

Full Volume Blood-Oxygen-Level-Dependent-Signal Parameter Estimation Using Particle Filters

Micah C. Chambers

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

Chris L. Wyatt, Chair
William T. Baumann
Aloysius. A. Beex
Daniel J. Stilwell

August 10, 2010
Blacksburg, Virginia

Keywords: BOLD Response, FMRI, Nonlinear Systems, Particle Filter, Bayesian Statistics
Copyright 2010, Micah C. Chambers

BOLD Parameter Estimation Using Particle Filters

Micah C. Chambers

(ABSTRACT)

Traditional methods of analyzing fMRI images use a linear combination of just a few static regressors. This work demonstrates an alternative approach using a physiologically inspired nonlinear model. By using a particle filter to optimize the model parameters, the computation time is kept low without requiring a linearization of the noise in the state variables. The activation results show regions similar to those found in SPM; however, there are some notable regions not detected by SPM. Though the parameters selected by the particle filter based approach presented here are more than sufficient to predict the BOLD response, more model constraints are needed to uniquely identify a single set of parameters. This ill-posed nature explains the large discrepancies found in other research that attempt to characterize the model parameters. Despite this, the reliance on a single point estimator of parameters is unnecessary, given the output of the particle filter is a full posterior probability at the final time step. For this reason the final distribution of parameters may still be medically relevant. For this same reason, additional model constraints may be performed in post-processing by a simple application of conditional probability. Concluding, this work presents not just a viable alternative to the traditional method of detecting activation, but an extensible technique of estimating the joint BOLD parameter distribution.

Contents

1	Introduction	1
1.1	Historic Context	2
1.2	Overview	2
1.3	FMRI	3
1.4	BOLD Physiology	4
1.5	Post Stimulus Undershoot	5
1.6	Properties of the BOLD Model	6
2	Methods of Analyzing FMRI	8
2.1	Statistical Parametric Mapping	8
2.1.1	Classical Activation Detection	8
2.1.2	Random Field Theory	9
2.1.3	General Linear Model	9
2.1.4	Hierarchical Linear Models	10
2.1.5	Discussion	11
2.2	Solving the BOLD Model	11
2.2.1	Linear Approximation	12
2.2.2	Nonlinear Least Squares	12
2.2.3	Unscented Kalman Filter	14
2.2.4	Hybrid Methods	19
2.3	Conclusion	19

3 Particle Filters	20
3.1 Introduction	20
3.2 Model	20
3.3 Derivation	21
3.3.1 Weighting	22
3.4 Calculating Weights	23
3.4.1 Basic Particle Filter Algorithm	24
3.5 Resampling	24
3.6 Weighting Function	28
3.7 Simple, Nonlinear Example	28
4 Methods	32
4.1 Prior Distribution	32
4.2 Model	35
4.3 Resampling	36
4.4 Choosing $P(y_k x_k)$	37
4.5 BOLD Noise	38
4.5.1 Detrending	39
4.6 Preprocessing	43
4.7 Particle Filter Settings	44
4.8 FMRI Configuration	45
5 Simulation	46
5.1 Single Time Series	46
5.1.1 Signal with Low Noise	47
5.1.2 High Noise	53
5.1.3 Pure-Noise, low magnitude	61
5.1.4 Pure Noise, High Magnitude	66
5.1.5 Single Voxel Review	67

5.2	Multi-voxel Simulation	69
6	Real Data	76
6.1	Results	76
6.2	Discussion	86
7	Conclusion	90

List of Figures

2.1	Canonical Hemodynamic Response Function	10
2.2	Example updates of a distribution using Kalman Filter, [1]	15
2.3	Distributions of state variables after simulating for .1s	17
2.4	Distributions of state variables after simulating for 1s	18
3.1	Particle Filter progression, note that the initial support is flat; the particles are equally spaced between -10 and 10	29
3.2	An Example Half Wave Rectifier Circuit, where G is the transformer gain, v_t is the activation voltage of the diode, $u(t)$ is the input at time t , C is the capacitance, R is the load resistance and v_y is the output voltage	30
3.3	Example Input/Output of the Half Wave Rectifier	30
4.1	Response to .1s impulses with the mean parameters from [2]	33
4.2	Response to .1s impulses with the mean parameters from [3]	34
4.3	Q-Q Plots of normalized resting state data	40
4.4	Q-Q Plots of resting state data, using the BOLD signal changes	41
4.5	Q-Q Plots of resting state data, after the de-trending	42
5.1	Test Signals with low noise compared to the clean signal.	47
5.2	A comparison of the preprocessed signals for the low noise case. This is the noisy input to the actual particle filter algorithm.	47
5.3	A comparison of the fitted signals for the low noise case.	48
5.4	Test Signals with high noise compared to the clean signal, $\sigma_x = .01, \sigma_y = .005$. .	53
5.5	A comparison of the preprocessed signals for the high noise case.	53

5.6	A comparison of the fitted signals for the high noise case.	54
5.7	Two particular preprocessed noise realizations for the high noise case.	55
5.8	The results for the noise realizations shown in Figure 5.7.	56
5.9	Converging histogram for parameters during run 1, as in Figure 5.7.	59
5.10	Converging histogram for parameters during run 2, as in Figure 5.7.	60
5.11	Time-series lacking any real signal. With, $\sigma_x = .01, \sigma_y = .005$	61
5.12	A comparison of the preprocessed signals for the signal-free case. ($\sigma_y = .01, \sigma_x = .005$)	62
5.13	Fits to the non-active, low noise signal. Note that the line is thick because all the fits overlap. This is all 11 fitted lines.	63
5.14	Fit from a single particle filter run, with the noise input.	63
5.15	Converging histogram for parameters when the signal consists purely of low level noise. Same run as Figure 5.14	65
5.16	Fits to the non-active, high noise signal. Note that the line is thick because all the fits overlap. This is all 11 fitted lines.	66
5.17	Fit from a single particle filter run, with the noise input.	67
5.18	A single time-series fit in the simulated slice.	70
5.19	Comparison of activation with greymatter, parameter regions.	71
5.20	Histogram of estimated parameters in section 1 with \sqrt{MSR} less than .85.	73
5.21	Histogram of estimated parameters in section 2 with \sqrt{MSR} less than .85.	74
5.22	Histogram of estimated parameters in section 3 with \sqrt{MSR} less than .85.	75
6.1	Sagittal, coronal and axial slices of SPM results (6.1(b) 6.1(c) 6.1(b)), as well as a series of axial slices, 6.1(a). Units of activation are in Student's T-scores. Higher indicates higher assurance that the signal cannot have occurred through noise alone.	78
6.2	Sagittal, coronal and axial (6.2(b) 6.2(c) 6.2(b)), as well as a series of axial slices, 6.2(a). Units of match is normalized residual. The lowest (best) levels were .7. The highest error shown is 1.	79
6.3	Sagittal, coronal and axial (6.2(b) 6.2(c) 6.2(b)), as well as a series of axial slices, 6.2(a). Units of match is bits (standard for base-2 Mutual Information). The highest (best) levels are .42. The worst shown is .1.	80
6.4	Section 1, Estimated vs. Actual BOLD response	81

6.5	Section 2, Estimated vs. Actual BOLD response	82
6.6	Section 3, Estimated vs. Actual BOLD response	84
6.7	Section 4, Estimated vs. Actual BOLD response	85
6.8	Section 5, Below threshold in both particle filter checks, but above threshold in SPM. Mutual Information of 0.0212822, T-Value of 4.17399 and <i>MSE</i> of 1.14171.	87
6.9	Section 6, MI of 0.335504, T Value: 2.49154, normalized error: 0.783348 Not visible in SPM	88
6.10	Section 7, .1052 in MI, 1.31534 in SPM (which left this flat) and 1.09992 normalized \sqrt{MSE} . Only visible in MI map.	89

List of Tables

1.1	Parameters found by various studies. (NC) indicates that the value wasn't calculated. [4] made use of the values from [5] where not explicitly stated	7
2.1	Parameters used to test Gaussianity of variables after being transitioned through the BOLD model	16
4.1	Prior distributions used in the particle filter.	35
5.1	Estimated Parameters on 11 different runs with low noise. First row is the true parameters, last is mean over the 11 runs. The \sqrt{MSR} is residual and \sqrt{MSE} is the error.	52
5.2	Typical Covariance matrix of the parameters at the end of a run.	52
5.3	Estimated Parameters on 10 different runs with high noise. First row is the true parameters. Note also that the red row is Run 1 and the blue row is Run 2, as named in Figure 5.7 and Figure 5.8	57
5.4	Estimated Parameters on 11 different runs with low noise and no signal present. . .	62
5.5	Mutual Information and the normalized \sqrt{MSE} , for signal/noise configurations. .	68
5.6	Actual parameters for each regions in the simulated slice.	72

Chapter 1

Introduction

Traditional methods of analyzing timeseries images produced by Function Magnetic Resonance Imaging (fMRI) use regression techniques with only linear scaling parameters. Though adding more degrees of freedom naturally mandates more computation, in this thesis I will discuss a Sequential Monte Carlo method of fitting a nonlinear model with seven degrees of freedom at a computation cost that would still allow real time calculations for multiple voxels. More practically, the method described here forms a completely separate method of detecting neural activity in comparison to the traditionally used Statistical Parametric Mapping (SPM). Though more computationally intense, this method is capable of modeling nonlinear effects, is conceptually simpler and provides more detailed output. Additionally, by using a separate particle filter for each voxel's time series it is possible to estimate parameters and make real-time predictions for small neural regions, a feature which could be useful towards real time fMRI ([6]). Future works will also benefit from the ability to apply conditions to the posterior distribution in post-processing without having to re-run the algorithm; for instance in light of additional physiological constraints. Modeling the BOLD response as a nonlinear system is the best way to determine the correlation of stimulus sequence with the BOLD response; yet in the past doing this on a large scale has been far too computationally taxing. The solution used here takes approximately 40 seconds for a single voxel's time series (5 minutes in length, with a quad core machine).

This thesis is organized as follows. In the introduction I will introduce fMRI, the method by which neural time changing data is detected. This section will also describe the basic form of the BOLD model - which drives the detectable changes in MR signal. chapter 2 will discuss other methods of analyzing fMRI images as well as other techniques that have been, or could be applied to the nonlinear regression model described here. chapter 3 derives the particle filter using Bayesian statistics and discusses some practical elements of implementing the particle filter algorithm. chapter 4 then goes into further detail about the specific particle filter configuration used in this work. This section also describes the pre-processing used before the particle filter is applied. The results are described separately for simulated data and real fMRI data in chapter 5 and chapter 6, respectively. Finally in chapter 7 there is a discussion of the usefulness and implications

of this technique as well as recommendations for direction of future works.

1.1 Historic Context

For the past twenty years, Functional Magnetic Resonance Imaging (fMRI) has been at the forefront of cognitive research. Despite its limited temporal resolution, fMRI is the standard tool for localizing neural activation. Whereas other methods of analyzing neural signals can be invasive or difficult to acquire, fMRI is quick and cheap, and its analysis straight forward. By modeling the governing equations behind the neural response that drives fMRI, it is possible to increase the power of fMRI. The underlying state equations hold important information about how individual brain regions react to stimuli. The model parameters on the other hand, hold important information about the patient's individual physiology including existing and future pathologies. In short, the long chain of events driving fMRI signals contain information beyond correlation with stimuli.

In the past fifteen years, a steady stream of studies have built on the original Blood Oxygen Level Dependent (BOLD) signal derivation first described by [7]. The seminal work by [8] attempted to explain the time evolution of the BOLD signal using a windkessel model to describe the local changes in Deoxygenated Hemoglobin content. Incremental improvements were made to this model until [2] brought all the changes together into a single complete set of equations. And while there have been numerous adaptations in the model, many of them summarized in [9], even the basic versions have less bias error than the empirically driven *Canonical Hemodynamic Model* [9], [10]. On the other hand BOLD signal models have numbers of parameters ranging from seven [11] to 50 [12] for a signal as short as 100 samples long. This number of parameters presents a significant risk of being under-determined and having high computation cost. In this work, only the simplest physiologically inspired model will be used (with 7 parameters), and steps will be taken to make the most of computation time.

1.2 Overview

Detecting neural activity using the changes in fMRI images is based on the so called Blood Oxygen Level Dependent (BOLD) signal. The BOLD signal is caused by minute changes in the ratio of Deoxygenated Hemoglobin to Oxygenated Hemoglobin in blood vessels throughout the brain. Because Deoxygenated Hemoglobin (DHb) is paramagnetic, higher concentrations attenuate the signal detected during T2-weighted Magnetic Resonance Imaging (MRI) techniques. The most common fMRI imaging technique, due to its rapid repetition time (TR), is Echo Planar Imaging (EPI). When axons become active, a large amount of ions quickly flow out of the cell. In order for this action potential to occur again (and thus for the neuron to fire again), an active pumping process must move ions back into the axon. This process of recharging the axon requires extra energy, which temporarily increases the metabolic rate of oxygen. On a massive scale (cubic mil-

limeter) this activation/recharge process happens continuously. However, when a particular region of the brain is significantly active, the action potentials occur more often, resulting in a local increase of the Cerebral Metabolic Rate of Oxygen (CMRO₂). Thus, blood vessels in an active area will tend to have less oxygenated hemoglobin (due to the increased rate at which oxygen is being consumed), and more deoxygenated hemoglobin, resulting in an attenuated FMRI signal. In compensation for activation, muscles that control blood vessels relax in that region to allow more blood flow, which actually overcompensates. This ultimately results in lower than average concentration of deoxyhemoglobin. Thus, the BOLD signal consists of a short initial dip in the MR signal, followed by a prolonged increase in signal that slowly settles out. It is this overcompensation that is the primary signal detected with FMRI imaging. This cascade of events is believed to consist of increased the local metabolism, blood flow, blood volume, and oxygenated hemoglobin. The differences in onsets of these effects is what causes the overcompensation that is observable in FMRI. Unfortunately, FMRI has no inherent unit of measurement, and thus signal levels are all relative: within a particular person, scanner and run.

1.3 FMRI

Magnetic Resonance Imaging, MRI, is a method of building 3D images non-invasively, based on the difference between nuclear spin relaxation times in different molecules. First, the subject is brought into a large magnetic field which causes nuclear spins to align. Radio Frequency (RF) signals may then be used to excite nuclear spin away from the base alignment. As the nuclei precess back to the alignment of the magnetic field, they emit detectable RF signals. Conveniently, the excitation of nuclear spins return their original state at different rates, called the T1 relaxation time, depending on the atoms excited. Additionally, the coherence of the spins also decay differently (and roughly an order of magnitude faster than T1 relaxation) based on the properties of the region. This gives two primary methods of contrasting substances, which form the basis of T1 and T2 weighted images. Additionally, dephasing occurs at two different rates, the T2 relaxation time, which is unrecoverable, and T2* relaxation, which is much faster, but possible to recover from via special RF signals. T1 relaxation times are typically on the order of seconds if a sufficiently strong excitation was applied. In order to rapidly acquire entire brain images, as done in Functional MRI, a single large excitation pulse is applied to the entire brain, and the entire volume is acquired in a single T1 relaxation period. Because the entire k-space (spatial-frequency) volume is acquired from a single excitation, the signal-to-noise-ratio is low in this type of imaging (Echo Planar Imaging).

Increasing the spatial resolution of EPI imaging necessarily requires more time or faster magnetic field switching. Increasing magnet switching rates though is difficult, because it can result in more artifacts, or even lower signal to noise ratios. The result is that at *best* FMRI is capable of 1 second temporal resolution. The signal is further diluted because each voxel contains the signal from a large number of neurons, capillaries and veins. Thus, the FMRI signal, which is sensitive to the chemical composition of materials, is the average signal from various types of tissue in addition

to the blood. As mentioned in [section 1.2](#), and explored in depth in [section 1.4](#), the usefulness of FMRI comes from the discerning of changes in Deoxyhemoglobin/Oxyhemoglobin. Therefore, it is necessary to assume that in the short term the only chemical changes will be in capillary beds feeding neurons. In practice this may not be the case, for instance near significant veins, and it may explain some of the noise seen in FMRI imaging (see [section 4.5](#)). Because MRI lacks units and certain areas will have a higher base MR signal, all FMRI studies deal with percent change from the base signal; rather than raw values. This also removes most of the structural data which is not helpful in determining neural activity.

1.4 BOLD Physiology

It is well known that the two types of hemoglobin act as a contrast agents in EPI imaging [8], [13], [7], however the connection between Deoxyhemoglobin/Oxygenated Hemoglobin and neural activity is non-trivial. Intuitively, increased metabolism will increase Deoxyhemoglobin, however blood vessels are quick to compensate by increasing local blood flow. Increased inflow, accomplished by loosening capillary beds, precedes increased outflow, driving increased blood storage capacity. Since the local MR signal depends on the ratio of Deoxyhemoglobin to Oxygenated Hemoglobin, increased volume of blood can effect this ratio if metabolism doesn't exactly match the increased inflow of oxygenated blood. This was the impetus for the ground breaking balloon model ([8]) and windkessel model ([14]). These works derive from first principals the changes in deoxyhemoglobin ratio and volume of capillaries based on a given flow. These were the first two attempts to quantitatively account for the shape of the BOLD signal as a consequence of the lag between the cerebral blood volume (CBV) and the inward cerebral blood flow (CBF). In fact [8] went so far as to show that a simple, well chosen blood flow waveform coupled with a square wave cerebral metabolic rate of oxygen (CMRO₂) curve, in the context of a balloon model, could fully account for the BOLD signal.

Although [8] demonstrated that a well chosen flow waveform could explain most features of the BOLD signal, there was still a matter of proposing a realistic waveform for the CBF and for the CMRO₂. [2] gave a reasonable and simple expression for CBF input, f , based on a flow inducing signal, s , in combination with the original balloon model where v is normalized cerebral blood volume (CBV), q is the normalized local deoxyhemoglobin/oxygenated hemoglobin ratio.

$$\dot{s} = \epsilon u(t) - \frac{s}{\tau_s} - \frac{f - 1}{\tau_f} \quad (1.1)$$

$$\dot{f} = s \quad (1.2)$$

$$\dot{v} = \frac{1}{\tau_0}(f - v^\alpha) \quad (1.3)$$

$$\dot{q} = \frac{1}{\tau_0}\left(\frac{f(1 - (1 - E_0)^f)}{E_0} - \frac{q}{v^{1-1/\alpha}}\right) \quad (1.4)$$

where ϵ is a neuronal efficiency term, $u(t)$ is a stimulus, and τ_f , τ_s are both time constants, E_0 is

the resting metabolic rate and α is Grubb's parameter controlling the balloon model.

This completed the basic balloon model, and was well summarized again in [15]. [16] refined the readout equation of the BOLD signal based on the deoxyhemoglobin content (q) and local blood volume (v), resulting in the final BOLD equation:

$$y = V_0((k_1 + k_2)(1 - q) - (k_2 + k_3)(1 - v)) \quad (1.5)$$

$$k_1 = 4.3 \times \nu_0 \times E_0 \times TE = 2.8 \quad (1.6)$$

$$K_2 = \epsilon_0 \times r_0 \times E_0 \times TE = .57 \quad (1.7)$$

$$k_3 = \epsilon_0 - 1 = .43 \quad (1.8)$$

Where $\nu_0 = 40.3s^{-1}$ is the frequency offset in Hz for fully de-oxygenated blood (at 1.5T), $r_0 = 25s^{-1}$ is the slope relating change in relaxation rate with change in blood oxygenation, and $\epsilon_0 = 1.43$ is the ratio of signal MR from intravascular to extravascular at rest. Although, these constants change with experiment (TE, ν_0, r_0), patient, and brain region (E_0, r_0), often the estimated values taken from [16] are taken as the constants $a_1 = k_1 + k_2 = 3.4$, and $a_2 = k_2 + k_3 = 1$ in studies using 1.5 Tesla scanners. While this model is more accurate than the static Hemodynamic Model used in SPM, it is not perfect.

1.5 Post Stimulus Undershoot

Although the most widely used, the BOLD model described in [Equation 1.4](#) and [Equation 1.8](#) have been extensively added on to. The most significant feature missing from the original model is the post-stimulus undershoot. The post-stimulus undershoot is the term used for a prolonged subnormal BOLD response for a period of 10 to 60 seconds after stimulus has ceased ([17], [18]).

Because [Equation 1.4](#) is not capable of producing such a prolonged undershoot, additional factors must exist. Two theories exist for the post stimulus undershoot. Recall that a lower than base signal means that there is an increased deoxyhemoglobin content in the voxel. The first and simplest explanation is that the post-stimulus undershoot is caused by a prolonged increase in CMRO2 after CBV and CBF have returned to their base levels. This theory is justified by studies that show CBV and CBF returning to the baseline before the BOLD signal ([19], [20], [21], [22], [23]). Unfortunately, because of limitations on fMRI and in vivo CBV/CBF measurement techniques it is difficult to isolate whether CBF and CBV truly have returned to their baseline. Other studies indicate that there can be a prolonged supernormal CBV ([18], [12], [24]), although none of these papers completely rule out the possibility of increased CMRO2. The discrepancies may in part be explained by a spatial dependence in the post-stimulus undershoot; described by [25]. [17] makes a compelling case that most of the post stimulus undershoot can be explained by combination of a prolonged CBV increase, and a prolonged CBF undershoot, and that the previous measurements showing a quick recovery of CBV were in fact showing a return to baseline by arterial CBV.

Regardless of the probability that CMRO2 and CBF are detached, research into the post-stimulus undershoot has led to the creation of much more in depth models. In [26] additional state variables

model oxygen transport, whereas [21] models CMR02 from a higher level, and somewhat more simply; though it still adds 9 new parameters. [12] introduces nonlinearities into the CBF equations as a method to explain the post-stimulus undershoot, which falls in line with a prolonged increase in CBF observed in [17]. Similarly [27] adds additional compartments to model the BOLD signal that result from venous and arterial blood. [9] compared these models and though it did not deal extensively with the post-stimulus undershoot, it did show incremental improvements in quality from additional parameters (over the basic Balloon model); though at the cost of greatly increased complexity. Importantly,[9] did show that by simply adding viscoelastic terms from [21], a slowed return to baseline is possible to model, without greatly increasing complexity. Regardless, because these models are more complex, and the parameters are not well characterized, in this work the simple Balloon model is used. Simplicity is even more important because it is the first time a particle filter has been used to calculate the BOLD parameters. Of course, future works could certainly benefit from the more advanced models, especially with the addition of viscoelastic effects.

In summary, there have been extensive refinements to the Balloon model, however, the increased complexity and lack of known priors make these models less desirable. Therefore, in this work where computation time is especially important, only the most basic model is used.

1.6 Properties of the BOLD Model

Since the first complete BOLD model was proposed by [5], several studies have analyzed its properties. The most important property is that the system is dissipative, and given enough time will converge to a constant value. This is found simply by analyzing the eigenvalues of the Jacobian of the state equations, ([9], [28]). The steady state of the Balloon model equations gives:

$$\begin{aligned}
 s_{ss} &= 0 \\
 f_{ss} &= \tau_f \epsilon u + 1 \\
 v_{ss} &= (\tau_f \epsilon u + 1)^\alpha \\
 q_{ss} &= \frac{(\tau_f \epsilon u + 1)^\alpha}{E_0} (1 - (1 - E_0)^{1/(\tau_f \epsilon u + 1)}) \\
 y_{ss} &= V_0((k_1 + k_2)(1 - q_{ss}) - (k_2 + k_3)(1 - v_{ss})) \tag{1.9}
 \end{aligned}$$

In real FMRI data, there is a significant nonlinearity in response; with short sequences responding disproportionately strong ([29], [30], [9]). This nonlinearity is accounted for in the Balloon model, although [9] shows that if there will be large variance in the length of signals, modeling Neural Habituation may be necessary to fully capture the range of responses. Stimuli that last longer than 4 seconds tend to be more linear, which is why block designs are so well accounted for by the General Linear Model ([29], [9]). For this paper we will use only a simple version of the Balloon model, described by [Equation 1.4](#), to keep the solution tractable, and because it is the most well studied.

Parameter	[2]	[3]	[4]	[9]
τ_0	$N(.98, .25^2)$	8.38 ± 1.5	.94	.27
α	$N(.33, .45^2)$	$.189 \pm .004$.4 (NC)	.63
E_0	$N(.34, .1^2)$	$.635 \pm .072$.6 (NC)	.33
V_0	.03 (NC)	$.0149 \pm .006$	(NC)	.16
τ_s	$N(1.54, .25^2)$	4.98 ± 1.07	2.2	2.04
τ_f	$N(2.46, .25^2)$	8.31 ± 1.51	.45	5.26
ϵ	$N(.54, .1^2)$	$.069 \pm .014$	(NC)	.89

Table 1.1: Parameters found by various studies. (NC) indicates that the value wasn't calculated. [4] made use of the values from [5] where not explicitly stated

Another interesting result of [9] was the sensitivity analysis. There it was found that the parameters are far from perpendicular, implying that exact inference of parameters may not be possible without constraint. This could explain the extreme discrepancies in [Table 1.1](#).

A summary of results from previous studies of the Balloon Parameters are shown in [Table 1.1](#).

Chapter 2

Methods of Analyzing FMRI

The ultimate purpose of this work is to provide a new tool for analyzing FMRI data.

Currently, FMRI is used to determine the location of responses to stimuli. The method of finding activation is discussed in [section 2.1](#). The goal of this work is to move away from the question "Is this region active" and instead ask "What does the activation look like". Answering this question necessitates more complex models and certainly will result in longer run times. Already there have been several other attempts to model the BOLD response; these works will be discussed in this chapter.

2.1 Statistical Parametric Mapping

Although not strictly the same as parameter calculation from FMRI, activation detection is similar and worth discussing. Estimation of parameters is a generalization of the idea of activation detection. Given the popularity of Statistical Parametric Mapping (SPM) it is important to draw a distinction between the methods proposed in this work.

2.1.1 Classical Activation Detection

The most basic method of analyzing FMRI data is through a standard T-test between resting state and active state samples. Simply put, the mean is calculated separately for non-stimulus and stimulus time intervals. A classic t-test may then be applied, giving the probability that the distributions actually have the same mean. Because of the correlated noise present in FMRI ([section 4.5](#)), it is necessary to apply some sort of high-pass filter to the data. Without applying such a filter, P values must be set extraordinarily high to prevent false positives [31]. If there truly is signal due to stimuli, the distributions will not actually be independent Gaussians because activation does not fit a square wave ([section 1.4](#)). For this reason other methods are often more used, as discussed in

[subsection 2.1.3.](#)

2.1.2 Random Field Theory

SPM methods make significant use of T-Tests across large regions; however, such T-tests work slightly differently than a single individual test. A t-test with a p-value of .05 over a modestly sized FMRI image, say 10,000 voxels, will on average generate 500 false positives. This is called the multiple comparison problem. Traditional Bonferroni Correction deals with this by requiring each test to pass with P value of $\frac{.05}{10000}$. The probability of a single false positive would then .05. Unfortunately this leads to unrealistically low p-values; so low that it would be impossible for any biological system to satisfy. To compensate, a Gaussian kernel is applied to smooth the image. This has the benefit of reducing the noise variance and decreasing the effective number of independent measurements. Because the number of independent measurements is smaller, Bonferroni correction can theoretically be applied with a lower scaling factor than the original voxel count [32]. A side effect of this, a single voxel activation is virtually impossible to detect.

2.1.3 General Linear Model

The most common FMRI analysis technique is SPM, though there are more advanced versions that the simple square wave method discussed in [subsection 2.1.1](#). Hierarchical Models are one important improvement that allows researchers to combine data across multiple runs, patients and stimuli (see [33] for more on Hierarchical Modeling). Hierarchical Models concatenate all the data into a single dataset, then perform a linear fit between a design matrix and the data. The design matrix encapsulates all known experimental factors such as stimuli, young/old, etc. The equation for a general linear model is:

$$Y(t) = X(t)\beta + \epsilon(t) \quad (2.1)$$

where $Y(t)$ is the smoothed or de-trended time course of measurements, $X(t)$ is the design matrix, β is a column vector of weights, and ϵ is the error. Thus for every time, the measurement is assumed to be a weighted sum of the columns of X plus some error. The calculation of β is then performed using a maximum likelihood or gradient descent search to minimize the error.

As mentioned previously, a square wave stimulus does not result in a square wave in the activation of brain regions. The BOLD signal is in fact a smoothed version of the stimuli. As such, when fitting an FMRI time course to the input, the input (X 's columns) is usually smoothed to reduce bias error. The best method, that maintains a linear fit, is convolving the input with a Hemodynamic Response Function (HRF). The *Hemodynamic Response Function* mimics the basic shape of BOLD activation, including a delay due to rise time and fall time. The fitting process is then a least squares fit over the space of the vector β . Therefore $Y(t)$ is estimated as a linear combination

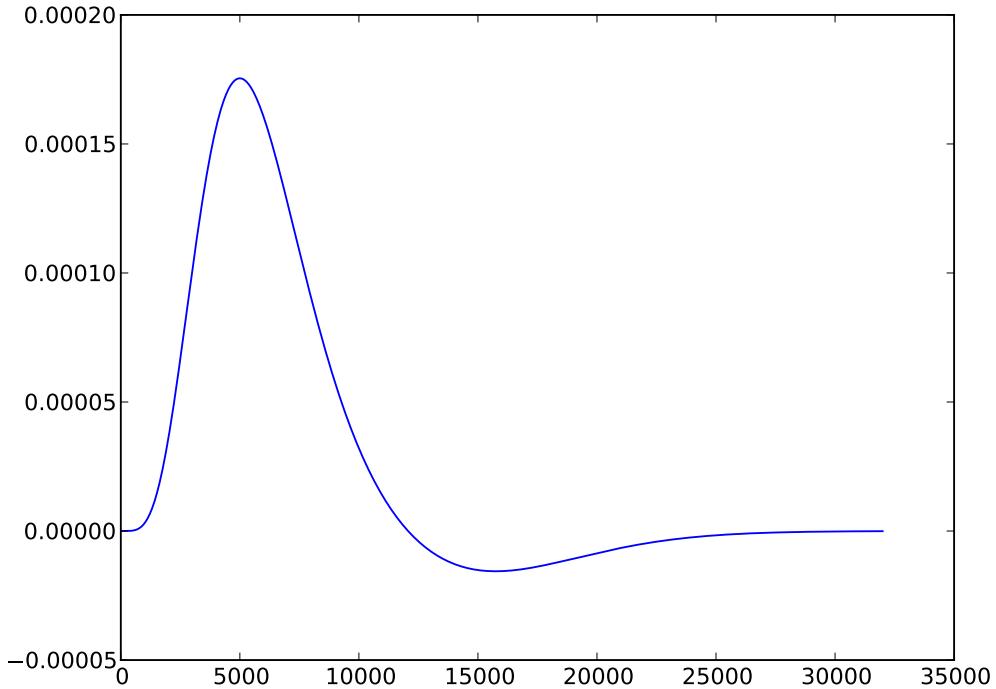


Figure 2.1: Canonical Hemodynamic Response Function

of the columns of X .

Smoothing the input with a single HRF poses certain problems. It is well known that different Hemodynamic Response Functions are necessary for different regions of the brain. The Canonical HRF that is most used, has been optimized for the visual cortex. As section 1.5 discussed, there are certainly variations in the shape of the BOLD response, both between brain regions and patients. [10] discusses the implications of choosing an incorrect HRF, the most important of which is a definite increase in false negatives. While an atlas of Hemodynamic Response Functions for each region could definitely mitigate this risk, it does not deal with variation between patients. Thus, the inability to fit parameters other than scale definitely hinders analysis of activation. As a result, there is significant bias error when using a linear fit to localize activation. Notably, studies will also have a bias toward the visual cortex, precisely because it is so well studied.

2.1.4 Hierarchical Linear Models

As mentioned previously and discussed extensively in [5] and [33], hierarchical models may be applied to account for systematic differences between subjects. For instance, if the study happens

to be a mix between young and old, incorporating that into the model is wise, regardless if that is the purpose of the test. The reason to do this is to account for additional variance that may not be present in similar studies. The Hierarchical form used by [5] is shown in [Equation 2.2](#).

$$\begin{aligned}
 Y(t) &= X_1(t)\theta_1 + \epsilon_1(t) \\
 \theta_1(t) &= X_2(t)\theta_2 + \epsilon_2(t) \\
 &\dots \\
 \theta_{n-1}(t) &= X_n(t)\theta_n + \epsilon_n(t)
 \end{aligned} \tag{2.2}$$

The Empirical Bayes algorithm is used in both in both [5] and [33]. As a consequence, point estimators are used for each θ , rather than the full distributions.

2.1.5 Discussion

In all, the GLM is useful for determining linear dependence of a set of regressors on the output. Unfortunately, as discussed in [section 1.6](#) there are significant nonlinearities that almost certainly cause false negatives in the Statistical Parametric Maps. Unfortunately nonlinear analyses have only recently become feasible, so the scale of the problem is still unknown. The problem is highlighted by the common scenario where no significant activation can be found in a single run [11] [3].

The static nature of the linear model also limits its inference power. Besides not permitting HRF differences between patients, there is no reasonable way to incorporate other forms of physiological data. Combined FMRI CBF or CBV imaging methods are rapidly getting better, as seen in [17]. These techniques could shed light on neural activation by providing extra measurements, yet a physiologically reasonable model is necessary for this. In reverse, activation detection methods also don't have the ability to identify pathologies based on state variables or parameters. For example decreased compliance of blood vessels could indicate, or even cause, a neurological condition that is not easily seen in other imaging modalities. Thus, the benefits of physiologically meaningful models are manifold.

2.2 Solving the BOLD Model

Unlike Statistical Parametric Mapping, the techniques described in this section are all attempts to learn some version of the BOLD model. Although [8] and [2] both proposed physiologically reasonable values for the model parameters, [34] was the first paper to calculate the parameters based on actual FMRI data. However, in that case, the voxels were chosen from regions that were detected as active by the GLM. It is therefore possible that the parameters are biased toward parameters that fit the linear model.

2.2.1 Linear Approximation

In [34], a novel combination of linear and nonlinear modeling was used to generate parameter estimates. Because it is impossible to calculate the partial derivative of the output with respect to parameters, $\frac{\partial \theta}{\theta}$, [34] approximates the partial. At each step of the Expectation-Maximization algorithm, the differential equation is integrated, calculating the residuals. Then, for each θ_i surrounding the current estimate of θ , a Volterra-Kernel expansion of the output y is generated. Generation of the Volterra Kernel is quick, and, since it is linear there is an analytical solution. Thus, once the Volterra Kernel exists, the value of y at that measurement point, for that θ_i can easily be found. Thus, one part of the partial is calculated at one point using $\frac{y(t) - y_i(t)}{\theta - \theta_i}$. This is repeated for every dimension of θ to give the full $\frac{\partial y}{\theta}(t)$. This is only at time t though, and it must be repeated for every measurement. Finally the full derivate matrix is filled out, and the next step in the E-M algorithm can proceed. The full E-M algorithm for estimating the states is notation-heavy and can be found in [34].

Although this is certainly an interesting method of performing non-linear regression, there are a few caveats. First, the partials are numerical approximations, based on approximate values (using Volterra-Kernels) of y . Importantly, the Volterra-Expansion of y is not able to model interactions between state variables; interactions that were found to cause interesting behavior in this work. In subsection 2.2.3, for the purpose of demonstrating nonlinearities in the signal, it was necessary to propagate state variables through 1 second of simulation. For the distribution of state variables mentioned in Table 2.1 it was necessary to use step sizes smaller than .001 to prevent unpredictable behavior. This is using the average parameters, thus the only complication came from the interplay between states, which can certainly lead to interesting behavior. Additionally, all the tests performed in [34] were on regions found to be active by the General Linear Model. For this reason, the reliability of the approximation is unknown for regions that are active but sufficiently nonlinear to avoid detection by conventional tests.

2.2.2 Nonlinear Least Squares

Rather than localizing activation, by using the physiologically plausible BOLD model, it is possible to determine the values of governing parameters.

Although there are certainly benefits to using a derived model, rather than a purely empirical model, there are serious implications. The first problem is that all the powerful classical techniques of gradient descent are off limits; since the model is a true nonlinear dynamical system with no closed form solution. The implication of this is that the calculation of a Jacobian for residuals won't work; and thus powerful techniques such as the Gauss-Newton method, which are helpful in most nonlinear problems, are off limits. Additionally, a gradient descent is difficult to perform without the ability to form partials of the output with respect to all the parameters.

Although anything requiring a Jacobian is out, there are other heuristic techniques that could po-

tentially illuminate the BOLD response. Simulated Annealing (SA) is a common method of optimizing high dimensional regression problems. The idea is to pick a random start, and then at each iteration pick a random nearby point, and if that point is below some energy constraint (energy is a function of the residual), called the temperature, the algorithm moves to that point and continues with the next iteration. The temperature is slowly lowered until no nearby points below the temperature can be found (or the temperature drops below the current point). There are variations of this, for instance it is common to require every movement to be in the downward direction (in terms of energy). Like most nonlinear optimization problems, there is no guarantee of an optimal solution, although the longer the algorithm is allowed to run, the better the solution will get. Since every step requires an entirely new run of the BOLD model, it can be extremely time consuming, which is why we are not using it here.

Algorithm 2.1 Simulated Annealing Algorithm

```

Initialize  $\Theta$ , or if there exists a decent estimate start there
Initialize temperature,  $T$  to value above initial energy
while  $E(\Theta) < T$  do
    repeat
        Pick  $\theta$  near  $\Theta$ 
        Calculate energy,  $E$ , of  $\theta$ 
    until  $E > T$ 
    Move to new estimate: set  $\Theta = \theta$ 
end while

```

Another potential method of interest is the use of Genetic Algorithms (GA). Genetic algorithms are similar to Simulated Annealing, in that they randomly move to better solutions based on a cost function. However; in genetic algorithms a single point estimate isn't used. Instead a population of estimates is generated, each with distinct parameters, and then each set of parameters is rated with a fitness function. Parameter sets that are good get a higher weight; then new parameter sets are generated by randomly combining pieces of the old parameter sets. The pieces are typically chosen at a rate proportional to the fitness of the donor; thus fit parameter sets tend to pass on their properties. In addition to this, random mutations may be introduced that come from no existing parent. The new generation is then rated with the fitness function again, and the entire process starts over. The stop condition for a genetic algorithm is typically based on some threshold for fitness or a maximum number of generations. As with all nonlinear searches, unless the function is known to be convex, there is no guarantee that a global minimum has been reached.

Although both these methods can be highly effective, they have the downside of requiring very high computation time. In this case of the BOLD model, each time the energy or fitness needs to be calculated, a large number of cycles must be spent re-simulating the BOLD model for the set of parameters. As I'll discuss in [chapter 3](#), the Particle Filter method is able to circumvent this re-calculation to some degree.

Algorithm 2.2 Genetic Algorithm

```

Initialize  $N$  estimates,  $E = \{\Theta_0, \Theta_1, \dots \Theta_N\}$ 
for  $G$  generations do
    Calculate fitness for each  $\Theta$ , Ex. for residual  $R$ ,  $1/R$  or,  $e^{-R}$ 
    for  $i$  in  $N$  do
        Randomly select two parents (with higher probability for more fit  $\Theta$ 's)
        Randomly merge parts of the two parents to form a new  $\Theta_i$ 
        At some low probability change one or two parameters in  $\Theta_i$ 
    end for
end for

```

2.2.3 Unscented Kalman Filter

The Unscented Kalman Filter (UKF) is a powerful Gaussian/Bayes filter that attempts to model the posterior distribution of dynamical systems as a multivariate Gaussian. The Unscented Kalman Filter (UKF) generalizes the Extended Kalman Filter by allowing the state update to be a function, g ,

$$X(t) = g(u(t), X(t-1)) \quad (2.3)$$

$$Y(t) = h(X(t)) \quad (2.4)$$

In order to estimate the posterior at t , a deterministic set of sigma points (often 2 per dimension, plus 1 at the mode of the multivariate distribution) weighted according to a Gaussian estimate of $X(t-1)$ are passed through the update equation. This set of points are then used to estimate the mean and covariance of $X(t)$. The benefit of this, is that it requires no Jacobian and only a few extra calculations to get a decent estimate of a posterior Gaussian. In the BOLD case, the set of equations we are modeling have no closed form solution, and finding the Jacobian is impossible without approximations. Although [11], [28] mention a Jacobian of the BOLD response, this is not strictly the case and is rather $\frac{\partial J}{\partial t}$ rather than a true Jacobian. This is important because the Extended Kalman filter depends on the Jacobian to map a Gaussian through the advancement of time. Thus the popular Extended Kalman Filter won't work in this case, whereas the Unscented Kalman Filter still does. In fact [28] uses the UKF to perform a similar type of analysis to the one performed in this work.

The difficulty of using a Kalman Filter, however, is that it assumes a multivariate Gaussian for the state variables, $X(t-1)$. The more nonlinear the system gets the more likely that the Gaussian will be insufficient to describe the distribution, $X(t)$. When this occurs, every step from $X(t+1)$ to $X(t)$ will introduce additional error in the posterior distribution. Furthermore, it is not really known what sort of underlying distributions may exist in such a mixed biological, mechanical, chemical system such as the brain. Assuming the parameters all to be Gaussian may in fact be a gross error. On the other hand, for small variances and short time steps the Gaussian distribution

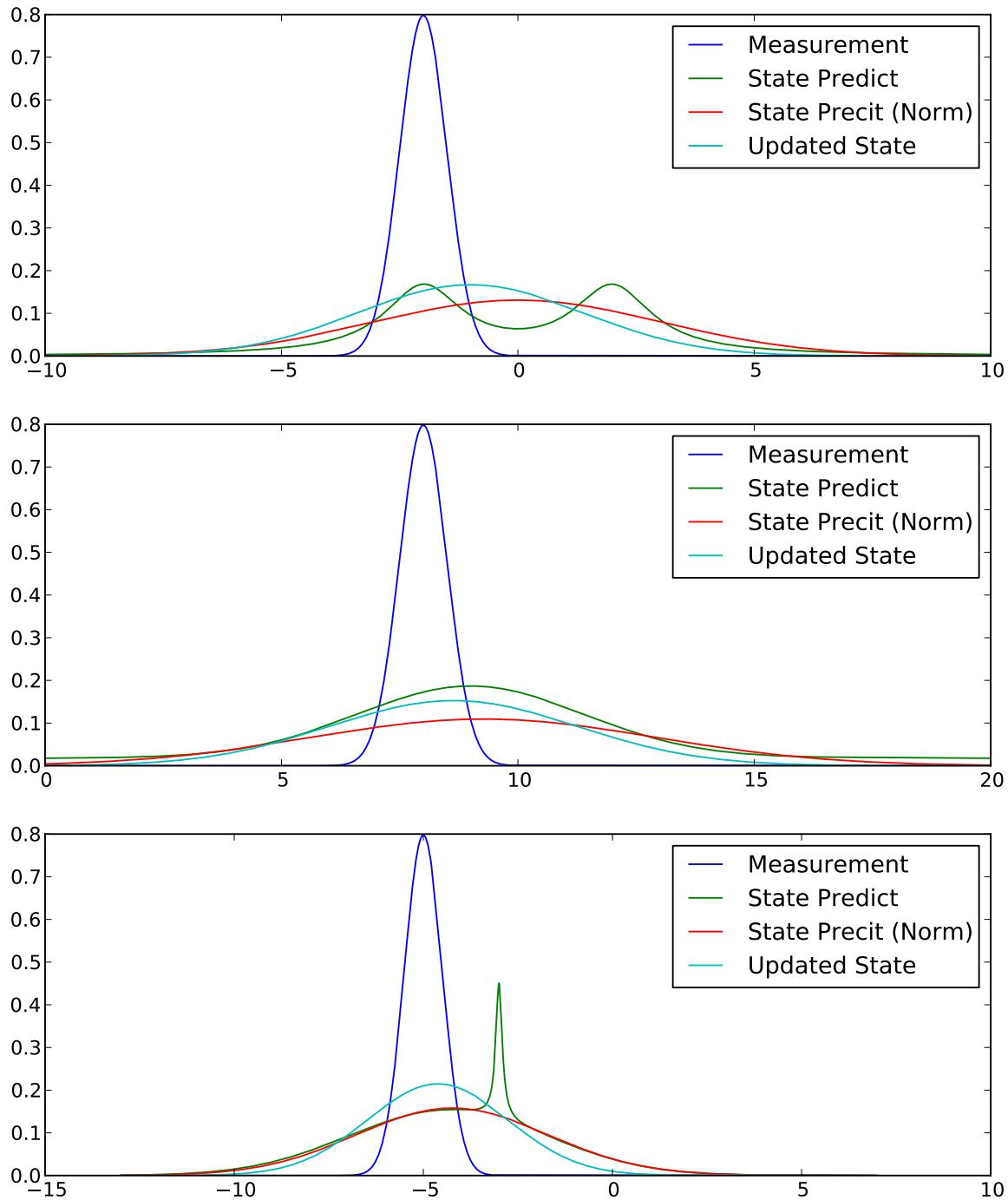


Figure 2.2: Example updates of a distribution using Kalman Filter, [1]

Parameter	Run 1
τ_0	.98
α	.33
E_0	.34
V_0	.03
τ_s	1.54
τ_f	2.46
ϵ	.54
V_t	N(1, .09)
Q_t	N(1, .09)
S_t	N(1, .09)
F_t	N(1, .09)

Table 2.1: Parameters used to test Gaussianity of variables after being transitioned through the BOLD model

is a good fit, and so in some limited cases the Unscented Kalman Filter could work well. These are non-trivial issues given that the assumption of Gaussianity is what allows the UKF to estimate the posterior using only the first and second moments; two parameters that don't uniquely describe most distributions.

To determine the amount of error incurred in a Gaussian estimate during a typical sample period, the states of BOLD equations were assigned according to four dimensional Gaussian. The states were then propagated through two seconds of simulation (a typical TR in fMRI) and then the resulting marginal distributions were compared with a Gaussian distribution. The purpose is to determine the degree to which the results of simulation will result in non-Gaussian output, given a Gaussian input. This also demonstrates the degree of nonlinearity present in the system. The parameters used are shown in [Table 2.1](#)

Notably s_t has intentionally been set to a non-equilibrium, but physiologically plausible value. The value of u is left at zero the entire time, so the system will decay naturally (see [section 1.4](#)), though initializing s at a non-zero level will drive the system for several seconds. [Figure 2.3](#) shows the results when the system is left on for 100 milliseconds after setting the variables according to [Table 2.1](#). The Q-Q plots fit well with a Gaussian, demonstrating that at this short time interval nonlinearities have not yet begun to effect the distribution. However, [Figure 2.4](#) us the result after 1 second, which is faster than most fMRI scanners are capable of sampling at. At that range the tails of the distributions for v and q are starting to deviate from the Gaussian distribution. As a result the uncertainty in y is deviating from the Gaussian distribution as well. This is important, because although approximating the distribution with a Gaussian based on the first two moments will work in the short run, there will be residual error in the distribution.

On the other hand, this effect is more limited if the initial variance is somewhat smaller. In tests with those cases, it took much longer for the nonlinearities to skew the distribution. That result could be encouraging to those looking to use the UKF, if the distributions are kept relatively thin.

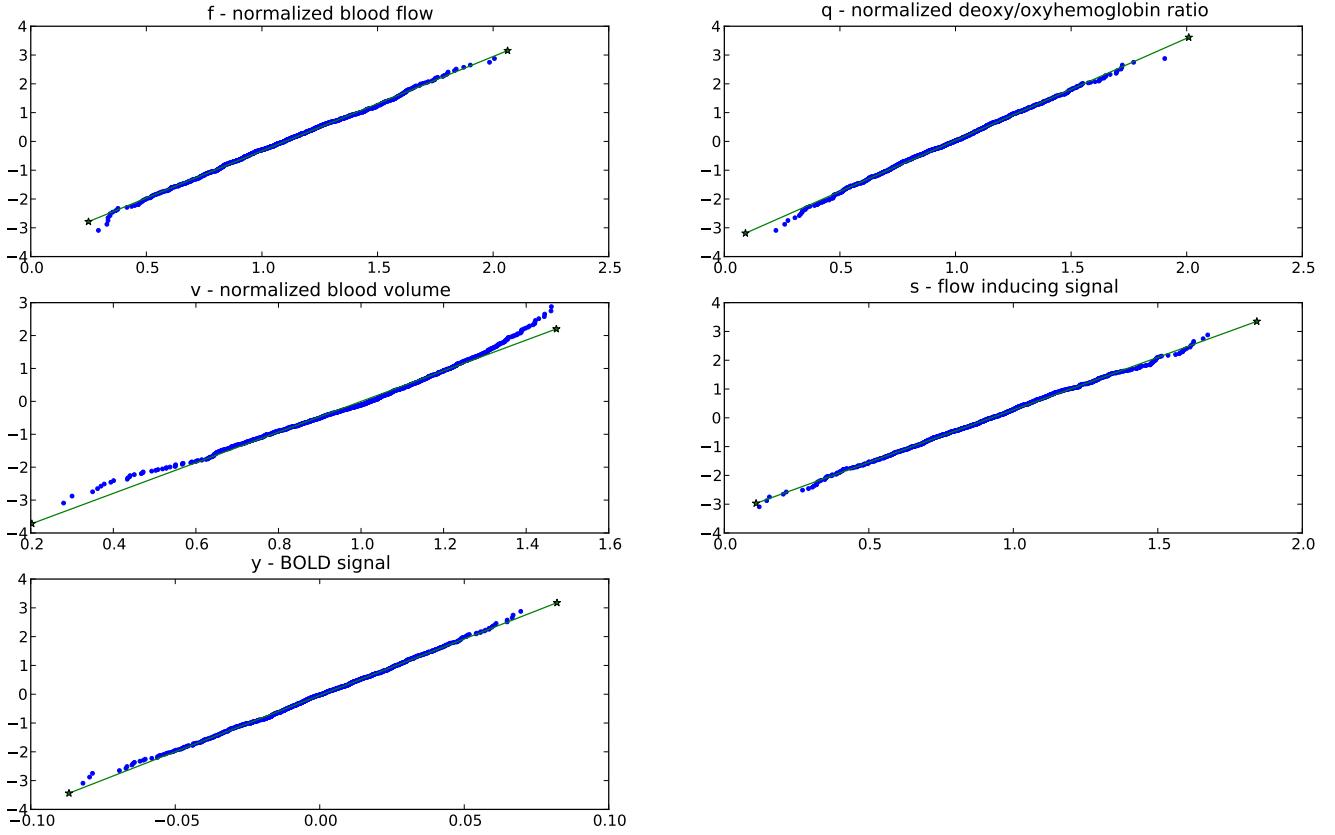


Figure 2.3: Distributions of state variables after simulating for .1s

Another potential problem with the UKF is that typically the sigma points, used to estimate the posterior probability, $P(X(t)|X(t-1), u(t))$, are located on the main axes. As a result, the covariances are not allowed to be effected by the state transition the same way the variances are. While this may be reasonable in low-dimensional systems, high dimensional systems have a much greater potential for interplay between the variables. While this problem is simple to fix (by using more sigma points off the main axes), there is a definite cost in complexity.

Ultimately, there is a distinct possibility that the nonlinearities of the BOLD model make Gaussian estimates unrealistic and thus less effective. More advanced tests where static variables such as α are varied as well could shed even more light on the issue. The trouble with using the UKF then to estimate parameters is that all eleven members of X would be treated like a single joint Gaussian distribution which certainly compound the issues of nonlinearity seen in [Figure 2.3](#) and [Figure 2.4](#)

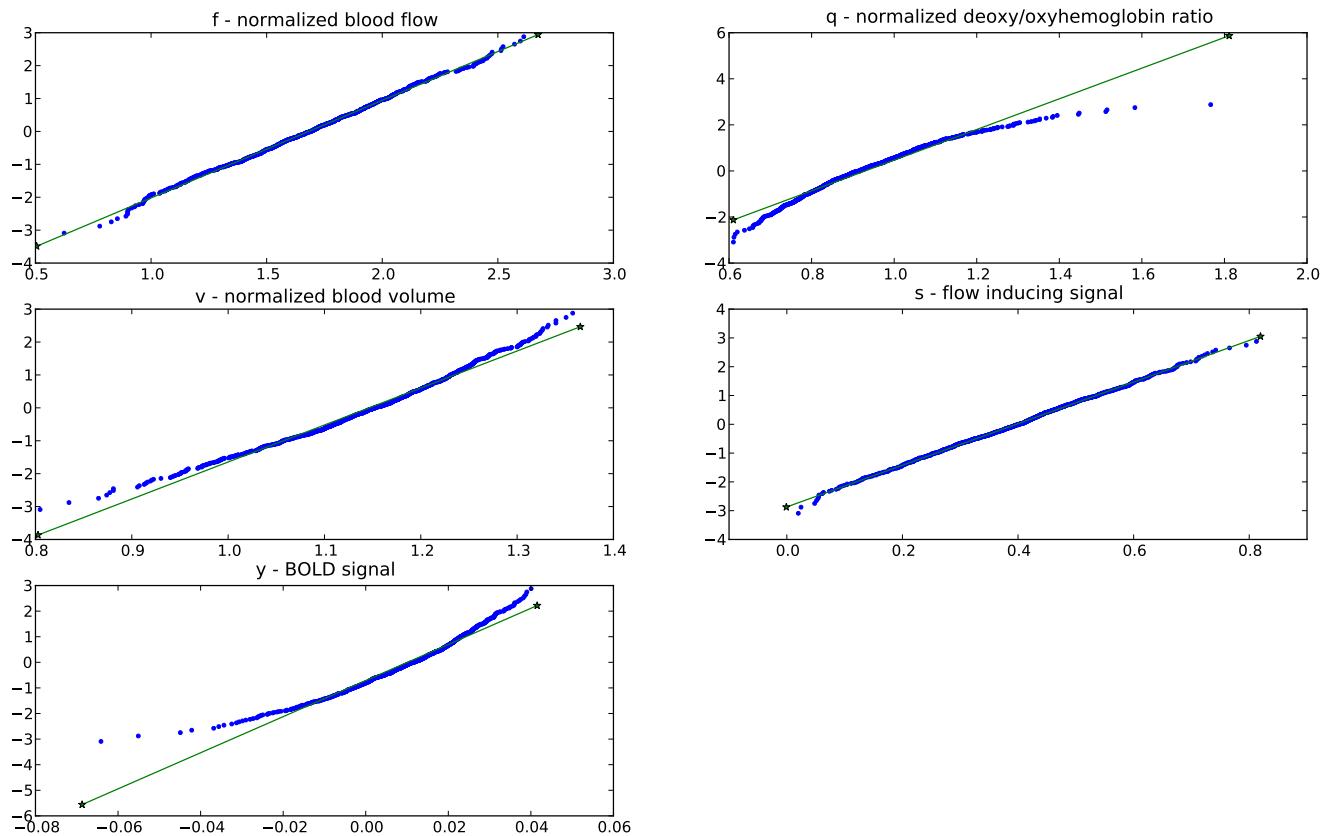


Figure 2.4: Distributions of state variables after simulating for 1s

2.2.4 Hybrid Methods

In [11], a maximum likelihood method for innovation processes was used, as described by [35]. [35] uses a similar construction to a Kalman filter, to break the time series into a series of innovations, for which Maximum Likelihood was performed. While this is probably the most likely solution to give the correct output, there are a few problems. First, every step in parameter space requires a recalculation of all the state variables. With two or three parameters this is fine, more than that, and calculations can become intractable. Additionally, there is no way to ensure a global minimum is reached. Although this is also true of the methods in subsection 2.2.2, those methods contained random elements designed to overcome this issue.

In [3], a hybrid particle filter/gradient descent algorithm was used to simultaneously derive the static and dynamic parameters, (classically known as parameters and state variables, respectively). A particle filter is used to calculate the state variables at each time; then the estimated distribution of the particles is used to find the most likely set of parameters that would give that distribution of state variables. This process is repeated until the parameters converge. Interestingly [3] comes to a very different set of parameter estimates as compared to the original [2] estimates (Table 1.1). In fact the results are significantly different from virtually every other study. The most obvious discrepancy is the larger time constants, τ_f , τ_s and τ_0 . While of course this could be poor convergence of the algorithm, there is another other possibility. Unlike all the other methods mentioned, excepting the methods in subsection 2.2.2, the algorithm described in [3] does not depend on prior distributions. It is possible then that the bias toward the prior in other methods screwed the results. While [3] is certainly in the minority; further exhaustive studies of the parameters, using unbiased techniques may be called for. A further comparison between the distributions found in [3] and [2] will be discussed in section 4.1, including graphs: section 4.1, Figure 4.1.

2.3 Conclusion

There is no ideal solution to solving this system of nonlinear equations. Exhaustive search type methods such as those employed by [3] and [4] have long run times even for a single voxel. While Volterra models are an interesting solution, there have not yet been exhaustive tests to determine whether such approximations work well throughout state space. The most promising method of those reviewed here is the Kalman filter based method. It is able to maintain a fast runtime while still approaching the solution. While the reliance on a Gaussian estimate to the true posterior distribution could cause problems, some modifications could make it powerful. The particle filter method proposed in the next section bears a strong resemblance to the unscented Kalman filter; albeit with more point estimating the posterior.

Chapter 3

Particle Filters

3.1 Introduction

Particle filters, a type of Sequential Monte Carlo (SMC) methods, are a powerful way of estimating the posterior probability distribution a set of parameters given a timeseries of measurements. Unlike Markov Chain Monte Carlo (MCMC) estimation, particle filters are designed for time-varying random variables. The idea behind particle filters is similar to Kalman Filters; however, unlike Kalman Filters, distributions are stored as an empirical distribution rather than the first two moments of a Gaussian. Thus particle filters are preferred when the model is nonlinear, and thus non-gaussian. This section is largely based on [36] and [1].

3.2 Model

The idea of the particle filter is to build an empirical distribution out of a large number of parameter sets, called particles. Each particle contains all the parameters and states needed to propagate the model forward. The particle filter begins with a wide distribution (called the Prior Distribution) of possible particles and then, as measurements come in, weights particles based on the quality of their output estimates. Thus parameter sets that tend to give good estimations of the measurements get weighted higher than parameter sets that give poor estimates. Although the reliance on a prior distribution can be troublesome, when the system being modeled has physical meaning, establishing reasonable ranges for parameters may be quite easy. Optimizing the prior can be more difficult though, unless the system has been extensively studied.

Suppose set or stream of measurements at discrete times are given, $\{Y_k, k = 1, 2, 3, \dots, K\}$, where K is infinite for a stream. Because k is a discrete time, let t_k define the continuous time of k . Suppose also that there is a hidden set of state variables, $X(t)$ that dictates the movement of $Y(t)$, although for most of the time dealings will be with $X_k = X(t_k)$. The goal of the particle filter

is to estimate the *distribution* of the true parameters Θ that dictates the movement of $X(t)$. The model also permits random motion in $X(t)$, so the particle filter also estimates the distribution of $X(t)$. The only difference between the members of parameter vector Θ and those of $X(t)$ is that the members of Θ have no known update equation. Members of both vectors are permitted to have some noise, although this may not be explicitly stated in the model. The generic, continuous, nonlinear system definition is shown in [Equation 3.1](#).

$$\begin{aligned}\dot{X}(t) &= f(t, X(t), u(t), \theta, \nu_x) \\ Y(t) &= g(t, X(t), u(t), \theta, \nu_y)\end{aligned}\tag{3.1}$$

$X(t)$ is vector of state variables, Θ is a vector of system constants, $u(t)$ is an input, $Y(t)$ the observation, and ν_x and ν_y are random variates. Although any of these variables could be a vector, for the sake of simplicity only Θ and $X(t)$ will be considered as such.

Although not necessary for particle filters in general, a few simplifying assumptions are made for this work. First, the systems are assumed to be time invariant. This assumption is based on the idea that if you paused the system for Δt seconds, when unfrozen the system would continue as if nothing happened. Few biological systems are predictable enough for them to be summarized by a time varying function, least of all the brain. While heart beats are certainly periodic and have an effect on the BOLD signal, the period varies too much for the system to be considered varying with time. Next, it assumes that input cannot directly influence the output, which in the case of the BOLD signal is a good assumption. Also, noise is considered to be additive. Finally, because the only difference between the members of $X(t)$ and Θ is an update function, from now on x will contain Θ . The assumptions now allow for a simplified version of the state space equations:

$$\dot{X}_k = f(X_{k-1}, u_k) + \nu_x\tag{3.2}$$

$$Y_k = g(X_k) + \nu_y\tag{3.3}$$

3.3 Derivation

The goal of the particle filter is to evolve an empirical distribution $P(x_k|u_{0:k}, Y_{0:k})$, that asymptotically approaches the true probability distribution $P(X_k|u_{0:k})$. Note that capital X will be used as the actual realizations of the state variable, whereas x will denote estimates of X . Additionally, the notation $a : b$ indicates the set $[a, b]$, as in $u_{a:b}$, which would indicate all the inputs from time a to time b . Considering the noise present in X , $P(X_k|u_{0:k})$ is not a single true value but probability distribution.

To begin with, the particle filter must be given a prior distribution, from which the initial N_p particles are drawn. A particle contains a weight as well as an estimate of X_k , which as already

stated, contains every variable needed to run the model. Then the prior is generated from a given distribution, $\alpha(X)$, by:

$$\{[x_0^i, w^i] : x_0^i \sim \alpha(X), w^i = \frac{1}{N_p}, i \in \{1, 2, \dots, N_p\}\} \quad (3.4)$$

Where N_p is the number of particles or points used to describe the prior using a Mixture PDF. Note that any exponents will be explicitly labeled as such, to avoid confusion with the particle numbering scheme.

Therefore, after the particle have been generated they should approximate $\alpha(X)$:

$$\alpha(X) \approx P(x_0) = \sum_{i=0}^{N_p} w^i \delta(X - x_0^i) dx \quad (3.5)$$

Where $\delta(x - x_0)$ is 1 if and only if $x = x_0$ (the Kronecker delta function).

If a flat prior is preferred, then each particle's weight could be scaled to the reciprocal of the density at the particle:

$$w^i = \frac{1}{\alpha(x_0^i)} \quad (3.6)$$

Whether or not to flatten the prior is a design decision. The reason this might be preferred over a direct uniform distribution is that the distribution width will inherently scale for increased particle counts although some distributions flatten out better than others. Either way, $\alpha(X)$ *must* be wide enough to incorporate any posterior that arises. If the prior is not sufficiently dense, the particle filter can compensate, if it is not sufficiently wide the particle filter won't converge.

3.3.1 Weighting

For all the following areas, the probabilities implicitly depend on $u_{0:k}$, so those terms are left off for simplicity.

Whenever a measurement becomes available it permits refinement of the posterior density. This process of incorporating new data is called sequential importance sampling, and eventually allows convergence. The weight is defined as

$$w_k^i \propto \frac{P(x_{0:k}^i | y_{0:k})}{q(x_{0:k}^i | y_{0:k})} \quad (3.7)$$

where q is called an *importance density*. The importance density is the density of the points, thus by dividing by this value, the weight should not depend on the location of the estimation points, but rather only on $P(x_{0:k}^i | y_{0:k})$, the probability of that particle being correct given all the measurements up to time k . Of course if there is a far off peak in the posterior that q does not have support points

in, there will be quantization errors, and that part of the density can't be modeled. This is why it is absolutely necessary that q fully covers $P(x_{0:k}^i | y_{0:k})$.

It is helpful to consider how the importance density affects the initial distribution. In the initial distribution, the weights are all the same; and for the sake of argument, let them all be scaled up to 1. Then

$$w_k^i q(x_{0:k}^i | y_{0:k}) = q(x_{0:k}^i | y_{0:k}) = P(x_{0:k}^i | y_{0:k}) \quad (3.8)$$

the estimated probability, $P(x_{0:k}^i | y_{0:k})$ depends only on the way the particles are distributed. As new measurements are incorporated, the weight will accumulate probabilities through time, which will be discussed next.

3.4 Calculating Weights

To calculate the weight of a particular particle, it is necessary to calculate both $q(x_{0:k}^i | y_{0:k})$ and $P(x_{0:k}^i | y_{0:k})$. Note that $q(x_{0:k}^i | y_{0:k})$ may be simplified by assuming that y_k doesn't contain any information about x_{k-1} . Technically this could be false; since later measurements may shed light on currently hidden changes in x . For practical applications though it is helpful assumption.

$$q(x_{0:k}^i | y_{0:k}) = q(x_{0:k}^i | y_{0:k-1}) \quad (3.9)$$

The choice of the importance density is another design decision; however it is common to use the integrated state equations. Although other importance density functions exist; for the particle filter used here, the standard importance density will be used: the modeled prior.

$$q(x_k | x_{k-1}, y_{0:k}) = P(x_k | x_{k-1}) \quad (3.10)$$

The benefit of this choice for importance density is that an approximation for $P(x_k | x_{k-1})$ is freely available: its simply the set of particles propagated forward in time using the state equations. Additionally it makes updating weights simple, as seen in [Equation 3.14](#).

The $q(x_{0:k}^i | y_{0:k})$ may then be simplified:

$$\begin{aligned} q(x_{0:k}^i | y_{0:k}) &= q(x_k | x_{0:k-1}, y_{0:k}) q(x_{0:k-1} | y_{0:k}) \\ &= q(x_k | x_{0:k-1}, y_{0:k}) q(x_{0:k-1} | y_{0:k-1}) \quad [\text{Equation 3.9}] \\ &= q(x_k | x_{k-1}, y_{0:k}) q(x_{0:k-1} | y_{0:k-1}) \quad [\text{Markov Property}] \\ &= P(x_k | x_{k-1}) q(x_{0:k-1} | y_{0:k-1}) \quad [\text{Equation 3.10}] \end{aligned} \quad (3.11)$$

Calculating $P(x_{0:k} | y_{0:k})$ is a bit more involved. First, using the assumption that the distribution of y_k is fully constrained by x_k , and that x_k is similarly fully constrained by x_{k-1} , we are able to make the good assumptions that:

$$\begin{aligned} P(y_k | x_{0:k}, y_{0:k-1}) &= P(y_k | x_k) \\ P(x_k | x_{0:k}, y_{0:k-1}) &= P(x_k | x_{k-1}) \end{aligned} \quad (3.12)$$

These are of course just re-statements of the state equations assumed by [Equation 3.2](#) and [Equation 3.3](#).

Additionally, for the particle filter y_k and $y_{0:k-1}$ are constant across all particles, thus $P(y_k|y_{0:k-1})$ can be dropped when the equality is changed to a proportion. Using these properties, $P(x_{0:k}^i|y_{0:k})$ may be broken up as follows (primarily using Bayes' Theorem):

$$\begin{aligned}
 P(x_{0:k}|y_{0:k}) &= \frac{P(y_{0:k}, x_{0:k})}{P(y_{0:k})} \\
 &= \frac{P(y_k, x_{0:k}|y_{0:k-1})P(y_{0:k-1})}{P(y_k|y_{0:k-1})P(y_{0:k-1})} \\
 &= \frac{P(y_k|x_{0:k}, y_{0:k-1})P(x_{0:k}|y_{0:k-1})}{P(y_k|y_{0:k-1})} \\
 &= \frac{P(y_k|x_{0:k}, y_{0:k-1})P(x_k|x_{0:k-1}, y_{0:k-1})P(x_{0:k-1}|y_{0:k-1})}{P(y_k|y_{0:k-1})} \\
 &= \frac{P(y_k|x_k)P(x_k|x_{k-1})P(x_{0:k-1}|y_{0:k-1})}{P(y_k|y_{0:k-1})} \\
 &\propto P(y_k|x_k)P(x_k|x_{k-1})P(x_{0:k-1}|y_{0:k-1}) \quad [P(y_k|y_{0:k-1}) \text{ is constant}]
 \end{aligned} \tag{3.13}$$

Plugging [Equation 3.10](#) and the result of [Equation 3.13](#) into [Equation 3.7](#) leads to:

$$\begin{aligned}
 w_k^i &\propto \frac{P(y_k|x_k^i)P(x_k^i|x_{k-1}^i)P(x_{0:k-1}^i|y_{0:k-1})}{P(x_k^i|x_{k-1}^i)q(x_{0:k-1}^i|y_{0:k-1})} \\
 &\propto w_{k-1}^i P(y_k|x_k)
 \end{aligned} \tag{3.14}$$

Thus, by making the following simple assumptions, evolving a posterior density requires no knowledge of noise distribution.

1. $f(t, x(t), u(t)) = f(x(t), u(t))$ and $g(t, x(t), u(t)) = g(x(t))$
2. The PDF $q(x_i(0))$ (the prior) fully covers $P(x_i(0))$
3. Markov Property: $P(x_k|x_{0:k-1}) = Pr(x_k|x_{k-1})$
4. $q(x_{0:k-1}|y_{0:k}) = q(x_{0:k-1}|y_{0:k-1})$

3.4.1 Basic Particle Filter Algorithm

From the definition of w_i , the algorithm to calculate an approximation of $P(X(t_k)|Y_{0:k})$ or $P(X(t_k + \delta t)|Y_{0:k})$ is simple.

3.5 Resampling

As a consequence of the wide prior distribution (required for a proper discretization of a continuous distribution), there will be a significant proportion of particles with insignificant weights. While

Algorithm 3.1 Sequential Importance Sampling

Initialize Particles:

for i : each of N_p particles **do**

$$x_0^i \sim \alpha(X)$$

$$w_0^i = \frac{1}{N_p}$$

end for

for k : each measurement **do**

for i : each particle **do**

$$x_k^i = x_{k-1}^i + \int_{t_{k-1}}^t f(x(\tau), u(\tau)) d\tau$$

$$w_k^i = w_{k-1}^i P(y_k | x_k)$$

end for

end for

$$P(x(t_k + \Delta t)) \approx \sum_{i=0}^{N_p} w_k^i \delta \left(x - \left(x_k^i + \int_{t_k}^{t_k + \Delta t} f(x(\tau), u(\tau)) d\tau \right) \right)$$

this does help describe the tails of the distribution, it means a lot of computation will be based. Instead, it would be preferable if most of the computation is spent on the most probable regions. Ideally the computation time spent on tails would be proportional to the actual size of the tails. In this case particle locations would match the true posterior and all weights would be equal. The case where a large number of the weights have become irrelevantly small is called *particle degeneracy*. In [37] an ideal calculation of the effective number of particles is found based on the particles' true weight. However, given that only an approximation for the true weight exists, they also provide a simple heuristic calculation of N_{eff} .

$$N_{eff} \approx \frac{\sum_{i=0}^{N_p} w_i}{\sum_{i=0}^{N_p} w_i^2} \quad (3.15)$$

Any quick run of a particle filter will reveal that unless the prior is particularly accurate, N_{eff} drops precipitously. To alleviate this problem a common technique known as resampling may be applied. The idea of resampling is to draw from the approximate posterior, thus generating a replica of the posterior with a better support. Therefore, a new set of particles may be drawn from the empirical distribution as follows:

$$\hat{x}_j \sim \left(\sum_{i=0}^{N_p} w_k^i \delta(x - x_k^i) \right) \quad (3.16)$$

For infinite particles this new distribution will match the old. Unfortunately, this isn't the truth in practice: since the support is still limited to the original particles, the number of *unique* particles can only go down. This effect, dubbed *particle impoverishment* can result in excessive quantization errors in the final distribution. However, there is a solution. Instead of sampling from the discrete distribution, a smoothing kernel is applied, and particles are drawn from that distribution. Because it is continuous, particle impoverishment cannot occur. The easiest way to sample from the continuous distribution is to break the re-sampling down into two steps. After calculating an

Algorithm 3.2 Resampling Algorithm

```

Calculate total weight,  $W = \sum_{i=0}^{N_p} w^i$ 
for all  $0 < i < N_p$  do
    Draw  $V$  from uniform range  $[0, W]$ 
     $C = W_t$ 
    for all  $0 < j < N_p$  and  $C < V$  do
         $C = C - w^j$ 
    end for
    Add  $[x^j, \frac{1}{N_p}]$  to the new distribution
end for

```

estimate of the scale of the original distribution, algorithm 3.2 is performed. Next, a distribution is generated based on the variance of the original distributions. Finally, for each particle in the discretely re-sampled distribution, a sample is drawn from the smoothing distribution and added to the particle. The process regularization is defined as:

$$x_i = x_i + h\sigma\epsilon \quad (3.17)$$

Where h is an optional bandwidth, σ is the standard deviation such that $\sigma\sigma^T = cov(x)$ and ϵ is drawn from the chosen kernel. [38] goes into significant depth and proves the optimality of the Epanechnikov Kernel for reducing MSE between the original and resampled distributions. However, [38] also espouses the usefulness of the Gaussian Kernel, due to the ease drawing samples from it, which for this work was more important.

Algorithm 3.3 Regularized Resampling Algorithm

```

Calculate Covariance,  $C$ , of empirical distribution,  $\hat{x}$ 
Find  $D$  such that  $DD^T = C$ 
Resample  $\hat{x}$  using algorithm 3.2
for  $0 < i < N_p$  do
    Draw  $\epsilon$  from the standard normal, same dimensionality as  $X$ 
     $x^i = x^i + hD\epsilon$ 
end for

```

It has been proposed by [39] that if the underlying distribution is non-Gaussian, then using the original bandwidth will over-smooth. In reality, over smoothing will only become an issue if resampling is performed over and over. Thus if resampling is performed at every step then this could certainly caused problems. If the distribution is over-smoothed then the algorithm may not converge as rapidly; however, because the bandwidth is still based on particle variance, which decays as particles are ruled out, it is still able to converge. In fact, over-smoothing is preferable to under smoothing, since over-smoothing simply slows convergence while under-smoothing could result

in false negatives. At the same time, as the number of dimensions in the state vector increases, the standard deviation based approximation becomes less effective ([36]). Because of the high dimensionality of the BOLD model, and limited measurements, it is helpful to have a broader bandwidth to explore the distribution.

Algorithm 3.4 Regularized Particle Filter

Initialize Particles:

for i : each of N_p particles **do**

$$x_0^i \sim \alpha(X)$$

$$w_0^i = \frac{1}{N_p}$$

end for

for k : each measurement **do**

for i : each particle **do**

$$x_k^i = x_{k-1}^i + \int_{t-1}^t f(x(\tau), u(\tau)) d\tau$$

$$w_k^i = w_{k-1}^i P(y_k | x_k)$$

end for

Calculate N_{eff} with [Equation 3.15](#)

if $N_{eff} < N_R$ (recommend $N_R = \min(50, .1N_p)$) **then**

Resample using algorithm [3.3](#)

end if

end for

$$\text{At } t + \Delta t, t \in T, P(x(t + \Delta t)) \approx \sum_{i=1}^{N_p} w_i(t) \delta \left(x - (x_i(t) + \int_t^{t+\Delta t} f(x(\tau), u(\tau)) d\tau) \right)$$

Nevertheless, because of the potentially wide smoothing factor applied by regularized resampling, performing this step at every measurement would allow particles a great deal of mobility. This mobility is the opposite of convergence, which is why regularized resampling should only be done when N_{eff} drops (say less than 50). Other than the periodic regularized resampling then, the regularized particle filter is nearly identical to the basic sampling importance sampling filter (SIS).

Thus, with regularized resampling, it is possible to prevent both particle degeneracy as well as particle impoverishment. An additional bonus is that as the particle filter converges, the density of particles in the area of the solution goes up. This has a similar effect to simulated annealing where, as the algorithm approaches the end, the random steps get smaller and smaller. There is a potential risk when resampling. If for some reason the solution is not covered by the new support the algorithm will fail to converge properly.

The ultimate effect of this regularized resampling is a convergence similar to simulated annealing or a genetic algorithm. Versions of x that are fit (give good measurements) spawn more children nearby which allow for more accurate estimation near points of high likelihood. As the variance of the estimated x 's decrease, the radius in which children are spawned also decreases. Eventually the radius will approach the width of the underlying uncertainty, ν_x and ν_y .

3.6 Weighting Function

Because the distribution of ν_y in [Equation 3.3](#) is unknown, it is necessary to choose a distribution for this. This distribution is important because $\nu_y \sim P(y_k|x(T))$, which is used for updating weights. Ideally this weighting function would exactly match the measurement error in the output. It is logical to assume this error is centered at zero and has a scale comparable to the signal levels. While a Gaussian function is the traditional choice, there are other reasonable distributions, given the unpredictable nature of the noise present in FMRI. The choice of this function will be discussed further in [section 4.4](#).

3.7 Simple, Nonlinear Example

A typical half wave rectifier takes a AC voltage circuit and removes one half (say the negative half) of the signal. The resulting waveform is still not DC, however it is then possible to use a capacitor to smooth the signal into something similar to DC, as shown in [Figure 3.3](#). There are other, more complex circuits that convert the negative portion into positive and waste less energy but here we will keep the system simple. Thus, let us consider a simple half wave rectifier circuit, shown in [Figure 3.2](#).

The half wave rectifier circuit smoothes the gaps between high voltage with a capacitor. Thus, when $u(t)G$ is less than v_t , the circuit will discharge the capacitor and maintain a non-zero voltage, but when $u(t)G$ is greater than v_t , the output voltage will be set by $u(t)G$ and the capacitor will charge up. We will assume a simple model for all the components, ignoring the complex nonlinear behavior that can occur in a diode.

For this example it will be assumed that the transformer simply scales the input by a constant factor, G . As discussed in the [section 3.2](#) any variable with uncertainty must be part of the state variable. Therefore the state variable is defined as: $X(t) = \{G, v_t, C, R_m, v_y\}$. The state equations would then be

$$v_y(t) = f(v_y(t-1), u(t)) = \begin{cases} u(t)G & \text{if } u(t)G - v_y \geq v_t \\ v_y(t-1) \left(1 - \frac{\delta t}{R_m C}\right) & \text{if } u(t)G - v_y < v_t \end{cases} \quad (3.18)$$

To run the particle filter is easy, since there exists a recursive definition of the dynamic state variable, v_y . To start with, an initial distribution must be assumed and while at first a Gaussian seems like a good idea, all the static state variables are strictly positive and thus not well suited to the Gaussian. Thus, instead the prior will start with a Gamma distribution. The gamma is defined as follows:

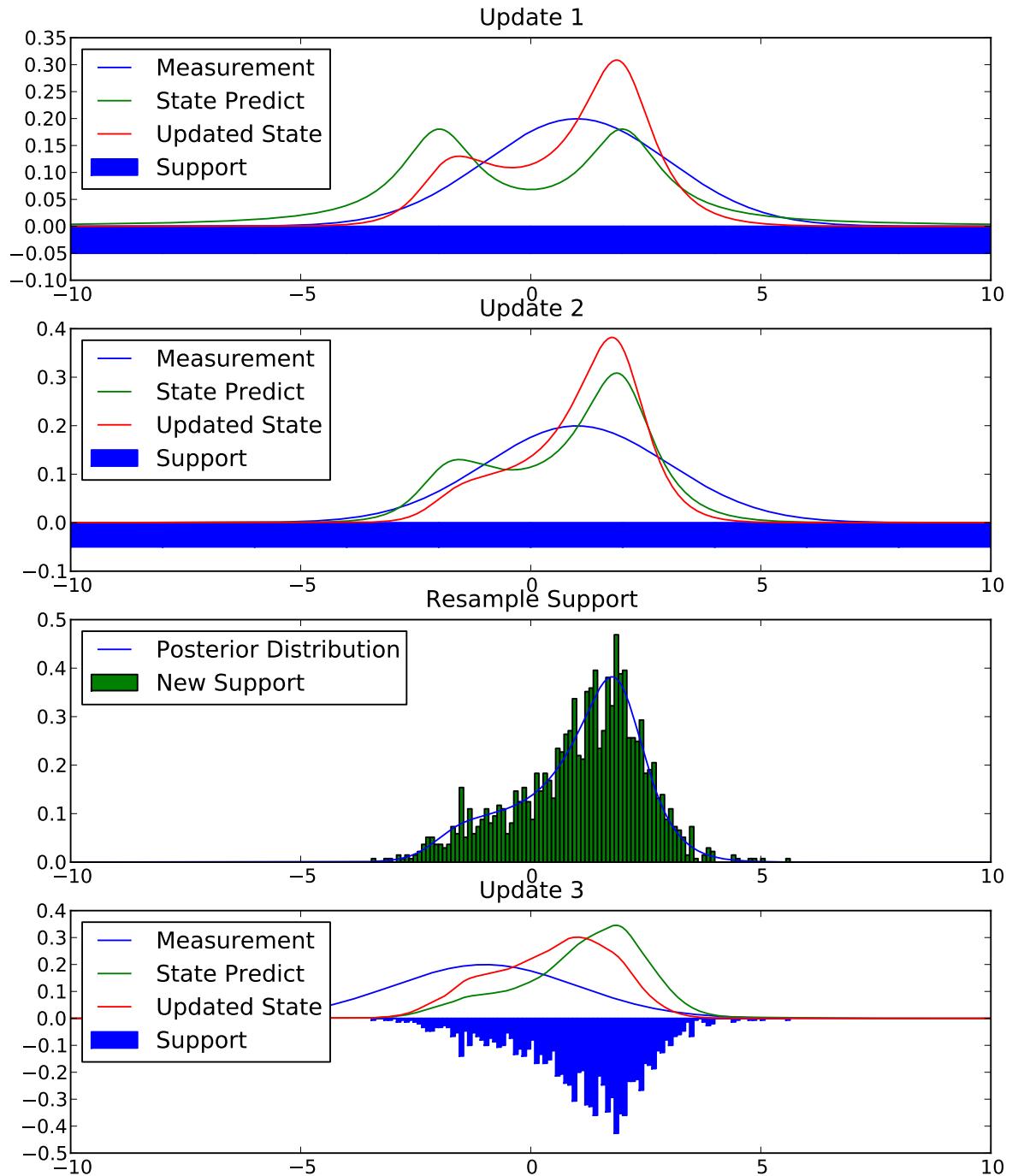


Figure 3.1: Particle Filter progression, note that the initial support is flat; the particles are equally spaced between -10 and 10

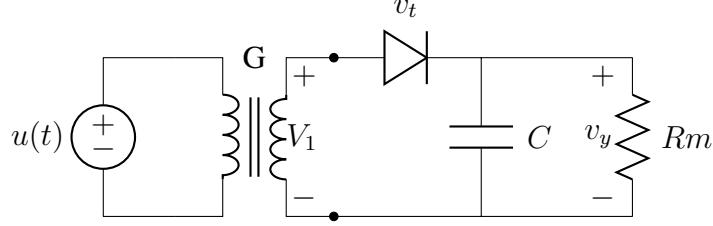


Figure 3.2: An Example Half Wave Rectifier Circuit, where G is the transformer gain, v_t is the activation voltage of the diode, $u(t)$ is the input at time t , C is the capacitance, R is the load resistance and v_y is the output voltage

Figure 3.3: Example Input/Output of the Half Wave Rectifier

$$X \sim \text{Gamma}(k, \theta) \rightarrow f(x) = x^{k-1} \frac{e^{-x/\theta}}{\theta^k \Gamma(k)} \quad (3.19)$$

where Γ is the gamma function. The margin for error is decided by the weighting function, which will be defined as $W(V_y, v_{yi})$, where V_y is the actual measurement, v_y is the estimate based on all the particles, and v_{yi} is the estimate by a particular (i^{th}) particle. The choice of this function is difficult, and although the Gaussian is typically used, in practice I found the exponential helpful in preventing particle deprivation. The algorithm is then,

Initialize N_p Particles:

for i in N_p **do**

$$G \sim \text{Gamma}\left(\frac{\mu_G^2}{\sigma_G^2}, \frac{\sigma_G^2}{\mu_G}\right)$$

$$v_t \sim \text{Gamma}\left(\frac{\mu_{v_t}^2}{\sigma_{v_t}^2}, \frac{\sigma_{v_t}^2}{\mu_{v_t}}\right)$$

$$C \sim \text{Gamma}\left(\frac{\mu_C^2}{\sigma_C^2}, \frac{\sigma_C^2}{\mu_C}\right)$$

$$R_m \sim \text{Gamma}\left(\frac{\mu_R^2}{\sigma_R^2}, \frac{\sigma_R^2}{\mu_R}\right)$$

$v_y = 0$, (Assume the system has been off for a long time)

let $X_i(0) = \{G, v_t, C, R_m, v_y\}$

let $w_i(0) = 1$ or to make a flat prior, $w_i(0) = \frac{1}{Pr(X_i(0))}$

end for

Run the Filter:

for t in Set of Measurement Times **do**

for i in N_p **do**

$$v_{yi}(t) = f(v_{yi}(t-1), u(t))$$

(All other members of $X_i(t)$ remain the same)

$$w_i(t) = w_i(t-1)W(V_y(t), v_y(t))$$

end for

end for

Initially the particles will have the same output, 0, however, as $u(t)$ changes, the response of each particle to that input will result in different outputs. Particles that have a v_{yi} near V_y will be weighted higher, and others farther away will be weighted lower. As the particle filter runs, weights will compound resulting in a distribution that asymptotically approaches the true joint distribution of the $X(t)$. Of course, as I mentioned in [section 3.5](#), particles weighted zero do not significantly contribute to the empirical distribution, so re-sampling may be necessary.

Chapter 4

Methods

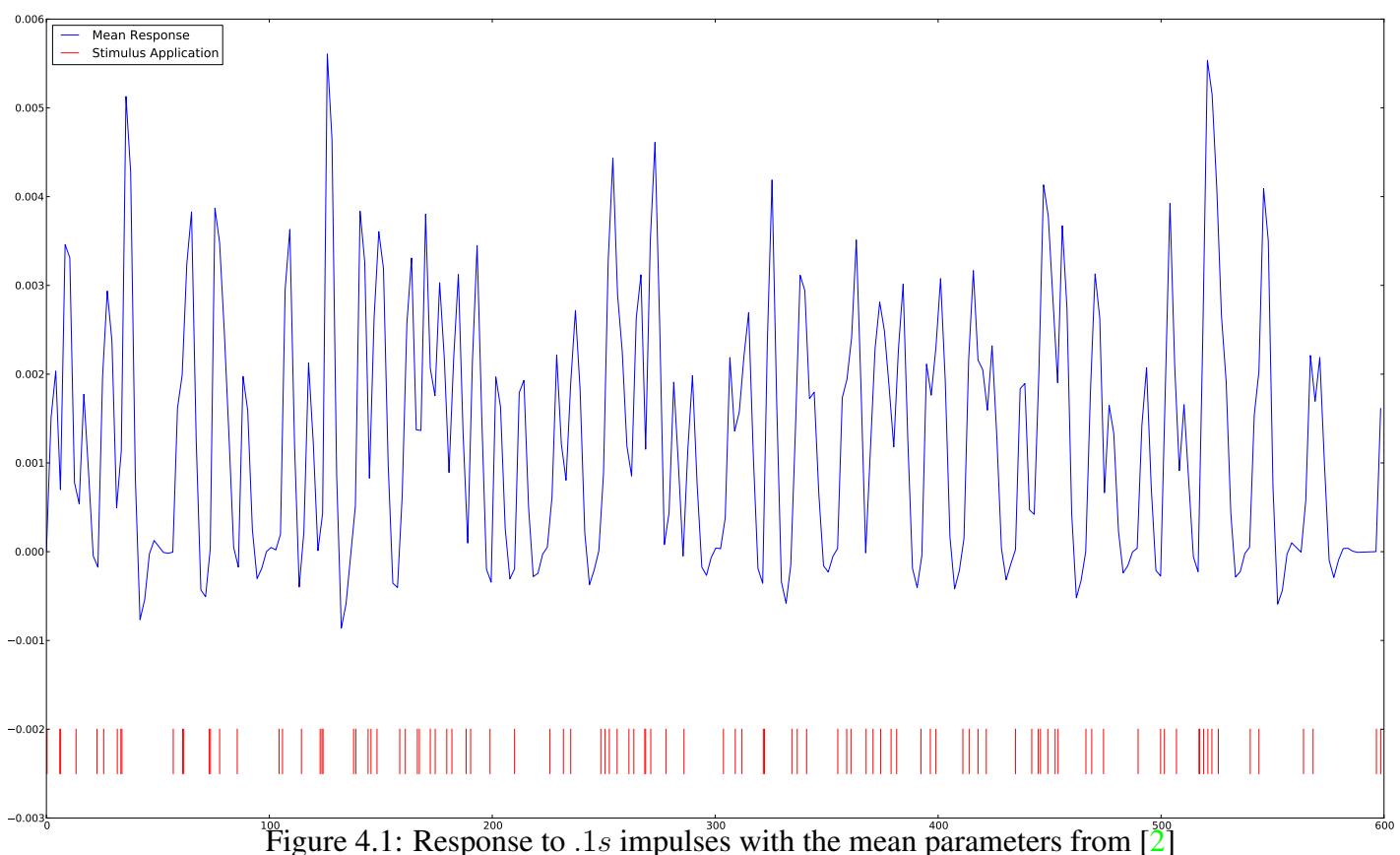
Although the particle filter I used for this work is a standard Regularized Particle filter, as described in [36], optimizing the particle filter for use with FMRI data is non-trivial.

4.1 Prior Distribution

For the BOLD model described in section 1.4, several different studies have endeavored to calculate parameters. The results of these studies may be found in Table 1.1, and the methods used for each may be found in chapter 2. Unfortunately, [2] only studied small regions; and most research with reasonable speed (including [34]) used these results as the source for their priors. The one exception is [3], which came to an extremely different distributions. For a particle filter, the choice of a prior is the single most important design choice. Consequently for this work, I wanted to be conservative; which meant going with the accepted result, [2]. This constrains the usefulness of the model to areas that fall within the prior distribution, yet will allow results to be comparable to other works. There is a significant need for better estimates of the physiological parameters; and, while physical experiments may not be possible, it would not be unreasonable to do a study with exhaustive simulated annealing or hill climbing tests for multiple regions and multiple patients. The purpose for this work is to determine the fitness of particle filters for this task. Ultimately this algorithm may become more useful as more studies are done on BOLD model that don't require priors.

There is an interesting anomaly with the priors found in virtually all the works that characterized the parameters, except [3]. The BOLD signal is universally recognized to be around 2–3%, maybe reaching 5% in extreme activation. Yet using the mean priors from [2], the signal response for a .1 second impulse only reaches maybe half a percent, as section 4.1 shows.

While this could be the result of a stimulus being too short to lead to strong activation, a similar stimulus scheme in real data showed a much larger response than half a percent as well. In



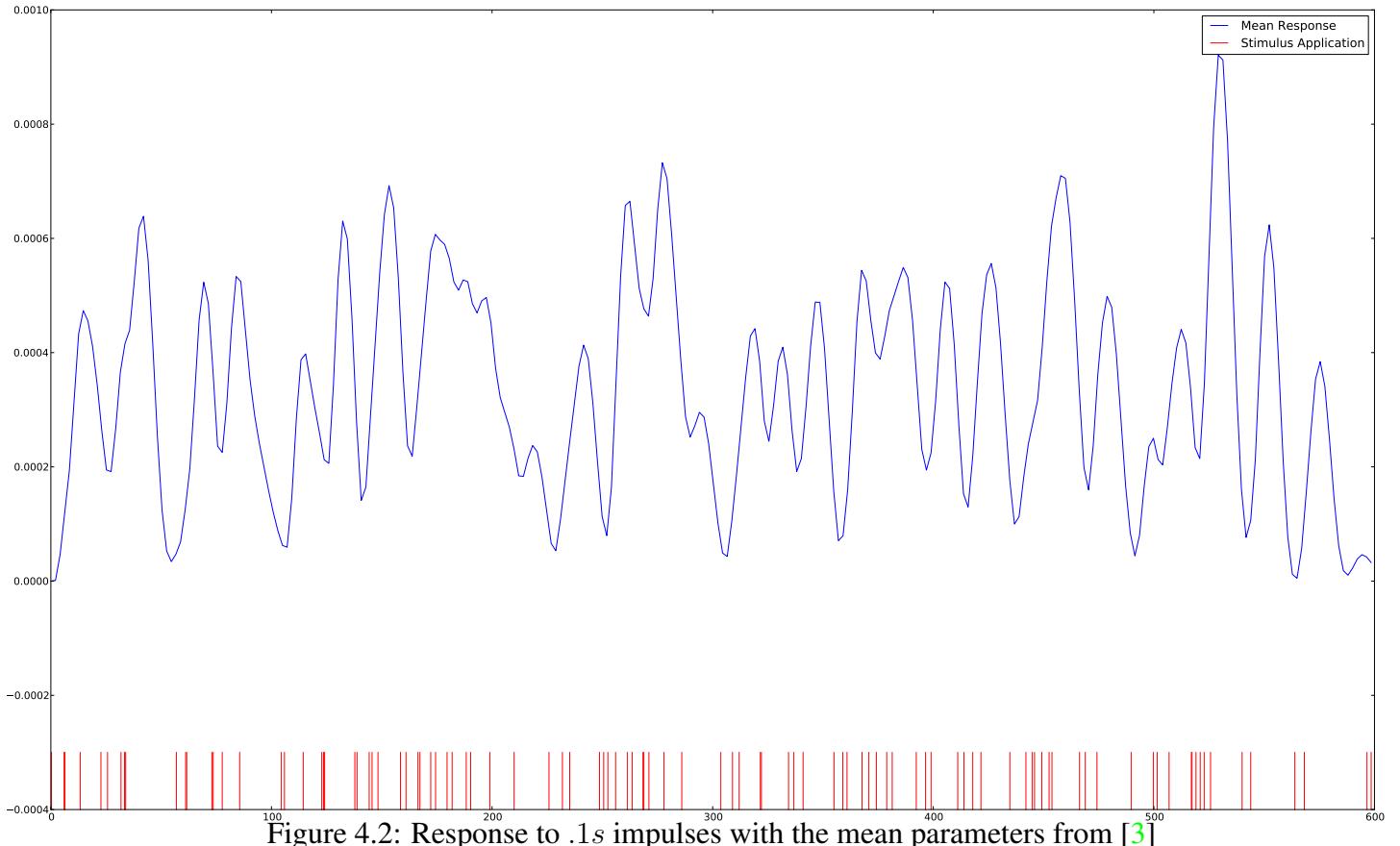


Figure 4.2: Response to .1s impulses with the mean parameters from [3]

fact, after applying de-trending, converting the image to percent-difference, and removing outliers ($BOLD > 10\%$ or $BOLD < -10\%$) the total variance across all *active* voxels was still around .02, indicating that in active voxels a signal peaking below .005 seems unlikely. Of course, if more restriction were placed on the outliers, its possible this standard deviations could be brought down. The parameter estimates by [3] are even more confusing, with peaks of well below .1% (Figure 4.1).

Its likely that these differences are due to some difference in preprocessing, although in [9] the signals were found to be peaking around 1%, unlike [2] which shows signals peaking at up to 3% or 4%. In my own tests, it seemed necessary for ϵ to reach well over 1.5 and V_0 to reach more than .4 to reach these peaks; of course other methods may be equally able. Therefore, to account for these discrepancies, somewhat broader distributions are used than the numbers used in [2] (which are widely used, [28]). The priors used in the particle filter may be found in Table 4.1.

Note that although the mean remains the same for all the parameters other than ϵ , the standard deviation is set much higher to account for the disagreement between studies (Table 1.1). Because all the parameters are taken to be strictly positive, and the standard deviations are approaching the mean, I used a gamma distribution. This prevents the Gaussian from placing parameters in the nonsensical territory of negative activation, or negative time constants.

Another aspect of the prior is using enough particles to get a sufficiently dense approximation of

Parameter	Distribution	μ	σ
τ_0	Gamma	.98	.25
α	Gamma	.33	.045
E_0	Gamma	.34	.03
V_0	Gamma	.04	.03
τ_s	Gamma	1.54	.25
τ_f	Gamma	2.46	.25
ϵ	Gamma	.7	.6

Table 4.1: Prior distributions used in the particle filter.

the prior. For 7 dimensions, getting a dense prior is difficult. Insufficiently dense particles will result in inconsistent results, of course the processing time will scale up directly with the number of particles. A dense initial estimate is important so that some particles land near the solution; but as the variance decreases the number of particles needed decreases as well. Thus, as a heuristic, initially the number of particles is set to 28,000, but after resampling, the number of particles is dropped to 1,000. Typically during the first few measurements the variance drops precipitously since most particles don't describe the system well. The particles that are left are in a much more compact location, allowing them to be estimated with significantly fewer particles. These numbers aren't set in stone, and depending on the complexity of the system or desired accuracy they could be changed; however, they seem to be the minimum that will give consistent results.

4.2 Model

As originally written in section 1.4 the state variables for the BOLD model are as follows:

$$\dot{s} = \epsilon u(t) - \frac{s}{\tau_s} - \frac{f - 1}{\tau_f} \quad (4.1)$$

$$\dot{f} = s \quad (4.2)$$

$$\dot{v} = \frac{1}{\tau_0}(f - v^\alpha) \quad (4.3)$$

$$\dot{q} = \frac{1}{\tau_0}\left(\frac{f(1 - (1 - E_0)^f)}{E_0} - \frac{q}{v^{1-1/\alpha}}\right) \quad (4.4)$$

The original assumption regarding particle filter models (section 3.2) included noise in the update of x , however that is not included here. The reason for the difference is that cloud of particles is, to some extent, able to model that noise. It is common, however, to actually model that noise in particle filters by adding a random value to each updated state variable. Because the purpose of this particle filter is to learn the underlying distribution of the static parameters, rather than precisely model the time course of the in the dynamic parameters ($\{s, f, v, q\}$) this noise is left out. Because the BOLD model is dissipative, when no stimuli are applied, all the particles will

decay to ($\{0, 1, 1, 1\}$) which they should. If evolution noise were added, the particles would then weight particles based on the results of that noise, rather than on the quality of the static parameters. Typical particle filters also use this state noise as an exploratory measure; however this method is less well suited when the system is dissipative.

Thus, at each time step, the states were linearized, and each state variable, s, f, v, q , were changed according their previous rate of change. Because of the difficulty of solving a system of nonlinear equations, I did not use the typical Runge-Kutta 4/5 technique to optimize step sizes. The cost of missing a feature in these differential equations typically leads to non-real numbers shortly down the road. Because the non-real numbers do not come until the solution has had the chance to update two or three more times, taking long steps can result in catastrophic and un-recoverable errors. Typically a step size of .001 was used, after finding that even .01 can at times lead to the state equations careening out of control.

4.3 Resampling

The algorithm for resampling is described in [section 3.5](#). The primary design decision for resampling is the regularizing kernel. As mentioned previously, the Gaussian kernel is convenient, because it is simple to sample from. As discussed in [section 3.5](#), as long as resampling is kept as a last resort, some amount of over-smoothing won't impair convergence. Therefore, for this work I chose a Gaussian kernel of bandwidth equal to the original distribution's covariance. Obviously this will apply a rather large amount of smoothing to the distribution; however, on average resampling is only applied every 20 to 30 measurements, and because randomization is being applied to model updates this gives the filter some mobility.

Because the regularized resampler will almost certainly over-smooth, it is necessary to wait until the N_{eff} drops relatively low (say below 25) before resampling is performed. As an additional measure against sharp drops in the N_{eff} that may quickly raise back up, resampling is only performed when two consecutive low N_{eff} 's are found. The danger in waiting longer to resample is that if there were no particles in the vicinity of the solution, then particle deprivation can occur. When this happens, the resampling stage will have an inappropriately small variance which will lead to an inappropriately small support for the distribution. After this, the distribution will be unrecoverable. Thus, when the N_{eff} gets down to extremely low values (below 5), usually this is an indication of particle deprivation; something that is also usually accompanied by fast drops in variance. When this happens, it is often preferable to return to a previous, wider estimate of scale for the regularization and continue from there. Thus to keep particle deprivation from affecting the results; a previous covariance matrix is used as the regularization bandwidth. In my algorithm I use the last covariance matrix for which the E_{eff} was greater than the threshold. This guarantees that there was at least some amount of particle diversity, and also helps prevent converging prematurely.

When some sort of particle deprivation has not occurred, regularized resampling with a large band-

width will slow down convergence. Thus as mentioned previously, to prevent over smoothing, the particle filter algorithm used here waits until the effective sample size drops below 25 for two measurement points in a row. This seemed to give good results, and avoids temporary drops in the N_{eff} caused by outliers.

4.4 Choosing $P(y_k|x_k)$

Choosing a representation of an unknown distribution tricky, and while the usual choice is a Gaussian distribution, there are reasons why it may not be the best. Studies of the noise in fMRI typically characterise noise as an additive noise process with Gaussian steps; a Wiener process.

As noted in section 4.5, the noise is not strictly Gaussian, nor is it strictly Wiener. As with any unknown noise however, it is necessary to make some assumption. If the weighting function ($P(y_k|x_k)$) exactly matches the measurement error, then the ideal particle filter will result. Particles with x_k 's that estimate y_k far out on the weighting function will quickly have weights near 0. Thus, an weighting function that exactly matches $P(Y(t)|X(t))$ will easily, and correctly throw out incorrect particles. However assuming that this function is not going to be found for arbitrary voxels, it is necessary to choose a function to approximate this probability. The cost of choosing an overly broad distribution for this function is an overly broad $P(x_k|y_k)$; and ultimately an overly broad $P(x_k|y_{0:k})$. On the other hand, an overly thin distribution will lead to, at best, an overly thin representation of these same distributions; but at worst will lead to particle deprivation (all particles being zero-weighted). If the *true* distribution were found, particle deprivation would indicate that that time series is not being affected in any way describable by the prior distribution/model. Particle deprivation due to an under-variant weighting function gives no information. Thus, the cost of a under estimating variance is higher; a fact that definitely needs to be taken into account.

Because of the need to err on the side of caution I tested several weighting functions. In addition to the Gaussian I also tested the Laplace and Cauchy distributions, both of which have much wider tales than the Gaussian. The benefit of the wider tailed distributions is that they don't down-weight particles quite as fast. Additionally, the Laplace distribution has the benefit of having a non-zero slope at the origin. This means that even if the distribution is made overly broad, it will still distinguish between particles that are near the origin.

After trial and error, I chose the weight standard deviation to be .005. Obviously the case of the Cauchy, this is simply the scale parameter. While I did attempt to automatically set the standard deviation, I found that the algorithm became too unpredictable. Essentially, if the weight function isn't fixed across voxels, very noisy time series with no actual signal will converge to nonsensical results. In the future, it may be possible to set the standard deviation by taking a small sample from resting data and using the sample standard deviation. Since this is the first attempt at using particle filters for modeling the BOLD model, in this work I set the standard deviation manually at .005, because it gives a more consistency and control.

4.5 BOLD Noise

As demonstrated in section 1.4 the BOLD response has been extensively studied and despite minor discrepancies, the cause of the BOLD signal is well known. However, as FMRI detects an aggregate signal over the space of cubic centimeters, there are plenty of sources of noise. Though local neurons act together (i.e. around the same time), the density of neurons, the density of capillaries, and slight differences in activation across a particular voxel can all lead to signal attenuation and noise.

A particularly difficult form of noise present in FMRI is a low frequency drift, often characterized as a Wiener process ([11]). Though not present in all regions, as much as ten to fifteen percent of voxels can be affected ([40]), thus it is prevalent enough to cause significant inference problems [31]. It is still not clear what exactly causes this noise noise comes from, although it is possible it is the result of magnets heating up, or some distortion in magnetic fields [31]. It is clear that this drift signal is not solely due to a physiological effects, given its presence in cadavers and phantoms [41]. Interestingly, it is usually spatially correlated, and more prevalent at interfaces between regions, though by no means limited to such areas. Though one potential source could be slight movement, given that co-registration of volumes to a single time point is standard, this is unlikely. Regardless, the problem mandates the use of a high pass filter to make inference reasonably powered [31].

In order to characterize the noise, I analyzed resting state data. During resting state, the patient is shown no images, and he is asked to avoid movement and complex thought, to the best of his abilities. Some limitations exist, for instance EPI sequences are noisy which could cause some stimulus in the auditory cortex, and of course long sequences could give the patient time for his mind to wander. Overall though there should be no activation, and thus the signal will consist entirely of noise. Therefore resting state data is perfect for analyzing the noise distribution. The locations were chosen from points all around the brain, all in grey matter voxels. The time series were also chosen because they were representative of different types of noise I found in the resting state data.

Because most methods (including the one used in this paper) assume the noise realizations are independent of each other, the auto-correlation is of particular interest (which is a necessary but not sufficient condition for independence). Gaussianity is also a common assumption made in studies of FMRI data, though that assumption is not made in this work. Regardless, comparing the distribution to a Gaussian is informative, so I used Q-Q plots to compare example data with the Normal distribution. Additionally, in FMRI data the noise is often considered to be Wiener [15]. Recall that a Wiener random process is characterized by steps that are Gaussian; in other words the difference between any two samples is Normally distributed. The simulations discussed in section 5.1 make use of this, by adding a Wiener random process to the overall signal. To determine whether the noise is in fact Wiener, the distribution of the difference between adjacent measurements was plotted against a Gaussian. Wiener processes further assume that the steps are independent; so the steps should have near flat autocorrelation functions.

Finally, removal of the drift is often performed with some variation of a high pass filter, so I

checked the noise distribution after applying such a filter (in this case the subtraction of a spline, see [section 4.6](#)). Here I wanted to know how effective my high pass filter at the removal of Wiener noise.

[Figure 4.3](#) shows the results with a regression line fit to the points on the Q-Q plot. Recall that in a Quartile-Quartile (Q-Q) plot, if the points plotted on the x-axis and the points on the y-axis come from the same *type* of distribution then all the points will fall on a line. Differences in the variance will cause the line to have a slope other than 1, while differences in the expected value will cause the fitted line to be shifted. In these Q-Q plots, the points are being compared to the standard Gaussian distribution, so the quality of the line fit determines how closely the points fit the Gaussian distribution. Note that in [Figure 4.3](#) the points have all been normalized (changed to percent difference).

A Wiener process should still conform to the normal distribution, with a variance proportional to the run-time. Note that [4.3\(a\)](#) and [4.3\(b\)](#) are well described by a Gaussian process with a small autocorrelation, [4.3\(c\)](#) and [4.3\(d\)](#) are not. In particular the tails of [4.3\(c\)](#) do not seem to fit the Gaussian well. Also note the significant autocorrelation in [4.3\(c\)](#) and [4.3\(d\)](#). As expected, the noise is not strictly Gaussian white noise. On the other hand, the steps do conform rather closely to the normal distribution. As expected, most of the autocorrelation disappears for the step data. Given that the steps seem to fit the Normal distribution, the low autocorrelation indicates that the steps could be Independently Distributed. Therefore, the noise does come close to a Wiener process.

De-trending the time-series by subtracting a spline fit to the distribution removed much of the autocorrelation present in [4.3\(c\)](#) and [4.3\(d\)](#), though not perfectly. Though the distributions still do not exactly fit the Normal, [4.5\(d\)](#) is much improved compared to [4.3\(d\)](#). In all, the de-trending is effectively removing Wiener noise.

4.5.1 Detrending

The non-stationary aspect of a Weiner process, presumably the result of integrating some ν_x is difficult to compensate for, and so methods have been developed to compensate for it. [\[40\]](#) and [\[41\]](#) have demonstrated that this component is prevalent, and may in fact be an inherent characteristic of fMRI. In some studies, as many as half the voxels benefit from detrending, presenting a serious barrier to inference ([\[31\]](#)). All the existing methods are performed during the preprocessing stage, rather than as an integral part of analyzing the BOLD signal. There is no shortage of detrending methods; however in a head to head comparison, [\[40\]](#), showed that in most cases subtracting off a spline works the best. The benefit of the spline versus wavelets, high pass filtering or other DC removal techniques is that the frequency response is not set. Rather, the spline is adaptive to the input, having a low cut off if the signal's median stays constant but a high cut off frequency if the signal's median tends to shift heavily over the course of time. In spite of this, the spline will still remove some amount of signal.

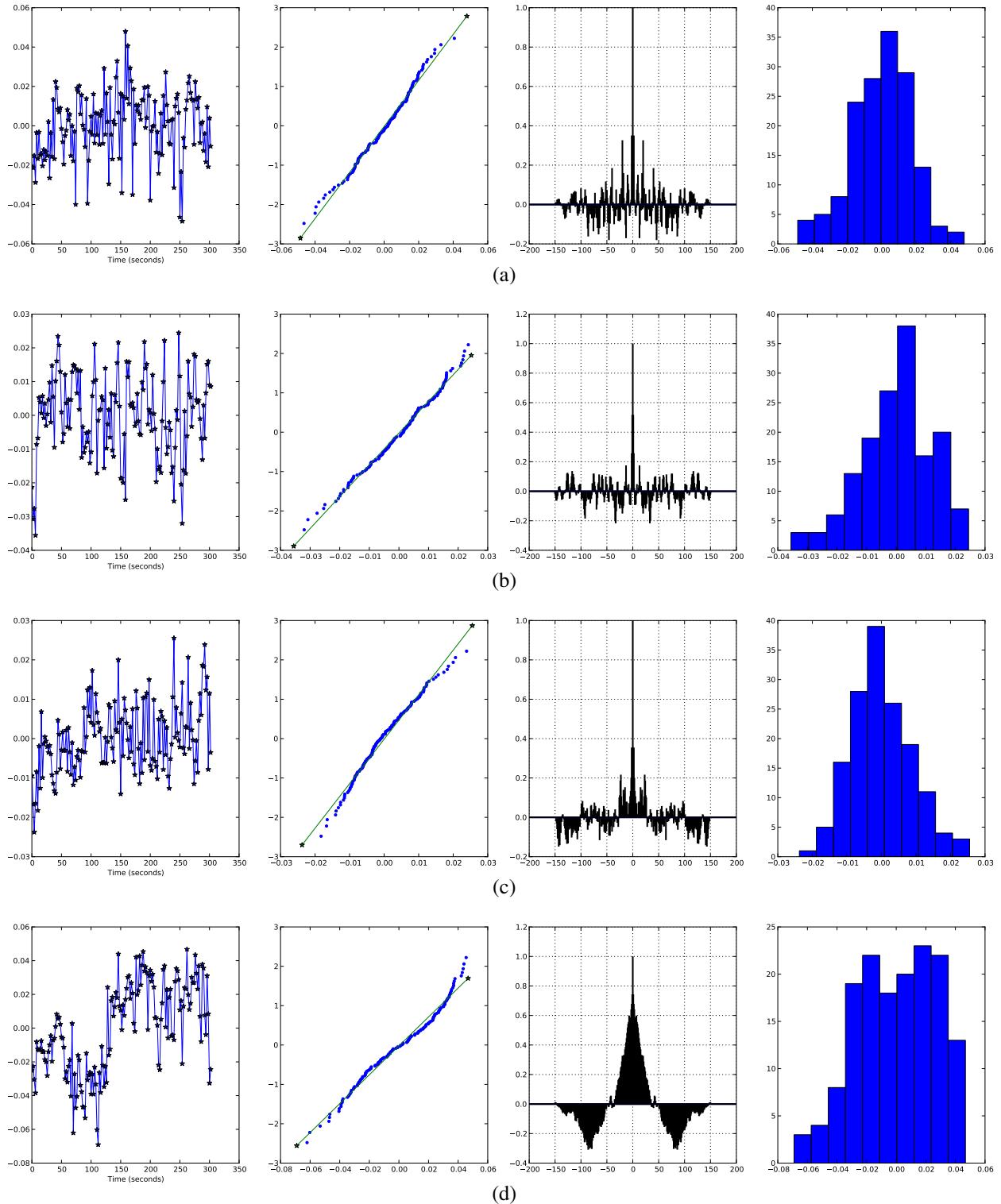


Figure 4.3: Q-Q Plots of normalized resting state data

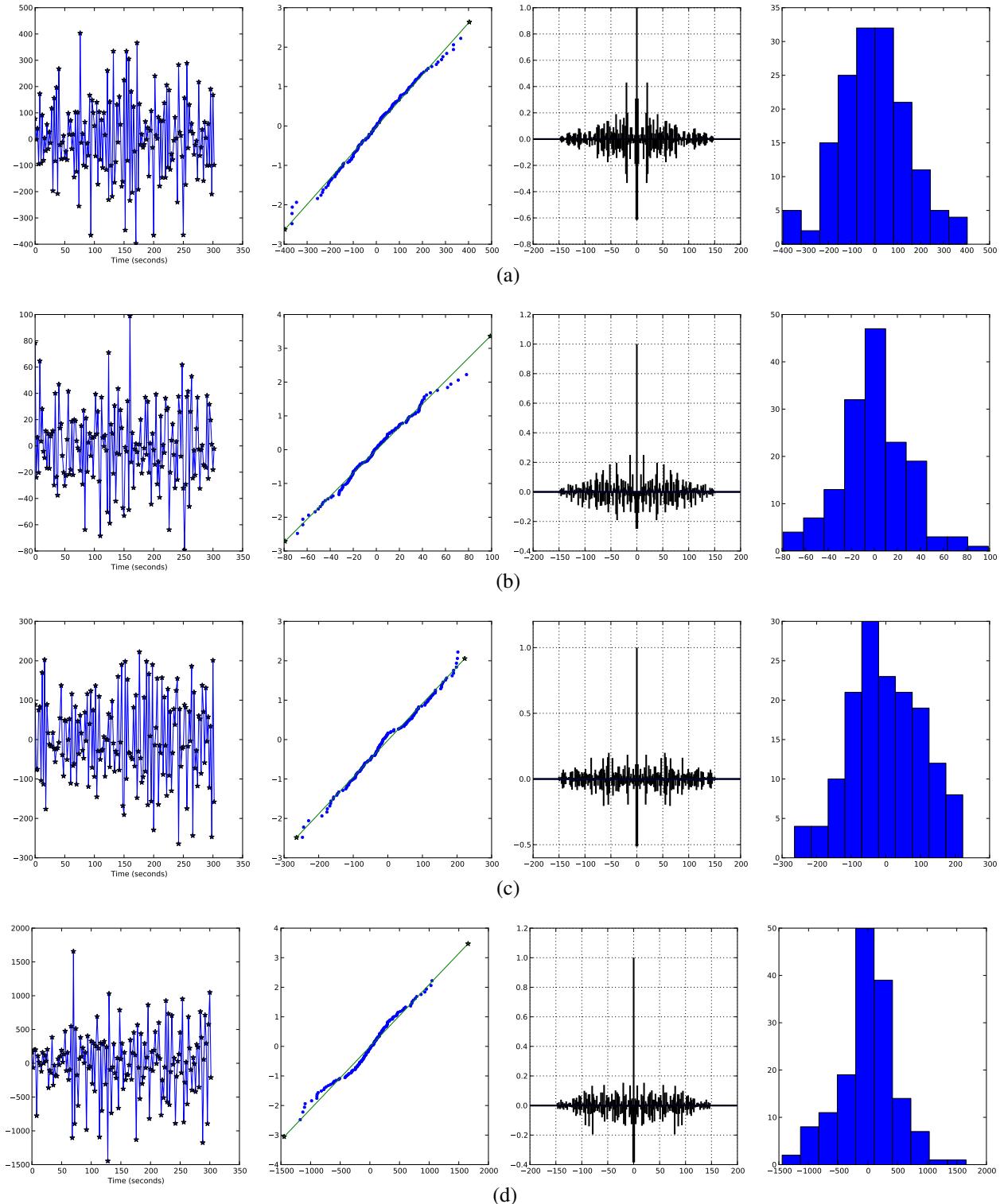


Figure 4.4: Q-Q Plots of resting state data, using the BOLD signal changes

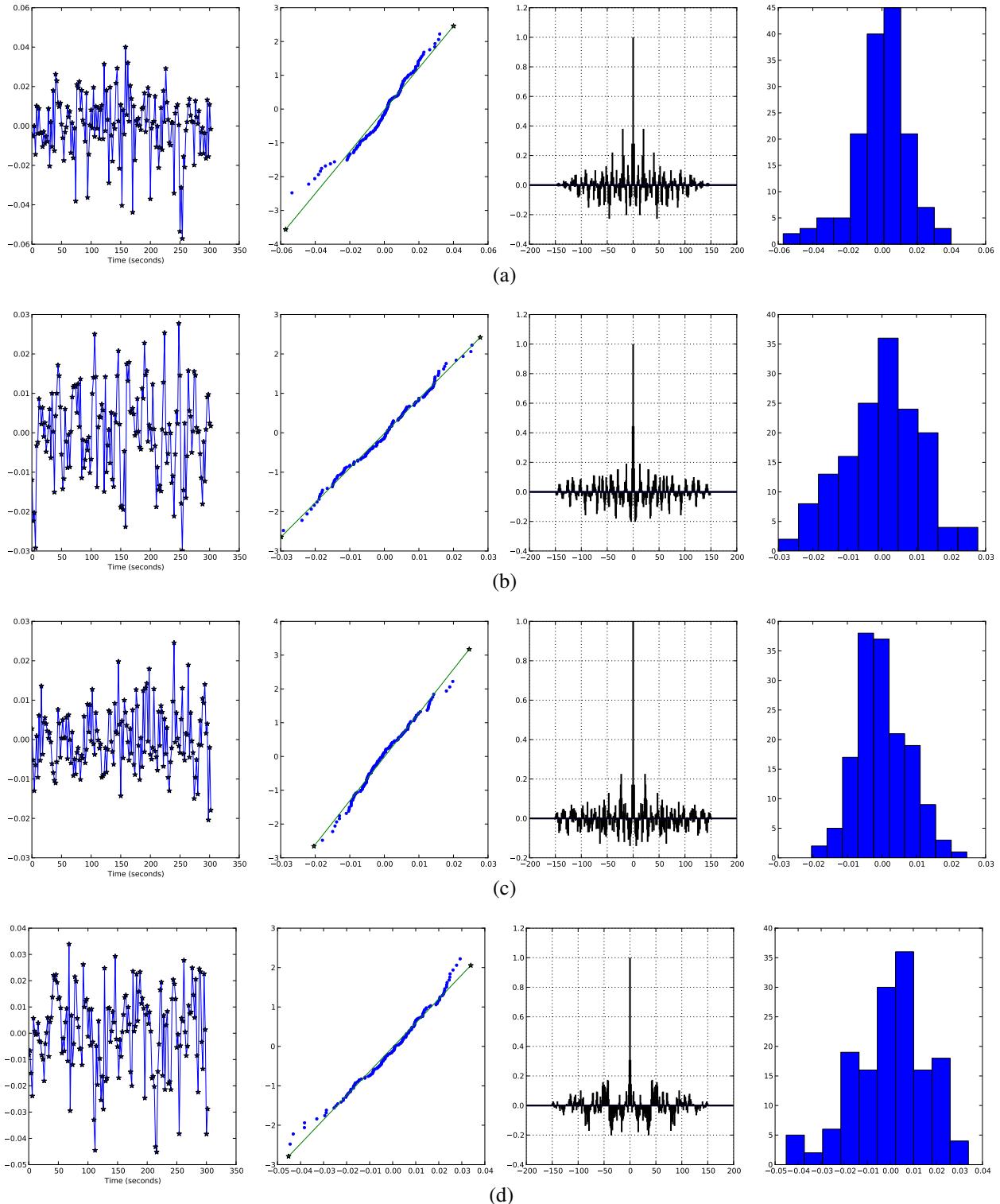


Figure 4.5: Q-Q Plots of resting state data, after the de-trending

The method I used to calculate the spline was picking one knot for every 20 measurements in an image. Thus a 10 minute session at a repetition time of 2.1 seconds would have 19 knots. The knot first and last knots were each given half the number of samples as the rest of the knots; which were all located at the center of their sample group. The median of each sample group was then taken and used as the magnitude for the group. Taking the median versus the mean seemed to work better, given the presence of outliers. There is potential to optimize the spline further using a canonical HRF to find resting points; however, for this to work the experiment would have to be designed with this in mind.

When using the median-based spline techniques, the normalized signal will, by definition have a median near zero. The problem with this is this is not the natural state of the BOLD signal. More specifically, when the signal is inactive, the BOLD response should be at 0% change from the base level; activation may then increase, or for short periods decrease from this base. After removing the spline, the BOLD resting state will be below 0%. This is problematic because it reduces the ability of an algorithm to learn. One quick and dirty solution is to add an arbitrary constant to each BOLD response. Of course this won't scale to whole-brain analysis, so a more effective technique is adding a DC gain parameter to the BOLD model. Like all the other model parameters, given enough measurements, correct value may be found. On the downside, adding another dimension increases the complexity of the model, for a variable that is obvious by direct observation.

Thus, I used a more conventional approach to deal with this. To determine the DC gain to be added to the signal I used a robust estimator of scale. The Median Absolute Deviation (MAD) proved to be accurate in determining how much to shift the signal up by. I tested both methods during the course of analysis, and found that the increase in model complexity far outweighed the slight increase in flexibility. Its possible that a more accurate method may exist; however, for this case the MAD works well, as [Figure 5.2](#) and [Figure 5.5](#) show.

$$y_{\text{gain},0:K} = 2\text{median}_{i=0:K}(y_i - \text{median}(y_{0:K})) \quad (4.5)$$

A serious concern with adding and subtracting arbitrary values to real data is whether this will create false positives. This is a legitimate concern; however, all the additions and subtractions done in this section have been low frequency changes and should not substantially change the BOLD response, being a higher frequency signal. The typical method of preprocessing is a high-pass filter with a cutoff of around 30 seconds; which should about match the spline with knots every fifteen measurements. [40] found that splines tended to far outperform the high-pass filter method.

4.6 Preprocessing

As discussed in the section on de-trending, the normal pipeline for analyzing FMRI involves a several preprocessing steps. The first and most important task is motion correction. To do this, a

single volume in time is chosen, and volumes at every other time are registered to this one volume. This corrects for motion by the patient as well as small changes in the magnetic fields that cause the image to shift. In conventional statistical parametric mapping, a gaussian smoothing filter is applied across the image as discussed in subsection 2.1.2. After this, detrending is performed as discussed by subsection 4.5.1. Recall that FMRI signal levels are unit-less and though detrending is not always necessary, at the very least the data must be converted into % difference from the baseline. Changing to % difference removes no real information from signal. This is the signal that was input into the delta based particle filter. Of course, most of the time analysis is performed on the direct signal; which mandates the removal of low frequency drift. The generally accepted method is to use a high pass filter, although the cutoff frequency is application dependent and often applied haphazardly.

4.7 Particle Filter Settings

The run-time for a single voxel depends on the several factors. First, the overall length of the signal being analyzed. For 1000 measurements it takes about 6 minutes. On the other hand, in real circumstances the length is only around 150 measurements and takes around 1 minute. The size of local linearization steps can certainly also make a large difference; and although it can be tempting to decrease the resolution, going above .001 seconds per step is not recommended. In most cases millisecond resolution is fine; however, when generating simulated data I found that every once in a while this was not enough. This is problematic in the actual particle filter since, given the large number of simultaneous integrations taking place, its probable that a few particles will fail and be unfairly thrown away. There are two possible outcomes of this. The typical case is at some point where all the particles are in a similar location and so all the particles fail together. The result is then particle deprivation - no particles with non-zero weights remain. At best this will cause a convergence to a less than optimal posterior probability, at worse at will result in a false negative. The other possible outcome is that low time constant (or fast moving) particles get pruned, leaving an excessively smoothed estimate of the output.

Although runge-kutta solvers could be applied, this will still not guarantee that features aren't missed. Its possible that a kind of stop-gap measure could be put into place; wherein particles that are about to be set to NaN are integrated again with finer grained steps. However oftentimes the non-real results don't occur until several time steps after the numbers become unrealistic. So for instance, the time step was too long, allowing f to go negative, resulting in extremely large values of q . There are many different ways where this sort of event can occur, and unfortunately sometimes there is no way to get back to before the state starting going out of control.

Another crucial factor for run time is how long before the first re-sampling occurs. Because the prior is represented initially with significantly more particles, if for some reason the effective number of particles stays high, resampling could take a long time to occur. For this reason, rather than allowing the particle filter to continue on with this large number of particles, after 20 seconds have

passed the algorithm forces resampling. Of course the choice of 20 seconds is somewhat arbitrary, but at the very least this gives some time for the particle to be re-weighted before the new distribution is drawn. Still the distribution will then be a thinned out version of the prior.

4.8 FMRI Configuration

For the FMRI data discussed in [chapter 6](#), tests were performed on a right handed volunteer using a Gradient EPI sequence on a 1.5T GE scanner with a 2.1s TR. Slices were 5mm thick and echo time was (todo).

Chapter 5

Simulation

Two levels of simulation are discussed in this section. First the simulation of single time series and second, a slice of simulated FMRI time series. The single time series tests are necessary to investigate the power of particle filters in identifying the BOLD model in a noisy environment. If the particle filter is unable to get back to the *true* time-series in spite of noise then the particle filter will not be useful. Additionally, determination of reasonable parameters from the time-series is also desirable. The reason to then simulate an entire slice is that Physics-Oriented Simulated Scanner for Understanding MRI (POSSUM) models noise more realistically, for instance by adding noise in the frequency domain. It also provides insight into the applicability of particle filters in modeling the BOLD signal on a large scale. Both sorts of tests are useful in determining the consistency of the particle filter. For the single voxel tests, each test will actually be performed eleven different times, each with a new noise realization. If runs fail to have consistent estimates of the underlying BOLD signal, then this would indicate a large variance error in the model. Similarly, in the Possum simulation there are actually only four discrete parameter sets driving the time-series. Nevertheless each voxel will have a different tissue composition and a different noise realization. Therefore the Possum simulation should also provide a good test of the consistency of the BOLD model.

5.1 Single Time Series

Given the state-space equations for the BOLD signal, simulating a single time series is relatively straight forward. After generating a true signal, identically and independently distributed (I.I.D.) Gaussian noise and a Wiener process with Gaussian I.I.D. steps were added to this true signal. Finally a carrier level is added, since BOLD is typically measured as a % difference from the baseline. The particle filter algorithm will immediately remove this by calculating the % difference, but adding a carrier level meant that the exact same algorithm used for simulated data could be used for the real data.

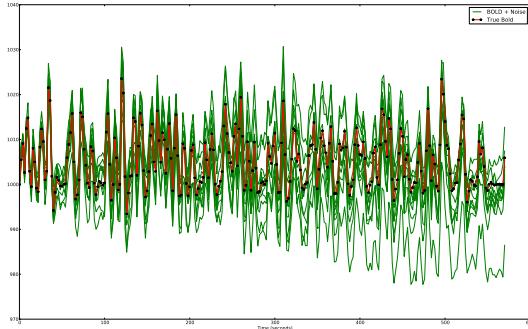


Figure 5.1: Test Signals with low noise compared to the clean signal.

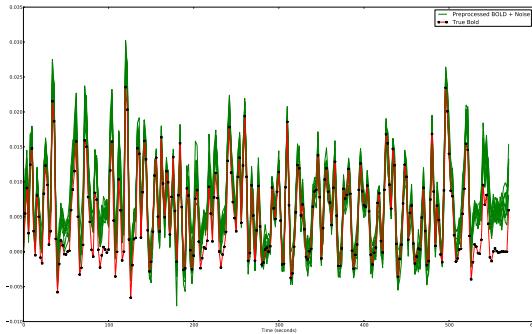


Figure 5.2: A comparison of the preprocessed signals for the low noise case. This is the noisy input to the actual particle filter algorithm.

Once this noisy simulated time series was generated, the particle filter algorithm was then run on single time series image. Here I will include two sets of tests to determine the power of the particle filter in modeling. The first set are used to demonstrate the ability of the particle filter to find the most likely set of parameters/state variables over the course of the run. The second type of test will investigate the problem of false-positives. As the first round of tests will show, given that there is an underlying BOLD signal, the particle filter is excellent at finding the most probable cause of the observed signal. However, as discussed in the false positive tests, even if there is no underlying signal, the particle filter will still determine the most likely BOLD signal; which may not be flat. Thus, it is important to investigate methods of determining false positives using other methods.

5.1.1 Signal with Low Noise

To begin the single voxel simulation; I generated a signal using the following parameters: $\{\tau_0 = 1.45, \alpha = .3, E_0 = .47, V_0 = .044, \tau_s = 1.94, \tau_f = 1.99, \epsilon = 1.8\}$. These same parameters were also used in the high noise case. The noise realizations were then generated based on a measurement noise (σ_y) of .001 and a drift standard deviation (σ_x) of .0005. The measurement noise as well as the steps of the drift were taken to be Gaussian For this low noise case, the eleven

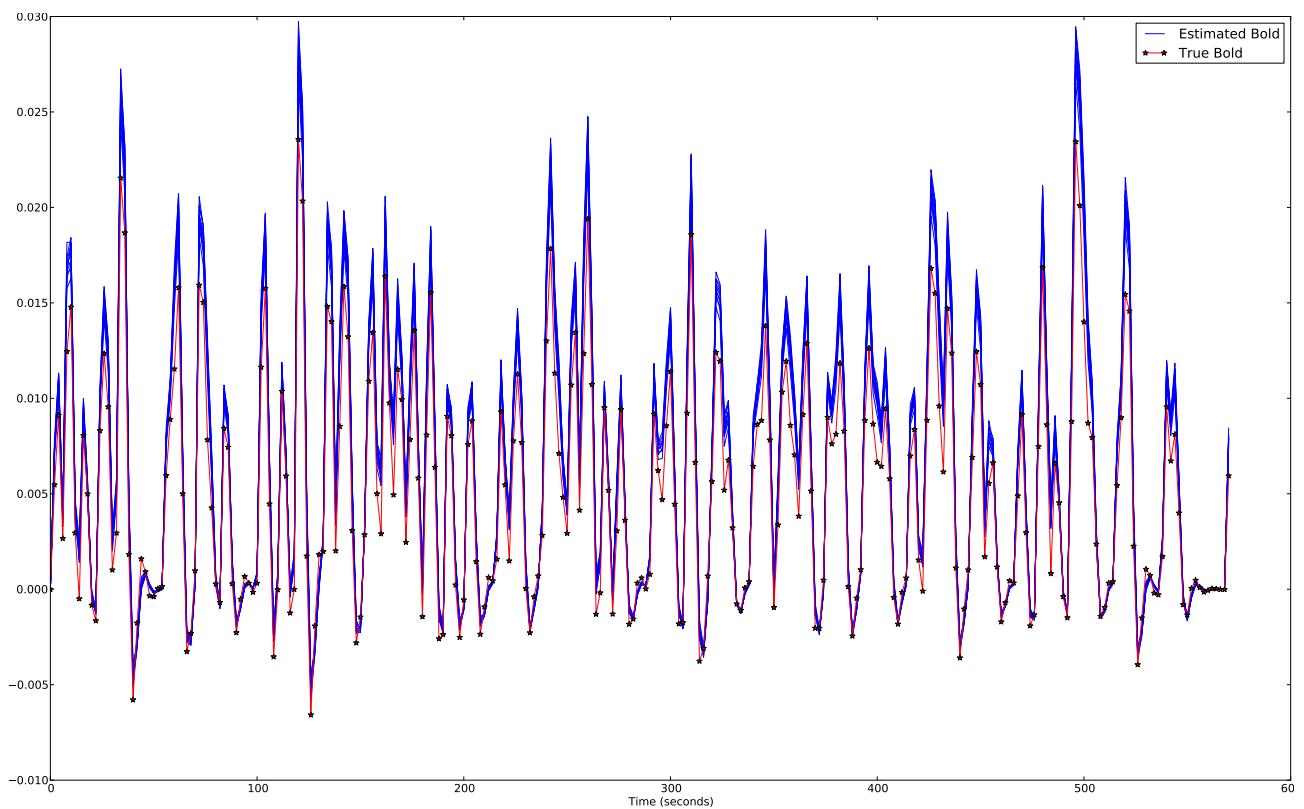


Figure 5.3: A comparison of the fitted signals for the low noise case.

realizations are shown in [Figure 5.1](#). The actual signal delivered into the particle filter was the result of some preprocessing to remove drift, as described in [section 4.6](#). The bias introduced into the signal by preprocessing may have some effect on the resulting fit; thus the preprocessed signal is compared to the true BOLD signal in [Figure 5.2](#). The preprocessing consists of several steps, which are discussed in detail in [section 4.6](#). The data shows that the spline struggles to match the signal near the end, although overall, the preprocessing seems to have been successful at removing trends. Finally, the set of fits to the true BOLD signal are shown in [Figure 5.3](#). The filtering property of the particle filter is clear from the results here. In several locations the output of the particle filter looks like a filtered version of the input. For instance toward the end, the estimates stay flat in spite of the preprocessed data drifting off. By this point, the algorithm has converged sufficiently to know that such a movement in signal isn't possible. A similar circumstance occurs at 100 seconds in. It is also worth noting the bias introduced by pre-processing results in a slightly larger peak signal. The final parameter sets are shown in [Table 5.1](#). Note that throughout the results, unless otherwise specified, the residual will be defined as the square root of the mean squared residual,

$$\text{Residual} = \sqrt{\sum (\hat{y}_k - y_k)^2} \quad (5.1)$$

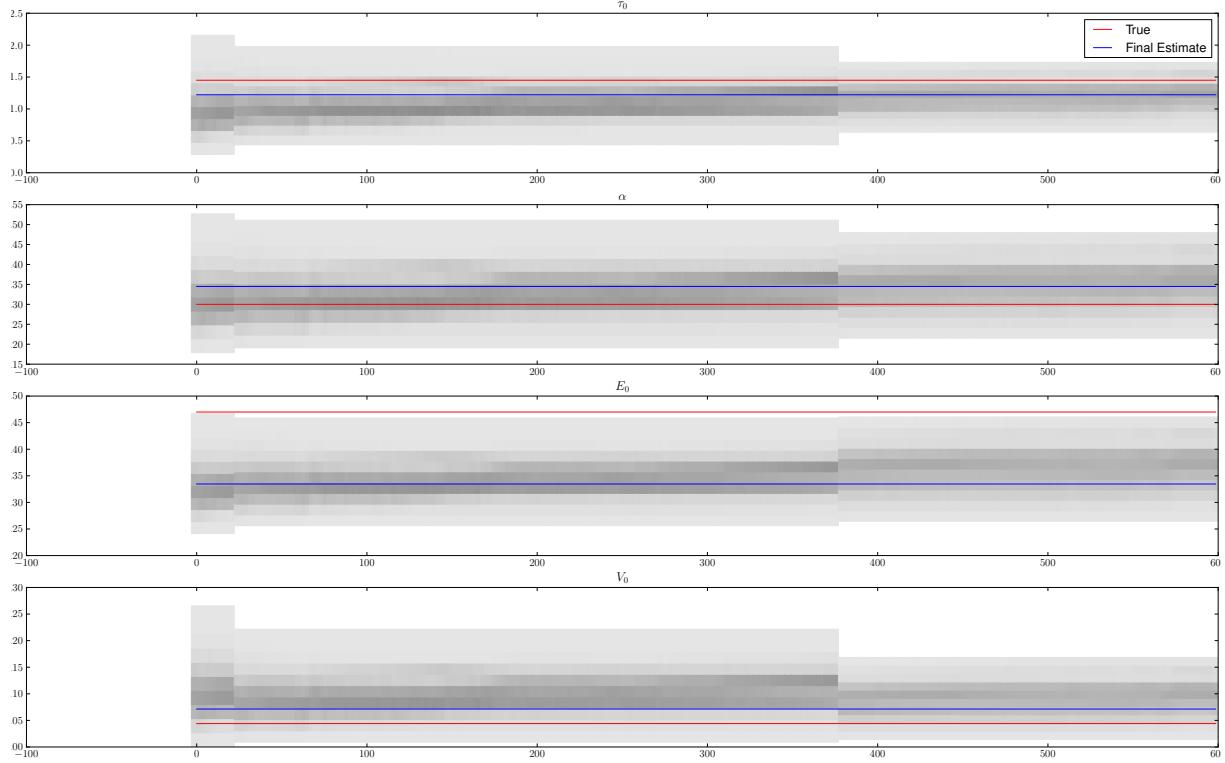
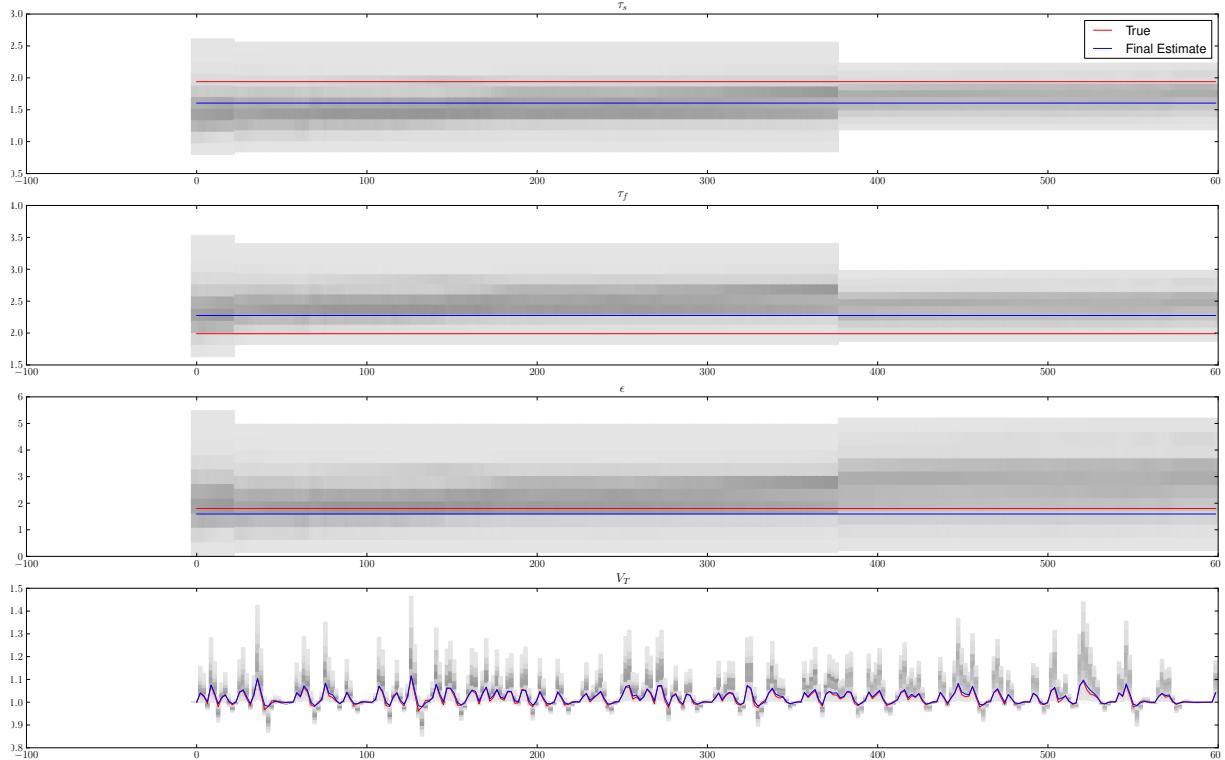
where \hat{y}_k is the estimated output at time k and y_k is the output sample at time k . Note the subtle distinction between this and the error which is defines as

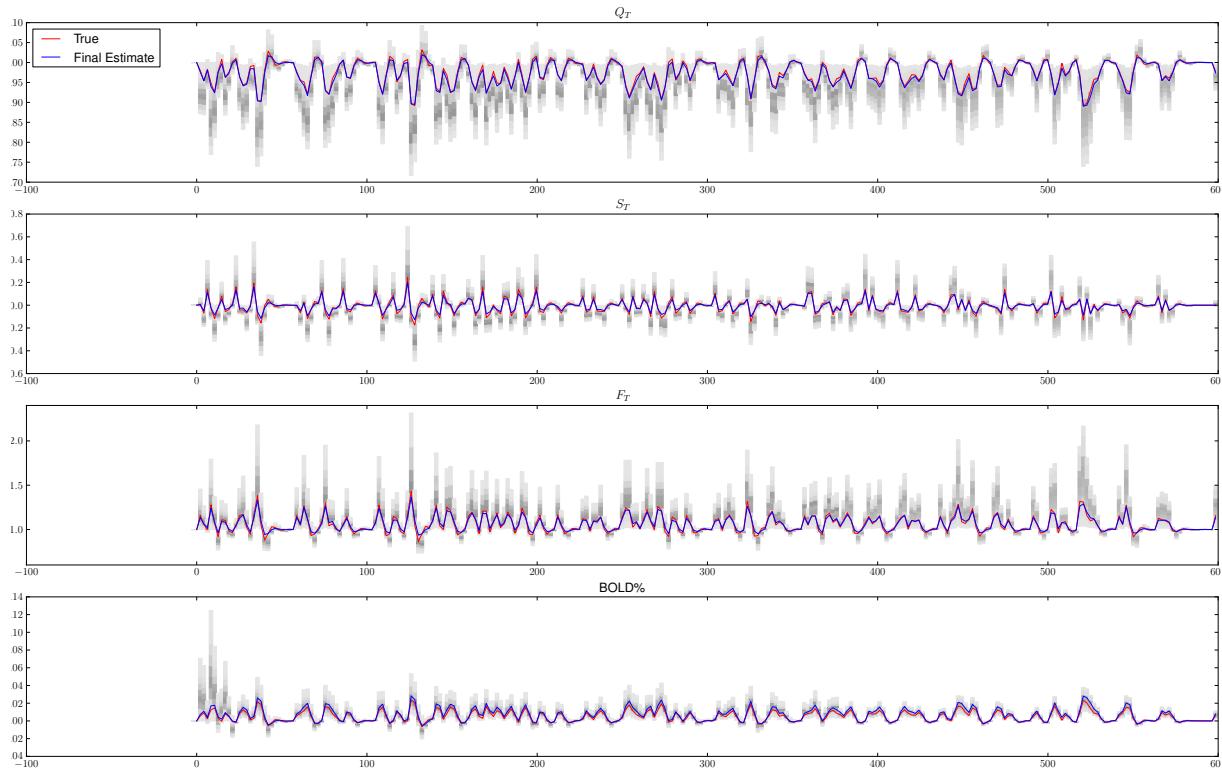
$$\text{Error} = \sqrt{\sum (\hat{y}_k - Y_k)^2} \quad (5.2)$$

where \hat{y}_k is the estimated output, and Y_k is the underlying (free of noise) output. Throughout this section and the next, the terms *error* and *residual* will be referencing these values, for an entire timeseries.

There are a few results worth noting in the final parameter estimates (which are the mean of the particle filter's posterior distribution). First the time constants vary greatly across runs with different noise realizations, yet the sum of the individual time constants (τ_f , τ_s and τ_0) seems to be more consistent. In general the time constants fall short of the true time constant. This could be a limitation based on the prior distribution (which notably has an initial mean below the true values) or it could be caused by the disassociated benefits of a correct time constant with regards to a correct output. It often takes several measurement periods before a difference in time constants becomes relevant to the weight. Its also possible that output is insensitive to differences in the time constants; although varying the activation durations could induce better estimates. The large variation of V_0 are also interesting, although as one final covariance matrix shows, there is a definite negative correlation between ϵ and V_0 . In general, with this admittedly small amount of noise, it would appear that the relation between a set of parameters/stimuli and the output is not injective; a time series is not unique to a single set of parameters. This is good justification that simultaneous blood volume or tagged flow calculations with the conventional FMRI could benefit the model.

The covariance matrix ([Table 5.2](#)) provides a great deal of insight into the way parameters evolve in the particle filter. The fact that there are so many covariance elements that are as big or bigger than variance elements indicates just loose the parameters set is. Every single parameters, including

(a) Converging histogram for τ_0 , α , E_0 , and V_0 of the first run, low noise simulation.(b) Converging histogram for τ_s , τ_f , ϵ , and V of the first run, low noise simulation.



(c) Converging histogram for Q , S , F , and $BOLD$ of the first run, low noise simulation.

τ_0	α	E_0	V_0	τ_s	τ_f	ϵ	$\sum \tau$	\sqrt{MSR}	\sqrt{MSE}
1.45	.3	.47	.044	1.94	1.99	1.8	5.38		
1.2221	0.3449	0.3346	0.0714	1.6045	2.2753	1.5945	5.1019	0.003211	0.009876
1.3749	0.3318	0.3630	0.0733	1.6408	2.1030	1.5763	5.1187	0.003055	0.009932
1.1660	0.3221	0.3406	0.0822	1.6477	2.3535	1.2452	5.1672	0.003289	0.009680
1.2318	0.3271	0.3403	0.0796	1.6270	2.1852	1.3033	5.0439	0.002847	0.009120
1.1832	0.3179	0.3472	0.0821	1.5496	2.2912	1.2782	5.0240	0.003006	0.009713
1.1424	0.334	0.3473	0.0737	1.6221	2.2908	1.4025	5.0553	0.002833	0.009485
1.3004	0.3596	0.3564	0.0768	1.5641	2.1323	1.6034	4.9968	0.003028	0.010219
1.2401	0.3460	0.3398	0.0891	1.6499	2.2366	1.2900	5.1265	0.003044	0.010080
1.1709	0.3274	0.3464	0.0826	1.5373	2.2826	1.3783	4.9909	0.003345	0.010329
1.1897	0.3434	0.3355	0.0798	1.5358	2.3075	1.4277	5.0330	0.003175	0.010015
1.184	0.3405	0.3502	0.0892	1.6103	2.2793	1.1645	5.0735	0.002889	0.009505
1.2187	0.3359	0.3456	0.0800	1.599	2.2488	1.3876	5.0665	0.003066	0.009814

Table 5.1: Estimated Parameters on 11 different runs with low noise. First row is the true parameters, last is mean over the 11 runs. The \sqrt{MSR} is residual and \sqrt{MSE} is the error.

	τ_0	α	E_0	V_0	τ_s	τ_f	ϵ
τ_0	0.0189481	-0.0014269	-0.0011267	-1.13e-05	-0.0025616	-0.0189559	0.0070405
α	-0.0014269	0.0026716	9.93e-05	-0.0002041	-0.0008632	-0.0016823	0.0071891
E_0	-0.0011267	9.93e-05	0.0010701	-0.0002277	-0.0001177	0.0001013	0.0016972
V_0	-1.13e-05	-0.0002041	-0.0002277	0.0005401	4.3e-06	4.56e-05	-0.0080494
τ_s	-0.0025616	-0.0008632	-0.0001177	4.3e-06	0.0128056	0.012878	-0.005516
τ_f	-0.0189559	-0.0016823	0.0001013	4.56e-05	0.012878	0.0416927	-0.0158182
ϵ	0.0070405	0.0071891	0.0016972	-0.0080494	-0.005516	-0.0158182	0.1567165

Table 5.2: Typical Covariance matrix of the parameters at the end of a run.

α seems to have a non-zero covariance. Why α and ϵ would be positively related is one of the most interesting element. Given the difficulty in mapping out a seven dimensional space, it is difficult to know why such correlations exist. Notice the covariance of τ_f and τ_0 is -0.019 whereas the variance of τ_0 and τ_f are 0.019 and 0.04 respectively. Clearly there is some measure of ill-defined behavior between the different time-constants, which is to be expected given the number steps before a stimuli affects the output. The convergence properties of the first run in [Table 5.1](#) demonstrates the migration of parameters to match the estimates. Given the relatively high variance of parameters, its likely that more measurements or perhaps more variation in the stimuli could further differentiate the parameters. Its also notable that the final mean of the parameters may not be the best point estimator, and that perhaps using the mode would be a better way to estimate the output. Regardless, the time series' generated from the estimated mean seems to match the correct signal very well ([Figure 5.3](#)).

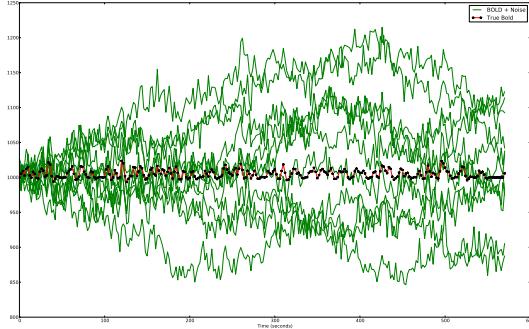


Figure 5.4: Test Signals with high noise compared to the clean signal, $\sigma_x = .01$, $\sigma_y = .005$

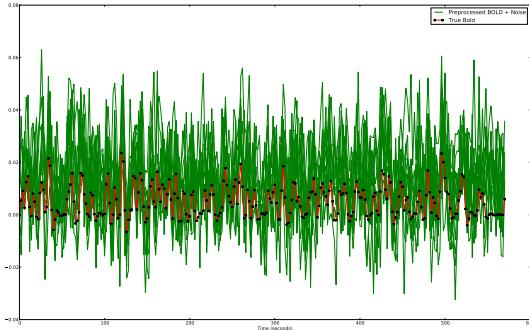


Figure 5.5: A comparison of the preprocessed signals for the high noise case.

5.1.2 High Noise

For the high noise simulation, the exact same procedure was performed as with the lower noise simulation except that σ_y and σ_x were set to .01 and .005, respectively. This is an order of magnitude higher than the previous tests, and indeed the noise appears to dominate the output, as seen in Figure 5.4. As before, the results of preprocessing are helpful in identifying the reason for bias in the estimated output signal; thus the preprocessed signals are shown in Figure 5.5. The results of the particle filter for each of the eleven runs are shown in Figure 5.6. As before, the preprocessing led the algorithm to somewhat higher activation levels, and it would appear that the subtleties of different time constants are being lost due to the noise. This is easily seen by observing the post stimulus undershoots; many of the estimated time series have no post-stimulus undershoot but rather decay directly to the base level.

It is interesting to consider how the preprocessing and noise may effect the parameters of the fitting model. Consider two runs, shown in Figure 5.7, out of the previous eleven. Run 1 has higher peaks, in spite of the fact that both signals are based on the same underlying signal. There also appears to be more drift than the 20 measurements per knot could fit, which explains run 1's prolonged increase at 170 seconds in. Interestingly, run 1 and run 2 have different aspects of the signal that match better. Run 2 had a much better match to the peaks, when compared to the

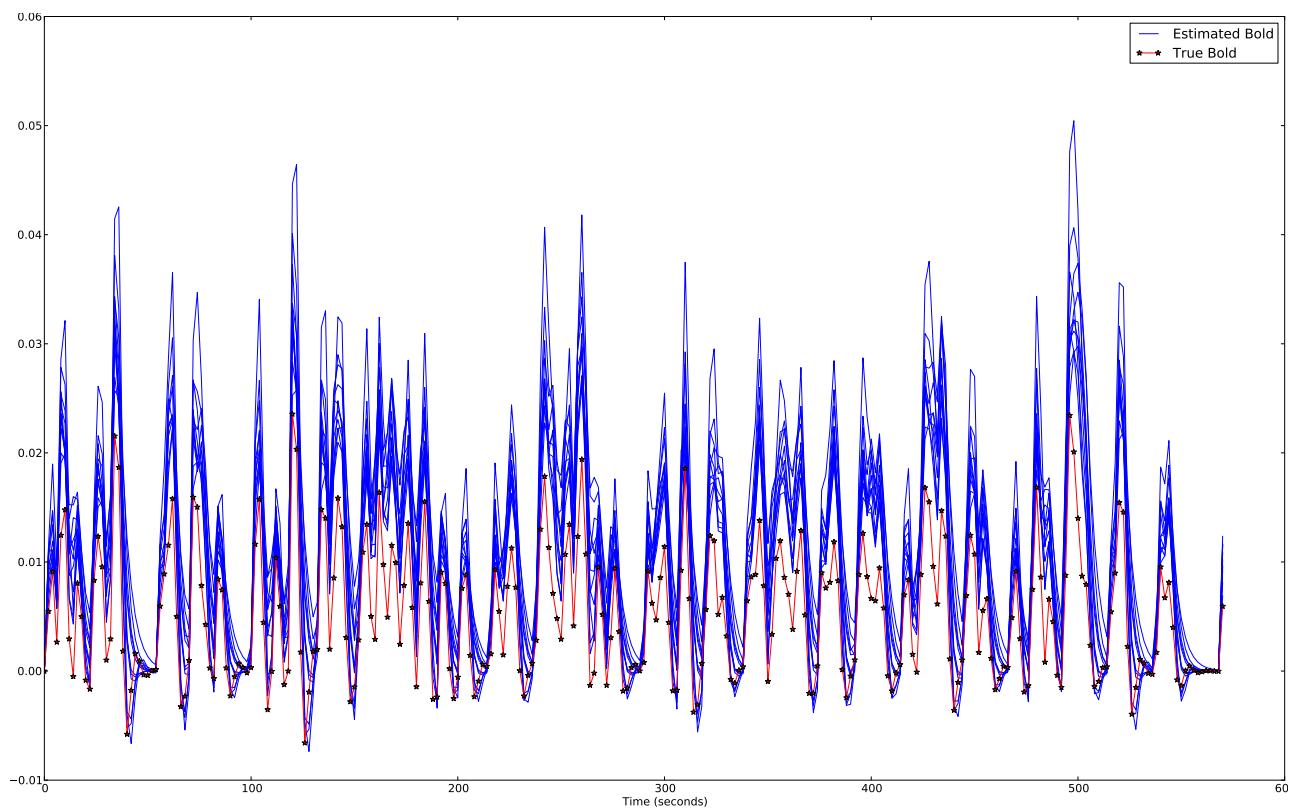


Figure 5.6: A comparison of the fitted signals for the high noise case.

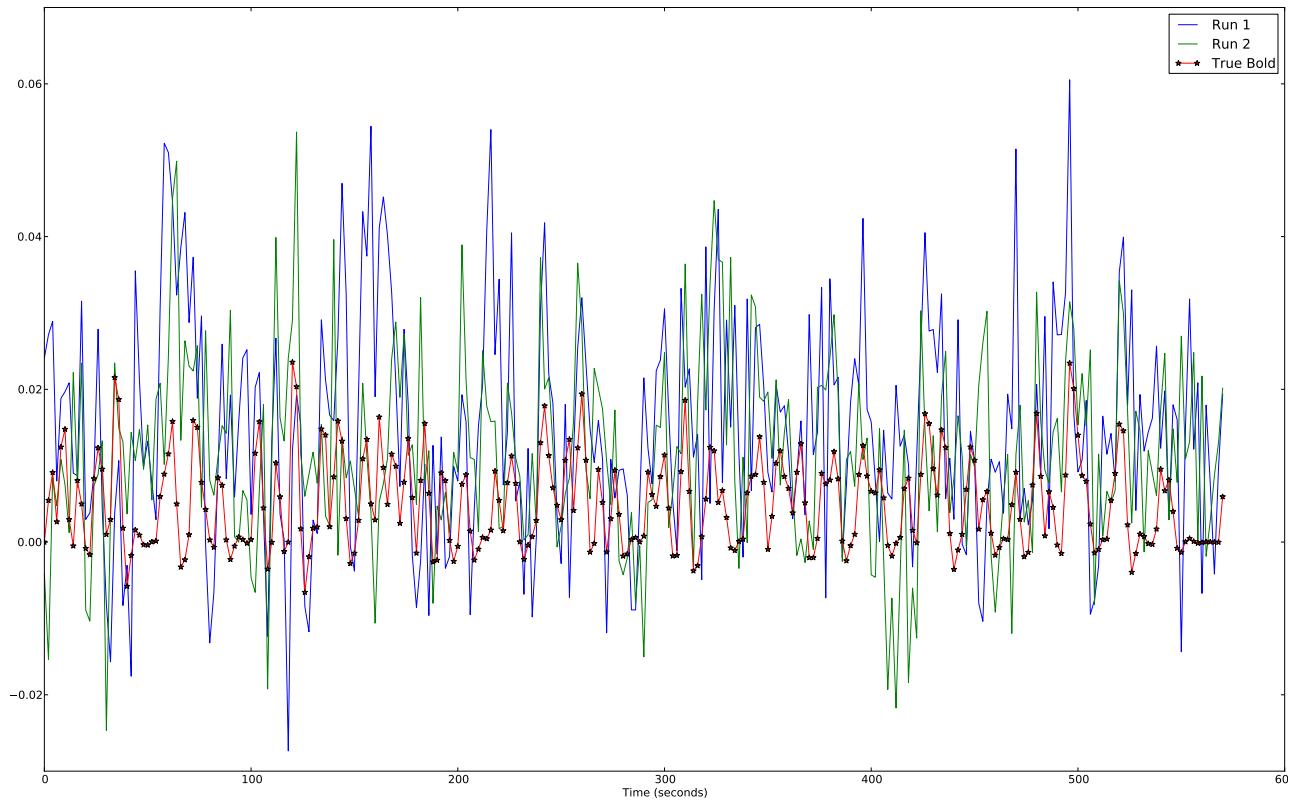


Figure 5.7: Two particular preprocessed noise realizations for the high noise case.

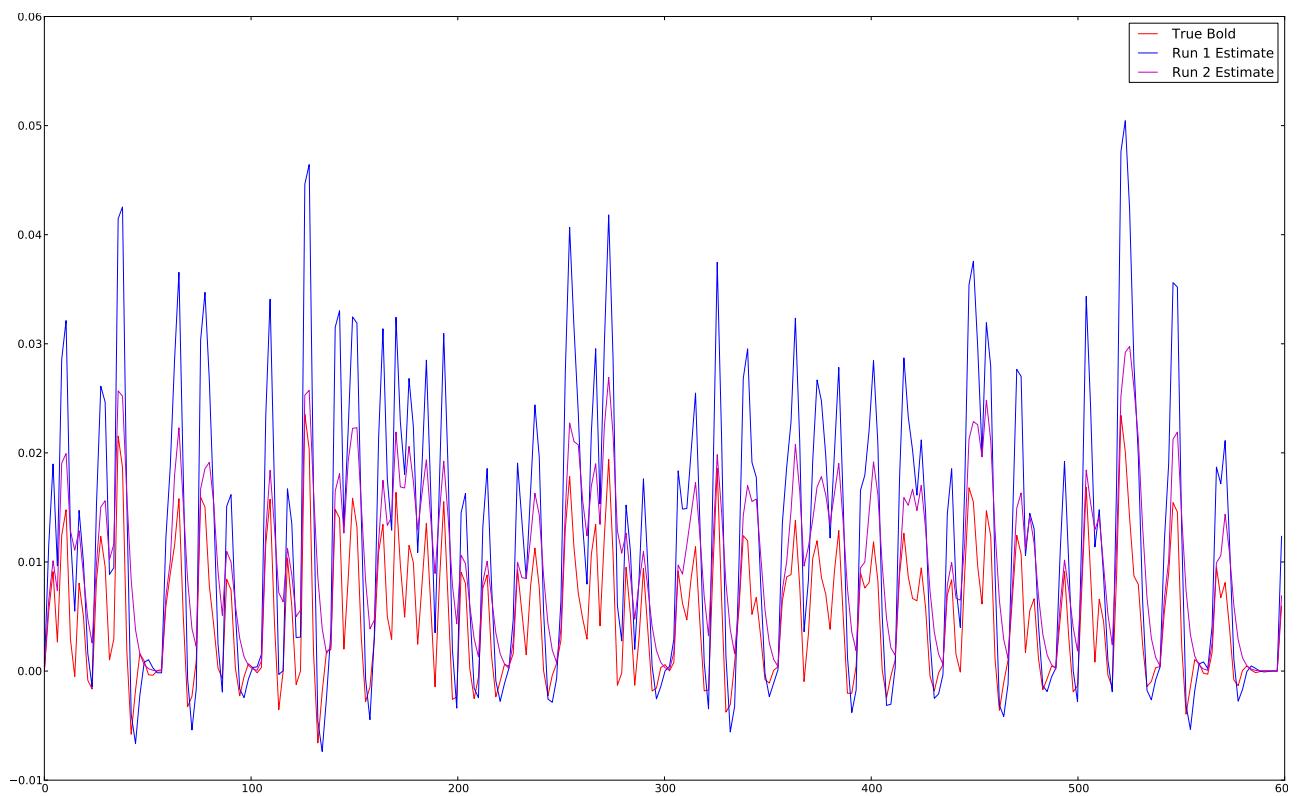


Figure 5.8: The results for the noise realizations shown in [Figure 5.7](#).

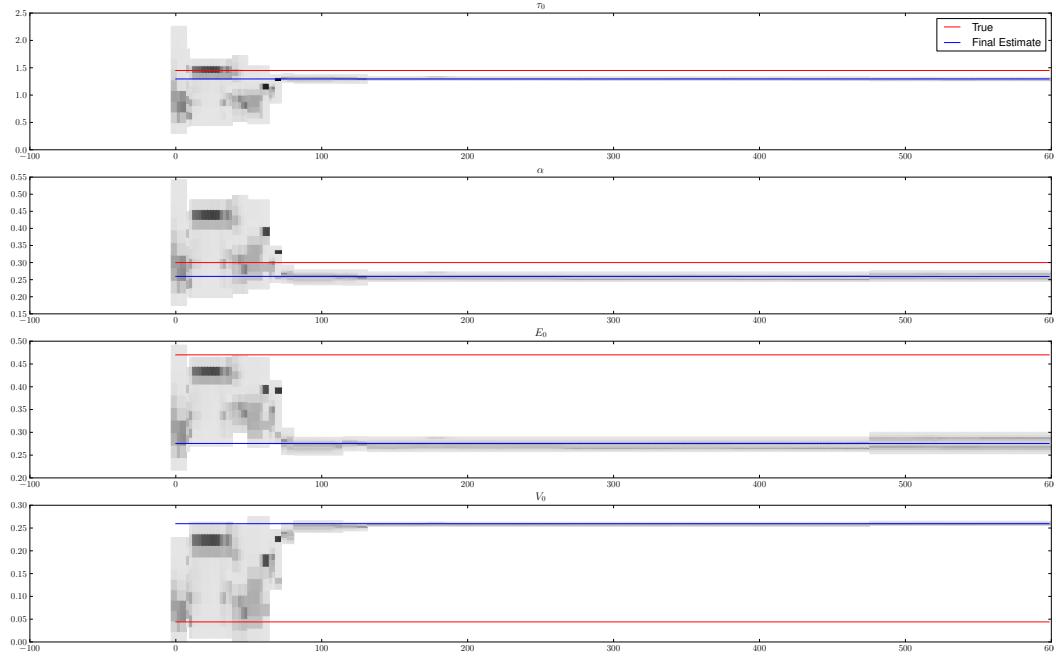
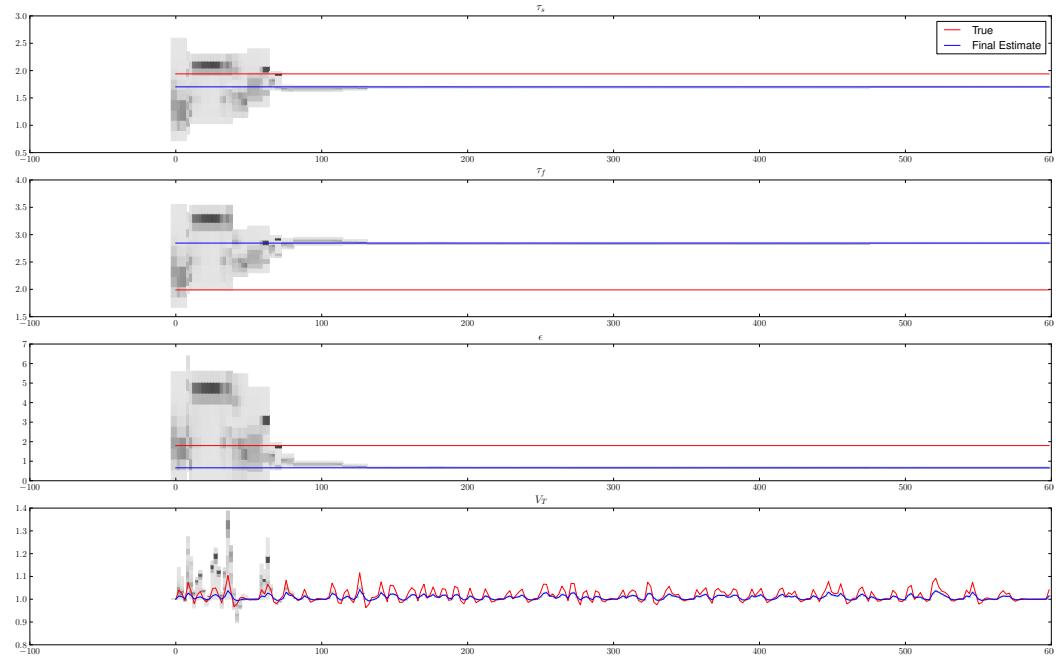
τ_0	α	E_0	V_0	τ_s	τ_f	ϵ	$\sum \tau$	\sqrt{MSR}	\sqrt{MSE}
1.45	.3	.47	.044	1.94	1.99	1.8	5.38		
1.1900	0.2349	0.4223	0.128	1.0147	2.4779	1.1168	4.6826	0.01406	0.01573
0.9721	0.2190	0.3051	0.061	0.5780	1.9960	3.4613	3.5461	0.01373	0.01378
1.5795	0.1415	0.3380	0.1089	0.5843	2.1247	1.7834	4.2885	0.01275	0.01577
1.1094	0.2374	0.5349	0.0351	1.2186	3.0736	2.3504	5.4016	0.01673	0.01154
1.1071	0.2753	0.3365	0.0316	1.5057	2.6518	4.1910	5.2646	0.01370	0.01222
0.5803	0.4793	0.4135	0.1189	0.9756	3.6902	1.0008	5.2461	0.01150	0.01316
1.2952	0.2596	0.2756	0.2595	1.7026	2.8458	0.6617	5.8436	0.01555	0.01790
1.5185	0.2199	0.2835	0.0742	0.8882	3.0771	1.7393	5.4838	0.01205	0.01246
0.6874	0.3283	0.3979	0.1561	1.0778	3.1158	0.6643	4.8810	0.01510	0.01258
1.0170	0.285	0.3474	0.0567	1.5877	2.6516	2.2852	5.2563	0.01249	0.01343
0.9925	0.298	0.3221	0.2094	0.4276	2.2108	1.0167	3.6308	0.01217	0.01506
1.0954	0.2708	0.3615	0.1126	1.0510	2.7196	1.8428	4.8659	0.01362	0.01397

Table 5.3: Estimated Parameters on 10 different runs with high noise. First row is the true parameters. Note also that the red row is Run 1 and the blue row is Run 2, as named in [Figure 5.7](#) and [Figure 5.8](#)

true signal, yet run 1 matched the post-stimulus undershoot better. Its worth noting however, that the post-stimulus undershoots here are much shorter than the observed prolonged post-stimulus undershoot discussed in [section 1.5](#). Despite the rather significant amount of noise, the ultimate results are actually rather good. [Table 5.3](#) shows the square-root mean-squared-error for all eleven runs, and highlights the two runs analyzed in [Figure 5.9](#) and [Figure 5.10](#).

There are a number of interesting convergence properties of the particle filter when more noise is present, as both [Figure 5.9](#) and [Figure 5.10](#) show. The particle filter seems to converge significantly faster; as points tend to be further out on the weighting function. This also causes significantly more resampling which is the explanation for the perceived jumps in resolution that occur from time to time. Here it is clear that the mode and the mean will not be substantially different. Because of the increased noise, its likely that the weighting function is not sufficiently wide to account for the measurement noise.

The parameters arrived at for all ten filter runs are shown in [Table 5.3](#). Clearly the additional noise have resulted in much more sporadic results. This is often the result when the particle filter converges too fast, a result of the weighting functions' variance being smaller than the measurement noise (.005 vs. .01). The error certainly suffers due to this effect.

(a) τ_0, α, E_0, V_0 , Run 1(b) $\tau_s, \tau_f, \epsilon, V$, Run 1

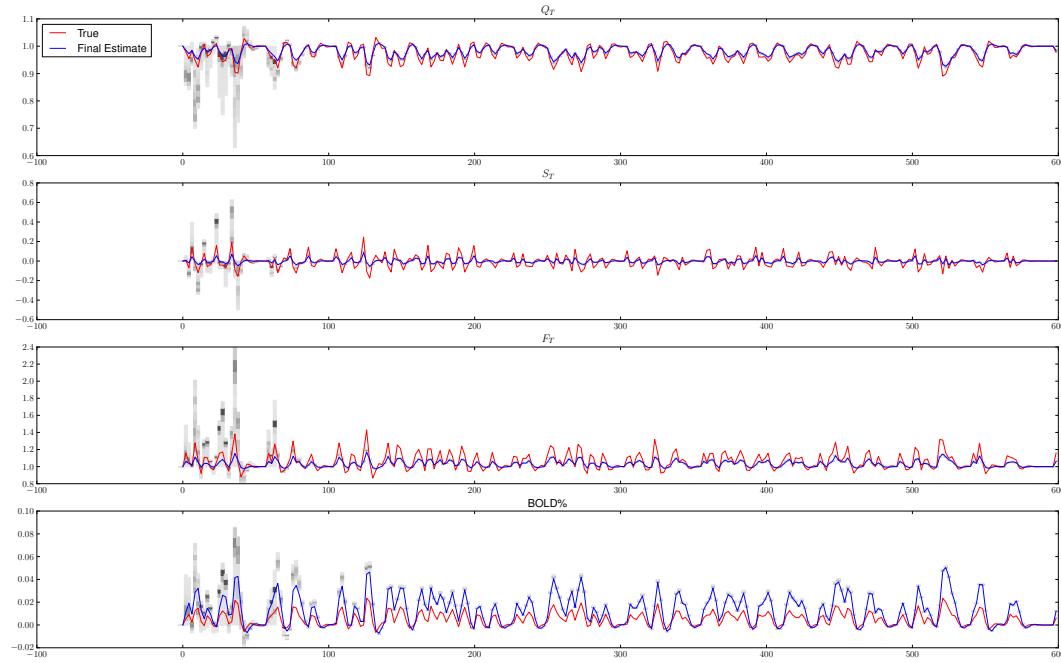
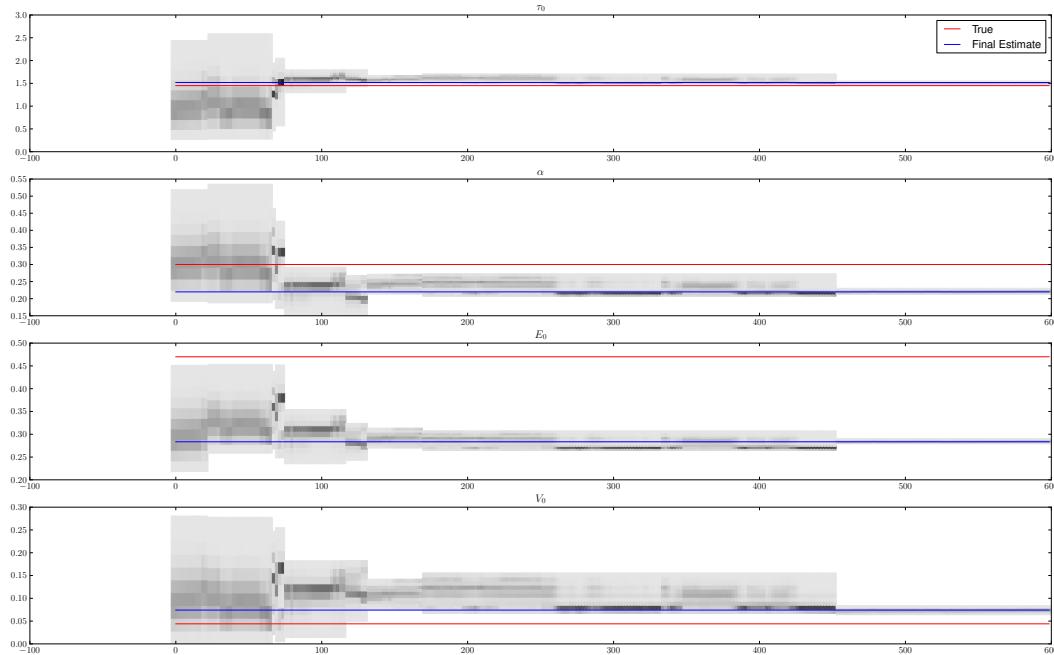
(c) $Q, S, F, BOLD$, Run 1

Figure 5.9: Converging histogram for parameters during run 1, as in [Figure 5.7](#).

(a) τ_0, α, E_0, V_0 , Run 2

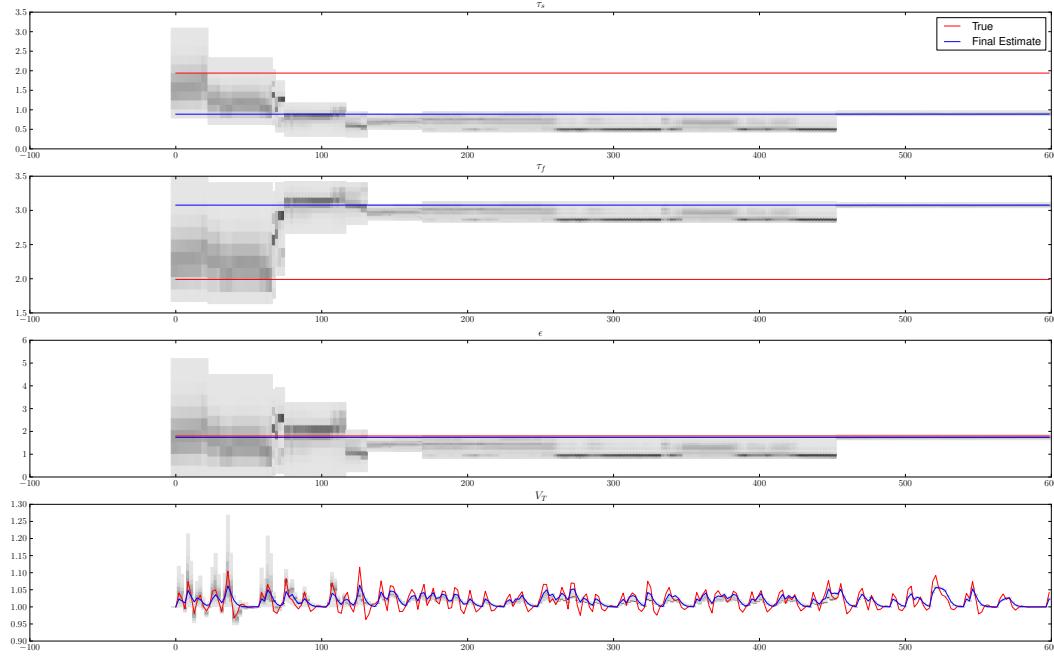
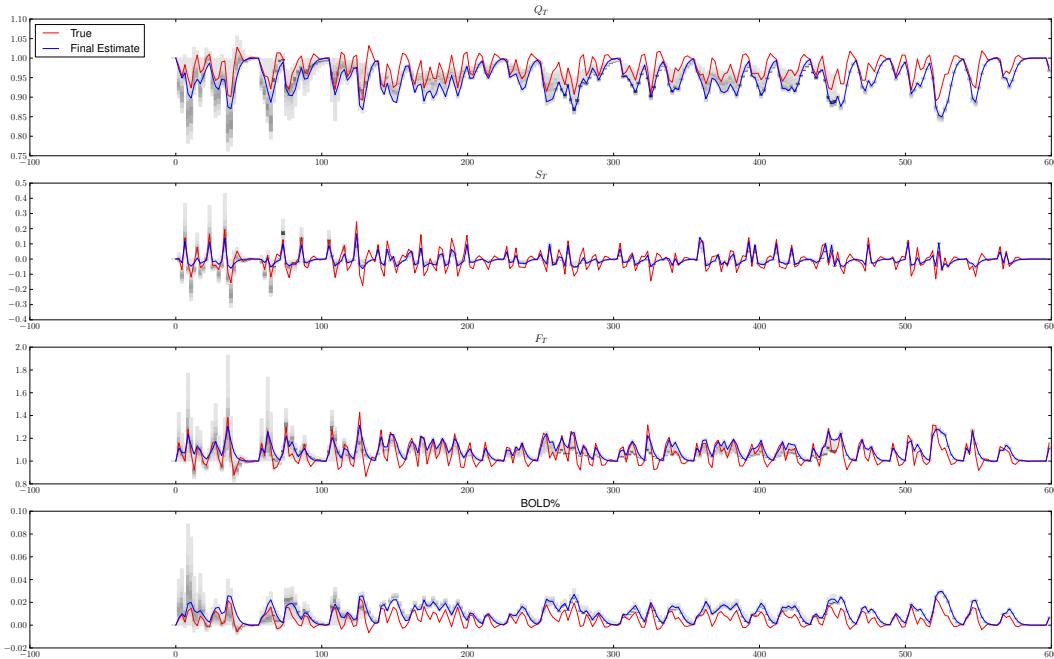
(b) $\tau_s, \tau_f, \epsilon, V$, Run 2(c) $Q, S, F, BOLD$, Run 2

Figure 5.10: Converging histogram for parameters during run 2, as in Figure 5.7.

5.1.3 Pure-Noise, low magnitude

The two final single-voxel tests force the particle filter to attempt to learn a noise-only time series. In the first test the noise used will be the same as that from the subsection 5.1.2, $\sigma_x = .01$, $\sigma_y = .005$. The parameters will be set the same as well, but the stimulus/input neuronal efficiency (ϵ) will be set to 0. In effect it is a region of the brain for which no visual stimulus induces activity. The question then is how the particle filter will respond to signal that does not correlate with the input, and how the output may be differentiated from a time series that does. The time series' are shown in Figure 5.11, the preprocessed versions are shown in Figure 5.12. The line fits are shown in Figure 5.13.

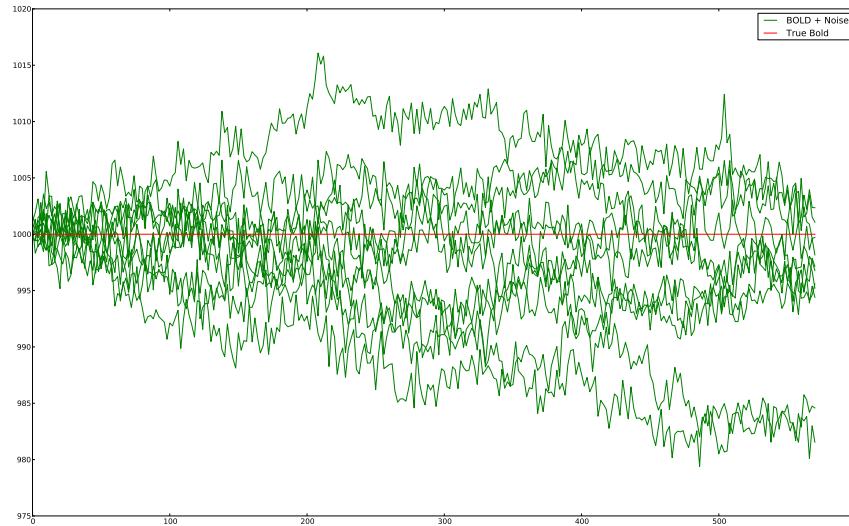


Figure 5.11: Time-series lacking any real signal. With, $\sigma_x = .01$, $\sigma_y = .005$

τ_0	α	E_0	V_0	τ_s	τ_f	ϵ	\sqrt{MSR}
1.0324	0.33211	0.34058	0.03012	1.40665	2.52079	0.5311	0.00167
0.98189	0.33047	0.3386	0.03014	1.45707	2.47232	0.45049	0.00159
1.0429	0.33224	0.34124	0.02946	1.4618	2.49245	0.43012	0.00165
1.02054	0.3321	0.33484	0.02586	1.45848	2.48741	0.4193	0.00151
1.0565	0.33405	0.33758	0.02791	1.43784	2.52545	0.47517	0.00152
1.01867	0.33528	0.33918	0.02782	1.48345	2.49605	0.44209	0.00156
1.051	0.33038	0.33837	0.02985	1.47651	2.48621	0.42719	0.00159
1.00281	0.32929	0.33988	0.0298	1.43519	2.49256	0.48899	0.00164
1.00893	0.33273	0.33982	0.0289	1.42903	2.49754	0.45688	0.00168
1.01289	0.33275	0.3376	0.02997	1.41188	2.49881	0.50628	0.00183
1.10247	0.33371	0.3419	0.02939	1.43774	2.53384	0.44079	0.00195
1.03009	0.33228	0.33905	0.02902	1.44506	2.50031	0.46076	0.00165

Table 5.4: Estimated Parameters on 11 different runs with low noise and no signal present.

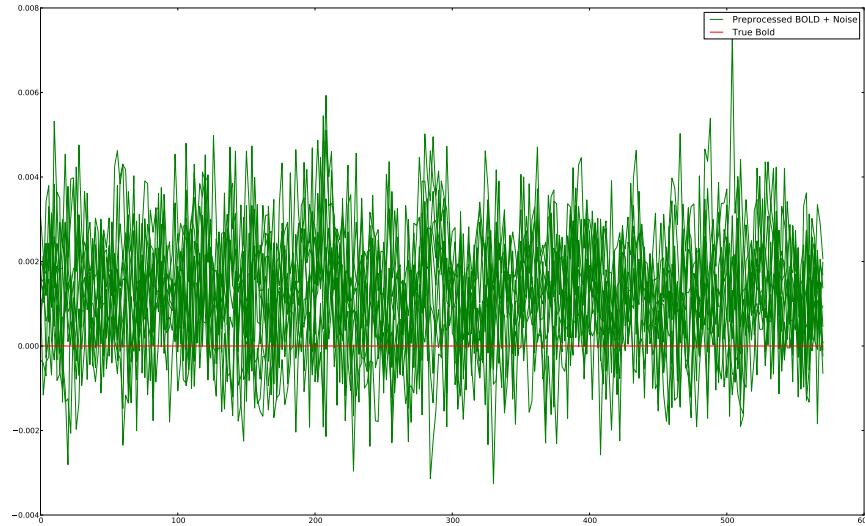


Figure 5.12: A comparison of the preprocessed signals for the signal-free case. ($\sigma_y = .01, \sigma_x = .005$)

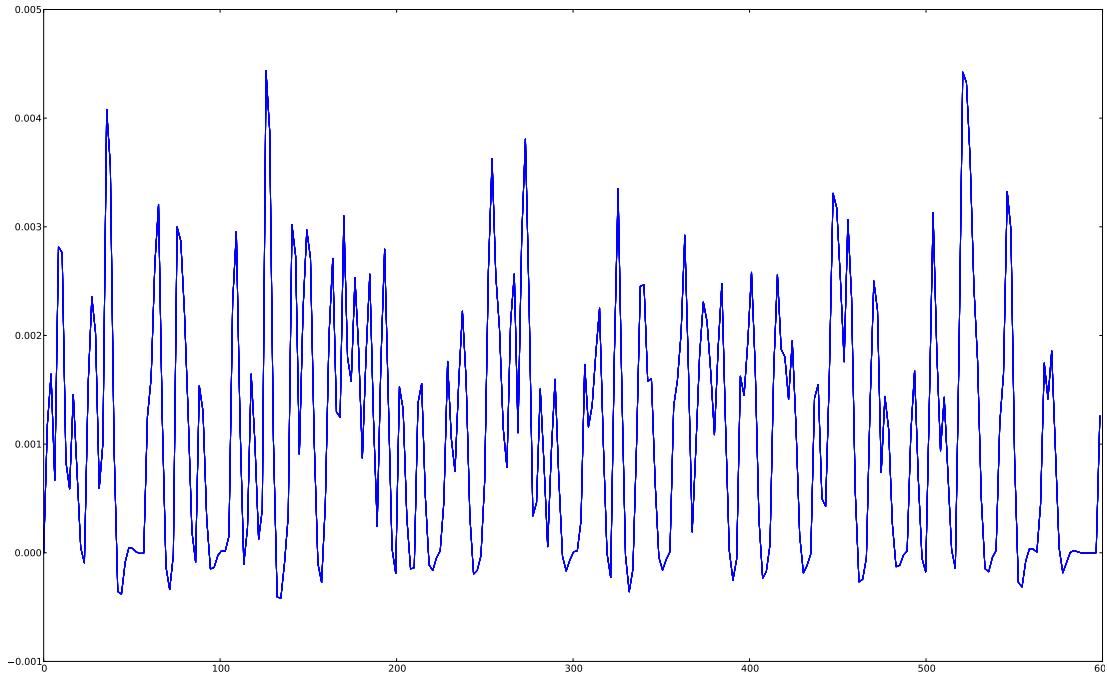


Figure 5.13: Fits to the non-active, low noise signal. Note that the line is thick because all the fits overlap. This is all 11 fitted lines.

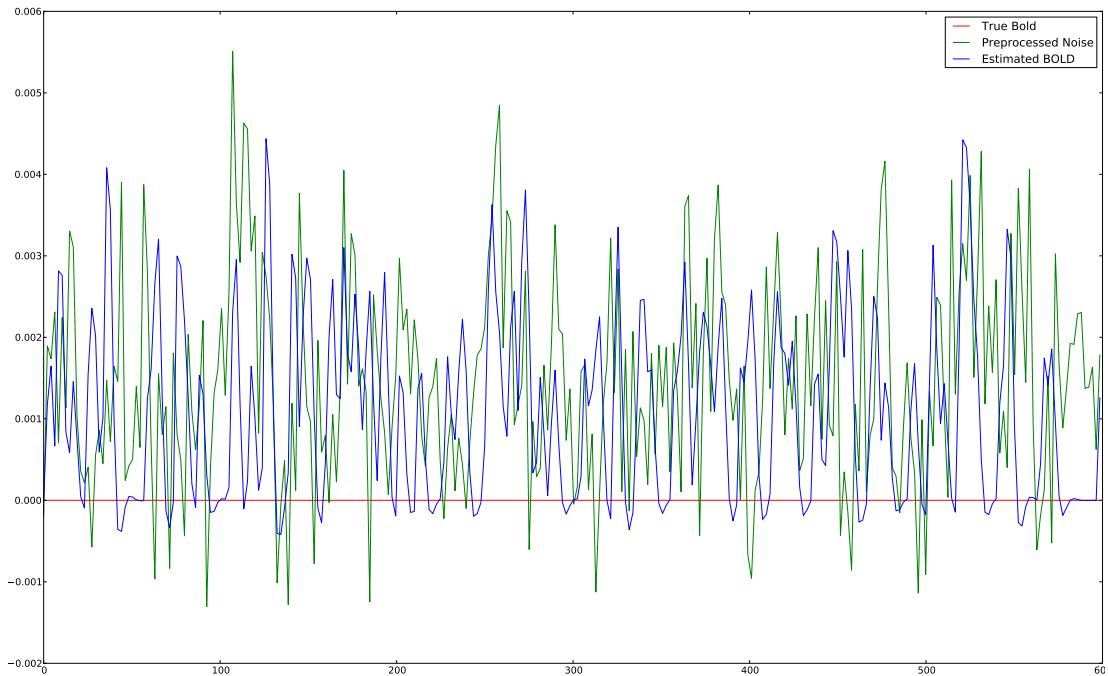
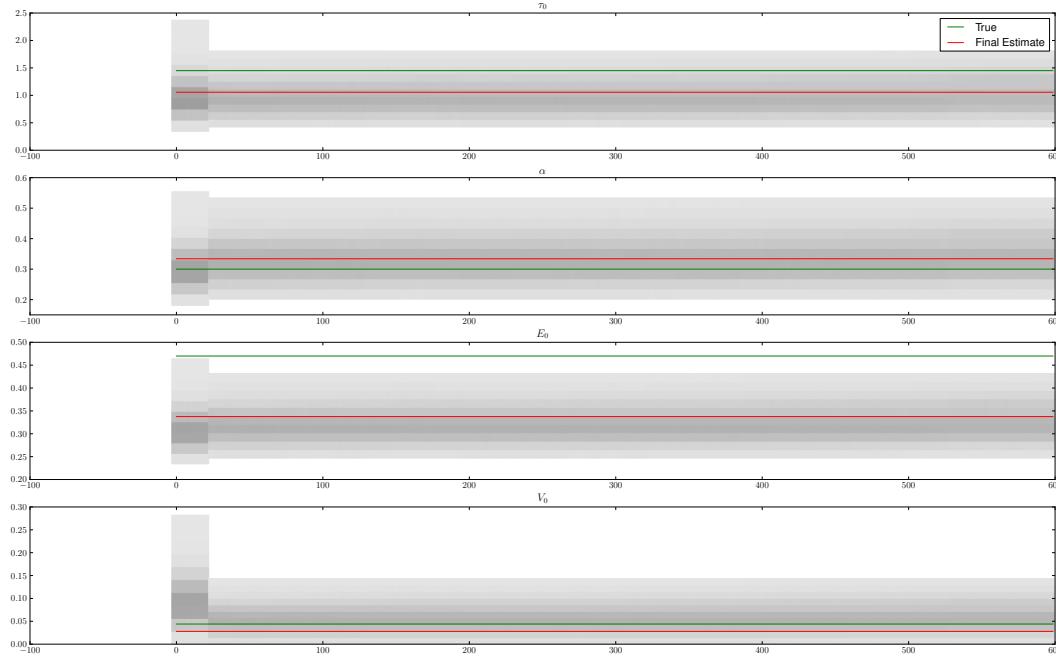
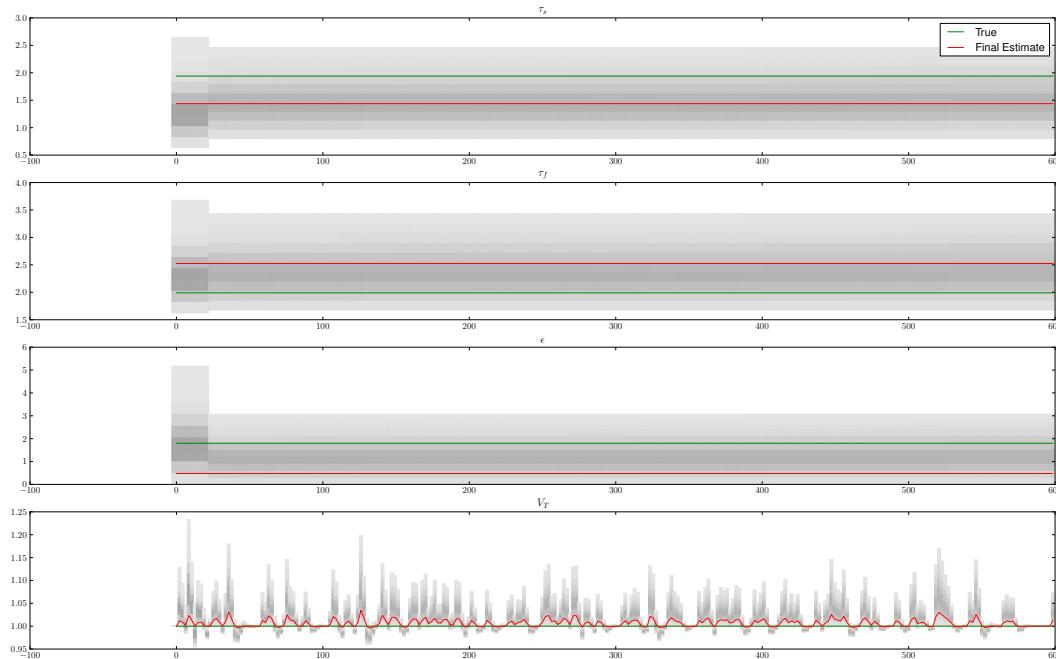


Figure 5.14: Fit from a single particle filter run, with the noise input.

(a) τ_0, α, E_0, V_0 (b) $\tau_s, \tau_f, \epsilon, V$

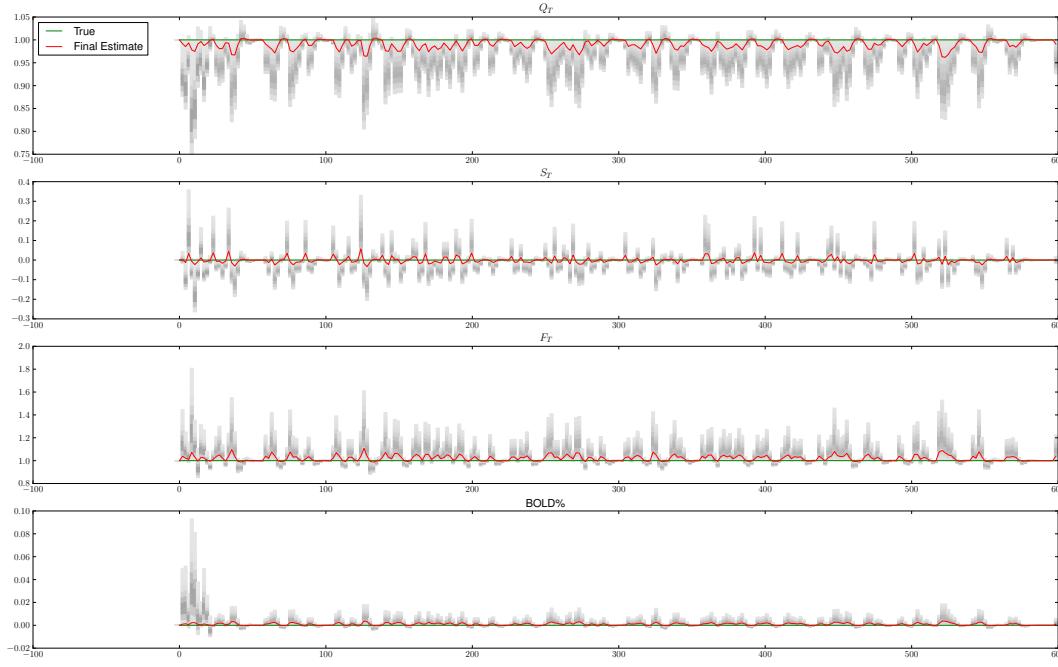
(c) $Q, S, F, BOLD$

Figure 5.15: Converging histogram for parameters when the signal consists purely of low level noise. Same run as [Figure 5.14](#)

The data shows that the parameters have not taken any visible turns into unrealistic territory; individual parameters will not be a distinguishing factor in determining the validity of the model. Additionally the residual also appears to be lower than the residual of the low noise simulation from earlier. The precision with which the convergence occurred is actually the best seen in any simulation. The reason for the low error is that the overall signal is significantly smaller than any other simulation in this thesis. The peaks never even 1% difference, with most staying even below .5% ([Figure 5.12](#)). Because the stayed well within the range of .005, which is the standard deviation of the weighting function the particles did not ever get substantially different weights. Thus, the reason the fits are in such agreement is that no particles gained increased weight, and so no convergence occurred at all, as further shown by [Figure 5.15](#). This is good evidence that a per time-series weighting function could benefit even out the convergence rate; and thus allow better determination between signals like this and low noise case from earlier. Thus we have now seen how the particle filter responds when the signal is completely random and low level, however there is also some possibility that the signal will be random, yet active. Thus one more test is necessary; time series with no underlying signal (as far as responding stimuli) but with enough noise to actually be considered a signal.

5.1.4 Pure Noise, High Magnitude

To determine how the particle filter will respond to active, yet unrelated portions of the brain, in this section random data will again be used, but with even higher standard deviations. To simulate this case another pure noise signal was generated using a σ_x of .1 and a σ_y of .05.

As before, the convergence all seems to follow a similar path, leaving almost no variance in the estimated time series (Figure 5.16). Now, however the peaks are comparable to peaks in a real timeseries. Interestingly the algorithm suffered from almost constant particle deprivation, meaning that the heuristic for rescuing the particle filter from particle deprivation, discussed in [section 4.3](#) in fact was working against the proper course of action. The proper course of action in this case would be for the particle filter to fail, since no particle can really properly estimate a random sequence. When this mechanism was removed, all 11 runs stopped due to particle deprivation (all weights hit zero). The problem with allowing particle deprivation to occur is that it can *rarely* occur if the prior happens to not be dense enough in the area of the solution. Thus, there is an issue of false positives vs. false negatives. A middle ground, such as allowing deprivation to occur if the recovery mechanism has already triggered once may be an acceptable compromise, however it would be difficult to strike the right balance. Instead, for the purposes here, I depend on other methods, described later, evaluate if the results were reasonable or not.

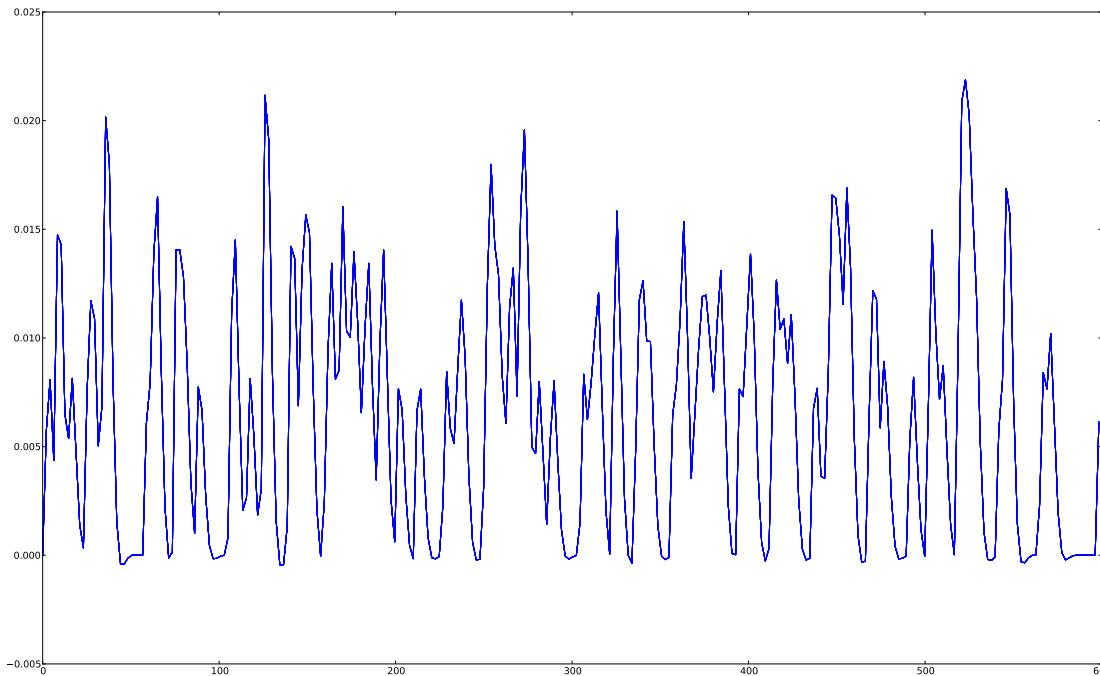


Figure 5.16: Fits to the non-active, high noise signal. Note that the line is thick because all the fits overlap. This is all 11 fitted lines.

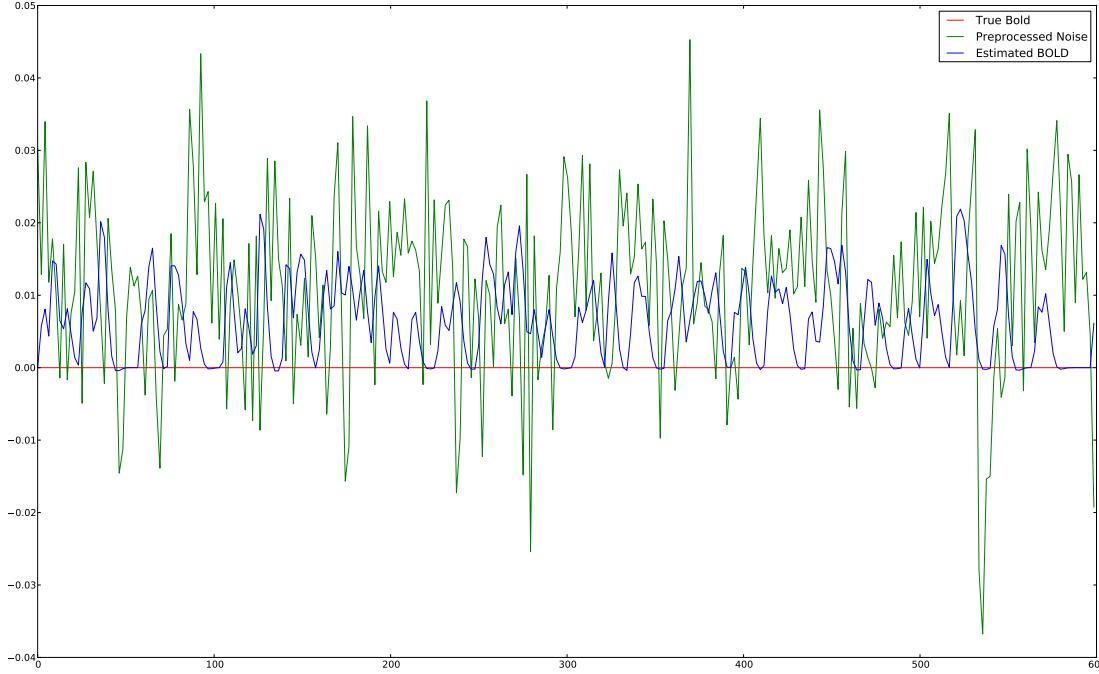


Figure 5.17: Fit from a single particle filter run, with the noise input.

From Figure 5.17 it is clear that, despite the fact that the input is completely detached from the signal, a non-zero response by the particle filter does lead to decreased residual. This is primarily due to the preprocessing which forces the signal to have a non-zero mean. While this scheme could create a type of artificial activation for the particle filter, in reality the residual will still be less than if the signal were left with zero mean, and the particle filter stayed flat. At best the particle filter will over-smooth, estimating a centerline to noisy signal. In such a case the residual will be the same as the alternative, although the mutual information will be different.

5.1.5 Single Voxel Review

The first step in determining the validity of a model is to provide some order of quality to rate the results by. Unfortunately, because of the variability in the signal levels the raw \sqrt{MSR} cannot perform this task. As demonstrated by subsection 5.1.3, a low residual does not necessarily indicate a good fit. Therefore, instead I normalized the signal based on a value proportional to the magnitude of the signal. Considering the tendency of FMRI noise to have large unexplainable peaks and troughs, rather than using MAX and MIN values to estimate the scale of the signal, I used a robust estimator of scale; the Median-Absolute-Deviation (MAD). This is an estimator of the standard deviation, and thus a good estimator of the scale of the input signal. The normalized residual values are shown in Table 5.5. A second potential method of gauging performance is mutual information. Mutual information is a method of measuring the interdependence of two

	Signal				No Signal			
	Low Noise		High Noise		$\sigma_y = .001, \sigma_x = .0005$		$\sigma_y = .01, \sigma_x = .005$	
	M.I.	N. Res.	M.I.	N. Res.	M.I.	N. Res.	M.I.	N. Res.
1	0.86687	0.47801	0.09077	1.03894	0.06326	1.29501	0.03024	1.33641
2	0.93975	0.53177	0.13767	0.95165	-0.01075	1.30175	-0.02677	1.33667
3	0.82382	0.5458	0.13505	0.99539	0.02345	1.26287	-0.0111	1.15957
4	0.94661	0.49824	0.04341	1.16129	-0.00906	1.43196	0.00147	1.09988
5	0.94281	0.46805	0.13718	1.03972	0.00663	1.25664	-0.00204	1.20107
6	0.92539	0.459	0.12337	1.00214	-0.00816	1.2708	0.01775	1.04589
7	0.98892	0.46096	0.15381	1.08847	0.02664	1.15441	0.03163	1.20543
8	0.98796	0.51838	0.11325	1.05962	0.03285	1.27456	0.01951	1.1225
9	0.8804	0.5253	0.09669	1.0157	0.01628	1.32024	0.01039	1.08637
10	0.88721	0.49211	0.18339	1.18996	0.00407	1.34456	0.00508	1.22135
11	0.96644	0.49092	0.10949	0.95368	0.03323	1.32522	-0.01284	1.11737
mean	0.92329	0.49714	0.12037	1.04514	0.01622	1.29437	0.00576	1.17568
min	0.82382	0.459	0.04341	0.95165	-0.01075	1.15441	-0.02677	1.04589
max	0.98892	0.5458	0.18339	1.18996	0.06326	1.43196	0.03163	1.33667

Table 5.5: Mutual Information and the normalized \sqrt{MSE} , for signal/noise configurations.

random variables. If two signals are truly independent, then the mutual information will be zero. Although ideally suited to Bernoulli random variables where there are a finite number of options, by using histograms it is possible to derive a joint distribution of two signals. The algorithm for mutual information is based on that joint distribution:

$$\sum_{x,y} p(x,y) \log_2 \left(\frac{p(x,y)}{p(x)p(y)} \right) \quad (5.3)$$

Unfortunately the number of bins causing bias in the output, thus to correct for this, I subtracted the estimated bias:

$$\text{bias} = \frac{N_{bins}}{2N\log(2)} \quad (5.4)$$

where N is the number of samples and N_{bins} is the number of bins. For all the mutual information estimates in this work 6 bins are used for the marginal distribution of each signal. Additionally, throughout log base 2 will be used. This leads to 36 total bins in the joint, so the bias is:

$$\text{bias} = \frac{18}{N} \quad (5.5)$$

After calculating the empirical mutual information, I subtracted the bias from the estimated mutual information. Note that this can result in negative mutual information, which should not technically be possible; so any negative mutual information was taken as 0.

Comparing the results of subsection 5.1.2 and subsection 5.1.4, distinguishing between these cases with normalized residual alone is difficult. On the other hand the minimum mutual information in the high noise case was .09077, and the maximum mutual information in the no signal, high noise case is .03163. This is a decent break, and should provide a means to detect what would otherwise be a false positive by the normalized residual. That said, mutual information is in some ways more permissive than residual based metrics. For instance, the addition of an arbitrary constant to a random variable does not effect mutual information. For this reason, throughout the rest of the results, both values are reported.

There were two primary purposes of these tests. First given the nature of monte-carlo techniques it is important to ensure consistency of results. Although the parameter sets were somewhat inconsistent, the quality of the fit was actually quite consistent across eleven runs. The second purpose was of course to determine how the particle filter responded to different signal to noise ratios.

5.2 Multi-voxel Simulation

To test the usefulness of the particle filter for I used a modified version of the FSL tool POSSUM to generate an entire FMRI image from a parameter map. The parameter map was generated by taking an existing activation map and assigning discrete parameter sets to each region. The result was a four dimensional (length x width x height x parameter) image with spatially varying parameters. Possum was then modified to take a parameter map and generate activation levels depending on the parameters at that point. The patch for POSSUM will be made available. As an unfortunate side effect of not using Possum's original activation scale, I manually added 750 to the total level of simulated Possum images. This is because the BOLD % difference levels were in the range of 50 - 100% from the base, about 5 times as large as they should have been. Ultimately this has no effect on the quality of the simulation and should not effect the underlying parameters (other than perhaps ϵ and V_0). Also worth noting is that the noise level was set to an SNR of 20, due to changes in the program the true signal-to-noise ratio was much lower, as seen in the fit in Figure 5.18.

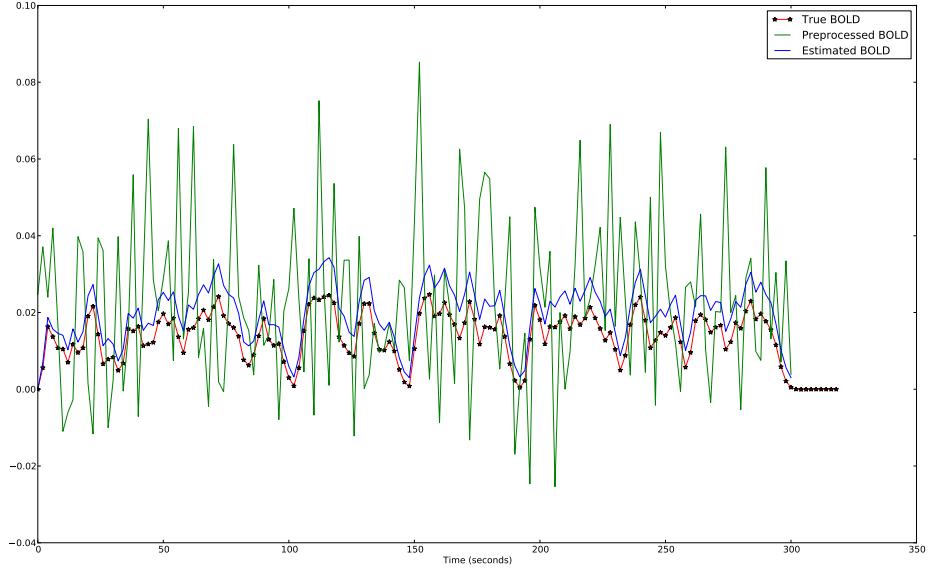


Figure 5.18: A single time-series fit in the simulated slice.

For each time-series in the simulated FMRI image, the final *static* parameters were saved into a parameter map. This parameter map may then be compared to the map used to generate the simulated data; additionally a new simulation using the calculated parameters may also be generated to test the difference in BOLD levels between the real parameters and the estimated ones. Since it is the parameters are far from orthogonal ([9]), its possible that two sets of parameters are functionally equivalent, despite large difference. This way, a quantitative difference between the two parameter sets was found.

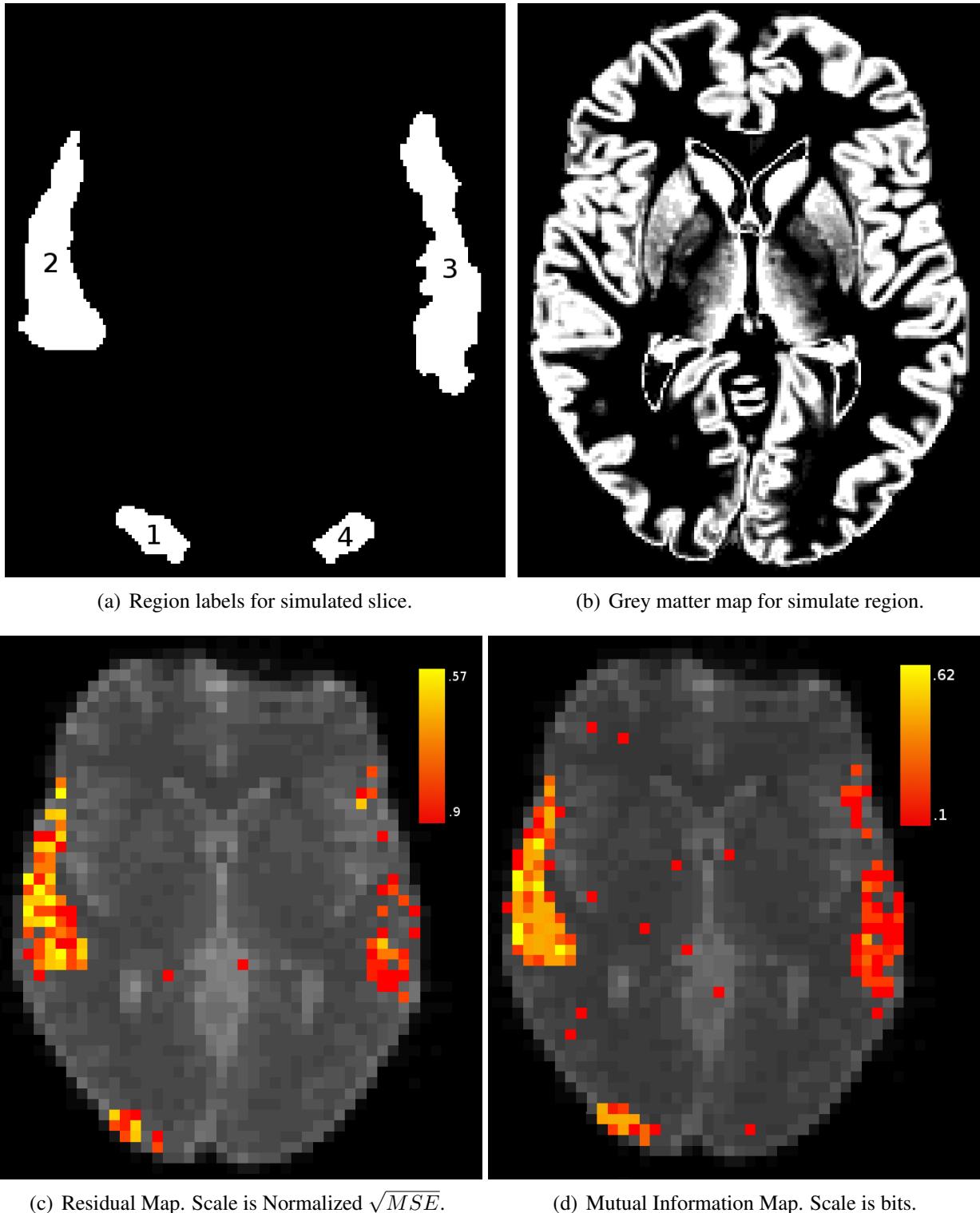


Figure 5.19: Comparison of activation with greymatter, parameter regions.

Region	τ_0	α	E_0	V_0	τ_s	τ_f	ϵ
1	1.454	.321	.369	.036	.994	2.774	1.348
2	1.151	.353	.380	.026	1.98	2.333	1.645
3	1.951	.317	.348	.027	1.657	3.719	.757
4	1.203	.310	.326	.036	2.168	2.272	.086

Table 5.6: Actual parameters for each regions in the simulated slice.

The regions are numbered according to [5.19\(a\)](#); the parameters for each region may be found in [Table 5.6](#).

Note that region 4 has a very low ϵ . For this reason, the only areas with significant estimates of the BOLD time series were 1,2 and 3. With such a low ϵ , region 4 was below the noise threshold. Notice that the regions 1 2 and 3 stick out in both the \sqrt{MSR} and the mutual information map, indicating that the particle filter was successful in matching those regions.

todo regenerate histograms

The histograms again demonstrate that a single point estimate of the parameters is elusive for the current model. However the data clearly show the power of the particle filter at identifying regions of activation; which usually performed with statistical parametric maps. The thresholds applied to this slice, both for mutual information and residuals is arbitrary, and needs further research. Tighter thresholds would certainly remove the false positives present in the images. Interestingly the false positives present in the Mutual Information map are different from those in the residual map. For this reason, it is possible that the best solution would make use of both metrics. Although at first glance it would appear that there are false negatives; this is not actually the case. POSSUM simulates different tissues, and white matter does not typically have a BOLD response. This is why there are holes in regions 2 and 3. These results certainly indicate that the particle filter is effective at regressing against a noisy signal. Therefore the next section moves forward with real FMRI data.

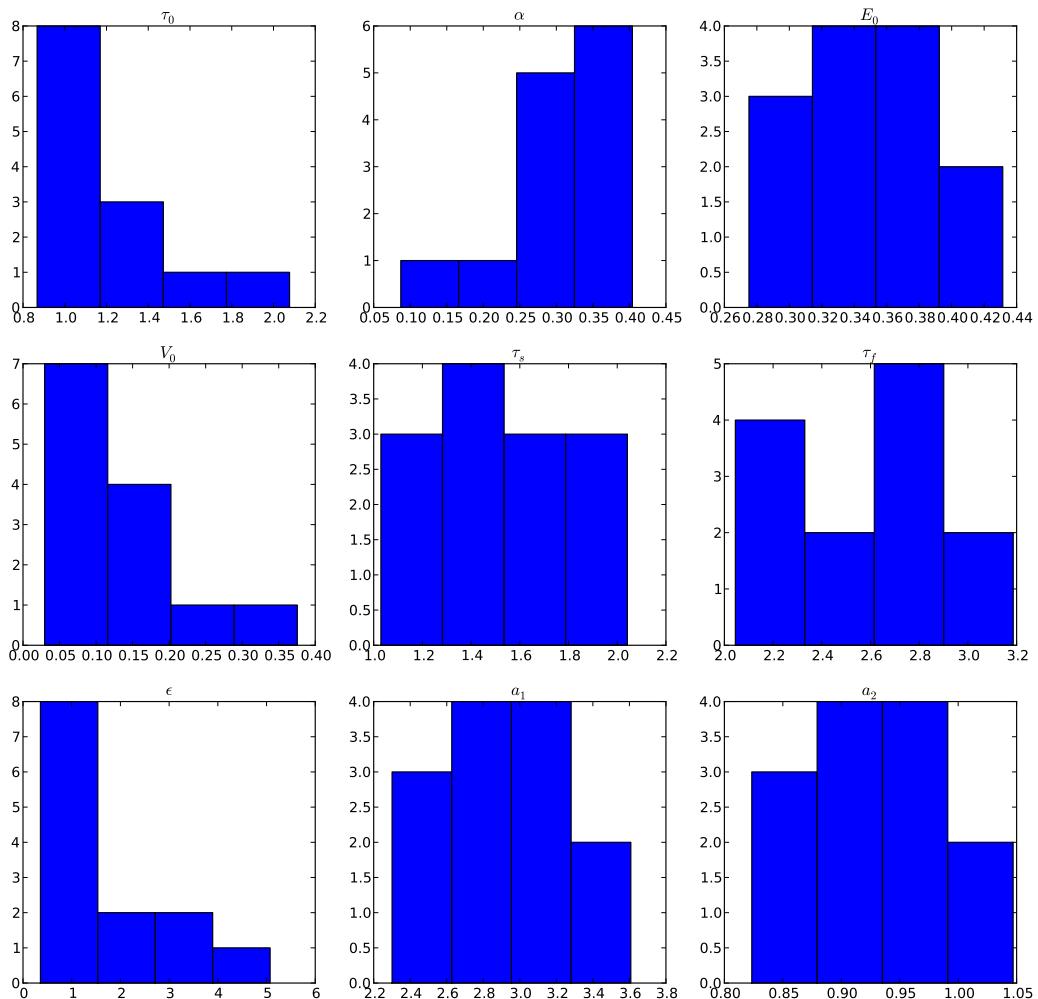


Figure 5.20: Histogram of estimated parameters in section 1 in voxels with mutual information greater than .14

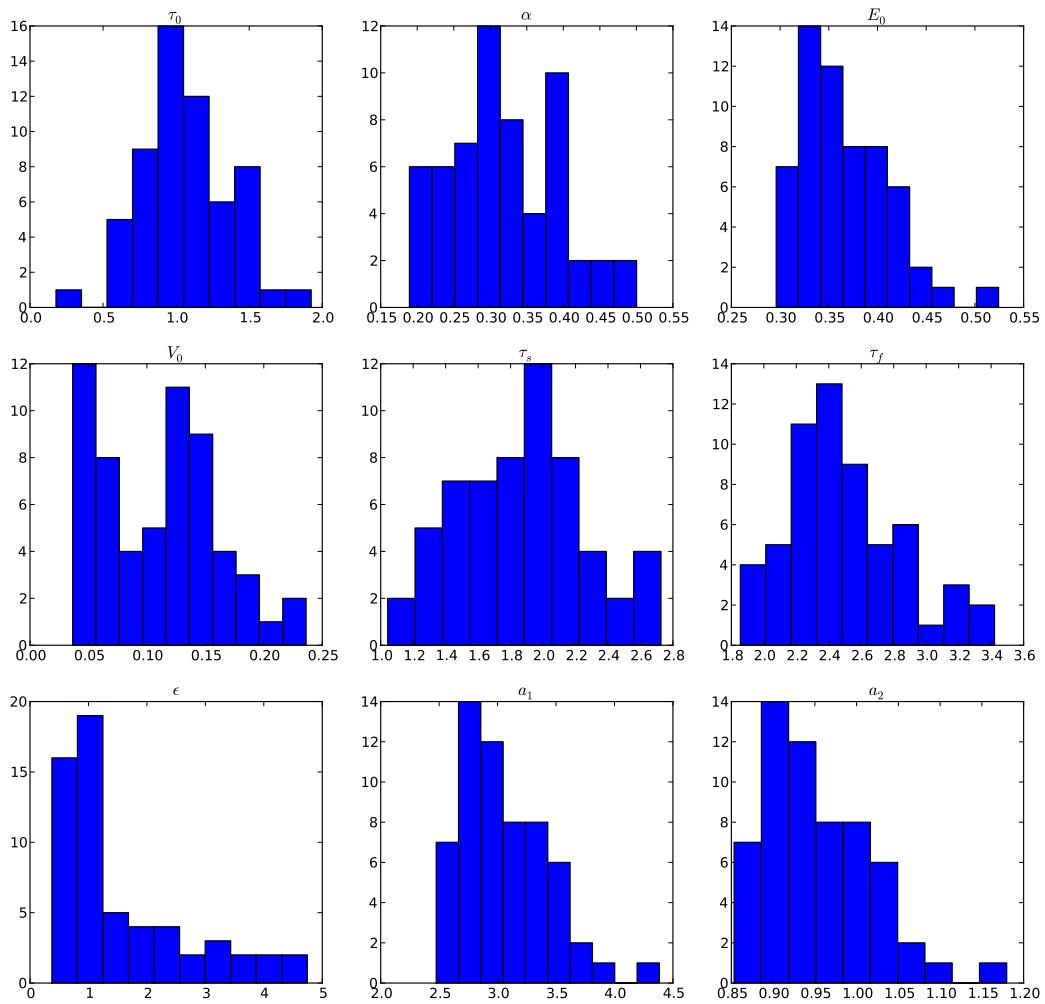


Figure 5.21: Histogram of estimated parameters in section 2 in voxels with mutual information greater than .14

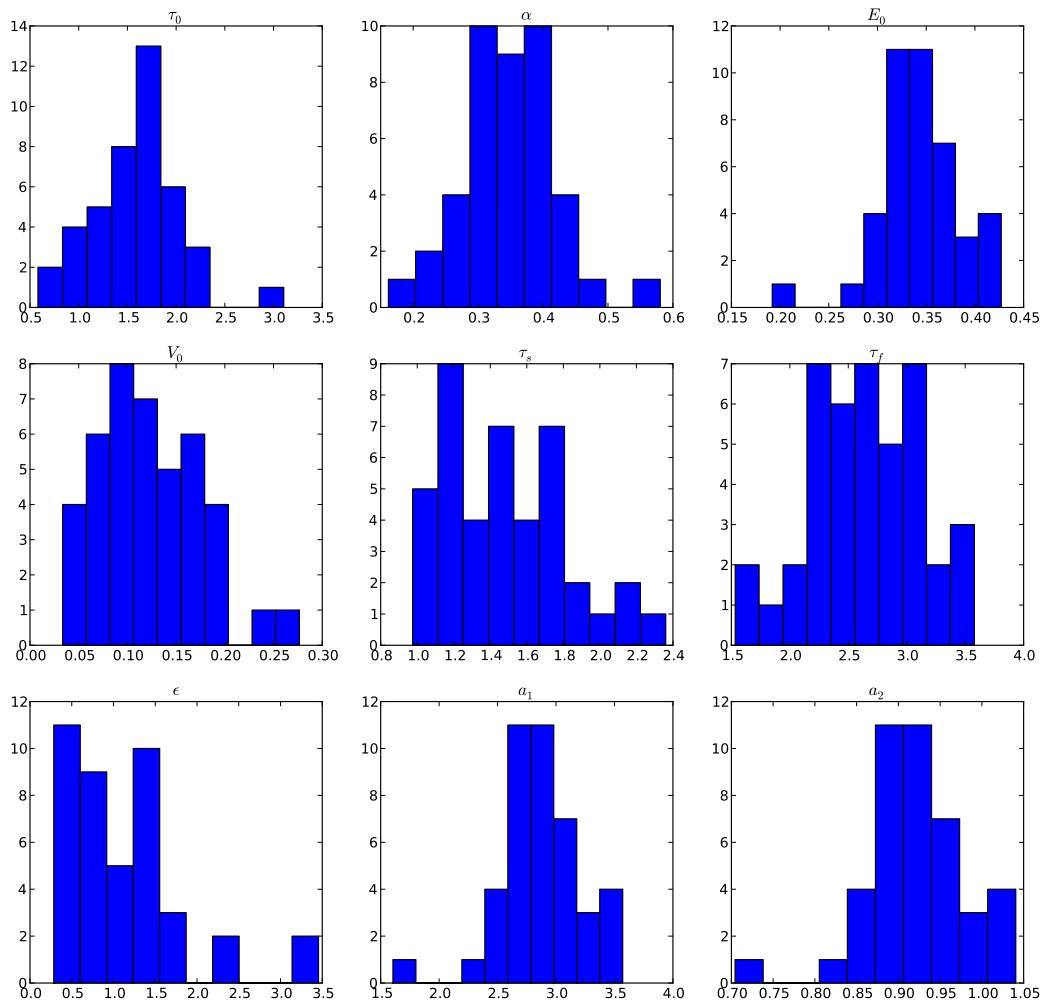


Figure 5.22: Histogram of estimated parameters in section 3 in voxels with mutual information greater than .14

Chapter 6

Real Data

Modeling the BOLD response is of course not of much use if it is only done in a single voxel. Although this algorithm will hopefully lead to more novel methods of analysis, the standard use for modeling the BOLD signal is to locate activation. Activation is defined as areas where the input seems to directly drive the BOLD response, as opposed to there being intermediate factors controlling it. Once areas where the BOLD model may be accurately estimated are found, integrating the model will allow for accurate estimation of the state between measurements, which could then be used for more advanced analysis, for instance of areas that are being driven by other brain regions. It all begins with localizing the first activation regions in the chain. Therefore this section compares the output of the particle filter with conventional SPM.

In reality, the purpose of statistical parametric mapping method is very different from the particle filter algorithm described in this work. SPM preprocesses the image by spatially smoothing the FMRI image (in this section SPM8 was used with an $8mm \times 8mm \times 8mm$ Gaussian kernel), whereas this is not done in the particle filter algorithm. Additionally, a spline was used to de-trend, rather than SPM8's high pass filter (with a cut off based on a globally estimated autocorrelation). Thus the preprocessing pipelines are different; but the output of SPM8 is also different. Whereas SPM outputs a t-statistic for each voxel, the output of the particle filter is a posterior probability distribution of the parameters at every voxel. To validate the quality of the particle filter results though it is necessary to compare the location and fit of the particle filter parameter mapping with SPM's T-score map.

6.1 Results

The results from T-values from SPM8 are shown in [Figure 6.1](#) (threshold of 4), and the results from the particle filter are shown in [Figure 6.2](#) and [Figure 6.3](#). Note that the scales for all three images are different, because the metrics are different. SPM measures using T-Tests to determine the likelihood of a false positive. [Figure 6.2](#) uses simple normalized residuals, meaning that lower

indicates less error. [Figure 6.3](#) measures in terms of the dependence between the measured signal and the estimated signal; thus higher indicates a better fit. The particle filter data shows a large number of false positives, however application of a threshold of .85 on the residual map removes these false positives. Similarly, in the mutual information map, the false positives may be eliminated by upping the threshold to .15. However, just because the results disagree with SPM does not necessarily mean they are false positives. SPM operates on smoothed data (8mm x 8mm x 8mm), so there are certainly active areas that have been missed because of the smoothing.

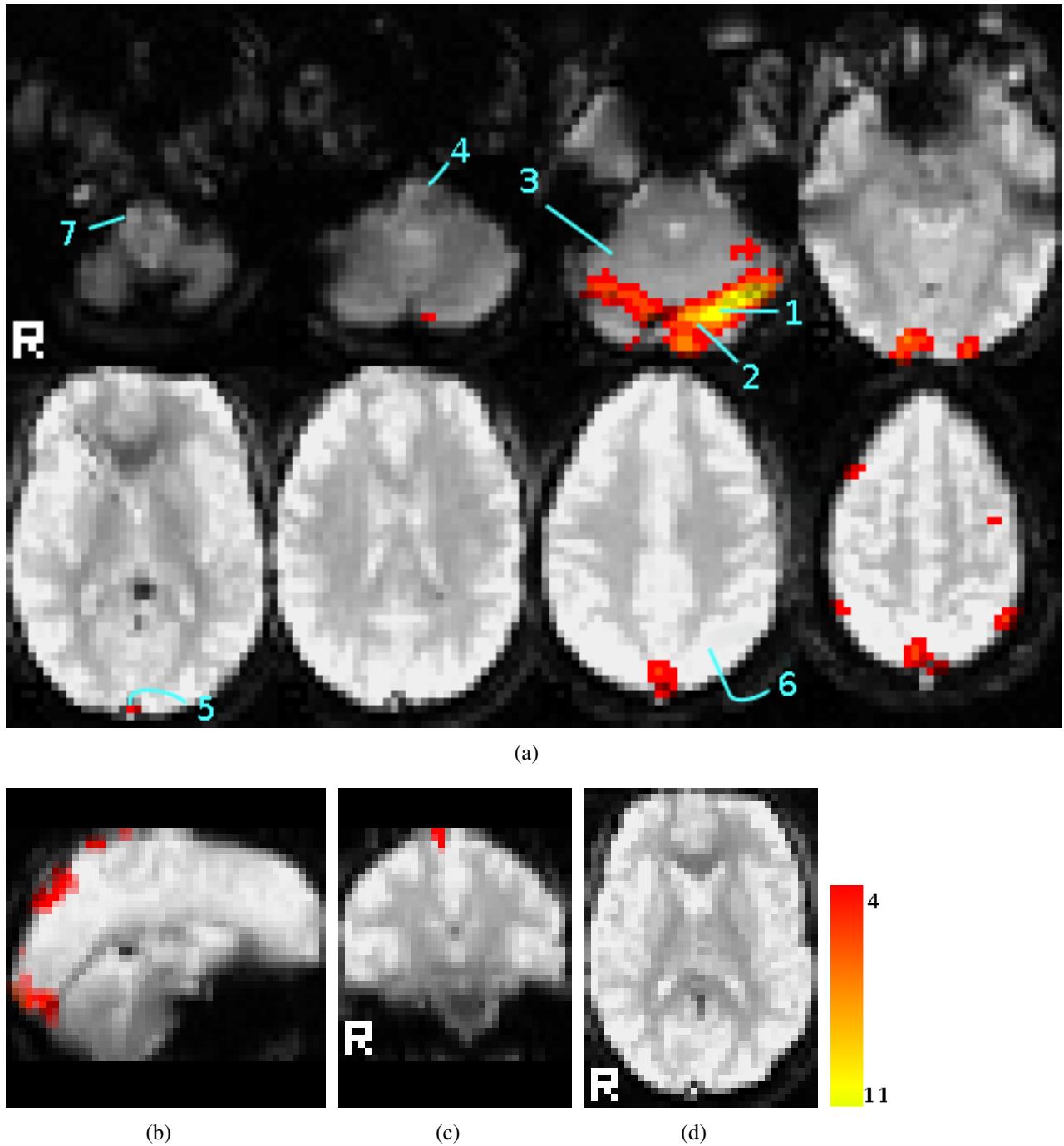


Figure 6.1: Sagittal, coronal and axial slices of SPM results (6.1(b) 6.1(c) 6.1(d)), as well as a series of axial slices, 6.1(a). Units of activation are in Student's T-scores. Higher indicates higher assurance that the signal cannot have occurred through noise alone.

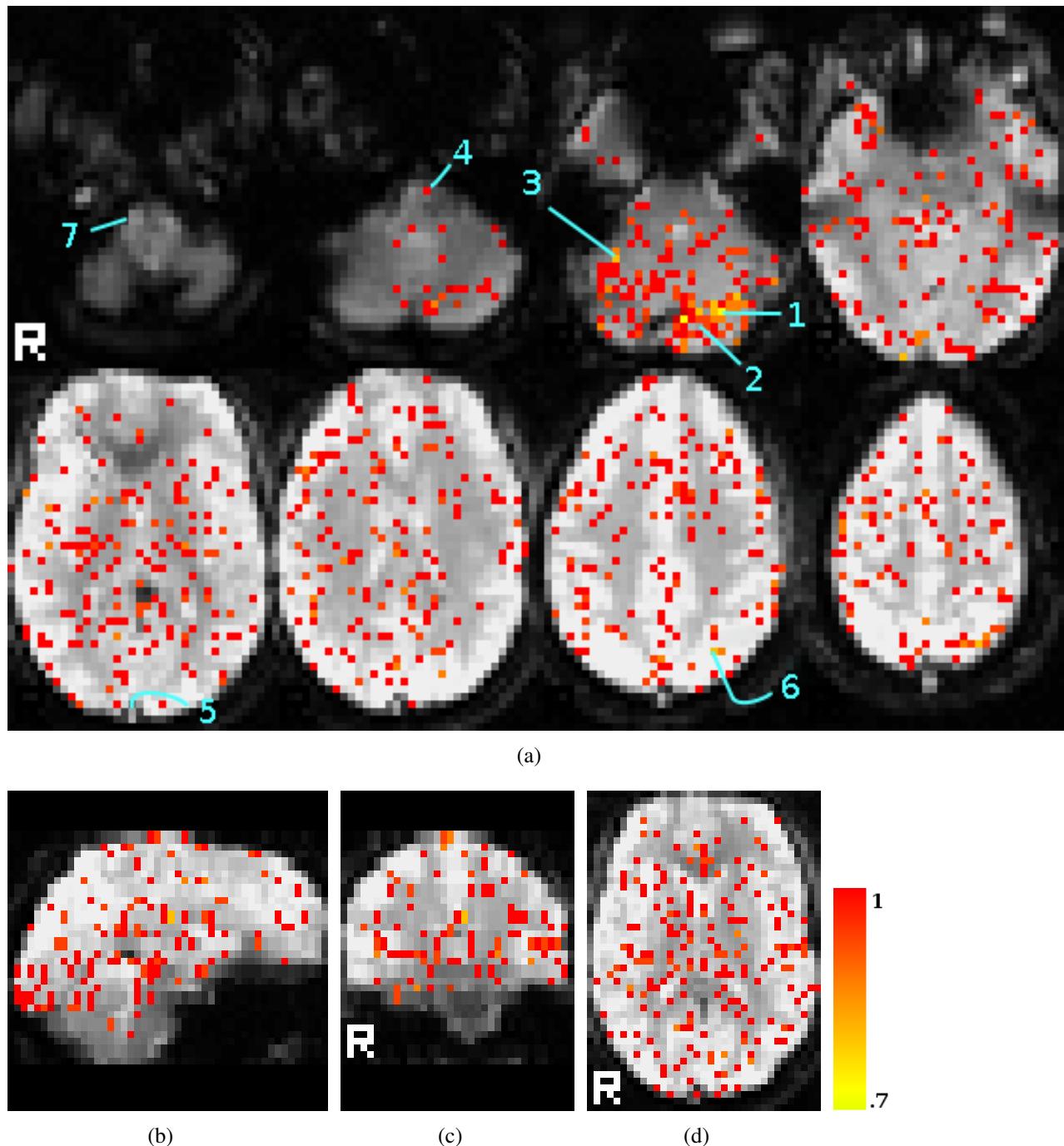


Figure 6.2: Sagittal, coronal and axial (6.2(b) 6.2(c) 6.2(b)), as well as a series of axial slices, 6.2(a). Units of match is normalized residual. The lowest (best) levels were .7. The highest error shown is 1.

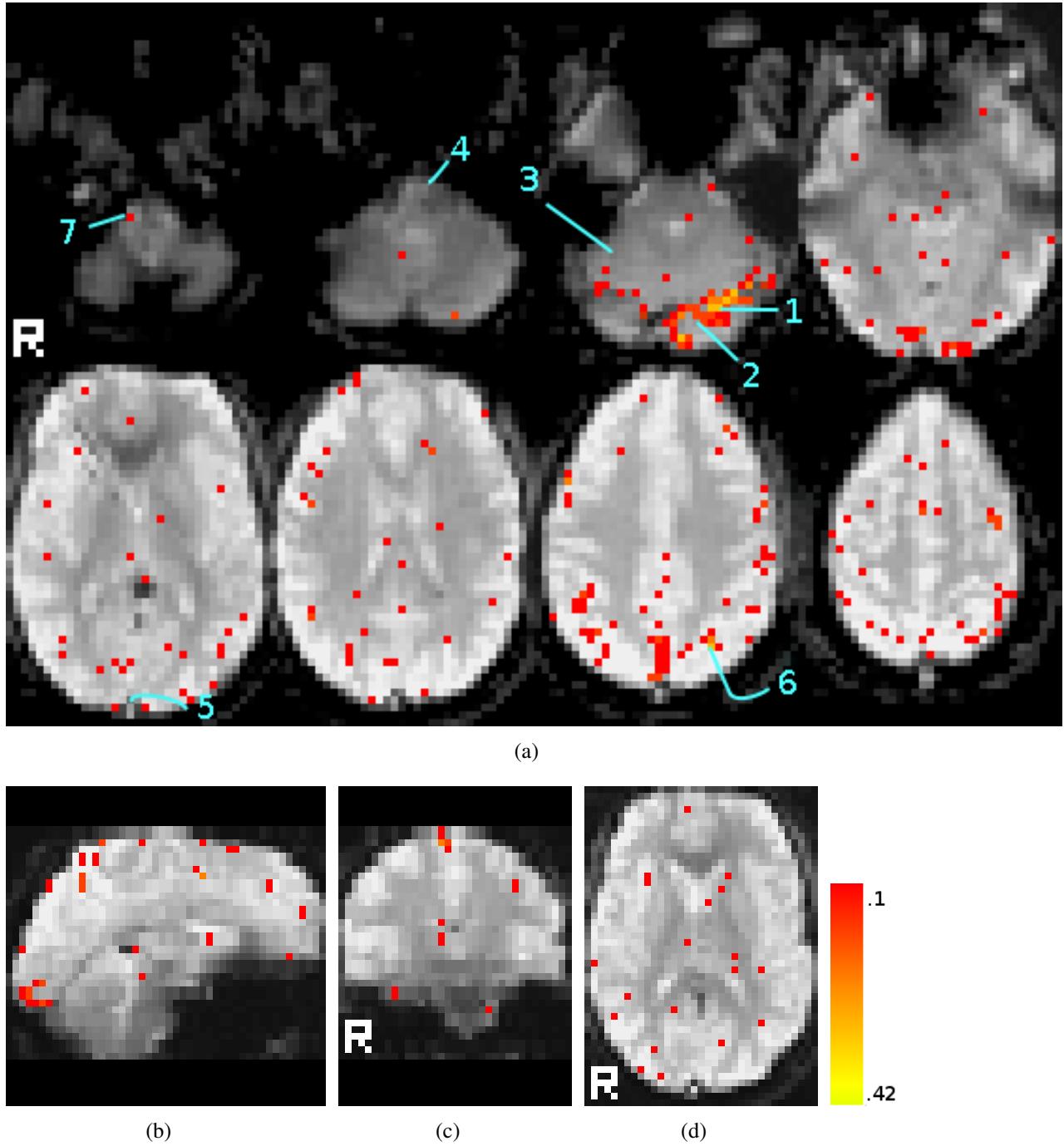
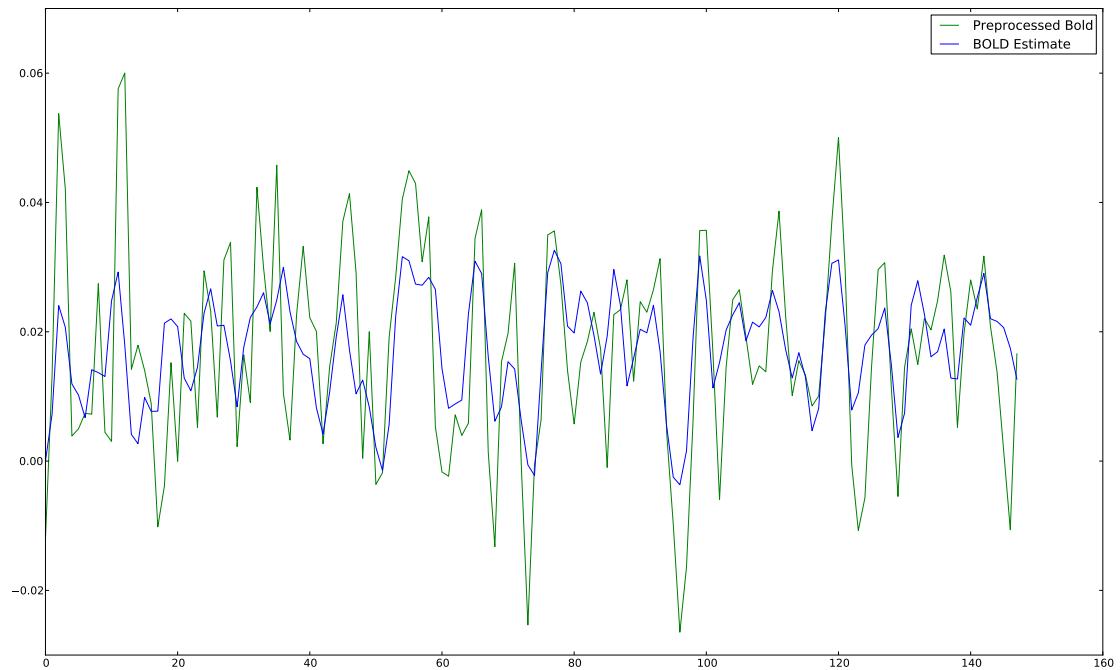
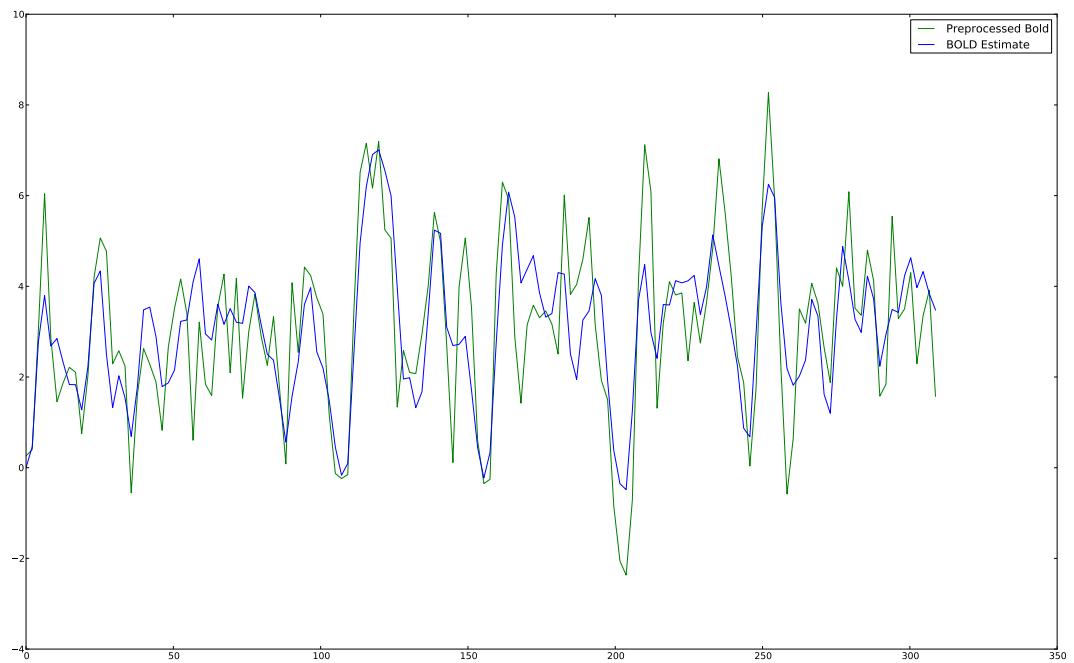


Figure 6.3: Sagittal, coronal and axial (6.2(b) 6.2(c) 6.2(d)), as well as a series of axial slices, 6.2(a). Units of match is bits (standard for base-2 Mutual Information). The highest (best) levels are .42. The worst shown is .1.

I chose several voxels to discuss further from Figure 6.2 and Figure 6.1. The first voxel, labeled

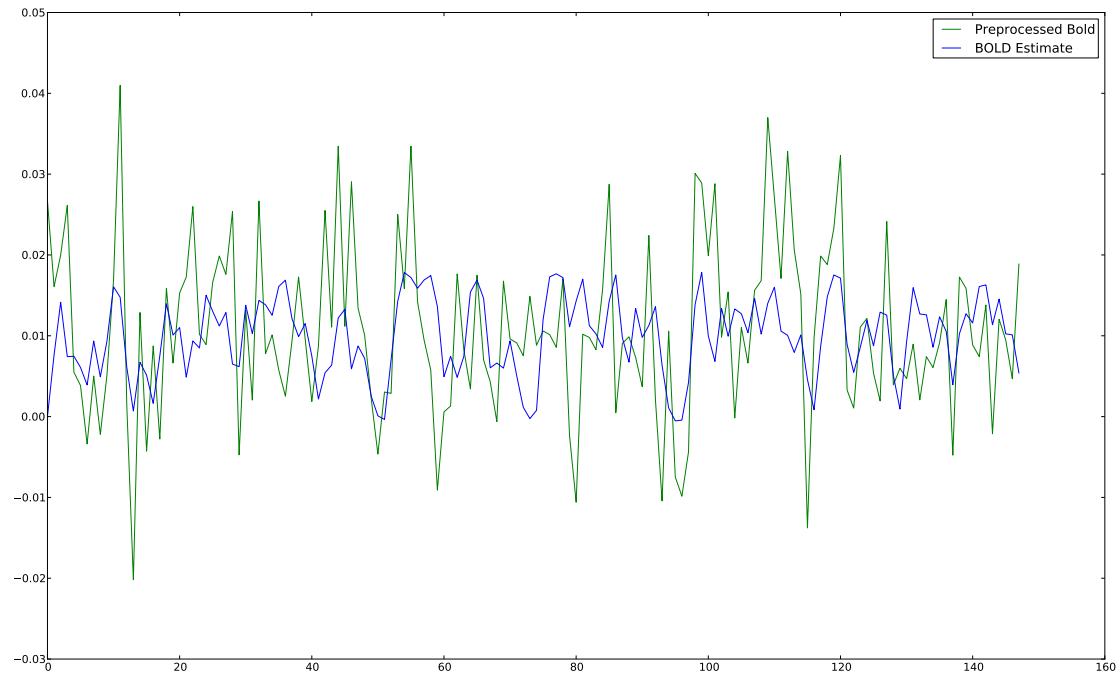


(a) Particle Filter

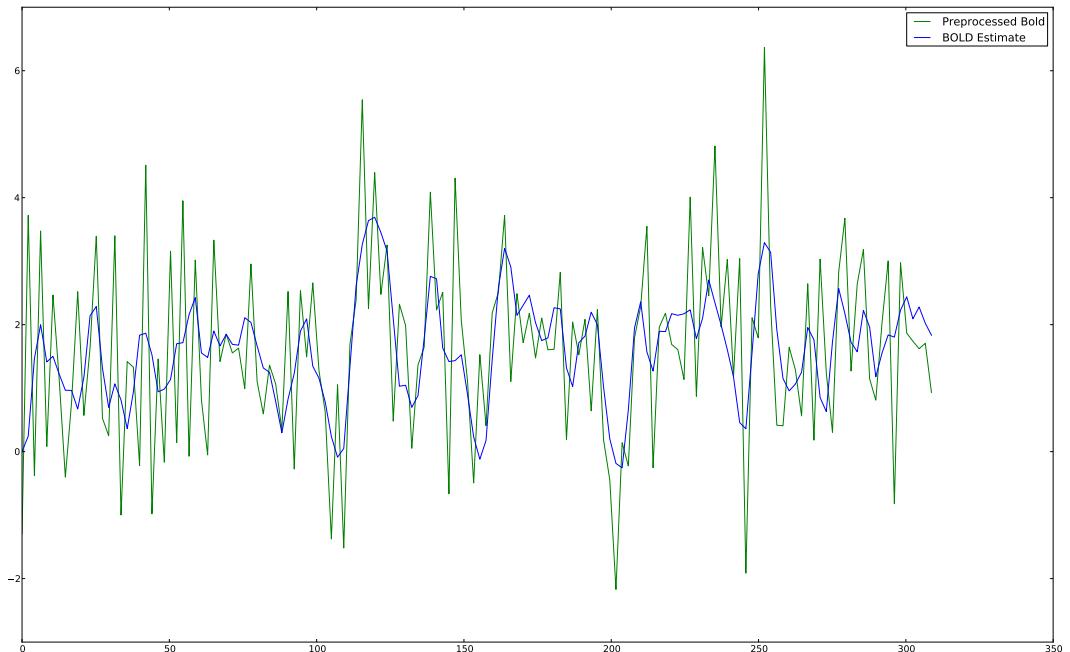


(b) SPM

Figure 6.4: Section 1, Estimated vs. Actual BOLD response



(a) Particle Filter



(b) SPM

Figure 6.5: Section 2, Estimated vs. Actual BOLD response

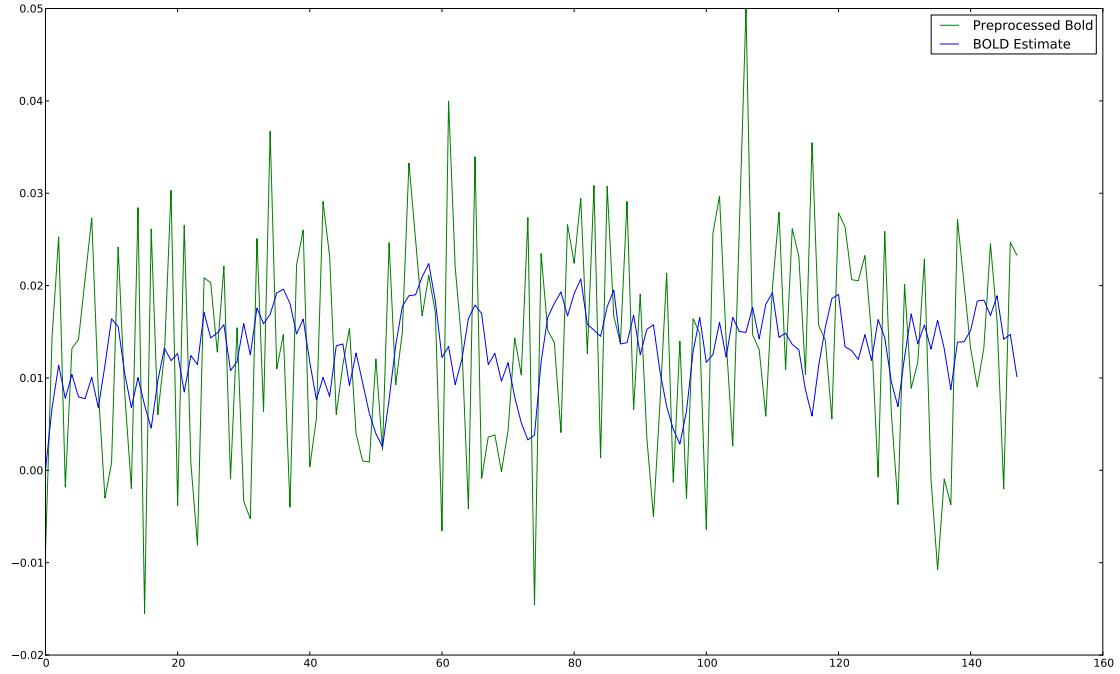
1, had a very high T-score as well as a low error (around .7). Thus, the fit should be very good in both the SPM and particle filter output; this comparison is shown in [Figure 6.4](#). Recall that SPM worked on a slightly less noisy time series because of the spatial smoothing; this explains the lack of the sharp peak in the particle filter's preprocessed data. Regardless, as expected, both fits are very good.

I chose the second voxel ([Figure 6.5](#)) because it was active in SPM and it would appear to be in a prime location to be active in the SPM image (given the results in the surrounding voxels). The fit, however shows just why the voxel did not have a particularly good error. Although certain peaks seem to fit rather well the input doesn't seem to correlate that well with the BOLD response. For instance 75 seconds in, the stimulus is not present, and so the signal should drop off; yet it doesn't. This is a good example of a voxel that seems to be reacting to something in addition to the direct stimulus. Even in what should be a rather smooth time-series in SPM, the BOLD signal looks very noisy.

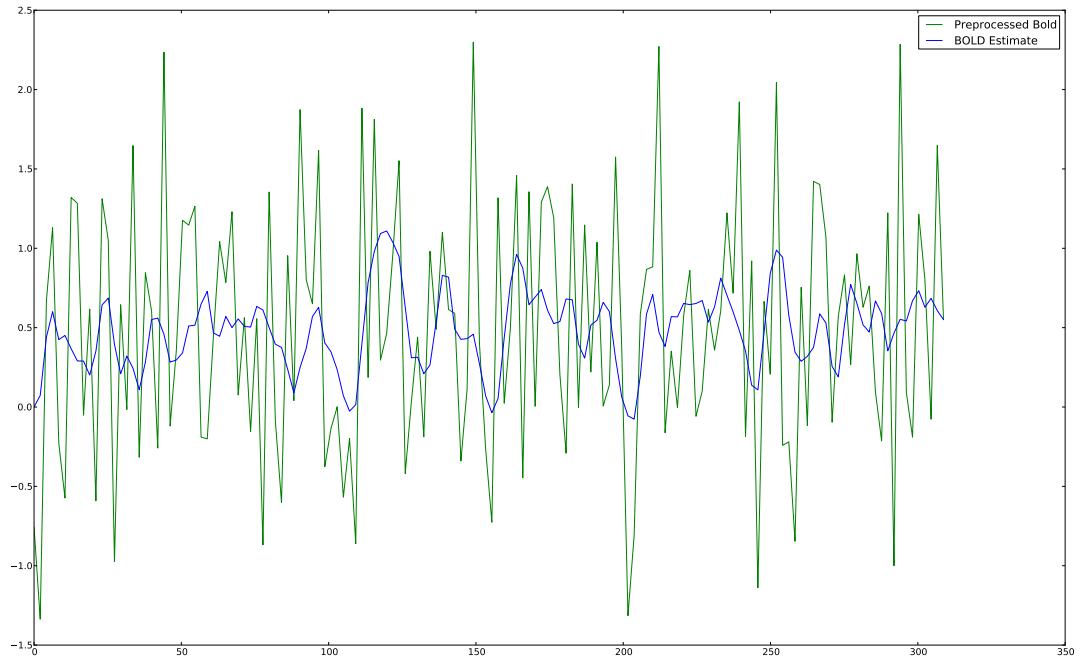
The third voxel, compared in [Figure 6.6](#), was far away from any other active voxels and yet had a very low (around .7) residual. In both preprocessed time-series the input is extremely noisy, yet by the normalized residual the response is good. This could be an example of a false positive, although the general trend of the response seems to follow the estimated response.

Whereas the third voxel was a region that was considered active by the particle filter algorithm but not SPM; the fourth voxel is the reverse. The particle filter did not find a good fit, but in SPM8 the voxel had a decently high T-score. [Figure 6.7](#) shows the line fit for each algorithm, and neither are particularly good; though in a few areas the upward or downward movements seem to be correlated. Regardless, this would seem to be a false positive from SPM8.

The fifth, and last single voxel to be analyzed is one more area that had a low residual according to the particle filter, and yet did not show up in SPM. The line fit is shown in [Figure 6.10](#), and unlike voxel 3 is certainly *not* a false positive. Regions such as this are a very good example of why spatial smoothing can be problematic. By having a more adaptive model, and thus reducing bias error, it is possible to get a very good fit to single voxels such as this, and thus the need for spatial smoothing is somewhat reduced.

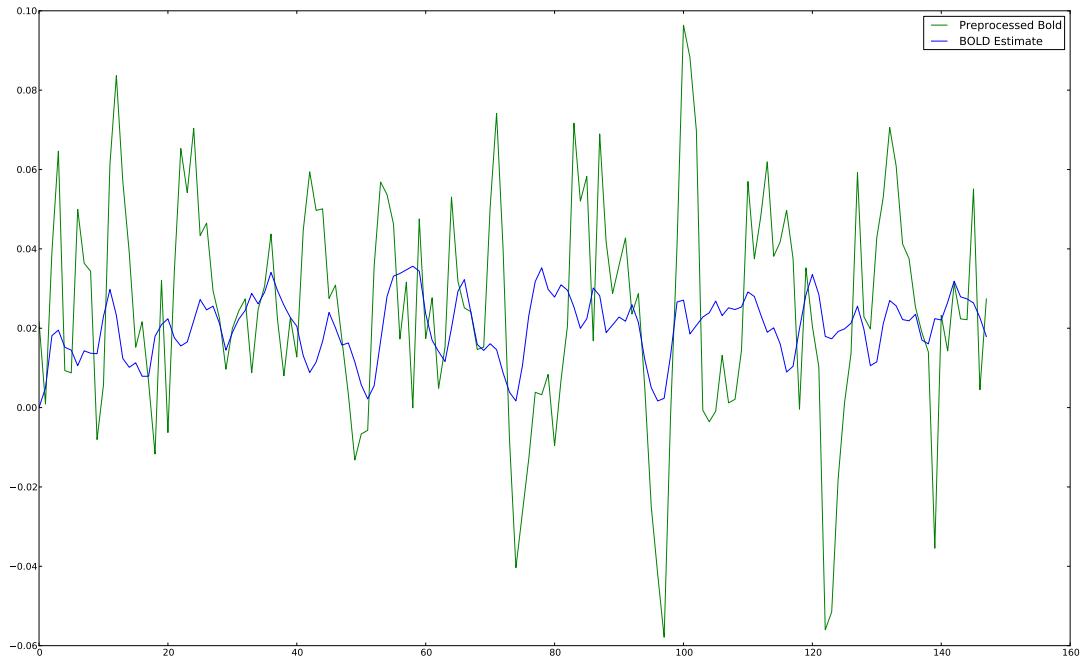


(a) Particle Filter

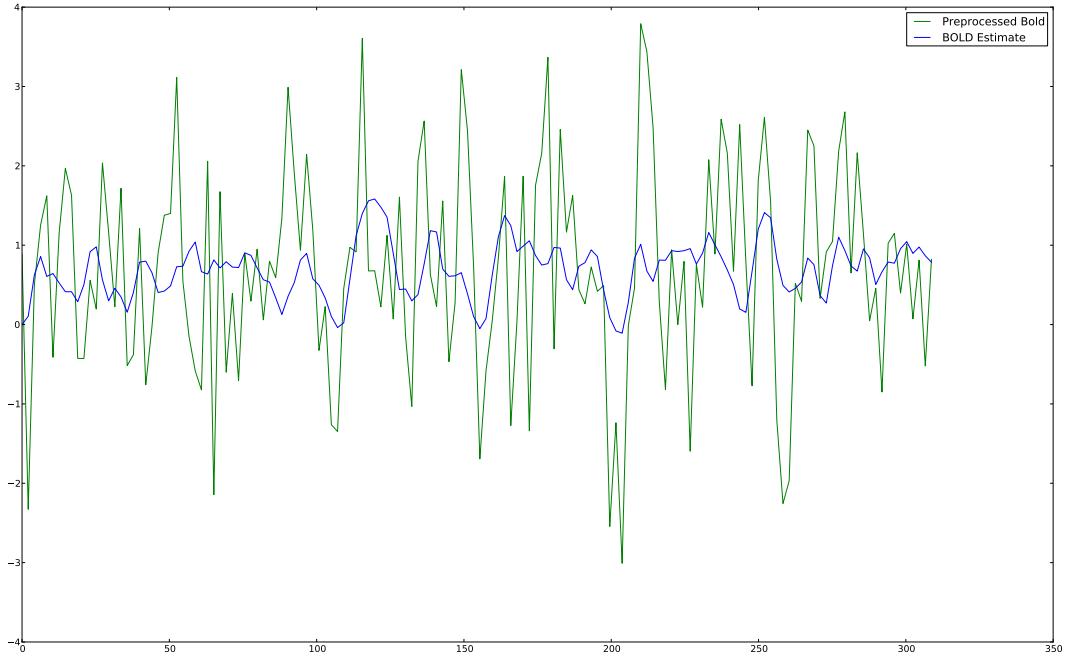


(b) SPM

Figure 6.6: Section 3, Estimated vs. Actual BOLD response



(a) Particle Filter

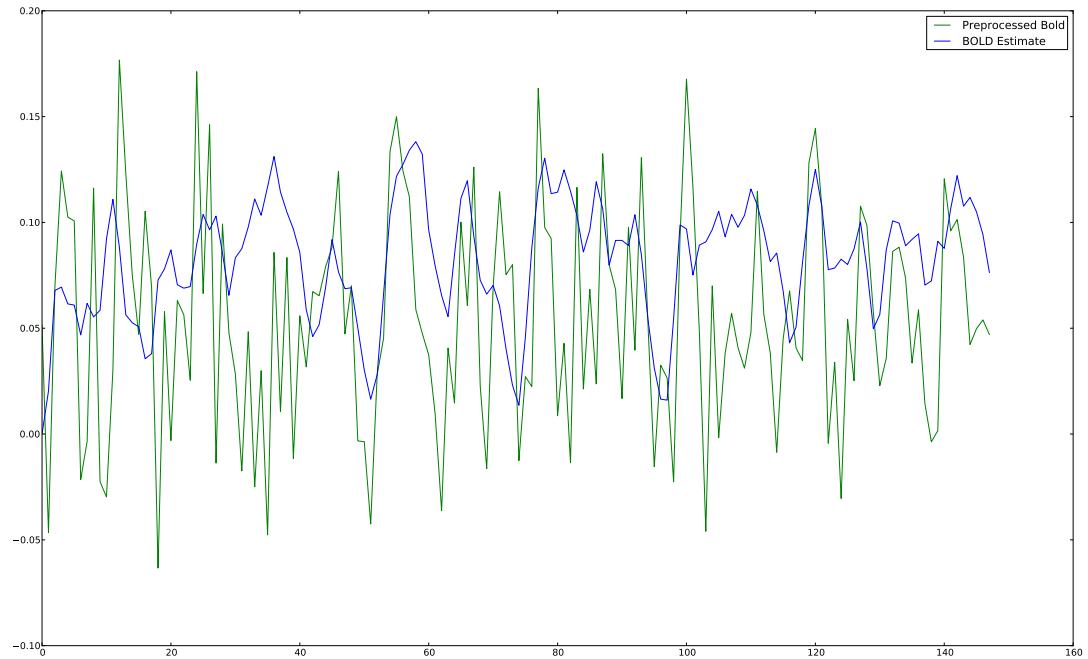


(b) SPM

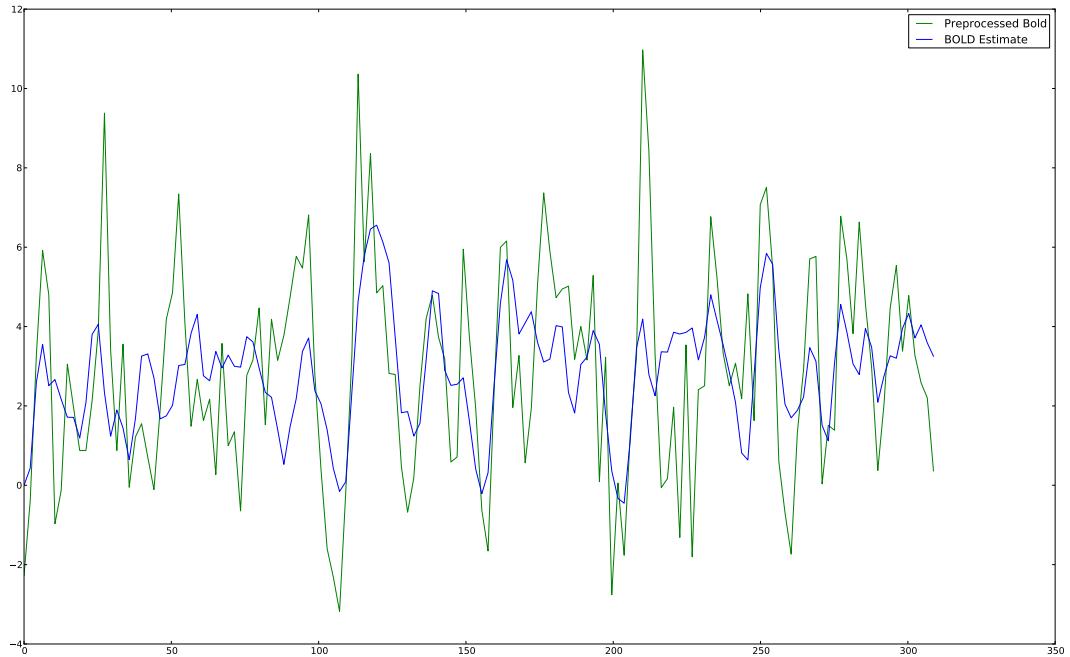
Figure 6.7: Section 4, Estimated vs. Actual BOLD response

6.2 Discussion

From the maps generated, the similarities to SPM8's results are encouraging. At the very least the output of the particle filter seems to meet the quality of SPM. The normalization of the residual is certainly necessary. Although there is no hard threshold on the normalized residual, it would appear that the normalization is providing a reasonable ordering of regression quality. The question of what threshold to use is a difficult one. The actual division of the residual by an estimate for the standard deviation of the signal would seem to mandate a threshold less than one. However, recall that the same media-absolute-deviation was added to the de-trended signal as a DC-gain. This means that a flat, no-response bold signal will actually have a residual greater than the standard deviation of the signal. Then again, the residual may be reduced for any point where the model spikes up due to stimuli. Ultimately if the model were to run right down the center of the distribution the residual would be equal to the MAD, and would have a normalized residual of 1. Thus, again, a normalized residual of 1 is at least as good as a flat line at the median. Other measures of fitness could also be used such as mean of the absolute error or mutual information. Although least squares is the best linear unbiased estimator, this signal is heavily nonlinear, so other estimators may exceed the OLS in terms of likelihood. Regardless, the normalized residual provides a good solution at this point, comparable to SPM.

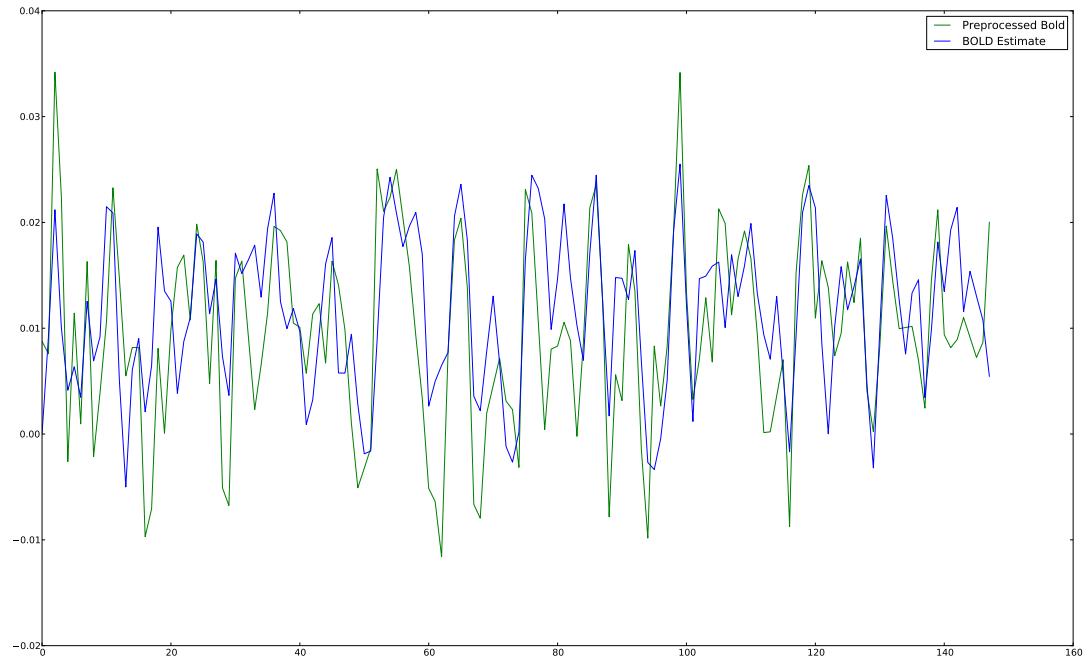


(a) Particle Filter

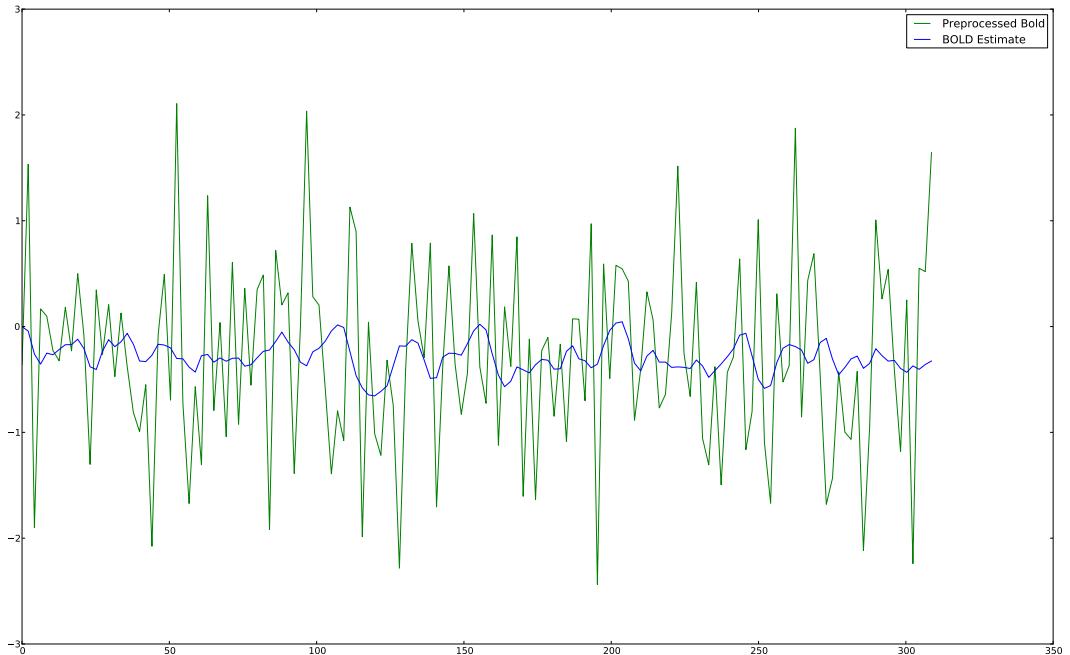


(b) SPM

Figure 6.8: Section 5, Below threshold in both particle filter checks, but above threshold in SPM. Mutual Information of 0.0212822, T-Value of 4.17399 and MSE of 1.14171.

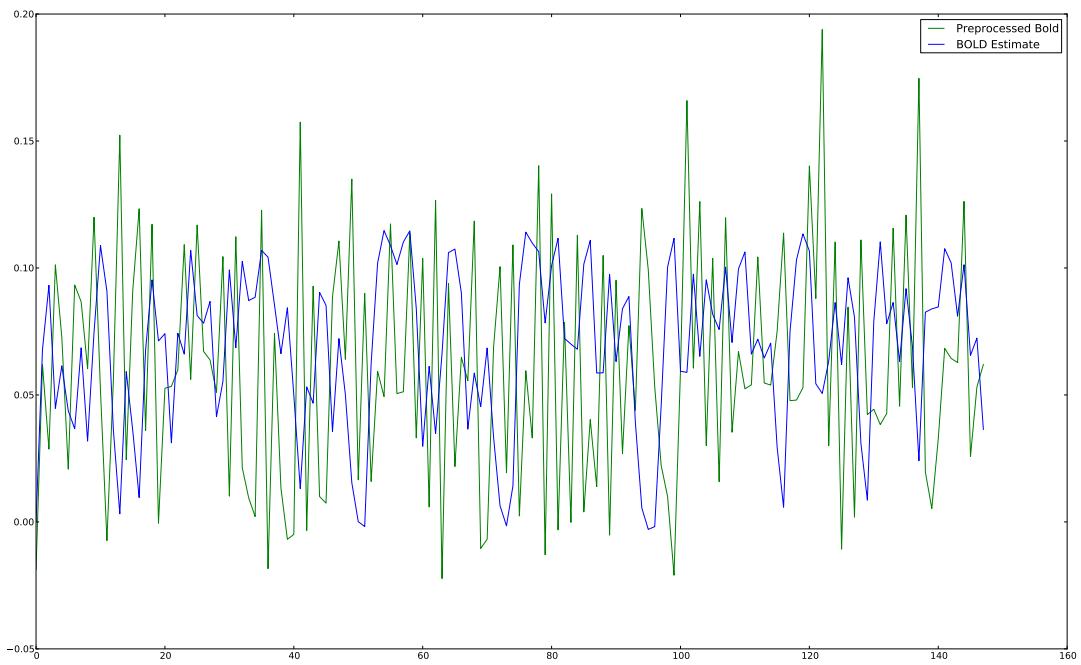


(a) Particle Filter



(b) SPM

Figure 6.9: Section 6, MI of 0.335504, T Value: 2.49154, normalized error: 0.783348 Not visible in SPM



(a) Particle Filter

Figure 6.10: Section 7, .1052 in MI, 1.31534 in SPM (which left this flat) and 1.09992 normalized \sqrt{MSE} . Only visible in MI map.

Chapter 7

Conclusion

In this paper I have proposed and demonstrated the use of a particle filter for the estimation of BOLD model parameters. In doing so, it has been possible to estimate the BOLD time-series and judge the quality of the estimated time-series with respect to the true BOLD signal. As result, it is possible to use the particle filter to localize areas where a known stimuli most directly drives neural activation. In the future it will be possible to use the estimated states (v, q, s, f) to drive other models and learn more about how regions of the brain interact.

A limitation often reached in previous works was an inconsistency of parameter estimates, most likely because of covariance between the model parameters. Although individual studies got consistent results, those results often differed widely from other similar studies. The reason for this is rather clear from the simulation results in [subsection 5.1.1](#). There is a significant amount of trade off between parameters to the point that a signal set of parameters is most likely not possible to derive from the BOLD response alone. It will therefore be beneficial to combine BOLD studies with cerebral blood flow or cerebral blood volume studies to gain multiple more measurements and further constrain the model. That said, the benefit of the particle filter is that it provides a full posterior distribution at the final time step. As such the true solution should be encoded in the particle filter's final distribution given priors that encompass the true parameters. This is beneficial in two ways; first, if, after the fact, some parameter becomes known from outside observation, it is then possible to construct a new probability conditional on the new observation. Secondly the results from multiple runs may be reasonably concatenated, using the final distribution from the previous run as the prior distribution of the next run. Rather than simply providing a starting point for parameters to converge from, it in fact continues convergence from the previous stopping point.

Although many versions of the BOLD model exist, and it is tempting to use more detailed models; from the results found here the issue of bias error from the BOLD model is not the biggest concern. Clarifying the distributions of parameters for the prior should be the first concern; currently no multi-patient full-volume studies have been done to estimate parameters. One future study that would be beneficial in this way would be an extensive study of what the priors should truly be. Although [2] gives an estimate of what is thought to be reasonable values, and later studies published

their estimate of the distributions, given the interplay between parameters it is unlikely that these priors are true to actual distribution that occurs *in vivo*. As I mentioned previously, the addition of simultaneous flow or volume measurements are another potentially powerful way to further confine the model, and thus deal with the elasticity of parameters. As I mentioned in [subsection 2.1.5](#), a chief advantage of using physiologically plausible models is that such data may in fact be added with relative ease.

Automatic detection of the noise level in the signal, to get a decent wieghting function.

In conclusion, using particle filters to estimate the BOLD response are a powerful method of fitting to noisy data. The technique also holds great promise as extensible platform to build more advanced models and techniques on top of. Integrating further information is necessary to move beyond the traditional Statistical Parametric Mapping and moving toward biologically and medically relevant FMRI scanning techniques.

Bibliography

- [1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press, 2005.
- [2] K. J. Friston, A. Mechelli, R. Turner, and C. J. Price, “Nonlinear responses in fMRI: the Balloon model, Volterra kernels, and other hemodynamics,” *NeuroImage*, vol. 12, pp. 466–477, 2000.
- [3] L. a. Johnston, E. Duff, I. Mareels, and G. F. Egan, “Nonlinear estimation of the BOLD signal.,” *NeuroImage*, vol. 40, no. 2, pp. 504–14, 2008.
- [4] V. a. Vakorin, O. O. Krakovska, R. Borowsky, and G. E. Sarty, “Inferring neural activity from BOLD signals through nonlinear optimization.,” *NeuroImage*, vol. 38, pp. 248–60, Nov. 2007.
- [5] K. J. Friston, W. Penny, C. Phillips, S. Kiebel, G. Hinton, and J. Ashburner, “Classical and Bayesian inference in neuroimaging: theory.,” June 2002.
- [6] R. C. DeCharms, F. Maeda, G. H. Glover, D. Ludlow, J. M. Pauly, D. Soneji, J. D. E. Gabrieli, and S. C. Mackey, “Control over brain activation and pain learned by using real-time functional MRI.,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, pp. 18626–31, Dec. 2005.
- [7] S. Ogawa, R. S. Menon, D. W. Tank, S. Kim, H. Merkle, J. M. Ellermann, and K. Ugurbilt, “Functional brain mapping by blood oxygenation level-dependent contrast magnetic resonance imaging A comparison of signal characteristics with a biophysical model,” *Biophysical Journal*, vol. 64, pp. 803–812, 1993.
- [8] R. B. Buxton, E. C. Wong, and L. R. Frank, “Dynamics of blood flow and oxygenation changes during brain activation: the balloon model,” *Magn. Reson. Med.*, vol. 39, pp. 855–864, 1998.
- [9] T. Deneux and O. Faugeras, “Using nonlinear models in fMRI data analysis: model selection and activation detection,” *NeuroImage*, vol. 32, no. 4, pp. 1669–1689, 2006.
- [10] D. a. Handwerker, J. M. Ollinger, and M. D’Esposito, “Variation of BOLD hemodynamic responses across subjects and brain regions and their effects on statistical analyses.,” *NeuroImage*, vol. 21, pp. 1639–51, Apr. 2004.

- [11] J. J. Riera, J. Bosch, O. Yamashita, R. Kawashima, N. Sadato, T. Okada, and T. Ozaki, “fMRI activation maps based on the NN-ARx model,” *NeuroImage*, vol. 23, pp. 680–697, 2004.
- [12] Y. Behzadi and T. T. Liu, “An arteriolar compliance model of the cerebral blood flow response to neural stimulus,” *NeuroImage*, vol. 25, pp. 1100–1111, 2005.
- [13] R. M. Weisskoff, C. S. Zuo, J. L. Boxerman, and B. R. Rosen, “Microscopic susceptibility variation and transverse relaxation : theory and experiment,” *Magnetic resonance in medicine*, vol. 31, no. 6, pp. 601–610, 1994.
- [14] J. Mandeville, J. Marota, C. Ayata, G. Zaharchuk, M. Moskowitz, B. Rosen, and R. Weisskoff, “Evidence of a cerebrovascular postarteriole Windkessel with delayed compliance,” *Journal of cerebral blood flow and metabolism : official journal of the International Society of Cerebral Blood Flow and Metabolism*, vol. 19, no. 6, pp. 679–689, 1999.
- [15] J. J. Riera, J. Watanabe, I. Kazuki, M. Naoki, E. Aubert, T. Ozaki, and R. Kawashima, “A state-space model of the hemodynamic approach: nonlinear filtering of BOLD signals,” *NeuroImage*, vol. 21, pp. 547–567, 2003.
- [16] T. Obata, “Discrepancies between BOLD and flow dynamics in primary and supplementary motor areas: application of the balloon model to the interpretation of BOLD transients,” *NeuroImage*, vol. 21, no. 1, pp. 144–153, 2004.
- [17] J. J. Chen and G. B. Pike, “Origins of the BOLD post-stimulus undershoot.,” *NeuroImage*, vol. 46, no. 3, pp. 559–68, 2009.
- [18] J. B. Mandeville, J. J. A. Marota, C. Ayata, M. A. Moskowitz, R. M. Weisskoff, and B. R. Rosen, “MRI Measurement of the Temporal Evolution of Relative CMRO₂ During Rat Forepaw Stimulation,” *Magnetic Resonance in Medicine*, vol. 951, pp. 944–951, 1999.
- [19] J. Frahm, J. Baudewig, K. Kallenberg, A. Kastrup, K. D. Merboldt, and P. Dechent, “The post-stimulation undershoot in BOLD fMRI of human brain is not caused by elevated cerebral blood volume.,” *NeuroImage*, vol. 40, pp. 473–81, Apr. 2008.
- [20] M. J. Donahue, R. D. Stevens, M. de Boorder, J. J. Pekar, J. Hendrikse, and P. C. M. van Zijl, “Hemodynamic changes after visual stimulation and breath holding provide evidence for an uncoupling of cerebral blood flow and volume from oxygen metabolism.,” *Journal of cerebral blood flow and metabolism : official journal of the International Society of Cerebral Blood Flow and Metabolism*, vol. 29, no. 1, pp. 176–85, 2009.
- [21] R. B. Buxton, K. Uludag, D. J. Dubowitz, and T. Lui, “Modeling the hemodynamic response to brain activation.,” *NeuroImage*, vol. 23 Suppl 1, pp. S220–33, 2004.
- [22] H. Lu, X. Golay, J. J. Pekar, V. Zijl, and P.c.m, “Sustained poststimulus elevation in cerebral oxygen utilization after vascular recovery,” *J. Cereb. Blood Flow Metab.*, vol. 24, pp. 764–770, 2004.

- [23] Q. Shen, H. Ren, and T. Q. Duong, “CBF, BOLD, CBV, and CMRO(2) fMRI signal temporal dynamics at 500-msec resolution.,” *Journal of magnetic resonance imaging : JMRI*, vol. 27, no. 3, pp. 599–606, 2008.
- [24] J. J. Chen and G. B. Pike, “Origins of the BOLD post-stimulus undershoot.,” *NeuroImage*, vol. 46, no. 3, pp. 559–68, 2009.
- [25] E. Yacoub, K. Ugurbil, and N. Harel, “The spatial dependence of the poststimulus undershoot as revealed by high-resolution BOLD- and CBV-weighted fMRI,” *J. Cereb. Blood Flow Metab.*, vol. 26, pp. 634–644, 2006.
- [26] Y. Zheng, J. Martindale, D. Johnston, M. Jones, J. Berwick, and J. Mayhew, “A model of the hemodynamic response and oxygen delivery to brain,” *Neuroimage*, vol. 16, pp. 617–637, 2002.
- [27] Y. Zheng, D. Johnston, J. Berwick, D. Chen, S. Billings, and J. Mayhew, “A three-compartment model of the hemodynamic response and oxygen delivery to brain.,” *NeuroImage*, vol. 28, no. 4, pp. 925–39, 2005.
- [28] Z. Hu, X. Zhao, H. Liu, and P. Shi, “Nonlinear Analysis of the BOLD Signal,” *EURASIP Journal on Advances in Signal Processing*, vol. 2009, pp. 1–14, 2009.
- [29] R. M. Birn, Z. S. Saad, and P. a. Bandettini, “Spatial heterogeneity of the nonlinear dynamics in the FMRI BOLD response.,” *NeuroImage*, vol. 14, pp. 817–26, Oct. 2001.
- [30] T. D. Wager, A. Vazquez, L. Hernandez, and D. C. Noll, “Accounting for nonlinear BOLD effects in fMRI: parameter estimates and a model for prediction in rapid event-related studies.,” *NeuroImage*, vol. 25, pp. 206–18, Mar. 2005.
- [31] A. T. Smith, K. D. Singh, and J. H. Balsters, “A comment on the severity of the effects of non-white noise in fMRI time-series.,” *NeuroImage*, vol. 36, no. 2, pp. 282–8, 2007.
- [32] K. J. Worsley, J. E. Taylor, F. Tomaiuolo, and J. Lerch, “Unified univariate and multivariate random field theory.,” *NeuroImage*, vol. 23 Suppl 1, pp. S189–95, 2004.
- [33] D. A. Hofmann, “An Overview of the Logic and Rationale of Hierarchical Linear Models,” *Journal of Management*, vol. 23, no. 6, 1997.
- [34] K. J. Friston, “Bayesian estimation of dynamical systems: an application to fMRI,” *NeuroImage*, vol. 16, pp. 513–530, 2002.
- [35] T. Ozaki, “The Local Linearization Filter with Application to Nonlinear System Identifications,” in *Proceedings of the first US/Japan Conference on the Frontiers of Statistical Modeling: An Informational Approach*, (Dordrecht), pp. 217–240, Kluwer Academic Publishers, 1994.

- [36] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking,” *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.
- [37] J. S. Liu and R. Chen, “Sequential Monte Carlo Methods for Dynamic Systems,” *Journal of American Statistical Association*, vol. 93, no. 443, pp. 1032–1044, 1998.
- [38] C. Musso, N. Oudjane, and F. LeGland, “Improving regularised particle filters,” *Sequential Monte Carlo methods in practice*, 2001.
- [39] M. Hürzeler, H. R. Künsch, M. Hurzeler, and H. R. Kunsch, “Monte Carlo Approximations for General State-Space Models,” *Journal of Computational and Graphical Statistics*, vol. 7, p. 175, June 1998.
- [40] J. Tanabe, D. Miller, J. Tregellas, R. Freedman, and F. G. Meyer, “Comparison of detrending methods for optimal fMRI preprocessing,” *NeuroImage*, vol. vol, pp. 15no4pp902–907, 2002.
- [41] A. M. Smith, B. K. Lewis, U. E. Ruttmann, F. Q. Ye, T. M. Sinnwell, Y. Yang, J. H. Duyn, and J. A. Frank, “Investigation of Low Frequency Drift in fMRI Signal,” vol. 533, pp. 526–533, 1999.