

**Optimal Partitioning and Coordination Decisions in
Decomposition-based Design Optimization**

by

James T. Allison

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in The University of Michigan
2008

Doctoral Committee:

Professor Panos Y. Papalambros, Chair
Professor Noboru Kikuchi
Professor Romesh Saigal
Associate Research Scientist Michael Kokkolaras
Terrance Wagner, Ford Motor Company

© James T. Allison

All Rights Reserved

2008

to Ellie, Jonathan, Brian, and Michael

Acknowledgments

I have been very fortunate to study under the masterful guidance of my advisor, Panos Papalambros. I am grateful for his remarkable support throughout my challenges and adventures at the University of Michigan; I have been inspired by his example. It was because of the opportunity to work with him that I chose to come to Michigan, and I undoubtedly made the right choice. I want to thank my dissertation committee for the time and effort they volunteered to guide my dissertation work. I want also to recognize my wife, Natalie, who has labored just as ardently as I toward my graduation, and my extended family and friends, who gave of themselves so that I could realize my ambitions.

Much of my work has been collaborative. Michael Kokkolaras has been a valuable mentor and research collaborator throughout my graduate studies. Thanks to my affiliation with the Optimal Design Laboratory at the University of Michigan, I've had fantastic opportunities to work with people from around the world, including Brian Roth, Emanuele Colomba, Guido Karsemakers, Simon Tosserams, David Walsh, and others. I want to thank all of my ODE friends for enriching my experiences, including Ryan Fellini, Erin MacDonald, Jeongwoo Han, Jarod Kelly, Kwang Jae Lee, Kuei-Yuan Chan, Jeremy Michalek, Subroto Gunawan, Marc Zawislak, Eric Rask, Mike Sasena, Bart Frischknecht, and many others. I am grateful to my other friends at the U of M, including Mike Cherry, April Bryan, Natasha Chang, Eduardo Izquierdo, Brian Trease, Ken Pollary, and others.

Many individuals contributed to the electric vehicle design case study. Kwang Jae Lee played an important role in system integration, optimization, and programming. Jeongwoo Han provided the Li-ion battery model, and Jarod Kelley developed the frame design and structural model. Emanuele Colomba helped define chassis design and overall vehicle geometry. Michael Alexander assisted with structural model development. Dushyant Wadivkar, Burit Kitterungsi, and Mikael Nybacka all provided support for the vehicle dynamics model, and Hisashi Heguri generously furnished tire model data.

Many experiences helped prepare me for graduate school. My involvement with the solar car team at the University of Utah not only helped me discover my research interests, but gave me glimpses of the possible. My fellow solar car teammates, including Andy Rahden, Jayme Allred, Tadd Truscott, and Brad Hansen, played a vital role. Eberhard Bamberg, also of the University of Utah, was an important mentor who offered timely direction and

encouraged me to seek the best possible graduate school experience. Dennis Rosier, my high school auto shop teacher, helped ignite my passion for learning, and Corinne Barney, my high school math teacher, inspired me to expand my horizons. My parents offered regular encouragement to live to my potential, and my father provided long hours of math tutoring through my early college years that were key to my academic success.

Finally, I would like to recognize support from the U.S. National Science Foundation, the Automotive Research Center at the University of Michigan, the Rackham Graduate School, and the University of Michigan College of Engineering and Department of Mechanical Engineering.

Table of Contents

Dedication	ii
Acknowledgments	iii
List of Tables	viii
List of Figures	ix
List of Symbols	xii
Abstract	xv
Chapter 1 Decomposition-based Design Optimization	1
1.1 Engineering System Design	2
1.2 Design Optimization	4
1.3 Decomposition-based Design Optimization	8
1.4 Introductory Example	11
1.4.1 Coordination Decisions	13
1.4.2 Partitioning Decisions	15
1.5 Dissertation Overview	16
Chapter 2 Partitioning and Coordination Decisions	18
2.1 Decomposition-based System Design	18
2.1.1 System Consistency	20
2.1.2 System Optimality	20
2.1.3 Distributed Optimization	21
2.2 Decision Framework	22
2.3 System Representation	24
2.3.1 Structural Matrix	24
2.3.2 Design Structure Matrix	25
2.3.3 Other Design Matrices	26
2.3.4 Reduced Adjacency Matrix	27
2.4 Partitioning and Coordination Decision-Making	28
2.4.1 Traditional Decision Techniques	29
2.4.2 Formal Decision Techniques	30
2.5 Summary	34

Chapter 3	Demonstration Examples	35
3.1	Air Flow Sensor Design	35
3.2	Turbine Blade Design	38
3.2.1	Analysis Model	39
3.2.2	System Analysis	43
3.3	Aircraft Family Design	44
3.3.1	Product Families in Aircraft Design	44
3.3.2	Aircraft Performance Analysis	45
3.3.3	Aircraft Family Problem Formulation	46
3.4	Generalized Truss Design	48
3.4.1	Truss Analysis	48
3.4.2	Truss Design Formulation	51
3.5	Electric Water Pump Design	52
3.5.1	Water Pump Design	53
3.5.2	Analysis Overview	55
3.5.3	Thermal Analysis	58
3.5.4	Motor Current Analysis	64
3.5.5	Motor Speed Analysis	65
3.5.6	Torque and Pressure Analysis	65
3.5.7	Optimization Results	68
3.6	Concluding Comments	69
Chapter 4	System Design Optimization Formulations	70
4.1	Single-Level Formulations	70
4.1.1	Multidisciplinary Feasible Formulation	71
4.1.2	Individual Disciplinary Feasible Formulation	72
4.1.3	All-at-Once Formulation	74
4.2	System Analysis for Single-level Formulations	75
4.2.1	Fixed Point Iteration	76
4.2.2	Example: Hidden Optima	78
4.2.3	Coupling Strength in Single-Level Formulations	79
4.3	Multi-Level Formulations	85
4.3.1	Classes of Multi-Level Formulations	86
4.3.2	Analytical Target Cascading	87
4.3.3	Example: Aircraft Family Design	91
4.3.4	Augmented Lagrangian Coordination	94
4.3.5	Example: Air Flow Sensor Design	96
4.4	Concluding Comments	98
Chapter 5	Optimal Partitioning and Coordination: Theoretical Framework	99
5.1	P/C Problem Formulation	99
5.2	P/C Problem Solution	101
5.3	Examples	103
5.4	Water Pump Electrification Example	107
5.5	Concluding Comments	108

Chapter 6	Extension to Larger Systems	110
6.1	Evolutionary Algorithms	110
6.2	Evolutionary Algorithm for Partitioning and Coordination	113
6.2.1	Partition Genotype Representation	114
6.2.2	Sequence Genotype Representation	115
6.2.3	Comparative Examples	117
6.3	Generalized Truss Design Problem	117
6.3.1	System Partitioning	118
6.3.2	Example: Eight-bar Truss	119
6.3.3	EA Results	120
6.4	Concluding Comments	122
Chapter 7	Consistency Constraint Allocation for Augmented Lagrangian Co-ordination	123
7.1	Parallel ALC	123
7.2	Linking Structure Analysis	126
7.2.1	Consistency Constraint Graphs	127
7.2.2	Valid Consistency Constraint Graphs	128
7.2.3	Example Consistency Constraint Graph	132
7.3	Optimal Partitioning and Coordination Decisions for Parallel ALC	132
7.4	Example: Electric Water Pump Design Problem	136
7.5	Concluding Comments	138
Chapter 8	Electric Vehicle Design	140
8.1	Vehicle Description	141
8.2	Powertrain Model	147
8.2.1	Vehicle Model	148
8.2.2	Induction Motor Model	153
8.2.3	Lithium Ion Battery Model	163
8.3	Vehicle Dynamics Model	166
8.3.1	Directional Stability	167
8.3.2	Steering Responsiveness	168
8.3.3	Quarter-Car Model	169
8.4	Structural Model	172
8.5	Mass Distribution and Packaging	173
8.6	Optimal P/C Decision Results	175
8.7	Concluding Comments	180
Chapter 9	Conclusion	181
9.1	Summary	181
9.2	Extension of Simultaneous P/C Decision Making	183
9.3	Contributions	184
9.4	Future Work	184
Bibliography		186

List of Tables

Table

1.1	Experimental results for evaluation sequence variation	13
1.2	Experimental results for evaluation sequence and partition variation	16
2.1	Summary of formal partitioning and coordination decision methods	33
3.1	Turbine blade design parameters	44
3.2	Design variables for the aircraft family design problem	47
3.3	Design constraints for the aircraft family design problem	48
3.4	Analysis functions and design variables for the electric water pump design problem	54
3.5	Electric water pump model parameters	56
3.6	Optimization results for the electric water pump design problem	68
4.1	ALC solution progress for the air-flow sensor problem	98
6.1	Redundancy in $\hat{\mathbf{p}}$ partition representation	114
6.2	Design parameters and optimal geometry for the 8-bar truss problem	120
8.1	EV design variables	144
8.2	EV coupling variables	144
8.3	EV design constraint parameters	147
8.4	Vehicle model parameters	153
8.5	Motor model parameters	163
8.6	Vehicle dynamics model parameters	172
8.7	Mass distribution and packaging model parameters	175

List of Figures

Figure		
1.1	Sample sequential design process: automotive example	3
1.2	Illustration of global and local optima in a nonlinear programming example	5
1.3	Aeroelastic analysis	9
1.4	Process for implementing decomposition-based design optimization	10
1.5	Dissertation hypothesis: Existence of coupling between partitioning and coordination decisions	11
2.1	Input and output relationships for a system of analysis functions	19
2.2	Aspects of partitioning and coordination decisions	23
2.3	Hypergraph for relationships in Eqs. (2.4)	25
2.4	Digraph of functional relationships expressed in the DSM	26
3.1	Vane airflow sensor schematic (after [34])	36
3.2	Simplified representation of a vane airflow sensor	36
3.3	Coupling relationship in airflow sensor analysis	37
3.4	GE J-79 turbojet engine turbine blades [57]	39
3.5	Turbine blade model schematic	40
3.6	Turbine blade coupling and functional relationships	43
3.7	Truss geometry and free-body diagram	49
3.8	Electrically driven centrifugal water pump [39]	54
3.9	Analysis interactions in electric water pump model	55
3.10	Schematic of permanent magnet DC motor	57
3.11	Section view of DC motor armature	58
3.12	Schematic of centrifugal water pump	58
3.13	DC motor thermal resistance model	60
4.1	MDF architecture	71
4.2	IDF architecture	73
4.3	AAO architecture	75
4.4	Two element coupled system	76
4.5	System with multiple fixed points	77
4.6	IDF optimization space visualization	79
4.7	Coupling relationship in airflow sensor analysis	80
4.8	Comparison of MDF and IDF solution time as a function of coupling strength	81
4.9	Comparison of MDF and IDF function evaluations as a function of coupling strength	82
4.10	Turbine blade coupling and functional relationships	83

4.11	Comparison of MDF and IDF solution time as a function of coupling strength	85
4.12	Hierarchical system analysis structure	88
4.13	ATC subproblem as an optimal value function	90
4.14	Influence of β on $RMS(\mathbf{c})$ (system consistency)	94
5.1	Independent (P, C) optimization approach	102
5.2	$P \rightarrow C$ sequential optimization	102
5.3	$C \rightarrow P$ sequential optimization	103
5.4	Simultaneous $(P C)$ optimization	103
5.5	CS and SS_{\max} histograms for first example system	104
5.6	Optimization results for \mathbf{A}_1	104
5.7	CS and SS_{\max} histograms for \mathbf{A}_2	106
5.8	Optimization results for \mathbf{A}_2	106
5.9	Optimal P/C results for pump problem	108
6.1	Typical evolutionary algorithm process	111
6.2	Genotype and phenotype representation partition size distributions	115
6.3	Subproblem sequence distribution	116
6.4	Combined subproblem sequence and partition size distribution	116
6.5	Surjective mapping from genotype space to phenotype space	116
6.6	EA results for first example system	117
6.7	EA results for second example system	117
6.8	EA results for third example system	118
6.9	Geometry and applied loads for the 8-bar truss problem	119
6.10	Non-dominated solutions for 8-bar truss problem	121
7.1	Analysis function digraph for example system	124
7.2	Subproblem graph	125
7.3	Condensed subproblem graph	125
7.4	Stage graph	126
7.5	Graph representation of consistency constraint options for x_1	133
7.6	ALC P/C results for electric water pump problem	136
7.7	Consistency constraint allocation option for point 3	138
8.1	Vehicle systems and interactions in the EV design problem	140
8.2	Top view of EV component layout	142
8.3	Relationships between analysis functions in the EV design problem	143
8.4	Simplified overview of EV powertrain model	147
8.5	Block diagram of dynamic vehicle model	148
8.6	Simplified federal urban drive schedule	149
8.7	2 DOF vehicle pitch model	151
8.8	Slip data for electric vehicle tire	153
8.9	Diagram of an induction motor	154
8.10	Equivalent circuit of an induction motor	155
8.11	Typical IM maximum torque curve	158
8.12	Example IM efficiency map	161
8.13	Example IM power loss map with points visited during SFUDS	162
8.14	Flat-wound lithium-ion battery cell (after [68])	164
8.15	Battery and motor power output during SFUDS cycle	165

8.16	Battery power output and charge and discharge limits during range test . . .	166
8.17	Quarter-car vehicle suspension model	169
8.18	Sample road profile created using gaussian random number generator and digital filters	171
8.19	FEA model of the EV frame	173
8.20	Optimal partitioning and coordination decision results for the EV problem .	176

List of Symbols

\circ	Hadamard product
β	method of multipliers penalty function multiplier parameter
γ	method of multipliers penalty function threshold parameter
π	optimal value function for all ALC subproblems
π_i	optimal value function for ALC subproblem i
ϕ	augmented Lagrangian penalty function
θ_{ij}	consistency constraint vector between subproblems i and j
Θ	consistency constraint matrix
v_i	number of subproblems linked by the i -th external linking variable
ζ_{ij}	data persistence metric
\mathbf{a}	vector of all analysis functions for a system
\mathbf{a}_i	i -th analysis function
$\hat{\mathbf{a}}_i$	analysis function vector that corresponds to $\hat{\mathbf{y}}_i$
\mathbf{A}	reduced adjacency matrix
$\bar{\mathbf{A}}$	subproblem graph adjacency matrix
B_m	m -th Bell number
B_{allow}	maximum allowed subproblem size imbalance
\mathbf{C}	coordination decision problem
\mathbf{c}_i	ATC consistency constraint for subproblem i
$\bar{\mathbf{c}}_{ij}$	ALC external consistency constraint between subproblems i and j
CM	correlation matrix
CS	coordination problem size
\mathcal{C}	set of consistency constraint graphs for all external shared design variables
DSM	design structure matrix
f	design objective function
FDT	functional dependence matrix
\mathbf{g}	inequality design constraint vector
\mathbf{g}_i	inequality design constraint vector for subproblem i
G_c	consistency constraint graph
\mathbf{h}	equality design constraint vector

\mathbf{h}_{aux}	auxiliary equality constraint vector
\mathbf{h}_i	equality design constraint vector for subproblem i
m	number of analysis functions in a system
n	number of design variables in a system
n^s	stage depth
n_{ai}	number of analysis functions in subproblem i
n_{P_i}	number of subproblems that share i -th external shared variable
$n_{x_{li}}$	number of local design variables in subproblem i
$n_{\bar{x}_s}$	number of external shared design variables
$n_{\bar{x}_s ci}$	number of consistency constraints for external shared variables in subproblem i
$n_{\bar{x}_s i}$	number of external shared design variables in subproblem i
$n_{\bar{x}_s m}$	approximate number of external shared design variable consistency constraints
$n_{\bar{y}_f}$	number of external feedback coupling variables
$n_{\bar{y}_f i}$	number of external feedback coupling variables for subproblem i
n_{y_i}	number of internal coupling variables in subproblem i
$n_{\bar{y}_i}$	number of consistency constraints for external coupling variables in subproblem i
$n_{\bar{y}_i i}$	number external input coupling variables for subproblem i
n^z	number of subproblems linked by linking variable z
n_z	number of external linking variables
N	number of subproblems
\mathbf{o}	analysis function sequence vector
\mathbf{o}_s	subproblem sequence vector
$\hat{\mathbf{o}}$	genotype analysis function sequence vector
\mathbf{p}	partition vector
$\hat{\mathbf{p}}$	genotype partition vector
p_{ij}	path from vertex i to vertex j
P	partitioning decision problem
P/C	partitioning and coordination
P_i	subproblem i
\mathbf{RM}	relation matrix
\mathbf{r}_{ij}	ATC responses from subproblem j to subproblem i
\mathbf{S}_{ij}	\mathbf{a}_i selection matrix for \mathbf{y}_{ij}
\mathbf{SM}	structural matrix
SS_i	size of subproblem i
SS_{max}	maximum subproblem size
\bar{SS}_{max}	average maximum subproblem size for each stage

\mathbf{t}_{ij}	ATC targets from subproblem j to subproblem i
\mathbf{v}	linear penalty weights
\mathbf{v}_i	linear penalty weights for subproblem i
\mathbf{w}	quadratic penalty weights
\mathbf{w}_i	quadratic penalty weights for subproblem i
\mathbf{x}	design variable vector
\mathbf{x}^*	optimal design variable vector
\mathbf{x}_i	design variables input to \mathbf{a}_i
\mathbf{x}_{ℓ_i}	design variables local to \mathbf{a}_i
\mathbf{x}_{s_i}	shared design variables associated with \mathbf{a}_i
$\bar{\mathbf{x}}_i$	design variables input to subproblem i
$\bar{\mathbf{x}}_i^{(j)}$	copy of $\bar{\mathbf{x}}_i$ local to subproblem i
$\bar{\mathbf{x}}_{\ell_i}$	design variables local to subproblem i
$\bar{\mathbf{x}}_{s_i}$	external shared design variables associated with subproblem i
$\bar{\mathbf{x}}_s^{ij}$	design variable copies shared between subproblems i and j , local to subproblem i
$\bar{\mathbf{x}}_{s\mathcal{C}_i}$	external shared design variables associated child elements of subproblem i
$\hat{\mathbf{x}}_i$	internal shared design variables for subproblem i
\mathcal{X}	set of feasible design points
\mathbf{y}	coupling variable vector
\mathbf{y}_p	consistent coupling variable vector
\mathbf{y}_{ij}	coupling variable vector from \mathbf{a}_j to \mathbf{a}_i
$\bar{\mathbf{y}}_i$	external coupling variables input to subproblem i
$\bar{\mathbf{y}}_{ij}$	coupling variable vector from \mathbf{P}_j to \mathbf{P}_i
$\bar{\mathbf{y}}_{\mathcal{C}_i}$	external coupling variables associated child elements of subproblem i
$\hat{\mathbf{y}}_i$	internal coupling variable vector for subproblem i
\mathbf{z}_i	vector of linking variables associated with \mathbf{a}_i
$\bar{\mathbf{z}}$	vector of all external linking variables
$\tilde{\mathbf{z}}$	vector of all copies of linking variable z
$\bar{\mathbf{z}}_i$	vector of external linking variables associated with subproblem i
$\bar{\mathbf{z}}_{ij}$	external linking variables between subproblems i and j
$\bar{\mathbf{z}}_{\bullet i}$	subproblem i output vector

Abstract

Successful design of complex modern products is a grand challenge for design organizations. The task is becoming increasingly important due to economic competition and concern over safety, reliability, and energy efficiency. Automotive and aerospace products, for example, are composed of numerous interdependent subsystems with a level of complexity that surpasses the capability of a single design group. A common approach is to partition complex design problems into smaller, more manageable design tasks that can be solved by individual design groups. Effective management of interdependency between these subproblems is critical, and a successful design process ultimately must meet the needs of the overall system. Decomposition-based design optimization techniques provide a mathematical foundation and computational tools for developing such design processes. Two tasks must be performed so that decomposition-based design optimization can be used to solve a system design problem: partitioning the system into subproblems, and determining a coordination method for guiding subproblem solutions toward the optimal system design. System partition and coordination strategy have a profound impact on the design process. The effect of partitioning and coordination decisions have been studied independently, while interaction between these decisions has been largely ignored. It is shown here that these two sets of decisions do interact: how a system is partitioned influences appropriate coordination decisions, and vice versa. Consequently, addressing partitioning and coordination decisions simultaneously leads to improved system design processes. The combined partitioning and coordination decision problem is a difficult combinatorial problem. An evolutionary algorithm that solves this decision problem effectively is presented. The set of all partitioning and coordination options for a specific formulation framework, augmented Lagrangian coordination (ALC), is derived, and a method for choosing Pareto-optimal solutions from amongst these options is described. Concepts and techniques are demonstrated using several engineering example problems. A detailed model for an electric vehicle design problem is presented that considers three vehicle systems: powertrain, chassis, and structure, and partitioning and coordination decisions for this problem are analyzed.

Chapter 1

Decomposition-based Design Optimization

Many products designed by engineers are too complex to be addressed by a single designer or even a single design group. Examples of products that are complex systems include aircraft, automobiles, and electronics. A common approach for addressing challenges associated with complex system design is to divide the product design task into smaller and more manageable design problems. For example, separate groups involved in automotive design may each be working on different vehicle systems, such as powertrain, frame, or chassis design. A primary challenge in this approach arises from interactions between the smaller design problems. These subproblems cannot be solved in complete isolation from each other. Decisions in one subproblem affect what decisions should be made in other subproblems. It is essential to coordinate subproblem solution such that the resulting subsystem designs are consistent with each other, and so that the subsystem designs together comprise an overall design that is optimal for the entire system, not just optimal for individual parts.

A decomposition-based approach to engineering system design requires considerable forethought before design activities can commence. Not all ways of dividing, or *partitioning*, a system are equal. Some partitions will enhance the effectiveness of the design process. In addition, the strategy for coordinating subproblems has significant influence over design process success. An appropriate system partition and an effective coordination strategy must be developed before the design process is launched. Choice of partition and coordination strategy should be considered together. The form of a system partition will influence how the subproblems are most effectively coordinated, and the intended coordination strategy will affect partitioning decisions. This dissertation addresses how to make partitioning and coordination decisions that lead to less complex and more effective decomposed design processes.

Computer-aided engineering (CAE) tools have reached the level of sophistication required for widespread use in engineering design. Engineering products can be designed and prototyped in a virtual environment, and then tested using computer simulations. Mathematical optimization techniques can be used to vary product design variables and test

virtual prototypes in search of designs that are optimal with respect to some criteria; optimization algorithms can reduce the number of virtual tests required to identify an optimal design. Application of CAE and optimization algorithms reduces the need for costly physical prototypes and expedites product development.

Complicated products can also be simulated successfully using CAE and designed using optimization algorithms. Often this involves the use of several separate, but interacting, CAE tools. The complexity of the simulations and the difficulty of the associated optimization problem can present a significant challenge. Many times a single optimization algorithm cannot manage the design of the system because of the large number of design variables or constraints, or because of the complex nature of the system. As with the human-based design process, a simulation-based design process can also be partitioned into smaller and easier to solve subproblems. Interactions exist between subproblems, requiring some type of coordination strategy to guide repeated subproblem solutions toward a consistent state and a design that is optimal for the entire system. Numerous methods have been developed to solve simulation-based design problems in this manner. We refer to this approach as decomposition-based design optimization.

Application of decomposition-based design optimization to a simulation-based system design problem also requires the a priori definition of a system partition and coordination strategy. The relationships between partitioning and coordination decisions are studied here. The results provide a deeper understanding into effective application of decomposition-based design optimization, aiding system designers in refining simulation-based design processes. Techniques for making optimal partitioning and coordination decisions, specifically for simulation-based design, are introduced here. Application of these techniques can further reduce product development time and cost when simulation-based design is a significant component of a product development process.

This chapter provides an overview of engineering system design and the use of mathematical optimization for engineering design. An overview of decomposition-based design optimization is then provided, followed by an example design problem that illustrates the impact of partitioning and coordination decisions on system design.

1.1 Engineering System Design

Complex systems are composed of several interacting members. Overall system behavior is not the simple sum or collection of constituent member behavior, but is something distinct that emerges from intricate member interrelationships. Analysis and design of complex systems requires something more than a dissociated approach; interactions must be

acknowledged, studied, and exploited to extract superior results.

Many modern engineered products are congruent with the above description of complex systems. They are composed of numerous subsystems and components interrelated through complex interactions. If each subsystem is designed independently, the resulting system will not reach its potential; it may not even work. Effects of system interactions can be discovered empirically, but this is an expensive and slow process. Formal techniques that explicitly manage interactions can enhance the system design process.

A typical system design approach used in industry involves designing subsystems in sequence. For example, consider an illustrative automotive design process where the structural, powertrain, chassis, and interior subsystems are designed in sequence. This approach does not account explicitly for all design interactions. A simplified schematic of this process is shown in Fig. 1.1. In this example the structural design is performed first, and is based on top-level vehicle requirements. Structural design also depends on the needs of the other subsystems, but since these have not been designed yet assumptions are made based on previous experience. The structural design is then fixed, and the powertrain is designed next based on top-level vehicle requirements. The process continues in sequence until the entire vehicle is designed.

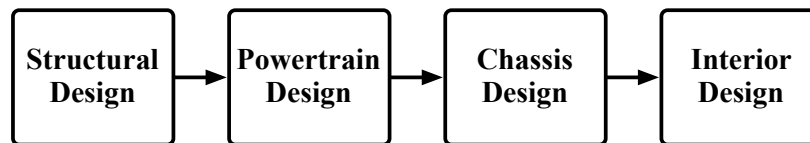


Figure 1.1 Sample sequential design process: automotive example

At later stages of the sequential process much of the system design is fixed and very little design freedom is allowed. For example, a designer in the last stage may discover that a small change in wheelbase would allow for a dramatically improved interior design, but since previous design tasks are fixed this adjustment cannot be made. Tradeoffs like this cannot be explored effectively in a sequential approach. Any interaction that is a feedback relationship is difficult to understand and manage since it is not handled explicitly. In this example feedbacks are addressed using past experience.

When a design process relies heavily on previous experience to handle interactions, new designs are limited to small perturbations from past designs [145]. Design organizations that use such processes face a significant challenge when attempting to design a product with a new topology and unfamiliar interactions. The automotive industry is currently being stretched in its ability to design vehicles with unconventional architectures, such as battery

electric (BEV) and hybrid electric (HEV) vehicles. Vehicle subsystems behave and interact in sometimes unexpected ways. Engineers can no longer rely on experience and intuition to resolve all difficulties that arise due to system interactions. Market pressure demands solutions before resolution can be obtained through experience. More sophisticated system design processes can help tackle this evolving problem.

The sequential process described above can be regarded as a single iteration of the block coordinate descent (BCD) algorithm [20]. The process could be iterated in an effort to address feedback interactions and generate an optimal system design, but the number of iterations required may exceed available time or resources. A more effective technique that addresses the needs of system design specifically is required. BCD does not retain enough design freedom in later stages, and design decisions do not account explicitly for effects on the complete system. Design freedom should be extended further into the process, and more complete system knowledge needs to be available and used earlier in the process [55].

1.2 Design Optimization

Mathematical optimization techniques are useful when quantitative decisions are to be made based on criteria that can be expressed using a mathematical model. These techniques have been applied for decades to operations research problems such as supply chain, scheduling, or network problems [22]. More recently, optimization has been established as a tool for engineering design [110]. This section reviews briefly fundamentals of optimization, and then discusses how to apply optimization to engineering design.

Mathematical optimization techniques seek to find the minimum of a function of n variables without evaluating all possible variable values. The function to be minimized is the objective function $f(\mathbf{x})$, and the variables it depends on are represented by the vector \mathbf{x} . All vectors in this dissertation are assumed to be row vectors unless otherwise noted. The variables can be real-valued ($\mathbf{x} \in \mathbb{R}^n$) or discrete; discrete variables may be integer ($\mathbf{x} \in \mathbb{Z}^n$), binary ($\mathbf{x} \in \{0, 1\}^n$), or categorical (e.g., $x \in \{\text{steel, aluminum, carbon fiber composite}\}$). Constrained optimization problems have limits on the values design variables may assume. Equality constraints require that design variables satisfy a relation exactly, while inequality constraints place bounds on values that design variables can assume. A constrained optimization problem in negative null form is stated as:

$$\begin{aligned}
& \min_{\mathbf{x}} && f(\mathbf{x}) \\
& \text{subject to} && \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\
& && \mathbf{h}(\mathbf{x}) = \mathbf{0},
\end{aligned} \tag{1.1}$$

The objective function is scalar valued, while the inequality ($\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$) and equality ($\mathbf{h}(\mathbf{x}) = \mathbf{0}$) constraints may be either scalar or vector valued. An optimization problem may have just equality constraints, just inequality constraints, or both. If the variables are continuous and the objective and all constraint functions are linear, then the optimization problem is known as a linear program (LP); the class of methods used to solve problems of this form is linear programming [38]. A more general class of optimization problems allows for nonlinear objective and constraint functions, but still requires that $\mathbf{x} \in \mathbb{R}^n$. These optimization problems are known as nonlinear programs (NLP), and nonlinear programming refers to techniques for solving NLPs [20]. Figure 1.2 illustrates an example NLP.

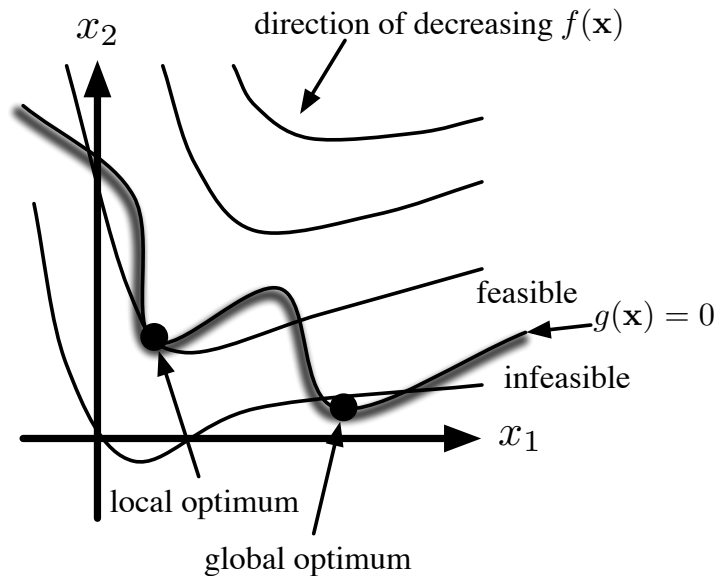


Figure 1.2 Illustration of global and local optima in a nonlinear programming example

The objective function in this example decreases in the direction of decreasing x_1 and x_2 . Level sets indicate the shape of $f(\mathbf{x})$. The line $g(\mathbf{x}) = 0$ is plotted. In this example $g(\mathbf{x})$ increases with decreasing x_1 and x_2 , so the region where the inequality constraint $g(\mathbf{x}) \leq 0$ is satisfied is the region above and to the right of the line $g(\mathbf{x}) = 0$. This problem has no equality constraints. The region in the variable space where all constraints are satisfied is known as the feasible region, and is indicated in the figure.

A point \mathbf{x} is optimal if moving in any feasible direction from the point causes an increase

in the objective function value. The symbol \mathbf{x}^* denotes an optimal variable vector. If \mathcal{X} is the set of all feasible points, a direction $\mathbf{s} \in \mathbb{R}^n$ is feasible if there exists a scalar $\alpha_u > 0$ such that $\mathbf{x} + \alpha\mathbf{s} \in \mathcal{X}$ for all $0 \leq \alpha \leq \alpha_u$. The example in Fig. 1.2 has two constrained optima. In each case the inequality constraint is active. A constraint is active if its removal changes the location of \mathbf{x}^* . Active inequality constraints are satisfied with equality. If a problem has multiple optima, the optimum with the lowest value is the global optimum. All others are local optima. A constrained NLP may have an unconstrained optimum if there exists a point in the interior of \mathcal{X} where all possible directions \mathbf{s} lead to increased $f(\mathbf{x})$.

The preceding paragraph informally describes optimality conditions for NLPs. In some cases optimality conditions can be used to derive a system of equations that can be used to solve for \mathbf{x}^* directly. Monotonicity analysis (MA) is another solution approach that exploits knowledge of monotonicity in objective and constraint functions to identify active constraints [110]. If MA cannot be used to completely solve a problem, it can possibly help reduce its complexity and provide helpful insights. If direct solution or solution through MA is not possible, an iterative algorithm for NLPs may be employed. Most are based on objective and constraint function gradients. Gradient-based algorithms can be applied only when functions are continuous and smooth. If a problem is also convex¹ a gradient-based algorithm will identify the global optimum, but without convexity these algorithms will find only local optima. Examples of successful gradient-based algorithms for NLPs include the generalized reduced gradient (GRG) method [91] and sequential quadratic programming (SQP) [69, 111]. Much of the development in subsequent chapters assumes that the optimization problems under consideration can be solved using a gradient-based algorithm.

Gradient-based algorithms are normally computationally efficient, i.e., they converge quickly and require relatively few function evaluations when compared to gradient-free methods. Unfortunately, gradient-based methods can fail when functions are non-differentiable or numerically noisy. They typically do not handle problems with discrete variables, and can perform inconsistently when applied to ill-conditioned problems. Several gradient free methods have been developed as alternatives. Evolutionary algorithms (EAs) use principles of natural selection to gradually improve the quality of a population of candidate solutions over several generations [49, 73]. Simulated annealing (SA) is another heuristic algorithm; it considers single design points (rather than a population) and uses a stochastic search technique with a positive probability of selecting worse points to aid escaping local optima [81]. This algorithm's name comes from the 'cooling schedule' that guides the probability of selecting an inferior point during the search. Another class of stochastic search algo-

¹A problem is convex if $f(\mathbf{x})$ is a convex function and if \mathcal{X} is a convex set.

rithms selects random points around a candidate solution to evaluate and determine a new candidate solution point [130]. As with SAs, these algorithms are not population-based. The distribution of search points is influenced by information obtained during the optimization process.

Heuristic methods such as EAs and SAs are not based on optimality conditions, so cannot guarantee that the solution obtained is an optimum. They can offer good approximations to the optimum for problems that cannot be solved with gradient-based methods. Another disadvantage of these methods is the large number of function evaluations that typically are required. An alternative exists that is based on optimality conditions involving subgradients, which are in essence a type of gradient for non-smooth functions defined in Clarke's calculus [32]. This alternative method is mesh adaptive direct search (MADS), and is positioned in the spectrum of optimization methods between gradient-based methods and heuristic methods [14]. MADS can handle non-smooth problems, but normally does not require as many function evaluations as EAs or SAs.

Engineering design problems can be posed naturally as constrained optimization problems. Normally at least one metric is readily identified as an objective function, such as some key performance metric. Engineering design problems are full of design requirements that can be expressed as constraints (e.g., stress, deflection, packaging, temperature, or vibration requirements). Constraints based on design requirements are termed design constraints. Objective and constraint functions can be calculated using analytical expressions or computer simulations. The inputs to these expressions or simulations are quantities that parametrically characterize the design of a product, such as physical geometry. A subset of these quantities is chosen as the set of design variables, \mathbf{x} . Design variables are allowed to vary in the design problem, while the remaining function inputs, termed design parameters, are held fixed when solving a design optimization problem. In summary, in engineering design optimization we seek to optimize the performance of a product subject to design constraints, with respect to design variables.

In some system optimization formulations the optimization algorithm selects values for quantities that are not design variables. To avoid confusion in terminology, we define decision variables to be any quantity that are variables in the design optimization problem. The set of decisions variables includes design variables and possibly other quantities.

Design optimization is a logical extension to CAE software. Tremendous effort has been devoted to development of accurate and efficient software for engineering analysis. These software can be used to analyze existing prototypes or products, or used as tools to test design alternatives proposed by engineers before fabrication. Engineers can also use CAE software to test manually adjusted design variables. This process can help build intuition

for a product, and can be a good approach when only a few design variables are used, but becomes unwieldy as the number of design variables increases. While manual search may be impractical in many cases, optimization algorithms can trace an efficient path toward the optimal design. A design optimization approach requires fewer function evaluations and leads to a superior design.

The effectiveness of design optimization depends on our ability to model accurately actual product behavior. The solution to a design optimization problem does not generate the optimal design for the real product, but the optimal design for a virtual product as represented by a mathematical model. CAE tools are becoming more mature and offer increasing levels of modeling accuracy, and have been a factor in the success of design optimization as a practical engineering tool.

1.3 Decomposition-based Design Optimization

Applying design optimization techniques to engineering system design presents additional challenges. As with general engineering system design, approaching the design of a system as a single entity may be impractical. A single optimization algorithm may not be capable of handling the demands of designing an entire system. The system optimization problem can be partitioned into smaller subproblems, each solved separately. The subproblems are linked through common design variables and analysis interactions. Subproblems must be solved in a way that leads to a system design that accounts for these links, and is optimal for the entire system. The task of guiding subproblem solutions toward an optimal system design is called coordination. This approach to engineering system design is known as decomposition-based design optimization.

A broad class of methods for decomposition-based design optimization, known as multidisciplinary design optimization (MDO), has been developed to address industry needs for engineering system design. Most have been developed for situations where several engineering analyses must be integrated for designing a single component or product, where each analysis pertains to a different aspect of the same component. Aeroelastic design is a standard example of an MDO application. Suppose an airplane wing is to be designed, giving heed to both structural and aerodynamic considerations. If the wing is sufficiently stiff, we can assume that the wing does not deform much when subject to aerodynamic pressure. This allows us to use the undeformed wing shape when conducting the aerodynamic analysis, making the aerodynamic analysis independent of the structural analysis. If the wing is not sufficiently stiff for this assumption to hold, the aerodynamic analysis should be based on the deformed wing shape as computed by the structural analysis. The structural analysis in

turn requires the aerodynamic pressure distribution over the wing surface to compute the deformed shape. The two analyses are coupled, as depicted in Fig. 1.3.

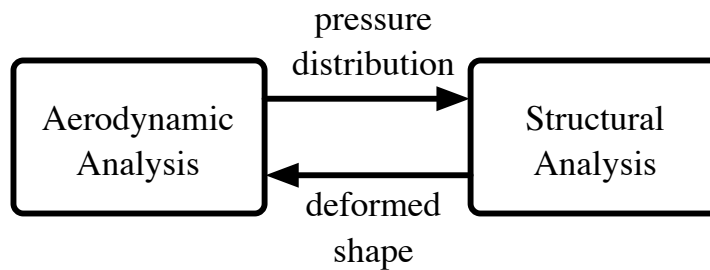


Figure 1.3 Aeroelastic analysis

Iterative techniques are required to find a state of pressure and deformation such that the analyses are consistent with each other. The task of solving coupled analyses simultaneously is known as multidisciplinary analysis (MDA). Multiphysics software has been developed to address specific types of MDA.

The presence of analysis coupling complicates system design. Many MDO methods are designed to manage this type of coupling. Cramer et al. identified several important MDO formulations [35], and Sobieski and Haftka provided an extensive review of early MDO methods [128]. Industry needs motivating MDO development are summarized in [55] and [61]. Common points include the need to compress design cycle time, improve product quality, increase design flexibility, and more competently characterize, manage, and exploit system interactions. Chapter 4 introduces and demonstrates several important methods for decomposition-based design optimization. These methods are more broadly applicable than to just systems partitioned by discipline; partitions may be along disciplinary, physical, process boundaries, or some combination thereof.

Implementation of decomposition-based design optimization requires the completion of two important preliminary steps. First, a system partition must be defined, and then a strategy for coordinating the solution of the resulting subproblems must be constructed. Making partitioning and coordination decisions can be viewed as a preprocessing step for optimal system design. Figure 1.4 illustrates these preliminary steps. The original, unpartitioned system is depicted in Fig. 1.4(a), where the vertices represent components of a system or analyses pertinent to system design, and the edges connecting the vertices represent interactions between the components or analyses. The first step is to decide which subproblem each component should belong to. The outcome of this step is a system partition, shown in Fig. 1.4(b). Once the partition is defined, a coordination strategy can be devised. An important aspect of many coordination strategies is the subproblem solution sequence,

illustrated in Fig. 1.4(c).

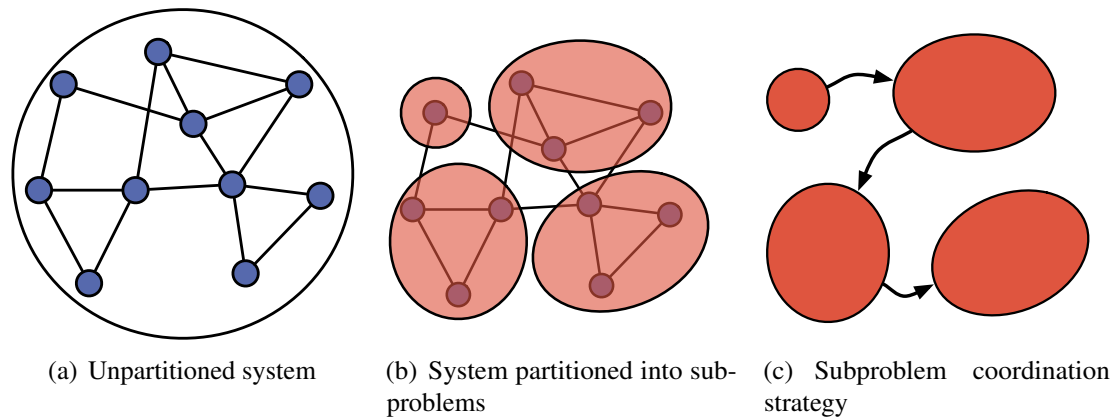


Figure 1.4 Process for implementing decomposition-based design optimization

While the focus here is on simulation-based design, an organizational analogy is instructive. Suppose the vertices in Fig. 1.4(a) represent positions or tasks within a design organization. Before a large organization can embark on product design, it needs structure [36]. First it is partitioned into groups. These groups can be formed according to discipline, project, or hybrid divisions. Interaction must occur both within and between groups. Patterns of interaction along with an organizational partition define organizational structure. Interaction patterns, analogous to coordination, can take several different forms. Bureaucratic organizations are structured into a hierarchy, which can be orderly and efficient in certain cases. Horizontal organizations are non-hierarchical; these can be more adaptable to changes than hierarchic organizations.

The way an organization is partitioned will influence what interaction patterns are most effective. In addition, the type of interaction patterns desired (e.g., hierarchic vs. non-hierarchic) will influence partitioning decisions. These two sets of decisions are in essence coupled. Moving back to the context of decomposition-based design optimization, system partitioning and coordination decisions are also coupled. How a system is partitioned will influence coordination decisions, and vice versa. This relationship is pictured in Fig. 1.5. Proving and investigating the relationship between partitioning and coordination decisions is the primary theme of this dissertation. Partitioning and coordination decisions have been studied independently, but the relationship between them has not been systematically analyzed. Subsequent chapters show that partitioning and coordination decisions are in fact coupled. Techniques are introduced for analyzing these decisions, studying intrinsic tradeoffs, and making optimal partitioning and coordination decisions for important classes of decomposition-based design optimization methods.

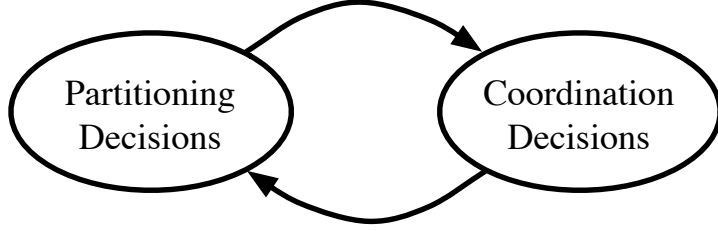


Figure 1.5 Dissertation hypothesis: Existence of coupling between partitioning and coordination decisions

1.4 Introductory Example

A simple design problem is used here to demonstrate the influence of partitioning and coordination decisions on computational expense. We assume that all system design optimization methods considered here are capable of finding the optimal system design, so improving computational efficiency is the objective of interest. This example design problem is based on four analytical functions; it is abstract and does not correspond to a physical system. It was chosen because it possesses an interesting analysis structure that will be useful for this demonstration, but is still simple enough to present concisely. The principles discussed here apply also to more sophisticated systems that involve CAE simulations. The functions in this problem are interdependent, forming a system of coupled equations:

$$\begin{aligned}
 a_1(x_1, y_{12}, y_{13}) &= 0.1x_1y_{12} + 0.8x_1y_{13} + b_1 \\
 a_2(x_2, y_{23}) &= x_2y_{23} + b_2 \\
 a_3(x_3, y_{31}, y_{32}) &= 0.1x_3y_{31} + 0.8x_3y_{32} + b_3 \\
 a_4(\mathbf{x}, y_{41}) &= (y_{41} - \hat{y}_{41})^2 + \|\mathbf{w} \circ \mathbf{x}\|_2^2
 \end{aligned}$$

where $\mathbf{b} = [2.0, 2.5, 3.0]$, $\mathbf{w} = [1.3, 1.5, 1.2]$, and the symbol \circ denotes the Hadamard product².

Each of the analysis functions in this system depends on at least one design variable and one coupling variable. The vector of all design variables is $\mathbf{x} = [x_1, x_2, x_3]$. Previous sections described how some analyses in a system can depend on the output of other analyses. Coupling variables are the quantities that are communicated between coupled analyses. For example, the analysis function a_1 depends on the outputs of a_2 and a_3 , indicated by the dependence of a_1 on the coupling variables y_{12} and y_{13} . The quantity passed from a_2 to a_1 is y_{12} , and the quantity passed from a_3 to a_1 is y_{13} . The vector of all coupling variables for

²The Hadamard product is element-by-element vector multiplication. For example, $[w_1, w_2, w_3] \circ [x_1, x_2, x_3] = [w_1x_1, w_2x_2, w_3x_3]$

the example system is $\mathbf{y} = [y_{12}, y_{13}, y_{23}, y_{31}, y_{32}, y_{41}]$. In general, the coupling variable y_{ij} is the quantity passed from analysis function j to analysis function i . Although some coupling variables in this system represent the same quantity, many decomposition-based design optimization methods utilize multiple coupling variable copies. This requires an explicit distinction between all coupling variables. Coupling variable and other system optimization notation will be defined more precisely in the following chapter.

The design problem based on the four analysis functions described above is an unconstrained minimization problem in three variables:

$$\min_{\mathbf{x}=[x_1, x_2, x_3]} a_4(\mathbf{x}, y_{41}) \quad (1.2)$$

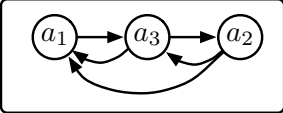
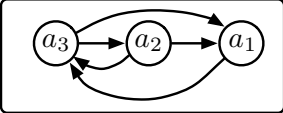
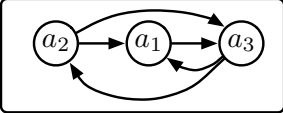
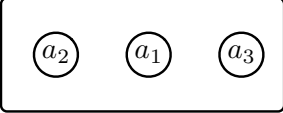
The optimization algorithm will choose iteratively new values for \mathbf{x} as it seeks a minimum value for a_4 . Since a_4 directly or indirectly depends on all design and coupling variables, all four analysis functions must be evaluated for the value of \mathbf{x} provided by the optimization algorithm at each iteration. Observe that this system possesses circular dependence between functions, also called feedback coupling. For example, a_1 depends on the output of a_2 , which depends on the output of a_3 , which depends on the output of a_1 . Therefore, a_1 cannot be computed using a sequential evaluation technique; an iterative scheme is required. A common approach, known as fixed point iteration (FPI), starts with a guess for unknown coupling variable values, evaluates the analysis functions in sequence, updates the initial guesses, and repeats until coupling variable values converge to a ‘fixed point’ [28]. At convergence the system is said to have coupling variable consistency. FPI does not always converge, and should be used with caution [6]. Coupling variable consistency is discussed in Section 2.1, and FPI convergence is addressed in Section 4.1.

Solution to Problem 1.2 implicitly requires a nested approach where an analysis algorithm, such as FPI, must be used to find a consistent set of coupling variables at every optimization iteration because of feedback coupling. In other words, the optimization problem is an outer loop process that seeks to find an optimal design vector, while the analysis algorithm is an inner loop process that seeks to find a consistent coupling variable vector for every design vector considered along the way. The computational expense added by the inner loop process can be considerable, and steps should be taken to minimize it. Analysis feedback coupling can be difficult to handle, but is present in many important engineering design problems. This dissertation discusses several techniques for understanding and managing analysis feedback coupling in system design optimization.

1.4.1 Coordination Decisions

Analysis function evaluation sequence can have a significant effect on inner loop expense [134]. Several evaluation sequences for the inner loop in Problem 1.2 were tested, and the number of function evaluations (NE) for each approach was recorded. The results using FPI as the inner loop algorithm are recorded in Table 1.1. The diagrams illustrate analysis function evaluation sequence and coupling variable communication patterns. When using FPI, it was observed that the sequences with fewer feedback coupling relationships had reduced computational expense. While this frequently is the case, it is possible for sequences with fewer feedbacks to require more NE due to relationships between functions.

Table 1.1 Experimental results for evaluation sequence variation
 Evaluation Sequence # Feedbacks NE (FPI) NE (EC)

	3	1476	392
	2	1162	504
	2	1138	208
	—	—	364

An important aspect of many coordination strategies is solution sequence; the foregoing results illustrate the importance of selecting a good sequence. Another essential component of coordination is the technique for satisfying consistency requirements. One option for satisfying these requirements is an iterative algorithm, such as FPI, that enforces consistency at every design step. An alternative approach uses the optimization algorithm to solve directly for a consistent coupling variable vector. This is done by adding \mathbf{y} to the set of decision variables for the optimization problem, and by adding auxiliary equality constraints to the optimization problem that ensure coupling variable consistency at optimization algorithm convergence. The coupling variable consistency problem is in effect a system of equations defined by the auxiliary equality constraints. The nested approach described above uses FPI to solve the coupling variable consistency problem at each optimization

iteration. The auxiliary equality constraint approach uses the optimization algorithm to solve the coupling variable consistency problem in tandem with the optimization problem, which can provide a computational advantage in some cases. The formulation for the example problem using this equality constraint approach is:

$$\begin{aligned}
& \min_{\mathbf{x}, \mathbf{y}} && a_4(\mathbf{x}, y_{41}) \\
\text{subject to} &&& y_{12} - a_2(x_2, y_{23}) = 0 \\
&&& y_{13} - a_3(x_3, y_{31}, y_{32}) = 0 \\
&&& y_{23} - a_3(x_3, y_{31}, y_{32}) = 0 \\
&&& y_{31} - a_1(x_1, y_{12}, y_{13}) = 0 \\
&&& y_{32} - a_2(x_2, y_{23}) = 0 \\
&&& y_{41} - a_1(x_1, y_{12}, y_{13}) = 0
\end{aligned} \tag{1.3}$$

The fourth row of Table 1.1 corresponds to the solution approach defined in Problem 1.3. In this example a substantial computational benefit is realized. A hybrid coordination strategy exists that incorporates aspects of the nested approach and the equality constraint approach. Balling and Sobieski suggested another alternative approach where the analysis functions are solved once in sequence, and auxiliary equality constraints are used to handle feedback coupling relationships only, rather than all coupling relationships [15]. This technique was used to solve the example problem using the three sequences listed in Table 1.1, and the results are given in the last column. Using the first sequence, we have three feedback relationships to satisfy using equality constraints. The design formulation is:

$$\begin{aligned}
& \min_{\mathbf{x}, y_{12}, y_{13}, y_{32}} && a_4(\mathbf{x}, y_{41}) \\
\text{subject to} &&& y_{12} - a_2(x_2, y_{23}) = 0 \\
&&& y_{13} - a_3(x_3, y_{31}, y_{32}) = 0 \\
&&& y_{32} - a_2(x_2, y_{23}) = 0
\end{aligned} \tag{1.4}$$

Consistency for the feedback coupling variables y_{12} , y_{13} , and y_{32} is satisfied via optimization equality constraints, and consistency for y_{23} , y_{31} , and y_{41} is satisfied through analysis function sequencing. The second sequence option has two feedback relationships to satisfy:

$$\begin{aligned}
& \min_{\mathbf{x}, y_{31}, y_{32}} && a_4(\mathbf{x}, y_{41}) \\
\text{subject to} &&& y_{31} - a_1(x_1, y_{12}, y_{13}) = 0 \\
&&& y_{32} - a_2(x_2, y_{23}) = 0
\end{aligned} \tag{1.5}$$

The third sequence has two feedback couplings, and the corresponding formulation is:

$$\begin{aligned}
& \min_{\mathbf{x}, y_{13}, y_{23}} && a_4(\mathbf{x}, y_{41}) \\
\text{subject to} &&& y_{13} - a_3(x_3, y_{31}, y_{32}) = 0 \\
&&& y_{23} - a_3(x_3, y_{31}, y_{32}) = 0
\end{aligned} \tag{1.6}$$

This last approach proved to be most efficient out of all seven coordination options presented, requiring only 208 function evaluations. This is an 86% reduction from the first option, showing the importance of making proper coordination decisions. The effect of partitioning decisions on this example problem will now be explored.

1.4.2 Partitioning Decisions

Analysis functions may be grouped together to form blocks of a system partition. In distributed optimization an optimization problem is defined for each of these blocks. When a single optimization problem is employed, these blocks can be used to divide up system analysis. Two approaches are considered here. In both approaches FPI is employed within each block to achieve coupling variable consistency between functions inside a block. In the first approach FPI is also used to satisfy coupling variable consistency for relationships between blocks. In the second approach auxiliary equality constraints enforce consistency between blocks. The results are summarized in Table 1.2.

The first approach exhibited a dramatic increase in function evaluations for every partition. The number of evaluations is multiplied because there exist three levels of nesting in the solution approach. No significant difference exists between partitions for the first approach in this example problem. The second and third partitions required the exact same number of function evaluations; in this case swapping the order of analysis blocks has no impact on solution expense. The second approach applied to the second partition required only 366 function evaluations. While this is not the lowest number observed, the result is significant because the same $a_1 \rightarrow a_3 \rightarrow a_2$ sequence required 392 evaluations with the unpartitioned auxiliary equality constraint approach. This shows that synergy exists between partitioning and coordination decisions for this example. Subsequent chapters develop a

more rigorous approach to exploring the interaction between partitioning and coordination decisions, and illustrate the tradeoffs that exist in making these decisions.

1.5 Dissertation Overview

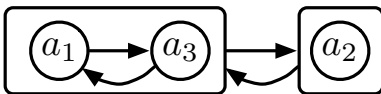
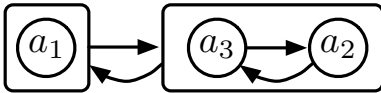
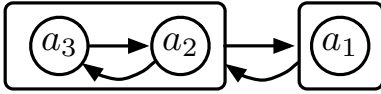
Chapter 1 introduced the concepts of engineering system design, design optimization, and decomposition-based design optimization. An introductory example was presented that illustrated the influence that partitioning and coordination decisions can have on the computational expense of solving a system design optimization problem. This example also demonstrated synergy between partitioning and coordination decisions, indicating that some type of interaction exists between these decisions. Subsequent chapters build on this result: a formal approach to making partitioning and coordination decisions is presented, and the relationship between them is studied.

Chapter 2 sets forth the terminology and notation used in decomposition-based design optimization. Formal techniques for compactly representing relationships within a system are reviewed, and previous techniques for making partitioning and coordination decisions are discussed.

Several engineering system design examples are used throughout this dissertation. The majority of these examples are described in Chapter 3. Enough detail is provided for each of the problems (with the exception of the aircraft family design problem) to facilitate replication. These examples may be reused to verify results or as test problems in other work.

Several important formulations for decomposition-based design optimization are pro-

Table 1.2 Experimental results for evaluation sequence and partition variation

Evaluation Sequence	<i>NE</i> (FPI)	<i>NE</i> (EC)
	3618	714
	3592	366
	3592	752

vided in Chapter 4. Examples are used to elucidate application of these formulations and to reveal properties relevant to partitioning and coordination decisions. Particular attention is paid to a class of formulations that is used in later chapters as the basis for partitioning and coordination decisions. Algorithms in this class have proven convergence properties and have the flexibility required to adapt to a wide range of problem structures.

Partitioning and coordination decisions for decomposition-based design optimization are formalized in Chapter 5. These decisions are shown to be coupled for three example systems. A technique is introduced for analyzing the tradeoff between coordination and subproblem solution expense. This analysis aids system designers in determining whether a decomposition-based approach should be used, and if so, what partitioning and coordination decisions may lead to reduced solution complexity. The techniques in Chapter 5 utilize exhaustive enumeration of partitioning and coordination options. While effective for small systems, exhaustive enumeration is impractical for use in analyzing larger systems. An evolutionary algorithm is presented in Chapter 6 that solves the optimal partitioning and coordination decision problem for larger systems. The evolutionary algorithm is applied to the three small example systems from Chapter 5, and its results are compared to the exact solutions obtained with exhaustive enumeration. The evolutionary algorithm is then applied to a structural design problem that is too large for exhaustive enumeration.

Coordination decisions addressed through Chapter 6 are limited to subproblem sequence. Another aspect of coordination decisions is how to structure the links between subproblems. This is addressed in Chapter 7 for a specific formulation called augmented Lagrangian coordination (ALC). A new class of parallel ALC implementations is introduced. ALC provides tremendous flexibility in linking structure, but manually determining an appropriate linking structure can be overwhelming due to the large number of possibilities. The techniques introduced in Chapter 7 provide a way to create systematically a linking structure tailored to the needs of a particular system, along with choosing a system partition and coordination strategy.

The examples used through Chapter 7 are of low to moderate complexity. A more involved design example is presented in Chapter 8. The design of a battery electric vehicle is addressed using decomposition-based design optimization. Several different vehicle systems are considered, including powertrain, structure, and chassis design. The model accounts for many important interactions, such as the influence of component sizing and location on vehicle dynamics. The optimal partitioning and coordination decision method presented in Chapter 7 is applied to the electric vehicle design problem. The concepts and results are summarized in Chapter 9. A systematic procedure is outlined for approaching partitioning and coordination decisions in system optimization and in other applications.

Chapter 2

Partitioning and Coordination Decisions

The preceding chapter introduced the concept of decomposition-based design optimization for engineering systems, and demonstrated that partitioning and coordination (P/C) decisions can have great impact on solution difficulty. This chapter constructs notation and terminology needed for more precise and detailed treatment of system design optimization and P/C decisions, reviews existing techniques for making P/C decisions, and establishes that no studies have addressed interaction between partitioning and coordination decisions or the potential impact of this interaction.

2.1 Decomposition-based System Design

The system design problems considered here involve multidisciplinary, coupled analyses where input/output properties are assumed to be known precisely. For example, in simulation-based design, each quantity of interest in the design problem is computed using a computer simulation, such as finite element analysis or multi-body dynamics simulation software. Each simulation must be provided a specific set of inputs to compute its corresponding outputs; this prescribes a definite information flow direction. Output quantities cannot be recast as inputs, and vice versa. This is contrast to equation-based design, where frequently one of several variables that appear in design equations can be selected as an output variable. Flexibility in the set of output quantities causes ambiguity in information flow. Exact knowledge of information flow in simulation-based design allows us to construct system design approaches that exploit directionality to reduce solution complexity.

Each simulation in a system design problem can be viewed as a vector-valued analysis function; it computes one output vector for every unique set of inputs. Simulations (i.e., analysis functions) frequently depend on outputs of other simulations in addition to design variables. When circular dependence exists among analysis functions the system is said to possess feedback coupling. Analysis function outputs can be design objectives, design constraints, or intermediate quantities. Figure 2.1 illustrates the possible relationships

between a system of m analysis functions.

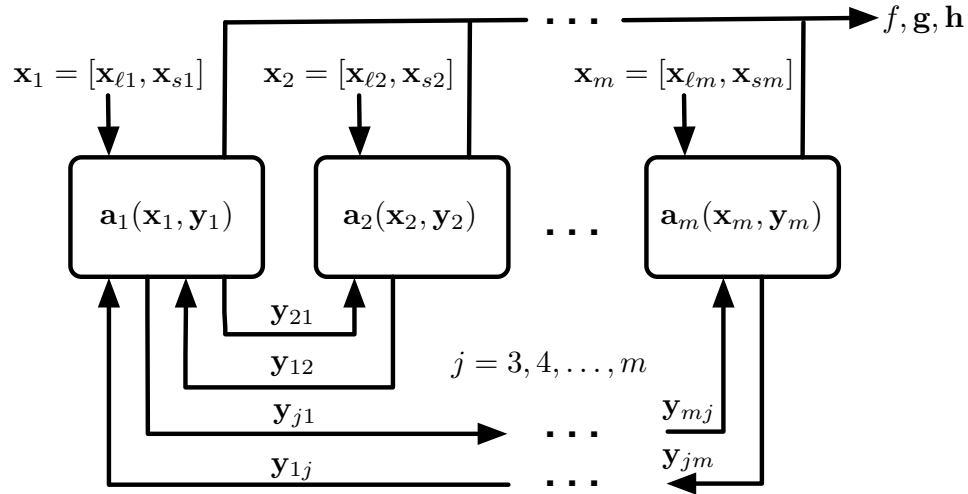


Figure 2.1 Input and output relationships for a system of analysis functions

The vector of quantities computed by the j -th analysis function and required as input to the i -th analysis function is termed analysis coupling variable y_{ij} . The vector of all coupling variables input to analysis i from any other analysis in the system is y_i , and all design variables required as input to analysis i form the vector x_i . In this manner, we define the i -th analysis function as $a_i(x_i, y_i)$. Design variables that are inputs to $a_i(x_i, y_i)$ only are local design variables $x_{\ell i}$; design variables that are inputs to $a_i(x_i, y_i)$ and at least one other function are shared variables $x_{s i}$. Shared and local variables together form $x_i = [x_{\ell i}, x_{s i}]$ (vectors are assumed to be row vectors). The collections of all design variables, coupling variables, and analysis functions are x , y , and $a(x, y)$, respectively. Shared and coupling variables for $a_i(x_i, y_i)$ comprise its set of linking variables z_i . Instantiations of the same quantities, e.g., variable, at different parts of the system will be referred to as copies of the quantity.

A system consists of several subsystems or components that frequently have competing needs. If each subsystem is optimized independently, the resulting system design may perform poorly. Effective system design addresses relationships between components, and seeks to improve overall system performance. Relationships within a system can take the form of analysis interactions (i.e., coupling variables), or shared design variables. A system design method must ensure that subsystems are compatible with each other by agreeing on coupling variable values and on shared design variables. A design method must also ensure that the system as a whole is optimized, rather than just the individual subsystems. The concepts of system consistency and optimality are discussed below, followed by a definition

of distributed optimization.

2.1.1 System Consistency

A system is consistent if the values of all copies of a shared variable agree for all shared variables, and if the value of every coupling variable is equal to its corresponding analysis output. In other words, system consistency is achieved if shared variable and coupling variable consistency requirements are met. It is essential that shared variable and coupling variable consistency are managed separately to achieve the most efficient solution processes for system design. System consistency is a necessary condition for system optimality.

Shared variable consistency is achieved if

$$x_q^{(k)} = x_q^{(l)} \quad \forall k \neq l, \quad k, l \in D_s(x_q) \quad (2.1)$$

is satisfied for all shared variables, where x_q is a component of \mathbf{x} that is shared among the analysis functions $\mathbf{a}_i(\mathbf{x}_i, \mathbf{y}_i) \forall i \in D_s(x_q)$, with $D_s(x_q)$ being the set of indices of analysis functions that depend on the shared variable x_q ; superscripts indicate the analysis function where the shared variable copy is input. A shared variable x_q is consistent if the value for x_q input to every analysis function that depends on x_q is identical.

Coupling variable consistency is achieved, if for every coupling variable,

$$\mathbf{y}_{ij} - \mathbf{a}_j(\mathbf{x}_j, \mathbf{y}_j) \mathbf{S}_{ij} = \mathbf{0} \quad (2.2)$$

is satisfied, where the boolean matrix \mathbf{S}_{ij} selects the components of \mathbf{a}_j that correspond to \mathbf{y}_{ij} . The set of all such equality constraints is $\mathbf{y} - \mathbf{a}(\mathbf{x}, \mathbf{y}) \mathbf{S} = \mathbf{0}$, where \mathbf{S} is a selection matrix that extracts the components of $\mathbf{a}(\mathbf{x}, \mathbf{y})$ that correspond to \mathbf{y} . These coupling variable consistency constraints are referred to as the system analysis equations. Equations (2.1) and (2.2) together form the system consistency constraints.

2.1.2 System Optimality

A system design optimization approach should ensure the resulting design is optimal with respect to the system's primary purpose. A single objective function that effectively represents this purpose should be selected. Section 1.1 described how design approaches that do not consider all aspects of a system together may not identify designs that are system optimal. An effective design approach accounts for all system interactions, optimizes the system objective function, and ensures system consistency. The following formulation meets

these requirements:

$$\begin{aligned}
 & \min_{\mathbf{x}} && f(\mathbf{x}, \mathbf{y}_p(\mathbf{x})) && (2.3) \\
 & \text{subject to} && \mathbf{g}(\mathbf{x}, \mathbf{y}_p(\mathbf{x})) \leq \mathbf{0} \\
 & && \mathbf{h}(\mathbf{x}, \mathbf{y}_p(\mathbf{x})) = \mathbf{0},
 \end{aligned}$$

where $\mathbf{y}_p(\mathbf{x})$ is a solution to the system analysis equations for a given design, and the objective and constraint function values are outputs of a subset of analysis functions. This formulation is known as multidisciplinary feasible (MDF) [35] or All-in-One (AiO), and implicitly achieves shared variable consistency. For every optimization iterate \mathbf{x} the system analysis equations must be solved for $\mathbf{y}_p(\mathbf{x})$ to satisfy Eq. (2.2) for every coupling variable. This nested approach ensures both coupling variable and shared variable consistency, as well as minimizes the system objective function $f(\mathbf{x}, \mathbf{y}_p(\mathbf{x}))$.

If no feedback loops exist among analysis functions, the system analysis equations can be satisfied simply by executing the analysis functions in the proper sequence; analysis function outputs provide the coupling variable values directly. An iterative algorithm is required for system analysis if feedback loops exist. Alternatively, the optimization algorithm can solve for $\mathbf{y}_p(\mathbf{x})$ using equality constraints to enforce coupling variable consistency. This enables coarse-grained parallel processing and can ease numerical difficulties associated with strongly coupled analysis functions [6]. The Individual Disciplinary Feasible (IDF) formulation is the simplest way to use this approach [35]. Balling and Sobieski suggested a hybrid approach that uses function sequencing to satisfy feedforward coupling relationships, and equality constraints to satisfy feedback coupling relationships [15]. This hybrid approach was demonstrated in Section 1.4.

2.1.3 Distributed Optimization

The two formulations for system optimization discussed above, MDF and IDF, are known as single-level formulations because a single optimization algorithm is involved. Single-level approaches become unwieldy when the number of design variables or constraints is large enough to approach optimization algorithm reliability or cost limits. Large systems can be divided into several smaller optimization problems, called subproblems. Interactions between subproblems must be managed such that the final result is consistent and optimal for the entire system. This is accomplished through a coordination strategy that works with the individual subproblems. This type of approach is known as multi-level or distributed

optimization, because optimization tasks are distributed throughout the system. Chapter 4 details several single and multi-level formulations, and discusses their appropriate application. The emphasis in subsequent chapters will be on partitioning and coordination decisions for problems solved using distributed optimization formulations.

Distributed optimization can employ equality constraints or penalty functions within subproblem optimization formulations to help satisfy system analysis equations. The set of design variables that are inputs for the functions in subproblem i and at least one other subproblem are the external shared variables \bar{x}_{si} . Coupling variables passed from functions in subproblem j to subproblem i are the external coupling variables \bar{y}_{ij} . External shared and coupling variables for subproblem i comprise its set of external linking variables \bar{z}_i . Independent subproblem solution requires local copies of both external coupling and shared variables. The coordination algorithm must ensure all copies match at convergence, satisfying the system consistency constraints. Some examples of coordination algorithms include optimization algorithms (e.g., Collaborative Optimization (CO) [25]), fixed point iteration (e.g., Analytical Target Cascading (ATC) [80]), Newton’s method (e.g., ATC [139]), and penalty methods (e.g., ATC and Augmented Lagrangian Coordination (ALC) [141]). Distributed methods are most beneficial when systems are large and sparsely connected [128], when the design environment is distributed [29], or when specialized optimization algorithms can be exploited for solving particular subproblems [88, 105].

2.2 Decision Framework

A method for decomposition-based design optimization (i.e., a *decomposition method*) is defined here to include both a system partition and a coordination strategy [145]. Before we can use a decomposition method for solving a system design problem, a system partition and a coordination strategy must be defined. The partitioning problem (P) is to decide which of m analysis functions should be clustered into each of the N subproblems. The coordination decision problem (C) is to specify a method for satisfying system consistency and system optimality requirements; this typically involves both an approach to consistency constraint management (i.e., linking structure) and an algorithm for guiding repeated subproblem solutions toward system optimality and consistency. Linking structure is reflected in problem formulation. Different types of problem formulations allow for different types of partitions, linking structure, and coordination algorithms. Problem formulation is therefore an elemental aspect of problem coordination. Figure 2.2 illustrates key aspects of the partitioning and coordination decision problems. Decision techniques presented through Chapter 6 consider the subproblem sequence aspect of coordination decisions only. Chapter

7 develops the theory required to understand the linking structure aspect of the coordination decision problem and demonstrates how to incorporate linking structure into partitioning and coordination decisions.

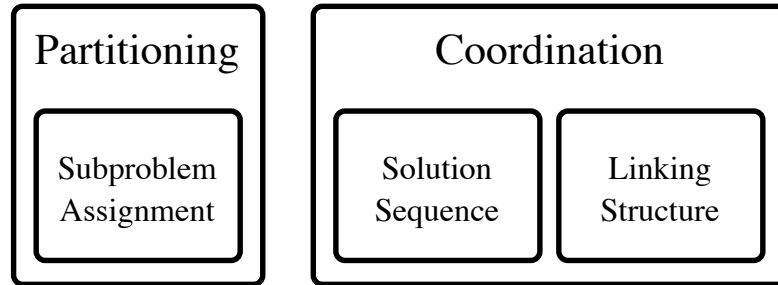


Figure 2.2 Aspects of partitioning and coordination decisions

A central result of this work is a method for automated solution of the combined partitioning and coordination decision problem. An ideal decision approach would start with a description of analysis functions and the system design problem, and determine a system partition, subproblem formulations, and a coordination algorithm that minimize the complexity and cost of the system design optimization process. Formulation decisions are difficult to encode and automate without any assumptions on the type of system design formulations to be used. This issue is addressed by assuming that a specific class of distributed optimization methods are used. This class includes penalty relaxation methods where consistency constraints are only satisfied at convergence and design constraints are satisfied at every subproblem solution, such as the ATC and ALC formulations. The system design problems are assumed to be quasiseparable, i.e., subproblems may share design variables, but not design constraints [67]. Most simulation-based design problems have a quasiseparable structure. Proofs exist for ATC [106] and ALC [141, 142] that demonstrate convergence under standard assumptions such as convexity, and show that the solution to the decomposed problem is identical to that of the original design problem. The coordination algorithm is assumed to be fixed point iteration (FPI), and therefore decomposition method convergence is subject to FPI convergence conditions [6]. Subproblem solution sequence can influence convergence rate significantly [134], and is a defining property of the coordination algorithm. The relationship between partitioning and subproblem sequence is studied first, and then a more sophisticated coordination decision model that incorporates linking structure decisions is introduced in Chapter 7.

2.3 System Representation

Analysis function interactions and dependence on design variables are important factors in partitioning and coordination decisions. Coupling variable and shared variable relationships influence both convergence of subproblems and the coordination algorithm. A technique for representing directional dependence on coupling variables and dependence on design variables compactly is required. Existence of these relationships can be illustrated using a graph, which can then be represented using an incidence or adjacency matrix. This section reviews matrix representations used in engineering design, and proposes a reduced adjacency matrix for use in making partitioning and coordination decisions.

2.3.1 Structural Matrix

Steward proposed the structural matrix (SM) for illustrating relationships in systems of equations [132]. SM rows correspond to equations, and columns to variables that appear in the equations. The system analysis constraints for the example in Section 1.4 can be used to illustrate the SM. Coupling variables and design variables are both treated as variables when constructing the SM. The system of equations to be represented is:

$$y_{31} = y_{41} = a_1(x_1, y_{12}, y_{13}) = 0.1x_1y_{12} + 0.8x_1y_{13} + b_1 \quad (2.4a)$$

$$y_{12} = y_{32} = a_2(x_2, y_{23}) = x_2y_{23} + b_2 \quad (2.4b)$$

$$y_{13} = y_{23} = a_3(x_3, y_{31}, y_{32}) = 0.1x_3y_{31} + 0.8x_3y_{32} + b_3 \quad (2.4c)$$

$$r_4 = a_4(\mathbf{x}, y_{41}) = (y_{41} - \hat{y}_{41})^2 + \|\mathbf{w} \circ \mathbf{x}\|_2^2 \quad (2.4d)$$

The response of the analysis function a_4 is not an input to any analysis function, so is not a coupling variable. The symbol r_4 is used to represent the quantity computed by a_4 . Since several of the coupling variables correspond to the same analysis function outputs, only three coupling variables are required to illustrate analysis structure in the SM:

$$\mathbf{SM} = \begin{array}{c|ccccccc} & y_{31} & y_{12} & y_{13} & r_4 & x_1 & x_2 & x_3 \\ \hline \text{Eq. (2.4a)} & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ \text{Eq. (2.4b)} & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ \text{Eq. (2.4c)} & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ \text{Eq. (2.4d)} & 1 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}$$

The SM is an incidence matrix for an undirected hypergraph¹, where variables that appear in equations are represented by hyperedges and equations by vertices. Figure 2.3

¹A hypergraph allows edges (hyperedges) to be adjacent to any number of vertices, rather than just two as in an undirected graph.

illustrates the hypergraph for the relationships in Eqs. (2.4).

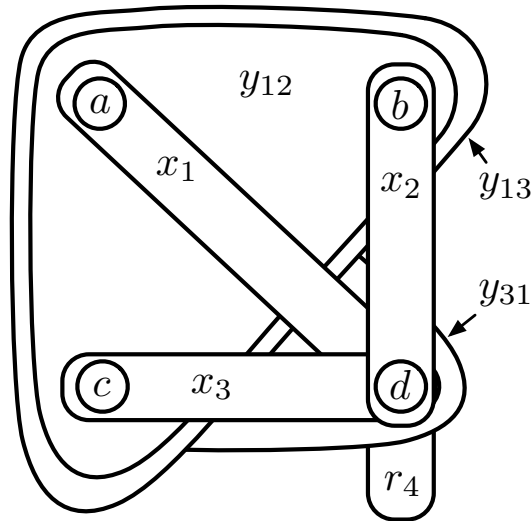


Figure 2.3 Hypergraph for relationships in Eqs. (2.4)

Suppose the four equations were partitioned into two blocks. The SM is clearly useful for assessing the number of links between blocks, which is useful for making partitioning decisions. The SM, however, provides no information about directionality, and cannot be used for determining a solution sequence. This can be resolved by specifying one variable in each equation to be an output variable, forming an output set. The SM and output set are sufficient for making combined partitioning and coordination decisions, but two different representation types must be used to provide complete information. The analysis functions in our example implicitly define an output set: $\{y_{12}, y_{13}, y_{31}, r_4\}$. In other systems of equations output variable choice may be arbitrary, particularly if every equation can be solved for any variable that appears in it. In simulation-based analysis, the output set is prescribed by the simulations.

2.3.2 Design Structure Matrix

Steward later introduced the design structure matrix (DSM) that describes the interrelationship between design elements, rather than equations and variables [133]. These design elements were originally described as either design tasks or parameters, although later DSM approaches typically limit design elements to either design tasks or parameters, but not both. The DSM is a square adjacency matrix where the elements represented by rows and columns are identical. The DSM is well suited for describing information flow direction, and has been used extensively in ordering design tasks to reduce feedback loops [27]. A DSM may be used to make combined partitioning and sequence decisions if its design elements include

both analysis functions and design variables. The DSM conveniently provides all necessary information in a single representation type. A DSM for the example in Section 1.4 can be constructed accordingly:

$$\mathbf{DSM} = \begin{array}{c|cccccccc} & a_1 & a_2 & a_3 & a_4 & x_1 & x_2 & x_3 & x_4 \\ \hline a_1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ a_2 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ a_3 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ a_4 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ x_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

The design elements in the above DSM have been grouped into analysis functions and design variables, and then ordered by index. This ordering is not necessary; the design elements may appear in an arbitrary order as long as the order of rows and columns are identical. Strictly speaking, this DSM is the transpose of the adjacency matrix for a directed graph (digraph) that shows the directional dependence of analysis functions on design variables and other analysis functions. Existence of an arc between vertices indicates a dependence relationship, and the arc direction depicts dependence direction. If a system's digraph contains a cycle, then feedback coupling exists. The digraph corresponding to this DSM is illustrated in Fig. 2.4.

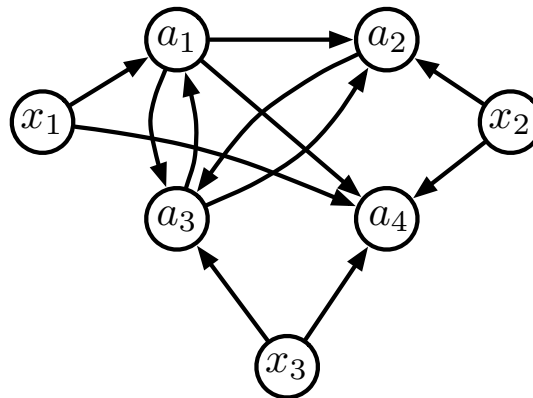


Figure 2.4 Digraph of functional relationships expressed in the DSM

2.3.3 Other Design Matrices

Another related matrix representation is the functional dependence table (FDT), introduced by Wagner [145]. The FDT is similar to the SM, but is intended specifically for partitioning

large equation-based design optimization problems. Instead of mapping variables to equations, the FDT maps design and coupling variables to objective and constraint functions. Each row represents a design function, and the columns correspond to design and coupling variables. The FDT, like the SM, can be viewed as the adjacency matrix for an undirected hypergraph [105]. It can be used to make partitioning decisions, but lacks directionality information and cannot be used for sequencing decisions without output set specification. The FDT and SM have significant similarities, but the FDT is intended specifically for aiding partitioning decisions in equation-based engineering design, while the SM applies to general systems of equations.

Other matrices associated with engineering design, but normally not used in P/C decisions, include the relation matrix (RM) and correlation matrix (CM) from quality function deployment [136], and the design matrix (DM) from axiomatic design [135]. The RM maps product engineering characteristics to customer requirements, and the CM describes correlations between engineering characteristics. The DM maps design variables to functional requirements, and is intended for evaluating independence of functional requirements when comparing design concepts.

2.3.4 Reduced Adjacency Matrix

A system of analysis functions in simulation-based design may be represented using an SM and an output set, or with a DSM. The latter is convenient because only one representation type is required. Modifying the DSM can improve its utility in computing metrics for P/C decisions. Taking note that design variables are independent quantities and their corresponding DSM rows therefore are zero, these rows can be omitted without loss of information. Design elements in the DSM are not constrained to appear in any particular order. Organizing the matrix such that analysis functions appear before design variables aids visualization of system structure. In addition, ordering functions and variables by index value is convenient for calculating metrics used in P/C decisions. This condensed and reordered matrix is termed the reduced adjacency matrix \mathbf{A} . The system adjacency matrix is $[\mathbf{A}^T, \mathbf{0}]$. The reduced adjacency matrix for the example system is:

$$\mathbf{A} = \begin{array}{c|cccccccc} & a_1 & a_2 & a_3 & a_4 & x_1 & x_2 & x_3 & x_4 \\ \hline a_1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ a_2 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ a_3 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ a_4 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}$$

The reduced adjacency matrix is designed to be consistent with other established notation.

In both the reduced adjacency matrix and the DSM, a true element in the i -th row and j -th column indicates the presence of arc $\langle j, i \rangle$. The index order for the elements of \mathbf{A} is also consistent with coupling variable index order; i.e., the existence of the coupling variable y_{ij} is signified by a true element in the i -th row and j -th column of the reduced adjacency matrix. This is consistent with the coupling variable index order set forth by Cramer et al. [35].

The difficulty of solving individual subproblems, as well as the complexity of the coordination algorithm, contribute to the overall solution expense, and must be estimated when making P/C decisions. An ideal estimate would account for the nature of analysis functions, their interactions, as well as subproblem and coordination algorithms. Producing and analyzing this information in most cases is more computationally intensive than actually solving the system design problem. More practical estimates are based on more easily obtained information. The existence of dependence relationships, expressed in the reduced adjacency matrix, are used to make such estimates. If assumptions are made about subproblem formulations and coordination algorithm, \mathbf{A} can be used to estimate the size of each subproblem and the coordination problem for a given system partition and coordination strategy. Subproblem solution difficulty typically increases with the number of analysis functions, design variables, and linking variables, although analysis function properties also influence difficulty. Similarly, coordination problem solution expense normally increases with the number of external consistency constraints [142]. Fine partitions reduce subproblem difficulty at the expense of more external consistency constraints, while coarse partitions ease coordination difficulty at the cost of more difficult subproblems. Chapters 5 and 7 will demonstrate how \mathbf{A} can be used to estimate relative contributions of subproblem and coordination difficulties to overall computational expense.

2.4 Partitioning and Coordination Decision-Making

Partitioning and coordination decisions have been treated largely as two independent tasks. Traditional decision techniques for each are based on guidelines or experience. Formal partitioning techniques have been studied thoroughly, and draw from strategies in graph partitioning. Formal coordination decision techniques have been limited to sequence decisions. Optimal sequence problems are standard operations research problems. Linking structure, the second primary component of coordination decisions, has not yet been studied thoroughly in the context of decomposition-based design optimization. Analysis of several formulations for decomposition-based design optimization has led to guidelines for selecting between formulations. A few decision methods have linked some aspect of partitioning and

coordination interaction, but coupling and tradeoffs present in the combined P/C decision problem have not been studied.

2.4.1 Traditional Decision Techniques

Subjective techniques for P/C decisions are in common use. Partitioning techniques will be discussed first, followed by coordination decision techniques. System design problems can conveniently be partitioned according to physical system divisions (object-based), or disciplinary divisions within the design organization (aspect-based) [145]. Partitions may also follow product, process, or organization divisions [29, 88]. These are typically subjective partitioning approaches based on engineering insight.

Design organizations may be aligned by discipline or object, and in some cases by both [36]. Organizational structure can have a profound impact on computational design, since design and analysis tools are developed by parts of the organization, and the organization requires tools that are congruent with its actual communication and decision structures. The automotive and aerospace industries have contributed substantially to the development of system design methods, and the organizational structure of each industry is reflected in the methods developed by them. Aerospace design is normally performed by design teams divided by discipline [16], and automotive design organizations usually possess a hierarchical structure aligned by physical subsystems [7]. Discipline-based partitions tend to have a non-hierarchical structure with feedback coupling. Object-based partitions tend to be hierarchical with unidirectional information flow.

Several authors have cited the importance of choosing the proper formulation for a system design problem, and some have proposed guidelines for selecting between them [6, 7, 8, 15, 23, 35, 74, 128]. Some of these selection guidelines are based on problem properties such as coupling strength and problem structure. For example, single-level methods, such as IDF, can handle large numbers of shared variables without increasing problem dimension. Multilevel methods, such as ATC, are a more natural fit for systems with many local variables, but relatively few shared or coupling variables. Balling and Sobieski have explained that single-level methods are best for analysis intensive problems (difficult system or subsystem analysis), while multi-level methods are best for design intensive problems (numerous local decisions to be made) [15]. Wagner identifies several categories of problem structure according to FDT representations, and offers suggestions for solution method according to which structure type a method matches best [145].

2.4.2 Formal Decision Techniques

The traditional techniques outlined above have been useful tools for the sometimes very complicated task of determining a system partition and coordination strategy. These techniques, however, are subjective, and success depends on designer experience and insight. Certain innovative (and possibly superior) solution approaches may be overlooked. The partitioning and sequence decision problems can be solved as optimization problems themselves, and may be viewed as preprocessing for system optimization. Formal optimization-based techniques may reveal insights about the system. Formal partitioning and subproblem sequence methods have been studied for use in decomposition-based optimization. This section reviews important partitioning developments, optimal sequencing techniques, and a few approaches that consider some aspect of both partitioning and sequencing.

When a system is represented using a graph, the system partitioning problem can be solved using graph partitioning techniques. Michelena and Papalambros demonstrated the use of spectral [105] and network reliability methods [103] to obtain partitions that minimize external linking variables and exactly balance subproblem sizes. Krishnamachari and Papalambros used linear integer programming to generate partitions that allow some subproblem size imbalance [84]. Chen, Ding, and Li introduced an iterative two-phase approach where the FDT is first ordered so that coupling relationships are banded along the diagonal, and then independent variable blocks and a system-wide linking variable block are formed [30]. Drăgan proposed a partitioning algorithm also based on the FDT, as well as one possible coordination strategy for managing the links between subproblems [47].

Some of the above approaches are decidedly efficient, but require numerous approximations and assumptions. For example, spectral methods can produce a partition in fractions of second, but require that the FDT hypergraph is approximated with a graph. A linear integer programming approach is also computationally efficient, but requires significant assumptions to achieve linearity. These methods also do not account for the dimension of subproblems associated with partition blocks; only the number of analysis function in each block. Another shortfall is that these methods do not account for directionality in functional dependence. The consistency constraints for shared and coupling variables are handled differently in many formulations, and coupling variable consistency constraints are formulated according to functional dependence direction. Omitting directionality from a decision model therefore reduces accuracy.

A large system design problem can take days or weeks to solve. More accurate assumptions in P/C decision optimization may increase the preprocessing cost to minutes or even a few hours, but the improved P/C decisions stand to decrease system optimization time and increase reliability of results by a much greater factor; the benefit of added accuracy in P/C

decisions may be well worth the cost for decomposition-based design optimization.

Sequencing and scheduling problems are standard topics in operations research [22]. Techniques developed for them have been applied to problems in decomposition-based design. A common approach is to order subproblems or design tasks such that feedback is minimized. Steward used the DSM with a ‘tearing’ algorithm to order design elements so that blocks of closely coupled tasks can be identified, forming a type of partition. In this approach partitions depend on sequence decisions; partitioning decisions cannot be made independently, and superior P/C decisions may be overlooked. Rogers introduced DeMAID, a heuristic DSM-based software tool for sequencing design tasks [115], and later DeMAID/GA, which utilized a genetic algorithm [73] to perform sequencing tasks [116]. Browning provided a review of methods based on the DSM [27]. Kroo suggested that, after sequencing has been used to minimize feedback loops, consistency constraints could be used to break any remaining feedback loops [85]. Meier, Yassine, and Browning reviewed DSM-based sequencing approaches and compared their objective functions, which primarily involved some combination of minimizing feedback, improving concurrency and modularity, or reducing computational expense [99]. The above sequencing approaches are primarily based on the DSM or a similar system representation which models the existence of functional dependence but does not express the nature of the dependence. A weighted DSM may be used, and Rogers and Bloebaum developed methods for sequencing design tasks that depend on coupling strength metrics, which are based on sensitivities between design tasks [114]. Alyaqout, Papalambros, and Ulsoy defined a new measure of coupling strength that accounts for optimization algorithm considerations [10]. This metric could also be used as a factor in P/C decisions. Most previous approaches for sequencing in design are based only on interactions between analysis functions (i.e., coupling variables). If the system representation does not include design variables, and if the decision approach does not otherwise account for them, sequence decisions are based on incomplete information.

A few approaches have accounted for some aspect of P/C interaction. Kusiak and Wang demonstrated a method that first partitions a system based on its FDT, and then identifies an efficient subproblem sequence using a precedence matrix [89]. This is similar to Steward’s SM approach, except that Steward first determined a sequence and then identified a partition. Meier et al. also described how partitions can be identified after a sequence is defined [99]. A sequential P/C decision process, however, cannot account for all P/C decision interactions. It will be shown that sequential or independent approaches can fail to identify Pareto-optimal P/C options, while a simultaneous approach does not. Altus, Kroo, and Gage developed a genetic algorithm that simultaneously determined function sequence as well as ‘breaks’ between functions that form a partition [9]. Only a single result was presented

with a prescribed number of subproblems, P/C decision tradeoffs were not studied, and subproblem order was not defined since parallel subproblem solution was assumed.

Table 2.1 summarizes the partitioning and coordination decision methods reviewed above. Early formal techniques for partitioning and coordination decisions emphasized precedence relationships between analysis functions or design tasks. These precedence relationships dominated the partitioning problem, and were in effect constraints on partitions that could be made. Many approaches based on the DSM, such as Steward's tearing algorithm, determine solution sequence first, and then apply heuristic rules or algorithms to derive an associated partition. Most approaches using the DSM are applied to engineering design process scheduling, rather than design optimization, and usually do not account for design variables. Later work, including the methods by Michelena and Papalambros, abandoned the use of precedence information in partitioning decisions and focused instead on function dependence on design variables (represented using the FDT without an output set). This adds degrees of freedom to the partitioning problem, but disregards information that is important both to partitioning and coordination decisions. The work introduced here brings precedence information back into the decision process. In addition to addressing precedence relationships among analysis functions, the decision methods presented here also deal with precedence among subproblems in a coordination strategy. The reduced adjacency matrix was developed as a compact matrix representation that balances precedence information used in early methods with dependence on design variables utilized by more recent methods. This more complete set of information is used to evaluate the optimization problem dimension for each subproblem, something that was only approximated by earlier methods.

The following chapters will describe a technique for optimizing partitioning and coordination decisions based on information contained in the reduced adjacency matrix for the problem at hand. This technique accounts for the coupling present between partitioning and coordination decisions. Although decisions are based on more complete models than have been used in the past, these models are still approximations for computational expense. Therefore the results presented are optimal with respect to the P/C decision model, and are approximately optimal with respect to computational expense.

Recall that coordination decisions involve both subproblem sequence and consistency constraint management; the latter issue has not yet been thoroughly investigated. Different distributed design optimization formulations provide varying levels of flexibility in how consistency constraints may be allocated in a decomposition method (i.e., linking structure). For example, CO completely prescribes allocation, while ATC allows consistency constraints for linking variables between subproblems to be assigned to any subproblem that is a

Table 2.1 Summary of formal partitioning and coordination decision methods

	Method	Representation	Application	Features
Steward 1965 [132]	tearing	SM, output set	sys. of eqns.	sequence then partition
Steward 1981 [133]	tearing	DSM	design process	heuristic algorithm
Rogers 1989 [115]	DeMAID	DSM	design process	commercially available
Kusiak and Wang 1993 [89]	decomposition	FDT, DSM	design process	partition then sequence
Rogers and Bloebaum 1994 [114]	subprob. sequence	DSM	design opt.	coupling strength based
Michelena and Papalambros 1995 [103]	network	FDT	design opt.	reliability principles
Rogers 1996 [116]	DeMAID/GA	DSM	design process	genetic algorithm
Altus et al. 1996 [9]	GA	DSM	design opt.	scheduling and decomp.
Michelena and Papalambros 1997 [105]	spectral	FDT	design opt.	graph theory
Krishnamachari and Papalambros 1997 [84]	LIP	FDT	design opt.	enables imbalance
Drăgan 2002 [47]	decomposition	FDT	sys. of eqns.	for parallel proc.
Chen et al. 2005	two-phase	FDT	design opt	independent blocks

common ancestor. ALC offers complete flexibility in consistency constraint allocation, an attractive feature for studying the effect of consistency constraint allocation decisions. Material presented through Chapter 6 only addresses the subproblem sequence aspect of coordination decisions, while a more sophisticated approach that considers consistency constraints only. Chapter 7 develops the theory required to understand the linking structure aspect of the coordination decision problem and demonstrates how to incorporate linking structure into partitioning and coordination decisions.

2.5 Summary

This chapter introduced important concepts from decomposition-based design optimization, which involves partitioning a system design problem into smaller subproblems. System optimization methods applied to a decomposed system must ensure that subproblem solutions are consistent, and that the resulting design is optimal for the overall system. Coordination strategies are used to guide repeated subproblem solutions toward a consistent and optimal state. Subproblem solution sequence and linking structure are aspects of a coordination strategy. A system partition and coordination strategy must be defined before solving a system design optimization problem. Qualitative techniques have been used to guide partitioning and coordination decisions; formal optimization methods offer an alternative decision approach, and utilize system structure representations, such as the DSM or FDT. The partitioning problem and the coordination decision problem have been formulated and solved as independent optimization problems, but have not yet been solved together. Some efforts have involved sequential approaches that account for some, but not all of the coupling between partitioning and coordination decisions. This dissertation demonstrates that partitioning and coordination decisions are coupled, and presents an automated decision technique that accounts fully for this coupling. The reduced adjacency matrix introduced in this chapter is a new system representation suitable for making both partitioning and coordination decisions. The system design problems considered in this dissertation are assumed to be simulation-based with quasiseparable structure. The next chapter introduces several system design problems used to demonstrate concepts throughout this dissertation.

Chapter 3

Demonstration Examples

Several engineering design optimization problems have been developed to demonstrate concepts and techniques put forth in this dissertation. All examples are suitable for solving with decomposition-based design optimization. Each example has specific properties that are useful for investigating important aspects of system design, and in most cases partitioning and coordination decisions. A detailed description of the analysis model and design problem for each example is presented here. Enough detail is provided for most examples to enable replication. The first example is an air flow sensor design problem with feedback coupling between two simple disciplinary analyses. Next, a turbine blade design problem is presented with somewhat more sophisticated disciplinary analyses. An approach to designing a product family for a fleet of aircraft is then described, which is the only design problem included here without sufficient information for problem replication due to reliance on commercial software. The fourth example introduces a generalized approach to truss design which can be applied to trusses of any size and a large variety of topologies. The last example presented in this chapter involves the design of an automotive electric water pump. An electric vehicle design case study is presented later in Chapter 8.

3.1 Air Flow Sensor Design

Vane airflow (VAF) sensors are used in automotive applications to monitor the rate at which air enters the engine for use in fuel injection control. A VAF sensor design problem was developed to investigate the effect of coupling between disciplinary analyses. It will be used in Chapter 4 to illustrate two different formulations for decomposition-based design optimization. The design problem incorporates structural and aerodynamic analyses, and aims at specifying a sensor that produces a desired result to airflow. The operation of a VAF sensor is described, followed by a simplified analysis model for the sensor.

A VAF sensor is illustrated in Fig. 3.1 [34]. Incoming air flows past the stator flap, which deflects in proportion to air flow velocity. A bypass channel reduces the sensor's

impedance on airflow. A potentiometer measures this deflection angle and provides a signal to the engine control unit.

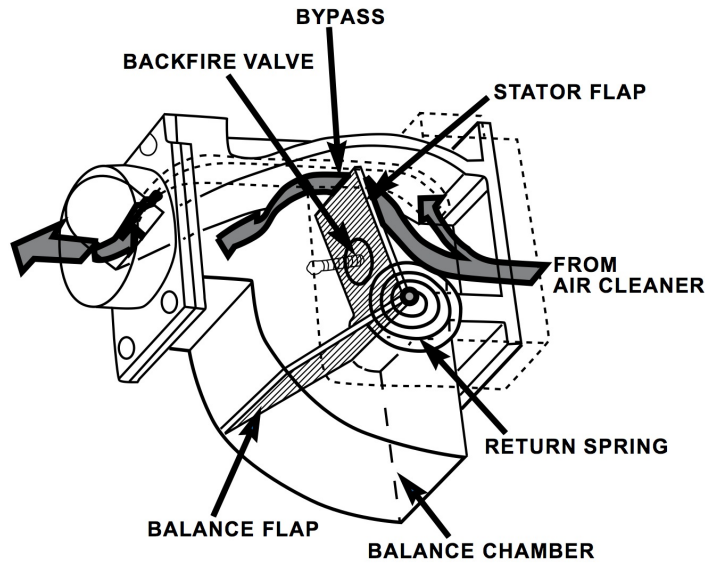


Figure 3.1 Vane airflow sensor schematic (after [34])

A simplified model of a VAF sensor is used in this design example (Fig. 3.2). The stator flap has length ℓ and width w , is attached to its base with a revolute joint, and is biased to the vertical position with a torsional spring of stiffness k . The plate is subject to horizontal air flow of speed v that results in a drag force F . The design objective is to choose ℓ and w such that the plate deflects an amount θ (for a fixed air speed) that closely matches a target deflection value $\hat{\theta}$. The plate area $A = \ell w$ is constrained to a fixed value, and the drag force on the plate must not exceed F_{max} . This task, summarized in Eq. (4.6), is in essence a sensor calibration problem.

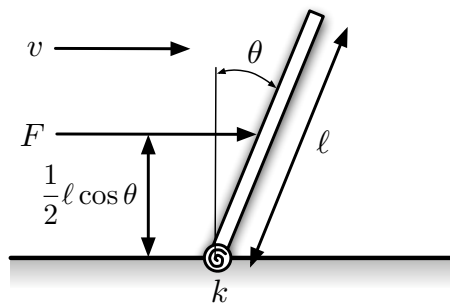


Figure 3.2 Simplified representation of a vane airflow sensor

$$\begin{aligned}
& \min_{\ell, w} && (\theta - \hat{\theta})^2 && (3.1) \\
& \text{subject to} && F - F_{\max} \leq 0 \\
& && \ell w - A = 0
\end{aligned}$$

The structural analysis computes the plate deflection θ for a given sensor design and drag force. Note that the governing equation cannot be solved directly for θ , requiring iterative solution.

$$k\theta = \frac{1}{2}F\ell \cos \theta \quad (3.2)$$

The aerodynamic analysis computes the drag force on the plate F for a given sensor design and plate deflection; C is a constant that incorporates air density and the drag coefficient, $C = \frac{1}{2}\rho C_D$, and A_f is the plate frontal area, $A_f = \ell w \cos \theta$.

$$F = CA_f v^2 = C\ell w \cos \theta v^2 \quad (3.3)$$

The analyses depend on each other—Fig. 3.3 illustrates this relationship. The coupling variables are $\tilde{\theta}$ and \tilde{F} . This notation is used to distinguish coupling variables from the corresponding analysis functions, $\theta(\ell, \tilde{F})$ and $F(\ell, w, \tilde{\theta})$. The shared variable is ℓ ($\mathbf{x}_{s1} = \mathbf{x}_{s2} = \ell$), and w is a local variable ($\mathbf{x}_{l1} = w$). Fixed point iteration can be used to find consistent values of \tilde{F} and $\tilde{\theta}$ for a given design ($\mathbf{x} = [\ell, w]$).

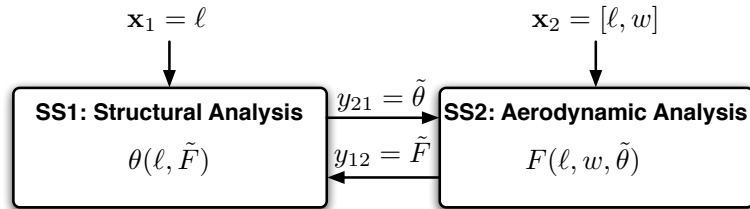


Figure 3.3 Coupling relationship in airflow sensor analysis

The optimal solution to this problem may be found using monotonicity analysis (MA) [110], and can be used to benchmark computational results. In MA active constraints can be identified by analyzing functions that vary monotonically with variables. The problem can then be reduced by making substitutions based on constraint activity. Sufficiently large deflection targets $\hat{\theta}$ in the VAF sensor problem will require a drag force that violates the maximum force constraint. There is a monotonic tradeoff between minimizing target deviation and minimizing drag force, and we can see that the minimum feasible target

deviation will correspond to the maximum drag force allowed by the inequality constraint. Thus, the drag force constraint is active when the target deflection is large, and we can substitute F_{\max} into Eq. (3.2) to solve for the optimal deflection θ^* :

$$\theta^* = \cos^{-1} \left(\frac{F_{\max}}{CA_f v^2} \right) \quad (3.4)$$

The aerodynamics equation and the area constraint can then be used to solve for the optimal length and width, respectively:

$$\ell^* = \frac{2k \cos^{-1} \left(\frac{F_{\max}}{CA_f v^2} \right) CA_f v^2}{F_{\max}^2}, \quad w^* = A/\ell^* \quad (3.5)$$

For parameter values $k = 0.050$ N/rad, $v = 40.0$ m/s, $C = 1.00$ kg/m³, $F_{\max} = 7.00$ N, and $\hat{\theta} = 0.250$ rad, the force constraint is active, and the optimal design is $[\ell^*, w^*] = [0.0365, 0.274]$. The drag coefficient of a finite flat plate is approximately 2.0, resulting in a value of $C = 1.00$ if we assume air density to be 1.00 kg/m³.

The solution of the design problem in Eq. (4.6) requires a nested solution approach. For every design proposed by the optimization algorithm, a fixed point iteration solution to Eqs. (3.2) and (3.3) to obtain consistent values for $\tilde{\theta}$ and \tilde{F} is required. The solution was obtained using a sequential quadratic programming algorithm [69, 111], and the result matches the MA solution.

3.2 Turbine Blade Design

The analysis and design of a turbine blade for a gas turbine engine is presented here. Emphasis is placed on the coupling between structural and thermal analysis, and this design example will be used to study the effects of analysis coupling between disciplinary analyses, as was the case with the air flow sensor design problem. This design example was introduced in [5] and appeared in [6]. Turbine blade design has been the ongoing subject of MDO studies [117]. The turbine blade model presented here is simplified enough to allow straightforward replication, yet still captures important interactions and tradeoffs. The model allows for easy adjustment of coupling strength, a feature required for a comparison of single-level methods presented in Section 4.2.

A turbine blade in a gas turbine engine is exposed to high temperature combustion gasses moving at high velocity, and is subject to high forces due to aerodynamic drag force and centripetal acceleration. Figure 3.4 illustrates turbine blades from a GE J-79 turbojet engine [57]. Each blade is attached to the rotor at the left of the figure, and combustion gasses

moving from the left cause the turbine to rotate.

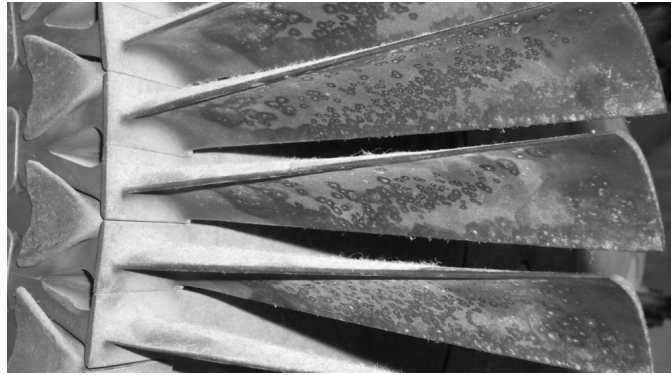


Figure 3.4 GE J-79 turbojet engine turbine blades [57]

Several phenomena were modeled in order to capture the design tradeoffs and coupling behavior, specifically: thermal expansion of the turbine blade in the axial direction, stress and elongation due to centripetal acceleration, aerodynamic drag force and the resulting bending stresses, and the temperature dependence of thermal conductivity, elastic modulus, and rupture stress.

The blade temperature profile depends upon its dilated length. Elongation due to thermal expansion or centripetal forces exposes more surface area to hot combustion gasses, affecting the heat transfer through the blade and the associated temperature profile. The model also captures the dependence of elastic modulus and thermal conductivity on temperature. Higher temperatures (caused by changes in length) result in lower stiffness, causing greater elongation. In summary, temperature depends on length, and length depends on temperature. Thus, turbine blade analysis consists of two coupled disciplinary analyses, similar to the previous example. The design task is to minimize the blade mass m and the heat transfer through the blade q . Both of these metrics influence turbine thermal efficiency.

3.2.1 Analysis Model

The turbine blade is modeled as a simple rectangular fin (Fig. 3.5). The design variables are the blade width w and thickness t . The blade has an initial undeformed length of L_0 , and is subjected to combustion gas at temperature T_g and velocity v_g . The blade is affixed to a rotor with angular velocity ω , resulting in an inertial force f_{ac} . The axial position x is measured from the blade base. Four failure modes are considered: melting, interference between the blade and the turbine housing due to elongation, and structural failure due to bending stress σ_b or axial stress σ_a . Several simplifying assumptions were made: constant coefficient of thermal expansion α , no internal blade cooling, constant inertial force f_{ac} over the blade,

and no lateral contraction. The dependence of thermal conductivity (k), elastic modulus (E), and rupture stress (σ_r) on temperature is modeled with curve fits based on empirical data.

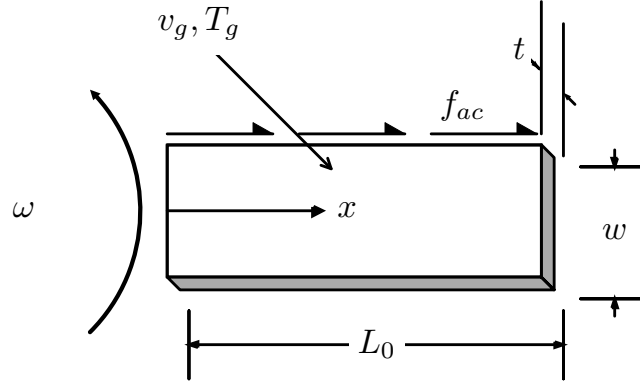


Figure 3.5 Turbine blade model schematic

The turbine blade optimization problem is presented in Eq. (3.6), which has been formulated as a single objective problem by creating a mass constraint. The coupling variables, $\tilde{T}(x)$ and \tilde{L} , are required to be consistent with the corresponding analysis functions, $T(w, t, \tilde{L}, x)$ and $L(\tilde{T}(x))$, at the solution.

$$\begin{aligned}
 & \min_{w, t} && q(w, t, \tilde{L}) && (3.6) \\
 & \text{subject to} && T(w, t, \tilde{L}, x) - T_{\text{melt}} \leq 0 \\
 & && \delta_{\text{total}}(\tilde{T}(x)) - \delta_{\text{allow}} \leq 0 \\
 & && \sigma_a(\tilde{L}, x) - \sigma_r(\tilde{T}(x), x) \leq 0 \\
 & && \sigma_b(t, \tilde{L}, x) - \sigma_r(\tilde{T}(x), x) \leq 0 \\
 & && m(w, t) - m_{\text{max}} \leq 0 \\
 & && \text{and } 0 \leq x \leq L_0 + \delta_{\text{total}}(\tilde{T}(x)).
 \end{aligned}$$

T_{melt} is the melting temperature, $\delta_{\text{total}}(\tilde{T}(x))$ is the blade elongation, δ_{allow} is the initial clearance between the blade and housing, and $\sigma_a(\tilde{L}, x)$, $\sigma_b(t, \tilde{L}, x)$, and $\sigma_r(\tilde{T}(x), x)$ are the axial, bending, and rupture stress distributions along the blade. The analysis for each discipline (structural and thermal) follows.

Structural Analysis

The structural analysis calculates blade mass ($m = wtL_0\rho$), where ρ is the blade density, the total blade elongation (δ_{total}), which is the sum of the thermal expansion δ_{th} and elongation due to axial acceleration δ_{ax} , and bending and axial stress distributions ($\sigma_b(x)$, $\sigma_a(x)$). We

begin with the elongation calculation. The first elongation term is calculated as follows:

$$\begin{aligned}
 d\delta_{th} &= \alpha(T(x) - T_0)dx & (3.7) \\
 \delta_{th} &= \int_0^{L_0} T(x)dx - \int_0^{L_0} \alpha T_0 dx \\
 \delta_{th} &= \int_0^{L_0} T(x)dx - \alpha T_0 L_0
 \end{aligned}$$

T_0 is the initial blade temperature, and α , the coefficient of thermal expansion, is assumed constant. The temperature profile, calculated by the thermal analysis, is required to evaluate δ_{th} . To calculate δ_{ax} , the axial load as a function of axial position is determined. The portion of the blade outboard of a position x pulls with load $P_a(x)$. The tangential velocity of the blade $v = \omega r$ is assumed to be constant over the blade length, and is valid if $L_0 \ll r$.

$$\begin{aligned}
 P_a(x) &= \int_x^{L_0 + \delta_{total}} \frac{v^2}{r} \rho A_c dx & (3.8) \\
 &= \frac{v^2}{r} \rho wt (L_0 + \delta_{total} - x) \\
 &= \omega^2 r \rho wt (L_0 + \delta_{total} - x)
 \end{aligned}$$

$$\begin{aligned}
 \delta_{ax} &= \int_0^{L_0 + \delta_{total}} \frac{P_a(x) dx}{A_c E(T(x))} & (3.9) \\
 &= \omega^2 r \rho \int_0^{L_0 + \delta_{total}} \frac{(L_0 + \delta_{total} - x)}{E(T(x))} dx
 \end{aligned}$$

$$\begin{aligned}
 \delta_{total} &= \int_0^{L_0} T(x) dx - \alpha T_0 L_0 & (3.10) \\
 &\quad + \omega^2 r \rho \int_0^{L_0 + \delta_{total}} \frac{(L_0 + \delta_{total} - x)}{E(T(x))} dx
 \end{aligned}$$

Since Eq. (3.10) is transcendental, an iterative solution procedure is required to solve for δ_{total} given $T(x)$.

The axial stress is a function of axial position, and is calculated with the relation $\sigma_a = P_a/A_c$, where P is the axial load, $A_c = wt$ is the cross sectional area as before, and $L = L_0 + \delta_{total}$ is the elongated length.

$$\sigma_a(L, x) = \omega^2 r \rho (L - x) \quad (3.11)$$

The aerodynamic load is calculated using $P_{\text{aero}} = \frac{1}{2}A_f C_D \rho v^2$, where $A_f = wL$ is the frontal area, C_D is the drag coefficient, ρ is the combustion gas density, and v is the combustion gas velocity (assumed perpendicular to the blade). For convenience the constant $K = \frac{1}{2}C_D \rho v^2$ is defined, giving $P_{\text{aero}} = KwL$. The total drag force acting on the blade outboard of a position x is $P_{\text{aero}}(x) = Kw(L-x)$, and the bending moment at point x is $M(x) = Kw \frac{(L-x)^2}{2}$, resulting in a bending stress of:

$$\sigma_b(w, L, x) = \frac{3K(L-x)^2}{4t^2} \quad (3.12)$$

Thermal Analysis

The thermal model, which calculates the temperature profile and heat transfer, was derived from the steady-state heat equation using constant base temperature and an adiabatic tip boundary condition [75]. The average convection coefficient \bar{h} was approximated using empirical correlations involving the average Nusselt number \bar{N}_u and the Prandtl number Pr : $\bar{N}_u = \frac{\bar{h}w}{k_g} = CRe_D^z Pr^{1/3}$. The combustion gas conduction coefficient is k_g , $Re_D = vw/v$ is the appropriate Reynold's number, z is an empirical exponent of 0.731, and C is the heat capacity of the combustion gas. Solving for \bar{h} , and substituting values for the other parameters with SI units (at $T_\infty = 900^\circ C$), we find: $\bar{h}(v, w) = 9.196v^{0.731}w^{-.269}$. The temperature profile and the heat transfer through the blade into the rotor at the point of attachment are found through solution of the heat equation with the appropriate boundary conditions:

$$T(w, t, L, x) = \frac{\cosh(s(L-x))}{\cosh(sL)}(T_b - T_\infty) + T_\infty \quad (3.13)$$

$$q(w, t, L) = wt(T_b - T_\infty) \tanh(sL) \sqrt{2\bar{h}(w+t)wtk} \quad (3.14)$$

where $s = \sqrt{2\bar{h}(t+w)/ktw}$.

Surrogate Models

Surrogate models based on empirical data [97] were employed in order to capture temperature dependence. The rupture stress σ_r for Inconel X-750 was approximated using a modified sigmoid function.

$$\sigma_r(T) = \frac{1300}{1 + e^{0.011(T-675)}} \quad (3.15)$$

The conductivity of the blade k was modeled using a linear fit. The dependence on

average temperature \bar{T} was captured from empirical data.

$$k(\bar{T}) = 6.8024 + 0.0172\bar{T} \quad (3.16)$$

A fourth-order polynomial was fit to the modulus of elasticity for the blade material.

$$E(T) = 209.8 - 0.0487T - 0.0002T^2 + 6 \cdot 10^{-7}T^3 - 6 \cdot 10^{-10}T^4 \quad (3.17)$$

3.2.2 System Analysis

Figure 3.6 illustrates the analysis problem structure. The system has two shared design variables, and no local design variables.

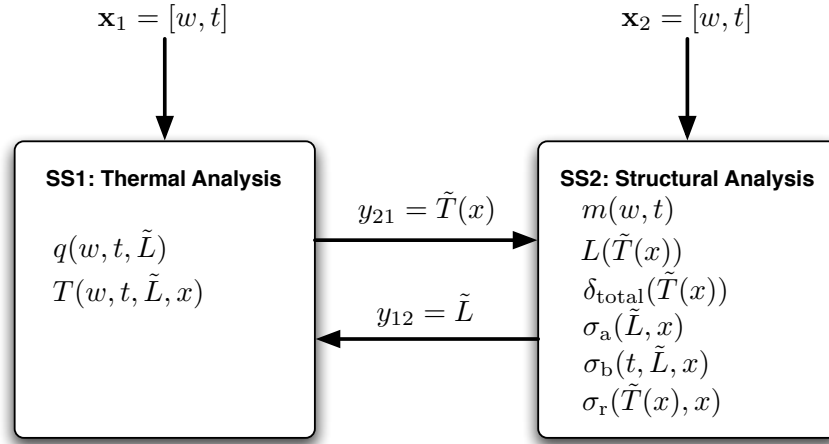


Figure 3.6 Turbine blade coupling and functional relationships

The analysis functions evaluated by the thermal analysis are the heat loss $q(w, t, \tilde{L})$ and the temperature distribution $T(w, t, \tilde{L}, x)$. The structural analysis evaluates several analysis functions, including the mass $m(w, t)$, dilated length $L(\tilde{T}(x))$, total deflection $\delta_{\text{total}}(\tilde{T}(x))$, and the bending, axial, and rupture stress distributions $\sigma_b(t, \tilde{L}, x)$, $\sigma_a(\tilde{L}, x)$, and $\sigma_r(\tilde{T}(x), x)$. Both design variables are shared, i.e., $\mathbf{x}_{s1} = \mathbf{x}_{s2} = [w, t]$. The function valued quantities (temperature and stress distributions) are discretized along the length of the blade to facilitate numerical calculations. Using the parameter values from Table 3.1 and a sample design of $[w, t] = [0.08, 0.005]$ (meters), the analysis outputs (using FPI) are $q = 0.2046$ W, $m = 0.1702$ kg, and $L = 0.057$ m.

The design problem presented in Eq. (3.6) was solved using two different system optimization approaches. The results of a parametric study on these solutions is presented in Section 4.1. Note that the third and fourth constraints in Eq. (3.6) are function-valued; these were discretized and implemented as vector-valued constraints. The mass was constrained

Table 3.1 Turbine blade design parameters

ρ	8510 kg/m ³	ρ_g	3.522 kg/m ³
L_0	0.05 m	C_d	2.0
α	$12.6 \cdot 10^{-6}$ m/K	v	100 m/s
r_b	0.5 m	T_b	300 ° C
ω	2100 rad/s	T_g	900 ° C
δ_{max}	0.05 m	ε	$1.0 \cdot 10^{-8}$

not to exceed 0.04 kg. The parameter values from Table 3.1 were used, and the optimal design was found to be $[w^*, t^*] = [0.0131, 0.0075]$ (both in meters).

3.3 Aircraft Family Design

A product family design problem was developed that addresses how to design commercial aircraft in an airline fleet that share some common components. This example will be used to illustrate multi-level formulations for decomposition-based design optimization in Chapter 4, and will be the subject of a parametric study on coordination algorithm parameters.

A product family is a set of individual products that share common components or subsystems and address a set of related market applications [101]. The motivation for component sharing is cost reduction in both development and manufacturing. In an aerospace context, a product family is usually comprised of a baseline aircraft and its derivatives or variants, but can also involve two or more aircraft with dissimilar missions that share only a few key parts or systems. The product family design problem presented here takes the latter approach, and was originally presented in [8]. It will be used to illustrate formulations for decomposition-based design optimization and for parametric studies on coordination algorithms. This section reviews product family design, discusses the analysis tools used, and presents the aircraft family design problem.

3.3.1 Product Families in Aircraft Design

As the aerospace industry has matured, emphasis has shifted from performance enhancement to cost reduction, efficiency, and quality improvement. An avenue for cost savings is improved manufacturing efficiency. Sharing major structural components can reduce tooling and assembly costs. A product family approach can also reduce operational costs, such as maintenance, or pilot cross-training programs if avionics systems are common across part of a fleet. Cost reductions realized through commonality typically come at the expense of a performance penalty [50, 51, 52, 53]. Common components cannot be optimized for

individual aircraft.

The objective of this product family study is to quantify the benefits of a product family and define the preliminary design of its members. It is critical that the optimization objective function helps quantify the tradeoffs present in product family design problems. Life cycle cost is an ideal objective function, but is unnecessarily complex since an accurate prediction of total cost is not required. An approximate cost model that incorporates both acquisition and costs was chosen as the objective function. The acquisition cost model, based on [96], is split into manufacturing and development costs. A manufacturing learning curve is applied such that cost decreases with the number of units produced. Development cost is non-recurring and is averaged over the total number of aircraft produced. The non-recurring cost for parts already developed for another aircraft in the family is significantly lower than for a new part. Fuel cost is based on the Breguet range equation [112]. Each type of aircraft in the product family is assumed to fly only one specific mission. The acquisition and fuel cost models are used to estimate ticket prices for each aircraft in the context of fleet operation. Important tradeoffs associated with commonality are effectively captured by the ticket price estimates.

3.3.2 Aircraft Performance Analysis

Aircraft performance is evaluated using the Program for Aircraft Synthesis Studies (PASS), an aircraft conceptual design tool [87]. A detailed description of this analysis is beyond the scope of this dissertation, and is presented in [86]. In addition to quantities computed by PASS, the aircraft family problem requires calculation of wing stresses. The wing model uses wing geometry design variables, such as wing sweep (Λ) and main wing aspect ratio (AR_{mw}), as well as aircraft takeoff weight W_{TO} , as inputs to a simple wing-box model where wing skin carried the bending load. If wing skin thickness along the wing span is designed such that the wing is fully stressed, the skin thickness is approximately quadratic along the span. This observation allows us to parameterize skin thickness along the entire main wing span using only three thickness values: T_1 (thickness at wing root), T_2 (thickness at 33% span), and T_3 (thickness at 67% span). The main wing is the only common component in this product family study: wing tip extensions can be unique for each aircraft. Wing weight is also influenced by the minimum gauge of available material and control surfaces and high-lift systems. Equation (3.18) accounts for these factors as well as minimum bending thickness was fit to data from existing aircraft.

$$W_{wing} = 1.35(W_{str} - W_{min}) + 4.9S_{wing}. \quad (3.18)$$

where W_{str} is the weight of material needed to resist bending, W_{min} is the weight of minimum gauge material, and S_{wing} is the wing area.

3.3.3 Aircraft Family Problem Formulation

The product family considered here is limited to two aircraft types, A and B , designed to fulfill missions 1 and 2, respectively. Mission 1 requires a range of 3400 nautical miles (nmi) and an aircraft capacity of 296 passengers. Mission 2 requires a range of 8200 nmi and an aircraft capacity of 259 passengers. Forecasts suggest a market need for 800 type A aircraft, and a need for 400 type B aircraft. In addition to mission requirements, constraints on other performance metrics, such as balanced field length and second segment climb, are included.

The aircraft family design problem is to minimize a composite cost measure for the fleet, subject to mission and performance constraints, as well as compatibility of common parts. Each aircraft type has 16 design variables ($x_{1i} \dots x_{16i}$, $i \in \{A, B\}$), which are described in Table 3.2.

Main wing commonality requires that the variables $x_{10i} \dots x_{16i}$ are equal for each aircraft. This requirement can be met by treating them as shared variables:

$$\mathbf{x}_s = [x_{10A}, \dots, x_{16A}] = [x_{10B}, \dots, x_{16B}]. \quad (3.19)$$

The local variables for aircraft A and B are

$$\mathbf{x}_{\ell A} = [x_{1A}, \dots, x_{9A}] \quad \text{and} \quad \mathbf{x}_{\ell B} = [x_{1B}, \dots, x_{9B}].$$

The complete set of design variables for the product family design problem is

$$\mathbf{x} = [\mathbf{x}_{\ell A}, \mathbf{x}_{\ell B}, \mathbf{x}_s].$$

Each aircraft must comply with a set of five performance constraints, whose numeric values are specific to the mission each aircraft is designed to fly (see Table 3.3).

The fleet composite cost metric is given in Eq. (3.20), and is the aircraft family design objective function. It is based on estimated ticket prices and the proportion of aircraft types in the fleet. The estimated ticket prices for aircrafts A and B are p_A and p_B , respectively, and the number of aircraft A and B in the fleet are n_A and n_B , respectively.

$$f(\mathbf{x}) = \frac{n_A}{n_A + n_B} p_A(\mathbf{x}_{\ell A}, \mathbf{x}_s) + \frac{n_B}{n_A + n_B} p_B(\mathbf{x}_{\ell B}, \mathbf{x}_s). \quad (3.20)$$

The elements of the aircraft family design problem have all been defined now. The

Table 3.2 Design variables for the aircraft family design problem

Variable	Name	Description	Aircraft A		Aircraft B	
			Variable Bounds	Variable Bounds	Variable Bounds	Variable Bounds
x_{1i}	W_{TO}	takeoff weight	300,000 - 450,000 lbs	450,000 lbs	450,000 - 600,000 lbs	
x_{2i}	$thrust$	sea level static thrust	50,000 - 70,000 lbs		75,000 - 105,000 lbs	
x_{3i}	X_{wing}	location of wing root leading edge	0.20 - 0.40		0.20 - 0.40	
x_{4i}	S_h/S_{ref}	nondimensional horizontal tail area	0.20 - 0.35		0.20 - 0.35	
x_{5i}	Alt_I	initial cruise altitude	32,000 - 45,000 ft		32,000 - 45,000 ft	
x_{6i}	Alt_F	final cruise altitude	32,000 - 45,000 ft		32,000 - 45,000 ft	
x_{7i}	$Mach$	Mach number at start of cruise	0.75 - 0.92		0.75 - 0.92	
x_{8i}	$flap_{TO}$	takeoff flap deflection	0.0 - 15.0		0.0 - 15.0	
x_{9i}	S_{wt}	wing tip extension area	0 - 125 ft^2		0 - 125 ft^2	
x_{10i}	S_{wm}	main wing area	2000 - 4000 ft^2		2000 - 4000 ft^2	
x_{11i}	AR_{wm}	main wing aspect ratio	7.0 - 12.0		7.0 - 12.0	
x_{12i}	(t/c)	thickness to chord ratio	0.80 - 0.14		0.80 - 0.14	
x_{13i}	Λ	wing sweep	20.0 - 35.0		20.0 - 35.0	
x_{14i}	T_1	skin thickness at root of main wing	0.06 - 2.5		0.06 - 2.5	
x_{15i}	T_2	skin thickness at 33% span of main wing	0.06 - 2.0		0.06 - 2.0	
x_{16i}	T_3	skin thickness at 67% span of main wing	0.06 - 1.5		0.06 - 1.5	

Table 3.3 Design constraints for the aircraft family design problem

Constraint	Name	Description	Aircraft A	Aircraft B
g_1	Range	min range	3,400 nmi	8,200 nmi
g_2	<i>TOFL</i>	max takeoff field length	7,000 ft	10,000 ft
g_3	<i>LFL</i>	max landing field length	5,200 ft	6,000 ft
g_4	γ_2	min 2 nd seg. climb gradient	0.024	0.024
g_5	<i>ST</i>	stability requirement	≥ 0	≥ 0
g_6	$\hat{\sigma}_1$	normalized stress at wing root	≤ 0	≤ 0
g_7	$\hat{\sigma}_2$	normalized stress at 33% span	≤ 0	≤ 0
g_8	$\hat{\sigma}_3$	normalized stress at 67% span	≤ 0	≤ 0

objective is to minimize a weighted average of ticket price for the aircraft fleet, with respect to aircraft geometry design variables defined in Table 3.2, subject to performance constraints given in Table 3.3, and subject to the main wing commonality requirement of Eq. 3.19. The commonality requirement may be handled implicitly by using the same value for the shared design variables; this eliminates the need for any equality design constraints. The design problem formulation is:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \end{aligned} \quad (3.21)$$

3.4 Generalized Truss Design

A generalized analysis and design formulation for structural trusses was developed for the purpose of testing methods for decomposition-based design optimization. The formulation enables definition of structural design problems of any size and with a wide variety of system topologies. The design problem is easily partitioned. These factors make this truss formulation ideal for empirical studies, as well as testing the effects of partitioning and coordination decision techniques on problems with a wide range of problem structures. The analysis of truss structures is reviewed in this section, followed by a definition of the design problem and generalized formulation.

3.4.1 Truss Analysis

Trusses are structural systems comprised of bars connected together at their ends via pin joints, and support one or more loads concentrated at pin joints [19]. Each bar is a structural member that bears an axial load, either in tension or compression. Bending moments are

not developed within truss members because pin joints cannot resist torque. Truss members must be configured such each member is constrained against rotation by adjacent members; i.e., the number of mechanical degrees of freedom in the system must be zero. Otherwise, the system would allow motion and it would be a mechanism rather than a structure. In a two-dimensional truss each member has three degrees of freedom before it is attached to any joints: vertical translation, horizontal translation, and rotation. Attaching the end of a member to a fixed joint removes two degrees of freedom, and connecting an end to a joint with one degree of freedom removes one degree of freedom. If we remove exactly enough degrees of freedom for the system to be a structure, the internal force of each member can be solved for using the equations of static equilibrium:

$$\sum_{k \in \mathcal{A}_i} \mathbf{f}_{ik} + \mathbf{F}_i + \mathbf{R}_i = \mathbf{0}, \quad \forall i \in \mathcal{J} \quad (3.22)$$

where \mathcal{A}_i is the set of all indices of joints connected to joint i , \mathcal{J} is the set of all joint indices, \mathbf{F}_i is the vector load applied to joint i , and \mathbf{R}_i are the reaction forces on joint i . Not all joints have applied loads, and only joints at a fixed ground location have reaction forces. We describe a truss in terms of its joints. A member is designated by the numbered joints it connects, e.g., member $\{2, 6\}$ is connected to joint 2 at one end and joint 6 at the other. The vector force \mathbf{f}_{ik} is the axial force in member $\{i, k\}$, which can be interpreted as the force exerted on joint i by member $\{i, k\}$. An example truss is illustrated in Fig. 3.7. Each joint is labeled. Joints 1 and 4 are ground joints. External loads \mathbf{F}_2 and \mathbf{F}_3 are applied to joints 2 and 3, respectively. Loads can be applied only at joints in truss systems without inducing any bending moments. A free-body diagram for member $\{2, 4\}$ is shown to the right of the truss.

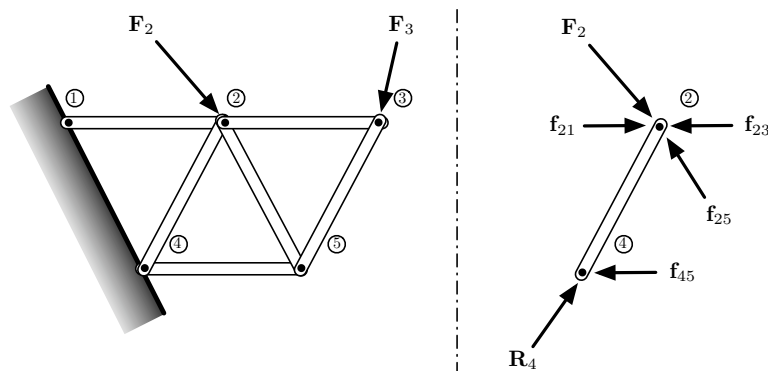


Figure 3.7 Truss geometry and free-body diagram

A truss can fail due to axial stress or compressive buckling. We assume here that failure does not occur at the joints. The internal forces can be used to evaluate axial stress and

failure conditions. The truss members are assumed to have a circular cross sections, and to be made of an isotropic material with stiffness modulus E . The axial stress of member $\{i, j\}$ is

$$\sigma_{ij} = \frac{\|\mathbf{f}_{ij}\|_2}{\pi r_{ij}^2} \quad (3.23)$$

where r_{ij} is the section radius of member $\{i, j\}$. The axial stress must not exceed the material failure stress σ_{allow} . Axial stress is normally the failure mode only when the member is in tension. Long, slender truss members tend to exhibit buckling failure before the axial stress reaches σ_{allow} . High axial compression can cause an unstable condition where perturbing a member in the lateral direction will cause a lateral deflections to increase without bound [59]. The equation governing the lateral deflection v of member $\{i, j\}$ with pin joints at each end under a compressive axial load P is

$$EIv'' + Pv = 0 \quad (3.24)$$

where I is the area moment of inertia for the member section. For a circular section, $I = \pi r_{ij}^4/4$. According to Euler buckling theory, buckling can occur when the solution to Eq. (3.24) v is non-zero. This requires that $\sin(n\pi L) = 0$, where L is the member length and $n = \{1, 2, 3, \dots\}$. The first mode of the solution to Eq. (3.24) occurs when $n = 1$, and the corresponding axial load is:

$$P_{cr} = \frac{\pi^2 EI}{L^2} \quad (3.25)$$

P_{cr} is the smallest compressive axial load under which buckling will occur according to Euler buckling theory. The largest compressive axial load that member $\{i, j\}$ can bear before the possibility of buckling in our model is:

$$b_{ij} = \frac{\pi^3 r_{ij}^4 E}{4L^2} \quad (3.26)$$

Some truss structures are over-constrained; they have more kinematic constraints than necessary to ensure zero degrees of freedom. The number of unknown forces exceeds the number of equilibrium equations. The internal forces cannot be solved from static equilibrium conditions alone, and therefore these structures are termed statically indeterminate. This problem can be resolved by including additional compatibility equations that relate internal forces to deformed joint positions. The larger system of equations that results can be solved for the internal forces and deformed joint positions. The undeformed (original) location of joint i is the vector $\mathbf{u}_i = [u_{xi}, u_{yi}]$, and the deformed location of joint i after

load application is $\mathbf{d}_i = [d_{xi}, d_{yi}]$. It is helpful to relate the vector form of the axial force in member $\{i, j\}$ to the magnitude of this force $f_{ij} = \|\mathbf{f}_{ij}\|_2$ and the deformed location of its joints, as shown in Eq. (3.27).

$$\mathbf{f}_{ij} = f_{ij} \frac{\mathbf{d}_j - \mathbf{d}_i}{\|\mathbf{d}_j - \mathbf{d}_i\|_2} \quad (3.27)$$

Note that the force exerted by member $\{i, j\}$ on joint j is $\mathbf{f}_{ji} = -\mathbf{f}_{ij}$. The compatibility equations ensure that the member ends connected to the same joint are collocated after deformation:

$$\|\mathbf{d}_i - \mathbf{d}_j\|_2 - \|\mathbf{u}_i - \mathbf{u}_j\|_2 - \frac{f_{ij} \|\mathbf{u}_i - \mathbf{u}_j\|_2}{\pi r_{ij}^2 E} = 0, \quad \forall \{i, j\} \in \mathcal{M} \quad (3.28)$$

where \mathcal{M} is the set of all unordered member index pairs. Note that the ordered pair $\{\mathcal{J}, \mathcal{M}\}$ comprises an undirected graph that describes truss topology.

3.4.2 Truss Design Formulation

The class of trusses considered in this generalized formulation include those with members secured via joints at each end, with two or more fixed ground joints, and at least one load is applied to a non-ground joint. Topologies that exhibit static indeterminacy are allowed in this formulation. In the design problem, not only can member radii be varied, but the position of certain joints can be specified. Thus, the design problem is a sizing and shape optimization problem. Joints whose undeformed locations are design variables, i.e., movable joints, are those that are not ground joints and have no applied loads. For convenience, the indices of all fixed ground joints is defined as the set \mathcal{G} , and the set of all joint indices with an applied external load force is \mathcal{L} . The vector of all movable joint locations is $\mathbf{m} = [\mathbf{u}_{i_1}, \mathbf{u}_{i_1}, \dots, \mathbf{u}_{i_k}]$, where $\{i_1, i_2, \dots, i_k\} = \mathcal{J} \setminus (\mathcal{G} \cup \mathcal{L})$, and $k = |\mathcal{J} \setminus (\mathcal{G} \cup \mathcal{L})|$.

The truss design problem is to select the radii of all members \mathbf{r} and positions of all movable joints \mathbf{m} such that the system mass is minimized without violating axial stress or buckling constraints. Ground and load joints have prescribed locations in the design problem, but the other joints are considered moveable. Since statically indeterminate systems are allowed, both structural compatibility and joint equilibrium equations are included in the analysis. State variables include internal member forces ($\mathbf{f} = [f_{i_1 j_1}, f_{i_2 j_2}, \dots, f_{i_k j_k}]$, $\{\{i_1, j_1\}, \{i_2, j_2\}, \dots, \{i_k, j_k\}\} = \mathcal{M}$, and k is the number of truss members), the deformed positions of non-ground joints ($\tilde{\mathbf{d}} = [\mathbf{d}_{i_1}, \mathbf{d}_{i_2}, \dots, \mathbf{d}_{i_k}]$ where $\{i_1, i_2, \dots, i_k\} = \mathcal{J} \setminus \mathcal{G}$ and $k = |\mathcal{J} \setminus \mathcal{G}|$), and the reaction forces ($\mathbf{R} = [\mathbf{R}_{i_1}, \mathbf{R}_{i_2}, \dots, \mathbf{R}_{i_{|\mathcal{G}|}}]$, $\{i_1, i_2, \dots, i_{|\mathcal{G}|}\} = \mathcal{G}$). The all-at-once (AAO) optimization formulation [35] for the general truss design problem

includes both design (\mathbf{m}, \mathbf{r}) and state ($\mathbf{f}, \tilde{\mathbf{d}}, \mathbf{R}$) variables as decision variables, and treats state equations as equality constraints:

$$\begin{aligned}
& \min_{\mathbf{m}, \mathbf{r}, \mathbf{f}, \tilde{\mathbf{d}}, \mathbf{R}} \sum_{\{i, j\} \in \mathcal{M}} \Omega_{ij} & (3.29) \\
\text{subject to: } & |\sigma_{ij}| - \sigma_{\text{allow}} \leq 0, \quad \forall \{i, j\} \in \mathcal{M} \\
& -f_{ij} - b_{ij} \leq 0, \quad \forall \{i, j\} \in \mathcal{M} \\
& \|\mathbf{d}_i - \mathbf{d}_j\|_2 - \|\mathbf{u}_i - \mathbf{u}_j\|_2 - \frac{f_{ij} \|\mathbf{u}_i - \mathbf{u}_j\|_2}{\pi r_{ij}^2 E} = 0, \quad \forall \{i, j\} \in \mathcal{M} \\
& \sum_{k \in \mathcal{A}_i} \mathbf{f}_{ik} + \mathbf{F}_i + \mathbf{R}_i = \mathbf{0}, \quad \forall i \in \mathcal{J} \\
\text{where: } & \sigma_{ij} = \frac{f_{ij}}{\pi r_{ij}^2}, \quad b_{ij} = \frac{\pi^3 r_{ij}^4 E}{4 \|\mathbf{u}_i - \mathbf{u}_j\|_2^2}, \quad \Omega_{ij} = \rho \pi r_{ij}^2 \|\mathbf{u}_i - \mathbf{u}_j\|_2
\end{aligned}$$

Design parameters include material density ρ , elastic modulus E , allowable stress σ_{allow} , fixed ground and load joint positions ($\mathbf{d}_i, \forall i \in \mathcal{G} \cup \mathcal{L}$), and applied loads ($\mathbf{F}_i, \forall i \in \mathcal{L}$). The mass of member $\{i, j\}$ is Ω_{ij} . A specific truss design example with eight members and two loads is presented in Section 6.3, and is used to demonstrate an evolutionary algorithm [49] for making partitioning and coordination decisions.

3.5 Electric Water Pump Design

This section describes a design optimization model for an automotive water (coolant) pump driven by a DC electric motor. It involves several strongly interacting analysis functions, and is suitable for use as a test problem for decomposition-based design optimization. It is used in Chapter 5 to demonstrate a deterministic approach to optimal partitioning and coordination decision-making, and is again used in Chapter 6 to compare the results of an evolutionary algorithm against deterministic results.

The analysis functions in the model for the electric water pump design problem are based on sets of explicit algebraic equations. A similar model was presented in [7] that involved the design of a belt-driven electric sump pump. The model presented here is more sophisticated, and uses an alternative approach to solve for the steady state operating speed. In [7] the analysis functions generated torque-speed curves for the pump and motor, and then solved for the intersection of those curves to find the operating point. Generating these curves was costly, but eliminated some of the feedback coupling between analysis functions. In this model the operating speed is treated as a coupling variable, and instead of determining the operating speed via torque-speed curves, it is solved for as an unknown

state variable. This results in strong coupling between analysis functions, but makes for a more interesting analysis structure that can be used for the analysis of decomposition-based design optimization methods as well as in the study of partitioning and coordination decision approaches.

3.5.1 Water Pump Design

Traditional automotive water pumps are belt driven by the vehicle engine, constraining pump shaft speed to engine speed multiplied by the belt speed ratio. In some cases, such as when a vehicle is idling after a period of high load, the water pump must provide flow and pressure sufficient to cool a hot engine at low shaft speeds. Pumps can be designed to operate efficiently at a specific operating point, but cannot be simultaneously efficient at high and low shaft speeds. An engine-driven pump operates at off-design flow conditions during much of its duty cycle due to large speed fluctuations. This characteristic of belt-driven pumps results in higher power consumption than pumps driven by a constant-speed source. A motor-driven pump can provide constant input speed, and further reduces energy consumption by only pumping when needed. Traditional water pumps are run continuously and utilize a thermostat-controlled bypass when engine cooling is not required.

Electrification of belt-driven automotive components is a promising means for improving fuel economy. Electrification of some components, such as cooling fans, has been incorporated for some time into production vehicles. Electrification of additional components is a more recent endeavor [77, 82, 98]. Surampudi et al. tested a speed-controlled electric water pump on a class-8 tractor and measured an 80% reduction in energy consumption [137].

The analysis model used in this design problem involves five interdependent analysis functions that compute performance metrics based on ten design variable values. These quantities are defined in Table 3.4. Design variables x_1 – x_5 define motor geometry, and x_6 – x_{10} define pump geometry. The motor is a permanent magnet brushed DC electric motor with four pole pairs, and the pump is single-stage centrifugal with six impeller blades and a single diffuser vane (Fig. 3.8). The motor directly drives the pump, so their shaft speeds are identical: $\omega_{\text{motor}} = \omega_{\text{pump}} = \omega$.

Several analysis interactions are modeled. For example, the temperature is computed based on the motor current and speed, but the temperature affects the electrical resistance and current, and the current influences the motor speed. The interdependence between analysis functions is illustrated in Fig. 3.9. Since pump pressure P and input shaft torque τ depend on identical sets of design variables and analysis outputs, only τ is included in the graph for simplicity and represents both P and τ calculations.

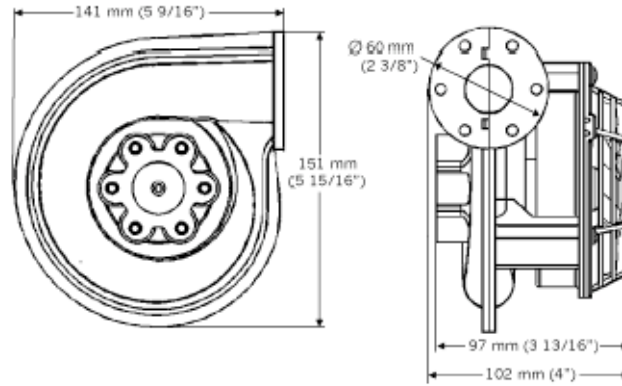


Figure 3.8 Electrically driven centrifugal water pump [39]

The design problem, formally stated in Problem (3.30) is to minimize the electric power P_e consumed by the water pump, while ensuring sufficient pressure differential, safe motor

Table 3.4 Analysis functions and design variables for the electric water pump design problem

Analysis Functions	
$T = a_1(I, \omega, d, d_2, d_3, L, \ell_c)$	motor winding temp. (K)
$I = a_2(\tau, T, d, d_2, d_3, L)$	motor current (amps)
$\omega = a_3(I, T, d, d_2, d_3, L, \ell_c)$	motor speed (rad/sec)
$\tau = a_4(\omega, D_2, b, \beta_1, \beta_2, \beta_3)$	pump drive torque (Nm)
$P = a_5(\omega, D_2, b, \beta_1, \beta_2, \beta_3)$	pressure differential (kPa)

Design Variables	
$x_1 = d$	motor wire diameter (m)
$x_2 = d_2$	inner motor armature diameter (m)
$x_3 = d_3$	outer motor armature diameter (m)
$x_4 = L$	motor armature length (m)
$x_5 = \ell_c$	motor commutator length (m)
$x_6 = D_2$	pump impeller diameter (m)
$x_7 = b$	pump impeller blade width (m)
$x_8 = \beta_1$	pump blade angle at inlet (rad)
$x_9 = \beta_2$	pump blade angle at outlet (rad)
$x_{10} = \beta_3$	pump diffuser inlet angle (rad)

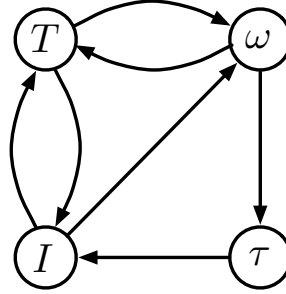


Figure 3.9 Analysis interactions in electric water pump model

temperature, compatible axial motor length, and a desired flow rate.

$$\begin{aligned}
 & \min_{\mathbf{x}} && P_e = VI \\
 & \text{subject to} && P \geq P_{min} = 100 \text{ kPa} \\
 & && T \leq T_{max} = 428 \text{ K} \\
 & && L + \ell_c \leq 0.2 \text{ m} \\
 & && Q = 1.55 \cdot 10^{-3} \text{ m}^3/\text{sec}
 \end{aligned} \tag{3.30}$$

The source voltage V is 14.4 volts. The pressure differential and flow constraints ensure the engine is adequately cooled. The flow constraint is implicitly satisfied during the torque and pressure analysis. The temperature constraint ensures the motor wire insulation is not damaged, and the constraint on axial motor length ($L + \ell_c$) is required for packaging.

The analysis functions are very strongly coupled; first and second-order algorithms failed in most cases to find a solution to the system of equations in Table 3.4. The design problem was successfully solved using mesh adaptive direct search [2] and the individual disciplinary feasible (IDF) formulation, which is described in Section 4.1. The minimal power consumption is 140 W, a substantial improvement over traditional water pumps of similar capacity, which consume nearly 300 W continuously [77]. The following sections describe in detail the calculation of each of the five analysis functions.

3.5.2 Analysis Overview

The functions in Table 3.4 are evaluated by solving systems of nonlinear algebraic equations that approximate motor and pump behavior under steady state operating conditions. Several equations are coupled and require iterative solution techniques. The motor winding temperature T is computed using a thermal resistance model similar to that found in [100], adapted for permanent magnet DC motors. Additional heat transfer correlations were obtained from

[75]. The motor current I and shaft speed ω are computed based on fundamental DC motor equations [78, 60] adapted to the specific geometry of this motor. The pump drive torque and pressure differential are computed for a prescribed flow rate Q using fluid mechanics equations for centrifugal pumps [144]. Model parameters used in this analysis are listed in Table 3.5. Motor and pump geometry will be described, followed by description of the analysis equations. Full description of the physics underlying the equations presented can be found in the references.

Table 3.5 Electric water pump model parameters

δ_a	0.002	motor air gap (m)
δ_h	0.010	motor housing thickness (m)
d_1	0.008	motor shaft diameter (m)
g	9.81	gravitational constant (m/s^2)
T_∞	350	engine compartment temp. (K)
n_a	1.50	slot/tooth ratio
p	4	one-half number of poles
n_p	0.55133	wire packing ratio
μ_0	18.27e-6	viscosity constant (Pa·s)
T_0	291.15	base temperature (K)
C	120	viscosity parameter (K)
C_p	1.009	air heat capacity (kJ/kg·K)
B_r	0.10	remanent magnetic flux density (T)
V	14.4	source voltage (V)
D_s	0	pump shaft diameter (m)
D_1	0.020	impeller inlet diameter (m)
D_3	0.150	volute throat mean inlet diameter (m)
D_4	0.032	exit flange inner diameter (m)
n_B	6	no. impeller blades
n_V	1	no. diffuser vanes
b_3	0.032	diffuser inlet width (m)
ρ_c	970	coolant density (kg/m^3)
Q	0.00155	flow rate (m^3/s)
C_{DF}	$1 \cdot 10^{-7}$	disk friction coefficient ($\text{m}^2\text{s}^2/\text{kg}$)
C_{SF}	$5 \cdot 10^{-3}$	skin friction coefficient (m^{-1})
C_{VD}	0.5	diffuser loss coefficient
C_{in}	0.8	diffuser approach coefficient

Motor Geometry

Figure 3.10 provides a side section view of the DC motor. The iron-cored armature, or rotor, of the DC motor rotates within cylindrical permanent magnets with remanent magnetic flux density B_r . The outer armature diameter is d_3 , the armature axial length is L , and the gap

between the armature and magnets is δ_a . The thickness of the magnets is δ_h . A shaft of diameter d_1 runs through the armature and is supported by bearings at each end of the motor housing. The commutator is a mechanical switch that routes electrical current through the correct armature windings at the appropriate time as the motor rotates. Stationary brushes sliding on the commutator provide the means for electrical conduction (brushes not shown). The commutator is mounted on the motor shaft and has outer diameter d_c and axial length ℓ_c .

Armature geometry is approximated as shown in Fig. 3.11. Insulated copper windings with diameter d run through each of the $2p = 8$ armature slots, where p is the number of magnetic pole pairs. Each slot has a depth of $(d_3 - d_2)/2$ and subtends the angle θ_s . The armature teeth separate the windings and each tooth subtends an angle of θ_t .

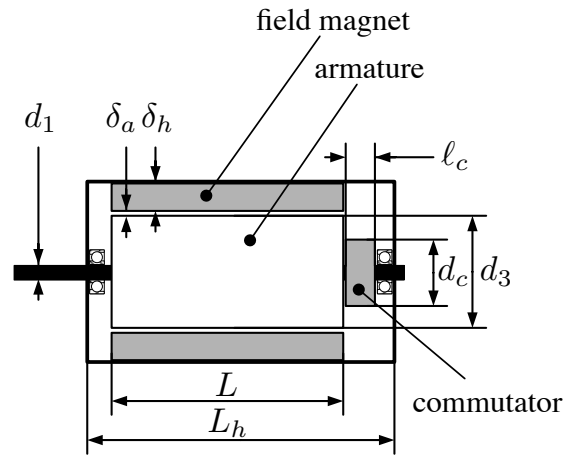


Figure 3.10 Schematic of permanent magnet DC motor

Pump Geometry

Figure 3.12 illustrates the geometry of the centrifugal water pump. The drive shaft of diameter D_s rotates the impeller, which has $n_B = 6$ blades. Coolant flows in through the inlet of diameter D_1 and is expelled radially outwards due to impeller rotation. As the coolant flows through the diffuser its velocity is reduced, but it experiences an increase in pressure according to Bernoulli's principle. The outer impeller diameter is D_2 , and the volute inlet diameter is D_3 . The inlet impeller blade angle is β_1 , and the outlet impeller blade angle is β_2 . The diffuser inlet blade angle (not shown) is β_3 . The exit flange diameter is D_4 . The impeller blade width is b and the diffuser inlet width is b_3 .

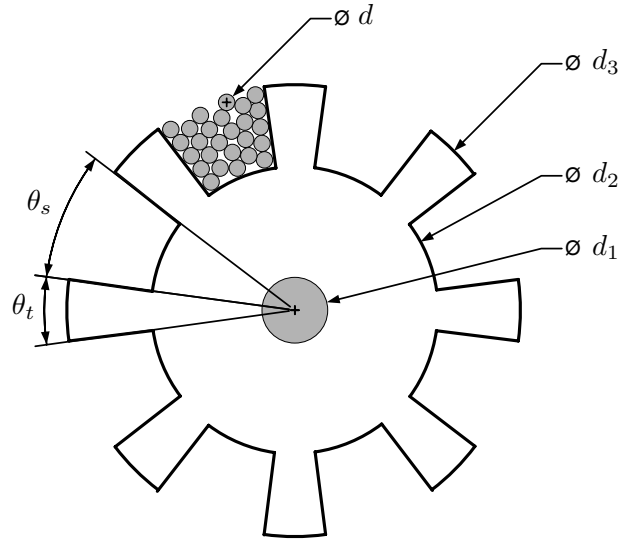


Figure 3.11 Section view of DC motor armature

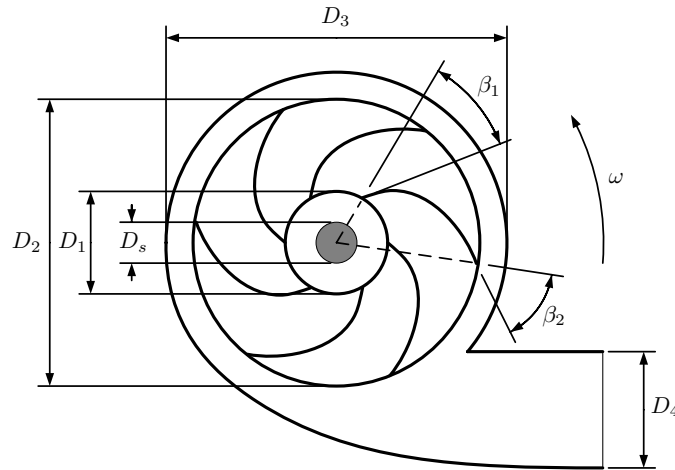


Figure 3.12 Schematic of centrifugal water pump

3.5.3 Thermal Analysis

The objective of the thermal analysis is to compute the armature winding temperature T given motor geometry and operating conditions. Motor geometry is simplified to enable analysis using a thermal resistance model. All heat generation is assumed to originate from armature windings, evenly distributed throughout the annular cylinder containing armature windings with outer diameter d_3 , inner diameter d_2 , and length L . First it will be shown how to compute heat generation due to I^2R losses. The thermal circuit and equations will then be presented, followed by a description of the solution process for the thermal analysis.

Wire Length and Heat Generation

Calculation of heat generation requires knowledge of armature electrical resistance, which depends on total wire length and wire temperature. The model for wire length assumes windings are made around each armature tooth, and accounts for wire curvature and extension beyond armature ends. The slot to tooth volume ratio, $n_a = \theta_s/\theta_t$, is assumed to be fixed at 1.5. The total volume occupied by wire passing through the slots V_s can be calculated using the following formulae:

$$\theta_s = \pi/p(1 + 1/n_a), \theta_t = \theta_s/n_a, \theta_p = \theta_s + \theta_t$$

$$A_s = \theta_s(d_3^2 - d_2^2)/8, V_s = n_p A_s L$$

where θ_p is the angle between poles, A_s is the section area of each slot, and $n_p = 0.55133$ is the packing ratio, i.e., the ratio of wire volume occupying slots to total slot volume. The ratio n_p was calculated based on close packing geometry. $A_w = \pi d^2/4$ is the sectional area of a single wire, and the total length of wire passing through armature slots is $\ell_s = V_s/A_w$. The total number of winding turns for all poles is $n_t = \ell_s/2L$. In reality n_t is integer valued, but is assumed to be relaxed to a continuous number here. This provides a reasonable approximation when n_t is large. The average wire length between slots for each winding turn is $\bar{\ell}_e = \theta_s(1/n_a + \pi/4)(d_2 + d_3)/4$, and the total length of wire is:

$$\ell = \ell_s + 2\bar{\ell}_e n_t \quad (3.31)$$

The resistivity of copper varies with temperature, and is approximated using a linear model:

$$\rho(T) = 1.72 \cdot 10^{-8}(1 + 0.00393(T - 293)) \quad (3.32)$$

The total heat generated by the armature windings due to I^2R losses is:

$$S = 4\rho\ell I^2/\pi d^2 \quad (3.33)$$

Thermal Resistance Model

The thermal resistance model used in this analysis to calculate wire temperature T is illustrated in Fig. 3.13. The heat source is the annulus ring of the armature containing the windings, and the thermal sink is the engine compartment at temperature T_∞ .

The first path in the circuit passes through R_1 , R_2 , and R_3 , and represents the thermal path directly from the armature through the air gap, magnets, and housing to the engine

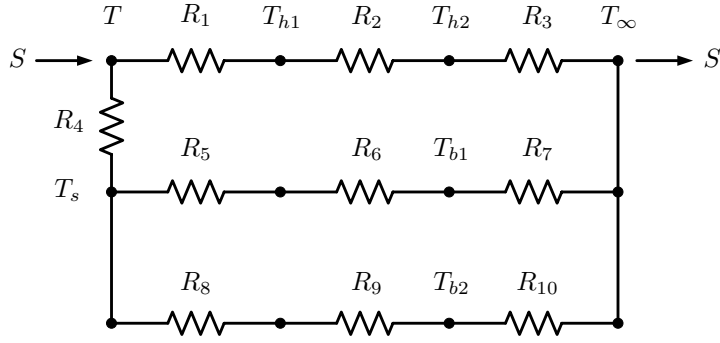


Figure 3.13 DC motor thermal resistance model

compartment. All other thermal energy flows inward through the inner armature R_4 , and then through the shaft to either side of the motor. The path through R_5 , R_6 , and R_7 corresponds to the side with the commutator, and the remaining path corresponds to the side without the commutator.

Before the thermal resistance formulae are detailed, models for material property dependence on temperature are presented. These are used in many of the resistance calculations. The models for air viscosity, density, and conductivity are:

$$\mu_{air}(T) = \mu_0(T_0 + C)(T/T_0)^{3/2}/(T + C) \quad (3.34)$$

$$\rho_{air}(T) = 1.01325/287.05T \quad (3.35)$$

$$k_{air}(T) = 1.5207 \cdot 10^{-11}T^3 - 4.8574 \cdot 10^{-8}T^2 + 1.0184 \cdot 10^{-4}T - 3.9333 \cdot 10^{-4} \quad (3.36)$$

The model for iron conductivity is linear:

$$k_{fe}(T) = 110.4676 - 0.1002T \quad (3.37)$$

R_1 : Convection between armature and field magnets

The Reynold's and Nusselt numbers for the gap between the armature and field magnets are $Re_1 = \omega d_3^2 \rho_{air}(T)/2\mu_{air}(T)$ and $Nu_1 = 0.318Re_1^{0.571}$, respectively. The convective

thermal resistance between the the armature and field magnets is:

$$R_1 = 2/Nu_1 k_{air}(T) \quad (3.38)$$

R_2 : Conduction through field magnets and housing

The resistance due to the thin motor housing is considered negligible compared to that of the field magnets, so only one material is considered in this resistance calculation. The inner diameter of the field magnets is $d_4 = d_3 + 2\delta_a$, and the outer diameter of the housing is $d_5 = d_4 + 2\delta_h$. The temperature of the inner magnet surface is T_{h1} and the temperature of the outer housing surface is T_{h2} . The thermal conductivity of the field magnets is calculated using Eq. (3.37) and the average magnet temperature $\bar{T}_h = (T_{h1} + T_{h2})/2$. The thermal resistance due to conduction through the field magnets and housing is:

$$R_2 = \ln(d_5/d_4)/(2\pi L k_{fe}(\bar{T}_h)) \quad (3.39)$$

R_3 : Convection from housing to engine compartment

The total length of the motor housing, accounting for space for bearings and clearance, is approximated as $L_h = L + 1.5\ell_c + 2d_1$. The volumetric thermal expansion coefficient in the calculation of R_3 is approximated as $\beta_{t3} = 1/T_{h2}$. The kinematic viscosity of air in this region is $\nu_{air3} = \mu_{air}(T_{h2})/\rho_{air}(T_{h2})$. The air heat capacity is $C_p = 1.009$, and the thermal diffusivity here is $\alpha_3 = k_{air}(T_{h2})/C_p \rho_{air}(T_{h2})$. The Prandtl number here is $Pr_3 = \nu_{air3}/\alpha_3$, and the Rayleigh and Nusselt numbers are:

$$Ra_3 = g\beta_{t3}(T_{h2} - T_\infty)L_h^3/\alpha_3\nu_{air3}$$

$$Nu_3 = \left(0.6 + \frac{0.387Ra_3^{1/6}}{\left((1 + (0.559/Pr_3)^{9/16})^{8/27} \right)} \right)^2$$

The thermal resistance due to free convection from the motor housing to the engine compartment is:

$$R_3 = 2/(k_{air}(T_{h2})Nu_3L_h) \quad (3.40)$$

R_4 : Conduction through armature core

The conductivity of the core is based on the average between the winding and shaft temperatures: $T_c = (T + T_s)/2$. It is assumed that the shaft within the core is at a constant temperature T_s throughout its volume and has no thermal resistance. The thermal

conductivity resistance through the armature core is:

$$R_4 = \ln(d_2/d_1)/2\pi Lk_{fe}(T_c) \quad (3.41)$$

R_5 : Conduction through motor shaft and bearings on commutator side

The section area of the shaft is $A_s = \pi d_1^2/4$, and the length of the shaft extending beyond the armature is $L_s = 1.5\ell_c + d_1$. The thermal resistance through the support bearing is assumed to be a constant 1.5 K/W. The thermal resistance through this section of shaft is:

$$R_5 = L_s/A_s k_{fe}(T_s) + 1.5 \quad (3.42)$$

R_6 and R_9 : Conduction through motor end bells

The motor end bells are the disk-shaped portions of the housing at each end of the motor. It is assumed that the conductive resistance through both end bells is negligible. Therefore, $R_6 = R_9 = 0$.

R_7 : Convection between end bell and engine compartment on commutator side

The volumetric thermal expansion coefficient in the calculation of R_7 is approximated as $\beta_{t7} = 1/T_{b1}$, where T_{b1} is the temperature of the commutator side end bell. The kinematic viscosity is $\nu_{air7} = \mu_{air}(T_{b1})/\rho_{air}(T_{b1})$, and the thermal diffusivity is $\alpha_7 = k_{air}(T_{b1})/C_p \rho_{air}(T_{b1})$. The Prandtl number is $Pr_7 = \nu_{air7}/\alpha_7$, and the Rayleigh and Nusselt numbers are:

$$Ra_7 = g\beta_{t7}(T_{b1} - T_\infty)d_3^3/\nu_{air7}\alpha_7$$

$$Nu_7 = \left(0.825 + \frac{(0.387Ra_7^{1/6})}{(1 + (0.492/Pr_7)^{9/16})^{8/27}} \right)^2$$

The thermal resistance due to free convection from the commutator side end bell to the engine compartment is:

$$R_7 = 8/\pi Nu_7 d_4 k_{air}(T_{b1}) \quad (3.43)$$

R_8 : Conduction through motor shaft and bearings on non-commutator side

The thermal resistance through the section of motor shaft extending beyond the armature on the non-commutator side is:

$$R_8 = 4/\pi d_1 k_{fe}(T_s) + 1.5 \quad (3.44)$$

R_{10} : Convection between end bell and engine compartment on non-commutator side

The volumetric thermal expansion coefficient in the calculation of R_{10} is approximated

as $\beta_{t10} = 1/T_{b2}$, where T_{b2} is the temperature of the non-commutator side end bell. The kinematic viscosity is $\nu_{air10} = \mu_{air}(T_{b2})/\rho_{air}(T_{b2})$, and the thermal diffusivity is $\alpha_{10} = k_{air}(T_{b2})/C_p\rho_{air}(T_{b2})$. The Prandtl number is $Pr_{10} = \nu_{air10}/\alpha_{10}$, and the Rayleigh and Nusselt numbers are:

$$Ra_{10} = g\beta_{t10}(T_{b2} - T_{\infty})d_3^3/\nu_{air10}\alpha_{10}$$

$$Nu_{10} = \left(0.825 + \frac{0.387Ra_{10}^{1/6}}{(1 + (0.492/Pr_{10})^{9/16})^{8/27}} \right)^2$$

The thermal resistance due to free convection from the non-commutator side end bell to the engine compartment is:

$$R_{10} = 8/\pi Nu_{10} d_4 k_{air}(T_{b2}) \quad (3.45)$$

Temperature equations

The total thermal resistance R and several intermediate resistance quantities are required for solution of the system temperatures:

$$R_{1,2,3} = R_1 + R_2 + R_3$$

$$R_{5,6,7} = R_5 + R_6 + R_7$$

$$R_{8,9,10} = R_8 + R_9 + R_{10}$$

$$R_{5-10} = 1/(1/R_{5,6,7} + 1/R_{8,9,10})$$

$$R_{4-10} = R_4 + R_{5-10}$$

$$R = 1/(1/R_{1,2,3} + 1/R_{4-10})$$

With these resistances defined we can now present the coupled system of equations based on the circuit in Fig. 3.13 that model the steady state thermal behavior of the DC motor:

$$T = T_{\infty} + RS \quad (3.46a)$$

$$\Delta T = T - T_{\infty} \quad (3.46b)$$

$$q_{p1} = \Delta T / R_{1,2,3} \quad (3.46c)$$

$$q_{p2} = \Delta T / R_{4-10} \quad (3.46d)$$

$$T_{h1} = T - q_{p1}R_1 \quad (3.46e)$$

$$T_{h2} = T_{h1} - q_{p1}R_2 \quad (3.46f)$$

$$T_s = T - q_{p2}R_4 \quad (3.46g)$$

$$q_{p3} = (T_s - T_{\infty}) / R_{5,6,7} \quad (3.46h)$$

$$q_{p4} = (T_s - T_{\infty}) / R_{8,9,10} \quad (3.46i)$$

$$T_{b1} = T_s - q_{p3}(R_5 + R_6) \quad (3.46j)$$

$$T_{b2} = T_s - q_{p4}(R_8 + R_9) \quad (3.46k)$$

where ΔT is the temperature drop between the armature windings and the engine compartment, and q_{p1} , q_{p2} , q_{p3} , and q_{p4} are heat flows through circuit paths formed by $R_{1,2,3}$, R_{4-10} , $R_{5,6,7}$, and $R_{8,9,10}$, respectively. Since many thermal resistance values depend on temperature values the system of equations is coupled, and must be solved using an iterative technique. A least squares approach has been found to be more successful at solving this system than nonlinear Gauss-Seidel or Newton's method.

3.5.4 Motor Current Analysis

The motor current model predicts the current through the motor armature I given the armature temperature T , required motor shaft torque τ , and motor geometry. The magnetic field depends on temperature and is approximated using the formula:

$$B = B_r(1 - 0.0017(T - 293))$$

where B_r is the remanent magnetic flux density (assumed constant). The magnetic flux is $\phi = \pi(d_2 + d_3)LB/2$, and the total number of conductors in the magnetic field is $Z = 2n_t$. The calculation for the number of winding turns n_t was given in Section 3.5.3. The motor current is:

$$I = \tau\pi / \phi pZ \quad (3.47)$$

3.5.5 Motor Speed Analysis

The objective of this analysis is to compute the motor speed ω given the motor current I and armature temperature T . The model accounts for both the electrical resistance through the armature windings and the voltage drop across the commutator. The resistance through the windings is $R_a = 4\rho\ell/\pi d^2$. Recall that Eq. (3.33) is used to model the dependence of copper resistivity on temperature. The length ℓ is computed according to the formulae in Section 3.5.3. The back emf, i.e., voltage generated due to motor rotation that opposes source voltage, is $\mathcal{E} = R_a I$. The voltage drop across the commutator consists of two components:

$$\Delta V_c = \Delta V_{c1} + \Delta V_{c2}$$

A model for each component was developed based on empirical data from [78]:

$$\Delta V_{c1} = 0.8692 - 0.6458 \cdot e^{-0.8207\rho_I}$$

$$\Delta V_{c2} = 1.0037 - 0.5912 \cdot e^{-1.32\rho_I}$$

where $\rho_I = I/A_b$ is the current density at the commutator/brush interface, and $A_b = \pi d_c \ell_c / 2$ is the brush contact area. It is assumed here that the commutator diameter is proportional to armature diameter: $d_c = d_3/2$. The motor speed is:

$$\omega = 2\pi(V - \Delta V_c - IR_a)/\phi Z p \quad (3.48)$$

3.5.6 Torque and Pressure Analysis

The analysis presented in this section computes the torque τ required to drive the pump at the specified motor speed ω and pump flow rate Q . The analysis also computed the pressure differential P across the pump inlet and outlet. The equations here are based on the model presented in [144], and much of the notation has been retained. Some modifications have been made to smooth analysis responses so that optimization can be performed more easily. Note that many of the terms used in the following equations have been defined in Tables 3.4 and 3.5.

The impeller blade tip speed at the outlet is $u_2 = \omega D_2/2$, and the radial fluid velocity relative to the impeller is $W_{m2} = Q/\pi D_2 b$. The fluid velocity leaving the impeller in the direction of the blade is $W_2 = W_{m2}/\cos(\beta_2)$. The Wiesner slip coefficient is:

$$\sigma = 1 - \frac{\sqrt{\sin(\pi/2 - \beta_2)}}{n_B^{0.7}}$$

The tangential velocity of the fluid leaving the impeller relative to a fixed coordinate system is $C_{t2} = u_2 \sigma - W_{m2} \tan(\beta_2)$, and the pump theoretical head is:

$$H_{th} = u_2 C_{t2} / g$$

where g is the acceleration of gravity. The disk friction loss is

$$D_{FH} = \frac{C_{DF} \rho_c \omega^3 (D_2/2)^5}{Q}$$

The fluid velocity at the inlet in the radial direction with respect to a fixed coordinate system is $C_1 = 4Q / (D_1^2 - D_s^2) \pi$. It is possible to configure a pump such that the drive shaft does not impede flow at the inlet. This is assumed to be the case, and therefore $D_s = 0$. The impeller blade tip speed at the inlet is $u_1 = \omega D_1 / 2$, and the velocity of the fluid entering the impeller in the direction of the blade is $W_1 = \sqrt{C_1^2 + u_1^2}$. The inlet flow angle is $\beta_{F1} = \tan^{-1}(u_1 / C_1)$. Several intermediate values must be determined to calculate the skin friction head loss:

$$\begin{aligned} \beta_{s1} &= 2\beta_{F1} - \beta_1 \\ T_1 &= \frac{\sqrt{\cos^2(\beta_{F1}) - \cos(\beta_1) \cos(\beta_{s1})}}{\cos(\beta_{s1})} \\ T_2 &= \begin{cases} T_1 & \text{if } R_{t1} \geq 0 \\ -T_1 & \text{if } R_{t1} < 0 \end{cases} \\ R_{t1} &= \cos(\beta_{F1}) / \cos(\beta_{s1}) \\ x_{L1} &= R_{t1} - T_2 \\ D_{QIN12} &= \frac{W_1^2}{2g x_{L1}^2} \left(1 - \frac{x_{L1} \cos(\beta_{F1})}{\cos(\beta_1)} \right)^2 \\ D_{H12} &= \frac{b D_2 \pi \cos(\beta_2)}{n_B (b + \pi D_2 \cos(\beta_2) / n_B)} \end{aligned}$$

The skin friction head loss is:

$$D_{QSF12} = \frac{C_{SF} (D_2 - D_1) (W_2 + W_1)^2}{8g \cos(\beta_2) D_{H12}}$$

The impeller diffusion loss as presented in [144] is computed differently depending on the ratio W_1 / W_2 . This introduces a discontinuity that can hinder optimization efforts. After

adding an exponential transition function this loss term is:

$$D_{\text{Qdif}} = \frac{W_1^2}{8g(1 + e^{4(1.4 - W_1/W_2)})}$$

The fluid velocity approaching the pump volute is:

$$C_3 = \sqrt{\left(\frac{C_{t2}d_2}{d_3}\right)^2 + \left(\frac{Q}{\pi d_3 b_3}\right)^2}$$

The curved diffuser leading to the pump outlet as shown in Fig. 3.12 is the pump volute. The volute throat velocity is:

$$C_{Q3} = \frac{Q}{\pi d_3 b_3 \cos(\beta_3)}$$

The volute head loss is:

$$D_{\text{QIN23}} = \begin{cases} D_{23} & \text{if } D_{23} \geq 0 \\ 0 & \text{if } D_{23} < 0 \end{cases}$$

where $D_{23} = C_{\text{in}}(C_3^2 - C_{Q3}^2)/2g$. The diffuser skin friction loss is:

$$D_{\text{QSF34}} = C_{\text{SF}} \frac{(d_3 - d_1)(C_{Q3} + C_1)^2}{8g \cos(\beta_3) \frac{b_3 d_3 \pi \cos(\beta_3)}{b_3 n_V + d_3 \pi \cos(\beta_3)}}$$

The diffuser expansion loss term in [144] is calculated differently depending on the value of C_{Q3}/C_1 . Another transition curve is defined here to eliminate the discontinuity:

$$\phi_{\text{QVD}} = \frac{1}{1 + e^{4(1.4 - C_{Q3}/C_1)}}$$

The diffuser expansion loss is:

$$D_{\text{QVD}} = \begin{cases} D_{24} & \text{if } D_{24} \geq 0 \\ 0 & \text{if } D_{24} < 0 \end{cases}$$

where:

$$D_{24} = \frac{(C_{\text{VD}} + \phi_{\text{QVD}}/4)C_{Q3}^2 - C_1^2 \phi_{\text{QVD}}/2}{2g}$$

The actual head developed by the pump is:

$$H_{ac} = H_{th} - D_{QIN12} - D_{QSF12} \\ - D_{QIN23} - D_{Qdif} - D_{QSF34} - D_{QVD}$$

The pressure differential and the shaft torque can now be computed:

$$P = H_{ac}\rho_c g \quad (3.49)$$

$$\tau = \frac{\rho g Q}{\omega} (H_{th} + D_{FH}) \quad (3.50)$$

3.5.7 Optimization Results

The solution to Problem (3.30) obtained using the mesh adaptive direct search algorithm and the IDF formulation is presented in Table 3.6. The minimal power consumption is 140 W, a substantial improvement over traditional water pumps of similar capacity, which consume nearly 300 W continuously [77]. Numerous starting points and algorithm parameters were tested, and the solution presented is the best that has been obtained. It is unknown whether this is the global solution. The stochastic nature of the optimization algorithm makes exact replication difficult, but similar results have been obtained from multiple starting points. Gradient-based algorithms have failed to find a feasible solution thus far.

Table 3.6 Optimization results for the electric water pump design problem

Optimal pump design	
$x_1 = d$	8.4 (mm)
$x_2 = d_2$	76.6 (mm)
$x_3 = d_3$	146 (mm)
$x_4 = L$	145 (mm)
$x_5 = \ell_c$	55.4 (mm)
$x_6 = D_2$	58.8 (mm)
$x_7 = b$	28.5 (mm)
$x_8 = \beta_1$	0.793 (rad)
$x_9 = \beta_2$	1.22 (rad)
$x_{10} = \beta_3$	0.913 (rad)

3.6 Concluding Comments

This chapter introduced several original system design examples, ranging from air flow sensor design to the design of a commercial aircraft fleet. Each example, with the exception of the aircraft family problem, was presented with enough detail for replication. Not only does this enable verification of results, but provides a small library of system design examples for use in other investigations. These examples are used throughout this dissertation to illustrate important concepts and to demonstrate partitioning and coordination techniques. The air flow sensor design problem and the turbine blade design problem are used in Section 4.2.3 to illustrate the influence of coupling strength on solution difficulty for two different single-level formulations. These two examples are useful for this study since their coupling strength can be varied easily. The ATC formulation is demonstrated using the aircraft family problem in Section 4.3.3, and a parametric study on ATC algorithm parameters using this example is presented. The air flow sensor problem is also used to help explain the ALC formulation in Section 4.3.5. The electric water pump design problem is used for comparing partitioning and coordination decision techniques in Chapters 5 and 6. An evolutionary algorithm for making partitioning and coordination decisions for larger systems is presented in Chapter 6, and illustrated using an eight-bar version of the generalized truss design problem.

An electric vehicle design problem was developed to illustrate the applicability of optimal partitioning and coordination decision techniques to a larger and more involved system design problem. Detailed description of this example problem is reserved until Chapter 8.

The following chapter introduces several important single and multi-level formulations for system design optimization. This dissertation emphasizes a class of multi-level formulations suitable for quasiseparable problems that have established convergence proofs, such as ATC and ALC. The review of single-level formulations helps establish concepts that are important to multi-level formulations.

Chapter 4

System Design Optimization Formulations

Solution of a system design problem using decomposition-based design optimization requires the formulation of one or more optimization problems. The numerous formulations that have been proposed can be classified according to several different criteria. One important distinction is whether a formulation is single-level or multi-level. Single-level formulations comprise a single optimization problem. Analysis functions may be distributed and temporarily decoupled, but a single optimization algorithm guides the entire system optimization process. Multi-level methods utilize distributed optimization—a separate optimization problem is defined for each subproblem.

This chapter reviews several important single and multi-level formulations. The single-level review provides important background for understanding multi-level formulations. The partitioning and decision model employed in subsequent chapters assume that a particular class of multi-level formulations is used. The formulation used in decomposition-based design optimization is a defining feature of an implementation. The type of formulation employed dictates what types of partitions, linking structures, and coordination algorithms may be used. Some formulations are better suited for problems with certain structures. Convergence proofs exist for a handful of formulations.

4.1 Single-Level Formulations

Cramer et al. presented three single level formulations and provided guidance and predictions regarding application and performance of these formulations [35]. In these single-level formulations, all decision making is centralized and performed by a single optimization algorithm. This implicitly guarantees shared design variable consistency. Coupling variable consistency can be enforced using either a system analysis algorithm, or auxiliary equality constraints. Single-level formulations can be effective at dealing with systems possessing strong interactions, but are not well suited for problems of large dimension, where multilevel formulations may be preferred [5, 7]. Balling and Sobieski provided a review of single-

level formulations [15], and Balling and Wilkinson implemented these formulations in the solution of analytical test problems [16]. Hulme and Bloebaum [74] compared the implementation of single-level formulations using many test problems of varying size and coupling strength, with emphasis on solution differences that appear to be due to numerical limitations relating to increased problem dimension. This section reviews the three primary single-level formulations and presents some insights into their application.

4.1.1 Multidisciplinary Feasible Formulation

The most basic formulation is the MDF approach, also known as ‘Nested Analysis And Design’ (NAND) or ‘All-in-One’ (AIO). A single system-level optimizer is used, and a separate algorithm performs the system analysis task of finding consistent values for all coupling variables. The optimizer supplies the system analyzer with a design \mathbf{x} , and the system analyzer returns the function values f , \mathbf{g} , and \mathbf{h} . The MDF problem formulation is given in Eq. (2.3) and repeated here:

$$\begin{aligned} \min_{\mathbf{x}} \quad & f(\mathbf{x}, \mathbf{y}_p(\mathbf{x})) \\ \text{subject to} \quad & \mathbf{g}(\mathbf{x}, \mathbf{y}_p(\mathbf{x})) \leq \mathbf{0} \\ & \mathbf{h}(\mathbf{x}, \mathbf{y}_p(\mathbf{x})) = \mathbf{0}, \end{aligned} \tag{4.1}$$

The vector of consistent coupling variables $\mathbf{y}_p(\mathbf{x})$ is computed at every step of the optimization process using a system analysis algorithm. Figure 4.1 illustrates this process.

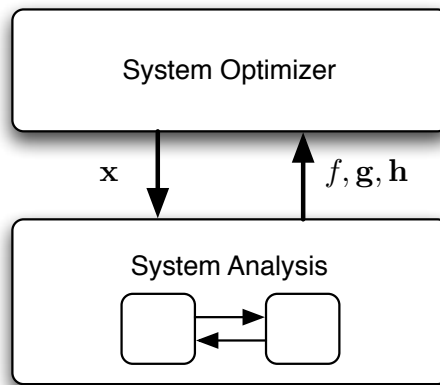


Figure 4.1 MDF architecture

The fixed point iteration (FPI) algorithm is a popular system analysis method for MDF. Section 4.2 will elucidate some challenges in using FPI in conjunction with system optimization. Other system analysis methods exist, but also exhibit their own difficulties, and

will not be discussed in this chapter. A design optimization strategy is classified as MDF if a complete system analysis is performed for every optimization iteration. The analysis is “nested” within the design. The optimizer is responsible to find the optimal design \mathbf{x}^* (the design solution), while the system analyzer is responsible to find $\mathbf{y}_p(\mathbf{x})$ (the system analysis solution).

This approach may be desirable if the subsystems are weakly coupled (fast system analysis convergence), and if the subsystem analyses are not computationally expensive. In addition, MDF eases the incorporation of legacy analysis tools. If a design organization already performs a complete analysis before making a design decision, MDF is a natural fit.

Although the merits of MDF are notable, its shortcomings must be clearly understood. MDF is dependent upon the effectiveness of the system analyzer. If the analyzer does not converge at any point in the process, the optimizer may fail. The nested analysis and optimization process required by MDF can be computationally inefficient, and this motivates approaches that eliminate the need for repeated system analysis [128]. In addition, typical MDF implementations cannot exploit the potential coarse-grained parallelism of distinct subsystem analyses. MDF has been aptly termed a ‘brute force’ approach [25].

4.1.2 Individual Disciplinary Feasible Formulation

In the IDF formulation, an analyzer for each subsystem is employed and a single system-level optimizer is used, but the optimizer, rather than a system analysis algorithm, coordinates the interactions between the subsystem analyses. The IDF architecture is illustrated in Fig. 4.2 using a two-element example system. The first analysis function computes the design constraints \mathbf{g}_1 and \mathbf{h}_1 , as well as the coupling variable \mathbf{y}_{21} . The second analysis function computes the objective function in addition to a coupling variable and a set of design constraints. The optimizer chooses values for both design and coupling variables: system analysis and design are performed simultaneously. Since the system optimizer provides all inputs required for all subsystems concurrently, subsystem analyses may be executed in parallel.

The IDF formulation is given in Eq. (4.2). It differs from the MDF formulation in that the decision variable vector includes both design variables \mathbf{x} and coupling variables \mathbf{y} , while auxiliary constraints \mathbf{h}_{aux} are added to ensure system consistency. This approach eliminates the need to solve for $\mathbf{y}_p(\mathbf{x})$ at each optimization iteration. In cases where solving the system

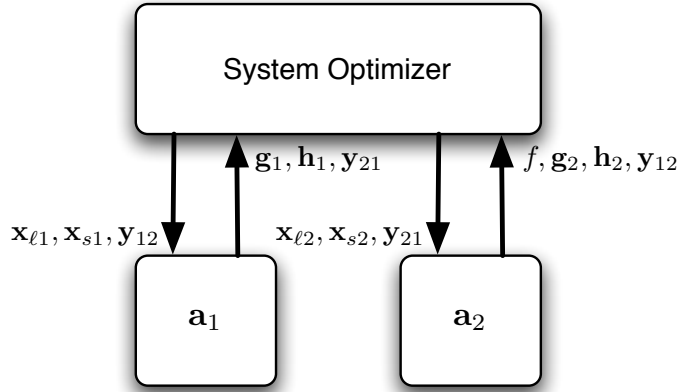


Figure 4.2 IDF architecture

analysis equations is difficult, IDF can offer great benefit.

$$\begin{aligned}
 & \min_{\mathbf{x}=[x_{\ell}, x_s], \mathbf{y}} && f(\mathbf{x}, \mathbf{y}) && (4.2) \\
 & \text{subject to} && \mathbf{g}(\mathbf{x}, \mathbf{y}) \leq \mathbf{0} \\
 & && \mathbf{h}(\mathbf{x}, \mathbf{y}) = \mathbf{0} \\
 & && \mathbf{h}_{\text{aux}}(\mathbf{x}, \mathbf{y}) = \mathbf{y} - \mathbf{a}(\mathbf{x}, \mathbf{y})\mathbf{S} = \mathbf{0}
 \end{aligned}$$

IDF facilitates coarse-grained parallelism, improves convergence properties, and drives the design toward better solutions if multiple analysis solutions exist. If the solution process is interrupted, the intermediate design may not be consistent or feasible. In contrast, an interrupted MDF solution will yield a consistent, but potentially infeasible, design. Since IDF does not require the frequently expensive task of achieving system consistency when far from the solution, the optimization algorithm can trace a more efficient path toward the solution and computational expense is reduced through the elimination of repeated system analysis steps [3].

IDF is more centralized than MDF, and the dimension of the optimization problem is increased since coupling variables are made decision variables. This increase in dimension can reduce numerical solution accuracy when the problem size is large, as evident in the results presented by Hulme and Bloebaum [74]. MDF may be preferable when the dimension of \mathbf{y} is much larger than the dimension of \mathbf{x} [3, 66]. Furthermore, auxiliary equality constraints can introduce numerical solution difficulties [3, 138].

Balling and Sobieski proposed a hybrid approach to handling coupling variable consistency [15]. In MDF the task of satisfying coupling variable consistency is nested within the optimization problem. In IDF it is part of the optimization problem. A hybrid approach

satisfies some consistency constraints using the optimization algorithm, and the rest using an approach nested within the algorithm. If all feedback coupling relationships are handled using auxiliary equality constraints, then the remaining feedforward relationships can be satisfied by simply executing the analysis functions in sequence and utilizing the most recently computed coupling variable values. An implementation of this idea was demonstrated in Section 1.4.

Section 4.2.2 will show that using the IDF formulation can help to find superior solutions that are hidden to MDF implementations, and Section 4.2.3 demonstrates that IDF results in improved computational efficiency for strongly coupled problems (as predicted in [35]). If a high level of centralization is acceptable, IDF may be an ideal design strategy.

4.1.3 All-at-Once Formulation

The All-at-Once (AAO) formulation confers additional tasks to the optimization algorithm beyond those for the IDF formulation. In IDF the optimization algorithm solves the system analysis equations, but relies on analysis algorithms within each analysis function to solve the associated governing equations. For example, an analysis function that evaluates structural integrity of a component may use the finite element method [93] to solve the governing elasticity equations and compute the analysis function outputs. The AAO formulation uses the optimization algorithm instead to solve these governing equations, eliminating the need for the analysis algorithm within each analysis function. The vector of state variables (\mathbf{s}) is added to the decision variable set, and the governing equations are cast as an additional set of auxiliary equality constraints. State variables quantify the state of a system; examples include velocity fields in computational fluid dynamics, strain fields in structural finite element analysis, and component velocities or accelerations in multibody dynamics. Governing equations are satisfied when the associated residuals are zero. The AAO process is illustrated in Fig. 4.3 using a simple example system.

The formulation of the AAO approach is given in Eq. (4.3). Auxiliary equality constraints ensure zero residuals at problem convergence, and the decision variables include \mathbf{x} and \mathbf{s} . The evaluation function $\mathbf{e}(\mathbf{x}, \mathbf{s})$ computes the residuals as well as design constraint and objective values. The selection matrix \mathbf{S}_w extracts the evaluation function outputs that are the evaluation function residuals \mathbf{w} .

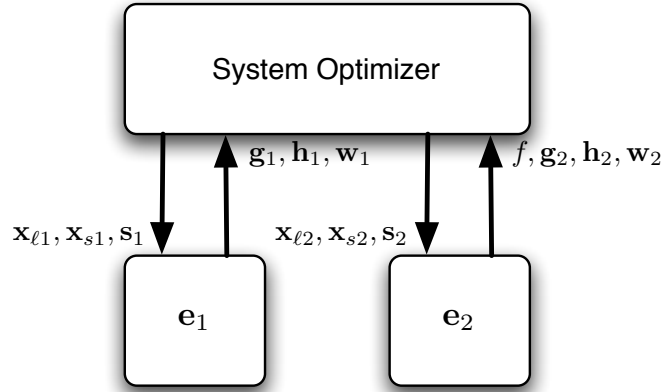


Figure 4.3 AAO architecture

$$\begin{aligned}
 & \min_{\mathbf{x}, \mathbf{y}, \mathbf{s}} && f(\mathbf{x}, \mathbf{s}) \\
 & \text{subject to} && \mathbf{g}(\mathbf{x}, \mathbf{s}) \leq \mathbf{0} \\
 & && \mathbf{h}(\mathbf{x}, \mathbf{s}) = \mathbf{0}. \\
 & && \mathbf{e}(\mathbf{x}, \mathbf{s})\mathbf{S}_w = \mathbf{0}.
 \end{aligned} \tag{4.3}$$

AAO centralizes both design and analysis, but still distributes evaluation of governing equations. This can lead to significant efficiency gains in some cases, but also results in high-dimension optimization problems. AAO can be difficult to implement, and tends to be utilized for very specialized applications where a benefit can be realized. AAO implementations do not fall under the rubric of simulation-based design exactly, which is the emphasis here. Nevertheless, a discussion of AAO is included for completeness, and the truss design example of Section 3.4 uses the AAO formulation.

4.2 System Analysis for Single-level Formulations

When MDF is employed, the system analysis equations of Eq. (2.2) may be solved with iterative methods such as Newton-Raphson or FPI [28]. FPI is regularly employed as the analysis tool for the MDF formulation. Due to its intuitive implementation, MDF is the most frequently utilized MDO strategy [25]. However, it should not be applied without recognition of its shortcomings. As an alternative to nesting FPI within an optimization algorithm, solution of some or all system analysis equations may be performed by the optimization algorithm, as is the case with IDF or AAO, which can alleviate difficulties

encountered with MDF implementations. This section reviews the nature of FPI, explores some issues with its use in MDF, and presents convergence conditions for FPI to aid intuition. These convergence conditions are foundational to understanding coupling strength, and important factor in P/C decisions. Haftka *et al.* [66] presented two definitions for coupling strength—the first accounting for the magnitude of inter-analysis derivatives, and the second for the relationship between these derivatives. This section, along with subsequent design examples, strengthens the position of the second definition. The concepts presented in this section help qualify assumptions used in later chapters concerning choice of problem formulation.

4.2.1 Fixed Point Iteration

A two-element coupled system is depicted in Fig. 4.4, which possesses feedback coupling, since a_2 depends on the output of a_1 and *vice versa*. Since \mathbf{x} is fixed during system analysis, it is omitted from the current discussion.

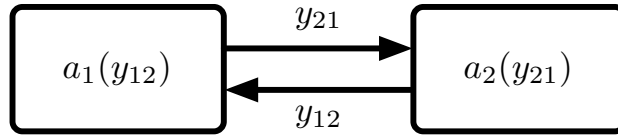


Figure 4.4 Two element coupled system

To employ FPI for system analysis, an initial guess is made for the input to the subsystem that is executed first, and the analyses are iteratively performed with updated coupling variable values until consistency is achieved, i.e., coupling variables match analysis outputs, satisfying Eq. (2.2). If the system meets certain criteria, this process will converge to a fixed point. The FPI algorithm for the two-dimensional example problem is [28]:

(Step 0) choose initial guess y_{12}^0 , set $k = 0$

(Step 1) $k = k + 1$

(Step 2) $y_{21}^k = a_1(y_{12}^{k-1})$

(Step 3) $y_{12}^k = a_2(y_{21}^k)$

(Step 4) if $\|\mathbf{y}^k - \mathbf{y}^{k-1}\| < \varepsilon$, then stop, otherwise go to **(Step 1)**.

When the stopping criterion in Step 4 is met, the system is epsilon-consistent, approximately satisfying Eq. (2.2). The norm in Step 4 is typically the Euclidian or infinity norm. Figure 4.5 illustrates the analysis space of a sample two-element system, which possesses

two fixed points at the intersections of the analysis functions. Following the algorithm above, FPI will converge to the fixed point \mathbf{y}_{PA} if y_{12}^0 is near this point, but never to \mathbf{y}_{PB} for any y_{12}^0 .

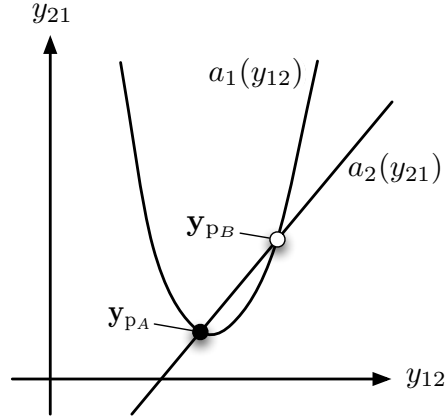


Figure 4.5 System with multiple fixed points

Studying this result, it can be seen graphically that if the line traced by a_{12} is steeper than the line traced by a_{21} in the neighborhood of a fixed point, then FPI will converge to that fixed point. This observation agrees with the well-known necessary and sufficient conditions for FPI convergence [71]:

$$\left(\frac{\partial a_{21}(y_{12})}{\partial y_{12}} \right)^{-1} < \frac{\partial a_{12}(y_{21})}{\partial y_{21}} \quad (4.4)$$

The derivatives in Eq. (4.4), in normalized form, are used by Rogers and Bloebaum [114] to quantify coupling strength between subsystems. Intuitively, higher sensitivity between subsystems will require more iterations during analysis. When the specific solution algorithm is FPI, however, computational effort depends instead on the relationship between these sensitivities. For example, if the relation in Eq. (4.4) is satisfied but is near equality, convergence will require numerous iterations, and will cycle without convergence if the inequality becomes equality. If we define coupling strength as the effort required to bring a coupled system into a consistent state, rather than just the influence that the subsystems have on each other, then coupling strength is more aptly quantified through a comparison of derivative values than through absolute derivative magnitudes.

4.2.2 Example: Hidden Optima

Consider the IDF formulation of the two-element system optimization problem given in Eq. (4.5).

$$\begin{aligned}
 \min_{\mathbf{x}, \mathbf{y}} \quad & f(\mathbf{x}, \mathbf{y}) = y_{12}^2 - 100y_{21} + 0.1\mathbf{x}\mathbf{x}^T & (4.5) \\
 \text{subject to} \quad & \mathbf{h}_{\text{aux}}(\mathbf{x}, \mathbf{y}) = \mathbf{y} - \mathbf{a}(\mathbf{x}, \mathbf{y}) = \mathbf{0} \\
 \text{where} \quad & a_{21}(y_{12}, x_1) = \phi_1(x_1)(y_{12} - \alpha_1)^2 \\
 & a_{12}(y_{21}, x_2) = \phi_2(x_2)y_{21} + \alpha_2 \\
 & \phi_1(x_1) = \frac{0.25}{1 + e^{x_1}} + 0.5 \\
 & \phi_2(x_2) = -\left(\frac{1}{1 + e^{x_2}} + 0.5\right) \\
 & \alpha_1 = 3, \quad \alpha_2 = 3.5
 \end{aligned}$$

For any $\mathbf{x} \in \mathbb{R}^2$, two fixed points exist, similar to the system in Fig. 4.5. FPI is capable of finding only a point with small y_{21} and large y_{12} , which is a local optimum. The second fixed point has the reverse properties, and leads to the global optimum. Even when started at the global optimum, the MDF implementation moves toward the inferior local optimum $f(\mathbf{x}_{MDF}^*) = -0.244$ at $\mathbf{x}_{MDF}^* = [-1.902, 2.273]$. The IDF implementation finds the global optimum $f(\mathbf{x}_{IDF}^*) = -975.692$ at $\mathbf{x}_{IDF}^* = [5.824, 7.754]$. It can be shown using Eq. (4.4) that MDF implemented using FPI is incapable of finding the global optimum. Physically meaningful models can also exhibit such behavior. Note that the MDF and IDF implementations solve identical design problems; the different solutions follow from limitations of FPI. The MDF formulation is indirectly limited by its dependence on available system analysis tools.

Although the optimization space for IDF is more complex, the problem design space can be explored more effectively. When the $f(\mathbf{x})$ response surface (as computed with FPI) is visualized graphically, only a single optimum is seen at \mathbf{x}_{MDF}^* . Since the IDF optimization space is in \mathbb{R}^4 , the objective function cannot be visualized easily. One approach is to plot the objective function along the line that connects the MDF and IDF solutions, i.e., plot $f(\lambda) = f(\lambda[\mathbf{x}_{IDF}^*, \mathbf{y}_{IDF}^*] + (1 - \lambda)[\mathbf{x}_{MDF}^*, \mathbf{y}_p(\mathbf{x}_{MDF}^*)])$, where \mathbf{y}_{IDF}^* is the coupling variable vector at the IDF solution, and $\mathbf{y}_p(\mathbf{x}_{MDF}^*)$ is the coupling variable fixed point computed by FPI at \mathbf{x}_{MDF}^* . The auxiliary constraints can be included in the visualization by adding a penalty for constraint violation to the objective function: $f'(\lambda) = f(\lambda) + 500\|\mathbf{h}_{\text{aux}}(\lambda)\|_2^2$, where $\mathbf{h}_{\text{aux}}(\lambda) = \mathbf{h}_{\text{aux}}(\lambda[\mathbf{x}_{IDF}^*, \mathbf{y}_{IDF}^*] + (1 - \lambda)[\mathbf{x}_{MDF}^*, \mathbf{y}_p(\mathbf{x}_{MDF}^*)])$. Figure 4.6 illustrates how using IDF can reveal optima that are hidden to MDF. For the points represented in this

plot, the auxiliary constraint violation is zero only at the MDF and IDF solutions. It is hypothesized that other methods that employ simultaneous analysis and design, such as analytical target cascading [79], share this desirable behavior with IDF.

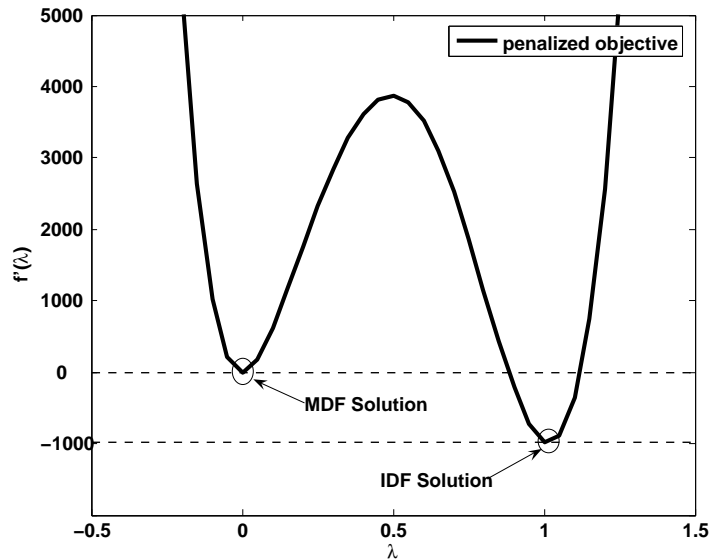


Figure 4.6 IDF optimization space visualization

Thus, although FPI implementation is straightforward, it presents several difficulties: FPI may not converge to an analysis solution; if multiple solutions exist, FPI may not find them all; the sequential nature of FPI prevents the parallel execution of analyses. When FPI is used as the system analysis tool for MDF, all of these same issues arise. The optimization problem may not converge, and when it converges the globally optimal solution may not be found. In addition, the resulting nested optimization and analysis process can be inefficient. These algorithmic considerations are critical factors in making problem formulation decisions.

This section explored issues associated with FPI, established the ability of IDF to find ‘hidden’ optima, and laid a foundation for the understanding of coupling strength. A demonstration of how coupling strength influences the computational performance of MDF and IDF implementations follows. These results combine to justify the use of IDF-type formulations in later chapters.

4.2.3 Coupling Strength in Single-Level Formulations

Two examples were detailed in Chapter 3 that allow modification of coupling strength. With these examples we can test how both the MDF and IDF formulations respond to changes in coupling strength. MDF is shown to be sensitive to changes in coupling strength, while IDF is not. This property makes IDF the preferred subproblem formulation type for P/C decision

models presented in later chapters.

Air-flow Sensor Design Problem

The first variable coupling-strength example is the vane air-flow sensor design problem from Section 3.1. It consists of coupled structural and aerodynamic analyses. The coupling variables are the stator deflection ($\tilde{\theta}$) and the drag force (\tilde{F}). The analysis structure is illustrated in Fig. 4.7.

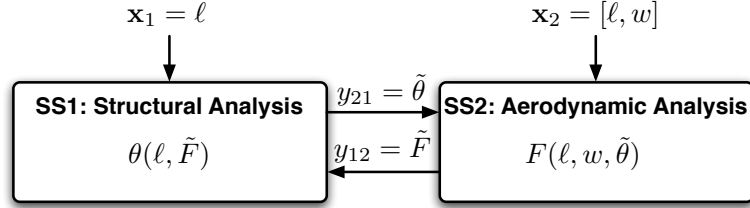


Figure 4.7 Coupling relationship in airflow sensor analysis

The objective in this design problem is to adjust the stator geometry such that the stator deflection is as close as possible to a target deflection ($\hat{\theta}$) for a given airspeed. MDF formulation is:

$$\begin{aligned}
 & \min_{\ell, w} && (\theta - \hat{\theta})^2 \\
 & \text{subject to} && \tilde{F} - F_{\max} \leq 0 \\
 & && \ell w - A = 0
 \end{aligned} \tag{4.6}$$

Consistent values for F and θ are obtained using FPI. The exact solution can be obtained using monotonicity analysis (MA). The MDF solution matches the MA solution described in Eq. (3.5). The IDF solution requires the addition of $\tilde{\theta}$ and \tilde{F} to the decision variable set, as well as auxiliary constraints on these values, as shown in Eq. (4.7). The IDF solution also matches the MA solution.

$$\begin{aligned}
 & \min_{\ell, w, \tilde{\theta}, \tilde{F}} && (\tilde{\theta} - \hat{\theta})^2 \\
 & \text{subject to} && \tilde{F} - F_{\max} \leq 0 \\
 & && \ell w - A = 0 \\
 & && \tilde{\theta} - \theta(\ell, \tilde{F}) = 0 \\
 & && \tilde{F} - F(\ell, w, \tilde{\theta}) = 0
 \end{aligned} \tag{4.7}$$

If the spring constant k in Eq. (3.2) is large, the plate deflection will be small, resulting in only minor changes to the frontal area and drag force. Quantitatively, increasing k will reduce $\partial F(\ell, w, \tilde{\theta})/\partial \tilde{\theta}$, but not affect $\partial \theta(\ell, \tilde{F})/\partial \tilde{F}$, resulting in reduced coupling strength between analyses. Conversely, small k results in high coupling strength; as verified experimentally, the number of iterations required for FPI convergence increases with decreasing k . Consequently, the computational expense of the MDF implementation is expected to increase with decreasing k . IDF eliminates the need to converge to consistent analysis results at points far from the optimal solution, but incurs its own computational overhead due to increased problem dimension. The value of k was varied from 0.01 to 0.20 N/rad, and the MDF and IDF computation times (on a 3.4 GHz Pentium 4 PC) were recorded. The result, displayed in Fig. 4.8, reveals that MDF does incur more computational expense with small values of k , as expected, while IDF is only slightly sensitive to changes in coupling strength. The MDF and IDF solutions agreed within 0.01% over the specified stiffness range.

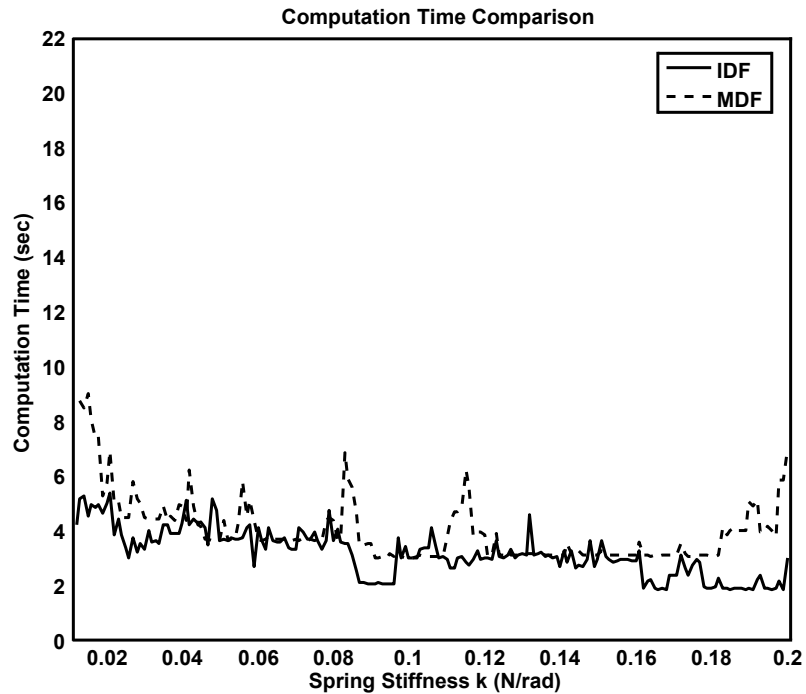


Figure 4.8 Comparison of MDF and IDF solution time as a function of coupling strength

Figure 4.9 compares the number of function evaluations required for the MDF and IDF implementations. A function evaluation is defined as the calculation of both structural and aerodynamic outputs, including calculations required for finite differencing. Since the analysis expense for this example is low, the additional computational overhead required for the IDF implementation is a significant factor in solution time. With respect to function evaluations, IDF solution expense truly is insensitive to coupling strength. It is also clear

from this plot that the noise displayed in Fig. 4.8 is purely computational. The next design example requires more analysis time, which is large compared to computational noise, resulting in a smoother plot of solution time.

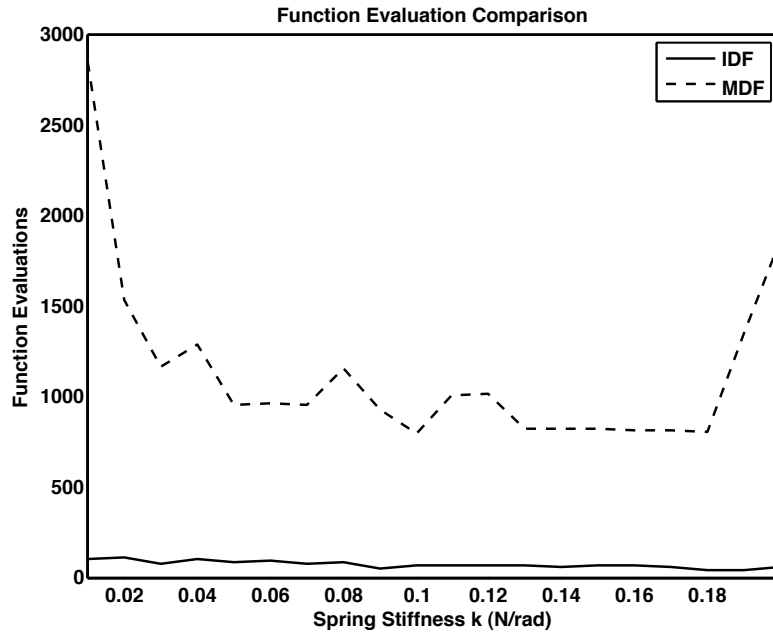


Figure 4.9 Comparison of MDF and IDF function evaluations as a function of coupling strength

At stiffness values much larger than the range displayed in Fig. 4.9, the design problem becomes infeasible since the equilibrium plate deflection is low enough that the resulting large frontal area incurs drag force values that exceed F_{\max} . It is interesting to note that in this infeasible domain MDF satisfies the drag force constraint and violates the area constraint, while IDF exhibits the converse. This phenomenon comes about because MDF finds consistent values for $\tilde{\theta}$ and \tilde{F} at each optimization iteration, while IDF does not. IDF is free to choose an infeasible \tilde{F} in order to satisfy the area constraint, but MDF does not have this flexibility. As stiffness is increased and the design problem approaches infeasibility, the MDF computation time increases as observed in the plot. At stiffness values below the displayed range MDF fails due to excessive coupling strength. In this design example MDF time increases with coupling strength due to excessive analysis effort, and increases with decreasing coupling strength due to excessive optimization effort for narrowly feasible design problems.

Turbine Blade Design Problem

The second example used to demonstrate the influence of coupling strength on MDF and IDF is the turbine blade design problem from Section 3.2. The objective is to find blade

geometry that minimizes heat lost through the blade into the turbine rotor (q), subject to temperature, geometric compatibility, stress, and mass constraints. The MDF formulation is:

$$\begin{aligned}
 & \min_{w,t} && q(w,t,\tilde{L}) \\
 & \text{subject to} && T(w,t,\tilde{L},x) - T_{\text{melt}} \leq 0 \\
 & && \delta_{\text{total}}(\tilde{T}(x)) - \delta_{\text{allow}} \leq 0 \\
 & && \sigma_{\text{a}}(\tilde{L},x) - \sigma_{\text{r}}(\tilde{T}(x),x) \leq 0 \\
 & && \sigma_{\text{b}}(t,\tilde{L},x) - \sigma_{\text{r}}(\tilde{T}(x),x) \leq 0 \\
 & && m(w,t) - m_{\text{max}} \leq 0 \\
 & && \text{and } 0 \leq x \leq L_0 + \delta_{\text{total}}(\tilde{T}(x)).
 \end{aligned} \tag{4.8}$$

The objective and constraint functions are computed by two coupled analysis: thermal and structural. The coupling variables are the blade temperature profile ($\tilde{T}(x)$) and the elongated blade length (\tilde{L}). Figure 4.10 illustrates the analysis structure for the turbine blade design problem.

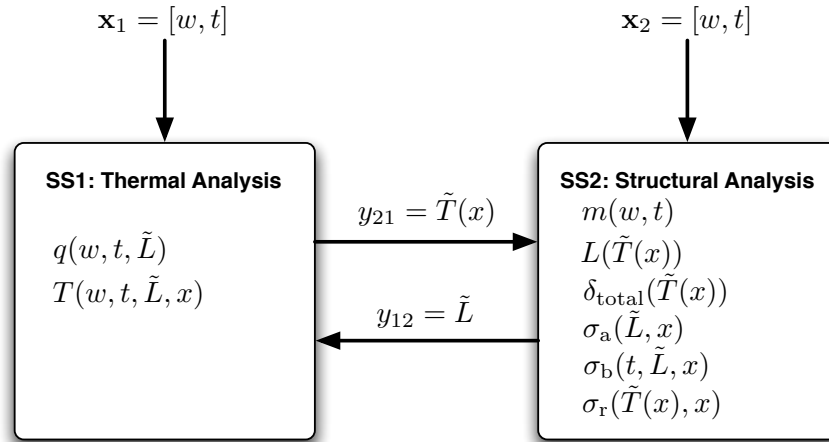


Figure 4.10 Turbine blade coupling and functional relationships

The IDF formulation is constructed by adding the coupling variables to the set of decision variables, and including auxiliary equality constraints on the coupling variables to ensure consistency:

$$\begin{aligned}
& \min_{w,t,\tilde{T}(x),\tilde{L}} && q(w,t,\tilde{L}) \\
\text{subject to} &&& T(w,t,\tilde{L},x) - T_{\text{melt}} \leq 0 \\
&&& \delta_{\text{total}}(\tilde{T}(x)) - \delta_{\text{allow}} \leq 0 \\
&&& \sigma_{\text{a}}(\tilde{L},x) - \sigma_{\text{r}}(\tilde{T}(x),x) \leq 0 \\
&&& \sigma_{\text{b}}(t,\tilde{L},x) - \sigma_{\text{r}}(\tilde{T}(x),x) \leq 0 \\
&&& m(w,t) - m_{\text{max}} \leq 0 \\
&&& \tilde{T}(x) - T(w,t,\tilde{L},x) = 0 \\
&&& \tilde{L} - L(\tilde{T}(x)) = 0 \\
&&& \text{and } 0 \leq x \leq L_0 + \delta_{\text{total}}(\tilde{T}(x)).
\end{aligned} \tag{4.9}$$

The function-valued coupling variable $\tilde{T}(x)$ was discretized and implemented as a vector-valued coupling variable, substantially increasing the optimization problem dimension.

The computation time required for both MDF and IDF solutions was recorded over a range of coupling strength levels, varied by adjusting the modulus of elasticity $E(T)$. A more compliant blade results in increased blade elongation and exposed surface area, increasing the impact that the structural analysis results have on the thermal analysis. The $E(T)$ curve from Eq. (3.17) was multiplied by a scaling factor to produce changes in coupling strength. Figure 4.11 illustrates the dependence of MDF and IDF computation time on this modulus multiplier, and hence the dependence on coupling strength.

As with the previous example, IDF computation time is insensitive to coupling strength, while MDF computation time increases with coupling strength. At modulus multiplier values larger than the range illustrated, very little change in computation time was observed. In contrast to the previous example, increased stiffness does not induce infeasibility, but does result in very weak coupling as expected. Since design infeasibility is not a confounding factor as in the VAF example, MDF time monotonically increases with coupling strength. A very stiff blade results in effectively independent analyses—only one or two FPI iterations are required for system analysis. At modulus multiplier values smaller than the range presented, MDF failed due to strong coupling.

In summary, weakly coupled systems with relatively few design variables can be solved efficiently with MDF, while strongly coupled systems require excessive iterations for the inner analysis loops of MDF. The computation time required for the IDF approach is virtually constant for all levels of coupling strength investigated here. In addition to verifying the predictions of IDF efficiency [35] for the case of a strongly coupled system, Figs. 4.8 and 4.11 show a very clear relationship between coupling strength and computational performance.

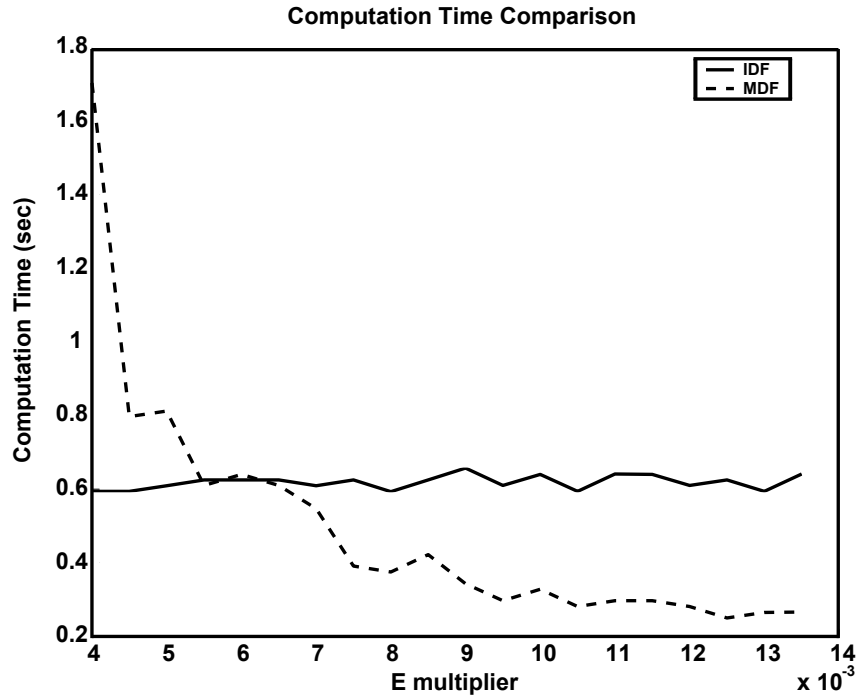


Figure 4.11 Comparison of MDF and IDF solution time as a function of coupling strength

An ideal P/C decision method would account for the strength, or nature, of coupling relationships. Unfortunately, such a method requires computing derivatives like those given in Eq. (4.4) over the entire system design and analysis space. Approximations must be made for a P/C decision method to be practical. The methods described in following chapters are based on the existence of coupling relationships, rather than nature. Existence is represented using the reduced adjacency matrix.

4.3 Multi-Level Formulations

The formulations for decomposition-based design optimization presented thus far utilize a single optimization problem, and are referred to as single-level methods. This section presents another class of methods that employ multiple optimization problems, termed multi-level methods. After a system is partitioned, an optimization problem is formulated for each partition block, forming a set of subproblems. The burden of optimization is distributed across the system. Another implication of distributed optimization in multi-level formulations is that shared variable consistency is not automatically satisfied; as with single-level formulations the coupling variable consistency constraints of Eq. (2.2) must be satisfied, but so must the shared variable consistency constraints of Eq. (2.1). As described in Chapter

1, solving each of these subproblems independently will not produce a design optimal for the entire system. Interactions between subproblems must be accounted for. These interactions are quantified using linking variables. A coordination algorithm guides the repeated solution of subproblems toward a consistent and optimal system design. This section provides an overview of multilevel methods, and presents two multilevel formulations in detail: analytical target cascading (ATC) and augmented Lagrangian coordination (ALC).

4.3.1 Classes of Multi-Level Formulations

Important distinctions between multi-level methods relate to how subproblem solutions are coordinated and how subproblem interactions are managed. A classical approach to coordinating a partitioned optimization problem is to use a master optimization problem to provide information to the subproblems in a nested manner. Each subproblem is linked directly to the master problem, but not directly to other subproblems. Subproblems are all at the 'lower level', and the master problem is at the top. This coordination approach is sometimes called bi-level nested.

The first formulations for bi-level nested coordination involved linear optimization problems. For example, the Dantzig-Wolfe decomposition method uses a dual problem for the master problem and a primal for each subproblem [38]. It is designed to work with the simplex method for linear programming [37], and accommodates shared design constraints. Wagner provided an extensive review of bi-level nested formulations [145]. Several other formulations in this category have been developed, including several for general nonlinear problems. Sobieski and Haftka reviewed several important multi-level formulations, including collaborative optimization (CO) [25] and concurrent subspace optimization (CSSO) [126]. In CO the master problem seeks to minimize the system objective function by varying targets for linking variables, and requires that the subproblem values match corresponding targets using equality constraints. Theoretical problems with the original CO formulation have been identified [4], and several modifications have been proposed [41, 42, 92, 125]. CSSO takes a different approach to managing subproblem interactions; each subproblem is solved using an approximation for other subproblems that is constructed using global sensitivity equations (GSEs). CSSO also allows design constraint violations at intermediate steps. In contrast, CO allows consistency constraint violation at intermediate steps, but enforces feasibility of design constraints. Bi-level system synthesis (BLISS) is another bi-level nested formulation that uses sensitivity-based subproblem approximations when solving subproblems, and allows consistency constraint violation until system convergence [127]. Haftka and Watson introduced a bi-level formulation that

allows temporary design constraint violation, and seeks to minimize the maximum design constraint violation until it reaches zero at convergence [67].

Bi-level formulations have proven useful, and some even have convergence proofs [67], but do have limitations. The dimension of the master problem may become unmanageable as the number of subproblems and interactions increases. Bi-level formulations force a two-level hierarchical problem structure, which may not be ideal for certain problem types.

Another class of formulations addresses these problems by allowing more than two levels in a problem structure, or even non-hierarchical problem structures. Penalty relaxation methods are used to allow violation of consistency constraints until system convergence. Two formulations belong to this class. Kim et al. introduced analytical target cascading (ATC) as a product development tool for determining consistent subsystem target values [79, 80]. ATC has since been applied as a formulation for decomposition-based design optimization in simulation based design. ATC requires that subproblems are linked in a purely hierarchical structure. It has been shown that this structure requirement can be relaxed to accommodate non-hierarchical links. Tosserams et al. developed a non-hierarchical generalization, called Augmented Lagrangian Coordination (ALC) [141, 142]. Both ATC and ALC have been proven to converge under standard nonlinear programming assumptions, such as local convexity [106, 141]. The ATC and ALC formulations will be presented in detail below. The partitioning and coordinations decision methods presented in later chapters assume that the problem formulation belongs to this class.

4.3.2 Analytical Target Cascading

Analytical target cascading was developed based on needs in the automotive industry to translate top-level product targets into detailed design specifications. It is applicable to systems that possess hierarchical relationships. An example of the analysis relationships in a hierarchical system is shown in Fig. 4.12. Each element in the hierarchy computes its own local analysis responses, and may require as inputs analysis responses (coupling variables) from lower level elements, in addition to local and shared variables. ATC can handle systems with any number of levels.

The analysis structure displayed in Fig. 4.12 depicts unidirectional information flow. Original ATC applications possessed this structure, and initial ATC formulations did not account for the possibility of feedback coupling. Design variables shared between subproblems with a common parent element are allowed. More recent ATC formulations [7, 140] allow multidirectional coupling, and coupling between same-level elements.

The objective of the ATC process is to determine design specifications for each element

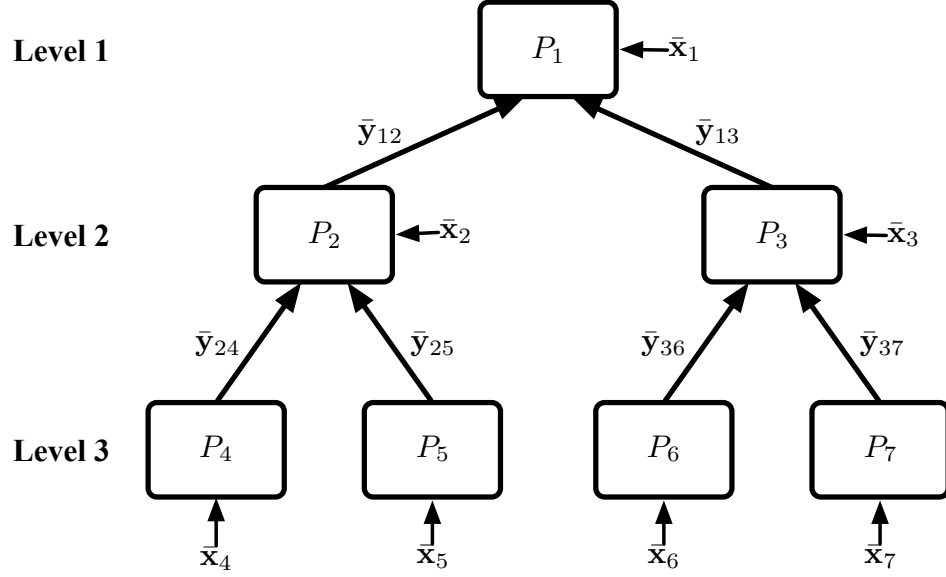


Figure 4.12 Hierarchical system analysis structure

in the hierarchy that account for interaction so that design teams can proceed with detail design independently. An optimization problem is formulated for each element. The formulation allows for a local objective and observes local design constraints. ATC allows the optimization algorithm to choose coupling variable values, and uses penalty functions to ensure system consistency. The ATC formulation for subproblem P_i is

$$\begin{aligned}
 & \min_{\bar{\mathbf{x}}_i, \bar{\mathbf{y}}_i, \bar{\mathbf{x}}_{s\mathcal{C}_i}, \bar{\mathbf{y}}_{\mathcal{C}_i}} && f_i(\bar{\mathbf{x}}_i, \bar{\mathbf{y}}_i) + \phi(\mathbf{c}_i(\bar{\mathbf{x}}_i, \bar{\mathbf{y}}_i, \bar{\mathbf{x}}_{s\mathcal{C}_i}, \bar{\mathbf{y}}_{\mathcal{C}_i})) \\
 & \text{subject to} && \mathbf{g}_i(\bar{\mathbf{x}}_i, \bar{\mathbf{y}}_i) \leq \mathbf{0} \\
 & && \mathbf{h}_i(\bar{\mathbf{x}}_i, \bar{\mathbf{y}}_i) = \mathbf{0}
 \end{aligned} \tag{4.10}$$

If P_i has child elements with shared design variables $\bar{\mathbf{x}}_{s\mathcal{C}_i}$ or coupling variables between them $\bar{\mathbf{y}}_{\mathcal{C}_i}$, the optimization problem for P_i sets target values for these quantities to be met by the child elements. The vector of consistency constraints is \mathbf{c}_i , which ensures that shared and coupling variables between child elements are consistent at ATC convergence, and that values for shared and coupling variables for P_i are consistent with targets set by its parent at ATC convergence. Calculation of f_i , \mathbf{c}_i , and the design constraints local to P_i (\mathbf{g}_i and \mathbf{h}_i), normally requires execution of the analysis functions belonging to P_i . ATC allows for the possibility that more than one subproblem has an objective function. If f_i is the local objective function for subproblem P_i , then the equivalent objective function for the entire system is $f(\bar{\mathbf{x}}, \bar{\mathbf{y}}) = \sum_{i=1}^N f_i(\bar{\mathbf{x}}_i, \bar{\mathbf{y}}_i)$.

The consistency constraints in the ATC subproblem formulation are relaxed using an

inexact penalty function $\phi(\mathbf{c}_i)$. Feasibility problems would occur during the ATC process if $\mathbf{c}_i = \mathbf{0}$ was instead cast as an equality constraint in the subproblem formulation. Several different penalty relaxation methods have been used in ATC implementations. The first was a simple quadratic penalty function where each component of the consistency constraint is multiplied by an importance weight, and then the squared Euclidean norm is taken of the resulting vector:

$$\phi(\mathbf{c}_i) = \|\mathbf{w}_i \circ \mathbf{c}_i\|_2^2 \quad (4.11)$$

Penalty weights were typically chosen subjectively based on experience. Large weights place high emphasis on consistency, but struggle to achieve system optimality. In addition, large weights lead to ill conditioning of subproblems and slow convergence. If weights are set too low the system may never achieve an acceptable level of consistency. Michalek and Papalambros developed a formal method for selecting quadratic penalty function weights that is based on derivatives obtained during the ATC solution process [102]. This approach is particularly useful when top-level product targets are unattainable ‘stretch’ targets.

Michalek’s penalty update method consists of an inner loop and an outer loop. The inner loop uses the ATC coordination process to minimize both the system objective function and system inconsistency. The balance between these two objectives is dictated by the penalty weights, which are fixed during an inner loop solution. An iteration of the outer loop involves an inner loop execution followed by a weight update calculation. The new weights are then used in the next inner loop execution. The outer loop process repeats until an acceptable level of consistency is achieved.

The inner loop process will now be described in more detail. It is helpful to view subproblems as optimal value functions. Figure 4.13 illustrates ATC subproblem P_i and its inputs and outputs. The quantities passed from the parent subproblem P_j to P_i are the targets \mathbf{t}_{ij} . These may be targets for shared or coupling variables between P_i and sibling subproblems, or quantities relating to a coupling relationship directly between P_i and P_j . The quantities passed from the P_i to P_j are the responses \mathbf{r}_{ji} to the targets \mathbf{t}_{ij} . Subproblem P_i can function as a parent as well, sending targets \mathbf{t}_{ki} to its subproblem P_k , and receiving the corresponding responses \mathbf{r}_{ik} . P_i may also have targets-response pairs with other child subproblems.

The inputs to P_i in the ATC process are \mathbf{t}_{ij} and \mathbf{r}_{ik} . The outputs \mathbf{t}_{ki} and \mathbf{r}_{ji} are functions of these inputs. Each subproblem thus can be considered an optimal value function. The collection of all ATC subproblems in a system comprises a set of coupled equations. A solution to the ATC process satisfies this system of equations, and can be obtained using an appropriate algorithm. This algorithm is the ATC coordination algorithm. Some form of fixed point iteration has been the nearly ubiquitous choice for ATC coordination algorithm.

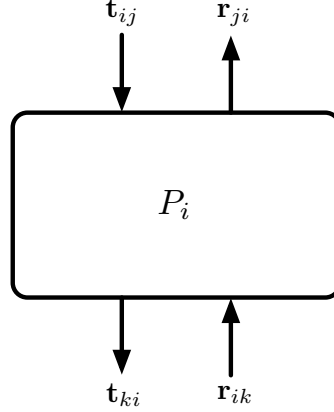


Figure 4.13 ATC subproblem as an optimal value function

The exception is a second order coordination algorithm proposed by Tosserams [139].

Improvements in the ATC penalty relaxation method have led to a dramatic boost to convergence rate. Tosserams et al. incorporated the augmented Lagrangian penalty function into the ATC formulation, along with the method of multipliers weight update strategy [140]. The penalty function includes both a linear and quadratic term:

$$\phi(\mathbf{c}_i) = \mathbf{v}_i \mathbf{c}_i^T + \|\mathbf{w}_i \circ \mathbf{c}_i\|_2^2 \quad (4.12)$$

The penalty weights for the linear and quadratic terms are \mathbf{v}_i and \mathbf{w}_i , respectively. At every outer loop iteration the following formulae are used to update the penalty weights:

$$\mathbf{v}^{k+1} = \mathbf{v}^k + 2\mathbf{w}^k \circ \mathbf{w}^k \circ \bar{\mathbf{c}}^k \quad (4.13)$$

$$w_i^{k+1} = \begin{cases} w_i^k & \text{if } |\bar{c}_i^k| \leq \gamma |\bar{c}_i^{k-1}| \\ \beta w_i^k & \text{if } |\bar{c}_i^k| > \gamma |\bar{c}_i^{k-1}| \end{cases} \quad i = 1, 2, \dots, n^c \quad (4.14)$$

where \mathbf{c} is the vector of all n^c consistency constraints in the ATC problem formulation, \mathbf{v} and \mathbf{w} are the corresponding vectors of linear and quadratic penalty weights, respectively, and superscripts indicate outer loop iteration number. The constant β controls how quickly the quadratic weights increase, and typically $1 < \beta < 3$ depending on coordination algorithm details and problem nature. The threshold γ specifies how much $\bar{\mathbf{c}}$ must improve before each w_i is updated. This approach is known as the method of multipliers [20].

Consistency is achieved if either the quadratic weights approach infinity, or if the linear weights converge to the consistency constraint Lagrange multipliers. The method of multipliers ensures the linear weights converge to the Lagrange multipliers, and if β is not set

too high ill conditioning is avoided. Convergence using a quadratic penalty alone frequently requires hundreds or thousands of outer loop iterations. The augmented Lagrangian penalty relaxation method along with the method of multipliers converges much more quickly in practice, sometimes in ten or fewer outer loop iterations [8]. In addition, the data required for Eqs. 4.13 and 4.14 are more easily obtained than the derivatives required for Michalek's update method.

Several coordination algorithm variants are possible. Rather than solving the inner loop exactly at every outer loop iteration, we can terminate the inner loop fixed point iteration solution when the system is consistent within a loose tolerance[140]. This approach is known as the inexact method of multipliers (INMOM), whereas solving the inner loop to within a very tight tolerance is called the exact method of multipliers (ENMOM). In another coordination alternative only a single pass of subproblem solutions is completed for every outer loop iteration. This option is known as the alternating directions method of multipliers (ADMOM). A smaller value for β must be employed with ADMOM; otherwise quadratic weights will increase too quickly. ADMOM has exhibited the greatest efficiency of any approach, but does not fit within the assumptions required for the ATC convergence proof. In practice ADMOM works well for most problems.

A modification of the ADMOM approach exploits the hierarchical structure of ATC. Subproblems in each level are linked only to subproblems in adjacent levels. Subproblems in odd levels do not depend directly on outputs from other subproblems in odd levels. Therefore, all odd-level subproblems may be solved in parallel without having to use subproblem responses from a previous iteration. The subproblems in even levels may then be executed simultaneously. We alternate between solving all odd level subproblems in parallel and all even level subproblems in parallel. This odd-even coordination approach facilitates an easily implemented parallel ATC solution process. Another possibility for a coordinating subproblems solved in parallel is Jacobi iteration [28]. Jacobi iteration is similar to fixed-point iteration, but when evaluating each subproblem we only use subproblem responses from the previous iteration. This way all subproblems may be solved in parallel. The advantage of parallelism must be weighed against the slower convergence of Jacobi iteration [21].

4.3.3 Example: Aircraft Family Design

The aircraft family design problem, introduced in Section 3.3, is used here to illustrate implementation of ATC. The objective of the aircraft family problem is to determine the design of two distinct aircraft used in a commercial airline fleet. Aircraft A is intended for

moderate range missions with up to 296 passengers. Aircraft B is intended for long range missions with up to 259 passengers. The main wing design is common between the aircraft, but all other components can be designed uniquely. The objective is to minimize a weighted average of p_A and p_B , the ticket prices for each aircraft, subject to performance constraints.

The problem is formulated as a bi-level ATC formulation with three subproblems. The top level problem P_1 seeks to attain agreement between the lower-level subproblems with respect to shared variables, while minimizing the system objective function f . The two lower-level problems, P_2 and P_3 , seek to match targets set by P_1 , while meeting local design performance constraints. P_2 corresponds to the design of aircraft A , and P_3 corresponds to the design of aircraft B . This system has no feedback, simplifying the ATC formulation. Many other options for partitioning and formulation this problem exist, but are not addressed here. In this partition each subproblem contains only one analysis function, so the sets of external linking variables are equivalent to linking variables, and this is reflected in the notation. For clarity in the ATC formulations, a superscript in parentheses indicates the subproblem in which a value is computed. Problem P_1 is formulated as:

$$\min_{\mathbf{x}_s^{(1)}, p_A^{(1)}, p_B^{(1)}} f(p_A^{(1)}, p_B^{(1)}) + \phi(\mathbf{c}_1) \quad (4.15)$$

$$\text{where: } \mathbf{c}_1 = [\mathbf{x}_s^{(1)} \ \mathbf{x}_s^{(1)} \ p_A^{(1)} \ p_B^{(1)}] - [\mathbf{x}_s^{(2)} \ \mathbf{x}_s^{(3)} \ p_A^{(2)} \ p_B^{(3)}]$$

The deviation vector \mathbf{c}_1 quantifies the difference between the targets set by P_1 and the achievable responses of P_2 and P_3 . The responses are fixed parameters with respect to P_1 . Note that f is a function only of target cost metrics, since these are independent decision variables in P_1 . The penalty function $\phi(\mathbf{c}_1)$ helps guide the ATC process toward consistency. The linear and quadratic penalty weights, \mathbf{v}_1 and \mathbf{w}_1 , are updated using the method of multipliers.

In this case the only analysis associated with P_1 is the simple weighted average system objective function. When combined with the penalty function the P_1 objective is a quadratic function, enabling direct solution without the use of an optimization algorithm. In addition, P_1 has no design constraints, and therefore can be solved by finding $\bar{\mathbf{x}}_1$ such that $\nabla_{\bar{\mathbf{x}}_1} f_1 = \mathbf{0}$, where $f_1 = f + \phi(\mathbf{c}_1)$. Problem P_2 is formulated as:

$$\begin{aligned}
& \min_{\mathbf{x}_s^{(2)}, \mathbf{x}_{\ell A}^{(2)}} && \phi(\mathbf{c}_2) \\
\text{subject to} &&& \mathbf{g}_A(\mathbf{x}_s^{(2)}, \mathbf{x}_{\ell A}^{(2)}) \leq \mathbf{0} \\
\text{where:} &&& \mathbf{c}_2 = \begin{bmatrix} \mathbf{x}_s^{(2)} & p_A^{(2)} \end{bmatrix} - \begin{bmatrix} \mathbf{x}_s^{(1)} & p_A^{(1)} \end{bmatrix}
\end{aligned} \tag{4.16}$$

The formulation of Problem P_3 is similar (one has simply to replace subscript or superscript 2 with 3 and subscript A with B). The ADMOM approach was used to solve the ATC problem.

The ATC subproblems P_2 and P_3 were solved using NOMADm [1], an implementation of mesh adaptive direct search [2, 14]. This algorithm effectively handled the non-smooth responses of the PASS analysis software. The mesh tolerance used in determining NOMADm convergence was 0.001, and subproblem optimizations typically required between 400 and 600 function evaluations. ATC required between 8 and 18 NOMADm optimizations to obtain a solution, depending on the value chosen for β in the penalty updates.

The P_1 subproblem objective function is quadratic, and required very little computational effort to solve. Two approaches were used to solve P_1 : solving for $\nabla_{\bar{\mathbf{x}}_1} f_1 = \mathbf{0}$ (where $f_1 = f + \pi(\mathbf{c})$), and using a gradient-based algorithm to minimize f_1 . The former was extremely efficient, but the latter proved more robust.

System consistency was quantified using the root mean square of the combined deviation vector

$$RMS(\mathbf{c}) = \sqrt{\frac{1}{|\mathbf{c}|} \mathbf{c} \mathbf{c}^T},$$

where

$$\mathbf{c} = [\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3], \quad |\mathbf{c}| = \text{cardinality of } \mathbf{c}.$$

The convergence of ATC was influenced strongly by the choice of β . Larger β values help force the system into tighter consistency, but can result in a stiff system that requires more iterations to converge. The problem was solved using a range of different β values to illustrate this influence. Figure 4.14 illustrates how larger values of β require more outer loop iterations. It was also observed that larger β values led to slightly larger objective function values, even when system consistency was approximately equal. This indicates that a stiff solution process can impede the identification of better designs.

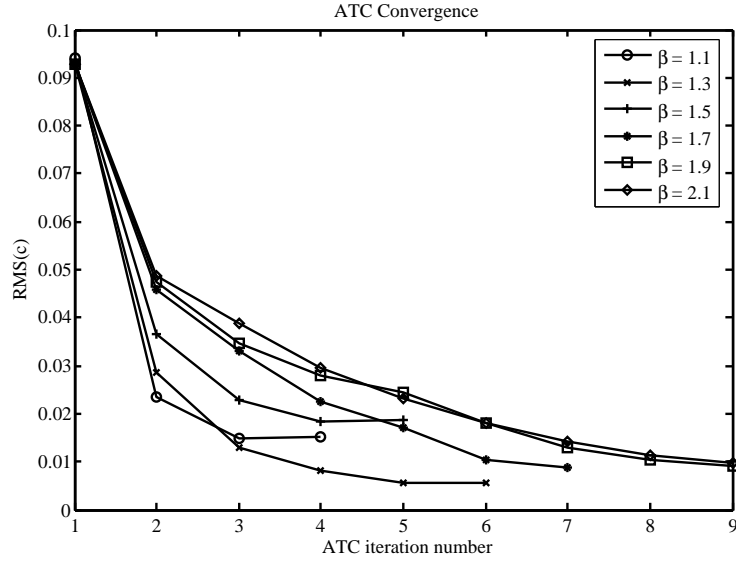


Figure 4.14 Influence of β on $RMS(c)$ (system consistency)

4.3.4 Augmented Lagrangian Coordination

The Augmented Lagrangian Coordination (ALC) formulation, also known as Augmented Lagrangian Decomposition, utilizes principles from ATC for the solution of a more general class of system design problems. The problem structure in ALC is not limited to hierarchical; any type of links between subproblems can be managed. Convergence properties are drawn from established nonlinear programming theory. ALC provides tremendous flexibility in problem linking structure, making it an ideal platform for studying the linking structure aspect of coordination decisions. ALC can handle linking functions in addition to linking variables. Linking functions depend on most or all design variables. This situation is uncommon in simulation-based design optimization, so accommodation of linking functions will not be included in the formulations here. ALC uses a double-loop coordination algorithm with augmented Lagrangian penalty relaxation, similar to ATC. The coordination algorithm variants described above, such as INMOM, ADMOM, and Jacobi iteration, all apply to ALC.

As with other distributed optimization formulations, local copies of linking variables must be used in ALC subproblems. These copies are allowed to vary independently, enabling independent subproblem solution, but are guided toward consistency using penalty functions. Internal shared design variables for subproblem i ($\hat{\mathbf{x}}_{si}$) do not require multiple copies since they are determined by the same optimization algorithm, and therefore require no consistency constraints. The internal coupling variables for subproblem i are $\hat{\mathbf{y}}_i$, and the corresponding set of analysis functions is $\hat{\mathbf{a}}_i(\bar{\mathbf{x}}_i, \hat{\mathbf{y}}_i, \bar{\mathbf{y}}_i)$, where $\bar{\mathbf{y}}_i$ are the external coupling

variables input to subproblem i and $\bar{\mathbf{x}}_i$ are the design variables for subproblem i . These values must be consistent. An MDF-type approach could be used here to satisfy internal consistency requirements with an analysis algorithm such as fixed point iteration. An alternative approach is to enforce internal consistency using auxiliary equality constraints:

$$\mathbf{c}_i(\bar{\mathbf{x}}_i, \hat{\mathbf{y}}_i, \bar{\mathbf{y}}_i) = \hat{\mathbf{y}}_i - \hat{\mathbf{a}}_i(\bar{\mathbf{x}}_i, \hat{\mathbf{y}}_i, \bar{\mathbf{y}}_i) = \mathbf{0} \quad (4.17)$$

This auxiliary equality constraint approach draws from the principles underlying the IDF formulation. Section 4.2 discussed how computational performance and robustness can be enhanced through using equality constraints instead of FPI to satisfy consistency requirements. The auxiliary constraints in Eq. 4.17 will be used in subsequent ALC formulations.

Definition of external linking variable consistency constraints is somewhat more involved. The copies of design variables shared between subproblems i and j , local to subproblem i , are $\bar{\mathbf{x}}_s^{ij}$. The coupling variables passed from subproblem j to i are $\bar{\mathbf{y}}_{ij}$, and the corresponding analysis functions are $\bar{\mathbf{a}}_{ij}(\bar{\mathbf{x}}_j, \hat{\mathbf{y}}_j, \bar{\mathbf{y}}_j)$. The external linking variables between subproblems i and j are $\bar{\mathbf{z}}_{ij} = [\bar{\mathbf{x}}_s^{ij}, \bar{\mathbf{y}}_{ij}]$. The external consistency constraints between subproblems i and j are:

$$\bar{\mathbf{c}}_{ij}(\bar{\mathbf{x}}_i, \bar{\mathbf{x}}_j, \hat{\mathbf{y}}_i, \hat{\mathbf{y}}_j, \bar{\mathbf{y}}_i, \bar{\mathbf{y}}_j) = [\bar{\mathbf{y}}_{ij} - \bar{\mathbf{a}}_{ij}(\bar{\mathbf{x}}_j, \hat{\mathbf{y}}_j, \bar{\mathbf{y}}_j), \bar{\mathbf{y}}_{ji} - \bar{\mathbf{a}}_{ji}(\bar{\mathbf{x}}_i, \hat{\mathbf{y}}_i, \bar{\mathbf{y}}_i), \bar{\mathbf{x}}_s^{ij} - \bar{\mathbf{x}}_s^{ji}] \quad (4.18)$$

Note that the components of $\bar{\mathbf{x}}_s^{ij}$ are part of the vector $\bar{\mathbf{x}}_i$, and $\bar{\mathbf{y}}_{ij}$ is part of the vector $\bar{\mathbf{y}}_i$. Certain variables that are input to the constraint function $\bar{\mathbf{c}}_{ij}$ are held fixed during subproblem solution, as indicated by the input arguments of the consistency constraint in the ALC subproblem formulation presented shortly.

The definitions above specify a very large number of consistency constraints for external shared variables. Only a subset of these constraints is required to ensure consistency. The number of possible ways to allocate consistency constraints is tremendous, and is a task beyond intuition for all but the smallest system design problems. Guidelines have been proposed for constructing bi-level or hierarchical ALC implementations [141, 142]. These recommendations are helpful, but do not capitalize on the potential benefit that could be realized through tailoring problem structure to meet the needs of a system. Chapter 7 develops the theory towards a rigorous, automated method for allocating consistency constraints.

After a set of consistency constraints is selected and allocated amongst the subproblems, an augmented Lagrangian penalty function is defined for the selected constraints on external linking variables:

$$\phi_{ij}(\bar{\mathbf{c}}_{ij}) = \mathbf{v}_{ij} \bar{\mathbf{c}}_{ij}^T + \|\mathbf{w}_{ij} \circ \bar{\mathbf{c}}_{ij}\|_2^2 \quad (4.19)$$

where \mathbf{v}_{ij} and \mathbf{w}_{ij} are vectors of penalty weights on the linear and quadratic terms, respectively, and \circ indicates the Hadamard product. Penalty weights are fixed for one or more inner loop executions, and are updated using the method of multipliers.

The set of indices for subproblems with external linking variables common to subproblem i is \mathcal{N}_i . The design inequality and equality constraints computed by analysis functions in subproblem i are \mathbf{g}_i and \mathbf{h}_i , respectively. The set of decision variables for subproblem i includes $\bar{\mathbf{x}}_i$, $\hat{\mathbf{y}}_i$, and $\bar{\mathbf{y}}_i$. The formulation of the optimization problem for subproblem i is:

$$\begin{aligned}
& \min_{\bar{\mathbf{x}}_i, \hat{\mathbf{y}}_i, \bar{\mathbf{y}}_i} && f_i(\bar{\mathbf{x}}_i, \hat{\mathbf{y}}_i, \bar{\mathbf{y}}_i) + \sum_{j \in \mathcal{N}_i | j > i} \phi_{ij}(\bar{\mathbf{c}}_{ij}(\bar{\mathbf{x}}_i, \bar{\mathbf{y}}_{ij}, \hat{\mathbf{y}}_i)) \\
& && + \sum_{j \in \mathcal{N}_i | j < i} \phi_{ji}(\bar{\mathbf{c}}_{ji}(\bar{\mathbf{x}}_i, \bar{\mathbf{y}}_{ij}, \hat{\mathbf{y}}_i)) \tag{4.20} \\
& \text{subject to} && \mathbf{g}_i(\bar{\mathbf{x}}_i, \hat{\mathbf{y}}_i, \bar{\mathbf{y}}_i) \leq \mathbf{0} \\
& && \mathbf{h}_i(\bar{\mathbf{x}}_i, \hat{\mathbf{y}}_i, \bar{\mathbf{y}}_i) = \mathbf{0}, \\
& && \mathbf{c}_i(\bar{\mathbf{x}}_i, \hat{\mathbf{y}}_i, \bar{\mathbf{y}}_i) = \hat{\mathbf{y}}_i - \hat{\mathbf{a}}(\bar{\mathbf{x}}_i, \hat{\mathbf{y}}_i, \bar{\mathbf{y}}_i) = \mathbf{0}
\end{aligned}$$

This formulation makes a distinction between shared and coupling variables, in contrast to the original ALC formulations [141, 142]. Neglecting this distinction requires an additional equality constraint for each analysis function that computes a coupling variable. Accounting for the distinction requires only one quantity to be passed for every coupling variable between subproblems, instead of two, and permits exploitation of coupling variable directionality when structuring an ALC implementation.

4.3.5 Example: Air Flow Sensor Design

The air flow sensor design from Section 3.1 is used here to illustrate ALC formulation and implementation. An ALC subproblem is formulated for the aerodynamic and structural aspects of the design problem. The structural subproblem is P_1 and the aerodynamic subproblem is P_2 . The formulation for the structural subproblem is:

$$\begin{aligned}
& \min_{\ell^{(1)}, F^{(1)}} && (\boldsymbol{\theta}^{(1)} - \hat{\boldsymbol{\theta}})^2 + \mathbf{v}_{12} \mathbf{c}_{12} + \|\mathbf{w}_{12} \circ \mathbf{c}_{12}\|_2^2 \\
& \text{subject to} && F^{(1)} - F_{max} \leq 0 \tag{4.21} \\
& \text{where} && \mathbf{c}_{12}(\ell^{(1)}, F^{(1)}) = [\boldsymbol{\theta}^{(1)} - \boldsymbol{\theta}^{(2)}, F^{(1)} - F^{(2)}, \ell^{(1)} - \ell^{(2)}]
\end{aligned}$$

Superscripts indicate the subproblem quantities are computed in. The constraint \mathbf{c}_{12}

was chosen arbitrarily for this implementation rather than \mathbf{c}_{21} . The stator deflection $\theta^{(1)}$ is computed using Eq. 3.2. The value of $\theta^{(2)}$ is a fixed parameter in the subproblem 1, and is determined by subproblem 2. θ and F are coupling variables, and ℓ is a shared design variable. Although the drag force analysis is part of subproblem 2, problem convergence was improved by locating it in subproblem 1. The formulation for subproblem 2 is:

$$\begin{aligned}
& \min_{\ell^{(2)}, w^{(2)}, \theta^{(2)}} && \mathbf{v}_{12} \mathbf{c}_{12} + \|\mathbf{w}_{12} \circ \mathbf{c}_{12}\|_2^2 \\
& \text{subject to} && \ell^{(2)} w^{(2)} - A = 0 \\
& \text{where} && \mathbf{c}_{12}(\ell^{(2)}, w^{(2)}, \theta^{(2)}) = [\theta^{(1)} - \theta^{(2)}, F^{(1)} - F^{(2)}, \ell^{(1)} - \ell^{(2)}]
\end{aligned} \tag{4.22}$$

The aerodynamic subproblem has no local design objective function, so the optimization objective is the penalty function on \mathbf{c}_{21} only. Note that the constraint \mathbf{c}_{21} is defined identically in subproblem 2 except for the input arguments. Here $\theta^{(2)}$ and $\ell^{(2)}$ are independent variables, while $\theta^{(1)}$, $F^{(1)}$, and $\ell^{(1)}$ are fixed parameters determined by subproblem 1. The drag force $F^{(2)}$ is computed using Eq. 3.3.

The ALC solution was implemented using the ENMOM approach and the result matched the optimal design determined by monotonicity analysis. The outer loop parameters used were $\beta = 2.2$ and $\gamma = 0.40$. Ten outer loop iterations were required to satisfy the convergence criterion of $\|\mathbf{c}\| \leq 1.0 \cdot 10^{-4}$. Inner loop convergence required that the normalized sum of subproblem objective functions was less than $1.0 \cdot 10^{-4}$. Table 4.1 summarizes the ALC solution process for this problem, where i is the outer loop iteration number, j is the inner loop iteration number, and n_f is the number of function evaluations required for each outer loop iteration. The number of inner loop iterations is larger at the beginning of the process, but decreases as the solution is approached. The number of function evaluations indicates that the difficulty of the subproblems gradually increased during most of the process, but then dropped off near the end.

Karsemakers studied the air flow sensor design problem at length and showed that modifying problem structure or the coordination algorithm had significant influence over computational expense [76]. In some cases the computational benefit exceeded one order of magnitude. Chapters 5 through 7 will develop a formal approach for choosing system partitions and coordination algorithms for ALC and related formulations.

4.4 Concluding Comments

This chapter reviewed several important formulations for decomposition-based design optimization, which fall under two categories: single-level and multi-level. Single-level formulations may employ a distributed analysis approach, but use a single optimization algorithm for the entire system. Single-level formulations meet the requirements of system consistency and optimality, and can help ease system analysis difficulties, but may be inappropriate for problems with a very large number of design variables. Multi-level formulations address this problem by using a distributed optimization approach; a separate optimization problem is defined for each subproblem, reducing the total number of design variables that must be managed by any single optimization algorithm. Several examples were used to explain the formulations covered in this chapter. Parametric studies on coupling strength and algorithm parameters were performed, providing important insights into the formulations reviewed. The understanding of system optimization formulations developed in this chapter is a foundation for the quantitative approach to partitioning and coordination decisions presented in the following chapter.

Table 4.1 ALC solution progress for the air-flow sensor problem

i	j	n_f
1	3	174
2	6	191
3	3	129
4	5	658
5	4	442
6	3	354
7	3	297
8	2	168
9	2	90
10	2	88

Chapter 5

Optimal Partitioning and Coordination: Theoretical Framework

Solving a system design optimization problem using decomposition-based design optimization requires that a system partition and coordination strategy be defined a priori. Partitioning and coordination (P/C) decisions can have a profound effect on computational expense of the solution process. P/C decisions should minimize the complexity of the resulting distributed optimization problem. The solution difficulty for a system design problem given a particular set of P/C decisions is difficult to estimate without actually solving the problem, but is required for P/C decision optimization to be beneficial. Complexity is approximated here by the coordination problem size (CS) and the maximum subproblem size (SS_{\max}). These metrics can be estimated using the reduced adjacency matrix for a system and the proposed system partition and coordination strategy.

The primary objective in this chapter is to demonstrate that P/C decisions are coupled; i.e., a system partition will influence what coordination strategy should be chosen, and vice versa. Section 1.1 described how an independent or sequential approach to design can lead to suboptimal solutions if coupling is present. This applies to the P/C optimal decision problem as well. Independent, sequential, and simultaneous approaches are formulated for solving the P/C problem, and it is shown that for three example systems only the simultaneous approach is successful. This is evidence of coupling between partitioning and coordination decisions. Coordination strategy decisions are limited to subproblem sequence in this chapter. This simplifies the analysis here, but provides sufficient modeling fidelity to identify coupling and study tradeoffs. Tradeoff information may be used to assess the appropriateness of decomposition-based design optimization for a particular system.

5.1 P/C Problem Formulation

The objective in the optimal P/C decision problem is to find a system partition and subproblem sequence that minimizes overall computational expense of the system design solution

process. Both subproblem solution and coordination problem solution difficulty contribute to this expense. If a general model for overall expense is unavailable, metrics for subproblem and coordination problem difficulty may be investigated instead. Problem size is used here to approximate relative computational difficulty¹. Each metric is to be minimized, but these can be conflicting objectives. For example, fine partitions reduce subproblem size but increase coordination problem size. When multiple objectives conflict, a single optimal solution cannot be found. The solution to a multiobjective optimization problem is instead a set of non-dominated solutions [110]. A point in the objective space is dominated if at least one objective function value can be improved without degrading the other objective function values. The set of non-dominated points is also called the Pareto set; a point in the Pareto set is Pareto-optimal. The Pareto set illustrates the tradeoff between conflicting objectives by showing how much one is degraded by improving the other. A multiobjective optimization study will be performed on the coordination problem and subproblem size metrics.

The coordination problem and subproblem size metrics were derived based on a distributed optimization formulation, such as ATC or ALC, where consistency is managed using a penalty relaxation method and the subproblems are coordinated using fixed-point iteration. The coordination problem size CS is defined as the total number of consistency constraints for external shared variables and feedback coupling variables, to be solved by the coordination algorithm:

$$CS = n_{\bar{x}_s m} + n_{\bar{y} f} \quad (5.1)$$

The number of external shared variable consistency constraints is approximately $n_{\bar{x}_s m}$, a metric based on the number of external shared variables. The number of feedback coupling variable consistency constraints in the coordination problem is equal to the number of feedback external coupling variables $n_{\bar{y} f}$. It can be shown that the minimum number of consistency constraints required for the i -th external shared variable is $n_{p_i} - 1$, where n_{p_i} is the number of subproblems that share the i -th external shared variable. Therefore, the sum of $n_{p_i} - 1$ over all $n_{\bar{x}_s}$ external shared variables is a reasonable approximation for the number of external shared variable consistency constraints: $n_{\bar{x}_s m} = \sum_{i=1}^{n_{\bar{x}_s}} (n_{p_i} - 1)$. The reason $n_{\bar{y} f}$ is used instead of the total number of external coupling variables $n_{\bar{y}}$ is to penalize feedback, which slows coordination convergence [21]. CS does not model how coordination problem size depends on consistency constraints allocation, and is therefore an approximation.

The size of subproblem i , SS_i , is defined as the number of associated decision variables, consistency constraints, and analysis functions. Since IDF-type subproblem formulations

¹Use of size metrics based on the existence of dependence relationships, rather than more sophisticated estimates, was discussed in Section 2.3

are assumed, no constraints are needed for internal shared variables, and one constraint is required for each internal coupling variable.

$$\begin{aligned}
SS_i = & (n_{\bar{x}_s i} + n_{x_{\ell} i} + n_{y_i} + n_{\bar{y}_{fi}}) \\
& + (n_{\bar{x}_s i} + n_{y_i} + n_{\bar{y}_{fi}}) \\
& + (n_{ai})
\end{aligned} \tag{5.2}$$

The number of external shared variables associated with subproblem i is $n_{\bar{x}_s i}$, the number of local variables is $n_{x_{\ell} i}$, the number of internal coupling variables is n_{y_i} , the number of coupling variables input from subproblems executed after subproblem i is $n_{\bar{y}_{fi}}$, and the number of analysis functions is n_{ai} . SS_{\max} is the maximum of all SS_i values. Previous approaches described in Section 2.4 used only the number of variables or analysis functions in the subproblem size metric, so the above formula is an improvement.

5.2 P/C Problem Solution

Four strategies can be used to solve the P/C decision problem. In the first strategy, labeled (P, C) , the P and C problems are solved independently. In the second strategy, labeled $(P \rightarrow C)$, the partitioning problem is solved first, and the resulting partition is used in solving the coordination decision problem. The third strategy, labeled $(C \rightarrow P)$, solves the partitioning problem using a coordination method definition obtained by first solving the coordination decision problem. The fourth strategy, labeled $(P||C)$, minimizes CS and SS_{\max} simultaneously, solving the actual Pareto-optimization problem. If partitioning and coordination decisions are coupled, only the simultaneous approach will successfully identify the Pareto set for the P/C decision problem. The examples will show that the first three strategies cannot capture CS – SS_{\max} tradeoff information or always identify Pareto-optimal solutions, providing evidence that interactions between partitioning and coordination decisions indeed exist and are important.

In the optimal P/C model a restricted growth string (RGS) [131], \mathbf{p} of length m , is used to specify the partition by prescribing which analysis functions belong to each subproblem. The value of p_i is the subproblem that analysis function i belongs to. Redundant representations of partitions are avoided since as an RGS, \mathbf{p} must satisfy:

$$p_1 = 1 \wedge p_i \leq \max\{p_1, p_2, \dots, p_{i-1}\} + 1 \tag{5.3}$$

Coordination decisions here are restricted to subproblem sequencing, defined by the vector \mathbf{o}_s , where the value of o_{si} is the evaluation position of subproblem i , and $o_{si} \neq o_{sj} \forall i, j \in \{1, 2, \dots, N\}$. In the (P, C) and $(C \rightarrow P)$ strategies coordination decisions are made without partitioning information, so it is impossible to specify a subproblem sequence and the analysis function sequence \mathbf{o} is used instead.

Two independent problems are solved in the (P, C) strategy, and the corresponding formulations are shown in Fig. 5.1. The independent partitioning problem seeks to find \mathbf{p} that minimizes a surrogate for CS , subject to a maximum imbalance constraint (B_{allow}) and a specified number of subproblems (N_{allow}). B is the maximum subproblem size difference incurred by \mathbf{p} , where $SS_i - 2n_{\bar{y}fi}$ is used instead of SS_i for subproblem size since $n_{\bar{y}fi}$ depends on \mathbf{o}_s , which is unavailable. The value used here for B_{allow} is proportional to system size: $B_{\text{allow}} = \lfloor 0.2(m+n) \rfloor$. The surrogate used for CS that does not depend on \mathbf{o}_s is $n_{\bar{x},m} + n_{\bar{y}}$. Forms of the independent partitioning problem have been solved previously [9, 30, 31, 84, 89, 103, 104, 105]. The independent coordination decision problem seeks to find \mathbf{o} that minimizes the number of feedback coupling variables n_{yf} . Since \mathbf{p} is unavailable, CS and SS_{max} again cannot be used. Versions of the independent coordination problem have also been solved previously [99, 115, 116, 133].

$\begin{aligned} \min_{\mathbf{p}} \quad & n_{\bar{x},m} + n_{\bar{y}} \\ \text{subject to} \quad & B \leq B_{\text{allow}} \\ & N = N_{\text{allow}} \end{aligned}$	$\min_{\mathbf{o}} \quad n_{yf}$
----------------------------------------------------------------------------------------------------------------------------------------------------------------------	----------------------------------

Figure 5.1 Independent (P, C) optimization approach

The $(P \rightarrow C)$ strategy [89] first solves the independent partitioning problem and then passes the result \mathbf{p}^* as a fixed parameter to the coordination decision problem (Fig. 5.2). Since a partition is defined the subproblem sequence can be used as the decision vector, and both CS and SS_{max} can be used in the formulation.

$\begin{aligned} \min_{\mathbf{p}} \quad & n_{\bar{x},m} + n_{\bar{y}} \\ \text{subject to} \quad & B \leq B_{\text{allow}} \\ & N = N_{\text{allow}} \end{aligned}$	$\xrightarrow{\mathbf{p}^*}$	$\begin{aligned} \min_{\mathbf{o}_s} \quad & CS \\ \text{subject to} \quad & SS_{\text{max}} \leq SS_{\text{allow}} \end{aligned}$
----------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------	------------------------------------------------------------------------------------------------------------------------------------

Figure 5.2 $P \rightarrow C$ sequential optimization

The $(C \rightarrow P)$ strategy begins with solving the independent coordination decision problem for the analysis function sequence \mathbf{o} (Fig. 5.3). Calculation of CS and SS_{max} in the second stage requires definition of a subproblem sequence. A heuristic is used here to map \mathbf{o} to \mathbf{o}_s : subproblems are ranked in ascending order according to the lowest value of o_i in each

subproblem to define the subproblem sequence.

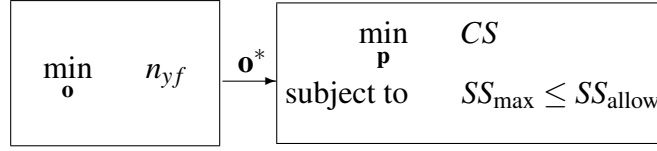


Figure 5.3 $C \rightarrow P$ sequential optimization

The $(P||C)$ strategy seeks optimal values for \mathbf{p} and \mathbf{o}_s simultaneously. The multiobjective problem to be solved is:

$$\min_{\mathbf{p}, \mathbf{o}_s} \{CS, SS_{\max}\} \quad (5.4)$$

The set of Pareto-optimal solutions can be obtained by solving the single-objective optimization problem shown in Fig. 5.4 and varying SS_{\max} as a parameter.

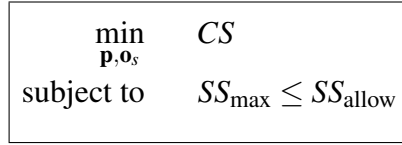


Figure 5.4 Simultaneous $(P||C)$ optimization

5.3 Examples

The four strategies were applied to two randomly generated reduced adjacency matrices to demonstrate the tradeoff between CS and SS_{\max} and the interaction between partitioning and coordination decisions. The optimal P/C decision problems were all solved using exhaustive enumeration, and the appropriate constraints were varied in an effort to generate Pareto sets.

The first example has five analysis functions and seven design variables; its reduced adjacency matrix is:

$$\mathbf{A}_1 = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Figure 5.5 depicts the histogram of all possible CS and SS_{\max} values for an exhaustive enumeration of all possible \mathbf{p} and \mathbf{o}_s combinations. The CS distribution is biased toward larger values, while the SS_{\max} is biased toward smaller values. This is expected since the number of possible sequences and partitions increase with N , and CS decreases with N while SS_{\max} increases with N . Figure 5.6 plots all P/C instances for \mathbf{A}_1 in the CS/SS_{\max} space. In

other words, every point represents a different system partition and coordination strategy option. In many cases several \mathbf{p} and \mathbf{o}_s combinations result in the same CS/SS_{\max} values.

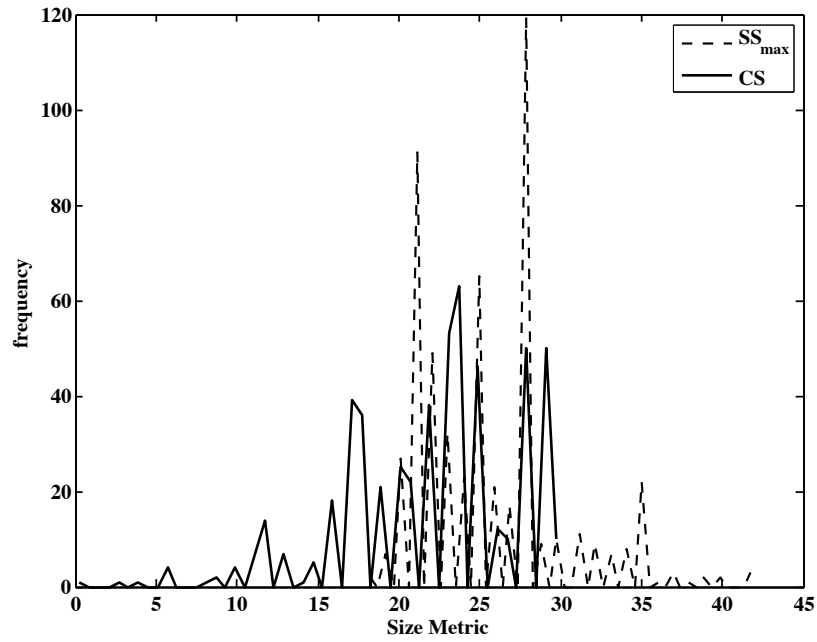


Figure 5.5 CS and SS_{\max} histograms for A_1

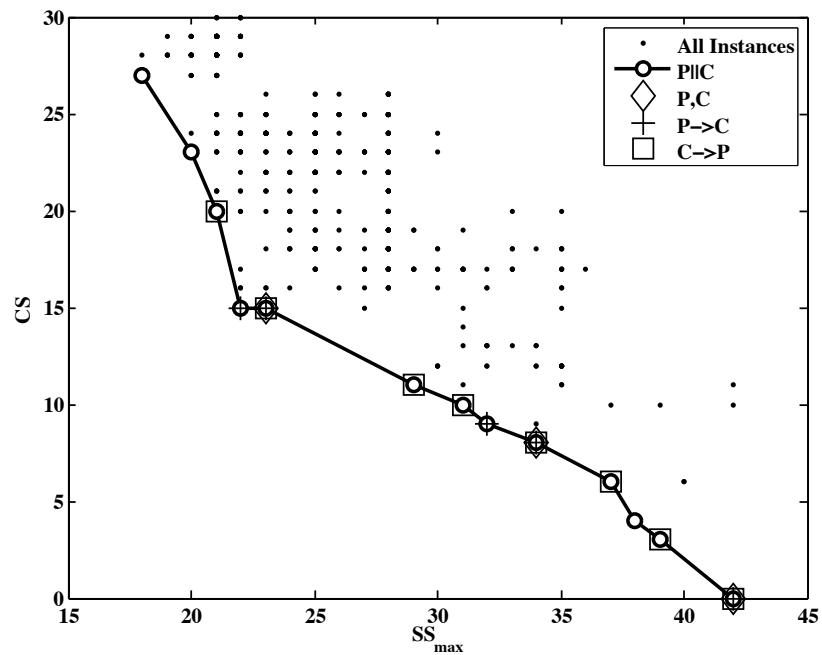


Figure 5.6 Optimization results for A_1

The minimum CS value of zero occurs when $N = 1$, which corresponds to a pure

IDF formulation for the system design problem (with a problem size of 42). In general, distributed optimization makes sense if subproblem size can be reduced from the IDF size through partitioning without requiring a large coordination problem. This is most likely to occur when \mathbf{A} is sparse. Complex products tend to have sparse adjacency matrices [88]. A minimum SS_{\max} value normally occurs when each analysis function is assigned to its own subproblem but is associated with a large coordination problem.

Figure 5.6 also shows solutions obtained by the four different strategies. As expected, $(P||C)$ finds all 12 Pareto points; (P, C) , $(P \rightarrow C)$, and $(C \rightarrow P)$ identify 2, 4 and 7 Pareto points, respectively. These latter strategies performed well for this small example in that they identified several Pareto-optimal points. A parametric study on B_{allow} values revealed that increasing allowed imbalance for the (P, C) and $(P \rightarrow C)$ approaches initially improves the number of Pareto points identified, but increasing B_{allow} much beyond $\lfloor 0.2(m+n) \rfloor$ does not continue to improve results. In all cases non-simultaneous approaches identified only a fraction of the Pareto set. In the next slightly larger example the performance discrepancy between simultaneous and non-simultaneous approaches is more significant. The second example has six analysis functions and ten design variables:

$$\mathbf{A}_2 = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

The biases in the CS and SS_{\max} distributions are now clearer in the histogram of P/C instances for \mathbf{A}_2 (Fig. 5.7). These distributions can influence the performance of algorithms other than exhaustive enumeration (such as genetic algorithms) for solving the optimal P/C problem [99].

Figure 5.8 shows CS and SS_{\max} values for all P/C instances for \mathbf{A}_2 . $P||C$ located all 9 Pareto points. No solutions to the non-simultaneous approaches are Pareto-optimal except for the trivial case of $N = 1$. This result is significant because if any non-simultaneous approach is used to make P/C decisions for this system, both subproblem and coordination problem size could be reduced further. This sub-optimality is expected to be more pronounced as system size and complexity increases.

CS - SS_{\max} tradeoff information can be used to assess system suitability for solution via distributed optimization since it illustrates the sensitivity of best-case solution expense to increases in partition refinement. If increasing N causes CS to rise sharply without appreciable SS_{\max} reduction, AiO or IDF may be preferable to distributed optimization.

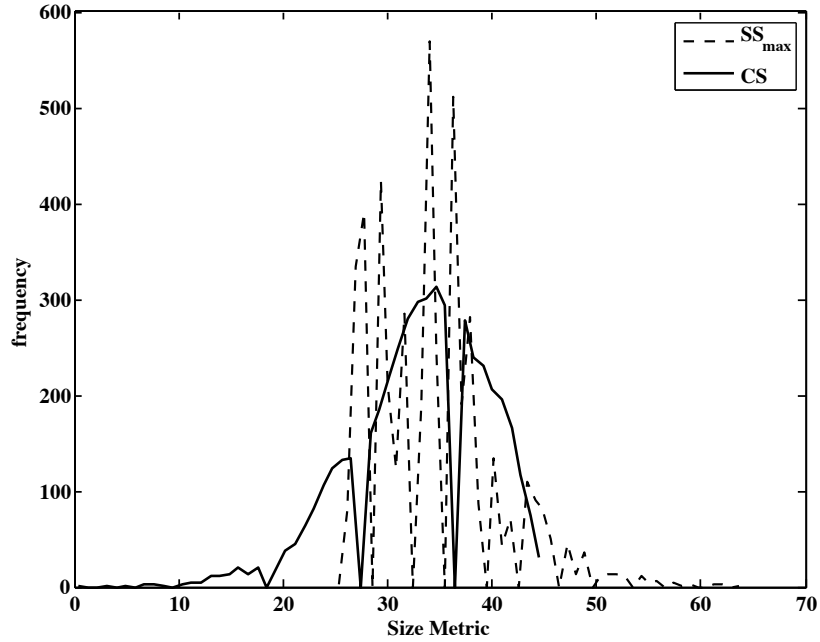


Figure 5.7 CS and SS_{\max} histograms for A_2

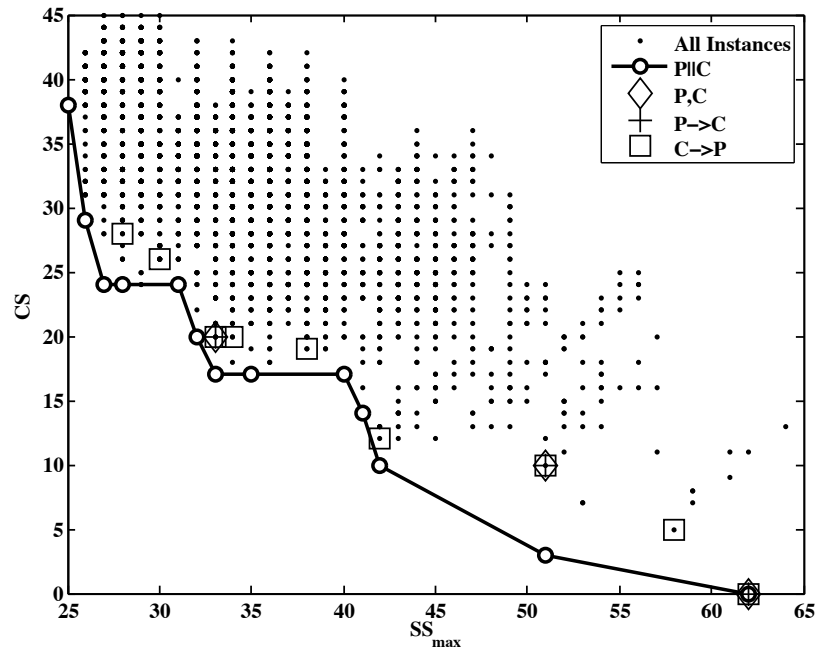


Figure 5.8 Optimization results for A_2

Thus, the second example is a good candidate for distributed optimization.

An interesting phenomenon is evident in Figure 5.8: there exists an instance where $SS_{\max} = 64$, which is greater than 62, the size of a single large subproblem. The corre-

sponding partition cuts across a very large number of linking variables, and the subproblem order maximizes feedback. It is conceivable that some systems could exhibit this behavior for most or all P/C options, making them exceptionally poor candidates for distributed optimization.

5.4 Water Pump Electrification Example

The previous two examples involved abstract systems defined only by their reduced adjacency matrices. This section applies the P/C methods outlined above to a system design problem that corresponds to a physical system. Section 3.5 detailed the analysis and design of a centrifugal water pump driven by a DC electric motor for automotive applications. Switching from mechanical belt drive to electric drive allows for substantially more efficient operation. The model involves five analysis functions that compute performance metrics based on ten design variable values. Design variables and analysis functions are summarized in Table 3.4. Several analysis interactions were modeled. Dependence relationships can be extracted from the analysis model, and are presented in the reduced adjacency matrix for this system:

$$\mathbf{A}_3 = \begin{array}{c|cccccccccccc} & T & I & \omega & \tau & d & d_2 & d_3 & L & \ell_c & D_2 & b & \beta_1 & \beta_2 & \beta_3 \\ \hline T & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ I & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ \omega & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ \tau & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{array}$$

The optimal P/C problem for the electric water pump system was solved using all four strategies. As can be seen in Fig. 5.9, $(P||C)$ and $(P \rightarrow C)$ identified all four Pareto points, while (P,C) and $(C \rightarrow P)$ were only able to identify one Pareto point (the trivial solution with $N = 1$).

Of particular interest is the initial low sensitivity of CS to increased N . SS_{\max} can be reduced from 28 to 19 with a CS of 1, making this system an excellent candidate for distributed optimization. Only the $(P||C)$ and $(P \rightarrow C)$ strategies can reveal this low sensitivity.

The matrix \mathbf{A}_3 represents a physical system, so P/C decisions made based on engineering intuition can be compared to optimal P/C modeling results. Dividing the system into motor and pump-related functions corresponds to the partition $\mathbf{p} = [1, 1, 1, 2]$. If the motor subproblem is solved first, then $CS = 1$ and $SS_{\max} = 20$, a good but suboptimal solution. Using a model-derived partition $\mathbf{p} = [1, 2, 3, 4]$ as a starting point to solve coordination

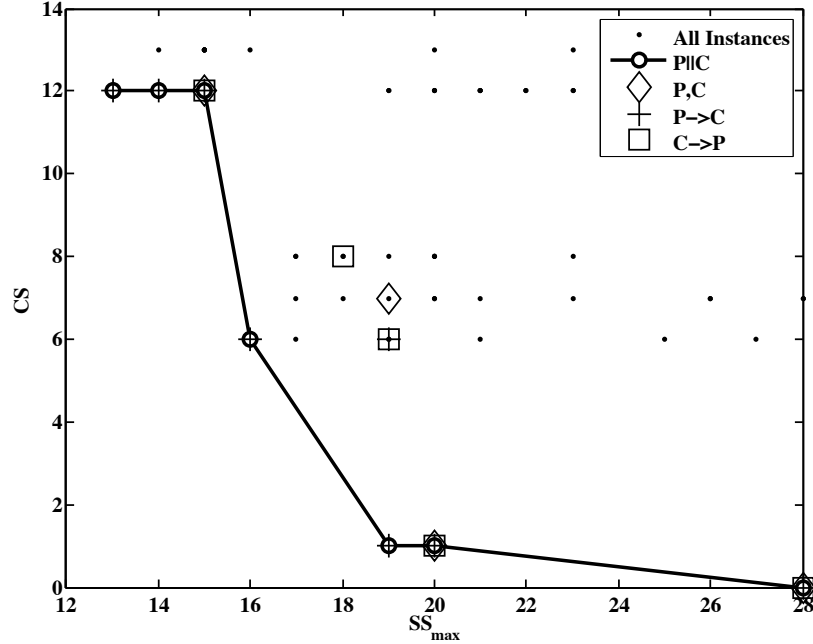


Figure 5.9 Optimal P/C results for pump problem

problem defined in Fig. 5.2 for the optimal sequence, the solution $\mathbf{o}_s^* = [4, 3, 2, 1]$ yields $CS = 12$ and $SS_{\max} = 13$, which is a Pareto point. In this simple example, intuitive and semi-intuitive approaches are rather effective, but cannot quantify the tradeoff between CS and SS_{\max} . Much larger systems are likely to realize greater benefits from the $(P||C)$ strategy, but algorithms more sophisticated than exhaustive enumeration are required in such implementations. One approach to solve the P/C problem for larger systems is presented in the following chapter.

5.5 Concluding Comments

A formal approach for simultaneous partitioning and coordination decision-making was presented in this chapter. The approach quantifies P-C tradeoffs by computing Pareto optima for minimum subproblem size and coordination problem size. Studying these tradeoffs helps determine whether a system design problem is an appropriate candidate for decomposition-based design optimization. The problem-size metrics proposed here captured P/C interactions in the examples successfully. Other metrics that approximate the tradeoff between coordination and subproblem expense can be used instead if desired. Simultaneous P/C optimization can lead to superior decomposition solutions. Comparison to non-simultaneous strategies confirmed the existence of P/C decision interaction, and demon-

strated the value of a simultaneous approach. Exhaustive enumeration was used to generate results for small examples, and a simplified coordination decision model incorporated only subproblem sequencing. Following chapters demonstrate how to solve the P/C problem for larger systems, and how to incorporate linking structure into coordination decisions.

Chapter 6

Extension to Larger Systems

The previous chapter demonstrated that partitioning and coordination decisions are coupled for several example systems, and illustrated tradeoffs present in these decisions. A multiobjective optimization problem was defined, and exact solutions were obtained in each case using exhaustive enumeration. The discrete decision space of Eq. (5.4) is vast; the number of possible partitioning and coordination instances increases exponentially with the number of analysis functions. The $P||C$ problem for systems with more than seven analysis functions cannot be solved in a practical amount of time using exhaustive enumeration. The partitioning and sequencing problems are themselves *NP*-complete [129] and *NP*-hard [124], respectively, making the combined partitioning and coordination problem an especially difficult problem.

Evolutionary algorithms (EAs) have proven to be an effective tool for approximately solving difficult combinatorial optimization problems. This chapter reviews the concepts of EAs and presents the results of applying an EA to a system too large for exhaustive enumeration. An EA developed for solving the $P||C$ problem is applied to the example systems of Chapter 5. Comparison of EA results to exact solutions provides validation of the algorithm. The truss design formulation from Section 3.4 is used to generate a system with eight analysis functions. The EA is then used to obtain an approximate Pareto set for this system. The results indicate that the EA is an effective solution technique for systems of practical size.

6.1 Evolutionary Algorithms

Evolutionary algorithms are a class of algorithms for solving a variety of difficult problems using a process patterned after natural evolution. In biological populations, individuals most well-suited for their environment tend to survive to reproduce and generate offspring. This natural selection process moves to increase the overall ‘fitness’ of a population over time. Two primary operations are exercised: variation and selection. Variation takes place when

parents reproduce to produce offspring with similar, but not identical, traits to their own. Occasionally random mutations occur, another form of variation. Selection mechanisms dictate which individuals reproduce or remain in the population: this applies evolutionary pressure toward increased population fitness. EAs use variation and selection on a population of candidate problem solutions to improve solution quality over successive generations [49]. The process typically used in EAs is outlined in Fig. 6.1.

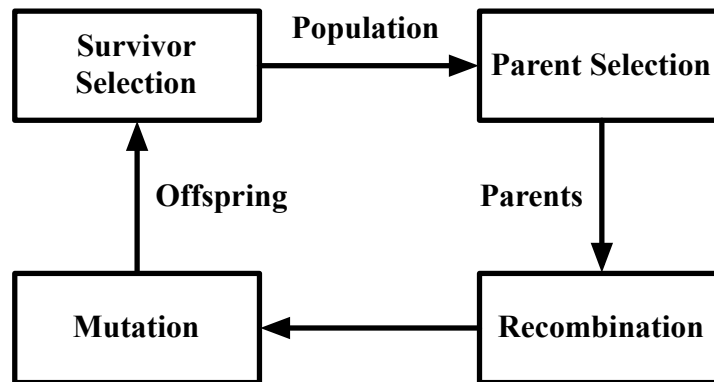


Figure 6.1 Typical evolutionary algorithm process

An EA is initialized with a randomized initial population. Each individual is defined by its chromosome, or genotype. The fitness of each individual is then evaluated based on its genotype. A selection process then chooses which individuals will be parents that produce offspring for the next generation. In recombination, portions of genotypes from two or more parents are assembled to create one or more offspring. Some of the offspring are then selected at random to undergo mutation, which is a small stochastic change in an individual’s genotype. The fitness of the offspring is evaluated, and the population for the succeeding generation is selected from the set of parents and offspring (and in some cases just offspring). This process is repeated until some termination criterion is met.

Evolutionary algorithms must balance the needs for global exploration and local search. Good exploration behavior aids identification of globally superior solutions, while local search proficiency enables the algorithm to rapidly converge on a precise solution. Population diversity enhances exploration performance and reduces the probability of converging to a local solution prematurely. Good global exploration also requires that the genotype encoding is capable of representing all possible candidates in the problem solution space, and that these candidates are reachable through some combination of variation operations. Increased mutation rates improve diversity, but may impede local search performance. Selection applies evolutionary pressure for higher fitness values, but too much pressure also may cause

premature convergence.

Genotype representation is an important aspect of implementing an EA for a specific problem. Examples of established genotype representation types include binary or real-valued strings. In EAs variation operators are applied to genotype representations of individuals. The representation of an individual in the original problem formulation is called its phenotype. Sometimes the phenotype representation is congruent with a standard genotype representation and can be used directly with modification to the EA. When this is not the case one of two approaches may be taken. First, the phenotype representation may be used directly if variation operators compatible with the representation are developed. A large body of work exists regarding specialized variation operators for solving specific types of problems [49]. Second, an appropriate genotype space must be defined along with a surjective mapping onto the phenotype space. Rothlauf developed a theory-based methodology for defining effective genotype representations [118]. An ideal representation should not increase problem difficulty, should enable variation operators to work properly, and should result in a process that is robust to solution location. To meet these requirements it is important that a representation have good locality and little bias toward particular genotypes. Good locality ensures that small changes in the genotype space result in small changes in the phenotype space. Poor locality impacts both global exploration and local search performance. A representation biased toward solution candidates with certain properties may result in failure to identify globally superior solutions.

An important feature of EAs is effectiveness for multiobjective problems (MOPs). The solution to an MOP is a set of non-dominated points (i.e., the Pareto set), rather than a single point solution. EAs are population-based, and are naturally equipped to seek after a set solution. Goldberg proposed a fitness function based on the dominance of candidate solutions, rather than the multiple objective values [65]. This approach enables an EA to identify an approximate Pareto set with a single execution of the algorithm. Traditional MOP solution approaches require that the optimization algorithm be solved multiple times with different objective function or constraint parameters.

Several variants of EAs have been developed to address specific problem types [49]. Each variant has a unique approach for implementing the EA process illustrated in Fig. 6.1. Genetic algorithms (GAs) were introduced by Holland [73], and are typically used for combinatorial optimization problems. Recombination is the primary variation operator, while mutation plays a lesser role. Genotypes are typically represented using binary or integer strings. Fogel, Owens, and Walsh developed evolutionary programming (EP) [54], commonly used for continuous optimization. Real-valued strings are normally used as the genotype representation, and mutation is the sole variation operator. Evolutionary strategies

(ES) were proposed by Rechenberg [113], and are also used for continuous optimization with real-valued genotype representations. Mutation is the primary variation operator, while recombination is secondary. A distinguishing feature of ES is self-adaptation. Most EAs have numerous algorithm parameters that must be tuned for efficient performance, and has been touted as a significant weakness concerning EA practicality. Self-adaptation includes algorithm parameters in the genotype so that as the algorithm progresses, it determines the ideal parameters required to solve the problem at hand effectively. Later on Koza introduced genetic programming (GP) as a technique specifically for automated computer code generation [83]. Some programming languages, such as Lisp [122], are naturally expressed in a tree¹ structure; GP utilizes a tree genotype representation, and specialized recombination and mutation techniques that operate directly on trees.

Eiben explained that EAs are effective at solving difficult problems with acceptable results in a reasonable amount of time [49]. While EAs are frequently applied to optimization problems, they are not strictly optimization algorithms [40]. EAs are not based on any type of optimality conditions, and therefore cannot guarantee optimality of results. Many optimization problems cannot be solved exactly in a practical amount of time, but EAs offer a way to obtain approximately optimal results in a satisfactory period of time.

While EAs have proven to be effective at solving difficult problems [48], no single algorithm within this class is ideal for solving all problems. Wolpert and Macready set forth the ‘no free lunch’ theorem, which asserts that if we average the performance of nonrevisiting² black box algorithms over all possible problems, all algorithms perform equally well [146]. In other words, if a particular algorithm is well-suited for one type of problem, then it will not be effective at solving a distinctively different type of problem. EAs should not be applied indiscriminately for solving problems, but the principles of EAs, along with problem-specific knowledge, should be brought to bear in constructing algorithms tailored to solve the problem at hand.

6.2 Evolutionary Algorithm for Partitioning and Coordination

The combined partitioning and coordination problem is a multiobjective combinatorial optimization problem. The system partition and the subproblem sequence are not represented easily using standard techniques or processed using standard variation operators. The

¹Trees are graphs without a cycle.

²A nonrevisiting algorithm does not evaluate the same candidate solution more than once. Under certain assumptions EAs fall under this category of black box algorithms.

phenotype representation for this problem is \mathbf{p} and \mathbf{o}_s . Rather than constructing new variation operators that work on both a restricted growth string and integer sequence, a separate genotype representation was defined with a mapping to the phenotype space. This enabled the use of standard variation operators. The EA used here closely resembles a standard GA for multiobjective problems. The primary differences include a customized genotype representation as well as the use of two different recombination operators and increased emphasis on mutation to improve global search properties. The first crossover type is arithmetic, where a random point along the line connecting two parents in the solution candidate space is selected as the offspring. The second crossover type is a simple single point crossover where a random point in the parent genotype strings divides them in two, and the substrings are swapped to form offspring. The remainder of this section details the new genotype representation and compares EA results to exact results for the example systems from Chapter 5.

6.2.1 Partition Genotype Representation

The system partition is represented in the genotype space using $\hat{\mathbf{p}}$, where $\hat{p}_i \in \{1, 2, \dots, m\}$ and $i = 1, 2, \dots, m$. The subproblem that the i -th analysis function belongs to is \hat{p}_i , but the vector $\hat{\mathbf{p}}$ is no longer constrained by restricted growth string requirements described in Eq. 5.3. Standard variation operators can be used with $\hat{\mathbf{p}}$. Note that the number of analysis functions m is the maximum possible number of subproblems N . The vector $\hat{\mathbf{p}}$ defines a partition, although not uniquely. There exist m^m possible ways to assign values to $\hat{\mathbf{p}}$, while the number of unique partitions is the m -th Bell number B_m [131]. Redundancy exists in a representation is more than one genotypes map to the same phenotype. The ratio $\mu = m^m / B_m$ quantifies redundancy incurred by using $\hat{\mathbf{p}}$, and increases quickly with m . For a system of size $m = 6$, $\mu = 229.83$. Table 6.1 illustrates how redundancy in the $\hat{\mathbf{p}}$ partition representation increases with system size.

Table 6.1 Redundancy in $\hat{\mathbf{p}}$ partition representation

m	m^m	B_m	μ
1	1	1	1.00
2	4	2	2.00
3	27	5	5.40
4	256	15	17.07
5	3,125	52	60.10
6	46,656	203	229.83

Redundant representation can enhance EA performance as long as they do not introduce significant bias toward particular solution types [118]. Strong bias can be a problem if

redundancy is not present in the solution space region that contains the global solution; this reduces the probability that the global solution will be identified.

An algorithm was developed that maps $\hat{\mathbf{p}}$ values in the genotype space to \mathbf{p} values in the phenotype space. Figure 6.2 illustrates (for $m = 6$) the normalized frequency of partition sizes using the phenotype and genotype representations, where N is the partition size. This illustrates that both distributions are biased toward intermediate partition sizes. While genotype bias impedes EA effectiveness, this representation was selected for its favorable properties under crossover and mutation. Development of a representation with less bias that works well with variation operators may lead to improved EA performance.

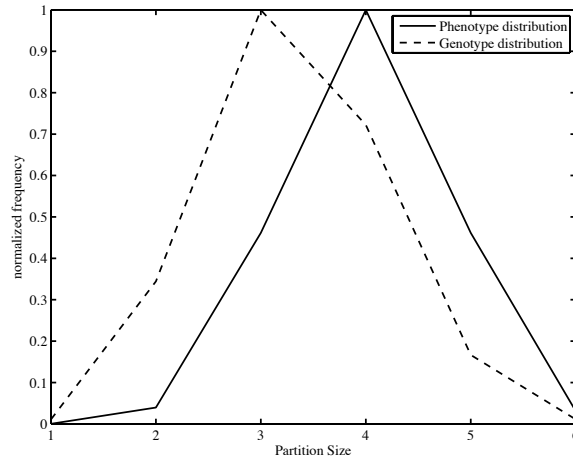


Figure 6.2 Genotype and phenotype representation partition size distributions

6.2.2 Sequence Genotype Representation

The subproblem sequence representation poses a challenge because the length of \mathbf{o}_s depends on \mathbf{p} . An extension of the random key (RK) representation [17] addresses this problem. A random key is a real-valued vector that can be used to encode an integer sequence. For example, suppose $\hat{\mathbf{o}}_s$ is a real valued vector of length N where $0 \leq \hat{o}_{si} \leq 1$, $i = 1, 2, \dots, N$. The components of $\hat{\mathbf{o}}_s$ are then sorted in ascending order, and the order of the original component indices after sorting defines the sequence. RK representations have proven to be more effective than using variation operators designed for sequence permutations. RKs exhibit high locality, and standard variation operators for real values are effective [119]. Relative order is preserved in RKs when crossover recombination operations are performed. Introduction of the RK representation rendered EAs an effective technique for operations research problems [124].

RK representation works well when the number of elements to be sequenced is fixed.

Since this is not the case here, an extension was made. The vector $\hat{\mathbf{o}}$ contains an element for each analysis function: $0 \leq \hat{o}_i \leq 1, i = 1, 2, \dots, m$. The meaning of $\hat{\mathbf{o}}$ depends on \mathbf{p} . If \mathcal{P}_j is the set of analysis function indices that belong to the j -th subproblem, then $\hat{o}_{sj} = \sum_{i \in \mathcal{P}_j} \hat{o}_i / |\mathcal{P}_j|$. The sequence \mathbf{o}_s is then obtained through sorting $\hat{\mathbf{o}}_s$.

The number of possible subproblem sequences increases with partition size, biasing distribution of candidate subproblem sequences toward finer partitions (Fig. 6.3). All possible pairs of \mathbf{p} and \mathbf{o}_s for a given system comprise its phenotype space. The distribution of all these instances for a system of size $m = 6$ is shown in Figure 6.4. The bias present here means that an EA will likely increase effort spent exploring candidate solutions with 4, 5, or 6 subproblems when $m = 6$.

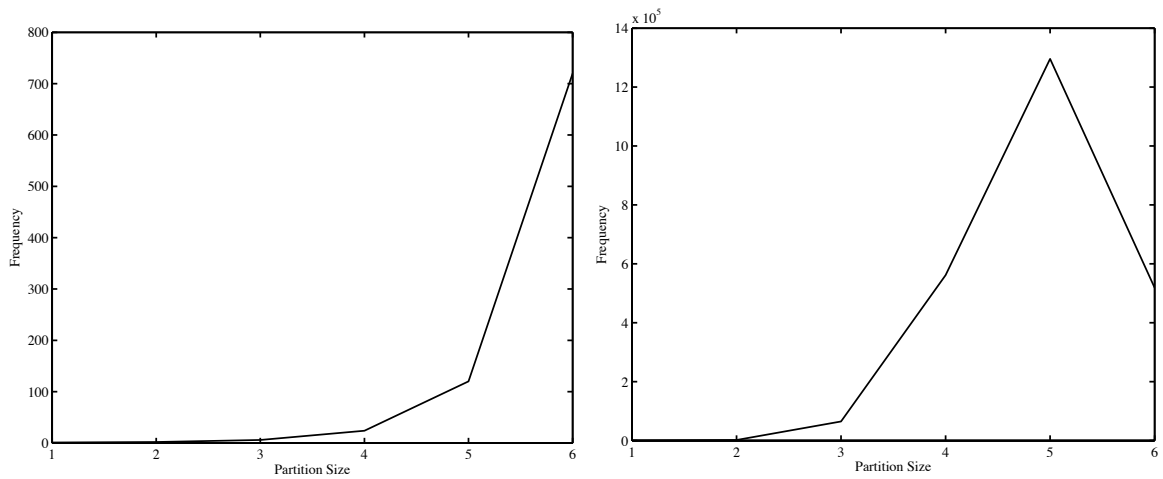


Figure 6.3 Subproblem sequence distribution **Figure 6.4** Combined subproblem sequence and partition size distribution

The genotype to phenotype map outlined above is summarized in Fig. 6.5.

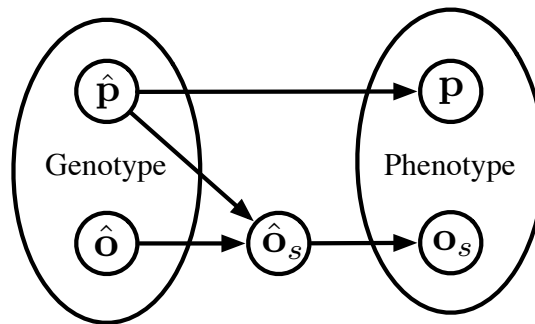


Figure 6.5 Surjective mapping from genotype space to phenotype space

6.2.3 Comparative Examples

The exact Pareto-optimal solutions for the three small example systems from Chapter 5 are compared here against results obtained using the EA described above. The first two example systems are defined by the adjacency matrices \mathbf{A}_1 and \mathbf{A}_2 from Section 5.3. The set of non-dominated points in the objective space identified by the EA was recorded for each system. Figures 6.6 and 6.7 compare these points against the known Pareto points. In System 1 the EA failed to identify two Pareto points, and two of non-dominated points were not Pareto points. In System 2 three Pareto points were not identified. It appears that the EA has difficulty identifying Pareto-optimal solutions with small SS_{\max} (i.e., fine partitions). This is unexpected given the representation bias toward fine partitions, and warrants deeper study.

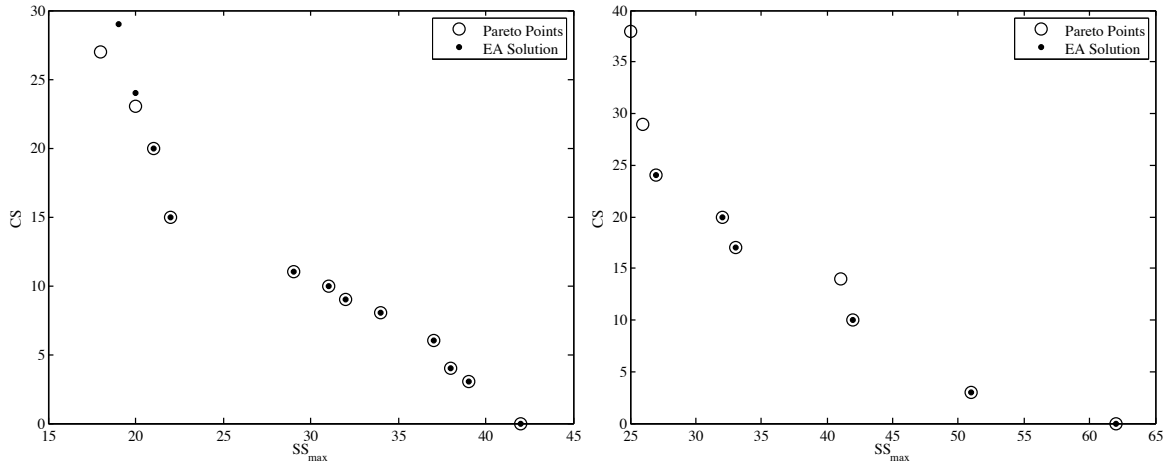


Figure 6.6 EA results for first example system **Figure 6.7** EA results for second example system

The third example system corresponds to the electric water pump design problem presented in Section 3.5; its analysis structure is defined by \mathbf{A}_3 from Section 5.4. Figure 6.8 illustrates that for this smaller system ($m = 4$), the EA successfully identified all four Pareto-optimal solutions.

6.3 Generalized Truss Design Problem

A generalized formulation for a class of truss design optimization problems was presented in Section 3.4. This section describes how it may be partitioned into subproblems, and presents a specific truss design example. The P/C problem for this system is then solved using the EA described above.

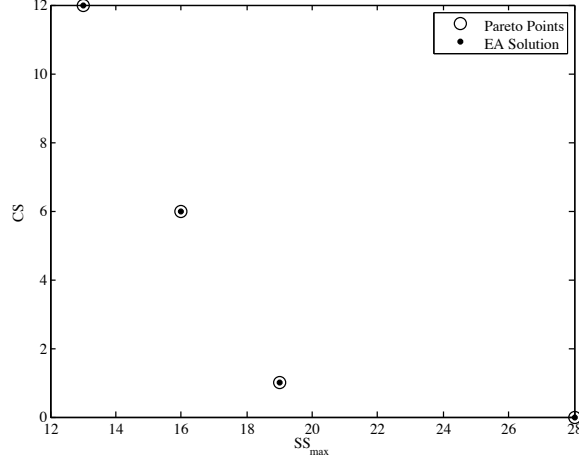


Figure 6.8 EA results for third example system

6.3.1 System Partitioning

Partitioning analysis function associated with the truss design problem into subproblems requires that we first define the meaning of an analysis function. Analysis functions can either be based on truss joints or members. Choosing the latter, the responses of interest for each member include its internal force, mass, stress, buckling criteria and state equation residuals. The analysis function for member $\{i, j\}$ computes these responses as functions of radius, undeformed and deformed joint locations, reaction forces, and internal forces of adjacent members:

$$[f_{ij}, \Omega_{ij}, \sigma_{ij}, b_{ij}, \Delta_{ij}] = \mathbf{a}_{q(i,j)}(r_{ij}, \mathbf{u}_i, \mathbf{u}_j, \mathbf{d}_i, \mathbf{d}_j, \mathbf{R}_i, \mathbf{R}_j, \mathbf{f}_i^{ij}, \mathbf{f}_j^{ij}) \quad (6.1)$$

The vector Δ_{ij} contains the three residual values for the structural compatibility and joint equilibrium state equations, which are constrained to be zero in Eq. (3.29). The function $q(i, j)$ maps the joint indices for member $\{i, j\}$ to the index of the analysis function that computes responses for that member. The two-dimensional vector \mathbf{f}_i^{ij} is the cumulative force from adjacent members acting on member $\{i, j\}$:

$$\mathbf{f}_i^{ij} = \sum_{\{i,k\} \in \mathcal{A}_i \setminus \{i,j\}} \mathbf{f}_{ik} \quad (6.2)$$

The truss member analysis functions follow the form $\mathbf{a}_k(\mathbf{x}_k, \mathbf{y}_k)$ introduced earlier, where $k = q(i, j)$, $\mathbf{x}_k = [r_{ij}, \mathbf{u}_i, \mathbf{u}_j, \mathbf{d}_i, \mathbf{d}_j, \mathbf{R}_i, \mathbf{R}_j]$, and $\mathbf{y}_k = [\mathbf{f}_i^{ij}, \mathbf{f}_j^{ij}]$. The coupling variables are calculated using member force values and geometry information (Eqs. (3.27) and (6.2)). Since f_{ij} is the only analysis output required by other analysis functions, it is the only

coupling variable, and is of prime interest when making partitioning and coordination decisions for the partitioned truss design problem. All other analysis outputs are local quantities.

These analysis functions can be clustered to form subproblems. If we use IDF-type subproblem formulations, when members in different subproblems are connected at common joints, the corresponding internal member forces are coupling variables between the subproblems. In addition, undeformed positions of common joints are shared decision variables. Deformed locations and reaction forces for common joints are also shared variables since the state variables \mathbf{d}_i and \mathbf{R}_i are treated as design variables. The objective function is additively separable, enabling the formation of local subproblem objective functions that consist of the mass of all members in a subproblem.

A wide variety of analysis structures and system sizes are available using this formulation depending on truss size and topology, making Eq. (3.29) a suitable platform for testing the performance of decomposition-based design optimization methods, as well as methods for combined partition and coordination decision-making.

6.3.2 Example: Eight-bar Truss

An eight-bar truss problem with topology adopted from [62] was formulated, partitioned, and solved for use in demonstrating the evolutionary algorithm on a system too large for an exhaustive enumeration approach to making P/C decisions. This truss is illustrated in Fig. 6.9.

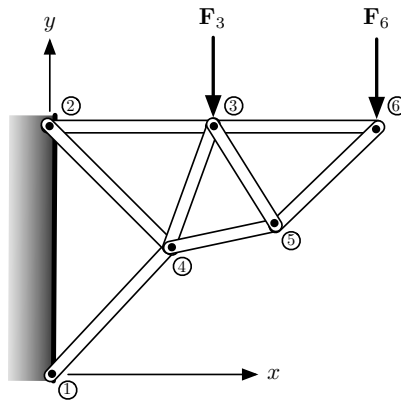


Figure 6.9 Geometry and applied loads for the 8-bar truss problem

The member radii values are $\mathbf{r} = [r_{14}, r_{24}, r_{23}, r_{34}, r_{45}, r_{35}, r_{36}, r_{56}]$, the movable joint positions are $\mathbf{m} = [\mathbf{u}_4, \mathbf{u}_5]$, the deformed positions of non-ground joints are $\tilde{\mathbf{d}} = [\mathbf{d}_3, \mathbf{d}_4, \mathbf{d}_5, \mathbf{d}_6]$, and the reaction forces are $\mathbf{R} = [\mathbf{R}_1, \mathbf{R}_2]$. The reduced adjacency matrix for this system

design problem is:

$$\mathbf{A}_4 = \begin{array}{c|cccccc} & \mathbf{a} & \mathbf{r} & \mathbf{m} & \tilde{\mathbf{d}} & \mathbf{R} \\ \hline \mathbf{a}_{q(1,4)} & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ \mathbf{a}_{q(2,4)} & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ \mathbf{a}_{q(2,3)} & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ \mathbf{a}_{q(3,4)} & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ \mathbf{a}_{q(4,5)} & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ \mathbf{a}_{q(3,5)} & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ \mathbf{a}_{q(3,6)} & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ \mathbf{a}_{q(5,6)} & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{array}$$

Observe that the submatrix formed by the first $m = 8$ rows and columns of \mathbf{A}_4 is symmetric. This is true for any system defined using Eqs. (3.29) and (6.1) since internal member forces are the only coupling variables. Even with this limitation a wide variety of interesting system interaction patterns can be studied.

The design parameters used in this problem and the optimal geometry are given in Table 6.2. The optimal mass is 1.80 kg; as expected, the stress constraints for the members in tension ($\{2, 3\}, \{2, 4\}, \{3, 5\}, \{3, 6\}$) are active, and the buckling constraints for the members in compression ($\{1, 4\}, \{3, 4\}, \{4, 5\}, \{5, 6\}$) are active.

Table 6.2 Design parameters and optimal geometry for the 8-bar truss problem

Design Parameters:				Optimal Geometry:			
ρ	$7.80 \cdot 10^3 \text{ kg/m}^3$	\mathbf{d}_3	$[300, 300] \text{ mm}$	r_{14}	3.44 mm	r_{36}	1.33 mm
E	200 GPa	\mathbf{d}_6	$[600, 300] \text{ mm}$	r_{23}	1.70 mm	r_{45}	2.74 mm
σ_{allow}	250 MPa	\mathbf{F}_3	$[0, -1000] \text{ N}$	r_{24}	1.10 mm	r_{56}	2.61 mm
\mathbf{d}_1	$[0, 0] \text{ mm}$	\mathbf{F}_6	$[0, -1000] \text{ N}$	r_{34}	2.50 mm	\mathbf{u}_4	$[232, 108] \text{ mm}$
\mathbf{d}_2	$[0, 300] \text{ mm}$			r_{35}	0.83 mm	\mathbf{u}_5	$[434, 180] \text{ mm}$

6.3.3 EA Results

The EA was used to solve Eq. (5.4) with the analysis structure defined by \mathbf{A}_4 , and the resulting non-dominated solutions are displayed in Fig. 6.10. The exact solution is unavailable due to system size, so the number of actual Pareto-optimal solutions identified is unknown. The EA parameters were adjusted until the same best set of non-dominated solutions was generated consistently over several runs. The approximate Pareto set illustrates the $CS-SS_{\text{max}}$ tradeoff for this system and indicates that this problem is a good candidate for

partitioning since SS_{\max} can be reduced by almost half before incurring much coordination expense.

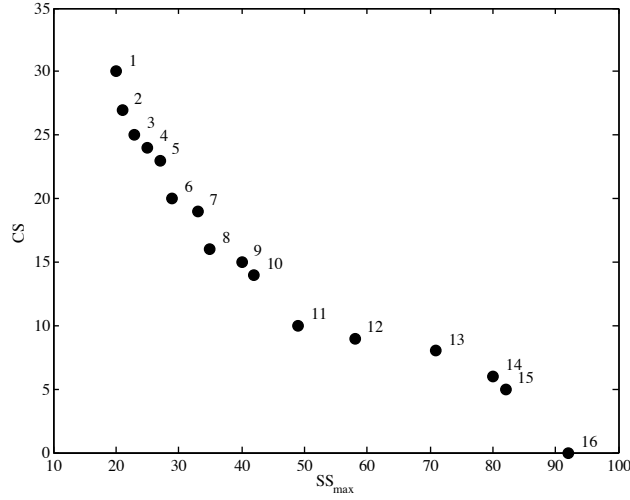


Figure 6.10 Non-dominated solutions for 8-bar truss problem

Point 16 corresponds to a partition with only one subproblem ($\mathbf{p} = [1, 1, 1, 1, 1, 1, 1, 1]$, $\mathbf{o}_s = [1]$) and has a large subproblem size ($SS = 92$) but no coordination expense. The solution approach represented by point 16 is equivalent to solving Eq. (3.29) directly without decomposition. Moving from point 16 to point 11 ($\mathbf{p} = [1, 1, 2, 1, 1, 2, 2, 2]$, $\mathbf{o}_s = [2, 1]$) requires dividing the analysis functions into two subproblems and increases the coordination problem size to 10, but reduces SS_{\max} from 92 to 49. Moving from point 11 to point 6 ($\mathbf{p} = [1, 2, 2, 1, 3, 4, 4, 4]$, $\mathbf{o}_s = [3, 4, 1, 2]$) also increases CS by 10, but only reduces SS_{\max} by 20. Point 11 appears to be an appropriate choice since moving away from it leads to a sharp increase in either CS or SS_{\max} .

Point 1 ($\mathbf{p} = [1, 2, 3, 4, 5, 6, 7, 7]$, $\mathbf{o}_s = [3, 5, 4, 2, 6, 1, 7]$) has a partition size of $N = 7$ and is the finest partition selected by the EA. Either reducing SS_{\max} below 20 is unachievable by choosing $\mathbf{p} = [1, 2, 3, 4, 5, 6, 7, 8]$, or the EA failed to identify a Pareto-optimal solution with a partition size of $N = 8$. The latter possibility is tenable given that the EA had difficulty identifying low SS_{\max} solutions in the comparative examples.

Although the EA cannot generate exact solutions, it provides valuable information for assessing the suitability of a system for decomposition-based design optimization and for making partitioning and coordination decisions. The sensitivity of a system design problem to increased partition size can be visualized using CS – SS_{\max} tradeoff data, and the EA efficiently identifies (approximate) Pareto-optimal solutions.

6.4 Concluding Comments

The optimal partitioning and coordination decision method of the previous chapter relied on an exhaustive enumeration approach, which is limited to systems with no more than six analysis functions. An evolutionary algorithm was developed for the solving the combined partitioning and coordination decision problem. The results from this algorithm were compared against exact solutions for small systems, demonstrating that a good approximation to Pareto-optimal solutions can be obtained using the EA. A formulation for a truss structure design with arbitrary size and topology was introduced as a test example. The EA successfully generated a set of approximate Pareto-optimal solutions for a truss design problem with eight analysis functions. Experience indicates that problems with up to a few dozen analysis functions may be solvable using this algorithm. Opportunity exists to analyze and refine the algorithm so that it can manage even larger design problems, and possibly include more sophisticated coordination decisions, such as the consistency constraint allocation decisions discussed in the following chapter.

Chapter 7

Consistency Constraint Allocation for Augmented Lagrangian Coordination

The Augmented Lagrangian Coordination (ALC) formulation, presented in Section 4.3.4, provides significant flexibility in problem linking structure. The way consistency constraints are allocated among subproblems defines ALC linking structure. While the ability to tailor linking structure to a particular problem can be a profound benefit, manually sifting through the numerous options for linking structure is an overwhelming task. This chapter develops the theory necessary to understand linking structure options for ALC, and shows how to include linking structure decisions along with subproblem sequence decisions in an optimal P/C decision approach for ALC. This is illustrated using a parallel version of ALC.

Techniques from constraint satisfaction programming are used to analyze linking structure for ALC. Graph theory is used to represent the structure of consistency constraints in an ALC formulation. An undirected graph, called the consistency constraint graph, is defined for every linking variable. It is shown that a set of consistency constraints for a linking variable in ALC is valid if and only if the corresponding consistency constraint graph is a spanning tree¹. This important result means that set of all possible ALC linking structure options for a system design problem can be defined, allowing the inclusion of linking structure decisions in the optimal partitioning and coordination decision problem for ALC. A method for solving this problem is described in detail, and demonstrated using the electric water pump design problem from Section 3.5.1.

7.1 Parallel ALC

This section introduces a new approach to formulating ALC problems for parallel computations where the number of subproblems exceeds the number of processors. An example system with six analysis functions is used to illustrate concepts:

¹A graph is a spanning tree if it is connected and contains no cycles.

$$\begin{aligned}
& a_1(x_1, y_{15}), \quad a_2(x_1), \quad a_3(x_6, y_{32}), \\
& a_4(x_1, x_2), \quad a_5(x_2, x_3, y_{52}, y_{54}), \quad a_6(x_4, x_5, y_{65})
\end{aligned}$$

The structure of this system can be visualized using a directed graph representation (Fig. 7.1), and compactly represented with its reduced adjacency matrix.

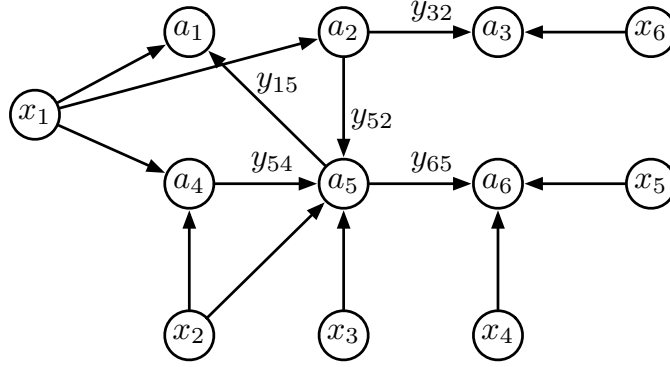


Figure 7.1 Analysis function digraph for example system

The $m \times (n + m)$ reduced adjacency matrix for the above example is:

$$\mathbf{A} = \begin{array}{c|cccccccccccc}
& a_1 & a_2 & a_3 & a_4 & a_5 & a_6 & x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\
\hline
a_1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
a_2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
a_3 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
a_4 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
a_5 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
a_6 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0
\end{array}$$

The ALC coordination algorithm specifies when each subproblem is to be solved, communicates values between subproblems, and updates penalty weights as needed. Coordination difficulty typically increases with the number of external linking variables [142]. The coordination of ALC subproblems can be viewed as the solution to a system of nonlinear equations where subproblems are optimal value functions and external linking variable copies are the unknown quantities. The subproblem i input arguments are $\bar{\mathbf{z}}_i = [\bar{\mathbf{x}}_{si}, \bar{\mathbf{y}}_i]$, and the outputs include updated values for $\bar{\mathbf{x}}_{si}$ and external coupling variables passed from subproblem i to other subproblems ($\bar{\mathbf{y}}_{\cdot i}$). The optimal value function for subproblem i is:

$$\bar{\mathbf{z}}_{\cdot i} = [\bar{\mathbf{x}}_{si}, \bar{\mathbf{y}}_{\cdot i}] = \pi_i(\bar{\mathbf{z}}_i) \tag{7.1}$$

The structure of the coordination problem can be analyzed using a directed graph where subproblems are represented by vertices, and the linking variables passed between subproblems correspond to arcs. Partitioning the example system from Fig. 7.1 using

$\mathbf{p} = [1, 2, 2, 3, 3, 4]$ results in the subproblem graph depicted in Fig. 7.2.

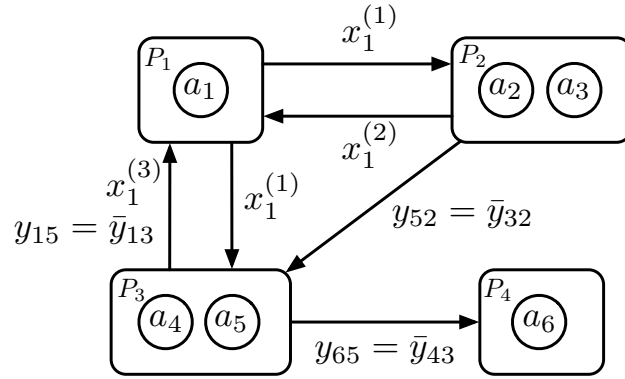


Figure 7.2 Subproblem graph

The shared variable superscripts indicate subproblem of origin. Figure 7.2 illustrates that only one quantity must be passed for each coupling variable, while shared variables require two. Original ALC formulations [141, 142] treat coupling variables as shared variables, increasing both subproblem and coordination burden. Note that while subproblems 2 and 3 share x_1 , copies of x_1 are not communicated between them. The reason for this arrangement will be discussed at length in the next section. Figure 7.3 illustrates the subproblem graph for this example in more compact form.

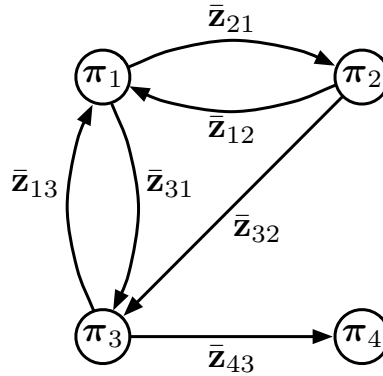


Figure 7.3 Condensed subproblem graph

The ALC coordination algorithm requires an inner and outer loop. The inner loop solves the system of equations formed by subproblem optimal value functions for the external linking variable values. The system of equations to be solved is $\bar{\mathbf{z}} = \boldsymbol{\pi}(\bar{\mathbf{z}})\mathbf{S}$, where $\bar{\mathbf{z}}$ is the set of all external linking variable copies, $\boldsymbol{\pi} = [\pi_1, \pi_2, \dots, \pi_N]$ is the optimal value function for all subproblems, and \mathbf{S} is a selection matrix that matches the outputs of $\boldsymbol{\pi}$ to the components of $\bar{\mathbf{z}}$. The outer loop computes new penalty weight values using inner loop results and the method of multipliers.

An algorithm for solving systems of nonlinear equations is used for the inner loop

problem. A typical approach is to apply fixed point iteration (i.e., nonlinear Gauss-Seidel) by solving each subproblem in sequence, providing the most recent linking variable information for each subproblem solution. Jacobi iteration may also be used to enable parallel solution of all subproblems. If the number of processors available is insufficient for complete parallel execution, block parallel Gauss-Seidel may be applied to blocks of subproblems sequenced into stages. The assignment of subproblems into stages is specified by \mathbf{s} , where the value of s_i is the stage that subproblem i belongs to. The inner loop stages for the running example system correspond to Fig. 7.4 if $\mathbf{s} = [1, 1, 2, 2]$. At each inner loop iteration subproblems 1 and 2 are solved in parallel using values for \bar{z}_{12} , \bar{z}_{21} , and \bar{z}_{13} from the previous inner loop iteration. The subproblems 3 and 4 are solved in parallel using \bar{z}_{31} and \bar{z}_{32} computed during stage 1, and \bar{z}_{43} from the previous inner loop iteration. Reducing the number of values obtained from the previous iteration through clever stage assignments can help speed inner loop convergence.

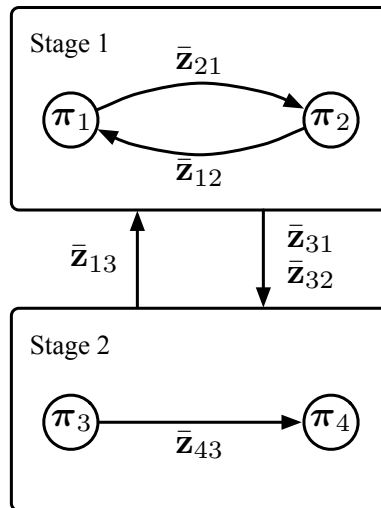


Figure 7.4 Stage graph

7.2 Linking Structure Analysis

One characteristic of formulations for decomposition-based design optimization that can be used to make distinctions between types of formulations is linking structure, i.e., different formulations allow specific approaches to structuring consistency constraints. Most methods require a bi-level or multi-level hierarchical constraint structure. ALC is unique in the flexibility it provides for consistency constraint structure, which enables potentially more efficient implementations where linking structure is tailored to the problem at hand. While flexibility is a beneficial feature, it may be difficult to manage. Early ALC approaches

rely on bi-level or multi-level hierarchical structures to guide linking structure decisions. Deciding between the numerous non-hierarchical possibilities is a task beyond intuition for all but the most simple systems. Optimization techniques can be effectively applied to this task, resulting in superior ALC implementations. A deeper understanding of consistency constraint structure is developed in this section using techniques from constraint satisfaction programming. This section develops the theory required to provably identify the set of valid consistency constraint allocation options for ALC, and the following section uses these results to define an optimal partitioning and coordination decision problem for ALC with linking structure considerations.

We will focus on consistency with respect to a single shared variable, z , that in general could be external or internal. The language below is appropriate for the external case. A system is consistent with respect to a linking variable when all pairs of linking variable copies are consistent:

$$z^{(i)} = z^{(j)} \quad \forall i \neq j, i, j \in \{1, 2, \dots, n^z\} \quad (7.2)$$

where $z^{(i)}$ is the copy of z associated with subproblem i , and n^z is the number of subproblems that share z . The above statement implies $n^z(n^z - 1)$ constraints are required to assure consistency with respect to z . Since $z^{(i)} = z^{(j)}$ is equivalent to $z^{(j)} = z^{(i)}$, the number of constraints can be reduced to $n^z(n^z - 1)/2$ by adopting the convention that the terms in the constraint $z^{(i)} = z^{(j)}$ are ordered such that $i < j$. It will be shown that certain subsets of consistency constraints can ensure consistency of a linking variable, and that $n^z - 1$ constraints is the minimum number required to ensure consistency. It will be demonstrated that these minimal constraint sets are linearly independent, which is a requirement of the augmented Lagrangian penalty method used in ALC.

7.2.1 Consistency Constraint Graphs

Montanari introduced the concept of using graphs to represent constraint sets, where vertices correspond to variables and edges correspond to constraints on variables whose vertices they connect [107]. These constraint graphs are helpful in analyzing constraint set structure and developing solutions for constraint satisfaction problems [143], and along with results from constraint programming provide a framework for understanding consistency constraints in system optimization.

A binary constraint is a constraint on at most two variables, and a binary constraint graph corresponds to a set of binary constraints [94]. The set of $n^z(n^z - 1)/2$ binary consistency constraints on a linking variable can be represented by the complete undirected graph K_{n^z} .

An edge $\{i, j\}$ represents the constraint $z^{(i)} = z^{(j)}$, which can be expressed in negative null form as $z^{(i)} - z^{(j)} = 0$. A convenient representation of this constraint is:

$$\boldsymbol{\theta}_{ij} \tilde{\mathbf{z}}^T = 0 \quad (7.3)$$

where $\boldsymbol{\theta}_{ij}$ is the constraint vector that corresponds to edge $\{i, j\}$, and $\tilde{\mathbf{z}}$ is the vector of all n^z copies of the linking variable z . More precisely:

$$\boldsymbol{\theta}_{ij} = \mathbf{e}_i - \mathbf{e}_j \quad (7.4)$$

$$\tilde{\mathbf{z}} = [z^{(1)}, z^{(2)}, \dots, z^{(n^z)}] \quad (7.5)$$

where \mathbf{e}_i is the i^{th} unit vector of length n^z . Two constraints are adjacent if their corresponding constraint graph edges are adjacent (i.e., they share a common variable). A consistency constraint graph G_c is defined as a subgraph of K_{n^z} that corresponds to a subset of the $n^z(n^z - 1)/2$ consistency constraints. The consistency constraint matrix Θ for G_c is composed of all constraint vectors $\boldsymbol{\theta}_{ij}$ that correspond to edges in G_c . The edges in G_c specify which consistency constraints are to be used in an ALC solution process.

7.2.2 Valid Consistency Constraint Graphs

Not every possible consistency constraint graph is valid for use with ALC. A consistency constraint graph is valid if its associated constraints are equivalent to the constraints specified by K_{n^z} , and if the rows of the corresponding Θ are linearly independent. The first requirement ensures complete consistency of the associated linking variable and the second is necessary for the success of the augmented Lagrangian penalty method used in ALC. After the development of preliminary concepts, necessary and sufficient conditions for the validity of constraint graphs will be given.

Two sets of constraints are equivalent if their feasible domains are equal. The task of finding reduced sets of constraints equivalent to some original set is known as problem reduction. A constraint is redundant if its removal does not change the feasible domain of a constraint set. The composition of adjacent constraints can induce implicit constraints. A constraint is said to be explicit if its corresponding edge exists in G_c , and implicit if it does not. G_c contains redundant constraints if any implicit constraints are identical to either explicit constraints or other implicit constraints [143]. The properties of consistency constraint graphs enable easy identification of implicit and redundant constraints for the purpose of problem reduction. A consistency constraint graph is minimal if it specifies the

fewest number of constraints required to ensure consistency.

Identification of implicit constraints requires application of a binary operator called constraint composition that generates a new constraint from two adjacent constraints [94].

Definition Let $\gamma_1(i, j)$ and $\gamma_2(j, k)$ be two binary constraints with a common variable $(z^{(j)})$ corresponding to vertex j , and let their composition be $\gamma_c(i, k)$. A *binary constraint composition* is valid if values for $z^{(i)}$ and $z^{(k)}$ satisfy $\gamma_c(i, k)$ if and only if there exists a value of $z^{(j)}$ such that $\gamma_1(i, j)$ and $\gamma_2(j, k)$ are satisfied.

In a consistency constraint graph two constraints with a common variable can be composed to form an implicit constraint by taking the vector sum of the corresponding constraint vectors.

Proposition 7.2.1 *The composition of the consistency constraints defined by θ_{ij} and θ_{jk} with the common variable $z^{(j)}$ is $\theta_{ik} = \theta_{ij} + \theta_{jk} = \mathbf{e}_i - \mathbf{e}_j + \mathbf{e}_j - \mathbf{e}_k = \mathbf{e}_i - \mathbf{e}_k$.*

Proof Let a_i and a_k be values for $z^{(i)}$ and $z^{(k)}$, respectively, such that $\theta_{ik}\tilde{\mathbf{z}}^T = 0$ is satisfied. By definition of θ_{ik} , $a_i = a_k$. By selecting a value a_j for $z^{(j)}$ such that $a_i = a_j = a_k$, the constraints $\theta_{ij}\tilde{\mathbf{z}}^T = 0$ and $\theta_{jk}\tilde{\mathbf{z}}^T = 0$ consequently are satisfied. Let b_i , b_j , and b_k be values for $z^{(i)}$, $z^{(j)}$, and $z^{(k)}$, respectively, that satisfy $\theta_{ij}\tilde{\mathbf{z}}^T = 0$ and $\theta_{jk}\tilde{\mathbf{z}}^T = 0$. Since this satisfaction implies $b_i = b_j$ and $b_j = b_k$, $b_i = b_k$ and the composed constraint $\theta_{ik}\tilde{\mathbf{z}}^T = 0$ is satisfied. Therefore, $\theta_{ik} = \theta_{ij} + \theta_{jk}$ is a valid constraint composition. ■

A higher than binary constraint composition is defined by the recursive application of a binary constraint composition. Binary consistency constraints that share a common variable have corresponding edges that are incident to the common variable vertex. At each stage of recursive composition a new edge can be included in the composition if it has a common vertex with the implicit edge generated by the intermediate composition. This occurs when all edges in a set to be composed lie in a connected path on G_c . Suppose p_{ij} is a connected path of length m between the vertices i and j defined by the sequence of unique vertices $\langle v_1, v_2, \dots, v_m, v_{m+1} \rangle$ where $v_1 = i$ and $v_{m+1} = j$. The constraint vector resulting from the extended composition of edges in p_{ij} is $\theta_{ij} = \sum_{\{k,l\} \in p_{ij}} \theta_{kl} = \mathbf{e}_i - \mathbf{e}_j$.

Proposition 7.2.2 *A constraint defined by θ_{ij} , whether implicit or explicit, can be obtained through composition if and only if a path p_{ij} exists in G_c .*

Proof If a path p_{ij} exists in G_c , extended constraint composition can be applied to obtain θ_{ij} :

$$\begin{aligned}
\theta_{ij} &= \sum_{\{k,l\} \in p_{ij}} \theta_{kl} \\
&= \mathbf{e}_{v_1} - \mathbf{e}_{v_2} + \mathbf{e}_{v_2} - \mathbf{e}_{v_3} + \dots + \mathbf{e}_{v_m} - \mathbf{e}_{v_{m+1}} \\
&= \sum_{k=1}^m \mathbf{e}_{v_k} - \sum_{k=2}^{m+1} \mathbf{e}_{v_k} \\
&= \mathbf{e}_{v_1} + \sum_{k=2}^m \mathbf{e}_{v_k} - \sum_{k=2}^m \mathbf{e}_{v_k} - \mathbf{e}_{v_{m+1}} \\
&= \mathbf{e}_{v_1} - \mathbf{e}_{v_{m+1}} = \mathbf{e}_i - \mathbf{e}_j
\end{aligned} \tag{7.6}$$

If a path p_{ij} does not exist in G_c , then at least one edge $\{k, l\}$ in every possible set of constraint edges will be pendant². If k is the pendant vertex, θ_{kl} will contribute \mathbf{e}_k to the constraint composition. Since only edge $\{k, l\}$ is adjacent to k , no constraint vector in the composition can annihilate \mathbf{e}_k . The case for l pendant is similar. Therefore, $\theta_{ij} = \mathbf{e}_i - \mathbf{e}_j$ cannot be obtained if p_{ij} does not exist in G_c . ■

Extended constraint composition leads to a necessary condition for the equivalence of K_{n^z} and G_c . If a consistency constraint graph can be shown to be equivalent to K_{n^z} its set of associated constraints will ensure complete consistency for the linking variable in consideration.

Proposition 7.2.3 *A consistency constraint graph G_c is equivalent to K_{n^z} if and only if G_c is connected.*

Proof If G_c is equivalent to K_{n^z} , G_c specifies either an explicit or an implicit edge for every constraint associated with K_{n^z} . Therefore, a path must exist between every pair of vertices, and G_c is connected. If G_c is connected, a path exists between every pair of vertices and a constraint exists between every pair of vertices in G_c , and the effective constraint sets and feasible domains of G_c and K_{n^z} are identical. ■

A consistency constraint graph is therefore minimal if it is connecting the required vertices using the fewest possible number of edges. By definition, a spanning tree uses the minimum number of edges ($n^z - 1$) to ensure a graph is connected.

Corollary 7.2.4 *A consistency constraint graph is minimal if and only if it is a spanning tree of K_{n^z} .*

²A vertex is pendant if its degree is one, i.e., it is adjacent to exactly one other vertex. An edge is pendant if it is incident to a pendent vertex.

If G_c is connected and uses more than $n^z - 1$ edges, then a cycle exists, and more than one path connects at least one pair of vertices. Such a graph is not minimal since at least one redundant constraint exists that could be removed. Since any consistency constraint can be composed through a composition of explicit constraints if G_c is connected, the set of explicit constraints corresponding to a minimally connected G_c can be viewed as a basis for the constraints in K_{n^z} . The constraint vectors in this set are in fact linearly independent, so indeed form a basis.

Proposition 7.2.5 *The constraint vectors corresponding to explicit edges in G_c are linearly independent if and only if G_c is acyclic.*

Proof If G_c is acyclic, at most one path exists between any pair of vertices. Therefore, if a constraint vector θ_{ij} can be obtained, either θ_{ij} is a column of Θ and edge $\{i, j\}$ exists in G_c , or a unique path p_{ij} with length greater than 1 exists such that θ_{ij} can be induced. If θ_{ij} is a column of Θ , edge $\{i, j\}$ is the only path p_{ij} , and no composition of other constraints will yield θ_{ij} . Since this is true for all explicit constraints, the columns of Θ are linearly independent. If G_c contains a cycle C , then two adjacent vertices on C (i and j) have at least two paths between them: the edge $\{i, j\}$ and $C \setminus \{i, j\}$. Therefore θ_{ij} is an explicit constraint that can be obtained through composition of other explicit constraints, and the columns of Θ are not linearly independent. ■

Corollary 7.2.6 *If G_c is minimal it is an acyclic spanning tree, and therefore has a linearly independent set of explicit consistency constraints.*

The independence properties of spanning trees are generalizable. If \mathcal{S} is the set of all spanning trees of a graph G and their power sets, and E is the set of all edges of G , (E, \mathcal{S}) is the cycle matroid of G . The maximal sets in \mathcal{S} are bases, and \mathcal{S} coincides with the sets of linearly independent columns of the incidence matrix of G [109]. Another result of Proposition 7.2.5 is that the set of all constraint vectors on a linking variable and all linearly independent sets of these vectors form a vector matroid that corresponds to the cycle matroid of K_{n^z} . The favorable properties of binary consistency constraints enable not only the straightforward identification of valid constraint sets, but also open the door to increased understanding of consistency constraints due to their link to spanning trees and cycle matroids.

The foregoing propositions lead to the main result of this section:

Proposition 7.2.7 *G_c is a valid consistency constraint graph if and only if G_c is a spanning tree of K_n .*

Proof If G_c is valid, the columns of Θ are linearly independent, and by Proposition 7.2.5 G_c is acyclic. It also follows from the validity of G_c that consistency is assured, i.e., G_c is equivalent to K_n . By Proposition 7.2.3 G_c is connected, and it follows that G_c is a spanning tree of K_n . Conversely, if G_c is a spanning tree of K_n , G_c is connected and acyclic. It follows from Propositions 7.2.3 and 7.2.5 that G_c ensures consistency and linear independence of constraints. Therefore, G_c is valid. ■

This result means that the set of consistency constraint allocation options for a linking variable z associated with n^z subproblems is defined by the set of all possible spanning trees for the complete graph K_{n^z} . These trees may be represented easily and algorithms exist for their enumerations. This makes inclusion of linking structure options in the optimal P/C decision problem for ALC practical.

7.2.3 Example Consistency Constraint Graph

The consistency constraint graph for x_1 from the example system in Fig. 7.1 is used here to demonstrate valid consistency constraint options and their graph representations. When the partition $\mathbf{p} = [1, 2, 2, 3, 3, 4]$ is used, x_1 is shared between subproblems 1, 2, and 3. The three available consistency constraints are displayed in Fig. 7.5(a) alongside graph edges that represent these constraints. One possible valid consistency constraint graph is shown in Fig. 7.5(b). The vector of x_1 copies is:

$$\tilde{\mathbf{z}} = [x_1^{(1)}, x_1^{(2)}, x_1^{(3)}] \quad (7.7)$$

and the linearly independent consistency constraint matrix for x_1 that corresponds to the edge set $\{\langle 1, 2 \rangle, \langle 1, 3 \rangle\}$ shown in Fig. 7.5(b) is:

$$\Theta = \begin{bmatrix} \theta_{12} \\ \theta_{13} \end{bmatrix} = \begin{bmatrix} 1 & -1 & 0 \\ 1 & 0 & -1 \end{bmatrix} \quad (7.8)$$

7.3 Optimal Partitioning and Coordination Decisions for Parallel ALC

The previous section demonstrated that the set of consistency constraints used for a linking variable must connect associated subproblems using a tree structure to meet ALC requirements for convergence and system consistency, namely, that consistency constraints are

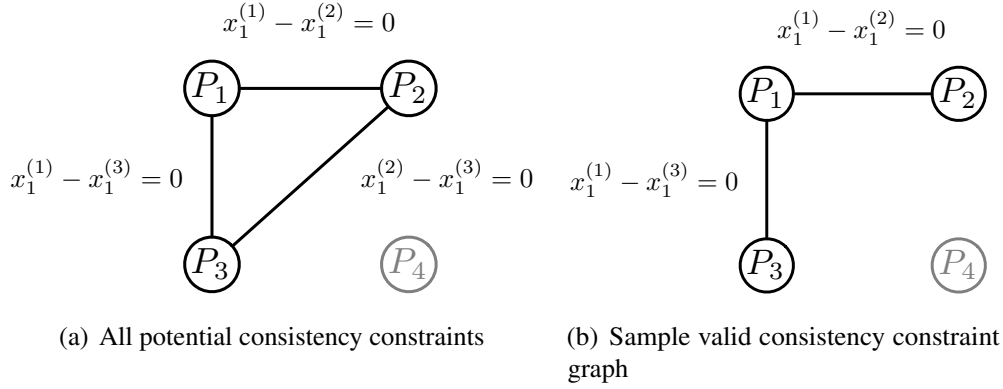


Figure 7.5 Graph representation of consistency constraint options for x_1

linearly independent and ensure all linking variable copies are consistent. Determining consistency constraint structure for every linking variable is an important coordination decision, and influences the computational expense and reliability of an ALC implementation. If v_i is the number of subproblems linked by the i -th external linking variable, then the number of valid options for allocating consistency constraints for this variable is the number of unique spanning trees for a graph with v_i vertices, or $v_i^{v_i-2}$. If n_z is the number of external linking variables in a problem, then $v_1^{v_1-2} \cdot v_2^{v_2-2} \cdot \dots \cdot v_{n_z-1}^{v_{n_z-1}-2} \cdot v_{n_z}^{v_{n_z}-2}$ is the number of alternative linking structure options for a problem with a given system partition. Including linking structure in a coordination decision model dramatically increases the decision space.

The number of linking structure alternatives in a problem can be reduced by exploiting the natural structure present in coupling variable relationships. An analysis function output that is a coupling variable may be communicated to one or more analysis functions. All analysis functions receiving this coupling variable as input link directly to the analysis function that computes the coupling variable; this structure forms a star graph, which is a spanning tree. While it is possible to use other trees for coupling variable consistency constraints, we assume here that the naturally occurring star graph is the consistency constraint graph used for each coupling variable. This reduces the number of trees that must be determined to the number of shared design variables.

The metrics used in the optimal P/C problem of Eq. (5.4) will not work for solving the optimal P/C problem for ALC with linking structure decisions. New metrics must be defined that can account for the parallel nature of the problem implementations, as well as allowing for linking structure decisions. The new approximation for relative coordination problem expense is based on the assumption that the block parallel Gauss-Seidel algorithm converges faster when linking variables input to subproblems are recently computed. In other words, if inputs to an optimal value functions were computed long before the function is evaluated,

convergence is slowed. Jacobi iteration is one extreme where all input data is from the previous iteration. Sequential Gauss-Seidel (FPI) uses the most recently available data. FPI is known to converge faster than Jacobi iteration for linear systems [21]. These arguments do not always extend to nonlinear systems, but are assumed to be a reasonable approximation to enable a priori P/C decisions based on a system's reduced adjacency matrix.

Once a system partition is defined, the subproblem graph can be constructed that describes external linking variable relationships, along with its associated adjacency matrix. \bar{A} is defined to be the $N \times N$ valued adjacency matrix for the subproblem graph in Fig. 7.3, where each entry indicates the dimension of the corresponding linking variable. The estimate for coordination expense here is:

$$CS = \sum_{i=1}^N \sum_{j=1}^N \zeta_{ij} \bar{A}_{ij}$$

The value of ζ_{ij} quantifies how many stages previous to the evaluation of P_i the linking variables \bar{z}_{ij} were computed. CS not only quantifies the number of linking variables in the coordination problem, but accounts for the length of time between linking variable calculation and use as an input. The metric ζ_{ij} is defined as follows:

$$\zeta_{ij} = \begin{cases} s_i - s_j & \text{if } s_i > s_j \\ n^s + s_i - s_j & \text{if } s_i \leq s_j \end{cases}$$

where $n^s = \max(\mathbf{s})$ is the stage depth (i.e., the number of stages in the implementation). The metric used here for quantifying subproblem sizes is more precise than the metric presented in Eq. (5.2) because the exact number of consistency constraints and decision variables in an ALC subproblem formulation are used. The previous metric made an approximation since linking structure information was not available. The size of the optimization problem for subproblem i used here is:

$$\begin{aligned} SS_i &= (n_{\bar{x}_s i} + n_{x_{\ell} i} + n_{y_i} + n_{\bar{y} l i}) \\ &\quad + (n_{\bar{x}_s c i} + n_{y_i} + n_{\bar{y} i}) \\ &\quad + (n_{a i}) \end{aligned}$$

The first four terms comprise the number of decision variables in subproblem i . The number of external shared variables associated with subproblem i is $n_{\bar{x}_s i}$, the number of local variables is $n_{x_{\ell} i}$, the number of internal coupling variables is n_{y_i} , and the number of external

input coupling variables is $n_{\bar{y}i}$. The next three terms express the number of consistency constraints in subproblem i . The number of consistency constraints for external shared variables is $n_{\bar{x}_s ci}$, the number of internal coupling variable consistency constraints is equal to n_{y_i} , and the number of consistency constraints for external coupling variables is equal to $n_{\bar{y}i}$. The last term is the number of analysis functions (n_{a_i}). The maximum subproblem size for each stage is computed, and $\bar{S}S_{\max}$ is the average of the maximum subproblem sizes.

The optimal P/C decision problem for parallel ALC with linking structure considerations is to simultaneously minimize CS and $\bar{S}S_{\max}$ by selecting a system partition \mathbf{p} , subproblem stage assignment \mathbf{s} , and a valid consistency constraint graph for each external shared design variable. The partition vector is defined as before. The i -th component of (s) is the stage that subproblem i is assigned to. External coupling variable consistency constraints are allocated according to the natural structure of each coupling variable. The length of the vector \mathbf{s} is N , which depends on \mathbf{p} . This complication is easily handled when the optimal P/C decision problem is solved with exhaustive enumeration. Section 6.2 illustrated how to manage this type of dependence when using an evolutionary algorithm. The linking structure decisions also depend on \mathbf{p} . System partition changes the set of external shared design variables, and the subproblems associated with each external shared design variable. As with stage assignment, linking structure can be handled with either exhaustive enumeration or an evolutionary algorithm. A set-valued decision variable \mathcal{C} is defined to represent problem linking structure. The cardinality of \mathcal{C} is equal to the number of external shared design variables in a problem with a given partition. Each member of this set defines the consistency constraint graph for one of the shared variables. One approach to representing a consistency constraint graph, which must be a spanning tree, is with an edge set. For example, the variable x_1 in Fig. 7.2 is shared between P_1 , P_2 , and P_3 , but the constraints on x_1 appear only in \bar{c}_{12} and \bar{c}_{13} , which are the consistency constraints connecting P_1 with P_2 and P_1 with P_3 , respectively. The edge set corresponding to these constraints for x_1 is $\{\langle 1, 2 \rangle, \langle 1, 3 \rangle\}$. The optimal P/C problem is:

$$\min_{\mathbf{p}, \mathbf{s}, \mathcal{C}} \{CS, \bar{S}S_{\max}\} \quad (7.9)$$

Specifying \mathbf{p} , \mathbf{s} , and \mathcal{C} defines completely a parallel ALC partition, coordination algorithm, and set of subproblem formulations. The number of P/C alternatives increases more quickly with problem size for this approach than without linking structure decisions. Exhaustive enumeration is only practical for systems with up to four analysis functions. Alternative solution techniques, such as evolutionary algorithms, must be employed for larger systems.

7.4 Example: Electric Water Pump Design Problem

The P/C decision method for ALC described above, which includes both stage assignment and linking structure decisions in the coordination decision problem, was applied to the electric water pump design problem from Section 3.5.1. This design problem has 9295 unique P/C alternatives, and two Pareto-optimal points were identified. All instances are displayed in the $CS-\bar{SS}_{\max}$ space in Fig. 7.6, and all partitioning and stage assignment options that correspond to these points are shown.

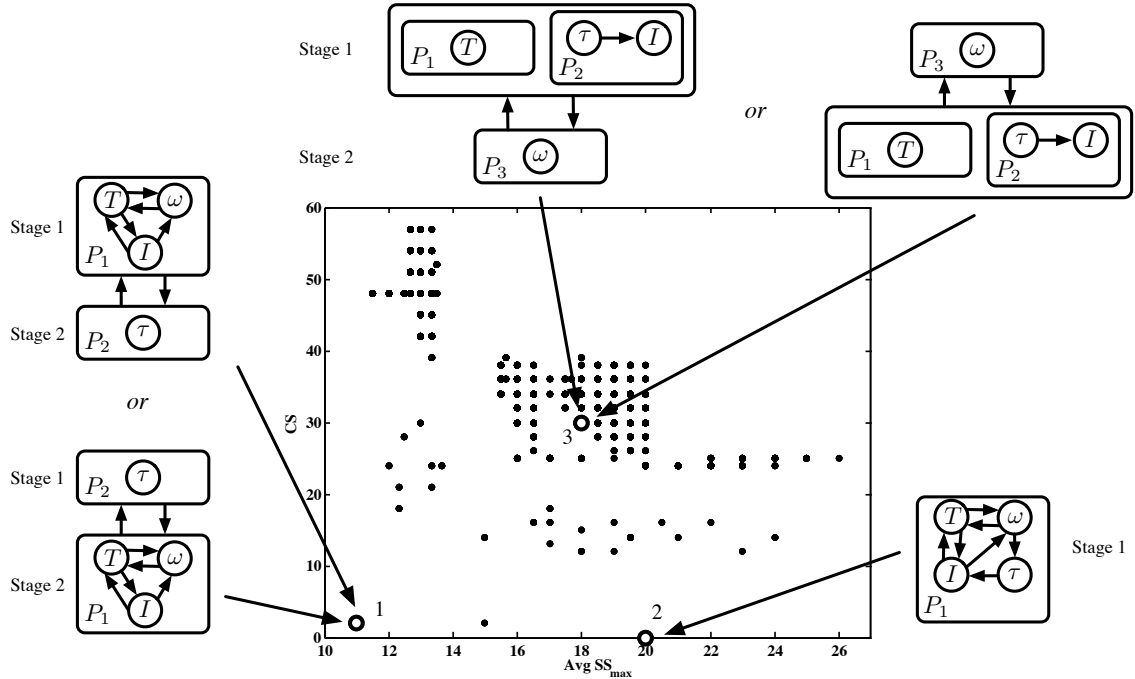


Figure 7.6 ALC P/C results for electric water pump problem

Point 1:

Two P/C decision instances correspond to point 1 in Fig. 7.6, and all share the same partition and problem size metrics:

$$\begin{aligned} CS &= 2 \\ \bar{SS}_{\max} &= 11 \\ \mathbf{p} &= [1, 1, 1, 2] \end{aligned}$$

Neither instance has any shared design variables, but can be distinguished by subproblem stage assignment:

Instance 1: $\mathbf{s} = [1, 2]$

Instance 2: $\mathbf{s} = [2, 1]$

Point 2:

Point 2 represents the IDF formulation for the electric water pump problem, where $CS = 0$ and $\bar{SS}_{\max} = 20$. Note that numerous P/C instances exist with larger subproblem sizes and nonzero coordination problem sizes. These points represent especially poor options for constructing an ALC formulation of the electric water pump problem. Note that moving from point 2 to point 1 reduces \bar{SS}_{\max} from 20 to 11, and requires a coordination problem size of just 2. This result is congruent with the analysis from Chapter 5 that indicates this design problem is a good candidate for decomposition-based design optimization.

Point 3:

A third point, not in the Pareto set, is examined for illustrative purposes. Point 3 corresponds to twelve unique P/C instances, all with the same partition and problem size metrics:

$$\begin{aligned} CS &= 30 \\ \bar{SS}_{\max} &= 18 \\ \mathbf{p} &= [1, 2, 3, 2] \end{aligned}$$

All twelve instances have the same set of external shared design variables:

$$\{x_1, x_2, x_3, x_4, x_5\}$$

The first four are shared between three subproblems, so several options exist for allocating the consistency constraints for these five variables. One possible set of valid consistency constraint graphs is shown in Fig. 7.7.

The twelve instances that correspond to point 3 are distinguished by consistency constraint allocation and stage assignment. The two stage assignments that appear here are:

Instances 1–6: $\mathbf{s} = [1, 1, 2]$

Instances 7–12: $\mathbf{s} = [2, 2, 1]$

These stage assignments are illustrated in Fig. 7.6, and both specify parallel solution of subproblems 1 and 2. No Pareto-optimal P/C instances specify parallel subproblem solution. This is due to both problem structure and the problem size metrics selected. Only CS penalizes stage depth (i.e., the number of stages in a parallel implementation). Other size metrics have been explored, such as the sum of all maximum subproblem sizes for each stage ($\sum SS_{\max}$). This metric penalizes stage depth, and when employed along with CS , the resulting Pareto set contains only single-stage P/C alternatives. An ideal metric would

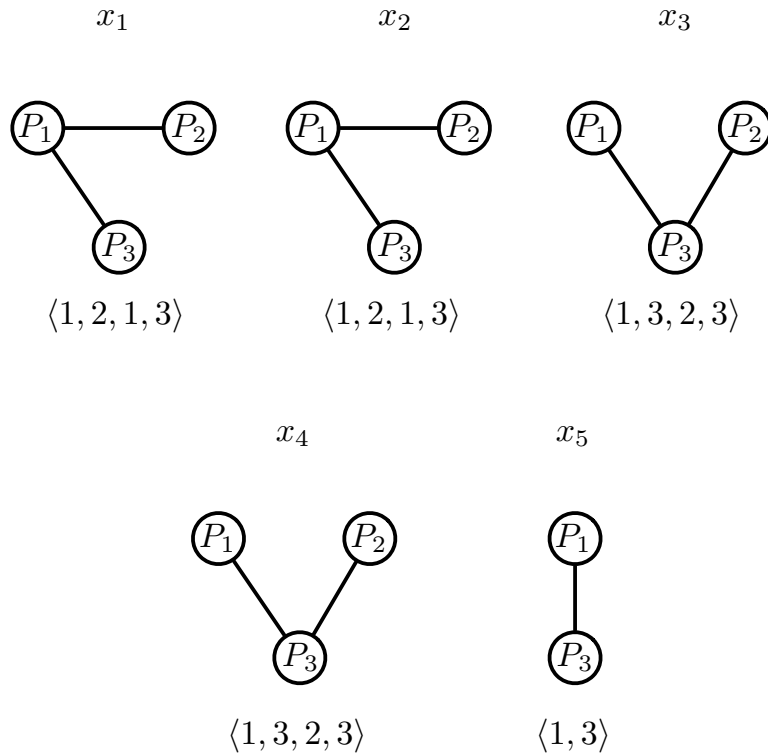


Figure 7.7 Consistency constraint allocation option for point 3

be an accurate estimate of computational expense. Since this is impractical to compute a priori for most problems, approximate metrics must be used. This work has established an approach for constructing implementations of decomposition-based design optimization, and one possible set of metrics has been proposed (i.e., CS and \bar{SS}_{\max}). These approximate two competing sources of computational expense: coordination problem and subproblem expense. Further work should be performed to analyze these and other objective function options.

7.5 Concluding Comments

This chapter introduced a new formulation technique for parallel ALC implementations, which was then used as a platform to study linking structure decisions. ALC linking structure is defined by the way consistency constraints on linking variables are allocated throughout a system design problem. Graph theory and techniques from constraint satisfaction programming were used to identify valid consistency constraint allocation options for ALC. This development enabled inclusion of linking structure decisions with the optimal P/C decision problem for ALC. The decision problem, defined in Eq. (7.9), was solved for the

electric water pump design problem. Opportunities for future work include an investigation of alternative problem size metrics. In the following chapter a detailed electric vehicle design problem is developed, and the solution to the optimal P/C decision problem for the electric vehicle problem is presented.

Chapter 8

Electric Vehicle Design

In this chapter a design problem and corresponding analysis model are developed for a small battery electric vehicle (EV) to demonstrate optimal partitioning and coordination decisions for an engineering system design problem that is more involved than examples presented previously.

The vehicle of interest is a small two-passenger EV intended primarily for urban travel, but capable of highway speeds. The problem objective is to specify the design of powertrain, suspension, and structural systems such that the energy consumed during urban travel is minimized, subject to vehicle performance constraints, including acceleration and range requirements. Figure 8.1 illustrates the vehicle subsystems considered here. Several analysis interactions are included in the vehicle model, such as the influence of component mass and location on vehicle dynamics, and the coupling between powertrain and vehicle dynamics.

This chapter describes in detail the vehicle analysis functions and AiO and ALC design formulations. Optimal partitioning and coordination decisions for parallel ALC solution of the EV problem are then presented.

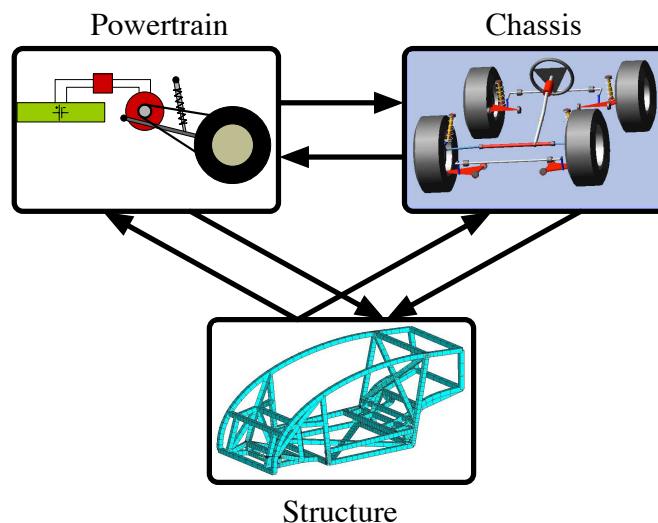


Figure 8.1 Vehicle systems and interactions in the EV design problem

8.1 Vehicle Description

Battery electric vehicles use chemical energy stored in rechargeable batteries as the sole power source to provide locomotion via one or more electric motors, whereas conventional automobiles utilize internal combustion (IC) engines directly coupled to the vehicle drive train, and hybrid electric vehicles (HEVs) employ both electric motors and internal combustion engines. As with HEVs, EVs can recover energy during braking by using drive motor(s) as generators. This energy is lost as waste heat in conventional vehicles, but can be stored for future use in EVs and HEVs. HEVs offer improved fuel economy over conventional vehicles, but at increased complexity and cost. EVs are mechanically more simple than HEVs or conventional vehicles, with very few moving parts, and provide very high levels of energy efficiency [33]. The primary challenge in EV design is the energy storage system; current battery technology does not enable EV range comparable to HEVs or conventional vehicles. While EVs without some type of range extender (such as a genset trailer [58]) are impractical for long-distance travel, they have great utility for short to medium distance excursions. The EV discussed in this chapter has a range of at least 100 miles (161 km), which is long enough to satisfy the needs of more than 95% of light-duty vehicle trips taken in 2001 [44]. This chapter introduces an integrated approach to EV design; simultaneous consideration of major vehicle systems enables analysis of subsystem interactions and tradeoffs, and can aid efforts in enhancing EV performance to improve energy efficiency and competitiveness with HEVs and conventional vehicles.

The EV under consideration here is a two-seat vehicle intended primarily for urban travel, but capable of highway operation. Figure 8.2 shows a top view of major vehicle components and dimensions. Each rear wheel is driven by an electric traction motor; a synchronous belt provides speed reduction between each motor and rear wheel. The motors are mounted on the rear suspension trailing arms, but are located near the trailing arm pivots to help minimize unsprung vehicle mass. The front suspension is a Macpherson strut configuration. The low rolling resistance P145/70R12 tires help reduce energy consumption. The vehicle belongs to the minicompact vehicle class, and has a track width of $W = 1.27$ m and a wheelbase of $L = 1.80$ m. The lithium ion battery width (b_w), length (b_ℓ), and longitudinal position (x_b) all vary in this design problem, but the battery must fit in the space indicated by the dashed box of width $b_{w\max} = 1.20$ m and length $b_{\ell\max} = 1.05$ m. The size and location of the battery has profound influence over vehicle dynamics performance. The coordinate system used in this model is illustrated in the figure.

In recent years great emphasis has been placed on the development of HEVs, and increasingly, EVs. These new vehicle configurations present substantial design challenges.

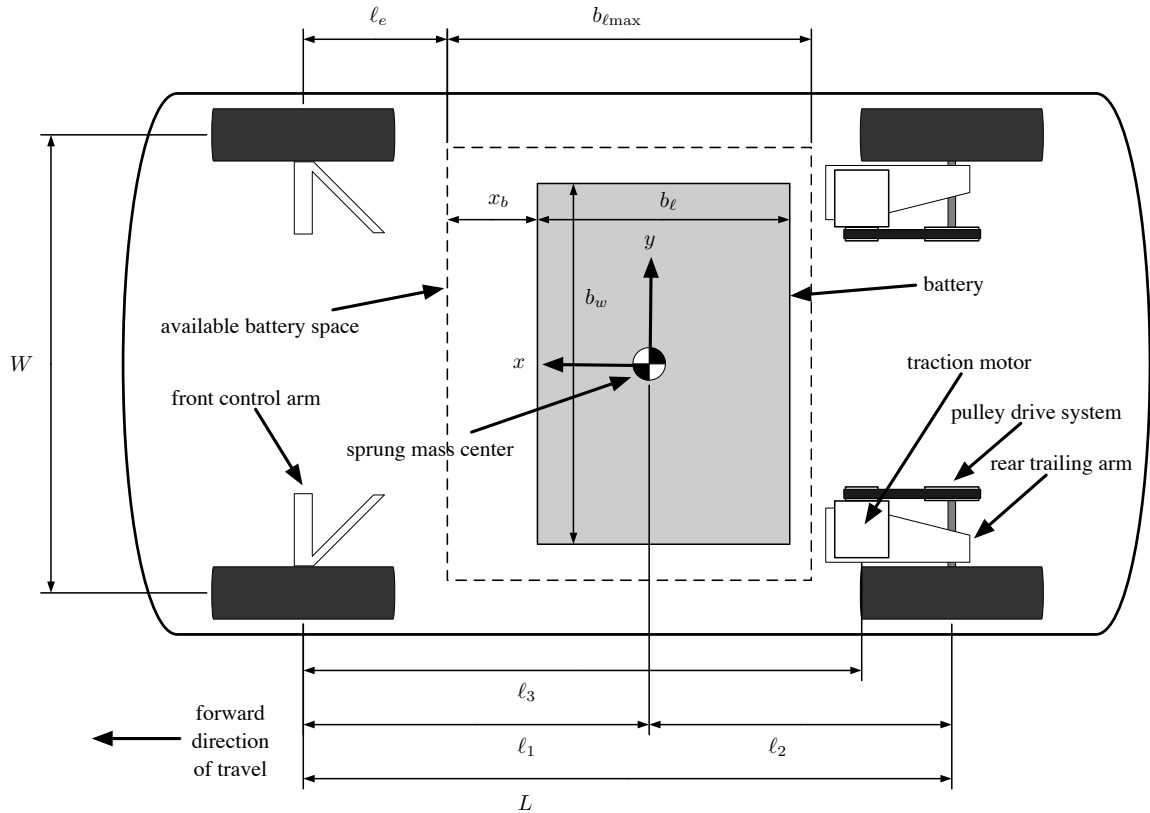


Figure 8.2 Top view of EV component layout

All vehicles are complex systems with numerous interactions. A large experience base and associated design rules are available to aid the development of conventional vehicles. HEVs and EVs lack this advantage, but more sophisticated design techniques to analyze and explicitly account for interactions can help compensate and support successful design of these new vehicles. An integrated vehicle design approach is required that simultaneously considers multiple vehicle subsystems and models the influence each subsystem has on the others.

An EV design model is presented in this chapter that demonstrates one approach to integrated vehicle design. Several important interactions are included in the model; they are illustrated in Fig. 8.3. The model consists of four analysis functions: powertrain analysis (\mathbf{a}_1), vehicle dynamics (\mathbf{a}_2), structural analysis (\mathbf{a}_3), and packaging and mass distribution analysis (\mathbf{a}_4). The powertrain analysis predicts energy consumption, range, and acceleration, and requires vehicle mass and inertia properties as input. The vehicle dynamics analysis calculates metrics for passenger comfort, handling, and suspension working space, and also uses vehicle mass and inertia properties. The dashed line between powertrain and vehicle dynamics functions indicates the presence of shared design variables between these

functions. The powertrain analysis includes dynamic coupling between the powertrain and vehicle suspension, and requires suspension design variables as input. The structural analysis computes frame stress and deflection under bending and torsion loads, and requires suspension forces and battery properties as input. The packaging and mass distribution function takes in physical properties of the other vehicle systems, computes overall vehicle mass and inertia properties, and evaluates packaging criteria. Many other interactions and subsystems could be considered; this design model is a foundational effort for integrated vehicle design, and serves as a starting point for more sophisticated future efforts.

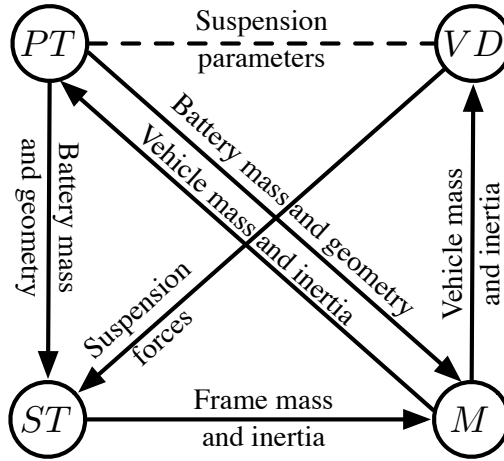


Figure 8.3 Relationships between analysis functions in the EV design problem

With the exception of the finite element model used for the structural analysis, the models for each of the analysis functions were independently developed. Current commercial CAE software do not support vehicle design efforts with the level of vehicle system integration required for this case study. Full control over models enabled the development of a design problem that allowed for management of the desired shared and coupling variables.

The powertrain, vehicle dynamics, structural, and packaging and mass distribution analysis functions are:

$$[b_m, b_w, b_\ell, \text{mpg}_e, \tau_V, \omega_V, P_V, t_{60}, R, C_b] = \mathbf{a}_1(m_s, h, \ell_1, I_y, k_s, c_s, p_r, \ell_s, r_m, n_c, R_r, b_I, b_W, b_L) \quad (8.1a)$$

$$[F_s, D_s, t_r, \delta_W, F_t, a_z] = \mathbf{a}_2(m_s, \ell_1, I_z, k_s, c_s) \quad (8.1b)$$

$$[l_f, h_f, m_f, I_{yf}, I_{zf}, \sigma_{fb}, \sigma_{ft}, K_b, K_t] = \mathbf{a}_3(F_s, b_m, b_w, b_\ell, d_f, t_f) \quad (8.1c)$$

$$[m_s, \ell_1, h, I_y, I_z, g_{p1}, g_{p2}] = \mathbf{a}_4(l_{fr}, h_f, m_f, I_{yf}, I_{zf}, b_m, b_w, b_\ell, x_b) \quad (8.1d)$$

The design variables that appear in Eqs. (8.1) are described in Table 8.1 along with their lower and upper bounds, and the coupling variables are listed in Table 8.2. These quantities

are described in more detail in subsequent sections. The remaining quantities that appear in Eqs. (8.1), but not in Tables 8.1 or 8.2, are design objectives or constraints, and will be described with the design problem formulation.

Table 8.1 EV design variables

variable	lower bnd.	upper bnd.	description
$x_1 = k_s$	5000	27,000	suspension stiffness per wheel (N/m)
$x_2 = c_s$	1500	4000	suspension damping rate per wheel (Ns/m)
$x_3 = p_r$	0.60	4.50	powertrain speed reduction ratio
$x_4 = \ell_s$	0.050	0.20	electric motor rotor axial length (m)
$x_5 = r_m$	0.09	0.13	electric motor rotor radius (m)
$x_6 = n_c$	8	22	electric motor winding turns per coil
$x_7 = R_r$	0.05	0.20	electric motor rotor resistance (Ω)
$x_8 = b_I$	0.70	2.0	battery electrode thickness scale
$x_9 = b_W$	0.50	2.75	battery electrode width scale
$x_{10} = b_L$	15	30	number of battery cell windings
$x_{11} = d_f$	0.010	0.060	frame member diameter (m)
$x_{12} = t_f$	0.00075	0.002	frame member wall thickness (m)
$x_{13} = x_b$	0.0	0.50	longitudinal battery location (m)

Table 8.2 EV coupling variables

m_s	sprung vehicle mass (kg)
h	height of sprung mass center above ground (m)
ℓ_1	distance between front axle and sprung mass center (m)
I_y	sprung mass pitch moment of inertia ($\text{kg}\cdot\text{m}^2$)
I_z	sprung mass yaw moment of inertia ($\text{kg}\cdot\text{m}^2$)
b_m	battery mass (kg)
b_w	battery width (m)
b_ℓ	battery length (m)
F_s	maximum suspension force (N)
m_f	frame mass (kg)
h_f	height of frame mass center above ground (m)
ℓ_f	distance between front axle and frame mass center (m)
I_{yf}	frame mass pitch moment of inertia ($\text{kg}\cdot\text{m}^2$)
I_{zf}	frame mass yaw moment of inertia ($\text{kg}\cdot\text{m}^2$)

The relationships between the analysis functions for the EV problem are summarized in its reduced adjacency matrix:

$$\mathbf{A}_5 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The EV design problem is to determine motor, battery, and frame geometry, as well as suspension parameters such that average energy consumption during urban travel is minimized, subject to constraints on acceleration performance, battery and motor feasibility, vehicle range, passenger comfort, handling, frame rigidity and stress, and packaging requirements. The MDF formulation is:

$$\begin{aligned}
& \min_{\mathbf{x}} && 1/\text{mpg}_e \\
& \text{subject to} && g_1 = \tau_V \leq 0 \\
& && g_2 = \omega_V \leq 0 \\
& && g_3 = t_{60} - t_{60\text{max}} \leq 0 \\
& && g_4 = R_{\text{min}} - R \leq 0 \\
& && g_5 = P_V \leq 0 \\
& && g_6 = C_b - C_{b\text{max}} \leq 0 \\
& && g_7 = -D_s \leq 0 \\
& && g_8 = t_r - t_{r\text{max}} \leq 0 \\
& && g_9 = \delta_W - \delta_{W\text{max}} \leq 0 \\
& && g_{10} = F_{t\text{min}} - F_t \leq 0 \\
& && g_{11} = a_z - a_{z\text{max}} \leq 0 \\
& && g_{12} = d_f - t_f/2 \leq 0 \\
& && g_{13} = \sigma_{fb} - \sigma_Y \leq 0 \\
& && g_{14} = \sigma_{ft} - \sigma_Y \leq 0 \\
& && g_{15} = K_{b\text{min}} - K_b \leq 0 \\
& && g_{16} = K_{t\text{min}} - K_t \leq 0 \\
& && g_{17} = g_{p1} = b_w - b_{w\text{max}} \leq 0 \\
& && g_{18} = g_{p2} = x_b + b_\ell - b_{\ell\text{max}} \leq 0
\end{aligned} \tag{8.2}$$

The objective function and the first six design constraints are computed by the powertrain analysis function. The objective is to minimize the average electrical energy consumed per mile, expressed as $1/\text{mpg}_e$, where mpg_e is the average number of miles the EV can travel on the electrical energy equivalent of one gallon of gasoline under urban driving conditions. A grid-to-wheels measure of electrical energy is used so that it can be compared to the familiar tank-to-wheels fuel economy metric of miles per gallon for conventional vehicles. The first

two constraints, motor torque and speed violation metrics, ensure that the electric motor is capable of powering the EV through the desired drive cycle without exceeding torque or speed limitations. The third constraint ensures adequate acceleration performance from 0 to 60 miles per hour (26.8 m/s). The fourth constraint requires the vehicle urban driving range R to achieve a minimum value without risking battery damage, and the fifth constraint requires that the battery power capacity is adequate to supply power needs throughout the driving and acceleration simulations. The sixth constraint is an upper bound on battery capacity C_b , set as an indirect cost constraint. It was found that EV geometry and dynamics allowed for a large enough battery to provide a range of several hundred miles, but the cost of such a large battery was prohibitive. Limiting battery capacity curbs battery expense.

The constraints $g_7 - g_{11}$ are computed by the vehicle dynamics analysis function. D_s is a measure of directional stability, and must be greater than zero. The rise time for a step steering maneuver is t_r , and it should be kept below $t_{r\max}$ to ensure a responsive steering system. The working space, or rattle space, of the suspension is δ_W , and must be less than $\delta_{W\max}$ due to kinematic suspension limitations. The minimum tire contact force is F_t ; it must be positive for model validity, and $F_{t\min}$ may be set to a positive force value as a roadholding requirement. The root mean square power spectral density of the sprung mass vertical acceleration while traveling over a moderately rough road is a_z , and limiting this value helps improve passenger comfort.

The constraints $g_{12} - g_{16}$ are evaluated by the structural analysis function. The first constraint ensures geometric compatibility of the frame members, and the next two ensure structural integrity of the frame during bending and torsion tests. Constraints g_{15} and g_{16} require that the frame achieves a certain level of bending and torsional stiffness.

The final two constraints are calculated by the packaging and mass distribution function, and relate to battery packaging. It is assumed that the battery is centered laterally on the vehicle, and that there is no room for vertical battery location adjustment. The ‘battery box’ is the volume within the vehicle, shown by the dashed lines in Fig. 8.2, that the battery must be contained in. Constraint g_{17} requires that the battery does not exceed the lateral bounds of the battery box, and g_{18} requires that the battery stays within the longitudinal bounds.

The constraints in the MDF formulation use several parameters; the values for these quantities are listed in Table 8.3. Initial optimization studies have identified feasible vehicle designs with urban equivalent fuel economy values of up to 190 mpg_e, including air conditioning and other accessory power loads. Solution of the AiO and partitioned EV design problems is part of ongoing work.

8.2 Powertrain Model

The powertrain model predicts vehicle range, acceleration, and energy efficiency over a given drive cycle. A detailed electric motor model is used that characterizes a motor based on geometric design variables. A backward-looking Simulink™ model was developed to simulate powertrain performance. This model incorporates the motor model, an empirical tire model, a detailed lithium-ion battery model, and a pitch-plane vehicle dynamics model that captures dynamic interactions between powertrain and chassis. Figure 8.4 provides a simplified overview of the powertrain model.

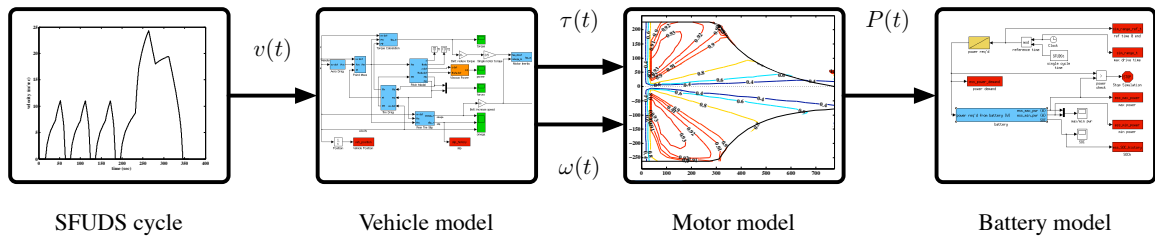


Figure 8.4 Simplified overview of EV powertrain model

At each time step of the powertrain simulation the longitudinal vehicle velocity $v(t)$ is obtained from a specified drive cycle. The simplified urban drive schedule (SFUDS) is the drive cycle used here [90]. The vehicle model then computes the electric motor torque $\tau(t)$ and speed $\omega(t)$ required to achieve the prescribed vehicle velocity, accounting for aerodynamic drag, tire slip and rolling resistance, and dynamic interaction between the powertrain and vehicle pitch motion. These motor torque and speed values are then used with a static power loss map to determine the electric power $P(t)$ that must be supplied to

Table 8.3 EV design constraint parameters

$t_{60\max}$	10.0 sec	maximum 0-60 acceleration time
R_{\min}	100 miles (161 km)	minimum urban range
$C_{b\max}$	250 Ah	maximum battery capacity
$t_{r\max}$	1.25 sec	maximum step steer rise time
$\delta_{W\max}$	0.16 m	maximum working space
$F_{t\min}$	0.0 N	minimum tire contact force
$a_{z\max}$	0.80 g	maximum discomfort metric
σ_Y	350 MPa	frame yield stress
$K_{b\min}$	6,000 kN/m	minimum frame bending stiffness
$K_{t\min}$	12,000 Nm/deg	maximum frame torsional stiffness
$b_{w\max}$	1.20 m	maximum battery width
$b_{\ell\max}$	1.05 m	maximum battery length

the motor power inverter by the battery. The motor power loss map is constructed by an induction motor model that depends on geometric motor design variables. The battery model accounts for the dynamic effects of the power demand $P(t)$, and computes the state of charge and the maximum charge and discharge power levels at each time point. The battery state at the end of the driving simulation is then used as a starting point for the charging simulation, which predicts the total energy required to recharge the battery back to its original state. This grid-to-wheels energy consumption is used to calculate fuel economy, which is the objective function for the EV design problem. Since there is no feedback between the battery and vehicle simulations, the battery simulation can be executed after the vehicle simulation is complete. This decoupled approach reduces simulation time significantly.

8.2.1 Vehicle Model

Figure 8.5 illustrates the components of the vehicle portion of the powertrain model. This model computes the motor torque and speed required to achieve the desired vehicle velocity and acceleration at each time step. Each component of this model is detailed below.

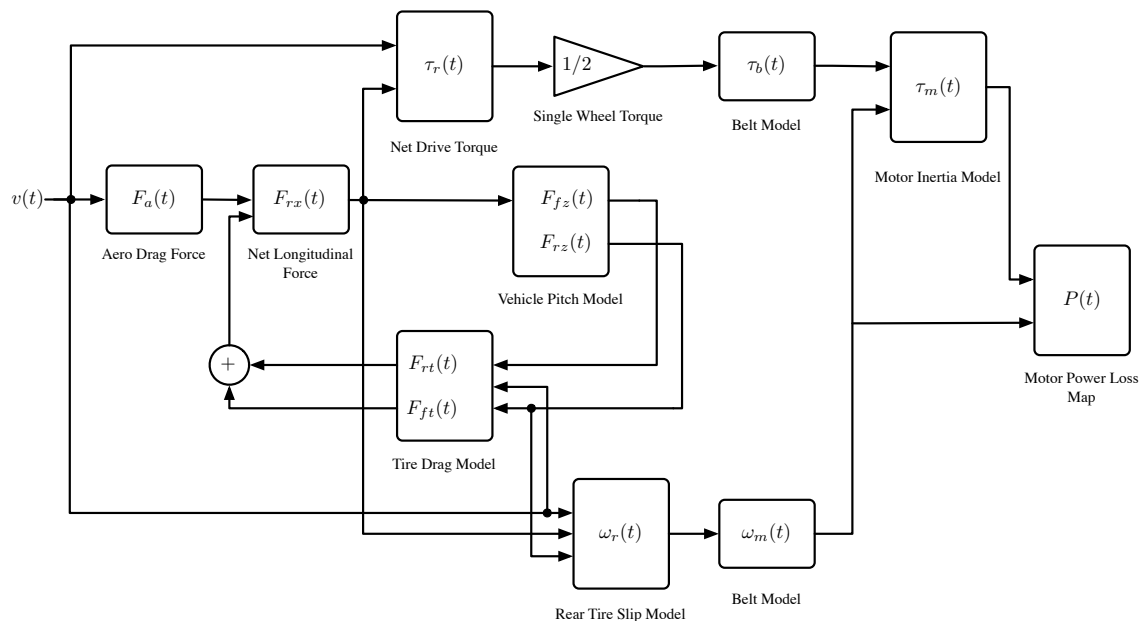


Figure 8.5 Block diagram of dynamic vehicle model

The SFUDS velocity profile used here is similar to the federal urban drive schedule (FUDS) used in U.S. urban fuel economy estimates, but lasts for only 360 seconds, whereas FUDS lasts for 1500 seconds. These two cycles have the same average speed and maximum acceleration and braking values [90]. The SFUDS profile, illustrated in Fig. 8.6, is used to assess EV range and vehicle energy consumption over that range.

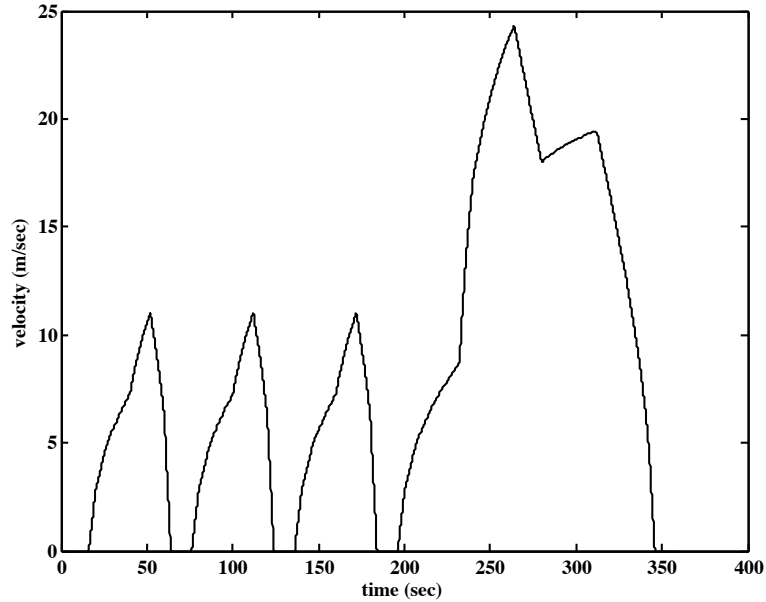


Figure 8.6 Simplified federal urban drive schedule

The powertrain model also evaluates the time required for the vehicle to accelerate from zero mph to sixty mph (t_{60}). A forward-looking version of the vehicle model is used to compute the acceleration time. As this simulation progresses the motor provides maximum available torque (which depends on motor speed), starting at zero velocity until the vehicle reaches sixty mph (26.8 m/sec). The simulation time is recorded, as well as the power consumption to determine if the battery can meet power needs for this acceleration test.

Aerodynamic Drag Model

The aerodynamic drag force F_a is computed as a function of longitudinal velocity, given by Eq. (8.3). The frontal area A_f and drag coefficient C_d are fixed parameters since the exterior vehicle dimensions do not change in this design problem. The density of air is ρ_a .

$$F_a = \frac{1}{2} C_d \rho_a A_f v^2 \quad (8.3)$$

Net Longitudinal Force

The net longitudinal force F_{rx} required to move the vehicle at the desired velocity is computed using a simple point-mass model, shown in Eq. (8.4).

$$F_{rx} = m\dot{v} + F_t + F_a \quad (8.4)$$

The total mass of the vehicle (m) is the sum of the sprung mass (m_s) and unsprung mass (m_{us}). Sprung mass includes all vehicle components completely supported by the suspension; unsprung mass includes components that move with the suspension. The force required to move the tires at the desired velocity profile is F_t , which includes rolling resistance and rotational inertia effects. F_t is composed of front and rear resistance terms:

$$F_t = F_{ft} + F_{rt}$$

Net Drive Torque

The net drive torque model calculates the torque that must be supplied by the electric motors to the rear wheels to match the desired velocity profile.

$$\tau_r = r_t(v)F_{rx} \quad (8.5)$$

The dynamic loaded radius of the tires $r_t(v)$ depends on the vehicle longitudinal velocity, and is estimated using a quadratic model based on empirical data. Tire properties were obtained for a fictitious low rolling resistance tire appropriate for this vehicle [72]:

$$r_t(v) = C_{t1} + C_{t2}v + C_{t3}v^2 \quad (8.6)$$

Vehicle Pitch Model

The vehicle model accounts for the dynamic coupling between the powertrain and the pitch motions of the electric vehicle, i.e., rotations about the y axis. This is accomplished using a two degree of freedom (DOF) state space model of the vehicle in the pitch plane, illustrated in Fig. 8.7. The pitch angle is θ_p , and the vehicle vertical position z is the vertical displacement of the center of mass from its equilibrium position. The distance between the center of mass and the front and rear axles is ℓ_1 and ℓ_2 , respectively. The mass of the vehicle supported by the suspension, i.e., the sprung mass, is m_s , and the pitch moment of inertia is I_y . The front and rear suspension forces on the sprung mass are F_{fz} and F_{rz} , respectively. Each of these forces is the sum of spring and damping forces for its respective axle. Spring forces are proportional to displacement, and damping forces are proportional to velocity.

The suspension stiffness and damping rates at each wheel are k_s and c_s , respectively. The equivalent stiffness and damping rates at the front axle are $k_f = 2k_s$ and $c_f = 2c_s$, and the rates at the rear axle are $k_r = 2k_s$ and $c_r = 2c_s$. The equations of motion for this system were linearized, and a state space model was derived. The four states required for this two DOF, second order system are z and θ_p , and their time derivatives \dot{z} and $\dot{\theta}_p$.

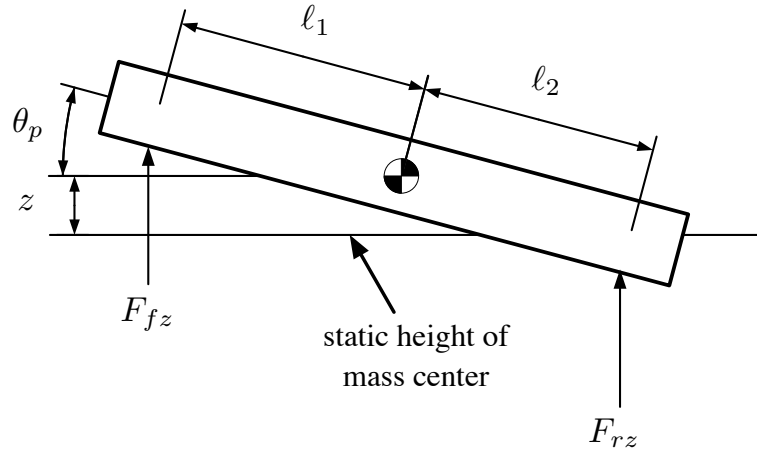


Figure 8.7 2 DOF vehicle pitch model

$$\begin{bmatrix} \dot{z} \\ \dot{\theta}_p \\ \ddot{z} \\ \ddot{\theta}_p \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -\frac{k_f+k_r}{m_s} & \frac{\ell_2 k_r - \ell_1 k_f}{m_s} & -\frac{c_f+c_r}{m_s} & \frac{\ell_2 c_r - \ell_1 c_f}{m_s} \\ \frac{\ell_2 k_r - \ell_1 k_f}{I_y} & -\frac{\ell_2^2 k_r + \ell_1^2 k_f}{I_y} & \frac{\ell_2 c_r - \ell_1 c_f}{I_y} & -\frac{\ell_2^2 c_r - \ell_1^2 c_f}{I_y} \end{bmatrix} \begin{bmatrix} z \\ \theta_p \\ \dot{z} \\ \dot{\theta}_p \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ M_p/I_y \end{bmatrix} \quad (8.7)$$

The input to the state space model is the pitch moment $M_p = F_{rx}(h + z)$ normalized with the pitch moment of inertia I_y , where h is the static height of the vehicle mass center above the ground. Since z is computed by the state space model, this relationship forms an algebraic loop. The suspension normal forces F_{fz} and F_{rz} are the desired outputs of the pitch model, and can be computed using state variable values.

$$F_{fz} = -2(k_f(z + \theta_p \ell_1) + c_f(\dot{z} + \dot{\theta}_p \ell_1)) \quad (8.8)$$

$$F_{rz} = -2(k_r(z - \theta_p \ell_2) + c_r(\dot{z} - \dot{\theta}_p \ell_2)) \quad (8.9)$$

Tire Drag Model

The tire drag model calculates the force required to move the vehicle tires at the prescribed velocity and acceleration. This resistance is due to tire rolling resistance and spin inertia. The angular velocity and acceleration of front and rear tires are assumed equal here to avoid an additional algebraic loop in the simulation, i.e., $\dot{\omega}_f = \dot{\omega}_r = \dot{v}_t(v)$. The tire drag forces are calculated using the following equations:

$$F_{ft} = F_{fz} C_r + \dot{v} I_y t \quad (8.10)$$

$$F_{fr} = F_{rz}C_r + \dot{v}I_{yt} \quad (8.11)$$

where C_r is the tire rolling resistance (assumed constant) [72], and I_{yt} is the rotational inertia of two wheel and tire assemblies. Note that the tire radius cancels from the equations.

Tire Slip Model

Any vehicle tire with a net torque will incur some amount of slip i , which means that the tire rotational velocity will differ from $r(v)/v$. Tire slip, as defined by Wong [147], varies from 0 when the tire angular velocity is equal to $r(v)/v$, to 1, when $v = 0$ and the tire angular velocity is positive:

$$i = 1 - \frac{v}{\omega_r r(v)} \quad (8.12)$$

where ω_r is the angular velocity of the driven wheel, which is the rear wheel for the EV. An alternative model is used here that is a linearization of Eq. (8.12), and simplifies wheel speed calculations under braking:

$$\omega_r = \frac{v(i+1)}{r(v)} \quad (8.13)$$

When the wheels are locked under braking, this model gives a slip value of $i = -1$. Tire slip depends on several factors, including tire normal and longitudinal forces. A lookup table based on empirical data for the same fictitious tire described above was used to estimate tire slip as a function of tire forces, enabling wheel speed calculation using Eq. (8.13). The data from this lookup table is plotted in Fig. 8.8

Belt Drive Model

Each rear wheel is driven by a separate electric motor via a synchronous drive belt. A speed reduction between the motor and wheel enables the use of smaller motors. The largest pulley that is geometrically compatible with the rear suspension is a 72 groove H pulley, with a radius of $r_w = 0.1455$ m. The smallest drive pulley that still maintains at least six teeth in contact with the belt is a 16 groove H pulley with a radius of $r_w = 0.0323$ m [108]. This limits the maximum speed reduction ratio to 4.5. The speed reduction ratio of the pulley is a design variable, and is given by:

$$p_r = \frac{\omega_m}{\omega_w} = \frac{r_w}{r_m} \quad (8.14)$$

where the motor and wheels speeds are ω_m and ω_w , respectively, and r_m is the drive pulley

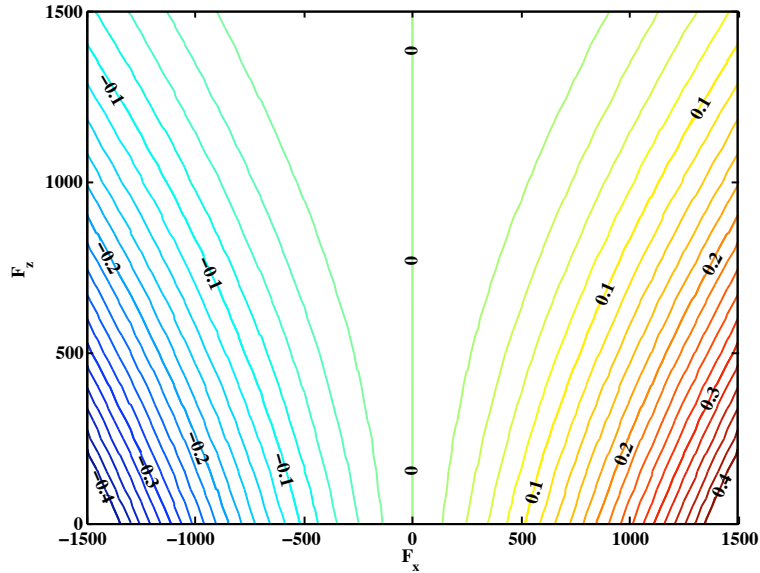


Figure 8.8 Slip data for electric vehicle tire

radius. This equation is applied appropriately in the vehicle model to calculate required motor torque and speed. The belt model is simplified by assuming no power transmission loss and zero belt compliance. A compliant belt model was considered, but the difference in results was found to be negligible.

The model parameters used in the vehicle model described above are summarized in Table 8.4. Note that some model values are neither design variables nor parameters, and have not been defined yet (e.g., h). These quantities are coupling variables, and will be defined shortly.

Table 8.4 Vehicle model parameters

C_d	0.30	drag coeff.	C_{t1}	0.240	constant r_t parameter
A_f	1.70 m	frontal area	C_{t2}	$2.57 \cdot 10^{-7}$	linear r_t parameter
ρ_a	1.20 kg/m ³	air density	C_{t3}	$2.55 \cdot 10^{-6}$	quadratic r_t parameter
I_{yt}	0.72 kg-m ²	tire inertia	C_r	0.0069	tire rolling resistance
m_{us}	154 kg	unsprung mass			

8.2.2 Induction Motor Model

The electric vehicle is propelled by two electric motors, or electric drive units, that consist of both an electric machine and a power inverter and controller. The electric machine converts electrical power into rotational mechanical power. The power inverter converts direct current (DC) electrical power to alternating current (AC) electrical power as required by the electric

machine. The rotor of an electric machine is connected to the output shaft, and is encased by a cylindrical stator.

Two types of motors are in popular use for electric and hybrid electric vehicles: permanent magnet brushless direct current (BLDC) motors, and AC induction motors (IMs). Each type employs a stator that produces a rotating magnetic field that interacts with the rotor's magnetic field, causing the rotor to spin. The stator typically is wound with three phases, and is supplied with three-phase AC power. The rotor in a BLDC machine is constructed of permanent magnet material, providing a continuous magnetic field. In contrast, an IM rotor is constructed with a stack of iron sheets, with conducting bars running through the rotor parallel to the output shaft. Conductive end rings electrically short these conducting bars. When the rotor is subjected to the stator's rotating magnetic field, electric currents are induced in the conducting bars, creating a rotating magnetic field of their own. A simplified diagram of an IM is shown in Fig. 8.9. BLDC motors are characteristically more efficient than IMs, but are more expensive due to permanent magnet material. A three-phase IM was chosen as the electric drive unit for the EV. A detailed model was developed that constructs the motor power loss map based on geometric motor design variables.

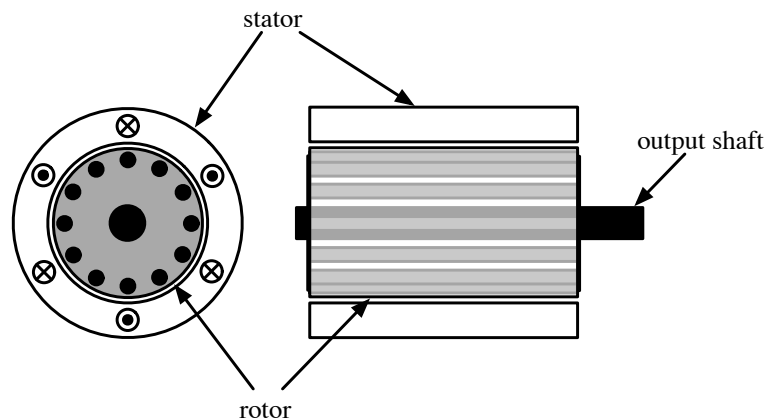


Figure 8.9 Diagram of an induction motor

Equivalent Circuit

The dynamic electrical behavior of an IM can be modeled using a equivalent circuit [24]. After several simplifications, the resulting equivalent circuit for a single phase of the IM consists of three inductance elements and two resistance elements, one of which is variable (Fig. 8.10). The mutual inductance between the rotor and stator is L_m ; the stator winding resistance is R_s ; the stator and rotor leakage inductances are L_{ls} and L_{lr} , respectively, and the electrical resistance through the rotor conductors is R_r . The AC power source has a root mean square voltage (RMS) of V_s .

The rotor conductors do not have a direct electrical connection to the stator, but rather

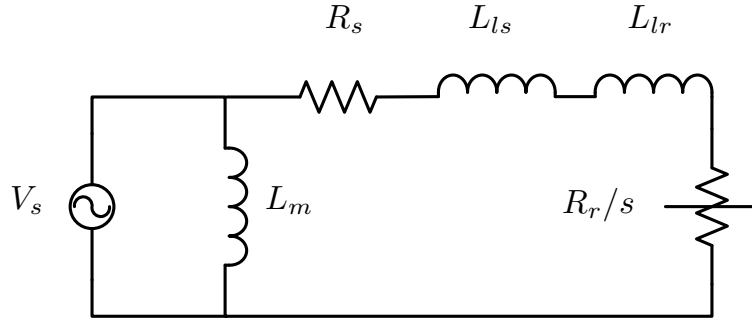


Figure 8.10 Equivalent circuit of an induction motor

have an indirect influence through electromagnetic interaction. The stator and rotor are coupled in a manner similar to the primary and secondary windings of a transformer. As with transformers, we can include the secondary windings in a single equivalent circuit if their properties are viewed in the reference frame of the primary circuit. In the IM equivalent circuit we do this by dividing the rotor resistance by the *slip* between the stator and rotor magnetic fields. The stator power supply frequency ω_e determines the speed at which the stator magnetic field rotates, and the rotor magnetic field rotation lags the stator field if there is any load on the motor. This lag is quantified by slip s , defined as:

$$s = \frac{\omega_e - \omega_r}{\omega_e} \quad (8.15)$$

where ω_r is the rotor electrical speed. The slip depends on supply voltage and frequency, motor construction, and load. Slip, or misalignment between the magnetic fields, gives rise to torque. If slip is zero, the magnetic fields are aligned, and no torque can be produced. At zero slip the motor is at synchronous speed, and as torque is applied slip increases monotonically until the breakdown torque T_{em} is reached. When $s = 1$ the motor is stalled, i.e., the rotor is stationary with the stator magnetic field rotating.

Motor Property Calculation

The dynamic motor model requires knowledge of several values that characterize an induction motor, such as the rotor inertia, frictional losses, inductance, electrical resistance of stator windings, and the maximum stator current. These values may be approximated using simple formulae. The estimated rotor mass is:

$$m_r = \pi r_m^2 \ell_s \rho_{fe} \quad (8.16)$$

where r_m is the rotor radius, ℓ_s is the rotor stack length, and ρ_{fe} is the density of iron. These

and other parameter values used in the induction motor model are summarized in Table 8.5. The rotational inertia of the rotor is approximately:

$$J_r = \frac{m_r r_m^2}{2} \quad (8.17)$$

A viscous friction model is used to approximate the parasitic torque on the motor from resistance in the bearings:

$$\tau_{\text{loss}}(J_r) = \omega_m c_m \quad (8.18)$$

where ω_m is the rotational speed of the motor output shaft in radians per second, and c_m is the viscous friction coefficient. A strong correlation was observed between J_r and c_m , and an exponential model was fit to empirical motor data:

$$c_m = C_{m1} \left(1 - \frac{C_{m2}}{e^{C_{m3} J_r}} \right) + C_{m4} J_r \quad (8.19)$$

The model parameters for this and other equations in this section are defined in Table 8.5. The mutual inductance is calculated from motor geometry using a model due to Amin [11]. The number of stator slots (N_s) and stator windings per phase (W_1) must first be calculated:

$$N_s = 2p_1 q m_1 \quad (8.20)$$

$$W_1 = 2p_1 q n_c \quad (8.21)$$

where p_1 is the number of pole pairs in the stator (i.e., the number of poles p divided by 2), q is the number of stator slots per phase per pole, m_1 is the number of phases, and n_c is the number of turns per coil. The mutual inductance is:

$$L_m = \frac{6\mu_0 W_1^2 r_m \ell_s}{\pi p_1^2 \delta_g} \quad (8.22)$$

where the effective air gap δ_g , adjusting for geometry and slot effects, is given by:

$$\delta_g = 0.06r_m - .0025 \quad (8.23)$$

The stator leakage inductance (L_{ls}) was observed to have a correlation with motor aspect ratio (ℓ_s/r_m) and mutual inductance in empirical data for several IMs. The following

relationship was derived for the stator leakage:

$$L_{ls} = L_m \left(0.07 - \frac{0.05}{1 + e^{(5.0 - \ell_s/r_m)/2}} \right) \quad (8.24)$$

In many IMs the rotor leakage inductance L_{lr} is close in value to the stator leakage inductance. This model is simplified by assuming these quantities are equal, and that the total leakage inductance is:

$$L_l = L_{ls} + L_{lr} = 2L_{ls} \quad (8.25)$$

The electrical resistance of the stator windings (R_s) is based on estimates for the winding radius (r_w) and total winding length (ℓ_w). Rather than specifying the outer stator radius as an independent design variable, it is assumed proportional to the rotor radius: $r_s = r_m(t_s + 1)$, where t_s is the stator radius proportionality factor. The winding radius and length are:

$$r_w = r_m \sqrt{\frac{n_a n_p ((t_s + 1)^2 - 1)}{W_1 m_1}} \quad (8.26)$$

$$\ell_w = 2\ell_s W_1 \quad (8.27)$$

and the stator winding resistance is:

$$R_s = \frac{k_e \rho_{cu} \ell_w}{\pi r_w^2} \quad (8.28)$$

where k_e is an end effects coefficient that accounts for additional winding length and resistance at the rotor stack ends, and ρ_{cu} is the resistivity of copper.

An important property of a motor is the maximum current (I_{sm}) the stator can tolerate before risk of failure. A quadratic model was fit to the maximum operating current of standard wire gauges:

$$I_{sm} = C_{I1} + C_{I2} d_w + C_{I3} d_w^2 \quad (8.29)$$

where $d_w = 2000r_w$ is the winding diameter in millimeters.

Power Loss Map Calculation

The power loss map is generated by stepping through numerous motor operating points and recording the steady state power consumption at those points. These raw data points are dispersed non-uniformly over the motor torque-speed space; interpolation is used to obtain power loss points over an evenly-spaced mesh. A few important aspects of the power loss map must be determined before computing power loss points. Figure 8.11 illustrates

a typical maximum torque curve for an IM. Note that the abscissa is ω_e , rather than the angular velocity of the output shaft ω_m . The ordinate is the developed torque τ_e , which does not account for any mechanical losses. The two regions of the maximum torque curve considered here are the constant flux region and the flux weakening region.

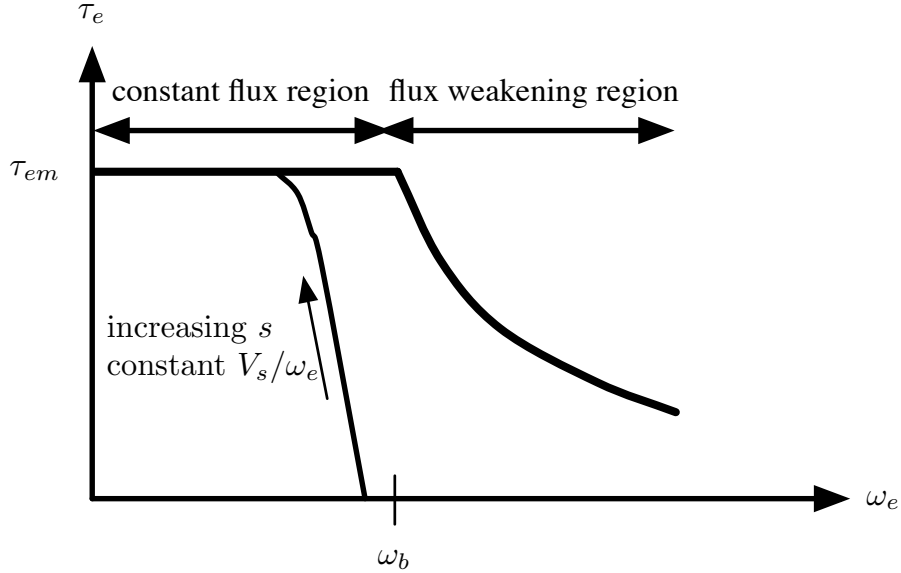


Figure 8.11 Typical IM maximum torque curve

The ratio V_s/ω_e is proportional to stator magnetic flux, and is held to a constant value of C_1 in the first region. The breakdown torque is given by:

$$\tau_{em} = \frac{3p}{4\omega_e} \cdot \frac{V_s^2}{\sqrt{R_s^2 + \omega_e^2 L_l^2 + R_s}} \quad (8.30)$$

The maximum torque is approximately constant in the constant flux region. The breakdown torque τ_{em} is limited by the maximum allowable stator current I_{sm} . If stator current were unlimited, C_1 could be increased to achieve arbitrarily high maximum torque values. This is not the case, so C_1 is set such that the stator current I_s is equal to I_{sm} when $\tau_e = \tau_{em}$ and ω_e is equal to the base speed ω_b . The base speed is the frequency at which the motor transitions from constant flux to constant power operation, which occurs when maintaining the ratio $V_s/\omega_e = C_1$ would require increasing V_s beyond the maximum power inverter voltage V_{sm} . Since V_s cannot be increased, the only way to increase speed beyond ω_b is to increase ω_e while holding V_s fixed at V_{sm} , reducing the ratio V_s/ω_e and stator magnetic flux. In the flux weakening region, under certain assumptions, the maximum torque curve follows an constant power isocurve.

The value of C_1 for a motor is obtained numerically by finding the value of ω_e that results in $I_s = I_{sm}$ when $V_s = V_{sm}$. This value of ω_e is the base speed ω_b . The stator current

for a given ω_e at V_{sm} can be found through analysis of the equivalent circuit in Fig. 8.10. The impedance of L_m is:

$$Z_1 = jL_m\omega_e \quad (8.31)$$

where j is the imaginary number $\sqrt{-1}$. The impedance of R_s , L_{ls} , L_{lr} , and R_r/s in series is:

$$Z_2 = (R_s + R_r/s_m) + j\omega_e(L_{ls} + L_{lr}) \quad (8.32)$$

The slip at torque breakdown s_m is used in evaluating Z_2 when computing the maximum stator current:

$$s_m = \frac{R_r}{\sqrt{R_s^2 + \omega_e^2 L_l^2}} \quad (8.33)$$

Other values of slip may be used in Eq. (8.32) when calculating stator current for operating points below maximum torque. The total circuit impedance is:

$$Z_T = \frac{Z_1 Z_2}{Z_1 + Z_2} \quad (8.34)$$

and at maximum power inverter voltage the stator current is:

$$I_{sm} = \frac{V_{sm}}{|Z_T|} \quad (8.35)$$

Now that ω_b is known, the base speed can be calculated in terms of rotor electrical frequency and output shaft rotational speed. First, Eq. (8.33) is used to calculate motor slip at ω_b and max torque (s_{mb}). The base rotor electrical frequency is:

$$\omega_{rb} = \omega_b(1 - s_{mb}) \quad (8.36)$$

and the base output shaft speed is:

$$\omega_{mb} = 2\omega_{rb}/p \quad (8.37)$$

The maximum torque in the constant flux region is almost constant torque, but not exactly due to stator effects. A small voltage adjustment is used to compensate for this:

$$V_s = \frac{\omega_e(V_{sm} - V_0)}{\omega_b} + V_0 \quad (8.38)$$

where V_0 is a voltage compensation parameter chosen such that the maximum torque in the constant flux region is as close to constant as possible. Four different operating regimes are considered in the power loss map calculation:

1. Forward motoring, constant flux: $\omega_m > 0$, $s > 0$, and V_s/ω_e constant
2. Forward motoring, flux weakening: $\omega_m > 0$, $s > 0$, V_s constant, and $\omega_e > \omega_b$
3. Forward regeneration, constant flux: $\omega_m > 0$, $s < 0$, and V_s/ω_e constant
4. Forward regeneration, flux weakening: $\omega_m > 0$, $s < 0$, V_s constant, and $\omega_e > \omega_b$

In the regenerative regimes the stator electrical frequency lags the rotor electrical frequency, resulting in negative slip, causing the motor to perform as a generator. During regeneration the vehicle is slowed and the motor converts mechanical kinetic energy into electrical energy, which may be stored for later use. The power loss map is different in the forward motoring and regeneration regimes for two primary reasons: frictional losses and stator resistance. Power loss values for the forward motoring, constant flux regime are obtained by stepping through ω_e values from just above zero through ω_b . At each value for ω_e the value of V_s is found using Eq. (8.38), and breakdown torque and output shaft speed is recorded. At each speed point we step through slip values from just above zero to s_m , which is found using Eq. (8.33). At each ω_e and s point, we evaluate the developed torque:

$$\tau_e = \frac{3pR_r}{2s\omega_e} \cdot \frac{V_s^2}{(R_s + R_r/s)^2 + \omega_e^2 L_l^2} \quad (8.39)$$

The output shaft speed is recorded for each of these points, and the net output torque is found:

$$\tau_{\text{net}} = \tau_e - \omega_m c_m \quad (8.40)$$

where c_m is evaluated using Eq. (8.19). The net mechanical output power is $P_{\text{out}} = \tau_{\text{net}}\omega_m$, and can be used to evaluate motor efficiency at that operating point. The stator current is evaluated using Eqs. (8.31) – (8.35), where V_{sm} and s_m are replaced with V_s and s , respectively. The electrical input power to the motor is:

$$P_{\text{in}} = m_1 I_s V_s \cos(\angle Z_T) \quad (8.41)$$

where $\angle Z_T$ is the angle between the imaginary and real parts of Z_T . The power factor is $\cos(\angle Z_T)$, and accounts for the influence of inductive elements on power consumption. In the forward regeneration, constant flux regime, the same values of ω_e are used, but slip is varied from just below zero to slip at breakdown speed (which is negative). Equation (8.39) is used again to evaluate developed torque at each point. Frictional losses increase the magnitude of net torque in this case since $\tau_e < 0$. Electrical output power calculations are performed using the same equations as for forward motoring, but $P_{\text{in}} < 0$, indicating power output. The breakdown torque during regeneration is slightly different than for the forward

motoring case:

$$\tau_{emR} = \frac{3p}{4\omega_e} \cdot \frac{V_s^2}{\sqrt{R_s^2 + \omega_e^2 L_l^2} - R_s} \quad (8.42)$$

In both flux weakening regimes the stator voltage is fixed at V_{sm} , and the supply frequency is increased from ω_b to a sufficiently large value. The same relations are used to determine power consumption during forward motoring and power production during forward regeneration. Reverse motoring and reverse regeneration are not considered here since the velocity profiles of interest never specify reverse vehicle motion. The efficiency map for a sample motor design is shown in Fig. 8.12. Both the maximum and minimum torque curves are displayed, adjusted for frictional losses.

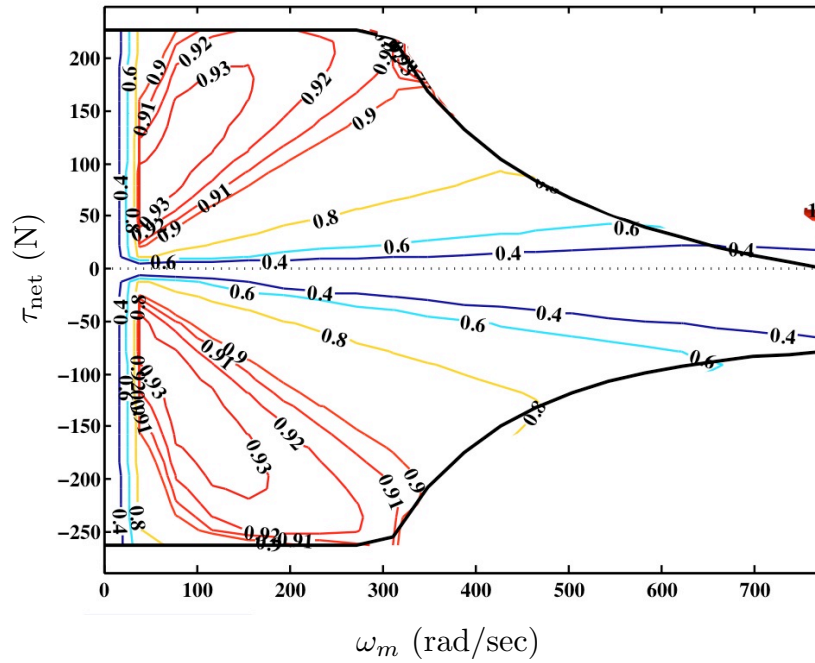


Figure 8.12 Example IM efficiency map

The power loss map for the same sample motor design is illustrated in Fig. 8.13. Power loss information outside the maximum and minimum torque curve envelope is not physically meaningful since the motor cannot operate at those torque and speed points. It is assumed here that the power inverter has perfect efficiency, so the power loss map does not account for power inverter losses. The sample motor was used in a powertrain simulation, and the points visited while following the SFUDS velocity profile are displayed on the map.

The maximum motor speed is limited by three considerations: viscous drag, structural integrity, and maximum inverter frequency. The maximum speed due to viscous drag (ω_{max1})

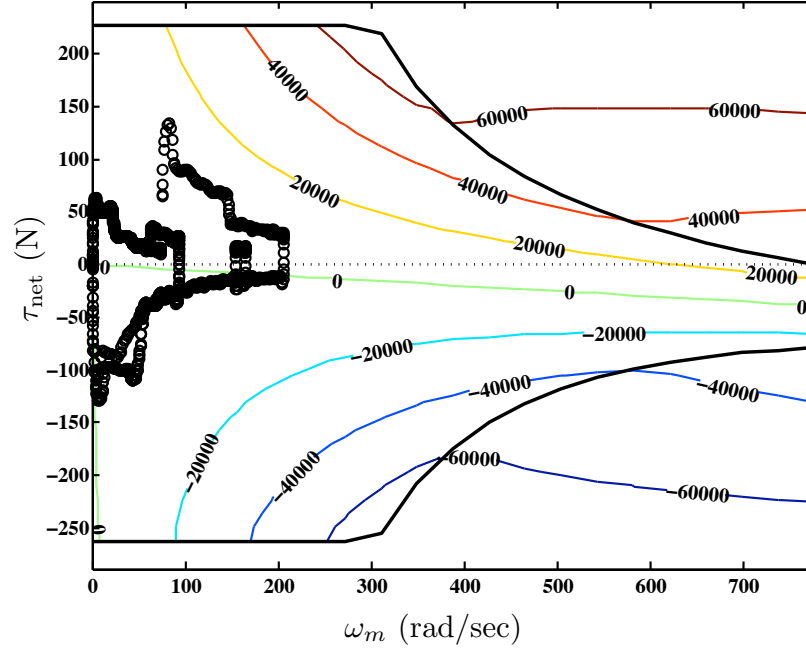


Figure 8.13 Example IM power loss map with points visited during SFUDS

is the point at which the maximum net torque curve intersects the motor speed axis, which is approximately 780 rad/sec in Fig. 8.13. The maximum speed that the motor can operate at safely without risking structural failure is estimated to be:

$$\omega_{\max 2} = \sqrt{\frac{8\sigma_{Yr}}{SFr_m^2\rho_{fe}(3+v)}} \quad (8.43)$$

where σ_{Yr} is the yield stress for the rotor material, SF is a safety factor, and v is Poisson's ratio for the rotor material. The maximum rotor speed that can be achieved before the power inverter exceeds its maximum frequency (ω_{inv}) is:

$$\omega_{\max 3} = \frac{2\omega_{\text{inv}}}{p} \quad (8.44)$$

The maximum rotor speed for a given motor is $\omega_{\max} = \min\{\omega_{\max 1}, \omega_{\max 2}, \omega_{\max 3}\}$, and a vehicle should be designed such that this value is never exceeded during anticipated vehicle usage. The speed violation constraint ω_v is ω_{\max} subtracted from the highest motor speed encountered during the powertrain simulations. Similarly, the torque violation constraint τ_v is a single value that expresses whether the required torque has exceeded the torque envelope defined by the maximum and minimum net torque curves. If $\tau_v \leq 0$, the motor is capable of supplying the necessary torque.

8.2.3 Lithium Ion Battery Model

The Lithium ion (Li-ion) battery provides all power for vehicle motion and accessory loads, including power steering and air conditioning. Battery design is the most important factor in EV performance [63], and important tradeoffs between power and energy density, durability, and safety must be addressed. A dynamic Li-ion battery model developed by Han [68], based on work by Doyle, Fuller, and Newman [46, 56], was used in the powertrain analysis function to assess the battery state of charge, and charge and discharge power limits, throughout the simulations.

The battery is composed of two pairs of battery packs; each pair is connected in series, and the two pairs are connected in parallel. Each pack is composed of four battery modules connected in series, and each module has twelve battery cells connected in series. Figure 8.14 illustrates the construction of each cell. An electrochemical reaction occurs in the separator between the negative and positive electrodes [121]. The rate of this reaction depends on both materials and cell geometry. Each electrode is backed by a current collector, and the electrodes are folded into a flat-wound cell arrangement. The battery design variables include the electrode thickness scale (b_l), which controls the thickness of the electrodes and separator, the battery width scale (b_w), which controls the electrode and cell width, and

Table 8.5 Motor model parameters

V_{sm}	460 V	max stator voltage
p	4	no. of stator poles
q	3	no. of slots per phase per pole
m_1	3	no. of motor phases
σ_{Yr}	300 MPa	rotor yield stress
ν	0.30 Pa	rotor Poisson's ratio
SF	4	rotor safety factor
ω_{inv}	1500 rad/sec	maximum inverter frequency
ρ_{fe}	7870 kg/m ³	iron density
C_{m1}	0.062	1st c_m parameter
C_{m2}	0.998	2nd c_m parameter
C_{m3}	0.940	3rd c_m parameter
C_{m4}	0.0513	4th c_m parameter
n_a	0.80	slot volume ratio
n_p	0.50	wire packing ratio
k_e	1.50	end effect coefficient
ρ_{cu}	$1.72 \cdot 10^{-8}$ Ω -m	copper resistivity
C_{I1}	0.0564	constant I_{sm} parameter
C_{I2}	-0.0237	linear I_{sm} parameter
C_{I3}	2.21	quadratic I_{sm} parameter

the number of cell windings or folds (b_L), which controls the thickness of each cell. The model includes allowance for packaging when estimating overall battery geometry and mass. The height of each cell is fixed such that the height of the battery is 11 cm, the maximum allowed by EV geometry.

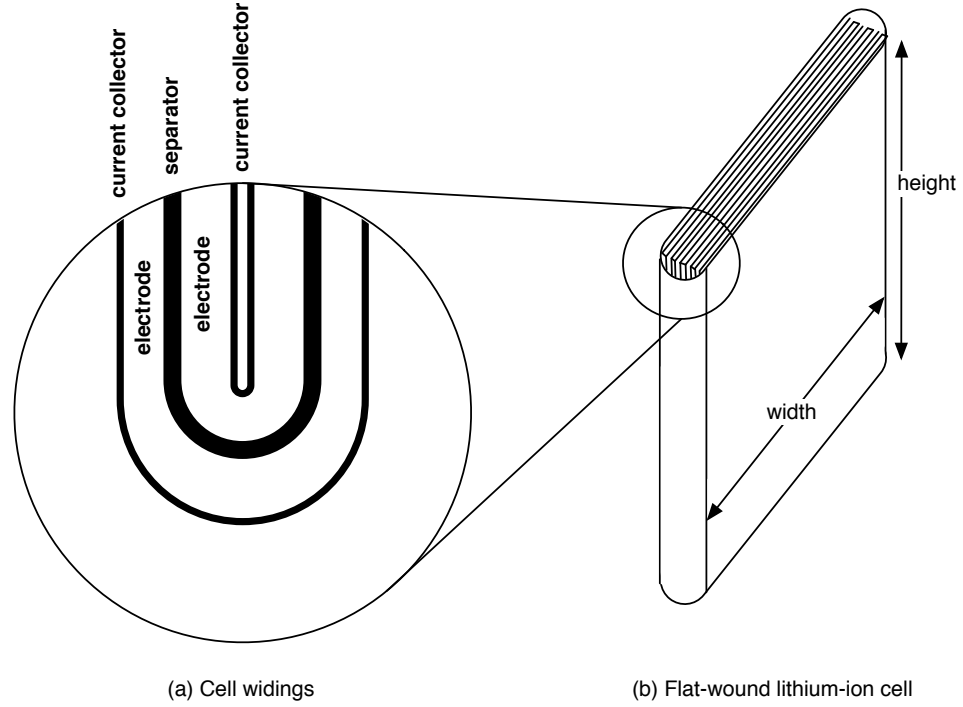


Figure 8.14 Flat-wound lithium-ion battery cell (after [68])

The dynamic battery model is a lumped-parameter model where the battery voltage is estimated by the equation:

$$v_{\text{net}}^{\text{bt}} = E^{\text{bt}} - R_0 I_1^{\text{bt}} - R_p I_p^{\text{bt}} \quad (8.45)$$

where E^{bt} is the battery open circuit voltage and R_0 and R_p are the cell internal ohmic and polarization resistances, respectively. The cell load and polarization currents are I_1^{bt} and I_p^{bt} , respectively, which can be determined using the differential equation:

$$\frac{dI_p^{\text{bt}}}{dt} = \frac{(I_1^{\text{bt}} - I_p^{\text{bt}})}{\tau_p} \quad (8.46)$$

where τ_p is the polarization time constant. The open-circuit voltage depends on material composition only, and its dependence on battery state of charge (SOC) can be approximated using the following polynomial:

$$E^{\text{bt}} = 4.03 \cdot \text{SOC}^4 - 11.96 \cdot \text{SOC}^3 + 11.99 \cdot \text{SOC}^2 - 3.53 \cdot \text{SOC} + 4.02 \quad (8.47)$$

The dependence of the remaining terms in Eq. (8.45) on SOC, and the polarization time constant, can be assessed using a hybrid pulse power characterization (HPPC) test [45]. The polarization resistance curve and time constant are nearly invariant apropos to SOC, so scalar values are used to represent these quantities. The ohmic resistance curve is discretized in this model, and is obtained for both charging and discharging conditions. The results of the HPPC test was modeled using an artificial neural network [70] to reduce simulation time for use in optimization.

No feedback exists from the battery model to the vehicle or motor model, so the dynamic battery model can be run independently of the vehicle model with the motor power demand curve $P(t)$ as input. Figure 8.16 shows an example battery output power curve. The gap between this curve and the mechanical motor output power curve illustrates power loss through the motor.

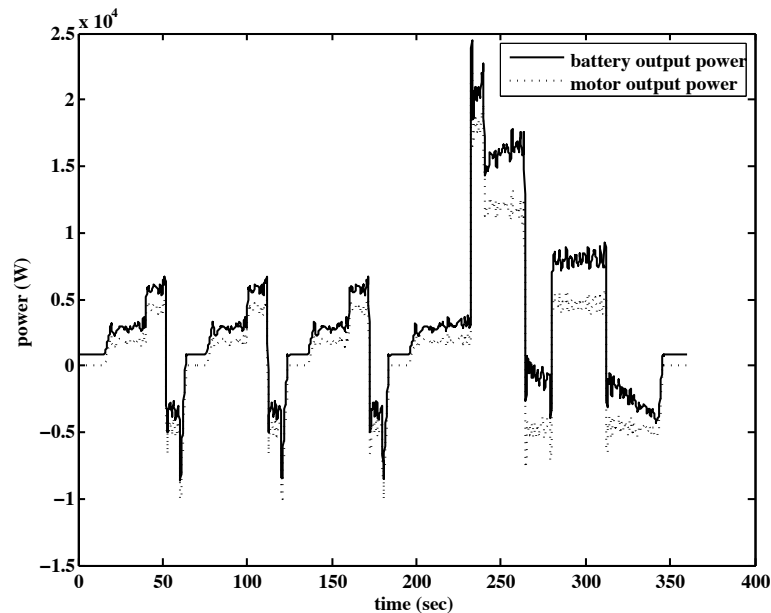


Figure 8.15 Battery and motor power output during SFUDS cycle

The vehicle range is determined by repeating the battery simulation with the SFUDS power curve as input until either the battery can no longer supply the required power to drive the SFUDS cycle, or the battery reaches its minimum allowed state of charge ($SOC_{min} = 0.30$). The latter requirement typically dominates the first. A constant accessory power demand of $P_{acc} = 750$ W is added to the motor power demand to account for additional electrical loads, such as power steering and air conditioning use. Figure 8.16 shows the battery output during a complete range test, and indicates the battery discharge ($P_u(t)$) and charge ($P_\ell(t)$) limits. The discharge power limit is never exceeded in this case, and the charge limit is only exceeded early on when the battery has a high state of charge and cannot

accept much power. It is assumed here that, if regenerative power input exceeds the charge limit, the excess energy is dissipated as waste heat through a resistor. The power demand, and charge and discharge limits, are also evaluated for the acceleration test, and used in calculating the power violation constraint g_5 . The power violation P_V is the maximum amount either $P_u(t)$ or $P_l(t)$ is violated by during range and acceleration tests.

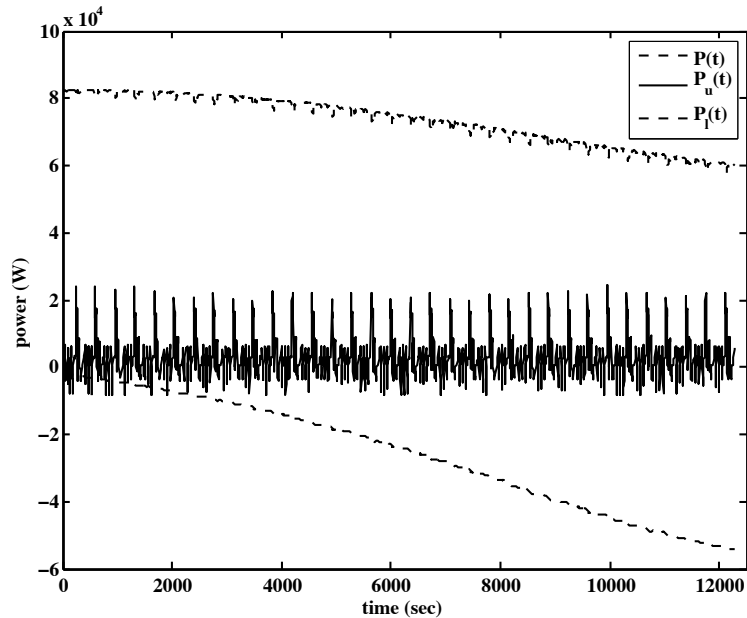


Figure 8.16 Battery power output and charge and discharge limits during range test

After the range test is complete, the resulting SOC is used as a starting point for a slow charging test to determine how much energy is required from the electrical grid to bring the battery back up to its original state of charge. The vehicle range (in miles) is then divided by the energy consumed in units of gallons of gasoline to obtain the grid-to-wheels fuel economy mpg_e , which is comparable to the familiar tank-to-wheels measure of fuel economy miles per gallon. The EV equivalent fuel economy is:

$$mpg_e = \frac{R}{E_c} \cdot 1.317 \cdot 10^8 \quad (8.48)$$

where E_c is the energy required to recharge the battery measured in Joules, and R is the EV urban range measured in miles.

8.3 Vehicle Dynamics Model

The vehicle dynamics analysis function uses three different models for the dynamic performance of the vehicle, including stability, steering responsiveness, driver comfort, and

roadholding metrics. A steady-state bicycle model is used to predict directional stability at a constant speed, and a dynamic bicycle model is used to evaluate steering responsiveness. A quarter-car model is used to compute metrics for driver comfort, suspension working space, and tire and suspension forces.

8.3.1 Directional Stability

The steering angle of the front tires required to guide a vehicle around a curve of radius R_c is:

$$\delta_f = \frac{L}{R_c} + \left(\frac{W_f}{C_{\alpha_f}} - \frac{W_r}{C_{\alpha_r}} \right) \frac{v^2}{gR_c} \quad (8.49)$$

where g is the acceleration of gravity, W_f and W_r are the static forces on the front and rear tires, respectively, and C_{α_f} and C_{α_r} are the cornering stiffnesses for the front and rear tires [147]. If the second term of Eq. 8.49 is positive, the steering angle is required to follow a curve of radius R_c increases with longitudinal velocity. This is a condition known as understeer. If the second term of Eq. 8.49 is negative, δ_f decreases with velocity. This oversteer condition can be unstable if v exceeds the critical velocity:

$$v_{\text{crit}} = \sqrt{\frac{gL}{-K_{us}}} \quad (8.50)$$

where K_{us} is the understeer coefficient:

$$K_{us} = \left(\frac{W_f}{C_{\alpha_f}} - \frac{W_r}{C_{\alpha_r}} \right) \quad (8.51)$$

Since v^2/gR_c is always positive, $K_{us} \geq 0$ indicates an understeer vehicle. The cornering stiffness values can vary with a number of factors. Here the dependence of cornering stiffness on tire normal force is modeled using a quadratic polynomial fit to empirical data for the EV fictitious tire [72]:

$$C_{\alpha_f} = C_{\alpha 1} + C_{\alpha 2}W_f + C_{\alpha 3}W_f^2 \quad (8.52)$$

The parameter values for Eq. (8.52) are given in Table 8.6. The formula for the rear cornering stiffness is similar, but depends on W_r instead. According to this steady-state stability model, directional stability is assured below a maximum intended speed v_{max} if:

$$D_s = L + \frac{v_{\text{max}}^2}{gK_{us}} \geq 0 \quad (8.53)$$

8.3.2 Steering Responsiveness

A two DOF second-order model was used to simulate dynamic vehicle motion in the yaw plane under steering input at a constant longitudinal velocity. The equations of motion are:

$$a_1 \dot{v}_y + a_2 \dot{\Omega}_z + a_3 v_y = a_4 \delta_f(t) \quad (8.54a)$$

$$b_1 \dot{\Omega}_z + b_2 \Omega_z + b_3 v_y = b_4 \delta_f(t) \quad (8.54b)$$

where:

$$\begin{aligned} a_1 &= m \\ a_2 &= m + \frac{2(\ell_1 C_{\alpha f} - \ell_2 C_{\alpha r})}{v} \\ a_3 &= \frac{2(C_{\alpha f} + C_{\alpha r})}{v} \\ a_4 &= 2C_{\alpha f} \\ b_1 &= I_z \\ b_2 &= \frac{2(\ell_1^2 C_{\alpha f} + \ell_2^2 C_{\alpha r})}{v} \\ b_3 &= \frac{2(\ell_1^2 C_{\alpha f} - \ell_2^2 C_{\alpha r})}{v} \\ b_4 &= 2\ell_1 C_{\alpha f} \end{aligned}$$

and v_y is the lateral vehicle velocity, Ω_z is the yaw rate, I_z is the yaw inertia, and $\delta_f(t)$ is a time-varying steering input. The corresponding state space model is:

$$\begin{bmatrix} \dot{v}_y \\ \dot{\Omega}_z \end{bmatrix} = \begin{bmatrix} -\frac{a_3}{a_1} & -\frac{a_2}{a_1} \\ -\frac{b_3}{b_1} & -\frac{b_2}{b_1} \end{bmatrix} \begin{bmatrix} v_y \\ \Omega_z \end{bmatrix} + \begin{bmatrix} \frac{a_4}{a_1} \\ \frac{b_4}{b_1} \end{bmatrix} \delta_f \quad (8.55)$$

The state variable of interest here is the yaw rate. When a vehicle is driving straight ahead after a step steering input δ_{f0} is applied, the lateral velocity and yaw rate increase from zero and then settle on steady-state nonzero values. A vehicle that approaches its steady-state yaw rate quickly is considered to be responsive. The yaw rate rise time (t_r), i.e., the time required for the yaw rate to increase from 10% to 90% of its steady state value, is the metric used to evaluate steering responsiveness. Constraint g_8 requires the yaw rate rise time to be less than $t_{r\max}$.

8.3.3 Quarter-Car Model

A quarter car model is used to simulate required suspension working space, tire and suspension forces, and vertical acceleration of the sprung mass [147]. The input is a road profile $z_0(x)$, which can be expressed as $z_0(t)$ if $v(t)$ is known, that excites motions in the sprung and unsprung masses. The quarter-car model is diagrammed in Fig. 8.17, and a corresponding state space model is given in Eq. (8.56).

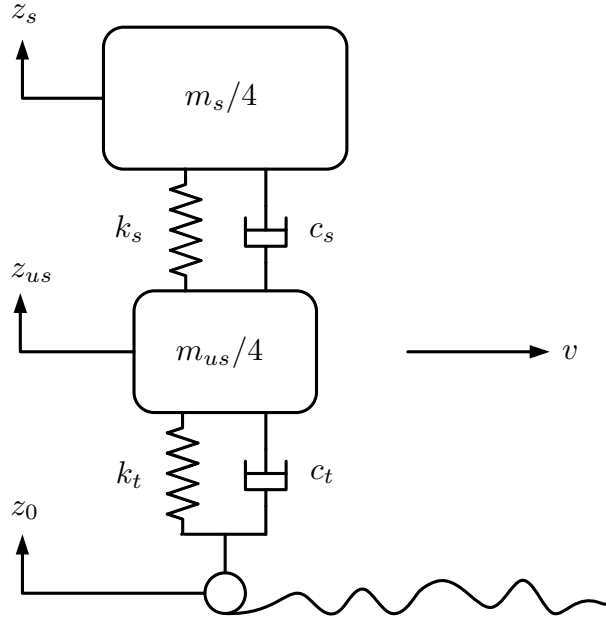


Figure 8.17 Quarter-car vehicle suspension model

$$\frac{d}{dt} \begin{bmatrix} z_{us} - z_0 \\ \dot{z}_{us} \\ z_s - z_{us} \\ \dot{z}_s \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\frac{4k_t}{m_{us}} & -\frac{4(c_s+c_t)}{m_{us}} & \frac{4k_s}{m_{us}} & \frac{4c_s}{m_{us}} \\ 0 & -1 & 0 & 1 \\ 0 & \frac{4c_s}{m_s} & -\frac{4k_s}{m_s} & -\frac{4c_s}{m_s} \end{bmatrix} \begin{bmatrix} z_{us} - z_0 \\ \dot{z}_{us} \\ z_s - z_{us} \\ \dot{z}_s \end{bmatrix} + \begin{bmatrix} -1 \\ \frac{4c_t}{m_{us}} \\ 0 \\ 0 \end{bmatrix} \dot{z}_0 \quad (8.56)$$

The first state variable is the tire deflection; the second is the unsprung mass velocity; the third is the suspension stroke, and the fourth is the sprung mass velocity. The tire force, including static vehicle weight, is:

$$F_{z_0} = k_t(z_{us} - z_0) + c_t \frac{d}{dt}(z_{us} - z_0) + \frac{mg}{4} \quad (8.57)$$

The minimum tire force during a simulation (F_t) is used in the constraint g_{10} to ensure positive force is maintained between the tire and road to ensure adequate roadholding on

rough terrain and to verify that the tire never loses contact with the road under normal driving conditions, a requirement for quarter-car model validity. The force that the suspension exerts on the sprung mass is:

$$F_{z_s} = -\frac{m_s \ddot{z}_s}{4} \quad (8.58)$$

Three different tests are performed using the quarter-car model: a ramp input test, a moderately rough road simulation, and a very rough road simulation. In all tests the tire damping rate (c_t) is assumed to be zero. In the ramp input test the vehicle approaches a ramp input at a grade of γ at a velocity of v_{r0} , and the maximum force on the sprung mass ($F_s = \max(F_{z_s}(t))$) is recorded for use in the structural analysis function. The suspension working space (δ_w) is the largest value of the third state variable during the ramp test, and is used in the constraint g_9 . The other two tests use stochastic inputs: $z_{0a}(x)$ is the spatial profile of a moderately rough road, and $z_{0b}(x)$ is the spatial profile of a very rough road. A constant velocity was used with each of these inputs: v_{a0} and v_{b0} . These profiles were generated using a gaussian random number generator and a series of filtering steps, and then analyzed using standard techniques for the international roughness index (IRI) [120]. The first step is to generate $N_p = \lceil L_p/\delta_p \rceil$ random data points with a variance of σ_p , and then apply a digital filter to remove high-frequency data beyond a spatial cutoff frequency of f_0 [12]. The spatial length of the profile is L_p , and the step size in the profile data is δ_p . The filter implemented is:

$$y_n = bx_n + ay_{n-1} \quad (8.59)$$

where x_n and y_n are the n -th unfiltered and filtered data points, respectively, and the filter coefficients are defined as:

$$\begin{aligned} a &= e^{-2\pi f_0 \delta_p} \\ b &= 1 - e^{-2\pi f_0 \delta_p} \end{aligned}$$

A second type of digital filter is then applied to the data as specified by IRI requirements. This moving average filter is defined by the formula:

$$y_n = \frac{1}{N_w} \sum_{i=n-N_w+1}^n x_i \quad (8.60)$$

where $N_w = \lceil L_0/\delta_p \rceil$ is the number of data points in the filter baselength L_0 . The profile is then refined using a finer step size and smoothed using spline interpolation so that input is more suitable for simulation purposes. A portion of one road profile is illustrated in Fig. 8.18, where unfiltered data is displayed alongside profiles generated after the first and

second filtering steps. The profile after the interpolation step is visually indiscernible from the fully filtered profile.

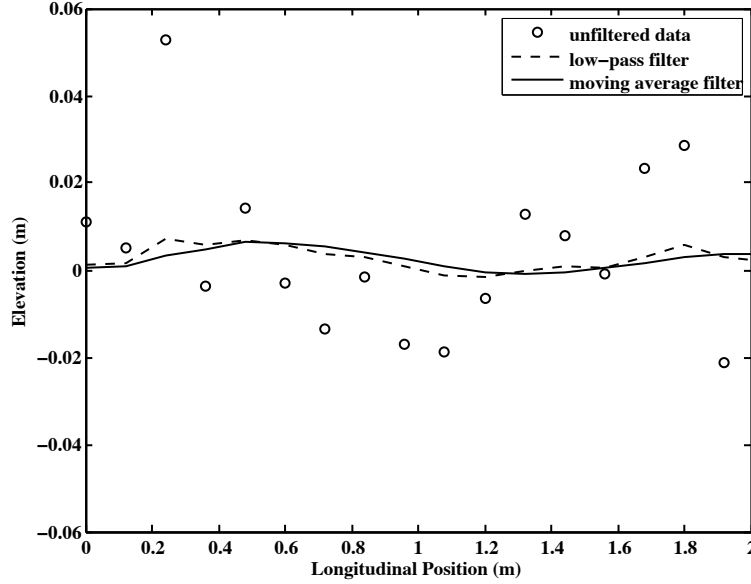


Figure 8.18 Sample road profile created using gaussian random number generator and digital filters

The roughness of a road profile may be quantified using the IRI, which is a measure of how much a vehicle suspension moves while traveling over a road surface. A standard quarter-car model is used to calculate the IRI for a road profile, where the vehicle parameters must satisfy:

$$\frac{k_s}{m_s} = 63.3, \quad \frac{k_t}{m_s} = 653, \quad \frac{c_s}{m_s} = 6, \quad \frac{m_{us}}{m_s} = 0.15$$

The IRI is the amount of suspension travel for the quarter-car model per distance travelled, and is usually measured in meters of suspension travel per kilometer of longitudinal distance. The suspension travel may be expressed as:

$$\delta_S = \int_0^{t_f} |z_s - z_{us}| dt \quad (8.61)$$

The profile $z_{0a}(x)$ has an IRI of 4.20, and corresponds to the roughness of an older, but undamaged, paved road. This profile was used in calculating the driver discomfort metric. The second profile, $z_{0b}(x)$, has an IRI of 7.37, corresponding to a maintained unpaved road or a damaged paved road. This profile was used to compute F_t .

Many criteria have been proposed for quantifying driver comfort. Smith, McGehee, and Healey reviewed many of these techniques, and concluded that a simple root mean square acceleration measurement is especially effective [123]. Gobbi and Mastinu described how

to use such a metric in ground vehicle optimization [64]. The metric used here is root mean square of the power spectral density of the sprung mass acceleration (\ddot{z}_s), denoted by a_z . BS 6841 suggests that an a_z value of up to 0.642 g 's ($1g = 9.81 \text{ m/s}^2$) results in a ride quality that is only 'a little uncomfortable' [26]. Constraint g_{11} limits a_z to 0.80 g while traveling over the profile $z_{0a}(x)$.

Table 8.6 Vehicle dynamics model parameters

$C_{\alpha 1}$	$-3.68 \cdot 10^3$	constant cornering stiffness parameter
$C_{\alpha 2}$	21.4	linear cornering stiffness parameter
$C_{\alpha 3}$	$2.70 \cdot 10^{-3}$	quadratic cornering stiffness parameter
δ_{f0}	0.02 rad	steering step input
v_{r0}	15.0 m/s	ramp test velocity
v_{a0}	15.0 m/s	velocity for stochastic profile a
v_{b0}	22.2 m/s	velocity for stochastic profile b
γ	13%	ramp test grade
f_0	0.15 cycle/m	filter cutoff frequency
L_0	0.25 m	filter baselength

8.4 Structural Model

The vehicle structure was modeled as a space frame and analyzed using finite element analysis (FEA) [93] to determine the stiffness and stress values for constraints g_{13} – g_{16} . The FEA model was created using ANSYSTM, and also was used to calculate mass and inertia properties of the vehicle frame required as input to the mass distribution and packaging function. The vehicle frame model is shown in Fig. 8.19 with the beam element divisions visible (911 beam elements in total). The size of frame elements is exaggerated in the figure, and frame members have circular cross sections. The frame material is AISI 4130 steel with a modulus of 205 GPa, Poisson's ratio of 0.30, and mass density of 7500 kg/m³.

The structural design was simplified by using only two design variables: t_f , the wall thickness of all frame members, and d_f , the outer diameter of all frame members. This design variable set requires a compatibility constraint (g_{12} in Problem (8.2)). The influence of the battery mass and stiffness is included in this model. Four structural tests are performed to evaluate constraints g_{13} – g_{16} :

1. Torsional stiffness ($K_{t\min}$): A moment about the x axis is applied via the rear suspension hardpoints while the front hardpoints are held fixed. The deflection is measured, and then the rotational angle and torsional stiffness is calculated. This stiffness should be at least 12,000 Nm/deg [95].
2. Bending stiffness ($K_{b\min}$): the front and rear axles are fixed while a vertical load is

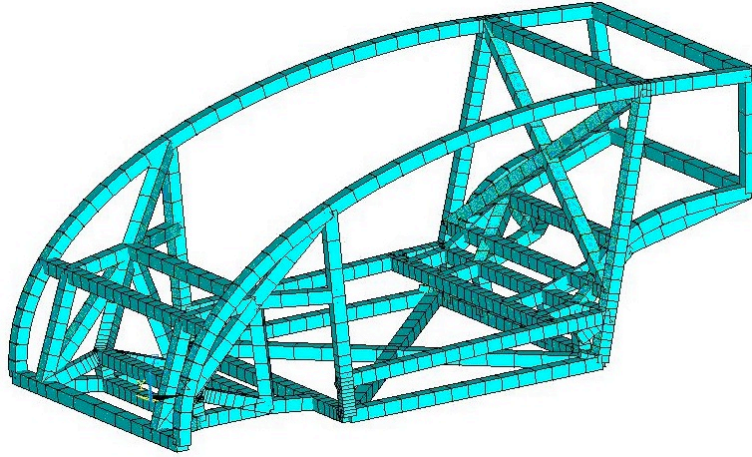


Figure 8.19 FEA model of the EV frame

applied at the center of the vehicle. The deflection is measured and used along with the load value to determine the vehicle bending stiffness, which should be at least 6 MN/m [95].

3. Torsional stress (σ_{ft}): The maximum suspension load F_s is applied in opposite vertical directions at the rear suspension hardpoints while holding the front suspension hardpoints fixed. The maximum Von Mises stress is recorded and must not exceed the maximum allowable stress of $\sigma_Y = 350$ MPa.
4. Bending stress (σ_{bt}): The maximum suspension load F_s is applied in the same vertical direction at the rear suspension hardpoints while holding the front suspension hardpoints fixed. The maximum Von Mises stress is recorded and must not exceed the maximum allowable stress of $\sigma_Y = 350$ MPa.

The structural model computational expense is higher than for the other models, so an artificial neural network [70] was constructed and used as a surrogate model.

8.5 Mass Distribution and Packaging

The mass distribution and packaging function computes vehicle mass and inertia properties and packaging criteria using geometry, mass, and inertia data from the three vehicle systems. Estimated mass and inertia properties for the vehicle without the frame or battery, known as the baseline vehicle properties, are combined with mass and inertia properties for the frame and battery to arrive at estimates for overall vehicle properties. This allows the model to account for the influence of frame and battery design changes on vehicle dynamics. The parameters used in this model are listed in Table 8.7. The first step is to determine the sprung

mass and its center of mass location:

$$m_s = m_0 + m_b + m_f \quad (8.62a)$$

$$h = \frac{1}{m_s} (m_0 h_0 + m_b h_b + m_f h_f) \quad (8.62b)$$

$$\ell_1 = \frac{1}{m_s} (m_0 \ell_{1_0} + m_b \ell_b + m_f \ell_f) \quad (8.62c)$$

where m_0 , m_b , and m_f are the baseline vehicle mass, battery mass, and frame mass, respectively; h_0 , h_b , and h_f are the center of mass heights for the baseline vehicle, battery, and frame, respectively; ℓ_{1_0} , ℓ_b , and ℓ_f are the longitudinal center of mass locations for the baseline vehicle, battery, and frame, respectively, measured from the front axle location. The longitudinal position of the battery mass center, measured as the distance between the front axle and the battery mass center, is given by:

$$\ell_b = \ell_e + x_b + b_\ell/2 \quad (8.63)$$

where ℓ_e is the distance between the front axle and the front edge of the available battery space, and x_b is the distance between this front edge and the front of the battery. The quantity x_b is a design variable that defines the longitudinal position of the battery. The battery inertia values about its own center of mass are:

$$I'_{yb} = \frac{1}{12} m_b (b_h^2 + b_\ell^2) \quad (8.64a)$$

$$I'_{zb} = \frac{1}{12} m_b (b_w^2 + b_\ell^2) \quad (8.64b)$$

where b_h , b_w , and b_ℓ are the battery height, width, and length, respectively. The battery inertia values about the vehicle center of mass are computed using the parallel axis theorem [18]:

$$I_{yb} = I'_{yb} + m_b \|[\ell_b, h_b] - [\ell_1, h]\|_2^2 \quad (8.65a)$$

$$I_{zb} = I'_{zb} + m_b (\ell_b - \ell_1)^2 \quad (8.65b)$$

The frame inertia values about the vehicle center of mass are:

$$I_{yf} = I'_{yf} + m_f \|[\ell_f, h_f] - [\ell_1, h]\|_2^2 \quad (8.66a)$$

$$I_{zf} = I'_{zf} + m_f (\ell_f - \ell_1)^2 \quad (8.66b)$$

where I'_{yf} and I'_{zf} are the frame pitch and yaw inertia values about the frame's mass center,

provided by the structural analysis function. The baseline sprung mass inertia values about the vehicle mass center are:

$$I_{y_0} = I'_{y_0} + m_f \|[\ell_{1_0}, h_0] - [\ell_1, h]\|_2^2 \quad (8.67a)$$

$$I_{z_0} = I'_{z_0} + m_f (\ell_{1_0} - \ell_1)^2 \quad (8.67b)$$

where I'_{y_0} and I'_{z_0} are the baseline vehicle inertia values for the sprung mass about its own center of mass. The sprung mass inertia values, including frame and battery, about its own mass center are:

$$I_y = I_{yb} + I_{yf} + I_{y_0} \quad (8.68a)$$

$$I_z = I_{zb} + I_{zf} + I_{z_0} \quad (8.68b)$$

The packaging constraints are computed as defined by g_{17} and g_{18} in Problem (8.2).

Table 8.7 Mass distribution and packaging model parameters

m_0	423 kg	baseline vehicle mass
h_0	0.610 m	baseline mass center height
h_b	0.355 m	battery mass center height
b_h	0.110 m	battery height
ℓ_{1_0}	0.935 m	baseline mass center longitudinal position
ℓ_e	0.490	battery box position
I'_{y_0}	299 kg-m ²	baseline vehicle pitch inertia
I'_{z_0}	872 kg-m ²	baseline vehicle yaw inertia

8.6 Optimal P/C Decision Results

The optimal partitioning and coordination decision method for ALC from Chapter 7 was applied to the reduced adjacency matrix for the EV design problem. The reduced adjacency matrix is:

$$\mathbf{A}_5 = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The results are illustrated in Fig. 8.20. Only 175 unique P/C instances exist for this problem, owing to the sparsity of \mathbf{A}_5 . Four Pareto-optimal points were identified, all of which corresponded to more than one ALC implementation instance except for point 4.

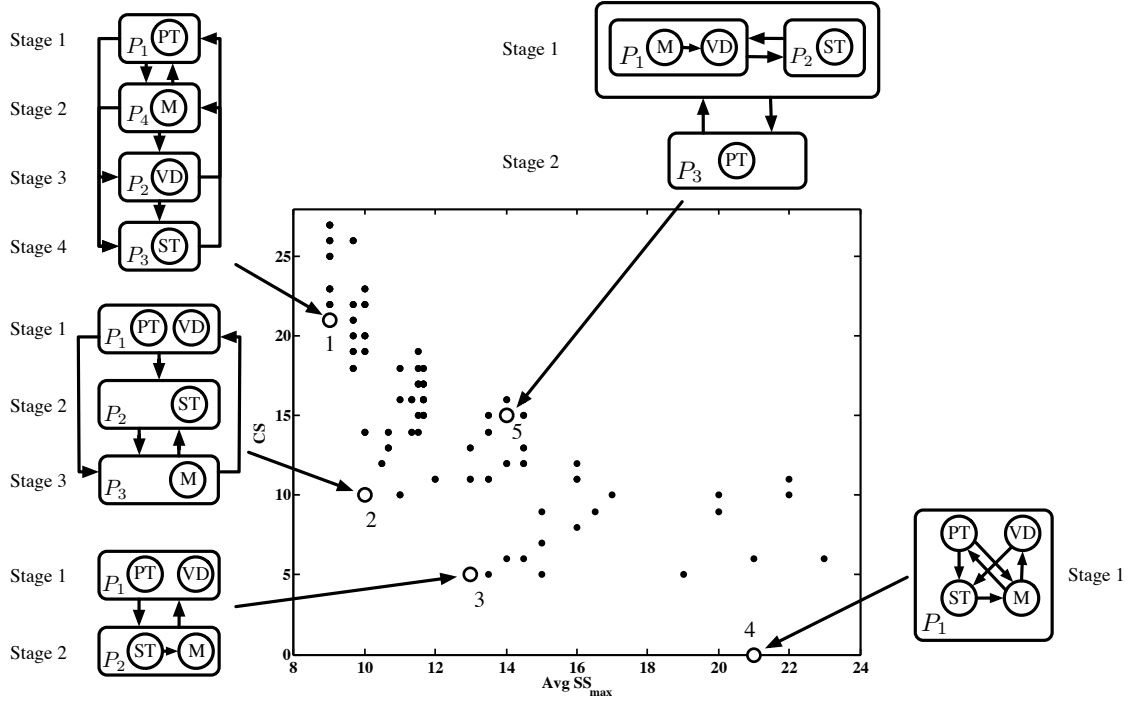


Figure 8.20 Optimal partitioning and coordination decision results for the EV problem

Point 1:

Four P/C decision instances correspond to point 1 in Fig. 8.20, and all share the same partition and problem size metrics:

$$\begin{aligned}
 CS &= 21 \\
 \bar{SS}_{\max} &= 9 \\
 \mathbf{p} &= [1, 2, 3, 4]
 \end{aligned}$$

All four instances have the same set of shared variables ($\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_{13}\}$), and the same shared variable consistency constraint allocation (G_c for \mathbf{x}_1 : $\{\langle 1, 2 \rangle\}$, \mathbf{x}_2 : $\{\langle 1, 2 \rangle\}$, \mathbf{x}_{13} : $\{\langle 3, 4 \rangle\}$). This consistency constraint assignment is the sole feasible option since each shared variable is shared only between two subproblems. The four separate instances are distinguished by subproblem stage assignment:

Instance 1: $\mathbf{s} = [1, 4, 2, 3]$

Instance 2: $\mathbf{s} = [2, 1, 3, 4]$

Instance 3: $\mathbf{s} = [3, 2, 4, 1]$

Instance 4: $\mathbf{s} = [4, 3, 1, 2]$

The ALC implementation for the first instance is illustrated in Fig. 8.20.

Point 2:

Three P/C decision instances correspond to point 2, and all share the same partition and problem size metrics:

$$\begin{aligned} CS &= 10 \\ \bar{S}S_{\max} &= 10 \\ \mathbf{p} &= [1, 1, 2, 3] \end{aligned}$$

Grouping the powertrain and vehicle dynamics analysis functions together reduces the number of external shared variables. All three instances have one shared variable only (\mathbf{x}_{13} }), and the only possible consistency constraint assignment is $\{\langle 2, 3 \rangle\}$ for \mathbf{x}_{13} . The subproblem stage assignment for each instance is as follows:

Instance 1: $\mathbf{s} = [1, 2, 3]$

Instance 2: $\mathbf{s} = [2, 3, 1]$

Instance 3: $\mathbf{s} = [3, 1, 2]$

The ALC implementation for the first instance is illustrated in Fig. 8.20.

Point 3:

Two P/C decision instances correspond to point 2, and all share the same partition and problem size metrics:

$$\begin{aligned} CS &= 5 \\ \bar{S}S_{\max} &= 13 \\ \mathbf{p} &= [1, 1, 2, 2] \end{aligned}$$

Again, the powertrain and vehicle dynamics analysis functions are placed in the same subproblem. The third and fourth analysis functions share one design variable, which can be eliminated from the set of external shared design variables by grouping these analysis functions into the same subproblem. Point 3 has no external shared design variables, and therefore no shared variable consistency constraint assignments. The subproblem stage assignment for each instance is as follows:

Instance 1: $\mathbf{s} = [1, 2]$

Instance 2: $\mathbf{s} = [2, 1]$

The ALC implementation for the first instance is illustrated in Fig. 8.20.

Point 4:

The fourth Pareto-optimal point corresponds to the IDF formulation with a single subproblem. IDF formulations do not have external shared design variables. The problem size metrics are:

$$\begin{aligned} CS &= 0 \\ \bar{S}_{\max} &= 21 \end{aligned}$$

Point 5:

The last point discussed here is not Pareto-optimal, but is included for comparison with the other points. Point 5 in Fig. 8.20 corresponds to two instances that share the same partition and problem size metrics:

$$\begin{aligned} CS &= 15 \\ \bar{S}_{\max} &= 14 \\ \mathbf{p} &= [1, 2, 3, 2] \end{aligned}$$

Analysis functions two and four are grouped together, but share no design variables, and have only one coupling variable relationship. This partition does not help reduce CS , and results in the maximal set of external shared variables: $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_{13}\}$. Since these variables link only two subproblems each, the only consistency constraint allocation option exists (G_c for $\mathbf{x}_1: \{\langle 1, 2 \rangle\}$, $\mathbf{x}_2: \{\langle 1, 2 \rangle\}$, $\mathbf{x}_{13}: \{\langle 3, 4 \rangle\}$). Point 5 differs from the others in that each instance specifies parallel subproblem solution:

Instance 1: $\mathbf{s} = [1, 2, 2]$

Instance 2: $\mathbf{s} = [2, 1, 1]$

The second instance is illustrated in Fig. 8.20. The EV problem structure is such that parallel subproblem solution does not offer an advantage when CS and \bar{S}_{\max} are the metrics used in solving the optimal P/C decision problem. All four Pareto-optimal points involve serial subproblem solution. Different problem structure, or an alternative CS metric that penalizes increased stage depth, may change this outcome. The work presented here establishes a new approach to constructing distributed optimization problems; further work is required to study metrics used to approximate solution difficulty.

In each Pareto-optimal partition with less than four subproblems, the powertrain and vehicle dynamics analysis functions are grouped together. These functions also share the

largest number of design variables between them. Placing \mathbf{a}_1 and \mathbf{a}_2 in the same subproblem has a large impact on reducing the number of external shared design variables. Recall from Chapter 7 that a variable shared between two subproblems requires two separate variable copies, while coupling variables require only one. These copies increase both coordination and subproblem size. Therefore, the grouping of \mathbf{a}_1 and \mathbf{a}_2 is expected; choosing partitions that lead to fewer external shared design variables is a good approach for reducing both coordination and subproblem size. This highlights the importance of distinguishing between shared and coupling variables, rather than treating all linking variables as shared variables.

Moving from point 4 to point 3 reduces $\bar{S}S_{\max}$ from 21 to 13, and only requires a coordination problem size of 5. This indicates that the EV problem is a good candidate for decomposition-based design optimization. Also note that several points exist in Fig. 8.20 with $\bar{S}S_{\max} > 21$ and $CS > 0$; these points correspond to exceptionally poor partitions that increase both coordination and subproblem size. As we move along the Pareto set starting with point 1, the number of subproblems increases, which is expected as more emphasis is placed on reducing CS . Note that the number of instances corresponding to each point also increases as we move toward finer partitions; this is also expected since stage assignment options increase with the number of subproblems.

Each of the 175 P/C instances for the EV problem specifies uniquely an ALC formulation. The formulation for the first instance of point 3 is presented here as an illustration. The external coupling variables in this case are:

$$\begin{aligned}\bar{\mathbf{y}}_{12} &= [\mathbf{y}_{14}, \mathbf{y}_{24}] \\ \bar{\mathbf{y}}_{21} &= [\mathbf{y}_{31}, y_{32}, \mathbf{y}_{41}]\end{aligned}$$

where:

$$\begin{aligned}\mathbf{y}_{14} &= [m_s, h, \ell_1, I_y] \\ \mathbf{y}_{24} &= [m_s, \ell_1, I_z] \\ \mathbf{y}_{31} &= [b_m, b_w, b_\ell] \\ y_{32} &= F_s \\ \mathbf{y}_{41} &= [b_m, b_w, b_\ell]\end{aligned}$$

The coupling variables between analysis functions three and four are internal coupling variables for subproblem 2:

$$\mathbf{y}_{43} = [m_f, h_f, \ell_f, I_{yf}, I_{zf}]$$

The ALC formulation for subproblems 1 and 2 are given in Eqs. (8.69) and (8.69), respectively. $\bar{\mathbf{S}}_{12}$ and $\bar{\mathbf{S}}_{21}$ are selection matrices that choose analysis function components

that correspond to coupling variables in consistency constraints. As discussed previously, this instance has no external shared design variables, so only coupling variables appear in consistency constraints. Subproblem 2 has internal coupling variables, which appear as decision variables and are assured to be consistent by the auxiliary equality constraint \mathbf{h}_{aux} .

$$\begin{aligned} \min_{\bar{\mathbf{x}}_1, \bar{\mathbf{y}}_{12}} \quad & 1/\text{mpg}_e + \phi(\bar{\mathbf{y}}_{12} - [\mathbf{a}_3, \mathbf{a}_4]\bar{\mathbf{S}}_{12}) \\ \text{subject to} \quad & [g_1, g_2, \dots, g_{11}] \leq \mathbf{0} \end{aligned} \quad (8.69)$$

$$\begin{aligned} \min_{\bar{\mathbf{x}}_2, \bar{\mathbf{y}}_{21}, \mathbf{y}_{43}} \quad & \phi(\bar{\mathbf{y}}_{21} - [\mathbf{a}_1, \mathbf{a}_2]\bar{\mathbf{S}}_{21}) \\ \text{subject to} \quad & [g_{12}, g_{13}, \dots, g_{18}] \leq \mathbf{0} \\ & \mathbf{h}_{\text{aux}} = \mathbf{y}_{43} - \mathbf{a}_3\mathbf{S}_{43} = \mathbf{0} \end{aligned} \quad (8.70)$$

8.7 Concluding Comments

The chapter introduced an electric vehicle design problem that included interactions between several important vehicle systems, including powertrain, chassis, and structure. The vehicle model was presented in detail, and the design problem was formulated to minimize energy consumption while meeting performance constraints. The techniques introduced in Chapter 7 were used to solve the optimal P/C decision problem, defined in Eq. (7.9), for the electric vehicle design problem. Four Pareto-optimal solutions were identified, and the ALC formulation for one of these solutions was presented. Design optimization results for the AiO and ALC solutions are being addressed in ongoing work.

Chapter 9

Conclusion

9.1 Summary

Many engineering systems are too complex to approach as single large design problems. A usual approach is to partition the system design problem into smaller subproblems that are more easily solved. These subproblems are coupled through analysis interactions and shared design variables. A coordination strategy is required to account for these interactions, and to help guide iterative solution of subproblems toward a state of consistency, and toward a design that is optimal for the overall system, not just for the individual components. Decomposition-based design optimization techniques utilize optimization algorithms to solve system subproblems, and coordination algorithms to guide the solution process. These techniques are especially useful when mathematical models exist that can predict system behavior. This dissertation focused on the case where computer simulations are used as the analysis functions in the system model.

The difficulty of solving the subproblems and the difficulty of the coordination problem both contribute to overall solution expense. Both of these factors are influenced by the system partition and coordination strategy. Partitioning and coordination decisions must be made before a system design problem can be solved using decomposition-based design optimization. This dissertation was centered on techniques for making partitioning and coordination decisions using information about problem structure, such as dependence relationships between analysis functions and on design variables. Previous techniques had addressed either the partitioning problem or some form of the coordination decision problem, but few had studied partitioning and coordination decisions together. It was shown here that partitioning and coordination decisions are coupled; this was accomplished by demonstrating that making partitioning and coordination decisions independently or in sequence leads to suboptimal results, while a simultaneous decision approach consistently identified better decisions.

A simultaneous decision approach requires precedence information as well as knowledge

of analysis function dependence on design variables. Early methods relied solely on precedence information to make partitioning decisions; later partitioning methods did not use directionality in making decisions. The simultaneous partitioning and coordination decision methods introduced here use both types of information. The reduced adjacency matrix was developed specifically to represent this information compactly, and is in a form convenient for computational purposes. The reduced adjacency matrix enables also calculation of subproblem optimization dimension; previous methods approximate this quantity using just the number of analysis functions or variables.

The optimal P/C methods of Chapters 5–7 involved the solution of nonlinear combinatorial optimization problems, which are NP-complete. Exhaustive enumeration may be used to solve these problems for small systems, and an evolutionary algorithm was developed for larger systems. While an efficient algorithm does not yet exist for solving these problems, in practice they can be solved fast enough to be of benefit. Some alternate methods, such as Michelena’s spectral partitioning method [105], make several simplifying assumptions that allow application of very fast solution algorithms. The optimal P/C methods presented here are more difficult to solve, but provide a more complete decision model with fewer approximations. In many cases the design problem solution time vastly exceeds P/C solution time, so the more accurate P/C decision methods may be preferable.

Two optimal P/C methods were presented. The first, given in Chapter 5, applies to distributed optimization methods with properties similar to ATC or ALC. The second method, presented in Chapter 7, was developed for a specific class of parallel ALC implementations. This allowed the use of very detailed problem size metrics. More importantly, limiting the P/C decision method to parallel ALC for quasi-separable problems enabled the addition of consistency constraint allocation to the P/C decision method. An automated method for allocating consistency constraints is especially useful for ALC due to its linking structure flexibility. ALC consistency constraints can be adapted to fit the structure of arbitrary non-hierarchical problems, but the immense number of linking structure options makes manual decision making impractical. The developed automated technique was required to take advantage of ALC’s flexibility in practice.

Several original engineering design problems were developed to demonstrate concepts throughout this dissertation. The majority are described in sufficient detail for replication. An electric water pump design problem and a structural design problem were used in demonstrating P/C decision making without consistency constraint allocation, and an electric vehicle (EV) design problem was developed to demonstrate P/C decision making with consistency constraint allocation. The EV problem emphasized analysis interactions and shared design variables between vehicle systems.

9.2 Extension of Simultaneous P/C Decision Making

The P/C decision methods presented here were developed with an eye toward system design problems that use simulation-based analysis functions. These problems have known input-output relationships, and normally have quasi-separable problem structures. While the specific techniques presented here apply to problems that meet these assumptions, the more general principle of simultaneous P/C decision making can be extended to other applications. For example, new decision models can be derived for distributed optimization formulations other than ATC or ALC, or for problems that are not quasi-separable. The steps for creating a new decision model are as follows:

1. Identify primary sources of computational expense
2. Isolate partitioning and coordination decisions that impact these sources
3. Define a set of decision variables for the P/C problem
4. Define P/C problem objective function(s)
5. Develop a mapping between decision variables and objective functions

Many system optimization methods employ some type of nested approach. ATC and ALC have subproblems nested within a fixed point iteration algorithm. Collaborative optimization and Dantzig-Wolfe decomposition have subproblems nested within a master optimization problem. Nested processes generally have two primary sources of computational expense: the inner loop and the outer loop. The methods presented here use CS to estimate outer loop expense, and SS_{\max} or $\bar{S}S_{\max}$ to model inner loop expense. In a nested process there is a natural tradeoff between these two sources of expense, and it is important in an optimal P/C method to address this tradeoff. A multi-objective optimization approach is a helpful approach for analyzing such tradeoffs.

This methodology is not limited to decomposition-based design optimization, but can apply to other processes with a nested structure or some type of coordination algorithm. For example, more general parallel computing could benefit from such an approach. A computing job must be partitioned into individual computing tasks, and then a schedule for these tasks must be defined that pays heed to precedence relationships and computational time. The details may be very different from decomposition-based design optimization, but as long as the steps above can be followed, a simultaneous partitioning and coordination decision method may be constructed for parallel computing applications.

9.3 Contributions

1. Partitioning and coordination decisions in decomposition-based design optimization were shown to be coupled, and that treating them independently leads to sub-optimal solution approaches.
2. A method for making optimal partitioning and coordination decisions was developed, and it was shown how this method can also be used to analyze the suitability of a decomposition-based approach for solving a particular system design optimization problem. The method employs the reduced adjacency matrix, a new system structure representation that combines functional dependence and precedence information required for making simultaneous P/C decisions.
3. A new evolutionary algorithm was developed that solves the optimal P/C problem for larger systems. The results of this algorithm compared well against exact results obtained using exhaustive enumeration.
4. The Augmented Lagrangian Coordination (ALC) formulation provides tremendous flexibility in problem linking structure. This allows a solution process to be tailored to the needs of a specific problem. Linking structure theory for ALC was developed, and it was shown that the number of linking structure options is very large. Manual linking structure decisions are manageable when based on bi-level or hierarchical structures, but this approach fails to exploit the full benefit of linking structure flexibility in ALC. This dissertation put forth a method for automating linking structure decisions for ALC in an optimal P/C decision method, enabling system designers to take full advantage of ALC.
5. Several original engineering design examples were presented. Sufficient detail was provided such that most examples can be replicated. These examples therefore may be used as a basis for future investigations in decomposition-based design optimization. These examples are an important contribution because suitable system design examples are notably lacking in the literature.

9.4 Future Work

Evolutionary Algorithm

An evolutionary algorithm was developed for the general partitioning and coordination decision method introduced in Chapter 5. This entailed a custom genotype representation for system partition and subproblem sequence. The more detailed method introduced in Chapter 7 for the parallel ALC P/C decision problem involved exhaustive enumeration, and was limited to systems with a maximum of four analysis functions; an evolutionary algorithm would increase this upper bound and improve the applicability of this method. A new genotype representation for stage assignment and consistency constraint allocation is required.

Adaptive P/C Decision Method

A key assumption throughout this dissertation has been that partitioning and coordination decisions must be made before solution of the system design problem commences; decisions must be made with incomplete or approximate information under this assumption. Techniques have been developed for updating parallel processing implementations dynamically, i.e., as more information about the system is gathered, or as solution requirements evolve, the parallel processing approach can be updated accordingly without restarting the solution process. It is proposed that this principle can be applied to partitioning and coordination decisions for decomposition-based design optimization. Early on in the solution process relatively little is known about a system. As the solution process progresses, information can be gathered and analyzed. These data may be used to determine whether dynamic changes in system partition or coordination strategy would be beneficial.

Alternate Coordination Algorithms

Fixed point iteration, or nonlinear Gauss-Seidel, is a nearly ubiquitous choice for coordination algorithm for methods similar to ATC or ALC. It is a relatively stable, zeroth-order algorithm for solving systems of nonlinear equations. Many other options exist for solving the coordination problem, such as successive over-relaxation [71], Newton's method [43], or Aitken's method [13]. The effect of these alternative algorithms on system optimization convergence should be studied.

Integrated Vehicle Design

The EV design problem introduced in Chapter 8 is a first step toward a more integrated approach to vehicle design. Several interactions were included in the design model, but many more interactions important to vehicle performance could be included. For example, the chassis subproblem could include more sophisticated simulations and vehicle maneuvers, or consider detailed suspension geometry in the design. Commercial software for simulating vehicle dynamics that is currently available is not amenable to the type of integration demonstrated in Chapter 8. Substantial software development is required before an integrated vehicle model is realizable. Another improvement includes a more sophisticated packaging model would allow for changes in important vehicle dimensions, such as wheelbase and track width, and propagate the effect of these changes throughout the vehicle.

Bibliography

- [1] NOMADm: Nonlinear Optimization for Mixed vARiables and Derivatives for Matlab. <http://www.afit.edu/en/ENC/Faculty/MAbramson/nomad.html>. (Accessed August 8, 2005).
- [2] M.A. Abramson, C. Audet, and J.E. Dennis Jr. Nonlinear programming with mesh adaptive direct searches. *SIAG/Optimization Views-and-News*, 17(1):2–11, 2006.
- [3] N.M. Alexandrov and R.M. Lewis. Algorithmic perspective on problem formulations in MDO. In *the proceedings of the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA, Sept. 6–8, 2000.
- [4] N.M. Alexandrov and R.M. Lewis. Analytical and computational aspects of collaborative optimization for multidisciplinary design. *AIAA Journal*, 40(2):301–309, 2002.
- [5] J.T. Allison. Complex system optimization: A review of analytical target cascading, collaborative optimization, and other formulations. Master’s thesis, Department of Mechanical Engineering, University of Michigan, 2004.
- [6] J.T. Allison, M. Kokkolaras, and P.Y. Papalambros. On selection of single-level formulations for complex system design optimization. *Journal of Mechanical Design, Transactions of the ASME*, 129(9), 2007.
- [7] J.T. Allison, M. Kokkolaras, M. Zawislak, and P.Y. Papalambros. On the use of analytical target cascading and collaborative optimization for complex system design. In *the proceedings of the 6th World Conference on Structural and Multidisciplinary Optimization*, Rio de Janeiro, Brazil, May 30–June 3, 2005.
- [8] J.T. Allison, B. Roth, M. Kokkolaras, I. Kroo, and P.Y. Papalambros. Aircraft family design using decomposition-based methods. In *the proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Portsmouth, Virginia, Sept. 6–8, 2006.
- [9] S. Altus, I. Kroo, and P. Gage. Genetic algorithm for scheduling and decomposition of multidisciplinary design problems. *Journal of Mechanical Design, Transactions of the ASME*, 118(4):486–489, 1996.
- [10] S.F. Alyaquot, P.Y. Papalambros, and A.G. Ulsoy. Quantification and use of system coupling in decomposed design optimization problems. In *the proceedings of*

International Mechanical Engineering Congress and Exposition, November 5-11, 2005.

- [11] B. Amin. *Induction Motors: Analysis and Torque Control*. Springer-Verlag, Berlin, 2001.
- [12] A. Antoniou. *Digital Filters: Analysis, Design, and Applications*. McGraw-Hill, 1993.
- [13] I.K. Argyros. Steffensen-aitken methods and implicit functions. *Pan American Mathematical Journal*, 11(1):91–96, 2001.
- [14] C. Audet and J.E. Dennis Jr. Mesh adaptive direct search algorithms for constraint optimization. *SIAM Journal on Optimization*, 17(1):188–217, 2006.
- [15] R.J. Balling and J. Sobieszczanski-Sobieski. Optimization of coupled systems: A critical overview of approaches. *AIAA Journal*, 34(1):6–17, 1996.
- [16] R.J. Balling and C.A. Wilkinson. Execution of multidisciplinary design optimization approaches on common test problems. *AIAA Journal*, 35(1):178–186, 1997.
- [17] J.C. Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6:154–60, 1994.
- [18] A. Bedford and W. Fowler. *Engineering Mechanics: Dynamics*. Addison Wesley, Menlo Park, CA, 1999.
- [19] A. Bedford and W. Fowler. *Engineering Mechanics: Statics*. Addison Wesley, Menlo Park, CA, 1999.
- [20] D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, second edition, 1999.
- [21] D.P. Bertsekas and J.N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, Belmont, MA, 1997.
- [22] D.P. Bertsekas and J.N. Tsitsiklis. *Operations Research: An Introduction*. Prentice Hall, eighth edition, 2006.
- [23] V.Y. Blouin, J.D. Summers, G.M. Fadel, and J. Gu. Intrinsic analysis of decomposition and coordination strategies for complex design problems. In *Collection of Technical Papers—10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Aug. 30-Sept. 1, 2004.
- [24] B.K. Bose. *Modern Power Electronics and AC Drives*. Prentice Hall, Inc., Upper Saddle River, NJ, 2002.
- [25] R.D. Braun. *Collaborative Optimization: An Architecture For Large-Scale Distributed Design*. Ph.D. dissertation, Stanford University, April 1996.

- [26] British Standards Institution. BS 6841: Measurement and evaluation of human exposure to whole-body mechanical vibration, 1987.
- [27] T.R. Browning. Applying the design structure matrix to system decomposition and integration problems: a review and new directions. *IEEE Transactions on Engineering Management*, 48(3):292–306, 2001.
- [28] S.C. Chapra and R.P. Canale. *Numerical Methods for Engineers*. McGraw-Hill, third edition, 1998.
- [29] B. Chen, J. Liu, Y. Zhong, and J. Zhou. Properties of the coupled factors in MDO and their application in collaborative optimization. In *the Proceedings of DETC'00, ASME 2000 Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Baltimore, MD, Sept. 10–13, 2000.
- [30] L. Chen, Z. Ding, and S. Li. A formal two-phase method for decomposition of complex design problems. *Journal of Mechanical Design, Transactions of the ASME*, 127(2):184–195, 2005.
- [31] L. Chen and S. Li. Analysis of decomposability and complexity for design problems in the context of decomposition. *Journal of Mechanical Design, Transactions of the ASME*, 127(4):545–557, 2005.
- [32] F.H. Clarke. Generalized gradients and applications. *Transactions of the American Mathematical Society*, 205:247–262, 1975.
- [33] Global Climate and Energy Project. A technical assessment of high-energy batteries for light-duty electric vehicles. Technical report, Stanford University, 2006.
- [34] Wells Manufacturing Corp. Making sense of engine airflow. *Counterpoint: The Electronic Diagnostic and Driveability Resource*, 3(3):1–3, 1999.
- [35] E.J. Cramer, J.E. Dennis Jr., P.D. Frank, R.M. Lewis, and G.R. Shubin. Problem formulation for multidisciplinary optimization. *SIAM Journal of Optimization*, 4:754–776, 1994.
- [36] R.L. Daft. *Organization Theory and Design*. Thomson South-Western, Ohio, eighth edition, 2004.
- [37] G.B. Dantzig. *Linear Programming and Extensions*. Princeton University Press, Princeton, NJ, 1963.
- [38] G.B. Dantzig and P. Wolfe. The decomposition principle for linear programs. *Operations Research*, 8:101–111, 1960.
- [39] Davies Craig PTY. LTD., Electric Water Pump Catalogue. www.daviescraig.com.au, (Accessed nov. 8, 2007).
- [40] K.A. De Jong. Genetic algorithms are not function optimizers. In *the proceedings of Foundations of Genetic Algorithms 2*, San Mateo, CA, 1993 1993.

- [41] V. DeMiguel and W. Murray. An analysis of collaborative optimization methods. In *the proceedings of the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Long Beach, CA, Sept. 6–8, 2000.
- [42] V. DeMiguel and W. Murray. A local convergence analysis of bilevel decomposition algorithms. *Optimization and Engineering*, 7(2):99–133, 2006.
- [43] J.E. Dennis Jr. and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. SIAM, Philadelphia, 1996.
- [44] U.S. Department of Transportation. 2001 National Household Travel Survey, Bureau of Transportation Statistics.
www.bts.gov/programs/national_household_travel_survey.
- [45] *PNGV Battery Test Manual Revision 3, DOE/ID-10597*, February 2001.
- [46] M. Doyle, T.H. Fuller, and J. Newman. Modeling of galvanostatic charge and discharge of the lithium/polymer/insertion cell. *Journal of The Electrochemical Society*, 140(6):1526–1533, 1993.
- [47] M. Drăgan. On decomposing large sparse systems of nonlinear equations. In *the proceedings of the 4th International Workshop on Symbolic and Numeric Algorithms for Scientific Computing, Timișoara, Romania*, October 9-12, 2002 2002.
- [48] A.E. Eiben. Evolutionary computing: the most powerful problem solver in the universe? *Dutch Mathematical Archive*, 5/3(2):126–131, 2002.
- [49] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, Germany, 2003.
- [50] R. Fellini. *A Model-Based Methodology for Product Family Design*. Ph.D. dissertation, University of Michigan, 2003.
- [51] R. Fellini, M. Kokkolaras, N. Michelena, P. Papalambros, A. Perez-Duarte, K Saitou, and P. Feynes. A sensitivity-based commonality strategy for family products of mild variation, with application to automotive body structures. *Structural and Multidisciplinary Optimization*, 27:89–96, 2004.
- [52] R. Fellini, M. Kokkolaras, P. Y. Papalambros, and A. Perez-Duarte. Platform selection under performance bounds in optimal design of product families. *Journal of Mechanical Design, Transactions Of the ASME*, 127(4):524–535, 2005.
- [53] R. Fellini, M. Kokkolaras, and P.Y. Papalambros. Quantitative platform selection in optimal design of product families, with application to automotive engine design. *Journal of Engineering Design*, 17(5):429–446, 2006.
- [54] L.J. Fogel, A.J. Owens, and M.J. Walsh. Artificial intelligence through a simulation of evolution. In *Biophysics and Cybernetic Systems*, pages 131–156. Spartan, Washington DC, 1965.

- [55] American Institute for Aeronautics and Astronautics Inc. (AIAA). Current state of the art in multidisciplinary design optimization. Technical report, MDO Technical Committee, 1991.
- [56] T.H. Fuller, M. Doyle, and J. Newman. Simulation and optimization of the dual lithium ion insertion cell. *Journal of The Electrochemical Society*, 141(1):1–10, 1994.
- [57] University of Michigan, Aerospace Engineering display. François-Xavier Bagnoud Building. April, 2006.
- [58] T.B. Gage and M.A. Bogdanoff. Low-emission range extender for electric vehicles. In R.K. Jurgen, editor, *Electric and Hybrid-Electric Vehicles*. SAE International, 2002.
- [59] J.M. Gere. *Mechanics of Materials*. Brooks/Cole, Pacific Grove, CA, fifth edition, 2001.
- [60] J.F. Gieras and M. Wing. *Permanent Magnet Motor Technology*. Marcel Dekker, Inc., New York, 2002.
- [61] J.P. Giesing and J.M. Barthelemy. A summary of industry MDO applications and needs, 1998. AIAA Paper 98-4737.
- [62] M. Giger and P. Ermanni. Evolutionary truss topology optimization using a graph-based parameterization concept. *Structural and Multidisciplinary Optimization*, 32(4):313–326, 2006.
- [63] Robert Bosch GmbH. *BOSCH Automotive Handbook*. John Wiley and Sons, sixth edition, 2004.
- [64] M. Gobbi and G. Mastinu. Analytical description and optimization of the dynamic behaviour of passively suspended road vehicles. *Journal of Sound and Vibration*, 245(3):457–481, 2001.
- [65] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.
- [66] R. Haftka, J. Sobieszczanski-Sobieski, and S.L. Padula. On options for interdisciplinary analysis and design optimization. *Structural Optimization*, 4(2):65–74, 1992.
- [67] R. Haftka and L. Watson. Multidisciplinary design optimization with quasiseparable subsystems. *Optimization and Engineering*, 6:9–20, 2005.
- [68] J. Han. *Sequential Linear Programming Coordination Strategy for Deterministic and Probabilistic Analytical Target Cascading*. Ph.D. dissertation, University of Michigan, 2008.
- [69] S.P. Han. Superlinearly convergent variable metric algorithms for general nonlinear programming problems. *Mathematical Programming*, 11:263–282, 1976.

- [70] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, second edition, 1998.
- [71] F.B. Hildebrand. *Introduction to Numerical Analysis*. McGraw-Hill, New York, second edition, 1974.
- [72] H. Hisashi. Bridgestone Corporation, personal communication, August 2007.
- [73] J.H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, MI, 1975.
- [74] K.F. Hulme and C.L. Bloebaum. Simulation-based comparison of multidisciplinary design optimization solution strategies using cascade. *Structural and Multidisciplinary Optimization*, 19(1):17–35, 2000.
- [75] F.P. Incropera and D.P. DeWitt. *Introduction to Heat Transfer*. John Wiley and Sons, 2002.
- [76] G.J.A. Karsemakers. Numerical investigation of two multidisciplinary engineering design problems using the augmented lagrangian coordination method. Technical report, Technische Universiteit Eindhoven, 2007.
- [77] J.G. Kassakian. Future of power electronics in advanced automotive electrical systems. In *the proceedings of the IEEE Annual Power Electronics Specialists Conference*, pages 7–14, 1996.
- [78] T. Kenjo and S. Nagamori. *Permanent-Magnet and Brushless DC Motors*. Oxford University Press, New York, 1985.
- [79] H.M. Kim. *Target Cascading in Optimal System Design*. Ph.D. dissertation, University of Michigan, 2001.
- [80] H.M. Kim, N.F. Michelena, P.Y. Papalambros, and T. Jiang. Target cascading in optimal system design. *Journal of Mechanical Design, Transactions of the ASME*, 125(3):474–480, 2003.
- [81] C.S. Kirkpatrick, C.D. Gelatt Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.
- [82] M. Kokkolaras, Z. Mourelatos, L. Louca, Z. Filipi, G. Delagrammatikas, A. Stefanopoulou, P.Y. Papalambros, and D. Assanis. Design under uncertainty and assessment of performance reliability of a dual-use medium truck with hydraulic-hybrid powertrain and fuel cell auxiliary power unit. *SAE 2005 Transactions: Journal of Passenger Cars - Mechanical Systems*, 2006. (Selected from the Proceedings of the SAE World Congress, April 11–14, 2005, Detroit, Michigan, paper no. 2005-01-1396).
- [83] J.R. Koza. *Genetic Programming*. MIT Press, Cambridge, MA, 1992.

- [84] R.S. Krishnamachari and P.Y. Papalambros. Optimal hierarchical decomposition synthesis using integer programming. *Journal of Mechanical Design, Transactions of the ASME*, 119(4):440–447, 1997.
- [85] I. Kroo, S. Altus, R. Braun, P. Gage, and I. Sobieski. Multidisciplinary optimization methods for aircraft preliminary design. In *the Proceedings of the 5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, pages 697–707, Panama City Beach, FL, Sept. 7–9 1994.
- [86] I.M. Kroo. AA241—Aircraft Design: Synthesis and Analysis. Course Notes, <http://adg.stanford.edu/aa241/AircraftDesign.html>. (Accessed May 27, 2005).
- [87] I.M. Kroo. An interactive system for aircraft design and optimization. In *the proceedings of the AIAA Aerospace Design Conference*, Irvine, CA, Feb. 3–6 1992.
- [88] A. Kusiak and N. Larson. Decomposition and representation methods in mechanical design. *Journal of Mechanical Design, Transactions of the ASME*, 117B:17–24, 1995.
- [89] A. Kusiak and J. Wang. Decomposition of the design process. *Journal of Mechanical Design, Transactions of the ASME*, 115(4):687–695, 1993.
- [90] J. Larminie and J. Lowry. *Electric Vehicle Technology Explained*. John Wiley and Sons, Chichester, 2003.
- [91] L.S. Lasdon, A.D. Waren, A. Jain, and M. Ratner. Design and testing of a generalized reduced gradient code for nonlinear programming. *ACM Transactions on Mathematical Software*, 4(1):34–50, 1978.
- [92] J.G. Lin. Analysis and enhancement of collaborative optimization for multidisciplinary design. *AIAA Journal*, 42(2):348–360, 2004.
- [93] D.L. Logan. *A First Course in the Finite Element Method*. Brooks/Cole, Pacific Grove, CA, third edition, 2002.
- [94] A.K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 02/ 1977.
- [95] D.E. Malen. Fundamentals of auto body structure design. ME513/AUTO513/MFG513 Course Packet, University of Michigan, 2005.
- [96] J. Markish. Valuation techniques for commercial aircraft program design. Master’s thesis, Massachusetts Institute of Technology, June 2002.
- [97] Matweb material property data. <http://www.matweb.com>. Accessed April 9, 2004.
- [98] P.J. McCleer. Electric drives for pump, fan, and compressor loads in automotive applications. In *the Proceedings of the IEEE International Symposium on Industrial Electronics*, pages 80–85, Jul. 10–14 1995.

- [99] C. Meier, A.A. Yassine, and T.R. Browning. Design process sequencing with competent genetic algorithms. *Journal of Mechanical Design, Transactions of the ASME*, 129(6):566–585, 2007.
- [100] P.H. Mellor, D. Roberts, and D.R. Turner. Lumped parameter thermal model for electrical machines of TEFC design. *IEE Proceedings B: Electric Power Applications*, 138(5):205–218, 1991.
- [101] M. Meyer and A. Lehnerd. *The Power of Product Platforms*. New York: Irwin Professional Publishing, 1997.
- [102] J.J. Michalek and P.Y. Papalambros. An efficient weighting update method to achieve acceptable consistency deviation in analytical target cascading. *Journal of Mechanical Design, Transactions Of the ASME*, 127(2):206–214, 2005.
- [103] N.F. Michelena and P.Y. Papalambros. Network reliability approach to optimal decomposition of design problems. *Journal of Mechanical Design, Transactions of the ASME*, 117(3):433–440, 1995.
- [104] N.F. Michelena and P.Y. Papalambros. Optimal model-based decomposition of powertrain system design. *Journal of Mechanical Design, Transactions of the ASME*, 117(4):499–505, 1995.
- [105] N.F. Michelena and P.Y. Papalambros. A hypergraph framework for optimal model-based decomposition of design problems. *Computational Optimization and Applications*, 8(2):173–196, 1997.
- [106] N.F. Michelena, H. Park, and P.Y. Papalambros. Convergence properties of analytical target cascading. *AIAA Journal*, 41(5):897–905, 2003.
- [107] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Information Sciences*, 7:95–132, 1974.
- [108] E. Oberg, F.D. Jones, H.L. Horton, and H Ryffel, H. *Machinery's Handbook*. Industrial Press Inc., New York, 26th edition, 2000.
- [109] J. Oxley. What is a matroid? *Cubo Matemática Educacional*, 5(3):179–218, 2003.
- [110] P.Y. Papalambros and D.J. Wilde. *Principles of Optimal Design: Modeling and Computation*. Cambridge University Press, New York, second edition, 2000.
- [111] M.J.D. Powell. A fast algorithm for nonlinearly constrained optimisation calculations. In G.A. Watson, editor, *Numerical Analysis Dundee*, pages 144–157. Springer-Verlag, Berlin, 1977.
- [112] D.P. Raymer. *Aircraft Design: A Conceptual Approach*. AIAA Educational Series, third edition, 1999.
- [113] I. Rechenberg. *Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Holzboog Verlag, Stuttgart, 1973.

- [114] J. L. Rogers and C. L. Bloebaum. Ordering design tasks based on coupling strengths. In *the proceedings of the 5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Panama City Beach, FL, Sept. 7–9 1994.
- [115] J.L. Rogers. DeMAID—a design manager’s aid for intelligent decomposition user’s guide. Technical Report TM-101575, NASA, 1989.
- [116] J.L. Rogers. DeMAID/GA - an enhanced design manager’s aid for intelligent decomposition (genetic algorithms). In *the Proceedings of the 6th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, pages 1497–1504, Seattle, WA, Sept. 4–6, 1996.
- [117] P. J. Röhl, B. He, and P. M. Finnigan. A collaborative optimization environment for turbine engine development. In *the Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, St. Louis, MO, Sept. 2–4 1998.
- [118] F. Rothlauf. *Representations for Genetic and Evolutionary Algorithms*. Springer-Verlag, Berlin, second edition, 2006.
- [119] F. Rothlauf, D.E. Goldberg, and A. Heinzl. Network random keys - a tree representation scheme for genetic and evolutionary algorithms. *Evolutionary Computation*, 10(1):75–97, 2002.
- [120] M.W. Sayers and S.M. Karamihas. The little book of profiling. Technical report, University of Michigan Transportation Research Institute, 1998.
- [121] B. Scrosati. Lithium rocking chair batteries: An old concept? *Journal of The Electrochemical Society*, 139(10):2776–2781, 1992.
- [122] P. Seibel. *Practical Common Lisp*. Apress, Berkley,CA, 2005.
- [123] C.C. Smith, D.Y. McGehee, and A.J. Healey. The prediction of passenger riding comfort from acceleration data. *Journal of Dynamic Systems, Transactions of the ASME*, 100:34–41, 1978.
- [124] L.V. Snyder and M.S. Daskin. A random-key genetic algorithm for the generalized traveling salesman problem. *European Journal of Operational Research*, 174(1):38–53, 2006.
- [125] I. Sobieski and I. Kroo. Collaborative optimization using response surface estimation. *AIAA Journal*, 38(10):1931–1938, 2000.
- [126] J. Sobieszczanski-Sobieski. Optimization by decomposition: A step from hierarchic to non-hierarchic systems. In *the Proceedings of the 2nd NASA/USAF Symposium on Recent Advances in Multidisciplinary Analysis and Optimization*, Hampton, VA, Sept. 28–30 1988.

- [127] J. Sobieszczanski-Sobieski, J.S. Agte, and R.R.J. Sandusky. Bilevel integrated system synthesis. *AIAA Journal*, 38(1):164–172, 2000.
- [128] J. Sobieszczanski-Sobieski and R.T. Haftka. Multidisciplinary aerospace design optimization: Survey of recent developments. *Structural Optimization*, 14(1):1–23, 1997.
- [129] A.J. Soper, C. Walshaw, and M. Cross. A combined evolutionary search and multi-level optimisation approach to graph-partitioning. *Journal of Global Optimization*, 29(2):225–241, 2004.
- [130] J.C. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Wiley-Interscience, 2003.
- [131] D. Stanton and D. White. *Constructive Combinatorics*. Springer-Verlag, New York, 1986.
- [132] D.V. Steward. Partitioning and tearing systems of equations. *J. SIAM Numerical Analysis Ser. B*, 2(2):345–365, 1965.
- [133] D.V. Steward. The design structure system: a method for managing the design of complex systems. *IEEE Transactions on Engineering Management*, EM-28(3):71–4, 1981.
- [134] D.V. Steward. *Systems analysis and management: structure, strategy, design*. Petrocelli Books, Inc., 1981.
- [135] N.P. Suh. *The principles of design*. Oxford University Press, New York, 1990.
- [136] L.P. Sullivan. Quality function deployment. *Quality Progress*, 19(6):39–50, 1986.
- [137] B. Surampudi, J. Redfield, G. Ray, A. Montemayor, M. Walls, H. McKee, T. Edwards, and M. Lasecki. Electrification and integration of accessories on a class-8 tractor. In *the proceedings of the SAE World Congress*, Detroit, MI, April 11–14 2005.
- [138] R. Thareja and R. Haftka. Numerical difficulties associated with using equality constraints to achieve multi-level decomposition in structural optimization. In *Collection of Technical Papers - AIAA/ASME/ASCE/AHS 27th Structures, Structural Dynamics and Materials Conference. Part 1: Structures and Materials.*, 1986.
- [139] S. Tosserams. Analytical target cascading: convergence improvement by sub-problem post-optimality sensitivities. Master’s thesis, Eindhoven University of Technology, 2004.
- [140] S. Tosserams, L.F.P. Etman, P.Y. Papalambros, and J.E. Rooda. An augmented Lagrangian relaxation for analytical target cascading using the alternating direction method of multipliers. *Structural and Multidisciplinary Optimization*, 31(3):176–189, 2006.

- [141] S. Tosserams, L.F.P. Etman, and J.E. Rooda. Augmented Lagrangian coordination for distributed optimal design in MDO. *To appear in the International Journal for Numerical Methods in Engineering*, 2007.
- [142] S. Tosserams, L.F.P. Etman, and J.E. Rooda. An augmented Lagrangian decomposition method for quasiseparable problems in MDO. *Structural and Multidisciplinary Optimization*, 34(3), 2007.
- [143] E. Tsang. *Foundations of constraint satisfaction*. Academic Press, San Diego, 1993.
- [144] J. Tuzson. *Centrifugal Pump Design*. John Wiley and Sons, New York, 2000.
- [145] T.C. Wagner. *A General Decomposition Methodology For Optimal System Design*. Ph.D. dissertation, University of Michigan, 1993.
- [146] D.H. Wolpert and W.G. Macready. No free lunch theorem for optimisation. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.
- [147] J.Y. Wong. *Theory of Ground Vehicles*. John Wiley and Sons, New York, third edition, 2001.