Copyright
by
John David Hedengren
2005

The Dissertation Committee for John David Hedengren
certifies that this is the approved version of the following dissertation:

REAL-TIME ESTIMATION AND CONTROL OF LARGE-SCALE NONLINEAR DAE SYSTEMS

Committee:

Thomas F. Edgar, Supervisor

R. Bruce Eldridge

Stephen B. Pope

S. Joe Qin

James B. Rawlings

REAL-TIME ESTIMATION AND CONTROL OF LARGE-SCALE NONLINEAR DAE SYSTEMS

by

John David Hedengren, B.S., M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of
The University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN
May 2005

For Sarah.

Acknowledgments

I wish to thank my advisor, Professor Thomas F. Edgar, for his insightful comments, on-demand support, and responsiveness. It has been a pleasure to work with him. I also wish to thank the other members of my committee, Prof. Eldridge, Prof. Pope, Prof. Rawlings, Prof. Qin, for their guidance and thought-provoking comments. Several of the original ideas in this dissertation were developed as a result of stimulating discussions with my committee members.

Many others have also contributed to this work. Sebastien Lextrait, Lina Rueda, Scott and Chris Harrison, Elaine Hale, Terry Farmer, Eric Haseltine, and others in the Edgar, Qin, and Rawlings research groups have given thoughtful suggestions. Don Bartusiak, Bob Fontaine, and Tyler Soderstrom contributed to my understanding of current challenges in control during an internship with ExxonMobil. Lastly, I wish to recongize my wife, Sarah, for her enabling support.

REAL-TIME ESTIMATION AND CONTROL OF LARGE-SCALE NONLINEAR DAE SYSTEMS

Publication No. ___________________

John David Hedengren, Ph.D.
The University of Texas at Austin, 2005

Supervisor: Thomas F. Edgar

Model-based control incorporates fundamental process knowledge to achieve improved monitoring and control performance. However, on-line model-based control is generally limited to linear models or nonlinear models of low-dimension. Rigorous models of dynamic process are often described by differential algebraic equations (DAEs). Many rigorous DAE models require too much computational effort to be implemented in real-time control applications, where control calculations must be performed on-line (i.e. in a few seconds). The principal focus of this dissertation is to reduce the computational requirements for large-scale model-based estimation and control. This objective is accomplished with a variety of strategies that are combined in an effective way to meet real-time constraints with limited computing resources. The principal strategies are adaptive storage and retrieval off-line to enable efficient on-line control, nonlinear DAE model reduction, and development of an explicit solution to moving horizon estimation (MHE). Both MHE and receeding horizon control (RHC) are developed to meet real-time constraints. In situ adaptive tabulation (ISAT) is used to store and retrieve control solutions. In addition to the adaptation for control applications, ISAT is developed as a general nonlinear function approximator and is shown to outperform neural networks in both interpolation and extrapolation. In addition, ISAT is designed to handle nonlinear functions with discontinuities or regions that are not continuously differentiable. With DAE model reduction, storage and retrieval of control solutions with ISAT, and the explicit solution to moving horizon estimation, real-time nonlinear model predictive control (NMPC) is feasible with large-scale DAE models.

Table of Contents

Acknowledgments **v**

Abstract **vi**

List of Tables **xiii**

List of Figures **xiv**

Nomenclature **xxvii**

Chapter 1. Introduction **1**

- 1.1 DAE Model Reduction 3
- 1.2 Storage and Retrieval 4
  - 1.2.1 *In Situ* Adaptive Tabulation 7
- 1.3 Research Objectives 8
- 1.4 Overview of this Dissertation 9

Chapter 2. The ISAT Algorithm **12**

- 2.1 Review of ISAT for Turbulent Combustion Simulations 15
- 2.2 Details of the ISAT Algorithm 18
  - 2.2.1 The ISAT Record 19
  - 2.2.2 Searching the Records with Binary Trees 20
  - 2.2.3 Record Retrieval 22
  - 2.2.4 Record Growth 22
    - 2.2.4.1 Definition of the EOA and growth point 23
    - 2.2.4.2 Transform the coordinates to map the EOA to a unit hypersphere 23
    - 2.2.4.3 Map the growth point to the transformed coordinates 25

2.2.4.4 Align one of the y-axes with the direction of the growth point 26
2.2.4.5 Grow the hypersphere into an ellipsoid that reaches the growth point 28
2.2.4.6 Transform the expanded ellipsoid back to the original coordinate system 30
2.2.5 Record Addition 30
2.2.5.1 Initial states and inputs 31
2.2.5.2 Final states 31
2.2.5.3 Sensitivity 32
2.2.5.4 Initial estimate of the EOA 35
2.2.5.5 Binary Tree Addition 36
2.3 Scaling to Large Scale Problems 37
2.4 Example: Comparison of ISAT and Neural Networks 37
2.5 Summary and Conclusions 42

Chapter 3. DAE Model Reduction **44**

3.1 Previous Work 45
3.1.1 Reduction of Differential Equations 46
3.1.1.1 Proper Orthogonal Decomposition 47
3.1.1.2 Balanced Covariance Matrices 49
3.1.1.3 Example Comparison of POD, BCM, and Linearization 51
3.1.2 Reduction of Algebraic Equations 54
3.1.2.1 Pairing Variables to Equations 54
3.1.2.2 Application to Large Scale Problems 55
3.1.2.3 Example: Flowsheet Model Reduction 55
3.1.2.4 Explicit Transformation of Implicit Sets 57
3.2 Adaptive Reduction of DAEs 57
3.2.1 Step 1: Reduction of ODEs 59
3.2.1.1 Predicting Variable Error 59
3.2.1.2 Correcting Variable Error 61
3.2.1.3 Controlling Variable Error 62
3.2.2 Step 2: Partitioning and Precedence Ordering 62

3.2.2.1 Example: Binary Distillation 64
3.2.3 Step 3: Explicit Transformation of Algebraic/Differential Equations 68
3.3 PDE Example: Unsteady Heat Conduction 69
3.4 Index-2 DAE Example: Large-Scale Distillation Model 72
3.5 Summary and Conclusions 79

Chapter 4. Storage and Retrieval in Nonlinear Model Predictive Control 81
4.1 Direct Methods for Solving NMPC Problems 83
4.1.1 Hybrid NMPC 84
4.1.1.1 Estimator 85
4.1.1.2 Regulator 86
4.1.2 Sequential NMPC 87
4.1.2.1 Estimator 87
4.1.2.2 Regulator 88
4.2 Example: Sequential NMPC 89
4.3 Example: ISAT vs. Neural Networks in Control Calculations . 93
4.4 Parameter Sensitivity Studies 97
4.5 Summary and Conclusions 98

Chapter 5. Nonlinear Model Predictive Control - The Explicit Solution 99
5.1 Explicit MPC (Linear Models) 100
5.1.1 Parameterization of Initial States 101
5.1.2 Parameterization of Active Sets 102
5.2 Explicit NMPC (Nonlinear Models) 104
5.2.1 Dynamic Programming 104
5.2.2 Artificial Neural Networks 105
5.2.3 Multiparametric NMPC 105
5.3 Approximate Nonlinear MPC 106
5.3.1 NMPC Formulation 107
5.3.2 Sensitivity of Optimal Control to Parameters 109
5.3.3 Defining the Critical Region 111

5.3.4 ISAT Approximate Control 112
5.3.4.1 First Scenario: Retrieval 113
5.3.4.2 Second Scenario: EOA Growth 114
5.3.4.3 Third Scenario: Addition 114
5.3.5 Summary of the ISAT Algorithm 115
5.4 Temperature Control of an Exothermic CSTR 117
5.5 Summary and Conclusions 123

Chapter 6. Moving Horizon Estimation - The Explicit Solution 125
6.1 Previous Work 125
6.2 Moving Horizon Estimation Problem Formulation 128
6.3 The Explicit MHE Solution 130
6.3.1 Fully observable systems 132
6.3.2 Partially observable systems 133
6.3.3 Example 1: Explicit versus Implicit MHE Solution . . . 134
6.4 Inequality Constraints in Explicit MHE 136
6.4.1 Example 2: Constrained Version of Example 1 139
6.5 Example 3: Flash Column Composition Estimation 146
6.5.1 Explicit MHE Scaling with Model Size 150
6.5.2 Explicit MHE Scaling to Long Time Horizons 151
6.6 Example 4: Two State CSTR 151
6.7 Conclusions 154

Chapter 7. Conclusions and Recommendations 156
7.1 Summary of Contributions 156
7.1.1 Development of a Superior Alternative to Neural Networks for Nonlinear Function Approximation 156
7.1.2 Automatic, Iterative DAE Model Reduction 157
7.1.3 Reduced Computational Effort for Nonlinear MPC . . . 157
7.1.4 Development of Explicit Nonlinear MPC 158
7.1.5 Development of Explicit Moving Horizon Estimation . . 158
7.2 Future Work 159
7.2.1 Model Reduction for the Non-Expert User 159

7.2.2 Model Reduction for Sensitivity Analysis 159
7.2.3 Model Reduction with the Open Equation Format . . . 160
7.2.4 Higher Order Piecewise Approximations in ISAT 160
7.2.5 Parameter Estimation 161

Appendices **162**

Appendix A. Binary Distillation Column Model **163**

Appendix B. Dual CSTR Model **166**

Appendix C. Flowsheet Model **169**

Bibliography **176**

Vita **190**

List of Tables

2.1 Elements of the ISAT record, along with the vector and matrix dimensions 20

3.1 Variables at each stage of the distillation column 65

4.1 Comparison of Direct Methods for NMPC 84

5.1 Elements of the ISAT record for NMPC storage and retrieval . 113
5.2 Elements of the ISAT record for the CSTR example 120

A.1 Variables 164
A.2 Equations 165

B.1 Variables 167
B.2 Equations 168

C.1 Feed Variables 170
C.2 Flash Column Variables 171
C.3 Heat Exchanger Variables 172
C.4 Splitter Variables 173
C.5 Mixer and Tank Variables 174
C.6 Partitioning and Precedence Ordering Implicit Block 175

List of Figures

1.1 Block diagram of a deterministic calculation of f based on independent variables s. The block diagram may represent open loop simulation or a simplification of closed loop control. . . . 4

1.2 ISAT stores solutions and sensitivities (A) to approximate f with multidimensional piecewise linear regions. 7

2.1 Block diagram of ISAT interaction with the reacting flow simulation. ISAT stores and retrieves the thermochemical properties involved in chemical reactions. Because the chemistry is decoupled from the other aspects of the simulation, the final chemical compositions are a unique function of the initial concentrations and time. 17

2.2 ISAT was originally designed to store and retrieve numerical integrations of chemistry evolution in turbulent combustion simulations. 19

2.3 Each node of a binary tree can either be a leaf or branch. The leaves of the binary tree are individual records of the ISAT database. A branch, on the other hand, points to two other nodes. All branches divide until a leaf terminates the line. . . 21

2.4 The center point of the ellipse is the origin. The growth point s_q becomes x_q after the translation and the EOA is defined in terms of x. 24

2.5 In a two dimensional example, the y-axes are shown relative to the x-axes. In the y-axes reference frame, the ellipse becomes a unit circle centered at the origin. 26

2.6 The growth point is translated to the new y reference frame. The magnitude is the distance between y_q and the origin. The normalized vector y_n has a unit length and points in the direction of y_q. 27

2.7 The axes are rotated so that one axis aligns with the growth point. This rotation is important so that the ellipse can be expanded along the aligned axis. 28

2.8 The circle is expanded into an ellipse that reaches the growth point. This is a minimum area expansion of a symmetric ellipse. 29

2.9 When the ellipse is transformed back to the original coordinates, x_q is on the ellipse perimeter. In addition, the ellipse is a symmetric minimum area expansion that includes the growth point and the original ellipse. 31

2.10 The binary tree is grown to include a new record. The growth creates a new node where the next closest record previously appeared. 36

2.11 The first eigenfunction of an L-shaped membrane used to compare ISAT and neural networks. The second and third eigenfunctions have also been shown in MathWorks' publications. . 38

2.12 ISAT approximation with an error tolerance of 0.5. Due to the high error tolerance, the approximation is very coarse with 12 linear regions. 39

2.13 ISAT approximation with an error tolerance of 0.1. The approximation is more refined with 48 linear regions. 40

2.14 ISAT approximation with an error tolerance of 0.01. The approximation includes 206 linear regions and ISAT closely approximates the original eigenfunction. 41

2.15 Neural network approximation to the eigenfunction. 41

3.1 A step decrease in reflux ratio produces a corresponding decrease in the distillate composition. A nonlinear 32 state model, a 3 state BCM reduced model, and a 3 state POD model are shown. 52

3.2 The same dynamic response as Figure 3.1 is shown in this plot. Here a nonlinear 32 state model, 1 state BCM and POD models, and 32 state linearized model are compared. 53

3.3 Flowsheet model involving equal molar feed streams of butane, pentane, hexane, heptane, and octane at 300 K and 1 ATM. The model has 229 variables with 12 ODEs and 217 AEs. . . . 56

3.4 5% Step Change in Reboiler Duty 68

3.5 Graphical representation of a 1-D simulation of heat transfer in a 1.0 m thich aluminum slab. The PDE is spatially discretized to a set of ODEs, one for every interior node. 69

3.6 Full 20 state model compared with reduced order models of 3, 2, and 1 states using POD. 70

3.7 Equation residuals have a strong correlation to the variable residuals. The singular values also decrease with an increase in model order, but have limited ability to predict variable error. 71

3.8 The predicted and actual variable errors are shown. The prediction is excellent for this example because the model is near steady state and nearly linear. 72

3.9 Using the equation residuals to predict variable error, the reduced models show improved accuracy. In this case, the 1 state reduced order model is sufficiently accurate. Without the correction, a 3 state model is required for similar accuracy (see Figure 3.6). 73

3.10 The multi-component distillation column model consists of 22 trays, multiple sidedraws, a condenser, a reboiler, and a sump at the column base. 74

3.11 The distillation column model is constructed from an object-oriented simulator. More advanced models are simply combinations of base level models connected by streams. 75

3.12 The reduced models are compared with the 353 state distillation model. 77

4.1 Model and computational reduction flowchart. 89

4.2 Closed loop response comparison for nonlinear MPC with ISAT with 5 states, nonlinear MPC with 5 states, nonlinear MPC with 32 states, and linear MPC. 92

4.3 Speed-up factor for each of the optimizations shown in Figure 4.2. The number above each curve indicates the average optimization cpu time on a 2 GHz processor. 92

4.4 Diagram of two CSTRs in series with a first order reaction. The manipulated variable is the heating rate to the first CSTR. . . 93

4.5 Neural net with one hidden layer and one output layer. The hidden layer is a hyperbolic tangent function and the output layer is a linear function. This neural net relates 7 inputs to 6 outputs. 94

4.6 Open loop step test for the dual CSTR model. The error control of ISAT adds records to the database when extrapolating outside of the training domain. 95

4.7 Small closed loop set point change within the training domain. 95

4.8 Large closed loop set point change outside of the training domain. 96

5.1 A retrieval occurs when the query point (ϕ_{q1}) is within the ellipsoid of accuracy (EOA) 114

5.2 The EOA is grown when the query point is outside the EOA but within the error tolerance for u_0^* 115

5.3 A record is added to the ISAT database when query point is outside the EOA and error tolerance for u_0^* 116

5.4 Diagram of the exothermic CSTR. The two state variables reactor concentration c_A and temperature T are controlled by the jacket cooling temperature T_c 117

5.5 Unsteady response of the reactor temperature due to moments of intense reaction followed by periods of gradual cooling. . . . 118

5.6 The optimal jacket temperature (T_c^*) is a unique function of reactor concentration (c_A), reactor temperature (T), and reactor temperature set point (T_{sp}). In this figure, the set point is fixed ($T_{sp} = 320K$) and c_A and T are varied. 120

5.7 Control performance of ISAT compared to NMPC. The controlled variable (CV) is the reactor temperature, the state variable (SV) is the reactor concentration, and the manipulated variable (MV) is the temperature of the cooling jacket. 122

5.8 Computational times of the estimator and regulator at each sampling instant. ISAT is fast because computation is limited to a matrix multiplication. 123

6.1 The explicit and implicit MHE solutions produce the same results. Substantial computational reduction is obtained with the explicit solution approach. 135

6.2 Variable x_1 is constrained between 0 and 0.2. The unconstrained solution violates the upper bound constraint on x_1. By incorporating the constraints the solution is improved. . . 139

6.3 Results for constrained explicit MHE (CE MHE) compared to optimization based MHE (labeled as MHE). Note that x_1 now meets the constraint condition. 140

6.4 First iteration of the explicit MHE solution. State x_1 violates the upper bound constraint of 0.2. 141

6.5 Second iteration for explicit MHE. Weighting is added to matrix Q_b where the x_1 trajectory formerly violated the upper bound constraint of 0.2. The constraint information is indicated as fictitious measurements even though x_1 is not actually measured. 142

6.6 Third iteration of explicit MHE. The explicit MHE solution is approaching the implicit MHE solution as the active set of constraints is refined. 143

6.7 Fourth iteration of explicit MHE. The explicit MHE solution almost agrees exactly with the implicit MHE solution. The final solution requires one more iteration for convergence. . . . 144

6.8 Final iteration of explicit MHE. In all, the explicit solution sequence required 5 iterations for convergence. 145

6.9 Flow sheet diagram of the flash column model. The flash column consists of a feed tank with unknown species compositions, a flash column, and vapor and liquid outlet streams. 146

6.10 The estimated states converge quickly to the real system even though the initial guess is poor. 147

6.11 Estimated compositions of C_4H_{10}, C_5H_{12}, and C_6H_{14} approach the actual values of 0.2. The other two compositions of C_7H_{16} and C_8H_{18} deviate significantly because the system is not fully observable. 148

6.12 The composition estimation is greatly improved by adding an inequality constraint to C_7H_{16}. Even though the system is not fully observable, the composition estimates closely approximate the actual values. 149

6.13 Explicit MHE scaling to large-scale model size. Both the nonlinear and linear approaches scale $O(x^2)$ in the number of floating point operations, where x is the number of variables in the model. 150

6.14 Explicit MHE scaling to horizon length. For nonlinear models, scaling is $O(x^2)$ in the number of floating point operations. For linear models, scaling is $O(x)$ where x is the horizon length. . 152

6.15 Estimation performance of the explicit solution MHE (eMHE) versus MHE. The state variable (SV) estimation is difficult to distinguish on the graph because the predictions are virtually identical for the two approaches. The only difference is the substantially lower computational effort required to reach a solution. 153

A.1 Diagram of a dynamic binary distillation column model with equilibrium stages 163

B.1 This model is a dual CSTR with an exothermic first-order reaction. It is the same model as the one used by Hahn [34], but with some minor modifications. 166

C.1 Diagram of the flowsheet model involving equal molar feed streams of butane, pentane, hexane, heptane, and octane at 300 K and 1 ATM. The model has 229 variables with 12 ODEs and 217 AEs. 169

Nomenclature

BCM	balanced covariance matrices
CE MHE	constrained explicit moving horizon estimation
CFD	computational fluid dynamics
CPU	central processing unit
CR	critical region
CSP	computational singular perturbation
CSTR	continuously stirred tank reactor
CV	controlled variable
DAE	differential algebraic equation
DASAC	differential algebraic sensitivity analysis code
DASPK	differential algebraic solver package
DDOF	dynamic degrees of freedom
DNS	direct numerical simulation
DOF	degrees of freedom
EOA	ellipsoid of accuracy
FORTRAN	formula translation

IC	integrated circuit
ILDM	intrinsic low-dimensional manifolds
ISAT	in situ adaptive tabulation
ISAT-RCCE	in situ adaptive tabulation - rate-controlled constrained equilibrium
IVP	initial value problem
LP	linear programming
LQR	linear quadratic regulator
MB	megabyte
MFLOPS	million floating point operations per second
MHE	moving horizon estimation
MILP	mixed integer linear programming
MIMO	multiple inputs multiple outputs
MINLP	mixed-integer nonlinear programming
mp-LP	multi-parametric linear program
mp-NLP	multi-parametric nonlinear programming
mp-QP	multi-parametric quadratic program
MPC	model predictive control (linear model and constraints)
MV	manipulated variable
NLP	nonlinear programming

NMPC	nonlinear model predictive control
ODE	ordinary differential equation
ODESSA	ordinary differential equation solver with explicit simultaneous sensitivity analysis
PID	proportional integral derivative
POD	proper orthogonal decomposition
PWA	piecewise affice
QP	quadratic programming
ROA	region of accuracy
ROM	reduced order model
SISO	single input single output
SQP	sequential quadratic programming
SV	state variable
NDDR	nonlinear dynamic data reconciliation
Δf	differences between f_q and nearby f vectors
Δs	differences between s_q and nearby s vectors
ϵ_{resid}	linear regression residuals
ϵ_{tol}	error tolerance
η	state constraint violation
λ	inequality constraint multiplier

$\mathbb{R}$	set of real numbers
ν	binary tree element
ν	equality constraint multiplier
Φ	cost function
ϕ	set of problem dependent parameters
ϕ_q	set of problem dependent parameters (query)
ρ	density
Σ_i	diagonal matrix of eigenvalues in i coordinates ($i = x$, y, z, or R)
σ_i	i^{th} eigenvalue of M
τ	number of non-zeros in the sparsity matrix
θ	adjustable parameters
$\tilde{A}$	modified sensitivity matrix
A	sensitivity matrix
a	binary tree element
a,b	lower and upper inequality constraint limits
A,B,C,D,E,α,β	linearized DAE state space matrices
c	error tolerance ($=\epsilon_{tol}$)
c	heat capacity
D,E,G	matrices that define the NMPC constraints

d,e,g	vectors that define the NMPC constraints
f	dependent variables
f	model equation residuals
f_s	dependent variables (stored)
f_{AE}	algebraic equations
f_{DAE}	differential algebraic equations
f_{est}	dependent variables (estimated)
f_{ODE}	ordinary differential equations
g	inequality constraint
g	output function
g_I	inactive inequality constraint
active	inequality constraint (at the equality bound)
h	equality constraint
h	inequality constraints
h	output function
h_L	specific liquid enthalpy
h_V	specific vapor enthalpy
h	specific enthalpy
I	identity matrix

J	sparsity matrix (or incidence matrix)
J	objective function
k	thermal conductivity
k_i^{hm}	harmonic mean of the thermal conductivity
L	Lagrangian function
$\tilde{M}_x^{zero-order}$	matrix defining the initial EOA based on a zero-order approximation
m	reduced model order
M_D	dependency matrix
M_i	matrix defining EOA in i coordinates ($i = x$, y, or z)
M_x^e	matrix that defines the expanded EOA in x coordinates
M_z	matrix defining EOA in x coordinates
$\dot{n}_L$	liquid molar flow rate
$\dot{n}_V$	vapor molar flow rate
$\hat{u}^*$	approximate optimal input
N	horizon length
n	full model order
n	horizon length
n	number of states
n	sparsity matrix size $(n \times n)$

$\tilde{P}$	Galerkin projection
P	projection matrix
p	model parameters
P_i^{sat}	saturation pressure of compound j
Q_i	unitary matrix of a Schur decomposition in i coordinates ($i = x$, y, z, or R)
Q_y	inverse of the measurement covariance
R	equation residual
R	orthonormal basis matrix
r	variable residual
s	independent variables
s_q	independent variable (query)
s_s	indenpendent variable (stored)
T	similarity transform
T	temperature
t	time
T_{ij}	similarity transform from j to i coordinates (i, $j = x$, y, or z)
u	inputs
u	reflux ratio
u^*	optimal inputs

$\bar{x}$	reduced model variables
X	linear regression matrix
x	dependent variable in x coordinates
x	differential equation variables
x	distance
x	states
x_i	composition at stage i
x_L	liquid mole fraction
x_q	dependent variable (query) in x coordinates
x_{ROM}	reduced order model states
x^c_{ROM}	reduced order model states with linear correction
x_A	liquid mole fraction
Y	linear regression matrix
y	algebraic equation variables
y	dependent variable in y coordinates
y	measured states
y	output variable
y_A	vapor mole fraction
Y_m	vecter of model values

y_m	vector of model states
y_n	normalized vector in y coordinates
y_q	dependent variable (query) in y coordinates
Y_s	vector of measurements
y_s	vector of measurements
z	algebraic or differential equation variables
z	dependent variable in z coordinates
z_q	dependent variable (query) in z coordinates

Chapter 1

Introduction

The dynamic modeling of chemical and biological processes, using first principles, usually leads to mathematical models that are systems of differential and algebraic equations. Two main classes exist: lumped parameter and distributed parameter models. The lumped parameter models are mostly systems of differential and algebraic equations. The differential equations stem from material and energy balances over different finite control volumes, while the algebraic equations usually describe the physical, chemical, and thermodynamic properties of the system. Often the algebraic relations represent dynamic mechanisms that occur infinitely fast. This time scale difference is used in model reduction of reaction networks, hence reduced models for the kinetics are sometimes DAEs. Fundamental models for distributed parameter systems consist of sets of partial, differential, and algebraic equations. In these models, the conservation laws are expressed around infinitesimal control volumes, leading to partial differential equations. Using different techniques, e.g., the method of lines, these models are approximated as DAEs so they can be numerically solved.

The quandary of an engineer who must develop a dynamic physicochemical or biological model to use in process control is that there is a very large range of possible models that can be used, from simple to complex. Every model incorporates assumptions that must be made by the modeler, who usually does not know a priori the impact of the assumptions on model accuracy or control quality. If the model is too complex (e.g., over 20 states), then the computation time for control calculations may be prohibitive, in the range of several hours, for a process that responds with time constants on the order of several minutes. What is needed is a methodology that allows the modeler to use a rigorous model, and imbeds model reduction and computation reduction into the CAD approach, so that time reductions by a factor of 100 can be achieved, permitting real-time calculations. Using rigorous models has been a recent trend in related fields, e.g., in process simulators such as HYSYS and in computational fluid dynamics software such as Fluent. Developing a methodology to use such models for process control is the main thrust of this research.

The proposed approach consists of unifying four steps in order to carry out model-based control of DAE systems in real-time (order of several minutes between control changes):

1. differential and algebraic equation reduction (new adaptive approach)
2. shift control calculations offline for efficient online retrieval, using in situ adaptive tabulation (new applications for estimation and control)
3. explicit solution to moving horizon state and parameter estimation

4. explicit solution to receeding horizon control

The challenge is to make this approach work for DAE systems with hundreds of variables, which has not been done successfully before.

1.1 DAE Model Reduction

Large scale first principles models can consist of hundreds of differential equations and thousands of algebraic equations. Solving the differential algebraic equations (DAEs) simultaneously in simulation and control applications can pose a numerical challenge. Other motivations for model reduction are for storage and retrieval of optimal control trajectories, insight into the model structure, and analysis of dynamic degrees of freedom.

Nonlinear model reduction approaches such as balanced covariance matrices (BCM) and proper orthogonal decomposition (POD) have been developed to optimally reduce the number of differential states. However, these model reduction approaches cannot reduce the number of algebraic equations. Because the algebraic equations often greatly outnumber the differential states, significant order reduction of the overall model is not achieved by POD and BCM, which construct a reduced model from a linear combination of the original states. During this transformation, physical significance of the variables is lost. In applications it is often desirable or required for a reduced model to retain physical significance of the original variables.

Other approaches have been suggested for DAE model reduction, but they generally suffer from poor scaling to large scale problems or extensive model configuration [20]. The proposed technique in this work has the advantage of good scaling for large scale problems and no special model manipulation. An added advantage is that the physical significance of the algebraic equations is retained. A major focus of this work is also in making the model reduction approach adaptive in order to achieve a specified level of accuracy compared to the orginal model. Being adaptive, the DAE model is reduced automatically with no prior training simulations.

1.2 Storage and Retrieval

In control, a group of inputs are used to determine a certain number of outputs. If the model is deterministic, the same set of independent variables (inputs) will always produce the same set of dependent variables (outputs). In block diagram form, the inputs (s) enter the system and leave as a set of outcomes (f) . Sensitivity information may also be optionally available from the function evaluation. The sensitivity matrix (A) reveals the amount that f

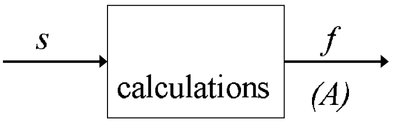

Figure 1.1: Block diagram of a deterministic calculation of f based on independent variables s. The block diagram may represent open loop simulation or a simplification of closed loop control.

changes with a small perturbation in s.

$$A = \frac{\partial f}{\partial s} \tag{1.1}$$

In some applications, it is desirable to store previously computed values of f in order to estimate future values of f without redoing the usual calculations. In this process of storage and retrieval it is desirable for the estimated values of f to be within some error tolerance (ϵ_{tol}) of the actual f.

$$|f - f_{est}| \leq \epsilon_{tol} \tag{1.2}$$

Costs associated with a storage and retrieval method include configuration costs, CPU time costs, and storage costs. Configuration costs are largely a function of the degree to which the method is generalized and automated. CPU time costs include the construction of a database and the retrieval time. Storage costs can be a factor if the dimension of s or f is large. Overall, storage and retrieval may be desirable if the following conditions exist.

1. Retrieval time is much faster than the original calculation
2. The same calculations are performed repeatedly but with different values
3. Real time constraints make the original calculation infeasible
4. The CPU time to generate the database is small compared with retrieval savings
5. Storage costs are small

Storing and retrieving solutions to sets of nonlinear algebraic equations can be accomplished in many ways. General criteria to benchmark storage and retrieval methods were given by Pope [66].

1. The CPU time required to create the store
2. The memory required for the store
3. Inaccuracies in the retrieved mapping (e.g., interpolation errors)
4. The CPU time required to retrieve from the store
5. The degree to which the technique is generally applicable and can be automated

An exhaustive review of all possible storage and retrieval techniques is beyond the scope of this work. However, one algorithm, the artificial neural network, has become a popular technique for nonlinear function approximation. *In situ* adaptive tabulation (ISAT) is then introduced as a new approach for storage and retrieval. Each of the algorithms is judged by the benchmark criteria.

Neural nets are networks of adaptable nodes which, through a process of learning from task examples, store knowledge about system behavior and make it available for later use [5]. The flexibility and general applicability of neural nets have been demonstrated by diverse applications across many fields of study. Neural nets are an effective tool to incorporate historical data for use in state estimation and control, although filtering and preconditioning the plant data are often time-consuming tasks [67]. One limitation of neural nets is the inability to extrapolate outside the training domain.

1.2.1 *In Situ* Adaptive Tabulation

In situ adaptive tabulation (ISAT) is a storage and retrieval method developed for direct numerical simulation (DNS) of turbulent combustion flames [66]. ISAT directly controls the approximation error by adding multidimensional linear regions to chart unmapped state space. In this way, extrapolation error is kept within specified error tolerances. Another desirable property of ISAT is that the store is constructed *in situ*, without previous training simulations or optimizations. For DNS, ISAT replaces the chemical reaction integrations to greatly enhance the speed of the calculation. As a black-box function approximator, ISAT gradually replaces the original function calculation by storing and retrieving previous computations (see Figure 1.2).

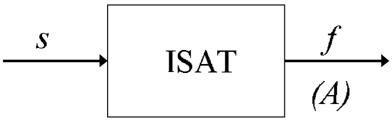

Figure 1.2: ISAT stores solutions and sensitivities (A) to approximate f with multidimensional piecewise linear regions.

1.3 Research Objectives

The main objective of this research is to develop techniques to apply large scale first principles models in real-time control. Detailed models of chemical manufacturing processes often consist of many thousands of DAEs. Solving large scale models in control applications can be computationally infeasible in real-time. Several strategies have been developed to make optimal approximations and simplications. Other objectives of this research include:

1. Optimally reduce the real-time computational requirements of nonlinear model predictive control (NMPC) for large scale models. Many techniques have been proposed to reduce the on-line requirements of NMPC [22] [23] [43], but are generally limited to single process units and small models with short control horizons.

2. Develop adaptive model reduction of DAE models to optimally reduce the model order. This optimal reduction of model order retains the most important dynamic degrees of freedom of the original model. Developing an adaptive approach means that training and application occur simultaneously in an iterative process.

3. Reduce the real-time computational requirements of dynamic state estimation while retaining the accuracy of large scale model based state estimation. Receeding horizon state estimation can be nearly as computationally demanding as the receeding horizon control problem. Because both are solved on-line, both must meet real-time cycle requirements.

4. Propose ISAT as a replacement for neural networks as a general nonlinear function approximator. One of ISAT's limitations was that a sensitivity calculation is required to add a new record to the database. Because many nonlinear function calculations do not include this feature, a modification to the algorithm is necessary.

1.4 Overview of this Dissertation

In this introductory chapter, storage and retrieval of open loop simulations and closed loop control is proposed with ISAT. ISAT efficiently stores multiple linear approximations of a nonlinear solution. It is a generic approach that is applied for storage and retrieval for real-time control. A brief overview of neural networks as a comparison, a history of ISAT development, and discussion of DAE model reduction provide some background for this research. Each of the research objectives is addressed in following chapters.

Chapter 2 gives details of the ISAT algorithm modified to adaptively approximate any nonlinear function. An approximation to the local sensitivity is developed with multivariable linear regression. Unlike neural networks, the ISAT mapping of the nonlinear surface is performed sequentially, thereby avoiding large global optimizations. ISAT and neural networks are directly compared in an illustrative example.

Chapter 3 introduces DAE model reduction. Because ISAT storage and retrieval is more efficient for smaller problems, significant effort has been devoted to extracting optimally reduced small and medium scale models from large scale models. In practice, many large scale models can be reduced with very little reduction in model accuracy. An adaptive DAE model reduction approach is proposed with the only tuning parameters being the required variable accuracy. The adaptive strategy simultaneously refines the reduced model structure and model order with an iterative approach.

Chapter 4 outlines the application of a combined model reduction and storage and retrieval for real-time NMPC. DAE simulations are stored and retrieved to reduce real-time control requirements by 85 times for the regulator. Application to state estimation is also outlined. In sequential or hybrid NMPC formulations, the same store can be accessed for state estimation and the regulator, leading to faster training of the ISAT database.

Chapter 5 proposes another way to dynamically store NMPC solutions. By parameterizing control solutions as a function of current states, NMPC solutions can be stored and retrieved for sequential, hyrbid, or simultaneous solution strategies. The proposed storage of optimal control is potentially more efficient than that of Chapter 4 and requires no customization of the nonlinear programming (NLP) sub-problems. A control study involving a continuously stirred tank reactor (CSTR) model demonstrates an application of ISAT in control.

Chapter 6 is the estimation counterpart to Chapter 5 on control. Chapter 5 reveals an explicit solution procedure for control to reduce the computational demands. However, the estimation problem must also be solved at every time horizon step with a computational load similar to the control problem.

An explicit solution to the unconstrained moving horizon estimation problem is proposed. This explicit solution is able to estimate the current states, parameters, and input or output disturbances. For constrained problems, an iterative solution technique is proposed to guarantee convergence in solution times that are close to the explicit solution. By combining the techniques of chapters 5 and 6, model predictive estimation and control can be implemented without computational hardware restrictions.

Chapter 2

The ISAT Algorithm

Model size, nonlinearity, sparsity, and other factors contribute to the ease or difficulty of obtaining a numerical solution in simulation and control calculations. Generally, small nonlinear models in the range of 100 states or less are amenable to real-time ($\approx$10 seconds or less cycle time) MPC implementation. By reducing the model size, larger models can be efficiently applied in real-time control applications.

Another real-time feasible MPC strategy involves shifting the computational burden off-line for efficient on-line retrieval. Storage and retrieval of control trajectories can eliminate the on-line computational burden of model predictive control. By reducing the control calculations to a simple lookup of precomputed solutions, advanced control can be applied to applications that do not merit large computational resources. The purpose of this chapter is to demonstrate the application of a storage and retrieval algorithm, ISAT, that compactly stores the precomputed control solutions, efficiently accesses the values to meet fast sampling constraints, and adaptively builds the store when new information is accessible. In applying ISAT to control calculations, some of the specially tailored features designed for the original application in simulation of reacting turbulent flows [66] had to be modified. However, this chapter does not actually discuss the specific tailoring to control. Instead, the ISAT algorithm is made generic to store and retrieve any deterministic nonlinear function. Subsequent chapters then take this generic framework to show control applications. By starting general and becoming specific, all applications of ISAT can be seen in one context. Also, this will aide application to other areas outside of control or combustion modeling.

Besides generalizing ISAT for a range of applications, a new development in this chapter is a more thorough explanation of the algorithm on a step-by-step basis. This is intended to expose all of the details to facilitate future development. One of the biggest limitations to widespread use of ISAT as a general nonlinear function approximator is the requirement of sensitivities. As a new development, sensitivities are estimated from a database of previous input-output data using linear regression. A filtering strategy is able to determine when sufficient data exist to form a locally accurate linear approximation.

In storage and retrieval, the goal is to retain the accuracy of the original calculations while substantially lowering the computational costs. Analogies to the ISAT method exist in many different industries and products. For example, computer systems are built with multi-layers of caching. One of the reasons that Pentium® processors are considered superior to Celeron® processors is the larger amount of cache. This cache stores and accesses frequently computed instructions and data and thereby improves the processor performance.

As another example, the computer can speed-up the effective download speed for internet connections by storing web pages on the hard disk. When a web site is visited again, the page can be loaded from the much faster hard disk. The common characteristics of these speed-up technologies are:

1. The first time through there is no speed-up. In fact, there may be some slow-down associated with building and storing the database.
2. Search time is generally fast compared to repeating the operation.
3. Storage costs for the database are low compared to the cost of repeating the operations.
4. There is a sufficient probability that the operation will be repeated, otherwise the database would serve mainly as an archive.
5. The system performance increases as the database matures and more operations are repeated.

ISAT is storage and retrieval algorithm for nonlinear functions. These nonlinear functions may be time intensive computer simulations, calculations that require real-time results, or for applications that do not merit substantial compuational power. As a data-based application, there is a phase of training associated with every application. As the database matures and retrievals occur, ISAT uses a binary tree architecture to ensure fast search time. With a parallel increase in data storage capacity and processor speed in modern computers, storage costs rarely become a factor in ISAT applications [21].

2.1 Review of ISAT for Turbulent Combustion Simulations

Detailed combustion models typically include reactants, products, and reaction intermediates that result from hundreds of reactions. These reaction timescales can range from 10^{-9} to 1 second. Models with a large range of timescales produce a stiff system that is difficult to integrate.

Analytical and numerical tools have been developed to optimally reduce the kinetic models. Some of these tools include sensitivity analysis, principal component analysis, and species lumping procedures [85]. Another tool is computational singular perturbation (CSP) as a formal way to apply partial-equilibrium approximations on an a priori basis [46].

Many of the methods for creating reduced mechanisms rely on steady-state or partial-equilibrium approximations. However, the reduced mechanisms are generally limited to a range of temperature, pressure, and/or species' concentrations, known as the thermochemical space. Outside of this defined space, large errors can occur. To overcome this deficiency, Mass and Pope proposed a new method for reducing the simulation burden of detailed chemical kinetics based on intrinsic low-dimensional manifolds (ILDM) [50] [51]. However, the ILDM method also had the following shortcomings [93].

1. Storage requirements increase dramatically as the manifold dimension increases.
2. The entire thermochemical space must be calculated for a fixed dimen-

sional manifold that cannot be easily adapted when a higher dimensional manifold is required.

3. For higher dimensional manifolds, the work to retrieve information is not trivial.

4. There is a lack of dynamic error control.

5. Existence, uniqueness, and continuity of the lower dimensional manifold are not guaranteed.

ILDM was created as a dimension reduction technique that gave modest improvements in computational performance. Later, Pope developed *in situ* adaptive tabulation (ISAT) to directly reduce the computational requirement without dimension reduction [66]. The ISAT method calculates and stores the data *in situ* rather than as a preprocessing step. Thus, only areas of the thermochemical space that are accessed are included in the database. Another benefit of ISAT is the addition of error control that seeks to limit the retrieved data is within a specified error tolerance [49]. In a turbulent flame simulation, a reduction by a factor of 1000 in the computational effort was demonstrated [66].

Consider how ISAT reduces computational time of simulating the complex chemistry in turbulent flames. Often tens of chemical species are linked together by thousands of possible chemical reaction pathways. Coupling the chemistry, convection, and diffusion in a simultateous simulation is often too computationally demanding. As a first step, the chemistry integration is isolated from other physical simulations, such as mixing, by one of many splitting schemes. During the course of the reacting flow simulation, integration queries consisting of initial states (ϕ_0), an integration time (Δt), and an error tolerance (ϵ_{tol}) are sent to ISAT many times. ISAT returns the final states (ϕ_f) of the chemistry integration within the specified error tolerance. Since Pope first

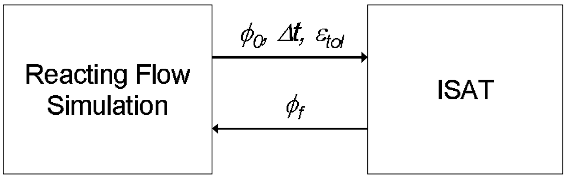

Figure 2.1: Block diagram of ISAT interaction with the reacting flow simulation. ISAT stores and retrieves the thermochemical properties involved in chemical reactions. Because the chemistry is decoupled from the other aspects of the simulation, the final chemical compositions are a unique function of the initial concentrations and time.

published the ISAT method, there have been numerous applications of ISAT in combustion to simulations that were previously quite formidable. Saxena and Pope [74] [75] simulated a piloted jet diffusion flame of $CH_4 - air$ with 16 species and 41 reactions. A significant speedup was not reported because the chemistry calculations took only 60% of the total CPU time. Other simulations have shown that up to 99.9% of the CPU calculation are chemistry related, making possible an overall reduction of 1000 times [66]. Shah and Fox [78] performed computational fluid dynamic (CFD) simulations of methane thermochlorination reactors involving 38 species with a speedup of 138 over direct integration. They also mentioned that ISAT has been successfully applied to a mechanism with 116 species and 447 reactions, although no further details were given. Xu and Pope [92] performed another simulation of piloted jet flames of methane with a parallel implementation of ISAT with an estimated speedup of 40.

There has been some interest in reducing the storage requirements for ISAT. Tang and Pope developed an extension that combines ISAT and model reduction through rate-controlled constrained equilibrium, abbreviated ISAT-RCCE [81]. ISAT-RCCE as applied to a mechanism with 32 species and 175 reactions shows a speedup factor of 500 over direct integration. Another method to reduce storage was proposed by Chen et al. [21]. The ISAT database is replaced by a neural net, thereby reducing the storage requirement from ≈100 MB to ≈1 MB. Even though there is a savings in memory, there is a loss of error control. The authors mention that by using a neural net, extrapolation would produce unpredictable results and that ISAT should be used for points outside the training domain.

2.2 Details of the ISAT Algorithm

The ISAT algorithm was originally developed for storage and retrieval of initial value problems (IVPs) involving ordinary differential equation (ODE) models (see Figure 2.2). ISAT was originally developed to store and retrieve ODE numerical integrations. Given the initial states, the final states are approximated by a linear extrapolation from a neighboring solution. ISAT at-

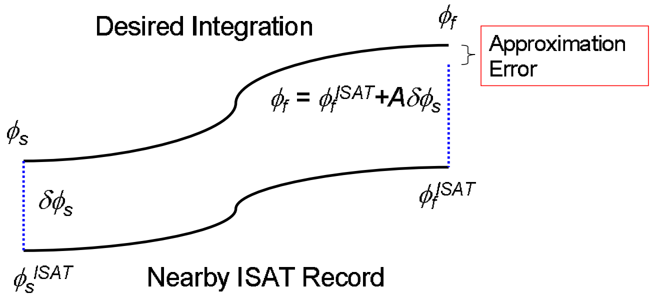

Figure 2.2: ISAT was originally designed to store and retrieve numerical integrations of chemistry evolution in turbulent combustion simulations.

tempts to control the approximation error by defining a region of accuracy around the initial state.

A generalized development of the algorithm is necessary for the broader application as a nonlinear function approximation tool for cases in which gradient information is not available. As a nonlinear function approximator, the potential applications of ISAT are greatly expanded. In subsequent chapters, the ISAT algorithm is applied to IVPs involving differential-algebraic equations (DAEs) and regulator solutions for nonlinear model predictive control (NMPC). A generalized form of the ISAT algorithm follows.

2.2.1 The ISAT Record

The basic unit of the ISAT database is the record. An ISAT record consists of the initial states and inputs, the final states, a sensitivity matrix, and an ellipsoid of accuracy (EOA). The sensitivity can be estimated when it is not explicitly available from the function calculation. The EOA is a matrix used to control the retrieval error. A distinguishing feature of ISAT over other storage and retrieval methods is the automatic error control. For optimal control, s is a set of parameters and initial states, f is a set of inputs to the system, A is a sensitivity of the inputs to s, and M is an ellipsoidal region about s that defines the linear approximation limit in order to achieve a desired accuracy.

Table 2.1: Elements of the ISAT record, along with the vector and matrix dimensions

ISAT Record Element	Symbol and Dimension
Independent variables	$s \in \mathbb{R}^m$
Dependent variables	$f \in \mathbb{R}^n$
Sensitivity	$A \in \mathbb{R}^{nxm}$
Ellipsoid of accuracy	$M \in \mathbb{R}^{mxm}$

2.2.2 Searching the Records with Binary Trees

When accessing the database, the only piece of information that is known is a query vector of initial conditions. Ideally, a stored record is retrieved that minimizes the approximation error. However, the approximation error cannot be verified without performing the calculation of interest, thereby negating the utility of storage and retrieval. Generally, closer records produce lower approximation errors because the linear approximation is locally accurate. The approximation error is sub-optimally minimized by selecting a record that minimizes a measure of closeness. In this case the measure of closeness is the difference between the query vector (s_q) and stored vector (s_s) .

$$x = |s_q - s_s| \tag{2.1}$$

Searching the ISAT records sequentially would require $O(N)$ operations to completely search the database and find the closest record. A more efficient search structure is the binary tree. A balanced binary tree requires $O(log_2(N))$ operations for locating a record. One of the drawbacks to binary tree searching is that the closest record is not always selected. To overcome this deficiency, multiple binary trees are used to increase the probability of finding the closest record. The records are equally divided among the binary trees to maintain a balance in search times. Once all of the binary trees are searched, a sequential search is performed to determine the closest record among the ones the binary trees selected. By adjusting the number of binary trees, an effective compromise is reached between the accuracy of the sequential search and the speed of the binary tree search.

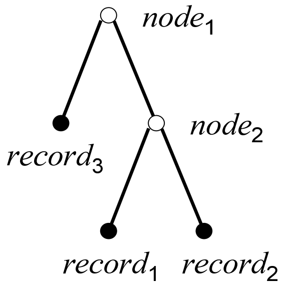

Figure 2.3: Each node of a binary tree can either be a leaf or branch. The leaves of the binary tree are individual records of the ISAT database. A branch, on the other hand, points to two other nodes. All branches divide until a leaf terminates the line.

Once a close record is located, ISAT performs one of three scenarios. These scenarios include retrieval, growth, and addition. Each of these is described in more detail below.

2.2.3 Record Retrieval

The automatic error control decides if retrieval is appropriate. The error control is accomplished with the ellipsoid of accuracy (EOA) with a center being the stored start. Another point, s_q is within the EOA if $x^T M_x x \leq \epsilon_{tol}^2$. If the query point is within the EOA then f is estimated (f_{est}) with a linear approximation using the sensitivity (A).

$$f_{est} = f_s + Ax \tag{2.2}$$

If $x^T M_x x > \epsilon_{tol}^2$ then the point s_q is outside of the EOA and a retrieval cannot be performed. Even though the query point is not inside the EOA, the linear approximation may still be within the error tolerance for f_{est}. The next step of the algorithm is to check the actual error.

2.2.4 Record Growth

When retrieval is not possible, the approximation error is computed. In order to check the actual error, an original function evaluation must determine the correct value of f $(f = function(s_q))$. If $|f - f_{est}| > \epsilon_{tol}$, the EOA should not be expanded. Instead a new record should be added to the ISAT database.

The growth step should be skipped and the algorithm jumps ahead to the ISAT addition phase (see section 2.2.5).

If $|f - f_{est}| \leq \epsilon_{tol}$, the EOA can be expanded to include s_q. This new region is a minimum volume ellipsoid that includes the new point, s_q, and the original EOA. The growth algorithm involves six steps. Each of the steps is described first in mathematical terms and subsequently with a two dimensional graphical example.

2.2.4.1 Definition of the EOA and growth point

After replacing ϵ_{tol} with c to simplify the notation, the EOA is defined by all possible query points that satisfy Equation 2.3.

$$x^T M_x x \leq c^2 \tag{2.3}$$

In the two-dimensional example shown in Figure 2.4, an ellipse is centered about the origin as an estimate to the region of accuracy. This ellipse can grow as the region of accuracy is revealed with further query points.

2.2.4.2 Transform the coordinates to map the EOA to a unit hypersphere

In this step a matrix T_{yx} is computed to map the original x-coordinates onto a new y-coordinate system that transforms the EOA into a unit hypersphere. A unit hypersphere is simply a higher-dimensional generalization of the three-dimensional sphere with radius of one. The matrix T_{yx} maps all points in x into the y coordinates with the relation $y = T_{yx}x$. Likewise, the

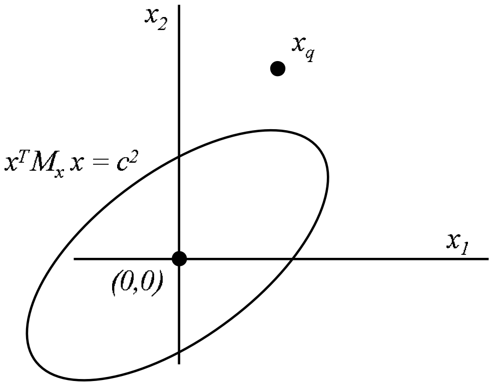

Figure 2.4: The center point of the ellipse is the origin. The growth point s_q becomes x_q after the translation and the EOA is defined in terms of x.

inverse of T_{yx} (or T_{xy}) maps y into the x coordinates with $x = T_{yx}^{-1}\, y = T_{xy} y$. The first subscript letter of T refers to the transformed coordinate system while the second subscript letter refers to the original coordinates.

A Schur decomposition gives $M_x = Q_x \Sigma_x Q_x^T$ with Q_x being a unitary matrix ($Q_x^T = Q_x^{-1}$). The square root of the diagonal matrix Σ_x is computed by taking the square root of the individual elements along the diagonal. The transformation matrix becomes $T_{yx} = c^{-1} \Sigma_x^{1/2} Q_x^T$.

It will now be shown that the coordinate transform does, in fact, transform the EOA to a unit hypersphere in the new coordinate system. First, the inverse of T_{yx} is found to be

$$T_{yx}^{-1} = T_{xy} = c Q_x \Sigma_x^{-1/2} \tag{2.4}$$

Making the substitution $T_{xy}y = x$ in the EOA equation gives

$$(cQ_x\Sigma_x^{-1/2}y)^T M_x(cQ_x\Sigma_x^{-1/2}y) = c^2 \tag{2.5}$$

Rearranging and substituting $M_x = Q_x\Sigma_x Q_x^T$ gives

$$c^2(y^T\Sigma_x^{-1/2}Q_x^T)Q_x\Sigma_x Q_x^T(Q_x\Sigma_x^{-1/2}y) = c^2 \tag{2.6}$$

The c^2 term cancels and $Q_x^T Q_x = I$ because Q_x is a unitary matrix. This leaves

$$y^T\Sigma_x^{-1/2}\Sigma_x\Sigma_x^{-1/2}y = 1 \tag{2.7}$$

Finally, because $\Sigma_x^{-1/2}\Sigma_x\Sigma_x^{-1/2} = I$ the EOA in transformed space becomes a unit hypersphere.

$$y^T I y = 1 \tag{2.8}$$

2.2.4.3 Map the growth point to the transformed coordinates

The same transformation matrix T_{yx} is used to transform the growth point to the new coordinates.

$$y_q = T_{yx}x_q \tag{2.9}$$

The magnitude and normalized direction of y_q are important for subsequent calculations. The magnitude is the Euclidean norm of y_q.

$$mag\,(y_q) = \|y_q\|_2 \tag{2.10}$$

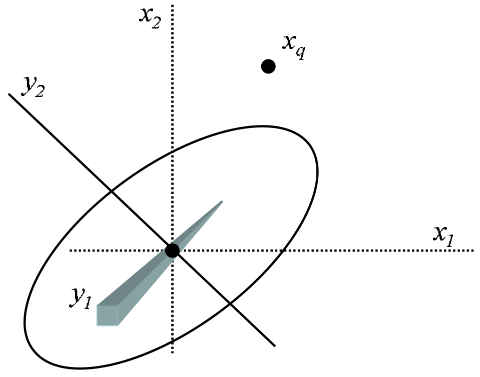

Figure 2.5: In a two dimensional example, the y-axes are shown relative to the x-axes. In the y-axes reference frame, the ellipse becomes a unit circle centered at the origin.

The normalized direction is simply the vector divided by the magnitude.

$$y_n = \frac{y_q}{\|y_q\|_2} \tag{2.11}$$

2.2.4.4 Align one of the y-axes with the direction of the growth point

One of the y-axes must be aligned with the direction of the growth point. This is accomplished by computing an orthonormal basis to y_n. An orthonormal basis is produced by first subtracting the outer product of y_n from the identity matrix of appropriate dimension.

$$R = I - y_n y_n^T \tag{2.12}$$

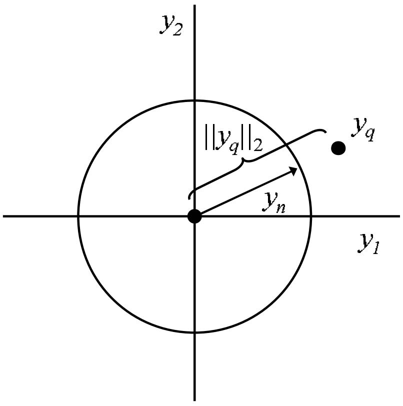

Figure 2.6: The growth point is translated to the new y reference frame. The magnitude is the distance between y_q and the origin. The normalized vector y_n has a unit length and points in the direction of y_q.

A Schur decomposition of R gives

$$R = Q_R \Sigma_R Q_R^T \tag{2.13}$$

The diagonal matrix Σ_R is equal to the identity matrix except that one of the diagonal elements is zero. This diagonal element corresponds to the axis that is aligned with y_q in the z coordinate system. The transformation matrix is the transpose of the unitary matrix from the Schur decomposition (or obtained more efficiently by a Householder transformation).

$$T_{zy} = Q_R^T \tag{2.14}$$

A new coordinate system is defined by $z = T_{zy}y$. Transforming the y coordinates to the z coordinates the EOA becomes

$$(T_{yz}z)^T I (T_{yz}z) = 1 \tag{2.15}$$

Rearranging gives

$$z^T T_{yz}^T T_{yz} z = 1 \tag{2.16}$$

Because T_{yz} has the special property of a unitary matrix that $T_{yz}^T T_{yz} = I$, the EOA is also a unit hypersphere in the z coordinates.

$$z^T I z = 1 \tag{2.17}$$

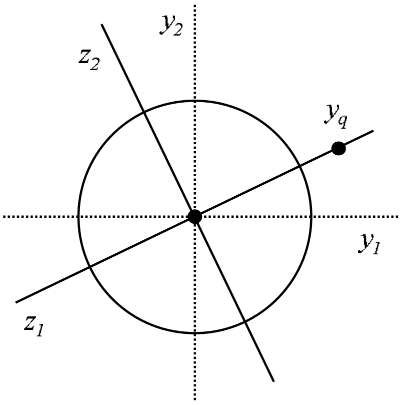

Figure 2.7: The axes are rotated so that one axis aligns with the growth point. This rotation is important so that the ellipse can be expanded along the aligned axis.

2.2.4.5 Grow the hypersphere into an ellipsoid that reaches the growth point

The half length of the axis, aligned with y_q, is expanded by modifying the appropriate element of the identity matrix. This is the same element that corresponds to the zero diagonal element of Σ_R. In this case, the first diagonal element is shown as the appropriate selection. The semi-axis of the i^{th} axis of an ellipsoid $z^T M_z z = c^2$ is $(c^2/\sigma_i)^{1/2}$ where σ_i is the i^{th} eigenvalue of M_z. In order to stretch the hypersphere into a minimum volume ellipsoid that includes z_q and the original EOA, the half length is lengthened to the magnitude of z_q. Because the magnitude of z_q is equal to that of y_q, the matrix element is set to $\|y_q\|_2^{-2}$.

$$M_z = \begin{bmatrix} \|y_q\|_2^{-2} & 0 & \cdots & 0 \\ 0 & 1 & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix} \tag{2.18}$$

The grown EOA is $z^T M_z z = 1$.

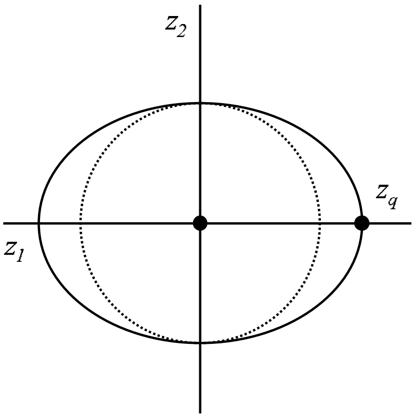

Figure 2.8: The circle is expanded into an ellipse that reaches the growth point. This is a minimum area expansion of a symmetric ellipse.

2.2.4.6 Transform the expanded ellipsoid back to the original coordinate system

The grown EOA is transformed back to the original coordinate system with inverse transformation matrices. The z coordinates are a function of the x coordinates according to

$$z = T_{zy} y = T_{zy} T_{yx} x \tag{2.19}$$

Substituting for z in the EOA equation and multiplying both sides of the equation by c^2 reverts back to the x coordinates and recovers the form of the original EOA. This is the minimum volume ellipsoid that includes the original ellipsoid and the growth point.

$$x^T M_x^e x = c^2 \tag{2.20}$$

with the expanded volume determined by

$$M_x^e = c^2 (T_{yx}^T T_{zy}^T M_z T_{zy} T_{yx}) \tag{2.21}$$

2.2.5 Record Addition

When $|f - f_{est}| > \epsilon_{tol}$ the EOA should not be expanded. Instead a new record should be added to the ISAT database. The core elements of an ISAT record are s, f, A, and M_x. Each of these elements is discussed in the subsequent sections.

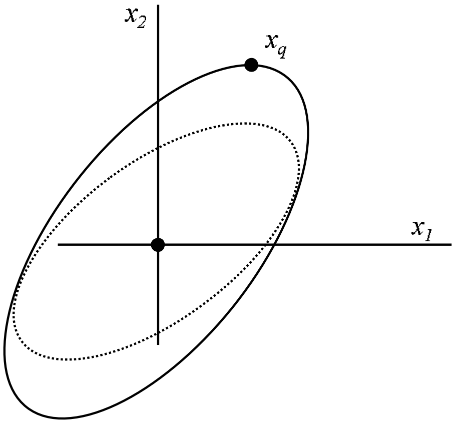

Figure 2.9: When the ellipse is transformed back to the original coordinates, x_q is on the ellipse perimeter. In addition, the ellipse is a symmetric minimum area expansion that includes the growth point and the original ellipse.

2.2.5.1 Initial states and inputs

The vector s is the query point that is not eligible for retrieval or growth. This point becomes the center of the new EOA.

$$s = s_q \tag{2.22}$$

2.2.5.2 Final states

The vector f comes from an original function calculation. There is no computational advantage with growth or addition because a complete calculation is required. Record growths and additions are part of the database building phase. The real advantage of ISAT occurs when retrievals greatly outnumber growths and additions.

2.2.5.3 Sensitivity

Sensitivity information may also be optionally available from the function evaluation. For dynamic systems, the intial state sensitivities can be solved simultaneously with the state equations. Automatic differentiation can improve the accuracy and reduce the computational burden that is required to obtain sensitivities. The sensitivity matrix A reveals the amount that f changes with a small perturbation in s.

$$A = \frac{\partial f}{\partial s} \tag{2.23}$$

When the sensitivity is not available from the function evaluation a statistical approximation can be made. At least m, where m is the dimension of the vector s, function evaluations are required to calculate an accurate sensitivity. The function evaluations can be obtained by sorting through a database of previous results or by generating new results.

When sorting through a database of previous results, care should be taken to select records that are close to s_q otherwise the sensitivity may not be locally accurate. The data filtering routine in this section may be modified based on the known characteristics of the function that is approximated. The filter suggested here is designed for functions that exhibit discontinuities, regions that are not continuously differentiable, or strong nonlinearities. When a sensitivity is requested, a multiple binary tree search is conducted to gather a set of $4m$ records that are close to s_q ($s_q \in \mathbb{R}^m$). Multivariate linear regression is used to obtain a sensitivity about s_q. If the residuals from the regression are less than the requested ISAT tolerance, the sensitivity is accepted as a local approximation. In the event that the requested ISAT tolerance is not met, data are filtered from the set by two alternate means. The first filter eliminates the record with the highest regression residual. The second filter removes the record that is furthest from s_q in the 1-norm sense. These filters remove records until the regression tolerances are met or until fewer than m linearly independent records remain. Linear independence of the records is examined with a singular value decomposition of the raw data set. Independence is maintained if m non-zero singular values exist.

When generating new results, m linearly independent vectors of s should be generated around s_q. These linearly independent vectors can be created by defining an orthonormal basis as $R = I - yy^T$, where y is any unit vector. Performing a function evaluation m times for all of the s vectors can be a cpu time intensive step.

Once the m function evaluations are completed, the sensitivity can be estimated through multivariate linear regression. Each of the s and corresponding f vectors are first subtracted from s_q and f_q.

$$\Delta s = s_q - s \tag{2.24}$$

$$\Delta f = f(s_q) - f = f_q - f \tag{2.25}$$

The linear regression model includes a residual vector, ϵ_{resid} , as an indication of how much each local result deviates from the linear model. A large residual indicates that a perturbation of s does not fit in with the linear model. This could indicate that the perturbation of s should be reduced to generate locally linear solutions to f.

$$\Delta f = A \Delta s + \epsilon_{resid} \tag{2.26}$$

The vectors are assembled into matrices X and Y.

$$X = \begin{bmatrix} \Delta s_1 \\ \vdots \\ \Delta s_m \end{bmatrix}^T \tag{2.27}$$

$$Y = \begin{bmatrix} \Delta f_1 \\ \vdots \\ \Delta f_m \end{bmatrix}^T \tag{2.28}$$

An estimate of the sensitivity is calculated by simple matrix multiplications.

$$A = Y X^T (X X^T)^{-1} \tag{2.29}$$

The regression approximation to the sensitivity can be poor if data are not in the local linear area. However, an inaccurate sensitivity approximation will not degrade the error control, but will likely decrease the efficiency of ISAT. Without an accurate first order approximation, the EOA size is limited to smaller local region. It is preferable to use integrated sensitivity analysis when available. For example, popular ODE and DAE integrators such as ODESSA, DASAC, and DASPK include the capability to compute the sensitivity simultaneously with the integration. This sensitivity information is used in the first order approximation of f_{est}.

2.2.5.4 Initial estimate of the EOA

An initial estimate of the EOA should be conservative for good error control. ISAT employs a first-order approximation for estimating f. Second order and higher terms have been truncated from this approximation.

$$f_{est} = f_s + Ax \tag{2.30}$$

By assuming a zero-order function estimation, the first-order term becomes an approximation to the truncation error.

$$f_{est} = f_s \tag{2.31}$$

$$\epsilon_{trunc} \approx Ax \tag{2.32}$$

Substituting ϵ_{trunc} for the error term in the EOA equation gives an approximation for the zero-order EOA.

$$x^T M_x x = \epsilon^2 = \epsilon_{trunc}^T \epsilon_{trunc} = (Ax)^T (Ax) = x^T (A^T A) x \tag{2.33}$$

$$M_x^{zero-order} = A^T A \tag{2.34}$$

Sometimes the zero-order approximation produces an EOA with a large principle axis because of a low sensitivity in a particular direction or because the dimension of f is less than s. To remedy this problem, the singular values of A are adjusted to be at least $\epsilon_{tol}^2/2$. To accomplish this, a singular value decomposition of A is performed to give $U \Sigma V^T$. Any diagonal elements below $\epsilon_{tol}^2/2$ are raised to the minimum value. The corrected matrix is reconstructed from the new diagonal matrix of singular values.

$$\tilde{A} = U \, \tilde{\Sigma} \, V^T \tag{2.35}$$

The corrected zero-order approximation of the EOA ensures that large principle axes are conservatively reduced.

$$\tilde{M}_x^{zero-order} = \tilde{A}^T \tilde{A} \tag{2.36}$$

2.2.5.5 Binary Tree Addition

Once all of the ISAT record elements are computed, the record is added to the binary tree. The growth of the binary tree involves the creation of a new node. In this case, the record added to the tree is $record_3$. Supposing that $record_3$ is closer to $record_2$, the tree is grown on the right branch with the creation of $node_2$. The new $node_2$ is defined by ν_2 and a_2 which are obtained from the following equations, where s_2 and s_3 belong to the new record and next closest record, respectively.

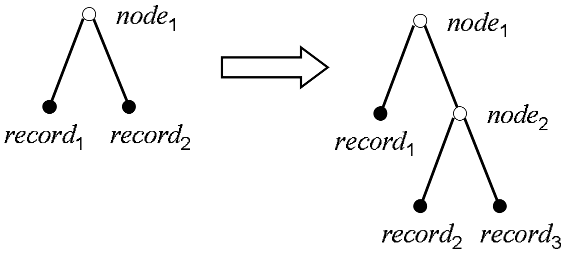

Figure 2.10: The binary tree is grown to include a new record. The growth creates a new node where the next closest record previously appeared.

$$\nu_2 = s_3 - s_2 \tag{2.37}$$

$$a_2 = \nu_2^T \left(\frac{s_3 + s_2}{2} \right) \tag{2.38}$$

As a final step, the $node_2$ pointers are linked to $record_2$ and $record_3$, and $node_1$ points to $node_2$.

2.3 Scaling to Large Scale Problems

One of the limitations of the ISAT method is that the storage requirements are proportional to n^2, where n is the total number of states [21]. Therefore, smaller models are better suited to computational reduction through ISAT. A practical limit may be on the order of 100 states (see examples in Chapter 4).

2.4 Example: Comparison of ISAT and Neural Networks

ISAT and neural networks are compared in this example. Neural networks were selected as a competitive alternative due to their success in control applications. For this example, all retrievals are purposely kept within the training domain. ISAT directly controls the most intuitive tuning parameter for nonlinear function approximation: The amount of error between the actual function and its approximation. Neural networks tuning parameters are centered on the network structure and optimization tolerance for convergence. These tuning parameters are less intuitive and lead to an indirect error control scheme. The first eigenfunction of an L-shaped membrane is selected as a test problem for the comparison (see Figure 2.11). One quadrant of the eigenfunction is linear while the three quadrants are a continuous nonlinear function.

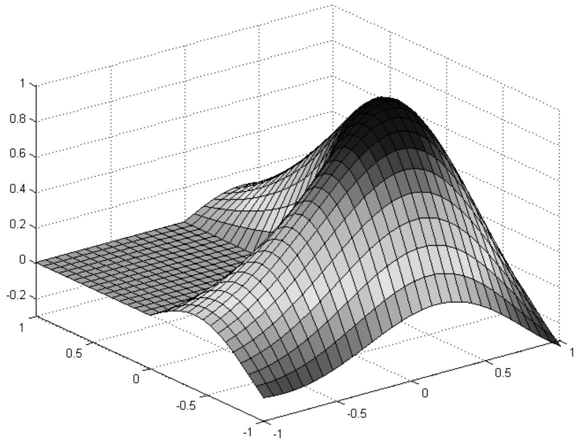

Figure 2.11: The first eigenfunction of an L-shaped membrane used to compare ISAT and neural networks. The second and third eigenfunctions have also been shown in MathWorks' publications.

The eigenfunction is a good test of nonlinear function approximation algorithms because it exhibits both linear and nonlinear regions with parts that are not differentiable. The horizontal axes x and y are the independent set. The vertical axis z is the dependent set. Data were generated by selecting coordinates of x and y at 31 equally spaced intervals for a total of 961 ($= 31^2$) function evaluations. On the graph, the intersection of two lines indicates a point where a function evaluation occurred.

Because the sensitivities are not available from the calculation, ISAT used a statistical approximation for the slope at each point. ISAT's principal tuning variable is the absolute tolerance for function approximation error (ϵ_{tol}). As the error tolerance is lowered, the number of linear regions in the ISAT approximation increases. To illustrate, the error tolerance was initially set at $\epsilon_{tol} = 0.5$. Because the z values range from -0.3 to 1.0, an error tolerance of 0.5 is extremely coarse. ISAT computed 12 linear regions to approximate the nonlinear function (see Figure 2.12). The shape of the nonlinear function is barely recognizable because the nonlinear region is approximated with only a handful of linear functions. One good aspect of the approximation is that the left quadrant is exactly represented by ISAT's linear approximation. Decreasing the error tolerance to $\epsilon_{tol} = 0.1$ produces considerably better results with a total of 48 linear regions. However, there are still regions of the approximation that approach the maximum error tolerance (see Figure 2.13). Finally, with an error tolerance of $\epsilon_{tol} = 0.01$ and 206 linear regions, the ISAT approximation resembles the original function (see Figure 2.14). For this example problem, the number of linear regions increases proportional to the reciprocal of ϵ_{tol}. With other applications of ISAT, ϵ_{tol} should be chosen to balance the costs of

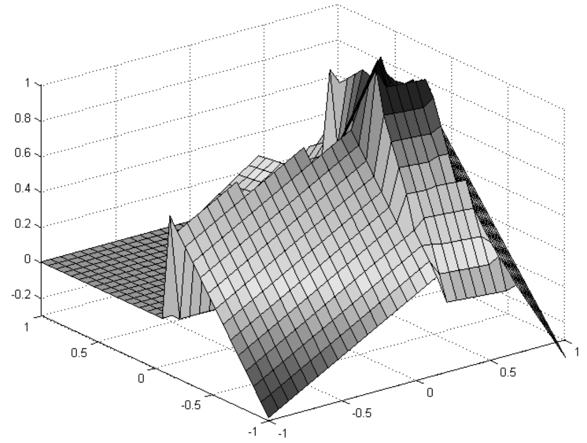

Figure 2.12: ISAT approximation with an error tolerance of 0.5. Due to the high error tolerance, the approximation is very coarse with 12 linear regions.

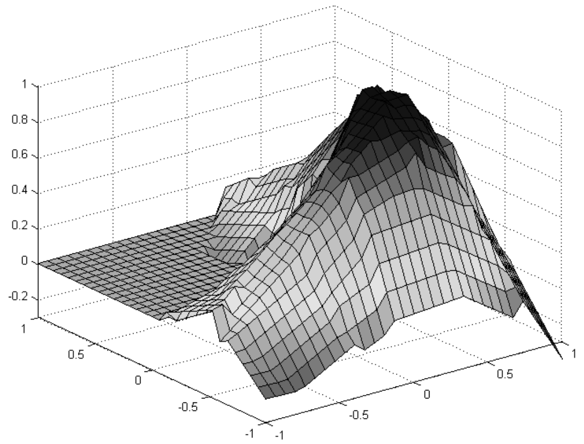

Figure 2.13: ISAT approximation with an error tolerance of 0.1. The approximation is more refined with 48 linear regions.

function approximation error and storage requirements. For comparison, the same function approximation was made with an artificial neural network. The neural network has two layers with a linear output layer of 1 neuron and a tangent function layer with 4 neurons. The neural network was generated and optimized using MATLAB's neural network toolbox. The neural network was trained with the same data that produced the ISAT database (see Figure 2.15). The approximation deviates significantly from the original function shown in Figure 2.11. Some of the key missing features are the non-continuously differentiable points, a quadrant that is exactly linear, and shape of the peak. A neural network is basically a nonlinear function with parameters that are optimized to fit a desired function. The neural network can approximate a wide range of nonlinear functions. However, some expertise is required to determine the number of layers, number of neurons in each layer, training data

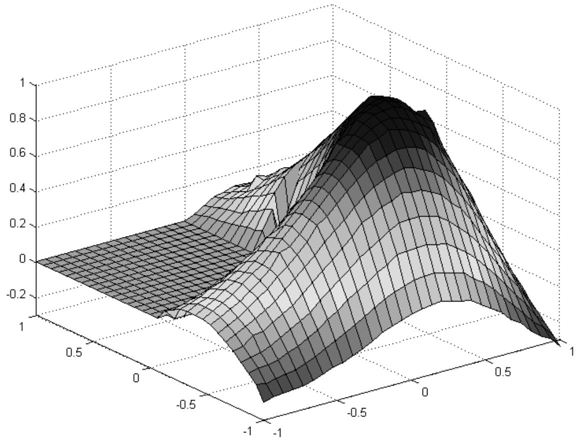

Figure 2.14: ISAT approximation with an error tolerance of 0.01. The approximation includes 206 linear regions and ISAT closely approximates the original eigenfunction.

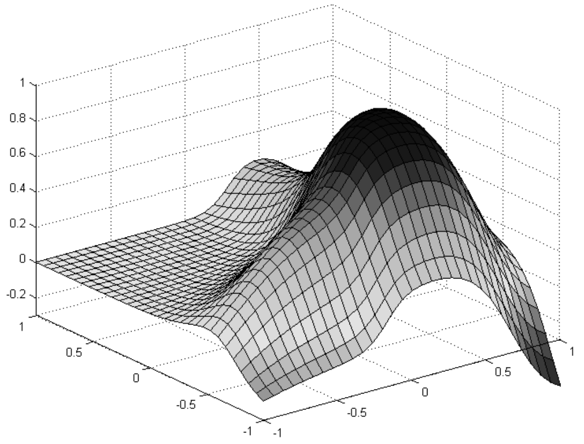

Figure 2.15: Neural network approximation to the eigenfunction.

set, and optimization to fit the nonlinear function. In addition, there is no error control to limit the amount of approximation error. The approximation error is determined by the structure and training of the neural network.

ISAT, as opposed to neural networks, uses linear regions to fit a desired function. Also, ISAT has direct control over the error tolerance, which is the most important tuning parameter for nonlinear function approximation algorithms. Other advantages of ISAT over neural networks are that no global optimization step is required to build the database. When ISAT encounters data outside of the training set, it either expands an existing linear region or adds a new linear region. The creation of new linear regions is determined by the error tolerance control. Also, ISAT can approximate functions that are not continuous or continuously differentiable.

2.5 Summary and Conclusions

This chapter outlines a new storage and retrieval algorithm for nonlinear functions. Although originally developed to reduce the computational burden of DNS in turbulent combustion, the algorithm shows promise as a general nonlinear function approximator. In this chapter, a description of the algorithm has been developed in a way that does not restrict the application of ISAT to one particular area. Although many of the details of the ISAT algorithm are presented in other papers, a more thorough explanation of the algorithm is given to clarify some of the details. In addition to reporting the details of ISAT, new features have been developed. Many functions do not produce an exact sensitivity. In the case when the sensitivity is not available, a statistical approximation is attempted. The statistical approximation is determined by collecting previous calculations close to the point of interest. This feature also identifies when insufficient data exist to provide an accurate sensitivity. With a sensitivity approximation, any nonlinear function can be stored and retrieved with ISAT.

The eigenfuction of an L-shaped membrane was used as a test problem to demonstrate ISAT's capabilities compared to neural networks. In subsequent chapters, it will be shown how ISAT applies in storage and retrieval of open-loop and closed-loop simulations. Open-loop simulations refer to simulations without optimization of particular model paramters. Closed-loop calculations seek to optimize decision variables to meet an objective. These applications in process control are further examples of ISAT's capability as a general storage and retrieval technique.

One of ISAT's limitations is that storage costs scale with the square of the system size. To overcome this deficiency, model reduction is incorporated to decrease the model order. The next chapter is devoted to model reduction because the efficiency of storage and retrieval can be poor for large scale problems. Because large scale systems are typically necessary to accurately model real-world phenomena, model reduction is used to reduce the model size within an acceptable range for storage and retrieval.

Chapter 3

DAE Model Reduction

A major obstacle to more NMPC applications is the rapid and reliable solution of the optimization problem in real-time [68]. To overcome this obstacle, several approaches have been suggested to reduce the computational overhead. In Chapter 4, the computational load is reduced by storing and retrieving solutions of DAE integrations of the model equations. In Chapter 5 the computation is shifted off-line by storing and retrieving optimal control solutions. Both techniques are much more efficient for lower dimensional problems. Model reduction generally does not significantly lower the computational cost of simulation and control. However, model reduction can enable off-line storage and retrieval for efficient on-line implementation. The model reduction strategies in this chapter are an important step in achieving computationally feasible model based control solutions.

DAE model reduction has traditionally been an *a posteriori* approach. Training simulations determine an acceptable reduced order model that may or may not be valid over the entire set of desired simulations. This chapter outlines a new *in situ* approach to adaptively determine the reduced model order during the desired simulations. One benefit of this new approach is more direct control over reduced model error. The model error, not the model order, becomes the principal tuning parameter. This change of tuning parameters is more intuitive because the technique automatically adjusts the model order to meet variable error tolerances.

3.1 Previous Work

DAEs consist of differential equations and algebraic equations. In the general form, the DAE problem is as follows

$$f_{DAE}(\dot{z},z,t) = 0 \tag{3.1}$$

where z is a vector of variables and t is a scalar. The DAE is nonlinear when the vector f is a nonlinear function of the $\dot{z}$, z, or t. In order for the problem to be a DAE, at least one of the coefficients of $\dot{z}$ must be zero. The DAE can be grouped into differential equations (f_{ODE}) and algebraic equations (f_{AE}). The variables are also divided into differential variables (x) and algebraic variables (y).

$$f_{ODE}(\dot{x},x,y,t) = 0 \tag{3.2}$$

$$f_{AE}(x,y,t) = 0 \tag{3.3}$$

Typically, the DAE equation residuals are time invariant and t can be eliminated from the general equation form. However, it is included in subsequent derivations for the sake of generality.

3.1.1 Reduction of Differential Equations

The main types of model reduction for nonlinear ODE models are projection methods, proper orthogonal decomposition (POD), balanced covariance matrices (BCM), perturbation methods, and model simplification [52]. Perturbation methods are useful for models where there is a large separation of time scales allowing the fast dynamics to be eliminated [89]. Even though there are many types of model reduction techniques, few are optimal in some sense. Two optimal nonlinear model reduction approaches are balanced empirical gramians [34] and POD [94]. Balanced empirical gramians were later found to be a special case of BCM [33]. The two step process, in POD and BCM, first computes a similarity transform from step or impulse simulations of the original model. Next, a Galerkin projection constructs the reduced states from a linear combination of the original states. The reduced set of states from BCM is optimal in capturing input to output dynamics of the original nonlinear system. POD is optimal in capturing input to state dynamics. During the Galerkin projection step, the physical meaning of the variables is lost but can be recovered by an inverse transform.

A Galerkin projection maps a full set of variables onto a reduced set of variables that make up the reduced model. For BCM, the Galerkin projection is a set of vectors that optimally captures the highest degree of input-output dynamics. For POD, the Galerkin projection is a set of orthogonal vectors but captures the highest degree of input-state dynamics. Each successive vector is the direction that maximizes the amount of variance in the model states while maintaining orthogonality to the previous directions. For the semi-explicit ODE model form

$$\dot{x}(t) = f(x(t)) \tag{3.4}$$

the Galerkin projection ($\tilde{P}$) is applied by defining a reduced set of variables

$$x(t) = \tilde{P}^T \bar{x}(t) + r(t) \tag{3.5}$$

where $r(t)$ is a state residual to account for the inaccuracy of the reduced model. The reduced model exactly represents the original model when the residual is retained.

$$\dot{\bar{x}}(t) = \tilde{P} f(\tilde{P}^T \bar{x}(t) + r(t)) + \tilde{P}\dot{r}(t) \tag{3.6}$$

At this point the residual and its derivative are typically set to zero and some of the system dynamics are necessarily lost due to the reduced order of the model.

$$\dot{\bar{x}}(t) = \tilde{P} f(\tilde{P}^T \bar{x}(t)) \tag{3.7}$$

Two techniques for obtaining a Galerkin projection are described in sections 3.1.1.1 and 3.1.1.2. POD is optimal in capturing input-state effects while BCM is optimal for input-output effects.

3.1.1.1 Proper Orthogonal Decomposition

POD is performed by analyzing the variance among the system states. This is accomplished by decomposing the covariance matrix of the states into eigenvectors and eigenvalues. The eigenvectors associated with the m largest eigenvalues become the pricipal directions in the reduced model. Here m is the order of the reduced model and n is the order of the original model. The Galerkin projection consists of the similarity transform (T) and a projection matrix (P). The projection matrix consists of the top m rows of an identity matrix. For POD, the similarity transform (T) is the transpose of the eigenvector matrix.

$$\tilde{P} = PT \tag{3.8}$$

The Galerkin projection takes a linear combination of states to form a reduced set. The model can either be reduced through truncation or residualization [34]. Truncation assumes that the transformed states corresponding to the lowest $(n-m)$ eigenvalues are constant (see Equation 3.9). Truncated reduced models perform better than residualized models with high frequency perturbations. One disadvantage is that there is usually some steady state offset.

$$\begin{bmatrix} \dot{\bar{x}}_1 \\ \vdots \\ \dot{\bar{x}}_m \\ \dot{\bar{x}}_{m+1} \\ \vdots \\ \dot{\bar{x}}_n \end{bmatrix} = \begin{bmatrix} \bar{f}_1(\bar{x},u) \\ \vdots \\ \bar{f}_m(\bar{x},u) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.9}$$

Residualization assumes that the derivates of the transformed states corresponding to the lowest $(n-m)$ eigenvalues are constant (see Equation 3.10). Residualized reduced models have no steady state offset, but perform worse than truncation for higher frequency responses. Residualization is often not desireable because the reduced model is a DAE of the same order as the original ODE.

$$\begin{bmatrix} \dot{x}_1 \\ \vdots \\ \dot{x}_m \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} \bar{f}_1(\bar{x}, u) \\ \vdots \\ \bar{f}_m(\bar{x}, u) \\ \bar{f}_{m+1}(\bar{x}, u) \\ \vdots \\ \bar{f}_n(\bar{x}, u) \end{bmatrix} \tag{3.10}$$

These definitions of truncation and residualization are consistent with those given for linear systems. For linear system truncation, the reduced states are set to zero because the variables are in deviation form. For linear system residualization, an explicit transformation of the reduced variables can be obtained.

3.1.1.2 Balanced Covariance Matrices

Nonlinear model reduction using balancing of covariance matrices has proven effective for sets of ODEs [33]. This method reduces the nonlinear model to a variable subspace that captures the most important dynamics of input-output behavior. One system studied by Hahn and Edgar [**?**] is a binary distillation column with the reflux ratio (u) as the manipulated variable and distillate composition (x_1) as the controlled variable. The simulated column contains 30 trays, a reboiler, and condenser. The 32 states are the compositions of the liquid at each stage. The ODE model is placed in the general nonlinear form.

$$\dot{x} = f(x, u) \tag{3.11}$$

$$y = h(x) \tag{3.12}$$

A similarity transform (T) is computed from the balancing of empirical gramians. The transformed variables are in order from most important to the least important for input/output behavior. The transformed system is shown in Equation 3.13. The Galerkin projection ($\tilde{P}$) is a combination of the similarity transform (T) and a projection matrix (P). The projection matrix consists of the top m rows of an identity matrix, where m is the order of the reduced model.

$$\dot{\bar{x}} = PTf(T^{-1}P^T\bar{x}, u) \tag{3.13}$$

$$y = h(T^{-1}P^T\bar{x}) \tag{3.14}$$

The reduced model is often written in a more concise form in terms of the reduced variables ($\dot{\bar{x}}$)

$$\dot{\bar{x}} = \bar{f}(\bar{x}, u) \tag{3.15}$$

$$y = \bar{h}(\bar{x}, u) \tag{3.16}$$

Hahn and Edgar [33] showed that a reduced system with 3 transformed variables shows excellent agreement with the full 32 state model on step tests. For example, the first transformed state is shown as a linear combination of the original 32 states.

$$\bar{x}_1 = \begin{bmatrix} 9.7 & 4.0 & 3.4 & \cdots & 0.08 & 0.07 & 0.24 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{30} \\ x_{31} \\ x_{32} \end{bmatrix} \tag{3.17}$$

As a physical interpretation, the relative importance of the 1st-3rd states (reflux drum composition and top rectification stages composition) on input/output behavior is much greater than the 30th-32nd states (lowest bottoms stages and reboiler composition). The relative weighting of stages 1-31 monotonically decreases until stage 32 where there is a slight increase from 0.07 to 0.24. This increase from stage 31 (sump) to stage 32 (the reboiler) can be attributed to the reboiler vapor that drives the separation. The similarity transform from the balancing of empirical gramians confirms that most of the important dynamics for control are found in the states at the top of the column. With the transformation, the nonlinear system can be reduced by truncation or residualization. By reducing the number of dynamic variables through truncation, the computational requirements are reduced by $\approx 40\%$. Equation 3.18 shows the form of the 3 state model by truncation.

$$\begin{bmatrix} \dot{\bar{x}}_1 \\ \dot{\bar{x}}_2 \\ \dot{\bar{x}}_3 \\ \dot{\bar{x}}_4 \\ \vdots \\ \dot{\bar{x}}_{32} \end{bmatrix} = \begin{bmatrix} \bar{f}_1(\bar{x}, u) \\ \bar{f}_2(\bar{x}, u) \\ \bar{f}_3(\bar{x}, u) \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{3.18}$$

The dyanamic response of this reduced system is shown in the subsequent section. It is also compared to a truncated model by POD and linearization.

3.1.1.3 Example Comparison of POD, BCM, and Linearization

POD, BCM, and linearization are compared for the 32 state binary distillation column model. In the first comparison, trucated models generated with POD and BCM are compared in an open-loop step test. The step test is generated by simulating a decrease in the reflux ratio, thereby lowering the purity of A in the distillate (see Figure 3.1). The response of the distillate concentration is traced for 120 minutes. BCM and POD perform equivalently for the 3 state reduced model. Because truncation was performed, instead of residualization, there is a slight steady state offset.

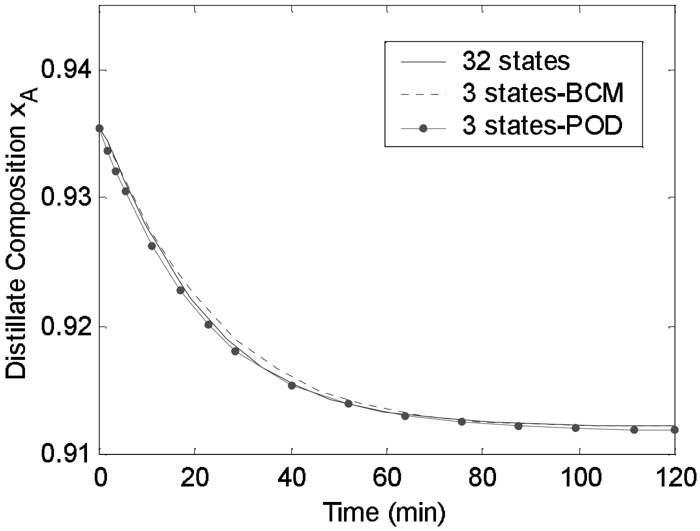

Figure 3.1: A step decrease in reflux ratio produces a corresponding decrease in the distillate composition. A nonlinear 32 state model, a 3 state BCM reduced model, and a 3 state POD model are shown.

Model simplication may include linearization. However, depending on the degree of nonlinearity and the state deviation from the original linearized values, a linear model may not capture the true dynamics. To illustrate this point, the 1 state truncated POD and BCM models are compared with a linearized model of 32 states (see Figure 3.2). Interestingly, both 1 state

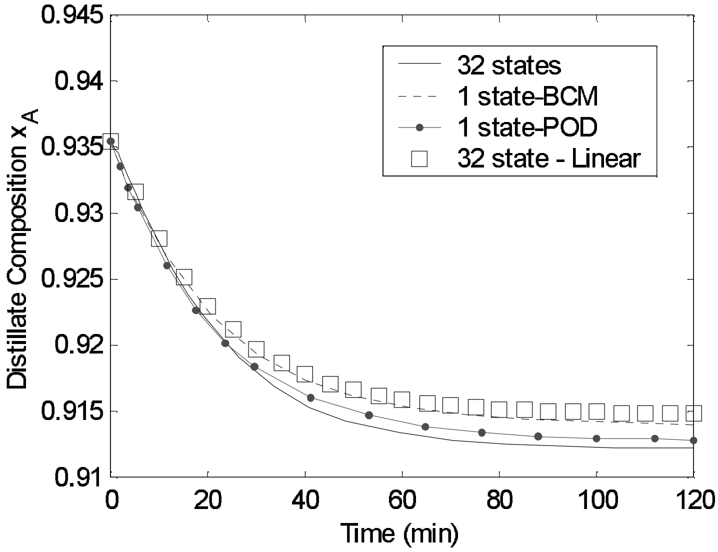

Figure 3.2: The same dynamic response as Figure 3.1 is shown in this plot. Here a nonlinear 32 state model, 1 state BCM and POD models, and 32 state linearized model are compared.

reduced models outperform the 32 state linearized model in dynamic response and steady state offset. This demonstrates the effectiveness of nonlinear model reduction compared to another model simplication strategy. Because POD and BCM are optimal in two unique ways, they will always outperform, with respect to their objectives, all other reduced or simplified models with the same or lower order.

3.1.2 Reduction of Algebraic Equations

None of the previously mentioned model reduction techniques can reduce the number of algebraic equations in DAE models. Because algebraic equations usually greatly outnumber the differential equations, reduction of the differential equations often does little to reduce the overall order (and also computational time) of a DAE model. Some attempts have been made to reduce the algebraic equations in a DAE model. These approaches search for an optimal precedence order and partitioning of the algebraic equations and variables. Obtaining a precedence order and partitioning can be done using a manual directed graph (digraph) as a graphical approach, using matrix methods to produce a block diagonal lower matrix [82], or through tearing [18]. These techniques attempt to maximize the number of algebraic variables that can be solved explicitly. However, this problem is NP complete, meaning that all possible combinations of variables must be attempted to find a maximum set of explicit equations [20]. Another technique for order reduction is through relaxation of the algebraic states [58]. Relaxation directly generates a Gaussian elimination scheme when the algebraic equations are linear or made linear.

3.1.2.1 Pairing Variables to Equations

Variables and equations are paired by rearranging the sparsity matrix to a maximum transversal. All variables are paired to equations when problem is completely specified (no degrees of freedom) and the maximum transversal is a zero-free diagonal [25]. The sparsity matrix (J), also known as the incidence matrix, is generated by identifying the variables that are contained in each equation. The differential states are generally specified for an initial value problem (IVP). The remaining variables are the differential state derivatives and the algebraic variables.

$$J_{ij} = \begin{cases} 1 & \text{if}\, y_j \,\text{or}\, \dot{x}_j \,\text{appears in}\, f_i \\ 0 & \text{otherwise} \end{cases} \tag{3.19}$$

Computing a zero-free diagonal involves changing the equation or variable orders. The order of the row and column are then matched to give a variable/equation pairing.

3.1.2.2 Application to Large Scale Problems

Another consideration relevant to large scale DAE systems is the computational time that is required for the analysis. In this paper, n and τ are the order of the matrix and the number of non-zeros, respectively. The maximum transversal algorithm has a worst case bound of $O(n\tau)$ although typical examples are more like $O(n) + O(\tau)$ [25]. The lower block triangular algorithm also exhibits excellent scaling for large problems with an upper bound of $O(n) + O(\tau)$ [26].

3.1.2.3 Example: Flowsheet Model Reduction

Flowsheet models typically consist of many individual models linked together by streams. If the model will be used in plant-wide control, model reduction is desirable to reduce the size of the model. In this example, there is a tank containing equal molar proportions of five liquid hydrocarbons. The stream exiting the tank is split into two with a splitter valve. The first stream is mixed with another feed stream of hydrocarbons, and the second stream passes through a heat exchanger. Overall the model includes 12 differential equations and 217 algebraic equations (see Figure 3.3). In this example, an

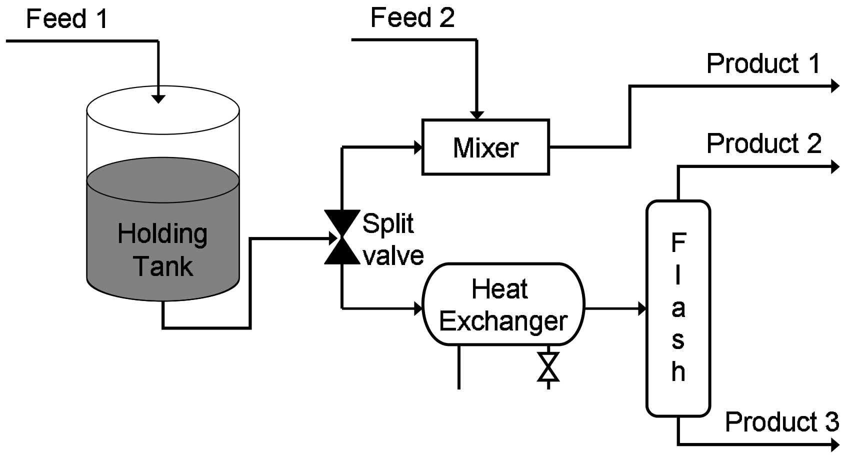

Figure 3.3: Flowsheet model involving equal molar feed streams of butane, pentane, hexane, heptane, and octane at 300 K and 1 ATM. The model has 229 variables with 12 ODEs and 217 AEs.

analysis of the algebraic equations was performed to determine independent sets of variables and equations and the solution order. The algebraic equations decomposed into 202 independent sets. There was one implicit set of 16 algebraic equations from the flash column. With the exception of this set, each of the algebraic variables can be solved independently with a paired algebraic equation by following the precedence ordering. Explicitly transforming the algebraic equations manually or with ISAT reduces the model order from 229 to 28. Details of the decomposition are given in Appendix C.

3.1.2.4 Explicit Transformation of Implicit Sets

Once the successive independent sets are identified, a storage and retrieval technique can be used to store and retrieve solutions to the groups of algebraic variables and equations. ISAT is a storage and retrieval algorithm that builds piecewise linear regions of the solution. ISAT controls the prediction error by defining an ellipsoid of accuracy (EOA) that specifies a local region about which the linear approximation is valid. If a query point is accessed outside of the EOA, the linear prediction error is exactly calculated. When the prediction error is above a specified tolerance, a new linear region is added to the database. Adding linear regions involves a sensitivity calculation and a conservative estimate of the new EOA. ISAT is used here to store and retrieve solutions to implicit blocks of variables and equations. An external statistical approximation of the sensitivity through linear regression is given as an alternative when an internal sensitivity calculation is not available.

3.2 Adaptive Reduction of DAEs

The total degrees of freedom (DOF) are equal to the order of the DAE model. The *dynamic* degrees of freedom (DDOF) are defined as the minimum order of a reduced model that shows good agreement with the full model. The DDOF are the underlying combination of variables that control the dynamics of the process. A more precise definition of the DDOF is the minimum order of a reduced order model (ROM) that meets accuracy criteria.

$$|x_{ROM} - x| \leq \epsilon_{tol} \tag{3.20}$$

The proposed adaptive reduction of DAEs iteratively adjusts the order of the ROM to meet the accuracy criteria. The adaptive approach consists of 3 successive steps applicable to any DAE of index-1 or index-2. The approach may also be applicable to higher index DAEs, although this idea has not been explored. With a few exceptions, many of the models encountered in practice are index-1 DAEs. In addition, a variety of techniques have been developed to transform higher index DAEs to lower index form, but that work is beyond the scope of this chapter.

1. Reduction of the differential equations
2. Partitioning and precedence ordering of the algebraic equations
3. Explicit transformation of algebraic/differential equations

As an index-1 DAE, the equations can be divided into differential and algebraic sets of equations and variables. Any variable that is present in differential form is classified a differential variable. Likewise, equations that are paired to differential variables in step 2 are classified as differential equations. Under this definition algebraic equations may contain differential variables. Each step is an extension of existing approaches, modified to automatically control the ROM error.

3.2.1 Step 1: Reduction of ODEs

To adaptively reduce the order of the differential equations, a measure of the reduced model accuracy must be introduced. When performing non-adaptive model reduction, the training simulations are performed, similarity transforms are generated, and singular values can be investigated to determine an acceptable number of states for the reduced model. However, when the training data set does not cover the entire nonlinear region of interest, the singular values may be a poor indication of reduced model accuracy outside of the training domain. One possible solution is to directly solve the full model and reduced model at various checkpoints to determine the accuracy of the reduced model. Another option that avoids the periodic solution of the full model is to control the equation residuals. For linear systems at steady state, the equation residuals and variable residuals are exactly related. A linearized model is used to predict the variable residuals from the equation residuals.

3.2.1.1 Predicting Variable Error

Ideally, one would like to adjust the order of the reduced model to control the variable errors directly. Barring simultaneous solution of the reduced and full order model, the variable error cannot be directly calculated for nonlinear systems. A new approach is to estimate the variable residual ($r(t)$) from the equation residual ($R(t)$).

$$\dot{\bar{x}}(t) = \tilde{P} f(\tilde{P}^T \bar{x}(t)) + R(t) \tag{3.21}$$

When the system is linear, the equation residuals are related to the variable residuals by the state matrix (A).

$$\dot{x}(t) = Ax(t) \tag{3.22}$$

with

$$x(t) = \tilde{P}^T \bar{x}(t) + r(t) \tag{3.23}$$

$$\dot{x}(t) = \tilde{P}^T \dot{\bar{x}}(t) + \dot{r}(t) \tag{3.24}$$

the linear reduced model becomes

$$\tilde{P}^T \dot{\bar{x}}(t) = A\left(\tilde{P}^T \bar{x}(t)\right) + Ar(t) - \dot{r}(t) \tag{3.25}$$

The equation residual and variable residual are related to each other by the state matrix and the variable residual derivative.

$$R(t) = Ar(t) - \dot{r}(t) \tag{3.26}$$

By assuming that the variable residual is locally constant, the variable residual derivative term can be ignored and an estimate of the variable residual can be obtained.

$$\hat{r}(t) = A^{-1}\left(R(t)\right) \tag{3.27}$$

By linearizing the nonlinear model, an estimate of the variable residuals can be obtained from the equation residuals. The predictive capability of this relation for nonlinear models depends on the severity of nonlinearity and closeness to the point of linearization.

3.2.1.2 Correcting Variable Error

A semi-explicit ODE model is a restricted form of the more general open equation format.

$$f(\dot{x}, x) = 0 \tag{3.28}$$

Applying the Galerkin projection to the open equation format changes the solution procedure. By reducing the number of variables and maintaining the same number of equations, extra degrees of freedom arise. Physically, this is the result of giving up some of the least important dynamic degrees of freedom. The reduced order model is solved by minimizing the residuals instead of finding equation roots.

$$R = f(\tilde{P}^T \dot{\bar{x}}, \tilde{P}^T \bar{x}) \tag{3.29}$$

Once a minimized residual solution is found, a variable correction can be applied from the predicted variable error (see Equation 3.27). The correction relies on a linearized version of the ODE portion of the DAE model. The corrected ROM is the sum of the ROM and the linear correction term.

$$x^c_{ROM} = x_{ROM} + A^{-1}R(t) \tag{3.30}$$

The correction is derived under the assumption that the linear model is locally accurate and that the fast dynamics have decayed. The correction may not perform well when either of these assumptions is not valid.

3.2.1.3 Controlling Variable Error

The minimized equation residuals will generally be small for a good ROM. As the order of the ROM is decreased, the equation residuals will generally increase. The differential equation model reduction approach is made adaptive by increasing or decreasing the number of states of the ROM to meet the required variable tolerances. This approach also involves a periodic update of the Galerkin projection. The rank of the projection matrix (P) is adjusted to meet the desired order of the ROM. The similarity transform (T) is periodically recomputed as more training simulations become available. Before simulations are added to the training set, the order of the ROM is equal to full order model. As the simulations proceed, the ROM order is iteratively decreased until the training set is mature and the true number of DDOF are determined.

3.2.2 Step 2: Partitioning and Precedence Ordering

The method proposed in this work differs from previous work by analyzing a dependency matrix M_D instead of the incidence matrix J [26]. It will be shown that M_D can reveal more information about variable dependencies. The dependency matrix M_D is derived by first linearizing the DAE.

$$A\dot{x}' + Bx' + Cy' + \alpha t' = 0 \tag{3.31}$$

$$Dx' + Ey' + \beta t' = 0 \tag{3.32}$$

The prime indicates deviation from reference values. A, B, C, D, and E are coefficient matrices and α and β are coefficient vectors. The reference values are selected to give non-zero coefficients for the deviation variables. Because the selection of reference values is arbitrary, the coefficients can be arbitrarily selected to be 1 if the equation contains the variable and 0 otherwise. In this case the matrix E is equivalent to the incidence matrix J. Rearranging and combining the linear differential and algebraic equations results in the following matrix form:

$$\begin{bmatrix} A & C \\ 0 & E \end{bmatrix} \begin{bmatrix} \dot{x}' \\ y' \end{bmatrix} = - \begin{bmatrix} B & \alpha \\ D & \beta \end{bmatrix} \begin{bmatrix} x' \\ t' \end{bmatrix} \tag{3.33}$$

The dependency matrix M_D reveals the solution dependencies among the linearized equations.

$$M_D = \begin{bmatrix} A & C \\ 0 & E \end{bmatrix}^{-1} \tag{3.34}$$

The variable dependency information in M_D can be illustrated by a linear system of $Ax = b$. When A is invertible, the solution to x is $A^{-1}b$. Each element of the vector x is computed from the corresponding row of A_{ij}^{-1} and the vector b.

$$x_i = \sum_j A_{ij}^{-1} b_j \tag{3.35}$$

However, the solution to x_i is independent of b_j if $A_{ij}^{-1} = 0 \ \forall \ j \neq i$. If x_i is independent of b_j then it is also independent of equation j. The dependencies in the linear system also apply to the corresponding nonlinear system. Therefore, linearizing the DAE reveals the structure of the nonlinear system dependencies.

The matrix M_D can be converted to lower triangular block diagonal form with Tarjan's algorithm [82]. Each block along the diagonal is a set of algebraic equations that require a simultaneous solution. The reduction of algebraic equations occurs by explicitly solving for independent groups of equations. Test cases with moderate sized DAE systems (< 300 states) show that many of the equations included in the implicit set can be transformed for explicit calculation. Once an algebraic variable is explicitly calculated, it can be removed from the model as a variable that the solver must calculate. Explicit approximations to implicit solutions can be attempted to further reduce the DAE order as Bosley did for batch distillation [19].

3.2.2.1 Example: Binary Distillation

A binary distillation column model, described in Appendix A, is employed to show a practical application of DAE model reduction. In this case the model is reduced to a set of ODEs, although the complete removal of all algebraic equations is not always possible. The DAE model has 52 differential equations and 233 algebraic equations. The independent variables are shown in Table 3.1. During the linearization step, the reference values are selected to give non-zero coefficients for the deviation variables. Since the reference values are arbitrary, the non-zero coefficients are shown by X if the equation

Table 3.1: Variables at each stage of the distillation column

Differential variables		
Symbol	Description	Units
$\dot{x}_A$	Liquid mole fraction	none
$\dot{h}$	Specific enthalpy	$\frac{J}{mol}$
Algebraic variables		
Symbol	Description	Units
y_A	Vapor mole fraction	none
x_L	Liquid mole fraction	none
T	Temperature	K
$\dot{n}_V$	Vapor molar flow rate	$\frac{mol}{sec}$
$\dot{n}_L$	Liquid molar flow rate	$\frac{mol}{sec}$
h_V	Specific vapor enthalpy	$\frac{J}{mol}$
h_L	Specific liquid enthalpy	$\frac{J}{mol}$
P_i^{sat}	Saturation pressure of compound j	Pa

contains the variable and 0 otherwise.

$$M\begin{bmatrix}\dot{x}\\ y\end{bmatrix} \rightarrow \left[\begin{array}{cc|cccccccc} X & 0 & X & 0 & 0 & X & 0 & 0 & 0 & 0 \\ 0 & X & 0 & 0 & 0 & X & X & X & 0 & 0 \\ \hline 0 & 0 & X & 0 & 0 & 0 & 0 & 0 & X & 0 \\ 0 & 0 & 0 & X & 0 & 0 & X & X & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & X & X \\ 0 & 0 & 0 & X & 0 & X & 0 & 0 & 0 & 0 \\ 0 & 0 & X & 0 & X & 0 & X & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & X & 0 & 0 & X & 0 & 0 \\ 0 & 0 & 0 & 0 & X & 0 & 0 & 0 & X & 0 \\ 0 & 0 & 0 & 0 & X & 0 & 0 & 0 & 0 & X \end{array}\right]\left[\begin{array}{c} \dot{x}_A \\ \dot{h} \\ \hline y_A \\ x_L \\ T \\ \dot{n}_V \text{ or } \dot{n}_L \\ h_V \\ h_L \\ P_A^{sat} \\ P_B^{sat} \end{array}\right] \tag{3.36}$$

The non-zero values of M_D show the dependencies between the variables and equations. The non-zero values of M_D in lower triangular block diagonal form are shown below with the corresponding variable order.

$$M_D = \left[\begin{array}{cccccccc|cc} X & X & X & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ X & X & X & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ X & X & X & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ X & X & X & X & 0 & 0 & 0 & 0 & 0 & 0 \\ X & X & X & 0 & X & 0 & 0 & 0 & 0 & 0 \\ X & X & X & 0 & X & X & 0 & 0 & 0 & 0 \\ X & X & X & X & X & X & X & 0 & 0 & 0 \\ X & X & X & X & X & X & X & X & 0 & 0 \\ \hline X & X & X & X & X & X & X & X & X & 0 \\ X & X & X & X & X & X & X & X & 0 & X \end{array}\right] \tag{3.37}$$

$$\left[\begin{array}{c} y \\ \dot{x} \end{array}\right] = \left[\begin{array}{c} T \\ P_A^{sat} \\ P_B^{sat} \\ h_L \\ y_A \\ h_V \\ x_L \\ \dot{n}_V \text{ or } \dot{n}_L \\ \hline \dot{x}_A \\ \dot{h} \end{array}\right] \tag{3.38}$$

The first three rows indicate that T, P_A^{sat}, and P_B^{sat} must be solved simultaneously since the corresponding equations form one block. The equations for P_A^{sat} and P_B^{sat} can be explicitly substituted into the bubble point temperature equation.

$$P = x_A P_A^{sat}(T) + (1 - x_A) P_B^{sat}(T) \tag{3.39}$$

For the cyclohexane / heptane binary mixtures, an explicit temperature solution is approximated by a second order polynomial in composition.

$$T = c_1 + c_2 x_A + c_3 x_A^2 \tag{3.40}$$

The vector c was calculated with a least squares fit with data from the sets $x_A \in \left[\begin{array}{cc} 0 & 1 \end{array} \right]$ to be $c = \left[\begin{array}{ccc} 385.42 & -21.57 & 3.736 \end{array} \right]^T$. The polynomial fit has a mean sample error of 0.012 K and a maximum sample error of 0.04 K.

The molar flow rates form the last block along the diagonal. Since the molar flow rate equations are linear, they can be solved explicitly. After solving the flow rates the dependency matrix indicates that the differential equation variables can now be solved explicitly. If extraneous algebraic equations were present in the model, they could be identified at this point since the equations for $\dot{x}_A$ and $\dot{h}$ have no further dependencies. By explicitly solving all of the algebraic equations, the model is in an ODE form. In this form, nonlinear model reduction techniques can be applied to further reduce the number of differential states.

As an ODE, the distillation column model is available for further model reduction through BCM or POD. POD was chosen for this example and the number of differential states was reduced to 26. Figure 3.4 shows the bottoms composition after a 5% increase in reboiler duty. The ODE model with 52 states approximates the 285 state DAE very well. The ODE model with 26 states also approximates the DAE model well but with a larger offset in the steady-state value of composition. ODE models with fewer than 20 states performed poorly, indicating that there are at least 20 dynamic degrees of freedom in the binary distillation column model.

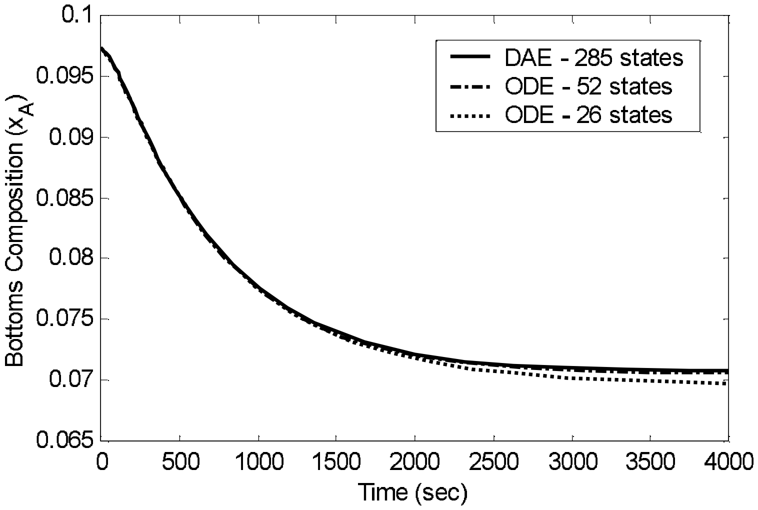

Figure 3.4: 5% Step Change in Reboiler Duty

3.2.3 Step 3: Explicit Transformation of Algebraic/Differential Equations

Because the implicit solution of a large number of algebraic equations is computationally expensive, variables that can be solved explicitly are removed from the set y [37]. However, some algebraic variables cannot be explicitly solved or solved independently of other variables. The following techniques can be used to automatically identify independent sets of variables and equations. The sets that require an implicit solution are explicitly approximated with ISAT. The process is called adaptive because the ISAT database adaptively adds records to the database to control the approximation error.

Chapter 4 introduces the storage and retrieval of differential equations with ISAT. Once the ISAT database is mature, meaning that mostly retrievals occur, the differential equations are rarely solved. Explicit transformation of the differential equations refers to a mature database, not a mathematical reformulation of the equations.

3.3 PDE Example: Unsteady Heat Conduction

A simple partial differential equation (PDE) example demonstrates the reduction approach with the number of reduced states controlled by the equation residuals. The example is an unsteady 1-D heat conduction problem using physical properties of aluminum (see Figure 3.5). The heat transfer dynamics are modeled with one PDE.

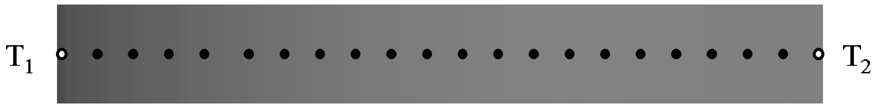

Figure 3.5: Graphical representation of a 1-D simulation of heat transfer in a 1.0 m thich aluminum slab. The PDE is spatially discretized to a set of ODEs, one for every interior node.

$$\rho c \frac{\partial T}{\partial t} = \frac{\partial}{\partial x}\left(k \frac{\partial T}{\partial x}\right) \tag{3.41}$$

The PDE is spatially discretized with the finite element approach to give a set of ODEs. The total slab thickness is 1.0 m and node points were placed every ≈4.8 cm for a total of 20 equally spaced interior nodes. The temperature of the nodes on the boundaries are specified by the boundary conditions.

$$\rho(T_i)c(T_i)\frac{\partial T_i}{\partial t} = \frac{1}{\Delta x^2}\left[k_i^{hm}\left(T_{i+1} - T_i\right) - k_{i-1}^{hm}\left(T_i - T_{i-1}\right)\right] \tag{3.42}$$

$$k_i^{hm} = \left(\frac{2k(T_{i+1})k(T_i)}{k(T_{i+1}) + k(T_i)}\right) \tag{3.43}$$

An estimate of the thermal conductivity at each finite element interface (k_i^{hm} is obtained by the harmonic mean of the thermal conductivities at the node temperatures. The slab of aluminum is initially at an ambient temperature of 25°C. At the initial time the temperature on the left side of the slab is increased to 100°C. After 100 minutes the temperature profile is nearly equal to the steady state linear temperature profile. Figure 3.6 shows the various reduced order models in comparison with the full state model. Any reduced order model above 3 states fits the correct solution well. The 2 state model

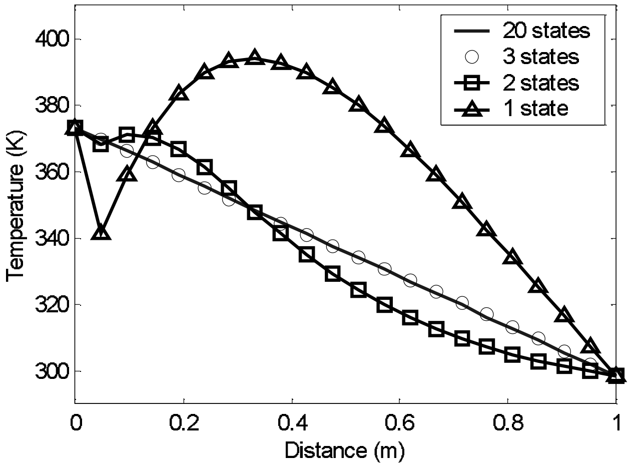

Figure 3.6: Full 20 state model compared with reduced order models of 3, 2, and 1 states using POD.

deviates from the correct solution as does the 1 state model, but more significantly. Singular values from POD, the 1-norm of the variable residuals, and 1-norm of the equation residuals are shown in Figure 3.7 on a semi-log plot. Singular values are generally used to determine the order of the reduced

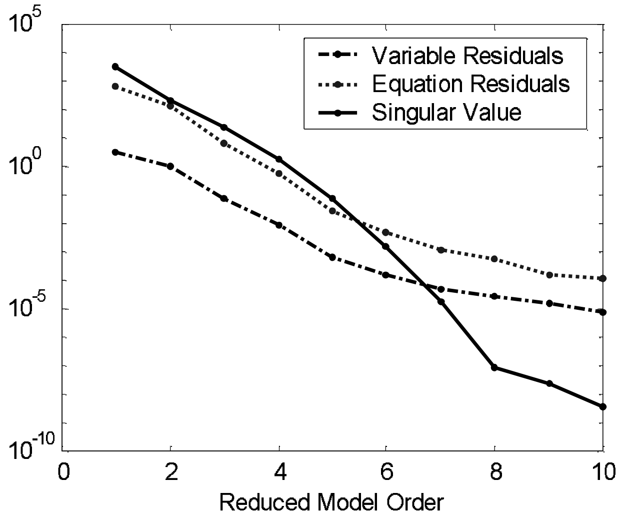

Figure 3.7: Equation residuals have a strong correlation to the variable residuals. The singular values also decrease with an increase in model order, but have limited ability to predict variable error.

model for POD in the *a posteriori* approach. However, singular values have little predictive capability as to the absolute accuracy of the model states. Equation residuals are more directly tied to variable error between the full order model states and reduced order model states. The variable residuals can be predicted by linearizing the model about the actual solution (see Figure 3.8). In this example, the finite element discretized differential equations are nearly linear. The model nonlinearities are in the temperature dependence of

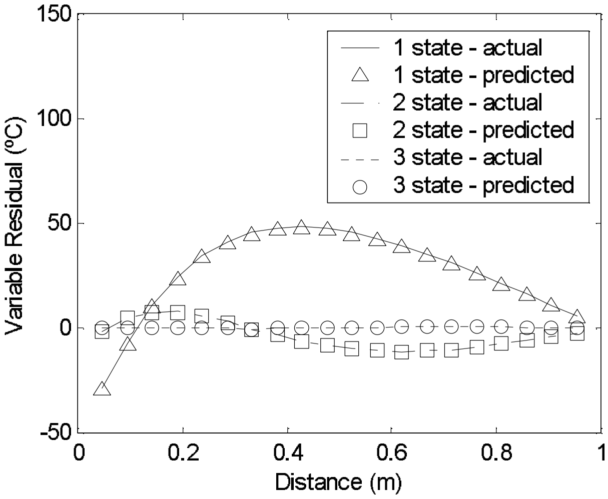

Figure 3.8: The predicted and actual variable errors are shown. The prediction is excellent for this example because the model is near steady state and nearly linear.

the aluminum thermal conductivity, heat capacity, and density. These properties do not change significantly in the temperature range of the simulation (25°C to 100°C). Because of near-linearity and proximity to steady-state, the equation residuals are excellent predictors of the variable residuals (see Figure 3.9). Further work needs to be done to validate the variable residual prediction with models of varying degrees of nonlinearity and for simulations with fast dynamics that are not captured by the reduced model.

3.4 Index-2 DAE Example: Large-Scale Distillation Model

The PDE example of Section 3.3 is a simple example to demonstrate the proposed model reduction approach. A more complex example is given in

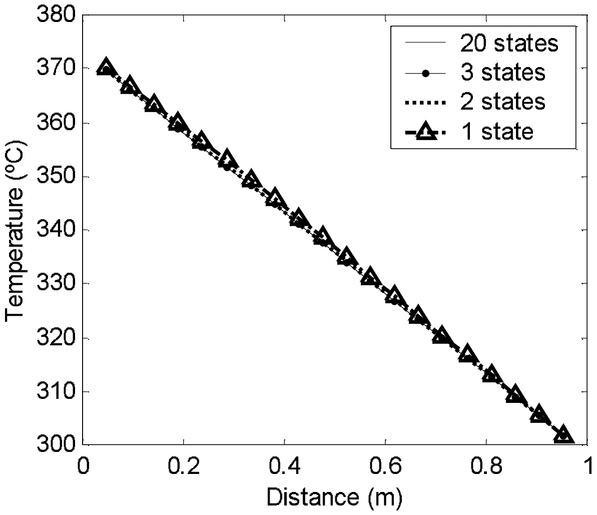

Figure 3.9: Using the equation residuals to predict variable error, the reduced models show improved accuracy. In this case, the 1 state reduced order model is sufficiently accurate. Without the correction, a 3 state model is required for similar accuracy (see Figure 3.6).

this section to demonstrate model reduction of a large-scale DAE. The model developed for this example is more indicative of the complexity of a real simulation and control problem for distillation. A diagram of the model is shown in Figure 3.10. The model is developed for a column of 22 trays, a condenser,

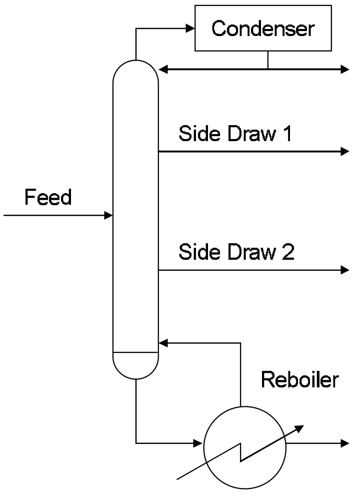

Figure 3.10: The multi-component distillation column model consists of 22 trays, multiple sidedraws, a condenser, a reboiler, and a sump at the column base.

a sump and reboiler, 2 sidedraws, and 1 feed stream. A linear paraffin mixture between C_7H_{16} and $C_{22}H_{46}$ is modeled by lumping successive sets of 3 hydrocarbons into 5 pseudo-components. The lumping procedure substantially reduces the model size by considering only 5 species (C_8H_{18}, $C_{11}H_{24}$, $C_{14}H_{30}$, $C_{17}H_{36}$, $C_{21}H_{44}$) that approximate the thermo-physical properties of the entire range. The simulated mixture is preheated and fed into the column at tray 12. Sidedraws are taken from trays 5 and 16. The species of interest for this study is $C_{11}H_{24}$ (approximates the $C_{10}H_{22}$ to $C_{12}H_{26}$ range), found in high concentration (>90% purity) in the sidedraw from tray 5 (Sidedraw 1).

Constructing a distillation column model from first principles is an effort intensive task that can require months of work. Models can also be constructed from process simulators, such as HYSYS, but the model equations are not exposed. The approach taken in this example was to develop an object-oriented simulator that automatically constructs the distillation model variables and equations from a subset of simpler models. This object-oriented approach is shown as a pyramid in Figure 3.11 with successive object layers. Moving up the pyramid indicates successively more complex models, formed

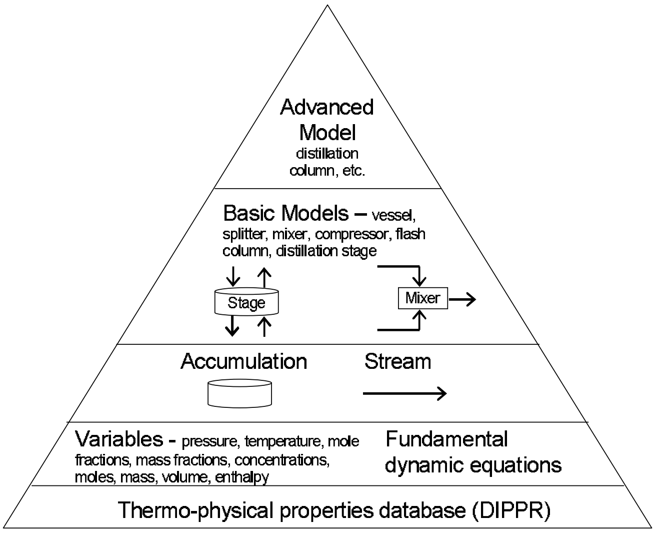
Advanced Model
distillation column, etc.
Basic Models – vessel, splitter, mixer, compressor, flash column, distillation stage
Stage
Mixer
Accumulation
Stream
Variables - pressure, temperature, mole fractions, mass fractions, concentrations, moles, mass, volume, enthalpy
Fundamental dynamic equations
Thermo-physical properties database (DIPPR)

Figure 3.11: The distillation column model is constructed from an object-oriented simulator. More advanced models are simply combinations of base level models connected by streams.

by combining the basic models (vessels, splitters, mixers, flash columns, distillation stages, etc.). The object-oriented modeling approach necessitates model equations in the open equation format ($0 = f(\dot{x}, x, u)$). For the distillation column in this study, the model is an index-2 DAE with 3250 states. The index of a DAE refers to the number of times the algebraic equations must be differentiated to transform the model into an ODE. Only a small subset of equations are index-2 with the majority being index-1. The index-2 equations are the 22 bubble point temperature equations, with 1 in each of the distillation stage sub-models. The model is reduced in three steps. The first step involves model reduction at the most basic modular level. Each stream (or accumulation) includes pressure, temperature, mole fractions, mass fractions, concentrations, molar flow rate (or moles), mass flow rate (or mass), volumetric flow rate (or volume), density, and enthalpy. Pressure, temperature, mole fractions, and molar flow rate (or moles) are used to uniquely specify the state of the mixture. All other variables are solved explicitly as a function of these variables and can be removed from the implicit set. Another reduction in variables is gained from the object-oriented framework. Instead of defining connection equations, two connecting streams can be merged into one stream object. The explicit transformation and stream merging reduces the model size from 3250 to 353 variables. The 353 variable model consists of 107 differential variables and 246 algebraic variables. The index-2 DAE is converted to index-1 form by differentiating the index-2 algebraic equations. The differentiated bubble point temperature equations are used to remove the temperature derivative in the energy balance. By removing the temperature derivatives, the index-2 equations become index-1 equations. In index-1 form, the algebraic equations are explicitly solved at each function call, thereby removing the 246 algebraic variables from the implicit set. The remaining differential variables are reduced with POD to 2, 5, or 8 ODEs. The reboiler heating rate is stepped by 25% from 2.0e7 $\frac{J}{min}$ to 2.5e7 $\frac{J}{min}$ at 10 minutes. The index-2 DAE of 353 variables is numerically integrated with DASPK and compared with the reduced order models (see Figure 3.12). The 2 state model shows

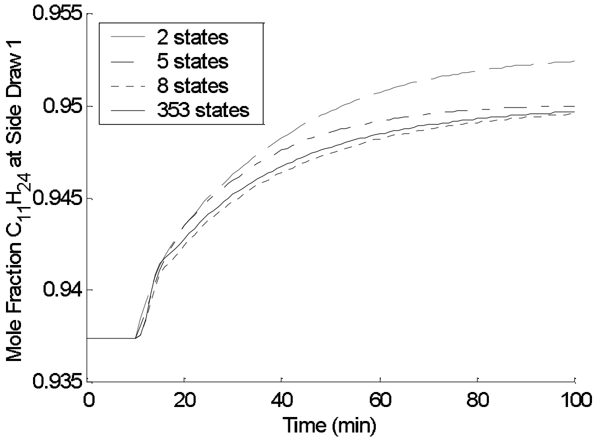

Figure 3.12: The reduced models are compared with the 353 state distillation model.

significant steady state offset, compared with the other reduced models. The 5 state model performs better in eliminating the offset, but the dynamic response deviates significantly. The 8 state model has excellent agreement with the full 353 state model in both dynamic response and gain. For this example, an 8 state model is recommended to approximate the full order model.

Even though the model is drastically reduced in size, no computational advantage is achieved. The 353 state model step test simulation required 41 CPU seconds on a 2 GHz Celeron processor, running FORTRAN. Based on LINPACK benchmarks for processor MFLOPS (million floating point operations per second), the step test required approximately 30 billion floating point operations. Each reduced model step response was generated in MATLAB, requiring approximately 31 billion floating point operations.

There are several reasons that the reduced model does not substantially reduce the computational burden. The primary reason is that all the equation residuals must still be computed to form the reduced order model equation residuals. The calculation of equation residuals requires 22 billion floating point operations for both the full and reduced models. This is an irreducible overhead, regardless of model size. With efficient DAE solvers, such as DASPK, no computational advantage is gained by solving the algebraic equations explicitly at each residual evaluation. An amount of computational overhead is also added in the index-2 to index-1 transformation.

Clearly, model reduction is not computationally advantageous for this example. Fortunately, there are a variety of other justifications for model reduction. One reason is the insight that model reduction provides. Even though the original model has thousands of variables, there are only 8 degrees of freedom that control the dynamic response. Another reason is for off-line storage and retrieval of control solutions for efficient on-line implementation. Off-line storage and retrieval presented in Chapters 4 and 5 is more efficient for smaller models, with an upper limit of about 100 states. Explicit moving horizon estimation of Chapter 6 scales quadratically with states so smaller models are more efficient, although typical examples have negligible computational expense. The model reduction strategies in this chapter are an important enabling step in achieving computationally feasible large-scale model-based control solutions.

3.5 Summary and Conclusions

Because many large scale DAE models consist mostly of algebraic equations, ODE model reduction techniques applied to DAE models are ineffective at significantly reducing the overall order of the model. An adaptive DAE model reduction technique is outlined in this chapter with special consideration for large scale models. The technique consists of three steps:

1. Adaptive POD reduces the number of differential states.
2. Algebraic states are partitioned into successive implicit sets of variables and equations by reconstructing the sparsity pattern into a lower triangular block form.
3. Explicit transformation of algebraic and differential equations.

Large scale models are often expressed in the open equation format. POD is applied to the open equation format by minimizing the equation residuals instead of finding roots. Once a minimized solution is found, the equation residuals provide an estimate of the variable accuracy. The differential variable accuracy is controlled by increasing or decreasing the order of the reduced model. In this way, POD is made adaptive while dynamically constructing the similarity transform. The estimate of the variable accuracy can also be used to improve the reduced model accuracy. In the 1-D unsteady heat conduction problem, it was shown that the correction reduced the DDOF from 3 states to 1 state. To reduce the algebraic variables, the variables and equations are restructured into successively independent sets. These independent sets are explicitly approximated with ISAT. ISAT directly controls the approximation error by expanding or adding piecewise linear regions. A flowsheet model example showed a reduction of algebraic variables by a factor of 8. A multi-component distillation column model is used todemonstrate reduction techniques on a large-scale index-2 DAE model. The model is reduced from 3250 states to 8 states with little loss of accuracy.

The explicit transformation of differential equations can be accomplished by ISAT by storing and retrieving integration solutions. This topic is further discussed in Chapter 4 with some numerical examples with ISAT. None of the examples in this chapter used ISAT although the applicable methods are discussed.

Chapter 4

Storage and Retrieval in Nonlinear Model Predictive Control

Simulation of physical processes described by differential-algebraic equations (DAEs) often involves hundreds of differential equations and thousands of algebraic equations. Storage and retrieval of these simulations may be desireable for a number of reasons. Some of these reasons include real-time optimization, real-time dynamic data reconciliation, computational reduction of complex simulations with DAE sub-models, and computational reduction of parameter sensitivity studies. Storing the simulations shifts the computational burden off-line for a more efficient on-line implementation.

This chapter focuses on a method to reduce the computational requirements of nonlinear model predictive control (NMPC) in real-time control applications. Nonlinear model identification is generally seen as a major obstacle to implementing NMPC. However, once an accurate nonlinear model is identified, the computational effort is often too great to implement the model in a real-time application. The approach in this paper is a two step process, model reduction followed by computational reduction. Model reduction is accomplished by computing balanced covariance matrices for the dynamic system. Computational reduction is accomplished by using the method of *in situ* adaptive tabulation (ISAT). ISAT was previously developed for computational reduction of turbulent flame direct numerical simulations and is extended to the sequential NMPC framework in this work. A case study is performed with a binary distillation column model with 32 states. By computing balanced covariance matrices and using ISAT, the computational speed is 85 times faster than the original NMPC while maintaining the accuracy of the nonlinear model. Because ISAT is a storage and retrieval method, it is compared to artificial neural networks in another case study, a dual CSTR model with 6 states. Open loop and closed loop step tests are performed to demonstrate the superior quality of ISAT in extrapolating outside of the training domain.

The three most significant obstacles to NMPC applications are nonlinear model development, state estimation, and rapid, reliable solution of the control algorithm in real time [68]. This chapter outlines an attempt to overcome the last two obstacles through a computational reduction technique formerly developed for turbulent combustion simulations [66]. ISAT stores and retrieves open loop simulations and thereby reduces the computational effort of integrating continuous dynamic first principles models. In direct multiple shooting implementations of NMPC [17] [83] open loop simulations are performed many times until an optimal trajectory of manipulated variables is found. Also, in sequential state estimation (dynamic data reconciliation) many open loop simulations are performed until an approximation to the unmeasured states is obtained [47]. Because the open loop simulations occupy a majority of the computational effort, ISAT has potential to greatly improve the speed of state estimation and dynamic optimization.

An important assumption for ISAT is that nearby integrations will likely be repeated. For storage and retrieval of a single step test or impulse response, very few integrations will likely be repeated and ISAT would likely show poor performance. In control applications, similar disturbances to the system can occur frequently or step changes to switch between product grades can happen regularly. The ISAT method is a framework for the estimator and controller to access data from previous calculations.

4.1 Direct Methods for Solving NMPC Problems

There are multiple methods for solving equivalent NMPC problems. Sequential, hybrid, and simultaneous formulations are three popular numerical techniques [17]. The sequential approach minimizes an objective function by manipulating the decision variables over a finite control and prediction horizon. The simultaneous approach accomplishes the same objective by manipulating both the decision variables and the state values at collocation nodes. The hybrid approach is a compromise between sequential and simultaneous approaches by allowing for the use of state of the art DAE solvers to control discretization. A summary of the three approaches is given by Binder [17] in Table 4.1. ISAT can be used to reduce the computational burden of sequential and hybrid NMPC by storing and retrieving the DAE simulations.

Table 4.1: Comparison of Direct Methods for NMPC

	Single Shooting	Multiple Shooting	Collocation
Solution approach	sequential	hybrid	simultaneous
Use of DAE solvers	yes	yes	no
Size of NLP	small	intermediate	large
Initial guesses	initial states	all node values	all node values
Solves highly unstable systems	no	yes	yes
DAE model fulfilled in each iteration step	yes	partially	no

By explicitly transforming the DAE model, the integrations of the model are reduced. Because integrations of the model are the overwhelming majority of computational effort, a drastic reduction in calculation time can be achieved. In the sections 4.1.2 and 4.1.1 the sequential and hybrid NMPC approaches are summarized to reveal the application of ISAT.

4.1.1 Hybrid NMPC

Hybrid NMPC is a compromise between small NLP problem sizes of the sequential approach and the incorporation of state constraints in the simultaneous approach. Hybrid NMPC also permits the use of state of the art DAE solvers to control discretization error. The state estimation and regulator algorithms can be formulated as NLP problems.

4.1.1.1 Estimator

Dynamic data reconciliation (filtering of data, state estimation, etc.) is necessary for NMPC when modeled states are not directly measured or data uncertainty is high due to noise [47]. ISAT can be employed in a sequential direct multiple shooting approach to data reconciliation. The N-step finite-horizon problem formulation is given by the following.

$$\min_{x,\eta} \Phi(x,\eta,y) \overset{def}{=} \sum_{k=-N}^{-1} [C(x_k, y_k, \eta_k)] + C(x_0, y_0, \eta_0) \tag{4.1a}$$

subject to

$$y\ given,\ u\ given,\ x_{k+1} = F(x_k, u_k),\ y_k = G(x_k, u_k) \tag{4.1b}$$

where Φ is the cost function, x is the sequence of model states ($x = x_{-N}$,x_{-N+1}, $\dots$,x_0), η is the sequence of state constraint violations ($\eta = \eta_{-N}$,η_{-N+1}, $\dots$,η_0), y is the sequence of measurements ($y = y_{-N}$,y_{-N+1}, $\dots$,y_0), u is the sequence of inputs ($u = u_{-N}$,u_{-N+1}, $\dots$,u_{-1}), and C is the cost function at each sampling instant. This formulation is similar to the dynamic optimization problem but instead of finding optimal inputs, optimal states are found that agree with the measured data (y). The terminal constraint was added to allow a different cost function for the most current measurement. With model-plant mismatch, the most current measurement is likely to be the most reliable and should therefore receive a greater weighting in the optimization.

ISAT is used in the direct single and multiple shooting solution techniques to state estimation [17] by storing and retrieving integrations of the continuous dynamic nonlinear model. Data reconciliation must occur before dynamic optimization in order to provide an estimate of the current states (x_0). It is critical for the cycle time of a real-time controller to operate faster than the response time of the process it is controlling. Cycle time selection depends on many factors including plant/model mismatch, stability margins for large disturbances, constraint violation tolerance, and economic factors.

4.1.1.2 Regulator

The N-step finite-horizon NMPC regulator is given by the following NLP problem. The length of the horizon, N, may be different for the estimator and regulator.

$$\min_{x,u,\eta} \Phi(x, u, \eta) \tag{4.2a}$$

subject to

$$x_0 \, given, \; x_{k+1} = F(x_k, u_k), \; Du_k \leq d, \; Gx_k - \eta_k \leq g, \; \eta_k \geq 0 \tag{4.2b}$$

In this problem formulation, Φ (the cost function) is typically quadratic in x (states), u (inputs), and η (state constraint violations) and therefore strictly convex. The symbols x, u, η denote sequences of vectors with $x = (x_1,x_2, \ldots,x_N)$, $u = (u_0,u_1, \ldots,u_{N-1})$, and $\eta = (\eta_1,\eta_2, \ldots,eta_N)$. The source of nonlinearity comes from the model function $F(x_k, u_k)$ that is solved by integrating the DAE model. ISAT fits into the NMPC scheme by storing and retrieving integrations of the continuous dynamic model. Typically the model is expressed in the open equation continuous format $F(\dot{x}, x, u) = 0$. ISAT numerically transforms the nonlinear model into a discrete semi-explicit form $x_{k+1} = F(x_k, u_k)$. Details of this transformation are given in Chapter 2

4.1.2 Sequential NMPC

Sequential NMPC leads to the smallest NLP problems with the fewest number of optimization variables. In this approach, only the manipulated variables are optimized while the state variables are explicitly calculated at each iteration. In addition to size, sequential NMPC is the simplest to program and can be used with dense matrix solvers. In addition to hybrid NMPC, sequential NMPC allows the use of DAE solvers to solve the model on the sub-node level. DAE solvers avoid the discretization challenges of collocation approaches. The estimator and regulator NLP problem formulations are included in this section to show the applicability of ISAT in reducing the computational burden.

4.1.2.1 Estimator

The optimization variables for sequential data reconciliation are the states at N time steps back from the current time (x_{-N}). The states at x_{-N} are manipulated to minimize the objective function.

$$\min_{x_{-N}} \Phi(x, y) \overset{def}{=} \sum_{k=-N}^{-1} [C(x_k, y_k)] + C(x_0, y_0) \tag{4.3a}$$

subject to

$$y\ given,\ u\ given,\ x_{k+1} = F(x_k, u_k) \tag{4.3b}$$

where y and u are sequences of vectors given by $y = (y_{-N}, y_{-N+1}, \ldots, y_0)$ and $u = (u_{-N}, u_{-N+1}, \ldots, u_{-1})$. ISAT is an integral part of the sequential NMPC estimator by integrating the nonlinear continuous dynamic model, effectively transforming the model into a discrete form ($x_{k+1} = F(x_k, u_k)$).

4.1.2.2 Regulator

The regulator formulation minimizes an objective function by manipulating the decision variables (u_k for $k = 0, (N-1)$). Only manipulated variable constraints are permitted in the sequential regulator. State variable constraints can be posed as soft constraints in the objective function. In hybrid NMPC the state constraints are softened with the introduction of η, the state constraint violations. Without the softening of the state constraints infeasible solutions may arise. One advantage of softening the state constraints is that proritization of the constraints occurs automatically. This may be desireable to meet safety constraints while sacrificing less important objectives such as economic constraints.

$$\min_u \Phi(x, u, \eta) \tag{4.4a}$$

subject to

$$x_0\, given,\ x_{k+1} = F(x_k, u_k),\ Du_k \le d \tag{4.4b}$$

where x, u, and η represent the sequences of vectors (x_1,x_2, …,x_N), (u_0,u_1, …,u_{N-1}), and (η_1,η_2, …,η_N). Again, ISAT can be applied for computational reduction by storing and retrieving solutions to the initial value problems. Integrating the model for the sequential NMPC regulator requires about 99% of the computational effort for the example problem shown in Section 4.2. By reducing that computational burden of integrating the model, the total

NMPC cycle time is reduced by approximately 100 times. The computational reduction is very problem dependent, but typical performance on a handful of problems tested have been in the range of 20-500 times faster.

4.2 Example: Sequential NMPC

Combining model reduction and computational reduction through ISAT exploits the strengths of both methods. Generally, the model reduction step decreases the number of dynamic variables but does not have sufficient computational speed-up. Computational reduction is more effective with a low number of dynamic variables and can reduce the computational time significantly. Figure 4.1 provides an overview of the combined approach for the 32 state distillation column model in Hahn and Edgar [34]. A case study has been performed with the distillation column model comparing NMPC/LMPC

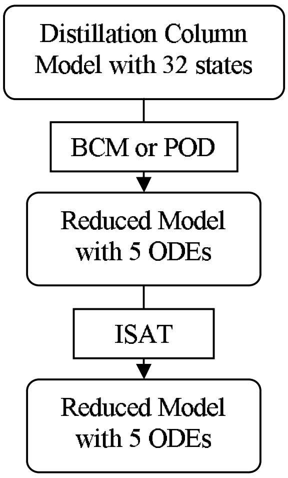

Figure 4.1: Model and computational reduction flowchart.

for the following models.

1. Nonlinear reduced model with 5 dynamic states and ISAT
2. Nonlinear reduced model with 5 dynamic states
3. Nonlinear model with 32 states
4. Linear model with 32 states

Models 1 and 2 are from full state model with 32 dynamic variables, reduced through truncation down to 5 dynamic variables. Using the sequential approach to MPC, the distillation column models are integrated multiple times in order to find optimal control moves that minimize a quadratic objective cost function.

Certain operational, safety, or economic constraints must be considered when developing control solutions to real systems. These limitations can be implemented as either hard or soft constraints in the MPC framework. Soft constraints are costs added to the objective function. In the author's opinion, soft constraints are the more intuitive method because the solver can choose to violate a constraint if the economic performance of the entire plant will be improved. In addition, the relative importance of each soft constraint is automatically considered. Hard constraints may be more desirable for some situations such as when safety is a concern. In this sequential NMPC approach, hard constraints can be implemented on the manipulated variables.

The nominal operating point for the reflux ratio is 3. Soft constraints limit the operating region to between 2 and 4. The reflux ratio (manipulated variable) is adjusted every 5 minutes. The control horizon is 10 minutes (2 time steps) and the prediction horizon is 15 minutes (3 time steps). Typical industrial MPC control horizons are generally longer to approximate the infinite horizon solution. The coarse discretization and short time horizon in this example are sub-optimal, but still show an instructive comparison of the reduced models in an MPC application. Longer time horizons should actually improve ISAT's performance due to more model integrations with each optimization. Because ISAT stores and retrieves model integrations, an increase in integrations will train the database faster. The main ISAT tuning parameter, ϵ_{tol}, is set to 10^{-3} for good accuracy. Figure 4.2 shows the closed loop responses. During the first control move, all MPC results are at the reflux ratio lower bound of 2. Figure 4.3 shows the speed-up factor (compared to 32 state NMPC) for the 5 optimization steps of Figure 4.2. The cpu times shown on the graph are from computations on a 2 GHz Celeron® processor. The results from this simulation use a previously trained ISAT database with 169 records. Without a previously trained database ISAT averages 30 times faster over the first 5 optimizations and adds 13 records to the new database. This case study shows that ISAT can exhibit significant computational reduction while preserving the accuracy of the nonlinear model.

Although applied with a model reduced through balanced covariance matrices, ISAT for NMPC can be used with any model reduction technique

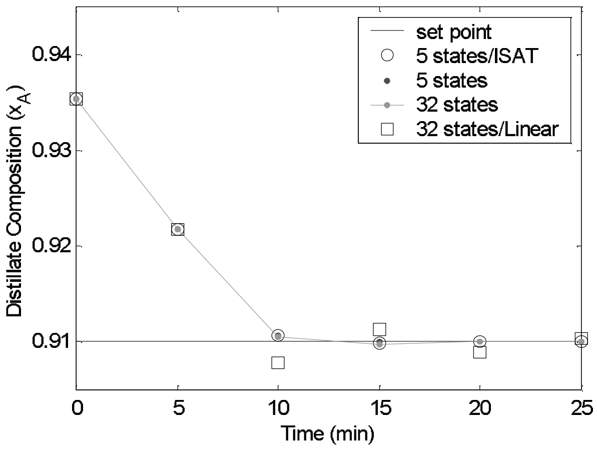

Figure 4.2: Closed loop response comparison for nonlinear MPC with ISAT with 5 states, nonlinear MPC with 5 states, nonlinear MPC with 32 states, and linear MPC.

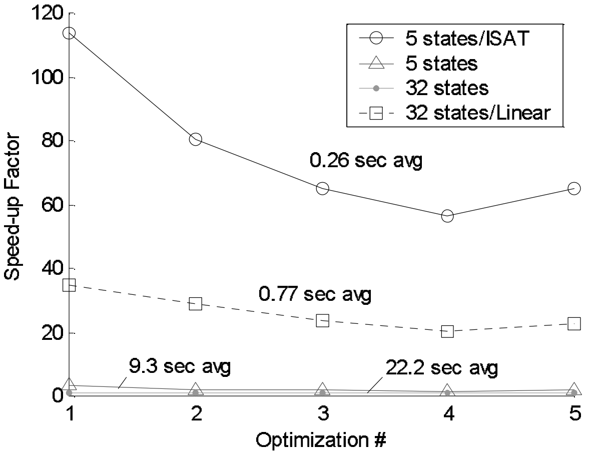

Figure 4.3: Speed-up factor for each of the optimizations shown in Figure 4.2. The number above each curve indicates the average optimization cpu time on a 2 GHz processor.

that reduces the number of dynamic degrees of freedom. In the case where the model already has a low number of variables, ISAT can be applied directly without a model reduction step.

4.3 Example: ISAT vs. Neural Networks in Control Calculations

As mentioned previously, a neural net is a type of storage and retrieval method. Hence it is instructive to compare ISAT and a neural net in a control application. The example model is a dual CSTR model (see Figure 11) with one manipulated variable (heat addition to the first tank), six states, and one controlled variable (temperature of the second reactor). The model was used by Hahn and Edgar [34] as a benchmark model for nonlinear model reduction (see Figure 4.4). The data were gathered from ISAT training. For the sake of comparison, the neural net used the same 1609 ISAT records for training. The neural net was constructed with MATLAB's neural net toolbox as one

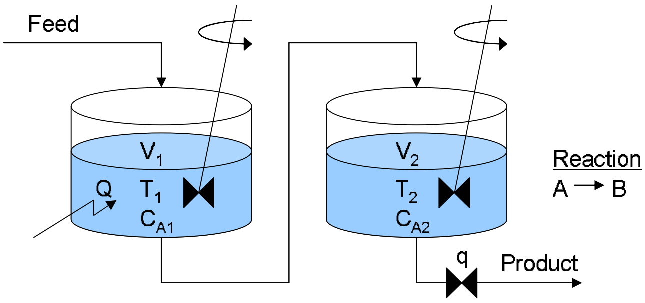

Figure 4.4: Diagram of two CSTRs in series with a first order reaction. The manipulated variable is the heating rate to the first CSTR.

nonlinear hidden layer and a linear output layer (see Figure 4.5). Before the

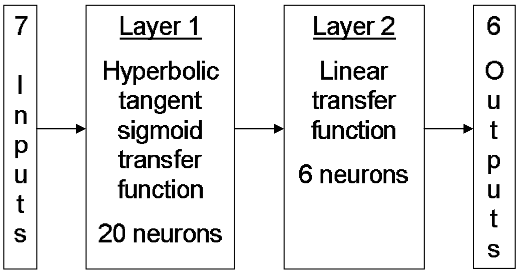

Figure 4.5: Neural net with one hidden layer and one output layer. The hidden layer is a hyperbolic tangent function and the output layer is a linear function. This neural net relates 7 inputs to 6 outputs.

training, the data were appropriately scaled for efficient implementation in the neural net. Figure 4.6 shows a large open loop step test, one that is outside those found in the training data. In this step test, the cooling is increased to the point that the irreversible reaction is extinguished and a large temperature step results. Up to about 5 minutes of simulated time, the neural net and ISAT perform similarly. To this point both accessed data that were within the training domain. Beyond 5 minutes ISAT is superior in agreement with the non-reduced model due to a built in error checking strategy. Before 5 minutes, the ISAT method performs mostly retrievals. Once ISAT detects large errors from retrievals, it starts adding records to the database. When the temperature reaches steady state, the ISAT algorithm performs retrievals again.

ISAT and the neural net were compared in a closed loop simulation with a small set point change inside the training domain (see Figure 4.7). All

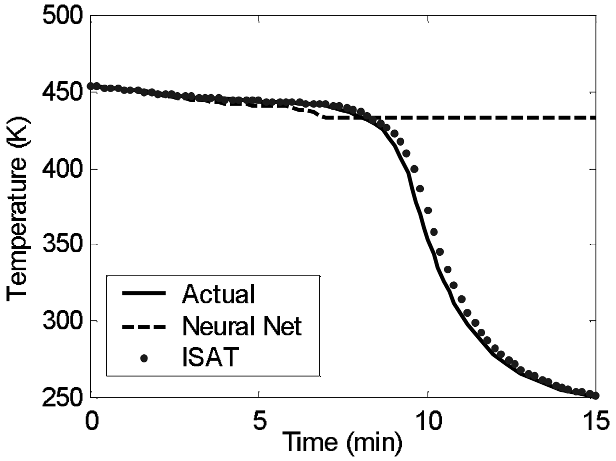

Figure 4.6: Open loop step test for the dual CSTR model. The error control of ISAT adds records to the database when extrapolating outside of the training domain.

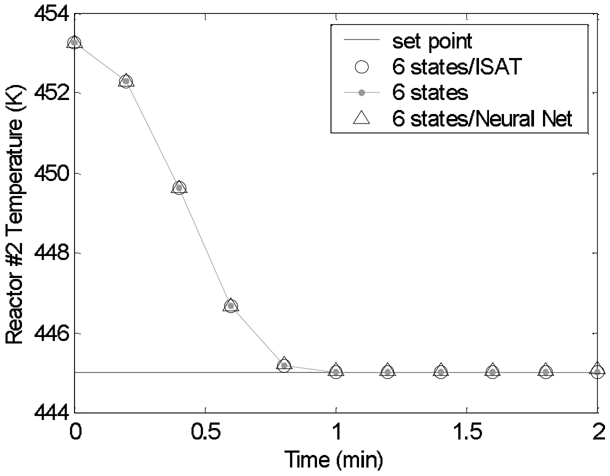

Figure 4.7: Small closed loop set point change within the training domain.

three show excellent agreement as they reach the new set point along the same trajectory. Next, a large set point change was performed to access a region of state space outside of the training domain (see Figure 4.8). For this step change, the neural net controller eventually becomes unstable. This is because the neural net does not have the capability to extrapolate outside of the data that was used to train it. In this respect, the ISAT method is superior because it detects when it has gone outside of the training domain and integrates the model to generate and add new data to the training set.

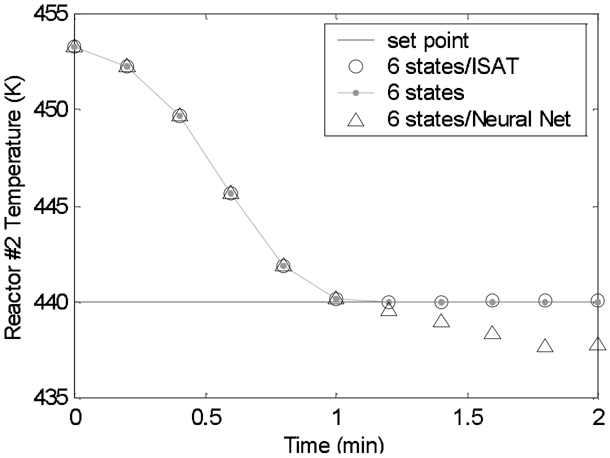

Figure 4.8: Large closed loop set point change outside of the training domain.

ISAT outperforms neural nets because of the internal error control that manages the amount of error. The only tuning parameter for ISAT is the amount of permissible error, ϵ_{tol}. On the other hand, neural nets have multiple tuning parameters such as number and type of layers, number of nodes in each layer, and a training optimization tolerance. ISAT requires no optimization step and can begin working *in situ* with no prior training set.

4.4 Parameter Sensitivity Studies

The dynamics of a controlled process can change due to fouling, disturbances, unusual operating states, ambient variations, and changes in product specifications [76]. When the process dynamics change significantly, an adaptive model is automatically tuned to provide satisfactory closed loop performance. Adaptive control can be achieved in the NMPC framework with the addition of adjustable parameters (θ) in the nonlinear model.

$$\frac{dx}{dt} = f(x(t), u(t), \theta) \tag{4.5}$$

$$0 = g(x(t), u(t), \theta) \tag{4.6}$$

The adjustable parameters can be obtained from a first principles model of the disturbance or a least squares optimization of the model using plant historical data. The addition of adjustable parameters poses an interesting challenge for ISAT's error control strategy. Because ISAT is a storage and retrieval method, drastically changing the parameters can invalidate the stored data. Therefore, a strategy will be devised to gradually change the parameters and simultaneously filter out the unaccessed data. By controlling the parameter transition, ISAT will still show significant computational reduction over the original NMPC. Another advantage of gradual parameter transition is that it avoids possible instabilities that can occur by switching controllers on-line.

Gradual parameter transition is a natural expression of the process dynamics when the system dynamics change slowly, such as for fouling or catalyst deactivation.

There may be some situations where the parameter transition should occur quickly (i.e. grade changes, large disturbances). In these situations, a gradual parameter transition is not appropriate. For a large change in the parameters, the ISAT database provides a first order approximation to the nonlinear model integration while new sensitivities are computed. In this way the real-time controller requirements are automatically met with a simplified model. Once the ISAT database is rebuilt, the controller will improve according to the predictive capabilities of the full nonlinear model.

4.5 Summary and Conclusions

This chapter outlines a new technique for computational reduction for NMPC. In this approach, model reduction through balanced covariance matrices is followed by computational reduction through ISAT. Although previously developed for turbulent flame simulations, ISAT can be directly applied because many open loop simulations are performed to find optimal inputs to the control problem. A case study with a binary distillation column model showed a speedup of 85 over the original NMPC. Like neural nets, ISAT reduces the computational cost through storage and retrieval. Another case study with a dual CSTR showed the advantage of using ISAT over neural nets when the simulation accessed data outside the training domain.

Chapter 5

Nonlinear Model Predictive Control - The Explicit Solution

Model predictive control (MPC) has traditionally been an expensive technology, confined to applications that justify substantial modeling effort, implementation costs, and computational resources (fast computers). The application of MPC has also been limited to processes with slow cycle times (slow processes) because it requires the solution to a constrained finite-horizon linear programming (LP) or quadratic programming (QP) problem at each sampling instant. The success (or failure) of MPC is due to the accuracy of the underlying model. This model is used to predict unmeasured or noisy states, coordinate multiple decision variables for optimal control, and meet safety and operational constraints. MPC is a type of optimal control because the optimization minimizes a cost function subject to constraints. At the solution of the minimized objective function the only way to get better performance is to increase the accuracy of the model, relax contraints, or modify the cost function to reflect more realistic price structures.

For some applications linear models are not sufficiently accurate. When models or constraints are nonlinear, a nonlinear programming (NLP) optimization must be solved at each cycle. Nonlinear MPC (NMPC) problems are often significantly more difficult to solve than QP problems solved in MPC. One difficulty that non-convex problems can lead to multiple local minima. Global minimum solvers are still the subject of active research. The increase in problem difficulty further restricts NMPC applications to those with slower processes and faster computers.

PID control, compared to MPC, is a relatively cheap technology that can be applied with slow computers to fast processes. However, PID control is not formulated to provide optimal model-based control, effective handling of constraints, or coordination of multiple decision variables.

5.1 Explicit MPC (Linear Models)

MPC is now suggested as a candidate to replace PID control thanks to recent developments in computational reduction of the MPC algorithm [56] [61] [65]. By computing all possible LP solutions off-line, the on-line portion is reduced to some conditional checking and simple matrix multiplications. This modification extends the potential of MPC to fast processes and simple computers (e.g. integrated circuit chips).

The linear quadratic regulator (LQR) with a linear model and quadratic objective function is a special case of MPC without constraints. Without constraints the linear solution of the Riccati equation is optimal for all possible initial states. An on-line implementation of LQR would consist of simply multiplying the state vector by the gain matrix to obtain the optimal control vector. With constraints, the optimal solution is a piecewise affine (PWA) linear function of the initial states. The linear regions are often refered to as characteristic or critical regions (CRs). Each region is bounded by a set of constraints. When the constraint boundary is crossed, the linear solution may no longer be exact. On-line retrieval of explicit MPC with constraints includes one extra step: location of the region with the correct active set of constraints. Once this region is located (via the checking of several conditions), the rest of the computation is identical to the LQR implementation.

5.1.1 Parameterization of Initial States

The development of multi-parametric linear programs (mp-LPs) started with the formulation of Gal and Nedoma [29] [28]. Acevedo and Pistikopoulos extended sensitivity analysis to mixed-integer linear programming (MILP) by solving mp-LP problems [2] [1]. Dua and Pistikopoulos generalized the model form by developing multiparametric analysis of mixed-integer nonlinear programming (MINLP) [24]. Bemporad et al. applied the mp-LP work to MPC applications with linear objective functions [10] [15] and mixed-integer models [11]. Pistikopoulos et al. extended the theory of mp-LPs to include multi-parametric quadratic programs (mp-QPs) [64]. This extension made possible the explicit LQR solution subject to constraints or in other words, explicit MPC [14] [12].

Even though an exact explicit solution is possible in theory for convex problems, there were some serious implementational issues that limited applications of explicit MPC to small systems, few constraints, and short control horizons [65]. A significant effort has been exerted to reduce these limitations. Bemporad and Filippi introduced suboptimal explicit MPC [13]. Adjacent critical regions are merged when an error tolerance can be met. Rossiter and Grieder used an interpolation scheme to reduce the storage requirements by 2-3 times and reduce the on-line computational costs by 10 times [72]. Johansen and Grancharova proposed a technique to logrithmically limit the on-line search times with a structured binary tree [31] [44]. Off-line, the regions are divided into successively smaller hypercubes until the error tolerances are met at each of the vertices. Grieder and Morari performed a complexity analysis of the on-line implementation to reduce the controller complexity by orders of magnitude at a performance cost of <%1 [32]. Tondel et al. increased the efficiency of the off-line calculation by deriving a new exploration strategy for sub-dividing the parameter space [86] [87]. Even with all of these improvements, the largest MPC problem reported in the literature is control of a laboratory model helicopter. The problem has 6 states, 2 manipulated variables, 8 constraints, is discretized in 0.01 second segments, has a control horizon of 0.5 seconds, and 4 input parameters [86].

5.1.2 Parameterization of Active Sets

In deriving state parameterized explicit MPC, the problem is transformed into a quadratic program form. In this form, Seron et al. suggested that the optimal control can be parameterized by the active set instead of current states [77]. While Seron, et al. proposed an analytical solution, Pannocchia et al. opened the approach to non-trivial problems by creating a numerical algorithm to solve the active set parameterized problem [60]. Each of the constraints can either be inactive, at the lower bound, or at the upper bound. Off-line a table of all possible solutions is generated. The on-line portion consists of finding the table value that predicts non-negative lagrange multipliers and manipulated variables (MVs) inside the constraint bounds. Storage and retrieval of a constrained linear quadratic controller solution for SISO systems has been proposed to replace PID control [61]. Two limitations of this algorithm are (1) constraints are restricted to lower and upper bounds on the MVs and (2) problem scaling is 3^N, where N is the horizon length. The theory for MIMO systems follows by simple extension, but full enumeration of all active sets is prohibitive due to 3^{mN} scaling, where m is the number of constrained inputs.

Muske and Badgwell developed offset free control in MPC by creating input or output integrating disturbances [57]. Pannocchia and Rawlings showed that an integrating disturbance must exist for every measurement to guarantee offset free control [59] [62]. The offset free control is included in the constrained LQ control of Pannocchia et al. [61]. Sakizlis et al. followed by incorporating offset free control into state parameterized explicit MPC [73].

5.2 Explicit NMPC (Nonlinear Models)

Fiacco developed the foundation for explicit NMPC with a sensitivity analysis of nonlinear systems [27]. Because there is rarely an exact explicit solution to NMPC, all computational reduction techniques for NMPC are approximate. The effectiveness of a particular technique depends on the control of the approximation error, storage requirements, speed of the off-line algorithm, speed of the on-line algorithm, and guarantees of stability. An explicit solution of NMPC in this section refers to an explicit numerical solution through storage and retrieval of previous computed solutions. An analytic explicit solution is not attempted.

5.2.1 Dynamic Programming

Dynamic programming was originally proposed by Bellman to solve optimal control problems [8]. The goal of dynamic programming is to find an optimal cost-to-go function, which can be used to solve for an optimal trajectory of inputs as a function of initial states. Recent approaches such as sequential reinforcement learning avoid dynamic programming dimensionality problems by operating on states as they occur sequentially [6]. Also, neuro-dynamic programming [45] [16] overcomes the curse of dimensionality by approximating the cost-to-go function with a neural net. Yet another technique that balances accuracy with computational speed is suboptimal dynamic programming with error bounds [48]. In summary, dynamic programming's curse of dimensionality has been partially remedied by algorithms that seek to reduce the storage and search times. However, applications to large scale problems are still infeasible.

5.2.2 Artificial Neural Networks

Neural nets are an effective tool to represent nonlinear models. Neural nets are networks of adaptable nodes which, through a process of learning from task examples, store experimental knowledge and make it available for later use [5]. The flexibility and general applicability of neural nets have been demonstrated by diverse applications across many fields of study. Kohonen nets are used in classification and fault detection, n-tuple nets in image processing and vision, and both multi-layer perceptrons and radial basis functions are used in signal processing and control [90]. Neural nets are an effective tool to incorporate historical data for use in state estimation and control, although filtering and preconditioning the plant data are often time-consuming tasks [67]. Parisini and Zoppoli suggested that multilayered feedforward neural networks could store optimal control (outputs) as a function of the current states (inputs) [63]. One widely known limitation of neural nets is the inability to extrapolate outside the training domain. This is due to a lack of explicit error control within the algorithm.

5.2.3 Multiparametric NMPC

For multiparametric analysis, suboptimal explicit MPC techniques have been developed to allow nonlinear models, nonlinear constraints, and nonquadratic objective functions. Bemporad et al. introduced multiparametric approximation of MINLP problems [9]. Johansen formerly utilized an mp-QP approximation to solve the mp-NLP sub-problem[42] but later decided to use the increased accuracy and computational expense of NLP sub-problems [43]. Hale and Qin [36] take a similar approach as Johansen but use simplices instead of hypercubes to map the nonlinear surface. A predictor-corrector method is used to obtain new points. The predictor is a linear extrapolation from an existing point to a new point of interest. If the active set of constraints changes, a condition is applied to find the active set boundary [35]. The corrector uses a Newton's method type algorithm to solve the NLP that converges rapidly because of the linear predictor initialization. One drawback is poor computational scaling with increasing number of parameters (in this case, number of states), but polynomial scaling in other dimensions.

5.3 Approximate Nonlinear MPC

Consider the continuous-time nonlinear differential algebraic equation (DAE) system

$$0 = f(\dot{x}(t), x(t), u(t), \theta) \tag{5.1}$$

where $\dot{x}(t) \in \mathbb{R}^p$ is the state derivative, $x(t) \in \mathbb{R}^n$ is the state, $u(t) \in \mathbb{R}^m$ is the input, and $\theta \in \mathbb{R}^q$ is a set of parameters. The dimension of $\dot{x}$ is equal to that of x for ODE models. The discrete-time nonlinear DAE system can be obtained by numerically integrating Equation 5.1 as an initial value problem (IVP), resulting in the explicit form that is solved sequentially on a sub-node level in optimal control problems

$$x_{k+1} = f(x_k, u_k, \theta) \tag{5.2}$$

or by orthogonal collocation, creating an implicit form that is solved on a sub-node level simultaneously in optimal control

$$0 = f(x_{k+1}, x_k, u_k, \theta) \tag{5.3}$$

where $x_k \in \mathbb{R}^n$ and $u(t) \in \mathbb{R}^m$. The indices (k) refer to the discretized step with the current time being 0. In optimal control, when a sampling instant occurs the current time is shifted to zero.

5.3.1 NMPC Formulation

For the current state (x_0) and parameters (θ), a typical NMPC algorithm solves the optimization problem

$$\Phi^*(x_0, u_{-1}, \theta) = \min_{x,u} \left(\sum_{i=1}^{N} f_i(x_i, u_{i-1}, \theta) \right) \tag{5.4a}$$

subject to

$$x_0 \, given \tag{5.4b}$$

$$u_{-1} \, given \tag{5.4c}$$

$$0 = f(x_{k+1}, x_k, u_k, \theta) \quad k = 1, \ldots, N-1 \tag{5.4d}$$

$$Dx_k \leq d \quad k = 1, \ldots, N \tag{5.4e}$$

$$Eu_k \leq e \quad k = 0, \ldots, N-1 \tag{5.4f}$$

$$G\left(u_k - u_{k-1}\right) \leq g \quad k = 0, \ldots, N-1 \tag{5.4g}$$

where D, E, and G are matrices and d, e, and g are vectors of appropriate dimension. The quantities x and u refer to the sequence of vectors $(x_1,x_2,\ldots,x_N)$ and $(u_0,u_1,\ldots,u_{N-1})$, respectively. The optimal solution to the NMPC problem is a unique function of the current states x_0, previous input u_{-1}, and the adjustable parameters, θ. The adjustable parameters can be feedforward or feedback model variables. An example of feedback variables are input or output integrating disturbances for offset free control [57] [62]. Feedforward parameters accommodate anticipated shifts in process dynamics or multiple model switching.

After the optimal control problem is solved the first input (u_0^*) is injected into the process. At the next sampling instant, a new estimate of the current states and parameters is obtained. NMPC is often referred to as receding horizon control (RHC) because the horizon of the optimal control problem shifts as time advances. The same optimal control problem is solved at every sampling instant, deterministically dependent on the updated variables assembled in ϕ.

$$\phi = \begin{bmatrix} x_0 \\ u_{-1} \\ \theta \end{bmatrix} \tag{5.5}$$

Even though the entire trajectory of optimal inputs are solved $(u^* = \{u_0^*, u_1^*, \ldots, u_{N-1}^*\})$, the only one required for optimal control is the first input, u_0^*. The storage and retrieval of optimal control can therefore be simplified to u_0^* as a unique function of ϕ.

5.3.2 Sensitivity of Optimal Control to Parameters

Ganesh and Biegler developed a reduced hessian strategy for sensitivity analysis of optimal flowsheets [30]. A part of their sensitivity derivation is given here. Sensitivities locally approximate the optimal solution with a 1st order solution. NMPC can be expressed more compactly with adjustable parameters ϕ, inequality constraints g, and equality constraints h.

$$\Phi^*(\phi) = \min_{x,u} \left(\Phi(x, u)\right) \tag{5.6a}$$

subject to

$$\phi \; given \tag{5.6b}$$

$$g(x, u, \phi) \leq 0 \tag{5.6c}$$

$$h(x, u, \phi) = 0 \tag{5.6d}$$

where x and u refer to the sequence of vectors $(x_1,x_2,\ldots,x_N)$ and $(u_0,u_1,\ldots,u_{N-1})$, respectively. The NLP is solved by minimizing the Lagrangian L

$$L(x, u, \phi) = \Phi(x, u, \phi) + \lambda g(x, u, \phi) + \nu h(x, u, \phi) \tag{5.7}$$

where Φ is the objective function, λ is the inequality constraint multiplier, and ν is the equality constraint multiplier. The Karush-Kuhn-Tucker (KKT) conditions are satisfied at the optimal solution.

$$\nabla\Phi(x, u, \phi) + \nabla g(x, u, \phi)\lambda + \nabla h(x, u, \phi)\nu = 0 \tag{5.8a}$$

$$\lambda g(x, u, \phi) = 0 \tag{5.8b}$$

$$\lambda \geq 0 \tag{5.8c}$$

$$g(x,u,\phi) \leq 0 \tag{5.8d}$$

$$h(x,u,\phi) = 0 \tag{5.8e}$$

The solution sensitivity reveals how the optimal solution changes with deviations in the the parameters ϕ. In order for a local sensitivity to exist, a few conditions must be met. First, the Lagrangian must be twice continuously differential in x and u and once in ϕ. Second, the constraint gradients must be linearly independent at the optimal solution. Finally, the second-order sufficiency conditions must be met. In generating the local sensitivities it is assumed that the active set does not change. The active constraints are

$$\nabla_x L(x,u,\phi) = 0 \tag{5.9a}$$

$$\nabla_u L(x,u,\phi) = 0 \tag{5.9b}$$

$$g_A(x,u,\phi) = 0 \tag{5.9c}$$

$$h(x,u,\phi) = 0 \tag{5.9d}$$

where $g_A$is the subset of g that are at the equality bound. The sensitivities are derived by taking the total derivative of the active constraints listed in Equation 5.9.

$$\mathrm{d}[\nabla_x L(x,u,\phi)] = \nabla_{xx} L \mathrm{d}x + \nabla_{ux} L \mathrm{d}u + \nabla_x g_A \mathrm{d}\lambda + \nabla_x h \mathrm{d}\nu + \nabla_{\phi x} L^T \mathrm{d}\phi = 0 \tag{5.10a}$$

$$\mathrm{d}[\nabla_u L(x,u,\phi)] = \nabla_{xu} L \mathrm{d}x + \nabla_{uu} L \mathrm{d}u + \nabla_u g_A \mathrm{d}\lambda + \nabla_u h \mathrm{d}\nu + \nabla_{\phi u} L^T \mathrm{d}\phi = 0 \tag{5.10b}$$

$$\mathrm{d}g_A(x,u,\phi) = \nabla_x g_A^T \mathrm{d}x + \nabla_u g_A^T \mathrm{d}u + \nabla_\phi g_A^T \mathrm{d}\phi = 0 \tag{5.10c}$$

$$\mathrm{d}h(x,u,\phi) = \nabla_x h^T \mathrm{d}x + \nabla_u h^T \mathrm{d}u + \nabla_\phi h^T \mathrm{d}\phi = 0 \tag{5.10d}$$

Each of the equations in 5.10 is divided by $\mathrm{d}\phi$. In the limit as $\mathrm{d}\phi$ shrinks to zero the local sensitivities become (with some rearrangement)

$$\begin{bmatrix} \nabla_\phi x^T \\ \nabla_\phi u^T \\ \nabla_\phi \lambda^T \\ \nabla_\phi \nu^T \end{bmatrix} = - \begin{bmatrix} \nabla_{xx} L & \nabla_{xu} L & \nabla_x g_A & \nabla_x h \\ \nabla_{ux} L & \nabla_{uu} L & \nabla_u g_A & \nabla_u h \\ \nabla_x g_A^T & \nabla_u g_A^T & 0 & 0 \\ \nabla_x h^T & \nabla_u h^T & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \nabla_{\phi x} L^T \\ \nabla_{\phi u} L^T \\ \nabla_\phi g_A^T \\ \nabla_\phi h^T \end{bmatrix} \tag{5.11}$$

where $\nabla_\phi x$ is the state sensitivity, $\nabla_\phi u$ is the input sensitivity, $\nabla_\phi \lambda$ is the active inequality constraint multiplier sensitivty, and $\nabla_\phi \nu$ is the equality constraint multiplier sensitivity. Equation 5.11 shows that the only elements required for a sensitivity calculation are the exact hessian and Lagrangian second partials with respect to the parameters. With analytical derivatives through automatic differentiation the sensitivity calculation speed can be greatly improved [91].

5.3.3 Defining the Critical Region

For unconstrained LQ problems the local sensitivity gives an exact optimal solution over all state space. In this case, the sensitivity is equivalent to the unconstrained LQR gain matrix. For constrained LQ problems the optimal solution is linearly dependent on the adjustable parameters ϕ within the same active constraint region. An individual query point ϕ_q can be tested to determine if it lies within this critical region (CR). A 1st order approximation of optimal variables at ϕ_q is determined.

$$x_q = x + \nabla_\phi x \left(\phi_q - \phi\right) \tag{5.12a}$$

$$u_q = u + \nabla_\phi u \left(\phi_q - \phi\right) \tag{5.12b}$$

$$\lambda_q = \lambda + \nabla_\phi \lambda \left(\phi_q - \phi\right) \tag{5.12c}$$

Equation 5.13 gives the qualifications for a point within the CR.

$$g_I(x_q, u_q, \phi_q) \leq 0 \tag{5.13a}$$

$$\lambda_q \geq 0 \tag{5.13b}$$

where g_I is the set of inactive inequality constraints. If any of these qualifications are not met it indicates that the active set changed and the point lies outside the CR.

5.3.4 ISAT Approximate Control

In situ adaptive tabulation (ISAT) dynamically stores and retrieves nonlinear functions with piecewise linear approximations. The error control strategy proposed in Pope [66] and with further details given by Hedengren and Edgar [40] may be ineffective for problems with constraints. The constraints can form a non-continuously differentiable or non-continuous function. This leads to regions of accuracy (ROA) that may not be ellipsoidal in the limit as the error tolerance approaches zero. Modifications to the ISAT algorithm are made to maintain error control for optimal control storage and retrieval. Specifically, the initial estimate of the ROA is eliminated for nonlinear problems and restricted to the active set constraint region of constrained LQ problems. Additionally, ellipsoid of accuracy (EOA) expansions are made only after an expanded validity check is performed. This section is a tailored version of ISAT for receeding horizon control. A general exposition on ISAT is found in Chapter 2, but in a formulation for general nonlinear function approximation. The notation is adapted here for the control problem and new ISAT features are introduced to exploit the unique properties of cLQ solutions. The basic unit of the ISAT database is the record. An ISAT record consists of the independent variables (ϕ), the dependent variables (u_0^*), a sensitivity matrix ($\frac{\partial u_0^*}{\partial \phi}$), an ellipsoid of accuracy (EOA), and a critical region (CR) (see Table 5.1). The memory required to store an individual ISAT record scales with $O((n+m+q)^2)$.

Table 5.1: Elements of the ISAT record for NMPC storage and retrieval

ISAT Record Element	Symbol and Dimension
Independent variables	$\phi \in \mathbb{R}^{n+m+q}$
Dependent variables	$u_0^* \in \mathbb{R}^m$
Sensitivity	$A \in \mathbb{R}^{m \times (n+m+q)}$
Ellipsoid of accuracy	$M \in \mathbb{R}^{(n+m+q) \times (n+m+q)}$
Critical region (cLQ only)	$CR \in \mathbb{R}^{(n+q) \times (n+m+q)}$

5.3.4.1 First Scenario: Retrieval

When ISAT receives a database request, it performs one of three scenarios. In the first scenario, the query (ϕ_{q1}) is inside a region of accuracy termed the ellipsoid of accuracy (EOA), centered about a close stored record, ϕ_s (see Figure 5.1). Retrievals are extremely fast because computations are limited to

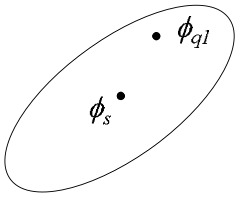

Figure 5.1: A retrieval occurs when the query point (ϕ_{q1}) is within the ellipsoid of accuracy (EOA)

a binary tree search, conditional checking, and matrix-vector multiplications. When the ISAT database is mature most of the operations are retrievals.

5.3.4.2 Second Scenario: EOA Growth

In the second scenario, the query is outside the EOA but inside the error tolerance for u_q^* and u_{s-q}^*. In this case, the EOA is expanded to include the tested query point (see Figure 5.2). For the second and third scenarios, ISAT has no computational advantage over directly solving the original NLP problem on-line. If real-time requirements prohibit an on-line NLP solution, an approximation to the optimal control can be obtained by using $\hat{u}_q^*$ anyway, but no guarantees of accuracy or stability are provided.

5.3.4.3 Third Scenario: Addition

In the final scenario, the query is outside the EOA and outside the error tolerance for u_q^* or u_{s-q}^*. A new ISAT record is added with an initial estimate of

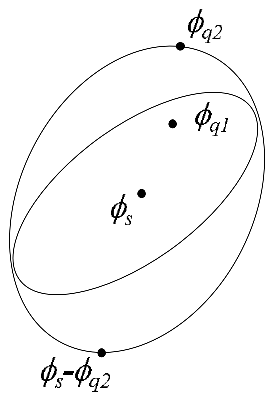

Figure 5.2: The EOA is grown when the query point is outside the EOA but within the error tolerance for u_0^*

the ROA (see Figure 5.3). For constrained LQ problems, the optimal control solution is linear with respect to ϕ. Therefore, an initial estimate of the ROA is the active set state space. For nonlinear problems with constraints, there is no accuracy guarantee. In this case, the initial estimate of the ROA is a zero-volume ellipsoid centered at u_q^*.

5.3.5 Summary of the ISAT Algorithm

ISAT can be summarized in 13 steps. Steps 1-5 are the retrieval steps, 6-11 attempt growth of the EOA, and 12 is a database addition. The last step is to inject either u_0^* or $\hat{u}_0^*$ into the process. Retrievals produce approximate optimal control within the desired error tolerance ϵ_{tol} whereas growths and additions produce exact answers.

1. locate nearby records with multiple binary tree searches

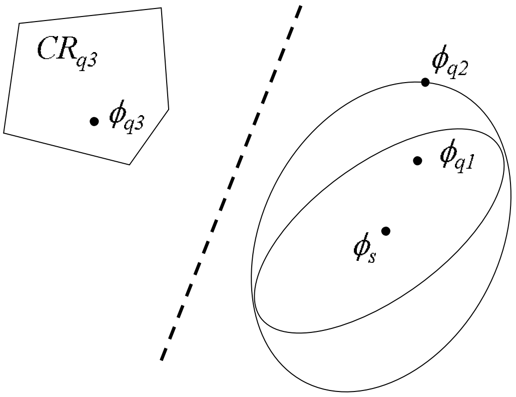

Figure 5.3: A record is added to the ISAT database when query point is outside the EOA and error tolerance for u_0^*

2. compute $\hat{u}_q^* = u_s + A\left(\phi_q - \phi_s\right)$
3. (for QP problems) if $\lambda_q \leq 0$ and $g_I(x_q, u_q, \phi) \leq 0$ go to 5
4. if $\phi_q^T M \phi_q \leq \epsilon_{tol}$ go to 5
5. set $\hat{u}_0^* = \hat{u}_q^*$, go to 13
6. solve the NLP (or QP) for ϕ_q
7. if $\left|\hat{u}_q^* - u_q^*\right| > \epsilon_{tol}$, go to 12
8. solve the NLP (or QP) for $(2\phi_s - \phi_q)$ to get u_{2s-q}^*
9. compute $\hat{u}_{2s-q}^* = u_s + A\left(\phi_s - \phi_q\right)$
10. if $\left|u_{2s-q}^* - \hat{u}_{2s-q}^*\right| > \epsilon_{tol}$, go to 12
11. grow EOA, set $u_0^* = u_q^*$, go to 13

12. add a new record to the database with a zero volume EOA (if QP, initial ROA is given by $\lambda_q \geq 0$) and $g_I(x_q, u_q, \phi) \leq 0$

13. inject u_0^* for optimal control or $\hat{u}_0^*$ for approximate optimal control

5.4 Temperature Control of an Exothermic CSTR

A simple academic problem is considered to show the applicability of ISAT to storage and retrieval of optimal control. A perfectly mixed, adiabatic CSTR has an exothermic reaction of compound A transformed into compound B. Temperature control of the reactor is a challenge due to the highly exothermic reaction ($\Delta H_{rxn} = 50{,}000 \ \frac{J}{mol}$). The temperature of the fluid in the jacket surrounding the CSTR is manipulated to control the temperature of the reactor fluid. The dynamics of the reactor are described by a set of ODEs generated from a mole balance on A and an energy balance on the reactor.

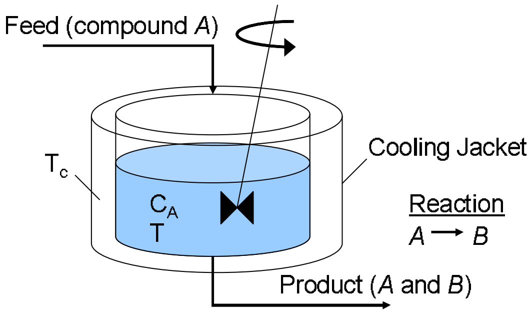

Figure 5.4: Diagram of the exothermic CSTR. The two state variables reactor concentration c_A and temperature T are controlled by the jacket cooling temperature T_c

At a constant cooling temperature of 305 K, the reactor temperature spikes continuously as the reactor goes through cycles of concentration buildup followed by moments of intense reaction (see Figure 5.5). The unsteady response of the reactor with a constant cooling jacket temperature suggests that unsteady control may be necessary when pushing the reactor to the stability limit. A sequential direct single shooting approach to dynamic optimization is used as the control algorithm. The N-step finite horizon NMPC is given by the following NLP problem.

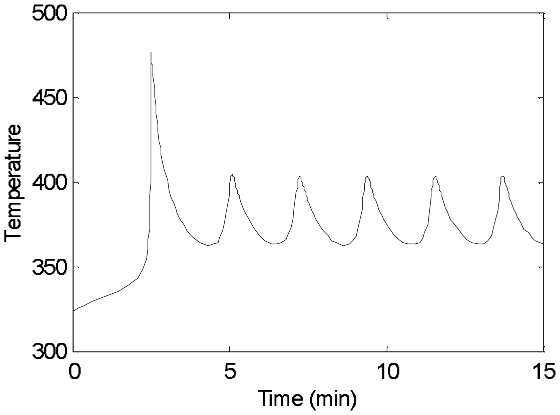

Figure 5.5: Unsteady response of the reactor temperature due to moments of intense reaction followed by periods of gradual cooling.

$$\Phi^*(x_0, T_{sp}) = \min_{x,u} \left(\sum_{i=1}^{N} (x_i - \theta)^T Q (x_i - \theta) \right) \tag{5.14a}$$

subject to

$$x_0 \, given \tag{5.14b}$$

$$x_{k+1} = f(x_k, u_k) \quad k = 1 \dots N \tag{5.14c}$$

$$Eu_k \leq e \quad k = 0 \ldots N-1 \tag{5.14d}$$

where

$$N = 40 \quad Q = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \quad E = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad e = \begin{bmatrix} 320 \\ -280 \end{bmatrix} \quad \theta = \begin{bmatrix} 0 \\ T_{sp} \end{bmatrix} \tag{5.14e}$$

Here x and u refer to the sequence of vectors $(x_1,x_2,\ldots,x_N)$ and $(u_0,u_1,\ldots,u_{N-1})$, respectively. In this problem formulation, Φ (the cost function) is quadratic in x (states) and therefore strictly convex. The source of nonlinearity comes from the model function $f(x_k, u_k)$ that is solved by integrating the ODE model. With a constant reactor temperature set point, the first optimal control step u_0^* is a unique function of the current concentration and temperature of the reactor. The optimal cooling jacket temperature ($u_0^* = T_c^*$) to drive the reactor temperature to 320 K was calculated for reactor concentrations between 0 and 1 $\frac{mol}{m^3}$ and reactor temperatures between 310 and 330 K (see Figure 5.6). Even though the model is highly nonlinear (reaction rate depends exponentially on temperature), the optimal control surface is surprisingly linear with respect to ϕ. With clipping of the ISAT predicted value to meet the control constraints, only one ISAT record is required to store all of the optimal control solutions with an error tolerance of 1.0 K (see Table 5.2). A realistic control problem was set up to test ISAT for a few set point changes. The control horizon is discretized into 1 minute segments. The estimator horizon is 40 minutes and the regulator horizon is 60 minutes. The temperature is sampled every 5 seconds and includes gaussian distribution noise with a standard deviation of 2 K. Concentration is sampled every 10 seconds with a standard deviation of

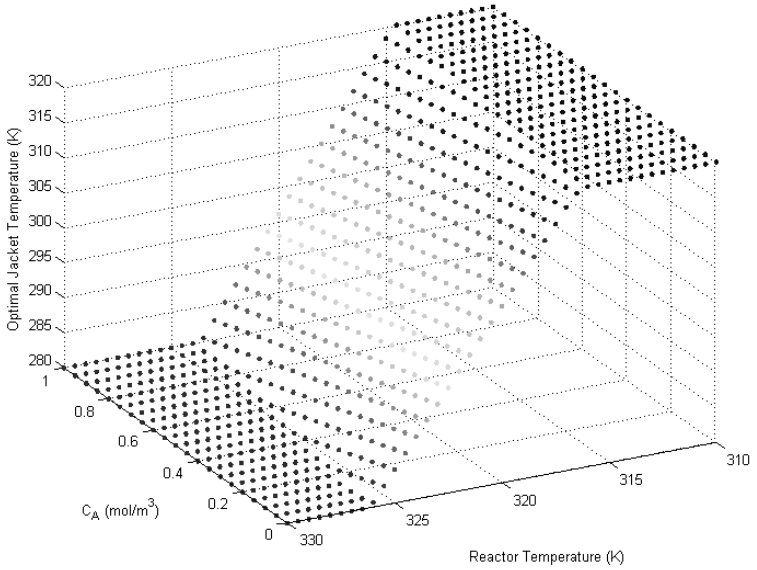

Figure 5.6: The optimal jacket temperature (T_c^*) is a unique function of reactor concentration (c_A), reactor temperature (T), and reactor temperature set point (T_{sp}). In this figure, the set point is fixed ($T_{sp} = 320K$) and c_A and T are varied.

Table 5.2: Elements of the ISAT record for the CSTR example

Element	Value
ϕ^T	$[c_A\ T\ T_{sp}] = [0.9\ 315.0\ 318.0]$
u_0^*	$T_c = 306.8$
A	$[\frac{\partial c_A}{\partial T_c}\ \frac{\partial T}{\partial T_c}\ \frac{\partial T_{sp}}{\partial T_c}] = [-6.227\ -4.081\ 4.889]$
M	$0_{3\times 3}$
CR	N/A

0.1 $\frac{mol}{m^3}$. Plant-model mismatch is introduced by using an activation energy of the first order ($A \rightarrow B$) reaction of 8750 $\frac{J}{mol}$ for the model and 8740 $\frac{J}{mol}$ for the plant. At the first sampling time the plant state is 0.951 $\frac{mol}{m^3}$ and 312.8 K. The estimated model states are 0.9 $\frac{mol}{m^3}$ and 300 K. The initial set point is 315 K. At 50 minutes the set point changes to 300 K, an unreachable set point. At 100 minutes the set point changes to 328 K, close to the NMPC closed loop stability limit. At 150 minutes the set point changes to 308 K (see Figure 5.7). While the two control performances are virtually indistinguishable, ISAT performance is actually slightly better because there is no time delay associated with computing the optimal control solution. Because ISAT operates one step ahead, it responds faster to set point changes and disturbances. This, however, is not the main advantage of using ISAT. The main advantage is that NMPC can be applied to processes with fast sampling times ($<$1 μsec) or simple computers (IC chips). In addition, enumerating the entire control solution off-line can reveal infeasible regions, stability limits, and other closed loop properties. For this example, the CPU times are shown in Figure 5.8. NMPC consists of at least two pricipal calculations: estimation and regulation. The estimator and regulator calculations averaged under 0.1 seconds with maximum calculation time of about 0.3 seconds. Both the estimator and regulator NLPs were solved with the VF13 SQP solver in FORTRAN using a direct single shooting solution approach. All calculations were performed on a 2.0 GHz Celeron processor. The estimation problem is not reduced with ISAT. Parameterizing the current states with all previous measurements is one

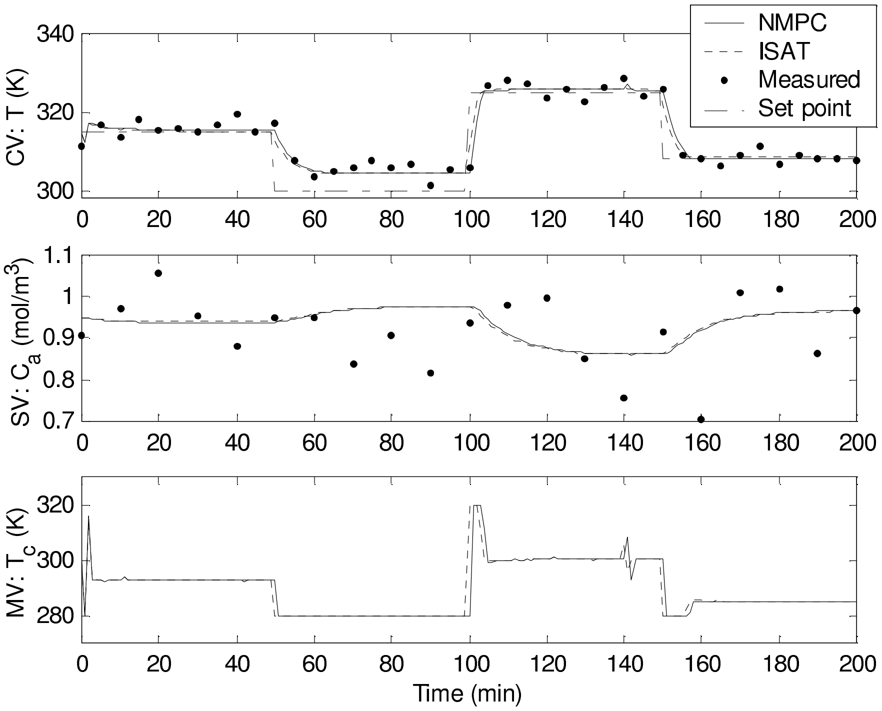

Figure 5.7: Control performance of ISAT compared to NMPC. The controlled variable (CV) is the reactor temperature, the state variable (SV) is the reactor concentration, and the manipulated variable (MV) is the temperature of the cooling jacket.

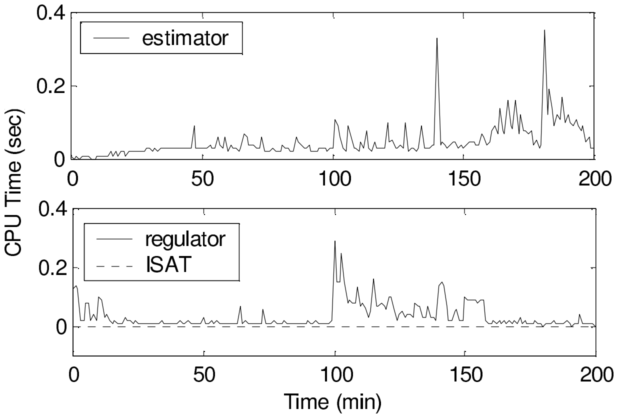

Figure 5.8: Computational times of the estimator and regulator at each sampling instant. ISAT is fast because computation is limited to a matrix multiplication.

possible solution. Another solution improves the solution speed but does not eliminate the on-line complexity of solving a NLP problem at every sampling instant [39].

5.5 Summary and Conclusions

MPC is now suggested as a candidate to replace PID control thanks to recent developments in off-line calculations for efficient on-line implementation. Up to this point, the proposed algorithms suffer from dimensionality problems. For state parameterization, control applications are limited to small models and short control horizons. For constraint parameterization, control applications are limited to short control horizons and low number of inputs. The ISAT algorithm proposed in this work overcomes the dimensionality problems by adaptively storing only those regions accessed in practice. ISAT efficiently handles both NLP problems and constrained LQ problems. ISAT reduces to an adaptive version of state parameterized constrained LQ when the error tolerance is reduced to zero. Future work is needed to efficiently solve the estimation problem on-line.

Chapter 6

Moving Horizon Estimation - The Explicit Solution

Estimation of model states and parameters from process measurements bridges the gap between the theoretical realm of mathematical models and the realistic realm of physical processes. Many approaches have been suggested to accomplish the reconciliation of model and process, with a range of tradeoffs [80]. Generally, the tradeoffs are centered on model form and size, computational expense, ease of implementation, robustness to process/model mismatch, and cultural factors such as understanding and acceptance.

The reconciliation process is an important precursor to other activities such as fault detection, product quality assurance, manual control, and model-based control. These model-based techniques need an accurate estimate of the current system variables to perform well. Without accurate state estimation, many of these tools would perform poorly or fail.

6.1 Previous Work

For dynamic nonlinear model-based control of chemical processes, the most popular feedback strategies in practice are the extended Kalman filter and a constant or integrating output disturbance variable [68]. The Kalman filter is optimal for unconstrained, linear systems subject to known normally distributed state and measurement noise [38]. The Kalman filter sequentially updates state estimates based on the magnitude of the error between the measurements and the model variables. The extended Kalman filter is an extension of the Kalman filter, developed for unconstrained, nonlinear DAE systems [7]. By linearizing the model about updated state estimates, the extended Kalman filter is able to predict the nonlinear state evolution, although sub-optimally [38]. Vachhani et al. proposed EKF with constraints, although the state augmentation strategy for parameter estimation is still a limitation [88].

State estimation of real systems may include changing measurement frequencies, multiple measurements at different sampling frequencies, measurement delay, large-scale nonlinear models, and constraints. Moving horizon estimation (MHE) is an optimization based approach that predicts state trajectories by using a time window that includes the most recent measurements [41] [54] [55] [71]. MHE is also known as nonlinear dynamic data reconciliation (NDDR) [47] [79]. MHE is a computationally tractable approximation to the optimal infinite horizon estimation [70]. All of the challenges of real system state estimation are naturally handled in the MHE framework. An estimate of the current states is typically obtained by solving a least squares optimization problem subject to the model constraints and inequality constraints that represent bounds on variables or equations. Most of the published work centers around different techniques that solve the same minimization problem.

Jang et al. iteratively linearized the nonlinear ODE model about a reference trajectory by computing sensitivities [41]. Liebman et al. first proposed a simultaneous NLP solution approach where the differential equations are transformed into algebraic equations through orthogonal collocation on finite elements [47]. Ramamurthi et al. proposed a two step process to implicitly estimate the input disturbances while explicitly calculating state estimates [69]. Albuquerque and Biegler exploited the MHE SQP structure to achieve linear computational scaling with horizon length for ODE models [3]. They later extended the technique to DAE systems [4].

A number of enhancements have extended the theoretical basis and functionality of MHE. McBrayer and Edgar proposed a bias detection and estimation strategy to improve state estimation [53]. Offset free estimation and control is achieved by augmenting the model with a number of disturbance variables equal to the number of measurements [57] [59]. Rao et al. derived sufficient conditions for MHE with linear systems subject to constraints [70]. They also suggested an infinite horizon approximation by weighting previous state estimates in the least squares problem.

A number of critical evaluations of the extended Kalman filter and MHE for nonlinear systems have been reported [38] [41] [71] . Each group determined that MHE consistently outperforms the Kalman filter and that it exhibits greater robustness to both poor initial state guesses and sub-optimal estimator tuning parameters. Their unanimous conclusion was that the only price of improvement is the greater computational expense required to solve the MHE optimization. The contribution in this work is to eliminate the greater computational expense by developing an explicit solution to the MHE optimization problem. Unlike the implicit optimization approach, the explicit solution result is guaranteed in a highly predictable computational time that is minimal even for large-scale nonlinear models. For state estimation problems with inequality constraints, an iterative procedure is added to determine the set of active constraints. An augmented objective function monitors the solution progression to guarantee convergence.

6.2 Moving Horizon Estimation Problem Formulation

The objective function of the MHE problem is a least squares function that seeks to minimize the difference between the model values and the measurements.

$$\begin{aligned} &\min_{x} J = (Y_s - Y_m)^T Q_y (Y_s - Y_m) \\ &\text{s.t.} \quad 0 = f(\dot{x}, x, u, p) \\ &\qquad\; y_s = g(x, u, p) \\ &\qquad\; a \geq h(x, u, p) \geq b \end{aligned} \tag{6.1}$$

where J is the objective function value, Y_s is a vector of measurements at all nodes, Y_m is a vector of model values at the sampling times, Q_y is the inverse of the measurement covariance, f is a vector of model equation residuals, x is the vector of model states, u is the vector of model inputs, p is the vector of model parameters, y_s is a vector of measurements, g is an output function, h is an inequality constraint function, and a and b are lower and upper limits, respectively. The optimization found in Equation 6.1 can be solved with a variety of numerical approaches [17]. The approach taken in this work is direct single shooting formulation where all future states in the horizon are uniquely specified by the initial state x_0, given sequence of inputs $u = (u_0, u_1, \ldots, u_{n-1})$, and given set of parameters p. At every iteration, the model equations are exactly satisfied.

Sensitivities of the initial conditions are computed to discretize the nonlinear model. In practice, this discretization step is the most computationally expensive part of the MHE calculation. For this study, it is assumed that the discrete model is readily available. The vectors y_m and y_s are successively stacked to form Y_m and Y_s where the horizon length is n.

$$Y_m = \begin{bmatrix} y_{m,0} \\ \vdots \\ y_{m,n} \end{bmatrix}, \quad Y_s = \begin{bmatrix} y_{s,0} \\ \vdots \\ y_{s,n} \end{bmatrix} \tag{6.2}$$

An infinite horizon approximation is added by incorporating a penalty on the deviation from previous model estimates. This penalty is added by augmenting the objective function with the least squares contribution of previous model estimates $\hat{X}_m$, weighted with a forgetting factor α. Disturbance variables (shown here as input disturbances), d, are included as state variables to achieve offset free estimation and control. The nonlinear inequality constraints are simplified by defining new states $z_k = h(x_k, u_k, p_k)$ and imposing inequality constraints on z_k.

$$\begin{aligned}
&\min_{x_0} J = (X_s - X_m)^T Q_x (X_s - X_m) + \alpha \left(\hat{X}_m - X_m\right)^T \left(\hat{X}_m - X_m\right) \\
&\text{s.t.} \quad \begin{bmatrix} x_{k+1} \\ d_{k+1} \\ p_{k+1} \end{bmatrix} = \begin{bmatrix} A_k & B_k & P_k \\ 0 & I & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} x_k \\ d_k \\ p_k \end{bmatrix} + \begin{bmatrix} B_k \\ 0 \\ 0 \end{bmatrix} u_k \\
&\qquad y_{s,k} = \begin{bmatrix} C_k & 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ d_k \\ p_k \end{bmatrix} + D_k u_k \\
&\qquad z_k = h(x_k, u_k, p_k) \qquad a \geq z_k \geq b
\end{aligned} \tag{6.3}$$

The vectors $\hat{X}_m$ and X_m are composed of model vectors $\hat{x}_m$ and x_m. Also, X_s is constructed from the measurements (see Equation 6.4b) and $Q_{x,k} = \left(C_k^T Q_{y,k} C_k\right)$.

$$X_m = \begin{bmatrix} x_{m,0} \\ \vdots \\ x_{m,n} \end{bmatrix}, \quad \hat{X}_m = \begin{bmatrix} \hat{x}_0 \\ \vdots \\ \hat{x}_n \end{bmatrix} \tag{6.4a}$$

$$X_s = \begin{bmatrix} \left(C_0^T C_0\right)^- C_0^T \left(y_{s,0} - D_0 u_0\right) \\ \vdots \\ \left(C_n^T C_n\right)^- C_n^T \left(y_{s,n} - D_n u_n\right) \end{bmatrix} \tag{6.4b}$$

Solution of this optimization problem is typically accomplished with an implicit solution technique. The next section develops an explicit solution to this problem.

6.3 The Explicit MHE Solution

For simplicity of the derivation, the augmented state matrix is reduced to a generic linear time-varying form of $x_{k+1} = A_k x_k + B_k u_k$ and $y_k = C_k x_k + D_k u_k$. All variables are in deviation form although not explicitly indicated here. The model evolution is a unique function of the initial states.

$$\begin{aligned}
&\omega_k = \prod_{j=0}^{k-1} A_j \quad \psi_k = \sum_{j=1}^{k} \left[\prod_{i=1}^{j-1} A_{i-k-j}\right] B_{k-j} u_{k-j} \\
&\Omega = \begin{bmatrix} I \\ A_0 \\ \vdots \\ A_{n-1} \dots A_0 \end{bmatrix} = \begin{bmatrix} I \\ \omega_1 \\ \vdots \\ \omega_{n-1} \end{bmatrix} \\
&\Psi = \begin{bmatrix} 0 \\ B_0 u_0 \\ \vdots \\ A_{n-2} \cdots A_1 B_0 u_0 + \cdots + B_{n-1} u_{n-1} \end{bmatrix} = \begin{bmatrix} 0 \\ \psi_1 \\ \vdots \\ \psi_{n-1} \end{bmatrix} \\
&X_m = \Omega x_0 + \Psi
\end{aligned} \tag{6.5}$$

The equations of X_m and Y_m are substituted into the objective function, making it a unique function of x_0. The explicit solution to the minimization problem is obtained by differentiating the objective function with respect to x_0, setting the differentiated function equal to zero, and algebraically manipulating the equation to solve for the estimated x_0 ($\hat{x}_0$) explicitly.

$$\hat{x}_0 = \left(\Omega^T \left(Q_x + \alpha I\right) \Omega\right)^{-1} \left(\Omega^T \left(Q_x X_s + \alpha X_m - \left(Q_x + \alpha I\right) \Psi\right)\right) \tag{6.6}$$

The explicit solution can be calculated when the inverse of $\Omega^T \left(Q_x + \alpha I\right) \Omega$ exists. The inverse exists when previous estimates are used to approximate the infinite horizon solution ($\alpha > 0$). An explicit solution does not exist when the system is unobservable and $\alpha = 0$. This property is consistent with the fact that an unobservable system possesses extra degrees of freedom leading to states that cannot be estimated from the available measurements. A more detailed definition of necessary and sufficient conditions for convergence is provided below.

Conditions for a unique solution are given separately for observable and unobservable systems. Also, the conditions apply equally for linear and nonlinear systems. Observability of nonlinear systems is established by analyzing the local observability along a reference trajectory [84]. As long as local observability is maintained, the nonlinear system is also completely observable.

6.3.1 Fully observable systems

To obtain a unique solution for fully observable systems, the following necessary conditions must be met:

1. Q_y has non-zero eigenvalues
2. The number of measurements is greater than or equal to the number of states, n

Without an infinite horizon approximation ($\alpha = 0$), the explicit solution reduces to the form derived in Ramamurthi et al. [69]. When the number of measurements is less than n, ω becomes rank deficient. With ω rank deficient, the product $\omega^T Q_y \omega$ is also rank deficient. Also, the product $\omega^T Q_y \omega$ becomes rank deficient when Q_y has at least one zero eigenvalue. Q_y can have a zero eigenvalue when zero weighting is given to a measurement. This situation may arise when particular measurements are eliminated from the optimization due to temporary faults in the sampling equipment or transmission delay.

Sufficient conditions guarantee a unique solution and consist of:

1. Q_y is positive definite
2. The estimation horizon is greater than or equal to the number of states, n

For observable systems, the observability matrix is full rank. When the horizon is equal to n, ω is exactly the observability matrix and therefore, full rank. When Q_y is positive definite, the product $\omega^T Q_y \omega$ is also positive definite. A positive definite matrix is invertible, so a unique solution exists to the eMHE problem.

6.3.2 Partially observable systems

For systems with unobservable states, convergence is guaranteed when the following conditions are met:

1. Q_y is positive definite
2. The objective function is augmented with an infinite horizon approximation ($\alpha > 0$)
3. ω is full rank

When Q_y is positive definite, Q_x is positive semi-definite. Adding an infinite horizon approximation increases by α the singular values (equivalent to the eigenvalues for symmetric positive semi-definite matrices) of $Q_x + \alpha I$. With ω full rank, the product $\omega^T(Q_x + \alpha I)\omega$ is positive definite and convergence is guaranteed.

6.3.3 Example 1: Explicit versus Implicit MHE Solution

This first example is used to illustrate that explicit and implicit solutions of the MHE problem give the same results. A single input, single output (SISO) second order model with stable roots is specified as the system.

$$G(s) = \frac{1}{s^2 + 2s + 1} \tag{6.7}$$

A conversion to discrete time is performed with a sampling frequency of 0.1 seconds. Normally distributed measurement noise with mean of zero and standard deviation of 0.1 is added to the output.

$$\begin{aligned} x_{k+1} &= \begin{bmatrix} .8144 & -0.0905 \\ 0.0905 & 0.9953 \end{bmatrix} x_k + \begin{bmatrix} 0.0905 \\ 0.0047 \end{bmatrix} u_k \\ y_k &= \begin{bmatrix} 0 & 1 \end{bmatrix} x_k + v_k \end{aligned} \tag{6.8}$$

The first state x_1 is unmeasured, but observable. The second state, x_2, is measured but corrupted by measurement noise. The states are both initially at zero while the initial guesses of the states are both set to one. A forgetting factor of 0.5 is added to the initial state in the time horizon. Figure 6.1 shows the results of 49 separate optimizations for both explicit MHE and optimization based MHE (labeled as MHE). Starting at time zero, every sampling instant MHE recalculates a new estimate of the current states. The explicit solutions agree closely with the implicit solutions. State x_1 converges quickly to the actual system values. State x_2 also gradually converges to the correct solution because of the forgetting factor that places weight on the erroneous initial guess. The computational effort required to compute a solution is drastically different. The explicit solution required 2506 floating point operations. On a

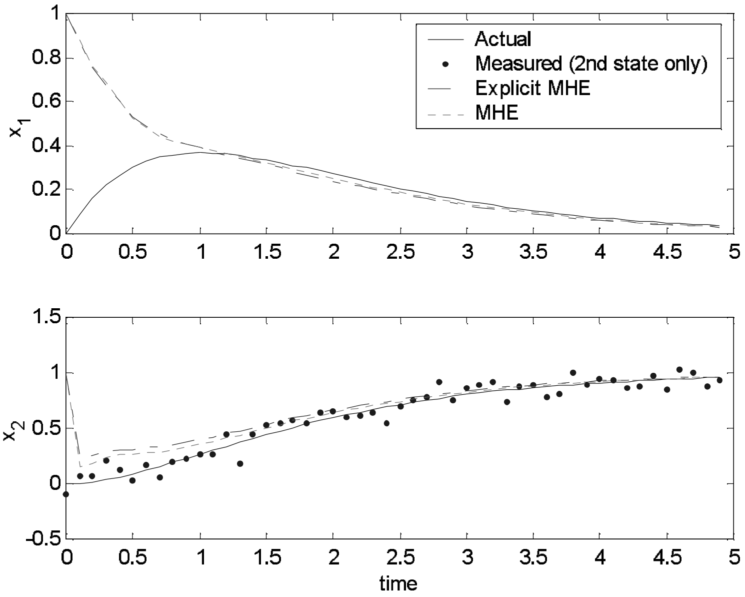

Figure 6.1: The explicit and implicit MHE solutions produce the same results. Substantial computational reduction is obtained with the explicit solution approach.

modern day computer operating at 10^3 million floating point operations per second (MFLOPS), the solution would require 0.0003 seconds. The implicit solution required 785,716 floating point operations for the last optimization with a horizon of 50 measurements, or roughly 0.08 seconds on modern computers. For this example, the explicit solution reduces computational requirements of MHE by 314 times.

6.4 Inequality Constraints in Explicit MHE

Ramamurthi et al. [69] did not include inequality constraints in deriving an explicit MHE solution. Inequality constraints represent physical limits on state variables or combinations of state variables. For example, mole fractions are always between 0 and 1. If the state estimation predicted a mole fraction outside of this range, that mole fraction would have little physical meaning and would decrease the credibility of the other results. Inequality constraints add valuable information to the state estimation. For systems that are partially unobservable, the inequality constraints bound the unobservable states, thereby increasing the level of system observability. However, an unobservable system cannot be made completely observable with inequality constraints. Additional actual measurements are the only way to make an unobservable system completely observable.

As previously mentioned, the inequality constraints $a \geq h(x_k, p_k) \geq b$ are simplified by creating new variables z_k and adding $z_k = h(x_k, p_k)$ to the set of state equations. Equivalent constraint information is retained by imposing inequality constraints on z_k ($a \geq z_k \geq b$). Imposing constraint information leads to a possible infeasible solution. To overcome this possibility, the inequality constraints are ranked according to the order of importance. This ranking is accomplished by softening the constraints and imposing successively higher weighting on more important constraints. Softening the constraints guarantees a feasible solution because the inequality constraints may be violated to meet the state equality constraints. Softening of the constraints is performed in practice by adding a penalty to the objective function for constraint violation.

$$\begin{aligned} &\min_{X_m} J + s_a^T Q_a s_a + s_b^T Q_b s_b \\ &\text{s.t.}, 0 = f(\dot{x}, x, u, p) \\ &\qquad y_s = g(x, u, p) \\ &\qquad s_a = a - X_m \\ &\qquad s_b = X_m - b \end{aligned} \tag{6.9}$$

The matrices Q_a and Q_b have diagonal elements that turn on (weighting > 0) or off (weighting $= 0$) to control the set of active constraints. A MHE problem with inequality constraints is iterative because the final set of active constraints is not known a priori. However, the prediction of states, disturbances, and parameters is still an explicit solution for a known set of active inequality constraints. The computational time required to solve a problem with inequality constraints is variable, equal to the time required for one explicit solution multiplied by the number of iterations. The explicit solution subject to the set of active inequality constraints is given in Equation 6.10.

$$\begin{aligned} &\hat{x}_0 = T^{-1} \left(\Omega^T \left(Q_x X_s + \alpha X_m + Q_a a + Q_b b - R\Psi \right) \right) \\ &\text{with } R = \left(Q_x + \alpha I + Q_a + Q_b \right) \text{ and } T = \left(\Omega^T R \Omega \right) \end{aligned} \tag{6.10}$$

Convergence is guaranteed by rejecting iterations that do not produce a sufficient decrease in the objective function. Each iteration yields a new set of active constraints that are predicted to give a decrease in the objective function. The initial set of active constraints is determined by computing an unconstrained MHE solution. Weighting is added to Q_a and Q_b for states that violate the inequality constraints. In successive iterations, weighting is removed for constraints with negative Lagrange multipliers ($\lambda_a = -2Q_a s_a$ and $\lambda_b = 2Q_b s_b$. If the new set of active constraints does not give a sufficient decrease in the objective function, the algorithm adjusts the weights on the active constraints. The parameter β in Equation 6.11 is reduced until a decrease in the objective function is obtained. An objective function decrease is always possible until convergence. However, a full step ($\beta = 1$) may not give a decrease in the objective function because of the nonlinear nature of constrained systems. The Lagrange multipliers are locally accurate, linear predictions of active constraint effects on the objective function. In the limit as *beta* approaches zero, the linear approximation becomes exact and therefore, a decrease in the objective function is guaranteed. Convergence of MHE with constraints is guaranteed by successively decreasing β until a sufficient decrease in the objective function is obtained at every iteration.

$$\begin{aligned} Q_a &= \beta Q_{a,k} + (1-\beta) Q_{a,k-1} \\ Q_b &= \beta Q_{b,k} + (1-\beta) Q_{b,k-1} \end{aligned} \tag{6.11}$$

Once a decrease in the objective function is obtained, the optimizer accepts the iteration and moves on to find a new search direction (new active set of constraints that is predicted to produce a decrease of the objective function).

This iterative sequence is terminated when the active set does not change from one iteration to the next.

6.4.1 Example 2: Constrained Version of Example 1

Constraints are added to the Example 1 problem to demonstrate the active set strategy. State x_1 is arbitrarily constrained between 0 and 0.2. State x_2 is not constrained. An input disturbance variable x_3 is added to achieve offset free estimation. Figure 6.2 shows the results of MHE with and without inequality constraints. The unconstrained solution is the first iteration for explicit MHE calculation. After 5 iterations (for each of the 49 separate

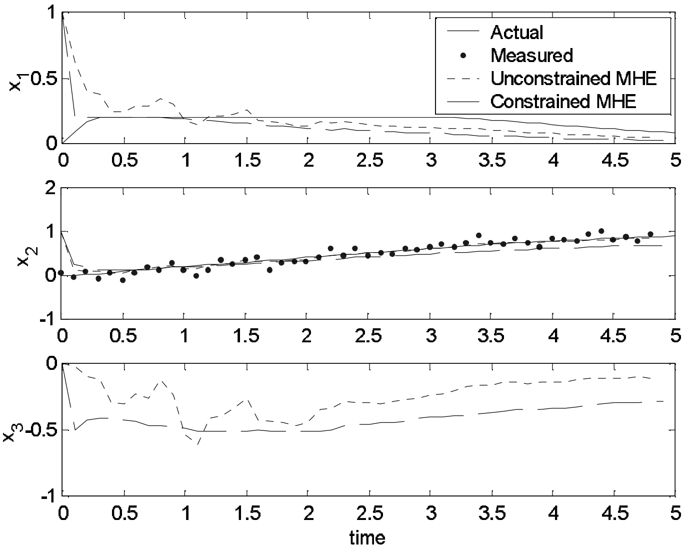

Figure 6.2: Variable x_1 is constrained between 0 and 0.2. The unconstrained solution violates the upper bound constraint on x_1. By incorporating the constraints the solution is improved.

optimizations), constrained explicit MHE (CE MHE) agrees closely with the implicit solution (MHE) as seen in Figure 6.3. Using the active set strategy

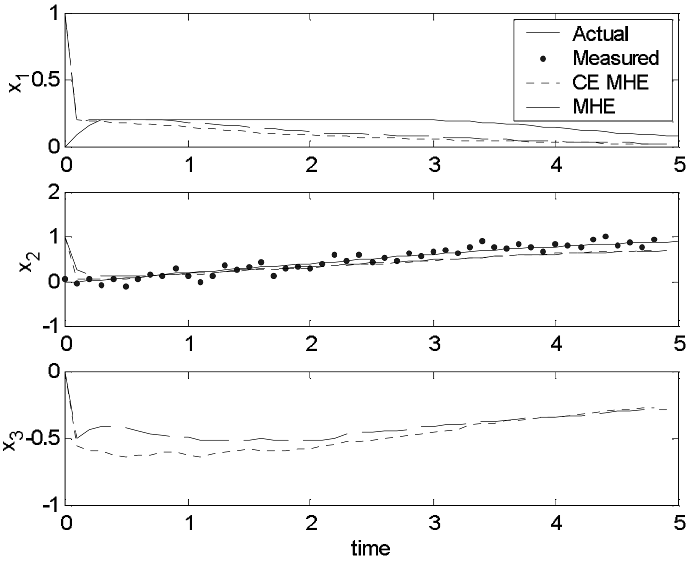

Figure 6.3: Results for constrained explicit MHE (CE MHE) compared to optimization based MHE (labeled as MHE). Note that x_1 now meets the constraint condition.

proposed in this section, explicit MHE converges to the constrained solution. The details of the convergence are not obvious because the only values reported in Figure 6.3 are the final predicted values. An iteration by iteration sequence is informative to show the convergence properties. The last optimization is taken as a example using all 50 data points with a horizon of 49. Figure 6.4 shows the first iteration. The third state (x_3) is the input disturbance variable. At the first iteration, the upper bound of x_1 is violated by the unconstrained explicit MHE solution. Once the unconstrained solution is computed, a search

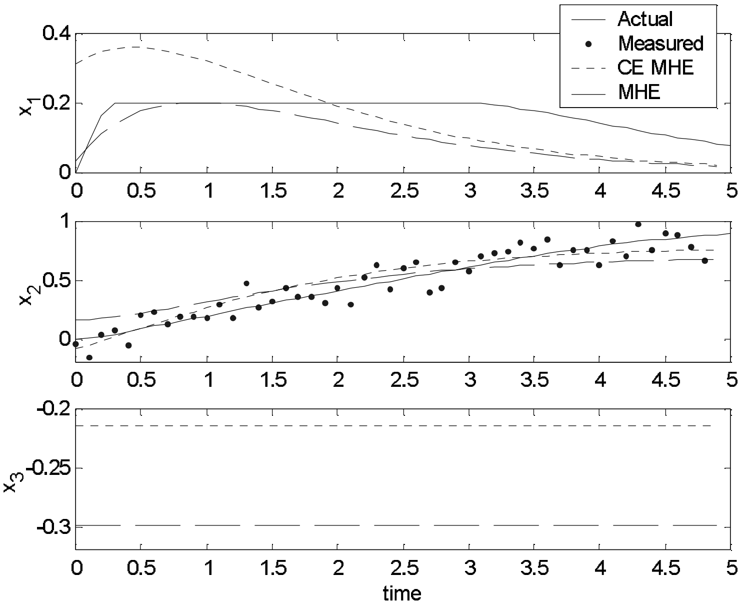

Figure 6.4: First iteration of the explicit MHE solution. State x_1 violates the upper bound constraint of 0.2.

is performed to identify all of the constraint violations. For the next iteration, penalties are added to the objective function for each of the violations. Figure 6.5 shows the results of the explicit solution with weighting placed at the former constraint violations. Negative Lagrange multipliers are identified for the next

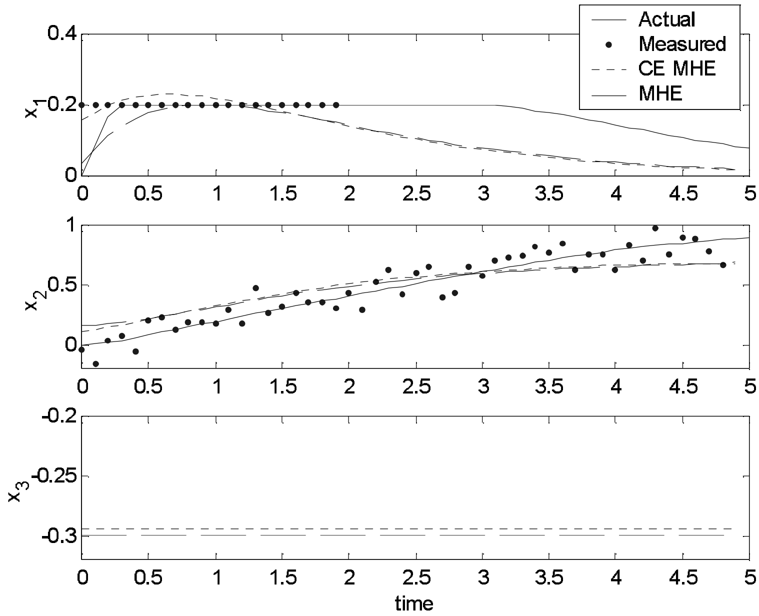

Figure 6.5: Second iteration for explicit MHE. Weighting is added to matrix Q_b where the x_1 trajectory formerly violated the upper bound constraint of 0.2. The constraint information is indicated as fictitious measurements even though x_1 is not actually measured.

iteration to determine the equality constraints that should be removed from the active set. Negative Lagrange multipliers are found between $0.0 \geq time \geq 0.2$ and $1.3 \geq time \geq 1.9$. For the next iteration these constraints are removed from the active set. No additional constraint violations are identified so no constraints are added to the active set. A decrease in the augmented objective function indicates that the iteration should be accepted. Figure 6.6 shows the third iteration. After the third iteration, negative Lagrange multipliers are found between $0.3 \geq time \geq 0.4$ and $1.1 \geq time \geq 1.2$. For the next iteration these constraints are removed from the active set and the explicit MHE is recalculated, resulting in a sufficient decrease in the augmented objective function. Figure 6.7 displays the fourth iteration. After the fourth iteration, negative Lagrange multipliers are found between $0.5 \geq time \geq 0.6$. These constraints are removed from the active set for the final iteration. Again, the augmented objective function decreases. Figure 6.8 displays the final iteration.

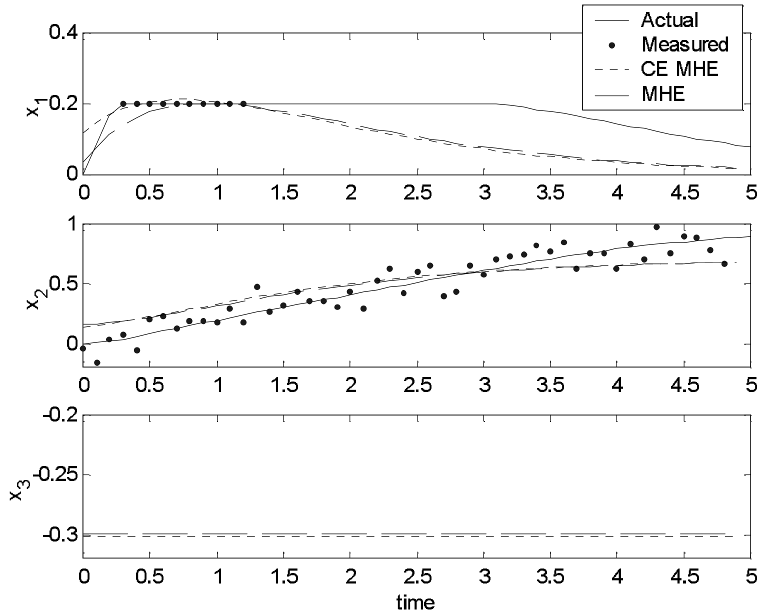

Figure 6.6: Third iteration of explicit MHE. The explicit MHE solution is approaching the implicit MHE solution as the active set of constraints is refined.

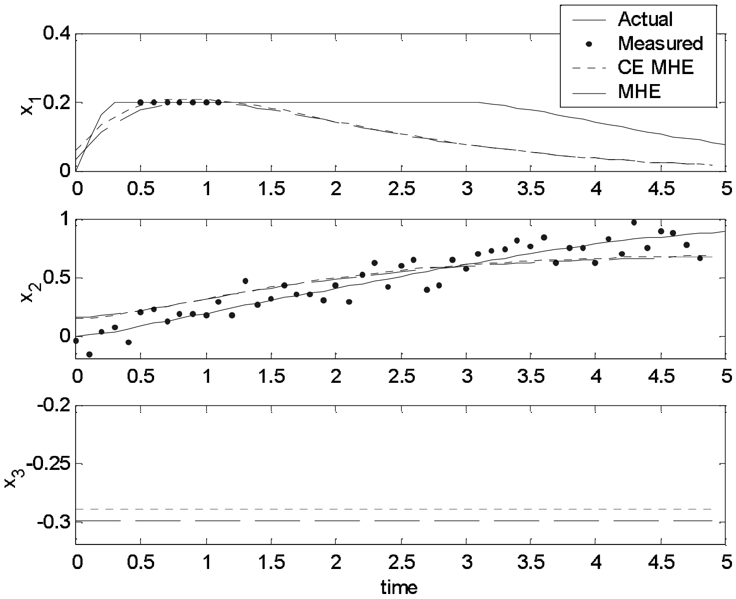

Figure 6.7: Fourth iteration of explicit MHE. The explicit MHE solution almost agrees exactly with the implicit MHE solution. The final solution requires one more iteration for convergence.

The final iteration of explicit MHE shows excellent agreement between both

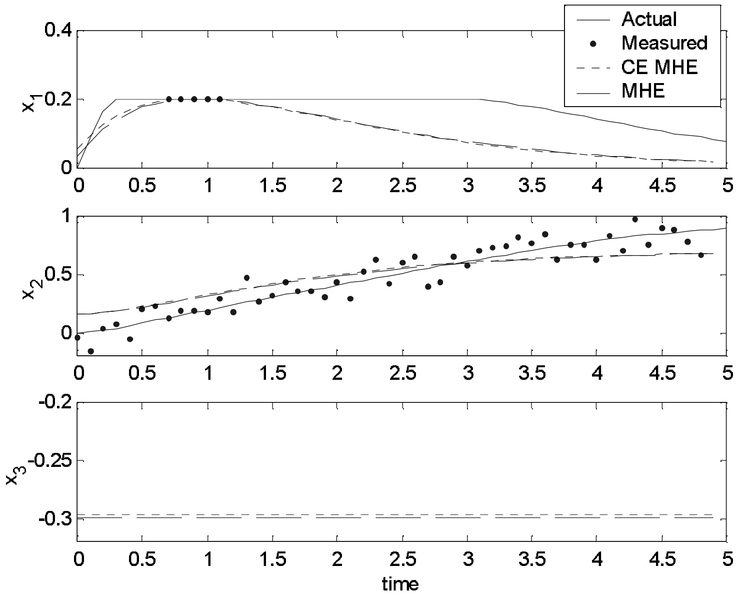

Figure 6.8: Final iteration of explicit MHE. In all, the explicit solution sequence required 5 iterations for convergence.

solutions. The input disturbance variable x_3 is defined as constant over the entire time horizon. This example shows the development of MHE for a constrained linear system, making the problem nonlinear. The MHE framework naturally incorporates constraints into the problem formulation. The explicit solution required approximately 2500 floating point operations per iteration for a total of 12,500. The implicit solution required approximately 122 million floating point operations before the optimizer reached the default maximum number of iterations. The inability of the implicit solution to converge quickly is most likely due to the deficiencies of direct single shooting with softened constraints. A more robust approach obtained by direct multiple shooting or collocation of the state equations would undoubtedly improve the implicit solution computational effort, but not to the level of the explicit MHE approach.

6.5 Example 3: Flash Column Composition Estimation

This section shows an example of MHE, but for a physically realistic process, a 17 state model of a flash column (see Figure 6.9. The unknown compositions are estimated from the temperature and flow rate measurements. When the liquid stream enters the column at a pressure below the liquid's vapor pressure, a fraction of the stream instantaneously flashes into the vapor phase. A rigorous nonlinear model of mass, energy, and thermodynamic equilibrium relationships predicts the dynamic behavior of the column. A diagram of the model is presented in Figure 6.9. The feed tank contains an equimolar hydrocarbon mixture of C_4H_{10}, C_5H_{12}, C_6H_{14}, C_7H_{16}, and C_8H_{18}. The feed and flash temperatures are measured as are the vapor and liquid flow rates.

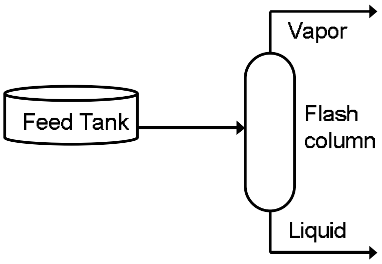

Figure 6.9: Flow sheet diagram of the flash column model. The flash column consists of a feed tank with unknown species compositions, a flash column, and vapor and liquid outlet streams.

Noise is added to the measurements with mean zero and standard deviation 0.5 for the temperatures and 0.02 for the flow rates. The objective is to estimate the feed tank compositions from the temperature and flow rate measurements. Figure 6.10 shows the measurements taken over the time horizon of interest. The 17 state model has 5 differential states and 12 algebraic states. For full observability, the observability matrix must have rank ≥ 5. For this example, the observability matrix is rank deficient at 3. This analysis reveals that temperature and flow measurements of a flash column can only be used to exactly estimate compositions of mixtures with ≤ 3 components. Alternatively, 2 additional compositions could be measured to make the system observable.

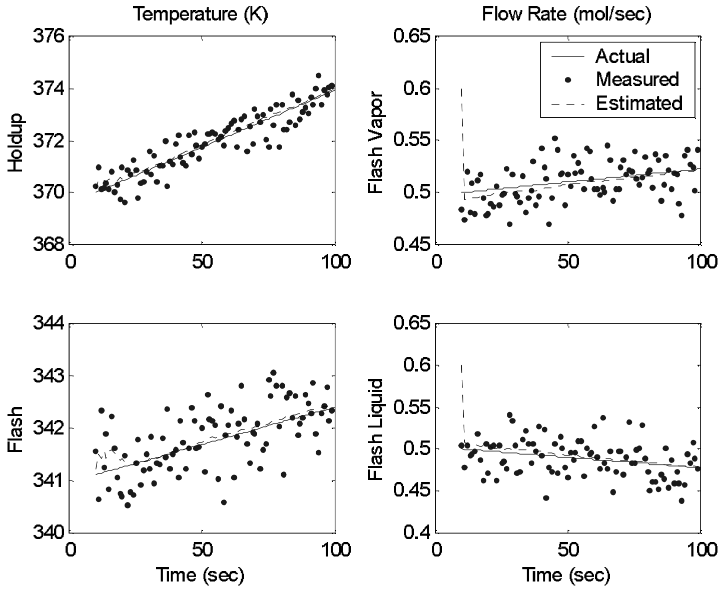

Figure 6.10: The estimated states converge quickly to the real system even though the initial guess is poor.

However, even if the system is not fully observable, some information can be reconstructed that can be better than the initial composition estimates. For this example, the compositions are initially estimated as 0.3 whereas the actual compositions are all located at 0.2. Figure 6.11 shows the estimation of the compositions over a 100 second time horizon. A forgetting factor of 0.5 on

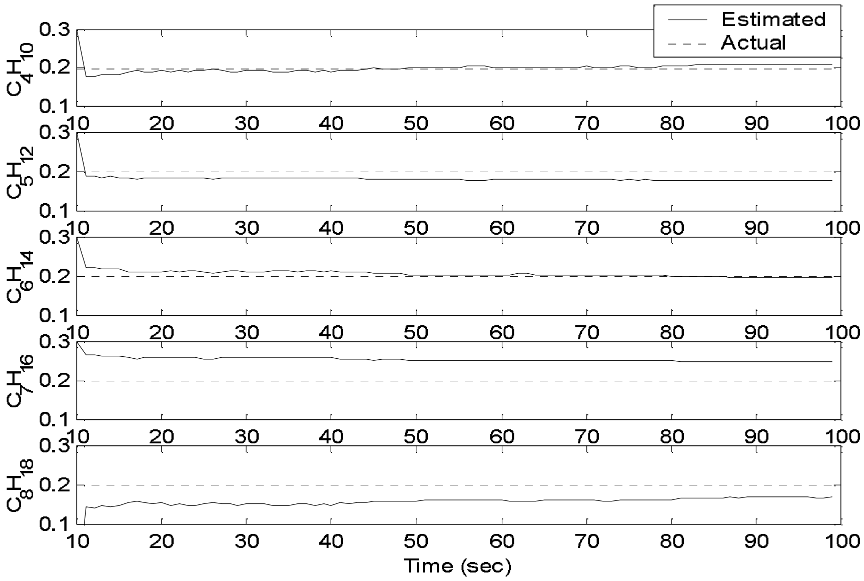

Figure 6.11: Estimated compositions of C_4H_{10}, C_5H_{12}, and C_6H_{14} approach the actual values of 0.2. The other two compositions of C_7H_{16} and C_8H_{18} deviate significantly because the system is not fully observable.

the initial state was used to incorporate previous estimates. The estimation is able to reconstruct the compositions of C_4H_{10}, C_5H_{12}, and C_6H_{14}. However, the other two compositions, C_7H_{16} and C_8H_{18}, deviate significantly from the correct solution. This deviation is a result of an unobservable system.

Inequality constraints can bound unobservable states to increase the accuracy of the estimation. For this example, suppose it is known that the composition of C_7H_{16} should not be above a composition of 0.22. This constraint information can be incorporated into the explicit MHE formulation to provide a better estimate of compositions. Figure 6.12 shows the results of bounding the C_7H_{16} composition. At the final solution the active constraint on C_7H_{16} composition has a Lagrange multiplier of +0.02, confirming that the constraint should be active. The estimation of the composition is greatly improved by incorporating additional information about the process in the form of an inequality constraint.

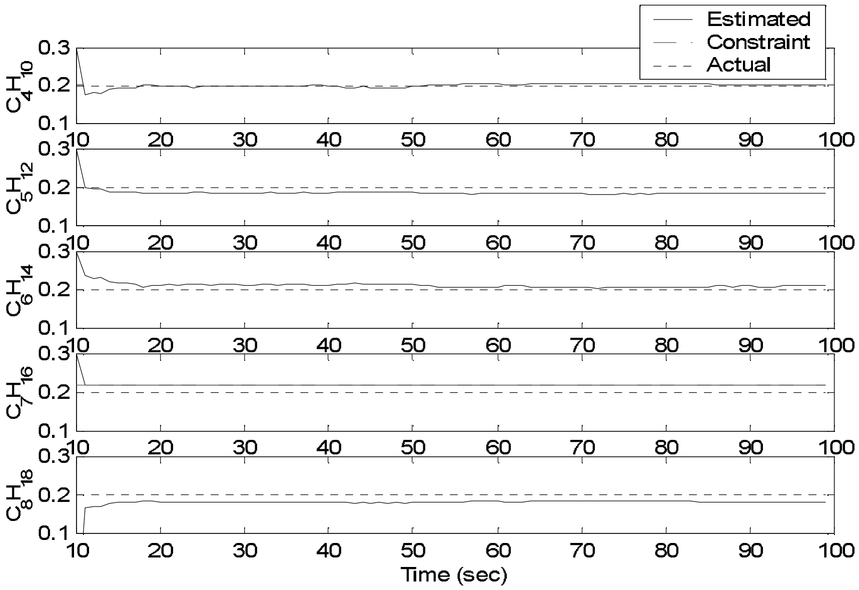

Figure 6.12: The composition estimation is greatly improved by adding an inequality constraint to C_7H_{16}. Even though the system is not fully observable, the composition estimates closely approximate the actual values.

6.5.1 Explicit MHE Scaling with Model Size

An important property of explicit MHE is computational scaling to large-scale problems. To test the scalability to large-scale problems, a series of 17 state flash columns are combined to form larger models. These successively larger models are solved for the linear and nonlinear case as seen in Figure 6.13. A horizon of 50 samples is used for all of the simulations. Both linear and nonlinear explicit MHE scale $O(x^2)$ in the number of floating point operations, although the linear approach scales approximately 6 times better than the nonlinear method. With computers that operate in the Gigahertz range, the computational feasibility of explicit MHE is excellent even for large-scale

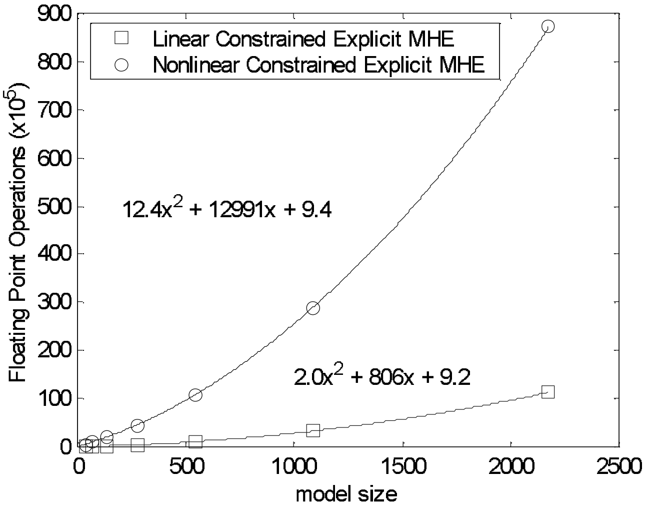

Figure 6.13: Explicit MHE scaling to large-scale model size. Both the nonlinear and linear approaches scale $O(x^2)$ in the number of floating point operations, where x is the number of variables in the model.

problems (10,000+ variables).

6.5.2 Explicit MHE Scaling to Long Time Horizons

Some estimation problems require long time horizons ($>$ 100 sampling intervals). Long time horizons may be necessary when the measurements have low signal to noise ratios, process measurements occur much faster than the process dynamics, or there is a large difference among the sampling frequencies of multiple measurements. Another reason for a long time horizon is for parameter estimation where a few parameters are estimated from a long time period of historical data. Figure 6.14 displays the effect of time horizon length on the number of floating point operations for the 17 state flash column model. For nonlinear models, the scaling is quadratic for increasing horizon length. For linear models the scaling is linear for increasing horizon length. The linear model scaling is particularly amenable for problems that may require a very long time horizon.

6.6 Example 4: Two State CSTR

State estimation of a CSTR is a popular benchmark test problem as found in Albuquerque and Biegler [3] and Haseltine and Rawlings [38], inter alii. A description of the model, variables, and equations is given in Appendix B. The purpose of this example is to estimate the computational load for different estimation strategies.

A realistic estimation problem was devised to test eMHE for a sequence

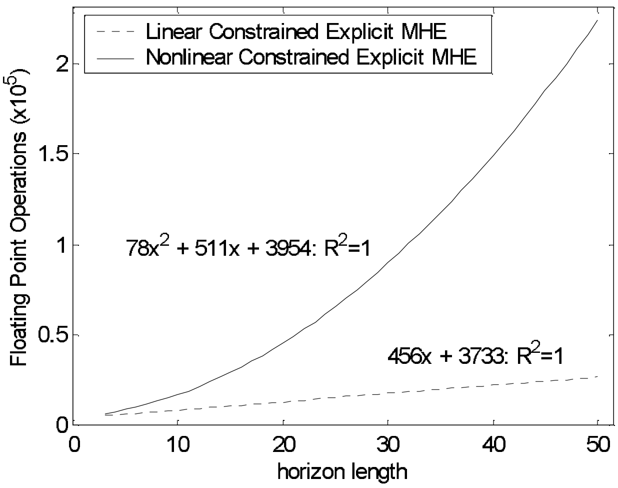

Figure 6.14: Explicit MHE scaling to horizon length. For nonlinear models, scaling is $O(x^2)$ in the number of floating point operations. For linear models, scaling is $O(x)$ where x is the horizon length.

of step responses. The estimator horizon is set to 60 minutes and divided into 1 minute segments. The temperature is sampled every minute and corrupted by normally distributed noise with a standard deviation of 5 K. Concentration is sampled every 10 minutes with a standard deviation of 0.01 $\frac{mol}{m^3}$. Plant-model mismatch is introduced by using an activation energy of the first order ($A \rightarrow B$) reaction of 8750 $\frac{J}{mol}$ for the model and 8740 $\frac{J}{mol}$ for the plant. The plant-model mismatch is introduced to cause deviation of the estimated response from the actual process. The steady state deviation can be eliminated by including parameter estimation or a disturbance variable. At the first sampling time the plant is assumed to be at steady state with a jacket cooling temperature of 300 K. At 20 minutes the cooling temperature is set to 290 K, followed by a step to 310 K at 60 minutes. At 70 minutes the cooling temperature returns to 290 K. Figure 6.15 shows the results of the MHE study. The eMHE solution averaged approximately 22,000 floating point operations

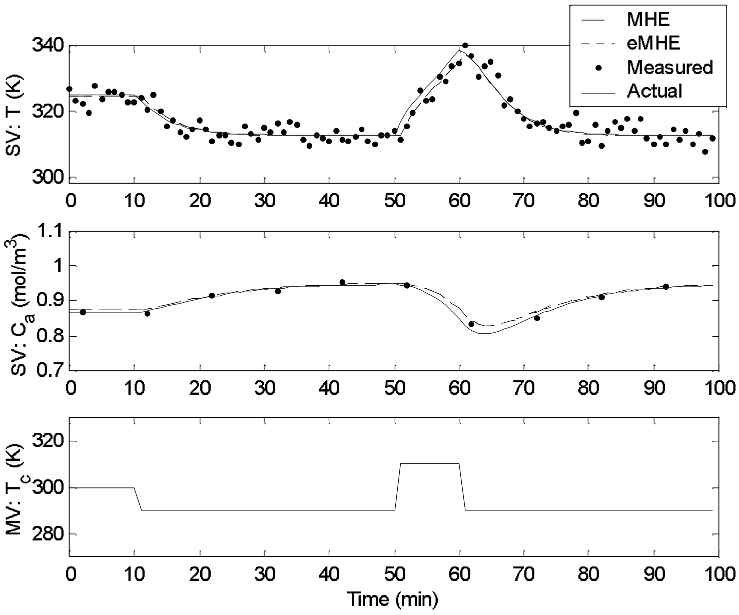

Figure 6.15: Estimation performance of the explicit solution MHE (eMHE) versus MHE. The state variable (SV) estimation is difficult to distinguish on the graph because the predictions are virtually identical for the two approaches. The only difference is the substantially lower computational effort required to reach a solution.

to compute a solution. The direct single shooting optimization MHE solution averaged approximately 40 million floating point operations. The CPU time results from Liebman et al. were performed on a computer that delivers approximately 1 MFLOPS with LINPACK benchmark tests [47]. He reported in 1992 solution times in the range of 1-100 seconds giving approximate computational effort in the range of 1-100 million floating point operations for sparse solvers and orthogonal collocation on finite elements. The explicit solution approach offers improved computational performance that is insensitive to convergence tolerance, poor initial conditions, strong nonlinearities, and other factors that influence the implicit solution approach.

6.7 Conclusions

Moving horizon estimation has been demonstrated to be a superior state estimation technique compared with the extended Kalman filter. The only disadvantage is the additional computational expense needed to solve the MHE optimization problem. This chapter outlines an explicit solution technique that removes the computational disadvantage for large scale nonlinear DAE systems that is guaranteed to converge when the system is fully observable or when previous estimates are incorporated into the optimization. Inequality constraints add variable bounds that can improve the state estimation, especially for systems that are not fully observable. An iterative approach is necessary to determine an active set of equality constraints from the full set of inequality constraints. The iterative solution has guaranteed convergence by selecting new active sets that generate a sufficient decrease in the augmented objective function. The computational expense of the most challenging problem in this chapter required 22,000 floating point operations, only a few micro-seconds with modern computational power. The computational expense of implicit optimization MHE is significantly more, with a possibility of convergence failure depending on the initial conditions selected, problem nonlinearity, choice of optimizer, etc.

Chapter 7

Conclusions and Recommendations

7.1 Summary of Contributions

The main focus of this dissertation is to reduce the computational requirements for large-scale DAE model-based estimation and control. This objective is accomplished by a variety of strategies that are combined in an effective way to meet real-time constraints with limited computing resources. The principal strategies are storage and retrieval off-line to enable efficient on-line control, nonlinear DAE model reduction, and development of explicit optimization solutions. Both moving horizon estimation and receeding horizon control are developed to meet real-time constraints.

7.1.1 Development of a Superior Alternative to Neural Networks for Nonlinear Function Approximation

ISAT, as formerly applied in combustion applications, was infeasible as a general nonlinear function approximator because it required sensitivity information. A statistical approximation to the sensitivity allows ISAT to store and retrieve any linear or nonlinear function with error control.

7.1.2 Automatic, Iterative DAE Model Reduction

Automatic order reduction of the differential equations is made possible by introducing a variable error predictor that iteratively determines the order of the reduced model. This variable error predictor can also be used to further reduce the model order by increasing the reduced order model accuracy. For the algebraic equations, a partitioning and precedence ordering is performed to divide the equations and variables into successively independent sets. Once partitioned, ISAT stores and retrieves solutions to the subsets of equations. By building a database of solutions, the equations are automatically explicitly transformed. The automatic transformation avoids the susceptability to error that would come from a manual explicit transformation on an equation by equation basis.

7.1.3 Reduced Computational Effort for Nonlinear MPC

The two most computationally expensive parts of nonlinear MPC are state estimation and regulation. Reduction by up to 100 times is possible by storing and retrieving the continuous nonlinear DAE integrations.

The state estimation is a dynamic data reconciliation of the model prediction and plant measurements. State estimation is a necessary step when full state feedback is impossible, plant-model mismatch is present, or measurements are corrupted with noise. By storing and retrieving DAE model integrations and sensitivities, the estimator calculation speed is greatly improved.

The regulator uses the results of the estimator to determine a set of optimal inputs that will minimize an objective function. By using ISAT to compute the model state evolution constraints, the computational time of ISAT is reduced by up to 100 times.

7.1.4 Development of Explicit Nonlinear MPC

By storing and retrieving optimal control solutions, an approximate explicit NMPC controller is developed. The error control embedded in ISAT ensures that variable error tolerances are not exceeded. For constrained QP problems, the initial estimate of the ROA is restricted to the state space with the same active set of constraints. In the limit as ϵ_{tol} goes to zero, the algorithm yields the exact explicit MPC solution. The dimensionality problems of previous approaches is overcome by only storing and retrieving solutions that are accessed in practice.

7.1.5 Development of Explicit Moving Horizon Estimation

Moving horizon estimation requires a solution to a NLP problem comparable in computational complexity to the control formulation. In order for model-based control to function, a current estimate of the states is necessary before every control optimization. An explicit solution to the MHE problem is developed for nonlinear ODE or DAE models. Inequality constraints can be enforced by iteratively defining the active set of variables at the constraint bounds. An explicit solution is guaranteed by weighting previous estimates in the objective function. Disturbances and parameters are simultaneously estimated with the states in one explicit solution.

7.2 Future Work

Several extensions of this work are possible. The proposed future work involves model reduction for the non-expert user, new applications for model reduction, a few suggested developments for ISAT, and explicit MHE for parameter estimation.

7.2.1 Model Reduction for the Non-Expert User

Automated model reduction for non-expert users can be incorporated into popular DAE solvers with the new techniques presented in Chapter 3. A user would specify an error tolerance for the original model variables and the solver could iteratively determine the reduced model size. Model reduction was not found to significantly reduce the computational expense of simulating a dynamic system. However, large computational advantages may exist for reduced model sensitivity analysis as explained in Section 7.2.2.

7.2.2 Model Reduction for Sensitivity Analysis

For initial states sensitivities, an additional n^2 variables are solved simultaneously with the original n variables. By reducing the model order to r states, the number of sensitivity variables is reduced to r^2. For parameter sensitivities, an additional $n\mathrm{x}p$ variables are solved simultaneously with the original variables. A model order of r states reduces the number of sensitivity variables to rxp. Efficient sensitivity calculations are important for the single and multiple shooting solution techniques for nonlinear MPC, MHE, and model parameter estimation.

7.2.3 Model Reduction with the Open Equation Format

Chapter 3 shows techniques for reduction of models in the open equation format ($0 = f(\dot{x}, x, u)$). The simultaneous optimization approach requires a discretization scheme such as orthogonal collocation on finite elements to convert the differential equations into the NLP form. A numerical example of model reduction using the simultaneous approach was never developed. Instead, the sequential solution approach was used because of programming simplicity. However, in order to use the sequential approach, the model must be converted to the semi-explicit form ($\dot{x} = f(x, u)$). This conversion was possible for all models that are shown in this dissertation, however, it may be necessary to work with the open equation form directly. An interesting study would be to compare the computational and theoretical properties of the simultaneous and sequential solution approaches for reduced models in simulation and control.

7.2.4 Higher Order Piecewise Approximations in ISAT

ISAT approximates nonlinear functions by building multi-dimensional piecewise linear local approximations. The storage and retrieval performance could be increased by developing higher order local approximations. For some applications, it is desirable for the function approximation to be continuous or continuously differentiable. A higher order approximator may increase the regions that meet these constraints. Another method to approximate continuously differentiable functions may be to retrieve multiple nearby linear records to generate higher order approximations.

7.2.5 Parameter Estimation

Chapter 6 demonstrates a very efficient implementation of MHE to estimate the states, parameters, and disturbance variables from an advancing horizon of measurements. Periodically, it may be desirable to estimate new model parameters from a long history of normal operating data. In practice, this is usually accomplished by selecting a few points at steady state operation, setting the derivatives in the model to zero, and solving an optimization problem to minimize the difference between the model and data. The drawbacks to this approach stem from limiting the parameter estimation to steady state data. Optimal parameter estimation would include all historical data that are deemed valid. These data sets may include product grade transitions or be segmented by periods of shut-down or corrupted measurements. Explicit MHE was shown to scale quadratically with respect to model size and horizon length and may offer a superior alternative to optimization-based parameter updates.

Appendices

Appendix A

Binary Distillation Column Model

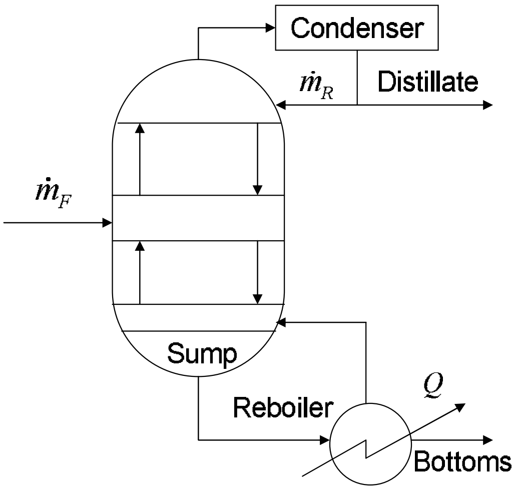

Figure A.1: Diagram of a dynamic binary distillation column model with equilibrium stages

Table A.1: Variables

Manipulated variables

Symbol	Description	Units
$\dot{m}_F$	Feed rate	$\frac{gm}{sec}$
$\dot{m}_R$	Reflux rate	$\frac{gm}{sec}$
$\dot{Q}$	Reboiler heating rate	$\frac{J}{sec}$

Variables at each of the 26 stages

Differential variables

Symbol	Description	Units
$\dot{x}_A$	Liquid mole fraction	none
$\dot{h}$	Specific enthalpy	$\frac{J}{mol}$

Algebraic variables

Symbol	Description	Units
y_A	Vapor mole fraction	none
x_L	Liquid mole fraction	none
T	Temperature	K
$\dot{n}_V$	Vapor molar flow rate	$\frac{mol}{sec}$
$\dot{n}_L$	Liquid molar flow rate	$\frac{mol}{sec}$
h_V	Specific vapor enthalpy	$\frac{J}{mol}$
h_L	Specific liquid enthalpy	$\frac{J}{mol}$
P_i^{sat}	Saturation pressure of compound j	Pa

Other variables

Symbol	Description	Units
n_L	Liquid molar holdup	mol
$MW_F(x_A)$	Molecular weight of feed stream	$\frac{gm}{mol}$
$MW_R(x_A)$	Molecular weight of reflux stream	$\frac{gm}{mol}$
P	Stage pressure	Pa
$h_j^V(T)$	Specific vapor enthalpy of compound j	$\frac{J}{mol}$
$h_j^L(T)$	Specific liquid enthalpy of compound j	$\frac{J}{mol}$

Table A.2: Equations

Differential Equations
Component A mole balance at each stage
$\dot{x}_A = \frac{1}{n_L}\left(y_{A_{in}}\dot{n}_{V_{in}} + x_{A_{in}}\dot{n}_{L_{in}} - y_{A_{out}}\dot{n}_{V_{out}} - x_{A_{out}}\dot{n}_{L_{out}} + \left(x_{A_{feed}}\frac{\dot{m}_{feed}}{MW_{feed}(x_A)}\right)\right)$
Energy balance at each stage
$\dot{h} = \frac{1}{n_L}\left(h_{V_{in}}\dot{n}_{V_{in}} + h_{L_{in}}\dot{n}_{L_{in}} - h_{V_{out}}\dot{n}_{V_{out}} - h_{V_{out}}\dot{n}_{L_{out}} + \left(h_{L_{feed}}\frac{\dot{m}_{feed}}{MW_{feed}(x_A)}\right) + \dot{Q}\right)$
Algebraic Equations
Raoult's law for VLE
$y_A = \frac{x_A P_A^{sat}}{P}$
Liquid mole fraction equation
$x_L = \frac{h-h_V}{h_L-h_V}$
Bubble point temperature equation
$P = x_A P_A^{sat} + (1-x_A)P_B^{sat}$
Vapor molar flow rate equation
$\dot{n}_{V_{out}} = \left(\dot{n}_{V_{in}} + \dot{n}_{L_{in}}\right)(1-x_L)$
Liquid molar flow rate equation
$\dot{n}_{L_{out}} = \left(\dot{n}_{V_{in}} + \dot{n}_{L_{in}}\right)x_L$
Vapor enthalpy equation
$h_V = y_A h_{VA}(T) + (1-y_A)h_{VB}(T)$
Liquid enthalpy equation
$h_L = x_A h_{LA}(T) + (1-x_A)h_{LB}(T)$
Pure component j saturated vapor pressure equation (DIPPR database)
$P_j^{sat} = \exp\left(A + \frac{B}{T} + C\ln(T) + DT^E\right)$

Appendix B

Dual CSTR Model

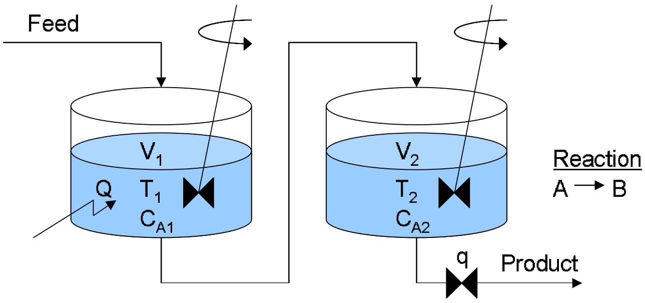

Figure B.1: This model is a dual CSTR with an exothermic first-order reaction. It is the same model as the one used by Hahn [34], but with some minor modifications.

Table B.1: Variables

Manipulated variables

Symbol	Description	Units
u	Valve position at the outlet of reactor #2	dimensionless
Q	Cooling heat flow from reactor #1	$\frac{J}{sec}$

State variables

Symbol	Description	Units
V_1	Volume of reactor #1	m^3
C_{A1}	Concentration of A in reactor #1	$\frac{mol}{m^3}$
T_1	Temperature of reactor #1	K
V_2	Volume of reactor #2	m^3
C_{A2}	Concentration of A in reactor #2	$\frac{mol}{m^3}$
T_2	Temperature of reactor #2	K

Other parameters

Symbol	Description	Units
C_{AF}	Concentration of A in the feed	$\frac{mol}{m^3}$
T_F	Feed temperature	K
q_F	Feed flow rate	$\frac{mol}{sec}$
q_1	Flow rate out of reactor #1	$\frac{mol}{sec}$
q_2	Flow rate out of reactor #2	$\frac{mol}{sec}$
k_0	Pre-exponential factor	$\frac{mol}{m^3-sec}$
E	Activation energy	$\frac{J}{mol}$
R	Universal gas constant (8.31451)	$\frac{J}{mol-K}$
ρ	Density of the liquid	$\frac{kg}{mol}$
c_p	Heat capacity of the liquid	$\frac{J}{kg-K}$
ΔH	Energy of reaction	$\frac{J}{mol}$
c	Constant relating valve position to flow rate	$\frac{mol}{sec-m^{3/2}}$

Table B.2: Equations

Equation(s)
Flow rates
$q_1 = c\sqrt{V_1 - V_2}$
$q_2 = c\sqrt{V_1}u$
Volume balances
$\frac{dV_1}{dt} = q_F - q_1$
$\frac{dV_2}{dt} = q_1 - q_2$
Component balances
$\frac{d(V_1 C_{A1})}{dt} = q_F C_{AF} - q_1 C_{A1} - k_0 C_{A1} V_1 \exp\left(-\frac{E}{RT_1}\right)$
$\frac{d(V_2 C_{A2})}{dt} = q_1 C_{A1} - q_2 C_{A2} - k_0 C_{A2} V_2 \exp\left(-\frac{E}{RT_2}\right)$
Energy balances
$\frac{d(V_1 T_1)}{dt} = q_F T_F - q_1 T_1 + \frac{\Delta H}{\rho c_p}\left(k_0 C_{A1} V_1 \exp\left(-\frac{E}{RT_1}\right)\right) - \frac{Q}{\rho c_p}$
$\frac{d(V_2 T_2)}{dt} = q_1 T_1 - q_2 T_2 + \frac{\Delta H}{\rho c_p}\left(k_0 C_{A2} V_2 \exp\left(-\frac{E}{RT_2}\right)\right)$

Appendix C

Flowsheet Model

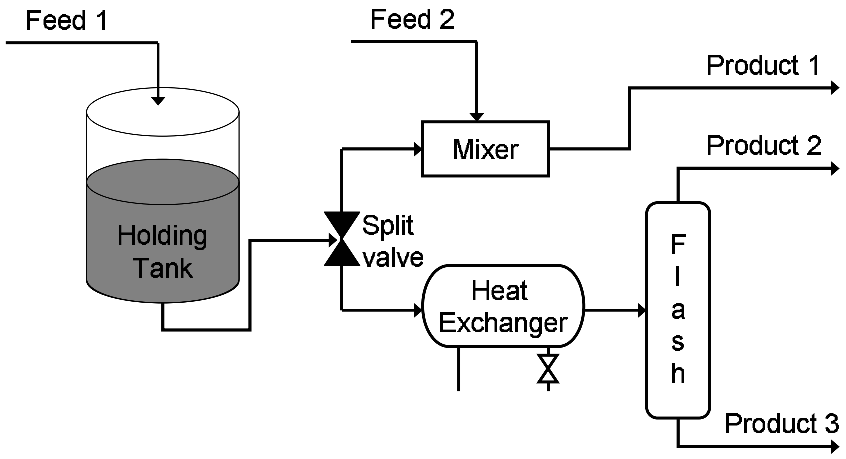

Figure C.1: Diagram of the flowsheet model involving equal molar feed streams of butane, pentane, hexane, heptane, and octane at 300 K and 1 ATM. The model has 229 variables with 12 ODEs and 217 AEs.

Tables C-1 - C-5 are a listing of all of the variables in the flowsheet model. The tables are divided by the model units. Table C-6 shows the irreduceable portion from the partitioning and precedence ordering of the variables and equations. This is the only set involving more than one variable and one equation and it consists of 16 variables and 16 equations from the flash column.

Table C.1: Feed Variables

Name	Name	Units
feed1.c(C4H10)	feed2.c(C4H10)	$\frac{kmol}{m^3}$
feed1.c(C5H12)	feed2.c(C5H12)	$\frac{kmol}{m^3}$
feed1.c(C6H14)	feed2.c(C6H14)	$\frac{kmol}{m^3}$
feed1.c(C7H16)	feed2.c(C7H16)	$\frac{kmol}{m^3}$
feed1.c(C8H18)	feed2.c(C8H18)	$\frac{kmol}{m^3}$
feed1.dens	feed2.dens	$\frac{kmol}{m^3}$
feed1.h	feed2.h	$\frac{J}{kmol}$
feed1.ndot	feed2.ndot	$\frac{kmol}{sec}$
feed1.vdot	feed2.vdot	$\frac{m^3}{sec}$
feed1.x(C8H18)	feed2.x(C8H18)	$none$
feed1.y(C4H10)	feed2.y(C4H10)	$none$
feed1.y(C5H12)	feed2.y(C5H12)	$none$
feed1.y(C6H14)	feed2.y(C6H14)	$none$
feed1.y(C7H16)	feed2.y(C7H16)	$none$
feed1.y(C8H18)	feed2.y(C8H18)	$none$

Table C.2: Flash Column Variables

Name		Units
flash.outlet_liq.c(C4H10)	flash.outlet_vap.c(C4H10)	$\frac{kmol}{m^3}$
flash.outlet_liq.c(C5H12)	flash.outlet_vap.c(C5H12)	$\frac{kmol}{m^3}$
flash.outlet_liq.c(C6H14)	flash.outlet_vap.c(C6H14)	$\frac{kmol}{m^3}$
flash.outlet_liq.c(C7H16)	flash.outlet_vap.c(C7H16)	$\frac{kmol}{m^3}$
flash.outlet_liq.c(C8H18)	flash.outlet_vap.c(C8H18)	$\frac{kmol}{m^3}$
flash.outlet_liq.dens	flash.outlet_vap.dens	$\frac{kmol}{m^3}$
flash.outlet_liq.h	flash.outlet_vap.h	$\frac{J}{kmol}$
flash.outlet_liq.mdot	flash.outlet_vap.mdot	$\frac{kg}{sec}$
flash.outlet_liq.ndot	flash.outlet_vap.ndot	$\frac{kmol}{sec}$
flash.outlet_liq.p	flash.outlet_vap.p	Pa
flash.outlet_liq.t	flash.outlet_vap.t	K
flash.outlet_liq.vdot	flash.outlet_vap.vdot	$\frac{m^3}{sec}$
flash.outlet_liq.x(C4H10)	flash.outlet_vap.x(C4H10)	$none$
flash.outlet_liq.x(C5H12)	flash.outlet_vap.x(C5H12)	$none$
flash.outlet_liq.x(C6H14)	flash.outlet_vap.x(C6H14)	$none$
flash.outlet_liq.x(C7H16)	flash.outlet_vap.x(C7H16)	$none$
flash.outlet_liq.x(C8H18)	flash.outlet_vap.x(C8H18)	$none$
flash.outlet_liq.y(C4H10)	flash.outlet_vap.y(C4H10)	$none$
flash.outlet_liq.y(C5H12)	flash.outlet_vap.y(C5H12)	$none$
flash.outlet_liq.y(C6H14)	flash.outlet_vap.y(C6H14)	$none$
flash.outlet_liq.y(C7H16)	flash.outlet_vap.y(C7H16)	$none$
flash.outlet_liq.y(C8H18)	flash.outlet_vap.y(C8H18)	$none$

Table C.3: Heat Exchanger Variables

Heat Exchanger Outlet	Units	Name	Units
hxc.outlet.c(C4H10)	$\frac{kmol}{m^3}$	hxc.reserve.c(C4H10)	$\frac{kmol}{m^3}$
hxc.outlet.c(C5H12)	$\frac{kmol}{m^3}$	hxc.reserve.c(C5H12)	$\frac{kmol}{m^3}$
hxc.outlet.c(C6H14)	$\frac{kmol}{m^3}$	hxc.reserve.c(C6H14)	$\frac{kmol}{m^3}$
hxc.outlet.c(C7H16)	$\frac{kmol}{m^3}$	hxc.reserve.c(C7H16)	$\frac{kmol}{m^3}$
hxc.outlet.c(C8H18)	$\frac{kmol}{m^3}$	hxc.reserve.c(C8H18)	$\frac{kmol}{m^3}$
hxc.outlet.dens	$\frac{kmol}{m^3}$	hxc.reserve.dens	$\frac{kmol}{m^3}$
hxc.outlet.h	$\frac{J}{kmol}$	hxc.reserve.h	$\frac{J}{kmol}$
hxc.outlet.mdot	$\frac{kg}{sec}$	hxc.reserve.m	kg
hxc.outlet.ndot	$\frac{kmol}{sec}$	hxc.reserve.n	$kmol$
hxc.outlet.p	Pa	hxc.reserve.p	Pa
hxc.outlet.t	K	hxc.reserve.t	K
hxc.outlet.vdot	$\frac{m^3}{sec}$	hxc.reserve.v	m^3
hxc.outlet.x(C4H10)	*none*	hxc.reserve.x(C4H10)	*none*
hxc.outlet.x(C5H12)	*none*	hxc.reserve.x(C5H12)	*none*
hxc.outlet.x(C6H14)	*none*	hxc.reserve.x(C6H14)	*none*
hxc.outlet.x(C7H16)	*none*	hxc.reserve.x(C7H16)	*none*
hxc.outlet.x(C8H18)	*none*	hxc.reserve.x(C8H18)	*none*
hxc.outlet.y(C4H10)	*none*	hxc.reserve.y(C4H10)	*none*
hxc.outlet.y(C5H12)	*none*	hxc.reserve.y(C5H12)	*none*
hxc.outlet.y(C6H14)	*none*	hxc.reserve.y(C6H14)	*none*
hxc.outlet.y(C7H16)	*none*	hxc.reserve.y(C7H16)	*none*
hxc.outlet.y(C8H18)	*none*	hxc.reserve.y(C8H18)	*none*

Table C.4: Splitter Variables

Split Outlet 1	Split Outlet 2	Units
split.frac2		$none$
split.outlet1.c(C4H10)	split.outlet2.c(C4H10)	$\frac{kmol}{m^3}$
split.outlet1.c(C5H12)	split.outlet2.c(C5H12)	$\frac{kmol}{m^3}$
split.outlet1.c(C6H14)	split.outlet2.c(C6H14)	$\frac{kmol}{m^3}$
split.outlet1.c(C7H16)	split.outlet2.c(C7H16)	$\frac{kmol}{m^3}$
split.outlet1.c(C8H18)	split.outlet2.c(C8H18)	$\frac{kmol}{m^3}$
split.outlet1.dens	split.outlet2.dens	$\frac{kmol}{m^3}$
split.outlet1.h	split.outlet2.h	$\frac{J}{kmol}$
split.outlet1.mdot	split.outlet2.mdot	$\frac{kg}{sec}$
split.outlet1.ndot	split.outlet2.ndot	$\frac{kmol}{sec}$
split.outlet1.p	split.outlet2.p	Pa
split.outlet1.t	split.outlet2.t	K
split.outlet1.vdot	split.outlet2.vdot	$\frac{m^3}{sec}$
split.outlet1.x(C4H10)	split.outlet2.x(C4H10)	$none$
split.outlet1.x(C5H12)	split.outlet2.x(C5H12)	$none$
split.outlet1.x(C6H14)	split.outlet2.x(C6H14)	$none$
split.outlet1.x(C7H16)	split.outlet2.x(C7H16)	$none$
split.outlet1.x(C8H18)	split.outlet2.x(C8H18)	$none$
split.outlet1.y(C4H10)	split.outlet2.y(C4H10)	$none$
split.outlet1.y(C5H12)	split.outlet2.y(C5H12)	$none$
split.outlet1.y(C6H14)	split.outlet2.y(C6H14)	$none$
split.outlet1.y(C7H16)	split.outlet2.y(C7H16)	$none$
split.outlet1.y(C8H18)	split.outlet2.y(C8H18)	$none$

Table C.5: Mixer and Tank Variables

Mixer	Tank Outlet	Tank Holdup	Units
mixer1.outlet.c(C4H10)	tank.outlet.c(C4H10)	tank.reserve.c(C4H10)	$\frac{kmol}{m^3}$
mixer1.outlet.c(C5H12)	tank.outlet.c(C5H12)	tank.reserve.c(C5H12)	$\frac{kmol}{m^3}$
mixer1.outlet.c(C6H14)	tank.outlet.c(C6H14)	tank.reserve.c(C6H14)	$\frac{kmol}{m^3}$
mixer1.outlet.c(C7H16)	tank.outlet.c(C7H16)	tank.reserve.c(C7H16)	$\frac{kmol}{m^3}$
mixer1.outlet.c(C8H18)	tank.outlet.c(C8H18)	tank.reserve.c(C8H18)	$\frac{kmol}{m^3}$
mixer1.outlet.dens	tank.outlet.dens	tank.reserve.dens	$\frac{kmol}{m^3}$
mixer1.outlet.h	tank.outlet.h	tank.reserve.h	$\frac{J}{kmol}$
mixer1.outlet.mdot	tank.outlet.mdot	tank.reserve.m (kg)	$\frac{kg}{sec}$
mixer1.outlet.ndot	tank.outlet.ndot	tank.reserve.n $(kmol)$	$\frac{kmol}{sec}$
mixer1.outlet.p	tank.outlet.p	tank.reserve.p	Pa
mixer1.outlet.t	tank.outlet.t	tank.reserve.t	K
mixer1.outlet.vdot	tank.outlet.vdot	tank.reserve.v (m^3)	$\frac{m^3}{sec}$
mixer1.outlet.x(C4H10)	tank.outlet.x(C4H10)	tank.reserve.x(C4H10)	$none$
mixer1.outlet.x(C5H12)	tank.outlet.x(C5H12)	tank.reserve.x(C5H12)	$none$
mixer1.outlet.x(C6H14)	tank.outlet.x(C6H14)	tank.reserve.x(C6H14)	$none$
mixer1.outlet.x(C7H16)	tank.outlet.x(C7H16)	tank.reserve.x(C7H16)	$none$
mixer1.outlet.x(C8H18)	tank.outlet.x(C8H18)	tank.reserve.x(C8H18)	$none$
mixer1.outlet.y(C4H10)	tank.outlet.y(C4H10)	tank.reserve.y(C4H10)	$none$
mixer1.outlet.y(C5H12)	tank.outlet.y(C5H12)	tank.reserve.y(C5H12)	$none$
mixer1.outlet.y(C6H14)	tank.outlet.y(C6H14)	tank.reserve.y(C6H14)	$none$
mixer1.outlet.y(C7H16)	tank.outlet.y(C7H16)	tank.reserve.y(C7H16)	$none$
mixer1.outlet.y(C8H18)	tank.outlet.y(C8H18)	tank.reserve.y(C8H18)	$none$

Table C.6: Partitioning and Precedence Ordering Implicit Block

Variable	Eqn #
flash.outlet_vap.x(C8H18)	203
flash.outlet_vap.h	214
flash.outlet_liq.h	195
flash.outlet_liq.x(C8H18)	192
flash.outlet_vap.x(C7H16)	191
flash.outlet_liq.x(C7H16)	199
flash.outlet_vap.x(C6H14)	190
flash.outlet_liq.x(C6H14)	198
flash.outlet_liq.ndot	194
flash.outlet_vap.x(C4H10)	188
flash.outlet_vap.ndot	196
flash.outlet_vap.x(C5H12)	197
flash.outlet_liq.x(C5H12)	189
flash.outlet_liq.x(C4H10)	218
flash.outlet_liq.t	229
flash.outlet_vap.t	193

Bibliography

[1] J. Acevedo and E.N. Pistikopoulos. A hybrid parametric/stochastic programming approach for mixed-integer linear problems under uncertainty. *Ind. Eng. Chem. Res.*, 36:2262–2270, 1997.

[2] J. Acevedo and E.N. Pistikopoulos. A multiparametric programming approach for linear process enginering problems under uncertainty. *Ind. Eng. Chem. Res.*, 36:717–728, 1997.

[3] J.S. Albuquerque and L.T. Biegler. Decomposition algorithms for on-line estimation with nonlinear models. *Computers and Chemical Engineering*, 19(10):1031–1039, 1995.

[4] J.S. Albuquerque and L.T. Biegler. Decomposition algorithms for online estimation with nonlinear DAE models. *Computers and Chemical Engineering*, 21(3):283–299, 1997.

[5] I. Aleksander and H. Morton. *An Introduction to Neural Computing.* International Thomson Computer Press, London, 1995.

[6] A.G. Barto. *Neural Systems for Control.* Academic Press, San Diego, CA, 1997.

[7] V.M. Becerra, P.D. Roberts, and G.W. Griffiths. Applying the extended kalman filter to systems described by nonlinear differential-algebraic equations. *Control Engineering Practice*, 9:267–281, 2001.

[8] R. Bellman. *Applied Dynamic Programming.* Princeton University Press, Princeton, NJ, 1962.

[9] A. Bemporad. Multiparametric nonlinear integer programming and explicit quantized optimal control. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 3167–3172, Maui, Hawaii, December 2003.

[10] A. Bemporad, F. Borrelli, and M. Morari. Explicit solution of lp-based model predictive control. In *Proceedings of the 39th IEEE Conference on Decision and Control*, pages 632–637, Sydney, Australia, December 2000.

[11] A. Bemporad, F. Borrelli, and M. Morari. Optimal controllers for hybrid systems: stability and piecewise linear explicit form. In *Proceedings of the 39th IEEE Conference on Decision and Control*, pages 1810–1815, Sydney, Australia, December 2000.

[12] A. Bemporad, F. Borrelli, and M. Morari. Model predictive control based on linear programming - the explicit solution. *IEEE Transactions on Automatic Control*, 47(12):1974–1985, December 2002.

[13] A. Bemporad and C. Filippi. Suboptimal explicit MPC via approximate multiparametric quadratic programming. In *Proceedings of the*

40th IEEE Conference on Decision and Control, pages 4851–4856, Orlando, FL, December 2001.

[14] A. Bemporad, M. Morari, V. Dua, and E.N. Pistokopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38:3–20, 2002.

[15] A. Bemporad, F.D. Torrisi, and M. Morari. Performance analysis of piecewise linear systems and model predictive control systems. In *Proceedings of the 39th IEEE Conference on Decision and Control*, pages 4957–4962, Sydney, Australia, December 2000.

[16] D.P. Bertsekas. Neuro-dyanamic programming: An overview. In *Proc. CPC VI*, pages 92–96, 2001.

[17] T. Binder, L. Blank, H.G. Bock, R. Burlisch, W. Dahmen, M. Diehl, T. Kronseder, W. Marquardt, J.P. Schlöder, and O.v. Stryk. *Online Optimization of Large Scale Systems*, chapter Introduction to model based optimization of chemical processes on moving horizons, pages 295–339. Springer-Verlag Berlin Heidelberg, 2001.

[18] W. Borutsky and F.E. Cellier. Tearing in bond graphs with depedent storage elements. In *IMACS MultiConference on Computational Engineering in Systems Applications*, volume 1, pages 1113–1119, Lille, France, 1996. CESA.

[19] J.R. Bosley. *An Experimental Investigation of Modeling, Control, and Optimization Techniques for Batch Distillation.* PhD thesis, University of Texas at Austin, 1994.

[20] E. Carpanzano. Order reduction of general nonlinear dae systems by automatic tearing. *Mathematical and Computer Modelling of Dynamical Systems*, 6(2):145–168, 2000.

[21] J.-Y. Chen, J. A. Blasco, N. Fueyo, and C. Dopazo. An economical strategy for storage of chemical kinetics: fitting in situ adaptive tabulation with artificial neural networks. In *Proceedings of the Combustion Institute*, volume 28, pages 115–121, 2000.

[22] M. Diehl. *Real-time Optimization for Large Scale Nonlinear Processes.* PhD thesis, Heidelberg University, Heidelberg, Germany, 2001.

[23] M. Diehl, H. G. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, , and F. Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12:577–585, 2002.

[24] V. Dua and E.N. Pistikopoulos. Algorithms for the solution of multiparametric mixed-integer nonlinear optimization problems. *Ind. Eng. Chem. Res.*, 38:3976–3987, 1999.

[25] I.S. Duff. On algorithms for obtaining a maximum transversal. *ACM Transactions on Mathematical Software*, 7(3):315–330, 1981.

[26] I.S. Duff and J.K. Reid. An implementation of tarjan's algorithm for the block triangularization of a matrix. *ACM Transactions on Mathematical Software*, 4(2):137–147, 1978.

[27] A.V. Fiacco. *Introduction to sensitivity and stability analysis in nonlinear programming*. Academic Press, London, 1983.

[28] T. Gal. *Postoptimal Analysis*. de Gruyter, Berlin, 2nd edition, 1995.

[29] T. Gal and J. Nedoma. Multiparametric linear programming. *Management Science*, 18:406–442, 1975.

[30] N. Ganesh and L.T. Biegler. A robust technique for process flowsheet optimization using simplified model approximations. *Computers and Chemical Engineering*, 11:553–565, 1987.

[31] A. Grancharova and T.A. Johansen. Approximate explicit model predictive control incorporating heuristics. In *2002 IEEE International Symposium on Computer Aided Control System Design Proceedings*, pages 92–97, Glasgow, Scotland, September 2002.

[32] P. Grieder and M. Morari. Complexity reduction of receding horizon control. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 3179–3190, Maui, Hawaii, December 2003.

[33] J. Hahn, T. F. Edgar, and W. Marquardt. Controllability and observability covariance matricies for the analysis and order reduction of stable nonlinear systems. *Journal of Process Control*, 13:115–127, 2003.

[34] J. Hahn and T.F. Edgar. An improved method for nonlinear model reduction using balancing of empirical gramians. *Computers and Chemical Engineering*, 26:1379–1397, 2002.

[35] E.T. Hale. *Multi-parametric nonlinear programming*. PhD thesis, University of Texas at Austin, 2005.

[36] E.T. Hale and S.J. Qin. Multi-parametric nonlinear programming and the evaluation of implicit optimization model adequacy. In *Proceedings of the 7th International Symposium on the Dynamics and Control of Process Systems*, Cambridge, MA, July 2004.

[37] K.M. Hangos and I.T. Cameron. *Process Modelling and Model Analysis*, volume 4. Academic Press, 2001.

[38] E.L. Haseltine and J.B. Rawlings. Critical evaluation of extended kalman filtering and moving-horizon estimation. *Ind. Eng. Chem. Res.*, 2004.

[39] J.D. Hedengren and T.F. Edgar. In situ adaptive tabulation for real time control. In *Proceedings of the American Control Conference*, Boston, MA, 2004.

[40] J.D. Hedengren and T.F. Edgar. In situ adaptive tabulation for real time control. *I&EC Research*, 2005.

[41] S.S. Jang, B. Joseph, and H. Mukai. Comparison of two approaches to on-line parameter and state estimation of nonlinear systems. *Ind. Eng. Chem. Process Des. Dev.*, 25:809–814, 1986.

[42] T.A. Johansen. On multi-parametric nonlinear programming and explicit nonlinear model predictive control. In *Proceedings of the 41st IEEE Conference on Decision and Control*, pages 2768–2773, Las Vegas, NV, December 2002.

[43] T.A. Johansen. Approximate explicit receding horizon control of constrained nonlinear systems. *Automatica*, 40:293–300, 2004.

[44] T.A. Johansen and A. Grancharova. Approximate explicit constrained linear model predictive control via orthogonal search tree. *IEEE Transactions on Automatic Control*, 48(5):810–815, 2003.

[45] N.S. Kaisare, J.M. Lee, and J.H. Lee. Simulation based strategy for nonlinear optimal control: Application to a microbial cell reactor. *International Journal of Robust and Nonlinear Control*, 13:347–363, 2003.

[46] S.H. Lam and D.A. Goussis. Understanding complex chemical kinetics with computational singular perturbation. In *Twenty-Second Symposium (International) on Combustion*, pages 931–941, Pittsburgh, PA, 1988. The Combustion Institute.

[47] M.J. Liebman, T.F. Edgar, and L.S. Lasdon. Efficient data reconciliation and estimation for dynamic processes using nonlinear programming techniques. *Computers and Chemical Engineering*, 16:963–986, 1992.

[48] B. Lincoln and A. Rantzer. Suboptimal dyanamic programming with error bounds. In *Proceedings of the 41st conference on decision and*

control, Las Vegas, NV, December 2002.

[49] K. Liu, S.B. Pope, and D.A. Caughey. Calculations of bluff-body stabilized flames using a joint probability density function model with detailed chemistry. *Combustion and Flame*, 141:89–117, 2005.

[50] U. Maas and S.B. Pope. Implementation of simplified chemical kinetics based on intrinsic low-dimensional manifolds. In *Twenty-Fourth Symposium (International) on Combustion*, pages 103–112, Pittsburgh, PA, 1992. The Combustion Institute.

[51] U. Maas and S.B. Pope. Simplifying chemical kinetics: intrinsic low-dimensional manifolds in composition space. *Combustion and Flame*, 88:239–264, 1992.

[52] W. Marquardt. Nonlinear model reduction for optimization based control of transient chemical processes. In *Proc. CPC VI*, volume 98 of *AIChE Symposium Series*, pages 30–60, 2001.

[53] K.F. McBrayer and T.F. Edgar. Bias detection and estimation in dynamic data reconciliation. *Journal of Process Control*, 5(4):285–289, 1995.

[54] H. Michalska and D.Q. Mayne. Moving horizon observers and observer-based control. *IEEE Transactions on Automatic Control*, 40(6):995–1006, 1995.

[55] P.E. Moraal and J.W. Grizzle. Observer design for nonlinear systems with discrete-time measurements. *IEEE Transactions on Automatic Control*, 40(3):395–404, 1995.

[56] K. Mukati and B. Ogunnaike. Stability analysis and tuning strategies for a novel next generation regulatory controller. In *Proceedings of the 2004 American Control Conference*, pages 4034–4039, Boston, MA, June 2004.

[57] K. R. Muske and T. A. Badgwell. Disturbance modeling for offset-free linear model predictive control. *Journal of Process Control*, 12:617–632, 2002.

[58] M. Otter, H. Elmqvist, and F.E. Cellier. Relaxing: A symbolic sparse matrix method exploiting the model structure in generating efficient simulation code. In *IMACS MultiConference on Computational Engineering in Systems Applications*, volume 1, pages 1–12, 1996.

[59] G. Pannocchia. Robust disturbance modeling for model predictive control with application to multivariable ill-conditioned processes. *Journal of Process Control*, 13(8):693–701, 2003.

[60] G. Pannocchia, N. Laachi, and J.B. Rawlings. A fast, easily turned, siso, model predictive controller. In *DYCOPS*, page 163, Boston, MA, 2004.

[61] G. Pannocchia, N. Laachi, and J.B. Rawlings. A candidate to replace PID control: SISO constrained LQ control. *AIChE Journal*, 2005.

[62] G. Pannocchia and J.B. Rawlings. Disturbance models for offset-free MPC control. *AIChE Journal*, 49(2):426–437, 2002.

[63] T. Parisini and R. Zoppoli. A receding-horizon regulator for nonlinear systems and a neural approximation. *Automatica*, 31(10):1443–1451, 1995.

[64] E.N. Pistikopoulos, V. Dua, N.A. Bozinis, A. Bemporad, and M. Morari. On-line optimization via off-line parametric optimization tools. *Computers and Chemical Engineering*, 24:183–188, 2000.

[65] E.N. Pistikopoulos, V. Dua, N.A. Bozinis, A. Bemporad, and M. Morari. On-line optimization via off-line parametric optimization tools. *Computers and Chemical Engineering*, 26:175–185, 2002.

[66] S.B. Pope. Computationally efficient implementation of combustion chemistry using in situ adaptive tabulation. *Combustion Theory Modelling*, 1:41–63, 1997.

[67] S.J. Qin. *Neural Systems for Control*, chapter Neural networks for intelligent sensors and control: Practical issues and some solutions. Academic Press, San Diego, CA, 1997.

[68] S.J. Qin and T.A. Badgwell. *Nonlinear Model Predictive Control*, chapter An overview of nonlinear model predictive control applications, pages 369–392. Birkhäuser Verlag, Boston, MA, 2000.

[69] Y. Ramamurthi, P.B. Sistu, and B.W. Bequette. Control-relevant dynamic data reconciliation and parameter estimation. *Computers and Chemical Engineering*, 17(1):41–59, 1993.

[70] C.V. Rao, J.B. Rawlings, and J.H. Lee. Constrained linear state estimation - a moving horizon approach. *Automatica*, 37:1619–1628, 2001.

[71] D.G. Robertson and J.H. Lee. A least squares formulation for state estimation. *Journal of Process Control*, 5(4):291–299, 1995.

[72] J.A. Rossiter and P. Grieder. Using interpolation to improve efficiency of multiparametric predictive control. *Automatica*, 2005.

[73] V. Sakizlis, V. Dua, J.D. Perkins, and E.N. Pistikopoulos. Robust model-based tracking control using parametric programming. *Computers and Chemical Engineering*, 28:195–207, 2004.

[74] V. Saxena and S.B. Pope. Pdf calculations of major and minor species in a turbulent piloted jet flame. In *Symposium (International) on Combustion*, pages 1081–1086, 1998.

[75] V. Saxena and S.B. Pope. Pdf simulations of turbulent combustion incorporating detailed chemistry. *Combustion and Flame*, 117:340–350, 1999.

[76] D.E. Seborg, T.F. Edgar, and D.A. Mellichamp. *Process Dynamics and Control*. Wiley, New York, 2nd edition, 2004.

[77] M.M. Seron, J.A. De Doná, and Graham C. Goodwin. Global analytical model predictive control with input constraints. In *Proceedings of the 39th IEEE Conference on Decision and Control*, pages 154–159, Sydney, Australia, December 2000.

[78] J.J. Shah and R.O. Fox. Computational fluid dynamics simulation of chemical reactors: application of in situ adaptive tabulation to methane thermochlorination chemistry. *I&EC Research*, 11:4200–4212, 1999.

[79] T.A. Soderstrom, T.F. Edgar, L.P. Russo, and R.E. Young. Industrial application of a large-scale dynamic data reconciliation strategy. *Industrial and Engineering Chemistry Research*, 39:1683–1693, 2000.

[80] M. Soroush. State and parameter estimations and their applications in process control. *Computers and Chemical Engineering*, 23:229–245, 1998.

[81] Q. Tang and S.B. Pope. Implementation of combustion chemistry by in situ adaptive tabulation of rate-controlled constrained equilibrium manifolds. In *Proceedings of the Combustion Institute*, volume 1, pages 1411–1417, 2002.

[82] R. Tarjan. Depth first search and linear graph algorithms. *SIAM Journal on Computing*, 1:146–160, 1972.

[83] M.J. Tenny, S.J. Wright, and J.B. Rawlings. Nonlinear model predictive control via feasibility–perturbed sequential quadratic programming. In

Texas-Wisconsin Modeling and Control Consortium, number TWMCC-2002-02, 2002.

[84] W.J. Terrell. Local observability of nonlinear differential-algebraic equations (daes) from the linearization along a trajectory. *IEEE Transactions on Automatic Control*, 46(12):1947–1950, December 2001.

[85] A.S. Tomlin, T. Turanyi, and M.J. Pilling. *Low-Temperature Combustion and Autoignition*, volume 35 of *Comprehensive Chemical Kinetics*, chapter Mathematical tools for the construction, investigation and reduction of combustion mechanisms, pages 293–437. Low-Temperature Combustion and Autoignition, 1997.

[86] P. Tondel, T.A. Johansen, and A. Bemporad. An algorithm for multiparametric quadratic programming and explicit mpc solutions. *Automatica*, 39:489–497, 2003.

[87] P. Tondel, T.A. Johansen, and A. Bemporad. Further results on multiparametric quadratic programming. In *Proceedings of the 42nd IEEE Conference on Decision and Control*, pages 3173–3178, Maui, Hawaii, December 2003.

[88] P. Vachhani, R. Rengaswamy, V. Gangwal, and S. Narasimhan. Recursive estimation in constrained nonlinear dynamical systems. *AIChE Journal*, 51(3):946–959, 2005.

[89] N. Vora and P. Daoutidis. Nonlinear model reduction of chemical reaction systems. *AIChE Journal*, 47:2320–2332, 2001.

[90] K. Warwick. *Neural Network Application and Control*, chapter Neural Networks: An Introduction. IEE, 1995.

[91] D. Wolbert, X. Joulia, B. Koehret, and L.T. Biegler. Flowsheet optimization and optimal sensitivity analysis using analytical derivatives. *Computers and Chemical Engineering*, 18(11/12):1083–1095, 1994.

[92] J. Xu and S.B. Pope. Pdf calculations of turbulent nonpremixed flames with local extinction. *Combustion and Flame*, 123(3):281–307, 2000.

[93] B. Yang and S.B. Pope. Treating chemistry in combustion with detailed mechanisms: In situ adaptive tabulation in principal directions. *Combustion and Flame*, 112:85–112, 1998.

[94] Y. Zhang, M.A. Henson, and Y.G. Kevrekidis. Nonlinear order reduction of discretized cell population models. In *Proceedings of the ACC*, pages 2383–2388, Denver, CO, 2003.

Vita

John David Hedengren was born in Milan, Italy on 3 July 1977, the son of David C. Hedengren and Marian Hedengren. He received the Bachelor of Science and Master of Science degree in Engineering from Brigham Young University. He applied to the University of Texas at Austin for enrollment in the Chemical Engineering program. He was accepted and started graduate studies in September, 2002.

Permanent address: 9604 Blue Creek Lane
Austin, Texas 78758

This dissertation was typeset with LaTeX[†] by 'the author'.

[†] LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.