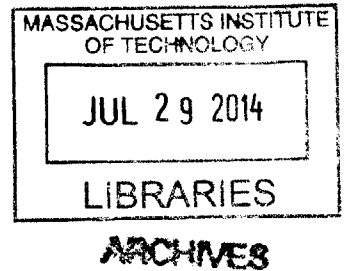


# Uncertainty Quantification in Safety Codes Using a Bayesian Approach with Data from Separate and Integral Effect Tests

by  
Joseph Paul Yurko  
S.B., Aerospace Engineering (2008)  
Massachusetts Institute of Technology  
S.M., Nuclear Science and Engineering (2010)  
Massachusetts Institute of Technology



Submitted to the Department of Nuclear Science and Engineering  
in partial fulfillment of the requirements for the degree of  
Doctor of Philosophy in Nuclear Science and Engineering  
at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2014


© 2014 Massachusetts Institute of Technology  
All rights reserved.

Signature redacted

Author .....

.....

Joseph Paul Yurko

 Department of Nuclear Science and Engineering  
May 20, 2014

Certified by .....

Signature redacted

Dr. Jacopo Buongiorno

Associate Professor of Nuclear Science and Engineering, MIT  
Thesis Supervisor

Certified by .....

Signature redacted

Dr. Robert Youngblood

Senior Risk Consultant, Idaho National Laboratory

Signature redacted Thesis Reader

Accepted by .....

.....

Mujid Kazimi

TEPCO Professor of Nuclear Engineering  
Chair, Department Committee on Graduate Studies



# Uncertainty Quantification in Safety Codes Using a Bayesian Approach with Data from Separate and Integral Effect Tests

by

Joseph Paul Yurko

Submitted to the Department of Nuclear Science and Engineering  
on May 20, 2014, in partial fulfillment of the  
requirements for the degree of  
Doctor of Philosophy in Nuclear Science and Engineering

## Abstract

Uncertainty quantification in thermal-hydraulic safety codes is a very challenging and computationally expensive endeavor. Methods are therefore needed to reduce that computational burden, while still providing a reasonable estimate for uncertainty. To do so, a Quantitative Phenomena Identification and Ranking Table (QPIRT) is implemented to screen down to key parameters that influence a figure of merit. From there, a surrogate model is built to approximate the complex input-output relationship of the safety code. The surrogate model type chosen is that of a probabilistic response surface following the Gaussian Process (GP) model framework. A GP prior is placed on the input/output functional relationship, which ultimately leads to a Bayesian non-parametric non-linear model of the safety code. The surrogate emulates the behavior of the long running computer code and thanks to the GP, provides a simple estimate to the additional uncertainty in making a prediction. In addition, for emulating multiple outputs together, which is difficult to do with standard GP models, Gaussian Process Factor Analysis (GPFA) models also known as Function Factorization with Gaussian Process Priors (FFGP) models were applied. The FFGP models are far more complicated than the standard GP model and so various simplifying approximations were made to enable fast yet accurate emulation of the safety code. All together a suite of surrogate models with varying levels of complexity and thus flexibility were developed for emulating the complex response from a safety code. These very computationally cheap surrogates are then used to propagate the uncertainty in the key parameters onto the FOM. Information from previous Separate and Integral Effect Tests is then used to calibrate those key

parameter distributions with Markov Chain Monte Carlo (MCMC). This allows the ultimate uncertainty of the figure of merit to be found conditioned on the knowledge gained from those past experiments.

Thesis Supervisor: Dr. Jacopo Buongiorno

Title: Associate Professor of Nuclear Science and Engineering, MIT

Thesis Reader: Dr. Robert Youngblood

Title: Senior Risk Consultant, Idaho National Laboratory

## Acknowledgements

This thesis is dedicated to Dr. Dana Kelly who tragically passed away near the start of the work. Without him, I might not have started to use the Gaussian Process, which ultimately became a central focus of my thesis. My future career ambitions have been greatly shaped by this work and Dr. Kelly deserves a lot of credit for pointing me in this direction.

I would like to thank my advisor Prof. Buongiorno for all his help and patience in completing this work. There were many ups and downs as I slowly figured out how to actually perform the task we set out to do. I want to thank Prof. Buongiorno for giving me the time to learn a lot of new things that I was not familiar with at the start. He offered me the flexibility to work on a project that I found very interesting and was very excited to be a part of.

Thank you Dr. Youngblood for joining this project and offering a lot of useful feedback and guidance along the way. Thank you for listening to many of my random and often long winded ideas on why (maybe) something was working, or not. You helped make sure I actually understood the point I was trying to make. Thank you for all of the help.

Many thanks to Prof. Todreas for giving really useful feedback on my thesis defense presentation slides.

Thank you to Dr. Mikkel Schmidt who personally answered my questions on the function factorization model.

Thank you to my fellow classmates in Course 22, especially everyone in my year.

Financial support for this research was provided by the Department of Nuclear Science and Engineering, the Idaho National Laboratory, and the Nuclear Regulatory Commission. Many thanks to the department staff that made sure I received that

funding.

I would also like to express my love and appreciation to my Mom, my Dad, and my two brothers, Nick and Ronnie, for putting up with the 10 years that I was away at school. It was a really long road and without your support I never would have made it this far.

Finally, I want to thank my beloved fiancée, Amy. Thank you for your patience and all of your support as I slowly got closer to finally finishing. I cannot wait to start the next chapter of our lives together. I love you.

# Contents

<b>1</b>	<b>Introduction</b>	<b>23</b>
1.1	Objectives and Contributions . . . . .	24
1.2	Organization of this Thesis . . . . .	26
<b>2</b>	<b>Bayesian Inference for Inverse Problems</b>	<b>29</b>
2.1	Inverse Problems . . . . .	29
2.2	Bayesian Inference . . . . .	31
2.3	Approximate Inference with MCMC . . . . .	34
2.3.1	Random-Walk Metropolis . . . . .	37
2.3.2	Adaptive Metropolis . . . . .	38
2.3.3	Demonstration Problem . . . . .	39
<b>3</b>	<b>Dimensionality Reduction Techniques</b>	<b>51</b>
3.1	QPIRT Methodology Overview . . . . .	52
3.2	Top Down Control Volume Balance Equations . . . . .	54
3.2.1	Local Level Equations . . . . .	55
3.2.2	System Level Balance Equations . . . . .	61
3.3	Feed and Bleed Transient Model . . . . .	68
3.3.1	RELAP5 Model . . . . .	68

3.3.2	RELAP Results . . . . .	70
3.4	QPIRT Analysis . . . . .	72
3.4.1	Local Level Top-Down Results . . . . .	73
3.4.2	System Level Top-Down Results . . . . .	81
3.4.3	Importance Threshold Value Sensitivity . . . . .	86
3.4.4	Bottom-Up Results . . . . .	90
<b>4</b>	<b>Surrogate Models using Gaussian Processes</b>	<b>93</b>
4.1	Surrogate Models Overview . . . . .	93
4.2	Standard Gaussian Process Regression . . . . .	96
4.2.1	Formulation . . . . .	96
4.2.2	Building the Emulator . . . . .	102
4.2.3	GPR Emulator-Based Uncertain Parameter Calibration . . . . .	109
4.3	Multi-Variate Output Emulators . . . . .	112
4.3.1	Function Factorization using Gaussian Process Priors . . . . .	114
4.3.2	FFGP Emulator-based Uncertain Parameter Calibration . . . . .	135
4.4	Calibration Demonstration . . . . .	136
4.4.1	One Uncertain Parameter FFGP model . . . . .	138
4.4.2	Two Uncertain Parameters FFGP model . . . . .	142
<b>5</b>	<b>Blowdown Problem Demonstration</b>	<b>159</b>
5.1	Blowdown Model . . . . .	165
5.1.1	Governing Equations . . . . .	168
5.1.2	Boundary and Initial Conditions . . . . .	169
5.1.3	Breakflow . . . . .	170
5.1.4	Heat Transfer Coefficients . . . . .	171



5.1.5	Physical Properties . . . . .	172
5.2	Numerical Scheme . . . . .	172
5.2.1	Spatial Discretization . . . . .	172
5.2.2	Time Discretization . . . . .	173
5.3	QPIRT Formulation . . . . .	175
5.3.1	Top-Down Equations . . . . .	175
5.3.2	Using the QPIRT . . . . .	178
5.4	Blowdown Model Results . . . . .	179
5.4.1	Nominal Case Results . . . . .	179
5.4.2	QPIRT Results . . . . .	182
5.5	Emulator-Based Calibration . . . . .	186
5.5.1	Building the Emulators . . . . .	186
5.5.2	Posterior Results . . . . .	194
5.5.3	Emulator Based Calibration Summary . . . . .	199
<b>6</b>	<b>EBR-II Model Calibration</b>	<b>207</b>
6.1	EBR-II . . . . .	209
6.1.1	Description of the Facility . . . . .	209
6.1.2	History of Testing at EBR-II . . . . .	215
6.2	Description of the chosen IET . . . . .	217
6.3	EBR-II IET RELAP Model . . . . .	219
6.3.1	RELAP User Defined Friction Factor Correlation . . . . .	224
6.3.2	RELAP Heat Transfer Models . . . . .	225
6.3.3	High and Low Pressure Piping Network . . . . .	226
6.3.4	Lower Plenums . . . . .	228
6.3.5	Core Channel . . . . .	232

6.3.6	Inner Blanket and Outer Blanket Channels . . . . .	236
6.3.7	Upper (Outlet) Plenum . . . . .	240
6.3.8	Conduction Within the Fluid Models . . . . .	240
6.3.9	Material Properties . . . . .	246
6.3.10	Boundary Conditions . . . . .	247
6.3.11	Initial Conditions . . . . .	249
6.3.12	Summary of Uncertain Parameters . . . . .	249
6.4	Pseudo SET RELAP Models . . . . .	252
6.4.1	Overview . . . . .	252
6.4.2	Core Channel SETs . . . . .	256
6.4.3	Blanket SETs . . . . .	276
6.5	IET Output Calibration Results . . . . .	288
6.5.1	LPP flow results . . . . .	290
6.5.2	TTC results . . . . .	299
6.5.3	OTC results . . . . .	306
6.6	Simultaneous Calibration . . . . .	317
6.6.1	Likelihood structure . . . . .	318
6.6.2	Core Channel SETs Simultaneous Calibration . . . . .	320
6.6.3	Blanket SETs Simultaneous Calibration . . . . .	324
6.6.4	All IET output Simultaneous Calibration . . . . .	325
6.6.5	All IET and SET simultaneous calibration . . . . .	335
<b>7</b>	<b>Summary, Conclusions, and Future Work</b>	<b>347</b>
7.1	Summary . . . . .	347
7.2	Conclusions . . . . .	352
7.3	Recommendations for Future Work . . . . .	353

7.3.1	Emulator Training with Variational Bayes . . . . .	353
7.3.2	Bayesian Monte Carlo . . . . .	354
7.3.3	Non-Informative Priors . . . . .	354
7.3.4	Dynamic Validation + Training + Calibration . . . . .	355
7.3.5	Uncertain Parameter Calibration Without MCMC . . . . .	357



# List of Figures

2-1	Sample Histories For Each MCMC Scheme . . . . .	42
2-2	Autocorrelation Lengths Over a Lag of 100 . . . . .	44
2-3	Estimated Posterior Densities . . . . .	46
2-4	Estimated Posterior Densities with 5% Observational Error . . . . .	47
2-5	Predictive Quantiles . . . . .	48
2-6	Posterior Covariance Structure . . . . .	49
3-1	RELAP5 Spatial Mesh Illustration . . . . .	64
3-2	RELAP5 Nodalization Modified for the Feed and Bleed Sequence . .	69
3-3	PCT for Multiple RELAP Runs with Different Operator Initiated Bleed Times . . . . .	71
3-4	Limiting Case Clad Temperature in Cell 11 Where Max PCT Occurs	73
3-5	Clad Temp. Weighted Contributions . . . . .	75
3-6	Clad Temp. Weighted Contributions During Phase II . . . . .	76
3-7	Fuel Temp. Weighted Contributions . . . . .	77
3-8	Liquid Temp. Weighted Contributions . . . . .	79
3-9	Liquid Temp. Weighted Contributions During Phase II . . . . .	80
3-10	Vapor Temp. Weighted Contributions . . . . .	81
3-11	Vapor Temp. Weighted Contributions During Phase II . . . . .	82

3-12 Sensitivity of the Number of Significant Processes (y-axis) to the Importance Threshold Value . . . . .	88
3-13 Sensitivity of the Fidelity of the Significant Processes to the Importance Threshold Value . . . . .	89
4-1 GPR Emulator Predictions . . . . .	107
4-2 Covariance Function Hyperparameter Histories . . . . .	109
4-3 True Functions in the Toy Problem . . . . .	127
4-4 Posterior Latent Training Variable Quantiles . . . . .	128
4-5 One Uncertain Parameter Training and Observational Data . . . . .	139
4-6 Posterior Observational Space Training Points . . . . .	140
4-7 Scaled Uncertain Parameter Sample Histories . . . . .	141
4-8 Estimated Scaled Posterior and Prior Densities . . . . .	142
4-9 Two Uncertain Parameters Training And Observational Data . . . . .	145
4-10 Friction Factor FFGP 2-factor 1-component likelihood noise hyperparameter . . . . .	147
4-11 Friction Factor FFGP 2-factor 2-component likelihood noise hyperparameter . . . . .	148
4-12 FFGP 2-factor 1-component calibrated posterior predictions . . . . .	150
4-13 FFGP 2-factor 2-component calibrated posterior predictions . . . . .	151
4-14 GPR calibrated posterior predictions . . . . .	152
4-15 GPR-based uncertain parameter posterior distributions . . . . .	155
4-16 1-component FFGP-based uncertain parameter posterior distributions	156
4-17 2-component FFGP-based uncertain parameter posterior distributions	157
5-1 Current emulator-based calibration approach flow chart . . . . .	163
5-2 Alternative data fusion flow chart . . . . .	164

5-3	Blowdown Tank Illustration . . . . .	166
5-4	Nominal Gas Pressure Results . . . . .	180
5-5	Nominal Temperature Results . . . . .	181
5-6	Fractional Contributions on the Gas Temperature . . . . .	183
5-7	Local Level Weighted Contributions . . . . .	184
5-8	Blowdown Model Training Data . . . . .	187
5-9	Blowdown GPR likelihood noise hyperparameter . . . . .	189
5-10	Blowdown GPR length scale hyperparameters . . . . .	190
5-11	Blowdown FFGP likelihood noise hyperparameters . . . . .	192
5-12	GPR Posterior Results . . . . .	197
5-13	FFGP 1-component-based posterior results . . . . .	203
5-14	FFGP 2-component-based posterior results . . . . .	204
5-15	FFGP 3-component posterior-based results . . . . .	205
5-16	FFGP 4-component-based posterior results . . . . .	206
6-1	EBR-II complex . . . . .	210
6-2	EBR-II Primary Sodium System . . . . .	212
6-3	EBR-II reactor arrangement . . . . .	213
6-4	EBR-II reactor vessel . . . . .	214
6-5	EBR-II primary pool Z-pipe view . . . . .	216
6-6	Transient Test No. 10 Core Outlet Temperature Data . . . . .	220
6-7	Transient Test No. 10 Top-of-Core Temperature Data . . . . .	221
6-8	Transient Test No. 10 Low Pressure Flow Rate Data . . . . .	222
6-9	EBR-II IET RELAP model schematic . . . . .	223
6-10	EBR-II lower plena . . . . .	229
6-11	Core and IB lower adaptor tubes within the IHPP . . . . .	230

6-12	Core subassemblies . . . . .	233
6-13	Core rods (DF and LB/UB) . . . . .	234
6-14	IB and OB subassemblies . . . . .	238
6-15	IB/OB bundle cross-section . . . . .	239
6-16	Nodalization diagram for axial heat conduction . . . . .	243
6-17	Axial conduction nodalization with multiple cells . . . . .	244
6-18	Axial and radial conduction illustration . . . . .	245
6-19	Total primary (inlet) flow rate boundary condition . . . . .	247
6-20	Total core power boundary condition . . . . .	248
6-21	G&G-1973 pressure drop “data” . . . . .	253
6-22	Generic “pseudo” SET RELAP model illustration . . . . .	255
6-23	CTDF FFGP Training set . . . . .	259
6-24	CTDF FFGP 2-factor 1-component likelihood noise hyperparameter . . . . .	260
6-25	CTDF FFGP 2-factor 2-component likelihood noise hyperparameter . . . . .	261
6-26	CTDF FFGP 2-factor 3-component likelihood noise hyperparameter . . . . .	262
6-27	CTDF FFGP 2-fact 1-comp-based calibration results . . . . .	264
6-28	CTDF FFGP 2-fact 2-comp-based calibration results . . . . .	265
6-29	CTDF FFGP 2-fact 3-comp-based calibration results . . . . .	266
6-30	CTDF FFGP Log-Joint-Posterior comparison . . . . .	269
6-31	GGcore FFGP training set . . . . .	272
6-32	GGcore FFGP 2-fact 3-comp likelihood noise hyperparameter . . . . .	274
6-33	GGcore FFGP 2-fact 3-comp-based calibration results . . . . .	275
6-34	IBCT FFGP training set . . . . .	277
6-35	IBCT FFGP 2-fact 3-comp likelihood noise hyperparameter . . . . .	279
6-36	IBCT FFGP 2-fact 3-comp-based calibration results . . . . .	280
6-37	IBGG FFGP training set . . . . .	281



6-38	IBGG FFGP 2-fact 3-comp likelihood noise hyperparameter . . . . .	282
6-39	IBGG FFGP 2-fact 3-comp-based calibration results . . . . .	283
6-40	OBBG FFGP training set . . . . .	285
6-41	OBBG FFGP 2-fact 3-comp likelihood noise hyperparameter . . . . .	286
6-42	OBBG FFGP 2-fact 3-comp-based calibration results . . . . .	287
6-43	LPP flow FFGP training set . . . . .	292
6-44	LPP GPR likelihood noise hyperparameter . . . . .	294
6-45	LPP GPR calibrated predictions . . . . .	295
6-46	LPP FFGP 2-fact 3-comp likelihood noise hyperparameter . . . . .	297
6-47	LPP FFGP emulator-based calibrated predictions . . . . .	299
6-48	LPP FFGP emulator-based calibrated uncertain parameter histograms	300
6-49	TTC FFGP training set . . . . .	302
6-50	TTC GPR likelihood noise hyperparameter . . . . .	303
6-51	TTC GPR calibrated predictions . . . . .	304
6-52	TTC FFGP 2-fact 4-comp likelihood noise hyperparameter . . . . .	307
6-53	TTC FFGP emulator-based calibrated posterior predictions . . . . .	308
6-54	TTC FFGP emulator-based calibrated uncertain parameter histograms	309
6-55	OTC FFGP training set . . . . .	310
6-56	OTC GPR likelihood noise hyperparameter . . . . .	311
6-57	OTC GPR calibrated predictions . . . . .	312
6-58	OTC FFGP 2-fact 5-comp likelihood noise hyperparameter . . . . .	314
6-59	OTC FFGP emulator-based calibrated predictions . . . . .	315
6-60	OTC FFGP emulator-based calibrated uncertain parameter histograms	316
6-61	Simultaneous CTDF and GGcore calibration uncertain parameter his- tograms . . . . .	321
6-62	Simultaneous calibration prediction results for CTDF . . . . .	322

6-63	Simultaneous calibration prediction results for GGcore . . . . .	323
6-64	Simultaneous Blanket SETs calibration uncertain parameter histograms	326
6-65	Simultaneous calibration prediction results for IBCT . . . . .	327
6-66	Simultaneous calibration prediction results for IBGG . . . . .	328
6-67	Simultaneous calibration prediction results for OBGG . . . . .	329
6-68	Simultaneous All IET output calibration uncertain parameter his- tograms . . . . .	331
6-69	Simultaneous calibration predictions results for the LPP output . . .	332
6-70	Simultaneous calibration prediction results for the TTC output . . .	333
6-71	Simultaneous calibration prediction results for the OTC output . . .	334
6-72	IET-SET simultaneous calibration uncertain parameter histograms .	337
6-73	IET-SET simultaneous calibration prediction results for CTDF . . . .	338
6-74	IET-SET simultaneous calibration prediction results GGcore . . . . .	339
6-75	IET-SET simultaneous calibration prediction results IBCT . . . . .	340
6-76	IET-SET simultaneous calibration prediction results IBGG . . . . .	341
6-77	IET-SET simultaneous calibration prediction results OBGG . . . . .	342
6-78	IET-SET simultaneous calibration prediction results LPP . . . . .	343
6-79	IET-SET simultaneous calibration prediction results TTC . . . . .	344
6-80	IET-SET simultaneous calibration prediction results OTC . . . . .	345

# List of Tables

- 3.1 Local Level Significant Processes . . . . . 74
- 3.2 System Level Phase Energy Significant Processes . . . . . 84
- 3.3 System Level Phase Momentum Significant Processes . . . . . 86
- 3.4 Bottom Up Step Important Closure Models and Empirical Correlations 91
  
- 5.1 Nominal Parameter Values . . . . . 167
- 5.2 Bottom-Up Step Uncertain Parameters . . . . . 185
- 5.3 Blowdown emulator build times . . . . . 193
- 5.4 Blowdown emulator-based calibration process computational times . . 202
  
- 6.1 Summary of IET model uncertain parameters . . . . . 251
- 6.2 “Pseudo” SET Names . . . . . 256
- 6.3 SET FFGP emulator type summary . . . . . 257
- 6.4 CTDF FFGP emulator build times . . . . . 259
- 6.5 CTDF Uncertain parameter calibration computational times . . . . . 270
- 6.6 GGcore FFGP emulator build times . . . . . 272
- 6.7 GGcore Uncertain parameter calibration computational times . . . . . 276
- 6.8 IET output emulator type summary . . . . . 291
- 6.9 IET output “best” emulator-based calibration computational times . 291

6.10 Summary of “best” FFGP emulators . . . . . 317

## Important Abbreviations

SET	Separate Effect Test
IET	Integral Effect Test
MCMC	Markov Chain Monte Carlo
RWM	Random Walk Metropolis
AM-MCMC	Adaptive Metropolis Markov Chain Monte Carlo
HMC	Hamiltonian Monte Carlo
QPIRT	Quantitative Phenomena Identification and Ranking Table
GP	Gaussian Process
GPR	Gaussian Process Regression
FF	Function Factorization
FFGP	Function Factorization with Gaussian Process priors
<i>i</i> -fact <i>k</i> -comp	<i>i</i> -factor <i>k</i> -component FFGP model
G&G-1973	Gopalakrishnan and Gillette (1973)
CTDF	Cheng & Todreas Driver Fuel bundle-specific $\Delta P$ SET
GGcore	Core Channel G&G-1973 $\Delta P$ SET
IBCT	Inner Blanket/Outer Blanket G&G-1973 bundle-specific $\Delta P$ SET
IBGG	Inner Blanket G&G-1973 channel $\Delta P$ SET
OBGG	Outer Blanket G&G-1973 channel $\Delta P$ SET
LPP	Low Pressure Plenum
TTC	Top-of-Core temperature
OTC	Core-Outlet-Temperature



# Chapter 1

## Introduction

Propagating parameter uncertainty for a nuclear reactor system code is a challenging problem due to often non-linear system response to the numerous parameters involved and lengthy computational times; issues that compound when a statistical sampling procedure is adopted, since the code must be run many times. Additionally, the parameters are sampled from distributions that are themselves uncertain. The current approach relies heavily on expert opinion for setting the assumed parameter distributions. Observational data is only used to validate predictions that “follow trends” and are “good enough” by “eyeball estimates”. All together, these complications lead to current uncertainty quantification (UQ) efforts relying on very conservative assumptions to make the problem feasible.

This work adopts a Bayesian framework that allows reducing the predictive uncertainty by calibrating parameters directly to observational data (also known as solving the inverse problem). Unlike the “eyeball” approach, Bayesian calibration is systematic and statistically rigorous, by calibrating the parameter distributions to the data, not simply tuning point values. With enough data, any biases from

expert opinion on the starting parameter distributions can be completely removed. Multiple levels of data are easier to handle as well, since Integral and Separate Effect Test (IET and SET) data can be used simultaneously in the calibration process. The only drawback of the Bayesian framework is the intense computational cost. It is more expensive than current UQ practices because the system code must be run tens to possibly hundreds of thousands of times in series. A very fast approximation to the system code is thus required to use the Bayesian approach. This work applies a relatively new class of statistical model, the Function Factorization with Gaussian Process (FFGP) priors model, to emulate the behavior of the system code. The FFGP model builds off of the more commonly used Gaussian Process (GP) emulator, but overcomes certain limiting assumptions inherent to the GP emulator. The FFGP model is therefore better suited to emulate the complex time series output produced by the system code. The surrogate is used in place of the system code to perform the parameter calibration, thereby allowing the observational data to directly improve the current state of knowledge.

## 1.1 Objectives and Contributions

The primary objective of this thesis is to perform Bayesian inference on system code (RELAP) models using both Separate and Integral Effect Test (SET and IET) data. In order to accomplish this task, a surrogate model better suited to emulate the complex time series behavior of the system code response needed to be developed. Additionally, many system code models have an enormous number of uncertain parameters that could be “tuned” to match data. Therefore a very fast, physics-based screening step needed to be developed to try and limit the number of important uncertain parameters to as few as possible.



The three main contributions of this thesis include:

- Development of a Quantitative Phenomena Identification and Ranking Table (QPIRT) methodology to identify significant uncertain parameters as viewed by the safety code.
- Development of an emulator-based calibration process for safety codes.
- Application of the emulator-based calibration process to the calibration of multiple RELAP models with multiple types of data, simultaneously.

The QPIRT provides a very fast screening step that ensures the selected parameters are involved only in the dominant physical processes controlling the figure-of-merit (FOM) and system response. Limiting the number of parameters that need to be inferred from the data improves the implementation of most computational Bayesian inference schemes. However, as will be described later, Bayesian inference is very computationally intensive. Thus even with limiting the focus to only the most important parameters, Bayesian inference is implausible due to the long run times of most safety codes. Computational Bayesian inference relies on drawing samples in series and therefore the safety code must be evaluated tens to possibly hundreds of thousands of times in series. Even if the safety code is “fast” and only takes 1 seconds to execute, running  $10^5$  samples in series would take over 27 hours to complete.

The emulator developed as part of this thesis is built off of the Gaussian Process (GP) framework, but overcomes some of the limiting assumptions within the GP model. This thesis is the first time the Function Factorization with Gaussian Process (FFGP) priors model - also known as Gaussian Process Factor Analysis (GPFA) - has been applied to emulate a reactor safety analysis code (RELAP). The FFGP emulator calibrated the uncertain parameters within a RELAP model of an EBR-II

loss-of-flow experiment. Additionally, data from various SETs were combined with the IET data to calibrate the uncertain parameters with the IET RELAP model simultaneously. This constituted a first of a kind Bayesian RELAP model.

## 1.2 Organization of this Thesis

Chapter 2 provides an introduction to Bayesian inference. Bayesian versus frequentist (classical) inference is compared to provide the reader with justification for why the Bayesian approach is used in this work. Approximate inference with Markov Chain Monte Carlo (MCMC) sampling is also discussed by applying several types of MCMC schemes to a simple inference problem. This chapter sets the foundation for why the emulator approach is required in order to use MCMC with safety codes.

Chapter 3 describes the QPIRT methodology in detail. The QPIRT equations are developed using the governing equations in RELAP5 and the process is applied to a Total Loss of main Feedwater Flow (TLOFW) accident with subsequent feed and bleed. The TLOFW demonstration shows how the QPIRT is capable of tracking through time and location, the dominant physical phenomena that influence the FOM.

Chapter 4 describes the emulators used in this work. The standard GP formulation is described first, before moving onto the FFGP emulator developed as part of this thesis. The FFGP (also known as the GPFA) model has been used in other fields before this thesis, but this is one of the first applications using the FFGP model to emulate a long-running computer code. This thesis is the first to do so in the nuclear field, however. In order to make predictions as quickly as possible with the FFGP model, several important approximations were made and they are discussed in detail in Chapter 4.

Chapter 5 applies the QPIRT + emulator-based calibration approach to the calibration of a gas blowdown model. The numerical solver is discussed in detail along with the QPIRT equations specific to the blowdown model. The QPIRT results demonstrate how easily it can screen down to a few key uncertain parameters, which are then used to build the various emulators developed in this work. The emulator-based results are then compared.

Chapter 6 applies the emulator-based calibration approach to an actual safety analysis scenario, calibrating an EBR-II RELAP loss-of-flow model. The EBR-II RELAP model is described in detail as well as the various SET models used in conjunction with the full EBR-II model (the IET model). The various emulators are compared for each of the models, and a simple way to choose the “best” emulator is described. After calibrating each model individually, all of the SET and IET data are used simultaneously.

Chapter 7 provides a summary of the thesis as discusses potential options for future work which could improve the emulator-based calibration approach even further.



## Chapter 2

# Bayesian Inference for Inverse Problems

### 2.1 Inverse Problems

Conceptually, inverse problems are very simple. We wish to estimate unknown objects, such as parameters or functions, from indirect noisy observations [1]. As a simple example, consider some output  $y$ , which is a function of the input  $x$ . The “forward problem” is simply computing  $y$  given  $x$ ,  $y = f(x)$ . But if we observe  $y$  and want to know the  $x$ -value that produced that particular  $y$ -value we need to compute the inverse function  $x = f^{-1}(y)$ . Thus, as the phrase “inverse problem” suggests, we want to solve a problem in reverse. To think of it another way, the forward problem is deductive logic: given a cause, we work out the consequences. Inverse problems however, require inductive logic: given certain observed effects, what are the underlying causes [2].

In the context of uncertainty quantification (UQ), forward UQ takes (assumed)

distributions on the input parameters and propagates those onto the output. This is straightforward to do with ordinary or standard Monte Carlo (MC) sampling (though it might be computationally expensive to do). Backwards UQ must then work in reverse, using the observed output to find the corresponding input parameter distributions. Therefore, backwards UQ is a special kind of inverse problem focusing on inferring out parameter distributions from noisy data. Another term used in the literature for this class of problem is model calibration. Though, some sources make the distinction that model calibration is a specific type of inverse problem with the goal of finding computer model input parameter values that result in outputs that agree well with observed data [1]. In the context of this thesis, all three terms backwards UQ, inverse problems and model calibration will all be used to mean essentially the same thing.

Although straightforward to understand, inverse problems are very challenging in practice. The inverse function may not exist, or even if it does, it might be nearly impossible to work out (such as finding the inverse function in a RELAP calculation). Additionally, many inverse problems are ill-posed in the sense that small perturbations in the output may lead to large errors in the inversion estimates [1]. For these reasons, the input parameters must be inferred from the observed output, which from here on will be known as the observational or measurement data. Statistical inference can be performed in one of several ways, using (classical) Frequentists or Bayesian methods. Probability, in the Bayesian sense, represents a *degree – of – belief* or plausibility: how much we think something is true, based on the available evidence. The Frequentist view however, defines probability as the long run relative frequency an event occurred, given (infinitely) many repeated trials [2]. There is a vast literature debating the pros and cons of the two approaches, regarding which I will not go into detail, if the reader is interested please consult References [3], [1], and

[4] (though admittedly these are very pro-Bayesian sources). Although some of the differences are philosophical, one of the major advantages of the Bayesian approach is that both the unknowns of interest and the data have distributions, whereas in the Frequentist approach the only variables that have distributions are the data. Also, estimators for the unknowns of interest must be chosen, which is not straightforward for non-linear problems, and those estimators are treated as fixed. The Frequentist approach would result in some confidence interval on the estimator of an input parameter, whereas the Bayesian approach results in a probability distribution on the input parameter itself. The primary objection to Bayesian inference is that it seems too subjective due to the use of the prior distribution (which will be described below). However, the prior may bring important structural information to the problem about the current state-of-knowledge about a parameter, from expert judgment or past experiments. Or, non-informative priors might be constructed to try and allow only the data to influence the solution to the inverse problem [5]. In the end, the main benefit of the Bayesian approach is that it is the proper logic of inference under uncertainty, which is the overall theme of this work.

## 2.2 Bayesian Inference

The fundamental concept of Bayesian analysis is that unknowns are treated as random variables. The power of this approach is that the established mathematical methods of probability theory are applied to develop informative representations of the state of knowledge regarding the unknowns [1]. Bayesian analysis relies on Bayes' theorem, which is a fundamental relationship of conditional probabilities. The unknowns are given what are commonly referred to as prior probability distributions, which as already mentioned reflect the current state of knowledge regarding their

possible values. These prior distributions are then refined or “updated” conditional on new information, through the likelihood function. The result is a posterior distribution that represents the new state-of-knowledge, as a combination of the prior evidence and the observed data. If  $\mathbf{x}$  is the vector of unknown parameters we want to learn more about, and  $\mathbf{y}$  is the vector of observed data, Bayes’ rule is written as:

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{\int p(\mathbf{y}|\mathbf{x})p(\mathbf{x})d\mathbf{x}}. \quad (2.1)$$

The numerator in Eq. 2.1 consists of the likelihood function,  $p(\mathbf{y}|\mathbf{x})$ , which gives the probability of the data given the unknown parameters, and the prior distribution on the unknowns,  $p(\mathbf{x})$ . The denominator is known by several names, the evidence, the marginal likelihood, or integral likelihood, since it integrates out (marginalizes) the unknown parameters. The denominator is therefore equivalent to,

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{x})p(\mathbf{x})d\mathbf{x}. \quad (2.2)$$

The evidence does not depend on the unknown parameters and is therefore a normalizing constant. Because of this, Bayes’ rule is commonly written simply as the posterior being proportional to the likelihood multiplied by the prior:

$$p(\mathbf{x}|\mathbf{y}) \propto p(\mathbf{y}|\mathbf{x})p(\mathbf{x}). \quad (2.3)$$

In statistics and probability sources, Eq. 2.3 is commonly written with  $\mathbf{x}$  and  $\mathbf{y}$  replaced with *hypothesis* and *data*, respectively. With this viewpoint, the power of Bayes’ rule lies in that it relates the quantity of interest, the probability the hypothesis is true given the data (the posterior), to the term we have a better chance at being able to assign, the probability that we would have observed the measured



data if the hypothesis was true (the likelihood) [2].

The difficulty in implementing Bayes' rule however, primarily stems from computing Eq. 2.2, the evidence. Evaluating the potentially (very) high dimensional integral could be intractable analytically, and/or very expensive computationally to compute. Analytical intractability may result from the fact that the likelihood function might involve a non-linear mapping between the unknowns and the observed data. This is the case for Bayesian model calibration of computer models, where the output of the computer model is a function of the unknown input parameters (as well as possibly other inputs which are not necessarily "unknown" such as time). The computer model will also be referred to as the computer code, the code, or sometimes as the simulator, and will be represented as  $f(\mathbf{x})$ . The likelihood function is therefore not between the unknown input parameters themselves, but between the output of the computer code and the observed data:  $p(y|f(\mathbf{x}))$ . With closed form solutions out of the picture, we have to resort to sampling based methods. The high computational cost of most sampling schemes was one of the barriers preventing more wide spread use of Bayesian inference for model calibration (though Laplace was the first to use Bayesian methods to infer the mass of Saturn from orbital observations in 1773). However due to faster computers, better numerical algorithms, and software packages such as R and OpenBUGS, Bayesian inference has really started to take off over the last two decades [6]. These sampling schemes are primarily Markov Chain Monte Carlo (MCMC) sampling strategies that try to draw samples from the posterior directly. The next section will briefly summarize several MCMC schemes used in this work.

## 2.3 Approximate Inference with MCMC

I will only summarize the key points of Markov Chain Monte Carlo (MCMC) that are important for its implementation. If the reader would like detailed discussions on its mathematical theory and formulation, please consult [7], [8], [9], and [4]. The basic idea behind MCMC is to construct a Markov chain whose stationary distribution is the target density we wish to sample [8]. The MCMC samples are theoretically guaranteed to be samples from the stationary distribution due to satisfying a condition known as detailed balance. Detailed balance ensures that a Markov chain is ergodic [8] which essentially means that all states are recurrent. In the context of Bayesian model calibration, that target density is the posterior distribution of the unknown parameters given the data,  $p(\mathbf{x}|\mathbf{y})$ . The theory of MCMC is similar to the theory of ordinary MC, except that now we are drawing correlated samples rather than independent and identically distributed (iid) samples. The samples are correlated due to the Markov chain and is the price we pay for not knowing the actual posterior distribution, but rather only the likelihood and the prior densities. The main benefit of MCMC is that it does not need to compute the evidence, Eq. 2.2 and therefore never needs to compute a potentially very high dimensional integral. MCMC gets around that integral by essentially a scientific method of “guessing and checking” which will become more clear below. The main downside however is there is no clear stopping point to MCMC. There are various heuristics and rules of thumb for when to stop the sampling, but in general determining when to stop can be quite tricky. For those reasons, some refer to MCMC as more of an art more than a science, with even the OpenBUGS user’s manual containing the following caution: “Beware: MCMC sampling can be dangerous!” [10]. That being said, MCMC is a very powerful tool that when used properly allows very high dimensional problems to be tackled in a

consistent framework.

All MCMC strategies involve some sort of random walk around the state space of the unknown parameters of interest. The more naive the sampler, the more random that walk is, while more efficient samplers may try to suppress the randomness through various mechanisms. The key ingredients in MCMC are:

- an initial guess,  $\mathbf{x}^o$
- a proposal (or transition) distribution,  $q(\mathbf{x}^*|\mathbf{x})$ , where  $\mathbf{x}^*$  is the proposed sample for the unknown parameters of interest
- an accept/reject rule for whether to keep the proposed sample or not

Finding a good initial guess for the unknown parameters might be very difficult. Sometimes the prior mean is a good starting point, but if the posterior distribution is quite different from the prior it might take the sampler a long time to move the parameter values to the correct region of the state space. The chain will gradually “forget” its initial state and will eventually converge to a unique stationary distribution, which does not depend on the sample number [7]. A poor starting point therefore really only impacts the length of time it takes for the chain to “forget”. But if a limited number of samples are used, that initial sequence may impact the results. That is why in practice it is common to discard a portion of the samples as “burn-in”, thereby neglecting the potentially poorly sampled values at the beginning. [9] discusses how burn-in is not actually needed in MCMC, but I use it as a simple approach to “find” a good starting value for the MCMC sampler.

The proposal distribution,  $q(\mathbf{x}^*|\mathbf{x})$ , is what determines the efficiency of the MCMC sampling scheme. The more naive and therefore more random the proposal, the less efficient the MCMC scheme is. Usually efficiency relates to how correlated the

samples are, where a more efficient proposal distribution allows the MCMC chain to “forget” the initial state faster. The proposal distribution is also what usually characterizes a particular MCMC sampling scheme. The specific types of proposal distributions I used in this work usually are Gaussian proposals centered on the current guess (random walk behavior) with a fixed covariance structure. The specific proposal distribution types will be discussed in more detail below.

The most common accept/reject rule is the Metropolis-Hastings (MH) acceptance criteria. The Metropolis criteria was developed specifically for symmetric proposal distributions, then extended to asymmetric proposals through the Hastings correction. The MH rule compares the posterior of the proposal value to the posterior of the current guess value. If the posterior increases, then accept the proposed value, but if the proposed posterior value is less than the current posterior value, randomly choose to accept the proposed value or not with acceptance probability (for symmetric proposals):

$$\alpha_M = \min \left( 1, \frac{p(\mathbf{x}^*|\mathbf{y})}{p(\mathbf{x}^t|\mathbf{y})} \right), \quad (2.4)$$

where  $t$  is the current sample in the chain. Thus, if the proposed value,  $\mathbf{x}^*$ , is less probable than the current guess,  $\mathbf{x}^t$ , we may still move there anyway. Instead of greedily moving to only more probable states, we occasionally allow “downhill” moves to less probable states [8]. This kind of behavior ensures that the fraction of time spent in each state  $\mathbf{x}$ , is proportional to the posterior distribution  $p(\mathbf{x}|\mathbf{y})$ . The other important point to note about Eq. 2.4 is that because it deals with ratios of the posterior density, the normalizing constant, the evidence (or marginal likelihood) cancels out. Hence, we are able to draw samples from the posterior distribution even if we cannot compute Eq. 2.2.

In order to decide when to stop the MCMC chain, an important heuristic is the

“mixing rate”. Explaining if a chain is “well-mixed” is best done through demonstrations and will be shown later. But in general, a “well-mixed” chain should not show a noticeable trend in the sampling pattern, which reflects a correlated structure in the samples. This kind of effect can be quantified a little better through the autocorrelation of the samples over some specified lag-length.

### 2.3.1 Random-Walk Metropolis

The simplest MCMC scheme is the Random-Walk Metropolis (RWM) sampler with a symmetric Gaussian proposal distribution. Here, the proposal distribution is centered at the current guess for the unknown parameters,  $\mathbf{x}^t$ , with a specified covariance matrix,  $s^2\Sigma$ . The scalar  $s^2$  is some constant set to facilitate “rapid mixing” [8]. Usually the covariance matrix  $\Sigma$ , is taken to be diagonal, thereby assuming uncorrelated parameters. If the unnormalized posterior density is written as  $h(\mathbf{x}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x})$ , the RWM update has the following steps:

- for current guess,  $\mathbf{x}^t$ , with current unnormalized posterior  $h(\mathbf{x}^t)$ , propose a move to  $\mathbf{x}^*$  with proposal density:  $q(\mathbf{x}^*|\mathbf{x}^t) = \mathcal{N}(\mathbf{x}^t, s^2\Sigma)$
- compute the proposed unnormalized posterior  $h(\mathbf{x}^*)$
- compute Metropolis acceptance probability using Eq. 2.4
- set the next sample value to be:  $\mathbf{x}^{t+1} = \begin{cases} \mathbf{x}^*, & \text{with probability } \alpha_M \\ \mathbf{x}^t, & \text{with probability } 1 - \alpha_M \end{cases}$

This update sequence is repeated the desired number of times. The scalar constant in the proposal covariance matrix,  $s^2$ , can be set by tuning the number of acceptances during the sampling chain to meet some optimal rate. The optimal rate comes about from the fact that if a proposal is very large, and so  $\mathbf{x}^*$  is very different from  $\mathbf{x}^t$ , the

chances of being accepted and thus the chances of the chain moving might be rather low and thus the chain will most likely “sit” at the same value for a long time. If the proposal is very close to the current guess, the chances of being accepted are very high, but very little new information is gained. An optimal “middle ground” exists where the proposal jumps “just far enough” at any one particular step in the MCMC sampling. [11] prove that for Gaussian proposals the asymptotically optimal value is  $s^2 = 2.38^2/D$ , where  $D$  is the number of parameters in  $\mathbf{x}$ . They also found that the optimal acceptance rate is 0.234 - so about 23% of the all the proposed samples are accepted. The asymptotic value is valid as the dimension  $D$  approaches infinity, but for smaller dimensional problems the asymptotic value may not necessarily be the best choice. Thus, the scalar constant may be tuned to try and force the sampling to stay near the optimal acceptance rate of 0.234. If the scalar constant is tuned, it can only be tuned during the burn-in phase and must be remained fixed afterwards. This is because the tuning impacts the memoryless-ness nature of the Markov chain. But, as long as the tuning is not continued through the entire sequence then the chain will gradually “forget that the tuning occurred”.

### 2.3.2 Adaptive Metropolis

Tuning the scalar constant,  $s^2$ , above is a very simple adaptive scheme. In effect tuning  $s^2$  adapts the variance of each element in  $\mathbf{x}$  equally, since it is important to remember that  $\mathbf{x}$  is a vector of all the unknown input parameters:  $\mathbf{x} = [x_1, x_2, \dots, x_D]^T$ . It is possible to tune the proposal variance of each element sequentially, but as the dimension  $D$  grows, this could become prohibitively expensive. Haario et al. (2001) set up an Adaptive Metropolis (AM) scheme that adapts the entire proposal covariance matrix  $\Sigma$ , by using the past samples of the Markov chain [12]. This allows

the proposal distribution to capture the covariance structure between the various elements in  $\mathbf{x}$ , which can greatly improve the efficiency of the MCMC sequence. To denote the fact that the proposal covariance matrix is no longer “hard-coded” but computed on the fly, it will be labeled as  $\hat{\Sigma}$ , and the proposal distribution is now:  $q(\mathbf{x}^*|\mathbf{x}^t) = \mathcal{N}(\mathbf{x}^t, \hat{\Sigma})$ . A single update step is identical to the update in the RWM except the proposal distribution uses  $\hat{\Sigma}$  instead of  $s^2\Sigma$ .

Early on, a set of pilot runs are used where the proposal covariance matrix is not updated, and is the only part of the AM sampling scheme that the potentially poor “hard-coded” guess proposal covariance matrix is used. After the initial pilot phase, adaptation of the proposal covariance matrix is allowed by using a specified number of past samples, after a certain number of iterations. For example, every 1000 iterations, the past 1000 samples will be used to compute empirical covariance matrix,  $\Sigma$ . [12] found this approach more computationally efficient than computing the empirical covariance matrix at every iteration using all of the past samples.

### 2.3.3 Demonstration Problem

To demonstrate Bayesian model calibration with approximate inference using MCMC, a method of manufactured solutions procedure is used where the observational data is generated using the computer model itself at specified true-parameter values. The goal is to then see if the MCMC sampling procedure is capable of yielding posterior distributions around these true-parameter values. The “computer model” is a simple turbulent friction factor model parameterized as:

$$f = \exp(b) Re^{-\exp(c)}, \quad (2.5)$$

where  $Re$  is the Reynolds number and two uncertain parameters:  $b$  and  $c$ . Eq. 2.5 is usually written as  $f = B \cdot Re^{-C}$ , but it was easier to parameterize the coefficient and exponent in the model to always have positive values by transforming them as:  $B = \exp(b)$  and  $C = \exp(c)$ . This transformation made it easier to use Gaussian distributions for the priors on the unknown parameters, which gives the original parameters,  $B$  and  $C$  to have log-normal prior distributions. With  $\mathbb{E}[x]$  denoting the mean value of  $x$ , the prior mean values were chosen so that the  $\exp(\mathbb{E}[b]) = 0.184$  and  $\exp(\mathbb{E}[c]) = 0.2$  (which correspond to  $B$  and  $C$  at the McAdams' friction factor correlation values). Each of the transformed parameters were then assumed to have prior variances so that 95% of the probability covered  $\pm 50\%$  around the prior means.

Twenty-five “data” points were generated at the specified true-parameter values at evenly spaced intervals between Reynolds numbers of 5000 and 45000. Using notation consistent with earlier sections, the data is denoted  $\mathbf{y} = [y_1, y_2, \dots, y_{N_O}]^T$ , where  $N_O = 25$  is the number of observational data points. The simple friction factor model shows that usually multiple types of input variables exists. The  $b$  and  $c$  parameters are the unknown or uncertain parameters, but the Reynolds number,  $Re$ , is not unknown. In fact, there are specified  $Re$  values that the data is “located” at that we wish to compare predictions against. Variables such as the Reynolds number are known commonly as control variables [13]. Other examples of control variables include time, boundary conditions such inlet mass flow rates, or perhaps the pressure that an experiment is conducted at. For notation purposes, the uncertain parameters will be denoted collectively as  $\theta = [b, c]^T$ , and the control variables will be denoted as  $\mathbf{x}_{cv} = [Re_1, Re_2, \dots, Re_{N_O}]^T$ . Denoting the prior means and variances as  $\mu_b, \mu_c$



and  $\sigma_b^2$ ,  $\sigma_c^2$ , respectively, the prior distribution is a multivariate normal (MVN):

$$p(\theta) = \mathcal{N} \left( \begin{bmatrix} \mu_b \\ \mu_c \end{bmatrix}, \begin{bmatrix} \sigma_b^2 & 0 \\ 0 & \sigma_c^2 \end{bmatrix} \right) \quad (2.6)$$

A Gaussian likelihood function is used to relate the data to the model prediction:

$$p(\mathbf{y}|f(\mathbf{x}_{cv}, \theta)) = \mathcal{N}(f(\mathbf{x}_{cv}, \theta), \Sigma_\epsilon). \quad (2.7)$$

The observational error covariance matrix is assumed to be diagonal, with the observational error (variance) to have  $\pm 10\%$  of the data value covering 95% of the probability.

Three different MCMC schemes are used to draw samples from the posterior distribution on the uncertain parameters,  $p(\theta|\mathbf{y})$ . All three use 5e4 burn-in samples and 5e4 posterior samples. The first is a RWM-sampler with the scalar constant  $s^2$  hard-coded to be  $s^2 = (1.2)^2$ . The second is a tunable-RWM where the  $s^2$  value is tuned to maintain the acceptance rate near the optimal value of 0.234. The tuning does not start till after the 200<sup>th</sup> iteration, and then checks the acceptance rate every 100<sup>th</sup> iteration. The last is the AM sampler, which does not adapt during the first half of burn-in (the first 2.5e4 samples) and then keeps the proposal covariance matrix fixed during the posterior sampling phase. When adapting, it computes the empirical covariance matrix every 1000 iterations using the past 1000 samples.

The sample histories for the three MCMC schemes for both the  $b$  and  $c$  parameters are shown below in Fig. 2-1.

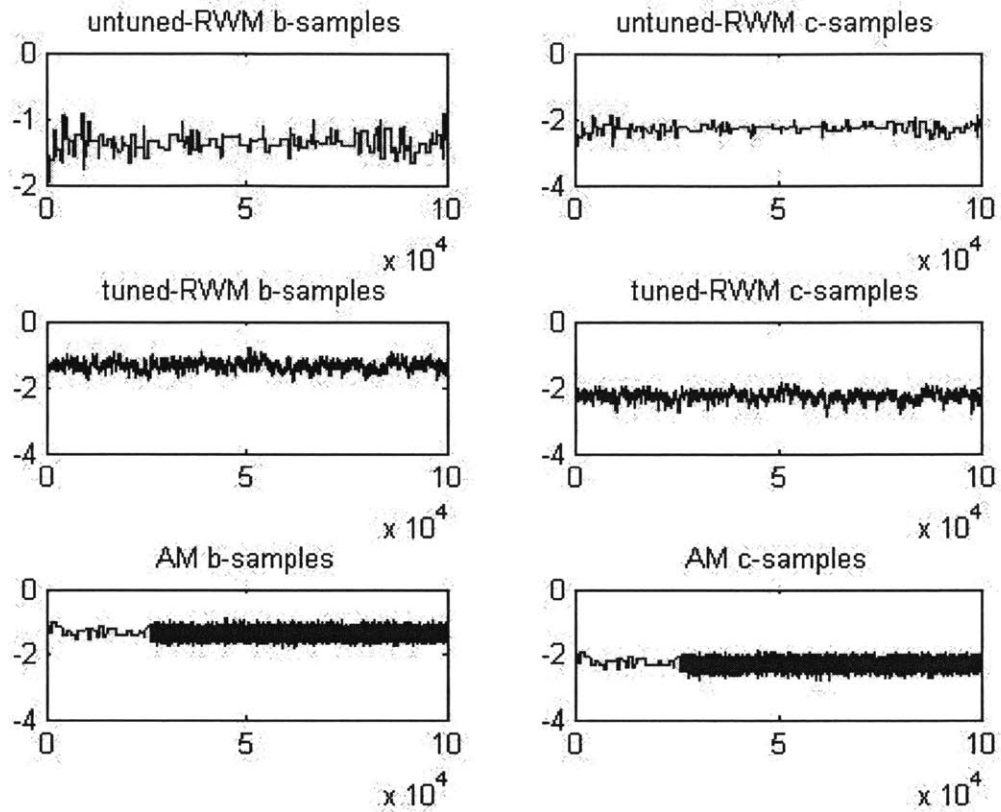


Figure 2-1: Sample Histories For Each MCMC Scheme

Fig. 2-1 illustrates poor, medium, and “well-mixed” behaviors. The top plots show the sample histories for the untuned RWM scheme. There are long stretches where the parameter values seem to “sit” still, which correspond to a large number of rejects. The middle plots show the sample histories for the tuned-RWM scheme, and although the mixing appears a lot better than in the untuned-RWM case, these samples are clearly not as “well-mixed” as the AM results, shown in the last row of plots. Before adapting the proposal covariance matrix, the samples shown in the

bottom row look very close to the untuned-RWM samples. But, once the adaption phase starts for the AM scheme, it is difficult to distinguish any noticeable trend in the sampling. This is idea of well-mixed sampling behavior, passing an “eye-ball” test. Quantifying “well-mixed” can be done using the autocorrelation length over a specified lag. Fig. 2-2 gives the autocorrelation length for the three schemes over a lag of 100. As shown below, the autocorrelation in the AM scheme drops off close to zero rather quickly, while both RWM schemes show that there still exists a very high correlation between the samples.

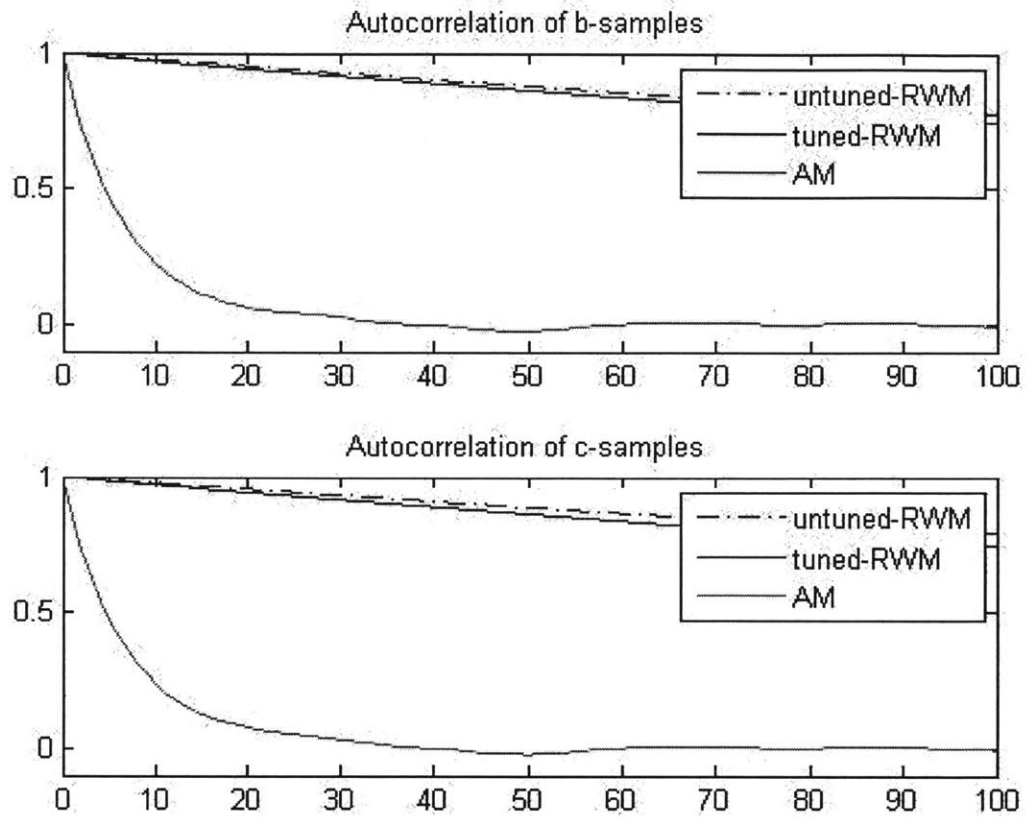


Figure 2-2: Autocorrelation Lengths Over a Lag of 100

But, even with the tuned-RWM scheme not being truly “well-mixed” the posterior samples can still give very good results. As shown in Fig. 2-3, the estimated posterior density using the tuned-RWM samples is very close to the estimated density from the AM samples. The correlation amongst the posterior samples impacts the MC standard error, with the standard error increasing as the correlation increases [9]. Since the tuned-RWM samples are much more tightly correlated than the AM samples, the effective number of independent samples in the tuned-RWM results is

very small. Whereas, for the AM results, about every 30<sup>th</sup> sample is effectively an independent sample draw from the posterior. For more challenging problems, with less data and more uncertain parameters, the medium mixing rate of the tuned-RWM samples might be an issue. But for a simple problem such as this, the increased MC standard error does not really impact the accuracy of the posterior samples at finding the true parameter values. The true value for the  $c$  parameter was intentionally placed rather far from the prior mean, at nearly two prior standard deviations away. But, the posterior sample modes are very close to the true  $c$  value, which is the red vertical line in the plots. As a check, the MCMC sampling was run again with the observational error cut in half, which should reduce the posterior variance and have the posterior modes even closer to the true parameter values - if the MCMC sampling is working correctly. As shown in Fig. 2-4 that is indeed the case, though it is somewhat difficult to see the posterior modes in the figure. The density shapes were estimated using the built-in MATLAB function `ksdensity.m`.

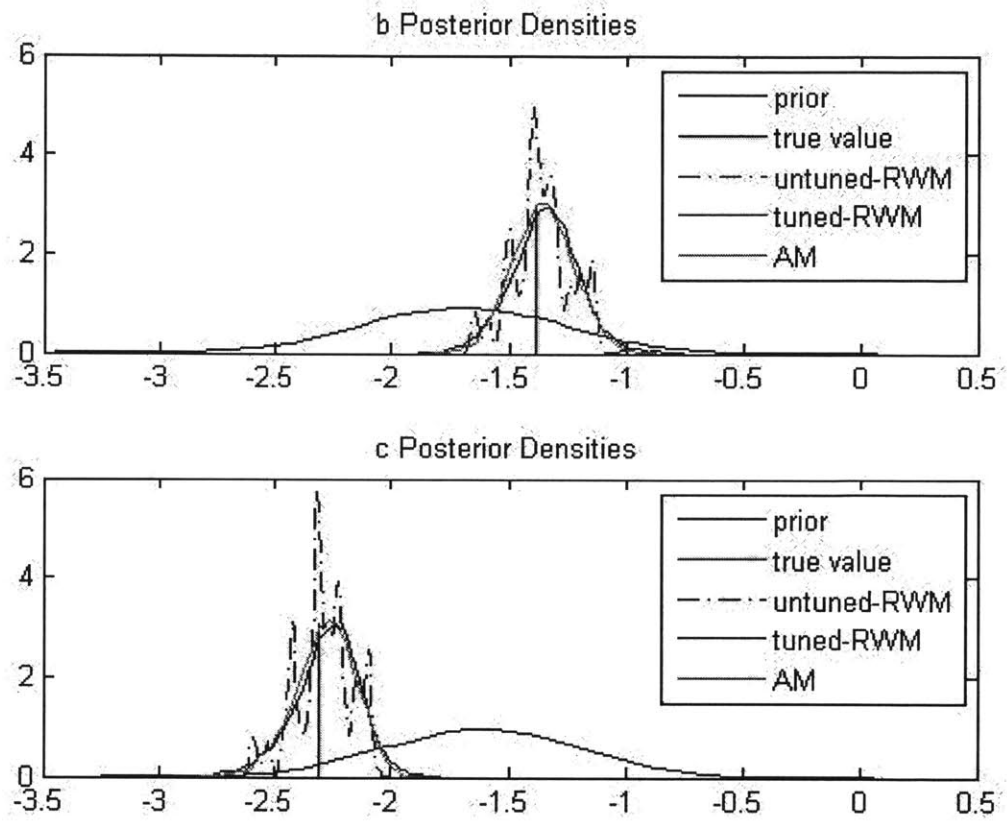


Figure 2-3: Estimated Posterior Densities

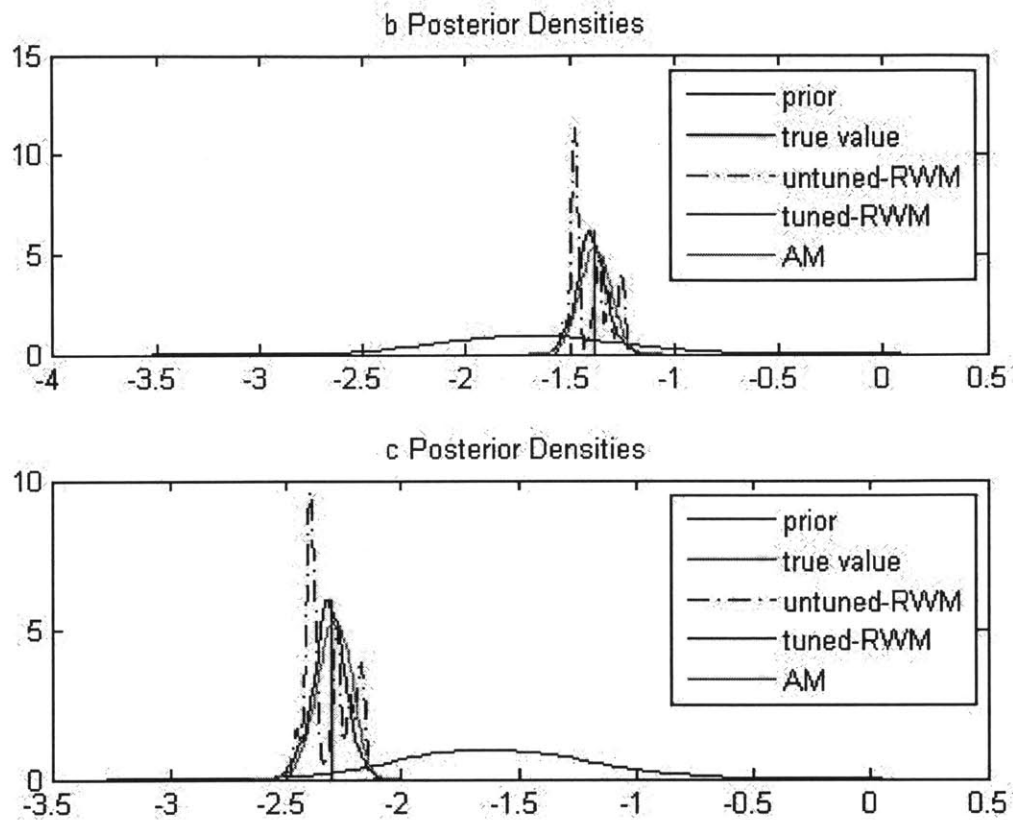


Figure 2-4: Estimated Posterior Densities with 5% Observational Error

As for the posterior predictions, the AM results show that the posterior output predictions follow the observational data very well. Fig. 2-5 shows the prior and posterior predictive quantiles at the 5%, 25%, 50%, 75%, and 95% cumulative probabilities in grey and blue, respectively. The observational data has error bars representing two standard deviations around the observed data mean values. In addition, the posterior covariance structure can be computed and is shown for the AM case results in Fig. 2-6. This figure was generated using the built-in MATLAB

function plotmatrix which places histograms along the diagonals and shows that the two parameters are very tightly correlated. The tight posterior correlation was not represented at all in the assumed prior distribution which means that correlation resulted from the data entirely.

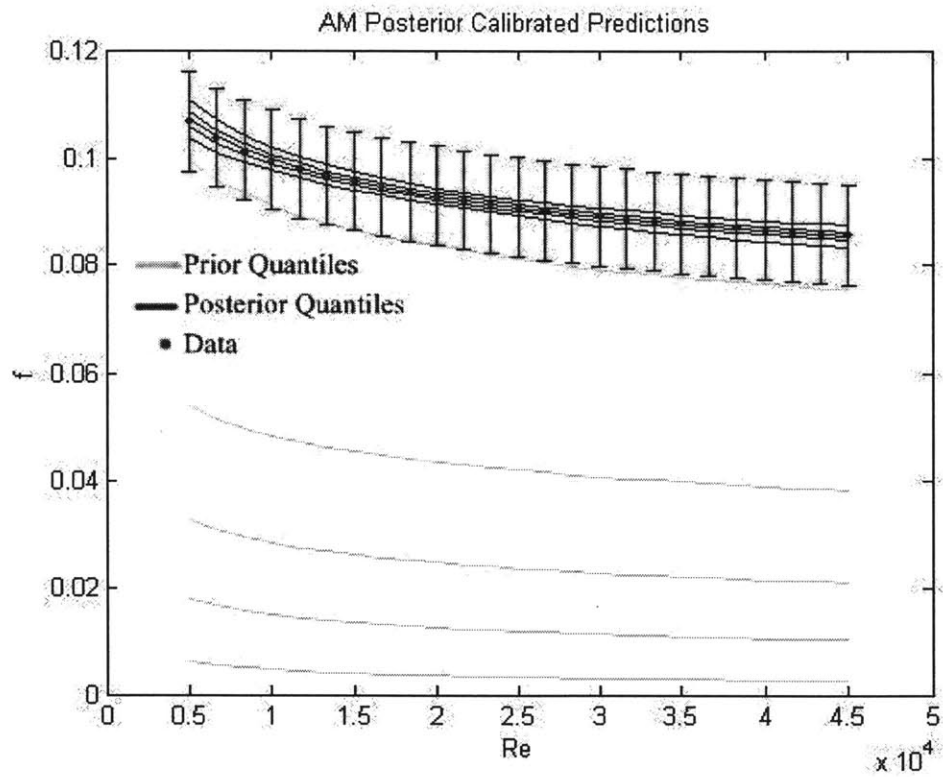


Figure 2-5: Predictive Quantiles



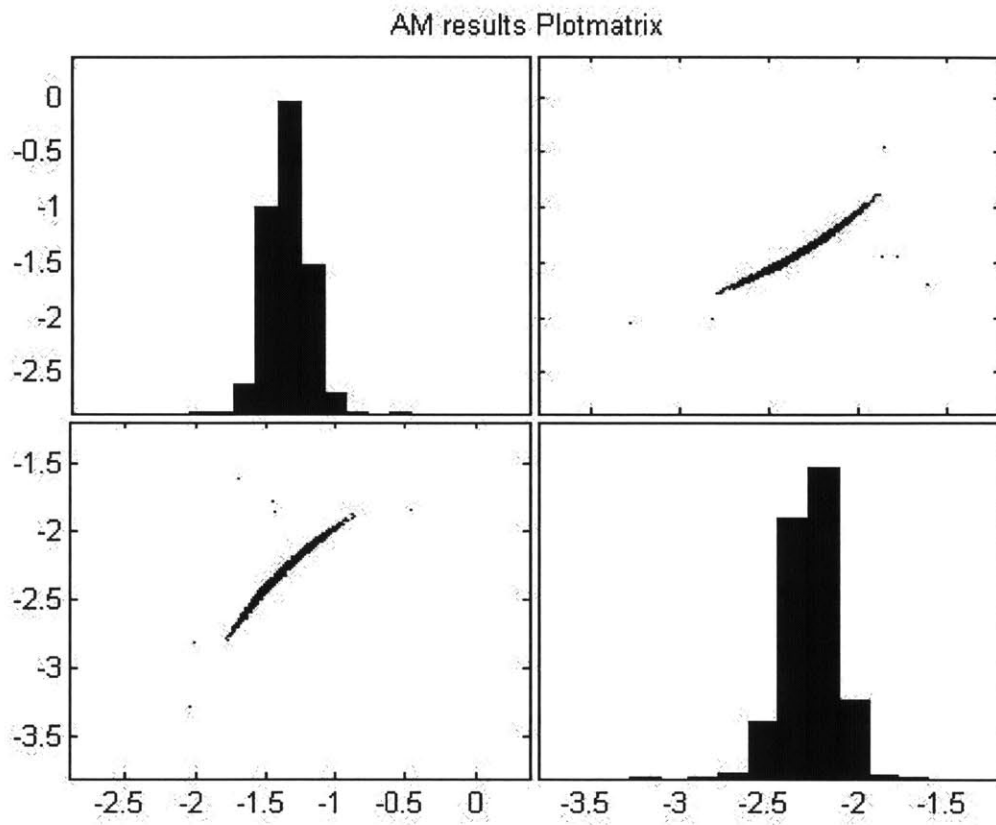


Figure 2-6: Posterior Covariance Structure



## Chapter 3

# Dimensionality Reduction

## Techniques

Before attempting to calibrate a model it is important to try and limit the number of parameters we have to work with. This is even more important in our case since we will ultimately be building surrogate models to approximate the long-running computer codes, and use those surrogates in their place when performing MCMC sampling. Limiting the number of input parameters limits the number of training runs that are required which ultimately means the entire process is faster. In addition, MCMC techniques become more complicated to implement as the number of parameters increases, and especially if a large number of those parameters are (tightly) correlated. In situations with a very large number of very correlated parameters, MCMC techniques exist but their computational cost starts to grow considerably [14]. So if we want to maintain fast inference methods, we need to limit the number of parameters we will ultimately be sampling with MCMC.

Reducing the number of inputs, also known as dimensionality reduction, can be

done in a wide variety of ways. Most are statistical based approaches that require some form of sampling [15]. Since we will be sampling with MCMC, we want to screen out the unimportant parameters in as few code runs as possible. For that reason, we have chosen to adapt the Phenomena Ranking and Identification Table (PIRT) process to rank physical phenomena using quantifiable measures from the code output directly. This Quantitative PIRT or QPIRT, is an engineering approach to screening that focuses on finding the most important physical processes that influence our key output metric, or figure of merit (FOM). Once those key physical processes are identified, their various empirical correlations and models are examined to pull out their uncertain or “tunable” parameters.

### 3.1 QPIRT Methodology Overview

The QPIRT process consists of two primary steps, with the overall goal being to identify parameters that significantly influence the figure of merit (FOM) for a specific transient of interest. The first step, the Top-Down step, determines which physical processes dominate the value of the FOM, by focusing on the governing equations used in the safety code (the two-fluid six equation model in RELAP5, for example). The second step, the Bottom-Up, step then examines in detail the dominant processes identified by the Top-Down step by analyzing the specific correlations used in the safety code to model those processes. Specific “tunable” parameters for each correlation are then identified, as well as relationships to other processes. Together, the Top-Down and Bottom-Up steps ensure that only physically relevant parameters are chosen for uncertainty propagation.

The Top-Down step itself consists of several hierarchical length scales. This approach is similar to the hierarchical scales used in the Zuber Pi-group approach in

[16]. The multiple scales are required because the physical phenomena governing the whole primary loop response may be different from the ones governing flow along the hot fuel rod within the core, for example. The multiple scales used in the QPIRT methodology consist of local (or cell volume) level, component, and system level. For each scale, the governing equations are cast in control volume balance form. The local level analyzes the governing equations affecting the value of the FOM directly, thereby ensuring the parameters that directly impact the FOM receive sufficiently detailed analysis. The specific FOM is solved for from the control volume balance equations and the various processes are weighted by their contribution onto that value. Processes with the largest weighted contributions are therefore considered dominant. This concept will be made clearer in the next section which discusses the various equations for a specific example transient.

With the local level scale capturing phenomena directly impacting the FOM, the higher level scales are required to capture indirect effects from components throughout the rest of the system. The component level scale has results from each cell in a particular component volume averaged together. The system level scale then has results from individual components volume averaged together over specific component groups (for example if there are several pipe component sections in the hot leg). The system level scale results in a system of equations with various physical phenomena occurring in multiple component groups. The various terms in the system level equations are weighted by their overall magnitude (no value in particular is solved for in the system level equations) where the processes with the largest magnitude are considered dominant. The system level balance equations therefore ensure the dynamic behavior of the system as a whole is properly characterized.

Results of the Top-Down step give a time history of the physical phenomena that dominate the FOM locally as well as a time and spatial history of physical

phenomena that dominate the system behavior, as viewed by the safety code. The Bottom-Up step then analyzes the correlations used to model each of these dominant physical phenomena. The Bottom-Up step is therefore where two-phase flow specifics such as flow regimes become important because even if the same process dominates throughout a transient (such as heat transfer rate to the liquid phase) the flow regime will dictate which specific correlations are used to model that process. At the conclusion of the Bottom-Up step, the QPIRT generates a list of the specific correlations along with their “tunable” parameters that act through the dominant physical processes to influence the FOM.

The easiest way to describe the QPIRT process is to demonstrate it on a physically relevant problem. The Top-Down equations will be formulated for a Total Loss of main Feedwater Flow (TLOFW) accident with subsequent feed and bleed. The next several sections will go through all of the equations that lead to the Top-Down step identifying the dominant physical phenomena during the TLOFW accident.

## **3.2 Top Down Control Volume Balance Equations**

As described previously, the Top-Down focuses on control volume balance formulations of the governing equations in the safety code. The local level consists of identifying relevant constituents involved in the transient and formulating appropriate balance equations consistent with the safety code. The equations will be problem dependent at the local level since the physical phenomena involved depend on the specific FOM. For example, if clad oxidation rate is the FOM, different local level equations will be used than if max peak clad temperature (PCT) is the FOM. The system level equations should most likely only depend on the system nodalization and code model used, since this level captures effects “carried throughout” the system.

The following equation formulations will be given for cases where max PCT is the FOM, such as for the feed-and-bleed transients currently being described. At the local level the four constituents involved are the clad, fuel, liquid phase and vapor phase. As will be seen, whenever a particular constituent's associated temperature value is significant in the local level clad energy equation, that constituent's energy equation is analyzed to see what influences that particular temperature value. System-level energy and momentum equations for the vapor and liquid phases are formulated to be consistent with the RELAP5 code model.

### 3.2.1 Local Level Equations

At the local, or cell, level the following energy control volume balance equations are used. For the clad:

$$\frac{d}{dt} [V \langle \rho c T \rangle_C] = -\dot{Q}_{C-L} - \dot{Q}_{C-V} + \dot{Q}_{F-C}. \quad (3.1)$$

The subscripts in Eq. 3.1 are  $C$  for the clad,  $L$  for the liquid phase,  $V$  for the vapor phase, and  $F$  for the fuel. The brackets  $\langle \cdot \rangle$  represent the volume averaging operator. The fuel energy balance equation is:

$$\frac{d}{dt} [V \langle \rho c T \rangle_F] = \dot{Q}_{GEN} - \dot{Q}_{F-C}. \quad (3.2)$$

The general control volume internal energy equation for either phase from the RELAP5 Manual (taken from RELAP5 Manual Vol. 1 Section 3.1.10.1.1) is:

$$\frac{d}{dt}(\alpha_k \rho_k u_k V) + \sum_j (A \alpha_k \rho_k u_k v_k)_j = \dot{Q}_w V + \dot{Q}_i V - \left[ P \frac{d\alpha_k}{dt} V + \sum (P A \alpha_k v_k)_j \right] + h_i \Gamma V + D_S V. \quad (3.3)$$

The subscript  $k$  denotes the phase, where  $l$  represents liquid and  $v$  for vapor. Note that the symbol  $V$  in Eq. 3.3 denotes the coolant cell volume, while the same symbol in Eq. 3.1 and 3.2 denotes the clad and fuel volumes, respectively. The summation terms in Eq. 3.3 are the sums of the convective internal energy flowing and flow work into and out of the control volume at the junction faces. The term,  $\dot{Q}_w V$ , is the heat from the clad transferred to the phase in the cell volume, while  $\dot{Q}_i V$  represents the interfacial heat transfer in the cell volume. The mass transfer term is  $h_i \Gamma V$  and the last term,  $D_S V$ , is the viscous dissipation term. The processes in Eqs. 3.1 - 3.3 will be defined in the subsequent sections.

### Clad Energy Equation

Equation 3.1 simply states that the time rate of change of the volume averaged internal energy of the clad is the balance of the heat transferred between the clad-to-liquid, clad-to-vapor, and fuel-to-clad. Applying Newton's law of cooling, the heat transfer process can be expressed in terms of heat transfer coefficients:

$$\frac{d}{dt} [V \langle \rho c T \rangle_C] = -h_L S_{CO} (T_{CO} - T_L) - h_V S_{CO} (T_{CO} - T_V) - h_{GAP} S_{GAP} (T_{FO} - T_{CI}). \quad (3.4)$$

In Eq. 3.4,  $h$  is the heat transfer coefficient,  $S$  denotes surface area, the subscript  $CO$  denotes the outer clad position,  $FO$  denotes the outer fuel position, and  $CI$  denotes



the inner clad position. The heat transfer between the fuel and clad is represented by the heat transfer through the gap using the gap conductance  $h_{GAP}$  and effective gap heat transfer area  $S_{GAP}$ . For cases where the FOM is the PCT, the outer clad temperature,  $T_{CO}$  is solved for from Eq. 3.4. First define the area heat transfer coefficients:

$$H_L = h_L S_{CO}, \quad H_V = h_V S_{CO}, \quad H_{GAP} = h_{GAP} S_{GAP}, \quad (3.5)$$

as well as the effective heat transfer coefficient acting on the outer clad:

$$H_{CO} = H_L + H_V. \quad (3.6)$$

Thus giving the outer clad temperature to be:

$$T_{CO} = \frac{H_{GAP}}{H_{CO}} (T_{FO} - T_{CI}) + \frac{H_L}{H_{CO}} T_L + \frac{H_V}{H_{CO}} T_V - \frac{1}{H_{CO}} \frac{d}{dt} [V \langle \rho c T \rangle_C]. \quad (3.7)$$

Equation 3.7 shows that the temperature values associated with each process as well as the fuel thermal capacitance are scaled by the effective heat transfer coefficient acting on the coolant side of the clad.

### Fuel Energy Equation

Equation 3.2 compares the internal heat generation rate in the fuel with the heat transfer rate to the clad. Using the definitions defined previously, Equation 3.2 is used to solve for the associated temperature for the fuel-to-clad heat transfer:

$$(T_{FO} - T_{CI}) = \frac{\dot{Q}_{GEN}}{H_{GAP}} - \frac{1}{H_{GAP}} \frac{d}{dt} [V \langle \rho c T \rangle_F]. \quad (3.8)$$

The fuel-to-clad heat transfer rate is thus associated with the temperature jump across the gap. Equation 3.8 examines the fuel's contribution to that temperature jump by scaling the heat source term and fuel thermal capacitance by the effective heat transfer coefficient through the gap.

### Phase Temperature Energy Equations

The remaining processes of the right hand side of Eq. 3.3 are given as (using RELAP5 notation), for the liquid phase:

$$\dot{Q}_i V = \left[ H_{if} (T^S(P_S) - T_l) - \left( \frac{1+\epsilon}{2} \right) \Gamma_W (h'_g - h'_f) \right] V, \quad (3.9)$$

$$D_S V = DISS_f V = V (\alpha \rho v^2)_l FWF, \quad (3.10)$$

$$h_i \Gamma V = - \left[ \Gamma_{ig} h_f^* + \Gamma_W h'_f \right] V. \quad (3.11)$$

And for the vapor phase:

$$\dot{Q}_i V = \left[ H_{ig} (T^S(P_S) - T_v) - \left( \frac{1-\epsilon}{2} \right) \Gamma_W (h'_g - h'_f) \right] V, \quad (3.12)$$

$$D_S V = DISS_g V = V (\alpha \rho v^2)_g FWG, \quad (3.13)$$

$$h_i \Gamma V = - \left[ \Gamma_{ig} h_g^* + \Gamma_W h'_g \right] V. \quad (3.14)$$

In the interfacial heat transfer terms given by Eqs. 3.9 and 3.12, the first term on the right hand side in the bracket represents the interfacial heat transfer in the bulk fluid, while the second term represents the interfacial heat transfer in the region “near the wall”. Likewise, in the mass transfer terms given by Eqs. 3.11 and 3.14, the first term on the right hand side in the brackets models the mass transfer in the bulk fluid, while the second models the mass transfer “near the wall”.

The heat transfer rate with the clad and the bulk fluid interfacial heat transfer are used to define a total effective heat transfer coefficient acting on each phase. For the liquid this is given as:

$$H_{tot}^L = H_L + H_{if}V, \quad (3.15)$$

and for the vapor phase:

$$H_{tot}^V = H_V + H_{if}V. \quad (3.16)$$

Using the above definitions, Eq. 3.3 is solved for each phase temperature. The liquid phase temperature is given as:

$$\begin{aligned} T_L = & -\frac{1}{H_{tot}^L} \frac{d}{dt} (\alpha_l \rho_l u_l V) - \frac{1}{H_{tot}^L} \sum_j (A \alpha_l \rho_l u_l v_l)_j \\ & - \frac{V}{H_{tot}^L} \left( \frac{1+\epsilon}{2} \right) \Gamma_w (h'_g - h'_f) \\ & + V \frac{H_{if}}{H_{tot}^L} T^S(P_s) + \frac{H_L}{H_{tot}^L} T_{CO} - \frac{PV}{H_{tot}^L} \frac{d\alpha_l}{dt} - \frac{P}{H_{tot}^L} \sum_j (A \alpha_l v_l)_j \\ & - \frac{V}{H_{tot}^L} \Gamma_{ig} h_f^* - \frac{V}{H_{tot}^L} \Gamma_w h'_f + \frac{V}{H_{tot}^L} (\alpha \rho v^2)_l FWF. \end{aligned} \quad (3.17)$$

And for the vapor phase temperature:

$$\begin{aligned}
T_V = & -\frac{1}{H_{tot}^V} \frac{d}{dt} (\alpha_v \rho_v u_v V) - \frac{1}{H_{tot}^V} \sum_j (A \alpha_v \rho_v u_v v_v)_j \\
& - \frac{V}{H_{tot}^V} \left( \frac{1-\epsilon}{2} \right) \Gamma_W (h'_g - h'_f) \\
& + V \frac{H_{ig}}{H_{tot}^V} T^S (P_S) + \frac{H_V}{H_{tot}^V} T_{CO} - \frac{PV}{H_{tot}^V} \frac{d\alpha_v}{dt} - \frac{P}{H_{tot}^V} \sum_j (A \alpha_v v_v)_j \\
& + \frac{V}{H_{tot}^V} \Gamma_{ig} h_g^* + \frac{V}{H_{tot}^V} \Gamma_W h'_g + \frac{V}{H_{tot}^V} (\alpha \rho v^2)_v FWG. \tag{3.18}
\end{aligned}$$

### Weighted Contributions

As seen in each constituent's energy equation, the various processes are scaled by the effective heat transfer coefficient acting on that particular associated temperature value. The value of each term in the Eqs 3.7, 3.8, 3.17, and 3.18 (the equations that solve for each associated temperature value) is considered to represent the contribution onto the FOM from a particular physical phenomena. Each term is then scaled, or weighted by the sum of absolute value of each term in the equation, thereby defining the weighted contribution of each physical phenomenon on the FOM. For example, for the clad-to-liquid heat transfer process, the associated weighted contribution is:

$$W_L = \frac{\left| \frac{H_L T_L}{H_{CO}} \right|}{\left| \frac{H_{GAP}}{H_{CO}} (T_{FO} - T_{CI}) \right| + \left| \frac{H_L T_L}{H_{CO}} \right| + \left| \frac{H_V T_V}{H_{CO}} \right| + \left| \frac{1}{H_{CO}} \frac{d}{dt} [V \langle \rho c T \rangle_C] \right|}. \tag{3.19}$$

Weighted contributions for the remaining physical processes in the clad energy equation as well as in the other constituent energy equations are defined similarly.

The phase energy equations, however, complicate the weighted contributions be-

cause of the convective energy flow terms. Convective energy outflow terms are considered “effects” from the various heat transfer processes in a cell, not a “cause”. Therefore, the convective energy outflow terms are neglected from the weighted contribution denominator terms. This assumption can be justified using the method of characteristics.

### **3.2.2 System Level Balance Equations**

The system-level balance equations correspond to summing up all the component-level balance equations. Thus, the system-level balance equation also represents the component-level balance equation, just on a bigger scale. Each physical process can then occur in each component group. To be general, the following system-level equations will represent the number of component groups in the system by just labeling the summation term with “ALL”. Also, from looking at Eq. 3.3, summing up around the entire system cancels out the inflow and outflow terms at all locations except at system wide source and sink locations (for example, flow from the ECCS or flow out of a break). The source and sink flow terms will be represented by “ $\Delta$ ” to show it is the difference between the system wide inflow and outflow terms.

#### **Phase Energy Equation**

The system-level phase energy equations follow directly from the local-level phase energy equation, given by Eq. 3.3. After computing the component-averaged process values and then component-group averaged process values and summing up around the entire system, the system level phase energy equation is given by:

$$\begin{aligned}
& \sum_n^{ALL} \left[ \frac{d}{dt} (\langle \alpha_k \rho_k u_k \rangle_n V_n) \right] + \Delta \left[ (A \alpha_k \rho_k u_k v_k)_{sys} \right] \\
& = \sum_n^{ALL} \left[ \langle \dot{Q}_W \rangle_n V_n \right] + \sum_n^{ALL} \left[ \langle \dot{Q}_i \rangle_n V_n \right] - \sum_n^{ALL} \left[ \left\langle P \frac{d\alpha_k}{dt} \right\rangle_n V_n \right] \\
& \quad \sum_n^{ALL} \left[ \langle h_i \Gamma \rangle_n V_n \right] + \sum_n^{ALL} \langle D_S \rangle_n V_n.
\end{aligned} \tag{3.20}$$

The subscript  $n$  denotes a particular component group.

Unlike the local level equations, weighted contributions to a specific value are not computed for the system level energy equations. Instead, a normalized fractional contribution is computed by scaling each term in Eq. 3.20 by the sum of the absolute values of all the terms. Doing this allows the max possible value of a fractional contribution at any timestep to be 1. The largest magnitude fractional contribution terms are then considered to dominate the system average phase energy at that particular time.

## Phase Momentum Equations

Since the system level equation results from summing up the various component groups in the system, the individual cell volume momentum equation will first be described. The different processes will be defined at the cell level with the corresponding system level processes defined just as in the system energy equation. The

phase control volume balance momentum equation in a particular cell is:

$$\begin{aligned}
 \frac{d}{dt} (\alpha_k \rho_k v_k V) + \sum_j (A \alpha_k \rho_k v_k v_k)_j \\
 = F_{wk} + F_{formk} + F_{visc,sk} + F_{VM,k} \\
 - \sum_j (PA \alpha_k)_j + \Gamma v_{sk} V + (\alpha_k \rho_k) B_z.
 \end{aligned} \tag{3.21}$$

Equation 3.21 is simply a force balance between all the forces acting on a particular phase in a cell volume. The first term on the LHS is the phase inertia and the second term is the acceleration of the phase at the control volume boundaries. The first three terms on the right hand side of Eq. 3.21 are the wall friction, form loss, and interfacial friction forces, respectively. The fourth term is the virtual mass force acting on phase  $k$ . The fifth term is the pressure force acting at the boundaries of the control volume. The sixth term is the momentum transfer due to mass exchange at the interface and the last term is the body force.

Before defining the friction terms, it is important to note that the RELAP5 phase momentum equations are not applied to the same cell volumes that the mass and energy equations are applied to. The RELAP5 difference equations for energy (and mass), on the cell volume level, balance the rate of energy (or mass) into and out of the cell boundaries with the source terms. This model requires the energy (and mass) terms defined as volume averaged properties while requiring knowledge of the velocities at the cell boundaries [17]. Momentum cells are therefore defined with their centers at the mass/energy cell boundaries, resulting in a staggered spatial mesh. Scalar properties such as pressure, internal energy, and void fraction are defined at the mass/energy cell centers while vector quantities such as velocity are defined at the cell boundaries. Using the RELAP5 notation the cell boundaries are

referred to as junctions. An illustration of this setup, from Volume I of the RELAP5 manual is given by Fig. 3-1.

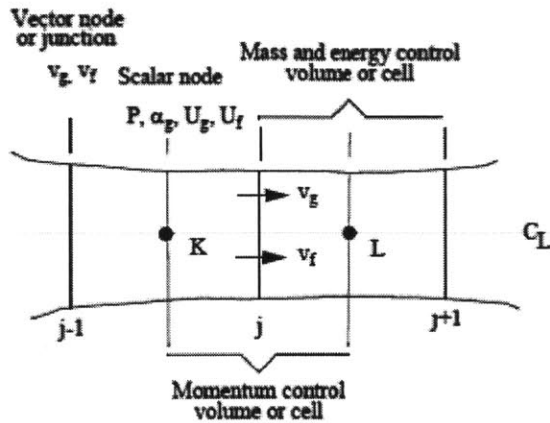


Figure 3-1: RELAP5 Spatial Mesh Illustration

The RELAP5 numerical scheme then integrates the mass/energy equations from junction to junction position over each mass/energy cell and integrates the momentum equations from cell center to cell center. Therefore, the momentum cells at the beginning and end of a particular straddle two different components. Since the current work for the system level analysis depends on defining component averaged forces, it would be ideal if friction forces were defined then over a mass/energy cell instead. Conveniently, RELAP5 computes a mass/energy cell volume averaged velocity, using velocities from the junctions. This velocity is used in the code to computing various correlations from friction factors to heat transfer coefficients. However, because the RELAP5 source code is not available, the current work can only use minor edit control variable calls to manipulate results as a post processing step and



is therefore limited to the minor edit information present. Wall friction coefficient information over the mass/energy cell volume is available, but interfacial friction coefficient and form loss coefficient information is only available at the junctions. The friction forces are thus computed two different ways, wall friction and virtual mass forces using mass/energy cell minor edit calls while the interfacial friction and form loss forces computed as contributions from the momentum cells “attached” to the mass/energy cell volumes.

The generic wall friction force in a cell volume is given by:

$$F_{wk} = -V (\alpha_k \rho_k v_k) FWK = -V v_k FWALK. \quad (3.22)$$

The wall friction coefficient,  $FWK$ , is related to the minor edit call  $FWALK$  by:

$$FWALK = \alpha_k \rho_k FWK. \quad (3.23)$$

The total wall friction force depends on whether the drift flux model is used for a particular flow regime. Therefore, defining a drift flux marker  $f_x$ , as being 1 if the drift flux model is used and 0 if it is not used, the total wall friction force is given as:

$$F_{wk} = -V v_k FWALK + f_x [\alpha_n V v_k FWALK - \alpha_k V v_n FWALN]. \quad (3.24)$$

In Eq. 3.24 the subscript  $n$  represents the phase other than phase  $k$ . This notation will be used for the interfacial friction terms as well.

The virtual mass force in RELAP5 is computed as:

$$F_{VM,k} = -VC \alpha_k \alpha_n \rho_m \frac{d}{dt} (v_k - v_n). \quad (3.25)$$

The value  $C$  is the virtual mass coefficient which in RELAP5 is independent of the flow regime and is only a function of the void fraction. The subscript  $m$  represents a mixture quantity.

The interfacial friction and form loss forces in a mass/energy cell volume are computed by summing up the contributions of the momentum cells in that particular mass/energy cell. Using the notation in Fig. 3-1, the mass/energy cell L has two momentum cells "acting" on it through junctions  $j$  and  $j + 1$ . The interfacial friction and form loss forces are therefore computed in the momentum cells associated with both junctions  $j$  and  $j + 1$ , then summed up based on the volume each of those momentum cells has in cell L.

The interfacial friction force from a junction  $j$  inside a particular cell  $i$  is given by:

$$(F_{visc,sk})_{ji} = \frac{1}{2} (\Delta x)_i A_j (\alpha_k \rho_k)_j (FIK)_j (v_n - v_k)_j. \quad (3.26)$$

Each junction in cell volume  $i$  has a length equal to half the cell length, that is why the volume occupied by junction  $j$  in cell  $i$  is defined as the junction area multiplied by  $\frac{1}{2} (\Delta x)_i$ . The phase fraction and phase density at the junction  $j$  are minor edit calls from RELAP5 determined using the interpolations of the values found at the two cell centers.

The form loss force in a particular momentum cell is defined as:

$$\begin{aligned} (F_{formk})_j &= (\alpha_k \rho_k v_k)_j (HLOSSK)_j A_j \\ &= (\alpha_k \rho_k v_k)_j \frac{1}{2} (FORMK)_k |v_k|_j A_j. \end{aligned} \quad (3.27)$$

The form loss force expression is therefore just a conventional expression for the form loss pressure drop multiplied by the junction area. The form loss coefficient,

*FORMK*, is the sum of the RELAP5 calculated abrupt area change value and the user defined form loss coefficient for that particular junction. The form loss force in cell *i* is then scaled by the length of that momentum cell in cell *i* with the total length of that particular momentum cell.

The total mass/energy cell force value is then the sum of the junction acting on that particular cell. Using the interfacial friction force as an example, the total cell volume friction in cell *i* is:

$$(F_{visc,sk})_i = \sum_j^J (F_{visc,sk})_{ji}. \quad (3.28)$$

The momentum transfer force due to mass exchange relies on a velocity of the phase at the interface. In RELAP5 that velocity is approximated by:

$$v_{sk} = v_I - v_k, \quad (3.29)$$

$$v_I = \lambda v_v + (1 - \lambda) v_l, \quad (3.30)$$

$$\lambda = \begin{cases} 0, & \Gamma > 0 \\ +1, & \Gamma < 0 \end{cases}. \quad (3.31)$$

The body force term captures not only the gravity force acting on a particular phase, but also the pump driving force. In a pump component, the standard body force term is replaced by the driving pump head, in Pascals, multiplied by the pump

component flow area. For all other components, the body force term  $B_z$  is given as:

$$B_z = g(\Delta z) A. \quad (3.32)$$

The system level phase momentum equation is set up just as the system level phase energy equation. Once the component group average values are computed and the component groups are summed up around the entire system, the system level phase momentum balance equation becomes (using the same notation discussed for the system level phase energy equations):

$$\begin{aligned} & \sum_n^{ALL} \left[ \frac{d}{dt} (\langle \alpha_k \rho_k v_k \rangle_n V_n) \right] + \Delta \left[ (A \alpha_k \rho_k v_k v_k)_{sys} \right] \\ &= \sum_n^{ALL} [F_{wk}] + \sum_n^{ALL} [F_{formk}] + \sum_n^{ALL} [F_{visc,sk}] + \sum_n^{ALL} [F_{VM,k}] \\ & - \Delta \left[ (PA \alpha_k)_{sys} \right] + \sum_n^{ALL} [\langle \Gamma v_{sk} \rangle_n V_n] + \sum_n^{ALL} [\langle (\alpha_k \rho_k) B_z \rangle_n]. \end{aligned} \quad (3.33)$$

Fractional contributions are similarly computed for the system level phase momentum equations.

### 3.3 Feed and Bleed Transient Model

#### 3.3.1 RELAP5 Model

The QPIRT methodology was applied to a TLOFW accident with subsequent feed and bleed. The MIT LB-LOCA PWR input deck was modified to simulate the desired transient. Figure 3-2 provides an illustration of the LB-LOCA input deck

nodalization. The model consists of two loops a single loop and a lumped triple loop to simulate a four loop nearly 3500 MWth PWR. The core is modeled with three channels, a core averaged channel, a hot assembly channel, and a hot rod channel. It is important to note that this input deck does not correspond to any plant in particular and uses only generic PWR geometry values. To handle the TLOFW accident, the LB-LOCA break was removed and an additional safety injection (SI) line was put in place on the singlet loop (or broken loop) side of Fig. 3-2. TRIP parameters were set to be consistent with an input deck given by INL for a feed and bleed transient.

### **RELAP5 Nodalization Diagram for Reference PWR**

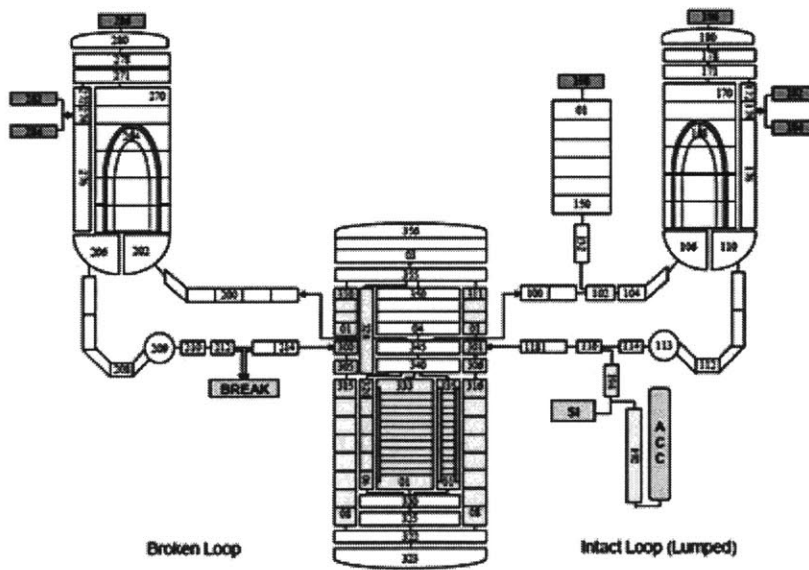


Figure 3-2: RELAP5 Nodalization Modified for the Feed and Bleed Sequence

The number of PORVs to open was selected to be consistent with the PORV flow area of standard four loop PWR plants and similar power levels. From a literature

search the flow area was chosen to simulate three PORVs being open [18], [19], [20].

### 3.3.2 RELAP Results

The transient was simulated by forcing shut off the main feedwater flow to the secondary side steam generators five seconds after the start of the simulation. Thirty seconds later the reactor was scrammed. PORV and SRV pressure set point were set to be consistent with the literature [18], [19], [20]. Once the primary system pressure reaches those set point values, the rapid opening and closing of the PORVs and SRVs maintains the system pressure at the set point value, while the system continues to heat up. If there is no operator initiated bleed through the PORV the pressure set points control the depressurization rate. Thus, the system pressure remains high and there is no way for the ECCS to inject into the primary system through the SI lines once the primary system begins generating vapor and uncovering the core. Operator action was modeled as a TRIP function that forced the PORV to remain open after a certain point in time. This time was varied until a limiting case was found. The limiting case is defined as being the case with the last possible time for operator initiated bleed to occur without the max PCT going above the NRC limit of about 1478 K (2200 °F). Figure 3-3 shows the PCT history results for five different cases each with the times for operator initiated bleed indicated on the figures. The horizontal line corresponds to the NRC limit. As the max PCT crosses the limit if the PORVs are forced open 40 minutes after loss of main feedwater flow, the limiting case is therefore defined as operator initiated bleed occurring at 39 minutes.

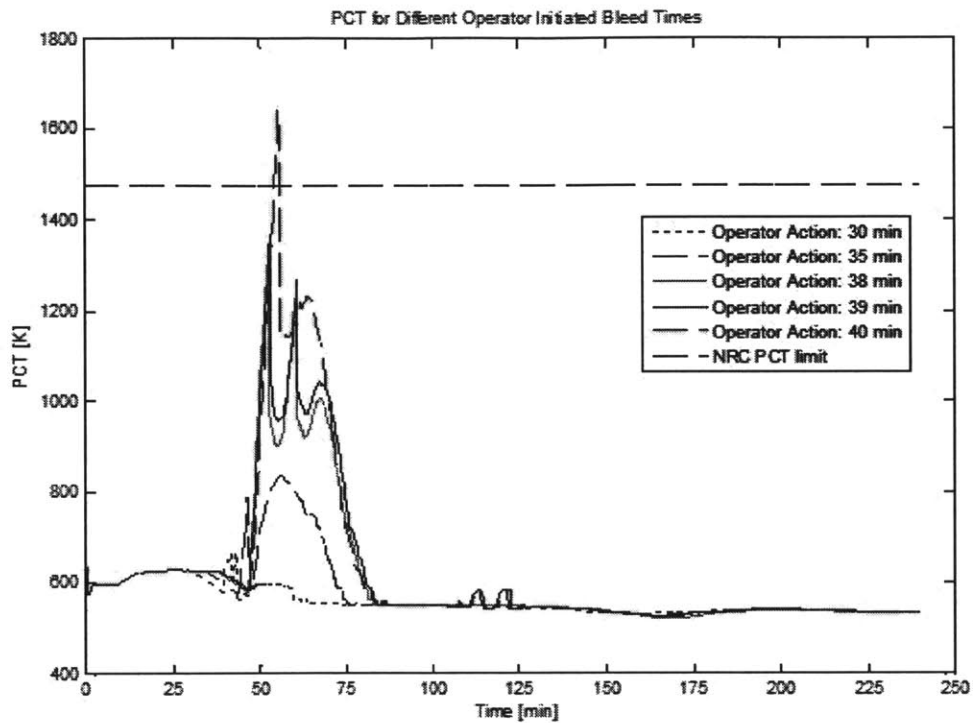


Figure 3-3: PCT for Multiple RELAP Runs with Different Operator Initiated Bleed Times

The QPIRT process was applied to this limiting case to determine the processes that significantly influence the PCT. Ultimately, the QPIRT process can also be used to identify specifically which physical phenomena keep the limiting case from crossing the NRC max PCT limit by comparing the 39 minute case to the 40 minute case.

### 3.4 QPIRT Analysis

From looking at the PCT results in Fig. 3-3, three distinct time phases are present. These three time phases are explicitly labeled in Fig. 3-4, which depicts the PCT results just for the limiting case of interest. Time Phase I lasts from the start of the transient to about 50 minutes. Between about 50 minutes and nearly 75 minutes is Time Phase II, and Time Phase III occurs from nearly 75 minutes to the end of the transient. QPIRT results will be given in each time phase. The Top-Down local level analysis includes the dominant processes over a particular time phase that significantly influence the clad temperature, fuel temperature liquid phase temperature and vapor phase temperature. The system level analysis gives the processes that significantly influence the system wide phase energy and phase momentum during a particular time phase.

A significant process was considered to be any process whose weighted contribution or fractional contribution was at least 10%. This value was chosen arbitrarily just for this work and a sensitivity analysis to this threshold value is discussed later on in Section 6.

For the limiting case of interest the PCT occurs in cell 11 of the hot rod channel. Thus, all local level results are from this cell location in the hot rod channel.



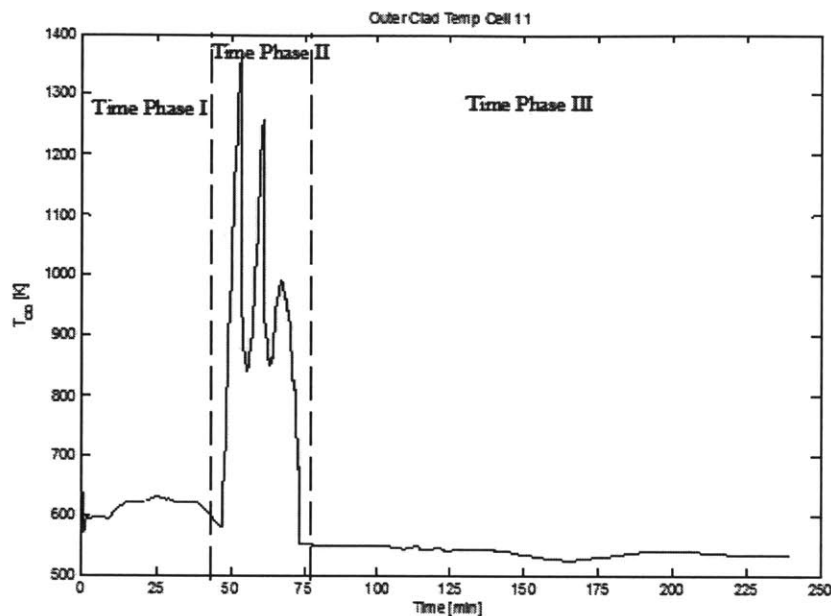


Figure 3-4: Limiting Case Clad Temperature in Cell 11 Where Max PCT Occurs

### 3.4.1 Local Level Top-Down Results

The Local Level Top-Down significant processes are given in Table 3.1 and the System Level Top-Down significant processes are given in Table 3.2. The results will be explained in detail in the sections that follow. The columns in Table 1 represent the significant processes that influence that particular constituent's temperature. The analysis starts with the clad temperature. For example, in Time Phase I, the only significant process that influences the clad temperature is the heat transfer to liquid. Thus, in Time Phase I the Fuel and Vapor constituent processes are ignored and the processes that influence the liquid phase temperature are examined.

For convective energy flow terms, "bottom" means the flow goes through the

bottom junction connected to the cell volume. “Top” therefore means the flow comes through the top junction and “cross-flow” means the flow comes through the junction connected to the adjacent hot assembly channel.

Table 3.1: Local Level Significant Processes

Time Phase	Clad Temp. (FOM)	Fuel Temp.	Liquid Temp.	Vapor Temp.
I	Clad-to-liquid heat transfer	Irrelevant	Convective energy flow, interfacial heat transfer	Irrelevant
II	Clad-to-vapor heat transfer, clad thermal capacitance, fuel-to-clad heat transfer, clad-to-liquid heat transfer	Internal heat generation, fuel thermal capacitance	Interfacial heat transfer, convective energy flow, liquid-phase thermal capacitance	Convective energy flow (both axial and cross)
III	Clad-to-liquid heat transfer	Irrelevant	Convective energy flow, interfacial heat transfer	Irrelevant

### Clad Temperature Results

The weighted contributions to the clad temperature are shown in Fig. 3-5. The Clad-to-Liquid heat transfer completely dominates the clad temperature value during Time Phases I and III. Figure 3-6 zooms in on Time Phase II to give a better view of the break down in this phase. Time Phase II corresponds to when that cell location in the hot rod channel has become uncovered. There are two interesting aspects of Time Phase II. The first is the significant influence of the Clad Thermal Capacitance on the clad temperature value. A Bottom-Up analysis of the clad material volumetric heat capacity shows a major spike in the value corresponding to a crystalline phase transition of the Zircaloy material. The second interesting point is

the role of the liquid phase on the clad temperature. It is difficult to see but at the times corresponding to the major temperature drops in Time Phase II at about 54 minutes and 62 minutes, the Clad-to-Liquid heat transfer rate becomes again significant. Thus, even though Time Phase II is characterized by lack of liquid locally, these two specific time instances have had enough liquid enter the fluid cell to allow the heat transfer rate to increase significantly.

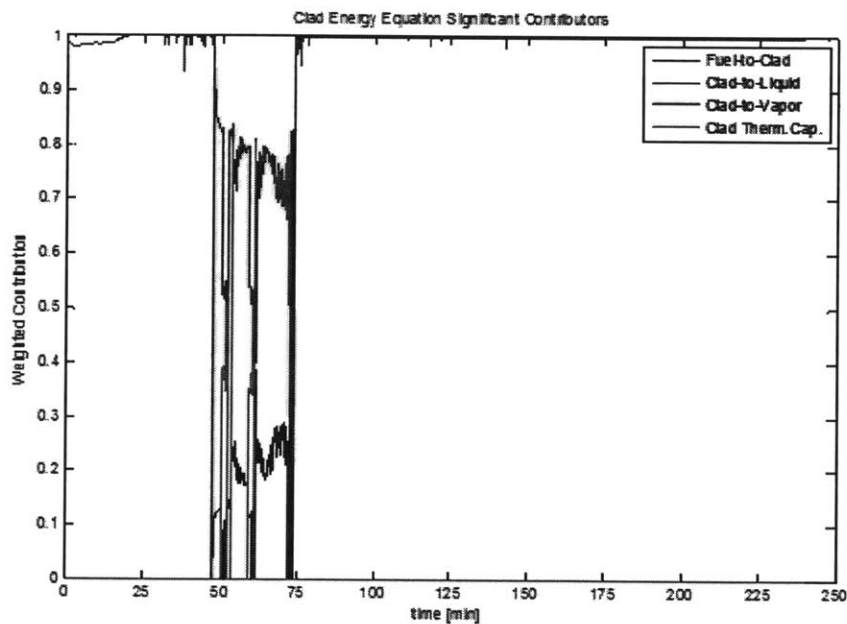


Figure 3-5: Clad Temp. Weighted Contributions

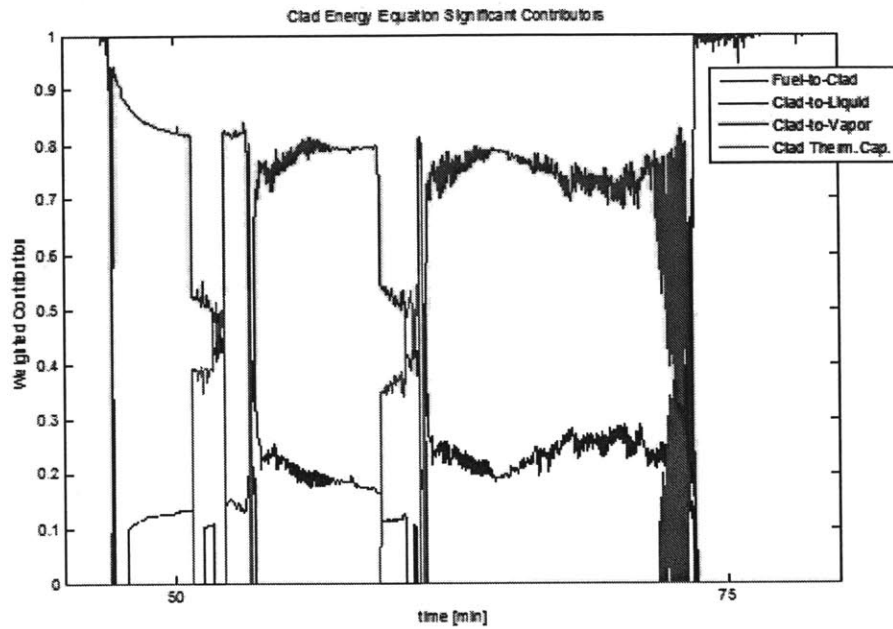


Figure 3-6: Clad Temp. Weighted Contributions During Phase II

### Fuel Temperature Results

The fuel temperature weighted contributions are shown in Fig. 3-7. The internal heat generation dominates Time Phases I and III while Time Phase II is characterized by a balance of the heat source with the fuel thermal capacitance.

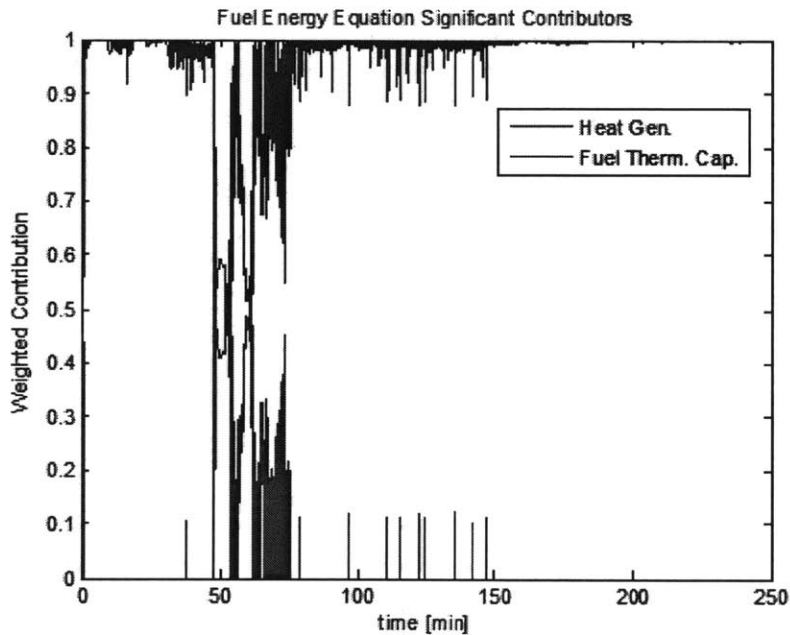


Figure 3-7: Fuel Temp. Weighted Contributions

### Liquid Temperature Results

The liquid temperature weighted contributions are plotted in Fig. 3-8 for the entire transient duration. Time Phases I and III are visible, but Fig. 3-9 zooms in on Time Phase II to give a more detailed view for this most important phase. The early part of the transient shows that the liquid temperature is dominated by the energy brought into the fluid cell via convection and the heat transfer rate with the clad. The contributions of these two processes stay roughly constant during the natural circulation phase of the transient, up until the liquid flow rate drops off and vapor begins to show up in the fluid volume, as indicated by the presence of interfacial heat transfer. Time Phase III interestingly enough shows that the clad-to-liquid

heat transfer rate is balanced by interfacial heat transfer and convective energy flow through the top junction. This represents that the local liquid flow field has not steadied out even though the clad temperature results are very smooth during Time Phase III. Eventually though, by about 150 minutes into the transient, the local liquid temperature weighted contributions steady-out as a balance between the heat transfer with the clad and interfacial heat transfer.

The liquid is important in Time Phase II only during those few key moments when the liquid presence causes a sudden spike in the heat transfer rate at the clad. So even though these two points in time are very short and sudden the processes that contribute to the liquid temperature will be important to the PCT. The interfacial heat transfer in the bulk fluid is always significant during this time period. At the two points in time that correspond to the sudden clad temperature drop, the liquid phase energy transient term becomes significant. Several minutes before these points in time, the convective energy flow through the bottom and cross-flow junctions also become significant. Interestingly, the heat transfer rate with the clad has an insignificant effect on the liquid temperature at both of these times. The clad therefore does not control the liquid phase temperature during Phase II even though at two very important times the liquid phase temperature impacts the clad temperature. This result is very different from Phases I and III where the clad and liquid phase temperatures are linked together.

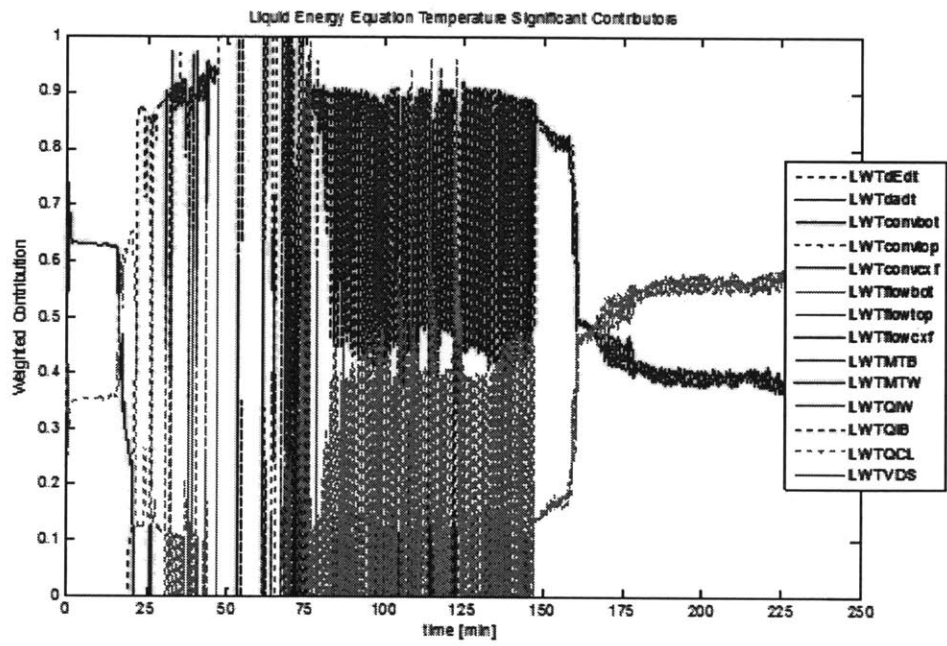


Figure 3-8: Liquid Temp. Weighted Contributions

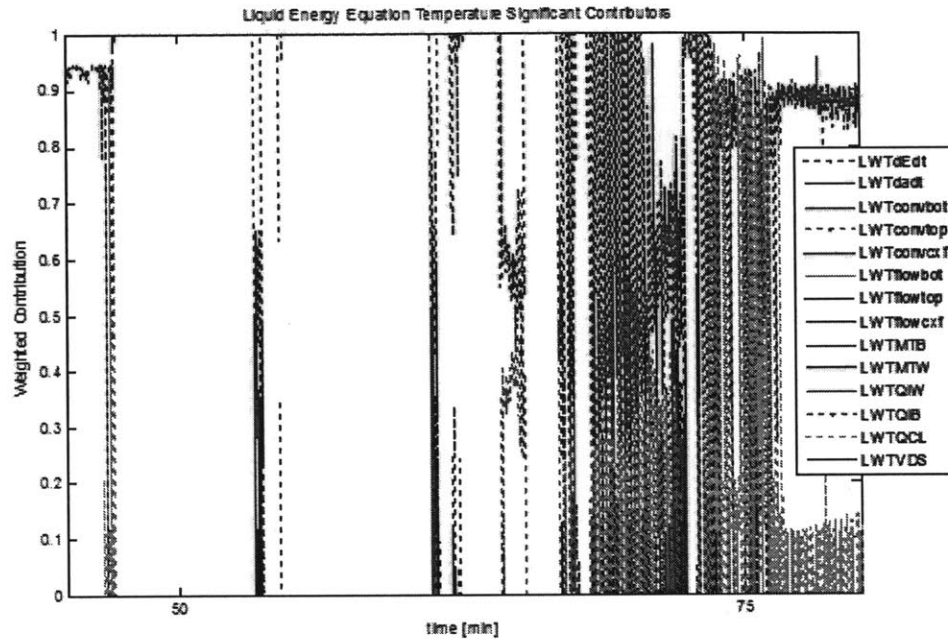


Figure 3-9: Liquid Temp. Weighted Contributions During Phase II

### Vapor Temperature Results

The weighted contributions onto the vapor phase temperature are shown in Fig. 3-10. Time Phases I and III show that the vapor temperature is dominated by the balance between interfacial heat transfer and convective energy flow from the bottom junction. The results for Time Phase II are zoomed in on in Fig. 3-11. The heat transfer rate with the clad is balanced by convective energy flow through the bottom junction for a majority of Phase II, but also with convective energy cross-flow as well. Cross-flow links the smaller hot rod channel with the larger hot assembly channel.

The convective energy flow terms represent energy brought into the local cell volume from another cell. This motivates the system level analysis by seeing what



processes and component groups significantly influence the system level flow dynamics.

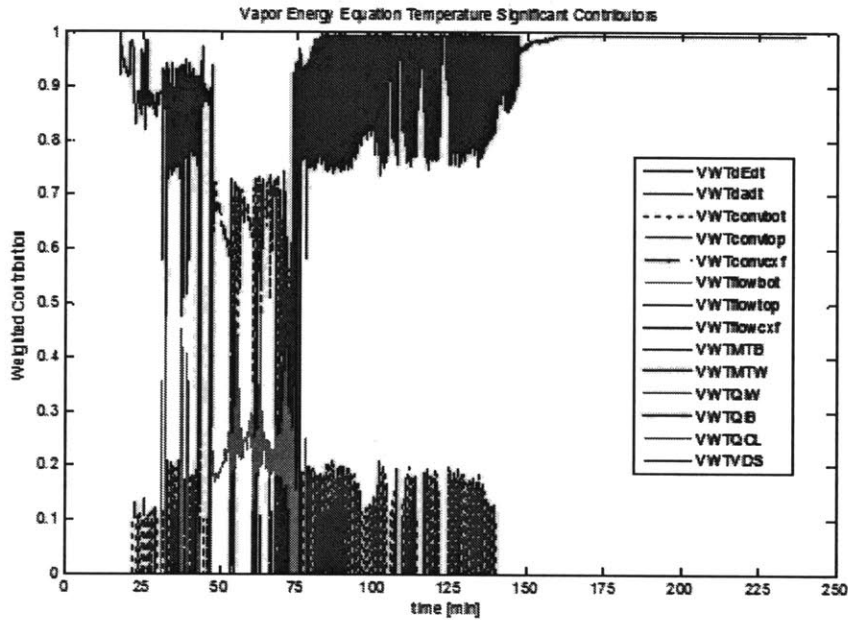


Figure 3-10: Vapor Temp. Weighted Contributions

### 3.4.2 System Level Top-Down Results

Due to the large amount of data generated from the system level equations, the results only go up to 100 minutes, or about half the length of local level results. This was mainly done to make handling the data easier while the QPIRT scripts used to manipulate the RELAP results were debugged.

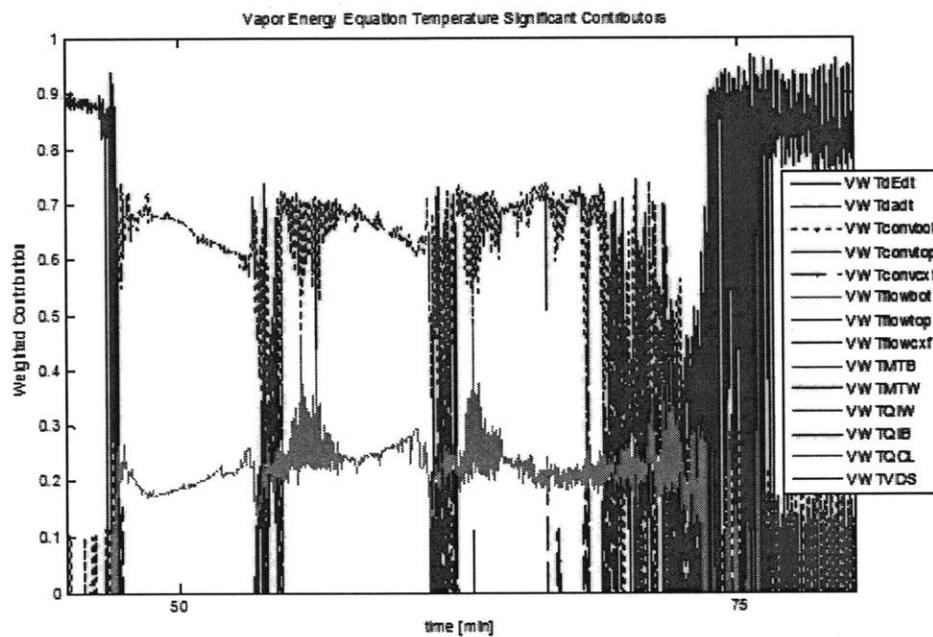


Figure 3-11: Vapor Temp. Weighted Contributions During Phase II

## System Level Phase Energy Equations

Tables 3.2 gives the dominant physical phenomena that govern the system level phase energy equations. Since the same physical process can occur in multiple component groups at once, the locations where the particular process occurs is listed in the adjacent column in Table 3.2.

The results from the system level energy equation relate directly to the known dynamics of the system throughout the transient. Early on in Time Phase I, there is an energy balance between the heat generated in the core and the heat transferred through the steam generators. However, once the secondary side of the steam generators dries out (since all feedwater was shutoff), the heat transfer through the primary side steam generators decreases dramatically, leading to an energy imbalance. This energy imbalance is captured in the system level liquid phase energy equation because the pressurizer energy transient term becomes significant once the heat transfer through the steam generators loses dominance. The pressurizer liquid energy transient term corresponds to the pressurizer filling up with liquid as the primary system heats up and the liquid expands. Once the pressurizer goes solid, liquid is ejected out the PORV. At this time more and more vapor begins to form in the primary system leading up to interfacial heat transfer and mass transfer terms becoming important. With the vapor phase, the primary dominant process is the convective energy outflow through the PORV. This makes sense and ties system level vapor energy equation to identifying when the PORV is opened to allow this to occur, which goes back to any uncertainty for when the operator can perform the action.

The system level energy equations results in essentially showing that the Bottom-Up step must be performed on the correlations in RELAP that model: wall-to-liquid heat transfer, interfacial heat transfer, and mass transfer. But it should be noted,

the location where each process occurs is important because different components may be in different flow regimes even if the same process is occurring, which results in different correlations to investigate.

Table 3.2: System Level Phase Energy Significant Processes

Time Phase	Liquid Energy		Vapor Energy	
	Process	Location	Process	Location
I	Wall-to-liquid heat transfer	Core and SGs	Interfacial heat transfer	Pressurizer
	Liquid-phase Thermal capacitance	Pressurizer, RPV (upper plenum, downcomer, upper head), core and core bypass	Vapor-phase thermal capacitance	Pressurizer, core, SGs
	Convective energy flow	PORV	Convective energy flow	PORV
II	Interfacial heat and mass transfer	Core	/	/
	Interfacial heat transfer	Core	Convective energy flow	PORV
	Liquid-phase thermal capacitance	Core, RPV (downcomer), SGs, pump, cold leg	Interfacial heat and mass transfer	Core, RPV (downcomer, upper plenum), pressurizer
	Wall-to-liquid heat transfer	Core	Wall-to-vapor heat transfer	Core
III	Interfacial heat transfer	Core	Convective energy flow	PORV
	Wall-to-liquid heat transfer	Core, SGs	Interfacial mass transfer	RPV (downcomer)
	Liquid-phase thermal capacitance	Core, pump, cold leg, RPV (upper plenum)	/	/

### System Level Phase Momentum Equations

The same process was carried out for the system level phase momentum equations. Table 3.3 gives the significant processes that influence the system level phase momentum equations and is setup the same way as Table 3.2.

At steady-state the pump provides the largest fractional contribution value, followed by the wall friction in the lumped loop steam-generator primary side tubes.

Table 3.3 does not give the fractional contribution values, but it is important to note here that these two processes alone do not provide even half of the total fractional contribution (a value of 1). This shows that there will be a potentially substantial influence on the number of processes considered significant by the choice of the importance threshold value. The results in both Tables 3.2 and 3.3 are determined using an importance threshold value of 10%. The next section will go into further detail analyzing the influence of varying that value.

Once the accident sequence begins, the main feature from the results, as shown in Table 3.3, is the balance between interfacial friction forces and buoyancy forces. This balance is the main influence on the liquid momentum equation, while several other processes contribute significantly to the vapor momentum equation. The pressure force from the PORV represents the PORV opening up and the system depressurizing, which again is consistent with the dynamics of the problem.

Table 3.3: System Level Phase Momentum Significant Processes

Time Phase	Liquid Momentum		Vapor Momentum	
	Process	Location	Process	Location
I	Body force (pump)	Pump	Pressure Force	PORV
	Wall friction	SG (lumped loop)	Body force (gravity)	Pressurizer, RPV (upper plenum, core bypass)
	Interfacial friction	Pressurizer, RPV (upper plenum), SG (lumped loop)	Interfacial friction	Pressurizer, SG (lumped loop), RPV (upper plenum)
II	Body force (gravity)	Core, RPV (upper head, lower plenum, downcomer), pressurizer	Phase Inertia	Pressurizer
	Interfacial friction	Core, pressurizer, cold leg (lumped loop), RPV (downcomer)	Interfacial friction	Core, pressurizer, cold leg (lumped loop)
	Body force (gravity)	SG (lumped loop), pressurizer, RPV (downcomer)	Body force (gravity)	Pressurizer
III	Interfacial friction	Core, pressurizer, RPV (downcomer)	Form loss	RPV (downcomer)
	/	/	Body force (gravity)	Pressurizer
	Body force (gravity)	Core, RPV (downcomer), SG (lumped loop)	Interfacial friction	Core, pressurizer, RPV (upper plenum, downcomer)

### 3.4.3 Importance Threshold Value Sensitivity

As stated previously, the threshold value for significance was rather arbitrarily chosen as 10%. Lowering the threshold value will allow more processes to be considered significant at a cost of adding more pieces to the Bottom-Up step. Ideally, the threshold value should give a minimal number of processes that still captures a major portion of the dynamics of the system response. The system level equations were reanalyzed using multiple threshold values ranging from 1% to the already chosen value of 10%. There are two key pieces of information here, the “fidelity” of the results and the total number of processes considered significant. The “fidelity” of the results refers to checking if the chosen significant fractional contributions sum

up close to one. Obviously the closer that value is to one, the more accurately the significant processes represent the total system dynamics.

Plots for the number of significant processes through time for the various threshold values used are given by Fig. 3-12. The four plots in Fig. 3-12 correspond to each one of the system level equations analyzed. The y-axis, in Fig. 3-12, is the number of significant processes and the colored lines in each sub-plot are the threshold values. Figure 3-13 then gives the corresponding "fidelity" of the chosen significant processes through time for the threshold values depicted in Fig. 3-12.

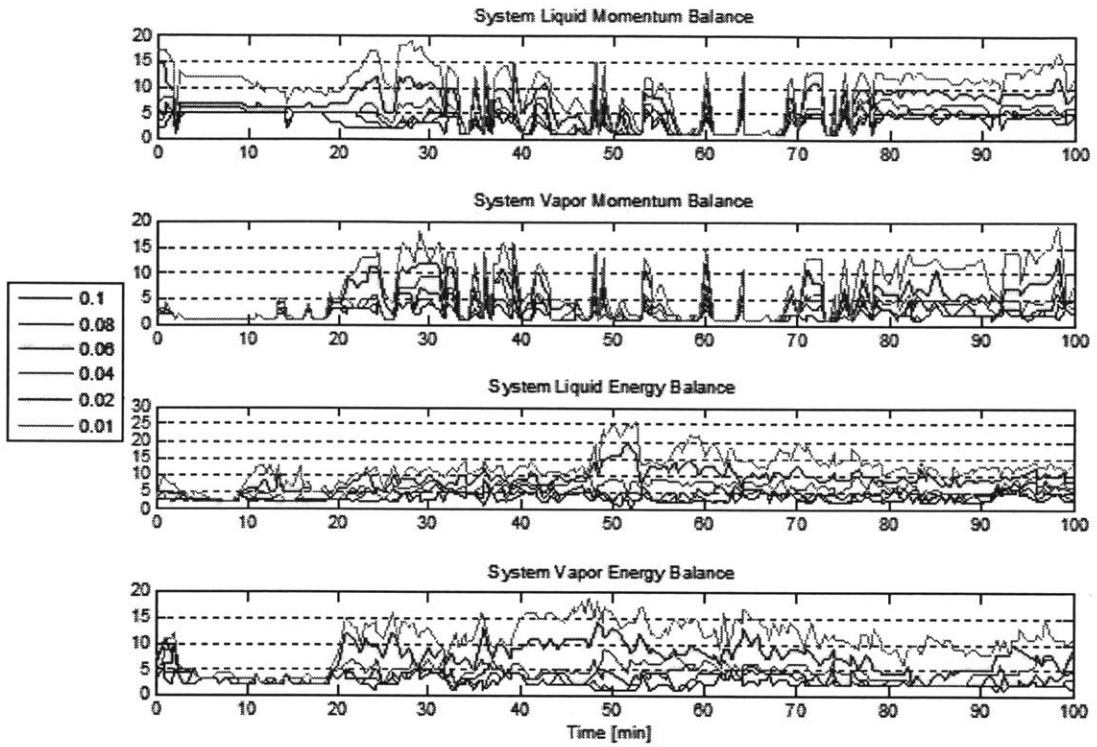


Figure 3-12: Sensitivity of the Number of Significant Processes (y-axis) to the Importance Threshold Value



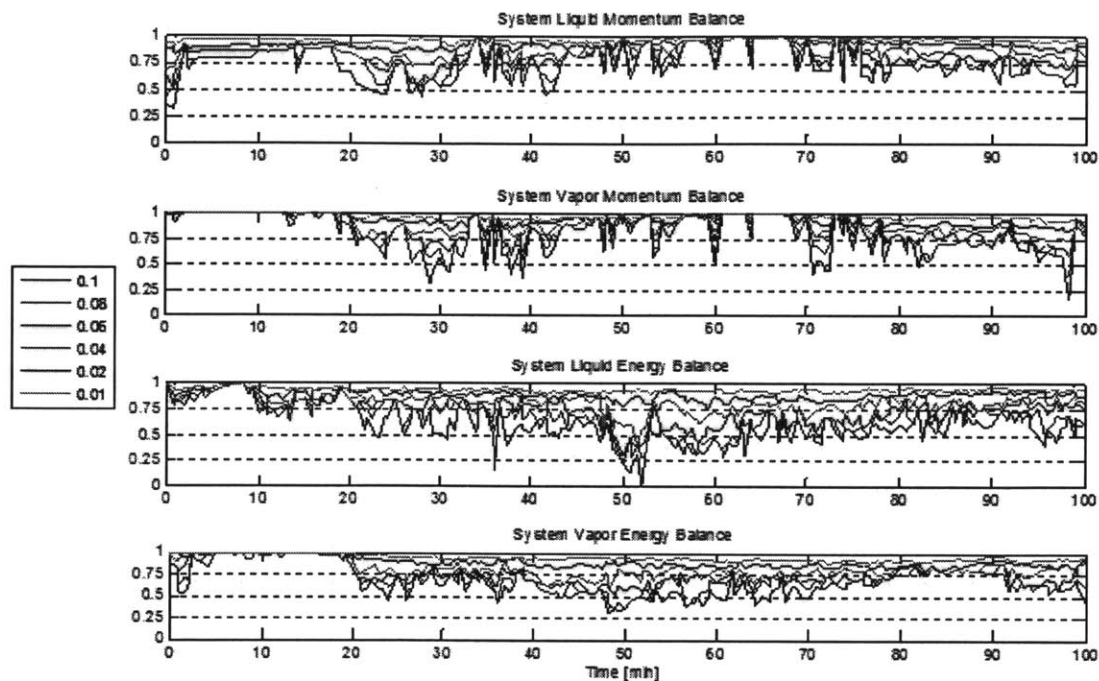


Figure 3-13: Sensitivity of the Fidelity of the Significant Processes to the Importance Threshold Value

As described in the previous section, the “fidelity” of the system level liquid momentum equation is very low at the steady-state condition (time equals zero). Reducing the threshold value raises the fidelity by capturing the influence of form loss and wall friction in the core. But to get the fidelity of the significant processes to be greater than 80%, the threshold value has to be as low as 2%, giving a total of about 15 processes to analyze in the Bottom-Up step just at the steady-state condition. The liquid level energy equation shows the opposite behavior at the steady-state condition, with the higher threshold value basically capturing the entire dynamics of the system. The difference in fidelity between the two equations comes

simply from the fact that the energy equation has only three really dominant terms, the heat transfer in the core and the two steam generators. The momentum equation on the other hand, balances the two pumps with the wall friction and form loss in multiple component groups.

As the accident progresses however, the opposite is true, as the fidelity of the higher threshold values in the liquid momentum equation improves while it worsens for the liquid energy equation. The vapor energy equation becomes more important through Time Phase II, as described in the local level results. Even though there are several points in time where the higher threshold values capture less than 20% of the liquid energy equation, the liquid phase is not as important during those times anyway. The fidelity of the higher threshold values are better for the vapor energy equation but are still only near 50% during the first half of Time Phase II. These results suggest that the threshold value of 10% might be too high to use to ensure that enough of the key significant processes are captured throughout the entire accident sequence.

#### **3.4.4 Bottom-Up Results**

The Bottom-Up step is more qualitative in nature, because it involves going through all of the physical phenomena identified from the Top-Down step. The various closure models and empirical correlations used in RELAP5 for the various physical phenomena are listed in Table 3.4 below. The uncertain parameters that a surrogate model would be built from would come from these models and correlations listed in Table 3.4.

Table 3.4: Bottom Up Step Important Closure Models and Empirical Correlations

<b><u>Physical Phenomena</u></b>	<b><u>Name of Correlations Used</u></b>
Wall-to-Coolant Heat Transfer	Dittus-Boelter Churchill-Chu Chen Chen-Sundaram-Ozkaynak Modified Bromley
Interfacial Heat Transfer – Bulk	Plesset-Zwick/Lee-Ryley + Nukiyama-Tamasawa Modified Unal-Lahey “Ad hoc” Taylor Bubble Criteria + Ishii-Mishima Modified Theofanos/modified Brown
Interfacial Heat Transfer – Near Wall	Saha-Zuber + Lahey + wall-to-coolant heat transfer correlations
Interfacial Mass Transfer – Bulk	Same as the interfacial heat transfer – bulk correlations
Interfacial Mass Transfer – Near wall	Same as the interfacial heat transfer – near wall correlations
Choked Flow	Ransom-Trapp
Wall Friction Factor	Zigrang-Sylvester approximation to Colebrook-White
Interfacial Friction	Zuber-Findlay Kataoka-Ishii Wallis Correlation/Ishii-Chawla EPRI
Material Properties	Clad Thermal Capacitance Fuel Thermal Capacitance Gap Conductance Model
Decay Heat Model	1973 ANS Standard
Pump	Farman-Anderson Pump Model



# Chapter 4

## Surrogate Models using Gaussian Processes

### 4.1 Surrogate Models Overview

Even with the uncertain parameters reduced to only the most important ones, as seen in Chapter 2, performing Bayesian inference still comes down to drawing thousands upon thousands of samples with MCMC. Applying Bayesian inference to relevant, large scale engineering problems, such as nuclear reactor safety analysis is therefore completely limited by the speed of drawing one sample. Since one sample means evaluating the likelihood function,  $p(\mathbf{y}|f(\mathbf{x}_{cv}, \theta))$ , the computer code,  $f(\mathbf{x}_{cv}, \theta)$ , must be evaluated thousands and thousands of times in series. Even if the computer code is considered fast and takes only 1 second to execute, Bayesian inference would be computationally intractable if  $1e5$  or more samples are needed.

We therefore need ways to quickly and accurately approximate the input/output relationship of the computer code. This approximation will then act as a computa-

tionally cheap surrogate model (also known as a meta-model) to the full long-running computer code in the Bayesian model calibration process. The long-running computer code is used to generate a set of training data which will, as the name implies, train the surrogate model so that it is able to match the input/output relationship, as closely as possible. The simplest of all surrogates is a response surface or best fit line in 1-D. Once the parameters in the best-fit line are estimated from the training data, the best-fit line can make a prediction of the output for any new input value. But, what if a linear relationship is not an accurate assumption for the input/output relationship? What if a quadratic, cubic or log-transformation is a better fit for the input/output relationship? This simple thought experiment illustrates a fundamental limitation of parametric models, where a parametric model is completely defined by the parameters in it (the slope and intercept of the best-fit line, for example). It would be better if the training data itself would dictate to the surrogate the trends, rather than a modeling choice before hand. Non-parametric models offer this kind of flexibility, and in many engineering disciplines, we already use non-parametric models quite frequently in the form of look-up tables. Flow regime maps, Critical Heat Flux (CHF) look-up tables, even material property data such as steam tables are non-parametric models. The one major drawback to a non-parametric model is that the training data is never discarded, so if a large number of training runs is required to accurately capture the input/output trends, all of that data must be retained and passed between computer functions appropriately.

Since we are focused on the comprehensive treatments of uncertainties in the Bayesian approach, we must account for the added layer of uncertainty in using the surrogate in place of the computer code. Therefore, we need a way to estimate how uncertain the surrogate is when making predictions at input values not used in the training set. In a linear interpolation look-up table, instead of simply interpolating

between two training points, the probability distribution needs to be known in order to propagate the surrogate's effect onto the output. This way, if the surrogate is poor, and is therefore uncertain about a prediction, that needs to be reflected in the output probability distribution. The surrogate is therefore a probabilistic model, which is often referred to as an *emulator* throughout the literature [13], [1], since it is trying to emulate the computer code behavior.

A very popular choice as an emulator in Bayesian non-parametric models is the Gaussian Process (GP). The underlying principles of the approach were developed in the 1960s in the geostatistics field where it was known as *Kriging* [15]. Since then Kriging has been widely used for optimization, but starting the late 1980s and early 1990s [21], [22], and [23] popularized the approach as Bayesian approximations to deterministic computer codes. The machine learning community has also extensively used GP models for both regression and classification (regression is used for continuous functions while classification is used for discrete datasets) [8], [24]. The most popular and (in my opinion) best resource on GP models is Rasmussen's Gaussian Processes for Machine Learning which includes a MATLAB package GPML to perform both GP regression and classification. The GPML package is a great asset as a starting point for using and applying GP models and most of the terminology and notation will be made consistent with that source. But even with all of their flexibility, GP models are still somewhat limited by certain assumptions which will be discussed later. Therefore in order to handle more complicated datasets (and thus more complicated input/output relationships) factor analysis techniques based on GP models have been developed [25], [26].

The rest of this chapter consists as follows. Section 4.2 summarizes the key fundamentals of standard GP regression, Section 4.3 extends GPs to the factor analysis techniques for more complicated problems, and Section 4.4 applies the various

emulator types to the simple friction factor demonstration problem.

## 4.2 Standard Gaussian Process Regression

### 4.2.1 Formulation

In the Bayesian framework a Gaussian Process (GP) prior is placed on the unknown computer code. The computer code, such as RELAP, is actually deterministic - meaning that the same output will result if the same input parameters and settings are used over and over. But, the output is in some sense unknown until the computer code is run, and will therefore be treated as a random variable. The GP prior is a distribution over the functional form of the input/output relationship. Formally, a GP is a collection of random variables, any finite number of which have a joint Gaussian distribution [24]. The relatively impressive name of Gaussian Process therefore really means a MVN distribution is used as a prior distribution. What makes the application special here, is that the covariance matrix of this MVN distribution will be computed using the training dataset. This feature will allow the GP model to try and interpolate (or rather regress) the training data thereby emulating the behavior of the long-running computer code.

I will use a slight break in notation from what I had earlier only to be consistent with Rasmussen's notation in GPML. The input  $\mathbf{x}$  will be all the inputs to the computer model that the GP is trying to emulate. The output  $f(\mathbf{x})$  is considered a random variable. A GP is completely specified by its mean function and covariance



function. The mean function  $m(\mathbf{x})$  and covariance function  $k(\mathbf{x}, \mathbf{x}')$  are defined as:

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))], \end{aligned} \quad (4.1)$$

and the GP is written as:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \quad (4.2)$$

An important aspect of Eq. 4.2 is that the covariance between the outputs is written as a function of the inputs. This is a key assumption in the simplicity of standard GP models and will be discussed in more detail later on. Following Rasmussen, as well as many other sources, the mean function is usually taken to be zero. Besides being the simplest approach, it also gives no prior bias to the trend of the data since no mean trend (such as linear or quadratic) is assumed. Covariance functions themselves depend on a set of hyperparameters, so even though the GP is a non-parametric model, these hyperparameters specify the covariance function and must be learned from the training data. The GP model is still considered a non-parametric model because the actual prediction requires regressing the training dataset. But because of the hyperparameters sometimes GP models are considered semi-parametric. Numerous covariance functions exist, ranging from the very simple to very complex neural net-like functions [24]. The most popular covariance function in the literature is the squared-exponential (SE) covariance function, which is usually parameterized in the following way:

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{x}_p - \mathbf{x}_q)^T M (\mathbf{x}_p - \mathbf{x}_q)\right), \quad (4.3)$$

where the subscripts  $p$  and  $q$  denote (potentially) two different values for the input vector  $\mathbf{x}$ . The hyperparameters in Eq. 4.3 are the signal variance  $\sigma_f^2$  and the matrix  $M$ , which is a symmetric matrix that is usually given as:

$$M = \text{diag}(l)^{-2}, \quad (4.4)$$

where  $l$  is a vector of positive values and each element,  $l_1, \dots, l_D$  plays the role of a characteristic length-scale for each input parameter. Loosely speaking, the length scale represents how far you need to move (along a particular axis) in input space for the function values to become uncorrelated [24]. Since each input parameter has its own unique length scale hyperparameter this formulation implements what is known as automatic relevance determination (ARD), since the inverse of the length-scale determines how relevant the input is: if the length-scale has a very large value, the covariance will become almost independent of that input. Linkletter et al. (2006) [27] used ARD to determine inactive/active input parameters using GP models.

Strictly speaking the GP model can interpolate the training data exactly if no noise is allowed between the training data points and the GP prior. However, I found allowing for noise makes building the emulator easier since it can prevent many ill-conditioning issues, which will be more readily present shortly. For that reason, the GP model is labeled a Gaussian Process Regression (GPR) model since it is trying to regress the training dataset within some allowable noise level. The GP prior is therefore placed on a latent (or hidden) function  $f(\mathbf{x})$ , that we wish to infer out from the noisy data,  $y$ . This viewpoint brings to light the signal processing nature of the GPR framework, since we wish to infer out the true signal pattern from noisy or “corrupted” data. In emulating computer codes, the training output is not noisy or “corrupt” but this setup provides a useful mathematical framework. The computer

model output of interest,  $y$ , is then related to the GP latent function,  $f(\mathbf{x})$ , as:

$$y = f(\mathbf{x}) + \epsilon, \quad (4.5)$$

where  $\epsilon$  is the assumed error structure. The error can take a wide variety of forms, but if a Gaussian likelihood is used with additive independent identitically distributed noise  $\epsilon$  with variance  $\sigma_n^2$ , the remaining calculations are analytically tractable. More complicated likelihoods and error structures can be used, but then the following calculations would not have analytical solutions (though those situations are very useful in other circumstances such as classification).

Before moving forward some important notation needs to be defined. If there are a total of  $N$  training points, the input parameters are stacked into an  $N \times D$  matrix of all of the training input values:

$$X = \begin{bmatrix} \mathbf{x}_1^T \\ \mathbf{x}_2^T \\ \vdots \\ \mathbf{x}_N^T \end{bmatrix}.$$

Each row of  $X$  contains the  $D$  input parameter values for that particular training case run. The training outputs values,  $y$ , are stacked into a vector of size  $N \times 1$  and written as  $\mathbf{y} = [y_1, y_2, \dots, y_N]^T$ . Since  $f(\mathbf{x})$  has a GP prior, and the likelihood function is Gaussian the prior on the training output is also Gaussian:

$$\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}(X, X) + \sigma_n^2 \mathbf{I}), \quad (4.6)$$

where  $\mathbf{K}(X, X)$  is the training set covariance matrix and  $\mathbf{I}$  is the identity matrix.

The training set covariance matrix requires applying the chosen covariance function to each pair of input parameter values:

$$\mathbf{K}(X, X) = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & k(\mathbf{x}_1, \mathbf{x}_2) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ k(\mathbf{x}_2, \mathbf{x}_1) & k(\mathbf{x}_2, \mathbf{x}_2) & \cdots & k(\mathbf{x}_2, \mathbf{x}_N) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & k(\mathbf{x}_N, \mathbf{x}_2) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}. \quad (4.7)$$

The training set covariance matrix is thus a full matrix. If the SE covariance function in Eq. 4.3 is used, each diagonal element of  $\mathbf{K}(X, X)$  is the signal variance  $\sigma_f^2$

Assuming for now that the covariance function hyperparameters are known, we want to use the training set to make predictions at new input parameter values, which Rasmussen refers to as test points. If there are  $N_*$  test points, the test input matrix is the  $N_* \times D$  matrix  $X_*$ . Under the GP model framework, the latent function at these new test points has the same GP prior as the training points, which will be written in vector form as

$$\mathbf{f}_* \sim \mathcal{N}(\mathbf{0}, \mathbf{K}(X_*, X_*)), \quad (4.8)$$

the test covariance matrix  $\mathbf{K}(X_*, X_*)$  has the same setup as the training set covariance matrix except now the elements in it are computed using the test input parameter values. As written, the test prior in Eq. 4.8 provides very little useful information, since it has no idea about the structure of the training dataset. In order to restrict the test function values to only those that agree with the training dataset, we must condition the test distribution on the training data distribution. To do this,

start with the joint prior, which is written out as:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N} \left( \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{K}(X, X) + \sigma_n^2 \mathbf{I} & \mathbf{K}(X, X_*) \\ \mathbf{K}(X_*, X) & \mathbf{K}(X_*, X_*) \end{bmatrix} \right), \quad (4.9)$$

where  $\mathbf{K}(X, X_*)$  is the cross-covariance matrix between the training and test input values of size  $N \times N_*$ , and  $\mathbf{K}(X_*, X)$  is its transpose. Standard multivariate normal theory allows us to write the conditional distribution  $p(\mathbf{f}_*|\mathbf{y})$  which gives the key predictive equations for GPR [24]:

$$\mathbf{f}_*|\mathbf{y} \sim \mathcal{N}(\bar{\mathbf{f}}_*, \text{cov}(\mathbf{f}_*)), \quad (4.10)$$

$$\bar{\mathbf{f}}_* \triangleq \mathbb{E}[\mathbf{f}_*|\mathbf{y}] = \mathbf{K}(X_*, X) [\mathbf{K}(X, X) + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{y}, \quad (4.11)$$

$$\text{cov}(\mathbf{f}_*) = \mathbf{K}(X_*, X_*) - \mathbf{K}(X_*, X) [\mathbf{K}(X, X) + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{K}(X, X_*). \quad (4.12)$$

In order to make a posterior prediction in the training data space (since  $f$  is the latent function), the posterior distribution is simply:

$$\mathbf{y}_* \sim \mathcal{N}(\bar{\mathbf{f}}_*, \text{cov}(\mathbf{f}_*) + \sigma_n^2 \mathbf{I}). \quad (4.13)$$

The posterior distribution of the test “targets”  $\mathbf{y}_*$  is simply the same as the latent function posterior distribution except with the additional error term added in.

Examining Eqs. 4.11 and 4.12 reveal the important features of GPR models. First, the posterior predictive covariance shrinks the prior covariance as witnessed by the subtraction between the first and second terms on the right hand side of

Eq. 4.12. Second, when making predictions at the training points ( $X_* = X$ ), the predictive uncertainty shrinks to the allowable error tolerance. The computational burden is completely determined by the computational cost of inverting the training set covariance matrix. If the training set is very large (greater than  $1e4$  data points for example) the cost of inverting the matrix might start to become prohibitive, and acceleration techniques might be needed. And lastly, if the training set covariance matrix is ill-conditioned, its inverse cannot be computed. That is why I allow some noise between the training data and the latent function, to make sure  $[\mathbf{K}(X, X) + \sigma_n^2 \mathbf{I}]$  is invertible. This surprisingly simple, and yet seemingly brute-force solution to ill-conditioning problems is widely used though it is usually only mentioned in footnotes of many references.

## 4.2.2 Building the Emulator

When writing the key GPR predictive equations, there was one important aspect left out: the fact that those predictions are also conditioned on the hyperparameter values. Using the SE covariance function and the likelihood noise structure above the complete set of hyperparameters is  $\phi = \{l, \sigma_f^2, \sigma_n^2\}$ . The posterior predictive distribution is then more formally written as  $\mathbf{f}_* | \mathbf{y}, \phi \sim \mathcal{N}(\bar{\mathbf{f}}_*, \text{cov}(\mathbf{f}_*))$ . The posterior predictive mean and covariance are the same however as given in Eqs. 4.11 and 4.12.

Training or building the emulator consists of learning those hyperparameters,  $\phi$ , out from the training dataset. The two main ways of building the emulator consists of the empirical Bayes approach and the fully Bayesian approach. The fully Bayesian method uses MCMC to draw samples from the posterior distribution of the hyperparameters while the empirical Bayes method uses optimization schemes to find point estimates to the hyperparameters. The main advantage of the empirical Bayes

method is that finding point estimates through optimization is much, much faster than using MCMC inference in the fully Bayesian approach. However, the drawback that I have found is that cross-validation is essential in order to determine if the “optimized” hyperparameters yield the “best” results. The optimization method must deal with very multi-modal functions that have several local maxima/minima which can cause the optimizer to get “stuck” at a poor choice location. The second drawback to the empirical Bayes approach is that by using point estimates it does neglect the influence of the hyperparameter uncertainty on the output. The entire model is still considered Bayesian because the GP itself is a statement of the probability of the latent function,  $f(\mathbf{x})$ . Most authors have found small differences between the final results of the empirical and fully Bayesian approaches [15], [13], but I have found the cross-validation is essential in more realistic problems. Because of this, I use the fully Bayesian approach, but will discuss both approaches anyway.

### Empirical Bayes

This is the approach used by Rasmussen in [24] as well as with the GPML MATLAB package, which includes a very useful optimization function. The main benefit of this approach, again is its speed. The hyperparameter values are optimized by maximizing the marginal likelihood, which recalling from Chapter 2 is the denominator in Bayes’ rule. Due to the change in notation from Chapter 2, the marginal likelihood is rewritten here as

$$p(\mathbf{y}|X) = \int p(\mathbf{y}|\mathbf{f}, X) p(\mathbf{f}|X) d\mathbf{f}. \quad (4.14)$$

The above probability statements explicitly condition on the training input matrix  $X$  just to show that we are working with the training set, not test inputs. The marginal likelihood again is integrating out the unknown variables of interest, which

in this setting is actually the latent function  $f(\mathbf{x})$  at the the training input points. The prior,  $p(\mathbf{f}|X)$  is just the GP prior specified above, and the likelihood as stated earlier is a Gaussian likelihood of the form

$$\mathbf{y}|\mathbf{f} \sim \mathcal{N}(\mathbf{f}, \sigma_n^2 \mathbf{I}). \quad (4.15)$$

Integrating out the latent function has actually already been done above in Eq. 4.6, but writing this out explicitly gives the function we wish to maximize during the optimization, the log-marginal likelihood:

$$\log [p(\mathbf{y}|X)] = -\frac{1}{2} \mathbf{y}^T [\mathbf{K}(X, X) + \sigma_n^2 \mathbf{I}]^{-1} \mathbf{y} - \frac{1}{2} \log [|\mathbf{K}(X, X) + \sigma_n^2 \mathbf{I}|] - \frac{N}{2} \log [2\pi]. \quad (4.16)$$

The three terms in Eq. 4.16 have easily interpretable roles. The first term on the right hand side is the data fit, the second term is the complexity penalty term depending only on the covariance function and the inputs, and the last term is a normalizing constant. The difficulty in performing the optimization results from the fact that Eq. 4.16 can be very multi-modal, especially as the number of input parameters increases. Cross-validation is important to determine which mode is the “best” to use, and sometimes the absolute global maximum does not always yield the best data fits [13].

## Full Bayesian

In the full Bayesian approach, a prior must be placed on the hyperparameters,  $p(\phi)$ , and the goal is to sample the posterior distribution conditioned on the training data.



Since the latent function is also unknown, the joint posterior is written as:

$$p(\mathbf{f}, \phi | \mathbf{y}, X) = \frac{p(\mathbf{y} | \mathbf{f}, \phi, X) p(\mathbf{f} | \phi, X) p(\phi)}{\int p(\mathbf{y} | \mathbf{f}, \phi, X) p(\mathbf{f} | \phi, X) p(\phi) d\mathbf{f} d\phi}. \quad (4.17)$$

But since the latent function variables can be integrated out, the posterior on the hyperparameters can be written, up to a normalizing constant for brevity as:

$$p(\phi | \mathbf{y}, X) \propto p(\mathbf{y} | \phi, X) p(\phi). \quad (4.18)$$

The “new” likelihood,  $p(\mathbf{y} | \phi, X)$ , is simply given by Eq. 4.6, where again the explicit conditioning on the hyperparameters and the training input matrix,  $X$  is to show it is for constructing the emulator. Drawing samples from the hyperparameter posterior can be done using the MCMC techniques discussed in Chapter 2. But, the hyperparameter prior, which will be referred to as the hyperprior, must still be specified. The easiest hyperprior to implement is the improper “flat” hyperprior  $p(\phi) \propto 1$ , which puts no bias *a priori* on the hyperparameter values.

Specifying useful hyperpriors for the GPR emulator is rather straightforward as described in multiple references [27], [28], [29], and [1]. The inputs are all scaled to be between 0 and 1 and the output is scaled to a standard normal. Thanks to this output scaling, we want to bias the signal variance  $\sigma_f^2$  to be close to 1. Thus making each diagonal element in the covariance matrix close to 1. To be consistent with the GPML toolbox notation, the hyperprior was setup on the  $\phi_f$  value instead using the change-of-variables transformation on the hyperprior. The hyperprior on the likelihood noise term is set to bias  $\sigma_n^2$  towards a small value since we want the GPR emulator to be very accurate relative to the training data. The simplest biasing hyperprior to use is a normal with a prior mean of  $10^{-6}$ . The length scale hyperpriors

are still relatively tricky to define but the specification from Higdon is used [27], [28], [29]. These length scale hyperpriors *a priori* bias the length scales to yield “smooth” input/output relationships and so only the training data can “push” the length scales to smaller values. Additionally, to make sure the covariance matrix is always invertible a small “nugget” value or “jitter” of  $10^{-6}$  was added to the diagonal of the covariance matrix. The nugget term is crucial in practical implementation of GP models but is rarely mentioned outside of foot notes [24]. The nugget adds a small amount of additional noise, preventing even a perfectly interpolating GPR model from interpolating the training set exactly. But this additional noise can prevent the covariance matrix from becoming ill-conditioned. There has been some detailed investigations into the nugget’s influence on the marginal likelihood [30] but for practical purposes the nugget is the easiest way to make sure the covariance matrix is always invertible.

### Toy Problem Demonstration

A simple 1-D example from [13] is used to illustrate the GPR emulator. This dataset was used because it is a relatively complicated shape and shows that the GPR emulator is trying to interpolate between the data points. The empirical Bayes GPR emulator is built using the GPML MATLAB toolbox and the full Bayes GPR emulator is built using the AM MCMC sampling scheme with  $2e4$  burn-in samples and  $2e4$  posterior samples. Nine training points are used to train the emulators and then 100 equally spaced test predictions, after the emulators have been built. Fig. 4-1 shows the posterior prediction results, where the grey shaded regions covers 95% of the probability of the empirical Bayes emulator prediction, and the blue lines are the 5%, 25%, 50%, 75%, and 95% predictive quantiles of the full Bayes emulator.

Both emulators pass through the training points with very small error and both exhibit similar behavior where away from the training points the uncertainty grows, as expected. Outside the training set bounds, both emulators predictive uncertainty increases sharply, which illustrates that standard GPR models are not very well suited for extrapolation. The empirical Bayes predictive mean is very similar to the full Bayes predictive median (which is equivalent to the predictive mean because the distribution is Gaussian).

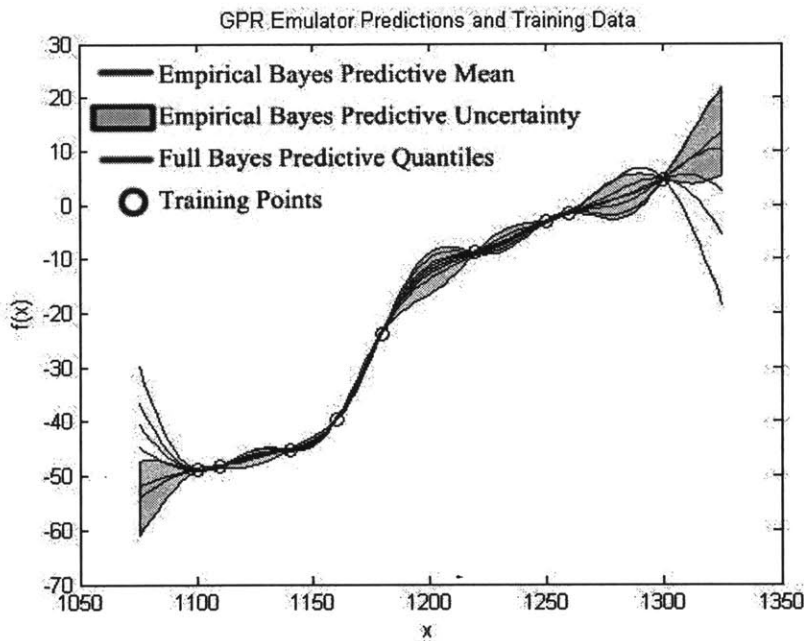


Figure 4-1: GPR Emulator Predictions

The differences in the two emulators is best seen though by looking at the covariance function hyperparameter values. Fig. 4-2 shows the sample histories of the length scale in the top plot and the signal variance in the bottom plot. In both plots, the red lines are the GPML toolbox optimized point estimates. Due to the signal

variance hyperprior, the posterior samples are centered around 1, but the GPML optimized value has no such restriction and is lower. Although it is difficult to see, the value of the GPML optimized length scale is sampled during the MCMC sampling of the full Bayes emulator. However, that particular value has a very low probability of occurring, which could be due to the hyperprior characteristics, or the optimization method got “stuck” in that particular local optimum. Either way, the final predictive results are similar between the two approaches, as expected, though the predictive uncertainty band is different, which is consistent with literature discussions on the topic [15]. The major difference is the construction time. The empirical Bayes approach takes about 0.2 seconds to construct, while the MCMC sampling takes 12 seconds total, or roughly  $3e-4$  seconds per MCMC iteration. In the context of emulating long-running computer codes, the additional training time due to MCMC sampling is almost negligible.

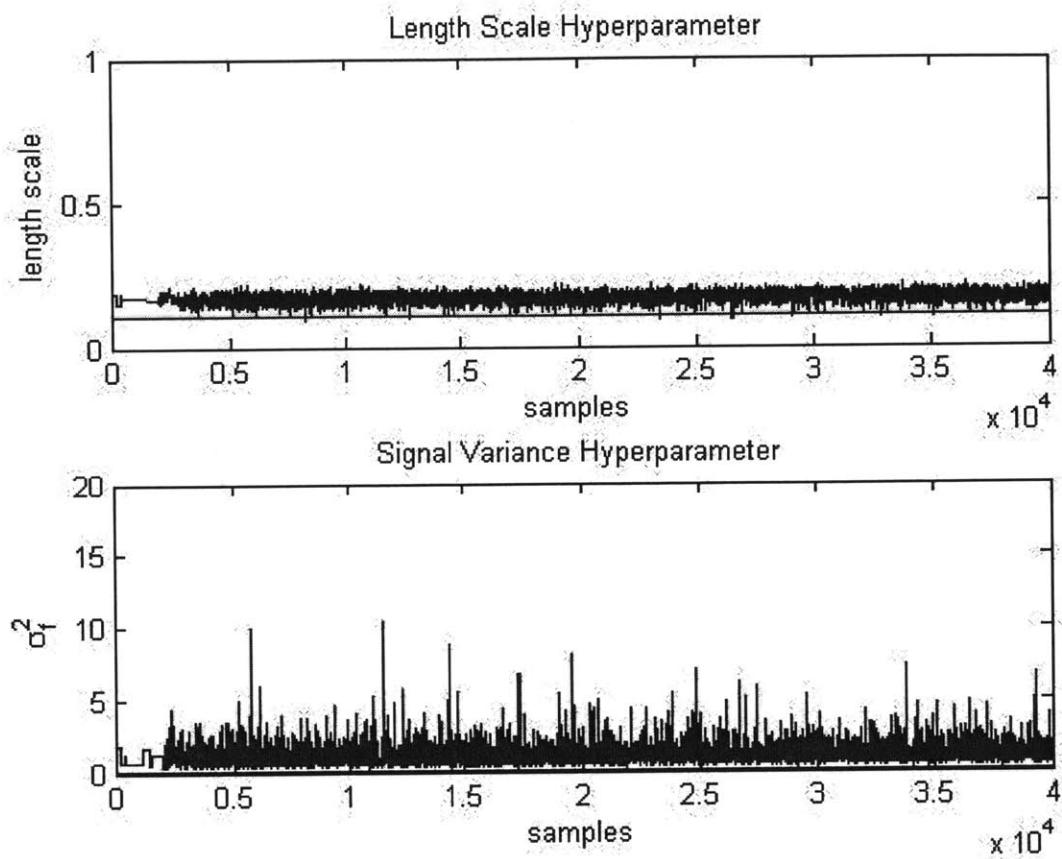


Figure 4-2: Covariance Function Hyperparameter Histories

### 4.2.3 GPR Emulator-Based Uncertain Parameter Calibration

Once the GPR emulator is constructed, it can be used to calibrate the uncertain parameters, in place of the simulator. In Chapter 2, uncertain parameter calibration with MCMC sampling was demonstrated by treating the simulator as a non-linear

mapping function between the uncertain parameters,  $\theta$ , and the simulator predictions. The likelihood function between the observational data and the uncertain parameters was then simply  $p(\mathbf{y}|f(\mathbf{x}_{cv}, \theta))$ . But, the likelihood function could be written in a “hierarchical-like” fashion with the “total” likelihood broken up into two parts. But first, the following discussion will use a slight change in notation to the notation used in Chapter 2. The observational data points will be denoted as,  $\mathbf{y}_o$ , the simulator predictions will now be denoted as,  $\mathbf{y}$ . This notational change hopefully helps prevent confusion between the latent variables and the simulator response. The first part of the “total” likelihood is then the likelihood between the observational data and the simulator predictions,  $p(\mathbf{y}_o|\mathbf{y})$ , while the second part is the likelihood between the simulator predictions and the uncertain parameters,  $p(\mathbf{y}|\mathbf{x}_{cv}, \theta)$ . The posterior distribution is now the joint-posterior distribution between the uncertain parameters and the simulator predictions conditioned on the observational data:

$$p(\mathbf{y}, \theta|\mathbf{y}_o) \propto p(\mathbf{y}_o|\mathbf{y})p(\mathbf{y}|\mathbf{x}_{cv}, \theta)p(\theta). \quad (4.19)$$

The likelihood between the observational data and the simulator predictions,  $p(\mathbf{y}_o|\mathbf{y})$ , is simply the assumed likelihood model for the experiment. As in the simple toy problem of Chapter 2, I use very simple Gaussian likelihood models with known measurement error values,  $\sigma_\epsilon^2$ . With  $N_O$  observational “locations” (data points), the vectors are setup as,  $\mathbf{y}_o = [y_{o,1}, y_{o,2}, \dots, y_{o,l}, \dots, y_{o,N_O}]^T$  and  $\mathbf{y} = [y_1, y_2, \dots, y_l, \dots, y_{N_O}]^T$ . Assuming a diagonal measurement error covariance matrix, the likelihood model can be written in a factorized form as:

$$p(\mathbf{y}_o|\mathbf{y}, \Sigma_\epsilon) = \prod_{l=1}^{N_O} p(y_{o,l}|y_l) = \prod_{l=1}^{N_O} \mathcal{N}(y_l, \sigma_{\epsilon,l}^2). \quad (4.20)$$

To complete the joint-posterior we need to know the likelihood between the simulator prediction and the inputs (where inputs denotes both the observational control variable locations  $\mathbf{x}_{cv,o}$  and the uncertain parameters  $\theta$ ). Since the simulator can be (very) non-linear and very complex it is essentially impossible to analytically write out this distribution. But thanks to the surrogate model, we can approximate this distribution using the GPR emulator posterior predictive distribution given in Eq. ???. Assuming the GPR emulator is already built from a given training set  $\mathcal{D} = \{\mathbf{y}, X\}$ , and the hyperparameters were determined using either the Empirical of Full Bayesian approach, the joint-posterior between the emulator estimated predictions  $\mathbf{y}_*$  and the uncertain parameters is:

$$p(\mathbf{y}_*, \theta | \mathbf{y}_o, \mathcal{D}, \phi) \propto p(\mathbf{y}_o | \mathbf{y}_*) p(\mathbf{y}_* | \{\mathbf{x}_{cv,o}, \theta\}, \mathcal{D}, \phi) p(\theta). \quad (4.21)$$

But because both the likelihood model in Eq. 4.20 is Gaussian and the emulator posterior predictive distribution is Gaussian, the emulator predictions can be integrated out of Eq. 4.21. The likelihood is now the GPR emulator modified likelihood, which is itself simply the emulator posterior predictive distribution with the measurement error added to the predictive variance:

$$\mathbf{y}_o | \{\mathbf{x}_{cv,o}, \theta\}, \mathcal{D}, \phi \sim \mathcal{N}(\bar{\mathbf{f}}_*, \text{cov}(\mathbf{f}_*) + \sigma_n^2 \mathbf{I} + \Sigma_\epsilon). \quad (4.22)$$

The (integrated) posterior distribution of  $\theta$  conditioned on the observational data is then

$$p(\theta | \mathbf{y}_o, \mathcal{D}, \phi) \propto p(\mathbf{y}_o | \{\mathbf{x}_{cv,o}, \theta\}, \mathcal{D}, \phi) p(\theta). \quad (4.23)$$

If more complicated likelihood models between the observational data and the simulator were assumed however Eqs. 4.22 and 4.23 would be different and potentially

analytically intractable.

### 4.3 Multi-Variate Output Emulators

As shown above, the GPR model provides a flexible framework for emulating non-linear functions from a specified number of training points. Their widespread use is due in many respects to their high degree of accuracy, and also due to their simplicity since all predictions are computed analytically, once the hyperparameters are known. However, their simplicity does impose certain important limitations, with the most important being that the covariance of the outputs is modeled as a function of the inputs. For many situations this assumption works fine, however in modeling multiple, or vector outputs, this means the covariance between the outputs themselves is neglected. Since we are ultimately trying to model a time series of temperatures, for example, neglecting the correlation between temperatures at different times means we cannot fully model the trajectory the temperature takes through the transient.

If the only input we were dealing with was time, then with a sufficient sized training set, standard GPR models would still provide very accurate approximations of the time series. However, we are concerned about a more complex situation where the output, the temperature, also depends on other inputs as well, namely a set of uncertain parameters that we ultimately want to calibrate against observational data. A particular output trajectory, or case run as I call it, gives the output at certain time locations for a specific set of uncertain parameters. Depending on the problem, a large number of case runs might be required for the GP model to sufficiently learn the output relationship to the various uncertain parameters. Applying standard GPR techniques to this training set requires taking only one point in time from each case run. For computationally expensive computer models this seems to be “throwing



away” a large amount of data when the training point is taken at the end of a long case run. In order to take multiple points in time from each case run, we need to be able to capture the correlation amongst the outputs themselves, in addition to the correlation between the inputs (which accounts for the correlation between case runs essentially). We would be maximizing the amount of information gained from a single case run, which would really improve the efficiency of the emulator approach if the computer model is very computationally expensive.

Adding this additional complexity is very challenging with two main views on how to do so: allow the additional covariance structure to occur in the GP covariance function directly or to perform some type of dimensionality reduction on the output. Allowing the covariance function to handle covariance between the outputs and inputs at the same time makes it very difficult to handle [31] and thus was not considered here. The other approach, of first applying a dimensionality reduction technique, simplifies the emulator construction by then allowing a series of standard GPR models to be summed together [29]. The main advantage of this approach is that once the dimensionality reduction has been performed, the standard GPR framework can be applied. Typically Principal Component Analysis (PCA) is used as the dimensionality reduction scheme [1]. The main disadvantage though is that the series of standard GPR models emulates that PCA-version of the output and thus must be transformed back to the original (high-dimensional) version of the output-space in order to compare to observational data (or equivalently transforming the observational data to the lower dimensional PCA-version of the output) and the uncertainty involved in that transformation needs to be accounted for. Additionally, the user must decide how many principle components to keep, and thus how many standard GPR models to build.

I have chosen an approach that uses the dimensionality reduction as part of the

emulator directly, instead of it being a transformation that must first be applied to the output. The method factorizes the high-dimensional output into the sum of products of lower-dimensional, “simpler”, functions. These “simpler” functions are given GP priors using a subset of the total input space [25]. Several names are used for this approach, and I primarily refer to Schmidt’s nomenclature in [25] and refer to it as Function Factorization with Gaussian Process Priors (FFGP) - I do not apply the warping function as he does and so do not refer to it as Function Factorization with Warped Gaussian Process Prior (FFWGP). The FFGP is a non-parametric factor analysis approach and is also referred to as Gaussian Process Factor Analysis (GPFA) models as in [26] and in several other statistical literature sources [32]. The appealing aspect of the factor analysis view is, each factor represents the functional relationship of the output to a particular subset of inputs, or to put it another way each factor captures the pattern or trend attributed to those particular inputs. Unlike the PCA transformation method, each factor has an easily interpretable role because the user predefined the inputs for that particular pattern. The PCA transformation method builds each of the standard GPR models using all of the input parameters, and so it becomes harder to interpret how each of the inputs relate to the output.

### 4.3.1 Function Factorization using Gaussian Process Priors

#### Model Formulation

The main idea of function factorization (FF) is to approximate a complicated function,  $y(\mathbf{x})$ , on a high dimensional space,  $\mathcal{X}$ , by the sum of products of a number of simpler functions,  $f_{i,k}(\mathbf{x}_i)$ , on lower dimensional subspaces,  $\mathcal{X}^i$ ,

$$y(\mathbf{x}) \approx \sum_{k=1}^K \prod_{i=1}^I f_{i,k}(\mathbf{x}_i). \quad (4.24)$$

In Eq. 4.24,  $I$  is the number of different factors and  $K$  is the number of different components within each factor. The terminology here is consistent with Schmidt's nomenclature, so the "components" is not the same as the "principal components" in PCA. The PCA "components" are somewhat like "factors" in the present usage. Due to the abstraction of Eq. 4.24, I will explain the model instead using just two factors,  $I = 2$ ,  $K = 1$ , and assume that the total input  $\mathbf{x}$ , is simply two input parameters:  $\mathbf{x} = \{x_1, x_2\}$ . Expanding the model to more factors is straight forward in theory, though book keeping is an important issue during implementation. The FF-model is written simply then as,

$$y(\mathbf{x}) \approx f_1(x_1) f_2(x_2). \quad (4.25)$$

A two-component version simply adds an additional product pair:

$$y(\mathbf{x}) \approx f_{1,1}(x_1) f_{2,1}(x_2) + f_{1,2}(x_1) f_{2,2}(x_2). \quad (4.26)$$

Comparing the above equations shows that if the output actually is the product of two separate functions of the two inputs, then Eq. 4.25 can reproduce the output exactly (within the allowable noise). But, if the output is the result of a more complex interaction of the inputs, then Eq. 4.25 is just an approximation, and allowing additional components as in Eq. 4.26 can help reduce the error between the output and the FF-model. Even though I have written Eqs. 4.25 and 4.26, where each factor acts upon a different one of the two inputs, in general any combination of the inputs can be used. Different factors can even act on the same set of inputs

as in [33] to study the effects of non-stationarity in the inputs.

The training dataset will consist of, as mentioned previously, a set of case runs where each case run has different set of values of the uncertain parameters. Choosing those particular values is very important but will not be discussed right now. For each case run, a specified number of control variable (which will usually be time) “locations” are chosen. The training output data,  $\mathbf{Y}$ , is then a  $M \times N$  matrix where  $M$  is the number of points taken per case run (number of control variable locations) and  $N$  is the number of case runs to make (the number of different uncertain parameter values). In general the training input data will be denoted as  $\mathbf{X}$ , which consists of the input parameter values for each of the case runs. Following the running example above of two factors, the input values consist of  $M$  separate locations of  $x_1$  and  $N$  separate case runs of  $x_2$ . And thus the training input is  $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2\}$  where  $\mathbf{x}_1$  is a  $M \times 1$  column vector and  $\mathbf{x}_2$  is an  $N \times 1$  column vector. The entire training dataset will be denoted as  $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$ . The FF-model, then becomes the matrix product of two vectors  $\mathbf{f}_1$  and  $\mathbf{f}_2$  for the one-component model:

$$\mathbf{Y} \approx \mathbf{f}_1 \mathbf{f}_2^T, \quad (4.27)$$

where T denotes the transpose of the vector. For the more complicated two-component model, the FF-model is the outer (tensor) product of matrices where each matrix stores that factor’s different components as column vectors:  $\mathbf{F}_1 = [\mathbf{f}_{1,1}^T, \mathbf{f}_{1,2}^T]^T$ ,  $\mathbf{F}_2 = [\mathbf{f}_{2,1}^T, \mathbf{f}_{2,2}^T]^T$  and the FF-model is:

$$\mathbf{Y} \approx \mathbf{F}_1 \mathbf{F}_2^T. \quad (4.28)$$

Each of the factorization functions then becomes a set of latent (hidden) variables

that must be learned from the training dataset. Performing Bayesian inference on the FF-model requires the specification of a likelihood function as well as priors on each of the factorizing functions. For likelihood models, I stick with the simple Gaussian likelihood between the FF-model output and the training data. But for the priors, this is where the GP framework is used, and GP priors are placed on each of the factorizing functions. I always use a specified zero mean function and the SE covariance function in Eq. 4.3, where each input for that factor has its own characteristic length scale. Denoting the covariance function as  $k(x, x')$  and the hyperparameters as  $\phi$ , the GP priors on each of the factors in the one-component example are:

$$f(x_1) \sim \mathcal{GP}(0, k(x_1, x'_1); \phi_1), \quad f(x_2) \sim \mathcal{GP}(0, k(x_2, x'_2); \phi_2). \quad (4.29)$$

Writing the GP priors in vector notation requires applying each of the covariance functions to their respective number of input pairs, giving the factor training covariance matrices  $\mathbf{K}_1(\mathbf{x}_1, \mathbf{x}_1)$  and  $\mathbf{K}_2(\mathbf{x}_2, \mathbf{x}_2)$ . The vectorized GP priors are:

$$\mathbf{f}_1 \sim \mathcal{GP}(0, \mathbf{K}_1(\mathbf{x}_1, \mathbf{x}_1); \phi_1) \quad \mathbf{f}_2 \sim \mathcal{GP}(0, \mathbf{K}_2(\mathbf{x}_2, \mathbf{x}_2); \phi_2). \quad (4.30)$$

With only 1-factor and 1-component the FFGP model reduces to the standard GPR emulator. However, there are important differences between the underlying assumptions of the two approaches. These differences are best highlighted when comparing 4.30 to the standard GPR emulator formulation described earlier. In standard GPR, the training covariance matrix consists of the covariance function applied to every input in the entire training set. For this scenario the total number of training points is  $NM$ , which means we will have to invert an  $NM \times NM$  to make

predictions. But in the FFGP setup, the training covariance matrices are applied only over their respective subsets of the total input space. Thus  $\mathbf{K}_1$  is size  $M \times M$  and  $\mathbf{K}_2$  is size  $N \times N$  and so the FFGP model could offer substantial computational savings if there are a large number of training inputs (very large  $NM$  value) since only two smaller matrices must be inverted to make predictions. For this reason the FFGP model is similar to certain sparse GP model approximations [24].

The other important difference between the FFGP model and standard GPR comes in how the training set is chosen. In standard GPR, ideally we want each training input point to offer as much new information as possible. If several training points all have the same value of  $x_1$  for example while  $x_2$  is varied, the standard GPR model does not learn anything about the relationship between  $x_1$  and  $y(\mathbf{x})$  from those points. That is why the concept of finding “space filling” designs for the training set are so very important in standard GPR models. Using the current nomenclature of case runs and control variable locations, the uncertain parameter inputs are fixed during a case run, and thus the model should only be able to learn about the control variables. The model learns about the uncertain parameters from the multitude of case runs that are used, but if the same control variable locations are used for each case run, the control variable pattern is not learned between case runs. However, in the FFGP or GPFA approach, learning a particular factor is maximized when the inputs for the other factors are held constant and thus all the variation is due to the change in that particular factor’s input. So if  $x_1$  corresponds to the control variable and  $x_2$  corresponds to the uncertain parameter, a particular case run maximizes the learning about the pattern of  $x_1$ . This concept is better illustrated by the manufactured solution example later on.

## FFGP Probability Model

The main price to pay for the added flexibility of the FFGP model is that training the emulator is far more computationally expensive than the standard GPR case. The primary reason for that is because the latent variables,  $\mathbf{f}_1$  and  $\mathbf{f}_2$ , cannot be integrated out analytically anymore. In the standard GPR model, with a Gaussian likelihood relating the latent variables to the target outputs, the GP prior allows integrating out the latent variables to yield a GP model for the outputs themselves. As shown in the Empirical Bayes section earlier, the GPR latent variables,  $\mathbf{f}$ , are integrated out and all predictions are compared to  $\mathbf{y}$ , the GPR training output variables. Now though, that integration is no longer analytically possible. Schmidt uses MCMC to sample the posterior distribution of the latent variables [25], though variational Bayes (VB) [26], and Expectation Propagation (EP) [33] methods can also be used to turn learning the latent variables into an optimization problem. For now, I have focused on using MCMC to sample the posterior latent variables. MCMC gives arbitrary accuracy in the limit of very long run times, and therefore in principle, can be used as a baseline to compare the VB and EP results if they are used in the future.

Parameterizing the likelihood function with the log-noise,  $\sigma_n^2 = \exp(2\phi_n)$ , the full joint posterior between the latent variables and all the hyperparameters (for the one-component model) is proportional to:

$$p(\mathbf{f}_1, \mathbf{f}_2, \phi_1, \phi_2, \phi_n | \mathcal{D}) \propto p(\mathbf{Y} | \mathbf{f}_1, \mathbf{f}_2, \phi_n) p(\mathbf{f}_1 | \phi_1) p(\mathbf{f}_2 | \phi_2) p(\phi_1) p(\phi_2) p(\phi_n). \quad (4.31)$$

As in all Bayesian inference, the joint posterior is simply proportional to the likelihood function times the prior, where the prior in Eq. 4.31 is factorized as the product of the GP priors for each factor and the priors on all of the hyperparameters - known as hyperpriors. The log-likelihood for the Gaussian likelihood function is

proportional to:

$$\log p(\mathbf{Y}|\mathbf{f}_1, \mathbf{f}_2, \phi_n) \propto -\frac{1}{2\sigma_n^2} \|\mathbf{Y} - \mathbf{f}_1\mathbf{f}_2^T\|_F^2 - \frac{NM}{2} \log \sigma_n^2, \quad (4.32)$$

where  $\|\cdot\|_F^2$  denotes the Frobenius norm. The log of the GP priors is just as in the case of standard GPR:

$$\log p(\mathbf{f}_i|\phi_i) \propto -\frac{1}{2} \log |\mathbf{K}_i| - \frac{1}{2} \mathbf{f}_i^T \mathbf{K}_i^{-1} \mathbf{f}_i, \quad (4.33)$$

where  $i = 1, 2$  for each of the two factors. Note that the prior formulation assumes that the two factors are independent. One of the benefits of performing inference with MCMC is that the no assumption in the posterior correlation structure is required. Thus, if the training data warrants it, the two factors can be correlated in the posterior.

### Implementation Issues

Drawing samples from the full joint posterior poses significant challenges. First, the latent variables within each factor are tightly correlated due to the GP priors. Second, the hyperparameters complicate matters further because they specify the covariance matrix for each GP prior, and are thus at a “higher” level from a hierarchical model point of view. Lastly, the likelihood function noise,  $\sigma_n^2$ , will dictate how well the FFGP model approximates the training data. When the noise level is large, the training output can be explained for many different values of the latent variables and covariance hyperparameters, which makes finding their posterior values difficult. If the noise level is small, then the FFGP model would prefer to want to explain the training output via the latent factors. Since we are using the FFGP model to



approximate a complex, long running computer simulator, which generated the training data, we want very little noise between the training output and the emulator. Even though this sounds like common sense it really helps the sampling of the latent variables to start out with a small  $\sigma_n^2$  value. An initial guess of  $10^{-6}$  for  $\sigma_n^2$  is used at the start of the FFGP emulator calibration.

The GPR emulator hyperprior specification was facilitated by scaling the training output to a standard normal. But a simple output scaling of the FFGP training output does not exist and so specifying the hyperpriors are also more difficult. Schmidt used improper “flat” hyperpriors for all of the hyperparameters and so I used the same approach. But the danger of using improper flat hyperpriors is that now the covariance function length scale hyperparameters might “jump” to values that lead to ill-conditioned covariance matrices even with a hard coded nugget value of  $10^{-6}$ . The simplest solution to this problem was to treat the nugget as a hyperparameter that must be learned from the training data as well. Each factor’s covariance function is then a modified SE covariance function:

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{x}_p - \mathbf{x}_q)^T M (\mathbf{x}_p - \mathbf{x}_q)\right) + \delta_{pq} \sigma_j^2, \quad (4.34)$$

where  $\delta_{pq}$  is 1 if  $p = q$  and 0 otherwise. Thus the nugget value,  $\sigma_j^2$  is added only to the diagonal of the covariance matrix, as stated already. The matrix  $M$  is specified the same way as for the standard GPR model where each input (for each factor) has its own length scale value. The length scales are also reparameterized so that  $l_{d,i} = \exp(\phi_{d,i})$ . Each factor was allowed to use a different nugget value and both were reparameterized in similar fashion to the likelihood noise reparameterization, thus  $\sigma_{j,i}^2 = \exp(2\phi_{j,i})$  where the subscripts  $j, i$  refers to the  $i$ -th factor’s nugget term. The length scale and signal variance hyperparameters for each factor are defined as

$\phi_i = \{[\phi_{1,i}, \dots, \phi_{D,i}], \phi_{f,i}\}$  where  $D, i$  is the number of inputs for factor  $i$ . And the two additional nugget hyperparameters are defined together as  $\phi_j = \{\phi_{j,1}, \phi_{j,2}\}$ . With improper “flat” hyperpriors,  $p(\phi_1) \propto 1$ ,  $p(\phi_2) \propto 1$ ,  $p(\phi_j) \propto 1$  and  $p(\phi_n) \propto 1$ , the joint-posterior is simply given as:

$$p(\mathbf{f}_1, \mathbf{f}_2, \phi_1, \phi_2, \phi_j, \phi_n | \mathcal{D}) \propto p(\mathbf{Y} | \mathbf{f}_1, \mathbf{f}_2, \phi_n) p(\mathbf{f}_1 | \phi_1, \phi_{j,1}) p(\mathbf{f}_2 | \phi_2, \phi_{j,2}). \quad (4.35)$$

As stated in the GPR model setup, these hyperparameter reparameterizations put all hyperparameters on the same order of magnitude, which I found makes specifying MCMC proposal distributions much easier. If the sampling needed to make a proposal jump for  $\sigma_{j,i}^2$  and it was a really tiny value then the same proposal jump of a length scale  $l_{d,i}$  that was near a value of 5 would be basically meaningless. But thanks to this reparameterization proposal jumps are easier to specify for all hyperparameters simultaneously which speeds up the MCMC sampling.

With that said a check is still required to make sure the covariance matrices are invertible. Following the GPML toolbox, I do not compute the covariance matrix directly, but first compute the cholesky decomposition which is essentially the “matrix square root” of the covariance matrix. For covariance matrix  $\mathbf{K}$  the cholesky decomposition is a lower triangular matrix defined as  $\mathbf{L}\mathbf{L}^T = \mathbf{K}$  (alternatively an upper triangular matrix can be used for the cholesky decomposition but I use the lower triangular format). The covariance matrix inverse is then computed using the cholesky decomposition as

$$\mathbf{K}^{-1} = \mathbf{L}^T \setminus (\mathbf{L} \setminus \mathbf{I}), \quad (4.36)$$

where the “backslash” notation  $\mathbf{A} \setminus \mathbf{b}$  is the vector  $\mathbf{x}$  that solves  $\mathbf{A}\mathbf{x} = \mathbf{b}$ . In MATLAB the cholesky decomposition is computed simply with the function call:

$L = \text{chol}(K, 'lower')$ . If  $K$  is ill-conditioned MATLAB will generate an error which would kill the emulator calibration scheme. But if the cholesky decomposition is computed using the function call:  $[L, p] = \text{chol}(K, 'lower')$  in MATLAB even if  $K$  is ill-conditioned MATLAB will not generate an error. If  $K$  is positive definite (and not thus not ill-conditioned)  $p=0$  and  $L$  is the cholesky decomposition defined as  $LL^T = K$ . If  $K$  however is not positive definite (and thus ill-conditioned)  $p > 0$  such that the returned  $L$  matrix is the cholesky decomposition of a smaller subset of  $K$  defined as  $LL^T = K(1 : (p - 1), 1 : (p - 1))$ . Thus during the MCMC sampling I use a check where if the MATLAB computed  $p$ -value for either factor covariance matrix is greater than zero those hyperparameter values are immediately rejected.

Another complication inherent to working with factor analysis models in general is the problem of “unidentifiability” [8]. Unidentifiability is best explained through an example. Consider the 2-factor 1-component model in Eq. 4.30; if the output  $y(\mathbf{x})$  is positive, the latent factors can be either both positive or both negative. Their signs do not actually matter as long as their product has the correct sign. Thus, there is not a truly unique solution. Although unidentifiability does not impact the predictive quality of the factor analysis model [8] many “tricks” exist to try and work around it. I have chosen to not to try and specifically deal with this issue primarily because I am using the FFGP model as a surrogate to the computer code, rather than trying to use the FFGP to interpret something about the structure of the output.

## Calibrating the Emulator with MCMC

To draw samples from the posterior, I break up the sampling into two steps following Ch. 5 in [9] which samples the latent variables using Hamiltonian Monte Carlo (HMC) with the hyperparameters fixed, then samples the hyperparameters with an-

other scheme with the latent variables fixed. HMC makes use of gradient information to suppress the randomness of the random-walk sampling behavior [34], [9] and is very useful for sampling a large number of highly correlated variables, which is just what the latent variables are. For the hyperparameters I use basic RWM sampling, which although is the most basic MCMC algorithm, seems to work well in the example problems I have worked with. All of the hyperparameters use the same proposal jumping variance thanks to the reparameterizations described above. If the reparameterizations were not used, different jumping variances would be required for each hyperparameter type, which makes specifying the RWM scheme more challenging. The RWM step is the slow step since the factor covariance matrices must be inverted (using the cholesky decomposition as shown above). The HMC step is faster since the covariance matrices are fixed and all computations are just matrix computations in MATLAB.

This approach is “Gibbs-like” in that each step is only sampling part of the total variable-space while the other part is fixed. In true Gibbs sampling the full-conditional distribution can be written out analytically and so all proposals are accepted [8]. Here though, the full-conditional distribution cannot be written out analytically which is why an accept/reject step must still be used for each proposal. Both steps evaluate the same expressions though because I assume improper “flat” hyperpriors. If proper hyperpriors were used they would only be evaluated when the hyperparameters are sampled. Following this, the “full-conditional-like” joint posterior distribution evaluated when the latent variables are sampled is:

$$p(\mathbf{f}_1, \mathbf{f}_2 | \mathcal{D}, \phi_1, \phi_2, \phi_j, \phi_n) \propto p(\mathbf{Y} | \mathbf{f}_1, \mathbf{f}_2, \phi_n) p(\mathbf{f}_1 | \phi_1, \phi_{j,1}) p(\mathbf{f}_2 | \phi_2, \phi_{j,2}). \quad (4.37)$$

The “full-conditional-like” joint posterior distribution evaluated when the hyperpa-

parameters are sampled is:

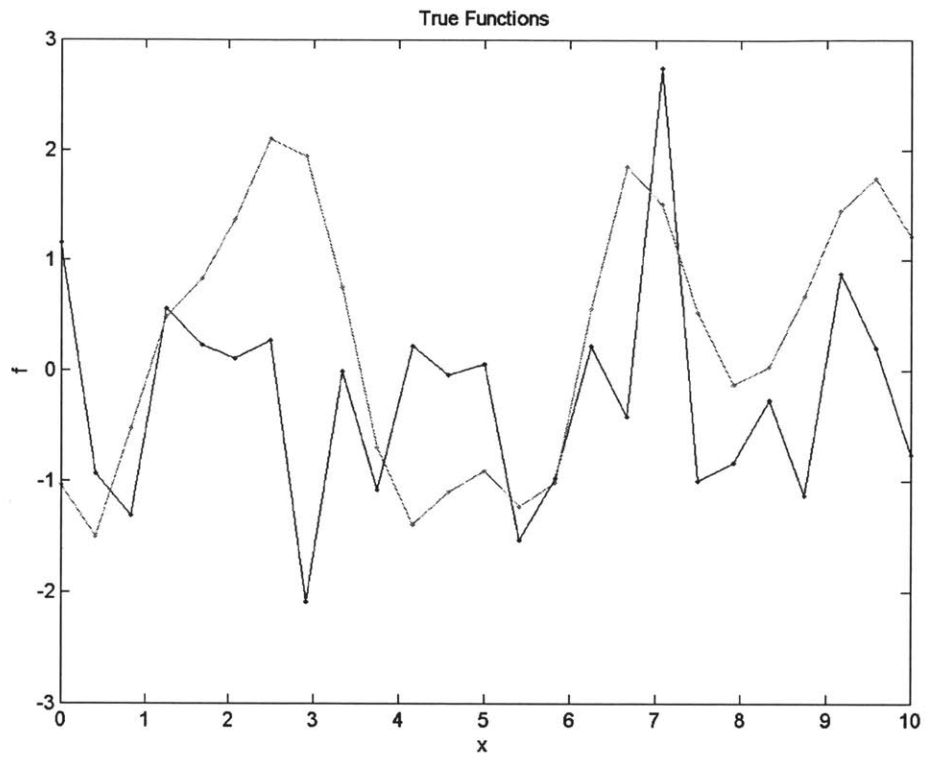
$$p(\phi_1, \phi_2, \phi_j, \phi_n | \mathcal{D}, \mathbf{f}_1, \mathbf{f}_2) \propto p(\mathbf{Y} | \mathbf{f}_1, \mathbf{f}_2, \phi_n) p(\mathbf{f}_1 | \phi_1, \phi_{j,1}) p(\mathbf{f}_2 | \phi_2, \phi_{j,2}). \quad (4.38)$$

I found from experience that during the MCMC sampling when the latent variables are being sampled at very high mixing rates the hyperparameter mixing rate is poor and vice versa. This does make some intuitive sense in that while the hyperparameters are “moving around” a single latent variable proposal might have a different covariance matrix compared to the previous sample, which changes the log-GP prior along with the change in latent variable values. Since a proposal is randomly accepted if it does not increase the log-joint-posterior, a poor guess in the hyperparameters would compound a poor guess in the latent variables or even suppress a decent guess in the latent variables leading to the proposal being rejected. But once “good” hyperparameter values are found, the sampling becomes dominated by the latent variable sampling due to the data-fit between the training output and the FF-model likelihood. The latent variables then have very high mixing rates while the hyperparameters are “stuck” these “good” values. Thus, the hyperparameter sampling essentially results in point estimates. Continuing to compute the factor covariance matrix inverses is essentially a waste and the results are almost an Empirical Bayes type of approach where the latent variable samples depend on the hyperparameter point estimates. For that reason I broke the FFGP training into two phases: the latent burnin phase that samples both hyperparameters and latent variables and the latent posterior phase where only the latent variables are sampled. Before the start of the latent posterior phase, the hyperparameters are fixed a specified value which I usually set as their mean values over the last half of the latent burnin phase (which allows for a hyperparameter burnin during the latent burnin phase). Additionally I

set the latent posterior phase initial values to the empirically estimated latent variable means computed from the last half of the latent burnin samples. I refer to this setup as the Empirical Bayes FFGP model, which has a very computationally intensive training process since the hyperparameter point estimates are themselves found from MCMC sampling.

### Toy Problem

To verify the FFGP model construction was working properly a method of manufactured solution approach was used. Two covariance functions were specified by specifying hyperparameter values. These covariance functions were used to create the covariance matrix for two GPs, a random vector was drawn from each GP. These vectors act as the latent functions in the FFGP model,  $\mathbf{f}_1$  and  $\mathbf{f}_2$ , and were then vector multiplied together to create the training matrix  $\mathbf{Y} = \mathbf{f}_1\mathbf{f}_2^T$ . With 25 training input points for each function, there are a total of 625 training data points. The true functions are shown in Fig. 4-3 below and are very non-linear. The FFGP model is constructed by trying to infer out the true latent function values at each of the training input locations. The MCMC sampling used  $5e4$  latent burn-in samples and  $10e4$  latent posterior samples, using the MCMC procedure described above. The training results are shown in Fig. 4-4 where the red are the true function values and the blue lines are the posterior quantiles of the latent factors at each of the input locations. This perfectly demonstrates the unidentifiability issue, with the latent factors mirroring the true function values. The posterior quantiles are very tight, representing that there is very little uncertainty in the posterior latent function distributions, which makes sense since by definition the training output is the vector product of two functions.



Functions.jpg

Figure 4-3: True Functions in the Toy Problem

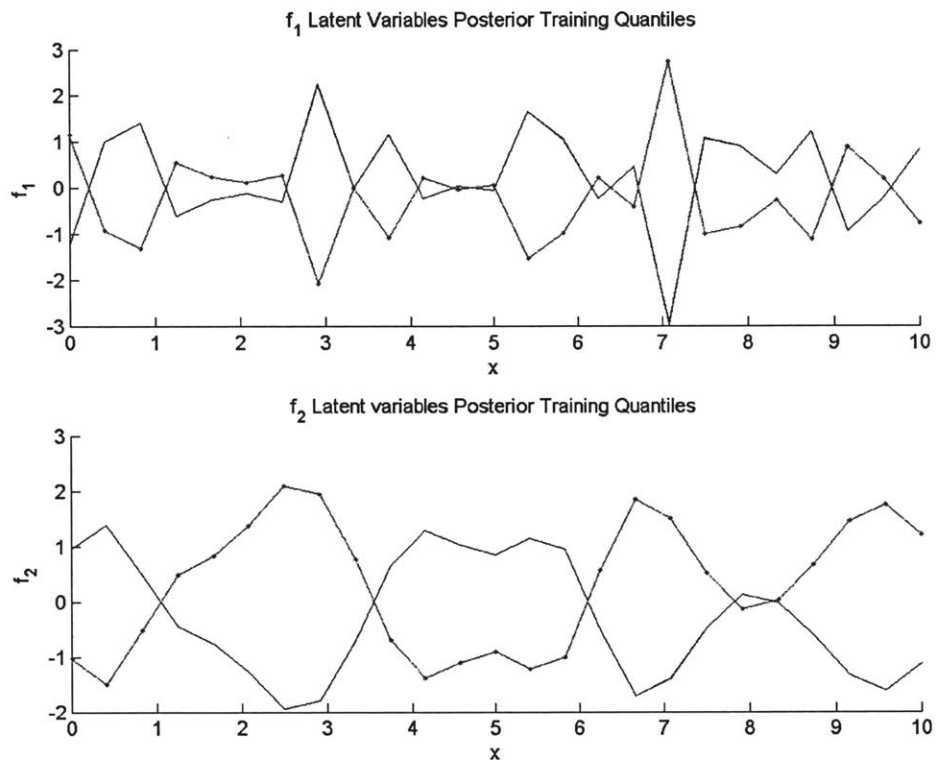


Figure 4-4: Posterior Latent Training Variable Quantiles



## Making Posterior Predictions

The FFGP posterior predictive distribution is far more complex than the GPR posterior predictive distribution. In order to make a prediction with the FF-model itself, the predictions in the latent space must first be known. Thanks to the GP prior placed on each of the factors (and potentially each of the components within each factor), predictions within the latent space follow the standard GPR theory [25]. All component latent variables within a particular factor will be stacked together into a single stacked-vector notation:  $\hat{\mathbf{f}}_i = [\mathbf{f}_{i,1}^T, \dots, \mathbf{f}_{i,K}^T]^T$ . Each of the factor stacked-vectors are then stacked again into a “super” stacked vector consisting of all latent variables in the FF-model:  $\hat{\mathbf{f}} = [\hat{\mathbf{f}}_1^T, \dots, \hat{\mathbf{f}}_I^T]^T$ . Since I have exclusively used only two factors in this work, relating the factor stacked-vectors to the factor matrices in Eq. 4.28 requires stacking the columns within each matrix on top of each other. A typical notation for that operation is  $\hat{\mathbf{f}}_1 = \text{vec}(\mathbf{F}_1)$  and  $\hat{\mathbf{f}}_2 = \text{vec}(\mathbf{F}_2)$ . All of the following assumes I am using the Empirical Bayes FFGP model where all the hyperparameter point estimates are denoted as  $\hat{\Xi}$ . The GP prior on the super-stacked FF-model latent variables with two factors is:

$$\hat{\mathbf{f}} \sim \mathcal{N} \left( \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \hat{\mathbf{K}}_1 & \mathbf{0} \\ \mathbf{0} & \hat{\mathbf{K}}_2 \end{bmatrix}; \hat{\Xi} \right), \quad (4.39)$$

where  $\hat{\mathbf{K}}_i$  is the block-factor training covariance matrix for factor  $i$ . Each sub-block within  $\hat{\mathbf{K}}_i$  is the covariance matrix for each component within factor  $i$ . For simplicity I assume each component within a particular factor has the same covariance function (and therefore covariance matrix). In general this does not have to be the case, but the number of hyperparameters would increase considerably if each component had its own covariance function and set of hyperparameters. The block-factor training

covariance matrix in a 2-component model with each component having the same covariance matrix is then just:

$$\hat{\mathbf{K}}_i = \begin{bmatrix} \mathbf{K}_i & 0 \\ 0 & \mathbf{K}_i \end{bmatrix}. \quad (4.40)$$

Defining the super-block training covariance matrix as  $\hat{\mathbf{K}}_f$  Eq. 4.39 can be rewritten as,

$$\hat{\mathbf{f}} \sim \mathcal{N}(\mathbf{0}, \hat{\mathbf{K}}_f; \hat{\Sigma}). \quad (4.41)$$

The super-stacked latent variable prediction prior is also just a GP prior just like Eq. 4.41,  $\hat{\mathbf{f}}_* \sim \mathcal{N}(\mathbf{0}, \hat{\mathbf{K}}_{**}; \hat{\Sigma})$ , where the subscript \*\* denotes the super-block prediction points self-covariance matrix. The term self-covariance means the covariance matrix is between just the prediction points themselves. This is an important distinction since in order to make predictions the cross-covariance between the training points and the predictions must be computed and will be denoted in super-block form as  $\hat{\mathbf{K}}_{f,*}$ . The joint prior between the super-stacked training and prediction latent variables is:

$$\begin{bmatrix} \hat{\mathbf{f}} \\ \hat{\mathbf{f}}_* \end{bmatrix} \sim \mathcal{N} \left( \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \hat{\mathbf{K}}_f & \hat{\mathbf{K}}_{f,*} \\ \hat{\mathbf{K}}_{f,*}^T & \hat{\mathbf{K}}_{**} \end{bmatrix}; \hat{\Sigma} \right). \quad (4.42)$$

The super-stacked joint prior in Eq. 4.42 is conceptually identical to the standard GP joint prior in Eq. ???. The only differences are in the construction of the super-block covariance matrices but the predictions can be conditioned on the training variables in exactly the same way using standard MVN theory. The super-stacked latent variable posterior predictive (conditional) distribution is then:

$$\hat{\mathbf{f}}_* | \hat{\mathbf{f}}, \hat{\Sigma} \sim \mathcal{N} \left( \mathbf{E} \left[ \hat{\mathbf{f}}_* | \hat{\mathbf{f}} \right], \text{cov} \left( \hat{\mathbf{f}}_* | \hat{\mathbf{f}} \right) \right), \quad (4.43)$$

where the posterior predictive (conditional) mean is:

$$\mathbb{E} \left[ \hat{\mathbf{f}}_* | \hat{\mathbf{f}}, \hat{\boldsymbol{\varepsilon}} \right] = \hat{\mathbf{K}}_{\mathbf{f},*}^T \hat{\mathbf{K}}_{\mathbf{f}}^{-1} \hat{\mathbf{f}}, \quad (4.44)$$

and the posterior predictive (conditional) covariance matrix is:

$$\text{cov} \left( \hat{\mathbf{f}}_* | \hat{\mathbf{f}}, \hat{\boldsymbol{\varepsilon}} \right) = \hat{\mathbf{K}}_{**} - \hat{\mathbf{K}}_{\mathbf{f},*}^T \hat{\mathbf{K}}_{\mathbf{f}}^{-1} \hat{\mathbf{K}}_{\mathbf{f},*}. \quad (4.45)$$

The term I put in parantheses - conditional - is actually very important here, because even though the standard GPR posterior predictive distribution was also a conditional distribution it was conditioned on the training output directly. The above posterior predictive conditional distribution is not conditioned on the training output, but rather the training latent variables. Thus to get the posterior predictions conditions only the training dataset rather than the training latent variables, the training latent variables must be integrated out. But we do not know the training latent variables own conditional distribution on the training dataset analytically. The FFGP emulator MCMC calibration summarizes the training latent variable posterior distributions as a large number of samples, so I empirically estimate the means and covariance matrix from those samples. Thus the true end result of the Empirical Bayes FFGP emulator calibration process is the latent variable empirically estimated means and covariance matrix. Due to the GP prior, the latent variables are biased to be Gaussian, but their posterior distributions are not necessarily Gaussian. From examining results from multiple problems used in this work, many of the training latent variables have nearly Gaussian posterior distributions. But a substantial fraction can also be very non-Gaussian with multi-modal posterior distributions. The Gaussian approximation therefore over estimates the variance of some latent variables.

The predictive distribution will also be approximated as a Gaussian with the posterior predictive distribution estimated using the Law of Total Expectation [8]:

$$\mathbb{E} \left[ \hat{\mathbf{f}}_* | \mathcal{D}, \hat{\Xi} \right] = \int \mathbb{E} \left[ \hat{\mathbf{f}}_* | \hat{\mathbf{f}}, \hat{\Xi} \right] p \left( \hat{\mathbf{f}} | \mathcal{D}, \hat{\Xi} \right) d\hat{\mathbf{f}}. \quad (4.46)$$

Substituting in Eq. 4.44 gives:

$$\mathbb{E} \left[ \hat{\mathbf{f}}_* | \mathcal{D}, \hat{\Xi} \right] = \hat{\mathbf{K}}_{\mathbf{f},*}^T \hat{\mathbf{K}}_{\mathbf{f}}^{-1} \int \hat{\mathbf{f}} p \left( \hat{\mathbf{f}} | \mathcal{D}, \hat{\Xi} \right) d\hat{\mathbf{f}}. \quad (4.47)$$

The part of the expression within the integral is simply the mean of the super-stacked training latent variables. Thus, the posterior predictive super-stacked latent variable means is:

$$\mathbb{E} \left[ \hat{\mathbf{f}}_* | \mathcal{D}, \hat{\Xi} \right] = \hat{\mathbf{K}}_{\mathbf{f},*}^T \hat{\mathbf{K}}_{\mathbf{f}}^{-1} \mathbb{E} \left[ \hat{\mathbf{f}} | \mathcal{D}, \hat{\Xi} \right]. \quad (4.48)$$

The Law of Total Covariance is used to estimate the posterior predictive covariance of the super-stacked latent variables. In words the Law of Total Covariance sums up the mean of the predictive conditional covariance with the covariance of the predictive conditional means, which is given as:

$$\text{cov} \left( \hat{\mathbf{f}}_* | \mathcal{D}, \hat{\Xi} \right) = \mathbb{E} \left[ \text{cov} \left( \hat{\mathbf{f}}_* | \hat{\mathbf{f}}, \mathcal{D}, \hat{\Xi} \right) \right] + \text{cov} \left( \mathbb{E} \left[ \hat{\mathbf{f}}_* | \hat{\mathbf{f}}, \mathcal{D}, \hat{\Xi} \right] \right). \quad (4.49)$$

Plugging in Eqs. 4.44 and 4.45 into the Law of Total Covariance and re-arranging the terms yields:

$$\text{cov} \left( \hat{\mathbf{f}}_* | \mathcal{D}, \hat{\Xi} \right) = \hat{\mathbf{K}}_{**} - \hat{\mathbf{K}}_{\mathbf{f},*}^T \left( \hat{\mathbf{K}}_{\mathbf{f}}^{-1} - \hat{\mathbf{K}}_{\mathbf{f}}^{-1} \text{cov} \left( \hat{\mathbf{f}} | \mathcal{D}, \hat{\Xi} \right) \hat{\mathbf{K}}_{\mathbf{f}}^{-1} \right) \hat{\mathbf{K}}_{\mathbf{f},*}. \quad (4.50)$$

Equations 4.48 and 4.50 are the approximate posterior predictive (super-stacked)

latent variable mean and covariance matrix. They are approximations because the training latent variable posterior distribution was approximated by a Gaussian with empirically estimated means and covariance matrix from the emulator calibration samples.

With the latent variable predictions known, now the FF-model predictions can be estimated. The FF-model predictive distribution is approximated as a Gaussian so the mean and variance for each predictive point is computed. To be consistent with the original FF-model setup, the super-stacked predictive latent variables are reorganized back into matrix form by first splitting up the super-stacked vector into the two factor vectors,  $\hat{\mathbf{f}}_* = \begin{bmatrix} \hat{\mathbf{f}}_{1*}^T \\ \hat{\mathbf{f}}_{2*}^T \end{bmatrix}^T$ , then by reshaping the stacked-factor vectors into matrices,  $\mathbf{F}_{1*} = \text{vec}^{-1}(\hat{\mathbf{f}}_{1*})$  and  $\mathbf{F}_{2*} = \text{vec}^{-1}(\hat{\mathbf{f}}_{2*})$ . The estimated FF-model predictive means are stored in an  $M_* \times N_*$  matrix denoted as  $\mathbf{H}_*$ , where  $M_*$  is the number of predictive control variable locations we wish to make and  $N_*$  is the number of cases we are making predictions at. In general, the FFGP emulator can therefore make a predictions at a large number of case runs all at once, which is something a computer code cannot do unless multiple instances are run simultaneously. But for the uncertain parameter calibration process we will only be making predictions at one case run at a time since each MCMC iteration gives one proposal for each uncertain parameter value. But to keep things general, the expressions will focus on the predictive FF-model distribution at the  $(m_*, n_*)$ -th predictive point where  $m_* = 1, \dots, M_*$  and  $n_* = 1, \dots, N_*$ . The main reason I am switching from matrix notation to a single point notation is because it makes the math easier to write out.

The FF-model approximate predictive mean requires computing the expectation

of the product of the two latent variable factors,

$$\mathbb{E}[\mathbf{H}_*(m_*, n_*)] = \sum_{k=1}^K \mathbb{E}[\mathbf{F}_{1*}(m_*, k) \mathbf{F}_{2*}(n_*, k)]. \quad (4.51)$$

The  $k$ -th component in the summation in Eq. 4.51 is simply the standard result for the product of two correlated random variables:

$$\begin{aligned} \mathbb{E}[\mathbf{F}_{1*}(m_*, k) \mathbf{F}_{2*}(n_*, k)] &= \mathbb{E}[\mathbf{F}_{1*}(m_*, k)] \mathbb{E}[\mathbf{F}_{2*}(n_*, k)] \\ &+ \text{cov}(\mathbf{F}_{1*}(m_*, k), \mathbf{F}_{2*}(n_*, k)). \end{aligned} \quad (4.52)$$

The FF-model approximate predictive variance is:

$$\text{var}(\mathbf{H}_*(m_*, n_*)) = \text{var}\left(\sum_{k=1}^K \{\mathbf{F}_{1*}(m_*, k) \mathbf{F}_{2*}(n_*, k)\}\right) + \sigma_n^2, \quad (4.53)$$

which is the variance of the summation of products of random variables (where again the summation is over the number of components used in the FF-model) along with the FF-model likelihood noise added in. Writing out the expression completely gives:

$$\begin{aligned} \text{var}(\mathbf{H}_*(m_*, n_*)) &= \sigma_n^2 + \sum_{k=1}^K \text{var}(\mathbf{F}_{1*}(m_*, k) \mathbf{F}_{2*}(n_*, k)) \cdots \\ &\cdots + 2 \sum_{1 \leq k < k' \leq K} \text{cov}(\mathbf{F}_{1*}(m_*, k) \mathbf{F}_{2*}(n_*, k), \mathbf{F}_{1*}(m_*, k') \mathbf{F}_{2*}(n_*, k')). \end{aligned} \quad (4.54)$$

The predictive covariance structure required in Eqs. 4.51 and 4.54 is quite complex and adds considerable computational time compared to assuming the predictive factors are independent. But, neglecting the latent predictive covariance structure can significantly over-estimate the FF-model predictive variance.

### 4.3.2 FFGP Emulator-based Uncertain Parameter Calibration

With the FFGP emulator posterior predictive distribution approximated as a Gaussian distribution a modified likelihood can be formulated similarly to the GPR-modified likelihood function described in Section 4.2.3. The main difference between the GPR-modified and the approximate FFGP-modified likelihood of course is the FFGP emulator is acting as the likelihood model between the observational data and the uncertain parameters,  $\theta$ . I always use factor 1 as the control variable factor and factor 2 as the uncertain parameter factor. As stated earlier only a single case run is being predicted at each MCMC iteration, so  $N_* = 1$ . The number of control variable predictions must at least equal the number of observational points. More control variable locations can be predicted if desired, but then the predictions that correspond to the appropriate control variable locations must be properly separated. For convenience for now, assume  $M_* = N_O$ . The FFGP emulator predictions are then a vector of size  $(N_O \times 1)$ .

The joint-posterior distribution between the emulator predictions,  $\mathbf{H}_*$ , and uncertain parameters,  $\theta$  is:

$$p(\mathbf{H}_*, \theta | \mathbf{y}_o, \mathcal{D}, \hat{\Xi}) \propto p(\mathbf{y}_o | \mathbf{H}_*) p(\mathbf{H}_* | \{\mathbf{x}_{cv,o}, \theta\}, \mathcal{D}, \hat{\Xi}) p(\theta). \quad (4.55)$$

Just as in Section 4.2.3 the likelihood model between the observational data and predictions is assumed to be Gaussian with a known observational error matrix  $\Sigma_\epsilon = \sigma_\epsilon^2 \mathbf{I}$ . Integrating out the predictions gives the uncertain parameter posterior distribution

which looks very similar in form to 4.23:

$$p\left(\theta|\mathbf{y}_o, \mathcal{D}, \hat{\Xi}\right) \propto p\left(\mathbf{y}_o|\{\mathbf{x}_{cv,o}, \theta\}, \mathcal{D}, \hat{\Xi}\right) p(\theta). \quad (4.56)$$

The FFGP-modified likelihood factorizes for simplicity each of the observational points:

$$p\left(\mathbf{y}_o|\{\mathbf{x}_{cv,o}, \theta\}, \mathcal{D}, \hat{\Xi}\right) = \prod_{l=1}^{N_o} p\left(y_{o,l}|\{x_{cv,o,l}, \theta\}, \mathcal{D}, \hat{\Xi}\right), \quad (4.57)$$

where for each observational location the FFGP-modified likelihood is:

$$p\left(y_{o,l}|\{x_{cv,o,l}, \theta\}, \mathcal{D}, \hat{\Xi}\right) \approx \prod_{l=1}^{N_o} \mathcal{N}\left(\mathbb{E}[\mathbf{H}_*(l)], \text{var}(\mathbf{H}_*(l)) + \sigma_\epsilon^2\right). \quad (4.58)$$

## 4.4 Calibration Demonstration

In order to truly know if the surrogate model approach is working, the surrogate models must be able to replicate the uncertain parameter posterior distributions resulting from Bayesian model calibration. If they are able to that means they accurately capture the potentially complicated input/output relationship of the computer model, and replicates the amount of new information gained by conditioning against observational data. Chapter 2 used a simple friction factor model to demonstrate Bayesian inference with MCMC. That same model will be used here as the “long-running” computer code we wish to emulate, but since it is not in fact computationally expensive the surrogate model results will be compared directly to the inference results found using the simulator in the MCMC sampling. This also demonstrates the steps involved in a real problem where the computer code is in fact too long to run to use in MCMC inference. These steps are:



- From the uncertain parameter prior distributions, choose bounding values on each of the uncertain parameters
- Generate “space-filling” training input values (I use the built in MATLAB function `lhsdesign.m`)
- Generate the training output by running the computer code at each of the training input points
- Build the emulator(s)
- Calibrate the uncertain parameters using MCMC inference with the emulator in place of the computer code

The simple friction factor model is repeated here for convenience, but it is important to not confuse the friction factor  $f$  with the latent variable notation:

$$f = \exp(b)Re^{-\exp(c)}. \quad (4.59)$$

The following discussion will also focus on why additional components in the FFGP model can improve performance, since the previous section only demonstrated a 2-factor 1-component model. The same method of manufactured solution approach is used where the “observational data” is generated using preset true values of the uncertain parameters. But now there is the additional layer of the emulator that must figure out those true uncertain parameters, not simply the computer model itself.

#### 4.4.1 One Uncertain Parameter FFGP model

To start out, only the  $b$ -parameter is considered uncertain, which means the simulator is itself the product of two separate functions, one for the control variable and one for the uncertain parameter. These two functions are simply:

$$f = g(b)g(Re), \quad g(b) = \exp(b), \quad g(Re) = Re^{-\exp(c)}. \quad (4.60)$$

The one-component FFGP model should then be able to exactly model this simulator, within the desired noise level. Fig. 4-5 below shows the training dataset along with the observational data used in this problem. The training data is in blue and consists of 15 case runs and 10 control variable locations chosen per case run ( $N = 15$  and  $M = 10$  using the notation from earlier). The observational data is shown as the red errorbars, where the errorbars correspond to  $\pm 2\sigma$  around the observational mean. The observational error was assumed to be 10% of the mean value of the friction factor. Note that the input in Fig. 4-5 is scaled between 0 and 1. I found that using that scaling of the input helps and gives better results than using the raw input ranges themselves. The raw  $Re$  number values range from 5000 to 45000, where the scaled 0 value corresponds to the raw value of 5000. The output however is not scaled and is shown at its actual raw values.

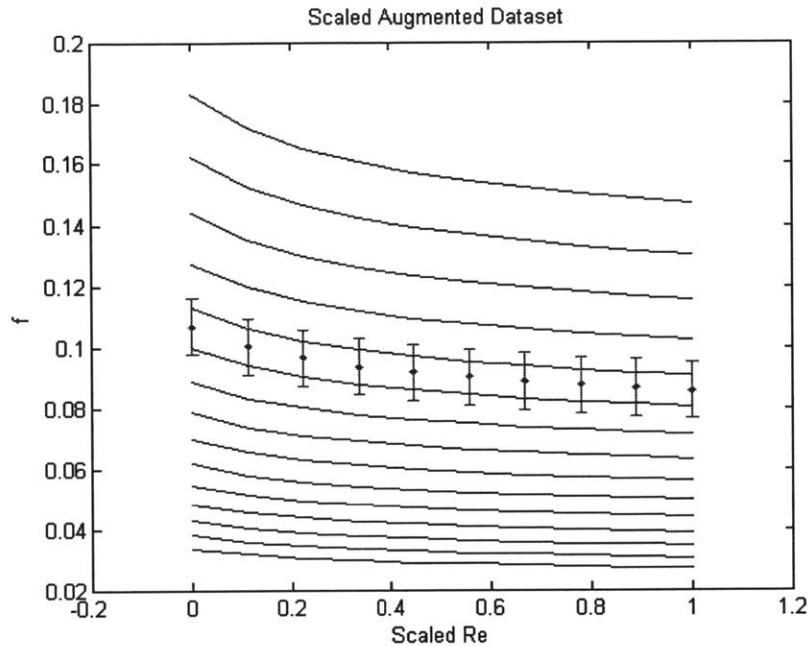


Figure 4-5: One Uncertain Parameter Training and Observational Data

The observational data shows that the true value of the  $b$ -parameter falls between two of the training case runs. Even with only one uncertain parameter, there are actually two inputs:  $Re$  and the  $b$ -parameter. If the standard GPR model was built, a space filling design would have to be used, such as Latin Hypercube Sampling (LHS) to generate input values that covers the input space, to try and maximize the information gained from each training point. But as discussed earlier, maximizing the information gained for each factor corresponds to holding the other factor's input constant and so it is very simple to choose training input values here. With only one input per factor, the “space filling” design is simply choosing equally spaced points in each input direction.

Emulator calibration is performed as described previously using HMC to sam-

ple the latent variables and RWM to sample the hyperparameters. The posterior results of the observation space training points are shown in Fig. 4-6 below. The red dots are the actual training output and although it is difficult to see, but the blue lines are the posterior quantiles on the emulator predictions at those training points. The quantiles are the 5<sup>th</sup>, 25<sup>th</sup>, 50<sup>th</sup>, 75<sup>th</sup>, and 95<sup>th</sup>-percent quantiles and are just very tightly packed together representing how little uncertainty exists in the posterior predictions. Again this has to do with the FFGP model capable of exactly representing the simulator in this case.

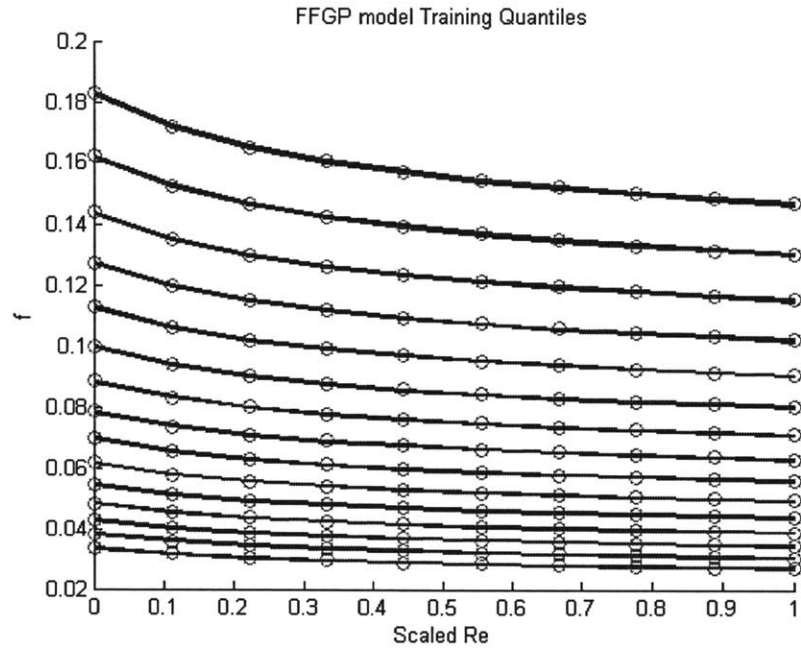


Figure 4-6: Posterior Observational Space Training Points

The calibrated emulator is then used in place of the simulator to calibrate the uncertain  $b$ -parameter. The  $b$ -parameter prior is assumed Gaussian with a mean value set at the McAdam’s friction factor correlation value,  $\log(0.184)$  and a variance

set so that the 95% of the prior probability is  $\pm 50\%$  around the prior mean. Scaling the  $b$ -parameter between 0 and 1 means that the prior mean is simply 0.5 and the prior standard deviation is 0.25. Posterior samples are drawn using RWM sampling due to the simplicity of this problem and the posterior samples are shown in Fig. 4-7. The red line is the true scaled value and as shown in the figure, the sampling quickly finds the true value and exhibits very well mixed behavior. Fig. 4-8 shows the estimated density of the posterior distribution in blue, the prior distribution in black, and the true value as the vertical red line. This shows how precise the posterior distribution on the  $b$ -parameter is.

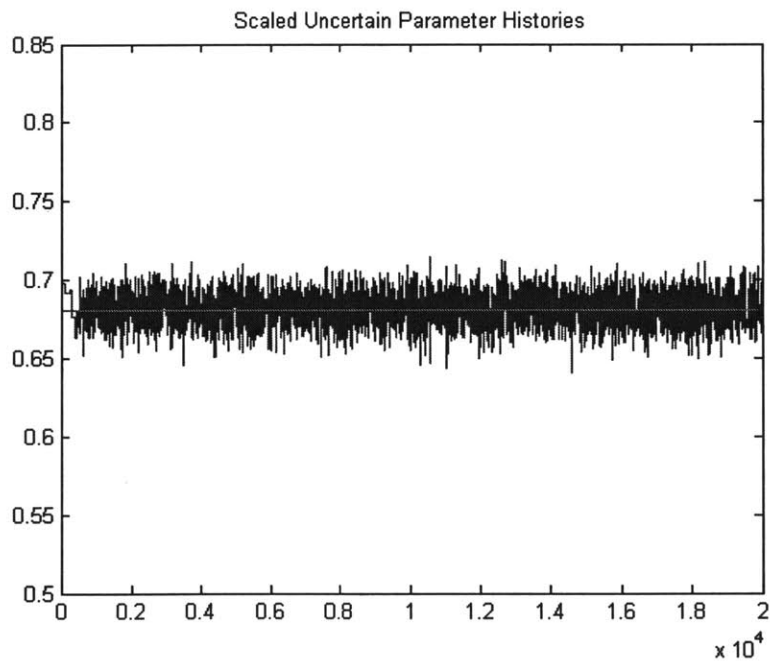


Figure 4-7: Scaled Uncertain Parameter Sample Histories

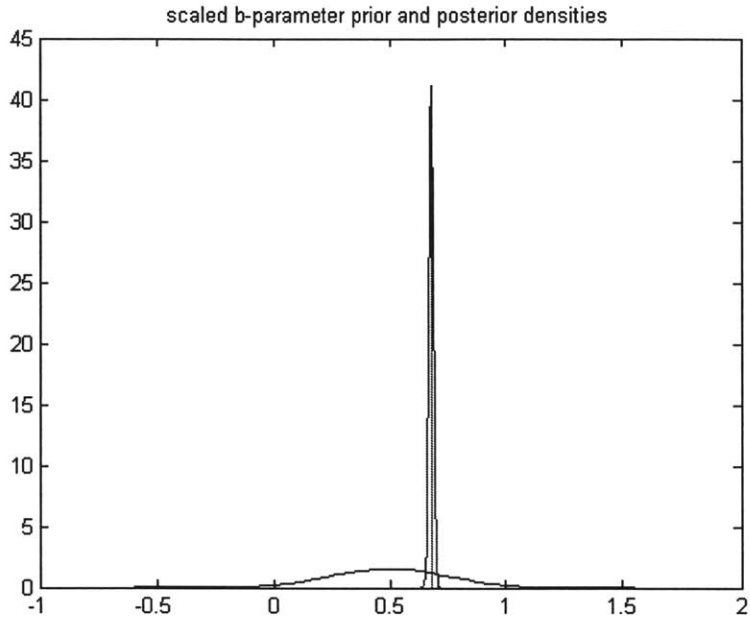


Figure 4-8: Estimated Scaled Posterior and Prior Densities

As already described, it was expected that for this simple demonstration the FFGP-model should perform very well. But it confirms that the FFGP model is at least working as expected.

#### 4.4.2 Two Uncertain Parameters FFGP model

Next, both the  $b$  and  $c$ -parameters are assumed uncertain. Sticking with a two-factor model means that the uncertain parameter factor is now a very complicated function and the FFGP-model assumption is no longer exactly correct. It is easy to see why this should be difficult for the two-factor one-component model, since one of the factors is now written as the latent function  $g(b, c)$  which cannot be written down

explicitly. A three-factor model could be used, but since the “book keeping” for that setup is more complicated I’ve stuck to the two-factor model approach and will add flexibility by adding additional components. The downside of doing this though is the training set for the uncertain parameters’ factor must now be “space filling”. I use latin hypercube sampling (LHS) to generate the uncertain parameter values with a built in MATLAB LHS function, `lhsdesign.m`. This function is not guaranteed to give the optimal training dataset and in fact the values it finds are usually very different when running it multiple times. Finding better points would improve the predictive capability, but for now this LHS procedure will be used.

The LHS generated training dataset is shown in Fig. 4-9 along with the observational data as red error bars as before. Fifty case runs were made and twenty five points were taken per case, giving  $N = 50$  and  $M = 25$ . Using more case helped guarantee the training dataset would “surround” or cover the observational data. Additionally, for comparison purposes I built a standard GPR emulator of the friction factor model, which as previously discussed has a different input structure than the FFGP emulator. The general rule of thumb for standard GPR emulator training set sizes is to use 10 training points for input [13]. With 2 uncertain parameters and 1 control variable, 30 training points should be enough but just to be on the safe side I increased that number to 50 training points. The GPR and FFGP training sets have the same case runs, but as shown by Fig. 4-9, the GPR training set takes only one point per case run while the FFGP emulator uses all 25 control variable locations. This illustrates how the FFGP emulator decomposes, or literally factorizes, the training set into smaller subsets. The FFGP emulator uses a total of  $NM = 1250$  training points but each factor has covariance matrices of sizes  $(M \times M) = (25 \times 25)$  for factor 1 and  $(N \times N) = (50 \times 50)$  for factor 2. If every training point was also used for the GPR emulator then the covariance matrix would

be size  $(NM \times NM) = (1250 \times 1250)$ . The FFGP emulator setup therefore can drastically reduce the computational burden and facilitate using as many training points as possible.

The other important point to take away from Fig. 4-9 is that choosing the training set is easier for the FFGP emulator than the GPR emulator. The true uncertain parameter values were set so that the observational data would not be near the cluster of training points. And as shown in Fig. 4-9 there are very few GPR training points near the observational data. Even though the FFGP training set uses the exact same case runs, since more control variable locations can be taken per case run, more information is learned from each case run. In general the training datasets shown in Fig. 4-9 are quite poor in the context of using the emulators for uncertain parameter calibration. Only a few case runs even lie within the error bars of the data. This suggests the prior bounds on the uncertain parameters which built the training set were very poor. And since this is a simple model with only 2 uncertain parameters it would be relatively easy to change the prior bounds to better surround the observational data. But, in a real problem with many uncertain parameters that interact in very complicated ways to produce the output modifying constantly rebuilding the training set would not only be difficult but potentially very time consuming. That is why this demonstration problem was purposely setup this way. It is more representative of the sometimes difficult task of building a training set from prior information and the goal is to show if the emulators can nearly reproduce the uncertain parameter calibration results if the simulator was used directly.

The emulators were built using the algorithms described previously. Unfortunately due to the different output scaling between the GPR emulator and FFGP emulator formulations it is difficult to compare the performance of the two emulator types after training is complete. But a simple way to compare the FFGP



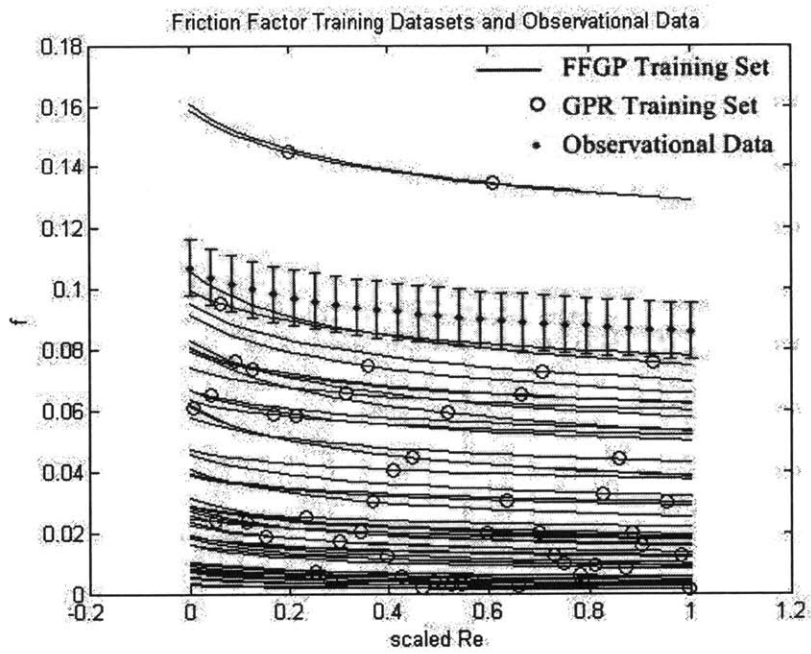


Figure 4-9: Two Uncertain Parameters Training And Observational Data

emulator training results between different components is through the likelihood noise hyperparameter,  $\phi_n$ . The more negative  $\phi_n$  is the smaller the likelihood noise since  $\sigma_n^2 = \exp(2\phi_n)$ . Figures 4-10 and 4-11 below show the FFGP likelihood noise hyperparameter training results for the 1-component and 2-component emulators, respectively. In each figure, the grey line is the initial guess, the blue line shows the samples and the red line shows the point estimate. It is very clear the 2-component FFGP emulator is far more accurate relative to the training set than the 1-component FFGP emulator. The  $\phi_n$  point estimate for the 1-component emulator gives a likelihood noise term that has a standard deviation over 45x that of the 2-component emulator. This illustrates the point how the 2-factor 1-component FFGP emulator is no longer an exact assumption of the simulator model and thus the additional component in the 2-factor 2-component emulator provides the extra flexibility needed to match the training set more accurately. The other point to take away from Figs. 4-10 and 4-11 is how once the  $\phi_n$  values reach a “steady” point the sampling rate drastically drops off. This is why the Empirical Bayes approach was used for the FFGP emulator training process.

With the FFGP emulators built they are used to calibrate the uncertain parameters using the FFGP-modified likelihood format described earlier. The uncertain parameters were sampled using the AM MCMC algorithm discussed in Chapter 2. A total of  $10^5$  samples were made with the first half discarded as burn-in. The calibrated posterior predictions for the 1-component and 2-component emulators are shown in Figs. 4-12 and 4-13, respectively. In both figures, the blue lines are the posterior quantiles of the predictive means and although difficult to see the black line is the mean of the predictive means. The green band is the total predictive uncertainty band of the emulator, spanning 95% of the emulator prediction probability and is  $\pm 2\sigma$  around the mean of the predictive means. The blue lines therefore represent

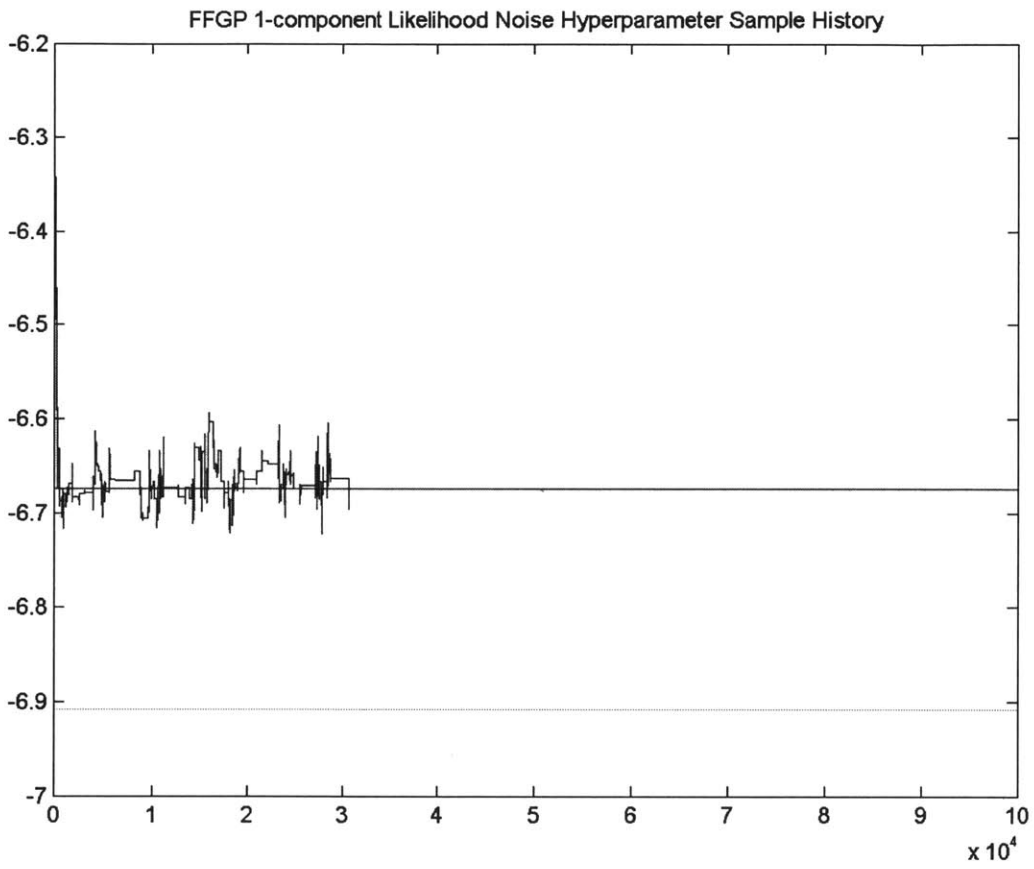


Figure 4-10: Friction Factor FFGP 2-factor 1-component likelihood noise hyperparameter

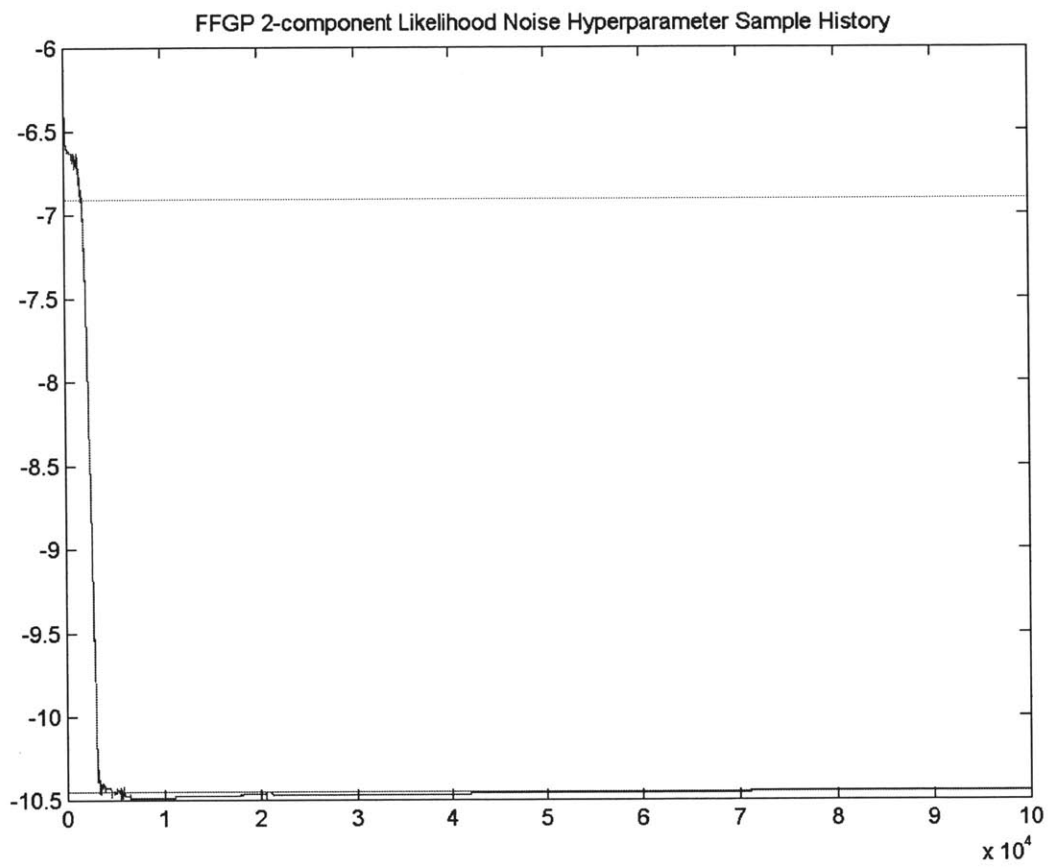


Figure 4-11: Friction Factor FFGP 2-factor 2-component likelihood noise hyperparameter

what the emulator feels the simulator uncertainty would be due to the propagating the uncertain parameter posterior distributions onto the simulator output. The green band includes the emulator predictive variance, and thus includes how uncertain the emulator is about a prediction. The closer the edge of the green band is to the blue lines means, the more certain the emulator is about its predictions. The 2-component emulator is not just far more accurate relative to the observational data than the 1-component but it is also more certain in its predictions. As shown in Fig. 4-13 the green band is very close to the spread in the blue lines, and those the emulator adds very little additional uncertainty to the predictions. The 1-component emulator however adds more uncertainty to its predictions, in addition to the fact it does not represent the trend in the observational data.

The uncertain parameters were also calibrated with the built GPR emulator using the GPR-modified likelihood format described earlier. The same number of samples were used in the AM MCMC algorithm. The GPR emulator calibrated posterior predictions are shown in Fig. 4-14, using the same format as the FFGP predictions. The GPR emulator adds less uncertainty to the predictions than the 1-component FFGP emulator but is more uncertain and less accurate relative to the data than the 2-component FFGP emulator. The predictions over the first half of the (scaled) Reynolds numbers are very accurate and are similar to the 2-component FFGP emulator predictions. But the later half of the (scaled) Reynolds number predictions, no longer have the mean prediction match the observational data and the GPR emulator predictive variance increases slightly. A better training dataset would most likely correct these inaccuracies in the GPR emulator predictions, but the fact the FFGP emulator was able to out perform the GPR emulator with this training set was encouraging.

But the real goal of using the FFGP emulators is to have a surrogate that can

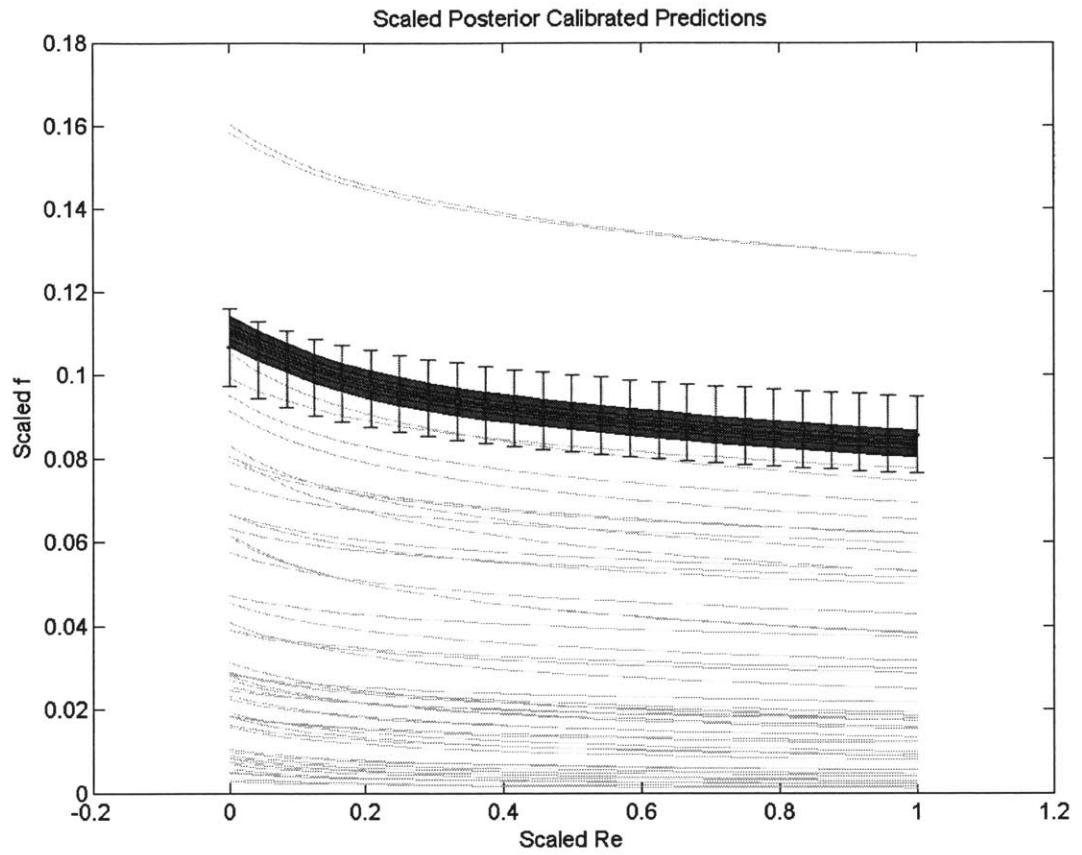


Figure 4-12: FFGP 2-factor 1-component calibrated posterior predictions

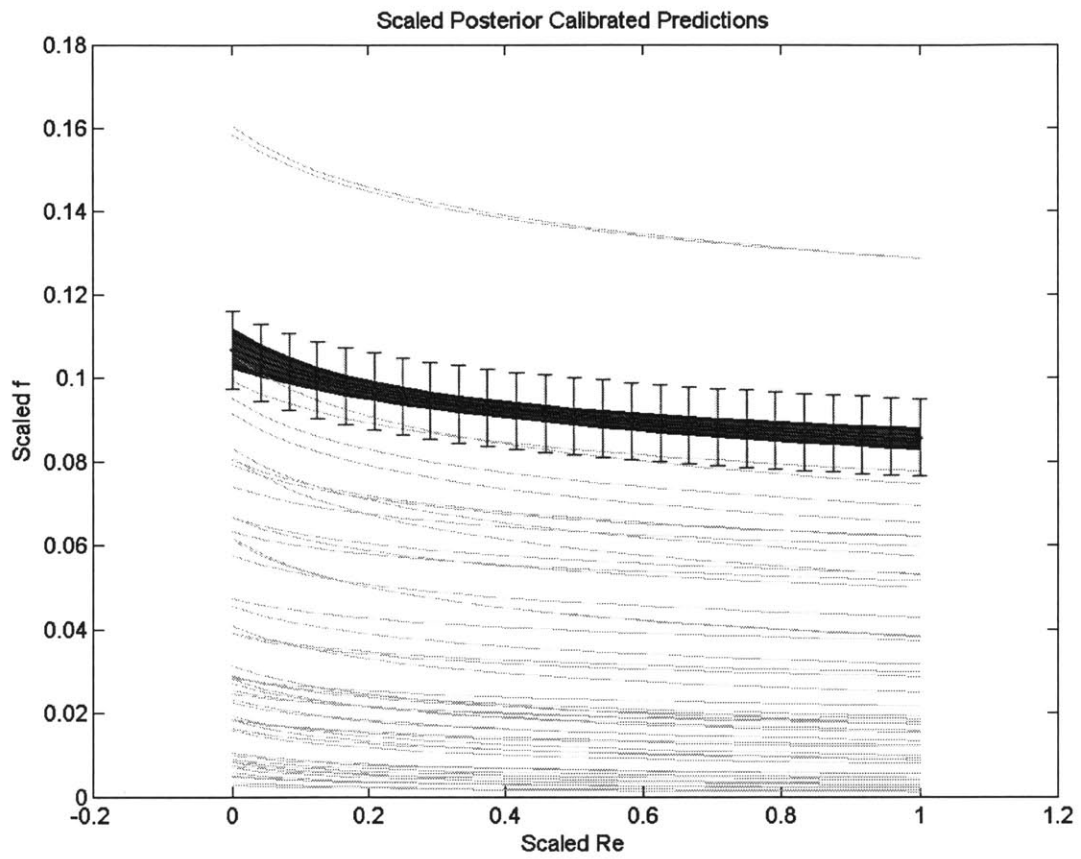


Figure 4-13: FFGP 2-factor 2-component calibrated posterior predictions

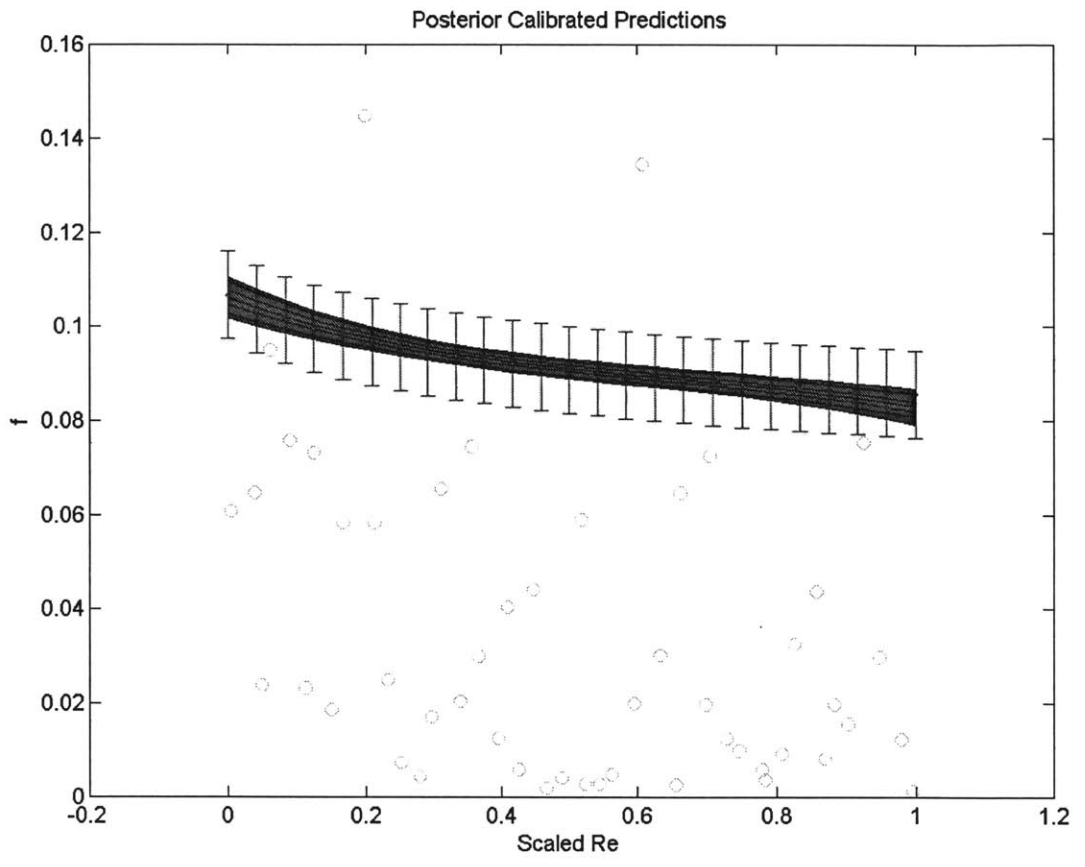


Figure 4-14: GPR calibrated posterior predictions



replace the simulator as accurately as possible in the uncertain parameter calibration process. Since the simulator in this demonstration problem, the friction factor model, is so computationally cheap we can use it in the AM MCMC sampling directly. The simulator based calibration results represent the best we could do and we want to be as close to these results as possible. Figures 4-15, 4-16, and 4-17 show the emulator based uncertain parameter posterior distributions compared to the simulator based results for the GPR, 1-component FFGP, 2-component FFGP, respectively. In each plot, the black curve is the prior distribution on the scaled parameter, where the scaled prior mean is 0.5 and has a scaled standard deviation of 0.25. The blue curve is the emulator based posterior distribution, the green curve is the simulator based posterior distribution, and the red vertical line is the true scaled parameter value. The simulator based posterior distribution are capable of finding the true parameter values quite well, with the posterior distribution modes almost directly at the true parameter values. If the observational error was assumed smaller then the posterior variance would be reduced. Although the GPR emulator is capable of finding the correct posterior modes the overall posterior distributions do not match the simulator based posterior distributions. Odd smaller second modes are present in both parameters. Most likely, these secondary modes are due to proposal values outside the training set, which would have significantly highly GPR emulator predictive uncertainty. With the GPR-modified likelihood structure a higher emulator predictive uncertainty effectively increases the overall noise. With more of the data variation explained by noise the parameters can take on values they would not normally be, since from the emulator's point of view the noisy prediction "overlaps" with the data's own error. The 1-component FFGP emulator based results also support this view, since the posterior distributions in Fig. 4-16 are still quite broad. The emulator is capable of shifting the posterior distributions in the right direction

but the additional emulator uncertainty prevents the MCMC sampling from resolving any additional information about the uncertain parameters. The 2-component FFGP emulator, however, is so accurate relative to the simulator that its uncertain parameter posterior distributions are almost identical to the simulator's uncertain parameter posterior distributions, as shown in Fig. 4-17. In more complex problems, we cannot expect the FFGP based results to always be as accurate as in this simple demonstration problem. However the FFGP emulator-based calibration is capable of matching the simulator-based calibration results as shown here. In more complex, and realistic problems however, we will not have the simulator-based results to compare with because the simulator will be just too slow to use. So it was important to verify through this method of manufactured solution problem that the FFGP emulator works as we expect it to.

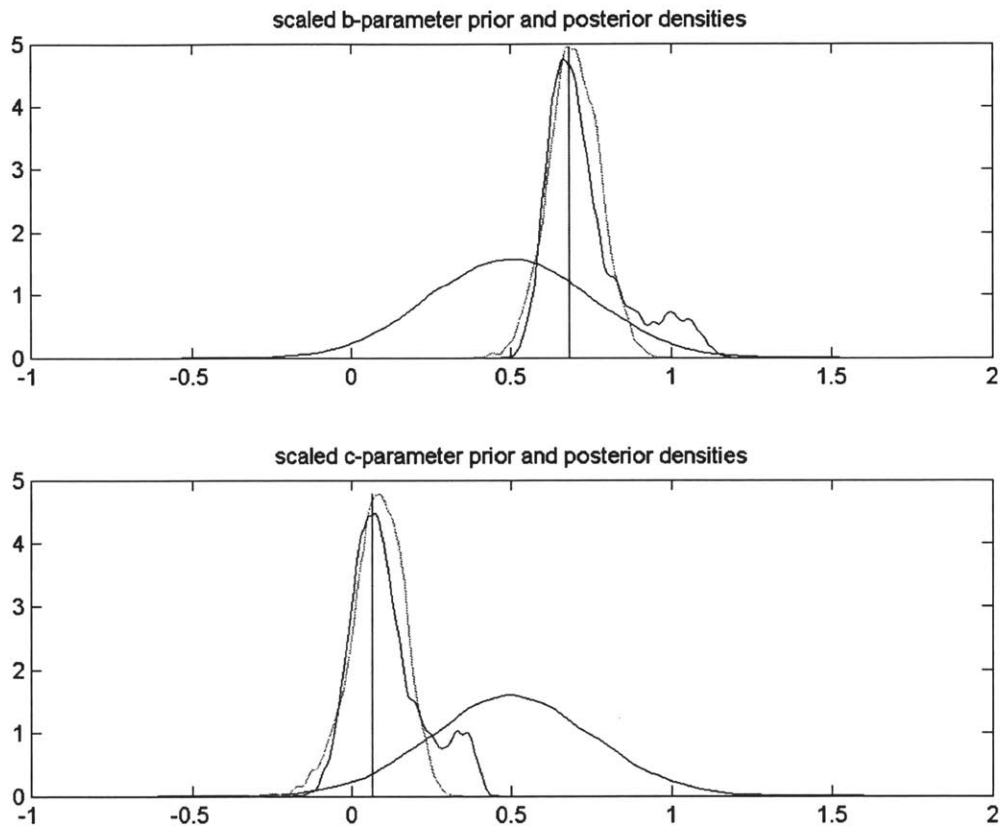


Figure 4-15: GPR-based uncertain parameter posterior distributions

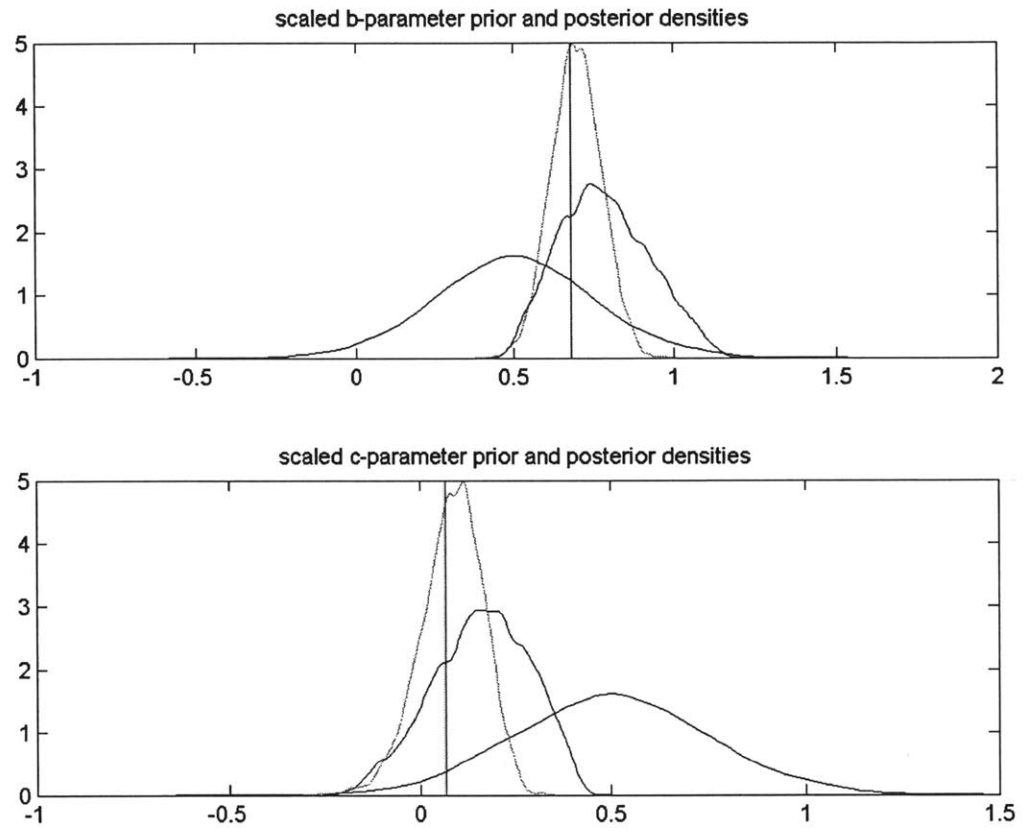


Figure 4-16: 1-component FFGP-based uncertain parameter posterior distributions

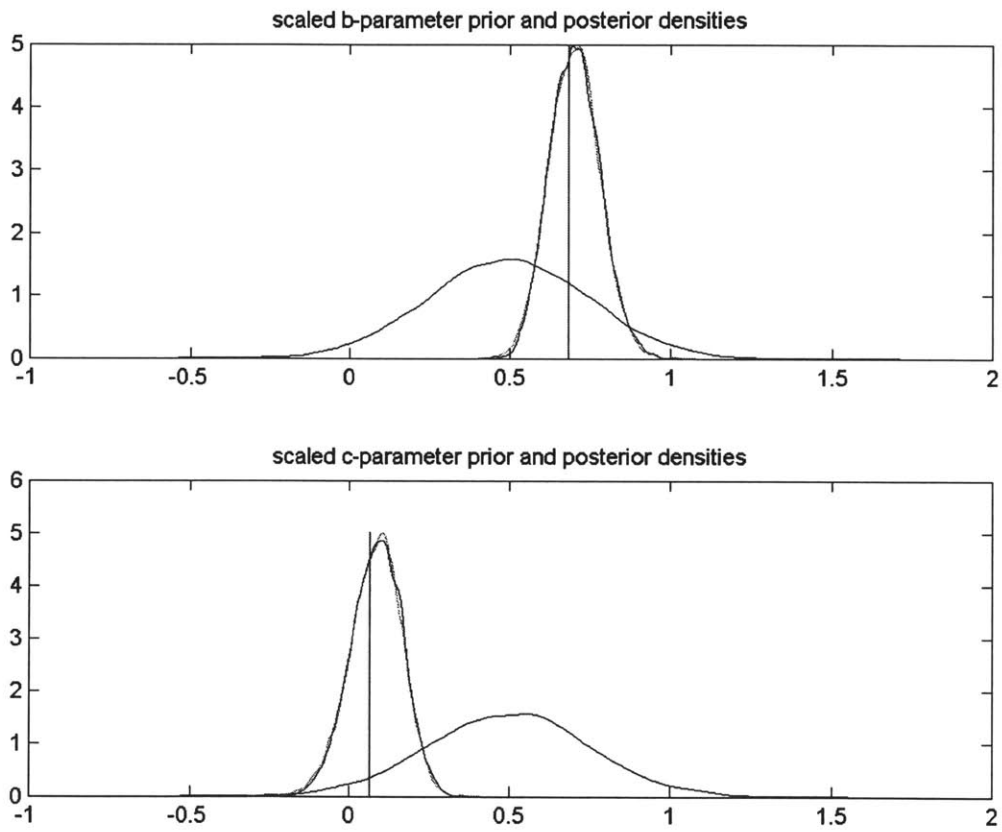


Figure 4-17: 2-component FFGP-based uncertain parameter posterior distributions



## Chapter 5

# Blowdown Problem Demonstration

The QPIRT + emulator-based approach to performing Bayesian model calibration will be demonstrated on a conceptually simple gas blowdown model. The long-running computer code in this case is a numerical solver of a gas blowdown transient which models a benchmark blowdown experiment of a vertical cylindrical vessel filled with nitrogen, performed at London's Imperial College [35]. This problem was also analyzed by Petruzzi et al. (2010) [36] as a demonstration of a data assimilation method for model calibration. Their approach, however, relies on the computation of local sensitivities in order to identify important parameters as well as to propagate parameter uncertainty onto the figure of merit (FOM), developed by Cacuci [37]. Petruzzi and Cacuci's work provides a useful comparison for the results of the current approach, but their approach requires the first derivatives to be computed alongside the computer model prediction. For complicated non-linear thermal hydraulic codes that we are ultimately interested in, setting up those computations in general may be very difficult, though adjoint methods [38] or automatic differentiation techniques [15] could be applied. Also, if the first derivatives are known, they can

be incorporated into the emulator framework to improve the emulator predictive accuracy by conditioning not only on the known output, but on the known gradient information as well [15], [39]. The gradient provides even more information about the relationship between the inputs and the outputs and therefore would only improve the emulator-based approach.

The QPIRT + emulator-based calibration approach has the following steps: First start out with the original computer model at the nominal parameter settings. These nominal values will most likely be set by expert opinion. The QPIRT process from Chapter 3 is applied in order to identify the key physical phenomena that dominate the FOM response. From there, the uncertain parameters within the various models of those key physical processes are identified in the QPIRT Bottom-Up step. Using the prior distributions on the uncertain parameters, which are most likely set by expert opinion, the original computer model (the simulator) is re-run a specified number of times to generate the training output data. The prior distributions set the maximum and minimum value bounds on the uncertain parameters when using space filling designs such as Latin Hypercube Sampling (LHS). Alternatively, the prior distributions could be used to bias the training input values so that they are clustered around certain values. In either case, the priors are what set the training input values. The training data set is then used to build the desired number of emulators, which as shown in Chapter 4 can range in levels of complexity. Following this, the uncertain parameters are calibrated using the MCMC techniques described in Chapter 2. The experimental data is required at this step. The GPR and FFGP-modified likelihood functions described in Chapter 4 are used in place of the simulator-based likelihood function in Chapter 2. The emulator contributed additional uncertainty is therefore completely accounted for. If multiple emulators are built, the calibration process can be run in parallel for each emulator type.



With multiple types of emulators, Bayesian Model Selection would identify the emulator that best explains the training data, by fulfilling Bayesian Occam's razor [8]. Occam's razor states that the simplest answer is usually the best and how it applies to model selection will be described later on. But for now, multiple emulators are used to compare the performance of the standard GPR model to the various FFGP levels of complexity, and the "best" emulator will not be quantitatively identified.

The current approach builds an emulator and then fixes that emulator when calibrating the uncertain parameters. Thus, there are two calibration steps: emulator calibration (which builds the emulator) followed by emulator-based calibration of the uncertain parameters. Alternatively, the emulator could be calibrated simultaneously with the uncertain parameters [28], [29]. Ultimately, both approaches create an emulator-modified likelihood function. As shown in Chapter 4, the current work's emulator-modified likelihood functions use the emulator posterior predictive distributions. Whereas, the approach in References [28], [29], used emulator-modified likelihood functions formulated around the emulator prior predictive functions. The alternative approach therefore makes posterior predictions conditioned on both the training data and the observational data simultaneously. In some sense the alternative approach is more of a data or information "fusion" method [29] rather than a calibration focused approach. Figures 5-1 and 5-2 provide flow charts of the two approaches. The downside of the alternative approach though is that the emulator is not built until after the entire uncertain parameter calibration process is complete. Thus, if multiple levels of data, such as Integral and Separate Effect Tests (IETs and SETs), are present, the emulators for all of the IET and SETs must be calibrated simultaneously. As will be shown in Chapter 6, IET and SET simultaneous calibration with already built emulators (following the current approach) is challenging enough. Including the various hyperparameters from multiple emulators with the uncertain

parameters in the calibration process would only complicate the MCMC sampling further. Choosing an appropriate MCMC scheme would be very challenging and sampling would be very slow. That is why the alternative approach was not used in favor the current work's emulator-based calibration process.

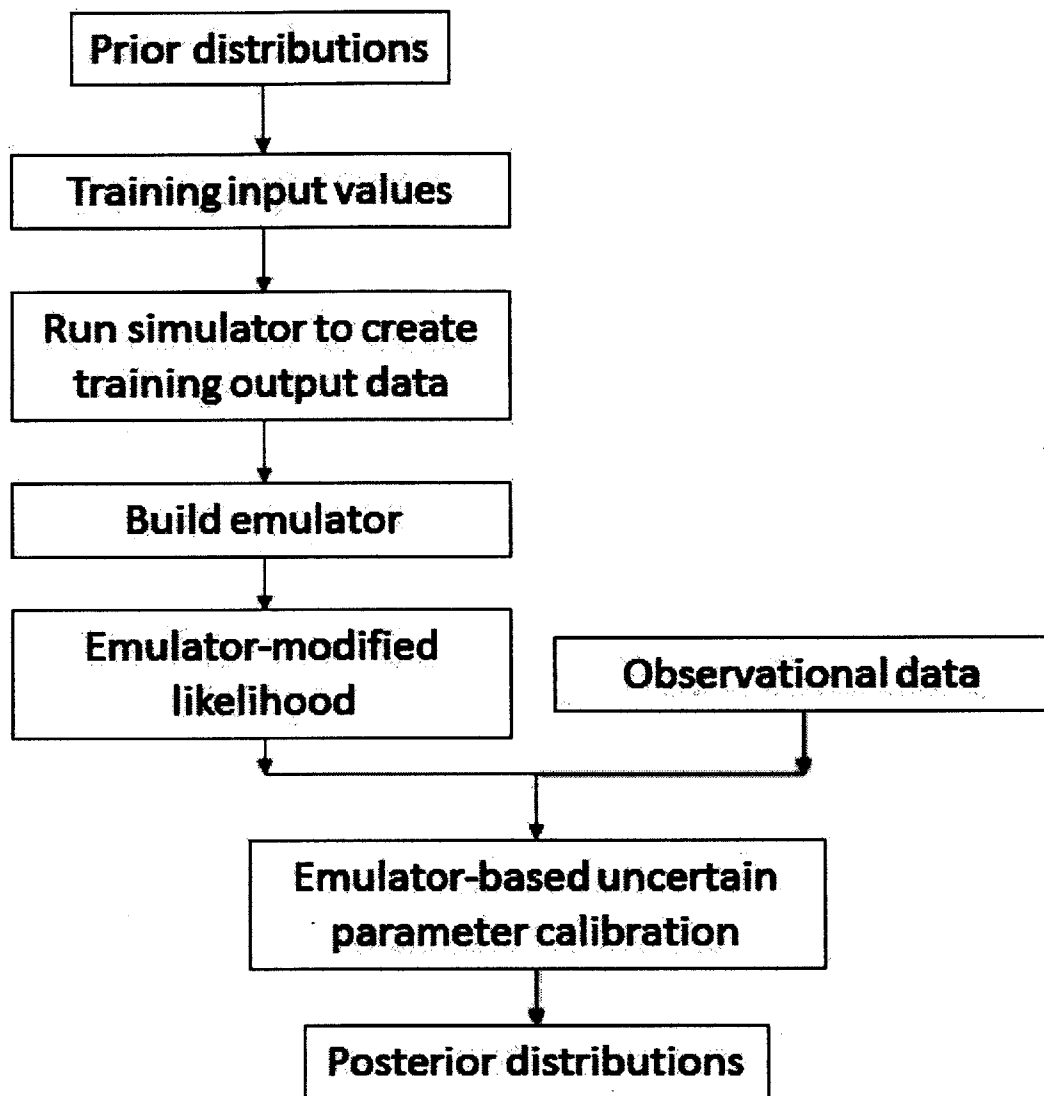


Figure 5-1: Current emulator-based calibration approach flow chart

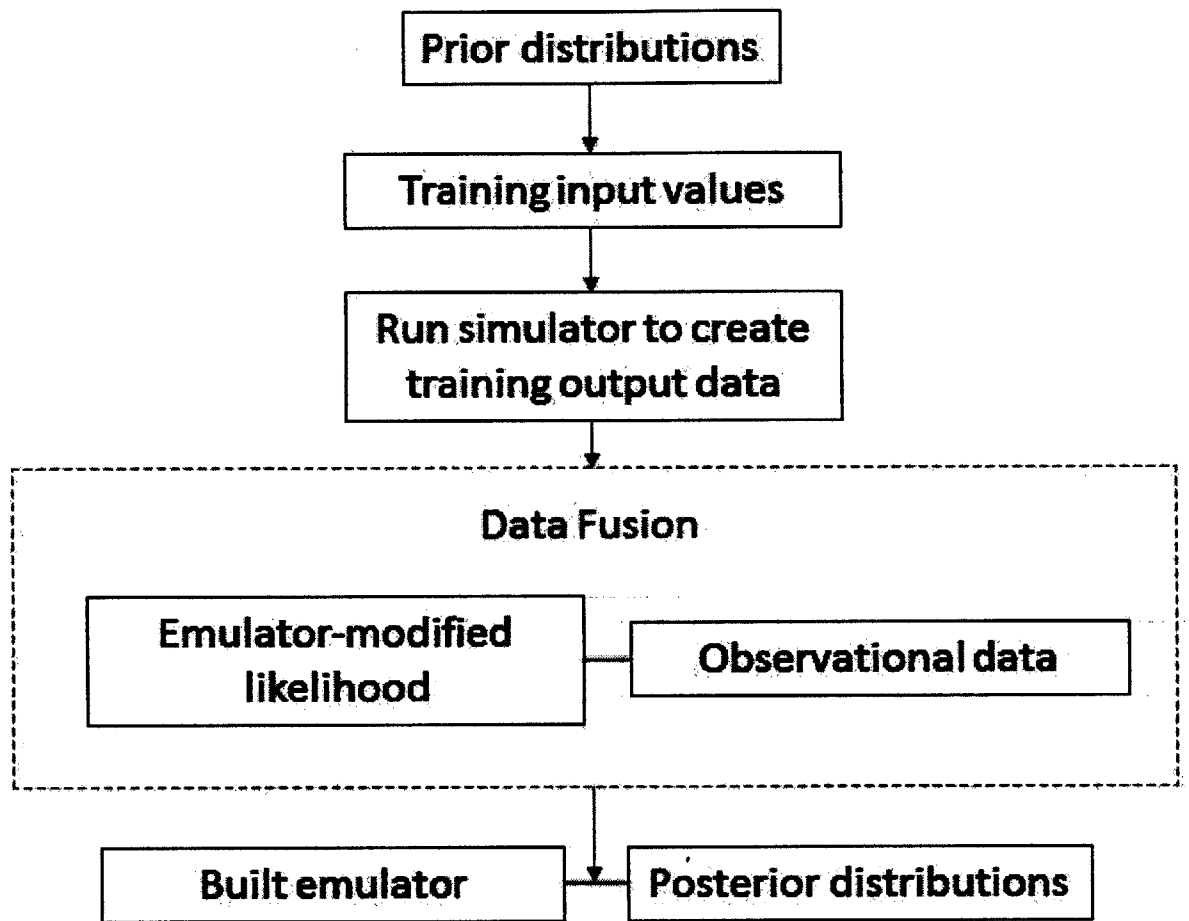


Figure 5-2: Alternative data fusion flow chart

The remaining sections of this Chapter are as follows: Section 5.1 describes the blowdown model and Section 5.2 describes the numerical solver used in [36], Section 5.3 describes how the QPIRT process is applied to that blowdown model, Section 5.4 then discusses the nominal results and the QPIRT findings. The emulator-based

calibration results are then described in Section 5.5.

## 5.1 Blowdown Model

The following notation is consistent with [36] and is therefore a break from some of the notation used previously. An illustration of the blowdown tank is given in Fig. 5-3 below. The tank is a cylindrical gas pressure vessel, of known dimensions, filled with high pressure nitrogen. Initially the tank is at the same temperature as the ambient air, outside the tank. So initially there is no heat transfer between gas, pressure vessel walls, and the ambient air. A nozzle of known flow area has a valve that is opened to start the gas blowdown. Table 5.1 gives the values for all the variables in the blowdown model. The gas tank dimensions, boundary and initial conditions, as well as the nominal uncertain parameter values are shown. The terms in Table 5.1 will be described in more detail in the sections that follow.

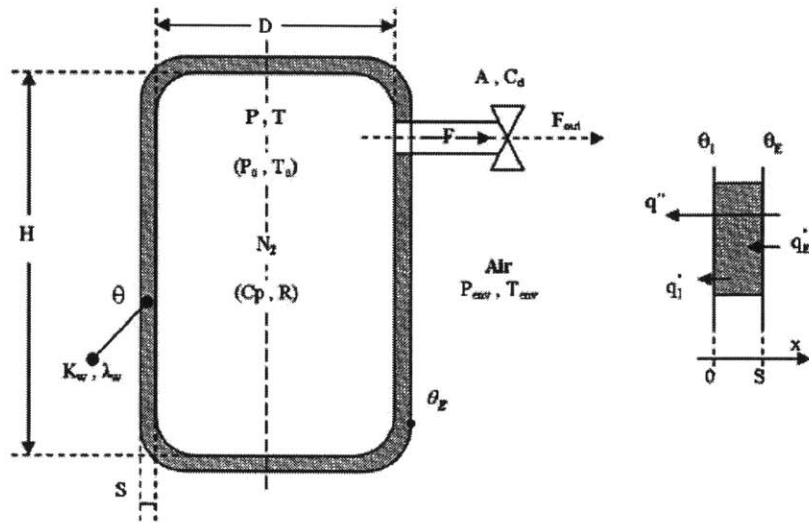


Figure 5-3: Blowdown Tank Illustration

Table 5.1: Nominal Parameter Values

Component	Description	Symbol	Value	Unit
Vessel	Height	$H$	152.4	cm
	Internal diameter	$D$	27.3	cm
	Wall thickness	$S$	2.5	cm
	Thermal conductivity	$k_w$	14.42	$\frac{W}{m-K}$
	Thermal diffusivity	$\alpha_w$	$4.5 \times 10^{-6}$	$\frac{m^2}{s}$
	Volume	$V$	89207.25	cm <sup>3</sup>
	Vessel surface area	$A_S$	14839.83	cm <sup>2</sup>
Break orifice	Equivalent area	$A$	0.31669	cm <sup>2</sup>
	Discharge coefficient	$C_d$	0.8	
Nitrogen (gas)	Specific gas constant	$R$	296.79	$\frac{J}{kg-K}$
	$c_p$ to $c_v$ ratio	$\gamma$	1.399	
Air	Pressure	$P_A$	1.01	bar
	Temperature	$T_A$	290.0	K
Initial conditions	Gas pressure	$P_0$	150.0	bar
	Gas temperature	$T_0 = T_A$	290.0	K
	Vessel wall temperature	$T_w(x, t = 0) = T_0$	290.0	K
Correlations	Laminar - constant	$a$	0.555	
	Laminar - exponent	$b$	1/4	
	Turbulent- constant	$c$	0.130	
	Turbulent - exponent	$d$	1/3	
Numerical	timestep	$\Delta t$	0.1	s
	Mesh size	$\Delta x$	0.25	cm

### 5.1.1 Governing Equations

The governing equations of interest are a mass and energy balance along with a rigid volume constraint:

$$\frac{dM}{dt} = -\dot{m}_{out}, \quad (5.1)$$

$$\frac{d}{dt}(Me) = -\dot{m}_{out}h_{out} + q_I'' A_S, \quad (5.2)$$

$$\frac{dV}{dt} = 0 = \frac{d}{dt}(Mv). \quad (5.3)$$

where:  $M$  is the gas mass,  $\dot{m}_{out}$  is the mass flow rate out the break,  $e$  is the gas specific internal energy,  $h$  is the gas specific enthalpy with the subscript *out* denoting the enthalpy at the break location,  $q_I''$  is the heat flux through the internal walls of the vessel,  $A_S$  is the vessel internal surface area, and  $V$  is the vessel internal volume while  $v$  is the gas specific volume. The governing equations assume a lumped average response of the gas within the tank. The lumped energy balance equation, in Eq. 5.2, neglects the kinetic and potential energy terms of the lumped average gas within the tank. The energy balance is therefore simply between the energy lost out the valve (the break location) and in addition to the governing equations, the following state relationships are required:

$$Pv = RT, \quad (5.4)$$

$$e = c_v T \quad h = c_p T. \quad (5.5)$$

Above,  $R$  is the specific gas constant, which is related to the specific heats by:  $R = c_p - c_v$ . Equations 5.1-5.5 can be re-written in terms of the two gas state



variables, the pressure  $P(t)$  and temperature  $T(t)$ :

$$\frac{dP}{dt} = \frac{1}{V} \left[ (\gamma - 1) q_I'' A_S - \gamma \dot{m}_{out} R T \right]. \quad (5.6)$$

$$\frac{dT}{dt} = \left( \frac{\gamma - 1}{V} \right) \left[ \frac{q_I'' A_S T}{P} - \frac{\dot{m}_{out} R T^2}{P} \right]. \quad (5.7)$$

In Eq. 5.6 and 5.7  $\gamma$  is the ratio of the specific heats  $\gamma = c_p/c_v$  for the gas.

The temperature distribution through the vessel wall,  $T_w(x, t)$ , is given by the transient heat conduction equation:

$$\frac{\partial}{\partial t} T_w(x, t) = \alpha_w \frac{\partial^2}{\partial x^2} T_w(x, t). \quad (5.8)$$

### 5.1.2 Boundary and Initial Conditions

The vessel internal,  $x = 0$ , and external wall,  $x = S$ , boundary conditions (BCs) are:

$$\left[ -k_w \frac{\partial T_w}{\partial x} \right]_{x=0} = h_N (T - T_{w,I}) = q_I''. \quad (5.9)$$

$$\left[ k_w \frac{\partial T_w}{\partial x} \right]_{x=S} = h_A (T_A - T_{w,E}) = q_E''. \quad (5.10)$$

Note that  $h_N$  and  $h_A$  denote the heat transfer coefficients between the vessel wall and the internal gas (nitrogen) and external air, respectively.

The initial conditions for the gas pressure and temperature are:

$$P(t = 0) = P_0 \quad T(t = 0) = T_0.$$

The initial temperature distribution through the vessel wall is:

$$T_w(x, t = 0) = T_0, \quad 0 \leq x \leq S.$$

The air outside the vessel is held at a fixed pressure and temperature throughout the transient:

$$P_{air}(t) = P_A \quad T_{air}(t) = T_A.$$

### 5.1.3 Breakflow

The mass flow rate out the break,  $\dot{m}_{out} = \dot{m}_{break}$ , is determined by the following correlation:

$$\dot{m}_{out}(P, T) = \begin{cases} \dot{m}_{choke}(P, T), & r(P) \leq r_c \\ \dot{m}_{choke}(P, T) \frac{\sqrt{r^{2/\gamma} - r^{\gamma+1/\gamma}}}{\sqrt{r_c^{2/\gamma} - r_c^{\gamma+1/\gamma}}}, & r_c < r(P) < 1 \\ 0, & r(P) \geq 1 \end{cases} \quad (5.11)$$

In Eq. 5.11  $r(P)$  is the ratio of the air pressure to the internal gas (nitrogen) pressure:

$$r(P) = \frac{P_A}{P(t)}.$$

The critical pressure ratio,  $r_c$ , at which choked flow starts, for an ideal gas is set by  $\gamma$  and is given by:

$$r_c = \frac{P_c}{P(t)} = \left( \frac{2}{\gamma + 1} \right)^{\gamma/(\gamma-1)} \quad (5.12)$$

with the critical pressure value given as:

$$P_c = P(t) \left( \frac{2}{\gamma + 1} \right)^{\gamma/(\gamma-1)} \quad (5.13)$$

Including an orifice discharge coefficient,  $C_d$ , the choked mass flow rate for an ideal gas is given by:

$$\dot{m}_{choke}(P, T) = C_d A \sqrt{\gamma \frac{P(t)^2}{RT(t)} \left( \frac{2}{\gamma + 1} \right)^{(\gamma+1)/(\gamma-1)}} \quad (5.14)$$

#### 5.1.4 Heat Transfer Coefficients

The heat transfer coefficients,  $h_N$  and  $h_A$  are consider only the heat transfer through natural convection between the gas and the vessel wall. The correlation is then based on the product of the Grashof ( $Gr$ ) and Prandtl ( $Pr$ ) numbers. The Grashof number is:

$$Gr_N = \frac{g\beta_N\rho_N^2}{\mu_N^2} H^3 (T_w(x=0) - T). \quad (5.15)$$

The Grashof number is defined between the gas temperature,  $T$ , and the vessel inner wall temperature  $T_w(x=0)$ . The Prandtl number is:

$$Pr_N = \frac{c_{pN}\mu_N}{k_N}. \quad (5.16)$$

The heat transfer coefficient for the internal vessel wall is given by:

$$h_N = \frac{k_N \overline{Nu}_N}{H}. \quad (5.17)$$

where  $H$  denotes the vessel height. The correlation for the Nusselt number ( $\overline{Nu}$ ) depends on whether the flow is laminar or turbulent and is given by:

$$\overline{Nu}_N = \begin{cases} a [Gr_N Pr_N]^b, & Gr_N Pr_N \leq 10^9 \\ c [Gr_N Pr_N]^d, & Gr_N Pr_N > 10^9 \end{cases} \quad (5.18)$$

The heat transfer coefficient for the vessel external wall,  $h_A$  is determined similarly.

### 5.1.5 Physical Properties

To be consistent with Petruzzi's solution procedure, look-up tables were created for the gas properties of nitrogen and air. The nitrogen property look-up table using double linear interpolation of the gas temperature and pressure while the air property look-up table only uses linear interpolation of the air temperature, since the outside air pressure is assumed to be roughly constant for this problem. The temperature values actually used in the look-up tables though are the average between the respective gas temperature and the wall temperature in contact with that gas. The gas properties in the look-up table are: the gas density  $\rho_i$ , the viscosity  $\mu_i$ , the thermal conductivity  $k_i$ , the specific heat  $c_{p,i}$ , and the thermal expansion coefficient  $\beta_i$ , where the subscript  $i$  is either  $N$  or  $A$ .

The required vessel physical properties, the thermal conductivity  $k_w$  and thermal diffusivity  $\alpha_w$ , are assumed to be constant with respect to the vessel wall temperature.

## 5.2 Numerical Scheme

### 5.2.1 Spatial Discretization

The vessel heat conduction equation, Eq. 5.8 was discretized using  $(J - 1)$  uniform intervals of width  $\Delta x = s/J$ . To handle the the boundary conditions, two additional intervals of thickness  $\Delta x/2$  were placed adjacent to each boundary location, denoted as mesh point 0 and  $J$ , respectively. The finite difference approximation to the spatial

second derivative in Eq. 5.8 at mesh point  $j$  is then:

$$\frac{\partial^2}{\partial x^2} T_w(x, t) = \frac{\partial^2}{\partial x^2} T_{w,j}(t) \approx \frac{T_{w,j+1}(t) - 2T_{w,j}(t) + T_{w,j-1}(t)}{(\Delta x)^2}. \quad (5.19)$$

Substituting Eq. 5.19 into Eq. 5.8, and applying the boundary conditions, ultimately gives a total of  $(J + 3)$  ordinary differential equations to be solved.

## 5.2.2 Time Discretization

Following Cacuci, the time discretization is performed in two steps. The first is a Forward Euler step for the gas pressure and temperature equations, Eq. 5.6 and Eq. 5.7, respectively. The second is a Backward Euler step for the vessel wall temperature equations. At the current timestep  $n$ , the current solution values are used to determine the current heat transfer coefficients,  $h_N^n$  and  $h_A^n$ , and breakflow rate,  $\dot{m}_{out}^n$ . These are then used to compute the solutions at the  $(n + 1)$  timestep for the gas pressure and temperature. Substituting in the Forward Euler approximation into Eq. 5.6 and 5.7 gives the discrete time equations to be:

$$P^{n+1} = P^n + \frac{\Delta t}{V} \{(\gamma - 1) h_N^n (T_{w,I}^n - T^n) A_S - \gamma \dot{m}_{out}^n R T^n\}. \quad (5.20)$$

$$T^{n+1} = T^n + \Delta t \left( \frac{\gamma - 1}{V} \right) \left\{ [h_N^n (T_{w,I}^n - T^n) A_S] \frac{T^n}{P^n} - \dot{m}_{out}^n \frac{R (T^n)^2}{P^n} \right\}. \quad (5.21)$$

The gas pressure and temperature values for the  $(n + 1)$  timestep are then passed into the boundary conditions for the vessel wall temperature equations Backward Euler step. Written in matrix form, the vector of vessel wall temperatures,  $\mathbf{T}_w =$

$(T_{w,0}, T_{w,1}, \dots, T_{w,J-1}, T_{w,J})^T$ , at the  $(n + 1)$  timestep are:

$$\mathbf{T}_w^{n+1} = \mathbf{C}^{-1}\mathbf{D}. \quad (5.22)$$

where the matrix  $\mathbf{C}$  and vector  $\mathbf{D}$  are given by:

$$\mathbf{C} = \begin{bmatrix} (1 + 2Fo + 2FoBi_N^n) & -2Fo & 0 & \dots & 0 \\ -Fo & (1 + 2Fo) & -Fo & \dots & 0 \\ \vdots & & \ddots & & \vdots \\ 0 & \dots & -Fo & (1 + 2Fo) & -Fo \\ 0 & \dots & 0 & -2Fo & (1 + 2Fo + 2FoBi_A^n) \end{bmatrix} \quad (5.23)$$

$$\mathbf{D} = \begin{bmatrix} (T_{w,0}^n + 2FoBi_N^n T^{n+1}) \\ T_{w,1}^n \\ \vdots \\ T_{w,J-1}^n \\ (T_{w,J}^n + 2FoBi_A^n T^{n+1}) \end{bmatrix} \quad (5.24)$$

In Eq. 5.23 and 5.24 the first and last rows are modified due to the boundary conditions, while the interior mesh points are the conventional tri-diagonal matrix setup. Also,  $Fo$  is the mesh Fourier number defined as:

$$Fo = \alpha_w \frac{\Delta t}{(\Delta x)^2}. \quad (5.25)$$

and  $Bi_N$  and  $Bi_A$  are the Biot numbers numbers for the interior vessel wall in contact with the nitrogen and the exterior vessel wall in contact with the air, respectively.

The Biot number is defined to be:

$$Bi = h \frac{\Delta x}{k_w}. \quad (5.26)$$

## 5.3 QPIRT Formulation

The FOM for this problem is the gas temperature. The governing equations are used to determine the relative importance between the various physical processes, in the Top-Down step. Various hierarchical length scales are used. The local-level examines the governing equations directly impacting the FOM, by comparing the weighted contribution of each physical process onto the value of the FOM. The system-level captures the influence of the entire system's dynamic response, by computing the fractional contribution of each process relative to the sum of all the processes. The Bottom-Up step then examines the dominant physical processes, to determine which parameters to include as the parameters of interest for the surrogate model construction.

### 5.3.1 Top-Down Equations

#### Gas Temperature

For the benchmark blowdown problem, however, the FOM is a lumped-parameter quantity. Thus, the local and system level hierarchical scales are the same. Therefore a slightly modified QPIRT is used as compared to that put forward in Chap 3. Fractional contributions are computed for each of the processes in Eq. 5.7. These

processes are easier to identify when Eq. 5.7 is re-written as:

$$\left(\frac{PV}{\gamma-1}\right) \frac{dT}{dt} + \dot{m}_{out}RT^2 - h_N(T_{w,I} - T)A_S T = 0. \quad (5.27)$$

The three physical processes of interest are then: the thermal capacitance of the gas (the first term on the LHS of Eq. 5.27), the break flow energy term, and the heat transfer rate with the vessel inner wall. The fractional contributions for each process are then defined as:

$$\mathcal{F}_{m1} = \frac{\left|\left(\frac{PV}{\gamma-1}\right) \frac{dT}{dt}\right|}{\mathcal{D}_m}, \quad (5.28)$$

$$\mathcal{F}_{m2} = \frac{|\dot{m}_{out}RT^2|}{\mathcal{D}_m}, \quad (5.29)$$

$$\mathcal{F}_{m3} = \frac{|h_N(T_{w,I} - T)A_S T|}{\mathcal{D}_m}, \quad (5.30)$$

where the denominator term  $\mathcal{D}_m$  is simply the sum of the numerators of each of the processes:

$$\mathcal{D}_m = \left|\left(\frac{PV}{\gamma-1}\right) \frac{dT}{dt}\right| + |h_N(T_{w,I} - T)A_S T| + |\dot{m}_{out}RT^2|. \quad (5.31)$$

Equations 5.28 - 5.31 show how simple the Top-Down QPIRT is to setup, especially for this problem. Each fractional contribution will take a value between 0 and 1 and the process is considered to be significant if the fractional contribution is at least 0.1.



## Vessel Inner Wall Temperature

If the heat transfer rate between the gas and vessel inner wall is significant, the processes impacting the vessel inner temperature,  $T_{w,I}$ , must also be examined. This follows the local-level formulation from Chap. 3 for the clad wall inner temperature. The vessel wall energy equation is simply a balance between the heat transferred from the air to the vessel and the heat transferred between the vessel and the nitrogen gas:

$$\frac{d}{dt} [V_w \langle \rho c T \rangle_w] = -h_N (T_{w,I} - T) A_S + h_A (T_A - T_{w,E}) A_S. \quad (5.32)$$

The brackets in Eq. 5.32 denote the average value with respect to the vessel wall. Solving Eq. 5.32 for  $T_{w,I}$  scales each of the processes by the heat transfer coefficient with nitrogen:

$$T_{w,I} = -\frac{1}{h_N A_S} \frac{d}{dt} [V_w \langle \rho c T \rangle_w] + T + \frac{h_A}{h_N} (T_A - T_{w,E}). \quad (5.33)$$

The weighted contribution for each process then scales each term on the RHS of Eq. 5.33 by the sum of their absolute values. The weighted contribution terms then like the fractional contributions, take on values between 0 and 1. The three processes of interest are the thermal capacitance of the vessel wall, heat transfer with the nitrogen gas, and heat transfer with the air outside, and their respective weighted contributions are:

$$\mathcal{W}_{w1} = \frac{\left| \frac{1}{h_N A_S} \frac{d}{dt} [V_w \langle \rho c T \rangle_w] \right|}{\mathcal{D}_w}, \quad (5.34)$$

$$\mathcal{W}_{w2} = \frac{|T|}{\mathcal{D}_w}, \quad (5.35)$$

$$\mathcal{W}_{w3} = \frac{\left| \frac{h_A}{h_N} (T_A - T_{w,E}) \right|}{\mathcal{D}_w}, \quad (5.36)$$

where the denominator term is simply the sum of each of the numerators:

$$\mathcal{D}_w = \left| \frac{1}{h_N A_S} \frac{d}{dt} [V_w \langle \rho c T \rangle_w] \right| + |T| + \left| \frac{h_A}{h_N} (T_A - T_{w,E}) \right|. \quad (5.37)$$

### 5.3.2 Using the QPIRT

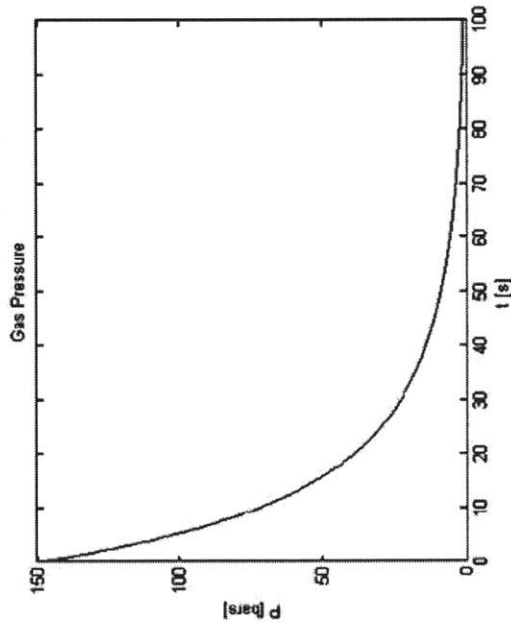
The results of the numerical solver are used to compute the fractional contributions for each of the three processes influencing the FOM, at each timestep. If a term is considered significant, then it is analyzed further through a local-level analysis and/or the Bottom-Up procedure. For this specific problem, the only term that would also have a local-level analysis performed is the vessel wall inner temperature, as described previously. This allows tracking through various processes which physical phenomena ultimately influence the FOM. The Bottom-Up step then looks at each process and examines the correlations and parameters used to compute each process term. For example, if the break flow energy term,  $\mathcal{F}_{m2}$ , is significant, then Eq. 5.11 is examined because it is the correlation that computes the break flow rate. If the heat transfer with the vessel wall is important, then besides performing the local-level analysis on  $T_{wI}$ , a Bottom-Up analysis is performed on Eq. 5.17 and Eq. 5.18 because that is how the heat transfer coefficient is computed. The end result of the QPIRT is a list of the physical processes and their associated parameters that influence the FOM through time.

## 5.4 Blowdown Model Results

### 5.4.1 Nominal Case Results

The nominal values for all of the parameters used in the numerical solver as given by [36] are summarized in 5.1. The time history of the computed gas pressure is shown in Fig. 5-4 and the gas temperature with inner and outer vessel wall temperatures are shown in Fig. 5-5. The same nominal case results, as determined by Petruzzi et al. (2010) are shown accompanying the current work's results. The present nominal results and Petruzzi's nominal results appear to be essentially the same. The gas pressure drops rapidly during the depressurization process essentially following an exponential trend. The gas temperature also drops rapidly early on, until leveling off and eventually rising due to heat transfer with the warmer vessel walls.

Current work nominal results



Petruzzi et al. results

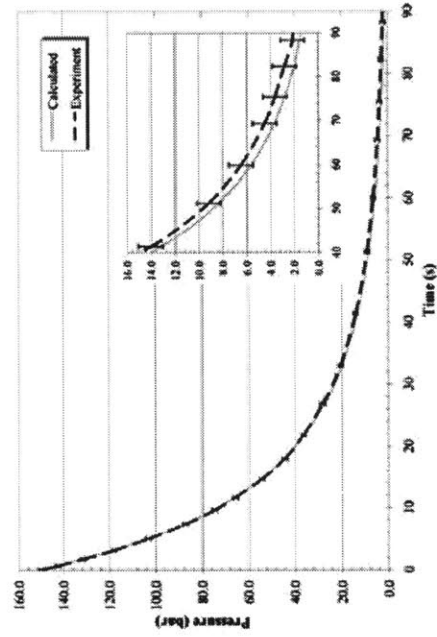
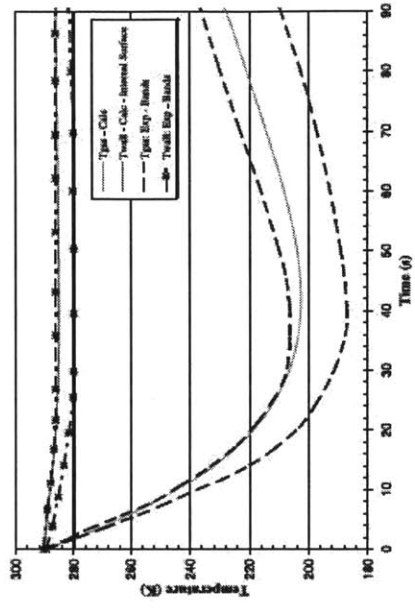


Figure 5-4: Nominal Gas Pressure Results

Petruzzi et al. results



Current work nominal results

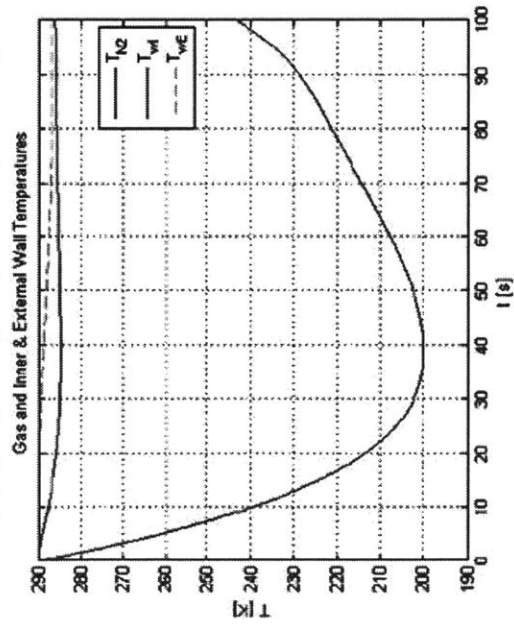


Figure 5-5: Nominal Temperature Results

## 5.4.2 QPIRT Results

### Top-Down Step

The plot of the fractional contributions for the three processes influencing the gas temperature is shown by Fig. 5-6. The fractional contribution results are in line with the physical intuition for the blowdown transient. While the gas temperature rapidly decreases, the break flow energy term and the gas thermal capacitance are the most important. Once the the gas temperature starts to level however, the heat transfer rate with the vessel wall becomes more and more important. After around 40 seconds, the heat transfer rate has a stronger influence on the gas temperature than the break flow energy term, which makes sense since the gas temperature is increasing. Then, around 85 seconds, the importance of the break flow energy term drops off dramatically, consistent with the gas pressure reaching nearly the ambient pressure. The gas temperature begins to rise at a faster rate now with the heat transfer rate becoming even more important at the end of the transient.

Since the heat transfer rate with the vessel wall is important, the local-level weighted contributions on  $T_{wI}$  are shown in Fig. 5-7. The heat transfer with the nitrogen dominates for the entire transient, but the thermal capacitance term is also significant.

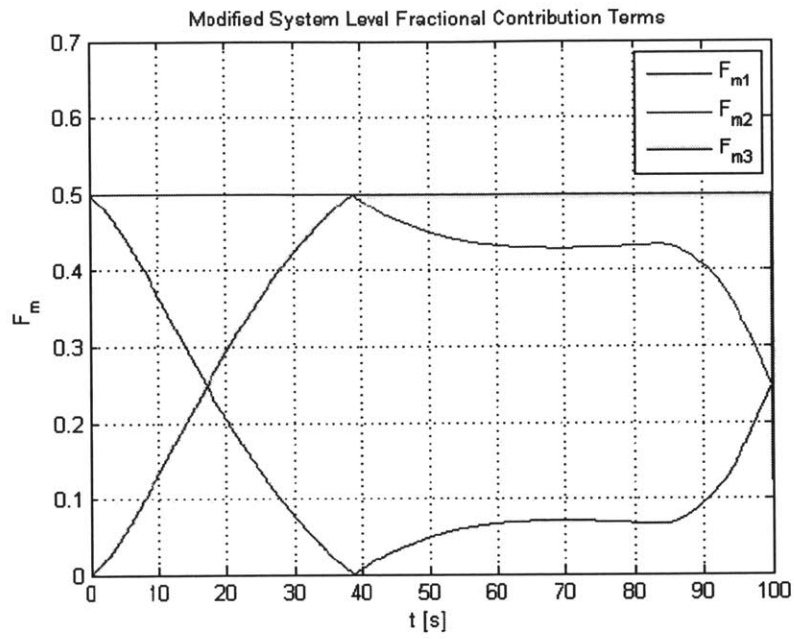


Figure 5-6: Fractional Contributions on the Gas Temperature

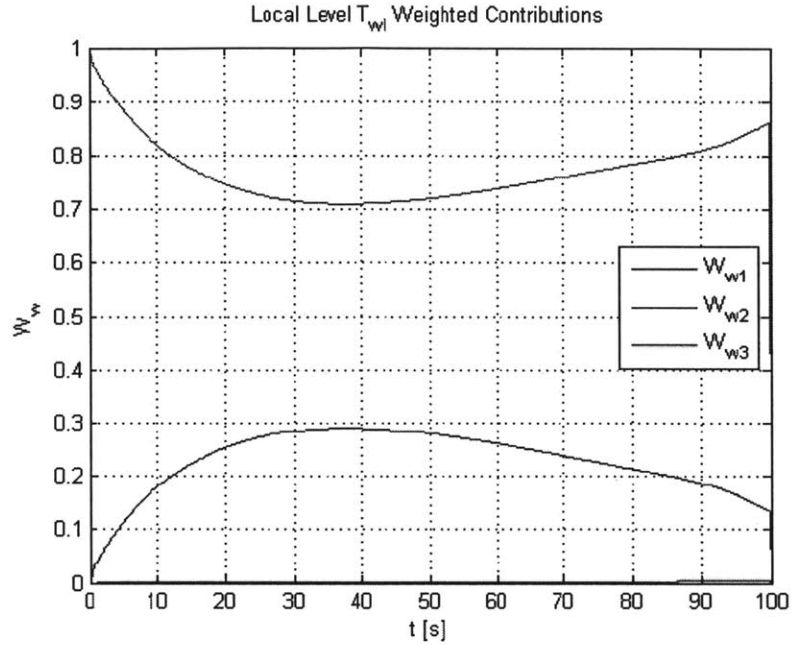


Figure 5-7: Local Level Weighted Contributions

### Bottom-Up Step

The Bottom-Up step examines each of the significant processes identified in the Top-Down step in order to choose which parameters to include in the GP emulator construction. Following Petruzzi et al. (2010), the parameters are all assumed to have normal distributions with the nominal values identified in Table 5.1 as the mean values. The standard deviations for each of the parameters are also given in [36]. For the present work, all parameters with standard deviations less than 0.5% of their mean values were disregarded. The geometrical terms, initial conditions, and the nitrogen density and specific heat state relationships all met this criterion and are therefore considered to be known values.



That leaves a total of eight parameters that are considered uncertain and will be included in the GP emulator construction. The uncertain parameter from the break flow energy term is the orifice discharge coefficient,  $C_d$ , which influences the break mass flow rate. The heat transfer between the vessel wall and the nitrogen has five uncertain parameters associated with it. The first two are the turbulent flow regime constant,  $c$ , and exponent,  $d$ , used to compute the Nusselt number in Eq. 5.18. The flow regime was always turbulent, so the laminar regime parameters were not important. The other three are the following nitrogen gas properties: thermal conductivity,  $k_N$ , viscosity,  $\mu_N$ , and the thermal expansion coefficient  $\beta_N$ . The final two parameters relate to the local-level analysis of the vessel wall temperature thermal capacitance term. The vessel wall volumetric heat capacity,  $(\rho c)_w$ , was computed from the vessel wall thermal conductivity,  $k_w$ , and thermal diffusivity,  $\alpha_w$ . Those eight parameters are summarized in Table 5.2 below, along with their associated uncertainty levels from Petruzzi et al (2010). These are the assumed prior distributions that are used in the emulator-based calibration process.

Table 5.2: Bottom-Up Step Uncertain Parameters

Parameter name	Parameter symbol	Uncertainty (%)
1	$C_d$	10
2	$c$	2
3	$d$	2
4	$k_N$	2
5	$\mu_N$	2
6	$\beta_N$	2
7	$k_w$	2
8	$\alpha_w$	2

## 5.5 Emulator-Based Calibration

### 5.5.1 Building the Emulators

A total of five emulators were used, a standard GPR model and four types of FFGP models. Each of the FFGP models have 2 factors, but with 1, 2, 3 and 4 components. For the standard GPR model, time is included as an input parameter giving a total of 9 inputs. The FFGP models treat the first factor as the time factor (or pattern) while the second factor is the uncertain parameter factor (or pattern). This setup is therefore the same as the setup in the simple friction factor verification problem in Chapter 4. Latin Hypercube Sampling (LHS) was used to generate 100 different values of the GPR input parameters, and the blowdown computer model was run 100 times. The time input corresponds to the point in time that output was taken at. The same 100 values of the uncertain parameters were used for the FFGP models, but now 15 equally spaced points in time were used as the input for the time factor. Thus, the GPR and FFGP model training data were generated from the same set of 100 case runs, the difference being the GPR model used only 1 point from each case run, a total of 100 training points, while the FFGP models “took” 15 points from each case run. It is important to note that the GPR model will be interpolating (or rather regressing) between those 100 training points in order to make a prediction, while the FFGP models use the 1500 training points to infer out patterns and those patterns are used to make predictions.

Both the GPR and FFGP training data are shown in Fig. 5-8. The GPR training data points are shown as black circles and the FFGP training points are in grey. The observational data are also shown in Fig. 5-8, as red dots with the error bars corresponds to the  $\pm 2\sigma$  around the data mean values. The observational data was

estimated from the Petruzzi et al. (2010) results shown in Fig. 5-5. As the figure shows, the training data set “encompasses” the observational data. If the observational data were outside the training data, then the training points were very poorly chosen and the prior distributions on the uncertain parameters must be reconsidered.

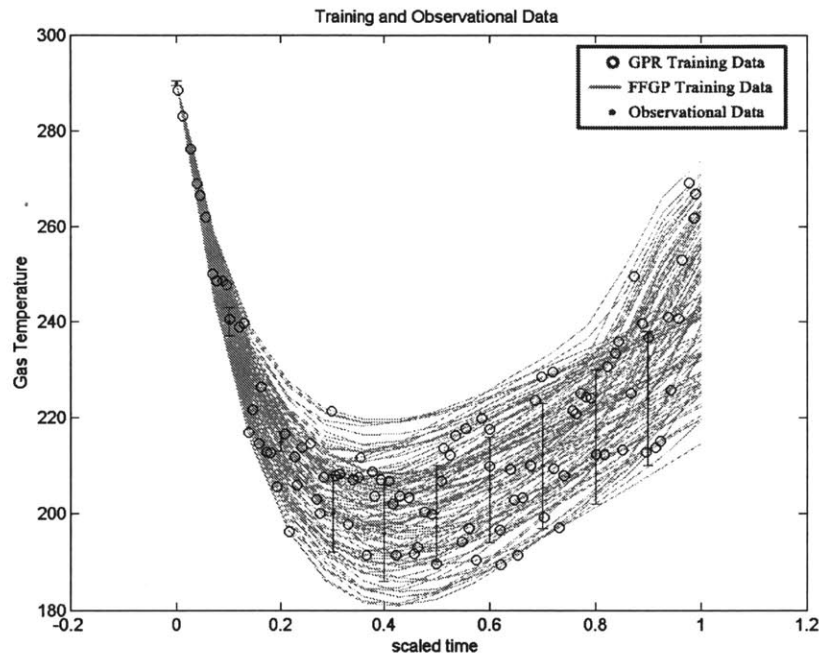


Figure 5-8: Blowdown Model Training Data

The GPR emulator was built using the MCMC sampling scheme described in Chapter 4. However, the hyperparameters were summarized as point estimates. This Empirical Bayes approach was only used to prevent MATLAB from reaching its RAM limit. Unless an incredibly large number of samples were used to train the emulator (such as 5 million samples or more), the RAM limit was never an issue for the blowdown problem. The Empirical Bayes approach was therefore used simply as

a precaution, and to be consistent with the GPR emulators used later on in Chapter 6. The GPR emulator likelihood noise hyperparameter is shown in Fig. 5-9 (along with the signal variance hyperparameter). The blue lines are samples and the red horizontal line is the point estimate. A total of  $1.5 \times 10^5$  samples were made, with the first  $5 \times 10^4$  samples discarded as burn-in. The length scale hyperparameters are shown in Fig. 5-10, where again the red lines denote the point estimates. As discussed in Chapter 4, the length scales indicate which of the inputs control the output response. Using the same notation from Chapter 4, the length scale for input  $d$  is  $l_d$  and its hyperparameter is  $\phi_d$ , and the two are related by  $l_d = \exp(\phi_d)$ . The closer  $l_d$  gets to 0, the more the output response is controlled by input  $d$ . A length scale value near zero corresponds to a very negative  $\phi_d$  value. Thus, the inputs can be ranked by how negative their length scale hyperparameter values are. Input 9, the time input, has the most negative  $\phi_d$  value and therefore dominates the output response. This makes sense since we are modeling the time series history of the gas temperature. The inputs whose  $\phi_d$  values are closer to 10 therefore have very little influence on the output response, as viewed by the GPR emulator. Therefore before even calibrating the blowdown model, the length scale hyperparameter values indicate that uncertain parameters 5 through 8 are not important to explaining the gas temperature response.

Each of the FFGP emulators were built using the “Gibbs-like” MCMC sampling procedure described in Chapter 4. The simplest way to compare FFGP emulator accuracy relative to the training data is to examine the likelihood noise hyperparameter value. The more negative the value, the more accurate the FFGP emulator is relative to the training data. Figure 5-11 shows the likelihood noise hyperparameter values for each of the four FFGP emulators. In each sub-plot, the grey line is the initial likelihood noise hyperparameter guess, the blue lines are the samples, and the

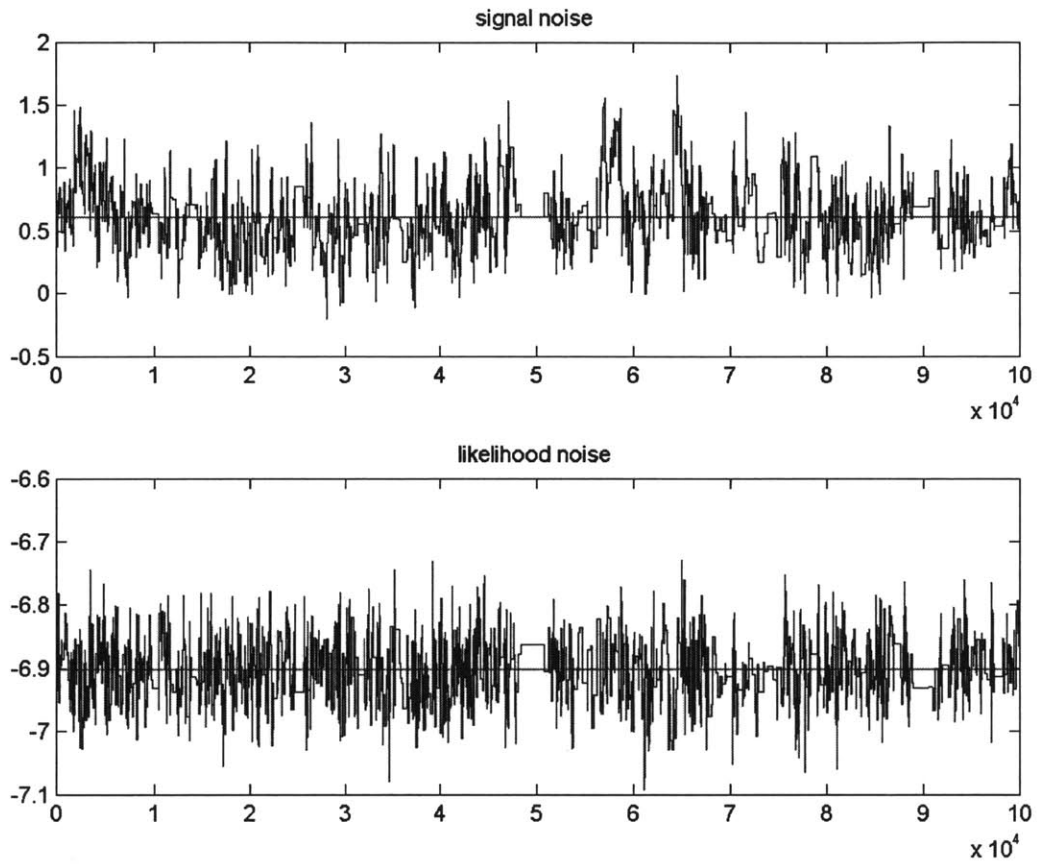


Figure 5-9: Blowdown GPR likelihood noise hyperparameter

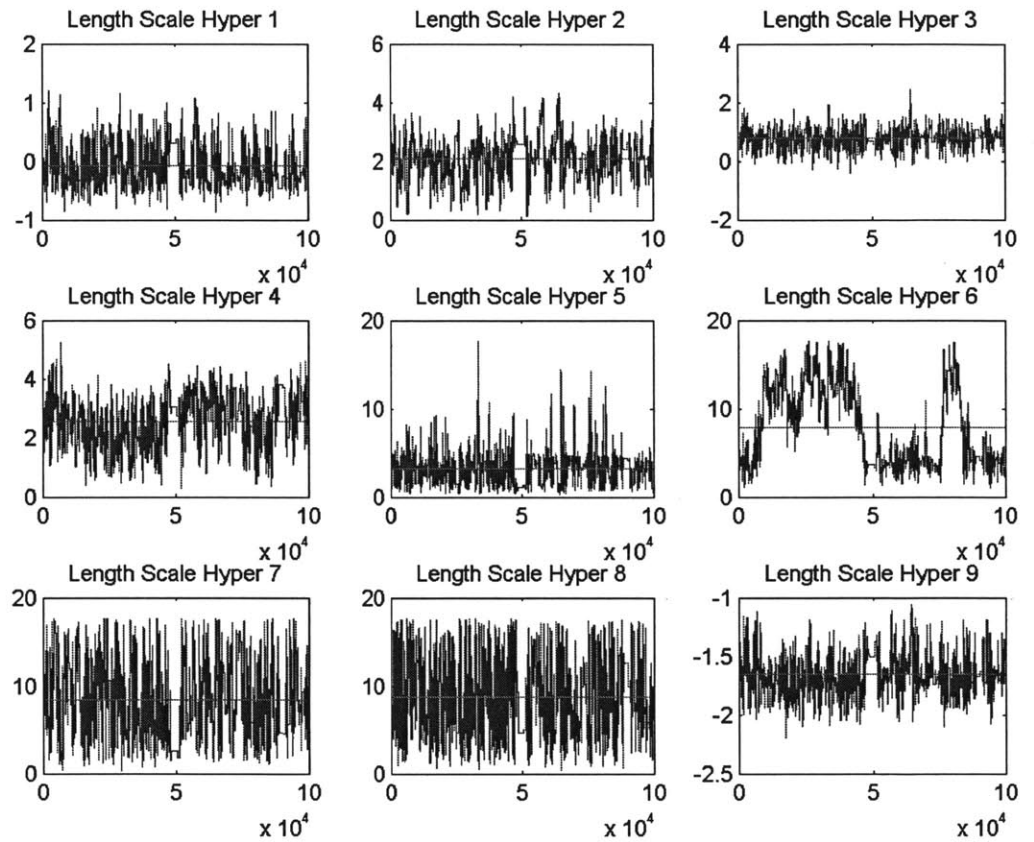


Figure 5-10: Blowdown GPR length scale hyperparameters

red line is the point estimate. Clearly, the 1 and 2-component emulators do not offer enough flexibility to accurately approximate the training data. The 3-component emulator is almost 50x more accurate relative to the training set than the 2-component emulator. The increased complexity between the 2 and 3-component emulators is clearly worth it. The 4-component emulator is then even more accurate than the 3-component emulator, though with a smaller relative change than the change between 2 and 3-components. A total of  $10^5$  latent burn-in samples were made, after which the latent variables were re-initialized at the mean values and sampling continued in the latent burn-in phase for another  $10^5$  samples.

The function factorization models are actually dealing with a much more complicated training data set than the standard GPR emulator training set. This is why the 1 and 2-component FFGP models are so innaccurate relative to the simulator. To understand why the data set is more complicated, just look at the grey lines in Fig. 5-8. The later half of the transient actually shows a substantial fraction of the trajectories that overlap. Remember that the QPIRT identified a change in the relationship between the physical phenomena, and thus the input parameters change their relationship with the output throughout the transient. This is known as non-stationary behavior in statistical models [24]. The FFGP models are using the stationary SE covariance function and thus more flexibility is required to capture these effects more accurately, which corresponds to adding complexity with more components. The standard GPR emulator, by only dealing with 100 training points actually does not “see” this complicated behavior. It is also using a stationary SE covariance function and using it to regress the training data, but with fewer data points it simply tries to average out any of that complicated behavior.

The computational time required to build each of the emulators is summarized in Table 5.3 below. The total time sums the time it takes to create the training set

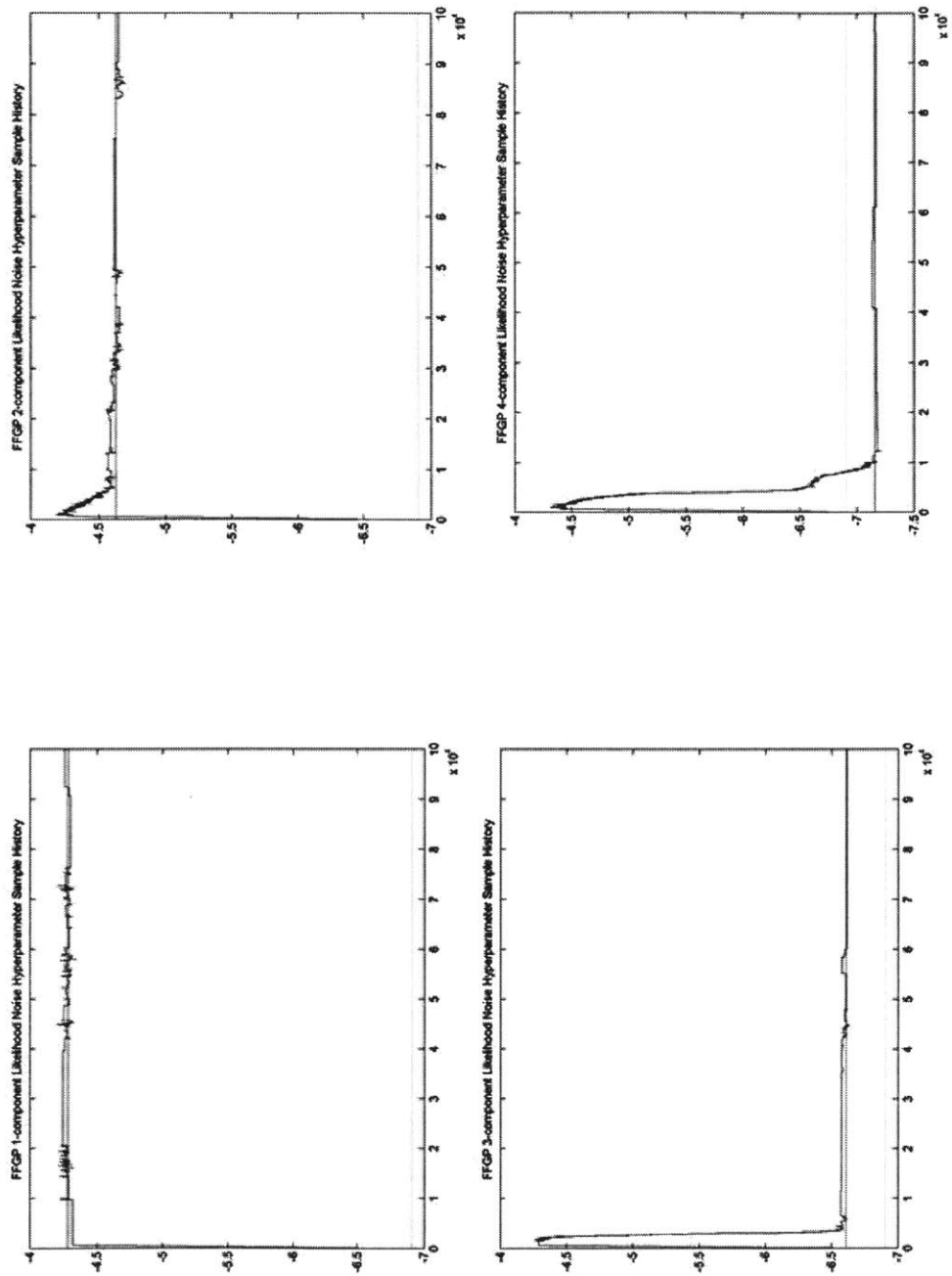


Figure 5-11: Blowdown FFGP likelihood noise hyperparameters



Table 5.3: Blowdown emulator build times

Emulator	Burn-in time [s]	Posterior time [s]	Total time [s]	Effective number of simulator runs
GPR	55.5	106.2	377.7	174
1-component	88.66	26.00	330.66	154
2-component	96.12	41.41	353.53	164
3-component	109.05	45.81	370.81	172
4-component	127.63	58.00	401.63	186

with the burn-in + posterior sampling times. Each case run took approximately 2.16 seconds, thus creating the training set took 216 seconds. The effective number of simulator runs corresponds to the number of times the simulator could have been run over the total emulator build time. The first key point from Table 5.3 is that building the most complex FFGP emulator adds less than 100 case runs worth of time on top of the time it takes to create the training set. This confirms that the even with the complex MCMC sampling required to build the FFGP emulators, the overhead due to training is small relative to the time to run the simulator. Each of the emulators could have been built in parallel, but for simplicity they were built in series, starting with the GPR emulator. The total time to build all of the emulators corresponds to effectively running the simulator 450 times. As described in Chapter 2, thousands upon thousands of simulator runs would be needed if the simulator was used directly to calibrate the uncertain parameters. Thus, even building each of the emulator sequentially, adds almost negligible time to the uncertain parameter calibration process relative to using the simulator directly.

## 5.5.2 Posterior Results

Emulator-based uncertain parameter calibration was performed just as in the Chapter 4 simple friction factor verification problem. The emulator-modified likelihood functions were used with the AM-MCMC scheme to draw samples from the posterior. Therefore, just as in the simple friction factor verification problem, the emulator's contribution to the total predictive uncertainty is accounted for. The posterior predictions are reported just as in Chapter 4, as well. The posterior predictive mean quantiles are all shown as blue lines and the total predictive uncertainty band is shown in green. If the emulator is very uncertain there will be a large gap between the outer blue lines (the 5<sup>th</sup> and 95<sup>th</sup> quantiles) and the edge of the green band. Ideally we want almost no difference between the outer blue lines and the edge of the green band.

The observational errors were all estimated from the figures in Petruzzi et al. (2010). A total of 9 observational data points were used because these were the 9 easiest to read off of the figures in their paper. As shown in the plot of the training data in Fig. 5-8, the last half of the transient has considerably more measurement error than the first half.

In the results to follow, the posterior predictive results will be on the left and the posterior uncertain parameter histograms will be on the right hand side plot. The posterior histograms are shown in blue and the prior histograms are shown in green. If the posterior histograms are essentially the same as the prior, then that particular parameter does not explain the data and therefore cannot be updated at all. The uncertain parameters are shown in their scaled values where 0 corresponds to the prior minimum bounding value and 1 is the prior maximum value. The scaled prior mean value is therefore 0.5 and all of the scaled prior standard deviations equal 0.25.

## Standard GPR emulator results

The posterior results using the standard GPR emulator in place of the blowdown simulator are shown in Fig. 5-12 below. The posterior predictive results follow the observational data while only the first and third uncertain parameters,  $C_d$  and  $d$ , respectively show any substantial change from their prior distributions. This means that really only  $C_d$ , the gas discharge coefficient which controls the break flow rate, and the heat transfer correlation exponent,  $d$ , are explained by the data. This is consistent with what we saw from the GPR length scale hyperparameters earlier. The GPR emulator also appears to be very accurate, with only a slight a difference between the posterior predictive mean quantiles and the total predictive uncertainty band near the end of the transient.

Looking at the posterior predictive results in more detail, we see that the first three and last five data points follow the posterior predictive median (which for a normal distribution is also the mean value) very well. But, the fourth and fifth data points lie on the outer “edge” of the posterior predictive probability distribution. Relating these results back to the Top-Down QPIRT findings, these two data points correspond to when the break flow rate and heat transfer rate are starting to balance out. The Top-Down QPIRT results given in Fig. 5-6, show that around this time, the physical phenomena start to change how they interact and relate to one another, indicated by the fact that the gas temperature starts to heat up afterwards. Over the first 30 seconds, the break flow is more dominant. After about 50 seconds, the physical phenomena are in nearly the same interactive relationship as shown by the QPIRT, which correspond to the last 5 data points. So it actually makes sense that the fourth and fifth data points are the most difficult to predict with high certainty compared to the other data points. Additionally, these two data points correspond to

when the observational error term increases rather significantly compared to the first three data points. Thus, it is not only harder to predict but more error is allowed in relating a prediction to the data. In order to better predict these two data points, more data would be necessary.

### **FFGP 1-component emulator results**

The posterior results using the FFGP 1-component emulator are shown in Fig. 5-13. With the posterior predictive mean quantiles taking up only a fraction of the total predictive uncertainty band, it is clear that The FFGP 1-component emulator is very uncertain. Therefore the emulator's uncertainty is capable of explaining most of the variation in the observational data. Even with the very uncertain FFGP 1-component emulator, the  $d$ -parameter posterior mode does move in a similar direction to the GPR-based posterior mode. However, the emulator predictive uncertainty is too large in the first half of the transient to allow for  $C_d$ -parameter to be updated.

### **FFGP 2-component emulator results**

The FFGP 2-component emulator-based results are shown in Fig. 5-14. Since the FFGP 2-component emulator is more accurate than the 1-component emulator, the total predictive uncertainty band is smaller in Fig. 5-14 than it was in Fig. 5-13. However, the emulator predictive uncertainty still dominates the total predictive uncertainty. The uncertain parameter posteriors are therefore similar to the 1-component emulator-based posteriors.

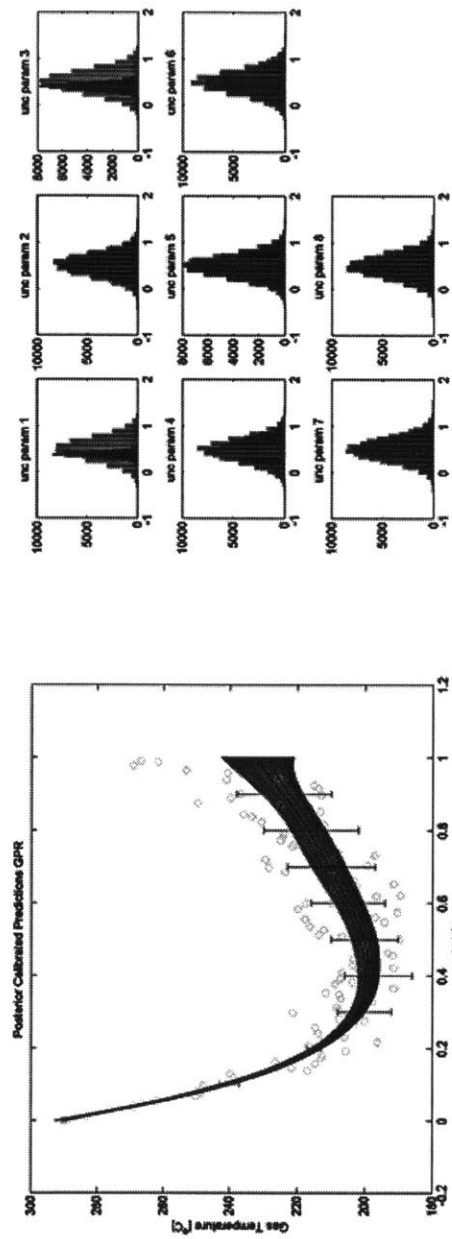


Figure 5-12: GPR Posterior Results

### FFGP 3-component emulator results

As shown in Fig. 5-11, the FFGP 3-component emulator is almost 50x more accurate than its 1 and 2-component counterparts. The 3-component emulator-based results reflect this, as shown in Fig. 5-15. The emulator contributes far less to the total predictive uncertainty, allowing not only the  $d$ -parameter to be more precisely known, but the  $C_d$ -parameter to be learned out as well. The first 10% of the transient, however, still has very large emulator predictive variance. This is an artifact of the Gaussian approximation to the training latent variable posterior samples. The additional uncertainty has minimal impact on the uncertain parameter posteriors however because the first data point is simply the initial condition, and it provides very little new “information” to the uncertain parameters. Later on in the transient, the emulator predictive uncertainty is much smaller, allowing the The first data point however is just the initial condition

The The FFGP 3-component-based results, in Fig. 5-15, are very similar to the GPR emulator results. The sample mixing rate is very high, and just as in the GPR case, only the  $C_d$  and  $d$ -parameters show any updating between the prior and posterior. The one minor difference in the posterior predictive probabilities is that the posterior trajectories appear slightly more smooth than the GPR results. What I mean is, if you look closely at the last half of the GPR posterior predictive trajectories, they exhibit a slight wavy shape pattern, which is absent in the FFGP 3-component posterior predictive trajectories. The FFGP model is more complicated and intended to capture the correlation amongst the outputs themselves, which is probably why that wavy-like pattern is absent.

### **FFGP 4-component emulator results**

The FFGP 4-component-based results are shown in Fig. 5-16. The emulator contributes very little additional uncertainty even at the beginning of the transient. The posterior predictive distribution is very similar to the GPR-based posterior predictive distribution. Only the fourth and fifth data points do not follow the posterior predictive medians. The uncertain parameter posterior distributions are also very similar to the GPR-based posterior distributions.

Both the FFGP 4-component-based and GPR-based total predictive uncertainties are less than the observational error. As stated earlier, the data was taken from reading values off of the figures in Petruzzi et al. (2010). Thus, one possible reason as to why this occurs is that the chosen data values were too “smooth”. The posterior predictive medians suggest the blowdown numerical model is capable of following these points correctly. If more scatter was present in the data, the predictive mean would only be able to regress the data, since the blowdown numerical model (and therefore the emulator) produces smooth time trajectories, not choppy trajectories. The total predictive uncertainty would then increase since many more possible temperature trajectories would regress the scatter in the data mean values. Another reason has to do with the simplicity of the numerical model itself.

### **5.5.3 Emulator Based Calibration Summary**

The blowdown problem was a very good demonstration of the pros/cons of the various emulator types and the challenges that each must overcome. The standard GPR emulator and the much more complicated FFGP 2-factor 4-component emulator yield very similar results. The fact that two very different emulators with very different underlying assumptions in their construction (though the GPR model is a special

case of the FFGP model) yield very nearly the same posterior results helps give confidence that these are the true posterior results.

The one minor difference between the GPR and FFGP 4-component posterior predictive distributions is that the FFGP emulator posterior trajectories appear slightly “smoother”. The last half of the GPR posterior predictive trajectories exhibit a slight wavy shape pattern, which is absent in the FFGP posterior predictive trajectories. The GPR emulator does not “know” the temperature at time  $t - 1$  when making a prediction at time  $t$ . It simply tries to interpolate (or rather regress) between all of the training points regardless of the “path history”. Due to the use of the two factors, the FFGP emulators are able to “know” the “history” of a time series. Factor 1, the time factor, captures the latent (or hidden) trend through time of the training set. Making predictions with the FFGP emulators therefore fully accounts for the correlation through time, and thus emulates the trajectory more precisely. As a whole though, the GPR emulator is a very good choice for this particular problem. Only two uncertain parameters, the gas discharge coefficient,  $C_d$ , and the heat transfer correlation exponent,  $d$ , could be updated by the observational data. The remaining six uncertain parameters could not be updated relative to their priors, and as shown by the GPR length scale hyperparameters, these do not control the gas temperature response as strongly as the  $C_d$  and  $d$ -parameters do. With 100 training points, the GPR emulator had more than enough training points to accurately approximate the response as these two parameters were varied during the MCMC sampling. As will be shown in Chapter 6 though, the assumptions inherent to the GPR emulator become more restrictive as the problem becomes more and more complex.

The blowdown problem also provided a more realistic application for where the speed of the emulator based approach is crucial to actually performing Bayesian model calibration. Table 5.4 summarizes the computational times required to cali-



brate the uncertain parameters with each of the emulator types. All of the emulator-based calibration processes used  $10^5$  MCMC samples, with the first half discarded as burn-in. The total emulator-based calibration time includes the time to create the training set, the time to build the specific emulator, and the time perform MCMC sampling with the specific emulator. The third column in Table 5.4 gives the total number of completed simulator (the blowdown numerical solver) runs that could be completed in the same time it takes to perform the entire emulator-based calibration process, for each emulator type. The last column, is the ratio of the time to complete a single emulator case run to the time it takes to make a prediction with each emulator. Table 5.4 makes it very clear that the emulators are all much faster than the numerical solver, with even the slowest FFGP emulator being over 1700x faster than the numerical solver. To clarify, a single emulator prediction denotes predicting the entire temperature trajectory, not simply a single point in time. The total time create the training set, to build all of the emulators (the training set only needs to be created once) and perform all of the emulator-based uncertain parameter calibration schemes was the equivalent of running the simulator 621 times. Thus, even though multiple emulators were built, and the 1 and 2-component FFGP emulators were very uncertain, having them as comparisons had almost negligible computational impact compared to using the simulator directly in the MCMC sampling. Running the simulator  $10^5$  MCMC samples would have taken 60 hours compared to the approximately 22 minutes it took to complete everything with the all of the emulators.

Table 5.4: Blowdown emulator-based calibration process computational times

Emulator type	Uncertain parameter calibration time [s]	Total emulator-based calibration time [s]	Effective number of simulator runs	Single simulator run to single emulator prediction ratio
GPR	67.30	445	206	3209.5
FFGP 1-comp	48.09	378.75	175	4491.6
FFGP 2-comp	60.02	413.55	191	3598.8
FFGP 3-comp	75.77	446.58	206	2850.7
FFGP 4-comp	121.25	522.88	242	1781.4

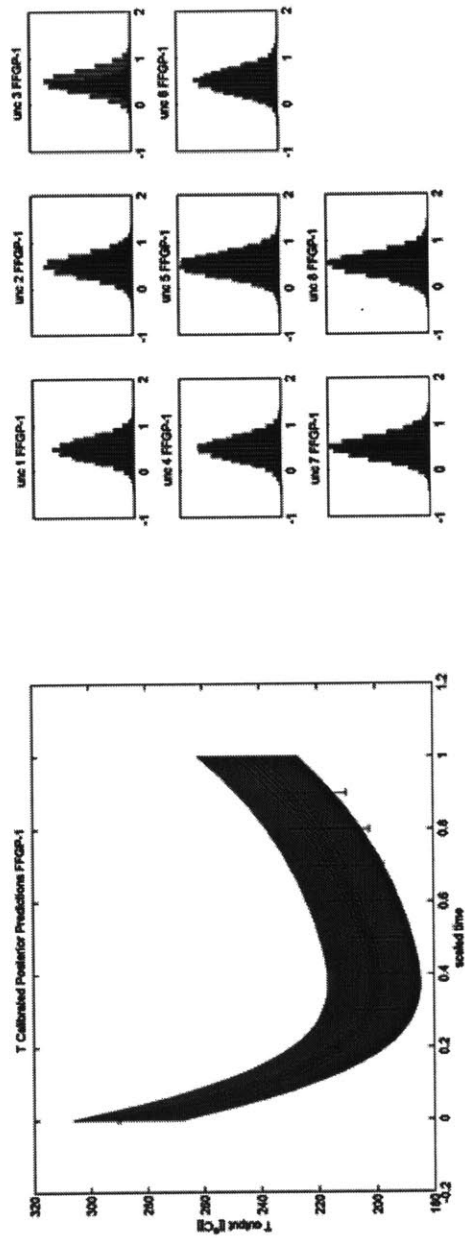


Figure 5-13: FFGP 1-component-based posterior results

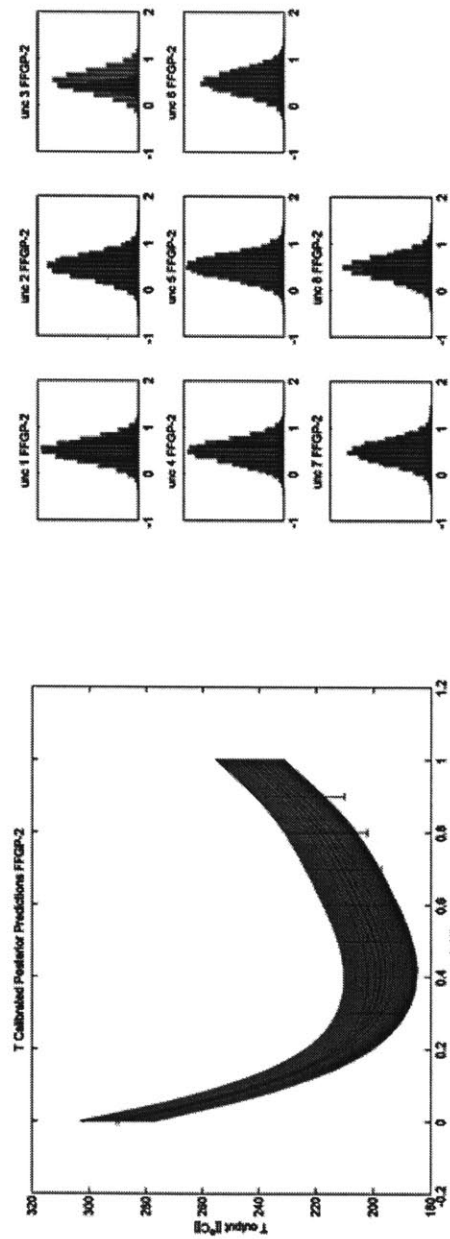


Figure 5-14: FFGP 2-component-based posterior results

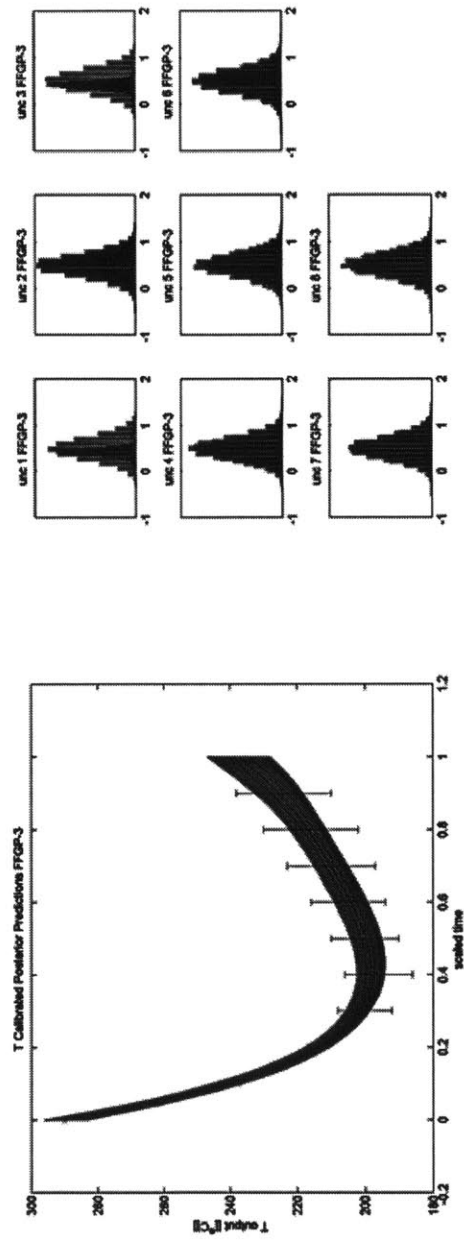


Figure 5-15: FFGP 3-component posterior-based results

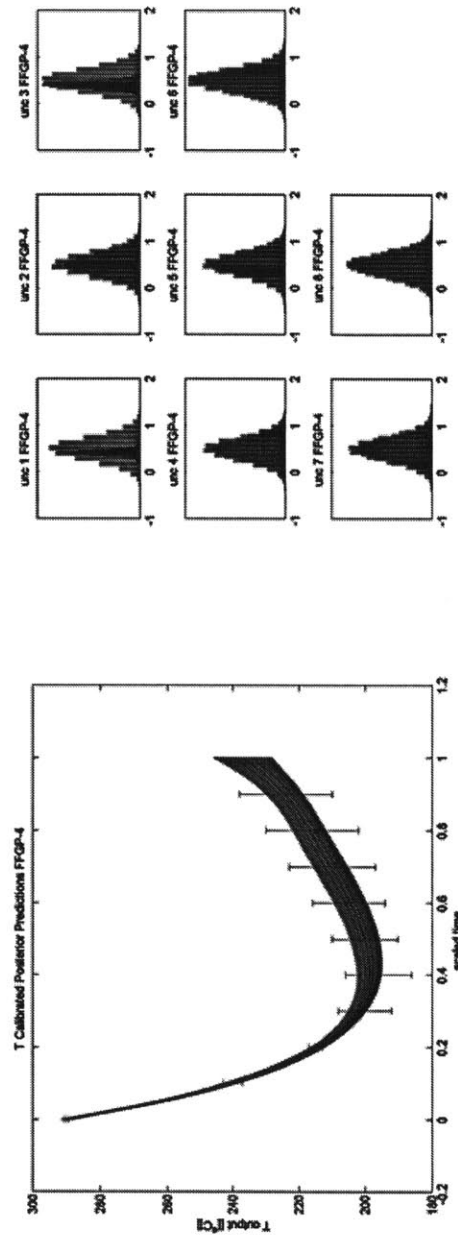


Figure 5-16: FFGP 4-component-based posterior results

## Chapter 6

# EBR-II Model Calibration

The overall goal is to demonstrate emulator-based calibration on a physically relevant situation with “realistic” modeling issues. The EBR-II simulation is performed using RELAP rather than my own numerical solver. This limits the amount of control I have over the uncertain parameters, since I am constrained by the RELAP input deck setup. Although somewhat of a minor point, it does impact certain choices in how the EBR-II model is setup and is not a trivial issue as will become more apparent later on. But more importantly, unlike the blowdown problem described in Chapter 5, the EBR-II model has a large number of uncertain parameters - 41 compared to only 8. These parameters may or may not be correlated with multiple physical phenomena coupled together. Additionally, Integral Effect Test (IET) data will be used in conjunction with Separate Effect Test (SET) to calibrate the uncertain parameters. The blowdown problem only used IET data - the gas blowdown data itself - to calibrate the uncertain parameters. With the large number of uncertain parameters in the EBR-II model, the SET data will help constrain some of the EBR-II uncertain parameters while the IET data influences all of the parameters.

Practically speaking, this means separate RELAP models must be constructed for each of the SET datasets as well as the IET data. From an error point of view, the posterior distributions of the uncertain parameters shared between the IET and SET models, consist of values that try to minimize the difference the data and emulator predictions for each of the models. The posterior distributions effectively regress the information from the IET and SET data, where the posterior distributions “satisfy” the IET data as constrained by the SET data. Uncertain parameters used only in the IET model are also impacted by the SET data since those shared parameters might be known more precisely (smaller posterior variance), which could drastically change the unshared parameter posterior distributions.

However, the QPIRT process, as described in Chapter 3, will not be applied to the EBR-II RELAP model. The primary reason is because as many user-defined functions were used as possible within the RELAP model. The exact correlations were therefore known before hand, and a QPIRT process was not needed to identify the important ones. Additionally no two-phase flow phenomena needed to be considered, since EBR-II uses single-phase liquid sodium as the primary coolant. That greatly reduced the number of required constitutive correlations and therefore uncertain parameters compared to a PWR model. Furthermore, since the RELAP model was built “from scratch” for this work, the modeler could decide to either lump various physical phenomena together, or split them up. The frictional form loss within the plenums, for example, could have been modeled by trying to model the various flow paths “exactly” or by simply lumping the friction form loss into a a few form loss coefficients. All together this greatly reduces the total number of uncertain parameters within the EBR-II RELAP model and we can afford to treat them all as uncertain.

The rest of the chapter is organized into the following sections. Section 6.1



provides a general description of the EBR-II facility and Section 6.2 describes the chosen IET and the data used in the IET. The RELAP model for the IET is then described in Section 6.3. The various SETs are described in Section 6.4, along with their associated emulator-based calibration results. The IET emulator based calibration results are given in Section 6.5. Simultaneous calibration is described in Section 6.6 with simultaneous results discussed after.

## **6.1 EBR-II**

### **6.1.1 Description of the Facility**

The Experimental Breeder Reactor II (EBR-II) was a sodium cooled fast-breeder-reactor plant. It generated about 20 MWe with a conventional steam cycle at a design power of 62.5 MWth [40]. It first achieved criticality in 1965 and operated for 30 years and served as a prototype to the Integral Fast Reactor (IFR) program. The EBR-II complex is shown in Fig. 6-1 below [41]. The EBR-II performed many important demonstration and validation experiments for sodium fast reactor (SFR) development including fuel validation, various thermal-hydraulic phenomena tests, and passive safety tests. The EBR-II testing program followed a “bootstrapping” approach which culminated in two historic tests on April 3, 1986: an unprotected (without scram) loss-of-flow from 100% power, and an unprotected loss-of-heat-sink at 100% power [42]. These two tests confirmed the inherent safety features of the pool-type SFR. Additionally, the experimental results have been used to validate many iterations of fast-reactor and sodium thermal-hydraulic computer codes for decades [43], [44], [45].

The EBR-II was a pool-type plant, with the reactor, primary coolant pumps

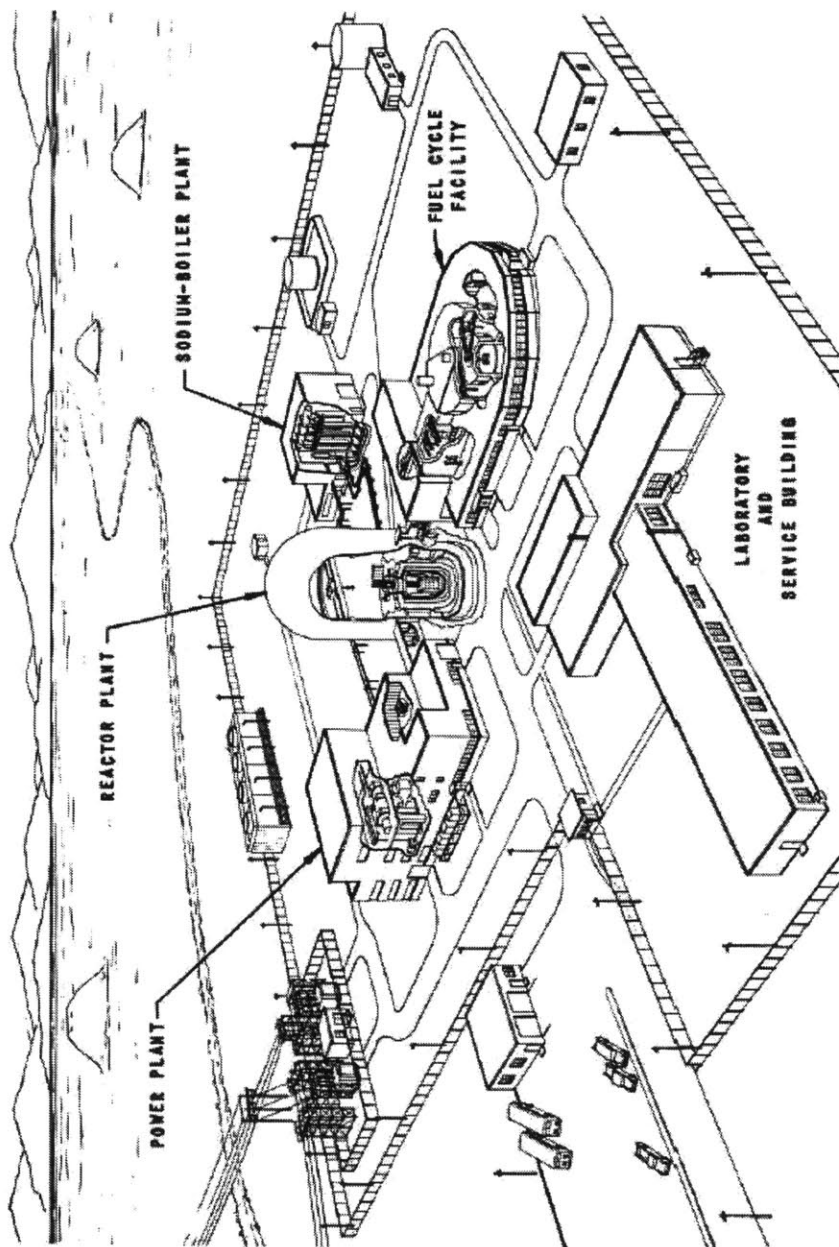


Figure 6-1: EBR-II complex

and intermediate heat exchanger (IHX) immersed in a large volume of sodium in the primary pool [40]. The nomenclature used for the EBR-II components depends somewhat on the reference. I will try to always use the terms used in the EBR-II Hazard Summary and Addendum to the Hazard Summary reports [41]. These documents were from the very early stages of the EBR-II program, the initial Hazard Report was published before construction began, but the Addendum was published after the initial construction was completed. So the core configuration described in the Addendum to the Hazard Report is actually different from the later references from the 1980s. But for consistency I use the Hazard Report terminology. Additionally, all of the figures showing schematics of the various components come from the Addendum to the Hazard Report. A schematic of the primary sodium system is shown in Fig. 6-2 [41]. The primary pumps are single-stage centrifugal pumps, each rated to produce 4500 gpm at 200 ft head. The pumps take a suction of sodium from the primary pool and discharge it to the high pressure plenum (HPP) via the high pressure pipe network and the low pressure plenum (LPP) via the low pressure pipe network. The flow split between the high and low pressure streams is controlled by the low pressure throttle valve. Roughly 84% of the primary flow feeds the HPP while the remaining 16% goes into the LPP. The HPP feeds the core (where the driver fuel is located) and inner blanket (IB) subassemblies while the LPP feeds the outer blanket (OB) subassemblies. The reactor consists of a total of 637 subassemblies, with 53 core, 12 control, 2 safety, 60 IB and 510 OB subassemblies. The subassembly types will be described in further detail later on. Figure 6-3 shows an "overhead" view of the reactor arrangement with the core subassemblies located in the center of the reactor. The sodium coolant exits the top of the various subassemblies where it all mixes in the upper (outlet) plenum before exiting the reactor through the "Z-pipe" to the IHX. A sideways view of the reactor vessel configuration is shown in Fig. 6-4.

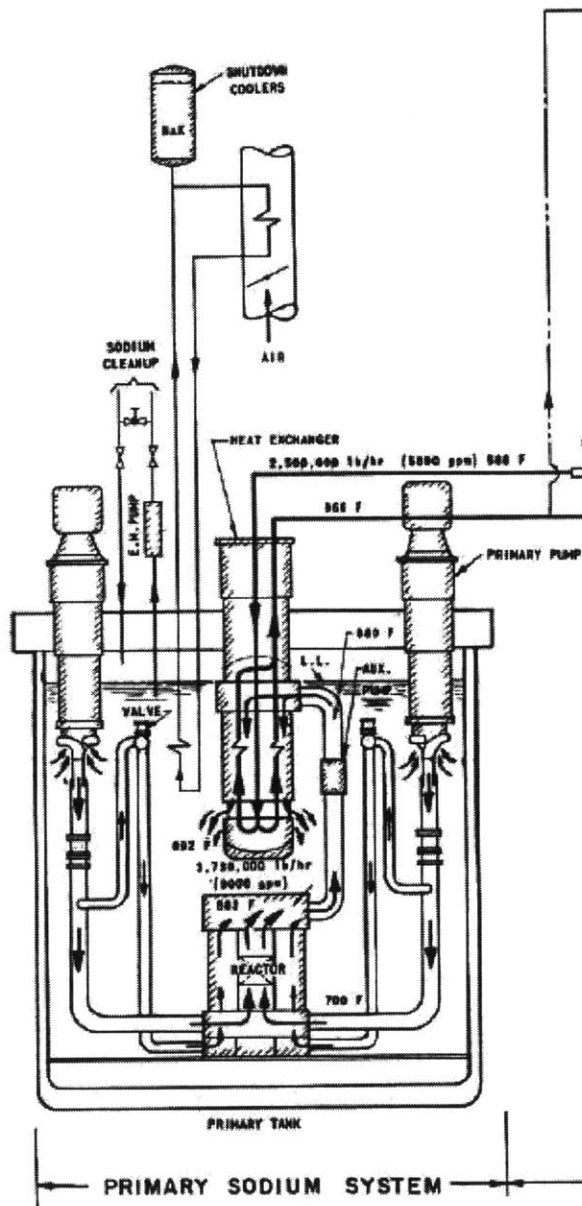


Figure 6-2: EBR-II Primary Sodium System

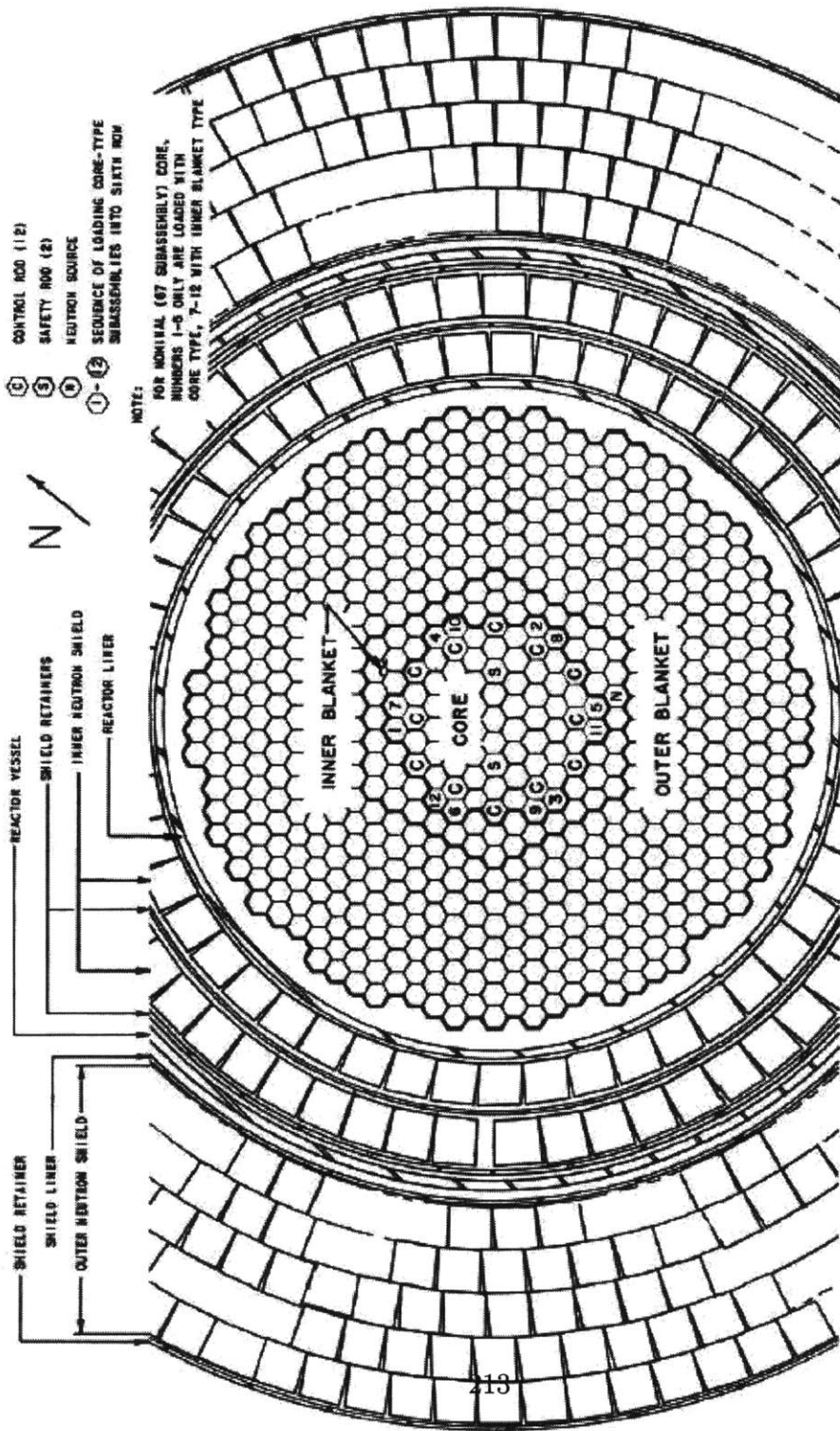
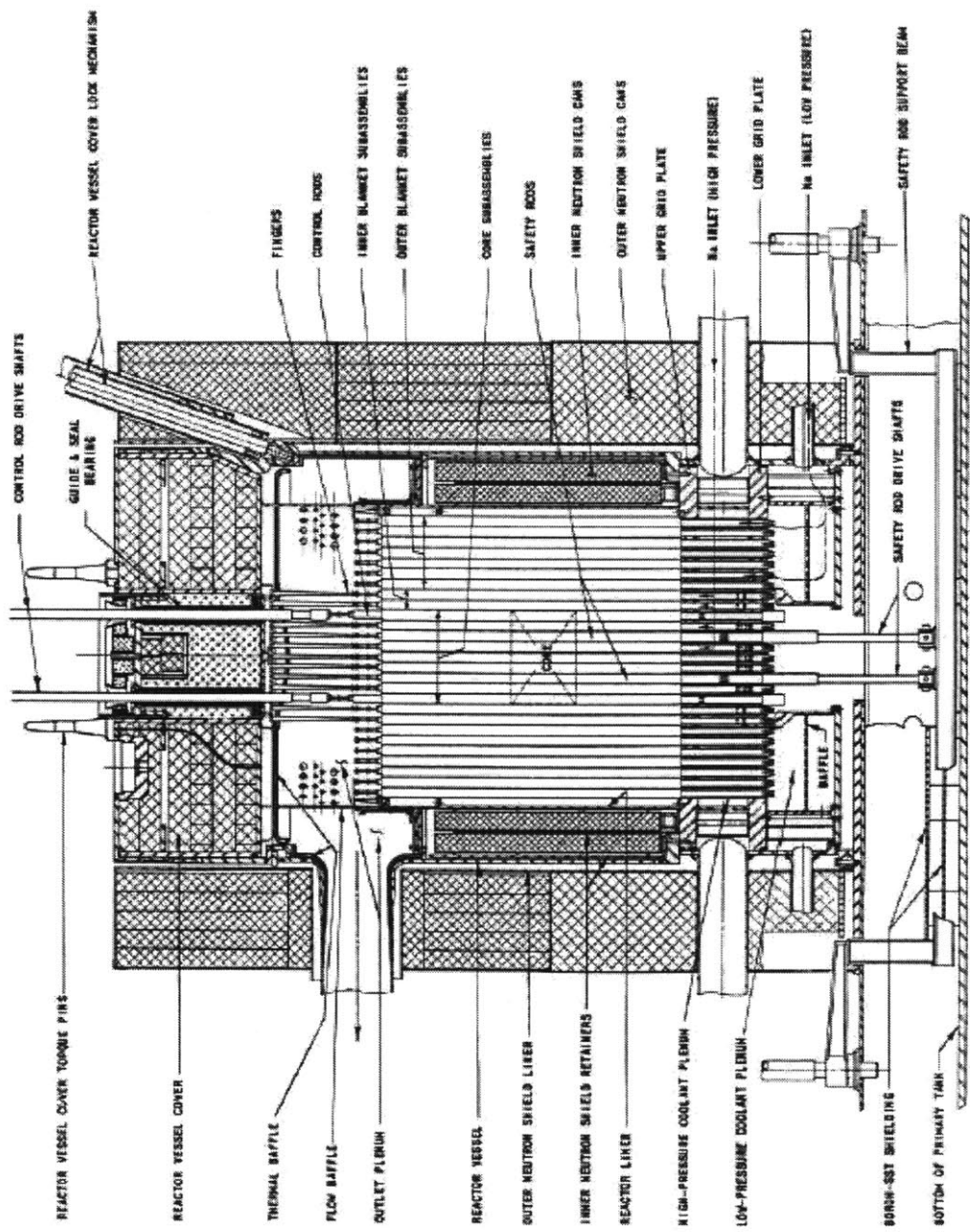


Figure 6-3: EBR-II reactor arrangement



NOTE: FOR PLAN VIEW SEE FIG. 3

Figure 6-4: EBR-II reactor vessel

The “Z-pipe” is a pipe that from the side literally looks like the letter “Z”. A safety-grade auxiliary electromagnetic pump in the Z-pipe provides about 5% flow to remove heat following reactor shutdown. The IHX is a counterflow tube-in-shell heat exchanger with the primary sodium on the shell side. Figure 6-5 shows the flow path from the primary pump inlets to the primary side IHX outlet from the view of the Z-pipe side of the primary pool. The secondary sodium removes the heat from the IHX and transports it to the steam-generator system. The secondary sodium pump is an electromagnetic pump rated at 6000 gpm. At full power, 32 kg/s of steam at 438°C and 8.70 MPa is delivered to the turbine.

### **6.1.2 History of Testing at EBR-II**

As already mentioned, the EBR-II testing program “bootstrapped” from test to test, starting with the mildest tests first in order to avoid any potential risks to the ongoing operation of the plant. The sequence was evolutionary in nature where the learning process would dictate new tests based upon answering new questions or resolving uncertainties from earlier tests. The EBR-II tests are commonly references by the name of the instrumented subassembly within the core. The XX07 series tests were the earliest, followed by the XX08 series of tests, and then the Shutdown Heat Removal Tests (SHRT) that used two instrumented subassemblies, XX09 and XX10. The XX07 tests evaluated the effects of the fission or decay power level and the secondary flowrate upon the natural convective core flowrate and temperature rise [42]. All but one of the XX07 tests were steady-state tests. The transient dynamics of the transition from forced to natural convective cooling were conducted in the XX08 series of tests that followed. These loss-of-flow tests were conducted from a variety of initial conditions including full primary flow at hot-standby conditions,

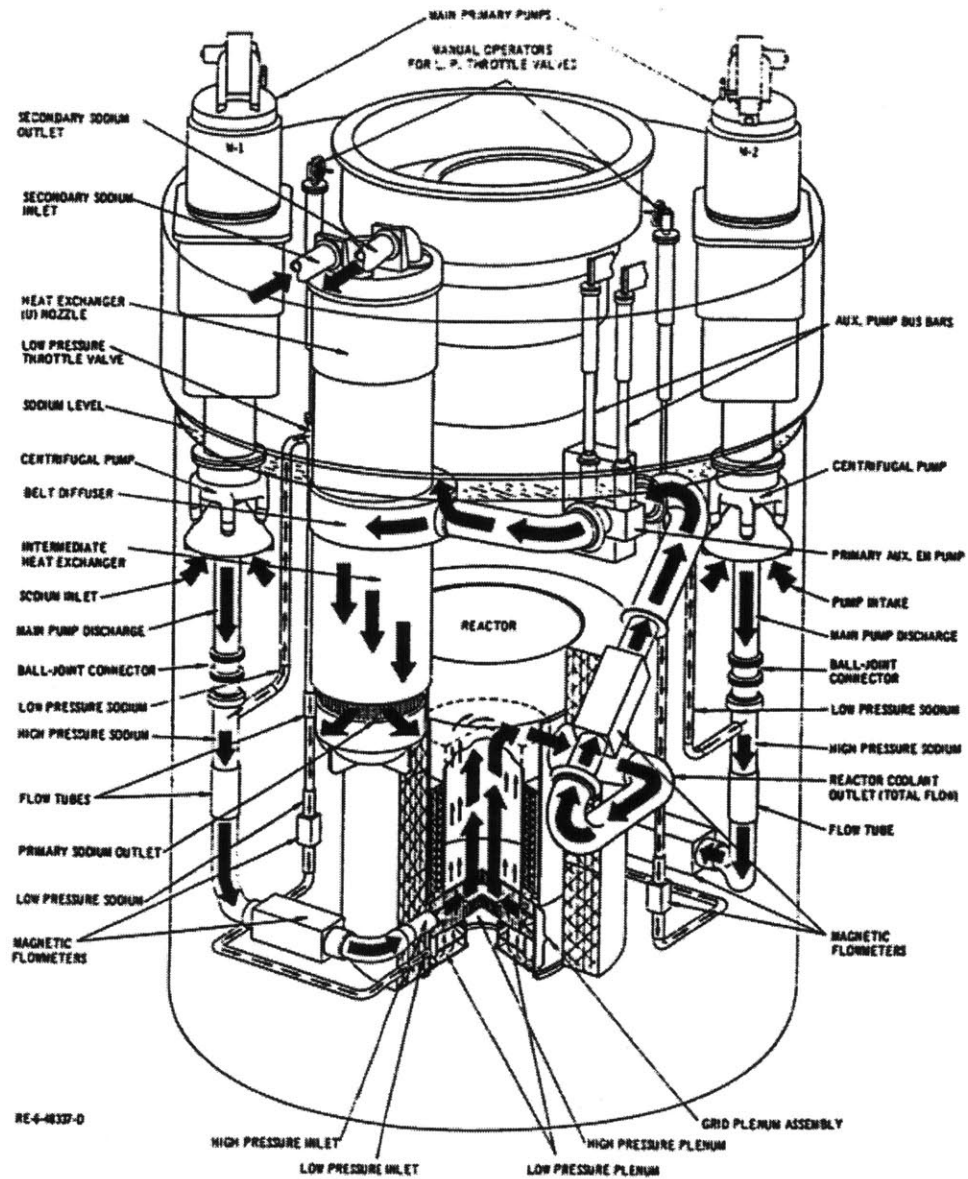


Figure 6-5: EBR-II primary pool Z-pipe view



auxiliary pump flow at decay power levels as well as other various combinations of reduced fission power and primary flow rates [42]. The XX08 tests essentially slowly built confidence that EBR-II could handle loss-of-flow tests at higher and higher initial power levels before the SHRT program was started. The SHRT program was the culmination of the testing to show the complete passive safety features of the EBR-II design in support of generic liquid metal pool type reactors.

## 6.2 Description of the chosen IET

In order to work with a “realistic” safety analysis modeling scenario, the IET of interest needed to be a transient, thus ruling out the XX07 series of tests. For this project, we wanted to focus strictly on thermal-hydraulic phenomena, not on any reactivity feedback or coupling issues to neutronics calculations. That ruled out the full power tests in the SHRT program, putting the focus on the XX08 series of tests at decay power levels. Another simplification was to try and find transients where the primary pumps would not have to be modeled in RELAP. Modeling pumps within RELAP can be very difficult because the pump homologous curves must be input into RELAP. Therefore, transition to natural circulation tests were the main candidates for tests to model. Looking through the available literature on the older XX08 series of tests at the lower power flow rates showed that most gave very little real data besides overall trends. One source however, by Baumann et al. gave a decent amount of data as well as provided a reference numerical model of the EBR-II reactor vessel to help give guidance on initial and boundary conditions [45]. They examined Transient Test No. 10 phase 2, which studied the transition from forced to natural convection at decay power levels, while examining the effect of flow reversal in the LPP. Flow reversal refers to the sodium flow within the OB channel

reversing from upward to downward during the transition from forced flow to natural convection. This test gave data therefore not just for the sodium temperature within the core but also the low pressure flow rate entering the LPP. So multiple types of data could be used from this transient which would be interesting to compare the posterior distribution results between using only temperatures compared to using only the low pressure flow rate data.

Additionally the other benefit of using Baumann et al., was it provided a discussion on the simulation of Transient Test No. 10 using the COMMIX-1A computer code. The numerical model was a 2-D axi-symmetric simulation of the test primarily focused on examining the flow reversal phenomena within the reactor vessel. So this is a completely different computer code style than RELAP which is a 1-D control volume code. Baumann et al. treated the total primary flow rate as a boundary condition into the reactor vessel, which meant the primary pumps are not modeled. The numerical model was responsible for splitting the total primary flow between the high and low pressure streams correctly, but the total flow rate itself was just a boundary condition. Of course this is a major approximation since the total flow rate, once the primary pumps are tripped, is a result of the buoyancy forces between the core, IHX, and primary pool. Not having to model the primary pumps within RELAP greatly simplified the model setup, without detracting of the goal of a "realistic" safety analysis scenario since as will be seen later on there are still a large number of uncertain parameters within this IET model. For these reasons I decided to follow the format of the Baumann et al. model by modeling just the primary flow loop from the primary pump inlet to the upper (outlet) plenum of the reactor vessel.

The Transient Test No. 10 observational data is shown in Figs. 6-6 through 6-8 compared against the COMMIX-1A predictions from Baumann et al. The term OTC refers to the core outlet temperature which is located just before the exit of the

instrumented XX08 subassembly. TTC refers to top-of-core temperature just before the exit of the driver fuel which as shown in 6-4 is in the middle of the reactor vessel. The OTC temperature therefore can include a temperature reduction compared to the TTC temperature due to the conduction within the sodium coolant, diffusing the temperature throughout the reactor. The normalized LPP flow is the low pressure flow rate normalized to the steady state LPP flow rate. As shown in Fig. 6-8, the LPP flow is in fact negative from almost 50 seconds to the end of the data at 200 seconds. The data are taken directly from Baumann et al., so the additional COMMIX-1A prediction is not relevant to the current work. The main point was to simply show the observational data.

As stated earlier, all of the observational data will be used to calibrate the RELAP model. This means the emulators must estimate the RELAP predictions for the OTC and TTC temperatures, as well as the LPP flow rate.

### 6.3 EBR-II IET RELAP Model

A schematic of the EBR-II Transient Test No. 10 RELAP model (from now on referred to as the IET RELAP model or just the IET model) is shown in Fig. 6-9. The different colors represent different component groups. The grey squares are the inlet/outlet boundary conditions. The dark blue shapes are the high pressure piping network and the lighter blue shapes are the low pressure piping network. The dark green shapes represent the HPP while the light green shapes represent the LPP. The core channel, which is divided up into the lower blanket (LB), driver fuel (DF), and upper blanket (UB) zones, is given by the bright red shapes. The IB channel is shown by the maroon shape beside the core channel. The OB channel is divided into two zones, the OB-lower adaptor (OBLA) tubes and the OB subassemblies, which

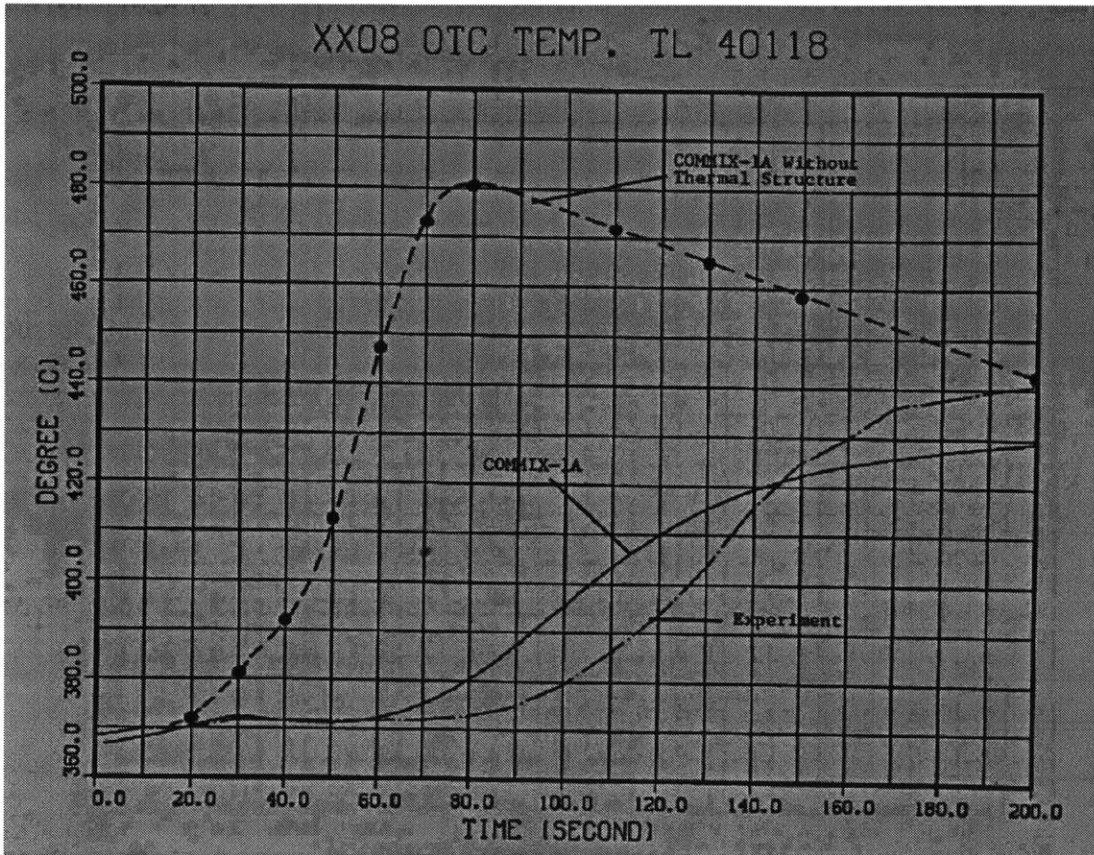


Figure 6-6: Transient Test No. 10 Core Outlet Temperature Data

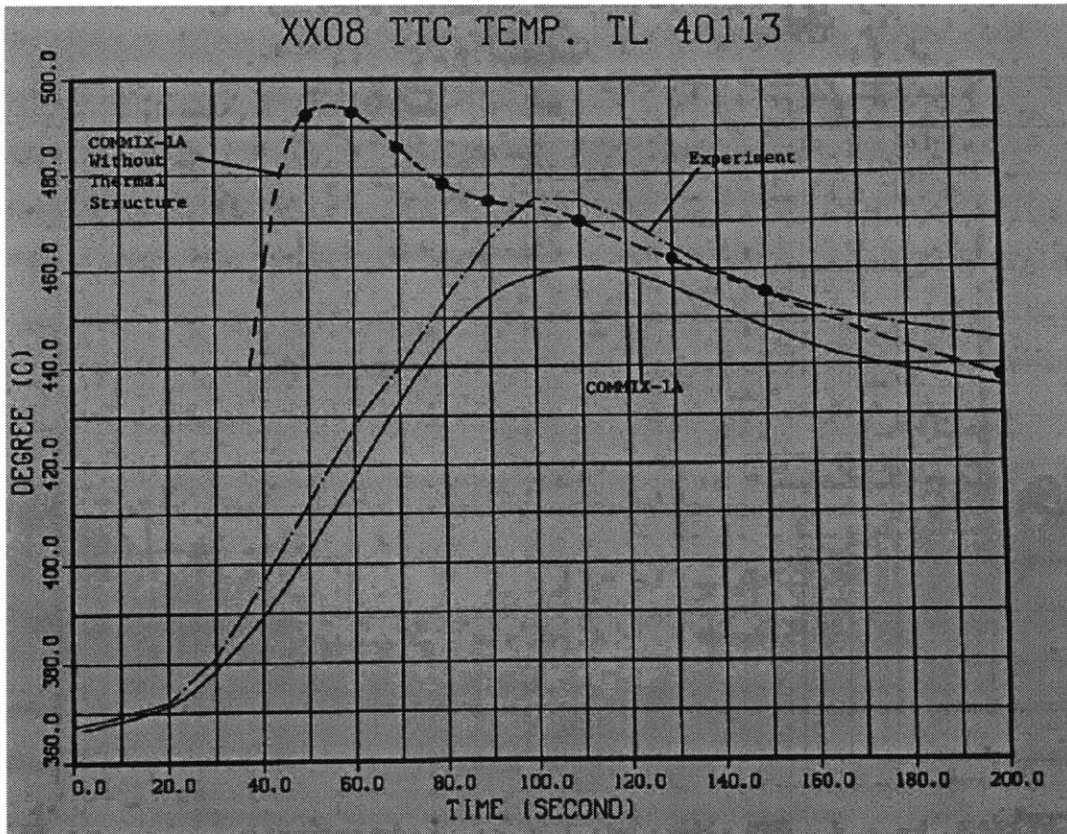


Figure 6-7: Transient Test No. 10 Top-of-Core Temperature Data

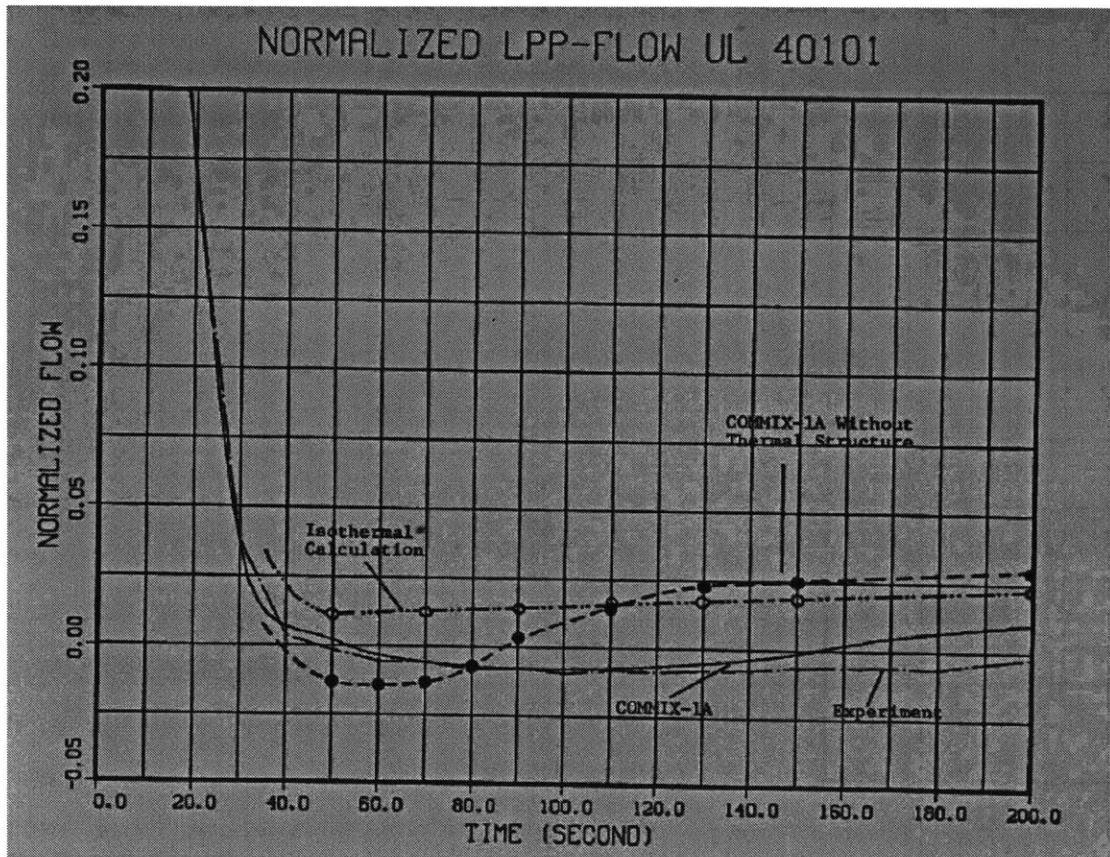


Figure 6-8: Transient Test No. 10 Low Pressure Flow Rate Data

are represented by the light pink shapes. The upper (outlet) plenum is the yellow rectangle.

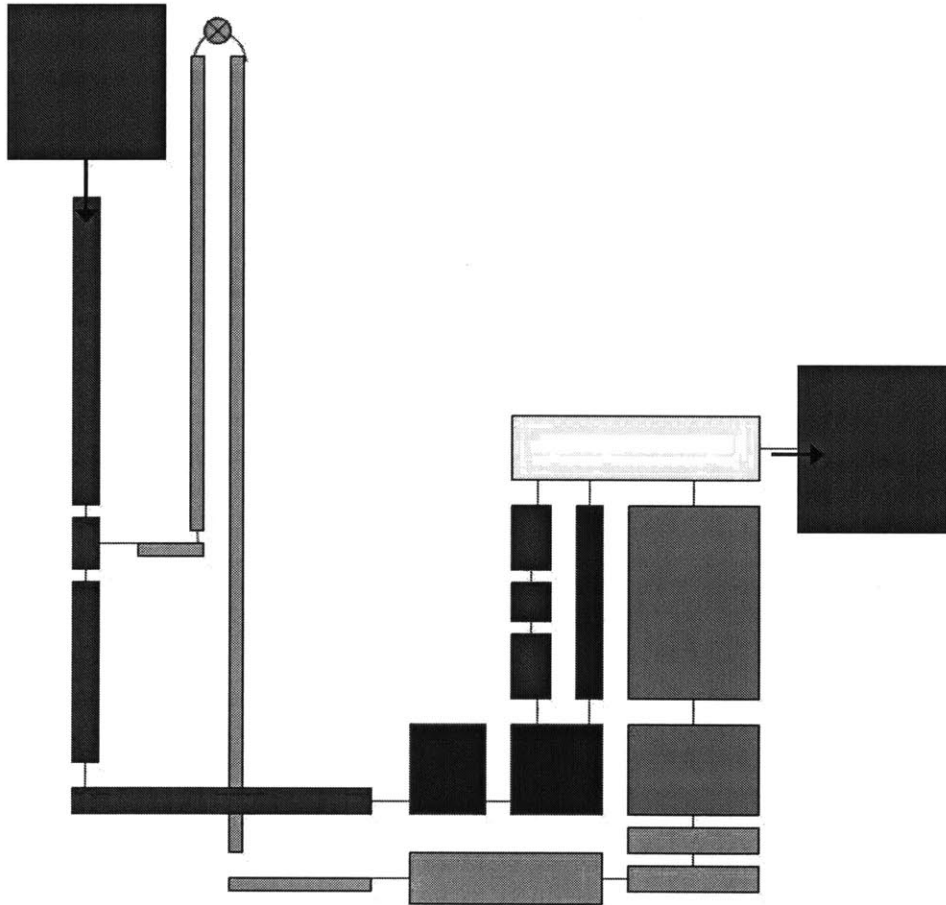


Figure 6-9: EBR-II IET RELAP model schematic

The following sections go through each of the component types in more detail. The main focus is to describe the various uncertain parameters used in each of the components, but the actual EBR-II components are compared to their RELAP model counterparts. But before describing the components, the RELAP user-defined

friction factor correlations are discussed as well as the RELAP heat transfer model setup. This aids the component description since I describe what components share friction factor parameters. RELAP is not setup to account for conduction within the liquid metal coolant, therefore additional steps were required to allow RELAP to include these effects. All of the conduction within the fluid for radial conduction are described together, followed by the axial conduction parameters. The material properties were not treated as uncertain in this work, but the material properties used are described as well. Lastly, the uncertain parameters in the boundary and initial conditions are discussed before summarizing all of the uncertain parameters within the IET RELAP model.

### 6.3.1 RELAP User Defined Friction Factor Correlation

In order to have access at changing the parameters within the friction factor correlations, all of the components within the IET RELAP model use the RELAP user-defined friction factor correlation. For the turbulent flow regime, defined as when  $Re \geq 3000$  for a fluid cell the user-defined friction factor is the standard friction factor formulation of:

$$f^T = A + B (Re)^{-C}. \quad (6.1)$$

In the laminar flow regime, defined for when the fluid cell Reynolds number is,  $0 \leq Re \leq 2200$ , the friction factor is:

$$f^L = \frac{64}{Re\Phi_S}. \quad (6.2)$$



In the transition regime, the friction factor is an interpolation between the above friction factor values defined as:

$$f = \left( 3.75 - \frac{8250}{Re} \right) (f_{3000}^T - f_{2200}^L) + f_{2200}^L, \quad (6.3)$$

where  $f_{3000}^T$  denotes the turbulent friction factor computed at  $Re = 3000$  and similarly  $f_{2200}^L$  denotes the laminar friction factor computed at  $Re = 2200$ .

The turbulent friction factor correlation is the same format as used in the simple friction factor calibration demonstrations in Chapter 4. Because of that, the  $A$ -parameter is set to 0 for all components. But unlike Chapter 4, the parameters are not reparameterized here. The  $B$  and  $C$ -parameters are referred to collectively as the turbulent friction parameters, and  $B$ ,  $C$ , and  $\Phi_S$ -parameters are collectively referred to as the friction parameters.

The user-defined friction factor correlation has hard coded flow regime transition Reynolds numbers within RELAP. For components that are modeling structures very different from pipes, such as the wire-wrapped fuel bundle within the driver fuel zone of the core, the flow regime transition Reynolds number could be very different from these values [46]. The the  $B$ ,  $C$ , and  $\Phi_S$ -parameters must therefore change accordingly during the calibration process to allow the predictions to match the data.

### 6.3.2 RELAP Heat Transfer Models

In RELAP only two heat transfer correlations exist for liquid-metal coolants. One for non-bundles and one for bundles. I only allow heat transfer within the reactor and so correlation intended for the bundles was used. That correlation is the Westinghouse

[46] given as:

$$Nu = 4.0 + 0.33 \left( \frac{P}{D} \right)^{3.8} \left( \frac{Pe}{100} \right)^{0.86} + 0.16 \left( \frac{P}{D} \right)^5, \quad (6.4)$$

where  $P/D$  is the pitch-to-diameter ratio of the rods, and  $Pe$  is the Peclet number,  $Pe = RePr$ . In order to modify the heat transfer coefficient, RELAP does not allow changing any of the coefficients or exponents in the Westinghouse correlation. But RELAP does allow heat transfer multipliers to be applied to Westinghouse correlation. The heat transfer uncertain parameters are therefore the multipliers applied by RELAP to each of the heat transfer coefficients computed internally within RELAP.

Heat structures must be defined within RELAP and connected to corresponding fluid cells. The specific reactor vessel components heat structures are described within their corresponding sections below.

### 6.3.3 High and Low Pressure Piping Network

There are two separate inlet piping networks, one for each of the two primary pumps. But to simplify the RELAP model nodalization, the two inlet piping networks were lumped together. Following the RELAP guidelines [17], the flow areas for all the pipes were then doubled while the hydraulic diameters of the pipes were maintained at their actual values. The inlet to the high pressure piping network represents the exit of the primary pumps. But as already described the primary pumps are not modeled so the inlet simply connects to the inlet boundary condition. The flow split between the high and low pressure piping is modeled as a TEE component within RELAP with the flow split junction given its own junction loss coefficient. All of the

high and low pressure pipe cells share the same friction factor uncertain parameters

The low pressure throttle valve is shown in Fig. 6-9 as the circle with the "X" within it. It is located at the correct elevation above the reactor center plane. The throttle valve itself is simply modeled as a single junction between two low pressure piping sections (the upward vertical portion to the left of the throttle valve and the downward vertical portion to the right of the throttle valve). It has its own loss coefficient as an uncertain parameter.

The flow path of the high and low pressure piping network is not modeled exactly. I match any elevation changes, but as shown in Fig. 6-5, the high and low pressure pipes have several twists and turns within the  $x - y$  plane (where  $x - y$  plane refers to a horizontal plane at a particular axial location). I do not model those particular twists. As shown in the IET model schematic in Fig. 6-9, the RELAP model is effectively a 2-D projection of the high and low pressure pipes in the  $z - y$  plane (where  $z - y$  plane refers to a vertical plane). The elevation changes are all correct but any additional frictional losses from the horizontal twists and turns are simply lumped into the various loss coefficients within the pipe network models. This lumping approach is reasonable because sodium can be modeled as an incompressible fluid, thus the spatial distribution of the form losses within the flow path does not affect the flow/pressure drop characteristics of the pipe. All of the elbow bends within the pipe network components share the same loss coefficient. Additionally the junctions that connect the high pressure pipe to the HPP and the low pressure pipe to the LPP have their own separate loss coefficients.

The number of fluid cell volumes used in each of the pipe sections was taken from a references on validating the MARS code with EBR-II data [43]. Although this reference looked at modeling an SHRT test, their piping network nodalization provided a useful starting point so that I did not have to perform a sensitivity study

on the nodalization scheme.

### **6.3.4 Lower Plenums**

The lower plenum are quite difficult in order to model all of the fluid flow paths exactly. Figure 6-10 shows an illustration of the lower plenum from the Addendum to the Hazard Report, with the locations of the HPP and LPP denoted in red. The LPP is located beneath the HPP and is annular in shape. It sits only below the OB subassemblies. The HPP sits beneath all of the subassemblies, but only connects to the core and IB subassemblies.

#### **High Pressure Plenum (HPP)**

The HPP flow path is very complex. Upon exiting the high pressure pipe, the high pressure stream enters an annular portion before flowing through a baffle plate. After the baffle plate the high pressure stream flows around the the OBLA tubes. From here, the high pressure stream enters an interior zone where the coolant enters into the core and IB lower adaptor tubes through their respective inlet nozzles. The inlet nozzles are a series of small holes at the bottom of the core and IB lower adaptor tubes. As shown in Fig. 6-11, the interior zone has a stepped lower grid which allows for orificing of the flow through the various rows of the core. The IB lower adaptor tubes have more of the inlet nozzle holes covered up by the steps than the hotter core channels.

The RELAP model of the HPP was broken into two zones, the outer HPP (OHPP) and the inner HPP (IHPP). The OHPP spans the flow path volume from the entrance of the HPP, through the baffle plate and around the OBLA tubes. The IHPP covers the interior portion of the HPP that has the orificing stepped grid and

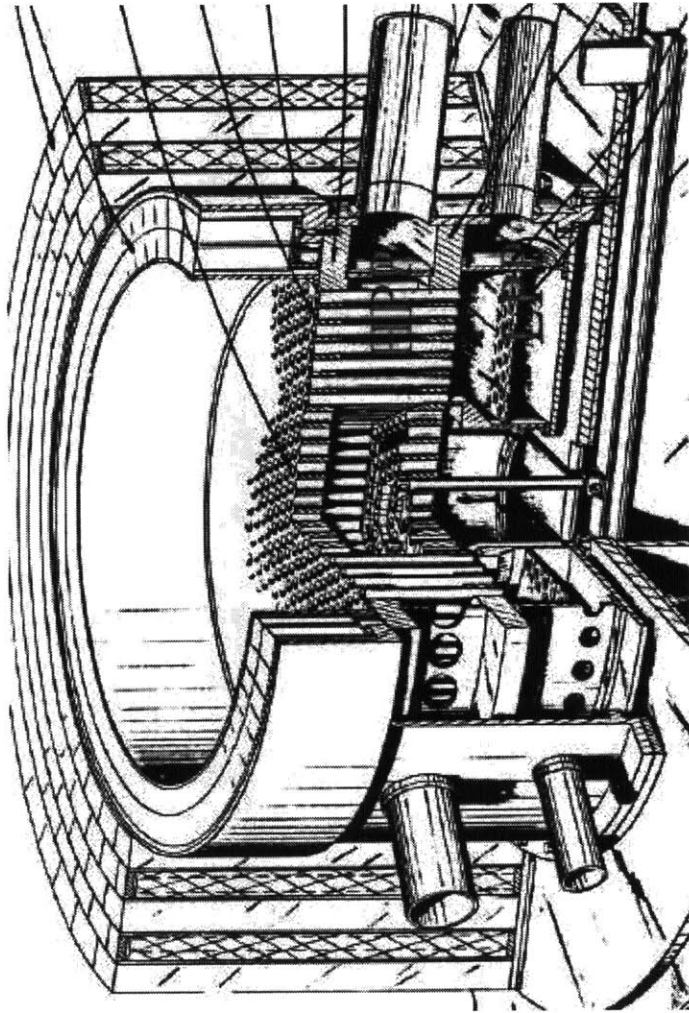


Figure 6-10: EBR-II lower plena



the core and IB lower adaptor tubes. These two zones are modeled as single volumes within RELAP with the wall friction turned off. Wall friction uses the friction factor coefficient (which as stated earlier is always the user-defined form for the IET model), and by turning it off all frictional losses within the two HPP volumes comes entirely from the loss coefficients within the HPP model. The OHPP inlet is given its own loss coefficient as well as the IHPP inlet. The IHPP inlet loss coefficient effectively models the frictional losses due to flowing passed the rows of OBLA tubes.

As a further simplification, any frictional losses due to entering the inlet nozzle holes, turning and flowing upward through the core and IB lower adaptor tubes is modeled just a two separate loss coefficients: one for the core channel inlet nozzles and the other for the IB channel inlet nozzles. Part of the reason for this was the exact dimension of the inlet nozzle holes is difficult to find in addition to finding the exact orificing layout of the stepped grid. Thus, the core and IB channel inlet nozzle flow areas are not modeled as the orificing layout shown in Fig. 6-11, but simply modeled as constant flow areas with loss coefficients applied to them.

### **Low Pressure Plenum (LPP)**

The LPP flow path is also quite complex. Upon exiting the low pressure piping, the low pressure flow stream enters an annular zone before flowing through a baffle plate. It then has to turn upwards within a second annular zone and passes through a second baffle plate where it then continues flowing upward and enters the OBLA inlet nozzles. The OBLA inlet nozzles, also shown in Fig. 6-11, are slightly more conventional nozzle shapes than the core and IB inlet nozzles.

The three flow zones described above are turned into three separate single volumes in RELAP, as shown in Fig. 6-9. The first LPP volume covers the annular zone

between the entrance from the low pressure piping to the first baffle plate. The second LPP volume is the lower inner annular zone where the flow turns upward to the second baffle plate. And the last LPP volume is the zone just before the OBLA inlet nozzles. The LPP frictional modeling format is the same as that used in the HPP, all wall friction is turned off and the frictional losses are lumped into several loss coefficients. The LPP inlet, from the low pressure piping, is given its own loss coefficient and each LPP baffle plate has its own loss coefficient. The second LPP baffle plate has the losses attributed to also turning the flow upwards lumped into it. The OBLA inlet nozzles also have their own loss coefficient.

### **6.3.5 Core Channel**

The core channel consists of 53 core subassemblies and 12 control + 2 safety subassemblies. For simplicity the control and safety subassemblies are lumped into the core subassemblies into the IET model. Any additional frictional losses due to the control and safety subassemblies will then be accounted for by the various loss coefficients within the IET model. The core subassemblies, shown in Fig. 6-12, are divided into three sections, the lower blanket (LB), driver fuel (DF), and upper blanket (UB). All dimensions are in inches because the figure was taken from the Addendum to the Hazard Report. The bundle is surrounded by a hexcan, as are all subassemblies within the reactor vessel. The LB and UB have the exact same geometry and are 19 pin hexagonal bundles with bare rods. All rods but the center use blanket material, while the center rod in the LB/UB sections is steel. The DF section is a wire-wrapped bundle with 91 rods, all with fuel material. Figure 6-13 gives the specifics of the layout of the DF and LB/UB rods. Fastener grids hold the three sections together.



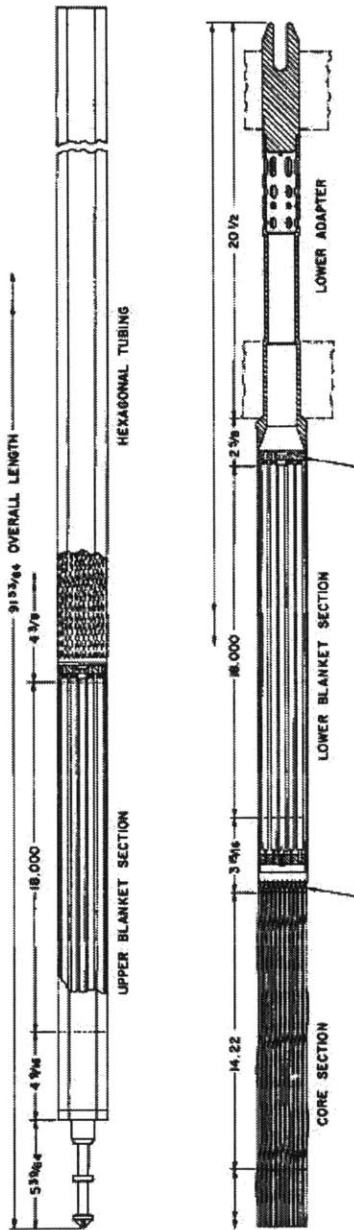


Figure 6-12: Core subassemblies

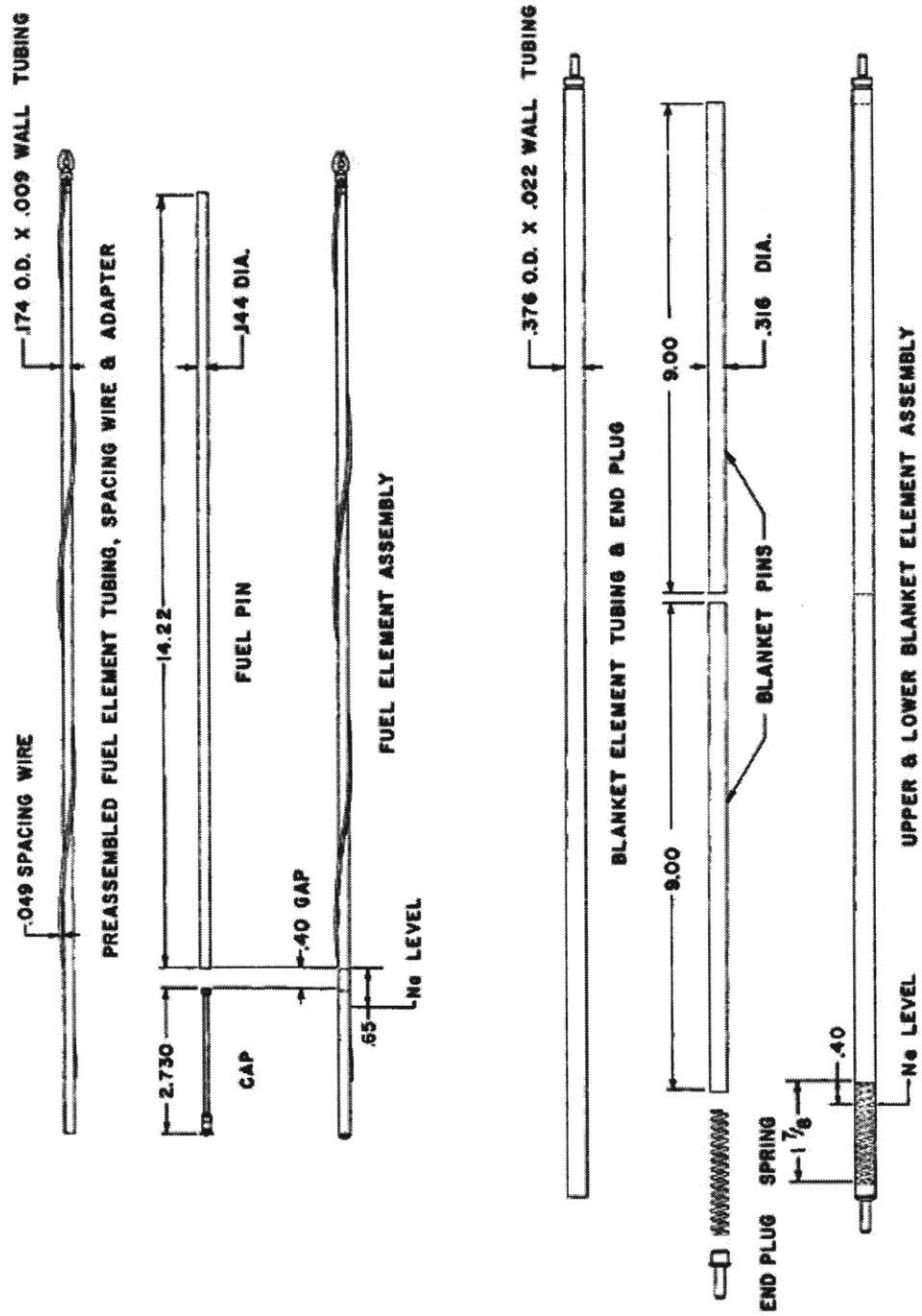


Figure 6-13: Core rods (DF and LB/UB)

## Fluid Cells

All of the core subassemblies are lumped together to form an average core channel. Thus, the IET RELAP model cannot distinguish between hotter and colder channels, it simply models an average-lumped core channel. As shown in Fig. 6-9, the core channel is divided into three zones, just like the actual core channel. The lowest zone is the LB, the middle zone is the DF, and the top zone is the UB. With all core subassemblies lumped together, the RELAP channels have a flow area equal to the total flow area of all of the respective zones, but the hydraulic characteristics of a single subassembly. The number of cell volumes in each zone was also taken to be consistent with Ref. [43]. The LB has 6 fluid cells, the DF has 5 fluid cells, and the UB also has 5 fluid cells. The LB zone has an additional fluid cell due to the "diffuser" portion of the subassembly located the first  $2\frac{5}{8}$  inches from the start of the core subassembly (above the lower support grid). The last cell in each zone represents the fastener portion that connects the zone to the next one, or in the case of the UB is the subassembly exit portion.

Since the LB and UB zones share the same geometry, they share the same friction uncertain parameters parameters. The DF zone has its own set of friction parameters. The fasteners would add to the frictional losses through the subassemblies due to additional form losses. However, I do not give each fastener its own loss coefficient. Any additional losses due to the fasteners are lumped into the core channel inlet nozzle loss coefficient.

## Heat Structures

In RELAP heat structures must be defined in order to generate and/or transfer heat to fluid cells. The temperature distribution through the heat structure is determined

using a finite difference approximation to the heat conduction equation along with the specified boundary conditions. The user must specify how the heat structure axial cells will connect to the axial cells in the corresponding fluid cell volumes. For simplicity, I used a one-to-one correspondence between heat structure cells and fluid cells, so one heat structure axial cell connects to one and only one fluid cell. The LB and UB heat structure setup was identical and represented the average LB and UB rod within the average LB/UB section, respectively. The DF heat structure represented the average driver fuel rod within the average driver fuel subassembly. To do so, the total heat transfer surface area for each rod type was equal to the total heat transfer surface area from all rods from all core subassemblies.

The nodalization of the core heat structures followed Memmott [47]. A total of 9 nodes, or 8 intervals, were used in the DF heat structure. Five intervals within the active fuel region, one interval for the thermal bond gap, and two intervals for the clad. The LB/UB heat structure meshes also used 9 nodes, or 8 intervals, but each interval was over the same material type, the blanket material. The steel center rod in the LB/UB sections was also included but since it does not have heat generation only 5 nodes were used. The axial power profiles in each section were setup to try and match Baumann et al. [45].

The LB/UB heat structures share the same heat transfer coefficient multiplier, while the DF has its own heat transfer coefficient multiplier.

### **6.3.6 Inner Blanket and Outer Blanket Channels**

The Inner Blanket (IB) and Outer Blanket (OB) channels are almost identical to each other. Both are 19 rod hexagonal bundles with bare rods within the hexcan. Figure 6-14 shows the IB and OB subassembly schematic while Fig. 6-15 shows a

cross section of the IB/OB bundle. The key differences are the lower adaptor inlet nozzles between the IB and OB subassemblies as well as the flow distribution strips within the bundles. As described earlier, the IB lower adaptor tubes are not modeled explicitly. The IHPP connects directly to the core and IB channels and any effect from the core and IB lower adaptor tubes are lumped into the core and IB inlet nozzle loss coefficients. Including the OBLA tubes was necessary to do the elevation differences within the reactor vessel, so as shown by Fig. 6-9 the LPP zone 3 connects to the OBLA tubes, which then connect to the OB channel. The flow distribution strips within the OB subassembly are larger and yield a non-trivial reduction in the OB subassembly flow area compared to the IB subassembly. Including the flow strip effect on the subassembly hydraulic characteristics was difficult though due to the change in the wetted perimeter. Thus, any additional frictional losses due to the flow strips was accounted for by IB and OBLA inlet nozzles..

## Fluid Cells

As with the core channel, the IB and OB channels represent the averaged-lumped IB and OB subassembly within the reactor vessel, respectively. The IB and OB axial cell lengths were set to be consistent with the lengths used in the core channel, giving a total of 15 axial cells in each. Since the flow strips are not explicitly modeled, the IB and OB bundles have the same geometry and therefore share the same friction parameters. The added frictional losses in the OB channel due to the flow strips is, again, accounted for by the OBLA inlet nozzle loss coefficient.

The OBLA tube volume are also an averaged-lumped representation of the OBLA tubes. Four axial cell volumes are used, and the OBLA tubes have their own set of friction parameters since they are different from the OB subassembly geometry (they

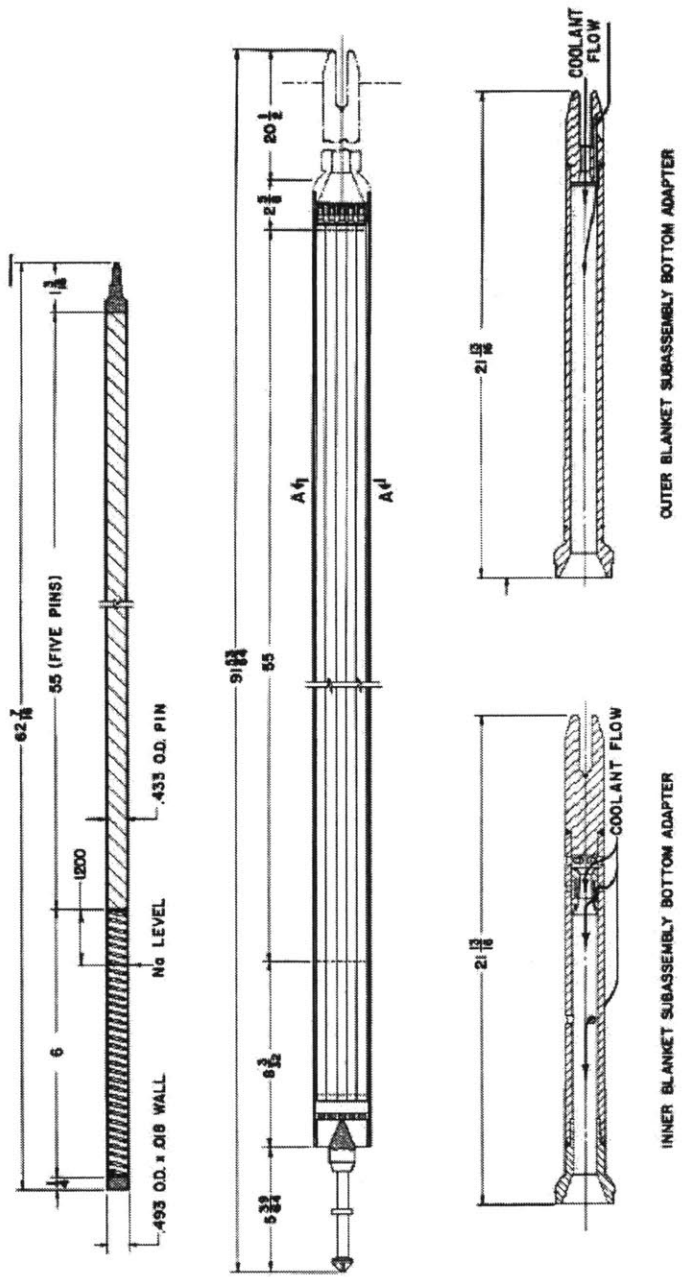


Figure 6-14: IB and OB subassemblies

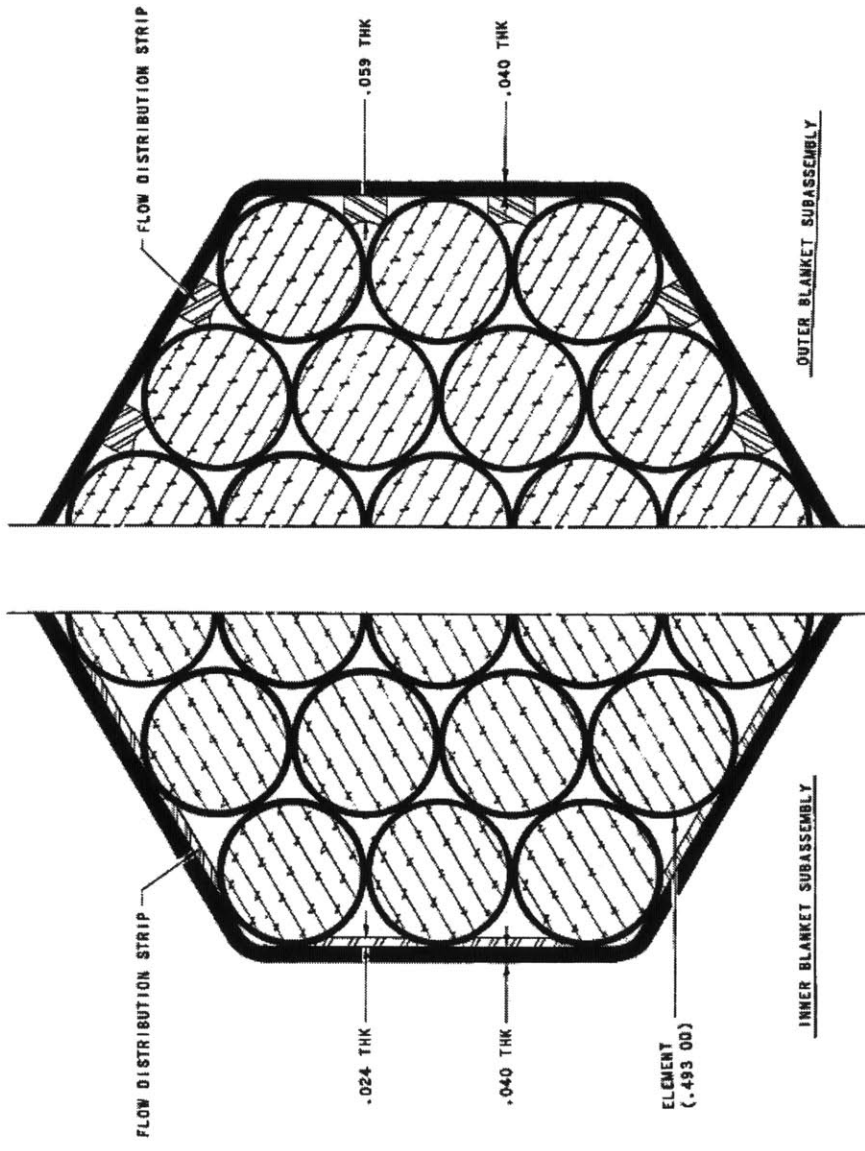


Figure 6-15: IB/OB bundle cross-section

are simply straight pipes). The connection between the OBLA tubes and the OB subassembly channel is modeled by a single junction but for simplicity there is no loss coefficient applied here. Any additional frictional losses are again simply lumped into the OBLA inlet nozzle loss coefficient.

### **Heat Structures**

The IB/OB heat structures are modeled similarly to the LB/UB heat structures. Nine nodes, 8 intervals, are used all within the same blanket material. The axial power profile is also setup to follow Baumann et al. [45]. The IB/OB heat structures also share the same heat transfer coefficient multiplier.

### **6.3.7 Upper (Outlet) Plenum**

The upper (outlet) plenum is single fluid volume, as shown in Fig. 6-9. The one complication is that all of the reactor vessel channels, the core channel, IB channel, and OB channel connect to this one volume. Additionally the outlet pipe which connects to the outlet boundary condition also connects to this single volume. Therefore a RELAP BRANCH component was used to model the upper (outlet) plenum. Like the lower plena volumes, all wall friction was turned off in the upper (outlet) plenum, but no additional loss coefficients were applied here.

### **6.3.8 Conduction Within the Fluid Models**

#### **Theory**

RELAP was originally developed for light water reactor analysis and so does not represent axial or radial heat conduction within the fluid because of the relatively



low thermal conductivity of water [48]. But with the thermal conductivity of sodium roughly 100 times that of water and the fact that we are modeling very low flow conditions during the transition from forced to natural convection, axial and radial heat conduction within the sodium coolant could be very important. Following Memmott (2009) [47] and Davis (2007) [48], axial and radial conduction was accounted for by creating “pseudo” heat structures whose source terms were determined by axial and radial conduction equations using the RELAP control variable scheme. Note RELAP control variables is the nomenclature within RELAP that allows computing quantities from variables within RELAP, just as how the QPIRT quantities were computed in Chapter 3. RELAP control variables does not refer to the control variable nomenclature I use for model calibration.

The axial and radial conduction models will be described for axial conduction only, but the theory can be extended to radial conduction as well. I will use nomenclature consistent with Davis (2007) for describing the axial conduction equations. But as will be seen shortly, the key difference for my implementation of the axial/radial conduction equations is I treat several key parameters as uncertain which must be inferred out from the data. Davis (2007) and Memmott (2009) treat those parameters as known values computed from the geometry and material properties.

Figure 6-16 illustrates a simple nodalization of three axial cells and two junctions. The temperature,  $T$ , at each cell volume center is assumed known. Fourier’s law is used to calculate the heat transfer rate,  $\dot{q}$ , between cell  $m - 1$ , and junction  $m - 1/2$ :

$$\dot{q}_{m-1,m-1/2} = -k_{m-1}A_{m-1} \frac{T_{m-1/2} - T_{m-1}}{0.5\Delta x_{m-1}}. \quad (6.5)$$

The heat transfer rate between junction  $m - 1/2$  and cell  $m$  is likewise computed as:

$$\dot{q}_{m-1/2,m} = -k_m A_m \frac{T_m - T_{m-1/2}}{0.5\Delta x_m}. \quad (6.6)$$

Davis (2007) defines a volume parameter,  $B$  as,

$$B_m = \frac{k_m A_m}{\Delta x_m}, \quad (6.7)$$

and a junction parameter,  $D$ :

$$D_{m-1,m} = \frac{-2B_{m-1}B_m}{B_{m-1} + B_m}. \quad (6.8)$$

Assuming no energy is stored at the junction between volumes,  $\dot{q}_{m-1,m-1/2}$  and  $\dot{q}_{m-1/2,m}$  can be equated to solve for the junction temperature:

$$T_{m-1/2} = \frac{B_{m-1}T_{m-1}B_mT_m}{B_{m-1} + B_m}. \quad (6.9)$$

Substituting in the junction temperature into either the expressions for  $\dot{q}_{m-1,m-1/2}$  and  $\dot{q}_{m-1/2,m}$  gives the heat transfer rate between cell  $m - 1$  and cell  $m$  as:

$$\dot{q}_{m-1,m} = D_{m-1,m} (T_m - T_{m-1}). \quad (6.10)$$

In more complicated situations when multiple cells are attached to a single cell volume, a parallel resistance thermal circuit can be used to solve for the heat transfer rate from cell  $j$  to cell  $m$ , as shown in Fig. 6-17. The heat transfer rate from cell  $j$

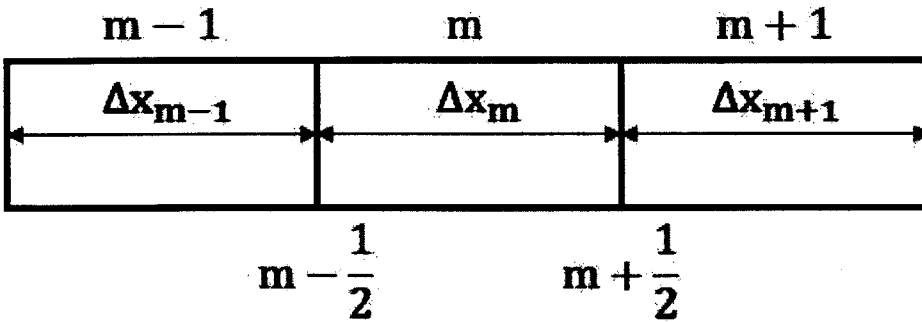


Figure 6-16: Nodalization diagram for axial heat conduction

to cell  $m$  is:

$$\dot{q}_{j,m} = -2B_j \left( \frac{B_m (T_m - T_j) + \sum_i B_i T_i - T_j \sum_i B_i}{B_m + \sum_i B_i} \right). \quad (6.11)$$

The total heat transfer rate received by cell volume  $m$  through the “face”  $m - 1/2$ , is simply the summation of the heat transfer rate from each of the cells that are connected to it:

$$\dot{q}_{m-1/2,m} = \sum_j \dot{q}_{j,m}. \quad (6.12)$$

A similar derivation can be made for the case of multiple cells attached to the outlet of cell volume  $m$ . The heat transfer rate from cell  $m$  to cell  $j$  through the “face”  $m + 1/2$  is:

$$\dot{q}_{m,j} = -2B_j \left( \frac{B_m (T_j - T_m) - \sum_i B_i T_i + T \sum_i B_i}{B_m + \sum_i B_i} \right). \quad (6.13)$$

The total heat added to a cell volume  $m$  due to axial conduction is then just the summation of the heat transferred through the inlet faces and outlet faces attached

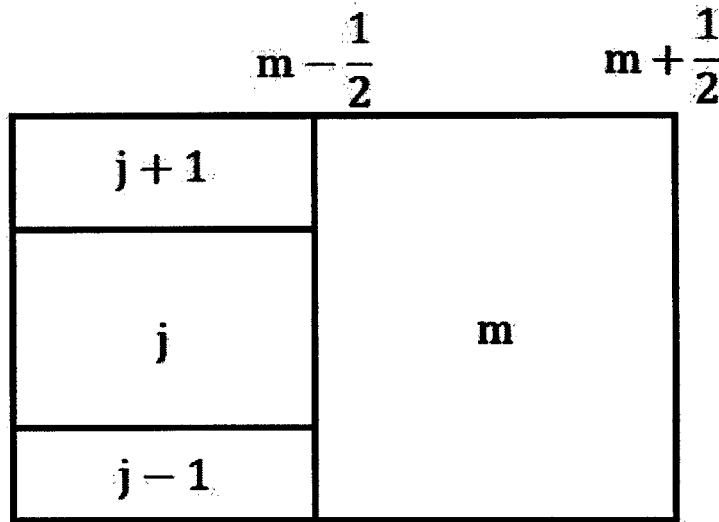


Figure 6-17: Axial conduction nodalization with multiple cells

to that cell:

$$\dot{Q}_m = \sum_{inlets} \dot{q}_{i,m} - \sum_{outlets} \dot{q}_{m,i}. \quad (6.14)$$

Similar equations can be derived for the heat transfer due to radial conduction. Though the volume parameters  $B_m$  must be defined accordingly since the radial distances are not as straightforward as the axial distance between cell centers in RELAP nodalizations.

### Implementation

As already stated, Davis (2007) and Memmott (2009) must compute the volume parameters for axial and radial conduction based on the specific geometry and fluid properties of the RELAP model. I however, treat each of the conduction volume parameters as an uncertain parameter. For simplicity, I assume the axial and radial

conduction volume parameters (which will just be referred to as the axial and radial conduction parameters) are constant within a component. So each cell volume within the DF component has the same axial and radial conduction parameter value, for example. Additionally, the LB and UB axial and radial conduction parameters are assumed to be the same. Since the core, IB, and OB channels are all connected to the upper (outlet) plenum, the upper plenum also has an axial conduction parameter associated with it. However, I do not include axial conduction between the core, IB and OB channels with their respective lower plena.

Figure 6-18 shows the conduction network between the various components within the reactor vessel. Axial conduction is determined between not only the cells within a particular component, but between the axial cells of adjacent components. The result is that the reactor vessel heat transfer is completely interconnected through axial and radial conduction.

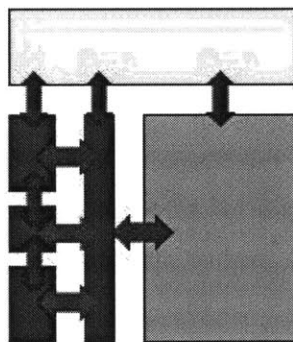


Figure 6-18: Axial and radial conduction illustration

The RELAP control variable system is used to compute the  $\dot{Q}_m$  value in each cell within the reactor vessel due to both axial and radial conduction. These quantities become the source terms to a set of “pseudo” heat structures that are connected to

the corresponding fluid cells. The dimensions of these pseudo heat structures are set so that they represent less than 1% of the actual heat structure volume attached to each component. For example, the pseudo heat structures due to axial and radial conduction attached to the DF fluid cells have their volume set so they are less than 1% of the total driver fuel rod and wire-wrapped spacer volume. Additionally the RELAP heat structure direct heating option is used. As the name implies, direct heating directly adds the heat structure heat to the fluid cell volume. Therefore it is not added by through a heat transfer coefficient. This ensures the axial and radial conduction "source terms" are added to their corresponding fluid cell volumes correctly.

### **6.3.9 Material Properties**

The material properties were all considered known. The main reason for this was to simplify the problem given the already large number of uncertain parameters involved. The driver fuel and blanket fuel material properties were considered constants while the stainless steel material properties were determined using a linear fit. All of the values were taken from Baumann et al.. The thermal bond gap effective thermal conductivity within the driver fuel rod however were estimated by computing the temperature drop from the Addendum to the Hazard Report results. The thermal bond gap volumetric heat capacity was estimated using sodium properties at the steady-state operating temperatures. The thermal bond typically represents a negligible thermal resistance and a negligible thermal capacity, which is why it was not treated as uncertain in this work.

### 6.3.10 Boundary Conditions

As stated earlier, the boundary conditions from Baumann et al. specify the total primary flow rate at the pipe inlet and the inlet coolant temperature. Additionally, the outlet pressure is held constant through the simulation. The grey boxes in Fig. 6-9 represent the generic boundary conditions to the IET RELAP model. The inlet mass flow rate is modeled using a time-dependent junction with a specified mass flow rate through time as a general table. The values were taken from the figure given in Baumann et al. and is shown in Fig. 6-19. The decay power was also input as a boundary condition as given in Baumann et al., and is shown in Fig. 6-20.

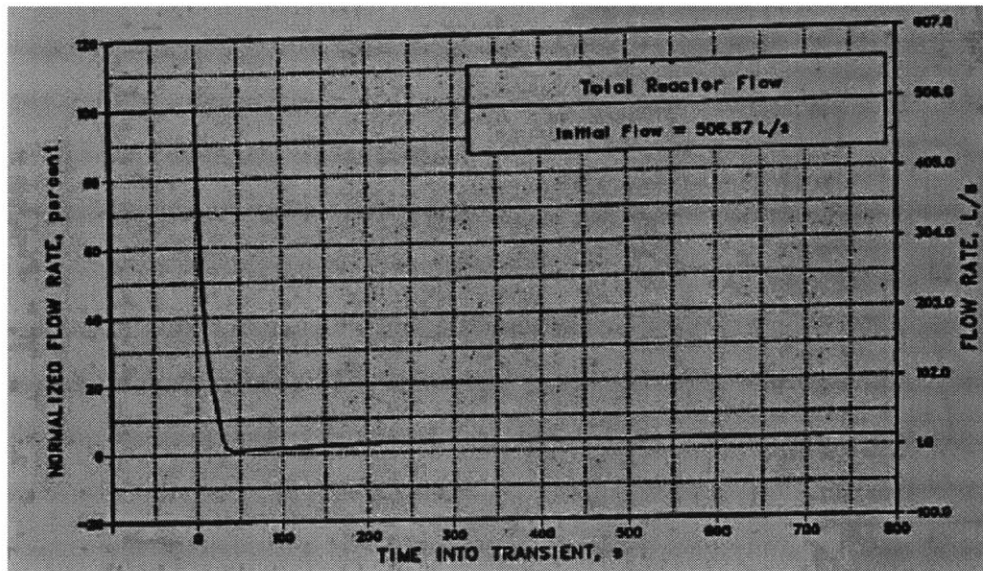


Figure 6-19: Total primary (inlet) flow rate boundary condition

An obvious point from looking at Fig. 6-19 is that it is somewhat difficult to read. The minimum total flow rate occurs somewhere between 40 and 50 seconds after the start of the transient. It is difficult to tell if the minimum value is 0.5% or 1% of nominal flow. Although those differences sound small, 1% of nominal flow

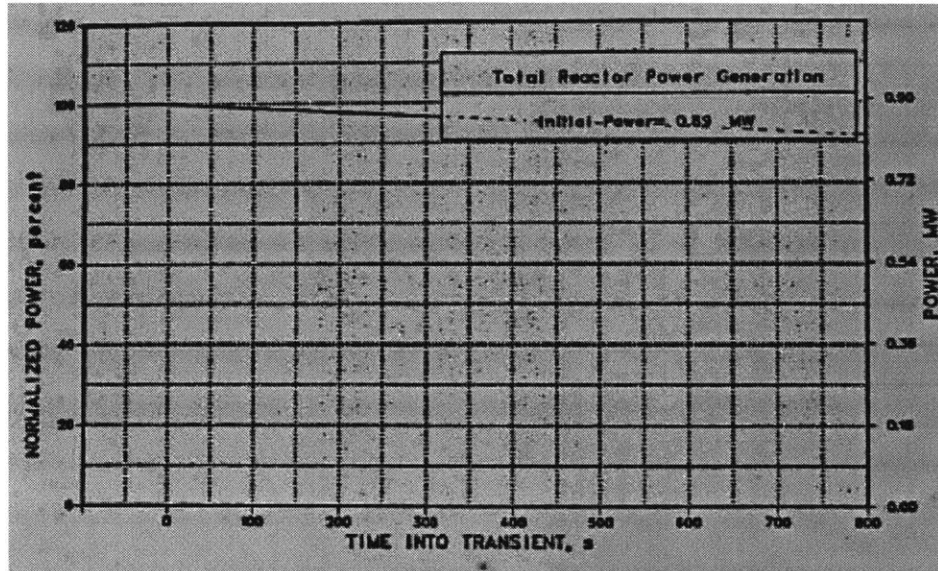


Figure 6-20: Total core power boundary condition

is 100% higher than a minimum flow rate of 0.5% of nominal flow. Likewise at the end of the transient, what I have been referring to as the natural circulation rate value can be between 2% and 5% of nominal flow. It is simply too difficult to read with any more accuracy than that from Fig. 6-19. Therefore, these two mass flow rates, the minimum flow rate and the natural circulation flow rate were considered uncertain.

Before the minimum flow rate is reached, during the pump coastdown phase it is easier to read a few of the values off of Fig. 6-19. So these values were input into the time dependent junction mass flow rate general table along with the uncertain minimum flow rate value. A linear trend was assumed between the minimum flow rate value and the natural circulation rate at the end of the transient simulation. Although this is not correct, it was a simple method of setting the inlet flow boundary condition during this phase.



### 6.3.11 Initial Conditions

The blowdown problem of Chapter 5 had a very simple initial condition, just a known gas temperature and pressure within the gas tank. Here though it is much more difficult to set the initial conditions through the entire IET RELAP model because each fluid cell volume requires a mass flow, pressure and temperature value assigned to it. Additionally, each of the heat structure temperature nodes must be initialized. Since the initial condition is the steady-state condition for the Transient Test No. 10 decay power level at the given nominal flow rate, I do not know all of the information a priori to initialize every single fluid cell and heat structure node. In RELAP, the standard procedure for this is to first run a “steady-state” calculation where a “steady-state” condition is achieved using the specified boundary conditions. The initial conditions to the “steady-state” calculation are just guesses, and of course the more accurate they are the better. Then using those results “re-start” the simulation with any additional information specific to the transient. I followed the RELAP manual on how to setup and perform the re-start [17].

Additionally, Baumann et al. state that the initial core decay power uncertainty was estimated to be on the order of 10% [45]. Therefore the initial total power level was had an uncertain multiplier applied to it to account for this uncertainty.

### 6.3.12 Summary of Uncertain Parameters

In total, 41 uncertain parameters are in the IET RELAP model. In order to build the training set, bounding values based on the parameter prior distributions needed to be established. For the friction parameters this was relatively straightforward by starting with McAdams turbulent friction parameter values [46] and a laminar shape factor value of 1, and increasing the uncertainty around those values. Additionally,

the SET models, which will be described shortly, helped define the bounding values on many of the friction parameters because the SETs constrain those particular uncertain parameters. For other parameters, such as loss coefficients and especially the radial conduction parameters this was much more challenging to do. The blowdown problem came with priors on each of the uncertain parameters, but priors do not exist for the loss coefficients and radial conduction parameters since they are specific to this IET model and cannot be generalized outside of this context. Therefore I needed to establish priors myself.

My basic strategy for setting bounding values is to run scoping studies that try and find ranges of uncertain parameter values that give predictions that “surround” the observational data. For smaller dimensional problems this, although maybe time-consuming if priors are not well established, is straightforward. But for the larger dimensional problems like the EBR-II IET RELAP model, this is very difficult to do. But, Table 6.1 provides the summary list of all 41 uncertain parameters along with their bounding values used to build the IET model training set. The uncertain parameter numbers are important because that is how they will be referred to in the calibration results later on. The number identifier was easier to use in MATLAB than defining their names. Notice how in Table 6.1 many of the loss coefficients and conduction parameters have very large ranges, with the maximum values being nearly 10x the minimum value. This shows just how uncertain I am in those values *a priori*.

Table 6.1: Summary of IET model uncertain parameters

number	uncertain parameter name	minimum value	maximum value
1	pipe network laminar shape factor	0.75	1.25
2	pipe network turbulent coefficient	0.138	0.23
3	pipe network turbulent exponent	0.15	0.25
4	pipe network flow split loss coefficient	0.65	1.7
5	pipe network elbow loss coefficient	0.35	0.9
6	low pressure throttle valve	5	50
7	HPP inlet loss coefficient	3	15
8	core channel inlet nozzle loss coefficient	5	40
9	IB channel inlet nozzle loss coefficient	1	50
10	IHPP OBLA tubes loss coefficient	3	15
11	LPP inlet loss coefficient	3	15
12	LPP baffle plate 1 loss coefficient	3	15
13	LPP baffle plate 2 loss coefficient	3	15
14	OBLA inlet nozzle loss coefficient	12.5	100
15	LB/UB turbulent coefficient	0.138	0.23
16	LB/UB turbulent exponent	0.15	0.25
17	LB/UB laminar shape factor	0.125	1.875
18	DF turbulent coefficient	0.138	0.23
19	DF turbulent exponent	0.15	0.25
20	DF laminar shape factor	0.125	1.875
21	IB/OB turbulent coefficient	0.023	0.23
22	IB/OB turbulent exponent	0.025	0.25
23	IB/OB laminar shape factor	0.125	5
24	OBLA turbulent coefficient	0.138	0.23
25	OBLA turbulent exponent	0.15	0.25
26	OBLA laminar shape factor	0.125	1.875
27	DF heat transfer coefficient multiplier	0.5	1.5
28	LB/UB heat transfer coefficient multiplier	0.5	1.5
29	IB/OB heat transfer coefficient multiplier	0.5	1.5
30	IB radial conduction parameter	500	4000
31	OB radial conduction parameter	500	4000
32	DF radial conduction parameter	500	4000
33	LB/UB radial conduction parameter	500	4000
34	OB axial conduction parameter	54.5	163.5
35	IB axial conduction parameter	6.5	19.5
36	DF axial conduction parameter	30	90
37	LB/UB axial conduction parameter	33	99
38	UP axial conduction parameter	75	225
39	minimum inlet flow rate	0.05	0.9
40	natural circulation flow rate	1.5	4.5
41	initial core power multiplier	0.9	1.1

## 6.4 Pseudo SET RELAP Models

### 6.4.1 Overview

A series of Separate Effect Tests (SETs) were used to help inform the various wall friction and inlet nozzle loss coefficients within the reactor vessel IET RELAP model. I call these SETs, “pseudo” SETs because the “data” is generated from an empirical correlation rather than coming from a true experiment. The main reason for this was simplicity. I could use the exact components from the IET RELAP model for each of the specific SET RELAP models. Rather than requiring a completely separate RELAP model for each of the specific SETs. Therefore the SETs can be viewed as updating the various uncertain parameter values to match a physical phenomena that the IET RELAP model was not originally capable of handling. Additionally, with the “pseudo” SET format I did not have deal with any potential discrepancy issues between the SETs and IET.

There are two types of SETs: a bundle specific SET and a channel SET. The bundle specific SET only calibrates the friction parameters within the RELAP user-defined friction factor correlation: the  $B$ ,  $C$ , and  $\Phi_S$ -parameters. The channel SET calibrates those same friction parameters as well as the channel inlet nozzle loss coefficient. The bundle specific SET has the “data” generated by the Cheng and Todreas friction factor correlation [46]. For the DF bundle, the Cheng & Todreas wire-wrapped bundle average friction factor correlation is used. While for the IB/OB bundle the Cheng & Todreas bare-bundle friction factor factor correlation is used. The channel SET “data” come from Gapalakrishnan & Gillette (1973) [49], which will be referred to as G&G-1973. Baumann et al. actually used the results from G&G-1973 to set the various loss coefficients and friction parameters in their COMMIX-1A

code for their own EBR-II model. I refer to G&G-1973 as “pseudo” data because the empirical correlations they derive for the EBR-II subassemblies originally came from hydraulically scaled water tests [49]. Therefore it is difficult to tell in the paper if the results I use come from experimental data directly or from the empirical correlations. The G&G-1973 “data” was taken directly from their paper and is shown in Fig. 6-21.

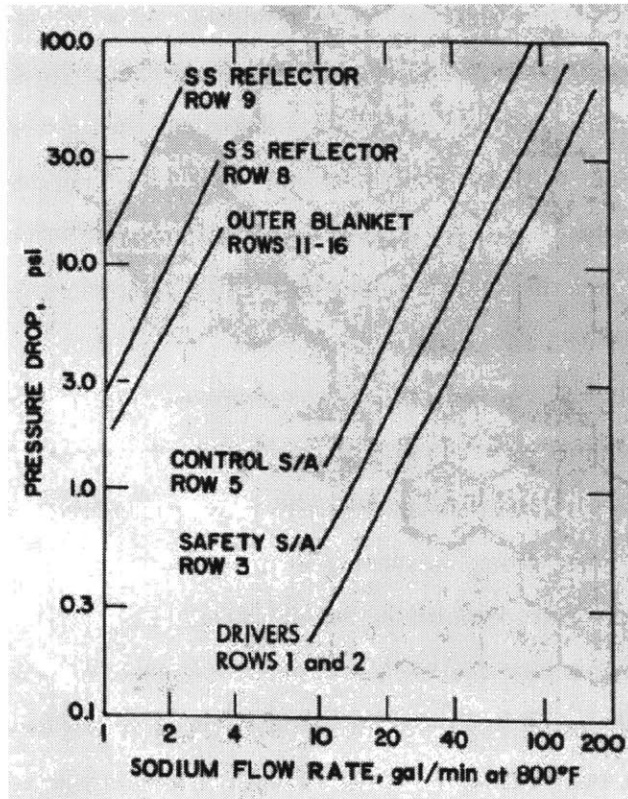


Figure 6-21: G&G-1973 pressure drop “data”

Each of the SETs have the same general RELAP model, as shown in Fig. 6-22. The dimensions and flow rates depend on the specific SET but the component layout is basically the same. The inlet boundary condition is a time dependent junction which sets the mass flow rate through the bundle or channel. This inlet

boundary condition is attached to a dummy volume, labeled as component 200 in Fig. 6-22. Dummy volume 200 is attached to the bundle/channel of interest labeled component 400 via junction 201. If the SET is a channel SET, junction 201 is where the inlet nozzle loss coefficient is applied. The bundle/channel component is attached to another dummy volume, labeled as component 100, which is itself attached to the outlet pressure boundary condition. Both dummy volumes, 200 and 100, have the wall friction turned off. Therefore the pressure drop across the bundle/channel component is determined by subtracting the pressures within dummy volumes 200 and 100. With the outlet pressure fixed, the inlet pressure within the inlet mass flow boundary condition adjusts depending on the frictional characteristics of the components and the specified mass flow rate. The resulting pressure drop is considered to be the RELAP predicted  $\Delta P$  value.

All of the emulators are built using the exact same formulation as described in Chapter 4. The standard GPR emulators are built using the Empirical Bayes approach, however. The main reason for this was to ensure that MATLAB never breached its RAM limit, which was never a concern for calibrating the SETs individually but could occur for simultaneous calibration described later. The Empirical Bayes approach did not use the GPML toolkit but rather found a point estimate from the posterior hyperparameter samples using the sampling procedure described in Chapter 4 for the GPR emulator. The emulator-based uncertain parameter calibration process is performed using the GPR-modified and FFGP-modified likelihood functions described in Chapter 4. A key difference though between the simple friction factor demonstration problem of Chapter 4 and the blowdown problem of Chapter 5, is that the uncertain parameter priors here are all assumed to be uniform between their bounding values. The first reason for doing this is I did not have any *a priori* bias towards any particular values, given the discussion earlier about the IET model

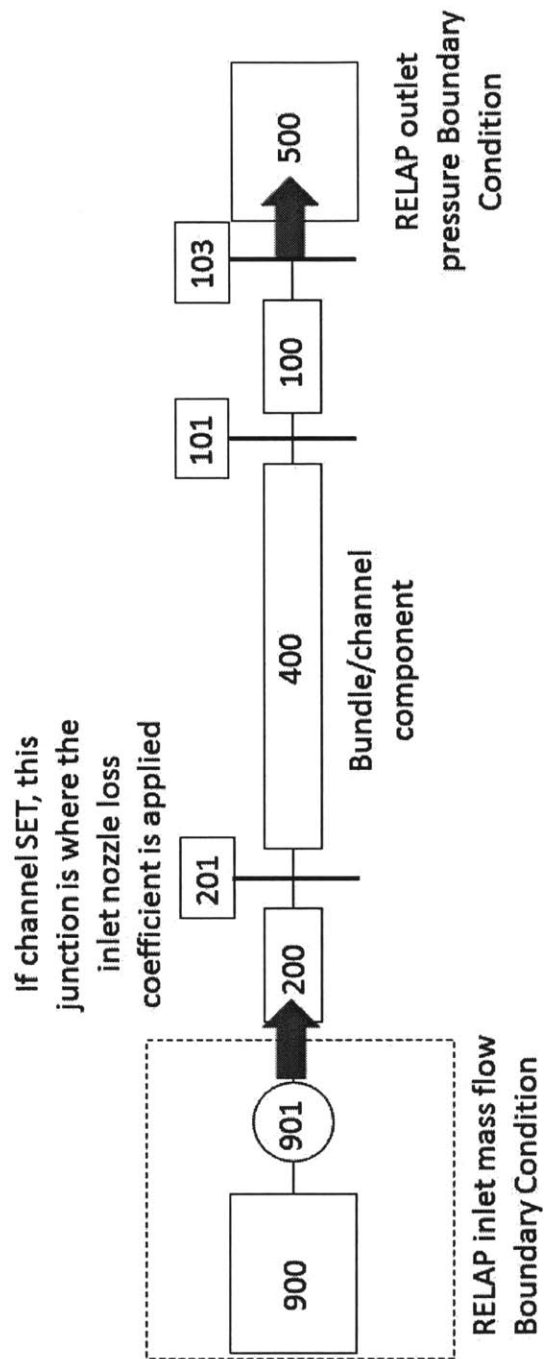


Figure 6-22: Generic “pseudo” SET RELAP model illustration

setup. But a uniform prior completely biases the uncertain parameter values to stay within the prior bounds, since the uniform prior has a density of 0 outside the bounding values. This should keep the uncertain parameters from going outside the training set, an issue which is better suited for future work. The following subsections give the emulator training results for each of the SET RELAP models. Then the emulator-based calibration results are given.

All of the pseudo SET names are summarized in Table 6.2 below. Each SET has its own RELAP model and “observational data”, as well as an accompanying “best” emulator. The criteria for deciding the “best” emulator will be described below. Additionally, Table 6.3 summarizes the various types of FFGP emulators built for each of the SET models. The “maximum number of components” column in Table 6.3 means that all FFGP emulators up to and including that many components were built for the corresponding SET model. For example, 1-component, 2-component, 3-component, and 4-component FFGP emulators were built for the GGcore model.

Table 6.2: “Pseudo” SET Names

Acronym	Full name
CTDF	Cheng & Todreas Driver Fuel bundle-specific $\Delta P$ SET
GGcore	Core Channel G&G-1973 channel $\Delta P$ SET
IBCT	IB/OB Cheng & Todreas bundle-specific $\Delta P$ SET
IBGG	IB G&G-1973 channel $\Delta P$ SET
OBGG	OB G&G-1973 channel $\Delta P$ SET

### 6.4.2 Core Channel SETs

#### Cheng & Todreas Driver Fuel SET

The Cheng & Todreas driver fuel (CTDF) pseudo SET computes the friction factor from the wire-wrapped bundle average Cheng & Todreas correlation. This friction



Table 6.3: SET FFGP emulator type summary

SET	Number of factors	Number of components
CTDF	2	3
GGcore	2	4
IBCT	2	4
IBGG	2	4
OBGG	2	4

factor is then used to compute the pressure drop across the DF subassembly bundle length. This pressure drop value is considered to be the “data” for the pseudo SET. Twenty-five data points are generated ranging from 1% of nominal core channel flow to 100% core channel flow. The observational error was assumed to be that 95% of the probability was covered by  $\pm 33\%$  around the mean data value of the first data point. This error (variance) value was assumed then to be constant for each of the 25 data points.

With 3 uncertain parameters, the DF friction parameters - which correspond to the IET numbers 18, 19, and 20 - a total of 50 RELAP case runs were used. The 50 different uncertain parameter values were determined using the MATLAB function 'lhsdesign' just as in Chapter 4. In the FFGP emulator setup, the uncertain parameter factor is factor 2, while the control variable factor is factor 1. Fifteen control variable locations were taken for the training set. But, the major difference between this current FFGP setup and the simple friction factor demonstration problem in Chapter 4 is that each control variable location (an inlet mass flow rate value) corresponds to a separate RELAP run. Thus, with 50 cases - 50 different values of the uncertain parameters - and 15 control variable locations RELAP had to be run 750 times. The CTDF FFGP training set is shown in Fig. 6-23, as grey lines, along with the Cheng & Todreas “observational data” points as red error bars.

The error bars are difficult to distinguish because the observational error is quite small. Each RELAP run took roughly 2 seconds, thus it took about 25 minutes to build the entire FFGP training set. As that computation time suggests, I did not take advantage of any parallel computing resources. Although my personal PC has a Quad Core processor (with 8 threads) I simply ran the training set in series. A benefit of creating the emulator training sets is that once the training inputs are all specified the training case runs could be run in parallel, which could drastically reduce the computational time of creating the training set. But, I always performed calculations in series to keep things simple.

The FFGP 2-factor 1-component, 2-component, and 3-component emulators were built using the training dataset shown in Fig. 6-23. As in Chapter 4, the FFGP emulator accuracy is compared by examining the likelihood noise hyperparameter point estimate values. Figures 6-24 through 6-26 give the likelihood noise hyperparameter samples in blue and the point estimate in red for the 1-component, 2-component, and 3-component FFGP emulators, respectively. Similarly to the simple friction factor demonstration problem in Chapter 4, the 2-component emulator is much more accurate relative to the training set than the 1-component emulator.

All FFGP emulators were built with a total of  $10^5$  latent burn-in samples and  $2 \times 10^5$  latent posterior samples with the first half of those discarded, yielding a total of  $10^5$  posterior samples saved. Table 6.4 summarizes the FFGP emulator build times. If the total emulator build times are compared to the the time it takes to run a single RELAP case,  $\sim 2$  s, then the CTDF RELAP model could be run 45, 58, and 68 times in the time it takes to build the 1-component, 2-component, and 3-component emulators, respectively.

The built FFGP emulators were then used to calibrate the 3 uncertain parameters in the CTDF SET model. The FFGP-modified likelihood structure described in

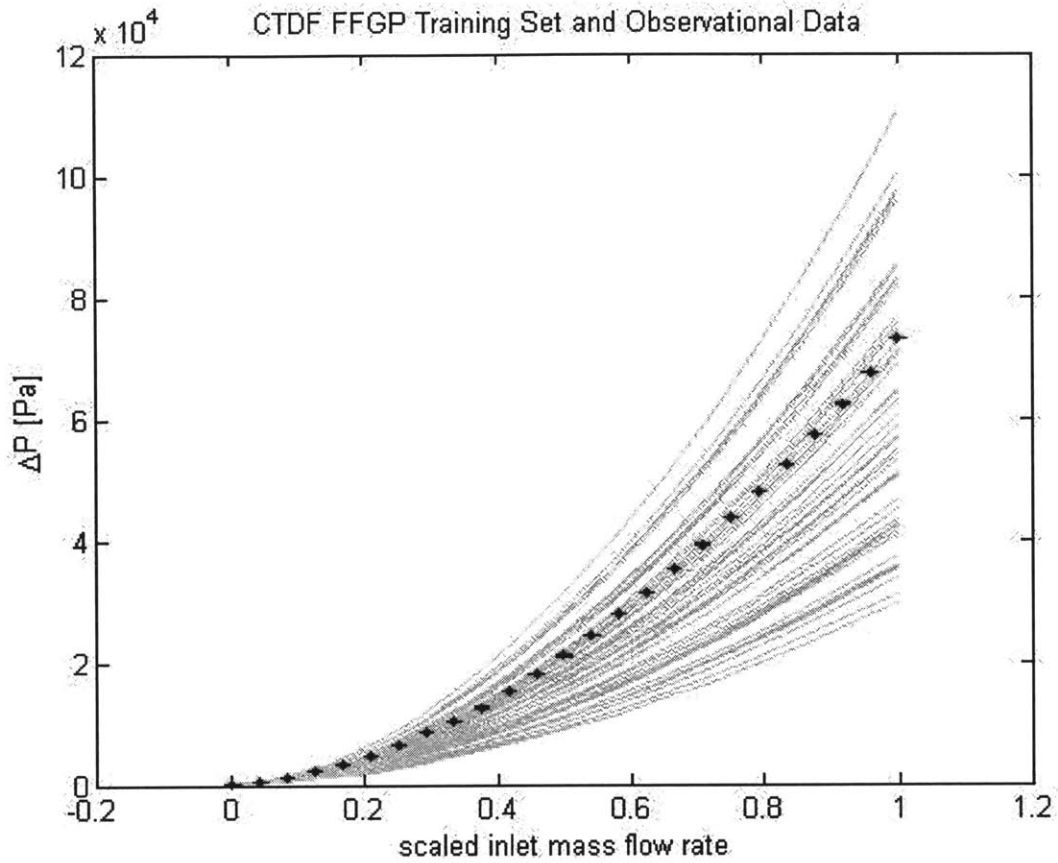


Figure 6-23: CTDF FFGP Training set

Table 6.4: CTDF FFGP emulator build times

FFGP type	total latent burnin time	avg. time per latent burnin iteration	total latent posterior time	avg. time per latent posterior iteration
2-fact 1-comp	54.73 s	$5.473 \times 10^{-4}$ s	36.59 s	$1.825 \times 10^{-4}$ s
2-fact 2-comp	63.47 s	$6.347 \times 10^{-4}$ s	52.98 s	$2.649 \times 10^{-4}$ s
2-fact 3-comp	71.23 s	$7.123 \times 10^{-4}$ s	65.64 s	$3.282 \times 10^{-4}$ s

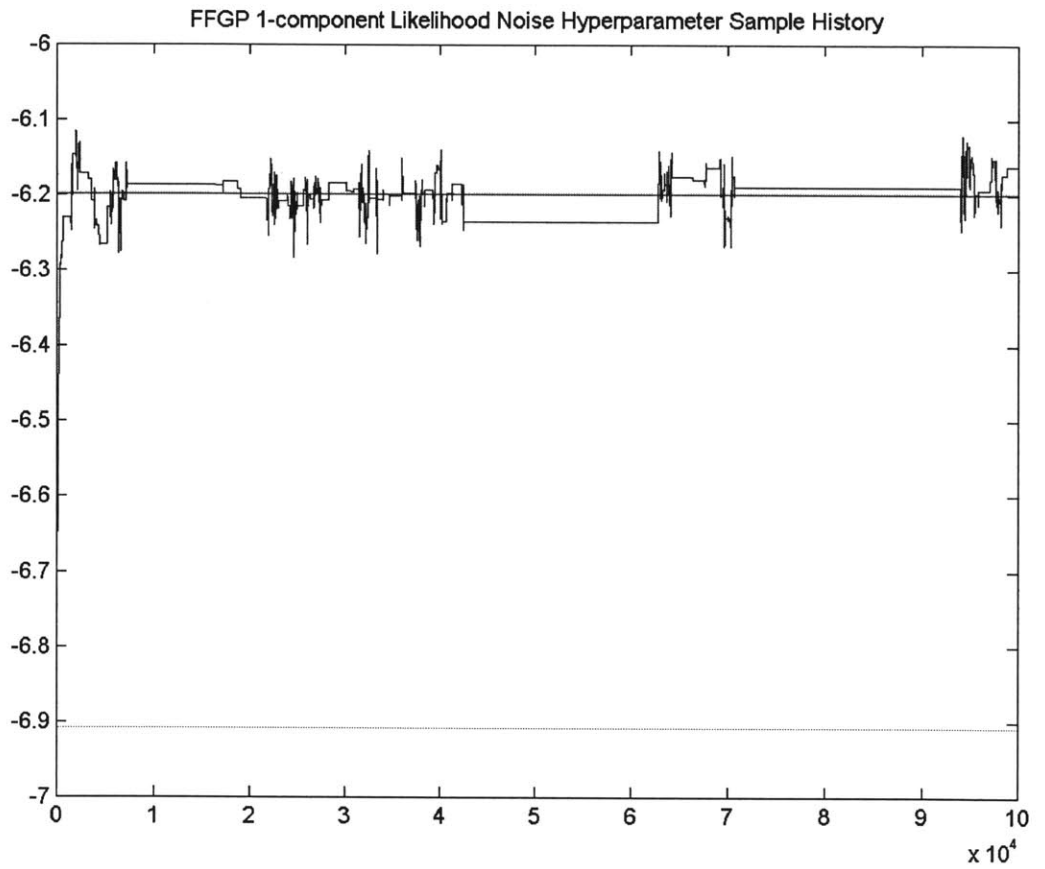


Figure 6-24: CTDF FFGP 2-factor 1-component likelihood noise hyperparameter

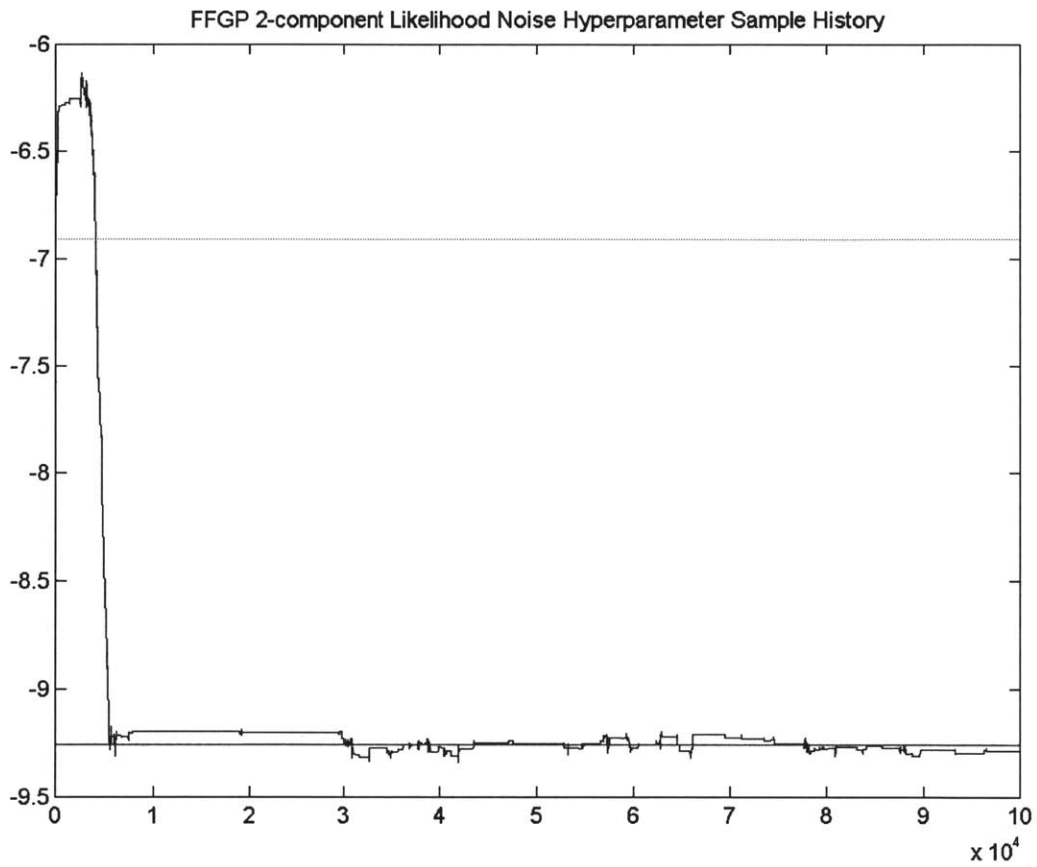


Figure 6-25: CTDF FFGP 2-factor 2-component likelihood noise hyperparameter

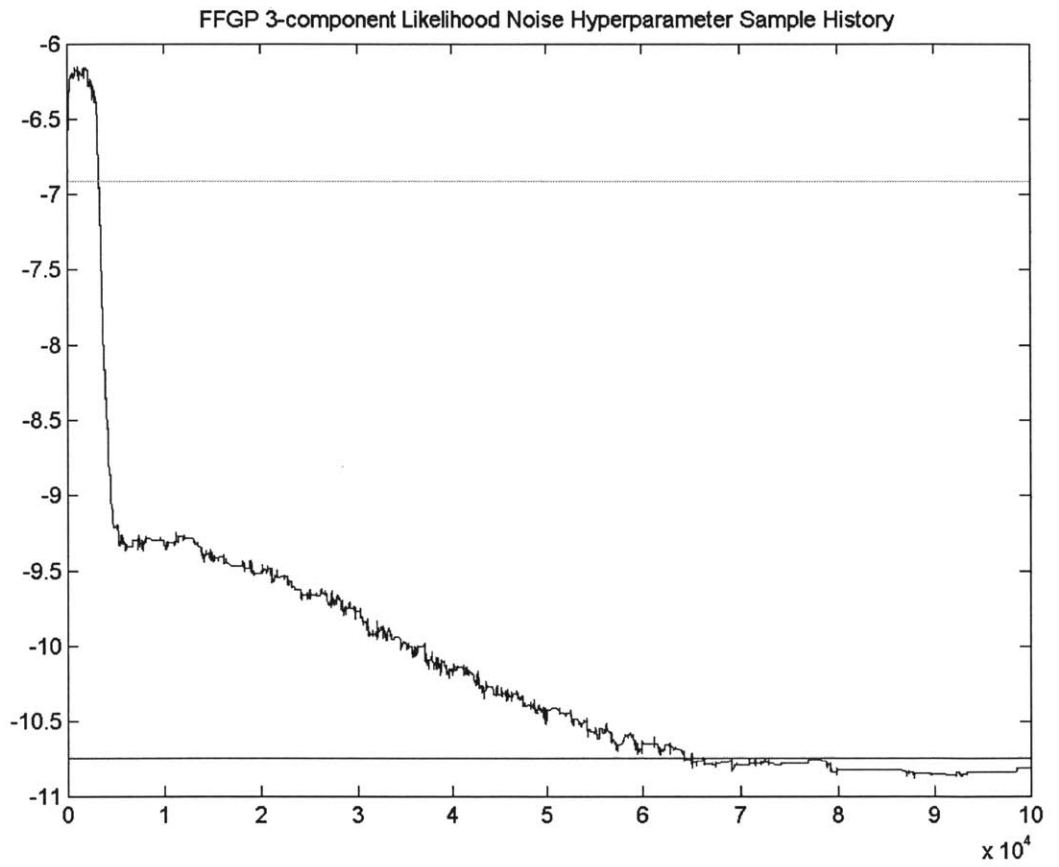


Figure 6-26: CTDF FFGP 2-factor 3-component likelihood noise hyperparameter

Chapter 4 was used. A total of  $10^5$  MCMC samples were made using the AM scheme, just as in Chapter 4. Figures 6-27 through 6-29 show the calibrated posterior predictions and the scaled posterior uncertain parameter histograms, for the 1-component, 2-component, and 3-component emulator-based calibration process. Again, in the plot of the posterior predictions, the blue lines are the posterior quantiles of the posterior predictive means, while the green band is the total predictive variance - including the emulator uncertainty. The posterior uncertain parameter histograms are shown over their scaled ranges. The scaled values of 0 and 1 correspond to the minimum and maximum values given in Table 6.1. I presented the uncertain parameters as their scaled values because it was easier to show them this way in MATLAB. For reference, the scaled prior histograms are shown in green behind the posterior histograms shown in blue.

Given the demonstration problem in Chapter 4, we expected the 1-component FFGP emulator to be very uncertain. Additionally, the likelihood noise hyperparameter training results in Figures 6-24 through 6-26 confirmed that additional components are necessary in order to emulate the CTDF model accurately. The prediction results shown in Figures 6-27 through 6-29 confirm this since the 1-component FFGP calibrated posterior predictions have a very large total predictive uncertainty band compared to the 2 and 3-component FFGP emulators. It makes sense that the additional uncertainty from the 1-component FFGP emulator prevents it from learning more about the uncertain parameters. Comparing the posterior histograms, the 1-component-based results show it is only able to properly identify the posterior on the  $C$ -parameter is somewhere within the lower half of the prior range of values. But what is concerning between the 2-component and 3-component-based results is the  $\Phi_S$ -parameter has completely different posterior distributions. It was expected that since both have very little error relative to the training set they should yield similar

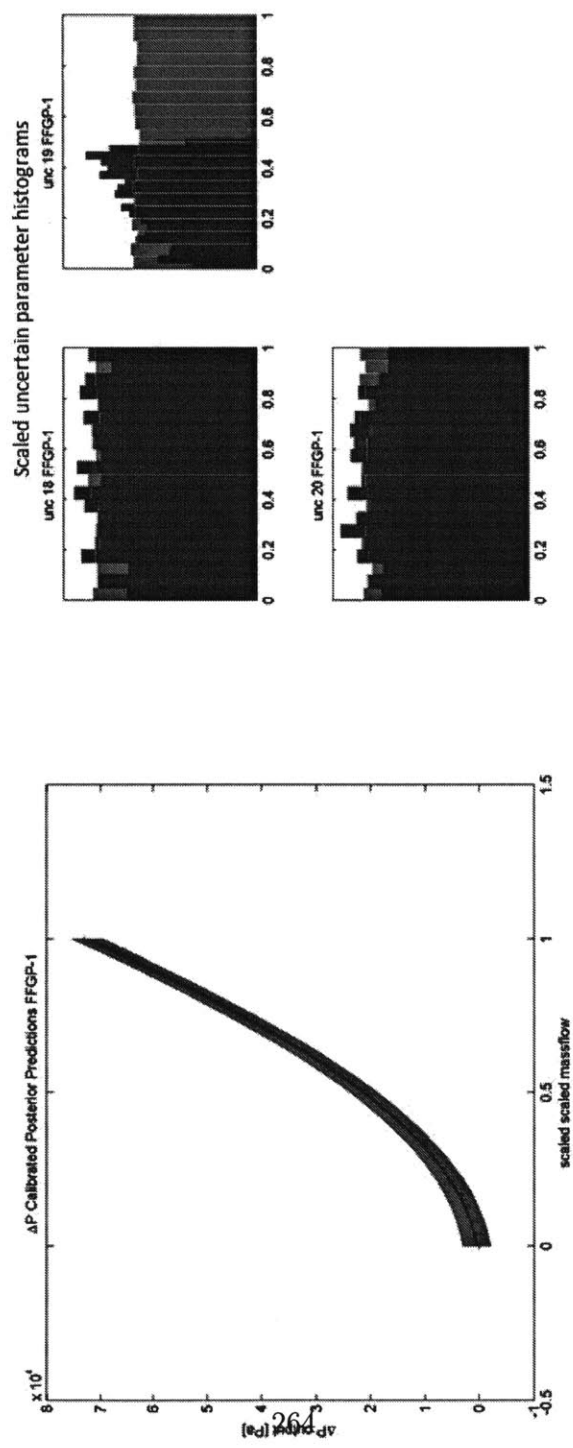


Figure 6-27: CTFD FFGP 2-fact 1-comp-based calibration results



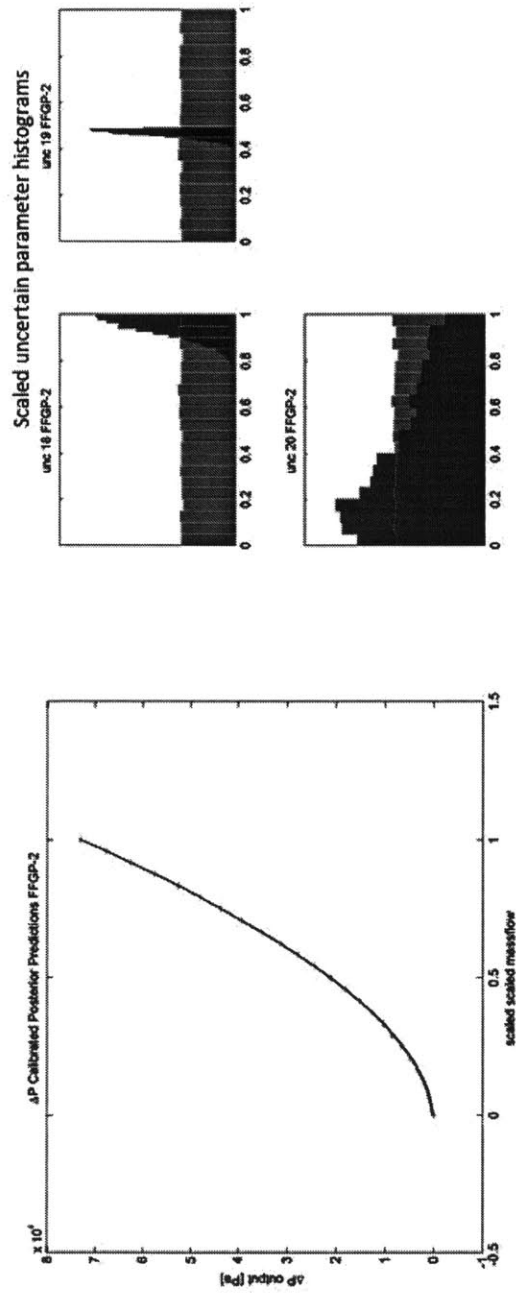


Figure 6-28: CTFD FFGP 2-fact 2-comp-based calibration results

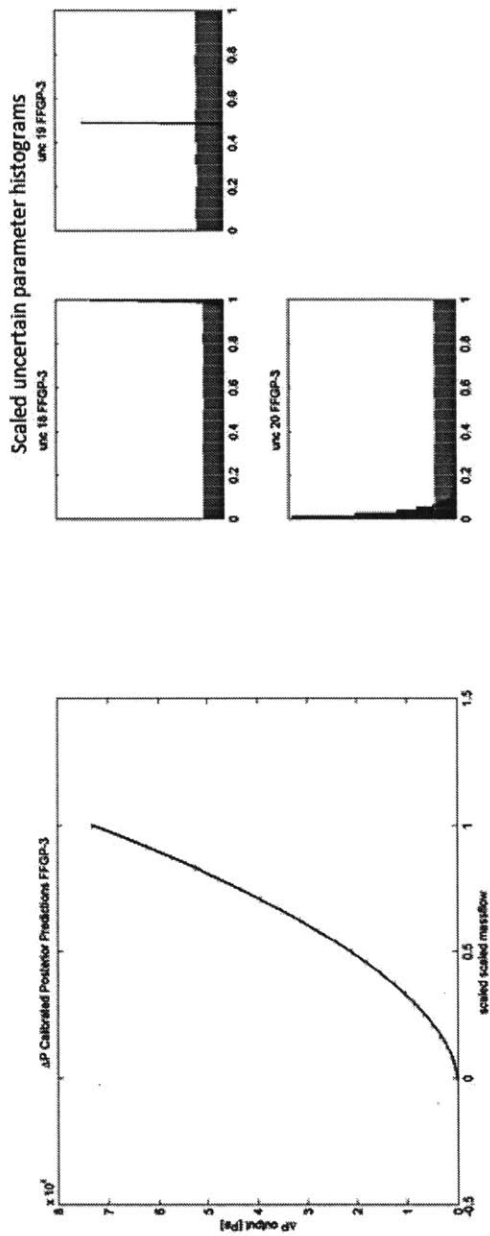


Figure 6-29: CTDF FFGP 2-fact 3-comp-based calibration results

posterior results. However, the 3-component emulator makes it appear that there is still too much additional uncertainty contributed by the 2-component emulator to fully resolve the uncertain parameter posterior distributions.

We are then faced with the issue of choosing which of the posterior distributions are “the best”. The most formal way to answer this is with Bayesian Model Selection which essentially tries to fulfill Bayesian Occam’s Razor by finding the “simplest answer” to a scenario [24]. The best solution - from a Bayesian Model Selection point of view - is the solution that maximizes the marginal likelihood which from Chapter 2 is the denominator in the posterior distribution. But, since we use MCMC sampling we never actually compute the marginal likelihood. In general, computing the marginal likelihood with MCMC sampling is very difficult [7], [9], [?]. Therefore, I will only use an “eye ball” test to decide the “best solution” for now. The section on future work in Chapter 7 will discuss how improvements to the FFGP emulator construction can help get around having to use an “eye ball” test and actually have an estimate to the marginal likelihood.

Figure 6-30 shows the sampled log-joint-posterior values for 2-component and 3-component-based calibration processes. Since the uncertain parameter priors are all uniform distributions over the scaled range of 0 to 1, the posterior essentially equals the likelihood. The exact way of computing the marginal likelihood is to then compute the average joint-posterior value from the large number of samples. The “eye ball” test simply looks at the posterior samples of the log-joint-posterior to see which appears to on average have the largest value. The 2-component “eye ball” average log-joint-posterior value is around 195, while the 3-component “eye ball” average log-joint-posterior value is around 145. To compare models, Bayes Factors are computed which take the ratio of the marginal likelihood from one model to another. If that ratio is large, the model in the numerator is considered to be better,

or more formally explain the data better, than the model in the denominator [4]. The “eye ball” Bayes Factor between the 2-component and 3-component based results is therefore on the order of  $10^{21}$ . Thus, the 2-component FFGP emulator is better than the 3-component emulator and we should take the 2-component FFGP emulator-based results as “better” or “more correct” than the 3-component FFGP emulator based results. This may seem counter-intuitive, since the 3-component emulator has more flexibility than the 2-component emulator. Bayesian model selection, however, penalizes models with too much complexity. It is essentially a Bayesian alternative to “overfitting” since a model with too many degrees of freedom can have a large number of possible combinations that still match the data. The “simplest solution” is therefore a balance between the accuracy and the degrees of freedom (the complexity) within a model. The 3-component emulator is “overfit”, in a Bayesian sense, to the training data preventing it from generalizing as well as the (slightly) simpler 2-component emulator.

A key aspect of using the emulator-based approach is to make sure there is considerable computational savings from building the emulator and applying the emulator to calibrate the uncertain parameters. Table 6.5 below summarizes the computational expense of the emulator-based calibration process. The major take away from Table 6.5 is that the FFGP emulators are almost an insignificant amount of time compared to just creating the training set. Once that training set is created the training the emulators for the CTDF model is fast and then using the emulators to calibrate the CTDF model is even faster. In the time it takes to build and apply the slowest FFGP emulator (the 3-component emulator) 100 RELAP runs of the CTDF could be made. That number is just for the approximately 2 second run time of a single CTDF RELAP case run. As stated before, if the CTDF RELAP model was used directly in the MCMC sampling, the CTDF RELAP model must be run 25

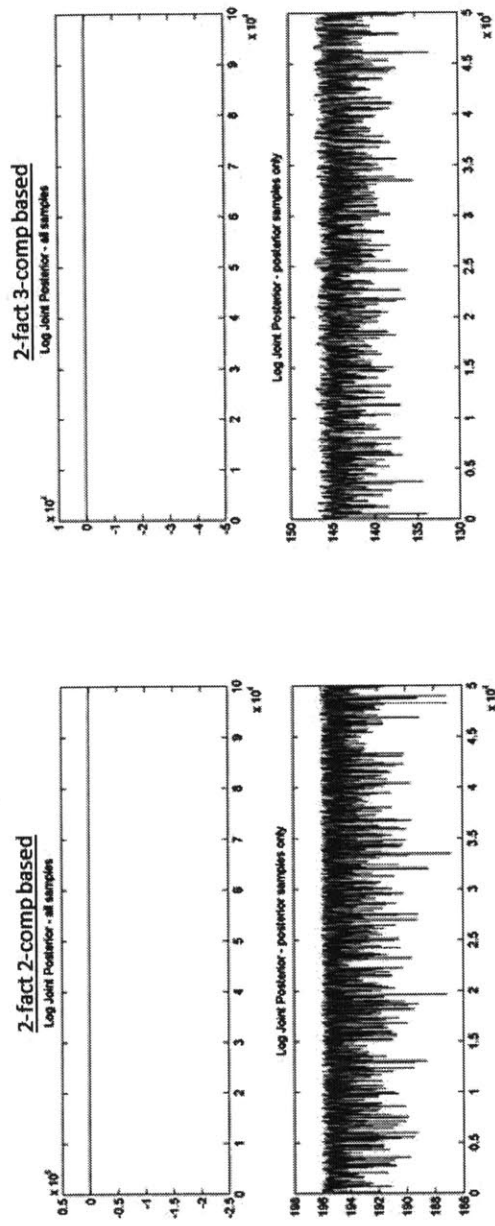


Figure 6-30: CTDF FFGP Log-Joint-Posterior comparison

separate times to compare to the 25 “data” points. If parallel computing resources were available to run on 25 separate processors then this would only take a little over 2 seconds per MCMC sample. Each MCMC iteration with the slowest FFGP emulator is over 3000x faster than the parallel computing direct simulator-based approach. Or, as in how I have run RELAP, if each RELAP run was made in series a single MCMC iteration would take roughly 50 seconds to complete. The slowest emulator is therefore over 75000x faster than the in-series simulator-based approach.

Table 6.5: CTFD Uncertain parameter calibration computational times

FFGP type	Avg time per MCMC sample with the emulator	Total emulator-based calibration time (including time to create training set)	Effective number of RELAP runs in the same amount of time
2-fact 1-comp	$4.199 \times 10^{-4}$ s	133.31 s (1633.31 s)	66 (816)
2-fact 2-comp	$5.711 \times 10^{-4}$ s	173.56 s (1673.56 s)	86 (836)
2-fact 3-comp	$6.496 \times 10^{-4}$ s	201.83 s (1701.83 s)	100 (850)

### Core Channel G&G-1973

The Core Channel G&G-1973 (GGcore) pseudo SET has a slightly different RELAP model than the generic pseudo SET RELAP model shown in Fig. 6-22. This is only because as shown in Fig. 6-9, the core channel is split into 3 sections, the lower blanket (LB), driver fuel (DF), and upper blanket (UB). The RELAP component 400 in Fig. 6-22 is therefore replaced by the core channel layout used in the IET

model directly. This means the LB/UB friction parameters are also included within the GGcore RELAP model. The GGcore SET setup and application is identical to the CTDF SET discussion above, but it simply includes more uncertain parameters.

As described earlier as well, the purpose of the GGcore SET is to include the inlet nozzle loss coefficient, since the G&G-1973 data is the pressure drop across the entire channel. The GGcore RELAP model therefore has a total of 7 uncertain parameters. Their corresponding IET uncertain parameter numbers are: 18, 19, 20 (the DF friction parameters), 15, 16, 17 (the LB/UB friction parameters) and 8 (the core channel inlet nozzle loss coefficient). The number of case runs in the training set was increased to 100 to accommodate more uncertain parameters, but the number of control variable locations was maintained at 15. A single RELAP run took ~3 seconds now instead of the ~2 seconds for the CTDF RELAP model. So it took rough 75 minutes to create the entire GGcore FFGP training set which is shown in Fig. 6-31. The observational “data” taken from the G&G-1973 reference [49] is shown as red error bars using the same assumptions on the observational error as used for the CTDF “data”.

The procedure of building various FFGP emulators and applying them to calibrate the uncertain parameters was the same as described for the CTDF RELAP model. The log-joint-posterior “eye ball” test was also used to determine which emulator was the “best”. To save space, only results for the “best” FFGP emulator will be shown. Table 6.6 provides the computational times required to build each of the GGcore FFGP emulators.

The 2-factor 3-component FFGP emulator was considered to be the “best”. Its likelihood noise hyperparameter training results are shown in Fig. 6-32 and its calibrated posterior results are shown in Fig. 6-33. As expected only the inlet nozzle loss coefficient is learned from the G&G-1973 data. The calibrated posterior

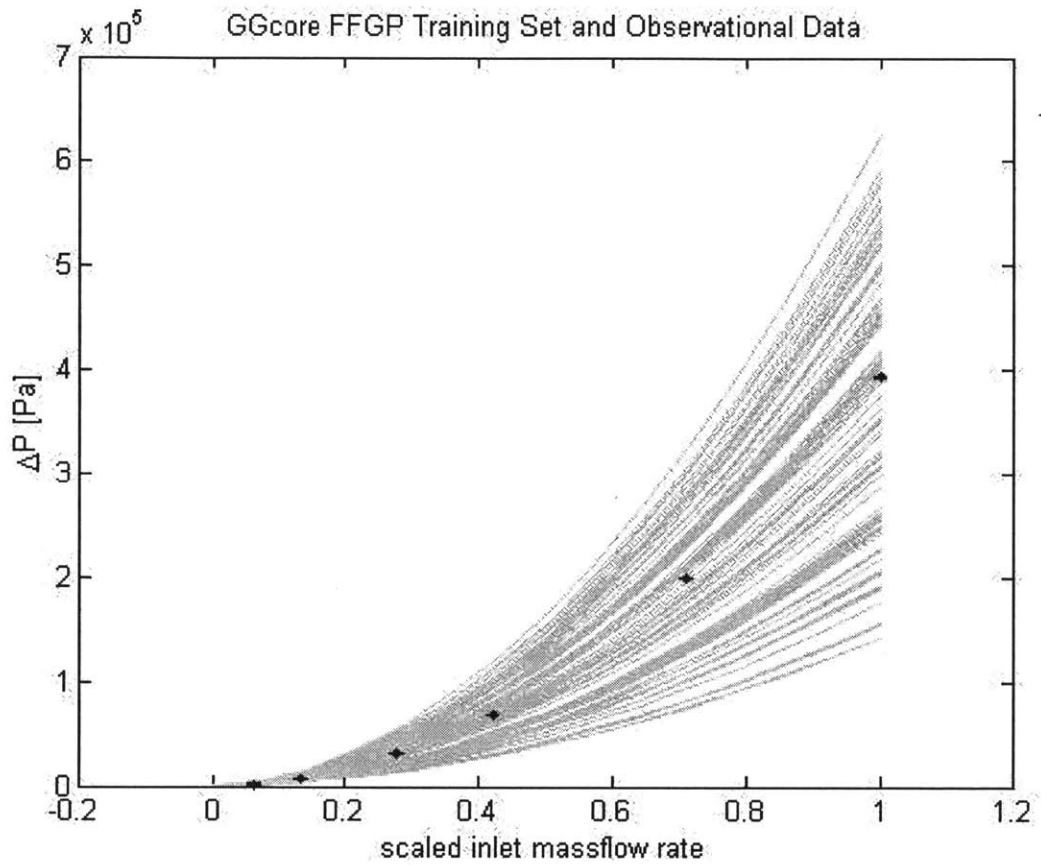


Figure 6-31: GGcore FFGP training set

Table 6.6: GGcore FFGP emulator build times

FFGP type	total latent burnin time [s]	avg. time per latent burnin iteration [s]	total latent posterior time [s]	avg. time per latent posterior iteration [s]
2-fact 1-comp	85.87	$8.587 \times 10^{-4}$	46.60	$2.330 \times 10^{-4}$
2-fact 2-comp	92.43	$9.243 \times 10^{-4}$	66.19	$3.31 \times 10^{-4}$
2-fact 3-comp	105.66	$1.057 \times 10^{-3}$	85.55	$4.227 \times 10^{-4}$
2-fact 4-comp	114.24	$1.1 \times 10^{-3}$	102.97	$5.149 \times 10^{-4}$



predictions have a larger amount of emulator uncertainty compared to the CTDF calibrated posterior predictions. The friction parameters exhibit bimodal posterior distributions, with modes at the bounding extremes of their priors. Originally, this was slightly surprising to see. The uniform priors do not bias the parameters to take any particular set of values, over the prior range. With a majority of the data variability explained by the inlet nozzle loss coefficient, the posterior friction posterior values simply allow the emulator predictions to match the G&G-1973 data. The GGcore model cannot “see” the specific physics dictating the pressure drop across the LB/UB and DF subassemblies. It only sees the channel wide response which is why it makes sense that the inlet nozzle loss coefficient (IET uncertain parameter 8) is learned the most from the data. It should be noted that there are only 6 observational data points in the GGcore SET, compared to the 25 data points in the CTDF SET. More data points would have allowed the inlet nozzle loss coefficient to be inferred with even higher precision.

The computational times required to perform the uncertain parameter MCMC sampling with each of the FFGP emulators are shown in Table 6.7. In order to achieve better mixing rates, the emulator-based uncertain parameter calibration needed to be run a total of  $2 \times 10^5$  (with the first half discarded as burn-in), compared to the  $10^5$  total samples used previously. Even though more samples were required the time to create the GGcore training set still dominates the total emulator-based calibration execution time. The slowest emulator the 2-factor 4-component FFGP emulator was not considered to be better than the 2-factor 3-component emulator, due to the Bayesian model selection “eye ball” test described previously. But even this complex, and relatively slow emulator takes the equivalent of 80 runs of the GGcore RELAP model. Creating the GGcore training set takes roughly 95% of the total emulator-based calibration time. Thus, again the emulator computational

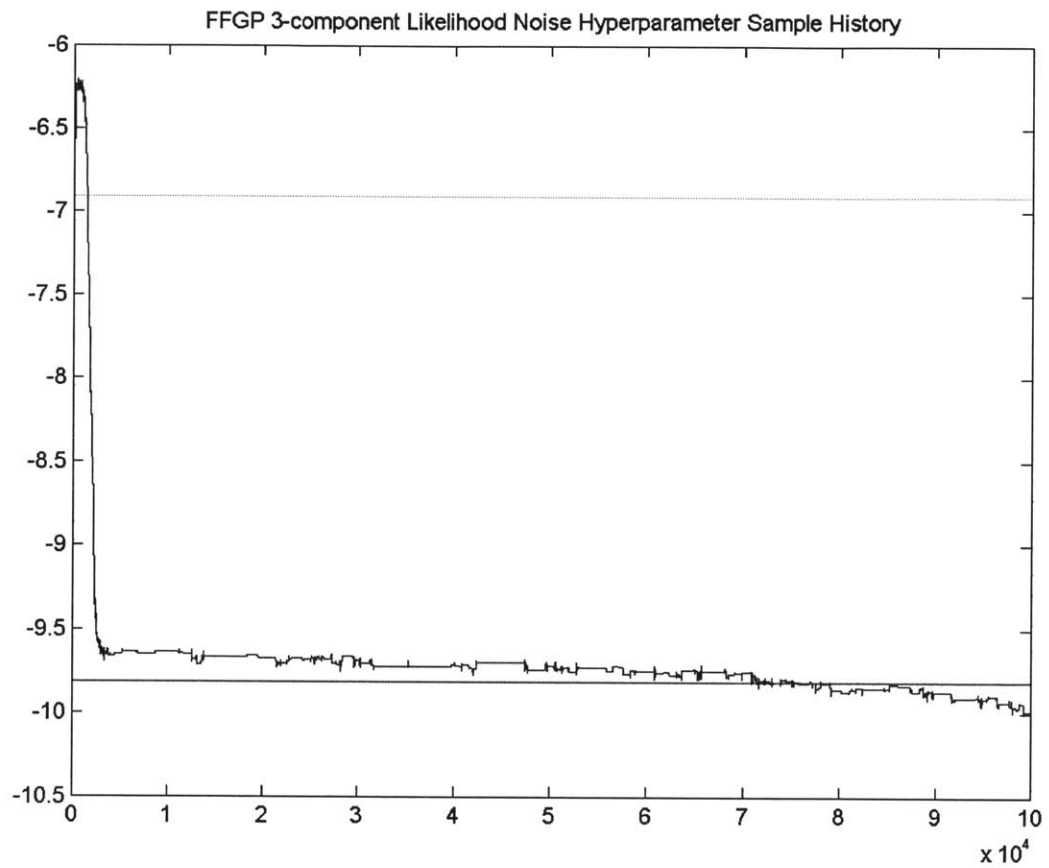


Figure 6-32: GGcore FFGP 2-fact 3-comp likelihood noise hyperparameter

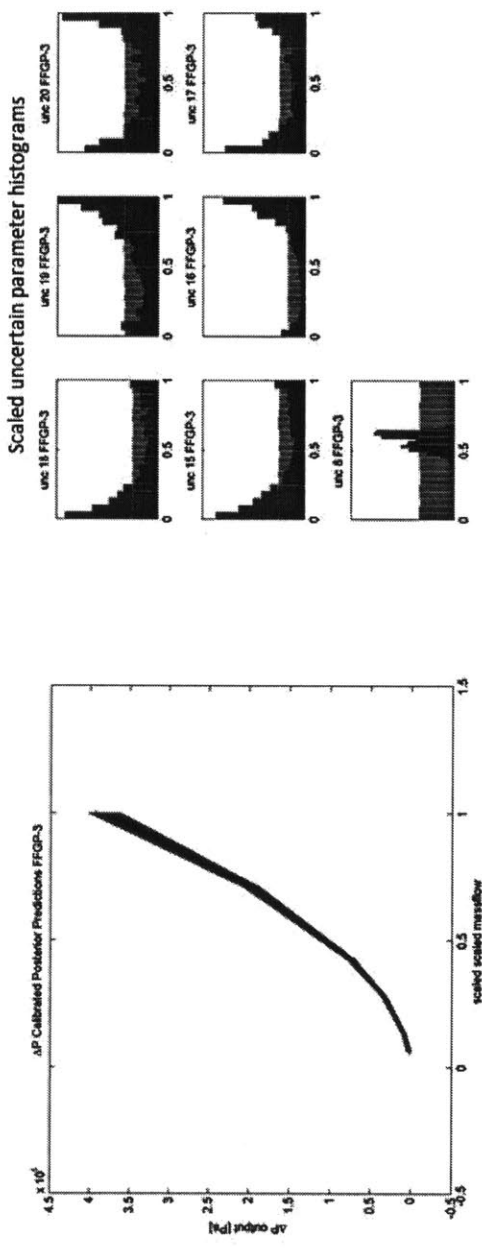


Figure 6-33: GGcore FFGP 2-fact 3-comp-based calibration results

demand is effectively negligible compared to running the RELAP model.

Table 6.7: GGcore Uncertain parameter calibration computational times

FFGP type	Avg time per MCMC sample with the emulator	Total emulator-based calibration time (including time to create training set)	Effective number of RELAP runs in the same amount of time
2-fact 1-comp	$5.075 \times 10^{-4}$	101.51 (4601.5)	33 (1533)
2-fact 2-comp	$6.162 \times 10^{-4}$	123.25 (4623.2)	41 (1541)
2-fact 3-comp	$7.341 \times 10^{-4}$	146.82 (4646.8)	48 (1548)
2-fact 4-comp	$1.2 \times 10^{-4}$	240.00 (4740)	80 (1580)

### 6.4.3 Blanket SETs

#### IB/OB Cheng & Todreas SET

The IB/OB Cheng & Todreas (IBCT) pseudo SET computes the friction factor using the Cheng & Todreas bare-rod bundle friction factor correlation. Unlike the wire-wrapped bundle average friction factor factor, the bare-rod bundle version required computing an averaging process of all the subchannels within the IB subassembly bundle [46]. But, other than that difference, the IBCT pseudo SET setup is identical to the CTDF pseudo SET setup. The friction factor is then used to compute the  $\Delta P$  across the IB subassembly length(which is identical to the OB subassembly bundle as described previously). The length is just the length of the bare-rods within the hex

can and so that any additional frictional losses are not included within this pseudo SET.

With 3 uncertain parameters, 50 case runs were used to create the training dataset. But, after trial and error, I found that using 25 control variable locations, instead of 15 provided better FFGP emulator results for the blanket SETs. The IBCT FFGP training set and observational data are shown in Fig. 6-34.

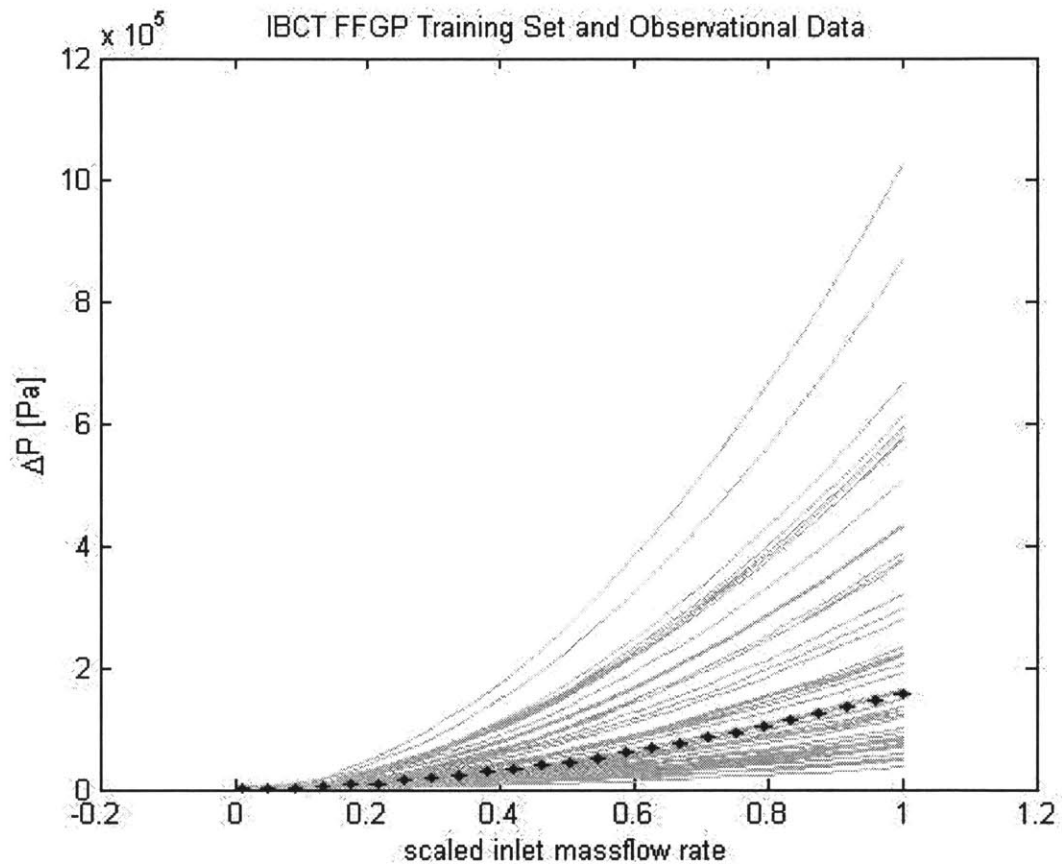


Figure 6-34: IBCT FFGP training set

The likelihood hyperparameter for the FFGP 2-factor 3-component emulator of

the IBCT is shown in Fig. 6-35. This was the “best” emulator as determined by my “eye ball” Bayesian Model Selection criteria and will be the only results shown for the IBCT model. Once built, the FFGP 2-factor 3-component emulator was used to calibrate the uncertain parameters in the IBCT RELAP model, using the FFGP-modified likelihood with the AM-MCMC scheme. The emulator based calibration results are shown in Fig. . The IB/OB friction parameters’ corresponding IET uncertain parameter numbers are: 22, 23 and 24. The posterior IB/OB friction parameters’ histograms look far more Gaussian-like than the DF friction parameters’ posterior histograms. One important thing to note about the IBCT SET compared to the CTDF SET is that the inlet mass flow rate for the IBCT SET is lower than that of the CTDF SET inlet mass flow rate. The core channel (and thus the driver fuel subassembly) receives a majority of the high pressure stream and so has a much higher mass flow rate through it than the IB channel. I present all of the results using the scaled inlet mass flow rate because it was simply easier to use in MATLAB a quantity between 0 and 1. But the max scaled inlet mass flow rate (a value of 1) for the IBCT SET was much smaller than the max scaled inlet mass flow for the CTDF SET. Therefore, the IBCT SET has more data points that fall within the transition and laminar flow regime, than the purely turbulent flow regime compared to the CTDF SET. That suggests why IB/OB laminar shape factor (IET uncertain parameter number 23) has a much larger posterior mode value than the posterior DF laminar shape factor as shown in Fig. 6-28. This also explains why the blanket SETs required more control variable locations than the core channel SETs, since the transition flow regime friction factor is an interpolation between the laminar and turbulent friction factors and is therefore a more complicated function.

The emulator-based calibration computational times for the IBCT model are very similar to the computational times for the CTDF model. Although the IBCT emula-

tors required more factor-1 (the control variable factor) training points, the factor-1 size does not impact the computational speed that much.

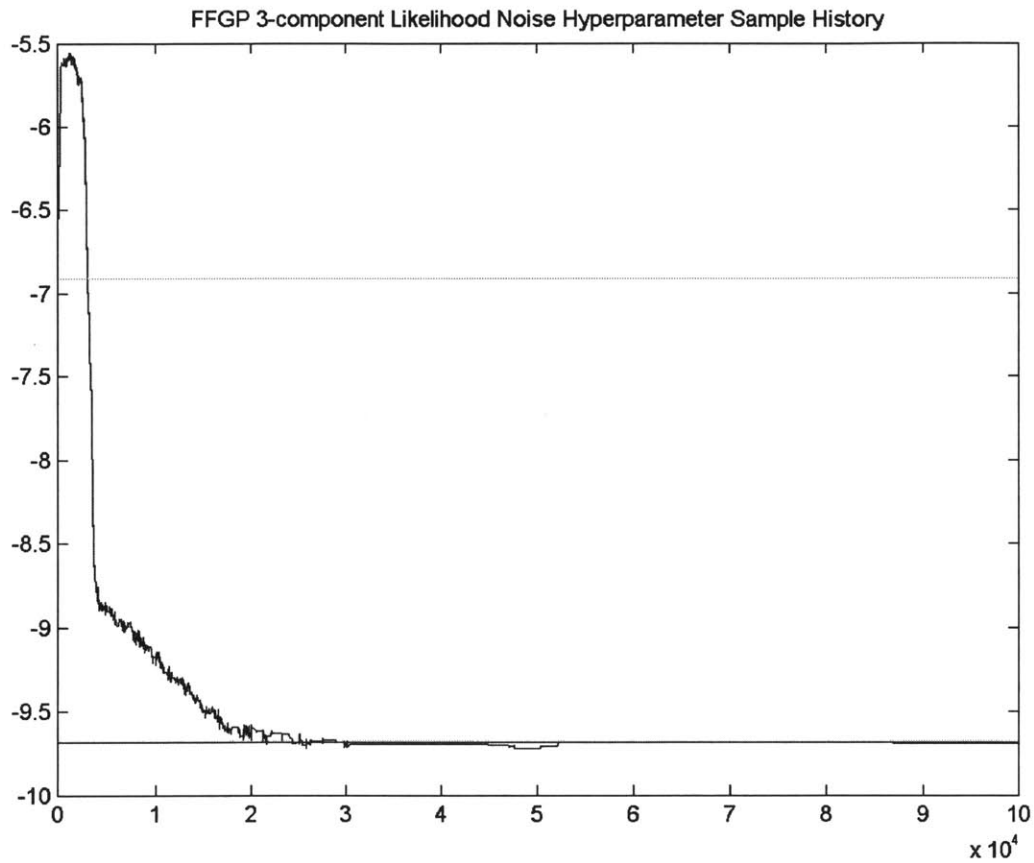


Figure 6-35: IBCT FFGP 2-fact 3-comp likelihood noise hyperparameter

### IB G&G-1973

The IB G&G-1973 (IBGG) pseudo SET is similar to the GGcore SET described previously. But, as expected the G&G-1973 data pertains to the IB channel rather

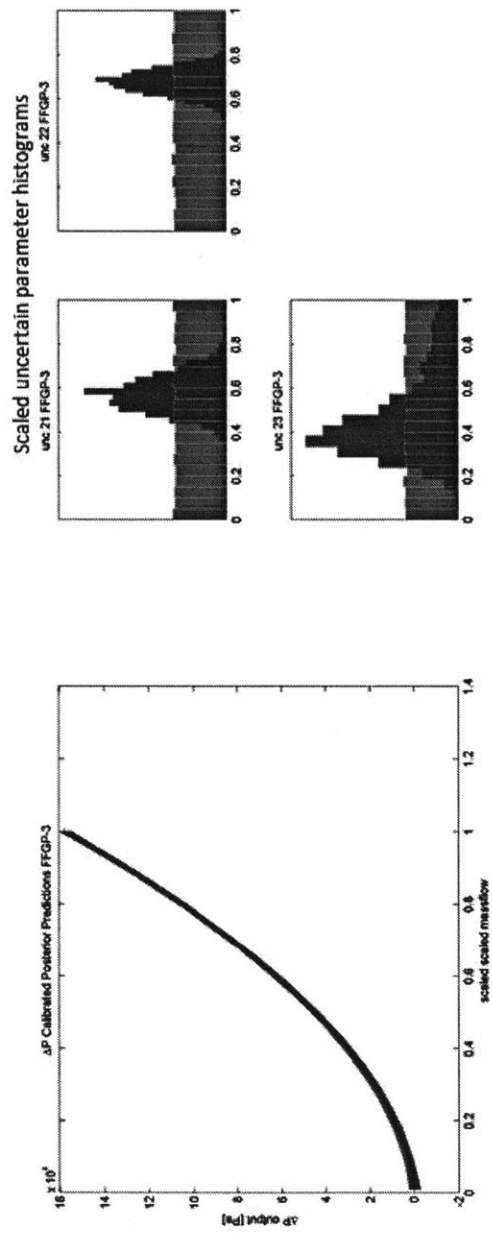


Figure 6-36: IBCF FFGP 2-fact 3-comp-based calibration results



than the core channel. Since the IB channel only has the IB subassembly in it, the only additional uncertain parameter is the IB channel inlet nozzle loss coefficient, IET uncertain parameter number 9. The IBGG training set is shown in Fig. 6-37 which had 75 case runs along with 25 control variable locations. The red error bars in Fig. 6-37 are the G&G-1973 observational data points.

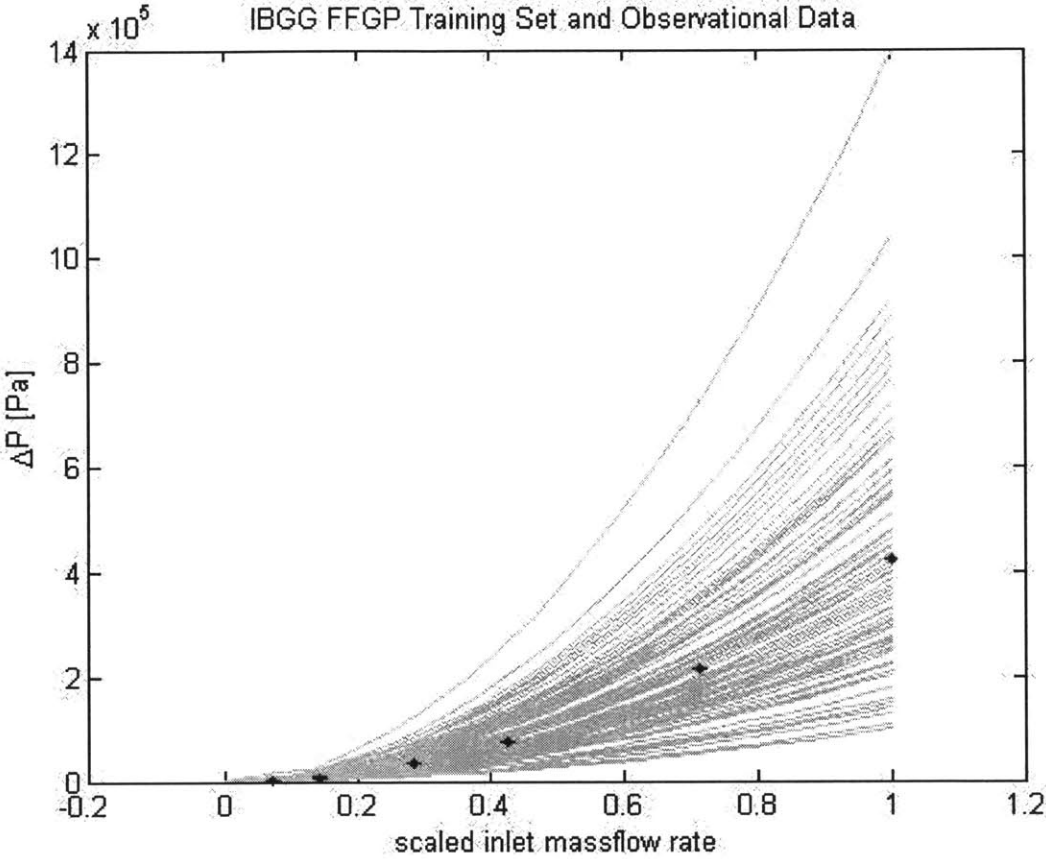


Figure 6-37: IBGG FFGP training set

The “best” FFGP emulator for the IBGG training set was the FFGP 2-factor 3-component emulator. The corresponding likelihood hyperparameter for the FFGP

2-factor 3-component emulator is shown in Fig. 6-38 and the emulator-based calibration results are shown in Fig. 6-39. The most striking feature of the uncertain parameter posterior histograms in Fig. 6-39 are how different the IB/OB friction parameter posteriors are from their corresponding posterior results from the IBCT SET. The question of which SET is “more correct” will be answered in the section on simultaneous calibration later on. The IBGG emulator computational times are also similar to their core channel counterparts, the GGcore emulators.

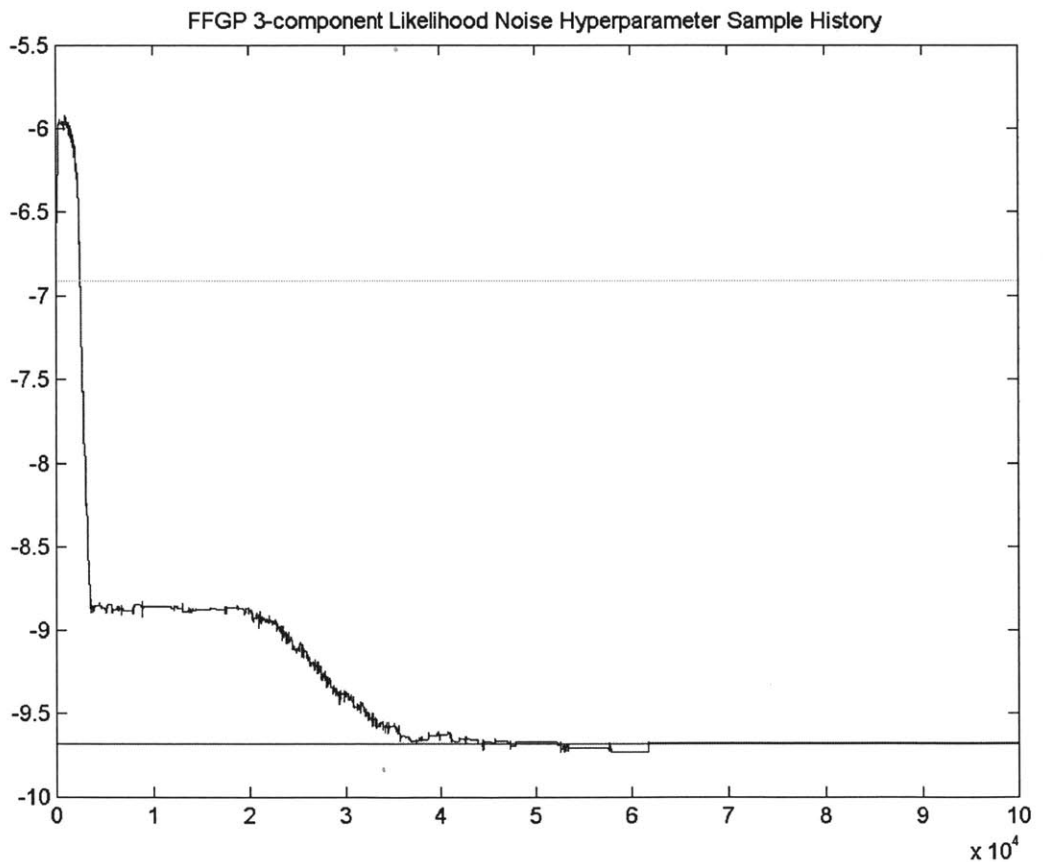


Figure 6-38: IBGG FFGP 2-fact 3-comp likelihood noise hyperparameter

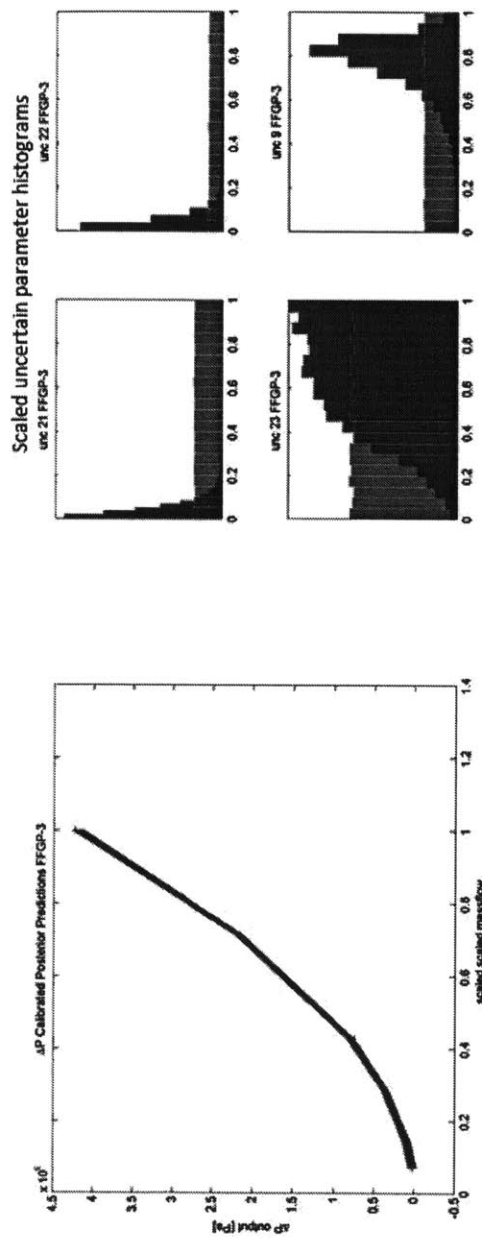


Figure 6-39: IBBG FFGP 2-fact 3-comp-based calibration results

## OB G&G-1973

The OB G&G-1973 (OBGG) pseudo SET has more in common with the GGcore SET than the IBGG SET did. The OBGG RELAP model includes the OBLA tubes beneath the OB subassembly channel, and so includes the 3 OBLA friction parameters. With the OBLA inlet nozzle loss coefficient, the OBGG RELAP model has 7 uncertain parameters, whose corresponding IET uncertain parameter numbers are: 21, 22, and 23 (IB/OB friction parameters), 24, 25, and 26 (OBLA friction parameters), and 14 (OBLA inlet nozzle loss coefficient). The OBGG training set uses 100 case runs, like the GGcore training set, and uses 15 control variable locations like the core channel SETs. The OBGG FFGP training set and G&G-1973 data are shown in Fig. 6-40.

The “best” FFGP emulator for the OBGG training set was also the FFGP 2-factor 3-component emulator. The corresponding likelihood hyperparameter and emulator-based calibration results are shown in Figures 6-41 and 6-42. The OBGG SET observational error is larger than for the other SETs, because I still use the same procedure as described in the CTDF section where I assumed 95% of the observational data probability falls with  $\pm 33\%$  around the data mean value for the first data point. The first data point for the OBGG SET data has a larger  $\Delta P$  than the first data points in the other SETs due to data points that I chose from Fig. 6-21. The scaled inlet mass flow rate was also not able to be as low of a fraction of the nominal OB channel flow rate (the low pressure flow rate) compared to the minimum flow rate values in the other SETs due to the G&G-1973 data set. The mass flow rate values are then still within the transition flow regime and not purely laminar flow. Which explains why the laminar shape factor posteriors are not changed all that much relative to the priors for the OBGG SET.

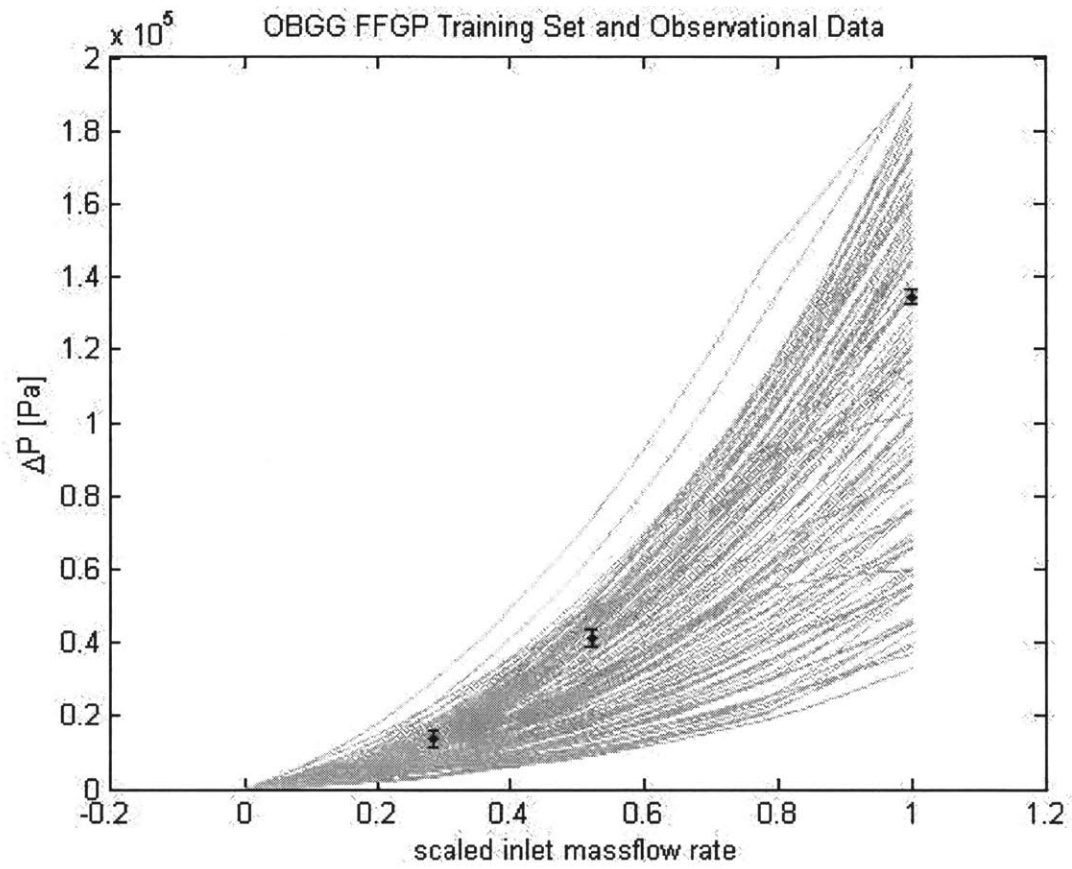


Figure 6-40: OBGG FFGP training set

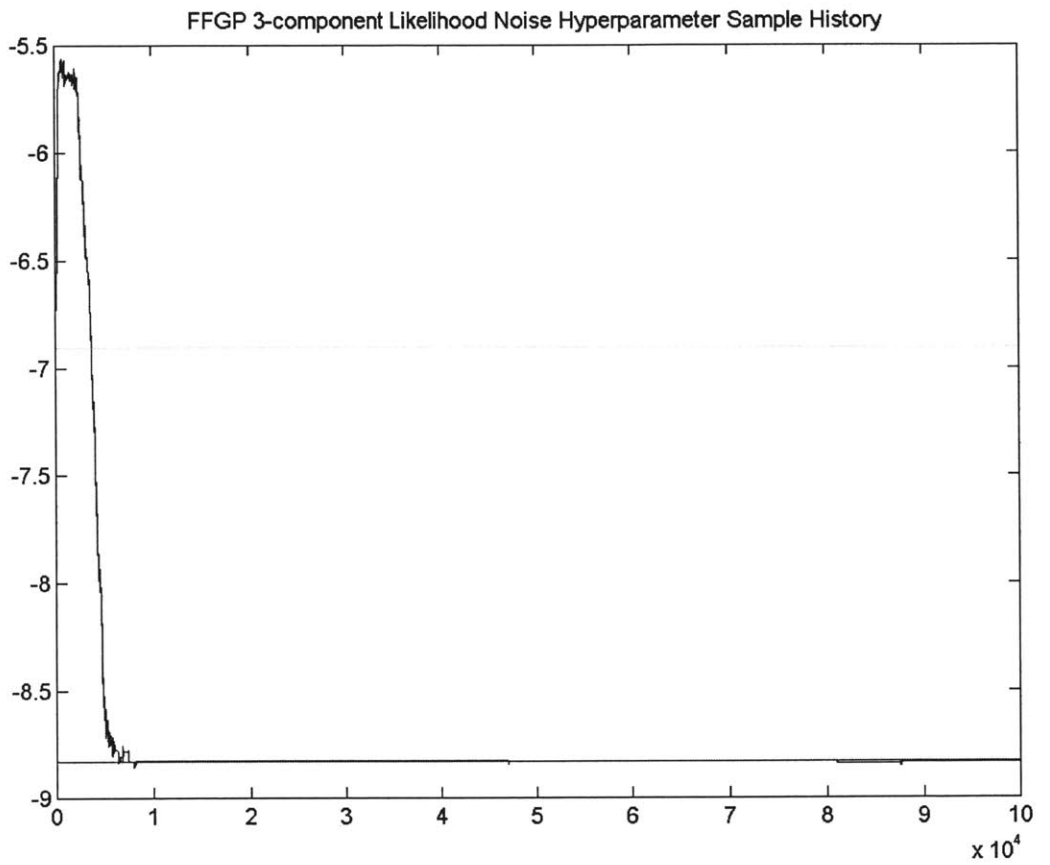


Figure 6-41: OBGG FFGP 2-fact 3-comp likelihood noise hyperparameter

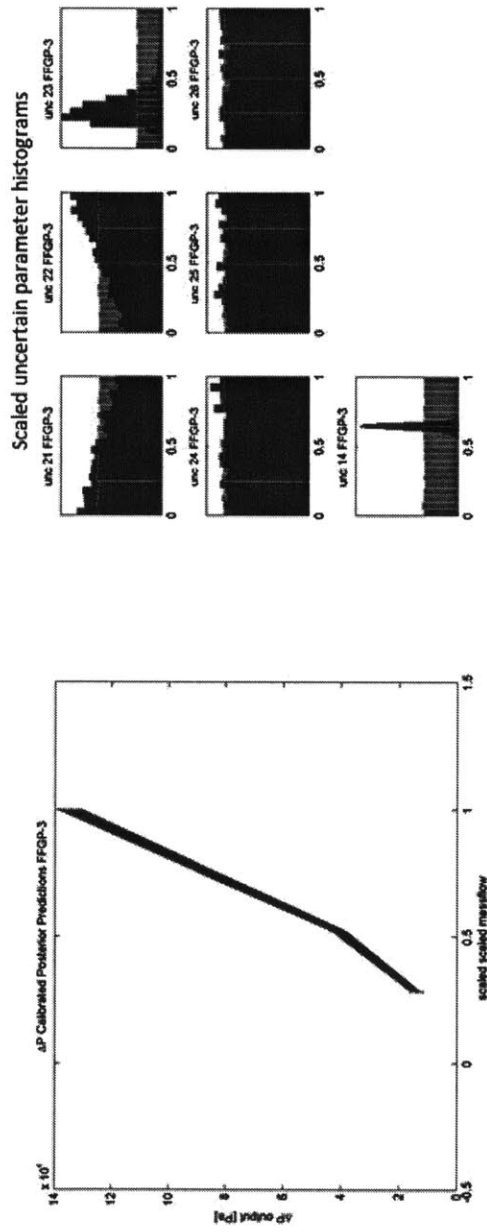


Figure 6-42: OBGG FFGP 2-fact 3-comp-based calibration results

## 6.5 IET Output Calibration Results

With the SET model calibrations completed, the IET RELAP model calibration was performed. Unlike the pseudo SETs however, the IET data is real data taken from the EBR-II Transient Test No.10 Phase 2, as described earlier. Three types of data are given in Baumann et al.: the low pressure flow through into the low pressure plenum, labeled as LPP flow; the top-of-core temperature, denoted TTC; and the core-outlet temperature, denoted OTC. Even though several references detailed the estimated measurement error within the XX08 instrumented subassembly [50], simply reading Transient Test No.10 measurement data off of Figures 6-6 through 6-8 constituted the largest source of error for the observational data. Therefore the IET observational error value is really just my own uncertainty in reading the figures since tabulated data was not given. The OTC and TTC errors were set to have 95% of the data probability to fall within  $\pm 3.5^{\circ}\text{C}$  around the data mean. The LPP flow measurement error was much harder to set. Reading Fig. 6-8 is very difficult, but I wanted to make sure the assumed measurement error was not large enough to allow the LPP flow rates to be positive during the second half of the transient. Therefore, I solved for the measurement error so even 2 standard deviations away from the mean gives a negative LPP flow value. I kept that measurement error value constant for all of the LPP flow data points, which meant LPP flow data set was much more precise compared to the OTC and TTC observational datasets.

In order to create the training sets for each output, fluid cell locations had to be chosen to correspond with the with the XX08 OTC and TTC thermocouple positions. Based upon the thermocouple locations in Ref. [50] the OTC thermocouple corresponded nearly to the cell center of the last cell in the UB section of the core channel. Also from Ref. [50] the TTC thermocouple position was very close to the



cell center of the next to last cell within the DF section of the core channel. The LPP flow was easier because this was just the low pressure stream flow rate and was taken at the junction connecting the low pressure piping to the low pressure lower plenum.

With 41 uncertain parameters a large number of RELAP cases needed to be run. Since this is a transient simulation a single case run produces the predictions at various points in time. I chose to use 500 case runs and 51 control variable locations (points in time) taken per case. The uncertain parameter values for the 500 case runs were determined using the MATLAB latin hypercube sampling function 'lhsdesign', with time included as an input. This allowed a standard GPR emulator to be built for each of the IET model output types as well. But of the 500 case runs 10 crashed, therefore 490 training cases were used, and so a total of 24,990 FFGP training points were used for each of the IET output types. A single IET RELAP model case run took between 40 and 42 seconds. The time difference I think had to do with the time-stepping scheme which might have needed to reduce to reduce the time step size depending on the values of the uncertain parameters for that particular case. But creating the training set for the IET model therefore took roughly 5.5 hours.

Building the emulators with these many training points was not a trivial task. During the latent burn-in phase of the FFGP training process, each MCMC sample required inverting 2 covariance matrices: factor 1 was size  $51 \times 51$  and factor 2 was size  $490 \times 490$ . Once the hyperparameter point estimates are found at the end of the latent burn-in phase, the latent posterior sampling phase is still quite complex since depending on the number of components used, thousands of latent variables must be sampled. With 3 components, factor 1 has  $51 \times 3=153$  training latent variables and factor 2 has  $490 \times 3=1470$  training latent variables. Without the Hamiltonian Monte Carlo (HMC) MCMC algorithm sampling a latent variable space of that size

with potentially very correlated latent variables would be almost impossible.

Only the “best” emulators will be shown in the following subsections for each of the IET outputs. The FFGP emulator-based calibration results will be compared to the GPR emulator-based calibration results for each output as well. Table 6.8 below summarizes various emulators built for each of the IET outputs. The “minimum number of components” column means that FFGP emulators with fewer components were not built for the corresponding IET output type. This saved time since the emulator training, as will be described below, was not trivial for the IET outputs. Table 6.9 summarizes the emulator-based calibration computational times for each of the IET outputs using their corresponding “best” emulators. The following subsections will describe each of the emulator-based calibration results in detail. Due to more complex nature of emulating the IET output, more samples were needed compared to the SET emulators. The total emulator-based calibration times include the time to create the training set in parentheses, but the training set does not need to be rerun for each of the outputs. Each RELAP run produces all three IET outputs. Thus the total emulator-based calibration computational time (including creating the training sets, building the “best” emulators for each of the IET outputs, and performing uncertain parameter calibration with each of the “best” emulators) took 526 minutes (approximately 8.77 hours), or the equivalent of running the IET RELAP model only 789 times.

### 6.5.1 LPP flow results

The LPP flow FFGP training set is shown in Fig. 6-43. The RELAP output was taken at 1 second intervals, so there are actually 201 LPP flow RELAP predictions. But, as stated already, I only used 51 time locations as for the control variable factor,

Table 6.8: IET output emulator type summary

Output	Output full name	Number of factors	Min. number of components	Max. number of components
LPP	Low pressure flow rate	2	2	3
TTC	Top-of-core Temp.	2	3	4
OTC	Core-outlet Temp.	2	4	5

Table 6.9: IET output “best” emulator-based calibration computational times

Output	Best FFGP emulator type	total latent burnin training time [minutes]	total latent posterior training time [minutes]	total uncertain parameter calibration time [minutes]	total emulator-based calibration time (including time to create training set) [minutes]
LPP	2-fact 3-comp	29	19	10	58 (388)
TTC	2-fact 4-comp	29	10	17	56 (386)
OTC	2-fact 5-comp	29	25	28	82 (412)

factor 1. All of the RELAP LPP flow predictions are shown in blue in Fig. 6-43, while the 51 time locations taken per case run are shown in grey. The Transient Test No.10 observational data is shown in red as usual. The LPP flow output is presented unitless as a normalized mass flow rate where the normalizing constant is the known low pressure nominal flow rate. As seen in Fig. 6-43, some of the cases yield LPP flow rates above the nominal low pressure flow rate, due to those case runs having frictional characteristics lower than the actual EBR-II system.

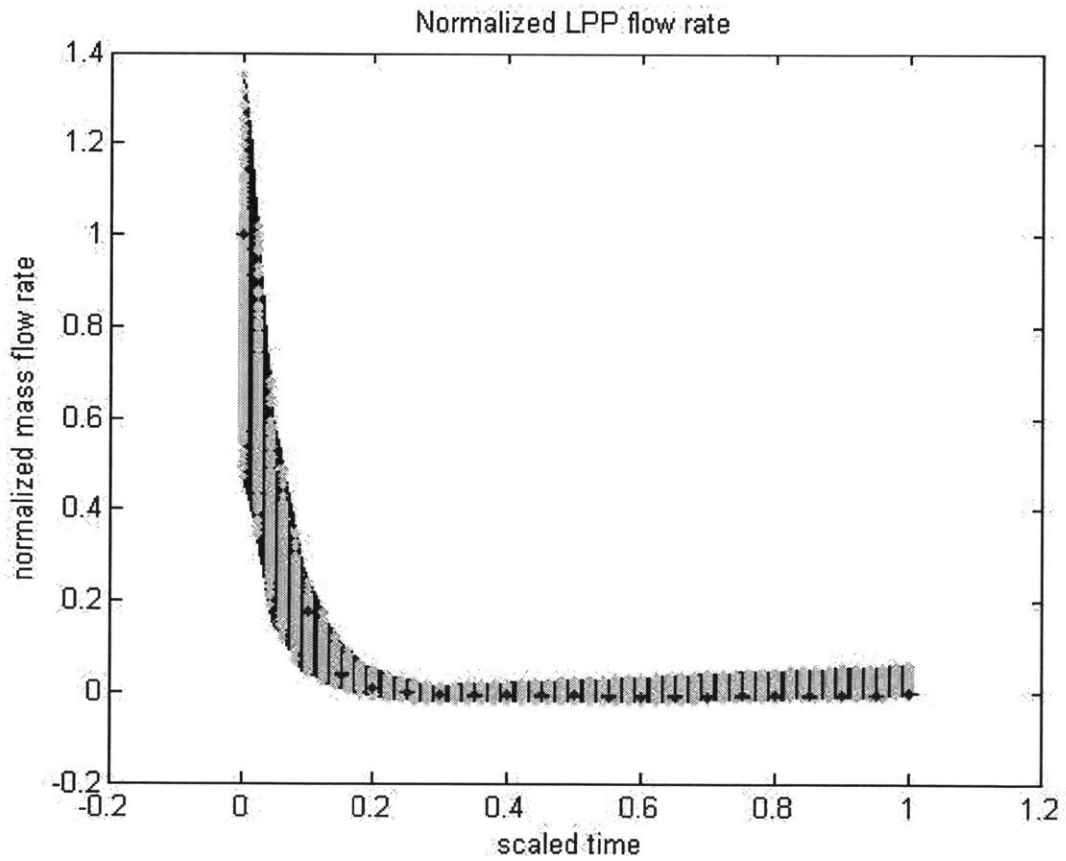


Figure 6-43: LPP flow FFGP training set

The LPP standard GPR emulator consisted of 42 input parameters, the first 41 were the IET model uncertain parameters and the 42<sup>nd</sup> input was time. Just as in the blowdown problem, only one training point was taken per case run so there were 490 training points. The GPR emulator was built using  $5 \times 10^4$  burn-in samples and  $5 \times 10^4$  posterior samples. To prevent RAM overload, the Empirical Bayes approach was used here as well with the hyperparameter point estimates taken as the mean values from the sampling results. Building the GPR emulator took roughly 74 minutes to complete and the likelihood hyperparameter (and covariance function signal noise) results are shown in Fig. 6-44. The mixing rate was very poor which was rather unusual for the GPR emulator training. I tried retraining multiple times, but the GPR emulator training results of the LPP flow output always came back similar. The reason for why I think the mixing rate is so poor actually is best seen from the GPR-based calibration results. The GPR-modified likelihood function was used to perform the AM-MCMC scheme with  $5 \times 10^5$  samples. Figure 6-45 shows the calibrated posterior scaled predictions. The GPR output is scaled to a standard normal, and the reason why Fig. 6-45 shows the scaled predictions is because it shows potentially why the GPR emulator can be such a poor choice for the LPP flow output. The first observational data point, besides being located outside the training set (shown as the grey dots in the figure), is roughly 8 standard deviations away from the scaled mean. The training points in the very beginning of the transient are all located at least 2 standard deviations away from the scaled mean. These training points are essentially extreme values, and with how the GPR emulator predictions work, may influence predictions closer to the rest of the training set that is around the scaled mean (which is 0). Building the GPR emulator is difficult because then it must find hyperparameter values that not only give “good” matches with the training points near the scaled mean, but also at the training points that are beyond

2 standard deviations away from the scaled mean. This is a difficult situation for the standard GPR emulator to handle and as shown in the posterior predictions, results in very high levels of predictive uncertainty.

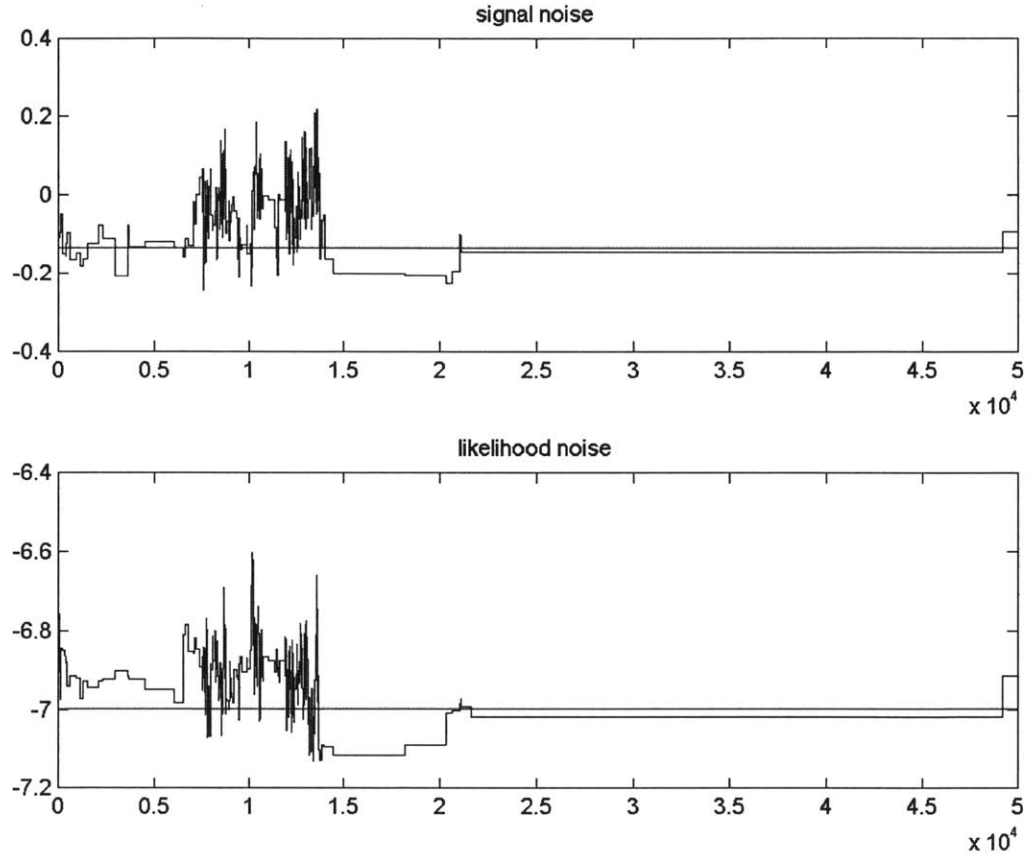


Figure 6-44: LPP GPR likelihood noise hyperparameter

The LPP FFGP 2-factor 3-component emulator was considered to be the “best”. The FFGP 2-factor 3-component emulator was built using  $10^5$  latent burn-in samples and  $5 \times 10^5$  latent posterior samples. I used such a large number of latent posterior

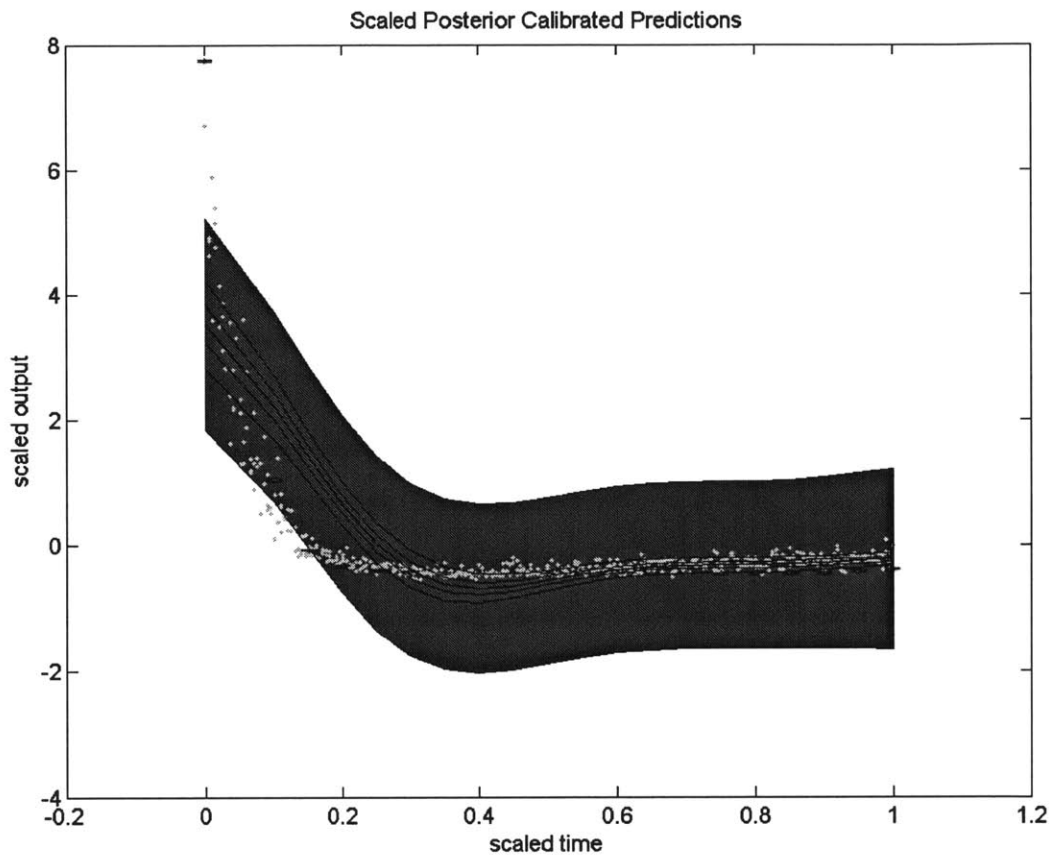


Figure 6-45: LPP GPR calibrated predictions

samples to try and guarantee a good mixing rate of the training latent variables. The latent burn-in phase took roughly 29 minutes and the latent posterior phase took roughly 19 minutes. The FFGP emulator training is faster for a few reasons, first the GPR emulator training uses a different MCMC scheme, the delayed rejection (DR) + adaptive metropolis (AM) scheme. So if the first stage proposal is rejected, the second stage proposal requires inverting the covariance matrix again. Therefore each MCMC iteration might have to invert 2 covariance matrices of size  $490 \times 490$ . As pointed out, each MCMC stage inverts two covariances, but inverting the  $51 \times 51$  factor 1 covariance matrix is very fast compared to inverting a second  $490 \times 490$  covariance matrix. Secondly, to try and speed up the MATLAB computations, I used the MATLAB MEX feature to write a compiled function to build smaller dimensional covariance matrices. The factor 1 covariance matrix is small enough to make use of that compiled function which greatly accelerates a single MCMC iteration. The larger factor 2 covariance matrix (and thus the GPR covariance matrix) are too large to make use of that faster compiled function. Figure 6-46 shows the likelihood noise hyperparameter for the LPP FFGP 2-factor 3-component emulator.

The built LPP FFGP 2-factor 3-component emulator was then used to calibrate the IET uncertain parameters. All of the IET uncertain parameters influence (some more than others) all three IET outputs. So calibrating the uncertain parameters with each IET output separately first, compared to calibrating with all three IET outputs simultaneously provides a useful demonstration of which of which parameters influence each output the most. Only  $5 \times 10^4$  samples were used in the FFGP emulator-based uncertain parameter calibration process because each iteration is relatively slow. A single AM-MCMC iteration with the LPP FFGP 2-factor 3-component emulator takes approximately 0.0117 seconds. And so these results were generated in about 10 minutes. The FFGP predictions are so slow for several



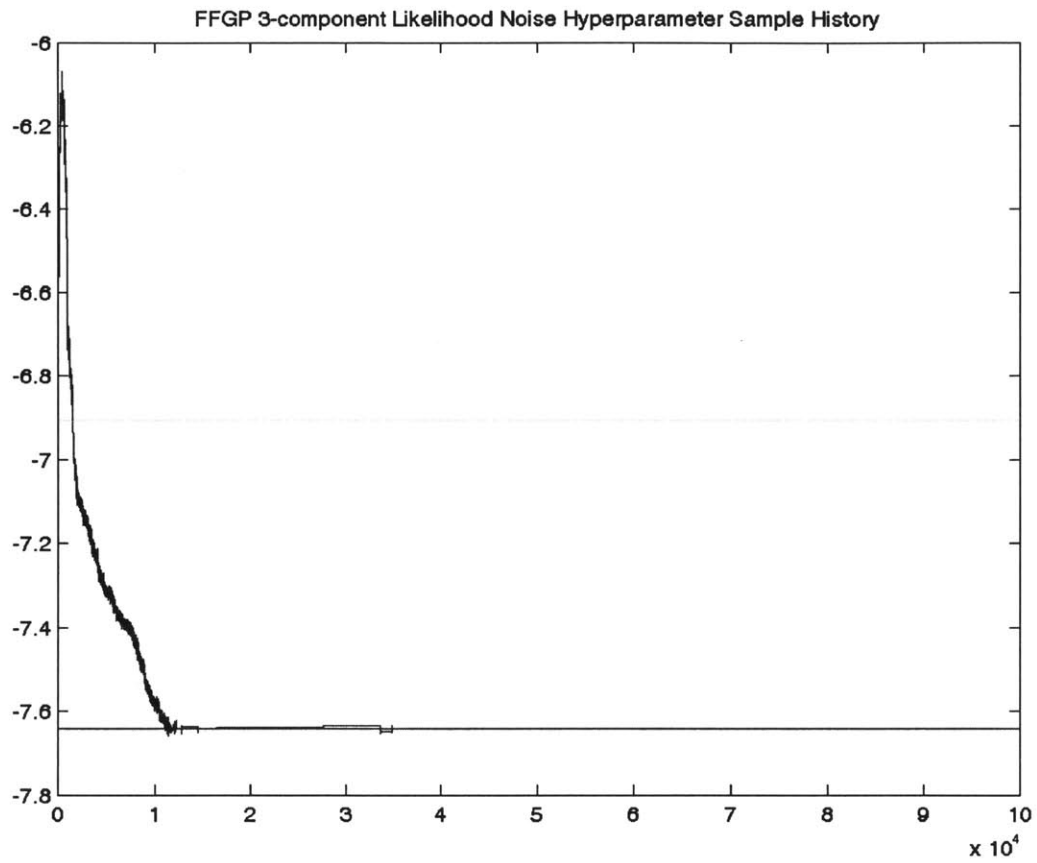


Figure 6-46: LPP FFGP 2-fact 3-comp likelihood noise hyperparameter

reasons, first of the size of the training set. Second, I make 201 predictions at one second (simulation time) intervals - corresponding to a one-to-one comparison in the amount of predictions I take from a single RELAP run. In order to make a prediction, each predictive point must be compared to each training set point (as shown in Chapter 4). Thanks to the structure of the FFGP emulator, each MCMC iteration is only changing the factor 2 (the uncertain parameter factor) predictive input, so this large factor 1 (the control variable - or time - factor) only needs predictive “information” only needs to be computed once. But, this does mean the matrix operations are larger than other emulators I have constructed as part of this thesis and that slows down the computation slightly. Lastly, computing the posterior predictive variance with contributions from covariances between each factor and each component are quite slow. If the posterior predictive covariance structure was ignored, the FFGP emulator prediction is almost twice as fast. But depending on the model, doing that can drastically over-estimate the posterior predictive variance.

The LPP FFGP emulator-based calibrated posterior predictions are shown in Fig. 6-47 and the associated posterior uncertain parameter histograms are shown in Fig. 6-48. The FFGP emulator-based posterior predictions follow the observational data very well. Many of the data points fall within the emulator total predictive variance, which does limit some of the posterior “learning” of the uncertain parameters. But it is very apparent the FFGP emulator predictions are far superior to the standard GPR emulator predictions for the LPP flow output. Several of the uncertain parameters that change the most from their uniform priors make intrinsic sense given that this is the LPP flow data. IET uncertain parameters 39 and 40, the uncertain inlet flow boundary condition parameters and IET uncertain parameter 6, the low pressure throttle valve clearly dominate the LPP flow data since they directly impact the low pressure flow rate. But since these are relatively few samples some of the movement

in other parameters might be more due to just the sampling not being the best.

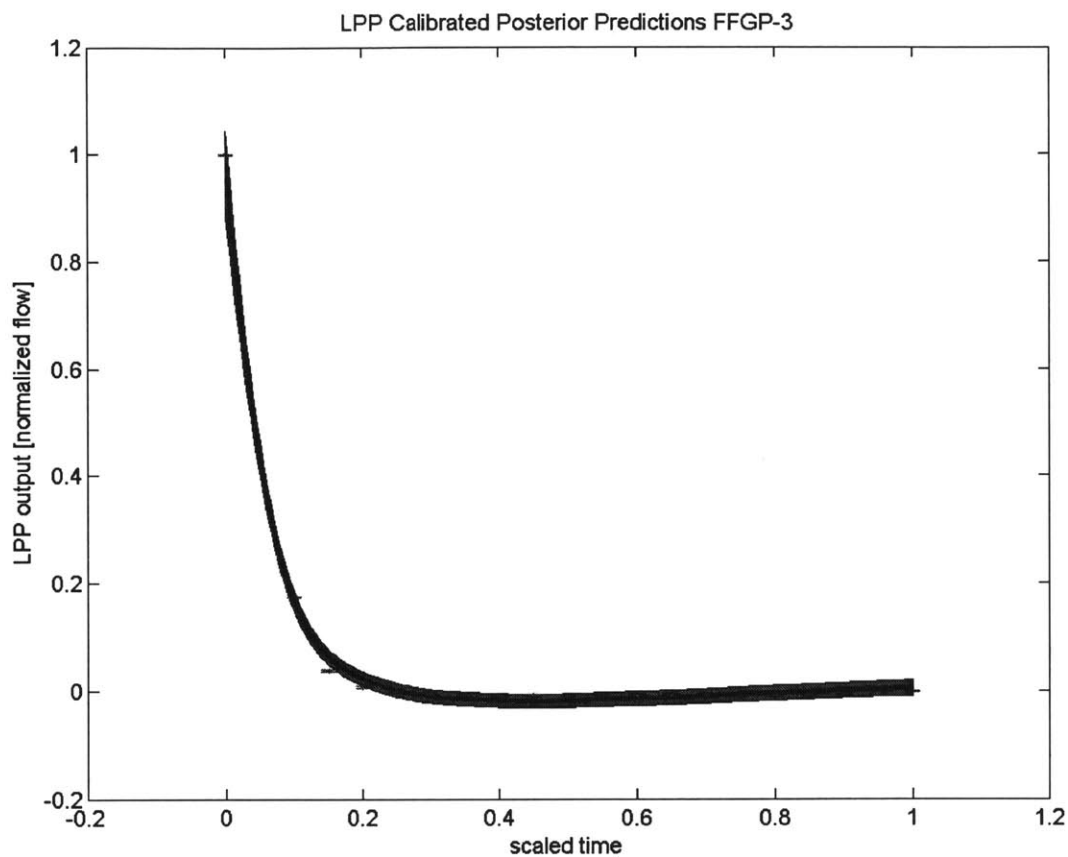


Figure 6-47: LPP FFGP emulator-based calibrated predictions

### 6.5.2 TTC results

The TTC output training set is shown in Fig. 6-49 and is setup just like the LPP flow training set. The TTC GPR emulator was built with the same number of samples and took about same amount of time to train as the LPP flow GPR emulator. Figure

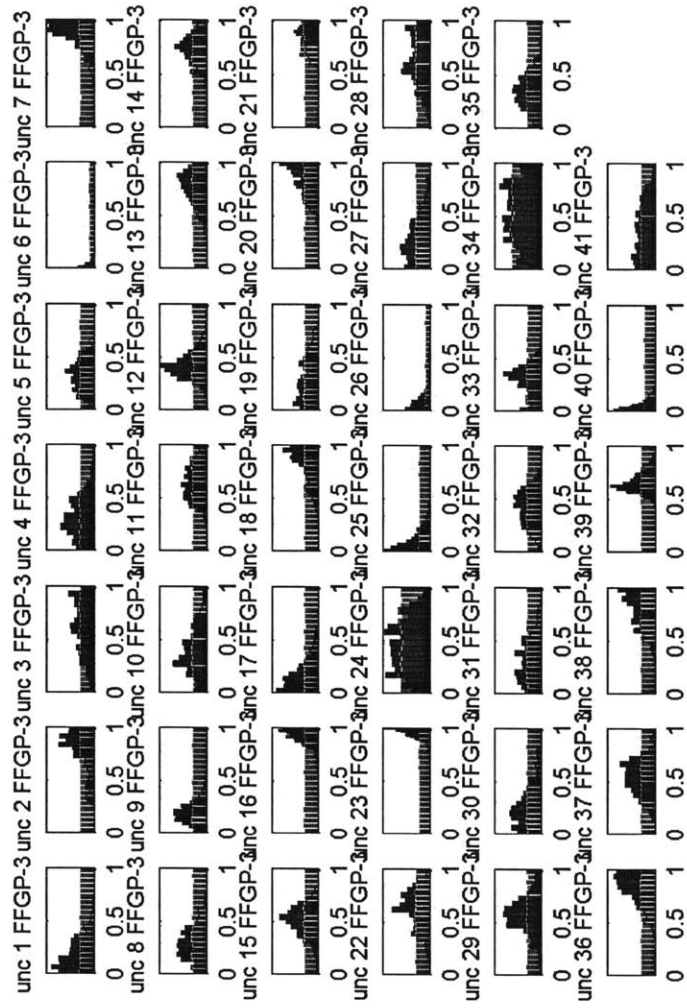


Figure 6-48: LPP FFGP emulator-based calibrated uncertain parameter histograms

6-50 shows the GPR emulator likelihood noise hyperparameter. The fact the mixing rate is so much better than the likelihood noise hyperparameter for the LPP flow output, suggests the GPR emulator should perform better on the TTC output than the LPP flow output. That is indeed the case as shown by the calibrated posterior GPR TTC predictions in Figure 6-51. The posterior predictions roughly follow the data points and the total emulator predicted variance is not much larger than the assumed observational error. As we expected from the blowdown problem in Chapter 5, the GPR emulator posterior predictions do not appear to follow a trajectory. The GPR emulator is simply trying to interpolate (regress, really) the various training points when making a prediction and so does not “see” if one point in time comes before or after another. But that said, if the FFGP emulators were not available, this would have been considered a pretty good emulator of the TTC output. Also, predictions are only made at points in time corresponding to a data point. This was done to save time for the GPR predictions since each prediction point must be compared against all training points.

The “best” FFGP emulator for the TTC output was the 2-factor 4-component emulator. Unfortunately, I found from experience that if I used  $5 \times 10^5$  latent posterior samples with a 2-factor 4-component or 5-component emulator, sometimes MATLAB would exceed its RAM limit. Therefore I used only  $2 \times 10^5$  latent posterior samples to build the TTC FFGP emulator. There are definitely ways to modify my existing FFGP training functions to completely avoid this issue. However, as described in the future work chapter later on, there are other changes that would be more beneficial to implement instead. The latent burn-in phase took about 29 minutes using  $10^5$  samples, and with fewer samples than the LPP FFGP emulator, the latent posterior sampling phase took about 10 minutes. The associated likelihood noise hyperparameter is shown in Fig. 6-52 and this alone shows just how complex

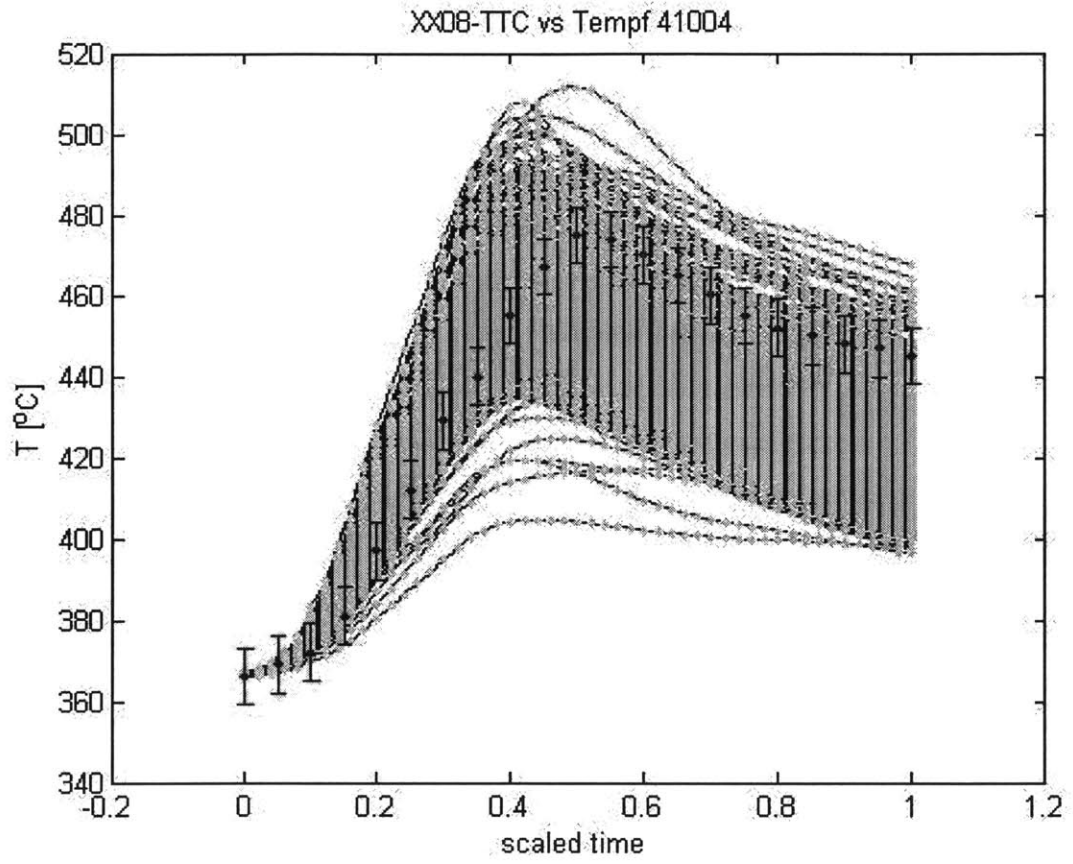


Figure 6-49: TTC FFGP training set

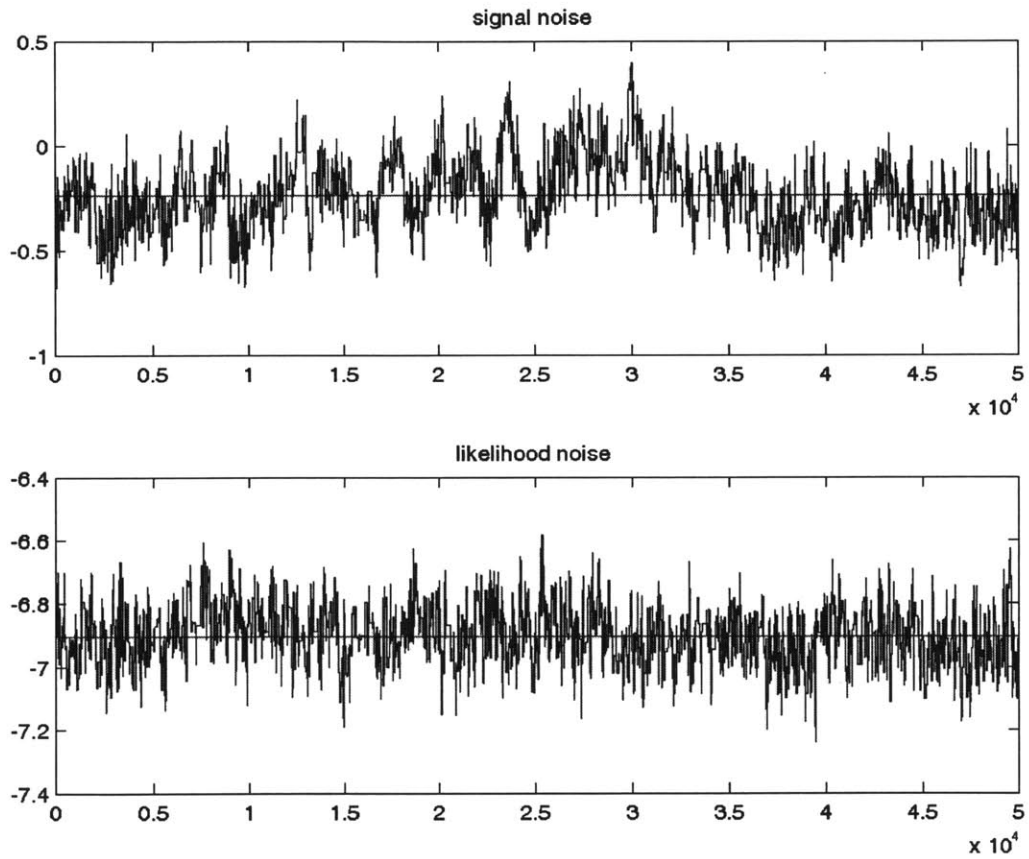


Figure 6-50: TTC GPR likelihood noise hyperparameter

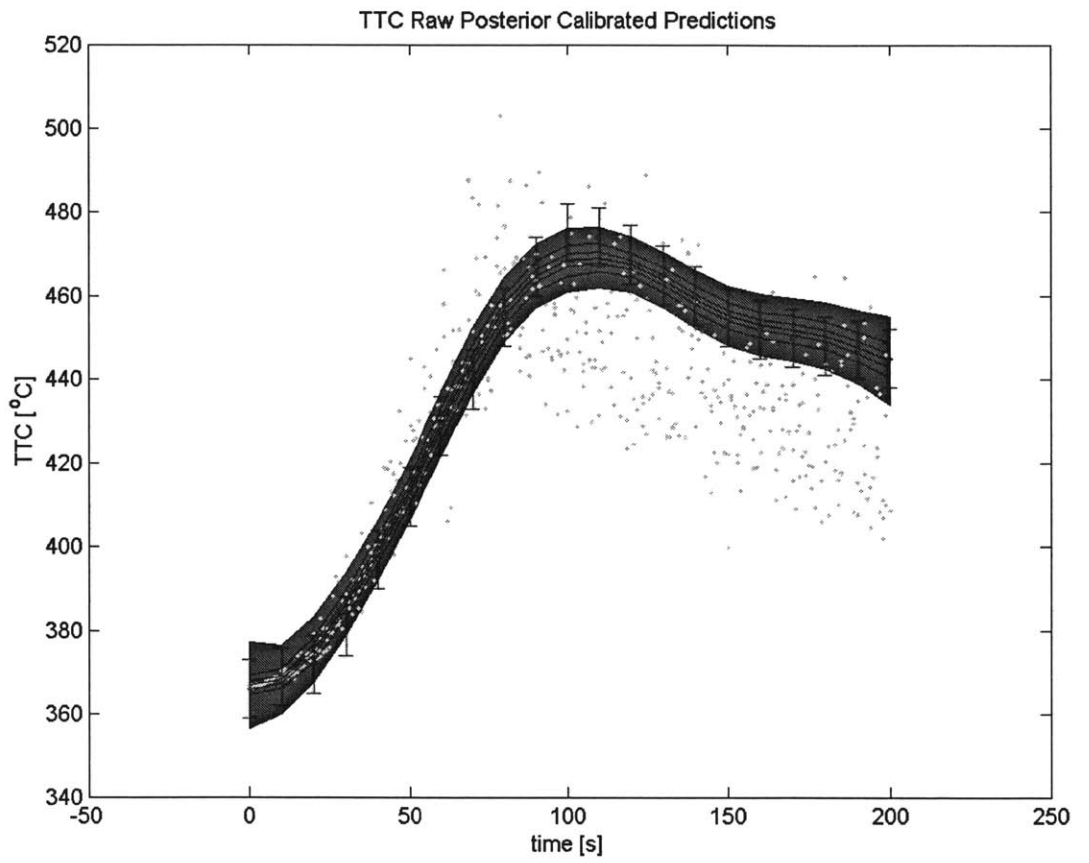


Figure 6-51: TTC GPR calibrated predictions



the TTC output is since even a 2-factor 4-component FFGP emulator has a likelihood noise hyperparameter value only slightly more negative than the initial guess value. Simpler problems would have a likelihood noise hyperparameter value around -10 at least for a 4-component FFGP emulator. However, the TTC relationship to all of the uncertain parameters is so complex, even with the thousands of latent variables within this FFGP emulator, it is still a difficult output to emulate. This built FFGP emulator was then used to calibrate the uncertain parameters in the IET model. The calibrated posterior TTC FFGP emulator predictions are shown in Fig. 6-53 and the emulator-based calibrated uncertain parameter histograms are shown in Fig. 6-54. The calibrated posterior FFGP-based TTC predictions are clearly better than the GPR-based predictions. There is even less total predictive variance initially which then increases as the transient simulation time increases. As with the LPP FFGP emulator, a single AM-MCMC iteration was relatively slow, taking  $\sim 0.0204$  seconds. The calibrated uncertain parameter histograms also appear to “make sense” in that the uncertain inlet flow boundary condition parameters as well as the initial power multiplier (IET uncertain parameters 39, 40, and 41) are quite different from their uniform priors. The radial and axial conduction parameters (IET uncertain parameters 30 through 37) have also been updated, which makes sense given that the conduction within the fluid will impact the temperature distribution at the low flow rates within the core channel. But the DF heat transfer coefficient multiplier (IET uncertain parameter 27) was not updated nearly as much as I originally expected it to. It is very important to remember that these are the scaled uncertain parameter values and so their absolute value ranges, as shown in Table 6.1 can vary significantly from parameter to parameter. The heat transfer coefficient multiplier prior ranges are only  $\pm 50\%$  around a prior mean of 1, while the radial conduction parameters' maximum bounds are 9x greater than their minimum bound. It is therefore diffi-

cult to assess just which of the parameters are more important since the conduction parameters priors were perhaps far too uncertain. Those priors did represent just how uncertain the conduction parameters were initially. Additionally, many of the friction parameters and loss coefficients might not be known precisely enough to help reduce the posterior variance in the heat transfer coefficient multipliers. Since the frictional characteristics of the reactor vessel will dictate the flow split between the high and low pressure streams as well as between the core and IB channel within the high pressure stream, if they are not known precisely enough it might be difficult to further resolve the heat transfer coefficient multipliers. This is where simultaneous calibration should help later on.

### 6.5.3 OTC results

The OTC output FFGP training set is shown in Fig. 6-55. The training set alone shows how complicated the OTC functional relationship is to the uncertain parameters. A handful of case runs behave completely differently from the majority of cases over the middle portion of the transient. Presumably these correspond to extreme values of the conduction parameters along with other uncertain parameters. Additionally, the observational data seems to fall between many of the training case runs, and sits in “white space” in Fig. 6-55. The training set alone therefore suggests that making emulator predictions on the OTC output might be challenging.

The GPR emulator for the OTC output was built following the same setup as the LPP and TTC output GPR emulators and took a similar amount of time. The GPR emulator likelihood noise hyperparameter is shown in Fig. 6-56. The mixing rate is similar to the TTC likelihood noise hyperparameter mixing rate. The calibrated posterior GPR predictions of the TTC output are then shown in Fig. 6-57. Although

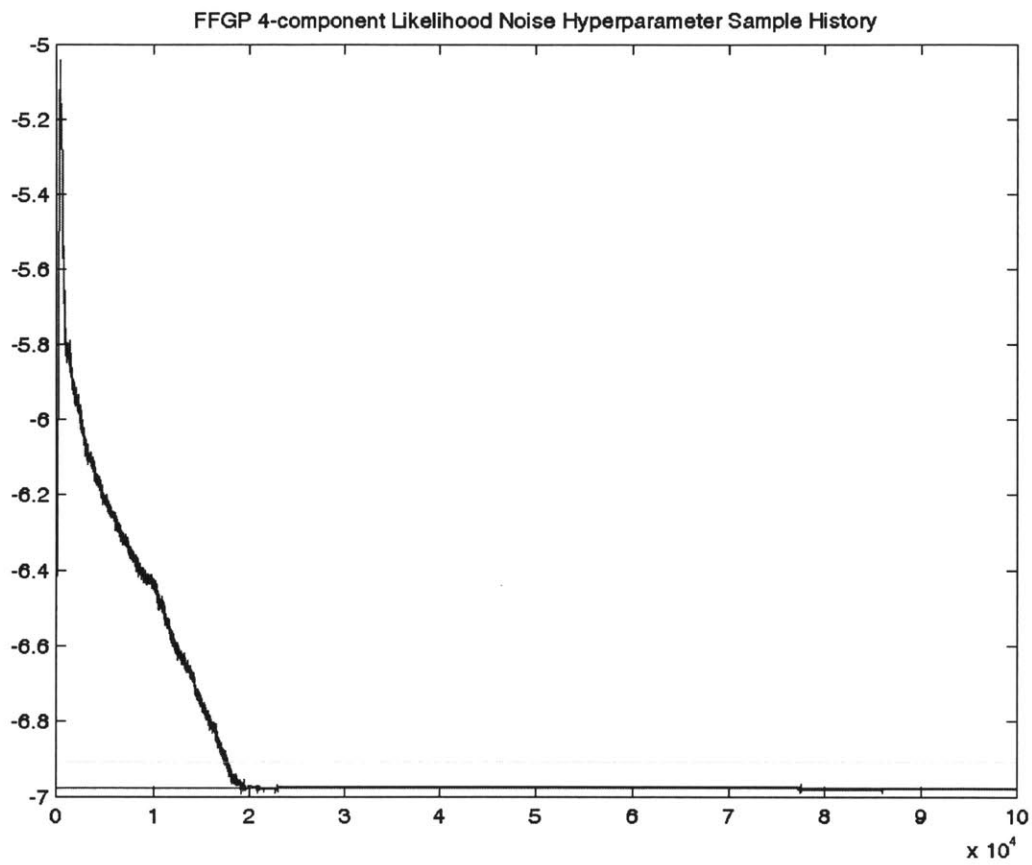


Figure 6-52: TTC FFGP 2-fact 4-comp likelihood noise hyperparameter

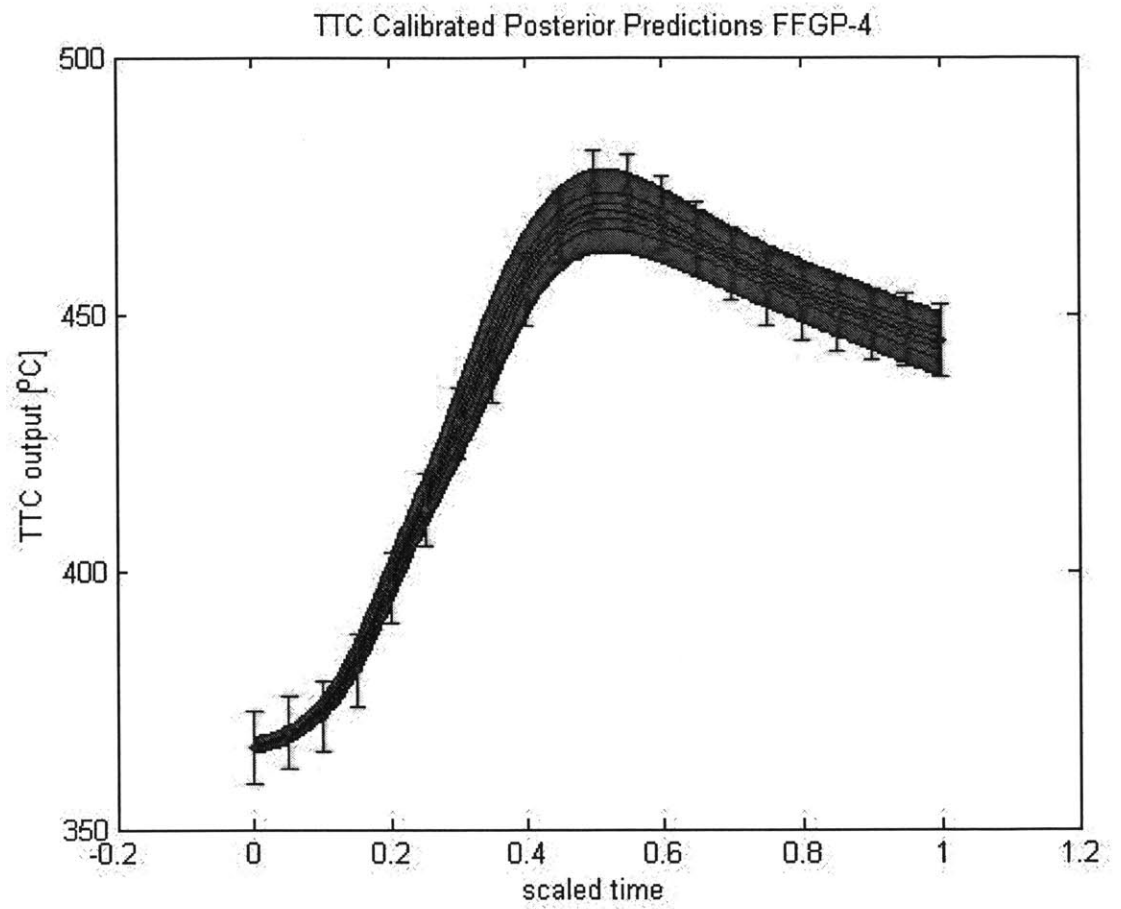


Figure 6-53: TTC FFGP emulator-based calibrated posterior predictions

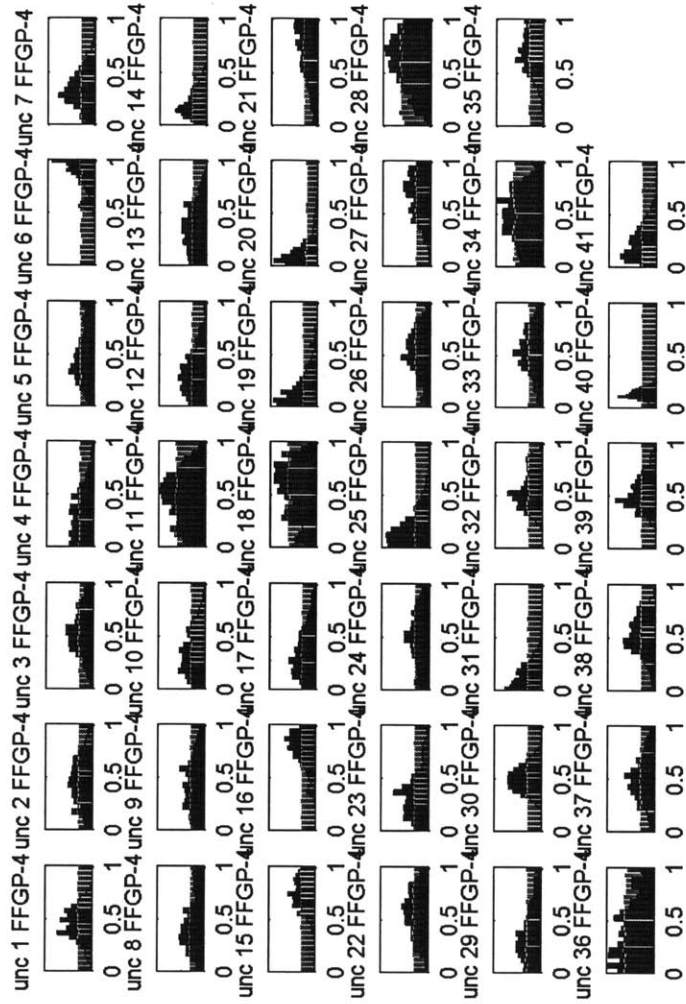


Figure 6-54: TTC FFGP emulator-based calibrated uncertain parameter histograms

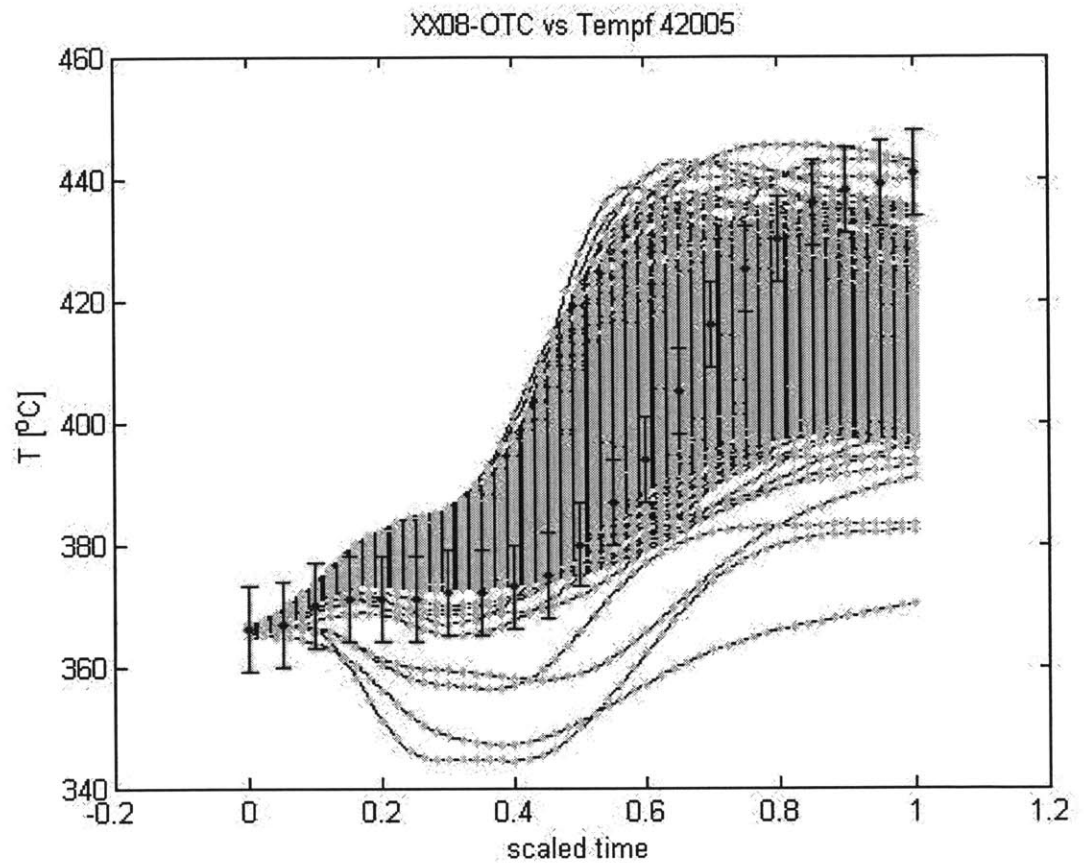


Figure 6-55: OTC FFGP training set

the predictions in general follow the observational data, the large amount of emulator total predictive variance clearly impacts the posterior results.

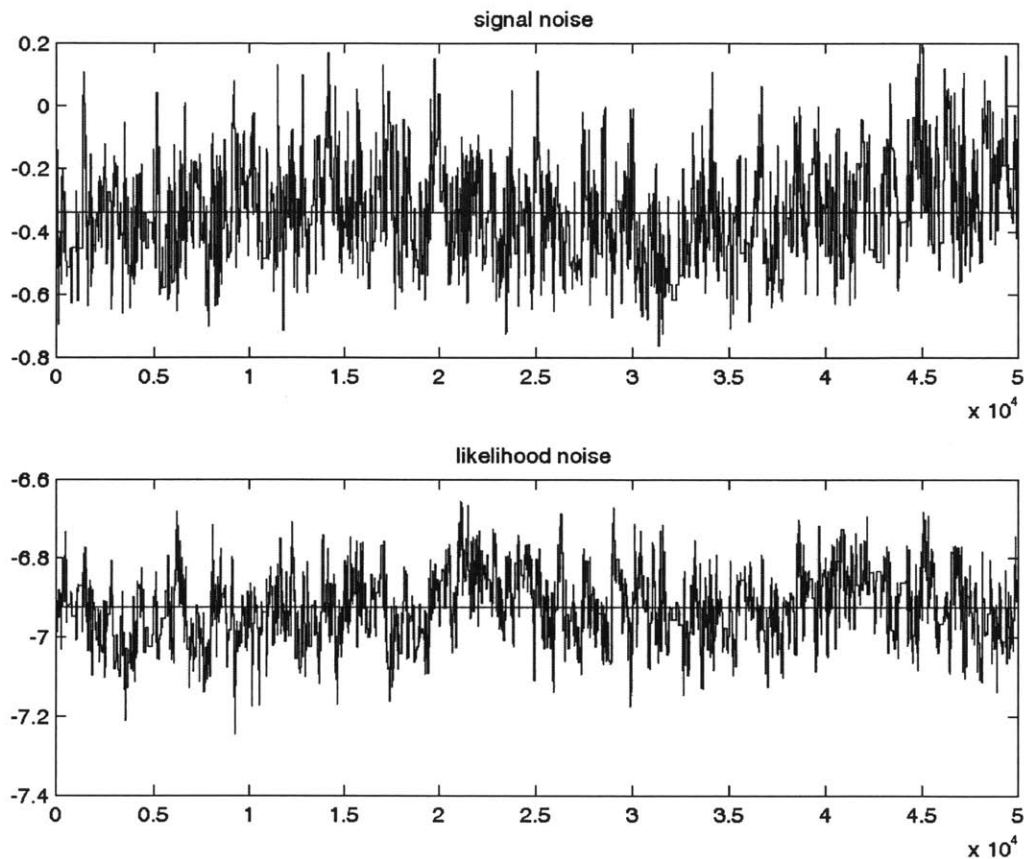


Figure 6-56: OTC GPR likelihood noise hyperparameter

In building the FFGP emulators for the OTC output, it took till a 2-factor 5-component emulator to start to get likelihood noise values low enough to yield an FFGP emulator reasonably accurate relative to the training set. The FFGP 2-factor 5-component likelihood noise hyperparameter is shown in Fig. 6-58. This

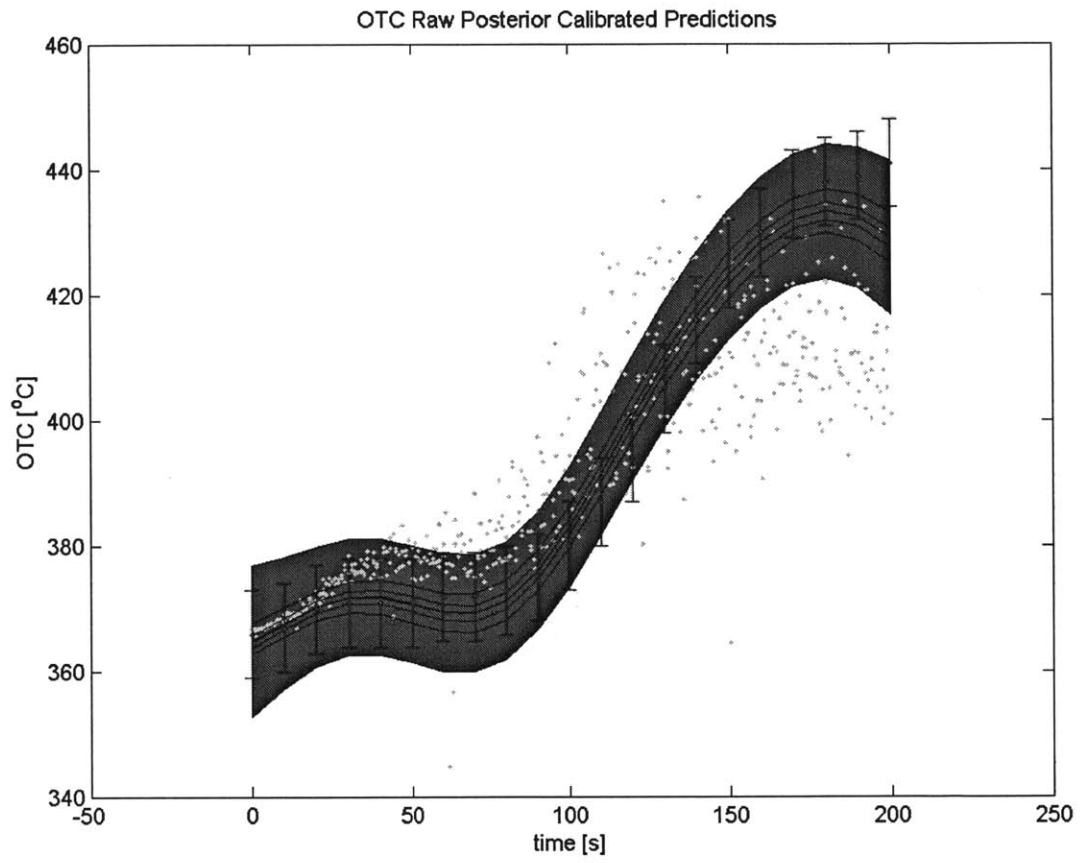


Figure 6-57: OTC GPR calibrated predictions



just further highlights how complex the OTC output is, with a large number of uncertain parameters all interacting together. Inherently this makes sense since the OTC output is the core outlet temperature which is located at the top of the UB portion of the core channel. Thus after being heated within the driver fuel section the hotter sodium can conduct its heat to the cooler portions within the reactor vessel. Conduction within the fluid should then be correlated to many of the friction parameters and loss coefficients within the IET model which dictate the flow rates through the system, as well as the uncertain inlet flow boundary condition parameters. As with the TTC 2-factor 4-component emulator, only  $2 \times 10^5$  latent posterior samples were used. Training took about 29 minutes for the latent burn-in phase, and about 25 minutes for the latent posterior sampling phase.

The calibrated posterior OTC FFGP emulator predictions are shown in Fig. 6-59 with the associated calibrated uncertain parameter histograms shown in Fig. 6-60. A single AM-MCMC iteration with this emulator took about 0.0336 seconds. The most striking feature about the posterior predictions is that there is still a relatively large amount of emulator total predictive variance. And this clearly impacts the how well the MCMC sampling can update the uncertain parameters. What is interesting though, is how the best match between the posterior predictive means and the data occurs when the OTC temperature is increasing, and therefore the coolant is heating up within the core channel. Conversely, when the OTC temperature is more level the predictive means are more innaccurate relative to the data. I think this suggests that when the conduction parameters are completely dominating the OTC response, by smoothing out the temperature within the core channel, the FFGP emulator struggles the most. The calibrated uncertain parameter histograms seem to reflect this, since the conduction parameters (IET uncertain parameters 30 through 38) are updated from their uniform priors. The uncertain inlet flow boundary condition parameters

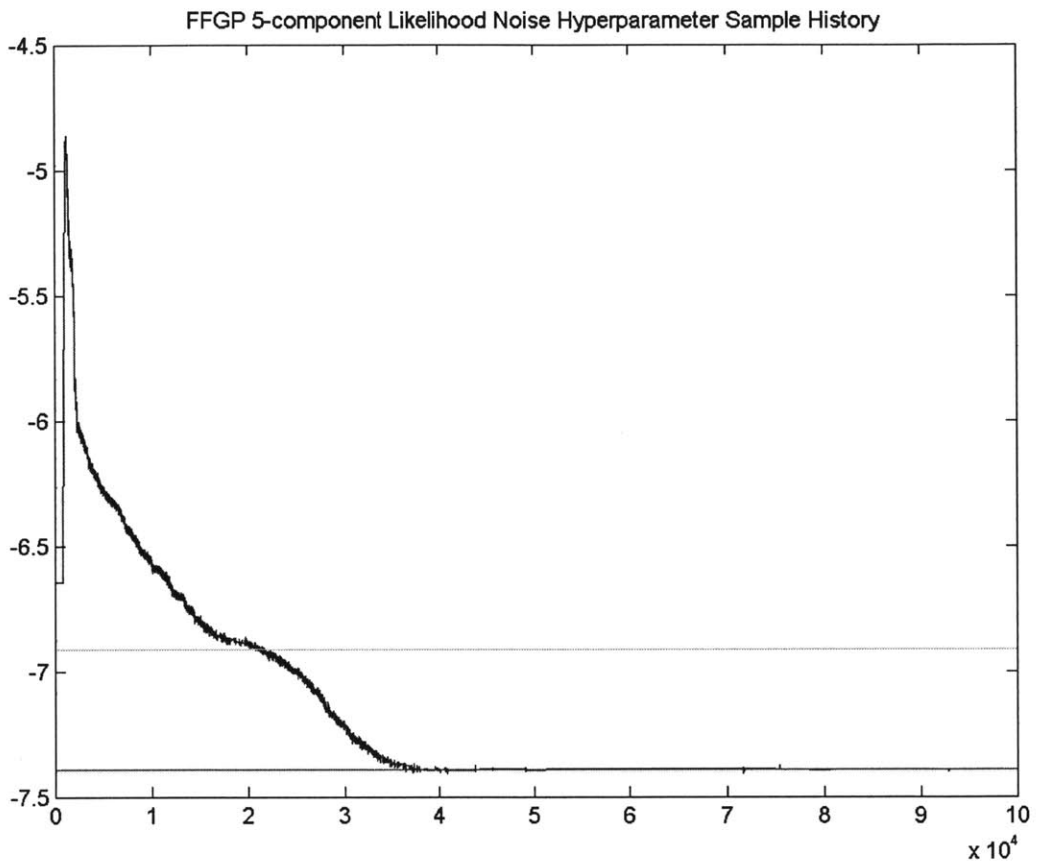


Figure 6-58: OTC FFGP 2-fact 5-comp likelihood noise hyperparameter

(IET uncertain parameters 39 and 40) show considerable updating which makes sense since the flow rate is so important in determining when conduction within the fluid becomes important.

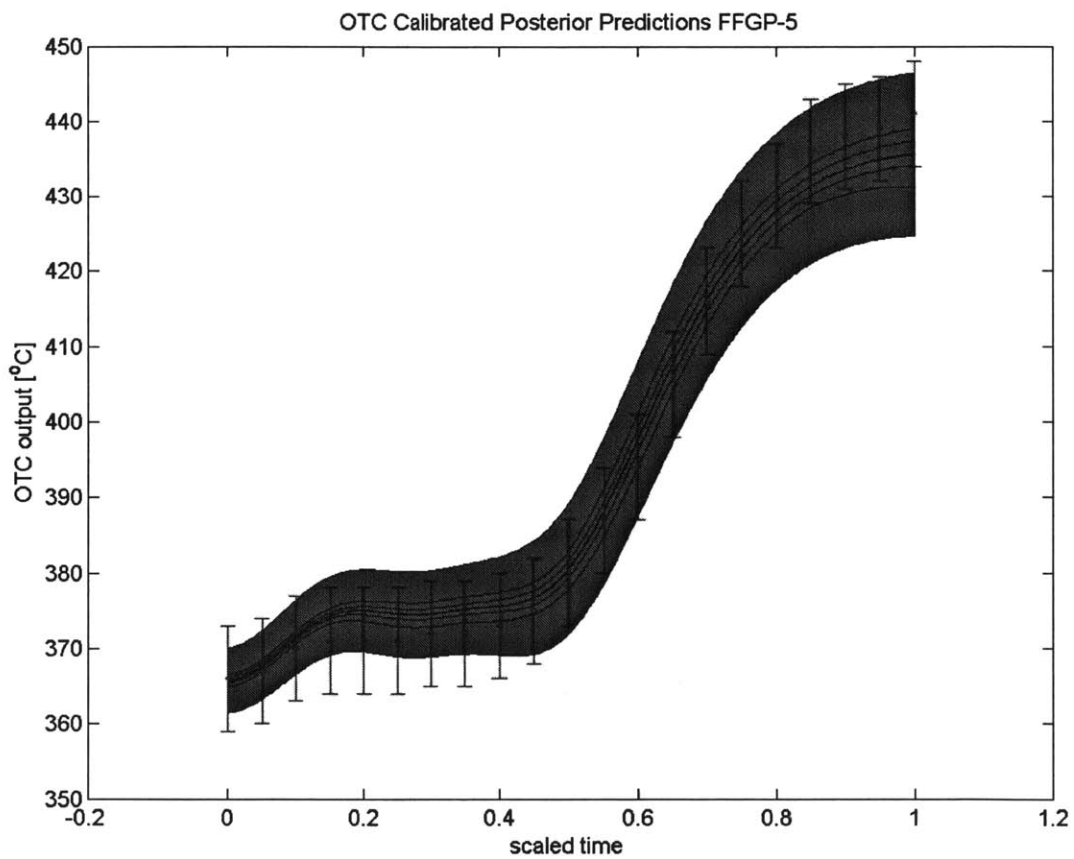


Figure 6-59: OTC FFGP emulator-based calibrated predictions

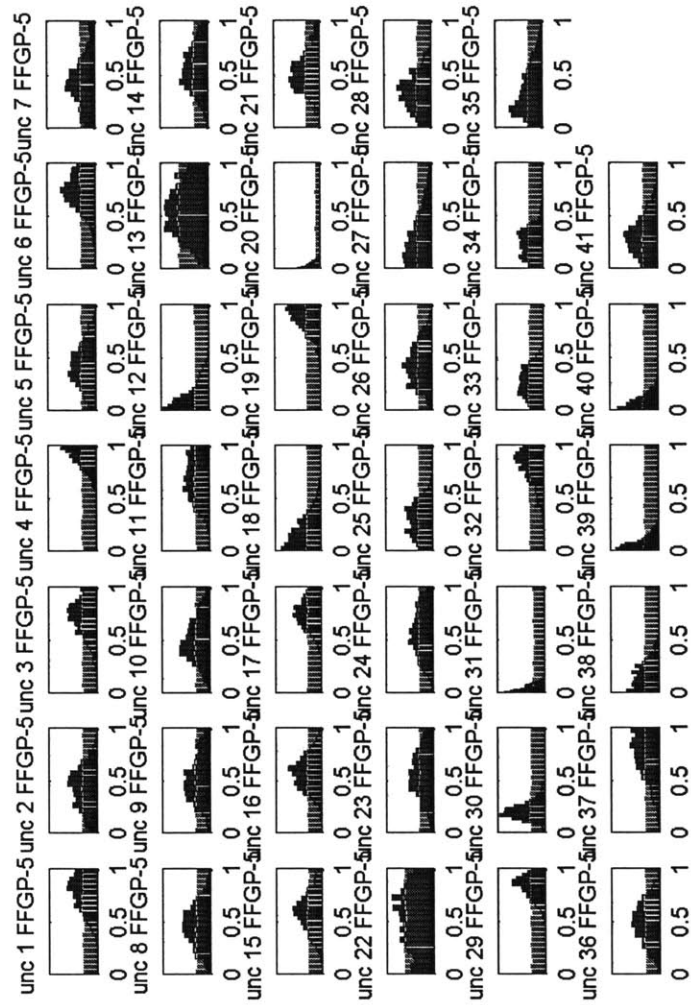


Figure 6-60: OTC FFGP emulator-based calibrated uncertain parameter histograms

## 6.6 Simultaneous Calibration

Simultaneous calibration, as the name implies, refers to calibrating multiple models at the same time. The uncertain parameters involved in each of the models are then updated using additional information. The IET model alone represents how simultaneous calibration is useful since all three IET outputs (and thus data types) depend on the 41 IET uncertain parameters. The LPP flow data can help inform various friction and loss coefficients through the system more precisely than the OTC or TTC data might be able to. And as already discussed, with the frictional characteristics of the system known more precisely that should help better inform the conduction parameters. Incorporating the information gained from the SETs only compounds this effect since the SETs are capable of updating their corresponding uncertain parameters to a high level of precision. Table 6.10 summarizes the “best” FFGP emulators for each of the SET models and IET outputs.

Table 6.10: Summary of “best” FFGP emulators

Model	FFGP type
CTDF	2-factor 2-component
GGcore	2-factor 3-component
IBCT	2-factor 3-component
IBGG	2-factor 3-component
OBGG	2-factor 3-component
LPP	2-factor 3-component
TTC	2-factor 4-component
OTC	2-factor 5-component

The following subsections will show the results of simultaneously calibrating multiple models together in various combinations. Each of the bundle-specific SETs will be calibrated simultaneously with their respective channel SET. This provides a simple demonstration and is easier to describe the results since the number of uncertain

parameters is smaller than the full IET case. After this, the simultaneous calibration of all three IET output types is performed, followed by the simultaneous calibration of all SETs with all three IET output types. This last case constitutes updating the IET uncertain parameters with the most amount of information available. Only the “best” FFGP emulators, as shown previously, will be used for each of the model types.

During simultaneous calibration, I make each of the various emulator predictions in series. This is obviously slower than if the computations were done in parallel, but I found in MATLAB sometimes making computations in parallel with the standard parallel loop, 'parfor', might mess up results. As a student, my MATLAB license does not cover the MATLAB Parallel Computing Toolbox, so maybe having that Toolbox enabled would remove any of the issues I encountered when I tried making emulator predictions in parallel. Obviously, if the simulators were all used directly, parallel computations would improve the overall speed as well. But using emulator predictions in series was just the simplest way to perform the simultaneous calibration process.

### 6.6.1 Likelihood structure

In order to perform simultaneous calibration, the simultaneous likelihood function must be setup. For simplicity I always assume that the various models are independent *a priori*. Thus only the observational data can introduce any correlation amongst the outputs. This is obviously not ideal for the three IET output types, but with the current FFGP formulation this is the simplest method. More complex FFGP formulations however would allow this assumption to be removed, as will be discussed in the future work chapter. In keeping with the notation from Chapter 2, the model

predictions are denoted as  $\mathbf{y} = \{\mathbf{y}_z\}_{z=1}^Z$  and the observational data is denoted as  $\mathbf{y}_o = \{\mathbf{y}_{o,z}\}_{z=1}^Z$ .  $Z$  is the total number of models, and  $\mathbf{y}_z$  denotes the predictions for model  $z$  specifically. The complete likelihood function between all models simply takes a factorized form:

$$p(\mathbf{y}_o|\mathbf{y}) = \prod_{z=1}^Z p(\mathbf{y}_{o,z}|\mathbf{y}_z). \quad (6.15)$$

The convenience of this assumption is that now the specific model  $z$  likelihood function  $p(\mathbf{y}_{o,z}|\mathbf{y}_z)$  can be the exact same emulator-modified likelihood function I described in Chapter 4 without any alterations. All of the uncertain parameters will still be denoted as  $\theta$  but now the uncertain parameters specific to model  $z$  will be defined as  $\theta_z$ . The various models may all use the exact same uncertain parameters (such as the case for the three IET output types) or they may share only a fraction of the uncertain parameters (such as between the various SETs, and the SETs between the IET outputs). The joint-posterior between all of the model predictions and all uncertain parameters is also just a factorized form, following Eq. 6.15:

$$p(\mathbf{y}, \theta|\mathbf{y}_o) \propto \prod_{z=1}^Z \{p(\mathbf{y}_{o,z}|\mathbf{y}_z) p(\mathbf{y}_z|\theta_z)\} p(\theta). \quad (6.16)$$

The model predictions can be integrated out using the emulator-modified likelihood functions. Using the notation from Chapter 4, the FFGP-modified likelihood function for model  $z$  is just:

$$p(\mathbf{y}_{o,z} | \{\mathbf{x}_{cv,o}^z, \theta_z\}, \mathcal{D}_z, \hat{\Xi}_z) = \prod_{l=1}^{N_{O,z}} p(y_{o,z,l} | \{x_{cv,o,l}^z, \theta_z\}, \mathcal{D}_z, \hat{\Xi}_z), \quad (6.17)$$

where  $\mathcal{D}_z$  and  $\hat{\Xi}_z$  denotes the  $z$ -th model's emulator training set and set of Empirical Bayes determined FFGP hyperparameters, respectively. The posterior distribution

on all uncertain parameters is then given as:

$$p\left(\theta|y_o, \left\{\mathcal{D}_z, \hat{\Xi}_z\right\}_{z=1}^Z\right) \propto \prod_{z=1}^Z \left\{ \prod_{l=1}^{N_{O,z}} p\left(y_{o,z,l} | \{x_{cv,o,l}^z, \theta_z\}, \mathcal{D}_z, \hat{\Xi}_z\right) \right\} p(\theta). \quad (6.18)$$

## 6.6.2 Core Channel SETs Simultaneous Calibration

The simultaneous calibration of the CTDF and GGcore SETs used the FFGP 2-factor 2-component emulator and 2-factor 3-component emulator for the CTDF model and GGcore model, respectively. A total of  $2 \times 10^5$  AM-MCMC samples were made with the first  $10^5$  samples discarded as burn-in. For each proposal during the sampling, the CTDF emulator prediction is made, followed by the GGcore emulator prediction. Then the proposal (un-normalized) log-joint-posterior value is computed using the FFGP-modified simultaneous joint-posterior structure in Eq. 6.18. A single iteration took only slightly longer than the sum of the two individual SET calibration iterations times, at about 0.0014 seconds. Figures 6-61 through 6-63, show the calibrated uncertain parameter histograms, associated CTDF calibrated posterior predictions and GGcore calibrated posterior predictions, respectively. The calibrated uncertain parameter histograms in Fig. 6-61 show exactly what we would expect to happen by using both the CTDF and GGcore “observational” datasets together: the DF turbulent friction parameters (IET uncertain parameters 18 and 19) are updated by the CTDF data while all the other uncertain parameters essentially match the results from when the GGcore SET was calibrated individually. The posterior histogram on the core channel inlet nozzle loss coefficient (IET uncertain parameter 8) looks more Gaussian than what was shown in Fig. 6-28, however. This can be due to several reasons. First, since the DF turbulent friction parameters are known more precisely the loss coefficient can be resolved more. And secondly, the mixing rate might be



better here.

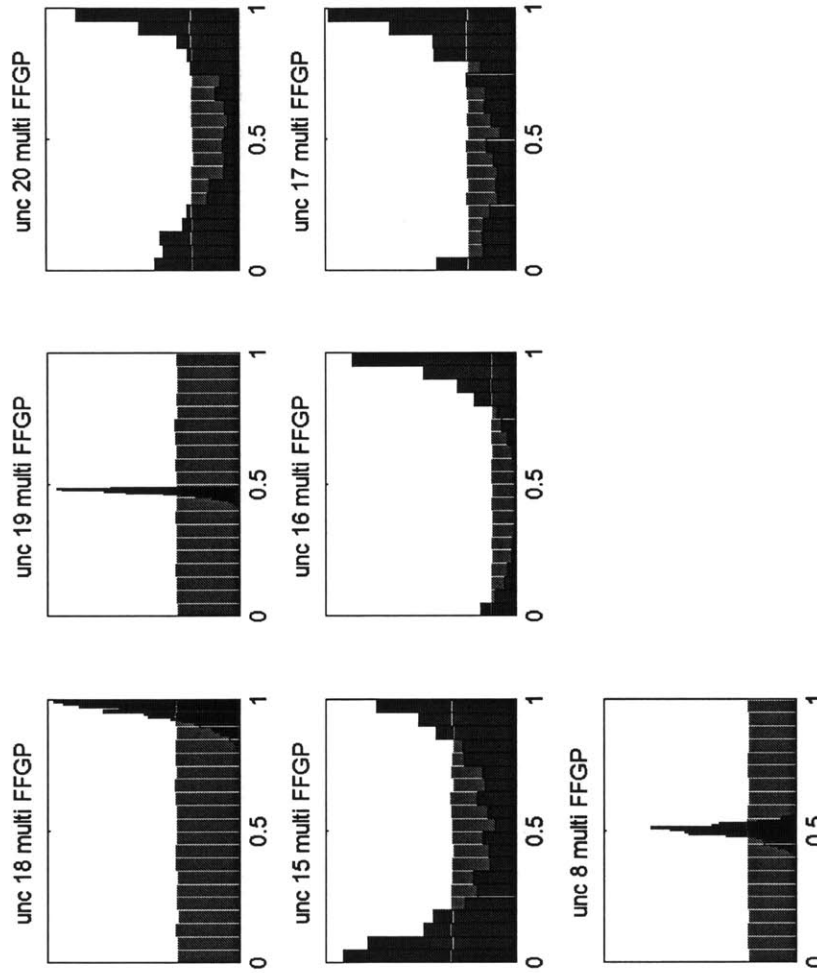


Figure 6-61: Simultaneous CTDF and GGcore calibration uncertain parameter histograms

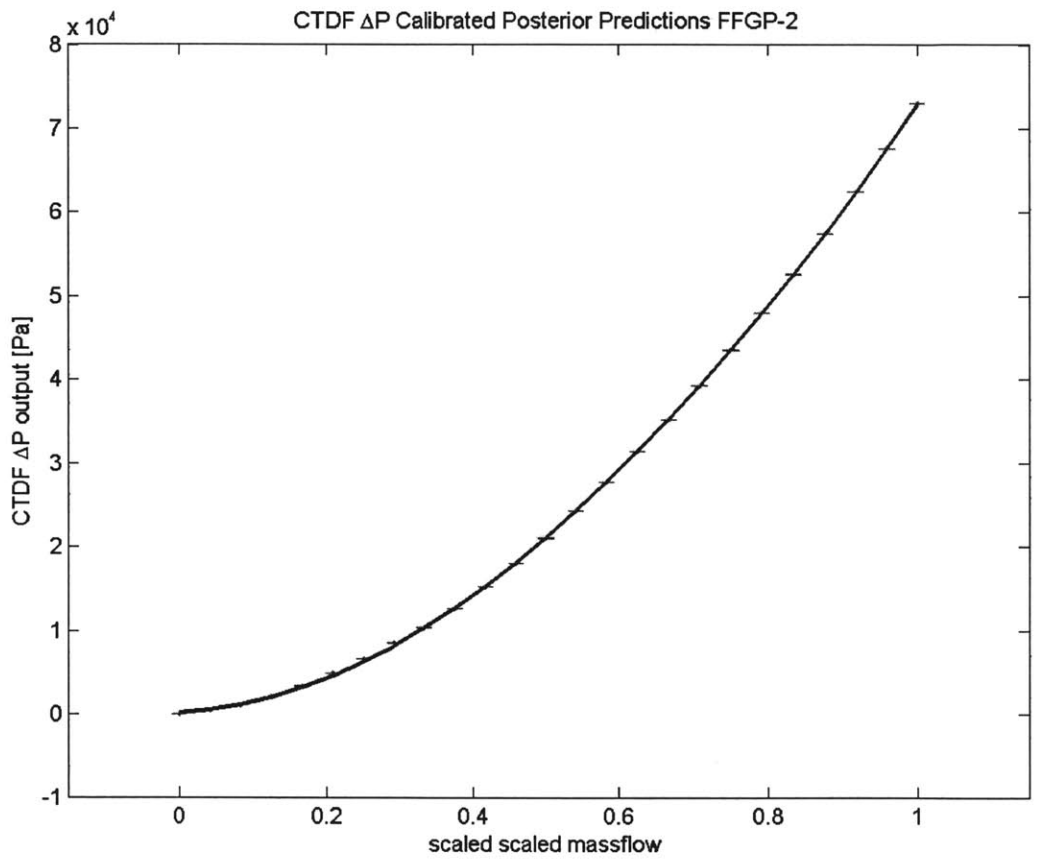


Figure 6-62: Simultaneous calibration prediction results for CTDF

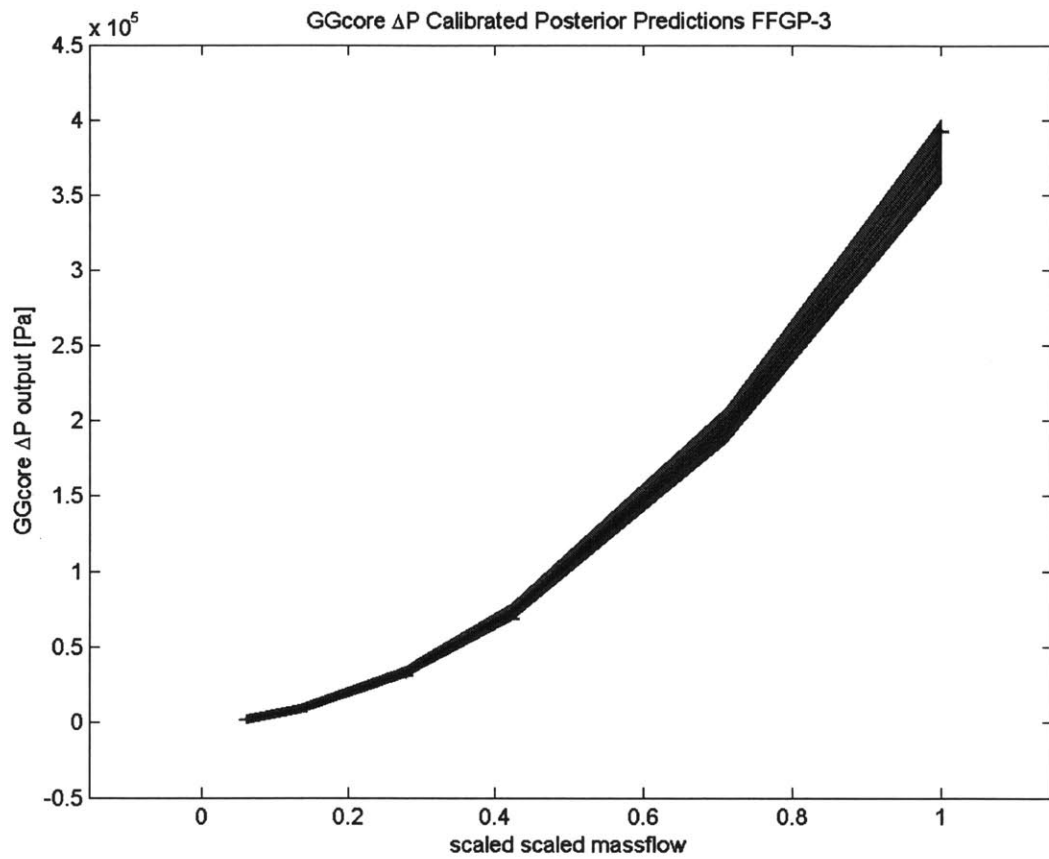


Figure 6-63: Simultaneous calibration prediction results for GGcore

### 6.6.3 Blanket SETs Simultaneous Calibration

Each of the blanket SET models, the IBCT, IBGG, and OBG, used their FFGP 2-factor 3-component emulator as part of the simultaneous calibration. All three share the IB/OB friction parameters (IET uncertain parameters 21, 22, and 23), while the IB channel loss coefficient is unique to the IBGG model, and the OBLA friction parameters along with the OBLA inlet nozzle loss coefficient are unique to the OBG model. As with the core channel SETs simultaneous calibration,  $2 \times 10^5$  AM-MCMC samples were made with the first half discarded as burn-in. For each proposal, the emulator predictions were also made in series and each iteration took approximately  $1.9 \times 10^{-3}$  seconds. The simultaneous calibrated posterior results for the calibrated uncertain parameters, IBCT predictions, IBGG predictions, and OBG predictions are shown in Figures 6-64 through 6-67, respectively. As we expect, the IB/OB turbulent parameters simultaneous calibrated histograms are really similar to the results shown in 6-36 when the IBCT model was calibrated individually. But, as shown in Fig. 6-39 with the IBGG individual calibration results, the posterior IB/OB turbulent parameter histograms were very, very different from the individual IBCT calibration results. So now with the IB/OB turbulent parameters constrained by the IBCT data, the IB channel inlet nozzle loss coefficient (IET uncertain parameter 9) was able to be resolved to a much higher level of precision. The IB channel inlet nozzle loss coefficient posterior mode is in the “direction” of the posterior mode found in the individual IBGG calibration results. This clearly illustrates that it dominates the IB channel pressure drop, since with the IB/OB friction parameters at completely different values its own posterior modes are still fairly close between the two cases. The OBLA inlet nozzle loss coefficient (IET uncertain parameter 14) is pretty close to its posterior histogram determined from the individual OBG calibration process.

This should be expected since the individual OBGG data was shown to have minimal influence from the IB/OB turbulent parameters and the IB/OB laminar shape factor posterior histogram was similar between the individual IBCT and OBGG calibration results. The OBLA friction parameters posterior results are essentially unchanged, signifying that there is not sufficient data in any of the datasets to better resolve them from their priors.

#### **6.6.4 All IET output Simultaneous Calibration**

All three IET output types were calibrated simultaneously using the FFGP 2-factor 3-component emulator for the LPP output, the FFGP 2-factor 4-component emulator for the TTC output, and the FFGP 2-factor 5-component emulator for the OTC output. A total of  $2.5 \times 10^5$  AM-MCMC samples were made with the first  $5 \times 10^4$  discarded as burn-in. I used fewer burn-in samples because a single iteration takes a lot longer when making the IET emulator predictions in series. A single iteration took about 0.0654 seconds so this particular run took about 4.5 hours to complete. But to put this in perspective, the IET model training set which had 500 case runs (initially but 10 crashed) took about 5.5 hours. So even though this is slow compared to any of the other emulator-based calibration processes that I have shown, this is still over 600x faster than using RELAP directly. If the emulator predictions were made in parallel however the simultaneous IET calibration scheme would be over 1100x faster than RELAP.

The simultaneous calibration results are shown in Figures 6-68 through 6-71. Even though these results are calibrated simultaneously with all three outputs, they are not weighted equally. The weighting comes from the FFGP-modified likelihood as the sum of the measurement error and total predictive variance. This fact down-

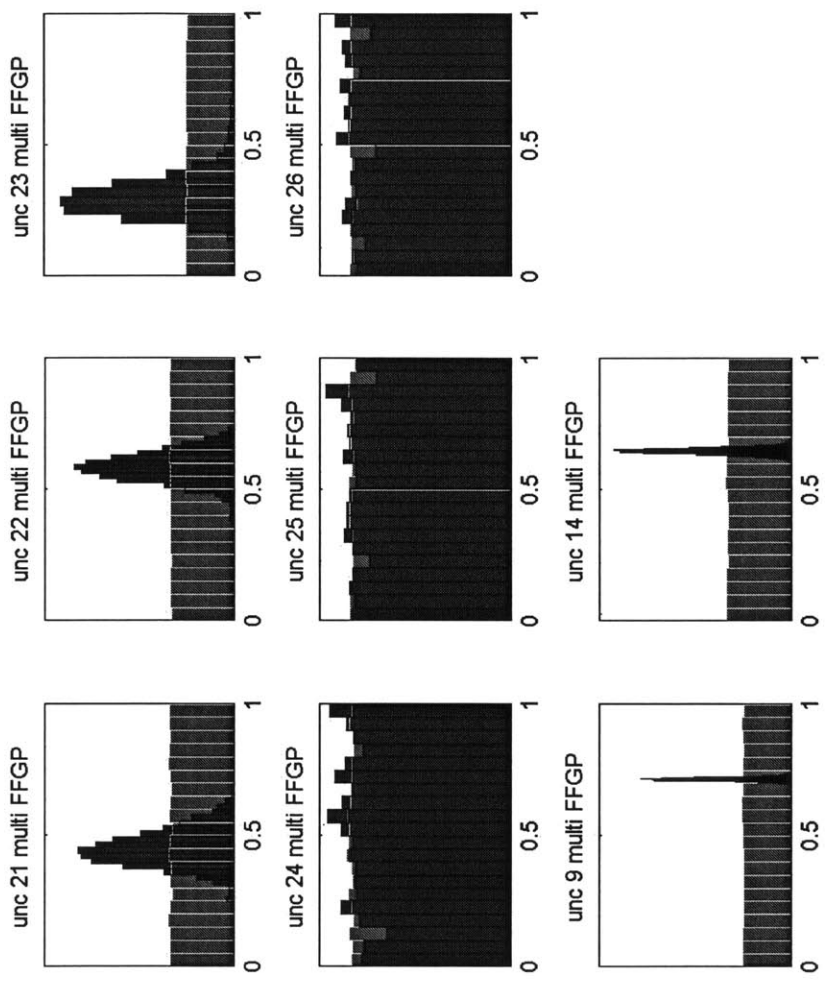


Figure 6-64: Simultaneous Blanket SETs calibration uncertain parameter histograms

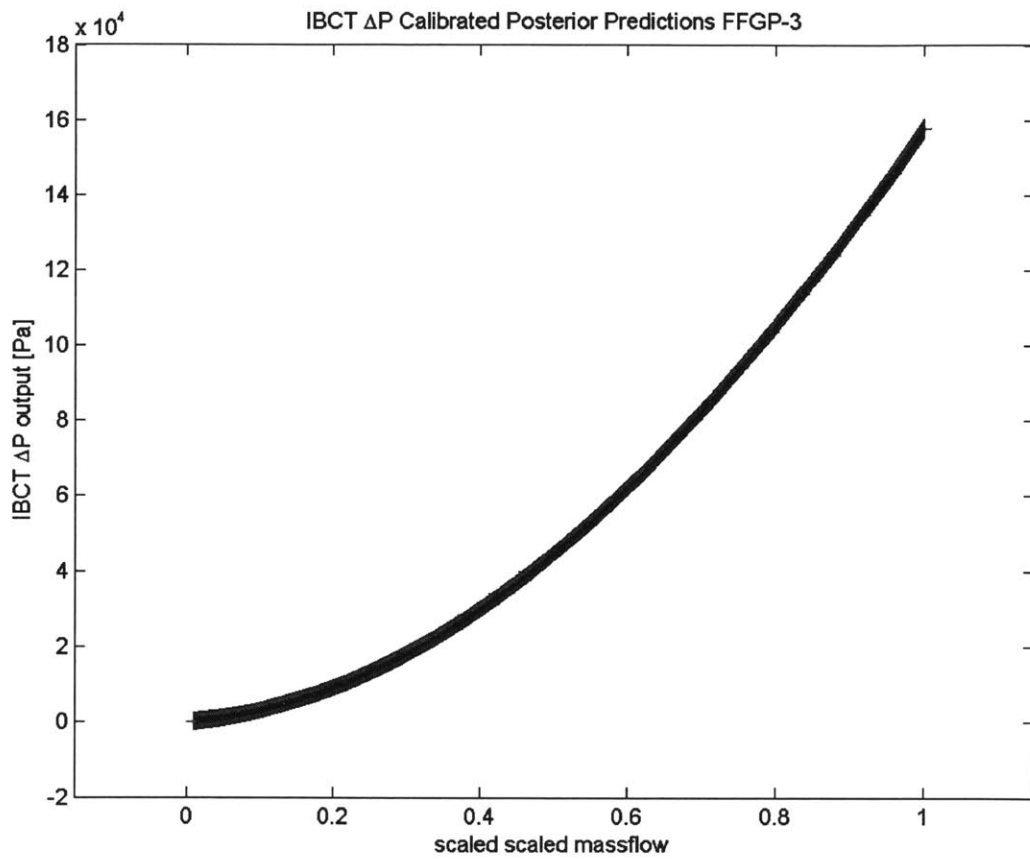


Figure 6-65: Simultaneous calibration prediction results for IBCT

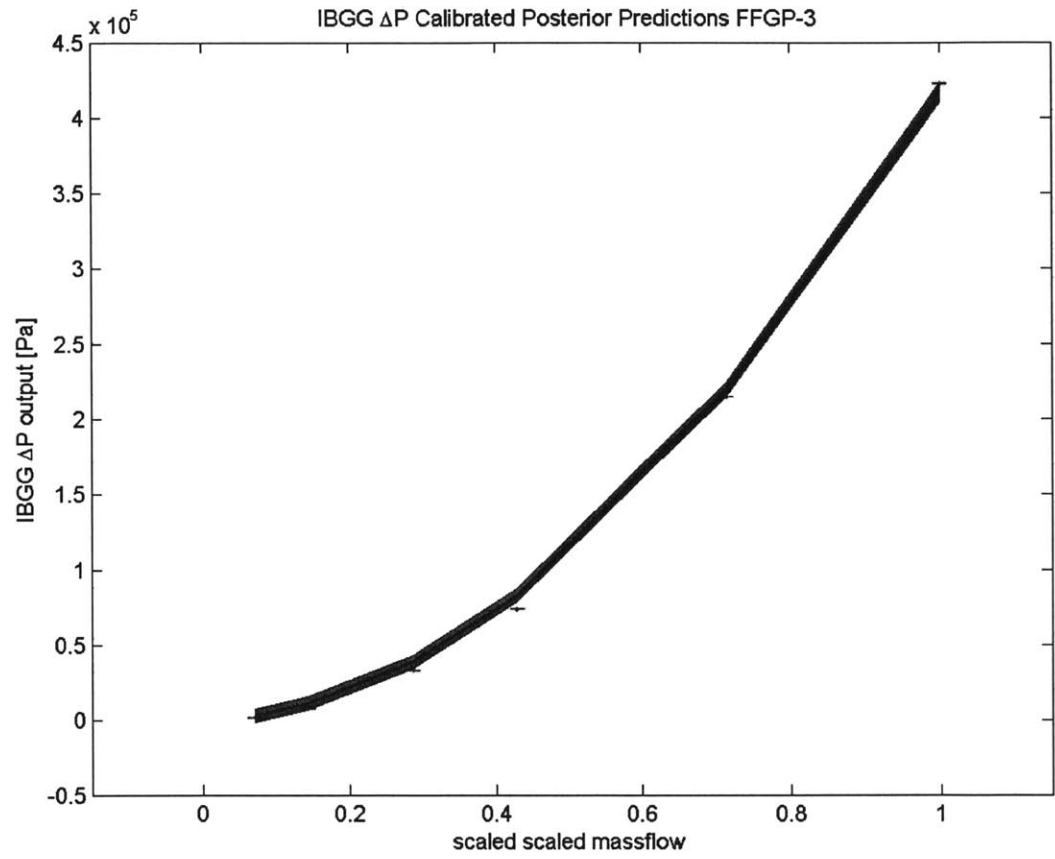


Figure 6-66: Simultaneous calibration prediction results for IBGG



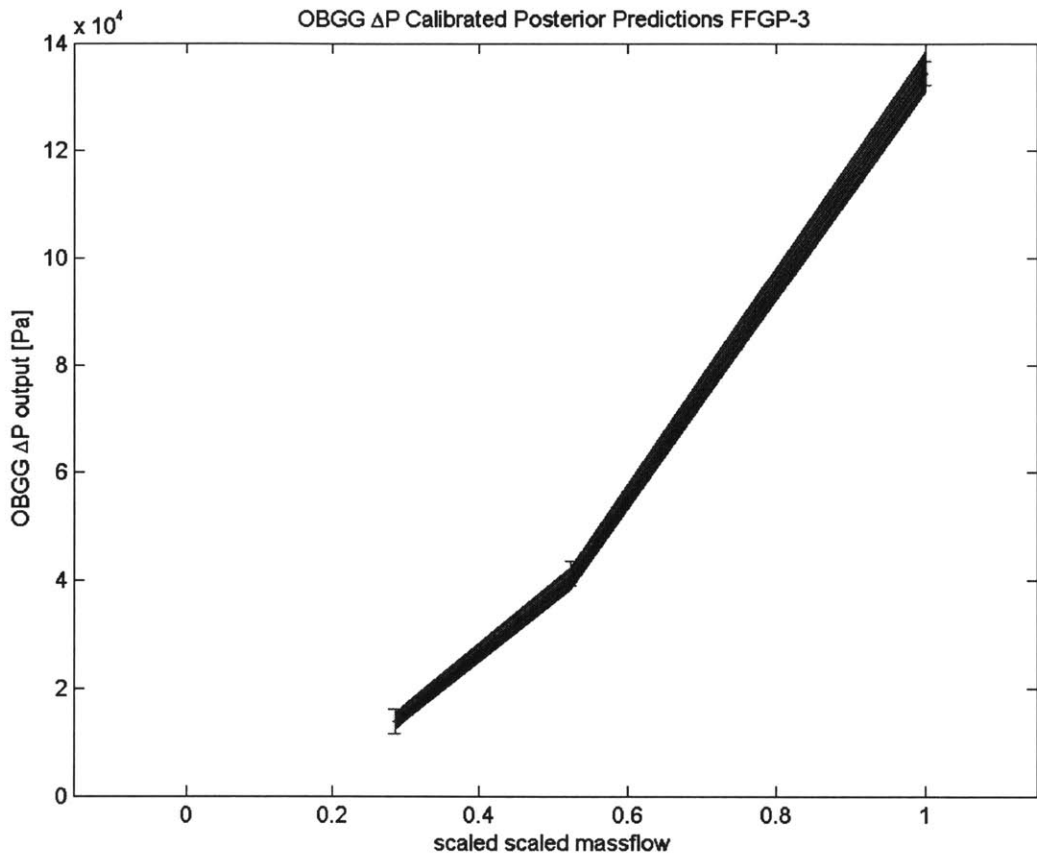


Figure 6-67: Simultaneous calibration prediction results for OBGG

weights the importance of the OTC and TTC data since they have a much larger measurement error as explained earlier. But the TTC and OTC data are down-weighted further because their respective emulators have a higher total predictive variances associated with them than the LPP flow emulator. The net effect means that the uncertain parameters are updated more in line with the LPP flow data than the OTC or TTC data. With the TTC emulator total predictive variance on the order of the measurement error, the simultaneous calibrated predictions effectively regress the TTC data. This effect is even more pronounced with the simultaneous calibrated OTC predictions. The OTC total predictive variance is larger than the measurement error and so as long as the measurement error and total predictive variance “overlap” as shown in Fig. 6-71, that might be “good enough” from the total log-joint-posterior if the LPP flow emulator predictions match the LPP flow data as accurately as possible. Within the uncertain parameter posteriors, the initial core power multiplier (IET uncertain parameter 41) illustrates this concept the best. The individual TTC and OTC calibration results skewed the its initial core power multiplier more to the left, with a broad posterior mode closer to the lower value ranges, while the individual LPP flow calibration results had the posterior more uniformly spread out over the prior range. The simultaneous posterior for the initial core power multiplier, as shown in Fig. 6-68, has its posterior mode almost directly in the center of the range, corresponding to a multiplier value of 1 (remember the uncertain parameter histograms are all the scaled values so the multiplier value of 1 is a scaled value of 0.5). Thus, the TTC and OTC data could not provide enough “effective” information due to the down-weighting from their total predictive variance and measurement error values compared to the LPP flow data. The other important take away from Fig. 6-68 is that many of the uncertain parameters are completely shifted to one side or the other of their prior ranges. This means that my prior

bounding values were not very good and were far too broad. But as stated earlier, those bounding values were very difficult to set and are specific to this particular EBR-II RELAP model.

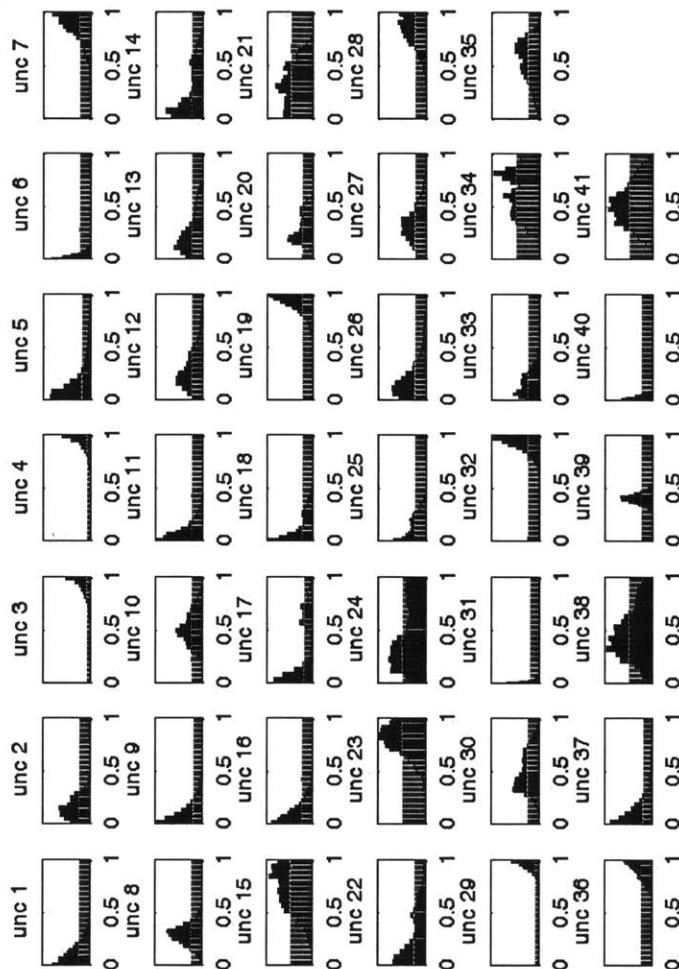


Figure 6-68: Simultaneous All IET output calibration uncertain parameter histograms

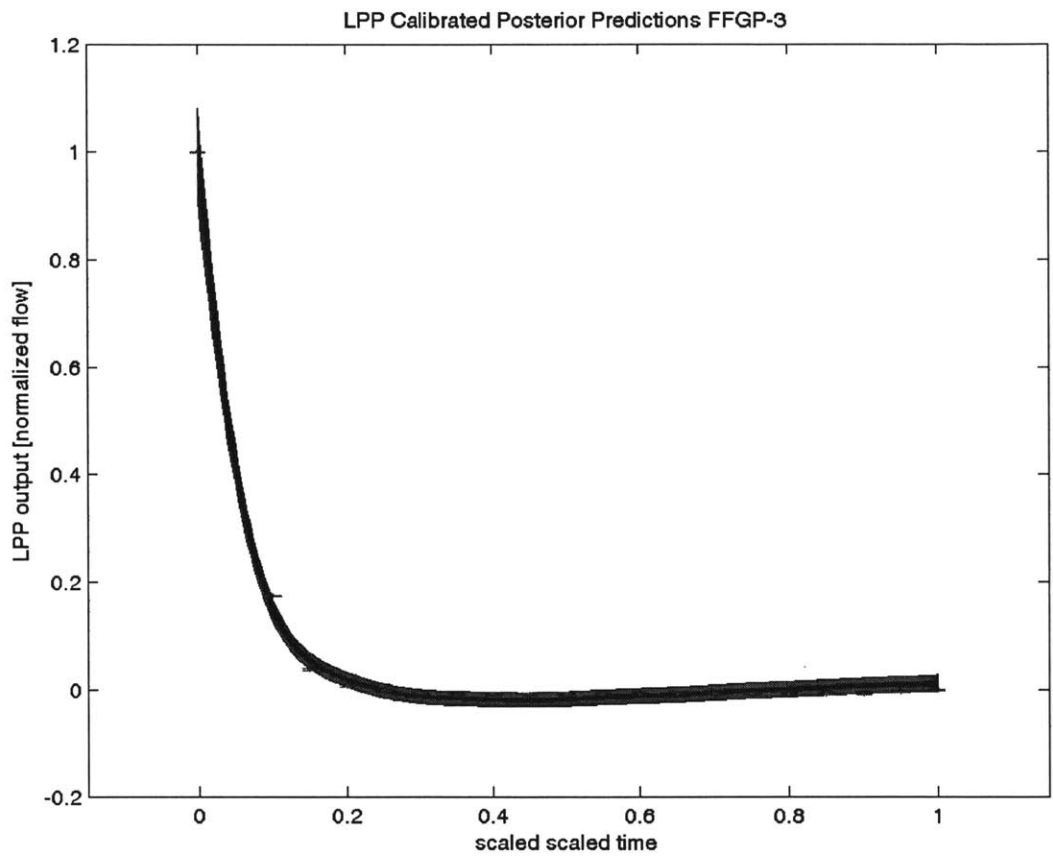


Figure 6-69: Simultaneous calibration predictions results for the LPP output

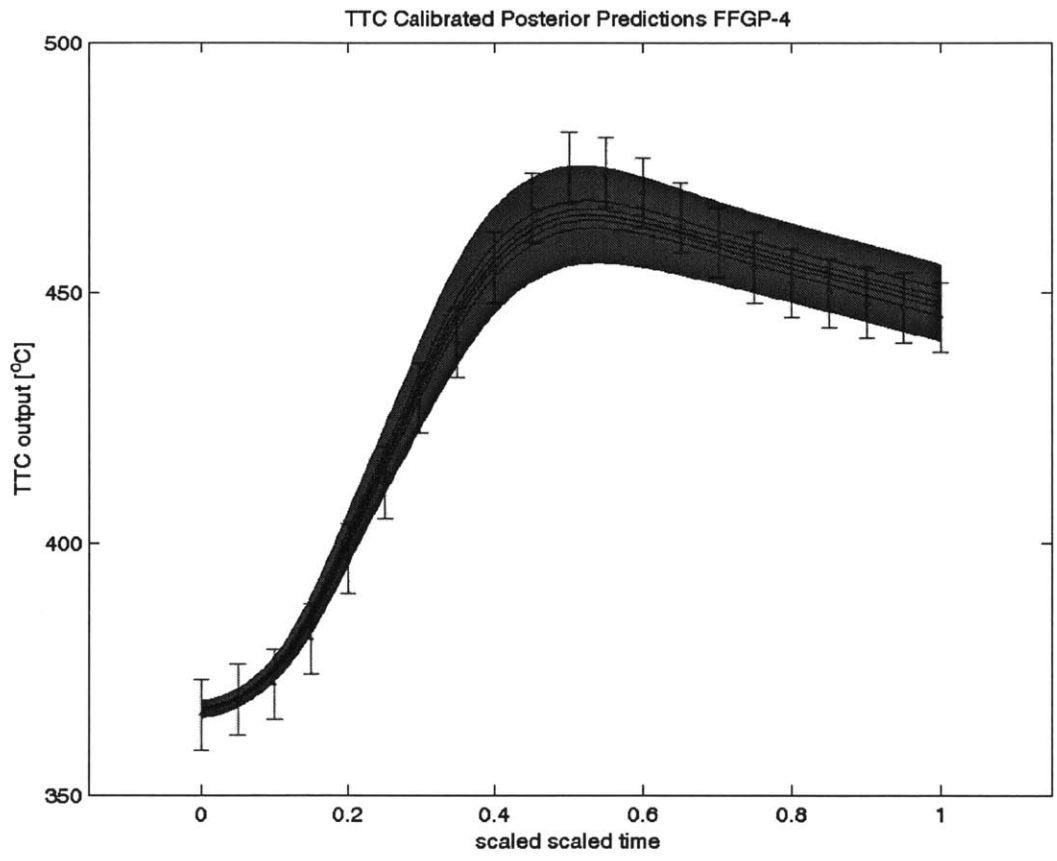


Figure 6-70: Simultaneous calibration prediction results for the TTC output

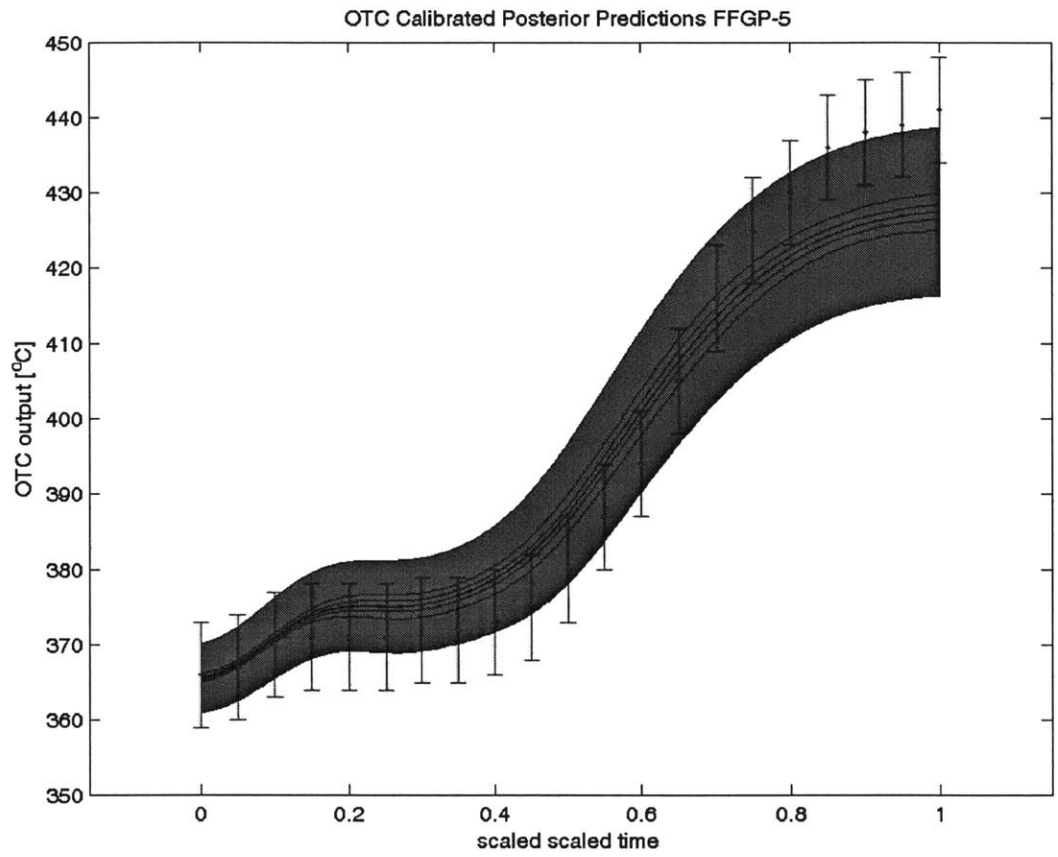


Figure 6-71: Simultaneous calibration prediction results for the OTC output

### 6.6.5 All IET and SET simultaneous calibration

Calibrating all of the SET models and IET outputs simultaneously accounts for as much possible when updating the uncertain parameters. The SET data will be very highly weighted in the complete likelihood function between they have very precise “data” and the SET emulators are all very accurate relative to their respective models. This will greatly restrain the IET output emulators since proposals that increase the IET output modified likelihoods but decrease the SET modified likelihoods will result in that proposal being (most likely) rejected. A single AM-MCMC iteration computed each emulator prediction in series again and took roughly 0.0723 seconds. A total of 8 emulators are used, which means each iteration emulates the behavior of 8 different RELAP models. If all RELAP models were run in series it would take about 56 seconds to complete. But as described earlier, the CTDF model would have to be run 25 times in order to have enough predictions to compare to CTDF data. The same would have to be done for the IBCT model, while the channel SET models would need 6 for GGcore, 6 for IBGG, and 3 for OBG. The total number of RELAP runs (across the 8 models) required to perform a single MCMC iteration is therefore 66. The emulator-based approach is therefore not only faster but would actually make parallelizing simpler since only 8 emulators need to be run in parallel compared to dealing with the 66 models that must be run in parallel each with potentially different run times.

The results shown use only a total of  $10^5$  AM-MCMC samples with the first half discarded as burn-in. Anytime I tried running more samples I would always get chains that had a lower “eye ball” log-marginal likelihood and would not pass the “eye ball” test of Bayesian Model Selection. Due to making predictions with 8 different models and a large number of uncertain parameters, this issue would occur

whether the emulators or the RELAP models themselves were used. But thanks to the emulators I was able to at least try running the MCMC sampling multiple times in order to compare the sampling results. Figures 6-72 through 6-80 provide the simultaneous IET-SET calibration results for the uncertain parameter histograms and each of the output types. The “down-weight” concept applies to even some of the OBGG SET results as shown in Fig. 6-77. The OBGG SET has the a larger measurement error value than the others, so the sampling will try and regress the OBGG data if it means the CTDF and IBCCT models are more accurate.

The major take away from the posterior uncertain parameter histograms is that their prior bounding values were far too wide. For those that give posteriors completely shifted to one side, such as the low pressure throttle valve loss coefficient (IET uncertain parameter 6), basically 80% of the prior range on their values were almost “meaningless” since those prior values were nowhere near the posterior values. The overally large prior ranges on many of the IET-only parameters area clearly contributing to the still large total predictive variance of the OTC output. With the SET-shared uncertain parameters constrained so tightly, many of the IET-only uncertain parameters also become very tightly as well. The reason for this, is most likely that for TTC and OTC emulator total predictive variance predictive explains a large amount of the data variation and so add very little “effective” information relative to finding values that match the LPP flow data the best. In order to decide if these are the absolute best posterior uncertain parameter values the TTC and OTC emulator predictive variance must be reduced. But even so, in the Bayesian context these posterior distributions represent the updated state-of-knowledge about the uncertain parameters that did not exist before performing the IET-SET simultaneous calibration process.



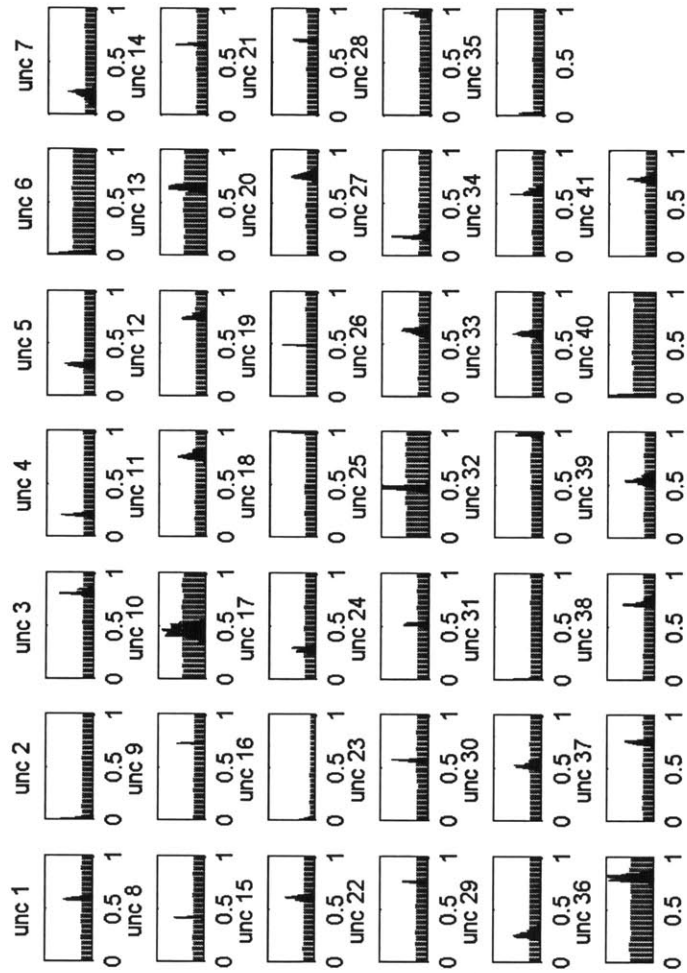


Figure 6-72: IET-SET simultaneous calibration uncertain parameter histograms

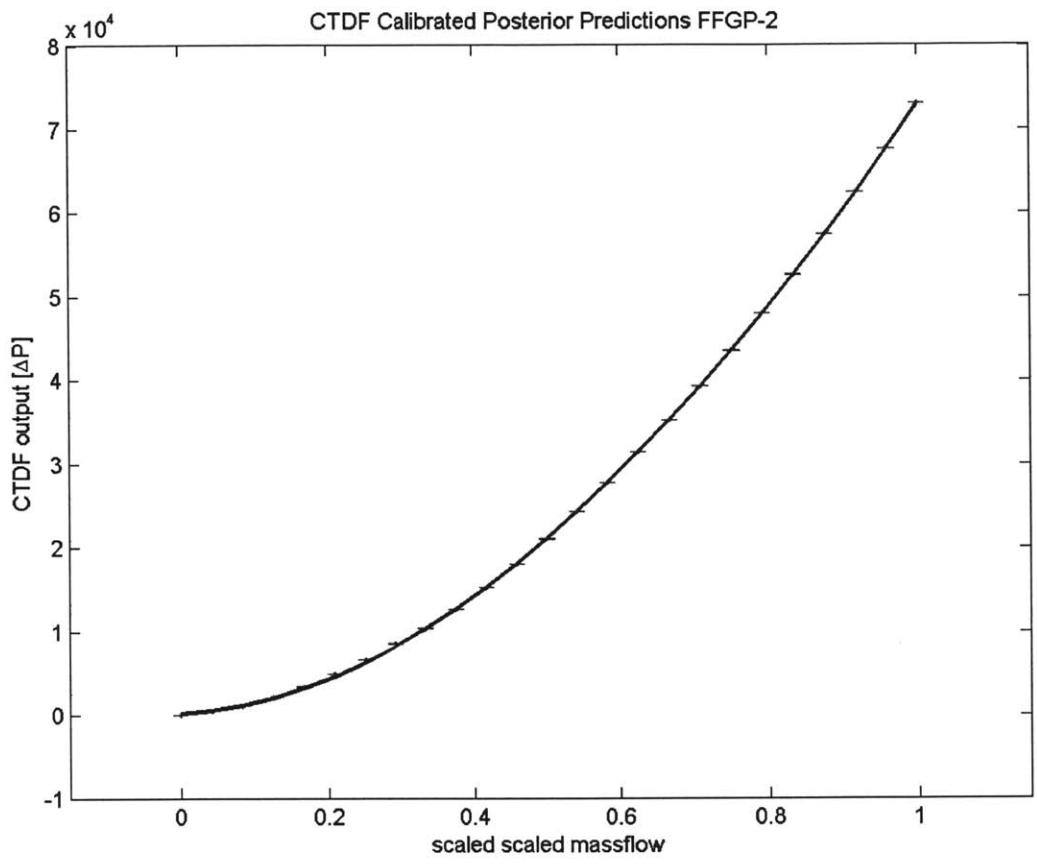


Figure 6-73: IET-SET simultaneous calibration prediction results for CTDF

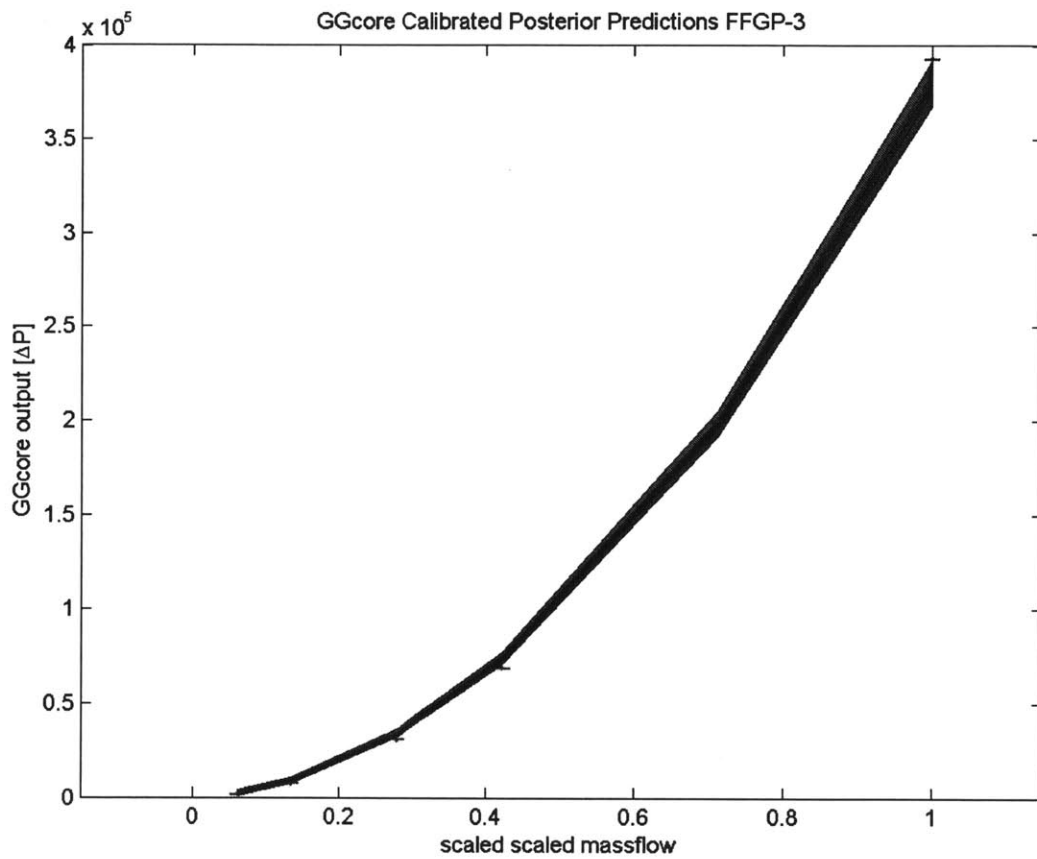


Figure 6-74: IET-SET simultaneous calibration prediction results GGcore

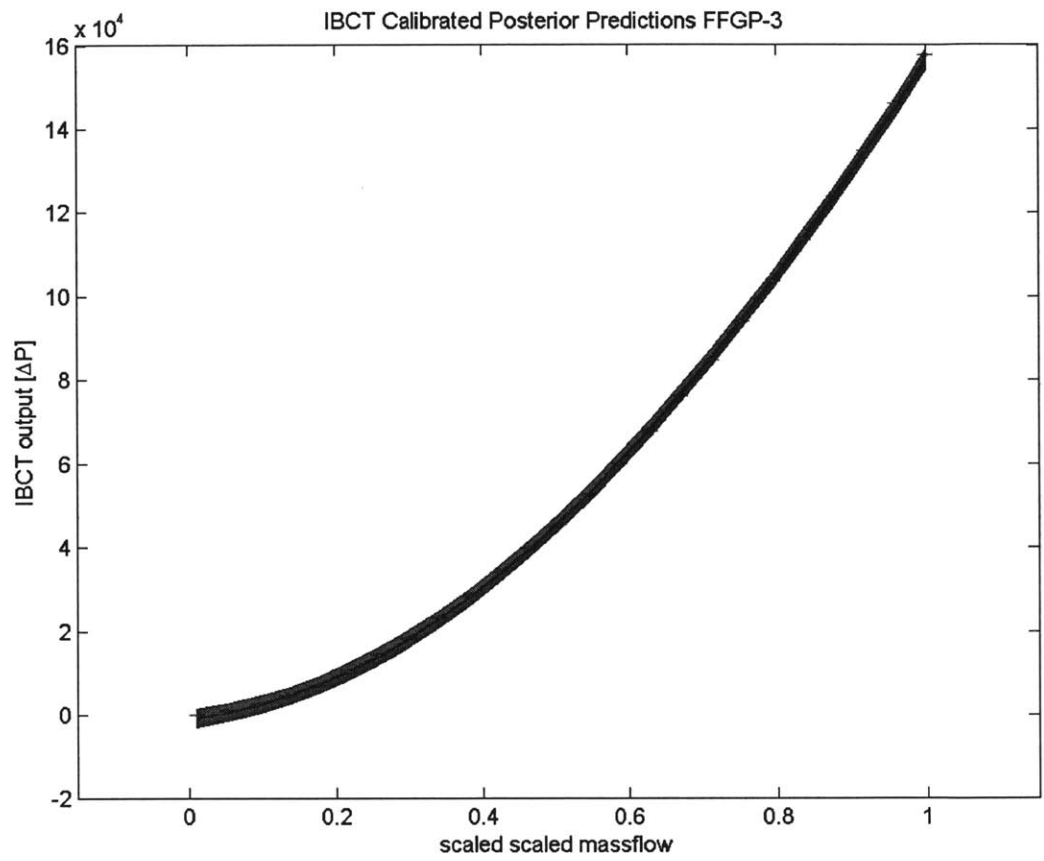


Figure 6-75: IET-SET simultaneous calibration prediction results IBCT

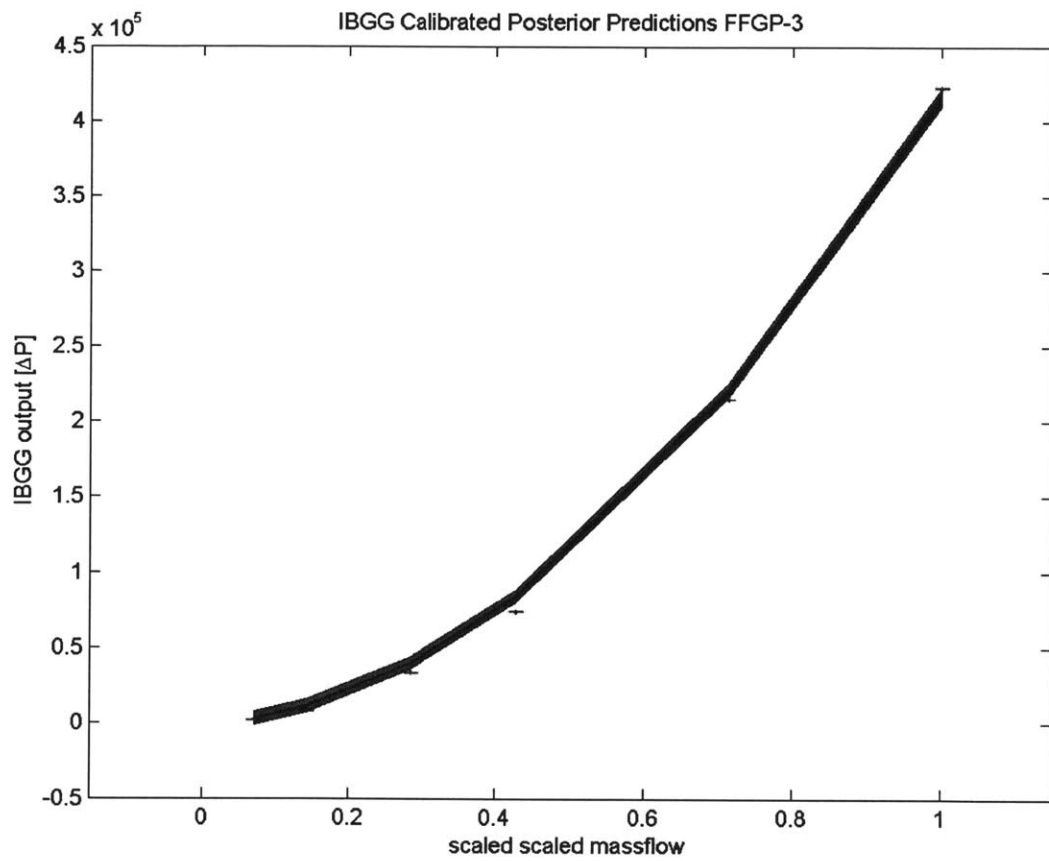


Figure 6-76: IET-SET simultaneous calibration prediction results IBGG

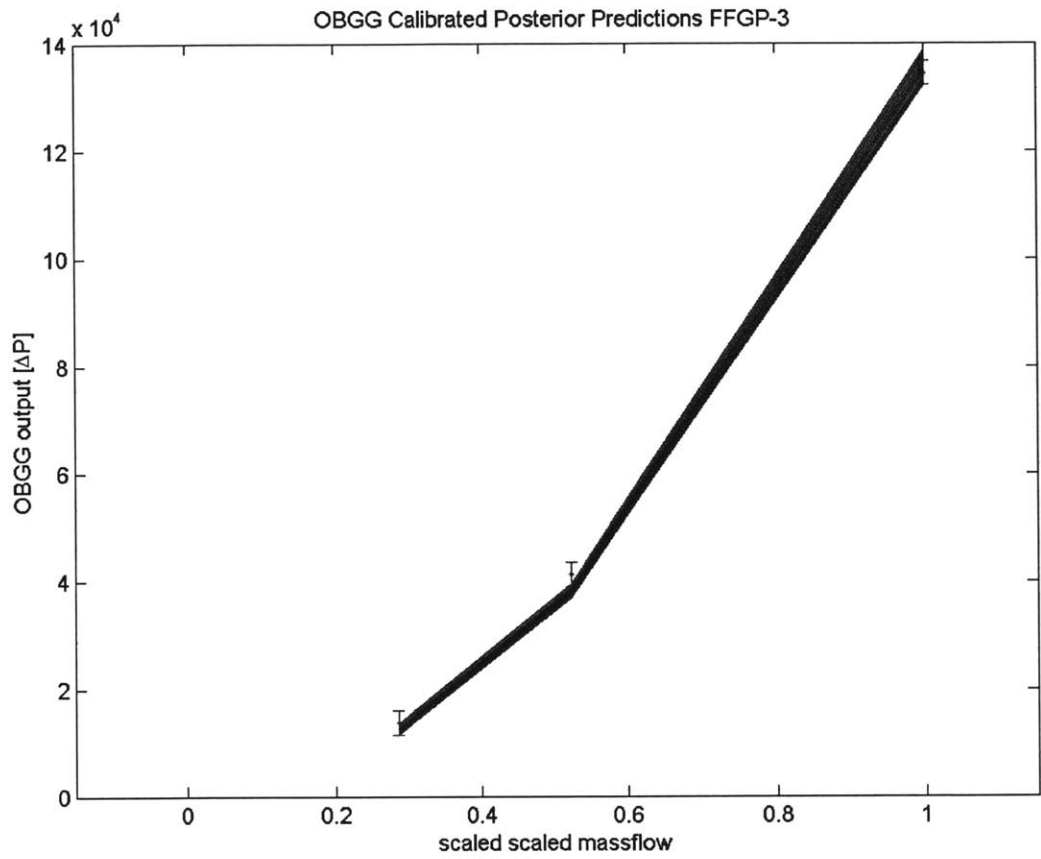


Figure 6-77: IET-SET simultaneous calibration prediction results OBGG

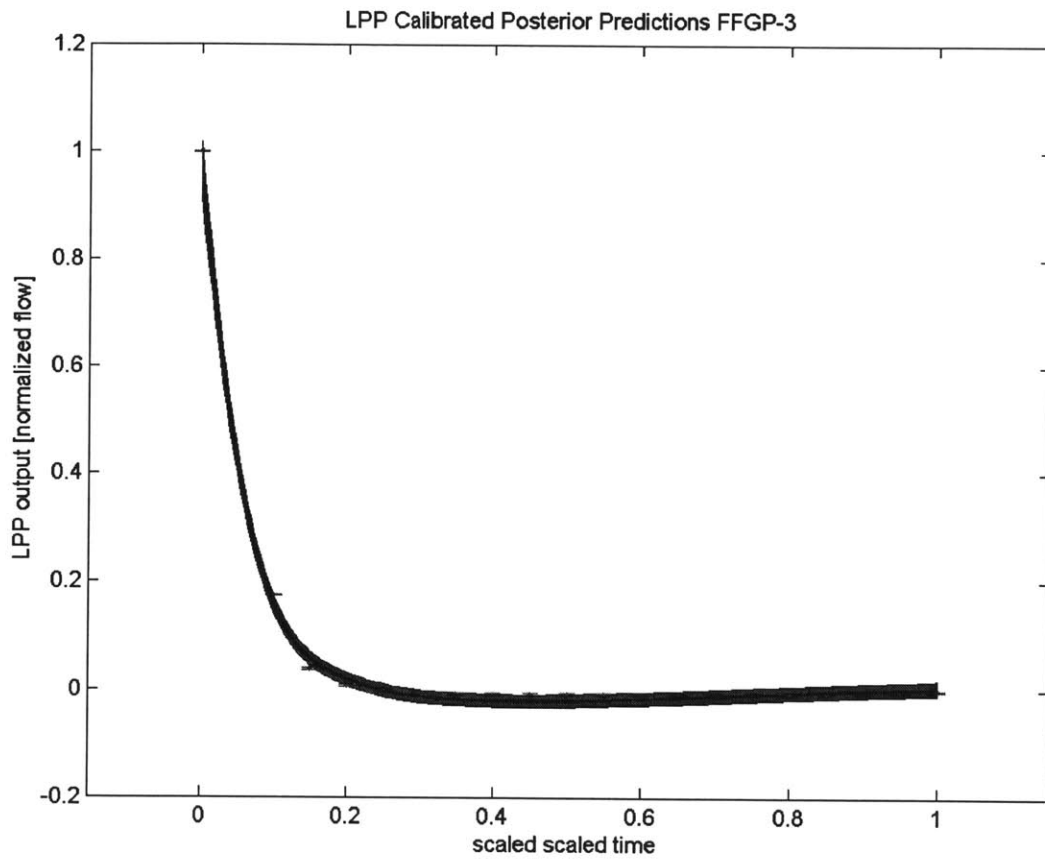


Figure 6-78: IET-SET simultaneous calibration prediction results LPP

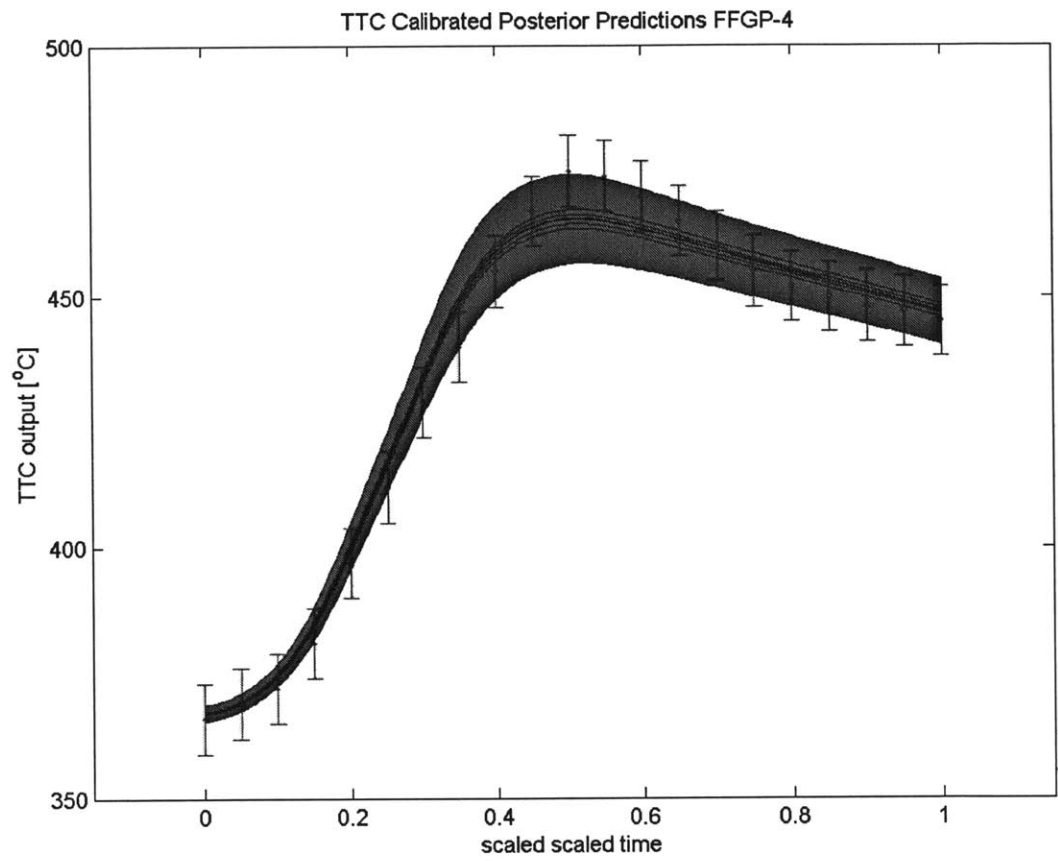


Figure 6-79: IET-SET simultaneous calibration prediction results TTC



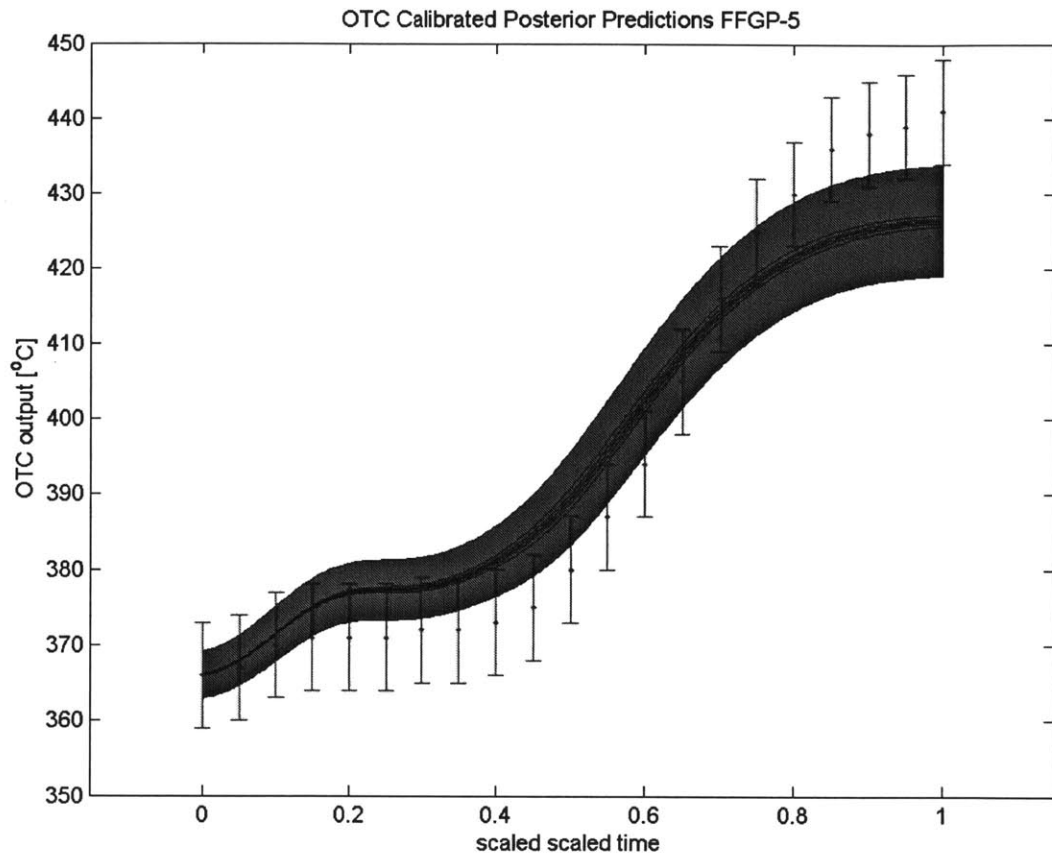


Figure 6-80: IET-SET simultaneous calibration prediction results OTC



# Chapter 7

## Summary, Conclusions, and Future Work

### 7.1 Summary

Bayesian inference provides a mathematically and statistically rigorous framework of solving inverse problems, which would otherwise be ill-posed or are analytically intractable. In the context of uncertainty quantification (UQ), observational data can be used to calibrate computer model predictions and infer out the numerous parameters within the computer model thereby properly performing backwards UQ. The resulting posterior distributions combine data with the expert judgement encoded within the priors. But for even relatively “fast” computer models practical implementation of Bayesian inference with Markov Chain Monte Carlo (MCMC) would simply take too long. A computer model that takes 1 second to run but needs  $10^5$  MCMC samples to achieve “good” mixing rates would take over 27 hours to complete. Surrogate models that emulate (the *emulator*) the behavior of the in-

put/output relationship of the computer model but are very computationally cheap allow MCMC sampling to be possible. An emulator that is 1000x faster than the computer model would need only 100 seconds to perform the same number of MCMC samples. As the computer model run time increases, the surrogate model becomes even more important because MCMC sampling would simply become impractical to even attempt.

The general objective of this thesis was the development and implementation of an emulator based calibration process for nuclear reactor safety analysis codes. The work was broken into three distinct parts: development of an emulator to help model the complex time series modeling of a reactor safety analysis code, a Quantitative PIRT methodology to help identify the dominant uncertain parameters thereby reducing the number of parameters involved in the calibration process, and applications of the emulator based calibration process to realistic safety analysis scenarios. Furthermore, we wished to combine multiple “levels” of data using Separate Effect and Integral Effect Tests (SETs and IETs) simultaneously. The hope was that the SET data would constrain the parameter distributions to only physically acceptable values during the MCMC sampling. The IET data would be prevented from simply finding unphysical parameter values that allow the emulator (and thus the computer model) predictions to match the data.

Chapter 2 provided a basic introduction to approximate Bayesian inference with MCMC sampling. A general discussion on the merits of Bayesian inference over classical frequentist inference was provided before describing why MCMC sampling is needed to implement Bayesian inference. A method of manufactured solutions demonstration problem - using a simple friction factor model - was used to compare the performance of several common MCMC schemes. The results showed that MCMC sampling was indeed capable of finding the “true” parameter values. But

since Bayesian inference deals with probability distributions and not simply point estimates, the parameter posterior distributions provided a sense of how uncertain the results were given the available data.

Chapter 3 described the Quantitative Phenomena Ranking and Identification Table (QPIRT) methodology. The QPIRT consists of two steps, the Top-Down step which focuses on the governing equations in the computer model and a Bottom-Up step which focuses on specific constitutive relations and closure models within the computer model. The Top-Down step ultimately tracks the dominant physical processes that influence the figure-of-merit (FOM) within the system over the duration the transient of interest. The Top-Down step recasts the computer model governing equations in control volume balance form at two hierarchical levels: the local and system levels. The local level equations are problem dependent focusing only on the physical phenomena that directly influence the FOM. The system level equations capture the effects “carried throughout” the system, and rank the physical processes that dominate the system level response. The QPIRT methodology was demonstrated on a Total Loss of main Feedwater Flow (TLOFW) accident with subsequent feed and bleed. The Top-Down step identified the key physical phenomena that influenced the FOM, the peak clad temperature (PCT). The Bottom-Up step then examined the specific constitutive relations used by the computer model, RELAP5, to model each of the physical processes identified in the Top-Down step. The result was a list of empirical correlations for each of the key physical phenomena that influence response of the PCT during the transient, as viewed by the computer code.

Chapter 4 goes into detail on the formulation of the surrogate models, or emulators developed in this work. All of the emulators are based on the Gaussian Process (GP) and so the first part of Chapter 4 summarizes the math behind standard Gaussian Process Regression (GPR). One major benefit of the GP framework is that it

gives a very simple estimate of the uncertainty in a prediction. Therefore the additional uncertainty of using the emulator in place of the computer model can be included in the calibration process. But, to overcome the limitations of the standard GPR emulator, Function Factorization with Gaussian Process (FFGP) priors was used. The basic premise of the FFGP emulator is to assume the output can be represented by the sum of a product of factors. These factors are given GP priors which must be then learned from the training data. One of the benefits of this formulation is that each factor can “work” over a smaller subspace of the total input space. Thus, even though the FFGP emulator is far more complex than the standard GPR emulator, it is more efficient at handling very large dimensional problems. FFGP models are also known as Gaussian Process Factor Analysis (GPFA) models, and as discussed in Chapter 4, they have been used extensively in other fields to decompose and analyze various data structures. But this thesis focused on using them to emulate a computer model and so the work focused on the practical issues involved with building and making predictions with them, as quickly as possible. Several approximations were made to the FFGP emulator which greatly improved their speed compared to most general setup. These approximations were also used to develop an FFGP-modified likelihood structure for uncertain parameter calibration. It was then very simple to estimate the approximate FFGP prediction contribution to the total predictive uncertainty. The same method of manufactured solutions problem from Chapter 2 was reworked in Chapter 4 but now with the emulators. The results showed that the FFGP emulator is flexible enough to accurately emulate the computer model (the simple friction factor model) and achieve posterior distributions on the uncertain parameters that almost exactly match those shown in Chapter 2.

Chapter 5 was the first application of the emulator based calibration process to a relevant safety analysis scenario. The computer model was a numerical solver for a

blowdown of a gas tank and the goal was to calibrate the uncertain parameters within the model using data taken from an actual gas blowdown experiment. The QPIRT process was demonstrated as well showing how the number of uncertain parameters was limited to 8. From here the emulators were constructed and used in place of the numerical model in the MCMC sampling. The results and performances of the various emulator types were compared, showing how the almost all uncertainty in the model prediction came from actually only 2 of the parameters. A single case run of the numerical solver took approximately 2.16 seconds, so running the numerical solver directly with  $10^5$  MCMC samples would have taken approximately 60 hours to complete. The total time to create the training set, build the most complex FFGP emulator and make predictions with that emulator using  $10^5$  MCMC samples took less than 10 minutes. With the emulator based approach in place now far more complex computer models can be calibrated.

Chapter 6 presents the emulator based calibration process applied to an EBR-II loss of flow transient. The EBR-II transient starts from decay power conditions, so only thermal-hydraulic phenomena needed to be considered. After presenting a brief introduction to the EBR-II facility, the RELAP model for the transient of interest of described (the IET model). With a total of 41 uncertain parameters within the IET model and 3 output types, SETs would greatly aid the calibration process. To simplify the work however, “pseudo” SETs with “data” generated from empirical correlations rather than actual experimental data were incorporated into the calibration process. The “pseudo” SETs effectively allowed calibrating the RELAP user-defined friction models with empirical friction factor correlations more relevant to the EBR-II geometry than the friction factor correlations available within RELAP directly. A total of 5 “pseudo” SETs were used and which specific uncertain parameters they calibrated were described in detail. The emulator based calibration

results for each of the SETs were shown as well as a simple “eye ball” test heuristic to decide which FFGP emulator type is the “best”. Then the IET RELAP model was calibrated using each of its 3 output types individually. Afterward the simultaneous calibration of all SETs and IET outputs was described by first showing the specific likelihood structure used for simultaneous calibration. The final results show the development of the 41 uncertain parameter posterior distributions from their uniform priors through the individual SET and IET output calibrations to their IET-SET simultaneously calibrated posterior distributions. The SETs were able to “learn” their specific uncertain parameters to a high degree of precision. This therefore allowed the IET model to “learn” out the remaining parameters to a high degree of precision as well. A single RELAP run of the IET model took approximately 40 seconds and so without the emulators, it would have been almost impossible to calibrate the IET RELAP to the data. Additionally, the emulators made including the SETs in the calibration process due to the approximate FFGP-modified likelihood structure. Limitations of the current setup however were noted with the predictive results showing the emulators contributing a large fraction of the total predictive uncertainty to one of the IET model outputs.

## 7.2 Conclusions

The emulator based calibration process was shown to make Bayesian inference with MCMC sampling possible. In each of the demonstrations presented in this thesis, the largest portion of the total computational time was actually creating the emulator training set. Building and making predictions with the emulators, although not trivial for the EBR-II IET emulators, were much faster than simply using the computer model (RELAP) directly. The posterior model predictions therefore are



far more useful than simply setting many of the numerous uncertain parameters to fixed values. By fully accounting for all uncertainty due to the uncertain parameters any discrepancy between the posterior predictions and the observational data can only be due to discrepancies inherent to the formulation of the model itself. This “model form” uncertainty, is a larger issue that without fully propagating the uncertain parameter uncertainty would be very difficult to assess on its own. But thanks to emulator based calibration, the uncertain parameter distributions are their posterior distributions conditioned on the available observational data.

## 7.3 Recommendations for Future Work

Each of the recommendations for future work revolve around the issue of improving the FFGP emulators even further. Because although they are very powerful, as pointed out in various parts of this thesis, they can be improved further. These improvements then lend to applying them in even more useful ways beyond just making predictions as part of MCMC sampling.

### 7.3.1 Emulator Training with Variational Bayes

As seen in Chapter 4, building the FFGP emulators is quite complex. This issue is only compounded further by the “art” of knowing when to stop MCMC sampling. An optimization based approach to building the FFGP emulators would provide a more useful “convergence” stopping criteria. As described in Chapter 4, the approximate FFGP predictive distributions estimate the training latent variable posterior distributions as Gaussians. So rather than estimating those Gaussians empirically, that structure could be assumed into their posterior distribution from the start. This is

essentially the concept behind Variational Bayes (VB) methods, where more complex posterior distributions are approximated by known (and easier to work with) distributions, such as Gaussians [8], [24]. This allows estimating the training marginal likelihood and so the training algorithm becomes essentially an optimization algorithm to maximize the training marginal likelihood. Multiple references have applied VB algorithms to GP models and other latent variable models [51], [52]. With the training posterior distributions easier to estimate, it becomes easier to generalize the predictive distribution to more factors. This would greatly enhance the flexibility of the current FFGP emulator setup.

### 7.3.2 Bayesian Monte Carlo

Bayesian Monte Carlo (BMC) is an alternative to Monte Carlo sampling to approximating integrals [53]. BMC uses a GP prior over the function that is getting integrated. In the context of UQ, BMC can be used to analytically propagate uncertain parameter distributions onto the output, as long as the parameter distributions are Gaussians. So it can help provide a very fast approximation to the prior uncertainty on the output of interest. The approximate FFGP posterior predictive distribution can be used in place of the GP prior, used in Ref. [53]. This could allow BMC to capture far more complex input/output relationships, required for reactor safety analysis codes.

### 7.3.3 Non-Informative Priors

Non-informative priors help remove any *a priori* bias on the parameter values from the prior distribution. Although the priors can be very important as ways of encoding prior expert opinion belief, it can be a useful exercise to use a non-informative prior

and compare the resulting posterior distributions. The simplest non-informative prior is an improper “flat” prior, just as the “flat” hyperprior used on the FFGP hyperparameters in Chapter 4. True non-informative priors do not add absolutely no bias, but rather bias the parameters to stay away from values outside the data [5]. Non-informative priors therefore depend on the likelihood function, and are in general quite difficult to compute [4]. Since the likelihood function cannot be explicitly written down for RELAP, the emulator-modified likelihood functions in Chapter 4 must be used. This is therefore an additional benefit that only the emulator-based approach is capable of. BMC techniques will most likely be needed because many non-informative priors require evaluating complex integrals.

### 7.3.4 Dynamic Validation + Training + Calibration

The IET-SET calibration results in Chapter 7 show that my prior bounding values on many of the uncertain parameters were very poor. Many of the parameters’ posterior histograms were completely skewed to one side of their prior bounds. Thus, a majority of the training values were effectively useless for many of the parameters. Ideally, we would want the training set to shift around to a region of the state-space to maximize the “information” from each training point, as well as to reduce the number of points required. Rasmussen developed an approach to use GPs speed up the computation of complex integrals by combining GPs with Hamiltonian Monte Carlo (HMC) [39]. The GP was used to approximate the derivatives of the complex function, and those derivatives were used in place of the true derivatives in the HMC proposal distribution (where HMC is the same MCMC scheme used to sample the FFGP training latent variables described in Chapter 4). Given an initial training set, Rasmussen first modified the likelihood function so that it tried to minimize the

error between the prediction and the data, while maximizing the GP predictive variance. This was used to identify new training point values until the GP was accurate to within some tolerance level. Additionally, this acts as a sort of dynamic validation since the additional training points are found within the uncertain parameter posterior state-space that the GP was uncertain about.

This setup could be adapted to the approximate FFGP posterior predictive distribution. Where now the RELAP output gradient relative to the uncertain parameters is estimated using the approximate FFGP posterior predictive distribution. After each new training point is identified, the FFGP emulator must be rebuilt and so this could only be used once the VB training algorithm is completed. This could help greatly improve the accuracy of the emulator since the training process itself actually quantifies what posterior uncertain parameter values give the greatest predictive uncertainty.

The additional benefit of this setup is the uncertain parameter calibration process could be carried out using the HMC scheme. Realistic safety analysis problems are similar to the EBR-II RELAP model in Chapter 7 with a large number of uncertain parameters, with many of them highly correlated. HMC is perfect then for such a scenario, but we must be able to compute the gradient. That is not currently available in RELAP. Even if an adjoint solver was implemented in RELAP, due to the highly non-linear nature of thermo-hydraulic modeling, the RELAP model would have to be run for each HMC proposal. Thus, it would become impossible to use RELAP for HMC. So the emulator could allow using HMC to help draw samples that would be too difficult with other MCMC schemes.

### 7.3.5 Uncertain Parameter Calibration Without MCMC

Since the FFGP emulator could be trained using VB rather than MCMC, it is possible to do something similar for the uncertain parameter calibration itself. The optimal maps procedure [54] actually does something similar to this idea already. The optimal maps approach finds a mapping function that “pushes” the prior to the posterior, by monitoring the marginal likelihood. Therefore the gradient information of the likelihood function, similar to HMC, must be computable. Rather than finding a mapping function, the approximate FFGP posterior predictive distribution could be used just as it is in the FFGP-modified likelihood format of Chapter 4. The VB framework could be modified to then try and maximize the marginal likelihood between the observational data and the emulator prediction. Although the optimization would be complex, a stopping criteria exists by monitoring the marginal likelihood convergence rate. Additionally, it would be very easy to compare the performances of multiple emulator types since the actual marginal likelihood would be estimated, so the “eye ball” test described in Chapter 5 is no longer necessary.



# Bibliography

- [1] L. Biegler, G. Biros, O. Ghattas, M. Heinkenschloss, D. Keyes, B. Mallick, Y. Marzouk, L. Tenorio, B. van Bloemen Waanders, and K. Willcox, *Large-scale inverse problems and quantification of uncertainty*. Chichester : John Wiley & Sons, cop., 2011.
- [2] D. S. Sivia, *Data Analysis: A Bayesian Tutorial Second Edition*. Oxford: Clarendon (Oxford Univ. Press), 2006.
- [3] E. T. Jaynes and G. L. Bretthorst, Eds., *Probability theory : the logic of science*. Cambridge University Press, 2003.
- [4] A. Gelman, J. B. Carlin, H. S. Stern, and D. B. Rubin, *Bayesian Data Analysis Second Edition*. Chapman & Hall, 2003.
- [5] G. E. P. Box and G. C. Tiao, *Bayesian Inference in Statistical Analysis*. John Wiley & Sons, 1973.
- [6] D. L. Kelly and C. L. Smith, "Bayesian inference in probabilistic risk assessment: The current state of the art," *Reliability Engineering & System Safety*, vol. 94, pp. 628–643, 2009.

- [7] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, *Markov Chain Monte Carlo in Practice*. Chapman and Hall, 1996.
- [8] K. P. Murphy, *Machine learning: a probabilistic perspective*. Cambridge, MA: MIT Press, 2012.
- [9] S. Brooks, A. Gelman, G. Jones, and X. Meng, *Handbook of Markov Chain Monte Carlo*. Taylor and Francis, 2011.
- [10] D. Lunn, A. Thomas, N. Best, and D. Spiegelhalter, “Winbugs – a bayesian modelling framework: concepts, structure, and extensibility,” *Statistics and Computing*, vol. 10, pp. 325–337, 2000.
- [11] G. O. Roberts and J. S. Rosenthal, “Optimal scaling for various metropolis-hastings algorithms,” *Statistical Science*, vol. 16, no. 4, pp. 351–367, 11 2001.
- [12] H. Haario, E. Saksman, and J. Tamminen, “An adaptive metropolis algorithm,” *Bernoulli*, vol. 7, pp. 223–242, 1998.
- [13] Managing Uncertainty in Complex Models. [Online]. Available: <http://www.mucm.ac.uk/>
- [14] A. Beskos, F. Pinski, J. Sanz-Serna, and A. Stuart, “Hybrid monte carlo on hilbert spaces,” *Stochastic Processes and their Applications*, vol. 121, no. 10, pp. 2201 – 2230, 2011. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0304414911001396>
- [15] A. J. Keane and P. B. Nair, *Computational Approaches for Aerospace Design: The Pursuit of Excellence*. John-Wiley & Sons, 2005.



- [16] N. Zuber, "Appendix d: a hierarchical, two-tiered scaling analysis, an integrated structure and scaling methodology for severe accident technical issue resolution," US Nuclear Regulator Commission, Washington, DC 20555, Tech. Rep. NUREG/CR-5809, November 1991.
- [17] *RELAP5-3D Code Manual*, Rev. 2.2 ed., INEEL-EXT-98-00834, 2003.
- [18] T. Liu and S. Chiang, "Safety evaluation of primary-side bleed-and-feed scenarios by an experimental loss of feedwater event at the iist facility," *Nuclear Technology*, vol. 136, pp. 204–220, 2001.
- [19] T.J., "Iist and bethsy counterpart tests on pwr total loss-of-feedwater transient with two different bleed-and-feed scenarios," *Nuclear Technology*, vol. 137, pp. 10–27, 2002.
- [20] C. Billa, F. D'Auria, and G. Galassi, "Accident management for a loss of feedwater in a pressurized water reactor," *Nuclear Technology*, vol. 98, pp. 277–288, 1992.
- [21] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and analysis of computer experiments," *Statistical Science*, vol. 4, no. 4, pp. 409–435, 1989.
- [22] W. J. Welch, R. J. Buck, J. Sacks, H. P. Wynn, T. J. Mitchell, and M. D. Morris, "Screening, predicting, and computer experiments," *Technometrics*, vol. 34, no. 1, pp. 15–25, February 1992.
- [23] N. Cressie, *Statistics for Spatial Data*. John Wiley & Sons, 1993.
- [24] C. Rasmussen, *Gaussian Processes in Machine Learning*, ser. Lecture Notes in Computer Science, U. v. L. Bousquet, O. and G. Rätsch, Eds. Heidelberg: Springer-Verlag, 2004, copyright by Springer.

- [25] M. Schmidt, "Function factorization using warped gaussian processes," in *Proceedings of the Twenty-Sixth International Conference on Machine Learning*. New York: ACM, 2009.
- [26] J. Luttinen and A. T. Ihler, "Variational gaussian-process factor analysis for modeling spatio-temporal data," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, Eds. Curran Associates, Inc., 2009, pp. 1177–1185.
- [27] C. Linkletter, D. Bingham, N. Hengartner, D. Higdon, and K. Ye, "Variable selection for gaussian process models in computer experiments," *Technometrics*, vol. 48, pp. 479–490, 2006.
- [28] D. Higdon, M. Kennedy, J. Cavendish, J. Cafoe, and R. Ryne, "Combining field data and computer simulations for calibration and prediction," *SIAM Journal on Scientific Computing*, vol. 26, no. 2, pp. 448–466, 2004. [Online]. Available: <http://epubs.siam.org/doi/abs/10.1137/S1064827503426693>
- [29] D. Higdon, J. Gattiker, B. Williams, and M. Rightley, "Computer model calibration using high-dimensional output," *Journal of the American Statistical Association*, vol. 103, no. 482, June 2008.
- [30] A. Pepelyshev, "The role of the nugget term in the gaussian process method," in *mODa 9 Advances in Model-Oriented Design and Analysis*, ser. Contributions to Statistics, A. Giovagnoli, A. C. Atkinson, B. Torsney, and C. May, Eds. Physica-Verlag HD, 2010, pp. 149–156.

- [31] S. Conti and A. O'Hagan, "Bayesian emulation of complex multi-output and dynamic computer models," *Journal of Statistical Planning and Inference*, vol. 140, pp. 640–651, 2010.
- [32] P. Ray and L. Carin, "Non-parametric bayesian modeling and fusion of spatio-temporal information sources," in *14th International Conference on Information Fusion*, Chicago, Illinois, USA, July 2011.
- [33] R. Adams and O. Stegle, "Gaussian process product models for nonparametric nonstationarity," in *International Conference on Machine Learning (ICML)*, 2008, pp. 1–8.
- [34] M. Hoffman and A. Gelman, "The no-u-turn sampler: Adaptive setting path lengths in hamiltonian monte carlo," *Journal of Machine Learning Research*, 2013.
- [35] M. A. Hague *et al.*, "Blowdown of pressure vessels. ii. experimental validation of computer model and case studies," *Trans. I. Chem. E.*, vol. 70 part B, no. 10, 1992.
- [36] A. Petruzzi, D. Cacuci, and F. D'Auria, "Best estimate model calibration and prediction through experimental data assimilation ii: Application to a blowdown benchmark experiment," *Nuclear Science and Engineering*, vol. 165, pp. 45–100, 2010.
- [37] D. Cacuci and M. Ionescu-Bujor, "Best-estimate model calibration and prediction through experimental data assimilation - i: Mathematical framework," *Nuclear Science and Engineering*, vol. 165, pp. 18–44, 2010.

- [38] D. C. C. Ionescu-Bujor and I. Navon, *Sensitivity and Uncertainty Analysis*. Chapman & Hall/CRC, 2005, vol. 2.
- [39] C. Rasmussen, "Gaussian processes to speed up hybrid monte carlo for expensive bayesian integrals," in *Bayesian Statistics 7. Proceedings of the Seventh Valencia International Meeting*, J. Bernardo, M. Bayarri, J. Berger, A. Dawid, D. Heckerman, A. Smith, and M. West, Eds. Oxford: Oxford University Press, 2003, pp. 651–659.
- [40] H. Planchon, R. Singer, D. Mohr, E. Feldman, L. Chang, and P. Betten, "The experimental breeder reactor ii inherent shutdown and heat removal tests - results and analysis," *Nuclear Engineering and Design*, vol. 91, pp. 287–296, 1986.
- [41] L. Koch, W. Loehenstein, and H. M. and, "Addendum to hazard summary report experimental breeder reactor-ii (ebr-ii)," Argonne National Laboratory, Tech. Rep., 1962.
- [42] G. Golden, H. Planchon, J. Sackett, and R. Singer, "Evolution of thermal-hydraulics testing in ebr-ii," *Nuclear Engineering and Design*, vol. 101, pp. 3–12, 1987.
- [43] K. Ha, H. Jeong, C. Cho, Y. Kwon, Y. Lee, and D. Hahn, "Simulation of the ebr-ii loss-of-flow tests using the mars code," *Nuclear Technology*, vol. 169, pp. 134–142, 2010.
- [44] H. Jeong, K. Ha, Y. Kwon, Y. Lee, D. Hahn, J. Cahalan, and F. Dunn, "Evaluation of the conduction shape factor with a cfd code for a liquid-metal heat transfer in heheat triangular rod bundles," *Nuclear Engineering and Design*, vol. 237, pp. 648–654, 2007.

- [45] W. Baumann, H. Domanus, D. Mohr, W. Sha, R. Schmitt, and J. Sullivan, "Ebr-ii in-vessel natural circulation analysis," Argonne National Laboratory, Tech. Rep. NUREG/CR-2821, 1982.
- [46] N. Todreas and M. Kazimi, *Nuclear Systems Volume I: Thermal Hydraulic Fundamentals, Second Edition*. CRC Press, 2011, vol. 1.
- [47] M. Memmott, J. Buongiorno, and P. Hejzlar, "On the use of relap5-3d as a subchannel analysis code," *Nuclear Engineering Design*, vol. 240, pp. 807–815, 2010.
- [48] C. Davis, "Evaluating the use of existing relap5-3d models to represent the actinide burner test reactor," Idaho National Laboratory, Tech. Rep. INL/EXT-07-12228, 2007.
- [49] A. Gopalakrishnan and J. Gillette, "Ebrflow-a computer program for predicting the coolant flow distribution in the experimental breeder reactor-ii," *Nuclear Technology*, vol. 17, pp. 205–216, March 1973.
- [50] J. Gillette, R. Singer, J. Tokar, and J. Sullivan, "Experimental study of the transition from forced to natural circulation in ebr-ii at low power and flow," *Journal of Heat Transfer*, vol. 102, pp. 525–530, 1980.
- [51] M. Lazaro-Gredilla, "Bayesian warped gaussian processes," in *Advances in Neural Information Processing Systems 16*. MIT Press, 2013.
- [52] M. Titsias, "Variational learning of induced variable in sparse gaussian processes," in *Proc. of 12th Int. Workshop on Artificial Intelligence and Statistics*, 2009, pp. 564–574.

- [53] C. R. R. Ghahramani, "Bayesian monte carlo," in *NIPS*, S. Becker, S. Thrun, and K. Obermayer, Eds., 2002, pp. 489–496.
- [54] T. A. Moselhy and Y. M. Marzouk, "Bayesian inference with optimal maps," *Journal of Computational Physics*, no. 23, p. 78157850", 2012.