illustrated in Figure 8. For example, the figure illustrates that all the connections in red (e.g., the motor – gearbox, internal combustion engine (ICE) – motor, Front Axle – Rear Axle, etc.) lead to invalid designs. Thus, by encoding this information into the design space, we can reduce the search space considerably. Similarly, the connections in green (e.g., ICE – Generator, Torque coupler – Axles, etc.) are desired and should be enforced.

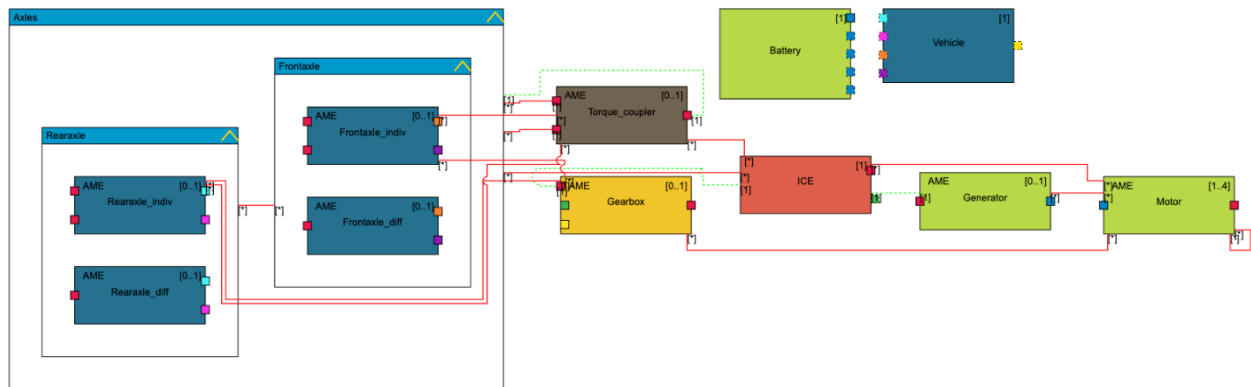


Figure 8: The ground truth design space constraints for the HEV powertrain

While the constraint information can be derived by an expert engineer by analyzing the design space alternatives, our goal is to leverage machine learning to realize this task. In particular, we investigate the use of dynamic programming to automatically synthesize and evaluate the constraints. We aim to achieve this by leveraging the graph-based representation of designs and the semantic relationships between the different components of the design.

3.2.2 Concept Details

As part of the design process, it is essential to understand the interactions and relationships between the different components that are being designed. This is particularly important during the early phases of design where the design spaces are extremely large. Often, these components belong to different disciplines making the design process extremely challenging. During the conceptual design stage, it is essential to understand how the different components interact so that suitable measures can be taken to reduce the number of design alternatives that have to be evaluated. To achieve this, we leverage the concepts of logic programming and rule induction. By leveraging knowledge about several designs, we aim to learn the relationships between the components of the design. That is, we would like to identify which combination of components and what connections between them, yields a good design. It is natural to represent them in the form of "first-order logics". Learnt rules can be used to constrain the design space reducing the number of design alternatives that need to be evaluated during the design process. This process can be viewed as that of subgraph matching to identify which subgraph occurs more frequently in the stored design.

Having identified a reoccurring subgraph, we abstract the graph to identify the corresponding template that best describes the subgraph. An example of a learnt rule is illustrated in Figure 9.

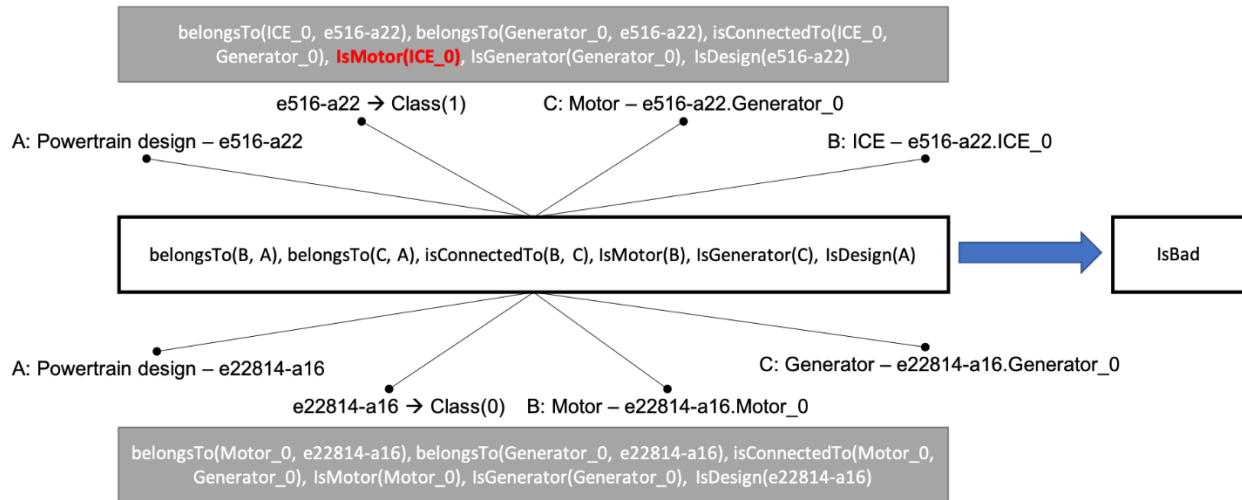


Figure 9: An illustrative example of a logic rule that can be used to build a constrained design space

The figure illustrates the basic idea pertaining to the use of rules as a means constrained design space generation. Consider the situation where several design examples are present in the knowledge graph. A couple of examples of the observed knowledge is shown as the HEV powertrain systems e3461-a2 and e22814-a16. By inspecting these systems, it is evident that the connection between the ICE and the motor is a bad design choice for the powertrain as it provides no reduction to the torque output by the ICE. By observing similar such connections for several other designs, we can abstract the knowledge that the connection between the ICE and the motor leads to a bad system design. This is represented in the logic formula in the center of the figure and evaluated by the representation on the top and the bottom. Thus, we can quickly narrow-down the design space by removing all combinations of ICE – motor thus giving us a smaller design space. The process of learning rules on knowledge graphs (or databases, in general) typically relies on the use of Markov Logic Networks (MLN). MLNs takes in a vast knowledge of data and potential predicates with which rule combinations are to be generated. By combining the predicates with the "constants" in the database, several potential rule combinations are generated. This process is extremely expensive as it requires a combinatorial search of the rule-base to create the potential good rules. Although, we can make the process faster by incorporating smartness into the search process. We track three (3) metrics that are used to evaluate the performance of the synthesized rule:

- The number of correct hits (i.e., true positives) in the knowledge base
- The number of false positives (i.e., when the knowledge base has data disproving the rule)
- The coverage of the rule (i.e., the fraction of data to which the formula is applicable)

A synthesized formula would need to optimize all 3 metrics to be considered good. An ideal formula would be one which has the largest number of correct hits, the smallest number of false positives and has the largest coverage on the knowledge base.

Each generated formula is compared against the knowledge base to quantify these three metrics. This is achieved by grounding the formula on all possible constants. As you can imagine, the number of combinations explodes quickly during this grounding process. Consider a formula which has n predicates, each predicate having m arguments. If each argument has k constants to which they can be grounded, then the total number of grounded formulae that must be evaluated is:

$$n^{m^k} \quad [14]$$

This is exorbitantly large even for small number of predicates. For example, when there are 5 predicates with 5 arguments and 5 constants, we must evaluate $2 * 10^{17}$ grounded formulae.

As indicated, each grounded formula is evaluated on the knowledge base for the above 3 metrics (number of hits, number of false positives and the total coverage). Both the number of hits and number of false positives are evaluated by comparing the generated grounded formula against the content in the knowledge base. For example, consider a formula that indicates that "System A is good if it is comprised of Component B and Component C, with Component B and Component C being connected to each other". One example of its grounded form is: "System e85318-a7 is good if it is comprised of Component motor_0 and Component battery_1, with Component motor_0 and Component battery_1 being connected to each other". To evaluate the goodness of the formula, its grounded representation is compared against all the knowledge that is present in the database in order to quantify its number of hits and misses. If the comparison process finds that the System e85318-a7 is comprised of the Component motor_0 and Component battery_1, then it retrieves its goodness metric. At this point, if the binary goodness indicator of the System e85318-a7 is true, i.e., System e85318-a7 is a valid design, then the number of hits for the formula increase. If this is not the case, i.e., System e85318-a7 is an invalid design, then, the number of false positives is incremented. If System e85318-a7 is not comprised of either component motor_0 or Component battery_1, then the corresponding formula is marked with a miss (i.e., its coverage is lowered).

This process is repeated for each formula and the formulae are sorted based on an overall evaluation metric that is a weighted sum of the 3 (normalized) metrics (equal weights are considered). The process returns the top k first-order logic formulae (measured based on the above 3 metrics) that can be used to constrain the design space while providing the design engineer with sufficient information for reasoning. The choice of first-order logics is natural as it is arguably similar to the manner in which humans' reason about data. Figure 10 illustrates the process of using MLN-based template search for design space constraining.

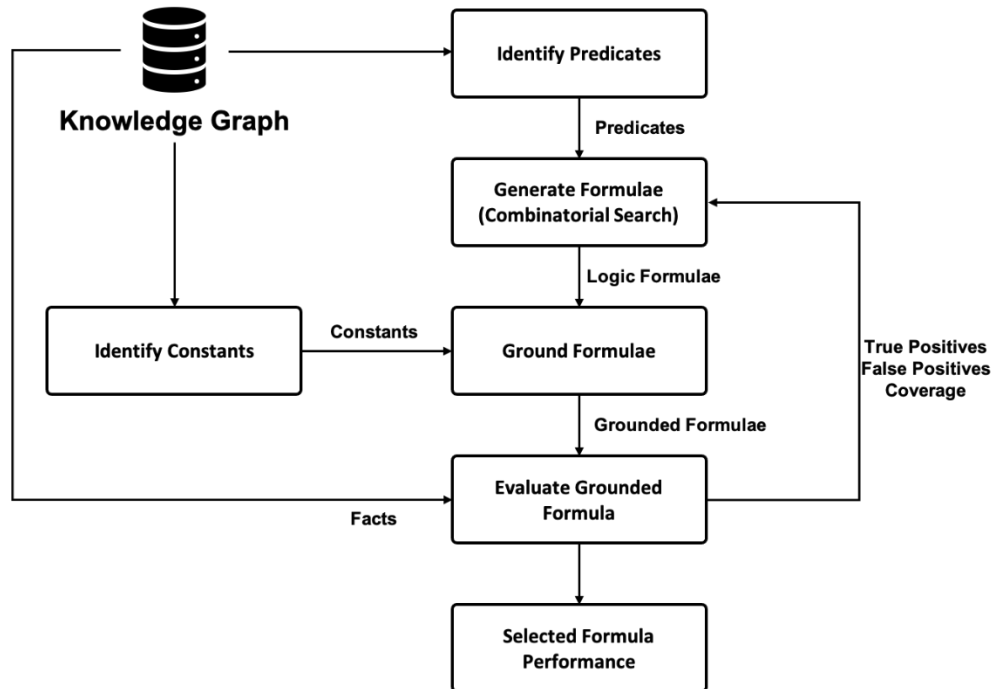


Figure 10: The process for discovery of logic formulae using template search.

To make the process of search more efficient, we leverage dynamic programming to synthesize the potential rule templates. Furthermore, we leverage a reverse-indexing mapping the design to its contents to prevent the necessity for exhaustive searches in the knowledge graph.

- As we generate potential templates that can be suitable, we want to minimize the synthesis of unfeasible templates. Thus, we leverage the information regarding the connections that is already present in the knowledge graph to only synthesize templates that are reasonable. For example, if in the knowledge graph we have never observed an `isConnectedTo` (motor, motor) relationship, then, our template construction engine will not enumerate this as a possible alternative.
- We index each design with the contents of the design. This way, when a grounded formula is generated, we need not have to search for the design through the vast database of designs. We can instead search it based on the architecture, component, and port ids. This optimizes the retrieval time when evaluating the performance of the formula.
- We use dynamic programming when generating formulae. This implies that when generating a formula of length 5, we utilize the information related to formulae of length 4 and synthesize new formulae. If we know that any one particular formula has poor performance when it has a length of 4, then we can conclude that it will also have poor performance when additional predicates are added onto it. Thus, we do not need to consider such poor performing predicates when generating the formulae of greater lengths. This is also benefitted by the fact that there are far fewer formulae of length 4 than of length 5. Thus, incrementally searching through the formula space and building on information from smaller formula chains, we can truncate the search space drastically and can quickly evaluate the

performance of the formula. Thus, in order to generate a formula of length 5, we first generate formulae of length 2, identify good formulae and use them to create formulae of length 3. This is repeated with formulae of length 3, 4 and then finally a formula of length 5 is generated.

3.2.3 Results

In comparison to the ground truth constraints, our dynamic programming algorithm can reconstruct the true constraints with a precision of 0.715, a recall of 0.882 and a F-Score of 0.789. We evaluate the constraints synthesized by the constraint solver by synthesizing 1500 design points and running them through the simulation. Once the generated designs are simulated, we compare the output of the constraint synthesis program (i.e., is feasible or not) against the actual output of the simulation. Based on this comparison, we observed that the synthesized constraint has an accuracy of 81.4%. The top generated results are presented in Figure 11 and Figure 12.

Learnt Constraint	Hit Count	Is Ground Truth
isConnectedTo(GENERATOR, MOTOR)	130	1
isConnectedTo(GEARBOX, MOTOR)	99	1
isConnectedTo(ICE, TORQUE_COUPLER)	68	1
isConnectedTo(FRONTAXLE_INDIV, TORQUE_COUPLER)	62	1
isConnectedTo(GENERATOR, TORQUE_COUPLER)	57	0
isConnectedTo(ICE, MOTOR)	54	1
isConnectedTo(REARAXLE_INDIV, TORQUE_COUPLER)	54	1
isConnectedTo(GEARBOX, GENERATOR)	41	0
isConnectedTo(FRONTAXLE_INDIV, GEARBOX)	34	1
isConnectedTo(GEARBOX, REARAXLE_INDIV)	26	1
isConnectedTo(GENERATOR, REARAXLE_INDIV)	21	0
isConnectedTo(FRONTAXLE_INDIV, GENERATOR)	18	0
isConnectedTo(FRONTAXLE_INDIV, REARAXLE_INDIV)	18	1
isConnectedTo(FRONTAXLE_INDIV, ICE)	15	1
isConnectedTo(ICE, REARAXLE_INDIV)	14	1
isConnectedTo(FRONTAXLE_INDIV, REARAXLE_DIFF)	10	1
isConnectedTo(FRONTAXLE_DIFF, REARAXLE_INDIV)	10	1
isConnectedTo(ICE, REARAXLE_DIFF)	9	1
isConnectedTo(FRONTAXLE_DIFF, ICE)	8	1
isConnectedTo(FRONTAXLE_DIFF, GENERATOR)	8	0
isConnectedTo(GENERATOR, REARAXLE_DIFF)	7	1

Figure 11: Comparison of the generated results with the ground truth

[illegible]

Figure 12: Pair-wise comparison of generated constraints against the ground truth

Thus, we can conclude that,

- The demonstrator illustrates that dynamic programming is a reasonable approach to learning the design space constraints from "historic" data.
- The ability to formulate the identified constraints as first order logics provides a human understandable representation for constraints which can be easily verified by experts.
- Since the approach favors grounded predicates that are more frequent (and the ones that are more accurate), in constraint chains that have multiple predicates, one or more of the predicates favors the more frequently occurring relationship (although it may be a valid relationship), while the remaining predicates seem to discover the constraint that makes the design invalid.

Both restrictive and prescriptive constraints can be learnt by the algorithm by simply changing the evaluation criterion from subgraph search from bad to good designs.

3.3 Design Reuse through Functional Abstractions

3.3.1 Background

Designers, during the process of engineering design, often create designs reusing knowledge, information, and elements from other existing designs. This process of reuse of knowledge or design elements saves the designer several hours of effort as it enables the transfer of past work into the current design effort. This is accomplished by identifying elements from the existing designs as a set of functions that must be performed by the new design. For example, if the design is tasked with producing lift to carry an aircraft, the designer may reuse a lifting surface such as a wing from an existing design or alternately consider a propeller and motor combination to produce the same effect. Although in this context of reuse of design elements, it is critical for a designer to consciously make a judgement as to what elements of the current design can potentially be reused in a future design and identify these a priori. While such practices certainly aid the design process, it places an unnecessary burden on the designer during the design time. Furthermore, often, the possible uses for design elements may be unknown due to design requirements that were not considered at the time of the design. This problem of design reuse finds applications in several fields of engineering ranging from system design for complex products (e.g., aircrafts, gas turbines, etc.) to molecular and chemical design for functionality to detailed design of integrated circuits. The following passages introduces a subset of application fields and the relevance of the reuse of design elements.

Electronic integrated circuits are widely used in manufacturing industry like amplifiers, logic units, counters, calculators, temperature sensors, and radio receivers, which can be seen everywhere in our lives. However, the design of electronic integrated circuits is a challenging problem in design area because of the large design space, complex performance tradeoffs and fast technological advancements. Given some specific design requirements, designers need to analyze the topology and derive equations for the performance metrics. While there are many mature very large-scale integration (VLSI) CAD tools in the market to help forge the digital designs, huge human efforts are still required for analog designs, even for experienced human experts. It is demanding to have some learning-based design assistance tools.

A complex product such as an aircraft or gas turbine often has several components each performing one or more function. For example, the tail of an aircraft not only provides stability to the aircraft

but also enables certain aspects of the aircraft's control. But the variety of possibilities in the architectural design space of these complex systems is extremely vast such that exploration of all possibilities is time consuming. In such cases, there is a benefit to having a system that can recommend possible architectural choices based on the set of design requirements in terms of the design functions that are to be performed by the complex system.

Molecular design relies on constructing a graph that represents the molecular / chemical which meets a set of design criteria. For example, consider the task of identifying a drug that functions as a pain reliever, a fever reducer, a congestion reliever and provides reliefs against numbness. Given that the number of combinations of the molecules in the construction of the drug are almost infinite, a design system should be capable of providing guidance to the designer about how the different functions can be satisfied by analyzing an existing database of drugs and the provided functions. This system can potentially yield de novo molecular design that combine existing molecules in previously unprecedented manners.

While the above three examples are meant to illustrate the need for reuse of design elements, they are by no means comprehensive, and several other applications can also benefit from the reuse of design components and capabilities.

Thus, we develop a system to reuse knowledge from existing successful designs, analyze them and store the useful information in the structured knowledge base. The system can easily search through the knowledge base and recommend the key topology to achieve the desired design functions. This will quickly inspire the designer to create novel designs and thus accelerate the design process.

The design reuse system enables and addresses the problem of the high human labor cost in creating new designs – circuits, products, and molecules – by introducing an AI learning and knowledge base. The design system has the following modules:

1. A graph encoder module that encodes the design topology from raw data storage into a mathematic graph representation for later learning process and storage in knowledge base.
2. A function encoder module that enables the decomposition of the design into different labelled regions with each part achieving some specific set of functions.
3. A design knowledge system module that stores all the graph representation and function representation of the designs in the knowledge base. It also contains the search engine to take queries from the designers and retrieve the corresponding circuits.

A user interface module that enables the designer-machine interactions. It takes queries from the designer and sends to the design knowledge system. It also provides the visualization and reasoning of the retrieved designs elements.

3.3.2 Implementation

We develop a system illustrated in Figure 13, that leverages existing design data to build a function reuse system which stores the information of the structures, parameters, and functionalities of designs. It is intended to identify the parts of the design that achieves the functions and recommend it to the user. The system consists of a data storage module, two data encoder modules, a knowledge base module and a user interface module. In the considered system, it is assumed that the original data is already generated and stored in the data storage in the format that is easy to operate with such as netlist for circuits, SMILE for molecules, or architectural graphs for system designs. The two modules of the encoders are to transform the raw data into the understandable

representations and then feed into the organized design knowledge system module. The design knowledge system contains two sub modules, one is the knowledge graph base module, the other is the query engine module. The knowledge graph base module stores the organized information of the design, and the query engine module understands the functional requirements sent from the user interface, retrieves the relevant information from knowledge graph base, and sends back to the user interface. The user interface module enables the user-machine interaction. We illustrate the key modules in detail as follows:

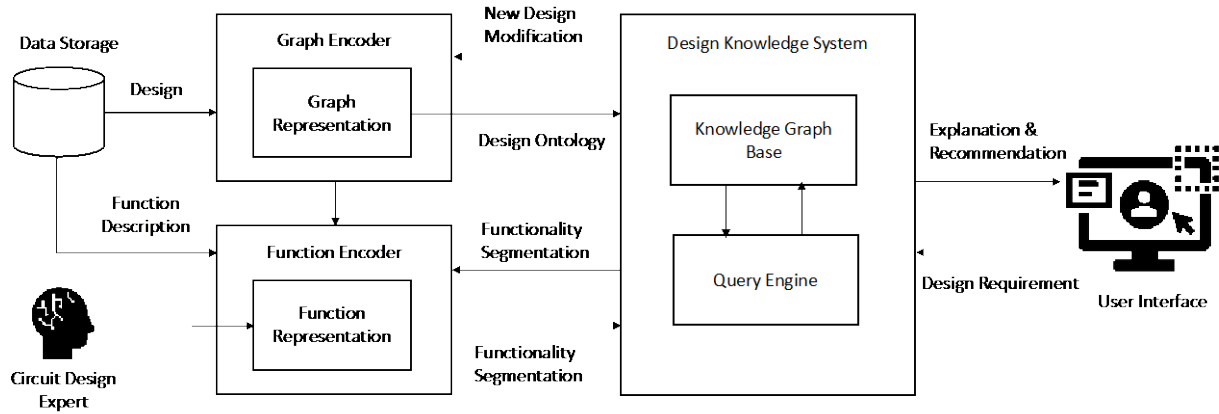


Figure 13: A conceptual overview of design functional reuse system

Graph Encoder: Figure 14 provides a conceptual illustration of how the encoding process is carried out. As an example, consider a netlist of the circuit from data storage module could be naturally encoded into a directed graph, where devices and intellectual property (IP) pins are represented as graph nodes and connections between devices at the graph edges [11]. The left part of figure displays the netlist representation of the circuit, and the right part of the figure is the transformed graph representation of the circuit. Similarly, Figure 15 illustrates the SMILE and the corresponding graph representation of a molecule. In addition, the graph representations are further transformed in the mathematical form of an adjacency matrix, a node feature matrix and edge feature matrix for the further modeling in function encoder module.

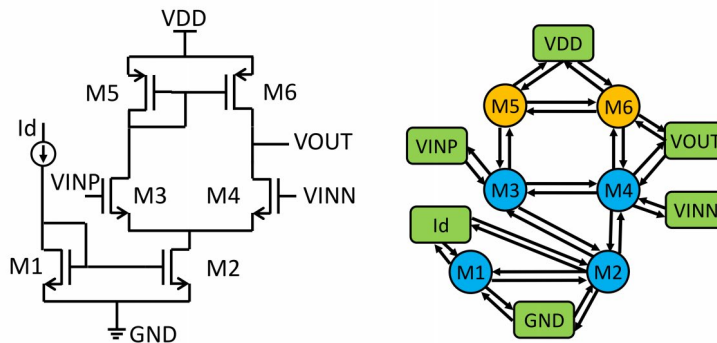


Figure 14: A conceptual illustration of the graph encoder

C12=C3C4=C5C6=C1C7=C8C9=C1C%10=C%11C(=C29)C3=C2C3=
 C4C4=C5C5=C9C6=C7C6=C7C8=C1C1=C8C%10=C%10C%11=C2
 C2=C3C3=C4C4=C5C5=C%11C%12=C(C6=C95)C7=C1C1=C%12C
 5=C%11C4=C3C3=C5C(=C81)C%10=C23

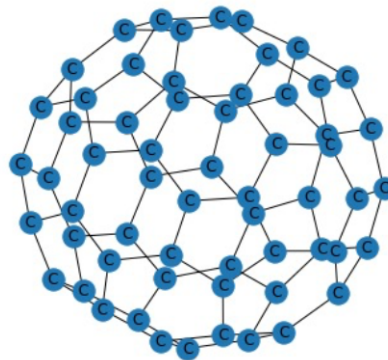


Figure 15: An illustration of the conversion of a SMILE to an undirected graph

Function Encoder: The function encoder identifies the key sub graph of the graph representation of the design which are most responsible for the desired functionality. We propose two ways to solve this. The first leverages domain expertise and the other one is to use the AI to automatically capture the important sub graph in the graph representation for specific functionality. The basic concept is illustrated in Figure 16. In the AI module, different types of the functionality will be first identified from the functional description, then the trained neural networks will quickly identify the corresponding sub parts of the circuit most likely contributing to the function. The neural network is trained in a graph classification framework. The graph classification network takes the mathematical graph representation of the design as input and applies Graph Attention Networks [12] to propagate local node and edge feature information through the graph given the structure of the graph by adjacency matrix. The graph attention networks are followed by a pooling layer to coarsen a graph into sub-graphs so that the node representations on coarsened graphs represent higher graph-level representations. A readout layer summarizes the final graph representation by taking the sum/mean of hidden representations of sub-graphs. A feed forward module takes to the results from the readout layers to predict its functional classes. The key function identifier is the Graph Attention Networks which adopt attention function to update edge weight between two connected nodes given their current node inputs. We select the edges with highest value of the attention weights to demonstrate the sub graphs achieving the function. Therefore, we can decompose the design into different key regions accomplishing different functions, which is our functional representation output of this module.

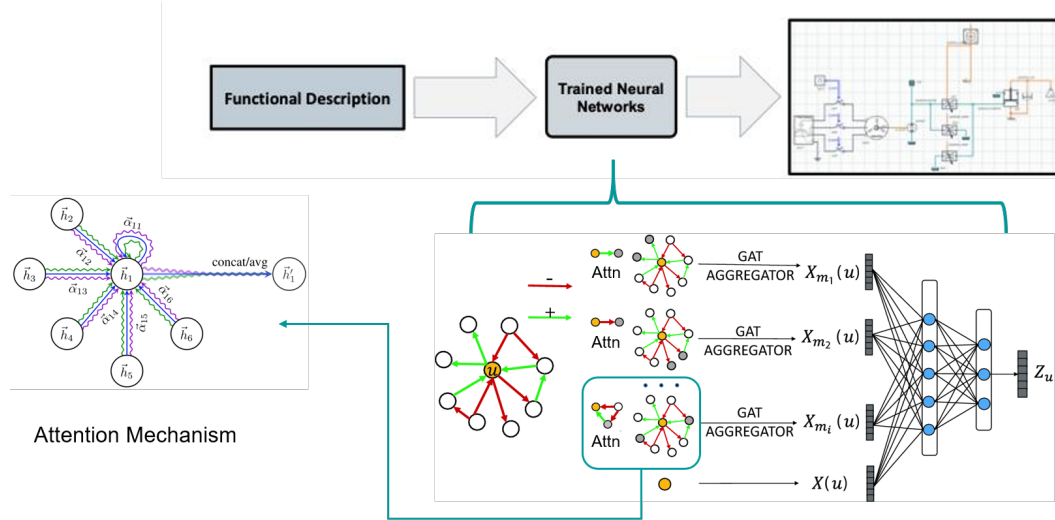


Figure 16. The concept of the AI modeling of function encoder.

Design Knowledge System: This module contains two sub modules, knowledge base and the query engine. The knowledge base stores the existing cleaned design in both graph representation and function presentation in graph-structured topology. It links descriptions, representation, and group information of the design. The query engine is to take the input from the user interface, search and retrieve the desired information.

3.3.3 Design Abstraction

3.3.3.1 Background

Modern software engineers have a vast array of powerful program libraries at their disposal. For tasks as diverse as building a reactive webpage, defining a neural network, simulating a multi-body physical system, or rendering high-fidelity graphical visualizations, these program libraries make it possible to construct complex systems quickly by invoking a rich range of modular program components. Automatic program synthesis systems, too, often leverage well-designed domain specific languages (DSLs) for domains that range from string editing to planning. For both human and automatic software engineering systems alike, well-designed libraries make it possible to efficiently write code that does what users want. One universal and organic way to construct a software library is to develop abstractions over existing code. Consider, for instance, the straight-line programs at the bottom of Figure 17(A), which use a generic set of base pixel-matrix strokes and transformation primitives to render technical drawings of hardware components. Faced with this verbose and clunky DSL, an experienced software engineer might start by defining additional functions that abstract out common useful functionality, like one for drawing regular polygons of arbitrary sizes. Here, our goal is to learn a richly augmented library of abstractions like these, tailored to any arbitrary domain, from a corpus of programs written using more generic initial primitives. We present Stitch, Figure 17(B), a fast and practical corpus-driven library learning system that discovers library abstractions from complex corpora comprising hundreds to thousands of programs in a matter of seconds. Stitch learns libraries that jointly minimize the size of the library and the length of programs written using the library. This joint compression criterion offers a principled way to capture our desiderata: the system should learn abstractions that capture the most common functionality across the domain, without otherwise bloating the library overall.

Prior library learning systems have made promising advances towards discovering common functionality from a corpus of training programs. These systems, however, take a deductive approach to library learning that is difficult to scale to larger datasets of longer and deeper input programs. These deductive systems attempt to refactor existing programs to expose shared structure and must represent and evaluate a large space of proposed refactoring to identify common functionality across the corpus. Prior approaches to this challenge combine a dynamic bottom-up approach to refactoring with version spaces to efficiently search over the refactored programs. However, these deductive approaches face daunting memory and search requirements as the training dataset scales in size and complexity.

3.3.4 Methodology

Stitch takes an alternate approach designed to discover useful libraries quickly from large-scale datasets. Our core contribution is a corpus-driven top-down synthesis algorithm, shown in Figure 17(B), that constructs abstractions directly from the initial primitives. Top-down synthesis is generally used in the inductive synthesis setting to search for programs which satisfy a specification of initial program behavior. Here, we instead use top-down synthesis to build higher-level library functions that capture common functionality across the training corpus, using the initial primitives in which the programs are expressed. Our core insight is that this search can be bounded much more tightly than in the usual inductive synthesis setting. We use syntactic pattern matching between a partially constructed abstraction and a corpus to heavily prune the search and precisely guide it towards abstractions that capture the largest and most frequently shared tree structure across the corpus.

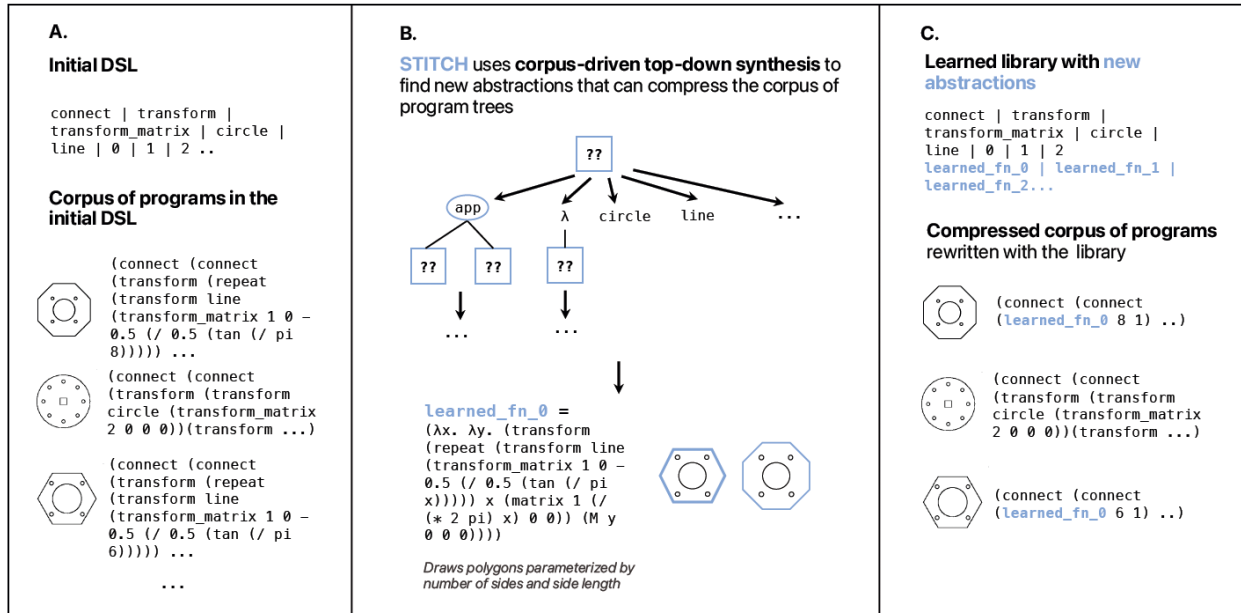


Figure 17: (A) Given an initial DSL of lower-level primitives and a corpus of programs written in the initial DSL, (B) Stitch uses a fast and memory efficient corpus-driven top-down synthesis algorithm to construct program abstractions which maximally compress shared structure across the corpus of program trees. (C) Stitch returns a learned library that augments the initial DSL with discovered function abstractions, and automatically rewrites the initial corpus using the learned library functions.

Stitch offers a prescriptive solution to the problem of defining a robust and useful abstraction library for a specific DSL. In particular, it provides a principled answer to the question, *What abstractions should be included in a good library?*

Consider a naive approach to library learning that simply enumerates all possible abstractions in the DSL. While the resulting library will, by construction, contain every function we might ever need, this “maximalist” approach is not especially useful: a downstream user will likely have a hard time picking out which functions to use from the vast, tangled set of overlapping abstractions. Instead, we would like to compute the utility of an abstraction and only include the maximally useful abstractions in our library.

While there are many different approaches one might consider, in Stitch, we adopt an information theoretic framing of utility that centers around compression. Specifically, it posits that the best abstraction is the one that maximally reduces the description length of a corpus of programs. Intuitively, we seek an abstraction that is large and frequently used in the corpus:

$$utility(A) = usages(A) * size(A) \quad [15]$$

To operationalize this compression-based utility metric, we need two things. First, we need a sufficiently large corpus of programs written in the DSL. Second, we need an efficient procedure for searching the corpus of programs for candidate abstractions. In this work, we focus on environments in which many example programs are available (as is often the case in real-world engineering). In these settings, the primary challenge is how to efficiently search through a large corpus of programs which are themselves potentially wide and deep. In the rest of this section, we build intuition for how Stitch tackles this search problem.

Top-down program synthesis: Our search algorithm builds on the top-down synthesis approach, which is popular in program synthesis but has primarily been applied in the inductive program synthesis setting where the goal is to synthesize a program which satisfies a set of provided input/examples. These methods construct program trees by iteratively refining partially completed programs that have unfinished holes until a complete program that meets the specification is produced. To deal with the challenge of guiding search there have been numerous extensions to top-down synthesis including type-directed synthesis which leverage type-based constraints on the holes, as well as probabilistic techniques to heuristically guide the top-down search.

Corpus-guided top-down search: Building on the prior work in top-down synthesis, our core insight is that we can synthesize abstractions themselves in a top-down fashion. As shown in Figure 18, our algorithm explores the space of abstractions through a top-down search over partial abstractions with holes. The optimal abstraction in this simple illustrative example is $\lambda\alpha.\lambda\beta.(+3(\star\alpha\beta))$. Since our top-down algorithm searches over abstraction bodies, our goal is to synthesize $(+3(\star\alpha\beta))$.

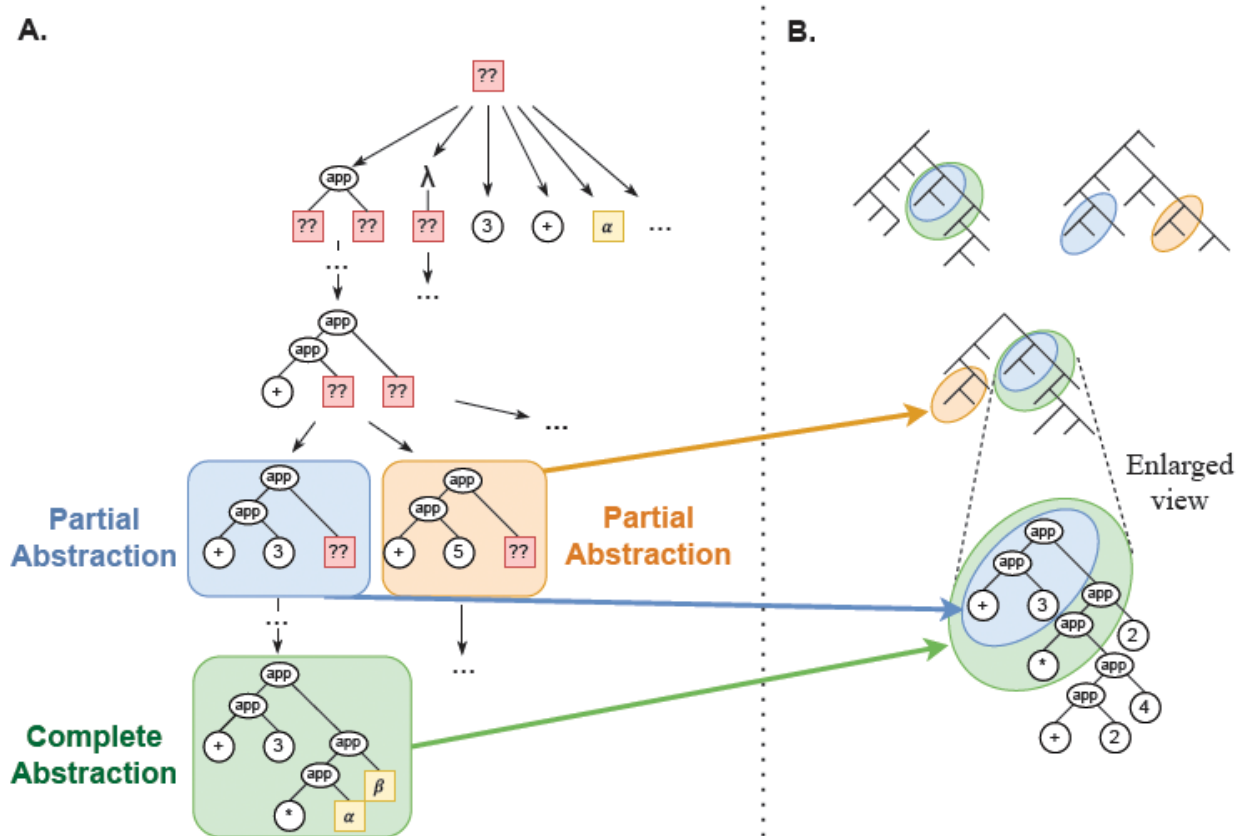


Figure 18: Corpus-guided top-down search. (A) Schematic of the top-down search procedure over partial abstractions (shown as function bodies). Partial abstractions can contain holes, or unfinished subtrees, denoted. Complete abstractions do not contain holes. (B) Partial and complete abstractions match at locations in the corpus. The blue partial abstraction can be expanded into the green complete abstraction, so the green abstraction matches in a subset of places that the blue abstraction does.

3.4 Design Representation as a Knowledge Graph

In this section, we present our investigation into the concept of using knowledge graphs as a representation for a design. This investigation presents a concept for the incorporation of domain knowledge across various scales into a representation of a design.

3.4.1 Background

As a team of designers execute a design process, considerable background knowledge is exercised in the creation of a design as the design process goes through the various steps of engineering. But unfortunately, the knowledge that was utilized in the creation of the design is lost once the design is generated. For example, the knowledge about characteristics of aircrafts such as characteristics of wings, propellers and their interactions play a significant role in the design of the vehicle. This knowledge is information that is often gathered through years of experience, typically documented in design textbooks and handbooks, and not explicitly captured in design workflows.

Thus, our concept investigation attempts to address the question, “*Can we expose the underlying knowledge contained in a design?*” We attempt to explicitly encode knowledge related to the design process into the design such that the creation of a new design can be based on existing ones.

As a motivating example, consider the case of conceptual design of an aircraft system. The problem is a highly multi-disciplinary one involving considerations of system architecture, e.g., vehicle propulsion choices, engine, tail section, and lifting device configurations, stability, and control, etc. A small portion of these considerations are illustrated in Figure 19.

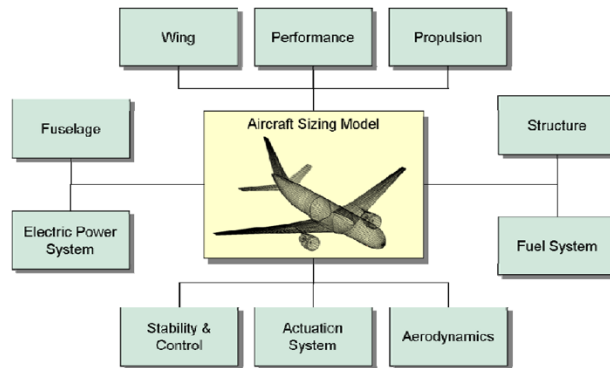


Figure 19: Multi-disciplinary considerations in the aircraft conceptual sizing and synthesis.

3.4.2 Methodology

Our technique employs a hierarchical representation of knowledge in a graphical format to capture a variety of knowledge, across different scales, associated with the design process. We present these knowledge elements across four scales:

- A design instance layer: This layer captures information about the instance of the design such as the choices made by the design team including the sizable parameters of the design (e.g., length of the wings), the configuration of the design (e.g., the type of the engine), the system architecture (e.g., three-engine powered by a tail mounted engine).
- The functional layer: This layer is meant to capture functional relationships between information in the design instance layer and functional characteristics of the design. For example, functions to be performed by components of the design such as a propeller that is meant to provide lift vs one that is meant to provide thrust can be encoded into this layer. This ensures that there is a clear mapping between the designer's intent and the designed product. This layer can further provide a mapping between the design and the functional requirements, if any, such that we are able to track the design characteristics and how they affect the design intent.
- The physics layer: We encode known physics relationships and the mapping between these relationships and information in the functional and design instance layers explicitly in the physics layer. This helps us understand how design variables have to be altered, either explicitly (e.g., in the case of how the coefficient of lift can affect the range of a vehicle) or implicitly (e.g., in the case of how the engine fuel consumption coefficients can alter the vehicle range, through the choice of the engine type).
- And finally, the experience layer: This layer is intended to capture information about the knowledge that is contained within the designer that is not explicitly represented through equations or relationships, but instead can be gained through years of experience. This

knowledge is typically encoded in design handbook as “rules-of-thumb” that provide soft guidance on selection of design characteristics.

The purpose of this representation is that the transformation from one design to another is as simple as modifying the design instance layer with new set of nodes, i.e., parameters, configurations, concepts, architecture, etc., while leveraging information from the abstracted representations of knowledge (i.e., functional, physics, and experience layers). The layers of the design hierarchy are illustrated in Figure 20.

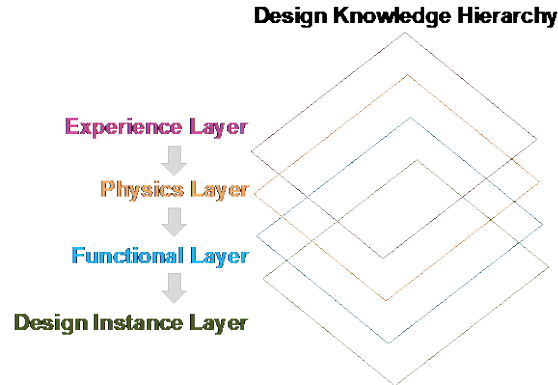


Figure 20: An illustration of the different layers of the design hierarchy encoded in a graphical format.

3.4.3 Results

We manually implement a proof-of-concept of the knowledge graph for the problem of conceptual design of a commercial aircraft. In the sizing and synthesis process, we capture information related to the design component choices, in this case, the choice of an engine from three alternatives. We identify design sizing parameters such as the lift-to-drag ratio, thrust-to-weight ratio, and the take-off weight of the aircraft, and identify how these factors can be utilized to optimize the range of the aircraft using the Breguet range equation. Further we capture experience-based knowledge such as the positive impact of the airfoil camber of the lift-to-drag performance of the aircraft while identifying how this information can adversely affect the thrust requirement. Several more information, e.g., the impact of density on the thrust and engine choice, the impact of altitude of operation on aerodynamic and propulsive behavior of the aircraft, and specific relationships given by the Breguet range equation are captured in the knowledge graph. Figure 21 illustrates the various layers collapsed onto a common plane, to simplify the visualization.

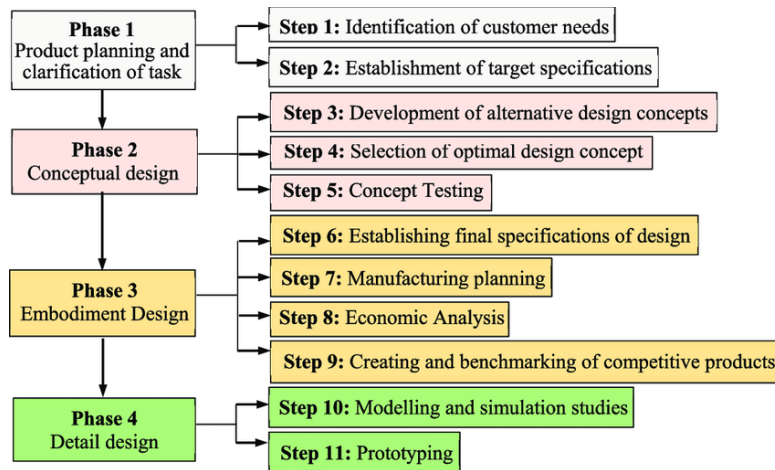


Figure 22: Engineering design stages from [13].

3.5.2 Methodology

We envision a master model that can be semi-automatically mapped to/from different models, level of details and different formats as needed for a particular step in the design process, including manufacturing. Although, multiple models in the design process cannot be eliminated, so the solution is to reduce the number of models. Most of the design steps should be able to use a “Master Model” without loss of information while others may use some 1-1 mapping. A collection of various models and mappings will be stored in an XML type document defined by a schema. Advantages of such schema include customizations via extensions and including additional meta-data regarding mappings. Models will be considered in two categories: geometry models and non-geometry models. Non-geometry models include function models, system models and others. Geometry-related models include B-Reps, meshes, voxels, etc. The non-geometry models need to have a mapping function manually defined with geometry models. All the non-geometry models will be included as collection and related mappings in the “master model.”

The draft schema:

```

#Design Master Model 0.1
{
  subPhases = [
    @planning.dmm@,
    @concept.dmm@,
    @preliminary.dmm@,
    @detail.dmm@
  ]
}

```

may be used as a starting point for detailing out the schema for a Design Master Model. A high-level design master model can reference subPhases. The phases are taken from typical engineering design phases shown in Figure 23.

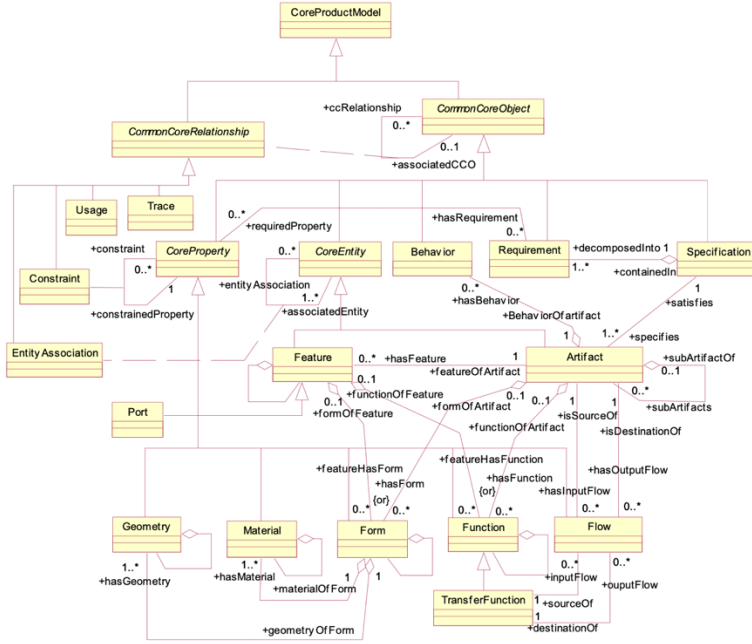


Figure 23: Core Product Model 2.0 [14, 15]

In the first phase (Planning.dmm), non-geometry documents and models will be referenced with meta-data. In the conceptual design phase, a mix of documents, sketches, models, 1-dimensional (1D) and 2-dimensional (2D) models including function and system models will be referenced. In the preliminary design phase, 2D models, 3D models including simulation models will be referenced. These phases of design master model will include mapping functions. Additionally, the high-level design master model file must reference the function and parameters that can be relied on in mapping models within a phase and across phases. We can rely on Core Product Model [14, 15], a semantic model built at the National Institute of Standards and Technology (NIST) to build product schema to encode the details in the Design Master Model and/or individual phases. Extensions of this model are also possible [16]. Regarding geometry models, there are two steps that we focus on, Design and Simulation. In design (preliminary and detail), CAD tools are used which rely on hybrid B-Rep and constructive solid geometry (CSG) models. In simulation, most methods used discretized models (mesh based, voxel based or others) of the design domain. We utilize the latest developments in simulation, with B-Splines [17] (including topology optimization using B-Splines [18, 19, 20]) to create a mapping between simulation and design representations with T-Splines [21]. A B-Spline $s(t)$ is a curve of order k defined as a linear combination of normalized basis functions of parameter $t (\in \mathbb{R})$ and knot vector ξ , weighted with control points x_i as shown in equation (16). The normalized basis functions are defined by Cox-de Boor recursion formulae as shown in equation (17). For all $k > 1$, the basis function follows equation (18). The knot vector ξ is any non-decreasing sequence of real numbers.

$$s(t) = \sum_{i=0}^n \mathbf{x}_i \beta_{i,\kappa}(t) \quad [16]$$

$$\beta_{i,1}(t) = \begin{cases} 1 & \text{if } \xi_i \leq t < \xi_{i+1} \\ 0 & \text{otherwise} \end{cases} \quad [17]$$

$$\beta_{i,\kappa}(t) = \frac{(t - \xi_i)}{\xi_{i+\kappa-1} - \xi_i} \beta_{i,\kappa-1}(t) + \frac{(\xi_{i+\kappa} - t)}{\xi_{i+\kappa} - \xi_{i+1}} \beta_{i+1,\kappa-1}(t) \quad [18]$$

B-Spline definition given above is dimension independent, implying that it can be used to define curves, surfaces, volumes etc. One aspect as the dimensionality increases, the control point grid is uniform. This implies that if there are 5 control points along one parameterization then there will be 5 control points along all other parameterizations based on the dimensionality.

Extension of B-Spline in the form of T-Splines was created to account for control points discontinuities. T-Splines are an extension of B-Splines that can have T-Junctions (unstructured) in control point grid [22]. As can be seen in Figure 24, the T-Spline control points are flexible in its arrangement than a B-Spline control point grid.

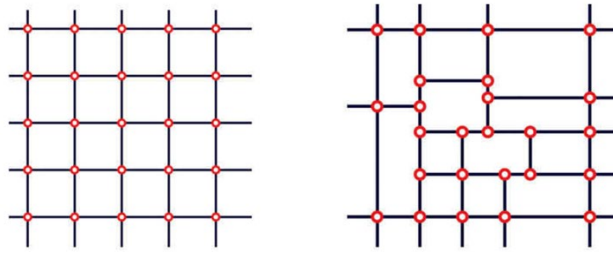


Figure 24 (a) B-Spline control point grid and (b) T-Spline control point mesh for surfaces

For a given T-mesh and degree p , let $A \subset \mathbb{Z}^2$ be the index set containing every α such that $\mathcal{S}_\alpha$ is an anchor. With local knot vectors ξ_α^1 and ξ_α^2 defined for every $\alpha \in A$, define functions B_α in the parameter space. For each $\alpha \in A$ choose a corresponding control point $P_\alpha \in \mathbb{R}^d$, where d is the chosen dimension for the physical space. For full generality, assume that each control point has an associated weight, w_α , and construct a set of T-spline blending functions as

$$R_\alpha(\xi) = \frac{w_\alpha B_\alpha(\xi)}{\sum_{\beta \in A} w_\beta B_\beta(\xi)}. \quad [19]$$

With such a definition, T-spline blending functions form a partition of unity. Finally, T-spline in physical space is given by.

$$S(\xi) = \sum_{\alpha \in A} P_\alpha R_\alpha(\xi). \quad [20]$$

Application of T-Splines to work directly with Iso Geometric Analysis (IGA) methods have been successfully demonstrated. As seen in Figure 25, beyond simulation, geometry and topology optimization can also be accomplished. This specifically removes the need of mesh generation for

finite element analysis (FEA) or postprocessing to get to CAD. It is well established that creating appropriate meshes that provide reliable results constitutes most of the simulation time for traditional models and physics. Eliminating the need for generating meshes reduces the time for product design. Truncated versions of B-Splines and T-Splines have also been proposed for multi-scale volumetric modeling and simulation [23].

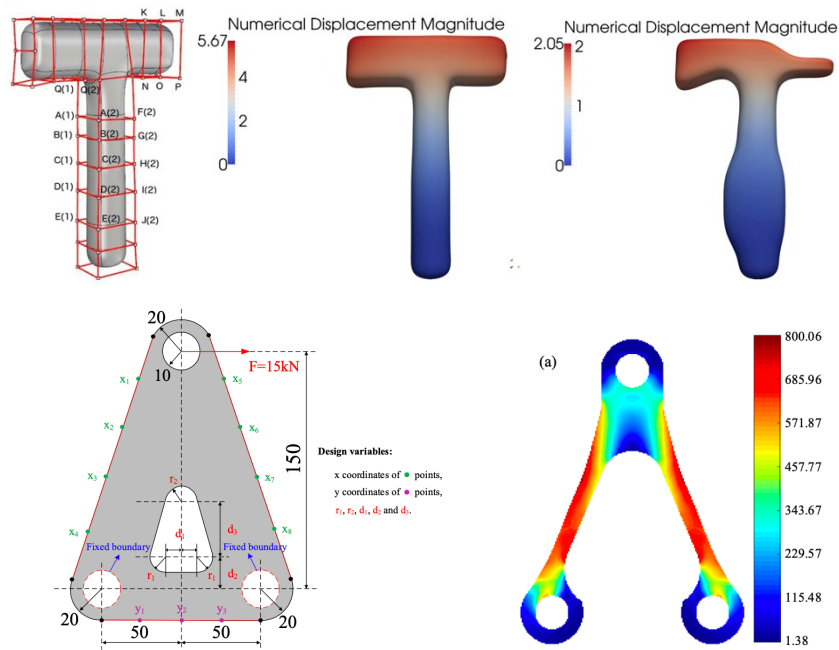


Figure 25: Examples of geometry and topology optimization using Isogeometric analysis with T-Splines.

3.5.3 Results

We explored the possibility of the use of a Design Master Model (DMM), an extensible schema to encapsulate and reference multiple different models, levels, and formats as part of product design. A high-level schema for referencing models from distinct phases, including non-geometry and geometry model is proposed for the DMM. Transformations that aid in developing mappings between models within/across a phase including meta-data can be included in the DMM. To encode the myriad details of conceptual and preliminary design, it is proposed to use Core Product Model 2.0. Finally, the latest developments in truncated T-Splines for volumetric representation are suggested (though not required) as part of DMM. This reduces the need to have separate geometry and simulation models, while supporting topology and geometry optimization. Furthermore, this reduces the work to manually map the results of simulation/optimization to geometry.

Although a candidate proposal for DMM has been developed as part of this effort, a lot remains to be done to realize the method. Future efforts should focus on building a framework and testbed to test the proposed Design Master Model. This would include completing the schema with design practitioners at various stages of design, manufacturing, and simulation. Followed by a suite of mapping tools that work with different software used in the design, manufacturing, and simulation.

A final task would be to test with a couple of academic design exercises followed by a demonstration with industrial engineering design using the method.

3.6 Design Composition with Conditional Variational Autoencoders

3.6.1 Introduction

This section covers the use of generative modeling to automatically complete partially specified designs. We demonstrate the use of generative graph models to synthesize designs given constraints that specify a certain set of components and / or a certain set of connections. Further, we demonstrate the ability to leverage a known design as a grounding constraint to synthesize other design(s) that are "mathematically" similar or different from the grounded design. Fundamentally, the concept leverages the graph representation of the designs which allows for an easy vectorization scheme. This is augmented by an exploration in the vectorized space for designs that are consistent with the specified constraints. The document considers the task of architectural design completion in which a partial design architecture is provided by the designer. We look at the problem of hybrid-electric vehicle powertrain design previously introduced.

3.6.2 Concept Details

The overall process is illustrated in Figure 26 where we begin with the generated data (i.e., design corpus) being used to train a latent space extraction model, such as a convolution neural network (CNN) variation autoencoder (VAE) (CNN-VAE). The latent space model is then used by an optimizer, which explores the learnt latent space (i.e., design manifold), to synthesize candidate designs that meet the partial design constraint specified by the designer. These constraints are then evaluated to generated objective value guiding the optimizer which makes the necessary modifications to the design parameters. Upon completion of this iterative synthesis process, the optimizer synthesizes a design that meets the specified constraints.

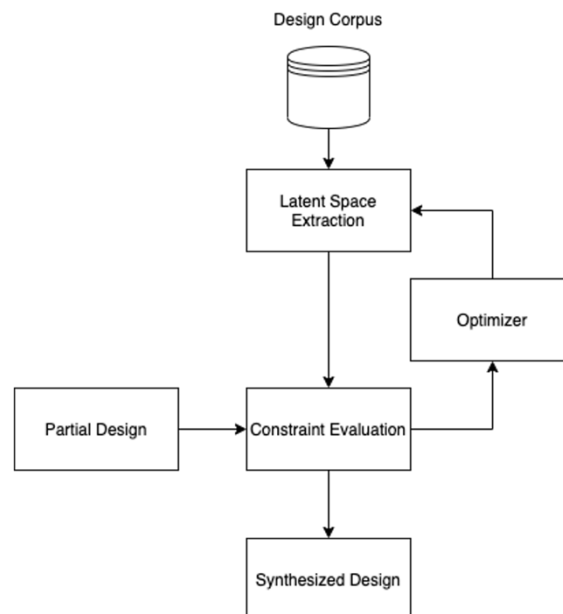


Figure 26: A high-level design completion workflow

Latent Space Modeling

Variational Autoencoders provide an unsupervised framework for learning complicated distributions. These models are built on top of standard approximators such as neural networks and can be leveraged for "generative" modeling wherein the approximators instead of learning deterministic transformations approximate the distributions represented by the data – $P(X)$ defined by data X . The primary concept in a variational autoencoder is that the vector X can be reconstructed by sampling from a latent vector z sampled from a standard normal distribution $N(0, I)$. Here the latent vector and the corresponding latent space can be learnt to model the distribution of the data such that a given datapoint X can be reconstructed accurately when sampled from the latent space parametrized by the weights of the approximator network θ .

Convolution neural networks are a popular choice among the various alternative approximators that are used to model spatially related data. These are primarily chosen for modeling images as they preserve the translative property of the image. We leverage this property of the convolution neural network by formulating the designs as a tensor of features and adjacencies. The process involved in training the CNN-VAE is illustrated in Figure 27. The process begins with the extraction of the feature matrix and the adjacency matrix from the design corpus. The feature matrix is a set of one-hot vectors that indicate the type of the component at a certain node location. The adjacency matrix is a binary matrix that indicates the connectivity of the graph, identifying how the different nodes are connected to each other. This compiled information is used to train the CNN-VAE model using a slightly modified loss formulation. The loss is a function given as:

$$\mathcal{L} = \lambda_1 * (\mathcal{KL}_{D_f} + BCE_f) + \lambda_2 * (\mathcal{KL}_{D_A} + BCE_A) \quad [21]$$

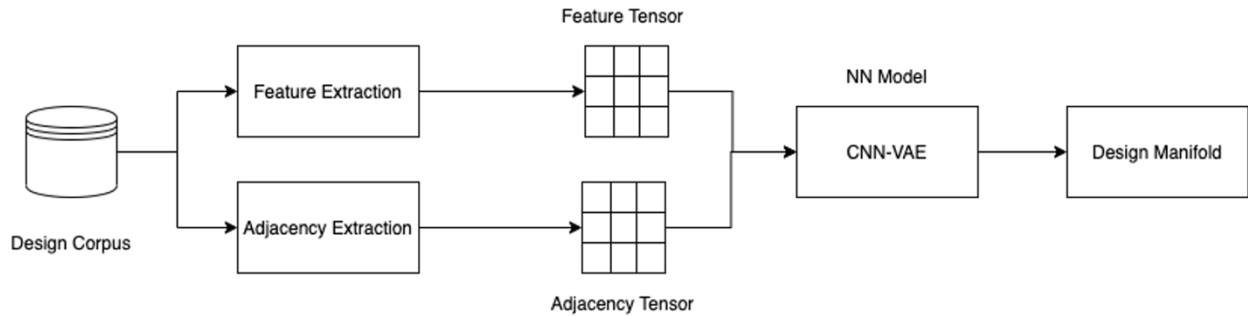


Figure 27: Process involved in training the CNN-CAE model

Figure 28 illustrates the network architecture of the CNN-VAE model. The model is comprised of two separate sets of CNN layers which embed and flatten both the feature matrix and the adjacency matrix. A concatenation layer combines the flattened feature and adjacency vectors and passes them through a set of fully connected layers to estimate the mean and standard deviation (log variance) of the latent space. This distribution is sampled from to yield the latent vector which then passes through a set of deconvolution layers to yield the reconstructed features and adjacency. These reconstructed matrices are compared against the input feature and adjacency matrices to yield the training loss which is then used to update the network weights end-to-end.

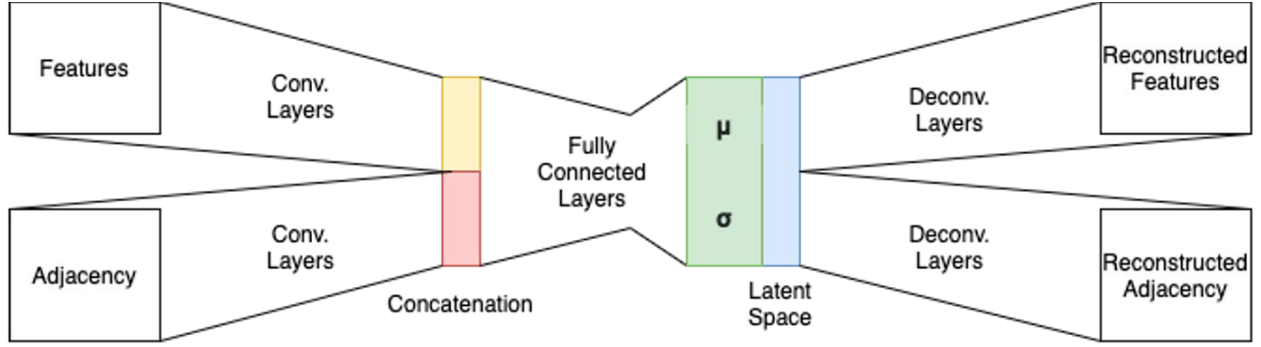


Figure 28: A conceptual illustration of the CNN-VAE network architecture

Automated Design Completion

We use Covariance Matrix Adaption Evolutionary Strategy (CMAES) as the optimizer to auto-complete a partial design. As the optimization strategy is well established, we do not describe the mathematical detail of the algorithm here, but instead focus on the formulation of the optimization problem. As a part of the optimization, we are given the following:

- A generative model $f(A, X|z; \theta)$
- A set of components that must be present in the design $\{c_i\}$, where $i \in \{1, n\}$
- A set of connections that must be present in the design $\{a_{ij}\} = \{0, 1\}$, where $i, j \in \{1, n\}$

With this setting, we can formulate the optimization problem as the minimization of the violation of constraints a set of constraints. These are:

- Each component can only belong to one type. This can be mathematically represented as $\sum_k x_{ik} = 1 \forall i \in \{1, n\}$
- All the requested components must be present in the design. This can be given as $x_{ic} = 1$ where c is the index of the component at location i .
- All the requested connections must be present in the design. This is given by, $A_{ij} = 1 \forall i, j \in \{a_{ij}\}$
- The components are not self-connected. This is given as $Tr(A) = 0$.

Thus, the overall objective function is given as:

$$f^* = \lambda_1 \sum_i \left(\sum_k x_{ik} - 1 \right)^2 + \lambda_2 \sum_{i^*} (x_{i^*k} - 1)^2 + \lambda_3 \sum_{i^*, j^*} (a_{i^*j^*} - 1)^2 + \lambda_4 Tr(A) \quad [19]$$

With the optimization formulation being the minimization of the violation of the constraints. Thus:

$$\min_x f^*$$

where, the feature matrix is defined as $x = f(A, X|z; \theta)$ and the adjacency matrix is given by $a = h(A, X|z; \gamma)$.

3.6.3 Results

The training loss associated with the CNN-VAE is illustrated in Figure 29.

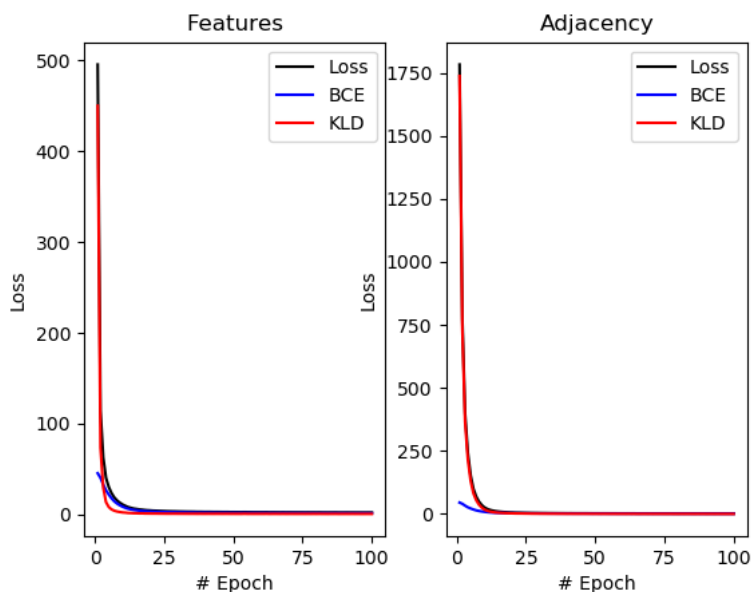


Figure 29: Training loss of the CNN-VAE algorithm

Upon evaluation of the algorithm on the HEV use-case, we observe that the optimizer is able to successfully reconstruct a complete design having 12 components and 53 connections, even when only 3 initial components and 2 connections are specified. The algorithm consistently demonstrates a completion rate of over 70% for the components and over 90% for the connections.

3.7 Design Completion with Graph Generative Models

3.7.1 Background

In the complex engineering design process, graphs are used to represent the state of a design at various stages of the design process. For example, graphs are the primary means of representation during the architectural design phase of complex product design, the design of a pharmaceutical drug, the design of an integrated circuit to name a few. In constructing these graph representations, the designer is often posed with questions of the type:

- Is the design complete?
- Will the resultant design have the desired performance?
- What are the modifications necessary for the design graph and in what sequence to achieve the desired goal?

The process of constructing these graph representations require considerable domain knowledge about both the functional, logical and physical behavior of the constituent nodes and edges. For example, in the system design use-case of the design of an unmanned aerial vehicle (UAV), the designer may seek to add propulsion devices to achieve a particular nature of thrust, as in the case of vertical take-off and landing in the vehicle where a designer would add vertical propellers and thrusters to achieve vertical thrust. Similarly, in the molecular design domain, to design a specific soap, the designer may synthesize an acidic molecule with the “COOH” carboxylic acid chain.

To address these issues, we develop a recommender system that assists designers in autocompleting the designs to achieve target performance characteristics. The recommender system builds on an existing database of designs and uses machine learning to automatically learn design patterns from the data. These patterns are then mapped to completion sequences to identify the shortest possible modification to the design to achieve the desired design goal.

3.7.2 Technical Details

The conditional graph completion problem is defined as follows:

*Consider the graph-label pair $(\mathbf{G}, \mathbf{y}) \in \{\mathbf{G}\} \times \mathbf{Y}$ satisfy a distribution $\mathcal{P}$.
Given a subgraph g_0 and a label y_0 , we want to sample full graphs $\mathbf{G}$ from $\mathbf{G} \sim \mathcal{P}((\mathbf{G}, \mathbf{y}) | g_0 \subseteq \mathbf{G}, \mathbf{y} = \mathbf{y}_0)$, meaning sampling from the conditional distribution where $\mathbf{G}$ has label y_0 and has g_0 as a subgraph.*

The challenges for conditional graph completion are multi-folds.

- Graphs have combinatorial structure with respect to node types, edge types, and adjacency. We need to learn good representation for graphs.
- Graphs have diverse distribution; we need a flexible generator to generate graphs in different structures and sizes.
- Graphs have various labels; we need a conditional generator that can generate graphs under arbitrary conditions.

To address these challenges, we develop a graph completion system, illustrated in Figure 30, that leverages an existing database of designs to learn completion patterns using a conditional graph completion algorithm. The graph completion algorithm is trained to automatically learn design rules (e.g., valence, aromaticity, component composition, electrical interference, etc.) from the observed data. Once trained, the graph completion algorithm is intended to provide design recommendations to the designers through a user interface that is integrated with the traditional design tools.

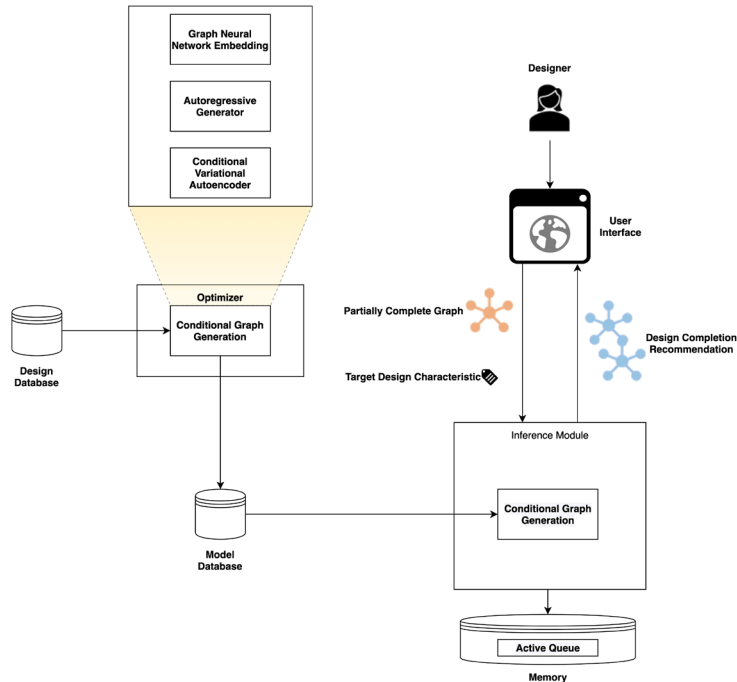


Figure 30: A conceptual overview of the design graph completion recommendation framework

The conditional graph generation framework that is used to provide the recommendation is comprised of three key components. The algorithm consists of:

- Graph neural network embedding module:** This module generates an embedded representation for each graph that is operated on by the completion framework. The algorithm implements the graph isomorphism network engine (GINE) with skip connection embedding technique. Specifically, it employs a one-hot embedding for the node and edge types and GINE information propagation is implementation at each step. This yields the node embedding for all the nodes in the graph which is then aggregated using a readout function to generate the graph embeddings. This process of embedding generation is illustrated in Figure 31(A).
- Auto-regressive generator:** Next, an auto-regressive generator module is used to generate completion sequences for the graph in a breadth-first search (BFS) order. A queue of active nodes is maintained in memory for the graph being completed at any point in the completion process. This active node indicates potential growth nodes to which new edges can be added. The auto-regressive generator determines from the existing graph, the head node from which the edge addition is carried out. At the same time, for the nodes in the active queue, we attempt to predict the node types. This is illustrated in Figure 31(B). If the node type predicted is “empty” or “null”, it is popped out of the active queue and the next node in the queue is used for analysis. At the same time, for the edge between the active node and the head node, an edge type is predicted (illustrated in Figure 31(C)). At each point along the generation, the completed graph, is supplied back to the user through

a user interface where the recommendations can be accepted or rejected. If the recommendations are accepted, the updated graph is passed back to the embedding algorithm to update the node embeddings and generate the next set of recommendations.

- Conditional Variational Auto-encoder:** Finally, we leverage a conditional variational autoencoder to synthesize graph completion patterns that are conditioned on a set of requirements. In Figure 31(B) and Figure 31(C), we notice that a vector of “z” is used as an input. These vector of values for “z” can be generated by training a conditional variational autoencoder that accepts as input a set of design characteristics “y” and generate a set of latent variables “z”. Each one of the generated “z” are then used to generate a new design recommendation for the user.

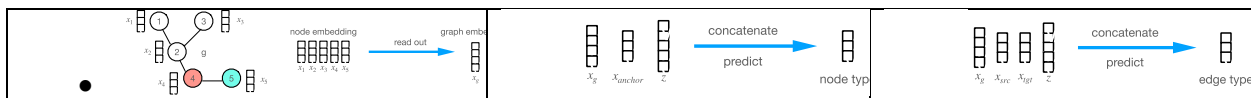


Figure 31: (A) Use of GINE for graph embedding; (B) Prediction of node type based on queued and head node embeddings; (C) Prediction of edge type based on the queued and head node embeddings

3.7.3 Results

Using the presented framework, we can achieve high reconstruction accuracies when experimenting on a molecule dataset. We demonstrate our method on QM9 dataset, which corresponds to the subset of all 133,885 species with up to nine heavy atoms (CONF) out of the GDB-17 chemical universe of 166 billion organic molecules. We filtered 9 graphs having only 1 or 2 nodes out. Since it is hard to learn a surrogate model to predict the labels provided in QM9, we use three simple labels in our demonstration: (#N, #O, isA), representing the number of nitrogen atoms, the number of oxygen atoms, and whether the molecule is aromatic.

We split the QM9 dataset with ratio 0.99 for training and 0.01 for test. We train our model on different size data sets to investigate how the performance relies on the size of training data. We train model on training data with ratio 0.001, 0.002, 0.004, 0.008, 0.016, 0.032, 0.064, 0.128. We recursively add graphs into the training data sets, hence smaller data set is always a subset of larger data set.

In evaluation, we sample subgraph and label (g_0, y_0) from $G = (V, E)$ in test data. The subgraph g_0 is sampled in a BFS order where the number of nodes is greater than or equal to 2 and less than or equal to $\max(2, |V| - 2)$. Based on these experiments, we observe that our algorithm has a high probability of generating designs that are both valid and have a high probability of meeting the design targets, as illustrated in Figure 32.

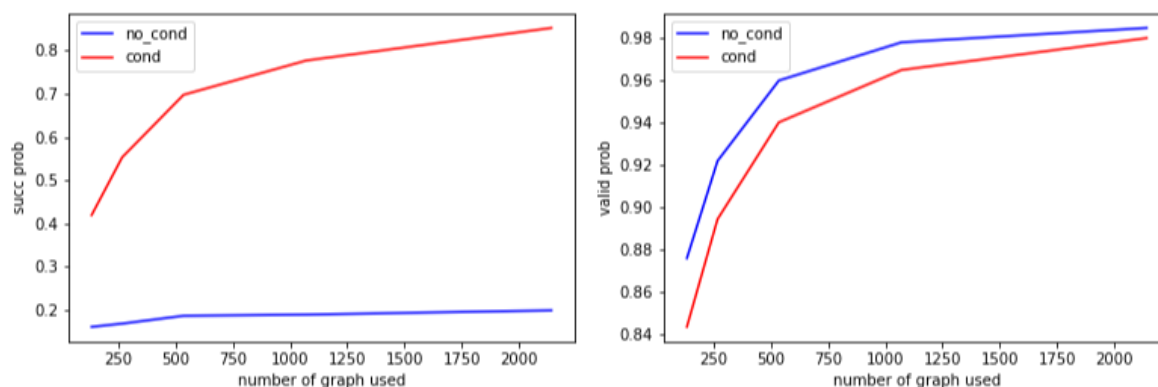


Figure 32: Plots of percentage of valid and successful graphs generated by the conditional generation framework on the molecule dataset.

Finally, we illustrate one example of the design completion patterns generated by the algorithm for the molecule dataset. Figure 33(A) illustrates the initial partial design graph supplied to the completion algorithm and Figure 33(B) illustrates the final completed graph generated. Figure 33(C) illustrates the completion sequence used to generate the design.

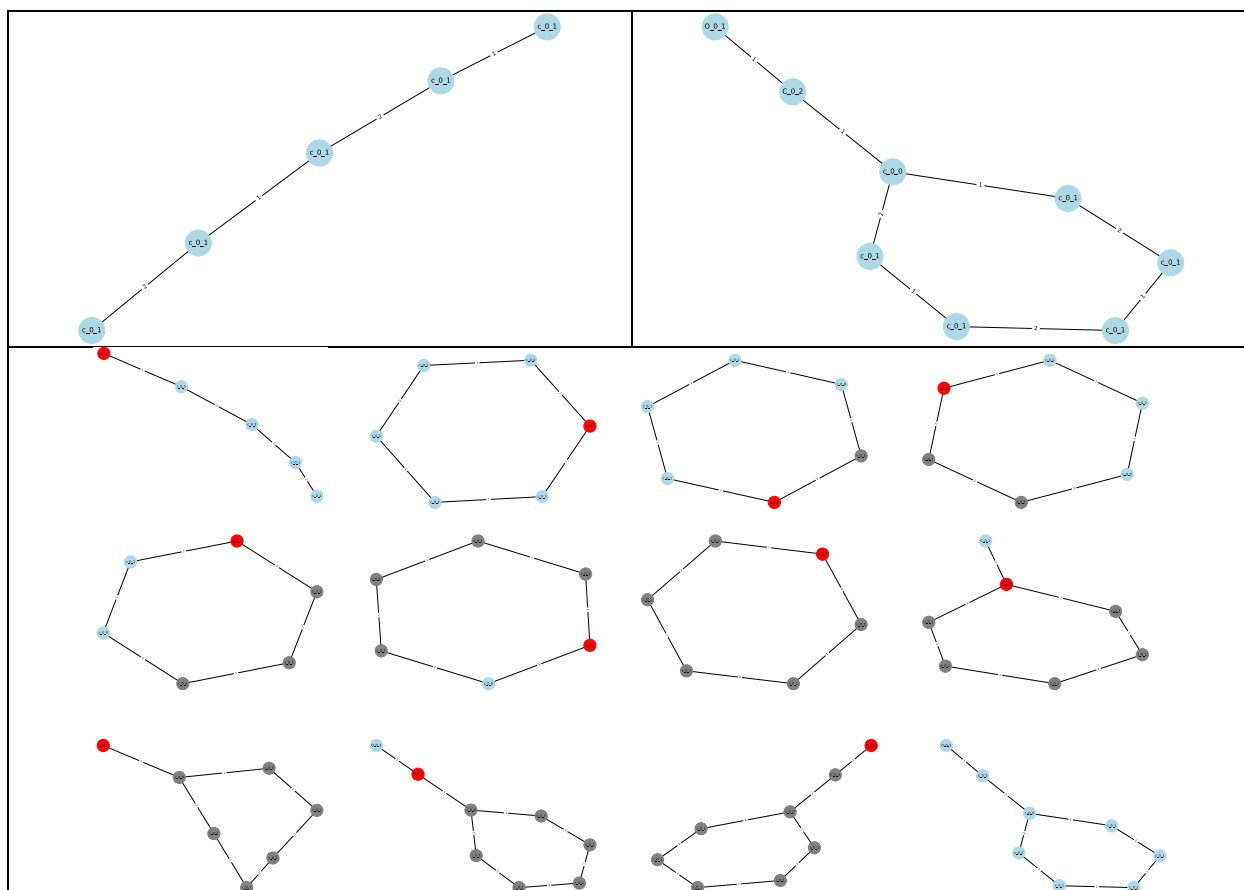


Figure 33: (A) Supplied initial partial molecule design; (B) Final completed molecule; (C) The completion sequence generated by the design graph recommendation algorithm.

3.8 Field Prediction with Reduced Order Modeling

3.8.1 Background

Exploring product design spaces with high-fidelity simulation on disciplines such as computational fluid dynamics (CFD), structural mechanics, heat transfer, among others is of great significance for many industries such as aerospace, electric cars, automotive, marine, etc. High-fidelity simulations are computationally expensive, and products are becoming increasingly complex needing multi-disciplinary design. This limits the amount of exploration that can be performed to achieve the stringent time-to-market requirements faced by several original equipment manufacturers (OEM)s.

Researchers have developed several approaches to computationally accelerate expensive simulations. Response surface and surrogate models predict the behavior of one or more response of the system as a function of its parametric representation [24, 25]. These methods are widely used in industries such as aerospace, marine, energy, among others [26]. As these methods model one or more global scalar responses (e.g., drag, lift, etc.), they rely on aggregate metrics computed from field responses (e.g., pressure and velocity distributions). While the aggregation process generates a handful of real-valued metrics that permit easy comparison of different designs, they lose a lot of information about the product performance. This additional lost information is often of great interest to the designer as it corresponds directly to the underlying physics. For example, the separation of flow on an airfoil can be directly visualized in a flow field and corrective actions can be taken to improve the airfoil performance when such information is available.

On the field modeling front, several different reduced order model (ROM) types have been developed to accomplish field modeling. Projection-based methods are a common intrusive ROM type which projects the full model equations onto the low-dimensional reduced space spanned by the bases [27]. Operator inference ROM are non-intrusive ROM type which infers the low-dimensional equations from trajectories and outputs of the full model [28]. Other non-intrusive ROM approaches use regression and interpolation methods to assess the ROM bases coefficients based on the model inputs; among this last group, Proper Orthogonal Decomposition with Interpolation (PODI) [29] has been widely used in many disciplines from aerodynamics to propulsion and reacting flows [30, 31, 32].

The most common ROM approaches restrict their modeling to changes in environmental conditions (e.g., boundary conditions) [28]. Changing geometries result in difficulties such as the underlying field responses are no longer identical (e.g., modifications to a CFD mesh) across multiple design instances and hence, the field snapshots extracted do not share the same dimensions and live on topologically different meshes. Some authors have extended the ROM approach to account for geometrical changes enabling design optimization / design space exploration [25, 26].

Solutions to these issues have been presented by several researchers to enable design space exploration and optimization with detailed but computationally efficient field modeling. Researchers have proposed an approach that substitutes traditional geometry and mesh generator steps in the computer-aided engineering (CAE) workflow with mesh morphing techniques that guarantee same dimensional snapshot spaces. For example, several authors have used Free From Deformation (FFD) as geometry and mesh generator to build ROMs which are applied to structural, aerodynamic and fluid dynamic problems [33, 24, 25]. Similar approaches have been applied to design in a naval industry as well [27]. However, this requires a modification of the corresponding workflow steps and tools which, if possible, maybe costly and time consuming.

3.8.2 Technical Details

This work proposes to augment standard CAE workflows with capability to account for geometric changes, and thus, enabling ROM building in the product design phase. The focus of the work is on black-box CAE workflows where designers cannot change steps and therefore the substitution of the geometry and mesh generation steps utilized by contemporary research works [34, 35, 36] is not feasible. The standard CAE workflow is complemented with an add-on workflow as shown in Figure 34. This add-on workflow takes as inputs the baseline design reference mesh and the results on a new geometry (from the black-box CAE workflow) in the design space.

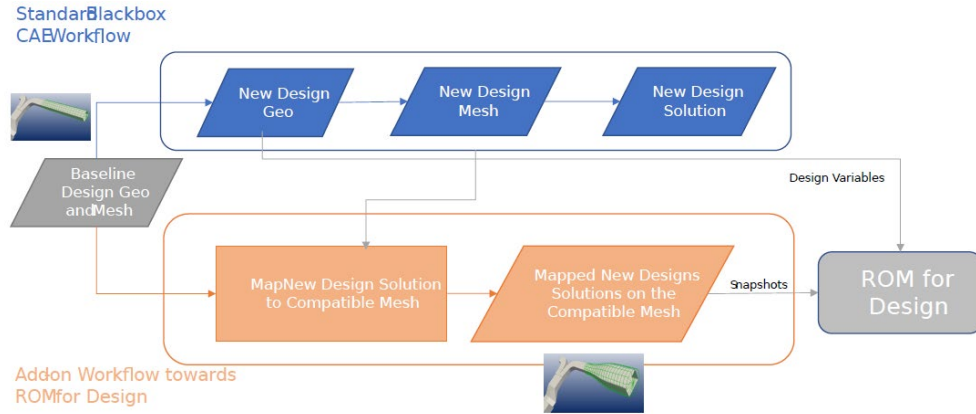


Figure 34. Add-on Workflow to Enable ROM for Design

This add-on workflow maps each new design CAE solution output by the black-box CAE workflow to a topologically unique snapshot space. This mapping action is shown as "Map New Designs Solutions to a Compatible Mesh" in Figure 34.

Once all the snapshots for the available geometry candidates in the design space are aligned into meshes with same dimension and morphology, a design-variable parametric ROM can be built; see block diagram called "ROM for Design" in Figure 34. Specifically, the proposed method uses PODI to construct the ROM with design capabilities. This method uses a proper orthogonal decomposition (POD) bases expansion to represent the field of interest $w_j(x, y, z)$ at a given design point j .

$$w_j(x, y, z, \mu_1, \dots, \mu_{N_\mu}) = \sum_k^{inf} \alpha_{j,k}(\mu_1^{(j)}, \dots, \mu_{N_\mu}^{(j)}) \Phi_k(x, y, z) \quad [22]$$

where Φ_k is the k -th base, $\alpha_{j,k}(\mu_1^{(j)}, \dots, \mu_{N_\mu}^{(j)})$ the expansion coefficient for the base k at the design point j , (x, y, z) the Cartesian space coordinates, and $\mu_l^{(j)}$ is the parameter l for the design j . Note that $\mu_l^{(j)} \mid l \in [1, \dots, N_\mu]$ where N_μ is the number of parameters in the space. From now on, the dependency on the Cartesian space will be implicitly assumed.

For practical reasons the expansion above is truncated to N_{bases} , an integer number, leading to a field of interest as

$$\mathbf{w}_j(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mu_1, \dots, \mu_{N_\mu}) = \sum_k^{N_{bases}} \alpha_{j,k}(\mu_1^{(j)}, \dots, \mu_{N_\mu}^{(j)}) \Phi_k(\mathbf{x}, \mathbf{y}, \mathbf{z}) \quad [23]$$

The N_{bases} -expansion coefficients, α_j , for each of the $j \in [0, \dots, N_{PODI}]$ training geometries are obtained by projecting each field of interest snapshot $\mathbf{w}_j$ on the bases Φ .

$$\alpha_j(\mu_1^{(j)}, \dots, \mu_{N_\mu}^{(j)}) = \Phi^T \mathbf{w}_j(\mu_1^{(j)}, \dots, \mu_{N_\mu}^{(j)}) \quad \text{for } j = 1, \dots, N_{PODI} \quad [24]$$

Finally, the PODI regressor, $regr$, which maps the j -th design parameters point, $[\mu_1^{(j)}, \dots, \mu_{N_\mu}^{(j)}]$, to the corresponding expansion coefficients $\alpha_{j,k}$ for $k = 1, \dots, N_{bases}$ is given by:

$$\alpha_{j,k} = regr(\mu_1^{(j)}, \dots, \mu_{N_\mu}^{(j)}) \quad \text{for } j = 1, \dots, N_{test} \quad k = 1, \dots, N_{bases} \quad [25]$$

Once the expansion coefficients are assessed, the field of interest is constructed according to Equation (7). The field model built with the method described in this section is referred as "ROM for Design" in the following sections of this work.

Mapping Design Instance Meshes to ROM-compatible Mesh:

The mapping method receives as inputs the Design Meshes and Solutions from the design space as inputs and entails two steps as shown in Figure 35. Mapping Method: Mesh Morphing and Snapshot Projections. Figure 35. First, a ROM-compatible mesh for the new design geometry is computed with a mesh morphing function. This first step takes the displacements of key location points on a new design geometry mesh with respect to the baseline design reference mesh; these displacements are used to morph the latter mesh into the former one which results into ROM-compatible mesh for the new design geometry, i.e., a mesh topologically identical to the reference design. pyGEM offers FFD and radial basis function (RBF) mesh morphing methods [37]. Several RBF basis functions are available in pyGEM: Gaussian spline, inverse multi-quadratic biharmonic spline, multi-quadratic biharmonic spline, thin plate spline, beckert wendland c2 basis, and polyharmonic spline. Additionally, each RBF basis function has configuration parameters, for example, the RBF radius which affects the shape of the basic functions.

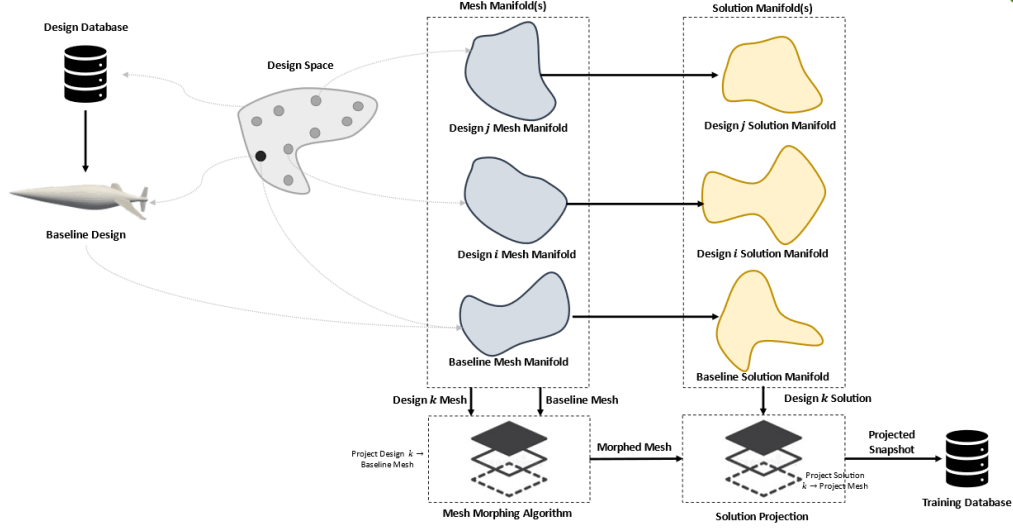


Figure 35. Mapping Method: Mesh Morphing and Snapshot Projections.

For each application considered, the best mesh morphing function is estimated by minimizing the mean squared error between morphed mesh and standard workflow mesh test key location points for new design geometries on the design space. This minimization problem leads to the selection of the most appropriate mesh morphing method and configuration parameter for the application. We observe that in both applications considered, RBF mesh morphing produces the best mesh alignments. Hence, the RBF method is applied to all the applications explored in this work. Once the ROM-compatible mesh is available, the second step for this mapping method projects the new design field response reported by the standard black-box CAE workflow into the ROM-compatible mesh producing aligned snapshots with the same dimensions and an evaluation on a similar topological mesh for the new design geometry; Scipy linear interpolation constructed by triangulating grid points with Qhull is the implementation for this second step [38]. This step is executed on all the new design geometries.

Performance metrics

Root mean squared errors (RMSE) are commonly used to measure algorithm performance. Since the method is tested against multiple unseen design points N_{test} , the mean value across all design test points is given by:

$$\overline{RMSE} = \frac{1}{N_{test}} \sum_j \sqrt{\frac{1}{N_{cells}} \sum_i \left(w_{j,GT}^{(i)} - w_{j,ROM}^{(i)} \right)^2} \quad [26]$$

where N_{cells} is the number of cells in the domain of interest; and $w_{j,ROM}^{(i)}$ and $w_{j,GT}^{(i)}$ are, respectively, the ROM and the ground-truth field of interest at cell i for the design point j . CFD or FEA simulations are used as the ground-truth.

The most important PODI settings are the number of bases N_{bases} used to represent the fields, and the number of training snapshots used to perform the PODI regression, N_{PODI} . Sensitivity studies on how the defined key performance indicators (KPIs) are influenced by these PODI settings could define settings applicability and method recommendations and limitations.

3.8.3 Results

Mechanical Design of a 3D Pressure Vessel

This first application studied is the design of cylindrical and capsule-shaped pressure vessel (PV) in an unmanned underwater vehicle (UUV). The field of interest for both design topologies is the von Mises stress.

The data pre-processing towards the ROM for Design model is as follows:

1. Design of Experiments (DoE) of up to 200 design points are performed.
2. 20 design points are set apart and used for testing the method $N_{test} = 20$
3. The standard CAE workflow is applied to the design points created by the DoE. This step generates the snapshots.
4. The training and testing snapshots are checked for lack of convergence or outlier values. Snapshots identified as such are removed from the sets.
5. The add-on workflow to enable ROM for Design, shown in Figure 34, is applied to all the remaining design points and corresponding snapshots from the DoEs.
6. The remaining training design points and their corresponding snapshots are used to assess the POD bases.
7. Several values of PODI number of snapshots, N_{PODI} , are chosen from the set immediately above with N_{PODI} ranging from [10 – 110] for the cylindrical topology and [10 – 50] for the capsule topology.
8. A varying number of bases N_{bases} is used for training the ROM proposed with ranges between [2 – 10] for both topologies.

The pressure vessel is designed for a depth rating of 4000 m. The pressure vessel at any depth encounters a uniform pressure load corresponding to the vehicle depth and must withstand buckling, fracture, and elastic failures. In our work, we only consider the case of elastic failure of the pressure vessel.

The pressure vessel has pressure loaded to the depth rating, i.e., 41.084 MPa, with a displacement constraint applied to the edge in contact with the fairing geometry. In addition, both self-weight and component weight loads are applied to the body for taking into consideration the pressure vessel weight, and the contained components. The pressure vessel is assigned a material of Titanium for the study which has the following mechanical properties:

Density	Young's Modulus	Poisson's Ratio	Yield Stress
4429kg/m ³	113.8 GPa	0.342	1070.0 MPa

The pressure vessels are meshed using an open-source 3D meshing software, Gmsh [39], and their structural response is computed using finite element analysis simulated in Calculix [40] while tracking their displacements, stresses, and strains.

Parametric variations of the pressure vessels to the shape and size, particularly the outer diameter, the wall thickness and the length of the pressure vessel, are used to generate data for the training and testing of the ROM for design model. The design parameters for the cylindrical and capsule topologies are varied by 5% of the baseline design's value.

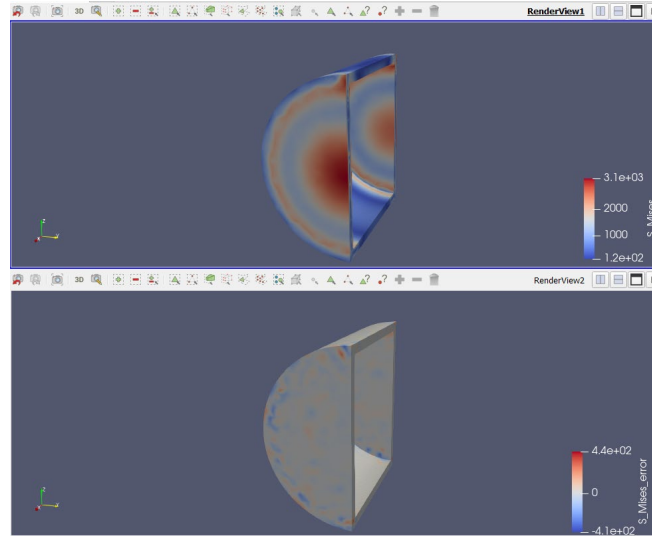


Figure 36. ROM for Design. Von Mises Stress MPa (above) and Error (below) on the Clipped Cylindrical PV at an Unseen/Test Design Point. Clipping Plane $y = 0$.

Figure 36 shows that the ROM for Design is capable of capturing, accurately, the von Mises stress field behavior for an unseen test design point. The below contour in Figure 36 shows the ROM for Design error; this contour shows that the max. and min. field errors are one order of magnitude lower than typical von Mises values and are restricted to very small areas of the domain; specifically, the points where the displacement constraints are applied. In the remaining part of the domain, the error in the estimation of the von Mises stress for ROM for design are two or more order of magnitudes lower than the actual stress values (i.e., error under 1%).

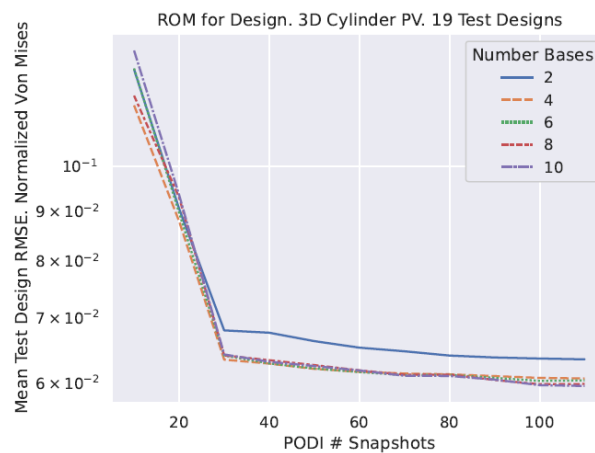


Figure 37. ROM for Design Accuracy. Relative Von Mises Stress (Nominal Value: 890.653 MPa)

Figure 37 shows the ROM for Design KPI against a varying number of N_{PODI} and N_{bases} . The mean RMSE decreases with the number of bases until saturation is reached somewhere between 6 and 10 bases for all studied snapshot number N_{PODI} ; this saturation is common in POD approaches. It is worth noticing that given the similar loading but geometry changes, just naively using results from other geometry in the same morphological mesh might provide a relative RMSE of around 15% which can be reduced to 6% by using the proposed algorithm. Additionally, snapshots with finer grids are expected to improve the accuracy of the POD bases, and thus, the ROM for Design accuracy. Regarding the behavior along the number of snapshots N_{PODI} , the mean RMSE decreases with increasing values of N_{PODI} with the rate of improvement to the RMSE decreasing at 30 snapshots. Finally, it is worth noticing that for the cylindrical PV one test point has convergence issues, therefore the number of samples in the test set is $N_{test} = 19$.

Hydrodynamic Design of a 3D Glider

This second application explored is the design of a 3D glider immersed in water. The baseline shape of the glider can be seen in Figure 38. OpenFoam [41] is used to simulate incompressible flow around this 3D glider geometry and drive the mesh and analysis workflow.

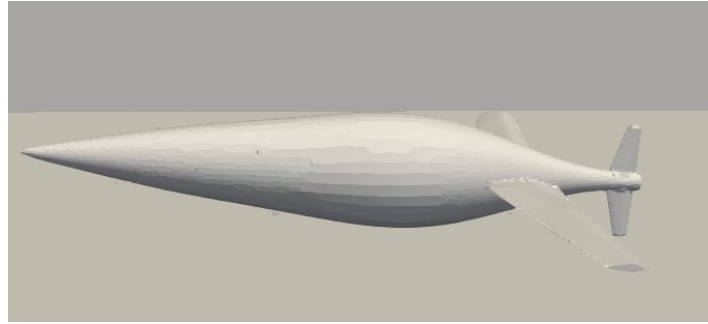


Figure 38. 3D Glider Geometry.

The explored design changes focus on two of the four airfoil sections which make up the glider wings. The wing sections are parametrized with class shape transformation (CST) functions [42]. Specifically, two airfoil section parameters are varied in sections 2 and 4. The design space $\mu = [\mu_{2,1}, \mu_{2,2}, \mu_{4,1}, \mu_{4,2}]$ variations are given by:

$$\mu_{sect,l} \in [0.95, 1.05] \mu_{sect,l}^{baseline} \quad [27]$$

where $\mu_{sect,l}$ is the l -th parameter value in the wing section $sect$, and $\mu^{baseline}$ the baseline design parameter value.

The data pre-processing towards the ROM for Design model generation is as follows:

1. DoEs of up to 103 design points are performed.
2. 20 design points are set apart and used for testing the method. $N_{test} = 20$.
3. The standard CAE workflow is applied to the design points created by the DoE. This step generates the snapshots.
4. The training and testing snapshots are checked for lack of convergence or outlier values. Those identified as such are removed from the sets.

5. The add-on workflow to enable ROM for Design, shown in Figure 34, is applied to all the remaining design points and corresponding snapshots from the DoEs.
6. The remaining training design points and their corresponding snapshots are used to assess the POD bases.
7. Several values of PODI number of snapshots, N_{PODI} , are chosen from the set immediately above the range between [5-65].
8. A varying number of bases N_{bases} is used for training the ROM proposed with the range between [3-13].

PODI regressors that assess the expansion coefficients are trained with N_{PODI} training snapshots from the corresponding design points. Once the PODI regressors are available, the expansion coefficients for each test design point are computed; subsequently the variable of interest for each design point is calculated. Finally, the field of interest for all the test designs is bench-marked against their ground-truth (Openfoam analysis) by the KPI equation.

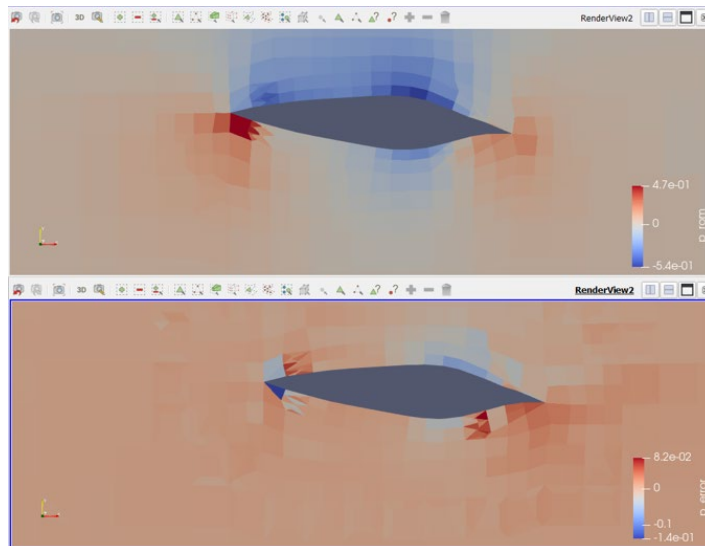


Figure 39. ROM for Design. Pressure-Density Ratio (above) and Error (below) at 70% Wingspan at an Unseen/Test Design Point Geometrv.

The field of interest is the pressure difference-density ratio $\frac{p-p_{inf}}{\rho}$ around the glider. Figure 39 shows that the ROM for Design is capable of capturing, accurately, the field behavior for an unseen test design point. The second contour in Figure 39 shows the ROM for Design error; this contours shows that the max. and min. field errors are one order of magnitude lower than typical field values and are restricted to very small areas of the domain; specifically, the leading and trailing edge of the airfoil sections. Given the coarse mesh used to obtain the CFD snapshots, even the "ground-truth" field values at these space locations obtained by CFD are to be treated with caution.

Figure 40 shows the ROM for Design KPI against a varying number of N_{PODI} and N_{bases} . The mean RMSE decreases with the number of bases until saturation is reached between 9 to 12 bases depending on the N_{PODI} ; this saturation is common in POD approaches; additionally, snapshots with finer grids are expected to push this limit up. Regarding the behavior along the number snapshots N_{PODI} setting, the mean RMSE decreases with increasing values of N_{PODI} ; also, the experiment shows some saturation for values close to 40 training snapshots.

Regarding time savings, the ROM for Design inference time is much shorter than the standard CAE design workflow: the former takes ~ 1 seconds whereas the latter takes ~ 600 seconds offering between two and three order of magnitudes speed-up (times when evaluating on an Intel(R) Core i7 CPU @2.7GHz).

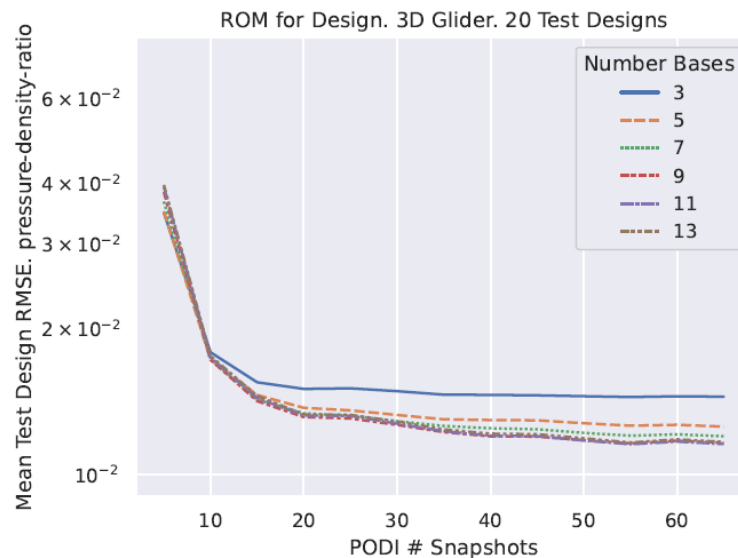


Figure 40. ROM for Design Accuracy. Relative Pressure-Density Ratio. 3D Glider.

3.8.4 Summary

The ROM for Design approach provides very fast field assessments for scenarios where there is no ability to edit the CAE workflow meshers and the CAE workflow evaluation is computationally expensive. In all the application cases the method results in field assessment three orders of magnitude or faster than the CAE standard workflow. The assessment shown in this work results in an average error of 6% or 1% with respect to the ground truth depending on the application case considered. These speed-up and accuracy numbers are applied only to the studied problems and employed meshes; other mesh sizes and problems will result in different accuracy and speed-up numbers. Method saturation appears in all three applications explored in this work; therefore, method users need to be aware about this behavior to benefit from it more efficiently.

The method has been tested against two different solvers: FEA and CFD. It is worth noting that the method is successful regardless of the solver used and their corresponding applications.

Use cases which fit within these scenarios are structural mechanics, heat transfer and computational fluid dynamics models with large number of cells and nodes which are very typical in the automotive, aerospace, energy, and many other industries. Also, scenarios without edit-access to the CAE workflow mesher is not too restrictive: neither black-box CAE workflow software close to the designers/analyst or design environments where analysts and designers are not very fluent with the mesher step of the workflow.

Fast first order evaluation on a parameter space makes the method very suitable for optimization and uncertainty quantification (UQ) where vast design spaces are usually explored. Even in design applications where accuracy is paramount the ROM for design can be used as a result initializer for a standard CAE workflow without native mesh morphing support.

The method benefit is its flexibility to plug and play different tools for each of the workflow steps. The workflow interfaces are snapshots, meshes and design parameters which can be easily aligned into the user-defined format regardless of the tool used for each of the CAE workflow step.

3.9 Inverse Dynamical Problems with Differentiable Solvers

3.9.1 Introduction

Operation is an important, costly phase of product and system life cycles. Operating products efficiently can not only save money but also extend their lifetime. Many use cases for efficient operation require fast resolution of Ordinary Differential Equations (ODEs) and Partial Differential Equations (PDEs).

Surrogate and reduced-order modelling have shortened the computation times with respect to traditional numerical methods such Finite Elements (FE) or Finite Volumes (FV). However, these methods do not provide gradients. These gradients and their use by optimizers are paramount for high-order real-time use cases such as inverse problems or calibration; in these use cases a sequence of differential equations (DE) solutions is assessed within an outer optimization loop, thus, inner loop quasi real-time DE assessment does not guarantee quasi real-time at the outer loop, especially for scenarios with many decision variables.

Differentiable time-marching scheme for ODEs enables the assessment of the gradient with respect to the ODE inputs. One example is the adjoint Sensitivity method [43] which can solve the ODE system if the dynamics are implemented in a computational graph; this method has been applied to traditional ML applications such as supervised learning and density estimation [44].

The above differentiable time-marching techniques have been combined with machine learning into a new field called Differentiable Physics. The idea of these approaches is to use discretized differentiable versions of physical models together with machine learning approaches such as neural networks to enhance the learning and training process [28]. The addition of the physics during learning leads to a reduction of the amount of data required. The poster examples for these approaches are inverse dynamical problems for ODEs where drastic resolution time reductions are achieved. However, these differentiable solvers have been mainly limited to ODE resolution given that differentiable PDE solvers are not commonly available. With internal Research and Development (R&D) Siemens has developed an approach to construct PDE solvers. With the DARPA funding, Siemens has 1) applied this approach to a new test problem and 2) extended this approach with operator inference ROMs [45].

3.9.2 Methodology

The PDE differentiable solver developed with internal R&D funding is composed of two main components: a) time marching solver equipped with the adjoint sensitivity method and b) a differentiable space discretizer which transforms PDEs into ODEs. The process to generate differentiable PDE solvers is as follows. First, a differentiable space discretizer is constructed to convert the PDE into a system of ODEs. The selection of the discretizer method depends on the actual ruling physics. Then, a discretizer transforming the PDE into an ODE is implemented as a computational graph which enables the leverage of its powerful automatic differentiation to achieve differential ability. Finally, the differentiable PDE solver is built by time marching with the adjoint sensitivity method on the computational graph representing the dynamics.

With DARPA R&D funding the above process has been extended in two aspects:

1. A finite difference discretizer for the 2D heat equation has been implemented in pytorch [46].
2. The operational inference [27] equations that reduced the discretized system of ODEs: it further accelerates the resolution of inverse dynamical problems.

These two new aspects will be tested against the unsteady 2D heat equation with two different boundary conditions in a unit square domain:

$$\frac{\partial T}{\partial t} = \nabla k(x, y) \cdot \nabla T(x, y) \quad x, y \in D = [0,1]^2; t \in [t_{ini}, t_f] \quad [28]$$

$$bc \ 1 \rightarrow T_l, T_b, T_t = 0; T_r = 1 \quad \partial D \quad [29]$$

$$bc \ 2 \rightarrow \text{linear edges with corner values } T_{bl} = 8, T_{br} = 6, T_{tr} = 7; T_{tl} = 5 \quad \partial D \quad [30]$$

The extensions will be tested for the inverse dynamical problems. These are problems where our dynamical system ruled PDEs or ODEs needs to meet a desired state at the final time, t_f ; this work focuses on the continuum problems/PDEs. Specifically, users set the desired state of the system or product to a target distribution $T^*(D, t = t_f)$ in t_f time units later; therefore, the initial conditions at t_{ini} needed to reach the desired state at t_f are sought. In practical terms, an optimizer will evaluate the differentiable PDE solver for a sequence of initial conditions with the goal to reduce the error between the solution state at t_f and the desired target one $T^*(D, t = t_f)$. Moreover, the gradient of this error between the PDE solution and the desired target is employed by the gradient-based optimizer to accelerate the resolution of the inverse dynamical problem.

It is worth mentioning the high dimension of the optimization problem it is being solved with this effort. Note that the variable to find during the optimization (inverse problem) is the PDE dependent variable field at the initial time, i.e., $T(D, t = t_{ini})$. The dimension of this variable depends on the space domain dimension and the discretization step. However typical grids used in engineering simulations for 2D and 3D can easily have dimensions over thousands of nodes and elements.

Other Implementation Details: The open-source *torchdiffeq* library [47] is employed in this work to use its ODE solvers equipped with the adjoint sensitivity method (both adaptive and fixed-step solvers). The gradient-based optimizer of choice is Stochastic Gradient Descent (SGD) from pytorch [46] with a learning rate of 5.

Metrics: The metric to be optimized will be the mean squared error (MSE), or loss, of the current PDE solution at t_f with respect to the desired state.

$$loss = \frac{1}{N} \sum_{i=1}^N \left(T^*(x_i, y_i, t = t_f) - \hat{T}(x_i, y_i, t = t_f) \right)^2 \quad [31]$$

where N is the number of discretized points in the domain; x_i and y_i the Cartesian locations for the discretized point i , the $\hat{\cdot}$ represents the PDE solution provided by the differentiable solver.

3.9.3 Results

The unsteady 2-D Heat Equation with boundary condition 1 is the first test for differentiable PDE solver applied to inverse dynamical problems. Note that the boundary conditions are Dirichlet zero value except in the right boundary where the value of the boundary condition is 1. The number of

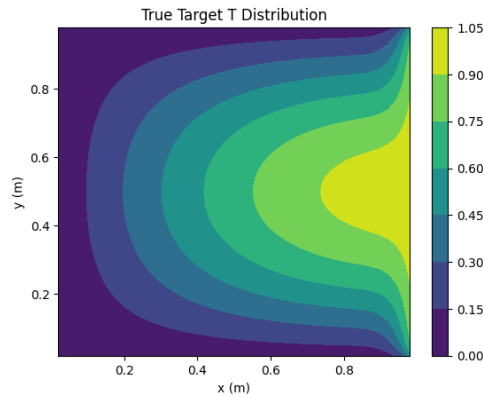


Figure 41. 2D Unsteady Heat Equation bc 1.
Target Distribution $T^*(D, t = t_f)$

nodal points in the domain D is $N = 2401$. The guess for the initial distribution is $T(D, t = t_{ini}) = 0$ at all points in D . Figure 41 shows the desired (or target) distribution at the final time $T^*(D, t = t_f)$. The gradient-based optimizer is run 20 times (same as the number of forward differentiable PDE resolutions).

Figure 42 shows the final distribution $\hat{T}(x_i, y_i, t = t_f)$ calculated by the gradient-based optimization driving the inverse problem that uses the differentiable PDE solver. The loss with the initial guess is 0.212, whereas that value after the 20 optimization steps is 0.0048 which translates into a 97.7% loss reduction.

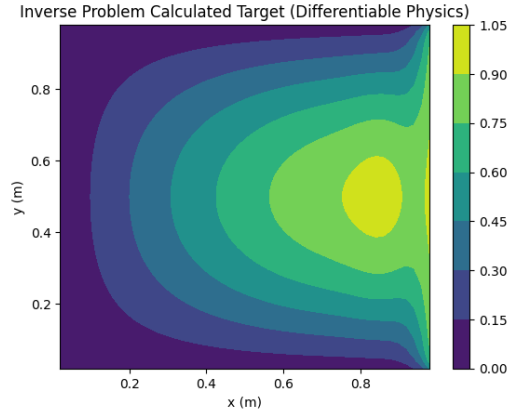


Figure 42. 2D Unsteady Head Equation bc1.
Calculated Distribution $\hat{T}(x_i, y_i, t = t_f)$

Finally, the time elapsed in resolving this inverse problem by a 6-core laptop Intel(R)Core-i7CPU@2.7GHz laptop was 26 minutes and 20 seconds; the majority of which was spent in solving the 20 forward PDE solutions. The unsteady 2-D Heat Equation with boundary condition 2 is the first test for differentiable PDE solver applied to inverse dynamical problems.

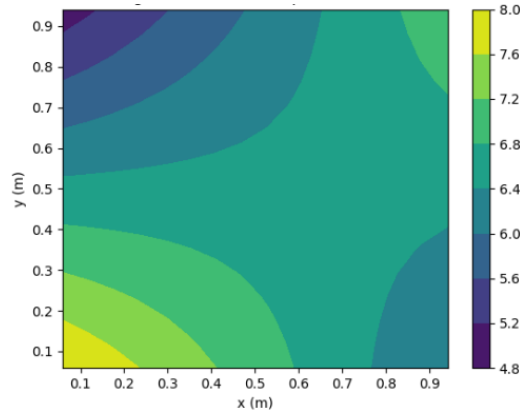


Figure 43. 2D Unsteady Head Equation bc 2.
Target Distribution $\hat{T}(x_i, y_i, t = t_f)$

Note that boundary condition has continuous piece-wise linear behavior. First the initial domain is discretized in 2401 points; these models are reduced with operational inference reduced order models to a much lower dimension of $N = 20$ than the original 2401 points. The guess for the initial distribution is 0 for all the reduced bases which means $T(D, t = t_{ini}) = 0$ at all points in D in the original high dimension space. Figure 43 shows the desired (or target) distribution at the final time $T^*(D, t = t_f)$. The gradient-based optimizer is run 20 times (same as the number of forward differentiable PDE resolutions).

Figure 44 shows the final distribution $\hat{T}(x_i, y_i, t = t_f)$ calculated by the gradient-based optimization driving the inverse problem that uses the differentiable PDE solver for the Operational Inference

reduced order model. The loss with the initial guess is 3.494, whereas that value after the 20 optimization steps is 0.045 which translates into a 98.7% loss reduction.

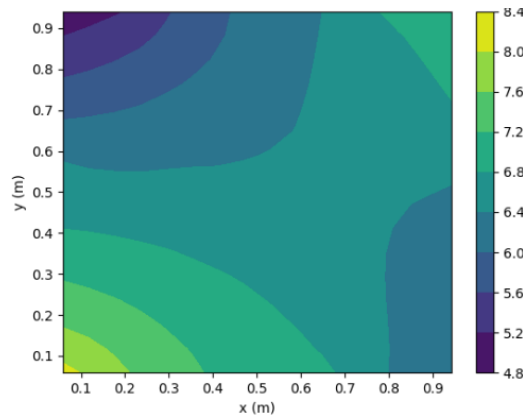


Figure 44. 2D Unsteady Head Equation bc2.
Calculated Distribution $\hat{T}(x_i, y_i, t = t_f)$

Finally, the time elapsed in resolving this inverse problem by a 6-core laptop Intel(R)Core-i7CPU@2.7GHz laptop was 0.6 seconds. This shows the significant acceleration of operator inference reduced order models while keeping an accurate PDE gradient assessment to efficiently find a suitable initial condition for the given target distribution in a large dimensional space.

3.10 Modeling Cross-Domain Interactions

Design optimization is a long and computationally expensive process. The reasons are manifold: a) large design spaces because of large number of design parameters, b) analysis of all the Key Performance Indicators for the relevant disciplines/output spaces, c) constraints on the design space that brand some design solutions as feasible.

Several surrogate modelling algorithms have been developed to mitigate some of the problems. However, when several output spaces need to be assessed direct prediction from input to each of the output spaces does not scale efficiently. In order to solve this some authors have proposed a Shared-GP model which embeds the simulation input and output spaces into a structured latent space [48]. This Shared-GP algorithm has been successfully proven for design problems with mechanical and topology optimization output spaces. More details about the algorithm are shown in Figure 45.

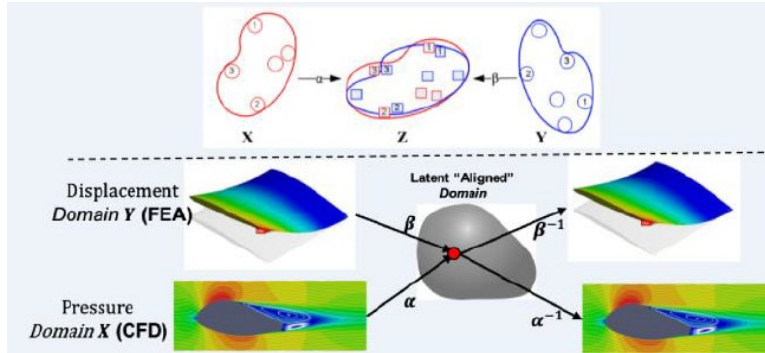


Figure 45: Conceptual illustration of the goals of cross-domain modeling

3.10.1 Evaluation 1: Dynamic Thermo-mechanical Problem

Consider a Modelica model previously developed with a flange connected to two supports via a spring-mass-damper and elasto-gap dampers. Additionally, the flange has a Thermal component, and a force is applied to it. More details about these components can be seen in the Modelica documentation [49]. In summary the problem has 17 parameters and two outputs spaces: the mechanical and thermal one.

First a proper dimensionality of 6 dimensions for the latent space was assessed for which the forward prediction (from input to output spaces) have high accuracy (R2 above 0.96). Later inference (from an output space to another output or the input space) is evaluated for the test data. Note that inference is the most challenging assessment. First inference from mechanical solution resulted in varying satisfactory results: a good inference of the mechanical input parameters such as spring constant, a mediocre inference of the thermal output space and a poor inference of thermal-related inputs as illustrated in Figure 46.

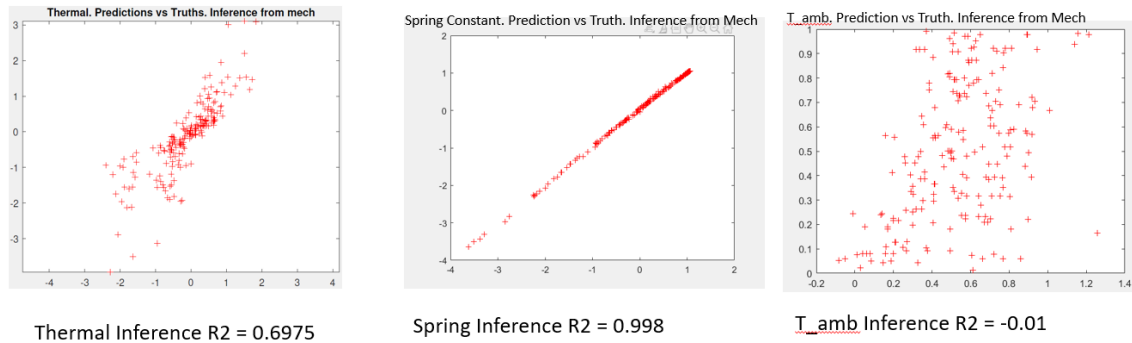


Figure 46: Performance of the algorithm in the dynamic thermo-mechanical problem

Secondly inferring from the thermal output space resulted in poor assessment. In Figure 47 one can see this for the inference of mechanical displacement, mechanical velocities and the input parameter forcing sine amplitude.

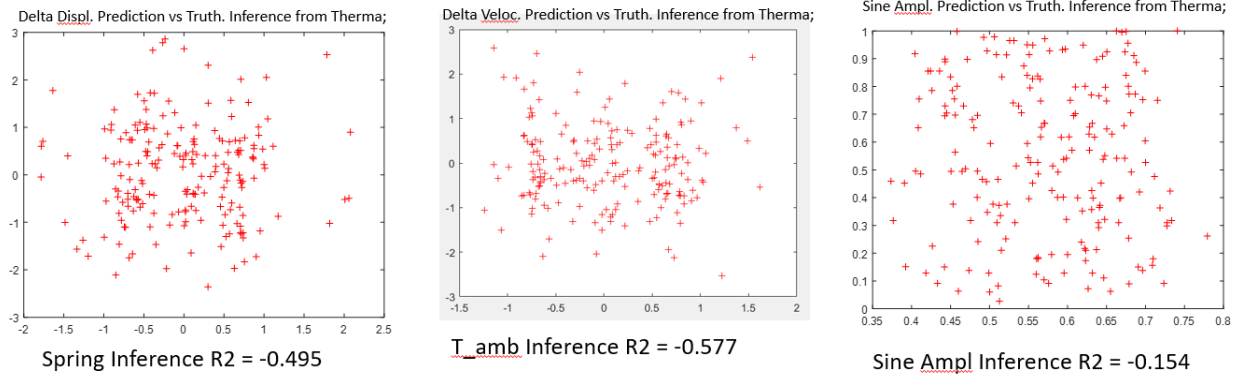


Figure 47: Inverse inference from the thermal domain

3.10.2 Evaluation 2: Bijective and non-bijective canonical problems

Given the results above, we proceeded with the method assessment on two canonical problems where we have full control of the problem. One problem is bijective and other is not bijective.

Bijective Output Space

$$y_1 = a_1^2 * a_3^{1.5} + a_2$$

$$y_2 = 1 + y_1 * \log(1 + y_1)$$

Non-bijective Output Space

$$y_1 = a_1^2 * a_3^{1.5} + a_2$$

$$y_2 = a_4 + y_1 * \log(1 + y_1)$$

Simulation inputs have a latent structure (to be discovered by GPLVM)

$$a_3 = \frac{a_2}{2}$$

[32]

The Shared-GP algorithm is applied to the two problems after proper dimensions for the latent space are found. Predictions (from input to output spaces) are very accurate as expected for both problems. The results for inference are different. For the bijective case, inference across output spaces is accurate (R2 ~0.97); however, for the non-bijective case the inference is quite poor (R2 ~0.19) as shown in Figure 48.

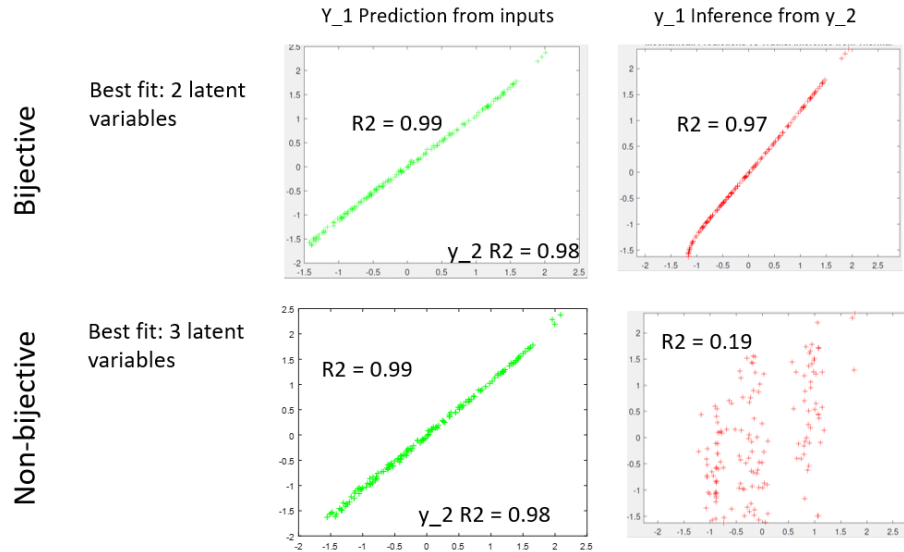


Figure 48: Comparison of performance in bijective and non-bijective cases

3.10.3 Summary

Shared-GP method has been successfully proven on a topology optimization problem for designing a part with mechanical constraints where we expect bijectivity between the stresses and the topology optimization densities [48]. However, the methodology fails for simple and complex non-bijectivity problems, the latter ones are quite common in engineering where disciplines experiment non-linear non-related behavior and each discipline solution usually follow different trends. The application of the methodology to bijective problems such as the topology optimization coupled with structural problem seems to be promising due to the dimensionality and design space reduction. For these design problems the Shared-GP formulation does not scale too good with the dataset size so potential implementation with neural networks rather than gaussian processes might be useful for researchers interested in the area.

3.11 Hybrid Topology Optimization

3.11.1 Introduction

Topology optimization (TO), a computational design method, can generate structures that achieve high physics-based performance objectives by optimizing the layout of the constitutive materials. Two main categories of TO exist: 1) density-based, which represents geometry as values between 0 (indicating no material) and 1 (solid material), and 2) level set-based, which utilizes iso-contours of signed distance fields to create shapes. The former is the most popular due to its relative simplicity but suffers from mesh dependency and produces structures with discontinuous edges, while level sets allow clear and smooth interfaces but are difficult to implement due to complicated shape gradients and the need to solve partial differential equations.

Both TO approaches have gained momentum in industries such as vehicle and aerospace for conceptualizing designs that are concurrently high strength and light weight. However, leveraging TO for simultaneous mechanical and aerodynamic design, e.g., minimizing drag coefficient while subject to a von Mises stress constraint, is not as well-developed, since the simulations for the two physics domains typically utilize different types of geometric representations. For example, most

works perform sequential optimization of an airfoil by first performing shape optimization on the external shape via methods with smooth contours, such as B-Splines, followed by a separate TO for the internal structure [50]. This leads to sub-optimal designs, as the gradients from the two potentially competing domains are not integrated into a single objective.

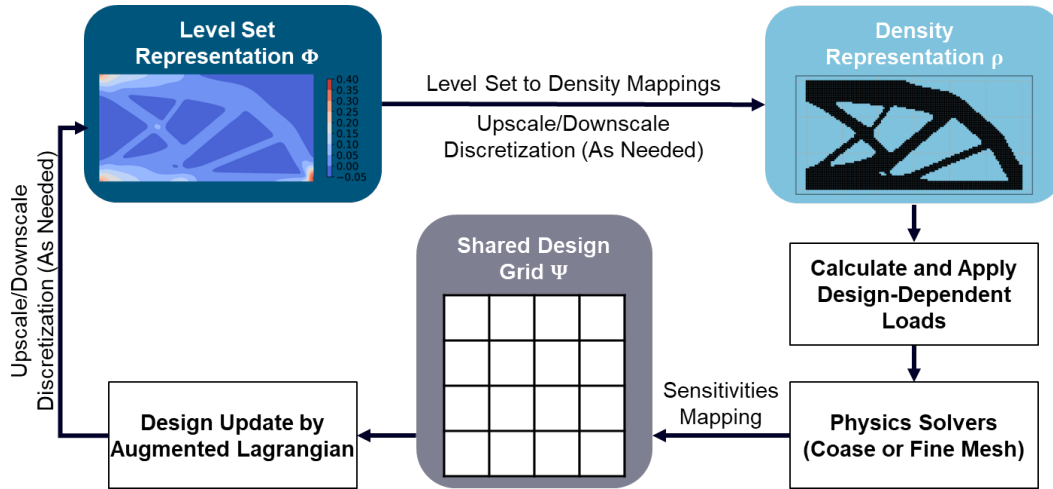


Figure 49: Flow of the hybrid TO framework, which can handle, if needed, multiple physics solvers and design-dependent loading. The shared design grid on which the design update is performed allows any number and type of solvers to be integrated for concurrent optimization.

3.11.2 Methodology

To allow concurrent TO of multiple physics, we propose a method that generates structures with smooth, level set-like representations while retaining the simplicity and efficiency of density-based approaches. This is accomplished by hybridizing the two common representations in TO. With little additional effort compared to conventional density-based methods, we utilize both geometric representations of the structure—densities for evaluation of the physics simulations, and level sets for evolution of the structure’s shape. The method also features the following advantages: 1) a smooth geometry, e.g., stereolithography file format (STL), can be immediately extracted from the optimal level set field without post-processing, 2) the continuity of the level set fields can be leveraged to upscale the design for certain cases, such as aerodynamic simulations, that prefer finer discretization, 3) the level sets further provide an efficient way to track pressure loaded surfaces as the design changes, and 4) a shared design field allows independent silos of physics solvers to be integrated by mapping their respective gradients onto a fixed grid. Figure 49 illustrates the flow of the proposed framework.

Fundamental Approach: Previously, we described the fundamentals of the proposed approach, which we briefly re-state here, with modifications to allow multi-physics problems. In contrast to traditional level set TO that solve multiple PDEs each iteration [51], we evolve the design with a first-order equation:

$$\Psi_{i+1} = \Psi_i + \Delta t V_n(\rho), \quad [33]$$

where Ψ_i is the shared design field on a fixed grid at time i and V_n is the normal velocity of the design, mapped onto the design grid. Note that Ψ_i is also a level set representation of the structure;

when the discretization of Ψ_i and Φ_i are the same, the fields are in fact also equivalent. However, we denote the two individually for the multi-physics cases where different solvers require different discretization or mesh.

The velocity V_n is obtained from the physics-based gradients on the density-based design representation, $\rho \in [0,1]$. In other words, for some physics-based objective function, f , that we wish to minimize, the velocity is $V_n = \partial f / \partial \rho$, which is much more tractable to obtain than in traditional level set approaches. Therefore, the density representation, ρ , is leveraged to update the level set representation, Φ .

To accommodate any number of constraints, agnostic to physics domains, we perform the shape evolution using a general augmented Lagrangian framework by defining the shape velocity as follows:

$$V_n = \frac{\partial J}{\partial \rho}, \quad [34]$$

where J is the augmented Lagrangian function for N constraints h_j ,

$$J(\Psi_i) = f(\Psi_i) + \frac{1}{N} \sum_{j=1}^N \left[\lambda_j h_j(\Psi_i) + \frac{\mu_j}{2} h_j(\Psi_i)^2 \right], \quad [35]$$

and λ_j are the multipliers while μ_j are penalty factors.

Design-Dependent Pressure Loading: As the velocities V_n correspond to the movement of the topology from iteration to iteration, it also proves to be useful for tracking the external surfaces of the design for design-dependent pressure loading. While most TO methods fix the boundary and loading conditions throughout the process, design-dependent loads change based on the topology and are more realistic for applications such as unmanned underwater vehicles fairings, where the external shape affects the pressure that they undergo. In literature, however, using only the density representation, which do not have clear 1/0 (i.e., no material/material) surfaces, require complex heuristic algorithms to identify external surfaces each iteration [52]. Meanwhile, with the level set representation, non-trivial shape integration and gradients are required for the same purpose [53]. Utilizing the hybrid representations, we propose a more straightforward method to both track and apply equivalent nodal loads on external surfaces. First, we define an auxiliary field, φ_i , to track the evolution of the external boundaries. Initially, φ_0 is defined as a fully solid ($\varphi_0 = 0$) field with the outer surface as the nodes where the initial pressure load is applied (e.g., the red line in Figure 50a-b). At each iteration, we update φ_i with the velocities V_n only at the identified surface nodes (red lines), then use edge detection methods (e.g., Scikit-image) to locate the new boundaries. Thus, the auxiliary field only tracks the loaded surface. Moreover, since this method only tracks external surfaces and ignores internal voids (Figure 50c-d), it works well for cases like UUVs, which must be airtight.

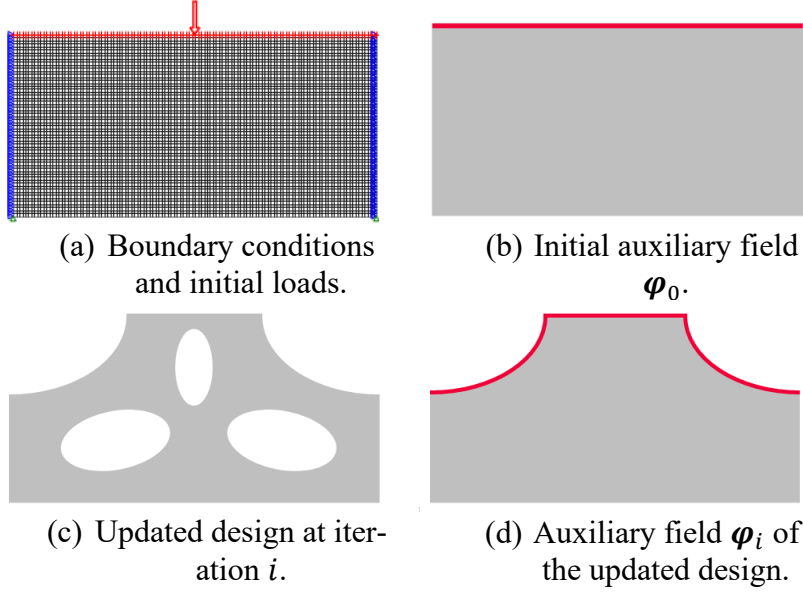


Figure 50: Example of the auxiliary field, φ_i , used to apply the design-dependent pressure loads for the problem shown in (a). External surfaces are outlines in red. The auxiliary field (d) undergoes the same changes as the design itself (c), but without internal voids.

Finally, given auxiliary φ_i , we utilize a method to linearly approximate the nodal pressure loads. The premise of the method was first proposed by [54], wherein all solid elements are presumed to undergo pressure loads on all sides (unless a node is fixed). Figure 51 illustrates this concept of an arbitrary 2D element (Figure 51a), where, for a nominal pressure value of p_0 and element side lengths of l_x and l_y ,

$$f_{x,e} = \frac{p_0}{4} l_x, \quad [36]$$

$$f_{y,e} = \frac{p_0}{4} l_y. \quad [37]$$

In the example, the 2×2 structure is fixed on the left edge; no loads will be applied there. At all other nodes, the nodal forces of each element (Figure 51a) are linearly summed to form the pre-calculated loads in Figure 51b. These pre-defined loads are saved in a constant matrix G . At each iteration, the design-dependent nodal forces are then calculated as $F_{p,i} = \rho(\varphi_i) \odot G$, where $\rho(\cdot)$ is a mapping that converts the auxiliary field into equivalent densities and $\odot$ is element-wise multiplication. Figure 51c demonstrates this for a simple design.

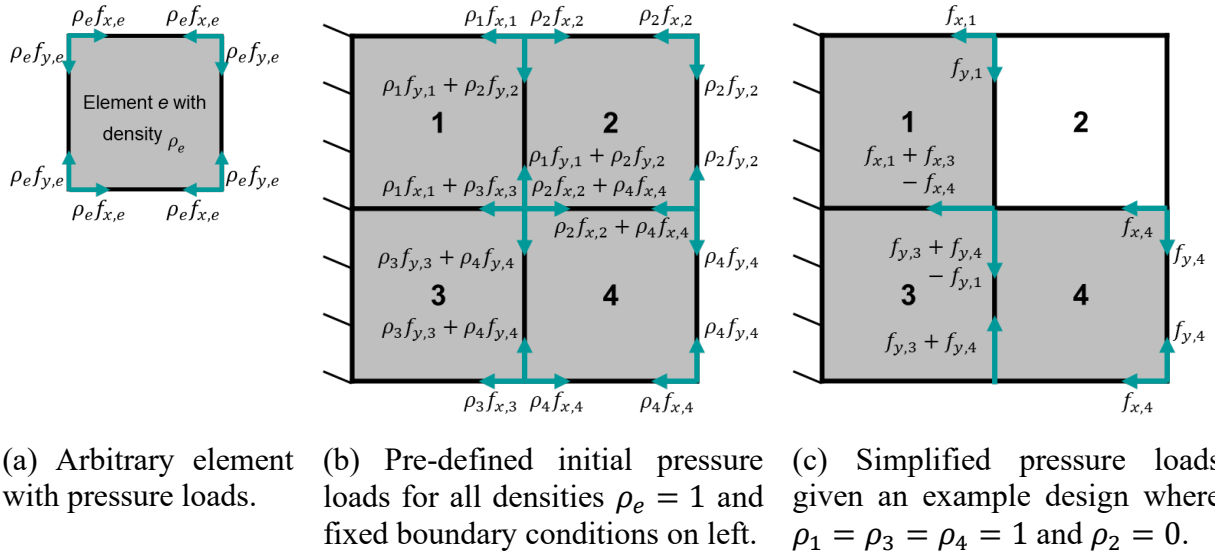


Figure 51: Illustration of the summation method to calculate design-dependent pressure loads.

In short, this simplified design-dependent approach is possible through the hybridized representations: we track the external boundaries using the level set field, then calculate and apply the pressure loads using the densities.

3.11.3 Results

Internal Structure Design Initialized from STL

First, we demonstrate the extension of the hybrid TO method to design the internal structure of a UAV fairing whose external shape is imported from an STL file (Figure 52a). Compared to a constant wall thickness that may result from parametric shape optimization, the proposed TO approach designs a variable width wall (Figure 52b).

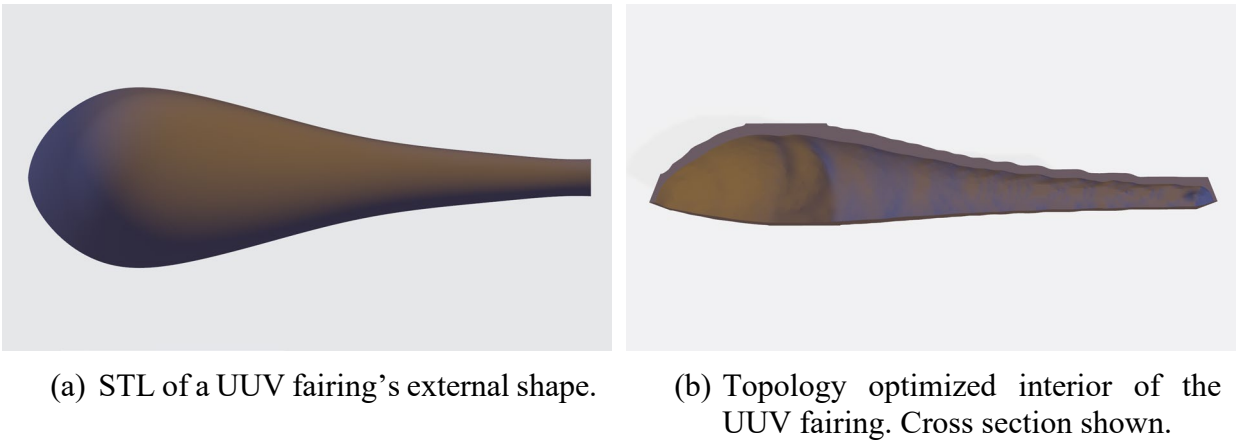


Figure 52: 3D result (b) for stress-constrained mass minimization, where the external shape is from an imported fairing design (a).

Design-Dependent Pressure Loading

To benchmark the proposed design-dependent pressure loading approach, we design a bridge following the boundary conditions in [52]. The initial boundary and load conditions are shown in Figure 53a, and the result where the loaded surface does not move is in Figure 53b. In comparison, with design-dependent loading, the optimized design (Figure 53d) has fewer small features branching from the top surface. This result matches that of the literature [52].

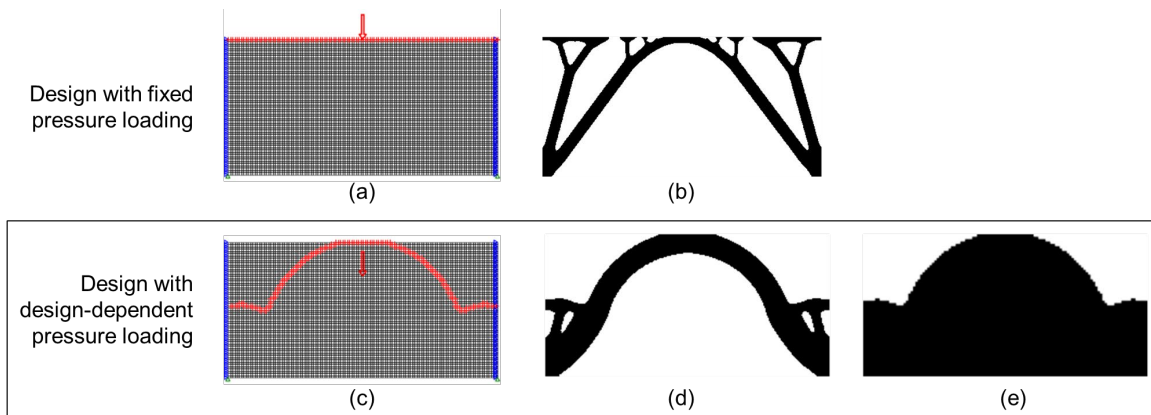


Figure 53: Benchmark results for a bridge design with fixed loads (a-b) and with the proposed design-dependent pressure loading (c-d), with its final auxiliary field (e).

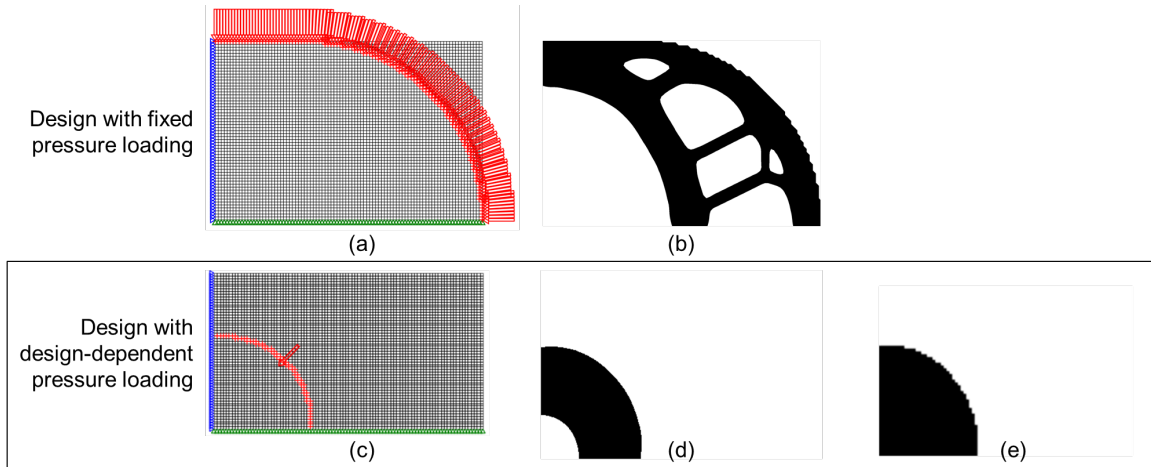


Figure 54: Optimized pressure vessels with fixed loads on the fixed outer surface (a-b) and with the proposed design-dependent pressure loading (c-d), with its final auxiliary field (e).

In addition, we perform hybrid TO of a pressure vessel with design-dependent loading. The optimal vessel with a fixed cylindrical outer surface is shown in Figure 54b. With design-dependent loading, however, the optimal design is in the form of an annulus. As [55] discusses, this makes sense both intuitively and analytically for a buoyant structure under these boundary conditions.

Aerodynamic Drag Force Minimization

To further demonstrate the applicability of hybrid TO to other physical domains, we perform aerodynamic drag minimization. To calculate the drag, we implement, using JAX and Python, a Newton solver for the Navier-Stokes equation based on [56]. The topology optimized airfoil design at a Reynold's number of 10, is depicted in Figure 55b, with an 82% reduction in the x-direction drag force.

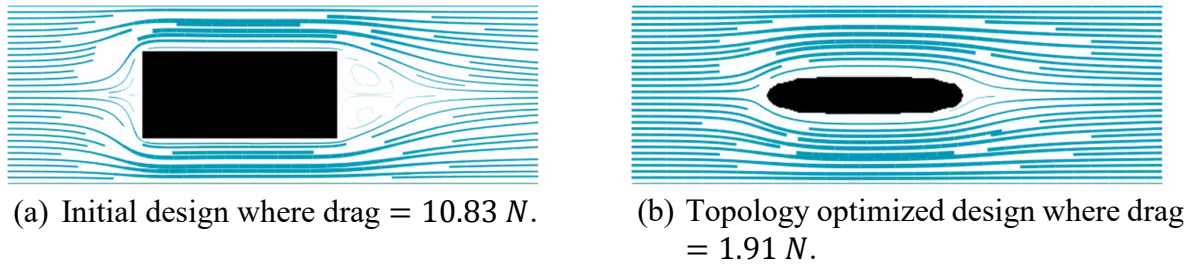


Figure 55: Benchmark result of a design with minimized drag at Reynold's number = 10. Streamlines of the fluid flow velocity are shown in black.

Stress-Constrained Aerodynamic Drag Minimization

The solvers for the two physics systems are performed on two independent meshes, i.e., linear elasticity with Fortran and fluid dynamics with Python/JAX. We demonstrate that our hybrid TO has the capability to integrate multiple physical domains simply by mapping sensitivities onto a common design grid. We perform a stress-constrained aerodynamic drag minimization with independent solid mechanics and aerodynamic solvers. We assume that the physics are de-coupled and that the flow velocity is low, i.e., the pressure load and stress come only from the hydrostatic pressure.

In the initial design (Figure 56a), the drag is 23.83, the volume fraction of the solid material is 0.081% of the entire design domain, and maximum von Mises stress is 124.62 MPa. Using hybrid TO, the optimal design (Figure 56b-c) has von Mises stress lower 11.92 MPa (less than the constraint of 15 MPa), reduces the volume to 0.077% while increasing drag only by a small amount to 26.78. The sacrifice in drag force is a direct result of the competition between the stresses due to the hydrostatic pressure and aerodynamics, showing that our proposed method can concurrently design under multiple physics.

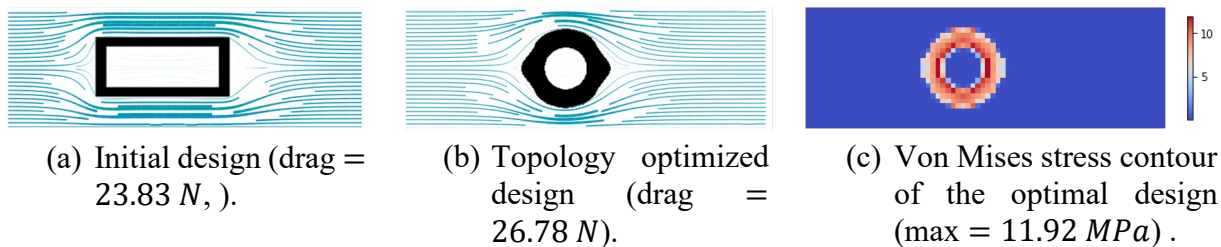


Figure 56: A UUV designed under stress-constrained drag minimization at Reynold's number = 10.

3.11.4 Observations

We find that TO using hybrid design representations and the augmented Lagrangian algorithm is effective for simple benchmark problems as well as more realistic UUV use cases under high pressure loads. Moreover, the ability to obtain both density-based and level set representations without post-processing allows the framework to be extensible to a variety of problems. For example, the densities allow more straightforward gradient calculation while level sets naturally lead to smooth surfaces, and level sets also provide a straightforward way to track the evolution of loaded surfaces, enabling design-dependent pressure loads. Leveraging these advantages, we have successfully extended the proposed hybrid method to design-dependent, aerodynamic, and/or multi-physics design.

3.12 Reinforcement Learning for Controller Design

3.12.1 Background

Design space exploration is a part of the design process where discrete components are chosen from a set of design component choices and their continuous parameters are tuned to meet the design requirements and objectives. The main challenge is to explore the large design-space and extract the best design among feasible design sets which is very sparse.

There is a large body of work focused on this topic and can be grouped into two directions: architectural and configurational designs as shown in Figure 57. Other relevant topics are Uncertainty Quantification, Graph Neural Network, Reinforcement Learning with uncertainty.

Design Level	Focus	Directions
Architecture	Hierarchical arch. design	Structure across fidelity, domains, components
	Exploration for novel arch.	Surprise, entropy, novelty
	Decoupling auxiliary design variables	Information decoupling, K-means
	Connecting performance goals across multi-domains	linearization, uncertainty propagation at set-points for control, robustness goals and infeasibility constraints
Configuration	Exploration for tuning	Surprise, entropy, novelty, curiosity
	Transfer tuning results across arch.	Subgraphs with similar structure

Figure 57: Architecture verse configuration design space exploration.

3.12.2 Methodology

The main motivation is to navigate through the design choice while updating the policy for better selection mechanism without being adversely affected by the off-policy data. Proximal policy optimization (PPO) [57] is an on-policy algorithm and can be used for environments with discrete and continuous action spaces which makes it particularly useful in design space exploration (DSE) tasks. For our design problems, we implemented multi-discrete action spaces since each compo-

ment has multiple selections and flattening out the whole design choices are computationally intractable. We implemented the PPO algorithm with 13 multi-discrete actions for 1 battery with 27 variants, 4 electronic speed controllers (ESCs) with 20 variants, 4 propellers with 417 variants, and 4 motor over 83 variants on Southwest Research Institute (SwRI) use case [58]. We start with a feasible initial design whose variant selection is [25, 17, 7, 4, 2, 239, 140, 312, 382, 31, 35, 11, 18]. This design achieves [365, 300, 0, 0] over four objectives. After running 8 hours, PPO finds a design with variant selection [24, 18, 6, 7, 12, 19, 365, 312, 382, 31, 35, 14, 9]. The resulting design achieves [348, 300, 352, 0] over four objectives. The progress of the exploration is illustrated in Figure 58.

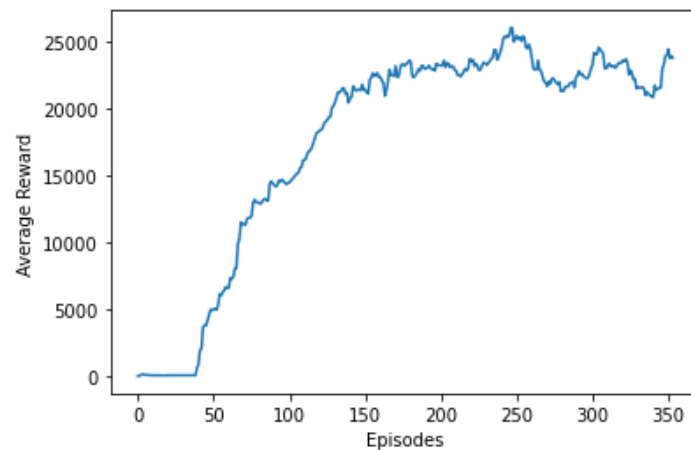


Figure 58: Progress of the RL algorithm in the design space exploration

We defined the reward as the average of four design objectives. We created offline data with one component type random policy and warm started PPO (otherwise it is too sparse to learn). The environment is quite slow, to speed up the reinforcement learning (RL) training, we reused previous designs. Our formulation tends to keep the next policy action constant. Therefore, it tries to find the best in one selection. Since the training is too slow for hyper-parameter tuning, we could not be able to optimize that. After our test on simplified problems, we realized that richer reward formulation is needed. The DSE creates very sparse reward choices, and this severely affects the performance of PPO. We created offline data to find feasible initial policies, and this helps at the start of the training. However, during the training, increasingly sparse rewards cause catastrophic forgetting.

We implemented further improvements such as

- generalized environment to multi-discrete + continuous variables case,
- extended observations with Mass, moment of inertia (MoI) and center of gravity (CoG) for richer representations,
- normalized observations and reward,
- extended PPO algorithm to multi-discrete + continuous variables.
- collected warm start data to initialize policy.
- trained initial policy with warm start data.
- trained PPO algorithm with this initial policy.

We ran the PPO algorithm for 2 days. Initialized policy trained with 1450 designs (with 32 design more than 0.5 score) and trained PPO algorithm with new 1900 designs (with 35 design more than 0.5 score). The max score was 0.6875 over 1 with individual scores [400.0, 300.0, 400.0, 0.0] compared to the principal investigator (PI) meeting result [348, 300, 352, 0]. Although the results are improved for the first three objectives, we could not get satisfactory results for the fourth objective and in general, in the sense of convergence and reaching the optima.

To make the reward richer, we included three additional pieces of information about the design.

1. **Trim found:** Compute reward parsing *forward*, *turn_300* and *turn_500* files. Reward is the sum of *forward_constraint*, *hover_constraint*, *turn_500_constraint*, *turn_300_constraint*, *max_speed*. The range is approximately 2.5-10.5.
2. **Trim not found & no collision:** If the files in step 1 are not available and there are no collisions, parse *fdm/trim/output.out* file. Reward is the mean of the magnitude of the entries of the lines starting with “UVWdot, PQRdot (m|r/s²)” (averaged over multiple entries). The range is approximately 0.5-4.7.
3. **Collision:** If there is a collision, parse *component_collisions.txt* by averaging collisions. The range is 0.0002-0.03.

We trained the new policy with PPO using the rich reward, but RL algorithm cannot get back to feasible designs (possibly non-feasible rewards are not guiding back to feasible designs).

3.12.3 Summary

DSE has exponential explosion on actions and observations, therefore its exploration mechanism needs to be heavily controlled by other mechanisms. The exploration of classical RL approaches including random exploration, noise modulation, (Monte Carlo Tree Search (MCTS) are not efficient enough for DSE type problems. Making reward richer helps but we need to define auxiliary rewards even for unsuccessful design to guide RL back to feasible designs.

One remedy could be learnt to limit the scope of action and its schedule along the DSE applied in PPO so that nearby designs contain richer reward information.

3.13 Bayesian Optimization for Spaces with Parametric Discrete Choices

3.13.1 Background

Bayesian optimization (BO) is a well-established approach to globally optimize functions, [59]. As opposed to other global optimization approaches, BO uses Bayesian statistics to sequentially build a probabilistic response surface that supports the optimization. BO coupled with automatic differentiation framework has extended the efficiency of the methodology to higher dimensional spaces than the ones of the early BO implementation, see Botorch [60].

Cyber-Physical system (CPS) design spaces are composed of not only continuous and numeric-discrete domains but also categorical ones; system component choices must be made from a pool of off-the-shelf components. Each member in the component pools has underlying characteristics which can be potentially used as measures of proximity between discrete choices.

BO formulations account for domains of different types: continuous, discrete, and categorical; however, the categorical formulations do not account for underlying numeric-discrete characteristics like the one present in cyber-physical components; therefore, the learning transferred across component choice members does not exploit the information contain in the characteristic spaces.

Adapting the mixed-integer BO to domains where discrete choice characteristics provide proximity/distance could enable a more efficient architecture and design optimization of these cyber-physical systems.

3.13.2 Methodology

To enable meaningful learning transfer across component alternatives, the component choice is modelled in the numeric-discrete characteristic space rather than in the categorical one. Distance and therefore, proximity can be measured on this new space at the expense of increasing the CPS space. Substituting the categorical space for component choices by a discrete-numeric characteristic space is straight-forward in the response-surface building step of the BO process. However, in the subsequent optimization step of the BO process, two methods have been explored:

1. Relaxing the characteristic optimization space from numeric-discrete to a continuous one where conventional continuous optimization algorithms can be used. Then, find the nearest neighbor which is feasible. The nearest neighbor search is performed with the k-dimensional tree which is a space-partitioning data structure for organizing points [61]. This method is referred to as “EI Nearest N” from now on.
2. Use mixed-integer optimization to optimize the continuous and numeric-discrete space. In this case a genetic algorithm (GA) capable of handling numeric-discrete variable is developed. The numeric-discrete mutation probability is assumed inversely proportional to the distance between component alternative characteristics, and the off-spring cross-over probability for each design dimension is 0.5 for each parent. This method is referred to as “EI MIO” from now on.

The two methods are applied to the maximization of a canonical problem to see which method performs better. This problem is made of 3 component decisions each with a choice of 5 choices; on the top of that there are 3 continuous design parameters. Figure 59 shows that the method “EI MIO” performs best compared to the method “EI Nearest N” and random searches for the canonical problem.

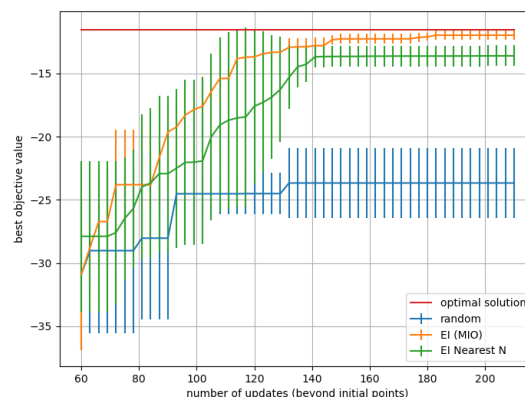


Figure 59: Maximization on canonical function- several methods

After this learning the focus is on testing the “EI MIO” method on scenarios closer and closer to realistic CPS problems.

3.13.3 Evaluation

This test involves the optimization of a design problem with 3 component decisions with 26, 30 and 30 alternatives for each decision respectively. The component characteristic space for each decision has dimensions 3, 4, and 2 respectively. Additionally, there are 3 continuous decisions to make during the optimization.

The optimization with “EI MIO” is compared with several state-of-the-art optimization algorithms provided by the commercial tool Simcenter HEEDS [62]; specifically, the algorithms to compare with are SHERPA (a hybrid adaptive proprietary algorithms), GA, Particle Swarm Optimization (PSO), and Simulated Annealing (SA). Please note that the optimal solution is known for this optimization test.

The results are shown in Figure 60 and Figure 61. The methods performance varies depending on the function evaluation budget. In general, the results show that the “EI MIO” performance is better than the SA and GA, whereas the performance is poorer than that of the PSO and SHERPA. Compared with GA, the “EI MIO” performance is worse than that of GA only in the range of small number of function evaluations. Similarly, compared with PSO, “EI MIO” performance gets closer to the that of PSO in the range of high number of function evaluations.

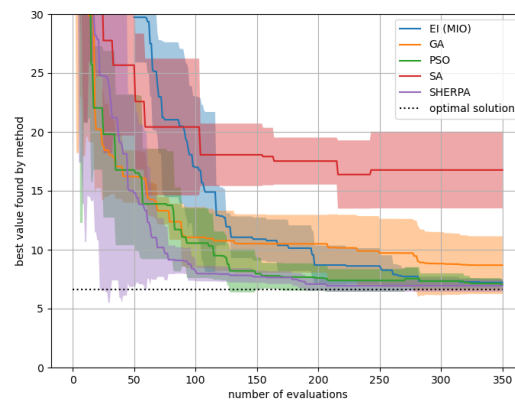


Figure 60: Minimization on large choice function

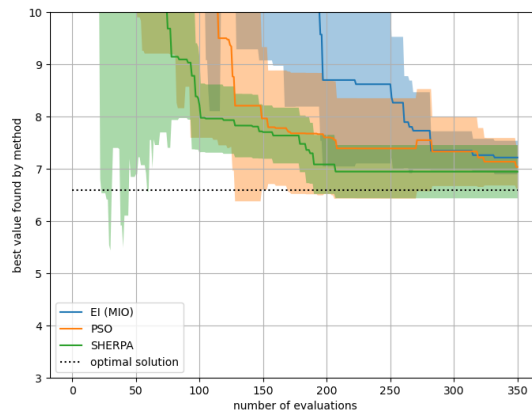


Figure 61: Minimization on large choice function. Zoom.

3.14 Symmetry Constrained Bayesian Optimization

3.14.1 Introduction

There is a consequent and consistent human attraction to symmetry. While symmetry abounds in nature (see e.g. Figure 62), it is also ever-present in human-made and designed objects. A modern aircraft has an engine layout which is symmetric about the center-line (nosecone to tail fin), a car has a symmetric wheel layout, and a hexacopter (HEX) has a symmetric arm layout about the geometric centroid. A human, like a submarine, has bilateral symmetry, and a human face is approximately symmetric about the vertical axis.

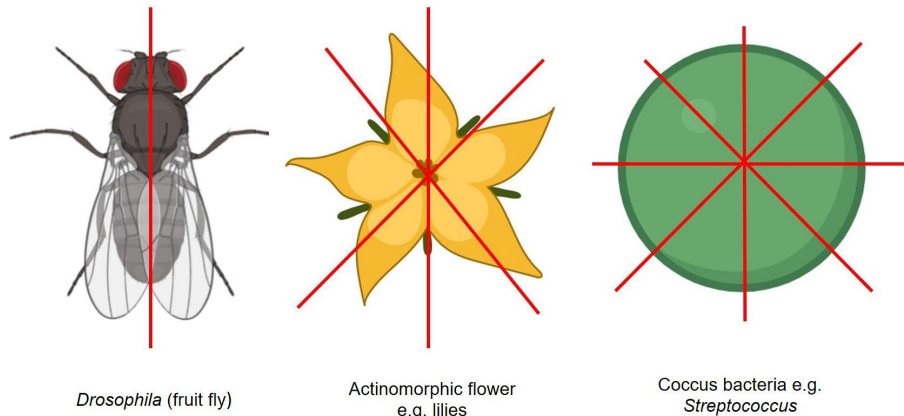


Figure 62: Figure depicting different symmetries found in biology; bilateral (*Drosophila*), radial (actinomorphic flowers) and spherical (coccus bacteria) symmetry [63]

Much of the appeal of regularity and symmetry is aesthetic, but more fundamentally (and importantly) these phenomena allow us to reason about and understand larger and more complex objects [64] by studying their smaller and simpler parts. By exploiting symmetry, we can reduce the complexity of a problem, and in doing so, reduce the number of variables we need to consider.

By extension we are deconstructing Aristotle's old adage which says that the "whole is greater than the sum of its parts". For our purposes it is perhaps more accurate to say that the whole is commensurate with the sum of its parts, if appropriately designed and assembled through the exploitation of symmetry. It is not surprising that symmetry has been studied extensively in the computational sciences, such as machine learning, computer science, operations research and mathematics. Symmetry exploitation in surrogate-based optimization problems is not as mature as other topics and is the topic we shall pursue herein. Specifically, by combining symmetry exploitation with Bayesian optimization [65].

In the quest to find an optimal design we are often faced with an enormous search space (which in our setting will contain both discrete and continuous design variables). We will assume that the best design does exist somewhere in this vast space (though it need not strictly speaking). The goal of a search algorithm is to locate that design – preferably as cheaply and as fast as possible. This task can be made considerably more efficient if we take symmetry into account. If several points in the search space are related by a symmetry, then we never want to visit more than one of them because they are equivalent [64]. And so, to make use of any present symmetry, two problems need to be solved:

Symmetry detection: We need to be able to detect the symmetries in the search space. This is a non-trivial task which is outside this scope. Rather, we will assume that the symmetry is known.

Symmetry exploitation: We need to be able to exploit the symmetries in the search space. This is our focus. We will also show that symmetry itself can be optimized over, to find an optimal design. To guide our discussion and expose the ideas we will make frequent reference to a practical design problem, that of finding an optimal hexacopter design will be our running example.

Hexacopter design study

Our goal is the design optimization of a hexacopter which is a type of unmanned aerial vehicle with six arms, as in Figure 63. The design space of the hexacopter is vast and includes continuous as well as discrete variables. But it is also clear that the hexacopter has multiple symmetries which we can exploit in our search for an optimal design.



Figure 63: Hexacopter Design. The design space includes the choice of rotors, motors, batteries, and other components. These choices are discrete. Continuous design variables include e.g. the length of the arms and the pitch of the rotors. To find an optimal design, discrete and continuous variables must be optimized simultaneously. This is a mixed-variable high-dimensional optimization problem.

3.14.2 Technical Details

In 1996 Crawford et al. [64] concluded that “tying symmetry exploitation to a specific search algorithm seems premature”. Yet despite their better judgement that is what we shall endeavor to do in this section. We introduce the method of symmetry constrained Bayesian optimization (SCBO) which is a modification of the BO algorithm that leverages the symmetry of the design space to reduce the number of evaluations to find the global minimum for valid designs. We will use the HEX example to illustrate the method.

We begin by assuming that the design space is symmetric. The design space is partitioned into $K = 6$ symmetric design-variable groups, one for each arm. The independent variables are the same for each arm and are contained in the independent sector (the tiled equilateral triangle). The developed algorithm is given in the algorithm block shown in Figure 64 which contrasts the difference between the traditional Bayesian optimization against our symmetry constrained BO:

Algorithm 1 Original BO with PR	Algorithm 2 Symmetry constrained BO with PR
<pre> 1: Input: $f : \text{dom}(\mathbf{X}) \times \mathbf{Z} \rightarrow \mathbb{R}$ 2: Initialise: $\mathcal{D}_0 \leftarrow \emptyset, \text{GP}_0 \leftarrow \text{GP}(\emptyset, k)$ 3: for $n = 1$ to N do 4: $(\mathbf{x}_n, \theta_n) \leftarrow \underset{(\mathbf{x}, \theta) \in \text{dom}(\mathbf{X}) \times \Theta}{\text{argmax}} \quad \mathbb{E}_{\mathbf{Z} \sim P(\mathbf{Z} \theta)}[\alpha(\mathbf{x}, \mathbf{Z})]$ 5: Sample $\mathbf{z}_n \sim P(\mathbf{Z} \theta)$ 6: Evaluate $y_n = f(\mathbf{x}_n, \mathbf{z}_n)$ 7: $\mathcal{D}_n \leftarrow \mathcal{D}_{n-1} \cup \{(\mathbf{x}_n, \mathbf{z}_n, y_n)\}$ 8: Update posterior GP_n given $\mathcal{D}_n$ 9: end for </pre>	<pre> 1: Input: $f : \text{dom}(\mathbf{X}) \times \mathbf{Z} \rightarrow \mathbb{R}$ 2: Initialise: $\mathcal{D}_0 \leftarrow \emptyset, \text{GP}_0 \leftarrow \text{GP}(\emptyset, k)$ 3: for $n = 1$ to N do 4: $(\mathbf{x}_n, \theta_n) \leftarrow \underset{(\mathbf{x}, \theta) \in \text{dom}(\mathbf{X}) \times \Theta}{\text{argmax}} \quad \mathbb{E}_{\mathbf{Z} \sim P(\mathbf{Z} \theta)}[\alpha(\mathbf{x}, \mathbf{Z})]$ 5: Sample $\mathbf{z}_n \sim P(\mathbf{Z} \theta)$ 6: for $\lambda \in \Lambda$ do 7: $y_n \leftarrow f(g(\mathbf{x}_n, \mathbf{z}_n, \lambda))$ 8: $\mathcal{D}_n \leftarrow \mathcal{D}_{n-1} \cup \{(g(\mathbf{x}_n, \mathbf{z}_n, \lambda), y_n)\}$ 9: end for 10: Update posterior GP_n given $\mathcal{D}_n$ 11: end for </pre>

Figure 64: Algorithm Bayesian optimization original and symmetry constrained.

3.15 Verification and Assurance of Cyber-Physical Systems

3.15.1 Background

The design of aircraft has seen a rapid shift over the past few decades, where designs that were originally highly mechanical with very few interacting components have now become highly multi-disciplinary with several interacting components. Further this transition has seen a drastic increase in the incorporation of embedded systems and software components, at various scales of the system design, making modern designs highly cyber-physical in nature. Thus, modern aircraft systems can be viewed as CPSs. Researchers [66] have demonstrated the disproportionate super-linear increase in the design, integration, and testing time as a function of the system’s complexity, with the complexity being characterized by the number of interactions between the different sub-systems. Modern aircraft designs lie on the higher end of this spectrum, and hence require not only longer development cycles, but also struggle with long verification, assurance, and certification processes.

Focusing on the verification, assurance and certification steps, a key drawback of established methods is that the certification standards, the design documentation, and the corresponding assurance evidence all produce artifacts that are for human consumption. This makes automation of verification and assurance processes highly complex as several subjective assessments of the generated artifacts are required to certify the designed vehicle for operation. Often in highly complex systems, as in the case of CPSs, this implies that there can be a lead time of as much as five years post-development to acquire the Authority to Operate (ATO). To address this issue, there have been several recent efforts [67, 68, 69], including efforts by DARPA [70] attempting to realize

revolutionary changes to systems engineering paradigms, that have attempted to automate the process of Assurance Case (AC) generation for complex products.

These efforts in automation of assurance and certification have addressed the tasks of generating ACs for CPS from processed or semi-processed information. They build on established standards such as Goal Structured Notation (GSN) [71] and Claim, Argument, Evidence (CAE) [72]. GSN is a widely used graphical representation of the AC relating the system certification requirements to evidence, that can be of the formal, subjective and/or inductive kind, through an argument reasoning strategy. Although, these standards have significantly improved and accelerated the process of construction of the AC for complex systems, as demonstrated by frameworks such as Advocate [68], ReSonAte [73], etc., there is still a gap in the literature as these frameworks require three inputs that are not "automatable" and require an expert-in-the-loop. These are,

- **Formalization of requirements:** Current practices require an expert to process the certification standards and extract the corresponding requirements from them. The expert is further required to extract the semantics from the requirements, identifying the context, assumption, and parameters to which the requirements correspond.
- **Construction of argument patterns:** ACs constructed using GSN require an argumentation template that is to be used to map the claims to the evidence supporting the assurance of the system. This argumentation template is heavily driven by domain knowledge and requires considerable understanding of not only the CPS, but also its supporting artifacts – such as the functional requirements, failure modes, etc. that are often written natural language without any formal representations.
- **Formalization of evidence:** Finally, as GSN standards enforce that certification of CPS needs concrete evidence in its assurance, there is a need to identify relevant and suitable evidence to quantitatively evaluate the confidence in the satisfaction of an assurance claim. The identification of relevant evidence and its assessment of satisfaction is another subjective and expert-in-the-loop task.

Motivated by the recent development of Generative Pre-trained Transformers (GPTs) and the cutting-edge large language models (LLM) products such as ChatGPT/GPT4 [74], PaLM [75], LLaMa [76], amongst several other LLMs, we develop an automated document processing pipeline to address the gaps in certification of complex CPS. We demonstrate that, with the appropriate prompt engineering, these LLMs can automatically extract information from well-established certification guidelines such as Title 14 of the Code of Federal Regulations (CFRs) and additional functional requirements specifications in portable document format (PDF) formats to aid the AC construction. To that end, we develop and demonstrate a fully automated framework for the verification and assurance of complex CPS that can significantly accelerate the certification of complex systems.

Our framework makes three contributions over the state-of-the-art,

- **Semantically linked Representation of Knowledge:** We introduce a semantically linked representation of knowledge that utilizes a graph database to perform complex reasoning about the CPS under consideration.
- **Automation of AC Generation:** We utilize LLMs to automate the construction of the AC such that the entire case can be generated without the need for subjective expert-in-the-loop assessments of system artifacts.

- **Automation of CPS Verification:** We develop a Digital Twin (DT)-based verification interface that permits the incorporation of simulation evidence in the assessment of assurance of the CPS, enabling assurance and evidence co-generation.

3.15.2 Methodology

Figure 65 provides an overview of the framework developed for the automation of AC generation for CPSs. The framework builds on a domain model (a.k.a., ontology / schema) supplied by an expert to automatically construct a semantically linked representation of the knowledge stored in various system artifacts. The system artifacts considered in our exploration include the certification documents such as Title 14 of CFR, functional requirement specification documents, system models defined in modeling languages such as System Modeling Language and associated DT evidence emerging from simulations. Information from the natural language documents (i.e., the certification requirements and the functional requirements) are automatically extracted from the PDF documents and populated in a KG establishing relationships between the requirements, the system, their instances, and the corresponding DT implementations. Our framework automates all the steps in gray, i.e., the extraction of information by the LLM, the construction of the KG, the construction and evaluation of the AC. The steps in white are manually executed, but these are one-time offline events. These include the construction of the domain model and the collection of the system artifacts.

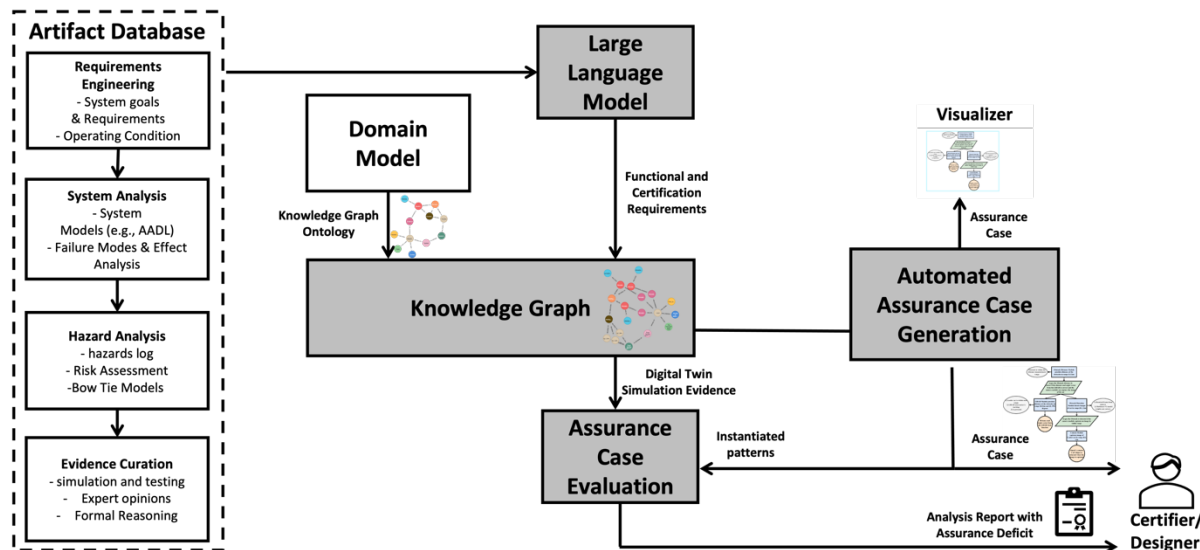


Figure 65: A Conceptual Overview of the Proposed Methodology

Assurance Information Extraction

Our pipeline begins with the information extraction step extracting information related to the system assurance from the system artifacts. To achieve this, we use an encoder-decoder transformer and prompt engineering techniques to extract the information for assurance from requirements documents which are written in natural language. The Transfer Learning with a Unified Text-to-Text Transformer (T5) [77] model to extract the assurance information. All T5 models were pre-trained on a huge corpus named Colossal Clean Crawled Corpus (C4) which is also introduced in

[77]. Due to our hardware limitations, we only use the pre-trained models tagged with t5-small, t5-base, t5-v1_1-small and t5-v1_1-base downloaded from Huggingface as the starting check points. The t5-small and t5-base have also been fine-tuned on downstream applications including text classification, abstractive summarization, question answering, and English to German, French, and Romanian translation. To compare and demonstrate the performance impact of pre-trained models, we also used the same configuration of T5 models without any pretraining or finetuning. To test the performance of our transformer models, we have domain experts manually annotate the item, paragraph ID, evidence, requirements, context, assumptions, and parameters from Federal Aviation Administration (FAA) regulation paragraphs, taking 48 instances as training set, 6 as validation set and 6 as test set.

All annotated Federal Aviation Administration (FAA) paragraphs are prepended with a prompt “Please find the requirements and other elements from the regulation”. Because the original T5 tokenizer is unable to detect new-line symbol (as known as “\n”), table symbol (as known as “\t”) and section symbol (“§”), we add these tokens into the vocabulary of tokenizer.

We train the models with the training data for 10000 epochs by default, with the starting learning rate of 2×10^{-5} . The neural network is updated with AdamW [78]. We use early stopping strategy when the performance does not improve for 500 epochs on the validation set and we report the final performance on the test set. In terms of performance, we use ROUGE F1 score [79] to measure the similarity between the annotated ground-truth and the system generated output.

Knowledge Graph Construction

Having extracted the information from the supplied artifacts, the information is semantically linked in a KG, implemented in Neo4J [80]. The domain model constructed by a domain expert, is quite generic and follows the standard GSN meta-model and is illustrated in Figure 66.



Figure 66: Domain Model of the KG Constructed using the Extracted Information

The domain model links information required to construct the AC to that of the DT. In particular, on the AC side, the domain model captures the relationship between the claims present in the certification and requirements documents to the system that the claim references, the underlying context under which the claim is valid, the assumptions made for the flight conditions, and the parameters that are associated with the claims. The identified system is linked to a DT representation which is captured in the system model. The system model further expands on the DT representation by linking the system to its intrinsic components and their functions. The functions are subsequently linked in the system model to a DT simulation in which the behavior of the system is simulated under the operating context and assumptions identified by the relevant claim of the AC.

The DT simulations are subsequently linked back to the system artifacts in the KG there by associating evidence with the claims, enabling quantitative assessment of the assurance of the CPS.

3.15.3 Results

Use-case Consideration

We apply the developed framework for the assurance of a landing gear system of an aircraft, following the details provided in [81, 82]. We evaluate the ability of the designed landing gear to meet both functional and certification requirements, while quantitatively identifying requirements that lack coverage. The functional requirements considered involve the responsiveness of the landing gear to pilot commands, measuring the time taken for the landing gear doors to open and close, and that of the landing gear retraction mechanism to extend and retract. Several certification requirements are considered and are extracted from the Title 14 of the CFR document. These include the landing gear positions at different flight speeds, the loads experienced by the landing gear mechanism and doors, while also capturing the requirements imposed by the landing gear on other subsystems. As we do not model the other subsystems, we primarily identify gaps in the assurance case that needs to be addressed with additional models and evidence.

We construct a system model of landing gear using Siemens System Modeling Workbench where a SysML representation of the CPS is constructed. This implementation of the system is illustrated in Figure 67. The system model captures three landing gears, their doors, the sensors associated with identification of the door status and their corresponding displays.

Performance of LLM

Table 4 demonstrates the basic information and ROUGE scores of all model configurations. We observe that performance with pre-training is much better than the counterpart without pre-training. Moreover, with proper finetuning, the models without “v1_1” tag are outperforming the ones with “v1_1”, although the latter possess more parameters. Our experiments also demonstrate the merit of transfer learning, or few-shot learning. Traditional information extraction requires annotation on 600 documents and more than 20000 sentences [83], which is prohibitive to extend the similar tasks to other domains. Our proposed approach significantly reduces the workload of annotation.

Table 4: Performance of pre-trained, un-pretrained and LoRA finetuned models. It is noted that t5-small and t5-base have been finetuned with downstream applications such as summarization and translation while the ones with v1_1 have not gone through any down-streaming application finetuning.

	ROUGE w. P.T.	ROUGE w/o P.T.	ROUTE w. LoRA
t5-small	0.7728	0.5718	0.7005
t5-base	0.8034	0.5057	0.7333
t5-v1_1-small	0.7254	0.5771	0.5621
t5-v1_1-base	0.7711	0.5763	0.7274

Constructed KG

We present a snapshot of the constructed KG, in Figure 69, resulting from the automated information extraction process. The KG illustrates the semantically linked requirement to the system model and the corresponding DT simulation. The executed simulation runs are populated as evidence in the KG enabling the evaluation of satisfaction of the requirement. On the AC side, the requirements, the associated parameters, and the constraints are automatically extracted and linked to the other elements in the graph enabling the construction of an AC by traversal of the resulting graph.



Figure 69: Example of the constructed knowledge graph capturing and linking assurance information to system models.

3.16 Advanced Simulations

The following subsection walks through some of the advanced simulations that we investigated as part of the development of the Systemic Generative Engineering framework.

3.16.1 Computational Design of Soft Matter Computers

3.16.1.1 Introduction

Soft matter materials, such as polymers, show promise for next generation materials for electronics and battery applications. As an example, organic field-effect transistors (OFETs) have demonstrated performance surpassing widely used amorphous silicon field-effect transistors (FETs) and have additional advantages over inorganic transistors due to being physically flexible, lightweight, can cover large-areas, and potential for low-cost mass-production [84]. Additionally, polymer structures can be synthesized with great complexity, which result in tunable (and potentially re-configurable) electrical and optical properties for more targeted applications.

The challenge with OFET design is that effects at multiple length scales need to be evaluated and included in the design process as shown in Figure 70. The key challenge is the lack of fundamental clarity at the microstructure (the packing and interactions of multiple molecules) which limits the material performance. For design of these materials, factors at each length scale impact the device performance and as such need to be considered.

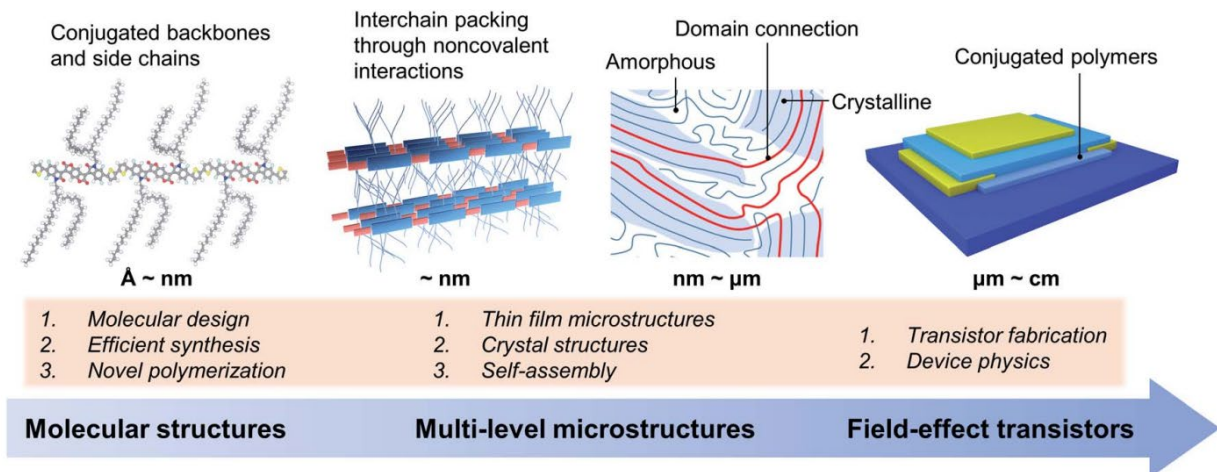


Figure 70: Overview of research efforts and relevant physical characteristics at different length scales [84]

Simulation, theory, and experimental efforts exist at each of the length scales with different levels of maturity, however current efforts at designing and developing OFETs are done in a fragmented manner – utilizing expert guidance or focusing on design on a specific set of length scales. Recent advances in manufacturing of these materials and machine learning techniques offer potential avenues for addressing these challenges.

3.16.1.2 Methodology

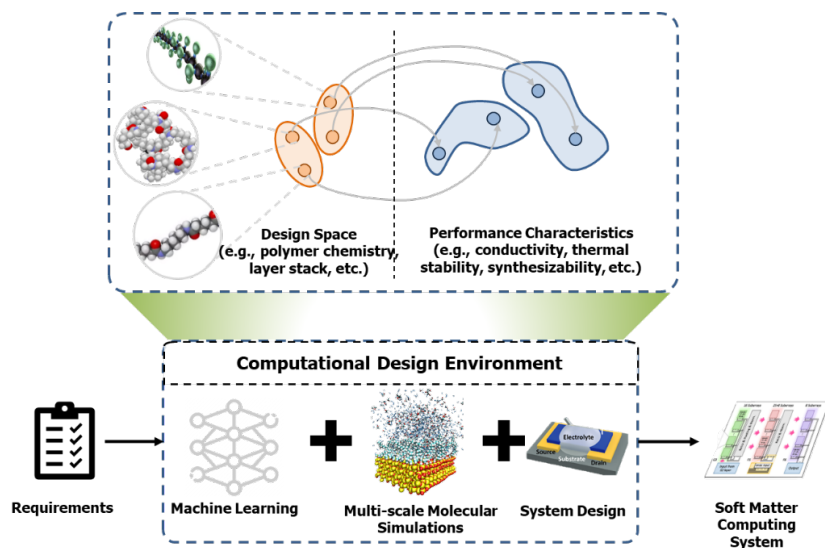


Figure 71: Overall design system for soft matter computing. Performance requirements are taken into consideration to identify the appropriate polymer chemistry, production methods and conditions, component, and system design through machine learning and multi-scale simulations.

We develop a computational design framework integrating data (from experiments and simulations) from all the relevant scales, as shown in Figure 71. The key parts of the design method revolve around integrating the results from experiments and multi-scale simulations. By integrating these methods, inverse and generative design tools can be used to develop a system to meet the product requirements. To develop this workflow, we investigated different simulations methodologies at each length scale and relevant machine learning techniques to link the different length scales and perform inverse design. The length scales and relevant technologies investigated follow.

Atomistic (molecule) to microstructure

At the lowest length scale, the chemical structure of the molecules controls the inherent ability of the material to behave as a semiconductor. However, the combination of the chemical structure and the processing conditions influence the molecular packing and resulting microstructure. Together, these two factors dictate the material performance.

Methods for inverse molecular design have been significantly improving over the last 3 years, enabling exploration of large chemical spaces, which enables design at the molecular level to meet required performance characteristics when large amounts of training data is available. However, for new molecules experimental data to train these inverse design methods on can be limited. Simulations allow us to supplement the available training data and enable these inverse design methods.

Simulation of these length scales typically require 3 different levels of simulation to account for the relevant interactions. Quantum, atomistic, and coarse-grained simulations would need to be combined.

- On the order of up to a hundred atoms, quantum level simulations (such as density-functional theory (DFT)) to determine the material characteristics such as the electron conductivity and band gap at the atomistic level.
- Atomistic level simulations account for molecule-molecule interactions; however, these simulations are informed by forcefields with fitted parameters. Ab initio molecular simulations combine the quantum and atomistic methods in a computationally intensive manner. Recent advances in machine learned force fields enable significant speedup of the simulations.
- To determine the microstructure, coarse-grained molecular simulations would be required to determine the packing and behavior of these materials on the order of hundreds of molecules. There are still various considerations required to integrate these levels of simulations and currently results generally provide a qualitative trend matching experimental results.

Multi-fidelity modeling, however, enables the combination of simulation and experimental results with different degrees of accuracy to produce a surrogate model that is more accurate than either source of data on its own.

As an example of molecular simulations used to determine polymer conductivity evaluations, we simulated different polymer chemistries to determine the ion conductivity. As a note, ion conductivity follows a different mechanism as compared to electrical conductivity in polymers, these simulations demonstrate the strengths and weaknesses inherent in atomistic-only simulations. Additionally, ion conductivity requires consideration of quantum calculations to be quantitatively accurate, the qualitative trends can usually be determined from atomistic and coarse-grained molecular simulations with readily available force fields, if the relevant interactions are captured for the atoms used in the simulations.

In this example, we consider the conductivity of chlorine ions in two polymers, namely: Poly (Ethylene Adipate) and Poly (1,6-hexamethylene adipate). The simulations were implemented in the Culgi software using the Optimized Potentials for Liquid Simulations All Atom (OPLS-AA) force field. To determine the ion conductivity, the simulations were performed at room temperature and the diffusion coefficients of the chlorine ions were determined from the mean squared displacement of the ions, a visual of the simulation is shown in Figure 72. The conductivity, σ , of the ions can then be determined using the Einstein relation:

$$\sigma = \frac{N e^2}{V k_B T} D$$

where, N is the number of ions, e is the elementary charge, D is the diffusion coefficient, V is the simulation volume, k_B is the Boltzmann constant, and T is the simulation temperature. The simulation results indicate that the conductivity for chlorine in Poly (Ethylene Adipate) is 18.9 S/m and in Poly (1,6-hexamethylene adipate) is 19.3 S/m. The conductivities do not quantitatively match the experimental values from reference [85], however they do match the trend, where the ion conductivity of Poly (1,6-hexamethylene adipate) is slightly higher than that of Poly (Ethylene Adipate).

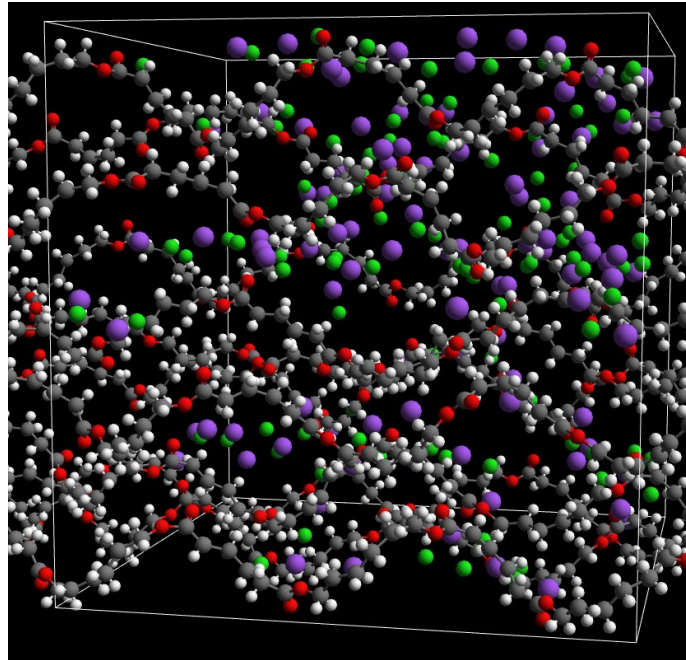


Figure 72: A step in the molecular simulation of Poly (1,6-hexamethylene adipate) to determine the ion conductivity.

Microstructure to transistor design

Once the characteristics of the materials have been determined, the materials are linked together to form a transistor. For example, with traditional transistors the shape and thickness of the different materials as well as the connectivity of the different materials (such as the gate, drain, and source materials) can be considered.

This would require building continuum simulation models based on the material characteristics and verifying the simulation performance. If the transistor is small enough (nm), molecular simulations could be used to evaluate the performance. Building a surrogate model would allow for quick evaluation of the transistor performance and link into the system design.

System design

At this level, the overall requirements are directly considered, such as designing a full adder using minimal material and cost. To design at this stage, known components could be used (such as existing transistors) with generative methods to develop the arrangement of the components in the full system. These systems can then be simulated using Simulation Program with Integrated Circuit Emphasis (SPICE) based simulators to evaluate the system performance.

Alternatively, the transistor current-voltage characteristics could be kept as a design variable and a system could be designed to minimize the number of transistors. Simulations and optimization can then be performed at the transistor-level to meet the required voltage-current characteristics. By using fully differentiable machine learning techniques (neural networks) in the methods at lower length scales and to link results between the length scales, inverse design techniques such as gradient-based optimization can be used to determine the material inputs such as the molecular

structure, processing conditions, and other system details to meet required material properties. This allows for multiscale design of the transistor.

During this study, we investigated the SPICE-based simulation of OFETs. As an example, to demonstrate this level of simulation we utilize the design of a full adder. The full adder forms a basic computing block consisting of numerous transistors (for example, in Figure 73, the implementation of the full adder is made up of 18 Metal Oxide Semiconductor Field Effect Transistor (MOSFETs). This implementation of the full adder forms the baseline analysis that provides the requirements to be achieved by organic material. We implemented this full adder in LTSpice, an open-source SPICE-based simulator, and executed a transient simulation over a 160 μs period.

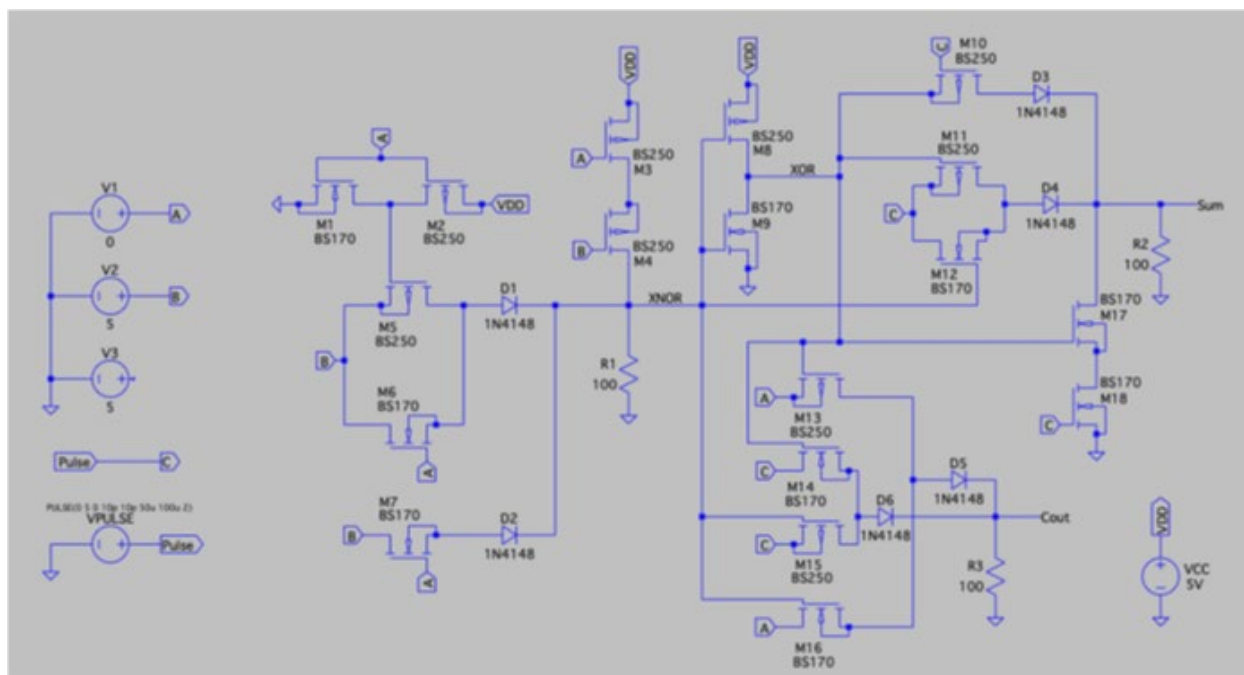


Figure 73: A schematic of the 18 MOSFET implementation of the full adder

The results of the simulation are illustrated in Figure 74, and these provide the requirements for the organic material that is to be used to replace either the MOSFETs themselves, or the entire full adder assembly. The requirements are in the form of the transient response time, which is required to be under 5 ns, i.e., 200 MHz operating frequency, and a voltage drop of under 0.2 V.

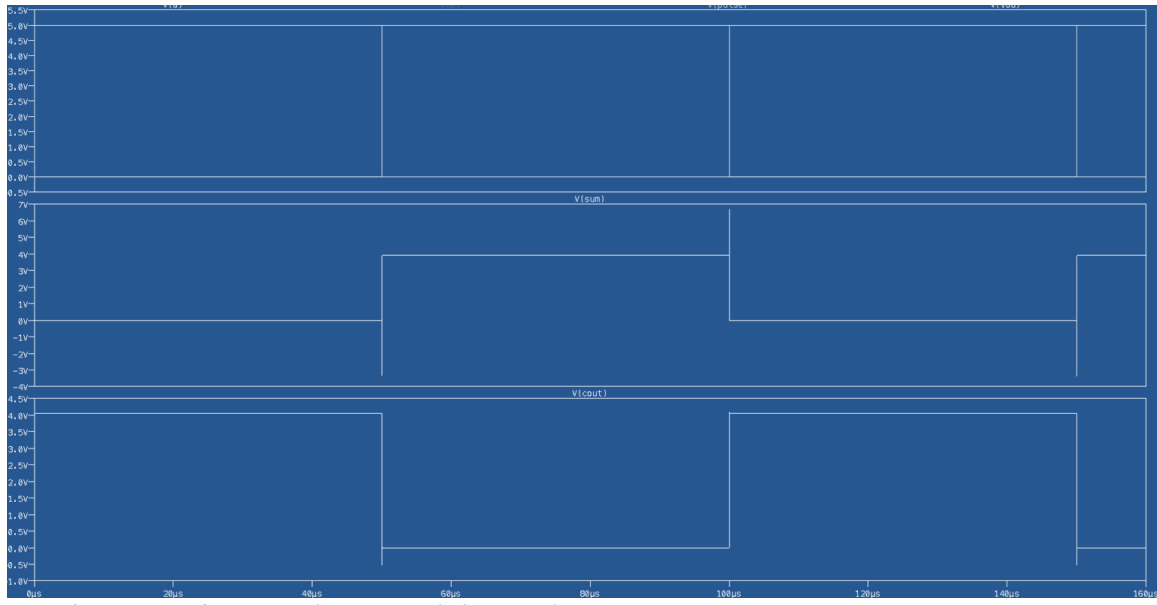


Figure 74: Results of the transient analysis illustrating the output voltages measured across the three output pins of the full adder.

Additionally, the simulations do not necessarily need to be run to properly determine all the relevant quantities. If the key behaviors are captured and other relevant properties can be determined much more efficiently, these could be used as inputs to a multi-scale surrogate model where inputs are built on each of the levels of simulation.

3.16.1.3 Conclusions

We develop a design platform for soft matter computing. This platform considers inputs from simulations and experiments at all the relevant length scales through machine learning techniques to integrate the data and provide generative and inverse design tools. In this report, select simulations have been demonstrated and other relevant simulation methodologies have been discussed. Overall, while some of the simulation methodologies still require some development, all the building blocks for the platform are feasible. However, building this platform in practice would require a mixture of experts to perform experiments and simulations at all the relevant scales. The largest gaps which require development are in the manufacturing challenges and the simulation of the microstructure, as simulating the microstructure with the necessary fidelity is computationally intensive and challenging to verify.

The potential future work to build the design platform, includes developing an interface that allows experimental data and simulation data to be combined and then targeting the most important scales to perform simulations and experiments for initial tests.

3.16.2 Real-time Hydrodynamic Simulations with Lattice Boltzmann Method

3.16.2.1 Motivation for the Lattice Boltzmann Method

The Lattice Boltzmann Method (LBM) is a powerful computational simulation method for fluid dynamics problems. It has gained rapid growing attentions as an emerging fast simulation method. Unlike classical CFD methods, such as finite elements and finite volume methods, the LBM simulates a fluid density on a lattice with streaming and collision operators [86].

More specifically, instead of solving conservation equations of macroscopic quantities such as mass, momentum and energy through Navier-Stokes equations, the LBM simulates fluid consisting of fictitious particles which experience consecutive propagation and collision processes over a discrete lattice. Because of local dynamic nature of particles, the LBM possess several computational advantages over CFD methods using conventional PDE solvers. Those advantages include: 1) extremely parallelizable, and 2) flexible in dealing with complex boundaries and incorporating microscopic interactions. Attracted by those above-mentioned advantages, we implement an LBM for real-time evaluation of flow Behaviors. The features of our LBM implementation include: 1) we implement a voxel approach to parameterize solid objects with any geometry; 2) we implement a bounce-back algorithm to handle non-slip boundary condition between fluids and solids; and 3) we implemented a multi-frequency relaxation approach which is more robust especially for fluids with higher Reynolds numbers.

3.16.2.2 Methodology

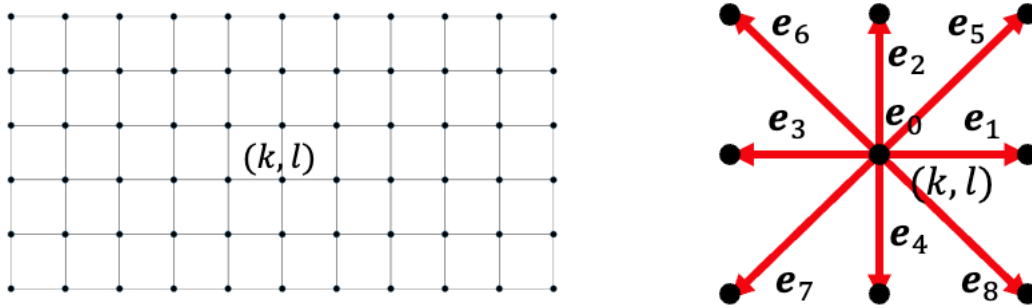


Figure 75: Illustration of a uniform lattice for LBM. For each lattice node, 9 velocity vectors are considered in D2Q9 model of LBM.

In this section, the LBM is described for 2D fluid problems. Here, we use the D2Q9 model of LBM, which is one of the most popular 2D LBM models, as illustrated in Figure 75. In this model, the simulation domain is divided into uniform square lattices and the velocity space is discretized in 9 velocity vectors, $e_i, i = 1, \dots, 9$, as follows:

$$e_i = \begin{cases} [0,0] & i = 0 \\ [\cos\left(\frac{(i-1)\pi}{2}\right), \sin\left(\frac{(i-1)\pi}{2}\right)] & i = 1,2,3,4 \\ \left[\cos\left(\frac{(2i-9)\pi}{4}\right), \sin\left(\frac{(2i-9)\pi}{4}\right)\right] \sqrt{2} & i = 5,6,7,8 \end{cases} \quad [38]$$

Based on the Chapman-Enskog expansion of the Boltzmann equation, we can obtain an evolution relation for every distribution function associated with the velocity vector as:

$$f_i(x + e_i, t + \Delta t) = f_i(x, t) + \Omega_{col}, \quad i = 1, \dots, 9 \quad [39]$$

where f_i is the probability distribution function associated with the velocity vector e_i , x is the position of the local lattice, Δt is the time step and Ω_{col} is the collision operator. The most popular collision operator in LBM is the Bhatnagar-Gross-Krook (BGK) operator [87], which defines Ω_{col} as:

$$\Omega_{col}(x, t) = \frac{\Delta t}{\tau} (f_i^{eq}(x, t) - f_i(x, t)) \quad [40]$$

where f_i^{eq} is the equilibrium distribution for velocity vector e_i , which is given by the following equation:

$$f_i^{eq}(x, t) = w_i \rho \left(1 + 3e_i \cdot u + \frac{9(e_i \cdot u)^2}{2} - \frac{3u \cdot u}{2} \right), \quad i = 1, \dots, 9 \quad [41]$$

where w_i are the weights for velocity vectors such that $w_0 = \frac{4}{9}$, $w_i = \frac{1}{9}$ for $i = 1$ to 4 and $w_i = \frac{1}{36}$ for $i = 5$ to 8.

Once we obtain the distribution functions for each velocity vectors, the macroscopic quantities for the fluid responses are obtained as

$$\begin{aligned} \rho &= \sum_i f_i(x, t) \quad \text{and} \\ \rho u &= \sum_i f_i(x, t) e_i \end{aligned} \quad [42]$$

Multiple relaxation time collision operator

Although being the most popular collision operator in LBM literature, the BGK collision operator assumes that the collision is only dependent on a dimensionless relaxation time parameter τ which is typically taken as $\tau = 3\nu + 0.5$ where ν is the LBM viscosity. However, it is observed that this operator become less robust when Reynold number becomes high. Thus, the multiple relaxation time (MRT) collision operator [88] was proposed and is adopted on our implementation.

In the MRT scheme, we define the collision operator as:

$$\Omega_{col}(x, t) = M^{-1} S (m^{eq} - m) \quad [43]$$

where $m = Mf$ with M being a transformation matrix to project the probability density distribution function f_i to velocity moments and $f = [f_1, f_2, \dots, f_9]^T$ is the vector of distribution functions. Notice that now Ω_{col} becomes a vector whose i th component corresponds the collision operator for velocity vector e_i . In sequence, the components of m are the density, energy, energy square, x component of momentum, x component of energy flux, y component of momentum, y component of energy flux, diagonal and off-diagonal component of the stress tensor.

The vector m^{eq} is the equilibrium velocity moments whose components are defined as:

$$\begin{aligned} m_0^{eq} &= \rho, & m_1^{eq} &= \rho(-2 + 3u \cdot u), & m_2^{eq} &= \rho(1 - 3u \cdot u) \\ m_3^{eq} &= \rho u_x, & m_4^{eq} &= -\rho u_x, & m_5^{eq} &= \rho u_y, \\ m_6^{eq} &= -\rho u_y, & m_7^{eq} &= \rho(u_x^2 - u_y^2), & m_8^{eq} &= \rho u_x u_y. \end{aligned} \quad [44]$$

The transformation matrix M is given by:

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -4 & -1 & -1 & -1 & -1 & 2 & 2 & 2 & 2 \\ 4 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & -2 & 0 & 2 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & 1 \\ 0 & 0 & -2 & 0 & 2 & 1 & 1 & -1 & -1 \\ 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \end{bmatrix},$$

and matrix S is a diagonal one containing the relaxation time for each component of m . In sequence, its diagonal components are taken as 0, 1.4, 1.5, 0, 1.6, 0, 1.6, τ and τ . Notice that if we take matrix S to be τI , where I is the identity matrix, the MRT scheme will reduce to the BGK scheme.

3.16.2.3 Algorithm Implementation

Below, we describe the implemented LBM algorithm. Since the LBM is an explicit scheme, it does not involve inversion of linear system, which render its superior parallelization capability. Assume we are already at time t_n , we already obtain the probability distribution for each velocity vector at each lattice. In order to obtain the solutions in the next step, we perform the following operations:

- 1) Perform collision operation at each lattice node x_k as:

$$f_k(x_k, t_n + \Delta t) = f_i(x_i, t_n) + \Delta t M^{-1} S [m^{eq}(x_k, t_n) - m(x_k, t_n)], \quad i = 1, \dots, 9 \quad [45]$$

- 2) Apply bounce back boundary condition on the fluid-solid interface to impose non-slip velocity boundary conditions.
- 3) Stream the density to the neighboring lattice nodes as:
- 4)

$$f_i(x_k + e_i, t_n + \Delta t) = f_i(x_k, t_n + \Delta t), \quad i = 1, \dots, 9 \quad [46]$$

- 5) Update macroscopic quantities:

$$\rho(x_k, t_n + \Delta t) = \sum_i f_i(x_k, t_n + \Delta t) \quad [47]$$

and

$$u(x_k, t_n + \Delta t) = \frac{\sum_i f_i(x_k, t_n + \Delta t) e_i}{\rho(x_k, t_n + \Delta t)} \quad [48]$$

- 6) Impose Dirichlet and Neumann boundary conditions on the lattice boundary.

This completes one time step of LBM operations to arrive time step t_{n+1} .

3.16.2.4 Geometric Representation

Since the focus of LBM simulation is to model how fluid flows through solids, there is a need to also represent solid regions in the simulation domain. In the implementation, this is done by using the signed distance functions. The basic idea is to define the solids through the constructive solid geometry approach where the signed distance function of a complex solid is defined as logic operations (e.g., union, intersection, and difference) between the signed distance functions of a set of

basic geometric primitives (e.g., circle, rectangle, lines). Since we have analytical expressions of those basic primitives, this approach can efficiently define the signed distance functions of very general geometries. The details on how to construct any 2D objects from primitives and the expressions of the signed distance functions for several basic primitives are discussed in reference [89].

Once we compute the signed distance function that describes the solid geometry, we evaluate this signed distance function on every node in the lattice. Those nodes which return negative signed distances are marked as solid nodes. During LBM simulation, we will then apply the bounce-back algorithm [90] after the collision operator every time step. After the bounce back algorithm is applied, we then perform the stream operator and continue the algorithm described in the Algorithm Implementation section.

In addition, we also note that the geometry description of solid can also be passed from topology optimization results. Due to the voxel nature of the topology optimization algorithm, we can simply map the voxel description of geometry from topology optimization to the lattice domain of the LBM. Since both topology optimization and LBM are operated on regular grids, this mapping process is quite trivially handled in the current implementation.

3.16.2.5 Results

In this section, we demonstrate the performance of the implemented LBM algorithm in simulating the flow behavior when passing through objects with various geometries. The simulation domain is defined as shown in Figure 76, where a solid obstacle is placed in the middle and the fluid flows enter via the left side and exit the domain on the right side. On the inflow side, uniform velocity profile is prescribed as $[u_0, 0]^T$ and, on the outflow side, we set the gradient of the velocity field to be zero. On the top and bottom of the simulation domain, non-slip boundary conditions are prescribed so that the velocity is 0.

The physical dimensions of the numerical simulation are set as follows: the domain size is 4m by 1m, the inflow velocity is 1m/s, and the fluid viscosity is 1e-3 Pa/s. The LBM uses dimensionless units with an 801 x 201 lattice and $\Delta x_{LBM} = \Delta t_{LBM} = 1$. This gives the lattice viscosity $\nu_{LBM} = 0.02$ and physical times step $\Delta t_{physical} = 5 \times 10^{-4}$ s.

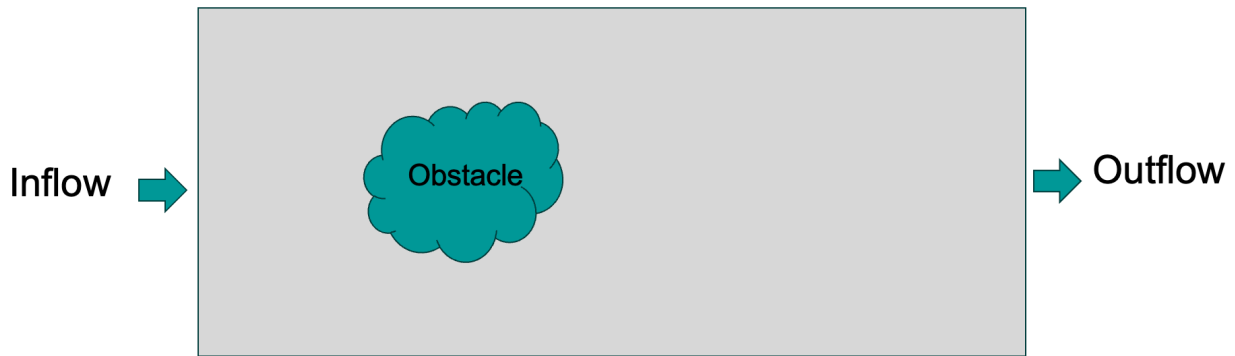


Figure 76: Illustration of the simulation domain and setup

To demonstrate the capability of the LBM implementation in handling a wide variety of obstacle geometries, this example, demonstrated three varieties, as shown in Figure 77. For all the cases, the signed distance function can be generated very easily following the procedure described above.

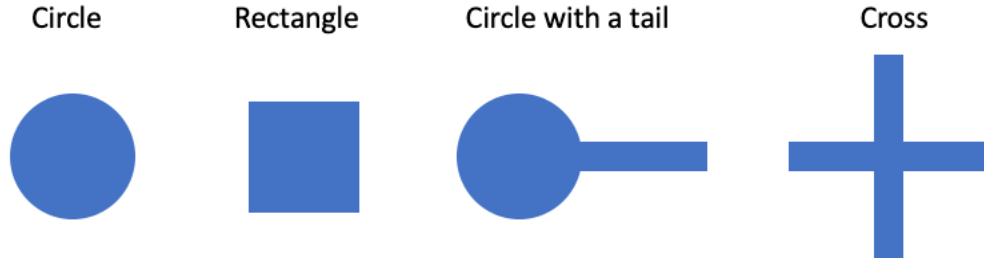


Figure 77: List of solid shapes considered in this example.

Figure 78 plots the results we obtained for the four solids respectively. For each case, the top row shows the velocity norm distribution at physical time $t = 45s$ and the bottom row shows the vorticity (i.e., $\frac{\partial u_x}{\partial y} - \frac{\partial u_y}{\partial x}$). From the results shown, the vortex behind the objects are well captured by LBM for all the cases. Notice that since LBM uses a fully explicit solver, it can be very efficiency parallelized in graphics processing unit (GPU) without requiring any linear systems.

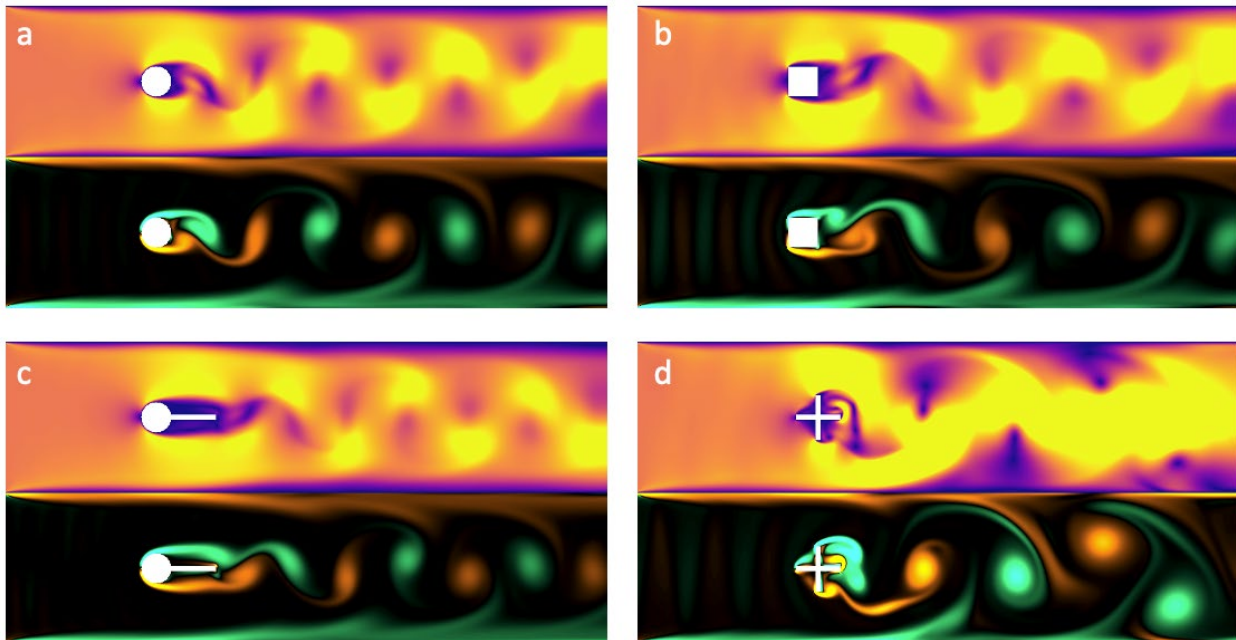


Figure 78: Velocity magnitude and vorticity distribution of all the cases at $t=45s$ obtained by LBM.

We implemented an LBM for fast simulation of fluid responses over solids of complex geometries which are represented by signed distance representations. With a GPU implementation, we can

achieve close-to-real time performance on an 800 x 200 lattice for turbulent fluid simulations. Our results show the promising potential of LBM as a fast fluid solver to quickly evaluate the fluid responses in engineering design.

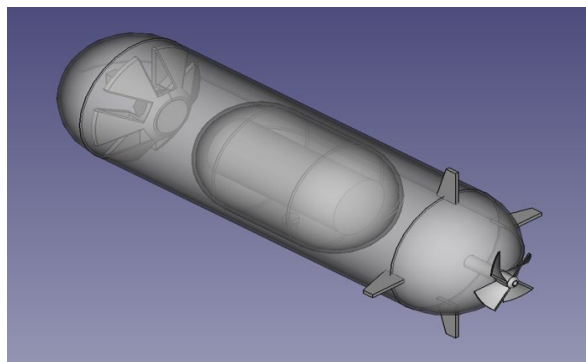
Accuracy and Efficiency LBM compared with other solvers.

In terms of accuracy, various studies have investigated the LBM and show it is able to achieve similar accuracy to the mainstream computational CFD solvers such as Finite Volume Method (FVM) and Finite Element Method (FEM). In terms of efficiency, the main advantage of LBM lies in its superior friendliness in parallel and GPU implementation. For example, reference [91] compares a GPU implementation of LBM and shows that it has 17 and 200 times smaller “seconds elapsed per core and million cells” (SPCMC) score [92] than Reynolds Averaged Modeling (RANS) and Discrete-Event Simulation (DES) simulators from commercial software, respectively. A smaller SPCMC score indicates a better efficiency since it is directly proportional to the computational time elapsed during the simulation. Furthermore, the SPCMC score decreases when the number of cells and/or the number of computational cores increases. As another example, for Simulating Solid-Liquid phase change problems with natural convection, [93] compares the performance between LBM and FVM in commercial software Fluent and shows that both methods deliver similar accuracy, but LBM is able to achieve 32x speed up even with a smaller time step and simpler Matlab implementation.

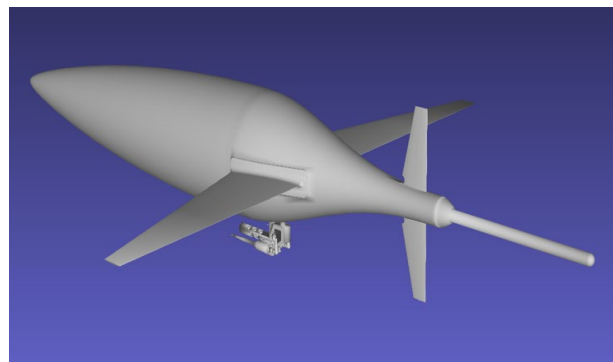
3.16.3 A Six Degree of Freedom Underwater Vehicle Simulator

3.16.3.1 Introduction

Unmanned/Autonomous Underwater Vehicles are increasingly popular in civilian and defense operations in complex and dangerous missions. General applications include oceanographic surveys [94, 95], exploration and extraction programs in the oceans, mine detection in military operations, etc. Many UUVs are designed for autonomous operations, which can dive from tens to hundreds of meters depending on the challenge of the mission. Furthermore, these vehicles are comparatively low-cost technology for underwater exploration.



(a)



(b)

Figure 79: (a). A cylinder-shaped UUV. (b). A glider-type UUV.

Figure 79 shows typical cylinder-shaped and glider-type UUVs designed in this work. Generally, a logistically and economically challenging problem is often handled by surface ships carrying large sonar arrays with adequate power supply and other amenities to facilitate smooth ocean surface bed scanning, documentation, and logging operations. The challenge problem involves designing a low-cost fleet of UUVs that includes the bottom and sub-bottom sonar scanning array. In this work, the conceptual UUV models produced by the generative design framework are tested and validated for their behavior under various motions, control strategies, and stability requirements to undertake the actual mission.

UUVs require autonomous guidance and control systems to perform underwater tasks unattended with minimal human interaction to execute their mission requirements. A UUV architecture consists of a guidance system to achieve the desired state of the vehicle and a navigation system to estimate the vehicle's current state. Lastly, a control system that calculates and applies correct forces using various actuators on the vehicle externally and internally. These actions minimize differences between desired and current states to guide the vehicle in the desired trajectory even in the presence of unknown disturbances. Disturbances could be external, such as ocean currents, and environmental factors, like temperature, humidity, and pressure changes. In addition, sensor system noises and vibrations are internal to the vehicle. A system architecture, including the various components required to guide the UUV to its desired trajectory successfully, is shown in Figure 80.

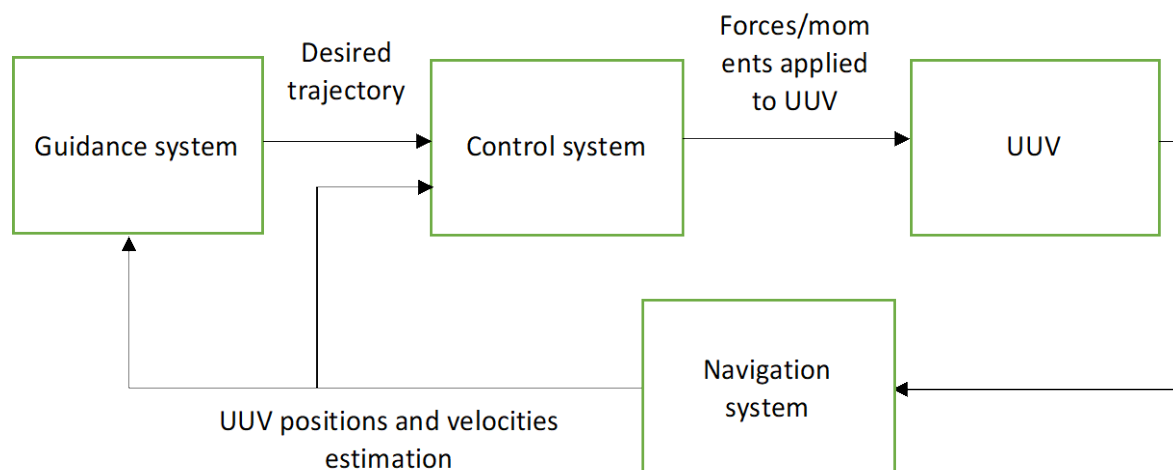


Figure 80: A system architecture of an autonomous UUV.

The guidance system is designed to maneuver the UUV under certain constraints and deliver the vehicle to a desired location. On a UUV, external forces and moments resulting from hydrostatics, hydrodynamic lift and drag, added masses, and the control inputs of the vehicle propeller and fins are all acting. The next section details UUV's kinematics, dynamical equations, and control strategies.

3.16.3.2 Methodology

Kinematics and Dynamical Equations of Motion

Modeling a UUV's motion involves a study of both statics and dynamics. A UUV moving through the oceans has six degrees of freedom (DOF), giving six different components. These six components are defined as surge, sway, heave, roll, pitch, and yaw (see Figure 81). The notations used for the vehicle in this report are based on Fossen [96].

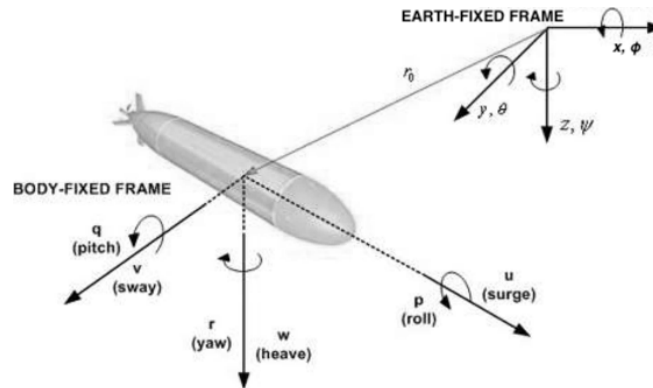


Figure 81: Body reference coordinate system.

Figure 81 shows a UUV's local body reference frame where the equations of motion are derived and a global/final earth-fixed reference frame where the final velocity components are transformed to the North, East, and Depth (NED) coordinates.

Table 5: Notations for six degrees of freedom UUV system.

DOF	Motion and velocities	Forces and moments	Linear & angular velocities	Position & Euler angles
1	motion in the x-direction (surge)	X	u	$X (N)$
2	motion in the y-direction (sway)	Y	v	$Y (E)$
3	motion in the z-direction (heave)	Z	w	$Z (D)$
4	rotation about the x-axis (roll)	K	p	ϕ
5	rotation about the y-axis (pitch)	M	q	θ
6	rotation about the z-axis (yaw)	N	r	ψ

Table 5 summarizes the notations used for the UUV's motion, rotation, velocities, and positions. The first three coordinates and their corresponding time derivatives in Table 5 are the position and translation motions along the x , y , and z -axis. The last three correspond to the orientation angles and rotational motions, respectively. When analyzing the motions, it is convenient to define two coordinate systems: The body-fixed and the earth-fixed reference frames. The body-fixed frame is fixed to the moving vessel in its origin O , often chosen to coincide with the UUV's center of

gravity, CG. The body axes X_0 , Y_0 , and Z_0 are the longitudinal, transverse, and vertical axes, respectively.

Since marine vessels like UUV tend to move in low velocities, it is approximated that the acceleration of a point on the earth can be neglected. This results in the earth-fixed reference frame being inertial. Therefore, the positions and orientations are described relative to the inertial, earth-fixed reference frame, while the linear and angular velocities are described in the body-fixed frame. The velocity transformation matrix then gives the vehicle's traveling path relative to the earth-fixed coordinate system, J . Finally, the earth-fixed velocities will then be transformed according to Equation below.

$$\dot{\eta}J = v$$

where, $v = [u \ v \ w \ p \ q \ r]^T$ is the linear and angular velocities vector, and $\eta = [x \ y \ z \ \phi \ \theta \ \psi]^T$ is the position and rotations vector. $\dot{\eta}$ is the time derivative. J is given by:

$$J = \begin{bmatrix} c(j)c(q) & -s(j)c(f) + c(j)s(q)s(f) & s(j)s(f) + c(j)c(f)s(q) & 0 & 0 & 0 \\ s(j)c(q) & c(j)c(f) + s(j)s(q)s(f) & -c(j)s(f) + s(q)s(j)c(f) & 0 & 0 & 0 \\ -s(q) & c(q)s(f) & c(q)c(f) & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & s(f)t(q) & c(f)t(q) \\ 0 & c(f) & -s(f) \\ 0 & \frac{s(f)}{c(q)} & \frac{c(f)}{c(q)} \end{bmatrix} \quad [49]$$

where $s() = \sin()$, $c() = \cos()$ and $t() = \tan()$. The detailed derivation of the transformation matrix is provided in [96].

The dynamic equation of motion representing the rigid-body dynamics, hydrodynamics (added mass, drag force, and lift), hydrostatics, control forces due to the control surfaces such as rudder and stern planes, and propulsion forces and moments is given by:

$$M\dot{v} + C(v)v + D(v)v + g(\eta) = \tau \quad [50]$$

where $M = M_{RB} + M_A$ (rigid body and added masses), $C = C_{RB}(v) + C_A(v)$ (Coriolis and centripetal terms of rigid body and added masses), $D(v)$ is the hydrodynamics/damping matrix and $g(\eta)$ is the hydrostatic forces due to the weight and buoyancy and τ is the control, the propulsion forces and moments. For the detailed derivation, refer to [96]. M , $C_{RB}(v)$, $C_A(v)$, $D(v)$ and $g(\eta)$ are provided below.

$$M = \begin{bmatrix} m - X_{\ddot{u}} & 0 & -X_{\ddot{w}} & 0 & mz_g - X_{\ddot{g}} & 0 \\ 0 & m - Y_{\ddot{v}} & 0 & -mz_g - Y_{\ddot{p}} & 0 & mx_g - Y_{\ddot{r}} \\ -X_{\ddot{w}} & 0 & m - Z_{\ddot{w}} & 0 & -mx_g - Z_{\ddot{q}} & 0 \\ 0 & mz_g - X_{\ddot{p}} & 0 & I_x - K_{\ddot{p}} & 0 & -I_{zx} - K_{\ddot{r}} \\ mz_g - X_{\ddot{g}} & 0 & -mx_g - Z_{\ddot{q}} & 0 & I_y - M_{\ddot{q}} & 0 \\ 0 & mx_g - Y_{\ddot{r}} & 0 & -I_{zx} - K_{\ddot{r}} & 0 & I_z - N_{\ddot{r}} \end{bmatrix} \quad [51]$$

$$c_{RB}(v) = \begin{bmatrix} 0 & 0 & 0 & m(y_g q + z_g r) & -m(x_g q - w) & -m(x_g r + v) \\ 0 & 0 & 0 & -m(y_g p + w) & m(z_g r + x_g p) & -m(y_g r - u) \\ 0 & 0 & 0 & -m(z_g p - v) & -m(z_g q + u) & m(x_g p + y_g q) \\ -m(y_g q + z_g r) & m(y_g p + w) & m(z_g p - v) & 0 & -I_{yz}q - I_{xz}p + I_z r & I_{yz}r + I_{xy}p - I_y q \\ m(x_g q - w) & -m(z_g r + x_g p) & m(z_g q + u) & I_{yz}q + I_{xz}p - I_z r & 0 & -I_{xz}r - I_{xy}q + I_x p \\ m(x_g r + v) & m(y_g r - u) & -m(x_g p + y_g q) & -I_{yz}r - I_{xy}p + I_y q & I_{xz}r + I_{xy}q - I_x p & 0 \end{bmatrix} \quad [52]$$

$$C_A(v) = \begin{bmatrix} 0 & 0 & 0 & 0 & -Z_{\dot{w}}w & Y_{\dot{v}}v \\ 0 & 0 & 0 & Z_{\dot{w}}w & 0 & -X_{\dot{u}}u \\ 0 & 0 & 0 & -Y_{\dot{v}}v & X_{\dot{u}}u & 0 \\ 0 & -Z_{\dot{w}}w & Y_{\dot{v}}v & 0 & -N_{\dot{r}}r & M_{\dot{q}}q \\ Z_{\dot{w}}w & 0 & -X_{\dot{u}}u & N_{\dot{r}}r & 0 & -K_{\dot{p}}p \\ -Y_{\dot{v}}v & X_{\dot{u}}u & 0 & -M_{\dot{q}}q & K_{\dot{p}}p & 0 \end{bmatrix} \quad [53]$$

$$D(v) = -\text{diag}\{X_u, Y_v, Z_w, K_p, M_q, N_r\} \\ -\text{diag}\{X_{|u|}u, Y_{|v|}v, Z_{|w|}w, K_{|p|}p, M_{|q|}q, N_{|r|}r\} \quad [54]$$

$$g(h) = \begin{bmatrix} (W - B) \sin(\theta) \\ -(W - B) \cos(\theta) \sin(\varphi) \\ -(W - B) \cos(\theta) \cos(\varphi) \\ -(y_g W - y_b B) \cos(\theta) \cos(\varphi) + (z_g W - z_b B) \cos(\theta) \sin(\varphi) \\ (z_g W - z_b B) \sin(\theta) + (x_g W - x_b B) \cos(\theta) \cos(\varphi) \\ -(x_g W - x_b B) \cos(\theta) \sin(\varphi) - (y_g W - y_b B) \sin(\theta) \end{bmatrix} \quad [55]$$

where $r_g = [x_g, y_g, z_g]$ and $r_b = [x_b, y_b, z_b]$ are the distance of the center of gravity and center of buoyancy to the coordinate system of the UUV. W is the weight of the UUV and B is the buoyancy force of the UUV. m is the mass of the UUV. $X_{\dot{u}}$, $Y_{\dot{v}}$, $Z_{\dot{w}}$ are the added mass cross-terms which are estimated using the strip theory. After constructing these matrices and vectors, time marching is performed using any time integration methods. This study uses Euler's time integration method for time-related computations.

Control System

UUV motion control is required to keep it in a trajectory or pre-defined course. In this study, the UUV is controlled in depth and heading using the Proportional, Integral, and Derivative (PID) control laws. To simplify the equations governing the control laws, without the loss of generality, the UUV is assumed to be a linear mass-damper-spring model with dampening over time. Consider the two equivalent systems of the mass-damper-spring system:

$$m\ddot{x} + d\dot{x} + kx = 0. \quad [56]$$

$$x + 2\zeta\omega_n\dot{x} + \omega_n^2 x = 0 \quad [57]$$

where ω_n is the natural frequency, and ζ is the relative damping ratio, and both are greater than zero. These quantities directly affect the PID controller gain values as:

$$\omega_n = \frac{1}{\sqrt{1 - 2\zeta^2 + \sqrt{4\zeta^4 - 4\zeta^2 + 2}}} \omega_b \quad [58]$$

where ω_n is the bandwidth frequency and positive. PID gains are given by:

$$K_p = (m + K_m)\omega_n^2 - k \quad [59]$$

$$K_d = 2\zeta\omega_n(m + K_m) - d \quad [60]$$

$$K_i = \frac{\omega_n}{10} K_p \quad [61]$$

where K_m is the optional mass gain. The above equations are computed using the pole placement of the algorithm by specifying ω_n and ζ . k and d are the stiffness and damping coefficient of the mass-damper-spring system and are considered as the values close to zero. Lastly, the PID controller equation is given as,

$$\tau = K_p \tilde{x} + K_d \dot{\tilde{x}} + K_i \int_0^t \tilde{x}(\tau) d\tau \quad [62]$$

with gains $K_p > 0$, $K_d > 0$ and $K_i > 0$ and tracking error $\tilde{x} = x - x_d$. For more details on the derivation, refer to [97]. In this study, for the depth control and heading, PD and PID control laws are employed with Euler's integration to perform the time-marching.

3.16.3.3 Results

In this section, the simulation results of the UUV are presented. The mass and maximum diameter of the UUV is 2677.44 kg and 1.1 m. The area of the fins is 5.29E-3 m², and the fin distance from C0 is 0.8 meters. Table 6 presents the tuned control parameters responsible for steering the UUV to the desired depth and heading. Due to the mass of the UUV, the stiffness of the external forces, such as wave disturbances, is larger. Any larger values of the controller parameters will slow down the UUV due to its induced negation forces. In addition, the larger controller gain values and frequencies will produce repetitive artifacts in the command/desired path of the UUV. Therefore, the gain parameters selected are relatively small in this simulation.

Table 6: Tuned control parameters.

Heading autopilot	Depth autopilot
$\omega_{n_\psi} = 0.004$ (natural frequency)	$\omega_{n_{dz}} = 0.05$ (desired natural frequency, reference model)
$\zeta_\psi = 0.25$ (relative damping ratio)	$K_p = 0.01$ (heave proportional gain)
$r_{max} = 1$ radian/sec (max. yaw rate)	$T_z = 100$ (heave integral gain)
$\psi_d = 0$ (initial position)	$K_{p_\theta} = 0.01$ (<i>pitch prop. gain</i>)
$\dot{\psi}_d = 0$ (initial velocity)	$K_{d_\theta} = 0.2$ (<i>pitch deri. gain</i>)
$\ddot{\psi}_d = 0$ (initial acceleration)	$K_{i_\theta} = 0.001$ (<i>pitch int. gain</i>)
$\omega_{n_d} = \omega_{n_\psi} / 5$ (desired natural frequency)	
$\zeta_d = 1$ (desired relative damping ratio)	

Figure 82-83 show the vehicle states, control surface trajectories due to the pre-defined heading/depth autopilot and step inputs and autopilot trajectory in NED coordinates, respectively. In Figure 82, the surge, heave, and sway velocities are lower. This is due to the mass of the UUV, and the smaller propeller parameters used in this work. Furthermore, the depth trajectory is slightly higher than the desired depth value of 30 m. Because, of the mass, the braking effort of the UUV has a smaller magnitude. Figure 83 and Figure 84 show the control trajectories match well with the desired command. Lastly, Figure 85 shows the trajectory of the UUV; it is slightly over 30 m due to the above-mentioned reason.

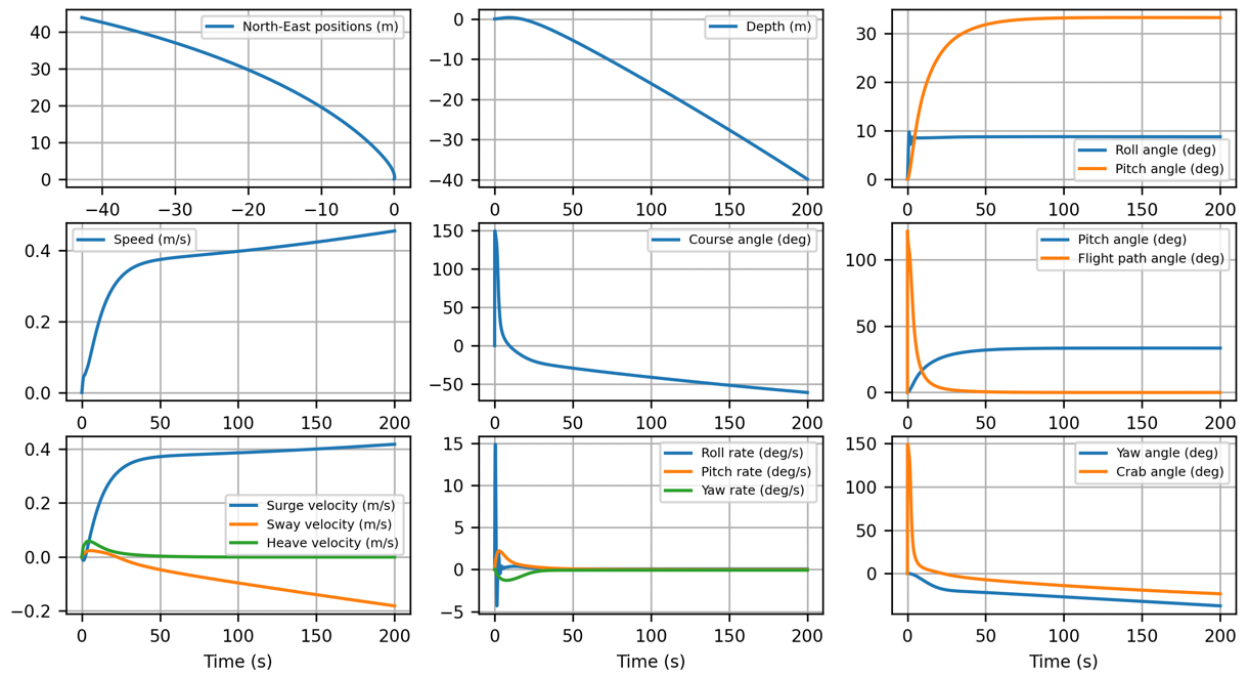


Figure 82: Various vehicle states during the simulation.

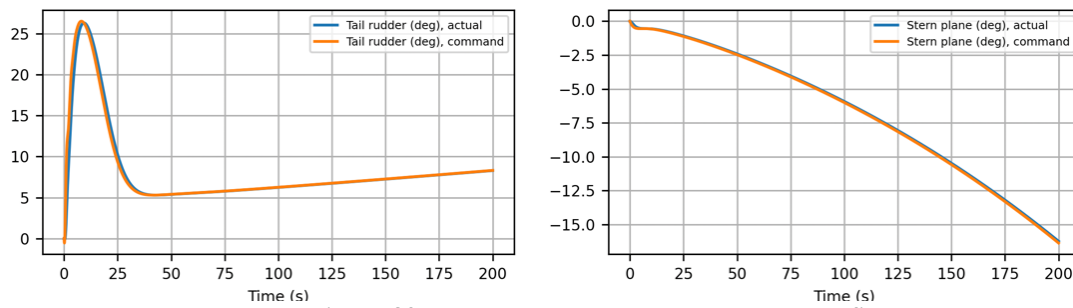


Figure 83: Rudder and stern planes control profile.

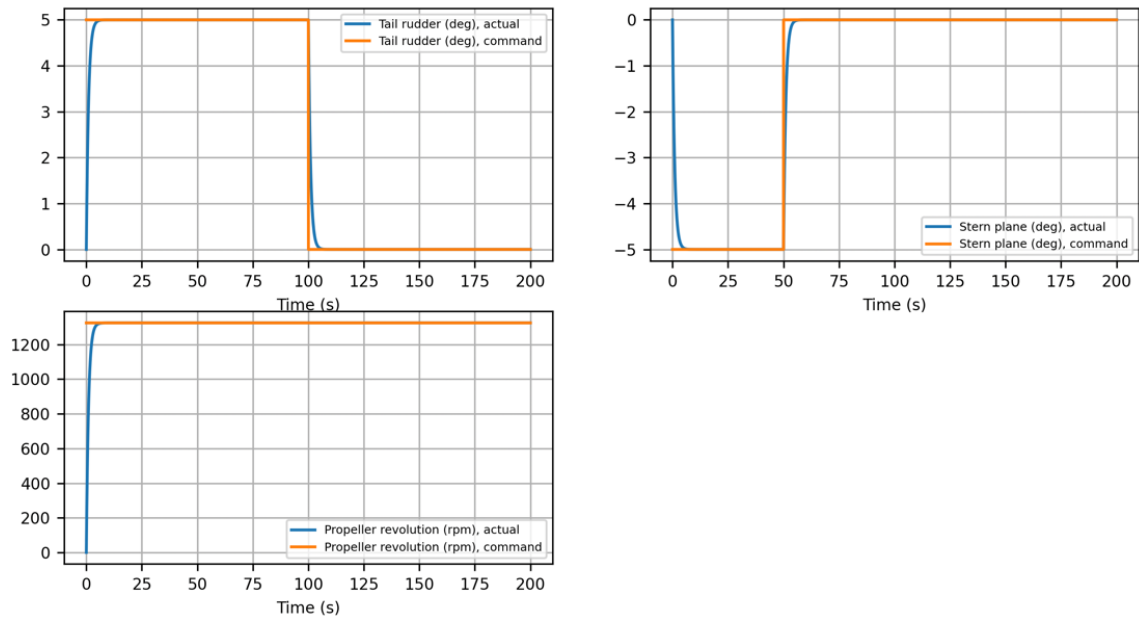


Figure 84: Step input rudder and stern plane control outputs.

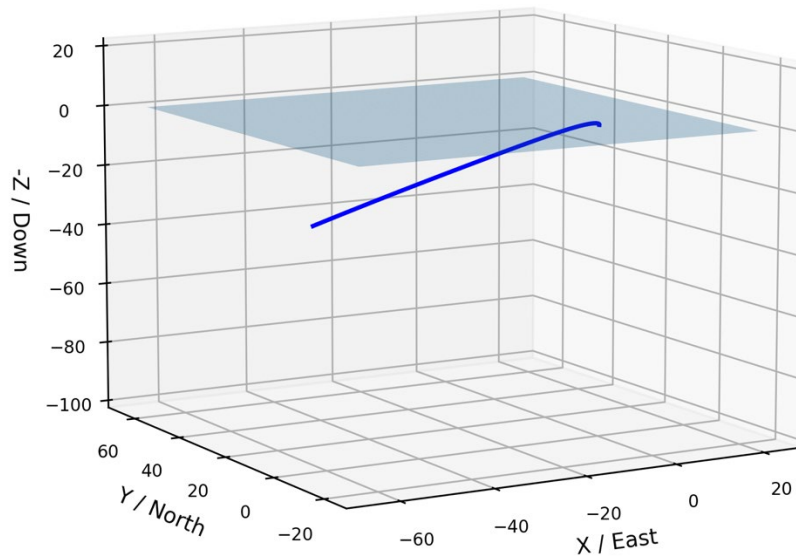


Figure 85: UUV trajectory over time. The blue plane refers to the surface water level, and the dark blue curve is the path trajectory.

4.0 RESULTS AND DISCUSSION

This section introduces the challenge problems to which the team has applied the Systemic Generative Engineering process and framework. Please note that the description of the challenge problems may be taken from the documentation shared by the TA3 teams to ensure an accurate representation. For each challenge problem that was completed, we highlight the details of the methodology employed to generate the results, the results that were achieved and potential areas for improvement in the AI tools. Table 7 summarizes the challenge problems considered and their status.

Table 7: A summary of the challenge problems considered in the program

Id	Title	Description	Organizer
1	Design of a UAV	Design of an unmanned aerial vehicle to optimally traverse a set of predefined paths	SwRI
2	Design of multiple UUVs	Design of a transit and sensing vehicle that measure and transmit sensing data in the arctic region	STR
3	Design of a UAM vehicle	Design of an urban air mobility vehicle capable of carrying two passengers and optimally traverse the predefined path	SwRI
4	Design of a propeller driven UUV	Design of a propeller driven unmanned underwater vehicle to carry and deploy an ocean bottom seismometer	STR
5	Design of a UAV (Phase 2)	Design of an unmanned aerial vehicle to optimally traverse a flight path carrying and deploying a payload.	SwRI
6	Design of a propeller driven UUV (Phase 2)	Design of a propeller driven UUV to carry and deploy an ocean bottom seismometer incorporating additional disciplines in the design consideration	STR
7	Design of 3D DNN Accelerator	Design a hardware accelerator to run a data processing algorithm at the edge	Siemens

4.1 Design of an Unmanned Aerial Vehicle

4.1.1 Problem Summary

The SwRI TA3 team organized a design challenge in which the AI co-designers were tasked with designing the optimal UAV to traverse a racetrack inspired by the American Institute of Aeronautics and Astronautics (AIAA) Design Build Fly competition. An illustration of the racetrack is provided in Figure 86. The racetrack is divided into four (4) paths that the UAV must traverse. The paths are a straight flight path of 2 KM (Path 1), a circular turn of 500 m radius (Path 3), a vertical rise and hover (Path 4) and the racing oval (Path 5). The details of the paths are provided below along with the scoring criteria used for each path. The AI co-designers were tasked with using an identical physical design for each track but are permitted to alter the cyber, i.e., control, design parameters.

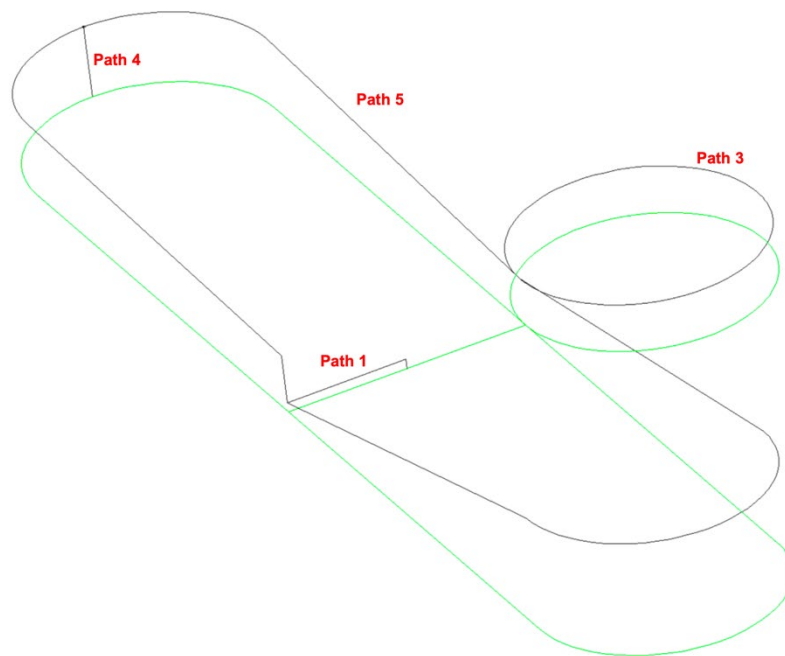


Figure 86: An illustration of the racetrack which serves as the inspiration for the simulation paths which are used to design the optimal UAV.

- **Path 1 (straight line):** A level steady flight of over 2000 meters in a straight line is required to start scoring. Then, the design gets 1 point per 10 meters (including the 2000 meters prior) in flight. A penalty of 10 times the lateral error in meters is introduced to penalize designs that do not exhibit stable flights.
- **Path 3 (circle):** The design is required to traverse a circle of diameter 1000 meters. Upon finishing flying the circle, the design gets 300 points. A penalty of 50 times the lateral error in meters is added for deviations from the circular path.
- **Path 4 (rise and hover):** If the design can hover for 200 seconds and reach a height of 150 meters, it'll score 200 points. Additionally, each extra second the UAV can hover for gets 1 point per second, up to a maximum of 400 points in total. Failing the hover benchmark also means failing the entire suite of following benchmarks. Further, a penalty is introduced when the vehicle violates the 150-meter height requirement. Thus, the vehicle has to be stable at the requested 150-meter height for a duration of at least 400 seconds to score full points on this benchmark.
- **Path 5 (racing oval):** The design is required to fly an oval consisting of two 750-meter straights, and two 600-meter diameter half circles. Finishing the course grants 200 points, and for every second less than 350 seconds taken to complete the course, an extra point is granted. Designs that exhibit poor path following capability are penalized significantly. If the deviation from the planned path exceeds 10 meters, the design gets 0 points.

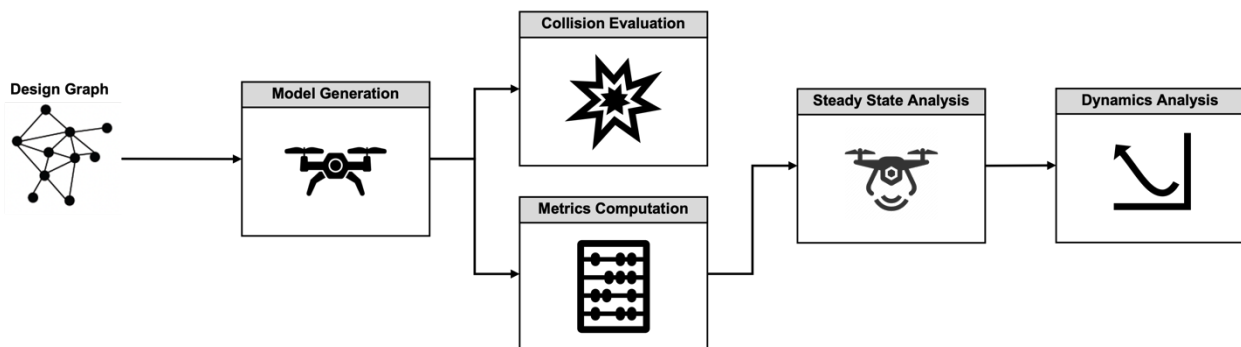
Table 8 presents a summarized view of the different flight paths which are implemented as test benches in OpenMETA by the design challenge organizers.

Table 8: Summary of the flight paths test benches for which the UAV is to be designed for

Type	Specification	Score
Rise and Hover	Vertical flight from 0 to 150 m then hover for a minimum of 200 seconds and a maximum of 400 seconds with minimal deviations from the target 150 m height at the end of 200 seconds.	400
Straight Line	North in a straight line for a minimum of 2000 m and a maximum of 4000 m with minimum deviation from the desired straight-line path.	400
Circle	Traverse a 1 km diameter circle clockwise with minimum deviations from the circular trajectory.	300
Racing Oval	Racking over with 2x 750 m straight paths and 2x 600 m diameter half circles that must be traversed counterclockwise with both minimum deviations from the oval trajectory and as quickly as possible.	481
Total	Sum of all paths	1581

Design Methodology

To validate the behavior of each design generated, we adapt the provided OpenMETA UAV simulator to a custom Python-based implementation. Our custom implementation of the simulation pipeline runs through the relevant racetracks and returns the associated score. The simulation environment takes as input a design graph, and constructs the geometry for the design, evaluates the geometry for collisions, computes the geometric metrics necessary for control simulations, executes a static analysis in which the trim states for the UAV are determined and finally, executes a dynamic simulation in which the control response of the UAV is evaluated. This process is illustrated in Figure 87. The CAD model generation is implemented in open-source CAD application, *FreeCAD*, and the UAV steady state and dynamics analysis for the different paths are implemented in *Fortran*.

**Figure 87: Simulation pipeline used to evaluate the performance of the designs generated by the framework**

As the task at hand is to design the complete UAV, we consider all three aspects of the traditional design process, i.e., the architecture design, preliminary design, and the detailed design. The design space is derived from the corpus supplied by SwRI. These can be given by:

- **Architecture Design:** In the design challenge considered, the architecture design is realized by optimizing for the topology of the design. The topology considers both mechanical and electrical connections that are present in the system architecture of the design. There are no limits placed on the number of components that can be present in a design and the optimizer is tasked with identifying the optimal topology that maximizes the performance in the given design challenge. One restriction that is placed on the design topology is the type of components that can be incorporated into the design. We limit the components to battery, motor, ESC, propellers, flanges, tubes, plates, connection hubs, wings, and servos.
- **Preliminary Design:** The task of preliminary design is to identify optimal selection of off-the-shelf components and their layout. The goal of the preliminary design task is to generate designs that are free from collisions and are optimally placed from the geometric properties' perspective (e.g., optimal moments of inertia, center of gravity, etc.). Several off-the-shelf components are available for the optimizer to select from. In particular, there are 34 batteries, 20 ESCs, 83 motors, 417 propellers, 68 wings, 27 servos, 2 flange connectors and 5 connection hubs.
- **Detailed Design:** As part of the detailed design, we optimize the scalable and continuous parameters of the design. Depending on the number of components selected and the type of component selected, there are a different number of scalable parameters. The parameters include the control parameters for the different segments of the racetrack (5 control parameter segments), and the scalable parameters of the components. The scalable parameters of the components are, 3 offset parameters for the battery, 3 offset parameters for the ESC, 1 length parameter for the carbon fiber tubes, and 1 angle parameter for the connection hub.

Table 9 summarizes the design space in which we can operate and generate new designs.

Table 9: Design space for the UAV system

Subsystem	Component	Type	Size
Propulsion	Battery	Discrete	34
	Battery Position	Continuous	3x # Battery
	ESC	Discrete	20
	ESC Offset	Continuous	3x # ESC
	Motor	Discrete	83
	Propeller	Discrete	417
Aerodynamics	Left Wing	Left Wing	68
	Right Wing	Right Wing	68
	Servo	Discrete	27
Structures	Carbon Fiber Tube	Discrete	1
	Tube Length	Continuous	# Arms
	Flange Connectors	Discrete	2
	Plate Connectors	Discrete	1
	Hub Connectors	Discrete	5
	Connection Angles	Continuous	# Hubs
Controls	Autopilot (LQR) Parameters	Continuous	20

We employ an optimization framework that jointly explores the topology of the design while also exploring the parametric spaces. Instead of directly optimizing the scores across the different paths, we reformulate the problem as being one employing sequential optimization. We first optimize the trim performance of the vehicle to determine the ideal design topology and then subsequently optimize for the control parameters of the design. This process is illustrated in Figure 88. The design topology along with the components that must be selected onboard the vehicle is first optimized using multiple optimization schemes. The optimization schemes evaluated include $(1 + \lambda)$ -EA CMAES optimization and greedy graph structure optimization such as greedy equivalence search. Having identified the best trimming designs, we next optimize for the control of these designs based on the scores reported in each test bench. Here, we use the 2-point differential evolution and hybrid-adaptive optimizer (SHERPA) algorithm.

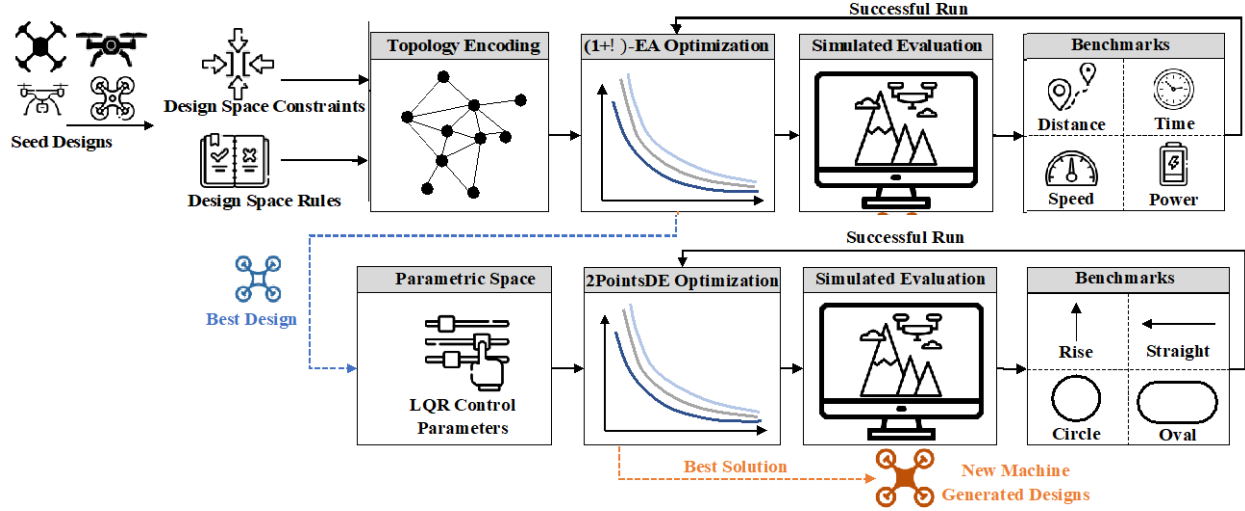


Figure 88: A sequential design process optimizing the design topology and control parameters for the UAV.

As several design topologies can produce high performing designs for the given mission, we execute the optimization process multiple times with different seeds to generate topologically different designs. The optimization formulation for the trim stage of the exploration is given by:

$$\begin{aligned} \text{Objectives: } & \max \left(\frac{t_f}{400} \right), \\ & \max \left(\frac{\min(v_l, v_{t_{300}}, v_{t_{500}})}{50} \right) \end{aligned} \quad [63]$$

Subject to constraints,

$\max \left(\frac{d_s(v_l)}{2000} \right) > 1$	Maximum Flight Distance Constraint Ensures Path 1 performance is satisfactory
$\max \left(\frac{t_f(v_l < 3)}{200} \right) > 1$	Maximum Flight Time Constraint Ensures Path 4 performance is satisfactory
$\max \left(\frac{d_s(v_{t_{300}})}{3142} \right) > 1$	Maximum Flight Distance Constraint Ensures Path 3 performance is satisfactory
$\max \left(\frac{A(v.)}{A_{max}} \right) < 1$	Maximum Battery Current Constraint Ensure battery current is within acceptable limits
$\max \left(\frac{P(v.)}{P_{max}} \right) < 1$	Maximum Motor Power Constraint Ensure motors power is within limits
$\max \left(\frac{A_m(v.)}{A_{m_{max}}} \right) < 1$	Maximum Motor Current Constraint Ensure motors current is within limits

The formulation translates to optimizing the maximum hover time (t_f) while simultaneously maximizing the minimum flight velocity across the different flight paths (v_l – flight speed for straight line, $v_{t_{300}}$ – flight speed for 300 m turn radius, $v_{t_{500}}$ – flight speed for 500 m turn radius), while ensuring that the flight path constraints are satisfied. The constraints are applied on the minimum distance traversed in flight path 1 ($d_s(v_l)$ – distance traversed at the lateral flight

speed), the minimum hover time in flight path 4 ($t_f(v_l < 3)$) – hover time for lateral speed less than 3 m/s), and the turning distance in flight path 3 ($d_s(v_{t_{300}})$) – distance traversed at chosen flight velocity for flight path 3). Furthermore, a set of vehicle’s electrical performance constraints are introduced to ensure that the electrical subsystem performance is capped by the maximum motor current, the maximum battery current and the maximum motor power drawn.

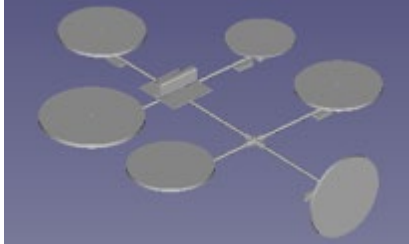
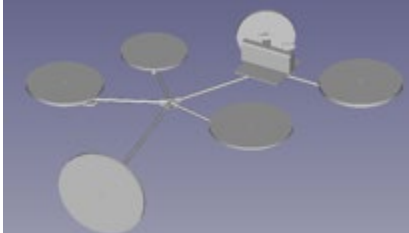
4.1.2 Results

As we execute the design process described above, we observe that the designs we obtain demonstrate significantly higher performances than the baseline seed designs provided. The seed designs provided by the design challenge organizers are presented in Table 10. As seen in the table, most of the seed designs are unable to successfully obtain any score in the test benches. In contrast, the designs that are generated using the Systemic Generative Engineering process demonstrate a considerable improvement in each of the paths and, also, the total score of the system. The top three designs are illustrated in Table 11. We observe that although no seed designs have multiple batteries in their design topology, the best performing design can generate designs that have multiple batteries and optimally distribute the electrical connections between the batteries and the propulsion systems. Further, as observed in the designs shown in Table 11, the design topologies of the generated designs are significantly different from those of the baseline. This illustrates that the algorithms can synthesize innovative designs, where innovation is defined as design topologies that are distinct from those observed.

Table 10: Seed designs provided by design challenge organizers for the UAV design challenge

Design	Name	Path 1	Path 3	Path 4	Path 5	Total
	Quadrotor	0	0	0	0	0
	HCopter	0	0	0	0	0
	QuadSpider	0	0	0	0	0
	HPlane	0	0	0	0	0
	HexRing	400	300	0	0	700

Table 11: Top-performing synthesized UAV designs using the Systemic Generative Engineering framework

Design	Path 1	Path 2	Path 3	Path 4	Total
	400	400	300	473	1573
	400	400	300	456	1556
	400	400	300	435	1535

4.2 Design of Arctic Exploration Unmanned Underwater Vehicle

4.2.1 Problem Summary

The Systems & Technology Research LLC (STR) TA3 team organized a second design challenge in which the AI co-designers were tasked with the conceptual design of a pair of UUV systems to support an environmental measurement challenge. The goal of the challenge was to identify UUV system designs that transit from a location near the edge of the northern ice sheet to a destination further north to deploy one or more sampling vehicles. The design target is to identify the system architecture and the components onboard for design of an expendable, underwater vehicle that can travel from the edge of the ice at 75N latitude to a point north of 85N latitude for a distance of 1000 km. This design requirement is illustrated in Figure 89. Once at the target destination, the transit vehicle is required to deploy one or more sampling vehicle whose role is to measure the conductivity, temperature, and depth (CTD) for depths between the surface and 3000 m for a period of 1 week while remaining within acoustic navigation and communication range of the transit vehicle. Once the sampling vehicle is deployed, the transit vehicle is required to breach the ice sheet and remain attached to the ice to transmit the measurements from the sampling vehicle to the home base.

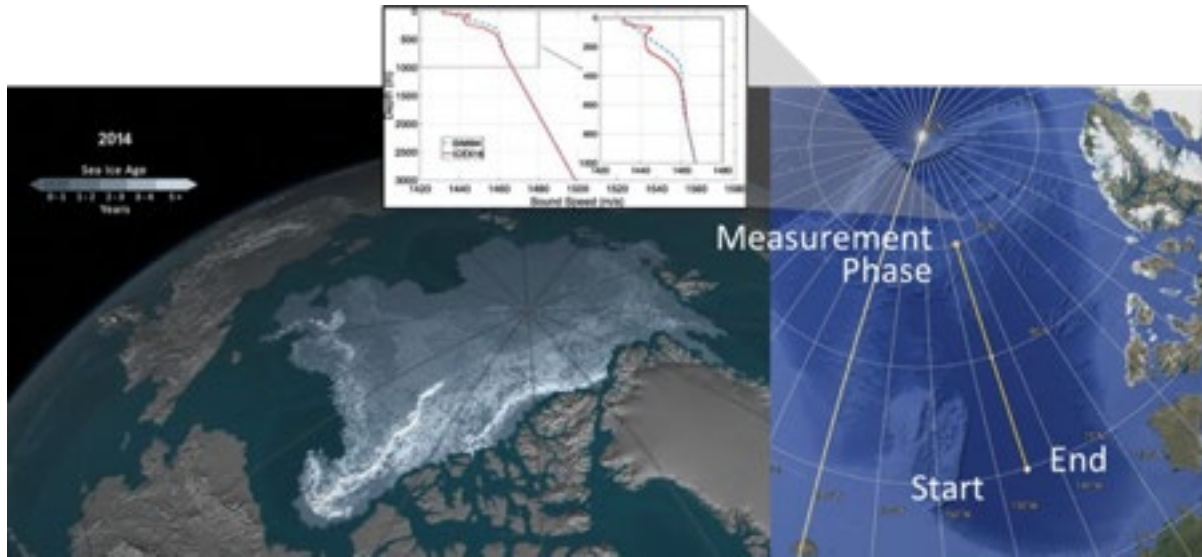


Figure 89: Overview of the mission requirement

The requirements for the transit vehicle are underspecified without specific requirements on the depth at which the vehicle must operate. On the other hand, the transit vehicle is required to carry an ice-penetration device. Furthermore, it is required to demonstrate a top-speed of 1.5 m/s even in adverse currents of up to 1 m/s. Finally, a constraint is enforced on the size of the transmit vehicle specifying the maximum value of the vehicle to be under 3.52 m³. The sampling vehicle is tasked with carrying a CTD sensor and attaching to the transit vehicle by some means.

During the execution of the design challenge, two variations were introduced to evaluate the ability of the AI co-designer to assess sensitivities and handle changes. The first variation required the transit vehicle to continue to the geographic north pole from its previous destination of 85 N latitude after the completion of the sampling mission. This variation permitted the vehicle to jettison any unnecessary payload and did not have to recover the sampling vehicle after the completion of the mission. The focus of the design challenge variation was to assess the effect of the additional transit requirement on the vehicle system characteristics (e.g., length, volume, and energy requirement). The variation relaxed the constraint on the volume of the vehicle, but instead introduced the volume constraint as a minimization objective for the design process. The second variation introduced a constraint on the transmit time in the second leg. In this variation, the vehicle had to reach the geographic north pole within 72 hours of the end of the sampling mission. The AI co-designers were permitted to redesign the vehicle such that the rated velocity of the transit vehicle could exceed the required 1.5 m/s velocity rating. The goal of this variation was to evaluate the impact of the additional time constraint on the time shape and size of the vehicle. A rough sketch of the base mission profile along with the variations is illustrated in Figure 90.

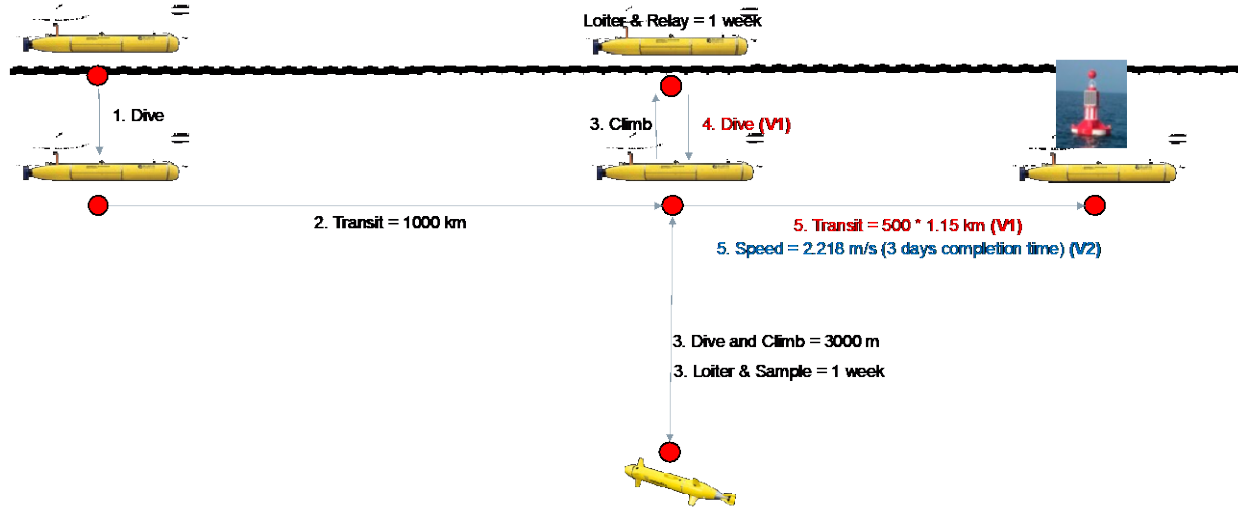


Figure 90: An illustration of the mission profile for which the UUV system has to be designed

4.2.2 Design Methodology

As the design challenge lacked a simulation framework to analyze the designed UUV with a custom system architecture for the specified mission, we developed an integrated mission-based simulation environment that permits the sizing and synthesis of both propeller driven and glider UUVs, while also permitting the use of a combination of both propulsion mechanisms. The design framework was based on first principles of physics and implemented a conceptual design sizing and synthesis framework that permitted the incorporation of mission phases and constraints to identify a feasible region of the design space that satisfied the requirements. The designs can then be optimized in terms of the high-level system parameters such as system weight, length, diameter, and glider wing areas. The governing equations are derived from the free-body diagram, illustrated in Figure 91. For steady state motion of the vehicle, i.e., no acceleration at any point in the mission, the vehicle must always follow the following steady-state equations of motion.

Along direction of motion:

$$\gamma L + F_B \cos(\theta_g - \alpha) = \beta W \cos(\theta_g - \alpha) + m * a_v \quad [64]$$

Perpendicular to direction of motion:

$$\beta W \sin(\theta_g - \alpha) + \delta T = D + F_B \sin(\theta_g - \alpha) + m * a_h \quad [65]$$

In Figure 91, θ_g represents the glide angle, θ_p represents the pitch angle at which the vehicle transits, and α represents the angle of attack for the propulsion system. In the equations presented above, γ represents the fraction of lift generated at any stage in the mission, β represents the instantaneous weight fraction of the system with respect to the original deployed weight, and δ represents the fraction of thrust contributing to the propulsion of the system.

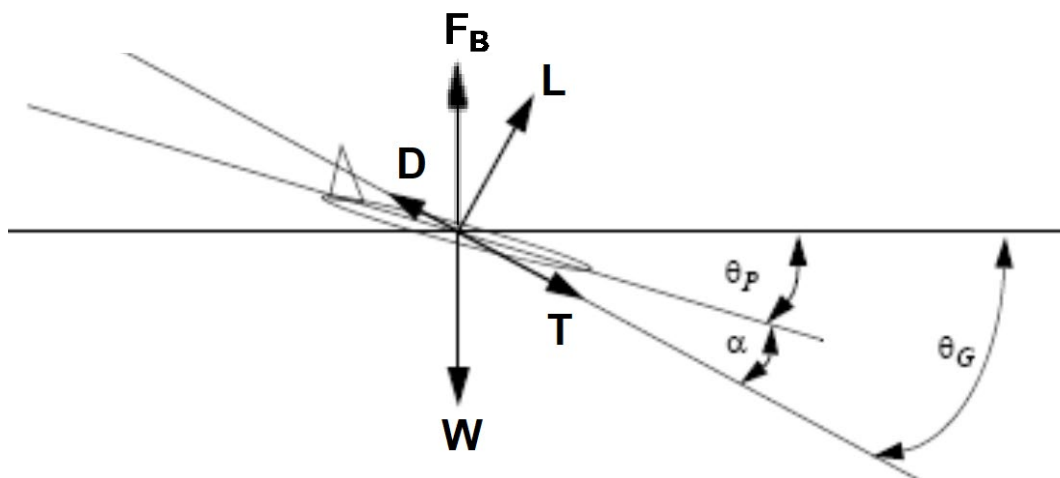


Figure 91: Free-body diagram guiding the governing equations of motion for the various phases of the mission.

These steady-state equations of motion are implemented for each phase of the mission, i.e., climb, transit, dive and loiter such that a constraint-based sizing of the vehicle can be performed. The sizing environment is developed in Python and is illustrated in Figure 92.

The process involves the use of the sizing environment to identify conceptual size of a known system architecture. The sizing environment ingests the vehicle architecture (i.e., fraction of propulsion to glider powered transit), the requirements from the definition of the mission (e.g., the various phases of the mission) and the performance requirements (e.g., the range constraint, volume constraint, etc.) to determine the size, weight and thrust (and thus, energy) requirement for the vehicle. Internally, the sizing process iteratively converges between a constraint analysis module, a mission analysis module, and a synthesis module. The constraint analysis module balances the power required for propulsion of the vehicle to meet the specified set of constraints and the power available from the thrusters and the energy sources. The mission analysis module balances the energy required to complete the mission and the energy available in the energy sources. Finally, the synthesis estimates the component weights based on an abstraction of the entire system. The process is iterative as an increase (or any change, in general) to the system weight will alter the power required to realize the same velocity constraints, which in turn will alter the energy consumption profile of the vehicle necessitating the need for large energy sources, which in turn will alter the weight of the energy sources. Thus, a fixed-point iteration algorithm is employed to converge to an optimal operating point for each design configuration. Finally, it must be noted that the sizing process is executed for each vehicle in the system, i.e., both the transit and the sampling vehicle.

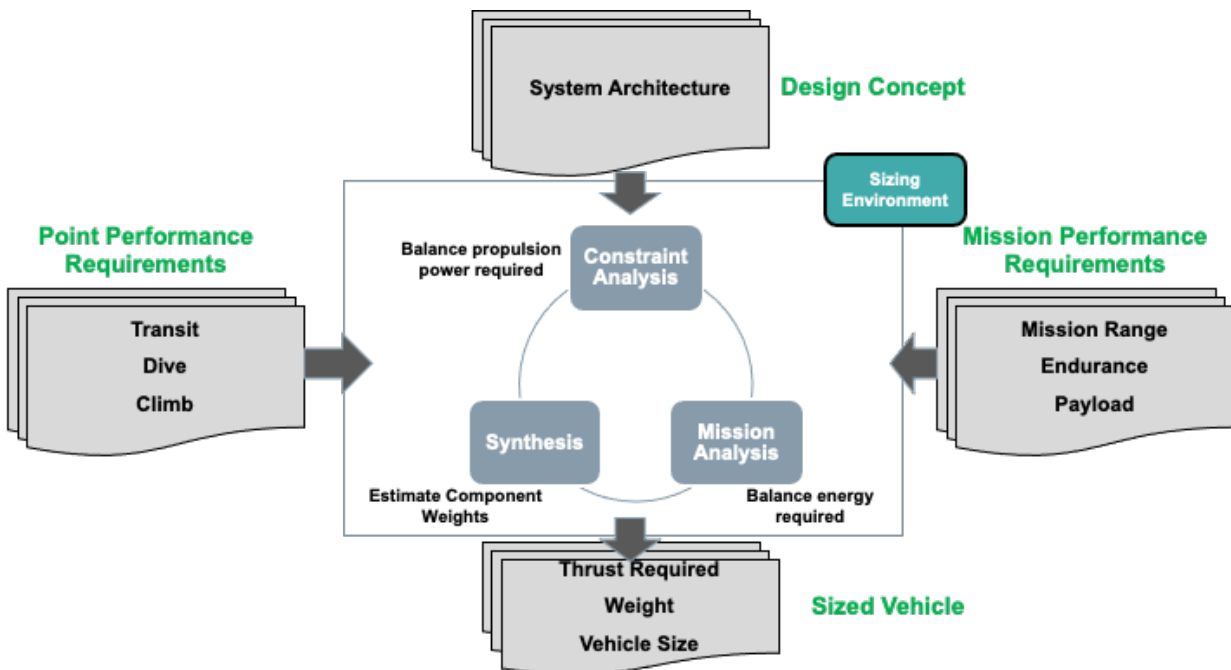
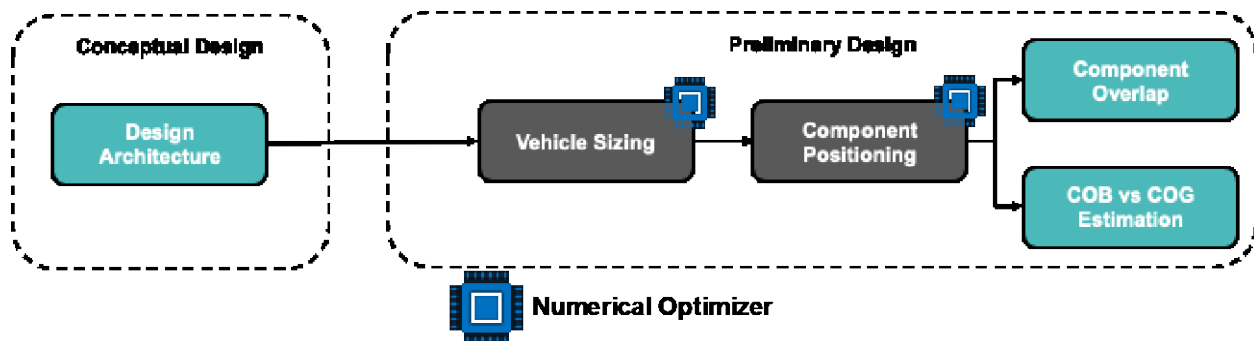
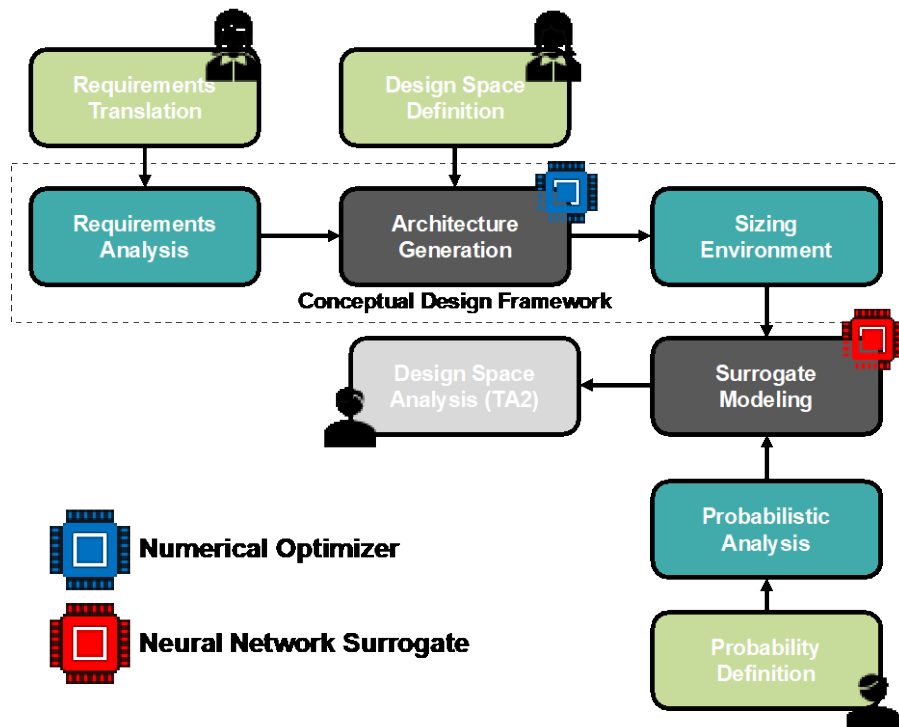


Figure 92: A conceptual illustration of the sizing environment for the UUV design

The sizing environment is next integrated with an optimization process which permits the exploration of the conceptual design space. The optimization process is divided into two steps. The first executes the conceptual sizing of the vehicle and the second optimizes the placement of the components inside the sized vehicle. The conceptual design process is illustrated in Figure 93.

The process begins with an analysis of the requirements to identify the mission constraints and the mission performance requirements. Once analyzed, the design space is defined in such a way that designs can be sampled of varying architectures. In this implementation, the impact of the sampled design architecture is such that the designs are translated to an effective single hull vehicle in terms of the area, length and diameter. The generated architectures are evaluated in the sizing environment where the vehicle characteristics such as the length, diameter and energy requirements are computed. The process is then integrated with a continuous optimizer (in this case SHERPA) that optimizes the design parameters to minimize the weight and the volume of the vehicle. The optimized vehicle's subsystem and component weight estimations are then used in the parametric optimization for the vehicle layout. This process is illustrated in Figure 94.

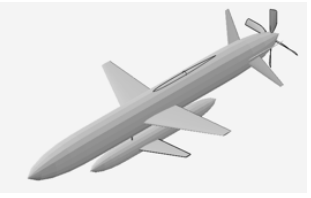
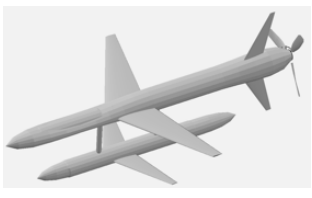
The process involves the use of numerical optimization to optimize the component placement such that the center of gravity and center of buoyancy are overlapping and there are no overlaps in any component placements. The component overlaps are estimated using a rectangular volumetric overlap analysis.



4.2.3 Results

The design process described is executed to estimate the design characteristics for the base case and the design variants. We design two vehicles for each design challenge variant. We introduce a new technology that enables energy regeneration using a hydro-generator for the second design in which the component characteristics are estimated using a surrogate model developed on a set of datapoints collected from the public domain. The outcomes of the design processes are documented in Table 12 where we show the system characteristics of the generated designs.

Table 12: Summary of the designs generated during the UUV design challenge

Design	Baseline	Design 1 (V1)	Design 2 (V1)	Design 1 (V2)	Design 2 (V2)
Model					
Range (km)	1000	1500	1500	1575	1575
Weight (kg)	126.841	156.314	171.1802	310.7701	303.0785
Energy (J)	6.63M	9.39M	12.17M	37.89M	33.93M
Battery Wgt. (kg)	35.595	50.40	35.52	93.46	88.79
Buoyancy Bladder Wgt. (kg)	18.648	11.314	80.06	-	-
Hull Length (m)	2.488	3.18	3.18	3.51	3.51
Hull Diame- ter (m)	0.32	0.48	0.48	0.538	0.538
Wingspan (m)	2.02	3.11	2.31	-	-
Velocity (m/s)	1.5	1.5	1.5	2.218	2.218
Regenerative	No	No	Yes	No	Yes
Depth Rating (m)	2027	1747	1384	300	300

4.3 Design of Urban Air Mobility Vehicle

4.3.1 Problem Summary

The Urban Air Mobility (UAM) design challenge organized by SwRI builds on the UAV design problem. In this setting, the problem definition was altered such that the design vehicle is of a large scale and is required to carry two (2) passengers. To accomplish this, a new set of design corpus was released while retaining the same design simulation interface. The design challenge employed the same four (4) paths that the vehicle must traverse, with a modification to the scoring on Path 1. The goal for Path 1 was to have the vehicle fly in a straight line for as long as possible. This implied that there was need to design a vehicle that minimized the energy consumption such that a given battery could last the longest. This motivated the use of vehicles that relied on wings and lift to propel the vehicle. Given that there was no theoretical maximum on the distance that could be traversed by the vehicle, Path 1 did not have a theoretical maximum on the score that can be achieved. But, due to an implementation issue where the simulation was terminated after 1000 seconds, and because the simulation could only support times until 50 m/s, there was an artificial

limit on the maximum score that could be achieved on Path 1. The scoring for Path 1 was updated to be:

$$score_1 = \begin{cases} 0 & \text{if } d_s(v_*) < 2000 \\ \frac{d_s(v_*)}{10} & \text{if } 2000 < d_s(v_*) < 4000 \\ 400 + \frac{d_s(v_*) - 4000}{100} & \text{otherwise} \end{cases} \quad [66]$$

Thus, given the 50 m/s limit on the velocity and the 1000 second limit on the simulation time, the maximum distance that could be traversed was 50,000 m. This implies that the maximum score that was achievable was 860. The UAM design challenge also introduced new design tools to be used by the AI co-designers in creating the designs. These included a Direct2CAD pipeline that generated a CAD model of the vehicle from a given design topology input in GraphML format and a battery checker python executable that estimated the validity of the selected battery for a given design architecture.

4.3.2 Design Methodology

To aid the design space exploration and to retain consistent results with the validation pipeline, the previously developed FreeCAD and Python interfaces were scrapped in favor of the Direct2CAD interface. A pipeline, in Python, automating the flow of information between the battery checker, Direct2CAD and the Fortran flight dynamics simulation module was developed. The pipeline directly operates with JavaScript Object Notation (JSON) representations of *networkx* graphs, which is the chosen representation for the designs. The updated design pipeline is illustrated in Figure 95.

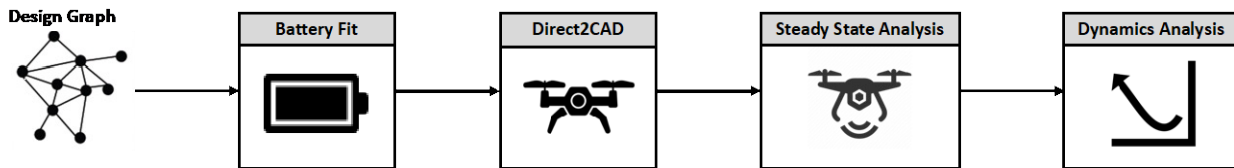


Figure 95: UAM analysis simulation pipeline developed

As in the case of the UAV, the Systemic Generative Engineering framework is tasked with designing both the vehicle architecture, the configuration, and the parameter values of the continuous variables. The design space details are provided below:

- **Architecture Design:** The architecture design of the UAM vehicle follows that of the UAV. Here, we consider the design topology as the architecture of the vehicle. This includes structural, propulsion, aerodynamic and electrical components with mechanical and electrical connections between the components. While there is no theoretical limit to the number of components that can be incorporated into the design, the simulation is unable to analyze designs that have more than 50 control inputs. The control inputs include actuation surfaces and propulsion systems.

- **Preliminary Design:** As with the UAV challenge, we determine the positioning of the various components and their types in preliminary design. The off-the-shelf components available for the optimizer to select from are 27 batteries, 102 motors, 940 propellers and 68 wings.
- **Detailed Design:** As part of the detailed design, we optimize the scalable and continuous parameters of the design such as the wing chord and span, the connector sizes, and the fuselage parameters. Additionally, we optimize the control parameters for the different segments of the racetrack (5 control parameter segment).

Table 13 summarizes the design space in which the exploration is executed.

Table 13: Design space for the UAM design challenge

Subsystem	Component	Type	Size
Propulsion	Battery	Discrete	27
	Battery Voltage, Percentage	Continuous	2x # Battery
	Motor	Discrete	83
	Propeller	Discrete	417
Aerodynamics	Wing	Discrete	68
	Wing Characteristics	Continuous	5x # Wings
Structures	Connectors	Discrete	1
	Connectors Radius, Length, Orientation	Continuous	3x # Connectors
	Fuselage	Discrete	1
	Fuselage Diameter, Length, Thickness	Continuous	3x # Fuselages
Controls	Autopilot (LQR) Parameters	Continuous	20

Our goal in this design challenge was to demonstrate the value of symbiosis between the AI co-designer and human designers. Thus, we develop a workflow that leverages human knowledge in exploring the design space. The workflow is illustrated in Figure 96. The design process begins with a set of designers providing a partially complete design sketch. These sketches are used in an auto-completion engine, i.e., design composition engine, to generate a full design. The design completion synthesizes several potential designs from the partial sketches provided and leverages a surrogate model along with a set of design constraints and a synthesized design space to identify the potentially top-ranking designs. These top-ranking designs are evaluated for trim performance with a goal of identifying designs that obtain the best trim characteristics. Such designs are down selected for detailed optimization. The optimization is carried out in three steps:

- We first execute a parametric optimization altering the continuous parameters for each design topology. Here the goal of the optimization is to improve the identified trim characteristics of the design, for e.g., increasing the maximum velocity of flight or increasing the maximum hover time. The optimization formulation for the trim is similar to the trim formulation used for the UAV simulation. It is given by:

$$\text{Objectives: } \max\left(\frac{d_s(v_l)}{4000}\right), \max\left(\frac{t_f}{400}\right), \max\left(\frac{\min(v_l, v_{t_{300}}, v_{t_{500}})}{50}\right) \quad [67]$$

Subject to constraints,

$\max\left(\frac{d_s(v_l)}{2000}\right) > 1$	Maximum Flight Distance Constraint Ensures Path 1 produces a non-zero score
$\max\left(\frac{t_f(v_l < 3)}{200}\right) > 1$	Maximum Flight Time Constraint Ensures Path 4 produces a non-zero score
$\max\left(\frac{d_s(v_{t_{300}})}{3142}\right) > 1$	Maximum Flight Distance Constraint Ensures Path 3 produces the maximum score
$\max\left(\frac{A(v)}{A_{max}}\right) < 1$	Maximum Battery Current Constraint Ensure battery current is within acceptable limits
$\max\left(\frac{P(v)}{P_{max}}\right) < 1$	Maximum Motor Power Constraint Ensure motors power is within limits
$\max\left(\frac{A_m(v)}{A_{m_{max}}}\right) < 1$	Maximum Motor Current Constraint Ensure motors current is within limits

- Next, we execute an electrical layout optimization in which the connections between the batteries and the propellers are optimized to yield the maximum flight time and maximum hover time. This executes a combinatorial optimization in which the electrical layout in the system is altered. The optimization formulation for the electrical layout optimization is given as,

$$\text{Objectives: } \max_{e_{ij}}\left(d_s(v_l)\right), \max_{e_i}(t_f) \quad [68]$$

Where $e_{ij} \in \{0, 1\} \forall i \in \{0, k\}, j \in \{0, n\}$, $k = \# \text{ batteries}, n = \# \text{ propellers}$

- Finally, we execute a control optimization in which we optimize the flight conditions to maximize the scores in the evaluation paths. The optimization formulation for the control optimization is given as,

$$\text{Objectives: } \max_{c_{ij}}\left(\sum_j s_j(c_{ij})\right) \quad [69]$$

Where $c_{ij} \in [0, 50] \forall i \in \{0, 5\}$ and $j \in \{1, 3, 4, 5\}$ and s_j is the score in Path j
The designs that report the best scores through this optimization process are supplied back to the designers so that they can make further tweaks to the design to improve the performance.

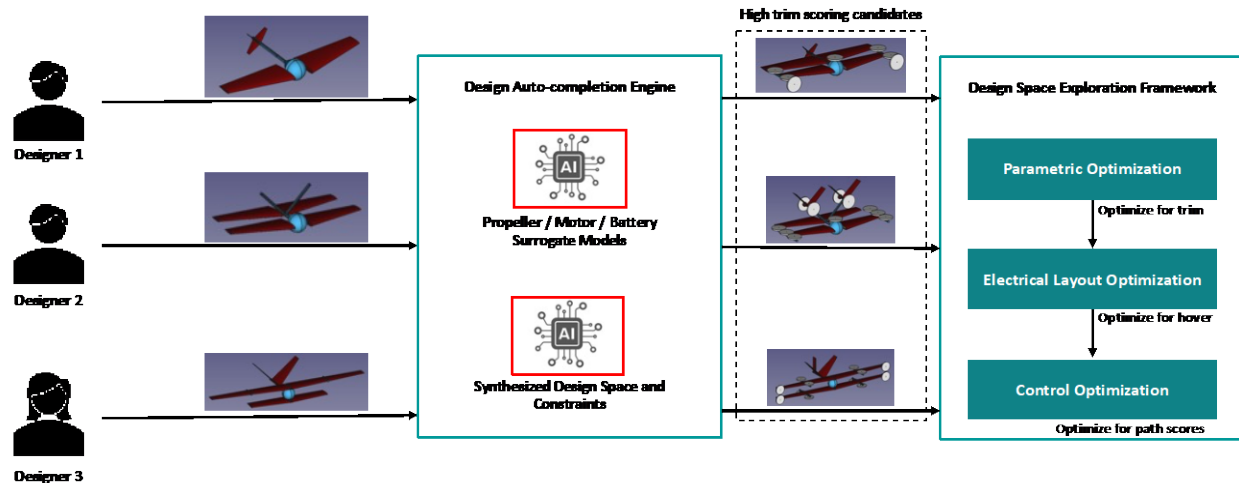


Figure 96: Design process workflow employed to explore the design space for the UAM design challenge.

4.3.3 Results

The design process when executed can generate designs that respect the structure provided by the user while significantly improving the performance of the resultant designs. We see that the completion algorithm can generate designs from sketches that have a high degree of incompleteness, where the samples that were generated had 25% – 90% of the design components and all the relevant connections between the components specified in the sketch provided. Furthermore, we demonstrate that both our design space synthesis and the constraint synthesis algorithms can learn with very few seed designs – in this case, just two (2) seed designs are provided by the design challenge organizers. In the case of the constraint synthesis algorithm, we also demonstrate the ability to adaptively learn the constraints during the design space exploration process. The seed designs and their respective performances are presented in Table 14. These are contrasted with the top 3 designs that are generated by the Systemic Generative Engineering framework, shown in Table 15.

Table 14: Seed UAM designs summary

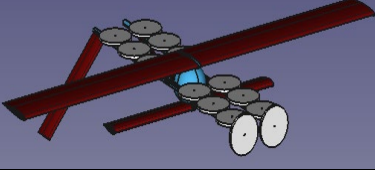
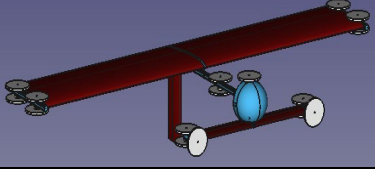
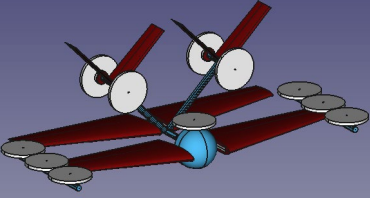
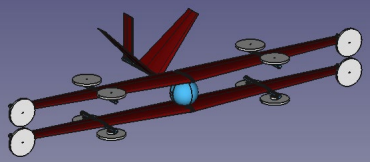
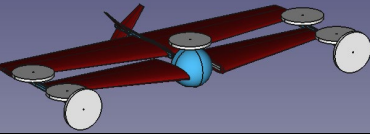
Design	Name	Path 1	Path 3	Path 4	Path 5	Total
	Trowel	570	300	236	482	1588
	Rake	415	300	0	396	0

Table 15: Top generated designs using the Systemic Generative Engineering framework

Design	Path 1	Path 2	Path 3	Path 4	Total
	400	850	300	482	2032
	400	770	300	482	1952
	400	760	300	482	1942

A further point of note on the designs generated by the Systemic Generative Engineering framework is that by optimizing for the trim performance, we find designs that are extremely robust. In fact, the three designs presented can achieve trim at all 150 design conditions (flight velocities of 1 – 50 m/s in the three flight paths – lateral flight, turning flight of 300 m, and turning flight of 500 m). This implies that the design of controllers for such systems is a very easy task as finding a stable operating point for the vehicle is simplified due to the availability of several potential operating points.

4.4 Phase-end Demo 2 – Design of Arctic Exploration Unmanned Underwater Vehicle

4.4.1 Problem Summary

As part of the Phase end demo, the STR team defined a design challenge focusing on the design of a UUV for deep arctic ocean exploration. As part of this design challenge, the AI co-designers are required to complete the conceptual design of a long distance, under-ice transit vehicle that roughly covers a 2000 km distance from a location off the coast of Alaska to 350 km away from the north pole. The vehicle will carry an Ocean Bottom Seismometer (OBS) payload that is required to be deployed at the target location within some tolerance. The vehicle is then required to return to the start location, and complete the entire trip, to and back, within 60 days, with each phase of the mission being capped at 30 days. Further, the vehicle is required to carry onboard an acoustic modem for communication and a navigation system to assist with the geolocation of the vehicle. The mission goals are illustrated in Figure 97.



Figure 97: An illustration of the mission that the UUV must complete.

Additional mission requirements dictating the depth and location at which the payload can be deployed is specified. The payload must be deployed when the vehicle is between 50 – 200 m from the ocean bottom, and the payload must be deployed within 50 km of the nominal drop point.

In addition to the high-level mission requirements, a set of environmental constraints are imposed on the mission. Mainly, an environmental model of the ocean density is utilized to model the effect of changing water density with the depth of the ocean. Secondly, a stochastic model of the ocean currents is used to simulate the effect of vehicle drift as a result of the ocean current. The ocean current model is also a function of the depth of the water.

The vehicle is required to additionally meet a set of design requirements dictating the certain functional requirements on the subsystems of the vehicle. In particular, the vehicle's pressure vessels are required to be rated for 4000 m depth. The vehicle must be capable of climb and dive angles of 30 degrees and the vehicle must demonstrate control authority to support a 6 deg/s heading rate at its maximum speed.

4.4.2 Design Methodology

Based on discussions with the subject matter experts at The University of Washington Applied Physics Laboratory (APL-UW) and STR, it was identified that propeller driven UUVs are more suited for the mission due to the restrictions on the communication and sensing devices. Thus, the Systemic Generative Engineering AI co-designer was restricted to explore only propeller driven UUVs of a fixed design architecture. The exploration permitted variations to the number of pressure vessels, the number of batteries and the number of floats that can be used within the propeller driven UUV.

As the only simulation provided to the AI co-designer was the mission simulator, we developed a design pipeline based on open-source tools that integrates multiple disciplines. The multi-disciplinary design pipeline is illustrated in Figure 98.

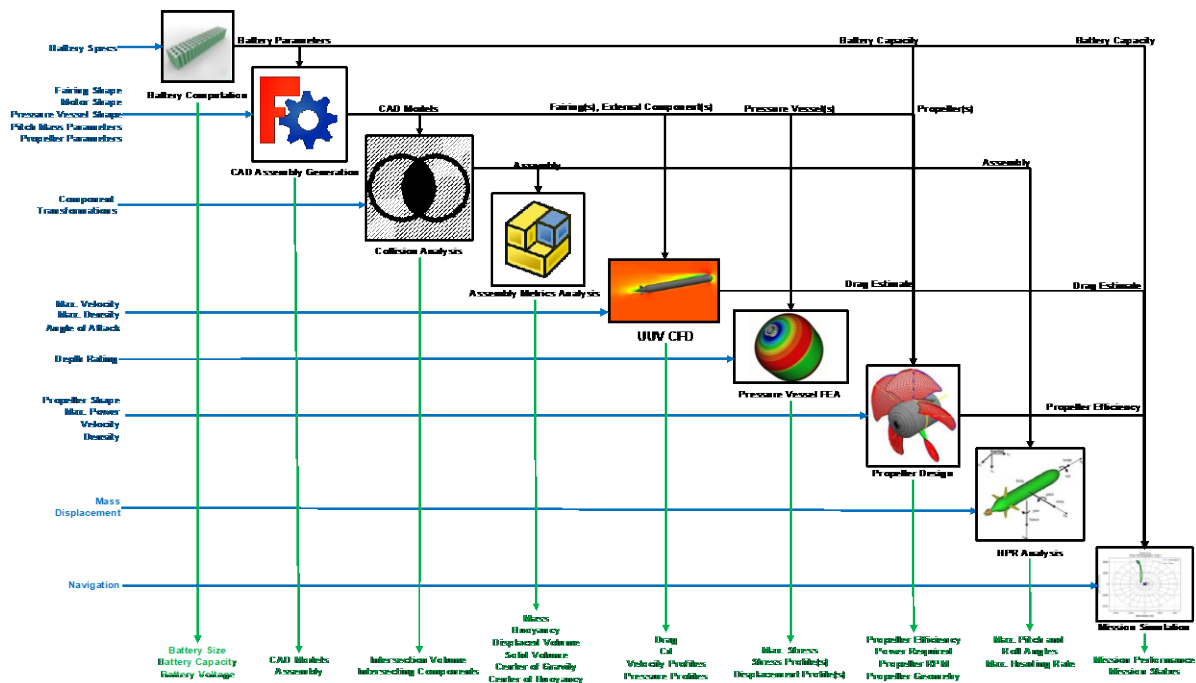


Figure 98: An integrated design pipeline for propeller based UUV design

Note that the design pipeline combines both design generation and simulations to estimate the overall design performance. Figure 98 represents one version of the Design Structure Matrix (DSM) that can be used to represent the design process of the UUV. The DSM illustrates the various simulation blocks in the diagonal of the matrix. The design inputs to each simulation are shown to the left of the matrix (in blue) and the outputs that are generated from the simulation are shown at the bottom of the matrix (in green). In the off-diagonal positions, the data communicated between the different simulations is illustrated (in black). The following provides a brief description of the design workflow:

- The design process takes as input a “design graph”. The design graph represents the system architecture in a format that is compatible with OpenMETA. The design graph is a JSON representation of a *networkx* graph. The graph contains information about the mission parameters (e.g., maximum velocity, operational density), system parameters (e.g., the total battery capacity), the components (e.g., fairings, pressure vessels, etc.) and their parameters (e.g., shape parameters of the fairing and pressure vessels), their locations (e.g., assembly locations for the pressure vessels or batteries), and the assembly connections (e.g., the mechanical and electrical connections that need to be realized in the assembly of the design).
- Based on this input graph, we generate the entire design using the workflow shown in Figure 98 or its variants. The workflow starts by estimating the size of the batteries based on a specification of the total battery capacity required for the mission. The battery capacity is an optimization parameter that we attempt to minimize during the optimization.
- The battery parameters, i.e., battery pack length and radius for each battery, along with the component specifications provided in the design graph, are fed to a CAD model generator. The CAD model generator is automated in FreeCAD. In addition to individual components,

the FreeCAD implementation generates the UUV assembly. The UUV assembly is generated based on coordinate axes defined in the CAD models and the coordinate axes mates that are defined as the edges of the design graph.

- Having generated the assembly, the workflow executes a “collision analysis” to compute the intersection between the different components in the assembly. This process involves the conversion of each component in the assembly into a STL and the subsequent estimation of overlaps in the STL meshes. The step provides an estimate of the overlapping components and the corresponding overlap volumes.
- With the generated geometry, we also execute the “assembly metrics” analysis. This analysis computes the overall system metrics such as mass, buoyancy, moments of inertia, solid and displaced volumes, the center of gravity, center of buoyancy and their separation. The metrics are used to estimate the overlap between the center of buoyancy and the center of gravity.
- Having estimated the metrics related to the geometry and the assembly, the process executes the computational fluid dynamics simulation to estimate the drag of the vehicle. The fairing used in the assembly is supplied to an automated pipeline built on OpenFOAM by Stevens University (i.e., DexOF tool). The CFD metric of relevance such as the fairing drag, the coefficient of drag, coefficient of lift and the side force coefficients are computed at the design target depth (i.e., density), velocity and angle of attack (both specified in the design graph) and the standard ocean conditions of 1027 kg/m³ density, 1 m/s velocity and 0° angle of attack.
- In parallel, the design process executes a finite element analysis on each of the pressure vessels to estimate the maximum von Mises stress that is encountered at the rated depth of 4000 m. The FEA automatically generates a mesh of each of the pressure vessels using Gmsh meshing tools and computes the stresses in the operation using the Calculix solver. The design parameters of interest in the FEA are the material of the pressure vessel and its shape parameters. The computed stresses are reported for each pressure vessel and passed onto the optimizer.
- The computed drag from CFD is passed onto a propeller sizing tool implemented in OpenProp. In addition, the desired operational density (i.e., function of depth), the target transit velocity and the propeller geometry parameters are used to estimate the optimal revolutions per minute (RPM) at which the propeller must operate to produce the required thrust at the rated velocity. OpenProp also estimates the power consumed by the propeller and its operational efficiency during this process. The propeller geometry is a design parameter that is modified during the optimization process.
- Next, the process executes the heading, pitch, and roll (HPR) analysis to estimate the maximum pitch and roll angles that can be achieved by moving pitch masses. The analysis algorithm is based on estimating the change to the center of gravity relative to the center of buoyancy due to the change in the pitch mass position. The angle arising from this angle yields the maximum pitch and roll angles that can be achieved.

- Finally, the mission simulator is executed to estimate the energy consumption, the mission completion and the success status based on the vehicle drag estimated from CFD and propeller efficiency designed. The optimizer alters the navigational controls, i.e., the segment-wise velocity and heading rates to ensure the completion of the mission.

While the developed design workflow automates the design process, there is still a limitation to the process. Several steps in the process are either manual or take a very long time to execute. This significantly hampers the exploration that can be performed in the design generation. These are primarily,

- **Geometry and assembly generation:** The process of geometry and assembly generation is traditionally extremely manual or requires very complex automation.
- **CFD analysis:** The results of the CFD analysis of the fairing are very sensitive to the mesh used in the analysis process. Typically, a finer mesh implies more accurate results. But on the other hand, the simulation times are directly proportional to the size of the mesh. In our experiments, the CFD analysis can take between 5 minutes to 6 hours on a 24-node compute cluster depending on the mesh sizes used.
- **Pressure Vessel FEM analysis:** As with the CFD analysis, the results and the simulation time of the FEM analysis is also dependent on the quality and size of the mesh generated, with the time and accuracy being traded off against each other. In our analysis, the FEM analysis took between 2 minutes and 2 hours on a 12-node compute cluster.
- **Propeller Design:** The OpenProp analysis, although not very time consuming, takes about 1 minute per operational optimization. The optimization relies on an enumeration of all possible RPMs to identify the one that utilizes the least power and yields the maximum efficiency.
- **Mission Simulation:** Owing to the stochasticity in the ocean current models, a single design which is optimal and successfully completes the mission in one simulation, may fail to complete the mission in another. Thus, there is a need to consider a mean performance of the vehicle to successfully complete.

Compiling the measures from the issues listed above, we notice that there is an issue pertaining to the simulation time, the automation of CAD generation and the stochasticity in the mission simulation framework. To alleviate these issues, we develop an AI-enhanced design framework that incorporates AI models within the design workflow presented in Figure 98. The updated workflow is presented in Figure 99.

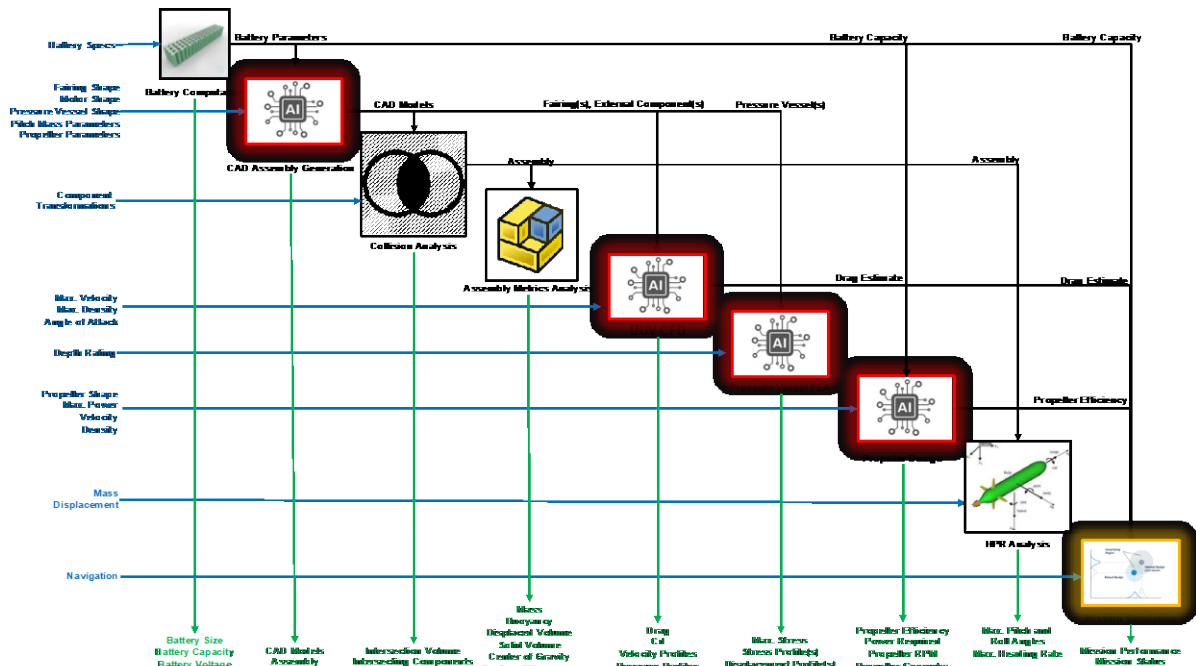


Figure 99: AI-enhanced design workflow for the design of a propeller-driven UUV

The AI-enhanced workflow has the following modifications:

- Automation of geometry and assembly generation:** A key development that permits the automation of the entire pipeline is the parametrization of the vehicle shape. The goal of the parameterization exercise was to identify a representation that is comprehensive but also simple. We seek a representation that encodes a variety of shapes within the fewest parameters. Furthermore, we seek to learn a representation with only a handful of seed designs that is invertible and explainable. We also recognize that the traditional shapes of UUVs are often very hydrodynamic in nature and that the performance of the vehicle is driven by this consideration. Thus, given these restrictions, we employ a class-shape transformation parametrization of the vehicle fairing. The CST parametrization of the fairing permits us to encode the entire shape in just a handful of parameters (in this case, 16) that allows us to explore a variety of fairing classes (e.g., thick nosed vehicles vs thin nosed vehicles) and shapes within a selected class (e.g., variations to curvatures of the fairing). Further, as both the class parameters and the shape parameters are continuous in nature, we can employ gradient-based search methods during the exploration of the design space to find the optimal fairing design yielding the least drag. This modification is represented as the first AI block (with a red highlight) in Figure 99.
- Acceleration of expensive simulations:** To accelerate the CFD, FEM and OpenProp analyses, we employ a set of surrogate models and reduced order models. The CFD simulation is replaced with a multi-scale model that not only predicts the flow field returning all the characteristics of interest, but also predicts the system level metrics such as the drag and normalized coefficients. The FEM analysis is replaced with a surrogate model that predicts the maximum stress that is expected to be observed in the pressure vessel for a combination of the pressure vessel's material and its shape parameters. Finally, the OpenProp analysis

is replaced with a simulation that predicts the optimal operating characteristics of the propeller (RPM, power, and efficiency) as a function of the geometric parameters and operational (thrust, velocity, and density) requirements of the propeller. These modifications are represented as the second, third and final AI blocks (with a red highlight) in Figure 99.

- **Robust design analysis of the UUV's mission performance:** To account for the effect of stochasticity in the ocean current model, we leverage a robust analysis process in which 5,000 simulations of the mission are executed in parallel. The probability of success measured as a ratio of number of successful missions to the total number simulated is maximized by altering the navigational parameters. This is represented by the final robust design block (with a golden highlight) in Figure 99.

With these modifications, we demonstrate that we can significantly accelerate the simulation time. The details of the parametrization and the surrogate / reduced order models are provided in section 3. Concretely, the new AI-enhanced algorithm takes under 30 seconds per design evaluation while the original workflow takes a minimum of 2 hours. This workflow is embedded in an optimization loop, as illustrated in Figure 100.

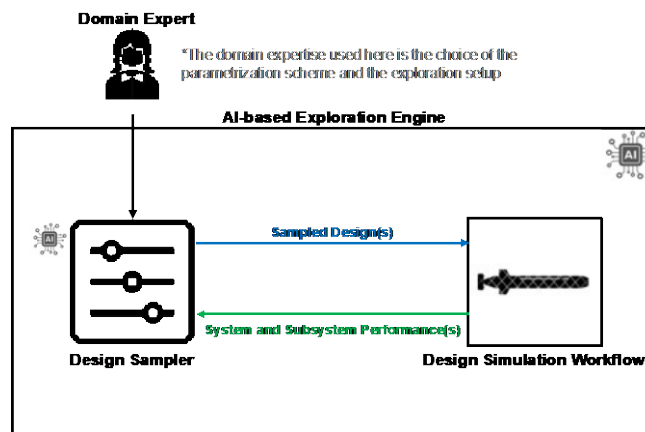


Figure 100: Integration of AI-based exploration with the UUV design analysis workflow

As part of the AI-based design process, we explored:

- **Invertible neural networks** are used to size the pressure vessel.
- **Surrogate-based hybrid adaptive optimization** algorithms are used to optimize the fairing and solve the UUV packing problem.
- **Probabilistic graph synthesis** generates the design architecture (i.e., number of pressure vessels, number of batteries, etc.)

The optimization formulation driving the design space exploration is given as:

$$\begin{aligned}
 &\text{Objectives:} \\
 &\min(\text{mass}), \min(\text{energy}), \min(\text{drag}), \min(d(\text{cob}) - d(\text{cog})), \\
 &\quad \max(p_s(\text{mission}))
 \end{aligned}
 \tag{70}$$

Subject to constraints:

$\max(stress_i) < stress_y$ $\forall i \in \{1, n\} \text{ where } n = \# PVs$	Pressure Vessel Constraint Maximum von Mises stress must be less than the yield stress of the material
$\sum c_i \cap c_j = 0$ $i = \{1, n\}, j = \{1, n\}, i \neq j$ where, c_k is component with index k and $n = \# components$	Collision Constraint Enforces that there should be no collision between any two components.
$\max(\theta_p) > 25 \text{ and } \min(\theta_p) < -25$ where, θ_p is the pitch angle	Pitching Constraint Enforces that the maximum and minimum pitching constraints are satisfied.
$mass - buoyancy < 5 N$	Stability Constraint Enforces that the difference between the system mass and buoyancy is within an acceptable margin.

The formulation translates as a multi-objective problem in which the task of the optimizer is to minimize the mass of the UUV, minimize the energy consumed by the UUV, the drag of the UUV fairing, the separation between the center of buoyancy and center of gravity and maximize the probability of success of the mission, subject to the pressure vessel stress, component collision / overlap, stability (i.e., buoyancy) and the pitching angle constraints. The optimization problem is solved using the multi-objective SHERPA algorithm in HEEDS which relies on the invertible neural networks to synthesize the pressure vessels to the constraints specified.

There is another assumption made in the construction of the design workflow presented in Figure 98. The design workflow assumes that there is no feedback between the different simulations. This assumption is incorrect. For example, in reality, there should be feedback between the propeller design and the battery analysis modules. Based on the energy consumed by the propeller, the batteries on the vehicle would have to be resized to minimize the system mass. Similarly, the design of the propeller and the pitch masses are also required to feedback to the geometry and assembly generation module. Finally, there is feedback between the mission simulation that estimates the total energy consumed by the vehicle in its operation and the battery sizing modules. To capture all this feedback in the design process, we implement a multi-disciplinary analysis and optimization inspired workflow, implementing two separate iterator algorithms. We first implement the multi-disciplinary feasible (MDF) design architecture, illustrated in Figure 101. The feedback loops are illustrated in red. The feedback loops necessitate a need for iterative convergence between the various disciplines that significantly slows down the optimization process. On the other hand, as all the designs are multi-disciplinary feasible, the designs themselves are all feasible and meet all the constraints (e.g., no collisions, sufficient battery capacity, pressure vessel stresses, etc.).

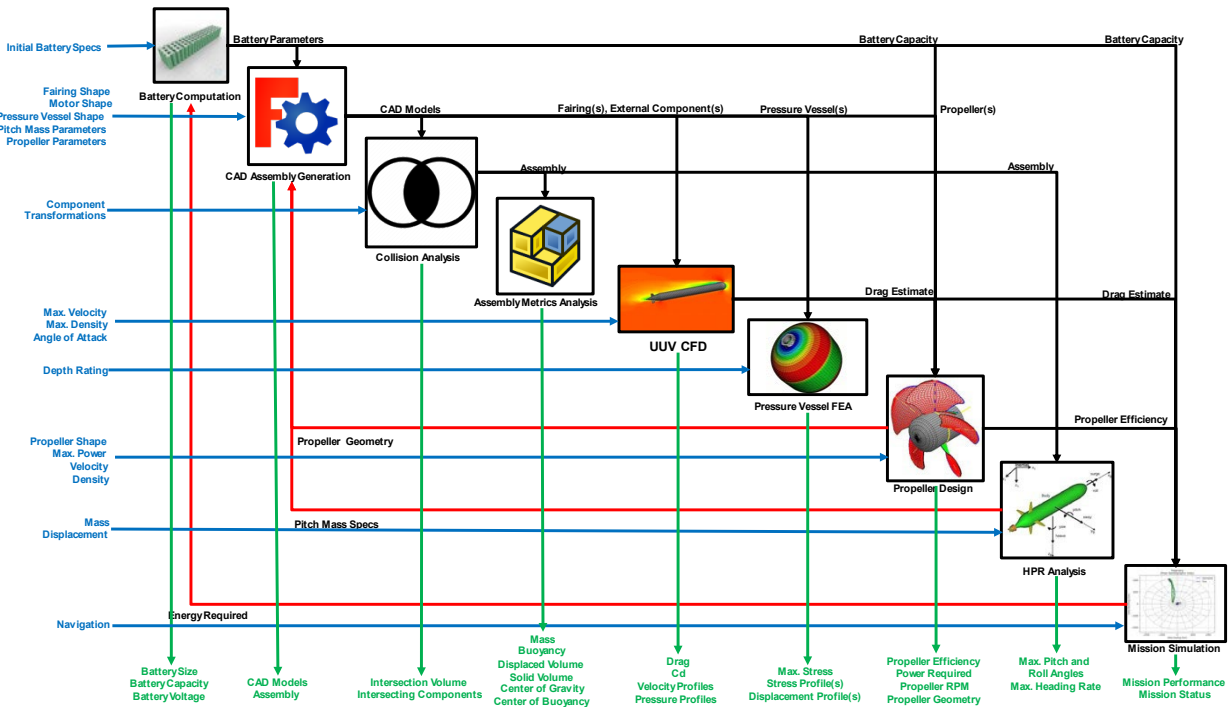


Figure 101: Multi-disciplinary feasible design architecture for the multi-disciplinary design of the UUV

To overcome this issue, we implement a second MDAO-based design architecture, the individual disciplinary feasible (IDF), illustrated in Figure 102. In contrast to the MDF implementation, the IDF uses a global controller that ensures that parameters of the various disciplines of interest are in sync in a global setting. This implies that each discipline has the freedom to explore its own design space but its interaction with the other disciplines is managed by the global controller. This permits significantly faster evaluation of the designs when compared against the MDF implementation, while having greater number of consistent designs.

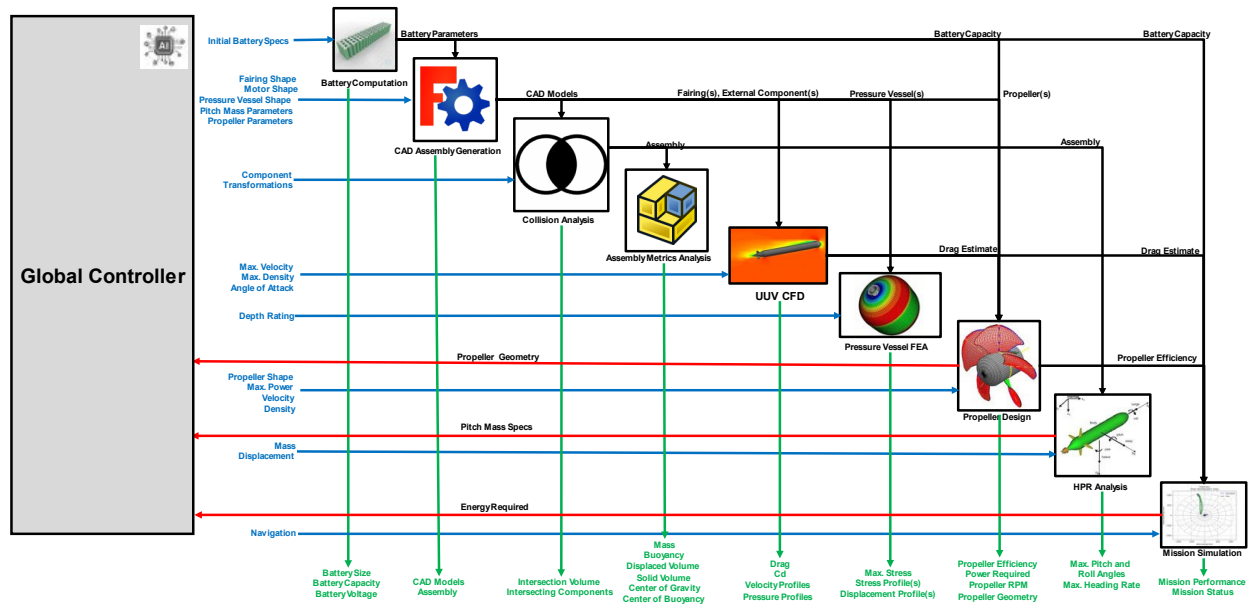
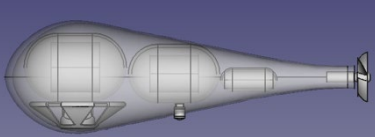
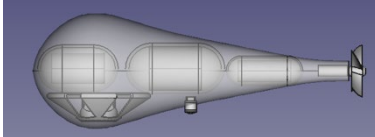
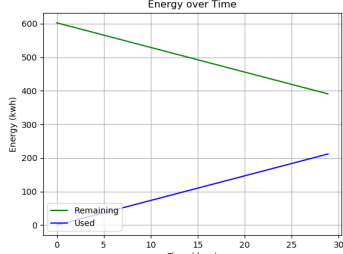
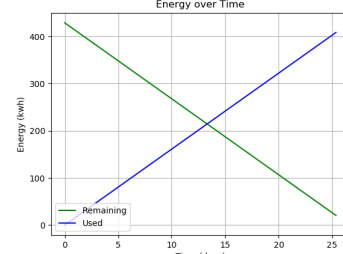


Figure 102: Individual Disciplinary Feasible MDAO architecture implementation of the UUV design analysis workflow

4.4.3 Results

Although the MDF implementation provided several feasible designs, in our experiments, the IDF implementation was the one that found the best performing design. The best design is characterized by the optimization formulation provided above. The results of an optimization process where the optimizer did not account for the energy consumption and another in which energy consumption of the vehicle was minimized are presented in Table 16.

Table 16: A summary of the optimal designs discovered by the Systemic Generative Engineering framework

Attribute	Optimized without energy consumption consideration	Optimized with energy consumption consideration
Length (m)	3.6375	3.03635
Max. Diameter (m)	1.17385	1.1504
Min/Max Displaced Volume (m ³)	2.5507	1.58356 / 1.617358
Min/max Vehicle Mass (kg)	2677.44	1602.0381 / 1693.3381
CoB-CoG offset (m)	X: 0.09, Y: 0.0, Z: 0.014	X: -0.0004, Y: 0.0, Z: 0.006
Hotel Power (W)	1.3	15.0
Drag Force at standard conditions (N)	36.7903	43.3689
Max. Available Thrust (N)	123.471	243.938
Thruster Efficiency (%)	0.831	0.836
Max. Vehicle Speed (m/s)	2.5	2.0
Number of Battery Cells	101177	72111
Volume of Battery Pack (m ³)	0.182	0.1299
Mass of Battery Pack (kg)	1418.328	1009.568
Total Energy Capacity (kWh)	602	429.06
Design Visualization		
Mission Energy Consumption Profile		

The key observation in the designs that are generated is that in comparison with the baseline tadpole-based design which is 6 m long and weighs over 3000 kg, both optimal designs are significantly smaller and lighter. Furthermore, the energy consumption optimized design optimally uses its energy such that minimal reserves are retained to account for the uncertainty in the mission. Although the vehicle geometry visually looks similar, there is a significant difference to the vehicle shapes both in the total size of the vehicle and the vehicle diameter. The optimal vehicle presented (Column 3) has a higher curvature and hence, a higher drag due to faster acceleration of flow over the fairing, but this is balanced by optimal usage of the batteries which significantly reduces the weight of the system.

Table 17, compares the performance of the different MDAO architectures in terms of their performances. The table illustrates that the IDF implementation finds a good balance between the baseline implementation and the MDF architecture.

Table 17: A comparison of the different MDAO architecture implementations

Attribute	Baseline	Multi-Disciplinary Feasible	Individual Disciplinary Feasible
Execution Time	1 hr 37 min 46 sec	6 hr 59 min 45 sec	1 hr 47 min 13 sec
Number of Samples	5000	223	1677
Number of Consistent Design	0	206	379
Number of Errors	119	19	3

4.5 Design of an Unmanned Aerial Vehicle (Phase 2)

4.5.1 Problem Summary

As part of the Phase 2 hackathon, the SwRI team devised a design challenge in which the AI co-designers are tasked with designing multiple optimal UAVs to traverse a predefined path carrying a payload of 0.5 kg. The vehicle is required to drop-off the payload at the end of the path and traverse back on the same path. A bonus is given to designs that can traverse the entire path – back and forth – with a single charge of the batteries, i.e., under 80% of charge used. Additional bonuses are provided when the vehicle lands safely, i.e., with a velocity of under 0.2 m/s. The flight path consists of having seven (7) segments. An illustration of the traversal path is shown in Figure 103.

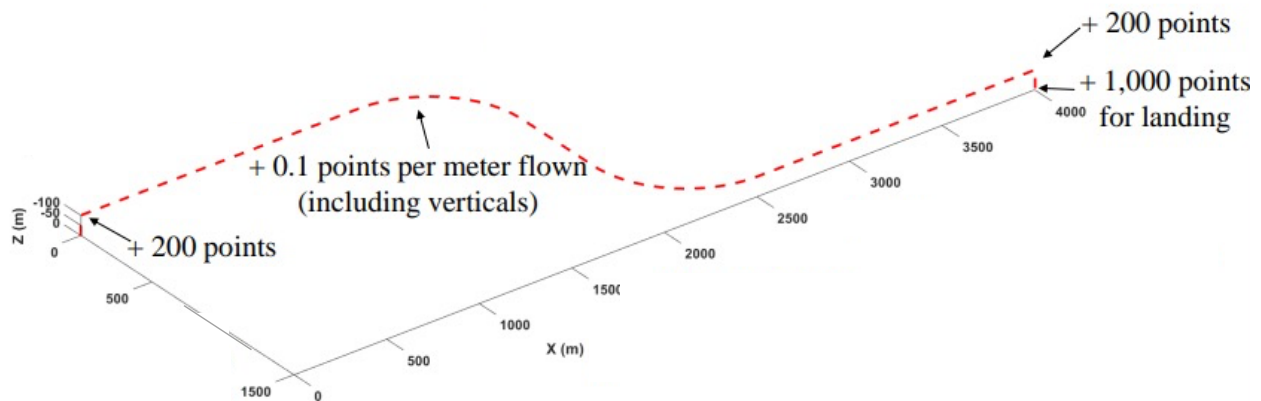


Figure 103: An illustration of the flight path for the design of an UAV in the Phase 2 hackathon

The details of the seven segments are:

- **Segment 1:** The UAV is tasked with a vertical ascent. The design must start from a rest position and ascend to a height of 100 m with under 30 m deviation from the flight path at any time. A score of 200 points is given to the design when it has successfully ascended to the required height of 100 m.
- **Segment 2:** The design is required to transition into a straight flight for a distance of 1500 m. During this straight path flight, the design transitions from a vertical acceleration to a horizontal acceleration. Along the path, the design is required to always stay within a box

of 15 m around the flight path. A score of 150 points is granted for successfully completing the segment at a rate of 0.1 points per meter flown.

- **Segment 3:** The design is required to execute a turn at a fixed radius. The design is tasked with executing a 500 m radius right quarter turn, totaling a distance of 785 m. In this segment, the design once more is allowed an error of 15 m boundary around the flight path. This segment too grants a 0.1 point per meter flown for a maximum of 78.5 points.
- **Segment 4:** The design is required to execute a 500 m straight flight. The constraints and setting of the flight are similar to that of segment 2, with the difference that the design is permitted to carry the momentum and acceleration from the turn into the straight flight path segment.
- **Segment 5:** The design is tasked with executing a left-handed turn at a fixed radius. The radius of the turn is 500 m, and the arc of turn is 90 degrees. As in the case of segment 3, the design is required to traverse the 785 m as quickly as possible while maintaining accurate navigational positioning.
- **Segment 6:** The penultimate segment of the flight path requires the design to decelerate over a straight-line distance of 1500 m. The requirement is that the horizontal acceleration of the vehicle should be as close to 0 as possible at the end of the path. This segment is a mirror opposite of segment 2.
- **Segment 7:** The final segment involves the design descending vertically from the 100 m height to the ground. The design is granted a bonus of 1000 points if the landing velocity is under 0.2 m/s. A further 200 points is given if the vehicle is successful in making the transition from a horizontal flight position to landing vertically.

The flight path is first executed with the design carrying a 0.5 kg payload and the same flight path with the payload removed. The removal of the payload is simulated by changing the mass of the payload to be 0.001 kg. If the design completes both the flight paths with under 80% battery consumption, the score is multiplied by a factor of 1.5. Additional points are provided to designs that complete the flight path in under 1000 seconds based on the speed of the time of completion. A breakdown of the scores achievable are shown in Table 18.

Table 18: Scoring criterion for Phase 2 hackathon

Scoring Condition	Points Achieved
Distance flown	0.1 point / meter
Transition from vertical to horizontal flight	200
Transition from horizontal to vertical flight	200
Successful safe landing	1000
Time bonus	1000 – time taken in seconds
Battery Usage Bonus	1.5x

Thus, the maximum total score realizable in the loaded and unloaded paths each is 2821, prior to the application of the battery usage bonus.

4.5.2 Design Methodology

We employ the same design process as in the Phase 1 end demo with improvements to the optimization algorithm and new surrogate models for the prediction of the UAV propeller thrust behavior. The surrogate models employed analyze the combination of propellers, battery and motors to predict the feasibility of the select combination considering the thrust produced at the maximum voltage – comparing it against the weight of the vehicle; the maximum power consumed by the propeller – comparing it against the maximum power of the motor; and the maximum current used by the propeller / motor combination – comparing it against the maximum current supplied by the battery. The surrogate model is built using a single propeller model in which a single battery, propeller and motor combination is used as illustrated in Figure 104.

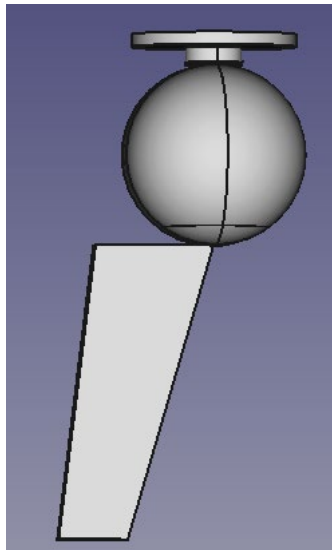


Figure 104: A conceptual single propeller-motor-battery combination model

Over 2,000,000 samples of various combinations are created to model the thrust behavior of the propeller. The model is parametrized by the voltage output of the battery; the characteristics of the motor such as the K_t , K_v , maximum current, maximum power, and the internal resistance; and the characteristics of the propeller such as the mass, the pitch, and the diameter. 1,400,000 samples are used to train a regression model. When evaluated, we observe very good prediction capability with an R^2 of over 0.99, illustrated in Figure 105.

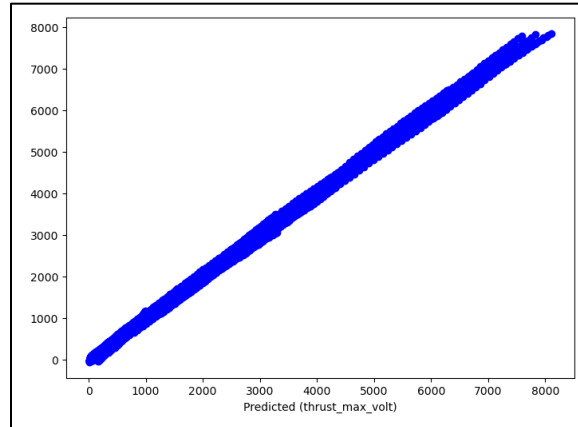


Figure 105: Actual vs Predicted Thrust produced by a combination at maximum voltage

The model is then used in an auto-completion engine that is used to predict the number of propeller and corresponding motors required to achieve successful trim. Once the vehicle is trimmed, an optimizer subsequently optimizes the scalable characteristics to achieve the best performance for the vehicle.

4.5.3 Results

As in the case of the Phase 1 demo, our design auto-completion process generates designs that respect the structure provided by the user while significantly improving the performance of the resultant designs. The seed designs and their respective performances are presented in Table 19. These are contrasted with the top 3 designs that are generated by the Systemic Generative Engineering framework, shown in Table 20. Finally, we illustrate that the designs generated by our design process are significantly better performant when compared against those created by expert designers who manually constructed the design over a 4-week time-period (from the end of the hackathon to the reporting date). The design created by the expert is illustrated in Table 21.

Table 19: Seed UAV designs summary

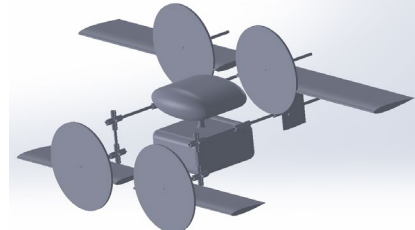
Design	Name	Loaded	Unloaded	Total
	Pick Axe	926	926	1852

Table 20: Top generated designs using the Systemic Generative Engineering framework

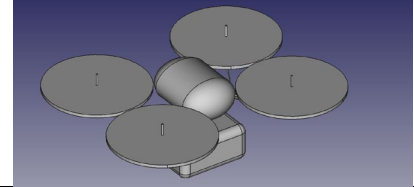
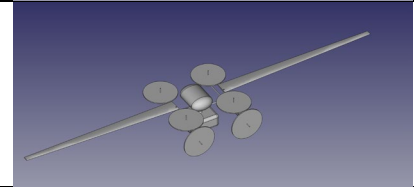
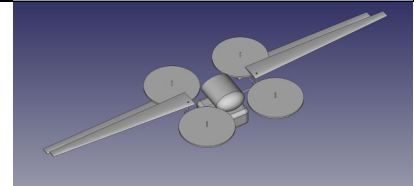
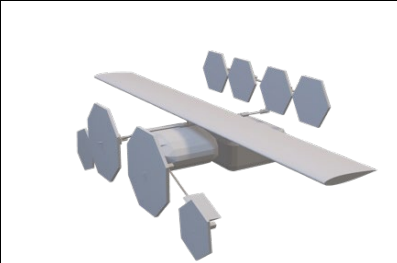
Design	Loaded	Unloaded	Battery Usage	Total
	2794	2793	0.73	8380.5
	2696	2691	0.82	5387
	2615	2615	0.87	5230

Table 21: Expert generated best design.

Design	Name	Loaded	Unloaded	Battery	Total
	Wedge	715	2451	Un-known	3166 (4749)

As in the case of Phase 1 end demo, the best design identified is extremely robust over, once more, all 150 design conditions (flight velocities of 1 – 50 m/s in the three flight paths – lateral flight, turning flight of 300 m, and turning flight of 500 m). This implies that the design process and the Systemic Generative Engineering framework is extremely good at identifying robust designs across design considerations. This is in part due to the formulation of our design objective as a multi-step process in which trim performance of the vehicle is paramount importance.

4.6 Design of an Unmanned Underwater Vehicle (Phase 2)

4.6.1 Problem Summary

As part of the Phase 2 hackathon, the STR team updated their design challenge to change the trajectory of the path to be taken by the UUV. The UUV is required to traverse around Queen Elizabeth Island for a distance of around 4300 km. The vehicle must travel through a set of 125 waypoints localizing by sounding to 11 beacons, each located around 20 km up from the mouth of the fjords. These fjord mouths are included among the 125 waypoints and are provided within the simulator library of missions. The mission path is illustrated in Figure 106.



Figure 106: Illustration of the mission path around the Queen Elizabeth islands where the OBS payload must be dropped off at (80° 23' 12" N, 85° 06' 46" W) at a depth of 547 m.

Knowing the 20 km distance to the sounding beacons, we were asked to localize within an uncertainty circle of 100m using a sonar line array of n hydrophones, using the angular resolution of the line array as a simple proxy metric for the localization error. As far as the simulator is concerned, all you will need is to be within standoff range to provide effective localization, which will be a function of the number of elements n .

As part of this design challenge, the AI co-designers are required to complete the conceptual design of a long distance, with a vehicle that roughly covers a 4300 km distance. As in the case of Phase 1 demo, the vehicle must carry an OBS payload that is required at the target location at waypoint 16 along the transit path. The vehicle is then required to return to the start location, and complete the entire trip, to and back, within 48 days.

4.6.2 Design Methodology

Our design methodology follows that presented in the Phase 1 demo. We augment the simulation toolchain with additional considerations to include the design of the battery pack, that of the internal structures, and the selection of the buoyancy and hydraulics devices to go onboard the vehicle. The simulation toolchain is illustrated in Figure 107.

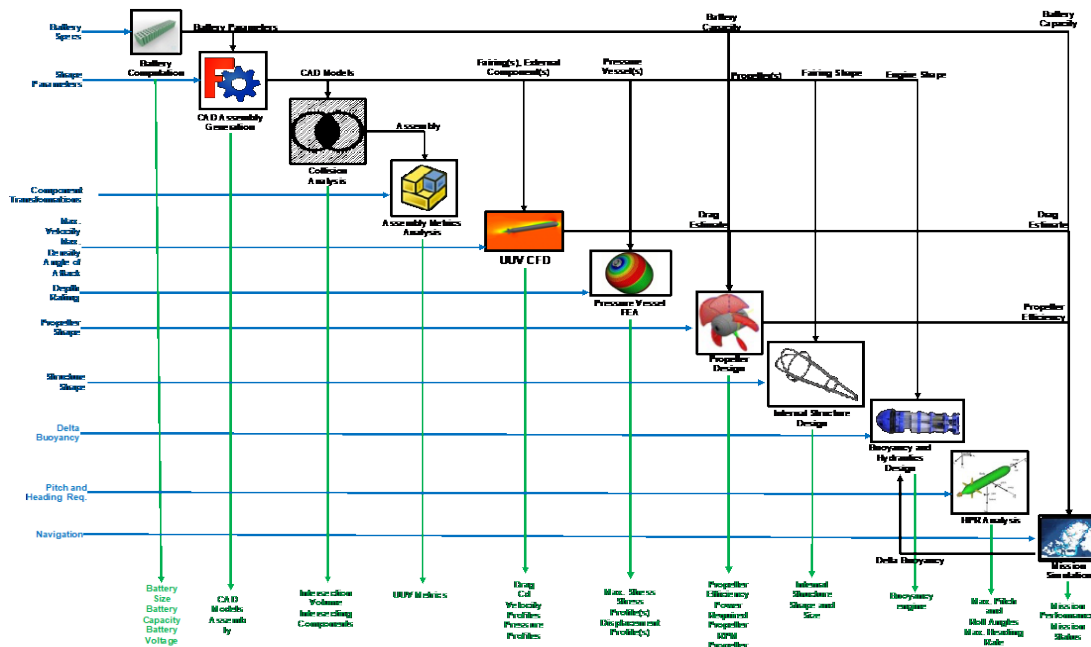


Figure 107: Design process used in the evaluation of the performance of the UUV design.

The key changes to the design flow in comparison to that used in Phase 1 demo is the incorporation of a detailed battery pack sizing analysis, the inclusion of the internal structure design tool, the consideration of the buoyancy and hydraulics design tool and the design of a detailed controller for the specified mission.

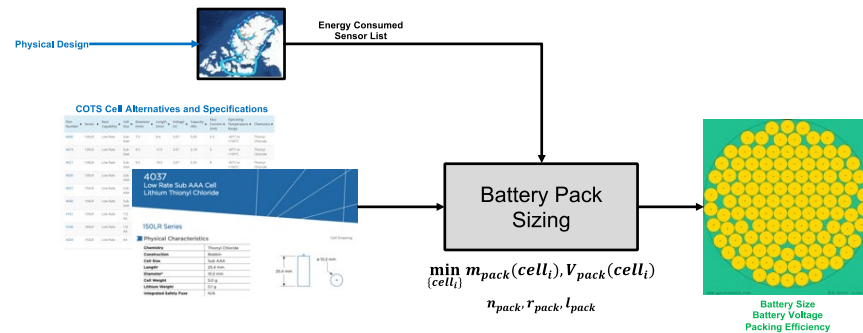


Figure 108: Battery Pack Sizing Considerations

- Battery Pack sizing:** As part of the energy analysis, we considered a set of 88 commercial-off-the-shelf battery cells for selection. The battery cells are evaluated such that we minimize the mass and volume of the entire battery pack while meeting the battery energy specification provided by the mission simulation. We develop an automatic parsing tool that automatically extracts information about the battery cells from the public domain, e.g., [98], and solves a discrete optimization problem in which the number of cells and the cell type is determined. An illustration of the battery pack design process is shown in Figure 108.

- Internal Structure design: Once the fairing is shaped to minimize the drag of the vehicle, the internal structure is automatically constructed from a set of specifications of the designed fairing. The design of the internal structure is carried out such that the stresses associated with the internal frame of the vehicle due to weight of the vehicle and the buckling effects observed are handled. We ensure that the maximum von-Mises stress is below the permissible threshold while the buckling factors are close to or under 1.0. A finite element analysis is carried out to evaluate the structural behavior of the internal structure to quantify the designed performance.
- Buoyancy and hydraulic device considerations: Based on the volume displacement requirement of the vehicle, we identify a commercially-off-the-shelf buoyancy engine and hydraulic motor that is added onto the list of devices to be incorporated into the vehicle. The oil reservoir, motor, valves, and the actuators are not designed, but chosen from a list of alternatives. These components are placed in pressure vessels while the bladder is a wet device.
- Controller Design: Our simulation pipeline integrates the mission simulator provided by STR as part of its design evaluation. The mission simulation not only evaluates the performance of the design for energy consumption, but also evaluates the designed controller for navigational performance. The controller design is carried out using expert judgement where a rule-based system is used to design the changes in control policy across the different positions in the vehicle's transit.

We employ reduced order modeling to replace the expensive finite element simulations of the internal structures with simpler 1D elements. The conversion from 3D to 1D models is carried out automatically without any human in the loop. The resultant design process having incorporated the AI elements is illustrated in Figure 109.

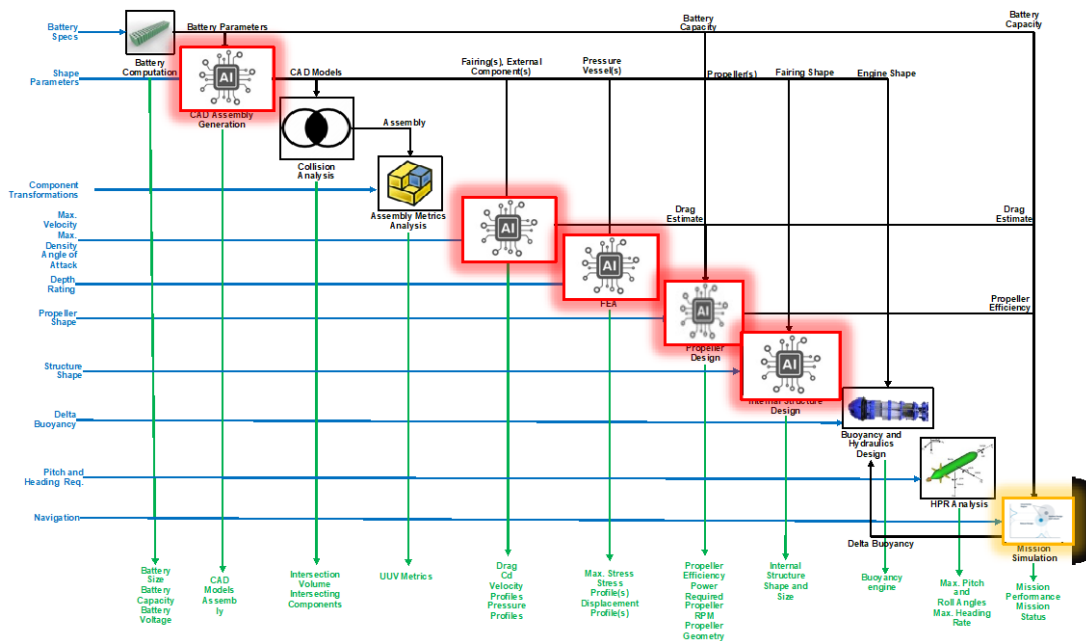


Figure 109: Design process incorporating AI methods to accelerate the simulation of the design.

We reuse AI models and methods developed in the Phase 1 demo for other elements of the design process. On the other hand, the design process is altered considerably to leverage a collaborative and concurrent subspace optimization MDAO setup. The design setup is illustrated in Figure 110. Here, we breakdown the entire DSM for the UUV design problem into several subspaces. The subspaces are: (i) hydrodynamics; (ii) structural mechanics; (iii) internal structural mechanics; (iv) CAD and assembly; and (v) controller design. We optimize each subspace independently with a global controller (i.e., collaborative optimization) handling the coordination and consistency between the different optimizers. The subspaces are augmented with additional supporting analysis that do not directly feed the optimizer but are important for the reporting metrics. These are the HPR stability analysis, the assembly metrics computation and the buoyancy and hydraulics design.

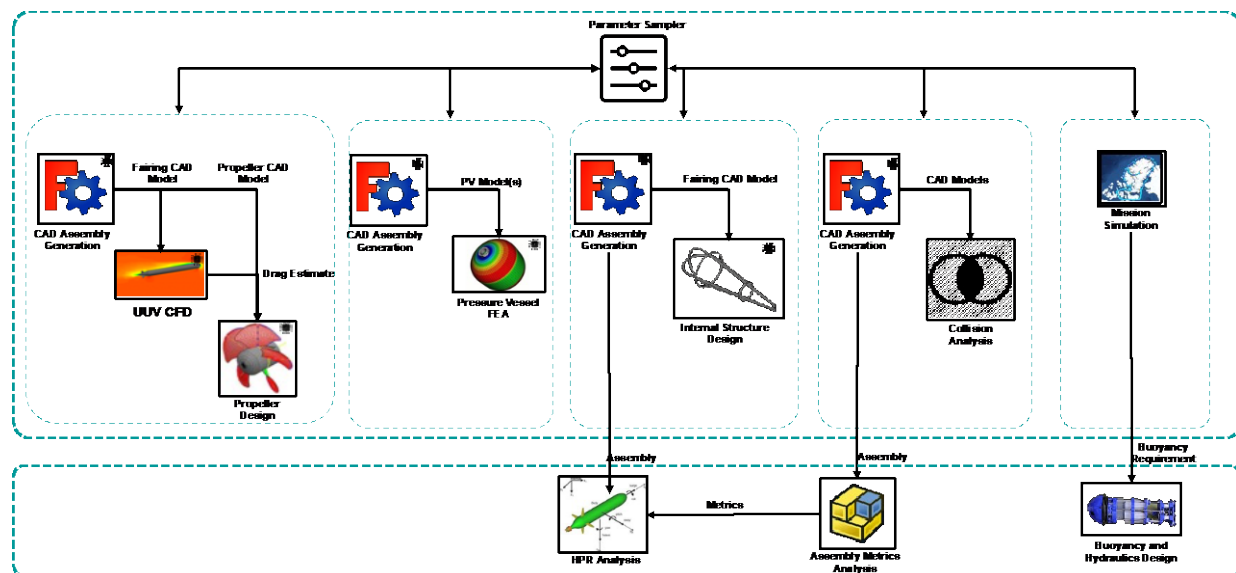


Figure 110: Subspaces considered in the design optimization of the UUV.

Figure 111 illustrates the optimization strategies utilized and the AI models considered as part of the design optimization process.

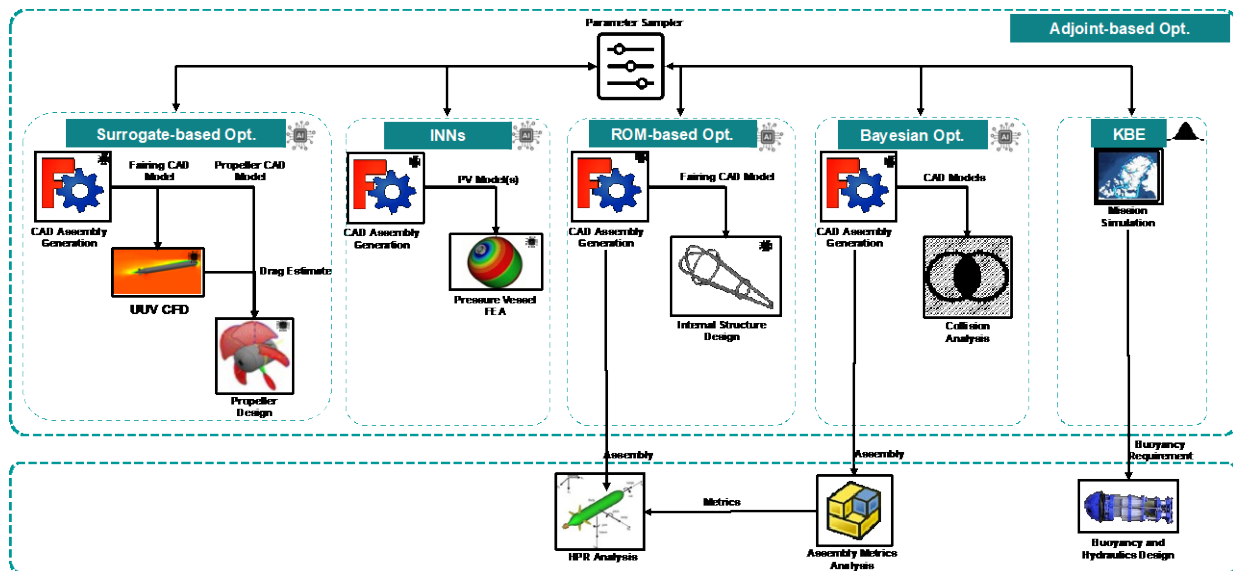


Figure 111: Illustration of methods and algorithms used in the subspace and collaborative optimization

We explored a variety of different methods to optimize different subspaces of the design. For example, a surrogate-based optimizer is used for the optimization of the fairing of the design and the propeller operating characteristics. An invertible neural network that works on a data-driven reduced order model is used to optimize each of the pressure vessels in the vehicle. A gradient-based optimizer building on model order reduction methods is used to optimize the internal structure of the design. Symmetry-constrained Bayesian optimization is used to optimize the 3D placement, packing and assembly of the vehicle. Finally, we use a rule-based design method to design a controller for the mission under consideration.

4.6.3 Results

The design process was applied over the course of a week to synthesize a design of the UUV. The designed UUV is illustrated in Figure 112 and the characteristics of the UUV are highlighted in Table 22.

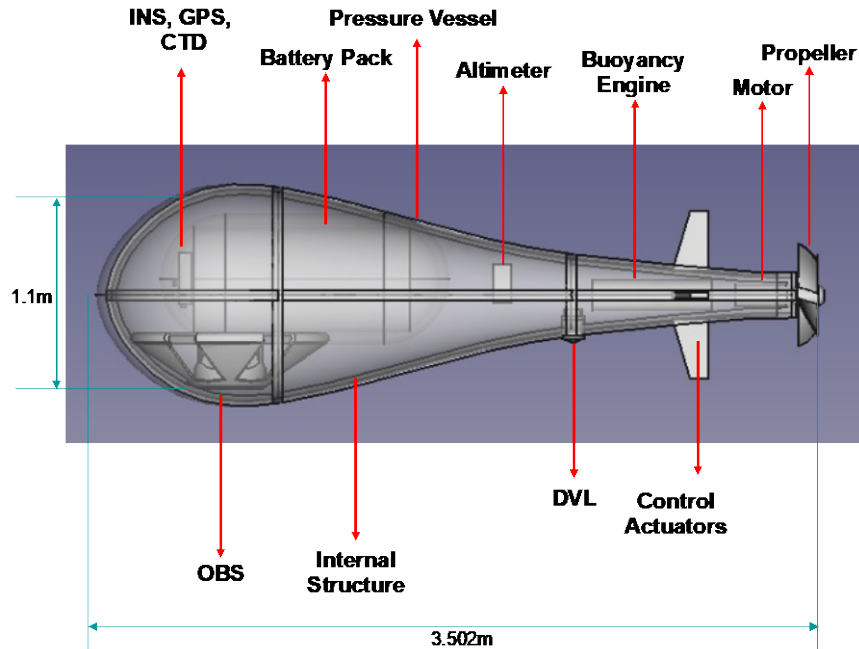


Figure 112: Illustration of the UUV along with the comprised components identifying their positions

Table 22: Characteristics of the designed UUV

Property (Unit)	Value
Mass (kg)	1422.3708
Fairing Length (mm)	3502.5
Displaced Volume (m ³)	1.3386 / 1.405
Drag (N) @ 1 m/s, 0 AoA, 1027 kg/m ³	52.5256 (w/o towed array) (OpenFOAM drag: 26.2633)
Powertrain Efficiency (-)	0.4 (assumed)
Center of Buoyancy (m)	[1.7402, 0.001, 0.0145]
Center of Gravity (m)	[1.7412, 0.0, -0.020]
Energy Capacity (kWh)	358.8
Buoyancy Compensation (kg)	69.58
# Towed Array Elements (-)	60

The controller design of the vehicle yields the following performances where we observe that the energy characteristics of the vehicle is optimized with negligible remaining battery energy. The performance of the controller against the waypoints along with the observed velocities, and the vehicle's energy consumption is illustrated in Figure 113.

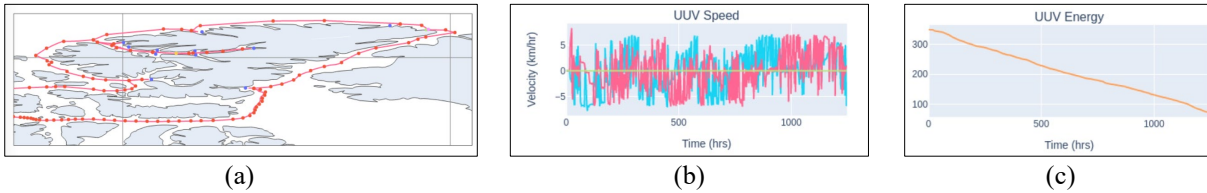


Figure 113: Designed controller performance illustrating the successful ability to meet all the waypoints and complete the mission with nominal unused energy remaining

4.7 Hardware-Software Co-Design of Deep Neural Network Accelerators

4.7.1 Introduction

The proliferation of industrial use of Deep Neural Networks (DNNs) has prompted considerable research into the design of specialized hardware accelerators, manifesting as Application-specific Integrated Circuits (ASICs) and Field-programmable Gate Arrays (FPGAs), to meet the strict needs for Power, Performance and Area (PPA), with a heavy focus on the latency and throughput of these models. Such hardware accelerators have enabled the application of DNNs to a variety of products that support tasks ranging from consumer needs involving autonomous driving, recommender systems, speech, and image recognition, etc. to matters of national security such as cyber-threat detection, entity recognition, and intelligence, surveillance, and reconnaissance (ISR).

Designing efficient ASICs for the purpose of DNN acceleration is a highly multi-disciplinary challenge where the design of hardware, software, and their interface (i.e., the dataflow) are all of equal importance. The inertia of traditional engineering practices, where siloed discipline-focused design methods are employed, limits the ability to meet the demands for high-performant ASICs due to long and iterative design cycles. Such design cycles involve trade-offs in hardware and model performances while jointly optimizing the resultant PPA. The design spaces associated with such an exploration is of mixed modality, i.e., involving both discrete and continuous design choices, and the set of discrete design alternatives can range from a few million to 10^{36} [1] and more.

The design challenge associated with this problem can be viewed as being threefold:

- The design tool chains that evaluate the DNN ASICs performance cannot be easily automated.
- The simulation cost associated with these design tool chains are large preventing rapid evaluation and comparison of the alternate designs.
- There is considerable domain knowledge and expert judgement required to efficiently explore these vast design spaces.

Recent investigations into the simulation of hardware accelerators such as MAESTRO [2], ASTRA-sim [3], Acceleergy [4], and SCALE-sim [5], have, to an extent, addressed the challenge of computational expense of the design process. These simulators can be viewed as system-level simulators that employ an abstracted description of the hardware accelerator, the DNN model and the associated dataflow to provide quick feedback on their design performances.

Furthermore, when coupled with exploration methods such as GAMMA [1] and Di-GAMMA [6], researchers have been able to demonstrate reasonable capability in the exploration of this complex design space. However, a key assumption made by these algorithms that explore the design space

is that the system-level architecture of the hardware accelerator is fixed. While the space of hardware characteristics (e.g., bit-width, bandwidth, etc.) and the dataflows have been explored to some detail, that of the hardware architectures has not yet been explored.

A recent development, Timeloop [7], presents an avenue to address this challenge by developing a description language for the hardware architecture, albeit in a limited and constrained setting. On the other hand, with growing model complexities and increasing energy efficiency demands, there are still considerable limitations with even the most efficiently designed hardware accelerators. This, coupled with the looming inability to realize the benefits of Moore's Law scaling, has motivated the push towards the development of Three-dimensional Integrated Circuits (3D-ICs) for the acceleration of DNNs. Several recent works [8, 9, 10] have proposed innovative implementations of DNN on 3D-ICs utilizing a variety of 3D stacking technologies ranging from heterogeneous stacking to fully homogeneous ones. While there have been considerable investigations automating [11, 12] the physical design, i.e., RTL-to-GDS, their application to the hardware accelerator design is still lacking. This is further complicated by the fact that evaluations of benefits that can be gained by changes to the system architecture for the 3D-IC is still understudied, with no tools providing the capability to automate this exploration.

Finally, 3D-ICs, in general, offer several additional dimensions and challenges to the design problem by not only exponentially expanding the design space, adding new degrees of freedom, but also introducing multi-physics considerations involving thermal management, efficient packaging, routing through layers, etc. These considerations involve simulations and computations that are prohibitively expensive for consideration at the conceptual design stage, i.e., system architecture design. Nonetheless, the choice of system architecture strongly influences the optimality of the resulting 3D-IC. Hence, it is critical to be able to incorporate these considerations of advanced simulations earlier in the design process.

The presented considerations necessitate a need for an integrated Modeling and Simulation (M&S) methodology that permits the exploration of hardware system architectures while also evaluating the associated performance through its realization, i.e., the hardware layout, its configurations and associated dataflow. In this work, we present such an integrated framework at the core of which is an efficient representation of the 3D-IC system architecture. The framework leverages machine learning techniques to efficiently explore the space of system architectures to jointly optimize the ASIC hardware topology, their configuration, and the associated dataflow.

The contributions of the current work are summarized as follows:

- **Efficient representation of 3D-IC system architectures:** Our methodology leverages concepts from representations developed for complex Cyber-Physical Systems and developed a description language that can be used to construct any arbitrary hardware architecture, 3D or otherwise.
- **Multi-fidelity Modeling and Simulation:** Our methodology enables the consideration of higher fidelities earlier in the design process by leveraging RTL-accurate modeling of elementary IP blocks and reusable chiplets in the architecture exploration stage without compromising on design exploration duration.
- **Efficient Exploration Strategy:** Our methodology leverages generative design-based exploration strategy that uses a mixture of machine learning and domain knowledge to synthesize optimal hardware accelerators.

4.7.2 Methodology Background

Design flow for DNN accelerators: The design of DNN hardware accelerators is a multi-faceted problem involving considerations from various disciplines including the DNN software model, its preparation for synthesis to hardware, and the subsequent synthesis of the hardware. An abstract representation of the process is illustrated in Figure 114. The process begins with data scientists / machine learning engineers leveraging frameworks such as Tensorflow, PyTorch, etc., in a high-level language such as Python to design DNN models that are trained on server-class hardware with GPU support. These models are trained with floating-point arithmetic with the goal of maximizing the model accuracy.

Once trained, the model is optimized, by machine learning engineers, for hardware deployment with considerations such as quantization, sparsity, and layer aggregation, etc., with an attempt to maintain the model's accuracy while minimizing the number of computes to be performed. The software optimized model is next prepared for hardware deployment which involves three steps, (i) a design of the hardware architecture; (ii) the High-level Synthesis (HLS) of the model for the designed hardware architecture using design IP; and (iii) the design of the physical layout of the hardware. These steps find the right balance between the prepared model's accuracy against the desired latency, throughput and PPA once deployed onto the hardware.

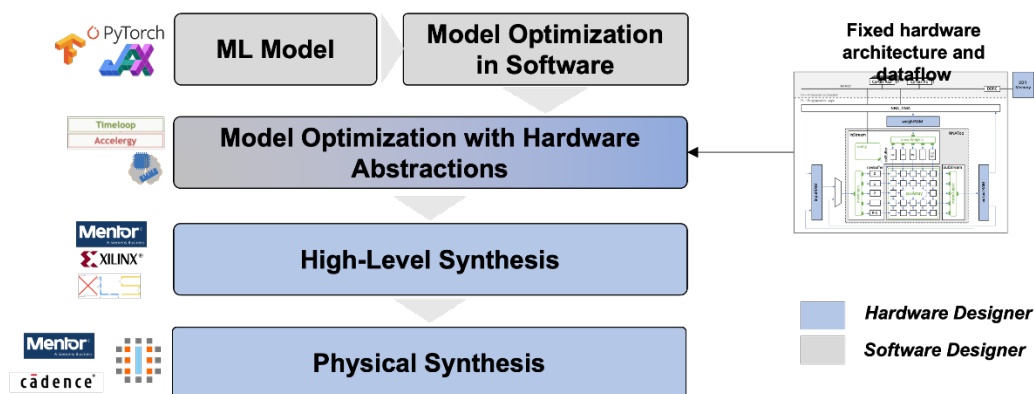


Figure 114: An abstract illustration of the steps involved in the design consideration of DNN hardware accelerator

The process ends with the physical layout resulting in an optimization implementation for FPGA or ASIC. From a hardware design perspective, the PPA computation is often carried out at either the HLS or the physical layout stage. Works such as [2,7] have investigated the ability to execute similar PPA analyses at the architecture level, but at the cost of considerable accuracy in the associated performance estimates. These works often require a specification of the hardware architecture, and explorations are only restricted to parameters in the mapping space, and/or a handful of hardware configurations (e.g., the processing element (PE) array size and buffer sizes).

Design considerations for 3D-IC: The move towards 3D-IC for DNN accelerators necessitates considerations that are non-traditional in typical DNN accelerator design workflows. These include ensuring the thermal and mechanical performance of the resultant designs, accounting for routing and interconnects across tiers, ensuring appropriate placement-based partitioning of logic

gates, etc. These new design considerations are illustrated in Figure 115. The new design considerations are post-hardware architecture definition, and hence any modification to the hardware architecture during a design space exploration has more significant impact on the design.

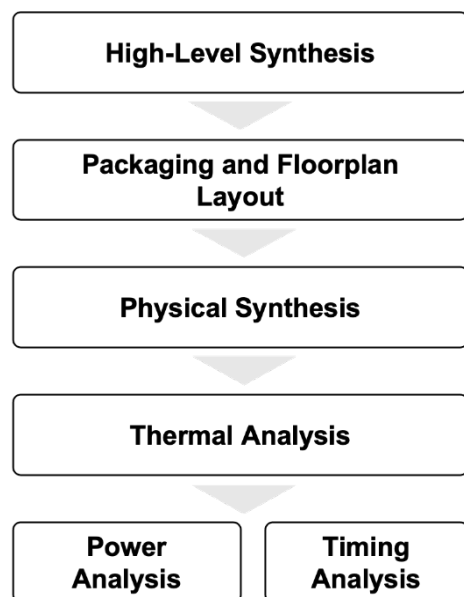


Figure 115: An illustration of the steps involved in the design workflow of 3D-ICs

A secondary consideration that directly affects the architecture design of 3D-ICs is the fact that these devices require a consideration of the mechanism for additional electrical pin connectors such as micro-bumps, through-silicon-vias, and interposers. Furthermore, bonding technologies and types introduce another degree of design freedom that increases the complexity of the design space.

Graphs as a Representation of a System Design: A system design can be represented as a graph, where the nodes represent the components of the system, and the edges represent the connections between the components. This concept is well established in the field of system design (e.g., SysML). In such a design task, an optimization algorithm is tasked with finding the optimal system graph that satisfies the design constraints while maximizing its objectives. Several works have explored the use of graphs as a representation of a system design, with the concepts being also available in both open source [13] and commercial products [14]. In the field of DNN accelerator design, the use of graphs to represent the hardware architecture is also well established, as in the case of Timeloop. Similarly, there are several works that

employ graph representations for high-level synthesis [15], several of which can also be applied to DNN accelerator design.

Generative Design: Generative design is the process of utilizing a procedural generator to synthesize product designs in an autonomous manner. While the term generative design has predominantly been used as a synonym for topology optimization, the concept of generative design extends to any generic product design. Generative design there exists a generator, e.g., an AI model, from which a variety of design can be sampled. The key task of a design workflow that employs generative design is defining the design generator in such a way that it has applicability to a variety of design problems. Typical methods to create such procedural generators can be machine learning-based, rule-based, contract-based, enumeration-based, or even random generators. The structure of the design generated can vary from complex 3D geometry to system topologies to even software implementations (e.g., program synthesis). These methods of generative design, owing to the ability to autonomously synthesize designs, are very amenable to design space exploration. There are several techniques that combine the concepts of generative design and design space exploration. These vary from incorporating design feedback to the procedural synthesizer (e.g., design objectives and constraints) with sequential updates to the sampling mechanism (e.g., optimization) to training conditional generative models to automatically synthesize designs that satisfy a handful of constraints in a single inference step.

Design Grammars: Design grammars are one of the possible techniques to enable generative design. A grammar is a set of synthesis rules that provide guidance to the manner in which designs are generated in a procedural manner. The concept of grammar-based synthesis originates from linguistics and language processing, but such methods have recently been used in a variety of applications such as robot design [13], polymer design [16], and unmanned vehicle system design [17]. In the context of system design, the set of synthesis rules of interest are those that result in a set of valid graphs that describe the system. It is important to note that a design grammar by itself does not yield a design. Each synthesis rule in the grammar only defines an atomic mutation to the design from its current representation to a new representation. Thus, in order to generatively synthesize a design from a grammar, there is a need to optimally select the sequence of synthesis rules that starts from an initial state of the design.

Monte Carlo Tree Search: MCTS is a stochastic search algorithm widely used across a variety of problem domains for optimization. MCTS combines Monte Carlo strategy, which relies upon random sampling and statistical evaluation, with tree based search techniques. Instead of performing exhaustive exploration of the entire search space, MCTS operates by identifying the potential best moves and then expanding the search tree. MCTS has been applied in Electronic Design Automation (EDA) space for design space exploration and optimization. Wavefront [18] applied a variant of Monte-Carlo Tree Search to solve Network-on-Chip design optimization problems with multiple objectives. The approach defined search space in the form of a Markov Decision Process (MDP) with graph rewriting rules and then applies weighted reward function to evaluate the terminal states. While several variations of MCTS exist, the basic formulation of the problem remains the same.

4.7.3 Design Process

In this work, we developed an integrated design process that permits the use of higher fidelity models during the conceptual system architecture design. The new design process is illustrated in Figure 116. The design process builds on a library of reusable components, i.e., chiplets, supplied by the design team, from which the hardware architectures are synthesized.

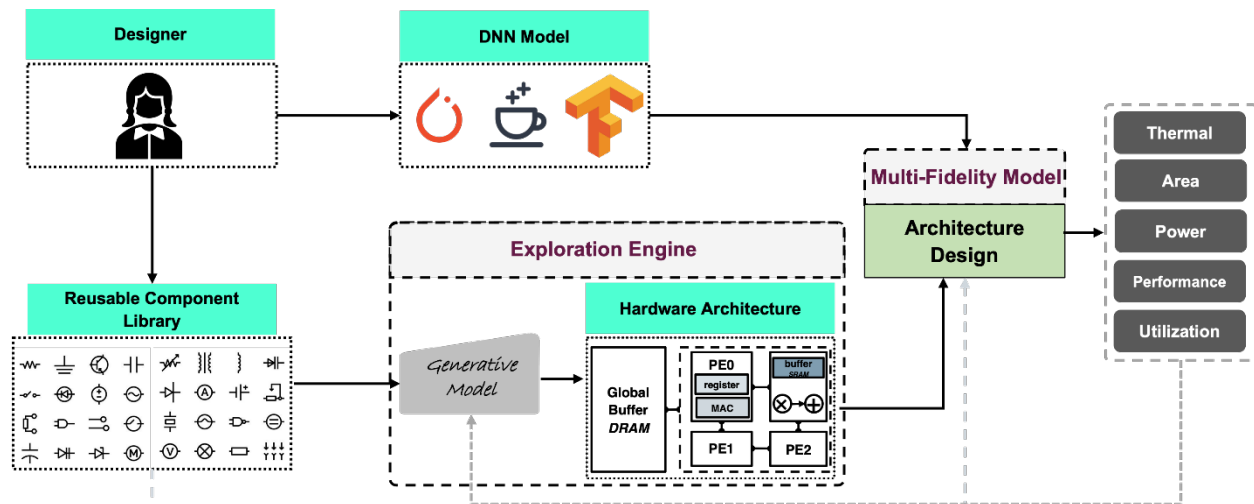


Figure 116: The proposed integrated hardware accelerator design framework

The synthesis is guided by a design grammar, which too is defined by the designer. The grammar is coupled with an exploration engine to synthesize the hardware architecture. Our multi-fidelity M&S engine evaluates the hardware architecture, estimating its PPA characteristics. The estimated PPA characteristics feedback to the exploration engine to update the system architecture, optimizing the hardware design.

Multi-fidelity Modeling and Simulation: The core element of the framework is its ability to incorporate higher-fidelity considerations earlier in the design process so that more informed decisions can be made. While analytical models such as Timeloop / Accelergy / SCALE-Sim offer a quick evaluation of the PPA characteristics of the design, they struggle with being sufficiently accurate. These inaccuracies often result in the discovery of design flaws, necessitating a redesign, which can be quite time-consuming. On the other hand, higher-fidelity methods that leverage RTL-level or physical layout level analyses can offer higher accuracy in the estimation of the PPA characteristics, at the cost of speed, and also requiring considerably more domain knowledge in executing the complete high-level synthesis and place-and-route workflows to reach the physical layout of the hardware. With consideration of design of optimal 3D hardware accelerators for DNNs, a key factor dictating the choice of the simulation model is the vastness of the design space. These shortcomings imply that there is a need for a good balance between the speed and accuracy of the simulation model, particularly for the task of design optimization. Hence, we introduce a multi-fidelity modeling framework, illustrated in Figure 117, that can provide quick feedback on the design performance while being accurate enough in its performance estimations.

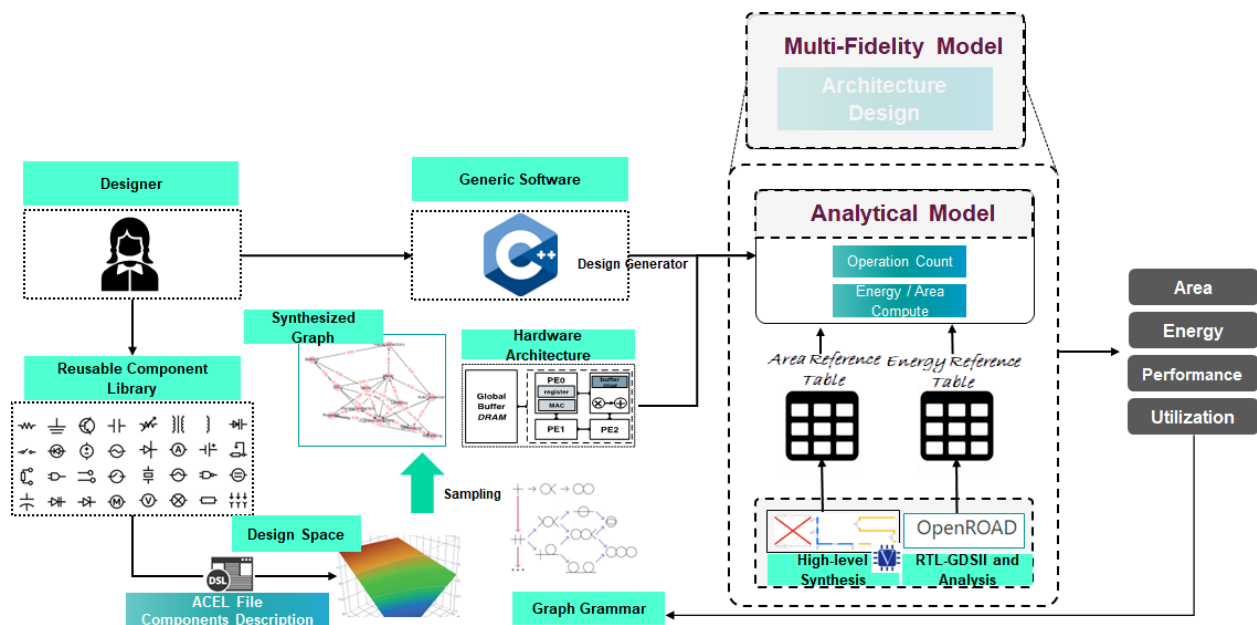


Figure 117: The M&S steps involved in the multi-fidelity design flow.

This simulation environment builds on the fact that our designs are constructed using reusable functional elements, e.g., chiplets. Thus, once the available set of chiplets are defined, an offline one-time estimation of their area and power characteristics is carried out using high-level synthesis and physical synthesis workflows. This information is stored in a database of available components and is called upon when the performance estimation of the entire system is performed. Therefore,

the performance estimate of the entire design is estimated through composition of the performance estimates of atomic lower-level blocks.

As illustrated in Figure 117, the M&S environment builds on existing open-source high-level synthesis tool - XLS [19], physical synthesis tool – DARPA funded tooling OpenROAD [20], and geometric modeling tool, FreeCAD [21]. We emphasize that the tools used in the pipeline are replaceable with other alternatives such as Catapult HLS [22], Xilinx Vitis [23], Aprisa PnR [24], OpenROAD [20], NX [25], and similar tools.

The M&S process follows the power estimation pipeline presented in OpenROAD and begins with the designer supplying a set of reusable elements. These reusable components are implemented as atomic elements (e.g., 4-bit adders, left shift register, etc.) in a high-level language, such as C++, or using a domain specific language, such as XLS or DSLX. These high-level implementations are synthesized to (System)Verilog. The synthesized Verilog is transformed to RTL which is then synthesized to binary sequential logic circuits. The synthesized logic circuits are used to estimate the utilization within the circuit that supplies the activities within the circuit at a netlist level to a power estimation module. Finally, using the estimated circuit activities, a power estimation step is carried out where a circuit packing and place-and-route operations are carried out at the netlist level. This step estimates the total power consumption of the synthesized physical circuit while also providing detailed decomposition of the associated area.

```
#####
system EDA_System_Standard:
...owns:
.....StorageComponents
.....ComputeComponents
.....ConnectionComponents
.....MiscComponents
.....BaseComponents
.....ChipletSubsystem
.....PESubsystem

#####
#
#####
collection StorageComponents:
...owns:
.....Register
.....RegisterFile
.....SRAM
.....DRAM
.....FIFO

component::StorageComponents.Register:
...ports::individual:
.....io::mechanical.Connector
.....io::electrical.Storage_Connector
...attributes::constants:
.....int|none|datawidth
.....int|ns|latency
.....str|none|action
.....str|none|CADMODEL
...variants:
.....variant.reg_16dw:
.....datawidth.=.16.|none|
.....latency.=.5.|ns|
.....action.=."access".|none|
.....CADMODEL.=."None".|none|

component::StorageComponents.RegisterFile:
...ports::individual:
.....io::mechanical.Connector
.....io::electrical.Storage_Connector
```

Figure 118: Design space representation using ACEL for IC

Simultaneously, these physical implementations are converted to electrical computer-aided designed (ECAD) representations which are used for thermal analysis. This entire process is executed for every component supplied by the user in the library of reusable components with the estimates of thermal, power and area estimates stored for use in the synthesis of the hardware accelerator. It is important to note that the components are meant to be atomic elements that are composed into 3D-ICs, and hence themselves are traditional 2D implementations.

Having generated these performance estimates for atomic components, the M&S calls upon them when evaluating the performance of a new hardware accelerator. This process is carried out using open-source frameworks such as Timeloop, STONNE [26], etc. that provide a cycle-level architecture simulation for the DNN accelerators. The process is carried out by having the user or an exploration algorithm supply a hardware architecture, the dataflow, and the problem description to the cycle-level simulators. These simulators compute the number of operations performed on each atomic block in the provided hardware architecture and using the component database as a look-up table, compute the total power consumed by the resultant hardware. The physical area is computed by the ECAD generator that computes the resultant area of the assembled set of components based on the interfaces provided. Finally, the

thermal performances of the atomic components used as heat sources to build a reduced order model of 1D elements to estimate the total temperature changes across the entire 3D-IC stack, providing quick feedback on the overall stack's thermal performance.

Graph Grammar-based Design Synthesis: Given the set of elementary IC components, i.e., chiplets, the goal of the design optimization is to explore, efficiently, over the set of possible 3D-ICs that can be synthesized by composing these chiplets together. However, the space of possible combinations is extremely large, and compositions must respect certain structures to form realizable ICs. Hence, we employ a graph grammar to provide the necessary structure to the design space by enforcing constraints on the way the 3D-IC designs can be synthesized. The set of synthesis rules contained in the graph grammar are manually defined a priori to the exploration process.

```
digraph::make_ic.{
  ..subgraph L.{
    ....IC.[require_label="IC"];
    ..}
  ..subgraph R.{
    ....Architecture.[label="architecture",
      base=true];
    ..}
}

digraph::expand_architecture.{
  ..subgraph L.{
    ....Architecture.[require_label="architecture"];
    ..}
  ..subgraph R.{
    ....Architecture.[label="ic_architecture",
      base=true];
    ....Chiplet.[label="base_chiplet"];
    ....Storage.[label="global_storage"];
    ....Counter.[label="counter",
      compclass="counter"];
    ....Adder.[label="adder", compclass="adder"];
    ....Architecture->.Chiplet;
    ....Architecture->.Storage;
    ....Architecture->.Counter;
    ....Architecture->.Adder;
    ..}
}

digraph::global_storage_DRAM.{
  ..subgraph L.{
    ....Global_Storage.
      [require_label="global_storage"];
    ..}
  ..subgraph R.{
    ....Global_Storage.[label="DRAM", depth=
      1024, width=32, datawidth=16,
      compclass="DRAM", storage_type="LPDDR4"];
    ..}
}

digraph::global_storage_SRAM.{
  ..subgraph L.{
    ....Storage.[require_label="global_storage"];
    ..}
}
```

Figure 119: Graph grammar used for design generation.

We choose to represent the 3D-IC design as a graph in which each node forms a physically realizable elementary IC component from the user-supplied database. We use a textual representation of design space using ACEL domain specific language as show in Figure 118 for the 3D-IC use case. A set of interfaces (i.e., connection ports) are associated with each of these components such that physical and electrical connections can be established between components. These connections form the edges of the design graph, with the physical connections made used to directly and automatically construct the geometric representation of the 3D-IC. Figure 119 illustrates the grammar which is used to sample and generate the designs.

Design Space Exploration: While the defined grammar enables the procedural generation of several designs, it needs to be combined with the M&S environment to optimize the generated design for the task at hand. In our implementation, we employ an MCTS algorithm to explore the space of 3D-IC architectures. Here the MCTS optimizer is tasked with identifying the optimal sequence of synthesis rule that must be applied to an initial empty design graph to maximize the performance of the 3D-IC design. The MCTS algorithm utilizes the sequence of selected synthesis rules as an indication of the state of the environment with the action space being the set of available synthesis rule.

4.7.4 Results

We validate our framework by generating system architecture design for two problem spaces. First, as a proof of concept for a simple adder for adding two numbers. The second for a real-world use

case in the context of co-optimization of UUV and a navigation system relying on image processing. We generate an optimal hardware architecture design and corresponding data mapping for the image processing problem using the framework. We apply the integrated design process specified as described in section 4.7.3 that permits the use of higher fidelity models during the conceptual system architecture design.

The first step of our methodology entails high fidelity RTL and Graphics Systems Design II (GDII) simulation and analytics for the components from the library utilized by the grammar. From the architectural component definition, we first generate a DSLX or C hardware definition file. The definition is then used to generate an intermediate representation and finally a Verilog definition using XLS. The Verilog hardware definition is passed as an input to OpenROAD pipeline which performs RTL and Place and Route (PnR) design generation steps and executes simulation for analyzing the PPA results. Figure 120 and Figure 121 show the results for XLS and OpenRoad simulations executed for Adder and Counter components.

```
=====
finish report_design_area
-----
Design area 4090 u^2 25% utilization.

[WARNING GUI-0076] QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
Elapsed time: 0:01.85[h]:min:sec. CPU time: user 1.64 sys 0.16 (97%). Peak memory: 170496KB.
cp results/sky130hd/_xls_adder/base/5_route.sdc results/sky130hd/_xls_adder/base/6_final.sdc
(/usr/bin/time -f 'Elapsed time: %E[h]:min:sec. CPU time: user %U sys %S (%P). Peak memory: %MKB.' s
-rd in deferresults/sky130hd/_xls_adder/base/6_final.def \
-rd in_files=../platforms/sky130hd/gds/sky130_fd_sc_hd.gds " \
-rd config_files=../platforms/sky130hd/fill.json \
-rd seal_files="" \
-rd out_file=results/sky130hd/_xls_adder/base/6_1_merged.gds \
-rd tech_file=../objects/sky130hd/_xls_adder/base/klayout.lyt \
-rd layer_map= \
-r /util/def2stream.py) 2>&1 | tee ./logs/sky130hd/_xls_adder/base/6_1_merge.log
[INFO] Reporting cells prior to loading DEF ...
[INFO] Reading DEF ...
[INFO] Clearing cells...
[INFO] Merging GDS/OAS files...
./platforms/sky130hd/gds/sky130_fd_sc_hd.gds
[INFO] Copying toplevel cell 'xls_adder'
INFO: Reading config file: ./platforms/sky130hd/fill.json
[INFO] Checking for missing cell from GDS/OAS...
[INFO] All LEF cells have matching GDS/OAS cells
[INFO] Checking for orphan cell in the final layout...
[INFO] No orphan cells
[INFO] Writing out GDS/OAS 'results/sky130hd/_xls_adder/base/6_1_merged.gds'
Elapsed time: 0:01.51[h]:min:sec. CPU time: user 1.17 sys 0.21 (91%). Peak memory: 383336KB.
ln -sf 6_1_merged.gds results/sky130hd/_xls_adder/base/6_final.gds
make elapsed
make[1]: Entering directory '/OpenROAD-flow-scripts/flow'
[INFO][FLOW] Using platform directory ./platforms/sky130hd
Log
Elapsed seconds
1.1_yosys 1
4.1_cts 2
5.2_TritonRoute 8
6.1_merge 1
6_report 1
make[1]: Leaving directory '/OpenROAD-flow-scripts/flow'
08/21/2023 04:43:27 AM - pipeline_generator - perform_openroad_analysis -thread MainThread -line 175
Design Area um^2: 4090.0
```

Figure 120: Adder Simulation

```
=====
finish report_design_area
-----
Design area 2295 u^2 25% utilization.

[WARNING GUI-0076] QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runt
Elapsed time: 0:01.56[h]:min:sec. CPU time: user 1.46 sys 0.09 (99%). Peak memory: 1
cp results/sky130hd/_xls_counter/base/5_route.sdc results/sky130hd/_xls_counter/
(/usr/bin/time -f 'Elapsed time: %E[h]:min:sec. CPU time: user %U sys %S (%P). Peak
-rd in deferresults/sky130hd/_xls_counter/base/6_final.def \
-rd in_files=../platforms/sky130hd/gds/sky130_fd_sc_hd.gds " \
-rd config_files=../platforms/sky130hd/fill.json \
-rd seal_files="" \
-rd out_file=results/sky130hd/_xls_counter/base/6_1_merged.gds \
-rd tech_file=../objects/sky130hd/_xls_counter/base/klayout.lyt \
-rd layer_map= \
-r /util/def2stream.py) 2>&1 | tee ./logs/sky130hd/_xls_counter/base/6_1
[INFO] Reporting cells prior to loading DEF ...
[INFO] Reading DEF ...
[INFO] Clearing cells...
[INFO] Merging GDS/OAS files...
./platforms/sky130hd/gds/sky130_fd_sc_hd.gds
[INFO] Copying toplevel cell 'xls_counter'
INFO: Reading config file: ./platforms/sky130hd/fill.json
[INFO] Checking for missing cell from GDS/OAS...
[INFO] All LEF cells have matching GDS/OAS cells
[INFO] Checking for orphan cell in the final layout...
[INFO] No orphan cells
[INFO] Writing out GDS/OAS 'results/sky130hd/_xls_counter/base/6_1_merged.gds'
Elapsed time: 0:01.27[h]:min:sec. CPU time: user 1.17 sys 0.08 (99%). Peak memory: 3
ln -sf 6_1_merged.gds results/sky130hd/_xls_counter/base/6_final.gds
make elapsed
make[1]: Entering directory '/OpenROAD-flow-scripts/flow'
[INFO][FLOW] Using platform directory ./platforms/sky130hd
Log
Elapsed seconds
1.1_yosys 1
4.1_cts 2
5.2_TritonRoute 6
6.1_merge 1
6_report 1
make[1]: Leaving directory '/OpenROAD-flow-scripts/flow'
08/21/2023 04:48:26 AM - pipeline_generator - perform_openroad_analysis -thread Main
Design Area um^2: 2295.0
```

Figure 121: Counter Simulation

In the next step, we generate different designs using the grammar that we have as shown in Figure 119. The designs are generated in conjunction with MCTS algorithm where the reward values come from Timeloop and Accelergy analytical models. The input to Timeloop/Accelergy are the problem definition (e.g., convolutional neural network definition), a mapping strategy which applies optimization to generate results and finally the architecture design from graph grammar improved over multiple iterations of MCTS. Figure 122 and Figure 123 shows 2 such generated

graphs for different depths of graph grammar and which are eventually converted to a Timeloop input format for evaluation.

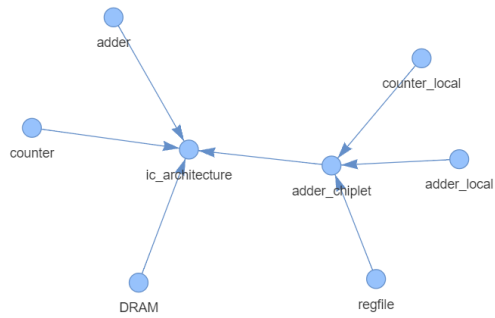


Figure 122: Design graph for depth 5

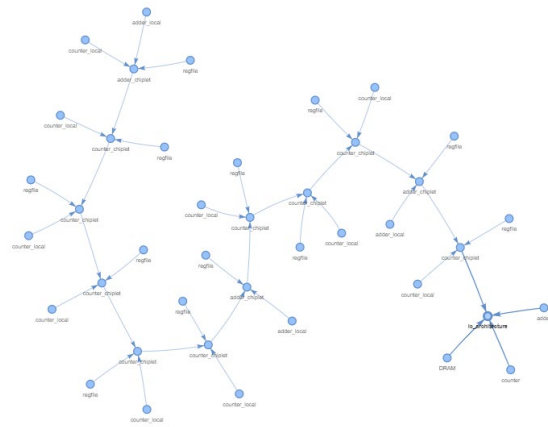


Figure 123: Design graph for depth 20

5.0 CONCLUSIONS

In the first phase of the project, we developed several novel innovative capabilities that demonstrate the benefit of AI-assisted design process as applied to complex CPS. We successfully applied the AI-assisted design process to multiple use-cases including the ones supplied by the TA3 teams in both the hackathon and the phase-end design challenges. We successfully validated our hypothesis that the AI-based methods can significantly accelerate the design process for complex CPS and can achieve this with a very small design corpus. In our experiments, we have been able to demonstrate a factor of 10x speed-up in design process of unmanned underwater vehicle and +50% improvement in design performance over the designs for unmanned aerial vehicles and urban air mobility vehicles.

In the second phase of the project, we were able to improve the technologies and integrate them into the Systemic Generative Engineering Framework (SyGEF). These technologies were demonstrated on the problems that were not only provided to us by the TA3 teams but also other design challenges such as electronic design automation. We successfully demonstrated an integrated design environment in which AI technologies can be used to significantly improve the design process of complex CPS and generated designs that are considerably better than the baseline designs and those generated by expert engineers. Our experiments demonstrate that we accelerated the UUV design process by a factor of over 500x and achieved +400% improvement in design performance over the baseline designs and +250% improvement in design performance over those created by expert designers in much shorter turnarounds (over 15x faster).

As part of the program, we successfully realized the following milestones identified in Table 23.

Table 23: Milestones achieved in Phase 1 and Phase 2 of the program

Identifier	Description
M1	Demonstrate the ability for the knowledge framework to extract design knowledge, requirements, and constraints from design corpus and seed designs and develop the appropriate representation for these in the knowledge graph.
M2	Demonstrate capability to extract logical and mathematical domain and cross-domain constraints from the information stored in the knowledge graph.
M3	Demonstrate the ability to automatically construct the design space for two domains with ability to mine, at minimum, designs with total of 50 components with 10 options per component.
M4	Demonstrate ability to generate and explore, at minimum, 5000 new designs in two domains.
M5	Demonstrate the ability to perform multi-disciplinary optimization using traditional MDAO techniques.
M6	Demonstrate design composition mechanism with ability to autocomplete partial designs to minimum of 50% completeness and 80% accuracy for connections; 40% completeness with 80% accuracy for interface components; and 20% completeness with 80% accuracy for functional components.
M7	Demonstrate cross-fidelity constraints extractor
M8	Demonstrate automatic construction of design space with four+ domains with ability to mine, at minimum, designs with total of 500 components with 10 options per component.
M9	Demonstrate ability to generate and explore, at minimum, 500,000 new design points in three domains.
M10	Demonstrate advanced DSE mechanism.
M11	Demonstrate advanced and automated design evaluation mechanism.
M12	Demonstrate advanced and fully automated design composition mechanism with ability to autocomplete partial designs to minimum of - 90% completeness with 90% accuracy for connections; 70% completeness with 90% accuracy for interface components; and 50% completeness with 90% accuracy for functional components.

6.0 REFERENCES

- 1 DARPA (n.d.). *Symbiotic Design for Cyber Physical Systems*. DAPRA - Symbiotic Design for Cyber Physical Systems. Retrieved September 1, 2023, from <https://www.darpa.mil/program/symbiotic-design-for-cyber-physical-systems>
- 2 Sztipanovits, J., Bapty, T., Neema, S., Howard, L., & Jackson, E. (2014). OpenMETA: A model-and component-based design tool chain for cyber-physical systems. In *From Programs to Systems. The Systems perspective in Computing: ETAPS Workshop, FPS 2014, in Honor of Joseph Sifakis, Grenoble, France, April 6, 2014. Proceedings* (pp. 235-248). Springer Berlin Heidelberg.
- 3 Amato, Giuseppe, et al. "Large-scale image retrieval with elasticsearch." *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 2018.
- 4 Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. "Roberta: A robustly optimized bert pretraining approach." *arXiv preprint arXiv:1907.11692*.
- 5 Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. "Bert: Pre-training of deep bidirectional transformers for language understanding." *arXiv preprint arXiv:1810.04805*.
- 6 Bordes, Antoine, et al. "Translating embeddings for modeling multi-relational data." *Advances in neural information processing systems* 26 (2013).
- 7 Wang, Zhen, et al. "Knowledge graph embedding by translating on hyperplanes." *Proceedings of the AAAI conference on artificial intelligence*. Vol. 28. No. 1. 2014.
- 8 Lin, Yankai, et al. "Modeling relation paths for representation learning of knowledge bases." *arXiv preprint arXiv:1506.00379*(2015).
- 9 Ji, Guoliang, et al. "Knowledge graph embedding via dynamic mapping matrix." *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*. 2015.
- 10 Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980*(2014).
- 11 Li, Yaguang, et al. "A customized graph neural network model for guiding analog IC placement." 2020 IEEE/ACM International Conference on Computer Aided Design (ICCAD). IEEE, 2020.
- 12 P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in Proc. of ICLR, 2017.
- 13 Pahl, G. and Beitz, W., 1988. Engineering design: a systematic approach. *Nasa Sti/recon Technical Report A*, 89, p.47350.
- 14 Fenves, Steven J., Sebti Foufou, Conrad Bock, Roger Bouillon, and Ram D. Sriram. "CPM 2: A revised core product model for representing design information." (2005). <https://doi.org/10.6028/NIST.IR.7185>

- 15 Fenves, S. J., Foufou, S., Bock, C., and Sriram, R. D. (February 28, 2008). "CPM2: A Core Model for Product Data." *ASME. J. Comput. Inf. Sci. Eng.* March 2008; 8(1): 014501. <https://doi.org/10.1115/1.2830842>
- 16 Biswas, Arpan, Steve J. Fenves, Vadim Shapiro, and Ram Sriram. "Representation of heterogeneous material properties in the Core Product Model." *Engineering with Computers* 24 (2008): 43-58.
- 17 Garau, E.M. and Vázquez, R., 2018. Algorithms for the implementation of adaptive isogeometric methods using hierarchical B-splines. *Applied Numerical Mathematics*, 123, pp.58-87.
- 18 Noël, L., Schmidt, M., Messe, C., Evans, J.A. and Maute, K., 2020. Adaptive level set topology optimization using hierarchical B-splines. *Structural and Multidisciplinary Optimization*, 62, pp.1669-1699.
- 19 Yang, A., Wang, S., Luo, N., Xiong, T. and Xie, X., 2022. A space-preserving data structure for isogeometric topology optimization in B-splines space. *Structural and Multidisciplinary Optimization*, 65(10), p.281.
- 20 Valentin, J., Hübner, D., Stingl, M. and Pflüger, D., 2020. Gradient-Based Two-Scale Topology Optimization With B-Splines on Sparse Grids. *SIAM Journal on Scientific Computing*, 42(4), pp.B1092-B1114.
- 21 Sederberg, T. W., Zheng, J., Bakenov, A., & Nasri, A. (2003). T-splines and T-NURCCs. *ACM transactions on graphics (TOG)*, 22(3), 477-484.
- 22 Scott, M. A., Borden, M. J., Verhoosel, C. V., Sederberg, T. W., & Hughes, T. J. (2011). Isogeometric finite element data structures based on Bézier extraction of T-splines. *International Journal for Numerical Methods in Engineering*, 88(2), 126-156.
- 23 Wassermann, B., Korshunova, N., Kollmannsberger, S., Rank, E., & Elber, G. (2020). Finite cell method for functionally graded materials based on V-models and homogenized microstructures. *Advanced Modeling and Simulation in Engineering Sciences*, 7(1), 1-33.
- 24 Forrester, Alexander, Andras Sobester, and Andy Keane. *Engineering design via surrogate modelling: a practical guide*. John Wiley & Sons, 2008.
- 25 Forrester, Alexander IJ, András Sóbester, and Andy J. Keane. "Multi-fidelity optimization via surrogate modelling." *Proceedings of the royal society a: mathematical, physical and engineering sciences* 463.2088 (2007): 3251-3269.
- 26 Keane, Andy, and Prasanth Nair. *Computational approaches for aerospace design: the pursuit of excellence*. John Wiley & Sons, 2005.
- 27 Berkooz, Gal, Philip Holmes, and John L. Lumley. "The proper orthogonal decomposition in the analysis of turbulent flows." *Annual review of fluid mechanics* 25.1 (1993): 539-575.
- 28 Peherstorfer, Benjamin, and Karen Willcox. "Data-driven operator inference for noninvasive projection-based model reduction." *Computer Methods in Applied Mechanics and Engineering* 306 (2016): 196-215.

- 29 Bui-Thanh, T., Murali Damodaran, and Karen Willcox. "Proper orthogonal decomposition extensions for parametric applications in compressible aerodynamics." *21st AIAA applied aerodynamics conference*. 2003.
- 30 Audouze, Christophe, Florian De Vuyst, and Prasanth B. Nair. "Nonintrusive reduced-order modeling of parametrized time-dependent partial differential equations." *Numerical Methods for Partial Differential Equations* 29.5 (2013): 1587-1628.
- 31 Butnaru, Daniel, et al. "Fast insight into high-dimensional parametrized simulation data." *2012 11th International Conference on Machine Learning and Applications*. Vol. 2. IEEE, 2012.
- 32 Lee, Kyunghoon, et al. "Reduced-order modeling of a high-fidelity propulsion system simulation." *AIAA journal* 49.8 (2011): 1665-1682.
- 33 Renganathan, S. Ashwin, Romit Maulik, and Vishwas Rao. "Machine learning for non-intrusive model order reduction of the parametric inviscid transonic flow past an airfoil." *Physics of Fluids* 32.4 (2020).
- 34 Salmoiraghi, Filippo, et al. "Free-form deformation, mesh morphing and reduced-order methods: enablers for efficient aerodynamic shape optimisation." *International Journal of Computational Fluid Dynamics* 32.4-5 (2018): 233-247.
- 35 Demo, Nicola, Marco Tezzele, and Gianluigi Rozza. "A non-intrusive approach for the reconstruction of POD modal coefficients through active subspaces." *Comptes Rendus Mécanique* 347.11 (2019): 873-881.
- 36 Demo, Nicola, et al. "Hull shape design optimization with parameter space and model reductions, and self-learning mesh morphing." *Journal of Marine Science and Engineering* 9.2 (2021): 185.
- 37 Tezzele, Marco, et al. "PyGeM: Python geometrical morphing." *Software impacts* 7 (2021): 100047.
- 38 Virtanen, Pauli, et al. "SciPy 1.0: fundamental algorithms for scientific computing in Python." *Nature methods* 17.3 (2020): 261-272.
- 39 Geuzaine, Christophe, and Jean-François Remacle. "Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities." *International journal for numerical methods in engineering* 79.11 (2009): 1309-1331.
- 40 Dhondt, Guido. "Calculix crunchix user's manual version 2.12." *Munich, Germany, accessed Sept 21* (2017): 2017.
- 41 Jasak, Hrvoje, Aleksandar Jemcov, and Zeljko Tukovic. "OpenFOAM: A C++ library for complex physics simulations." *International workshop on coupled methods in numerical dynamics*. Vol. 1000. 2007.
- 42 Kulfan, Brenda M. "Recent extensions and applications of the 'CST' universal parametric geometry representation method." *The Aeronautical Journal* 114.1153 (2010): 157-176.
- 43 Pontryagin, Lev Semenovich. *Mathematical theory of optimal processes*. Routledge, 2018.

- 44 Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K., "Neural ordinary differential equations," *Advances in neural information processing systems*, Vol. 31, 2018.
- 45 Thuerey, N., Holl, P., Mueller, M., Schnell, P., Trost, F., and Um, K., "Physics-based deep learning," *arXiv preprint arXiv:2109.05237*, 2021.
- 46 Paszke, Adam, et al. "Pytorch: An imperative style, high-performance deep learning library." *Advances in neural information processing systems* 32 (2019).
- 47 Ricky Chen et al. torchdiffeq: Differentiable ode solvers with full gpu support and o(1)-memory backpropagation.
- 48 Xing, Wei, et al. "Shared-Gaussian process: Learning interpretable shared hidden structure across data spaces for design space analysis and exploration." *Journal of Mechanical Design* 142.8 (2020).
- 49 Modelica Documentation. <http://doc.modelica.org>
- 50 Gibert Martínez, I., Afonso, F., Rodrigues, S. and Lau, F., 2021. A sequential approach for aerodynamic shape optimization with topology optimization of airfoils. *Mathematical and Computational Applications*, 26(2), p.34.
- 51 Van Dijk, N.P., Maute, K., Langelaar, M. and Van Keulen, F., 2013. Level-set methods for structural topology optimization: a review. *Structural and Multidisciplinary Optimization*, 48(3), pp.437-472.
- 52 Kumar, P., 2023. TOPress: a MATLAB implementation for topology optimization of structures subjected to design-dependent pressure loads. *Structural and Multidisciplinary Optimization*, 66(4), p.97.
- 53 Emmendoerfer Jr, H., Silva, E.C.N. and Fancello, E.A., 2019. Stress-constrained level set topology optimization for design-dependent pressure load problems. *Computer Methods in Applied Mechanics and Engineering*, 344, pp.569-601.
- 54 Kremar, J.M., 2020. Topology Optimization of Irregular Shaped Pressure Vessels Using a Level-set Method (Thesis, Clemson University).
- 55 Picelli, R., van Dijk, R., Vicente, W.M., Pavanello, R., Langelaar, M. and van Keulen, F., 2017. Topology optimization for submerged buoyant structures. *Engineering optimization*, 49(1), pp.1-21.
- 56 Alexandersen, J., 2023. A detailed introduction to density-based topology optimization of fluid flow problems with implementation in MATLAB. *Structural and Multidisciplinary Optimization*, 66(1), p.12.
- 57 J. Schulman, et al. "Proximal Policy Optimization Algorithms." *arXiv:1707.06347*, 2017.
- 58 SwRI use case, <https://git.isis.vanderbilt.edu/SwRI/flight-dynamics-model>.
- 59 Jones, Donald R., Matthias Schonlau, and William J. Welch. "Efficient global optimization of expensive black-box functions." *Journal of Global optimization* 13 (1998): 455-492.

- 60 Balandat, Maximilian, et al. "BoTorch: A framework for efficient Monte-Carlo Bayesian optimization." *Advances in neural information processing systems* 33 (2020): 21524-21538.
- 61 Cazals, Frédéric, and Claude Puech. "Bucket-like space partitioning data structures with applications to ray-tracing." *Proceedings of the thirteenth annual symposium on Computational geometry*. 1997.
- 62 Siemens, P. L. M. "Simcenter HEEDS." *Academic Research* (2017).
- 63 Polak, Michal, and Robert Trivers. "The science of symmetry in biology." *Trends in ecology & evolution* 9.4 (1994): 122-124.
- 64 Crawford, J., Ginsberg, M., Luks, E., & Roy, A. (1996). Symmetry-breaking predicates for search problems. *KR*, 96(1996), 148-159.
- 65 R. Garnett. Bayesian optimization. Cambridge University Press, 2023.
- 66 Laughlin, B., Briceno, S. I., and Mavris, D. N., "A virtual experimentation platform enabling the design, testing, and verification of an unmanned aerial vehicle through cyber-physical, component-based design," 14th AIAA Aviation Technology, Integration, and Operations Conference, 2014, p. 2720.
- 67 Moitra, A., Cuddihy, P., Siu, K., Meng, B., Interrante, J., Archer, D., Mertens, E., Quick, K., Robert, V., and Russell, D., "A Semantic Reference Model for Capturing System Development and Evaluation," 2022 IEEE 16th International Conference on Semantic Computing (ICSC), IEEE, 2022, pp. 173–174.
- 68 Denney, E., Pai, G., and Pohl, J., "AdvoCATE: An assurance case automation toolset," Computer Safety, Reliability, and Security: SAFECOMP 2012 Workshops: Sassur, ASCoMS, DESEC4LCCI, ERCIM/EWICS, IWDE, Magdeburg, Germany, September 25-28, 2012. Proceedings 31, Springer, 2012, pp. 8–21.
- 69 Ramakrishna, S., Jin, H., Dubey, A., and Ramamurthy, A., "Automating Pattern Selection for Assurance Case Development for Cyber-Physical Systems," Computer Safety, Reliability, and Security: 41st International Conference, SAFECOMP 2022, Munich, Germany, September 6–9, 2022, Proceedings, Springer, 2022, pp. 82–96.
- 70 DARPA, "Automated Rapid Certification of Software (ARCOS)," URL <https://www.darpa.mil/program/automated-rapid-certification-of-software>.
- 71 Kelly, T., and Weaver, R., "The goal structuring notation—a safety argument notation," Proceedings of the dependable systems and networks 2004 workshop on assurance cases, Vol. 6, Citeseer, 2004.
- 72 Kane, M. T., "An argument-based approach to validity." *Psychological bulletin*, Vol. 112, No. 3, 1992, p. 527.
- 73 Ramakrishna, S., Hartsell, C., Dubey, A., Pal, P., and Karsai, G., "A methodology for automating assurance case generation," arXiv preprint arXiv:2003.05388, 2020.
- 74 OpenAI, "GPT-4 Technical Report," 2023.

- 75 Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al., "Palm: Scaling language modeling with pathways," arXiv preprint arXiv:2204.02311, 2022.
- 76 Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G., "LLaMA: Open and Efficient Foundation Language Models," 2023.
- 77 Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J., "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," *Journal of Machine Learning Research*, Vol. 21, No. 140, 2020, pp. 1–67. URL <http://jmlr.org/papers/v21/20-074.html>.
- 78 Loshchilov, I., and Hutter, F., "Decoupled Weight Decay Regularization," 2019.
- 79 Lin, C.-Y., "ROUGE: A Package for Automatic Evaluation of Summaries," *Text Summarization Branches Out*, Association for Computational Linguistics, Barcelona, Spain, 2004, pp. 74–81. URL <https://aclanthology.org/W04-1013>.
- 80 Neo4j, "Neo4j," URL <https://neo4j.com>.
- 81 Boniol, F., and Wiels, V., "The landing gear system case study," *ABZ 2014: The Landing Gear Case Study: Case Study Track*, held at the 4th International Conference on Abstract State Machines, Alloy, B, TLA, VDM, and Z, Toulouse, France, June 2-6, 2014. *Proceedings 4*, Springer, 2014, pp. 1–18.
- 82 Boniol, F., Wiels, V., Aït-Ameur, Y., and Schewe, K.-D., "The landing gear case study: challenges and experiments," 2017.
- 83 Strassel, S., and Mitchell, A., "Multilingual Resources for Entity Extraction," *Proceedings of the ACL 2003 Workshop on Multilingual and Mixed-language Named Entity Recognition*, Association for Computational Linguistics, Sapporo, Japan, 2003, pp. 49–56. doi:10.3115/1119384.1119391, URL <https://aclanthology.org/W03-1507>.
- 84 Yao, Z. F., Wang, J. Y., & Pei, J. (2021). High-performance polymer field-effect transistors: from the perspective of multi-level microstructures. *Chemical Science*, 12(4), 1193-1205.
- 85 Schauser, N. S., Kliegle, G. A., Cooke, P., Segalman, R. A., & Seshadri, R. (2021). Database creation, visualization, and statistical learning for polymer Li⁺-electrolyte design. *Chemistry of Materials*, 33(13), 4863-4876.
- 86 Chen, Shiyi, and Gary D. Doolen. "Lattice Boltzmann method for fluid flows." *Annual review of fluid mechanics* 30.1 (1998): 329-364.
- 87 Bhatnagar, Prabhu Lal, Eugene P. Gross, and Max Krook. "A model for collision processes in gases. I. Small amplitude processes in charged and neutral one-component systems." *Physical review* 94.3 (1954): 511.
- 88 Lallemand, Pierre, and Li-Shi Luo. "Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability." *Physical review E* 61.6 (2000): 6546.

- 89 Talischi, Cameron, et al. "PolyMesher: a general-purpose mesh generator for polygonal elements written in Matlab." *Structural and Multidisciplinary Optimization* 45 (2012): 309-328.
- 90 Perumal, D. Arumuga, and Anoop K. Dass. "A Review on the development of lattice Boltzmann computation of macro fluid flows and heat transfer." *Alexandria Engineering Journal* 54.4 (2015): 955-971.
- 91 Delgado-Gutiérrez, Arturo, et al. "An efficient implementation of the graphics processing unit-accelerated single-step and simplified lattice Boltzmann method for irregular fluid domains." *Physics of Fluids* 34.12 (2022).
- 92 Schubiger, Alain, Sarah Barber, and Henrik Nordborg. "Evaluation of the lattice Boltzmann method for wind modelling in complex terrain." *Wind Energy Science* 5.4 (2020): 1507-1519.
- 93 Yang, X. H., et al. "Comparison of LBM and FVM for simulation of solid-liquid phase change problem with natural convection." *IOP conference series: earth and environmental science*. Vol. 701. No. 1. IOP Publishing, 2021.
- 94 Foresti G.L. [2001] Visual Inspection of Sea Bottom Structures by an Autonomous Underwater Vehicle, IEEE Transactions on Systems, Man and Cybernetics - Part B: cybernetics, Vol. 31, No. 5, October 2001.
- 95 Dhanak M.R, An P. E., Holappa K. [2001] An AUV Survey in the Littoral Zone: Small Scale Subsurface Variability Accompanying Synoptic Observations of Surface Currents, IEEE Journal of Oceanic Engineering, Vol. 26, No. 4, October 2001.
- 96 Thor I Fossen. Guidance and control of ocean vehicles, volume 199. Wiley New York, 1994.
- 97 Fossen, Thor I. Handbook of marine craft hydrodynamics and motion control. John Wiley & Sons, 2021.
- 98 ElectroChem, Battery Cell Product Information, Retrieved September 1, 2023 from URL <https://electrochemsolutions.com/resources/product-information/default.aspx>

APPENDIX A – PUBLICATIONS AND PRESENTATIONS

Completed Events

- “*Deep Generative Model for Efficient 3D Airfoil Parameterization and Generation*”, Chen, W., Ramamurthy, A., AIAA SciTech, January 2021.
- “*Path Auxiliary Proposal for MCMC in Discrete Space*”, Sun, H., Dai, H., Xia, W., Ramamurthy, A., International Conference on Learning Representations, 2022. [Spotlight presentation]
- “*Combinatorial and Parametric Gradient-Free Optimization for Cyber-Physical System Design*”, Zheng, H., Betz, J., Ramamurthy, A., Jin, H., Mangharam, R., DESTION @ International Conference on Cyber Physical Systems, 2022.
- “*Reduced Order Model for Standard CAE Design Workflows*”, Valenzuela del Rio, J., Ramamurthy, A., AIAA SciTech, January 2023.
- “*Top-Down Synthesis for Library Learning*”, Bowers, M., Olausson, T., Wong, C., Grand, G., Ellis, K., Tenenbaum, J., Solar-Lezama, A., ACM SIGPLAN Principles of Programming Languages, January 2023.
- “*Game-theoretic Objective Space Planning*”, Hongrui Zheng, Zhijun Zhuang, Johannes Betz, Rahul Mangharam, IEEE RSS, July 2023.
- “*Local_INN: Implicit Map Representation and Localization with Invertible Neural Networks*”, Zirui Zang, Hongrui Zheng, Johannes Betz, Rahul Mangharam, ICRA 2023, May 2023.
- “*MEGA-Dagger: Imitation Learning with Multiple Imperfect Experts.*”, Sun, Xiatao, Shuo Yang, and Rahul Mangharam, IROS 2023, October 2023.
- “*Accelerating Online Reinforcement Learning via Supervisory Safety Systems*”, Benjamin Evans, Johannes Betz, Hongrui Zheng, Herman A. Engelbrecht, Rahul Mangharam, Hendrik W. Jordaen, IROS 2023, October 2023.
- “*Physics Constrained Motion Prediction with Uncertainty Quantification*”, R Tumu, L Lindemann, T Nghiem, R Mangharam, IEEE Intelligent Vehicle, 2023.
- “*Towards Explainability in Modular Autonomous Vehicle Software*”, Hongrui Zheng, Zirui Zang, Shuo Yang, Rahul Mangharam, IEEE Intelligent Vehicle, 2023.

Future Events

- “*Learning to Verify and Assure Cyber-Physical Systems*”, Jin, H., Zhang, T., Ramamurthy, A., Hamza, A., Malinoski, M., AIAA SciTech, January 2024.
- “*Differentiable Partial Differential Equation Solvers with Ordinary Differential Equation Neural Networks*”, Valenzuela del Rio, J., Ramamurthy, A., AIAA SciTech, January 2024.

LIST OF SYMBOLS, ABBREVIATIONS, AND ACRONYMS

1D	1-Dimensional
2D	2-Dimensional
3D	3-Dimensional
AC	Assurance Case
AI	Artificial Intelligence
AIAA	American Institute of Aeronautics and Astronautics
APL-UW	The University of Washington Applied Physics Laboratory
ASICs	Application-specific Integrated Circuits
ATO	Authority to Operate
AUV	Autonomous Underwater Vehicle
BERT	Bidirectional Encoder Representations from Transformers
BFS	Breadth First Search
BGK	Bhatnagar-Gross-Krook
BMW	Bayerische Motoren Werke
BO	Bayesian Optimization
BPE	Byte-level Byte-Pair-Encoding
C4	Colossal Clean Crawled Corpus
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAE	Claim, Argument, Evidence
CFD	Computational Fluid Dynamics
CFR	Code of Federal Regulations
CMAES	Covariance Matrix Adaption Evolutionary Strategy
CNN	Convolution Neural Network
CoG	Center of Gravity
CONF	Nine heavy atoms (CONF) out of the GDB-17 chemical universe
COTS	Commercial Off The Shelf
CPS	Cyber Physical Systems
CSG	Constructive Solid Geometry
CST	Class-shape Transformation
CTD	Conductivity, Temperature, and Depth
CyPhyML	Cyber-physical models
DARPA	Defense Advanced Research Projects Agency
DBPedia	DBpedia (from "DB" for "database") is a project aiming to extract structured content from the information created in the Wikipedia project.
DE	Differential Equations

DES	Discrete-Event Simulation
DFT	Density-Functional Theory
DMM	Design Master Model
DNN	Deep Neural Network
DoE	Design of Experiments
DOF	Degrees of Freedom
DSE	Design Space Exploration
DSL	Domain Specific Languages
DSLX	DSLX is a domain specific, dataflow-oriented functional language used to build hardware that can also run effectively as host software.
DSM	Design Structure Matrix
DSM	Domain Specific Model
DT	Digital Twin
ECAD	Electrical Computer-Aided Design
EDA	Electronic Design Automation
EI (MIO)	Multi-Echelon Inventory Optimization
ESC	Electronic Speed Controller
FAA	Federal Aviation Administration
FE	Finite Elements
FEA	Finite Element Analysis
FEM	Finite Element Method
FET	Field-Effect Transistors
FFD	Free Form Deformation
FPGA	Field-programmable Gate Arrays
FV	Finite Volumes
FVM	Finite Volume Method
GA	Genetic Algorithm
GDB	Generated Database
GDII	Graphic Design System II
GDS	Graphic Data System
GINE	Graph Isomorphism Network Engine
Gmsh meshing tools	finite-element mesh generator for geometry description, meshing, solving and post-processing.
GPT	Generative Pre-trained Transformers
GPU	Graphic Processing Unit
GraphML	an XML-based file format for graphs
GSN	Goal Structuring Notation
HEV	Hybrid Electric Vehicle
HEX	Hexacopter

HLS	High-level Synthesis
HPR	Heading Pitch Roll
IC	Integrated Circuit
ICE	Range extender in hybrid electric cars
ICE	Internal Combustion Engine
IDF	Individual Disciplinary Feasible
IGA	Iso Geometric Analysis
IP	Intellectual Property
ISR	Intelligence, Surveillance, and Reconnaissance
JAX	a Python library that provides high-performance computing in machine learning research
JSON	JavaScript Object Notation
KG	Knowledge Graph
KPI	Key Performance Indicator
K _t	Torque constant
KV	Constant Velocity
LBM	Lattice Boltzmann Methods
LLM	Large Language Models
LoRA	Low-Rank Adaptation
M&S	Modeling and Simulation
MCMC	Markov Chain Monte Carlo
MCTS	Monte Carlo Tree Search
MDAO	Multidisciplinary Design Analysis and Optimization
MDF	Multi-Disciplinary Feasible
MDP	Markov Decision Process
ML	Machine Learning
MLN	Markov Logic Networks
MoI	Moment of Inertia
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
MRT	Multiple Relaxation Time
MSE	Mean Squared Error
NED	North, East, Down
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
OBS	Ocean Bottom Seismometer
ODE	Ordinary Differential Equations
OEM	Original Equipment Manufacturer
OFET	Organic Field-Effect Transistor

OPLS-AA	Optimized Potentials for Liquid Simulations All Atom
PDE	Partial Differential Equations
PDF	Portable Document Format
PE	Processing Element
PI	Principal Investigator
PID	Proportional, Integral, and Derivative
PnR	Place and Route
POD	Proper Orthogonal Decomposition
PODI	Proper Orthogonal Decomposition with Interpolation
PPA	Power, Performance and Area
PPO	Proximal Policy Optimization
PSO	Particle Swarm Optimization
PV	Pressure Vessel
QA	Question and Answering domain, a sub-domain of NLP.
QM9	QM9 is a database of quantum chemical properties for a comprehensive chemical space of small organic molecules.
R&D	Research and Development
RANS	Reynolds Averaged Modeling
RBF	Radial Basis Function
RGL	Register Transfer Level
RL	Reinforcement Learning
RMSE	Root Mean Squared Errors
RoBERTa	Robustly Optimized BERT Pretraining Approach
ROM	Reduced Order Model
ROS	Robot Operating System
RPM	Revolutions per Minute
RTL	Register Transfer Level
SA	Simulated Annealing
SCBO	Symmetry Constrained Bayesian Optimization
SDCPS	Systemic Design for Cyber Physical Systems
SGD	Stochastic Gradient Descent
SGEF / SyGEF	Systemic Design Engineering Framework
Shared-GP	a generative model for the design problem where the simulator inputs, designs and QoIs are jointly embedded into a structured latent space.
Simcenter HEEDS	a powerful design space exploration and optimization software interfacing with CAD and CAE tools to drive product innovation.
SHERPA	Simultaneous Hybrid Exploration that is Robust, Progressive, and Adaptive
SMILE	simplified molecular-input line-entry system.

SOTA	State of The Art
SPCMC	Seconds Elapsed Per Core Million Cells
SPICE	Simulation Program with Integrated Circuit Emphasis
STL	Stereolithography file format
STR	System & Technology Research LLC
SwRI	Southwest Research Institute
SysML	System Models
T5	Transfer Learning with a Unified Text-to-Text Transformer
TA	Technical Area
TO	Topology Optimization
UAM	Urban Air Mobility
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
UQ	Uncertainty Quantification
UUV	Unmanned Underwater Vehicle
VAE	Variational Autoencoder
VLSI	Very Large-Scale Integration
XLS	Accelerated HW Synthesis
XML	eXtensible Markup Language