Full Brain 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

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 works 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 below a minute per voxel without requiring a linearization of the noise in the state variables. The activation results show regions similar those found in SPM; however, there are some notable regions not detected by SPM. Though the parameters selected by the particle filter based approach 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 attempted to characterize the model parameters. For this reason the final distribution of parameters is more medically relevant than a single estimate. Because the output of the particle filter is a full posterior probability, the reliance on the mean to estimate parameters is not strictly necessary. 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

List of Figures

List of Tables

Chapter 1

Introduction

Traditional methods of analyzing timeseries images produced by Functional Magnetic Resonance Imaging (FMRI) perform regression using the linear combination of explanatory variables. Though adding nonlinear degrees of freedom naturally mandates more computation, in this thesis I will discuss a Sequential Monte Carlo method of fitting a nonlinear model a computation cost that would still allow real time calculations for multiple voxels. More practically, this method is an alternative method of detecting neural activity from 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 single 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 recalculating parameters; for instance to impose 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 Core 2 Duo Q6600).

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 pipeline. The results are described separately for simulated data and real FMRI data in Chapter 5 and Chapter 6, respectively. Finally in Chapter 9 there is a discussion of the usefulness and implications of this technique as well as recommendations for the 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 patients 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 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 [7]. 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 becomes 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 millimeter) 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, if everything else stay the same, blood vessels in an active area will tend to have less oxygenated hemoglobin, and more deoxygenated

hemoglobin (due to the increased rate at which oxygen is being consumed). This would then result in an attenuated FMRI signal. However, to compensate for activation, muscles that control blood vessels relax allowing increased 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 the overcompensation that provides the primary signal detected with FMRI. This cascade of events is believed to drive a prolonged increased in local metabolism, blood flow, blood volume, and oxygenated hemoglobin. The differences in onsets and recovery times of these variables are what causes the distinguishing characteristics of the BOLD signal. Unfortunately, FMRI 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 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, whereas T2 relaxation times are usually less than 100ms. 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 EPI.

Increasing the spatial resolution of EPI necessarily requires more time or faster magnetic field switching. Increasing magnet switching rates can result in more artifacts and lower signal to noise ratios. The result is that at best FMRI is capable of 1 second temporal resolution. The signal is also diluted because each voxel contains a variety 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 discerning of changes in Deoxyhemoglobin/Oxyhemoglobin. Therefore, it is necessary to assume that the only chemical changes will be in capillary beds feeding neurons. In practice this may not be the case, for instance near large veins, and it may explain some of the noise seen in FMRI imaging (see Section 4.2.1. Be-

cause MRI is unitless 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.

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. Since the local MR signal depends on the ratio of Deoxyhemoglobin to Oxygenated Hemoglobin, increased blood volume affects 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 given flow waveform. 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).

Although [8] demonstrated that a well chosen flow waveform could explain most features of the BOLD signal, it stopped short of proposing a realistic waveform for the CBF and CMRO2. In [2] Friston et. al. gave a reasonable and simple expression for CBF input based on a flow inducing signal. Additionally, in the same work Friston et. al. proposed a simple method of estimating metabolic rate: as a direct function of the inward blood flow. By combining these equations with the balloon model from [8], it is possible to predict the BOLD signal directly from a stimulus time course.

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

$$\dot{f} = s \tag{1.2}$$

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

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

where s is a flow inducing signal, f is the input blood flow (CBF), v is normalized cerebral blood volume (CBV), and q is the normalized local deoxyhemoglobin. The parameters controlling blood flow are ϵ, which is a neuronal efficiency term, $u(t)$, which is the stimulus, and τ_f, τ_s which are time constants. The parameters for the evolution of blood volume are E_0 which the resting metabolic rate and α which is Grubb's parameter controlling the balloon model. τ_0 is a single time constant controlling the speed of v and q.

This completed balloon model was well summarized again in [15]. Ogawa et. al. [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 regions 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, there are other additions which give it more degrees of freedom.

1.5 Post Stimulus Undershoot

Although the most widely used, the BOLD model described in Equation 1.4 and Equation 1.8 has been extended in various fashions. 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 be at play. Two prominent theories exist to explain 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]. In [17] a compelling case is made 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 (which has little effect on the BOLD signal).

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 venous and arterial blood. In [9] Deneux et. al. compared these models and though that work did not deal extensively with the post-stimulus undershoot, it did show incremental improvements in quality from the additional parameters. 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.

In summary, there have been extensive refinements to the Balloon model; however, the increased complexity and lack of known priors make these models difficult to work with. Additional degrees of freedom could also make parameter estimation intractable.

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 state equation Jacobian, [9, 28]. The steady state of the Balloon model equations gives:

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

where the parameters are all the same as in Equation 1.4

In real FMRI data, there is a significant nonlinearity in response; with short sequences responding disproportionately strongly [29, 30, 9]. This nonlinearity is accounted for in the Balloon model, although [9] shows that when duration of stimuli varies greatly, modeling Neural Habituation is necessary to fully capture the range of responses. In both [29] and [9] it was found that stimuli lasting longer than 4 seconds tend to be more linear, which is why block designs are so well accounted for by the General Linear Model (see Section 2.1.3).

Another interesting result of [9] was the sensitivity analysis. There it was found that the parameters are far from perpendicular, and that very different parameters could give nearly identical BOLD output. The means that that without constraining parameter values, they may not be precisely ascertainable. This could explain discrepancies in previous studies (Table 1.1).

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

Chapter 2

Alternative Techniques

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 simple localization, and move toward characterizing the response curve. Doing so necessitates more complex models and requires more computation. 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 may be considered generalization of the idea of activation detection. Given the popularity of Statistical Parametric Mapping (SPM) it is important to draw a distinction between it and the methods proposed in this work.

2.1.1 Classical Activation Detection

The most basic method of analyzing FMRI data is by 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 4.2.1), 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 Section 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 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]. As 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 than the simple square wave method discussed in Section 2.1.1. Hierarchal 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 general linear model is defined as:

$$Y(t) = X(t)\beta + \epsilon(t) \tag{2.1}$$

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 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 [10]. The

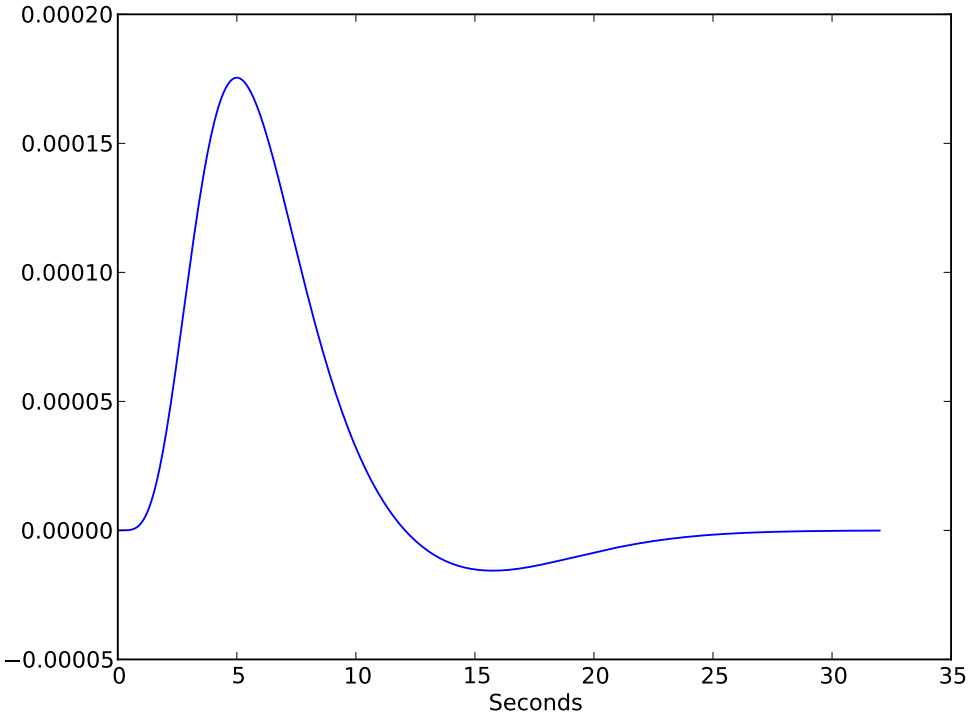


Figure 2.1: Canonical Hemodynamic Response Function (y-axis unts are arbitrary because normalization is performed).

Canonical HRF that is most often used, has been optimized for the visual cortex. As discussed in Section 1.5, there are certainly variations in the shape of the BOLD response, over both brain regions and patients. Handwerker et. al. [10] discuessed 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 account for variation between patients. Thus, the inability to fit parameters other than scale hinders analysis of activation.

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 mixes young and old, incorporating that into the model is wise, regardless of whether it is the goal of the test. The reason to do this is to account for additional variance that may not be otherwise explainable. 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]. Note that in Empirical Bayes point estimators are used for each θ, rather than 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 potential use. Besides not allowing HRF differences between patients, there is no reasonable way to incorporate other forms of physiological data. Combined FMRI CBF or CBV imaging methods are improving, as seen in [17]. These techniques could shed light on neural activation by providing extra measurements, yet a physiologically reasonable model is necessary to incorporate this extra data. 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.

2.2 Approaches to the Balloon Model

Unlike Statistical Parametric Mapping, the techniques described in this section are all attempts to regress 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 Polynomial Approximation

In [34], a novel combination of linear and nonlinear modeling was used to generate parameter estimates. Because there is no closed form solution to the ballon model, it impossible to calculate the Jacobian matrix $\frac{\partial Y}{\partial \theta}$. Thus [34] approximates the partial using a Volterra Kernel. At each step of the Expectation-Maximization (EM) algorithm, the differential equation is integrated and the residuals calculated. 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 there is an analytical solution calculating partial derivatives is easy. The full E-M algorithm for estimating the states is notation-heavy and can be found in [34].

There are a few caveats with this method of optimization. First, the partials are based on an approximate values (using Volterra-Kernels) of y. Importantly, the Volterra-Expansion of y is not able to model interactions between state variables [34]; which certainly increases error. 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

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 (although Section 2.2.1 circumvented this by calculating a Volterra Series approximation). Without a Jacobian for residuals, the Gauss-Newton method is impossible. Additionally, a gradient descent is slow without the ability to form partials of the output with respect to

all the parameters.

Still, there are other heuristic algorithms that could illuminate the BOLD response. Simulated Annealing (SA) is a common method of optimizing high dimensional regression problems. First the program selects a random starting point, then at each iteration it selects a random point in the neighborhood. If the that new 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. 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. Since every step requires a re-integration of the balloon model, it can be extremely time consuming, which is why I did not use it here.

Algorithm 2.1 Simulated Annealing Algorithm

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

Genetic algorithms (GA) 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. New parameter sets are generated by randomly combining pieces of the old parameter sets. The pieces are chosen at a rate proportional to the fitness of the donor; thus good parameter sets tend to pass on their properties. In addition to this, random mutations are introduced that come from no existing parent. The fitness function is then used to weight the new generation, 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 simulated annealing (or any generic non-linear optimization), 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 high computation time. In the 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, $E = \{\Theta_0, \Theta_1, ...\Theta_N\}$
for G generations **do**
  Calculate fitness for each Θ, Ex. for residual R, $1/R$ or, e^{-R}
  for i in N **do**
    Randomly select two parents (with higher probability for more fit Θ's)
    Randomly merge parts of the two parents to form a new Θ_i
    With low probability introduce random mutations to parameters in Θ_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,

$$X(t) = g(u(t), X(t-1)) \tag{2.3}$$
$$Y(t) = h(X(t) \tag{2.4}$$

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. Hu et. al. used the UKF to perform a similar type of analysis to the one performed in this work [28].

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 could be a gross error. On the other hand, for small variances and short time steps the Gaussian distribution is a good fit, and so in some cases the Unscented Kalman Filter could work quite well. The assumption of Gaussianity is what allows the UKF to estimate the posterior using only the first and second moments; however, if this assumption is violated significant error will result.

To determine the amount of error incurred in a Gaussian estimate during a typical sample period, I assigned the states of BOLD equations according to a four dimensional Gaussian. I then

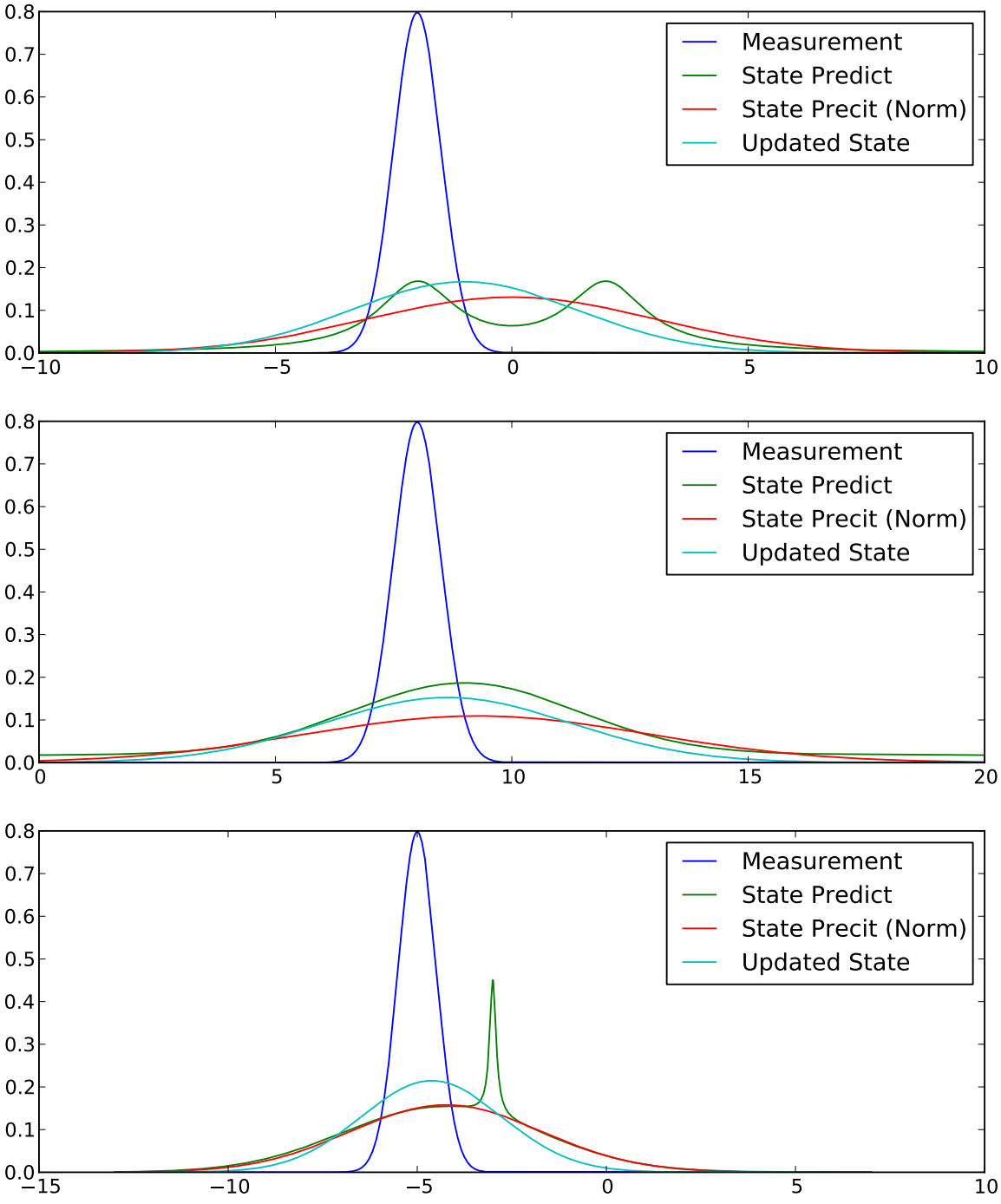


Figure 2.2: Example updates of a distribution using Kalman Filter, [1]

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

propagated the states through two seconds of simulation (a typical TR in FMRI) and plotted the resulting marginal distributions against a Gaussian distribution. This also demonstrates the degree of nonlinearity present in the system. The parameters used are shown in Table 2.1

In order to drive the system without input I intentionally set s_t a non-steady state, but physiologically plausible, value. The value of u is left at zero the entire time, so the system would decay naturally (see Section 1.4) Figure 2.3 shows the results when the system after the system ran for 100 milliseconds. The Q-Q plots fit will with a Gaussian, demonstrating that at this short time interval nonlinearities have not yet begun to effect the distribution. However, Figure 2.4 shows 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 started to deviate from the Gaussian distribution. As a result the uncertainty in y deviated from the Gaussian distribution as well. This is important, because although approximating the distribution with a Gaussian based on the first two moments will worked in the short run, within a period less than typical measurement rates the distribution deviated from a Gaussian substantially.

Thus, even without introducing variation in the model parameters, a distinct nonlinearity and non-Gaussianity was present. More advanced tests where parameters (especially α) are varied would certainly introduce more error into the Gaussian estimate. For this reason, estimating the posterior distribution using only the first two moments, which is the basis for the UKF, is not wise.

2.2.4 Hybrid Methods

In [15], a maximum likelihood method for innovation processes was used, as described by [35]. Ozaki et. al. used a similar construction to a Kalman filter to filter the BOLD signal. The method used in [15] performs maximum likelihood on the innovations rather than the absolute signal levels. By using a Local Linearization Filter, innovation noise such as noise in $\dot{f}$ is turned into simple

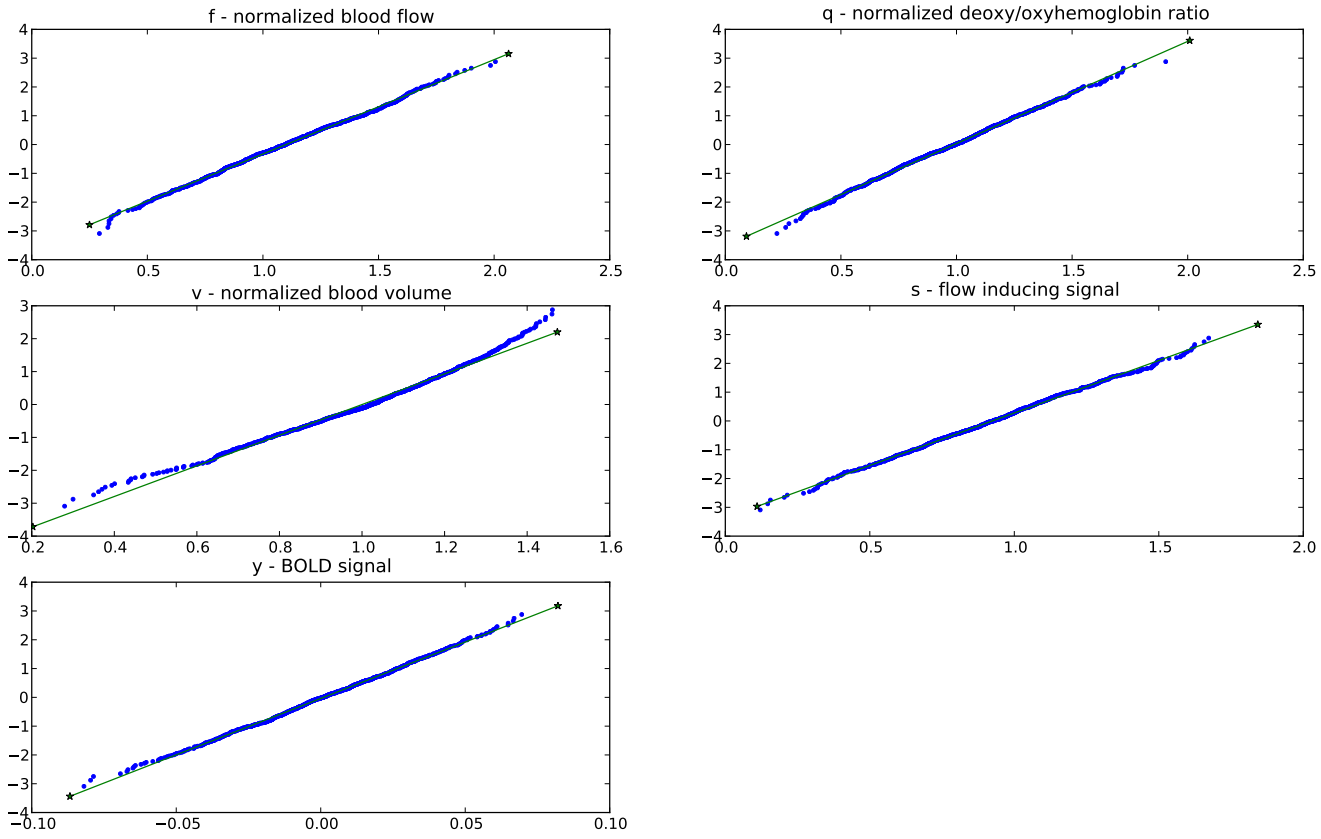


Figure 2.3: Distributions of state variables after simulating for .1s

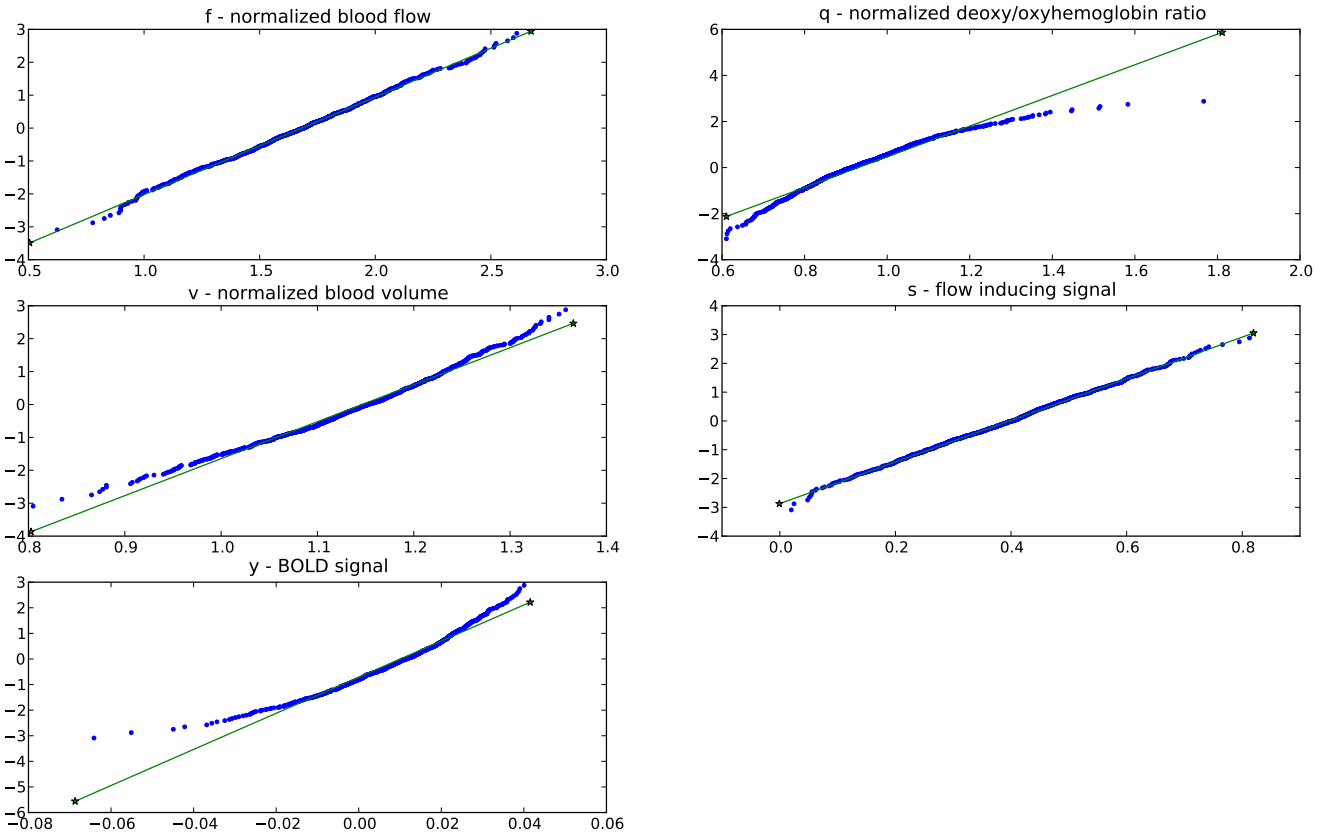


Figure 2.4: Distributions of state variables after simulating for 1s

Gaussian noise. This approach is useful when the DC portion of the signal is of no use; however, it depends greatly on the type of signal. I found that this method was not very effective for the BOLD signal, when I used it with the particle filter approach discussed in the next chapter.

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 Section 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 skewed their 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 4.3.1.

2.2.5 Previous Particle Filter Approaches

In [36] a particle filter based approach was used to integrate the BOLD equations. The method used in that work focused primarily on estimating the BOLD output and state equations as a nonlinear stochastic differential equation. The primary difference between [36] and this work is that [36] took the parameters as a given. Thus, differences in the BOLD output were taken to be primarily driven by stochastic changes in the underlying state equations. Because the parameters were not allowed to change, the estimate of the BOLD signal was not very good. The fact that the differences in BOLD response cannot be explained solely by stochastic elements is important, however. The particle filter library created in that work, $dysii$, forms the basis for the particle filter used in this work, and was extremely well designed. The work also clearly presents the particle filter; both its derivation and use.

2.3 Conclusion

Currently 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 points estimating the posterior.

Chapter 3

Particle Filters

3.1 Introduction

Particle filters, a type of Sequential Monte Carlo (SMC) method, are a powerful method for estimating the posterior probability distribution 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 empircal distribution rather than the as the first two moments of a Gaussian. Thus, particle filters are preferred when the model is nonlinear, and non-gaussian. This section is largely based on [37] 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 is often troublesome, when the system being modeled has physical meaning, establishing reasonable ranges for parameters may be quite easy. Optimizing the prior distribution can be more difficult, unless the system has been extensively studied.

Suppose a set or stream of measurements at discrete times are given, $\{Y_k, k = 1, 2, 3, ...K\}$, where K may be infinite. 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 drives the value of $Y(t)$. Throughout this section 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 memebers 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.

I will also make a few simplifying assumptions 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, its assumes that input cannot directly influence the output, which in the case of the BOLD signal is a good assumption. 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 Sequential Importance Sampling

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 previously

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

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

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 $\alpha(X)$:

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

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:

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

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

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

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. Note that 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 cannot be modeled. This is why it is absolutely necessary that q fully covers $P(x^i_{0:k}|y_{0:k})$.

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

$$w^i_k q(x^i_{0:k}|y_{0:k}) = q(x^i_{0:k}|y_{0:k}) = P(x^i_{0:k}|y_{0:k}) \tag{3.8}$$

the estimated probability, $P(x^i_{0:k}|y_{0:k})$ 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.

3.3.2 Calculating Weights

To calculate the weight of a particular particle, it is necessary to calculate both $q(x^i_{0:k}|y_{0:k})$ and $P(x^i_{0:k}|y_{0:k})$. Note that $q(x^i_{0:k}|y_{0:k})$ 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 a helpful assumption.

$$q(x^i_{0:k}|y_{0:k}) = q(x^i_{0:k}|y_{0:k-1}) \tag{3.9}$$

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 model prior.

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

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 $q(x^i_{0:k}|y_{0:k})$ may then be simplified:

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

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}, I make the good assumptions that:

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

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, $P(x^i_{0:k}|y_{0:k})$ may be broken up as follows (primarily using Bayes' Theorem):

$$\begin{aligned}
P(x_{0:k}|y_{0:k}) &= \frac{P(y_{0:k},x_{0:k})}{P(y_{0:k})} \\
&= \frac{P(y_k,x_{0:k}|y_{0:k-1})\cancel{P(y_{0:k-1})}}{P(y_k|y_{0:k-1})\cancel{P(y_{0:k-1})}} \\
&= \frac{P(y_k|x_{0:k},y_{0:k-1})P(x_{0:k}|y_{0:k-1})}{P(y_k|y_{0:k-1})} \\
&= \frac{P(y_k|x_{0:k},y_{0:k-1})P(x_k|x_{0:k-1},y_{0:k-1})P(x_{0:k-1}|y_{0:k-1})}{P(y_k|y_{0:k-1})} \\
&= \frac{P(y_k|x_k)P(x_k|x_{k-1})P(x_{0:k-1}|y_{0:k-1})}{P(y_k|y_{0:k-1})} && \text{[Equation 3.12]} \\
&\propto P(y_k|x_k)P(x_k|x_{k-1})P(x_{0:k-1}|y_{0:k-1}) && [P(y_k|y_{0:k-1}) \text{ is constant}]
\end{aligned} \tag{3.13}$$

Plugging Equation 3.10 and the result of Equation 3.13 into Equation 3.7 leads to:

$$\begin{aligned}
w^i_k &\propto \frac{P(y_k|x^i_k)\cancel{P(x^i_k|x^i_{k-1})}P(x^i_{0:k-1}|y_{0:k-1})}{\cancel{P(x^i_k|x^i_{k-1})}q(x^i_{0:k-1}|y_{0:k-1})} \\
&\propto w^i_{k-1}P(y_k|x_k)
\end{aligned} \tag{3.14}$$

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 prior distribution PDF, $q(x_i(0))$, covers $P(x_i(0))$
3. Markov Property: $P(x_k|x_{0:k-1}) = Pr(x_k|x_{k-1})$
4. $q(x_{0:k-1}|y_{0:k}) = q(x_{0:k-1}|y_{0:k-1})$

3.3.3 Basic 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. This basic form of the particle filter is given in algorithm 3.1.

3.4 Sequential Importance Resampling

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

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**
    $x_k^i = x_{k-1}^i + \int_{t-1}^{t} f(x(\tau), u(\tau))d\tau$
    $w_k^i = w_{k-1}^i P(y_k|x_k)$
  end for
end for
$P(x(t_k + \Delta t)) \approx \sum_{i=0}^{N_p} w_k^i \delta \left(x - (x_k^i + \int_{t_k}^{t_k+\Delta t} f(x(\tau), u(\tau))d\tau) \right)$

weights. While this does help describe the tails of the distribution, it means a great deal of computation will be wasted. 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 [38] 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}.

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

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:

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

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 regularization process is defined as:

$$x_i = x_i + h\sigma\epsilon \tag{3.17}$$

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. [39] goes into significant depth and proves the optimality of the Epanechnikov Kernel for reducing MSE between the original and resampled distributions. However, [39] 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 ϵ from the standard normal, same dimensionality as X
  $x^i = x^i + hD\epsilon$
end for

It has been proposed by [40] 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 very often. Thus if resampling is performed at every step then this could certainly cause 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 leave

gaps in the distribution. Moreover, 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
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 could hinder convergence, which is why regularized resampling should only be done when N_{eff} drops off (less than 50). Other than the periodic regularized resampling, the regularized particle filter is identical to the basic sampling importance sampling filter (SIS).

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. However, there is risk in resampling. If for some reason the solution is not covered by the new support the algorithm may not be able to reach the true value. 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.

3.5 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. Typically it is assumed that this error is additive, 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 4.3.3.

3.6 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 I 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. I will assume a simple model for all the components, ignoring the complex nonlinear behavior that can occur in a diode.

For this example I will assume 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

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

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:

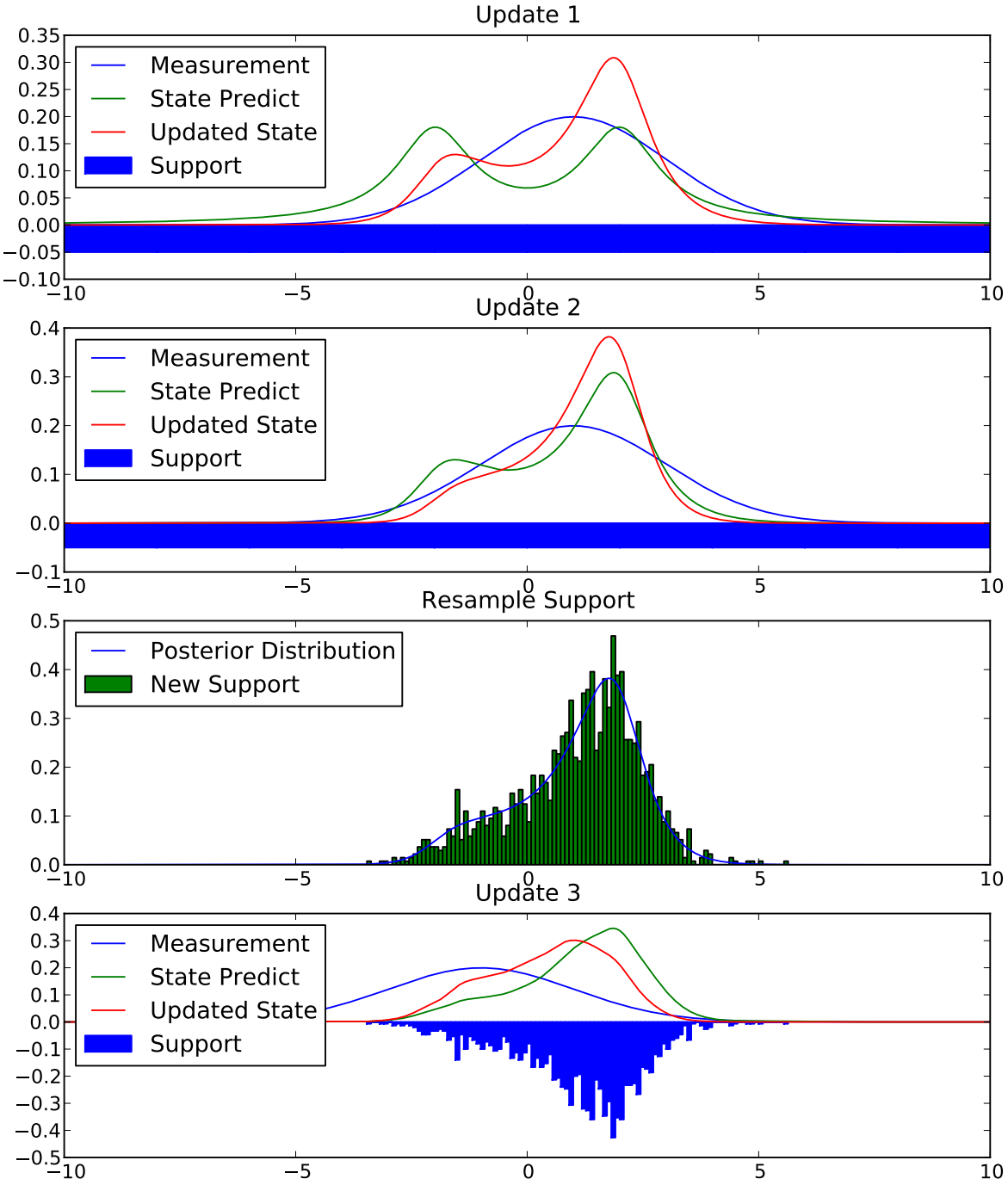


Figure 3.1: Particle Filter progression, note that the initial support is flat; the particles are equally spaced between -10 and 10

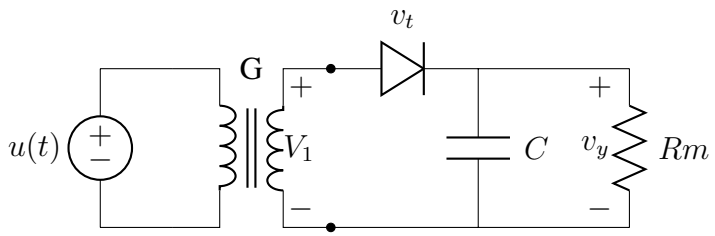


Figure 3.2: An Example Half Wave Rectifier Circuit, where G is the transformer gain, v_t is the activation voltage of the diode, $u(t)$ is the input at time t, C is the capacitance, R is the load resistance and v_y is the output voltage

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

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

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**
$G \sim Gamma(\frac{\mu_G^2}{\sigma_G^2}, \frac{\sigma_G^2}{\mu_G})$
$v_t \sim Gamma(\frac{\mu_{v_t}^2}{\sigma_{v_t}^2}, \frac{\sigma_{v_t}^2}{\mu_{v_t}})$
$C \sim Gamma(\frac{\mu_C^2}{\sigma_C^2}, \frac{\sigma_C^2}{\mu_C})$
$R_m \sim Gamma(\frac{\mu_R^2}{\sigma_R^2}, \frac{\sigma_R^2}{\mu_R})$
$v_y = 0$, (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 converging to a distribution that asymptotically approaches the true joint distribution of the $X(t)$. As I mentioned in Section 3.4, particles with weights approaching zero do not significantly contribute to the empirical distribution, so re-sampling will be necessary.

Chapter 4

Methods

Although the particle filter is a standard Regularized Particle filter, as described in [37], optimizing the particle filter for use with FMRI data is non-trivial.

4.1 Model

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

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

$$\dot{f} = s \tag{4.2}$$

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

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

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 account for that noise. It is common, however, to 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. It also helps that detrending is applied before the particle filter and that the BOLD model is dissipative. When no stimuli are applied, all the particles decay to ($\{0, 1, 1, 1\}$). Typical particle filters also use this state noise as an exploratory measure; however this method is less necessary when good priors are available.

For all the analyses in this work, 1400 integration points per second were used. Typically a step

size of 0.001 was sufficient, however, from time to time 0.001 can still be too high for the BOLD model.

4.2 Preprocessing

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 Section 2.1.2. After this, detrending is performed which is discussed in Section 4.2.2. Recall that FMRI signal levels are unit-less and though detrending is not always necessary, the data must always be converted into % difference from baseline. The generally accepted method is to use a high pass filter, although the cutoff frequency is application dependent and often applied haphazardly. Before going into the detrending used in this work, it is necessary to discuss the type of noise present in FMRI.

4.2.1 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 noise sources . 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 many as ten to fifteen percent of voxels can be affected ([42]), thus it is prevalent enough to cause significant inference problems [31]. It is still not clear what exactly causes this noise, although one possibility is the temperature difference in scanner magnetic coils[31]. It is clear that this drift signal is not solely due to a physiological effects, given its presence in cadavers and phantoms [43]. Interestingly, it is usually spatially correlated, and more prevalent at interfaces between regions. Though one potential source could be slight movement, co-registration is standard, making this unlikely. Regardless, the problem mandates the use of a high pass filter [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. Overall though there should be very little activation, and thus the signal consists entirely of noise. Therefore resting state data is perfect for analyzing noise. The locations were chosen from points all around the

brain, all in grey matter voxels. These time series were chosen because they were representative of different types of noise found in the resting state data.

The resting state was gathered in the exact same way as the data in Section 6.1, except without the stimuli.

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 needed in this work. Regardless, comparing the distribution to a Gaussian is informative, so Q-Q plots are used 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 and independent. 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 steps were plotted against a Gaussian.

Finally, removal of the drift is often performed with a high pass filter, so analyzed the distribution after subtracting of a spline, (see Section 4.2).

Figure 4.1 shows the results with a regression line fit to the points. 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 be collinear. 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. Note that in Figure 4.1 the points have all been normalized (changed to percent difference).

Note that 4.1(a) and 4.1(b) are well described by a Gaussian process with a small autocorrelation, 4.1(c) and 4.1(d) are not. In particular the tails of 4.1(c) do not seem to fit the Gaussian well. Also note the significant autocorrelation in 4.1(c) and 4.1(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 seem to 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 4.1(c) and 4.1(d), though not perfectly. Though the distributions still do not exactly fit the Normal, 4.3(d) is much improved compared to 4.1(d). In all, the de-trending is effectively removing Wiener noise.

4.2.2 Detrending

The non-stationary aspect of a Weiner process, presumably the result of integrating some ν_x is difficult to compensate for, and so many methods have been developed to compensate for it. [42] and

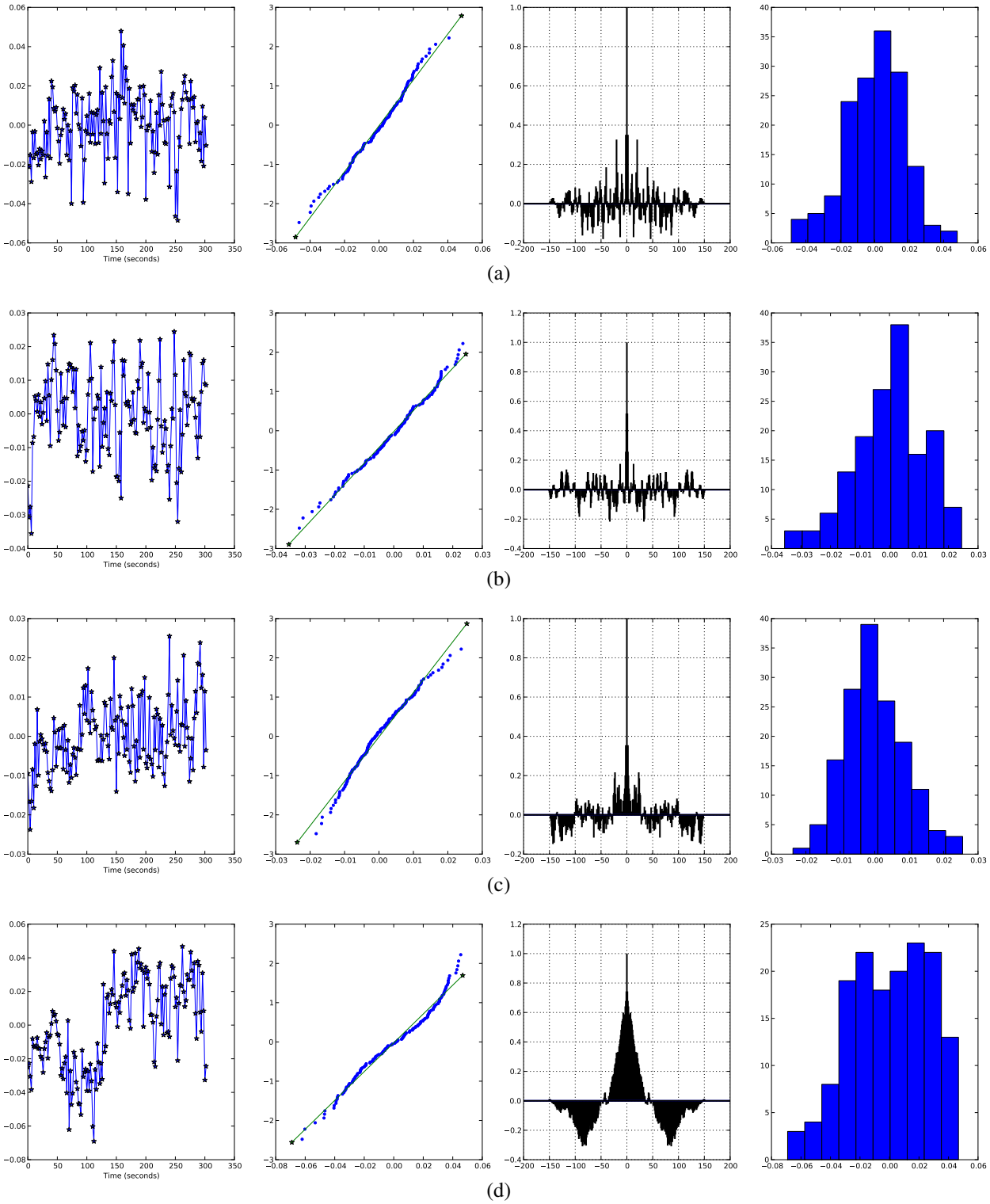


Figure 4.1: Q-Q Plots of normalized resting state data

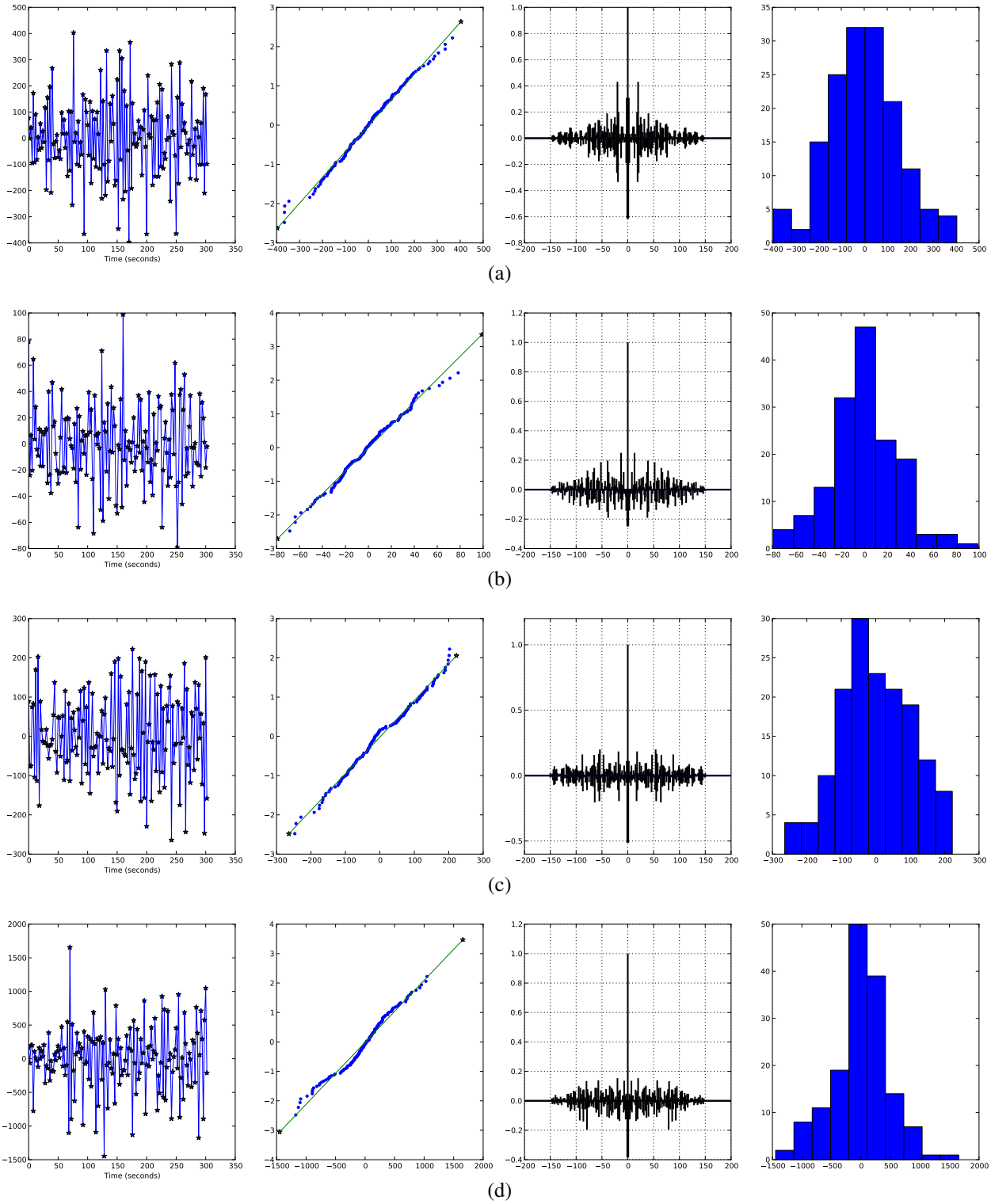


Figure 4.2: Q-Q Plots of resting state data, using the BOLD signal changes

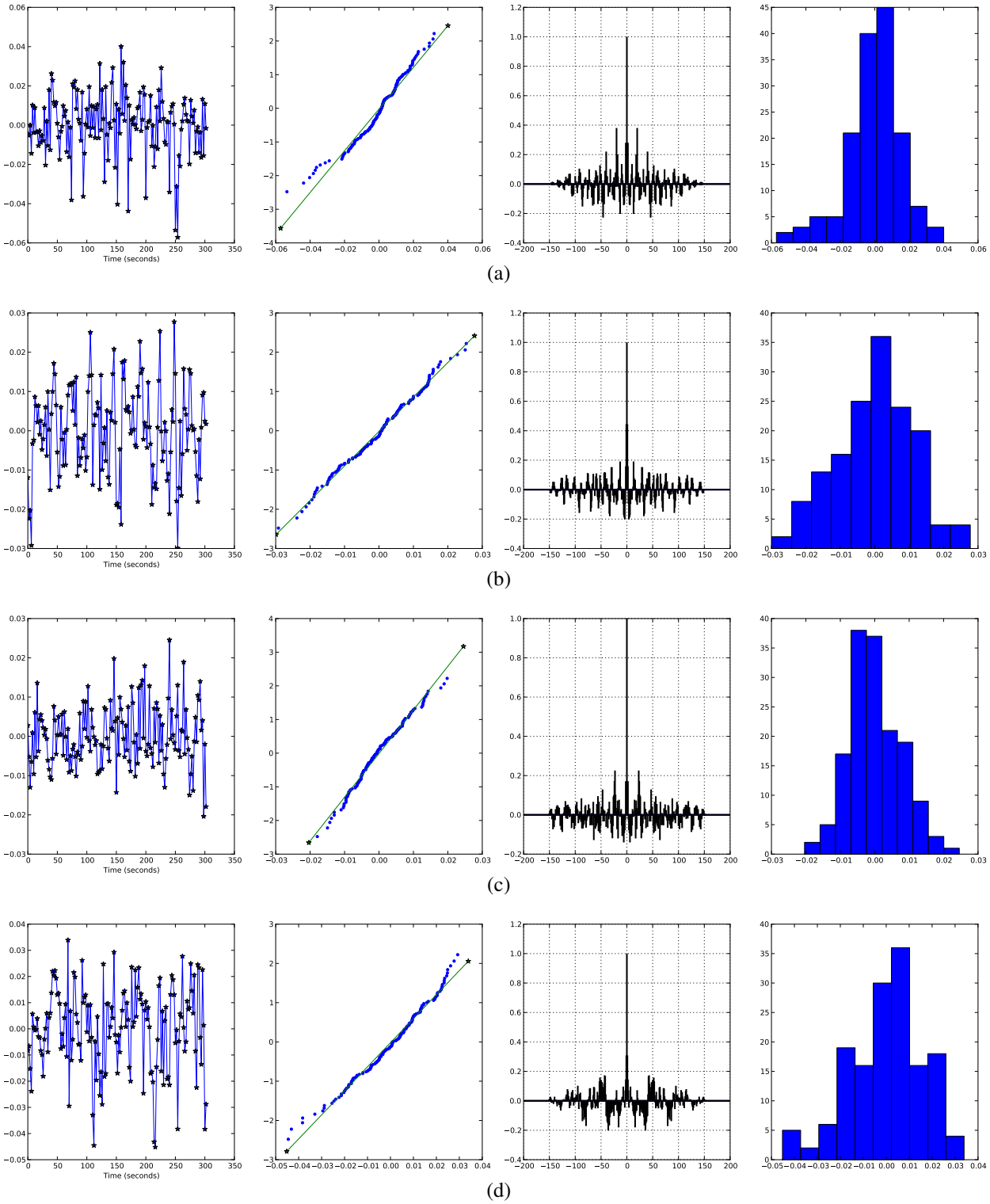


Figure 4.3: Q-Q Plots of resting state data, after the de-trending

[43] have demonstrated that this component is prevalent, and may in fact be an inherent characteristic of FMRI. It has been reported that in some studies as many as half the voxels benefited from detrending ([31]). In a head to head comparison, [42], showed that in most cases subtracting off a spline worked 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. Unfortunately no method will perfectly remove noise, and no method will leave the signal untouched.

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.

Problematically, after removing the DC component of the signal, by definition the signal will have a median near zero. Unfortunately 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. Because most of the BOLD signal is above baseline, after removing the spline the BOLD resting state will be below 0%. This reduces the ability of an algorithm to learn. One method of accounting for this is to simply add a DC gain model parameter. Like all the other model parameters, with enough measurements, a viable parameter would fall. Yet adding another dimension increases the complexity of the model, when the parameter is relatively easy to estimate by visual inspection. In this work a simpler approach was used. To determine the DC gain 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. Other methods may work better, however the MAD worked well, as Figure 5.4 and Figure 5.7 show.

$$y_{\text{gain},0:K} = 1.4826\text{median}_{i=0:K}(y_i - \text{median}(y_{0:K})) \tag{4.5}$$

A serious concern when adding and subtracting arbitrary values to real data is whether this will create false positives. This is a legitimate concern; however, a boosted response does not effect how well the BOLD model predicts the actual measurements.

4.3 Particle Filter Settings

There are quite a few options when using a particle filter; those options will be discussed in this section.

4.3.1 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 regions deemed active by the General Linear Model; and most other research (including [41]) used these results as the source for their priors. The one exception is [3], which came to a extremely different distributions. For a particle filter, the choice of a prior is the single most important design choice. A very wide prior will require more particles to be sufficiently dense, a very thin (low variance) prior may miss the true parameters. Consequently, for this work it was natural to use priors that will give results consistent with previous works, [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.

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 0.1 second impulse only reaches half a percent, as Figure 4.4 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 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 4.5).

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

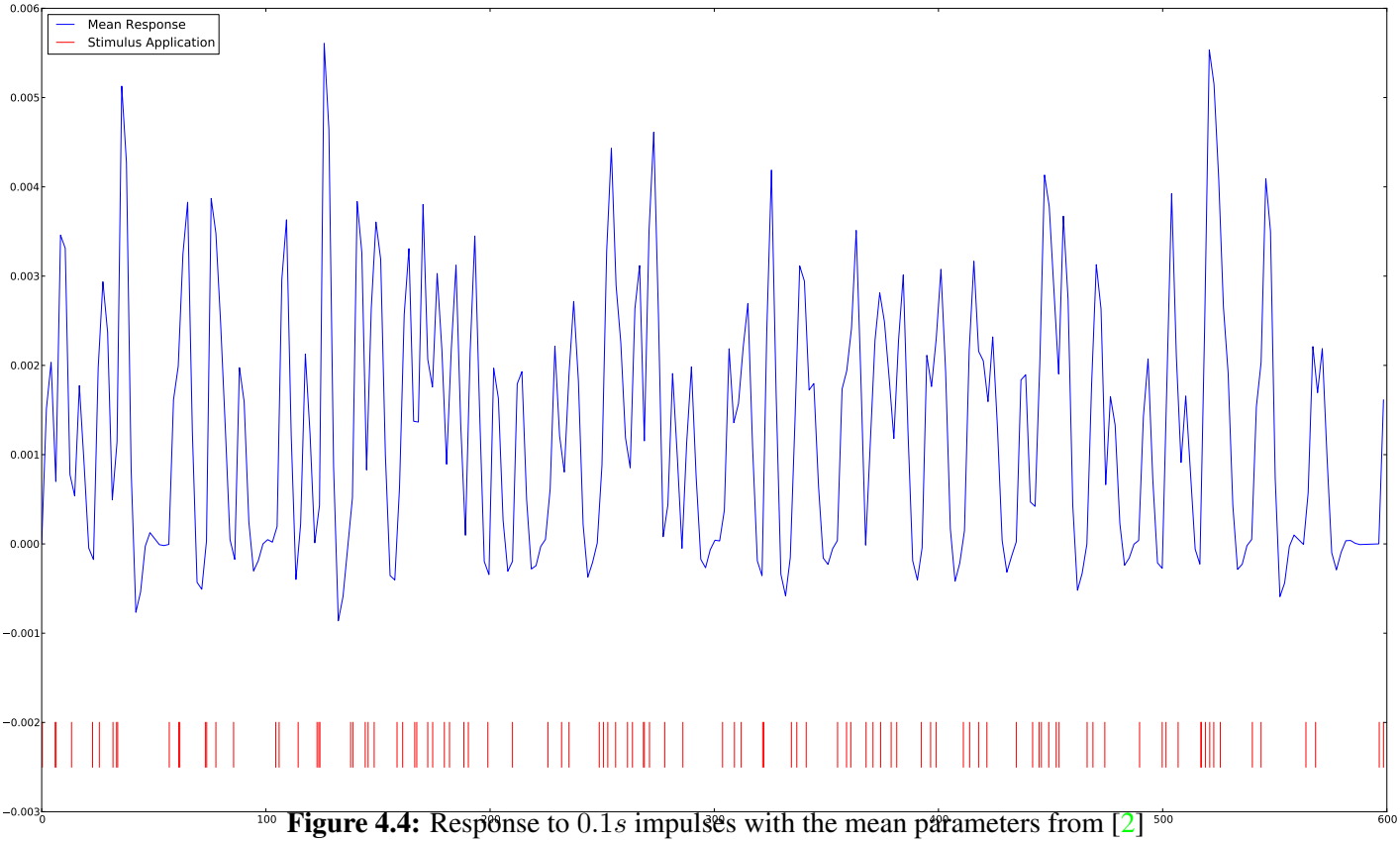


Figure 4.4: Response to $0.1s$ impulses with the mean parameters from [2]

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 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 was set to 16,000, but after resampling, the number of particles was dropped to 1,000. Typically during the first few measurements the variance dropped precipitously because most particles were far from a solution. 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.

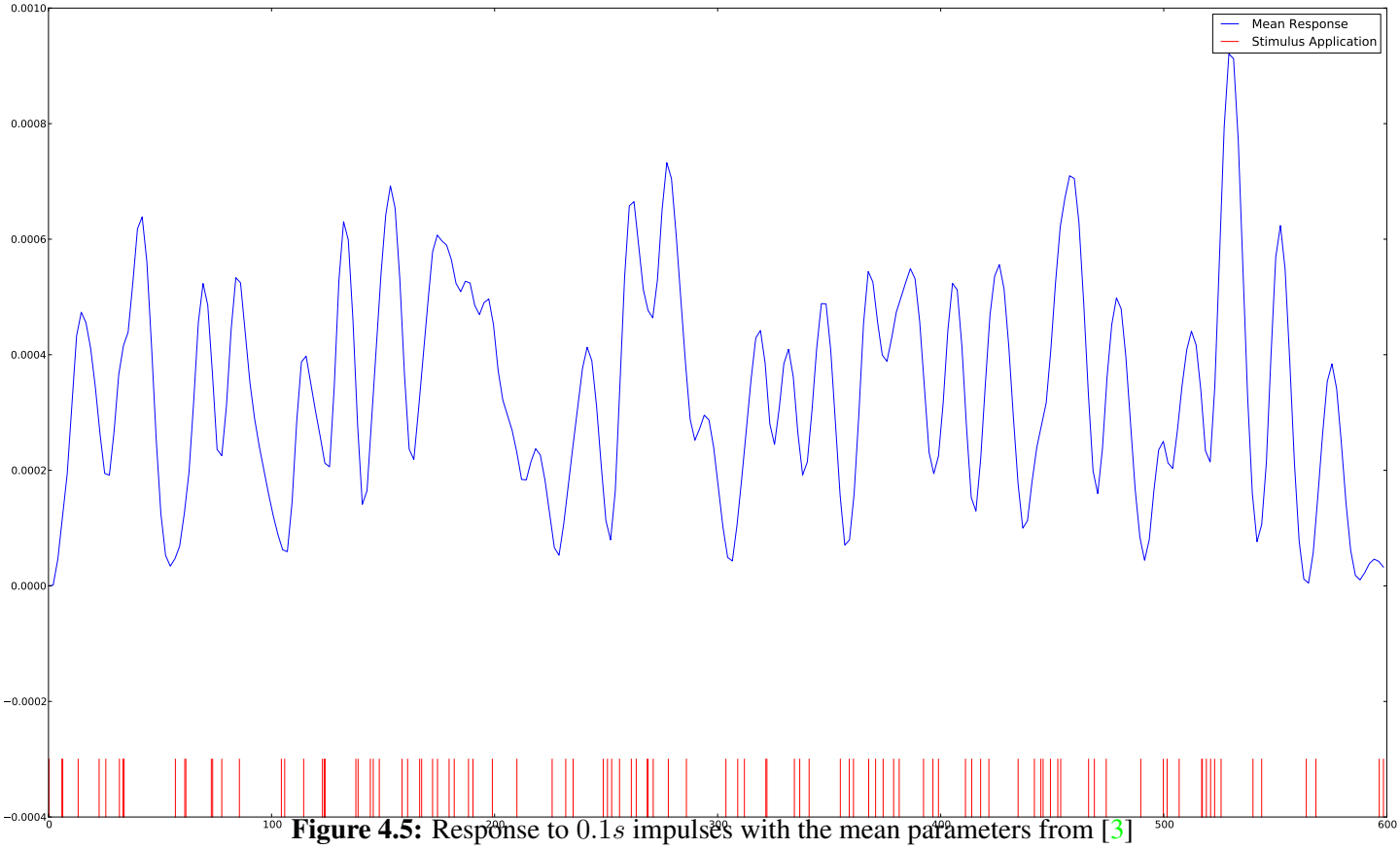


Figure 4.5: Response to $0.1s$ impulses with the mean parameters from [3]

4.3.2 Resampling

The algorithm for resampling is described in Section 3.4. When regularizing, the Gaussian kernel is convenient, because it is simple to sample from and long tailed. As discussed in Section 3.4, as long as resampling is kept as a last resort, some over-smoothing doesn'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.

Re-sampling is a not strictly necessary, but increases the effectiveness of the particle filter by adjusting the support to emphasize areas of higher probability. Re-sampling is slow because it requires re-drawing all the particles. It also closes off avenues of investigation, and is designed to over-smooth to prevent overly thinning the support. For all these reasons, resampling was only performed when the N_{eff} dropped below 50 (for 1000 particles). As a measure against sharp drops in the N_{eff} caused by a large spike in error, resampling was only performed when two consecutive low (< 50) N_{eff}'s were found.

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.

4.3.3 Choosing $P(y_k|x_k)$

The choice of $P(y_k|x_k)$ is the second most important design decision, behind the prior. While the conventional choice for an unknown distribution is the Gaussian, there are reasons why it may not be the best in this case. As noted in Section 4.2.1, 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 repeatedly estimate y_k with large residual will quickly have weights near 0. Thus, a weighting function that exactly matches $P(Y(t)|X(t))$ will easily, and correctly throw out incorrect particles. The cost of choosing an overly broad distribution for this function is slow convergence. On the other hand, an overly thin distribution will lead to particle deprivation (all particles being zero-weighted). 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. Wider tailed distributions don't down-weight particles as fast; and converge more slowly (and perhaps more accurately). The Laplace distribution also has the benefit of a non-zero slope at the origin; which means that even it will distinguish between particles even near the origin.

After trial and error, for this work I chose a zero-mean Gaussian with standard deviation of $.005$. While I made some attempts to automatically set the standard deviation, results were often unpredictable. If the weight function and scale aren't fixed across voxels, very noisy time series with no actual signal converged 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 gave the best consistency.

4.3.4 Runtime

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 40 seconds (for 1000 particles, 1500 integration points and a Quad Core CPU). The size of local linearization steps are also crucial although going above 0.001 seconds per step is not recommended. In most cases millisecond resolution is fine; however, when generating simulated data I found that at times it was still not enough every once. 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. To prevent such events, 1500 integration points per second were used throughout the tests.

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. The choice of 20 seconds is arbitrary, but at the very least it gives a more optimized version of the prior.

Chapter 5

Simulation

Two levels of simulation are discussed in this section; first of single voxels, and second of a single slice (64x64 voxels). The single time series tests investigate the ability of particle filters to estimate parameters of the BOLD model in a noisy environment. Single voxel tests were performed eleven different times, each with a new noise realization. Repeating tests with different noise realizations demonstrates the particle filter's resiliance to noise and explores the variance of the model.

The slice simulation was performed using Physics-Oriented Simulated Scanner for Understanding MRI (POSSUM) which models noise realistically. POSSUM demonstrates the particle filter modeling the BOLD signal on a large scale. In the POSSUM simulation there were four discrete parameter sets driving the timeseries, although each voxel had a different tissue composition and a different noise realization. Therefore the POSSUM simulation was a good test of the applicability of particle filters for performing whole brain analysis.

Note that except for Section 5.1.1 the same stimulus sequence that was used in real data (Chapter 6) was used in the simulations. This sequence may be found in Section 6.1, but because of the availability of ground truth in simulations is not displayed in this section.

5.1 Single Time Series

Given the state-space equations for the BOLD signal, simulating a single time series was 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 the true signal. Finally a carrier level was added, since BOLD is typically measured as a % difference from the baseline. The particle filter algorithm immediately removes 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.

Once this noisy simulated time series was generated, the particle filter algorithm was run on the

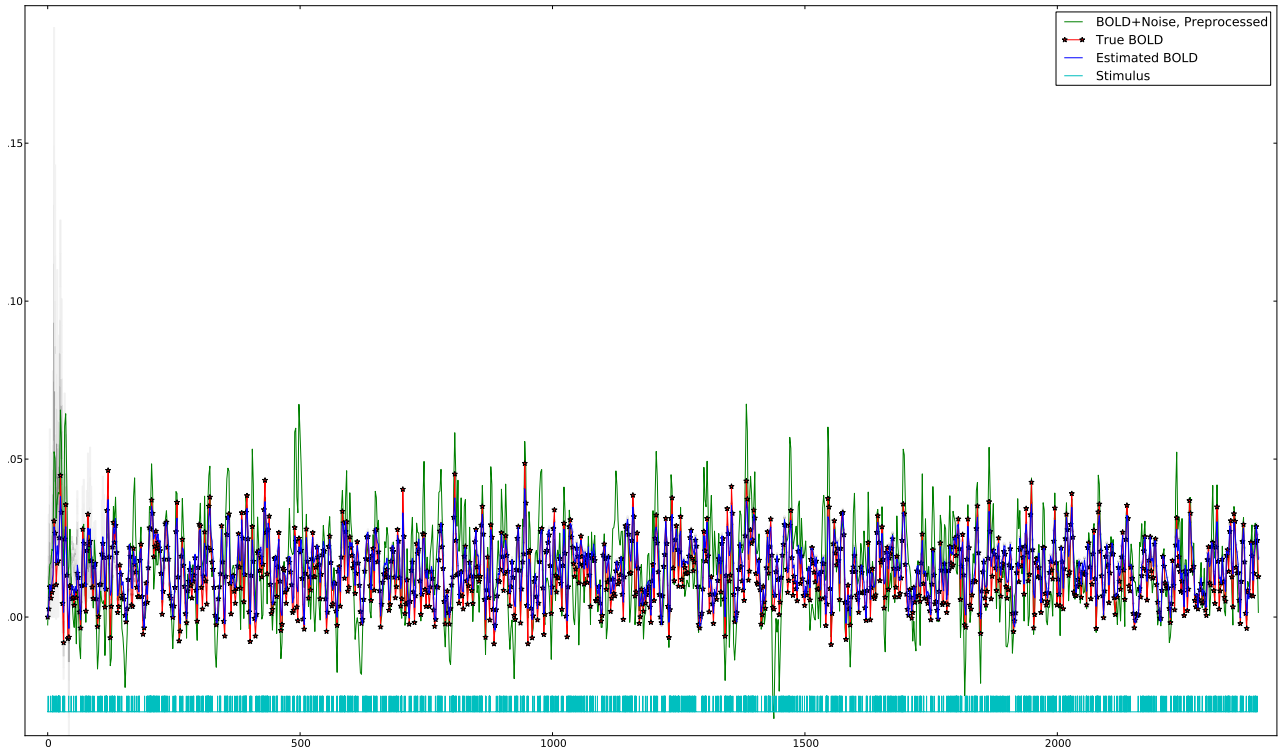


Figure 5.1: BOLD estimate converging for a very long FMRI run. Darker bars indicate bins with more regions.

single time series image. Five sets of tests determine the power of the particle filter in modeling. The first test demonstrates the particle filter on a very long FMRI run, and discusses the inherent learn-ability of the BOLD model (Section 5.1.1). The next two (Section 5.1.2 and Section 5.1.3) demonstrate the ability of the particle filter to find the most likely set of parameters/state variables over the course of a run identical to the real run in Chapter 6. The last two tests (Section 5.1.4 and Section 5.1.5) investigate the problem of false-positives. As the first round of tests show, given the presence of an underlying BOLD signal, the particle filter is excellent at finding the most probable cause of the observed signal. As discussed in Section 5.1.4 and Section 5.1.5, even when there is no underlying signal, the particle filter may still converge. Because of this problem, it was necessary to investigate methods of identifying false positives.

5.1.1 Ideal Analysis

To begin the single voxel simulation; I generated a signal using the following parameters: $\{\tau_0 = 1.45, \alpha = 0.3, E_0 = 0.47, V_0 = 0.044, \tau_s = 1.94, \tau_f = 1.99, \epsilon = 1.8\}$. These same parameters were used throughout this chapter. Noise was generated based on measurement noise (σ_y) of 0.001 and drift standard deviation (σ_x) of 0.0005. The measurement noise as well as the steps of the

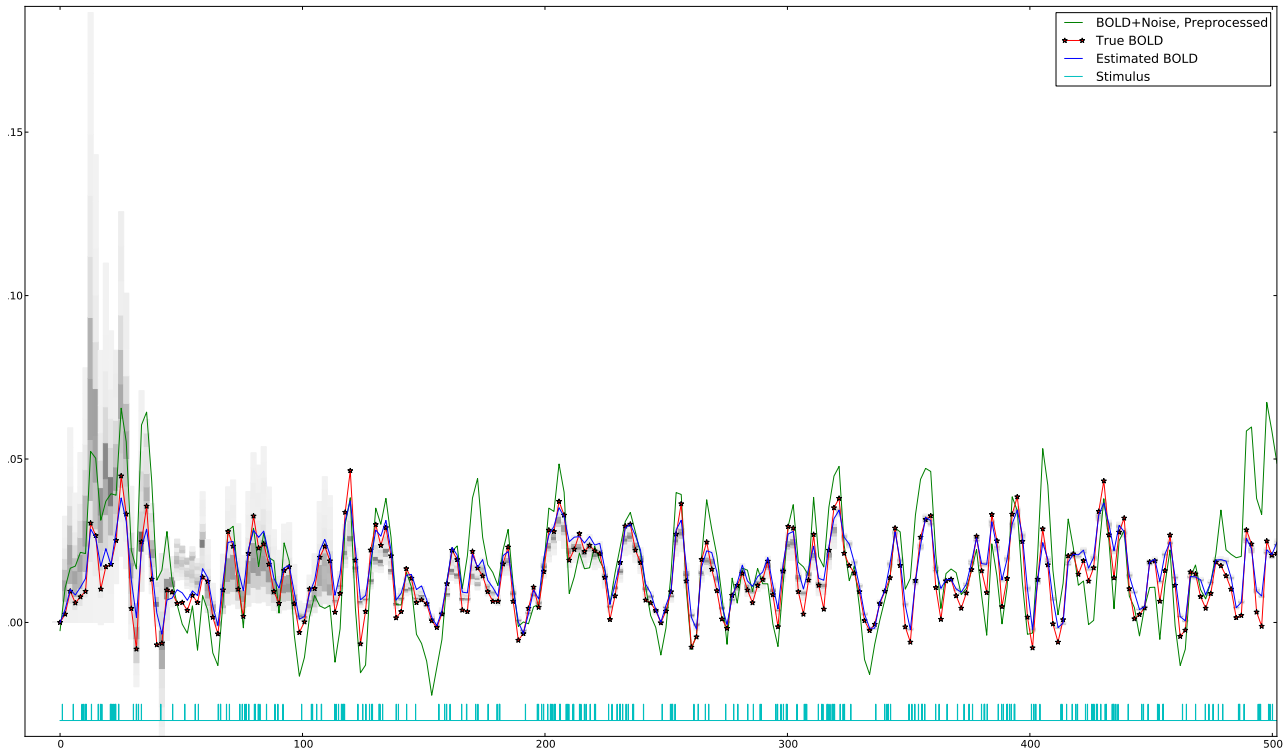


Figure 5.2: BOLD estimate converging for a very long FMRI run, first 500 seconds. Darker bars indicate bins with more particles.

	τ_0	α	E_0	V_0	τ_s	τ_f	ϵ
τ_0	0.0004334	5.2e-05	-6.95e-05	3.3e-06	0.0001628	-2e-07	0.0001798
α	5.2e-05	7.9e-06	-6.4e-06	3e-07	1.04e-05	-1.92e-05	2.58e-05
E_0	-6.95e-05	-6.4e-06	1.9e-05	-9e-07	-4.11e-05	-3.24e-05	-3.92e-05
V_0	3.3e-06	3e-07	-9e-07	1e-07	1.1e-06	9e-07	1e-06
τ_s	0.0001628	1.04e-05	-4.11e-05	1.1e-06	0.0001589	0.0001518	7.88e-05
τ_f	-2e-07	-1.92e-05	-3.24e-05	9e-07	0.0001518	0.0002966	-2.34e-05
ϵ	0.0001798	2.58e-05	-3.92e-05	1e-06	7.88e-05	-2.34e-05	0.0001966

Table 5.1: Covariance matrix of the parameters at the end of Figure 5.1.

	τ_0	α	E_0	V_0	τ_s	τ_f	ϵ
τ_0							
α	0.889884						
E_0	-0.7661395	-0.5230723					
V_0	0.6244049	0.4239271	-0.7964774				
τ_s	0.6204843	0.295425	-0.7481253	0.3440421			
τ_f	-0.0004259	-0.3966881	-0.4314174	0.1962954	0.6990775		
ϵ	0.6158116	0.6558179	-0.641348	0.2846632	0.4458142	-0.097079	

Table 5.2: Correlation of parameter estimates at the end of Figure 5.1.

drift were taken to be Gaussian. The actual signal delivered into the particle filter was the result of preprocessing to remove drift, as described in Section 4.2.

To test how well the particle filter would do with plenty of data, and determine the inherent variance in the model parameters, a very long simulation with randomly generated impulse stimuli was created. The preprocessed timeseries and the final estimate are shown in Figure 5.1; the final covariance matrix of the parameters is in Table 5.1. Note that even though the BOLD response converged well (Figure 5.1), the parameters still have significant correlation (Table 5.2). Based on the histograms in the first 500 seconds, the parameters converged to their final values well before the end. Although this is only a single test, the correlation (Table 5.2) of the parameters indicates significant parameter indeterminability. When the input consists entirely of impulses, the best parameters are not one particular set, but a spectrum. Note that the correlation is in parameters whose priors are completely independent. It is possible that varying the type of input could give improved results, although informal tests did not show significant difference. In spite of the noisy input (green line in Figure 5.1), the estimates of the BOLD were actually very close to the true (noise-free) BOLD signal.

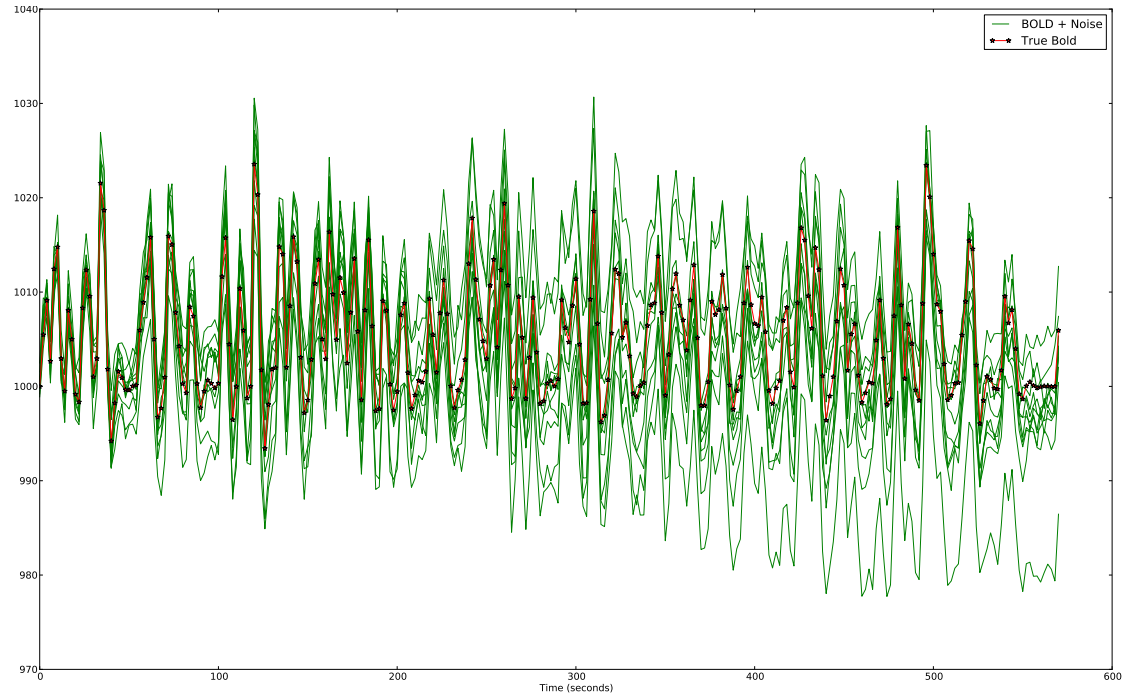


Figure 5.3: Test Signals with low noise compared to the clean signal.

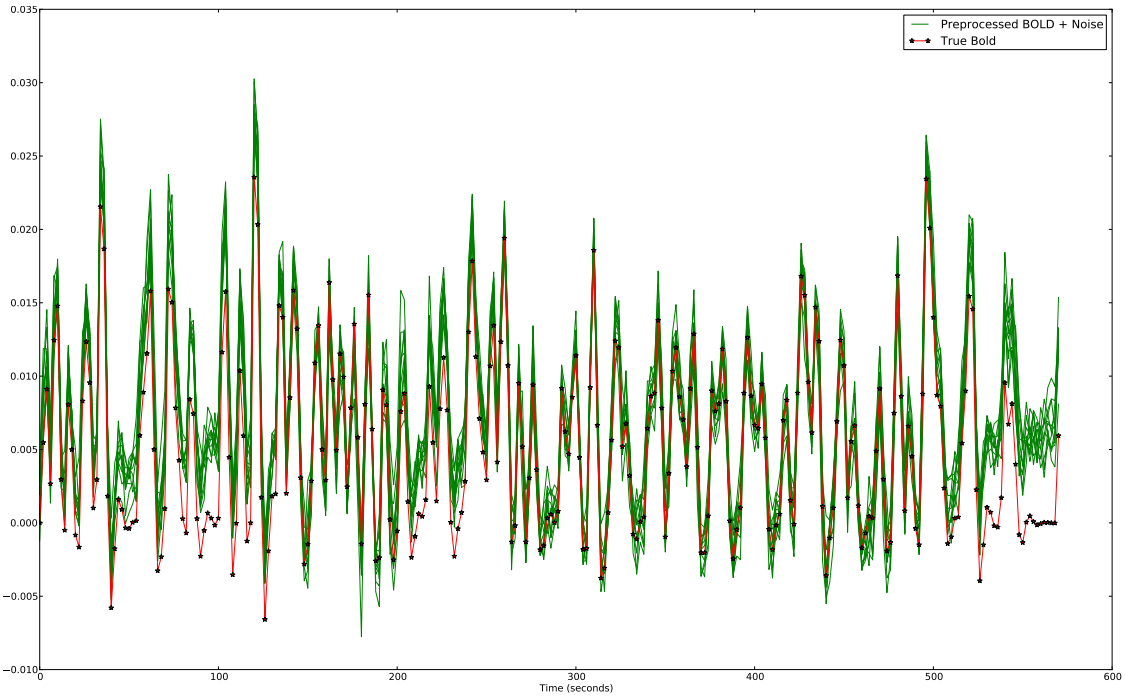


Figure 5.4: A comparison of the preprocessed signals for the low noise case. The noisy input to the actual particle filter algorithm.

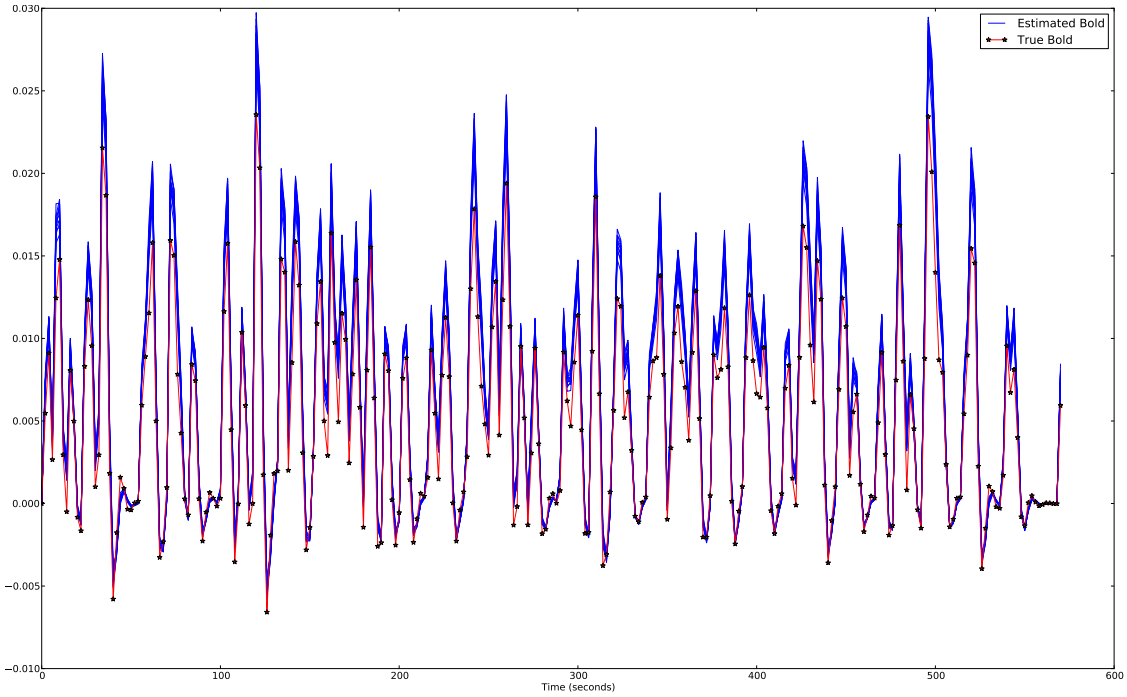


Figure 5.5: A comparison of the fitted signals for the low noise case.

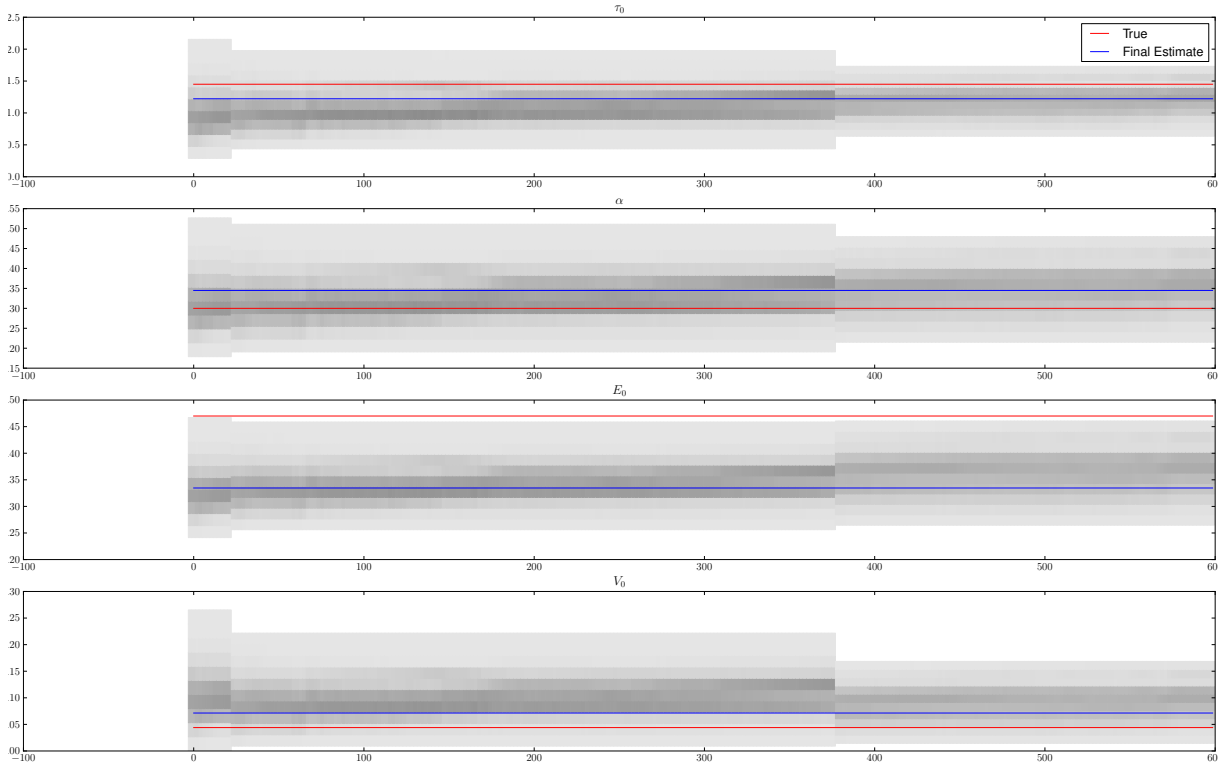
5.1.2 Simulation with Low Noise

The following tests all use less data (shorter sequences), on par with the average length of an FMRI session. For this low noise case ($\sigma_y = 0.001,\ \sigma_x = .0005$), the eleven realizations are shown in Figure 5.3. The bias introduced into the signal by preprocessing could have some effect on the resulting fit; thus the preprocessed signal is compared to the true BOLD signal in Figure 5.4. Overall, Figure 5.4 shows that the preprocessing was effective at removing trends, although the spline did struggle to match the signal torward the end. This slight drift effect was caused by the final median being well above the base level. At several time points the particle filter successfully filtered the input (Figure 5.5). For instance, in the last 30 seconds the estimates stay flat in spite of the preprocessed data drifting off. By this point, the algorithm had converged sufficiently to prevent such inexplicable movement. A similar circumstance occurs at around 100 seconds in. A combination of noise and preprocessing biased the results toward a peak signal above the true peaks. The final parameter sets are shown in Table 5.3.

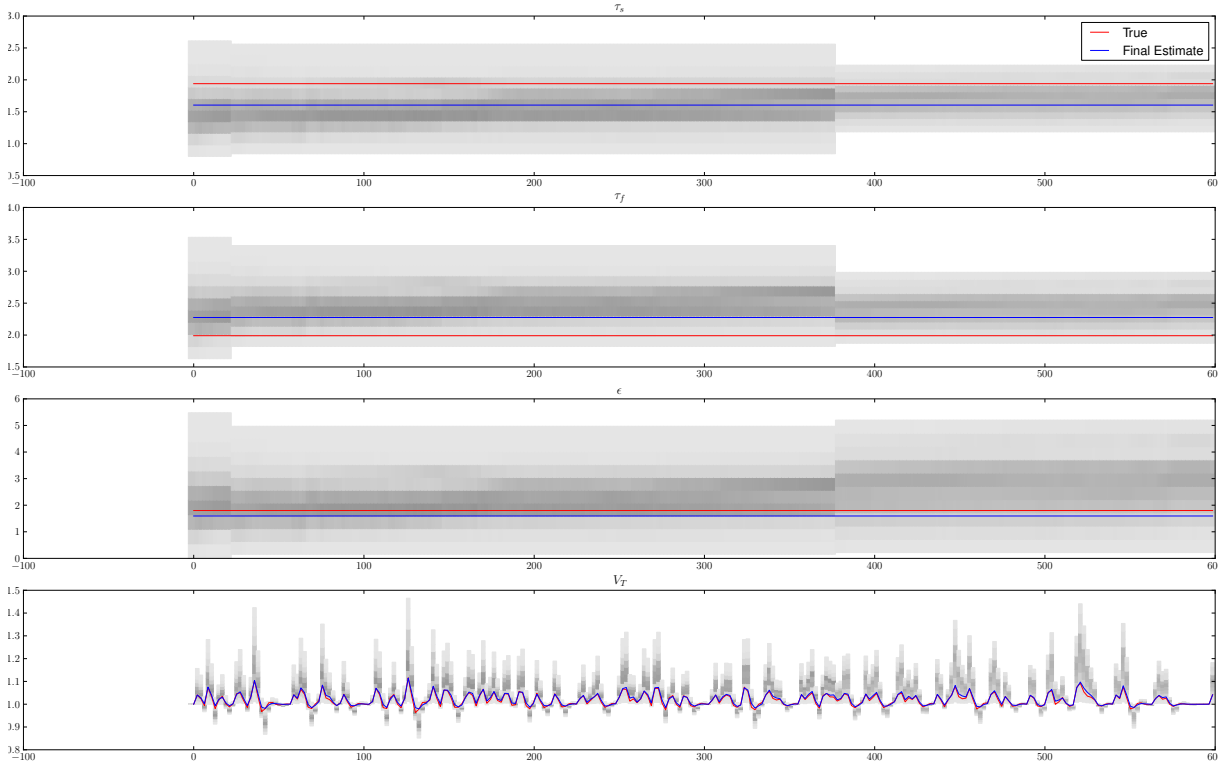
Note that throughout the results, unless otherwise specified, the term *residual* will be defined as the square root of the mean squared residual,

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

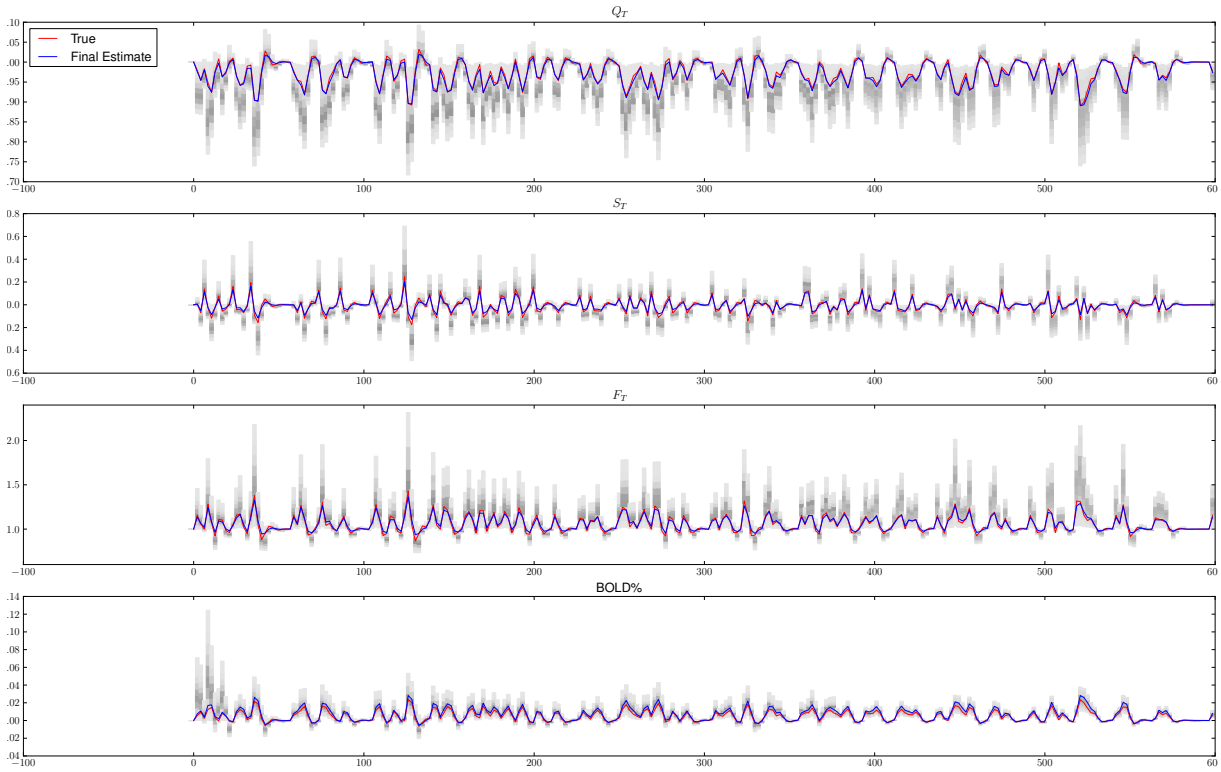
where $\hat{y}_k$ is the estimated output at time k and y_k is the preprocessed output sampled at time k.



(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.

Figure 5.6: Convergence of the first run from Table 5.3. The bars represent a histogram, where darker bars indicate more particles in that bin.

τ_0	α	E_0	V_0	τ_s	τ_f	ϵ	$\sum \tau$	Residual	Error
1.45	0.3	0.47	0.044	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 5.3: Estimated Parameters on 11 different runs with low noise. First row is the true value, and last is the average.

Note the subtle distinction between this and what will be termed the *error* which is defines as

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

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

There are a few important results in the final parameter estimates (which are the mean of the particle filter's posterior distribution). First, the time constants vary greatly across runs, yet the sum of the individual time constants (τ_f, τ_s and τ_0) were more consistent. In general the time constants fell 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 that other parameters compensated. It is also possible that the output is insensitive to small differences in the time constants. The convergence properties of the first run in Table 5.3 demonstrates the migration of parameters through the run. In spite of the significant differences in parameter estimates, the estimated BOLD consistently performed well (Figure 5.5).

5.1.3 Simulation with High Noise

For the high noise simulation, the exact same procedure was followed as in Section 5.1.2 except that σ_y and σ_x were set to 0.01 and 0.005, respectively. This is an order of magnitude higher than the previous tests, and indeed the noise appears to dominate the output, as Figure 5.7 shows. The results of the particle filter for each of the eleven runs are shown in Figure 5.8. The noise and

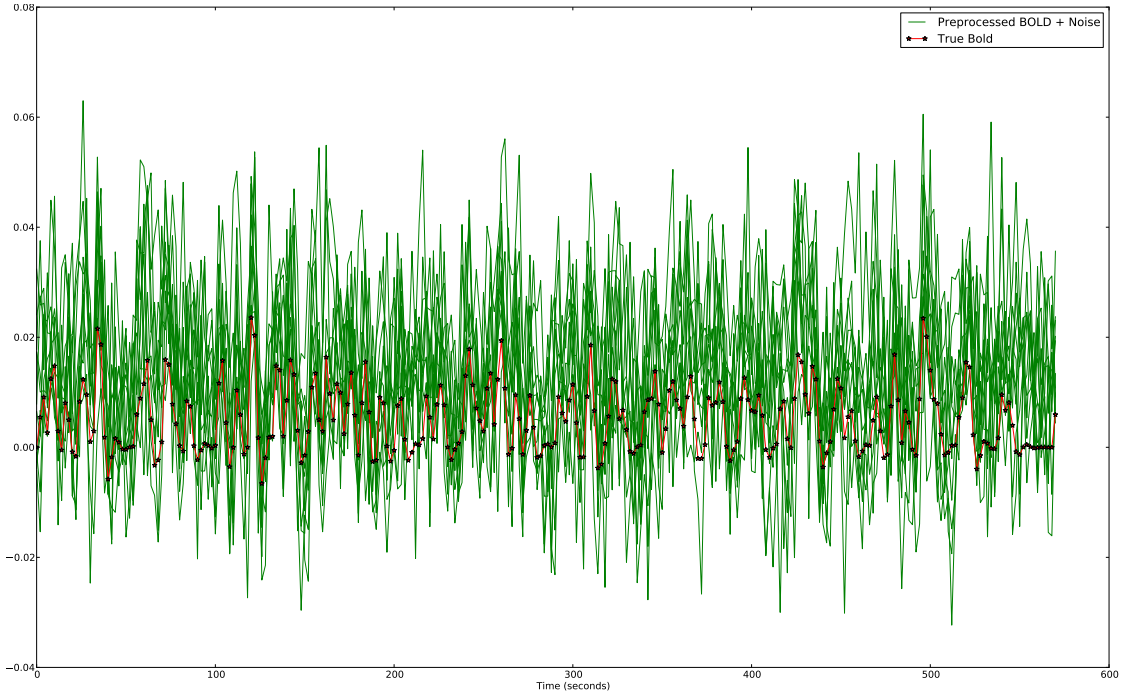


Figure 5.7: A comparison of the preprocessed signals for the high noise case.

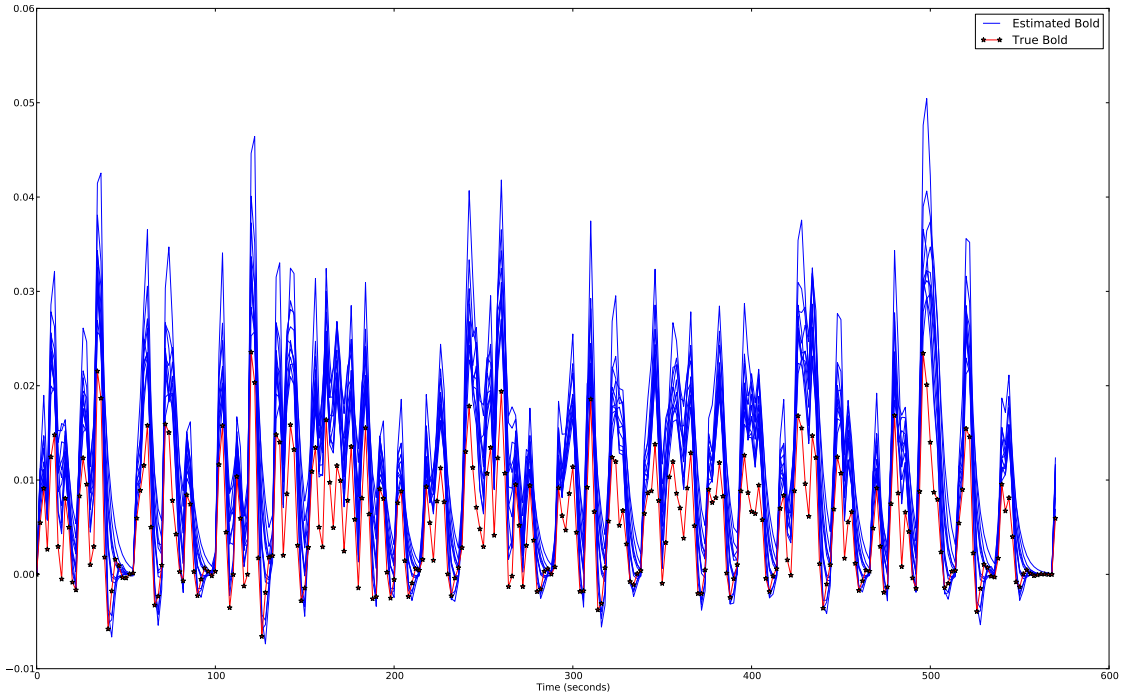


Figure 5.8: A comparison of the fitted signals for the high noise case.

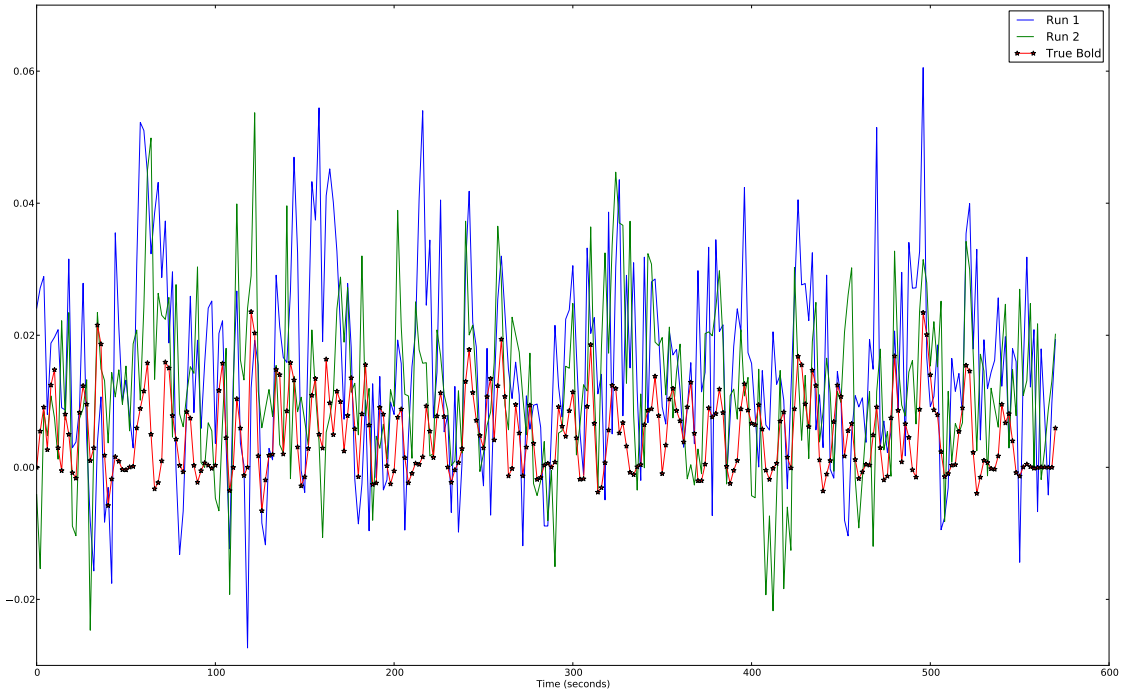


Figure 5.9: Two particular preprocessed noise realizations for the high noise case.

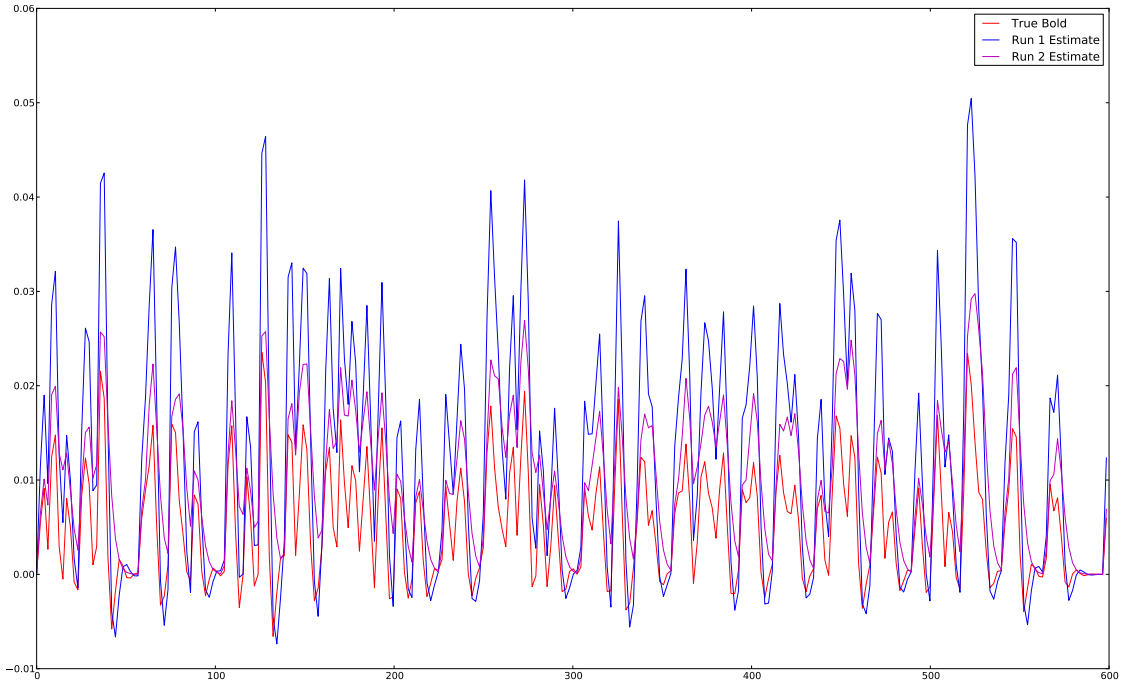


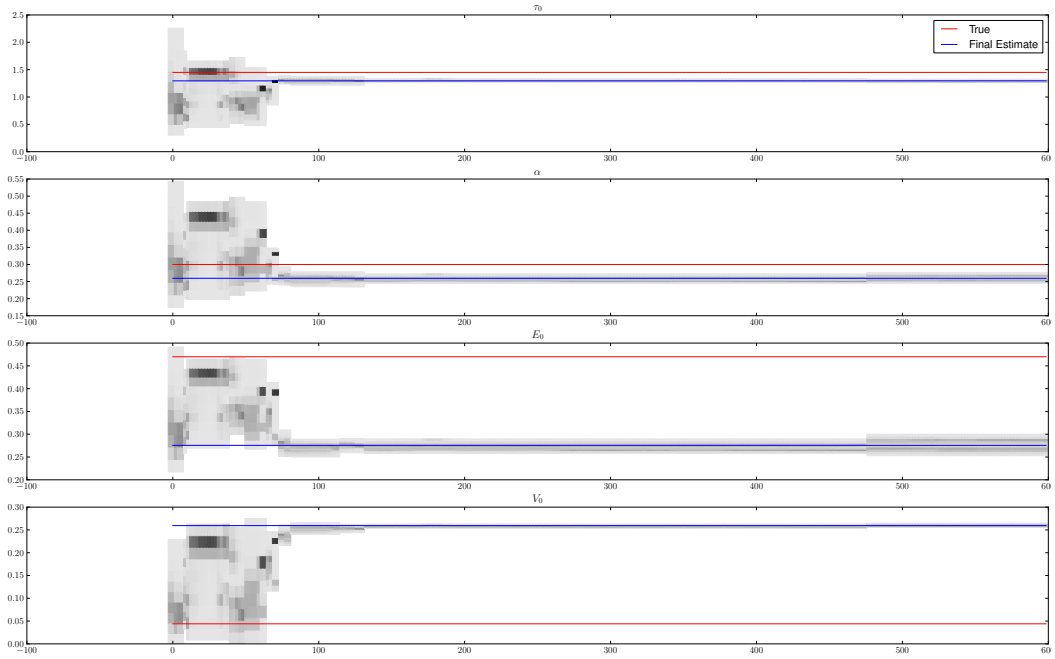
Figure 5.10: The results for the noise realizations shown in Figure 5.9.

τ_0	α	E_0	V_0	τ_s	τ_f	ϵ	$\sum \tau$	Residual	Error
1.45	0.3	0.47	0.044	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

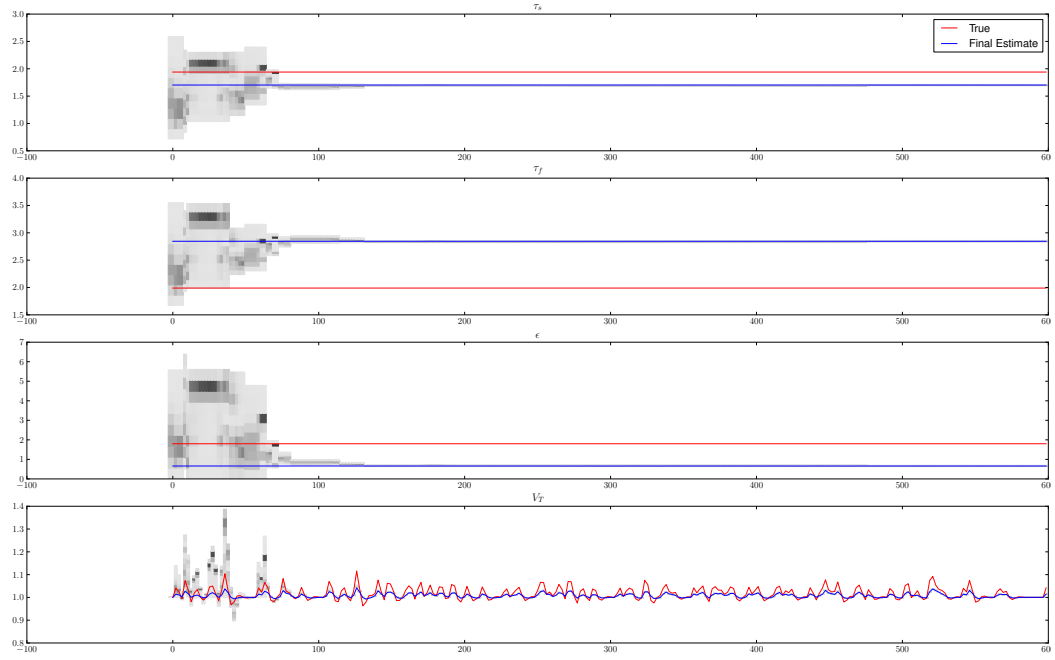
Table 5.4: Estimated Parameters on 11 different runs with high noise. First row contains the true parameters, last row contains the mean. The red row is Run 1 and the blue row is Run 2 from Figure 5.9 and Figure 5.10, respectively.

preprocessing again led the estimates to higher peak activation levels, and the subtleties of different time constants are being lost in the noise.

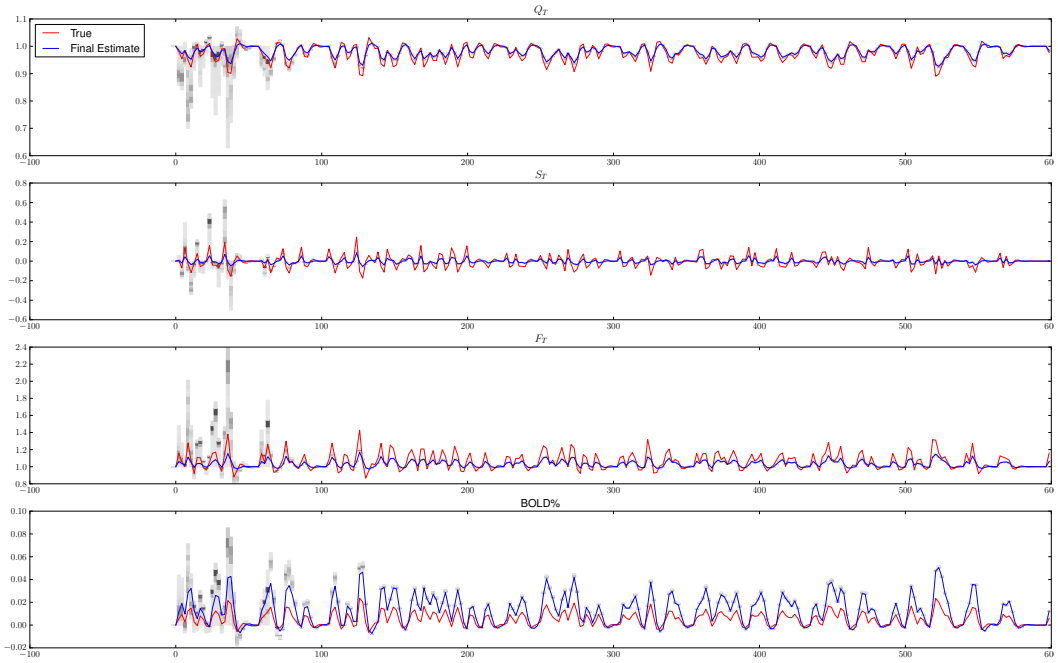
Figure 5.9 shows two of the runs in more detail. There appears to be more drift than the 20 measurements per knot could fit, which explains the prolonged increase at 170 seconds in Run 1; although such areas permeate the preprocessed signals. Interestingly, Run 1 and Run 2 emphasize different aspects of the signal. Run 2 had a much better match to the peaks, when compared to the true signal, yet Run 1 matched the post-stimulus undershoot better. Table 5.4 shows the error for all eleven runs, and highlights the two runs analyzed in Figure 5.11 and Figure 5.12.



(a) τ_0, α, E_0, V_0, Run 1

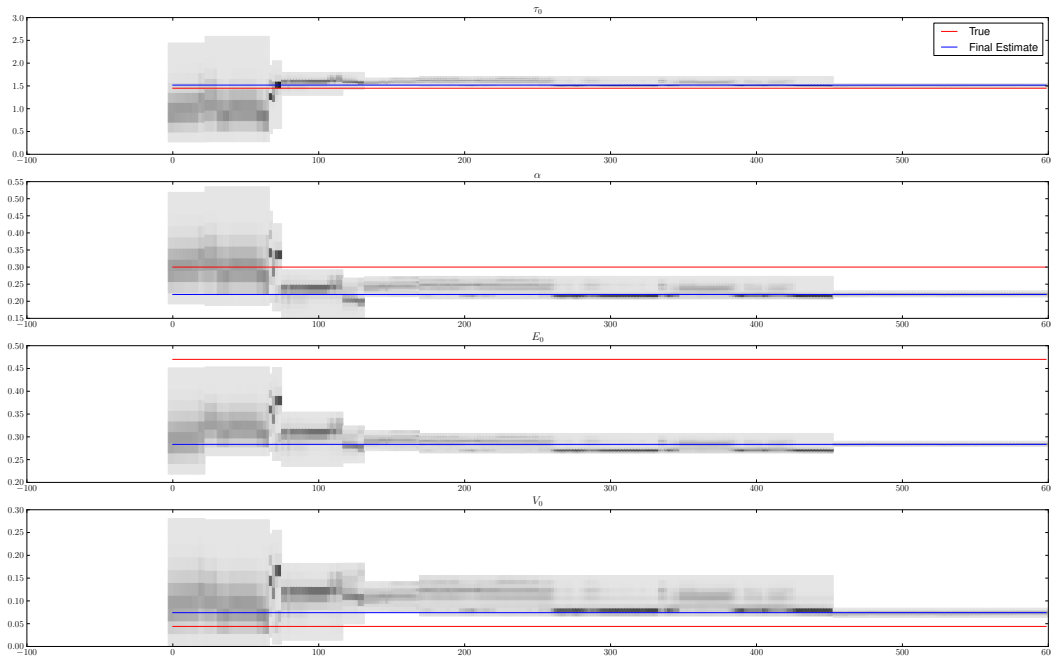


(b) τ_s, τ_f, ϵ, v, Run 1

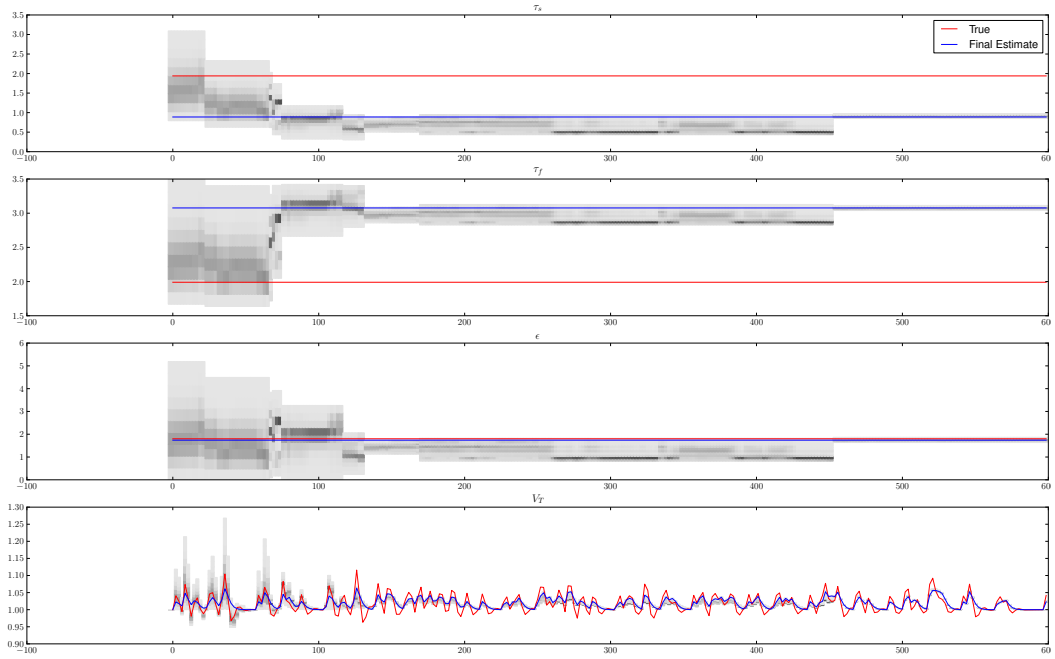


(c) q, s, f, $BOLD$, Run 1

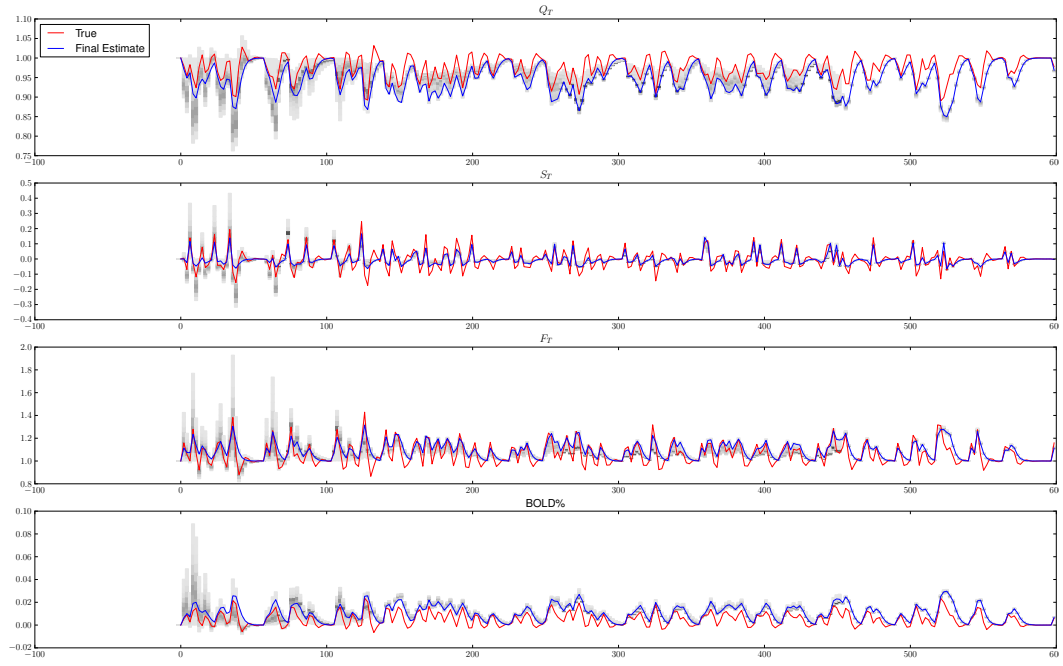
Figure 5.11: Converging histogram for parameters during run 1, as in Figure 5.9.



(a) τ_0, α, E_0, V_0, Run 2



(b) τ_s, τ_f, ϵ, υ, Run 2



(c) q, s, f, $BOLD$, Run 2

Figure 5.12: Converging histogram for parameters during run 2, as in Figure 5.9.

The particles converged much faster when more noise was present (Figure 5.6 vs. Figure 5.11, Figure 5.12). This also caused significantly more resampling which is the explanation for the perceived jumps in the histograms. Clearly the additional noise resulted in much more sporadic results (Table 5.4). This is often the result when the particle filter converges too fast, in this case the result of the weighting function's variance being smaller than the measurement noise (0.005 vs. 0.01). The error clearly suffers due to this effect (Table 5.4). Note that neither Run 1 or Run 2 estimated the underlying state well (5.11(c) and 5.12(c)), whereas in the previous test the particle filter was extremely successful in this area (5.6(c)).

5.1.4 Pure Noise, Low magnitude

The next two single-voxel tests forced the particle filter to attempt to learn a noise-only time series. In this test the noise was the same as that from the Section 5.1.3, $\sigma_x = 0.01, \sigma_y = 0.005$. The stimulus neuronal efficiency (ϵ) was set to 0, in effect simulating a brain region with no response to the stimuli. This test was used to determine how the output of a pure noise timeseries is different from that of a simple noisy signal (as in the previous two sections). The preprocessed signals are shown in Figure 5.13 and line fit for each run is shown in Figure 5.14.

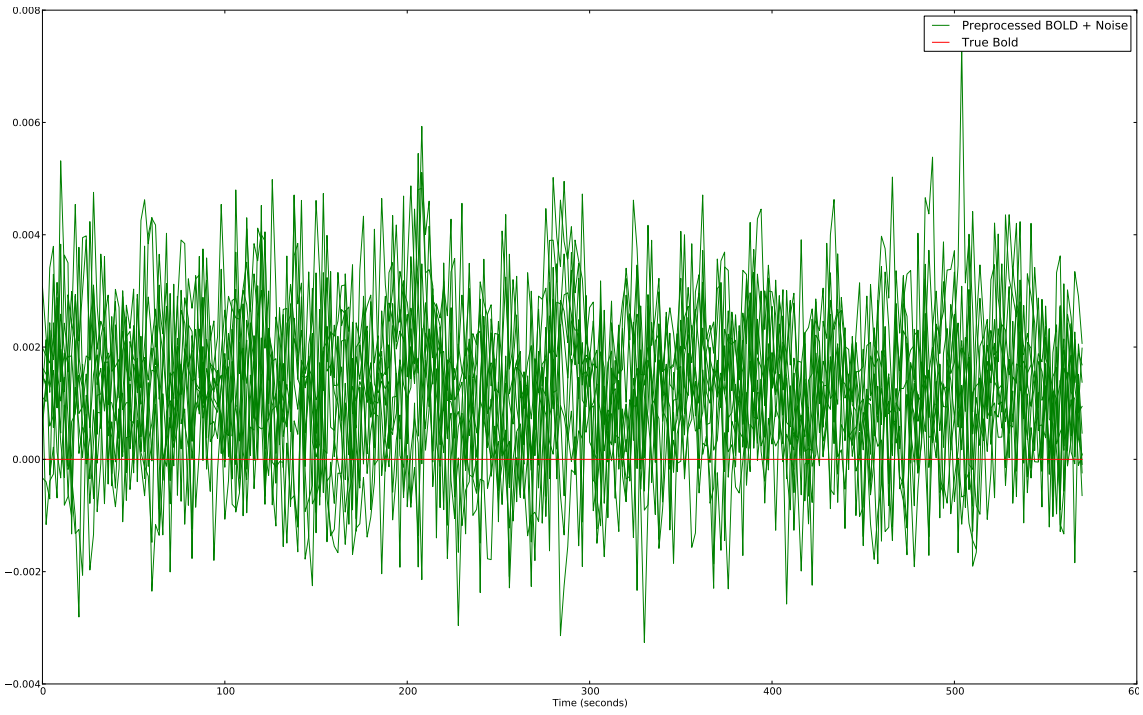


Figure 5.13: Comparison of the preprocessed signals for the low noise signal-free case ($\sigma_y = 0.01, \sigma_x = 0.005$).

τ_0	α	E_0	V_0	τ_s	τ_f	ϵ	Residual
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 5.5: Estimated Parameters on 11 different runs with low noise and no signal present.

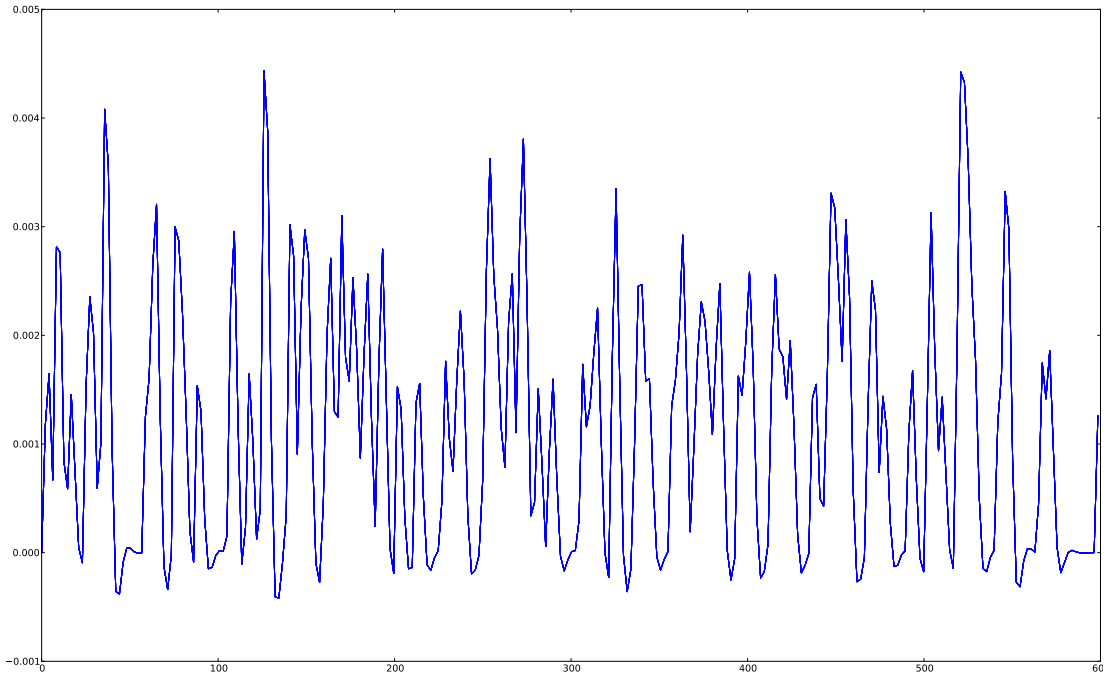


Figure 5.14: Fits to the non-active, low noise signal. Note that the line is thick because all the estimates overlap. All 11 fitted lines.

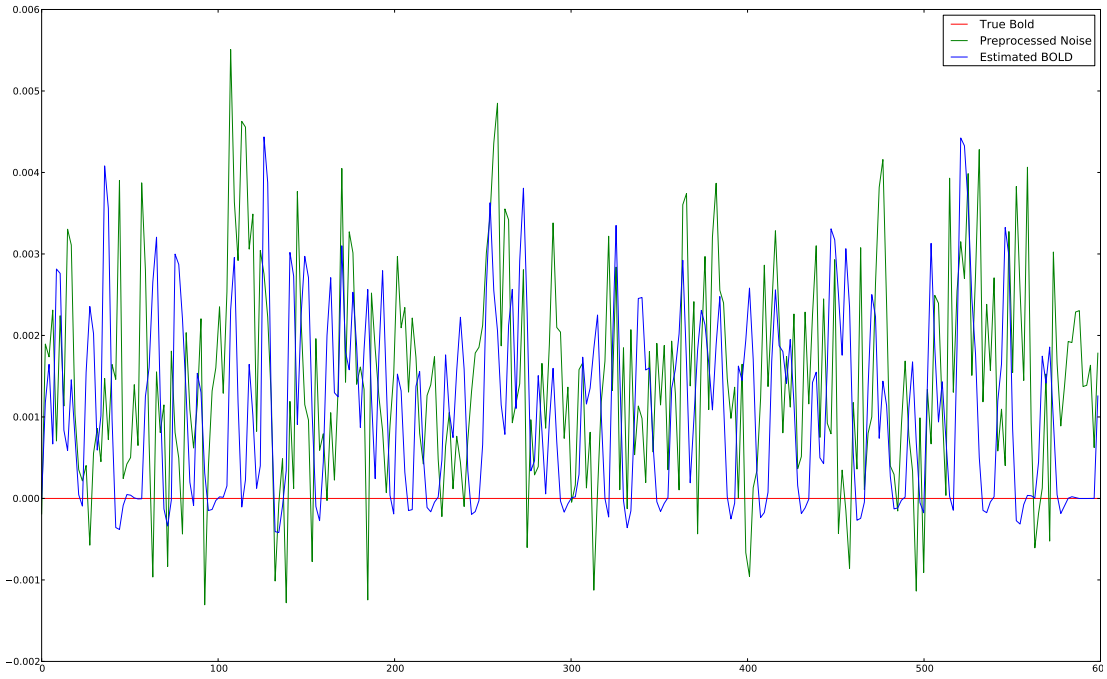
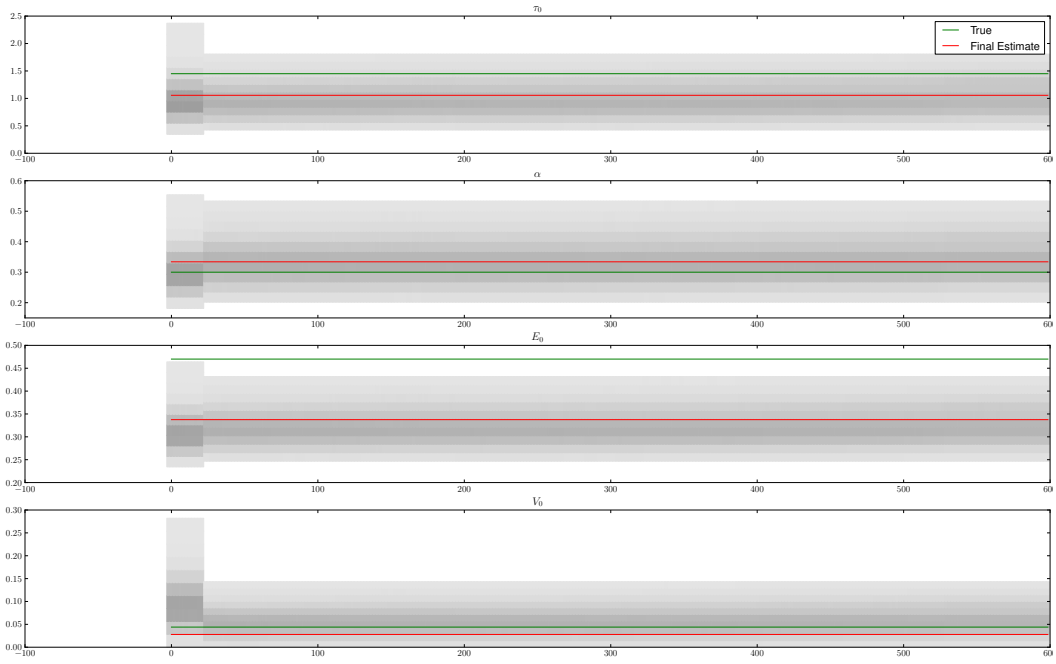
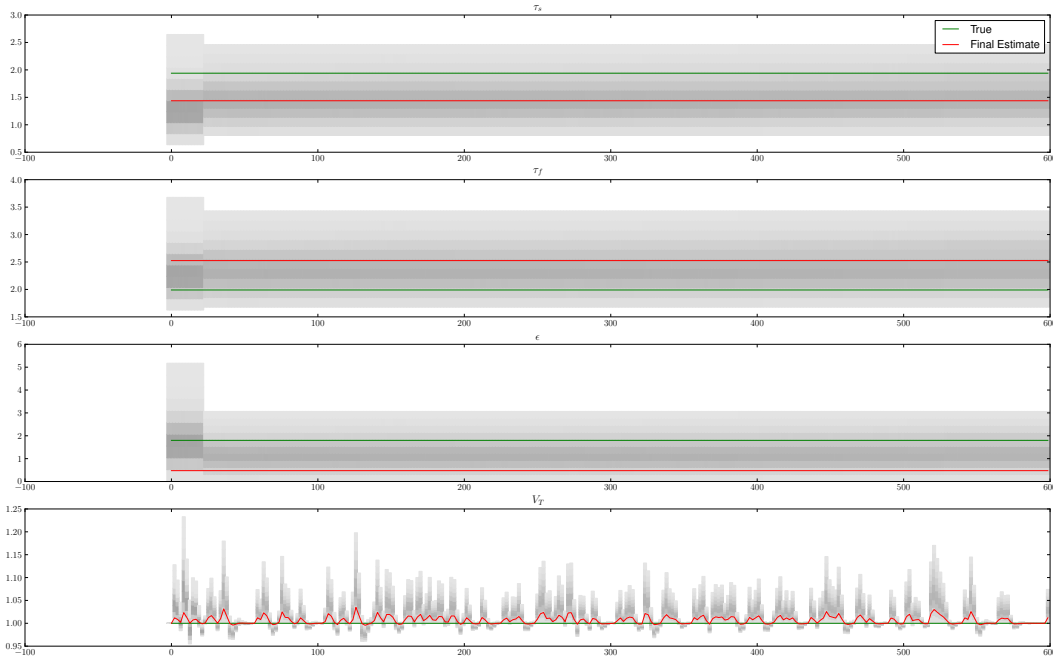


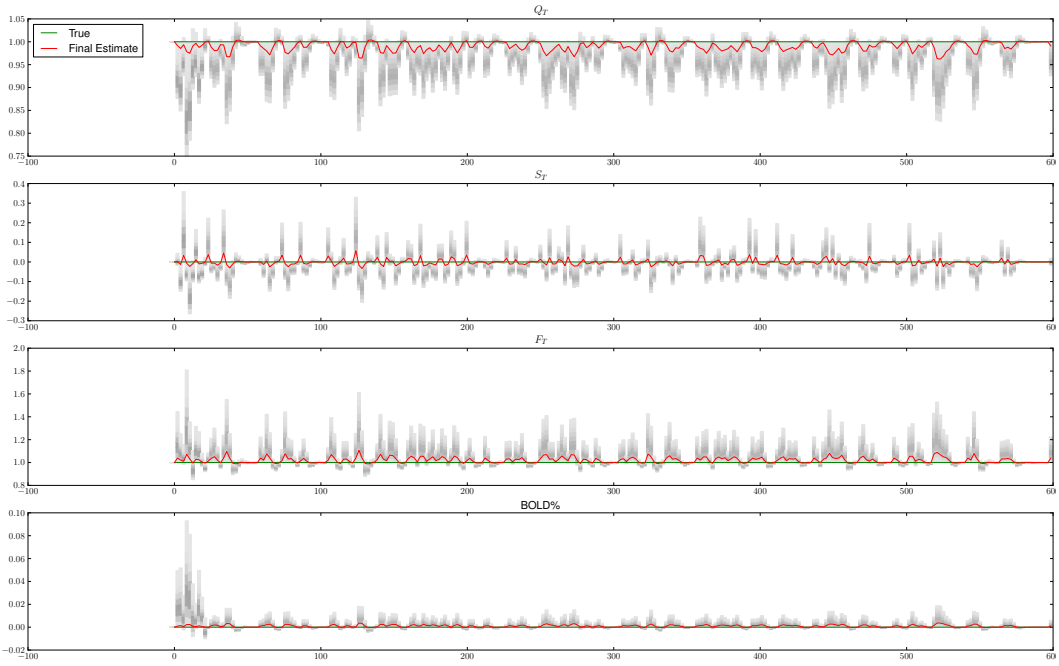
Figure 5.15: Fit from a single particle filter run, with the noise input.



(a) τ_0, α, E_0, V_0



(b) τ_s, τ_f, ϵ, v



(c) q, s, f, $BOLD$

Figure 5.16: Converging histogram for parameters when the signal consists purely of low level noise. Same run as Figure 5.15

The data shows that the parameters did really not converge Figure 5.16. The peaks never even reached 1% difference (Figure 5.13) so the signal stayed well within the range of 0.005, the standard deviation of the weighting function. Note that the residuals were actually lower than the residuals in the low noise simulation from Section 5.1.2 and the parameter estimates were extremely consistent across 11 runs. The low residual was caused by the overall signal being significantly smaller than any previous simulation. From these results it would seem that lack of convergence could be a differentiating factor from a signal with an actual signal; however, the next section casts doubt on this possibility.

5.1.5 Pure Noise, High Magnitude

To determine how the particle filter responds to active, yet unrelated portions of the brain, this section repeats the test of Section 5.1.4 with much higher noise peaks. To simulate this case another pure noise signal was generated using σ_x of 0.1 and σ_y of 0.05.

As before, the convergence all follows a similar path, leaving almost no variance in the estimated time series (Figure 5.17). Interestingly the algorithm suffered from almost constant particle deprivation, meaning that the heuristic for rescuing the particle filter from particle deprivation, discussed in Section 4.3.2, 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 in good data if the prior distribution does not cover a solution.

Because of the preprocessing steps, the pure noise signal can look markedly like a real signal (Figure 5.18). The preprocessing causes the particle filter to converge to a non-zero response in spite of the fact that the input does not correlate with the stimuli in any way (Figure 5.17).

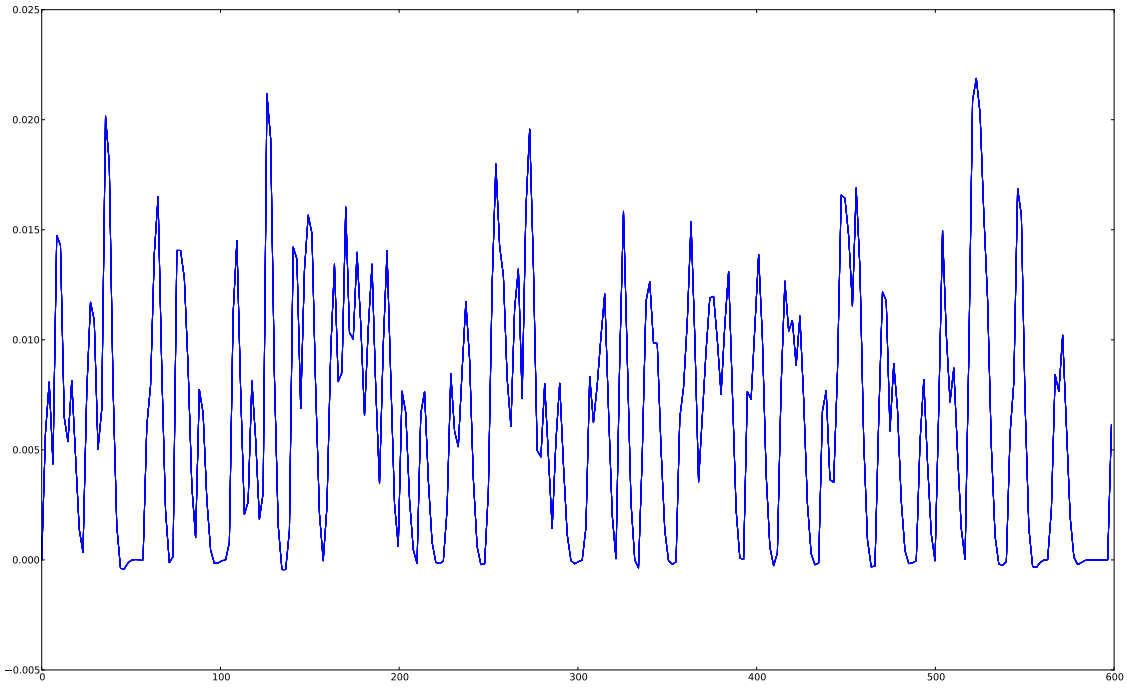


Figure 5.17: BOLD estimates for the non-active, high noise signal. Note the line thickness is caused by all the estimates overlapping.

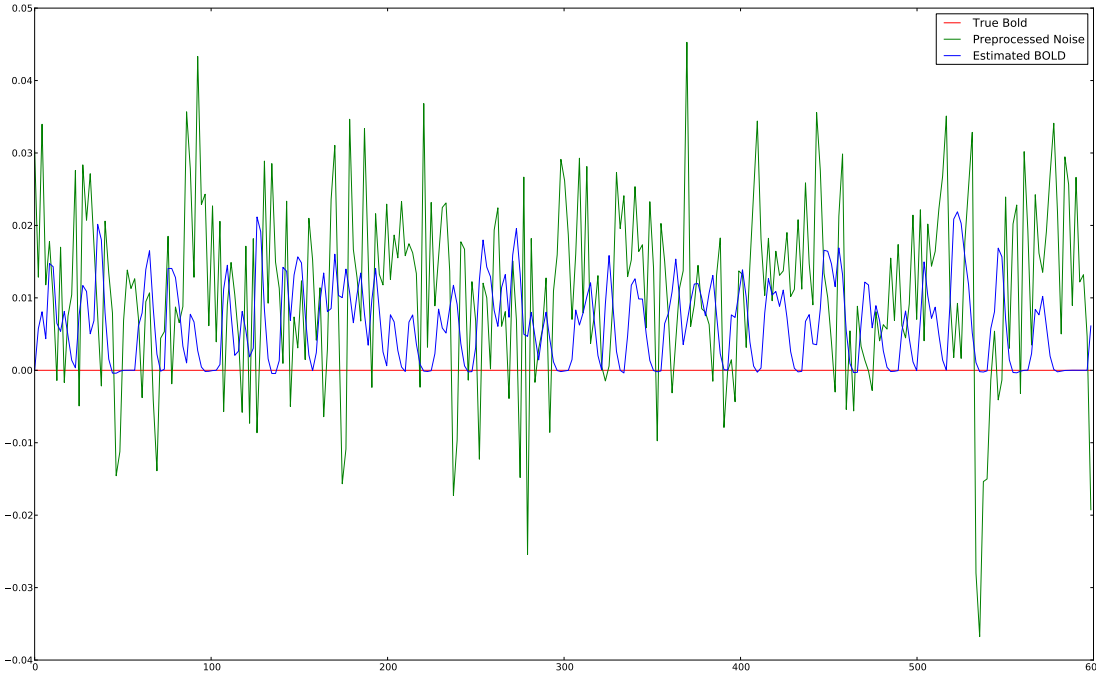


Figure 5.18: Fit from a single particle filter run, with the noise input.

5.1.6 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 residual cannot perform this task. As demonstrated by Table 5.5, 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) which is defined in equation Equation 4.5. 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 5.6. A second potential method of gauging performance is mutual information. Mutual information is a method of measuring the interdependence of two random variables. If two signals are truly independent, then the mutual information will be zero. Although ideally suited to discrete distributions, 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:

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

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

$$\text{bias} = \frac{N_{bins}}{2Nlog(2)} \tag{5.4}$$

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:

$$\text{bias} = \frac{18}{N} \tag{5.5}$$

Note that subtracting the bias 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 Section 5.1.3 and Section 5.1.5 in Table 5.6, distinguishing between these cases with ether normalized residual or mutual information is not clear cut. While the average mutual information is in fact more than 10 times the average mutual information in the two non-signal cases, the maximum mutual information of the low noise/no signal case exceeds the minimum M.I. of the high noise/signal case. The Low Noise/ signal determination is easier to make; given the minimum mutual information is above .8 and the maximum normalized residual is below .6. However, it is worth noting that the worst case scenario for mutual information (maximum) in the low noise/no signal case does not concur with the worse case (minimum) normalized residual. There is no reason why this has to be the case, but it is beneficial. In other words, if it were necessary to make a statement that a particular voxel were active or inactive; the accuracy would improve if both techniques were used with loose restrictions.

	Signal				No Signal			
	Low Noise		High Noise		$\sigma_y = 0.001, \sigma_x = 0.0005$		$\sigma_y = 0.01, \sigma_x = 0.005$	
	$M.I.$	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 5.6: Mutual Information and the normalized $\sqrt{MSE}$, for signal/noise configurations.

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 inconsistent, the quality of the fit was actually quite consistent over eleven runs. The second purpose was of course to determine how the particle filter responded to different signal to noise ratios. From Section 5.1.2, the particle filter performed extremely well in filtering out nonsensical measurements.

5.2 Multi-voxel Simulation

To test the usefulness of the particle filter on a larger scale, 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. It is worth noting is that the noise level was set to an SNR of 20, but due to changes in the program the true signal-to-noise ratio was much lower, as seen in the in Figure 5.19. The mean SNR for each region was calculated only for the voxels with a Signal-To-Noise ratio above 0.1.

For each time-series in the simulated FMRI image, the final parameters were saved into a parameter map. This parameter map could then be compared to the map used to generate the simulated data; additionally a new simulation using the calculated parameters could also be generated to

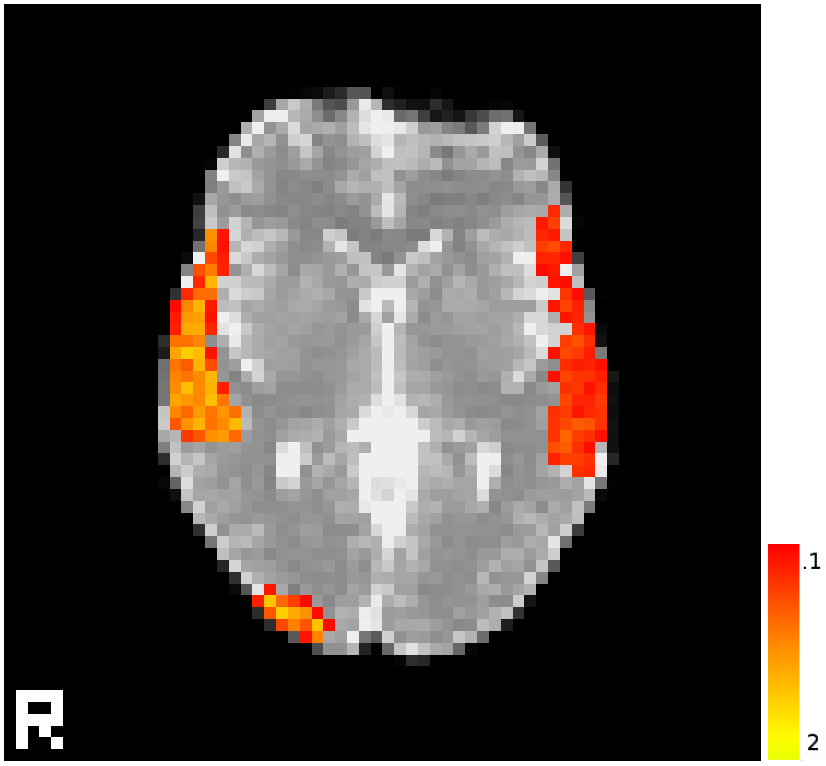
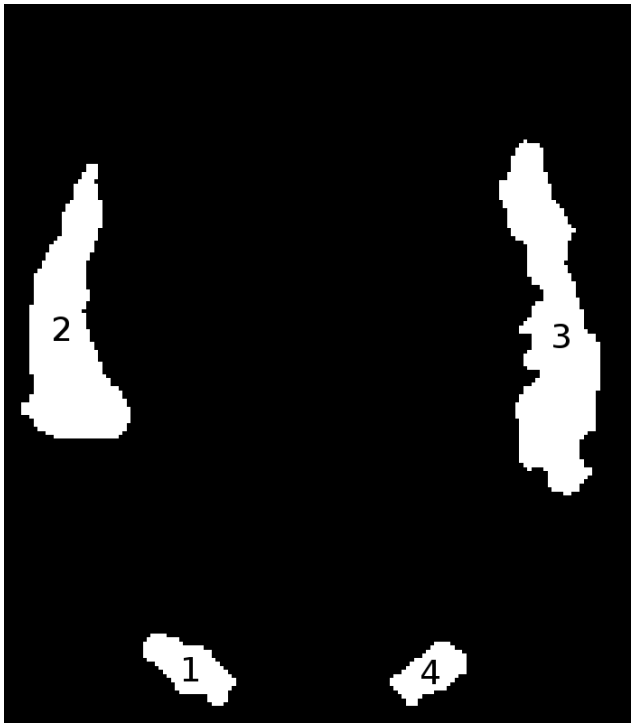


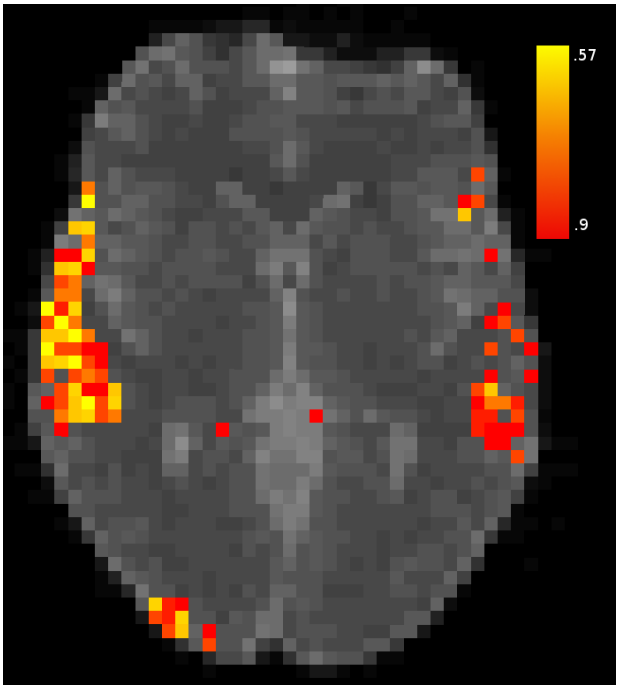
Figure 5.19: SNR Map of POSSUM simulated data. Region 1 mean SNR was 0.8, Region 2 mean SNR was 0.97, and Region 3 mean SNR was 0.39.

test the difference in BOLD levels between the real parameters and the estimated ones. Since the parameters were far from orthogonal [9], this provided a quantitative difference between the two parameter sets.

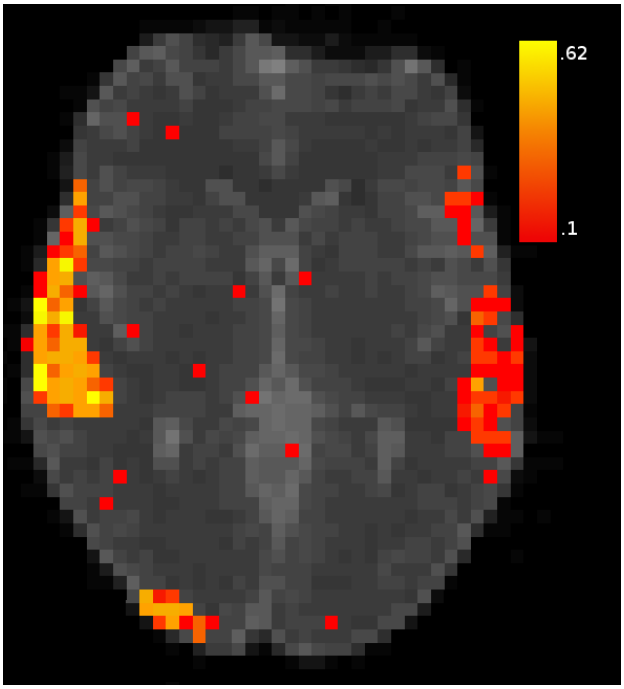


(a) Region labels for simulated slice.

(b) Grey matter map for simulate region.



(c) Residual Map. Scale is Normalized $\sqrt{MSE}$. Lower (yellow) is better.



(d) Mutual Information Map. Scale is bits. Higher (yellow) is better.

Figure 5.20: Comparison of activation with greymatter, parameter regions.

Region	τ_0	α	E_0	V_0	τ_s	τ_f	ϵ
1	1.454	0.321	0.369	0.036	0.994	2.774	1.348
2	1.151	0.353	0.380	0.026	1.98	2.333	1.645
3	1.951	0.317	0.348	0.027	1.657	3.719	0.757
4	1.203	0.310	0.326	0.036	2.168	2.272	0.086

Table 5.7: Actual parameters for each regions in the simulated slice.

The regions are numbered according to 5.20(a); the parameters for each region may be found in Table 5.7.

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 an 3 stick out in both the residual and the mutual information map, indicating that the particle filter was successful in matching those regions. Mutual information was an extremely succesfull metric, with the exception of a few false positives. Thus, another heatmap (Figure 5.21) is helpful to show what threshold would remove those false positives.

The histograms again demonstrate that a single point estimate of the parameters is elusive for this set of parameters. However the data clearly show the power of the particle filter at identifying regions of activation; which is 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 removes the false positives present in the images at the cost of false negatives. As noted in Section 5.1.6 the false positives present in the Mutual Information map are different from those in the residual map. This furthers the argument from for combining the two metrics to increase power. Although at first glance it would appear that there are false negatives in the Figure 5.20; 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 indicate that the particle filter is effective at regressing against a noisy signal.

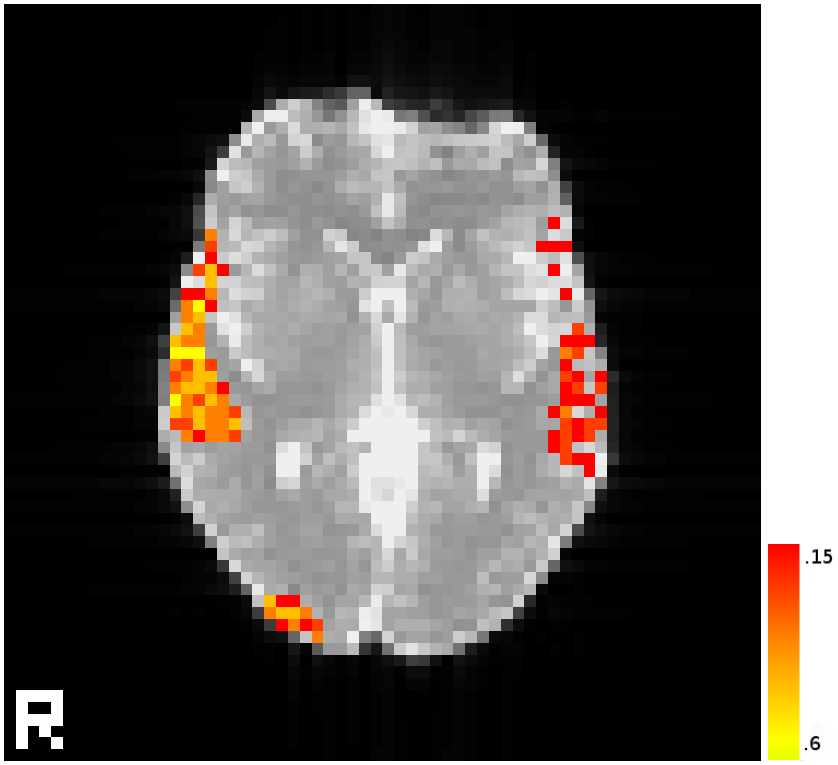


Figure 5.21: More stringent mutual information heatmap. Higher (yellow) is better.

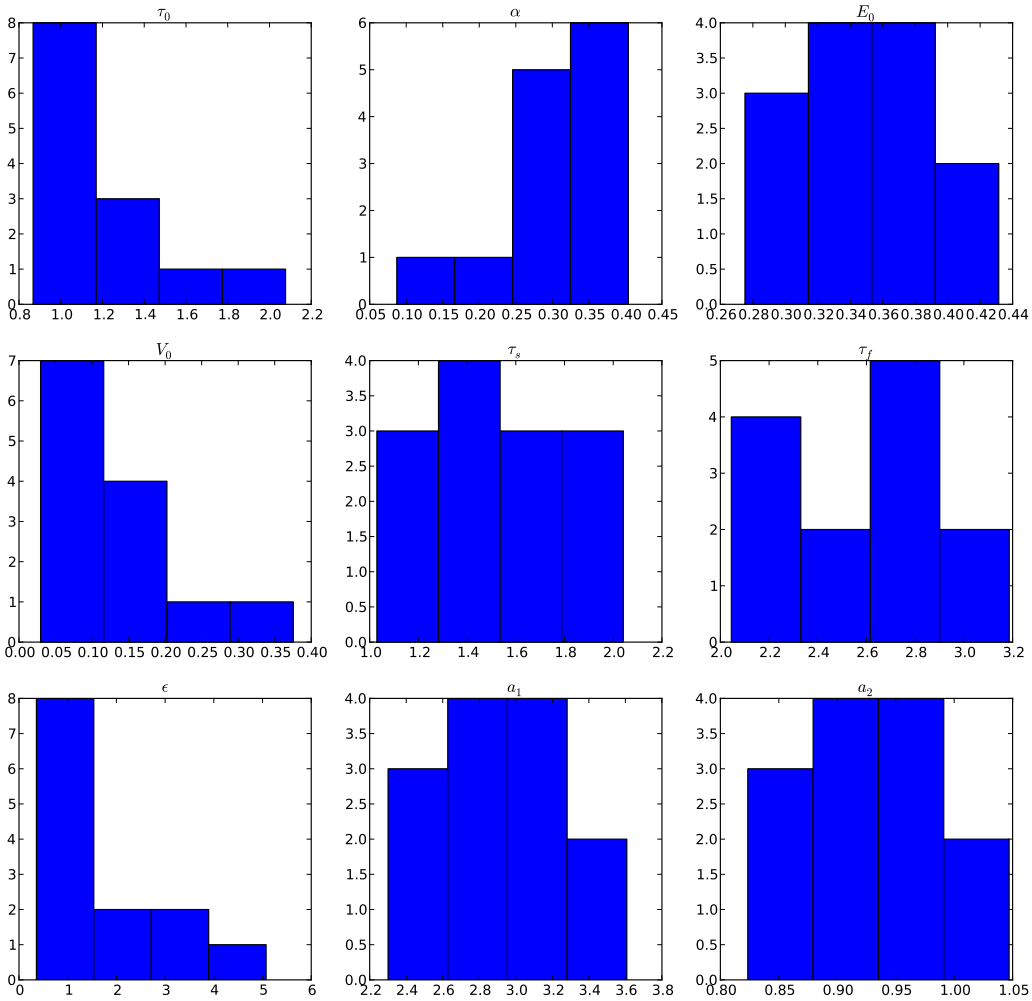


Figure 5.22: Histogram of estimated parameters in section 1 in voxels with mutual information greater than 0.14

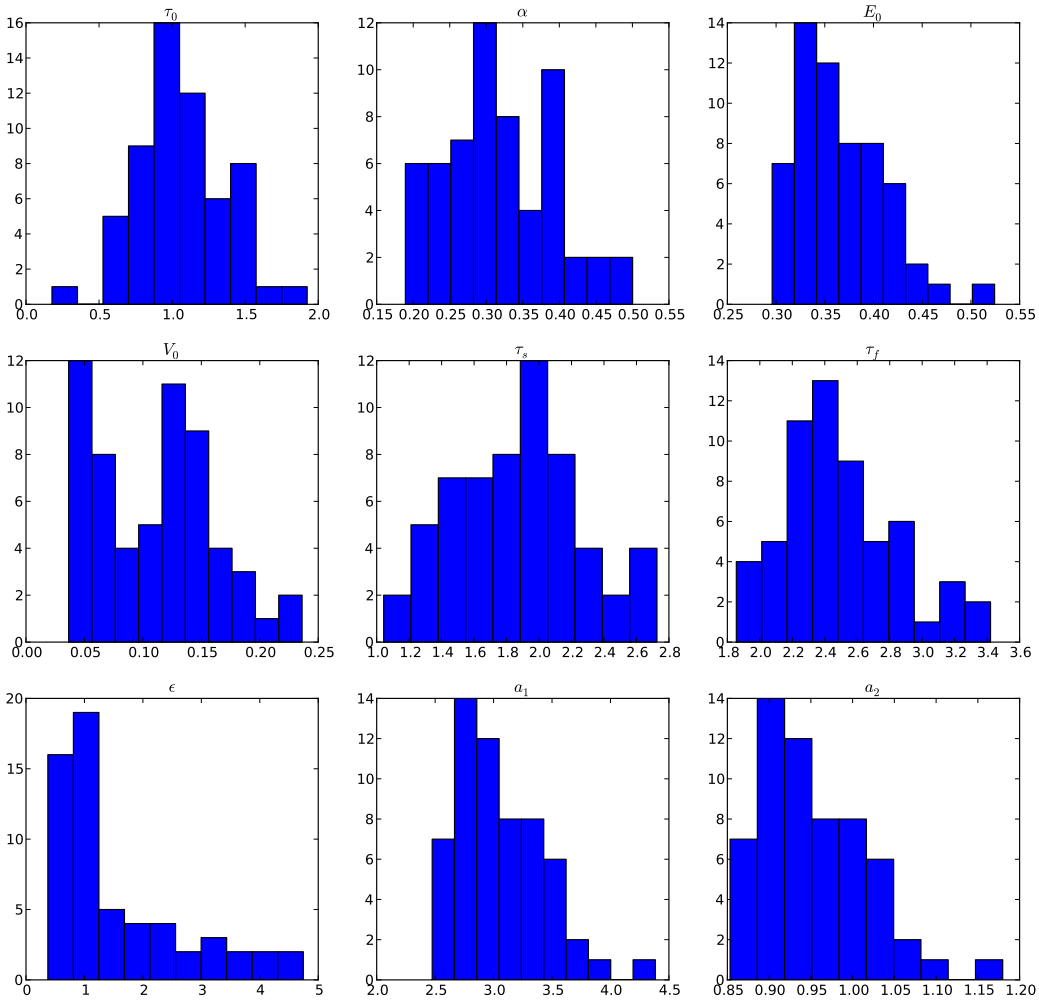


Figure 5.23: Histogram of estimated parameters in section 2 in voxels with mutual information greater than 0.14

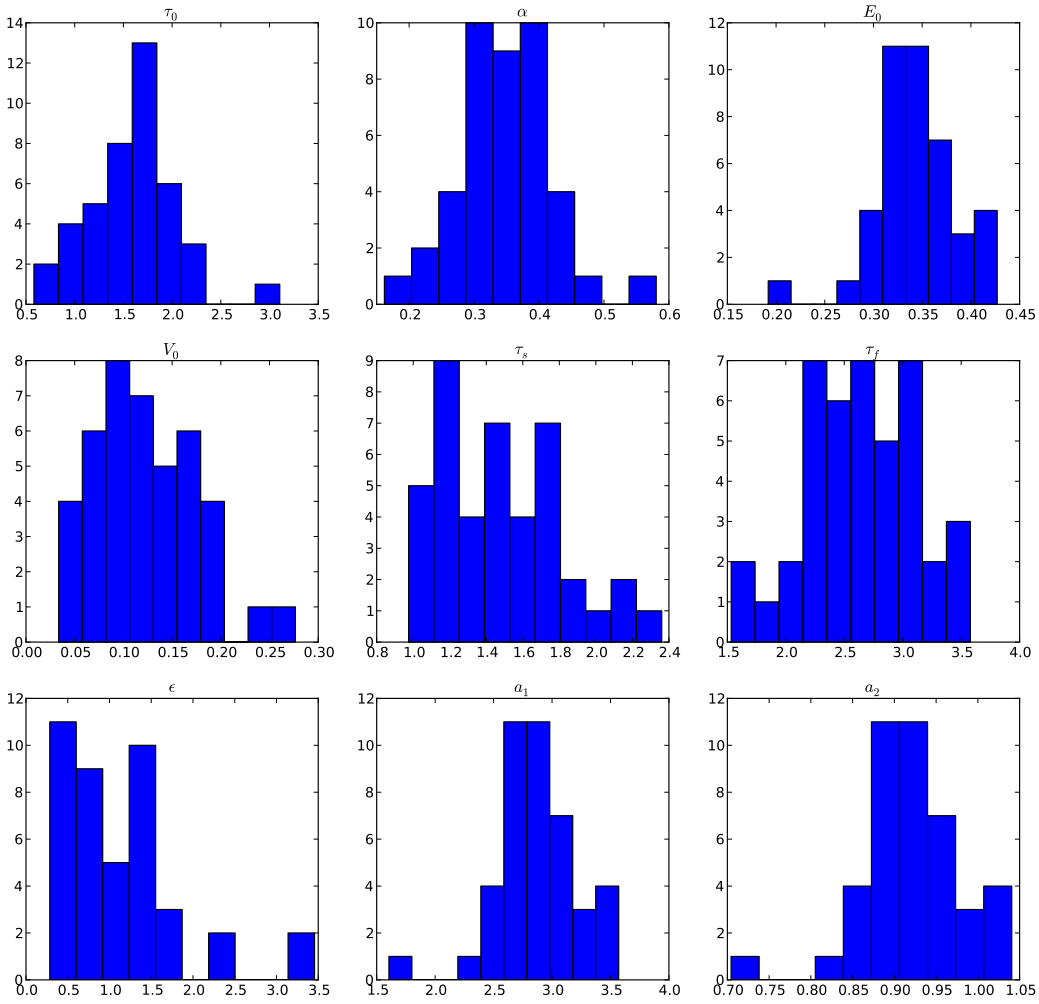


Figure 5.24: Histogram of estimated parameters in section 3 in voxels with mutual information greater than 0.14

Chapter 6

Real Data

The ultimate goal of modeling the BOLD response is to estimate parameters in real FMRI data. The most basic use of modeling the BOLD signal is to locate activation. A voxel is considered active when the stimulus is the primary drive for the BOLD response. This is in contrast to most voxels which have are controlled by intermediate or complete unrelated factors. Inactive regions cannot be modeled because their input is unknown, therefore parameter estimates in such regions are impossible. Because SPM is the de facto standard for localizing activation, this section compares its output with that of the particle filter.

Note that SPM must pre-process the image with a spatial smoothing filter. For this work SPM8 was smoothed with a $8\text{mm} \times 8\text{mm} \times 8\text{mm}$ Full-Width Half Maximum (FWHM) Gaussian kernel. Additionally, SPM8 applied a high pass filter (with a cut off based on a globally estimated autocorrelation). Thus the preprocessing pipeline of SPM is very different from that of the particle filter. SPM8 also outputs a T-statistic for each voxel, whereas the particle filter's primary output is a posterior probability distribution of the parameters at every voxel. To validate the quality of the particle filter, the results were compared with SPM, both in terms of the location and the fit.

6.1 Experiment Configuration

For the FMRI data discussed in Chapter 6, tests were performed on a right handed volunteer using a GE SIGNA HDx 1.5 Tesla scanner with a single echo EPI sequence. Slice spacing was 5mm, and pixel sizes were 3.75mm. Repitition Time was 2.1s, Echo Time was 40 ms and the imaging frequency was set to 63.854MHz. The image resolution was $64 \times 64 \times 28$. The subject was presented with either a single or double flash and was asked to respond with a right handed or lefted handed finger tap, respectively. The FMRI began 18.9 seconds before the beginning of the experiment, to allow for transients in to image to settle out. The timing of the flashes are shown below with time 0 corresponding to the beginning of the 10th TR (so 9 images were dropped).

The timing of the single flashes were:

1.706, 11.944, 17.063, 18.769, 34.125, 39.244, 44.231, 47.644, 49.350, 61.294, 64.706, 66.413, 69.825, 71.531, 73.238, 76.650, 80.063, 90.169, 96.994, 110.644, 117.469, 120.881, 130.988, 132.694, 154.875, 158.288, 161.700, 165.113, 168.525, 176.925, 178.631, 183.750, 190.575, 204.225, 205.931, 211.050, 216.038, 222.863, 226.275, 236.513, 248.456, 255.281, 258.563, 263.681, 273.919, 277.331, 287.569, 292.688, 294.394, 299.381

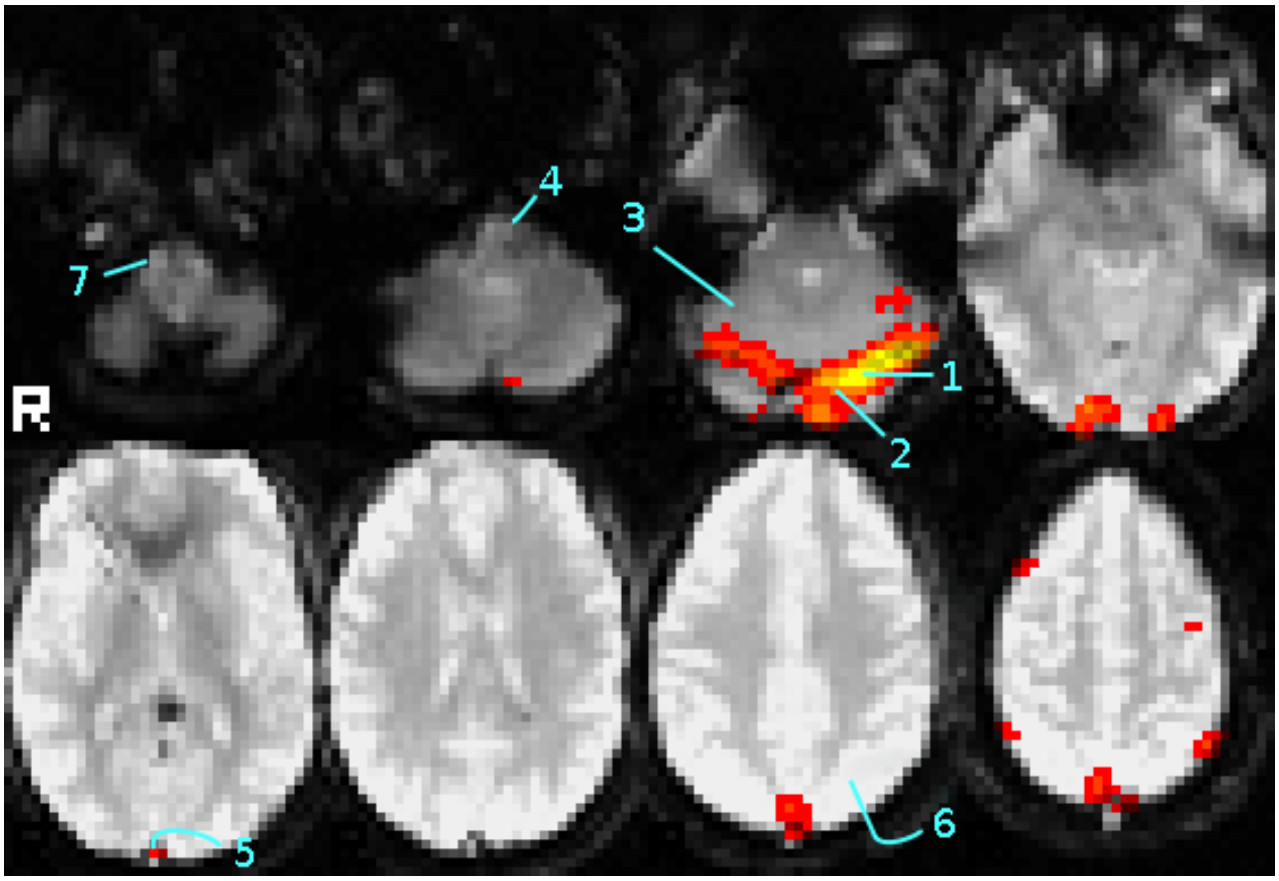
and the timing of the double flashes were:

0.131, 6.825, 20.475, 27.300, 35.831, 52.763, 54.469, 59.588, 86.756, 91.875, 107.231, 108.938, 112.350, 114.056, 115.763, 119.175, 126.000, 134.400, 136.106, 141.225, 144.638, 156.581, 159.994, 166.819, 171.806, 175.219, 185.456, 188.869, 202.519, 212.756, 217.744, 221.156, 227.981, 229.688, 233.100, 243.338, 245.044, 246.750, 250.163, 261.975, 270.506, 272.213, 280.744, 282.450, 289.275, 296.100, 301.088, 304.500

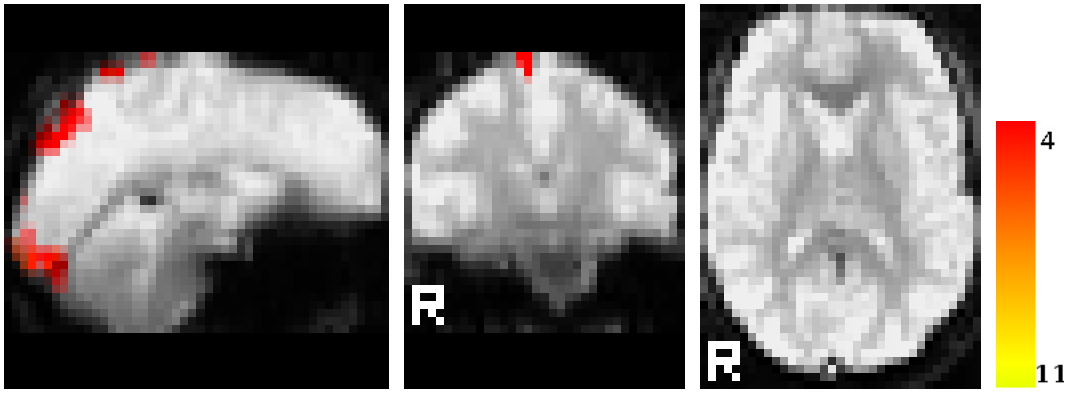
After dropping the first 9 volumes, each of the remaining volumes was co-registered with the first. At this point the SPM method diverges from the experimental method. For the particle filter, detrending was then applied as discussed in Section 4.2.2 and the resulting data was processed with the particle filter. To generate the SPM output in this section, the co-registered data was spatially smoothed, and then filtered with an adaptive cut off (which is built into the SPM analysis). For the SPM analysis, the Canonical HRF was used and model time derivatives were included although not in the contrast vector. All other settings used in SPM8 were left at default for analyzing FMRI data.

6.2 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 elminated 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.

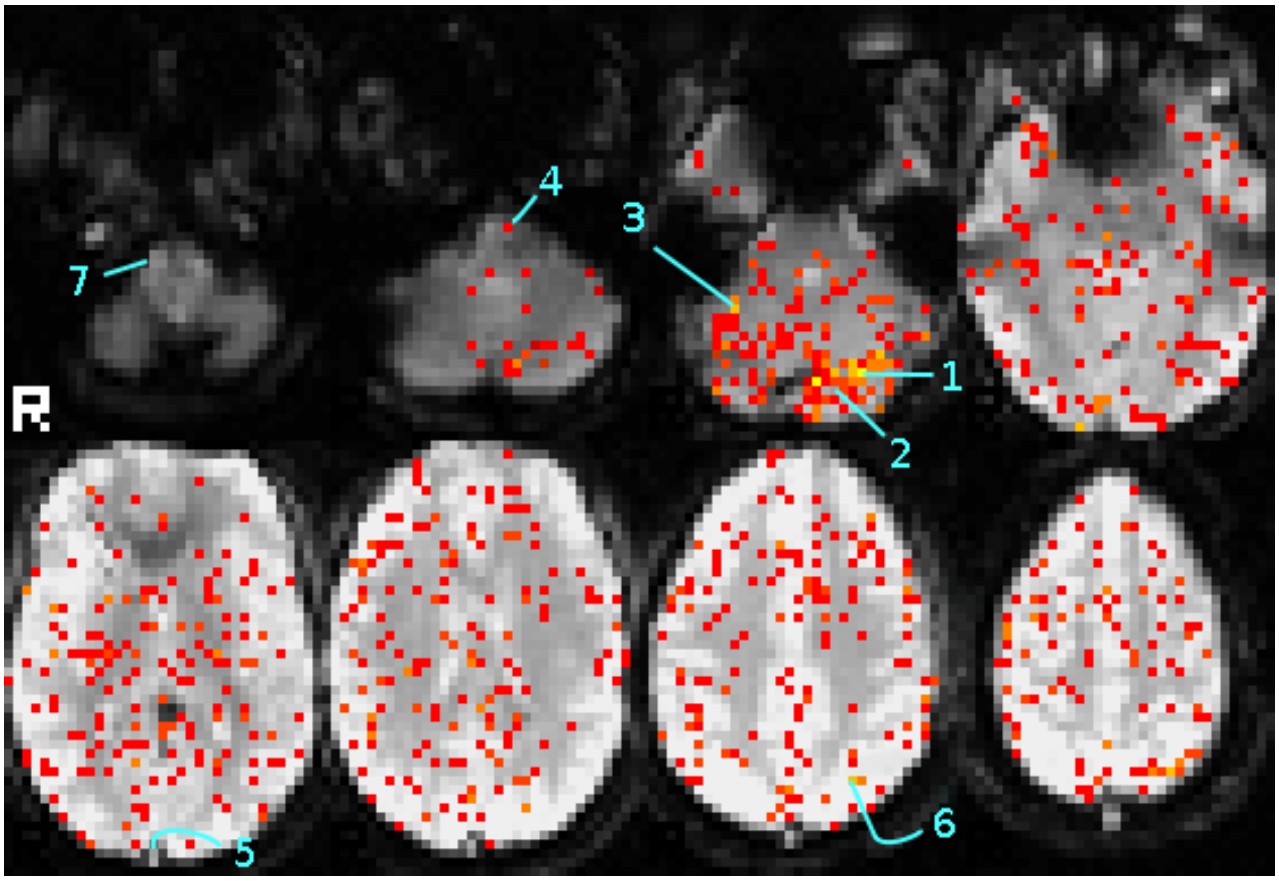


(a)

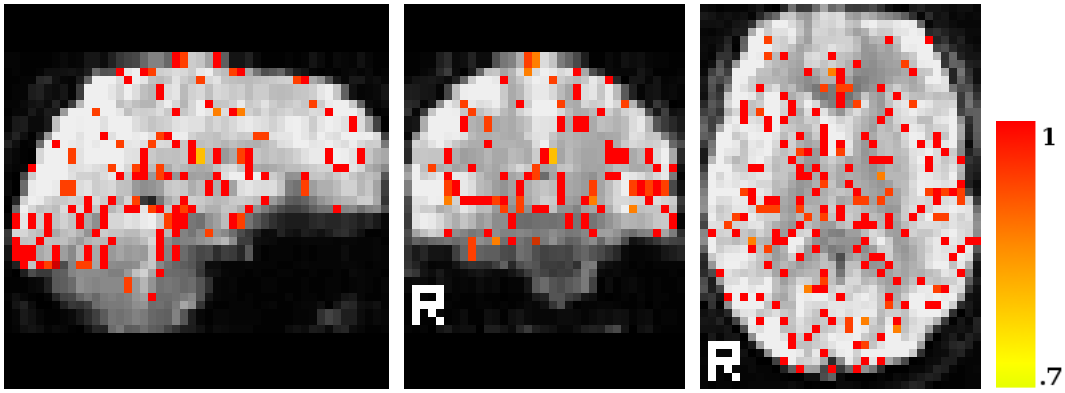


(b) (c) (d)

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

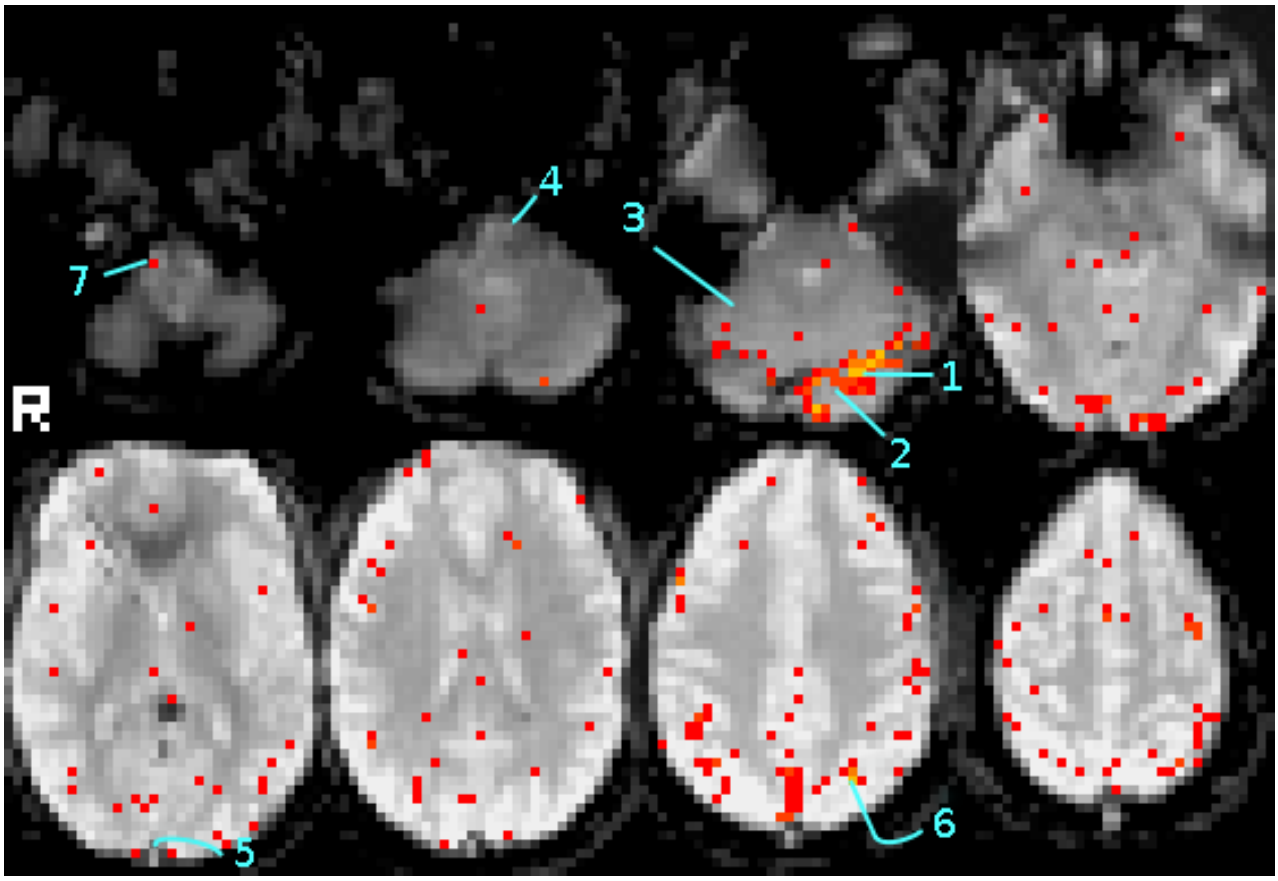


(a)

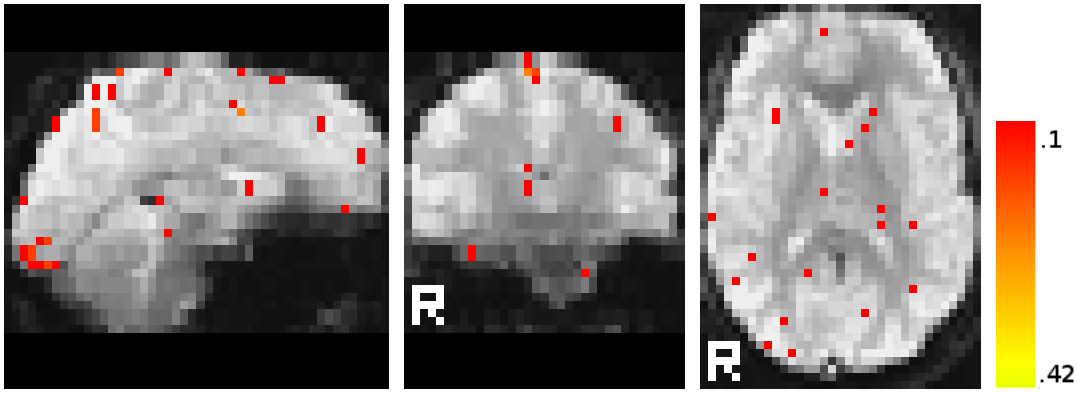


(b) (c) (d)

Figure 6.2: Particle Filter results measured in normalized $\sqrt{MSE}$. Units of match is normalized residual where the the lowest (best) levels shown are .7 and the highest error shown (threshold) is 1. Sagittal, coronal and axial slices are shwon in 6.2(b), 6.2(c), and 6.2(d), respectively. A series of axial slices are shown in 6.2(a).

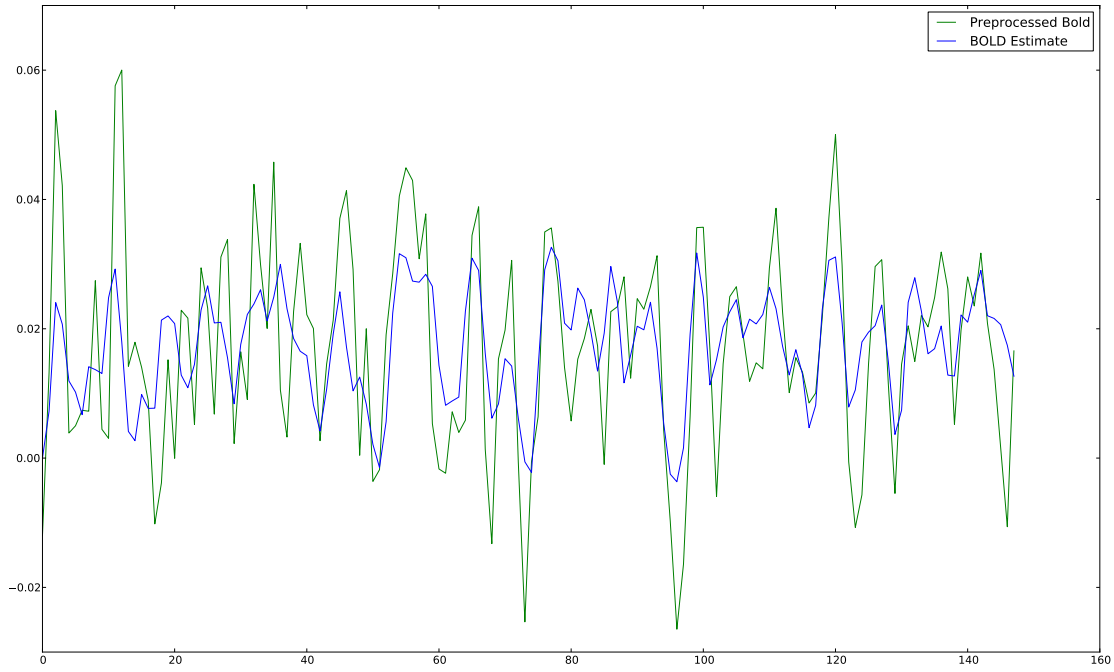


(a)

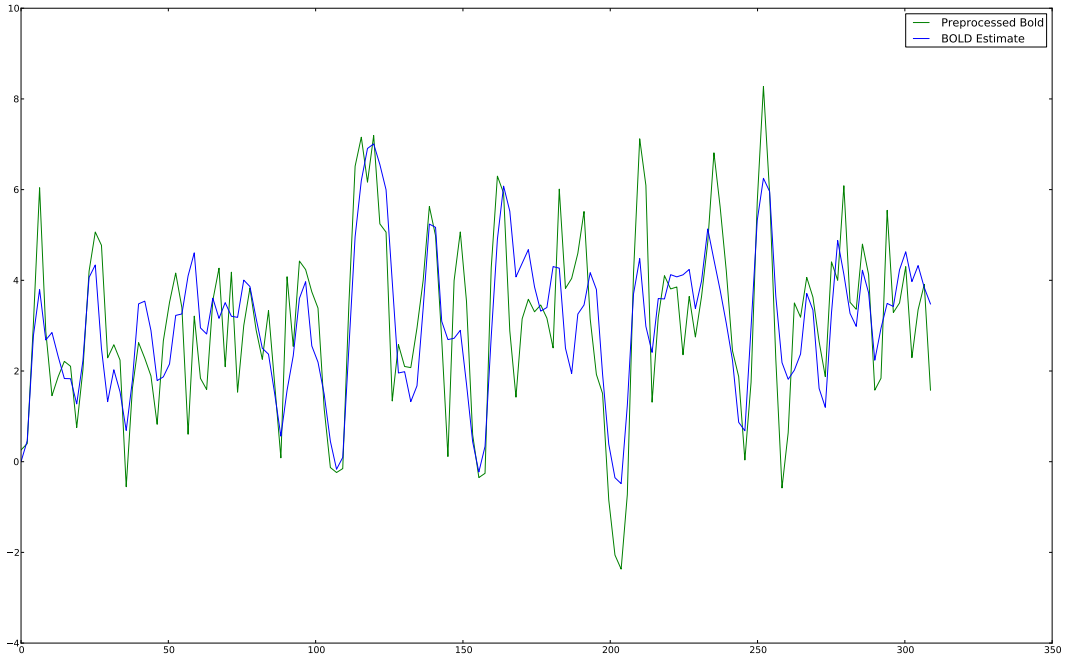


(b) (c) (d)

Figure 6.3: Particle Filter results measured in mutual information. Units of match is bits (standard for base-2 Mutual Information). The highest (best) levels are .42 and the worst shown (threshold) is .1. Sagittal, coronal and axial slices are shwon in 6.3(b), 6.3(c), and 6.3(d), respectively. A series of axial slices are shown in 6.3(a).

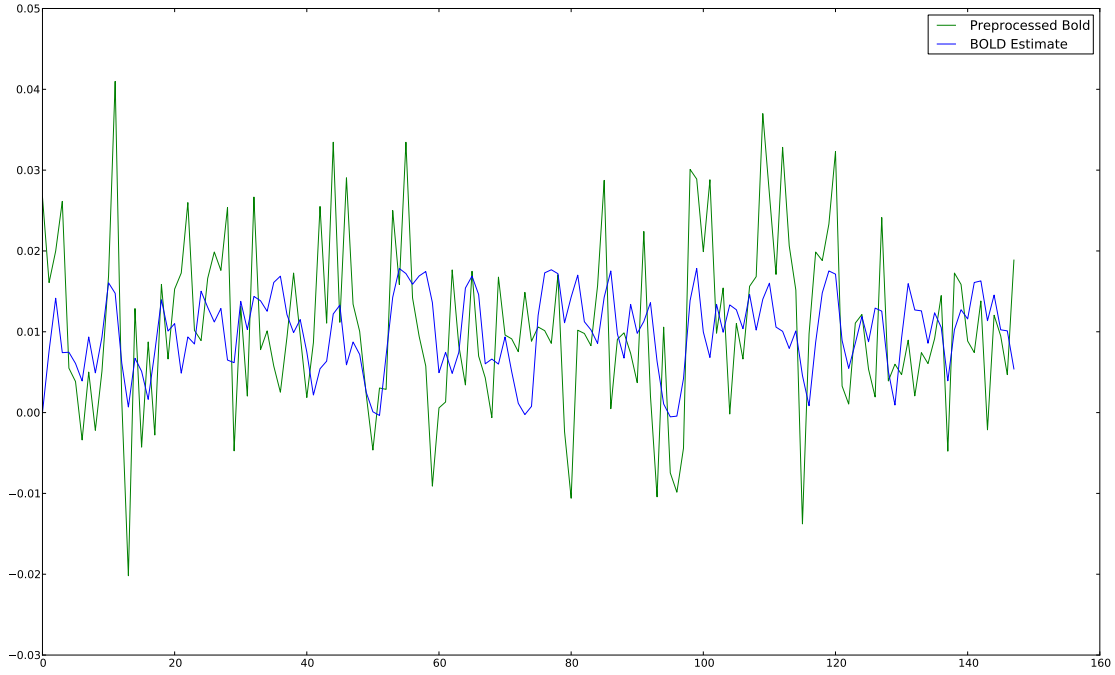


(a) Particle Filter

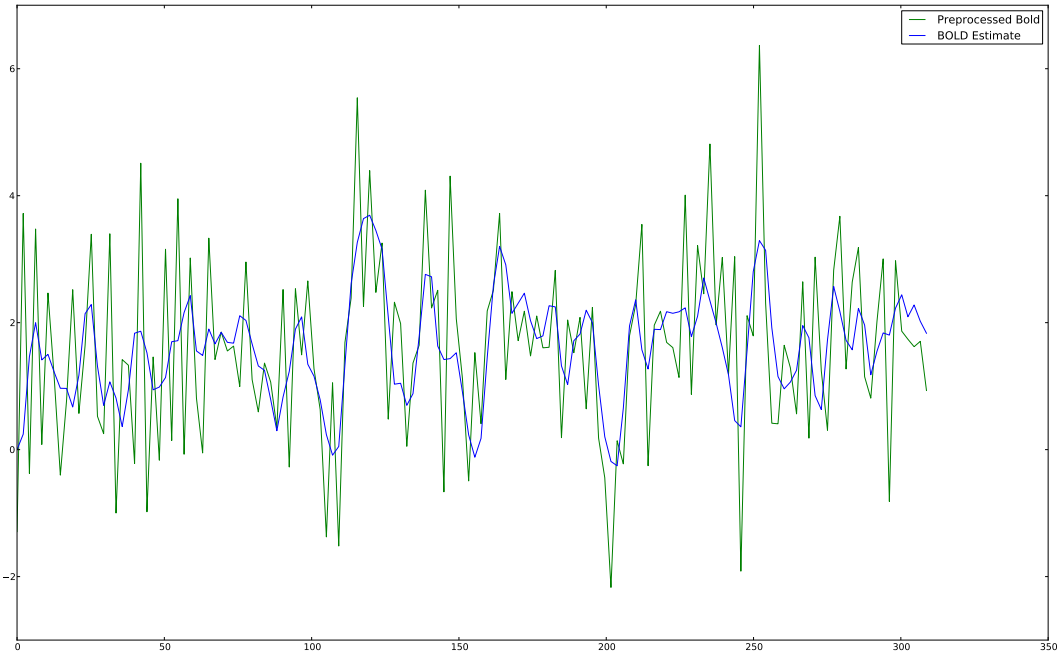


(b) SPM

Figure 6.4: Section 1, Estimated vs. Actual BOLD response. T-Score: 10.71, Mutual Information: 0.33, Residual: 0.72.



(a) Particle Filter



(b) SPM

Figure 6.5: Section 2, Estimated vs. Actual BOLD response. T-Score: 6.97, Mutual Information: 0.04, Residual: 1.02.

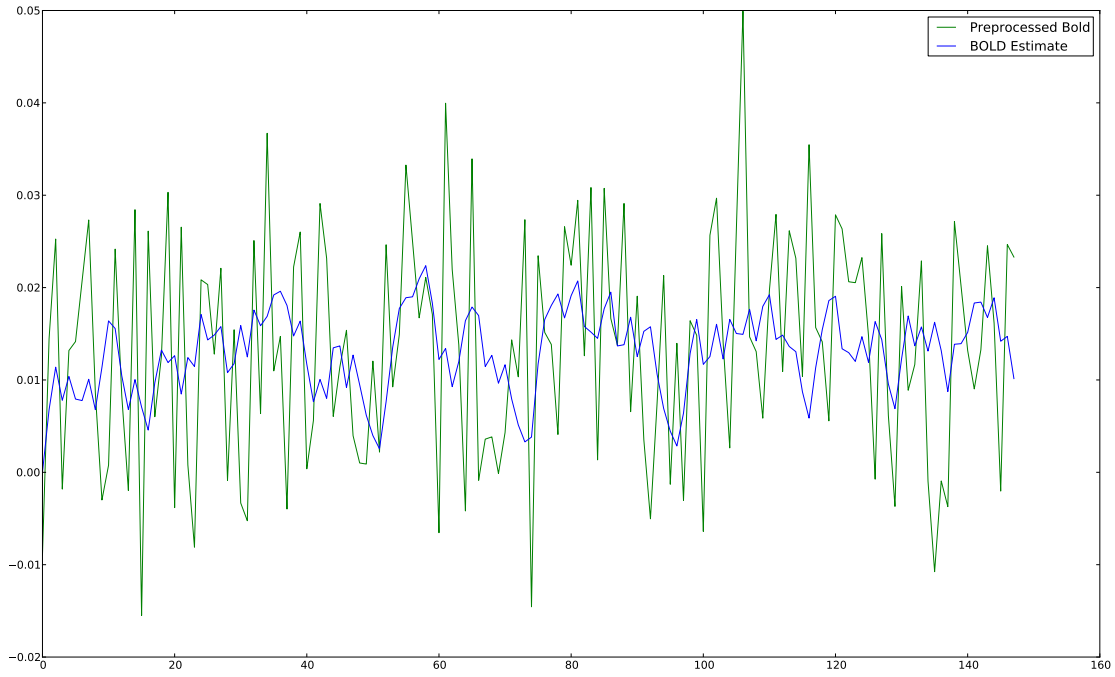
I chose several voxels to discuss further from Figure 6.2 and Figure 6.1. The first voxel, labeled 1, had a very high T-score, high mutual information, as well as a low error (around .7). Thus, the fit should be very good in both the SPM and particle filter output; this comparison is shown in Figure 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 work.

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 residual was high and the mutual information was low. This is a prime example of a false positive due to the large smoothing kernel applied by SPM. For instance at 75 seconds, the stimulus is not present, and so the signal should drop off; yet it doesn't. While a few peaks seem to match, most of the signal does not correlate with the expected state progression. This voxel is not being driven direcly by the input, although it may be gated or driven through intermediate region.

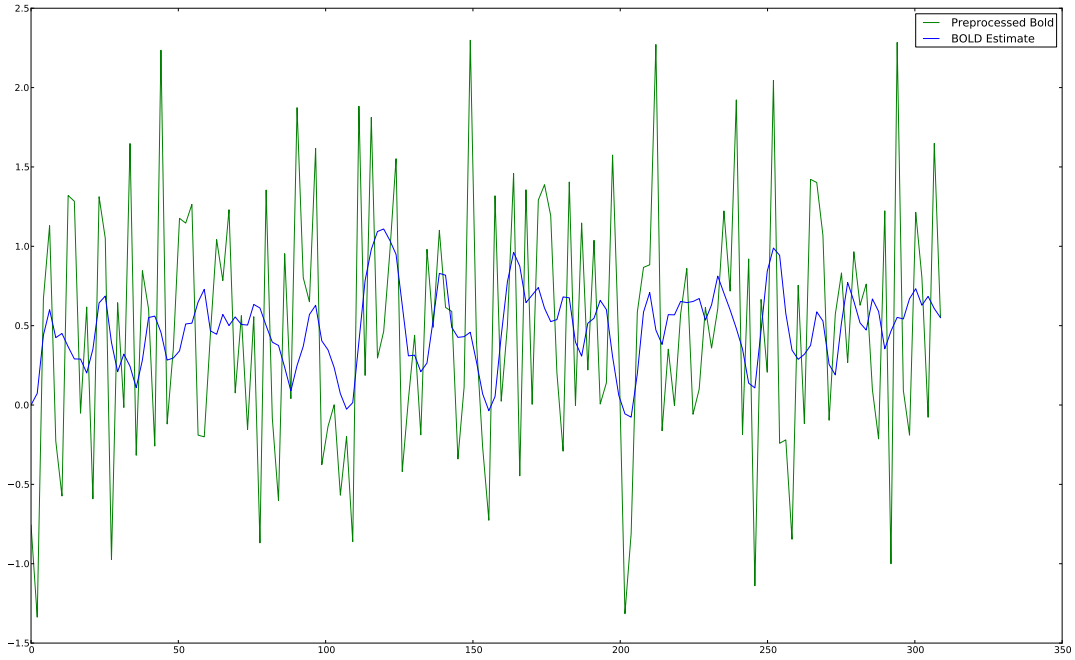
The third voxel, compared in Figure 6.6, was far away from any other active voxels and yet had a very low (around .7) residual. At the same time, the mutual information was high enough to make the point suspect. Although the estimated signal seems to run down the middle of the measurement signal; it is clear that there is a significant amount of noise present in the signal that is not being explained by the particle filter, In both preprocessed time-series the input is extremely noisy, yet by the normalized residual the response is good. This is an example of a false positive from the residual metric.

The fourth voxel (Figure 6.7) selected for anyalysis had a relatively high residual, and was not picked for activation by SPM. The mutual information was only 0.06 for the image. It is simple to see why the residual was not acceptable in this case. At the same time, the peaks do seem to correlate with the measurement peaks. Regardless, this is an example of a false positive from the mutual information metric.

The fifth voxel (Figure 6.8) is an example of a region with increased activation due to smoothing. The fit provided by the BOLD particle filter is not very good, and the time series input to SPM is far less noisy. Regardless this is another case where the peaks seem to match, yet nothing else does. Take for instance the measurements from 30 seconds to 40 seconds. At that point there is a significant spike in the BOLD signal, yet the actual measured signal declines. This is an indication that the BOLD signal is not directly being driven by the input. It is likeley that the cause of the activation in this reason is not true activation, but recieved from surrounding active regions through spatial smoothing.

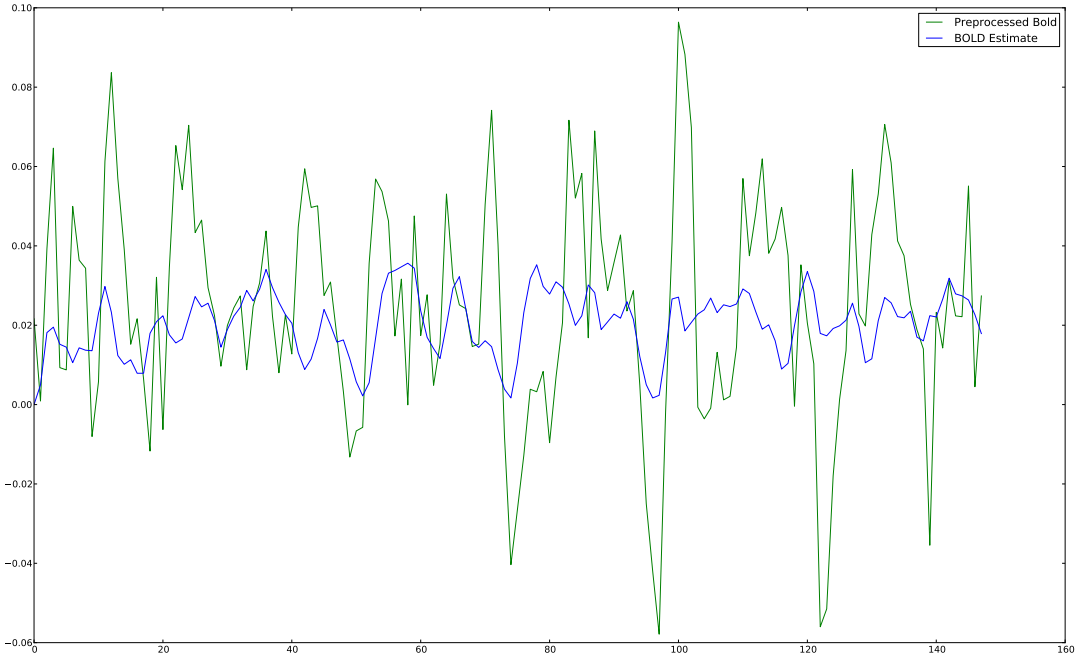


(a) Particle Filter

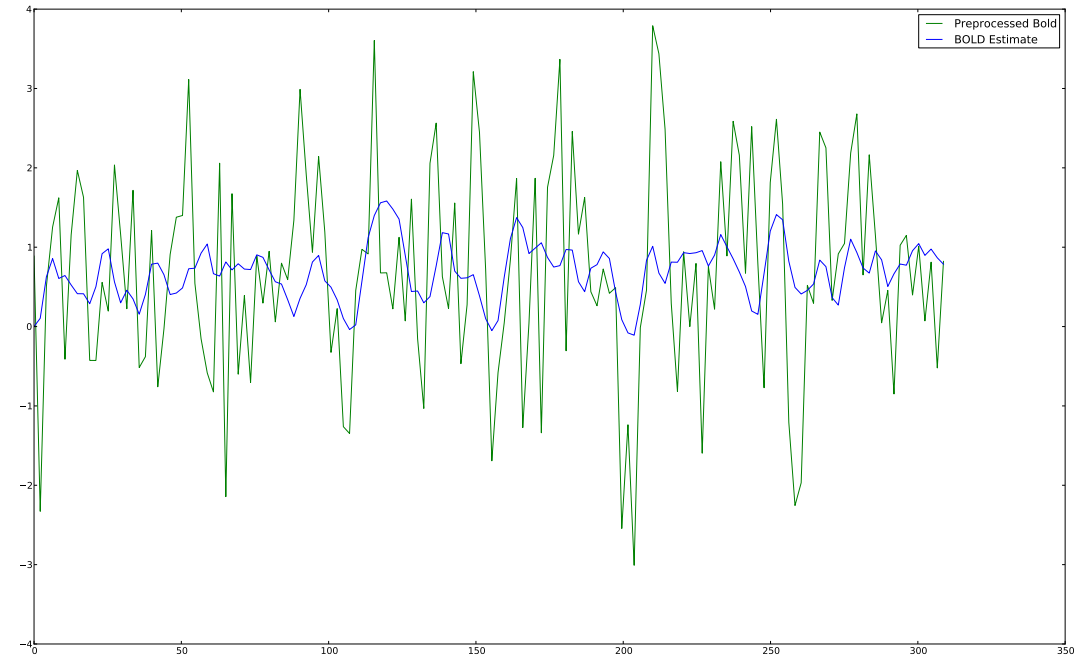


(b) SPM

Figure 6.6: Section 3, Estimated vs. Actual BOLD response. T-Score: 2.85, Mutual Information: -0.03, Residual: 0.81.



(a) Particle Filter



(b) SPM

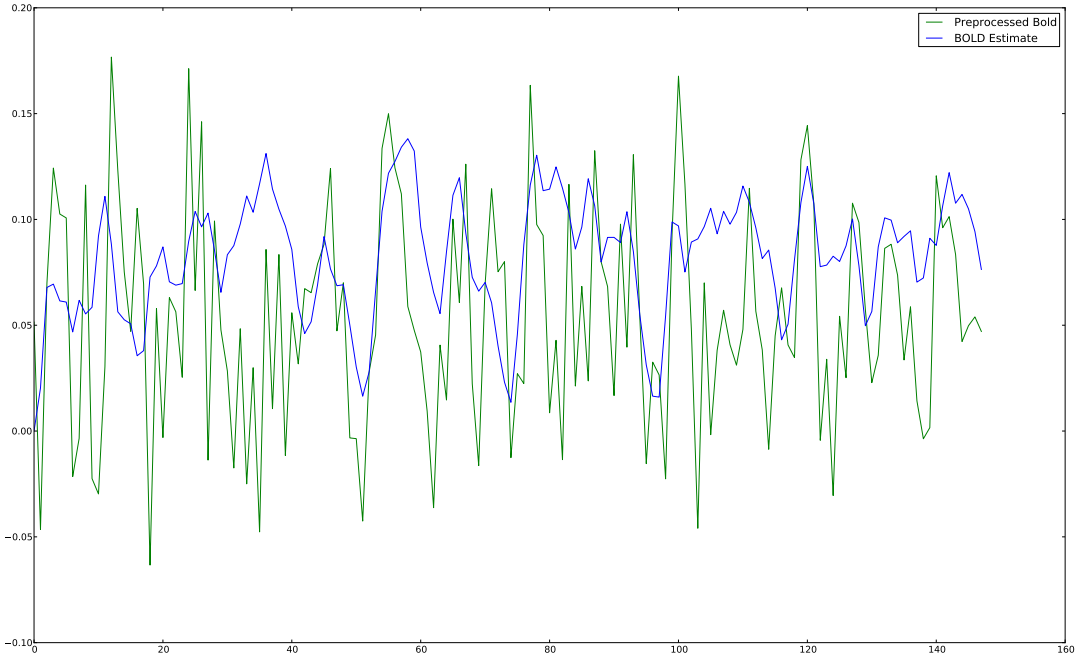
Figure 6.7: Section 4, Estimated vs. Actual BOLD response. T-Score: 0.50, Mutual Information: 0.06, Residual: 0.95.

Voxel 6 (Figure 6.9) is another region that is active according to both metrics used in this work, yet was missed by SPM. The time series in Figure 6.9 shows an extremely good fit, perhaps the best in this batch, for the particle filter. In contrast, the fit for SPM is abysmal. In spite of the fact that several other areas around are active as well, the smoothing seems to have completely wiped out activation in this voxel.

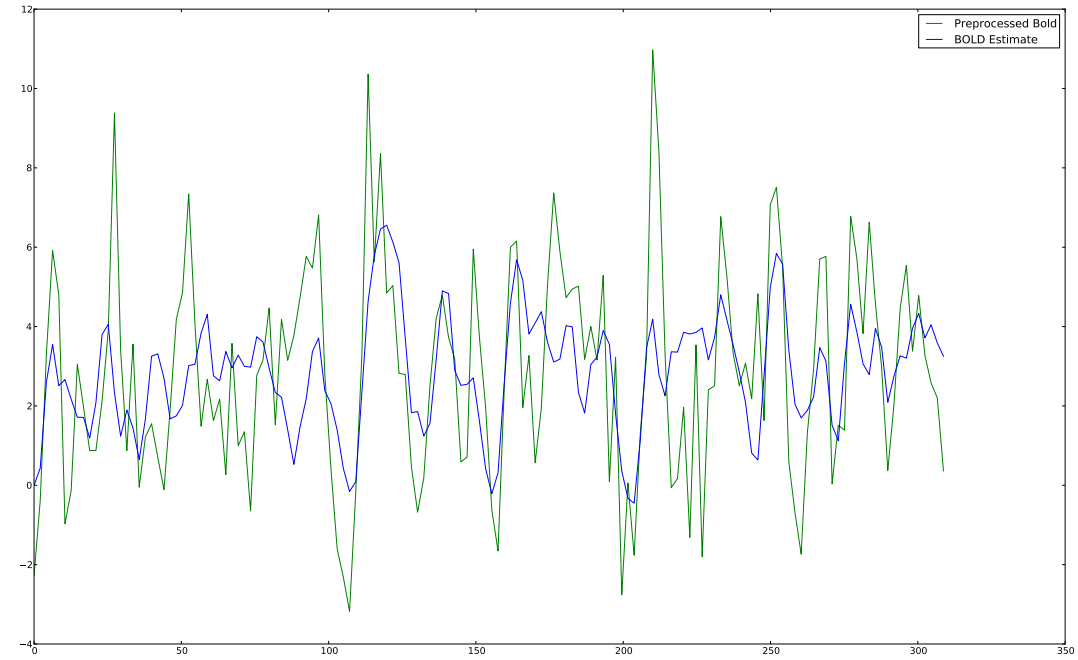
Finally voxel 7 is a completely ambiguous time series. While the thresholds applied here are empirical, this voxel is on the borderline of the thresholds for both mutual information and the residual. The signal itself is extremely noisy, it should the voxel should be rejected. However this case is reminiscent of the pure-noise tests performed in the previous chapter. This time series an extremely good example of the danger of false positives. In spite of the fact that the signal oscillates far faster than the BOLD estimate, because of that, it is somehow able to maintain a mutual information value of $.1052$ and a residual of just 1.09992. As such, a clear method of detecting these sorts of regions is necessary if the results of the particle filter algorithm are toe be trusted.

6.3 Parameter Estimates

Although the parameters are not uniquely identifiable by a single time-series, that does not mean estimating them is not without benefit. The parameters still contain useful information about the system. Additionally, as an aggregate they form a distribution of the feasible parameters parameters for a particular patient. Therefore Figure 6.11 through Figure 6.17 contain the parameter maps for the system and Figure 6.18 is a histogram across all voxels for which the mutual information was greater than $.15$. As before, the threshold is not scientifically derived, yet in tests this threshold provided a decent balance to remove most of the questionably active voxels in the system.

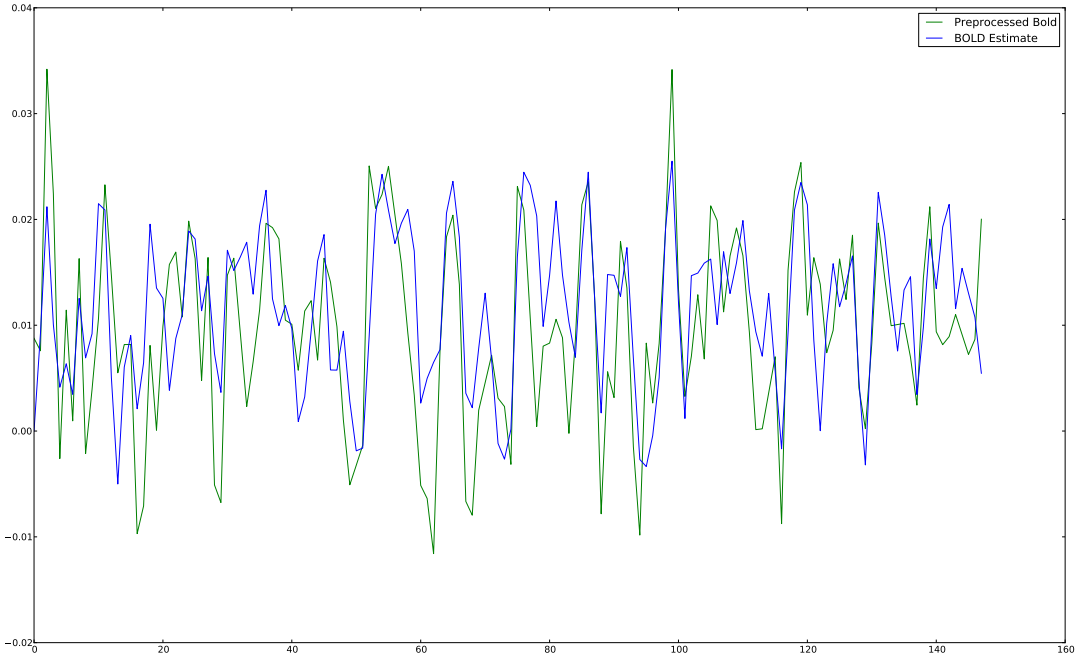


(a) Particle Filter

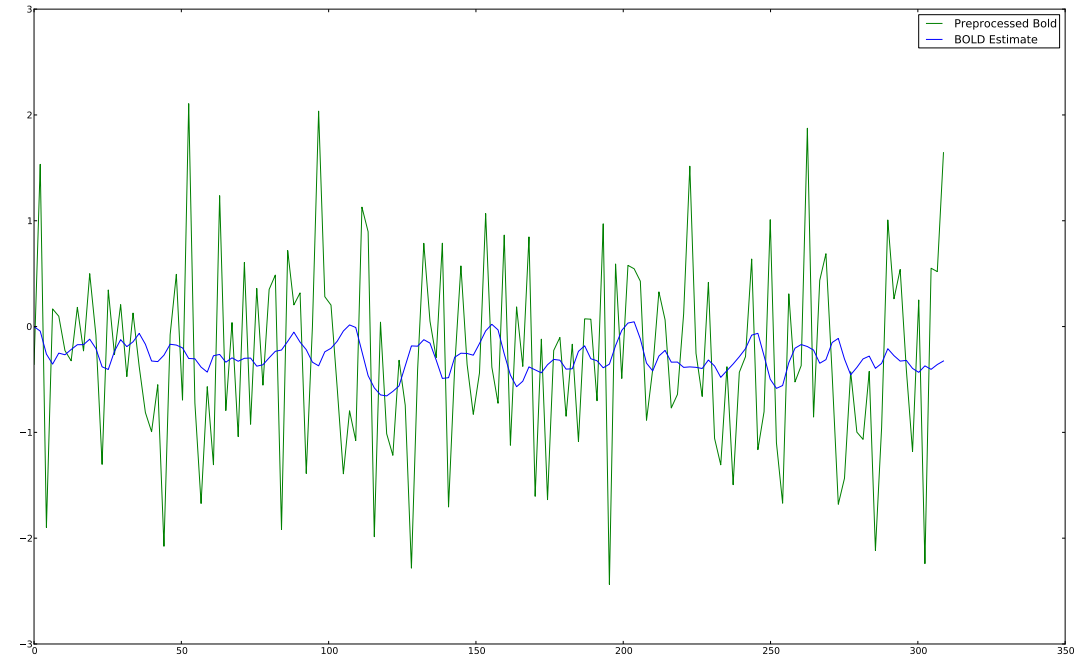


(b) SPM

Figure 6.8: Section 5, Estimated vs. Actual BOLD Response. T-Score: 4.17, Mutual Information: 0.02, Residual: 1.14.

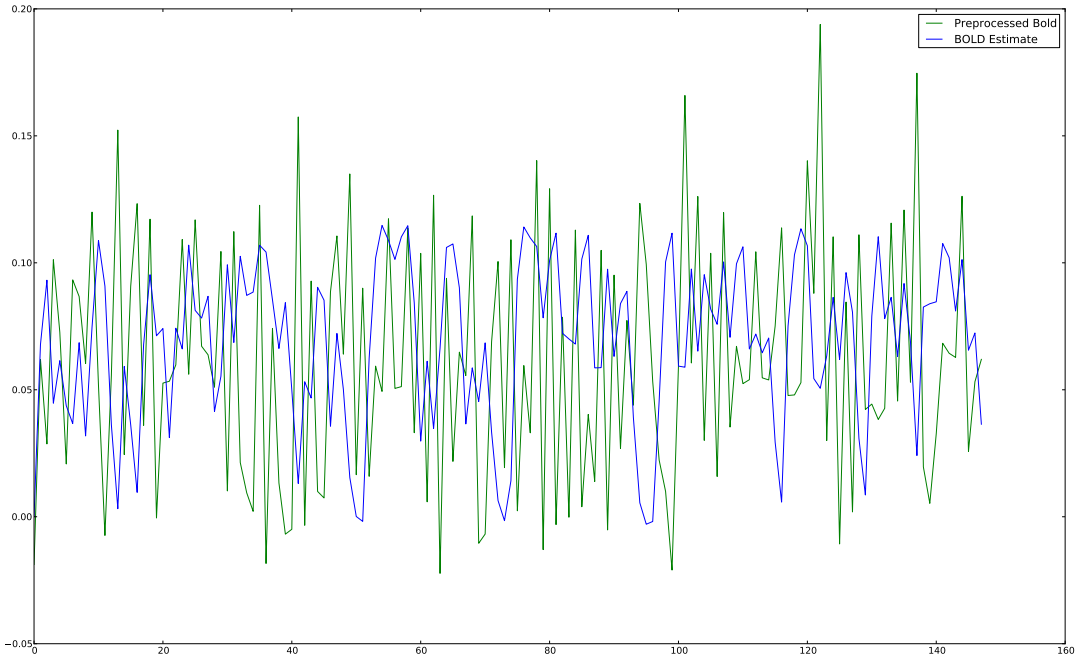


(a) Particle Filter



(b) SPM

Figure 6.9: Section 6, Estimated vs. Actual BOLD Response. T-Score: 2.49, Mutual Information: .34, Residual: 0.78.



(a) Particle Filter

Figure 6.10: Section 6, Estimated vs. Actual BOLD Response. T-Score: 1.32, Mutual Information: 0.11, Residual: 1.10.

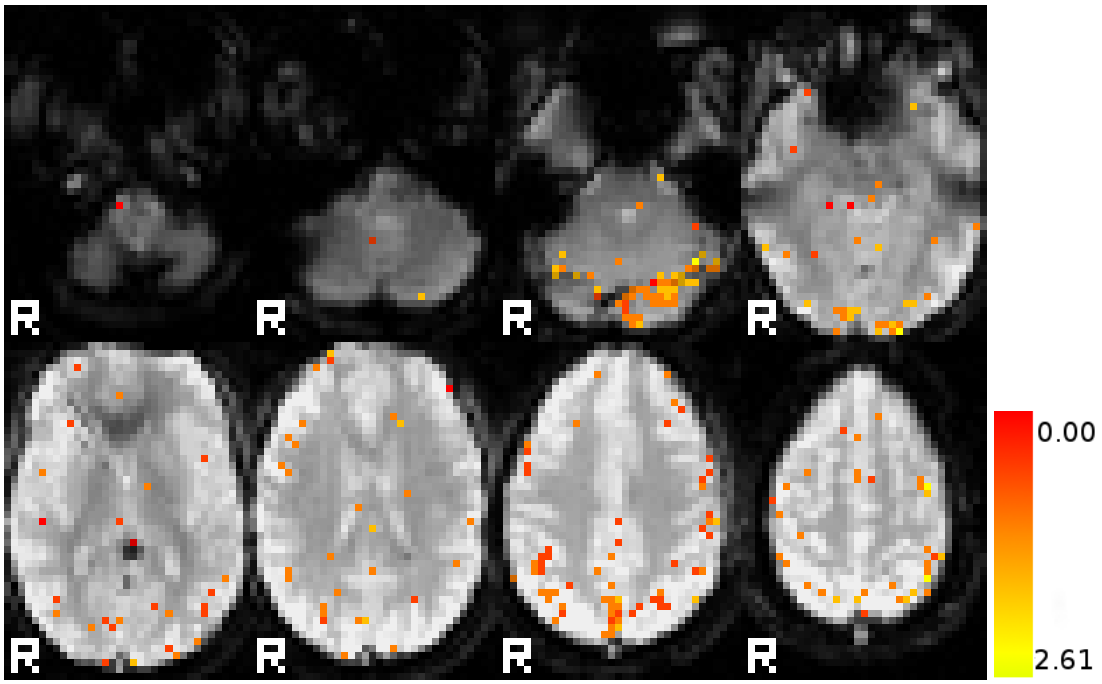


Figure 6.11: τ_0 Estimates

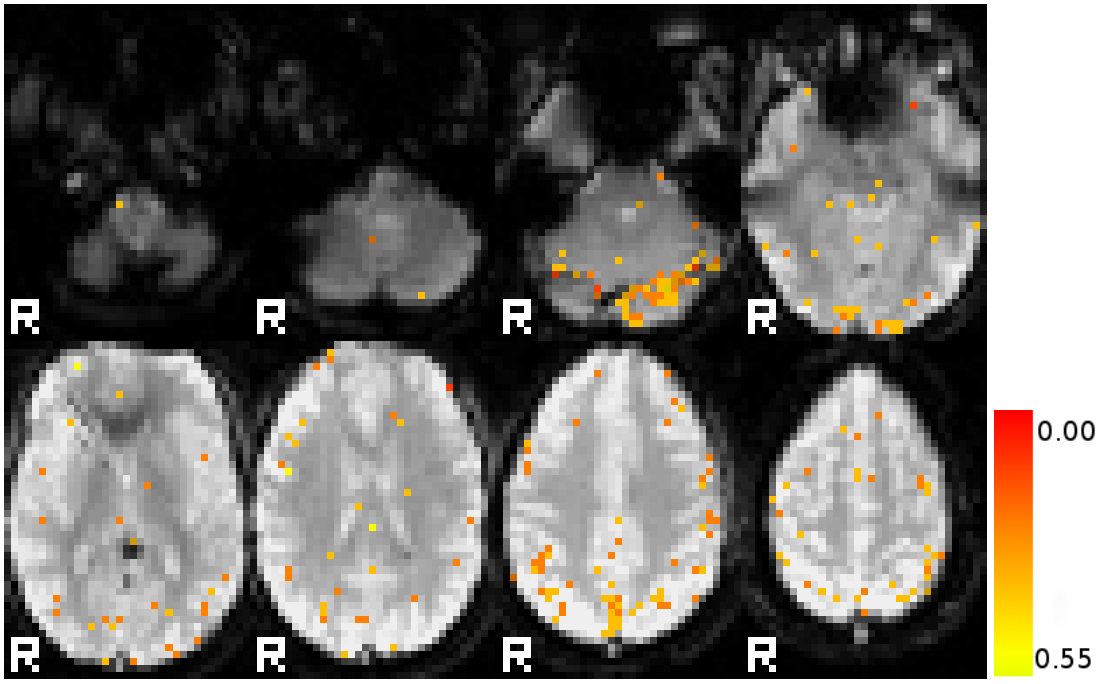


Figure 6.12: α Estimates

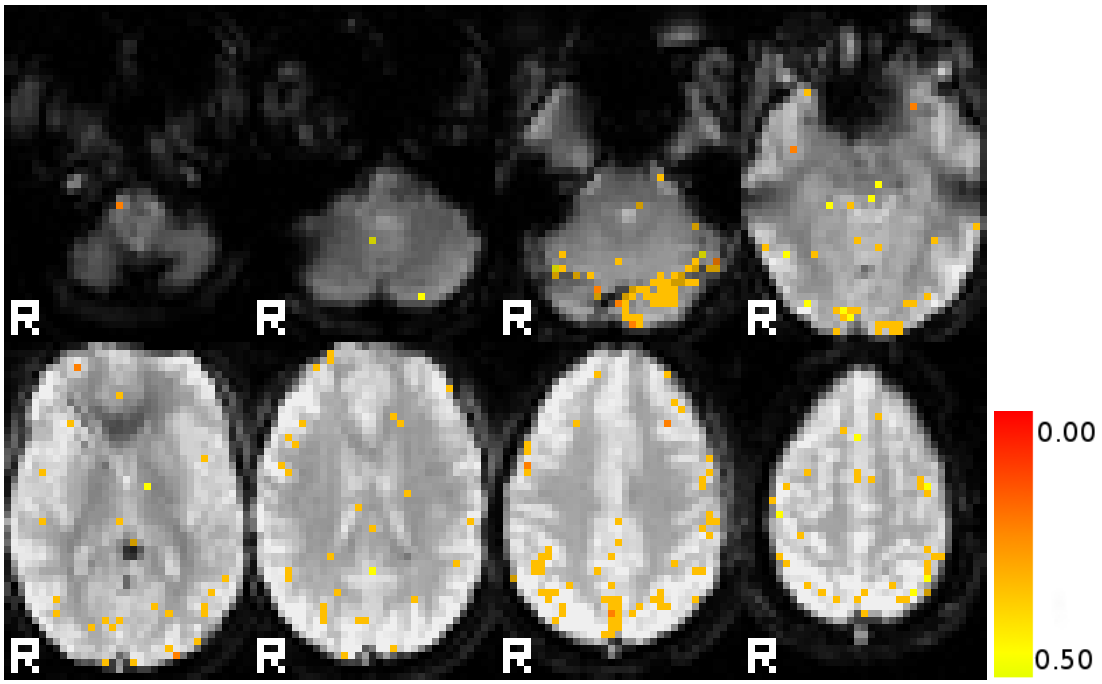


Figure 6.13: E_0 Estimates

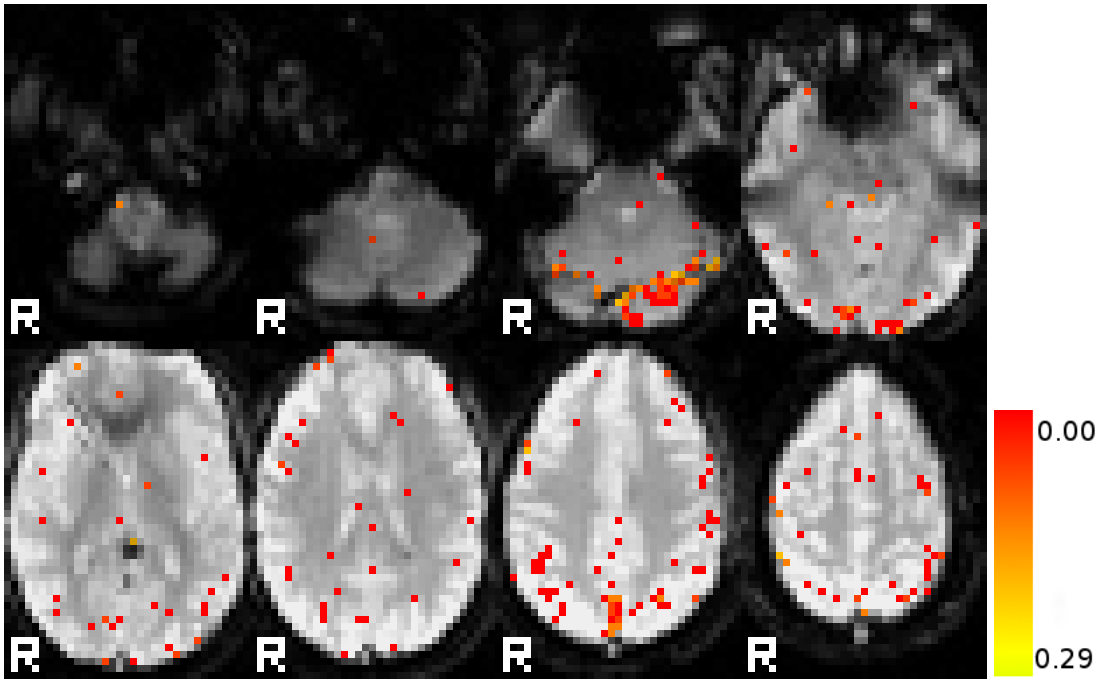


Figure 6.14: V_0 Estimates

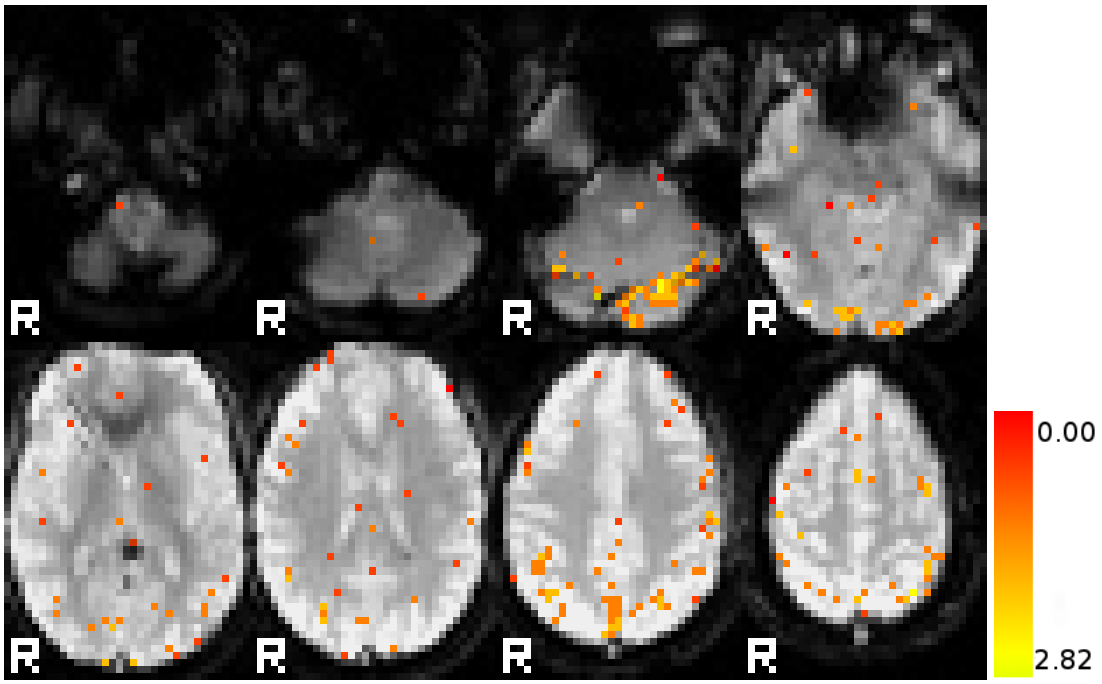


Figure 6.15: τ_f Estimates

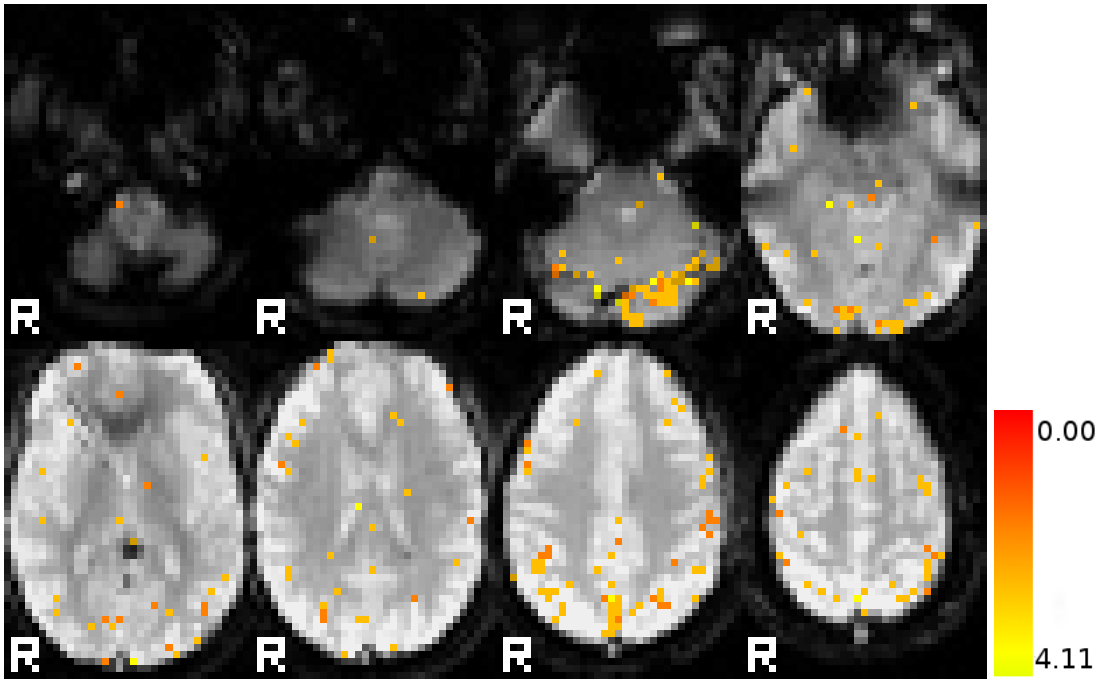


Figure 6.16: τ_s Estimates

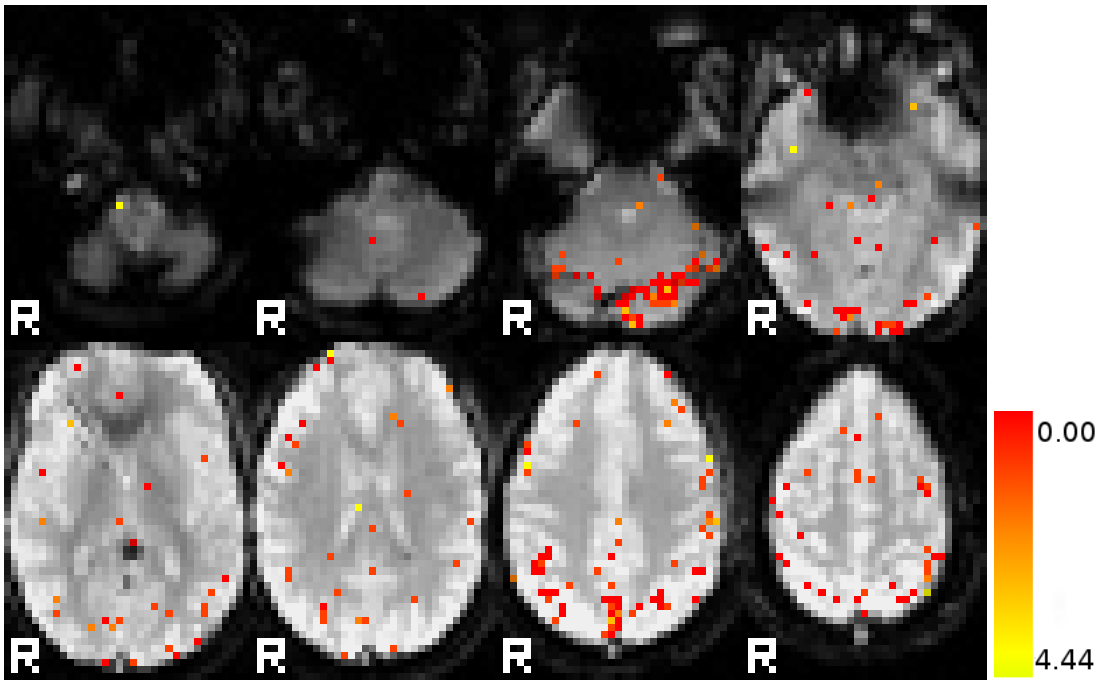


Figure 6.17: ϵ Estimates

Regions with poor fit cannot have reliable parameter estimates because the input did not meaningfully correlate input with output. Therefore before creating the maps, each parameters map was masked to regions with mutual information greater than .15. I chose mutual information over the residual because in the maps comparing regression fitness, the mutual information maps had more coherency and less randomness. Additionally, in the single voxel tests (Section 5.1.6) there was more separation between the no signal case and the signal case than there was in the residual metric.

The parameter maps show consistency parameter across active regions, however, the histogram of parameter estimates across all regions with $M.I. > .15$ is the more interesting result (Figure 6.18).

6.4 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. Mutual information also shows promising results. Many of the differences between SPM and the particle filter seem to be driven more by the pre-processing methods than the regression methd. The reason for SPM applying such extrensive smoothing is to combat false positives. While this is extremely

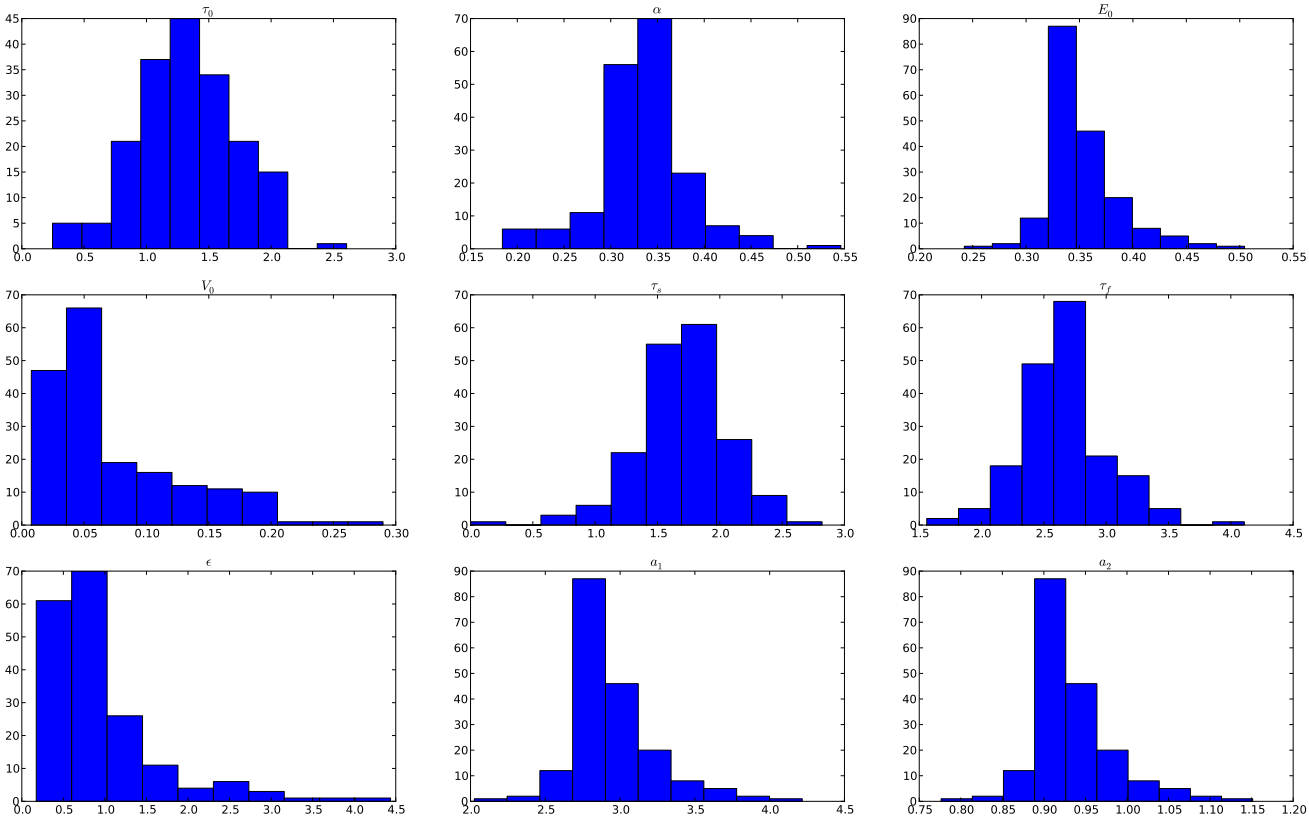


Figure 6.18: Histogram of parameters in active regions ($M.I. > .15$).

important, regions such as Figure 6.9 show the problem with doing so. In all, the particle filter method does very good job of estimating parameters that fit the measurements; and at the very least this method is a workable, albiet computationally intensive, alternative to SPM.

Chapter 7

Discussion

The results unequivocally show that the parameters of the BOLD model are under constrained. While unsurprising given the sensitivity analysis in [9], it is still an important limitation when calculating parameters. Thus, other methods that depend on a point estimate of the parameters, such as least squares or kalman filters (which uses the first two moments) are limited. While estimates of the BOLD signal may still be correct, the estimate of the underlying parameters and state variables are suspect. Similarly, as the histograms in Section 5.2 and Section 6.3 show, using the mean to estimate parameters also makes little sense in the particle filter case. Attributing the variance of estimate to the underlying parameter distribution would also be a mistake; given that variance persisted even in simulated data. While parameters definitely vary from person to person and cortex to cortex, the estimated distributions in Section 6.3 are wider than the true underlying distribution. For this reason and because of the nonlinear relation between parameters, any analysis should take the entire posterior estimate into account, rather than relying on an idealized Gaussian. In this sense, particle filters represent an important step forward in BOLD parameter estimation. Representing the uncertainty in parameters as a Gaussian is insufficient; so using a particle filter or Bayesian estimate of the posterior is not simply an enhancement, but a necessary precaution.

Although point estimates of parameters are not dependable estimates of the true parameters, they still useful. Analysis of differences in parameters could be medically relevant for instance. Additionally, a single estimate for parameters is still able to form an accurate estimate of the BOLD signal. This trait bridges the gap with earlier types of activation tests, such as the statistical parametric mapping. As the results in Chapter 6 show, the heatmaps, especially those of mutual information, closely resembled the results of SPM. While the activation tests were more sensitive, there were some false positive, though at this point it is difficult to quantify. It is worth noting that certain enhancements could be useful in reducing false positives; for instance by using the maximum likelihood, or median of the final distribution. Future work may shed further light on these techniques.

7.1 Particle Filter Review

The Particle Filter algorithm was originally designed for on-line parameter estimation. For this reason, there is no guarantee of optimality or even convergence for finite measurements. However, for the BOLD nonlinear ODE this is less of a concern than it might first appear to be. For this particular problem there can be no guarantee of a global minimum, and although other techniques guarantee a local minimum, the particle filter doesn't settle to one of these local minima because it can settle into multiple ones. This difference typifies the major difference between the particle filter and competing approaches.

One difficulty with the use of a particle filter with a finite number of datapoint is the calculation of a good weight function $P(y_k|x_k)$ that will converge reasonably quickly. If the weight function does not sufficiently differentiate particles, the final distribution will no be significantly different from the prior distribution. On the other hand, if the weight function is too thin, it will unfairly eliminate viable particles. Given sufficient measurements it is better to let the algorithm take longer to converge, because the convergence will be better (more accurate). The particle filter takes longer to run than Volterra approximation method from Section 2.2.1; however, it is free from the uncertainty of whether a quadratic approximation is sufficient for the BOLD model.

The particle filter certainly has significant advantages over other estimation procedures discussed in Chapter 2. The most important advantage is that it provides an estimate of the posterior probability, rather than a single estimate. While it is natural to want a simple estimate of parameters, such an estimate is impossible with this particular model. The results are more difficult to interpret, but this is a necessity. The fact that the final distribution is not dependent on any particular distribution is also advantageous. Of particular note; the final distribution does not need to conform to any parametric distribution. While the particle filter was not fast for full brain calculations, its speed was sufficient on a quad core machine to perform real time calculations of small regions (approximate run time .27 seconds per voxel-measurement). Today it would be possible to perform real time analysis of 10 voxels on an average quad core. The algorithm also scales well and does not require burdensome amounts of memory (approximately 11 megabytes). For this reason this algorithm is perfect for extension to vector or video card based processors.

A more practical benefit with the particle filter is that it is mathematically simple. An understanding of Bayesian statistics is all that is necessary to understand how the particle filter works. This is in contrast to the Volterra based approach of [2], which is quite complex. Additionally very few assumptions are needed for the particle filter. It is a common problem in traditional statistics for assumptions to be unrealistic which means the results may be invalid or at least in question. Fewer and more realistic assumptions mean that the particle filter is more robust to unforeseen difficulties in FMRI data.

Chapter 8

Future Work

8.1 Improving the Particle Filter

There are a few areas where future research may improve upon the current work. The first is modifying the Prior Distribution. The choice in this paper to use the distribution calculated in [2], may not be optimal and so is open to modification. The second major area is in preprocessing, in particular the method used to remove drift in the FMRI signal. This is an area of ongoing research and so future algorithms could be strengthened by applying the latest technique rather than a spline. Third, the experiment itself may be modified to improve results. And the last method is improving the , by using a more advanced version.

8.1.1 Prior Distribution

The prior distribution used in this work is listed in Table 4.1, and is based on the findings of [2]. Unfortunately, that result depended on a quadratic approximation (Volterra Series) which has not been extensively tested (at least not in a published work). As such, the prior is probably not complete. Additionally, the current prior is based only around the BOLD output, which is why it is capable of generating good estimates of the BOLD signal. However, it would be advantagous to have in-vivo estimates of the actual parameters. Better estimates could be found using population studies with additional measurements as discussed in Section 8.1.3. Changes to this prior could boost the power of the BOLD particle filter algorithm described here.

Starting with a flattened prior (by setting weights to be non-uniform after the initial particles have been drawn) was not used in the final analysis. The reason for not flattening the prior was a practical issue with weighting points in a 7 dimensional joint distribution. Often differences in particle weights approached machine precision, meaning that the flattened prior was actually far from flat. Instead the distribution became unpredictable, which made the results unpredictable. This is quantization issue that will exist unless the prior were made to be deterministic or the

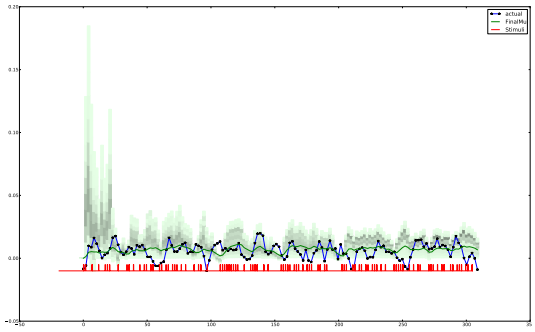


Figure 8.1: Large variance in time constants over-smoothing.

initial number particles was raised to computationally prohibitive levels.

Another improvement that could be made to the prior is increasing the initial variance. This would allow the particle filter to learn a wider range of parameters; at the cost of decreased convergence rate and support density. In tests, both real and simulated, increasing the variance tended to allow the particle filter to over-smooth (ex Figure 8.1). In essence the particle filter tended to converge to a mean where the time constants dominated the results by smoothing out all the peaks and troughs. The result is that no other parameters alter the estimated BOLD signal. In Figure 8.1, the particle filter still had not fully converged, and given more measurements it could still converge to a better estimate. Thus, increases in the prior variance necessarily must be accompanied by more measurements, although re-presenting data could be used to artificially extend the measurements.

8.1.2 Detrending

A large number of de-trending methods were tested, but all had some major problem. One viable method that could be utilized given the right experimental design is detrending based on areas of low activity. This has the advantage that it wouldn't require an arbitrary constant to be added to the pre-processed signal for the BOLD model to fit properly. The disadvantage of this approach is that it could hide long fall times by normalizing them out. It also requires periodic breaks in the stimulus, which could reduce value of those samples for fitting purposes.

Another potential way to deal with detrending is to linearize. This would use the delta between measurements for fitting rather than the direct value. This has the advantage of not requiring detrending and thus gives nearly raw data to the particle filter for processing. The effectiveness of this method depends directly on the type of stimulus. In an event driven stimulus with sparse stimuli this could be extremely effective because the DC level has minimal data; however the longer high levels are held the less effective this will be. On the other hand, splines will also reduce the plateaus, so its possible linearizing may be just as effective. I found in tests that large drift-low white noise type signals performed much better with a linearization approach, as one might expect. The results were equal to or worse in real data; but that was only for one particular

stimulus sequence.

A subtle difference that could be made in detrending is how the percent difference is calculated. In this work the spline was subtracted, and then the result was divided by the average of the initial signal. A more correct method may be dividing by the spline value at that particular point. The reason for not dividing by the spline at that point is that it could be less stable, and in a sense the trend has already been removed. The difference is not particularly large though (dividing by 1020 rather than 1000 for instance) as long as the spline didn't have heavy swings.

Finally, rather than adding a constant to the detrended data before applying the particle filter, a DC gain parameter could be added to the model. In tests, this could work well, however, it often did not. The problem with this could simply be the addition of another degree of freedom without any increase in the number of particles. Increasing the number of particles further though can be computationally intractable. Thus, while this is in a sense the correct solution, it is an impractical one.

8.1.3 Experimental Changes

One definite way of improving the results of the particle filter is more measurements. While increasing the sample rate of FMRI scanners may not be possible, simultaneous measurements of volume and flow is possible [28], albeit at 9T in a cat. Just one of those measurements though would be extremely powerful when incorporated into the BOLD model. By adding another measurement, especially for one of the other state variables (like blood volume and blood flow), the variance in the parameter estimates would certainly drop. Of course, more measurements may be gained by simply performing longer FMRI tests, or concatenating several together. Even more basic, its possible to artificially increase the number of measurements by presenting them the time-series several times. This activity is similar to the process used in neural networks, and gives the particle filter longer to converge to the optimal estimate. On the downside, this increases runtime and artificially reduces variance. Thus it does not necessarily improve the results. In tests, it did reduce the final covariance but did not lead to better estimates.

Additionally, given the non-linearities in the system, differences in stimuli can make a huge difference in the observability of parameters. The mentality for using a physiologically based nonlinear model for BOLD signal is to model those nonlinearities. Its logical then that nonlinear parameters may not be identifiable when the input is primarily an impulse response. Therefore, a wide range of inputs may shed further light on the parameters than found in this work.

As discussed in Section 4.3.3, choosing a weighting function is difficult. Automatically estimating measurement error could improve the quality of the particle filter results. Although I made some attempts to do this, finding a generic, consistent solution is complex, and often depends on the experimental design. This problem strikes at the very of heart of the difficulty in analyzing FMRI, and so it is not likely to be solved soon.

8.2 Applications

There are a number of advantages to the particle filter approach presented here. In the past FMRI data has been analyzed strictly for determining correlation between a stimulus and response. With this new method the correlation is simply a means to determining the joint posterior distribution of the parameters. While only regions that correlate with the input will be calculable, this method exploits that correlation to constrain the prior distribution of the parameters. The resulting distribution, while difficult to visualize because of the high-dimensionality, could nevertheless be correlated with neural pathologies. Despite the fact that the parameters are under-determined, the final distribution is still a reduction in the uncertainty of the parameters. This reduction in uncertainty naturally contains information which may be exploited based on non-parametric statistical analyses.

Because of the limitations present in every image modality in neuroimaging, its becoming increasingly clear that combining data from multiple sources will be necessary to push neurology forward. In order to do so however, the output of each source needs to fully represent what information that source can provide. Combining the sources using Bayesian statistics is promising yet often difficult because full probability distributions are hard to come by. However, in this case, the particle filter provides a full posterior which is extremely versatile. Therefore, future works will easily be able to plug in data from multiple sources if they all output Bayesian posteriors. For instance, if two different modalities have calculated the probability distribution of neural efficiency, those two beliefs may be combined into one conditional belief for the probability of neural efficiency.

An advantageous aspect of using a physiological model such as this, is that it permits estimates of otherwise hidden parameters. In particular the BOLD model gives an estimate of the value of the flow inducing signal, s. Having this value available opens up new avenues for determining inter-regional dependencies. All the regions for which the value of s are trustworthy correlate closely with the original stimuli, so determining the connection between their values of s would be less than edifying. Instead, the values of s, which in some sense is the activation in a particular region, could be used to drive other inputs. Thus, the particle filter could be re-run with the time-series of a particular voxel's s value as an additional stimulus. In this way, it could be possible to determine chains of events. This is just one possible benefit being able to determine the time course of the hidden state variables present in the BOLD equations; the potential benefits of being able to determine this information is limitless. Of course, improving the quality of the parameter estimates is necessary if such information is to be fully trusted.

Chapter 9

Conclusions

This work has demonstrated the use of the particle filter to learn parameters of the BOLD model. Since the inception of the BOLD equations, many attempts have been made use FMRI data to to learn these parameters. These attempts have typically either been extremely slow or relied on extensive assumptions. While the particle filter method is not quick, 60 seconds to analyze each voxel is well within the capabilities of the typical research lab. Previous attempts have also treated the problem as if there were a single solution.

One significant finding of this work, that would not be clear without calculating a full posterior distribution, is the interplay between the parameters. The results of simulations clearly demonstrate that identifying a single set of parameters is not possible with this model. Although sensitivity tests in [9] certainly hinted at this, the current work clearly demonstrates this fact. Therefore, any single estimate of parameters is insufficient for analysis. As such, approaches to the BOLD model that do not treat the parameters as distributions will not be able to overcome the inherent indeterminability of the parameters. Besides the Unscented Kalman Filter, this is the only approach that accomplishes this task without repeatedly calculating the parameters to build a distribution. The Unscented Kalman Filter, of course, is limited to Gaussian estimation which limits its ability to estimate the posterior.

The primary reason this method has not been used before is the high dimensionality of the system. In [36] the idea of learning the parameters is floated as the correct method, yet learning 7 parameters with a Monte Carlo method is often intractable. The concern was that so many parameters creates a prohibitively large search space, that requires too many particle to learn properly. To overcome this difficulty, instead of starting the algorithm with 1000 particles, the initial number of particles was set to 16,000. This meant that when the search space was the largest, the number of particles was sufficiently dense to represent the prior distribution. Then resampling was performed for the first time, the number of particles was dropped, since the weight of most the particles had dropped to 0. The result is a particle filter algorithm that spends computing resources only where they are really needed.

Many of the approaches to determining areas of neural activation aim to be Bayesian, yet ultimately must make limiting assumptions about the distributions that turn the approach into classical statistics. This approach is a true Bayesian non-parametric approach that given enough measurements will approach the true probability distribution of the parameters. While the conclusion that the parameters are not uniquely identifiable is a disappointing one in light years of research attempting to learn these parameters, it in fact further demonstrates the need to estimate probability distribution rather than a single parameter estimate. At the same time, the output of the particle filter is still about to give good estimates of the BOLD output, and not dependent on heavy Gaussian smoothing. Therefore, all the current experimental paradigms to determine regional activity are still viable. To borrow a computer term, the particle filter algorithm is backward compatible with the current methods.

Concluding, the particle filter provides comparable performance with conventional tests, but also provides a platform for a wide range of future uses beyond what is possible with the methods applied in the past.

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 CMRO 2 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, D. E. Glaser, R. N. a. Henson, S. Kiebel, C. Phillips, and J. Ashburner, "Classical and Bayesian inference in neuroimaging: applications.," *NeuroImage*, vol. 16, pp. 484–512, June 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] L. Murray and A. Storkey, "Continuous Time Particle Filtering for fMRI," *Advances in Neural Information Processing Systems*, vol. 20, pp. 1049–1068, 2008.

[37] 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.

[38] 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.

[39] C. Musso, N. Oudjane, and F. LeGland, "Improving regularised particle filters," *Sequential Monte Carlo methods in practice*, 2001.

[40] 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.

[41] K. J. Friston, "Bayesian estimation of dynamical systems: an application to fMRI," *NeuroImage*, vol. 16, pp. 513–530, 2001.

[42] 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.

[43] A. M. Smith, B. K. Lewis, U. E. Ruttimann, 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.