

SYSTEM ANALYSIS THROUGH BOND GRAPH MODELING

by

Robert Thomas McBride

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements

For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2005

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Dissertation Committee, we certify that we have read the dissertation prepared by Robert Thomas McBride entitled System Analysis through Bond Graph Modeling and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy

_____ Date: May 3, 2005
Dr. François E. Cellier

_____ Date: May 3, 2005
Dr. Malur K. Sundareshan

_____ Date: May 3, 2005
Dr. Hal S. Tharp

_____ Date: May 3, 2005
Dr. Parviz E. Nikravesh

_____ Date: May 3, 2005
Dr. Cho Lik Chan

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

_____ Date: May 3, 2005
Dissertation Director: Dr. François E. Cellier

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at The University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgment of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgment the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: Robert Thomas McBride

ACKNOWLEDGMENTS

I want to express my gratitude to my advisor Dr. François Cellier. Without his knowledge and experience in controls, and bond graph modeling, this dissertation would not have been possible. His knowledge of modeling, simulation, and control theory advanced my understanding of these topics greatly. Nor could I have accomplished this goal without his guidance, support, and encouragement. I feel very fortunate to have had the opportunity to work with him on my degree. Thank you François!

I want to thank Raytheon for allowing me to pursue this degree by allowing me the time away from work and for the continuing-education financial support. I want to thank in particular; Brett Ridgely for his support and encouragement, Curt Mracek for patiently answering my incessant questions, and Steve Ingle for his input in the review of my writing and his patience with my time away from work.

I want to thank my children for their support during this process; Lucia, Ashton, Anthony, Connor, and Adam. I hope that they will love the university experience as much as I.

I want to thank Kristi Brean and Brook Gomez for their help in taking care of my children during the writing of this dissertation, and while I was away presenting papers. I was always at ease knowing that Adam was safe when I could not be there.

I want to thank my parents, Tom and Jan McBride, for instilling the desire to seek out truth and understanding and allowing me the agency to do so.

— *Robert*

To Shannon,

for her unwavering loyalty,

encouraging support,

and lasting friendship . . .

TABLE OF CONTENTS

LIST OF FIGURES.....	9
LIST OF TABLES.....	15
ABSTRACT.....	16
CHAPTER 1: Introduction.....	17
1.1 Problem Statement.....	17
1.2 Plan of Dissertation.....	18
1.3 Summary of Contributions.....	20
CHAPTER 2: Related Work	21
2.1 Introduction.....	21
2.2 System Lagrangian and Bond Graph Construction	21
2.3 Object-Oriented Bond Graph Modeling	22
2.4 System Efficiency Measurement Through Bond Graph Modeling	23
2.5 Optimal Gain Selection Using the Bond Graph Efficiency Measurement	24
CHAPTER 3: Bond Graph Modeling.....	26
3.1 Introduction.....	26
3.2 An Introduction to Bond Graph Modeling.....	28
3.2.1 Power Bonds and Conjugate Variables.....	28
3.2.2 Bond Graph Junctions.....	31
3.2.3 1-Port Elements.....	32
3.2.4 Basic 2-Port Elements.....	34
3.2.5 Power Flow Diagrams.....	35
3.2.6 Causality	38
3.2.7 Bond Graph Causal Mark Assignments.....	45
3.2.8 Bond Graph Equation Formulation.....	47
3.2.9 Conversion of Bond Graph Variables to Common State Variables	51
3.3 Bond Graph Construction from the Lagrangian	53
3.3.1 Lagrangian to Hamiltonian Transformation	55
3.3.2 Lagrangian to Bond Graph Development	59
3.3.3 Lagrangian to Bond Graph Development: Pendulum Example, One Degree of Freedom	62
3.3.4 Lagrangian to Bond Graph Development: Gyroscope Example	66
3.4 Brown's Lagrangian Bond Graphs	80
3.5 Conclusions.....	82
CHAPTER 4: Object-Oriented Bond Graph Modeling	84
4.1 Introduction.....	84
4.2 Dymola.....	85
4.2.1 Equation Sorting	87
4.2.2 Algebraic Loops.....	92
4.2.3 Structural Singularities: The Higher Index Problem	100
4.3 The Dymola Bond Graph Library.....	108
4.3.1 Connectors	108
4.3.2 Bonds	110

TABLE OF CONTENTS (continued)

4.3.3	Junctions	113
4.3.4	Passive Elements.....	116
4.3.5	Sources.....	122
4.3.6	Sensors.....	124
4.4	A Gyroscopically Stabilized Platform: An Object-Oriented Bond Graph Example	129
4.4.1	The Gyroscope Model.....	129
4.4.2	Inertial Rate Sensor Model	133
4.4.3	Platform Model	140
4.4.4	Stabilized Platform.....	143
4.4.5	Camera Model.....	144
4.4.6	Stabilized Platform with a Two-Gimbal Camera.....	147
4.4.7	Simulation and Results	149
4.5	Conclusions.....	157
CHAPTER 5: System Efficiency Measurement Using the Power Flow Information from a Bond Graph Model		
158		
5.1	Introduction.....	158
5.2	Servo Positioning System	159
5.2.1	Servo Positioning System: Complete, Non-Linear System	161
5.2.2	Servo Positioning System: Complete, Linearized System.....	167
5.3	Power Flow Considerations	180
5.4	Servo Controllers	181
5.4.1	Linear Control Schemes.....	181
5.4.2	Non-Linear Control Scheme 1	183
5.4.3	Non-Linear Control Scheme 2	185
5.5	Step Response Comparisons	188
5.5.1	Step Response Comparisons of Linear Systems.....	188
5.5.2	Step Response Comparisons of Non-Linear Systems.....	204
5.6	Conclusions.....	214
CHAPTER 6: Optimal Gain Comparison Using the Power Flow Information of Bond Graph Modeling		
215		
6.1	Introduction.....	215
6.2	Two Degree of Freedom Missile	217
6.3	Linear Pitch Autopilot.....	226
6.3.1	Missile Pitch Autopilot: 3-Loop Controller.....	226
6.3.2	Missile Pitch Autopilot with Actuator Dynamics.....	230
6.4	The Autopilot Gain Selection Process	235
6.4.1	The Autopilot Gain Selection Process: The Performance Index	235
6.4.2	Linearized Missile Pitch Dynamics for Gain Optimization.....	236
6.4.3	Linear Missile Pitch Dynamics: Sample Optimal Gains	245
6.5	Actuator Power Flow Analysis Using Optimal Autopilot Gains.....	251
6.5.1	Actuator Power Flow Efficiency from the Optimal Gain Set.....	251

TABLE OF CONTENTS (continued)

6.5.2	Optimal Efficiency Comparisons.....	258
6.6	Nonlinear Pitch Autopilot: An SDRE Approach.....	264
6.6.1	LQR Formulation and General Solution.....	264
6.6.2	LQR General Solution for Nonzero Feed-Through.....	265
6.6.3	LQR Solution for Nonzero Feed-Through and Output Feedback	268
6.6.4	LQR Tracking Solution for Nonzero Feed-Through, Output Feedback and Zero Steady State Error.....	269
6.6.5	Dymola Implementation of the LQR Tracking Solution	271
6.6.6	Nonlinear Autopilot Results	279
6.7	Power Flow Analysis with Varying Mass Parameters.....	286
6.7.1	Center of Gravity Shift.....	286
6.8	Conclusions.....	291
CHAPTER 7: Summary		293
7.1	Contributions.....	293
7.1.1	Modeling.....	293
7.1.2	Simulation.....	293
7.1.3	System Analysis.....	294
7.1.4	Controller Design.....	295
7.2	Future Work.....	296
7.2.1	Modeling.....	296
7.2.2	Simulation.....	296
7.2.3	System Analysis and Controller Design	297
APPENDIX A1: Dymola Model, SDRE Code Listing		298
APPENDIX A2: Dymola Model, Riccati4 Code Listing		302
APPENDIX A3: Dymola Model, Heig4 Code Listing		306
APPENDIX A4: Dymola Model, Gen_Eigs Code Listing		310
APPENDIX A5: Dymola Models, Misc. Functions, Code Listing		313
A5.1	QuadRoots.....	313
A5.2	Complex_Mult.....	313
A5.3	Complex_Div.....	313
APPENDIX B1: Symmetry of Hamiltonian Eigenvalues		315
B1.1	Eigenvalue Symmetry about the Real Axis	315
B1.2	Hamiltonian Eigenvalue Symmetry about the Imaginary Axis	316
APPENDIX B2: Vandermonde Representation of Controller Canonical Eigenvectors		317
APPENDIX C: Glossary of Terms.....		319
REFERENCES		320

LIST OF FIGURES

Figure 3.1.	Power Bond with the Flow of Power from A to B	29
Figure 3.2.	Bond Graph Junctions and Conjugate Variable Relationships	31
Figure 3.3.	Ideal Sources	33
Figure 3.4.	Basic Bond Graph 1-Port Elements	34
Figure 3.5.	Basic 2-port Elements	34
Figure 3.6.	Circuit Example	36
Figure 3.7.	Simplified Circuit Power flow Diagram	37
Figure 3.8.	Completed Circuit Power flow Diagram	38
Figure 3.9.	Causal Marks	39
Figure 3.10.	Necessary Causality	40
Figure 3.11.	Possible Causal Assignments for 2-Port Elements	41
Figure 3.12.	Integral Causal Assignments for 1-Port Elements	41
Figure 3.13.	Differential Causal Assignments for 1-Port Elements	42
Figure 3.14.	Possible Causal Assignments for a Resistive Element	43
Figure 3.15.	Causal Assignments on Bond Graph Junctions	44
Figure 3.16.	Completed Circuit Bond Graph	47
Figure 3.17.	General Bond Graph Structure Developed from the Lagrangian	61
Figure 3.18.	Single Degree of Freedom Pendulum	62
Figure 3.19.	Pendulum 1-Junction	64
Figure 3.20.	Complete Bond Graph of Pendulum	65
Figure 3.21.	Gyroscope Diagram	67
Figure 3.22.	Gyroscope Integral 1-Junctions	70
Figure 3.23.	Gyroscope 1-Junctions: <i>I</i> -Element Connections	71
Figure 3.24.	Gyroscope MGY Connections	74
Figure 3.25.	Complete Gyroscope Bond Graph	75
Figure 3.26.	Ball Joint Table: Schematic	80
Figure 3.27.	Ball Joint Table: Brown's Bond Graph	81
Figure 3.28.	Ball Joint Table: Bond Graph from Current Method	81
Figure 4.1.	Power Sensing Bond Model: Equation Window	85
Figure 4.2.	Power Sensing Bond Model: Diagram Window	86
Figure 4.3.	Power Sensing Bond Model: Icon Window	87
Figure 4.4.	Spring Mass Damper: Example	88
Figure 4.5.	Spring Mass Damper: Dymola Equation Window	89
Figure 4.6.	Wheatstone Bridge Circuit Example: Dymola Diagram Window	92
Figure 4.7.	Bond Graph of Wheatstone Bridge Circuit	93
Figure 4.8.	Complete Bond Graph of Wheatstone Bridge Circuit	94
Figure 4.9.	Gear Train Example	101
Figure 4.10.	Gear Train Bond Graph	103
Figure 4.11.	Bond Graph Connector	108

LIST OF FIGURES (continued)

Figure 4.12.	<i>e</i> -Bond Connector	109
Figure 4.13.	<i>f</i> -Bond Connector.....	110
Figure 4.14.	A-Causal Bond.....	111
Figure 4.15.	<i>e</i> -Bond.....	112
Figure 4.16.	<i>f</i> -Bond	112
Figure 4.17.	3 Bond 0-Junction.....	113
Figure 4.18.	<i>Three-Port Zero</i>	114
Figure 4.19.	<i>Three-Port One</i>	115
Figure 4.20.	3 Bond 1-Junction.....	115
Figure 4.21.	<i>Passive One-Port</i>	116
Figure 4.22.	R-Element Model.....	117
Figure 4.23.	mR-Element Model.....	118
Figure 4.24.	I-Element Model	118
Figure 4.25.	C-Element Model.....	119
Figure 4.26.	<i>TwoPort</i>	119
Figure 4.27.	Transformer Model	120
Figure 4.28.	Modulated Transformer Model.....	120
Figure 4.29.	Gyrator Model.....	121
Figure 4.30.	Modulated Gyrator Model	121
Figure 4.31.	Effort Source Model	122
Figure 4.32.	Modulated Effort Source Model	123
Figure 4.33.	Flow Source Model.....	123
Figure 4.34.	Modulated Flow Source Model	124
Figure 4.35.	Effort Sensor	125
Figure 4.36.	Flow Sensor	125
Figure 4.37.	<i>P</i> Sensor	126
Figure 4.38.	<i>Q</i> Sensor.....	127
Figure 4.39.	Power Sensor on an <i>e</i> -Bond.....	127
Figure 4.40.	Power Sensor on an <i>e</i> -Bond: Icon Window	128
Figure 4.41.	Gyroscope Model.....	130
Figure 4.42.	Gyroscope Model: Equation Window (A).....	130
Figure 4.43.	Gyroscope Model: Equation Window (B).....	131
Figure 4.44.	Gyroscope Model: Icon Window.....	132
Figure 4.45.	Pitch Gyroscope	133
Figure 4.46.	Effective Inertia	134
Figure 4.47.	Pitch Gyro Icon Window	135
Figure 4.48.	Yaw Gyroscope.....	136
Figure 4.49.	Yaw Gyro Icon Window.....	136
Figure 4.50.	Roll Gyro	137
Figure 4.51.	Roll Gyro Icon Window.....	137
Figure 4.52.	Inertial Rate Sensor Model	138
Figure 4.53.	Sensor Delays.....	139
Figure 4.54.	Inertial Rate Sensor Model Icon	139
Figure 4.55.	Platform Channel Bond Graph.....	140

LIST OF FIGURES (continued)

Figure 4.56.	Platform Body	141
Figure 4.57.	Platform Body Icon	142
Figure 4.58.	Platform Body Controller	142
Figure 4.59.	Stabilized Platform	143
Figure 4.60.	Stabilized Platform Icon	144
Figure 4.61.	Camera Model	145
Figure 4.62.	Camera Model Icon	145
Figure 4.63.	Camera Controller	146
Figure 4.64.	Platform and Camera	147
Figure 4.65.	Inertial Commands – Body Motion	148
Figure 4.66.	Platform and Camera Icon	148
Figure 4.67.	Platform and Camera Simulation Model	149
Figure 4.68.	Platform Position Commands (deg)	151
Figure 4.69.	Platform Pitch Response (deg)	152
Figure 4.70.	Platform Yaw Response (deg)	152
Figure 4.71.	Platform Roll Response (deg)	153
Figure 4.72.	Actual and Sensed Achieved Positions (deg)	154
Figure 4.73.	Camera Pitch Response (deg)	155
Figure 4.74.	Camera Yaw Response (deg)	155
Figure 4.75.	Camera Roll Response (deg)	156
Figure 5.1.	Fin Positioning System	160
Figure 5.2.	Motor Bond Graph	161
Figure 5.3.	Motor Bond Graph: Battery	162
Figure 5.4.	Motor Bond Graph: Coil and Shaft	163
Figure 5.5.	Gear Train and Fin Dynamics	165
Figure 5.6.	Backlash Model	166
Figure 5.7.	Linear Fin Dynamics Model (Integral Causal Model)	169
Figure 5.8.	Linear Fin Dynamics Model (Differential Causal Model)	170
Figure 5.9.	Linear Mechanical Fin Model (Integral Causal Model)	171
Figure 5.10.	Linear Mechanical Fin Model (Differential Causal Model)	172
Figure 5.11.	Bode Comparison for Different Values of KBL	175
Figure 5.12.	Linear Actuator with Fin Dynamics	179
Figure 5.13.	<i>PID</i> Bode Plots	182
Figure 5.14.	Linear Controller/Actuator	183
Figure 5.15.	Non-Linear Controller 1	184
Figure 5.16.	Non-Linear Controller 2	185
Figure 5.17.	Content of the Y3 Element	187
Figure 5.18.	Linear Actuator with Power Signal Analysis	188
Figure 5.19.	Dymola Code for Vector Normalization	189
Figure 5.20.	Three PID Actuators with Power Signal Analysis	190
Figure 5.21.	5° Step Response, No Hinge Moment	191
Figure 5.22.	Power Input: 5° Step Response, No Hinge Moment	191
Figure 5.23.	Power Output: 5° Step Response, No Hinge Moment	192

LIST OF FIGURES (continued)

Figure 5.24.	Fin Power : 5° Step Response, No Hinge Moment	193
Figure 5.25.	Energy Input: 5° Step Response, No Hinge Moment	194
Figure 5.26.	Energy Output: 5° Step Response, No Hinge Moment.....	194
Figure 5.27.	Fin Energy : 5° Step Response, No Hinge Moment.....	196
Figure 5.28.	Integral(Fin Energy): 5° Step Response, No Hinge Moment.....	196
Figure 5.29.	Efficiency Worst Case	197
Figure 5.30.	5° Step Response, Hinge Moment = -0.6 (N*m/deg.)	199
Figure 5.31.	η : 5° Step Response, Hinge Moment = -0.6 (N*m/deg.).....	199
Figure 5.32.	5° Step Response, Hinge Moment = -6 (N*m/deg.)	200
Figure 5.33.	η : 5° Step Response, Hinge Moment = -6 (N*m/deg.).....	200
Figure 5.34.	20° Step Response, Hinge Moment = 0 (N*m/deg.).....	201
Figure 5.35.	η : 20° Step Response, Hinge Moment = 0 (N*m/deg.)	201
Figure 5.36.	20° Step Response, Hinge Moment = -0.6 (N*m/deg.)	202
Figure 5.37.	η : 20° Step Response, Hinge Moment = -0.6 (N*m/deg.)	202
Figure 5.38.	20° Step Response, Hinge Moment = -6 (N*m/deg.)	203
Figure 5.39.	η : 20° Step Response, Hinge Moment = -6 (N*m/deg.).....	203
Figure 5.40.	Two Non-Linear Actuators with Power Signal Analysis	205
Figure 5.41.	5° Step Response, Hinge Moment = 0 (N*m/deg.).....	205
Figure 5.42.	Quantizer I/O for 5° Step Response, No Hinge Moment.....	206
Figure 5.43.	5° Step Response (zoom), Hinge Moment = 0 (N*m/deg.).....	207
Figure 5.44.	η : 5° Step Response, Hinge Moment = 0 (N*m/deg.)	207
Figure 5.45.	5° Step Response, Hinge Moment = -0.6 (N*m/deg.)	208
Figure 5.46.	η : 5° Step Response, Hinge Moment = -0.6 (N*m/deg.)	208
Figure 5.47.	5° Step Response, Hinge Moment = -6 (N*m/deg.)	209
Figure 5.48.	η : 5° Step Response, Hinge Moment = -6 (N*m/deg.).....	209
Figure 5.49.	20° Step Response, Hinge Moment = 0 (N*m/deg.).....	210
Figure 5.50.	20° Step Response (zoom), Hinge Moment = 0 (N*m/deg.).....	211
Figure 5.51.	η : 20° Step Response, Hinge Moment = 0 (N*m/deg.)	211
Figure 5.52.	20° Step Response (zoom), Hinge Moment = -0.6 (N*m/deg.).....	212
Figure 5.53.	η : 20° Step Response, Hinge Moment = -0.6 (N*m/deg.).....	212
Figure 5.54.	20° Step Response (zoom), Hinge Moment = -6 (N*m/deg.).....	213
Figure 5.55.	η : 20° Step Response, Hinge Moment = -6 (N*m/deg.).....	213
Figure 6.1.	Autopilot Loop with the Autopilot Design Assumption.....	215
Figure 6.2.	Two Degree of Freedom Missile	218
Figure 6.3.	Missile Distance Definitions.....	220
Figure 6.4.	Missile Pitch Plane Dynamics Bond Graph.....	221
Figure 6.5.	Dymola Pitch Plane Dynamics: Diagram and Icon Windows.....	222
Figure 6.6.	Dymola Pitch Plane Dynamics: Equation Window	222
Figure 6.7.	Dymola Pitch Plane Dynamics: Parameter Values.....	223
Figure 6.8.	Wing and Fin Chord Definitions.....	224
Figure 6.9.	Dymola Pitch Plane Instantiation.....	224
Figure 6.10.	Angle of Attack.....	225
Figure 6.11.	Missile Body Acceleration.....	225
Figure 6.12.	Classic Three Loop Autopilot.....	226

LIST OF FIGURES (continued)

Figure 6.13.	Classic Three Loop Autopilot: Dymola Model	227
Figure 6.14.	Three Loop AP: Closed Loop System	227
Figure 6.15.	Three Loop AP: Angle of Attack	228
Figure 6.16.	Three Loop AP: Achieved Acceleration	229
Figure 6.17.	Three Loop AP: Necessary Fin Deflection	229
Figure 6.18.	Three Loop AP: Closed Loop System with Actuator	231
Figure 6.19.	Angle of Attack with Actuator Dynamics	232
Figure 6.20.	Achieved Acceleration with Actuator Dynamics	232
Figure 6.21.	Necessary Fin Deflection with Actuator Dynamics	233
Figure 6.22.	Angle of Attack with Actuator Dynamics: Zoom	233
Figure 6.23.	Achieved Acceleration with Actuator Dynamics: Zoom	234
Figure 6.24.	Necessary Fin Deflection with Actuator Dynamics: Zoom	234
Figure 6.25.	Linearized Pitch Plane Block Diagram	239
Figure 6.26.	Open Loop Linear and Nonlinear Angle of Attack	240
Figure 6.27.	Open Loop Linear and Nonlinear Missile Body Acceleration	241
Figure 6.28.	Linear Pitch Plane and Autopilot Block Diagram	242
Figure 6.29.	Closed Loop Linear and Nonlinear Angle of Attack	243
Figure 6.30.	Closed Loop Linear and Nonlinear Missile Body Acceleration	243
Figure 6.31.	Closed Loop Linear and Nonlinear Necessary Fin Deflection	244
Figure 6.32.	Optimal Gain Selection Angle of Attack	248
Figure 6.33.	Optimal Gain Selection Missile Body Acceleration	248
Figure 6.34.	Optimal Gain Selection Necessary Fin Deflection	249
Figure 6.35.	Optimal Gain Selection: $(\text{Command} - AZ)^2$	250
Figure 6.36.	Optimal Gain Selection: Performance Index	250
Figure 6.37.	Nonlinear Missile with Optimal Gains: Angle of Attack	252
Figure 6.38.	Nonlinear Missile with Optimal Gains: Body Acceleration	252
Figure 6.39.	Nonlinear Missile with Optimal Gains: Fin Deflection	253
Figure 6.40.	Nonlinear Missile with Optimal Gains: Body Acc. (zoom)	253
Figure 6.41.	Nonlinear Missile with Optimal Gains: Fin Deflection (zoom)	254
Figure 6.42.	Autopilot η_{AP} : Autopilot Efficiency Signals for Gain Sets 1-3	255
Figure 6.43.	Optimal Gain Set 4: Body Acceleration	257
Figure 6.44.	Optimal Gain Set 4: Body Acceleration (zoom)	257
Figure 6.45.	Optimal Gain Set 4: Autopilot Efficiency η_{AP}	258
Figure 6.46.	Body Acceleration: Gain Sets 4-6	260
Figure 6.47.	Autopilot η_{AP} Gain Sets 4-6	261
Figure 6.48.	Nonlinear PI for Gain Sets 4-6	262
Figure 6.49.	Nonlinear PI for Gain Sets 4-6 (zoom)	262
Figure 6.50.	Autopilot Efficiency η_{AP} : Gain Sets 4-6 (zoom)	263
Figure 6.51.	SDRE Autopilot: Icon and Diagram Window	272
Figure 6.52.	Algebraic Riccati Equation Solver <i>Riccati4</i> : Diagram Window	273
Figure 6.53.	Hamiltonian Eigenvalue Solver <i>Heig4</i> : Diagram Window	275
Figure 6.54.	Pitch Plane Dynamics with SDRE Autopilot	279
Figure 6.55.	Pitch Plane Dynamics with SDRE Autopilot and Actuator	280

LIST OF FIGURES (continued)

Figure 6.56.	SDRE: Achieved Acceleration	281
Figure 6.57.	SDRE: Angle of Attack	281
Figure 6.58.	SDRE: Achieved Fin Deflection.....	282
Figure 6.59.	SDRE: Steady State Gain K_{SS}	282
Figure 6.60.	SDRE: Performance Index.....	283
Figure 6.61.	Achieved Acceleration: SDRE, Set 2, Set 4	284
Figure 6.62.	Angle of Attack: SDRE, Set 2, Set 4	284
Figure 6.63.	Achieved Fin Deflection: SDRE, Set 2, Set 4	285
Figure 6.64.	Autopilot Efficiency η_{AP} : SDRE, Set 2, Set 4	285
Figure 6.65.	Achieved Acceleration: CG Shift	287
Figure 6.66.	Angle of Attack: CG Shift	288
Figure 6.67.	Fin Deflection: CG Shift.....	289
Figure 6.68.	Performance Index: CG Shift.....	289
Figure 6.69.	Unit Step Performance Index: CG Shift (zoom).....	290
Figure 6.70.	Autopilot Efficiency η_{AP} : CG Shift.....	290

LIST OF TABLES

Table 3.1.	Effort and Flow Definitions in Multiple Engineering Domains	30
Table 3.2.	Generic State Variable Definitions in Different Domains	48
Table 4.1.	Camera and Gyro Values used for Simulation	150
Table 5.1.	Model Parameter Values.....	167
Table 5.2.	Linearized Backlash Modeling Options.....	174
Table 5.3.	<i>PID</i> Controllers.....	181
Table 6.1.	Missile Dynamics Variable Description	219
Table 6.2.	Optimal Gain Table.....	247
Table 6.3.	Added Gain Set	256
Table 6.4.	Suboptimal Gain Sets.....	259

ABSTRACT

Modeling and simulation form an integral role in the engineering design process. An accurate mathematical description of a system provides the design engineer the flexibility to perform trade studies quickly and accurately to expedite the design process. Most often, the mathematical model of the system contains components of different engineering disciplines. A modeling methodology that can handle these types of systems might be used in an indirect fashion to extract added information from the model.

This research examines the ability of a modeling methodology to provide added insight into system analysis and design. The modeling methodology used is *bond graph modeling*. An investigation into the creation of a bond graph model using the Lagrangian of the system is provided. Upon creation of the bond graph, system analysis is performed. To aid in the system analysis, an object-oriented approach to bond graph modeling is introduced. A framework is provided to simulate the bond graph directly. Through object-oriented simulation of a bond graph, the information contained within the bond graph can be exploited to create a measurement of system efficiency. A definition of system efficiency is given. This measurement of efficiency is used in the design of different controllers of varying architectures. Optimal control of a missile autopilot is discussed within the framework of the calculated system efficiency.

CHAPTER 1: Introduction

1.1 Problem Statement

Modeling and simulation form an integral role in the engineering design process. An accurate mathematical description of a system provides the design engineer the flexibility to perform trade studies quickly and accurately to expedite the design process. Most often, the mathematical model of the system contains components of different engineering disciplines. The ability to accurately model these types of systems is therefore a necessity among the engineering community.

Bond graph theory began in the 1960's at MIT by H. M. Paynter [Cel91]. The basic idea behind the theory is to create a map of the power flow through a system. Since the first law of thermodynamics applies to all types of energy in all engineering domains [War95], mapping the system's power flow helps the system engineer understand the transfer of power at the boundary of engineering disciplines. Upon developing the power flow diagram, further research lead to causal assignments identifying the causal relationships between state variables [Mon91]. By using the power flow diagram, with the noted causal relationships, creating system equations becomes a relatively straightforward task.

Most often, bond graphs are used to generate system equations. Once the system equations are obtained the bond graph is usually discarded, along with the power flow map and the causal relationship indicators. Useful information is often lost as a result.

Much potential exists to gain further insight into the system model by keeping the power flow diagram and causal indicators. The research presented here provides methods of system analysis by closely monitoring the system power flow through specific bonds of the bond graph model.

1.2 Plan of Dissertation

Chapter 2 is a survey of literature related to the research presented in subsequent chapters. In Chapter 3, a discussion on the creation of bond graphs is given. The discussion explains bond graph basics and then moves to advanced bond graph creation using the Lagrangian of a system [McB01, Mei98, Lag1788].

Upon creation of the bond graph it is usually the modeler's task to formulate system equations and then implement these equations in some executable code in order to perform model simulation. Chapter 4 uses an object-oriented modeling platform called Dymola [Dym], and introduces a bond graph library within this modeling framework [Cel93]. This modeling framework allows the user to build a bond graph in Dymola and simulate the bond graph directly, thus eliminating the need to create further code. Also, since the system model is a bond graph, and not just a set of state equations obtained from a bond graph, it is possible to utilize the bond graph's power flow diagram and causal map to the modeling engineer's advantage. A fairly complex system is modeled in Chapter 4 to demonstrate the flexibility provided by the bond graph library.

Chapter 5 provides a means of measuring the efficiency of a system by monitoring the power input to the system and the power delivered as output. The monitoring of power on any element of the model is a straightforward task when using a bond graph, since the bond graph itself is a power flow map of the system. The efficiency measurement is used as a benchmark for comparing controllers of different topologies. It is shown that this analysis is not limited to linear systems only, but is equally effective for nonlinear systems. The efficiency measurement of a system is common to thermostatic problems [Bej97, Cen89, War95]. Bond graph modeling allows this efficiency analysis to be performed on dynamic systems of all engineering domains [McB05c].

Chapter 6 utilizes the efficiency measurement obtained in Chapter 5 to compare the system efficiencies for a controller with different gain sets [McB05a]. In doing this comparison the control design engineer obtains a measurement of optimality of the system. The optimal efficiency signal is created by using a state dependent Riccati equation approach. This optimal efficiency signal is then compared to efficiency signals obtained from a linear controller. The usefulness of this analysis is that a limit of efficiency is obtained such that the control design engineer is alerted to gain sets that violate linear constraints on nonlinear systems. Also, it is shown that the efficiency signal can further be used to determine the need for controller gain redesign given that the original system parameters have changed.

1.3 Summary of Contributions

The main contributions of this thesis are as follows:

- A method for creating a bond graph from the Lagrangian of a system is given. The method presented here further improves upon methods that have been presented previously.
- An object-oriented bond graph library is given. The library allows the modeling engineer to build a bond graph such that the bond graph is an executable code. In this way potential errors are eliminated in that the bond graph modeler does not need to generate state equations.
- The bond graph library allows the modeling engineer to use the power flow diagram directly. The power flow diagram provides a means for measuring the efficiency of a system. A definition of system efficiency is given.
- It is shown how the system efficiency measurement can be used to compare control schemes of different architectures. Both linear and nonlinear controllers are compared.
- It is shown how the system efficiency signal can be used to measure the optimality of a constrained optimization design.
- It is shown how the efficiency signal can help determine if a nonlinear system approaches the violation of linear constraints.
- It is shown how the efficiency signal can be used to determine if a controller redesign is necessary for a system of which the parameter values have changed.

CHAPTER 2: Related Work

2.1 Introduction

This chapter presents an overview of work related to this research. Bond graph research is often concerned with the development of system state equations [Kar90, Kar83]. This research, however, focuses on system analysis that can be done directly from the bond graph itself.

2.2 System Lagrangian and Bond Graph Construction

Early on in the development of bond graph theory Karnopp presented a method for generating a bond graph from the Lagrangian of a system [Kar69]. The Lagrangian bond graph method shown by Karnopp gives correct but complicated bond graph structures. The method of Karnopp was later improved upon in Brown's presentation of *Lagrangian Bond Graphs* [Bro72]. Brown's method also provides a correct bond graph structure but uses complicated formulae involving inertia terms for transformer and gyrator moduli. The method presented in this research further improves upon the method of Brown to reduce the complexity of transformer moduli [McB01]. An in depth discussion is presented in Sections 3.3 and 3.4.

Research is being conducted to apply Lagrangian bond graphs with a finite element discretization scheme to simulate a wide range of high order, solid continuum dynamics problems [Fah99, Fah94]. Also, Lagrangian bond graphs have been used to develop

formalism for modeling kinematic joints [Fav98, Zei95, Mas91]. The applications of Lagrangian bond graphs are many [Bro72, McB05a, McB05b, McB05c]. This research provides further insight into the creation of the Lagrangian bond graph by noting natural gyrator modulations of system I -element.

A similar research path is currently under investigation. This path is concerned with the creation of Lagrange equations given a bond graph structure. By obtaining conservation laws of different energy domains, using bond graphs, Lagrangian-Hamiltonian mechanics can be extended to deal with dissipative elements and non-potential fields [Muk05, Muk97, Kar77].

2.3 Object-Oriented Bond Graph Modeling

The ability to use models in a plug-and-play fashion gives the modeler a great advantage in the field of design and simulation. Bond graph modeling allows the user to easily model systems that cross engineering domains. The creation of a bond graph library, within an object-oriented framework, allows the designer to create models that cross multiple engineering domains and simultaneously simulate the models [Cel03a, McB03]. The object-oriented nature of the simulation software allows the reuse of models, eliminates equation generation errors, and removes the users from the difficulties of programming the code. The research presented here provides a bond graph library that can represent electrical, mechanical, hydraulic systems, etc. This library does account for

dissipated energy in the form of heat. However, complete thermodynamic systems cannot be modeled using this library.

Current research in this area involves the creation of a thermodynamic bond graph library. The creation of a bond graph library that deals with thermodynamic systems, in an object-oriented manner, provides the users with the ability to model convective flows [Cel03b, Gre01a, Gre01b].

Similar research in this area is concerned with defining the role of the library designer and the role of the user. The ability of the user to employ the library's models with confidence that the components operate correctly, without knowing their internal workings, provides the modeler with greater flexibility [Urq03a, Urq03b]. A bond graph methodology, being based on the first law of thermodynamics, helps provide confidence in the correctness of the library components.

2.4 System Efficiency Measurement Through Bond Graph Modeling

Power flow information of a bond graph can be used to develop the state equations for a given system. However, once the equations of motion are obtained, often the power flow map and the system's causal relationships are discarded. As a result, useful information is lost. This research uses the power flow information of a bond graph to develop a method for measuring the efficiency of a system. By monitoring the input power, and the output power, an efficiency measurement can be defined [McB05c].

A similar research path uses the power flow information from a bond graph of a system to measure the relative value of energy elements in a system. By evaluating the amount of energy used by a particular branch of the system, it is possible to determine which system elements are relatively unimportant. The relatively unimportant elements can then be eliminated, thus reducing the overall size of the system. Thus, a systematic approach to system order reduction can be formulated [Lou02, Lou99]. This research path provides a further use for the power flow map, and causal information map, that are naturally obtained in a bond graph model.

2.5 Optimal Gain Selection Using the Bond Graph Efficiency

Measurement

The thermodynamic power flow of a system, and the causal relationships among system variables, provide insight that can be exploited to develop a controller for the system. Bond graphs naturally provide the power flow through the system and the system's internal causal relationships. Extracting information from a bond graph of a system to aid in the system's controller design is ongoing research. The research presented here focuses on the definition of autopilot efficiency. This analysis can be used to compare different controller designs, or to compare efficiencies of different gain sets, within the same design [McB05a, McB05b].

There is current research that utilizes the causal properties of the bond graph to determine structural control properties of thermo-fluid systems. The causal information

of a thermo-fluid bond graph can be utilized to ultimately design sensor placements for observability and fault detection isolation [Sar04]. This naturally leads to an investigation of the relationship between the causal loops/paths in a system and the system controllability/observability. Research on bond graph based methods for analysis and design of control systems is ongoing [Jun05].

Also, a research branch exists that is interested in utilizing the bond graph method of model reduction to formulate a control law for large-scale systems [Liu02]. The bond graph model reduction method is a method that can be applied to nonlinear models, as well as, linear models [Lou02, Lou99].

CHAPTER 3: Bond Graph Modeling

3.1 Introduction

Modeling and simulation form an integral role in all of science and engineering. A scientist can iterate through the scientific process at a much greater pace by performing experiments on a model of a system versus experimentation on a full-scale system. Engineers are able to iterate quickly through the design process by modeling their designs prior to implementing them in hardware. Control engineers, for example, use modeling by first attempting to control a model of a system prior to controlling the actual system.

The process of modeling is one in which a set of cause and effect relationships are defined between parameters that represent physical characteristics of a system. These parameters, or variables, are chosen such that the key information for understanding the system can be extracted from the model through simulation. Simulation is then the ability to view how the model acts over a period of time. Understanding the cause and effect relationships between the variables of a system is essential to developing a meaningful model.

The modeling process exists in all science and engineering domains. Electrical engineers use circuit diagrams to represent electrical circuits, mechanical, and civil engineers use free-body diagrams to represent forces and moments between components of their respective systems. Thermodynamics and chemical system descriptions yet use other techniques to help the user develop the necessary cause and effect relationships

between the describing parameters of the system. Each technique of the various domains differs from one another in that they each describe different aspects of the physical world. Each technique has meaning only in the engineering domain for which it is intended. Thus, systems that cross multiple disciplines of science and engineering can, therefore, be very difficult to model.

The laws of thermodynamics are relevant to systems in all science and engineering domains. The first law of thermodynamics states that energy cannot be created nor destroyed but simply changes from one form to another. By modeling the flow of energy from one form to another, a methodology that describes systems in multiple energy domains is obtained. One such methodology is *bond graph* modeling.

Bond graph modeling lends itself very well to assisting the user in the organization of cause and effect information. It is a methodology that maps power flow throughout the system. Bond graphs also map signal flow throughout the system allowing the user to define the cause and effect relationships between all describing variables of the system. Since power flow laws are the same regardless of the energy domain that is being described, bond graphs are able to connect model sub-systems of different domains together to form a larger, mixed-domain model, in a concise and meaningful way. The ability to map power flow across energy domain boundaries, and map signal flow information across the same boundaries, is an indispensable aid in the user's quest to form cause and effect relationships within interdisciplinary systems.

Bond graph modeling is a graphical modeling technique that preserves the computational structure and the topological structure of the system being modeled. H.M.

Paynter, an MIT professor developed bond graph modeling as an interdisciplinary modeling technique that simultaneously conveys the topological structure and the computational structure of a system [Cel91, pp. 258-265].

3.2 An Introduction to Bond Graph Modeling

The first law of thermodynamics states that energy is neither created nor destroyed, but is simply transformed from one form to another [Cen89, pp. 23, 80]. Bond graph modeling maps the flow of power through a system. By keeping track of the power in a system, the energy is accounted for as well since energy is the time integral of power. Power is a convenient entity in modeling, since it can be described as the multiplication of two conjugate variables regardless of the engineering domain of its origin. A bond graph maps the power flow through a system and simultaneously describes the relationships between the conjugate variables in each branch of the system. In this way, an accounting of all the energy of a system, and the conjugate variable relationships, are used to develop the describing equations of a system.

3.2.1 Power Bonds and Conjugate Variables

Bond graphs represent the power flow through a system by using a series of connections called power bonds. Figure 3.1 shows a half arrow power bond symbolizing the power flow from point A to point B. Each power bond has a set of conjugate

variables associated with it. Naturally, the multiplication of these conjugate variables is power.

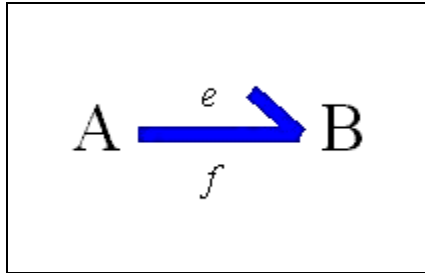


Figure 3.1. Power Bond with the Flow of Power from A to B

Each conjugate variable pair is made up of one *flow* variable and one *effort* variable. Figure 3.1 represents the effort and flow variables with an e on the harpoon side of the bond, representing the *effort*, and an f on the opposite side, representing the *flow*. The *effort/flow* conjugate combination exists regardless of the engineering discipline that the conjugate pair describes. For example, a conjugate combination found in the electrical domain is current, as a flow variable, and voltage as the effort variable. Thermodynamics uses entropy flow, as the flow variable, and temperature as the effort variable. The multiplication of both sets of conjugate variables is power. By accounting for these conjugate combinations throughout the system in question, a methodology for deriving system equations can be established even for systems that cross multiple engineering disciplines. Table 3.1 lists examples of conjugate variables that are commonly found in engineering systems.

	Effort	Flow
	e	f
Electrical	Voltage u [V]	Current i [A]
Translational Motion	Force F [N]	Velocity v [m/s]
Rotational Motion	Torque T [N*m]	Angular Velocity ω [rad/sec]
Hydraulic	Pressure p [N/m ²]	Volumetric Flow q [m ³ /sec]
Chemical	Chemical Potential μ [J/mole]	Molar Flow v [mole/sec]
Thermodynamic	Temperature T [K]	Entropy Flow dS/dt [W/K]

Table 3.1. Effort and Flow Definitions in Multiple Engineering Domains

Bond graph modeling is able to model systems that cross engineering domains by keeping track of the effort/flow conjugate combinations throughout a multi-discipline system. Often, the conjugate combinations in a bond graph are simply expressed with the generalized variables e and f . The modeling process is simplified considerably by keeping the conjugate combinations in this generic form.

3.2.2 Bond Graph Junctions

Power bonds are connected together at *junctions*. There are two types of junctions in bond graph modeling. Each of the junction types are set up such that the amount of power coming into the junction equals the amount of power leaving the junction. No creation of power, or power storage, is allowed in a bond graph junction.

The first type of bond graph junction is referred to as a *zero-junction* (0-junction). Each of the power bonds connected to a zero junction have equal effort terms. The flow terms of the power-bonds connected to the zero junction sum to zero, i.e., $\text{flow}_{\text{in}} - \text{flow}_{\text{out}} = 0$, shown in figure 3.2.

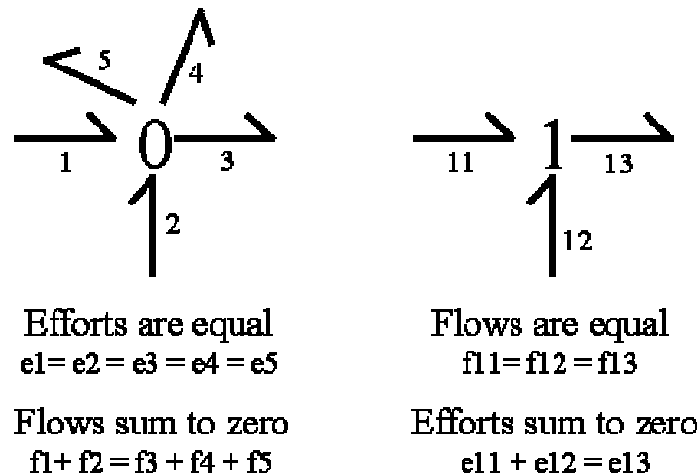


Figure 3.2. Bond Graph Junctions and Conjugate Variable Relationships

The second type of bond graph junction is a *one-junction* (1-junction). The power bonds connected to a one junction have equal flow terms. The effort terms of the power-bonds connected to the one junction sum to zero, i.e., $\text{effort}_{\text{in}} - \text{effort}_{\text{out}} = 0$.

Figure 3.2 shows a zero and a 1-junction and the implied meanings of the effort and flow variables. The 0-junction of figure 3.2 shows five power bonds connected to the junction, and the 1-junction of figure 3.2 shows three power bonds connected to the junction to indicate that each junction may have an unlimited number of power bond connections. The power bonds, on each of the junctions in figures 3.2, are arbitrarily numbered to keep track of the conjugate variables associated with them.

It is clear from figure 3.2 that each type of junction conserves power in that power into the junction is equal to the power out of the junction. By holding one of the conjugate variables equal on all bonds connected to the junction, the other conjugate variable must then sum to zero, i.e., incoming minus outgoing equals zero.

3.2.3 1-Port Elements

Bond graphs use five types of idealized *1-port* elements. Two of these elements are *active* and the remaining three are *passive*. The two *active 1-port* elements are the idealized bond graph sources, consisting of an *effort source* and a *flow source*. Each of these elements is shown in figure 3.3. Bond graph sinks are represented by reversing the direction of the power arrow opposite that shown in figure 3.3.

Bond graphs use three types of idealized *passive 1-port* elements. Two of these elements are energy storage elements and the other is a dissipative element. Each of these elements exchanges power from one form to another in their own unique way. These elements are considered to be *passive*, since they do not contain any sources of

power. They are called *1-port* elements, since the exchange of power from one form to another occurs at a single location or *port* [Cel91, Kar83, Kar90].

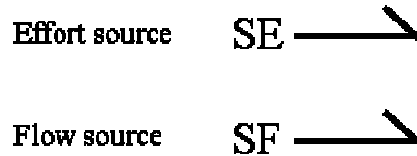


Figure 3.3. Ideal Sources

The three types of passive *1-port* elements are represented by an R , I , and C for resistive, inductive, and capacitive, respectively. Figure 3.4 shows each of these elements in bond graph notation. The resistive element represents electrical resistance, mechanical friction, thermal resistance, etc., depending on the domain in which it is used. The inductive element represents electrical inductance, mechanical mass, or rotational mass moment of inertia depending on the domain in which it is used. The capacitive element represents electrical capacitance, mechanical compliance, thermal capacitance, hydraulic capacitance, etc., depending on the domain in which it is used. Each of the passive *1-port* elements has a single power bond attached. The single power bond, with the element at the end of the bond, shows the exchange of power at a single location.

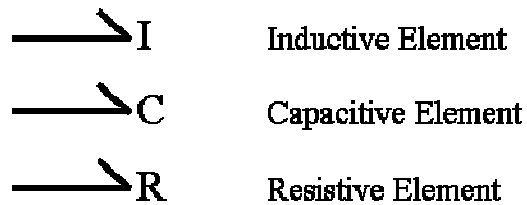


Figure 3.4. Basic Bond Graph 1-Port Elements

3.2.4 Basic 2-Port Elements

Two types of basic *2-port* elements exist in bond graph modeling. These elements are used at the boundaries of different engineering domains. Similar to the passive *1-port* elements the *2-port* elements do not contain power sources, thus they are passive. Also, the *2-port* elements, do not store, or dissipate power. For each of the *2-port* elements, power-in equals power-out.

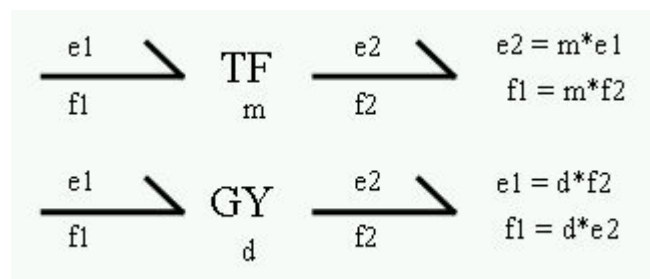


Figure 3.5. Basic 2-port Elements

The *2-port* elements are a transformer, represented by a *TF*, and a gyrator represented by a *GY*. Figure 3.5 shows each of these elements and the implied relationships among

the conjugate variables. As seen in figure 3.5, the transformer relates the effort on one side of the transformer to the effort on the other side, and the flow on one side to the flow on the other. The gyrator relates the effort on one side of the gyrator to the flow on the other. Note that the relationships shown in figure 3.5 satisfy the power-in equals power-out criterion.

An example of the use of a transformer is an ideal mechanical gear train. An ideal gear train does not store or dissipate power. The angular velocity of the output gear is a multiple of the angular velocity of the input gear. The input torque is the same constant multiplied by the output torque. This multiple is represented in figure 3.5 by the symbol m .

An example of a gyrator is an ideal electric motor. The angular velocity of the motor shaft is a multiple of the input voltage. The motor current is the same constant multiplied by the shaft torque. This multiple is represented in figure 3.5 by the symbol d .

Nonlinear transformers and gyrators use the same relationships as described in figure 3.5. The difference is that the modulus is allowed to vary with time. These elements are represented by an *MTF*, for a *modulated transformer*, and an *MGY*, for a *modulated gyrator*.

3.2.5 Power Flow Diagrams

At this point, all basic elements of bond graph modeling have been presented. The first step in creating a bond graph is to create a power flow diagram. The power flow

diagram is created by connecting the bond graph elements presented above, such that the power flow through the system is mapped. This is best presented via an example. Figure 3.6 shows a circuit diagram and the corresponding power flow diagram. The power flow diagram represents the power flow through the circuit. The bonds shown in the power flow diagram of figure 3.6 have been arbitrarily numbered to facilitate discussion.

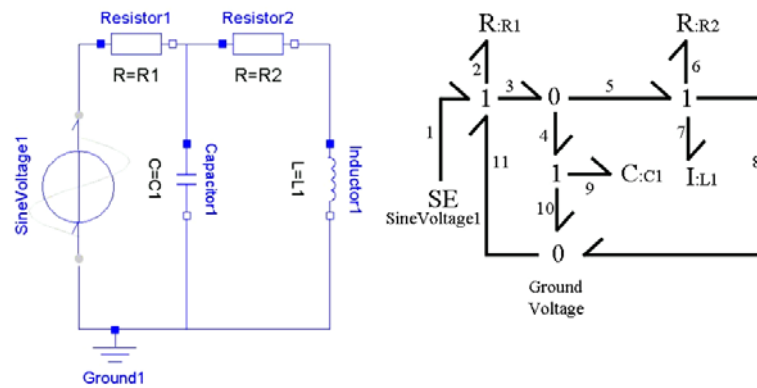


Figure 3.6. Circuit Example

The sinusoidal input voltage of figure 3.6 is represented as an effort source in the power flow diagram. The 1-junctions represent locations in the circuit with common current flow. The 0-junctions of the power flow diagram represent the nodes of the circuit since, for each node, the voltages (efforts) are the same across each path of the node. Series circuit elements are connected at 1-junctions since the current flowing through the elements is the same for each element.

As seen in figure 3.6 the power flow diagram preserves the topological structure of the circuit diagram. The power flow diagram is an intuitive representation of the power flow through the circuit.

The power flow diagram of figure 3.6 can be simplified, however. The voltage represented by the 0-junction that connects bonds 8, 10, and 11 is the ground voltage.

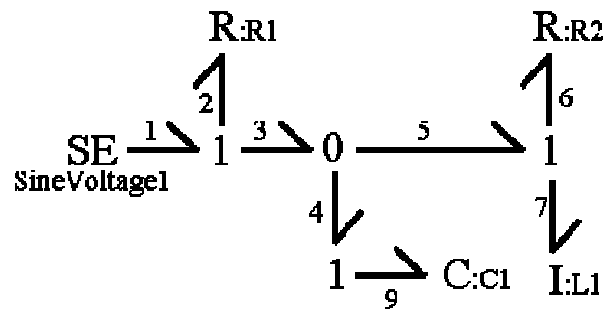


Figure 3.7. Simplified Circuit Power flow Diagram

Since this represents zero volts in the circuit, this 0-junction forces bonds 8, 10, and 11 to have a value of zero for each of their respective effort variables. Thus, the power-bonds 8, 10, and 11 have zero power in them, since power is effort * flow. Bonds with zero power can be erased from the diagram. The simplified diagram is shown in figure 3.7.

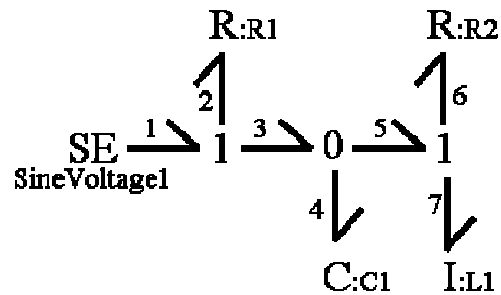


Figure 3.8. Completed Circuit Power flow Diagram

The power flow diagram of figure 3.7 can be simplified yet further. This simplification comes from noting the 1-junction connecting bonds 4 and 9. The flow variables of bonds 4 and 9 are equal by definition of a 1- junction. Also, a 1-junction cannot store or create power so the effort on bond 4 must equal the effort on bond 9. Thus, this 1-junction can be removed and replaced with a single bond. This is true for all junctions that have only two bonds connected [Cel91, Bro01, Kar90]. With this simplification, the power flow diagram is complete and is shown in figure 3.8.

3.2.6 Causality

The power flow diagrams of figures 3.6 through 3.8 are considered *A-causal* bond graphs. They lack one essential bond graph assignment. That is the assignment of bond graph *causality*. Causality shows the direction of the effort and flow information for each bond of the power flow diagram [Kar83]. Upon assigning the direction of the effort/flow information throughout the power flow diagram, the necessary causal relationships

between all of the variables are defined, thus the term *causality*. Figure 3.9 shows two power-bonds with bond graph *causal marks*, and the implied direction of the effort/flow information. Two power-bonds are shown to emphasize the fact that the causal mark is independent of the power flow direction.

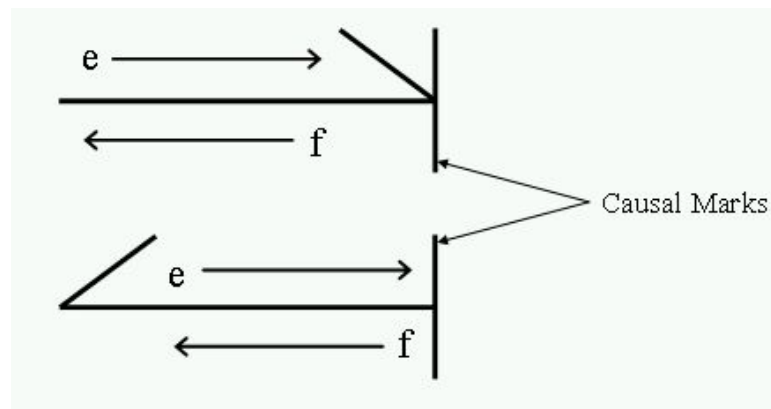


Figure 3.9. Causal Marks

As shown in figure 3.9 the effort information always moves opposite the flow information. The effort information moves toward the causal mark and the flow information moves away from the causal mark [Kar83 pp. 85-89].

Obviously there exists a necessary causal assignment for the bond graph sources. An effort source defines the effort on its connecting power-bond and a flow source defines the flow on its connecting power bond. The necessary causal assignments for bond graph sources are shown in figure 3.10

The *SE* element of figure 3.10 shows that the effort information moves from left to right, as defined by the causal mark. The half arrow shows that the *SE* element is

modeled as a source and not a sink. Similarly, the SF element of figure 3.10 shows that the flow information moves from left to right as defined by the causal mark, and the half arrow depicts this element as a source as well.

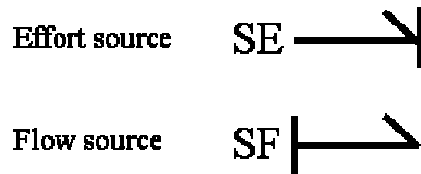


Figure 3.10. Necessary Causality

Figure 3.11 shows the possible combinations for the causal assignments for the 2-port elements. Also shown in figure 3.11 are the implied relationships between the conjugate variables determined by each set of causal marks. The relationships between the conjugate variables are defined by the causal mark, since the signal flow information must remain consistent with the definition of the causal mark shown in figure 3.9. Note; in figure 3.11, the conjugate variable equations for each case maintain the required power-in equals power-out relationship.

The remaining 1-port elements have two possible combinations for causal mark assignments. Each possibility implies specific relationships between the conjugate variables. For the I and C elements there exists either an integral relationship or a differential relationship between the conjugate variables. Figure 3.12 shows the integral relationship with the proper causal mark for the I and C elements.

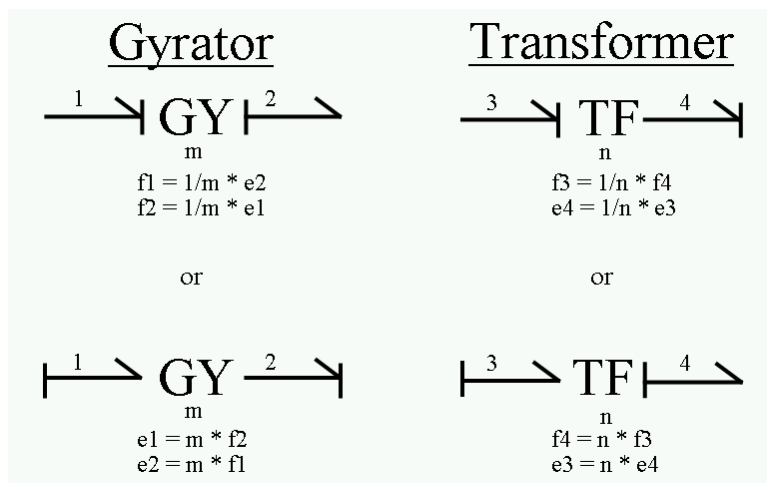


Figure 3.11. Possible Causal Assignments for 2-Port Elements

This type of causal mark is often referred to as *integral causality*. The relationship between the conjugate variables is shown to the right of the bond graph elements in block diagram form. The block diagram helps clarify the meaning of the causal mark and the signal flow information.

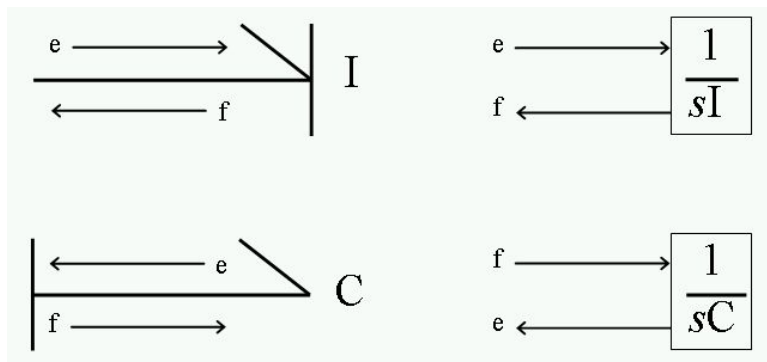


Figure 3.12. Integral Causal Assignments for 1-Port Elements

As seen in figure 3.12 the integrators of a system are implied by the integral causal marks of the system. Thus, for each integral causal mark the order of the system is incremented by one [Bro01, Cel91, Kar83, Kar90].

Figure 3.13 shows the differential causal marks for the 1-port I and C elements. The differential causal mark will occur in a bond graph only when a structural singularity is present in the system [Cel91 pp. 264-265]. See Section 4.2.3 for further discussion on structural singularities.

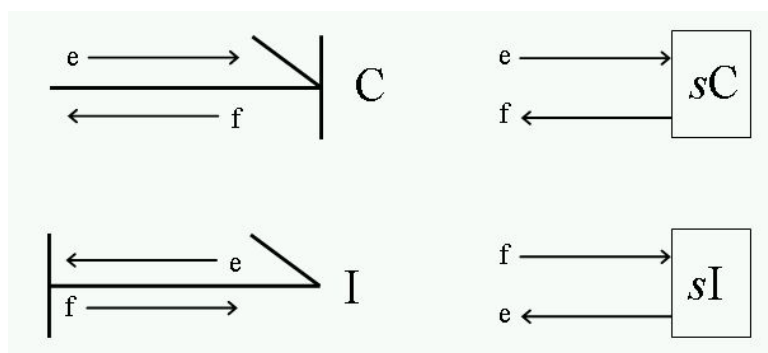


Figure 3.13. Differential Causal Assignments for 1-Port Elements

The only other 1-port element that has not yet been discussed in terms of causality is the R element. The causal stroke on the resistive element implies neither an integral nor a differential relationship between the conjugate variables. The causal stroke implies a linear relationship between the conjugate variables. In the electrical domain, this linear relationship is simply Ohm's law written in terms of voltage or current depending on the position of the causal stroke. This is easily seen in figure 3.14.

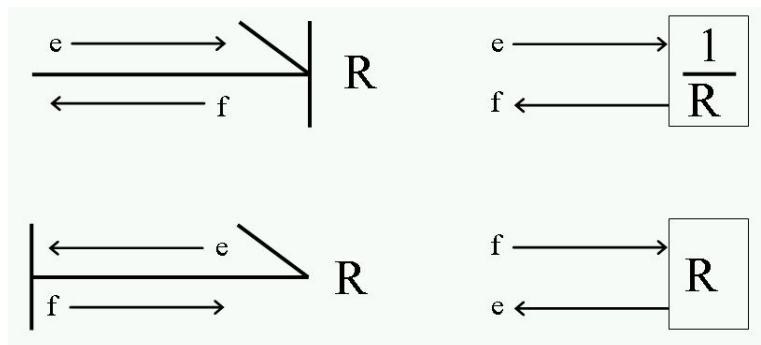


Figure 3.14. Possible Causal Assignments for a Resistive Element

It was shown in figure 3.2 that all bonds connected to a 0-junction have the same effort conjugate variable and all bonds connected to a 1-junction have the same flow conjugate variable. The causal assignments on the bonds of a 0-junction, and 1-junction, determine the source of the effort information, and flow information, respectively. This concept is clarified in figure 3.15. Figure 3.15 is the same as figure 3.2 with causality added to the bonds. Also, the effort equality statement for the 0-junction and the flow equality statement for the 1-junction have been rearranged slightly to emphasize which bond is responsible for the effort/flow information. Obviously, only one bond can be responsible for the source of effort/flow information. Thus, for a 0-junction only one bond can have a causal mark next to the 0. For a 1-junction only one bond can have a causal mark away from the 1. The bond that is the *odd man out* is the bond that defines the necessary information for the junction. For the purposes of figure 3.15 the defining bond on each junction has been picked arbitrarily. Typically in a bond graph the causal assignments around junctions will be obvious.

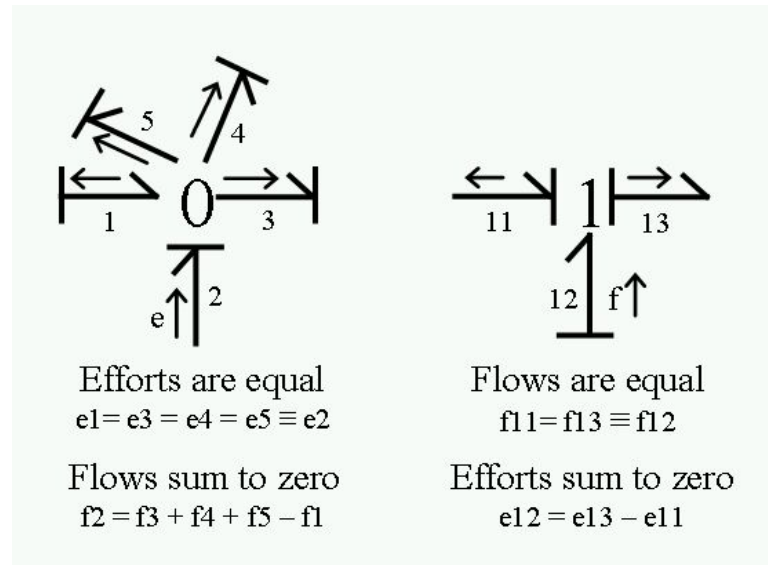


Figure 3.15. Causal Assignments on Bond Graph Junctions

The 0-junction in figure 3.15 shows that the effort terms are all equal, due to the fact that it is a 0-junction, and that they are all defined by bond 2. The effort signal information comes from bond 2 and is then spread to the rest of the bond graph by the other bonds, as is shown by the full arrows next to the bonds. The full arrows have been included in figure 3.15 for illustration only. Similarly, the 1-junction shows that all of the flows are equal, due to the fact that it is a 1-junction, and that all of the flows are defined by bond 12. The flow information comes from bond 12 and is then communicated to the rest of the bond graph via the remaining bonds on the 1-junction, as shown again by the full arrows in figure 3.15.

3.2.7 Bond Graph Causal Mark Assignments

Figure 3.8 shows a completed power flow diagram for the example circuit of figure 3.6. In order to complete the bond graph this diagram must have causality assignments. Causal assignments are made using the following steps;

1. Assign required causal marks to all sources. There is no choice on assigning a causal mark on a source, thus the causal marks are predetermined.
2. If step one provides the defining bond on a junction, then the causal marks for the bonds of the junction are also defined. Thus far, no choices have been made. Typically, step one will not determine the causal marks for entire junctions although this is not always the case.
3. If either step one or step two provides causal marks for any 2-port elements then the connecting causal marks are also defined. Steps two and three should be repeated until all bonds are assigned, where there is no choice on their assignments.
4. Choose an unassigned C or I element. Assign an integral causal mark. Repeat steps two and three. For the C and I elements, integral causality is preferred. Differential causality will only be used when steps two or three force the 1-port element to have differential causality.
5. Repeat step four for all C and I elements.
6. Potentially there may be unassigned R elements. If this is the case, choose an R element and assign an arbitrary causal mark. Repeat steps two and three as before.

7. Repeat step 6 until all bonds have a causal assignment. Every time there exists an unassigned R element whose causality is determined by an arbitrary assignment, there exists one algebraic loop in the resistor network. See Section 4.2.2 for a discussion on algebraic loops.

Figure 3.16 shows the complete bond graph for the circuit example of figure 3.6. The causal assignments were made in the following order;

1. Bond 1 was assigned, since this is the only source in the bond graph. This complies with step one above. Steps two and three are then skipped since no other causal assignments can be made at this point.
2. Bond 4 was assigned an integral causal mark for the C element. Upon assigning bond 4, bonds 3 and 5 are also assigned to comply with the causality rules for a 0-junction. In turn, the assignments for bonds 1 and 3 force the causal mark of bond 2 to follow the causal rules of a 1-junction. This complies with steps four and two above.
3. Next, bond 7 is assigned an integral causal mark which repeats step four for the I element. The last causal mark, on bond 6, is then forced to comply with the causal mark rules of a 1-junction.

Simply by looking at the bond graph of figure 3.16, it is seen that this system is a 2nd order system and no algebraic loops are present in the equations. The 2nd order system information is found by counting the number of integral causal 1-port elements. The algebraic loop information comes from the fact that there were no resistive elements without a causal mark upon finishing the integral causal assignments. Although this

information is obvious from the simple circuit diagram of figure 3.6, this information may not be so obvious for larger systems, especially ones that cross multiple engineering domains.

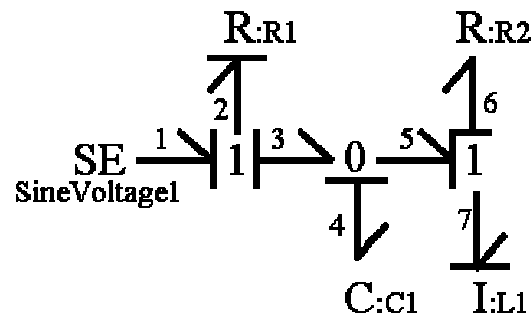


Figure 3.16. Completed Circuit Bond Graph

3.2.8 Bond Graph Equation Formulation

Bond graph equation formulation is a straightforward methodology that utilizes the signal flow information of the causal marks and the addition of efforts/flows on 1-junctions/0-junctions. Bond graphs use two types of generic state variables to express the dynamic equations. The first type is represented with the variable p and the second type is represented by a q . Table 3.2 shows each of these variables and their respective interpretations within various engineering domains [Bro01, Cel91, Kar83]. The selection of these state variables, as described by table 3.2, is done for a very specific purpose. The time derivative of the state variable p is an effort, in all domains, and the time

derivative of the state variable q is a flow. Naturally there is no chemical or thermodynamic momentum [Cel91]. Thus, these entries are left blank.

	p	q
Translational	Momentum [kg*m/s]	Displacement [m]
Rotational	Angular Momentum [kg*m ² /s]	Angular Displacement [rad]
Electrical	Flux Linkage [Wb = H*A]	Charge [C]
Hydraulic	Integral of Pressure [Pa*s]	Volume [m ³]
Chemical	————	Number of moles [n]
Thermodynamic	————	Entropy [S = J/K]

Table 3.2. Generic State Variable Definitions in Different Domains

The following steps are taken to create the state equations:

1. Select an I or C 1-port element with an integral causal mark.
2. Write the appropriate state equation beginning, $\dot{p} =$, or $\dot{q} =$ for an I or C element, respectively.

3. Use the conjugate variables as intermediate variables in developing the state equations. The equation is not complete if an intermediate e or f remains in the equation.
4. Use the signal flow of causality and summation properties in the appropriate locations.

These steps are best shown by means of an example. The circuit bond graph of figure 3.16 has one \dot{p} equation and one \dot{q} equation.

$$\dot{q}_4 = f_3 - f_5 \quad (3.1)$$

Starting with the C element of bond 4, equation 3.1 sums the flows around the 0-junction. The signal flow information of bond 3 leads to

$$f_3 = f_2 = \frac{e_2}{RI} \quad (3.2)$$

Subscripts denote bond numbers where non-subscript numbers denote circuit element values, i.e., LI is the inductance value found in the circuit diagram, which is the same as the bond graph value I_7 .

In order to solve for e_2 in equation 3.2 the efforts are summed around the 1-junction leading to

$$e_2 = e_1 - e_3 = SE_1 - e_4 = SE_1 - \frac{q_4}{C1} \quad (3.3)$$

Equation 3.3 results in an expression that involves no intermediate variables. However equation 3.1 still involves the intermediate variable f_5 .

$$f_5 = f_7 = \frac{p_7}{L1} \quad (3.4)$$

Substituting the results of equation 3.4, 3.3 into 3.2, and these results into 3.1 creates

$$\dot{q}_4 = \frac{1}{R1} \left[SE_1 - \frac{q_4}{C1} \right] - \frac{p_7}{L1} \quad (3.5)$$

Repeating the equation generation steps for the I element of bond 7 results in

$$\dot{p}_7 = e_5 - e_6 = e_4 - R2 * f_6 = \frac{q_4}{C1} - R2 * f_7 = \frac{q_4}{C1} - R2 * \frac{p_7}{L1} \quad (3.6)$$

Equation 3.6 was written in one continuous statement as shown because, after some practice, it is possible to write bond graph equations almost by inspection of the bond graph with a thought process similar to the one shown by equation 3.6. Rather than writing many small equations and substituting each time, it is usually possible to write the equation out as is shown by equation 3.6. Equation 3.6 written in a reduced form is shown by

$$\dot{p}_7 = \frac{q_4}{C1} - R2 * \frac{p_7}{L1} \quad (3.7)$$

The two state equations for the circuit diagram are then given by equations 3.5 and 3.7. Obviously these two equations are not in the common form for electrical circuits where voltage and current are state variables. Converting bond graph equations to common equations for each engineering domain is discussed in Section 3.2.9. The two equations form two, first-order, coupled, differential equations. For a linear system this set is easily converted into state-space form. This process is the case for any bond graph. The equations naturally form n coupled, first-order, differential equations.

3.2.9 Conversion of Bond Graph Variables to Common State Variables

The equations generated by bond graphs result in a set of non-standard state variable formulations that may be unintuitive to a user that is accustomed to common state variables. This situation was readily shown by the circuit example with equations 3.5 and 3.7. Obviously, these state variables can be converted into common state variables by using the proper transformation matrix.

Bond graph state variables can be converted into common state variables by using a diagonal transformation matrix. For every p in the electrical domain, the transformation matrix will have a $1/I$ on the corresponding diagonal. For every q in the electrical domain, the transformation matrix will have a $1/C$ on the corresponding diagonal. Similarly, in the mechanical domain a p state variable is transformed to a common mechanical state variable with a $1/I$ on the transformation matrix diagonal. The mechanical domain q , however, already defines a displacement and therefore has no need of transformation. Thus, a mechanical domain q will simply have a 1 on the corresponding transformation matrix diagonal.

Equation 3.8 combines equations 3.5 and 3.7 in state-space form. Again, subscripts denote bond numbers and non-subscript numbers denote circuit diagram values.

$$\begin{bmatrix} \dot{q}_4 \\ \dot{p}_7 \end{bmatrix} = \begin{bmatrix} -\frac{1}{C1 * R1} & -\frac{1}{L1} \\ \frac{1}{C1} & -\frac{R2}{L1} \end{bmatrix} \begin{bmatrix} q_4 \\ p_7 \end{bmatrix} + \begin{bmatrix} \frac{1}{R1} \\ 0 \end{bmatrix} SE_1 \quad (3.8)$$

These state variables are strictly electrical, thus the corresponding transformation matrix is given by

$$T = \begin{bmatrix} \frac{1}{C_4} & 0 \\ 0 & \frac{1}{I_7} \end{bmatrix} = \begin{bmatrix} \frac{1}{C1} & 0 \\ 0 & \frac{1}{L1} \end{bmatrix} \quad (3.9)$$

The resulting transformed state equations are shown as

$$\begin{bmatrix} \dot{q}_4 \\ \frac{q_4}{C1} \\ \dot{p}_7 \\ \frac{p_7}{L1} \end{bmatrix} = \begin{bmatrix} -\frac{1}{C1 * R1} & -\frac{1}{C1} \\ \frac{1}{L1} & -\frac{R2}{L1} \end{bmatrix} \begin{bmatrix} q_4 \\ \frac{q_4}{C1} \\ p_7 \\ \frac{p_7}{L1} \end{bmatrix} + \begin{bmatrix} \frac{1}{C1 * R1} \\ 0 \end{bmatrix} SE_1 \quad (3.10)$$

From the definitions of p and q for the electrical domain in table 3.2, the state variables in equation 3.10 can be rewritten as the voltage across the capacitor and the current through the inductor, respectively. This variable change is shown with

$$\begin{bmatrix} \dot{V}_{cap} \\ \frac{V_{cap}}{C1} \\ \dot{i}_{ind} \\ \frac{i_{ind}}{L1} \end{bmatrix} = \begin{bmatrix} -\frac{1}{C1 * R1} & -\frac{1}{C1} \\ \frac{1}{L1} & -\frac{R2}{L1} \end{bmatrix} \begin{bmatrix} V_{cap} \\ \frac{V_{cap}}{C1} \\ i_{ind} \\ \frac{i_{ind}}{L1} \end{bmatrix} + \begin{bmatrix} \frac{1}{C1 * R1} \\ 0 \end{bmatrix} Sinvoltage_1 \quad (3.11)$$

Equation 3.11 gives the same state-space representation that is given when classical methods are used to derive the circuit equations.

The example shown has been a simple second-order electrical circuit. This simple system was chosen to illustrate the basic technique of creating a bond graph and obtaining the dynamic equations. Obviously, it is not necessary to perform such complex manipulations to obtain the state-space representation for such a simple system when classical methods suffice. However, for a complex system that crosses multiple engineering domains, the bond graph technique gives a straight forward approach to

obtaining dynamic equations, and is therefore very useful. The state space equations obtained through the bond graph method suffice for all modeling purposes as shown in equation 3.8 with no need to transform them into any other state variable representation. However, the transformation shown in this section may be necessary when attempting to communicate the equations of motion from a bond graph to those unfamiliar with this modeling technique.

3.3 Bond Graph Construction from the Lagrangian

This section presents a method for developing a bond graph representation of a system from the Lagrangian of the system. Often the Lagrangian of a system is readily available from texts or other sources. Although the system equations can be derived directly from the Lagrangian, there is still benefit in viewing the system in bond graph representation. Some of these advantages are as follows:

1. Viewing the power flow through the system gives insight into the inter-relationships of the state variables. This insight may point out the possibility of simplifying assumptions. The bond graph often makes it clear what the assumptions need to be and their effects on the overall system.
2. Once a bond graph is obtained, whether from the Lagrangian or by conventional methods, it is a straight forward operation to connect the bond graph to larger systems.

3. By repeating the second advantage allows one to use bond graphs in an object-oriented fashion.

Conventionally bond graphs are designed to help the user obtain the system equations. However, for the reasons listed above, it is sometimes desirable to find the bond graph representation of a system itself.

Often systems that have complex mechanical geometries are difficult to model. The Lagrangian approach is often a preferred method of developing the system equations, since potential and kinetic energies are easier to account for than forces and moments for such systems. Since the Lagrangian is the sum of energies in the system, it is an integral away from the power flow in the system. This fact can be exploited to create a bond graph model of the system.

Although the Lagrangian method is a common method of developing the dynamic equations of a system, it turns out that bond graph theory is more closely related to the Hamiltonian of a system than the Lagrangian. The distinction between these two methods is described in the following section. Thus, during the bond graph development from the Lagrangian elements of the Hamiltonian formulation are used extensively. The starting point of the bond graph derivation is the Lagrangian, and not the Hamiltonian, for reasons that become apparent in the mathematical formalism of the Hamiltonian. These formalisms are explained below.

3.3.1 Lagrangian to Hamiltonian Transformation

The Lagrangian is defined as the sum of the kinetic energy of the system minus the sum of the potential energy of the system. For a conservative system, the Lagrangian is defined by [Mei98 pp. 68, Lag1788]

$$\mathcal{L} = T - V = 0 \quad (3.12)$$

Where T is the sum of all kinetic energies of the system and V is the sum of all potential energies of the system. The kinetic and potential energies of the system are written in terms of a set of generalized coordinates q_i . To obtain the dynamic equations of a system, given the Lagrangian, the Lagrange equation is applied separately for each of the generalized coordinates.

$$\frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = 0 \quad (3.13)$$

Thus, it is noted that the Lagrangian is a function of the generalized coordinates q_i , the generalized velocity \dot{q}_i , and time, i.e., $\mathcal{L}(q_i, \dot{q}_i, t)$. Also, equation 3.13 shows that for each generalized coordinate, the Lagrange method provides one, second-order equation.

The Hamiltonian is related to the Lagrangian via a transformation known as a *Legendre* transformation [Mei98 pp. 93, 342-343]. This transformation is derived in equations 3.14 through 3.21.

A function of two variables, $f(x,y)$, has a derivative, df , given by

$$df = \frac{\partial f}{\partial x} dx + \frac{\partial f}{\partial y} dy \quad (3.14)$$

Equation 3.14 can be written as

$$df \equiv udx + vdy \quad (3.15)$$

The Legendre transformation changes the variables from dx and dy to du and dv with the following transformation:

$$g \equiv f - ux \quad (3.16)$$

The derivative of equation 3.14, dg , is shown as

$$dg = df - udx - xdu \quad (3.17)$$

Substituting equation 3.15 into equation 3.17 yields

$$dg = udx + vdy - udx - xdu \quad (3.18)$$

Simplifying

$$dg = vdy - xdu \quad (3.19)$$

Equation 3.19 has the same form as equation 3.15, which, by definition of 3.15, leads to

$$x \equiv -\frac{\partial g}{\partial u} \quad (3.20)$$

and

$$v \equiv \frac{\partial g}{\partial y} \quad (3.21)$$

By applying the Legendre transformation to the Lagrangian of a system, the Hamiltonian is obtained. The Hamiltonian is defined as a function of the generalized coordinates, the generalized momentum, and time, i.e., $\mathcal{H}(q, p, t)$, without loss of generality the subscripts have been dropped. The Legendre transformation of the Lagrangian is shown by equations 3.22 through 3.32. The first step, as in the derivation of the Legendre transformation is to find $d\mathcal{L}$.

$$d\mathcal{L}(q, \dot{q}, t) = \frac{\partial \mathcal{L}}{\partial q} dq + \frac{\partial \mathcal{L}}{\partial \dot{q}} d\dot{q} + \frac{\partial \mathcal{L}}{\partial t} dt \quad (3.22)$$

Applying the Legendre transformation to the Lagrangian, in terms of the Hamiltonian, yields

$$\mathcal{H}(q, p, t) = p\dot{q} - \mathcal{L}(q, \dot{q}, t) \quad (3.23)$$

Applying the chain rule to equation 3.23, in order to find $d\mathcal{H}$, yields

$$d\mathcal{H} = p d\dot{q} + \dot{q} dp - d\mathcal{L} \quad (3.24)$$

Substituting equation 3.22 into equation 3.24 gives

$$d\mathcal{H} = p d\dot{q} + \dot{q} dp - \frac{\partial \mathcal{L}}{\partial q} dq - \frac{\partial \mathcal{L}}{\partial \dot{q}} d\dot{q} - \frac{\partial \mathcal{L}}{\partial t} dt \quad (3.25)$$

The generalized momentum is defined as

$$p \equiv \frac{\partial \mathcal{L}}{\partial \dot{q}} \quad (3.26)$$

Substituting equation 3.26 into equation 3.25 results in

$$d\mathcal{H} = \dot{q} dp - \frac{\partial \mathcal{L}}{\partial q} dq - \frac{\partial \mathcal{L}}{\partial t} dt \quad (3.27)$$

Also, substituting the definition of the generalized momentum, equation 3.26, into the Lagrange formulation of equation 3.13 yields

$$\frac{d}{dt} p - \frac{\partial \mathcal{L}}{\partial q} = 0 \quad (3.28)$$

Solving equation 3.28 for \dot{p} gives

$$\dot{p} = \frac{\partial \mathcal{L}}{\partial q} \quad (3.29)$$

Substituting equation 3.29 into 3.27 yields the final form of $d\mathcal{H}$.

$$d\mathcal{H} = \dot{q}dp - \dot{p}dq - \frac{\partial \mathcal{L}}{\partial t} dt \quad (3.30)$$

By noting the final steps of the Legendre transformation, equation 3.30 results in equations 3.31 through 3.33.

$$\dot{q} = \frac{\partial \mathcal{H}}{\partial p} \quad (3.31)$$

$$\dot{p} = -\frac{\partial \mathcal{H}}{\partial q} \quad (3.32)$$

$$\frac{\partial \mathcal{H}}{\partial t} = -\frac{\partial \mathcal{L}}{\partial t} \quad (3.33)$$

Equation 3.33 is a mathematical formalism and is of little consequence. Equations 3.31 and 3.32, however, form a meaningful result. For each generalized coordinate the Lagrange equation results in a single, second-order equation. The Hamiltonian method provides two, first-order, coupled equations for each generalized coordinate. Furthermore, the form of the equations given by the Hamiltonian formulation is similar to the form of equations that are derived by the bond graph method. The reason that the form of the Hamiltonian equations is not identical to the bond graph form is simply a question of the definition of the generalized momentum. The Hamiltonian formulation lumps mass terms together where the bond graph formulation does not. Thus, the bond graph method of equation development and the Hamiltonian are very closely related but do not form an exact one to one mapping.

An underlying assumption in the derivation of the Hamiltonian, given above, is that the system be conservative. Methods have been developed to lift this restriction for the Hamiltonian formulation but they are somewhat tedious [Tve98]. The Lagrange method for developing the equations of motion is not restricted to conservative systems only, but the non-conservative elements also add tedium to the process. The bond graph approach handles non-conservative systems quite readily. Furthermore, the bond graph approach allows the user to easily change the thermodynamic boundary of the system and track the energies of the non-conservative elements (dissipative elements) through to their transformation into heat and entropy, if so desired. The approach given here focuses on the creation of the bond graph from the Lagrangian. Also, there is ongoing research into the reverse approach, i.e., obtaining the Lagrangian for a given bond graph [Muk05, Muk97, Kar77]. This approach creates the Lagrangian, taking into account the non-conservative and external forces acting on the system.

3.3.2 Lagrangian to Bond Graph Development

The method for creating the bond graph from the Lagrangian follows these general steps:

1. Assume that the system is conservative. Drop the non-conservative elements from the Lagrangian if the system is non-conservative. The non-conservative elements will be added back in after the general structure of the bond graph has been created.

2. Note the flow terms in the Lagrangian. The kinetic energy terms in the Lagrangian will have the form $\frac{1}{2} I * f^2$ where I is an inertia term and f is a flow term.
3. Assign bond graph 1-junctions for each distinct flow term in the Lagrangian found in step 2.
4. Note the generalized momentum terms. The generalized momentum terms are noted by taking the partial of the Lagrangian with respect to the time derivative of the generalized coordinate, i.e., $p_i = \frac{\partial \mathcal{L}}{\partial \dot{q}_i}$, where p_i is the i_{th} generalized momentum and q_i is the i_{th} generalized coordinate. Note that this is the first step in developing the Hamiltonian as well.
5. For each generalized momentum equation, found in step 4, solve for \dot{q}_i . This step determines the form of the I -elements and how they connect to each of their corresponding 1-junctions. Often a generalized momentum will include a summation of many inertia elements. One of these inertial elements will have integral causality while the rest have derivative causality. Inertia elements that are scaled by some factor should be connected through a transformer, with the appropriate scale factor, to derivative causal I -elements.
6. Naturally, the equations derived from the Lagrangian show the balance of efforts around each 1-junction. Thus, the efforts on the 1-junctions are given by the equations of motion derived from the Lagrangian. Namely, $\frac{d}{dt} \frac{\partial \mathcal{L}}{\partial \dot{q}_i} - \frac{\partial \mathcal{L}}{\partial q_i} = 0$ gives the effort balance around the i_{th} 1-junction.

7. Develop the Hamiltonian for the conservative system by applying equation 3.23. Obviously this step may be impractical. For many systems the bond graph structure is apparent from steps 1 through 6. If the structure is not apparent, the Hamiltonian will add valuable insight due to the Hamiltonian formulation of n first-order equations.
8. Add non-conservative elements, where needed, on the bond graph structure.
9. Add external forces where needed as bond graph sources.
10. Use bond graph methods to simplify, if desired.

After completing steps 1-7, the structure of the bond graph will be apparent. The overall bond graph structure will have the general form shown in figure 3.17.

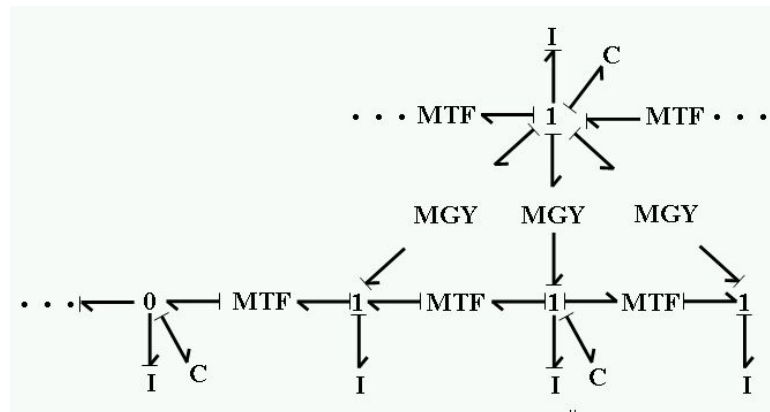


Figure 3.17. General Bond Graph Structure Developed from the Lagrangian

The process of creating a bond graph from the system Lagrangian is best shown by means of an example.

3.3.3 Lagrangian to Bond Graph Development: Pendulum Example, One Degree of Freedom

Figure 3.18 shows a pendulum with a single degree of freedom with mass m and mass moment of inertia μ . The length L represents the distance from the pivot point to the center of gravity.

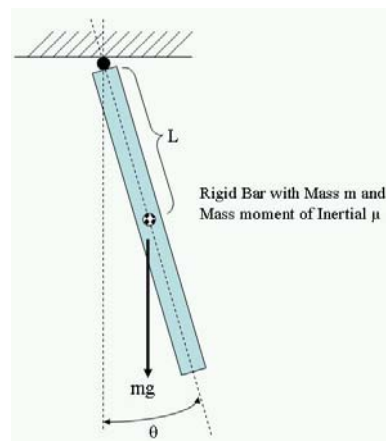


Figure 3.18. Single Degree of Freedom Pendulum

The Lagrangian is given by

$$\mathcal{L} = \frac{1}{2}m(L\dot{\theta})^2 + \frac{1}{2}\mu\dot{\theta}^2 - mg(L - L\cos(\theta)) = 0 \quad (3.34)$$

By noting the single degree of freedom θ , and by inspection of the Lagrangian, it is evident that the bond graph must have a single 1-junction to represent the flow term $\dot{\theta}$.

The partial of the Lagrangian with respect to $\dot{\theta}$ gives the generalized momentum shown by

$$p = \frac{\partial \mathcal{L}}{\partial \dot{\theta}} = mL^2 \dot{\theta} + \mu \dot{\theta} \quad (3.35)$$

Solving equation 3.35 for $\dot{\theta}$ yields

$$\dot{\theta} = \frac{p}{mL^2 + \mu} \quad (3.36)$$

Equation 3.36 shows that the 1-junction has two I -elements attached representing the mass moment of inertia, μ , and mass m . The L^2 term must then be modeled as a transformer.

The Hamiltonian is found by applying the Legendre transformation of equation 3.23:

$$\mathcal{H} = p * \dot{\theta} - \mathcal{L} = \frac{1}{2} m (L \dot{\theta})^2 + \frac{1}{2} \mu \dot{\theta}^2 + mgL + mgL \cos(\theta) \quad (3.37)$$

Equation 3.37 can be written as

$$\mathcal{H} = \frac{1}{2} \frac{p^2}{(mL^2 + \mu)} + mgL + mgL \cos(\theta) \quad (3.38)$$

Applying equations 3.31 and 3.32 provides the complete Hamiltonian equations. These are shown in equations 3.39 and 3.40, respectively. Note that equation 3.39 is a repeat of equation 3.36, as it should be, since the equation for $\dot{\theta}$ should be the same regardless of the method used to obtain it.

$$\dot{\theta} = \frac{\partial \mathcal{H}}{\partial p} = \frac{p}{mL^2 + \mu} \quad (3.39)$$

$$\dot{p} = -\frac{\partial \mathcal{H}}{\partial \theta} = mgL \sin(\theta) \quad (3.40)$$

Equation 3.39 shows the I -elements and how they connect to the 1-junction. Equation 3.40 shows that there is another bond on the 1-junction with an effort term equal to the left hand side of equation 3.40. Also, note that eliminating the variable p gives the same second-order equation that would have been found had the Lagrange method been used, i.e.,

$$(mL^2 + \mu)\ddot{\theta} = mgL \sin(\theta) \quad (3.41)$$

The 1-junction of figure 3.19 obviously must have three bonds attached to it, one for each of the terms in equation 3.41. These three effort terms sum around the 1-junction of figure 3.19. Also, in figure 3.18, the flow of the 1-junction is explicitly stated for clarification.

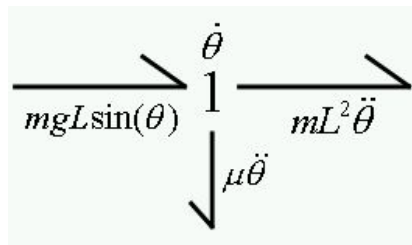


Figure 3.19. Pendulum 1-Junction

The power arrows shown in figure 3.19 reflect the signs of equation 3.41.

It is clear at this point that the remaining elements of the pendulum bond graph consist of two I -elements, two transformers and a source of effort. The complete bond graph is shown in figure 3.20. Again, the bonds have been arbitrarily numbered. The

bond graph shows a single degree of freedom on the integral-causal bond 3, and a differential causal bond 5. For this bond graph, the user might have chosen bond 5 to be integral-causal. This choice makes bond 3 differential-causal, and the transformer constant between bonds 4 and 5 would then be inverted to $1/L$.

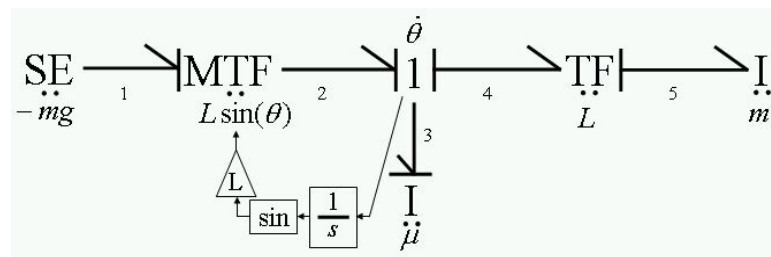


Figure 3.20. Complete Bond Graph of Pendulum

The effort source of figure 3.20 has been defined with a $-mg$. Often in bond graph representation sinks are modeled by assigning a positive term to the source element but showing the power-arrow towards the source.

The MTF in figure 3.20 represents a *modulated transformer*. This 2-port element is the same as the transformer element shown in Section 3.2.4, however, the transformer modulus is allowed to vary with time. A full arrow in bond graph terminology represents a pure signal. There is no power flow on the full arrows. Common block diagram algebra is used to describe the mathematics of the signal arrows. As seen in figure 3.20 the modulated transformer relates e_2 to e_1 by $e_2 = e_1 * L * \sin(\theta)$. The signal arrows begin at the 1-junction. Since all bonds connected to the 1-junction have the same flow, the flow value is the input to the signal arrow.

The equations for the bond graph of figure 3.20 are developed in equation 3.42-3.44.

$$\dot{p}_3 = SE_1 * L \sin(\theta) - L * e_5 \quad (3.42)$$

Equation 3.43 occurs in this form due to the differential causal assignment on bond 5.

$$e_5 = I_5 * \dot{f}_5 = I_5 L * \dot{f}_4 = I_5 L * \dot{f}_3 = I_5 L * \frac{\dot{p}_3}{I_3} \quad (3.43)$$

Plugging equation 3.43 into 3.42 and solving for \dot{p}_3 yields the final bond graph equation.

$$\dot{p}_3 \left(1 + \frac{I_5 L^2}{I_3} \right) = SE_1 * L \sin(\theta) \quad (3.44)$$

Equation 3.44 and 3.41 are shown to be equivalent by noting that $p_3 = \mu \dot{\theta}$, $I_5 = m$, $I_3 = \mu$, and $SE_1 = -mg$.

The single degree of freedom pendulum shows the method for creating the bond graph from the Lagrangian of a system but the example was simple enough that the bond graph could have been obtained by inspection of the free-body diagram. The following section develops the bond graph of a gyroscope using the approach shown here. The gyroscope model is a complicated system. Developing a bond graph for a gyroscope model without using a Lagrangian approach would be a much more difficult task.

3.3.4 Lagrangian to Bond Graph Development: Gyroscope Example

The bond graph formulation of a gyroscope demonstrates the usefulness of the Lagrangian/Hamiltonian approach. Creating a bond graph of a gyroscope without the aid

of a Lagrangian/Hamiltonian approach would be a difficult task due to the complexity of the system.

3.3.4.1 Gyroscope Lagrange Equations

Figure 3.21 shows a diagram of the gyroscope to be modeled. This model has two gimbals the mass of which will not be neglected in the development of the model.

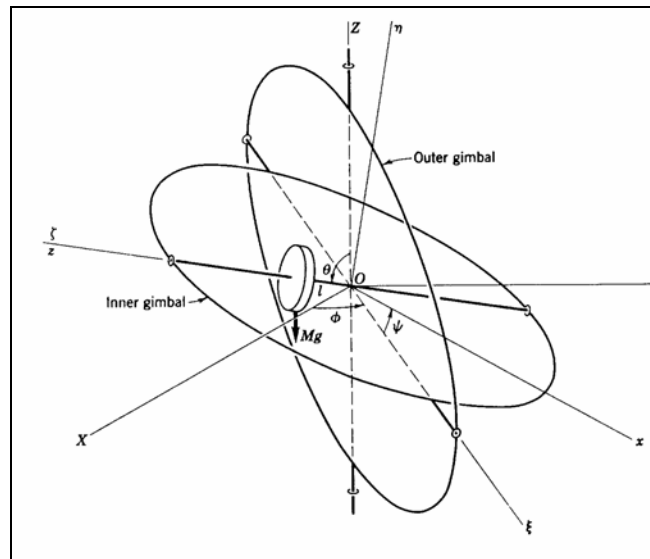


Figure 3.21. Gyroscope Diagram

The distance l , shown in figure 3.21, however, is set to zero since this is the most common implementation of a gyroscope model. This minor simplification causes the potential energy terms in the Lagrangian to disappear.

The Lagrangian of the gyroscope is given by equation 3.45, where θ , ϕ , and ψ , the three Euler angles, are the generalized coordinates of the system [Mei98 pp. 386-389].

$$L = T = \frac{1}{2} \left[(A + A')\dot{\theta}^2 + (A + B')\dot{\phi}^2 \sin^2 \theta + C(\dot{\phi} \cos \theta + \dot{\psi})^2 + C'\dot{\phi}^2 \cos^2 \theta + C''\dot{\phi}^2 \right] \quad (3.45)$$

The moment of inertia of the rotor about the symmetry axis ζ is denoted as C , and A is the moment of inertia of the rotor about any transverse axis through the point O . The moments of inertia of the inner gimbal about the axes ξ , η , and ζ , are denoted by A' , B' , and C' , respectively. The moment of inertia of the outer gimbal about the inertial axis Z is denoted by C'' . The corresponding Lagrange equations are given by equations 3.46 through 3.48.

$$\begin{aligned} N_{\phi} &= \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\phi}} \right) - \frac{\partial L}{\partial \phi} = \\ &= (A + B')\ddot{\phi} \sin^2 \theta + 2(A + B')\dot{\phi}\dot{\theta} \sin \theta \cos \theta + \\ &+ C(\ddot{\phi} \cos \theta - \dot{\phi}\dot{\theta} \sin \theta + \ddot{\psi}) \cos \theta + \\ &- C(\dot{\phi} \cos \theta + \dot{\psi})\dot{\theta} \sin \theta + C'\ddot{\phi} \cos^2 \theta - 2C'\dot{\phi}\dot{\theta} \sin \theta \cos \theta + C''\ddot{\phi}. \end{aligned} \quad (3.46)$$

$$N_{\psi} = \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\psi}} \right) - \frac{\partial L}{\partial \psi} = C(\ddot{\phi} \cos \theta - \dot{\phi}\dot{\theta} \sin \theta + \ddot{\psi}) \quad (3.47)$$

$$\begin{aligned} N_{\theta} &= \frac{d}{dt} \left(\frac{\partial L}{\partial \dot{\theta}} \right) - \frac{\partial L}{\partial \theta} = \\ &= (A + A')\ddot{\theta} + C(\dot{\phi} \cos \theta + \dot{\psi})\dot{\phi} \sin \theta - (A + B' - C')\dot{\phi}^2 \sin \theta \cos \theta \end{aligned} \quad (3.48)$$

The generalized torques are given by N_{ϕ} , N_{ψ} , and N_{θ} . The Lagrangian equations are three, second-order, coupled differential equations resulting in a sixth-order system. The state variables of this system are θ , $\dot{\theta}$, ϕ , $\dot{\phi}$, ψ , and $\dot{\psi}$. However, the state variables ϕ , and ψ do not show up in the above equations. Thus, the system can be described entirely by four state equations. The four state equations consist of equations 3.46

through 3.48 and the simple equation $\dot{\theta} = \frac{d}{dt}(\theta)$. The resulting system is a fourth-order, nonlinear system.

3.3.4.2 Gyroscope Bond Graph

This section builds the bond graph from the Lagrangian of the gyroscope [McB01]. The step by step bond graph creation process is as follows:

1. Step one of the bond graph creation process is not needed since this system is already a conservative system.
2. In order to note the flow terms of the Lagrangian for this conservative system it is necessary to rewrite the Lagrangian. The Lagrangian can be rewritten such that each flow term has the form $\frac{1}{2} I * f^2$. Equation 3.45 becomes equation 3.49. Equation 3.49 shows that the bond graph will have three 1-junctions with integral causality. The three 1-junctions represent the flow terms $\dot{\theta}$, $\dot{\phi}$, and $\dot{\psi}$.

$$L = \frac{1}{2}[(A + B')\sin^2 \theta + (C + C')\cos^2 \theta + C'']\dot{\phi}^2 + \frac{1}{2}(A + A')\dot{\theta}^2 + \frac{1}{2}C\dot{\psi}^2 + C\dot{\phi}\dot{\psi}\cos\theta \quad (3.49)$$

3. The 1-junction assignment is shown in figure 3.22.

$$\mathbf{1}$$

$$\dot{\theta}$$

$$\mathbf{1}$$

$$\dot{\psi}$$

$$\mathbf{1}$$

$$\dot{\phi}$$

Figure 3.22. Gyroscope Integral 1-Junctions

4. The i^{th} generalized momentum is found by $p_i = \frac{\partial \mathcal{L}}{\partial \dot{q}_i}$. The generalized

momentum equations are

$$p_\phi = \frac{\partial L}{\partial \dot{\phi}} = [(A + B') \sin^2 \theta + (C + C') \cos^2 \theta + C''] \dot{\phi} + C \dot{\psi} \cos \theta \quad (3.50)$$

$$p_\theta = \frac{\partial L}{\partial \dot{\theta}} = (A + A') \dot{\theta} \quad (3.51)$$

$$p_\psi = \frac{\partial L}{\partial \dot{\psi}} = C \dot{\psi} + C \dot{\phi} \cos \theta \quad (3.52)$$

5. Solving equations 3.50 through 3.52 for the respective \dot{q}_i terms yields equations 3.53 through 3.55. Equation 3.53 has a sum of inertia elements in the denominator. This sum contains sine and cosine terms that are connected by transformers to the $\dot{\phi}$ 1-junction. Equation 3.53, and 3.55 both have sums in the numerators. This indicates that these two flows are connected via a 0-junction,

since flows sum around a 0-junction. Also, the sum of C and C' cannot be on the same I -element as are A and B' due to the zero junction flow addition. Equation 3.54 implies that the θ 1-junction is a standard junction with a single I -element. Figure 3.23 reflects these updates to the bond graph.

$$\dot{\phi} = \frac{P_\phi - C\dot{\psi} \cos \theta}{[(A + B')\sin^2 \theta + (C + C')\cos^2 \theta + C'']} \quad (3.53)$$

$$\dot{\theta} = \frac{P_\theta}{(A + A')} \quad (3.54)$$

$$\dot{\psi} = \frac{P_\psi - C\dot{\phi} \cos \theta}{C} \quad (3.55)$$

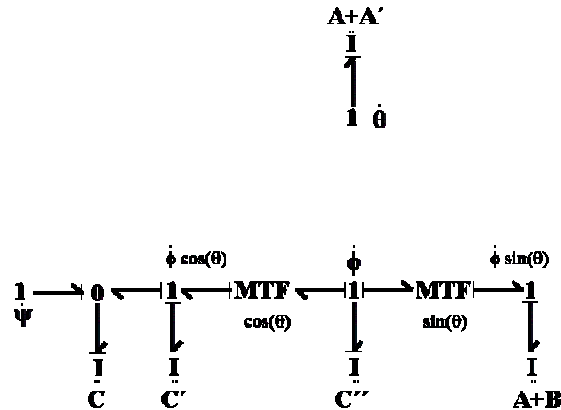


Figure 3.23. Gyroscope 1-Junctions: I -Element Connections

6. Taking the time derivative of equations 3.50 through 3.52 yields equations 3.56 through 3.58. This is a step in developing the Lagrange equations for the

gyroscope with the Lagrangian in the form of equation 3.49. This step also shows \dot{P}_i equations as a function of \dot{q}_j and q_j .

$$\begin{aligned} \dot{p}_\phi = \frac{d}{dt} \frac{\partial L}{\partial \dot{\phi}} = & [(A + B') \sin^2 \theta + (C + C') \cos^2 \theta + C''] \ddot{\phi} + \\ & + 2(A + B') \dot{\phi} \dot{\theta} \sin \theta \cos \theta - 2(C + C') \dot{\phi} \dot{\theta} \sin \theta \cos \theta + \\ & + C \ddot{\psi} \cos \theta - C \dot{\psi} \dot{\theta} \sin \theta \end{aligned} \quad (3.56)$$

$$\dot{p}_\theta = \frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}} = (A + A') \ddot{\theta} \quad (3.57)$$

$$\dot{p}_\psi = \frac{d}{dt} \frac{\partial L}{\partial \dot{\psi}} = C \ddot{\psi} + C \ddot{\phi} \cos \theta - C \dot{\phi} \dot{\theta} \sin \theta \quad (3.58)$$

The last step in completing the Lagrange equations for the Lagrangian of 3.49 is shown in equations 3.59 through 3.61.

$$\frac{\partial L}{\partial \phi} = 0 \quad (3.59)$$

$$\frac{\partial L}{\partial \theta} = (A + B') \dot{\phi}^2 \sin \theta \cos \theta - (C + C') \dot{\phi}^2 \sin \theta \cos \theta - C \dot{\psi} \dot{\phi} \sin \theta \quad (3.60)$$

$$\frac{\partial L}{\partial \psi} = 0 \quad (3.61)$$

Two of the three complete Lagrange equations are obtained by setting equations 3.56, and 3.58 equal to zero. The third complete Lagrange equation is obtained by subtracting equation 3.57 from equation 3.60 to obtain

$$(A + A')\ddot{\theta} - (A + B')\dot{\phi}^2 \sin \theta \cos \theta + (C + C')\dot{\phi}^2 \sin \theta \cos \theta + C\dot{\psi}\dot{\phi} \sin \theta = 0 \quad (3.62)$$

Figure 3.23 is beginning to show the general bond graph form shown in figure 3.17. The modulated gyrator connections can be seen by noting the effort terms of equation 3.62. Recall that a gyrator provides an effort term by multiplying a flow term with the gyrator modulus. Three of the effort terms of equation 3.62 are rewritten in equations 3.63 and 3.65 to reflect a flow term multiplied by a gyrator modulus.

$$e = [(A + B')\dot{\phi} \sin \theta] \dot{\phi} \cos \theta = Mf \quad (3.63)$$

$$e = [(C + C')\dot{\phi} \cos \theta] \dot{\phi} \sin \theta = Mf \quad (3.64)$$

$$e = [C\dot{\psi}] \dot{\phi} \sin \theta = Mf \quad (3.65)$$

Equations 3.64 and 3.65 can be combined to group the C terms together. This is desired since the C inertia appears only once in the bond graph of figure 3.23.

$$e = [C\dot{\psi} + C\dot{\phi} \cos \theta] \dot{\phi} \sin \theta = Mf \quad (3.66)$$

$$e = [C'\dot{\phi} \cos \theta] \dot{\phi} \sin \theta = Mf \quad (3.67)$$

The flow terms of equations 3.63, 3.66 and 3.67 are already established in figure 3.23. The modulus terms are momentum terms. These terms are also established in figure 3.23 as the momentum terms of the derivative causal I -elements, and the integral causal I -

element with inertia C . Thus, the bond graph of figure 3.23 becomes the bond graph of figure 3.24. Note that the 0-junction summation is also reflected in equation 3.66.

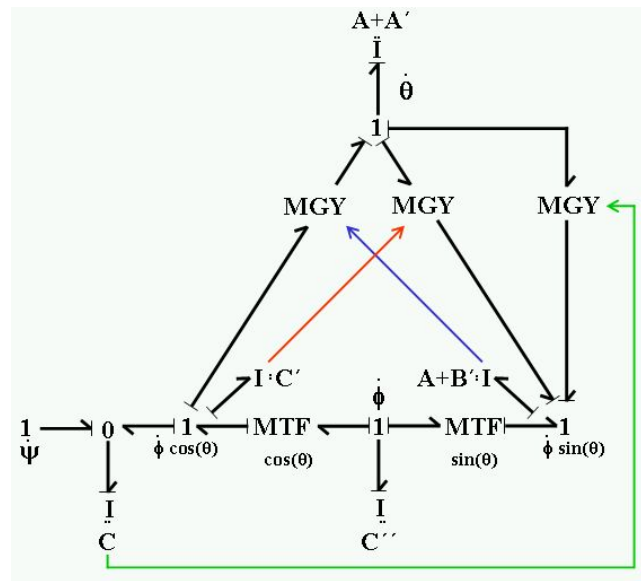


Figure 3.24. Gyroscope MGY Connections

The gyroscope bond graph of figure 3.24 is complete with the exception of the effort sources. The signal arrows indicate that the *MGY* elements use momentum signals as their moduli. The elegance of the bond graph construction method presented here is that much of the complicated mathematics of the Lagrange equations is provided within a relatively simple structure. The differential causal elements account for much of the Lagrange equation complexity. The cross-coupled flow terms of equations 3.56 and 3.62 are accounted for through the differential causal elements connected through modulated gyrators and transformers. The transformer modulations themselves are natural elements

of the bond graph, i.e., they are not created by over-complicated gyrator and transformer moduli.

Input torque sources are added to the Lagrange equations 3.56, 3.58 and 3.62 to obtain equations 3.68 through 3.70. These equations are identical to those obtained in equations 3.46 through 3.48, with the sources in bond graph notation.

$$SE_{\phi} = [(A + B')\sin^2 \theta + (C + C')\cos^2 \theta + C'']\ddot{\phi} + 2(A + B')\dot{\phi}\dot{\theta} \sin \theta \cos \theta - 2(C + C')\dot{\phi}\dot{\theta} \sin \theta \cos \theta + C\ddot{\psi} \cos \theta - C\dot{\psi}\dot{\theta} \sin \theta \quad (3.68)$$

$$SE_{\psi} = C\ddot{\psi} + C\ddot{\phi} \cos \theta - C\dot{\phi}\dot{\theta} \sin \theta \quad (3.69)$$

$$SE_{\theta} = (A + A')\ddot{\theta} - (A + B')\dot{\phi}^2 \sin \theta \cos \theta + (C + C')\dot{\phi}^2 \sin \theta \cos \theta + C\dot{\psi}\dot{\phi} \sin \theta \quad (3.70)$$

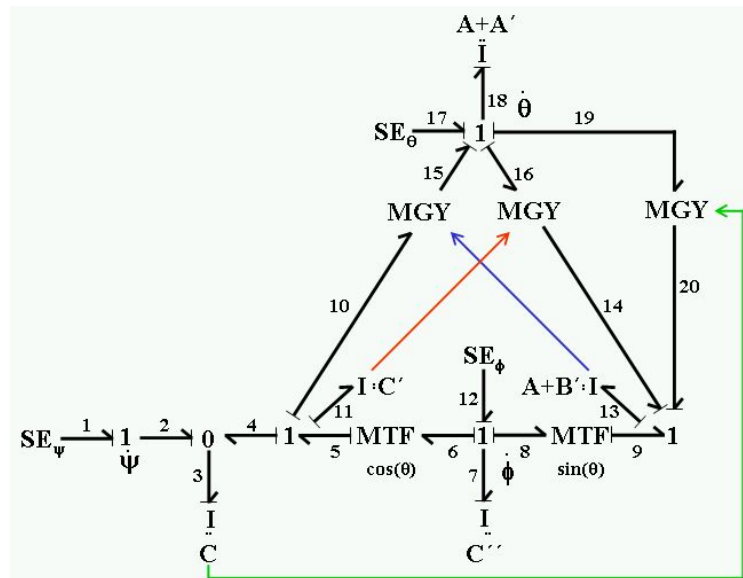


Figure 3.25. Complete Gyroscope Bond Graph

Figure 3.25 shows the effort sources added to the gyroscope bond graph. The bonds have been assigned arbitrary numbers to aid in the bond graph equation derivation. Note, bonds 1 and 2 can be collapsed into a single bond but have been left as shown to specifically indicate the ψ 1-junction.

3.3.4.3 Gyroscope Bond Graph Equations

The bond graph equations for figure 3.25 are best derived by first defining a few of the variables that will appear often in the equation derivation. Obviously the flows of the derivative causal elements will be convenient terms to have predefined. Also, the momentums used to modulate the gyrators will be used often. These predefinitions are shown as

$$f_5 = \dot{\phi} \cos \theta \quad (3.71)$$

$$\dot{f}_5 = \ddot{\phi} \cos \theta - \dot{\phi} \dot{\theta} \sin \theta \quad (3.72)$$

$$f_9 = \dot{\phi} \sin \theta \quad (3.73)$$

$$\dot{f}_9 = \ddot{\phi} \sin \theta + \dot{\phi} \dot{\theta} \cos \theta \quad (3.74)$$

$$P_3 = Cf_3 = C(f_2 + f_5) = C(\dot{\psi} + \dot{\phi} \cos \theta) \quad (3.75)$$

$$P_{11} = C'f_{11} = C'f_5 = C'\dot{\phi} \cos \theta \quad (3.76)$$

$$P_{13} = (A + B')f_{13} = (A + B')f_9 = (A + B')\dot{\phi} \sin \theta \quad (3.77)$$

$$P_{18} = (A + A')f_{18} = (A + A')\dot{\theta} \quad (3.78)$$

$$P_7 = C''f_7 = C''\dot{\phi} \quad (3.79)$$

The simplest bond graph equation is the integral causal I -element on bond 3. This is because the effort on the 0-junction is defined entirely by the effort source on bond 1. The result yields

$$\dot{P}_3 = e_3 = e_2 = e_1 = SE_\psi \quad (3.80)$$

Naturally, equation 3.75 can be used to write equation 3.80 in terms of the Lagrangian variables. Equation 3.81 is identical to equation 3.47 and equation 3.69.

$$\dot{P}_3 = C(\ddot{\psi} + \ddot{\phi} \cos \theta - \dot{\phi} \dot{\theta} \sin \theta) = SE_\psi \quad (3.81)$$

The next equation is taken by summing the efforts around the integral causal I -element on bond 18.

$$\dot{P}_{18} = e_{17} + e_{15} - e_{16} - e_{19} = SE_\theta + P_{13}f_{10} - P_{11}f_{14} - P_3f_{20} \quad (3.82)$$

$$\dot{P}_{18} = SE_\theta + P_{13}f_5 - P_{11}f_9 - P_3f_9 \quad (3.83)$$

Using the predefinitions, equation 3.83 becomes

$$\dot{P}_{18} = SE_\theta + (A + B')\dot{\phi}^2 \sin \theta \cos \theta - C'\dot{\phi}^2 \sin \theta \cos \theta - (C\dot{\psi} + C\dot{\phi} \cos \theta)\dot{\phi} \sin \theta \quad (3.84)$$

Simplifying

$$\dot{P}_{18} = SE_\theta + (A + B' - C - C')\dot{\phi}^2 \sin \theta \cos \theta - C\dot{\psi}\dot{\phi} \sin \theta \quad (3.85)$$

Using equation 3.78 to find \dot{P}_{18} , equation 3.85 can be written

$$(A + A')\ddot{\theta} + (-A - B' + C + C')\dot{\phi}^2 \sin \theta \cos \theta + C\dot{\psi}\dot{\phi} \sin \theta = SE_\theta \quad (3.86)$$

Equation 3.86 is identical to equations 3.48 and 3.70.

The last equation is the most complex of the three bond graph equations. This equation relies on both derivative causal elements, both modulated transformers and all three modulated gyrators.

$$\dot{P}_7 = e_{12} - e_6 - e_8 = SE_\phi - e_5 \cos \theta - e_9 \sin \theta = SE_\phi - (e_5 \cos \theta + e_9 \sin \theta) \quad (3.87)$$

At this point it is much easier to work with e_5 and e_9 separately and substitute the end result back into equation 3.87.

$$e_5 = e_{10} + e_{11} + e_4 = P_{13}f_{15} + C'f_{11} + SE_\psi = P_{13}\dot{\theta} + C'f_5 + SE_\psi \quad (3.88)$$

Using predefined equations 3.72 and 3.77, equation 3.88 becomes:

$$e_5 = (A + B')\dot{\phi}\dot{\theta} \sin \theta + C'\ddot{\phi} \cos \theta - C'\dot{\phi}\dot{\theta} \sin \theta + SE_\psi \quad (3.89)$$

Multiplying both sides of equation 3.89 by $\cos \theta$ and substituting equation 3.81 for SE_ψ , equation 3.89 yields

$$\begin{aligned} e_5 \cos \theta &= (A + B')\dot{\phi}\dot{\theta} \sin \theta \cos \theta + C'\ddot{\phi} \cos^2 \theta + \\ &- C'\dot{\phi}\dot{\theta} \sin \theta \cos \theta + (C\ddot{\psi} + C\ddot{\phi} \cos \theta - C\dot{\phi}\dot{\theta} \sin \theta) \cos \theta \end{aligned} \quad (3.90)$$

Simplifying equation 3.90 gives

$$e_5 \cos \theta = (A + B' - C - C')\dot{\phi}\dot{\theta} \sin \theta \cos \theta + (C + C')\ddot{\phi} \cos^2 \theta + C\ddot{\psi} \cos \theta \quad (3.91)$$

Similarly for e_9

$$e_9 = e_{13} - e_{14} - e_{20} = (A + B')f_{13} - P_{11}f_{16} - P_3f_{19} = (A + B')f_9 - P_{11}\dot{\theta} - P_3\dot{\theta} \quad (3.92)$$

Using predefined equation 3.74, 3.75, and 3.76, e_9 becomes

$$\begin{aligned} e_9 &= (A + B')(\ddot{\phi} \sin \theta + \dot{\phi}\dot{\theta} \cos \theta) - C'\dot{\phi}\dot{\theta} \cos \theta - C(\ddot{\psi} + \dot{\phi} \cos \theta)\dot{\theta} = \\ &= (A + B')\ddot{\phi} \sin \theta + (A + B')\dot{\phi}\dot{\theta} \cos \theta - C'\dot{\phi}\dot{\theta} \cos \theta - C\ddot{\psi}\dot{\theta} - C\dot{\phi}\dot{\theta} \cos \theta \end{aligned} \quad (3.93)$$

Multiplying both sides of equation 3.93 by $\sin \theta$

$$\begin{aligned}
e_9 \sin \theta = & (A + B')\ddot{\phi} \sin^2 \theta + (A + B')\dot{\phi}\dot{\theta} \sin \theta \cos \theta + \\
& - C'\dot{\phi}\dot{\theta} \sin \theta \cos \theta - C\dot{\psi}\dot{\theta} \sin \theta - C\dot{\phi}\dot{\theta} \sin \theta \cos \theta
\end{aligned} \tag{3.94}$$

Equation 3.95 is a rearrangement of the terms in equation 3.94 in a similar form as equation 3.91.

$$e_9 \sin \theta = (A + B' - C - C')\dot{\phi}\dot{\theta} \sin \theta \cos \theta + (A + B')\ddot{\phi} \sin^2 \theta - C\dot{\psi}\dot{\theta} \sin \theta \tag{3.95}$$

Adding equations 3.95 and 3.91 yields

$$\begin{aligned}
e_5 \cos \theta + e_9 \sin \theta = & 2(A + B' - C - C')\dot{\phi}\dot{\theta} \sin \theta \cos \theta + \\
& + \ddot{\phi}[(A + B')\sin^2 \theta + (C + C')\cos^2 \theta] + \\
& + C\ddot{\psi} \cos \theta - C\dot{\psi}\dot{\theta} \sin \theta
\end{aligned} \tag{3.96}$$

Substituting equation 3.96 into 3.87 and using predefined equation 3.79 for \dot{P}_7 yields

$$\begin{aligned}
C''\ddot{\phi} = & SE_\phi - 2(A + B' - C - C')\dot{\phi}\dot{\theta} \sin \theta \cos \theta + \\
& - \ddot{\phi}[(A + B')\sin^2 \theta + (C + C')\cos^2 \theta] + \\
& - C\ddot{\psi} \cos \theta + C\dot{\psi}\dot{\theta} \sin \theta
\end{aligned} \tag{3.97}$$

Solving equation 3.97 for SE_ϕ completes the bond graph equation derivation.

$$\begin{aligned}
SE_\phi = & \ddot{\phi}[(A + B')\sin^2 \theta + (C + C')\cos^2 \theta + C''] + \\
& 2(A + B' - C - C')\dot{\phi}\dot{\theta} \sin \theta \cos \theta + \\
& + C\ddot{\psi} \cos \theta - C\dot{\psi}\dot{\theta} \sin \theta
\end{aligned} \tag{3.98}$$

Equation 3.98 is identical to equations 3.46 and 3.68. Equations 3.81, 3.86 and 3.98 form the bond graph equations for the gyroscope model in figure 3.25. The simple equation

$\dot{\theta} = \frac{d}{dt}(\theta)$ must be added to the set to complete the state equations of the model.

3.4 Brown's Lagrangian Bond Graphs

An alternate method for developing a bond graph model from the Lagrangian of a system was shown by F. Brown [Bro72]. The general method produces bond graphs with a fairly simple structure. However, in some cases the transformer moduli are overly complex. An example system, used by Brown is a ball joint table assembly, shown in figure 3.26.

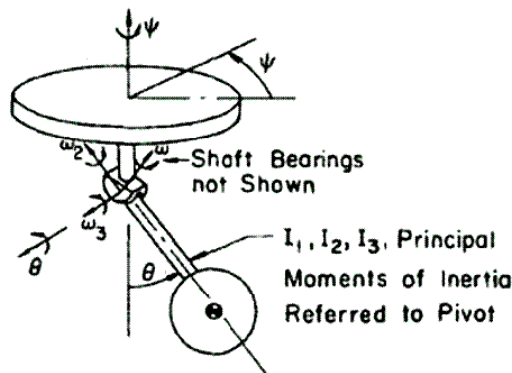


Figure 3.26. Ball Joint Table: Schematic

The Lagrangian for this system is

$$L = \frac{1}{2} I_1 \dot{\psi}^2 \sin^2 \theta + \frac{1}{2} I_2 \dot{\psi}^2 \cos^2 \theta + \frac{1}{2} I \dot{\psi}^2 + \frac{1}{2} I_3 \dot{\theta}^2 + mgl \cos \theta \quad (3.99)$$

The definitions of ω_1 , ω_2 , and ω_3 in figure 3.26 are as follows:

$$\omega_1 = \dot{\psi} \sin \theta \quad (3.100)$$

$$\omega_2 = \dot{\psi} \cos \theta \quad (3.101)$$

$$\omega_3 = \dot{\theta} \tag{3.102}$$

The method described by Brown groups the inertia terms of the Lagrangian together into a symbolic energy term.

$$E = \frac{1}{2} I' (T \dot{\psi})^2 \tag{3.103}$$

Where I' is arbitrarily set at 1 and T is

$$T = \sqrt{I + I_2 - (I_2 - I_1) \sin^2 \theta} \tag{3.104}$$

The resulting bond graph by Brown is shown in figure 3.27.

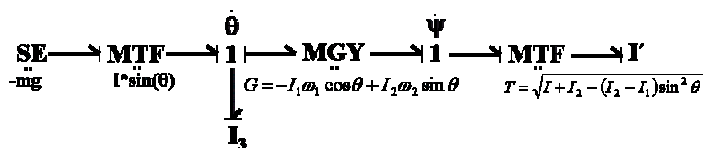


Figure 3.27. Ball Joint Table: Brown's Bond Graph

As seen in figure 3.27 Brown's bond graph contains a very complex transformer structure with a fictitious I -element.

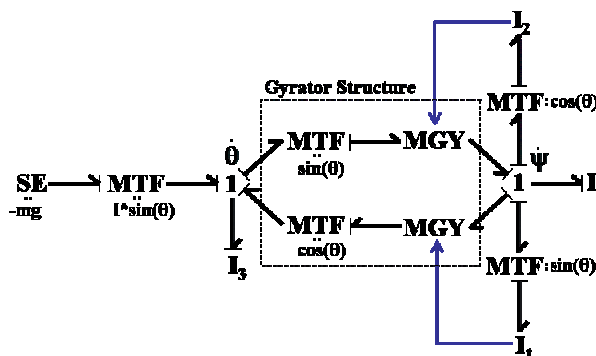


Figure 3.28. Ball Joint Table: Bond Graph from Current Method

The method presented in Section 3.3 produces a similar bond graph structure but recognizes that the gyrator modulus contains momentum elements. The bond graph is drawn in figure 3.28 with this detail called out specifically. Naturally, either bond graph may be converted to the other through inspection. For example, it can easily be seen that the gyrator structure of figure 3.28 gives the same equation of the gyrator in figure 3.27. This is due to the fact that they are both correct representations of the same system. The methods for creating the bond graph from the Lagrangian, however, are different. The method presented here does not need overly complicated transformer and gyrator moduli. Each transformer/gyrator modulus is very simple. Also, the method presented here specifically shows that the gyrator moduli are momentums of differential causal elements of the bond graph, which is insightful when trying to understand the interrelationships of the state variables.

3.5 Conclusions

This chapter introduced bond graph modeling. Bond graph modeling was presented as a method of system representation that maps the power flow through the system. Since the bond graph deals with power flow, the modeling method can be used with equal efficiency in all energy domains. Consequently, this modeling method is very useful when modeling systems that cross multiple energy domains.

Also, the bond graph maps the flow of information through a system by assigning causality to the modeling elements. The causal mapping is done in a consistent and meaningful manner that gives insight to the model. The causal and power flow information are used together to determine the state equations of the system.

This chapter also presented a method for converting the Lagrangian into a bond graph model. Lagrangian and Hamiltonian elements of the system were used to create the bond graph model. A pendulum and gyroscope were used as examples of bond graph creation, given the Lagrangian.

The method presented here was compared to a method presented previously by F. Brown. The method presented here shows that, for rigid body systems, gyrators are often modulated via a momentum signal from another part of the bond graph. This insight is unclear when using Brown's method. Also in Brown's method, although the structure of the bond graph is very simple and compact, the transformer moduli contain complicated functions of the inertias of the system.

CHAPTER 4: Object-Oriented Bond Graph Modeling

4.1 Introduction

As seen in Chapter 3, bond graph modeling is a useful tool for generating the equations of motion of a mechanical system from the power flow map of a physical system. However, once the equations of motion have been generated it is still necessary to develop computer code in order to perform simulation and analysis of the system. Ideally, the computer code would be generated directly from the bond graph model of a system with the simulation software.

Modelica [Dym] is a modeling framework used for simulation of complex physical systems. Within the Modelica framework, models of systems can be included as sub-models within larger systems. The code for the overall system is then generated automatically, thus allowing the user to use the sub-models in an object-oriented, plug and play manner. *Dymola* [Dym, Brü02] is a software package that enables the user to use the Modelica framework with a graphical interface.

Dymola/Modelica, however, have no knowledge of bond graph modeling. For this reason a bond graph library was developed to combine the object-oriented abilities of Dymola with the power based system representation of bond graph modeling [Cel03a].

4.2 Dymola

The Dymola framework allows the user to create system models such that they can be used in the next upper hierarchical level. Dymola uses four windows to define each model: an *equation window*, a *diagram window*, an *icon window*, and a *documentation window*.

The lowest level of a model/sub-model resides in the equation window, i.e., when all sub-models are expanded to the lowest hierarchical level, the code will reside in the equation window. The equation window contains code written in the Modelica language [Brü02]. Modelica is a fairly straightforward language to understand and can be read by anyone possessing basic programming knowledge in FORTRAN or C. The equation window of a bond graph power sensing bond is shown in figure 4.1.

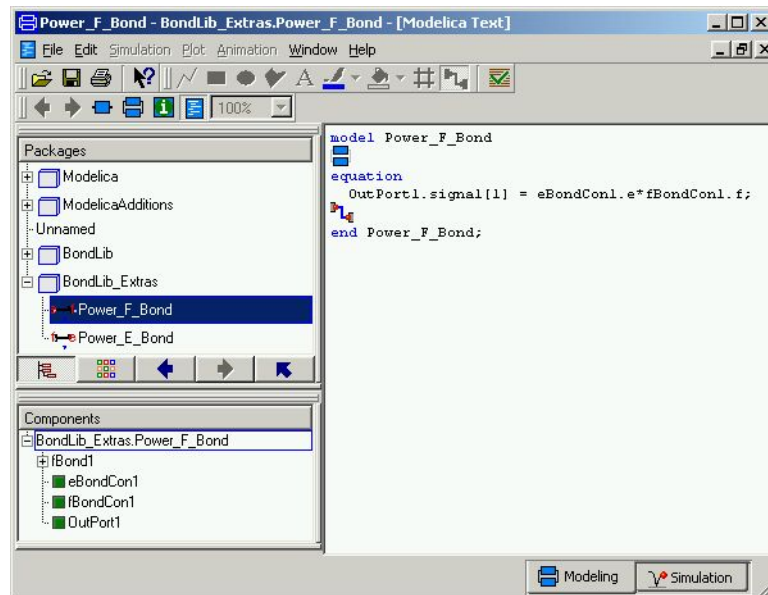


Figure 4.1. Power Sensing Bond Model: Equation Window

The power sensing bond of figure 4.1 is dependent on other bond graph models as can be seen by the diagram window of the same model. This window is shown in figure 4.2. The diagram window allows the user to include sub-models and connect them in a block diagram fashion. In figure 4.2 the sub-model, *fbond1*, is included in an object-oriented fashion. *fbond1* is connected to two output ports labeled *e* and *f*, such that the model can later be dropped into larger models. The output signal that is output at the bottom of figure 4.2 is defined in the equation window, shown in figure 4.1.

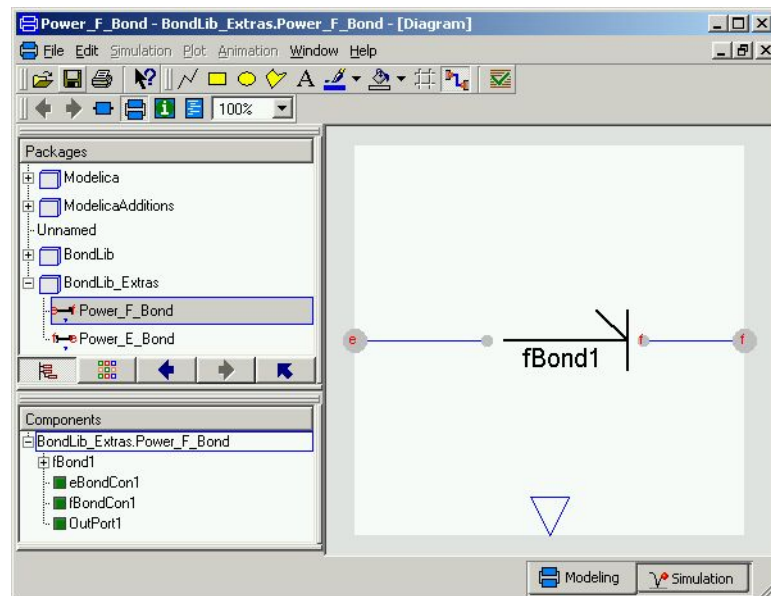


Figure 4.2. Power Sensing Bond Model: Diagram Window

The icon window of the power sensing bond model is shown in figure 4.3. The icon window contains a graphical description of how the model will be represented at the next hierarchical level. The text *name*, shown in figure 4.3, leaves a generic text placeholder

such that the user can assign a distinct name at the next hierarchical level. This is necessary in case multiple instances of the same model are used at the upper level.

The *Power_F_Bond* model was used here to show the workings of the three primary Dymola windows. The internal details of the *Power_F_Bond* model are discussed in Section 4.3.

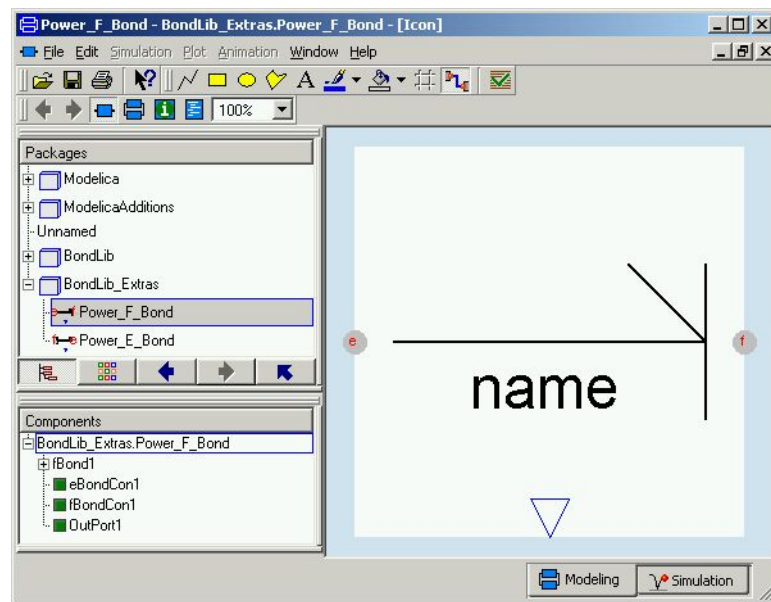


Figure 4.3. Power Sensing Bond Model: Icon Window

4.2.1 Equation Sorting

In a laboratory setting, a motor may be driven forward such that an input voltage drives the motor to observe the angular velocity of the motor shaft as the output. Also, it is possible to hook the motor up backwards and input a torque to the motor shaft and observe the generated current as the output. Similarly, Dymola allows the user to hook up models in different configurations. Dymola must then be able to sort the equations

such that they can be converted into assignment statements at compilation time. For example, it is perfectly acceptable to write in the equation window of a Dymola model the following

$$A + B + C = D + E + F \quad (4.1)$$

At the time the model is compiled Dymola must sort through all equations to create assignment statements for all variables. It is possible that in one instance of a model containing equation 4.1, the equation is solved for A . And, in another instance, it is solved for E , depending on the manner in which the model is connected at the upper hierarchical level. This is a powerful advantage that Dymola has over other modeling software since the object-oriented capabilities allow the user to plug and play models as they would in a laboratory setting [Brü02, Cel93].

As an example of how this is accomplished, consider the system shown in figure 4.4. Here a 4th order, spring, mass, damper system is shown with gravity neglected and $X1 = 0$, $X2 = 0$ are defined as the unloaded spring displacements for $K1$ and $K2$, respectively.

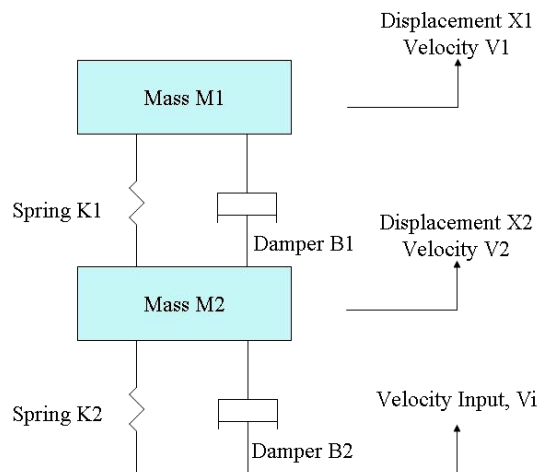
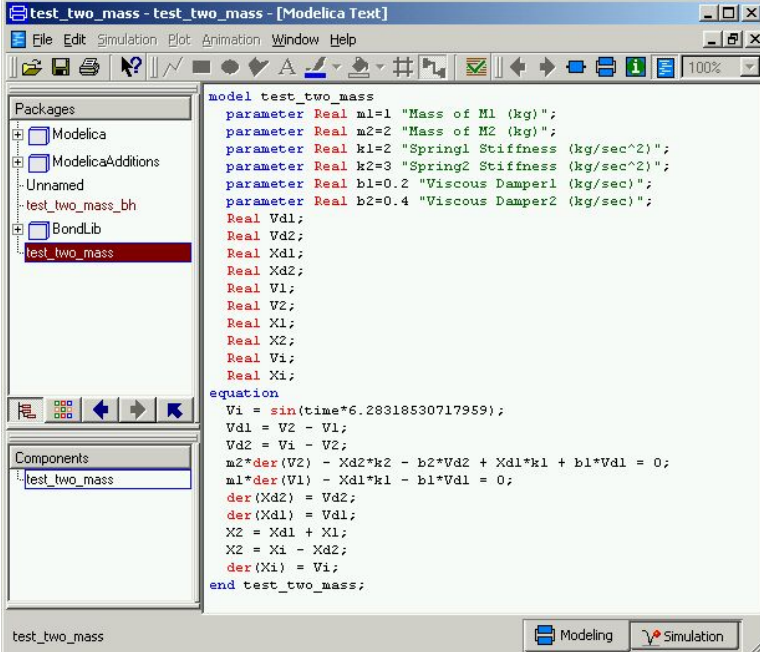


Figure 4.4. Spring Mass Damper: Example

A possible set of equations for this system is

$$\begin{aligned}
 V\delta 1 &= V2 - V1 \\
 V\delta 2 &= Vi - V2 \\
 m2 * \dot{V}2 - X\delta 2 * K2 - B2 * V\delta 2 + X\delta 1 * K1 + B1 * V\delta 1 &= 0 \\
 m1 * \dot{V}1 - X\delta 1 * K1 - B1 * V\delta 1 &= 0 \\
 \dot{X}\delta 2 &= V\delta 2 \\
 \dot{X}\delta 1 &= V\delta 1 \\
 X2 &= X\delta 1 + X1 \\
 X2 &= Xi - X\delta 2 \\
 \dot{X}i &= Vi
 \end{aligned}
 \tag{4.2}$$

Naturally, equation set 4.2 does not represent a minimum set of equations, since a minimum set of equations would make the example trivial. Dymola does not require that a set of equations be minimal either.



```

model test_two_mass
  parameter Real m1=1 "Mass of M1 (kg)";
  parameter Real m2=2 "Mass of M2 (kg)";
  parameter Real k1=2 "Spring1 Stiffness (kg/sec^2)";
  parameter Real k2=3 "Spring2 Stiffness (kg/sec^2)";
  parameter Real b1=0.2 "Viscous Damper1 (kg/sec)";
  parameter Real b2=0.4 "Viscous Damper2 (kg/sec)";
  Real Vd1;
  Real Vd2;
  Real Xd1;
  Real Xd2;
  Real V1;
  Real V2;
  Real X1;
  Real X2;
  Real Vi;
  Real Xi;
equation
  Vi = sin(time*6.28318530717959);
  Vd1 = V2 - V1;
  Vd2 = Vi - V2;
  m2*der(V2) - Xd2*k2 - b2*Vd2 + Xd1*k1 + b1*Vd1 = 0;
  m1*der(V1) - Xd1*k1 - b1*Vd1 = 0;
  der(Xd2) = Vd2;
  der(Xd1) = Vd1;
  X2 = Xd1 + X1;
  X2 = Xi - Xd2;
  der(Xi) = Vi;
end test_two_mass;

```

Figure 4.5. Spring Mass Damper: Dymola Equation Window

The equation window of Dymola is shown in figure 4.5 for this example. The icon window, and diagram window, for this model have been left blank since this model is self-contained and not intended to be used at any higher hierarchical level.

Note the *der* notation. This is the symbol Dymola uses to show the derivative of a variable. The notation is somewhat misleading as no numerical derivatives are calculated. The equation sorter works such that only numerical integration is needed to perform the simulation. Also note that the equations shown in figure 4.5 are in the exact same form as those in equation set 4.2.

In order to represent the sorting algorithm's functionality, an equation notation is adopted. A bracket around an entity indicates that the equation is solved for that entity. An underlined entity indicates that the variable is known from some other source. This notation was developed by Elmqvist, and Otter [Elm94]. For example, if equation 4.1 were solved for E the equation would be written as shown in equation 4.3 indicating that the values for A , B , C , D , and F must be obtained from some other source.

$$\underline{A} + \underline{B} + \underline{C} = \underline{D} + [E] + \underline{F} \quad (4.3)$$

In order to sort the equations shown in figure 4.5, Dymola first solves any statement containing a *der* for this entity. Naturally, if $der(Y)$ is known, then Y is also known as it is the output of the integration scheme. With this, equation set 4.2 becomes that shown in equation set 4.4. V_i is known as it is an input to the system.

$$\begin{aligned}
V\delta 1 &= \underline{V2} - \underline{V1} \\
V\delta 2 &= \underline{Vi} - \underline{V2} \\
m2 * [\dot{V}2] - \underline{X\delta 2} * K2 - B2 * V\delta 2 + \underline{X\delta 1} * K1 + B1 * V\delta 1 &= 0 \\
m1 * [\dot{V}1] - \underline{X\delta 1} * K1 - B1 * V\delta 1 &= 0 \\
[\dot{X}\delta 2] &= V\delta 2 \\
[\dot{X}\delta 1] &= V\delta 1 \\
X2 &= \underline{X\delta 1} + X1 \\
X2 &= \underline{Xi} - \underline{X\delta 2} \\
[\dot{X}i] &= \underline{Vi}
\end{aligned} \tag{4.4}$$

All equations containing a single unknown variable are then solved and the information is extended to the remaining equations. Equation set 4.4 becomes

$$\begin{aligned}
[V\delta 1] &= \underline{V2} - \underline{V1} \\
[V\delta 2] &= \underline{Vi} - \underline{V2} \\
m2 * [\dot{V}2] - \underline{X\delta 2} * K2 - B2 * \underline{V\delta 2} + \underline{X\delta 1} * K1 + B1 * \underline{V\delta 1} &= 0 \\
m1 * [\dot{V}1] - \underline{X\delta 1} * K1 - B1 * \underline{V\delta 1} &= 0 \\
[\dot{X}\delta 2] &= V\delta 2 \\
[\dot{X}\delta 1] &= V\delta 1 \\
\underline{X2} &= \underline{X\delta 1} + [X1] \\
[X2] &= \underline{Xi} - \underline{X\delta 2} \\
[\dot{X}i] &= \underline{Vi}
\end{aligned} \tag{4.5}$$

For all equations in set 4.5, there exists exactly one bracketed variable with the other variables underlined. Assignment statements can be generated and this system can then be solved using an integration scheme.

4.2.2 Algebraic Loops

Obviously, the above sorting system breaks down in the presence of an algebraic loop. Systems containing algebraic loops are all too common. These systems occur in resistive networks and are easily shown by a bond graph example.

4.2.2.1 Algebraic Loops within Bond Graph Modeling

This section discusses the typical bond graph method of eliminating algebraic loops. As discussed in Chapter 3, after all integral and necessary causal marks are defined, if there still exists bond graph R -elements with unassigned causality then there is at least one algebraic loop in the system. There is exactly one algebraic loop for every free-causal R -element in the system.

The Wheatstone bridge circuit shown in figure 4.6 is created entirely in the diagram window. Each element of the circuit is a sub-model from the Electrical Library that is standard with the Dymola software.

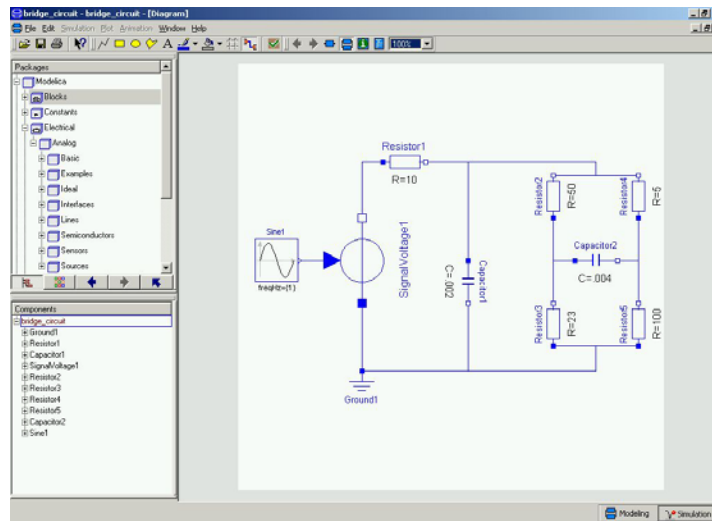


Figure 4.6. Wheatstone Bridge Circuit Example: Dymola Diagram Window

The icon window has been left blank since this model is not intended to be dropped into higher models. The equation window has been left blank since all the equation information is found within the sub-models.

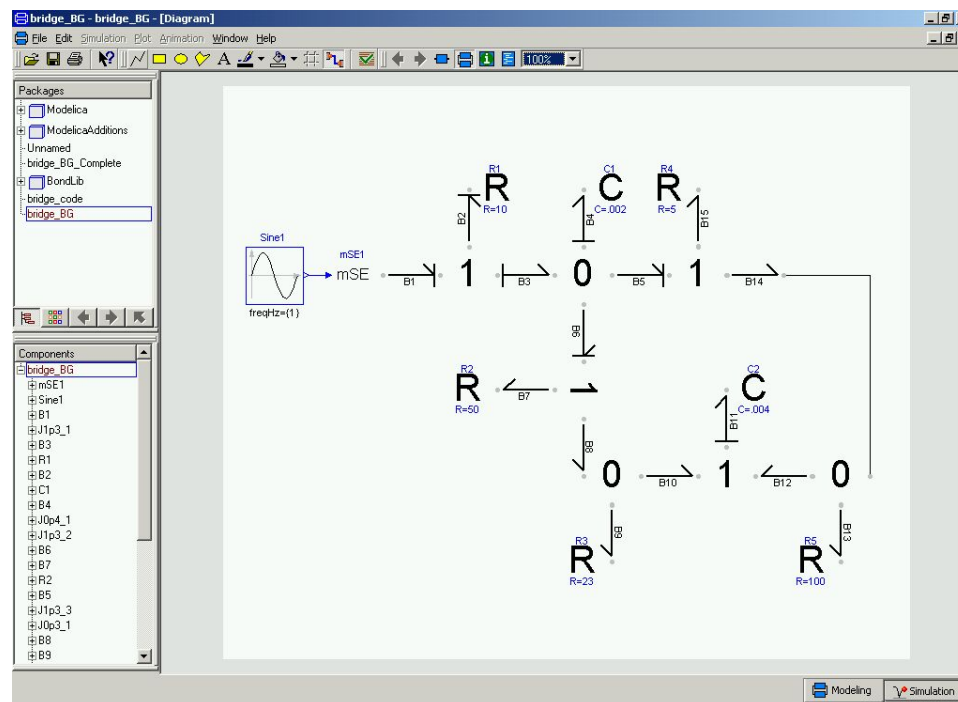


Figure 4.7. Bond Graph of Wheatstone Bridge Circuit

A bond graph model of the Wheatstone bridge circuit is shown in figure 4.7. The n^{th} bond in figure 4.7 is designated by B_n . All necessary causalities and integral causalities have been assigned. Notice that not all bonds have causal marks. The unassigned R -elements make up a network of resistors that contains an algebraic loop. A complete causal bond graph can be obtained by choosing the causality on bond B_9 to define the

effort signal on the 0-junction. Alternatively, the model could be completed by choosing the causality on bond $B13$ to define the effort on the 0-junction. Either way, the final sets of equations are identical. The completed model is shown in figure 4.8.

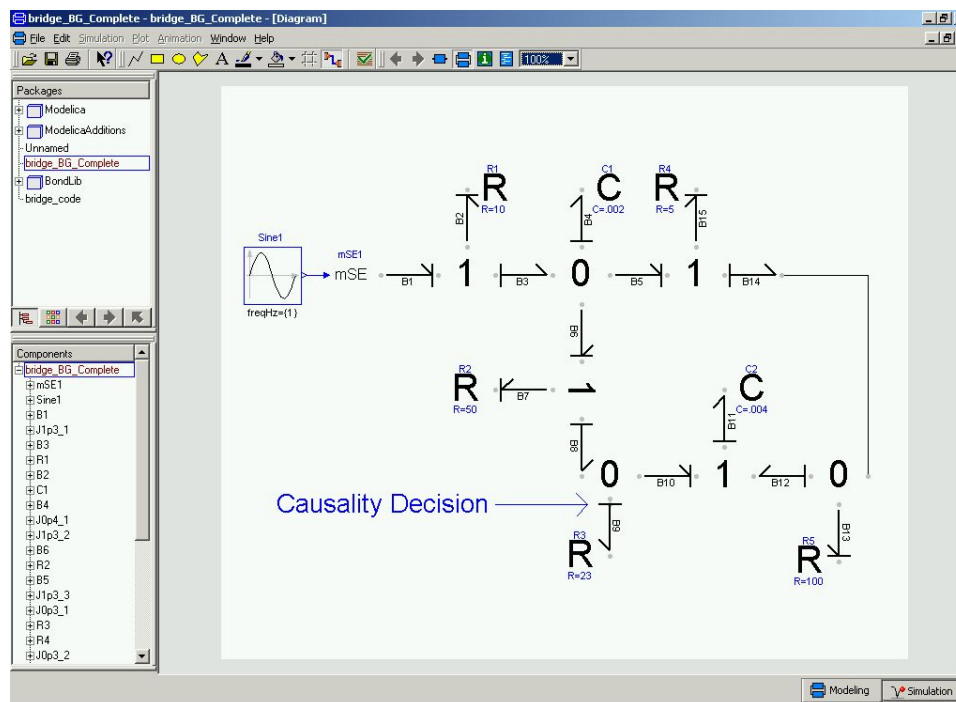


Figure 4.8. Complete Bond Graph of Wheatstone Bridge Circuit

In order to complete the causal assignments of the bond graph in figure 4.7, a minimum of one causality choice was made on the unassigned R -elements. This observation implies that there is a single algebraic loop in the equations [Cel91, Kar90, Kar83] that needs to be broken with a single *tearing* variable [Elm94]. The tearing variable will be the effort signal on $R3$, since this was the causal assignment choice that completed the bond graph. A variable used to break an algebraic loop is referred to as a tearing variable. The tearing variable is then e_{B9} , or the effort on bond $B9$. In order to

derive the equations for a bond graph containing an algebraic loop, the equation derivation rules of Chapter 3 are followed. The normal bond graph equation derivation rules apply up until the tearing variable is encountered. The tearing variable should be left as an intermediate variable in the equation. However, the tearing variable should be the only intermediate variable in the equation. After each equation is complete, an extra equation for the tearing variable is derived. This is best shown via the above example. The Wheatstone bridge example is a second-order system which can be seen by the number of integral causal marks on the bond graph. Following the rules of Chapter 3, the first set of bond graph equations is derived as

$$\dot{q}_{B4} = f_{B3} - f_{B6} - f_{B5} = f_{B2} - f_{B7} - f_{B15} \quad (4.6)$$

$$\dot{q}_{B4} = \frac{e_{B2}}{R1} - \frac{e_{B7}}{R2} - \frac{e_{B15}}{R4} \quad (4.7)$$

$$\dot{q}_{B4} = \frac{1}{R1} \left[mSE1 - \frac{q_{B4}}{C1} \right] - \frac{1}{R2} \left[\frac{q_{B4}}{C1} - e_{B9} \right] - \frac{1}{R4} \left[\frac{q_{B4}}{C1} - e_{B12} \right] \quad (4.8)$$

$$\dot{q}_{B4} = \frac{1}{R1} \left[mSE1 - \frac{q_{B4}}{C1} \right] - \frac{1}{R2} \left[\frac{q_{B4}}{C1} - e_{B9} \right] - \frac{1}{R4} \left[\frac{q_{B4}}{C1} - \left(\frac{q_{B11}}{C2} - e_{B9} \right) \right] \quad (4.9)$$

$$\dot{q}_{B4} = \frac{1}{R1} \left[mSE1 - \frac{q_{B4}}{C1} \right] - \frac{1}{R2} \left[\frac{q_{B4}}{C1} - e_{B9} \right] - \frac{1}{R4} \left[\frac{q_{B4}}{C1} - \frac{q_{B11}}{C2} + e_{B9} \right] \quad (4.10)$$

Equation 4.10 is in typical bond graph equation form, with the exception of the intermediate variable e_{B9} . In equation 4.10, all intermediate variables have been eliminated with the exception of the tearing variable. Rearranging equation 4.10 to isolate e_{B9} yields

$$\dot{q}_{B4} = \frac{mSE1}{R1} + \frac{q_{B11}}{C2R4} + e_{B9} \left(\frac{1}{R2} - \frac{1}{R4} \right) - \frac{q_{B4}}{C1} \left(\frac{1}{R1} + \frac{1}{R2} + \frac{1}{R4} \right) \quad (4.11)$$

Similarly;

$$\dot{q}_{B11} = f_{B12} = f_{B14} - f_{B13} = f_{B15} - f_{B13} = \frac{e_{B15}}{R4} - \frac{e_{B13}}{R5} \quad (4.12)$$

$$\dot{q}_{B11} = \frac{1}{R4} \left[\frac{q_{B4}}{C1} - e_{B12} \right] - \frac{e_{B12}}{R5} = \frac{1}{R4} \left[\frac{q_{B4}}{C1} - \left(\frac{q_{B11}}{C2} - e_{B9} \right) \right] - \frac{1}{R5} \left(\frac{q_{B11}}{C2} - e_{B9} \right) \quad (4.13)$$

$$\dot{q}_{B11} = \frac{1}{R4} \left[\frac{q_{B4}}{C1} - \frac{q_{B11}}{C2} + e_{B9} \right] - \frac{1}{R5} \left(\frac{q_{B11}}{C2} - e_{B9} \right) \quad (4.14)$$

$$\dot{q}_{B11} = \frac{q_{B4}}{R4C1} + e_{B9} \left(\frac{1}{R5} + \frac{1}{R4} \right) - \frac{q_{B11}}{C2} \left(\frac{1}{R5} + \frac{1}{R4} \right) \quad (4.15)$$

In order to eliminate the tearing variable from equations 4.11 and 4.15, a bond graph equation is written to express the tearing variable. Since this is a known algebraic loop, the tearing variable equation will be a function of bond graph states and itself, as shown in the following equation:

$$e_{B9} = F(q_{B4}, q_{B11}, e_{B9}) \quad (4.16)$$

The tearing variable equation for the Wheatstone bridge example is derived in equations 4.17-4.20.

$$e_{B9} = R3 * f_{B9} = R3[f_{B8} - f_{B10}] = R3[f_{B7} - f_{B12}] = R3[f_{B7} - (f_{B14} - f_{B13})] \quad (4.17)$$

$$e_{B9} = R3[f_{B7} - f_{B15} + f_{B13}] = R3 \left[\frac{e_{B7}}{R2} - \frac{e_{B15}}{R4} + \frac{e_{B13}}{R5} \right] \quad (4.18)$$

$$e_{B9} = R3 \left[\frac{1}{R2} \left(\frac{q_{B4}}{C1} - e_{B9} \right) - \frac{1}{R4} \left(\frac{q_{B4}}{C1} - e_{B12} \right) + \frac{e_{B12}}{R5} \right] \quad (4.19)$$

$$e_{B9} = R3 \left[\frac{1}{R2} \left(\frac{q_{B4}}{C1} - e_{B9} \right) - \frac{1}{R4} \left(\frac{q_{B4}}{C1} - \left(\frac{q_{B11}}{C2} - e_{B9} \right) \right) + \frac{1}{R5} \left(\frac{q_{B11}}{C2} - e_{B9} \right) \right] \quad (4.20)$$

Equation 4.20 shows the algebraic loop explicitly. If the sorting algorithm does not recognize how to break an algebraic loop, then the steps taken to derive equation 4.20 repeat for every occurrence of the term e_{B9} , creating indefinite recursion. Obviously, e_{B9} needs to be determined algebraically in equation 4.20. Equation 4.20 becomes

$$e_{B9} = \frac{R3[q_{B4}C2R5(R4 - R2) + q_{B11}C1R2(R5 + R4)]}{C1C2[R2R4R5 + R3R4R5 + R2R3R5 + R2R3R4]} \quad (4.21)$$

Upon defining e_{B9} algebraically, the result can then be plugged into equations 4.11 and 4.15 to completely remove the intermediate variables from the equation set. The algebraic loop has been broken.

The bond graph model of the Wheatstone bridge circuit not only predicted that an algebraic loop exists, but also indicated what variable is needed as a tearing variable. This information is not included in the circuit diagram of the same model. The causality information on a bond graph provides added insight that is not readily available through other types of models.

As seen in the above example, algebraic loops can appear in relatively simple bond graph models. Even a simple system containing an algebraic loop, can contain fairly complex recursion. Obviously, software that is intended to simulate bond graphs models directly must be able to recognize when an algebraic loop exists, and how to break it.

4.2.2.2 Algebraic Loops within Dymola

Dymola uses an algorithm by Tarjan [Tar72] to sort the equations describing a system. Naturally the equations need to be sorted horizontally, as described in Section 4.2.1, and also vertically. Vertical sorting simply implies that the algorithm must order the equations from top to bottom such that all information is available when needed. The Tarjan algorithm, in order to be robust, must also be able to handle algebraic loops.

The Tarjan algorithm creates a *structure incidence* matrix. This matrix has a row for every equation in the system and a column for every unknown variable. Naturally for the system to be solvable this matrix must be square, one equation per unknown variable. The matrix is populated with ones or zeros. In the $\langle i, j \rangle$ location of the matrix, a zero indicates that the j^{th} variable does not exist in the i^{th} equation. For example, equation set 4.22 has a structure incidence matrix shown in equation 4.23.

$$\begin{aligned} f_1(x_1, x_2, x_3) &= 0 \\ f_2(x_2) &= 0 \\ f_3(x_1, x_2) &= 0 \end{aligned} \tag{4.22}$$

$$\begin{array}{rcccl} & x_1 & x_2 & x_3 & \\ f_1 : & 1 & 1 & 1 & \\ f_2 : & 0 & 1 & 0 & \\ f_3 : & 1 & 1 & 0 & \end{array} = S \tag{4.23}$$

By interchanging rows and columns of the structure incidence matrix it is possible to rearrange it into lower-triangular form. Equation 4.23 becomes

$$\begin{array}{rcccl}
 & x_2 & x_1 & x_3 & \\
 f_2 : & 1 & 0 & 0 & \\
 f_3 : & 1 & 1 & 0 & \\
 f_1 : & 1 & 1 & 1 & \\
 \end{array} = S \quad (4.24)$$

Equation 4.24 represents a system of equations that are sorted both horizontally and vertically.

It is possible that the structure incidence matrix, after row and column permutations, is not entirely lower-triangular. In this case, it will be lower-triangular with blocks on the diagonal. This occurs in the presence of algebraic loops. For example, equation set 4.25 has a structure incidence matrix shown in equation 4.26.

$$\begin{array}{l}
 f_1(x_1, x_2, x_3) = 0 \\
 f_2(x_1, x_2) = 0 \\
 f_3(x_1, x_2) = 0
 \end{array} \quad (4.25)$$

$$\begin{array}{rcccl}
 & x_1 & x_2 & x_3 & \\
 f_1 : & 1 & 1 & 1 & \\
 f_2 : & 1 & 1 & 0 & \\
 f_3 : & 1 & 1 & 0 & \\
 \end{array} = S \quad (4.26)$$

Equation 4.26 can be permuted to equation 4.27, which has a 2x2 block on the diagonal.

$$\begin{array}{rcccl}
 & x_1 & x_2 & x_3 & \\
 f_3 : & 1 & 1 & 0 & \\
 f_2 : & 1 & 1 & 0 & \\
 f_1 : & 1 & 1 & 1 & \\
 \end{array} = S \quad (4.27)$$

In this case, either x_1 or x_2 can be used as the tearing variable. Algebraic loops will create rank deficiency in the structure incidence matrix.

For dynamic systems, the equations have the form $\dot{X} = f(X, U, t)$. Here \dot{X} is considered an unknown for the Tarjan algorithm, and X is considered known, since it is the output of the integration routine.

The Tarjan algorithm does not solve the algebraic loops. In the Modelica framework, matrix techniques are used for algebraic loops that are linear and moderate in dimension, and Newton iteration is used on large and/or nonlinear algebraic loops.

It is seen here that the Modelica framework using the Tarjan algorithm together with Newton iteration/matrix techniques is capable of handling bond graph models.

4.2.3 Structural Singularities: The Higher Index Problem

In a system, a structural singularity occurs when a potential degree of freedom is constrained such that its dynamics are described entirely as a function of other state variables. For example, two inertias connected by an ideal gear train create a structural singularity. Normally, each of these inertias separately has its own degree of freedom. The ideal gear train, however, constrains them such that they act as a single degree of freedom.

4.2.3.1 Structural Singularities within Bond Graph Modeling

The gear train example is described in figure 4.9 between the rotational inertias I_1 and I_2 . The bond graph for this system is shown in figure 4.10. Notice the differential causal mark on bond B_2 . Differential causality in a bond graph shows up in the presence of a structural singularity [Cel91]. Normally the I -elements would have their own integral causal mark indicating a degree of freedom, but the gear train couples the two inertias together such that movement by one can be described entirely by the dynamics of the other.

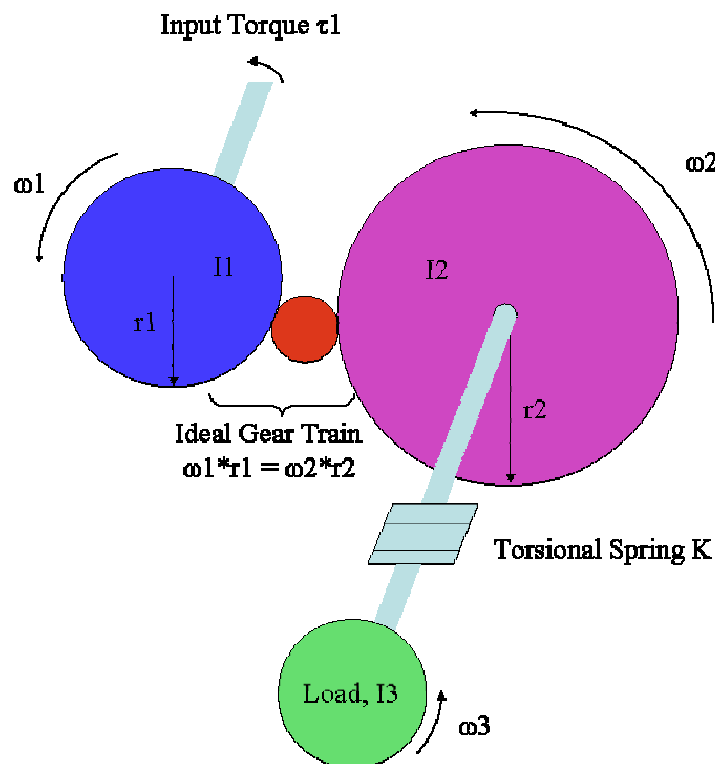


Figure 4.9. Gear Train Example

The number of integral causal marks on the bond graph indicates that the system is a 3rd order system. The inertia of the small gear between $I1$ and $I2$ has been neglected. The bond graph shows that, even though there are four energy storage elements, two of these elements are coupled together as one. The transformer constant comes from the gear ratio of the gear train. The gear train, in figure 4.9, shows the relationship between the angular velocities of the gears. This relationship must hold true for the transformer in between the 1-junction representing $\omega1$ and the 1-junction representing $\omega2$. The causal marks on the 1-junction show that the flow signal moves from the $\omega2$ 1-junction to the $\omega1$ 1-junction. Therefore, the transformer constant must be $r1/r2$ as shown in figure 4.10.

Also notice that the differential causal mark could have been placed on $B1$. This would have caused the dynamics of $I1$ to be described by $I2$. As in the last section, a choice on causality led to an algebraic loop. The same is true here only the loop comes about in a slightly different manner. If the $I1$ element were differential causal, then the causality on the transformer would change as well, which would in turn invert the direction of the flow signal. Thus, the transformer constant would need to be inverted as well.

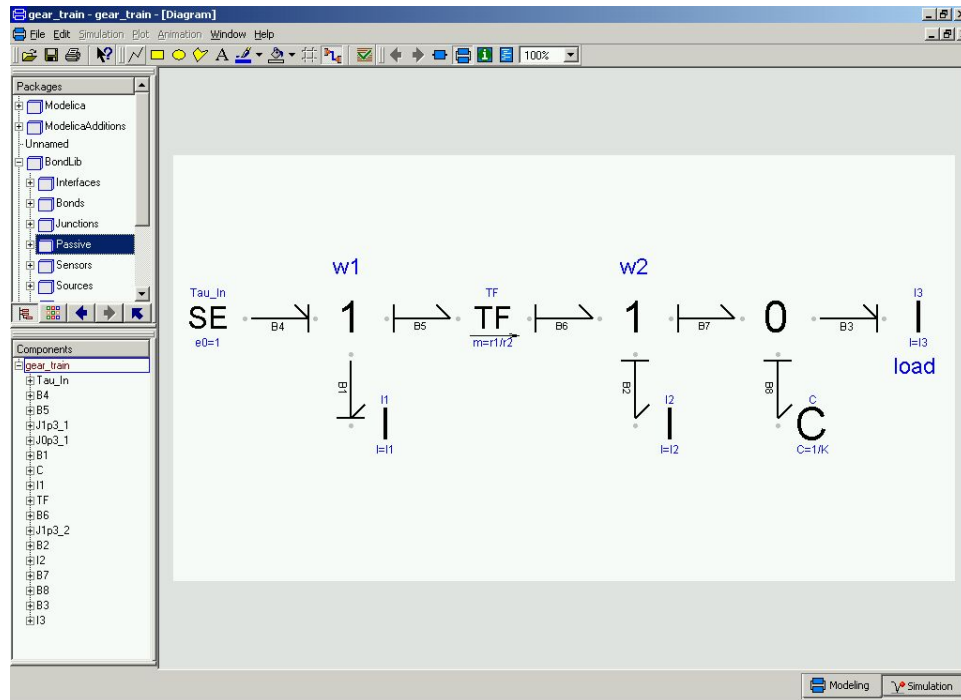


Figure 4.10. Gear Train Bond Graph

The first equation of the above bond graph is derived as before

$$\dot{P}_{B1} = SE - e_{B5} = SE - \frac{r1}{r2} e_{B6} = SE - \frac{r1}{r2} [e_{B2} + e_{B7}] = SE - \frac{r1}{r2} [e_{B2} + e_{B8}] \quad (4.28)$$

$$\dot{P}_{B1} = SE - \frac{r1}{r2} \left[e_{B2} + \frac{q_{B8}}{C} \right] \quad (4.29)$$

Naturally the question is; what to do with e_{B2} ? The answer lies back in the definition of the differential causal elements in figure 3.13.

$$e_{B2} = I2 * \dot{f}_{B2} \quad (4.30)$$

Now \dot{f}_{B2} can be followed through the bond graph in the same fashion as f_{B2} .

$$\dot{f}_{B2} = \dot{f}_{B6} = \frac{r1}{r2} \dot{f}_{B5} = \frac{r1}{r2} \dot{f}_{B1} = \frac{r1}{r2} \frac{\dot{P}_{B1}}{I1} \quad (4.31)$$

Substituting 4.31 and 4.30 into 4.29 yields

$$\dot{P}_{B1} = SE - \frac{r1}{r2} \left[\frac{r1}{r2} \frac{I2}{I1} \dot{P}_{B1} + \frac{q_{B8}}{C} \right] \quad (4.32)$$

Equation 4.32 shows an explicit algebraic loop on the derivative of the state variable \dot{P}_{B1} .

Equation 4.32 needs to be solved algebraically for \dot{P}_{B1} . This type of algebraic loop will always occur in the presence of a differential causal element. Equation 4.32 becomes

$$\dot{P}_{B1} = \frac{SE * I1 * (r2)^2}{[I1 * (r2)^2 + I2 * (r1)^2]} - \frac{q_{B8} * I1 * r2 * r1}{C [I1 * (r2)^2 + I2 * (r1)^2]} \quad (4.33)$$

Equation 4.31 shows how the generic acceleration term moves through the transformer, i.e., $\dot{f}_{B6} = \frac{r1}{r2} \dot{f}_{B5}$. In general, one must use the chain rule to develop this

relation, as shown in equation 4.34, since the transformer value may vary with time.

$$\frac{d}{dt}(f_{B6}) = \frac{d}{dt} \left(\frac{r1}{r2} f_{B5} \right) = f_{B5} \frac{d}{dt} \left(\frac{r1}{r2} \right) + \frac{r1}{r2} \frac{d}{dt} (f_{B5}) \quad (4.34)$$

Naturally $r1/r2$ is constant, so equation 4.33 reduces to the form shown in equation 4.31.

The other two equations of the gear train bond graph will not touch the differential causal bond. For completeness they are

$$\dot{q}_{B8} = f_7 - f_3 = f_6 - \frac{P_{B3}}{I3} = f_5 \frac{r1}{r2} - \frac{P_{B3}}{I3} = f_1 \frac{r1}{r2} - \frac{P_{B3}}{I3} \quad (4.35)$$

$$\dot{q}_{B8} = \frac{P_{B1}}{I1} \frac{r1}{r2} - \frac{P_{B3}}{I3} \quad (4.36)$$

$$\dot{P}_{B3} = e_{B3} = e_{B3} = \frac{q_{B8}}{C} \quad (4.37)$$

This section shows that bond graphs detect structural singularities and provide the necessary means to solve them. Once the differential causal relationship is determined, the structural singularity will usually generate an algebraic loop involving one, or more, of the state derivatives.

4.2.2.2 Algebraic Loops within Dymola

Dymola uses a different method for handling structural singularities. An algorithm developed by Pantelides [Pan88] is implemented in the Modelica framework to accomplish this task. The Pantelides algorithm, like Tarjan, also uses the structure incidence matrix. When a structural singularity exists in a system, the incidence matrix will have a row that is empty. Thus, an equation exists that is not dependent on any unknowns.

For the gear train example above the typical equations are found by balancing the torques on the inertia elements. This set of equations is shown as

$$\begin{aligned}
 \tau_{in} &= I1 * \dot{\omega}1 + \frac{r1}{r2} [I2 * \dot{\omega}2 + k * \phi_8] \\
 \dot{\phi}_8 &= \omega8 \\
 k * \phi_8 &= I3 * \dot{\omega}3 \\
 \omega2 &= \omega8 + \omega3 \\
 r1 * \omega1 &= r2 * \omega2
 \end{aligned} \tag{4.38}$$

The unknowns are ordered $\dot{\omega}1$, $\dot{\omega}2$, $\dot{\omega}3$, $\dot{\phi}_8$, and $\omega8$. Note that $\omega8$ is an unknown, since there is no equation containing $\dot{\omega}8$. Thus, the structure incidence matrix is

$$S = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (4.39)$$

The last equation does not depend on any unknown variables. This is indicative of a structural singularity. The equation that is independent of the unknowns is a constraint equation. Upon finding a constraint equation, the Pantelides algorithm differentiates it with respect to time. Thus, the constraint equation of the above example becomes

$$r1 * \dot{\omega}1 = r2 * \dot{\omega}2 \quad (4.40)$$

Equation 4.40 is then added to the list of equations. As a result, there is now one equation too many. In order to balance out the number of equations with the number of unknowns, one of the integral relationships is relaxed. For example, for the added equation 4.40 it is no longer assumed that $\omega1$ is known from the integration of $\dot{\omega}1$.

Thus, $\omega1$ is added to the list of unknowns. The new structure incidence matrix becomes

$$\hat{S} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (4.41)$$

The new structure incidence matrix does not have a row of all zeros. The unknowns are now $\dot{\omega}1$, $\dot{\omega}2$, $\dot{\omega}3$, $\dot{\phi}_8$, $\omega8$, and $\omega1$. The equations consist of equation set 4.38 and the added equation 4.40. The added equation, together with the added unknown, increases the index of the structure incidence matrix, thus the term *the higher index problem*

[Pan88]. Also, note that the constraint equation 4.40 also comes out of the bond graph derivation as equation 4.34. Equation 4.41, however, indicates an algebraic loop. Row 1 and row 6 are identical, which creates a block element on the diagonal after the matrix has been permuted.

The relaxation of the state from its derivative should not be surprising. The end result is that this *state* is not really a state at all. Its dynamics can be described entirely as a function of the other true states. The fact that the derivative of the variable and the variable itself, are not determined by an integral equation implies that the relationship must be dependent upon other parameters. This was seen in the bond graph derivation of equations in the presence of a structural singularity.

Upon derivation of the constraint equation, it is possible that new variables are created. For example the variable $\dot{\psi}$ is created from ψ , when there is no instance of the variable $\dot{\psi}$ in the previous set of equations. In this case, there already exists an equation defining ψ . This equation must also be differentiated. This process continues until no new variables are created.

The Modelica framework is capable of handling algebraic loops and structural singularities; both occur often in physical system modeling.

4.3 The Dymola Bond Graph Library

By creating small, object-oriented models that represent bond graph elements it is possible to create a bond graph library in the Dymola/Modelica framework [Cel03a, Cel03b]. A model is created for each of the bond graph elements described in Chapter 3.

4.3.1 Connectors

The most basic model in the bond graph library is the connector. The connector defines a 3-tuple [Cur84] signal to pass bond graph information in and out of models at higher hierarchical levels. The 3-tuple is $\langle e, f, d \rangle$ representing effort and flow, while d is a direction variable where +1 indicates the signal information is into the connector, -1 indicates the signal information is out of the connector. The equation window and icon window, for the connector are shown simultaneously in figure 4.11.

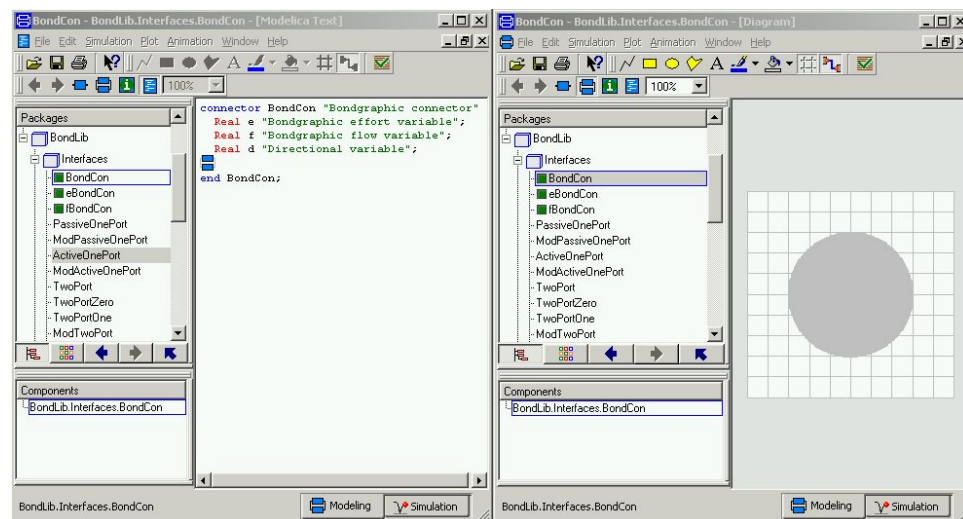


Figure 4.11. Bond Graph Connector

The iconic representation of the connector is a grey dot. The equation window does nothing more than declare the 3-tuple elements.

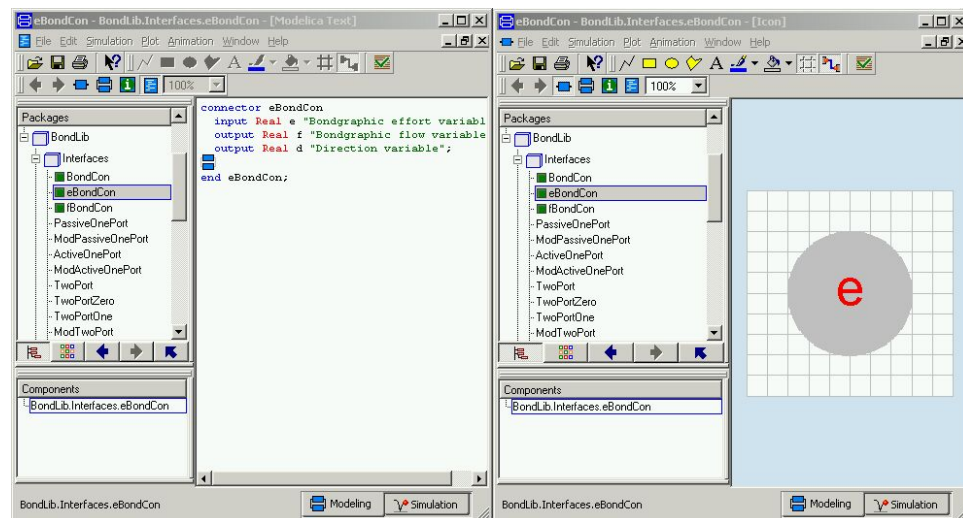


Figure 4.12. *e*-Bond Connector

There are two other bond graph connector models that will be used in causal bond models. As seen in Section 4.2.1 the Modelica framework assumes all variables to be a-causal. The equation sorting algorithm assigns causality. However, causality is an important feature of bond graph modeling. The Dymola models can force causality by declaring a variable as *input* or *output*.

Figure 4.12 shows an *e*-bond connector. The *e* variable is declared as an input and the other two elements of the 3-tuple are declared as outputs. The iconic representation of the *e*-bond is a grey dot with the letter *e* in the center.

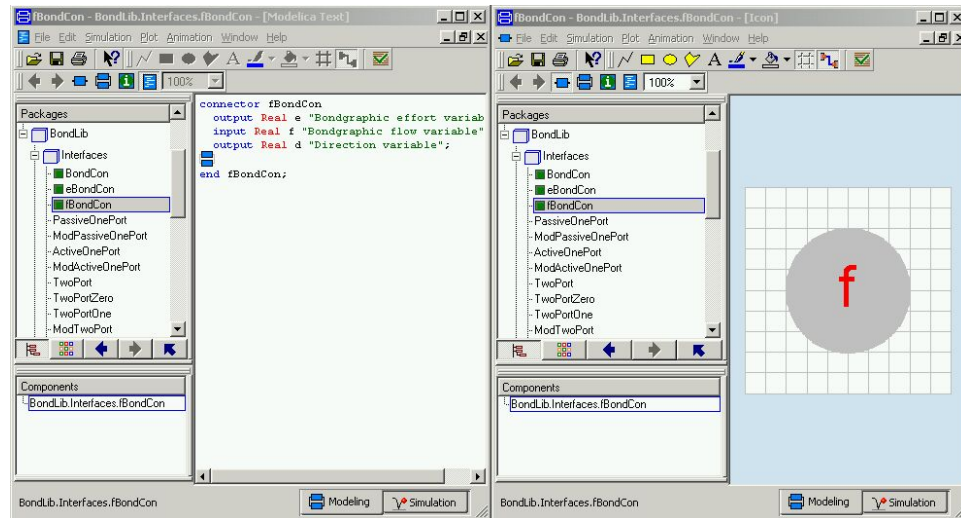


Figure 4.13. f -Bond Connector

Similarly the f -bond connector, shown in figure 4.13, declares the f variable as the input and the other two variables as outputs. With these connectors, it is now possible to define bond graph elements.

4.3.2 Bonds

An a-causal bond is created by dragging two bond graph connectors into the model. Naturally, since two instances of the same sub-model are used in the a-causal bond model, it is necessary to give them separate names. They are named *BondCon1* and *BondCon2*, respectively. Figure 4.14 shows the a-causal bond model. In the icon portion of figure 4.14, *BondCon1* is at the left of the bond and *BondCon2* is at the right of the bond.

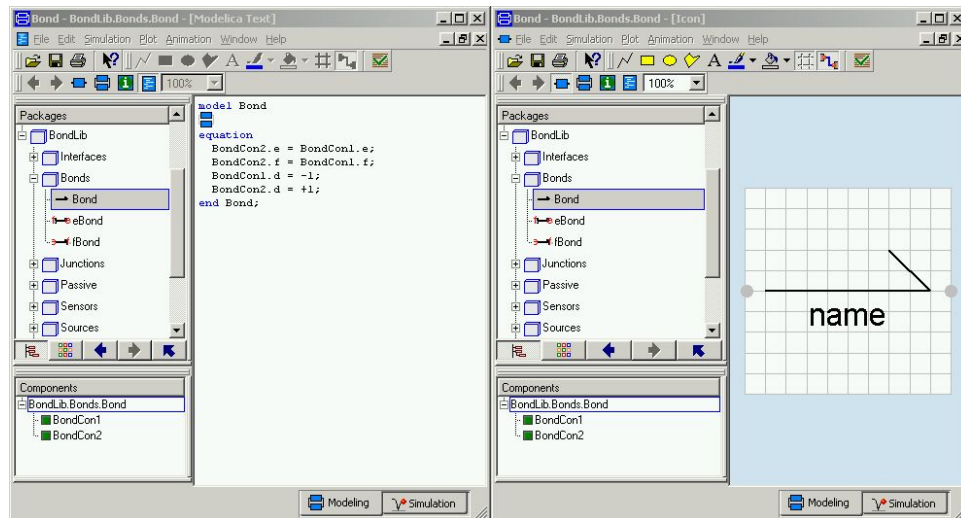


Figure 4.14. A-Causal Bond

As seen in the equation window portion of figure 4.14, the efforts of the two connectors have been set equal, as well as, the flows. The direction variables have been set to -1 for *BondCon1* and +1 for *BondCon2*. The iconic representation is a bond graph half arrow with no causality. The *name* designation is a generic placeholder for naming multiple instances of the same model at the higher, hierarchical level. This placeholder is created by inserting the text “%name” in the icon window.

Naturally the causal bond models need the *e*-bond connector and the *f*-bond connector. These two sub-models will be used to create the causal relationships needed in a bond with assigned causality. The first of the two causal bonds is the *e*-bond. Figure 4.15 shows the *e*-bond with the *f*-bond connector to the left of the icon and the *e*-bond connector to the right. The input/output definitions of the two connectors are used to define the causality relationship of this bond. The icon is a bond with the appropriate causal assignment.

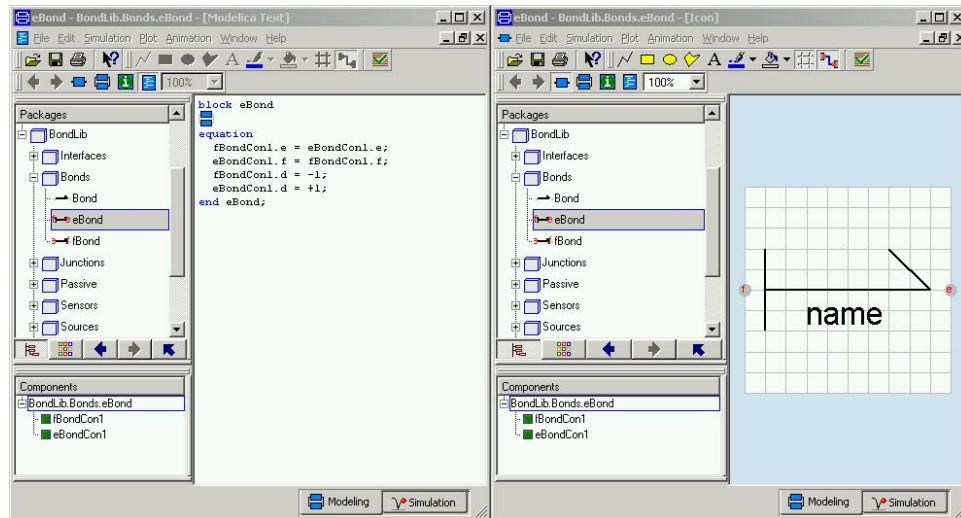


Figure 4.15. *e*-Bond

Notice that the model has been declared as a *block* in figure 4.15. In *blocks*, all variables must have pre-assigned causalities.

An *f*-bond is constructed in a similar fashion. Figure 4.16 shows the equation and icon windows for the *f*-bond. Here, the location of the connectors has been switched.

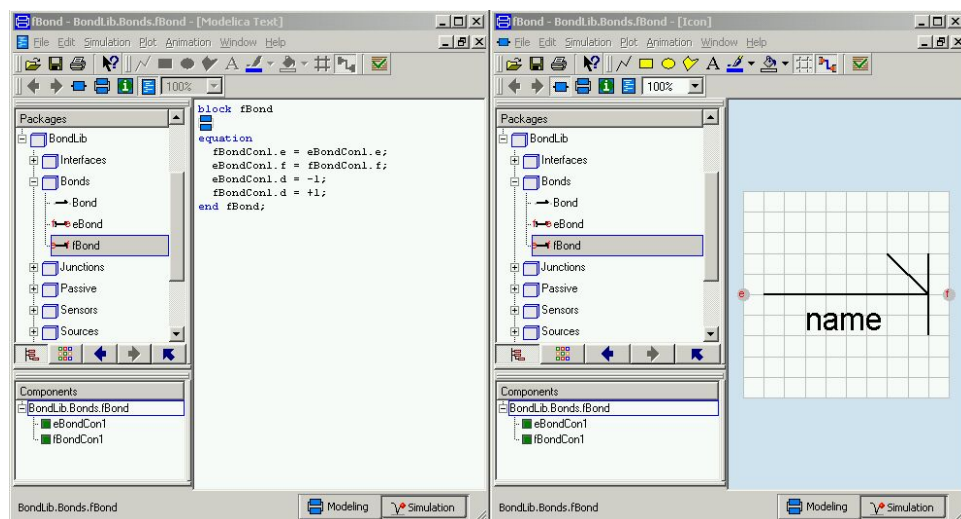


Figure 4.16. *f*-Bond

The bond models are complete and ready to be dropped into Dymola bond graph models.

4.3.3 Junctions

Figure 4.17 shows a 3 bond 0-junction. *Dymola* provides matrix manipulation syntax that is similar to that in *MATLAB* [Mat]. In order to take advantage of this syntax, as seen in the equation window of figure 4.17, the 0-junction inherits the bond connector information through another model called *ThreePortZero* which is shown in figure 4.18.

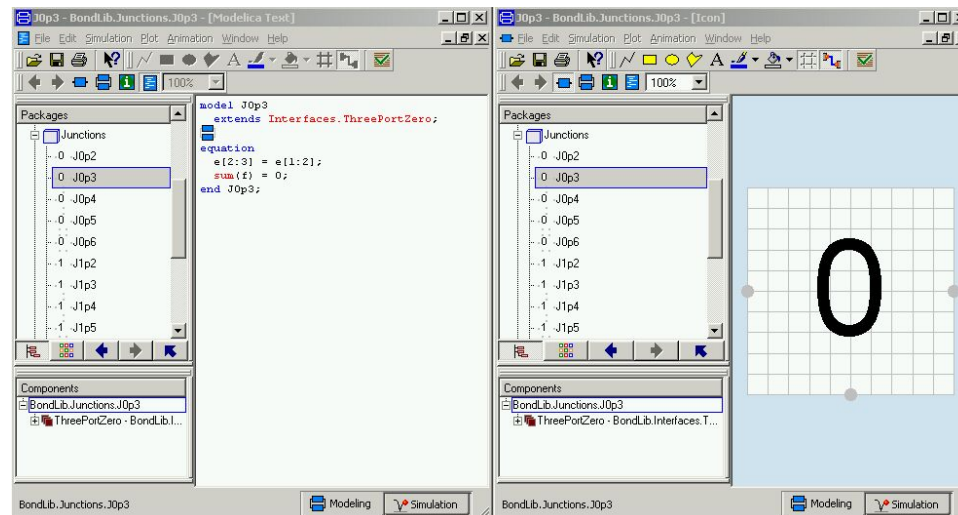


Figure 4.17. 3 Bond 0-Junction

Three bond connector models have been dropped into the model *ThreePortZero*, named *BondCon1*, *BondCon2*, and *BondCon3*. The purpose of *ThreePortZero* is to map

e and f variables into a vector. Note that the d variable is multiplied by the f variable to give the proper sign of the f signal for summing around the 0-junction.

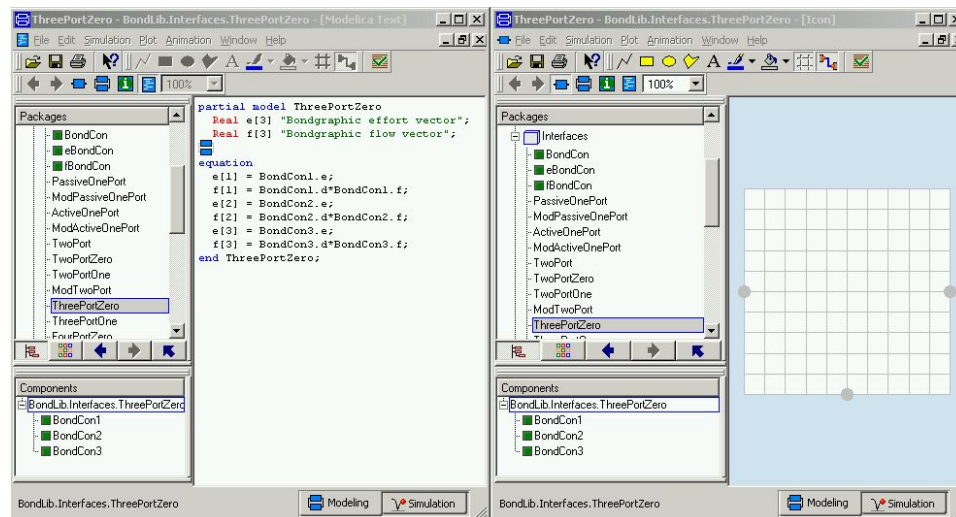


Figure 4.18. Three-Port Zero

Figure 4.17 shows the efforts being set equal around the 0-junction and the flows summing to zero.

The bond graph junction models will need to have n bond graph connector models dropped in to connect n bonds. In bond graph modeling, there is no limit to the number of bonds that can be connected to a junction. However, in order to code a junction model, it is necessary to know how many bonds will be connected to it a priori. To solve this, many junction models are created with n ranging from 2 to 6. By using combinations of these junction models any number of bonds can be connected to a junction. Similar to the 3 bond 0-junction model shown in figures 4.17 and 4.18, there is a 0-junction model for 2, 4, 5, and 6 bond connections.

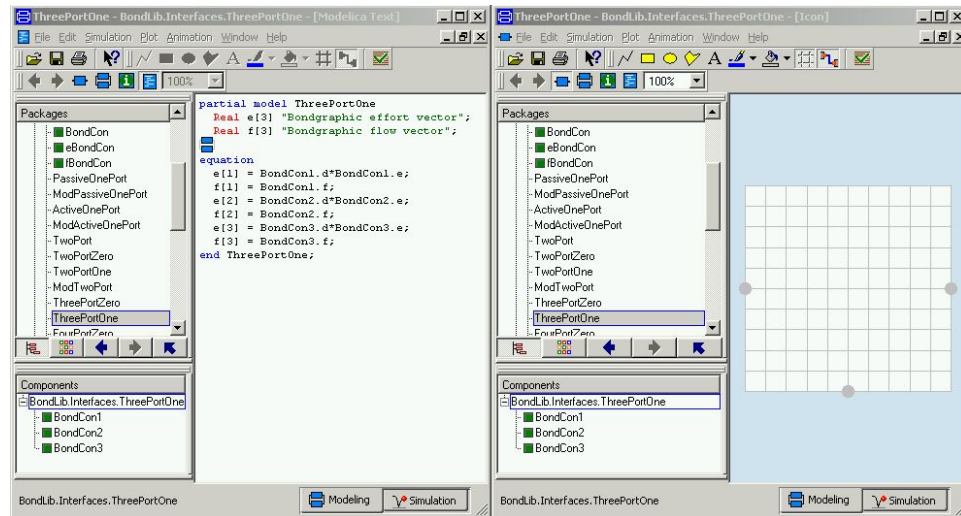


Figure 4.19. *Three-Port One*

The 1-junction models are created in a similar fashion. The 3 bond 1-junction model is shown in figures 4.19 and 4.20. The direction variable is multiplied by the effort variable for summing around the 1-junction. The efforts sum to zero and the flows are equated.

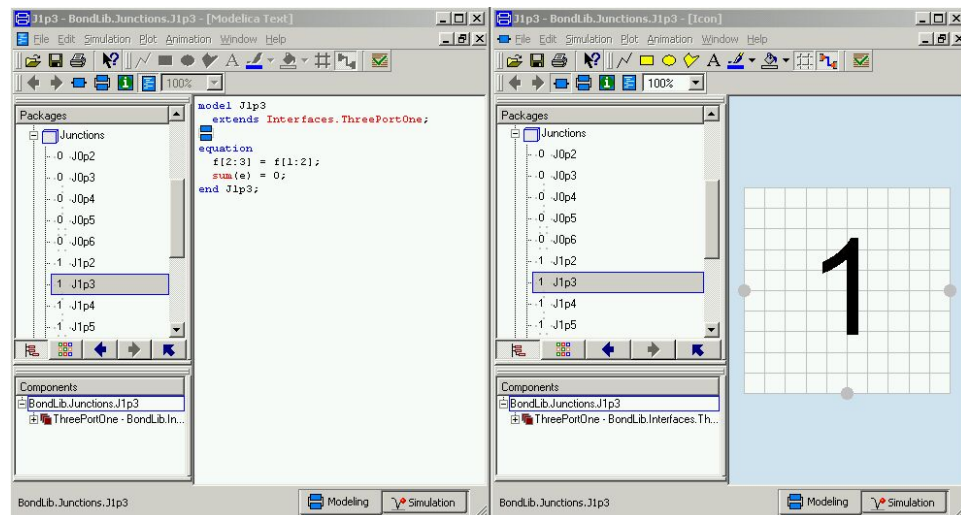


Figure 4.20. *3 Bond 1-Junction*

Note that no causality information is provided to the junction models. The causality information will be inherited from the bonds connected to the junctions. The junction models are now ready to be used in larger models.

4.3.4 Passive Elements

The passive element models need one bond connector model associated with them. Similar to the junction models, this is done through an inheritance structure. Using the inheritance structure makes the creation of these models easier, since the inheritance commands can be typed into the equation window rather than dragging the bond connector model in for each instance. The inheritance model is called *PassiveOnePort*. This model is shown in figure 4.21. *PassiveOnePort* has the bond connector model dragged into it and a definition of e and f . This code can be inherited by all passive elements by using a Modelica *Extends* command in the equation window.

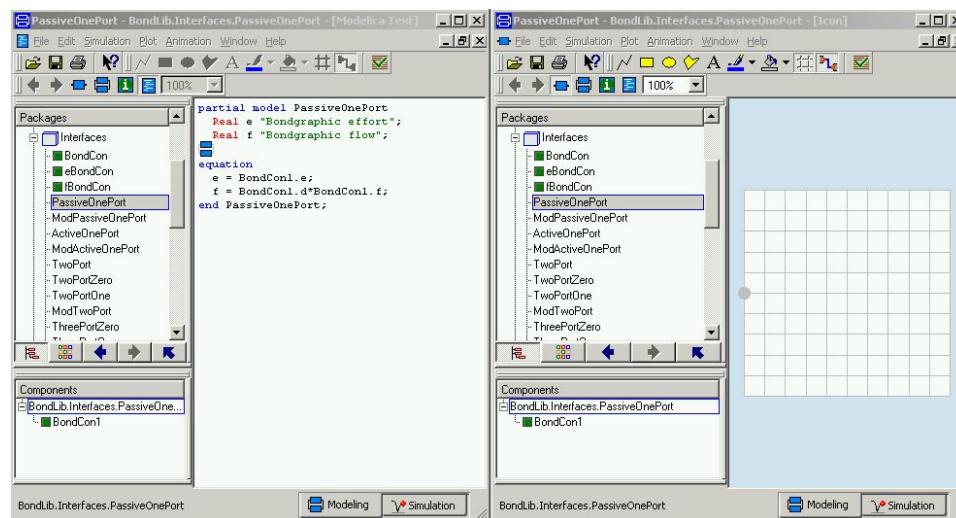


Figure 4.21. *Passive One-Port*

Figure 4.22 shows a bond graph R -element. The bond graph resistive equation is stated in the equation window. The variable R is declared as a parameter, which means that the user defines the parameter upon dropping it into the model. The parameter will be constant throughout the simulation. The value for the parameter R defaults to 1. The parameter value is echoed in the icon of the model by including the text “ $R=\%R$ ” in the icon window.

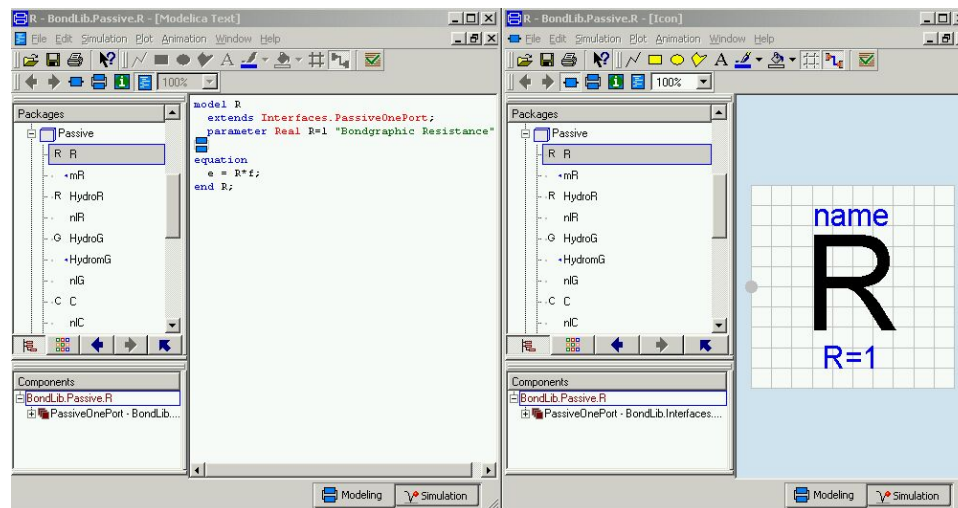


Figure 4.22. R-Element Model

A nonlinear resistive element is shown in figure 4.23. This model does not declare the R variable as a parameter. The R variable is passed into the model via an input signal connector. In this way a nonlinear, or modulated, R -element is created.

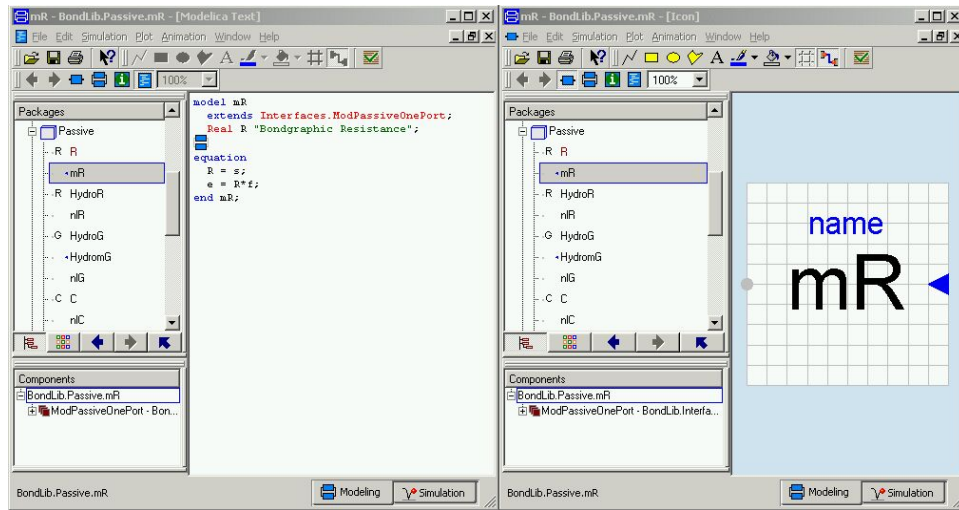


Figure 4.23. mR-Element Model

The I and C elements are created similar to the R -element. They are shown in figures 4.24 and 4.25, respectively. The I and C elements contain the Modelica *der* command to express their relationship with the e and f variables.

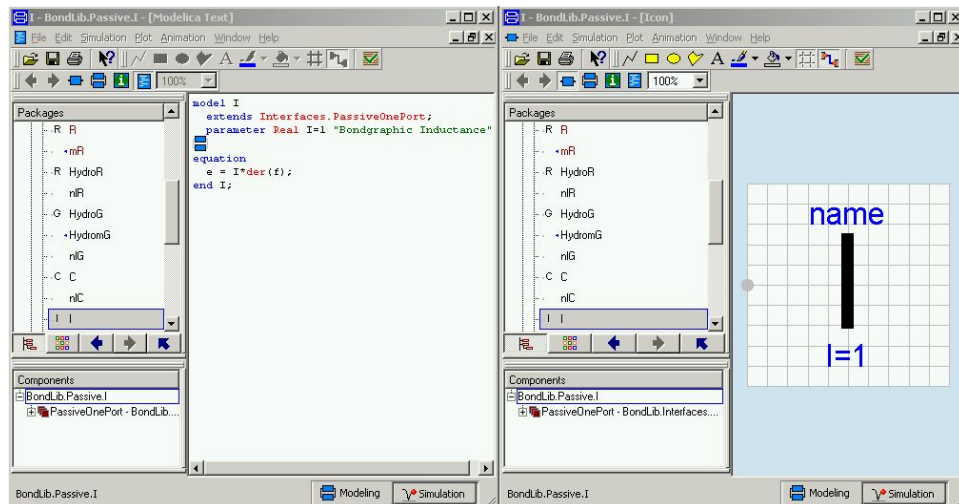


Figure 4.24. I-Element Model

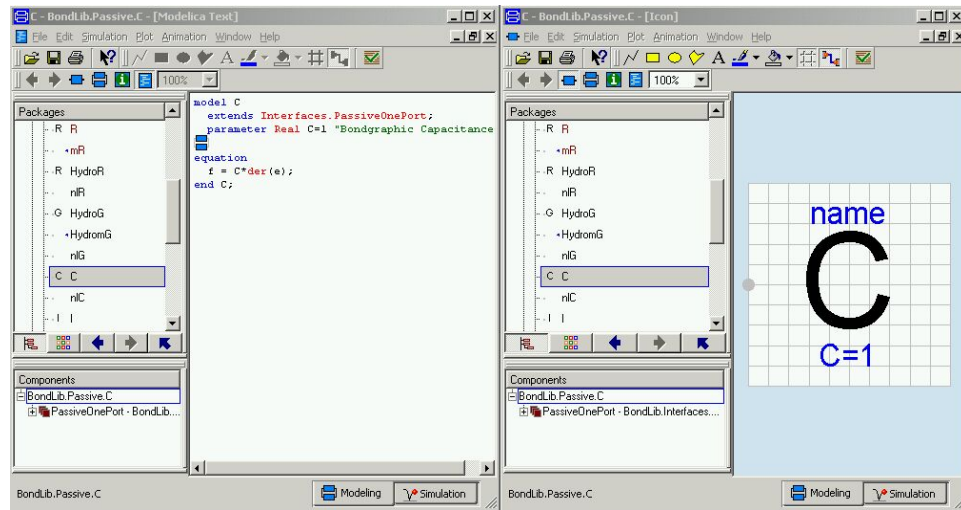
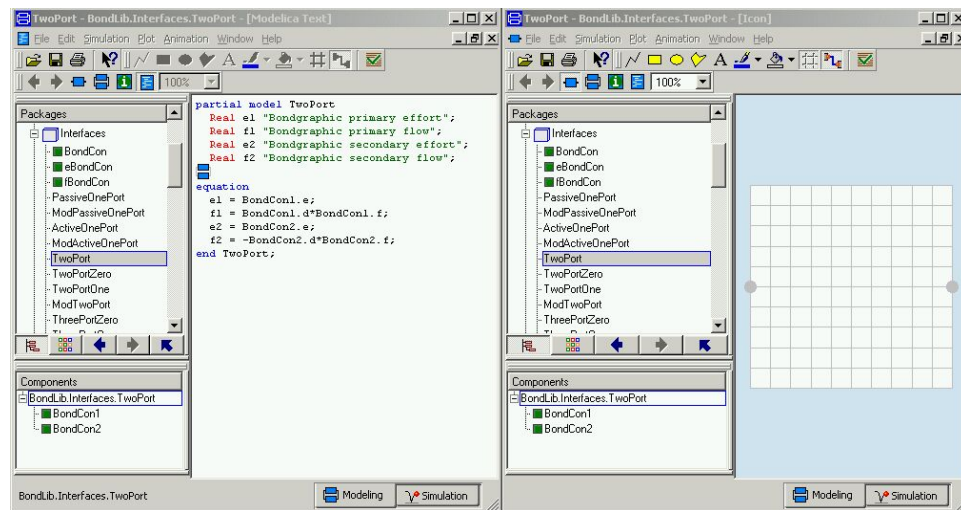


Figure 4.25. C-Element Model

Figure 4.26. *TwoPort*

The transformer and gyrator models are also passive models. They inherit their connector information from a model called *TwoPort*. This model is shown in figure 4.26. *TwoPort* is similar to *PassiveOnePort* in that it is intended to ease the creation of the two-

port models. The direction variables are used to define the sign convention of the f variables.

Two transformer models are shown in figures 4.27 and 4.28. Figure 4.27 shows the linear transformer and figure 4.28 shows the modulated transformer.

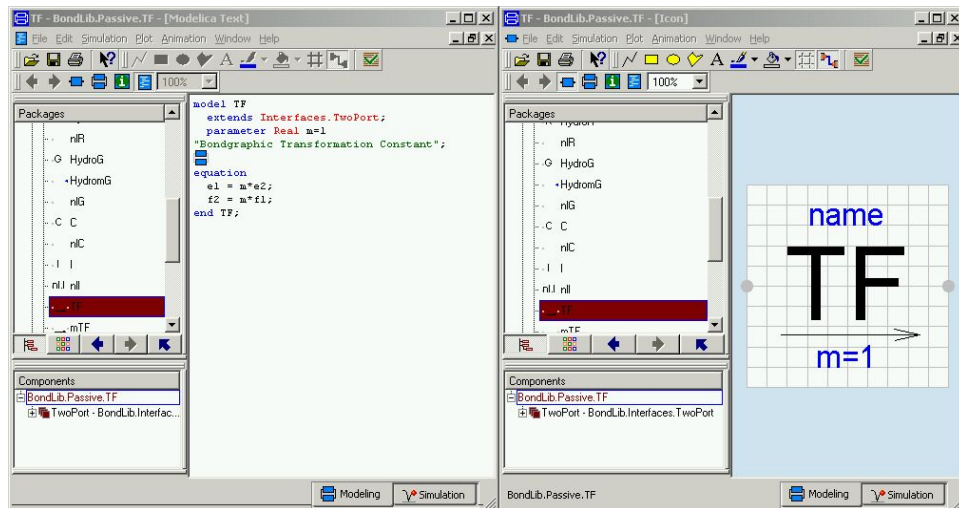


Figure 4.27. Transformer Model

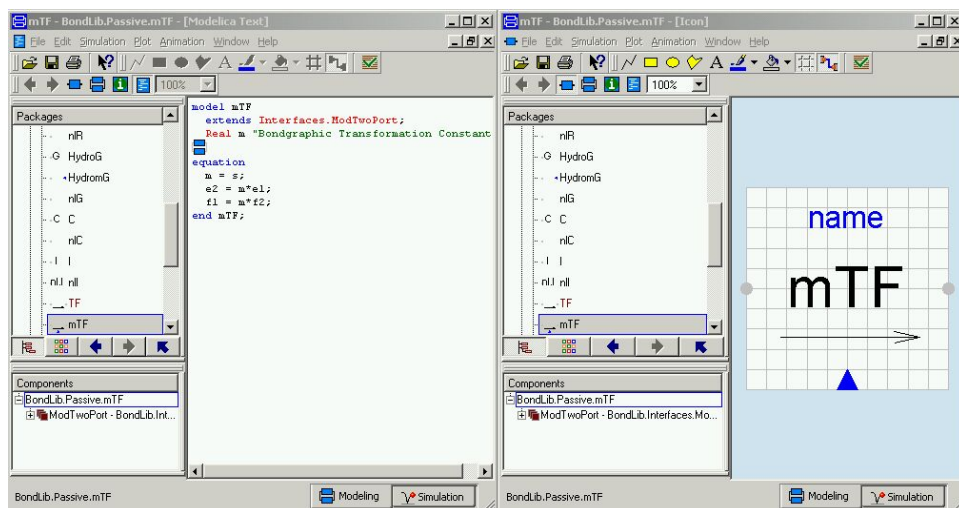


Figure 4.28. Modulated Transformer Model

The transformer model defines the transformer constant as a parameter and the modulated transformer model receives the transformer constant as a signal from outside the model. The equation window shows the necessary relationships between e and f such that no power is stored, or created in the transformer.

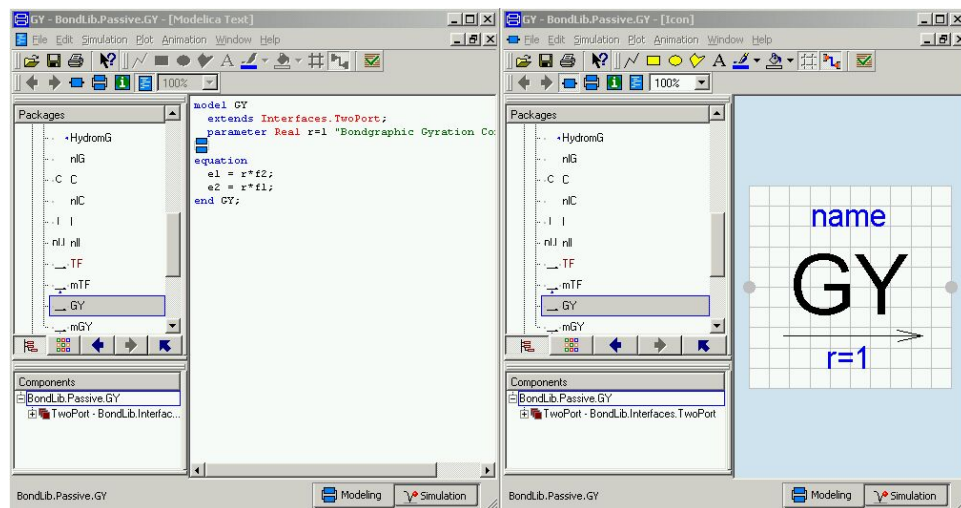


Figure 4.29. Gyrator Model

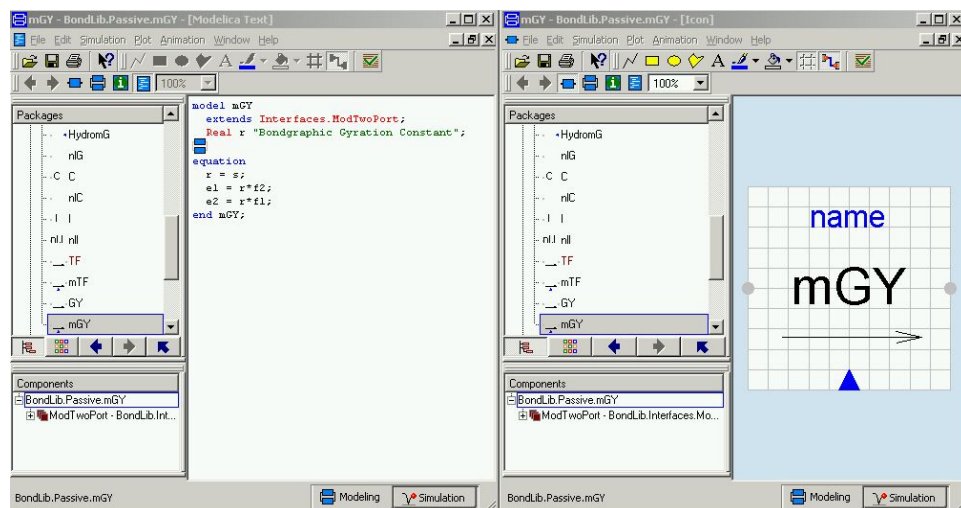


Figure 4.30. Modulated Gyrator Model

The gyrator models are constructed in a similar fashion. For completeness, the gyrator model and modulated gyrator model are shown in figure 4.29 and 4.30, respectively. None of the passive models contain causal information. This information is inherited from the bonds attached to these elements.

4.3.5 Sources

The source models inherit their bond connector model information from the model *ActiveOnePort*. This model is similar to *PassiveOnePort* of figure 4.21, but the direction of the flow variable is negated. Figure 4.31, and 4.32 show an effort source, and modulated effort source models, respectively.

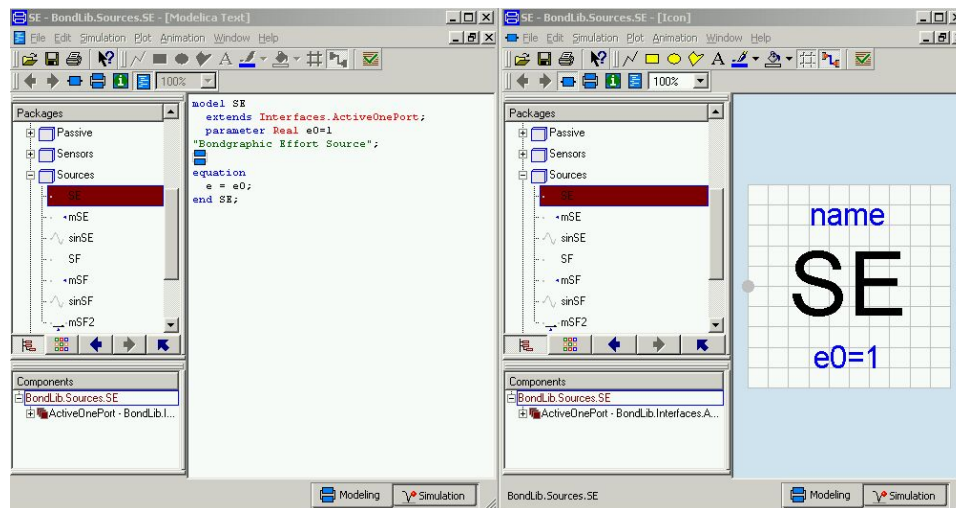


Figure 4.31. Effort Source Model

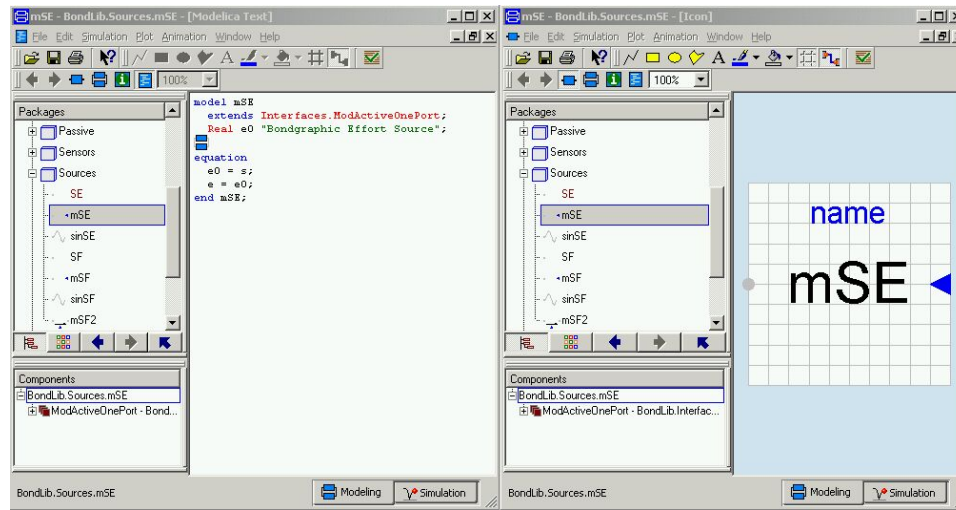


Figure 4.32. Modulated Effort Source Model

For completeness, the flow sources are shown in figures 4.33 and 4.34.

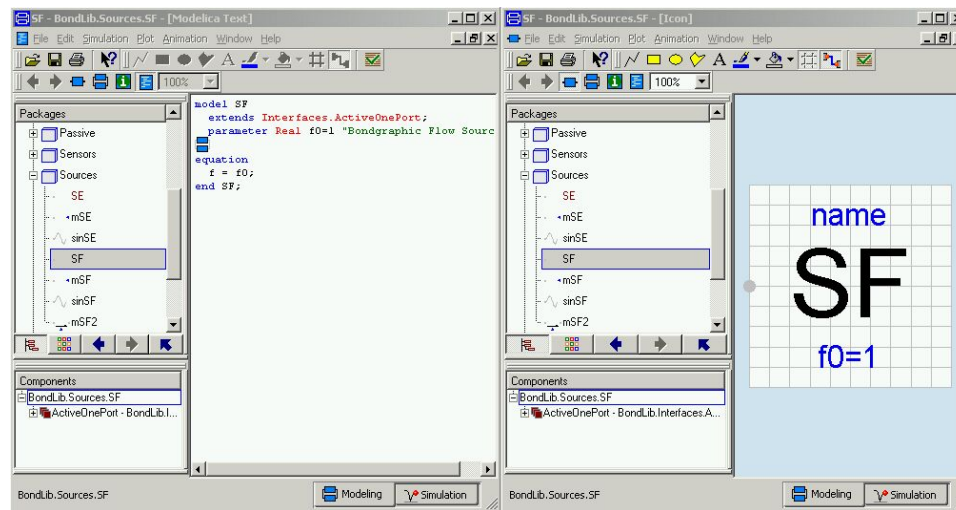


Figure 4.33. Flow Source Model

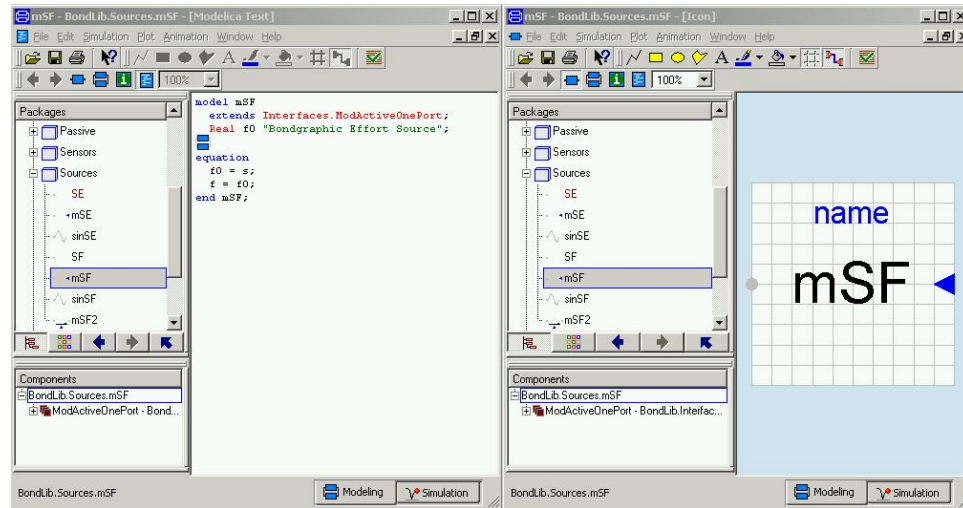


Figure 4.34. Modulated Flow Source Model

Bond graph source elements have a required causality associated with them. The source models in the bond graph library inherit their causality from the bond connected to them, and they define causality by setting either the e variable, or the f variable, for effort sources, and flow sources, respectively. Thus, if the source element is connected to the incorrect bond, an error will be generated at the time the model is compiled.

4.3.6 Sensors

There are a few sensor models in the bond graph library; an effort sensor, flow sensor, P sensor, and Q sensor. Aside from these four models, additional power sensor models are used extensively in this dissertation.

The effort and flow sensor models are similar to passive element models in that they rely on *PassiveOnePort* for the connector inheritance. The input to these models comes

from the bond graph 3-tuple, and the output is a signal generated by either the e variable or f variable. Figures 4.35 and 4.36 show the effort sensor and flow sensor, respectively.

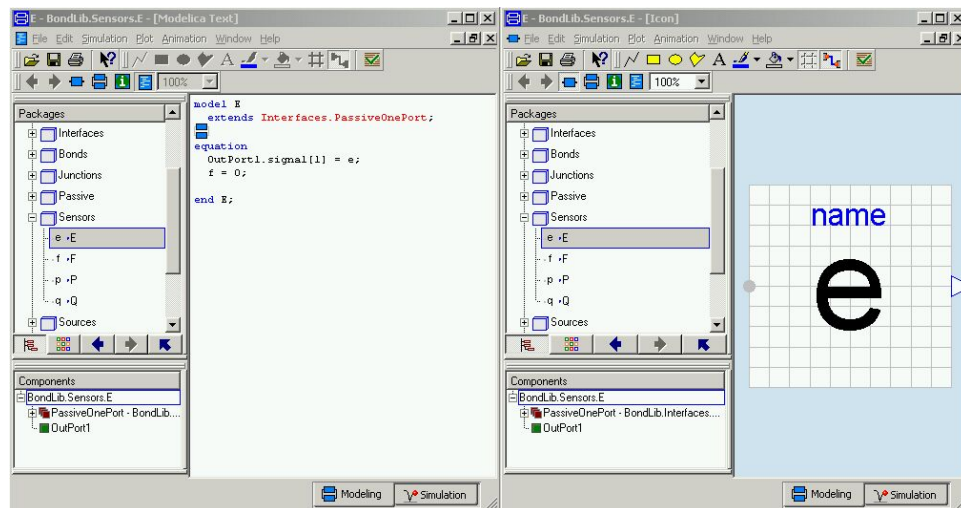


Figure 4.35. Effort Sensor

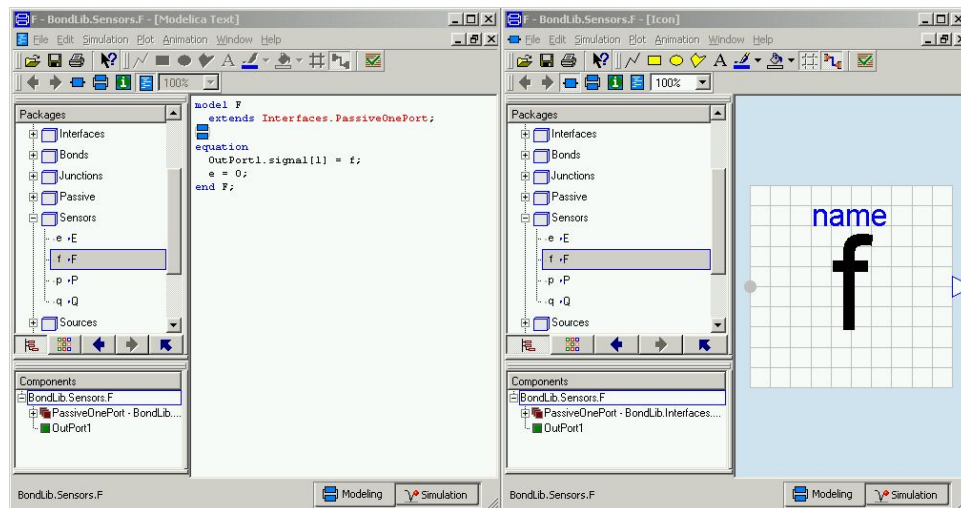


Figure 4.36. Flow Sensor

The P and Q sensor are constructed in a different manner. Up until now, the models in the bond graph library have not used the diagram window. All of the information needed to construct the model was either inherited or explicitly stated in the equation window. The icon window has been used to show the iconic representation of the model at higher hierarchical levels. This window usage is not so for the P and Q sensors. These models are constructed exclusively in the diagram window, with the exception of the *PassiveOnePort* inheritance statement in the equation window. The sub-models used to create the P and Q sensor models all come from the bond graph library and are discussed above. Figures 4.37 and 4.38 show the P and Q sensor models, respectively. In these two figures, however, the equation window is replaced with the diagram window. The integrator, used on the output signal in both models, comes from the *Modelica.Blocks.Continuous* library that comes standard with the Dymola software.

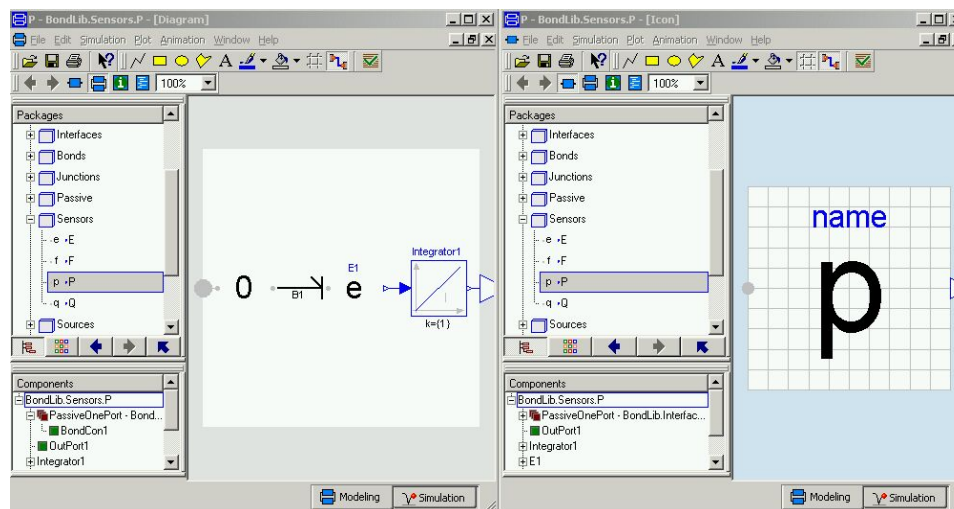


Figure 4.37. P Sensor

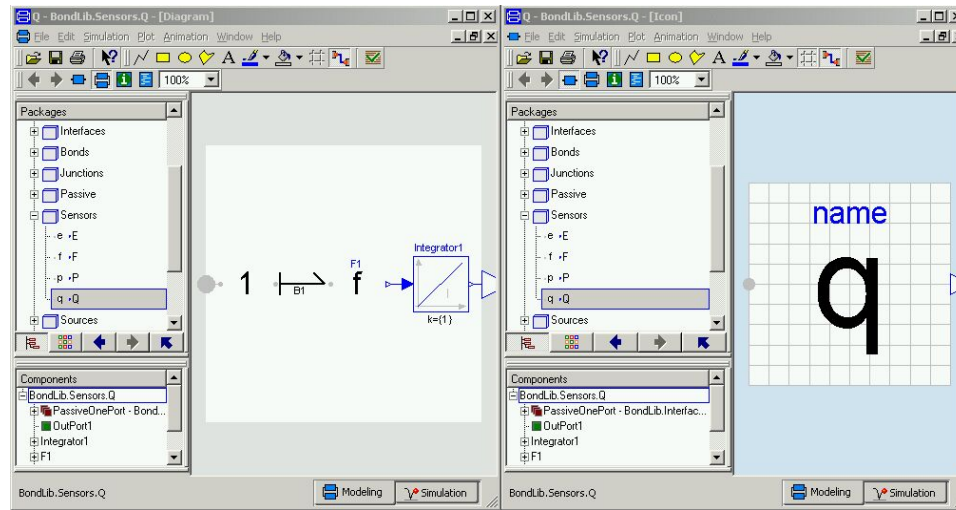


Figure 4.38. Q Sensor

Naturally \dot{P} is an effort, and \dot{Q} is a flow. Thus, an effort sensor and flow sensor were used in these models. The 0/1-junctions, together with the bonds, ensure that the sensor models provide the proper causality to the model that they are connected to.

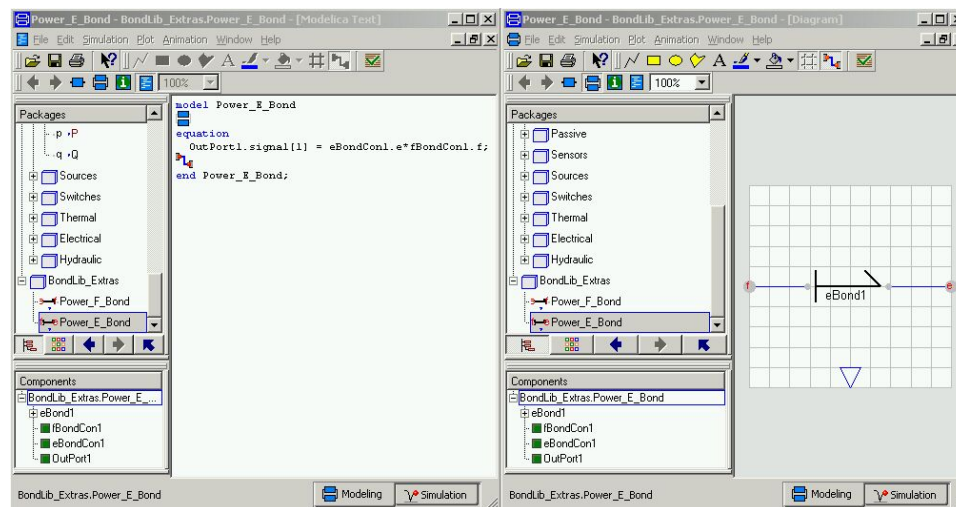


Figure 4.39. Power Sensor on an e -Bond

The power sensor models use all three windows. These models are intended to sense the power flow through a bond. Thus, there are two power sensor models; one for an e -bond model and one for an f -bond model. The power sensor for the e -bond model is shown in figure 4.39.

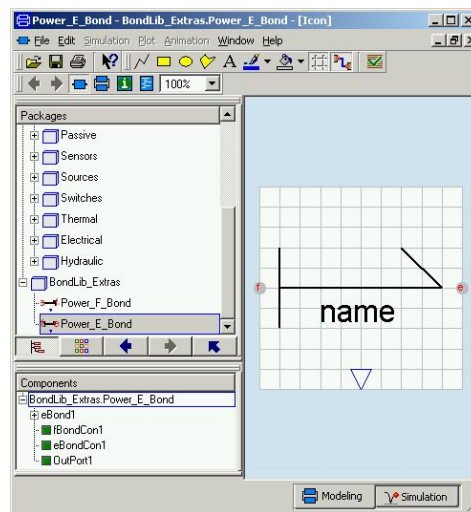


Figure 4.40. Power Sensor on an e -Bond: Icon Window

Figure 4.39 shows the equation window on the left and the diagram window on the right. The icon window for this model is shown in figure 4.40. The power sensor for the f -bond can be seen in figures 4.1-4.3. These figures were introduced earlier, as an example of the three different windows used by Dymola to describe a model. Naturally, the power sensor models act the same as the bond models, since the bond model is the only thing connecting input to output.

These models can now be used to create large bond graph models. The large bond graph models themselves can be used in an object-oriented fashion to create very complex models.

4.4 A Gyroscopically Stabilized Platform: An Object-Oriented Bond Graph

Example

This section uses the bond graph library to create a model of a gyroscopically stabilized platform. This model has many levels built up from the bond graph library. The gyroscopically stabilized platform serves as a good example of how the bond graph library, used in an object-oriented manner, can create very complex systems [McB03].

4.4.1 The Gyroscope Model

The bond graph of figure 4.41 is the gyroscope model of Chapter 3. Here the bond graph has been constructed using the bond graph library of the previous section. Figure 4.41 looks no different than a bond graph generated by any other bond graph drawing tool. There is a major difference, however. The model can be dropped into larger models as an object. Also, the Dymola framework used to create the models knows nothing of bond graph modeling. This independence adds a degree of flexibility to the modeling environment.

The gyroscope model of figure 4.41 is a very complicated model. This model takes signal inputs from outside the model and converts them into effort sources using the

modulated effort source model. There are two modulated transformer elements with modulation signals of $\sin(\theta)$ and $\cos(\theta)$. The signal θ is calculated by integrating the flow off of the $\dot{\theta}$ 1-junction. This detail is done in the equation window since the diagram window is very busy.

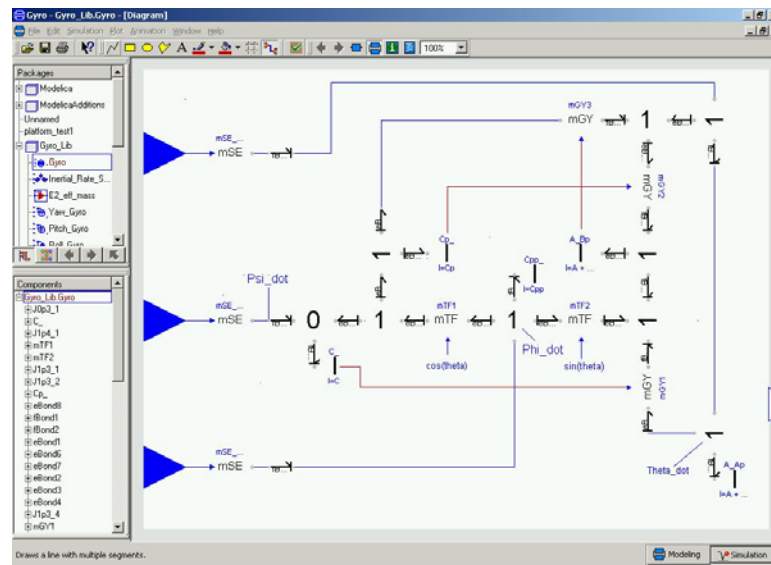


Figure 4.41. Gyroscope Model

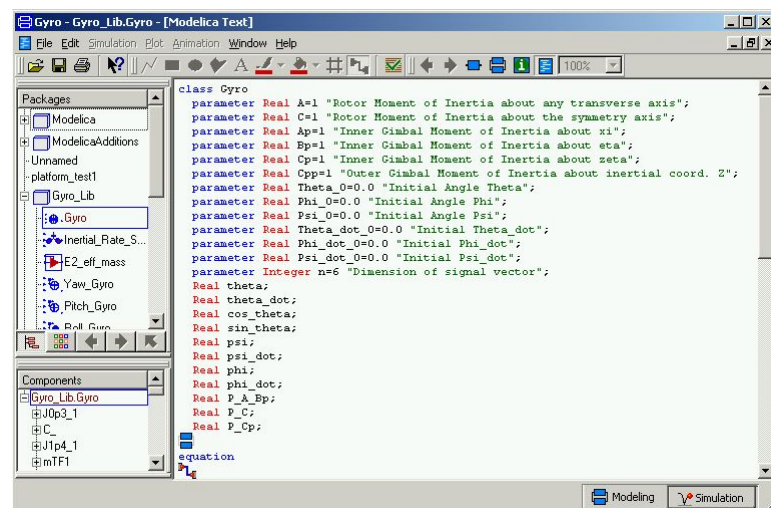


Figure 4.42. Gyroscope Model: Equation Window (A).

Figure 4.43. Gyroscope Model: Equation Window (B).

Also done in the equation window is the output vector signal definition. The large amount of code in the equation window prevents it from being included as a single figure. Two figures have been created. Figure 4.42 shows the parameter and variable declarations and figure 4.43 shows the equations. The parameter and variable declarations simply define all of the variables needed to run the model.

The code in the *when initial() then* clause initializes the states of the system. This section of code is included to add the ability to initialize the states of the gyroscope model to values other than zero.

The next section equates the variables *theta_dot*, *phi_dot*, and *psi_dot* to their appropriate bond graph values. The *der* function is used to integrate these values to get

the variables θ , ϕ , and ψ . The variables ϕ , and ψ are used for outputs only. The variable θ is an output, but it is also used as an input to the modulated transformers as mentioned above.

The variables P_{A_Bp} , P_{Cp} , and P_C are the momentums off of the A_Bp , Cp , and C , I -elements, respectively. These momentums are used as input signals to the modulated gyrotor elements $mGY3$, $mGY2$, and $mGY1$, respectively. The output vector is a 6-tuple defined as $\langle \theta, \dot{\theta}, \phi, \dot{\phi}, \psi, \dot{\psi} \rangle$. This discussion completes the dynamics of the equation window. All other gyroscope dynamics are contained in the bond graph model of the diagram window. The gyroscope model icon is shown in figure 4.44.

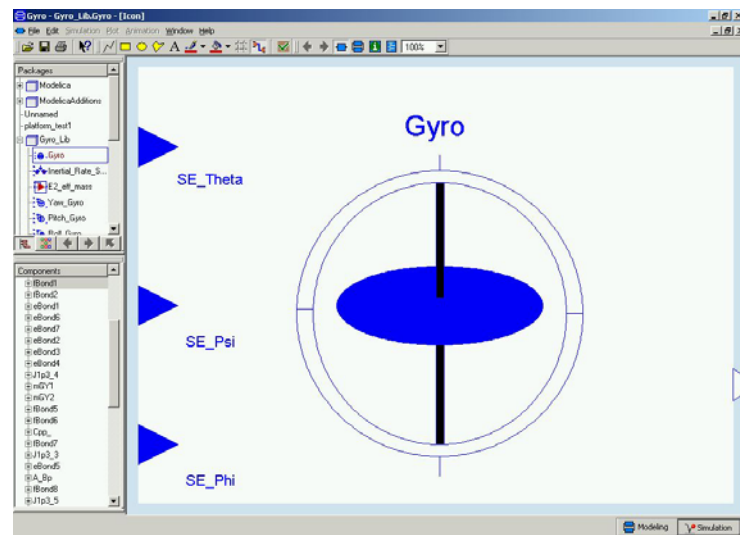


Figure 4.44. Gyroscope Model: Icon Window.

4.4.2 Inertial Rate Sensor Model

Inertial pitch, yaw, and roll rates can be sensed from three gyroscopes by orienting the gyroscope models in three different directions. The above gyroscope model can be used in an object-oriented fashion in three different instances to accomplish this task.

4.4.2.1 Pitch Gyro

Figure 4.45 shows the gyroscope model of figure 4.41 oriented to sense pitch motion. The signals labeled SE_Roll , SE_Pitch , and SE_Yaw are torques that move the platform body. The gains scale these torques to the appropriate value such that, after scaling, the inputs are the effective torques that act on the gyroscope body.

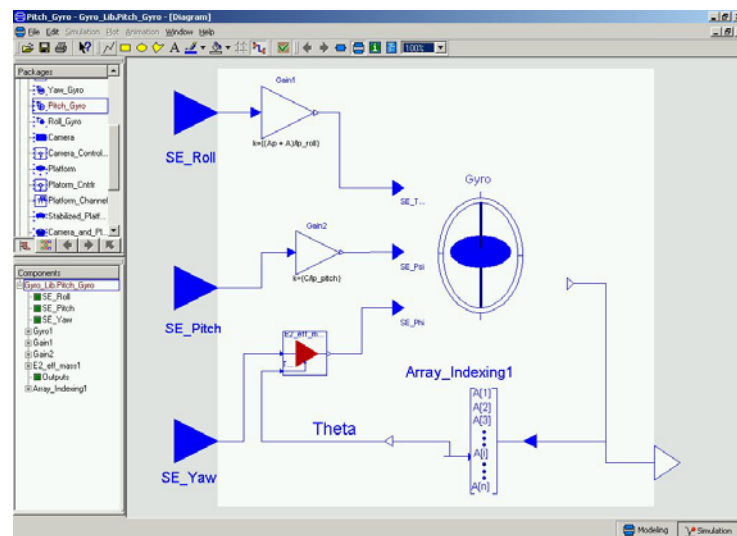


Figure 4.45. Pitch Gyroscope

The gain on the SE_Yaw torque is more complicated than just a gain. The formula for scaling the yaw torque is

$$Gyro_Yaw_Torque = \frac{Platform_Yaw_Torque}{I_{Platform_{Yaw}}} \left[(A + B') \sin^2(\theta) + C' \cos^2(\theta) + C'' \right] \quad (4.42)$$

A , B' , C' , and C'' are gyroscope inertia values and θ is a gyroscope Euler angle, as described by Section 3.3.4. $Platform_Yaw_Torque$ is the SE_Yaw input, and $I_{Platform_{Yaw}}$ is the platform, yaw moment of inertia. This same formula is shown in figure 4.46 with the icon window to the left and the diagram window to the right.

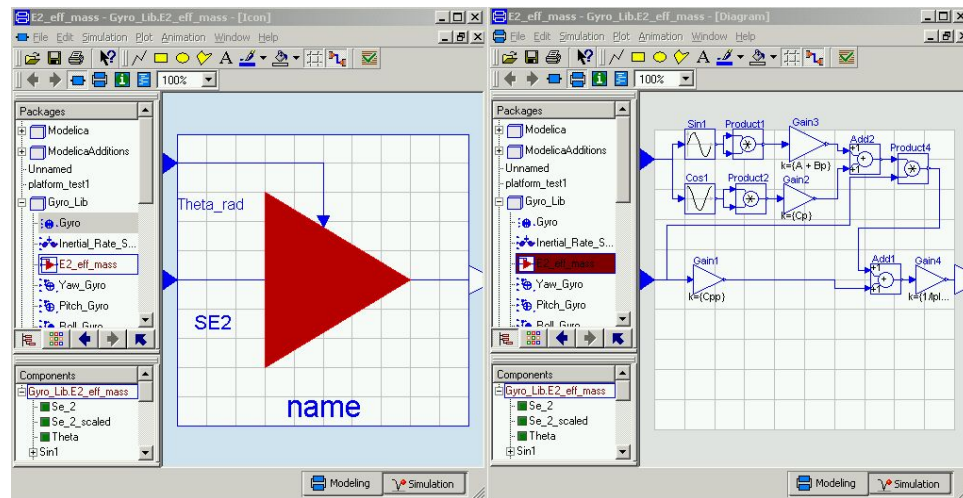


Figure 4.46. Effective Inertia

The instantiated effective inertia model in figure 4.45 has been flipped about the horizontal axis, thus the gyroscope variable θ enters nearer the bottom of the icon.

This scaling of inertias is a byproduct of the simulation environment. In the physical world, the gyroscope would be strapped to the platform body. Thus, any torques *felt* by the gyroscope would naturally have the appropriate magnitude.

The gyroscope in figure 4.45 is oriented such that the Euler angle rate $\dot{\psi}$ is a measurement of inertial pitch rate. By monitoring this angular rate, and integrating, the gyroscope can measure the pitch, and pitch rate of the platform body. The icon window for the pitch gyro model is shown in figure 4.47.

The model labeled *Array_Indexing1* is simply a de-multiplex model. The input to this block is the 6-tuple $\langle \theta, \dot{\theta}, \phi, \dot{\phi}, \psi, \dot{\psi} \rangle$ from the gyro and the output is θ .

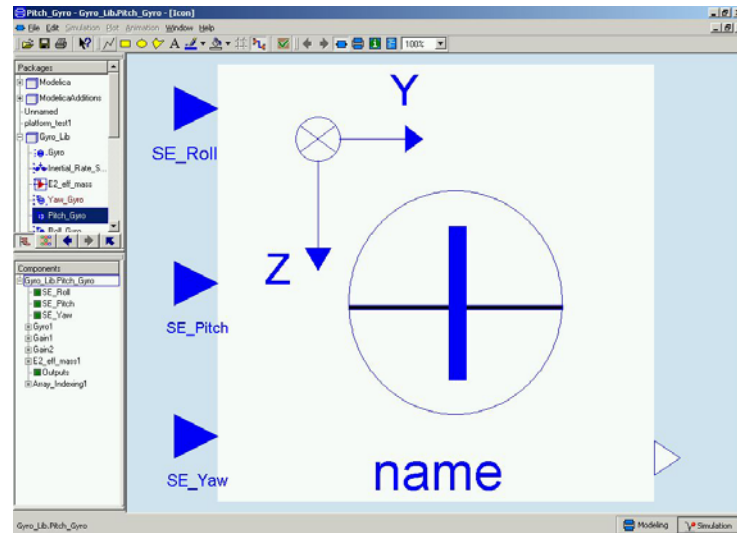


Figure 4.47. Pitch Gyro Icon Window

4.4.2.2 Yaw Gyro

Figure 4.48 shows the gyroscope model oriented such that the platform yaw motion can be measured by monitoring the angle θ . Figure 4.49 shows the corresponding icon window. Here, the effective inertia calculations are identical in form to those shown in Section 4.4.2.1. The appropriate roll, pitch, and yaw platform inertias are associated with

the signal values. Thus, the $IPlatform_{Yaw}$ value in equation 4.42 becomes $IPlatform_{Pitch}$ in this model.

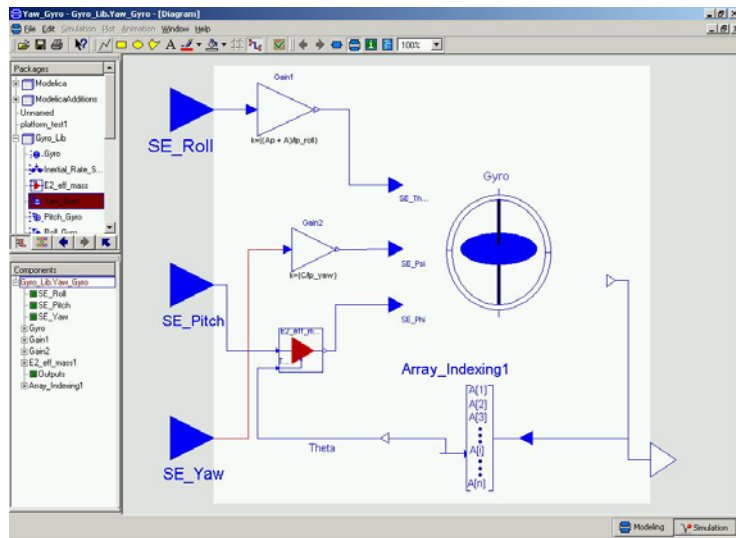


Figure 4.48. Yaw Gyroscope

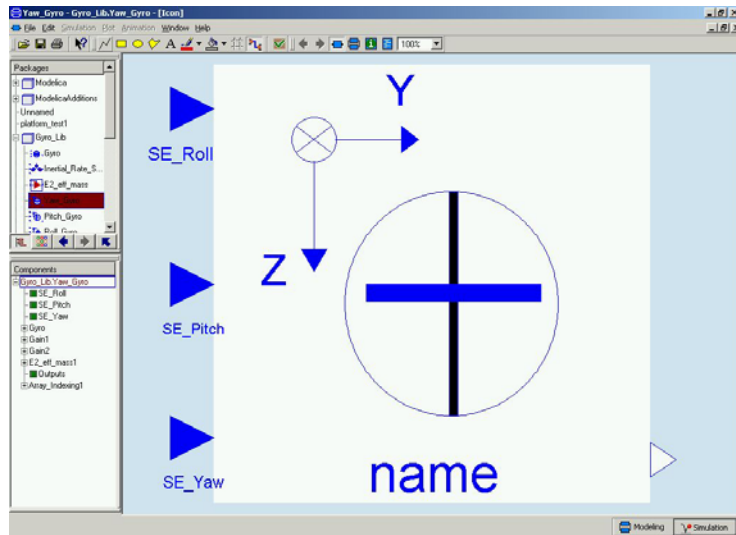


Figure 4.49. Yaw Gyro Icon Window

4.4.2.3 Roll Gyro

Figure 4.50 shows the gyroscope model oriented to sense inertial roll motion of the platform. The corresponding icon window is shown in figure 4.51.

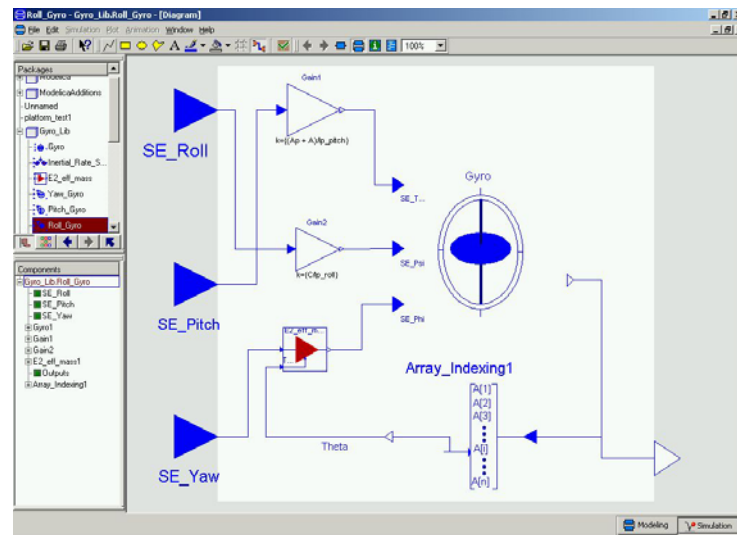


Figure 4.50. Roll Gyro

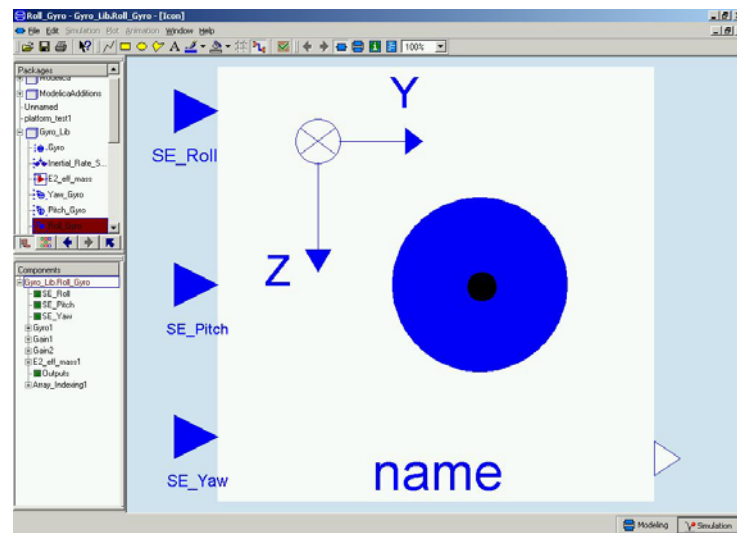


Figure 4.51. Roll Gyro Icon Window

Similar to the previous gyro orientation models, the form is identical but the inertia parameters, and the torque signals, are connected in a roll orientation.

The complete inertial rate sensor model is created by dropping the pitch, yaw, and roll gyro models into a single model. This model is shown in figure 4.52.

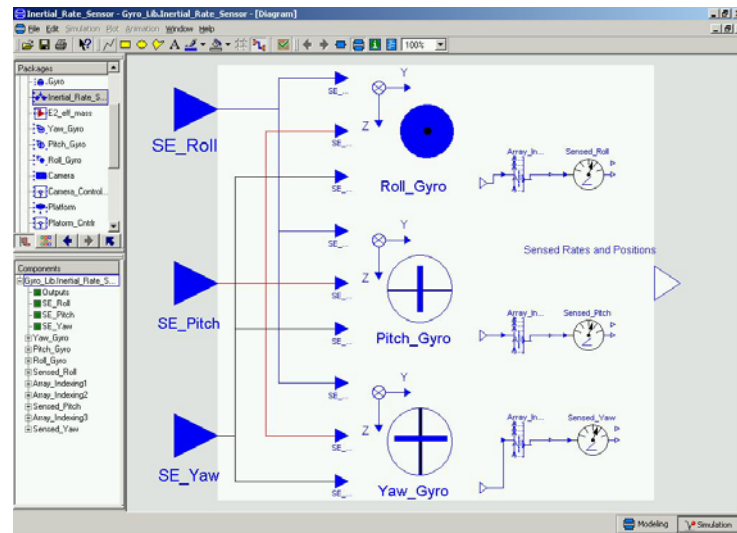


Figure 4.52. Inertial Rate Sensor Model

The three *Array_Indexing* models de-multiplex (demux) the gyro 6-tuples to output the last element, which corresponds to the Euler angle rate $\dot{\psi}$ of each gyro. These three signals are sent through a sensor block. The sensor block simply consists of an integrator and sensor delays. The model is setup to subtract off initial conditions, if needed. This model is shown in figure 4.53 with the icon window on the left and diagram window on the right. As seen in figure 4.53, both the rate and angle are outputs of the sensor delay model.

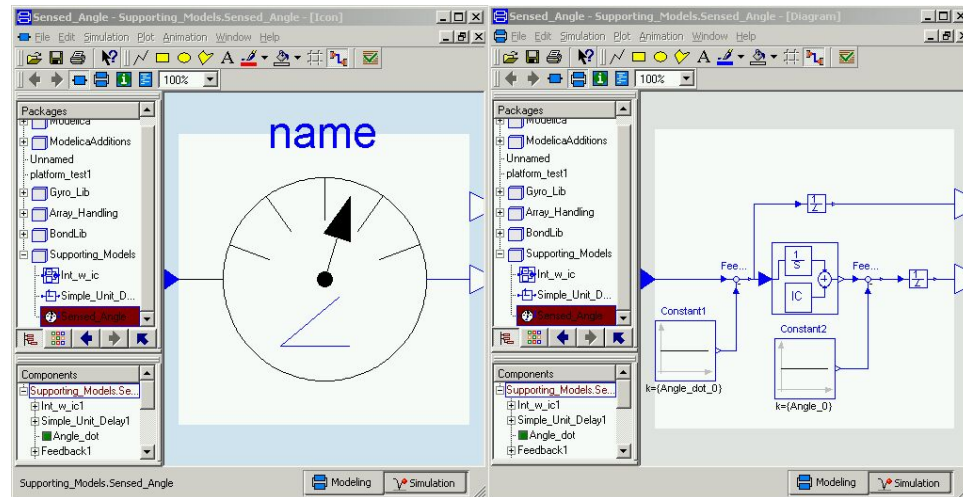


Figure 4.53. Sensor Delays

The output of the inertial rate sensor model is again a 6-tuple. The signals are the sensed rates and angles of the three gyros, i.e., \langle sensed roll rate, sensed roll angle, sensed pitch rate, sensed pitch angle, sensed yaw rate, sensed yaw angle \rangle . The icon for the inertial rate sensor model of figure 4.52 is shown in figure 4.54.

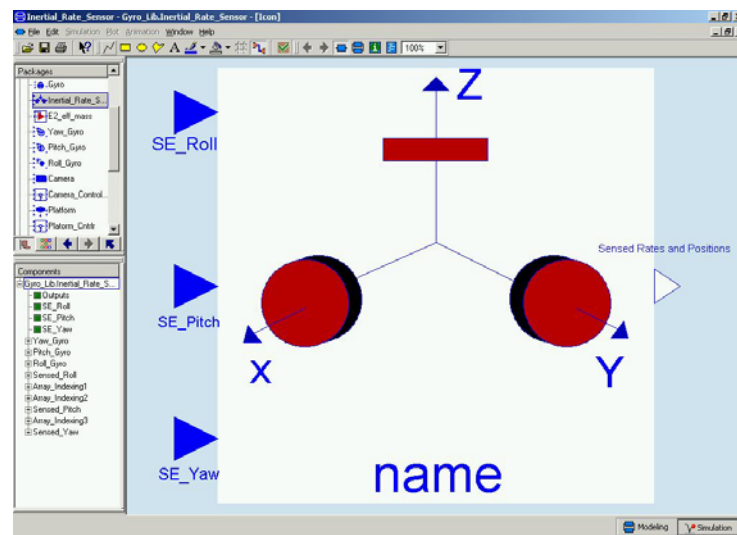


Figure 4.54. Inertial Rate Sensor Model Icon

The inertial sensor rate model is now ready to be used in a larger model. By attaching the above model to a platform model it is possible to sense the platform roll, pitch, and yaw degrees of freedom.

4.4.3 Platform Model

The platform model for this example is a very simple model. The platform model is kept very simple, since the focus of the platform example is the use of the gyroscope model as an object-oriented bond graph.

The platform model consists of roll dynamics, pitch dynamics and yaw dynamics. For simplicity, these models are independent of one another. Each of these three channels is modeled with a simple bond graph consisting of an effort source, and platform body inertia. This model is shown in figure 4.55

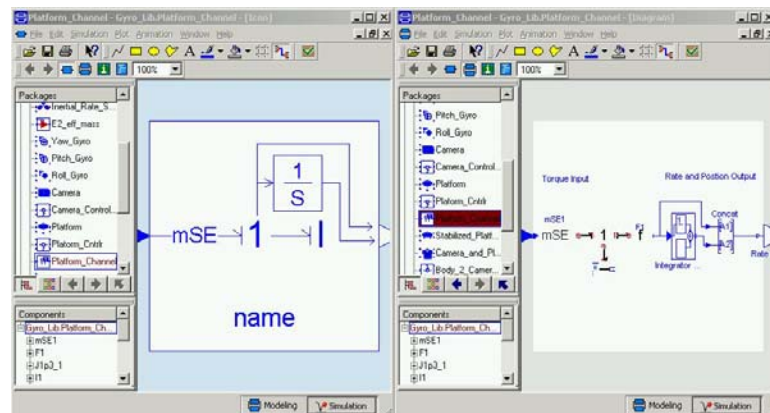


Figure 4.55. Platform Channel Bond Graph

As seen in figure 4.55, the platform channel bond graph is very simple. The torque signal is converted to a bond graph value through the mSE model and the output is the flow off of the 1-junction. The 1-junction flow and position are the outputs of the model. The icon labeled *Concat* is a multiplex model that makes a 2-D vector from two 1-D vectors.

Three instances of the platform channel bond graph are used to model the complete platform body. This model is shown in figure 4.56.

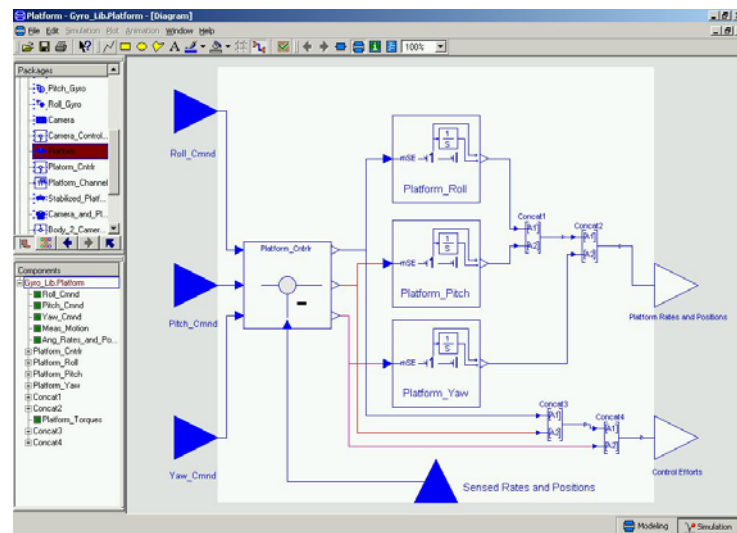


Figure 4.56. Platform Body

The outputs of this model consist of a vector of the three body rates and positions, and a vector of the control efforts used to control the body motion. The control efforts output from this model are input to the mSE elements of the platform channel bond graphs. As

such, the control efforts are the body torques used to move the platform body. The icon of the platform body model is shown in figure 4.57.

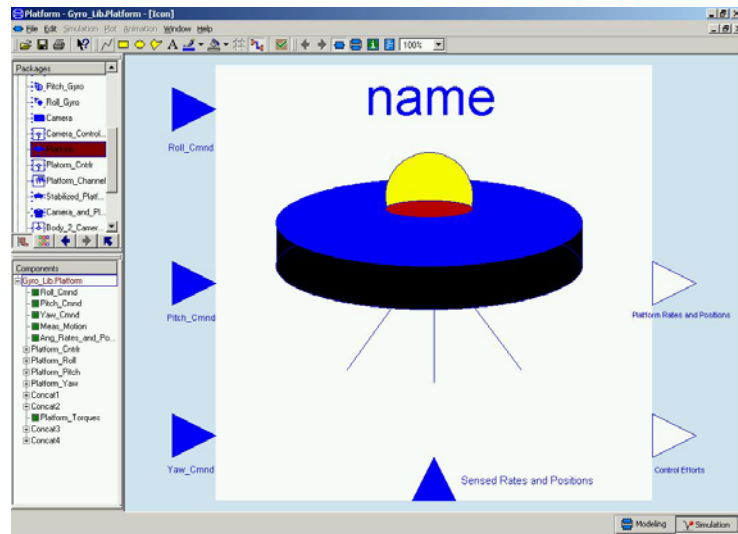


Figure 4.57. Platform Body Icon

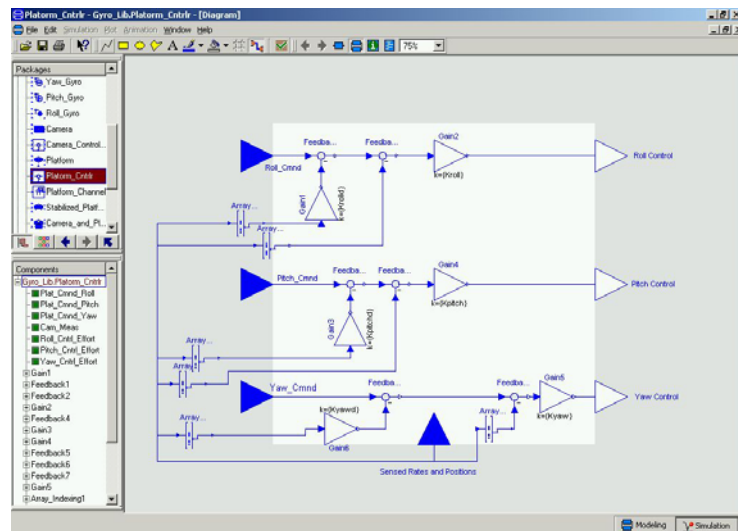


Figure 4.58. Platform Body Controller

The icon labeled *Platform_Cntrlr* is the body controller. This model is a simple rate and position, proportional feedback controller to control the body position. This model is shown in figure 4.58. The platform model is now ready to be connected to the inertial rate sensor model.

4.4.4 Stabilized Platform

The platform and rate sensor models are combined to form a stabilized platform model. This model is shown in figure 4.59.

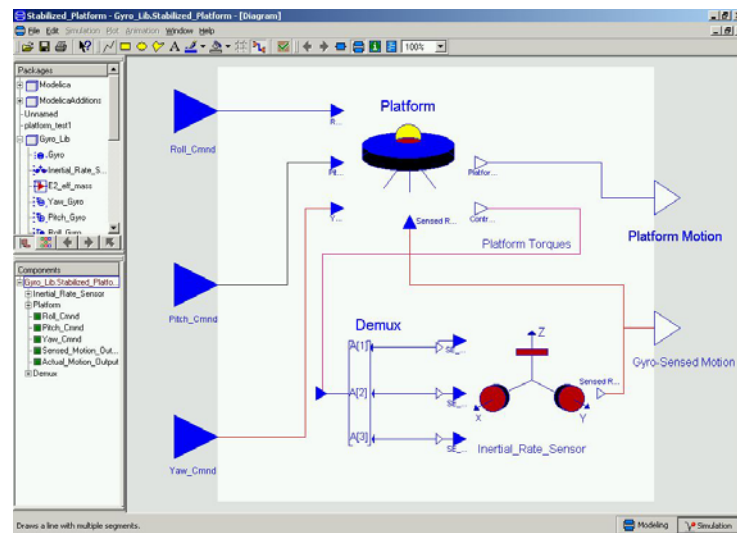


Figure 4.59. Stabilized Platform

The inputs to this model are body position commands. The outputs consist of two vectors: first the true body rates and position, and second the sensed body rates and positions. As can be seen by the diagram window of figure 4.59, the gyroscope model

outputs are used as feedbacks to the platform model to stabilize the position. The icon of the stabilized platform is shown in figure 4.60.

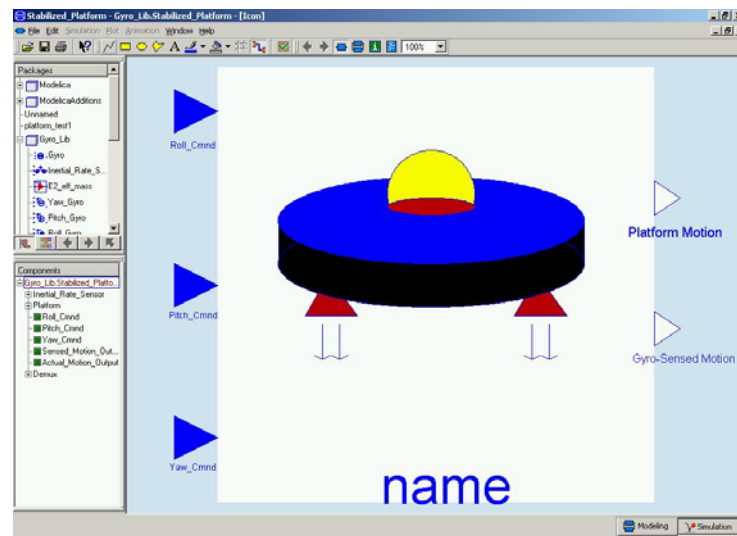


Figure 4.60. Stabilized Platform Icon

This completes the stabilized platform model. The stabilized platform uses three instances of the gyroscope model, and three instances of the platform channel model. The bond graph library forms the basis for the stabilized platform model. The controller models, and supporting mathematics models, are built from the Dymola standard libraries.

4.4.5 Camera Model

Another layer of complexity is added to the stabilized platform model. A gimballed camera is attached to the platform body. The camera sits on gimbals in order to remain

pointed at a fixed inertial point regardless of the platform body motion. The camera model uses another instantiation of the gyroscope model. The dynamics of the camera gimbals are identical to the gimbals of the gyroscope. However, the implementation of these dynamics is very different.

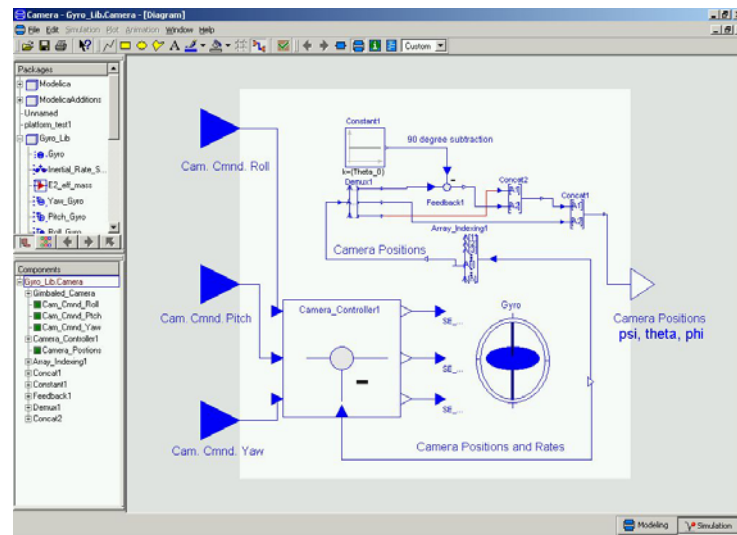


Figure 4.61. Camera Model

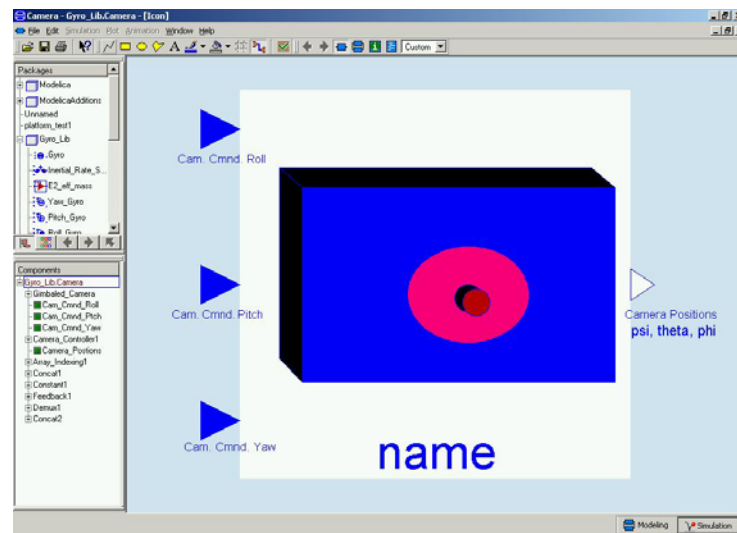


Figure 4.62. Camera Model Icon

The camera model is shown in figure 4.61, and the icon in figure 4.62. Figure 4.61 shows that the camera model dynamics are defined by an instantiation of the gyroscope model. The initial angle on θ is 90° , due to the definition of θ in the gyroscope model. The subtraction on the output signal of θ removes this value.

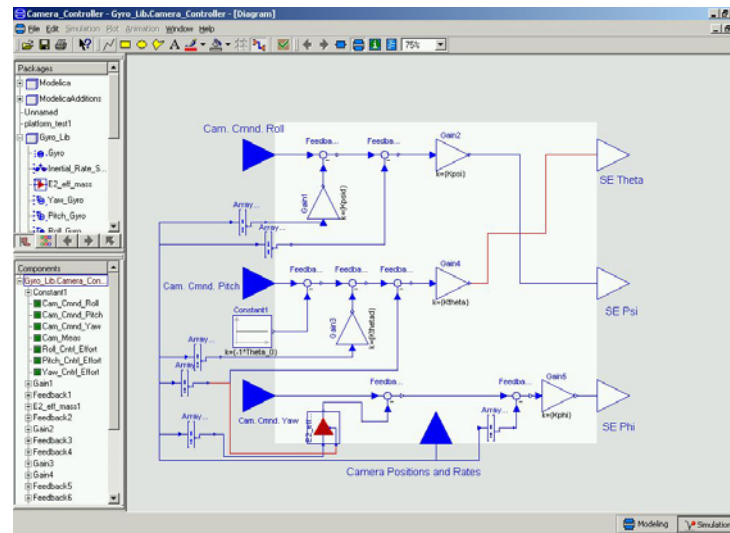


Figure 4.63. Camera Controller

The camera controller model, inside the *Camera_Controller1* icon of figure 4.61, is shown in figure 4.63. The controller has a simple proportional position and rate control law. The inputs from the left are the camera roll, pitch and yaw position commands. The outputs on the right are the θ , ψ , and ϕ torques that will be used in the gyroscope bond graph. The 90° rotation on the variable θ is also seen in the *SE theta* channel. The effective mass model is included in this controller to scale the torque on ϕ appropriately. From figure 4.63, it is apparent that a roll command induces a ψ motion

in the camera, a pitch command induces a θ camera motion, and a yaw command induces a ϕ camera motion. With this orientation, and figure 3.20, it is seen that the camera is oriented in the same fashion as the roll gyro. The camera model is ready to be attached to the stabilized platform.

4.4.6 Stabilized Platform with a Two-Gimbal Camera

The model attaching the camera to the stabilized platform is shown in figure 4.64.

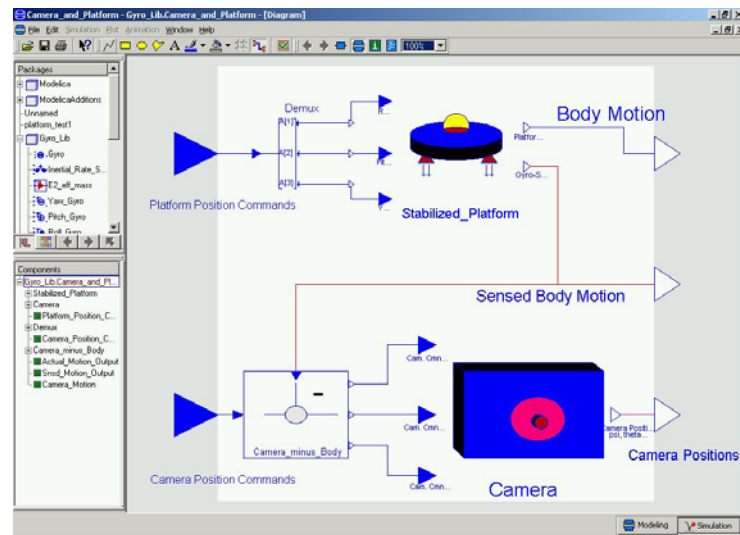


Figure 4.64. Platform and Camera

As seen in figure 4.64, the sensed body motion is fed into a block to subtract the body motion from the camera position commands. In this way, the effective camera position commands remain fixed in inertial space. Since the body motion subtracted off of the camera commands is sensed, it is expected that some error exists between the camera's

idea of an inertial fixed position and true inertial fixed position. This subtraction model is straight forward and shown in figure 4.65.

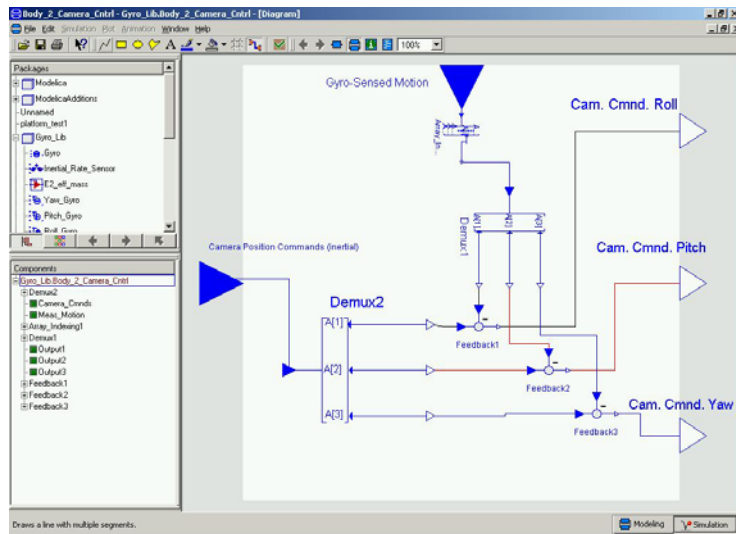


Figure 4.65. Inertial Commands – Body Motion

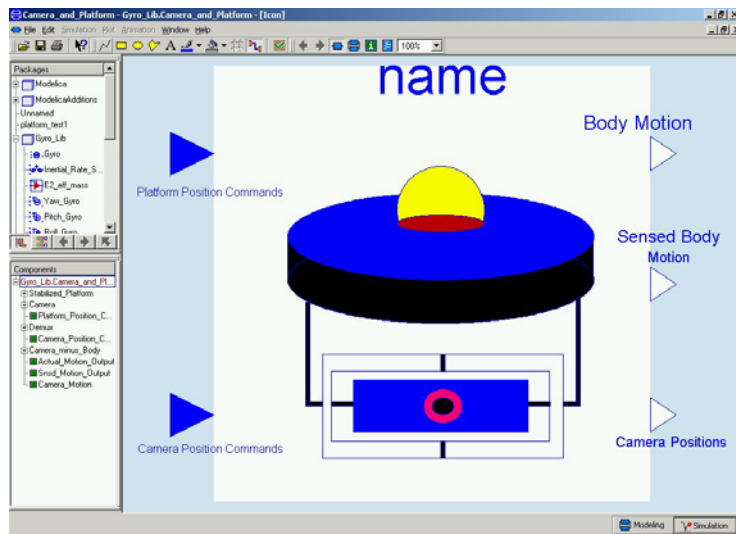


Figure 4.66. Platform and Camera Icon

The icon of the completed model of figure 4.64 is shown in figure 4.66. This camera model contains four instances of the gyroscope bond graph model of figure 4.41. The gyro model is a high fidelity model that includes gimbal inertia effects. Other sub-models are low fidelity models. One of the advantages of working with object-oriented models is that improvements to the fidelity of a core sub-model are reflected in every instance. The camera and stabilized platform model is now complete and ready for simulation.

4.4.7 Simulation and Results

The complete model used for the simulation is shown in figure 4.67.

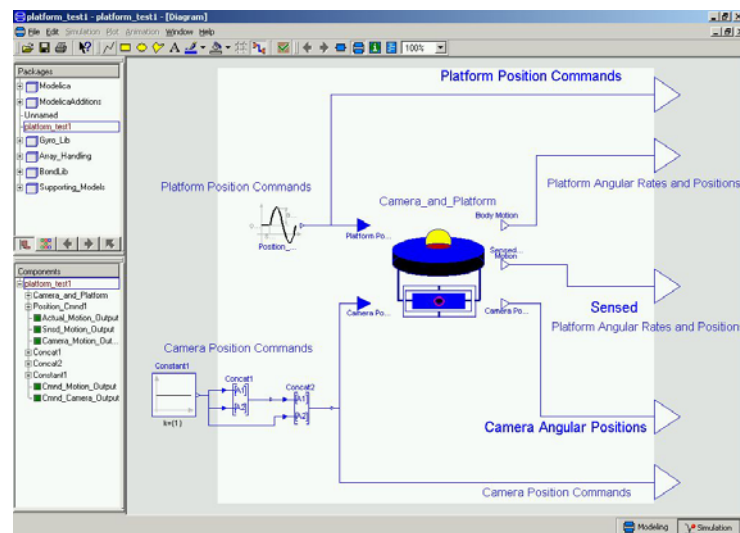


Figure 4.67. Platform and Camera Simulation Model

The simulation is setup to run from 0 to 15 seconds. Table 4.1 lists the inertia parameter values for the camera and gyros used during the simulation. The inertia values were created arbitrarily for this simulation. The inertia values used for the platform are $I_{pitch} = 5E4 \text{ kg}\cdot\text{m}^2$, $I_{yaw} = 4E4 \text{ kg}\cdot\text{m}^2$, and $I_{roll} = 3.5E4 \text{ kg}\cdot\text{m}^2$. The sensor delays were set at 1 kHz.

	Gyros	Camera
A	$400 \text{ kg}\cdot\text{m}^2$	$1800 \text{ kg}\cdot\text{m}^2$
C	$900 \text{ kg}\cdot\text{m}^2$	$3600 \text{ kg}\cdot\text{m}^2$
A'	$40 \text{ kg}\cdot\text{m}^2$	$160 \text{ kg}\cdot\text{m}^2$
B'	$80 \text{ kg}\cdot\text{m}^2$	$320 \text{ kg}\cdot\text{m}^2$
C'	$40 \text{ kg}\cdot\text{m}^2$	$160 \text{ kg}\cdot\text{m}^2$
C''	$75 \text{ kg}\cdot\text{m}^2$	$300 \text{ kg}\cdot\text{m}^2$
$\dot{\psi}_0$	1500 rad/sec	0 rad/sec
$\dot{\theta}_0$	0 rad/sec	0 rad/sec
$\dot{\phi}_0$	0 rad/sec	0 rad/sec

Table 4.1. Camera and Gyro Values used for Simulation

The simulation run consisted of commanding the camera to a specified point and then commanding the platform to move. The camera, initially at 0° about all three inertial axes, was commanded to maintain a constant inertial position of 57.3° about all three

inertial axes. These inertial camera angles are expected to remain unchanged while the platform is commanded to oscillate through a series of maneuvers. The platform was commanded to pitch, yaw, and roll in a sinusoidal motion. This commanded sinusoidal motion is described in figure 4.68. As seen in figure 4.68, roll, pitch, and yaw were commanded to react to different magnitude and frequency sine waves. As seen in figure 4.68 the roll, pitch, and yaw command frequencies are .25Hz, .5Hz and 1Hz, respectively.

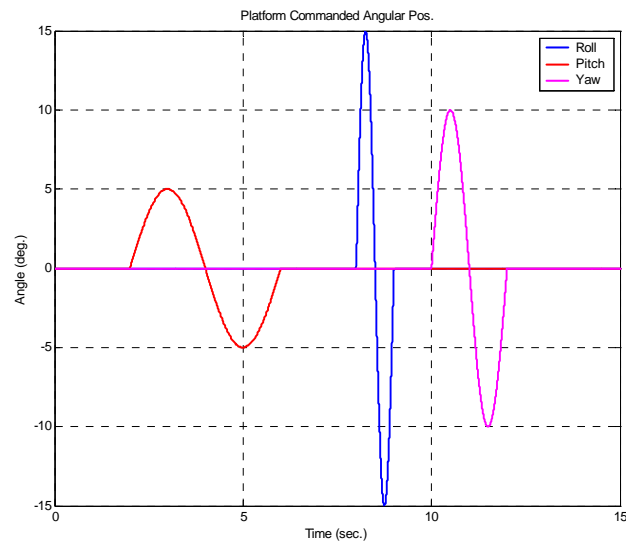


Figure 4.68. Platform Position Commands (deg)

The pitch achieved response is shown in figure 4.69. The response is shown in the top portion of the plot and the error is shown in the bottom portion. As seen in the error portion of figure 4.69, the pitch response matches the command very well with an error of less than 1° .

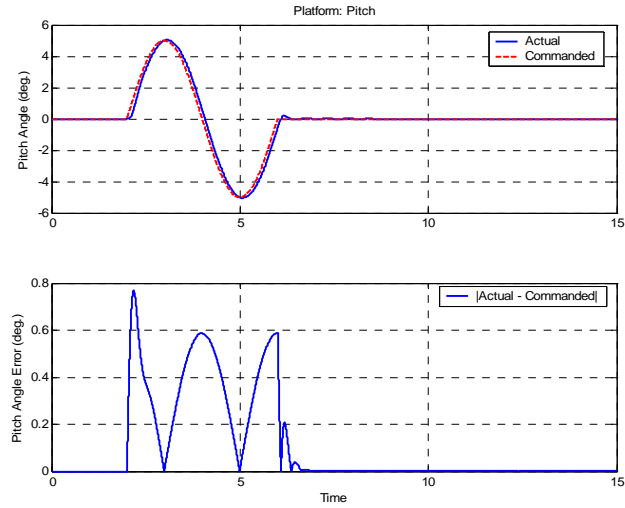


Figure 4.69. Platform Pitch Response (deg)

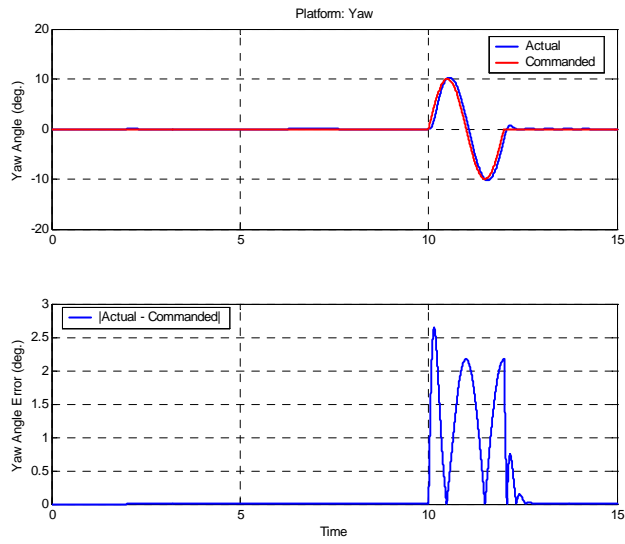


Figure 4.70. Platform Yaw Response (deg)

The yaw achieved response is shown in figure 4.70. As seen in the error portion of figure 4.70, the yaw response generates up to 2.64° error. The frequency of the input command is larger for the yaw response than for the pitch.

The roll achieved response is shown in figure 4.71. The error portion of figure 4.71 shows the roll response generates just over 7° error. The frequency of the roll input command is the largest of the three channels.

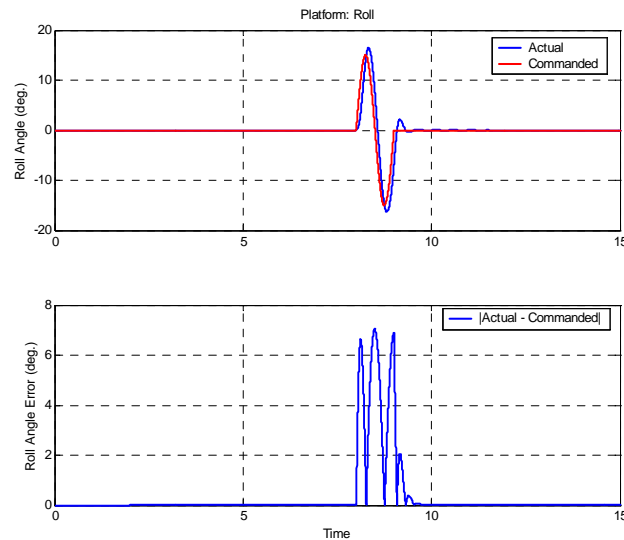


Figure 4.71. Platform Roll Response (deg)

Figure 4.72 compares the sensed positions to the achieved positions. This figure compares all three channels at once to show that the sensed values are very close to the actual. This result is not surprising considering the simplistic sensor model used. Note that the error in the sensed and achieved values follows the same trend as the commanded

and achieved values, i.e., roll, yaw, and pitch are the order of the channels from greatest error to least.

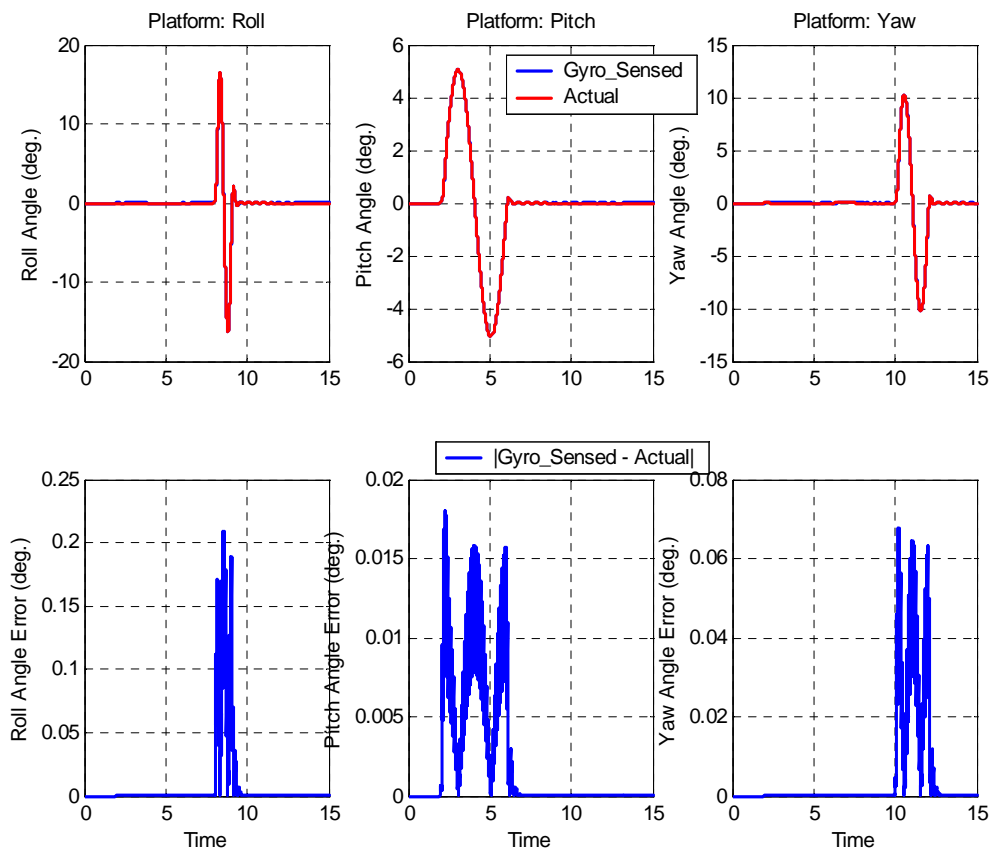


Figure 4.72. Actual and Sensed Achieved Positions (deg)

Figures 4.73 through 4.75 show the camera's response for the three channels. The top portion of each plot shows three separate signals; camera commanded position, camera body position, and the platform achieved position. The bottom portion of each plot shows the camera commanded position minus the camera achieved position minus the platform body motion. This signal represents the inertial pointing error.

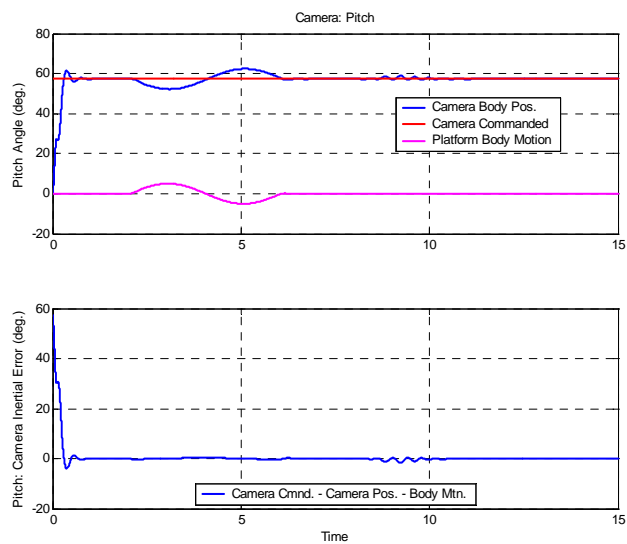


Figure 4.73. Camera Pitch Response (deg)

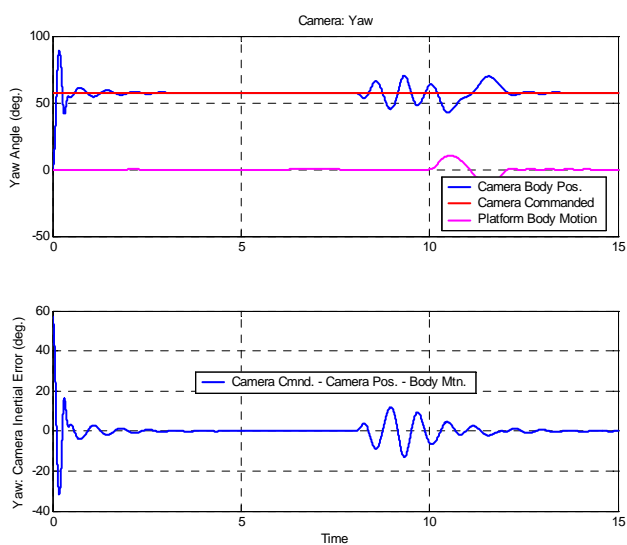


Figure 4.74. Camera Yaw Response (deg)

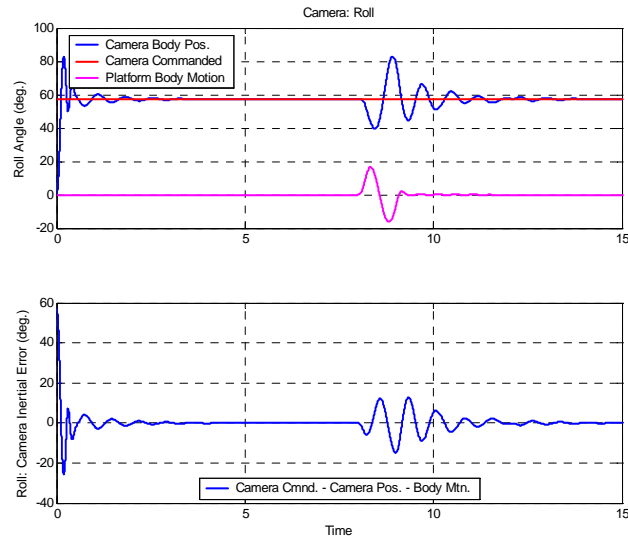


Figure 4.75. Camera Roll Response (deg)

Figure 4.73 shows that the camera pitch response follows the command and subtracts out the platform motion very well. The lower portion of figure 4.73 shows the inertial error which remains close to zero throughout the simulation. Also evident from figure 4.73 is the cross coupling between channels. Camera pitch motion is influenced by yaw and roll motions.

Figure 4.74 and 4.75 show that the cross coupling between the roll and yaw motions is even greater. It is clear from these plots that the simplistic camera control scheme of figure 4.63 needs to include cross coupling terms in the camera roll and yaw channels.

The stabilized platform example has shown that object-oriented bond graph modeling is a useful and powerful technique in the modeling process. The stabilized platform was developed by using four instances of a complex gyroscope model. The ability to create a

bond graph and drop it into a higher hierarchical level adds a degree of flexibility that, to the best knowledge of the author, does not exist in any other simulation framework.

4.5 Conclusions

This chapter has shown that in order to simulate a bond graph model directly, the software must be able to handle both algebraic loops and structural singularities. The Tarjan algorithm and the Pantelides algorithm are quite capable of handling these respective problems. The Dymola software package uses these algorithms to handle these two issues. Also, Dymola is object-oriented in that sub-models can be dropped into models at a higher hierarchical level. Thus, Dymola is a prime candidate for creating a bond graph based simulation environment.

A bond graph library was presented within the Dymola framework. This library takes full advantage of Dymola's ability to sort equations, solve algebraic loops, and handle structural singularities. Also, the object-oriented nature of Dymola provides the ability to use bond graph models, created with the bond graph library, in an object-oriented fashion.

The design and simulation of a system can be done quickly and meaningfully using the tools presented in this chapter. As an example of the ability to model meaningful systems, a complicated gyroscope model, created with the bond graph library, was used in four separate instances to create a gyroscopically stabilized platform model.

CHAPTER 5: System Efficiency Measurement Using the Power Flow Information from a Bond Graph Model

5.1 Introduction

Bond graph modeling is a method of modeling a physical system by mapping the power flow through the system. As shown in Chapter 3, the power flow information of a bond graph can be used to develop the state equations for a given system. However, once the equations of motion are obtained, often the power flow map and the system's causal relationships are discarded. As a result, useful information is lost.

In this chapter, the power flow information of a bond graph is used to develop a method for measuring the efficiency of a system. Here, a method is presented in which the power flow information can be used to monitor the effectiveness of a control scheme. The energy output of a system divided by the energy put into the system gives a feel for the efficiency of the system. Controller effectiveness can be measured by viewing the system's energy response to a given input. In this manner, controllers of different topology can be compared to each other in a meaningful way [McB05c].

In a given system, limitations exist on the amount of power that the system can use at any given time. These limitations can be used to determine the theoretical limit of the system's response. In the case of a control system, often the performance of the system is limited by the thermodynamic bounds of the input power supply. A properly designed controller will make use of the total available energy delivered by the input power supply. Since bond graphs map the power flow through a system, a bond graph model

can be used to monitor the system's thermodynamic response as compared to its theoretical limit. In this way, one can determine if the control system has been properly designed.

This chapter shows, by means of an example of a servo positioning system, a method in which a bond graph model can be used to compare a system's response to its theoretical thermodynamic limit. Separate control schemes are shown and the system responses are compared. A comparison of the effectiveness of the controllers is made, by observing their ability to utilize the power supplied by the motor. By monitoring the power flow through the actuator bond graph, for a given controller, one can get an idea of the controller's ability to effectively use the energy available to the system.

This chapter, however, is not concerned with controller stabilization. The control schemes that are compared by the method presented here are assumed stable. This analysis is meant to serve as a method of comparing the performance of controllers, regardless of the control architecture, given that the controllers to be compared are stable.

5.2 Servo Positioning System

The equations obtained from a bond graph model are identical to equations obtained through other modeling techniques. However, in this research the power flow map is used specifically to monitor the used energy in the system and compare it to how much energy the system could potentially use. Thus, bond graph modeling lends itself naturally to this analysis.

A controller, motor, and load dynamics are shown in figure 5.1. This system represents a fin positioning system used in flight control. The model of figure 5.1, and all models presented in this chapter are created in DYMOLA using the bond graph library as described in Chapter 4. The motor and fin dynamics are objects that contain bond graphs within.

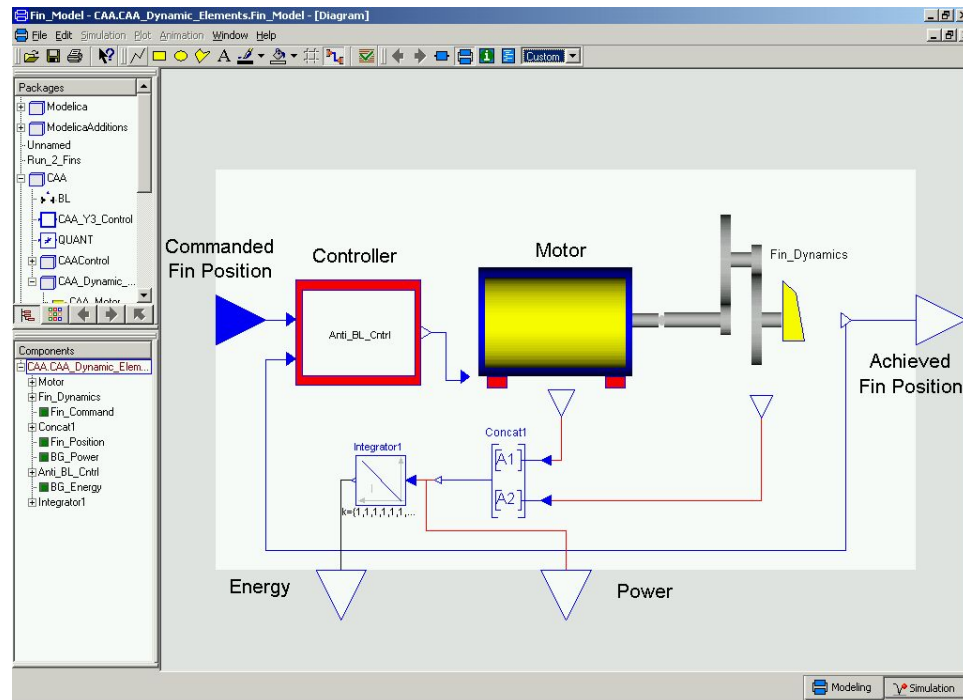


Figure 5.1. Fin Positioning System

Figure 5.1 shows the fin command as an input. The outputs are shown as achieved fin position, energy and power. The output power vector is the power flow through selected bond graph elements of the motor and fin dynamics models. The energy vector is the integral of the power vector.

5.2.1 Servo Positioning System: Complete, Non-Linear System

Inside the motor block of figure 5.1 is a bond graph model of the motor dynamics. The bond graph model of the motor dynamics is shown in figure 5.2.

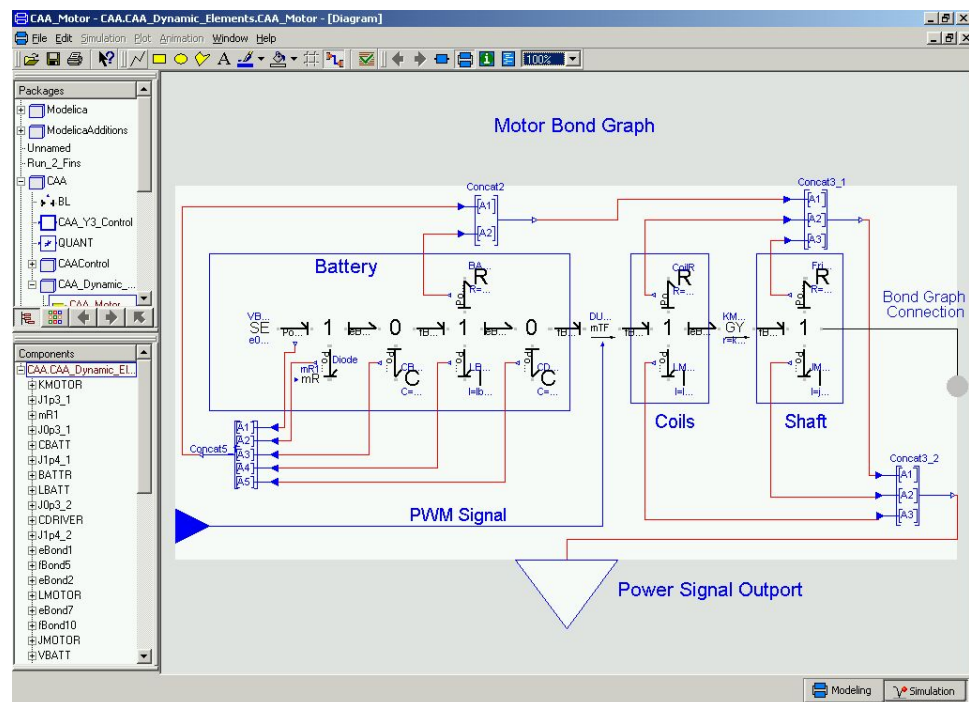


Figure 5.2. Motor Bond Graph

In order to make figure 5.2 more readable, figures 5.3 and 5.4 have been included, which zoom in on the details of figure 5.2.

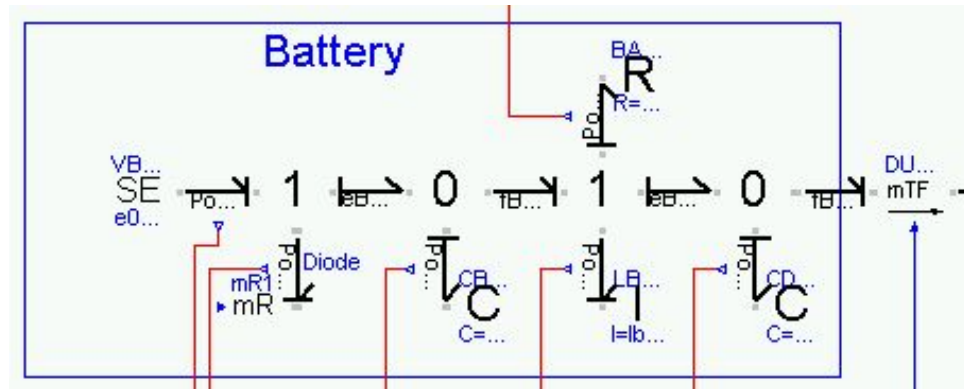


Figure 5.3. Motor Bond Graph: Battery

The motor dynamics consists of three parts; battery dynamics, motor coils, and shaft dynamics. The bond graph model of the motor could have been divided at the output of the coils (at the gyrator), which would have lumped the shaft dynamics into the fin dynamics model. This partitioning would have been a natural choice, if the models were broken by energy domain, separating electrical from mechanical. However, the models were built as shown to represent the physical system. A motor naturally has an electrical input and shaft dynamics as the output. This small detail will later influence the models when the fin dynamics model is linearized.

The battery has two capacitors to keep the voltage near the mTF element as constant as possible [Ther]. The capacitor nearest the mTF element is $cdriver$, the remaining capacitor is $cbatt$. An inductor is placed between them to avoid a structural singularity. The resistor between the two capacitors is a model of the resistive loss through the inductor. The battery has a protective diode that is modeled as a nonlinear resistor. This

diode is seen in figure 5.3 as the modulated resistor element. The logic for this resistance is as follows:

if (Voltage of the capacitor $cbatt >$ Input voltage $e0$) then

$$mRl = rr$$

else

$$mRl = rf$$

end

In the Modelica language, the above logic is represented by the equivalent statement

$$mRl.s = \text{if } vbatt > CBATT.e \text{ then } rf \text{ else } rr;$$

Although this detail is not explicitly shown in the bond graph diagram of figure 5.2, it is stated in the Dymola equation window of the model.

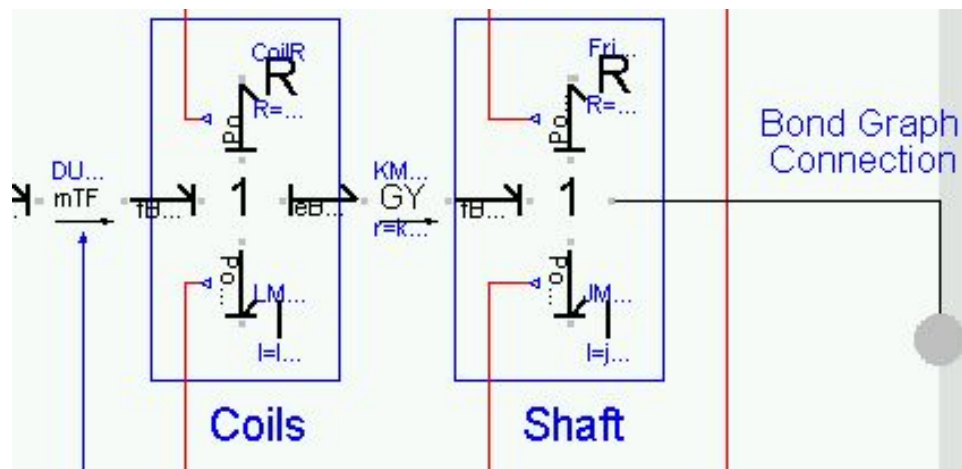


Figure 5.4. Motor Bond Graph: Coil and Shaft

Shown in figure 5.2 is a modulated transformer. This element is used to implement the pulse width modulated (PWM) signal. As seen by the motor bond graph, the PWM signal limits the amount of power flow from the battery to the rest of the servo-system. In this way, the fin can be commanded to a specific position. It is seen in figure 5.2 that no usable power comes from the controller. All of the power supplied to the motor is stored in the battery section. The controller simply limits the power flow through the modulated transformer.

As seen in figure 5.2 the power signals on some of the bonds are passed to the higher levels of the Dymola model. This chapter focuses on the analysis that can be performed by monitoring these power signals. The elements in figure 5.2 labeled *concat* act to multiplex the power signals such that they can be passed to the upper levels of the model through a single vector.

The fin dynamic equations are modeled in the bond graph of figure 5.5. The output of the model is the fin position.

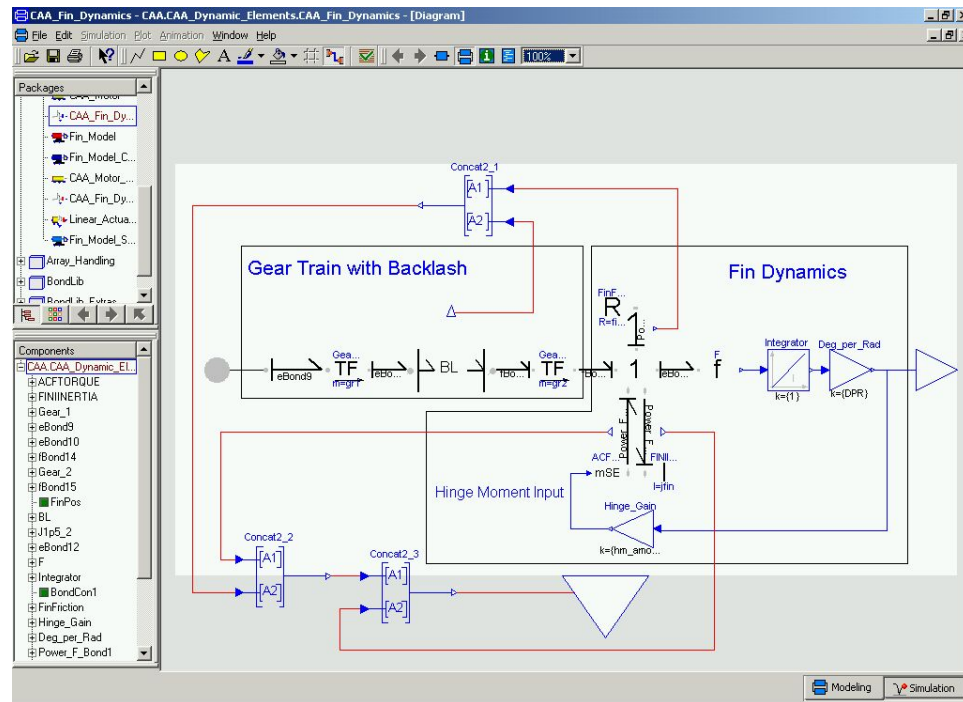


Figure 5.5. Gear Train and Fin Dynamics

The fin position is calculated by integrating the flow off of the 1-junction. The model of figure 5.5 includes a modulated effort source that is used to model the hinge moment torque. This hinge moment torque represents the aerodynamic load on the fin. For simplicity, the hinge moment torque is modeled proportional to the fin position.

Similar to the motor bond graph of figure 5.2, the power signals from specific bonds have been multiplexed and passed to the upper levels of the model to be used in the analysis of the model.

The fin dynamics model of figure 5.5 shows a nonlinear backlash element. The backlash model is expanded and shown in figure 5.6. As seen in figure 5.6, the gear positions on both sides of the backlash model are sensed.

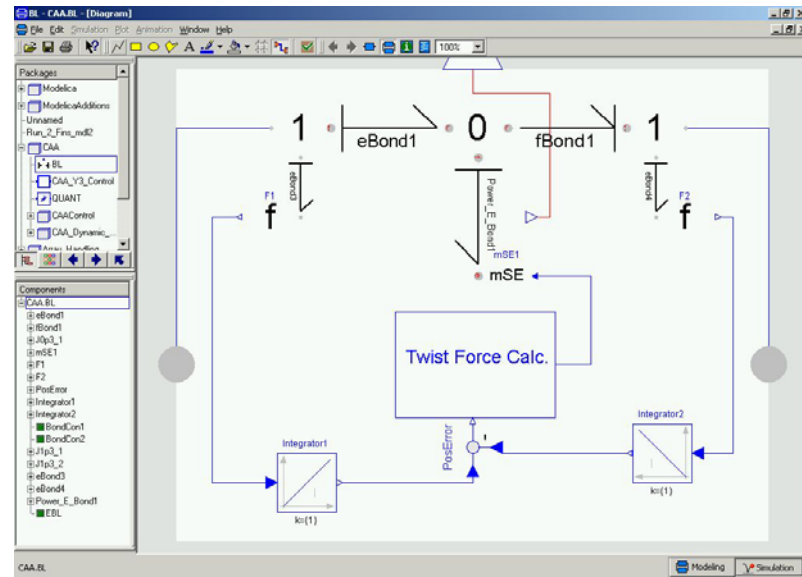


Figure 5.6. Backlash Model

A specified amount of torque is calculated and sent back through a modulated effort source depending on whether or not the gears are in contact. Although the bond graph depicts the torque transmitted as an effort source, it is really modeled as a torsional spring as follows:

if (*position error* > *backlash/2*) then

$$Twist = (position\ error - backlash/2)$$

else if (*position error* < - *backlash/2*) then

$$Twist = (position\ error + backlash/2)$$

else

$$Twist = 0$$

end

The modulated effort source is then assigned a value of $k*Twist$, where k is a torsional spring constant that is set at a stiff value (340 N*m/deg). This nonlinearity causes problems in the position control of the fin. The value of k is passed into the backlash model as KBL denoted in table 5.1. Similarly, the value of $backlash$ is passed in as ABL . Table 5.1 gives a list of values used in the actuator model.

Variable	Parameter	Figure	Value	Description
e0	vbatt	5.3	200	Supply Voltage (Volts)
mR1	rf	5.3	0.001	Forward-Biased Diode (Ohms)
mR1	rr	5.3	500	Reverse-Biased Diode (Ohms)
CB	cbatt	5.3	0.002	Supply Capacitor (Farads)
BA	battr	5.3	2	Supply Resistance (Ohms)
LB	lbatt	5.3	0.001	Supply Inductance (Henrys)
CD	cdriver	5.3	1.00E-05	Driver Capacitance (Farads)
LM	lmotor	5.4	6.00E-04	Motor Inductance (Henrys)
JM	jmotor	5.4	4.50E-07	Motor Inertia (kg*m ²)
KM	kmotor	5.4	1.5	Motor Gyrtator Value (Volts*Sec/Rad)
CoilR	Rcoil	5.4	0.6	Motor Coil Resistance(Ohms)
Fri	shaft_friction	5.4	0.3	Shaft Friction (N*m*Sec/Rad)
gr1	gr1	5.5	0.02	Gear 1 Ratio
gr2	gr2	5.5	0.25	Gear 2 Ratio
jfin	jfin	5.5	3.00E-04	Fin Inertia (kg*m ²)
Hinge Gain	hm_amount	5.5	-0.6	N*m/(Deg of fin deflection)
FinF	fin_fric	5.5	0.04	Fin Friction (N*m*Sec/Rad)
k	KBL	5.6	334	Backlash Spring K (N*m/Deg.)
$backlash$	ABL	5.6	0.025	Amount of Play in Backlash (Deg.)

Table 5.1. Model Parameter Values

5.2.2 Servo Positioning System: Complete, Linearized System

Bond graph models can be linearized by changing the nonlinear bond graph terms to linear terms. This linearization can help in the analysis of the system in that analysis can first be applied to the linear system and then extrapolated to the nonlinear system. This

section focuses on the linearization of the fin positioning system presented in the previous section.

5.2.2.1 Linearized Fin Dynamics

The dominant nonlinearity of the fin positioning system, described above, is the presence of backlash. By noting the structure of the bond graph in figure 5.6, it is seen that the backlash nonlinearity can be replaced by a bond graph C element, where the C element has the value of $1/KBL$. The mechanical implication of doing this substitution is that the gears are always in contact, yet the gear teeth have some compliance in them. The resulting fin dynamics bond graph is shown in figure 5.7.

Note that the causal marks on the bond graph of figure 5.5 are essentially the same as those in figure 5.7. Integral causality is maintained on all elements by choosing this method of linearizing the gear backlash. A potential drawback of this method, however, is that the stiff spring constant KBL used to calculate the compliance of the gear teeth may place a pole far to the left of the rest of the poles in the system creating a *stiff* system. Stiff systems cause difficulty in model simulation. “Stiffness occurs when some components of the solution decay more rapidly than others.”[Lam91] Dymola allows the user to choose between many integration schemes to help alleviate this difficulty.

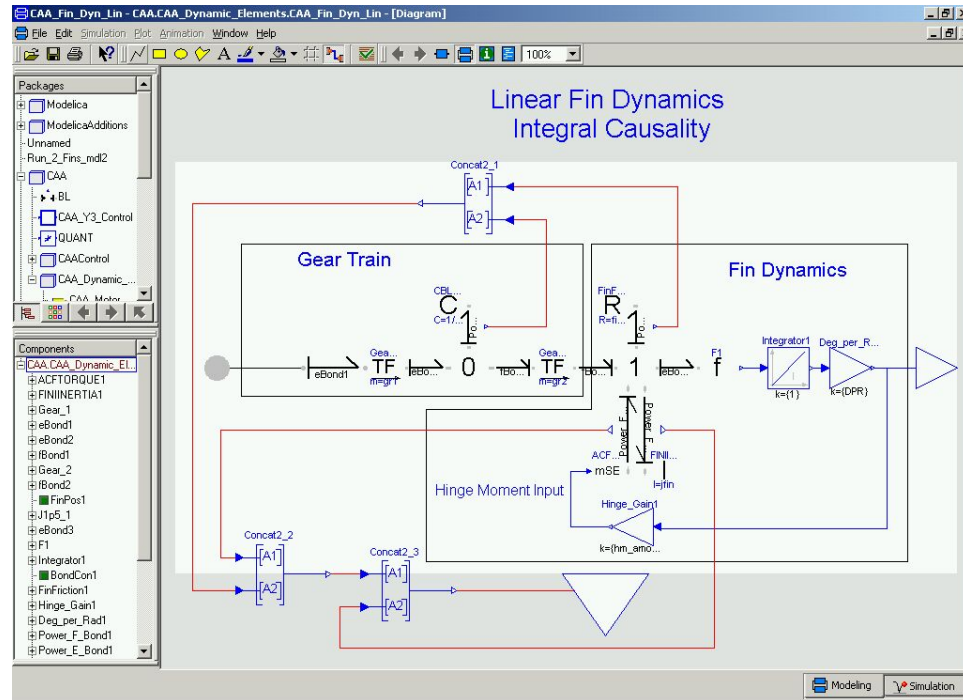


Figure 5.7. Linear Fin Dynamics Model (Integral Causal Model)

Another method of linearizing the backlash would be to remove the *BL* element of figure 5.5 entirely. In doing this substitution, the gear train *TF* elements, *gr1*, and *gr2*, are linked together forcing the user to change the causality of the bond graph. One possible choice of causality is shown in figure 5.8.

The backlash element has been replaced by a two-port 0-junction. Since all power is conserved in a bond graph junction, and the 0-junction of figure 5.8 has only two ports, the 0-junction has absolutely no effect on any of the bond graph equations. It has been left in the bond graph of figure 5.8 to aid the reader in comparing the different bond graph structures of figures 5.5, 5.7 and 5.8.

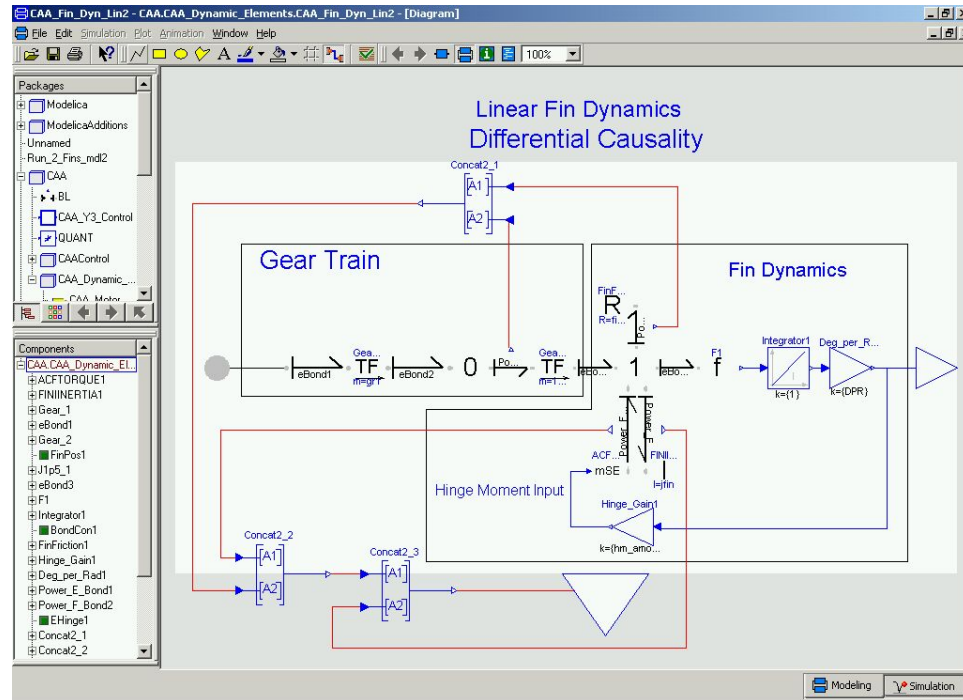


Figure 5.8. Linear Fin Dynamics Model (Differential Causal Model)

Figure 5.8 shows that the I -element representing the fin inertia has a differential causal mark. This causes a structural singularity in the bond graph. As discussed in Chapter 4, Dymola is capable of handling structural singularities, but it does make the algorithm more numerically intensive.

An alternative differential causal model is to keep the fin I -element integral causal and reverse the causalities of the bonds on the gear train transformers. This option, however, changes the causalities of the elements upstream, forcing the user to change the bond graphs of two separate Dymola models. Thus, this choice is not used here.

The bond graphs of figures 5.9 and 5.10 show the complete mechanical dynamics of the motor and fin of both the integral causal and differential causal models, respectively.

Here the electrical dynamics are represented entirely as an effort source. The purpose of such an analysis helps to determine if the chosen value of KBL is fine as is, too stiff, or not stiff enough.

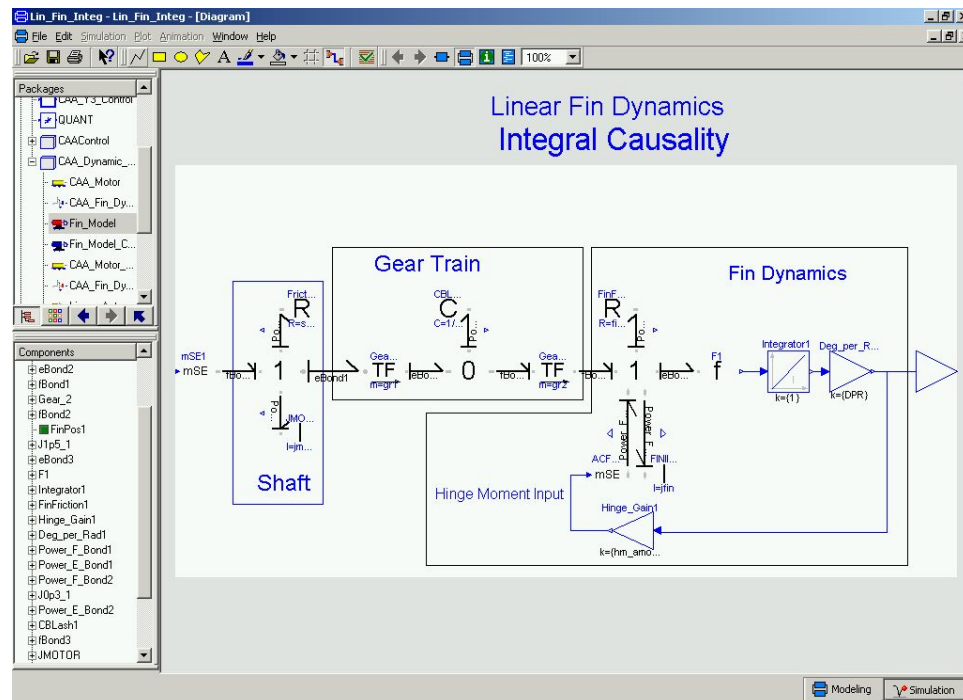


Figure 5.9. Linear Mechanical Fin Model (Integral Causal Model)

The bond graph state space equations for figure 5.9 are shown in equations 5.1 and 5.2. As seen by the bond graph of figure 5.9, these equations are 4th order linear, coupled, ODE's.

$$\begin{bmatrix} \dot{P}_{shaft} \\ \dot{q}_{GearTrain} \\ \dot{P}_{Fin} \\ \dot{q}_{FinPos} \end{bmatrix} = \begin{bmatrix} \frac{-shaftFriction}{jmotor} & -gr1 * KBL & 0 & 0 \\ \frac{gr1}{jmotor} & 0 & \frac{-gr2}{jfin} & 0 \\ 0 & gr2 * KBL & \frac{-fin_fric}{jfin} & hm_amount * DPR \\ 0 & 0 & \frac{1}{jfin} & 0 \end{bmatrix} \begin{bmatrix} P_{shaft} \\ q_{GearTrain} \\ P_{Fin} \\ q_{FinPos} \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} mSe1 \quad (5.1)$$

$$FinPos = \begin{bmatrix} 0 & 0 & 0 & DPR \end{bmatrix} \begin{bmatrix} P_{shaft} \\ q_{GearTrain} \\ P_{Fin} \\ q_{FinPos} \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} mSe1 \quad (5.2)$$

DPR represents the conversion from radians to degrees.

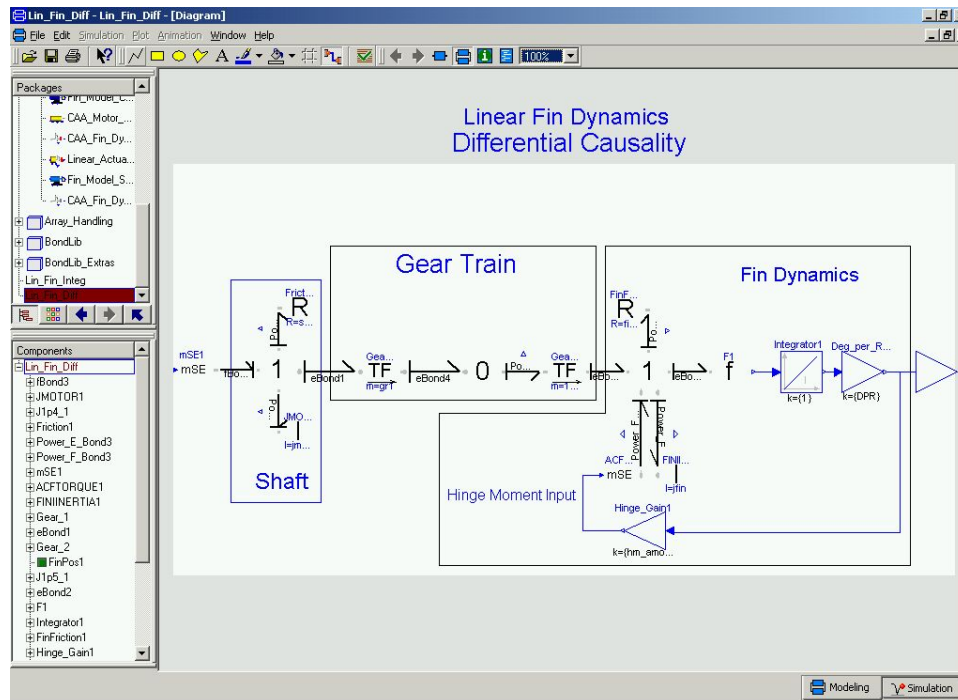


Figure 5.10. Linear Mechanical Fin Model (Differential Causal Model)

Similarly, the state space equations of the bond graph, shown in figure 5.10, are given in equations 5.3 and 5.4. As seen by the bond graph of figure 5.10, these equations are 2nd order, linear, coupled ODE's. The 2nd order, state space equations are somewhat more difficult to derive due to the differential causal element of the bond graph.

$$\begin{bmatrix} \dot{P}_{shaft} \\ \dot{q}_{FinPos} \end{bmatrix} = \begin{bmatrix} \frac{\text{shaft_friction} + \text{fin_fric} * \left(\frac{\text{gr1}}{\text{gr2}}\right)^2}{jmotor + jfin * \left(\frac{\text{gr1}}{\text{gr2}}\right)^2} & \frac{\left(\frac{\text{gr1}}{\text{gr2}}\right) * \text{hm_amount} * \text{DPR} * jmotor}{jmotor + jfin * \left(\frac{\text{gr1}}{\text{gr2}}\right)^2} \\ \frac{\text{gr1}}{\text{gr2} * jmotor} & 0 \end{bmatrix} \begin{bmatrix} P_{shaft} \\ q_{FinPos} \end{bmatrix} + \begin{bmatrix} \frac{jmotor}{jmotor + jfin * \left(\frac{\text{gr1}}{\text{gr2}}\right)^2} \\ 0 \end{bmatrix} mSel \quad (5.3)$$

$$FinPos = \begin{bmatrix} 0 & \text{DPR} \end{bmatrix} \begin{bmatrix} P_{shaft} \\ q_{FinPos} \end{bmatrix} + \begin{bmatrix} 0 \end{bmatrix} mSel \quad (5.4)$$

Obviously, equations 5.3 and 5.4 represent the system when the gear tooth stiffness is infinite. The modeling engineer has to choose among the options described in table 5.2.

Option	Description	Differential Causality?	Stiff System?	Disadvantages
1	A relatively low value for KBL, e.g. KBL = 34	No	No The model diverges from the others below .01 Hz	This option, although numerically desirable, does not represent the physical system.
2	A mid-range value for KBL, e.g. KBL = 340	No	No	This model has a low decrease in magnitude and phase at low freq.
3	Choose a large value for KBL, e.g. KBL = 3400	No	Yes	This stiff system causes integration difficulties yet the freq. response is not that different from option 2.
4	Choose the 2 nd order, differential causality model.	Yes	No	Dymola can handle the structural singularity but the physical system more then likely does not have infinite gear tooth stiffness.

Table 5.2. Linearized Backlash Modeling Options

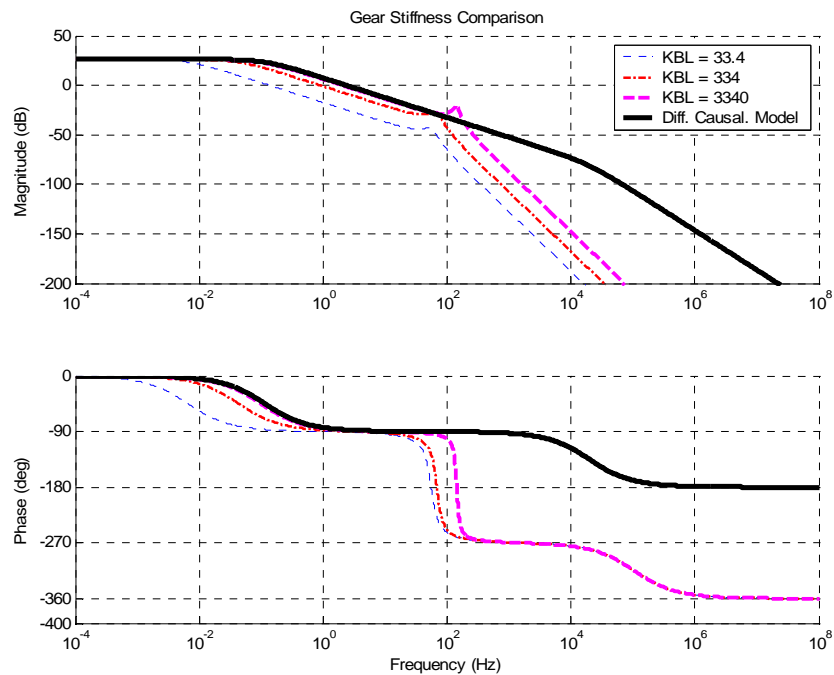


Figure 5.11. Bode Comparison for Different Values of KBL

Given the options from table 5.2 and viewing the frequency response from figure 5.11, option three will be used as the linearized version of the fin dynamics. This analysis also helps verify the KBL value used in the nonlinear backlash model.

5.2.2.2 Linearized Motor Dynamics

There are two nonlinearities in the motor dynamics model of figure 5.2, the modulated resistor $mr1$ and the PWM control signal input to the modulated gyrator. The first nonlinearity is simple enough to change. The linearization is made by simply assuming that the value of $mr1$ is constant and set to the forward-bias diode value, rf ,

which implies that the capacitor voltage $cbatt$ never exceeds the input voltage $vbatt$. This set of linearizing assumptions results in the state space matrices shown in equations 5.5 through 5.8. These equations represent both the motor and linearized fin dynamics in terms of the values defined in table 5.1. The state vector for equations 5.5 is

$$X = [q_{cbatt} \quad p_{lbatt} \quad q_{cdriver} \quad p_{lmotor} \quad p_{jmotor} \quad q_{backlash} \quad p_{jfin} \quad q_{fin_pos}]^T.$$

$$A = \begin{bmatrix} -\frac{1}{rf*cbatt} & -\frac{1}{lbatt} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{cbatt} & \frac{1}{lbatt} & -\frac{1}{cdriver} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{lbatt} & 0 & -\frac{PWM}{lmotor} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{PWM}{cdriver} & -\frac{lmotor}{Rcoil} & -\frac{kmotor}{jmotor} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{kmotor}{lmotor} & -\frac{shaft_friction}{jmotor} & -gr1*KBL & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{gr1}{jmotor} & 0 & -\frac{gr2}{jfin} & 0 \\ 0 & 0 & 0 & 0 & 0 & gr2*KBL & -\frac{fin_fric}{jfin} & DPR*hm_amount \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{jfin} & 0 \end{bmatrix} \quad (5.5)$$

$$B = \begin{bmatrix} -\frac{1}{rf} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \quad (5.6)$$

$$C = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad DPR] \quad (5.7)$$

$$D = [0] \quad (5.8)$$

The input to this system is the constant voltage $vbatt$. The system is controlled by changing the value of the PWM signal, which is an internal element to the A matrix,

showing up on terms (3, 4) and (4, 3). Obviously, for constant values of PWM these state space matrices are entirely linear. This insight, at first glance, does little to help when attempting to control the fin's position. However, if the integration technique is set up such that the motor/fin dynamics are integrated at a much higher rate than the control signal is updated, then, in between control signal updates, the above system can be treated as a linear system.

Further investigation into the bond graph of figure 5.2 shows that the model can be broken at the battery/coil, i.e., where the PWM signal enters the model. The causal marks on the gyrator show that the output of the battery is the effort signal on the capacitor *cdriver* and the output of the coils is the current in the inductor *lmotor*. These outputs serve as corresponding inputs to the respective models, the capacitor voltage is an input to the coil model and the inductor current is an input to the shaft model. Breaking the model in this fashion, converts equations 5.5-5.8 to equations 5.9-5.12. Equations 5.9 and 5.10 represent the battery portion only; equations 5.11 and 5.12 represent the rest of the system. The single input to equation 5.11 allows the user to perform analysis on the type of control effort that needs to be generated to position the fin at the desired location in the desired amount of time. Breaking the model in this fashion aids in the controller design process. The bond graph modeling approach allows the user to break the state space matrices in a straight-forward manner.

$$\begin{bmatrix} \dot{q}_{cbatt} \\ \dot{p}_{lbatt} \\ \dot{q}_{cdriver} \end{bmatrix} = \begin{bmatrix} -\frac{1}{rf} & 0 & 0 \\ \frac{1}{cbatt} & -\frac{rbatt}{lbatt} & -\frac{1}{cdriver} \\ 0 & \frac{1}{lbatt} & 0 \end{bmatrix} \begin{bmatrix} q_{cbatt} \\ p_{lbatt} \\ q_{cdriver} \end{bmatrix} + \begin{bmatrix} \frac{1}{rf} & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} vbatt \\ PWM * f_{lmotor} \end{bmatrix} \quad (5.9)$$

$$e_{cdriver} = \begin{bmatrix} 0 & 0 & \frac{1}{cdriver} \end{bmatrix} \begin{bmatrix} q_{cbatt} \\ p_{lbatt} \\ q_{cdriver} \end{bmatrix} + \begin{bmatrix} 0 & 0 \end{bmatrix} \begin{bmatrix} vbatt \\ PWM * f_{lmotor} \end{bmatrix} \quad (5.10)$$

$$\begin{bmatrix} \dot{p}_{lmotor} \\ \dot{p}_{jmotor} \\ \dot{q}_{backlash} \\ \dot{p}_{jfin} \\ \dot{q}_{fin_pos} \end{bmatrix} = \begin{bmatrix} -\frac{Rcoil}{lmotor} & -\frac{km}{jmotor} & 0 & 0 & 0 \\ \frac{km}{lmotor} & -\frac{shaft_friction}{jmotor} & -gr1 * KBL & 0 & 0 \\ 0 & \frac{gr1}{jmotor} & 0 & -\frac{gr2}{jfin} & 0 \\ 0 & 0 & gr2 * KBL & -\frac{fin_fric}{jfin} & hm_amount * DPR \\ 0 & 0 & 0 & \frac{1}{jfin} & 0 \end{bmatrix} \begin{bmatrix} p_{lmotor} \\ p_{jmotor} \\ q_{backlash} \\ p_{jfin} \\ q_{fin_pos} \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} [PWM * e_{cdriver}] \quad (5.11)$$

$$\begin{bmatrix} fin_pos \\ f_{lmotor} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & DPR \\ \frac{1}{lmotor} & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_{lmotor} \\ p_{jmotor} \\ q_{backlash} \\ p_{jfin} \\ q_{fin_pos} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} [PWM * e_{cdriver}] \quad (5.12)$$

Equations 5.11 and 5.12 show that a completely linear model is created when the entire battery model is collapsed down to an effort source. This insight is obvious from the bond graph of figure 5.2. This linear reduction is shown in figure 5.12. Inside the block labeled *Linear_Fin* is the bond graph of figure 5.9. The input is the control effort and the

outputs are fin position and select power signals from the bond graph, as before. A significant difference of this motor model compared to those presented above, is that the controller must output a different control effort, since the control effort is converted directly into the input effort through the mSe element. For the analysis presented here, the output voltage of the battery is assumed to remain at 200 volts.

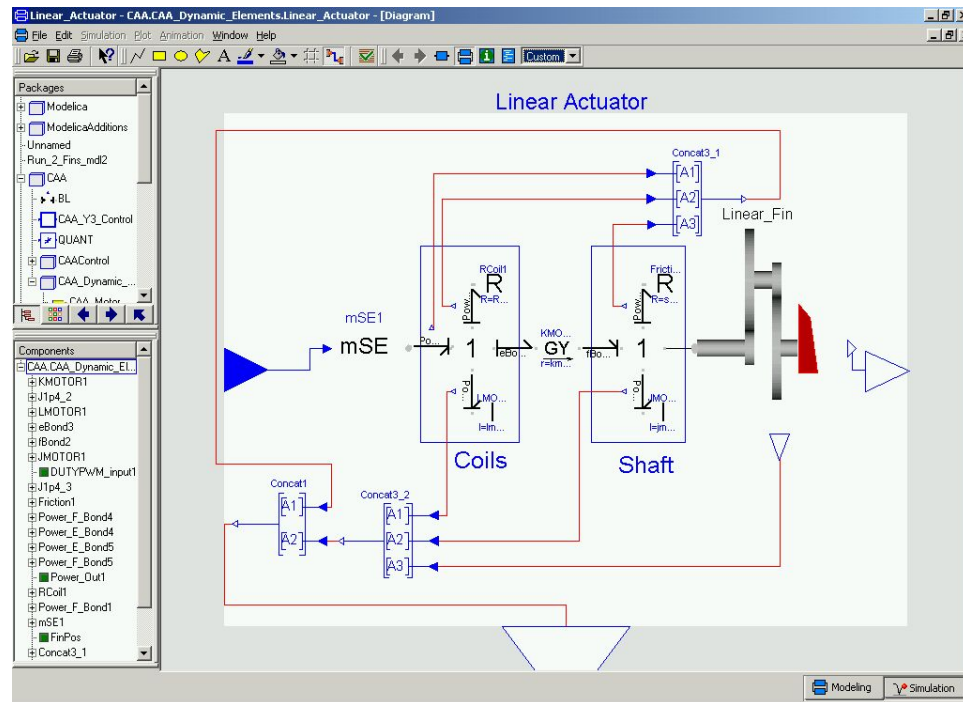


Figure 5.12. Linear Actuator with Fin Dynamics

Thus, including a gain of 200 on the control signal, outside of the controller, the controller output will more closely resemble the nonlinear PWM control signal.

5.3 Power Flow Considerations

As can be seen in figures 5.1 and 5.2, no usable power comes from the controller of the nonlinear fin positioning system. The controller simply signals the power flow in the actuator through the modulated transformer. This signal governs the amount of power flow throughout the entire system. By monitoring the power flow through the system, a method can be derived to classify the effectiveness of the controller. A comparison of the effectiveness of different controllers can be made by observing their ability to utilize the power supplied by the motor. By monitoring the power flow through the bond graph, one can get an idea of the effectiveness of any given controller. In order to monitor the power flow through the bond graph, the bonds connected to sources and passive elements are special bonds that provide an additional power-signal output, as previously shown in figures 5.2, 5.5 and 5.6.

The bond graph of figure 5.2 shows that a power limit exists in the system. It is clear that the maximum power delivered to the fin cannot be more than exists in the power supply (battery). In order to monitor the ability of the controller to utilize the power from the battery, an integrator has been connected to the power-bond of the input source in figure 5.2. Obviously, the power signals can be integrated to form energy signals. By dividing the output energy by the input energy, a control designer can get an idea of how efficient the controller is. Naturally, this analysis must be performed after a control scheme has been found.

5.4 Servo Controllers

Separate control schemes are presented in this section. Three linear control schemes are used for the linearized system and two nonlinear control schemes are used on the nonlinear system. All control schemes operate under the assumption that fin position is the only state available for feedback.

5.4.1 Linear Control Schemes

The linear control schemes to be compared are three separate *PID* [Fra94] controllers that are intended to be used to control the linearized fin dynamics model shown in figure 5.12. The three *PID* controllers are shown in table 5.3.

<i>PID</i> Controller Number	Controller Transfer Function
<i>PID1</i>	$\frac{0.095 (S + 0.0909)}{S}$
<i>PID2</i>	$\frac{0.8055 (S^2 + 120.4762 S + 4.4872)}{S^2 + 993.949 S}$
<i>PID3</i>	$\frac{67.8408(S^2 + 101.7574S + 1.485)}{S^2 + 100000.3774S}$

Table 5.3. *PID* Controllers

Obviously, *PID1* is really just a *PI* controller but can be considered a *PID* with $T_D=0$.

The Bode plots for each of the three controllers are shown in figure 5.13.

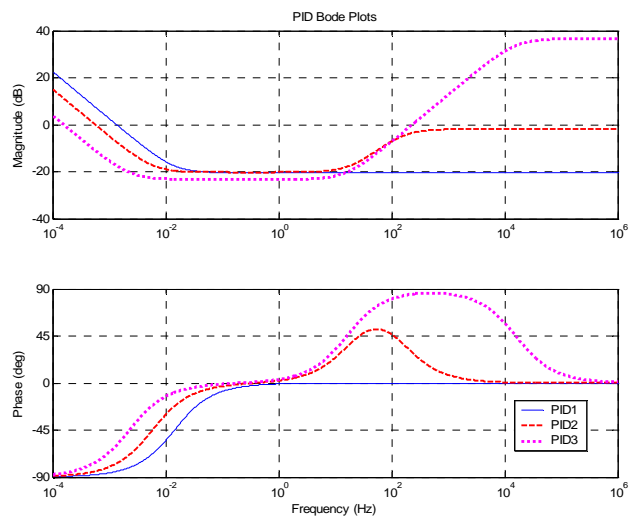


Figure 5.13. PID Bode Plots

The output of the controllers is multiplied by a gain of 200 before it enters the fin dynamics model. This setup is shown in figure 5.14. The icon labeled *Linear_Actuator* contains the bond graph of figure 5.12.

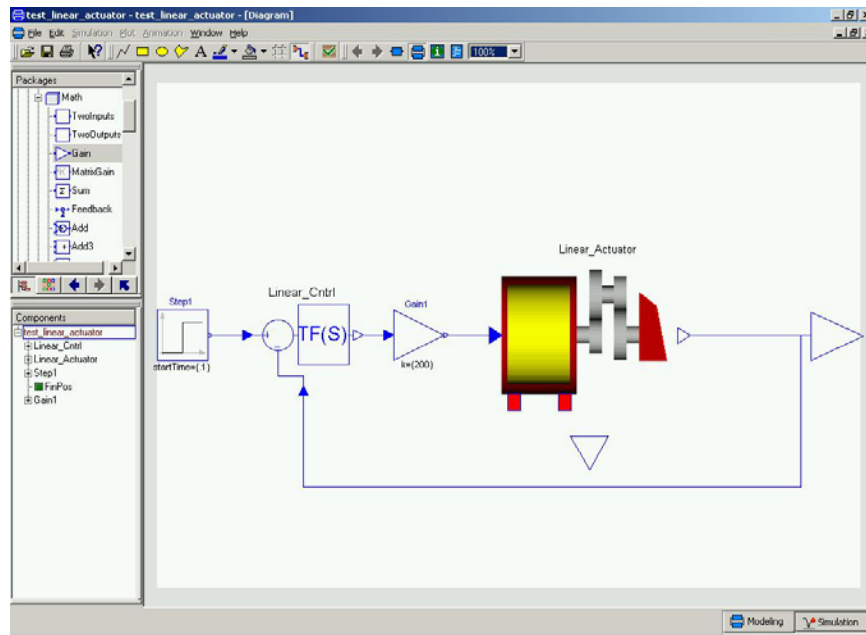


Figure 5.14. Linear Controller/Actuator

5.4.2 Non-Linear Control Scheme 1

The first nonlinear controller is shown in figure 5.15. This figure represents the internal workings of the controller block in figure 5.1. The block labeled PI_z contains the transfer function stated by equation 5.13, which is the Z -transform of the continuous $PID1$ [Fra98] presented in the previous section. This discrete transfer function operates with a sample rate of 6000 Hz.

$$G(Z) = \frac{0.095 * [z - 0.999985]}{z - 1} \quad (5.13)$$

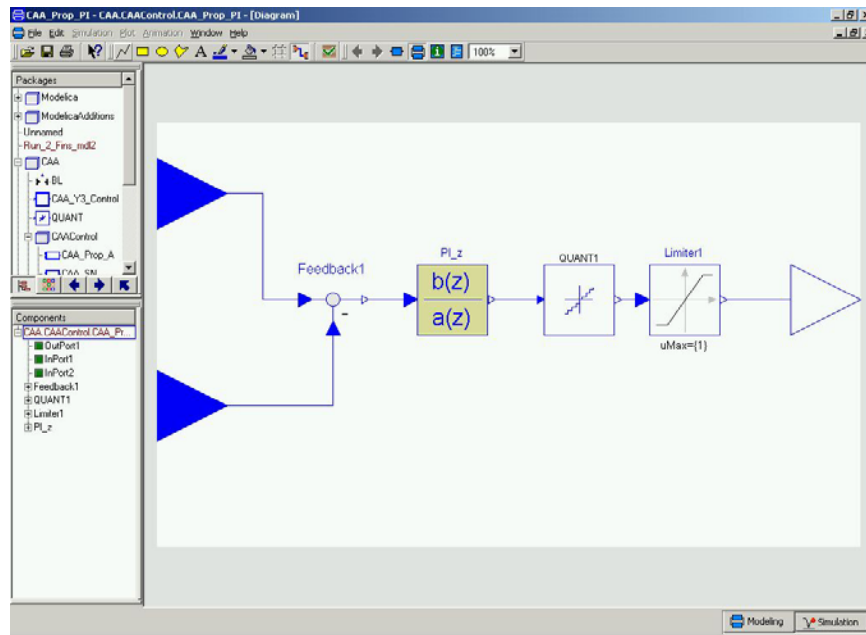


Figure 5.15 Non-Linear Controller 1

The quantizer relationship is shown as

$$output = 0.015 * \text{sign}(input) * \text{floor}\left(\frac{input}{0.015}\right) \quad (5.14)$$

The ± 1 limit is to keep the PWM output of the controller within the physical capabilities of the modulated transformer in figure 5.2. Unlike the PID controllers of the previous section, there is no gain of 200 on the output since this gain is created when the *PWM* signal is multiplied by the output of the battery through the modulated gyrator of figure 5.2.

5.4.3 Non-Linear Control Scheme 2

Figure 5.16 shows the second controller considered to handle the system of figure 5.1. The quantizer block and the ± 1 limit block are the same for the two nonlinear controllers. Controller 2, however, is designed to operate at 1200 Hz. The delay on the output of controller 2 accounts for latency between the controller and the actuator.

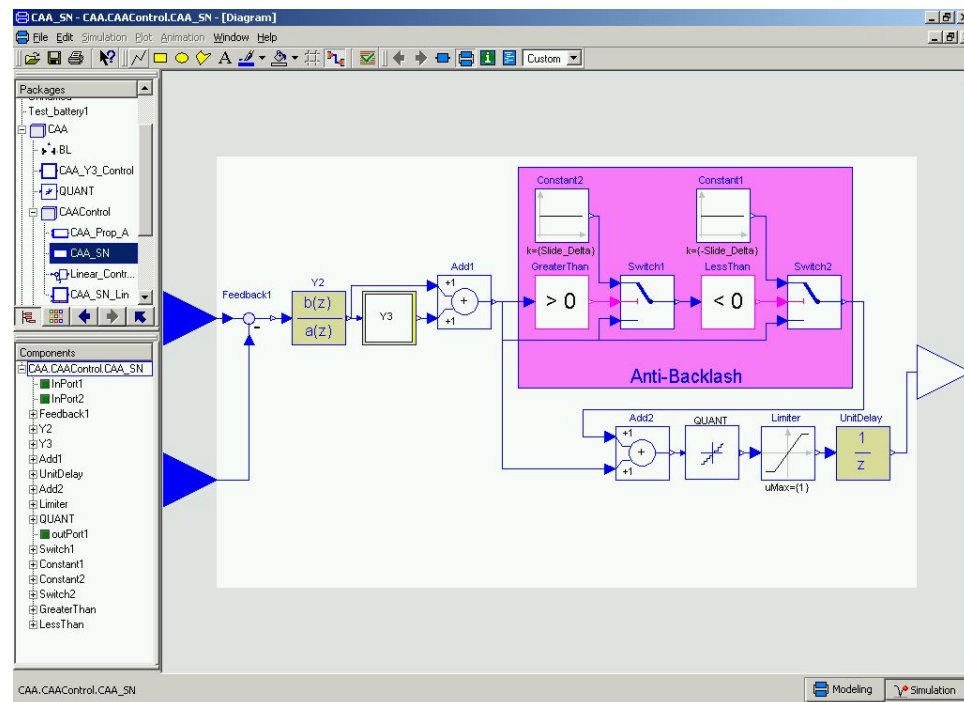


Figure 5.16. Non-Linear Controller 2

As seen in figure 5.16, controller 2 has a distinctive anti-backlash element in the controller to counteract the backlash in the gear train. This backlash element adds a sign-dependent bias to the control signal to push the gear train through the backlash region, such that the gears stay in contact as much as possible. As expected, the backlash in the

system causes a limit cycle in the steady-state response. The anti-backlash element works to reduce the effects of the gear train backlash. The anti-backlash bias, denoted *Slide_Delta* in figure 5.16, is set at 0.02, which is the amount added to the *PWM* signal to counter-act the gear train backlash. The anti-backlash element is setup to add the *Slide_Delta* amount only if the input is non-zero.

The transfer function for the element *Y2* is given by equation 5.15 with a sample rate of 1200 Hz. *Y2* provides phase lead at lower frequencies.

$$Y2(z) = \frac{0.172 * [z - 0.688]}{z - 0.453} \quad (5.15)$$

The *Y3* element of figure 5.16 is expanded and shown in Figure 5.17. The *Y3* element serves as a nonlinear limiter, which depending on the conditions shown in figure 5.17, moves a controller zero between 0 and -1 in the *z*-plane, which adds conditional phase shift to the control signal. The value used for L_{ii} is 0.11, and the value of L_{io} is 0.06.

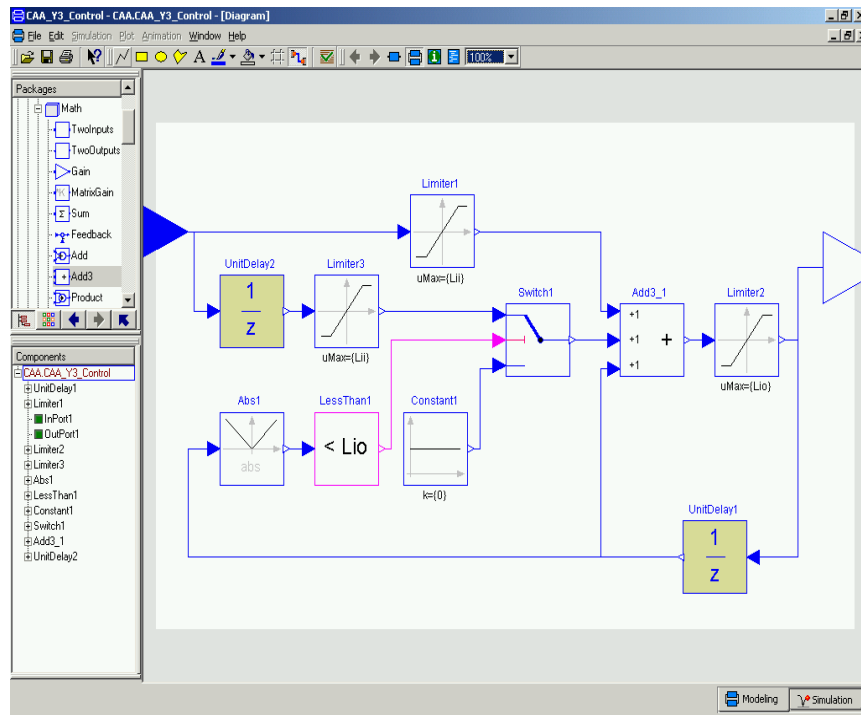


Figure 5.17. Content of the Y3 Element

Nonlinear controller 1 (*NC1*) is a much simpler scheme that does not actively try to cancel out the backlash of the system. The backlash will then add phase lag to the overall system. Nonlinear controller 2 (*NC2*) actively attempts to account for the backlash in the gear train. The anti-backlash in controller 2 allows for a controller that can run at a slower rate without significant phase loss.

5.5 Step Response Comparisons

5.5.1 Step Response Comparisons of Linear Systems

Step responses of the system in figure 5.14 for each of the *PID* controllers are shown in this section. Both 5° and 20° fin position commands are used as step inputs. The hinge moment amount, used to simulate an aerodynamic load on the fin, is set at 0, -6 and -6 $N*m/(Deg \text{ of fin deflection})$. These three settings simulate zero, medium and high fin loads which will demonstrate each controller's ability under varying circumstances.

The Dymola model of figure 5.14 has been modified to include the analysis of the power signals from the bond graphs. This modification is shown in figure 5.18.

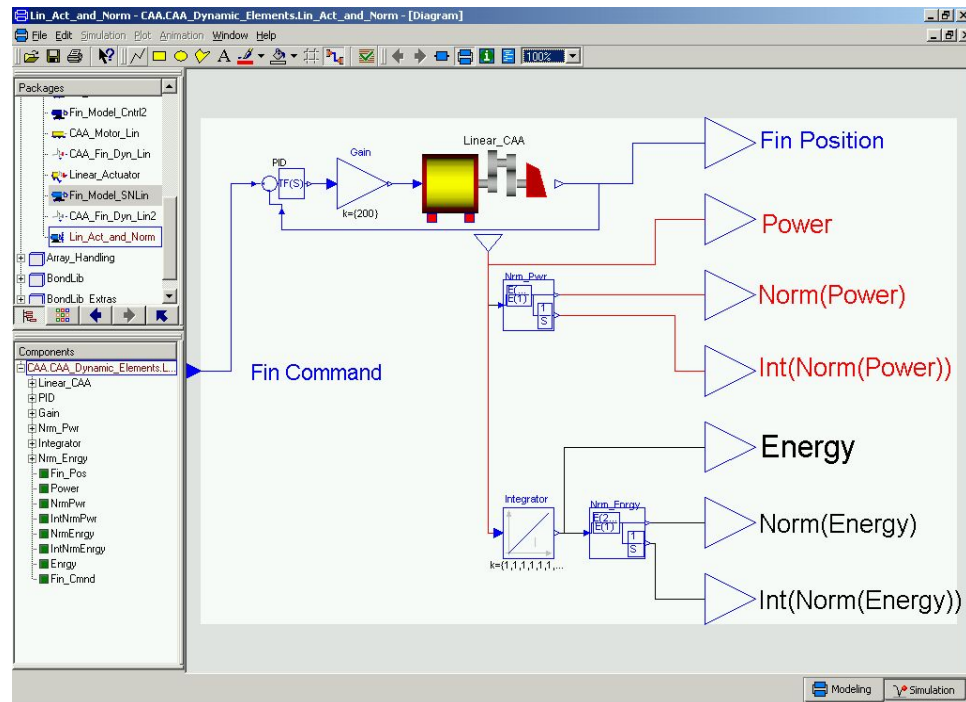


Figure 5.18. Linear Actuator with Power Signal Analysis

The power signal vector is passed through an analysis block labeled *Nrm_Pwr*. This block divides the 2^{nd} through the n^{th} element of the input vector by the 1^{st} element. This normalizes the j^{th} element by the 1^{st} element. The new normalized vector is of dimension $n-1$. This $n-1$ normalized vector is passed as an output. Also, this normalized vector is passed through an integrator to calculate the integral of the normalization. The Dymola code for this block is shown below in figure 5.19.

```

model Normalized_Energy
parameter Integer n(min=2) = 2 "Order of input vector";
parameter Real dn_min=1E-6 "Min. Divisor (divide by zero protection)";
Real x[n - 1];
Real int_x[n - 1];
Real dn;
Real dnl;
equation
dnl = if (abs(Input.signal[1]) < dn_min) then dn_min*sign(Input.signal[1])
      else Input.signal[1];
dn = if (dnl < dn_min) then dn_min else dnl;
for i in 1:n - 1 loop
  x[i] = Input.signal[i + 1]/dn;
  der(int_x[i]) = x[i];
  Norm_Energy.signal[i] = x[i];
  Int_Norm_Energy.signal[i] = int_x[i];
end for;
end Normalized_Energy;

```

Figure 5.19. Dymola Code for Vector Normalization

The first element of the power vector is the power on the *mSE* input bond of figure 5.12. In this fashion, all of the output power signals from the bond graph of figure 5.12 are normalized by the input power. Figure 5.18 shows the power signal passing through an integrator to create an energy signal. Each element of the vector is integrated separately. This energy vector is passed through another normalization block, called

Nrm_Energy , to perform the same analysis on the energy vector as was done on the power vector. Plots of these vectors are also included in the step response analysis.

By using the Dymola model of figure 5.18 in an object-oriented fashion, the three PID controllers can be compared at once. This model is shown in figure 5.20.

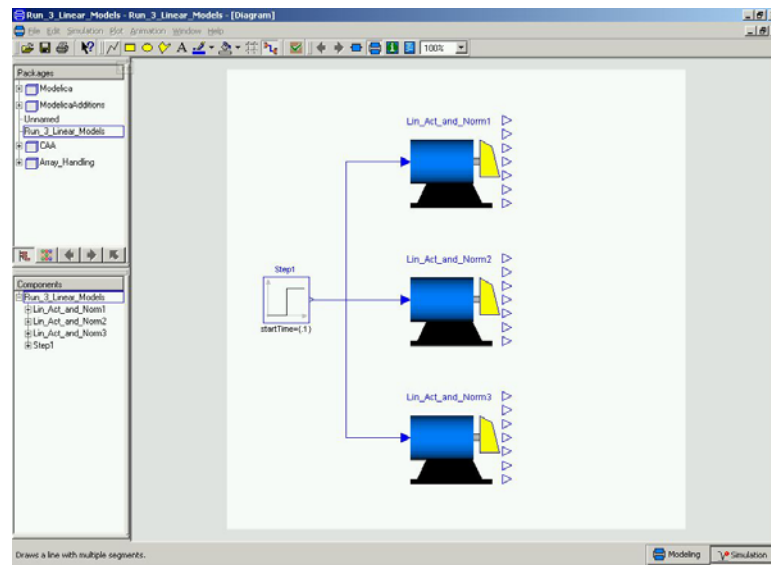


Figure 5.20. Three PID Actuators with Power Signal Analysis

As seen in figure 5.20, the same input is used in all three models. The step starts at 0.1 seconds. Figure 5.21 shows the three step responses for a 5° step command with no hinge moment load. Obviously *PID1* gives the best response of the three, since it rises just as quick, settles faster and has less high frequency oscillation.

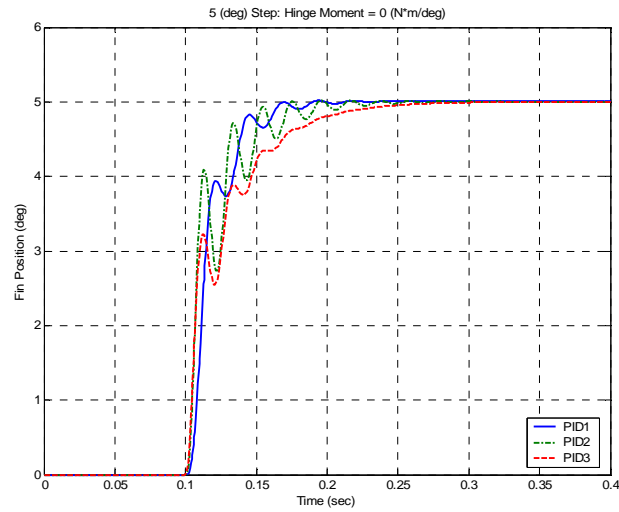


Figure 5.21. 5° Step Response, No Hinge Moment

Figure 5.22 shows the input power for the three controllers. This signal is equivalent to 200 times the control effort.

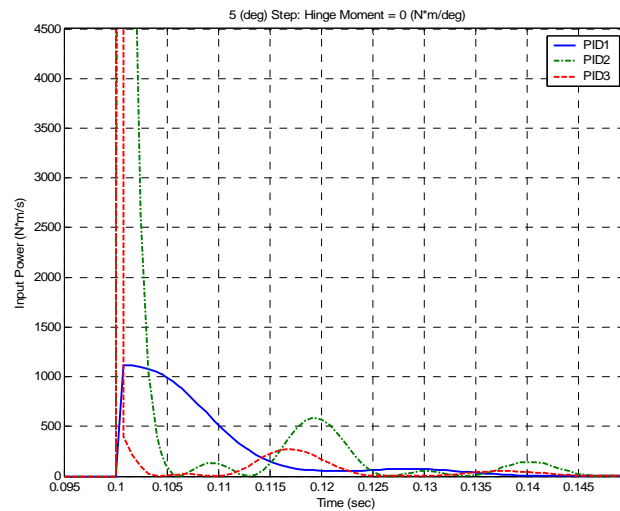


Figure 5.22. Power Input: 5° Step Response, No Hinge Moment

Initially a relatively small amount of power is delivered to the system by *PID1* compared to the other two controllers. The amount of power delivered to the fin for each of the controllers is shown in figure 5.23. Similar to the input power, the power delivered to the fin is smaller for *PID1* than the other two controllers.

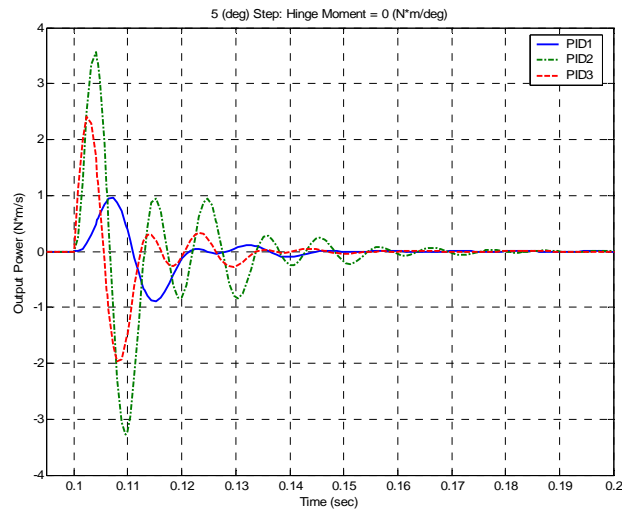


Figure 5.23. Power Output: 5° Step Response, No Hinge Moment

However, figure 5.23 does not clarify how efficient the controllers are. It simply states that less power was delivered to the fin for *PID1*.

The output power is normalized by the input power to get an idea of the efficiency of the controller. An efficiency calculation is obtained by dividing the output power signal by the input power signal for each of the three controllers. This calculation is shown in figure 5.24.

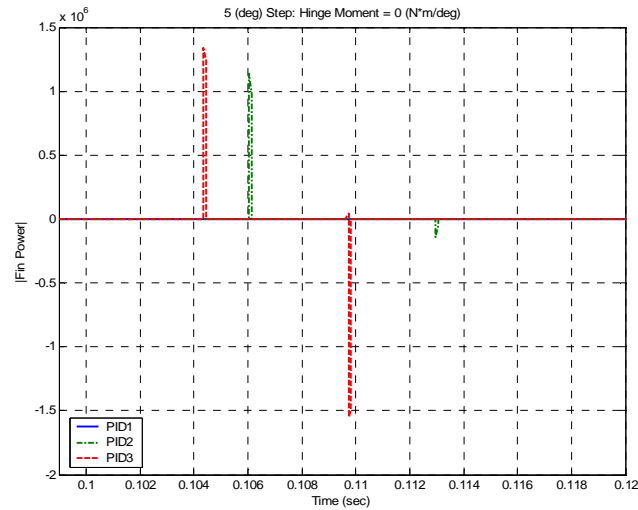


Figure 5.24. |Fin Power|: 5° Step Response, No Hinge Moment

The signals of figure 5.24 are impossible to compare. There are very large differences in the scales of the signals mostly due the input power signal getting close to zero when the output power signal is non-zero, as seen in figure 5.22 and 5.23. These large values, caused by division by numbers close to zero, make the comparison useless. The integration of the normalized power would still not make a fair comparison due to the integration of the large values shown in figure 5.24.

Although this analysis proved useless for the power signals, this same analysis can be used on the energy signals. The input energy signals are shown in figure 5.25. It is not surprising that *PID1* delivers less energy to the system than the other two controllers. This normalization analysis of the energy signals, at this point, seems more feasible, since the energy signals do not go to zero after the step input. Obviously, they will not go to

zero since a positive amount of power was needed to move the fin to begin with, and energy is the time integral of power.

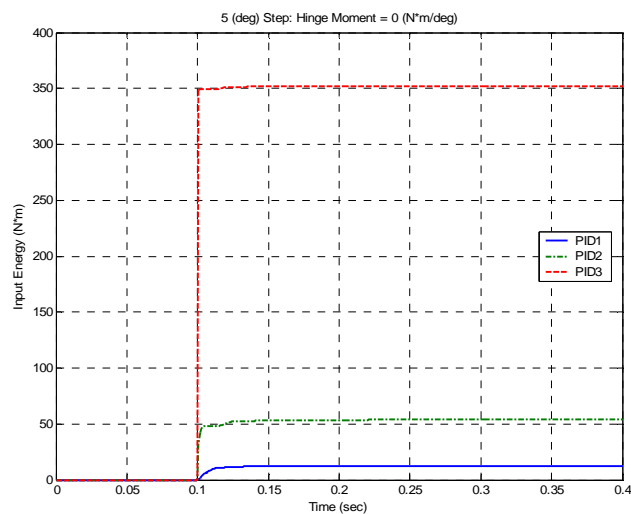


Figure 5.25. Energy Input: 5° Step Response, No Hinge Moment

Figure 5.26 shows the output energy, i.e., the energy that is used to move the fin.

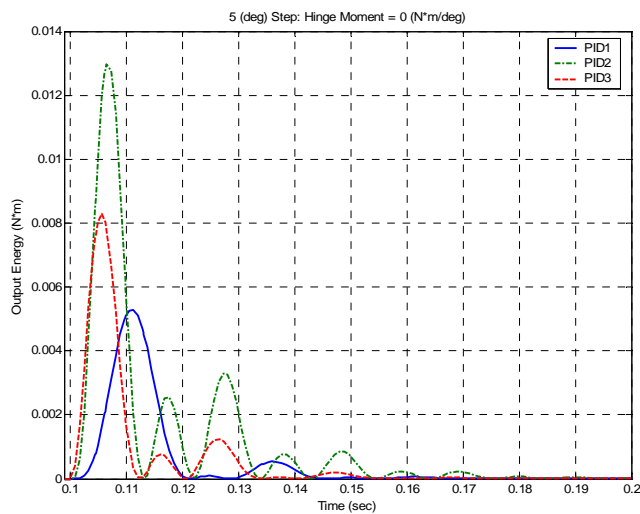


Figure 5.26. Energy Output: 5° Step Response, No Hinge Moment

PIDI delivers less energy to the fin than the other two controllers. Again, by normalizing the output energy by the input energy one can create a measurement of efficiency. The output energy divided by the input energy is shown in figure 5.27. This unitless signal does not have the divide by zero problems that the normalized power signal had, after the step input is applied, since the input energy is positive for all time. This is true for all stable controllers. The input energy is zero before the step input is applied, yet the normalized energy signals shown in figure 5.27 all have a value of zero up until the step input is applied at a time equal to 0.1 seconds. This detail is somewhat artificial, but is not very significant. The way that this is accomplished is through the divide by zero protection logic shown in figure 5.19. When the divisor is below a certain threshold, 1E-6 by default, then the divisor is set equal to this value. Thus, the denominator is non-zero. The numerator, however, is zero. Since no energy has yet been delivered to the system, the output energy must, by definition, be zero. Thus, the numerator is identically zero, and therefore the output of the calculation is zero.

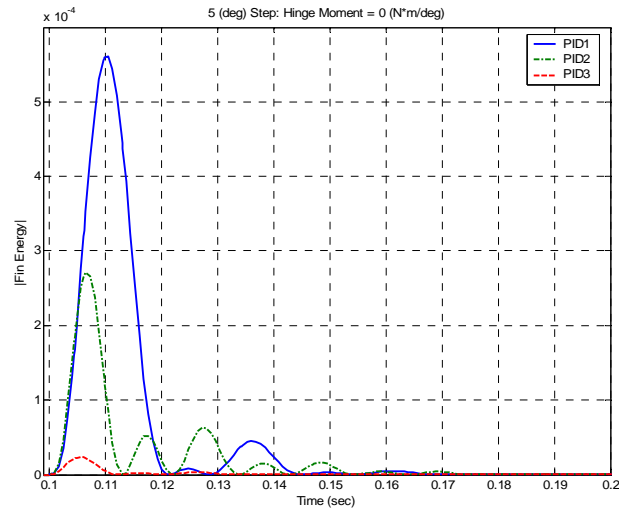


Figure 5.27. |Fin Energy|: 5° Step Response, No Hinge Moment

The slightly oscillatory nature of the signals in figure 5.27 still makes them difficult to compare. One can *sum up* the values of these signals by integrating them over time. The integrated signals are shown in figure 5.28.

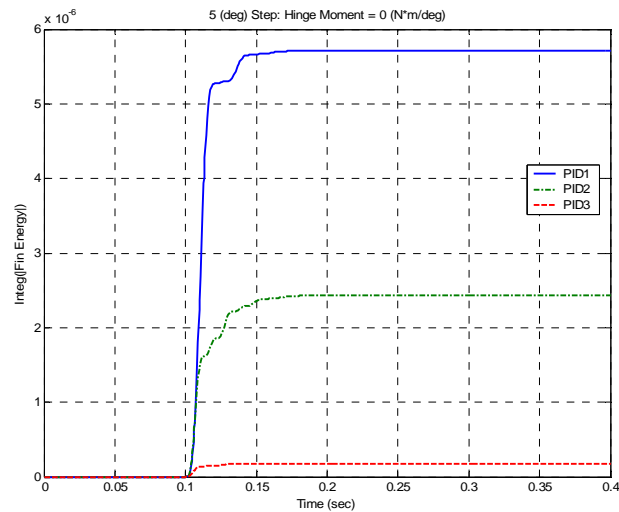


Figure 5.28. Integral(|Fin Energy|): 5° Step Response, No Hinge Moment

Figure 5.28 shows a clear and concise calculation of efficiency. Therefore a measurement of controller efficiency can be defined by

$$\eta_{controller} = \int_o^{if} \left[\frac{\int_0^\tau (OutputPower) dt}{\int_0^\tau (InputPower) dt} \right] d\tau = \int_0^{if} \frac{OutputEnergy}{InputEnergy} dt \quad (5.16)$$

The symbol η is taken from the thermal dynamic symbol for thermal dynamic 1st law efficiency [Bej97, War95].

Figure 5.28 shows *PID1* is the most efficient of the three controllers. Obviously, the worst outcome for any stable system is a zero value for η , where the *InputPower* is non-zero. This absolute worst case occurs for trivial systems, which is shown graphically in figure 5.29.

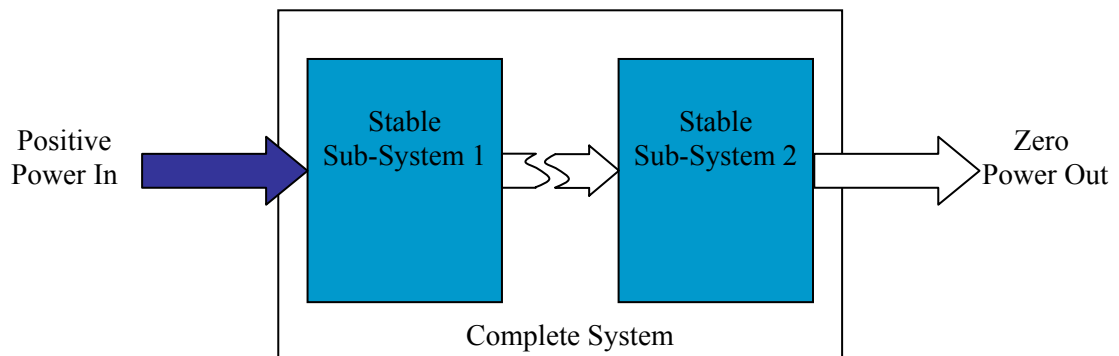


Figure 5.29. Efficiency Worst Case

Obviously, the complete system of figure 5.29 is stable. The calculation of η is zero for all time regardless of the input, since the two sub-systems are not connected together.

Once the power input to a system has moved from zero to a non-zero value, it is impossible for the *InputEnergy* term of equation 5.16 to return to zero. This behavior is true for any single input, physical system. In order for the *InputEnergy* to return to zero, the system source would need to become a sink, and the system would need to sink as much power into that element as it received. This is in violation of the 2nd law of thermodynamics.

The efficiency calculation shown in figure 5.28 is for an ideal case. In this case, there was no external load to the system, so the controller was able to apply zero power to the system once the fin reached the commanded position. In this case, the efficiency signals rose sharply and then flattened out. Equation 5.17 shows the efficiency definition expanded to bond graph generic terms. At $t=tf$ the angular rate of the fin, $Flow_{Out}$, is zero, since the fin reached steady state. This condition allows the controller efficiency to reach a constant value.

$$\eta_{controller} = \int_0^{tf} \left[\frac{\int_0^{\tau} (OutputPower) dt}{\int_0^{\tau} (InputPower) dt} \right] d\tau = \int_0^{tf} \left[\frac{\int_0^{\tau} (Effort_{Out} * Flow_{Out}) dt}{\int_0^{\tau} (Effort_{In} * Flow_{In}) dt} \right] d\tau \quad (5.17)$$

In the case where the output power approaches, but never reaches zero, the efficiency signal will not reach a constant value. This behavior can be seen by applying an external load to the linear systems.

The step responses and efficiency measurements have been provided for different input commands and external load configurations in figures 5.30 through 5.39. Figures 5.30 and 5.31 correspond to a 5° step with $-0.6 \text{ N}\cdot\text{m}/\text{deg}$ hinge moment.

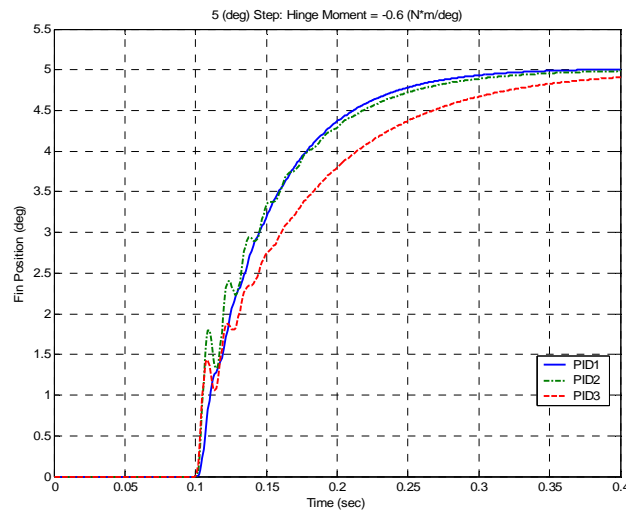


Figure 5.30. 5° Step Response, Hinge Moment = $-0.6 \text{ (N}\cdot\text{m}/\text{deg.)}$

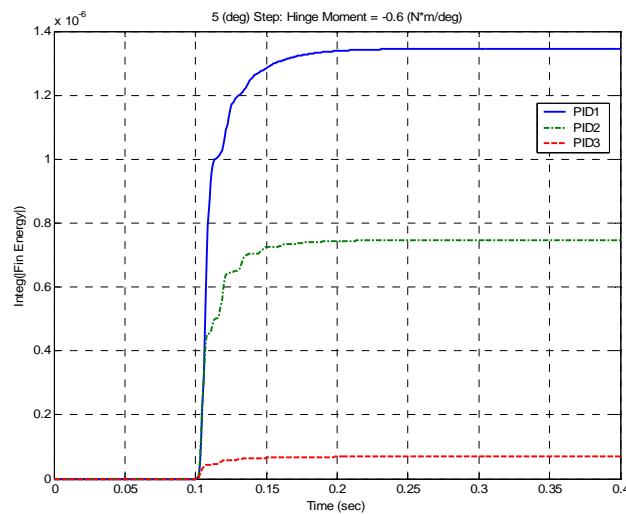


Figure 5.31. η : 5° Step Response, Hinge Moment = $-0.6 \text{ (N}\cdot\text{m}/\text{deg.)}$

Figures 5.32 and 5.33 correspond to a 5° step with $-6 \text{ N}^*\text{m}/\text{deg}$ hinge moment.

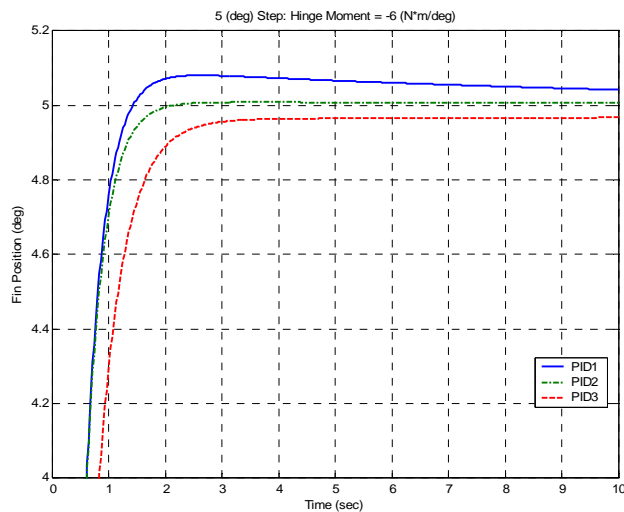


Figure 5.32. 5° Step Response, Hinge Moment = $-6 \text{ (N}^*\text{m}/\text{deg.)}$

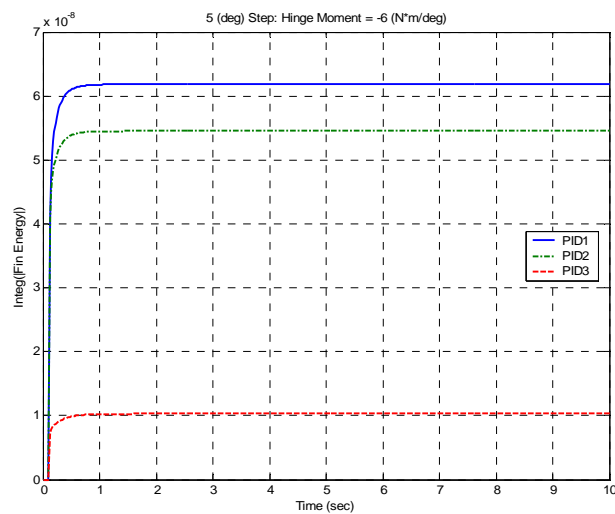


Figure 5.33. η : 5° Step Response, Hinge Moment = $-6 \text{ (N}^*\text{m}/\text{deg.)}$

Figures 5.34 and 5.35 correspond to a 20° step with $0 \text{ N}\cdot\text{m}/\text{deg}$ hinge moment.

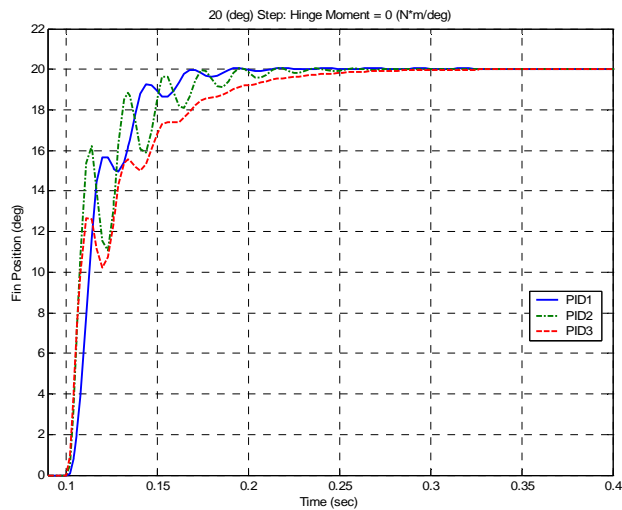


Figure 5.34. 20° Step Response, Hinge Moment = $0 \text{ (N}\cdot\text{m}/\text{deg.)}$

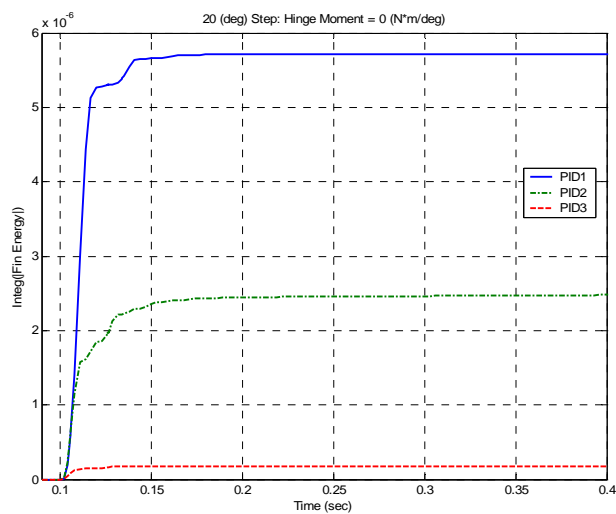


Figure 5.35. η : 20° Step Response, Hinge Moment = $0 \text{ (N}\cdot\text{m}/\text{deg.)}$

Figures 5.36 and 5.37 correspond to a 20° step with $-0.6 \text{ N}^*\text{m/deg}$ hinge moment.

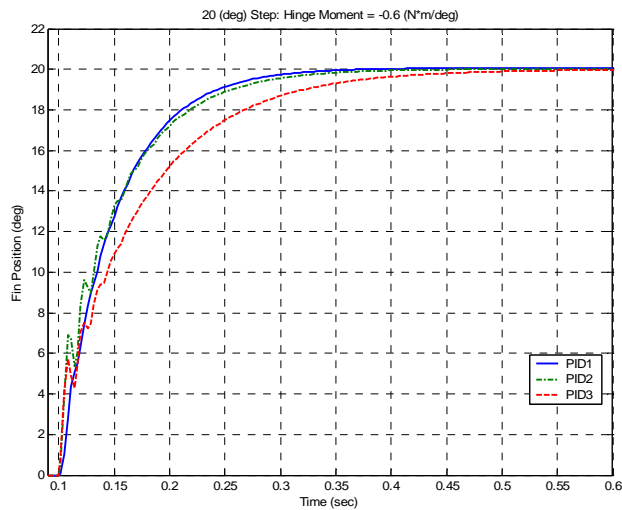


Figure 5.36. 20° Step Response, Hinge Moment = $-0.6 \text{ (N}^*\text{m/deg.)}$

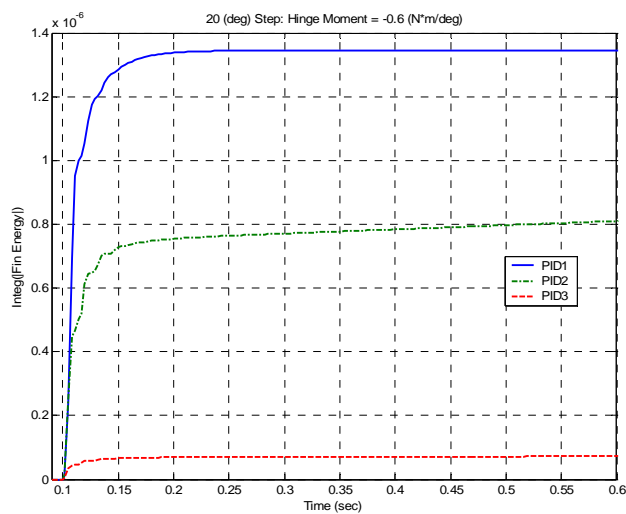


Figure 5.37. η : 20° Step Response, Hinge Moment = $-0.6 \text{ (N}^*\text{m/deg.)}$

Figures 5.38 and 5.39 correspond to a 20° step with $-6 \text{ N}\cdot\text{m}/\text{deg}$ hinge moment.

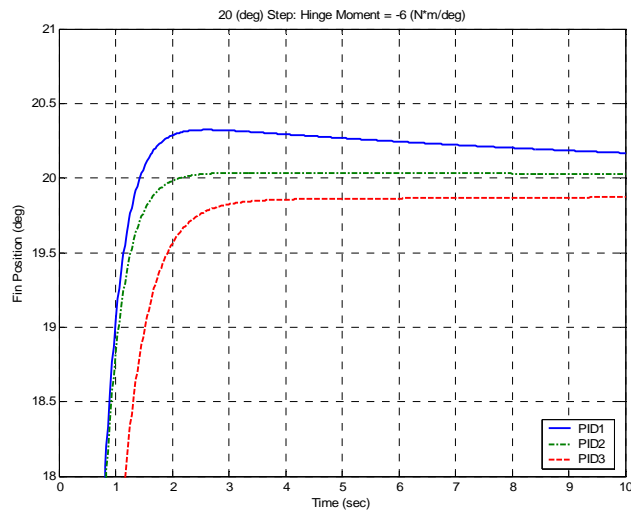


Figure 5.38. 20° Step Response, Hinge Moment = $-6 \text{ (N}\cdot\text{m}/\text{deg.)}$

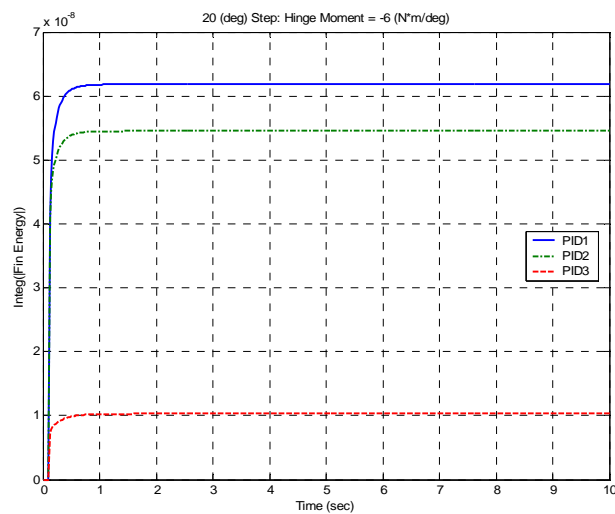


Figure 5.39. η : 20° Step Response, Hinge Moment = $-6 \text{ (N}\cdot\text{m}/\text{deg.)}$

As seen in figures 5.30 through 5.39, the value of η must be viewed together with the overall shape of the curve. In each of the cases presented above, *PIDI* exhibits the best response. The plot of η with respect to time for *PIDI* rises sharply in each case and then flattens off. This comparison shows that initially the energy put into the system is delivered directly to the fin, as much as possible. Once the fin is close to the commanded value, the controller then stops the fin motion.

For this analysis, no restrictions are made on the architecture of the controller, nor are there any restrictions on the linearity of the system. The only restriction is that the closed loop system be stable. Thus, this analysis provides a good method for comparing the efficiency of control schemes for a given system.

5.5.2 Step Response Comparisons of Non-Linear Systems

The same analysis is performed in this section for the complete nonlinear system. Nonlinear controllers 1 and 2, *NL1* and *NL2* respectively, are connected to the complete nonlinear motor model of figure 5.2. Similar to the Dymola model of figure 5.20, using the Dymola model of figure 5.2 in an object-oriented fashion allows the two control scheme simulations to be performed at once. This model is shown in figure 5.40. The normalization analysis block has not been included on the power signals, as it was for the linear systems of figure 5.20. Since the power signal analysis was shown to be of little use, it is not calculated. The models labeled *Fin_Cntrl1* and *Fin_Cntrl2* contain the nonlinear fin dynamics, together with *NL1* and *NL2*, respectively. Analysis for the same load conditions as the linear systems is performed and the results are shown.

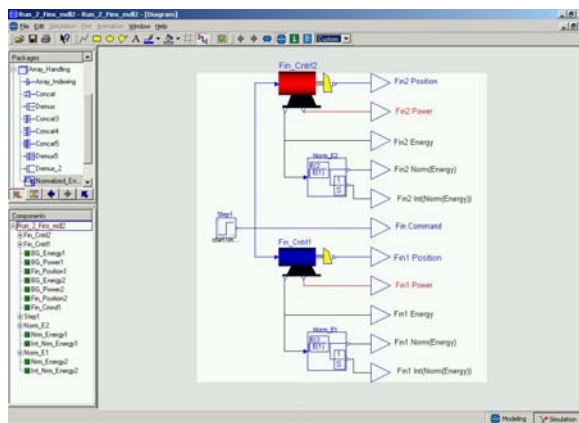


Figure 5.40. Two Non-Linear Actuators with Power Signal Analysis

Recall, that *NL1* is the discretized version of *PID1* with a quantization element and an output limit, shown in figure 5.15. *NL2* contains the anti-backlash element and considerable nonlinear logic as described by figures 5.16 and 5.17. Figure 5.41 shows the step response for the 5° step command with $0(\text{N}\cdot\text{m}/\text{deg})$ hinge moment load.

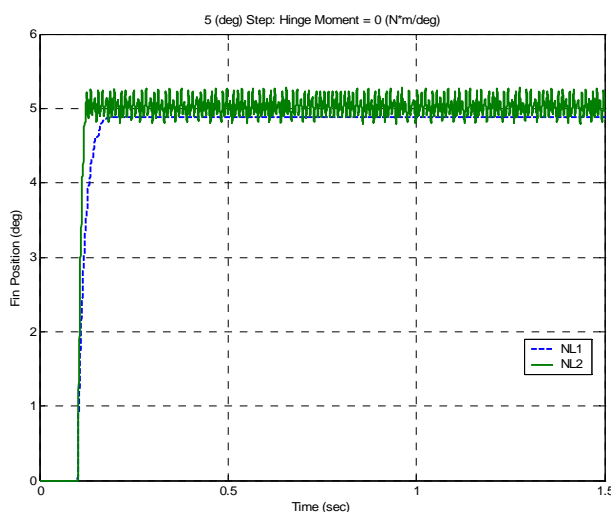


Figure 5.41. 5° Step Response, Hinge Moment = $0(\text{N}\cdot\text{m}/\text{deg})$

The backlash effects are most clearly visible in figures 5.41 through 5.43 near 0.15 seconds as NLI is rising. The stair step, as the fin moves into position, is caused by the backlash. For zero hinge moment load, the steady state error from NLI is caused by quantization of the control effort. The controller NLI is able to zero out the fin position within the quantization limit. After that, although the error input into the controller is non-zero, and the output of the controller is non-zero, it is not big enough to generate a quantized control effort. Figure 5.42 shows the input to the quantizer together with the quantized output.

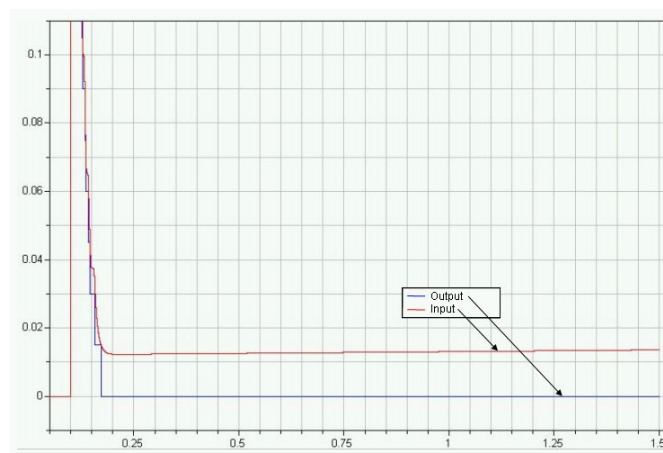


Figure 5.42. Quantizer I/O for 5° Step Response, No Hinge Moment

The input to the quantizer is the output of the discretized $PIDI$, as can be seen in figure 5.15. The positive slope on the quantizer input is from the integral term of $PIDI$. The error is slowly integrating up until it will eventually cause a small output. This small quantizer output is a very slow limit cycle that will keep the fin from reaching zero steady state error.

Figure 5.43 through 5.47 show the step responses and efficiency calculations for 5° step command with 0 (N*m/deg), -6 (N*m/deg) and -6 (N*m/deg) hinge moment load. The step response plots have been zoomed to view the controller's response near a 0° feedback error.

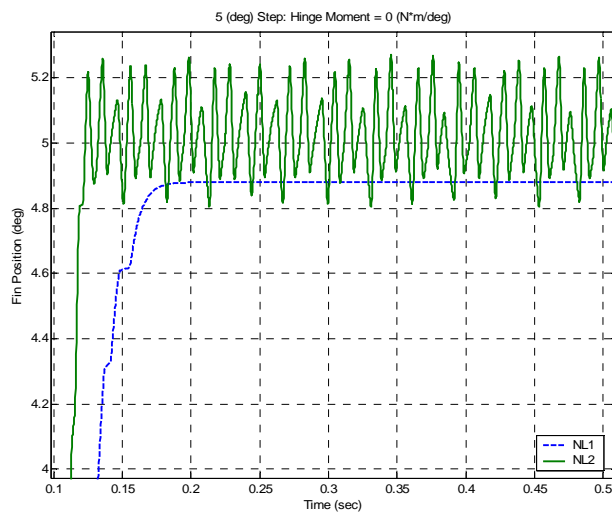


Figure 5.43. 5° Step Response (zoom), Hinge Moment = 0 (N*m/deg.)

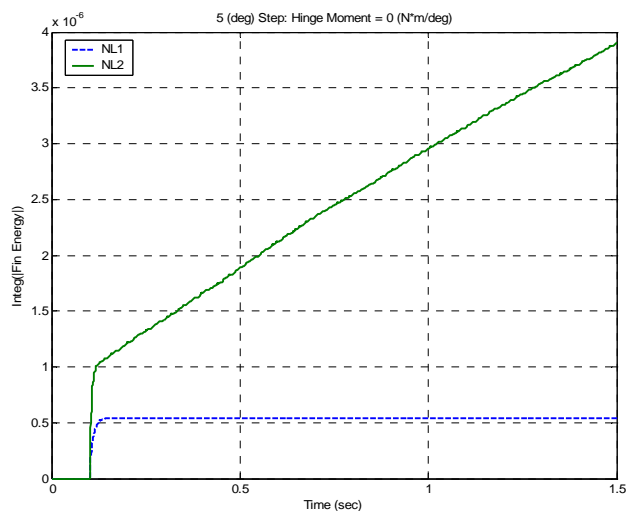


Figure 5.44. η : 5° Step Response, Hinge Moment = 0 (N*m/deg.)

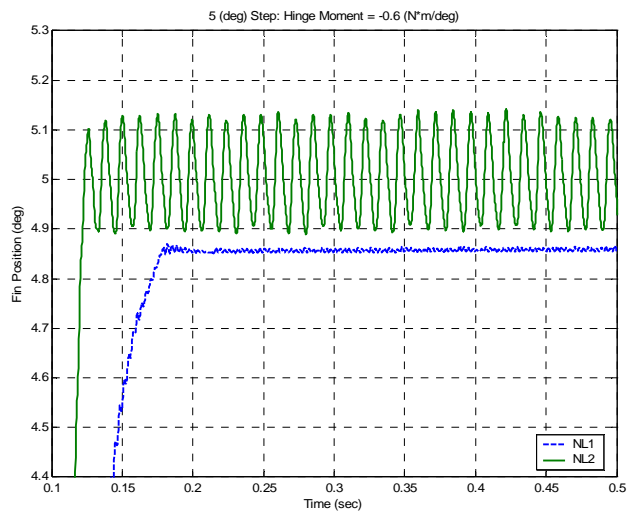


Figure 5.45. 5° Step Response, Hinge Moment = -0.6 ($\text{N}\cdot\text{m}/\text{deg}$.)

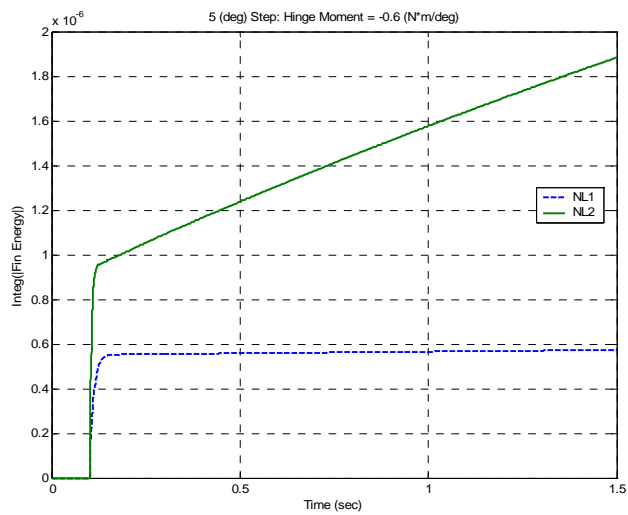


Figure 5.46. η : 5° Step Response, Hinge Moment = -0.6 ($\text{N}\cdot\text{m}/\text{deg}$.)

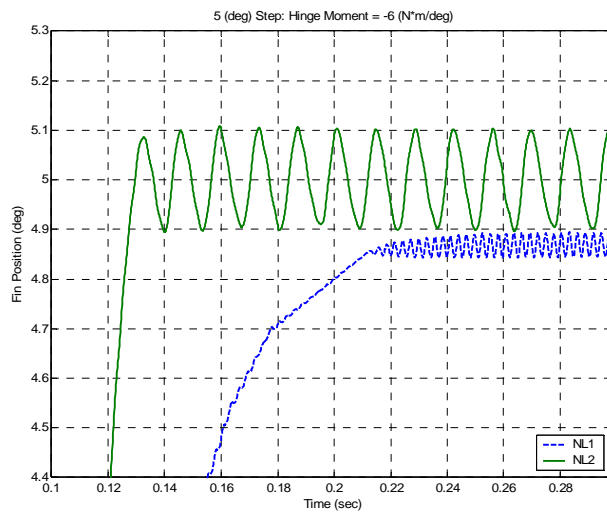


Figure 5.47. 5° Step Response, Hinge Moment = -6 (N*m/deg.)

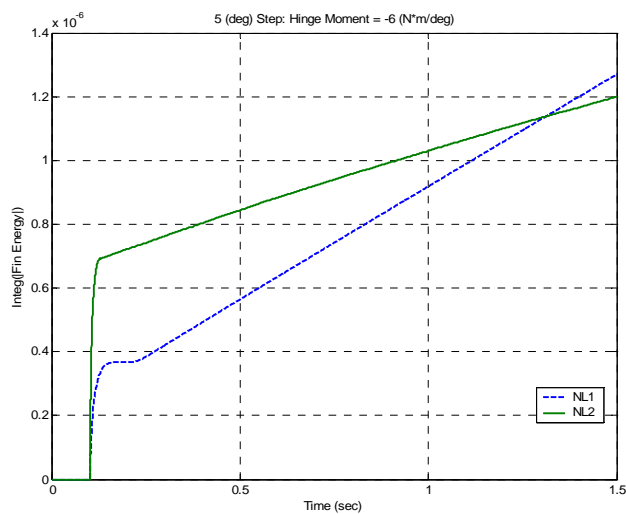


Figure 5.48. η : 5° Step Response, Hinge Moment = -6 (N*m/deg.)

The oscillation of the *NL1* response on the loaded configurations is caused by the external hinge moment moving the fin away from the commanded value. Thus, the controller cannot settle with the quantization limit as it does in the unloaded case.

The nonlinear elements of the *NL2* cause the oscillation about the commanded value, in figure 5.43, 5.45, and 5.47. Figures 5.44, 5.46 and 5.48 show that *NL2* makes good use of the input power initially to achieve a quick rise time, but continually use system power to sustain the oscillation about the commanded position. *NL1* shows more of an ideal use of input power for low to medium hinge moment load configurations. However, for large hinge moment load configurations, *NL2* is clearly superior, as seen in figure 5.48.

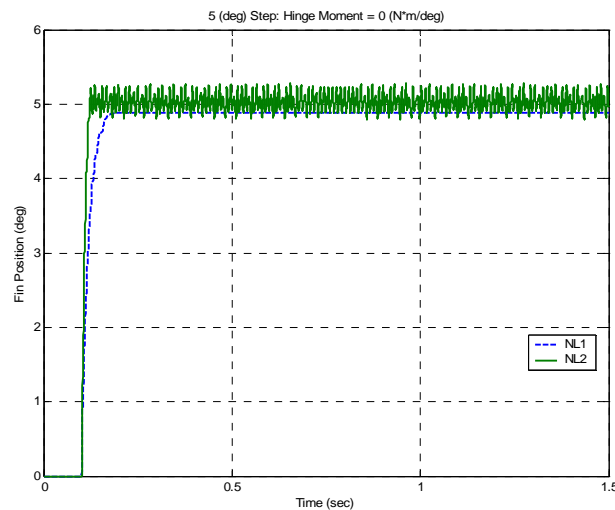


Figure 5.49. 20° Step Response, Hinge Moment = 0 (N*m/deg.)

Figures 5.49 through 5.54 show the step responses and efficiency calculations for 20° step command with 0 (N*m/deg), -6 (N*m/deg) and -6 (N*m/deg) hinge moment load.

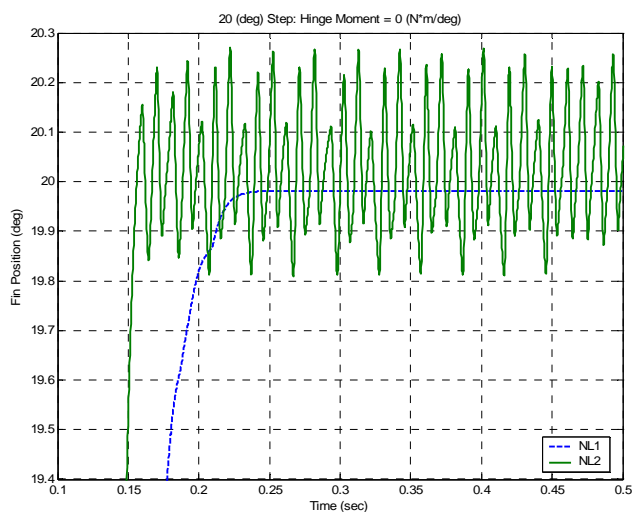


Figure 5.50. 20° Step Response (zoom), Hinge Moment = 0 (N*m/deg.)

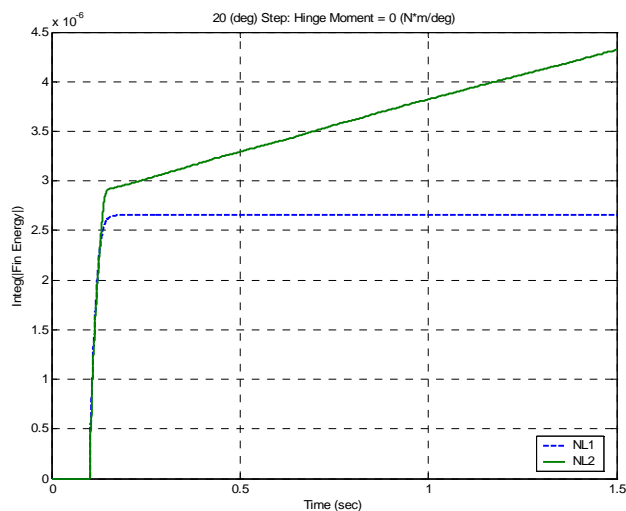


Figure 5.51. η : 20° Step Response, Hinge Moment = 0 (N*m/deg.)

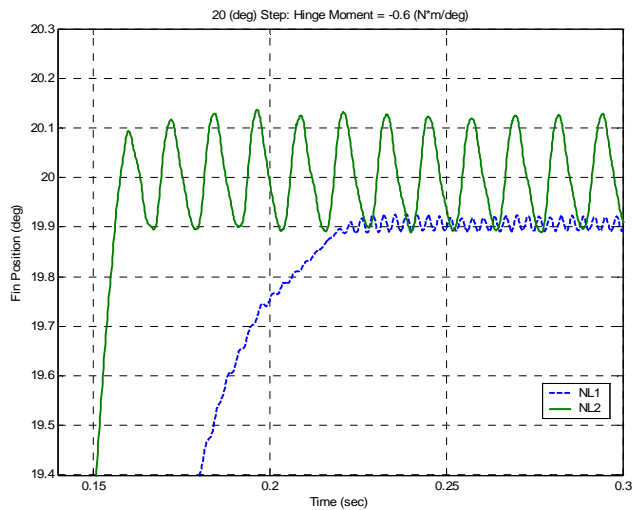


Figure 5.52. 20° Step Response (zoom), Hinge Moment = -0.6 (N*m/deg.)

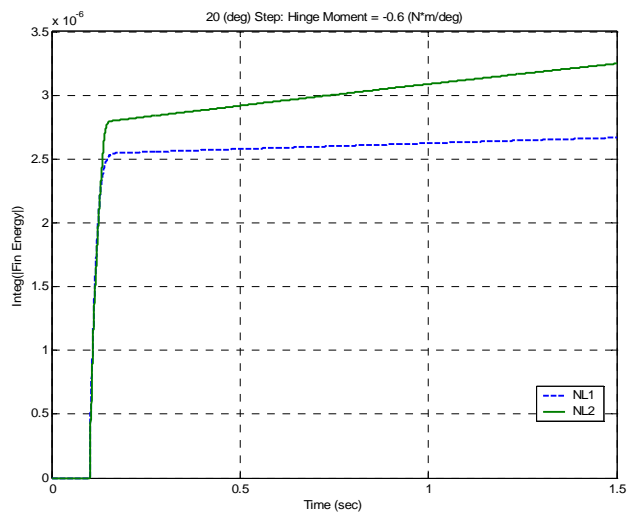


Figure 5.53. η : 20° Step Response, Hinge Moment = -0.6 (N*m/deg.)

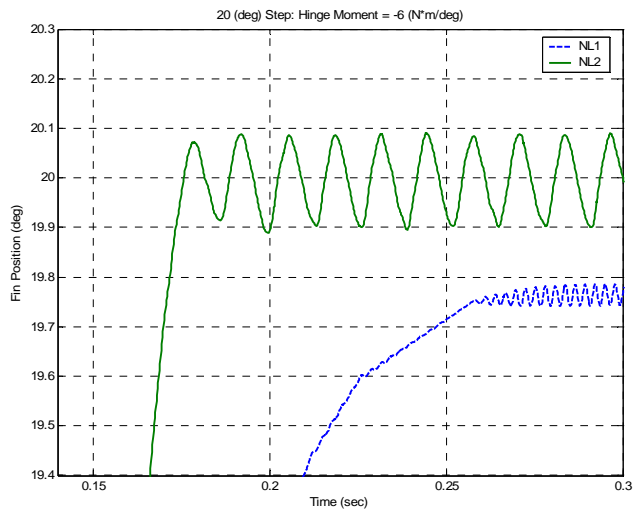


Figure 5.54. 20° Step Response (zoom), Hinge Moment = -6 (N*m/deg.)

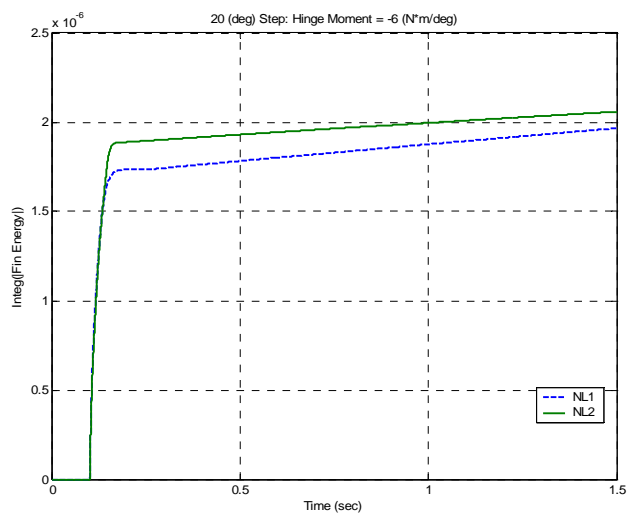


Figure 5.55. η : 20° Step Response, Hinge Moment = -6 (N*m/deg.)

The results for the 20° step inputs are identical to the 5° step input results. For large hinge moment loads, *NL2* makes better use of the input power. Also, although the oscillation about the commanded value is undesirable from a power consumption point of view, it at least ensures that the achieved fin position passes through the commanded value periodically, even with the backlash and quantization errors. *NL1* has non-zero steady state error.

5.6 Conclusions

This chapter presented a nonlinear actuation system that was built in Dymola with the bond-graph library of Chapter 4. Bond graph methods were used to linearize the system.

This chapter showed how the power flow through a bond graph model of the plant can be used to compare the effectiveness of different control schemes regardless of the architecture of the controller design, and without limiting the analysis to linear systems.

The controller efficiency was defined as $\eta_{controller} = \int_o^{tf} \left[\frac{OutputEnergy}{InputEnergy} \right] dt$ in equation

5.16. The 2nd law of thermodynamics was used to prove that the *InputEnergy* cannot be zero for any single-input physical system after an initial input has been given.

Separate control schemes were presented for both the linearized actuation system and the nonlinear system. The system response of each control scheme was compared using the definition of controller efficiency to show the ability of each controller to utilize the available energy in the system.

CHAPTER 6: Optimal Gain Comparison Using the Power Flow Information of Bond Graph Modeling

6.1 Introduction

This chapter applies the controller efficiency measurement, developed in Chapter 5, to an autopilot design. Chapter 5 was concerned with comparing two separate controllers of different architecture to determine which is more efficient. The same analysis can be applied to comparing two separate autopilots using the power flow through the actuator to determine the more efficient autopilot design. Also, power flow analysis is used in this chapter to determine the optimality of controller gains for a classical three loop autopilot.

Typically in the design of an autopilot, the actuator dynamics are ignored during the design process. Figure 6.1 shows a block diagram of a missile system. In the design of an autopilot, the sensor dynamics and actuator dynamics are omitted, as shown in figure 6.1. Often in a missile system, the power flow through the actuator limits the response of the system, since fin position applies the control effort that influences the body dynamics.

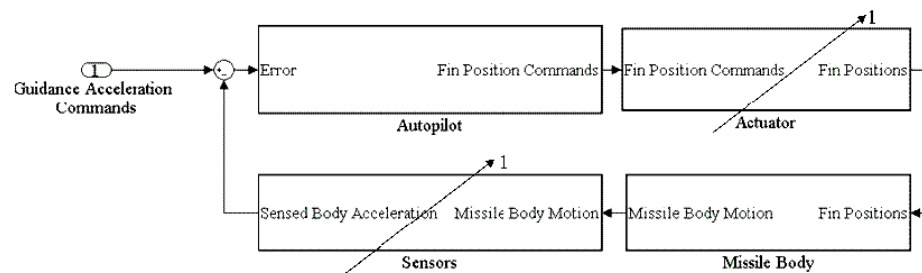


Figure 6.1. Autopilot Loop with the Autopilot Design Assumption

During the autopilot design process, the actuator dynamics are assumed to be ideal in that the commanded fin position (output from the autopilot) is the achieved fin position (input to the missile dynamics). Naturally, the system response will only worsen with the inclusion of the actuator dynamics. The actuator dynamics become a bottleneck for the system response, since the actuator has limited power flow. The autopilot forms a loop around these dynamics. Thus, the autopilot control structure will influence the actuator power flow. This insight provides a means for classifying the efficiency of the autopilot in the same fashion that was presented in Chapter 5. Autopilot efficiency can be defined by the actuator efficiency. Autopilot efficiency analysis can be used to compare different controller designs or to compare efficiencies of different gains within the same design.

In this chapter, the nonlinear fin positioning system of Chapter 5 is used to measure the controller efficiency of the autopilot. The nonlinear fin actuator of Section 5.4.3 is used as the actuation system throughout this chapter. As seen in figure 6.1, the autopilot loop encompasses the actuator controller/dynamics loop. Thus, the autopilot is another level removed from the power flow in the actuator.

In the design of a controller, the selection of controller gains is the most time consuming and ad hoc of tasks. The difficulty lies in the fact that optimization tools cannot always find global optima, thus the solution found is more than likely sub-optimal. However, trying to find a more optimal solution quickly becomes cost ineffective. The controller gain selection process lacks a method for measuring the balance between 'good' and 'good enough'. The question of whether the controller gains

need to be optimized further, or re-optimized in the instance of an existing design, often goes unanswered.

This chapter uses the efficiency calculation method of Chapter 5 for evaluating the efficiency of controller gains for a given control system. If the current set of controller gains makes the most efficient use of the actuator's available energy, compared to other controller gain sets, and these gains satisfy classical control criteria, then this set of gains is as close to the optimal solution as it needs to be. Further optimization would then be a waste of time and money, since the system performance cannot be improved. On the other hand, if the current set of gains does not make good use of the system's available energy, then the gains need to be optimized further [McB05a].

This chapter uses the servo-positioning system of Chapter 5 to control a two degree-of-freedom missile model. The controller discussion of Chapter 5 discussed primarily the control of the servo-system itself. Here, the controller discussion focuses on autopilot control of a missile system. The actuator control scheme is buried inside the missile/autopilot system.

6.2 Two Degree of Freedom Missile

A two degree of freedom missile is used in this chapter to apply the analysis of Chapter 5 to a missile/autopilot system. Full blown missile dynamics are not necessary to demonstrate the usefulness of the methods developed in this chapter. Explanation of the analysis is better done on a two degree of freedom (2DoF) missile system to keep the

complexity at a minimum. In this fashion, the usefulness of the analysis is not clouded by the complexity of a six degree of freedom (6DoF) system. Although a 2DoF is used here, this analysis has been tested using a complete missile 6DoF system [McB05b]. It was found that the 6DoF analysis produces similar results as the 2DoF analysis.

The missile dynamics model presented here can be found in the book *Tactical and Strategic Missile Guidance* by Zarchan [Zar02 pp. 461-465]. Figure 6.2 shows a schematic of a two degree of freedom missile.

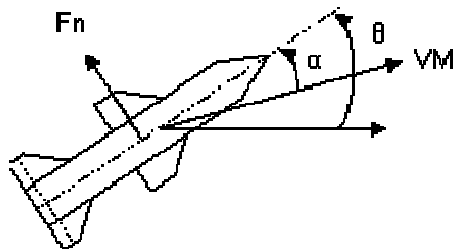


Figure 6.2. Two Degree of Freedom Missile

The two degrees of freedom, as described by figure 6.2, consist of a translational degree-of-freedom normal to the missile body, and a rotational degree-of-freedom about an axis coming out of the page. The normal force is described by equations 6.1 through 6.5.

$$m * A_z = Q * S_{ref} * C_N \quad (6.1)$$

$$Q = \frac{\rho * V_m^2}{2} \quad (6.2)$$

$$S_{ref} = \frac{\pi d^2}{4} \quad (6.3)$$

$$C_N = 2\alpha + \frac{1.5 * S_{plan} * \alpha^2}{S_{ref}} + \frac{8 * S_w * \alpha}{\beta * S_{ref}} + \frac{8 * S_T (\alpha + \delta)}{\beta * S_{ref}} \quad (6.4)$$

$$\beta = \sqrt{Mach^2 - 1} \quad (6.5)$$

A description of the variables is given in table 6.1.

Variable	Description	Variable	Description
m	Missile Mass	Mach	Missile Velocity with Respect to Sound
Az	Lateral Acceleration	β	Normalized Speed
Q	Dynamic Pressure	d	Missile Body Diameter
S_{ref}	Reference Area	ρ	Air Density
C_N	Normal Force Coefficient	δ	Tail Deflection
S_{plan}	Planform Area $\approx LM * d$	LM	Missile Body Length
S_w	Wing Area	X_{CPN}	Dist. from Nose Center of Pressure to Tip
S_T	Tail Area	X_{CG}	Dist. from the Center of Gravity to Tip
α	Angle of Attack	X_{CPB}	Dist. from Body Center of Pressure to Tip
θ	Missile Body Angle	X_{CPW}	Dist. from Wing Center of Pressure to Tip
V_m	Missile Velocity	X_{HL}	Dist. from Hinge Line to Tip

Table 6.1. Missile Dynamics Variable Description

Note that S_{plan} is approximated by the length of the missile multiplied by the missile body diameter. Also note that, although axial motion of the missile is not a degree of freedom, it is still necessary to define the missile's velocity in this direction in order to properly calculate the normal force of the missile.

The moment is described by

$$I_{yy} * \ddot{\theta} = Q * S_{ref} * d * C_M \quad (6.6)$$

and

$$C_M = 2\alpha \left(X_{cg} - X_{CPN} \right) + \frac{1.5 * S_{plan} * \alpha^2}{S_{ref}} \left(X_{cg} - X_{CPB} \right) + \frac{8 * S_w * \alpha}{\beta * S_{ref}} \left(X_{cg} - X_{CPW} \right) + \frac{8 * S_T (\alpha + \delta)}{\beta * S_{ref}} \left(X_{cg} - X_{HL} \right) \quad (6.7)$$

The missile distance values are described graphically in figure 6.3.

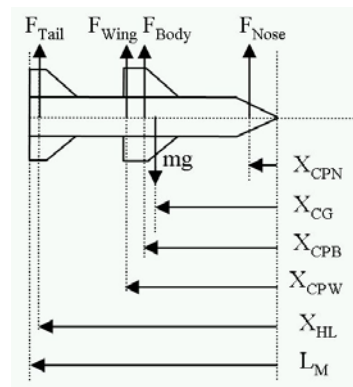


Figure 6.3. Missile Distance Definitions

The relationship between the angle of attack and the missile body angle is given by

$$\dot{\alpha} = \dot{\theta} - \frac{A_z}{V_m} \quad (6.8)$$

As seen by equations 6.1-6.8, the 2 degree-of-freedom missile is described by a set of non-linear ODE's. The bond graph representation of this system is shown in figure 6.4.

Figure 6.5 shows a Dymola model of the pitch plane dynamics, diagram window and icon window.

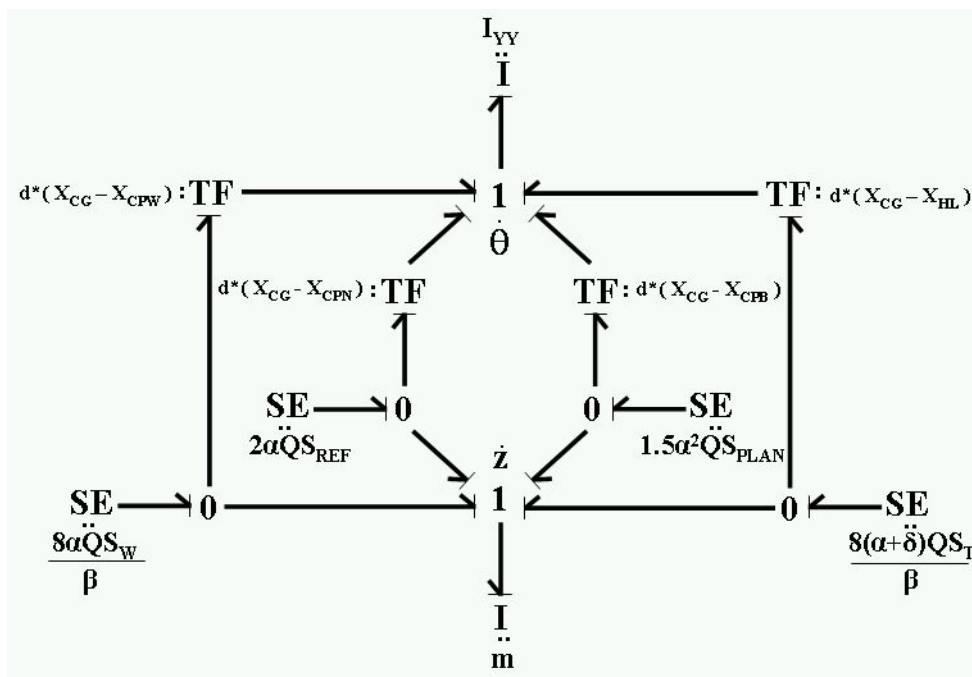


Figure 6.4. Missile Pitch Plane Dynamics Bond Graph

Note that in figure 6.5 sensors have been added to the bond graph to detect the flows on the 1-junctions and to detect the effort on missile mass. These values are used in the equation window to calculate the angle of attack, which is fed back into the effort sources as described by equations 6.4 and 6.7.

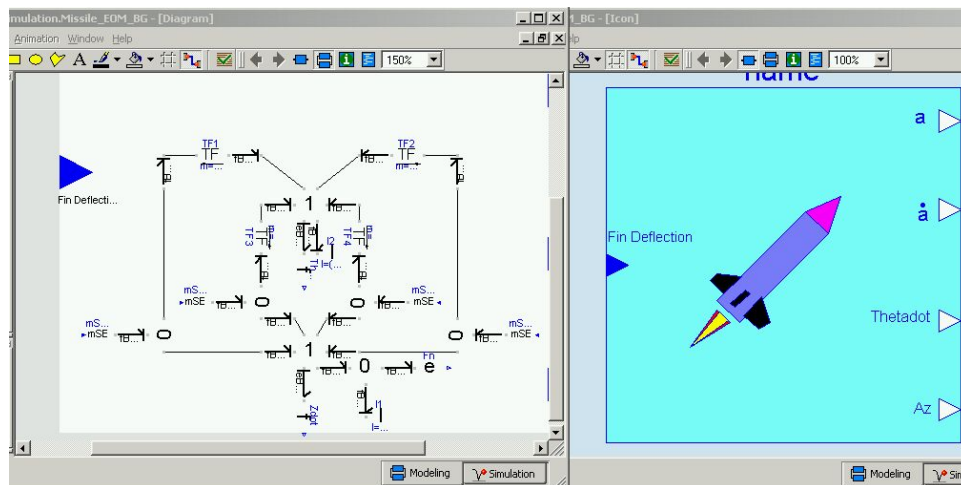


Figure 6.5. Dymola Pitch Plane Dynamics: Diagram and Icon Windows

Figure 6.6 shows the equation window. The equation window has been broken into two sections: the section on the left is the upper portion of the window showing parameter and variable declarations, the section on the right is the lower portion of the window showing the equations used to execute the model.

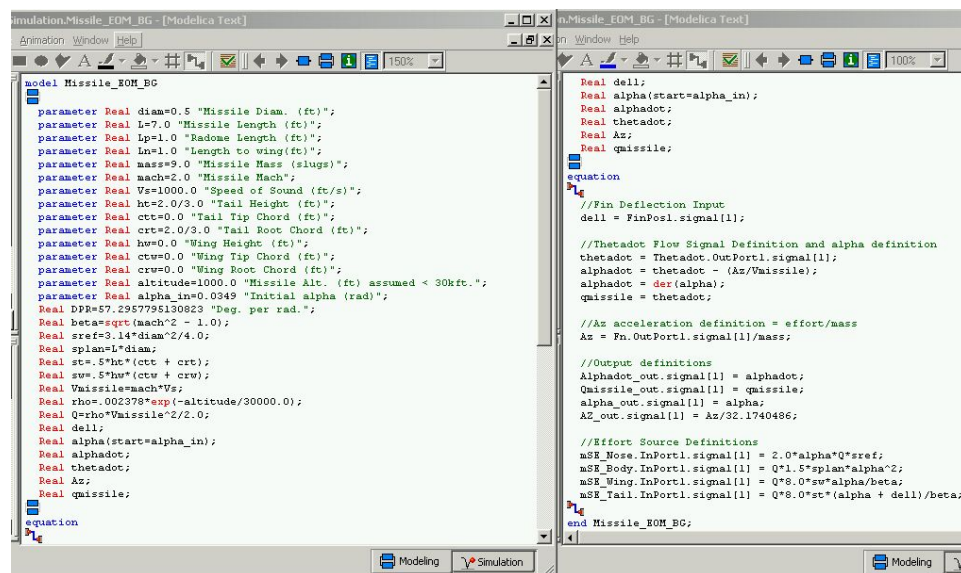


Figure 6.6. Dymola Pitch Plane Dynamics: Equation Window

The equations used to execute the model use the fin deflection input, and body motion values of the bond graph, to calculate the effort source inputs. The effort source signals are then sent back into the bond graph. In this fashion, the effort sources are modulated with the angle of attack value. Also shown in figure 6.6 are a number of outputs. These output signals are used to pass information to an upper hierarchical level.

Note the parameters, on the left portion of figure 6.6, are declared using default values. These default values will not be used in this chapter. The parameter values that will be used throughout this chapter are shown in figure 6.7. Figure 6.7 is a screen capture of the parameter assignment window that is activated upon instantiation of the pitch plane dynamics model.

The screenshot shows a window titled "Missile_EDM_BG in test_airframes". It has two tabs: "General" and "Add Modifiers". The "General" tab is active. The window is divided into three sections: Component, Model, and Parameters.

Component Section:

- Name: Missile_EDM_BG
- Comment: (empty)

Model Section:

- Path: AP_Simulation.Missile_EDM_BG
- Comment: (empty)

Parameters Section:

Parameter	Value	Description
diam	1	Missile Diam. (ft)
L	20	Missile Length (ft)
Lp	3	Radome Length (ft)
Ln	4	Length to wing(ft)
mass	31	Missile Mass (slugs)
mach	3	Missile Mach
Vs	1000	Speed of Sound (ft/s)
ht	2	Tail Height (ft)
ctt	0	Tail Tip Chord (ft)
crit	2	Tail Root Chord (ft)
hw	2	Wing Height (ft)
ctw	0	Wing Tip Chord (ft)
ctw	6	Wing Root Chord (ft)
altitude	0	Missile Alt. (ft) assumed < 30kft.
alpha_in	0	Initial alpha (rad)

At the bottom of the window are "OK" and "Cancel" buttons.

Figure 6.7. Dymola Pitch Plane Dynamics: Parameter Values

The parameter values used in this chapter were also taken from Zarchan [Zar02 pp 465-466]. Note that in figures 6.6, and 6.7, a parameter for the initial angle of attack is included. This value is used in the equation window as an initial condition on the integration of α . Also, figure 6.6 and figure 6.7 show that the model uses chord lengths. These dimensions are described in figure 6.8.

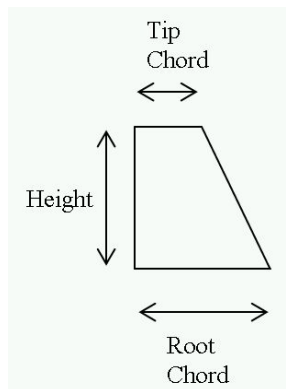


Figure 6.8. Wing and Fin Chord Definitions

Figure 6.9 shows a Dymola instantiation of the pitch plane dynamics bond graph. This upper hierarchical level was used to subject the 2DoF model to a fin deflection of 5° .

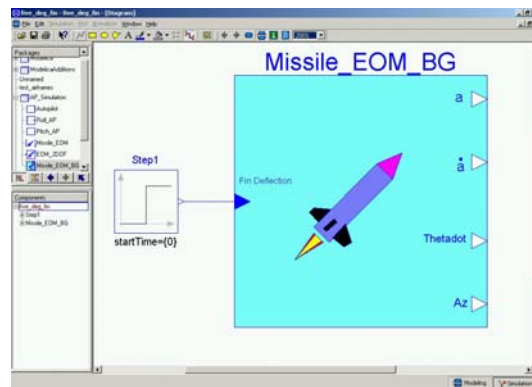


Figure 6.9. Dymola Pitch Plane Instantiation

Angle of attack and body acceleration results for two seconds of simulation are shown in figures 6.10 and 6.11, respectively.

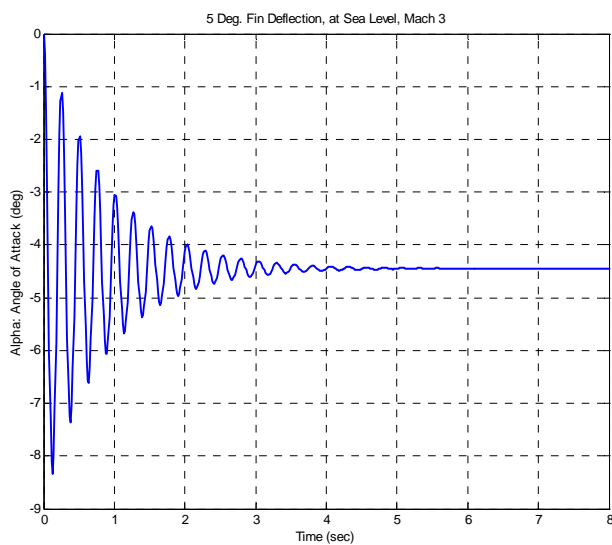


Figure 6.10. Angle of Attack

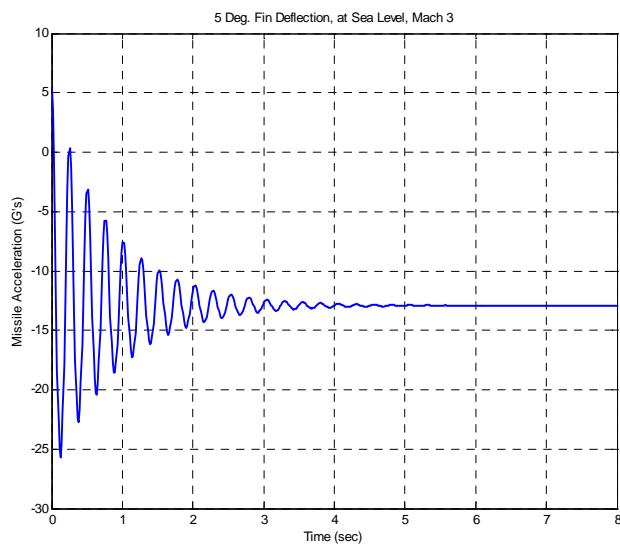


Figure 6.11. Missile Body Acceleration

Figures 6.10 and 6.11 show that without an autopilot, a 5° fin deflection will eventually result in -12.92 G's of body acceleration. The system is stable, but very oscillatory. The angle of attack settles down to about -4.448° degrees after about 2 seconds. The inclusion of an autopilot can improve the response of this system by reducing the amount of ringing and reducing the settling time.

6.3 Linear Pitch Autopilot

6.3.1 Missile Pitch Autopilot: 3-Loop Controller

A classic three loop autopilot [Zar02 pp. 507-509] will be used throughout this chapter. The classic three loop design is shown in figure 6.12 which is drawn in the form of figure 6.1.

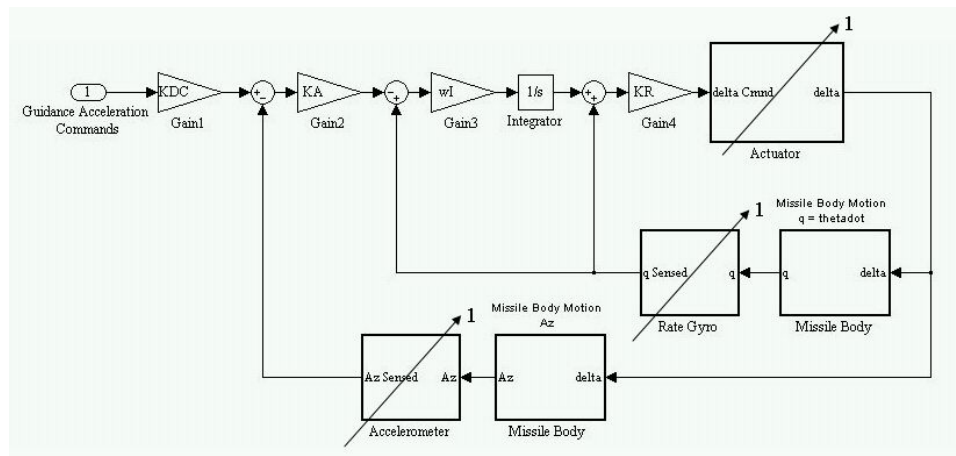


Figure 6.12. Classic Three Loop Autopilot

A Dymola model of the above autopilot is shown in figure 6.13, with the diagram window on the left and the icon window on the right.

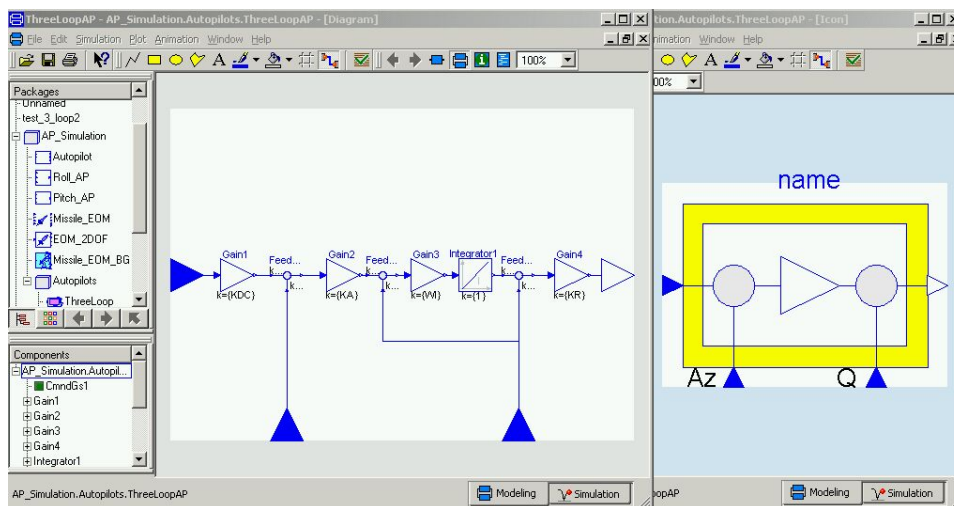


Figure 6.13. Classic Three Loop Autopilot: Dymola Model

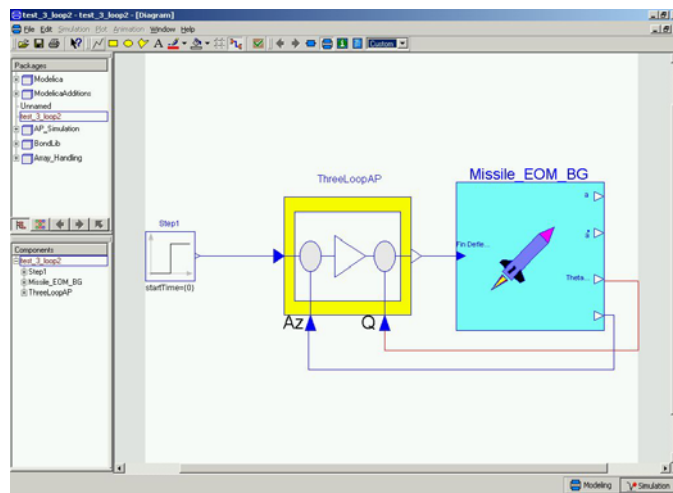


Figure 6.14. Three Loop AP: Closed Loop System

The pitch plane dynamics/autopilot connection is shown in figure 6.14. This system is connected in the form of figure 6.12 in that the actuator and sensor dynamics have been omitted from the closed loop system. The omission of these dynamics implies that the system is assumed ideal during the autopilot design. The system in figure 6.14 was simulated using controller gains $K_A=0.06493$ rad/(G*s), $W_I = 11.2$ rad/s, $K_R = 0.098$ s, and $K_{DC} = 1.165$. A 12.92 G step command was used as the input, which allows a comparison of missile response to figures 6.10 and 6.11. Angle of attack and achieved G's are shown in figures 6.15 and 6.16, respectively. Also, the control effort of fin position is shown in figure 6.17.

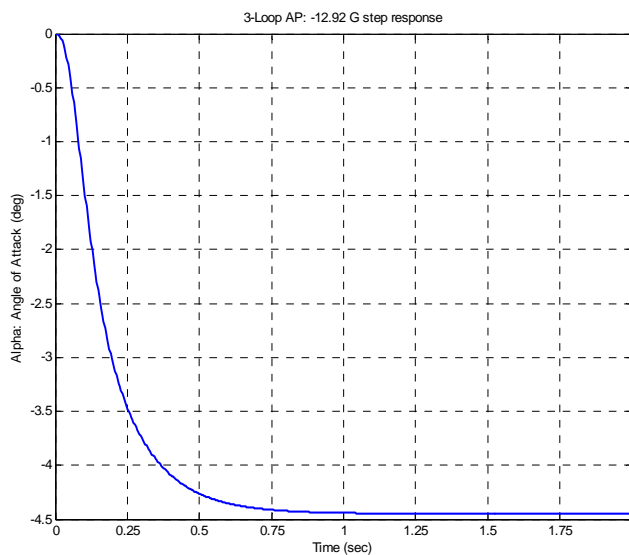


Figure 6.15. Three Loop AP: Angle of Attack

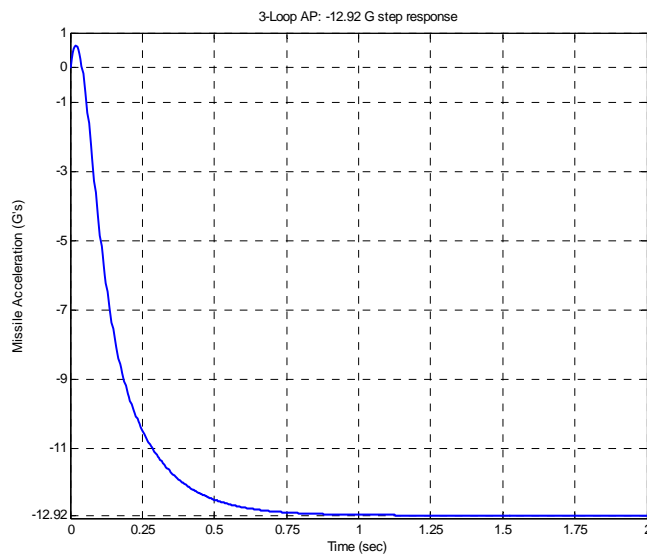


Figure 6.16. Three Loop AP: Achieved Acceleration

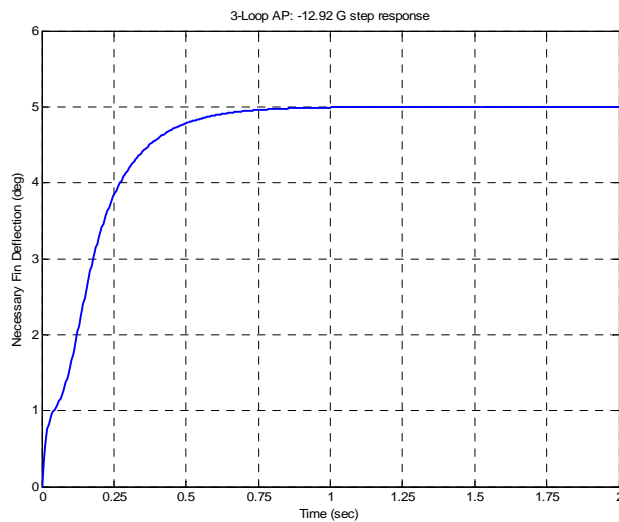


Figure 6.17. Three Loop AP: Necessary Fin Deflection

The achieved acceleration response with the 3-loop autopilot, in figure 6.16, is considerably better than the open loop response of figure 6.11. The overshoot is removed and the settling time is reduced considerably with the closed loop system. Figure 6.15 shows that the angle of attack behaves much better than the open loop response of figure 6.10. Figure 6.17 verifies that a 5° fin deflection is still necessary to achieve the desired amount of acceleration with or without the controller.

6.3.2 Missile Pitch Autopilot with Actuator Dynamics

Naturally the step response shown in Section 6.3.1 will only worsen with the inclusion of the actuator dynamics. As mentioned previously, the actuator dynamics become a bottleneck for the system response, since the actuator has limited power flow. The autopilot forms a loop around these dynamics. Thus, the autopilot control structure, and choice of gains, influences the actuator power flow.

Figure 6.18 shows the Dymola model of figure 6.14, including the actuator dynamics. Note that the vector normalization code of figure 5.19 has been included in the closed loop autopilot model to calculate the efficiency of the actuator power flow. The gain between the autopilot and the actuator is to convert the actuator input to degrees. The gain between the actuator and the body dynamics model is to convert the fin position to radians. The actuator model is the non-linear controller 2 of figure 5.40 that contains the anti-backlash element. Recall that this model contains a hinge moment model proportional to fin deflection. For this simulation, the hinge moments were set at $-0.6 \text{ N*m}/(\text{Deg of fin deflection})$.

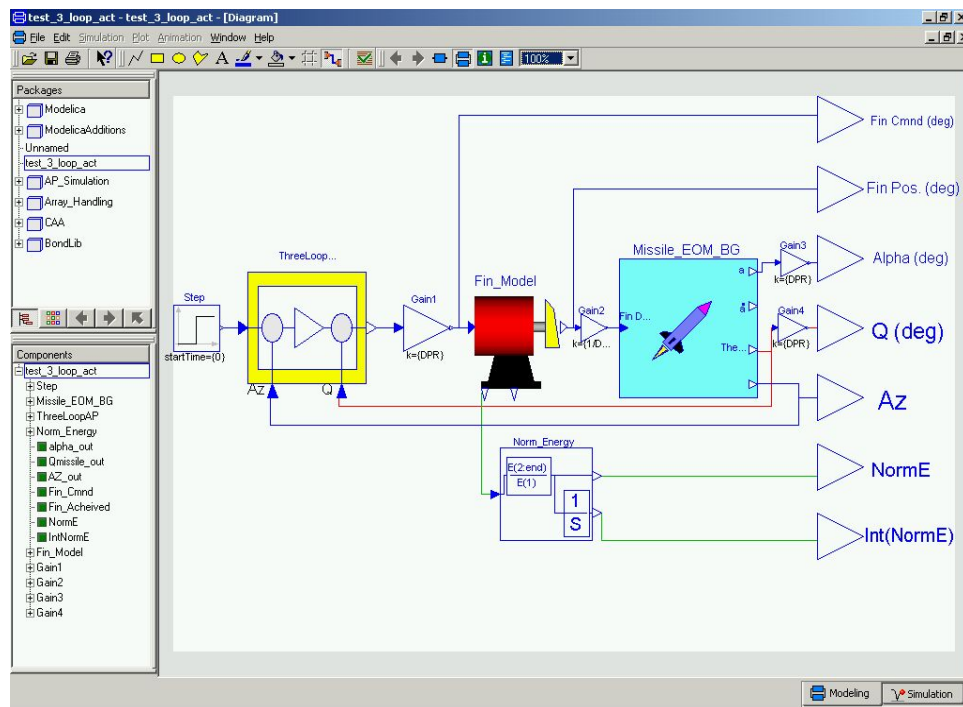


Figure 6.18. Three Loop AP: Closed Loop System with Actuator

The -12.92 G step response simulation was repeated for the above system. Figure 6.19 through 6.21 show angle of attack, achieved acceleration, and achieved fin position for the system of figure 6.18. These plots contain the results of the previous section, included as over plots, for reference. Thus, the influence of the actuator dynamics can be seen by looking at plots 6.19 through 6.21 without referring to figures 6.15 through 6.17.

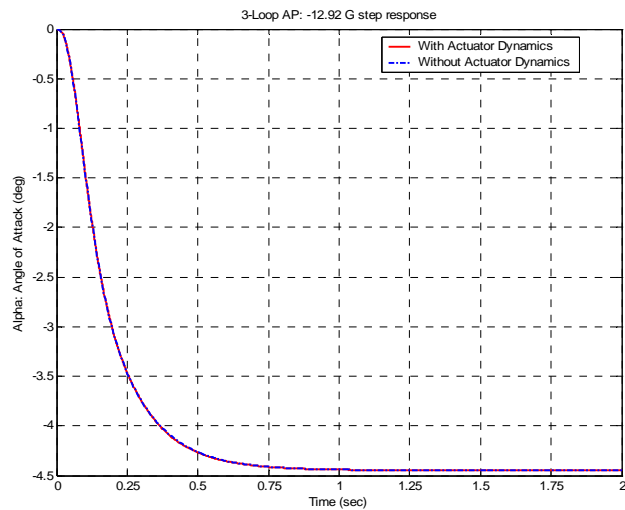


Figure 6.19. Angle of Attack with Actuator Dynamics

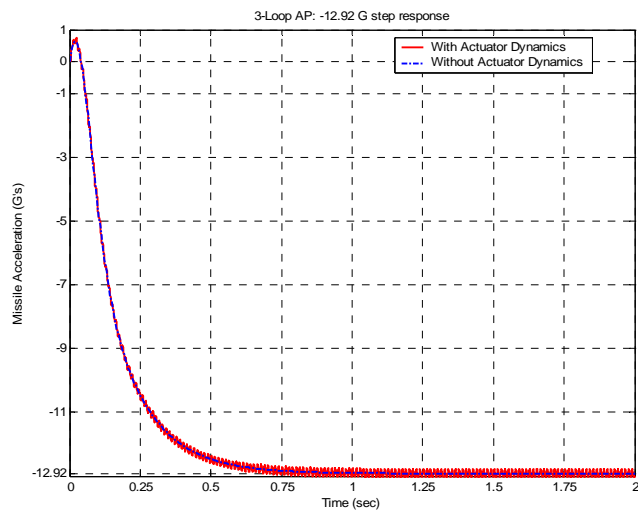


Figure 6.20. Achieved Acceleration with Actuator Dynamics

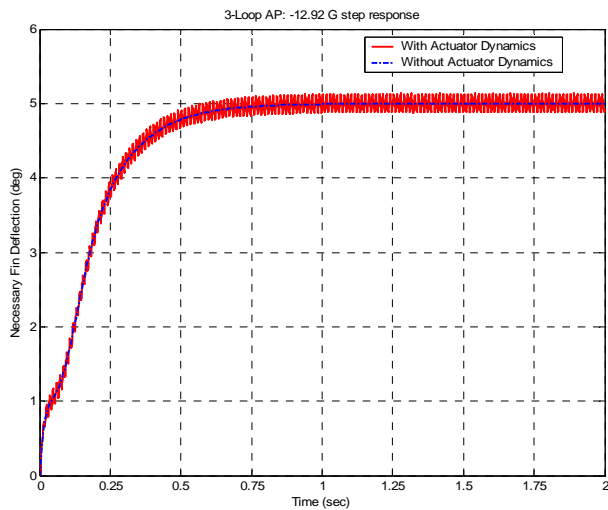


Figure 6.21. Necessary Fin Deflection with Actuator Dynamics

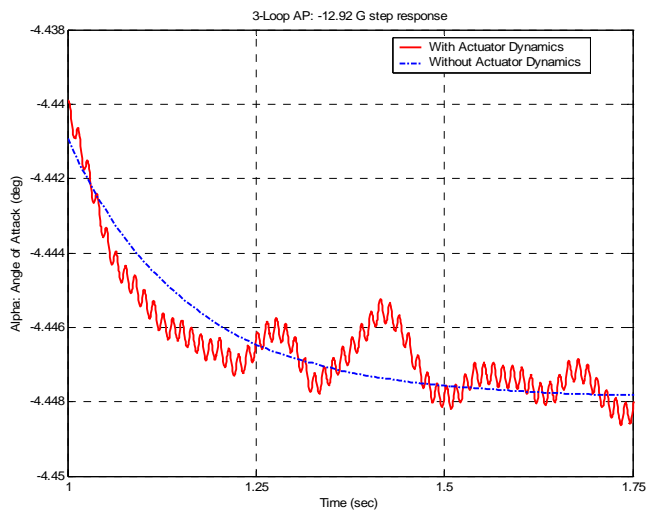


Figure 6.22. Angle of Attack with Actuator Dynamics: Zoom

Figures 6.22 through 6.24 are a repeat of figures 6.19 through 6.21 zoomed in to point out the detailed difference between the ideal actuator dynamics and the nonlinear actuator model.

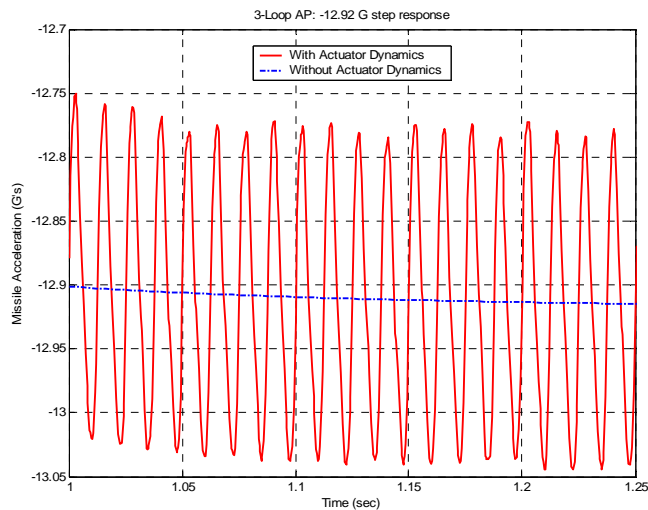


Figure 6.23. Achieved Acceleration with Actuator Dynamics: Zoom

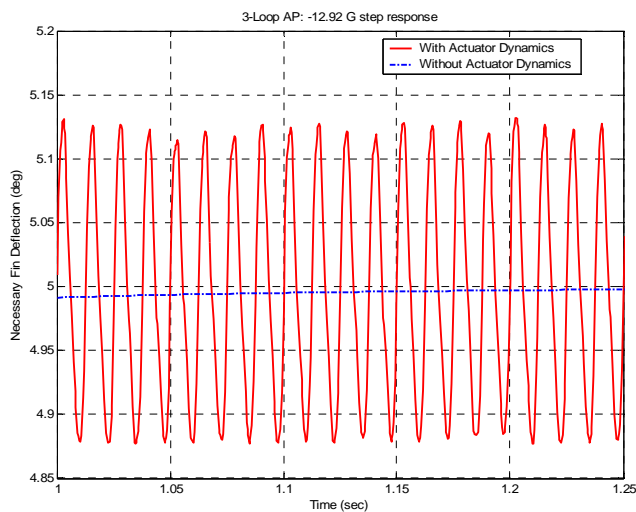


Figure 6.24. Necessary Fin Deflection with Actuator Dynamics: Zoom

The ~ 80 Hz frequency, seen in all three plots, is due to the nonlinearities of the actuator. Obviously, the autopilot gain selection does not drive the actuator near its power limit since figures 6.22 through 6.24 show the actuator dynamics are much faster

than the missile response. Proper gain selection will bring the missile response time much closer to the actuator response time.

6.4 The Autopilot Gain Selection Process

As seen in figure 6.1, during the autopilot design process the dynamics of the actuator are assumed ideal. Thus, actuator dynamics do not form a part of the gain selection process.

6.4.1 The Autopilot Gain Selection Process: The Performance Index

Typically, the autopilot gain selection process involves the minimization of a constrained performance index. A common performance index, PI , is shown in equation 6.9. This performance index is meant to be minimized [Clo96b].

$$PI = \int_0^{tf} \left[w_1 (A_{ZC} - A_{ZA})^2 + w_2 (A_{YC} - A_{YA})^2 + w_3 P^2 \right] dt \quad (6.9)$$

A_{YC} and A_{ZC} are the commanded accelerations in the Y and Z directions, respectively. Similarly, A_{YA} and A_{ZA} are the achieved accelerations in Y and Z directions. P is the missile roll rate, and w_1 - w_3 are the respective weights. Equation 6.9 shows the performance index for a 6DoF missile model. For the 2DoF missile model, equation 6.9 reduces to

$$PI = \int_0^{tf} \left[(A_{YC} - A_{YA})^2 \right] dt \quad (6.10)$$

Typical constraints for this optimization are shown in equations 6.11 through 6.13.

$$\text{Gain Margin} > 3dB \quad (6.11)$$

$$\text{Phase Margin} > 20^\circ \quad (6.12)$$

$$\begin{aligned} \text{Overshoot} < 20\% \\ \text{Undershoot} < 30\% \end{aligned} \quad (6.13)$$

For a step response y , overshoot is defined as $(\max(y) - \text{step command})/\text{step command}$, and undershoot is defined as $\text{abs}(\min(y))/\text{step command}$.

Obviously, the constraints of gain and phase margin imply that the system is linear. Linearization of the missile pitch plane dynamics model is therefore necessary.

6.4.2 Linearized Missile Pitch Dynamics for Gain Optimization

6.4.2.1 Pitch Plane Linearization

Linearization of the pitch plane dynamics is relatively straightforward. Equation 6.4 can be written as

$$C_N = \alpha \left[2 + \frac{1.5 * S_{plan} * \alpha}{S_{ref}} + \frac{8 * S_w}{\beta * S_{ref}} + \frac{8 * S_T}{\beta * S_{ref}} \right] + \delta \frac{8 * S_T}{\beta * S_{ref}} = \alpha * C_{N\alpha} + \delta * C_{N\delta} \quad (6.14)$$

Where

$$C_{N\alpha} = 2 + \frac{1.5 * S_{plan} * \alpha}{S_{ref}} + \frac{8 * S_w}{\beta * S_{ref}} + \frac{8 * S_T}{\beta * S_{ref}} \quad (6.15)$$

and

$$C_{N\delta} = \frac{8 * S_T}{\beta * S_{ref}} \quad (6.16)$$

Equation 6.15 shows that $C_{N\alpha}$ is not constant but a function of α . The angle γ is defined as

$$\gamma = \theta - \alpha \quad (6.17)$$

For a small angle assumption for both angles α and θ , it is seen, from figure 6.2, that

$$\gamma \approx \sin \gamma \approx \frac{V_z}{V_m} \quad (6.18)$$

Therefore, for constant missile velocity, the turning rate $\dot{\gamma}$ can be approximated by

$$\dot{\gamma} \approx \frac{A_z}{V_m} \quad (6.19)$$

Solving equation 6.1 for A_z and substituting the result into equation 6.19 yields

$$\dot{\gamma} \approx \frac{Q^* S_{ref}^* C_N}{m^* V_m} \quad (6.20)$$

Substituting equation 6.14 into 6.20

$$\dot{\gamma} \approx \frac{Q^* S_{ref}^* [\alpha^* C_{N\alpha} + \delta^* C_{N\delta}]}{m^* V_m} = -Z_\alpha^* \alpha - Z_\delta^* \delta \quad (6.21)$$

Where

$$Z_\alpha = -\frac{Q^* S_{ref}^* C_{N\alpha}}{m^* V_m} \quad (6.22)$$

and

$$Z_\delta = -\frac{Q^* S_{ref}^* C_{N\delta}}{m^* V_m} \quad (6.23)$$

Rearranging equation 6.17, differentiating with respect to time, and substituting equation 6.21:

$$\dot{\alpha} = \dot{\theta} - \dot{\gamma} = \dot{\theta} + Z_{\alpha} * \alpha + Z_{\delta} * \delta \quad (6.24)$$

The moment coefficient equation 6.7 can be written in a similar form as equation 6.14.

$$C_M = \alpha * C_{M\alpha} + \delta * C_{M\delta} \quad (6.25)$$

Where

$$\begin{aligned} C_{M\alpha} = & 2 \left(X_{cg} - X_{CPN} \right) + \frac{1.5 * S_{plan} * \alpha}{S_{ref}} \left(X_{cg} - X_{CPB} \right) \\ & + \frac{8 * S_w}{\beta * S_{ref}} \left(X_{cg} - X_{CPW} \right) + \frac{8 * S_T}{\beta * S_{ref}} \left(X_{cg} - X_{HL} \right) \end{aligned} \quad (6.26)$$

and

$$C_{M\delta} = \frac{8 * S_T}{\beta * S_{ref}} \left(X_{cg} - X_{HL} \right) \quad (6.27)$$

Using equation 6.25, equation 6.6 can be written

$$\ddot{\theta} = \frac{Q * S_{ref} * d * C_M}{I_{yy}} = \frac{Q * S_{ref} * d}{I_{yy}} \left[\alpha * C_{M\alpha} + \delta * C_{M\delta} \right] \quad (6.28)$$

Equation 6.28 can be simplified by writing

$$\ddot{\theta} = \alpha * M_{\alpha} + \delta * M_{\delta} \quad (6.29)$$

Where

$$M_{\alpha} = \frac{Q * S_{ref} * d * C_{M\alpha}}{I_{yy}} \quad (6.30)$$

and

$$M_{\delta} = \frac{Q * S_{ref} * d * C_{M\delta}}{I_{yy}} \quad (6.31)$$

Note that M_{α} and Z_{α} are both functions of α . M_{δ} and Z_{δ} are constants. However, equations 6.24 and 6.29 form a linear set of equations only if all four parameters M_{α} , M_{δ} , Z_{α} , and Z_{δ} can be treated as constants. In order to bypass this difficulty, a trim condition is defined. The missile trim condition is defined as the combination of α and δ that create zero moment on the missile body. Thus, for a given δ , equation 6.7 is set to 0 and solved for α . This value of α is used to calculate M_{α} and Z_{α} , which are then treated as constants, making the equation set linear about the trim condition.

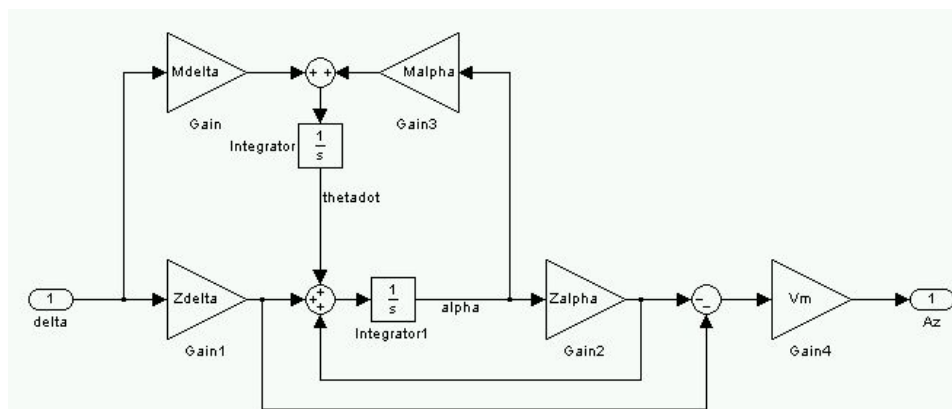


Figure 6.25. Linearized Pitch Plane Block Diagram

Figure 6.25 shows the linear set of equations in block diagram form. The state space equations are shown in equation 6.32 and 6.33.

$$\begin{bmatrix} \ddot{\theta} \\ \ddot{\alpha} \end{bmatrix} = \begin{bmatrix} 0 & M_\alpha \\ 1 & Z_\alpha \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \alpha \end{bmatrix} + \begin{bmatrix} M_\delta \\ Z_\delta \end{bmatrix} \delta \quad (6.32)$$

$$A_Z = \begin{bmatrix} 0 & -Z_\alpha V_M \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \alpha \end{bmatrix} + \begin{bmatrix} -Z_\delta V_M \end{bmatrix} \delta \quad (6.33)$$

Figures 6.26 and 6.27 show a comparison of the open loop, linear system to the open loop nonlinear system. The nonlinear signals are those presented in figures 6.10 and 6.11, respectively. α_{trim} is -4.4489° , for $\delta = 5^\circ$, *Mach* 3, and *Altitude* 0.

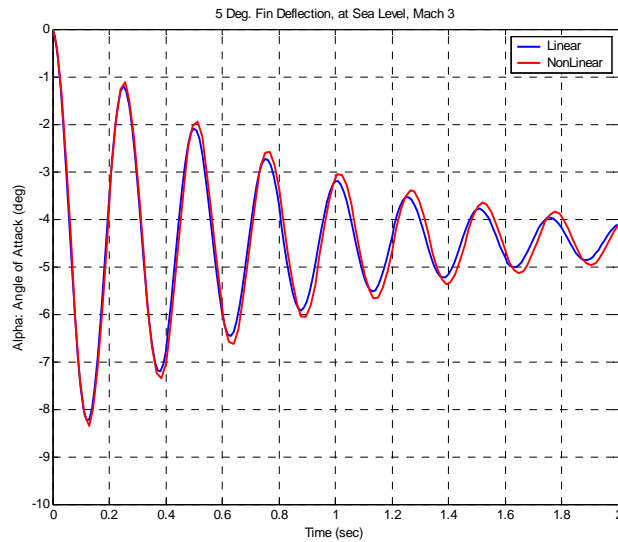


Figure 6.26. Open Loop Linear and Nonlinear Angle of Attack

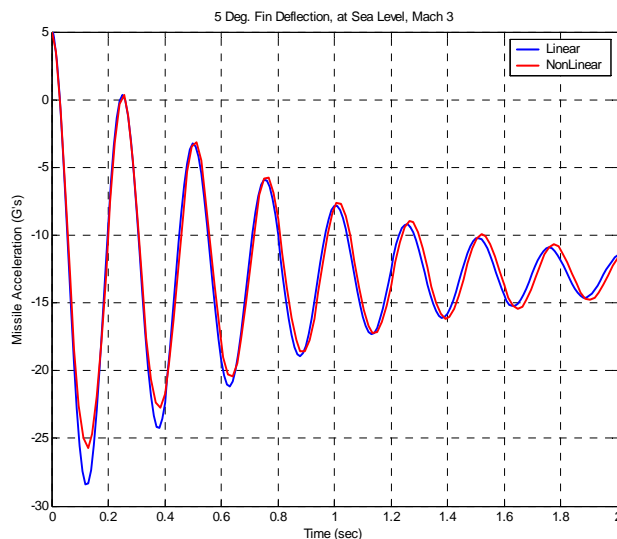


Figure 6.27. Open Loop Linear and Nonlinear Missile Body Acceleration

Figures 6.26 and 6.27, show a reasonable match for the linear system versus the nonlinear system.

6.4.2.2 Linearized Pitch Plane and Three Loop Autopilot

Figure 6.28 shows the block diagram of the linear system within the three loop autopilot. This system is a 3rd order system that has state space equations shown in equations 6.34 and 6.35.

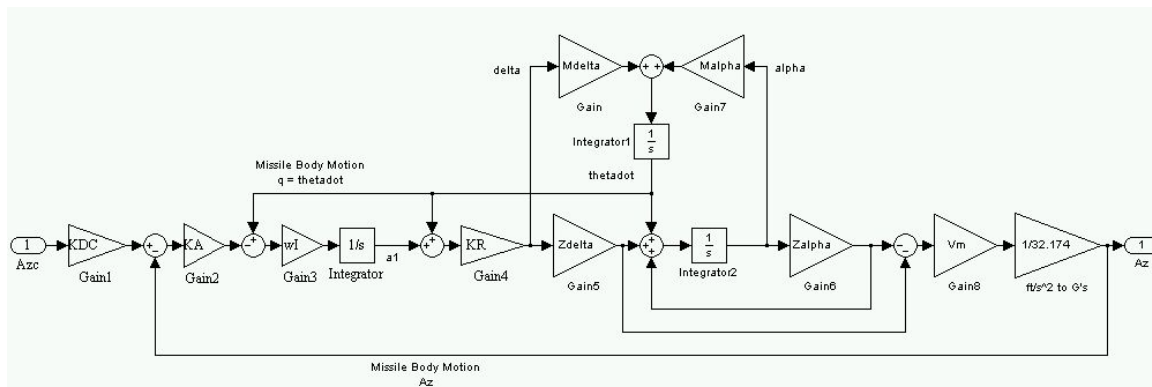


Figure 6.28. Linear Pitch Plane and Autopilot Block Diagram

$$\begin{aligned}
 \begin{bmatrix} \dot{a}_1 \\ \dot{q} \\ \dot{\alpha} \end{bmatrix} &= \begin{bmatrix} \frac{-WI * KA * Z_\delta * KR * V_M}{g} & WI - \frac{WI * KA * KR * Z_\delta * V_M}{g} & \frac{-WI * KA * Z_\alpha * V_M}{g} \\ M_\delta * KR & M_\delta * KR & M_\alpha \\ Z_\delta * KR & 1 + Z_\delta * KR & Z_\alpha \end{bmatrix} \begin{bmatrix} a_1 \\ q \\ \alpha \end{bmatrix} + \\
 &+ \begin{bmatrix} -WI * KA * KDC \\ 0 \\ 0 \end{bmatrix} A_{ZC}
 \end{aligned}
 \tag{6.34}$$

$$A_Z = \begin{bmatrix} \frac{-V_M * KR * Z_\delta}{g} & \frac{-V_M * KR * Z_\delta}{g} & \frac{-V_M * Z_\alpha}{g} \end{bmatrix} \begin{bmatrix} a_1 \\ q \\ \alpha \end{bmatrix} + [0] A_{ZC}
 \tag{6.35}$$

The symbol g represents the unit conversion from $\text{ft}/(\text{sec}^2)$ to G 's. The linear and nonlinear plots, with the inclusion of the three loop autopilot, are shown in figures 6.29 through 6.31. The nonlinear signals are those shown previously, in figures 6.19 through 6.21, without actuator dynamics.

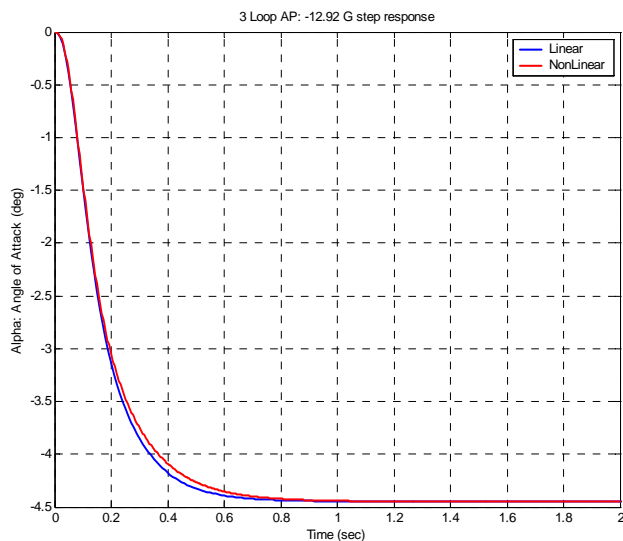


Figure 6.29. Closed Loop Linear and Nonlinear Angle of Attack

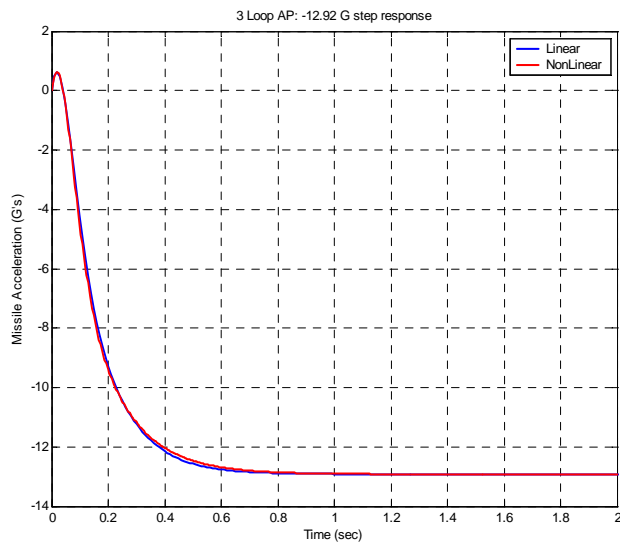


Figure 6.30. Closed Loop Linear and Nonlinear Missile Body Acceleration

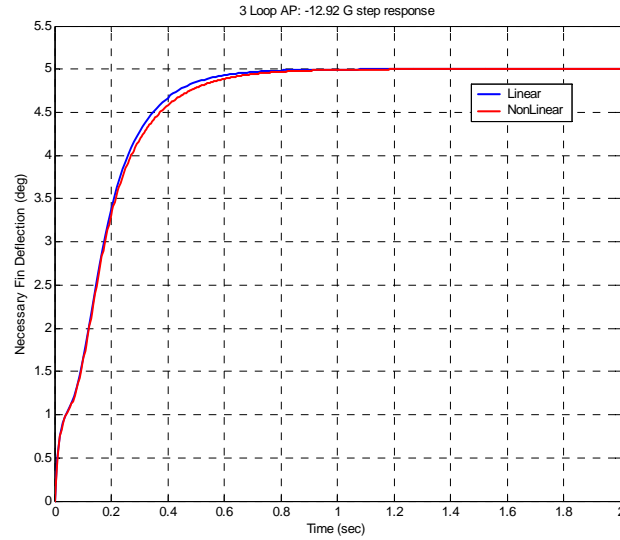


Figure 6.31. Closed Loop Linear and Nonlinear Necessary Fin Deflection

The closed loop transfer function of the state space equations is shown in equations 6.36 through 6.38.

$$G(S) = C * (SI - A)^{-1} B + D = \frac{Num}{Den} \quad (6.36)$$

$$Num = KDC * KR * WI * KA * \frac{Vm}{g} (-Z_{\delta} S^2 + M_{\alpha} Z_{\delta} - M_{\delta} Z_{\alpha}) \quad (6.37)$$

$$\begin{aligned} Den = S^3 - & \left(M_{\delta} * KR + Z_{\alpha} - WI * KA * KR * \frac{Vm}{g} * Z_{\delta} \right) S^2 + \\ & - (M_{\alpha} + M_{\delta} * KR * WI + Z_{\delta} M_{\alpha} * KR - Z_{\alpha} M_{\delta} * KR) S + \\ & + KR * WI * (Z_{\delta} M_{\alpha} - Z_{\alpha} M_{\delta}) * \left(1 + KA * \frac{Vm}{g} \right) \end{aligned} \quad (6.38)$$

Equation 6.37 shows that the system zeros are a function of flight condition and not influenced by the selection of autopilot gains. This of course is expected. The third order denominator, however, is dependent on the selection of four autopilot gains. This insight

can be exploited to place the closed loop poles as desired and maintain zero steady state error.

6.4.3 Linear Missile Pitch Dynamics: Sample Optimal Gains

This section focuses on selecting autopilot gains for the closed loop pitch plane missile. The previous section developed the closed loop transfer function symbolically for the linearized pitch plane system.

A third order denominator can be written in the general form of equation 6.39, for poles $p1$, $p2$, and $p3$.

$$\begin{aligned} Den &= (S - p1)(S - p2)(S - p3) = \\ &= S^3 - (p1 + p2 + p3)S^2 + (p1p2 + p1p3 + p2p3)S - p1p2p3 \end{aligned} \quad (6.39)$$

Equating the terms of equation 6.38 and 6.39, gives three equations to define the autopilot gains as a function of the system poles.

$$p1 + p2 + p3 = M_\delta * KR + Z_\alpha - WI * KA * KR * \frac{Vm}{g} * Z_\delta \quad (6.40)$$

$$p1p2 + p1p3 + p2p3 = -(M_\alpha + M_\delta * KR * WI + Z_\delta M_\alpha * KR - Z_\alpha M_\delta * KR) \quad (6.41)$$

$$-p1p2p3 = KR * WI * (Z_\delta M_\alpha - Z_\alpha M_\delta) * \left(1 + KA * \frac{Vm}{g}\right) \quad (6.42)$$

The four autopilot gains can be determined explicitly by adding an equation to the set, which forces zero steady state error. The zero steady state error equation is found by

forcing the last terms of equation 6.37 and 6.39, to be equal, thus creating four equations with four unknowns.

$$-p_1 p_2 p_3 = KDC * KR * WI * KA * \frac{Vm}{g} (M_\alpha Z_\delta - M_\delta Z_\alpha) \quad (6.43)$$

Equations 6.40 through 6.43 form an equation set with four equations and four unknowns, which can be solved for the four autopilot gains. Obviously, untangling this set to solve for the autopilot gains is quite computationally intensive. In order to solve equations 6.40 through 6.43, for the four autopilot gains, the *Maple* symbolic manipulator was employed [Map]. The results are shown in equations 6.44 through 6.47.

$$KA = -g (Md^2 p_1 p_2 p_3 - Md^2 Za^3 + p_1 Zd^2 Ma^2 + p_1 Md^2 Za^2 + p_2 Md^2 Za^2 - 2 p_3 Md Za Zd Ma - 2 p_2 Md Za Zd Ma + p_3 Md^2 Za^2 + p_3 Zd^2 Ma^2 - Za Zd^2 Ma^2 + p_2 Zd^2 Ma^2 - Md^2 Za Ma + Md Zd Ma^2 - 2 p_1 Md Za Zd Ma - Md^2 Za p_1 p_3 - Md^2 Za p_1 p_2 - Md^2 Za p_2 p_3 + Md Zd Ma p_2 p_3 + Md Zd Ma p_1 p_3 + Md Zd Ma p_1 p_2 + 2 Md Za^2 Zd Ma) / (Vm (p_1 Md^2 Za^2 - 2 p_1 Md Za Zd Ma + p_1 Zd^2 Ma^2 - 2 p_3 Md Za Zd Ma + p_3 Md^2 Za^2 + p_3 Zd^2 Ma^2 + 2 Md Za^2 Zd Ma - Md^2 Za p_1 p_2 - Za Zd^2 Ma^2 + Md Zd Ma^2 + Md Zd Ma p_1 p_3 - Md^2 Za p_2 p_3 - Md^2 Za Ma + Md Zd Ma p_2 p_3 + Md Zd Ma p_1 p_2 - Md^2 Za^3 - 2 p_2 Md Za Zd Ma + p_2 Md^2 Za^2 + p_2 Zd^2 Ma^2 - Md^2 Za p_1 p_3 + Zd^2 Ma p_1 p_2 p_3 - Zd Md Za p_1 p_2 p_3)) \quad (6.44)$$

$$KR = - (Zd^2 Ma^2 + Zd^2 Ma p_1 p_3 + Zd^2 Ma p_1 p_2 + Zd^2 Ma p_2 p_3 + Md Zd Ma p_2 - 2 Md Za Zd Ma + Md Zd Ma p_3 + Md Zd Ma p_1 - Zd Md Za p_1 p_3 - Zd Md Za p_2 p_3 + Zd p_1 p_2 p_3 Md - Zd Md Za p_1 p_2 - Md^2 Za p_3 - Md^2 Za p_1 + Md^2 Za^2 - Md^2 Za p_2) / ((-Md Za + Zd Ma) (-Md^2 + Zd^2 Ma - Zd Md Za)) \quad (6.45)$$

$$KDC = -p_1 p_2 p_3 (-Md^2 + Zd^2 Ma - Zd Md Za) / (Md^2 p_1 p_2 p_3 - Md^2 Za^3 + p_1 Zd^2 Ma^2 + p_1 Md^2 Za^2 + p_2 Md^2 Za^2 - 2 p_3 Md Za Zd Ma - 2 p_2 Md Za Zd Ma + p_3 Md^2 Za^2 + p_3 Zd^2 Ma^2 - Za Zd^2 Ma^2 + p_2 Zd^2 Ma^2 - Md^2 Za Ma + Md Zd Ma^2 - 2 p_1 Md Za Zd Ma - Md^2 Za p_1 p_3 - Md^2 Za p_1 p_2 - Md^2 Za p_2 p_3 + Md Zd Ma p_2 p_3 + Md Zd Ma p_1 p_3 + Md Zd Ma p_1 p_2 + 2 Md Za^2 Zd Ma) \quad (6.46)$$

$$WI = -((-Md Za + Zd Ma) (Zd Ma p_3 + Zd Ma p_2 + Zd Ma p_1 - Zd Za Ma + Zd p_1 p_2 p_3 + Md Ma - p_1 Md Za + Md p_1 p_2 + Md p_1 p_3 - Md Za p_3 + Md p_2 p_3 + Md Za^2 - p_2 Md Za)) / (Zd^2 Ma^2 + Zd^2 Ma p_1 p_3 + Zd^2 Ma p_1 p_2 + Zd^2 Ma p_2 p_3 + Md Zd Ma p_2 - 2 Md Za Zd Ma + Md Zd Ma p_3 + Md Zd Ma p_1 - Zd Md Za p_1 p_3 - Zd Md Za p_2 p_3 + Zd p_1 p_2 p_3 Md - Zd Md Za p_1 p_2 - Md^2 Za p_3 - Md^2 Za p_1 + Md^2 Za^2 - Md^2 Za p_2) \quad (6.47)$$

Although equations 6.44 through 6.47, are lengthy and complicated, they can be used in a *MATLAB* script to determine the necessary autopilot gains for a given set of pole locations. The optimization necessary to minimize the performance index of equation 6.10, while satisfying the constraint equations 6.11 through 6.13, becomes a matter of selecting pole locations for the system, and calculating the performance index, or rejecting the pole locations if the constraints are not met. For the flight condition described by figure 6.7, mach 3, altitude 0, the system zeros are at ± 39.0639376644535 . A non-minimum phase zero is a common occurrence for airframe dynamic systems [Zar02].

Variable	Set 1	Set 2	Set 3
<i>KA</i>	0.07836	0.10696	0.06493
<i>KR</i>	0.30587	0.23254	0.09800
<i>WI</i>	36.11504	24.68886	11.20000
<i>KDC</i>	1.13686	1.10027	1.16500
<i>PI</i>	0.06014	0.06105	0.11335
<i>Gain Marg.</i>	3.465 dB	3.000 dB	12.803 dB
<i>Phase Marg.</i>	180°	61.535°	180°
<i>Overshoot</i>	0%	0.92%	0%
<i>Undershoot</i>	30.00%	25.60%	4.90%
<i>Pole1</i>	51.51505 (-1 + i)	32.68964 (-1 + i)	-22.449 + 21.864i
<i>Pole2</i>	51.51505 (-1 - i)	32.68964 (-1 - i)	-22.449 - 21.864i
<i>Pole3</i>	-17.17168	-29.27869	-7.83147

Table 6.2. Optimal Gain Table

Table 6.2 shows three sets of gains with the performance index, and constraint values, using $\alpha_{\text{trim}} = -4.4489^\circ$. Gain set 1 has the lowest performance index and is limited by the undershoot constraint. Gain set 2 is limited by the gain margin constraint but has a slightly higher performance index. Gain set 3 is the original set of gains used to

introduce the three loop autopilot in Section 6.3.1. Gain set 3 has almost double the performance index value than the other two sets. Obviously, gain set 3 is nowhere near the optimum but has been included for reference. Figures 6.32 and 6.33, show the relatively slow rise time of gain set 3.

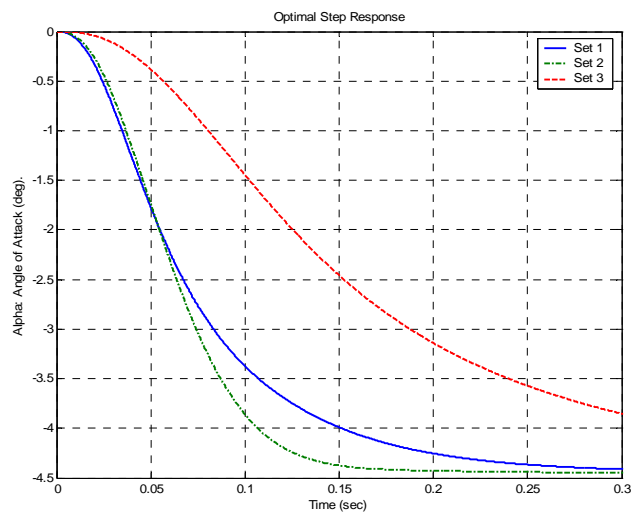


Figure 6.32. Optimal Gain Selection Angle of Attack

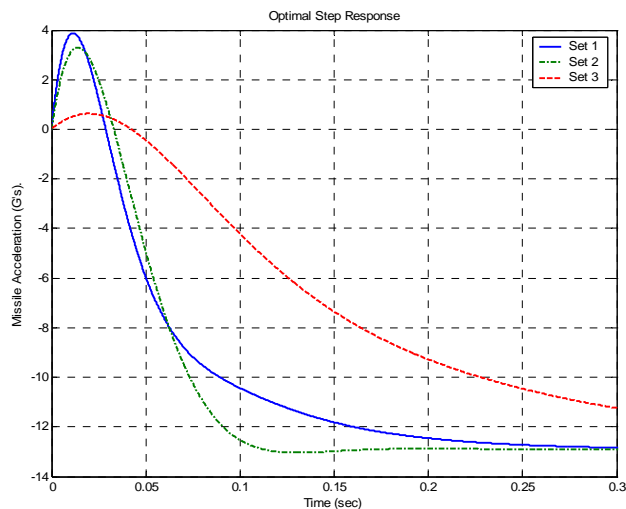


Figure 6.33. Optimal Gain Selection Missile Body Acceleration

Figures 6.32 through 6.34, show the step responses for gain sets 1 through 3. Figure 6.32 shows angle of attack, figure 6.33 shows missile acceleration, and figure 6.34 shows the necessary fin deflection to achieve the response.

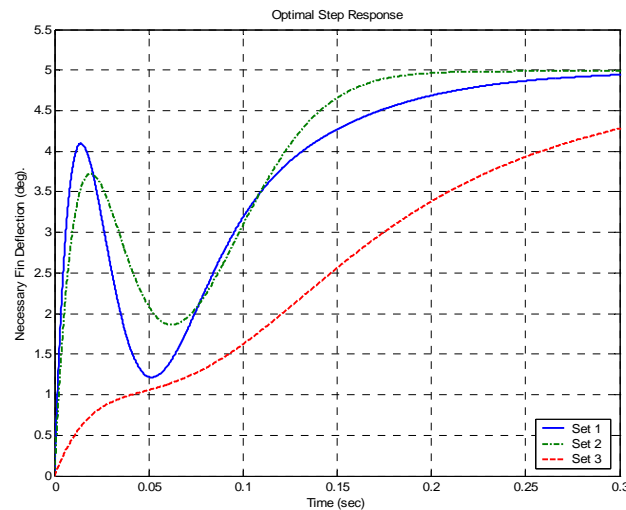


Figure 6.34. Optimal Gain Selection Necessary Fin Deflection

Naturally, the optimal gains will have zero steady state error, since this requirement was factored into the gain selection process. Figure 6.33 shows that gain set 1 achieves the desired acceleration much faster than set 3. Figure 6.34 shows that gain set 1 requires much more response from the actuator than gain set 3.

It is interesting to note that gain set 2 seems more optimal than gain set 1, when simply viewing the time response of figures 6.32 through 6.34. Gain set 2 does not have as much non-minimum phase response as gain set 1, gain set 2 has a faster settling time, which was not considered in the performance index, and gain set 2 does not require as

much initial fin dynamics as gain set 1. Regardless, gain set 1 is the optimal set according to the performance index criteria. Figures 6.35 and 6.36, graphically show the performance index calculation. Figure 6.35 shows the derivative of the performance index, and 6.36 is the performance index, with respect to time.

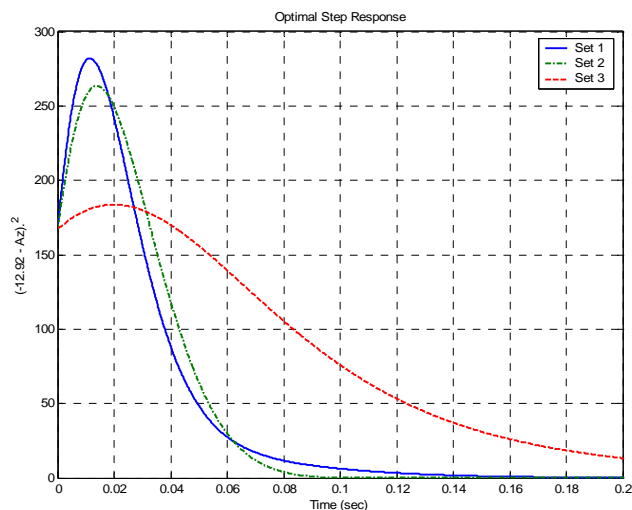


Figure 6.35. Optimal Gain Selection: $(\text{Command} - AZ)^2$

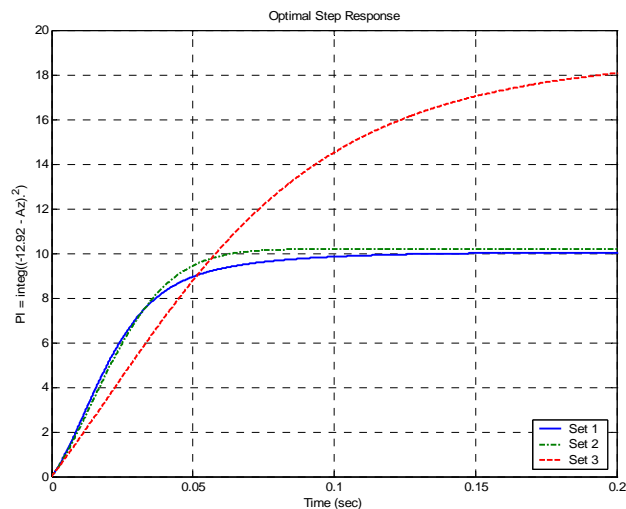


Figure 6.36. Optimal Gain Selection: Performance Index

Figures 6.35 and 6.36, show how the performance index of gain set 1 is lower than the performance index of gain set 2. During the first 0.025 seconds, gain set 2 has the lower performance index. However, the slightly wider peak of gain set 2, in figure 6.35, causes the performance index to increase beyond the performance index of gain set 1.

6.5 Actuator Power Flow Analysis Using Optimal Autopilot Gains

Naturally, the optimal gain set, developed in the last section, is intended to be used in the complete nonlinear system with actuator dynamics. The linear assumptions in the plant, and the idealized omission of actuator dynamics, are simplifications used to develop the autopilot gain set.

In this section the nonlinear plant/actuator is tested using the gain sets derived from the linear plant model. The nonlinear plant is used to determine the optimal gains. The optimal gains are then used to define autopilot efficiency, η_{AP} , using the efficiency calculation method of Chapter 5.

6.5.1 Actuator Power Flow Efficiency from the Optimal Gain Set

The three sets of gains, from the previous section, are used to control the complete nonlinear model with actuator dynamics, described in figure 6.18. The step responses for

these models are shown in figures 6.37 through 6.39. As before, the three plots correspond to angle of attack, missile acceleration, and fin deflection, respectively.

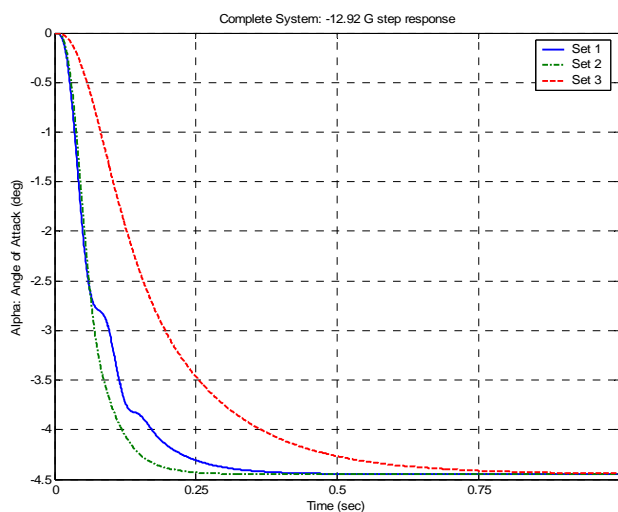


Figure 6.37. Nonlinear Missile with Optimal Gains: Angle of Attack

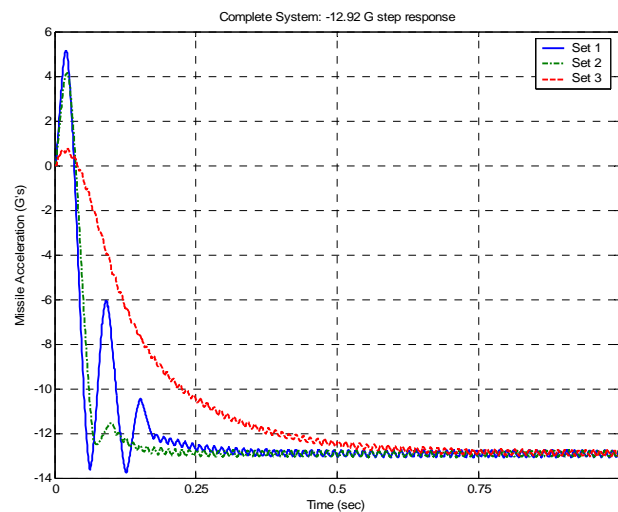


Figure 6.38. Nonlinear Missile with Optimal Gains: Body Acceleration

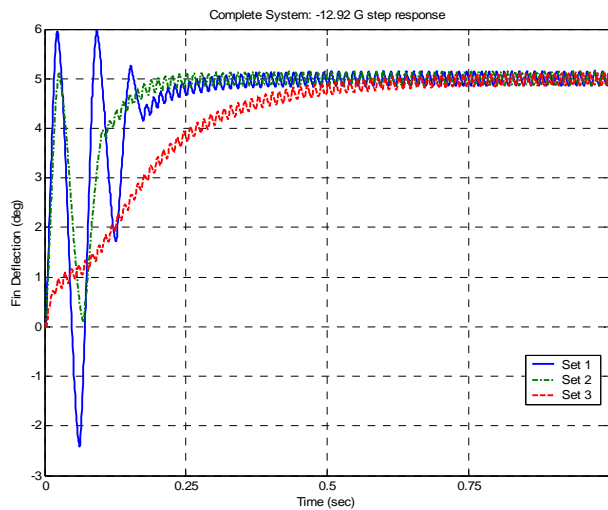


Figure 6.39. Nonlinear Missile with Optimal Gains: Fin Deflection

For clarity, figures 6.38 and 6.39 have been zoomed to show the first 0.25 seconds of activity. These signals are shown in figures 6.40 and 6.41 respectively.

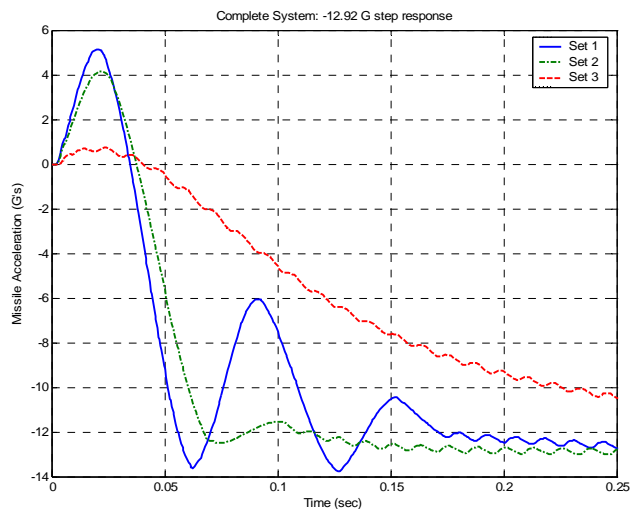


Figure 6.40. Nonlinear Missile with Optimal Gains: Body Acc. (zoom)

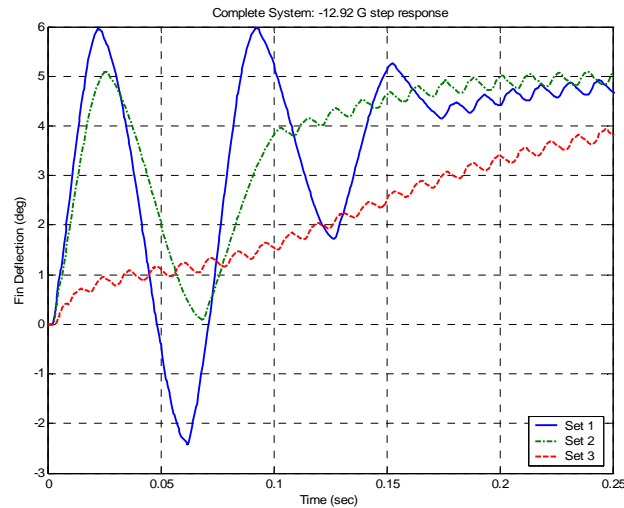


Figure 6.41. Nonlinear Missile with Optimal Gains: Fin Deflection (zoom)

By looking at figures 6.37 through 6.41, it is not apparent that gain set 1 is the optimal set of gains. This fact is masked by the nonlinear system dynamics. The optimal gains have pushed the overall system's response time closer to the actuator's response time. As in the linear case of the previous section, the response shown by gain set 2 looks much more optimal. The efficiency signal is shown in figure 6.42. The actuator power efficiency measurement reveals that gain set 1 provides the more optimal response.

It was shown in Chapter 5 that the desired shape of the efficiency signal is an initial steep rise and then an abrupt flattened signal. Figure 6.42 shows that gain set 1, clearly is more optimal than the other two sets. Gain sets 1 and 2 were very close from a performance index point of view, but are clearly different from an efficiency point of

view. Gain set 1 rises higher initially, and has a flatter slope after the sharp rise. Using 0.25 seconds and 1 second as reference points, the calculated slopes of the three signals are $4.8952e-007$, $7.9617e-007$, and $9.9538e-007$, for gain sets 1, 2, and 3, respectively. Gain set 1 has the smaller slope.

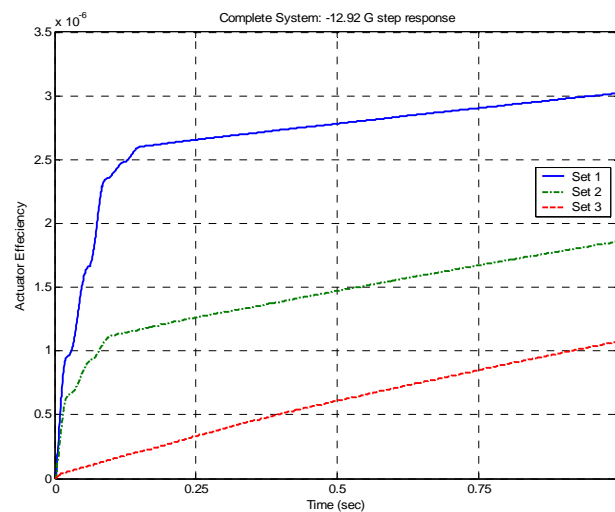


Figure 6.42. Autopilot η_{AP} : Autopilot Efficiency Signals for Gain Sets 1-3

Figure 6.42 clearly shows that gain set 1 is the more optimal, since it rises higher, initially, and has a flatter slope. Recall that gain set 1 was chosen because it reached the 30% undershoot constraint, identically, in the linear case. Gain set 2 was chosen because it reached the 3dB gain margin constraint, identically. Figure 6.40 shows that both gain sets 1 and 2, violate the 30% undershoot criteria in the nonlinear case, since the achieved acceleration for these sets peak up above 4 g's (30% of 12.92 g's is 3.876 g's). The nonlinearities of the pitch plane dynamics, and the nonlinear actuator dynamics, cause the

complete model to act differently than the linear, ideal system. Due to the performance difference between linear and nonlinear models, often, gains are optimized using the complete nonlinear model [Rei93]. The gain and phase margin criteria are calculated using the linear model, and the overshoot/undershoot criteria are calculated with the time response of the complete nonlinear model. This works well when the autopilot scheme is linear, otherwise gain and phase margins are meaningless.

In order to meet the 30% undershoot criteria, another gain set is added to the list of table 6.2. Gain set 4 is shown in table 6.3, along with gain sets 1 and 2 for reference. Gain set 4 is limited by the gain margin constraint. Also, the undershoot of the linear system is at 24.05%, which will place the nonlinear undershoot right at the 30% limit. Naturally, the performance index of gain set 4 is higher than that of gain sets 1 and 2, since the undershoot was further constrained.

Variable	Set 1	Set 2	Set 4
<i>KA</i>	0.07836	0.10696	0.11625
<i>KR</i>	0.30587	0.23254	0.21848
<i>WI</i>	36.11504	24.68886	21.86830
<i>KDC</i>	1.13686	1.10027	1.09226
<i>PI</i>	0.06014	0.06105	0.06224
<i>Gain Marg.</i>	3.465 dB	3.000 dB	3.000 dB
<i>Phase Marg.</i>	180°	61.535°	45.0055°
<i>Overshoot</i>	0%	0.92%	6%
<i>Undershoot</i>	30.00%	25.60%	24.05%
<i>Pole1</i>	51.51505 (-1 + i)	32.68964 (-1 + i)	-26.845 + 28.459i
<i>Pole2</i>	51.51505 (-1 - i)	32.68964 (-1 - i)	-26.845 - 28.459i
<i>Pole3</i>	-17.17168	-29.27869	-36.70808

Table 6.3. Added Gain Set

Figures 6.43 and 6.44 show the achieved acceleration for gain set 4. Figure 6.44 zooms in on the undershoot to show that gain set 4 achieves 30% undershoot, identically, for the nonlinear system. The amount of undershoot is shown explicitly in the achieved acceleration plot of figures 6.43 and 6.44.

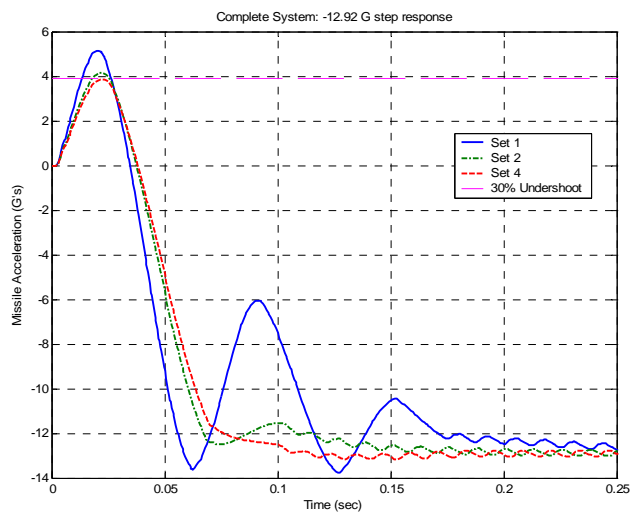


Figure 6.43. Optimal Gain Set 4: Body Acceleration

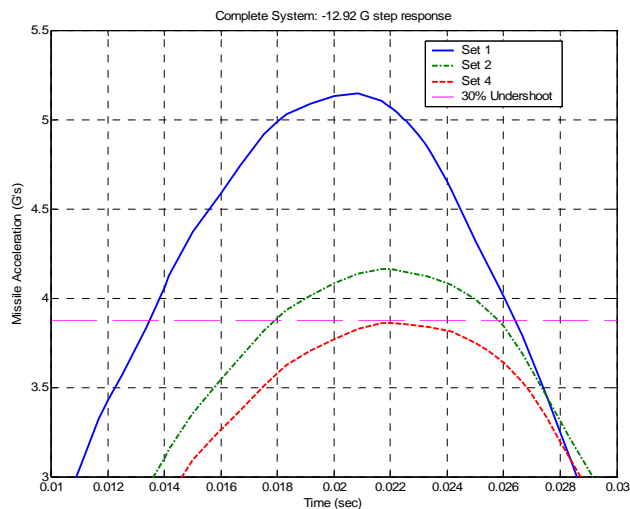


Figure 6.44. Optimal Gain Set 4: Body Acceleration (zoom)

Figure 6.45 shows the autopilot efficiency for gain set 4. Naturally, gain set 4 is not as efficient as sets 1 and 2. Gain set 4 is the optimal set of gains that meets both linear and nonlinear design constraints. Since gain set 4 meets all requirements for the constrained optimum, it is defined as the optimal gain set. The corresponding efficiency signal is then defined as the optimal autopilot efficiency, for the three loop autopilot. The optimal efficiency signal can then be used as a benchmark when comparing controllers of different designs.

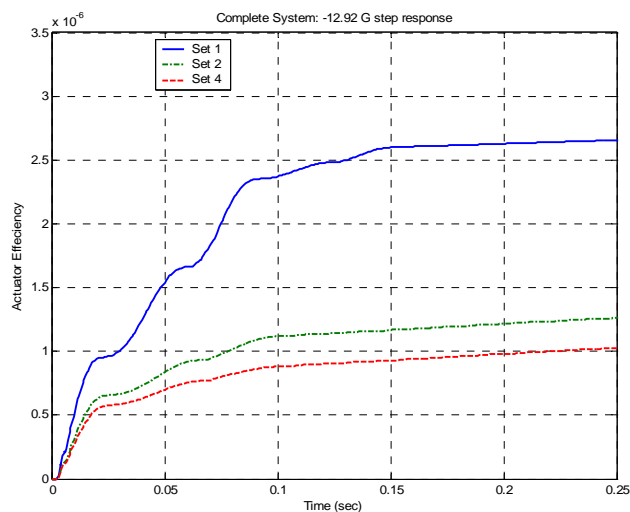


Figure 6.45. Optimal Gain Set 4: Autopilot Efficiency η_{AP}

6.5.2 Optimal Efficiency Comparisons

In the previous section, the optimal efficiency signal for a given controller design was defined as the actuator efficiency of the optimal gain set. This signal can be used to

compare controller efficiencies of varying architectures. In the design of a nonlinear controller, for a nonlinear plant, the performance constraints of gain and phase margin are of no use. However, a η_{AP} signal, defined with linear tools, can be used to warn the design engineer when the nonlinear system is beginning to approach areas of low stability margin. The efficiency of the actuator, for a linear autopilot, can be used as a limit when considering nonlinear designs. As seen in figure 6.45, controller designs with efficiency signals that show greater efficiency than the defined optimum, can be discarded with the assumption that the design violates the design constraints. Unfortunately, it is not true that if the efficiency of the controller design is less than that of the defined optimum, then a constraint has not been violated. It is necessary for a design to have an efficiency signal less than or equal to, the optimal efficiency in order to be a potential candidate. However, this condition is not sufficient to guarantee that all constraints have not been violated. In order to illustrate this point, two more gain sets were added to the list in table 6.3. These two gain sets are described in table 6.4.

Variable	Set 4	Set 5	Set 6
<i>KA</i>	0.11625	0.11601	0.12330
<i>KR</i>	0.21848	0.21630	0.18061
<i>WI</i>	21.86830	21.68998	21.07446
<i>KDC</i>	1.09226	1.09245	1.08698
<i>PI</i>	0.06224	0.06243	0.06355
<i>Gain Marg.</i>	3.000 dB	3.089 dB	1.949927 dB
<i>Phase Marg.</i>	45.0055°	46.3686°	24.425°
<i>Overshoot</i>	6%	6%	9%
<i>Undershoot</i>	24.05%	23.70%	22.92%
<i>Pole1</i>	-26.845 + 28.459i	-26.577 + 28.174i	-22.429 + 32.989i
<i>Pole2</i>	-26.845 - 28.459i	-26.577 - 28.174i	-22.429 - 32.989i
<i>Pole3</i>	-36.70808	-36.70808	-29.68812

Table 6.4. Suboptimal Gain Sets

From the previous section, gain set 4 is the gain set that provides the defined optimal efficiency. Gain set 4 is limited by the gain margin constraint, and undershoot constraint using the nonlinear actuator/plant. Gain set 5 has been chosen intentionally to meet all constraints but with a slightly larger performance index than the defined optimum. Gain sets that have a lower performance index, but violate design constraints, were considered in the previous section, and shown in figure 6.45. Thus, gain set 6 has been chosen intentionally to violate the gain margin constraint, but have a higher performance index than gain set 4, and therefore are not as optimal. Neither gain sets 5, nor 6, will violate the undershoot constraint for the nonlinear system since they have a linear undershoot of no more than 23.7%.

The step responses, shown in figure 6.46, look very similar for each of the three sets of gains.

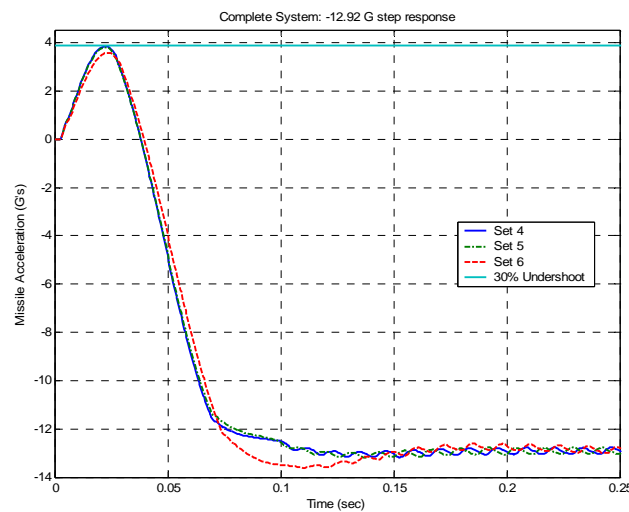


Figure 6.46. Body Acceleration: Gain Sets 4-6

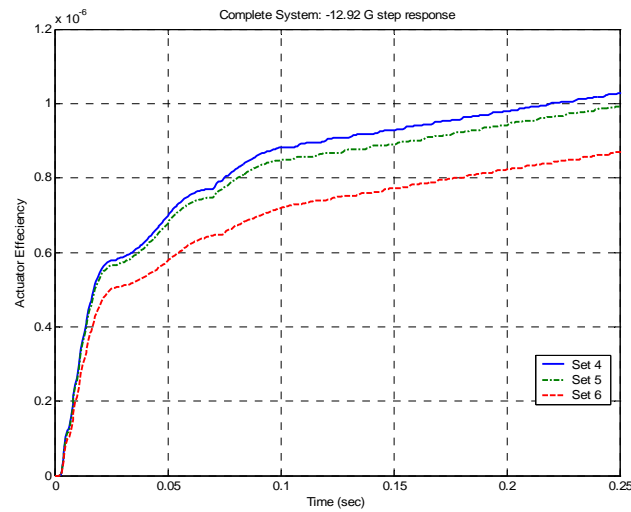


Figure 6.47. Autopilot η_{AP} Gain Sets 4-6

Figure 6.47 shows the efficiency signals, and that the efficiencies of these three gain sets are easily compared. Gain set 5 does not violate any constraints and shows a lower efficiency signal. Gain set 6 violates the constraints yet is not as efficient as gain set 4, thus showing that a lower efficiency signal does not guarantee that the design (gain sets in this case) meets all constraints.

Gain sets 4 and 5, meet all linear and nonlinear criteria. However, if the efficiencies of gain sets 5 and 6, had been created with a nonlinear design, then the gain margin information for these sets would not have been known.

Thus far, all performance index calculations have been done for the linear plant. Figure 6.48 shows the performance index, as a function of time, calculated using the achieved acceleration of the nonlinear plant. Figure 6.48 shows these signals for gain sets 4 through 6, normalized to a unit step.

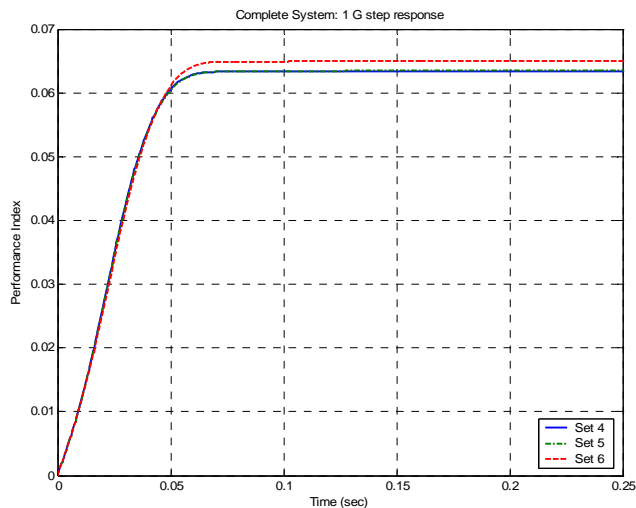


Figure 6.48 Nonlinear PI for Gain Sets 4-6

As expected, gain sets 4 and 5 are nearly indistinguishable, since the linear performance indices were so close. Figure 6.49 zooms in on the details of figure 6.48 in the 0.05 second region.

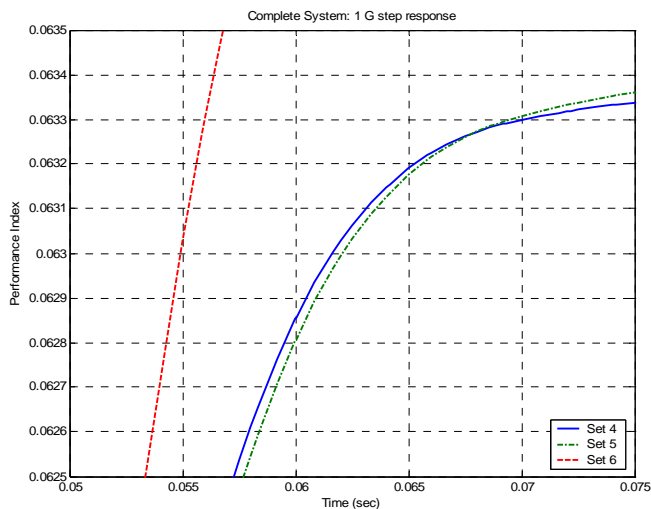


Figure 6.49. Nonlinear PI for Gain Sets 4-6 (zoom)

It is interesting to note, that gain set 5 shows a smaller performance index than gain set 4 up until 0.0678 seconds. Figure 6.47 shows that the efficiency signals clearly distinguishes gain set 4 as the more optimal set, even prior to 0.0678 seconds, even though gain set 5 has a smaller performance index during the initial time period. Figure 6.50 zooms in on figure 6.47 using the same time scale as figure 6.49.

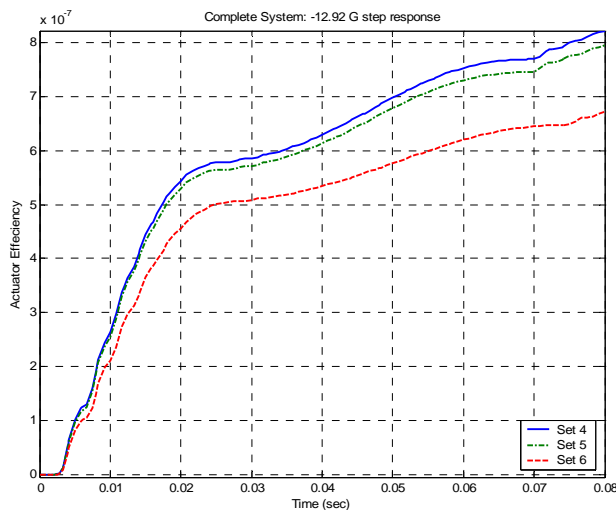


Figure 6.50. Autopilot Efficiency η_{AP} : Gain Sets 4-6 (zoom)

It has been shown here, that the autopilot efficiency signal, η_{AP} , can be used to measure the efficiency of an autopilot design, and how these efficiencies can be compared to a predetermined efficiency signal. The predetermined efficiency signal was calculated by finding a set of gains that satisfies the frequency domain constraints in the linear case, and the time domain constraints in the nonlinear case. This signal, generated by gain set 4, is then defined as the optimum. This efficiency signal is then used as a

boundary of efficiency, that can be used to benchmark efficiency signals generated by any other autopilot architecture. If the efficiency is greater than the optimum, the design can be discarded under the assumption that it violates a constraint. If the efficiency is less than the optimum, then no conclusion can be made.

6.6 Nonlinear Pitch Autopilot: An SDRE Approach

This section introduces a nonlinear autopilot design that is based on solving the state dependent Riccati equation (SDRE) at each time step, to determine feedback gains [Clo96a]. The basic approach behind the design follows the standard LQR problem.

6.6.1 LQR Formulation and General Solution

The standard linear quadratic regulator (LQR) problem is described by equations 6.48 through 6.53 [Mra05, Kir98]. The performance index to be minimized is

$$\min_u J = \int_0^{\infty} (z^T Q z + u^T R u) dt \quad (6.48)$$

Subject to the dynamics

$$\dot{x} = Ax + Bu \quad (6.49)$$

$$y = Hx \quad (6.50)$$

Q is a positive semi-definite matrix and R is a positive definite matrix. The optimal state feedback is

$$u = Kx \quad (6.51)$$

where

$$K = -R^{-1}B^T P \quad (6.52)$$

and P is the stabilizing solution to the algebraic Riccati equation

$$0 = A^T P + PA - PBR^{-1}B^T P + H^T QH \quad (6.53)$$

Rearranging equations 6.32 and 6.33, the missile pitch dynamics can be written as

$$\begin{bmatrix} \dot{\alpha} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} Z_\alpha & 1 \\ M_\alpha & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} Z_\delta \\ M_\delta \end{bmatrix} \delta \quad (6.54)$$

$$A_Z = \begin{bmatrix} -Z_\alpha V_M & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} -Z_\delta V_M \end{bmatrix} \delta \quad (6.55)$$

Unlike equations 6.32 and 6.33, equations 6.53 and 6.55 allow Z_α and M_α to be functions of α , as described by equations 6.15, 6.22, 6.26, and 6.30. Thus, trim condition is not used for this analysis.

6.6.2 LQR General Solution for Nonzero Feed-Through

Note, that equations 6.54 and 6.55 are in the form of equations 6.49 and 6.50, with the exception of the nonzero feed-through term of equation 6.55. Mracek and Ridgely show how to handle the linear quadratic optimal control problem for a system containing a nonzero feed-through in the plant equations [Mra05]. The analysis is as follows:

$$\min_u J = \int_0^\infty (z^T \tilde{Q}z + u^T \tilde{R}u) dt \quad (6.56)$$

Subject to the dynamics

$$\dot{x} = Ax + Bu \quad (6.57)$$

$$z = Hx + Lu \quad (6.58)$$

Substituting equation 6.58 into 6.56 yields

$$\min_u J = \int_0^{\infty} [(Hx + Lu)^T \tilde{Q}(Hx + Lu) + u^T \tilde{R}u] dt \quad (6.59)$$

Let

$$Q = H^T \tilde{Q}H \quad (6.60)$$

$$S = H^T \tilde{Q}L \quad (6.61)$$

$$R = \tilde{R} + L^T \tilde{Q}L \quad (6.62)$$

The resulting performance index is

$$\min_u J = \int_0^{\infty} [x^T Qx + x^T Su + u^T S^T x + u^T Ru] dt \quad (6.63)$$

The Hamiltonian is then

$$H = x^T Qx + x^T Su + u^T S^T x + u^T Ru + \left(\frac{\partial J^*}{\partial x} \right)^T [Ax + Bu] \quad (6.64)$$

Taking the partial of H with respect to u and setting to 0 yields

$$\frac{\partial H}{\partial u} = 2S^T x + 2Ru + B^T \left(\frac{\partial J^*}{\partial x} \right) = 0 \quad (6.65)$$

Solving equation 6.65 for u

$$u^* = -R^{-1} \left[\frac{1}{2} B^T \left(\frac{\partial J^*}{\partial x} \right) + S^T x \right] \quad (6.66)$$

The Hamilton-Jacobi equation is

$$\begin{aligned}
-\frac{\partial J^*}{\partial t} = H^* &= x^T(Q - SR^{-1}S^T)x + \left(\frac{\partial J^*}{\partial x}\right)^T Ax + \\
&-\frac{1}{4}\left(\frac{\partial J^*}{\partial x}\right)^T BR^{-1}B^T\left(\frac{\partial J^*}{\partial x}\right) - x^T SR^{-1}B^T\left(\frac{\partial J^*}{\partial x}\right)
\end{aligned} \tag{6.67}$$

Assuming

$$J^* = x^T Px \tag{6.68}$$

$$\frac{\partial J^*}{\partial t} = x^T \dot{P}x \tag{6.69}$$

$$\frac{\partial J^*}{\partial x} = 2Px \tag{6.70}$$

Substituting equations 6.69 and 6.70, into the Hamilton-Jacobi equation, and rearranging terms yields

$$-x^T \dot{P}x = x^T \left[(A - BR^{-1}S^T)^T P + P(A - BR^{-1}S^T) - PBR^{-1}B^T P + (Q - SR^{-1}S^T) \right] x \tag{6.71}$$

For $\dot{P} \rightarrow 0$ the algebraic Riccati equation is

$$0 = (A - BR^{-1}S^T)^T P + P(A - BR^{-1}S^T) - PBR^{-1}B^T P + (Q - SR^{-1}S^T) \tag{6.72}$$

P is the stabilizing solution to equation 6.72. Using full state feedback, the optimal control is given by

$$u^* = -R^{-1}(B^T P + S^T)x = Kx \tag{6.73}$$

6.6.3 LQR Solution for Nonzero Feed-Through and Output Feedback

Equation 6.73 assumes full state feedback. In the case of the missile system, output feedback is desired. Mracek and Ridgely define *full state observability* as the requirement that C^{-1} and $[I+K C^{-1}D]^{-1}$ exist [Mra05]. The three-loop autopilot uses missile achieved acceleration A_z , and missile pitch rate q , signals as feedback to the autopilot. Keeping this same standard, the dynamic equations have the form

$$\dot{x} = Ax + Bu = \begin{bmatrix} \dot{\alpha} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} Z_\alpha & 1 \\ M_\alpha & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} Z_\delta \\ M_\delta \end{bmatrix} \delta \quad (6.74)$$

$$z = Hx + Lu = A_z = \begin{bmatrix} -Z_\alpha V_M & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} -Z_\delta V_M \end{bmatrix} \delta \quad (6.75)$$

$$y = Cx + Du = \begin{bmatrix} A_{zerror} \\ q \end{bmatrix} = \begin{bmatrix} -Z_\alpha V_M & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} -Z_\delta V_M \\ 0 \end{bmatrix} \delta \quad (6.76)$$

The following argument eliminates the dependence of full state feedback from equation 6.73

$$u^* = Kx \quad (6.77)$$

$$y = Cx + Du^* \quad (6.78)$$

Solving equation 6.78 for x yields

$$x = C^{-1}[y - Du^*] \quad (6.79)$$

Pre-multiplying both sides of equation 6.79 by K

$$Kx = KC^{-1}[y - Du^*] \quad (6.80)$$

Equating 6.77 and 6.80 yields

$$u^* = KC^{-1}[y - Du^*] \quad (6.81)$$

Solving equation 6.81 for u^*

$$u^* = [I + KC^{-1}D]^{-1} KC^{-1}y \quad (6.82)$$

Equation 6.82 shows the optimal control calculation as a function of output feedback.

Using equation 6.73 for K , 6.82 becomes

$$u^* = -[I - R^{-1}(B^T P + S^T)C^{-1}D]^{-1} R^{-1}(B^T P + S^T)C^{-1}y = K_{opt}y \quad (6.83)$$

Therefore, K_{opt} is

$$K_{opt} = -[I - R^{-1}(B^T P + S^T)C^{-1}D]^{-1} R^{-1}(B^T P + S^T)C^{-1} \quad (6.84)$$

Equation 6.84 shows the optimal gain set for output feedback, and a nonzero feed-through term in the plant equations.

6.6.4 LQR Tracking Solution for Nonzero Feed-Through, Output Feedback and Zero Steady State Error

The steady state gain must be defined before the optimal control autopilot can be implemented [Mra05]. The output feedback y , was defined in equation 6.76 as A_{Zerror} and q , which can be written

$$y = \begin{bmatrix} A_{Zerror} \\ q \end{bmatrix} = \begin{bmatrix} A_{Zm} - K_{ss}A_{Zc} \\ q \end{bmatrix} = \begin{bmatrix} Hx + Lu - K_{ss}A_{Zc} \\ q \end{bmatrix} = Cx + Du - \begin{bmatrix} K_{ss}A_{Zc} \\ 0 \end{bmatrix} \quad (6.85)$$

Substituting 6.85 into 6.83 gives

$$u = K_{opt} [Cx + Du] - K_{opt} \begin{bmatrix} K_{ss} A_{zc} \\ 0 \end{bmatrix} \quad (6.85)$$

Thus, the optimal control is

$$u = [I - K_{opt} D]^{-1} K_{opt} \left[Cx - \begin{bmatrix} K_{ss} A_{zc} \\ 0 \end{bmatrix} \right] \quad (6.86)$$

Using the control of equation 6.86, in the dynamic equations 6.74 and 6.75, closed loop state space matrices can be found.

$$\dot{x} = A_c x + B_c A_{zc} \quad (6.87)$$

$$A_{zm} = C_c x + D_c A_{zc} \quad (6.88)$$

Where the state vector is the same as before and the closed loop state space matrices are

$$A_c = A + B[I - K_{opt} D]^{-1} K_{opt} C \quad (6.89)$$

$$B_c = -B[I - K_{opt} D]^{-1} K_{opt} \begin{bmatrix} K_{ss} \\ 0 \end{bmatrix} \quad (6.90)$$

$$C_c = H + L[I - K_{opt} D]^{-1} K_{opt} C \quad (6.91)$$

$$D_c = -L[I - K_{opt} D]^{-1} K_{opt} \begin{bmatrix} K_{ss} \\ 0 \end{bmatrix} \quad (6.92)$$

Equations 6.90, and 6.92, can be written such that K_{ss} is not part of the state space matrices.

$$B_c = -B'_c K_{ss} = -B[I - K_{opt} D]^{-1} K_{opt} \begin{bmatrix} K_{ss} \\ 0 \end{bmatrix} \quad (6.93)$$

$$D_c = -D'_c K_{ss} = -L[I - K_{opt} D]^{-1} K_{opt} \begin{bmatrix} K_{ss} \\ 0 \end{bmatrix} \quad (6.94)$$

Where

$$B'_c = B[I - K_{opt}D]^{-1}K_{opt} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad (6.95)$$

$$D'_c = L[I - K_{opt}D]^{-1}K_{opt} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad (6.96)$$

For zero steady state error, equation 6.97 must hold.

$$\lim_{s \rightarrow 0} [C_c (SI - A_c)^{-1} B_c + D_c] = 1 \quad (6.97)$$

Therefore

$$-C_c A_c^{-1} B_c + D_c = [C_c A_c^{-1} B'_c - D'_c] K_{ss} = 1 \quad (6.98)$$

Solving equation 6.98 for K_{ss}

$$K_{ss} = [C_c A_c^{-1} B'_c - D'_c]^{-1} \quad (6.99)$$

The optimal control autopilot can now be implemented in Dymola.

6.6.5 Dymola Implementation of the LQR Tracking Solution

The Dymola model used to implement the nonlinear optimal feedback has many layers. The discussion here uses a top down approach.

6.6.5.1 SDRE Autopilot

The top hierarchical level is shown in figure 6.51, depicting both the icon and diagram layer. The icon layer shows missile angle of attack α , missile pitch rate q , and acceleration error as inputs. The outputs are the optimal control and the steady state gain. The diagram window shows that the autopilot calls a Riccati equation solver. No connections are shown to the inputs of the *Riccati4* block. These connections are made in the equation layer of the model. The equation window contains the code that calls *Riccati4*, calculates the optimal gains, and calculates the steady state gain. A complete code listing can be found in Appendix A1.

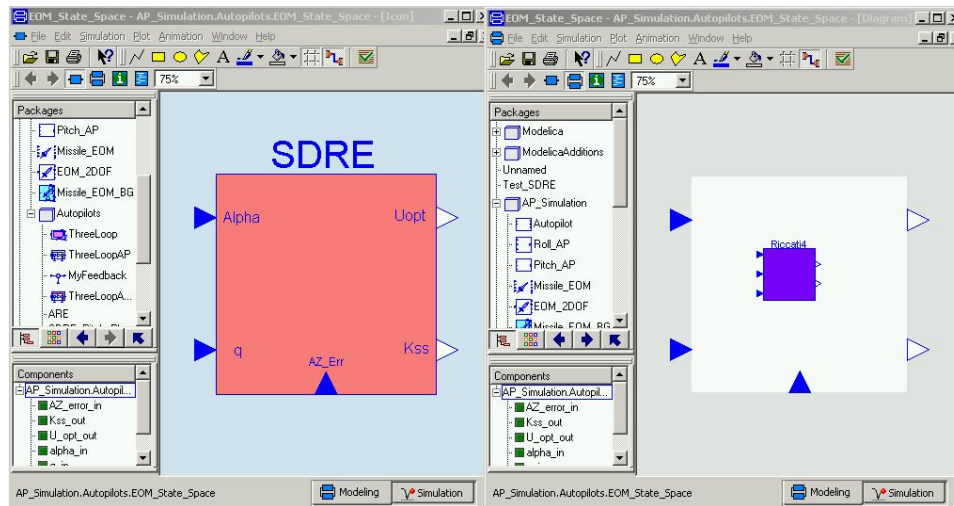


Figure 6.51. SDRE Autopilot: Icon and Diagram Window

For a given value of α , three matrices Ah , Bh , and Ch are defined such that

$$Ah = (A - BR^{-1}S^T) \quad (6.100)$$

$$Bh = BR^{-1}B^T \quad (6.101)$$

$$Ch = (Q - SR^{-1}S^T) \quad (6.102)$$

These three matrices are sent to the Riccati equation solver. These three matrices come from equation 6.72. The solution of the Riccati equation is used to determine the optimal control, and the steady state gain, as described in the previous sections.

6.6.5.2 Algebraic Riccati Equation Solver *Riccati4*

Unfortunately, Dymola does not have a linear algebra library. Thus, the code to solve the Riccati equation was done from the ground up. Figure 6.52 shows the diagram window of the *Riccati4* solver.

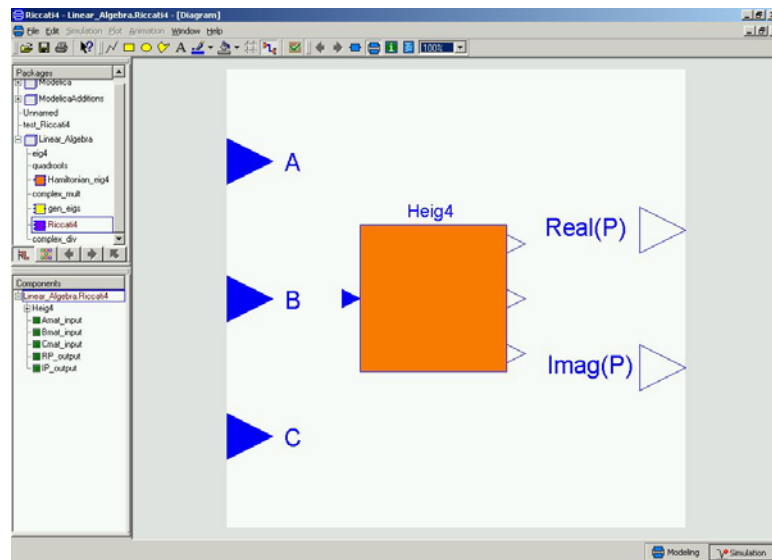


Figure 6.52. Algebraic Riccati Equation Solver *Riccati4*: Diagram Window

The *Riccati4* solver is set up to receive three, separate, 2x2 matrices A , B , C and find the stabilizing solution P , such that

$$0 = A^T P + PA - PBP + C \quad (6.103)$$

[Zho96 pp. 328-333]. This algorithm was not set up for the general nxn case since the intent here is to stabilize the 2nd order pitch dynamics model. Appendix A2 contains a code listing for the equation window of figure 6.52.

The Hamiltonian matrix is formed such that

$$H = \begin{bmatrix} A & -B \\ -C & -A^T \end{bmatrix} \quad (6.104)$$

The 4x4 Hamiltonian matrix is sent to an eigenvalue/eigenvector solver *Heig4*. *Heig4* passes back the four eigenvalues, and four eigenvectors of H . The eigenvalues of the Hamiltonian matrix are symmetric about both axes in the complex plain. A proof of this is found in Appendix B1. Thus, for the 4x4 Hamiltonian, two of the eigenvalues have negative real parts. The eigenvectors, V_n and V_m , associated with the stable eigenvalues, λ_n and λ_m , are used to form the two 2x2 matrices X_1 and X_2 .

$$X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} V_{n1} & V_{m1} \\ V_{n2} & V_{m2} \\ V_{n3} & V_{m3} \\ V_{n4} & V_{m4} \end{bmatrix} \quad (6.105)$$

The stabilizing solution of the algebraic Riccati equation [Zho96 pp. 333-341] is then

$$P = X_2 X_1^{-1} \quad (6.106)$$

Dymola does not handle complex numbers directly, so the eigenvalues are passed back from *Heig4* with two variables for each eigenvalue, representing the real and imaginary parts. The same is true for the elements of the eigenvectors. Thus, *Riccati4* passes out a

real matrix, and an imaginary matrix, of the solution P . The imaginary matrix associated with P should always be zero, but is passed out for debugging purposes.

6.6.5.3 Hamiltonian Eigenvalue Solver *Heig4*

Figure 6.53 shows the diagram window of *Heig4*. This routine receives the Hamiltonian 4×4 matrix and passes out a 4×4 matrix, for the real part of the eigenvectors, a 4×4 matrix for the imaginary part of the eigenvectors, and a 1×8 vector containing the four real parts of the eigenvalues, and four imaginary parts of the eigenvalues. The Hamiltonian matrix is passed to a generalized eigenvector solver, *gen_eigs*, internal to this routine.

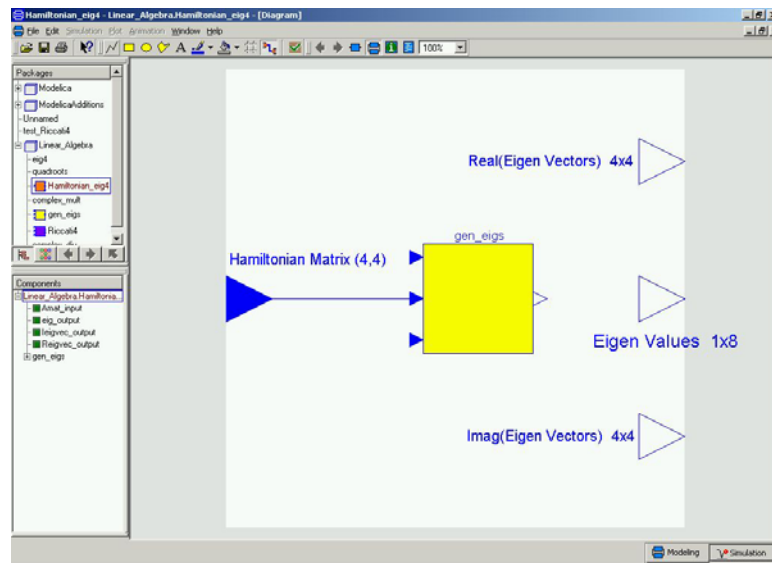


Figure 6.53. Hamiltonian Eigenvalue Solver *Heig4*: Diagram Window

A complete code listing of the equation window can be found in Appendix A3. To find the eigenvalues of the Hamiltonian matrix, the characteristic polynomial was found by converting the matrix to controller canonical form [Kai80 pp. 50-51]. A B vector is defined as

$$B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad (6.107)$$

The controllability matrix is then

$$C_{con} = [B \quad HB \quad H^2B \quad H^3B] \quad (6.108)$$

C_{con} is then inverted. lr defined as the last row of $[C_{con}]^{-1}$. The transformation matrix that converts the Hamiltonian into controller canonical form is then

$$T = \begin{bmatrix} lr \\ lrH \\ lrH^2 \\ lrH^3 \end{bmatrix} \quad (6.109)$$

Thus,

$$\tilde{H} = THT^{-1} \quad (6.110)$$

is in controller canonical form. The characteristic polynomial is then

$$CharPoly = S^4 - \tilde{H}(4,4)S^3 - \tilde{H}(4,3)S^2 - \tilde{H}(4,2)S - \tilde{H}(4,1) \quad (6.111)$$

Appendix B1 shows that the eigenvalues of the Hamiltonian matrix are symmetric about both axes in the complex plain. This symmetry property forces all odd powers of S , of

the characteristic polynomial, to have a coefficient of zero. Thus, both $\tilde{H}(4,4)$ and $\tilde{H}(4,2)$ are zero. The characteristic polynomial can be written

$$CharPoly = S^4 - \tilde{H}(4,3)S^2 - \tilde{H}(4,1) \quad (6.112)$$

Also, for a controllable system, H does not have any eigenvalues on the imaginary axis [Zho96]. The symmetry property forces $-\tilde{H}(4,1)$ to be positive. Thus, the 4th order characteristic polynomial can be reduced to two 2nd order polynomials, with real coefficients

$$CharPoly = S^4 - \tilde{H}(4,3)S^2 - \tilde{H}(4,1) = (S^2 + a1S + a2)(S^2 + b1S + b2) \quad (6.113)$$

such that

$$a2 = b2 = \sqrt{-\tilde{H}(4,1)} \quad (6.114)$$

$$b1 = \sqrt{2 * b2 + \tilde{H}(4,3)} \quad (6.115)$$

and

$$a1 = -b1 \quad (6.116)$$

The values of $a1$, $a2$, $b1$ and $b2$ are all real. Thus, the eigenvalues of H can be found by using the quadratic equation on each of the 2nd order polynomials. A Dymola function was written to find the roots of a 2nd order polynomial using the quadratic equation. Two values are passed back for each root, one for the real part of the root, and one for the imaginary part. For completeness, a listing of the quadratic equation function is found in Appendix A5.

The eigenvectors are found using a Vandermonde matrix. Appendix B2 shows that, for a matrix \tilde{A} in controller canonical form, the eigenvectors are

$$\tilde{V} = \begin{bmatrix} \lambda_1^0 & \lambda_2^0 & \cdot & \cdot & \cdot & \lambda_n^0 \\ \lambda_1^1 & \lambda_2^1 & \cdot & \cdot & \cdot & \lambda_n^1 \\ \cdot & \cdot & \cdot & & & \cdot \\ \cdot & \cdot & & \cdot & & \cdot \\ \cdot & \cdot & & & \cdot & \cdot \\ \lambda_1^{n-1} & \lambda_2^{n-1} & \cdot & \cdot & \cdot & \lambda_n^{n-1} \end{bmatrix} \quad (6.117)$$

where λ_j is the j^{th} eigenvalue. Thus,

$$\tilde{H} = THT^{-1} = \tilde{V}\Lambda\tilde{V}^{-1} \Rightarrow H = T^{-1}\tilde{V}\Lambda\tilde{V}^{-1}T \quad (6.118)$$

Therefore, the eigenvector matrix, V , corresponding to the Hamiltonian matrix can be written

$$V = T^{-1}\tilde{V} \quad (6.119)$$

It is important to note that the possibility of finding a repeated eigenvalue for the Hamiltonian is limited to two, i.e., there can be no more than two repeated roots for the 4th order Hamiltonian. Repeated roots occur when the roots sit on the real axis such that there are two at $+\lambda$, and two roots at $-\lambda$. Therefore a generalized eigenvector routine, *gen_eigs*, is employed to check for a repeated root situation. A complete code listing for *gen_eigs* is found in Appendix A4.

6.6.6 Nonlinear Autopilot Results

6.6.6.1 Ideal Actuator

Figure 6.54 shows the pitch plane bond graph dynamics controlled by the nonlinear autopilot, without actuator dynamics. The acceleration error is input to the SDRE autopilot, along with missile states α and q . SDRE calculates the value of K_{ss} and U_{opt} , as described by the previous section.

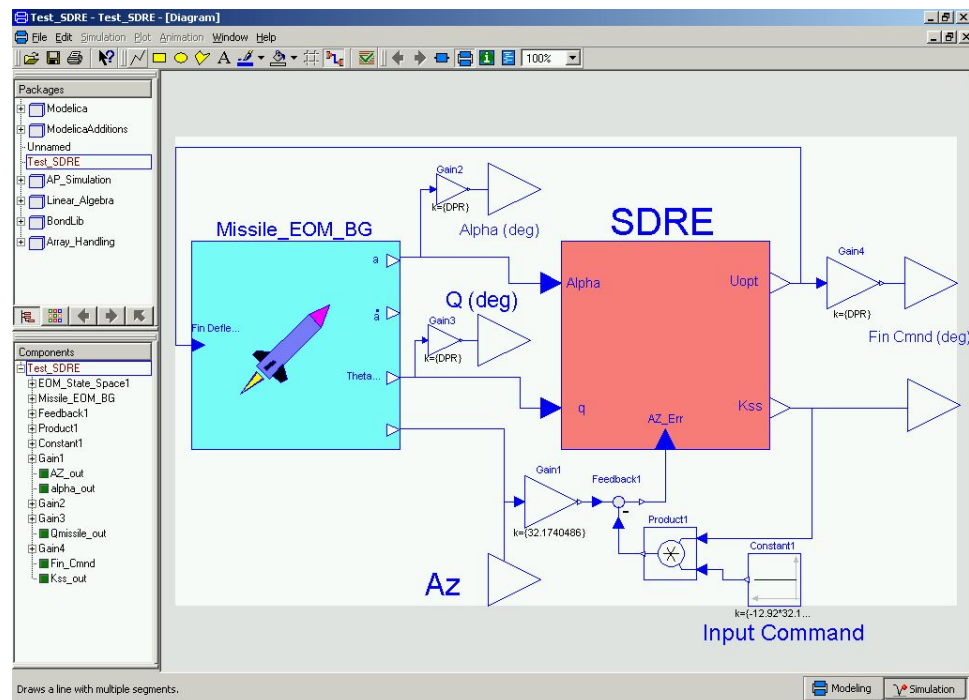


Figure 6.54. Pitch Plane Dynamics with SDRE Autopilot

As before, the input command is a step of -12.92 G's.

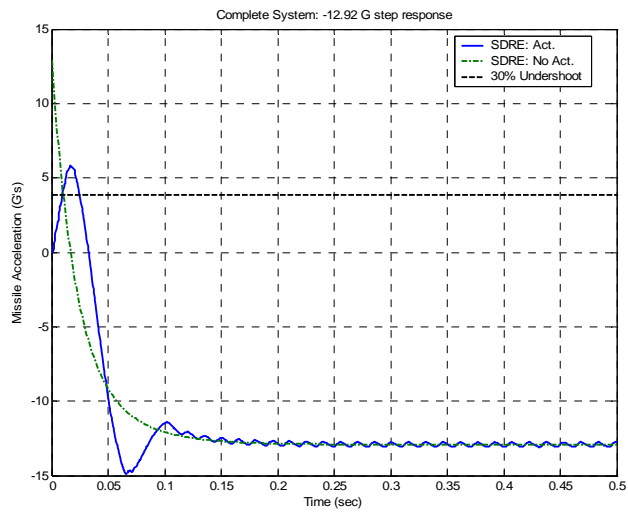


Figure 6.56. SDRE: Achieved Acceleration

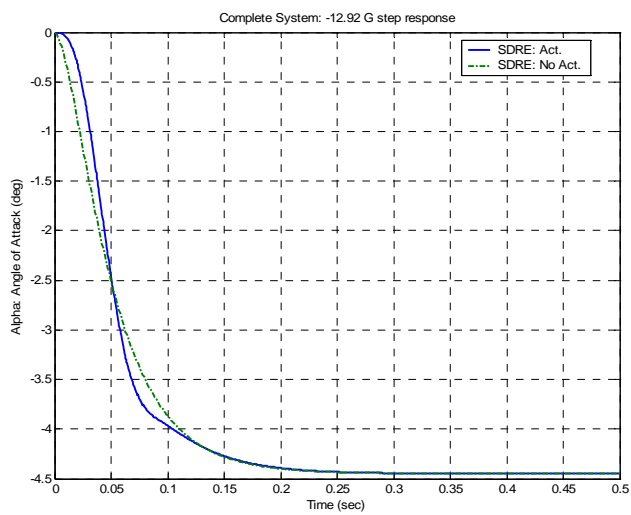


Figure 6.57. SDRE: Angle of Attack

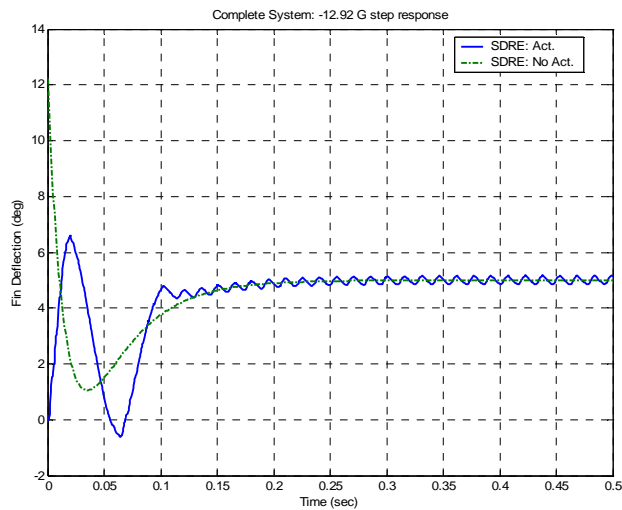


Figure 6.58. SDRE: Achieved Fin Deflection

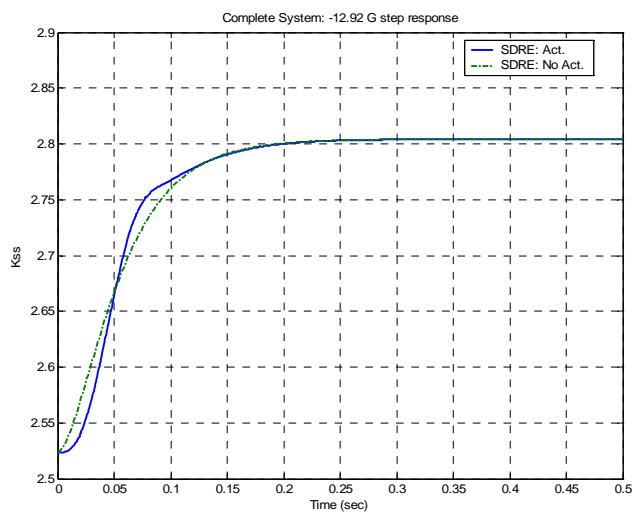


Figure 6.59. SDRE: Steady State Gain K_{ss}

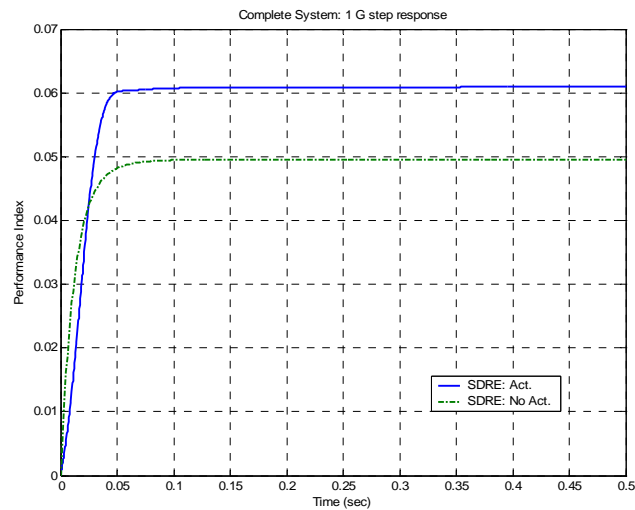


Figure 6.60. SDRE: Performance Index

Figure 6.56 shows the amount of undershoot of the achieved acceleration is greater than 30%. The undershoot of the SDRE autopilot, with actuator dynamics, is 45.18%. The amount of undershoot, however, can no longer be adjusted with the SDRE autopilot. Also seen in figure 6.56, the actuator dynamics cause a steady state oscillation due to the backlash in the system. Figure 6.57 shows that the fin dynamics do not cause much difference in the angle of attack response. Figure 6.58 shows the same steady state oscillation in the fin response. Figure 6.59 shows how the steady-state gain, calculated by the SDRE autopilot, changes with time. Figure 6.60 shows an increase in the performance index due to the presence of the fin dynamics.

6.6.6.4 Linear and Nonlinear Autopilots Compared

Figures 6.61 through 6.64 show comparisons of the SDRE autopilot and the linear, three-loop autopilot, for gain sets 2 and 4.

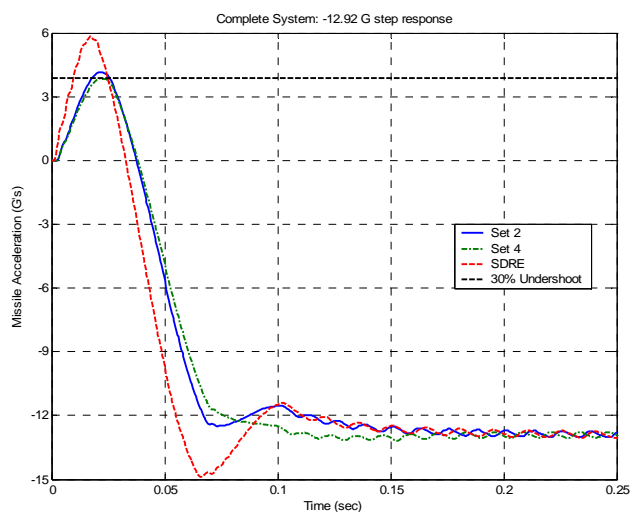


Figure 6.61. Achieved Acceleration: SDRE, Set 2, Set 4

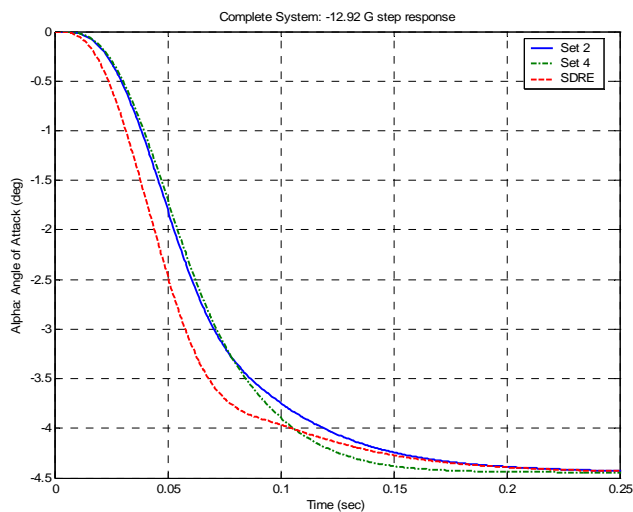


Figure 6.62. Angle of Attack: SDRE, Set 2, Set 4

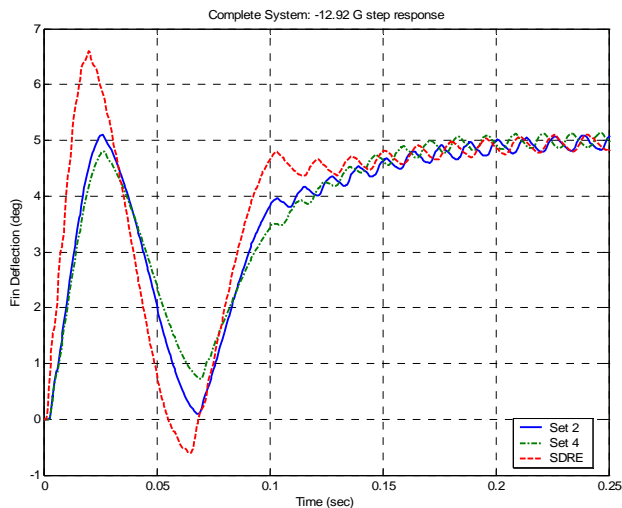


Figure 6.63. Achieved Fin Deflection: SDRE, Set 2, Set 4

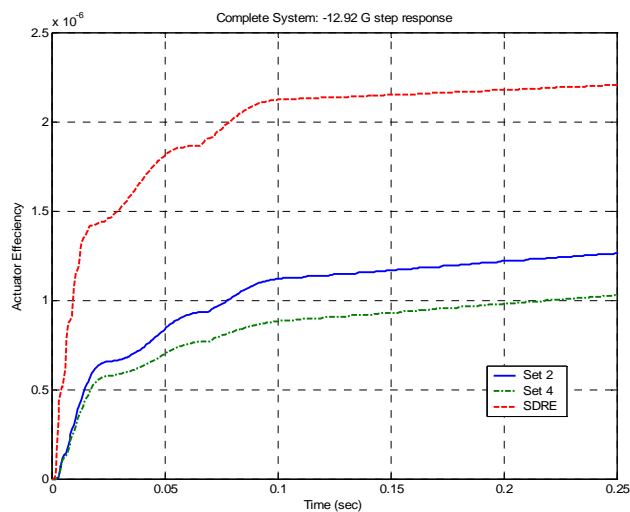


Figure 6.64. Autopilot Efficiency η_{AP} : SDRE, Set 2, Set 4

The signals are more pronounced for the SDRE autopilot, in each of the plots 6.61 through 6.64. The SDRE autopilot gives the most efficient signal in figure 6.64. It is

interesting to note that the three efficiency signals have the same general shape. The signal with the least efficiency of the three is that of gain set 4, which is the gain set with the most constraints, at 24.05% undershoot. Obviously, if gain set 4 were held as the standard of efficiency, the SDRE signal would be rejected under the assumption that some constraint has been violated. This can be seen by the 45.18% undershoot of the SDRE response. However, since the SDRE response is optimal, by solving the algebraic Riccati equation at each time step, the SDRE response may be used to find a set of linear three-loop autopilot gains by relaxing some of the design constraints.

6.7 Power Flow Analysis with Varying Mass Parameters

Often, parts procurement becomes an issue for aging systems. The introduction of new parts over time may eventually lead to the question of whether or not the current controller design is close to its optimum. If the gains of an existing control scheme are in doubt, as to whether or not more performance might be obtained in their re-optimization, or if the existing gains sufficiently control the current system, then this analysis provides a method in determining the cost benefit of a controller re-design.

6.7.1 Center of Gravity Shift

For the missile system described in this chapter, the center of gravity (cg) was simply calculated as half of the missile's length. This section shows how the autopilot efficiency signal varies with a changing center of gravity.

Shifting the cg towards the nose of the missile has a stabilizing effect. Shifting the cg towards the tail of the missile has a destabilizing effect. Figure 6.3 shows that a smaller value for the center of gravity represents a shift forward, and larger values shift the center of gravity aft. Thus, larger values are limited much more than smaller values. For this analysis the center of gravity is set at $X_{cg} = [8.5, 8.875, 9.25, 9.625, 10, 10.375, 10.75]$ ft, where 10 ft is the nominal value used in the previous sections of this chapter. This section uses the three-loop autopilot, with gain set 4, for the analysis.

Figure 6.65 shows how the achieved acceleration changes with a shift in cg.

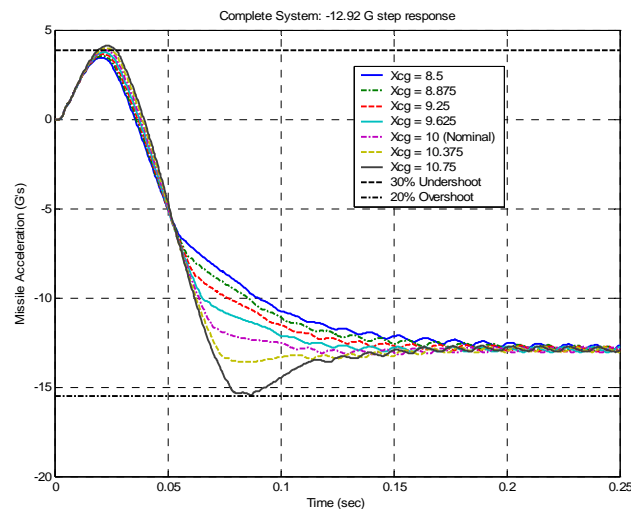


Figure 6.65. Achieved Acceleration: CG Shift

Recall that gain set 4 was designed to reach the 30% undershoot limit in the nonlinear case. Thus, a destabilizing cg shift will immediately violate this boundary. A cg shift to 10.375 ft has an undershoot of 30.87%, and a cg shift to 10.75 ft has an undershoot of 31.79%. Obviously, these constraint violations alone may not be sufficient to indicate the

need for re-optimization of the controller gains. Also, seen in figure 6.65, is that the rise times and settling times, with the cg changes, remain essentially unchanged.

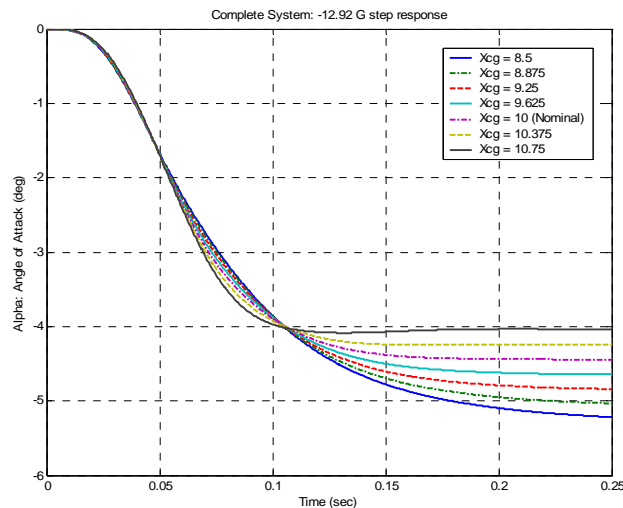


Figure 6.66. Angle of Attack: CG Shift

Figure 6.66 shows the effect that a cg shift has on the achieved angle of attack. A stabilizing cg shift requires more angle of attack to achieve the same amount of acceleration. A destabilizing shift in cg requires less angle of attack to achieve the same amount of acceleration.

Figure 6.67 shows the effect that a cg shift has on the required control effort to achieve the same amount of acceleration. A stabilizing cg shift requires more fin deflection to achieve the desired missile acceleration, and a destabilizing cg shift requires less fin deflection.

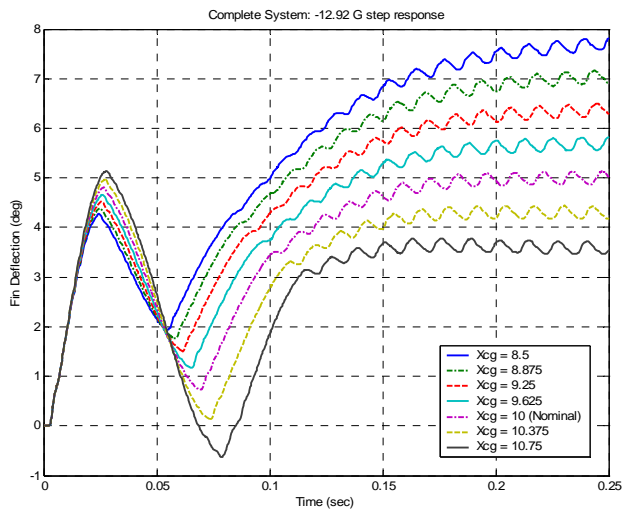


Figure 6.67. Fin Deflection: CG Shift

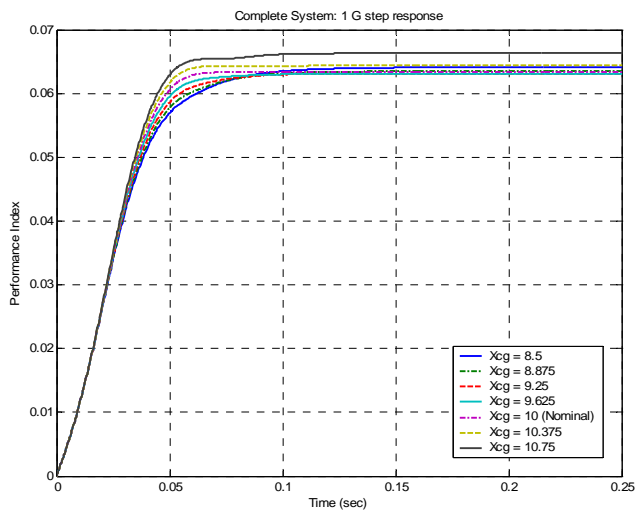


Figure 6.68. Performance Index: CG Shift

Figure 6.68 shows the effect that a cg shift has on the unit step performance index. The performance index values do not change in a monotonic fashion with a change in cg. Figure 6.69 shows the same plot zoomed in to help illustrate this further.

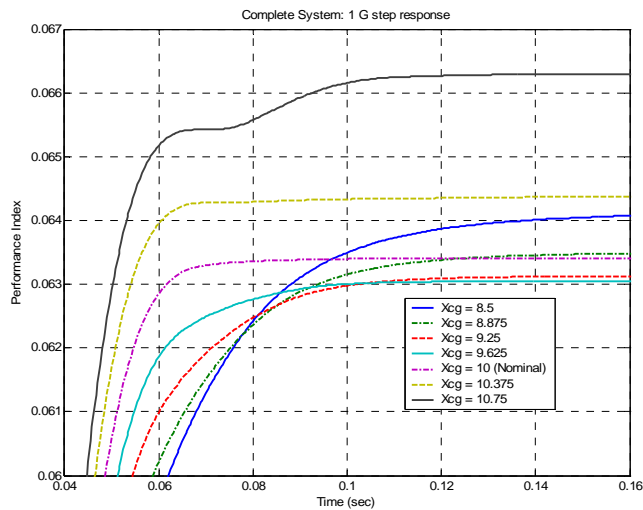


Figure 6.69. Unit Step Performance Index: CG Shift (zoom)

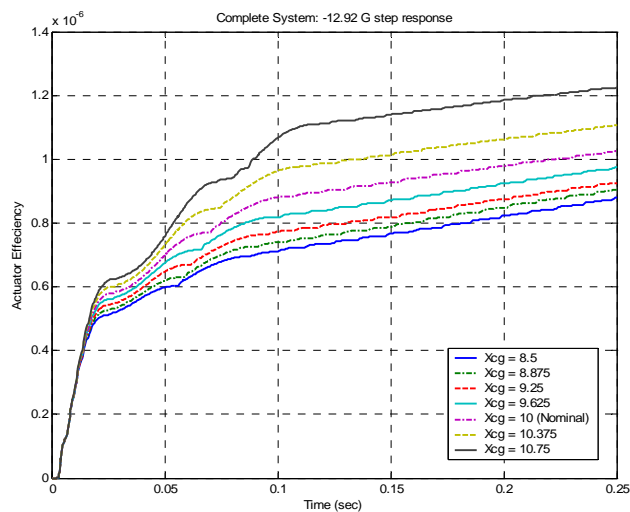


Figure 6.70. Autopilot Efficiency η_{AP} : CG Shift

Figure 6.70 shows that a stabilizing cg shift decreases the efficiency of the autopilot, and a destabilizing shift increases the efficiency of the autopilot. The efficiency signals can be used to determine a threshold to signify the need for controller redesign. Too

much increase in efficiency reduces the stability of the missile. Too much decrease in efficiency signifies the loss of performance. The design engineer can use this information to determine upper and lower efficiency values to signal the need for gain re-optimization. In the case of a new controller design, the efficiency signal can be used to help determine an allowable amount of mass parameter shift.

6.8 Conclusions

This chapter introduced a two-degree of freedom bond graph of a missile pitch dynamics model. A linear, three-loop autopilot was given for use in the gain selection process. Actuator efficiency, developed in Chapter 5, was used to measure the efficiency of different sets of autopilot gains. It was shown that the efficiency signal, of the nonlinear system, can be used to set an upper limit of efficiency to determine the violation of design constraints.

A nonlinear autopilot was developed, which solves the algebraic Riccati equation at each time step. This optimal solution was used to determine the efficiency signal of an optimal autopilot design. The optimal efficiency signal was compared to the efficiency signals of the linear autopilot with different gain sets. It was shown that the optimal autopilot produced an efficiency signal that is much more proficient than those of the linear controller with different gain sets.

Often the system parameters, such as mass, length, and center of gravity, of aging systems change over time, due to component changes, parts procurement, etc. Thus, after

time, the controller gains used may no longer perform as they did on the original system. This chapter provides a method for determining the need for a gain redesign using the efficiency measurement of the autopilot. This analysis was performed on the two-degree of freedom missile bond graph, using the linear autopilot, for a change in center of gravity. It was shown that the efficiency signal can be used to set an upper and lower limit of efficiency to signal the need of a gain redesign.

CHAPTER 7: Summary

7.1 Contributions

The contributions of this thesis are divided among; modeling, simulation, system analysis and controller design.

7.1.1 Modeling

Bond graph modeling was introduced as a means of generating dynamic equations for systems that cross multiple engineering domains. Since the bond graph deals with power flow, the modeling method can be used with equal ability in all energy domains.

A method for converting the Lagrangian of a system into a bond graph model was presented. Lagrangian and Hamiltonian elements of the system were used to create the bond graph model. Systems with complex geometries are often described by the Lagrangian. The Lagrangian can be converted into a bond graph model using the method presented here.

7.1.2 Simulation

A method was provided to simulate a bond graph model directly. Simulation difficulties arising from structural singularities, and algebraic loops were studied. The ability of the Dymola software to handle these difficulties was investigated.

A bond graph library was presented within the Dymola framework. This library takes full advantage of Dymola's ability to sort equations, solve algebraic loops, and handle structural singularities. Also, the object-oriented nature of Dymola provides the ability to use bond graph models, created with the bond graph library, in an object-oriented fashion. Object-oriented bond graph modeling is now possible.

The design and simulation of a system can be done quickly and meaningfully using the Dymola object-oriented bond graph framework. As an example of the ability to model meaningful systems, a complicated gyroscope model, created with the bond graph library, was used in four separate instances to create a gyroscopically stabilized platform model.

7.1.3 System Analysis

A nonlinear actuation system that was built using the bond-graph library was presented. Bond graph methods were used to linearize the system.

The power-flow through a bond graph model was used to compare the effectiveness of different control schemes. The analysis presented can be performed on controllers of varying architectures, and is not limited to linear systems. The controller efficiency was

defined as $\eta_{controller} = \int_o^{tf} \left[\frac{OutputEnergy}{InputEnergy} \right] dt$. The 2nd law of thermodynamics was used

to prove that the *InputEnergy* cannot be zero for any single input, physical system, after an initial input has been given.

Various control schemes were presented for both the linear system and the nonlinear system. The system response of each control scheme was compared using the definition of controller efficiency. In this manner, the ability of a controller to utilize the available energy in the system is observed.

7.1.4 Controller Design

A two-degree of freedom bond graph model of missile pitch dynamics was introduced. A linear, three-loop autopilot was given for use in the gain selection process. Actuator efficiency was used to measure the efficiency of different sets of autopilot gains. It was shown that the efficiency signal, of the nonlinear system, can be used to set an upper limit of efficiency to determine the violation of design constraints.

An SDRE autopilot was developed which must solve the algebraic Riccati equation at each time step. This optimal solution was used to determine the efficiency signal of an optimal autopilot design. The optimal efficiency signal was compared to the efficiency signals of the linear autopilot with different gain sets. It was shown that the optimal autopilot produced an efficiency signal that is much more proficient than those of the linear controller with different gain sets. The optimal efficiency provides a benchmark for the linear design.

A method is provided for determining the need for a gain redesign using the efficiency measurement of the autopilot. This analysis was performed on the two-degree of freedom missile bond graph using the linear autopilot, for a change in center of

gravity. It was shown that the efficiency signal can be used to set an upper and lower limit of efficiency to signal the need of a gain redesign.

7.2 Future Work

7.2.1 Modeling

Added insight into a model was obtained by viewing it from both Lagrangian and bond graph modeling view points. Other modeling methodologies exist for dealing with multibody systems. One such methodology is known as *Kane's method* [Kan80]. Kane's method is well suited for dealing with spacecraft dynamics. The potential exists for further insight into spacecraft dynamics modeling by mapping Kane's method into the bond graph method. Upon retrieving a bond graph model, bond graph based analysis would be possible, such as efficiency measurement, and power flow model reduction.

7.2.2 Simulation

The bond graph library presented here can be further expanded to include bond graph models of mechanical joints. In this way, a multibody library can be created where each component contains a bond graph. The advantage of having an underlying bond graph allows the modeling engineer to monitor power flow in the system.

Naturally this expansion is not limited to mechanical joints. Chemical reaction components, hydraulic components, and thermodynamic components would further expand the applications of object-oriented bond graph modeling.

7.2.3 System Analysis and Controller Design

This research uses the power flow information of a bond graph to develop a method for measuring the efficiency of a system. Power flow analysis can further be used to develop a controller that monitors and limits power flow through certain areas of the plant. By monitoring the power flow through a specific bond graph branch, and selecting a control law that keeps the power flow on this branch below a specific threshold, a control system can be created that is designed to protect specific portions of the plant. This control scheme can be set up in an observer design fashion where a bond graph model is used as the observer.

The efficiency measurement defined in this research was not limited to linear models. The power flow and causal relationships of a bond graph hold for both linear and nonlinear bond graph models. Currently, there is continuing research involved in understanding the relationships between bond graph causal loops/paths and system controllability/observability [Sar04]. The causal loop techniques developed can be further extrapolated to nonlinear systems such that controllability and observability of nonlinear systems can be determined.

APPENDIX A1: Dymola Model, SDRE Code Listing

```

model EOM_State_Space
// Author: Robert McBride
// EOM_State_Space is also referred to as SDRE
  parameter Real diam=0.5 "Missile Diam. (ft)";
  parameter Real L=7.0 "Missile Length (ft)";
  parameter Real Lp=1.0 "Radome Length (ft)";
  parameter Real Ln=1.0 "Length to wing(ft)";
  parameter Real mass=9.0 "Missile Mass (slugs)";
  parameter Real mach=2.0 "Missile Mach";
  parameter Real Vs=1000.0 "Speed of Sound (ft/s)";
  parameter Real ht=2.0/3.0 "Tail Height (ft)";
  parameter Real ctt=0.0 "Tail Tip Chord (ft)";
  parameter Real crt=2.0/3.0 "Tail Root Chord (ft)";
  parameter Real hw=0.0 "Wing Heigh (ft)";
  parameter Real ctw=0.0 "Wing Tip Chord (ft)";
  parameter Real crw=0.0 "Wing Root Chord (ft)";
  parameter Real altitude=1000.0 "Missile Alt. (ft) assumed < 30kft.";
  parameter Real wric=2.0 "Qh[1,1] weight";
  parameter Real Rric=1.0 "R weight";
protected
  constant Real DPR=57.2957795130823 "Deg. per rad.";
  constant Real pi=3.14159265358979 "PI";
  Real beta=sqrt(mach^2 - 1.0);
  Real sref=pi*diam^2/4.0;
  Real splan=L*diam;
  Real st=.5*ht*(ctt + crt);
  Real sw=.5*hw*(ctw + crw);
  Real rho=.002378*exp(-altitude/30000.0);
  Real Vm=mach*Vs;
  Real Q=rho*Vm^2/2.0;
  Real Xcpn=.67*Lp;
  Real An=.67*Lp*diam;
  Real Ab=(L - Lp)*diam;
  Real Xcpb=(.67*An*Lp + Ab*(Lp + .5*(L - Lp)))/(An + Ab);
  Real Xhl=L - .5;
  Real Xcg=L/2;
  Real Xcpw=Lp + Ln + .7*crw - .2*ctw;
  Real Iyy=mass*L^2/12;
  Real alpha;
  Real thetad;
  Real Az_err;
  Real a11;
  Real a21;
  Real b1;
  Real b2;
  Real h1;
  Real l1;
  Real I[2, 2];
  Real A[2, 2];
  Real B[2, 1];
  Real C[2, 2];
  Real Ci[2, 2];
  Real D[2, 1];
  Real Qw[1, 1];
  Real Rw[1, 1];
  Real Qh[2, 2];
  Real Rh[1, 1];
  Real Rhi[1, 1];
  Real Sh[2, 1];
  Real Ah[2, 2];
  Real Bh[2, 2];
  Real Ch[2, 2];
  Real Pric[2, 2];

```

```

Real Hric[1, 2];
Real Lric[1, 1];
Real Ktemp[1, 2];
Real Temp[1, 1];
Real Temp2[1, 1];
Real Temp2i[1, 1];
Real Kopt[1, 2];
Real X[2, 1];
Real Y[2, 1];
Real I1[1, 1];
Real ind[2, 1];
Real Acc[2, 2];
Real Acci[2, 2];
Real Bpc[2, 1];
Real Ccc[1, 2];
Real Dpc[1, 1];
Real KDtmpi[1, 1];
Real KDtmp[1, 1];
Real KD[1, 1];
Real Kssi[1, 1];
output Real Kss[1, 1](start=[1]);
output Real Uopt[1, 1];
public
  Modelica.Blocks.Interfaces.InPort AZ_error_in(n=1) annotation (extent=[-
    10, -100; 10, -80], rotation=90);
  Modelica.Blocks.Interfaces.OutPort Kss_out(n=1) annotation (extent=[100,
    -70; 120, -50]);
  Modelica.Blocks.Interfaces.OutPort U_opt_out(n=1) annotation (extent=[100
    , 50; 120, 70]);
  Modelica.Blocks.Interfaces.InPort alpha_in(n=1) annotation (extent=[-120
    , 50; -100, 70]);
  Modelica.Blocks.Interfaces.InPort q_in(n=1) annotation (extent=[-120, -70
    ; -100, -50]);
  Linear_Algebra.Riccati4 Riccati4 annotation (extent=[-40, -20; 20, 40]);
equation

//Read in the input signals
alpha = alpha_in.signal[1];
thetad = q_in.signal[1];
Az_err = AZ_error_in.signal[1];
//Fill the state vector and Meas. vector Y
X[1, 1] = alpha;
X[2, 1] = thetad;
Y[1, 1] = Az_err;
Y[2, 1] = thetad;
//Assign weights
Qw[1, 1] = wric;
Rw[1, 1] = Rric;
I1[1, 1] = 1.0;

//Create the state space matrices A, B, C, D, H, L
a11 = -Q/(Vm*mass)*(2*sref + 1.5*splan*alpha + 8*sw/beta + 8*st/beta);
a21 = Q*diam/Iyy*(2*sref*(Xcg - Xcpn) + 1.5*splan*alpha*(Xcg - Xcpb) + (
  Xcg - Xcpw)*8*sw/beta + 8*st*(Xcg - Xhl)/beta);
b1 = -8*Q*st/(Vm*mass*beta);
b2 = (Xcg - Xhl)*8*st*Q*diam/(beta*Iyy);
// D = -8*Q*st/(Vm*mass*beta);
h1 = -a11*Vm;
l1 = -b1*Vm;
A[1, 1] = a11;
A[1, 2] = 1.0;
A[2, 1] = a21;
A[2, 2] = 0.0;
B[1, 1] = b1;
B[2, 1] = b2;
C[1, 1] = h1;

```

```

C[1, 2] = 0.0;
C[2, 1] = 0.0;
C[2, 2] = 1.0;
D[1, 1] = 11;
D[2, 1] = 0.0;
Hric[1, 1] = h1;
Hric[1, 2] = 0.0;
Lric[1, 1] = 11;
//Create the matrix inputs to Riccati4 Ah, Bh, Ch
Qh = transpose(Hric)*Qw*Hric;
Sh = transpose(Hric)*Qw*Lric;
Rh = Rw + transpose(Lric)*Qw*Lric;
Rhi[1, 1] = 1/(Rh[1, 1]);
Ah = A - B*Rhi*transpose(Sh);
Bh = B*Rhi*transpose(B);
Ch = Qh - Sh*Rhi*transpose(Sh);
//Call Riccati4
for i in 1:2 loop
    for j in 1:2 loop
        Riccati4.Amat_input.signal[j + (i - 1)*2] = Ah[j, i];
        Riccati4.Bmat_input.signal[j + (i - 1)*2] = Bh[j, i];
        Riccati4.Cmat_input.signal[j + (i - 1)*2] = Ch[j, i];
        I[j, i] = if (j == i) then 1.0 else 0.0;
    end for;
end for;
//Fill the matrix Pric with the ARE Solution
for i in 1:2 loop
    for j in 1:2 loop
        Pric[j, i] = Riccati4.RP_output.signal[j + (i - 1)*2];
    end for;
end for;
//Calculate optimal feedback gain.
Ktemp = -Rhi*(transpose(B)*Pric + transpose(Sh));
Ci*C = I;
Temp = Ktemp*Ci*D;
Temp2[1, 1] = Temp[1, 1] + 1;
Temp2i*Temp2 = I1;
Kopt = Temp2i*Ktemp*Ci;
//Calculate the optimal input Uopt.
Uopt = Kopt*Y;
U_opt_out.signal[1] = Uopt[1, 1];
KDtmp = Kopt*D;
KDtmpi[1, 1] = 1 - KDtmp[1, 1];
KD*KDtmpi = I1;

ind[1, 1] = 1;
ind[2, 1] = 0;

//Calculate the closed loop matrices
Acc = A + B*KD*Kopt*C;
Acci*Acc = I;
Bpc = B*KD*Kopt*ind;
Ccc = Hric + Lric*KD*Kopt*C;
Dpc = Lric*KD*Kopt*ind;
//Calculate Kss
Kssi = Ccc*Acci*Bpc - Dpc;
Kss*Kssi = I1;

Kss_out.signal[1] = Kss[1, 1];

annotation (Diagram, Icon(
    Rectangle(extent=[-100, 100; 100, -100], style(fillColor=43,
        fillPattern=1)),
    Text(
        extent=[-58, 138; 54, 94],
        style(fillColor=43, fillPattern=1),
        string="SDRE"),

```

```
Text(
    extent=[-16, -62; 18, -80],
    style(fillColor=43, fillPattern=1),
    string="AZ_Err"),
Text(
    extent=[-94, 68; -60, 50],
    style(fillColor=43, fillPattern=1),
    string="Alpha"),
Text(
    extent=[-96, -52; -62, -70],
    style(fillColor=43, fillPattern=1),
    string="q"),
Text(
    extent=[60, 70; 94, 52],
    style(fillColor=43, fillPattern=1),
    string="Uopt"),
Text(
    extent=[66, -50; 100, -68],
    style(fillColor=43, fillPattern=1),
    string="Kss"));
end EOM_State_Space;
```

APPENDIX A2: Dymola Model, Riccati4 Code Listing

```

model Riccati4
  Linear_Algebra.Hamiltonian_eig4 Heig4 annotation (extent=[-50, -40; 30, 40]);
  // Author: Robert McBride
  // Solve the Algebraic Riccati Eq.  $A'X + XA - X*B*X + C = 0$ 
  input Real A[2, 2];
  input Real R[2, 2];
  input Real mQ[2, 2];
  output Integer indx1;
  output Integer indx2;
  output Real RP[2, 2];
  output Real IP[2, 2];
protected
  Real mAt[2, 2];
  Real RX1[2, 2];
  Real RX2[2, 2];
  Real IX1[2, 2];
  Real IX2[2, 2];
  Real RX1inv[2, 2];
  Real IX1inv[2, 2];
  Real H[4, 4];
  Real RV1[4, 1];
  Real RV2[4, 1];
  Real RV3[4, 1];
  Real RV4[4, 1];
  Real RV_used[4, 4];
  Real RV_lout[4, 1];
  Real IV_lout[4, 1];
  Real RV_2out[4, 1];
  Real IV_2out[4, 1];
  Real IV1[4, 1];
  Real IV2[4, 1];
  Real IV3[4, 1];
  Real IV4[4, 1];
  Real IV_used[4, 4];
  Real Re1;
  Real Re2;
  Real Re3;
  Real Re4;
  Real tst1;
  Real tst2;
  Real tst3;
  Real tst4;
  Real detX1inv[2];
  Real detX1inv1[2];
  Real detX1inv2[2];
  Real div1[2];
  Real div2[2];
  Real div3[2];
  Real div4[2];
  // constant Integer i1=3;
  // constant Integer i2=4;
  constant Integer n=2;
public
  Modelica.Blocks.Interfaces.InPort Amat_input(n=4) annotation (extent=[-100
    , 50; -80, 70]);
public
  Modelica.Blocks.Interfaces.InPort Bmat_input(n=4) annotation (extent=[-100
    , -10; -80, 10]);
public
  Modelica.Blocks.Interfaces.InPort Cmat_input(n=4) annotation (extent=[-100
    , -70; -80, -50]);
  Modelica.Blocks.Interfaces.OutPort RP_output(n=4) annotation (extent=[80,
    20; 100, 40]);

```

```

Modelica.Blocks.Interfaces.OutPort IP_output(n=4) annotation (extent=[80, -
40; 100, -20]);
equation
//Build the Hamiltonain
for i in 1:n loop
  for j in 1:n loop
    A[j, i] = Amat_input.signal[j + (i - 1)*n];
    R[j, i] = -Bmat_input.signal[j + (i - 1)*n];
    mQ[j, i] = -Cmat_input.signal[j + (i - 1)*n];
  end for;
end for;
mAt = -transpose(A);
H[1, 1] = A[1, 1];
H[1, 2] = A[1, 2];
H[2, 1] = A[2, 1];
H[2, 2] = A[2, 2];
H[1, 3] = R[1, 1];
H[1, 4] = R[1, 2];
H[2, 3] = R[2, 1];
H[2, 4] = R[2, 2];
H[3, 1] = mQ[1, 1];
H[3, 2] = mQ[1, 2];
H[4, 1] = mQ[2, 1];
H[4, 2] = mQ[2, 2];
H[3, 3] = mAt[1, 1];
H[3, 4] = mAt[1, 2];
H[4, 3] = mAt[2, 1];
H[4, 4] = mAt[2, 2];
//Call Heig4
for i in 1:4 loop
  for j in 1:4 loop
    Heig4.Amat_input.signal[j + (i - 1)*4] = H[j, i];
  end for;
end for;
//Collect the eigenvectors output from Heig4
for i in 1:2*n loop
  RV1[i, 1] = Heig4.Reigvec_output.signal[i];
  RV2[i, 1] = Heig4.Reigvec_output.signal[i + 4];
  RV3[i, 1] = Heig4.Reigvec_output.signal[i + 4 + 4];
  RV4[i, 1] = Heig4.Reigvec_output.signal[i + 4 + 4 + 4];
  IV1[i, 1] = Heig4.Ieigvec_output.signal[i];
  IV2[i, 1] = Heig4.Ieigvec_output.signal[i + 4];
  IV3[i, 1] = Heig4.Ieigvec_output.signal[i + 4 + 4];
  IV4[i, 1] = Heig4.Ieigvec_output.signal[i + 4 + 4 + 4];
end for;
Re1 = Heig4.eig_output.signal[1];
Re2 = Heig4.eig_output.signal[3];
Re3 = Heig4.eig_output.signal[5];
Re4 = Heig4.eig_output.signal[7];
//Find eigenvalues with neg. real parts
tst1 = if (Re1 < 0.0) then 1.0 else 0.0;
tst2 = if (Re2 < 0.0) then 1.0 else 0.0;
tst3 = if (Re3 < 0.0) then 1.0 else 0.0;
tst4 = if (Re4 < 0.0) then 1.0 else 0.0;
//Collect the corresponing eigenvectors
for j in 1:4 loop
  RV_used[j, 1] = tst1*RV1[j, 1];
  RV_used[j, 2] = tst2*RV2[j, 1];
  RV_used[j, 3] = tst3*RV3[j, 1];
  RV_used[j, 4] = tst4*RV4[j, 1];
  IV_used[j, 1] = tst1*IV1[j, 1];
  IV_used[j, 2] = tst2*IV2[j, 1];
  IV_used[j, 3] = tst3*IV3[j, 1];
  IV_used[j, 4] = tst4*IV4[j, 1];
end for;
//Find the indices of the stabilizing eigenvectors

```

```

indx1 = if (RV_used[1, 1] > .5) then 1 else if (RV_used[1, 2] > .5) then 2
      else 3;
indx2 = if (indx1 == 1) then if (RV_used[1, 2] > .5) then 2 else if (
      RV_used[1, 3] > .5) then 3 else 4 else if (indx1 == 2) then if (RV_used[1
      , 3] > .5) then 3 else 4 else 4;
RV_1out[:, 1] = if (indx1 == 1) then RV_used[:, 1] else if (indx1 == 2)
      then RV_used[:, 2] else RV_used[:, 3];
RV_2out[:, 1] = if (indx2 == 2) then RV_used[:, 2] else if (indx2 == 3)
      then RV_used[:, 3] else RV_used[:, 4];
IV_1out[:, 1] = if (indx1 == 1) then IV_used[:, 1] else if (indx1 == 2)
      then IV_used[:, 2] else IV_used[:, 3];
IV_2out[:, 1] = if (indx2 == 2) then IV_used[:, 2] else if (indx2 == 3)
      then IV_used[:, 3] else IV_used[:, 4];
//Create X1 and X2 to solve P = X2*inv(X1)
RX1[1, 1] = RV_1out[1, 1];
RX1[1, 2] = RV_2out[1, 1];
RX1[2, 1] = RV_1out[2, 1];
RX1[2, 2] = RV_2out[2, 1];
RX2[1, 1] = RV_1out[3, 1];
RX2[1, 2] = RV_2out[3, 1];
RX2[2, 1] = RV_1out[4, 1];
RX2[2, 2] = RV_2out[4, 1];
IX1[1, 1] = IV_1out[1, 1];
IX1[1, 2] = IV_2out[1, 1];
IX1[2, 1] = IV_1out[2, 1];
IX1[2, 2] = IV_2out[2, 1];
IX2[1, 1] = IV_1out[3, 1];
IX2[1, 2] = IV_2out[3, 1];
IX2[2, 1] = IV_1out[4, 1];
IX2[2, 2] = IV_2out[4, 1];
//Calculate inv(X1)
detXlinv1 = complex_mult(RX1[1, 1], IX1[1, 1], RX1[2, 2], IX1[2, 2]);
detXlinv2 = complex_mult(RX1[2, 1], IX1[2, 1], RX1[1, 2], IX1[1, 2]);
detXlinv[1] = detXlinv1[1] - detXlinv2[1];
detXlinv[2] = detXlinv1[2] - detXlinv2[2];

//detXlinv = complex_mult(RX1[1, 1], IX1[1, 1], RX1[2, 2], IX1[2, 2])-
complex_mult(RX1[2, 1], IX1[2, 1], RX1[1, 2], IX1[1, 2]);
div1 = complex_div(RX1[2, 2], IX1[2, 2], detXlinv[1], detXlinv[2]);
div2 = complex_div(-RX1[1, 2], -IX1[1, 2], detXlinv[1], detXlinv[2]);
div3 = complex_div(-RX1[2, 1], -IX1[2, 1], detXlinv[1], detXlinv[2]);
div4 = complex_div(RX1[1, 1], IX1[1, 1], detXlinv[1], detXlinv[2]);

RXlinv[1, 1] = div1[1];
RXlinv[1, 2] = div2[1];
RXlinv[2, 1] = div3[1];
RXlinv[2, 2] = div4[1];
IXlinv[1, 1] = div1[2];
IXlinv[1, 2] = div2[2];
IXlinv[2, 1] = div3[2];
IXlinv[2, 2] = div4[2];

//Calculate P = RP+IPi (IP should always be zero)
RP = RX2*RXlinv - IX2*IXlinv;
IP = IX2*RXlinv + RX2*IXlinv;
for i in 1:2 loop
  for j in 1:2 loop
    RP_output.signal[j + (i - 1)*2] = RP[j, i];
    IP_output.signal[j + (i - 1)*2] = IP[j, i];
  end for;
end for;

annotation (Diagram(
  Text(extent=[-92, 66; -50, 52], string="A"),
  Text(extent=[-92, 6; -50, -8], string="B"),
  Text(extent=[-92, -54; -50, -68], string="C"),

```



```
Text(extent=[36, 38; 78, 24], string="Real(P)",
Text(extent=[38, -22; 80, -36], string="Imag(P)"), Icon(Text(
  extent=[-64, 102; 70, 78],
  style(fillColor=6, fillPattern=1),
  string="%name"), Rectangle(extent=[-80, 80; 80, -80], style(fillColor
    =77, fillPattern=1)))));
end Riccati4;
```

APPENDIX A3: Dymola Model, Heig4 Code Listing

```

model Hamiltonian_eig4
//Author: Robert McBride
//This routine is referred to as Heig4
  input Real A[4, 4];
  output Real eig1[4] "Real(e1) imag(e1) Real(e2) imag(e2)";
  output Real eig2[4] "Real(e3) imag(e3) Real(e4) imag(e4)";
  output Real RV[4, 4];
  output Real IV[4, 4];
protected
  constant Integer n=4;
  Real B[n, 1];
  Real AB1[n, 1];
  Real AB2[n, 1];
  Real AB3[n, 1];
  Real cm[n, n];
  Real icm[n, n];
  Real I[n, n];
  Real cpoly[1, n + 1];
  Real lr[1, n];
  Real lr1[1, n];
  Real lr2[1, n];
  Real lr3[1, n];
  Real T[n, n];
  Real iT[n, n];
  Real Ah[n, n];
  Real a1;
  Real a2;
  Real b1;
  Real b2;
  Real RVh[4, 4];
  Real IVh[4, 4];
  Real e1_2[2];
  Real e2_2[2];
  Real e3_2[2];
  Real e4_2[2];
  Real e1_3[2];
  Real e2_3[2];
  Real e3_3[2];
  Real e4_3[2];
public
  Modelica.Blocks.Interfaces.InPort Amat_input(n=16) annotation (extent=[-100
    , -10; -80, 10]);
  Modelica.Blocks.Interfaces.OutPort eig_output(n=8) annotation (extent=[80,
    -10; 100, 10]);
  Modelica.Blocks.Interfaces.OutPort Ieigvec_output(n=16) annotation (extent=
    [80, -70; 100, -50]);
  Modelica.Blocks.Interfaces.OutPort Reigvec_output(n=16) annotation (extent=
    [80, 50; 100, 70]);
  Linear_Algebra.gen_eigs gen_eigs annotation (extent=[-20, -30; 40, 30]);
equation
//Create the A matrix
  for i in 1:n loop
    for j in 1:n loop
      A[j, i] = Amat_input.signal[j + (i - 1)*4];
    end for;
  end for;
//Create B = [0 0 0 1]'
  for i in 1:n - 1 loop
    B[i, 1] = 0.0;
  end for;
  B[4, 1] = 1.0;
//Create the controllability matrix
  AB1 = A*B;

```

```

AB2 = A*AB1;
AB3 = A*AB2;
for j in 1:n loop
    cm[j, 1] = B[j, 1];
end for;
for j in 1:n loop
    cm[j, 2] = AB1[j, 1];
end for;
for j in 1:n loop
    cm[j, 3] = AB2[j, 1];
end for;
for j in 1:n loop
    cm[j, 4] = AB3[j, 1];
end for;
for i in 1:n loop
    for j in 1:n loop
        I[i, j] = if (i == j) then 1.0 else 0.0;
    end for;
end for;
//Invert the controllability matrix
icm*cm = I;
//Retrieve the last row of the inverted controllability matrix
for i in 1:n loop
    lr[1, i] = icm[n, i];
end for;
//Create the transformation matrix T
lr1 = lr*A;
lr2 = lr1*A;
lr3 = lr2*A;
for j in 1:n loop
    T[1, j] = lr[1, j];
end for;
for j in 1:n loop
    T[2, j] = lr1[1, j];
end for;
for j in 1:n loop
    T[3, j] = lr2[1, j];
end for;
for j in 1:n loop
    T[4, j] = lr3[1, j];
end for;
iT*T = I;
//Transform A to controller canonical form
Ah = T*A*iT;
//Create the characteristic polynomial
cpoly[1, 1] = 1;
for j in 1:n loop
    cpoly[1, j + 1] = -Ah[4, 4 - j + 1];
end for;
//Break the 4th order char. poly. into two 2nd order char. polys.
b1 = cpoly[1, 2] - a1;
a2*b2 = cpoly[1, 5];
a2 = b2;
b1 = sqrt(2*b2 - cpoly[1, 3]);

//Find the roots of the two 2nd order char. polys.
//    eig1[1]=real(eigval1),eig1[2]=imag(eigval1)
//    eig1[3]=real(eigval2),eig1[4]=imag(eigval2)
eig1 = quadroots(1, a1, a2);
eig2 = quadroots(1, b1, b2);

//Pass out the eigenvalues
for i in 1:n loop
    eig_output.signal[i] = eig1[i];
end for;
for i in 1:n loop

```

```

    eig_output.signal[i + 4] = eig2[i];
end for;

//Create the eigenvector Vandermonde matrix
for i in 1:n loop
    RVh[1, i] = 1;
    IVh[1, i] = 0;
end for;
RVh[2, 1] = eig1[1];
IVh[2, 1] = eig1[2];
RVh[2, 2] = eig1[3];
IVh[2, 2] = eig1[4];
RVh[2, 3] = eig2[1];
IVh[2, 3] = eig2[2];
RVh[2, 4] = eig2[3];
IVh[2, 4] = eig2[4];
e1_2 = complex_mult(RVh[2, 1], IVh[2, 1], RVh[2, 1], IVh[2, 1]);
e2_2 = complex_mult(RVh[2, 2], IVh[2, 2], RVh[2, 2], IVh[2, 2]);
e3_2 = complex_mult(RVh[2, 3], IVh[2, 3], RVh[2, 3], IVh[2, 3]);
e4_2 = complex_mult(RVh[2, 4], IVh[2, 4], RVh[2, 4], IVh[2, 4]);
e1_3 = complex_mult(e1_2[1], e1_2[2], RVh[2, 1], IVh[2, 1]);
e2_3 = complex_mult(e2_2[1], e2_2[2], RVh[2, 2], IVh[2, 2]);
e3_3 = complex_mult(e3_2[1], e3_2[2], RVh[2, 3], IVh[2, 3]);
e4_3 = complex_mult(e4_2[1], e4_2[2], RVh[2, 4], IVh[2, 4]);
RVh[3, 1] = e1_2[1];
IVh[3, 1] = e1_2[2];
RVh[3, 2] = e2_2[1];
IVh[3, 2] = e2_2[2];
RVh[3, 3] = e3_2[1];
IVh[3, 3] = e3_2[2];
RVh[3, 4] = e4_2[1];
IVh[3, 4] = e4_2[2];
RVh[4, 1] = e1_3[1];
IVh[4, 1] = e1_3[2];
RVh[4, 2] = e2_3[1];
IVh[4, 2] = e2_3[2];
RVh[4, 3] = e3_3[1];
IVh[4, 3] = e3_3[2];
RVh[4, 4] = e4_3[1];
IVh[4, 4] = e4_3[2];

//Call gen_eigs
gen_eigs.eig_input.signal[1] = eig1[1];
gen_eigs.eig_input.signal[2] = eig1[3];
gen_eigs.eig_input.signal[3] = eig2[1];
gen_eigs.eig_input.signal[4] = eig2[3];
gen_eigs.eig_input.signal[5] = eig1[2];
gen_eigs.eig_input.signal[6] = eig1[4];
gen_eigs.eig_input.signal[7] = eig2[2];
gen_eigs.eig_input.signal[8] = eig2[4];

//Use inv(T) to calculate the eigenvectors from the
//control canonical eigenvectors
RV = iT*RVh;
IV = iT*IVh;
for i in 1:4 loop
    gen_eigs.eig_vec_input.signal[i] = RV[i, 1];
    gen_eigs.eig_vec_input.signal[i + 4] = RV[i, 2];
    gen_eigs.eig_vec_input.signal[i + 8] = RV[i, 3];
    gen_eigs.eig_vec_input.signal[i + 12] = RV[i, 4];
    gen_eigs.eig_vec_input.signal[i + 16] = IV[i, 1];
    gen_eigs.eig_vec_input.signal[i + 20] = IV[i, 2];
    gen_eigs.eig_vec_input.signal[i + 24] = IV[i, 3];
    gen_eigs.eig_vec_input.signal[i + 28] = IV[i, 4];
end for;

//Pass out the eigenvectors

```

```

for j in 1:n loop
    Reigvec_output.signal[j + (i - 1)*4] = RVh[j, i];
    Ieigvec_output.signal[j + (i - 1)*4] = IVh[j, i];
end for;
end for;*/
for i in 1:16 loop
    Reigvec_output.signal[i] = gen_eigs.eig_vec_output.signal[i];
    Ieigvec_output.signal[i] = gen_eigs.eig_vec_output.signal[i + 16];
end for;

annotation (Diagram(
    Text(extent=[-98, 36; -30, -2], string="Hamiltonian Matrix (4,4)"),
    Text(extent=[-6, 80; 90, 40], string="Real(Eigen Vectors) 4x4"),
    Text(extent=[-6, -40; 90, -80], string="Imag(Eigen Vectors) 4x4"),
    Text(extent=[60, -6; 124, -32], string="Eigen Values 1x8"), Icon(
        Rectangle(extent=[-80, 80; 80, -80], style(fillColor=45)), Text(
            extent=[-56, 106; 52, 82], string="%name"));
connect(Amat_input, gen_eigs.Amat_input) annotation (points=[-88, 0; -17, 0
], style(
    color=3,
    fillColor=6,
    fillPattern=1));
end Hamiltonian_eig4;

```

APPENDIX A4: Dymola Model, Gen_Eigs Code Listing

```

model gen_eigs
// Author: Robert McBride
  input Real A[4, 4];
  input Real eig1[4];
  input Real eig2[4];
  input Real RV[4, 4];
  input Real IV[4, 4];
  output Real eig_vecs[32];
protected
  constant Real tol=0.00001;
  constant Integer n=4;
  Integer tst12;
  Integer tst13;
  Integer tst14;
  Integer tst23;
  Integer tst24;
  Integer tst34;
  Real R1_in[n, 1];
  Real R2_in[n, 1];
  Real R3_in[n, 1];
  Real R4_in[n, 1];
  Real I1_in[n, 1];
  Real I2_in[n, 1];
  Real I3_in[n, 1];
  Real I4_in[n, 1];
  Real R1_out[n, 1];
  Real R2_out[n, 1];
  Real R3_out[n, 1];
  Real R4_out[n, 1];
  Real I1_out[n, 1];
  Real I2_out[n, 1];
  Real I3_out[n, 1];
  Real I4_out[n, 1];
  Real RLam1_I[n, n];
  Real RLam2_I[n, n];
  Real RLam3_I[n, n];
  Real RLam4_I[n, n];
  Real gen_R2[n, 1];
  Real gen_R3[n, 1];
  Real gen_R4[n, 1];
/* public
Modelica.Blocks.Interfaces.InPort Amat_input(n=16) annotation (extent=[-100
, -10; -80, 10]);
Modelica.Blocks.Interfaces.InPort eig_input(n=8) annotation (extent=[-100
, -70; -80, -50]);
Modelica.Blocks.Interfaces.InPort eig_vec_input(n=32) annotation (extent=[-100
, 50; -80, 70]);
Modelica.Blocks.Interfaces.OutPort eig_vec_output(n=32) annotation (extent=[80,
-10; 100, 10]);*/
equation
//Fill the 4x4 A matrix with the elements of the 16x1 input vector Amat_input
  for i in 1:n loop
    for j in 1:n loop
      A[j, i] = Amat_input.signal[j + (i - 1)*4];
    end for;
  end for;
//Fill the 1x2 eig vectors with the elements of the 8x1 input vector eig_input
  eig1[1] = eig_input.signal[1];
  eig2[1] = eig_input.signal[2];
  eig3[1] = eig_input.signal[3];
  eig4[1] = eig_input.signal[4];
  eig1[2] = eig_input.signal[5];
  eig2[2] = eig_input.signal[6];

```

```

    eig3[2] = eig_input.signal[7];
    eig4[2] = eig_input.signal[8];
//Fill the 4x4 RV, and IV matrices with the 32x1 input vector eig_vec_input
    for i in 1:n loop
        for j in 1:n loop
            RV[j, i] = eig_vec_input.signal[j + (i - 1)*4];
            IV[j, i] = eig_vec_input.signal[j + (i - 1)*4 + 16];
        end for;
    end for;
//Expand the eigenvector matrix to 4 separate vectors
    for i in 1:n loop
        R1_in[i, 1] = RV[i, 1];
        R2_in[i, 1] = RV[i, 2];
        R3_in[i, 1] = RV[i, 3];
        R4_in[i, 1] = RV[i, 4];
        I1_in[i, 1] = IV[i, 1];
        I2_in[i, 1] = IV[i, 2];
        I3_in[i, 1] = IV[i, 3];
        I4_in[i, 1] = IV[i, 4];
//Make a lambda*I matrix for each eigen value with the real part of the eigen value.
//For the 4th order system if the gen. eigvec. is needed then the imag. part is zero.
        for j in 1:4 loop
            RLam1_I[i, j] = if(i==j) eig1[1] else 0;
            RLam2_I[i, j] = if(i==j) eig2[1] else 0;
            RLam3_I[i, j] = if(i==j) eig3[1] else 0;
            RLam4_I[i, j] = if(i==j) eig4[1] else 0;
        end for;
    end for;

//Calculate an generalized eigenvector, gen_R*, whether its needed or not.
    (RLam2_I - A)*gen_R2 = -R2_in;
    (RLam3_I - A)*gen_R3 = -R3_in;
    (RLam4_I - A)*gen_R4 = -R4_in;

//Test to see if the generalized eigenvectors are needed.
    tst12 = if (abs(eig1[1] - eig2[1]) < tol) then if (abs(eig1[2] - eig2[2])
        < tol) then 1 else 0 else 0;
    tst13 = if (abs(eig1[1] - eig3[1]) < tol) then if (abs(eig1[2] - eig3[2])
        < tol) then 1 else 0 else 0;
    tst14 = if (abs(eig1[1] - eig4[1]) < tol) then if (abs(eig1[2] - eig4[2])
        < tol) then 1 else 0 else 0;
    tst23 = if (abs(eig2[1] - eig3[1]) < tol) then if (abs(eig2[2] - eig3[2])
        < tol) then 1 else 0 else 0;
    tst24 = if (abs(eig2[1] - eig4[1]) < tol) then if (abs(eig2[2] - eig4[2])
        < tol) then 1 else 0 else 0;
    tst34 = if (abs(eig3[1] - eig4[1]) < tol) then if (abs(eig3[2] - eig4[2])
        < tol) then 1 else 0 else 0;

//The first eigenvector is never generalized
    R1_out = R1_in;
    I1_out = I1_in;
    I2_out = I2_in;
    I3_out = I3_in;
    I4_out = I4_in;

//Assign a generalized eigenvector if needed else pass back the vector that was input.
    R2_out = if (tst12 == 1) then gen_R2
        else if (tst23 == 1) then gen_R2
        else if (tst24 == 1) then gen_R2
        else R2_in;
    R3_out = if (tst13 == 1) then gen_R3
        else if (tst34 == 1) then gen_R3
        else R3_in;
    R4_out = if (tst14 == 1) then gen_R4
        else R4_in;
    for i in 1:4 loop

```

```
    eig_vecs[i] = R1_in[i, 1];
end for;
for i in 1:4 loop
    eig_vecs[i + 4] = R2_in[i, 1];
end for;
for i in 1:4 loop
    eig_vecs[i + 8] = R3_in[i, 1];
end for;
for i in 1:4 loop
    eig_vecs[i + 12] = R4_in[i, 1];
end for;
for i in 1:4 loop
    eig_vecs[i + 16] = I1_in[i, 1];
end for;
for i in 1:4 loop
    eig_vecs[i + 20] = I2_in[i, 1];
end for;
for i in 1:4 loop
    eig_vecs[i + 24] = I3_in[i, 1];
end for;
for i in 1:4 loop
    eig_vecs[i + 28] = I4_in[i, 1];
end for;
end gen_eigs;
```


APPENDIX A5: Dymola Models, Misc. Functions, Code Listing

A5.1 QuadRoots

```

function quadroots
//Author: Robert McBride
//Solve a*S^2 + b*S + C = 0
  input Real a;
  input Real b;
  input Real c;
  output Real roots[4] "Real(root1) imag(root1) Real(root2) imag(root2)";
protected
  Real i1;
  Real i2;
  Real r1;
  Real r2;
algorithm
//Calculate the imaginary part of root 1.
  i1 := if (b^2 > 4*a*c) then 0.0 else sqrt(4*a*c - b^2)/(2*a);
//Calculate the imaginary part of root 2.
  i2 := -i1;
//Calculate the real part of root 1.
  r1 := if (b^2 > 4*a*c) then -b/(2*a) + sqrt(b^2 - 4*a*c)/(2*a) else -b/(2*a);
//Calculate the real part of root 2.
  r2 := if (b^2 > 4*a*c) then -b/(2*a) - sqrt(b^2 - 4*a*c)/(2*a) else -b/(2*a);
  roots[1] := r1;
  roots[2] := i1;
  roots[3] := r2;
  roots[4] := i2;
end quadroots;

```

A5.2 Complex_Mult

```

function complex_mult
//Author: Robert McBride
//Calculate (r1+i1*i)*(r2+i2*i)
  input Real r1;
  input Real i1;
  input Real r2;
  input Real i2;
  output Real m1[2] "Real((r1+i1*i)*(r2+i2*i)) Imag((r1+i1*i)*(r2+i2*i))";
algorithm
  m1[1] := r1*r2 - i1*i2;//Real part of the multiplication
  m1[2] := r1*i2 + r2*i1;//Imag part of the multiplication
end complex_mult;

```

A5.3 Complex_Div

```

function complex_div
//Author: Robert McBride
//Calculate (r1+i1*i)/(r2+i2*i)
  input Real r1;

```

```
input Real i1;
input Real r2;
input Real i2;
output Real m1[2] "Real((r1+i1*i)/(r2+i2*i)) Imag((r1+i1*i)/(r2+i2*i))";
protected
Real alpha;
Real beta;
Real gamma;
Real delta;
Real temp_num[2];
Real temp_den;
algorithm
alpha := r1;
beta := i1;
gamma := r2;
delta := i2;
//Multiply num and den by conj(den)
temp_num := complex_mult(alpha, beta, gamma, -delta);
temp_den := gamma^2 + delta^2;

//Perform division
m1[1] := temp_num[1]/temp_den;
m1[2] := temp_num[2]/temp_den;

end complex_div;
```

APPENDIX B1: Symmetry of Hamiltonian Eigenvalues

B1.1 Eigenvalue Symmetry about the Real Axis

The eigenvalues of any real valued matrix are symmetric about the real axis of the complex plain. To see this consider a real valued matrix A with an eigenvalue λ and an eigenvector V . Thus

$$AV = \lambda V \quad (\text{B1.1.1})$$

Taking the conjugate (without transpose) of both sides

$$\text{conj}(AV) = \text{conj}(\lambda V) \quad (\text{B1.1.2})$$

which can be written

$$\text{conj}(A) * \text{conj}(V) = \text{conj}(\lambda) * \text{conj}(V) \quad (\text{B1.1.3})$$

Since A is real valued B1.1.3 can be written

$$A * \text{conj}(V) = \text{conj}(\lambda) * \text{conj}(V) \quad (\text{B1.1.4})$$

Thus, if λ is an eigenvalue then $\text{conj}(\lambda)$ must also be an eigenvalue, QED.

For real valued λ , $\text{conj}(\lambda) = \lambda$. No new information is obtained. For complex λ_1 , an eigenvalue and eigenvector can be found by $\lambda_2 = \text{conj}(\lambda_1)$, and $V_2 = \text{conj}(V_1)$ [Cur84, Kai80, Zho96].

B1.2 Hamiltonian Eigenvalue Symmetry about the Imaginary Axis

The algebraic Riccati equation (ARE) is

$$AX + XA + XRX + Q = 0 \quad (\text{B1.2.1})$$

where A , Q , and R are real $n \times n$ matrices with Q and R symmetric. The Hamiltonian matrix associated with the ARE is

$$H := \begin{bmatrix} A & R \\ -Q & -A^* \end{bmatrix} \quad (\text{B1.2.2})$$

Introducing an $n \times n$ transformation matrix J with the property $J^2 = -I$

$$J := \begin{bmatrix} 0 & -I \\ I & 0 \end{bmatrix} \quad (\text{B1.2.3})$$

Note $J^{-1} = -J$. Use this matrix to transform H in the following fashion

$$J^{-1}HJ = -JHJ = -H^* \quad (\text{B1.2.4})$$

herefore H is similar to $-H^*$. Thus, if λ is an eigenvalue, then $-\text{conj}(\lambda)$ is also an eigenvalue, QED [Zho96 pp.327-328].

Note that the eigenvalues of H are symmetric about both axes. This implies that the characteristic polynomial of H does not contain odd powers of S . This insight helps in finding the eigenvalues of H in that the characteristic polynomial can be broken down into n , second-order polynomials with real coefficients.

APPENDIX B2: Vandermonde Representation of Controller Canonical Eigenvectors

Given a matrix A that can be transformed into controller canonical form \tilde{A} , with a transformation matrix T , such that

$$\tilde{A} = TAT^{-1} \quad (\text{B2.1})$$

where

$$\tilde{A} = \begin{bmatrix} 0 & 1 & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & 0 & 1 & \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \cdot & \cdot & \cdot & 1 \\ a_0 & a_1 & a_2 & \cdot & \cdot & \cdot & a_{n-1} \end{bmatrix} \quad (\text{B2.2})$$

Let λ_i be an eigenvalue of A . Then there exists an eigenvector Vh_i such that

$$\tilde{A}Vh_i = Vh_i\lambda_i \quad (\text{B2.3})$$

Let $Vh_i(1) = 1$, since the eigenvalue/eigenvector problem is over-determined. Thus, from B2.2 and B2.3

$$Vh_i = \begin{bmatrix} 1 \\ \lambda_i \\ \lambda_i^2 \\ \cdot \\ \cdot \\ \cdot \\ \lambda_i^{n-1} \end{bmatrix} \quad (\text{B2.4})$$

The complete eigenvalue/eigenvector problem becomes

$$\tilde{A}Vh = Vh\Lambda \quad (\text{B2.5})$$

where Λ is a diagonal matrix made up of the eigenvalues of A , and Vh is a Vandermonde matrix formed from the eigenvalues as shown by equation B2.3, QED [Kai80 pp. 54-55].

Since equation B2.1 can be written

$$A = T^{-1}\tilde{A}T \quad (\text{B2.6})$$

and B2.5 shows

$$\tilde{A} = Vh\Lambda Vh^{-1} \quad (\text{B2.7})$$

Substituting B2.7 into B2.6 gives

$$A = T^{-1}Vh\Lambda Vh^{-1}T \quad (\text{B2.8})$$

Therefore $T^{-1}Vh$ form the eigenvector matrix of A .

APPENDIX C: Glossary of Terms

2DoF	Two Degree of Freedom
6DoF	Six Degree of Freedom
ARE	Algebraic Riccati Equation
cg	Center of Gravity
LQR	Linear Quadratic Regulator
NC1	Nonlinear Controller 1
NC2	Nonlinear Controller 2
PID	Proportional Integral Derivative
PWM	Pulse Width Modulation
SDRE	State Dependent Riccati Equation

REFERENCES

- Bej97 Bejan, A., *Advanced Engineering Thermodynamics*. 2nd Edition, 1997, John Wiley and Sons, Inc.
- Bro01 Brown, F. T., *Engineering System Dynamics*, 2001, Marcel Dekker, Inc.
- Bro72 Brown, F. T., "Lagrangian Bond Graphs," *Journal of Dynamic Systems, Measurement, and Control*, Trans. ASME, September 1972, pp. 213-221.
- Brü02 Brück, D., Elmqvist, H., Mattsson, S. E., and Olsson, H., "Dymola for Multi-Engineering Modeling and Simulation," *Proceedings 2nd International Modelica Conference*, Munich, Germany, 2002, pp. 55:1-9, http://www.modelica.org/Conference2002/papers/p07_Brueck.pdf.
- Cel03a Cellier, F. E., McBride, R. T., "Object-Oriented Modeling of Complex Physical Systems Using the Dymola Bond-Graph Library," *Proceedings International Conference on Bond Graph Modeling*, Orlando, Florida, 2003, pp. 157-162.
- Cel03b Cellier, F. E., Greifeneder, J., "Object-Oriented Modeling of Convective Flows using the Dymola Thermo-Bond Graph Library," *Proceedings International Conference on Bond Graph Modeling*, Orlando, Florida, 2003, pp. 198-204.
- Cel93 Cellier, F. E., Elmqvist, H., "Automated Formula Manipulation Supports Object-Oriented Continuous-System Modeling," *IEEE Control Systems*, 1993, 13(2), pp. 28-38.
- Cel91 Cellier, F. E., *Continuous System Modeling*. 1991, New York: Springer-Verlag.
- Cen89 Çengel, Y. A., Boles, M. A., *Thermodynamics, an Engineering Approach*. 1989, McGraw Hill, Inc.
- Clo96a Cloutier, J. R., Mracek, C. P., "Missile Longitudinal Autopilot Design Using the State-Dependent Riccati Equation Method," *Proceedings Nonlinear Problems in Aviation and Aerospace*, Florida, 1996, pp. 143-176.
- Clo96b Cloutier, J. R., D'Souza, C. N., Mracek, C. P., "Nonlinear Regulation and Nonlinear H_∞ Control Via the State-Dependent Riccati Equation Technique," *Proceedings Nonlinear Problems in Aviation and Aerospace*, Florida, 1996, pp. 117-142.

- Cur84 Curtis, C. W., *Linear Algebra, an Introductory Approach*, 1984, New York: Springer Verlag.
- Dym DYMOLA Dynamic Modeling Laboratory, World Wide Web page <http://www.Dynasim.se>.
- Elm94 Elmqvist, H., Otter, M., "Methods for Tearing Systems of Equations in Object-Oriented Modeling," *Proceedings European Simulation Multiconference*, Barcelona, Spain, 1994, pp. 326-332.
- Fah99 Fahrenthold, E. P., Koo, J. C., "Discrete Hamilton's Equations for Viscous Compressible Fluid Dynamics," *Computer Methods in Applied Mechanics and Engineering*, 1999, 178(1-2), pp. 1-22
- Fah94 Fahrenthold, E. P., Wargo, J. D., "Lagrangian Bond Graphs for Solid Continuum Dynamics Modeling," *Journal of Dynamic Systems, Measurement and Control*, Trans. ASME, June 1994, pp. 178-192.
- Fav98 Favre, W., Scavarda, S., "Bond Graph Representation of Multibody Systems with Kinematic Loops," *Journal of the Franklin Institute of Engineering and Applied Mathematics*, 335B (4), May 1998, pp. 643-660.
- Fra98 Franklin, G. F., Powell, J. D., Workman, M., *Digital Control of Dynamic Systems*. 3rd Edition, 1998, Addison-Wesley Longman, Inc.
- Fra94 Franklin, G. F., Powell, J. D., Emami-Naeini, A., *Feedback Control of Dynamic Systems*, 1994, Addison-Wesley Publishing, Inc.
- Gre01a Greifeneder, J., Cellier, F. E., "Modeling Convective Flows using Bond Graphs," *Proceedings International Conference on Bond Graph Modeling*, Phoenix, Arizona, 2001, pp. 276-284.
- Gre01b Greifeneder, J., Cellier, F. E., "Modeling Multi-Phase Systems using Bond Graphs," *Proceedings International Conference on Bond Graph Modeling*, Phoenix, Arizona, 2001, pp. 285-291.
- Jun05 Junco, S., Donaire, A., "BG-Supported Synthesis – and Position – Tracking Controllers for Brushless DC-Motor Drives," *Proceedings International Conference on Bond Graph Modeling*, New Orleans, Louisiana, 2005, pp. 245-251.

- Kai80 Kailath, T., *Linear Systems*. 1980, New Jersey: Prentice Hall, Inc.
- Kan80 Kane, T. R., Levinson, D. A., "Formulation of Equations of Motion for Complex Spacecraft," *Journal of Guidance and Control*, 1980, 3(2), pp. 99-112
- Kar90 Karnopp, D. C., Margolis, D. L., Rosenberg, R. C., *System Dynamics: A Unified Approach*. 1990, John Wiley & Sons, Inc.
- Kar83 Karnopp, D. C., Rosenberg, D. L., *Introduction to Physical System Dynamics*. 1983, McGraw-Hill, Inc.
- Kar77 Karnopp, D. C., "Lagranges Equations for Complex Bond Graph Systems," *Journal of Dynamic Systems, Measurement, and Control*, Trans. ASME, December 1977, pp. 300-306.
- Kar69 Karnopp, D. C., "Power-Conserving Transformation: Physical Interpretations and Applications using Bond Graphs," *Journal of the Franklin Institute*, 1969, 288(3), pp. 175-201.
- Kir98 Kirk, D. E., *Optimal Control Theory, An Introduction*, 1998, New York: Dover Publications, Inc.
- Lag1788 Lagrange, J. L., "Mechanique Analytique". Paris, 1788.
- Lam91 Lambert, J. D., *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*, 1991, New York: John Wiley.
- Liu02 Liu, Z. Y., Wagner, J., "Nonlinear Model Reduction for Dynamic and Automotive System Descriptions," *Journal of Dynamic Systems Measurement and Control*, Trans. ASME, December 2002, pp. 637-647.
- Lou02 Louca, L. S., Stein, J. L., "Ideal Physical Element Representation from Reduced Bond Graphs," *Proceedings of the Institution of Mechanical Engineers, Part I-Journal of Systems and Control Engineering*, 2002, 216(I1), pp. 73-83.
- Lou99 Louca, L. S., Stein, J. L., "Energy-Based Model Reduction of Linear Systems," *Proceedings International Conference on Bond Graph Modeling*, San Francisco, California, 1999.
- Map Maple, Maplesoft, World Wide Web page <http://www.maplesoft.com>.

- Mas91 Maschke, B., "Geometrical Formulation of Bond Graph Dynamics with Applications to Mechanisms," *Journal of the Franklin Institute of Engineering and Applied Mathematics*, 1991, 328(5-6), pp. 723-740.
- Mat MATLAB, The Language of Technical Computing, World Wide Web page <http://www.mathworks.com>.
- McB05a McBride, R. T., Cellier, F. E., "Optimal Controller Gain Selection Using the Power Flow Information of Bond Graph Modeling," *Proceedings International Conference on Bond Graph Modeling*, New Orleans, Louisiana, 2005, pp. 228-232.
- McB05b McBride, R. T., *Quality Metric for Controller Design*, Internal Raytheon Document, Raytheon Missile Systems, Tucson AZ 85734, 2005.
- McB05c McBride, R. T., Cellier, F. E., "System Efficiency Measurement through Bond Graph Modeling," *Proceedings International Conference on Bond Graph Modeling*, New Orleans, Louisiana, 2005. pp. 221-227.
- McB03 McBride, R. T., Cellier, F. E., "Object-Oriented Bond-Graph Modeling of a Gyroscopically Stabilized Camera Platform," *Proceedings International Conference on Bond Graph Modeling*, Orlando, Florida, 2003, pp. 157-223.
- McB01 McBride, R. T., Cellier, F. E., "A Bond Graph Representation of a Two-Gimbal Gyroscope," *Proceedings International Conference on Bond Graph Modeling*, Phoenix, Arizona, 2001, pp. 305-312.
- Mei98 Meirovitch, L., *Methods of Analytical Dynamics*, 1998, New York: Dover Publications Inc.
- Mon91 Montbrundifilippo, J., Delgado, M., Brie, C., Paynter, H. M., "A Survey of Bond Graphs – Theory, Applications and Programs," *Journal of the Franklin Institute of Engineering and Applied Mathematics*, 1991, 328(5-6), pp. 565-606.
- Mra05 Mracek, C. P., Ridgely, D. B., "Missile Longitudinal Autopilots: Connections Between Optimal Control and Classical Topologies," Raytheon Missile Systems, Tucson AZ 85734, April 2005, to be published at *AIAA Guidance and Control Conference*, San Francisco, California, Aug. 2005.

- Muk05 Mukherjee, A., Vikas, R., Dasgupta, A., "A Procedure for Finding Invariants of Motions for General Class of Unsymmetric Systems with Gauge-variant Umbra Lagrangian Generated by Bond Graphs," *Proceedings International Conference on Bond Graph Modeling*, New Orleans, Louisiana, 2005, pp. 11-16.
- Muk97 Mukherjee, A., Samantaray, A. K., "Umbra-Lagranges Equations through Bond Graphs," *Proceedings International Conference on Bond Graph Modeling*, Phoenix, Arizona, 1997, pp. 168-174.
- Pan88 Pantelides, C. C., "The Consistent Initialization of Differential-Algebraic Systems," *SIAM Journal of Scientific Statistical Computation*, 1988, 9(2), pp. 213-231.
- Rei93 Reichert, R. T., Nichols, R., Rugh, W., *Gain Scheduling for H_{∞} Controllers: A Flight Control Example*, Internal JHU Document, The John Hopkins University JHU/ECE-92/03, Baltimore Maryland 21218, 1993.
- Sar04 Samantaray, A. K., Medjaher, K., Bouamama, B. O., Staroswiecki, M., Dauphin-Tanguy, G., "Component-Based Modelling of Thermofluid Systems for Sensor Placement and Fault Detection," *Simulation-Transactions of the Society for Modeling and Simulation International*, 2004, 80(7-8), pp. 381-398.
- Tar72 Tarjan, R. E., "Depth First Search and Linear Graph Algorithms," *SIAM Journal on Computing*, 1972, 1, pp.146-160.
- Ther ThermoAnalytics, Inc., World Wide Web page
<http://www.thermoanalytics.com/support/publications/batterymodelsdoc.html>
- Tve98 Tvester F., "Deriving the Hamilton Equations of Motion for a Non-conservative System using a Variational Principle," *Journal of Mathematical Physics*, 1998, 39, pp. 1495-1500.
- Urq03a Urquia A., Dormido, S., "Object-Oriented Description of Hybrid Dynamic Systems of Variable Structure," *Simulation Trans. of the Society for Modeling and Simulation International*, 2003, 79(9), pp. 485-493.
- Urq03b Urquia A., Dormido, S., "Object-Oriented Design of Reusable Libraries of Hybrid Dynamic Systems – Part One: A Design Methodology," *Mathematical and Computer Modeling of Dynamical Systems*, 2003, 9(1), pp. 65-90.
- War95 Wark, K. *Advanced Thermodynamics for Engineers*, 1995, McGraw-Hill, Inc.

- Zar02 Zarchan, P., *Tactical and Strategic Missile Guidance, Fourth Edition*.
Virginia: Vol. 199, 2002, Progress in Astronautics and Aeronautics, American
Institute of Aeronautics and Astronautics, Inc.
- Zei95 Zeid A. A., Overholt, J. L., "Singularly Perturbed Bond Graph Models for
Simulation of Multibody Systems," *Journal of Dynamic Systems
Measurement and Control*, Trans. ASME, Sep 1995, pp. 401-410.
- Zho96 Zhou, K., Doyle, J. C., Glover, K., *Robust and Optimal Control*, 1996, New
Jersey: Prentice Hall, Inc.