

Hybrid Dynamic Modeling and Identification

by

Daniel Martin Webb

B.S., University of Arkansas at Fayetteville, 1997

A thesis submitted to the
Faculty of the Graduate School of the
University of Colorado in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
Department of Chemical and Biological Engineering
2007

This thesis entitled:
Hybrid Dynamic Modeling and Identification
written by Daniel Martin Webb
has been approved for the Department of Chemical and Biological Engineering

W. Fred Ramirez

Dhinakar S. Kompala

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Webb, Daniel Martin (Ph.D., Chemical and Biological Engineering)

Hybrid Dynamic Modeling and Identification

Thesis directed by Prof. W. Fred Ramirez

A new technique for identifying hybrid mechanistic/empirical dynamic models was demonstrated, as well as several new techniques for improving the speed and convergence properties of the iterative dynamic programming optimal control algorithm.

A hybrid dynamic model is a mechanistic model developed using an understanding of the underlying system combined with an empirical model, typically a neural network. Models of this type have been used for several years in bioreactor modeling research because the conservation equations and reaction stoichiometry are usually known but the reaction kinetics usually are not. The neural network part of these hybrid dynamic models is challenging to identify because there is not a direct link between the neural network weights and the error (measured vs. predicted states).

A new more general technique for identifying these systems was introduced where the neural network outputs are treated as controls in an optimal control problem, and the optimal control objective function minimizes the error between measured and predicted states. Once the neural network outputs are found using optimal control, the neural networks can be trained as usual. This method identified a model where one state had only five data per run without using a state estimator.

Iterative dynamic programming (IDP) was the optimal control algorithm chosen for this technique because of its lack of model restrictions, but it suffers from two serious problems for this application: computational expense and noisy control trajectories. A new two-step method was proposed where basic IDP is used to solve the problem to some tolerance, then a modified IDP is used to solve the problem to a higher tolerance. The modified IDP used in the second step includes control filtering and/or damping

methods which quickly reduce control noise. Several new methods were developed for control smoothing as well as an improved algorithm for stagewise linear continuous controls and a new algorithm for stagewise linear discontinuous controls. These methods decreased computation time several orders of magnitude for the Park-Ramirez bioreactor optimal control problem with 50 control stages.

Dedication

To my granny Ruth: lover of books and words, my favorite storyteller, and a woman who never spoke to me about her beliefs, but demonstrated them through action and example.

Acknowledgements

“Dan, finish your dissertation first. Love, Deb.”

“Daniel, get back to work. Your father.”

Thank you to Fred Ramirez, who has been a great source of ideas and direction. Enjoy your retirement, you have earned it!

Thank you to my wife Janet, who gave me the best gift I ever received when she gave herself to me in marriage. She has been ever-patient with me as this dissertation took more and more of my time.

Thank you to my uncle Jerry, who gave me the second best gift I ever received when he gave me a TI 99/4A computer when I was eight years old.

Thank you to my parents, who always supported and encouraged my education.

Thank you to all the people at school, especially to Dongmei and Chris.

Contents

Chapter		
1	Introduction	1
2	The Park-Ramirez bioreactor model	6
3	Iterative dynamic programming	12
3.1	The optimal control problem	12
3.2	Overview of solutions to the optimal control problem	13
3.3	Introduction to IDP	13
3.4	Basic IDP algorithm	14
3.4.1	State and control grids	15
3.4.2	Finding the best control for a state grid at each stage	16
3.4.3	Iteration optimal	16
3.4.4	Continuing iterations	17
3.5	IDP basic algorithm - subalgorithms	17
3.6	Stagewise constant IDP	17
3.7	A two-step method for efficient IDP	20
4	Iterative dynamic programming efficiency	21
4.1	Choosing IDP parameters	21
4.1.1	Number of passes and iterations	22

4.1.2	Number of state grids	22
4.1.3	Number of test controls	22
4.1.4	Number of stages	23
4.1.5	Number of controls	23
4.2	Parallel implementation of IDP	24
4.3	Control precision and error	25
4.4	Integrator tolerance	27
4.5	Adaptive control region size	32
4.5.1	Adaptive control region size methods	32
4.5.2	Adaptive control region parameters	34
4.5.3	Stopping criteria	39
4.6	Performance index control sensitivity	39
5	Iterative dynamic programming control smoothing	43
5.1	The active control problem	43
5.1.1	Examination of the active control problem	44
5.1.2	Park-Ramirez optimal control problem using basic IDP	46
5.1.3	Control filtering	49
5.1.4	Simulated annealing control filtering	55
5.1.5	Control damping	61
5.1.6	Pivot point test controls	68
5.1.7	Solo test controls	74
5.2	Stagewise linear continuous IDP	74
5.3	Stagewise linear discontinuous IDP	78
5.4	Variable stage lengths	82
5.5	Adaptive control region parameters for the two-step procedure	87
5.6	Two-step conclusions	91

5.7	Comparison with previously reported results	91
5.7.1	IDP Implementation	91
5.7.2	Two-step method	93
6	Hybrid dynamic model identification	94
6.1	Identification methods	94
6.1.1	Derivative method	95
6.1.2	Limitations of the derivative method	96
6.1.3	Integral method	96
6.1.4	Limitations of the integral method	98
6.2	Optimal control for identification (Webb-Ramirez method)	99
6.3	Ensuring feasible states for dynamic models	101
6.4	Identifying the hybrid Park-Ramirez model	103
6.5	Artificial data	103
6.6	Identifying the hybrid Park-Ramirez model parameter functions $v(t)$	106
6.6.1	Effect of IDP parameters on $v(t)$ identification	106
6.6.2	$v(t)$ identification using basic IDP	106
6.6.3	$v(t)$ identification using basic IDP with smoothing	111
6.6.4	$v(t)$ identification using two-step IDP	115
6.6.5	$v(t)$ identification using two-step IDP and P_T data	120
6.7	Identifying $v(\mathbf{X}, \omega)$ using $v(t)$ data	124
6.7.1	Parameter function sensitivity for improved identification	124
6.7.2	Extracting $v(\mathbf{X}, \omega)$ training data from $v(t)$ data	124
6.7.3	Neural network domain transformation	125
6.7.4	Neural network training	126
6.8	Identifying $v(\mathbf{X}, \omega)$ for the Park-Ramirez model	126
6.8.1	Identifying Park-Ramirez $v(\mathbf{X}, \omega)$: 0 P_T data, no noise	129

6.8.2	Identifying Park-Ramirez $v(\mathbf{X}, \omega)$: 0 P_T data, with noise	134
6.8.3	Identifying Park-Ramirez $v(\mathbf{X}, \omega)$: 5 P_T data, no noise	136
6.8.4	Identifying Park-Ramirez $v(\mathbf{X}, \omega)$: 5 P_T data, with noise	141
6.8.5	Identifying Park-Ramirez $v(\mathbf{X}, \omega)$: 5 P_T data, with noise, no sensitivity	143
6.8.6	Identifying Park-Ramirez $v(\mathbf{X}, \omega)$: 100 P_T data, no noise . . .	145
6.8.7	Identifying Park-Ramirez $v(\mathbf{X}, \omega)$: 100 P_T data, with noise . .	150
6.8.8	S discontinuity for improved $v(\mathbf{X}, \omega)$ identification	155
6.8.9	Optimal control of the identified hybrid model	158
6.9	Discussion	159
6.9.1	Identification methods	159
6.9.2	State data noise	162
7	The Flux-balance bioreactor model	163
7.1	Introduction	163
7.2	Model Development	164
7.2.1	Overall model structure and state equations	164
7.2.2	Kinetic Model	165
7.2.3	Uptake Rate Capacities	167
7.2.4	Uptake Rates	168
7.2.5	Maintenance	168
7.2.6	Cell Growth	169
7.2.7	Foreign Protein Production	170
7.2.8	Acetate Consumption	171
7.2.9	Carbon Dioxide generation rate	172
7.2.10	Cell Death	172
7.2.11	Oxygen Mass Transport	173

7.2.12	Flux-balance Model Yields	173
7.3	Results and Discussion	176
7.4	Conclusions	179
8	Conclusions and Recommendations	185
8.1	Recommendations for IDP optimal control	185
8.1.1	General recommendations for all IDP problems	185
8.1.2	Recommendations for two-step IDP	186
8.2	IDP smoothing techniques	187
8.3	Recommendations for Webb-Ramirez identification	188
8.4	Conclusions	188
9	Possible topics for future research	190
9.1	IDP for high-speed real-time control	190
9.2	Dealing with very noisy state data	191
9.2.1	State damping for noise reduction	194
9.3	Approaching test control and state boundaries	195
	Bibliography	197
	Appendix	
A	Software	202
A.1	Philosophy	202
A.2	Programming language and compiler tools	203
A.3	IDP library	204
A.4	Model framework library	204
A.5	Dissertation software tools	205

B Complete $v(t)$ identification results for one case

Tables

Table

2.1	Park-Ramirez model parameters	8
4.1	IDP iterations required to reach 99% or 99.9% of optimal Φ for 25 stages	36
4.2	IDP iterations required to reach 99% or 99.9% of optimal Φ for 50 stages	37
4.3	Default step one parameters for Park-Ramirez optimal control problem .	38
5.1	Basic IDP computational cost for Park-Ramirez optimal control problem	49
5.2	Step one parameters for stagewise linear discontinuous Park-Ramirez problem	80
5.3	Adaptive region size history iterations and k_r effect on SSE_n (25 stages)	89
5.4	Adaptive region size history iterations and k_r effect on SSE_n (50 stages)	90
5.5	Basic IDP vs. two-step computational cost for Park-Ramirez optimal control problem	93
6.1	Parameters for Park-Ramirez identification problem	107
6.2	IDP smoothing parameters for Park-Ramirez identification problem . .	111
6.3	Neural network parameters for Park-Ramirez model $v(\mathbf{X}, \omega)$ identifica- tion	126
7.1	Symbols for flux-balance model differential equations	166
7.2	Flux-balance model yields	174

7.3 Flux-balance model parameters 175

Figures

Figure

2.1	Kinetic parameter μ for the Park-Ramirez bioreactor model.	8
2.2	Kinetic parameter f_p for the Park-Ramirez bioreactor model.	8
2.3	Kinetic parameter ϕ for the Park-Ramirez bioreactor model.	9
2.4	Optimal control of the Park-Ramirez bioreactor model.	10
4.1	Effect of Gaussian control error on performance index error	26
4.2	Effect of control offset on performance index error	26
4.3	Effect of integrator relative tolerance on performance index	28
4.4	Computation time vs. integrator relative tolerance	28
4.5	Effect of integrator relative tolerance on optimal control solution	30
4.6	Computation time vs. integrator absolute tolerance	31
4.7	Effect of stopping parameter γ_{stop}	40
4.8	Performance index sensitivity ($\xi_{\Phi u}$)	42
5.1	Intermediate IDP solution of a simple bioreactor problem	45
5.2	Basic IDP solution of Park-Ramirez problem (low iterations)	47
5.3	Basic IDP solution of Park-Ramirez problem (high iterations)	48
5.4	IDP: First-order control filtering (10 stages)	52
5.5	IDP: First-order control filtering (25 stages)	53
5.6	IDP: First-order control filtering (50 stages)	54

5.7	IDP: Simulated annealing control filtering (10 stages)	58
5.8	IDP: Simulated annealing control filtering (25 stages)	59
5.9	IDP: Simulated annealing control filtering (50 stages)	60
5.10	IDP: Relative control damping (10 stages)	64
5.11	IDP: Relative control damping (25 stages)	65
5.12	IDP: Relative control damping (50 stages)	66
5.13	Effect of minimal control damping	67
5.14	IDP: Pivot point test controls (10 stages)	71
5.15	IDP: Pivot point test controls (25 stages)	72
5.16	IDP: Pivot point test controls (50 stages)	73
5.17	IDP: Stagewise linear continuous controls	77
5.18	IDP: Stagewise linear discontinuous controls	81
5.19	IDP: Variable stage lengths sequentially after controls (10 stages)	85
5.20	IDP: Variable stage lengths sequentially after controls (25 stages)	86
6.1	Controls used to generate artificial state data	105
6.2	$v(t)$ identification: basic IDP (10 stages, 0 P_T data, with noise)	108
6.3	$v(t)$ identification: basic IDP (25 stages, 0 P_T data, with noise)	109
6.4	$v(t)$ identification: basic IDP (50 stages, 0 P_T data, with noise)	110
6.5	$v(t)$ identification: smoothed basic IDP (10 stages, 0 P_T data, with noise)	112
6.6	$v(t)$ identification: smoothed basic IDP (25 stages, 0 P_T data, with noise)	113
6.7	$v(t)$ identification: smoothed basic IDP (50 stages, 0 P_T data, with noise)	114
6.8	$v(t)$ identification: two-step IDP (10 stages, 0 P_T data, with noise) . . .	117
6.9	$v(t)$ identification: two-step IDP (25 stages, 0 P_T data, with noise) . . .	118
6.10	$v(t)$ identification: two-step IDP (50 stages, 0 P_T data, with noise) . . .	119
6.11	$v(t)$ identification: two-step IDP (10 stages, 5 P_T data, with noise) . . .	121
6.12	$v(t)$ identification: two-step IDP (25 stages, 5 P_T data, with noise) . . .	122

6.13	$v(t)$ identification: two-step IDP (50 stages, 5 P_T data, with noise)	123
6.14	$v(t)$ identification: two-step IDP (10 stages, 0 P_T data, no noise)	130
6.15	$v(t)$ identification: two-step IDP (25 stages, 0 P_T data, no noise)	131
6.16	$v(t)$ identification: two-step IDP (50 stages, 0 P_T data, no noise)	132
6.17	$v(\mathbf{X}, \omega)$ identification (0 P_T data, no noise)	133
6.18	$v(\mathbf{X}, \omega)$ identification (0 P_T data, with noise)	135
6.19	$v(t)$ identification: two-step IDP (10 stages, 5 P_T data, no noise)	137
6.20	$v(t)$ identification: two-step IDP (25 stages, 5 P_T data, no noise)	138
6.21	$v(t)$ identification: two-step IDP (50 stages, 5 P_T data, no noise)	139
6.22	$v(\mathbf{X}, \omega)$ identification (5 P_T data, no noise)	140
6.23	$v(\mathbf{X}, \omega)$ identification (5 P_T data, with noise)	142
6.24	$v(\mathbf{X}, \omega)$ identification (5 P_T data, with noise, no sensitivity)	144
6.25	$v(t)$ identification: two-step IDP (10 stages, 100 P_T data, no noise)	146
6.26	$v(t)$ identification: two-step IDP (25 stages, 100 P_T data, no noise)	147
6.27	$v(t)$ identification: two-step IDP (50 stages, 100 P_T data, no noise)	148
6.28	$v(\mathbf{X}, \omega)$ identification (100 P_T data, no noise)	149
6.29	$v(t)$ identification: two-step IDP (10 stages, 100 P_T data, with noise)	151
6.30	$v(t)$ identification: two-step IDP (25 stages, 100 P_T data, with noise)	152
6.31	$v(t)$ identification: two-step IDP (50 stages, 100 P_T data, with noise)	153
6.32	$v(\mathbf{X}, \omega)$ identification (100 P_T data, with noise)	154
6.33	$v(\mathbf{X}, \omega)$ identification (no noise, substrate discontinuity)	156
6.34	$v(\mathbf{X}, \omega)$ identification (with noise, substrate discontinuity)	157
6.35	Optimal control of identified hybrid Park-Ramirez model (no noise)	160
6.36	Optimal control of identified hybrid Park-Ramirez model (with noise)	161
7.1	Flux-balance kinetic model overview	180
7.2	Flux-balance kinetic model metabolism	181

7.3	Training results for flux-balance model using industrial data	182
7.4	Training results for flux-balance model using industrial data	183
7.5	Optimal control for flux-balance model	184
9.1	Optimal control using an Euler integrator (10 stages)	192
9.2	Optimal control using an Euler integrator (25 stages)	193
9.3	Example of identification problem overfitting	194
B.1	$v(t)$ identification: two-step IDP for data run 2 (50 stages, 5 P_T data, with noise)	208
B.2	$v(t)$ identification: two-step IDP for data run 3 (50 stages, 5 P_T data, with noise)	209
B.3	$v(t)$ identification: two-step IDP for data run 4 (50 stages, 5 P_T data, with noise)	210
B.4	$v(t)$ identification: two-step IDP for data run 5 (50 stages, 5 P_T data, with noise)	211
B.5	$v(t)$ identification: two-step IDP for data run 6 (50 stages, 5 P_T data, with noise)	212
B.6	$v(t)$ identification: two-step IDP for data run 7 (50 stages, 5 P_T data, with noise)	213
B.7	$v(t)$ identification: two-step IDP for data run 8 (50 stages, 5 P_T data, with noise)	214

Chapter 1

Introduction

The original goal of this study was to improve modeling techniques for optimization and control of industrial fed-batch bioreactors. The bioreactor modeling problem is a very difficult one because any model of the process is always a very crude approximation of the real biological system.

While some (Domach et al., 2000) have attempted highly detailed mechanistic models, these models are not typically useful in an industrial setting because the simplifications and assumptions necessary to create the models don't hold in an industrial setting. For instance, in the stationary phase there are mechanisms of growth inhibition and cell death that are not well understood, so purely mechanistic bioreactor models typically do very poorly in the stationary phase.

Some others (Xiong and Zhang, 2005) have gone to the other extreme and created purely empirical models, such as neural network models. While these purely empirical models will be successful if enough data are available, they share the weakness of all empirical models of poor extrapolation. In addition, even though the mass and energy balances are known for this problem, these models throw this fundamental information away. The model will recapture this information from the data, but only where sufficient data are available.

A middle ground between these two modeling philosophies is the hybrid dynamic model (Psichogios and Ungar, 1992). In a hybrid dynamic model, a mechanistic model

is combined with an empirical model. This can be in parallel (for example, a neural network could represent an error term that is added to a mechanistic model), or in series (for example, a neural network could represent a reaction rate term in a set of reaction mass balance equations). A series model where reaction rate terms are replaced with neural networks was called a parameter function neural network by Tholudur and Ramirez (1996) because the neural networks capture the dynamic model parameters in conservation equations.

An interesting and difficult problem is how to identify these parameter functions, especially if data are noisy or limited or both as is often the case with biological systems. Good results were obtained by Tholudur et al. (1999) by creating curve-fits through the experimental data, obtaining the state derivatives from the curve-fits, and using this to solve directly for the values of the parameter functions. This technique is useful, but is not possible if any states are unmeasured. A more general technique is introduced in Chapter 6 where the hybrid model identification problem is treated as an optimal control problem. This leads to difficulties of its own, because the optimal control problem can be ill conditioned and computationally expensive in the general case.

The iterative dynamic programming (IDP) optimal control algorithm of Luus (1990) described in Chapter 3 was chosen to solve the identification optimal control problem because of its robust behavior and lack of restrictions on the system. However, IDP is generally very computationally expensive and sometimes leads to active (noisy) controls, and both of these problems are harmful for the identification problem.

To reduce the active control and speed problems, a new two-step procedure is introduced where regular IDP is used to solve the problem to some specified tolerance or number of iterations, then a modified IDP is used to solve the problem to some additional tolerance or number of iterations. The modified IDP used in the second step includes control filtering and/or damping methods which reduce the control activity

much faster than regular IDP.

Chapter 4 explores improvements to the first part of this two-step process by considering the effect of control tolerance on achievable optimal control performance, introducing several new methods for adaptive control region adjustment, and discussing methods for deciding when to stop IDP.

Chapter 5 develops the second part of the two-step IDP procedure by comparing several new and existing methods for control smoothing with IDP. An improved implementation of stagewise linear continuous control discretization is given, as well as a new algorithm for stagewise linear controls that allows discontinuity at each control stage boundary.

The model of Park and Ramirez (1988) is introduced in Chapter 2. This model was designed for optimal control studies in order to maximize secreted heterologous protein from a yeast system. The mechanistic version of this model is used as the optimal control problem to illustrate efficiency and smoothing in Chapters 4 and 5. The hybrid version of this model, with reaction rate terms replaced with neural networks, is used to illustrate the optimal control identification technique in Chapter 6.

Chapter 6 explores the hybrid dynamic model identification problem. The current methods are described along with their limitations, which leads to the proposal of the new Webb-Ramirez technique where the hybrid model identification problem is treated as an optimal control problem, with the parameter function values as the controls. The practical consideration of maintaining feasible states with this method is discussed, followed by an example of the method using artificial data from the Park-Ramirez model. The effect of IDP parameters on the first part of the method is studied, then both parts of the method are used to identify the parameter function neural networks from the artificial data. Finally, the optimal controls of the mechanistic and the identified models are compared to further test the quality of the identification.

Chapter 7 presents a flux-balance model for an industrial scale fed-batch bioreac-

tor. This model addresses the original goal of improving industrial fed-batch bioreactor performance but is not directly related to the hybrid model approach in the other chapters. The flux-balance model is a mechanistic model based on the model by Lin et al. (2001) which was designed to model growth of *E. coli* along with respiratory condition and acetate production. This model attempts to predict the glucose and oxygen flux through the cell respiratory chain in order to predict oxygen starvation leading to acetate production, along with the changes in yields that go along with a shift from aerobic to anaerobic growth.

Software implementation of everything described in this work is available under the GNU GPL free software license (Free Software Foundation, 1991) at <http://danielwebb.us/research>.

Main contributions to the state of the art

- Introduced and demonstrated a new technique to identify hybrid dynamic models using an optimal control algorithm.
- Improved the iterative dynamic programming (IDP) algorithm in several ways:
 - * Developed a new two-step procedure to quickly obtain the optimal performance index and a smooth control profile (Section 3.7 and Chapter 8).
 - * Added three new ways to adaptively shrink the control region (Section 4.5).
 - * Demonstrated a new test for early stopping (Section 4.5.3).
 - * Demonstrated a new test control that extracts performance sensitivity for each stage and control (Section 4.6).
 - * Introduced a new stochastic filter that improves smoothness with much

less effect on the performance index than the current filtering technique (Section 5.1.4).

- * Demonstrated a new type of test control that greatly speeds convergence for smooth problems (Section 5.1.6).
- * Improved the algorithm for stagewise linear continuous controls so that multiple state grids can be used (Section 5.2).
- * Introduced a new algorithm for stagewise linear discontinuous controls for problems where many control discontinuities are expected (Section 5.3).
- * Demonstrated a new way to use variable stage lengths requiring one less dimension than the previous method (Section 5.4).

Chapter 2

The Park-Ramirez bioreactor model

The Park-Ramirez bioreactor model (Park and Ramirez, 1988) describes cell growth and heterologous protein production and secretion for genetically-engineered yeast. It has been widely used (Franco-Lara and Weuster-Botz, 2005; Balsa-Canto et al., 2005; Sarkar and Modak, 2004) in optimal control studies because it has multiple separate singular arcs.

The state equation is the set of ordinary differential equations

$$\begin{aligned}\dot{X} &= \mu X - \frac{q}{V}X \\ \dot{S} &= -Y\mu X + \frac{q}{V}(S_f - S) \\ \dot{P}_T &= f_P X - \frac{q}{V}P_T \\ \dot{P}_M &= \phi(P_T - P_M) - \frac{q}{V}P_M \\ \dot{V} &= q\end{aligned}\tag{2.1}$$

where

- X is cell concentration
- S is substrate (glucose) concentration
- P_T is total protein concentration
- P_M is secreted protein concentration
- V is system volume
- Y is the substrate yield per cell growth
- μ is specific cell growth rate
- f_P is specific protein production rate
- ϕ is specific protein secretion rate
- q is substrate feed rate
- S_f is substrate feed concentration

Initial conditions (time t_0) are

$$\begin{aligned}
 X(t_0) &= X_0 \\
 S(t_0) &= S_0 \\
 P_T(t_0) &= 0.0 \\
 P_M(t_0) &= 0.0 \\
 V(t_0) &= V_0
 \end{aligned} \tag{2.2}$$

and the kinetic parameters are

$$\mu = \frac{21.87S}{(S+0.4)(S+62.5)} \tag{2.3}$$

$$f_P = \frac{S e^{-5.0S}}{0.1+S} \tag{2.4}$$

$$\phi = \frac{4.75\mu}{0.12+\mu} \tag{2.5}$$

Figures 2.1, 2.2 and 2.3 are graphs of μ , f_P , and ϕ respectively. Table 2.1 gives the parameter values for all model parameters used for this study.

t_0	0
X_0	$1.0 \frac{\text{g}}{\text{L}}$
S_0 (optimal control)	$5.0 \frac{\text{g}}{\text{L}}$
S_0 (identification)	$0.1 \frac{\text{g}}{\text{L}}$
V_0	1.0L
S_f	$20.0 \frac{\text{g}}{\text{L}}$
t_f	15.0hr
q	$0.0 \leq q \leq 4.0 \frac{\text{L}}{\text{hr}}$

Table 2.1: Park-Ramirez model parameters.

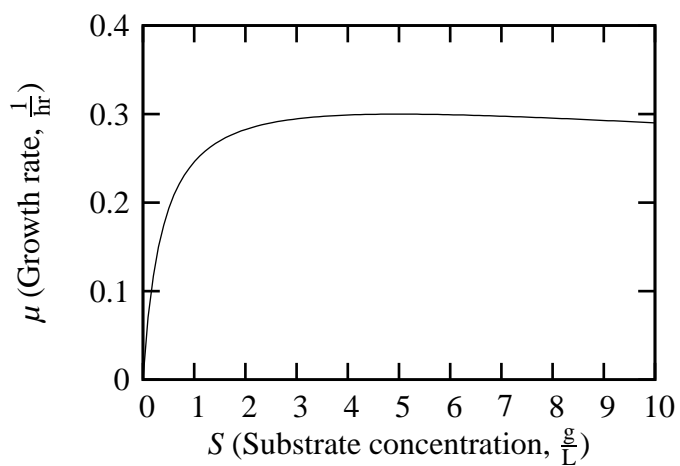


Figure 2.1: Kinetic parameter μ for the Park-Ramirez bioreactor model.

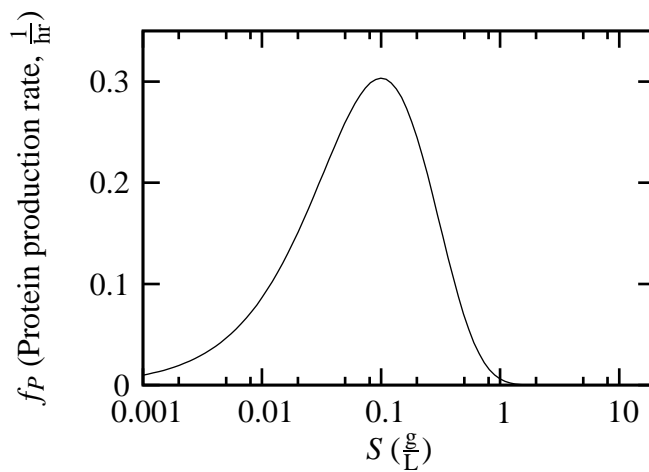


Figure 2.2: Kinetic parameter f_P for the Park-Ramirez bioreactor model.

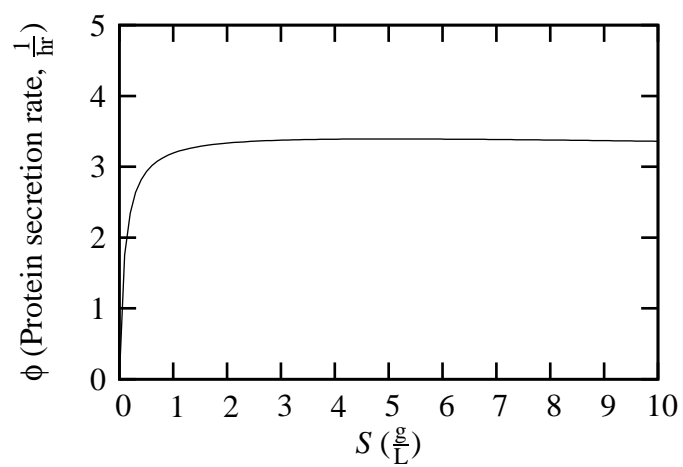


Figure 2.3: Kinetic parameter ϕ for the Park-Ramirez bioreactor model.

The optimal control problem for this model is to maximize the total mass of secreted protein at the final time. The optimal control performance index Φ (see Section 3.1) is thus

$$\Phi = P_M(t_f) \cdot V(t_f). \quad (2.6)$$

Figure 2.4 is the optimal control for the Park-Ramirez bioreactor model. The solution is a period of growth at the maximum growth rate at $S = 5.0 \frac{g}{L}$, a zero feed period until S falls to the maximum protein production concentration of $S = 0.1 \frac{g}{L}$, a period where feed maintains the maximum protein production concentration, and finally a period of maximum feed rate to stimulate as much protein secretion as possible before the final time. The optimal performance index for this problem is 32.76g to four significant figures. The performance index will be reported without units for the rest of the study.

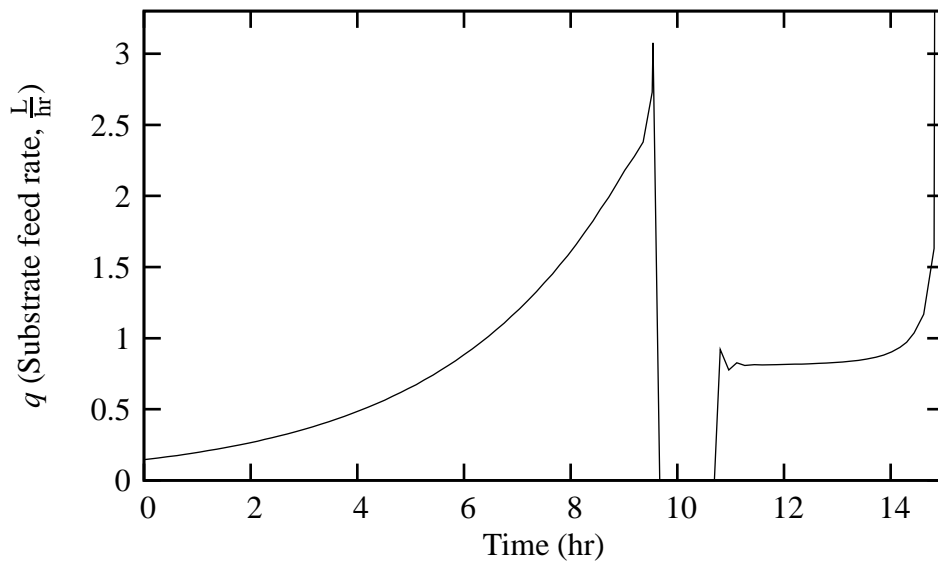


Figure 2.4: Optimal control of the Park-Ramirez bioreactor model.

The optimal control problem for the Park-Ramirez model will be used in Chapters 4 and 5 to illustrate a new two-step approach to solving the optimal control problem using iterative dynamic programming, and the hybrid version of this model with kinetic

parameters replaced with neural networks will be used to illustrate a new hybrid model identification technique in Chapter 6.

Chapter 3

Iterative dynamic programming

In this chapter, the optimal control problem is described, an overview of current solution methods is presented, and the basic iterative dynamic programming (IDP) optimal control algorithm is introduced.

3.1 The optimal control problem

We will consider the continuous dynamic model described by the vector differential equation

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, \mathbf{u}, t) \quad (3.1)$$

where \mathbf{x} is an $(I \times 1)$ state vector, \mathbf{u} is an $(J \times 1)$ control vector, and $\mathbf{x}(t = 0)$ is known. Each control has a fixed bound. The performance index is defined as

$$\Phi = \psi(\mathbf{x}(t_f)) \quad (3.2)$$

where ψ is an arbitrary function and t_f is the final time.

The optimal control problem is to choose the control policy $\mathbf{u}(t)$ in the time interval $0 \leq t \leq t_f$ that minimizes or maximizes the performance index. Although the performance index is only a function of the final states for algorithmic simplicity, this is still a general formulation since any performance function ψ can be transformed into a function of \mathbf{x} at the final time using pseudo state variables. The terms ‘minima’ and ‘local minima’ will refer to extrema and local extrema for the rest of the study, whether the performance index is being maximized or minimized.

3.2 Overview of solutions to the optimal control problem

There are two basic types of methods for solving the optimal control problem: indirect methods and direct methods. Indirect methods (Bryson and Ho, 1975; Ramirez, 1994) use the classical necessary conditions for optimality. Direct methods discretize the infinite dimensional problem into finite elements and fall into three basic types: control parametrization methods (Kraft, 1985), where only the controls are discretized, collocation methods (Neuman and Sen, 1973; Tsang et al., 1975), where both the controls and the states are discretized, and iterative dynamic programming. A non-linear programming technique such as sequential quadratic programming is used to find the parameters for the discretized model for control parametrization methods. There have also been attempts to use stochastic techniques such as genetic programming (Upreti, 2004) although these methods are not commonly used.

3.3 Introduction to IDP

Iterative dynamic programming (IDP) is an optimal control algorithm that extends the concept of Bellman's dynamic programming (Bellman, 1957) to the optimal control problem for discrete or continuous systems (Luus, 1990). In general, IDP is a very robust algorithm that places fewer restrictions on the system and is simpler to use than other algorithms (Luus, 2000). IDP only requires that the model be integrable, while control parametrization methods must extract sensitivity information using derivatives. IDP does not take derivatives of the model or use costate or sensitivity equations. Luus and Bojkov (1994) found that IDP consistently found the global optimum for a bifunctional catalyst problem when control parametrization using sequential quadratic programming found numerous local optima.

The main weaknesses of IDP relative to other optimal control methods are high computation cost and non-smooth control trajectories (Tholudur and Ramirez, 1997).

Fikar et al. (1998) compared IDP with control parametrization methods for two distillation problems. IDP took approximately 4 and 12 times as much computer time as control parametrization methods for the two problems. This should not be surprising since IDP is a stochastic approach instead of a gradient approach. For systems where a large number of stages are needed the difference between IDP and control parametrization is likely to be even greater. While a stochastic approach will never match the speed of a gradient approach for a smooth problem, the speed improvements introduced in this study narrow the performance gap between IDP and gradient methods. These two weaknesses of IDP are especially harmful to the solution of the Park-Ramirez bioreactor optimal control problem and to the solution of the hybrid dynamic identification problem examined in Chapter 6, so the changes to IDP explored in Chapters 4 and 5 focus on reducing these problems.

The basic idea behind IDP, like any application of dynamic programming, is to break up the optimal control problem into a series of simpler subproblems which are then solved separately. This is done by breaking the optimal control problem into a multistage decision problem with the control as a time-series sequence of decisions.

This work is presented as a practical explanation of the IDP algorithm and improvements to it. The derivation of IDP from dynamic programming principles is covered more fully by Tholudur (1998), and the authoritative reference for the IDP algorithm is the book by Luus (2000). The explanation here will start by describing the basic IDP algorithm with stagewise constant controls in Sections 3.4–3.6, then develop several modifications in the following chapters.

3.4 Basic IDP algorithm

The foundation of dynamic programming and IDP is Bellman's Principle of Optimality (Bellman, 1957):

An optimal policy has the property that whatever the initial state and the initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

In other words, if you know where you need to end up, you can start at the end and work backwards. This concept allows you to break up a very difficult problem into many smaller, less difficult problems. In the case of the optimal control problem, the problem is broken into K stages of length L , and the controls for the last stage are found first. The last stage start and end times are known, but the states at the beginning of the last stage are not. The two questions to answer are “what states to begin the last stage with?” and “how to choose the controls to try for the last stage?”

3.4.1 State and control grids

At the start of each iteration of IDP, two grids are created, one for states and one for controls. The control grid **TestControlGrid** is chosen first: for each stage, each of R test controls is chosen using some random distribution centered on the previous iteration optimal control. The exception to this is that the first test control uses the previous iteration’s best control. The “best control” at the start of the algorithm is arbitrarily chosen, such as the midpoint of the control limits.

Once the control grid is chosen, the model is integrated from start to finish for each of S state grids to create the **StateGrid** data structure. The control from **TestControlGrid** used for a state grid integration is the corresponding control from the test control grid (ie. controls for the integration to find **StateGrid** state point 2 come from **TestControlGrid** test control 2). Therefore, there must be at least as many test controls as state grids.

3.4.2 Finding the best control for a state grid at each stage

Once **TestControlGrid** and **StateGrid** are created there are options for the controls to use and the states to start the last stage with. For each state grid, the states stored in **StateGrid** are used as the initial condition for the last stage. The model is integrated from the start of the last stage to the final time, once for each of the possible test controls in **TestControlGrid**, and the one that results in the best final performance index is stored in **OptimalControlGrid**.

The algorithm then moves back to stage $K - 1$. The initial state condition for each state grid is obtained from **StateGrid** at stage $K - 1$. For each state grid, all test controls in **TestControlGrid** are evaluated to integrate the model across stage $K - 1$. Since we only want to choose between the test controls for the current stage, the previously found optimal for the state grid closest to the current system state is used to integrate across stage K . The test control with the best final performance index is stored in **OptimalControlGrid**.

The algorithm continues working backwards through the stages, integrating just the current stage for each of the S state grids using the R possible test controls and then continuing to the model final time using previously found best controls. **OptimalControlGrid** is built up for each state grid and stage in this manner.

3.4.3 Iteration optimal

Once the best controls for all state grids are found and stored in **OptimalControlGrid**, the model is integrated from start to finish to find the iteration optimal control. As each stage boundary is passed, the state grid closest to the current system state is chosen to be the state grid for that stage. The control from **OptimalControlGrid** corresponding to that state grid is used for that stage. These controls are saved in **OptimalControl** and are the iteration optimal control.

3.4.4 Continuing iterations

After each iteration, the allowed range for controls is reduced by the control region contraction factor γ_C . Typically γ_C is in the range of 0.90 to 0.98. After a certain number of iterations, the control region is increased using a region restoration factor γ_R , which brings the control region partway back to its original size. The control region size thus follows a descending saw pattern, which Luus (2000) used instead of a single continuously decreasing control region to prevent premature collapse of the control region. Typically γ_R is in the range of 0.6 to 0.9. A group of iterations between restoration is called a pass.

3.5 IDP basic algorithm - subalgorithms

The complete IDP algorithm is broken into several sub-algorithms in this presentation. Several variations of the algorithm presented in this work have common subparts, and this allows the presentation of each variation to be complete and precise without redundancy.

Algorithm 1 Initialization algorithm

1. Choose the number of test controls R
 2. Choose the number of state grids S
 3. Choose the initial control region size $r_{j,0}$ for each control j
 4. Choose the region contraction factor γ_C
 5. Choose the region restoration factor γ_R
 6. Choose the number of iterations per pass A
 7. Choose the number of passes B
-

3.6 Stagewise constant IDP

The complete IDP algorithm for stagewise constant controls similar to the one given by Luus (2000) is started by executing the pass algorithm (Algorithm 2), which calls Algorithm 5 for each iteration. The code implementing this algorithm is described

Algorithm 2 Pass algorithm

1. Execute Algorithm 1 (initialization algorithm)
 2. **for** pass b from $0..B - 1$ **do**
 3. **for** each control j from $1..J$ **do**
 4. **for** iteration a from $0..A - 1$ **do**
 5. Set control region size $r_j = \gamma_C^a \gamma_R^b r_{j,0}$
 6. **end for**
 7. Execute Algorithm 5, 7 or 8 (single iteration algorithm)
 8. **end for**
 9. **end for**
-

Algorithm 3 Grid generation algorithm

1. Divide the time interval $[0, t_f]$ into $1..K$ time stages, each of length L
 2. **for** stage $1..K$ **do**
 3. Center control region r_j for each control j on the previous iteration optimal control (or $r_{j,0}$ if the first iteration)
 4. Set the first test control in **TestControlGrid** to the previous iteration optimal control for this stage
 5. **for** test control j from $2..R$ **do**
 6. Set **TestControlGrid** for this stage and test control using the uniform distribution within the control region r_j
 7. **end for**
 8. **end for**
 9. **for** state grid from $1..S$ **do**
 10. **for** stage $1..K$ **do**
 11. At beginning of stage, record states in **StateGrid** for this state grid
 12. Choose controls from **TestControlGrid** corresponding to this state grid
 13. Integrate to the next stage
 14. **end for**
 15. **end for**
-

Algorithm 4 Extract **OptimalControl** from **OptimalControlGrid** algorithm

1. **for** stage $1..K$ **do**
 2. At the beginning of the stage, pick the state grid in **StateGrid** closest to the current system state
 3. Find the control in **OptimalControlGrid** corresponding to the state grid from the previous step
 4. save the control in **OptimalControl**
 5. Integrate to the next stage using the control
 6. **end for**
-

in Appendix A.

Algorithm 5 Iteration algorithm for stagewise constant controls

1. Execute Algorithm 3 (**TestControlGrid** and **StateGrid** generation algorithm)
 2. **for** each stage backward from $K..1$ **do**
 3. **for** each of the S state grids **do**
 4. Choose the state condition at the start of the stage from **StateGrid**
 5. **for** each of the R test controls **do**
 6. Integrate the current stage using test control from **TestControlGrid**
 7. **for** each of the remaining stages **do**
 8. Choose the state grid in **StateGrid** whose state is closest to the current system state
 9. Choose the controls from **OptimalControlGrid** corresponding to the state grid in the previous step and integrate to the next stage
 10. **end for**
 11. If performance index is the best of the test controls being evaluated, store test control in **OptimalControlGrid** for this stage and state grid
 12. **end for**
 13. **end for**
 14. **end for**
 15. Execute Algorithm 4 to extract **OptimalControl** from **OptimalControlGrid**
-

3.7 A two-step method for efficient IDP

In the early iterations of the solution process when the control region is large, IDP is more robust than gradient-based algorithms because it tests many combinations of control paths, instead of just finding the overall downhill direction and following it. Once the performance index is close enough to the optimum that local minima are not a concern, the tables turn. Gradient-based algorithms often converge increasingly rapidly to high precision as a solution is approached. IDP, on the other hand, still converges slowly because there is no assumption of smoothness near the solution.

A naïve solution to this problem is to use IDP until the performance index fails to decrease for some number of iterations, then stop. This is not feasible for problems with low control sensitivity (such as both the problems in this study) because the controls remain far from the optimal controls until the performance index has converged to very high precision.

Another potential solution is to use IDP until the performance index is close to the solution, then use a gradient-based algorithm to convergence. However, IDP can solve problems that gradient-based algorithms can't, so this still does not fully solve the problem.

The method advocated here is a two-step procedure: first using basic IDP with adaptive region size updates (Section 4.5) until sufficiently close to the solution (Section 4.3), then using one of the control smoothing techniques from Chapter 5 until controls are stable. The first step is needed because the smoothing methods in Chapter 5 often lead to local minima, and the second step is needed to speed convergence of the control profile. This two-step approach retains the advantages of IDP while making it computationally more competitive with gradient-based algorithms.

The two-step method will be developed throughout Chapters 4–6, then a summary of the method and recommendations will be presented in Chapter 8.

Chapter 4

Iterative dynamic programming efficiency

IDP is a computationally expensive algorithm, requiring millions of complete system integrations to find the optimal control for problems with low control sensitivity such as the Park-Ramirez optimal control problem. This chapter deals with general ways to speed up the IDP algorithm that are applicable to the basic IDP algorithm and both steps of the two-step procedure introduced in Section 3.7.

4.1 Choosing IDP parameters

Understanding how IDP parameters affect computation cost is important in order to reduce it. The limiting step in IDP is the model integration while evaluating each test control (steps 6 and 9 in Algorithm 5). To simplify the analysis, all integration will be considered equivalent for an equal amount of time. For example, 15 integrations of the model between $t = 14$ and $t = 15$ will be considered equivalent to one integration between $t = 0$ and $t = 15$. This assumption could be inaccurate with a variable stepsize integrator since some control paths may require a smaller average stepsize than others. However, it is approximately correct and allows an estimate of the order of computation for each parameter. It also allows a better comparison than processor time when comparing results using different programming languages, different processor speeds, and different processor architectures (see Section 5.7).

There are five parameters that affect computation time with the basic IDP al-

gorithm: number of passes (B), number of iterations (A), number of state grids (S), number of test controls (R), and number of stages (K). The number of equivalent integrations I_{eq} follows directly from the IDP algorithm and is given by

$$I_{\text{eq}} = B \cdot A \left(S \cdot R \sum_{k=0}^{K-1} 1 - \frac{k}{K} \right) \quad (4.1)$$

which simplifies to

$$I_{\text{eq}} = B \cdot A \left(1 + \frac{K(K-1)S \cdot R}{2} \right). \quad (4.2)$$

4.1.1 Number of passes and iterations

The number of passes and iterations can more effectively be chosen by using adaptive region size determination (see Section 4.5).

4.1.2 Number of state grids

Luus observed that even some very difficult problems can be solved with one state grid (Luus et al., 1992), although a basic fed-batch bioreactor problem needed a minimum of 23 state grids to solve (Hartig et al., 1995) and a problem so simple it can be solved by common-sense reasoning required 25 state grids (Luus, 1998). In another study, Bojkov and Luus (1993) concluded that more than three state grids did not increase the convergence rate or chance of finding the global optimum. Obviously only one state grid should be used if possible.

4.1.3 Number of test controls

In the early work on IDP (before the multi-pass method was introduced), the number of test controls was balanced with the region contraction factor to prevent early region collapse. With adaptive region size (Section 4.5), the number of test controls is balanced with the number of history iterations to prevent early region collapse. In general, a small number of test controls is likely to be more efficient (Luus, 2000).

This study uses 5 test controls for the Park-Ramirez optimal control problem and 6 test controls for the hybrid Park-Ramirez identification problem.

4.1.4 Number of stages

The most important IDP parameter with respect to computational cost by far is the number of stages. This is true not only because of the number of equivalent integrations in Equation 4.2, but because of the effect described in Section 5.1 where problems with low sensitivity have increasingly active control policies as more stages are used.

4.1.5 Number of controls

Although not strictly a parameter of IDP but a parameter of the optimal control problem, it should be mentioned that the single most important parameter with respect to computation is the number of controls. Bellman (1961) called the effect of the number of controls on computation the “curse of dimensionality.” This curse can be simply illustrated by imagining sampling 10 points along a line (a one-dimensional space). To get equivalent coverage of a ten-dimensional space, 10^{10} sampling points would be required. The beauty of IDP is that it reduces this curse somewhat by limiting the allowable control domain using state grids, but the curse is still with us. Imagine an optimal control path for a single control that is only accessible from 1% of the control region. To reach that path with a high probability, the number of iterations times the number of test controls should be several hundred before the region is reduced beyond the path. If there were three similar controls, the number of iterations times the number of test controls would need to be in the millions.

4.2 Parallel implementation of IDP

Although the implementation used in this study is a single-processor implementation, one way to speed up IDP is to create a parallelized version of the algorithm (Hartig et al., 1999) which runs on multiple processors. IDP is most easily parallelized in two ways:

- Each state grid evaluation (step 3 of Algorithm 5 for stagewise constant controls) is performed on a separate processor.
- Each test control evaluation (step 5 of Algorithm 5 for stagewise constant controls) is performed on a separate processor.

For most problems which require only one state grid, only the second option makes sense. However, a large number of test controls is less efficient than a small number, so the expected return on investment for parallelization will be approximately one order of magnitude for problems with a single state grid and two orders of magnitude for problems with many state grids.

Using a parallel version of IDP may make sense if a computational cluster is available using standard parallelization libraries such as PVM (Sunderam et al., 1994) or MPI (Gropp et al., 1996) since these clusters typically contain 10 to 100 processors. As of this writing, multiprocessor single machines typically have 8 or less processors, but there are designs announced by chip manufacturers that will supposedly produce tens or possibly even hundreds of processor ‘cores’ on a single processor die. If these products do come to market, a parallelized version of IDP will probably be needed to take advantage of these multiple cores.

4.3 Control precision and error

In an industrial setting, a control can only be set within a certain tolerance. In the case of a fed-batch bioreactor such as the Park-Ramirez optimal control problem, controlling the substrate feed rate to within a few percent would be considered good control. A reasonable question to ask is: “how much will control tolerance affect the achievable performance index?” For example, for online optimal control it makes no sense to obtain the optimal performance index to within 0.01% if control precision limits the attainable performance index to within 1% of the optimal.

To explore how imperfect control affects the Park-Ramirez optimal control problem, two cases were examined: noisy controls and control offset. To test how control noise affects the performance index, the optimal control profile was obtained normally, then a second control profile was created by adding Gaussian noise to the first. The system was simulated with both sets of controls to find the final performance index for each case and obtain the relative performance index error. Figure 4.1 shows the relative error in performance index caused by control noise. Substrate feed noise of 2% will lead to a performance index error of approximately 1%.

Another common problem in an industrial setting is control offset. To examine the effect of control offset on performance index, the optimal control profile was obtained normally, then a second control profile was created by adding a relative offset to the first. The system was simulated with both sets of controls to find the final performance index for each case and obtain the relative performance index error. Figure 4.2 shows this result. Control offset of 2% will lead to a performance index error of approximately 0.1%, indicating that control offset is probably less serious than control noise for this problem.

An analysis of the control error’s effect on achievable performance index is one way to decide how many iterations to use for the first step in the two-step procedure.

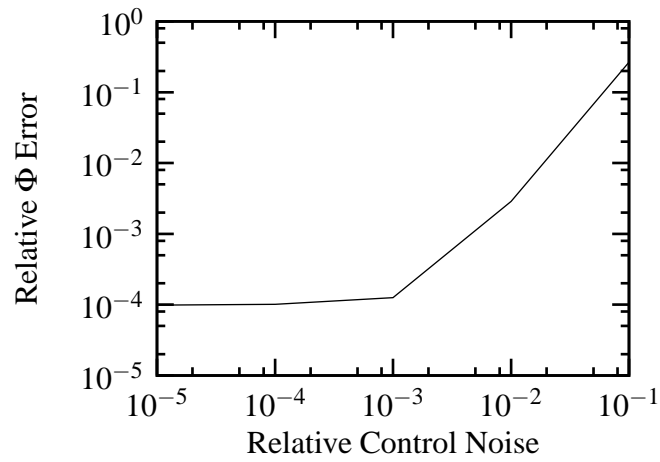


Figure 4.1: Effect of Gaussian control noise on performance index error for the Park-Ramirez optimal control problem. The standard deviation of the Gaussian noise is the control value times the relative control noise.

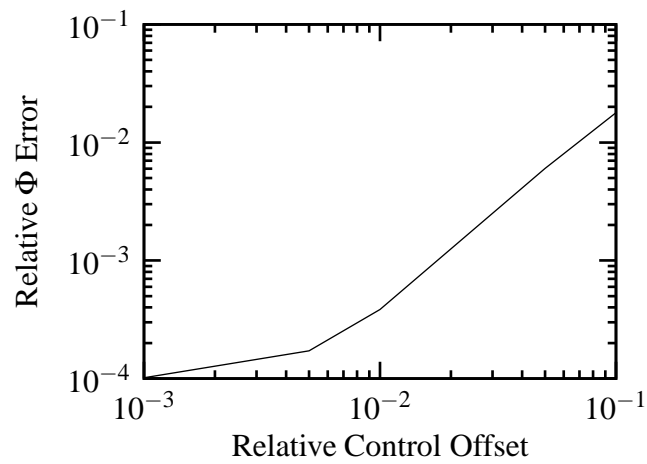


Figure 4.2: Effect of relative control offset on performance index error for the Park-Ramirez optimal control problem.

For this study, a relative control noise of 2% will be assumed, so during the first step of the two-step procedure the basic IDP algorithm will be run for enough iterations to get within 1% of the optimal performance index. The number of iterations needed to do this will be found in Section 4.5.

4.4 Integrator tolerance

A custom Runge-Kutta 4/5 integrator was used for the model integration. It is important with all integrators to choose appropriate error tolerances so that the proper balance is struck between solution accuracy and speed. The procedure for obtaining reasonable integrator tolerances used here was:

- (1) Start with a very small absolute tolerance so that relative tolerance dominates at all times during the integration.
- (2) Choose a reasonable relative tolerance, balancing speed and accuracy.
- (3) Increase the absolute tolerance if needed.

An absolute tolerance of 10^{-11} was initially chosen. At this small value, relative tolerance will dominate throughout the model integration. When examining the effect of integrator relative tolerance, there are three important questions:

- How does the relative tolerance affect the performance index?
- How does the relative tolerance affect the control profile?
- How much computation is used versus relative tolerance?

Figure 4.3 shows the performance index for simulations run 100 iterations using a wide range of integrator relative tolerance. Surprisingly, the final performance index is nearly identical except for the case where relative tolerance is 10^{-1} . The computation

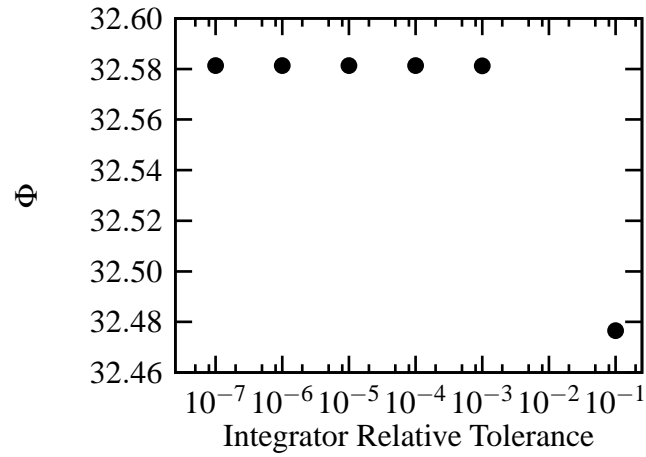


Figure 4.3: Effect of integrator relative tolerance on performance index for the Park-Ramirez optimal control problem.

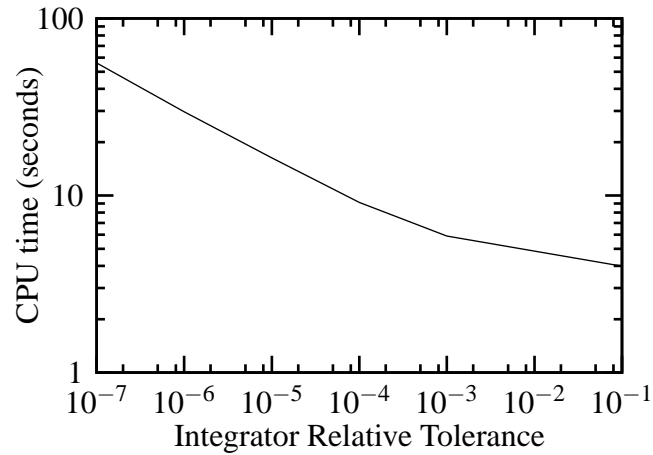


Figure 4.4: Computation time vs. integrator relative tolerance for the Park-Ramirez optimal control problem. Absolute tolerance was 10^{-11} .

times corresponding to the simulations are given in Figure 4.4 and demonstrate the importance of picking the integrator tolerance wisely.

Figure 4.5 shows the effect of integrator relative tolerance on a stagewise linear continuous solution of the Park-Ramirez optimal control problem. In line with the finding that the performance index is very insensitive to integrator relative tolerance, the control profile also seems to be very insensitive. Based on these results, an integrator relative tolerance of 10^{-4} was conservatively chosen for all solutions of the Park-Ramirez optimal control problem.

Once relative tolerance was chosen, the computational effect of absolute tolerance (Figure 4.6) was found to be insignificant, so the absolute tolerance was left at 10^{-11} .

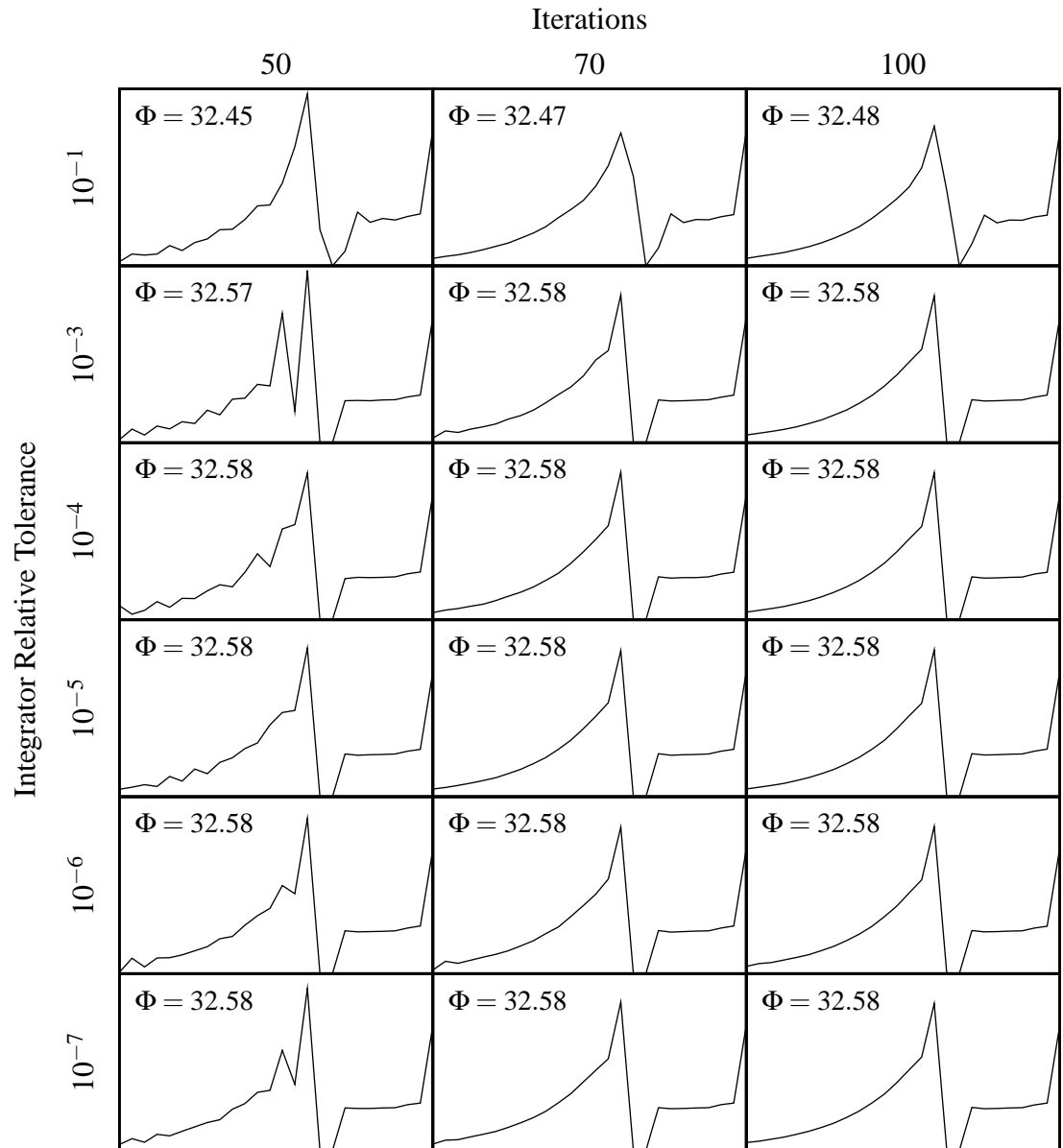


Figure 4.5: Effect of integrator relative tolerance on solutions to the 25 stage Park-Ramirez optimal control problem. Each subplot in the grid shows substrate feed rate vs. time.

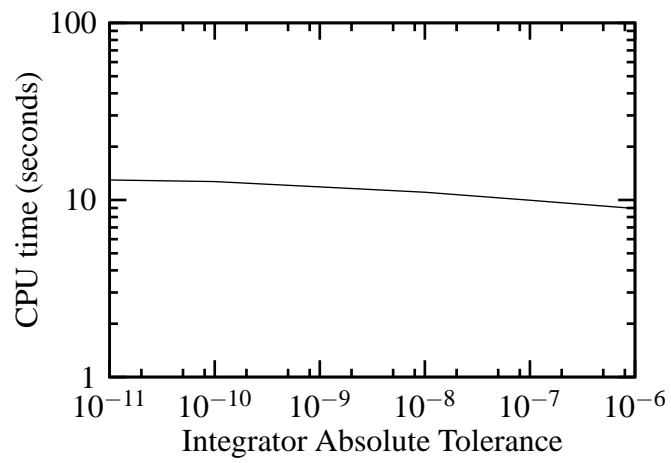


Figure 4.6: Computation time vs. integrator absolute tolerance for the Park-Ramirez optimal control problem. Relative tolerance was 10^{-4} .

4.5 Adaptive control region size

4.5.1 Adaptive control region size methods

The basic IDP algorithm uses a predetermined control region size sequence, which often leads to many iterations where no progress is made because the region is either too large or too small. Mekarapiruk and Luus (1999) proposed that the region size for a control at each stage be equal to the change in that control during the previous pass, shown as Equation 4.3. Also, a lower limit for region size was chosen to keep the control region from collapsing to zero.

The Mekarapiruk-Luus method is evaluated at the end of each pass and is

$$r_{j,k}^* = \left| u_{j,k}^{\text{start}} - u_{j,k}^{\text{end}} \right|, \quad j = 1, \dots, J \quad \text{and} \quad k = 1, \dots, K \quad (4.3)$$

where $r_{j,k}^*$ is the new control region size for the next pass, $u_{j,k}$ is the value of control j at stage k , $u_{j,k}^{\text{start}}$ is the best value of $u_{j,k}$ at the start of the previous pass and $u_{j,k}^{\text{end}}$ is the best value of $u_{j,k}$ at the end of the previous pass. This method has three weaknesses:

- It only changes the region size once per pass, and some problems can be solved in so few iterations using the two-step method that this results in irregular region size updates. This is a minor cost for most problems currently being studied, but if problems with very expensive model evaluation were being studied this could be important.
- It only contracts the control region and never expands it. It is conceivable that the ideal control action at a given stage could increase at some time during the solution. This effect is in fact observed during the solution of the two problems in this study.
- It has been observed by watching animations of IDP progress that controls tend to “bounce,” especially when pivot point test controls (Section 5.1.6) are used.

It is possible for a control to have the same value at the start and end of a pass while bouncing to other values during the pass. With the Mekarapiruk-Luus method the region size will collapse permanently to the minimum in one pass if this happens.

Three similar new methods are proposed here for adaptive control region size update. All store a history of the optimal control for the previous A_H iterations and update the region size after every iteration. Passes are not used for these methods. All methods also have a minimum relative control region size and a maximum decrease in region size per iteration for each control and stage. The minimum relative control region size is

$$r_{j,\min} = \alpha_r \cdot r_{j,0} \quad (4.4)$$

where $r_{j,\min}$ is the minimum region size for control j , α_r is the minimum relative region parameter, and $r_{j,0}$ is the initial region size for control j . For this study α_r is always 0.001. The maximum decrease in region size per iteration is

$$r_{j,k}^{a+1,\min} = r_{j,k}^a \exp \left[\frac{\ln(\beta_r)}{A_H} \right] \quad (4.5)$$

where $r_{j,k}^{a+1,\min}$ is the smallest allowed region size for the next iteration and β_r is the minimum fraction the region size can contract in A_H iterations. For this study β_r is always 0.01. This allows the region to shrink most of the way to the minimum region size in A_H iterations.

The Maximum Delta Method chooses the new region size to be the maximum change of a control between any two consecutive iterations during the history, specifically

$$r_{j,k}^{a+1} = k_r \cdot \text{MAX} \left(\left| u_{j,k}^{a-n} - u_{j,k}^{a-n-1} \right| \right),$$

$$n = 0, \dots, A_H - 2; j = 1, \dots, J; k = 1, \dots, K. \quad (4.6)$$

The Average Delta Method chooses the new region size to be the average change of a control between all pairs of consecutive iterations during the history, specifically

$$\begin{aligned} r_{j,k}^{a+1} &= k_r \cdot \text{AVG}(|u_{j,k}^{a-n} - u_{j,k}^{a-n-1}|), \\ n &= 0, \dots, A_H - 2; j = 1, \dots, J; k = 1, \dots, K. \end{aligned} \quad (4.7)$$

The Span Method chooses the new region size to be the maximum range of a control between any two iterations during the history, specifically

$$\begin{aligned} r_{j,k}^{a+1} &= k_r \left(\text{MAX}(u_{j,k}^{a-n}) - \text{MIN}(u_{j,k}^{a-n}) \right), \\ n &= 0, \dots, A_H - 1; j = 1, \dots, J; k = 1, \dots, K. \end{aligned} \quad (4.8)$$

For all three methods, $r_{j,k}^{a+1}$ is the new control region for the next iteration $a + 1$ for control j and stage k , k_r is the region contraction factor, $u_{j,k}^a$ is the value of control j at stage k for iteration a , J is the number of controls, K is the number of stages, AVG is the average operator (iterated only over the n index), MIN is the minimum operator (iterated only over the n index), and MAX is the maximum operator (iterated only over the n index).

The Mekarapiruk-Luus method has only one parameter (iterations per pass) but to allow equivalent comparison with the three new methods it was also assumed to use a region contraction factor and was multiplied by k_r as with the other methods. Also, A_H will mean the iterations per pass for the Mekarapiruk-Luus method.

4.5.2 Adaptive control region parameters

The adaptive control region size methods replace the fixed region contraction (γ_C) and restoration (γ_R) factors with A_H and k_r , so it is helpful to study the effects of these two parameters. In general, it is known that using γ_C and γ_R that cause the region to contract too rapidly will lead to local minima being found. It is expected, then, that combinations of A_H and k_r which cause the region to contract too rapidly will also lead

to local minima. The expected effect is that if region contraction is too aggressive poor solutions will result, but if the region contraction is too slow it will be less efficient.

To compare the four methods and determine suitable values for the parameters k_r and A_H for the Park-Ramirez optimal control problem, several parametric studies using five values of A_H and five values of k_r were performed. Table 4.1 shows the number of iterations required to reach 99% and 99.9% of the optimal performance index for 25 stages, and Table 4.2 shows the number of iterations required to reach 99% and 99.9% of the optimal performance index for 50 stages.

The case with 25 stages does not have a clear pattern to predict when local minima will be found, although most of the local minima are found with the Mekarapiruk-Luus and the Span methods, which are the two methods most similar conceptually. The problems with the Mekarapiruk-Luus method are mostly expected to cause over-contraction of the control region which can lead to local minima or less efficient solution because the region is at its minimum for much of the run.

The case with 50 stages follows the expected trend of finding local minima when the number of history iterations is small, although the k_r does not always have a large effect. Interestingly, for the 50 stage case the Mekarapiruk-Luus method works as well as the new methods despite its poor performance in the 25 stage case.

In Section 4.3, it was decided that step one of the two-step procedure should obtain the performance index within approximately 1% of the optimal performance index. The Mekarapiruk-Luus and Span methods were ruled out for this because they found several local minima in the 25 stage case, and the Maximum Delta method was chosen over the Average Delta method because it seems more predictable in the 50 stage case. However, this selection was somewhat arbitrary since all of the methods were fine with $A_H = 27$ or $A_H = 35$.

Based on these results, 40 iterations using the Maximum Delta method with $A_H = 27$ and $k_r = 1.5$ was chosen for step one of the two-step procedure. The full

Target Φ

		$0.99\Phi^{\text{optimal}}$					$0.999\Phi^{\text{optimal}}$						
		A_H					A_H						
			9	14	20	27	35		9	14	20	27	35
Mekarapiruk-Luus	k_r	0.7	29	23	23	24	24	0.7	82	65	57	62	62
		1.0	24	24	23	24	24	1.0	86	68	64	81	69
		1.5	21	24	25	24	24	1.5	—	67	75	81	72
		2.0	—	26	26	24	24	2.0	—	95	—	82	75
		4.0	37	29	30	24	24	4.0	—	87	—	100	79
Average Delta	k_r	1.0	24	25	24	24	24	1.0	37	40	46	43	45
		2.0	25	24	24	24	24	2.0	42	30	43	47	50
		4.0	22	23	24	24	24	4.0	31	32	39	57	54
		8.0	29	25	24	24	24	8.0	56	31	65	72	61
		16.0	25	27	26	24	24	16.0	41	63	—	75	68
Maximum Delta	k_r	0.7	22	24	23	24	24	0.7	28	46	51	65	54
		1.0	23	23	24	24	24	1.0	39	31	43	70	78
		1.5	22	25	25	24	24	1.5	33	45	62	78	80
		2.0	19	24	25	24	24	2.0	33	36	70	79	86
		4.0	29	32	27	24	24	4.0	77	96	101	108	99
Span	k_r	0.7	22	25	24	24	24	0.7	45	44	65	71	64
		1.0	22	25	25	24	24	1.0	36	61	58	65	57
		1.5	21	25	25	24	24	1.5	54	37	—	80	78
		2.0	20	25	29	24	24	2.0	32	53	—	81	81
		4.0	28	31	31	24	24	4.0	64	110	—	108	100

Table 4.1: IDP iterations required to reach 99% or 99.9% of the optimal performance index Φ^{optimal} for the 25 stage Park-Ramirez optimal control problem. One state grid and five random test controls were used. A ‘—’ indicates the target performance index was not reached within 200 iterations.

Target Φ

		$0.99\Phi^{\text{optimal}}$					$0.999\Phi^{\text{optimal}}$						
		A_H					A_H						
			9	14	20	27	35		9	14	20	27	35
Mekarapiruk-Luus	k_r	0.7	27	25	27	27	27	0.7	55	50	48	42	45
		1.0	29	26	26	27	27	1.0	61	41	41	41	44
		1.5	26	52	27	27	27	1.5	57	—	48	45	44
		2.0	29	47	27	27	27	2.0	56	—	45	45	42
		4.0	83	83	28	27	27	4.0	—	—	127	52	43
Average Delta	k_r	1.0	28	25	27	27	27	1.0	39	33	49	49	49
		2.0	37	26	26	27	27	2.0	43	—	—	37	46
		4.0	25	26	27	27	27	4.0	38	42	—	40	49
		8.0	50	60	28	27	27	8.0	—	—	46	40	45
		16.0	57	30	29	27	27	16.0	—	51	45	46	40
Maximum Delta	k_r	0.7	24	43	26	27	27	0.7	—	—	44	41	45
		1.0	23	54	27	27	27	1.0	42	—	44	43	44
		1.5	48	99	28	27	27	1.5	—	—	38	39	46
		2.0	47	69	29	27	27	2.0	—	—	49	42	43
		4.0	72	76	30	27	27	4.0	—	—	66	43	43
Span	k_r	0.7	32	60	26	27	27	0.7	36	—	41	38	48
		1.0	28	51	29	27	27	1.0	—	—	35	42	48
		1.5	64	54	26	27	27	1.5	—	—	41	42	51
		2.0	57	75	30	27	27	2.0	—	—	44	51	49
		4.0	71	29	28	27	27	4.0	—	71	72	43	56

Table 4.2: IDP iterations required to reach 99% or 99.9% of the optimal performance index Φ^{optimal} for the 50 stage Park-Ramirez optimal control problem. One state grid and five random test controls were used. A ‘—’ indicates the target performance index was not reached within 200 iterations.

set of parameters used for step one of most two-step solutions for the rest of the study are in Table 4.3.

In general, the history iterations should be large enough that there is a high probability that several representative control changes happen within the history. This can be determined by running IDP on the model and observing how often test controls are chosen. Unless high efficiency is very important, it is easier and safer to err by picking the history iterations too large.

Stage control discretization	Stagewise constant
Stage length mode	Fixed uniform stage length
Iterations	40
State grids	1
Random test controls	5
Adaptive control region method	Maximum Delta
Adaptive control region history	27 iterations
Adaptive control region k_r	1.5

Table 4.3: Default IDP parameters for step one of the two-step procedure for the Park-Ramirez optimal control problem. Model parameters are in Table 2.1 (page 8).

4.5.3 Stopping criteria

For the original IDP algorithm with a fixed number of iterations and passes, the stopping criterion was simply that the number of iterations and passes was complete. For adaptive region size, it can be useful to have stopping criteria so that the algorithm can finish as soon as possible.

One possible method would be to stop when the performance index falls below some threshold, but this is problematic because the best performance index will change as model parameters change. A better method is to stop IDP when the performance index stops improving. The relative improvement stopping parameter γ_{stop} is defined as

$$\gamma_{\text{stop}} = \frac{\Phi^H - \Phi^0}{\Phi^0 \cdot A_S} \quad (4.9)$$

where Φ^H is the performance index furthest back in the stopping history, Φ^0 is the current performance index, and A_S is the number of stop history iterations.

Stopping works well when there is continuous and predictable improvement (such as the Park-Ramirez optimal control problem to reasonable precision), but not for high precision or difficult problems, because improvements may be sudden after many iterations with little or no improvement. For these problems a fixed number of iterations is still advised. Also, most solutions in this study use a fixed number of iterations for presentation simplicity. Figure 4.7 presents final solutions for a range of γ_{stop} . Notice that more stages require a smaller γ_{stop} to obtain a smooth control profile, as explained in Section 5.1.1.

4.6 Performance index control sensitivity

If two additional test controls are added, one slightly higher and one slightly lower than the current optimal control, performance index control sensitivity information can be extracted during each IDP iteration. IDP always integrates the model to the final time and obtains the performance index for every test control evaluation, so

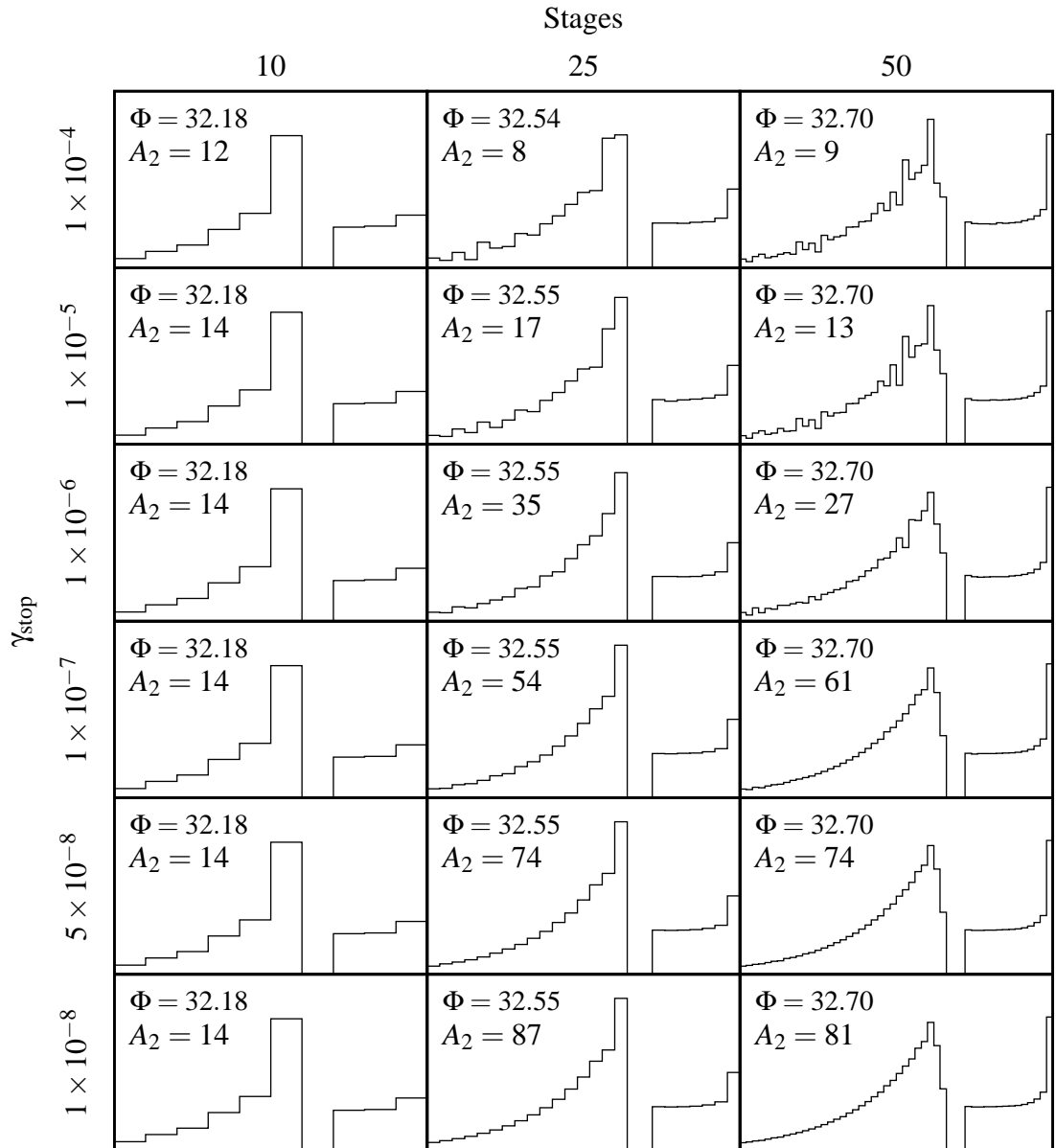


Figure 4.7: Effect of stopping parameter γ_{stop} on the two-step IDP solution of the Park-Ramirez optimal control problem. Table 4.3 (page 38) gives the IDP parameters for the first step. The IDP parameters for the second step were the same as the first except for using 3 random test controls and 2 pivot point test controls (Section 5.1.6). A_2 is the number of iterations for step two to reach γ_{stop} .

this is trivial to implement. In the current implementation, the forward and backward sensitivities are used instead of a central difference sensitivity so that information about which controls have reached a bound does not have to be stored.

The forward and backward normalized performance index control sensitivities $\xi_{\Phi u}^+$ and $\xi_{\Phi u}^-$ for each control and stage are defined as

$$\xi_{\Phi u}^- = \left| \frac{\Delta\Phi \cdot M_{u_j} \cdot K}{M_{\Phi}(u_j - \Delta u_j)} \right| \quad (4.10)$$

and

$$\xi_{\Phi u}^+ = \left| \frac{\Delta\Phi \cdot M_{u_j} \cdot K}{M_{\Phi}(u_j + \Delta u_j)} \right| \quad (4.11)$$

where $\Delta\Phi$ is the change in performance index corresponding to control change Δu_j , M_{Φ} is a typical or maximum value of the performance index used to normalize the term, M_{u_j} is a typical or maximum value of control u_j used to normalize the term, and K is the number of stages. For this study $\Delta u_j = 0.001M_{u_j}$. Because sensitivity calculation is only run once at the end, the integrator relative tolerance was conservatively chosen to be 10^{-7} for sensitivity calculation.

Figure 4.8 shows the normalized performance index control sensitivities for the Park-Ramirez optimal control program. The sensitivities are not particularly low, as might be expected given the discussion of the active control problem in Section 5.1.1. A more precise explanation is that active controls are a problem where the sensitivity to the specific values of the controls are low even if the sensitivity to the average values of the controls are not. In other words, any model that acts to filter the effect of active controls will have this problem, so any problem where the effect of a control is integrated will have this problem to some degree. Many optimal control problems have this characteristic, so if a large number of stages are needed to resolve a control profile accurately, this will generally be a problem with IDP.

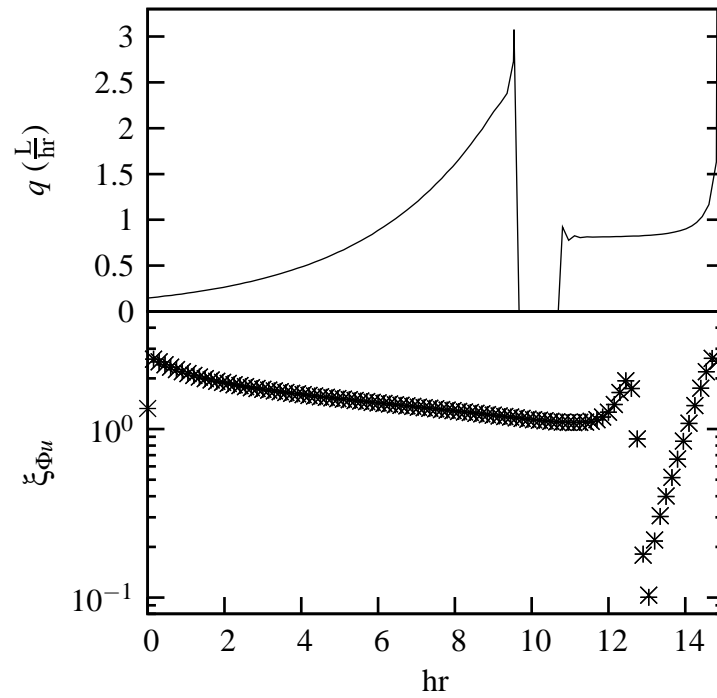


Figure 4.8: Optimal substrate feed (top) and normalized performance index sensitivity (bottom) for the Park-Ramirez optimal control problem. The + and × symbols are the forward and backward difference sensitivities respectively.

Chapter 5

Iterative dynamic programming control smoothing

This study proposes a two-step approach, introduced in Section 3.7, to solve the optimal control problem as rapidly and robustly as possible, where the problem is first solved with the basic IDP algorithm using adaptive control region size to a certain point, then solved the rest of the way using IDP with control smoothing techniques. This chapter discusses the second step in this process by describing several methods for IDP control smoothing and using them during the second step of the two-step method for the Park-Ramirez optimal control problem. A summary of these methods and recommendations for their use in the context of the whole process is given in Chapter 8. Stagewise linear controls are also discussed here since they can obtain a smoother control profile for a given number of stages.

5.1 The active control problem

For certain problems, IDP tends to find highly active control policies unless a large number of iterations is used. As more stages are used, this problem becomes even more severe. An examination of this problem in Section 5.1.1 will be followed by an evaluation of techniques designed to address it: a control filtering algorithm in Section 5.1.3, a new simulated annealing control filter in Section 5.1.4, control damping using modifications to the performance index function in Section 5.1.5, and a new kind of IDP test control in Section 5.1.6. Stagewise linear controls in Section 5.2 can also decrease

the average deviation of the calculated optimal control from the true optimal control, and can also decrease the active controls problem indirectly by requiring fewer stages for a given control resolution.

5.1.1 Examination of the active control problem

The objective of the optimal control problem is to minimize the performance index (Equation 3.2). The basic IDP algorithm always chooses controls that decrease the performance index at every decision, which can sometimes lead to situations where the performance index quickly approaches the optimal value but the controls at each stage are not similarly close to their optimal value until many orders of magnitude more computation. The short explanation for this behavior is that the performance index sensitivity for any given control at a stage may be very low, and thus its precise value doesn't affect the performance index much. Because IDP is a stochastic algorithm that tries controls randomly across the control region, this problem affects it much more severely than gradient algorithms, which normally try nearby controls.

Consider the simple fed-batch bioreactor described by

$$\begin{aligned}\dot{X} &= g(S) - \frac{q}{V}X \\ \dot{S} &= -\frac{1}{Y}g(S) + \frac{q}{V}(S_f - S) \\ \dot{V} &= q\end{aligned}\tag{5.1}$$

where X is cell concentration, $g(S)$ is a growth rate function with a maximum at some concentration of substrate S^* , q is the substrate feed rate, S_f is the substrate feed concentration, Y is the growth yield on substrate, and V is the system volume.

If the objective is to maximize final cell concentration, the intuitively obvious solution is to control q so that $S = S^*$ at all times in order to maximize $g(S)$ at all times and thus maximum final cell concentration.

To discover why IDP has trouble with active controls in this case, suppose that Figure 5.1 is part of the substrate state and substrate feed control profiles when the performance index is close to the optimum. The final stage is represented by k_f , and q^* is the optimal control. The control values at stages $k_f - 2$ and k_f are higher than optimal, and at stages $k_f - 3$ and $k_f - 1$ are lower than optimal.

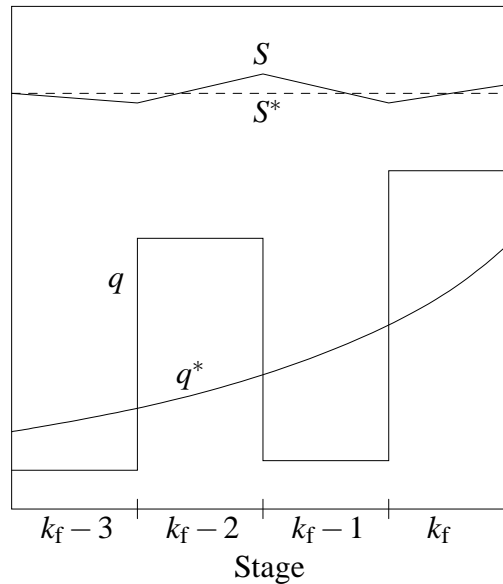


Figure 5.1: State and control for intermediate IDP solution of a simple bioreactor problem.

First consider how the IDP algorithm could improve the control for the final stage k_f for the case of a single state grid. Because the final performance index of the last stage is only affected by the choice of control for the last stage, any change must decrease the overall deviation of S from S^* over stage k_f . However, it is clear that any significant decrease in the control for the last stage will increase this deviation and will not be chosen.

IDP works backwards from the final stage, so after choosing the new control at stage k_f , the control at stage $k_f - 1$ is found. Since the optimal for stage k_f was already chosen too high, the value of control for stage $k_f - 1$ will need to be too low in order to

keep from increasing the deviation of S from S^* .

In other words, because IDP only looks at one stage at a time, it is stuck for this type of model once it gets reasonably close to the optimal control. In practice, because the objective $S = S^*$ is not quite met, there is a small window of control change to work with, but it is obvious that it will take a very large number of iterations to smooth the control. This effect is the reason for the unexpected result reported by Tholudur and Ramirez (1997) that increasing the number of stages increases IDP's control activity. As the number of stages increases, the window of possible control change narrows, increasing the effect described above.

For the case of multiple state grids, the situation is more complex. In that case, state grids other than the first are generated with test controls instead of the previous stage optimal control, so it becomes possible that the state at stage k_f allows a control move towards the optimal at stage $k_f - 1$. However, the computational load of IDP is directly proportional to the number of state grids (Equation 4.2), so increasing the number of state grids is not a good solution to this problem.

5.1.2 Solving the Park-Ramirez optimal control problem using basic IDP

Figures 5.2 and 5.3 present the basic IDP solutions to the Park-Ramirez optimal control problem for several combinations of stages and iterations. Each subfigure of Figures 5.2 and 5.3 is a separate IDP solution; they do not show a progression of a single run through more and more iterations. All subfigures of all Park-Ramirez optimal control solution figures share the same x and y scaling throughout this study.

As expected from the discussions in Section 3.7 and 5.1.1, the basic IDP algorithm quickly finds a performance index within 1% of the true optimal, but with active controls when using 25 or more stages. Clearly a large amount of computation is needed to obtain the correct control trajectory for the Park-Ramirez bioreactor problem with the basic stagewise constant IDP algorithm.

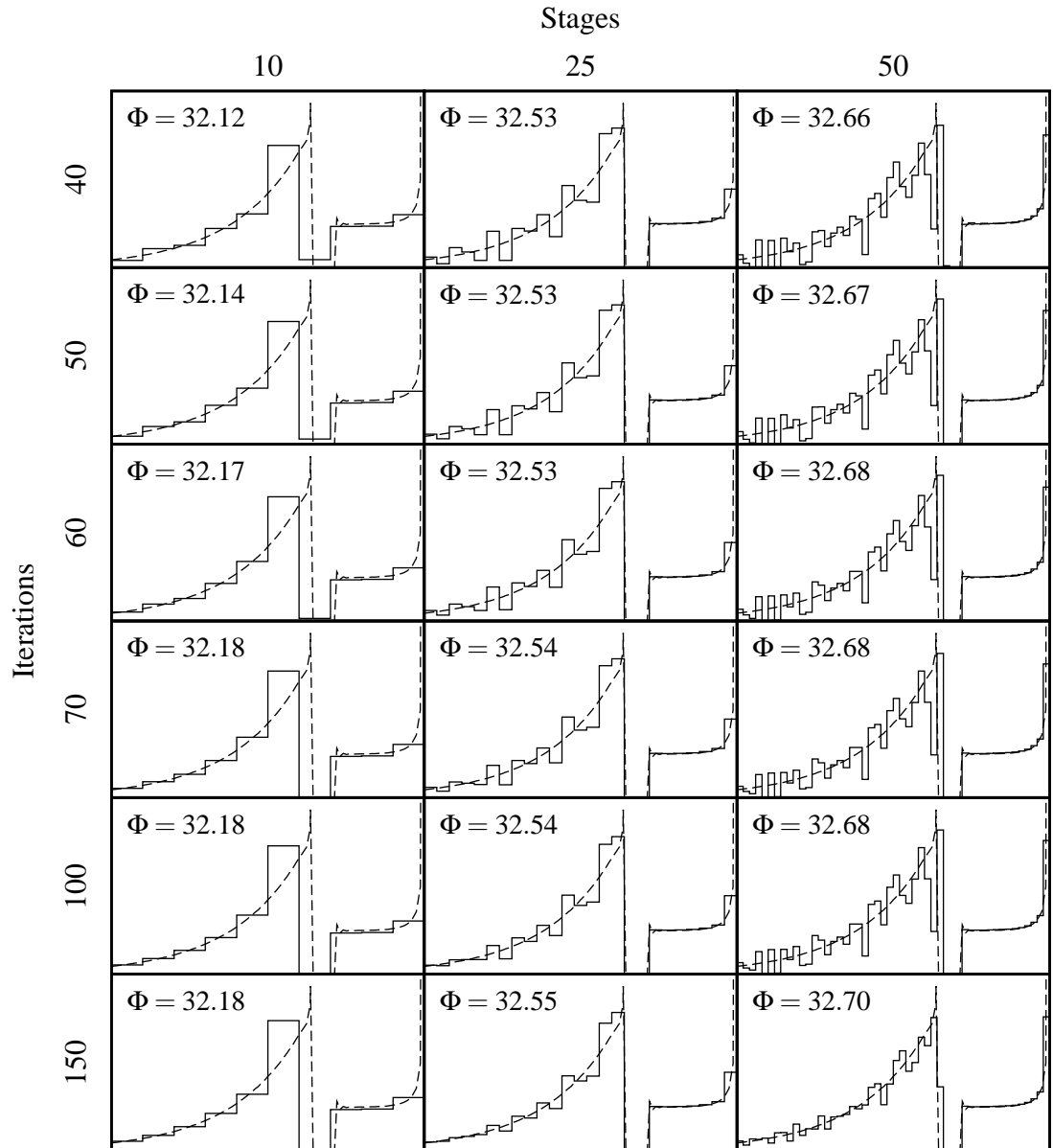


Figure 5.2: Solutions to the Park-Ramirez optimal control problem using the basic IDP algorithm. Table 4.3 (page 38) gives the IDP parameters used except for the number of iterations which is shown on this figure. The dashed line is the true solution to the problem with $\Phi = 32.76$. All subfigures share the same X and Y scale.

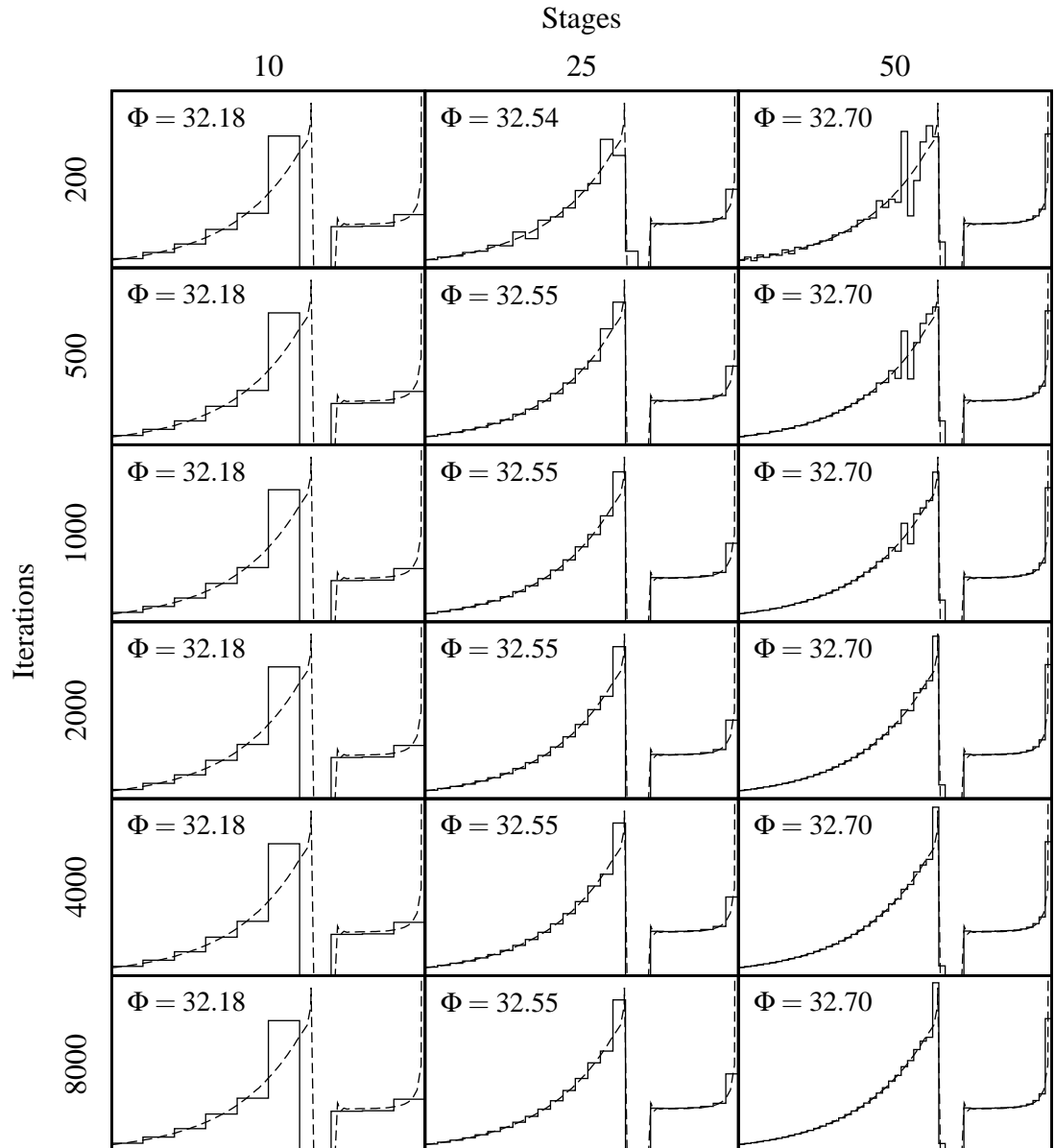


Figure 5.3: Solutions to the Park-Ramirez optimal control problem using the basic IDP algorithm. Table 4.3 (page 38) gives the IDP parameters used except for the number of iterations which is shown on this figure. The dashed line is the true solution to the problem with $\Phi = 32.76$. All subfigures share the same X and Y scale. An integrator relative tolerance of 10^{-6} was used for this figure.

Stages	Iterations	Equivalent integrations	Computer Time (sec)
10	60	1.4×10^4	3.5
25	1000	1.5×10^6	420
50	4000	2.4×10^7	3700

Table 5.1: Computational cost of the basic IDP algorithm to obtain a visually smooth control profile for the Park-Ramirez optimal control problem. Table 4.3 (page 38) gives the IDP parameters used except for the number of iterations which is shown on this table. The processor used was an AMD Athlon 2100+. Equivalent integrations were found using Equation 4.2.

Although all other solutions of the Park-Ramirez optimal control problem in this study use an integrator relative tolerance of 10^{-4} , a relative tolerance of 10^{-6} was required for the cases in Figure 5.3 to obtain smooth control profiles, especially with 50 stages. This is probably because the available control window for smoothing described in Section 5.1.1 was so small that a higher relative tolerance was required to find it, although this hypothesis was not tested.

Notice that the basic IDP algorithm with 10 stages finds a stable control profile with just 60 iterations, although the performance index is nearly 2% from the true optimal performance index (Figure 2.4), and the controls have a large average deviation from the true optimal control because of the coarse discretization. Table 5.1 displays the computational cost and actual computer time for the visually smooth solution for 10, 25, and 50 stages. This demonstrates again what a profound effect the number of stages has on the solution to the optimal control problem with IDP.

5.1.3 Control filtering

Tholudur and Ramirez (1997) proposed smoothing overactive controls by applying a first-order or median filter to the control policy after each IDP iteration. The general idea is that by forcing the control to a suboptimal but smoother policy after each iteration, the controls that are moved towards the optimal will stay there while the

controls that are moved away from the optimal by the filter will be moved back by the regular IDP algorithm.

The first-order filter for a single control is

$$u_k^{a+1} = \beta_f u_k^a + (1 - \beta_f) u_{k-1}^a \quad k = 2, \dots, N \quad (5.2)$$

where k is the stage index, u_k^{a+1} is the control for the next iteration at stage k , u_k^a is the control for the current iteration at stage k , and β_f is an adjustable tuning factor. If $\beta_f = 1.0$, the filter has no effect, and as the filter moves towards 0 the filter is more active. Tuning factor values in the range of 0.90 to 0.999 are typical.

The median filter replaces each control by the median of the surrounding F controls, where F is an adjustable filter window. The median filter is less effective than the first-order filter and is not considered further.

Figures 5.4, 5.5 and 5.6 show the effect of the first-order control filter on the Park-Ramirez optimal control solution for 10, 25, and 50 stages respectively. The filter works well, and is much better than unfiltered IDP at returning a smooth control profile. However, because the filter changes the control outside of the regular IDP algorithm and normally harms the performance index, there is a balance between too much and not enough filtering. The ideal selection of the first-order filter tuning parameter β_f depends on the problem, number of stages and number of iterations. The filter is applied every iteration so the more iterations, the more filtering. The filter parameter β_f should be adjusted along with the number of iterations or the expected number of iterations if using a stopping criterion other than a fixed number of iterations.

When first-order filtering is used throughout the IDP solution (without the two-step method), the case in Figure 5.6 with $\beta_f = 0.99$ reaches a local minima of $\Phi = 32.19$ even though the control profile appears similar to the other cases (results not shown). This problem with local minima was seen for all the smoothing techniques to some degree but it was worst with first-order control filtering and control damping. If a

solution is needed as quickly as possible with the least risk of finding local minima, the simulated annealing filter (Section 5.1.4) is a better choice.

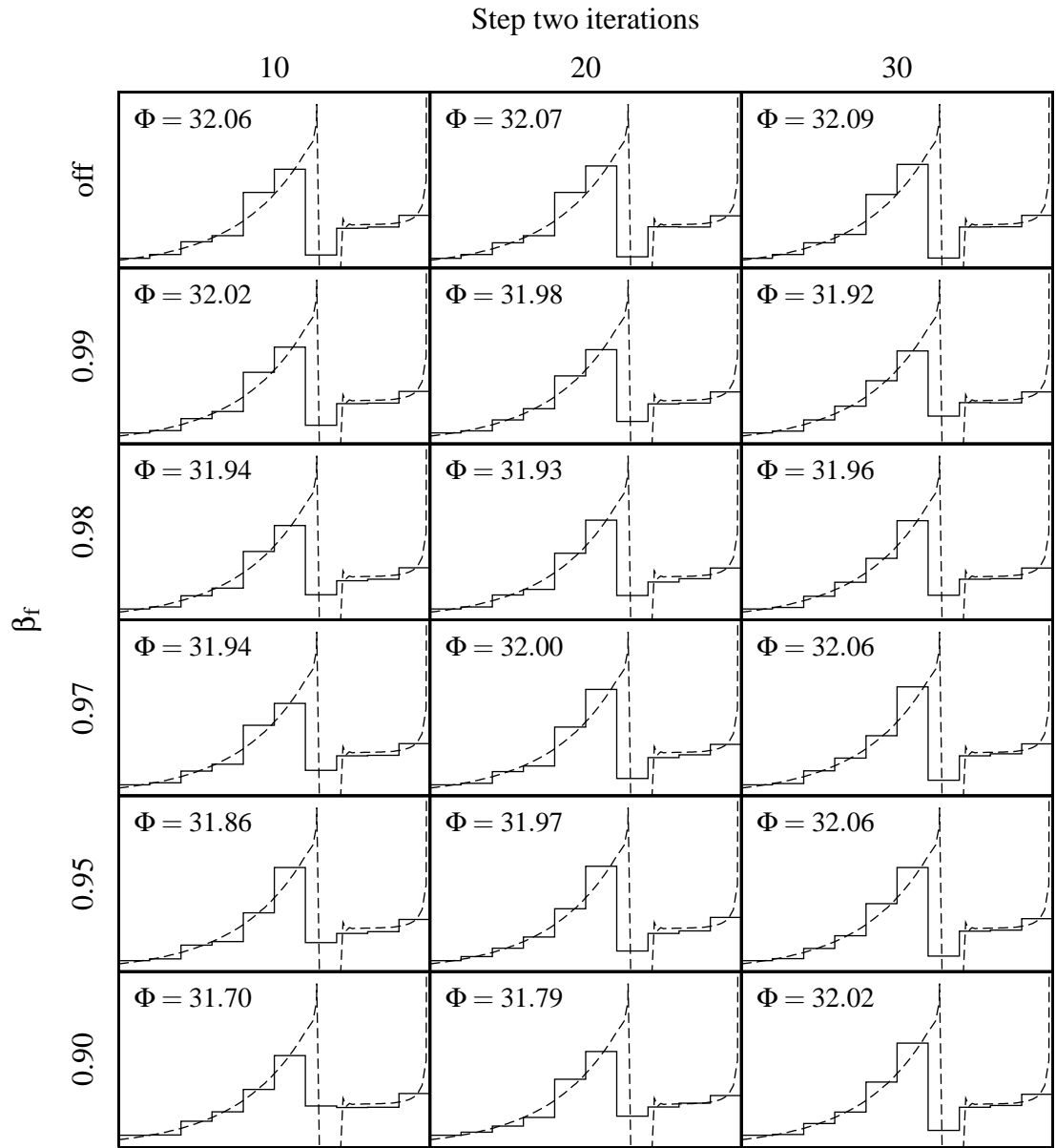


Figure 5.4: Two-step IDP solutions to the 10 stage Park-Ramirez optimal control problem using first-order control filtering during the second step. Table 4.3 (page 38) gives the IDP parameters for the first step. The IDP parameters for the second step were the same as the first except for using first-order control filtering with β_f shown on this figure and the number of iterations shown on this figure. The dashed line is the true solution to the problem with $\Phi = 32.76$. All subfigures share the same X and Y scale.

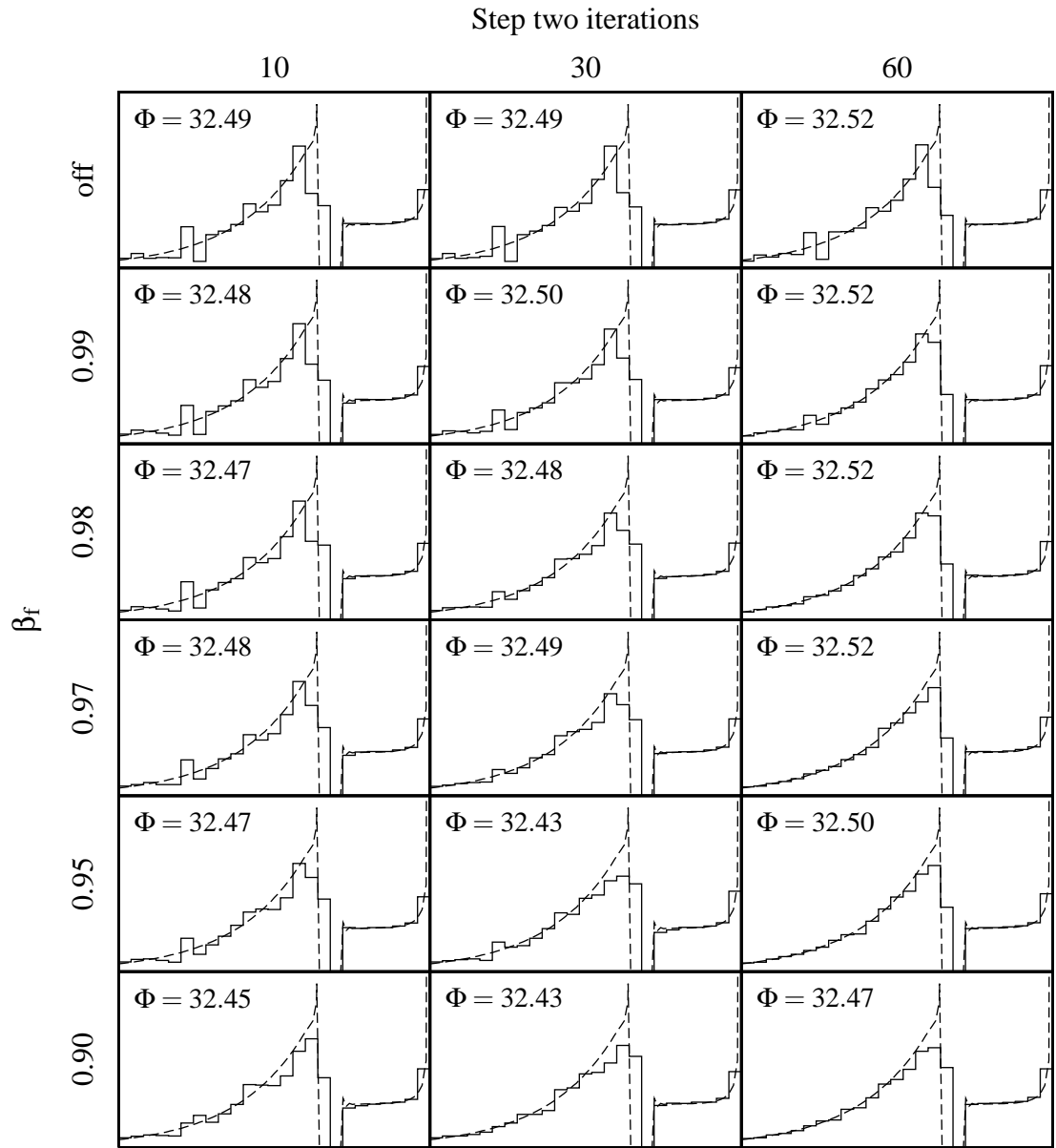


Figure 5.5: Two-step IDP solutions to the 25 stage Park-Ramirez optimal control problem using first-order control filtering during the second step. Table 4.3 (page 38) gives the IDP parameters for the first step. The IDP parameters for the second step were the same as the first except for using first-order control filtering with β_f shown on this figure and the number of iterations shown on this figure. The dashed line is the true solution to the problem with $\Phi = 32.76$. All subfigures share the same X and Y scale.

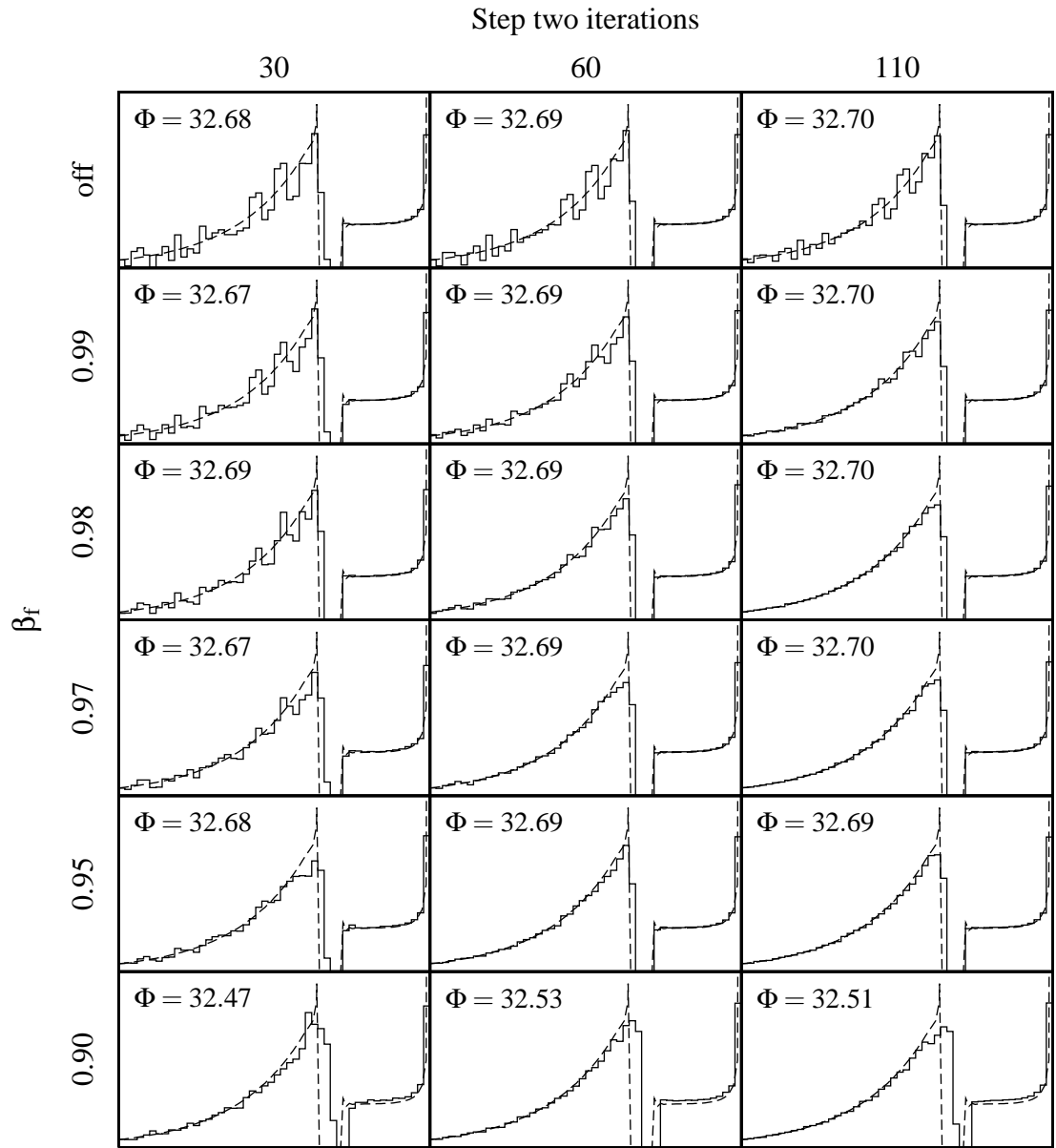


Figure 5.6: Two-step IDP solutions to the 50 stage Park-Ramirez optimal control problem using first-order control filtering during the second step. Table 4.3 (page 38) gives the IDP parameters for the first step. The IDP parameters for the second step were the same as the first except for using first-order control filtering with β_f shown on this figure and the number of iterations shown on this figure. The dashed line is the true solution to the problem with $\Phi = 32.76$. All subfigures share the same X and Y scale.

5.1.4 Simulated annealing control filtering

The IDP algorithm with filtering alternates between reducing the performance index and smoothing the control profile, which are often opposing goals. While working with the filtering technique, it was seen that sometimes parts of the control profile that were important and correct were changed when the filter factor was high enough to quickly get a smooth profile. The simulated annealing filter is an attempt to modify the effect of filter smoothing so that it smooths more vigorously where the negative effect on performance index is less. In keeping with the stochastic nature of IDP, it is also randomized, and the most obvious way to do that is with a simulated annealing type algorithm.

The simulated annealing filter is implemented by modifying the iteration algorithm at the point where the best test control is chosen (ie. Algorithm 5 step 11 for stagewise constant controls). The test control selection step is modified from “choose the test control with the best resulting performance index” to Algorithm 6. A test control is considered smoother in step 5 of Algorithm 6 if it is closer to the average of the controls at the stage before and after it. For the first or last stage, a test control is smoother if it is closer to a line drawn through the closest two interior points.

Algorithm 6 Test control selection for simulated annealing filter

1. **if** any test controls result in an improvement over the previous optimal performance index **then**
 2. choose the test control with the best performance index as normal
 3. **else**
 4. **for** each test control **do**
 5. **if** test control leads to a smoother control profile than the current optimal control **then**
 6. Assign this test control the selection index γ_s from Equation 5.3
 7. **end if**
 8. **end for**
 9. Choose the test control with the largest selection index
 10. **end if**
-

The equation for the selection index γ_s is

$$\gamma_s = \exp \left[\frac{-\Delta\Phi}{T \cdot M_\Phi} \right] - R(0, 1) \quad (5.3)$$

where $\Delta\Phi$ is the change in performance index for the test control, T is the temperature, M_Φ is a typical or maximum value of the performance index used to normalize the term, and $R(0, 1)$ is a random number between 0 and 1. M_Φ was chosen to be 30.0 for the Park-Ramirez optimal control problem as a typical value of the performance index seen during the course of the solution.

The temperature T cools with the control region size and iteration:

$$T = \left[\frac{T_i}{J} \sum_j^J \frac{r_j}{r_{j,0}} \right] \cdot \left[1 + \frac{a}{A} (\alpha_A - 1) \right] \quad (5.4)$$

where T_i is the initial temperature, j is each of J controls, r_j is the region size for control j averaged over all stages, $r_{j,0}$ is the initial region size for control j averaged over all stages, a is the current iteration index, A is the number of iterations, and α_A is the iteration cooling parameter. The first term in Equation 5.4 cools the temperature as the control region shrinks, and the second term cools the temperature as iterations progress. The second term in Equation 5.4 will equal α_A at the final iteration. For the Park-Ramirez optimal control problem α_A was chosen to be 0.5. This value is small enough that smoothing has lessened at the end of the solution to prevent it from picking a poor control during the final iterations, but large enough to converge as quickly as possible.

The first test control always contains the previous iteration optimal control, so $\Delta\Phi$ will be 0 and the exponential term in Equation 5.3 will be 1. If the temperature is very low, any test control which decreases the performance index will cause the exponential term to tend towards 0 and won't be selected. If the temperature is very high, the exponential term will tend towards 1 and the test control will be chosen randomly from those that smooth the control profile, regardless of the effect on performance index.

5.1.4.1 Simulated annealing filtering with the two-step method

Figures 5.7, 5.8 and 5.9 show the effect of the simulated annealing filter for 10, 25 and 50 stages respectively. An initial temperature in the range of 0.0005–0.001 worked best for all cases examined. Lower temperatures do not smooth the control profile, while higher temperatures lead to wild oscillation caused when poor choices made by the filter are over-corrected.

5.1.4.2 Simulated annealing filtering without the two-step method

The simulated annealing filter was the smoothing method least likely to find the most common local minimum for the 25 and 50 stage cases when used without the two-step procedure (results not shown). It is the best choice to find a solution quickly with a good performance index if control smoothness is not critical. The control will not be as smooth as with pivot point test controls (Section 5.1.6), but it seems to be much less likely to find local minima.

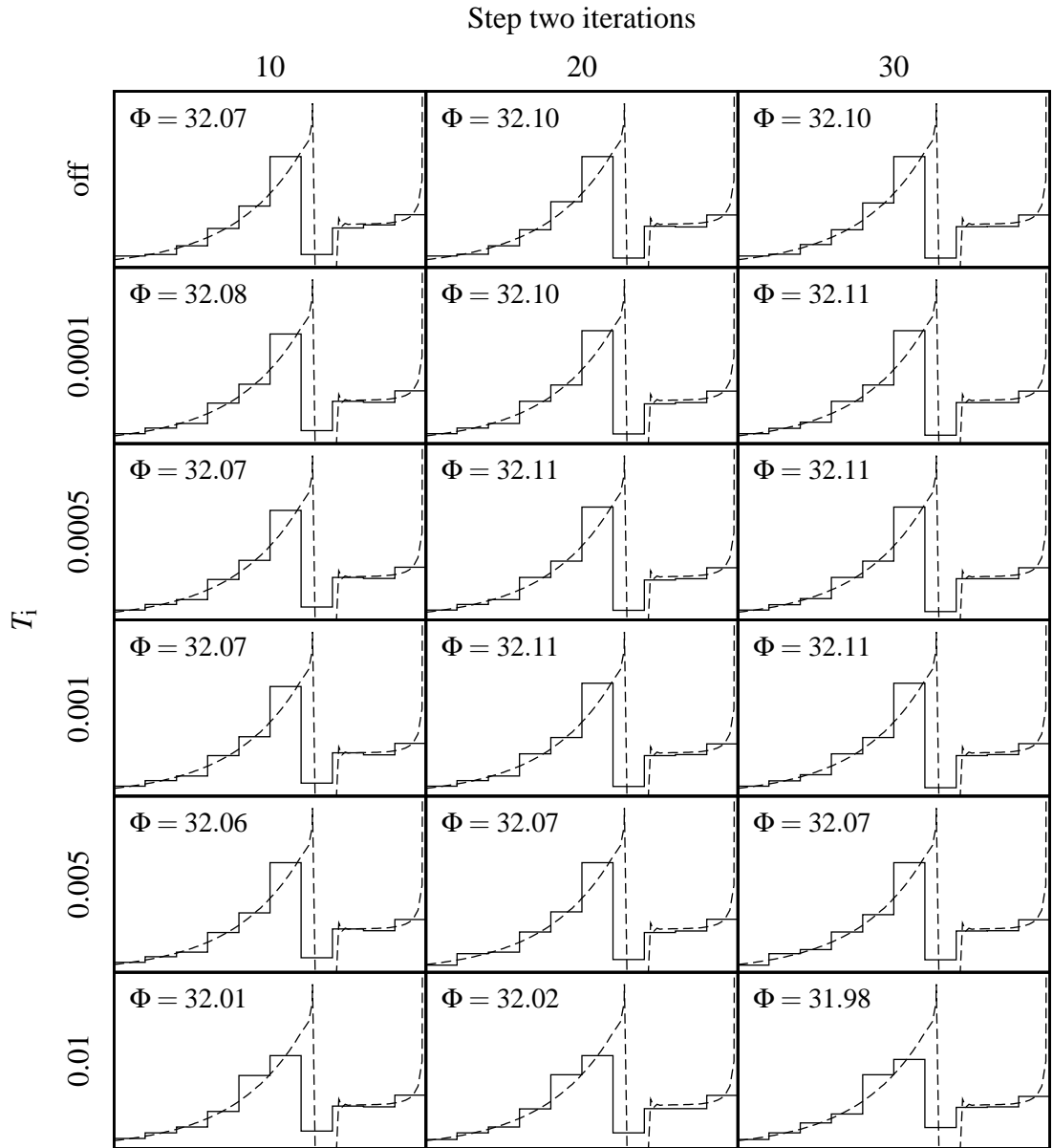


Figure 5.7: Two-step IDP solutions to the 10 stage Park-Ramirez optimal control problem using simulated annealing control filtering during the second step. Table 4.3 (page 38) gives the IDP parameters for the first step. The IDP parameters for the second step were the same as the first except for using simulated annealing control filtering with T_i shown on this figure and the number of iterations shown on this figure. The dashed line is the true solution to the problem with $\Phi = 32.76$. All subfigures share the same X and Y scale.

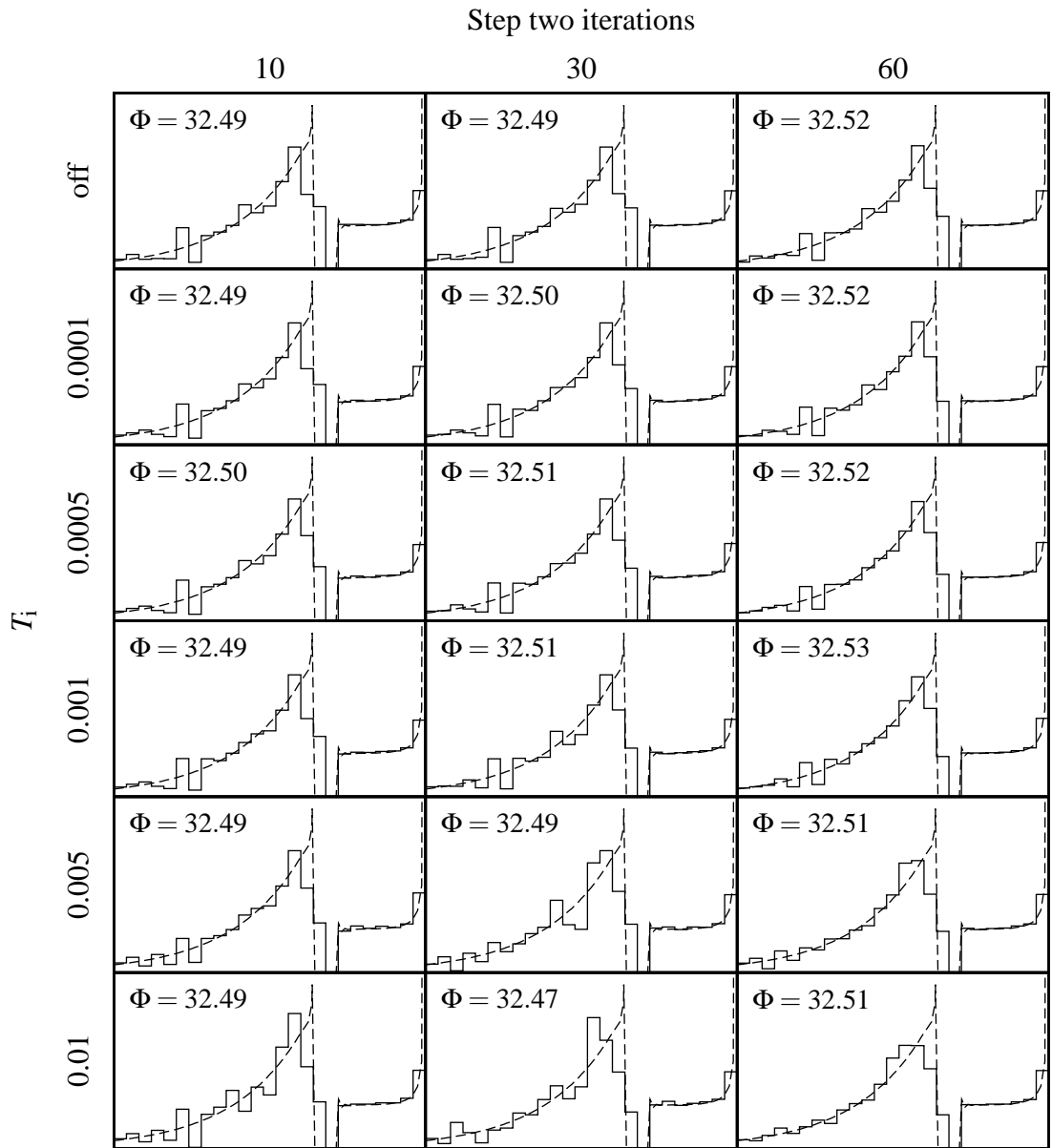


Figure 5.8: Two-step IDP solutions to the 25 stage Park-Ramirez optimal control problem using simulated annealing control filtering during the second step. Table 4.3 (page 38) gives the IDP parameters for the first step. The IDP parameters for the second step were the same as the first except for using simulated annealing control filtering with T_i shown on this figure and the number of iterations shown on this figure. The dashed line is the true solution to the problem with $\Phi = 32.76$. All subfigures share the same X and Y scale.

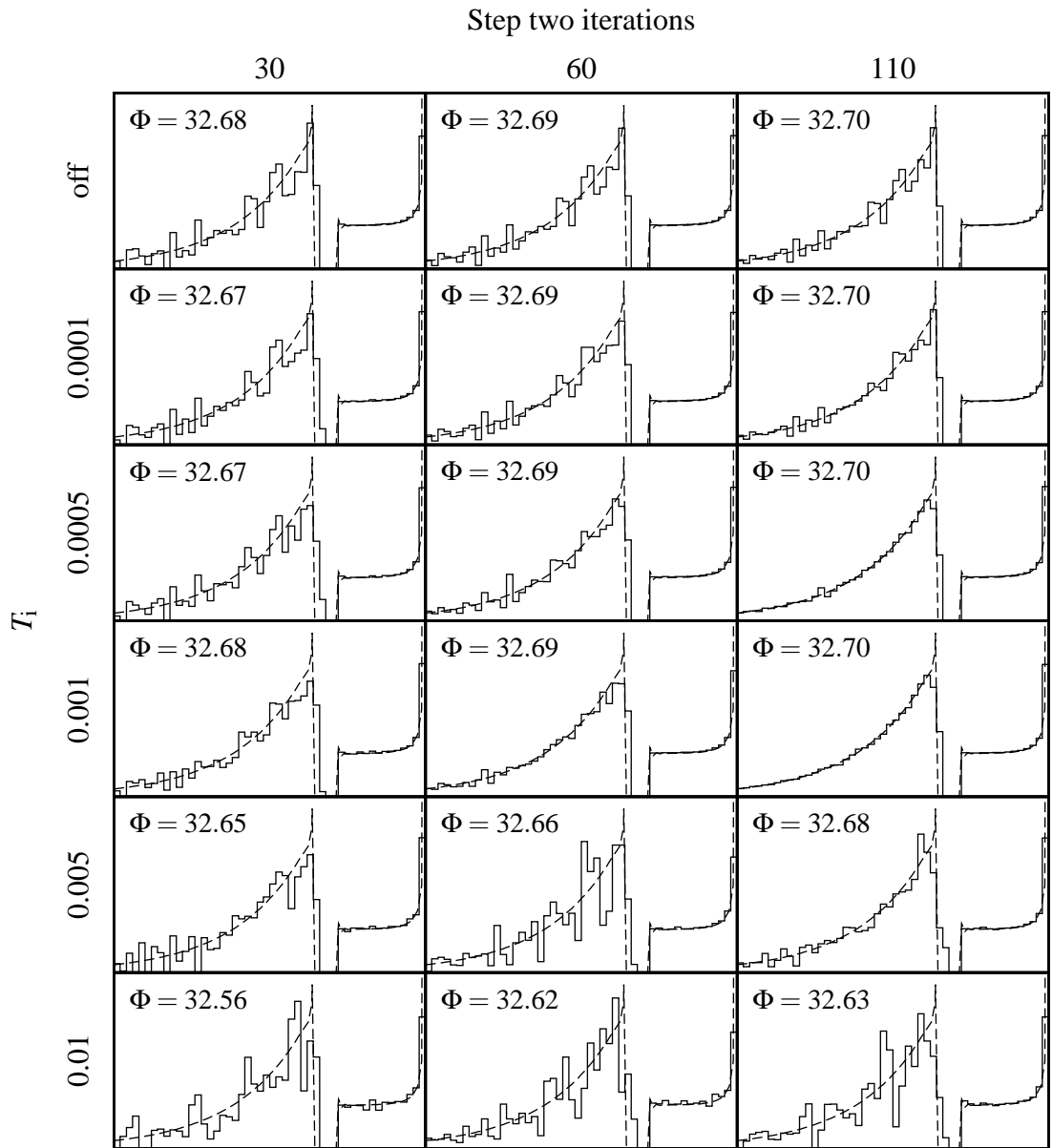


Figure 5.9: Two-step IDP solutions to the 50 stage Park-Ramirez optimal control problem using simulated annealing control filtering during the second step. Table 4.3 (page 38) gives the IDP parameters for the first step. The IDP parameters for the second step were the same as the first except for using simulated annealing control filtering with T_i shown on this figure and the number of iterations shown on this figure. The dashed line is the true solution to the problem with $\Phi = 32.76$. All subfigures share the same X and Y scale.

5.1.5 Control damping

5.1.5.1 Fixed control damping

Another way to reduce control activity is to punish it in the performance index function, by adding the control damping term Ψ_{ud} to the regular performance index function ψ in Equation 3.2. The control damping term used in this study is the normalized discrete second derivative

$$\Psi_{ud} = \frac{M_{\Phi}}{K} \sum_j^J \left[\frac{\beta_{u_j}}{M_{u_j}} \sum_{k=3}^K |u_{j,k} + u_{j,k-2} - 2u_{j,k-1}| \right] \quad (5.5)$$

where M_{Φ} is a typical or maximum value of the performance index, K is the number of stages, j is each of J controls, β_{u_j} is a non-negative damping factor for each control u_j , M_{u_j} is a typical or maximum value of control j , and $u_{j,k}$ is the value of control j at stage k . Larger values of β_{u_j} will lead to smoother control policies.

Because the control damping term changes the problem, it is best to use a small amount of damping. Much like the tolerances of an integrator should be tested by changing them to see if the solution changes, the control damping should be varied to make sure that it isn't changing the solution.

5.1.5.2 Relative control damping

A problem with the control damping technique described in the previous section is that because the damping term Ψ_{ud} is added to the performance index, the effect of the control damping factors β_{u_j} depend on the value of the performance index. The value of the performance index can be very different even for different runs of the same model due to different initial conditions, so unless M_{Φ} is updated for each new case, the damping will be inconsistent.

To deal with this problem relative damping is used, where the fixed damping factors β_{u_j} are adjusted after each iteration so that the control damping term Ψ_{ud} is

some fraction of the performance index. The first-order adjustment equation is

$$\beta_{u_j}^{a+1} = \beta_{u_j}^a + \frac{\beta_{u_j}^a}{\tau_{uA}} \left[\frac{\beta_{u\Phi} \cdot \Phi}{\Psi_{ud}} - 1 \right] \quad j = 1 \dots J \quad (5.6)$$

where $\beta_{u_j}^{a+1}$ is the value of β_{u_j} for the next iteration, $\beta_{u_j}^a$ is the value of β_{u_j} at the current iteration, τ_{uA} is the first-order time constant (in iterations), and $\beta_{u\Phi}$ is the desired value of Ψ_{ud} as a fraction of Φ . This formula requires initial values for all the β_{u_j} and keeps the ratio between them fixed. For this study $\tau_{uA} = 5$ and the initial β_{u_j} were 10^{-10} .

5.1.5.3 Control damping with the two-step procedure

Figures 5.10, 5.11 and 5.12 show the effect of relative control damping on the Park-Ramirez optimal control solution for 10, 25 and 50 stages respectively. This technique is effective for smoothing the Park-Ramirez optimal control problem using the two-step procedure. However, simulated annealing control filter and pivot point test controls lead to a smooth control profile with less rounding of sharp but correct control changes at discontinuities, so this technique is not a first choice for smoothing. It is recommended in combination with other methods with $\beta_{u\Phi} < 10^{-3}$.

5.1.5.4 Control damping without the two-step procedure

Control damping is not recommended above a minimal amount when used throughout the IDP solution (without the two-step procedure). During testing of these techniques using many different combinations of parameters, this technique often found local minima and poor control profiles when used with $\beta_{u\Phi} > 10^{-3}$, so it is not recommended above this amount. It is hypothesized that this is because large damping severely limits the potential control paths.

5.1.5.5 Control damping for subtle smoothing

Even if other techniques are used to address the active control problem, a small amount of damping may be useful if a smooth control policy is preferred over a non-smooth control policy. Figure 5.13 illustrates this effect. Both subfigures of Figure 5.13 have 50 stages, stagewise linear continuous controls and identical IDP parameters except $\beta_{u\Phi}$. They are both solved to convergence. The zigzag in the undamped case moves the transition from exponential to zero feed slightly forward in time. The performance index payoff for the more optimal transition time is higher than the cost for the suboptimal zigzag, so the algorithm chooses it. With a small amount of damping this balance is shifted, and the algorithm chooses to move the transition forward in time by reaching zero feed a stage later. The performance index (excluding the damping term) is only 0.01% worse for the damped case, but the control profile is clearly smoother. However, this only addresses the goal of control smoothness, not of control profile accuracy. The control profile in the damped case is probably no closer to the optimal control profile than the undamped case. Control damping in the range $10^{-4} < \beta_{u\Phi} < 10^{-3}$ seems useful for this kind of smoothing effect.

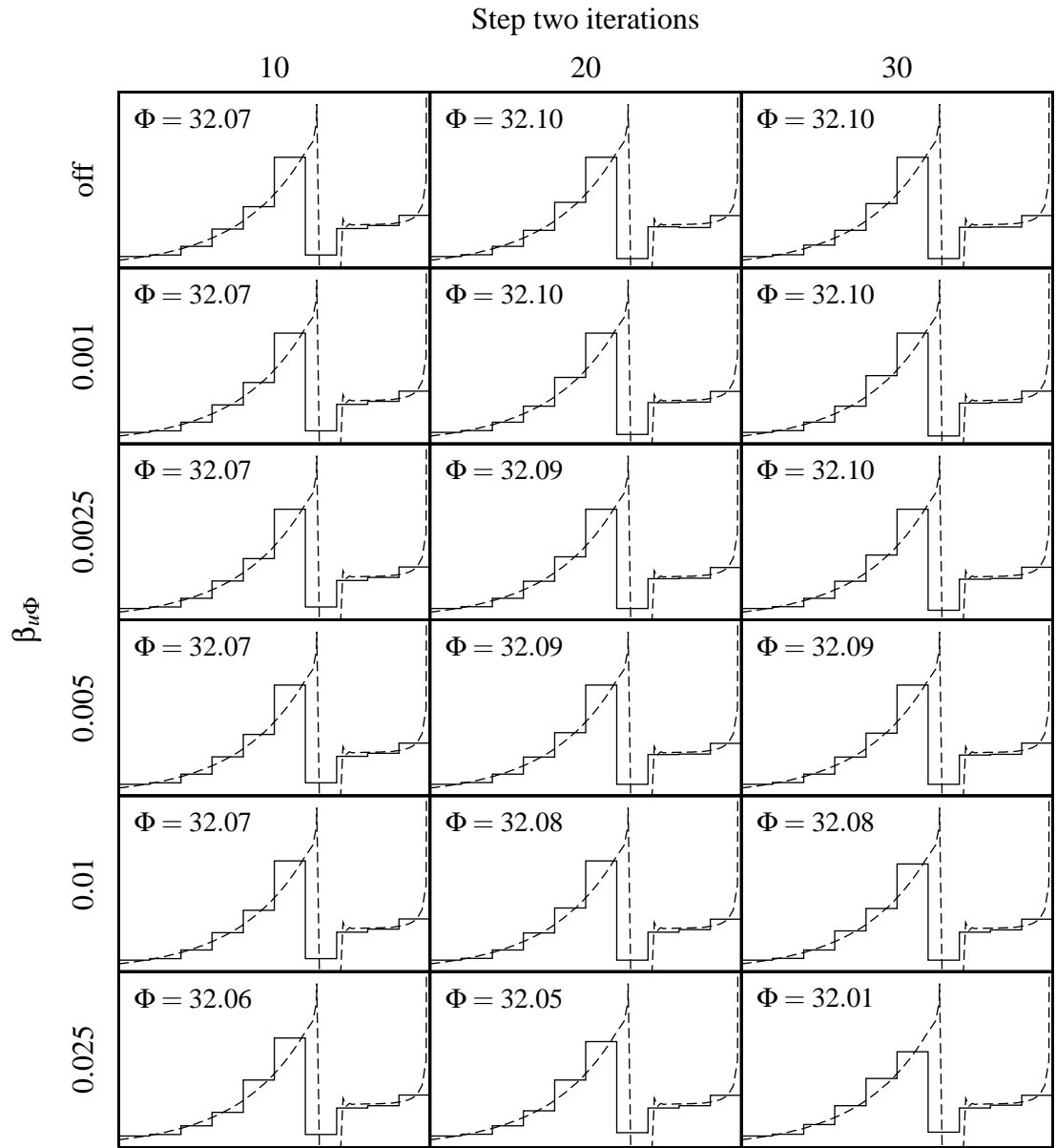


Figure 5.10: Two-step IDP solutions to the 10 stage Park-Ramirez optimal control problem using relative control damping during the second step. Table 4.3 (page 38) gives the IDP parameters for the first step. The IDP parameters for the second step were the same as the first except for using relative control damping with $\beta_{u\Phi}$ shown on this figure and the number of iterations shown on this figure. The dashed line is the true solution to the problem with $\Phi = 32.76$. All subfigures share the same X and Y scale.

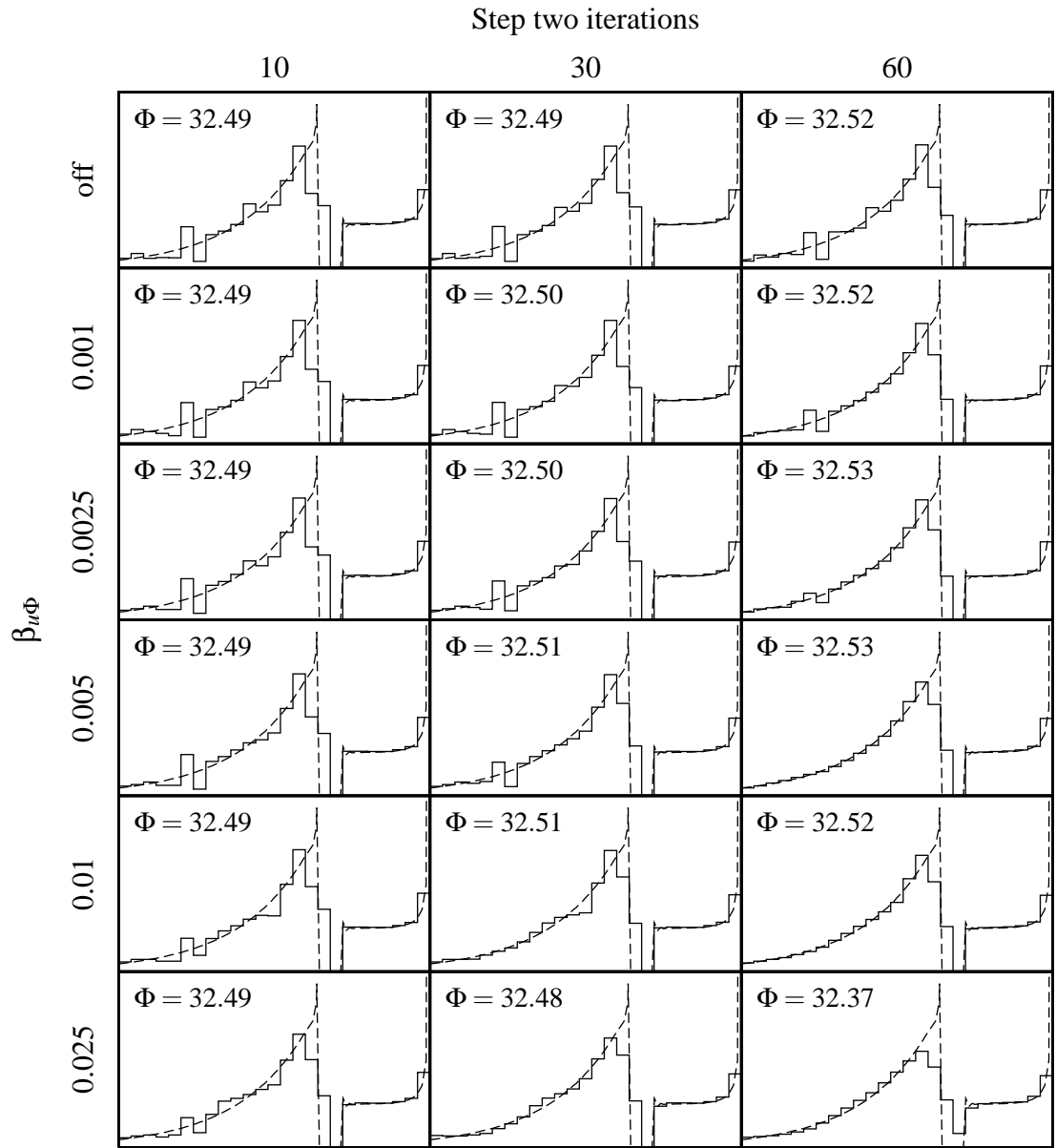


Figure 5.11: Two-step IDP solutions to the 25 stage Park-Ramirez optimal control problem using relative control damping during the second step. Table 4.3 (page 38) gives the IDP parameters for the first step. The IDP parameters for the second step were the same as the first except for using relative control damping with $\beta_{u\Phi}$ shown on this figure and the number of iterations shown on this figure. The dashed line is the true solution to the problem with $\Phi = 32.76$. All subfigures share the same X and Y scale.

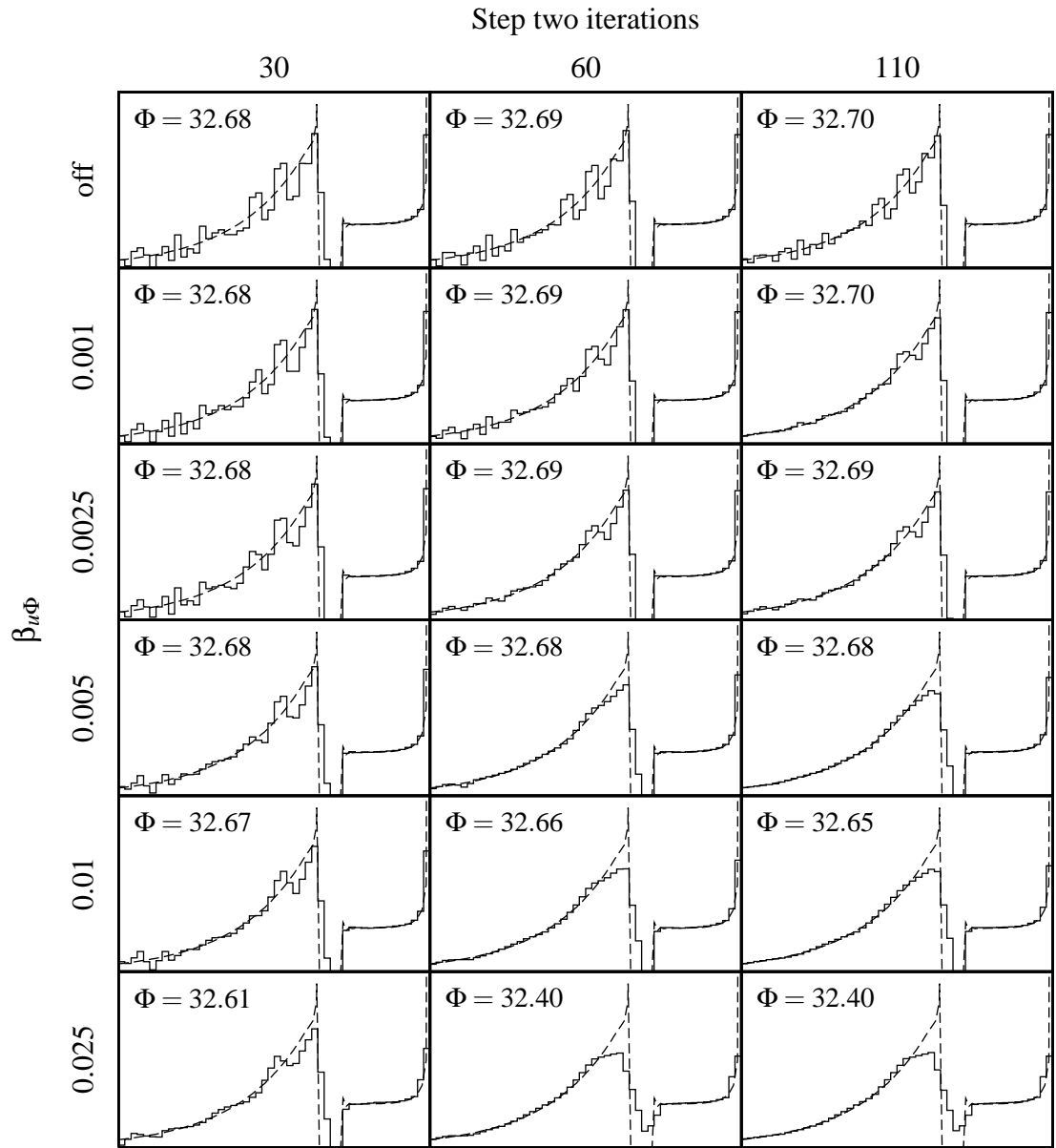


Figure 5.12: Two-step IDP solutions to the 50 stage Park-Ramirez optimal control problem using relative control damping during the second step. Table 4.3 (page 38) gives the IDP parameters for the first step. The IDP parameters for the second step were the same as the first except for using relative control damping with $\beta_{u\Phi}$ shown on this figure and the number of iterations shown on this figure. The dashed line is the true solution to the problem with $\Phi = 32.76$. All subfigures share the same X and Y scale.

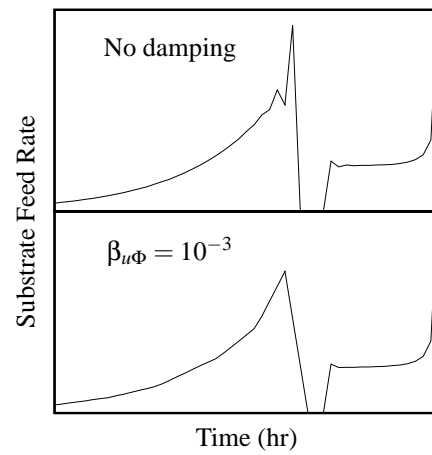


Figure 5.13: Example of the effect of minimal control damping on the Park-Ramirez optimal control problem with 50 stages and stagewise linear continuous controls. The damped case performance index is 0.01% worse than the undamped case.

5.1.6 Pivot point test controls

The examination of why IDP may have difficulty smoothing active control profiles in Section 5.1.1 led to an observation: once close to the optimal control, the **average** value of the control needs to remain approximately the same, but the **specific** values at a given stage need to change. A new kind of test control, called a “pivot point” test control, is added to the test control grid **TestControlGrid** in the IDP algorithm to accomplish this.

If a control is assumed to be near optimal and the average control trajectory is assumed to be correct, a natural way to try to improve the control is to increase the value of the control at the current stage while decreasing the value of the control at the next stage by the same amount (or the inverse). This will be called a “pivot point” control strategy in this study. For pivot point test controls, the control for the stage immediately following the test stage is derived from the test stage control instead of using the value in **OptimalControlGrid**. This control is changed the same amount from the optimal as the current test control under consideration, but in the opposite direction. For stages after that, the controls are found as normal from **OptimalControlGrid**.

Because the use of pivot point test controls assumes the average control trajectory is correct, control dependency is discounted and each control is tested independently. This is accomplished in the algorithm by giving each pivot point control in **TestControlGrid** $\frac{1}{J}$ odds (where J is the number of controls) of being active for any given stage and control. Inactive pivot point test controls use the previous iteration optimal from **OptimalControl**.

Half of the time, this strategy causes adjacent stage controls to move closer together, and half the time to move apart. This is a good thing because discontinuities can also become stuck in the same way as described for smooth controls.

It’s important that the pivot point strategy works within the IDP philosophy of

testing several alternative control strategies and remembering the one that reduces the performance index the most. Unlike control filtering, it doesn't force the controls into suboptimal trajectories, and unlike control damping, it doesn't change the problem by changing the performance index function. Finally, it is the only parameterless method of smoothing active controls with IDP. While the fraction of test controls that are pivot points can be considered a parameter, the choice of this value is not critical. If this value is chosen poorly, an increase in iterations will make up for the poor choice, unlike control filtering where increasing iterations can lead to over-filtering.

5.1.6.1 pivot point test controls with the two-step procedure

Figures 5.14, 5.15 and 5.16 show the effect of pivot point test controls on the Park-Ramirez optimal control solution for 10, 25 and 50 stages respectively. Pivot points converge to smooth trajectories for this problem more quickly than filters or damping, and without the rounding effect seen at the end of the exponential arc with damping. Approximately 50% pivot point test controls seems to be a good fraction, although there is clearly a large range where the technique works well.

5.1.6.2 pivot point test controls without the two-step procedure

When used without the two-step procedure, pivot point test controls converge the 25 stages case to a smooth control profile with $\Phi = 32.50$ in just over 20 iterations and the 50 stages case to a smooth control profile with $\Phi = 32.55$ in just over 60 iterations (results not shown). Unfortunately, pivot point test controls converge so quickly that they can lead to local minima. The two-step method was originally devised to address this problem with pivot point test controls and is especially successful with them because they always improve the original performance index in Equation 3.2, unlike all other smoothing methods in this study. Because the performance index always improves, pivot point test controls used with the two-step method will never find any

local minima that the basic IDP algorithm has passed in the first step of the two-step procedure.

Another interesting effect seen when pivot point test controls are used without the two-step procedure is that the solution may go through extremely active control trajectories in early iterations when the control region size is large, sometimes even alternating values to the control limits. This is possible because the algorithm always finds a control policy with a lower performance index, and with this particular problem high control activity doesn't hurt the performance index much.

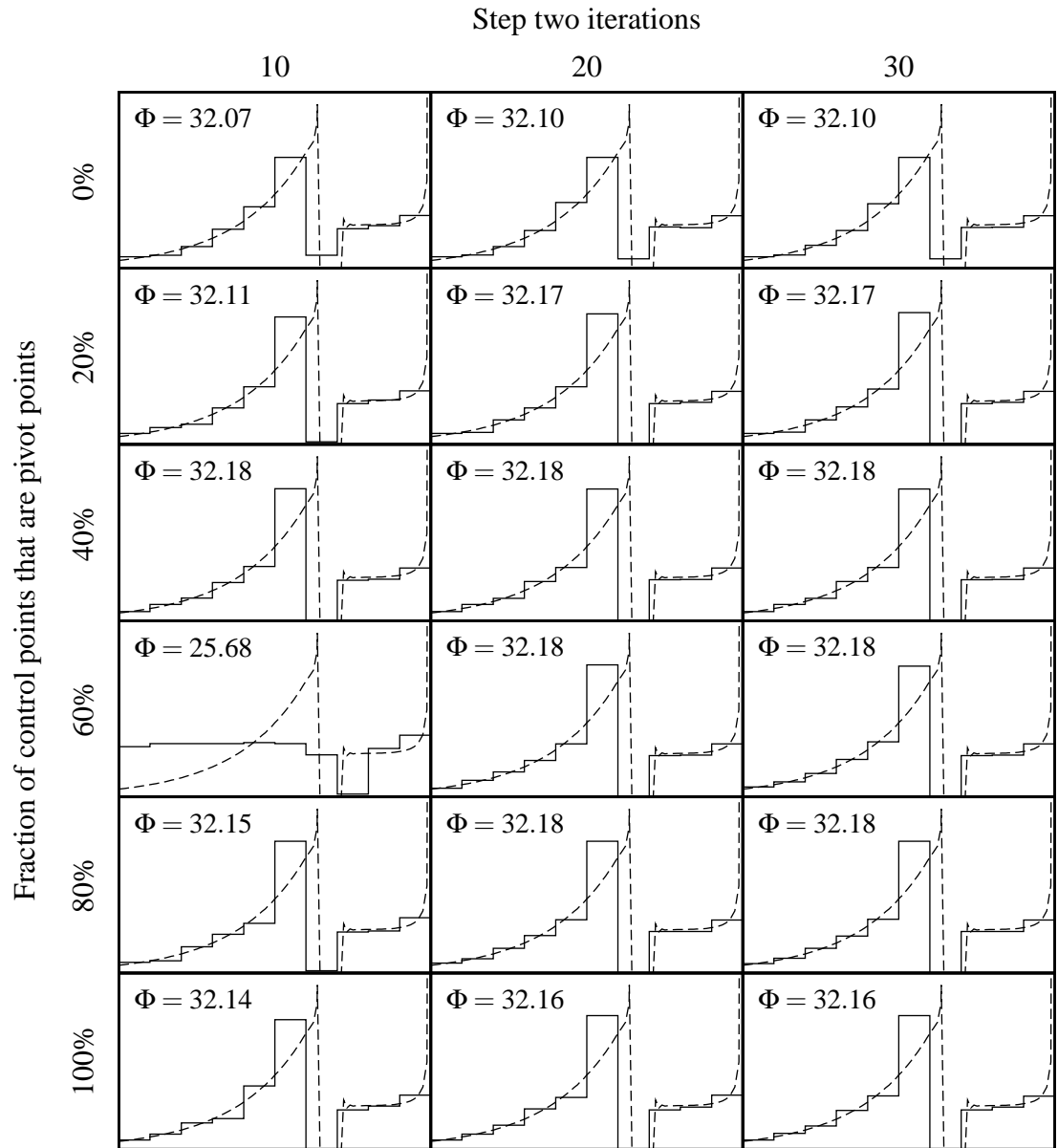


Figure 5.14: Two-step IDP solutions to the 10 stage Park-Ramirez optimal control problem using pivot point test controls during the second step. Table 4.3 (page 38) gives the IDP parameters for the first step. The IDP parameters for the second step were the same as the first except for using pivot point test controls with fraction of test controls that are pivot point test controls shown on this figure and the number of iterations shown on this figure. The dashed line is the true solution to the problem with $\Phi = 32.76$. All subfigures share the same X and Y scale.

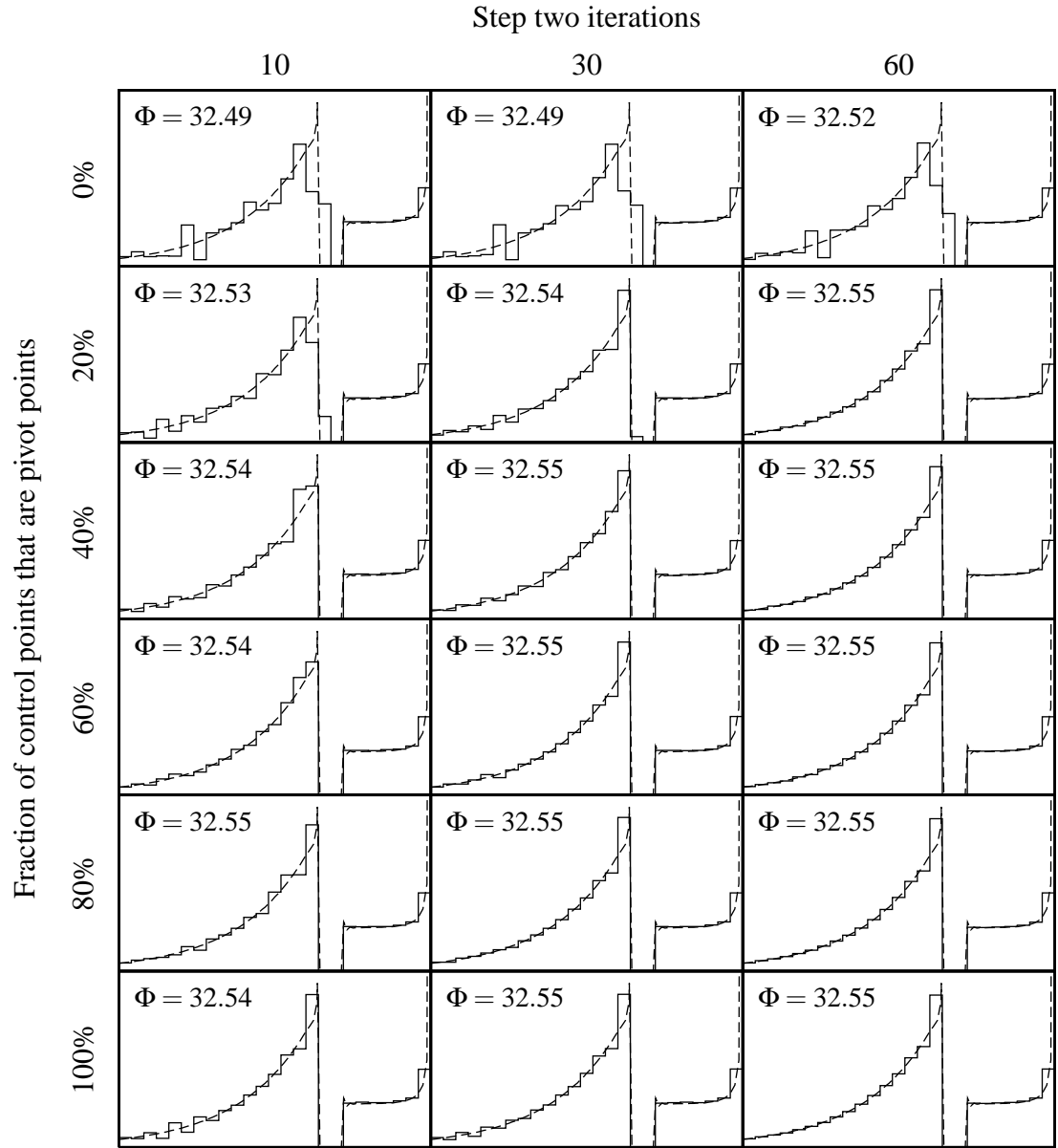


Figure 5.15: Two-step IDP solutions to the 25 stage Park-Ramirez optimal control problem using pivot point test controls during the second step. Table 4.3 (page 38) gives the IDP parameters for the first step. The IDP parameters for the second step were the same as the first except for using pivot point test controls with fraction of test controls that are pivot point test controls shown on this figure and the number of iterations shown on this figure. The dashed line is the true solution to the problem with $\Phi = 32.76$. All subfigures share the same X and Y scale.

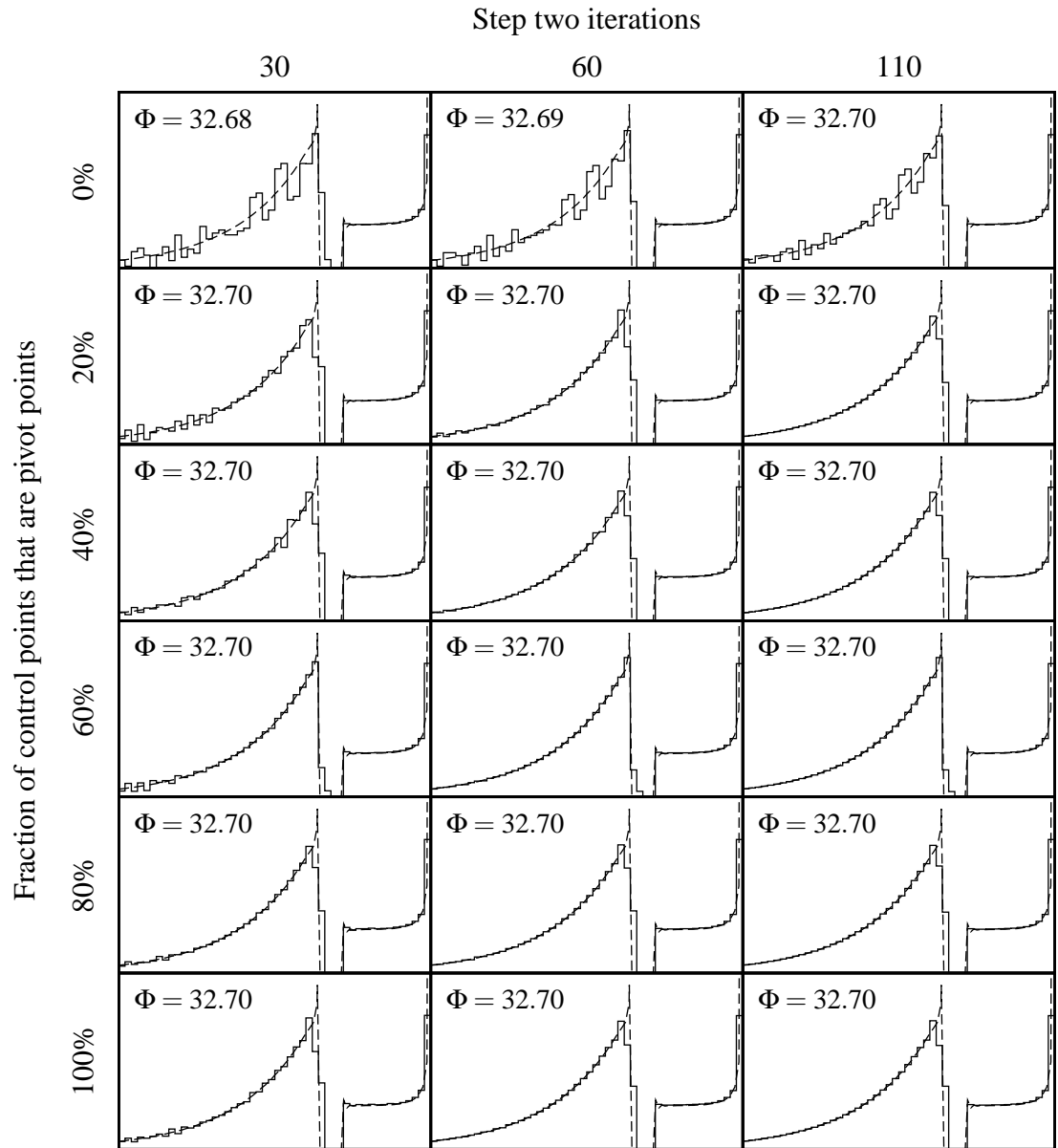


Figure 5.16: Two-step IDP solutions to the 50 stage Park-Ramirez optimal control problem using pivot point test controls during the second step. Table 4.3 (page 38) gives the IDP parameters for the first step. The IDP parameters for the second step were the same as the first except for using pivot point test controls with fraction of test controls that are pivot point test controls shown on this figure and the number of iterations shown on this figure. The dashed line is the true solution to the problem with $\Phi = 32.76$. All subfigures share the same X and Y scale.

5.1.7 Solo test controls

IDP random test controls choose random values for all controls. When there is more than one control, the curse of dimensionality means that a set of test controls is much less likely to improve the performance index as the number of controls increases. If the control profile is assumed to be close to the optimal control profile (as with step two of the two-step method), it may greatly increase convergence speed if only one control is tested at a time. “Solo” test controls choose a random test value for a single control, but use the previous iteration optimal control for the other controls. Which control will have a test value is chosen randomly. This method is not demonstrated for the Park-Ramirez optimal control problem since it only has one control, but the method was found to increase convergence speed for multi-control problems and was used in the hybrid identification problem in Chapter 6.

5.2 Stagewise linear continuous IDP

In addition to control filtering, control damping, and pivot point test controls, another potential way to decrease the average deviation of the calculated optimal control trajectory from the real optimal control trajectory is to use stagewise linear controls instead of stagewise constant controls. Luus (1993) presented an algorithm for stagewise linear continuous controls which is also detailed in Chapter 7 of his book *Iterative Dynamic Programming* (Luus, 2000). That algorithm requires only a minor change in the basic stagewise constant IDP algorithm, but has two deficiencies:

- The initial and final controls for the final stage are both chosen simultaneously, so the final stage is dimension $2J$ instead of dimension J for the rest of the stages. This problem becomes more severe as the number of controls J increases.
- Because the control for the start of the last stage is already chosen when the

next to last stage is being considered, there is no way to use more than one state grid.

Algorithm 7 is a replacement for Algorithm 5 that solves the two problems above and implements stagewise linear continuous controls for IDP. The main difference between this algorithm and the stagewise linear continuous control algorithm by Luus is that the first stage instead of the last stage has both initial and final controls chosen. This may seem like a trivial change, but in fact requires several changes to the algorithm and solves both the problems of the original algorithm. The code to implement this new algorithm is also slightly more complex than for the algorithm by Luus.

For stagewise linear control, there are two controls for each stage: one at the start of the stage and one at the end of the stage. In this new algorithm, the end of stage control is the control under decision, and the start of stage controls are obtained in several ways as detailed in the algorithm. **TestControlGrid**, **OptimalControlGrid** and **OptimalControl** contain start and end controls for stage 1, but only end controls for the remaining stages. With this scheme, all control decisions are of dimension J , and multiple state grids may be used.

Figure 5.17 presents solutions to the Park-Ramirez optimal control problem using stagewise continuous controls. The performance index value using stagewise linear continuous controls is roughly the same as with stagewise constant controls. With 10 stages, the exponential feed part of the control profile is much more accurate, but the transitions and zero-feed parts of the control profile are poor. The 25 and 50 stage cases illustrate that stagewise linear controls may still need a high number of stages to accurately match the optimal control profile, but the control profile accuracy is higher than with stagewise constant controls for the same number of stages. There was no downside to using this method for the two problems in this study, and it is used for all the remaining examples.

Algorithm 7 Iteration algorithm for stagewise linear continuous controls

1. Execute Algorithm 3 (**TestControlGrid** and **StateGrid** generation algorithm)
 2. **for** each stage backward from $K..1$ **do**
 3. **for** each of the S state grids **do**
 4. Choose the state condition at the start of the stage from **StateGrid**
 5. **for** each of the R test controls **do**
 6. Choose the start of stage controls from **TestControlGrid** using the test controls used to generate the current state grid
 7. Choose the end of stage controls from **TestControlGrid** for the current test controls
 8. Integrate the test stage
 9. **for** each of the remaining stages **do**
 10. Choose the start of stage controls to be the end side controls from the previous stage
 11. Choose the state grid in **StateGrid** whose state is closest to the current system state
 12. Choose the end of stage controls from **OptimalControlGrid** corresponding to the state grid in the previous step
 13. Integrate to the next stage
 14. **end for**
 15. If performance index is the best of the test controls being evaluated, store test control in **OptimalControlGrid** for this stage and state grid
 16. **end for**
 17. **end for**
 18. **end for**
 19. **for** each of the R test controls {stage 1 start side controls} **do**
 20. Choose the start of stage controls from **TestControlGrid**
 21. Choose the end of stage controls from **OptimalControlGrid** corresponding to state grid 1
 22. Integrate to the next stage
 23. Integrate the remaining stages as in steps 9 – 14
 24. If performance index is the best of the test controls being evaluated, store test control in **OptimalControlGrid** for this stage and state grid
 25. **end for**
 26. Execute Algorithm 4 to extract **OptimalControl** from **OptimalControlGrid**
-

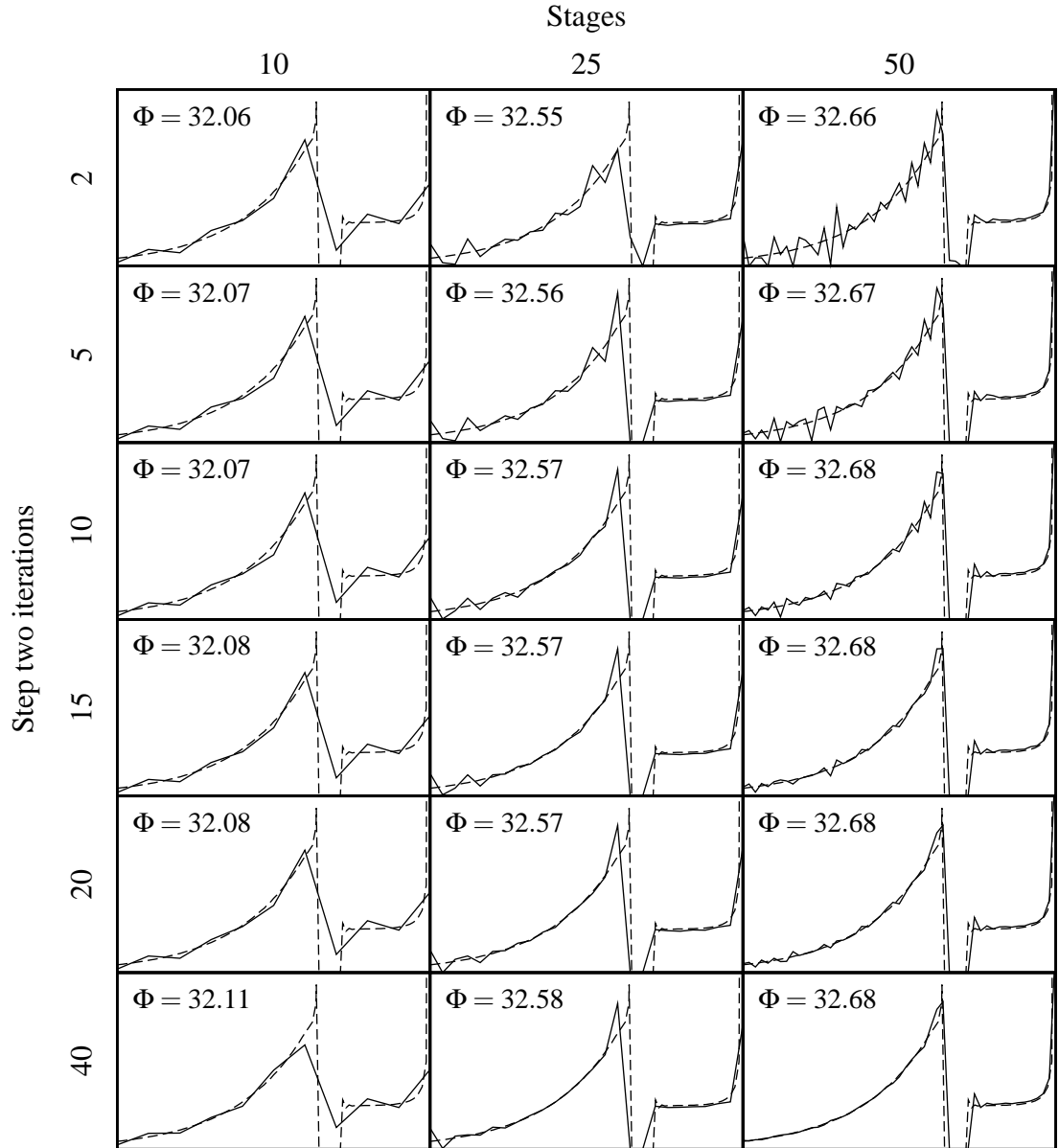


Figure 5.17: Two-step IDP solutions to the Park-Ramirez optimal control problem using stagewise linear continuous controls. Table 4.3 (page 38) gives the IDP parameters for the first step with the exception that stagewise linear continuous controls are used. The IDP parameters for the second step were the same as the first step except that 3 random test controls and 2 pivot point test controls were used, control damping was used with $\beta_{u\Phi} = 10^{-4}$, and the number of iterations is shown on this figure. The dashed line is the true solution to the problem with $\Phi = 32.76$. All subfigures share the same X and Y scale.

5.3 Stagewise linear discontinuous IDP

Section 5.2 presented a version of the single iteration algorithm using stagewise linear continuous controls. That algorithm requires a continuous control policy so that only one value of control is required per stage, as with the stagewise constant controls algorithm. If many discontinuities are expected it may still be an advantage to use stagewise linear controls instead of stagewise constant controls, but without the restriction that they be continuous.

Algorithm 8 is a replacement for Algorithm 5 that uses stagewise linear controls with discontinuities at the stage boundaries. This requires two values of control for each stage since there must be a starting and ending control for each stage, so **TestControlGrid**, **OptimalControlGrid** and **OptimalControl** contain start and end controls for all stages. Because there are two values of control for each stage, stagewise linear discontinuous problems have twice as many effective stages compared with a stagewise constant or stagewise linear continuous problem with the same number of actual stages.

Figure 5.18 presents solutions to the Park-Ramirez optimal control problem with stagewise linear discontinuous controls. Notice that the linear discontinuous case develops very active control policies compared with the linear continuous case, even with many more iterations. Part of the reason is that there are twice as many test controls for a given number of stages, but the effect is too large to be explained only by the increase in effective stages. Interestingly, the 25 stage linear discontinuous case has an equivalent control profile and higher performance index than the 50 stage linear continuous case in Figure 5.17, although the total number of iterations was significantly higher for the 25 stage linear discontinuous case.

The 50 stage linear discontinuous case is a very poor solution with poor performance index and control profile. This is caused by poor convergence during step one, even if step one iterations were increased to 180 iterations (results not shown). This al-

Algorithm 8 Iteration algorithm for stagewise linear discontinuous controls

1. Execute Algorithm 3 (**TestControlGrid** and **StateGrid** generation algorithm)
 2. **for** each stage backward from $K..1$ **do**
 3. **for** each of the S state grids **do**
 4. Choose the state condition at the start of the stage using **StateGrid** for this state grid
 5. **for** each of the R test controls {Find the best end side controls} **do**
 6. Choose the start of stage controls from **TestControlGrid** using the test controls used to generate the current state grid
 7. Choose the end of stage controls from **TestControlGrid** for the current test controls
 8. Integrate to the next stage
 9. **for** each of the remaining stages **do**
 10. Choose the state grid in **StateGrid** whose state is closest to the current system state
 11. Choose the start of stage and end side controls from **OptimalControlGrid** corresponding to the state grid in the previous step
 12. Integrate to the next stage
 13. **end for**
 14. If performance index is the best of the test controls being evaluated, store test control in **OptimalControlGrid** for this stage and state grid
 15. **end for**
 16. **for** each of the R test controls {Find the best start side controls} **do**
 17. Choose the start of stage controls from **TestControlGrid** for the current test controls
 18. Choose the end of stage controls from **OptimalControlGrid** corresponding to the current state grid
 19. Integrate to the next stage
 20. Integrate the remaining stages as in steps 9 – 13
 21. If performance index is the best of the test controls being evaluated, store test control in **OptimalControlGrid** for this stage and state grid
 22. **end for**
 23. **end for**
 24. **end for**
 25. Execute Algorithm 4 to extract **OptimalControl** from **OptimalControlGrid**
-

gorithm is not discussed further because the problems in this study do not have a large number of discontinuities.

Stage discretization	Stagewise linear discontinuous
Iterations	80
State grids	1
Random test controls	5
Adaptive control region method	Maximum Delta
Adaptive control region history	40 iterations
Adaptive control region k_r	1.5

Table 5.2: IDP parameters for step one of the two-step procedure for the stagewise linear discontinuous Park-Ramirez optimal control problem. Model parameters are in Table 2.1 (page 8).

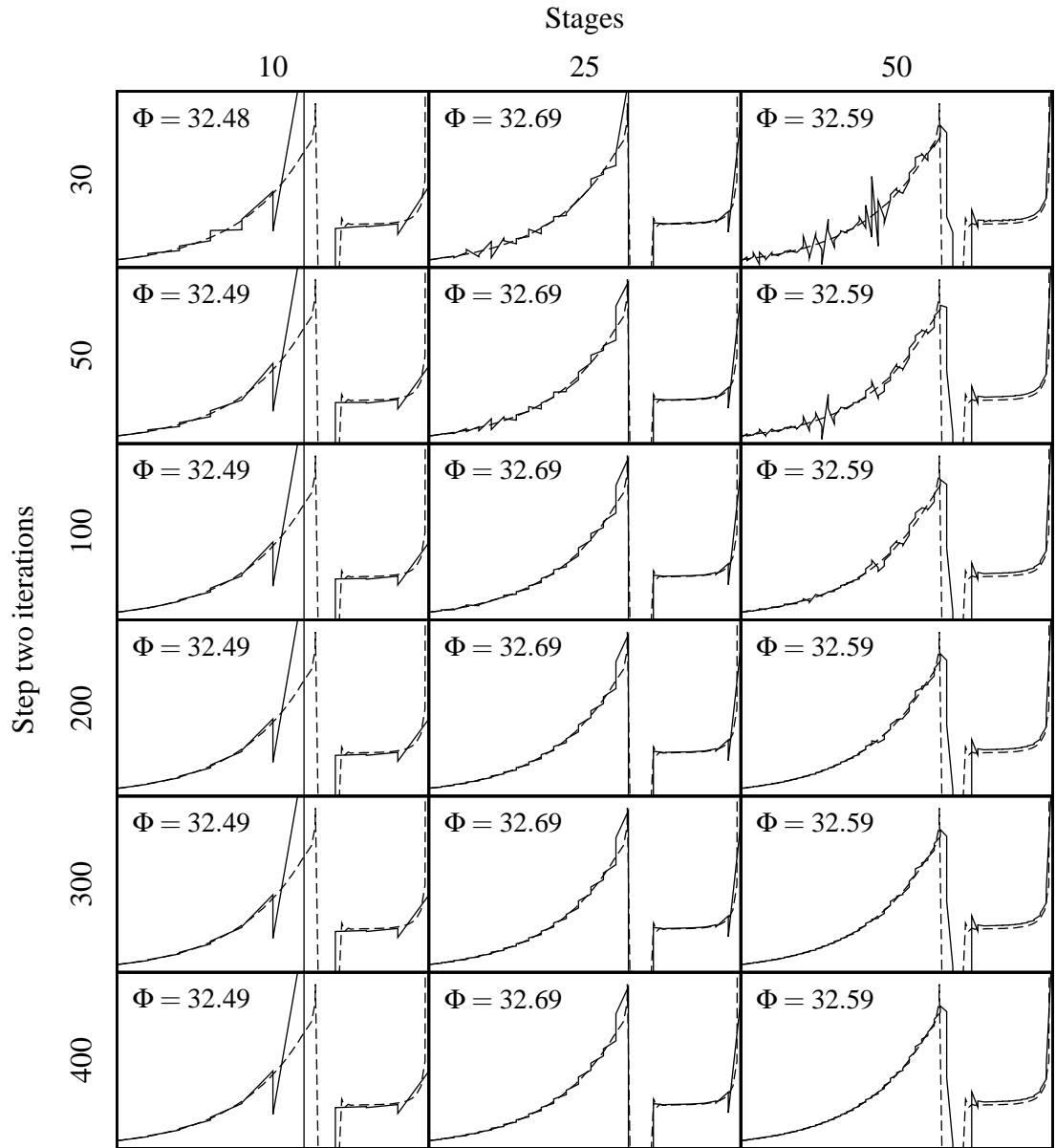


Figure 5.18: Two-step IDP solutions to the Park-Ramirez optimal control problem using stagewise linear discontinuous controls. Table 5.2 (page 80) gives the IDP parameters for the first step. The IDP parameters for the second step are the same as the first except that 3 random test controls and 2 pivot point test controls were used, control damping was used with $\beta_{u\Phi} = 10^{-4}$, and the number of iterations is shown on the figure. The dashed line is the true solution to the problem with $\Phi = 32.76$. All subfigures share the same X and Y scale.

5.4 Variable stage lengths

Bojkov and Luus (1996) introduced a modification to the basic IDP algorithm to allow variable stage lengths. This was accomplished by adding the stage length as an additional control and transforming the ODE equation so that a new adjusted time is used. For the fixed final time case, a punishment factor is also added to the performance index as a way to force the final time to be correct. Their technique allows variable stages with only one additional dimension added to the problem.

However, for some problems even a single additional dimension may be a burden. For this reason, a variation proposed here is to alternate an iteration improving the controls with an iteration improving the stage lengths. This approach doubles the number of iterations but does not increase the dimension of the optimal control problem. If stage lengths are found sequentially after the controls instead of simultaneously with the controls, the assumption must be made that the problem is smooth with respect to the interaction between the two. Also, the stage length iteration only uses a single state grid, since the assumption of problem smoothness in the stage length implies that a single state grid will be sufficient.

This study is concerned with obtaining the best performance index and most accurate control profile with the least amount of computation for fixed final time problems, so variable stages are only used during the second step in the two-step procedure. However, the cases presented in this section were all tried with variable stages during both steps of the two-step procedure and no difference was seen (results not shown). For minimal-time optimal control or other problems where the final time is not fixed, variable stages may need to be found simultaneously with controls for both steps of the two-step procedure since the stage length is part of the problem and not just a way to get a more accurate switching time for discontinuities.

The stage length region size used with variable stages can use adaptive region size

update analogous to the adaptive control region size update. For this study, all cases with variable stage lengths updated the stage length region size \mathbf{s} after each iteration using the Maximum Delta method of adaptive region size adjustment with 27 history iterations and $k_r = 1.0$.

One additional parameter needed for variable stage lengths is the stage length initial region size \mathbf{s}_i :

$$\mathbf{s}_i = \frac{\beta_s t_f}{K} \quad (5.7)$$

where β_s is the stage length initial region parameter, t_f is the model final time, and K is the number of stages. If $\beta_s = 1$ then \mathbf{s}_i will be the same as the initial value of the stage lengths, so reasonable values of β_s are between 0 and 1.

Figures 5.19 and 5.20 show variable stage length IDP solutions of the Park-Ramirez optimal control problem for 10 and 25 stages respectively. The reason to try variable stage lengths for this problem is to reduce the number of stages, so only the 10 and 25 stage cases were evaluated. Variable stage lengths should allow the stage boundaries to move to discontinuities and for stagewise linear continuous controls it should allow the length of the stages crossing the discontinuities to shrink. This can be seen clearly in Figure 5.20 with 60 iterations, where the performance index is significantly closer to the optimal performance index than with fixed stage length linear continuous controls with 50 stages (Figure 5.17 on page 77). The high performance index indicates that the switching time is found more accurately since the performance index is very sensitive to the discontinuity switching times. However, the control profile is not as close to the optimal control profile as the 50 stage case with fixed stage lengths and linear continuous controls. For this problem, variable stage lengths help achieve a higher performance index, but do not help reduce the number of stages if control profile accuracy is important. With 10 stages, there does not appear to be any difference between variable stage lengths and the comparable fixed stage length case in

Figure 5.17.

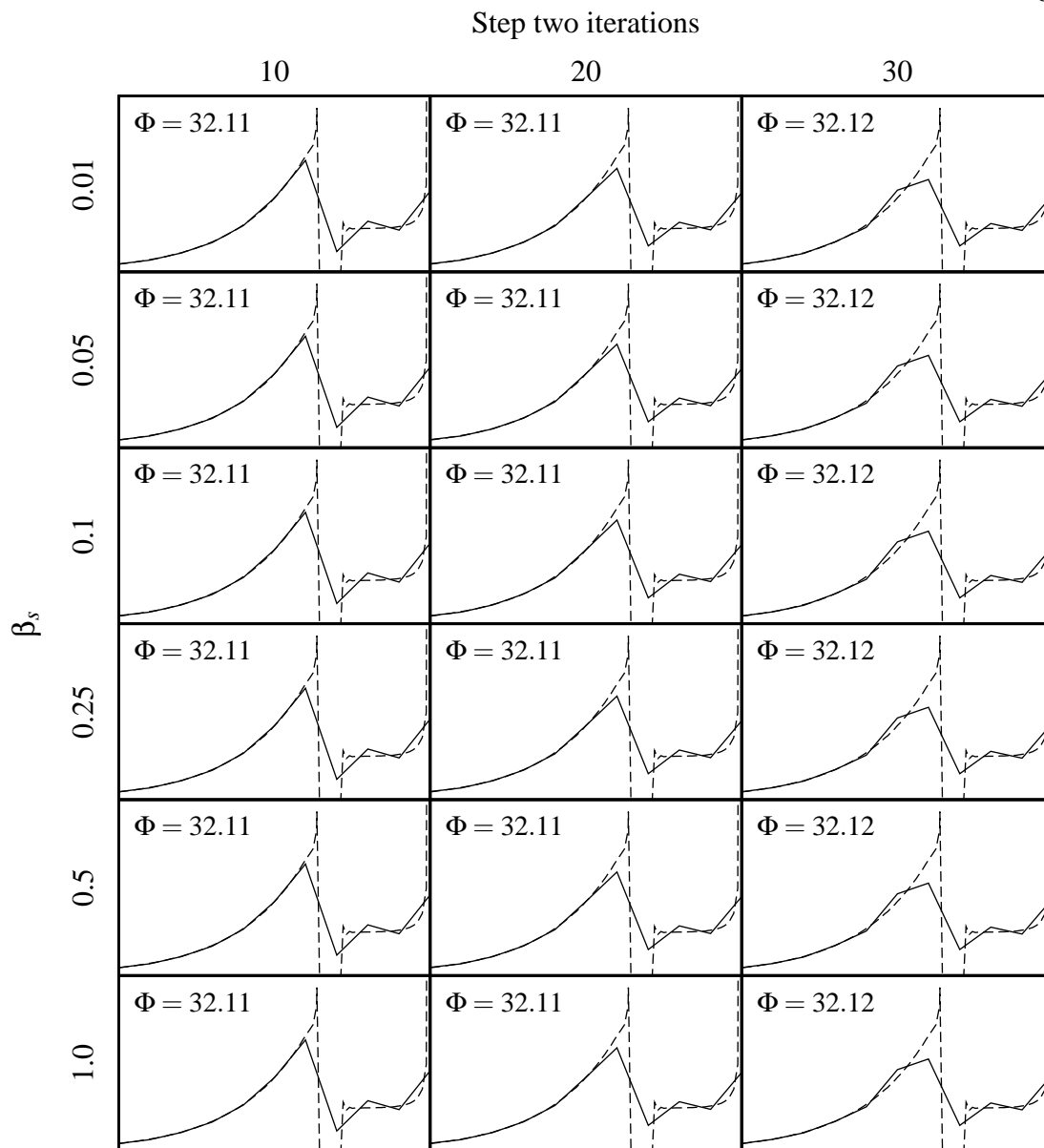


Figure 5.19: Two-step IDP solutions to the 10 stage Park-Ramirez optimal control problem using variable stage lengths solved sequentially after controls. Table 4.3 (page 38) gives the IDP parameters for the first step with the exception that stagewise linear continuous controls were used. The IDP parameters for the second step were the same as the first except that variable stage lengths solved sequentially after controls were used, 3 random test controls and 2 pivot point test controls were used, the number of iterations is shown on this figure, and the variable stage parameter β_s is shown on this figure. The dashed line is the true solution to the problem with $\Phi = 32.76$. All subfigures share the same X and Y scale.

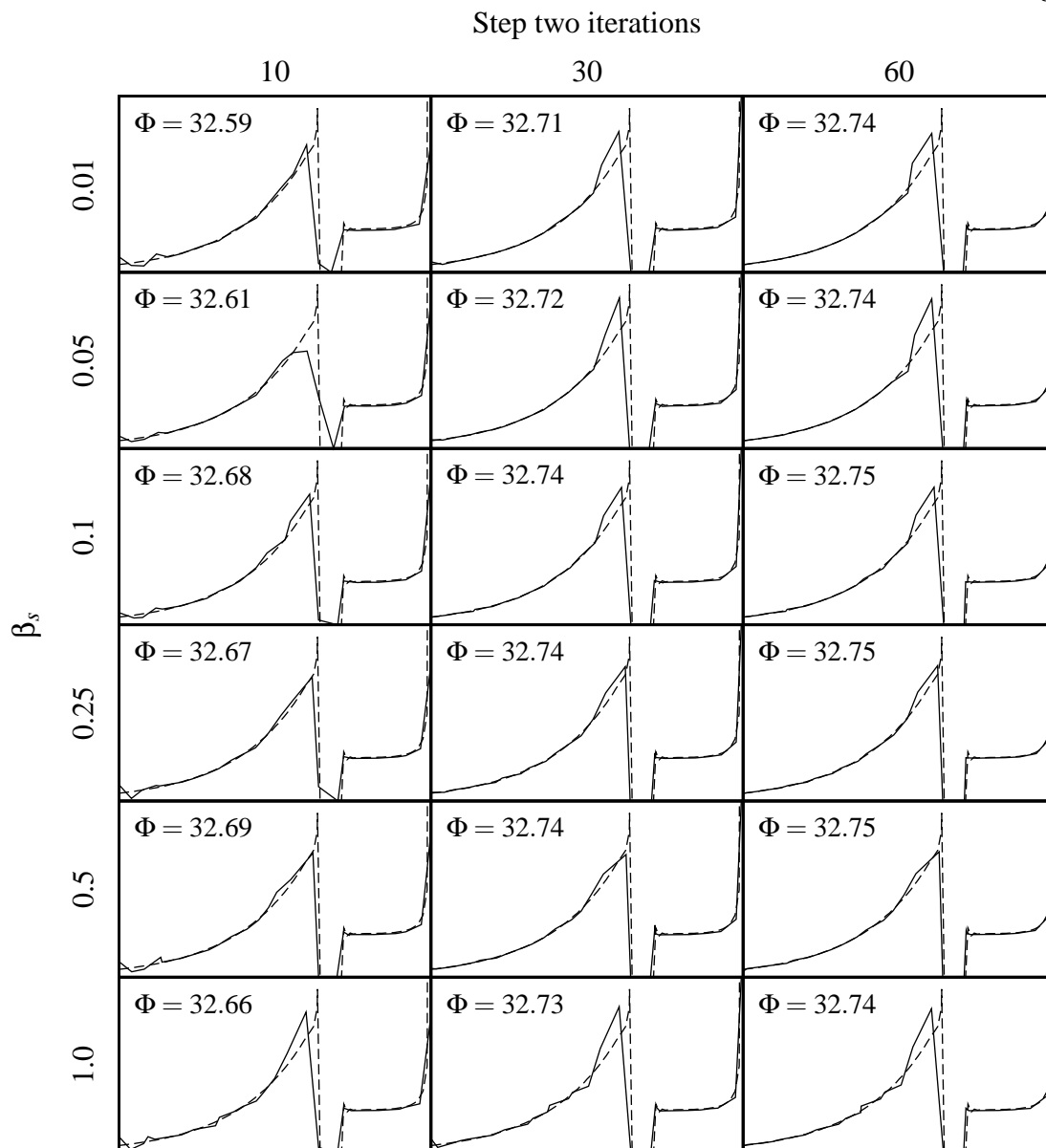


Figure 5.20: Two-step IDP solutions to the 25 stage Park-Ramirez optimal control problem using variable stage lengths solved sequentially after controls. Table 4.3 (page 38) gives the IDP parameters for the first step with the exception that stagewise linear continuous controls were used. The IDP parameters for the second step were the same as the first except that variable stage lengths solved sequentially after controls were used, 3 random test controls and 2 pivot point test controls were used, the number of iterations is shown on this figure, and the variable stage parameter β_s is shown on this figure. The dashed line is the true solution to the problem with $\Phi = 32.76$. All subfigures share the same X and Y scale.

5.5 Adaptive control region parameters for the two-step procedure

Section 4.5.2 examined the adaptive control region parameters for the first part of the two-step procedure, but it is also interesting to see their effect on the complete procedure. The goal of the two-step procedure is not only to obtain a performance index close to the optimal performance index, but also to also obtain a control profile that is close to the optimal control profile, so a simple test of performance index as in Section 4.5.2 is not sufficient. Instead, each test case was compared to a reference control profile generated using a conservative application of the two-step procedure with a fixed region size program. The reference control profile was generated using a first step of 4 passes of 40 iterations per pass with $\gamma_C = 0.95$ and $\gamma_R = 0.72$ and 5 random test controls. The second step was 6 passes of 50 iterations per pass with $\gamma_C = 0.95$, $\gamma_R = 0.8$, 3 random test controls and 2 pivot point test controls.

Each tested combination of adaptive region size method, A_H , and k_r resulted in a normalized sum squared error between the test case controls and the reference controls:

$$SSE_n = \frac{1}{J \cdot K} \sum_j \frac{1}{M_{u_j}^2} \sum_k (u_{j,k}^* - u_{j,k})^2 \quad (5.8)$$

where J is the number of controls, K is the number of stages, M_{u_j} is a typical or maximum value of control j used to normalize the term, $u_{j,k}^*$ is the value of reference control j at stage k , and $u_{j,k}$ is the value of test case control j at stage k .

Tables 5.3 and 5.4 show the two-step normalized sum squared error for 25 and 50 stages, respectively. For the 25 stage case, the error is generally lowest after 20 step two iterations using $A_H = 9$. This indicates that the control region for $A_H > 9$ is larger than needed, leading to more test controls which are not selected. On the other hand, the results for 20 step two iterations using $A_H = 9$ are the most variable, so there seems to be a tradeoff between efficiency and stability. For 60 step two iterations, there is very little difference among methods and parameters.

For the 50 stage case with 20 step two iterations a similar trend is seen for $A_H = 9$

where some values of k_r led to the lowest error but some led to higher error. The best results for the 50 stage case are with 60 step two iterations and $A_H > 9$. This is expected since the 50 stage problem is more difficult and a larger A_H leads to a more gradual region contraction. The Mekarapiruk-Luus method has generally poor results in this case except at $A_H = 35$. It is interesting that the Mekarapiruk-Luus method gives a fair result at $A_H = 14$ and $k_r = 0.7$ or $k_r = 4.0$ but not intermediate values of k_r .

These results suggest that the number of history iterations is a more important parameter than k_r . For basic IDP or two-step IDP, faster solution may be possible using small values of A_H , but large values of A_H are more likely to get a good solution reliably.

		$SSE_n \times 10^4$ after 20 step 2 iterations					$SSE_n \times 10^4$ after 60 step 2 iterations						
Mekarapiruk-Luus	k_r	A_H					A_H						
			9	14	20	27	35		9	14	20	27	35
		0.7	22	19	18	18	18	0.7	10	10	10	10	9
		1.0	32	9	18	18	18	1.0	10	10	11	10	10
		1.5	23	47	18	18	18	1.5	10	15	9	13	13
		2.0	8	86	18	18	18	2.0	10	14	16	10	12
4.0	11	4	18	18	18	4.0	10	13	9	10	11		
Average Delta	k_r	A_H					A_H						
			9	14	20	27	35		9	14	20	27	35
		1.0	9	52	18	18	18	1.0	10	10	10	10	11
		2.0	9	52	18	18	18	2.0	10	10	10	10	11
		4.0	20	52	18	18	18	4.0	10	10	10	10	11
		8.0	8	52	18	18	18	8.0	10	11	9	10	11
16.0	8	52	18	18	18	16.0	10	10	10	11	11		
Maximum Delta	k_r	A_H					A_H						
			9	14	20	27	35		9	14	20	27	35
		0.7	13	52	18	18	18	0.7	10	10	10	10	11
		1.0	7	52	18	18	18	1.0	10	11	9	10	11
		1.5	15	55	18	18	18	1.5	10	10	12	11	11
		2.0	7	41	18	18	18	2.0	10	10	10	12	11
4.0	26	23	18	18	18	4.0	10	11	11	24	10		
Span	k_r	A_H					A_H						
			9	14	20	27	35		9	14	20	27	35
		0.7	8	52	18	18	18	0.7	10	10	9	10	11
		1.0	8	52	18	18	18	1.0	10	11	9	10	11
		1.5	4	55	18	18	18	1.5	10	10	12	11	11
		2.0	7	41	18	18	18	2.0	10	10	10	12	11
4.0	26	22	18	18	18	4.0	10	11	8	24	10		

Table 5.3: Adaptive region size method and parameters A_H and k_r effect on SSE_n (Equation 5.8) for two-step IDP solutions to the 25 stage Park-Ramirez optimal control problem. Table 4.3 (page 38) gives the IDP parameters for the first step. The IDP parameters for the second step were the same as the first except for using 3 random test controls, 2 pivot point test controls, and the adaptive region size parameters shown on this table.

		$SSE_n \times 10^4$ after 20 step 2 iterations					$SSE_n \times 10^4$ after 60 step 2 iterations						
Mekarapiruk-Luus	k_r	A_H					A_H						
			9	14	20	27	35		9	14	20	27	35
		0.7	45	32	32	32	32	0.7	24	13	14	24	13
		1.0	33	44	32	32	32	1.0	29	20	26	23	14
		1.5	54	40	32	32	32	1.5	22	19	22	24	13
		2.0	44	55	32	32	32	2.0	23	43	26	24	13
4.0	76	46	32	32	32	4.0	64	15	27	27	24		
Average Delta	k_r	A_H					A_H						
			9	14	20	27	35		9	14	20	27	35
		1.0	29	29	32	32	32	1.0	12	5	5	16	12
		2.0	29	29	32	32	32	2.0	12	6	5	16	12
		4.0	27	29	32	32	32	4.0	11	6	5	16	12
		8.0	47	31	32	32	32	8.0	37	12	5	16	12
16.0	52	34	32	32	32	16.0	19	13	16	12	12		
Maximum Delta	k_r	A_H					A_H						
			9	14	20	27	35		9	14	20	27	35
		0.7	23	29	32	32	32	0.7	13	14	5	16	12
		1.0	25	29	32	32	32	1.0	14	5	12	16	12
		1.5	42	30	32	32	32	1.5	15	10	3	12	13
		2.0	30	35	32	32	32	2.0	10	14	4	10	20
4.0	52	59	32	32	32	4.0	16	20	14	19	13		
Span	k_r	A_H					A_H						
			9	14	20	27	35		9	14	20	27	35
		0.7	22	29	32	32	32	0.7	12	14	5	16	12
		1.0	25	29	32	32	32	1.0	9	12	12	15	12
		1.5	18	28	32	32	32	1.5	10	7	3	3	13
		2.0	25	36	32	32	32	2.0	11	14	3	19	19
4.0	33	63	32	32	32	4.0	23	11	14	9	13		

Table 5.4: Adaptive region size method and parameters A_H and k_r effect on SSE_n (Equation 5.8) for two-step IDP solutions to the 50 stage Park-Ramirez optimal control problem. Table 4.3 (page 38) gives the IDP parameters for the first step. The IDP parameters for the second step were the same as the first except for using 3 random test controls, 2 pivot point test controls, and the adaptive region size parameters shown on this table.

5.6 Two-step conclusions

This section presents a conservative guideline for finding a smooth solution without finding local minima. Although conclusions drawn on the solution of one model are limited, a few suggestions may be useful. The two-step method was clearly superior to basic IDP with smoothing techniques for the Park-Ramirez optimal control problem, so the two-step method should be used. In general, stagewise linear continuous controls should be used since they appear to have no downside. A low or intermediate number of stages should be tried (10-20), only increasing this if the problem requires it. For the second step, two random test controls and two pivot point test controls should be used. If more smoothing is needed add control damping with $\beta_{u\Phi} = 10^{-4}$, simulated annealing filter with $T_i = 10^{-3}$ and $\alpha_A = 0.01$, and increase the number of iterations.

If these suggestions do not work, a more intuitive feel for the IDP algorithm will probably be needed to know what to try next. A more complete summary of IDP recommendations is in Chapter 8.

5.7 Comparison with previously reported results

5.7.1 IDP Implementation

Part of this study is a new C library implementation (Webb, 2006) of all the IDP variations presented. All options (ie. control stage discretization, filter modes, etc.) are built into a single dynamically-linked library and available via runtime settings. The code is written in ISO1999 C in an object-oriented style and released¹ under a free software license (Free Software Foundation, 1991), so it is easy to extend, and the extended versions can also be freely distributed.

For the Park-Ramirez optimal control problem, the IDP C library developed for this study is approximately 9 times slower than the FORTRAN implementation used

¹ See Appendix A

by Luus (2000). The C library performs about 300 equivalent integrations per second of the Park-Ramirez bioreactor model on a Pentium II/360 processor when used with a Runga-Kutta 4/5 integrator with a relative tolerance of 10^{-6} . Based on the results reported in Luus (2000), the FORTRAN code used in that study performs about 2700 equivalent integrations per second on a Pentium II/350 processor. Part of this difference is the price paid for using an abstract library implementation of the algorithm, with its large volume of pointer passing and function calls, which would be expected to reduce the speed of an abstract C library compared to a FORTRAN implementation somewhat. Part of the difference may also be explained by the observation that the Runga-Kutta 4/5 integrator used in this study often overestimates the local relative error and thus solves at higher precision than requested (results not shown) and that Runga-Kutta 4/5 may not be as efficient as the integrator used by Luus.

This difference should not be too surprising to someone skilled in computer programming. Problems implemented in FORTRAN are nearly as fast as hand-optimized assembler code; the lack of pointers and other features allow a FORTRAN compiler to highly optimize the resulting machine code. C programs require more manual optimization effort to achieve a similar performance, and even then will not match the performance of FORTRAN. The C library implementation has not had much optimization work, and there is probably room for improvement.

On the other hand, IDP implemented in the high-level interpreted language Matlab was approximately 20 times slower than the C library (results not shown). The C library is a middle ground between the extensibility and reusability of an interpreted language implementation and the speed of a FORTRAN implementation. For problems where the model evaluation is expensive, the difference between implementations will rest almost entirely on the efficiency of the model evaluation and integrator, since the model integration accounts for almost all computation.

Stages	Iterations		Equivalent Integrations		Computer Time (sec)	
	Basic IDP	Two-step	Basic IDP	Two-step	Basic IDP	Two-step
10	60	60	1.4×10^4	1.4×10^4	3.5	5.0
25	1000	70	1.5×10^6	1.1×10^5	420	25
50	4000	100	2.4×10^7	6.1×10^5	3700	57

Table 5.5: Computational cost of the basic IDP algorithm compared with the two-step method using pivot point test controls to obtain a visually smooth control profile for the Park-Ramirez optimal control problem. The basic IDP case is identical to Table 5.1 (page 49). The two-step IDP case is the same as the basic IDP case except with 2 pivot point test controls and 3 random test controls during the second step. The processor used was an AMD Athlon 2100+, and equivalent integrations were found using Equation 4.2.

5.7.2 Two-step method

To compare algorithmic efficiency independently of implementation efficiency, algorithmic cost will be reported in “equivalent integrations” (I_{eq}). See Section 4.1 (page 21) for the definition of equivalent integrations and how they are calculated. Table 5.5 compares the basic IDP algorithm with the two-step method using pivot point test controls. For this particular problem where performance index is only expected within 0.1% of the optimal, the two-step method greatly reduces the computation cost of many stages.

Chapter 9 of Luus (2000) explores the Park-Ramirez optimal control problem extensively using the basic stagewise constant controls IDP algorithm. The initial solution was with 15 stages and 2000 iterations for $I_{eq} = 1.0 \times 10^6$. In order to get a completely smooth control profile iterations were continued for an additional $I_{eq} = 2.5 \times 10^6$. A case with 20 stages was then discussed, with a very smooth control profile at $I_{eq} > 7.1 \times 10^7$. These values are consistent with the equivalent integrations required to obtain a visually smooth control profile using the basic IDP algorithm implemented for this study.

Chapter 6

Hybrid dynamic model identification

Hybrid mechanistic/empirical dynamic models are a way to model a system when some information about the system's structure is well known, but some information is poorly known. An example of this is in bioreactor modeling, where it is common to know the functional form of the reaction stoichiometry but not the functional form of the reaction rates. Splitting the model into known and unknown structures brings more information into the model than a strictly empirical model, with less work than developing a comprehensive mechanistic model. However, model identification can be more difficult than with purely empirical models or purely mechanistic models, and this topic has not been fully explored.

Section 6.1 discusses the two methods currently used for hybrid dynamic model identification along with their limitations. Sections 6.2 and 6.3 introduce a novel and more general identification technique where the identification problem is treated as an optimal control problem. The hybrid Park-Ramirez model is identified using artificial data and this new method in Sections 6.4–6.8, and a discussion follows.

6.1 Identification methods

Consider the simple one-dimensional ODE hybrid dynamic model

$$\begin{aligned}\dot{X} &= v(X, \omega) \cdot X \\ X(t=0) &= X_0\end{aligned}\tag{6.1}$$

where X is the state variable and $v(X, \omega)$ represents a neural network or other function estimator that takes X and a vector of static parameters ω as its input. The function estimator v will be called a “parameter function” because it is used as a dynamic model parameter to be identified using data.

The identification problem for this model is to find the values of the static parameters ω given a set of measured state (X_m) vs. time data. This set may come from multiple system runs with different state initial condition for each run. The basic difficulty of this problem is that it is not immediately obvious how to link the parameters ω with the state errors (measured state values vs. predicted state values) since there is an intermediate mathematical step (the ODE integration) between them. There are currently two methods to identify ω : the derivative method and the integral method.

6.1.1 Derivative method

If only a single run is considered, Equation 6.1 can be rearranged to

$$v(t) = \frac{\dot{X}}{X}. \quad (6.2)$$

We call this method the derivative method because the key step is obtaining the state derivative estimate from the state data. The state derivative estimate is normally obtained by curve-fitting the data and taking the derivative of the curve-fit. This means there are really two identification steps with the derivative method: first the $v(t)$ identification for each run using curve-fitting, then the ω identification using the results of the $v(t)$ identification.

To identify ω using $v(t)$, a discrete set of input-output training data is extracted from the $v(t)$ data for all runs along with the corresponding states. The neural networks $v(\mathbf{X}, \omega)$ are then be trained normally on the input-output set.

6.1.2 Limitations of the derivative method

One disadvantage of the derivative method is that noise in the data amplifies the uncertainty of the \dot{X} estimate (Himmelblau, 1970). State estimators can be used, but the problem of accurately estimating \dot{X} is still theoretically a problem with this method if data are noisy, especially if the number of data for each run is very limited (Oliveira, 2004). Tholudur et al. (1999) overcame this problem by developing an accurate curve-fit method for the system they studied. Laursen et al. (2007) obtained good results using the derivative method with smoothing spline fits to very noisy industrial data. The problem of noise therefore can be addressed successfully for the derivative method.

A more important limitation of this method is that it requires all states be measured and to have sufficient data for curve-fitting. For the system modeled by the Park-Ramirez bioreactor model, the total protein concentration is expected to be more difficult to measure than the secreted protein, so an interesting question is if the model parameters can be identified with very little or no total protein concentration data.

6.1.3 Integral method

Another way to identify ω is to minimize the sum squared error between the measured state data and the model prediction:

$$E = \sum_z \sum_p^{P_z} (X_m^p - X^p)^2 \quad (6.3)$$

where Z is the number of runs, P_z is the number of data for each run z , X_m^p is the measured state value for sample p , and X^p is the model predicted state value corresponding to sample p .

The integral method finds $\frac{\partial E}{\partial \omega}$ so that the ω which minimizes E can be found using standard minimization techniques such as Levenberg-Marquardt. It is called the integral method because the sensitivity information which links ω and E is obtained during a model integration that has additional sensitivity equations.

The integral method can be broken into four parts:

- (1) Take the partial derivative of Equation 6.3 with respect to X to get $\frac{\partial E}{\partial X}$.
- (2) Take the partial derivative of the model equation with respect to v to get $\frac{\partial X}{\partial v}$.

This results in the sensitivity equation, which for Equation 6.1 is

$$\frac{dG}{dt} = vG + X \quad (6.4)$$

where $G = \frac{\partial X}{\partial v}$. The sensitivity equation is integrated together with the model equation with the initial condition $G = 0$ since the state at the initial condition is fixed and thus does not depend on v .

- (3) Take the partial derivative of the parameter function v with respect to ω . For the case of feedforward neural networks the backpropagation technique gives $\frac{\partial v}{\partial \omega}$.
- (4) The multiplication chain rule for derivatives gives

$$\frac{\partial E}{\partial \omega} = \frac{\partial E}{\partial X} \frac{\partial X}{\partial v} \frac{\partial v}{\partial \omega}. \quad (6.5)$$

The integral method is executed by integrating the model equations and the sensitivity equations once for each data run to build a final value for $\frac{\partial E}{\partial \omega}$ at the end of all integrations. The evaluation of $\frac{\partial E}{\partial \omega}$ is called by a nonlinear parameter identification algorithm such as Levenberg-Marquardt to solve for ω . This is a direct method, where ω is identified directly from E , unlike the derivative method and the Webb-Ramirez method proposed in Section 6.2.

This method would seem to be a better training method than the derivative method because it avoids noise amplification during derivative estimation and can handle unmeasured states. Oliveira (2004) found it superior to the derivative method for two problems with noisy data. There are also other examples of this method in the literature (Psichogios and Ungar, 1992; Teixeira et al., 2005).

6.1.4 Limitations of the integral method

The integral method is not possible if the partial derivatives in steps 2 or 3 cannot be evaluated. An example of a model where $\frac{\partial X}{\partial v}$ is difficult to evaluate is the bioreactor model by Xu et al. (1999), where logic elements within the model make it difficult to obtain numerical or analytical derivatives.

An example of a function estimator where $\frac{\partial v}{\partial \omega}$ is difficult to calculate is a radial basis function (RBF) neural network when some states are estimated. RBF training starts by executing a clustering algorithm on the input data. If any estimated states are input for the RBF neural network, these states must come from the model, which changes each iteration as the training progresses. The clustering algorithm must operate on changing states, so the derivative $\frac{\partial v}{\partial \omega}$ can't be calculated because the clustering algorithm has no derivative. RBF neural networks are particularly attractive for the hybrid dynamic identification problem, because they can have both a training data density estimator and an output confidence bound for all predictions (Leonard et al., 1992). These confidence indicators are important because they allow model-based optimization to avoid parts of the state domain where data density or prediction confidence is low.

Finally, the integral method requires that the model states be continuous. If states have discontinuities, the sensitivity equation becomes meaningless and there is no way to link parameter weights with the error. A practical case where state discontinuity is useful for identification will be shown in Section 6.8.8. Although state discontinuity is not currently common in dynamic models, optimization of mixed continuous/discrete systems is an active topic of research (Barton et al., 1998).

6.2 Optimal control for identification (Webb-Ramirez method)

The more general formulation of the hybrid model identification problem is: given the model

$$\mathbf{g}(\dot{\mathbf{X}}, \mathbf{X}, \mathbf{v}(\mathbf{X}, \omega)) = 0 \quad (6.6)$$

$$\mathbf{X}(t = 0) = \mathbf{X}_0 \quad (6.7)$$

find the static parameters ω that minimize the error

$$E = \sum_z^Z \sum_p^{P_z} (\mathbf{X}_m^p - \mathbf{X}^p)^2 \quad (6.8)$$

where \mathbf{X} is a state vector, $\mathbf{v}(\mathbf{X}, \omega)$ is a vector of functions which take \mathbf{X} and ω as input, Z is the number of runs, P_z is the number of data for each run z , \mathbf{X}_m^p is the vector of measured state values for sample p , and \mathbf{X}^p is the vector of model predicted state values corresponding to sample p .

The Webb-Ramirez method proposed here to identify ω is:

- (1) Identify $\mathbf{v}(t)$ for each run by treating Equations 6.6–6.8 as an optimal control problem where $\mathbf{v}(t)$ are the controls, Equation 6.6 is the system model and Equation 6.8 is the objective function to be minimized ($\Phi = E$).
- (2) Identify ω with $\mathbf{v}(t)$ and state data from step 1 using normal identification methods for $\mathbf{v}(\mathbf{X}, \omega)$, such as backpropagation for feed-forward neural networks.

If IDP is the optimal control algorithm used in step 1, there are no limitations on the system model or the parameter functions except that there must be an algorithm to move the model states forward in time (ie. an ODE integrator for an ODE model, a DAE integrator for a DAE model, etc.).

Algorithm 9 Limit enforcement algorithm for state low limit during optimal control

1. $\Omega_X = 0$
2. $\Omega_{\dot{X}} = 0$
3. **if** $X \geq X_l$ **then**
4. return
5. **end if**
6. **if** $X \leq X_L$ **then**
7. $\Omega_X = 1 + \frac{X_l - X}{X_L}$
8. $X = X_L$
9. **if** $\dot{X} < 0$ **then**
10. $\Omega_{\dot{X}} = -\dot{X}$
11. $\dot{X} = 0$
12. **end if**
13. return
14. **end if**
15. **if** $\dot{X} < 0$ **then**
16. $\lambda = 2 - \exp \left[\ln(2) \frac{X_l - X}{X_l - X_L} \right]$
17. $\dot{X} = \lambda \dot{X}$
18. $\Omega_X = 1 - \lambda$
19. $\Omega_{\dot{X}} = (\lambda - 1) \dot{X}$
20. **end if**

 symbols:

- X state value (from integrator)
 - X_l state soft limit
 - X_L state hard limit
 - \dot{X} state derivative (from model equation evaluation)
 - λ state derivative adjustment
 - Ω_X state punishment term
 - $\Omega_{\dot{X}}$ derivative punishment term
-

6.3 Ensuring feasible states for dynamic models

For many dynamic models, states are all self-limiting. For example, with bioreactor models the cell concentration can never run away because substrate limits the maximum growth rate. Substrate concentration can't fall completely to zero because cells take up less substrate as the substrate concentration falls, and so on. With the Webb-Ramirez identification scheme, states can run away because the model is no longer self-limiting. The IDP algorithm may choose the maximum value for the growth rate even when substrate is gone, which will lead to runaway cell concentration and negative substrate concentration if substrate concentration is not limited.

This causes two problems with numerical integrators. First, some models are not defined for some state values, such as negative values of cell concentration or substrate concentration for a bioreactor model. When states are running away, the derivative of other states may be so high that the integrator takes steps into illegal values of the state during test steps. More importantly, runaway states can create a stiff model even if the model is normally non-stiff. This stiffness will cause an efficient non-stiff integrator such as Runge-Kutta 4/5 to slow down immensely.

States should be limited within a reasonable range regardless of the parameter function values. Algorithm 9 is the limit enforcement algorithm, and is always executed after the system model equation (Equation 6.6) is evaluated. This allows it to modify the state derivatives or states seen by the integrator. In Algorithm 9, there is a "soft" and "hard" limit for each state X . States are never allowed to go beyond the hard limits. Between the soft and hard limits the states are not changed but the derivative is adjusted towards zero.

For example, if substrate concentration in the Park-Ramirez bioreactor model has a soft lower limit of $0.01 \frac{g}{L}$ and a hard lower limit of $0.001 \frac{g}{L}$, an integrator that calls the model function with a substrate concentration of $0.0005 \frac{g}{L}$ will have the substrate con-

centration changed to $0.001 \frac{g}{L}$ and the derivative returned will be zero. If the integrator calls the model with a substrate concentration of $0.005 \frac{g}{L}$, the substrate concentration will not be changed, but if the derivative is negative it will be adjusted towards zero (Algorithm 9 step 17). This provides a gradual adjustment of the derivative instead of a sharp cutoff to avoid numerical problems.

The soft and hard low limits were 10^{-4} and 10^{-5} , and the soft and hard high limits were 20 and 25 for all states except volume for the Park-Ramirez model. Volume can never run away for this model if the feed rate is bounded so no limits were needed.

Additionally, there may need to be a way to encourage the optimal control algorithm to move the parameter function away from the hard limits. This is accomplished by adding the state punishment variable Ω_X and the derivative punishment variable $\Omega_{\dot{X}}$ to the performance index:

$$\Phi^* = \Phi - \Omega_X - \Omega_{\dot{X}} \quad (6.9)$$

and using the modified performance index Φ^* . The punishment variables are proportional to how far past the limit a state or derivative is, but the specific magnitude of these punishments should not be critical since the optimizer should move away from them by the end of the optimization anyway for the identification problem. This punishment method is similar to the method used by Mekarapiruk and Luus (1997), with the exception that all boundary punishment is combined into a single punishment state in this implementation. For the Webb-Ramirez identification of the Park-Ramirez model, punishment was found to harm the results by finding poor local minima, and was not used. A hypothesis for the reason is that punishment limits potential solution paths, much like control damping is hypothesized to harm the optimal control solution by limiting solution paths.

An addition comment should be made here on the state change in step 8 of Algorithm 17. Because most integrators do not allow the state derivatives user routine

to change the values of the states, a custom version of the Runge-Kutta 4/5 integrator was created which allows states to be changed by the user state derivatives function. A runtime option in the library can also disable the state change so that other integrators can be used.

6.4 Identifying the hybrid Park-Ramirez model

The Park-Ramirez bioreactor model introduced in Chapter 2 is a hybrid dynamic model if the three reaction rates are replaced with neural networks. This model was used to explore hybrid dynamic model identification with the following steps:

- (1) Generate artificial state data sets for one or more runs using the Park-Ramirez bioreactor model and the mechanistic parameter functions (Equations 2.3, 2.4, and 2.5). Noise may be added to this data.
- (2) Using the artificial state data runs from step 1, identify the parameter function values with respect to time $v(t)$ using the Webb-Ramirez identification technique with IDP for each run.
- (3) Using $v(t)$ values for all runs from step 2 simultaneously, identify the parameter function neural networks $v(\mathbf{X}, \omega)$ using backpropagation training.
- (4) Compare the identified neural network parameter functions $v(\mathbf{X}, \omega)$ with the mechanistic parameter functions from step 1.
- (5) Compare the optimal substrate feed profile found for the mechanistic model with the optimal substrate feed profile for the identified model.

6.5 Artificial data

Generating artificial data means the parameter function and state values will be known, so the identification technique can be evaluated more easily than if real data

are used. There are five basic decisions to make when generating artificial data for this evaluation:

- (1) How many data runs are created?
- (2) What are the initial state values for each data run?
- (3) How many state data are recorded for each data run?
- (4) What is the control profile for each data run?
- (5) How much noise is added to the state data?

To reduce the complexity of the identification study, the initial state values used for all runs of the Park-Ramirez bioreactor model were identical and are given in Table 2.1 (page 8). To generate an artificial data run, the Park-Ramirez model equations were integrated using control $q^z(t)$ from Figure 6.1 where z is the run index. The mechanistic parameter functions were used during this integration: growth rate (μ , Equation 2.3), protein production rate (f_P , Equation 2.4), and protein secretion rate (ϕ , Equation 2.5).

Gaussian white noise was added to the state data when generating the artificial data sets. The noisy state value X^* is

$$X^* = X + N(\mu = 0, \sigma = \sigma_N X) \quad (6.10)$$

where X is the state value, N is a random variable of the normal distribution, μ is the mean of N , and σ is the standard deviation of N . The standard deviation parameter σ_N is 0 or 0.01 for this study.

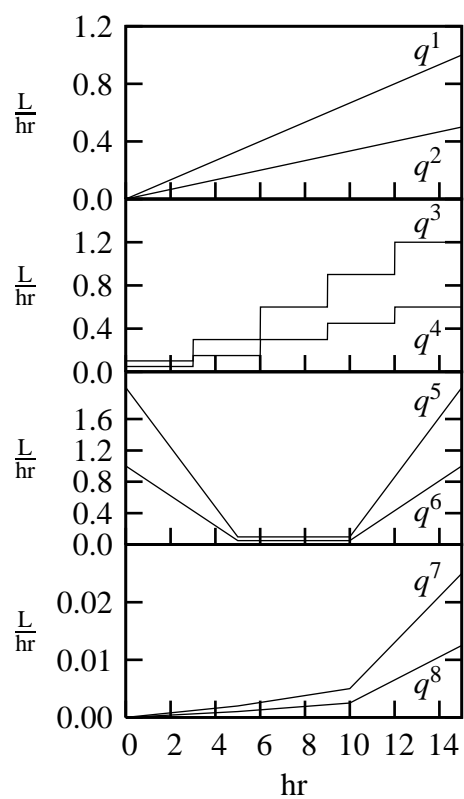


Figure 6.1: Park-Ramirez model substrate feed rate used to generate artificial state data for identification studies. Each control profile q^z corresponds to data run z .

6.6 Identifying the hybrid Park-Ramirez model parameter functions $v(t)$

6.6.1 Evaluating the effect of IDP parameters on Webb-Ramirez $v(t)$ identification

In Sections 6.6.2–6.6.5, the hybrid Park-Ramirez model parameter function values $\mu(t)$, $f_P(t)$, and $\phi(t)$ are identified using one run of artificial data generated with control profile q^1 from Figure 6.1. Since the initial state values, control profile, and data generation parameters were all constant, the same state data set was used for all cases. This allowed the effect of IDP parameters on the Webb-Ramirez identification technique to be determined for the Park-Ramirez model.

The first step in the overall process was to determine the appropriate adaptive region update parameters for IDP. By observing the change in performance index and through some trial and error, the history iterations were chosen to be 100, 150, and 200 iterations for 10, 25, and 50 stages respectively. These values are much larger than for the Park-Ramirez optimal control problem because there are three “controls” and the identification problem is much more difficult. The adaptive region parameter k_r was chosen to be 1.5, although it was determined in Section 4.5.2 that the value of this parameter is probably not critical as long as it is within a reasonable range.

6.6.2 Webb-Ramirez $v(t)$ identification using basic IDP

Figures 6.2, 6.3, and 6.4 present the $v(t)$ identification using basic IDP for 10, 25, and 50 stages respectively. The performance index for Webb-Ramirez identification is the state sum squared error (Equation 6.8), so the algorithm’s only objective is choosing parameter functions that get the state lines through the state data points in Figures 6.2–6.4. This objective was mostly accomplished after 1800 iterations for 10, 25, and 50 stages. The total protein is not fit perfectly because there is no P_T data. However, the real goal of the identification is to obtain the correct parameter function values, and

that goal was not accomplished for f_P or ϕ for any of the cases. Notice that the 25 and 50 stage cases have the kind of control action seen with the basic IDP solution of the Park-Ramirez optimal control problem (Figures 5.2 and 5.3 on pages 47 and 48).

The value of the growth rate parameter function μ is very important for the performance index since all states are strongly affected by the cell concentration, so the growth rate parameter function fits much better than the other two parameter functions. The protein production rate parameter function f_P is more important to the state fit until the last one third of the run, so it fits relatively well until then. The protein secretion rate parameter function is the least sensitive and is very difficult to capture since it only affects one state. Also, for the middle three-fourths of the run, the value of the protein secretion rate parameter function doesn't affect the state at all, so its value is essentially random. This problem will be discussed more in Section 6.7.1.

Number of state data (except P_T)	100
Number of P_T data	0, 5, or 100
Data noise (Equation 6.10)	$\sigma_N = 0$ or $\sigma_N = 0.01$
Stage control discretization	Stagewise linear continuous
Stage length mode	Fixed uniform stage length
Iterations (for single-step IDP)	1800
Iterations (first step for two-step IDP)	600
Iterations (second step for two-step IDP)	1200
State grids	1
Random test controls	6
Adaptive control region method	Maximum Delta
Adaptive control region history (10 stages)	100 iterations
Adaptive control region history (25 stages)	150 iterations
Adaptive control region history (50 stages)	200 iterations
Adaptive control region k_r	1.5

Table 6.1: Data generation and IDP parameters for the hybrid Park-Ramirez identification problem. Model parameters are in Table 2.1 (page 8).

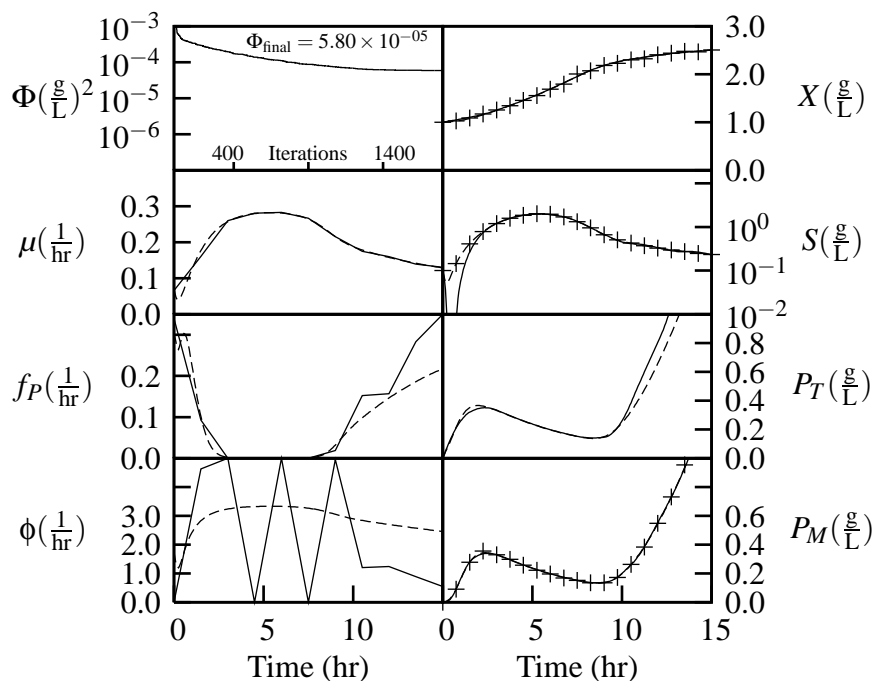


Figure 6.2: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , and P_M using basic IDP with **10 stages**. The data generation and IDP parameters used are in Table 6.1 (page 107) with **0 P_T data** and $\sigma_N = 0.01$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index.

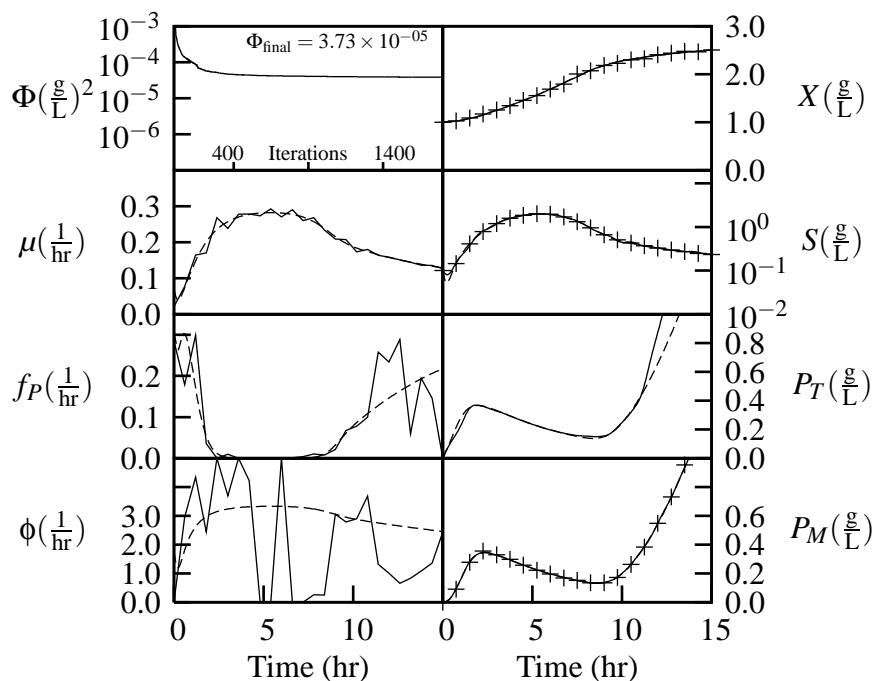


Figure 6.3: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , and P_M using basic IDP with **25 stages**. The data generation and IDP parameters used are in Table 6.1 (page 107) with **0 P_T data** and $\sigma_N = 0.01$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index.

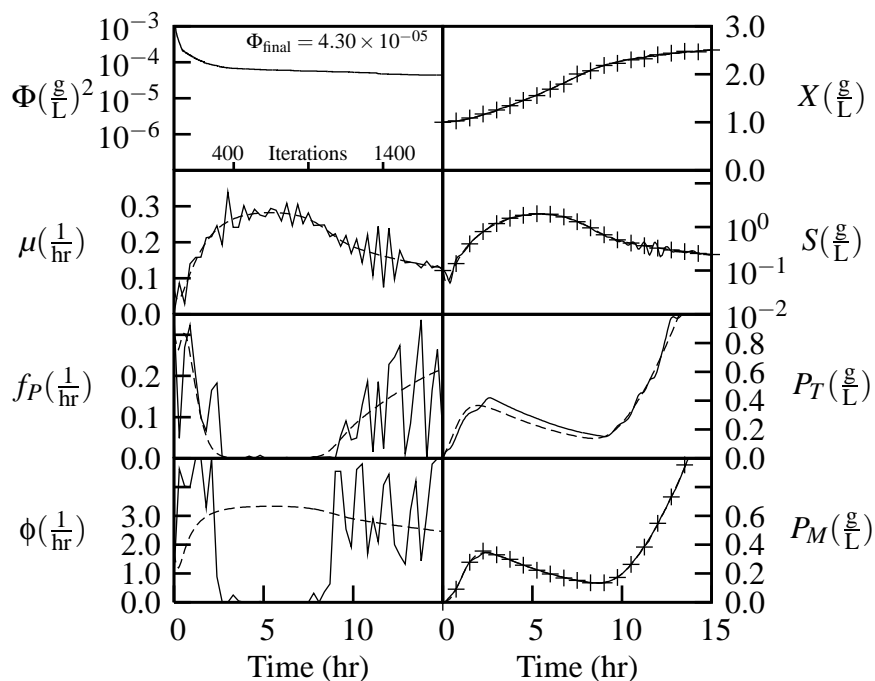


Figure 6.4: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , and P_M using basic IDP with **50 stages**. The data generation and IDP parameters used are in Table 6.1 (page 107) with **0 P_T data** and $\sigma_N = 0.01$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index.

6.6.3 Webb-Ramirez $v(t)$ identification using basic IDP with smoothing

This section presents the $v(t)$ identification using basic IDP with smoothing (simulated annealing filter and control damping). The cases are identical to Section 6.6.2, except that IDP smoothing techniques were used with the smoothing parameters in Table 6.2. Figures 6.5, 6.6, and 6.7 show the $v(t)$ identification results for 10, 25, and 50 stages respectively. The states fit approximately the same as without smoothing, but the parameter function identification is clearly superior, especially for f_P .

Random test controls	2
Solo test controls	2
Pivot point test controls	2
Control damping	$\beta_{u\Phi} = 0.001$
First-order filter	off
Simulated annealing filter	$T_i = 0.01, \alpha_A = 0.01$
Performance index magnitude	$M_\Phi = 5 \times 10^{-5}$

Table 6.2: IDP smoothing parameters for the hybrid Park-Ramirez identification problem. Parameters in this table supersede those in Table 6.1 when both are used.

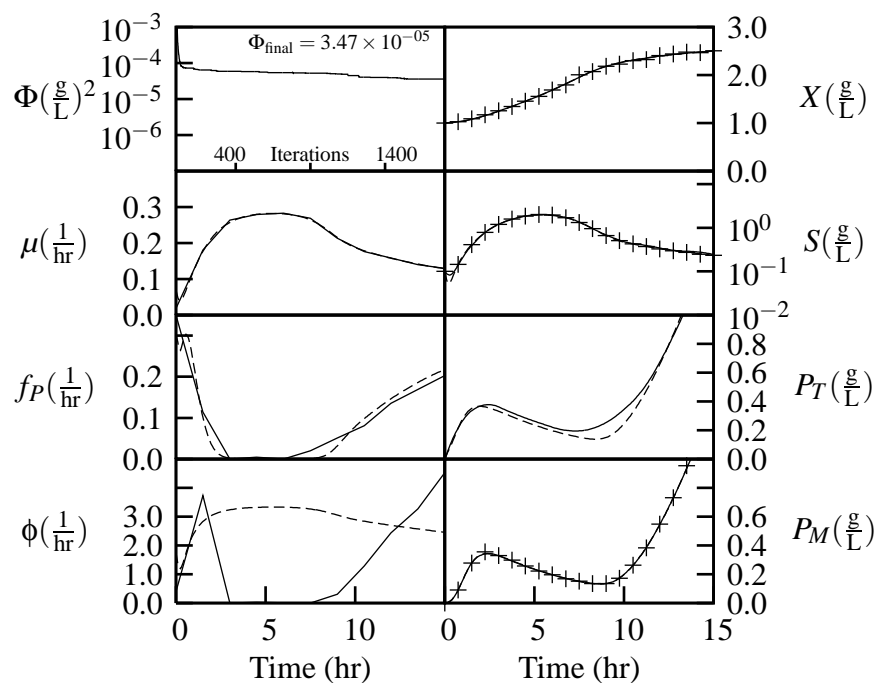


Figure 6.5: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , and P_M using smoothed basic IDP with **10 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **0 P_T data** and $\sigma_N = 0.01$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index.

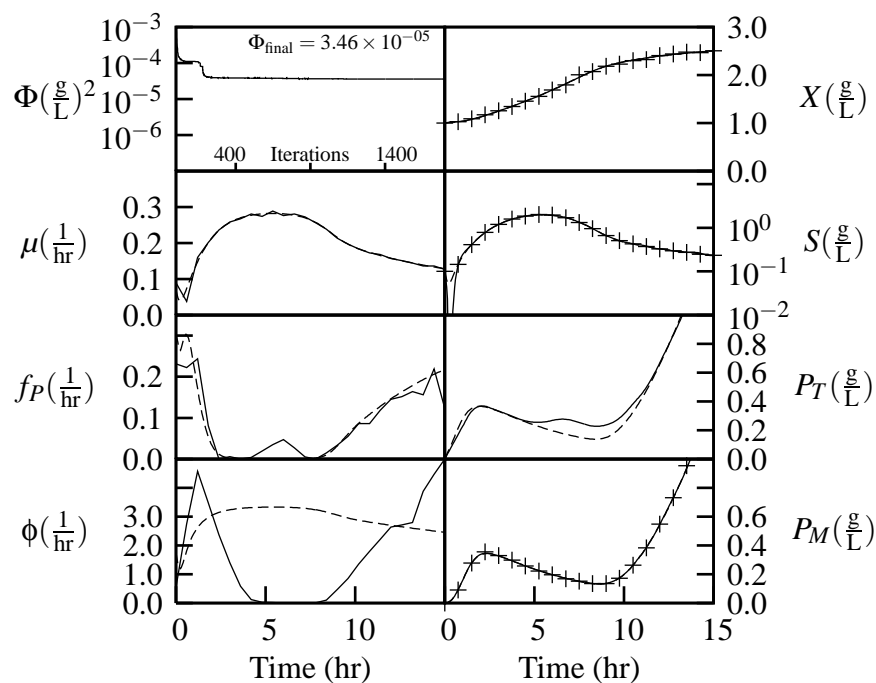


Figure 6.6: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , and P_M using smoothed basic IDP with **25 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **0 P_T data** and $\sigma_N = 0.01$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index.

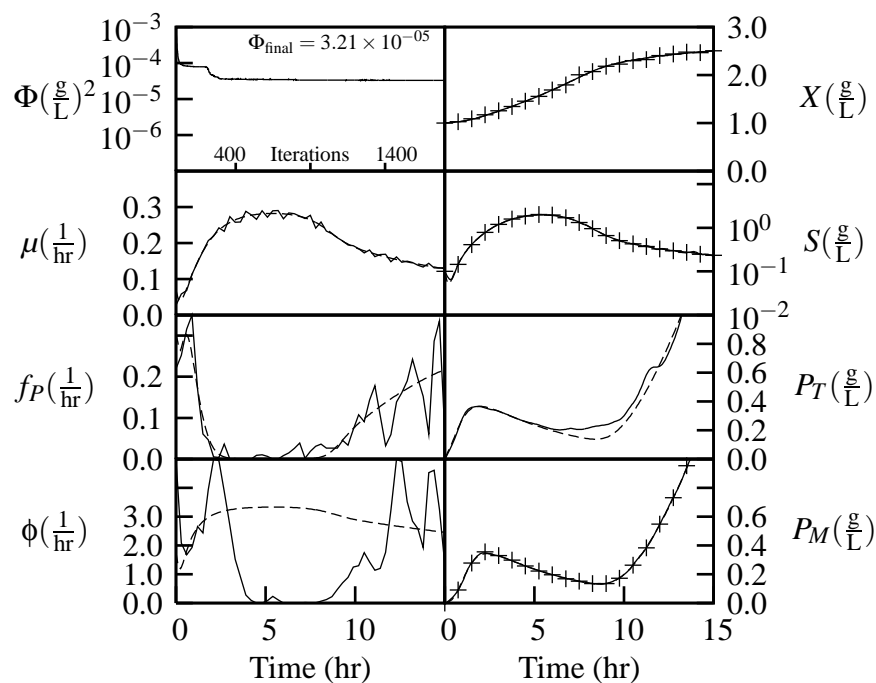


Figure 6.7: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , and P_M using smoothed basic IDP with **50 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **0 P_T data** and $\sigma_N = 0.01$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index.

6.6.4 Webb-Ramirez $v(t)$ identification using two-step IDP

The two-step method for IDP introduced in Section 3.7 found optimal control profiles with less likelihood of finding local minima than basic IDP with smoothing, so it is interesting to see if this method obtains similar results to basic IDP with smoothing for the Park-Ramirez identification problem.

For the Park-Ramirez optimal control problem, the number of step one iterations was chosen based on obtaining the performance index within some tolerance. Here, however, it is more important to obtain the correct parameter function profile, so instead of using the performance index as a guide, the step one iterations were chosen based on the control profile activity during the solution shrinking some arbitrary amount. Observing that the control profiles of the runs in Section 6.6.2 no longer changed much after 600 iterations, step one of the two-step IDP method was chosen to have 600 iterations.

Figures 6.8, 6.9, and 6.10 present the Webb-Ramirez $v(t)$ identification using two-step IDP for 10, 25, and 50 stages respectively. The total number of iterations for each of the four cases presented in each figure is the same as in the previous two sections, but they are split into 600 step one iterations plus 1200 step two iterations. The control profile used for these studies was control q^1 from Figure 6.1, and the data generation parameters used are given in Table 6.1 with $P_T = 0$ and $\sigma_N = 0.01$.

Although Webb-Ramirez identification was presented as a way to identify the parameter functions when some states are unmeasured, it is not magic. In Figure 6.9, the total protein concentration is very poorly identified starting in the middle of the run. This does not hurt the performance index because the measured state data are still fit very well. The problem is that the missing P_T data allows the alternate (and wrong) combination of f_P too high around 7 hours and ϕ too low between 9 and 15 hours. In optimization terms, the system must still be observable to get good results no matter

what identification method is used.

The two-step IDP method appears to give similar results to basic IDP with smoothing, although the difference is not significant. However, since the number of iterations is the same, the two-step method should be preferred since it has less likelihood of finding local minima.

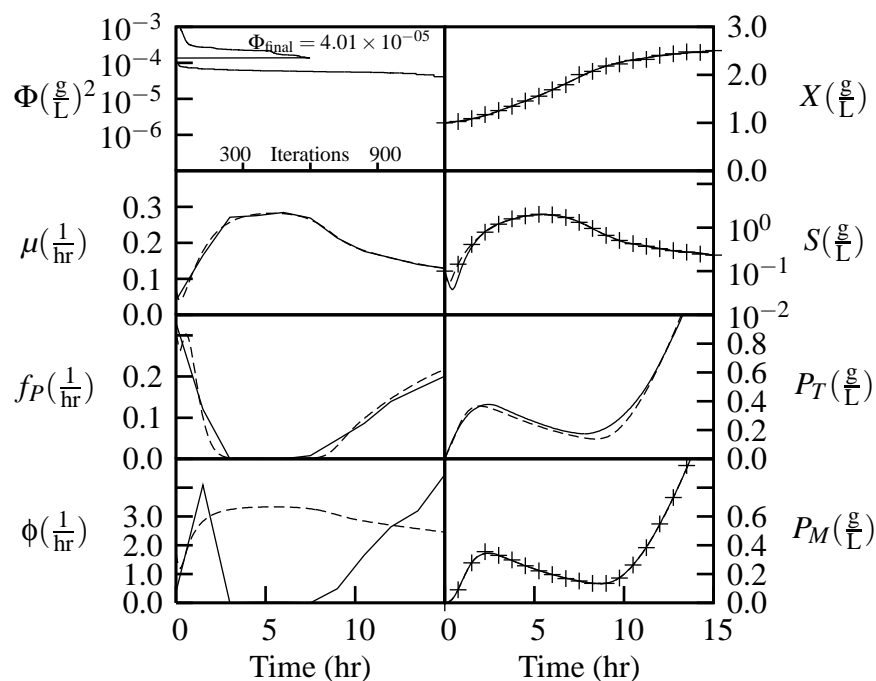


Figure 6.8: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , and P_M using two-step IDP with **10 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **0 P_T data** and $\sigma_N = 0.01$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

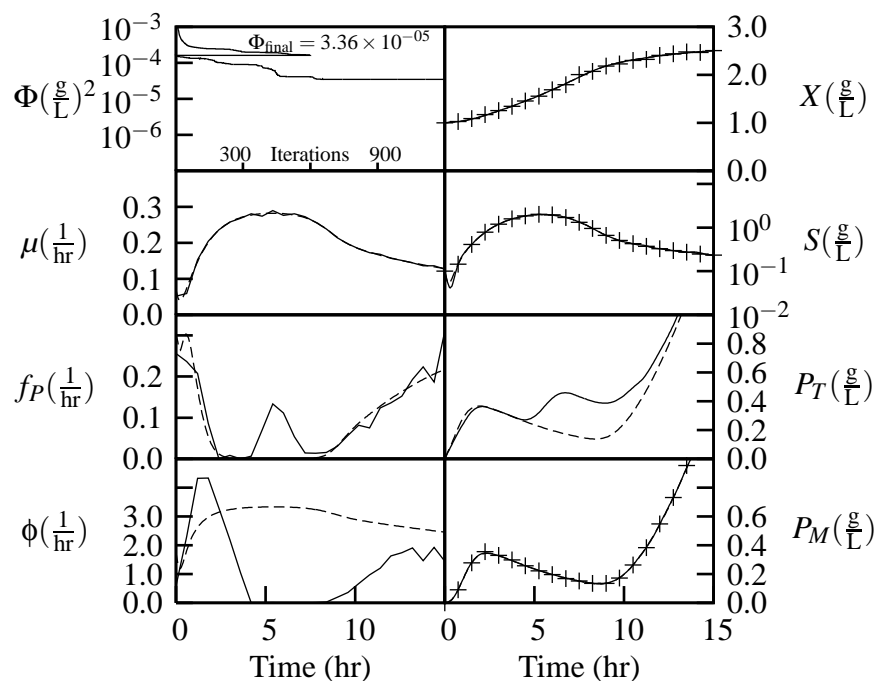


Figure 6.9: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , and P_M using two-step IDP with **25 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **0 P_T data** and $\sigma_N = 0.01$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

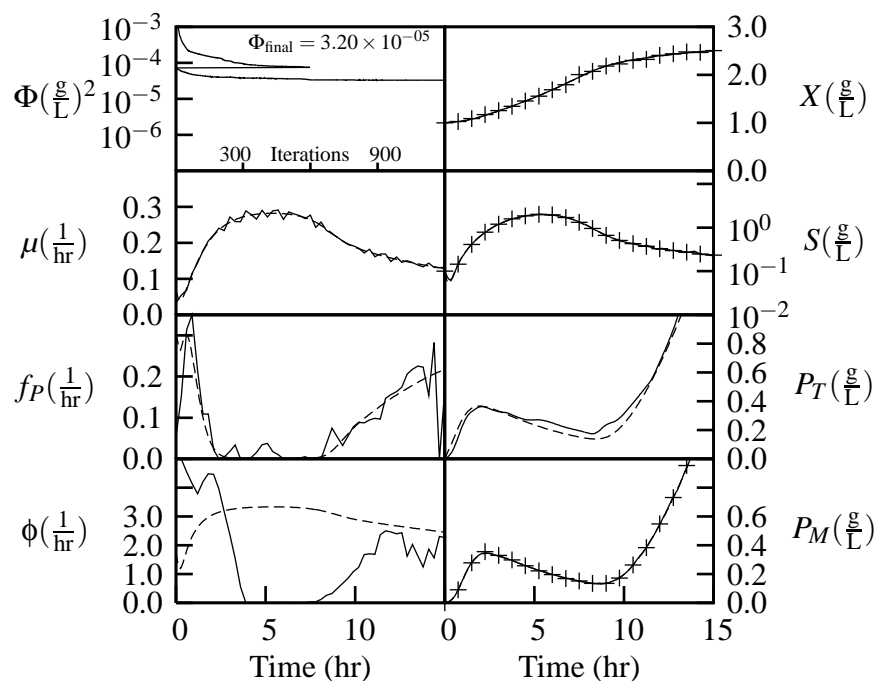


Figure 6.10: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , and P_M using two-step IDP with **50 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **0 P_T data** and $\sigma_N = 0.01$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

6.6.5 Webb-Ramirez $v(t)$ identification using two-step IDP and P_T data

In the previous section it was discovered that although the Webb-Ramirez identification method works well with two-step IDP, the system is not fully observable with total protein concentration unmeasured. This section repeats the cases from the previous section except with 5 P_T data per run. Figures 6.11, 6.12, and 6.13 present the Webb-Ramirez $v(t)$ identification using two-step IDP for 10, 25, and 50 stages respectively for these new cases with P_T measured.

There are still some identification problems when only 5 P_T data are used. In Figure 6.13, f_P is much too low and ϕ is much too high at the beginning of the run. As when no P_T data were measured, the lack of P_T data at the beginning allows the underdetermined dependency between these two parameters to degrade the solution. However, even this small amount of P_T data greatly improves the $v(t)$ identification overall and allows all the model parameter functions $v(\mathbf{X}, \omega)$ to be identified as will be seen in Section 6.8.

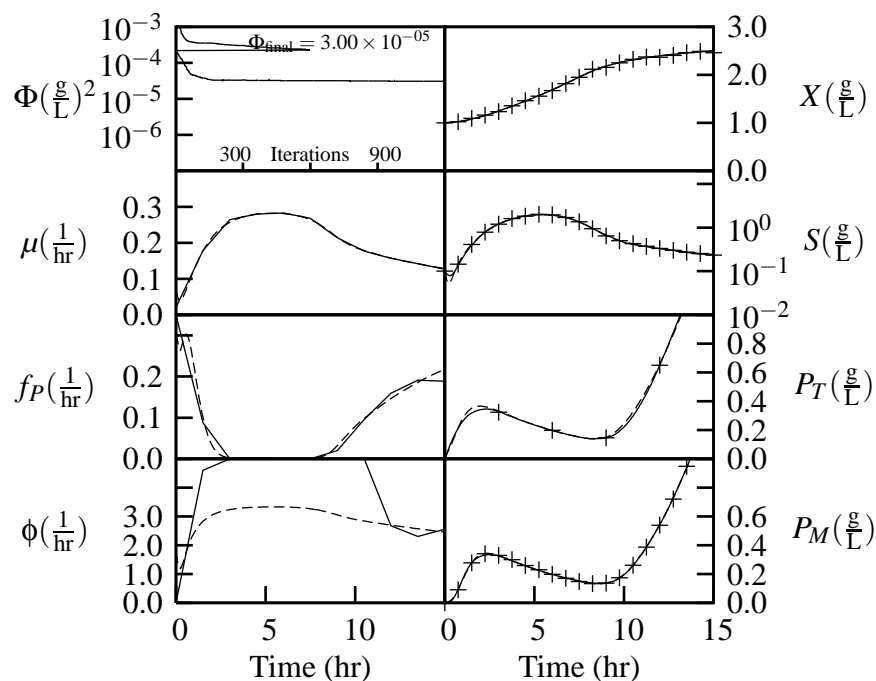


Figure 6.11: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , P_T , and P_M using two-step IDP with **10 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **5 P_T data** and $\sigma_N = 0.01$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

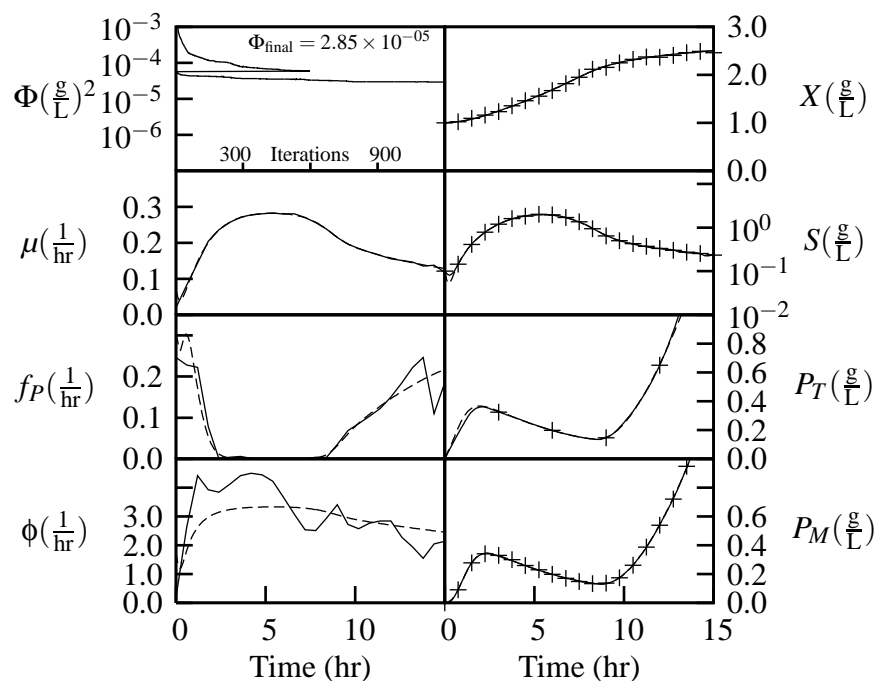


Figure 6.12: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , P_T , and P_M using two-step IDP with **25 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **5 P_T data** and $\sigma_N = 0.01$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

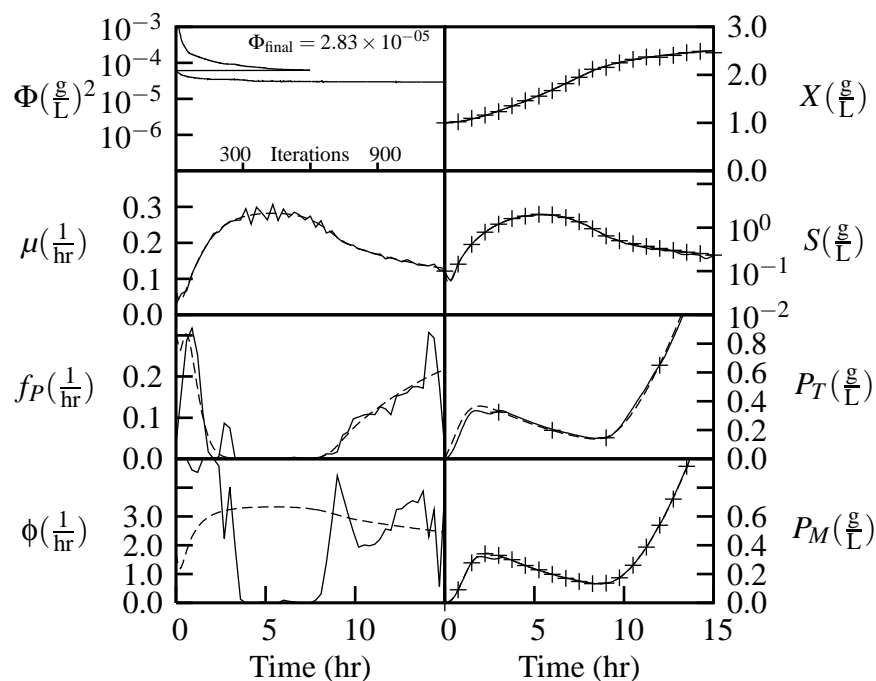


Figure 6.13: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , P_T , and P_M using two-step IDP with **50 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **5 P_T data** and $\sigma_N = 0.01$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

6.7 Identifying $v(\mathbf{X}, \omega)$ using $v(t)$ data

6.7.1 Parameter function sensitivity for improved identification

For either the derivative method or the Webb-Ramirez method, the first step is to identify parameter function $v(t)$ values for each run. These identified $v(t)$ values are then used along with the corresponding states to train a neural network or other universal function approximator to represent the parameter function.

However, there may be parts of the time domain where $v(t)$ is identified but is meaningless. For example, for the Park-Ramirez bioreactor model, protein secretion requires a driving force of higher total protein concentration than secreted protein concentration in the broth. If this driving force does not exist for parts of the time domain then the identified secretion parameter function value is meaningless since any value is as good as any other. If these meaningless data are used to train a neural network to represent the parameter function, the resulting neural network will be degraded.

The method used here to deal with this problem is to replicate $v(\mathbf{X}, \omega)$ training data more where $v(t)$ has more effect on the error E in Equation 6.8. Analytical values of $\frac{\partial E}{\partial v(t)}$ could be used via a sensitivity equation analogous to the sensitivity equation used with the integral method, but since part of the reason for proposing the Webb-Ramirez method was to avoid the analytical sensitivity equation it makes more sense to use the numerical approach from Section 4.6 (page 39) which is already easily available when using IDP. The forward and backward sensitivities from Section 4.6 are used in Algorithm 10 which is described in the next section.

6.7.2 Extracting $v(\mathbf{X}, \omega)$ training data from identified $v(t)$ data and $v(t)$ sensitivity

The second step of the Webb-Ramirez identification method is to train the parameter functions $v(\mathbf{X}, \omega)$ using $v(t)$ data from the first step. How to best extract the

training data is an open question, but it was accomplished for this study with Algorithm 10. Algorithm 10 integrates the model through each run, extracting input-output training pairs for each $v(\mathbf{X}, \omega)$ identification. The training pairs are created at IDP stage midpoints, which results in significantly less noise than if the training pairs are created at the stage boundaries. Training pairs are replicated proportionally to their sensitivity, so that more sensitive parts of the set receive more emphasis during training.

Algorithm 10 can use predicted or data state values for the training inputs. Normally, predicted state values should be used since these values are always available, even for unmeasured states. Also, if predicted state values are so bad that using data state values is necessary, it is important to make sure the $v(t)$ identification is good enough to produce a meaningful identification of $v(\mathbf{X}, \omega)$. In the case of the Park-Ramirez model studied here, the parameter functions need to be identified over a substrate concentration of approximately five orders of magnitude ($10^{-3} \frac{\text{g}}{\text{L}} - 10^2 \frac{\text{g}}{\text{L}}$). For very low substrate concentration, even slight errors in the growth rate function will lead to huge relative errors in the substrate concentration, so it was better to use substrate data values for this model.

6.7.3 Neural network domain transformation

Neural networks are best at handling inputs with a domain of $(-1, 1)$ or $(0, 1)$, so another consequence of the large range of substrate concentration used for this model is that the neural networks needed a logarithmically transformed substrate concentration S° . Substrate concentration was transformed so that

$$S = 10^{-3} \Rightarrow S^\circ = 0$$

and

$$S = 10^2 \Rightarrow S^\circ = 1$$

Input neurons	1
Output neurons	1
Hidden neurons (μ)	5
Hidden neurons (f_P)	4
Hidden neurons (ϕ)	3
Input neuron activation function	Linear
Hidden neuron activation function	Logistic sigmoid
Output neuron activation function	Linear
Training epochs	20,000

Table 6.3: Neural network parameters for Park-Ramirez model $v(\mathbf{X}, \omega)$ identification.

using the transformation

$$S^\circ = \frac{\log_{10} S + 3}{5}. \quad (6.11)$$

6.7.4 Neural network training

Once the input-output set is created using all runs, it is used to train the parameter functions $v(\mathbf{X}, \omega)$ with an appropriate identification technique for the particular function estimator used. Feedforward neural networks were used in this study, and the FANN neural network software library (Nissen, 2003) was used to train the neural networks using the parameters in Table 6.3. A naïve application of neural network training was used, without generalization techniques such as validation and early stopping. Although these techniques might improve the parameter function capture somewhat, a better way to improve results may be to use an outlier detection algorithm since the Webb-Ramirez identification method produced many outliers in the $v(\mathbf{X}, \omega)$ training sets shown in the remainder of the chapter.

6.8 Identifying $v(\mathbf{X}, \omega)$ for the Park-Ramirez model

The parameter function neural network identification for $\mu(S^\circ, \omega_1)$, $f_P(S^\circ, \omega_2)$, and $\phi(S^\circ, \omega_3)$ was explored using several scenarios:

Algorithm 10 Algorithm to extract $v(\mathbf{X}, \omega)$ input-output training data from $v(t)$ data.

1. **for** each data run **do**
 2. Initialize the system model
 3. **for** each stage **do**
 4. **for** each parameter function **do**
 5. Create training point using predicted or data state values for inputs and $v(t)$ for output
 6. Replicate training point β_R times
 7. Integrate to the next stage using $v(t)$ parameter function values
 8. **end for**
 9. **end for**
 10. **end for**
-

For this study:

$$\beta_R(\mu) = \text{MIN} \left[1, \text{floor} \left(\beta_A(\mu) \cdot \frac{t_f}{t_f - t} \cdot \text{MAX} [\xi_{\Phi u}^-(\mu), \xi_{\Phi u}^+(\mu)] \right) \right] \quad (6.12)$$

$$\beta_R(f_P) = \text{MIN} \left[1, \text{floor} \left(\beta_A(f_P) \cdot \frac{t_f}{t_f - t} \cdot \text{MAX} [\xi_{\Phi u}^-(f_P), \xi_{\Phi u}^+(f_P)] \right) \right] \quad (6.13)$$

$$\beta_R(\phi) = \text{MIN} \left[1, \text{floor} \left(\beta_A(\phi) \cdot \frac{t_f}{t_f - t} \cdot \text{MAX} [\xi_{\Phi u}^-(\phi), \xi_{\Phi u}^+(\phi)] \right) \right] \quad (6.14)$$

The normalized performance index control sensitivities $\xi_{\Phi u}^-$ and $\xi_{\Phi u}^+$ are described in Section 4.6 (page 39). M_{Φ} in Equations 4.10 and 4.11 is 5×10^{-5} . The training data replication Equations 6.12–6.14 were chosen so that data had 0 or 1 replicates. The term with final time t_f and time t compensates for the fact that earlier data points affect states for more of the simulation. The data replication adjustment term β_A is an arbitrary adjustment term for each parameter function. For this study, $\beta_A(\mu) = 0.001$, $\beta_A(f_P) = 0.08$, and $\beta_A(\phi) = 0.2$. These values were chosen manually by reducing each one until the corresponding data lost most obvious outliers. The floor operator converts a floating point value to the integer value closest to zero.

No P_T Data	
Noiseless Data	Section 6.8.1 (page 129)
Noisy Data	Section 6.8.2 (page 134)
5 P_T Data	
Noiseless Data	Section 6.8.3 (page 136)
Noisy Data	Section 6.8.4 (page 141)
Noisy Data, no Sensitivity	Section 6.8.5 (page 143)
100 P_T Data	
Noiseless Data	Section 6.8.6 (page 145)
Noisy Data	Section 6.8.7 (page 150)
Discontinuous Substrate	Section 6.8.8 (page 155)

Noisy data had 1% noise added as described in Section 6.5 (page 103). The eight control profiles in Figure 6.1 (page 105) and the data generation parameters in Table 6.1 (page 107) were used to generate eight artificial data runs for the cases in this section. The first step of the Webb-Ramirez identification technique was applied to each of the data runs, then the eight runs of identified $v(t)$ data were used to create an input-output data set and each parameter function neural network $v(\mathbf{X}, \omega)$ was trained following the method in Section 6.7.2. Unless otherwise specified, the sensitivity data replication scheme was used during the creation of the input-output neural network training set.

To provide a quantitative way to compare identification quality for different cases, Equation 6.15 was used to give an arbitrary but consistent measure of the goodness of fit between the parameter functions used to generate the data and the identified parameter functions. It is integrated across the logarithmically transformed substrate so that goodness of fit is important across all five orders of magnitude of substrate concentration. This measure is shown for each parameter function in the figures.

$$\text{SSE} = \int_0^1 [v(S^\circ)_{\text{real}} - v(S^\circ)_{\text{model}}]^2 dS^\circ \quad (6.15)$$

6.8.1 Identifying Park-Ramirez $v(\mathbf{X}, \omega)$: 0 P_T data, no noise

Given the results of the $v(t)$ identification without P_T state data in Section 6.6.4 (page 115), the μ and f_P parameter function neural networks were expected to do a reasonably good job of capturing the mechanistic kinetic rate functions with no P_T data, but the ϕ parameter function was not. The first step of the Webb-Ramirez identification is shown for run 1 of the 8 artificial data runs in Figures 6.14, 6.15, and 6.16 for the 10, 25, and 50 stages IDP cases respectively. Figure 6.17 shows the original parameter function values, the input-output training data from the process described in Section 6.7.2 (page 6.7.2), and the trained parameter function neural networks. The growth rate neural network $\mu(S^\circ, \omega_1)$ captured the mechanistic kinetic rate function very well because growth rate does not depend on P_T data, but surprisingly, f_P is also captured very well, probably because P_M follows P_T very closely most of the time. As expected, the unresolved dependency between f_P and ϕ severely degraded the identification of ϕ .

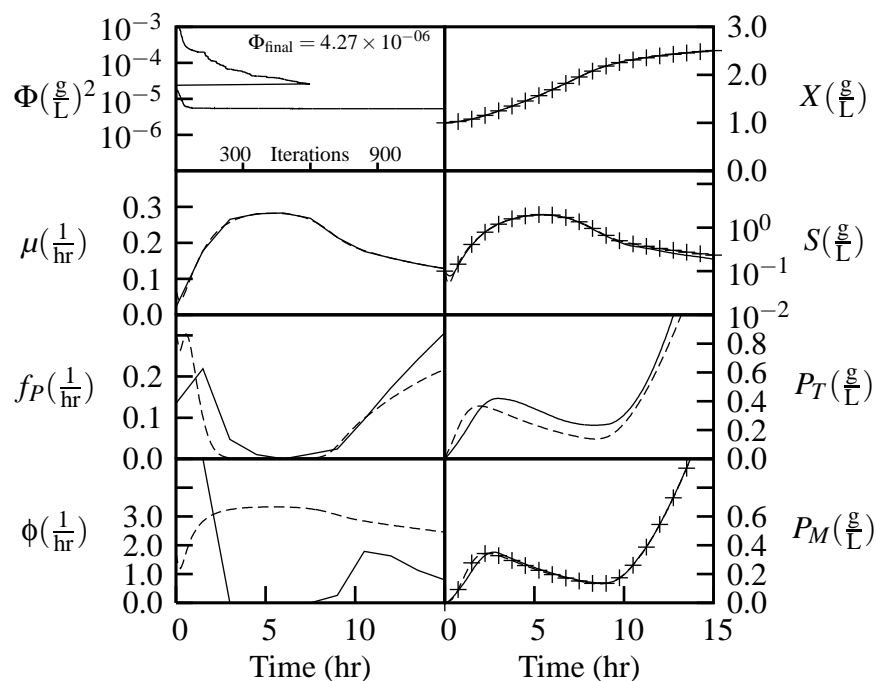


Figure 6.14: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , and P_M using two-step IDP with **10 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **0 P_T data** and $\sigma_N = \mathbf{0}$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

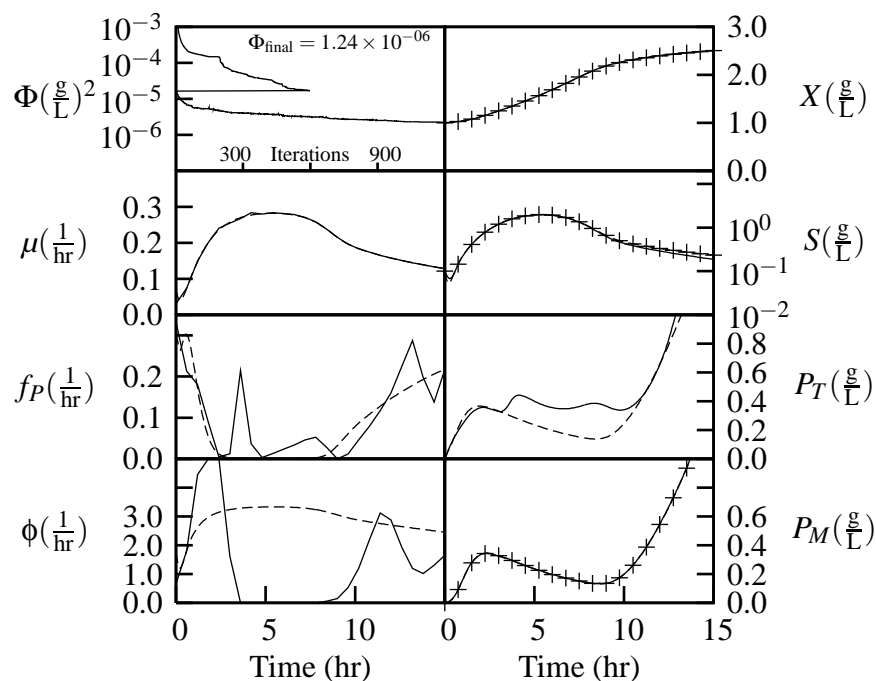


Figure 6.15: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , and P_M using two-step IDP with **25 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **0 P_T data** and $\sigma_N = \mathbf{0}$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

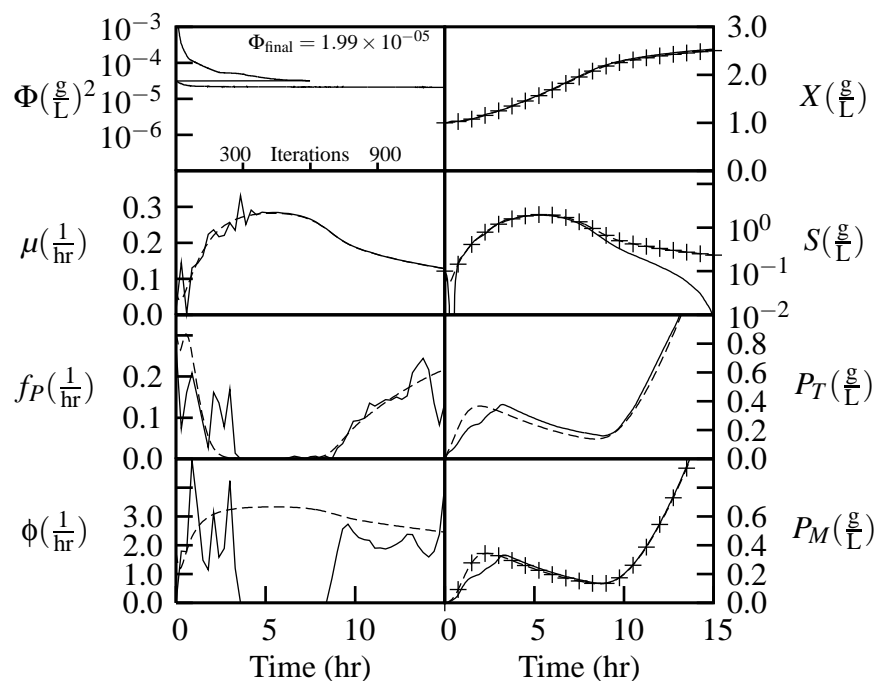


Figure 6.16: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , and P_M using two-step IDP with **50 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **0 P_T data** and $\sigma_N = \mathbf{0}$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

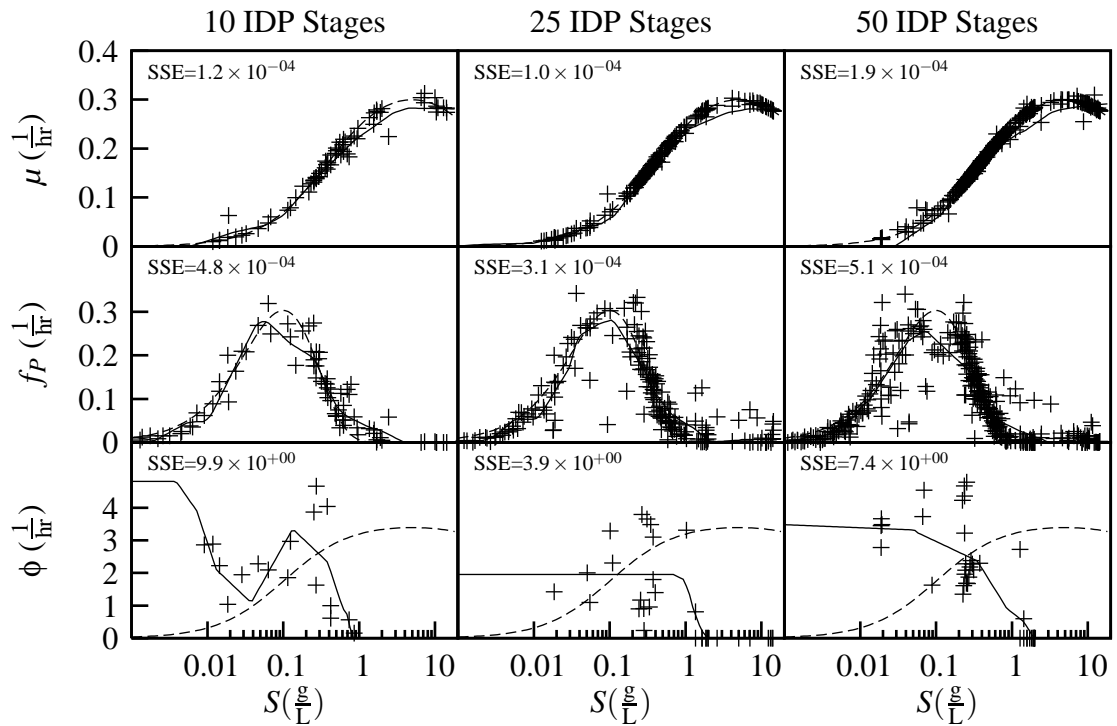


Figure 6.17: Hybrid Park-Ramirez model parameter function neural network identification. The data generation and IDP parameters used for the $v(t)$ identification are given in Table 6.1 (page 107) and 6.2 (page 111) with $\mathbf{0 P_T}$ data and $\sigma_N = \mathbf{0}$. The dashed lines are the mechanistic parameter functions (Equations 2.3–2.5). Neural network training data from Algorithm 10 are marked with + symbols, and the solid lines are the neural network outputs. SSE is a measure of the goodness of fit between the mechanistic parameter functions and the identified functions, and is defined by Equation 6.15 (page 128).

6.8.2 Identifying Park-Ramirez $v(\mathbf{X}, \omega)$: $0 P_T$ data, with noise

This section repeats the case in Section 6.8.1 except that the artificial data was generated using $\sigma_N = 0.01$ instead of $\sigma_N = 0$. The first step of the Webb-Ramirez identification for run 1 of the 8 artificial data runs is the same as the case in Section 6.6.4 (Figures 6.8–6.10 starting on page 117), and the $v(\mathbf{X}, \omega)$ input-output data and training is shown in Figure 6.18.

The $v(\mathbf{X}, \omega)$ identification results for the case with noise are similar to the noiseless case in the previous section. The 50 stage case is most affected, which is not surprising since the 50 stage case is more prone to noise in general.

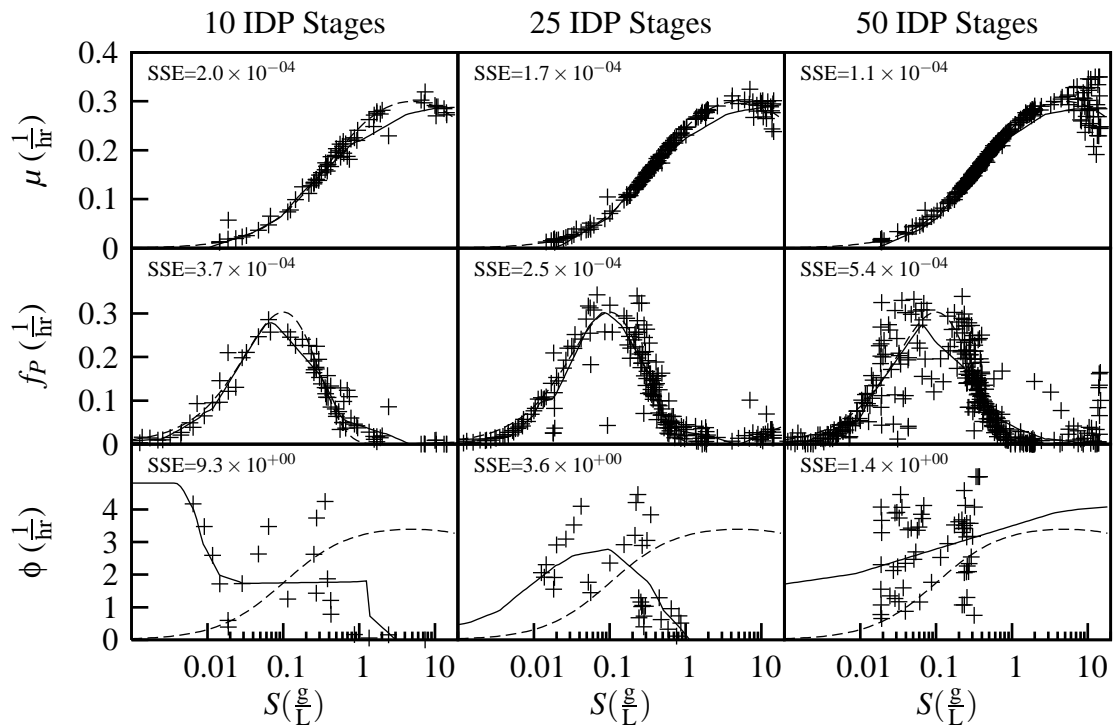


Figure 6.18: Hybrid Park-Ramirez model parameter function neural network identification. The data generation and IDP parameters used for the $v(t)$ identification are given in Table 6.1 (page 107) and 6.2 (page 111) with $\mathbf{0} \mathbf{P}_T$ data and $\sigma_N = \mathbf{0.01}$. The dashed lines are the mechanistic parameter functions (Equations 2.3–2.5). Neural network training data from Algorithm 10 are marked with + symbols, and the solid lines are the neural network outputs. SSE is a measure of the goodness of fit between the mechanistic parameter functions and the identified functions, and is defined by Equation 6.15 (page 128).

6.8.3 Identifying Park-Ramirez $v(\mathbf{X}, \omega)$: 5 P_T data, no noise

This section repeats the case from Section 6.8.1 except with five P_T data per run instead of none. The results of the $v(t)$ identification for run 1 of the 8 artificial data runs are shown in Figures 6.19, 6.20, and 6.21 for the 10, 25, and 50 stages IDP cases respectively, and the $v(\mathbf{X}, \omega)$ training is shown in Figure 6.22.

As expected, the small amount of P_T data dramatically improved the capture of the $\phi(S^\circ, \omega_3)$ neural network. Although the capture is not perfect, it is reasonably good considering only 8 data runs were used and the large range of substrate concentration captured by the model. One remaining deficiency in the identification is that ϕ for $S > 1 \frac{\text{g}}{\text{L}}$ is not captured because there is no training data in that region.

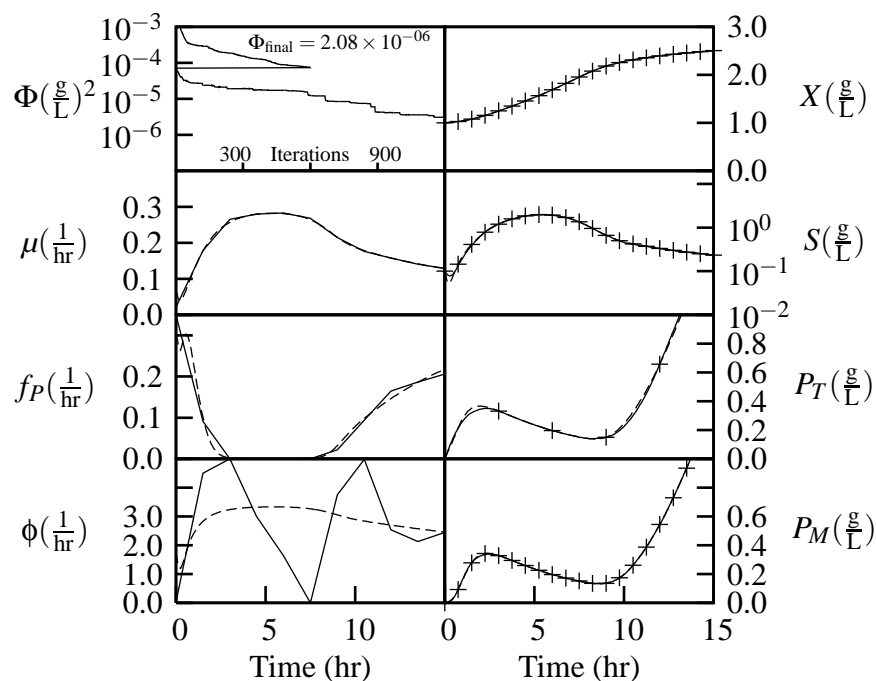


Figure 6.19: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , P_T , and P_M using two-step IDP with **10 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **5 P_T data** and $\sigma_N = \mathbf{0}$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

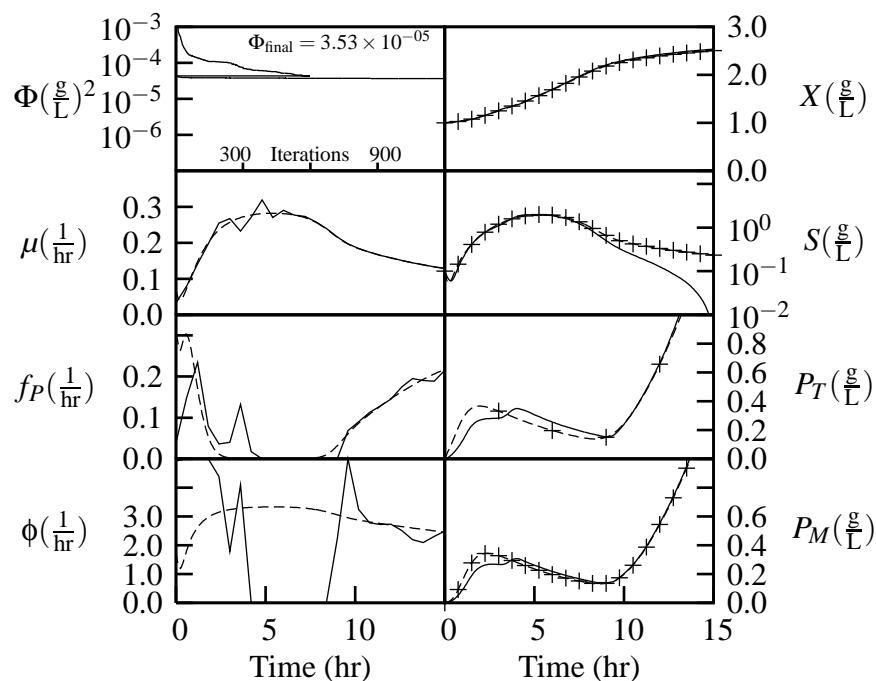


Figure 6.20: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , P_T , and P_M using two-step IDP with **25 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **5 P_T data** and $\sigma_N = \mathbf{0}$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

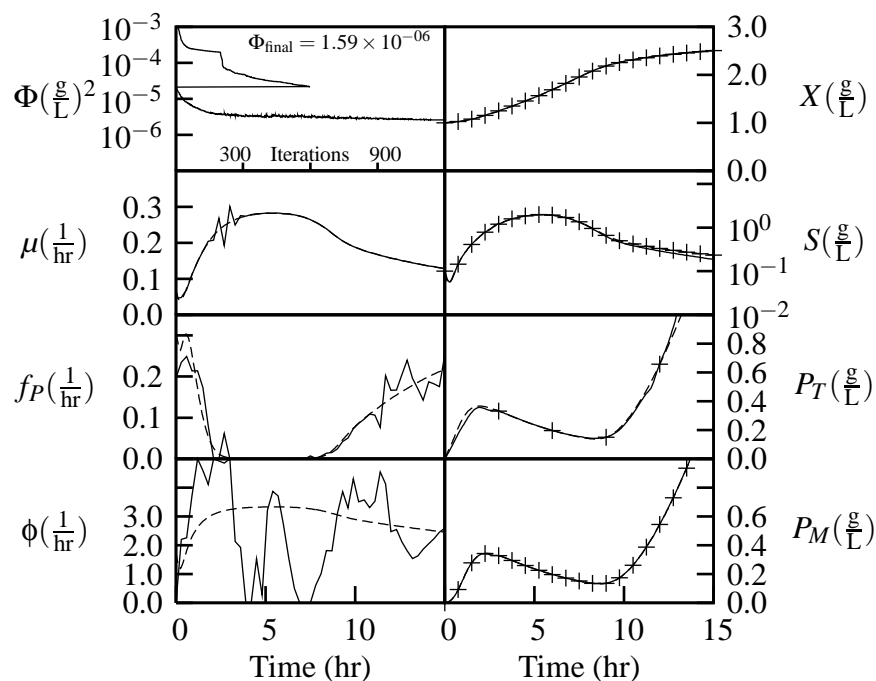


Figure 6.21: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , P_T , and P_M using two-step IDP with **50 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **5 P_T data** and $\sigma_N = \mathbf{0}$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

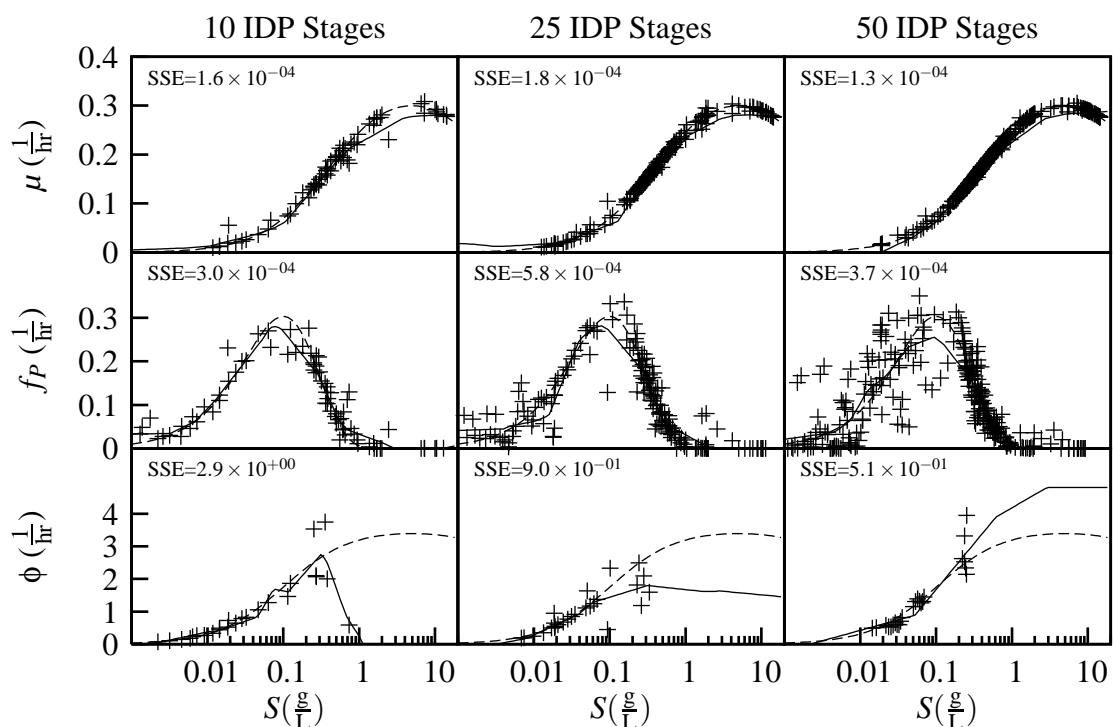


Figure 6.22: Hybrid Park-Ramirez model parameter function neural network identification. The data generation and IDP parameters used for the $v(t)$ identification are given in Table 6.1 (page 107) and 6.2 (page 111) with **5 P_T data** and $\sigma_N = \mathbf{0}$. The dashed lines are the mechanistic parameter functions (Equations 2.3–2.5). Neural network training data from Algorithm 10 are marked with + symbols, and the solid lines are the neural network outputs. SSE is a measure of the goodness of fit between the mechanistic parameter functions and the identified functions, and is defined by Equation 6.15 (page 128).

6.8.4 Identifying Park-Ramirez $v(\mathbf{X}, \omega)$: 5 P_T data, with noise

This section repeats the study of Section 6.8.3 except that the artificial data was generated using $\sigma_N = 0.01$ instead of $\sigma_N = 0$. The first step of the Webb-Ramirez identification for run 1 of the 8 artificial data runs is the same as the case in Section 6.6.5 (Figures 6.11–6.13 starting on page 121), and the $v(\mathbf{X}, \omega)$ input-output data and training is shown in Figure 6.23.

For the 5 P_T cases, noise seems to degrade the training data slightly for the 25 and 50 stage cases, but makes no noticeable difference in the 10 stage case. In general, these results suggest that fewer IDP stages should be used as data noise increases.

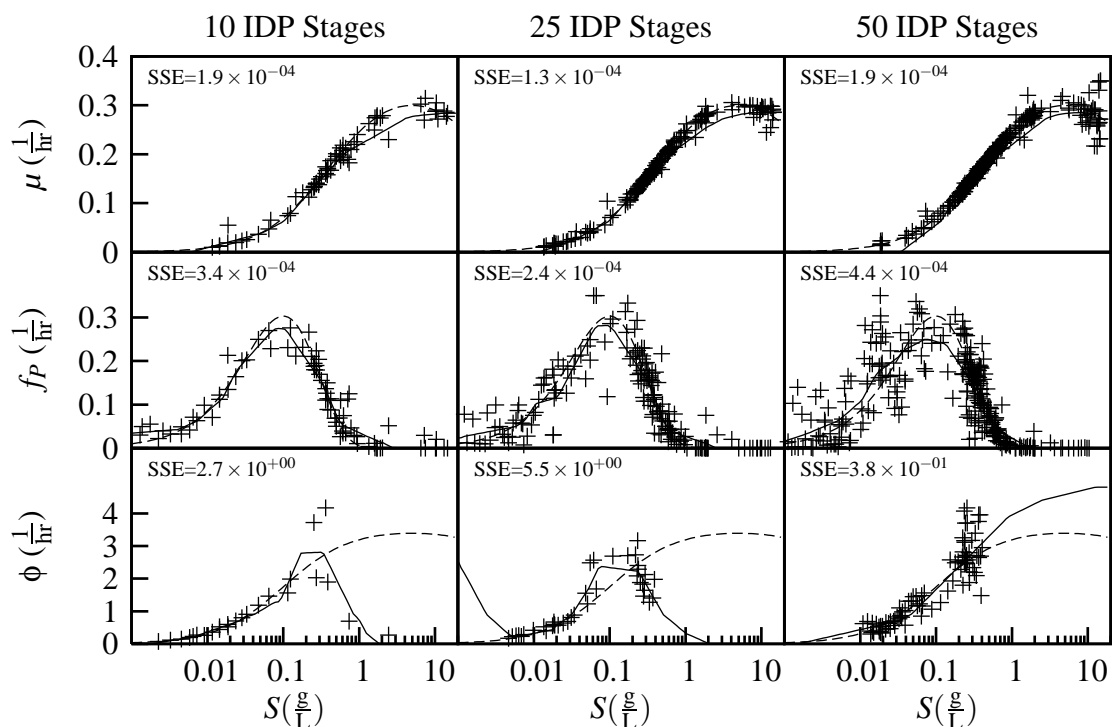


Figure 6.23: Hybrid Park-Ramirez model parameter function neural network identification. The data generation and IDP parameters used for the $v(t)$ identification are given in Table 6.1 (page 107) and 6.2 (page 111) with **5 P_T data** and $\sigma_N = \mathbf{0.01}$. The dashed lines are the mechanistic parameter functions (Equations 2.3–2.5). Neural network training data from Algorithm 10 are marked with + symbols, and the solid lines are the neural network outputs. SSE is a measure of the goodness of fit between the mechanistic parameter functions and the identified functions, and is defined by Equation 6.15 (page 128).

6.8.5 Identifying Park-Ramirez $v(\mathbf{X}, \omega)$: 5 P_T data, with noise, no sensitivity

To see the difference made by the sensitivity data replication scheme in Algorithm 10, the neural network training for the case in the previous section (Figure 6.23) was repeated using one data replicate for all data points in Algorithm 10 ($\beta_R = 1$). The neural network training for this case is shown in Figure 6.24.

The μ and f_P data and neural network appear to be similar whether or not sensitivity data replication is used, but the ϕ identification changes drastically. The points missing from the ϕ training set were verified to be from parts of the run data where secretion was low, so this method appears to accomplish the same effect as an analytical sensitivity would.

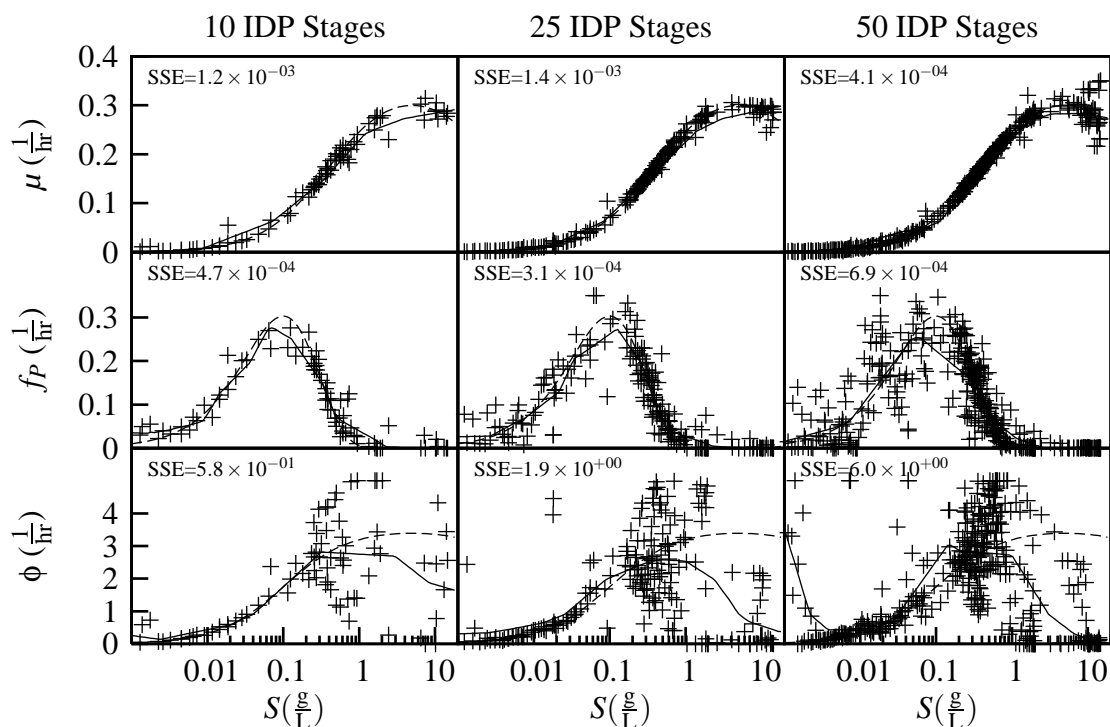


Figure 6.24: Hybrid Park-Ramirez model parameter function neural network identification without sensitivity data replication. The data generation and IDP parameters used for the $v(t)$ identification are given in Table 6.1 (page 107) and 6.2 (page 111) with **5 \mathbf{P}_T data** and $\sigma_N = \mathbf{0.01}$. The dashed lines are the mechanistic parameter functions (Equations 2.3–2.5). Neural network training data from Algorithm 10 are marked with + symbols, and the solid lines are the neural network outputs. SSE is a measure of the goodness of fit between the mechanistic parameter functions and the identified functions, and is defined by Equation 6.15 (page 128).

6.8.6 Identifying Park-Ramirez $v(\mathbf{X}, \omega)$: 100 P_T data, no noise

This section repeats the case from Section 6.8.3 except with 100 P_T data per run instead of 5. The results of the $v(t)$ identification for run 1 of the 8 artificial data runs are shown in Figures 6.25, 6.26, and 6.27 for the 10, 25, and 50 stages IDP cases respectively, and the $v(\mathbf{X}, \omega)$ training is shown in Figure 6.28.

The $f_P(t)$ and $\phi(t)$ identifications are nearly perfect when using 100 P_T data, leading to $v(\mathbf{X}, \omega)$ identification that is nearly perfect as well. The exceptions are a little error in the 10 stage case, a few outliers in the $v(\mathbf{X}, \omega)$ training sets, and the lack of ϕ data for $S > 1\frac{g}{L}$.

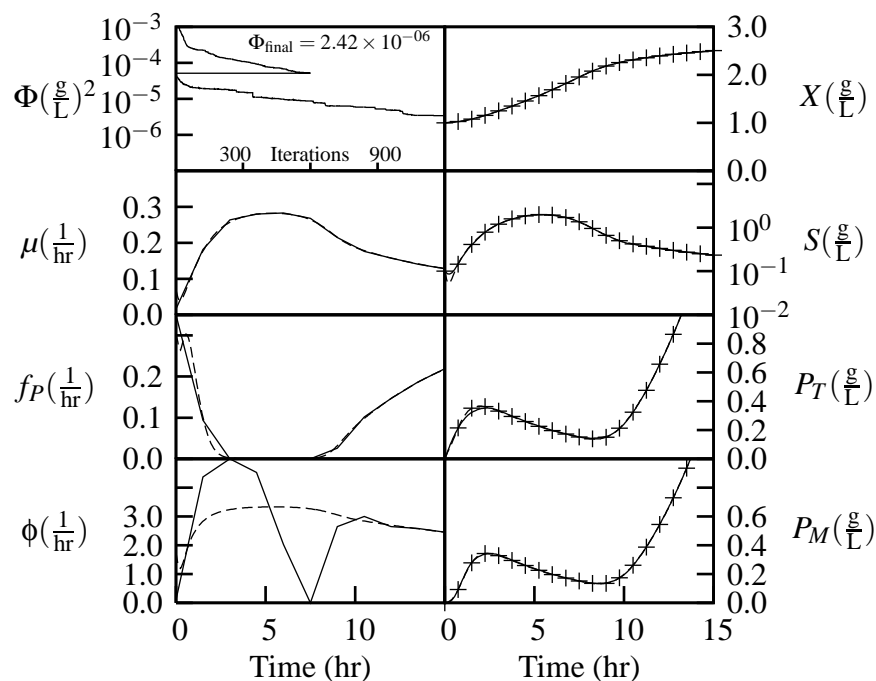


Figure 6.25: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , P_T , and P_M using two-step IDP with **10 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **100 P_T data** and $\sigma_N = \mathbf{0}$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , P_T , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

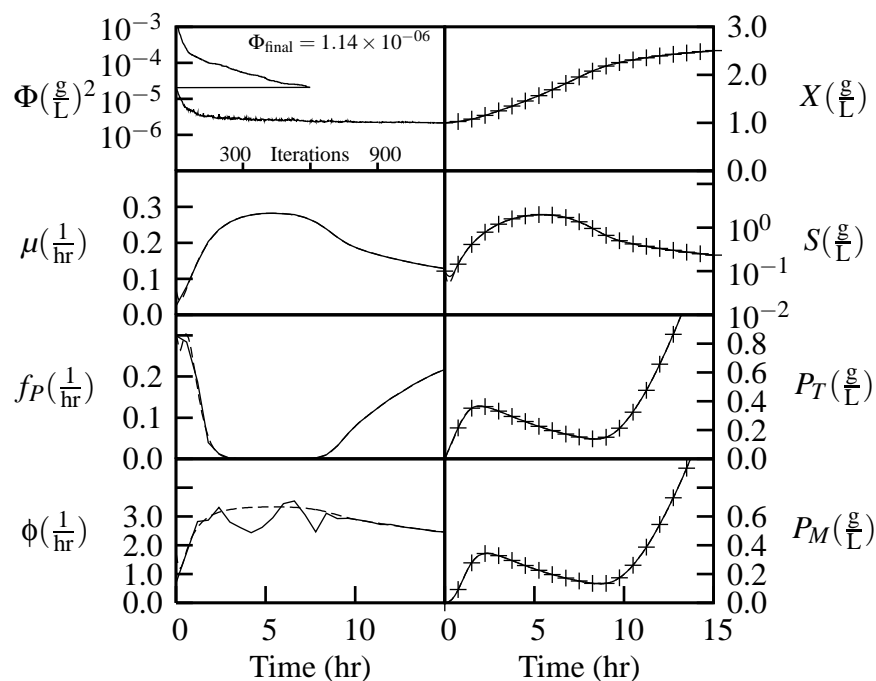


Figure 6.26: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , P_T , and P_M using two-step IDP with **25 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **100 P_T data** and $\sigma_N = \mathbf{0}$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , P_T , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

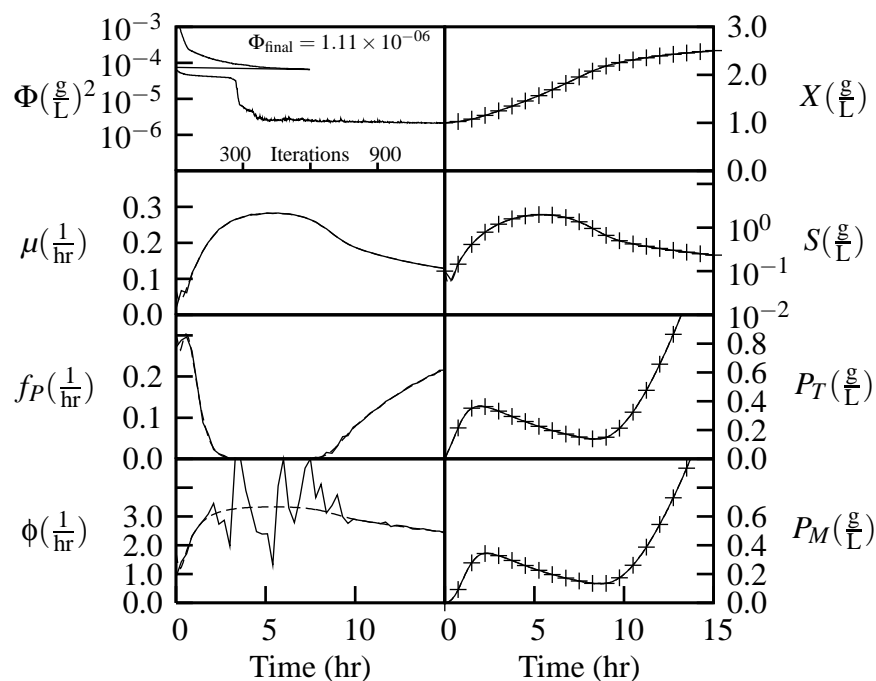


Figure 6.27: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , P_T , and P_M using two-step IDP with **50 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **100 P_T data** and $\sigma_N = \mathbf{0}$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , P_T , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

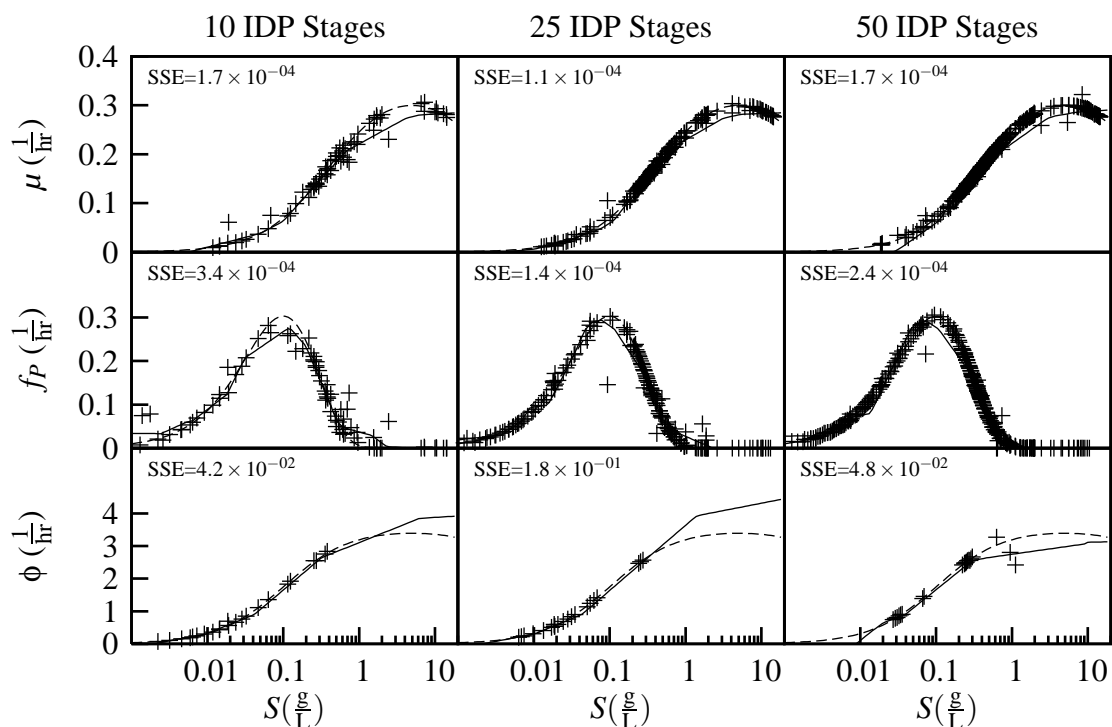


Figure 6.28: Hybrid Park-Ramirez model parameter function neural network identification. The data generation and IDP parameters used for the $v(t)$ identification are given in Table 6.1 (page 107) and 6.2 (page 111) with **100 P_T data** and $\sigma_N = 0$. The dashed lines are the mechanistic parameter functions (Equations 2.3–2.5). Neural network training data from Algorithm 10 are marked with + symbols, and the solid lines are the neural network outputs. SSE is a measure of the goodness of fit between the mechanistic parameter functions and the identified functions, and is defined by Equation 6.15 (page 128).

6.8.7 Identifying Park-Ramirez $v(\mathbf{X}, \omega)$: 100 P_T data, with noise

This section repeats the study of Section 6.8.6 except that the artificial data was generated using $\sigma_N = 0.01$ instead of $\sigma_N = 0$. The first step of the Webb-Ramirez identification is shown for run 1 of the 8 artificial data runs in Figures 6.29, 6.30, and 6.31 for the 10, 25, and 50 stages IDP cases respectively, and the $v(\mathbf{X}, \omega)$ input-output data and training is shown in Figure 6.32. The results are nearly the same as the noiseless case in the previous section, although the noise slightly affects the $\phi(t)$ identification when 50 IDP stages are used.

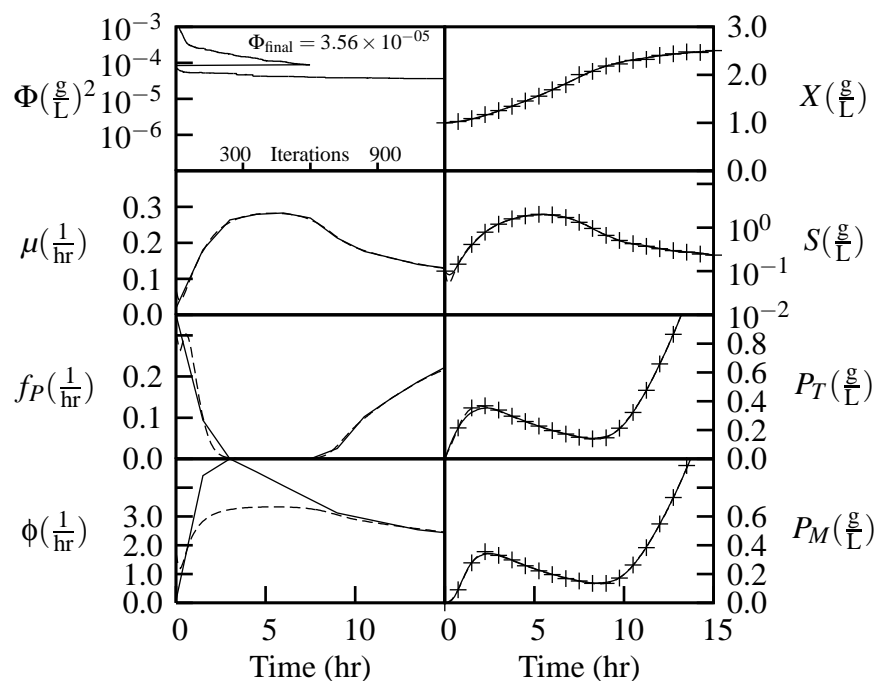


Figure 6.29: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , P_T , and P_M using two-step IDP with **10 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **100 P_T data** and $\sigma_N = 0.01$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , P_T , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

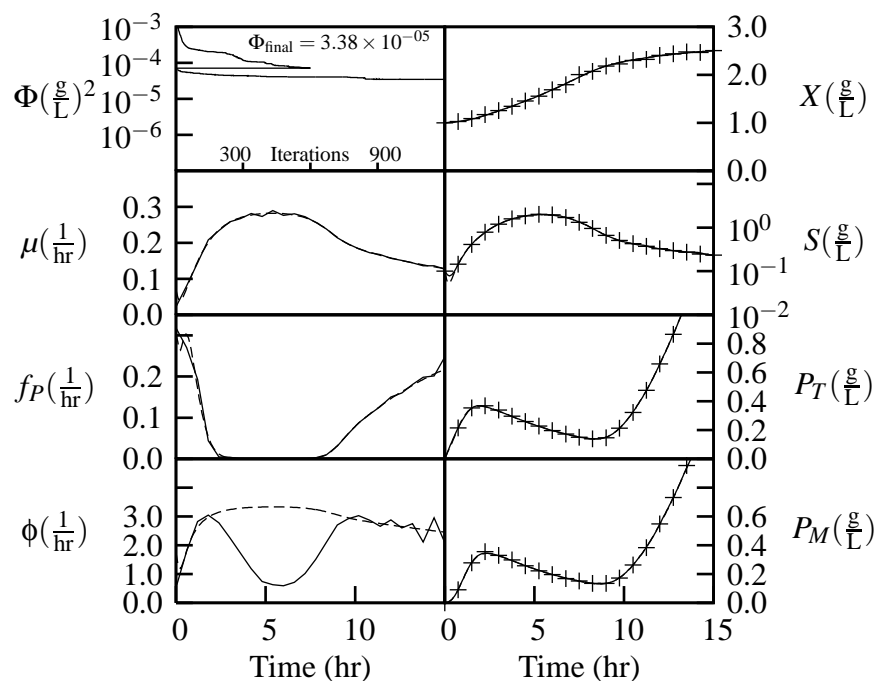


Figure 6.30: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , P_T , and P_M using two-step IDP with **25 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **100 P_T data** and $\sigma_N = 0.01$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , P_T , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

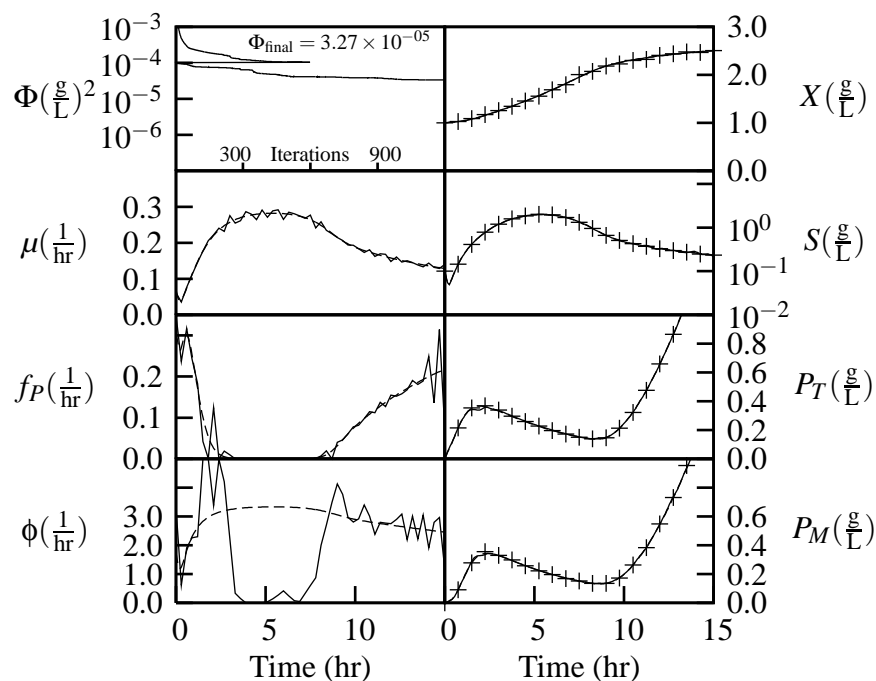


Figure 6.31: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , P_T , and P_M using two-step IDP with **50 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **100 P_T data** and $\sigma_N = 0.01$. Control q^1 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , P_T , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

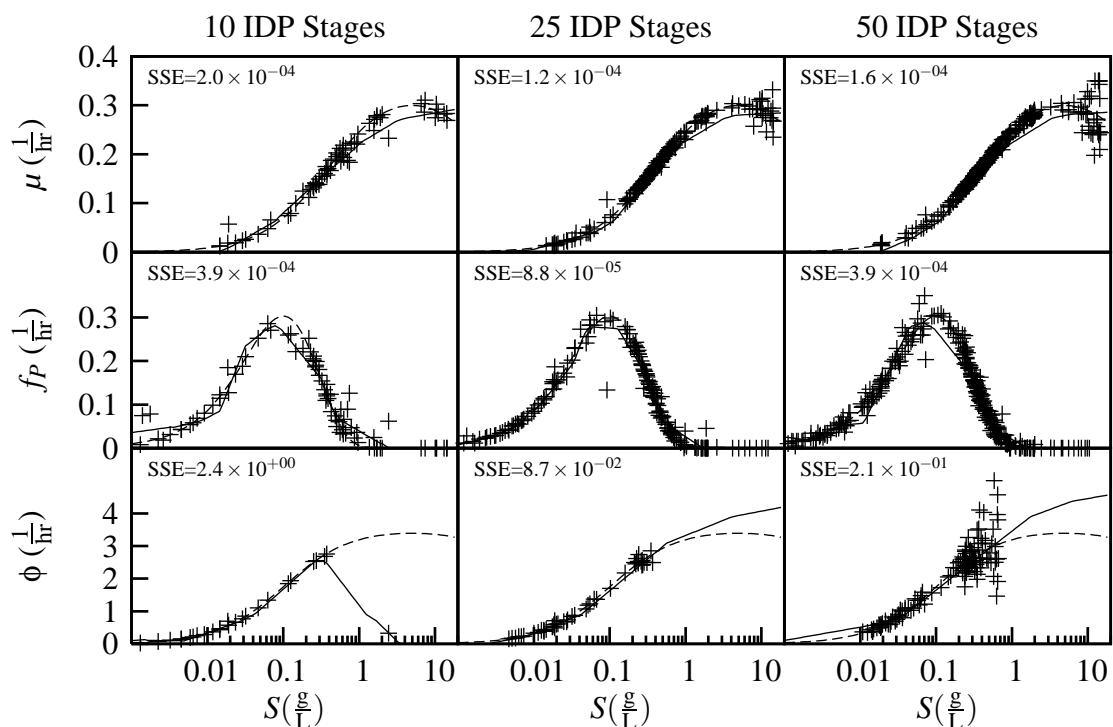


Figure 6.32: Hybrid Park-Ramirez model parameter function neural network identification. The data generation and IDP parameters used for the $v(t)$ identification are given in Table 6.1 (page 107) and 6.2 (page 111) with **100 P_T data** and $\sigma_N = \mathbf{0.01}$. The dashed lines are the mechanistic parameter functions (Equations 2.3–2.5). Neural network training data from Algorithm 10 are marked with + symbols, and the solid lines are the neural network outputs. SSE is a measure of the goodness of fit between the mechanistic parameter functions and the identified functions, and is defined by Equation 6.15 (page 128).

6.8.8 Substrate discontinuity for improved $v(\mathbf{X}, \omega)$ identification

It is clear from the previous results in Section 6.8 that ϕ is not being identified well for $S > 1 \frac{\text{g}}{\text{L}}$. It is easy to understand why: if S is high, f_P falls near zero, and thus any secretion while S is high only lasts until the existing internal protein is secreted. The time constant for this secretion is around 10 minutes, so it is difficult to capture this fleeting measurement by raising S using the maximum substrate feed rate.

A better solution is to cause a step change in S to a high concentration, which allows a short time where S is high and secretion is happening. This can be accomplished for the experimental system by dumping a measured amount of substrate manually into the tank. In the simulation, the model function was modified to manually set S once each run at $t = 14.7\text{hr}$. The time 14.7hr was chosen because that is the start of the last stage for 50 stages. Only the 50 stage case was examined for this technique because 10 or 25 stages are not able to capture the very fast dynamics. To get data for a range of substrate concentration, the substrate at 14.7 hr was set to $S = (z + 2) \frac{\text{g}}{\text{L}}$ where z is the run index from Figure 6.1 (page 105). The cases in this section were generated in the same way as previous identification cases except with this manual substrate discontinuity.

Figure 6.33 presents the case with noiseless artificial data, and Figure 6.34 presents the case with noisy artificial data. For both the noiseless and the noisy cases, ϕ was identified well only with 100 P_T data. This makes sense; the substrate increase results in a short-lived dynamic where the difference between P_T and P_M is crucial, so sufficient P_T and P_M data are a prerequisite. It is hard to draw conclusions about the effect of noise, because significant random variation was seen for this case depending on initial conditions or random number seed.

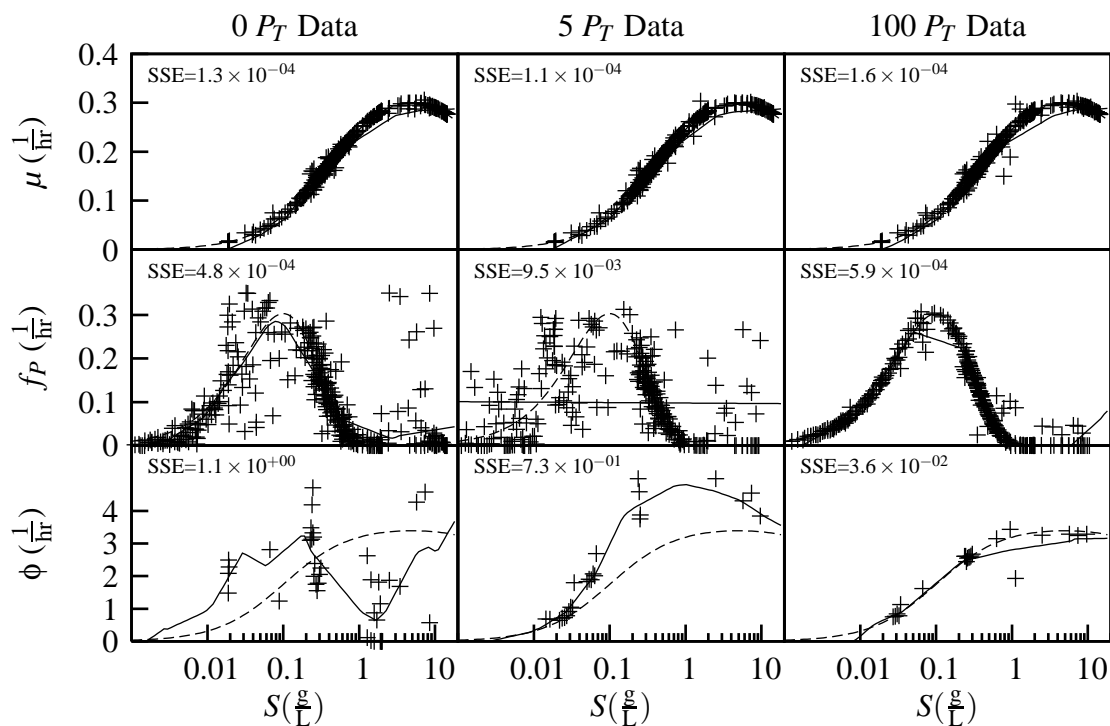


Figure 6.33: Hybrid Park-Ramirez model parameter function neural network identification with manual substrate discontinuity. The data generation and IDP parameters used for the $v(t)$ identification are given in Table 6.1 (page 107) and 6.2 (page 111) with **0, 5 or 100 P_T data** and $\sigma_N = \mathbf{0}$, with the exception that S was increased manually to $z + 2 \frac{g}{L}$ where z is the run index. The dashed lines are the mechanistic parameter functions (Equations 2.3–2.5). Neural network training data from Algorithm 10 are marked with $+$ symbols, and the solid lines are the neural network outputs. SSE is a measure of the goodness of fit between the mechanistic parameter functions and the identified functions, and is defined by Equation 6.15 (page 128).

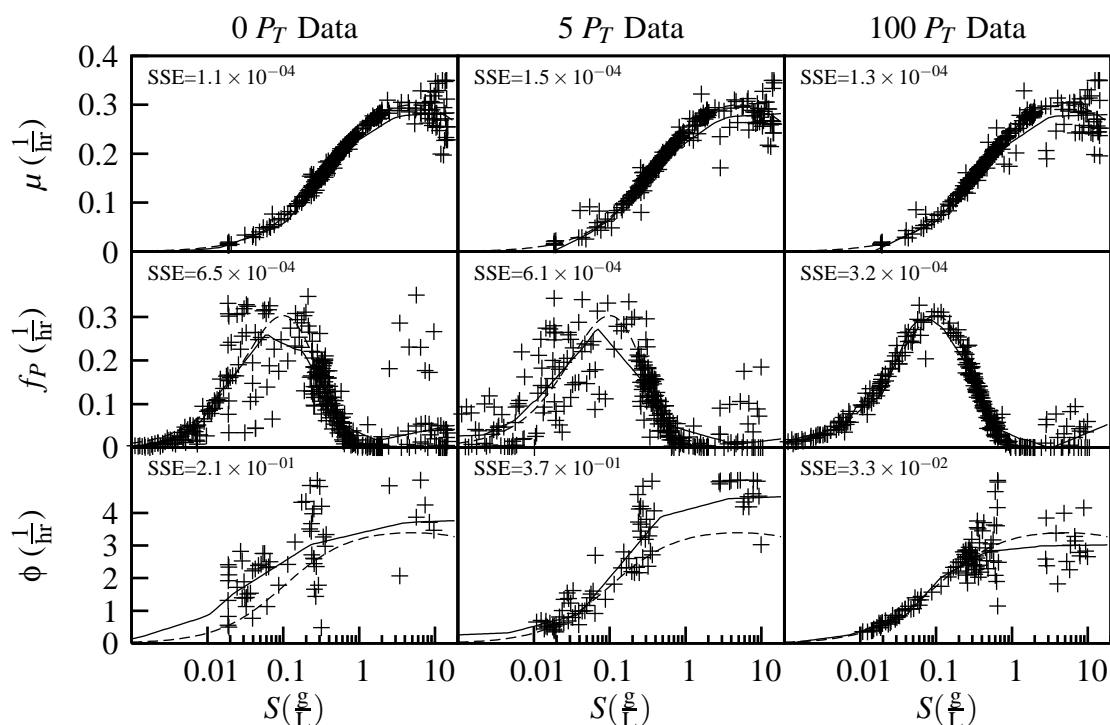


Figure 6.34: Hybrid Park-Ramirez model parameter function neural network identification with manual substrate discontinuity. The data generation and IDP parameters used for the $v(t)$ identification are given in Table 6.1 (page 107) and 6.2 (page 111) with **0, 5 or 100 P_T data** and $\sigma_N = 0.01$, with the exception that S was increased manually to $z + 2\frac{g}{L}$ where z is the run index. The dashed lines are the mechanistic parameter functions (Equations 2.3–2.5). Neural network training data from Algorithm 10 are marked with $+$ symbols, and the solid lines are the neural network outputs. SSE is a measure of the goodness of fit between the mechanistic parameter functions and the identified functions, and is defined by Equation 6.15 (page 128).

6.8.9 Optimal control of the identified hybrid model

One test of the quality of the identification in this chapter is to see if the optimal control for the identified model is the same as the optimal control for the model used to generate the artificial data. Figure 6.35 presents the optimal control found using IDP with 50 stages for the mechanistic model (Equations 2.3–2.5 starting on page 7) and the identified model corresponding to the 100 P_T data case in Figure 6.33. The optimal feed rate for the identified model is lower, and the final performance index is significantly lower. This difference is explained by the fact that the identified μ and f_P are slightly lower than the mechanistic versions. A modified model was created by increasing μ and f_P by 8%, leading to a much closer match to the expected optimal control. Because the kinetic parameters are integrated, errors in the kinetic parameters are amplified when finding an optimal control based on a final measure such as the total secreted protein production. This indicates that effort spent improving the $v(\mathbf{X}, \omega)$ identification from $v(t)$ data will probably be a good investment. In this case a better fit would probably have been obtained if outlier detection and more sophisticated neural network training had been used. Figure 6.36 presents the same results as Figure 6.35 for the case with noisy artificial data (Figure 6.34), with no significant differences in the result.

Another test of the quality of the identification in this chapter is to apply the optimal control for the identified model to the mechanistic model used to generate the artificial data. This is what would happen in practice if this method were used since the real model would not be known. When $q_{\text{identified}}$ from Figure 6.35 was applied to the mechanistic Park-Ramirez model, the performance index was 23.2, and when $q_{\text{identified}}$ from Figure 6.36 was applied to the mechanistic Park-Ramirez model, the performance index was 24.0. Once again, a small error in the f_P and ϕ parameter functions caused significant harm to the goal of optimizing the system. This kind of problem is not unique to this method, though. Any system where the objective is highly sensitive to the

values of the model parameters will be difficult to optimize offline. In reality, almost any system complex enough to need the kind of advanced modeling work discussed here will also need to use online model predictive control instead of offline optimal control to achieve the best results.

Although the identified optimal control does not perfectly match the mechanistic model optimal control, it is close enough that the same same four control features are seen: an exponential phase where growth is maximized, a drop to zero feed while substrate concentration drops, a constant feed phase where protein production is maximized, and finally a short time at maximum feed rate to stimulate secretion. Because the basic features of the real optimal control were seen and the kinetic parameters were relatively close to the real values, it should be expected that model predictive control of the real system using the identified model would have very good results.

6.9 Discussion

6.9.1 Identification methods

This study introduced a new more general method for identifying hybrid dynamic models, and verified that it can identify the hybrid Park-Ramirez model using only a few data runs. The Webb-Ramirez identification method has obvious value in situations where the limitations of the derivative and integral methods preclude their use, but more research is needed to make evidence-based recommendations about which method is superior for any given problem.

For problems where it can be used, the integral method will probably be the best method. It should be the least affected by state noise since it trains on all runs simultaneously, and it is very efficient. There is also no need with the integral method for sensitivity analysis as in Section 6.7.1.

There still may be two advantages to using a two-step identification method even

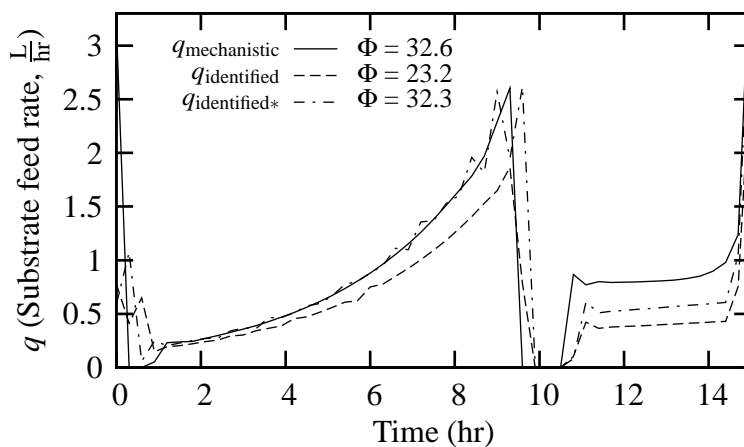


Figure 6.35: Optimal substrate feed rate for the hybrid Park-Ramirez model found using IDP with 50 stages. Model parameters are given in Table 2.1 (page 8). $q_{\text{mechanistic}}$ is the optimal substrate feed rate found when using the mechanistic kinetic rate parameters (Equations 2.3–2.5 starting on page 7). $q_{\text{identified}}$ is the optimal substrate feed rate found when using the parameter function neural networks identified in the 100 P_T data case in Figure 6.33. $q_{\text{identified}^*}$ is the optimal substrate feed rate found when using the same model as $q_{\text{identified}}$ except that μ and f_P are increased by 8%. The theoretical optimal Φ is approximately 32.7, and is not shown because it overlaps $q_{\text{mechanistic}}$ so closely.

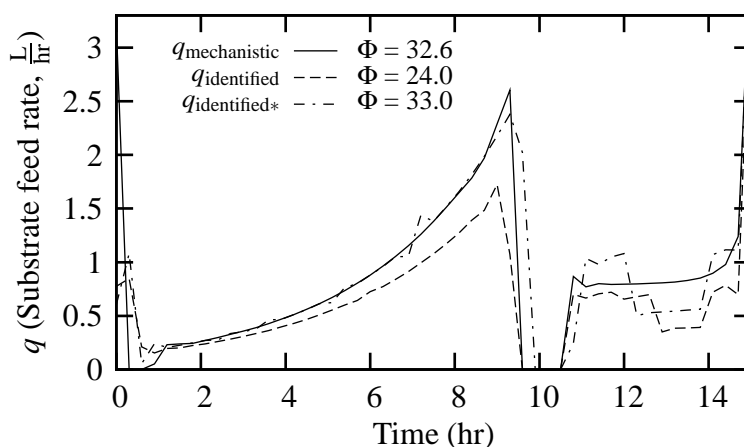


Figure 6.36: Optimal substrate feed rate for the hybrid Park-Ramirez model found using IDP with 50 stages. Model parameters are given in Table 2.1 (page 8). $q_{\text{mechanistic}}$ is the optimal substrate feed rate found when using the mechanistic kinetic rate parameters (Equations 2.3–2.5 starting on page 7). $q_{\text{identified}}$ is the optimal substrate feed rate found when using the parameter function neural networks identified in the 100 P_T data case in Figure 6.34. $q_{\text{identified}^*}$ is the optimal substrate feed rate found when using the same model as $q_{\text{identified}}$ except that μ and f_P are increased by 8%. The theoretical optimal Φ is approximately 32.7, and is not shown because it overlaps $q_{\text{mechanistic}}$ so closely.

if the integral method is possible: first, run-to-run variation of parameter function profiles may be observable that would not be observable when training ω directly, especially in the case of outliers. Second, it may be better to break the large problem into several smaller problems. This is similar to the principle of dynamic programming that it is better to break a large sequential decision problem into smaller problems and solve them separately. Because IDP can be used to find the values of $v(t)$, it may be less likely that the final solution will result in local minima, even with complex interactions between the different v . Both of these hypotheses are left for future study.

Finally, the derivative method has the compelling advantage of simplicity. For this one simple reason, it should probably be the first method attempted for any problem where all the states are measured.

6.9.2 State data noise

Data noise is a big problem for the Webb-Ramirez identification method with two-step IDP since it caused a significant amount of error in the $v(t)$ values. The state data noise seems to lead to numerous local minima very close to the optimal control, a relatively uncommon case for optimal control problems. This affects the integral identification method less than the Webb-Ramirez identification method even though the objective function is the same, because the integral method uses all runs simultaneously and noise is reduced through averaging. Some potential solutions to the state noise problem are presented in Section 9.2 as a topic for future research.

Chapter 7

The Flux-balance bioreactor model

7.1 Introduction

A number of protein pharmaceuticals are currently being produced using recombinant DNA technology. By adding DNA plasmids to a host bacterium, the genetic code is modified, which allows production of large amounts of foreign protein. This method is important because for many complex proteins a direct chemical synthesis is either impossible or more expensive than production using a recombinant bacterial cultivation. There are several aspects of bacterial cultivation that are difficult and expensive. Some major concerns for industrial cultivation are optimizing product yield, reducing product variability, and providing a good on-line control system. Because pilot-plant experiments are expensive, these goals should be accomplished with as few test cultivations as possible. All these goals suggest that a good predictive model of the system is desirable.

A good model for this application should have the following properties:

- It should predict the system states moderately well without experimental data. This allows the model to be used even for early development when data is minimal.
- It should be simple enough that it can be used for model predictive control.

- It should contain as much fundamental and experimental knowledge as possible. An example of fundamental knowledge is conservation principles, which always apply. An example of experimental knowledge is the behavior of the overflow metabolism of *E. coli* reported by many researchers (Chang et al., 1999).

We propose a model that meets these requirements by combining concepts presented in several previous works, and test this model using data from an industrial pilot plant. As much as possible, it is based on literature values of simple kinetic parameters such as the Monod growth parameters. In addition, an ideal model would improve as more data become available, which would only be possible by incorporating this model into a hybrid model with a data-driven component.

7.2 Model Development

There have been many cell growth models presented in the literature (Domach et al., 2000; Ramirez and Bentley, 1999; Tholudur and Ramirez, 1999; Schubert et al., 1994). The goal of this research is to combine and improve on past modeling approaches to be useful for industrial cultivation.

7.2.1 Overall model structure and state equations

Our approach is to start with a simple mass conservation model that is similar to past approaches (Ramirez and Bentley, 1999; Tholudur and Ramirez, 1996; Lin et al., 2001; Xu et al., 1999). The component mass balances reflect that mass is conserved for any physical component. In general, the terms of the state differential equations are accumulation, generation (such as the cell growth rate), input (glucose flow rate to the reactor) and the dilution effect (D). Equation 7.1 defines differential equations for each of the mass fraction states in the model: cell mass fraction (X), dead cell mass fraction

(X_d), glucose mass fraction (S), acetate mass fraction (A), foreign protein mass fraction (P), and oxygen mass fraction (O). The full symbol list for Equations 7.1 and 7.2 is given in Table 7.1.

$$\begin{aligned}
 \text{Cells: } \dot{X} &= (\mu_X - \mu_{X,d} - D) \cdot X \\
 \text{Dead Cells: } \dot{X}_d &= \mu_{X,d} \cdot X - D \cdot X_d \\
 \text{Glucose: } \dot{S} &= -q_S \cdot X + \frac{\dot{f}_S C_S}{T} - D \cdot S \\
 \text{Acetate: } \dot{A} &= (\mu_A - q_A) \cdot X - D \cdot A \\
 \text{Protein: } \dot{P} &= \mu_P \cdot X - D \cdot P \\
 \text{Oxygen: } \dot{O} &= (O_{\text{sat}} - O) \cdot k_O - q_O \cdot X - D \cdot O
 \end{aligned} \tag{7.1}$$

Equation 7.2 is the differential equation for the system total mass (T). Temperature and pH are assumed constant in this model.

$$\text{TotalMass: } \dot{T} = \dot{f}_S \tag{7.2}$$

Cell consumption rates are denoted by q_i , while cell production rates are denoted by μ_i , where i is replaced by a subscript representing the particular state. The units of all the q_i and μ_i rates are mass of i per mass cells per time. Initial conditions are chosen using data.

7.2.2 Kinetic Model

The accumulation, input and dilution rates are straightforward, so the complexity of the model is contained in the generation terms. The model differential equations are derived from mass conservation principles, but the cell consumption and production rates must be obtained from a fundamental understanding of the process, or estimated using a data-driven technique such as neural networks.

The first widely-used model for cell growth rate was reported by Monod. This

$X, X_d, S, A, P,$ O	system concentration states	mass fraction
T	total mass of the reactor	mass
D	dilution rate $D = \left(\frac{\dot{f}_S}{T}\right)$	1 / time
μ_X	cell production rate function	1 / time
$\mu_{X,d}$	cell death rate function	1 / time
q_S	glucose consumption rate function	1 / time
μ_A	acetate production rate function	1 / time
q_A	acetate consumption rate function	1 / time
μ_P	protein production rate function	1 / time
\dot{f}_S	mass flow rate of glucose feed	mass / time
C_S	concentration of glucose feed	mass fraction
O_{sat}	oxygen saturation concentration	mass fraction
k_O	oxygen mass transfer coefficient function	1 / time
q_O	oxygen consumption rate function	1 / time

Table 7.1: Symbols for Model Differential Equations.

model is still used because the parameters are easy to determine experimentally and it represents many systems reasonably well. It also makes sense because the shape is the same as the Michaelis-Menten reversible enzyme equation. Many of the relationships in this kinetic model are represented by a Monod form when the shape of the data suggests such a relationship.

Figure 7.1 is a block diagram of the overall kinetic model. Mass flows are represented by solid lines and information flow is represented by dotted lines. The model is a simplified flux-balance model for glucose and oxygen, and thus the carbon and oxygen mass balances close automatically. Uptake capacity is the uptake rate of a substance if the concentration of the substance is “high”. Both glucose and oxygen uptake rate capacities are assumed to depend on growth rate. These uptake rate capacities are then used along with glucose and oxygen concentration to find the glucose and oxygen uptake rates. Glucose and oxygen are then consumed by maintenance and cell growth. Part of cell growth is diverted to foreign protein production after induction. Any remaining oxygen is available for acetate consumption.

This model is most similar to the model presented by Neubauer et al. (2003), with several additions: it incorporates maintenance burden due to acetate as a function of acetate concentration, it is arranged so that most parameters are from literature sources and not experiment, and it incorporates oxygen mass transfer limitation due to cell death.

7.2.3 Uptake Rate Capacities

Glucose and oxygen uptake rate capacities ($q_{S,\text{cap}}$ and $q_{O,\text{cap}}$) may change with growth rate (μ_X). The model for these uptake rate capacities is a Monod expression in the growth rate. Lin et al. (2001) developed a method for estimating the maximum specific uptake capacities of glucose and oxygen, and this technique was used in a study of an *E. coli* system by Neubauer et al. (2003). Oxygen uptake rate in continuous

aerobic systems was explored by Calhoun et al. (1993). Reasonable ranges for the parameters in Equations 7.3 and 7.4 are derived from these studies. Table 7.3 presents the values of all parameters.

Growth rate is smoothed using a first order filter to prevent numerical instability. The filter time constant was one hour, but the value of the time constant makes little difference to the model states within a reasonable range.

$$q_{S,\text{cap}} = q_{S,\text{cap},\text{min}} + \frac{(q_{S,\text{cap},\text{max}} - q_{S,\text{cap},\text{min}})\mu_X}{q_{S,\text{cap},k_c} + \mu_X} \quad (7.3)$$

$$q_{O,\text{cap}} = q_{O,\text{cap},\text{min}} + \frac{(q_{O,\text{cap},\text{max}} - q_{O,\text{cap},\text{min}})\mu_X}{q_{O,\text{cap},k_c} + \mu_X} \quad (7.4)$$

7.2.4 Uptake Rates

Glucose and maximum oxygen uptake (q_S and $q_{O,\text{max}}$) in Equations 7.5 and 7.6 are Monod equations with the maximum term replaced by the rate capacities. Note that the kinetic model is designed so that all glucose uptake is consumed, but oxygen is only consumed as required for oxidation and aerobic anabolism, meaning that the oxygen uptake rate (q_O) may be less than the maximum oxygen uptake rate ($q_{O,\text{max}}$).

$$q_S = \frac{q_{S,\text{cap}}S}{(q_{S,k_c} + S)} \quad (7.5)$$

$$q_{O,\text{max}} = \frac{q_{O,\text{cap}}O}{q_{O,\text{max},k_c} + O} \quad (7.6)$$

7.2.5 Maintenance

Acetate decreases cell yield on glucose ($Y_{X/S}$) in both continuous (Gonzalez and Russell, 1997) and batch (O'Beirne and Hamer, 2000) cultures. This is taken into account in this model by increasing the glucose rate used for maintenance. However, a more common value seen in the literature is the cell yield. With the assumptions that glucose goes to either maintenance or cell growth, and that acetate doesn't affect

glucose uptake rate, Equation 7.7 describes the maintenance rate ($q_{S,m}$) vs. acetate concentration if aerobic cell yield on glucose ($Y_{\frac{X}{S},ox}$) vs. acetate concentration is known.

$$q_{S,m} = q_{S,m0} - q_S \int_0^A \frac{\partial Y_{\frac{X}{S},ox}}{\partial A} dA \quad (7.7)$$

In the case of the data reported by O'Beirne and Hamer (2000), the slope of cell yield vs. acetate is fairly constant, which allows a single parameter ($q_{S,m,kA}$) to describe the relationship between maintenance and acetate concentration. Equation 7.8 defines the system demand for maintenance glucose.

$$q_{S,m} = q_{S,m0} + q_S \cdot q_{S,m,kA} \cdot A \quad (7.8)$$

Glucose uptake may be uncoupled from glucose catabolism if glucose concentration is high (Russell and Cook, 1995). Figure 7.2 and Equation sets M1 and N1 describe the calculation within the maintenance block of Figure 7.1. Oxygen yield per glucose used for energy is $Y_{\frac{O}{S},en}$, acetate yield per glucose metabolised anaerobically is $Y_{\frac{A}{S},anaer}$, and glucose consumed anaerobically for energy is $q_{S,anaer}$.

$$q_S = q_{S,m} \quad (M1.1)$$

$$q_O = q_{S,m} \cdot Y_{\frac{O}{S},en} \quad (M1.2)$$

$$q_{S,anaer} = (q_{S,m} - q_S) \quad (N1.1)$$

$$q_A = q_{S,anaer} \cdot Y_{\frac{A}{S},anaer} \quad (N1.2)$$

7.2.6 Cell Growth

Cell growth is assumed to consume the rest of the available glucose, either aerobically or anaerobically. The cell growth fluxes are calculated using Equations M2 and

N2 with Figure 7.2.

$$q_S = q_{S,avail} \quad (\text{M2.1})$$

$$q_{S,anab} = \frac{q_S \cdot Y_{\frac{X}{S},ox}}{\Psi_{\frac{X}{S}}} \quad (\text{M2.2})$$

$$q_{S,en} = q_S - q_{S,anab} \quad (\text{M2.3})$$

$$q_O = q_{S,en} \cdot Y_{\frac{O}{S},en} + q_{S,anab} \cdot Y_{\frac{O}{S},anab} \quad (\text{M2.4})$$

$$q_{S,anaer} = (q_{S,avail} - q_S) \quad (\text{N2.1})$$

$$q_{S,anaer,anab} = \frac{q_{S,anaer} \cdot Y_{\frac{X}{S},anaer}}{\Psi_{\frac{X}{S}}} \quad (\text{N2.2})$$

$$q_{S,anaer,en} = q_{S,anaer} - q_{S,anaer,anab} \quad (\text{N2.3})$$

$$q_A = q_{S,anaer,en} \cdot Y_{\frac{A}{S},anaer} \quad (\text{N2.4})$$

Equation set M2 describes glucose and oxygen fluxes of aerobic metabolism. Glucose transferred to the mass in cells is $q_{S,anab}$, yield of cells per glucose transferred to the mass in cells is $\Psi_{\frac{X}{S}}$, and yield of oxygen transferred to the mass in cells per glucose transferred to the mass in cells is $Y_{\frac{O}{S},anab}$.

Equation set N2 describes glucose and oxygen fluxes of anaerobic metabolism, with the anaerobic equivalents of the entities used in Equation set M2.

7.2.7 Foreign Protein Production

After inducer is added, foreign protein production is estimated to be a fixed fraction of cell growth in a scheme similar to Neubauer et al. (2003). The parameter P_α is fitted to experimental data. This assumes that protein yield on glucose is the same as the cell yield on glucose.

$$\begin{aligned}\mu_P &= P_\alpha \cdot \mu_X \\ \mu_X &= (1 - P_\alpha) \mu_X\end{aligned}\tag{7.9}$$

Foreign protein production dependence on inducer concentration is not considered in this study. With the data available, inducer was used in large enough concentration that induction was considered on or off.

7.2.8 Acetate Consumption

Oxygen capacity remaining after all glucose is consumed is available for acetate oxidation for cell growth. Equation 7.10 is the maximum rate of acetate consumption by the cells if unlimited oxygen were available, using a Monod term with respect to acetate concentration, and a second term to account for inhibition by glucose. Equation 7.11 is the cell growth rate due to acetate consumption, Equation 7.12 is the rate of acetate used for energy for this process, and Equation 7.13 is the rate of oxygen consumed in this process. If the rate of oxygen required exceeds the amount available, Equations 7.10–7.13 are scaled back so that the available oxygen is consumed.

It is assumed that some glucose is fed all the time or available in the system, since this has been shown to increase acetate consumption (Xu et al., 1999), and this is a realistic condition for many industrial cultivations.

$$q_A = \left[\frac{q_{A,\max} A}{q_{A,k_c} + A} \right] \left[\frac{q_{A,i}}{q_{A,i} + S} \right]\tag{7.10}$$

$$\mu_{X,A} = q_A \cdot Y_{X/A}\tag{7.11}$$

$$q_{A,\text{en}} = q_A - \frac{\mu_{X,A}}{\Psi_{X/A}}\tag{7.12}$$

$$q_{O,A} = q_{A,\text{en}} \cdot Y_{O/A}\tag{7.13}$$

7.2.9 Carbon Dioxide generation rate

The carbon dioxide generation rate is assumed proportional to the oxygen consumption rate for energy.

$$\mu_{CO_2} = Y_{\frac{CO_2}{O}} \cdot q_{S,en} \cdot Y_{\frac{O}{S},en} \quad (7.14)$$

7.2.10 Cell Death

Lewis et al. (2004) used multi-parameter flow cytometry to study the effects of inclusion bodies on cells. They found huge stress caused by the inclusion bodies, although it is unknown if this stress is caused by the metabolic burden of the foreign protein, the physical imposition of the inclusion body, or both. Castan and Enfors (2000) found that the level of free DNA in the cell broth increases after induction. Assuming the free DNA comes from cell lysis, a logical assumption is that inclusion bodies correlate with cell lysis.

Few modeling studies in the literature address cell death in high-density recombinant cultivation. This information is important because a relative small fraction of cell lysis leads to concentrations of free DNA in the cell broth which will significantly inhibit oxygen transport.

In this model, the rate of cell death is proportional to the concentration of foreign protein. With this assumption, predicted dead cell concentration for the current data set looks qualitatively like the DNA concentration reported by Castan and Enfors (2000).

$$\mu_{X,d} = \tau_{X,d} \cdot P \quad (7.15)$$

Because free DNA concentration is not known for this data set, $\tau_{X,d}$ is fit indirectly through its effect on other states.

7.2.11 Oxygen Mass Transport

It is well known that oxygen mass transport sometimes decreases during high density cultivation. Castan and Enfors (2000) made a convincing argument that free DNA in the cell broth is the culprit in cases with significant cell lysis, and plotted the relationship between the oxygen mass transport rate and DNA concentration in the cell broth. This relationship can be modeled approximately as a simple inhibition by DNA, and the inhibition parameter $k_{O,i}$ is obtained from this study.

$$k_O = k_{O,0} \left(\frac{k_{O,i}}{k_{O,i} + X_d} \right) \quad (7.16)$$

7.2.12 Flux-balance Model Yields

There are two kinds of yield used in the model. The first kind is the traditional yield given in the literature. An example in this model is $Y_{X,ox}$, which answers the question: “what is the ratio of mass cells created to mass substrate consumed during aerobic metabolism?”. Another question is: “what is the ratio of mass cells created to mass substrate consumed during aerobic metabolism, considering only the substrate that physically goes into creating the cells, and ignoring the substrate catabolised for energy to drive the process”. The yield used to answer this second question will be called the “anabolic yield”, with the symbol Ψ . Following Xu et al. (1999), we take the anabolic yield to be the ratio of carbon content between the two substances. For example, the anabolic yield of cells from glucose is found in Equation 7.17.

$$\Psi_{\frac{X}{S}} = \frac{0.033 \frac{\text{molC}}{\text{gS}}}{0.04 \frac{\text{molC}}{\text{gX}}} = 0.83 \frac{\text{gX}}{\text{gS}} \quad (7.17)$$

The values and sources for all yields used in the model are presented in Table 7.2.

Yield	Value	Range	Literature	Fitted	Units	Sensitivity	Sources
$Y_{\bar{S},ox}$	0.61	0.60–0.61	0.42–0.60	Yes	g/g	24	1,2
$Y_{\bar{S},anaer}$	0.15		0.15		g/g	0.6	3
$Y_{\bar{A}}$	0.35		0.29–0.40		g/g	0.02	3
$Y_{\bar{A},anaer}$	0.67		0.67		g/g	2.3	3
$Y_{\bar{O},en}$	1.067		1.067		g/g	14	4
$Y_{\bar{O},anab}$	0.16		0.16		g/g	3.5	5
$Y_{\bar{A}}$	1.067		1.067		g/g	0.02	4
$Y_{\frac{CO_2}{O}}$	1.38		1.38		g/g	0.45	6
$\Psi_{\bar{S}}$	0.83		0.83		g/g	21	5
$\Psi_{\bar{A}}$	0.86		0.86		g/g	0.01	5

-
- 1 Luli and Strohl (1990)
 - 2 Han et al. (2003)
 - 3 Xu et al. (1999)
 - 4 stoichiometry
 - 5 chemical formula
 - 6 Andersen and Vonmeyenburg (1980)

Table 7.2: Flux-balance model yields

Parameter	Value	Range	Literature	Fitted	Units	Sens.	Sources
$q_{O,\text{cap},\text{min}}$	0.23	0.233–0.252	0.05-0.16	Yes	1/hr	4.8	1,2,3,4
$q_{O,\text{cap},\text{max}}$	0.32	0.314–0.319	0.26-0.55	Yes	1/hr	13	1,2,3,4
q_{O,cap,k_c}	0.15		0.15		1/hr	0.9	1
$q_{S,\text{cap},\text{min}}$	0.54	0.687–0.750	0.13-0.55	Yes	1/hr	3.1	1,2,3
$q_{S,\text{cap},\text{max}}$	0.77	0.697–0.736	0.90–1.5	Yes	1/hr	10	1,2,3
q_{S,cap,k_c}	0.10		0.10		1/hr	0.03	1
q_{O,max,k_c}	9.6E-9		9.6E-9		g/g	0.05	5
q_{S,k_c}	5.0E-5		5.0E-5		g/g	0.01	6
$q_{S,m0}$	0.04		0.04		1/hr	1.8	6
$q_{S,m,kA}$	26		26		g/g	1.3	7
$q_{A,\text{max}}$	0.14		0.14		1/hr	0.18	6
q_{A,k_c}	0.05		0.05		g/g	0.14	6
$q_{A,i}$	5.0E-5		5.0E-5		g/g	0.04	8
P_α	0.43	0.380–0.431	N/A	Yes	fraction	0.47	2
$k_{O,0}$	1070	1290–1500	N/A	Yes	1/hr	0.65	
$k_{O,i}$	4.2E-3		N/A		g/g	0.13	9
$\tau_{X,d}$	0.51	0.213–0.246	N/A	Yes	1/hr	0.06	

-
- 1 Lin et al. (2001)
 - 2 Neubauer et al. (2003)
 - 3 Han et al. (2003)
 - 4 Calhoun et al. (1993)
 - 5 Alexeeva et al. (2000)
 - 6 Xu et al. (1999)
 - 7 O’Beirne and Hamer (2000)
 - 8 same as q_{S,k_c}
 - 9 Castan and Enfors (2000)

Table 7.3: Flux-balance model parameters

7.3 Results and Discussion

The model fitted parameters were trained using 13 industrial pilot-plant data runs using SGE strains of *E. coli* in a 1000 liter bioreactor to produce recombinant protein. The system is induced to produce foreign protein nine hours after inoculation. The software used to train, simulate, and find the optimal control for the system is available at <http://danielwebb.us/research/>.

Tables 7.2 and 7.3 show the values of all parameters used in the model and indicate if they were fitted or not. The fitted parameters were found by minimizing the sum squared error between the model and data for optical density, glucose concentration, acetate concentration, foreign protein concentration, oxygen concentration, oxygen uptake rate and carbon dioxide evolution rate. The error for each data type was normalized by dividing the raw sum squared error by the total number of data points for that data type and also dividing by the squared maximum data value for that data type.

Representative training results are shown as Figure 7.3. Error plots have the same ordinate scale as their corresponding state plot. The number of data points was reduced for clarity in all sub-figures of Figures 7.3 and 7.4 except acetate and protein concentration.

Most of the states are predicted moderately well by the model, and the error plots show that there are no consistent run-to-run trends because the errors are randomly distributed between runs. The poorest fit is for acetate concentration (Figure 7.4c), where the typical error is the same order of magnitude as the average value. Acetate is extremely difficult for a simple model to predict, because it is essentially the integration of the difference between glucose uptake capacity and metabolic capacity. Even a small error predicting these systems will quickly lead to a large error in acetate concentration. If the model were used for on-line control, this problem could be addressed by slightly lowering the oxygen uptake capacity parameters to give a margin of safety.

Another noticeable state error is the carbon dioxide transport rate after induction (Figure 7.3g), which is uniformly under-predicted. This indicates that the model is under-predicting the fraction of oxygen going to energy production after induction. Increasing the acetate maintenance parameter (q_{S,m,k_A}) 30% brings the prediction of all states very close to data for four of the thirteen runs, but for the other runs leads to runaway acetate production which stalls cells growth after induction. In general the remaining states show run-to-run error that doesn't suggest any obvious biases other than the two discussed already.

Parameter sensitivity is defined such that if a 1% change in a parameter causes a 1% change in the final normalized error, that parameter's sensitivity is 1.0. This is defined formally in Equation 7.18.

$$Sensitivity = \left| \frac{\partial Error}{\partial Parameter} \cdot \frac{Parameter}{Error} \right| \quad (7.18)$$

This sensitivity allows a comparison of different parameters' importance, and is reported for each parameter and yield in Tables 7.2 and 7.3. It is obvious that the oxygen and glucose uptake capacities contain the most important parameters, and these are also the parameters that show large variations in the literature. This indicates that a small number of parameters should be able to capture much of the behavior of an *E. coli* cultivation, but that these parameters will still have to be found experimentally.

The glucose uptake capacity minimum parameter $q_{S,cap,min}$ is probably not very accurate for this data set, because when growth rate is low, glucose uptake is essentially limited by the rate of glucose input to the system, not the glucose uptake capacity parameters.

The Range column in Tables 7.2 and 7.3 gives the range of the parameters seen when the parameters were fitted one run at a time. To get an idea which parameters vary the system the most, the relative range of each parameter was multiplied by the

sensitivity (calculation not shown). This calculation shows that most of the system variability lies in just five parameters: $q_{O,\text{cap},\text{min}}$, $q_{O,\text{cap},\text{max}}$, $q_{S,\text{cap},\text{min}}$, $q_{S,\text{cap},\text{max}}$, and $Y_{S,\text{ox}}$.

An optimal control study was performed using the parameters found by fitting all runs simultaneously. The objective function was formulated to maximize final protein concentration with the restriction that the final mass must be less than 1000 kg. Iterative dynamic programming (Luus, 2000) was used with 30 stages, 10 state points, 15 control points, and with a contraction factor of 0.9. In addition, a first-order filter on the control (Tholudur and Ramirez, 1997) was used to obtain a smooth control profile. The initial conditions were chosen using one of the data runs. The oxygen uptake capacity maximum parameter ($q_{O,\text{cap},\text{max}}$) was reduced 7% to make sure that the model didn't overpredict oxygen uptake.

Figure 7.5 presents some of the results from the optimal control study. Points in the glucose feed rate plot are the smoothed glucose feed rate data and the line is the calculated optimal control. For other plots, lines correspond to the optimal control and points correspond to the model prediction using the data control. The model prediction for the run chosen was very close to the data.

Optimizing a complex system is often more difficult than simply optimizing the most difficult subsystem. A common example in bacteria cultivation is controlling glucose feed to hold dissolved oxygen concentration constant, in order to avoid acetate production and thus hopefully obtain optimal production. This optimal control study shows that this strategy may not lead to optimal production. Interestingly, the optimal control has submaximal glucose and oxygen uptake during the middle part of the cultivation, which differs from common industrial practice. The optimal control is predicted to produce a 30% increase in protein compared to the data case.

7.4 Conclusions

A new more complete flux-balance mathematical model for high-density recombinant *E. coli* cultivation was presented which predicts concentration of cells, dead cells, glucose, acetate, foreign protein, and oxygen. Oxygen uptake rate and carbon dioxide evolution rate are also predicted. The model fits an industrial pilot-plant data set reasonably well for a simple model with nine adjustable parameters. Acetate was difficult to fit due to the high sensitivity of acetate production to glucose and oxygen uptake rates. However, if the main use of the model is optimization or control, the concentration of acetate is not the main consideration, but its effect on the rest of the system. For this case, a safety margin could be incorporated into acetate production by slightly lowering oxygen uptake parameters.

The model is simple and fast enough to be used in an industrial on-line optimal control system. It could also be incorporated into a hybrid model so that the model improves as more data become available.

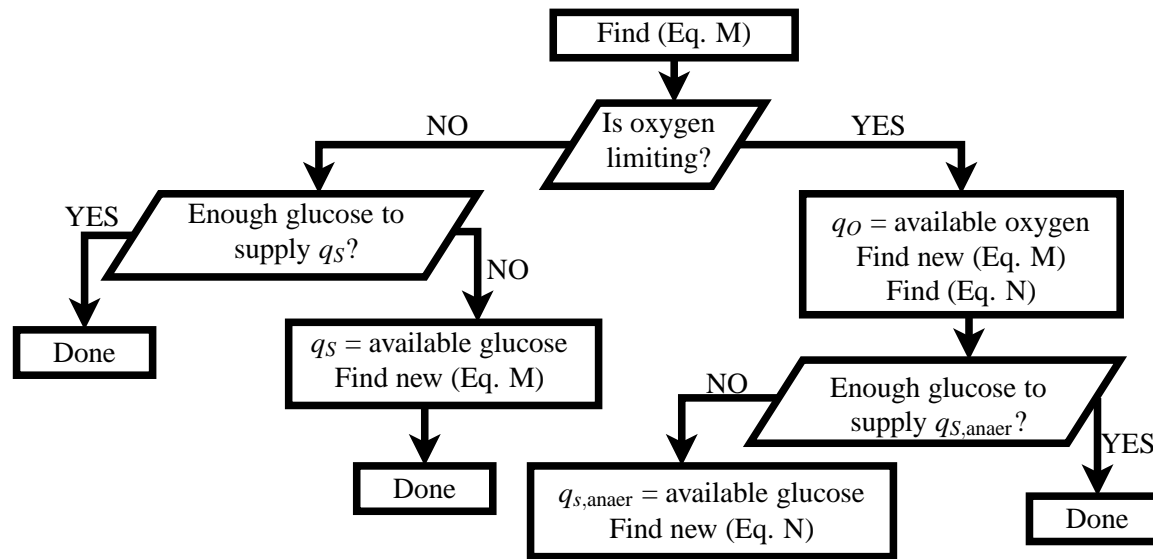


Figure 7.2: Kinetic Model Metabolism

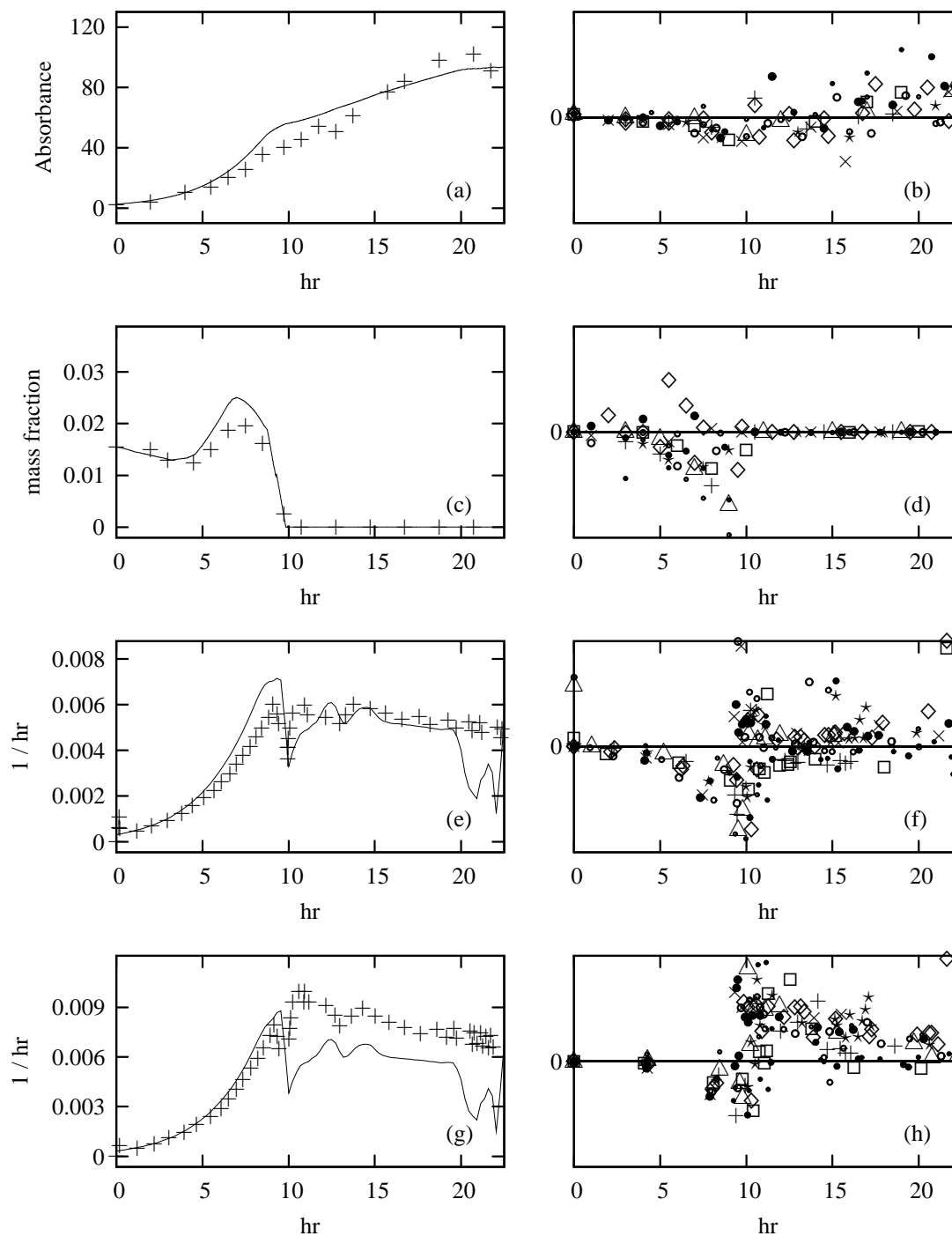


Figure 7.3: Training results for model using industrial data. Model (line) vs. data (points) of a representative run are given for: (a) optical density; (c) glucose concentration; (e) oxygen transport rate; (g) carbon dioxide transport rate. Errors for all runs are given in: (b) optical density error; (d) glucose concentration error; (f) oxygen transport rate error; (h) carbon dioxide transport rate error. Error plots have the same ordinate scale as the corresponding state plot.

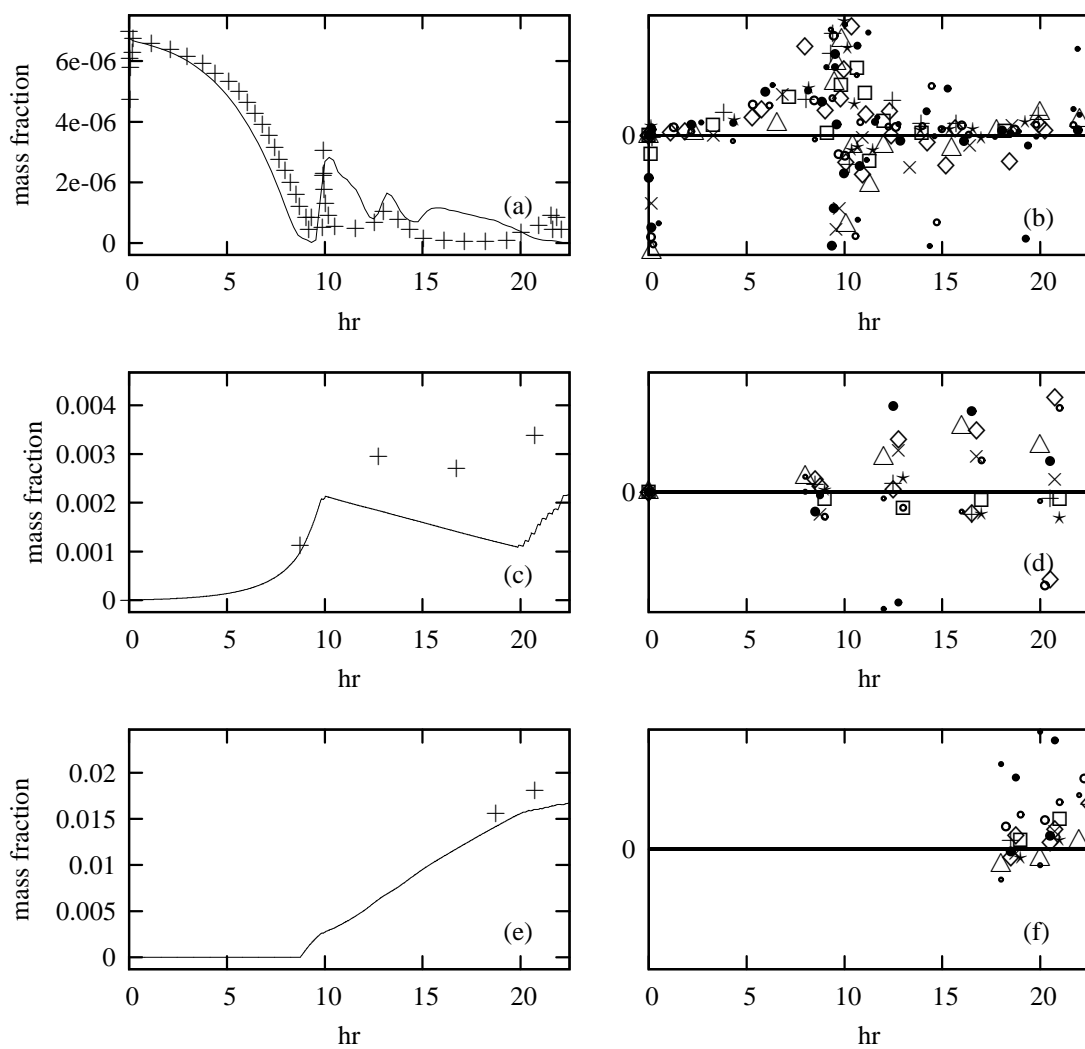


Figure 7.4: Training results for model using industrial data. Model (line) vs. data (points) of a representative run are given for: (a) oxygen concentration; (c) acetate concentration; (e) foreign protein concentration. Errors for all runs are given in: (b) oxygen concentration error; (d) acetate concentration error; (f) foreign protein concentration error. Error plots have the same ordinate scale as the corresponding state plot.

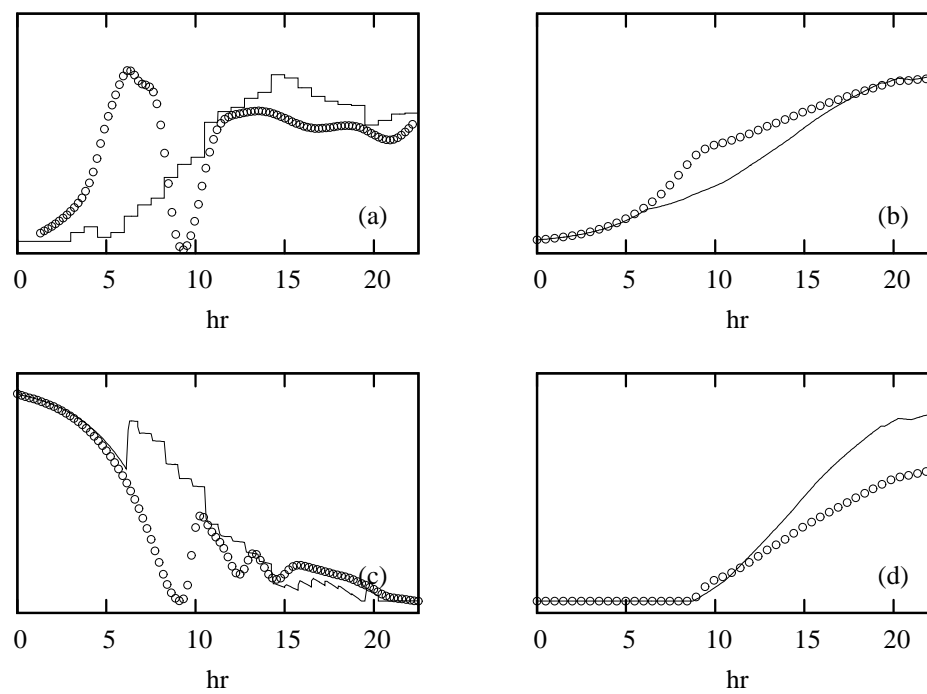


Figure 7.5: Optimal control calculated using IDP (lines) and model using actual control (circles): (a) glucose feed rate; (b) optical density; (c) oxygen concentration; (d) foreign protein concentration.

Chapter 8

Conclusions and Recommendations

8.1 Recommendations for IDP optimal control

The optimal control problem can be a very difficult problem, requiring appropriate algorithms, intuition, skill, and sometimes luck to get the global optimum. Simple rules and all-purpose algorithms may fail. This section gives some general guidelines for using the IDP algorithm to solve the fixed final time optimal control problem.

A new problem should always be attempted first using the basic IDP algorithm with as few stages, state grids, and test controls as possible using stagewise constant controls or stagewise linear continuous controls. If that gives an acceptable solution, you can go home early that day because the solution is likely to be fast and stable with respect to initial conditions and model parameters.

If the solution using a low number of stages is not acceptable because the performance index is not good enough or because coarse stage discretization causes important control features to be missed, the basic IDP algorithm with more stages should be tried, possibly adding variable stage lengths.

8.1.1 General recommendations for all IDP problems

- Use as few stages as possible.
- Use as few test controls as possible. The minimum is two (previous optimal

plus one test control), but sometimes a few more are needed to reliably get the global optimal control. It is probably better to increase adaptive region size history iterations than to use more test controls.

- Use one state grid if possible. Most problems do not need multiple state grids, but a few can't be solved without them.
- Integrator tolerances have a major impact on computation speed for IDP. Use the largest integrator tolerances that still give good results.
- If accurate control values are important, use stagewise continuous linear controls, and try stagewise discontinuous linear controls if there are several control discontinuities.
- If exact switching times are important for discontinuous controls, use variable stage lengths solved sequentially after controls.
- Find a reasonable combination of adaptive control region parameters k_r and A_H through trial and error. Try $k_r = 1.5$ and watch the progression of an IDP run to choose A_H (see Section 4.5.2).

8.1.2 Recommendations for two-step IDP

- Run the basic IDP algorithm with adaptive control region size until the performance index is sufficient (see Section 4.3). Use the number of iterations required plus a few more as the number of iterations for the first step of the two-step procedure.
- Use a small relative control damping (Section 5.1.5), the simulated annealing filter with a low initial temperature (Section 5.1.4), and pivot point test controls (Section 5.1.6).

- Try different combinations of smoothing techniques from Section 5.1 if the previous smoothing recommendation does not work.

8.2 IDP smoothing techniques

This section gives recommendations relating to the five smoothing methods described in Sections 5.1.3–5.1.7 and considers their strengths and weaknesses for use during the second step of the two-step IDP method. Although many different combinations of IDP and problem parameters were tested during this study, there were only two problems used to draw these conclusions, so they should be considered limited. Which method works best will probably depend somewhat on the problem.

The simulated annealing filter is worth trying on any problem with a moderate initial temperature (0.1–1.0) and a small value of α_A , since with these parameters the filter will be inactive by the end of the run. In this study, the simulated annealing filter with a low initial temperature was found to be somewhat useful even during step one of the two-step procedure for the identification problem.

Control damping seems to be mainly useful in low amounts ($\beta_{u\Phi} < 10^{-3}$) to smooth small or subtle control activity such as seen in Figure 5.13. In larger amounts it seems to do more harm than good.

For smooth problems, pivot point test controls may solve the active control problem without the need for other methods, so pivot point test controls should be tried first. All other smoothing methods were observed to harm the solution if their parameters were poorly chosen. There are still ways for pivot point test controls to get hung up, such as when two consecutive stages are too high balanced by one following stage that is too low. The simulated annealing filter with a low temperature will break these hang-ups and allow pivot point test controls to do the rest.

The results for the first-order filter presented in Section 5.1.3 may be slightly misleading, although this was not intended. Under many other conditions for the Park-

Ramirez optimal control problem, the first-order filter performed much more poorly than with the problem parameters chosen for the examples in Chapter 5. The first-order filter often led to a poor performance index or rounded control profiles when strong enough to sufficiently smooth the control profile. However, if the simulated annealing filter gives poor results, the first-order filter should be tried since it works in a different way.

Solo test controls were not always converge faster than

8.3 Recommendations for Webb-Ramirez identification

- There seems to be an optimal number of IDP stages for this method. Too few stages will miss fast model dynamics and too many stages will result in noisy training data. Use as few stages as possible to capture important state dynamics. Use less than half as many control stages as state data if possible. Noise amplifies quickly as the ratio of stages to data passes $\frac{1}{2}$.
- If states are dropping to very small values or running away to infinity, use state limiting and limit punishment (Section 6.3).
- Use sensitivity data replication (Section 6.7.2) if there are parameter functions that have sensitivity problems (such as ϕ in this study).

8.4 Conclusions

A new method for hybrid dynamic model identification was introduced, where parameter functions are treated as controls in an optimal control problem. The method was demonstrated on a simple bioreactor model with artificial data, and the neural network parameter functions were able to capture the functions used to produce the artificial data. This new method can identify several kinds of models the two current hybrid model identification methods can not identify.

The Webb-Ramirez hybrid identification technique is very demanding on the optimal control algorithm used with it, because it needs the control profile to be close to the optimal control profile in addition to minimizing the performance index. Iterative dynamic programming (IDP) was chosen to solve the optimal control problem because it places no restrictions on the model, but in its original form it was unsuitable because of its tendency toward active controls for this kind of problem.

The two-step IDP method of solving the optimal control problem was proposed to address the active control problem. When used with a control smoothing technique during the second step, it was found to greatly increase solution speed for 25 stages or more without increasing the likelihood of finding local minima.

Five smoothing methods were evaluated for use with the two-step IDP method. All five methods were useful for some of the cases for each of two problems studied, but more research is needed to draw firm conclusions about the relative merits of these methods.

Chapter 9

Possible topics for future research

9.1 IDP for high-speed real-time control

IDP is generally considered to be a slow algorithm. Several improvements were made in this study to improve the speed of IDP convergence for use with general purpose computers, but another interesting and useful application is implementation on specialized hardware for embedded real-time control.

Low-cost digital signal processor (DSP) chips are optimized around the multiply-add cycle, which is a perfect match for both feed-forward neural networks and simple integrators such as the Euler or Runge-Kutta integrators. Interestingly, the Euler method can do a fairly good job of solving the Park-Ramirez optimal control problem even with very large step sizes, as shown in Figures 9.1 and 9.2.

A custom implementation of IDP for these chips would be extremely fast by optimal control standards. With the speed improvements discussed in this work, real-time solution on such a processor should be possible for a problem similar to the Park-Ramirez optimal control problem in well under one second. For example, the 10 stages case in Figure 9.1 with an Euler stepsize of 0.05 hr and 50 iterations requires 12,500 equivalent integrations (Equation 4.2) and thus 4×10^6 model evaluations. If a simple model requires 10 multiply-add operations per model evaluation, this requires approximately 4×10^7 operations per optimal control solution. Some modern DSPs evaluate $> 10^8$ operations per second, meaning control could respond in sub-second time. Al-

though this kind of speed is certainly not necessary for this problem, there are surely many problems that could benefit from fast real-time optimal control.

9.2 Dealing with very noisy state data

The noise added to the artificial state data in the identification study in Chapter 6 was relatively low, although large enough to be noticeable in the figures. Industrial data are often much noisier than that, but very high noise caused the Webb-Ramirez identification technique to return very poor $v(t)$ values in the first step of the method.

A simple way to deal with high noise, especially if the data are very dense, is to curve-fit the data and use the curve-fit to create data for the Webb-Ramirez method to use. However, this is not feasible if there is little data for a state, and one of the reasons to propose the Webb-Ramirez in the first place was to avoid the curve-fitting step used in the derivative method (Section 6.1.1 on page 95).

Another potential technique to deal with the noise problem is to add a damping term to the optimal control performance index which punishes the state second derivatives (Section 9.2.1). This was only moderately helpful when trying to identify the hybrid Park-Ramirez model using Webb-Ramirez identification with very noisy data (results not shown). However, it was observed that the $v(t)$ values returned when very noisy state data were used were very noisy, but seemingly in a random way. If the noisy $v(t)$ returned from the first step of Webb-Ramirez identification are noisy but random, the first step of the method could be repeated multiple times with different starting conditions or random number generator seeds and then averaged, leading to a greatly improved $v(t)$ identification.

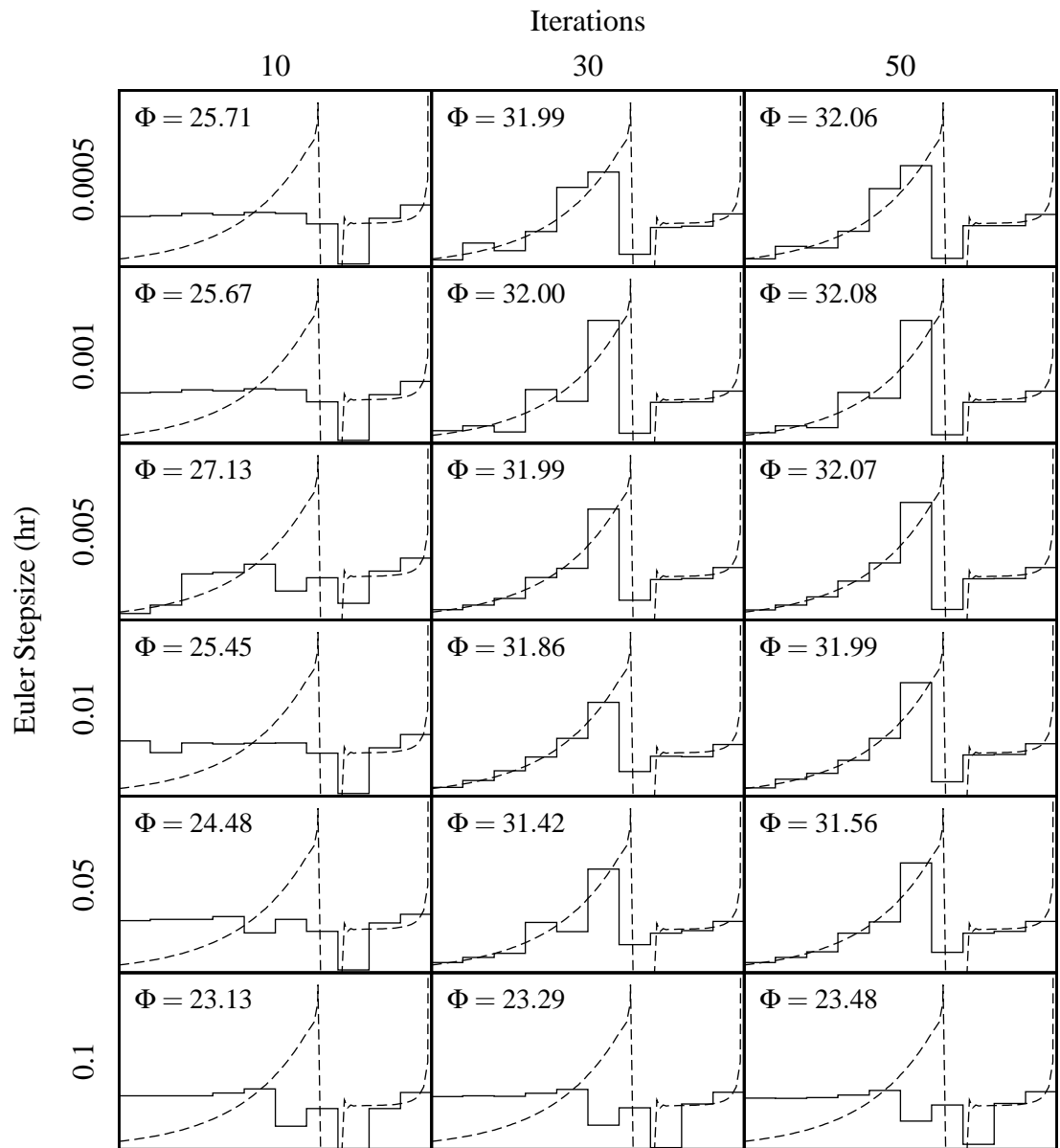


Figure 9.1: Solutions to the Park-Ramirez optimal control problem using an Euler integrator (10 stages).

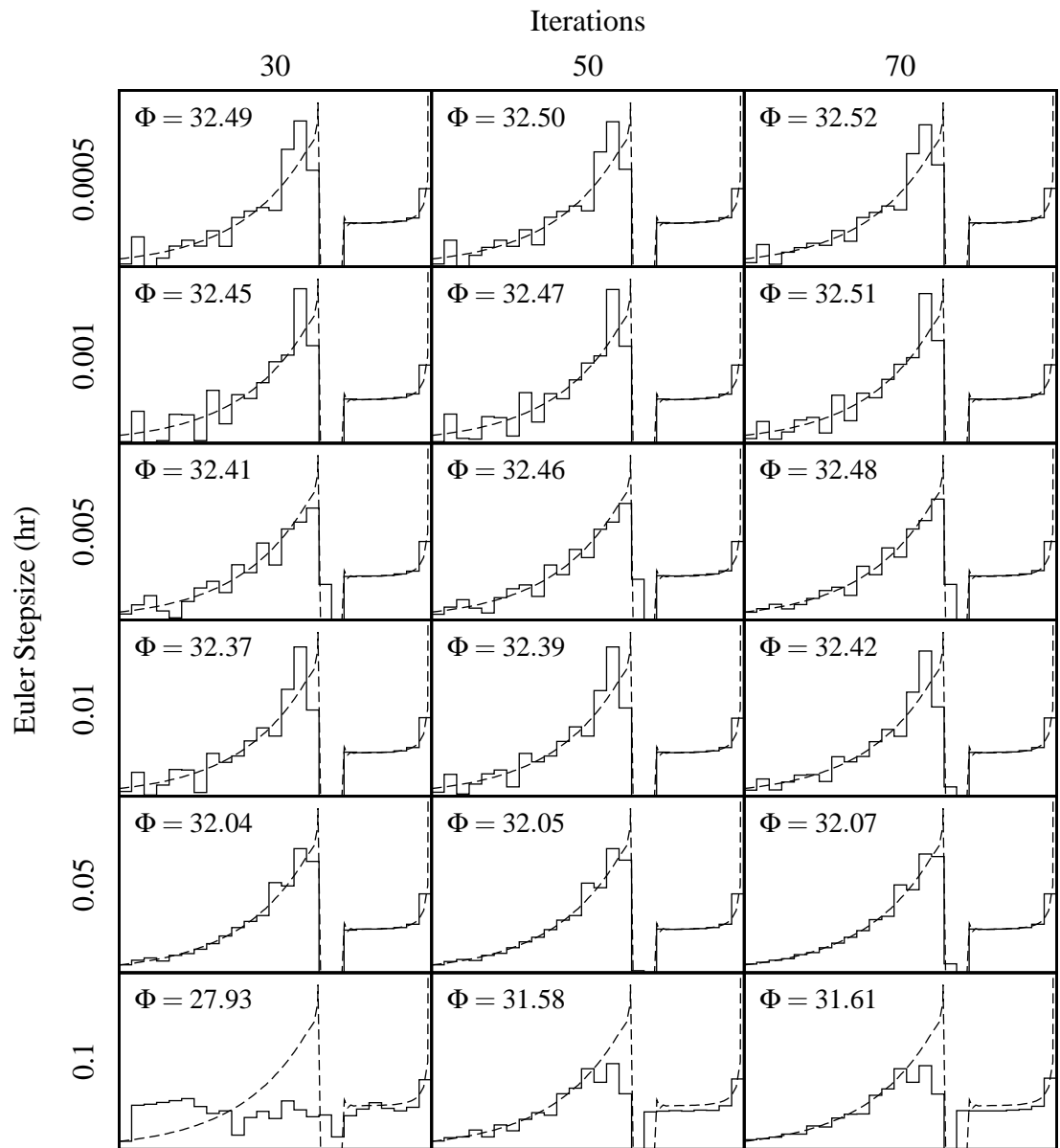


Figure 9.2: Solutions to the Park-Ramirez optimal control problem using an Euler integrator (25 stages).

9.2.1 State damping for noise reduction

9.2.1.1 Fixed state damping

State damping may be useful for the identification problem when limited state data are available. No matter what identification technique is used some assumption must be made about the domain between data points. For example, if only the data points are considered important, Figure 9.3a is a better solution than Figure 9.3b since all data points are predicted by the model. However, it is intuitively expected that Figure 9.3a is a poor solution that has over-fit the data.

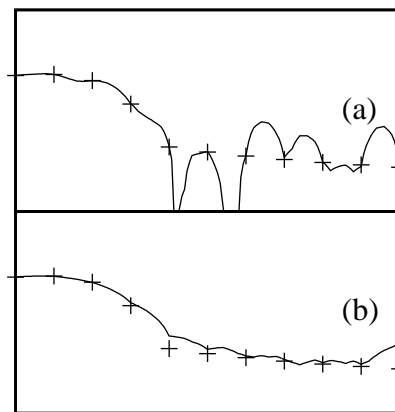


Figure 9.3: Example of identification problem overfitting. (a) without state damping, (b) with state damping.

One way to choose an arbitrary smoothness between states is to add a punishment term to the performance index. A possible state damping term is

$$\Psi_{Xd} = t_f \cdot M_{\Phi} \sum_i^I \left[\frac{\beta_{X_i}}{M_{X_i}} \int_0^{t_f} \left| \frac{d^2 X_i}{dt^2} \right| dt \right] \quad (9.1)$$

where t_f is the final time, M_{Φ} is a typical or maximum value of the performance index used to normalize the term, i is each of I states, M_{X_i} is a typical or maximum value of

state i , and β_{X_i} is an arbitrary damping factor. Higher values of β_{X_i} will lead to control policies which cause smoother state profiles.

Like control damping, this technique changes the problem. When a suitable value of state damping is found it should be checked by solving the problem with a slightly different value of state damping factor to see if the solution changes significantly.

9.2.1.2 Relative state damping

A problem with the state damping technique described in the previous section is that because the damping term ψ_{Xd} is added to the performance index, the effect of the state damping factors β_{X_i} depend on the value of the performance index. The value of the performance index can be very different even for different runs of the same model if the amount of noise is different from run to run, so unless M_Φ is updated for each new case, the damping will be inconsistent.

To deal with this problem relative state damping is used, where the fixed damping factors β_{X_i} are adjusted after each iteration so that the state damping term ψ_{Xd} is some fraction of the performance index. The first-order adjustment equation is

$$\beta_{X_i}^{a+1} = \beta_{X_i}^a + \frac{\beta_{X_i}^a}{\tau_{XA}} \left[\frac{\beta_{X\Phi} \cdot \Phi}{\psi_{Xd}} - 1 \right] \quad i = 1 \dots I \quad (9.2)$$

where $\beta_{X_i}^{a+1}$ is the value of β_{X_i} for the next iteration, $\beta_{X_i}^a$ is the value of β_{X_i} at the current iteration, τ_{XA} is the first-order time constant (in iterations), and $\beta_{X\Phi}$ is the desired value of ψ_{Xd} as a fraction of performance index Φ . This formula requires initial values for all the β_{X_i} and keeps the ratio between them fixed.

9.3 Approaching test control and state boundaries

A common theme in stochastic optimization is that more samples should better cover the problem domain. In the case of IDP, though, more state grids, iterations and test controls do not always increase the coverage of the state or control domain spaces

to the full possible domain. To see why, imagine the absurd case of a problem with 1000 IDP stages. With so many stages, the control will appear to be approximately constant at the expected value of the random variable used to generate the random test control values. One way to deal with this might be to give the random variable used to generate test controls a momentum in time. For example, a high value at stage 10 would mean the value at stage 11 would be likely to be high as well. Another way to deal with this is to use a uniform permutation of test controls (Tassone and Luus, 1993). This method requires $2J + 1$ test controls, where J is the number of controls, so it is only useful for smaller numbers of controls.

A more difficult problem is the problem of how to cover more of the possible state domain within the state bounds. The point of the state grids concept is to increase coverage of the state domain, but even using many state grids with a uniform permutation of test controls there is no guarantee that good coverage of the state domain will result. Because of this, IDP does not guarantee the ability to find the global optimum even if infinite state grids and test controls are used. One possibility to span the state domain better would be to first solve the optimal control problem with the objective of finding state grids that span the state limits, then starting the normal IDP solution from that point. Finding the controls that span the state space could be a difficult optimal control problem in itself.

Resolving these two problems would bring IDP closer to the goal of finding the global optimal control even for very difficult problems when sufficient samples are taken, while still maintaining IDP's efficiency compared to a direct solution of the discrete control optimal control problem.

Bibliography

- Alexeeva, S., De Kort, B., Sawers, G., Hellingwerf, K. J. and de Mattos, M. J. T. (2000), 'Effects of limited aeration and of the arcab system on intermediary pyruvate catabolism in *Escherichia coli*', Journal of Bacteriology **182**(17), 4934–4940.
- Andersen, K. and Vonmeyenburg, K. (1980), 'Are growth rates of *Escherichia coli* in batch cultures limited by respiration?', Journal of Bacteriology **144**(1), 114–123.
- Balsa-Canto, E., Vassiliadis, V. S. and Banga, J. R. (2005), 'Dynamic optimization of single- and multi-stage systems using a hybrid stochastic-deterministic method', Industrial and Engineering Chemistry Research **44**(5), 1514–1523.
- Barton, P. I., Allgor, R. J., Feehery, W. F. and Galan, S. (1998), 'Dynamic optimization in a discontinuous world', Industrial and Engineering Chemistry Research **37**(3), 966–981.
- Bellman, R. E. (1957), Dynamic Programming, Princeton University Press, Princeton, N.J.
- Bellman, R. E. (1961), Adaptive Control Processes, Princeton University Press, Princeton, NJ.
- Bojkov, B. and Luus, R. (1993), 'Evaluation of the parameters used in iterative dynamic programming', Canadian Journal of Chemical Engineering **71**, 451–459.
- Bojkov, B. and Luus, R. (1996), 'Optimal control of nonlinear systems with unspecified final times', Chemical Engineering Science **51**, 905–919.
- Bryson, A. and Ho, Y. (1975), Applied Optimal Control, Hemisphere: New York.
- Calhoun, M., Oden, K., Gennis, R., Demattos, M. and Neijssel, O. (1993), 'Energetic efficiency of *Escherichia coli* - effects of mutations in components of the aerobic respiratory-chain', Journal of Bacteriology **175**(10), 3020–3025.
- Castan, A. and Enfors, S.-O. (2000), 'Characteristics of a DO-controlled fed-batch culture of *Escherichia coli*', Bioprocess and Biosystems Engineering **22**(6), 509–515.

- Chang, D.-E., Shin, S., Rhee, J.-S. and Pan, J.-G. (1999), 'Acetate metabolism in a pta mutant of *Escherichia coli* w3110: Importance of maintaining acetyl coenzyme a flux for growth and survival', Journal of Bacteriology **181**, 6656–6663.
- Domach, M. M., Leung, S. K., Cahn, R. E., Cocks, G. G. and Shuler, M. L. (2000), 'Computer model for glucose-limited growth of a single cell of *Escherichia coli* b/r-a', Biotechnology and Bioengineering **67**(6), 827–840.
- Fikar, M., Latifi, M., Fournier, F. and Creff, Y. (1998), 'Control vector parametrisation versus iterative dynamic programming in dynamic optimisation of a distillation column', Computer and Chemical Engineering **22**, S625–S628.
- Franco-Lara, E. and Weuster-Botz, D. (2005), 'Estimation of optimal feeding strategies for fed-batch bioprocesses', Bioprocess and Biosystems Engineering **27**(4), 255–262.
- Free Software Foundation (1991), 'GNU general public license'. <http://www.gnu.org/copyleft/gpl.html>.
- Gonzalez, F. D. and Russell, J. B. (1997), 'The ability of *Escherichia coli* o157:h7 to decrease its intracellular ph and resist the toxicity of acetic acid', Microbiology **143**, 1175–1180.
- Gropp, W., Lusk, E., Doss, N. and Skjellum, A. (1996), 'A high performance, portable implementation of the MPI message passing interface standard', Parallel Computing **22**(6), 789–828.
- Han, L., Enfors, S.-O. and L., H. (2003), '*Escherichia coli* high-cell-density culture: carbon mass balances and release of outer membrane components', Bioprocess and Biosystems Engineering **25**(4), 205–212.
- Hartig, F., Keil, F. and Luus, R. (1995), 'Comparison of optimization methods for a fed-batch reactor', Hungarian Journal of Chemical Engineering **23**, 141–148.
- Hartig, F., Mandel, K. and Keil, F. J. (1999), 'Parallelization of iterative dynamic programming', Periodica Polytechnica Chemical Engineering **43**(1), 3–16.
- Himmelblau, D. M. (1970), Process analysis by statistical methods, Wiley, New York.
- Kraft, D. (1985), 'On converting optimal control problems into nonlinear programming problems', Computational Mathematical Programming **15**, 261–280.
- Laursen, S. O., Webb, D. and Ramirez, W. F. (2007), 'Dynamic hybrid neural network model of an industrial fed-batch fermentation process to produce foreign protein', Computers and Chemical Engineering **31**(3), 163–170.
- Leonard, J., Kramer, M. and Ungar, L. (1992), 'A neural network architecture that computes its own reliability', Computers and Chemical Engineering **16**(9), 819–835.

- Lewis, G., Taylor, I., Nienow, A. and Hewitt, C. (2004), 'The application of multi-parameter flow cytometry to the study of recombinant *Escherichia coli* batch fermentation processes', Journal of Industrial Microbiology and Biotechnology **31**(7), 311–322.
- Lin, H., Mathiszik, B., Xu, B., S.-O., E. and Neubauer, P. (2001), 'Determination of the maximum specific uptake capacities for glucose and oxygen in glucose-limited fed-batch cultivations of *Escherichia coli*', Biotechnology and Bioengineering **73**(5), 347–357.
- Luli, G. and Strohl, W. (1990), 'Comparison of growth, acetate production, and acetate inhibition of *Escherichia coli* strains in batch and fed-batch fermentations', Applied and Environmental Microbiology **56**(4), 1004–1011.
- Luus, R. (1990), 'Application of dynamic-programming to high-dimensional nonlinear optimal-control problems', International Journal of Control **52**(1), 239–250.
- Luus, R. (1993), 'Piecewise linear continuous optimal control by iterative dynamic programming', Industrial & Engineering Chemistry Research **32**, 859–865.
- Luus, R. (1998), Use of iterative dynamic programming for optimal singular control problems, in 'Proceedings of the International Conference on Optimization Techniques and Applications (ICOTA '98)', pp. 286–289.
- Luus, R. (2000), Iterative Dynamic Programming, Boca Raton, Florida : Chapman & Hall/CRC.
- Luus, R. and Bojkov, B. (1994), 'Global optimization of the bifunctional catalyst problem', Canadian Journal of Chemical Engineering **72**(1), 160–163.
- Luus, R., Dittrich, J. and Keil, F. (1992), 'Multiplicity of solutions in the optimization of a bifunctional catalyst blend in a tubular reactor', Canadian Journal of Chemical Engineering **70**, 780–785.
- Mekarapiruk, W. and Luus, R. (1997), 'Optimal control of inequality state constrained systems', Industrial and Engineering Chemistry Research **36**, 1686–1694.
- Mekarapiruk, W. and Luus, R. (1999), 'Iterative dynamic programming with adaptive scheme for region size determination', Hungarian Journal of Industrial Chemistry **27**, 235–240.
- Neubauer, P., Lin, H. and Mathiszik, B. (2003), 'Metabolic load of recombinant protein production: Inhibition of cellular capacities for glucose uptake and respiration after induction of a heterologous gene in *Escherichia coli*', Biotechnology and Bioengineering **83**(1), 53–64.
- Neuman, C. and Sen, A. (1973), 'A suboptimal control algorithm for constrained problems using cubic splines', Automatica **9**, 601–603.

- Nissen, S. (2003), Implementation of a fast artificial neural network library (fann), Technical report, Department of Computer Science University of Copenhagen (DIKU). <http://fann.sf.net>.
- O'Beirne, D. and Hamer, G. (2000), 'The utilisation of glucose/acetate mixtures by *Escherichia coli* w3110 under aerobic growth conditions', *Bioprocess Engineering* **23**(4), 375–380.
- Oliveira, R. (2004), 'Combining first principles modelling and artificial neural networks: a general framework', *Computers and Chemical Engineering* **28**(5), 755–766.
- Park, S. and Ramirez, W. F. (1988), 'Optimal production of secreted protein in fed-batch reactors', *AIChE Journal* **34**, 1550–1558.
- Psichogios, D. C. and Ungar, L. H. (1992), 'A hybrid neural network-first principles approach to process modeling', *AIChE Journal* **38**, 1499–1511.
- Ramirez, D. and Bentley, W. (1999), 'Characterization of stress and protein turnover from protein overexpression in fed-batch e-coli cultures', *Journal of Biotechnology* **71**(1-3), 39–58.
- Ramirez, W. F. (1994), *Process control and identification*, Academic Press: San Diego.
- Russell, J. B. and Cook, G. M. (1995), 'Energetics of bacterial growth: Balance of anabolic and catabolic reactions', *Microbial Reviews* **59**(1), 48–62.
- Sarkar, D. and Modak, J. (2004), 'Genetic algorithms with filters for optimal control problems in fed-batch bioreactors', *Bioprocess and Biosystems Engineering* **26**(5), 295–306.
- Schubert, J., Simutis, R., Dors, M., Havlik, I. and Lubbert, A. (1994), 'Bioprocess optimization and control - application of hybrid modeling', *Journal of Biotechnology* **35**, 51–68.
- Sunderam, V. S., Geist, G. A., Dongarra, J. and Manchek, R. (1994), 'The PVM concurrent computing system: Evolution, experiences, and trends', *Parallel Computing* **20**(4), 531–545. <http://citeseer.ist.psu.edu/article/sunderam94pvm.html>.
- Tassone, V. and Luus, R. (1993), 'Reduction of allowable values for control in iterative dynamic programming', *Chemical Engineering Science* **48**, 3864–3867.
- Teixeira, A., Cunha, A. E., Clemente, J. J., Moreira, J. L., Cruz, H. J., Alves, P. M., Carrondo, M. J. T. and Oliveira, R. (2005), 'Modelling and optimization of a recombinant bhk-21 cultivation process using hybrid grey-box systems', *Journal of Biotechnology* **118**(3), 290–303.
- Tholudur, A. N. (1998), *Neural Network Modeling and Optimization of Biotechnology Processes*, PhD thesis, University of Colorado at Boulder.

- Tholudur, A. and Ramirez, W. F. (1996), 'Optimization of fed-batch bioreactors using neural network parameter function models', Biotechnology Progress **12**, 302–309.
- Tholudur, A. and Ramirez, W. F. (1997), 'Obtaining smoother singular arc policies using a modified iterative dynamic programming algorithm', International Journal of Control **68**, 1115–1128.
- Tholudur, A. and Ramirez, W. F. (1999), 'Neural-network modeling and optimization of induced foreign protein production', AIChE Journal **45**, 1660–1670.
- Tholudur, A., Ramirez, W. F. and McMillan, J. D. (1999), 'Mathematical modeling and optimization of cellulase protein production using trichoderma reesei rl-p37', Biotechnology and Bioengineering **66**, 1–16.
- Tsang, T., Himmelblau, D. and Edgar, T. (1975), 'Optimal control via collocation and non-linear programming', International Journal of Control **21**, 763–768.
- Upreti, S. (2004), 'A new robust technique for optimal control of chemical engineering processes', Computers and Chemical Engineering **28**(8), 1325–1336.
- Webb, D. M. (2006), 'Daniel M. Webb research page'. <http://danielwebb.us/research/>.
- Xiong, Z. H. and Zhang, J. (2005), 'Neural network model-based on-line re-optimisation control of fed-batch processes using a modified iterative dynamic programming algorithm', Chemical Engineering and Processing **44**(4), 477–484.
- Xu, B., Jahic, M. and Enfors, S.-O. (1999), 'Modeling of overflow metabolism in batch and fed-batch cultures of *Escherichia coli*', Biotechnology Progress **15**, 81–90.

Appendix A

Software

This appendix summarizes all the software used in the completion of this research and dissertation. Much of the software used was original work, but as Newton said, “If I have seen a little further it is by standing on the shoulders of Giants.” This is nearly always true with software because of the huge foundation that is required for a system such as this.

A.1 Philosophy

If nature has made any one thing less susceptible than all others of exclusive property, it is the action of the thinking power called an idea, which an individual may exclusively possess as long as he keeps it to himself; but the moment it is divulged, it forces itself into the possession of every one, and the receiver cannot dispossess himself of it. Its peculiar character, too, is that no one possesses the less, because every other possesses the whole of it. He who receives an idea from me, receives instruction himself without lessening mine; as he who lights his taper at mine, receives light without darkening me. That ideas should freely spread from one to another over the globe, for the moral and mutual instruction of man, and improvement of his condition, seems to have been peculiarly and benevolently designed by nature, when she made them, like fire, expansible over all space, without lessening their density in any point, and like the air in which we breathe, move, and have our physical being, incapable of confinement or exclusive appropriation.

Thomas Jefferson; from a letter to Isaac McPherson, Monticello, August 13, 1813.

The parallels between the evolution of science and the evolution of software are compelling. Both build knowledge incrementally. Both consist principally of ideas and not tangible goods. Once developed, the advances of both can be spread without cost.

It should seem odd to a scientist that the advances in one of these fields are freely shared while the advances in the other are hoarded and dispensed like a prized brandy. Some say that algorithms and methods should be shared freely while their software implementation should not because there is a fundamental difference between them, but the difference is only a difference of degree, not a difference in kind.

With this in mind, I only used software for this research which allows this Jeffersonian expansion of knowledge, and my original software is released¹ under such a license as well. I also encourage other researchers to release their original works under similar licenses so that the progress of software can advance as quickly as the progress of science. As science becomes more and more dependent on software, I believe it will become clear that this is as much a practical necessity as a utopian ideal.

A.2 Programming language and compiler tools

Web links are like leprechauns and they disappear as soon as you put them to paper, so I will refer to all software by name only.

All original software was written in ISO 1999 standard C, and compiled and linked using the GNU project's gcc compiler toolchain, version 3.3.5. The compiler toolchain is wrapped by the autotools toolchain which provides platform independence for the build. The operating system for all work was the Debian GNU/Linux distribution, and Vim was used for all text editing. The Subversion version management software was used to keep track of the many revisions of software, protect against accidental changes, and to revert to previous versions of the software when ideas failed to pan out.

¹ <http://danielwebb.us/research>

A.3 IDP library

The IDP library implementation² will probably be the most useful to others since it is novel and I aimed for a professional level of quality. This library has only a few dependencies, all written in C and included. The use of this library is very simple and requires only four basic things:

- (1) A user function that returns the performance index given the state values.
- (2) A user function that returns the state initial values.
- (3) A user function that returns the state derivatives given the state values.
- (4) A main function which sets all the needed IDP settings and calls the IDP solver.

Notice that this is for non-stiff ODE models, for which a custom Runge-Kutta 4/5 integrator was written. The public-domain integrator `daskr`³ is also included with the package but has not been as well-tested as the Runge-Kutta ODE integrator. `Daskr` can integrate stiff ODE models and DAE models using an implicit method.

An example program (`test.c`) is included with the library to demonstrate the optimal control solution of the Park-Ramirez bioreactor model in about 200 lines of C code.

A.4 Model framework library

The “model” library is a relatively large library written to manage the IDP solution of so many different combinations of IDP parameters. It is essentially a wrapper to load and save data, choose IDP settings, and implement the Webb-Ramirez identification method. For the studies in this dissertation, the model library was called at the command line by a bash script which organized the runs by group (ie. the 50 stage case

² <http://danielwebb.us/research/idp>

³ written by Linda R. Petzold, Peter N. Brown, Alan C. Hindmarsh, and Clement W. Ulrich.

of the pivot point test controls study or the 10 stage case of a $v(t)$ identification study). This bash script, model binaries, and data files were finally passed to remote computers for execution so that multiple computers could be used. All simulations shown or mentioned in this study would require approximately one week using four 2Ghz computers if run consecutively.

The Webb-Ramirez identification method requires a tricky implementation to accomplish the measurement of E (measured vs. predicted states in Equation 6.8 on page 99). The error before the IDP stage under consideration is fixed, but the error after the IDP stage under consideration changes with each new test control. For this reason, the IDP library passes the stage integration back to the model library so that it can handle the calculation of E correctly without needlessly complicating the IDP library.

A.5 Dissertation software tools

All typesetting was accomplished with the \LaTeX typesetting system, which is a set of macros originally developed by Leslie Lamport to simplify use of the \TeX typesetting system by Donald Knuth. Many extension packages were used, including the `amsmath` package for mathematics, the `pslatex` package for improved Postscript support, the `natbib` package for enhanced bibliography support, and the `algorithm` package for the algorithm figures. The `latex-beamer` package was used to generate the PDF computer slide presentation for defense of this dissertation. This \LaTeX source code was edited using the \LaTeX plugin for Vim, and was compiled using the Web2C implementation of \LaTeX and \TeX .

All figures were generated using the gnuplot plotting software. Data was loaded dynamically by gnuplot from the model library data files, and gnuplot command files were preprocessed with the m4 text preprocessor to make the large number of figures in this dissertation manageable. Animations⁴ were generated by combining a sequence

⁴ <http://danielwebb.us/research/idp>

of gnuplot-generated fixed plots with the ffmpeg mpeg4 video encoder.

Appendix B

Complete $v(t)$ identification results for one case

Only the first run from the eight runs of $v(t)$ identification results were shown in the main text. This appendix presents the remaining seven runs (Figures B.1 – B.7) for the case with 50 IDP stages, 5 P_T data and 1% data noise from Section 6.8.4.

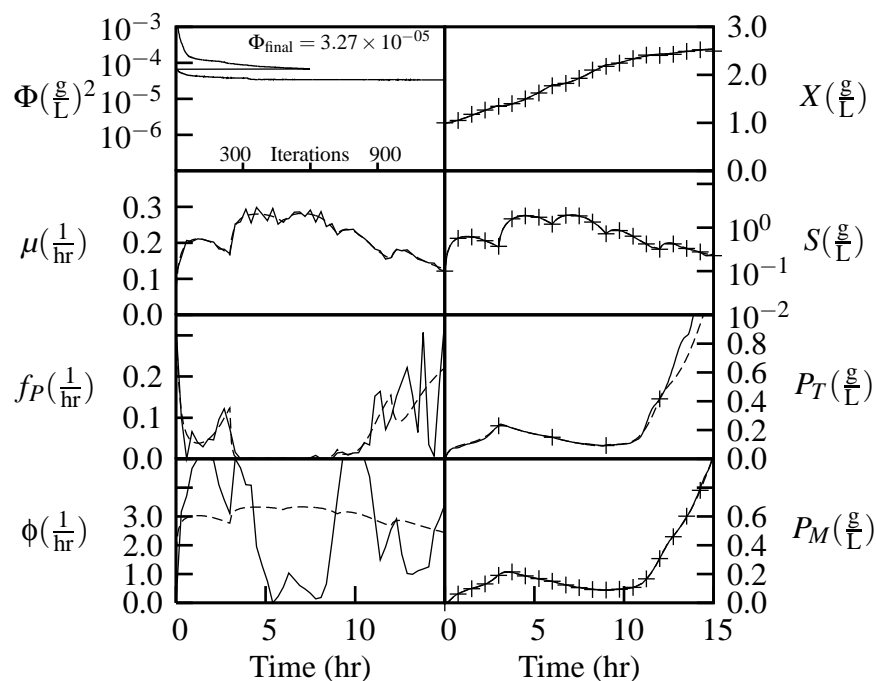


Figure B.1: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , P_T , and P_M using two-step IDP for data run 2 with **50 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **5 P_T data** and $\sigma_N = 0.01$. Control q^2 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

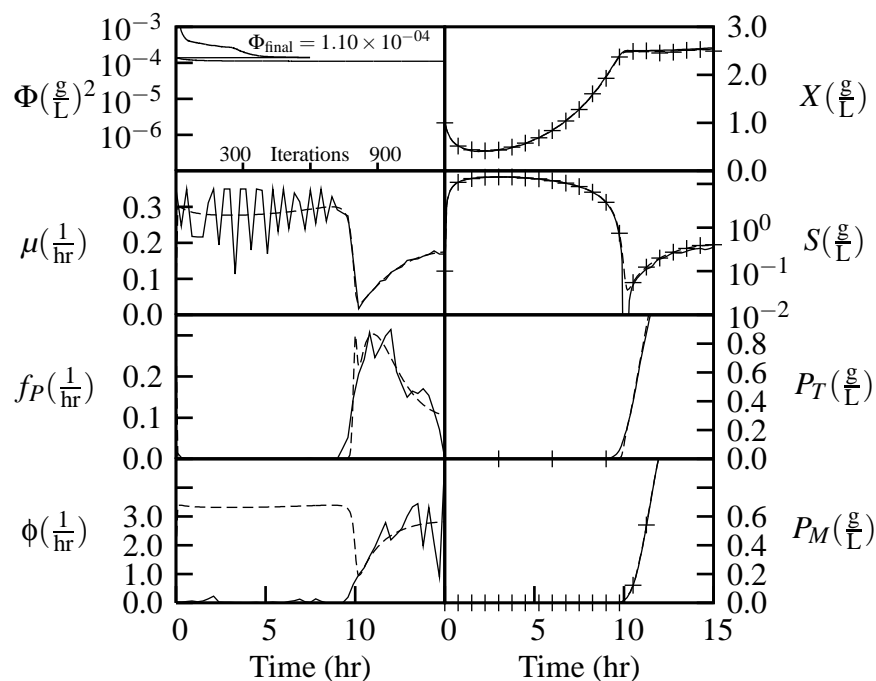


Figure B.2: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , P_T , and P_M using two-step IDP for data run 3 with **50 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **5 P_T data** and $\sigma_N = 0.01$. Control q^3 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

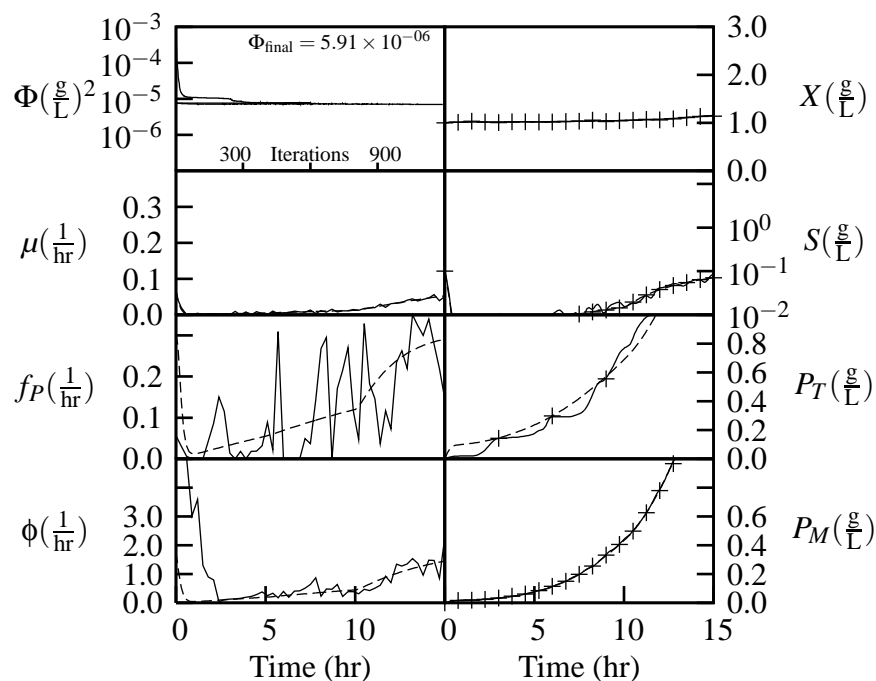


Figure B.3: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , P_T , and P_M using two-step IDP for data run 4 with **50 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **5 P_T data** and $\sigma_N = 0.01$. Control q^4 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

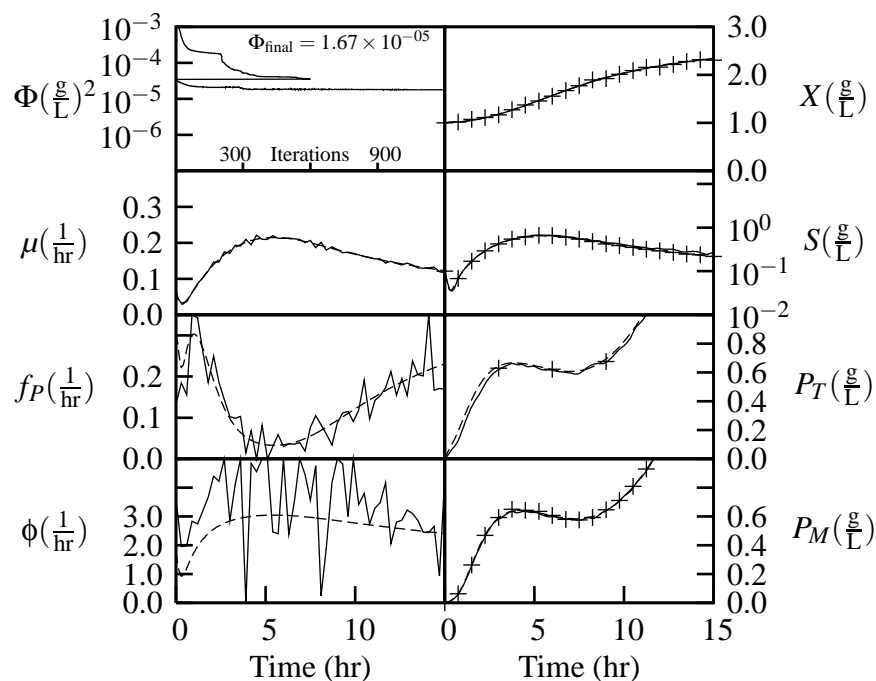


Figure B.4: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , P_T , and P_M using two-step IDP for data run 5 with **50 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **5 P_T data** and $\sigma_N = 0.01$. Control q^5 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

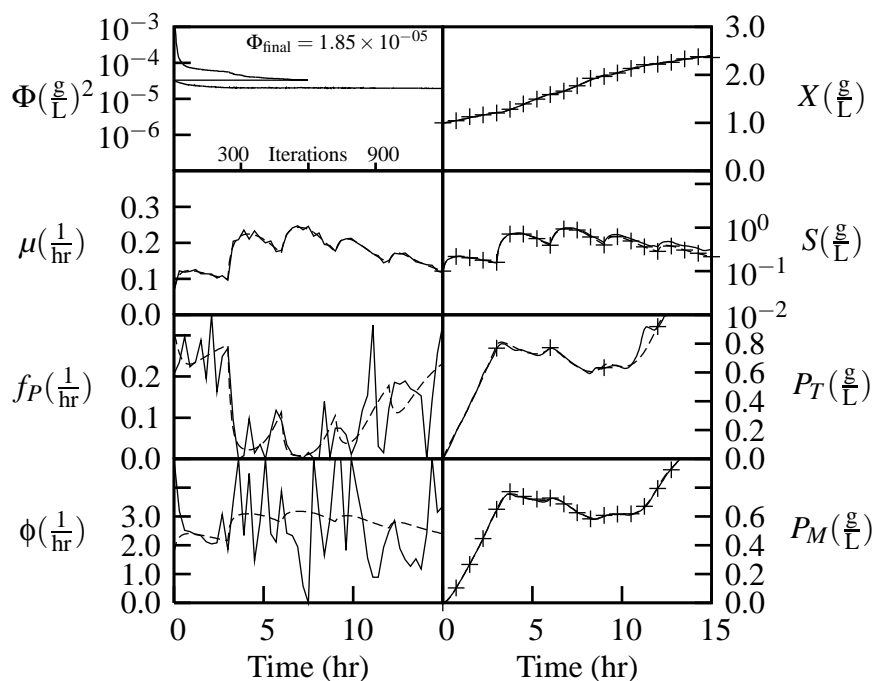


Figure B.5: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , P_T , and P_M using two-step IDP for data run 6 with **50 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **5 P_T data** and $\sigma_N = 0.01$. Control q^6 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

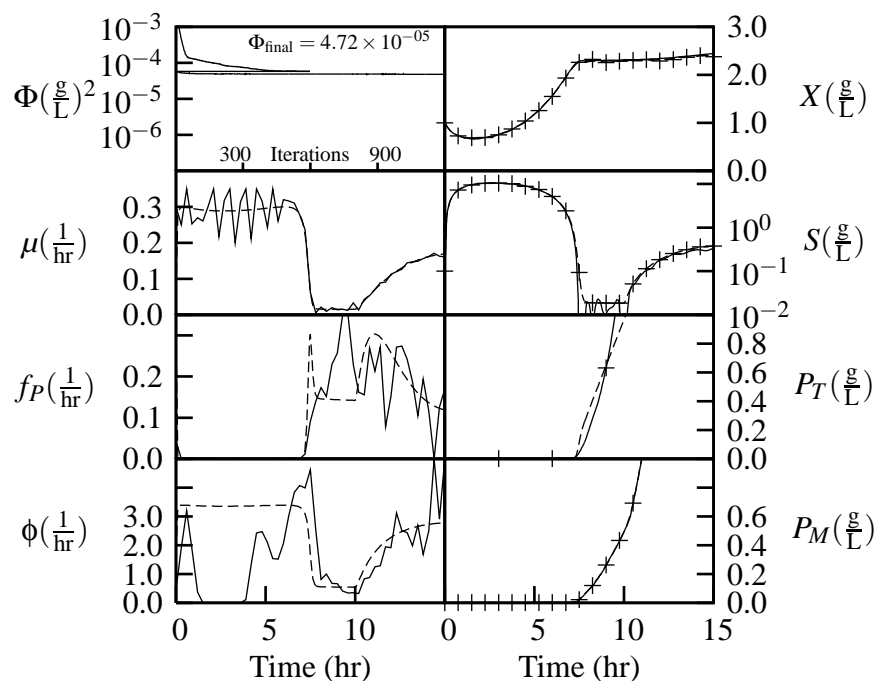


Figure B.6: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , P_T , and P_M using two-step IDP for data run 7 with **50 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **5 P_T data** and $\sigma_N = 0.01$. Control q^7 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.

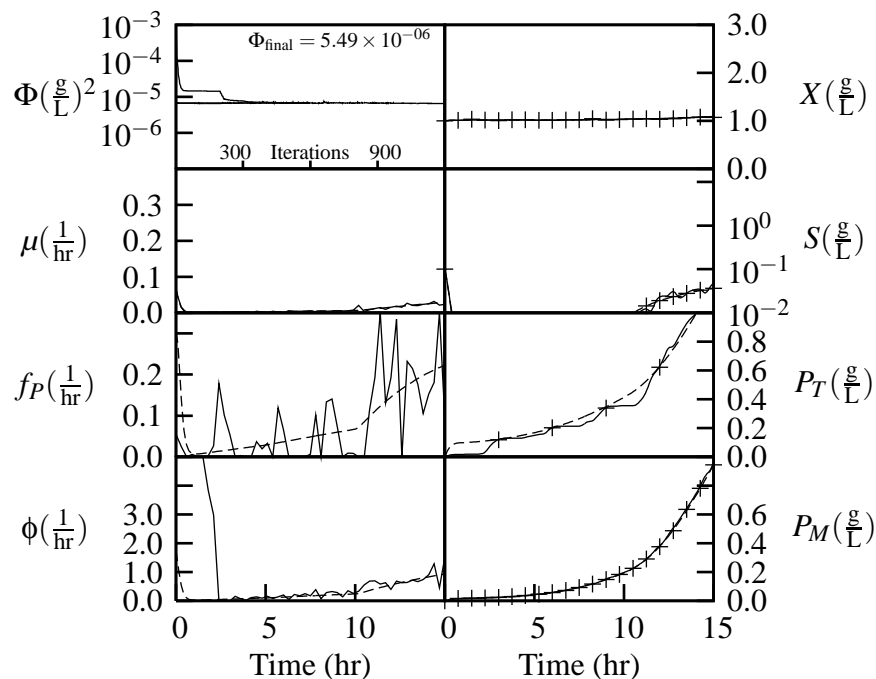


Figure B.7: Webb-Ramirez identification of the hybrid Park-Ramirez bioreactor model parameter functions $\mu(t)$, $f_P(t)$, and $\phi(t)$ from artificial data X , S , P_T , and P_M using two-step IDP for data run 8 with **50 stages**. The data generation and IDP parameters used are in Tables 6.1 (page 107) and 6.2 (page 111) with **5 P_T data** and $\sigma_N = 0.01$. Control q^8 from Figure 6.1 (page 105) was used to generate the artificial state data for the run in this figure, which are represented by the + symbol. Only every fifth X , S , and P_M datum is shown for clarity. Dashed lines are the parameter function and state values during the artificial data generation, and solid lines are the identified parameter function values and their corresponding states. The Φ plot shows the performance index vs. iterations and the final performance index. Iterations for steps one and two of two-step IDP are each counted from 1 which causes the discontinuity in the Φ plot.