BOLD Parameter Estimation Using Particle Filters

Micah Chambers

Virginia Polytechnic Institute and State University

Contents

Chapter 1

Introduction

For the past twenty years Functional Magnetic Resonance Imaging (FMRI) has been at the forefront of cognitive research. Despite it's 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 relatively quick and cheap, and its analysis relatively straight forward.

By modeling the governing equations behind the neural response that drives FMRI, it is possible to make the analyses of FMRI even more powerful. 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 observable FMRI signal contains more information than which regions respond to which 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 [1]. The seminal work by [2] attempted to explain the time evolution of the BOLD signal using a windkessel model to describe the time local changes in Deoxygenated Hemoglobin content. Various papers made substantial improvements to this model until [3] 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 [4], even the most basic version has been shown to have less bias error than the convolution based "Canonical Hemodynamic Model" [4], [5]. On the other hand many of the BOLD signal models have far more parameters than a simple scaling parameter. In fact, the number of parameters range from seven [6] to 50 [7] per signal time course; a signal which may be as short as 100 samples long. Thus, even in a small FMRI image of the brain (20x20x20), the number of parameters could easily exceed 10,000. Clearly this number of parameters presents a significant risk of being under-determined and could suffer catastrophic variance error; to make no mention of the computation cost. In this work a method of countering these problem is presented with the use of a particle filter.

1.1 Overview

Detecting neural activity using the changes in FMRI images is based on the so called Blood Oxygen Level Dependent (BOLD) signal. The BOLD signal is caused by minute changes in the ratio of Deoxygenated Hemoglobin to Oxygenated Hemoglobin in blood vessels throughout the brain. Because Deoxygenated Hemoglobin (DHb) is paramagnetic, higher concentrations attenuate the signal detected during T2-weighted Magnetic Resonance Imaging (MRI) techniques. The most common FMRI imaging technique, due to its rapid repetition time (TR), is Echo Planar Imaging (EPI). When axons 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 very active, the action potentials occur significantly more often, resulting in significant local increase of the Cerebral Metabolic Rate of Oxygen (CMRO2). Thus, blood vessels in a very active area will tend to have less oxygenated hemoglobin (due to the increased rate at which oxygen is being consumed), and more deoxygenated hemoglobin, resulting in an attenuated FMRI signal. In compensation for activation, muscles that control blood vessels relax in that region to allow more blood flow, which actually overcompensates. This ultimately results in lower than average concentration of deoxyhemoglobin. Thus, the BOLD signal consists of a short initial dip in the MR signal, followed by a prolonged increase in signal that slowly settles out. It is this overcompensation that is the primary signal detected with FMRI imaging. This cascade of events is believed to consist of increased the local metabolism, blood flow, blood volume, and oxygenated hemoglobin. The differences in onsets of these various effects is what causes the overcompensation that is observable in FMRI. Unfortunately, FMRI has no inherent unit of measurement, and thus signal levels are all relative: within a particular person, scanner and run.

1.2 FMRI

Magnetic Resonance Imaging, MRI, is a method of building 3D images non-invasively, based on the difference between nuclear spin relaxation times in various molecules. First, the subject is brought into a large magnetic field which causes nuclear spins to align. Radio Frequency (RF) signals may then be used to excite nuclear spin away from the base alignment. As the nuclei precess back to the alignment of the magnetic field, they emit detectable RF signals. Conveniently, the excitation of nuclear spins return their original state at different rates, called the T1 relaxation time, depending on the atoms excited. Additionally, the coherence of the spins also decay differently (and quite a bit faster than T1 relaxation) based on the properties of the region. This gives two

primary methods of contrasting substances, which form the basis of T1 and T2 weighted images. Additionally, dephasing occurs at two different rates, the T2 relaxation time, which is unrecoverable, and $T2^*$ relaxation, which is much faster, but possible to recover from via special RF signals. T1 relaxation times are typically on the order of seconds if a sufficiently strong excitation was applied. In order to rapidly acquire entire brain images, as done in Functional MRI, a single large excitation pulse is applied to the entire brain, and the entire volume is acquired in a single T1 relaxation period. Because the entire k-space (spatial-frequency) volume is acquired from a single excitation, the signal-to-noise-ratio is very low in this type of imaging (Echo Planar Imaging).

Increasing the spatial resolution of EPI imaging necessarily requires more time or faster magnetic field switching. Increasing magnet switching rates though is difficult, because it can result in more artifacts, or even lower signal to noise ratios. The result is that at *best* FMRI is capable of 1 second temporal resolution. The signal is further diluted because each voxel contains the signal from a large number of neurons, capillaries and veins. Thus, the FMRI signal, which is sensitive to the chemical composition of materials, is the average signal from various types of tissue in addition to the blood. As mentioned in section 1.1, and explored in depth in section 1.3, the usefulness of FMRI comes from the discerning of changes in Deoxyhemoglobin/Oxyhemoglobin. Therefore, it is necessary to assume that in the short term the only chemical changes will be in capillary beds feeding neurons. In practice this may not be the case, for instance near significant veins, and it may explain some of the noise seen in FMRI imaging (see section 1.5. Because MRI lacks units and certain areas will have a higher base MR signal, all FMRI studies deal with percent change from the base signal; rather than raw values. This also removes most of the structural data which is not helpful in determining neural activity.

1.3 BOLD Physiology

It is well known that the two types of hemoglobin act as a contrast agents in EPI imaging [2], [8], [1], however the connection between Deoxyhemoglobin/Oxygenated Hemoglobin and neural activity is non-trivial. Intuitively, increased metabolism will increase Deoxyhemoglobin, however blood vessels are quick to compensate by increasing local blood flow. Increased inflow, accomplished by loosening capillary beds, precedes increased outflow, driving increased blood storage capacity. Since the local MR signal depends on the ratio of Deoxyhemoglobin to Oxygenated Hemoglobin, increased volume of blood can effect this ratio if metabolism doesn't exactly match the increased inflow of oxygenated blood. This was the impetus for the ground breaking balloon model ([2]) and windkessel model ([9]). These works derive from first principals the changes in deoxyhemoglobin ratio and volume of capillaries based on a given flow. These were the first two attempts to quantitatively account for the shape of the BOLD signal as a consequence of the lag between the cerebral blood volume

(CBV) and the inward cerebral blood flow (CBF). In fact [2] went so far as to show that a simple, well chosen blood flow waveform coupled with a square wave cerebral metabolic rate of oxygen (CMRO2) curve, in the context of a balloon model, could fully account for the BOLD signal.

Although [2] demonstrated that a well chosen flow waveform could explain most features of the BOLD signal, there was still a matter of proposing a realistic waveform for the CBF and for the CMRO2. [10] gave a reasonable and simple expression for CBF input,f, based on a flow inducing signal, s, in combination with the original balloon model where v is normalized cerebral blood volume (CBV), q is the normalized local deoxyhemoglobin/oxygenated hemoglobin ratio.

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

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

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

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

where ϵ is a neuronal efficiency term, $u(t)$ is a stimulus, and τ_f, τ_s are both time constants, E_0 is the resting metabolic rate and α is Grubb's parameter controlling the balloon model.

This completed the basic balloon model, and was well summarized again in [11]. [12] 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:

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

$$k_1 = 4.3 \times \nu_0 \times E_0 \times TE = 2.8 \tag{1.6}$$

$$K_2 = \epsilon_0 \times r_0 \times E_0 \times TE = .57 \tag{1.7}$$

$$k_3 = \epsilon_0 - 1 = .43 \tag{1.8}$$

Where $\nu_0 = 40.3s^{-1}$ is the frequency offset in Hz for fully de-oxygenated blood (at 1.5T), $r_0 = 25s^{-1}$ is the slope relating change in relaxation rate with change in blood oxygenation, and $\epsilon_0 = 1.43$ is the ratio of signal MR from intravascular to extravascular at rest. Although, these constants change with experiment (TE, ν_0, r_0), patient, and brain region (E_0, r_0), often the estimated values taken from [12] 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 very accurate, it is not perfect.

1.4 Post Stimulus Undershoot

Although the most widely used, the BOLD model described in Equation 1.4 and Equation 1.8 have been extensively added on to. The most significant feature missing from the original model is the ”post-stimulus undershoot”. The ”post-stimulus” undershoot is the name for a prolonged subnormal BOLD response for a period of 10 to 60 seconds after stimulus has ceased ([13], [14]).

Because Equation 1.4 is not capable of producing such a prolonged undershoot, additional factors must exist. Two theories exist for the post stimulus undershoot. Recall that a lower than base signal means that there is an increased deoxyhemoglobin content in the voxel. The first and simplest explanation is that the post-stimulus undershoot is caused by a prolonged increase in CMRO2 after CBV and CBF have returned to their base levels. This theory is justified by quite a few studies that show CBV and CBF returning to the baseline before the BOLD signal ([15], [16], [3], [17], [18]). 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 seems to indicate that there can be a prolonged supernormal CBV ([14], [7], [19]), 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 [20]. [13] makes a compelling case that most of the post stimulus undershoot can be explained by combination of a prolonged CBV increase, and a prolonged CBF undershoot, and that many of the previous measurements showing a quick recovery of CBV were in fact showing a return to baseline by arterial CBV.

Regardless of the probability that CMRO2 and CBF are detached, research into the post-stimulus undershoot has led to the creation of much more in depth models. In [**?**] additional state variables model oxygen transport, whereas [3] models CMR02 from a higher level, and somewhat more simply; though it still adds several new parameters. [7] 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 [13]. Similarly [21] adds additional compartments to model the BOLD signal that result from venous and arterial blood. [4] compared these various models and though it did not deal extensively with the post-stimulus undershoot, it did show incremental improvements in quality from additional parameters (over the basic Balloon model); though at the cost of greatly increased complexity. Importantly,[4] did show that by simply adding viscoelastic terms from [3], a slowed return to baseline is possible to model, without greatly increasing complexity. Regardless, because of many of these models are extremely complex, and the benefits have yet to be proved, in this work the simple Balloon model will be used. Simplicity is even more important because it is the first time a particle filter has been used to calculate the BOLD parameters. Of course, future works could certainly benefit from the more advanced models, especially with the addition of viscoelastic effects.

In summary, there have been extensive attempts to refine the Balloon, many of them extremely precise. However, the trade off between increased complexity and increased flexibility is extremely important. Given that the basic BOLD model has 7 parameters, for this work, where computation time is very important, that model is the best fit.

1.5 Noise

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

A particularly difficult form of noise present in FMRI is a low frequency drift, often characterized as a Wiener process ([6]). Though not present in all regions, as much as ten to fifteen percent of voxels can be affected ([22]), thus it is prevalent enough to cause significant inference problems [23]. It is still not clear what exactly causes this noise noise comes from, although it is possible it is the result of magnets heating up, or some distortion in magnetic fields [23]. It is clear that this drift signal is not solely due to a physiological effects, given its presence in cadavers and phantoms [24]. Interestingly, it is usually spatially correlated, and more prevalent at interfaces between regions, though by no means limited to such areas. Though one potential source could be slight movement, given that co-registration of volumes to a single time point is standard, this seems unlikely. Regardless, the problem mandates the use of a high pass filter to make inference reasonably powered [23].

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

Because most methods (including the one used in this paper) assume the noise realizations are independent of each other, the auto- correlation is of particular interest (which is a necessary but not sufficient condition for independence). Gaussianity is also a common assumption made in studies of FMRI data, though that assumption is not made in this work. Regardless, comparing the distribution to a Gaussian is informative, so I used Q-Q

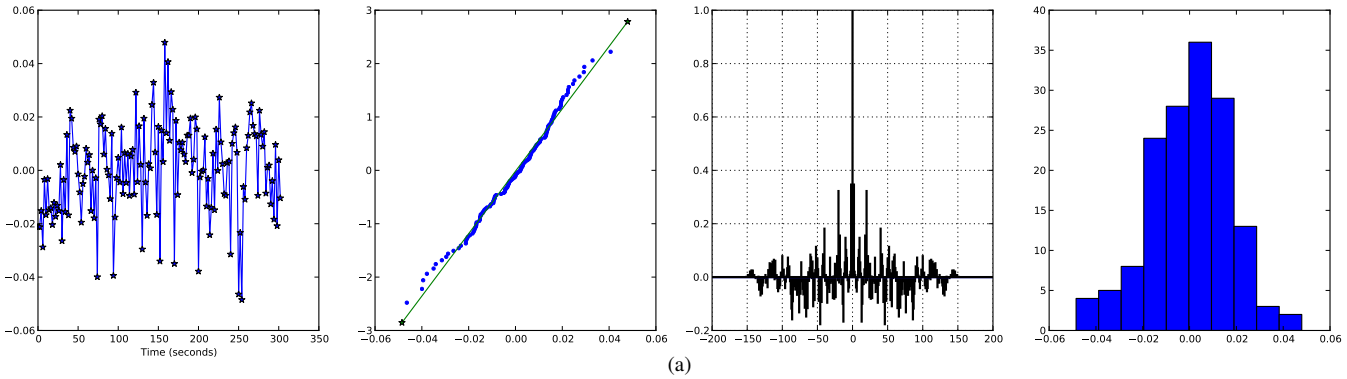
plots to compare example data with the Normal distribution. Additionally, in FMRI data the noise is often considered to be Wiener [11]. Recall that a Wiener random process is characterized by steps that are Gaussian; in other words the difference between any two samples is Normally distributed. The simulations discussed in **??** make use of this, by adding a Wiener random process to the overall signal. To determine whether the noise is in fact Wiener, the distribution of the difference between adjacent measurements was plotted against a Gaussian. Wiener processes further assume that the steps are independent; so the steps should have near flat autocorrelation functions.

Finally, removal of the so called "drift" is often performed with some variation of a high pass filter, so I checked the noise distribution after applying such a filter (in this case the subtraction of a spline, see section 4.5). Here I wanted to know how effective my high pass filter at the removal of Wiener noise.

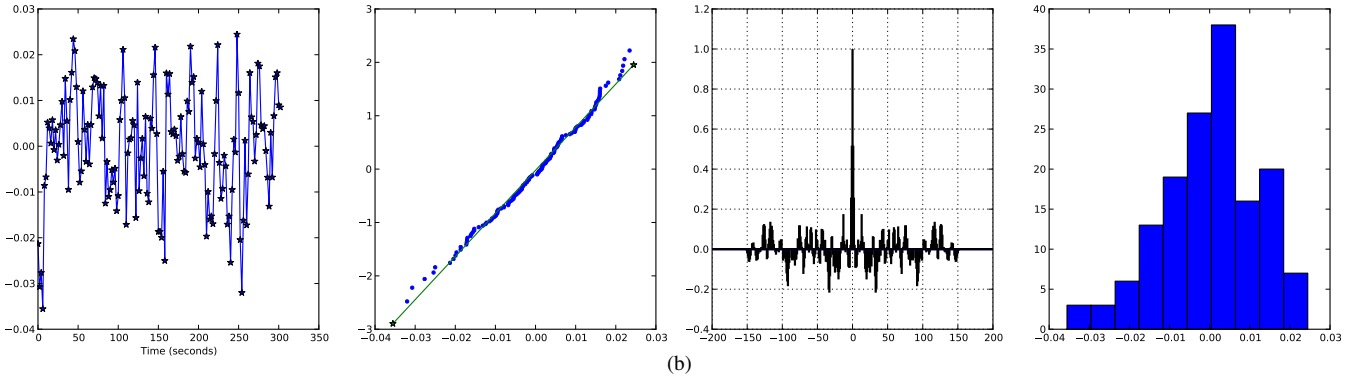
Figure 1.1 shows the results with a regression line fit to the points on the Q-Q plot. Recall that in a Quartile-Quartile (Q-Q) plot, if the points plotted on the x-axis and the points on the y-axis come from the same *type* of distribution then all the points will fall on a line. Differences in the variance will cause the line to have a slope other than 1, while differences in the expected value will cause the fitted line to be shifted. In these Q-Q plots, the points are being compared to the standard Gaussian distribution, so the quality of the line fit determines how closely the points fit the Gaussian distribution. Note that in Figure 1.1 the points have all been normalized (changed to percent difference).

A Wiener process should still conform to the normal distribution, with a variance proportional to the run-time. Note that 1.1(a) and 1.1(b) are relatively well described by a Gaussian process with a small autocorrelation, 1.1(c) and 1.1(d) are not. In particular the tails of 1.1(c) do not seem to fit the Gaussian well. Also note the significant autocorrelation in 1.1(c) and 1.1(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 assumption that the noise follows a Wiener process seems to be correct.

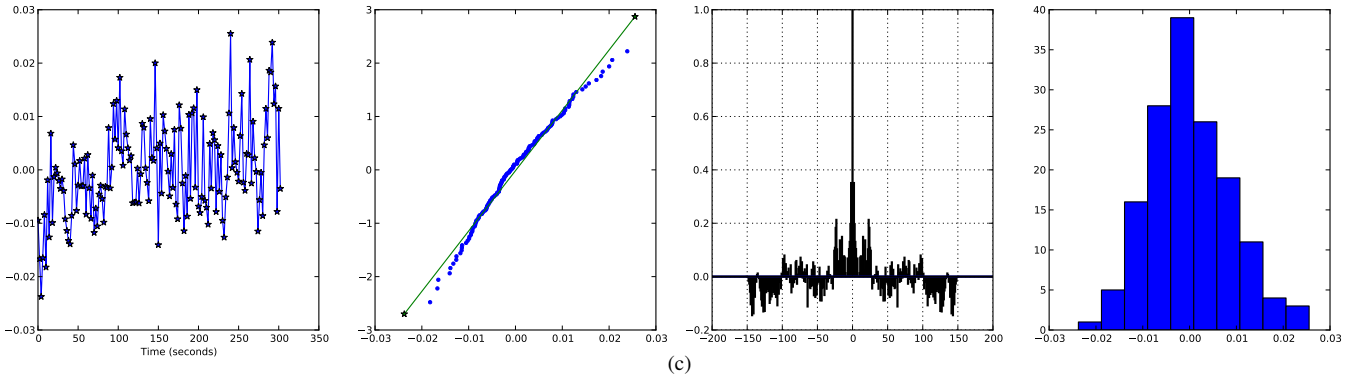
De-trending the time-series by subtracting a spline fit to the distribution removed much of the autocorrelation present in 1.1(c) and 1.1(d), though clearly not perfectly. Though the distributions still do not exactly fit the Normal, 1.3(d) clearly is much improved compared to 1.1(d). In all, the de-trending is effectively removing Wiener noise.



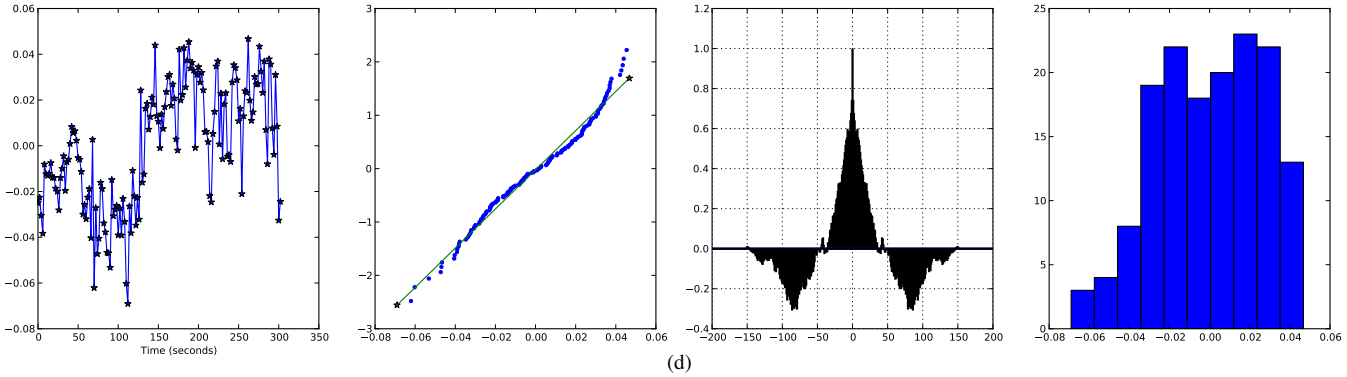
(a)



(b)

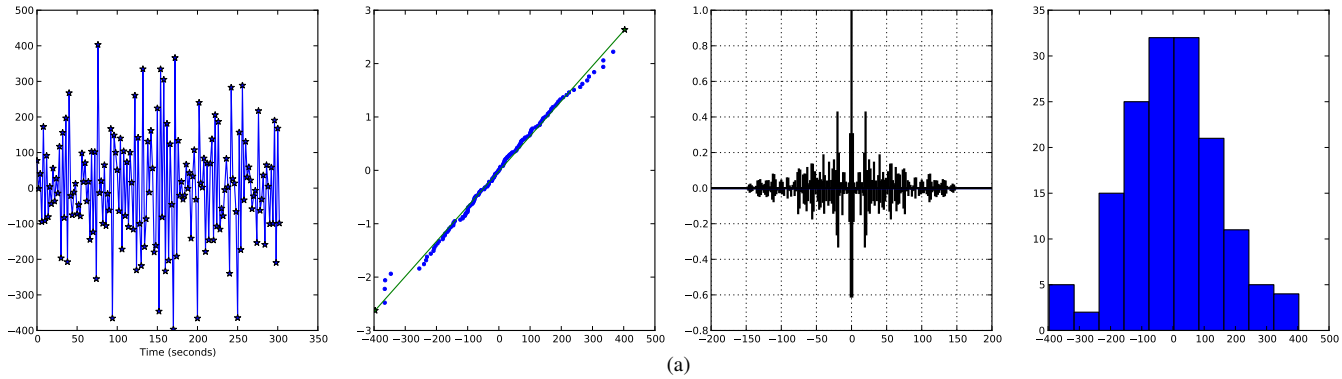


(c)

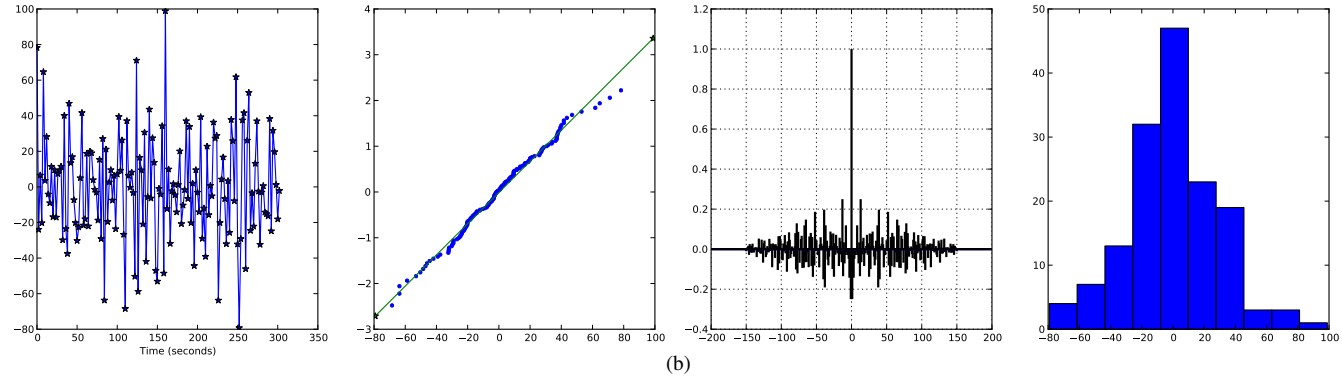


(d)

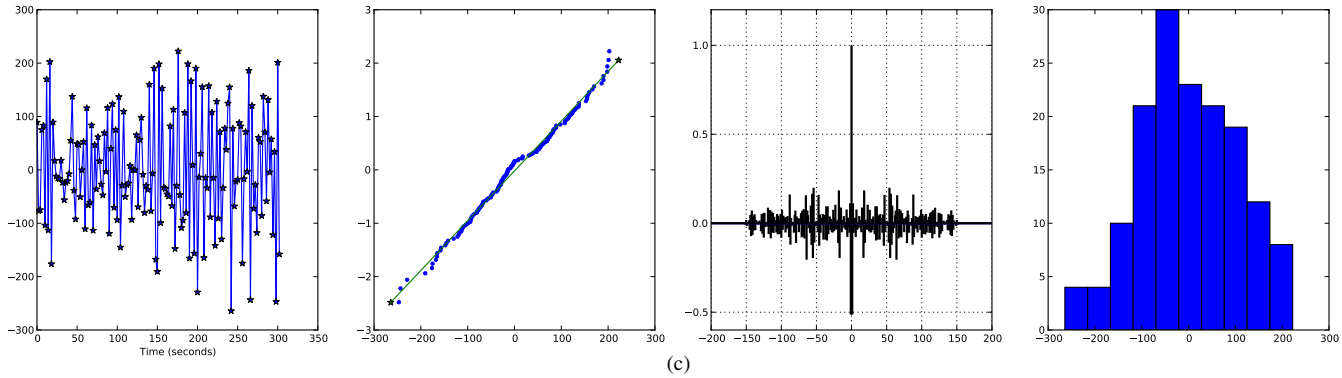
Figure 1.1: Q-Q Plots of normalized resting state data



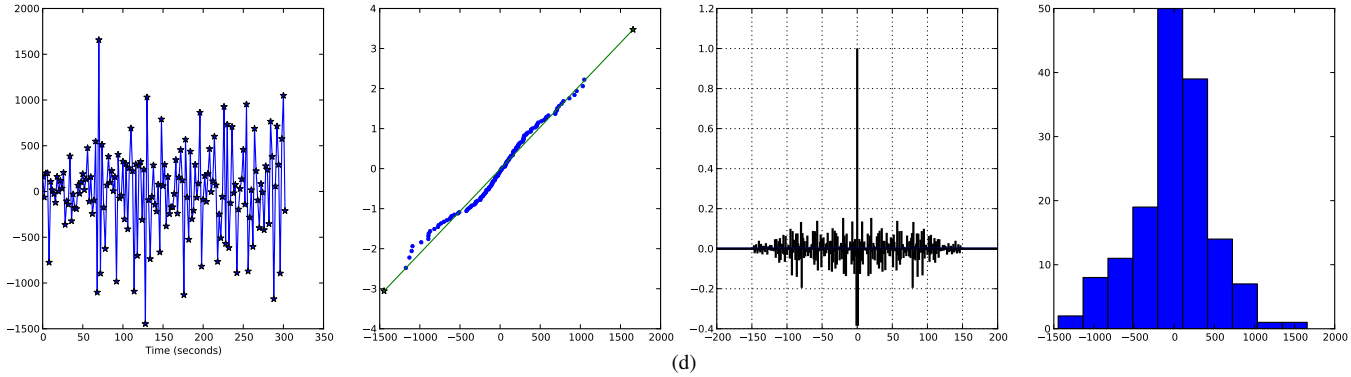
(a)



(b)

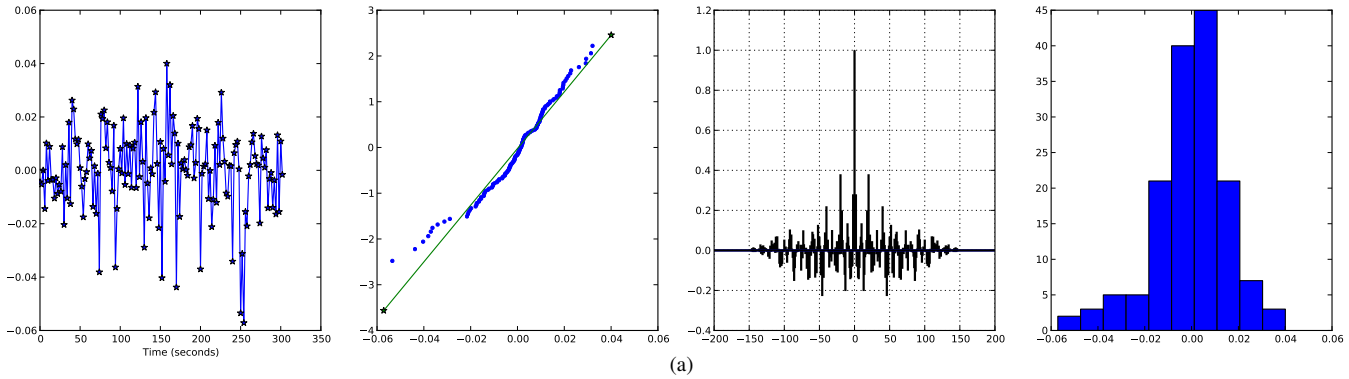


(c)

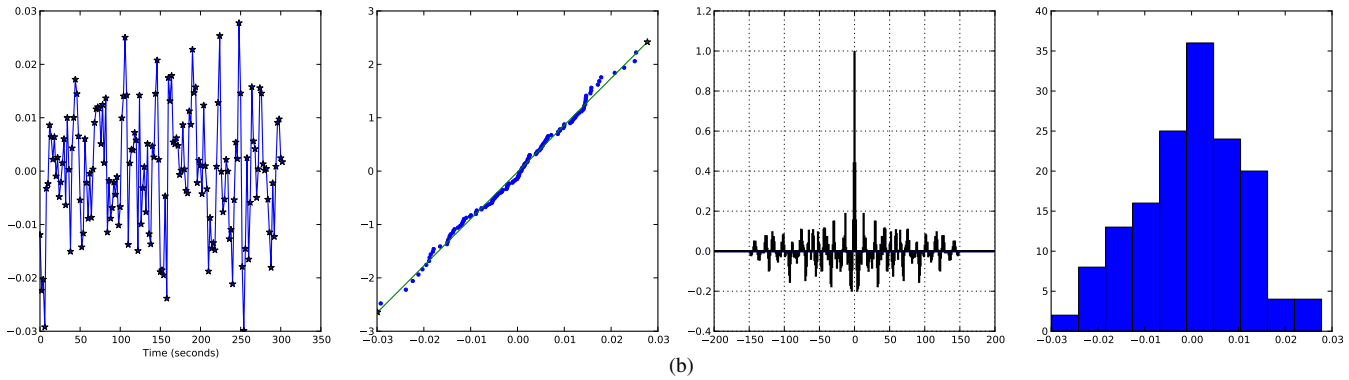


(d)

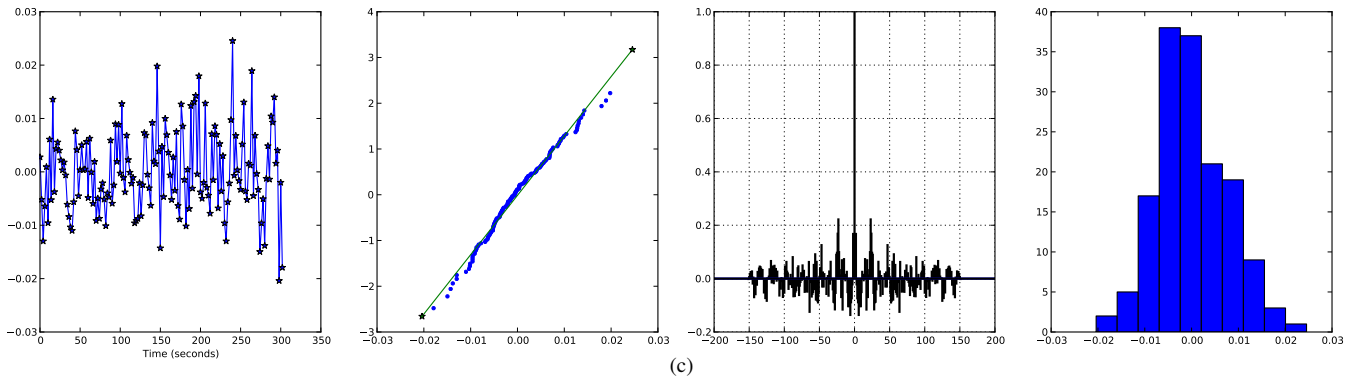
Figure 1.2: Q-Q Plots of resting state data, using the BOLD signal changes



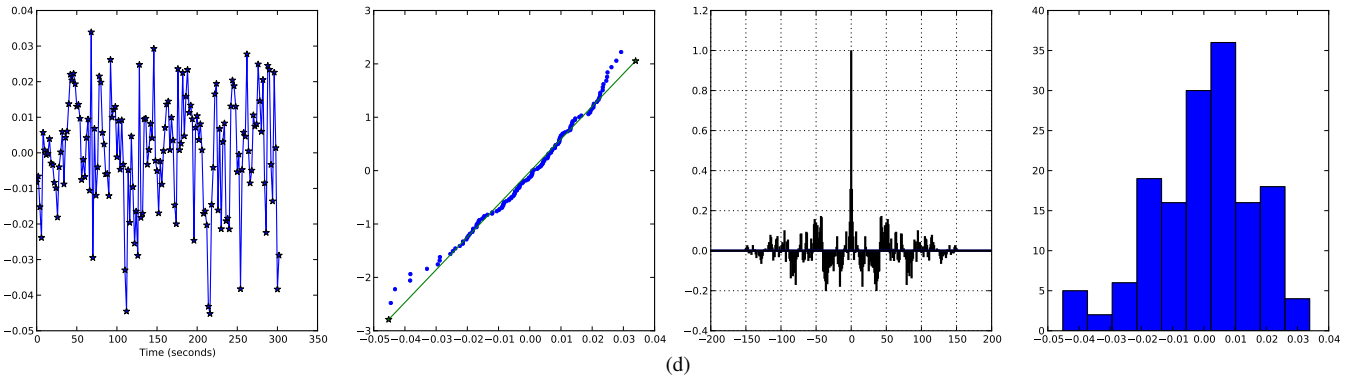
(a)



(b)



(c)



(d)

Figure 1.3: Q-Q Plots of resting state data, after the de-trending

1.6 Properties of the BOLD Model

Since the first complete BOLD model was proposed by [25], several studies have endeavored to analyze properties of that model. The most important property is that the system is dissipative, and given enough time will converge to a constant value. This is found simply by analyzing the eigenvalues of the Jacobian of the state equations, ([4], [26]). The steady state of the Balloon model equations gives:

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

In real FMRI data, there is a significant nonlinearity in response; with short sequences responding disproportionately strong ([27], [28], [4]). This nonlinearity is accounted for in the Balloon model, although [4] shows that if there will be large variance in the length of signals, modeling Neural Habituation may be necessary to fully capture the range of responses. Stimuli that last longer than 4 seconds tend to be more linear, which is why block designs are so well accounted for by the General Linear Model ([27], [4]). For this paper we will use only a simple version of the Balloon model, described by Equation 1.4, to keep the solution tractable, and because it is the most well studied.

Another interesting result of [4] was the sensitivity analysis. There it was found that the parameters are far from perpendicular, implying that exact inference of parameters may not be possible without constraint. This could explain the extreme discrepencies in Table 1.1.

Several studies have been able to calculate distributions for the Balloon parameters, those results are summarized in Table 1.1

Necessary? [Image with two different α's] [image comparing the results of 10% changes in various signals]

Parameter	[10]	[29]	[30]	[4]
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. [30] made use of the values from [25] where not explicitly stated

Chapter 2

Prior Works

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

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

2.1 Statistical Parametric Mapping

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

2.1.1 Classical Activation Detection

The most basic method of analyzing FMRI data is through a standard T-test between "resting state" and "active state" samples. Simply put, the mean is calculated separately for non-stimulus and stimulus time intervals. A classic t-test may then be applied, giving the probability that the distributions actually have the same mean. Because of the correlated noise present in FMRI (section 1.5), it is necessary to apply some sort of high-pass filter to the data. Without applying such a filter, P values must be set extraordinarily high to prevent false positives [23]. 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 1.3). For this reason other methods are often more used, as discussed in subsection 2.1.3.

2.1.2 Random Field Theory

SPM methods make significant use of T-Tests across large regions; however, such T-tests work slightly differently than a single individual test. A t-test with a p-value of .05 over a modestly sized FMRI image, say 10,000 voxels, will on average generate 500 false positives. This is called the multiple comparison problem. Traditional Bonferroni Correction deals with this by requiring each test to pass with P value of $\frac{.05}{10000}$. The probability of a single false positive would then .05. Unfortunately this leads to unrealistically low p-values; so low that it would be impossible for any biological system to satisfy. To compensate, a Gaussian kernel is applied to smooth the image. This has the benefit of reducing the noise variance and decreasing the effective number of independent measurements. Because the number of independent measurements is smaller, Bonferroni correction can theoretically be applied with a lower scaling factor than the original voxel count [31]. A side effect of this, a single voxel activation is virtually impossible to detect.

2.1.3 General Linear Model

The most common FMRI analysis technique is SPM, though there are more advanced versions that the simple square wave method discussed in subsection 2.1.1. Hierarchal Models are one important improvement that allows researchers to combine data across multiple runs, patients and stimuli (see [32] for more on Hierarchical Modeling). Hierarchical Models concatenate all the data into a single dataset, then perform a linear fit between a design matrix and the data. The design matrix encapsulates all known experimental factors such as stimuli, young/old, etc. The equation for a general linear model is:

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

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 significantly 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

Figure 2.1: Hemodynamic Response Function todo

so called "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. The "Canonical" HRF that is most used, has been optimized for the visual cortex. As section 1.4 discussed, there are certainly variations in the shape of the BOLD response, both between brain regions and patients. [5] discusses the implications of choosing an incorrect HRF, the most important of which is a definite increase in false negatives. While an atlas of Hemodynamic Response Functions for each region could definitely mitigate this risk, it does not deal with variation between patients. Thus, the inability to fit parameters other than scale definitely hinders analysis of activation. As a result, there is significant bias error when using a linear fit to localize activation. Notably, studies will also have a bias toward the visual cortex, precisely because it is so well studied.

2.1.4 Hierarchical Linear Models

As mentioned previously and discussed extensively in [25] and [**?**], hierarchical models may be applied to account for systematic differences between subjects. For instance, if the study happens to be a mix between young and old, incorporating that into the model is wise, regardless if that is the purpose of the test. The reason to do this is to account for additional variance that may not be present in similar studies. The Hierarchical form used by [25] is shown in **??**.

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

The Empirical Bayes algorithm is used in both in both [25] and [**?**]. As a consequence, point estimators are used for each θ, rather than the full distributions.

2.1.5 Conclusion

In all, the GLM is extremely 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 relatively common scenario where no significant activation can be found in a single run [6] [29].

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

2.2 Solving the BOLD Model

Unlike Statistical Parametric Mapping, the techniques described in this section are all attempts to learn some version of the BOLD model. There have been quite a few efforts to quantify the parameters of the various BOLD models. Although [2] and [10] both proposed physiologically reasonable values for the model parameters, [33] was the first paper to calculate the parameters based on actual FMRI data. However, in that case, the voxels were chosen from regions that were detected as active by the GLM. It is therefore possible that the parameters are biased toward parameters that fit the linear model.

2.2.1 Linear Approximation

In [33], a novel combination of linear and nonlinear modeling was used to generate parameter estimates. Because it is impossible to calculate the partial derivative of the output with respect to parameters, $\frac{\partial\theta}{\theta}$, [33] approximates the partial. At each step of the Expectation- Maximization algorithm, the differential equation is integrated, calculating the residuals. Then, for each θ_i surrounding the current estimate of θ, a Volterra-Kernel expansion of the output y is generated. Generation of the Volterra Kernel is relatively quick, and, since it is linear there is an analytical solution. Thus, once the Volterra Kernel exists, the value of y at that measurement point, for that θ_i can easily be found. Thus, one part of the partial is calculated at one point using $\frac{y(t)-y_i(t)}{\theta-\theta_i}$.

This is repeated for every dimension of θ to give the full $\frac{\partial y}{\theta}(t)$. This is only at time t though, and it must be repeated for every measurement. Finally the full derivate matrix is filled out, and the next step in the E-M algorithm can proceed. The full E-M algorithm for estimating the states is notation-heavy and can be found in [33].

Although this is certainly an interesting method of performing non-linear regression, there are a few caveats. First, the partials are numerical approximations, based on an approximate values (using Volterra-Kernels) of y. Importantly, the Volterra-Expansion of y is not able to model interactions between state variables; interactions that were found to cause interesting behavior in this work. In subsection 2.2.3, for the purpose of demonstrating nonlinearities in the signal, it was necessary to propagate state variables through 1 second of simulation. For the distribution of state variables mentioned in Table 2.1 it was necessary to use step sizes smaller than .001 to prevent unpredictable behavior. This is using the average parameters, thus the only complication came from the interplay between states, which can certainly lead to interesting behavior. Additionally, all the tests performed in [33] were on regions found to be active by the General Linear Model. For this reason, the reliability of the approximation is unknown for regions that are active but sufficiently nonlinear to avoid detection by conventional tests.

2.2.2 Nonlinear Least Squares

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

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

Although anything requiring a Jacobian is out, there are other heuristic techniques that could potentially illuminate the BOLD response. Simulated Annealing (SA) is a common method of optimizing high dimensional regression problems. The idea is to pick a random start, and then at each iteration pick a random nearby point, and if that point is below some energy constraint (energy is a function of the residual), called the temperature, the algorithm moves to that point and continues with the next iteration. The temperature is slowly lowered until no nearby points below the temperature can be found (or the temperature drops below the current point). There are variations of this, for instance it is common to require every movement to be in the downward direction

(in terms of energy). Like most nonlinear optimization problems, there is no guarantee of an optimal solution, although the longer the algorithm is allowed to run, the better the solution will get. Since every step requires an entirely new run of the BOLD model, it can be extremely time consuming, which is why we are not using it here.

Algorithm 2.1 Simulated Annealing Algorithm

```
Initialize 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
```

[simulated annealing image?]

Another potential method of interest is the use of Genetic Algorithms (GA). Genetic algorithms are similar to Simulated Annealing, in that they randomly move to better solutions based on a cost function. However; in genetic algorithms a single point estimate isn't used. Instead a population of estimates is generated, each with distinct parameters, and then each set of parameters is rated with a fitness function. Parameter sets that are good get a higher "fitness"; then new parameter sets are generated by randomly combining pieces of the old parameter sets. The pieces are typically chosen at a rate proportional to the fitness of the donor; thus "fit" parameter sets tend to pass on their properties. In addition to this, random mutations may be introduced that come from no existing parent. The new generation is then rated with the fitness function again, and the entire process starts over. The stop condition for a genetic algorithm is typically based on some threshold for fitness or a maximum number of generations. This is problematic, since it doesn't really provide any guarantee of even reaching a local minima, although in practice it can be quite effective.

[genetic algorithm picture]

Algorithm 2.2 Genetic Algorithm

```
Initialize N estimates, E = {Θ_0, Θ_1, ...Θ_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
    At some low probability change one or two parameters in Θ_i
  end for
end for
```

Although both these methods can be highly effective, they have the downside of requiring very high com-

putation time. In this case of the BOLD model, each time the energy or fitness needs to be calculated, a large number of cycles must be spent re-simulating the BOLD model for the set of parameters. As we'll discuss in **??**, the Particle Filter method is able to circumvent this re-calculation to some degree.

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 esimate of a posterior Gaussian. In the BOLD case, the set of equations we are modeling have no closed form solution, and finding the Jacobian is impossible without approximations. Although [6], [26] mention a Jacobian of the BOLD response, this is not strictly the case and is rather $\frac{\partial J}{\partial t}$ rather than a true Jacobian. This is important because the Extended Kalman filter depends on the Jacobian to map a Gaussian through the advancement of time. Thus the popular Extended Kalman Filter won't work in this case, whereas the Unscented Kalman Filter still does. In fact [26] uses the UKF to perform a similar type of analysis to the one performed in this work.

The difficulty of using a Kalman Filter, however, is that it assumes a multivariate Gaussian for the state variables, $X(t-1)$. The problem with this is that when a system is significantly nonlinear, the state at $X(t)$ will almost certainly be non-Gaussian, and thus estimating $X(t+1)$ with $X(t)$ as a multivariate Gaussian will perpetually introduce error in the distribution. Furthermore, it is not really known what sort of underlying distributions may exist in such a mixed biological, mechanical, chemical system such as the brain. Assuming the parameters all to be Gaussian may in fact be a gross error. On the other hand, for small variances and short time steps the gaussiann distribution is a good fit, and so in some limited cases the Unscented Kalman Filter could work well. These are non-trivial issues given that the assumption of Gaussianity is what allows the UKF to estimate the posterior using only the first and second moments; two parameters that don't uniquely describe most distributions.

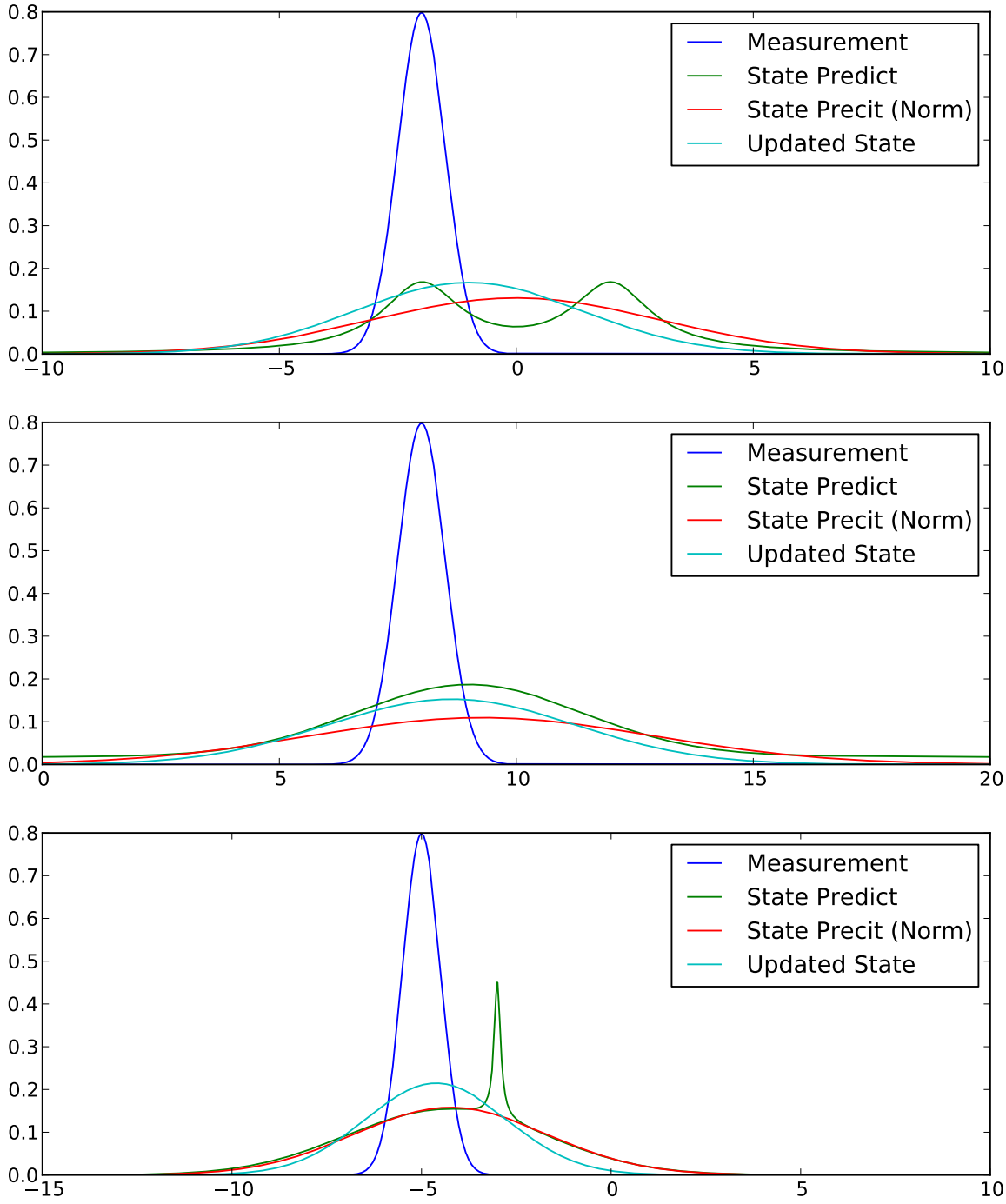


Figure 2.2: Example updates of a distribution using Kalman Filter, [34]

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

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

Notably s_t has intentionally been set to a non-equilibrium, but physiologically plausible value. The value of u is left at zero the entire time, so the system will decay naturally (see section 1.3), though initializing s at a non-zero level will drive the system for several seconds. Figure 2.3 shows the results when the system is essentially left on for 100 milliseconds after setting the variables according to Table 2.1. The Q-Q plots fit very will with a Gaussian, demonstrating that at this short time interval nonlinearities have not yet begun to effect the distribution. However, Figure 2.4 us the result after 1 second, which is faster than most FMRI scanners are capable of sampling at. At that range the tails of the distributions for v and q are clearly starting to deviate from the Gaussian distribution. As a result the uncertainty in y is deviating from the Gaussian distribution as well. This is important, because although approximating the distribution with a Gaussian based on the first two moments will work in the short run, there will be residual error in the distribution.

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

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

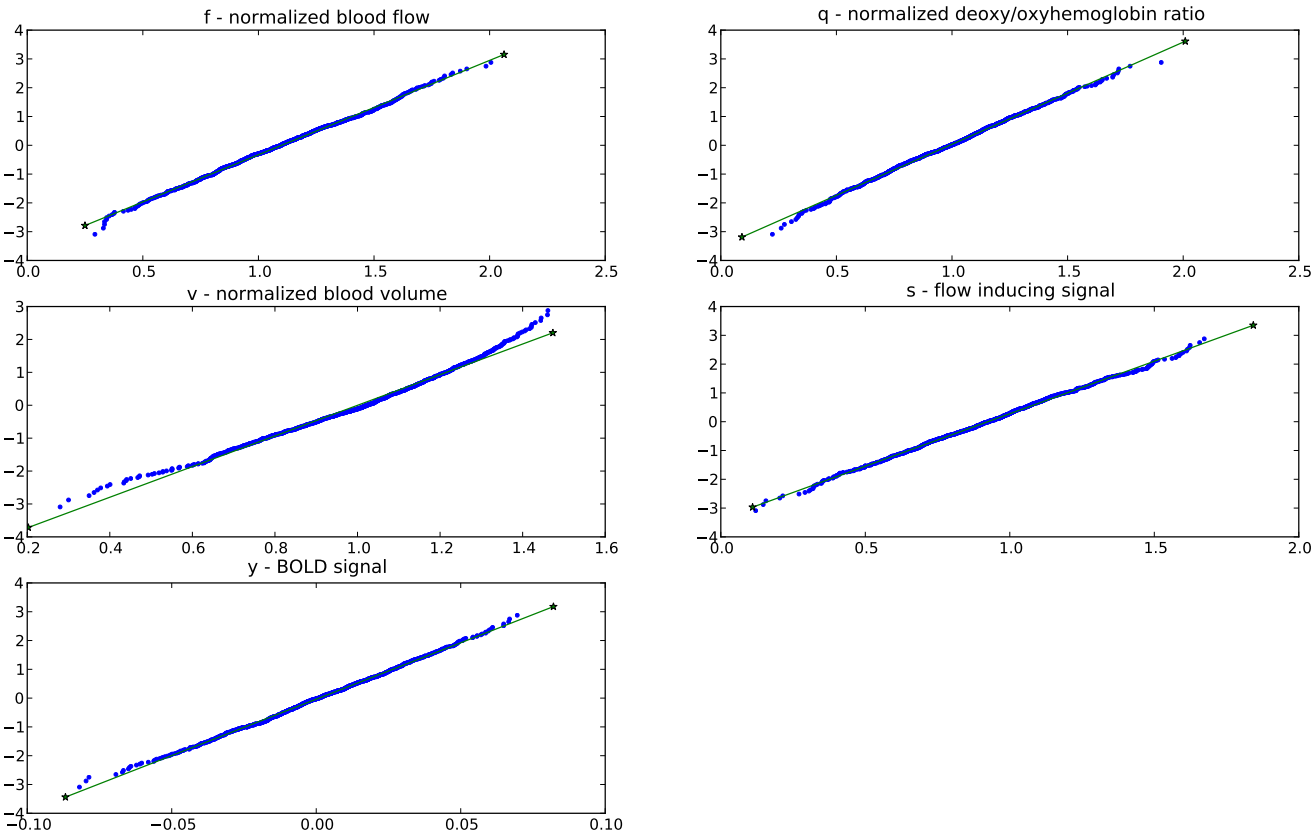


Figure 2.3: Distributions of state variables after simulating for .1s

variables. While this problem is relatively easy to fix (by using more sigma points off the main axes), there is a definite cost in complexity.

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

2.2.4 Hybrid Methods

In [6], a maximum likelihood method for innovation processes was used, as described by [35]. [35] uses a similar construction to a Kalman filter, to break the time series into a series of innovations, for which Maximum Likelihood was performed. While this is probably the most likely solution to give the correct output, there are

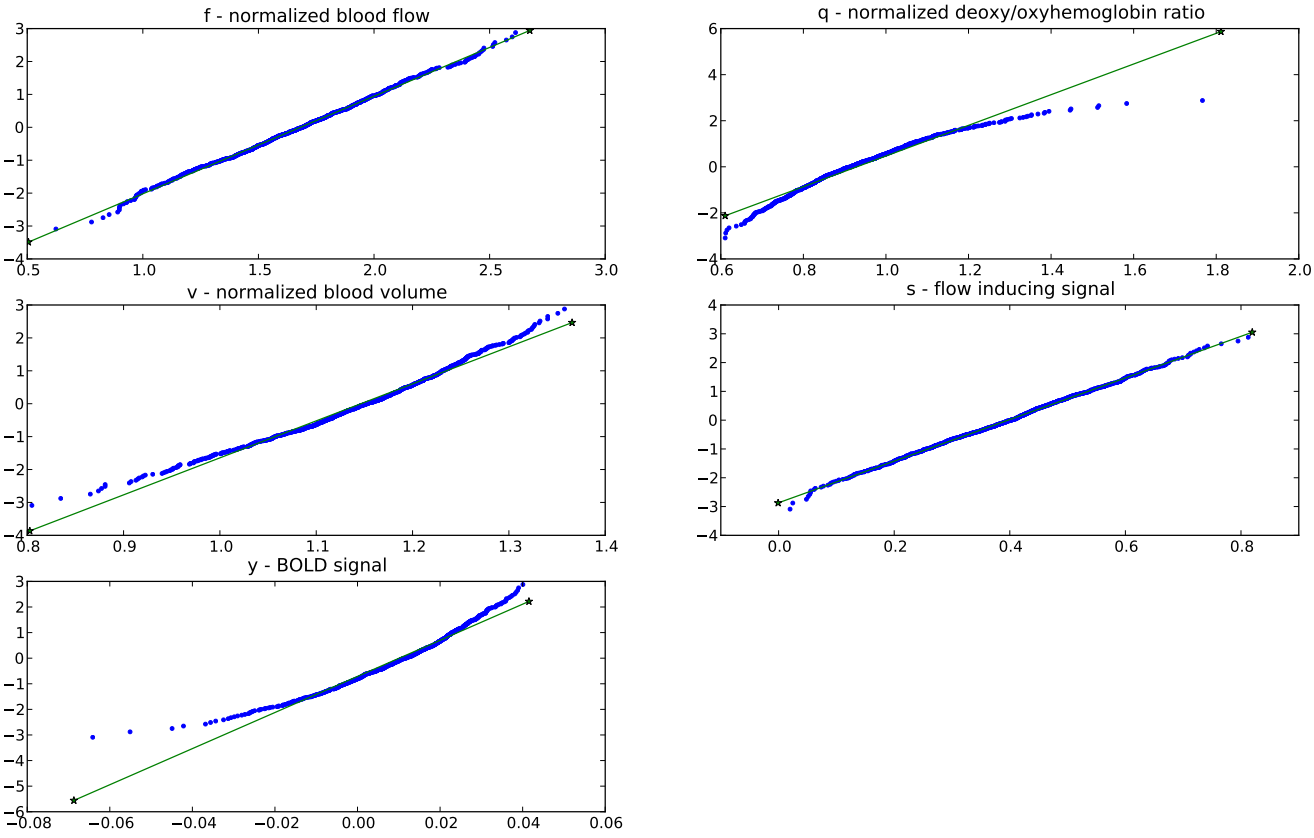


Figure 2.4: Distributions of state variables after simulating for 1s

a few problems. First, every step in parameter space requires a recalculation of all the state variables. With two or three parameters this is fine, more than that, and calculations can become intractable. Additionally, there is no way to ensure a global minimum is reached. Although this is also true of the methods in subsection 2.2.2, those methods contained random elements designed to overcome this issue. [results?]

In [29], 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). Essentially 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 [29] comes to a very different set of parameter estimates as compared to the original [10] estimates (Table 1.1). In fact the results are significantly different from virtually every other study. The most obvious discrepancy is the significantly longer 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 **??**, the algorithm described in [29] does not depend on prior distributions. It is possible then that the bias toward the prior in other methods screwed the results. While [29] is certainly in the minority; further exhaustive studies of the parameters, using unbiased techniques may be called for.

[graph of friston parameters vs. johnston]

[might want to mention dynamic causal modeling]

2.3 Conclusion

Clearly there is no ideal solution to solving this system of nonlinear equations. Exhaustive search type methods such as those employed by [29] and [30] can have extremely 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 very 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 extremely powerful. The particle filter method proposed in the next section bears a strong resemblance to the unscented Kalman filter; albeit with more point estimating the posterior.

Chapter 3

Particle Filters

3.1 Introduction

Particle filters, a type of Sequential Monte Carlo (SMC) methods, are a powerful way of estimating the posterior probability distribution a set of parameters given a timeseries of measurements. Unlike Markov Chain Monte Carlo (MCMC) estimation, particle filters are designed for time-varying random variables. The idea behind particle filters is extremely 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 thus non-gaussian.

3.2 Model

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

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

will be with $X_k = X(t_k)$. The goal of the particle filter is to estimate the *distribution* of the true parameters Θ that dictates the movement of $X(t)$. The model also permits random motion in $X(t)$, so the particle filter also estimates the distribution of $X(t)$. The only difference between the members of parameter vector Θ and those of $X(t)$ is that the 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.

Although not necessary for particle filters in general, a few simplifying assumptions are made for this work. First, the systems are assumed to be time invariant. This assumption is based on the idea that if you paused the system for Δt seconds, when unfrozen the system would continue as if nothing happend. 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. Also, noise is considered to be additive. Finally, because the only difference between the members of $X(t)$ and Θ is an update function, from now on x will contain Θ. The assumptions now allow for a simplified version of the state space equations:

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

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

3.3 Derivation

The goal of the particle filter is to evolve an empirical distribution $P(x_k|u_{0:k}, Y_{0:k})$, that asymptotically approaches the true probability distribution $P(X_k|u_{0:k})$. Note that capital X will be used as the actual realizations of the state variable, whereas x will denote estimates of X. Additionally, the notation $a : b$ indicates the set

$[a, b]$, as in $u_{a:b}$, which would indicate all the inputs from time a to time b. Considering the noise present in X, $P(X_k|u_{0:k})$ is not a single true value but probability distribution.

To begin with, the particle filter must be given a prior distribution, from which the initial N_p particles are drawn. A particle contains a weight as well as an estimate of X_k, which as already stated, contains every variable needed to run the model. Then the prior is generated from a given distribution, $\alpha(X)$, by:

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

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. Of course if there is a far off peak in the posterior that q does not have support points in, there will be quantization errors, and that part of the density can't be modeled. This is why it is absolutely necessary that q fully covers $P(x^i_{0:k}|y_{0:k})$.

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.

3.4 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 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 modeled 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 extremely 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}, we are able to make the very 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 (mostly 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 relatively easy assumptions, evolving a posterior density requires no knowledge of noise distribution.

1. $f(t, x(t), u(t)) = f(x(t), u(t))$ and $g(t, x(t), u(t)) = g(x(t))$

2. The PDF $q(x_i(0))$ (the prior) fully covers $P(x_i(0))$

3. Markov Property: $P(x_k|x_{0:k-1}) = Pr(x_k|x_{k-1})$

4. $q(x_{0:k-1}|y_{0:k}) = q(x_{0:k-1}|y_{0:k-1})$

3.4.1 Basic Particle Filter Algorithm

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

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

3.5 Resampling

As a consequence of the wide prior distribution (required for a proper discretization of a continuous distribution), there will be many particles with insignificant weights. While this does help describe the tails of the distribution, it means a lot of computation will be based. Instead, it would be preferable if most of the computation is spent on the most probable regions. Ideally the computation time spent on tails would be proportional to the actual size of the tails. In this case particle locations would match the true posterior and all weights would be equal. The case where a large number of the weights have become irrelevantly small is called "particle degeneracy". In [36] 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:

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

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

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

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

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. [37] goes into significant depth and proves the optimality of the Epanechnikov Kernel for reducing MSE between the original and resampled distributions. However, [37] also espouses the usefulness of the Gaussian Kernel, due to the ease drawing samples from it, which for this work was more important.

It has been proposed by [38] 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 re-sampling is performed over and over. Thus if resampling is performed at every step then this could certainly caused problems. If

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

the distribution is over-smoothed then the algorithm may not converge as rapidly; however, because the bandwidth is still based on particle variance, which decays as particles are ruled out, it is still able to converge. In fact, over-smoothing is preferable to under smoothing, since over-smoothing simply slows convergence while under-smoothing could result in false negatives. At the same time, as the number of dimensions in the state vector increases, the standard deviation based approximation becomes less effective ([39]). 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 is the opposite of convergence, which is why regularized resampling should only be done when N_{eff} drops (say less than 50). Other than the periodic regularized resampling then, the regularized particle filter is nearly identical to the basic sampling importance sampling filter (SIS).

Thus, with regularized resampling, it is possible to prevent both particle degeneracy as well as particle impoverishment. An additional bonus is that as the particle filter converges, the density of particles in the area of the solution goes up. This has a similar effect to simulated annealing where, as the algorithm approaches the

end, the random steps get smaller and smaller. There is a potential risk when resampling. If for some reason the solution is not covered by the new support the algorithm will fail to converge properly.

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

3.6 Weighting Function

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

3.7 Simple, Nonlinear Example

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

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

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

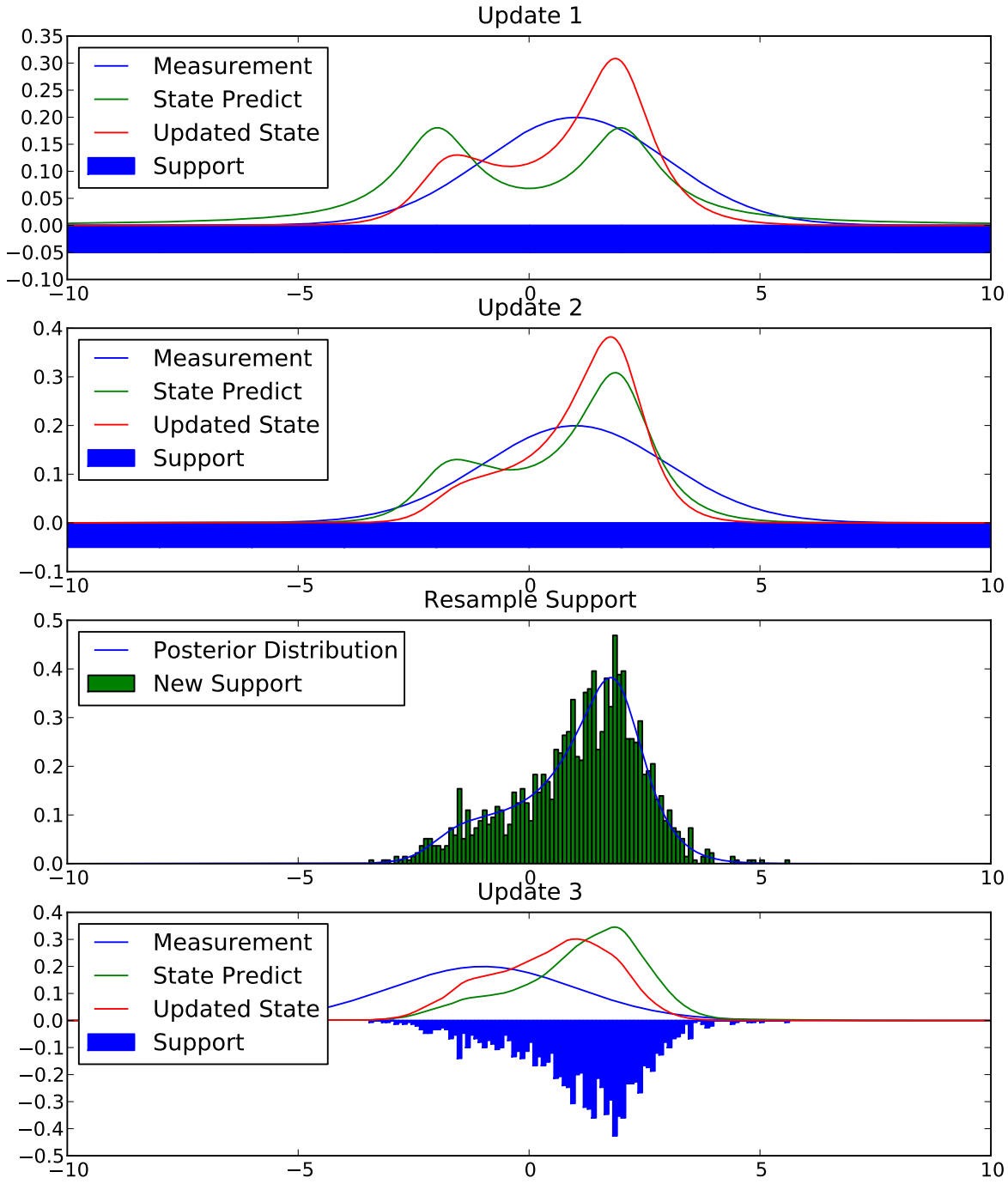


Figure 3.1: Particle Filter progression, note that the initial support is flat; the particles are equally spaced between -10 and 10

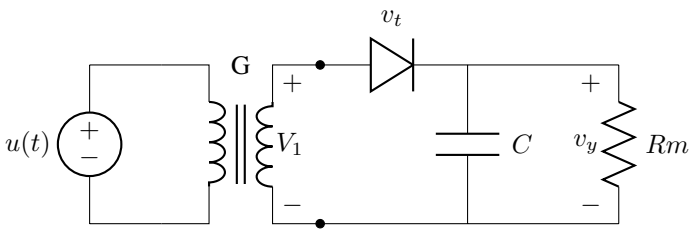


Figure 3.2: An Example Half Wave Rectifier Circuit, where G is the transformer gain, v_t is the activation voltage of the diode, $u(t)$ is the input at time t, C is the capacitance, R is the load resistance and v_y is the output voltage

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

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

To run the particle filter is relatively easy then, 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:

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

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 resulting in a distribution that asymptotically approaches the true joint distribution of the $X(t)$. Of course, as I mentioned in section 3.5, particles weighted zero do not significantly contribute to the empirical distribution, so re-sampling may be necessary.

Chapter 4

Methods

Although the particle filter I used for this work is a standard Regularized Particle filter, as described in [39], optimizing the particle filter for use with FMRI data is non-trivial.

4.1 Prior Distribution

For the BOLD model described in section 1.3, 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, [10] only studied small regions; and most research with reasonable speed (including [33]) used these results as the source for their priors. The one exception is [29], which came to an extremely different distributions. For a particle filter, the choice of a prior is the single most important design choice. Consequently for this work, I wanted to be conservative; which meant going with the accepted result, [10]. This constrains the usefulness of the model to areas that fall within the prior distribution, yet will allow results to be comparable to other works. There is a significant need for better estimates of the physiological parameters; and, while physical experiments may not be possible, it would not be unreasonable to do a study with exhaustive simulated annealing or hill climbing tests for multiple regions and multiple patients. The purpose for this work is to determine the fitness of particle filters for this task; because with good priors it can be extremely fast. Ultimately this algorithm may become more useful as more studies are done on BOLD model that don't require priors.

There is an interesting anomaly with the priors found in virtually all the works that characterized the parameters, except [29]. The BOLD signal is universally recognized to be around $2-3\%$, maybe reaching 5% in extreme activation. Yet using the mean priors from [10], the signal response for a .1 second impulse only reaches maybe half a percent, as Figure 4.1 shows.

While this could be the result of a stimulus being too short to lead to strong activation, a similar stimulus

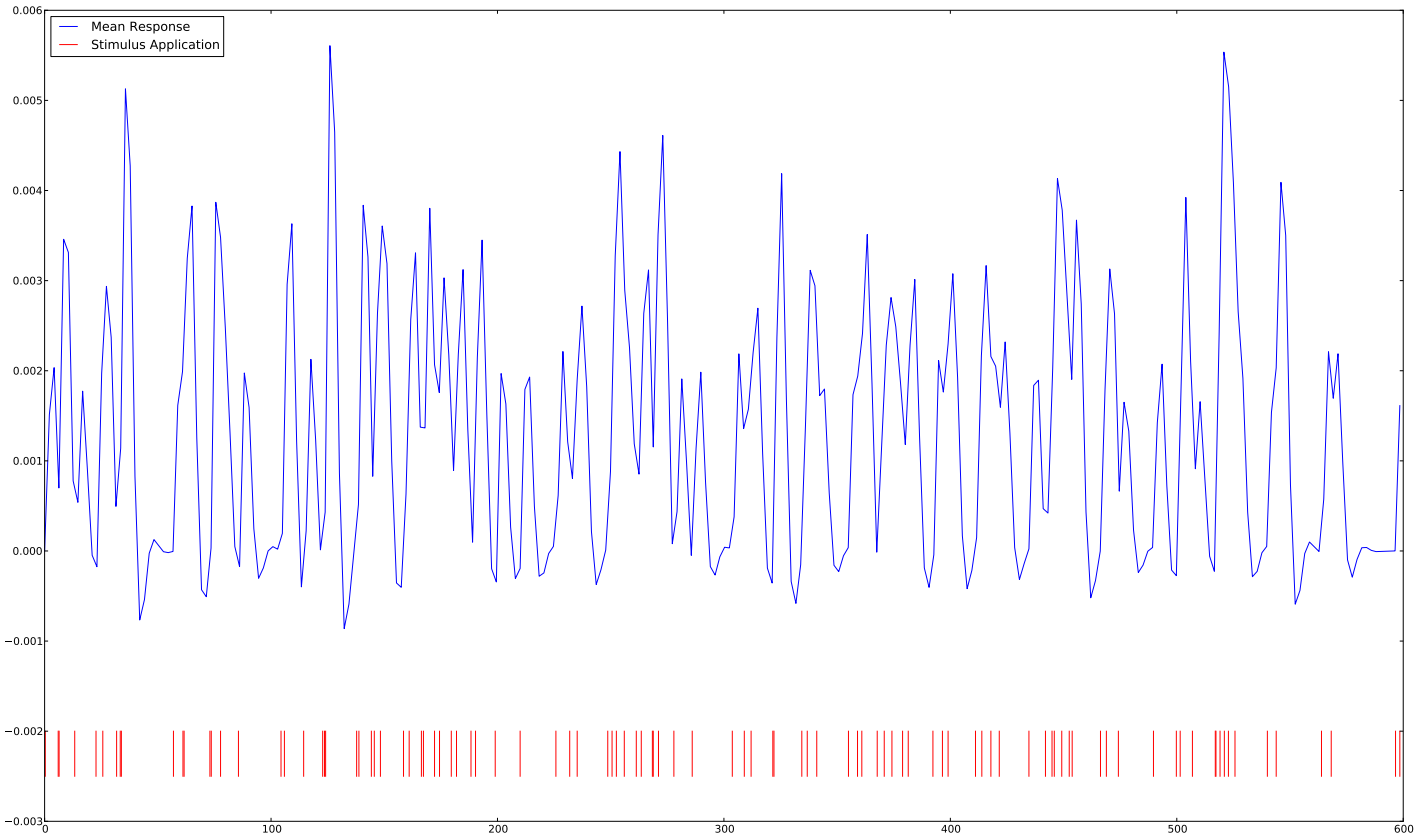


Figure 4.1: Response to $.1s$ impulses with the mean parameters from [10]

scheme in real data showed a significantly 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 [29] are even more confusing, with peaks of well below $.1\%$ (**??**).

Its likely that these differences are due to some difference in preprocessing, although in [4] the signals were found to be peaking around 1%, unlike [10] 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 [10] (which are widely used, [26]). The priors used in the particle filter may be found in Table 4.1.

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

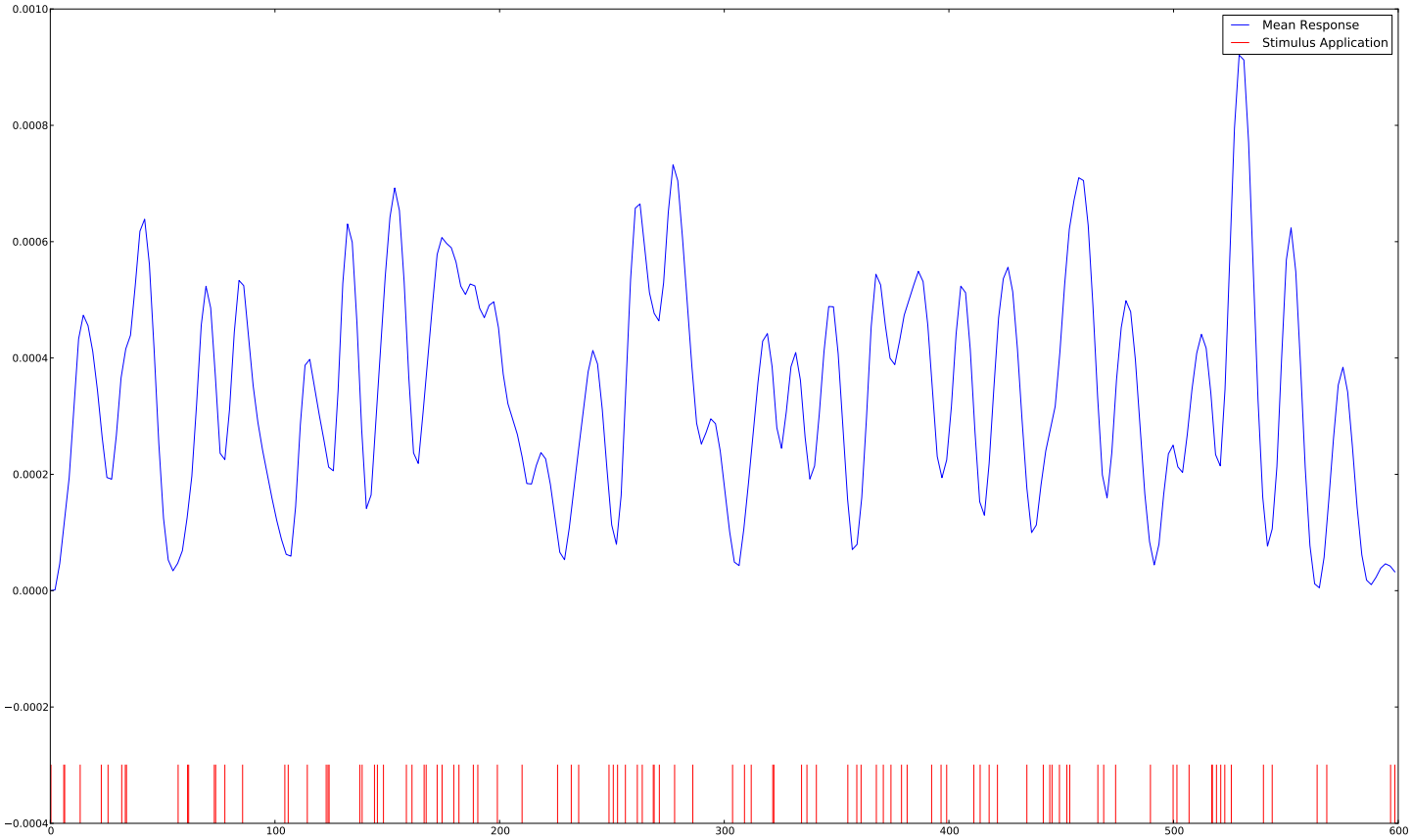


Figure 4.2: Response to $.1s$ impulses with the mean parameters from [29]

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:

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 extremely difficult. Insufficiently dense particles will result in inconsistent results, of course the processing time will scale up directly with the number of particles. A dense initial estimate is important so that some particles land near the solution; but as the variance decreases the number of particles needed decreases as well. Thus, as a heuristic, initially the number of particles is set to 28,000, but after resampling, the number of particles is dropped to 1,000. Typically during the first few measurements the variance drops precipitously since most particles don't describe the system well. The particles that are left are in a much more compact location, allowing them to be estimated with significantly fewer particles. These

numbers aren't set in stone, and depending on the complexity of the system or desired accuracy they could be changed; however, they seem to be the minimum that will give consistent results.

4.2 Model

As originally written in section 1.3 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 model that noise. It is common, however, to actually model that noise in particle filters by adding a random value to each updated state variable. Because the purpose of this particle filter is to learn the underlying distribution of the static parameters, rather than precisely model the time course of the in the dynamic parameters ($\{s, f, v, q\}$) this noise is left out. Because the BOLD model is dissipative, when no stimuli are applied, all the particles will decay to ($\{0, 1, 1, 1\}$) which they should. If evolution noise were added, the particles would then weight particles based on the results of that noise, rather than on the quality of the static parameters. Typical particle filter uses also use this state noise as an exploratory measure; however given the high dimensionality of the system, a vast number of particles would be necessary for this to come to fruition. The noise in the BOLD signal is such that several particles all tend to be equally correct; the particle filter will still leave variance in the solution; which turned out to be the case.

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

4.3 Resampling

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

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

When some sort of particle deprivation has not occurred, regularized resampling with a large bandwidth will slow down convergence. Thus as mentioned previously, to prevent over smoothing, the particle filter algorithm used here waits until the effective sample size drops below 25 for two measurement points in a row. This seemed to give good results, and avoids temporary drops in the N_{eff} caused by outliers.

4.4 Choosing $P(y_k|x_k)$

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

As noted in section 1.5, the noise is not strictly Gaussian, nor is it strictly Wiener. As with any unknown

noise however, it is necessary to make some assumption. If the weighting function ($P(y_k|x_k)$) exactly matches the measurement error, then the ideal particle filter will result. Particles with x_k's that estimate y_k far out on the weighting function will quickly have weights near 0. Thus, an weighting function that exactly matches $P(Y(t)|X(t))$ will easily, and correctly throw out incorrect particles. However assuming that this function is not going to be found for arbitrary voxels, it is necessary to choose a function to approximate this probability. The cost of choosing an overly broad distribution for this function is an overly broad $P(x_k|y_k)$; and ultimately an overly broad $P(x_k|y_{0:k})$. On the other hand, an overly thin distribution will lead to, at best, an overly thin representation of these same distributions; but at wost will lead to particle deprivation (all particles being zero-weighted). If the *true* distribution were found, particle deprivation would indicate that that time series is not being affected in any way describable by the prior distribution/model. Particle deprivation due to an under-variant weighting function gives no information. Thus, the cost of under variance is significantly higher; a fact that definitely needs to be taken into account.

Because of the need to err on the side of caution I tested several weighting functions. In addition to the Gaussian I also tested the Laplace and Cauchy distributions, both of which have much wider tales than the Gaussian. The benefit of the wider tailed distributions is that they don't down-weight particles quite as fast. Additionally, the Laplace distribution has the benefit of having a non-zero slope at the origin. This means that even if the distribution is made overly broad, it will still distinguish between particles that are near the origin. Results with real data compare the three distribution in section 5.6.

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

4.4.1 Classical De-trending

The non-stationary aspect of a Weiner process, presumably the result of integrating some ν_x is difficult to compensate for, and so various methods have been developed to compensate for it. [22] and [24] have demonstrated that this component is prevalent, and may in fact be an inherent characteristic of FMRI. In some studies, as many as half the voxels benefit from detrending, presenting a serious barrier to inference ([23]). All the existing methods are performed during the preprocessing stage, rather than as an integral part of analyzing the BOLD

signal. There is no shortage of theories on the "best" method of detrending; however in a head to head comparison, [22], showed that in most cases subtracting off a spline works the best. The benefit of the spline versus wavelets, high pass filtering or other DC removal techniques is that the frequency response is not set. Rather, the spline is adaptive to the input, having a low cut off if the signal's median stays constant but a high cut off frequency if the signal's median tends to shift heavily over the course of time. In spite of this, the spline will still remove some amount of signal, just like all of these methods, which is why I considered the method proposed in subsection 4.4.2.

The method I used to calculate the spline was picking one knot for every 15 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. It seems that there is potential to optimize the spline further using a canonical HRF to find resting points; however, for this to work the experiment would have to be designed with this in mind.

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

Thus, I used a more conventional approach to deal with this. To determine the DC gain to be added to the signal I used a robust estimator of scale. The Median Absolute Deviation (MAD) proved to be extremely accurate in determining how much to shift the signal up by. I tested both methods during the course of analysis, and found that the increase in model complexity far outweighed the slight increase in flexibility. Its possible that a more accurate method may exist; however, for this case the MAD seems to work as it should, as Figure 4.3 shows.

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

A serious concern with adding and subtracting arbitrary values to real data is whether this will create false

Figure 4.3: image of de-spline'd lines with "true" lines

Figure 4.4: frequency response graphs, highlighting noise frequency range and signal frequency range

positives. This is a legitimate concern; however, all the additions and subtractions done in this section have been very low frequency changes and should not substantially change the BOLD response, being a relatively high frequency signal. The typical method of preprocessing is a high-pass filter with a cutoff of around 30 seconds; which should about match the spline with knots every fifteen measurements. [22] found that splines tended to far outperform the high-pass filter method.

4.4.2 Delta Based Inference

The alternative to these sorts of low frequency manipulation is to go around the problem in another way. Here, I propose and test a different method of dealing with the so called "drift". Instead of comparing the direct output of the particle filter with the direct measurement, the algorithm would compare the change in signal over a single TR, with the result of integrating the model for the same period. In most signal processing cases this would be foolish, but that is because the general assumption is that all noise is high frequency. Considering the fact that every BOLD analysis pipeline uses a high pass filter, whereas low poss temporal filter are rarely applied, it makes sense that a derivative type method could work. The benefit of particle filters is that they are extremely robust method of inference, and I would assert that the particle filter ought to be given as *raw* data as possible. While taking direct measurements without de-trending would give awful results, using the difference removes the DC component and turns what is usually assumed to be a Weiner process into a simple Gaussian random variable.

$$\Delta y = y(t) - y(t-1) = g(x(t)) - g(x(t-1)) + \nu_y(t) - \nu_y(t-1) + \nu_d(t) - \nu(t-1) \tag{4.6}$$

Even if ν_d is some other additive process, the difference will still be closer to I.I.D. than a Wiener process, as the autocorrelation of the δy shows in **??** in section 1.5. All the assumptions made originally for the particle filter still hold, and all of the parameters may be distinguished based on the step sizes, thus it is not unreasonable to consider matching the string of step sizes rather than string of direct readings.

4.5 Preprocessing

As discussed in the section on de-trending, the normal pipeline for analyzing FMRI involves a great deal of preprocessing. 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 takes care of motion by the patient as well as some small changes in the magnetic fields that cause the image to shift. After this, a structural image of the patient is often registered to an atlas so that a grey-matter mask may be created. Since all the ”activation” should occur in grey matter voxels, it made sense to restrict analysis to only these sections; although there may be partially grey-matter areas that are missed. This significantly eases computation time though.

In normal statistical parametric mapping, a gaussian smoothing filter is applied across the image as discussed in subsection 2.1.2. After this, detrending is performed discussed by subsection 4.4.1. Recall that FMRI signal levels are unit-less and even though detrending isn't always necessary, at the very least the data must be converted into % difference from the baseline. Changing to % difference removes no real information from signal. This is the signal that was input into the delta based particle filter. Of course, most of the time analysis is performed on the direct signal; which mandates the removal of low frequency drift. The generally accepted method is to use a high pass filter, although the cutoff frequency is application dependent and often applied haphazardly.

Chapter 5

Results

5.1 Simulation

I performed two types of simulations. First, I simulated a single BOLD time-series based on a randomly selected set of model parameters. Second I used a modified version of the FSL tool possum to generate a noisy FMRI image.

5.1.1 Single Timeseries

This process was relatively straight forward, given the state-space equations for the BOLD signal. After a "true" signal was generated, I then added a identically and independently distributed (I.I.D.) Gaussian noise and a Wiener process with Gaussian I.I.D. steps. Finally a carrier level was added, since BOLD is typically measured as a % difference from the base level. Adding a carrier level meant that the exact same algorithm which was used for real data could be used for the simulated data.

Once this noisy simulated time series was generated, the particle filter algorithm was run on this single timeseries image. In order to find the best setting for the particle filter I ran quite a few tests. Here I will include two sets of tests to determine the power of the particle filter in modeling. The first test is low noise $\{\sigma_y = .001, \sigma_x = .0005\}$, while the second set of tests had noise of $\{\sigma_y = .01, \sigma_x = .005\}$, where σ_y is the measurement noise, and σ_x is the wiener step size. Both signals used the parameters ($\tau_0 = 1.45, \alpha = .3, E_0 = .47, V_0 = .044, \tau_s = 1.94, \tau_f = 1.99, \epsilon = 1.8$). The particle filter used the parameters defined in Table 4.1 (**??**), thus the particle filter was not centered over the correct values.

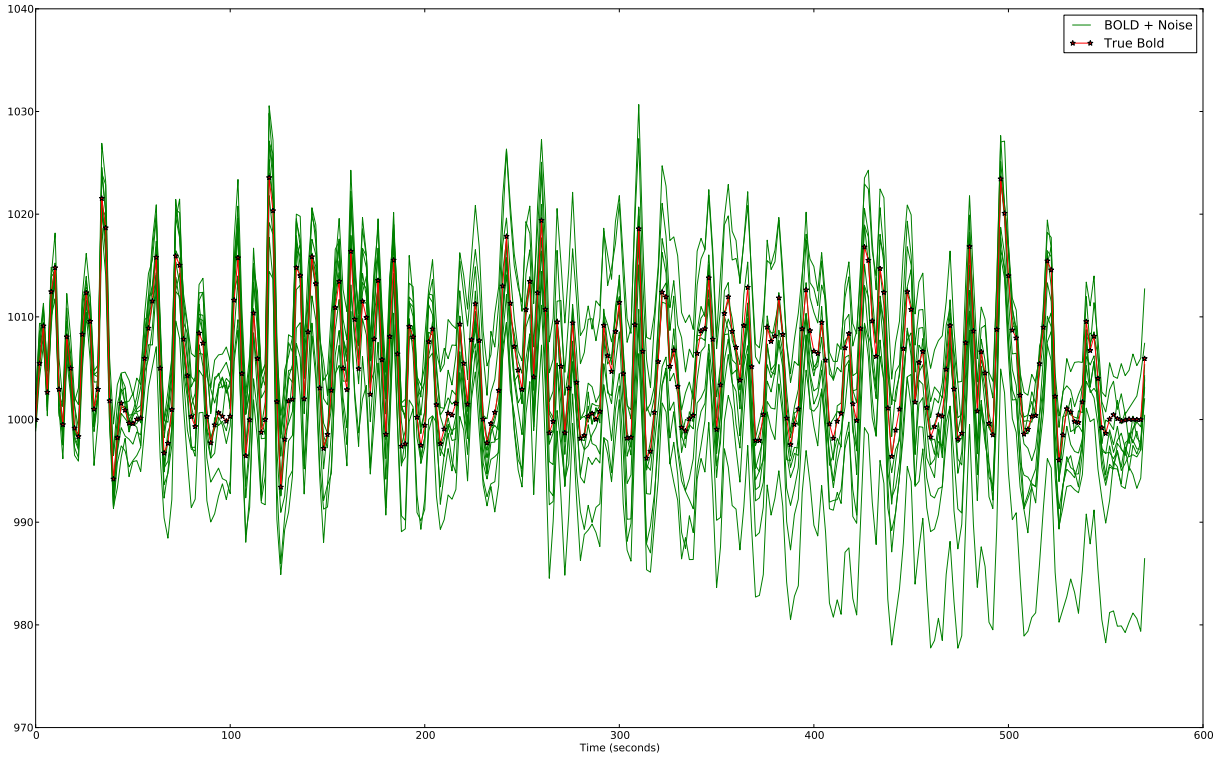


Figure 5.1: Test Signals with low noise compared to the clean signal

Low Noise Simulation

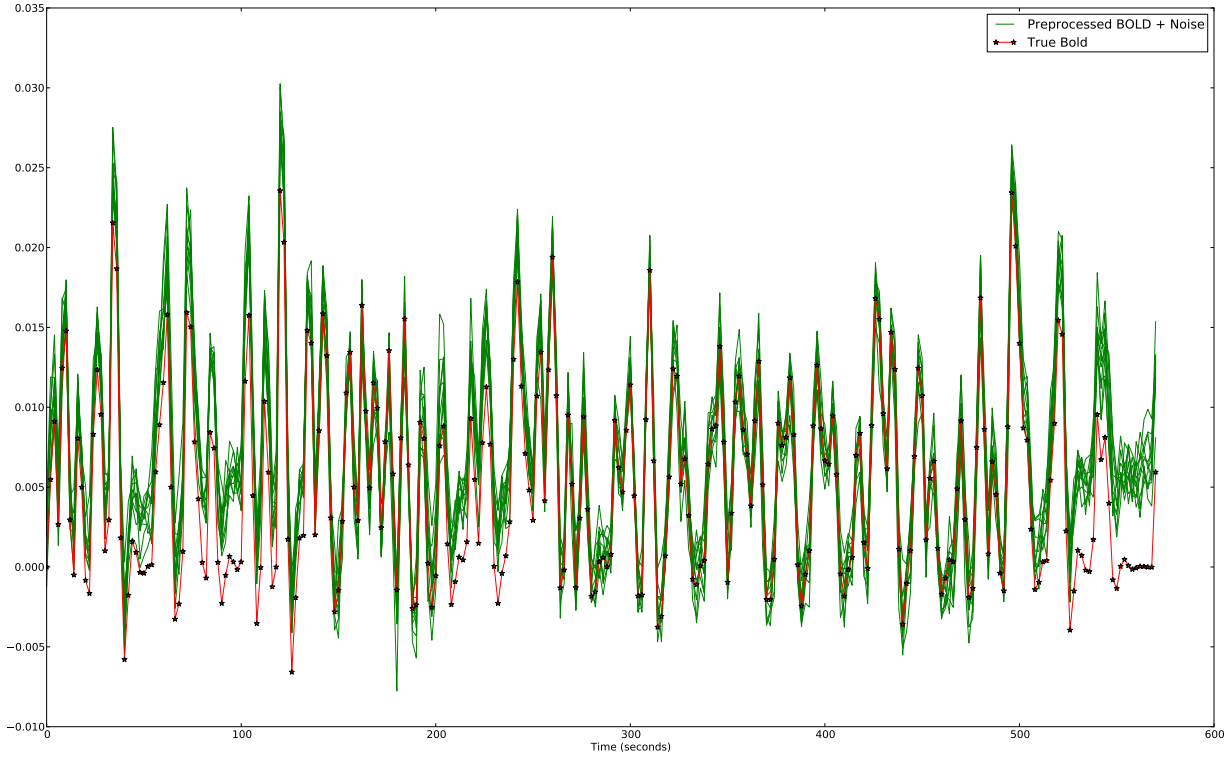
For the low noise case, the ten realization are shown in section 5.1.1.

The resulting fits compared to the *true* BOLD signal are shown in Figure 5.1.1. The fits are good, but the noise is relatively low. The preprocessed signal compared to the true BOLD signal is shown in Figure 5.1.1.

The preprocessing consists of several steps, which are discussed in detail in **??**. Obviously the spline struggles a little bit at the end, as the graph shows, although overall the fit is pretty well. Finally, the set of fits to the preprocessed data, and in essence to the "true" BOLD signal is shown in Figure 5.1.1.

It becomes clear now the reason for the term particle filter. In several locations the output of the particle filter looks like a filtered version of the input. For instance toward the end, the estimates stay flat in spite of the preprocessed data drifting off a bit. By this point, the algorithm had converged sufficiently to know that that wasn't possible. A similar circumstance occurs at 100 seconds in. In general the fits are good, although because of the slightly higher signal levels in the preprocessed signals, the fits tend to overestimate the peak a bit. The final parameter sets are shown in **??**

There are a few things worth noting here. First the time constants vary greatly across runs with different noise realizations, yet the sum of the individual time constants , τ_f, τ_s and τ_0 seem to be more consistent. In



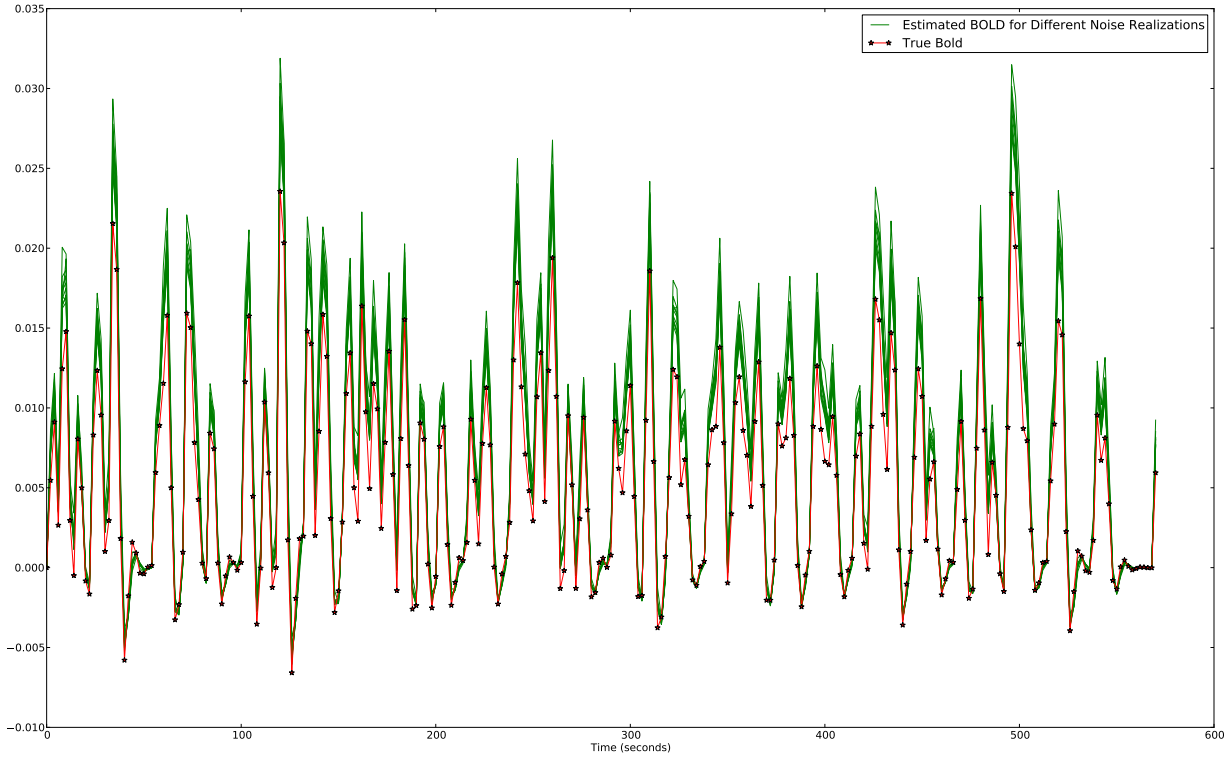


Figure 5.3: A comparison of the fitted signals for the low noise case.

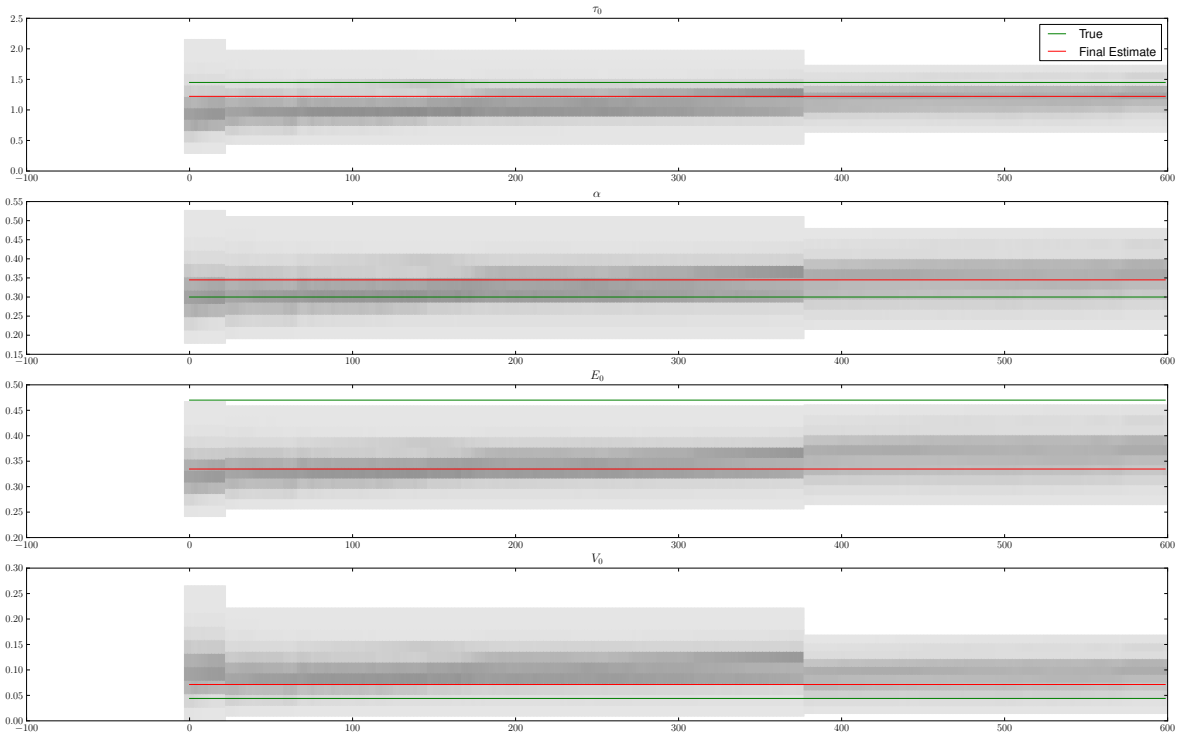
τ_0	α	E_0	V_0	τ_s	τ_f	ϵ	$\sum \tau$	$\sqrt{MSE}$ (Res.)	$\sqrt{MSE}$
1.45	.3	.47	.044	1.94	1.99	1.8	5.38		
1.22214	0.3449	0.33462	0.07138	1.60446	2.27529	1.59454	5.10189	0.00321067	0.00987647
1.37493	0.33183	0.36296	0.07327	1.64076	2.10299	1.57627	5.11867	0.00305475	0.00993227
1.16604	0.32205	0.34057	0.08217	1.64768	2.35351	1.24515	5.16722	0.00328932	0.00967961
1.23176	0.32707	0.34028	0.07961	1.62698	2.18519	1.30326	5.04394	0.00284719	0.00912005
1.1832	0.31787	0.34718	0.0821	1.54961	2.29115	1.27817	5.02396	0.00300634	0.00971258
1.1424	0.33395	0.34725	0.07366	1.62208	2.29084	1.4025	5.05531	0.00283287	0.00948483
1.30041	0.3596	0.35643	0.07679	1.56406	2.13234	1.60338	4.99681	0.00302802	0.01021885
1.24008	0.34601	0.33978	0.08903	1.64989	2.23655	1.29004	5.12651	0.00304378	0.01007964
1.1709	0.32739	0.34644	0.08255	1.53734	2.28262	1.37828	4.99087	0.00334488	0.01032886
1.18967	0.34344	0.33554	0.07976	1.53582	2.30746	1.42774	5.03295	0.00317542	0.01001503
1.184	0.34053	0.35017	0.08917	1.61025	2.27926	1.16448	5.07352	0.00288855	0.00950536
1.21868	0.33588	0.34557	0.07995	1.59899	2.24884	1.38762	5.06651	0.00306562	0.00981396

Table 5.1: Estimated Parameters on 10 different runs with low noise. First row is the true parameters, last is mean over the 10 runs. The $\sqrt{MSE}$ (Res.) is the square root of the mean square of the residuals. Essentially this is the is the MSE between the estimated signal and the noisy signal which was available to the particle filter. Square root of MSE is the actual *error*, with the true signal, which of course was not available to the particle filter.

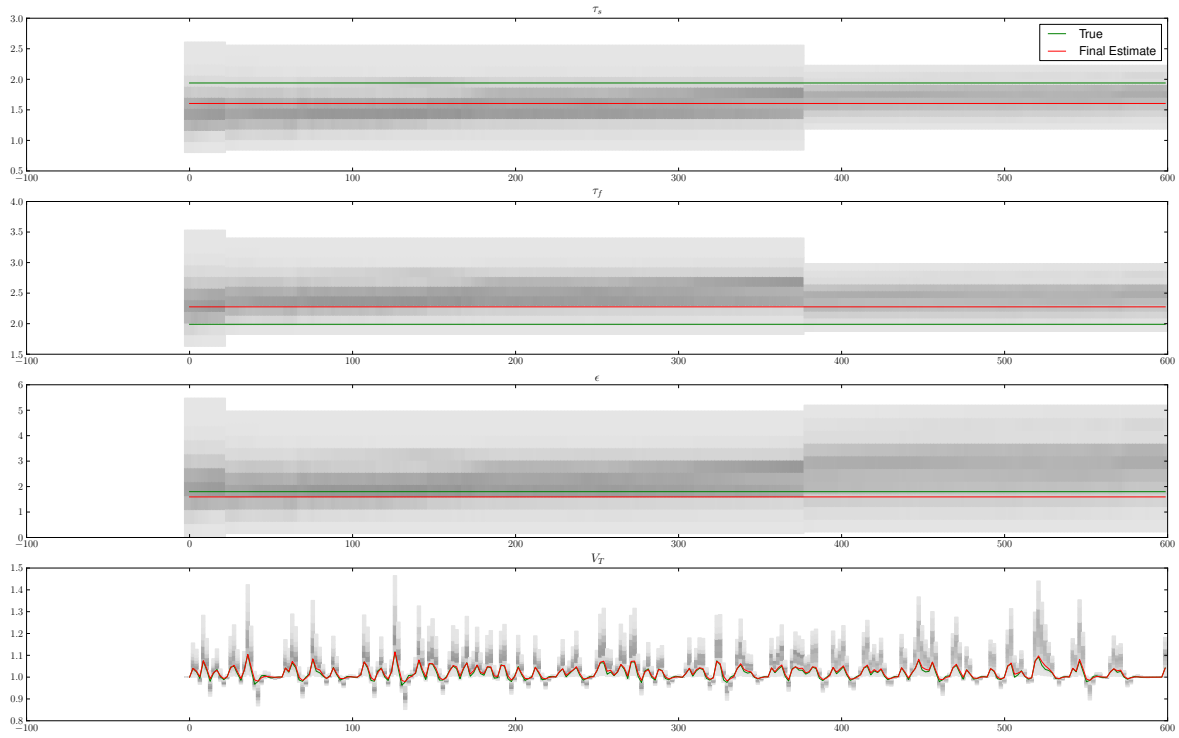
	τ_0	α	E_0	V_0	τ_s	τ_f	ϵ
τ_0	0.0189481	-0.0014269	-0.0011267	-1.13e-05	-0.0025616	-0.0189559	0.0070405
α	-0.0014269	0.0026716	9.93e-05	-0.0002041	-0.0008632	-0.0016823	0.0071891
E_0	-0.0011267	9.93e-05	0.0010701	-0.0002277	-0.0001177	0.0001013	0.0016972
V_0	-1.13e-05	-0.0002041	-0.0002277	0.0005401	4.3e-06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

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

general the time constants are falling short of the true time constant. This could be a limitation based on the prior distribution (which notably has mean values below the values used in the simulation) or it could be caused by the delayed benefit of having a correct time constant. Even if one particle has a better time constant than another, if the difference isn't severe, by the time this makes a difference in the weight, the particle with the better τ will have been weighted based on other parameters several times. Another interesting result in the huge variation in the levels of V_0. In general, with this admittedly small amount of noise, it would appear that the relation between a set of parameters/stimuli with the output is not injective. In other words, it would appear that a timeseries is not unique to a set of parameters. This is good justification that simultaneous blood volume or tagged flow calculations with the conventional FMRI could definitely benefit the model.

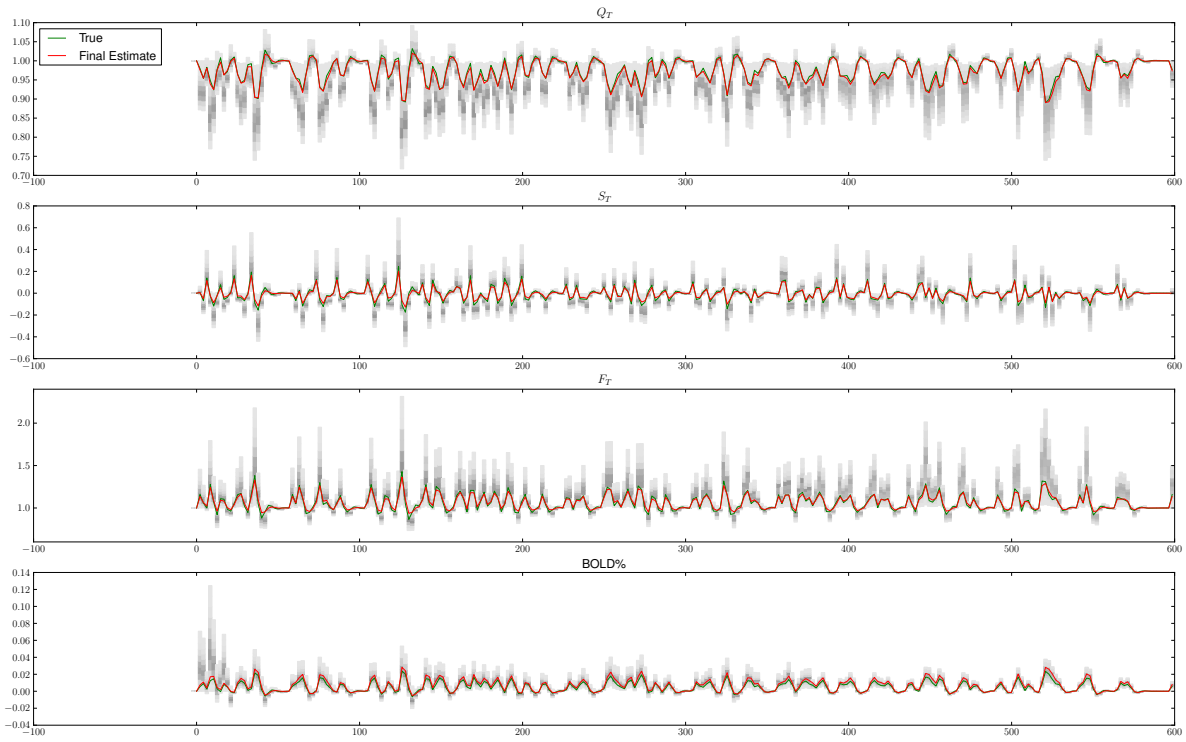


(a) Converging histogram for τ_0, α, E_0, and V_0 of the first run, low noise simulation.



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

The covariance matrix (Table 5.2) also confirms the idea that the τ parameters are interchangeable as far



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

as the BOLD signal goes. Notice the covariance of τ_f and τ_0 is -0.019 whereas the variance of τ_0 and τ_f are 0.019 and 0.04 respectively. Clearly they are related. The convergence properties of the first run in Table 5.1 may be viewed in 5.1.1.

High Noise Simulation

For the high noise simulation, the exact same procedure was performed again, but with measurement error and drift standard deviations an order of magnitude higher. The noisy signals versus the base signal are shown in **??** and the preprocessed signals which the particle filter attempt to fit are in Figure 5.1.1.

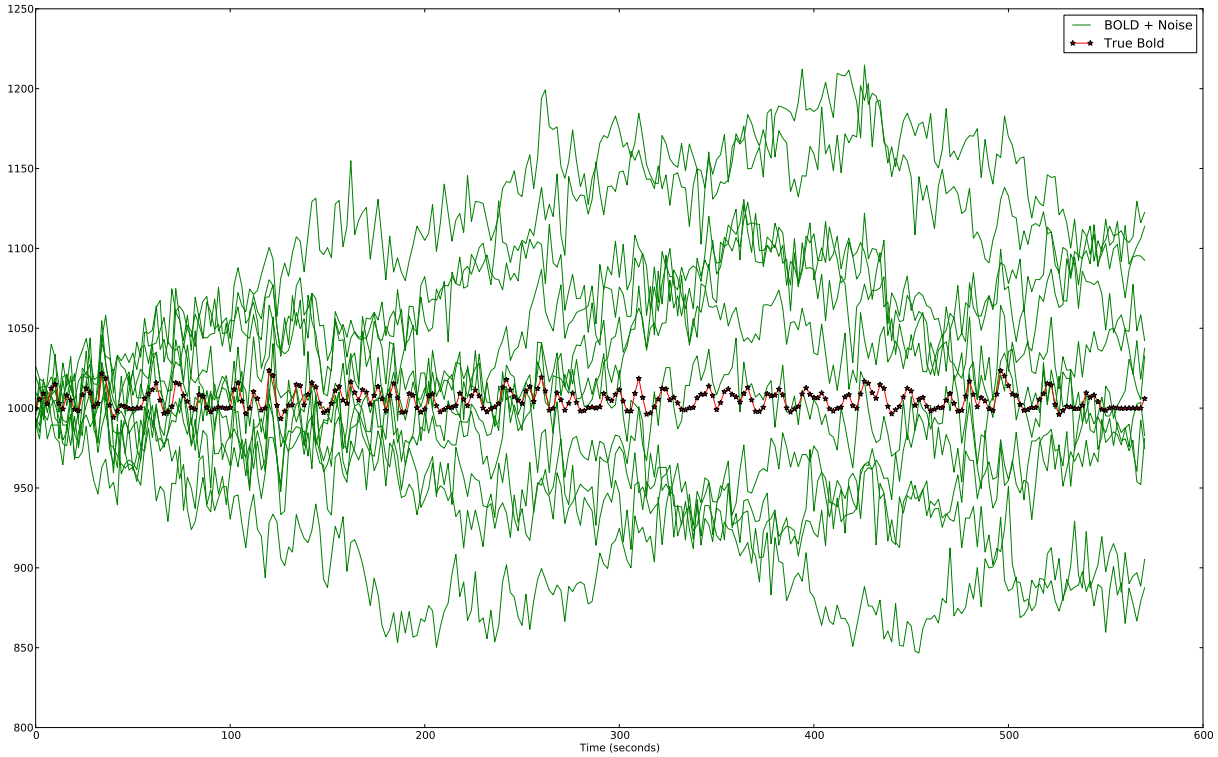


Figure 5.4: Test Signals with high noise compared to the clean signal, $\sigma_x = .01, \sigma_y = .005$

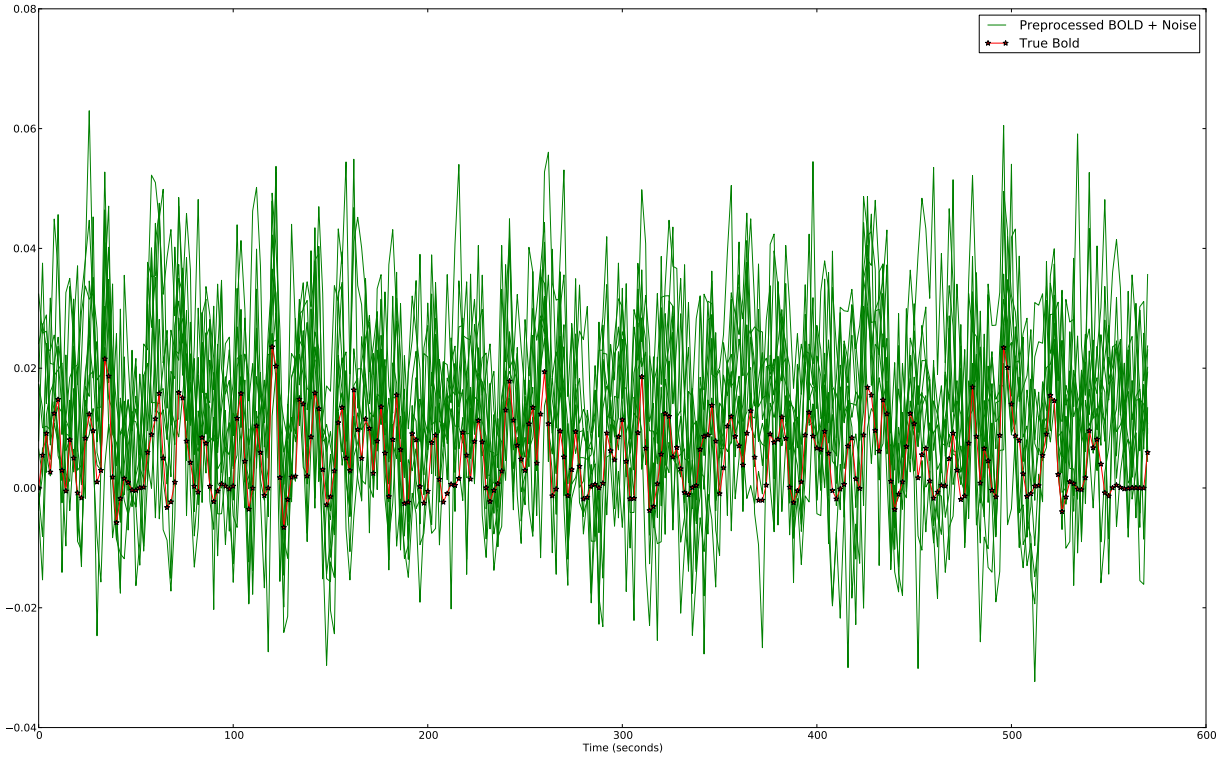


Figure 5.5: A comparison of the preprocessed signals for the high noise case.

The results of the particle filter, for each of the ten runs may be seen in Figure 5.1.1. Clearly the preprocessing led the algorithm to somewhat higher activation levels, and it would appear that the subtleties of different time constants are also lost for most of the runs, although a few seem to manage quite accurate results.

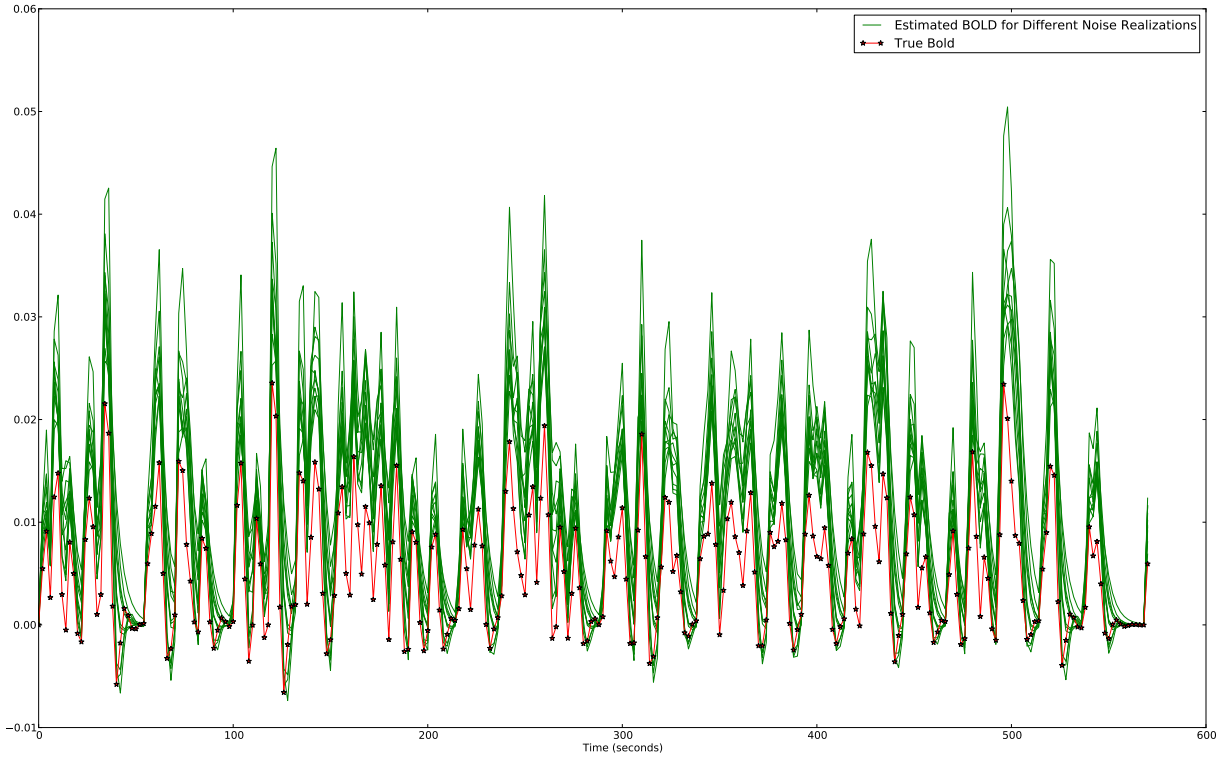


Figure 5.6: A comparison of the fitted signals for the high noise case.

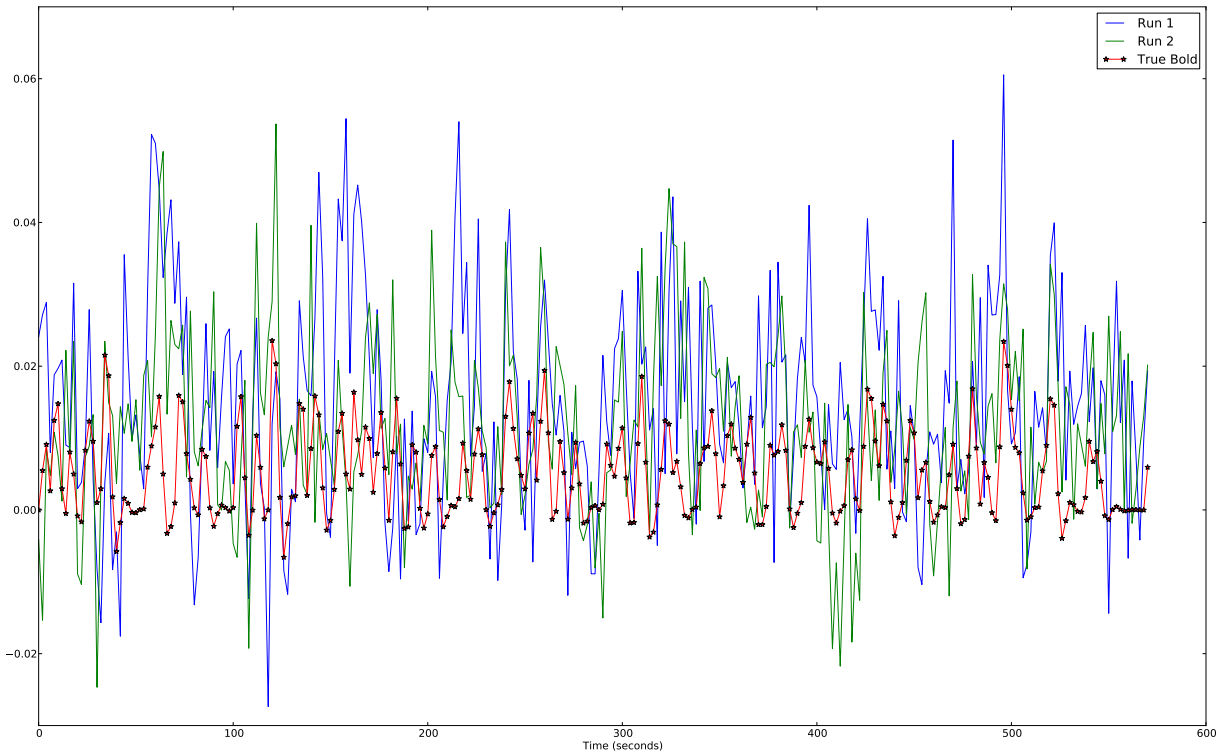


Figure 5.7: Two particular preprocessed noise realizations for the high noise case.

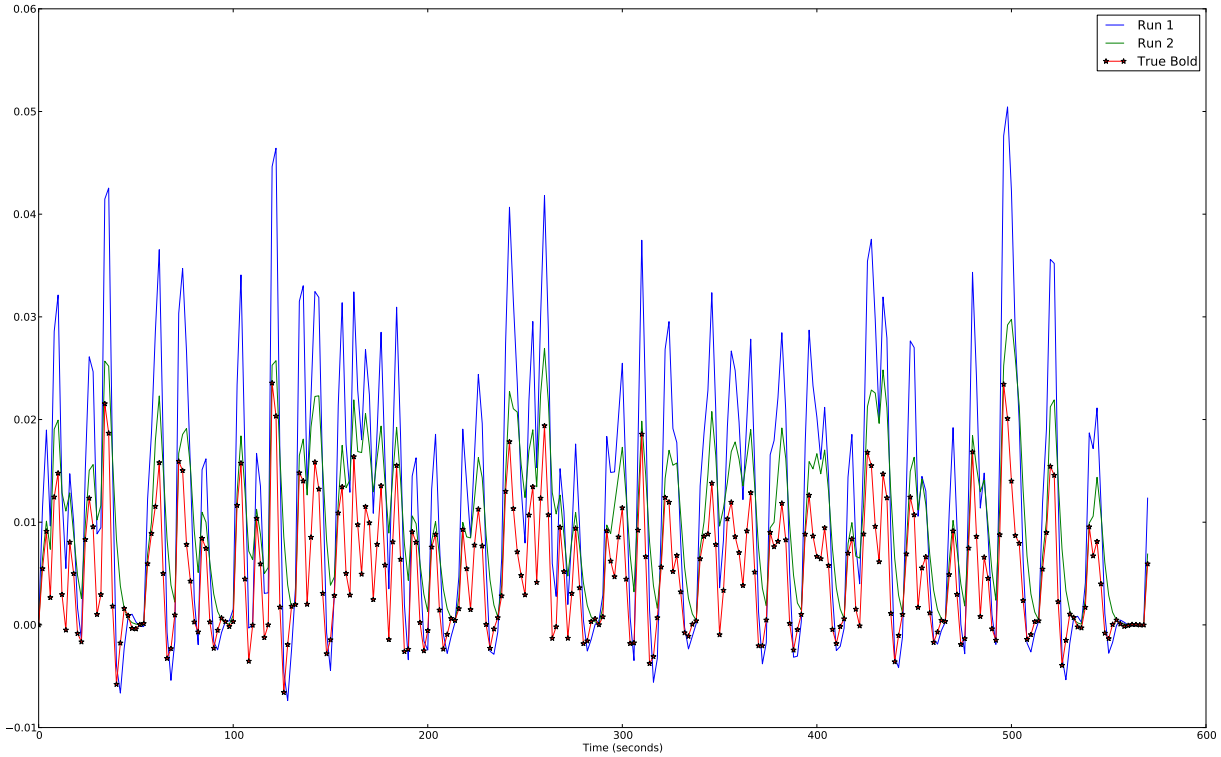
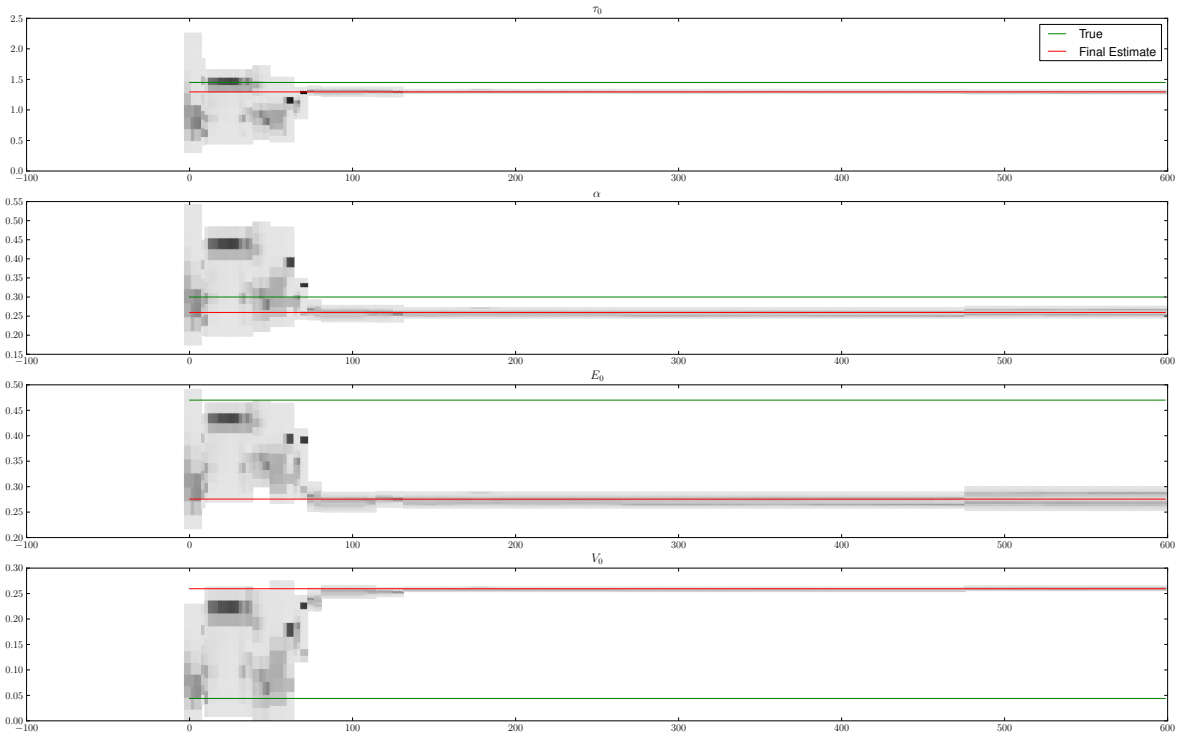
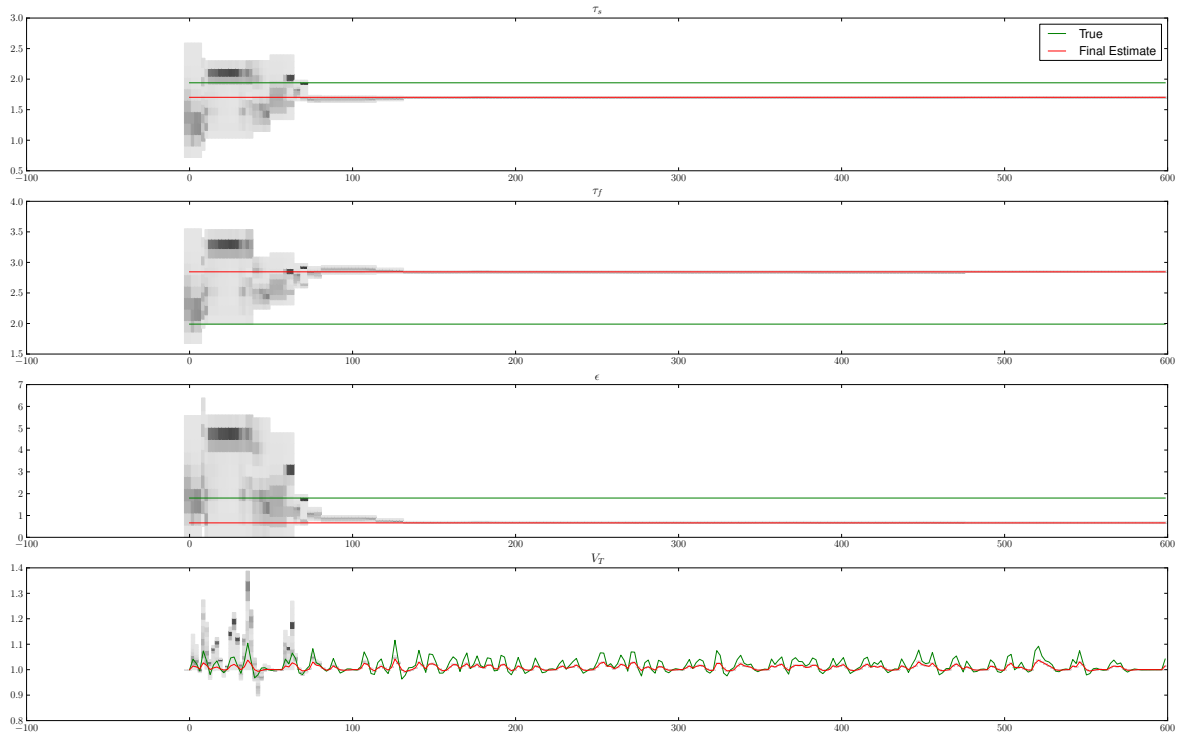


Figure 5.8: The results for the noise realizations shown in Figure 5.1.1.

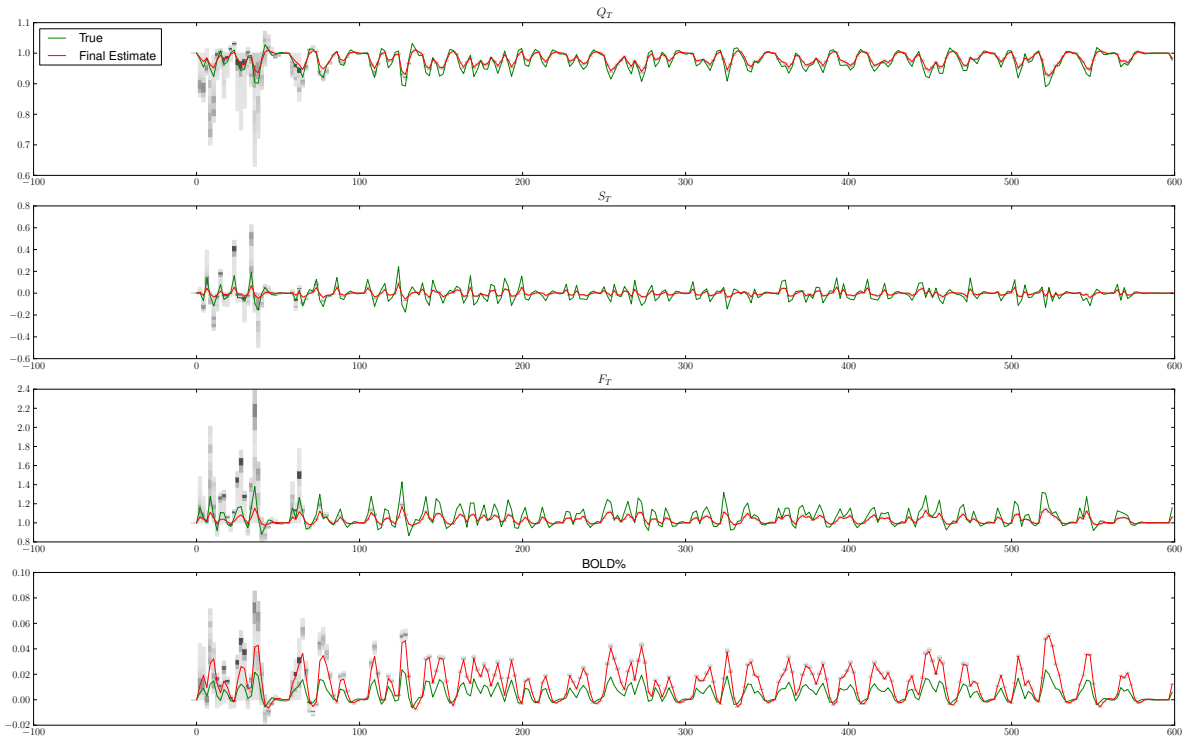
It is interesting to consider how the preprocessing and noise may effect the parameters of the fitting model. For instance run 1 in Figure 5.1.1 certainly seems to be more biased toward higher peaks than run 2. There also appears to be more drift than the 20 points per knot could fit, which explains the prolonged increase at 170 seconds. Despite run 1's bias toward higher peaks, for some reason the particle filter was able to get a much better match for the modeled post-stimulus undershoot (which is likely significantly shorter than it should be). Ultimately the results are pretty good. **??** shows the mean squared error for all two runs, and highlights the two runs analyzed here and in 5.1.1 and 5.1.1.



(a) τ_0, α, E_0, V_0



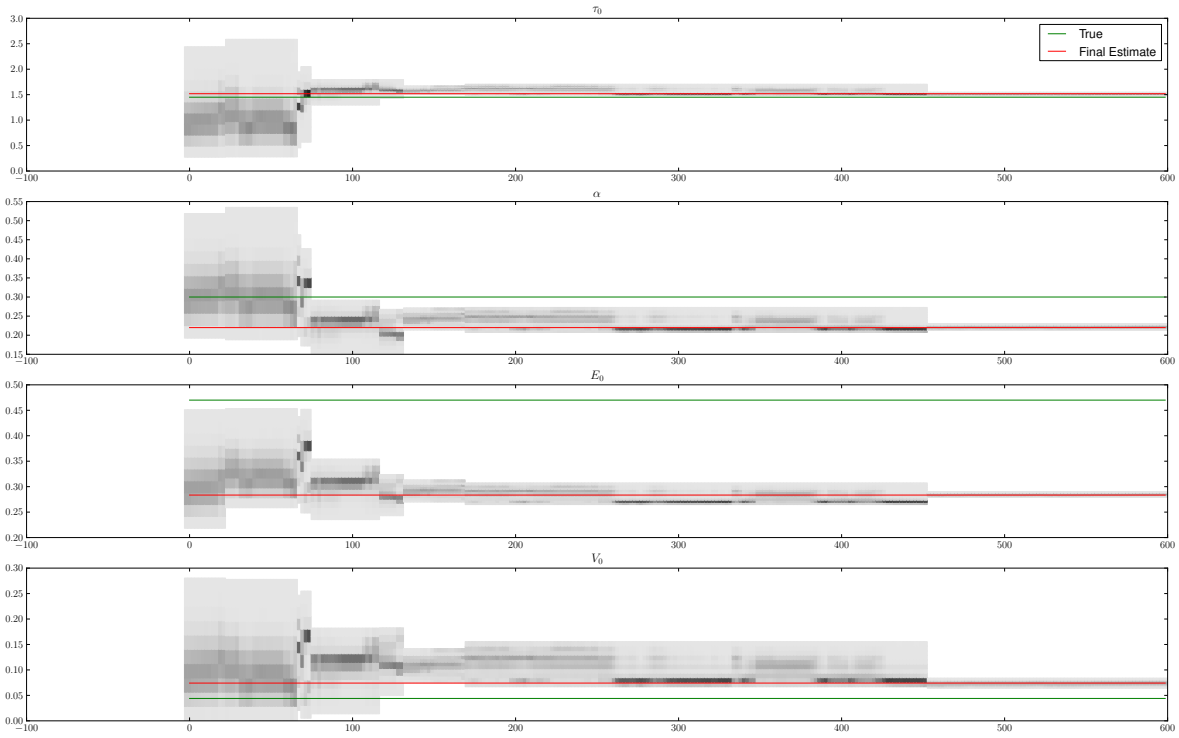
(b) $\tau_s, \tau_f, \epsilon, V$



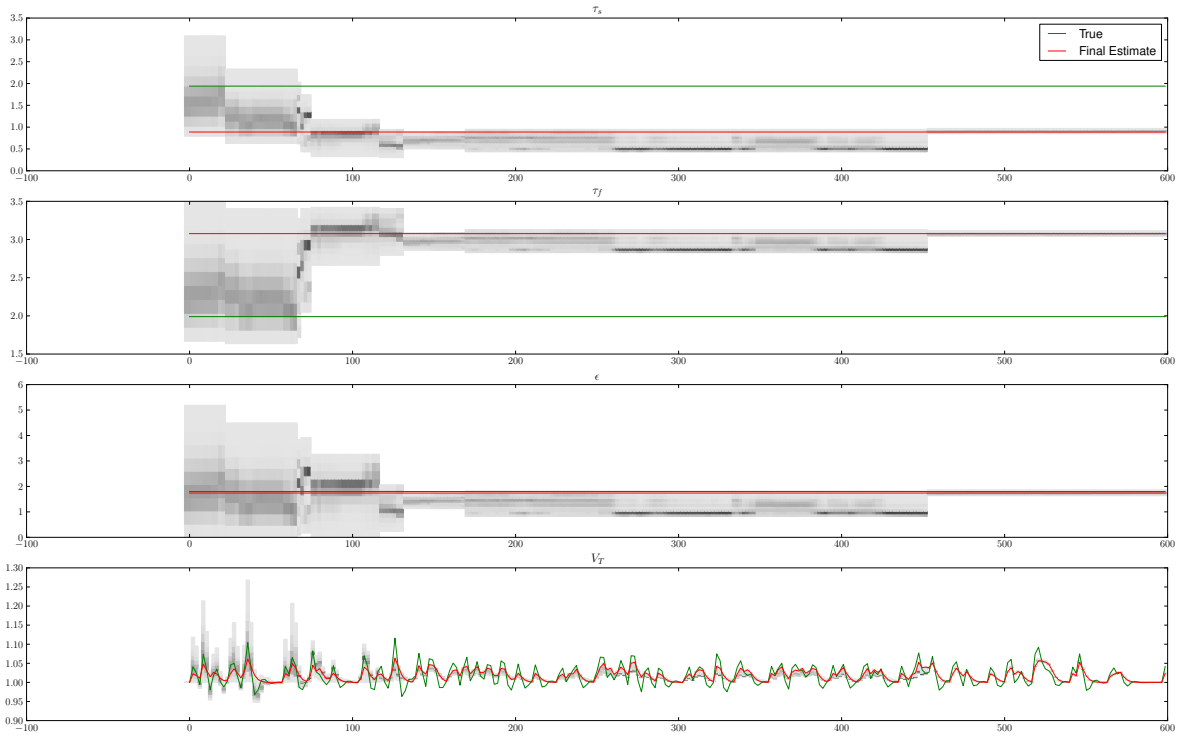
(c) $Q, S, F, BOLD$

Figure 5.9: Converging histogram for parameters during run 2, as in Figure 5.1.1.

There are a number of interesting convergence properties of the particle filter when more noise is present, as both 5.1.1 and 5.1.1 show. Obviously the particle filter seems to converge significantly faster, as points tend to be further out on the weighting function. This also causes significantly more resampling which is the explanation for the seeming jumps in resolution that occur from time to time.



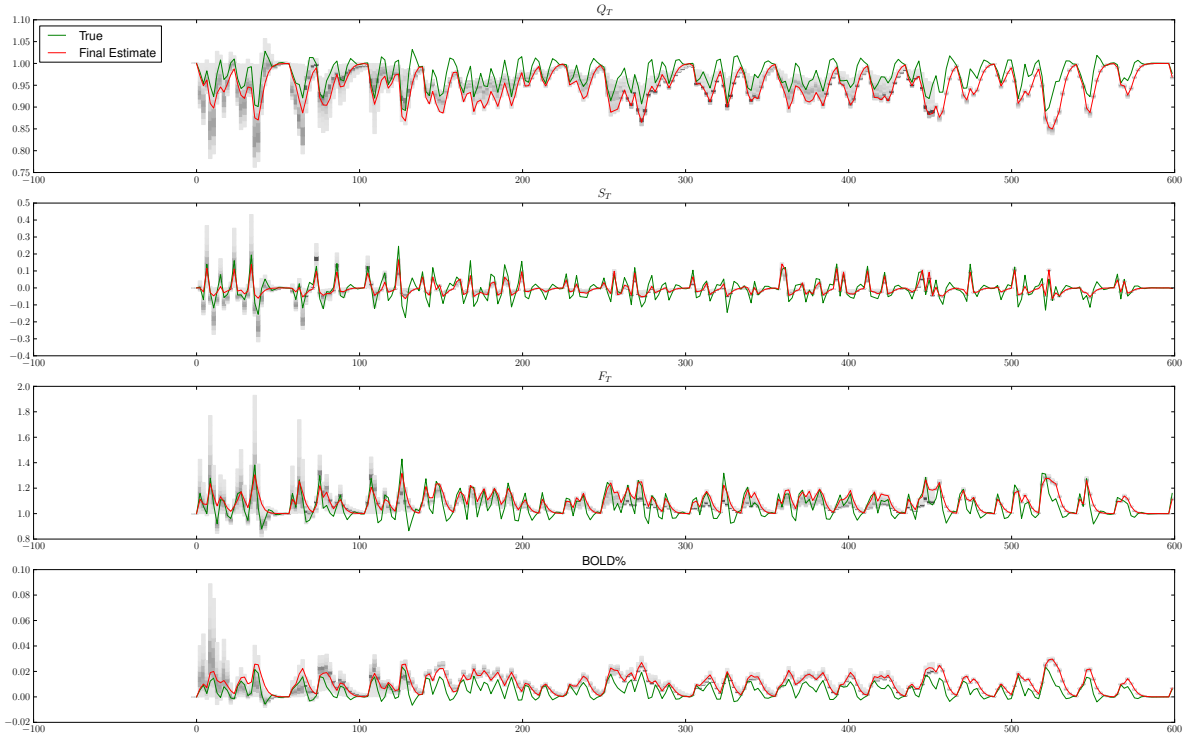
(a) τ_0, α, E_0, V_0



(b) $\tau_s, \tau_f, \epsilon, V$

τ_0	α	E_0	V_0	τ_s	τ_f	ϵ	$\sum \tau$	$\sqrt{MSE}$ (Res.)	$\sqrt{MSE}$
1.45	.3	.47	.044	1.94	1.99	1.8	5.38		
1.19002	0.23495	0.42228	0.128	1.01468	2.47795	1.11677	4.68265	0.0140648	0.01572919
0.9721	0.21902	0.30505	0.061	0.57801	1.99596	3.46135	3.54608	0.0137314	0.01377685
1.57947	0.14153	0.33798	0.1079	0.5843	2.12475	1.78343	4.28852	0.01275372	0.01577449
1.10937	0.2374	0.53491	0.0351	1.21862	3.07365	2.35039	5.40164	0.0167258	0.0115433
1.10712	0.27535	0.33651	0.03161	1.50567	2.65181	4.19099	5.2646	0.01369793	0.01221607
0.58026	0.47931	0.41354	0.11894	0.97563	3.69018	1.00084	5.24607	0.01149511	0.01315637
1.29515	0.25957	0.27559	0.25952	1.70265	2.8458	0.66172	5.84361	0.01555035	0.01789781
1.5185	0.21987	0.28348	0.07417	0.88822	3.07706	1.73929	5.48378	0.01205351	0.01246202
0.68736	0.3283	0.39786	0.15614	1.07781	3.1158	0.66432	4.88097	0.01510364	0.01257713
1.01703	0.28497	0.34741	0.05672	1.58774	2.65157	2.28519	5.25634	0.012493	0.01343459
0.99247	0.29795	0.32207	0.20943	0.42757	2.21081	1.01674	3.63085	0.01216522	0.01505545
1.09535	0.27075	0.36152	0.11259	1.05099	2.71958	1.84282	4.86592	0.01362132	0.01396575

Table 5.3: Estimated Parameters on 10 different runs with high noise. First row is the true parameters. Note also that the blue row is Run 1 and the red row is Run 2, as used named this section



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

Figure 5.10: Converging histogram for parameters during run 2, as in Figure 5.1.1.

The parameters arrived at for all ten filter runs are shown in Table 5.3.

a series of tests to determine the convergence rate of the particle filter, the number of particles that were required, how weighting functions compared, how different de-trending methods compared with each other and, finally the variance of the result. By running the exact time-series with different noise realizations, it was

possible to determine the model variance. As the reader may know, the error of an estimator may be calculated as:

$$MSE(\Theta) = Var(\Theta) + Bias(\Theta)^2 \tag{5.1}$$

The variance is an expression of how much the result would change for different noise realizations, whereas the bias is an expression of how well the model matches the true underlying model. In this case, because the same model is being used in the particle filter and underlying simulation, the bias is actually zero. Obviously when this is calculated using *real* data with an unknown underlying state space equation, there will be some amount of bias error, but assuming that the noise is similar to the noise used in these tests, the model variance will actually be about the same. Thus calculating the model variance is extremely helpful in calculating how well determined our model is, and how consistent it will be for real data. A single timeseries, as opposed to the thousands present in a real image, makes it easier to compare the output with the ground truth, with various parameters.

Second I used a modified version of the FSL tool POSSUM to generate an entire FMRI image from a parameter map. The parameter map was generated by taking an existing activation map and assigning discrete parameter sets to each region. The result was a four dimensional (length x width x height x parameter) image with spatially varying parameters. Possum was then modified to take a parameter map and generate activation levels depending on the parameters at that point. The patch for POSSUM will be made available. As an unfortunate side effect of not using Possums' original activation scale, I manually added 750 to the total level of simulated Possum images. This is because the BOLD % difference levels were in the range of 50 - 100% from the base, about 5 times as large as they should have been. Ultimately this should not have an effect on the parameters (other than perhaps ϵ and V_0). For each time-series in the simulated FMRI image, the final *static* parameters were saved into a parameter map. This parameter map may then be compared to the map used to generate the simulated data; additionally a new simulation using the calculated parameters may also be generated to test the difference in activation levels between the real parameters and the estimated ones. Since it is clear that the parameters are not fully orthogonal ([4]), its possible that two sets of parameters are functionally equivalent, but have different parameters. This way, an absolute quantitative difference between the two parameter sets may be found.

5.2 Real Data

Finally, we also performed inference based on real FMRI data. The scanner we used...

Before performing tests on a full image, I tested the results of the particle filter on regions deemed active

and non-active by statistical parametric mapping.

5.3 Single-Voxel Simulation

The results

5.4 Single-Voxel Analysis

This section discusses the results when the particle filter was applied on a single voxel. The parameters are the same as those used later for entire image analysis; however, the results are more in-depth.

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 10 to 15 minutes. On the other hand, in real circumstances the length is only around 150 measurements. The number of integration can certainly make a large difference, however dropping below 1000 (.001 seconds) is definitely not recommended.

For a period I considered 1000 to be a fine number; however when generating simulated data I found that every once in a while 1000 was not enough. This is problematic in the actual particle filter since, given the large number of simultaneous integrations taking place, its likely that a few particles will fail and be weighted at 0 because of this problem. Additionally, because the typical case where a failure would occur is at fast moving times/parameters the particles all tend to fail together. The result is particle deprivation - no particles with non-zero weights remain. The other possible outcome is that low time constant particles get pruned resulting in excessively smoothed estimates for the time series'. Its possible that a kind of stop-gap measure could be put into place; wherein particles that are about to be set to NaN are integrated again with finer grained steps. However many times the non-real results don't occur until several time steps after the numbers get strange. So for instance, the timestep was too long, allowing f to go negative, resulting in extremely large values of q. There are many different ways where this sort of event can occur, and unfortunately sometimes there is no way to get back to before the state starting going out of control.

Another crucial factor for run time is how long before the first re-sampling occurs. Because the prior is represented initially with significantly more particles, if the model fits very well, or for some reason the effective number of particles stays high, resampling could take a long time to occur. When this happens the particles filter can take a factor of 10 longer to run. However, if the particle count isn't initially set high, there is a much larger chance of particle deprivation occurring. Since there is no real way to know how long it will take to resmaple the first time, there is little the algorithm can do to fix this (except perhaps forcing resampling after some period of time). On the other end of the spectrum, if the time series doesn't match the model at all, particle deprivation

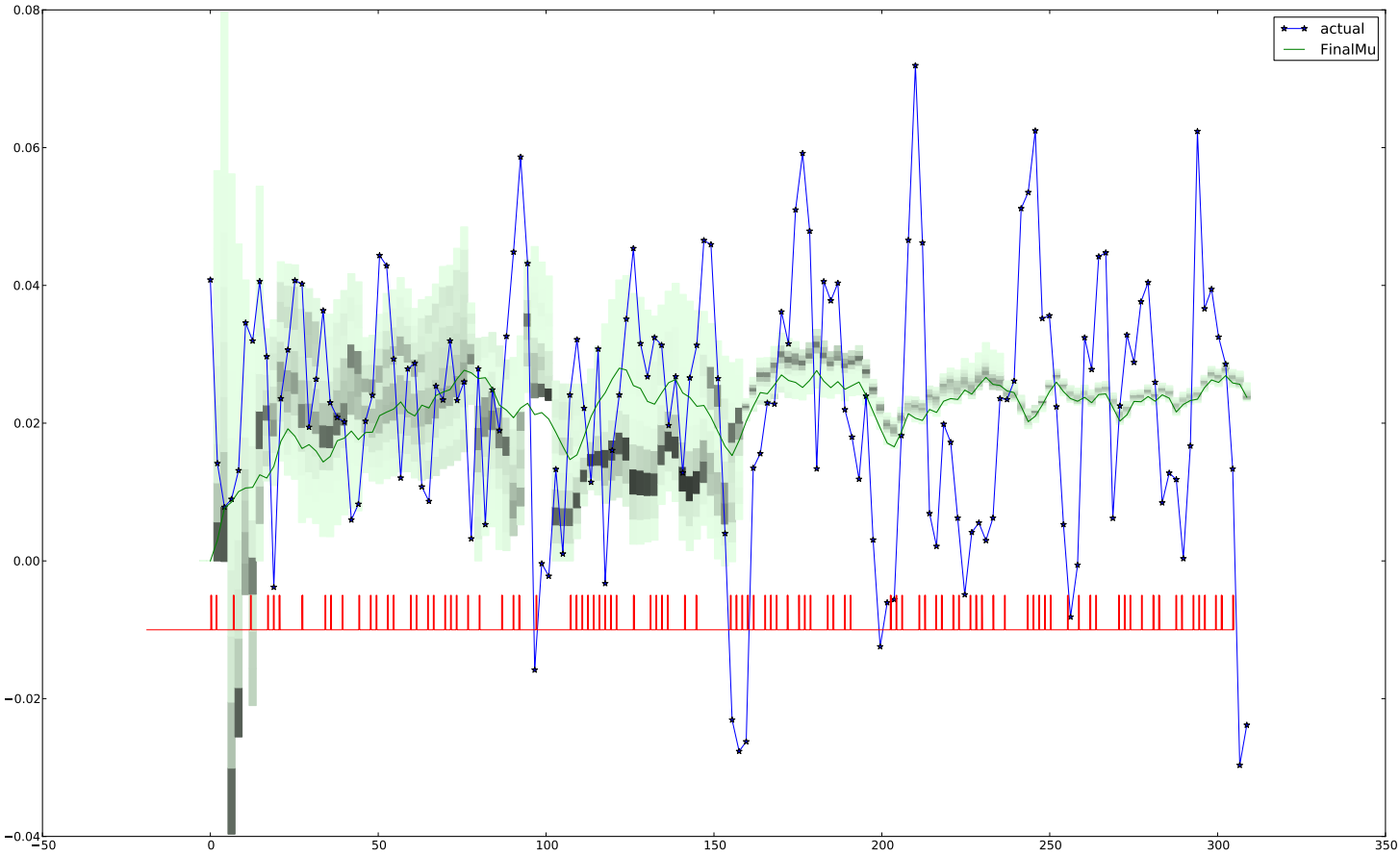


Figure 5.11: Particle Filter converging to values that make little sense, because the voxel did not correlate with the input in any known way

will occur extremely quickly. The upshot of this is that the particle filter is able to identify these sections very quickly, and thus not waste much time there. The difficulty though, is the regions in between the perfect fit and the awful fit. If particle deprivation does occur, did it occur randomly to an activated region or did it occur inevitably because the region doesn't fit. The reason for having so many initial particles is to give density to the distribution to make false negatives more rare. Perhaps the correct method is to still set the initial particles very high, but if for a long time no resampling occurs, halve the standard deviation of the measurement error. Thus, if the weighting function is not discriminating enough, force it to be more picky about the results. Of course this could lead to particle deprivation as well if the standard deviation is brought down too quickly. Of course, if a fat-tailed distribution is used for the weighting function, or the standard deviation of a Gaussian weighting function is sufficiently large, the particle filter will simply converge to meaningless values. The question of whether

For this reason, it is important to threshold based on the MSE, and the impulse response. Naturally the MSE is intended to catch very voxels with a very poor fit, but like section 5.4, the algorithm may have found a line

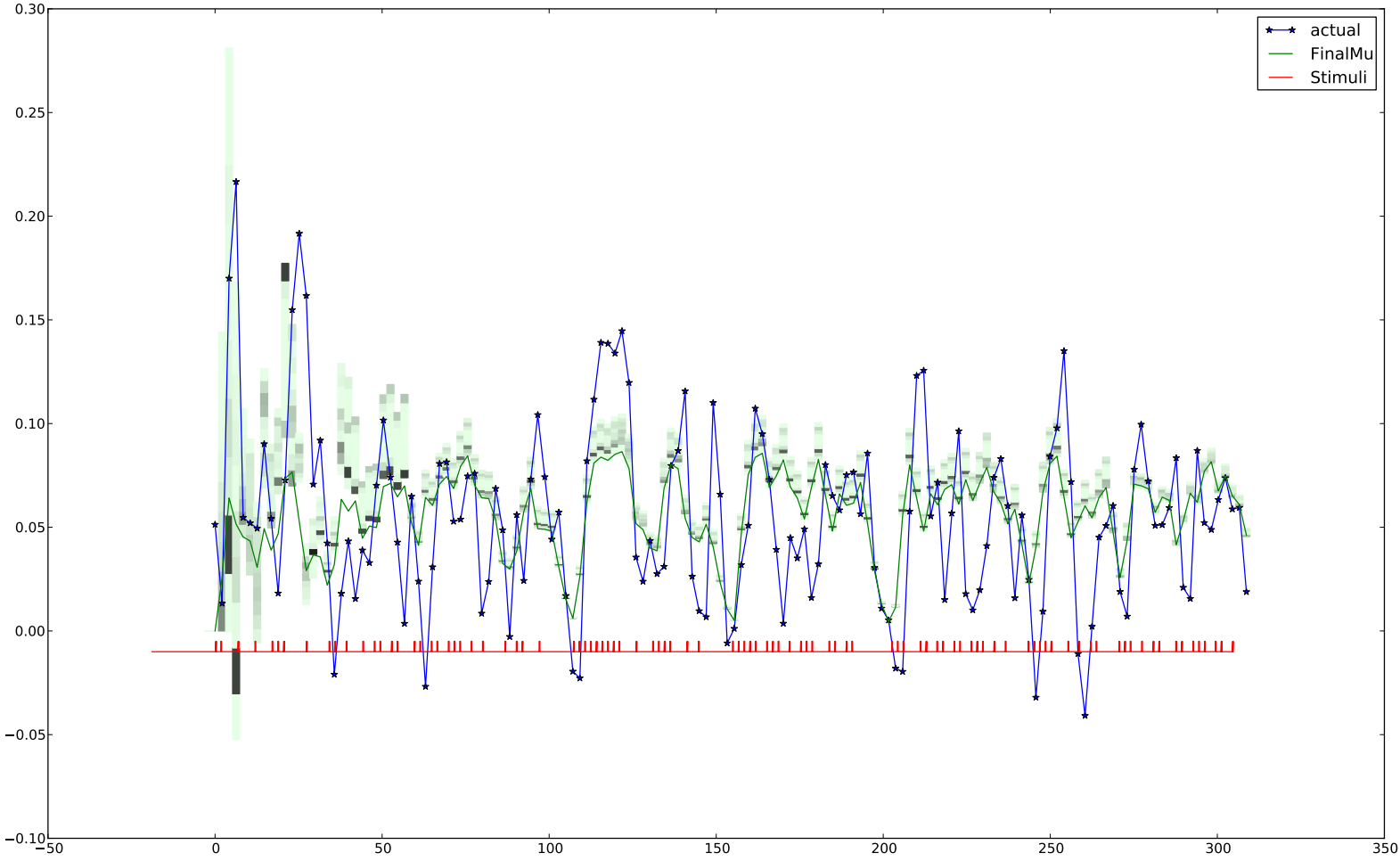


Figure 5.12: Example of a timeseries that laplace performs better on than gaussian. The large spike at the beginning can quickly eliminate all particles

down the middle that gives a decent mean squared error. In this case the excessive smoothing and low actual activation level will easily be distinguished by finding the peak response to single short pulse.

The choice of a prior, as discussed previously, is extremely important. While a prior may have the potential to give good results, being a monte-carlo algorithm there is the possibility for inconsistencies. Thus, increasing the variance of the time-constants may allow additional flexibility, it will also cause additional model variance. Case in point, the exact same algorithm run twice with standard deviations of $.35, .35, .35$ for the time constants resulted in two very different fits, see **??**.

For this reason, I actually lowered the standard deviattions of the time constants to prevent over-smoothing. This resulted in the more consistent, but slightly worse results seen in Figure 5.4

5.5 Particle Collapse Recovery

This heuristic was initially intended to deal with cases where the particle filter prematurely converged; and to reverse this effect. In reality, however it did not perform well, and so it was not used in full-brain analysis. Ultimately it did not prevent collapse of the distribution, and it often caused dramatic increased in the variance of parameter estimates. It also pronged the analysis time of inactive time series. Essentially the effect was exactly opposite of what one would hope.

5.6 Weighting Function Comparison

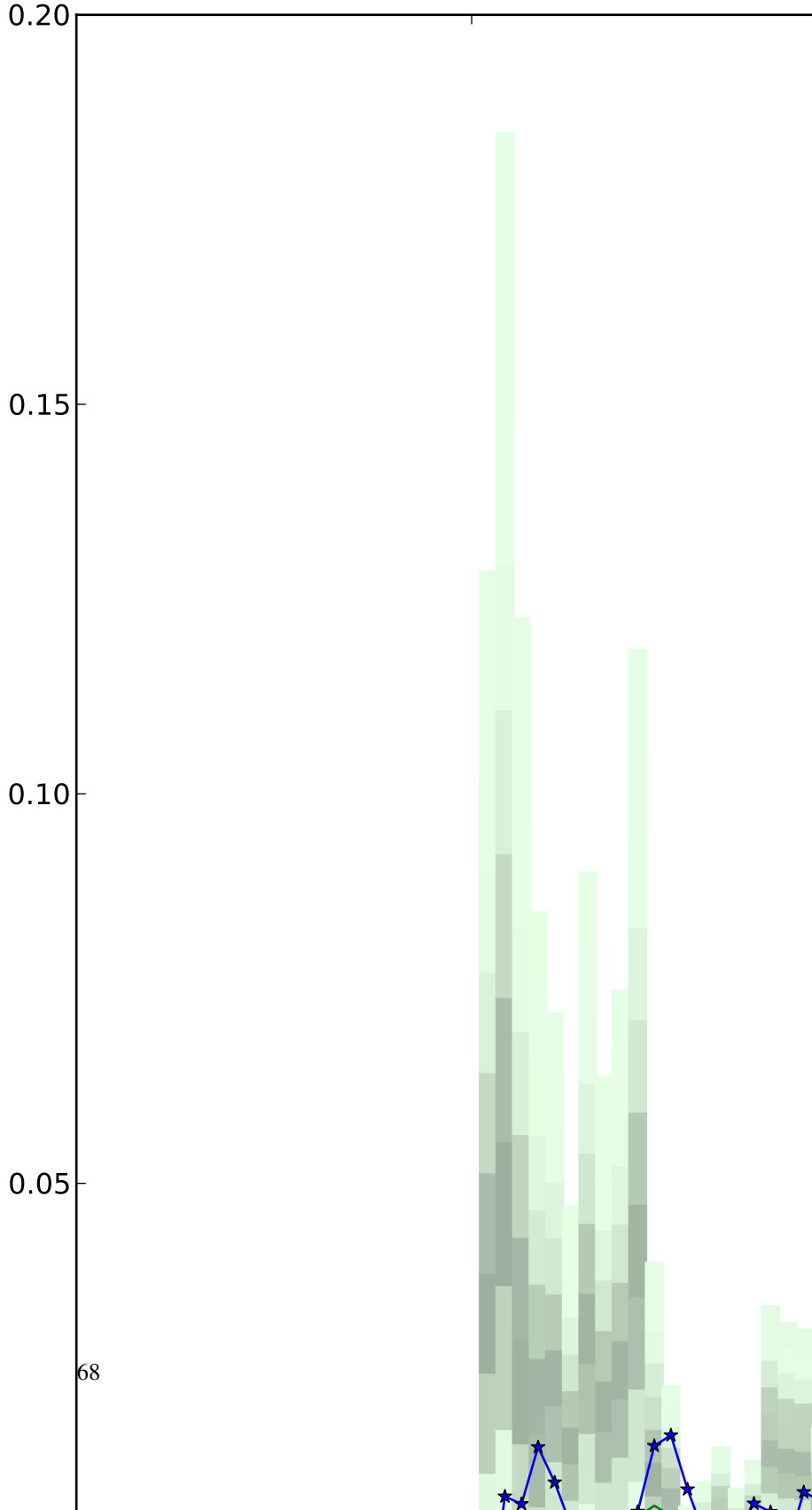
5.7 Single Time-Series Simulation

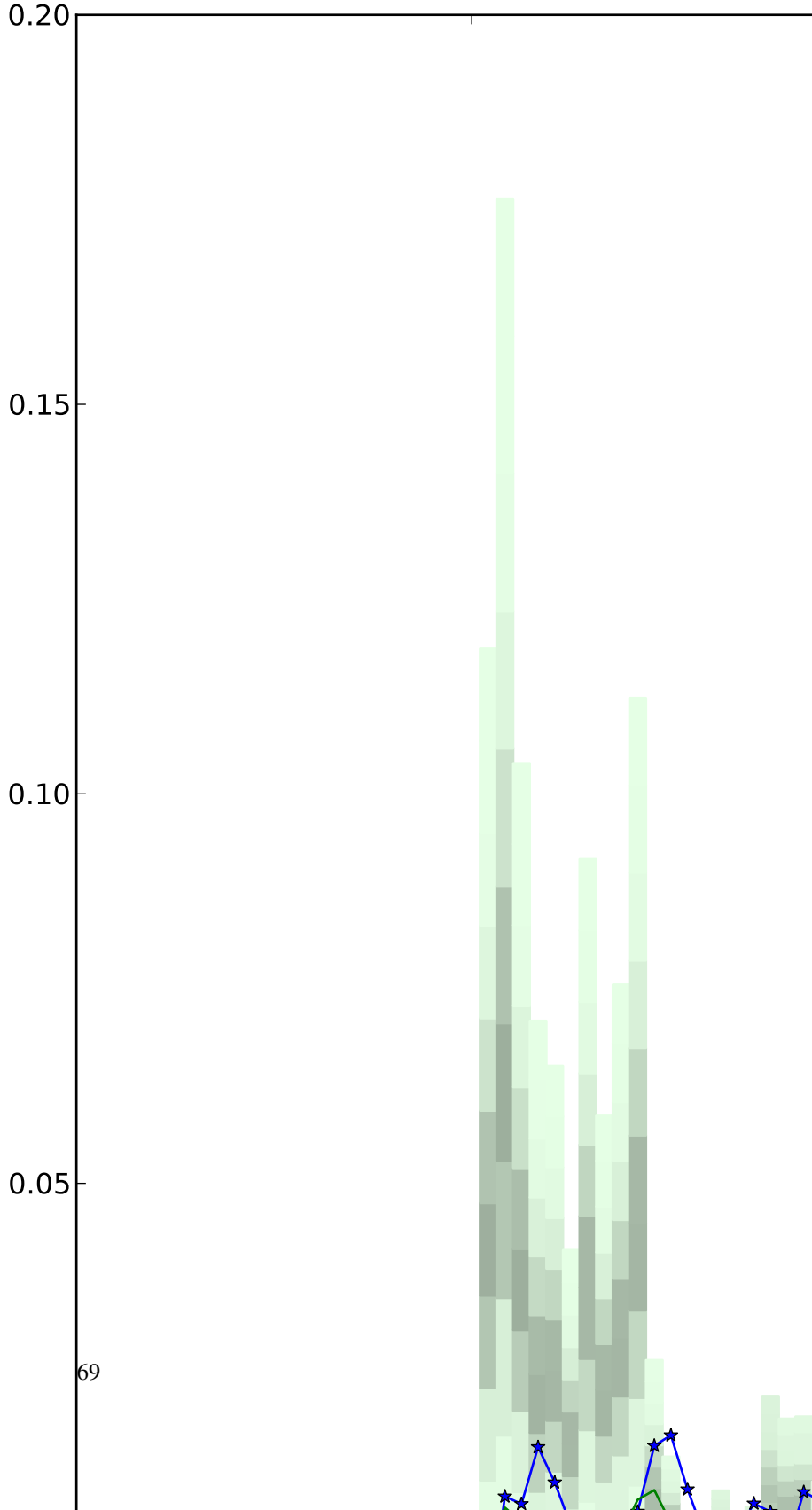
Graphs:

For simulated data, single timeseries:

For {delta, DC/Spline}, {exponential, gaussian, cauchy}, {biased, unbiased initial}, {100, 500, 1000} particles

1. Ground truth vs. Estimated signal during particle filter run
2. Ground truth vs. Estimated signal with final parameter set
3. True Parameters vs. Final Parameter Sets
4. Variance of final parameters when faced with same ground truth, different noise
5. MSE of (a new timeseries based on X(t) vs. ground truth) for all t


0.20
0.15
0.10
0.05


0.20
0.15
0.10
0.05

6. Estimator Variance based on different noise runs

7. Final Particle Distribution

For Simulated Data, Full Volume:

5.8 Simulated Volume

1. Parameter Map

2. Error map of parameters

3. Histogram of %errors between parameters

4. Activation Map based on a single region with high ϵ, compared with linear

Final parameter distribution among active regions. Q-Q plots?

5.9 FMRI Data

....

image comparing epsilon-map with GLM activation map

Chapter 6

Conclusion

6.1 Future Work

*A more extensive investigation of the parameters distribution *Using a compliance model *restrict parameters more (may not be possible)

Bibliography

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] J. J. Chen and G. B. Pike, "Origins of the BOLD post-stimulus undershoot.," *NeuroImage*, vol. 46, no. 3, pp. 559–68, 2009.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] J. J. Chen and G. B. Pike, "Origins of the BOLD post-stimulus undershoot.," *NeuroImage*, vol. 46, no. 3, pp. 559–68, 2009.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] K. J. Friston, W. Penny, C. Phillips, S. Kiebel, G. Hinton, and J. Ashburner, "Classical and Bayesian inference in neuroimaging: theory.," June 2002.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] D. A. Hofmann, "An Overview of the Logic and Rationale of Hierarchical Linear Models," *Journal of Management*, vol. 23, no. 6, 1997.

[33] K. J. Friston, "Bayesian estimation of dynamical systems: an application to fMRI," *NeuroImage*, vol. 16, pp. 513–530, 2002.

[34] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press, 2005.

[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] 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.

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

[38] 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.

[39] 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.