Brain tissue temperature dynamics during functional activity and possibilities for optical measurement techniques

by

Greggory H. Rothmeier

Under the Direction of A. G. Unil Perera, Mukesh Dhamala

Abstract

Regional tissue temperature dynamics in the brain is determined by the balance of the metabolic heat production rate and heat exchange with blood flowing through capillaries embedded in the tissue, the surrounding tissues and the environment. Local changes in blood flow and metabolism during functional activity can upset this balance and induce transient temperature changes. Invasive experimental studies in animal models have established that the brain temperature changes during functional activity are observable and a definitive relationship exists between temperature and brain activity. We present a theoretical framework that links tissue temperature dynamics with hemodynamic activity allowing us to non-invasively estimate brain temperature changes from experimentally measured blood-oxygen level dependent (BOLD) signals. With this unified approach, we are able to pinpoint the mechanisms for hemodynamic activity-related temperature increases and decreases. In addition to this, the potential uses and limitations of optical measurements are discussed.

INDEX WORDS: Functional magnetic resonance imaging, Blood oxygen level dependent, Brain temperature, Functional near-infrared spectroscopy

Brain tissue temperature dynamics during functional activity and

possibilities for optical measurement techniques

by

Greggory H. Rothmeier

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

in the College of Arts and Sciences

Georgia State University

2012

Brain tissue temperature dynamics during functional activity and

possibilities for optical measurement techniques

by

Greggory H. Rothmeier

Committee Chairs:	Co-	A. G. Unil Perera
		Mukesh Dhamala
Committee:		Brian Thoms
		D. Michael Crenshaw

Electronic Version Approved:

Office of Graduate Studies

College of Arts and Sciences

Georgia State University

May 2012

Dedication

This is dedicated to my parents who made me go to college and to Brooke who inspired me to go to graduate school. If I wasn't lucky enough to have all of you I would probably be working for Geek Squad.

Acknowledgments

Table of Contents

List of Tables

List of Figures

List of Abbreviations

BOLD	Blood Oxygen Level Dependent
CBF	Cerebral Blood Flow
$CMRO_2$	Cerebral metabolic rate of O_2
CSF	Cerebral Spinal Fluid
deoxyHb	Deoxyhemoglobin
fMRI	Functional Magnetic Resonance Imaging
fNIRS	Functional Near-Infrared Spectroscopy
OLM	Oxygen Limitation Model
oxyHb	Oxyhemoglobin
ROI	Region of Interest
TRS	Time Resolved Spectroscopy

1 Introduction

Since its invention in the 1950's [4] and later development in the 1970's [5], Magnetic Resonance Imaging (MRI) has allowed physicians and scientists a detailed view within the human body. Beginning in the 1990's, it was realized that changes in the local metabolic state affect the local magnetic resonance and provide an indication of brain activity [6, 7]. This is possible because hemoglobin is diamagnetic in an oxygenated state and paramagnetic when deoxygenated. Thus as the local concentration of deoxyhemoglobin changes, the local magnetic susceptibility is also altered.

By combining this effect with a discovery made earlier in 1986 [8], MRI becomes a powerful tool for measuring brain activity. Fox and Raichle [8] found that with an increase in neuronal activity came an increase in local cerebral blood flow (CBF) that exceeded the increase in cerebral metabolic rate of oxygen ($CMRO_2$). The change in local tissue oxygenation created by uncoupled changes in CBF and $CMRO_2$ is referred to as the blood oxygen dependent (BOLD) response [7]. A schematic of the model for generation of the fMRI BOLD response is provided in Fig. 1.1.

A stimulus within a region of the brain induces either an excitatory or inhibitory (or a combination) neuronal response. An increase in excitatory neuronal activity triggers an increase in CBF which overcompensates for the increase in $CMRO_2$ [8]. Conversely, an increase in inhibitory neuronal activity does not induce a change in CBF. Increasing $CMRO_2$ increases the concentration of deoxyhemoglobin (deoxyHb) while increasing CBF delivers more blood to the tissue thereby increasing the concentration of oxyhemoglobin (oxyHb) and increasing local tissue oxygenation. The change in blood oxygenation is detected by the fMRI as the BOLD response [7].

Along with changes in the BOLD response, changes in CBF and $CMRO_2$ also affect the local tissue temperature. As glucose is metabolized, heat is released that is primarily
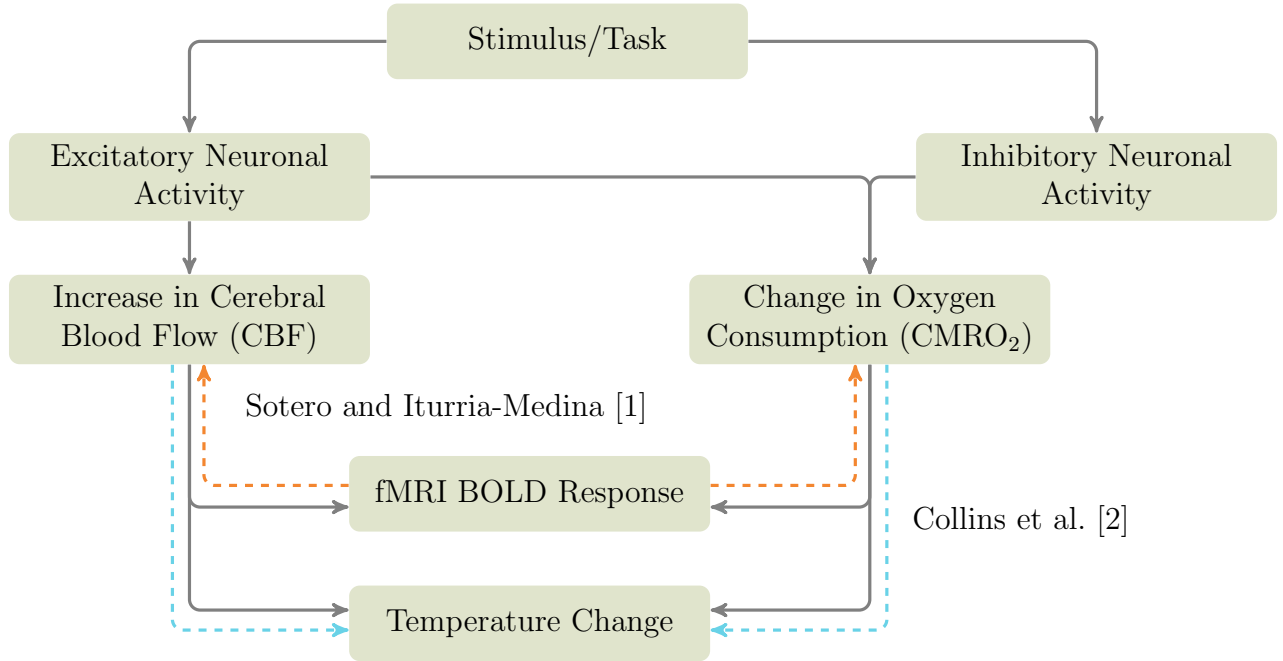
Figure 1.1   Generation of the fMRI BOLD response from changes in neuronal activity. Black arrows indicate a causal relationship while colored dashed-arrows indicate existing models for the relationship. The orange line (•) shows the model proposed by Sotero and Iturria-Medina [1] to calculate cerebral blood flow and metabolism and the blue line (•) shows how the model proposed by Collins et al. [2] is used to calculate temperature. Modified after Sotero and Trujillo-Barreto [3].

dissipated by convection to blood. The human brain relies on CBF to remove excess heat from the brain [9]; thus, a change in rate of blood flow will affect how quickly heat can be dissipated (or delivered as we'll explore in section 2.3). Since both BOLD and temperature are dependent on CBF and $CMRO_2$, it is possible to use the BOLD data collected to calculate the temperature change. As will be further discussed in section 2.3.1, the resulting temperature change cannot be easily characterized for the entire brain because it's behavior is spatially dependent.

Experimental measurements of activity-induced brain temperature changes are mixed in whether an increase in brain activity increases or decreases local tissue temperature [10, 11, 12, 13, 14]. Current temperature models predict that an increase in activity will result in a decrease in temperature [1, 15, 16]. This is generally the prediction because these models

generalize the resting-state conditions of the voxel (ignoring spatial dependence) which puts the blood temperature below the resting state tissue temperature. An increase in blood flow then acts as a coolant for the tissue and lowers the temperature (more on this in section 2.1).

An accurate model for brain temperature could be valuable for many clinical applications [17]. It has been shown that treating severe head injuries with induced mild hypothermia greatly improves the clinical outcome [18]. A similar result has been shown in patients being treated for brain ischemia [19, 20] and stroke [21]. Additionally, it has been shown that mild hypothermia can be neuroprotective in infants undergoing heart surgery [22]. During treatment, the brain temperature is currently inferred by indirect means. This can be achieved by measuring the temperature of the temporalis muscle [23], the tympanic canal [24, 25], nasopharynx [26] or by the use of a venous bulb in the jugular [27, 28]. While commonly used, their reliability has not been validated [25, 28]. A model of brain temperature could let physicians better understand the relationship between brain temperature and the measurements they are able to make. This could make hypothermia-based treatments more effective since the brain temperature could be better controlled.

I will explore a model which calculates temperature using the fMRI BOLD response for the entire brain, thereby accounting for spatial dependencies. Additionally, in chapter 3 I'll explore optical measurement techniques including functional near-infrared spectroscopy (fNIRS) and the use of thermal imaging cameras to measure activity-induced brain temperature changes.

2 Calculating Temperature Changes using the fMRI BOLD Response

2.1 Introduction

Using fMRI to find brain temperature is enticing because it is noninvasive. Existing efforts to model temperature changes be can categorized into two classes. The first class approaches the problem by considering a single region of interest (ROI) deep within the brain (single-voxel approach) while the second approach considers the brain and head as an entire system (multi-voxel approach). The single-voxel approach is unable to explain the experimental observation of an increase in temperature during a task; however a multi-voxel approach is able to account for this observation.

2.1.1 Single-Voxel Methods

Numerous single-voxel approaches have been created [1, 15, 16] which differ largly only in their formulation of Penne's Bioheat Eqution [29]. Although different approaches consider different contributions to the temperature change, they all narrow the problem down to a single region of interest. By simplifying the model, the heat equation can be simplified and the calculation is much easier to undertake. However, since the brain is not homogenous, the values used for parameters such as heat production and thermal conductivity are taken from an average of the tissues. As a result, this reduces the level of accuracy these models can achieve.

The most recently published iteration of a single-voxel model was published by Sotero and Iturria-Medina [1] in 2011. Their formulation of Penne's Bioheat Equation is as follows [29, 1].

$$
\begin{aligned}
C_{tissue}\frac{dT(t)}{dt} = (\Delta H^\circ - \Delta H_b)CMRO_2\mid_0 m(t) - \rho_b C_b CBF\mid_0 f(t)(T(t) - T_a) \\
- \frac{C_t}{\tau}(T(t) - T_0)
\end{aligned}
\tag{2.1}
$$

where C$_{tissue}$ is the specific heat of the tissue, ΔH° is the enthalpy released in the oxidation

of glucose, ΔH_b is the enthalpy used to release oxygen from hemoglobin, CMRO$_2$ $|_0$ is the metabolic rate of oxygen at rest, ρ_b is the blood density, C_b is the specific heat of blood, CBF$|_0$ is the cerebral blood flow at rest, T_a is the arterial blood temperature, C_T is the specific heat for the tissue, and τ is a time constant for conductive heat loss. The values used are provided in Table 2.2.

One advantage of using Eq. (2.1) is that the resting state temperature can be analytically determined by substituting $\frac{dT(t)}{dt} = 0$ [1].

$$T_0 = T_a + \frac{(\Delta H \mid^\circ - \Delta H_b)CMRO_2 \mid_0}{\rho_B C_B CBF \mid_0} \tag{2.2}$$

If the values provided in Table 2.2 are substitued into Eq. (2.2), a resting temperature of 37.3057°C is found (using $T_a = 37$°C). Regardless of the arterial blood temperature, since the second term of Eq. (2.2) is always positive the resting state temperature will always be greater than the arterial blood temperature. Thus, an increase in blood flow will remove heat from the ROI, thereby lowering the temperature. As will be further discussed in section 2.3.1, this is an acceptable result for the majority of the brain; however a multi-voxel approach reveals that a region of the brain exists where a single-voxel model is unable to predict temperature changes. Since it is a simpler model than a multi-voxel model, it is easier to first understand the principles of Penne's Bioheat Equation using this model before discussing how it can be applied to the entire head.

While Eq. (2.1) is appears complicated, conceptually the equation can be easily understood:

$$change\ in\ temperature\ =\ heat\ generated\ by\ metabolism\ -\ heat\ lost\ due\ to\ convection$$
$$-\ heat\ lost\ due\ to\ conduction \tag{2.3}$$

The system is a balance between heat generation (metabolism) and heat transfer (conduction and convection). The direction of heat transfer by convection is determined by the difference

between the voxel temperature and the arterial blood temperature $(T(t) - T_a)$. Similarly, the direction of heat transfer by conduction is determined by the difference between the voxel temperature and the temperature of the surrounding tissue $(T(t) - T_0)$. Since T_a is less than T_0, an increase in blood flow $(f(t))$ will remove heat from the voxel thereby decreasing the temperature. Conversely, an increase in metabolism $(m(t))$ without a corresponding change in blood flow, will result in tissue warming.

2.1.2 Multi-Voxel Methods

The multi-voxel approach to calculating brain tissue temperature alleviates many of the issues that a single-voxel approach has. The most prominent advantage a multi-voxel approach has is a result of it accounting for a voxel's location relative to the surface of the head and other voxels. By accounting for a voxel's location, the same BOLD response in two different locations can have vastly different effects on the local tissue temperature (more on this in section 2.3.1).

Multi-voxel methods use a three-dimensional implementation of Penne's Bioheat Equation [2].

$$\rho c \frac{dT}{dt} = k \nabla^2 T - \rho_{blood} f(t) w c_{blood} (T - T_{blood}) + m(t) Q_m \qquad (2.4)$$

where ρ is the tissue density, c is the specific heat of the voxel, k is the thermal conductivity, ρ_{blood} is the blood density, w is perfusion by blood, c_{blood} is the specific heat of blood, T_{blood} is the arterial blood temperature, and Q_m is the baseline metabolic heat production. $f(t)$ and $m(t)$ are the time-dependent changes in blood flow and metabolism. These two factors determine the short-term change in temperature and are calculated from the fMRI BOLD response; however what makes this approach more complete than a single-voxel approach is that the relatively slow conductive heat loss makes for a different equilibrium temperature at each voxel. This effect can only be captured by considering the entire head. The approach we use is a multi-voxel approach, so more details about this model are discussed in section 2.2.

2.2 Our Approach

Our approach combines a multi-voxel model (section 2.1.2) with a model for calculating the change in metabolism and blood flow from the BOLD response. Penne's Bioheat Equation (Eq. (2.4)) [29, 1] includes three terms. The first and second terms describe heat exchange by conduction to surrounding tissues (or air) and convection to blood, respectively. The third term describes heat generated by metabolism. On shorter time scales, the first two terms dominate and are sufficient for determining activity-induced temperature changes; however, the third term becomes important on longer time scales because it plays a role in determining the resting-state temperature.

Conductive heat transfer to surrounding tissues is a comparatively slow process, but on larger time scales, conduction plays an important role in determining the resting state temperature. When calculating the temperature change, it is important to first have an accurate resting state temperature since it can either be greater than or less than the arterial blood temperature. By considering the entire head, the model we use is able to accurately determine a resting state temperature for each voxel, enabling far more accurate temperature calculations than what is capable with single-voxel approaches.

Figure 2.1 gives a schematic of the temperature calculation procedure. The orange blocks represent the required data. The first thing to be done is establish the resting-state temperature for each voxel within the head. The details of this procedure are given in section 2.2.2, but in summary a T1 contrast image is segmented using SPM8 (Fig. 2.2) and is combined with tissue-specific parameters (Table 2.1). The resulting dataset is then used to determine the resting-state temperature by repeatedly applying Eq. (2.4) until the temperature stabilizes for all voxels.

The resting-state time slices from the fMRI BOLD dataset are averaged to create a resting state representative slice. The remaining BOLD data is then normalized to this in order to have the normalized change in BOLD response ($\frac{\Delta S}{S_0}$ in Eq. (2.16)). This can then be used with Eqs. (2.9), (2.15) and (2.16) to create a time series for the change in blood flow and
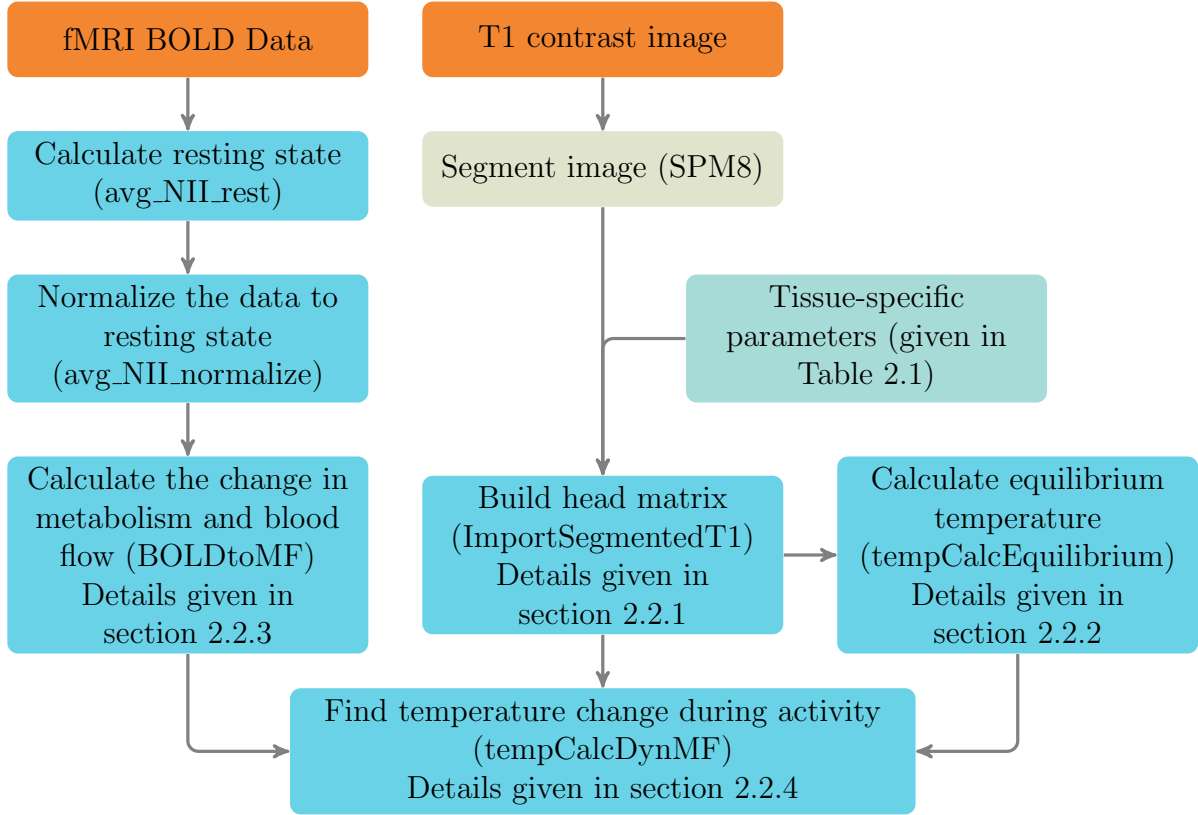
Figure 2.1   The procedure used to calculate temperature from BOLD data. Orange blocks (•) represent data, the sandy-colored block (•) is a step done using SPM8 and the teal blocks (•) are steps done using a function provided within temptools (appendix A). The name of the function used is in parentheses.

metabolism. More details about this procedure are given in section 2.2.3.

The following sections provide a detailed explanation of the theory behind our modeling approach. The code used to implement this procedure is provided and documented in appendix A.

2.2.1   Preparing the model of the head

In order to begin the temperature calculating procedure, a model of the head must first be created. Using SPM8 (http://www.fil.ion.ucl.ac.uk/spm/), we segmented a T1 contrast image of the head into five different tissue types: bone, cerebral spinal fluid, gray matter, white matter and soft tissue. It was assumed that soft tissue voxels that are in contact with air are more appropriately labeled as skin, so in total we are left with a model of the head
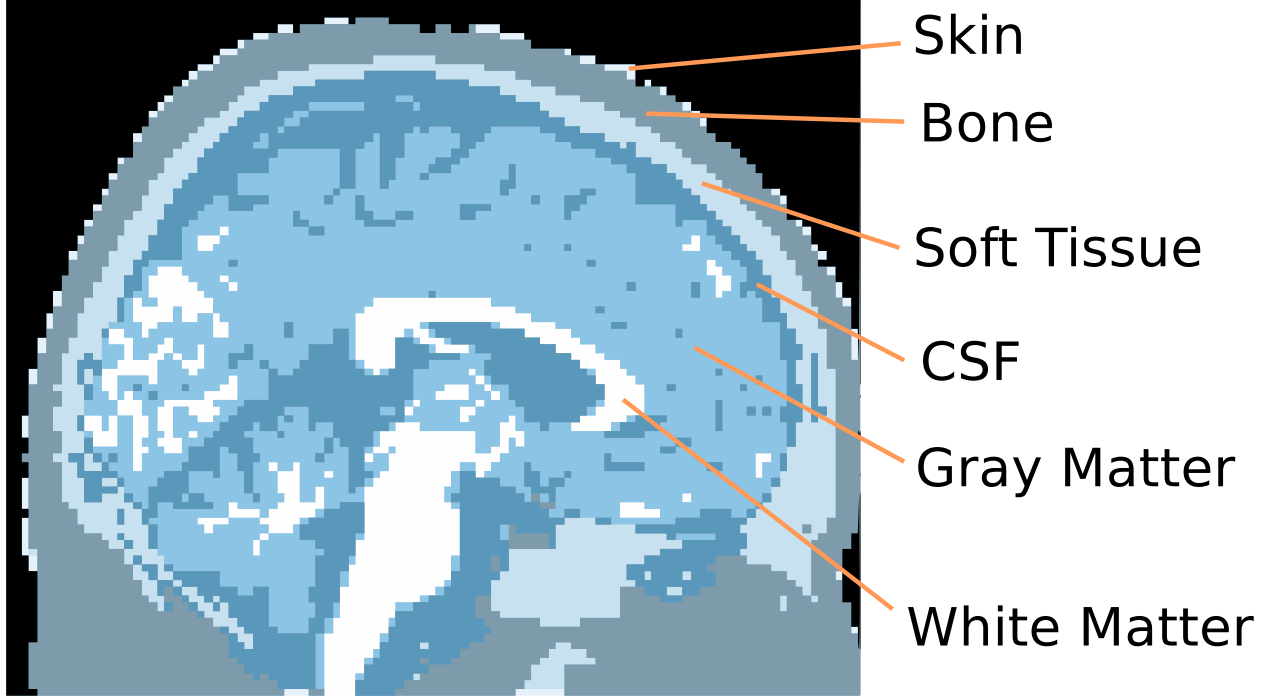
Figure 2.2   Slice of the segmented head. Each color represents a different tissue type.

separated in to six tissue types (Fig. 2.2). The advantage this has is that we are able to use tissue specific parameters when doing the calculations, thereby improving the accuracy of the results. The parameters used are available in Table 2.1. The code used to create the head matrix is discussed in appendix A.1.

Tissue	f_0 100 $ml/(g\ min)$	ρ kg/m^3	c $J\ kg^{-1}\,{}^\circ C^{-1}$	k $W\ m^{-1}\,{}^\circ C^{-1}$	Q_m W/m^3
Bone	3	1,080	2,110	0.65	26.1
Cerebrospinal Fluid	0	1,007	3,800	0.50	0
Gray Matter	67.1	1,035.5	3,680	0.565	15,575
White Matter	23.7	1,027.4	3,600	0.503	5,192
Muscle	3.8	1,041	3,720	0.4975	687
Skin	12	1,100	3,150	0.342	1,100

Table 2.1   Tissue-specific parameters used to calculate the temperature change (values from Collins et al. [2]).

2.2.2 Calculating the equilibrium temperature

The first step in calculating the temperature change is to know the resting state temperature for each voxel within the head. Our approach was to have the initial temperature for all tissue voxels set to 37°C and air voxels are kept at 24°C. The starting temperature of the tissue does not affect the final resting state temperature; however, starting off at drastically different values could greatly increase the calculating time required before the temperature stabilizes. The finite difference implementation of Penne's Bioheat Equation (Eq. (2.4)) is used to update the temperature (derivation provided in section 2.2.4). The temperature is updated until the temperature for every voxel has stabilized ($\frac{dT}{dt} < 10^{-6}$ °C/s). Since temperature changes due to changes in neuronal activity are typically greater than 10^{-2} °C, a change in temperature less than 10^{-6} °C/s is sufficiently small that transient temperature changes are negligible and temperature can be considered stabilized. The code used to calculate the equilibrium temperature is detailed in appendix A.3.

2.2.3 Calculating Metabolism and Blood Flow Changes from the BOLD response

This is the critical step where we use fMRI BOLD data to calculate the normalized change in metabolism and blood flow. The method used [1] is an assemblage of a couple other works [30, 31, 32, 33, 34, 35]. It starts by using the relation between metabolism and blood flow proposed by Buxton et al. [30]:

$$m(t) = f(t)\frac{E(t)}{E_0} \tag{2.5}$$

where E_0 is the oxygen extraction at rest and $E(f)$ is

$$E(f) = 1 - (1 - E_0)^{\frac{1}{f(t)}} \tag{2.6}$$

Table 2.2   Parameters used to solve the single-voxel implementation of Penne's Bioheat Equation. (modified from Sotero and Iturria-Medina [1])

Parameter	Meaning	Value
T_a	Arterial blood temperature	37°C
C_{tissue}	Tissue Heat Capacity	3.664 J/(gK)
ΔH°	Enthalpy released by oxidation of glucose	4.710^5 J
ΔH_b	Enthalpy used to release O_2 from hemoglobin	2.810^4 J
$CMRO_2 \mid_0$	Cerebral metabolic rate of O_2 consumption at rest	0.026310^{-6} mol/(gs)
$CBF\mid_0$	Cerebral blood flow at rest	0.0093 cm^3/(gs)
ρ_b	Blood density	1.05 g/cm^3
C_B	Blood heat capacity	3.894 J/(gK)
τ	Time constant for conductive heat loss from the ROI to the surrounding tissue	190.52 s
a, b, c	Parameters of the gamma function fitted from E(f) vs. f	0.4492, 0.2216, -0.9872
A	Maximum BOLD signal change	0.22
α	Steady state flow-volume relation	0.4
β	Field-strength dependent parameter	1.5

Variable	Meaning	
m(t)	$CMRO_2$ normalized to baseline	
f(t)	CBF normalized to baseline	
T(t)	Temperature	
W(t)	Lambert W Function	
$\frac{\Delta S(t)}{S_0}$	Change in BOLD signal normalized to rest	

in accordance with the oxygen limitation model [31]. Combining Eq. (2.5) with Eq. (2.6) yields

$$m(t) = \frac{f(t)}{E_0}\left[1 - (1 - E_0)^{\frac{1}{f(t)}}\right]$$ (2.7)

Sotero and Iturria-Medina [1] goes about solving Eq. (2.7) by adjusting $E(t)$ data generated by Eq. (2.6) and fitting it to the gamma function for the f range (0.7–2.0) that is within experimentally reported values [32, 33, 34]:

$$\frac{E(f)}{E_0} = af^c(t)e^{-bf(t)}$$ (2.8)

where a, b and c are fitting parameters (values provided in Table 2.2). From this approximation we have the final form of metabolism:

$$m(t) = af^{c+1}(t)e^{-bf(t)}.$$ (2.9)

As proposed by Davis et al. [35], the BOLD signal changes ($\frac{\Delta S(t)}{S_0}$) can be described in terms of $m(t)$ and $f(t)$:

$$\frac{\Delta S(t)}{S_0} = \frac{S(t) - S_0}{S_0} = A(1 - f^{\alpha-\beta}(t)m^\beta(t))$$ (2.10)

Substituting Eq. (2.9) into Eq. (2.10) yields

$$f(t)e^{-\frac{b\beta}{\alpha+\beta c}f(t)} = \left(\frac{\left(A - \frac{\Delta S(t)}{S_0}\right)}{Aa^\beta}\right)^{\frac{1}{\alpha+\beta c}}$$ (2.11)

where A is the maximum change in BOLD signal. Multiplying each side by $-\frac{b\beta}{\alpha+\beta c}$ gives

$$-\frac{b\beta}{\alpha+\beta c}f(t)e^{-\frac{b\beta}{\alpha+\beta c}f(t)} = -\frac{b\beta}{\alpha+\beta c}\left(\frac{\left(A - \frac{\Delta S(t)}{S_0}\right)}{Aa^\beta}\right)^{\frac{1}{\alpha+\beta c}}$$ (2.12)

which can be solved by using the Lambert W function

$$z = W(x) \tag{2.13}$$

where $W(x)$ is a solution for z in the equation

$$ze^z = x \tag{2.14}$$

Finally, $f(t)$ is obtained from Eq. (2.12)

$$f(t) = \frac{\alpha + \beta c}{b\beta} W(y(t)) \tag{2.15}$$

where

$$y(t) = -\frac{b\beta}{\alpha + \beta c} \left[\frac{(A - \frac{S(t)}{S_0} - 1)}{Aa^\beta} \right]^{\left(\frac{1}{\alpha + \beta c}\right)} \tag{2.16}$$

is a function of the BOLD signal. Using Eqs. (2.9), (2.15) and (2.16) allows for the metabolism and blood flow to be calculated from the BOLD signal (values used are provided in Table 2.2).

In order to process the files, the BOLD dataset is stored as a separate NIFTI (*.nii) file for each time step. The first step in processing the data for temperature calculations is to determine a resting state BOLD signal (S_0). The resting state is calculated by taking the voxel-wise mean of the data when the subject is at rest (i.e. the first and last 20 seconds). This results in one data set where each voxel is a mean of all of the voxels at the location over time (S_0). In order to calculate the metabolism and blood flow, the BOLD dataset needs to be normalized to this resting state ($\frac{\Delta S(t)}{S_0}$).

Once $\frac{\Delta S(t)}{S_0}$ is known for each time step, Eqs. (2.9), (2.15) and (2.16) can be used to calculate the metabolism and blood flow. The implementation of this procedure is discussed in appendix A.2.

2.2.4 Calculating the change in temperature in the active brain

As discussed in section 2.1.2, the foundation of the model is Penne's Bioheat Equation, Eq. (2.4). The metabolism and blood flow time-courses calculated in section 2.2.3 are used to scale the baseline heat production and blood profusion. This, in turn, induces a change in the temperature.

Penne's Bioheat Equation (Eq. (2.4)) is implemented using the first-order forward finite difference method:

$$\dot{T}(t) \approx \frac{T(t+l) - T(t)}{l} \tag{2.17}$$

where $T(t)$ is the temperature at time t and l is the time step size. Next, we can approximate $\nabla^2 T$ by using the second order central finite difference method applied for each coordiante:

$$
\begin{aligned}
\frac{\partial^2 T(t; x, y, z)}{\partial x^2} &\approx \frac{T(t; x+h, y, z) - 2T(t; x, y, z) + T(t; x-h, y, z)}{h^2} \\
\frac{\partial^2 T(t; x, y, z)}{\partial y^2} &\approx \frac{T(t; x, y+h, z) - 2T(t; x, y, z) + T(t; x, y-h, z)}{h^2} \\
\frac{\partial^2 T(t; x, y, z)}{\partial z^2} &\approx \frac{T(t; x, y, z+h) - 2T(t; x, y, z) + T(t; x, y, z-h)}{h^2}
\end{aligned}
\tag{2.18}
$$

where

$$\nabla^2 T = \frac{\partial^2 T(t; x, y, z)}{\partial x^2} + \frac{\partial^2 T(t; x, y, z)}{\partial y^2} + \frac{\partial^2 T(t; x, y, z)}{\partial z^2} \tag{2.19}$$

and the step size in each coordinate direction is given by h. In the case of processing BOLD data, h is the voxel size. Equation (2.19) can be subtitued into Penne's Bioheat Equation (Eq. (2.4))

$$\rho c \dot{T} = k_x \frac{\partial^2 T}{\partial x^2} + k_y \frac{\partial^2 T}{\partial y^2} + k_z \frac{\partial^2 T}{\partial z^2} - \rho_{blood} f(t) w c_{blood}(T - T_{blood}) + m(t) Q_m \tag{2.20}$$

where $\dot{T}$ is the first-derivative of temperature with respect to time, k_x is the thermal conductivity in the x-direction, k_y is the thermal conductivity in the y-direction and k_z is the thermal conductivity in the z-direction. Substituting Eqs. (2.17) and (2.18) into Eq. (2.20)

yields

$$\rho c \frac{T(t+l;x,y,z) - T(t;x,y,z)}{l} \approx \frac{1}{h^2}[k_x(T(t;x+h,y,z) - 2T(t;x,y,z) + T(t;x-h,y,z))$$
$$+ k_y(T(t;x,y+h,z) - 2T(t;x,y,z) + T(t;x,y-h,z))$$
$$+ k_z(T(t;x,y,z+h) - 2T(t;x,y,z) + T(t;x,y,z-h))]$$
$$- \rho_{blood} f(t) w c_{blood}(T - T_{blood}) + m(t)Q_m \qquad (2.21)$$

which is then rearranged to solve for $T(t+l;x,y,z)$

$$T(t+l;x,y,z) \approx T(t;x,y,z) + \frac{l}{\rho c h^2}[k_x(T(t;x+h,y,z) - 2T(t;x,y,z) + T(t;x-h,y,z))$$
$$+ k_y(T(t;x,y+h,z) - 2T(t;x,y,z) + T(t;x,y-h,z))$$
$$+ k_z(T(t;x,y,z+h) - 2T(t;x,y,z) + T(t;x,y,z-h))]$$
$$- \rho_{blood} f(t) w c_{blood}(T - T_{blood}) + m(t)Q_m \qquad (2.22)$$

The time step size (l) can be picked arbitrarily, however the spatial step size (h) is limited to the voxel spacing.

The final equation (Eq. (2.22)) gives a method for calculating the next time step ($T(t+l)$) from the current time step ($T(t)$) for each voxel. By using the central difference to solve $\nabla^2 T$, the voxels on all six sides of the current voxel are considered in the heat conduction. The implementation of this equation is discussed in appendix A.4.1.

2.3 Results

In order to understand the behavior of the model, we first applied it simulated BOLD data that was generated by convolving a boxcar function with the hemodynamic response function provided by SPM8. After we understood the behavior of the model, we then applied it to experimental BOLD data that was collected by Dhamala et al. [36].
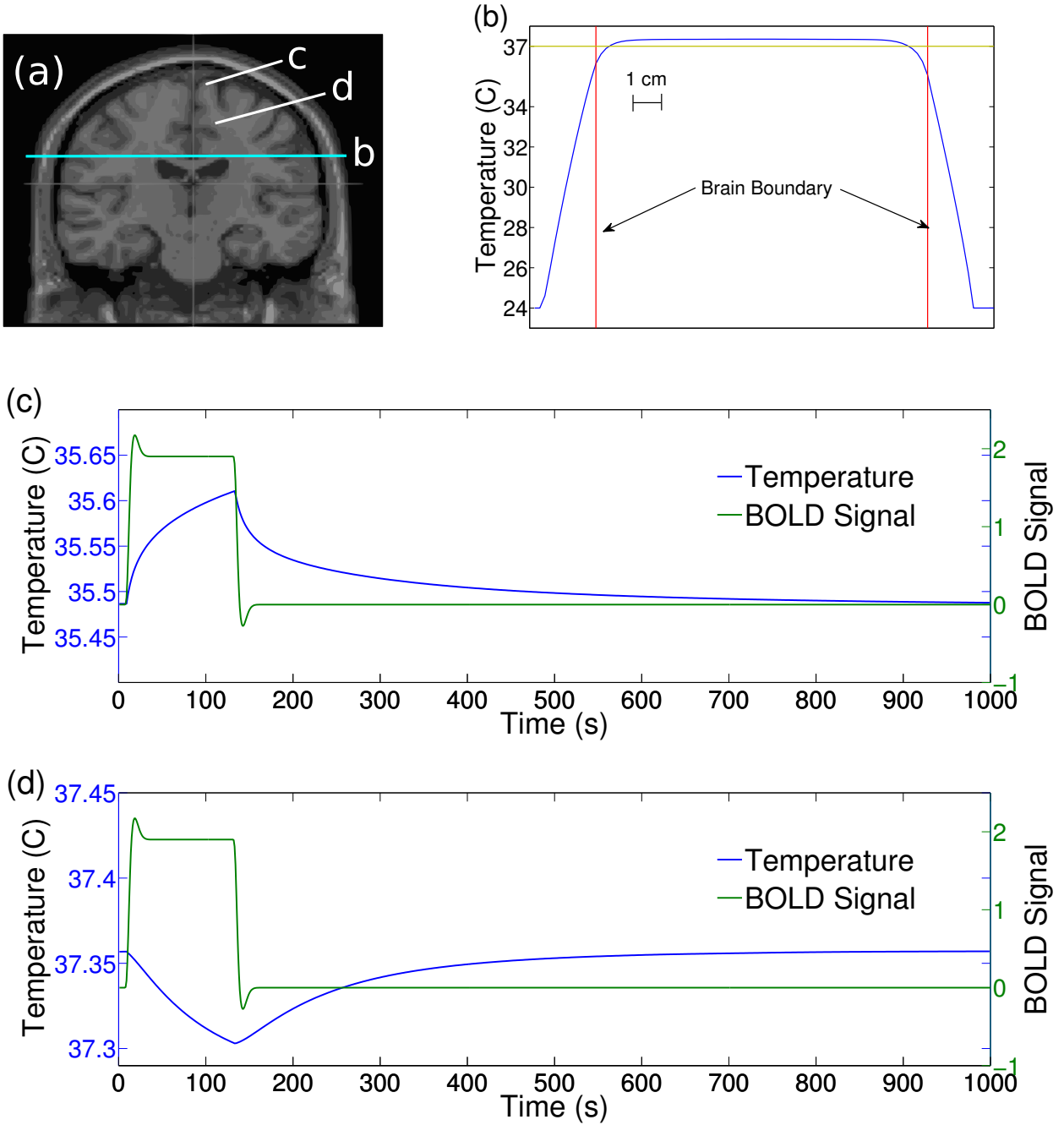
Figure 2.3   Temperature changes using simulated BOLD signals. (a) Slice of the head (y = -12) with indicators of the locations for parts (b)-(d). (b) Equilibrium temperature along a line through the head. Red lines indicate the brain boundary and the gold line indicates the blood temperature (37°C) used for calculations. Inside the brain, a 4-6 mm thick shell is created where the equilibrium temperature is less than the blood temperature. Within this shell, (c) the temperature rises with increased brain activity while (d) tissue deeper in the brain experiences the opposite effect.

2.3.1 Using Theoretical BOLD Data

To better understand the behavior of the tissue temperature model and the characteristics of temperature changes, BOLD activity was simulated in two ways: (i) from a nonlinear hemodynamic model [37] using stimulus or response function, or (ii) by convolving a boxcar function with the canonical hemodynamic response function provided by SPM 8 and corresponding temperature changes were calculated. Figure 2.3(c,d) shows a typical simulated BOLD response used (green curve) along with the change in temperature (blue) for two voxels at different locations in the brain (locations indicated by Fig. 2.3(a)).

Although both voxels have the same BOLD data, the demonstrate contrasting changes in temperature. This can be best understood by considering the equilibrium temperature of each voxel. Figure 2.3(b) is a plot of the equilibrium temperature (blue line) along a line passing through the head (path indicated by the teal line in part (a) of the same figure). The vertical red lines indicate the boundary between the brain and surrounding tissues and the horizontal yellow line is an indication of the blood temperature (37°C). Two regions exist within the brain that lead to contrasting temperature behaviors.

The majority of the brain tissue is at a resting-state temperature that is less than the blood temperature (region 1). For voxels within this region, a behavior like that shown in Fig. 2.3(d) is to be expected. The primary contribution to an increase in the BOLD response is an increase in local blood flow. Since the blood temperature is cooler than the tissue temperature, it removes heat from the tissue thereby lowering the temperature. Single-voxel models are able to account for this result because their assumptions about the location of a voxel are consistent with being located within this region.

The second region is comprised of a thin (4–6 mm) layer of brain tissue that is closest to the surface of the head. As a result of its proximity to the surface of the head, conductive heat lost to the air puts the resting-state temperature of voxels in this region below the arterial blood temperature. As a result, when there is an increase in blood flow (increase in BOLD), the warmer blood will increase the voxel temperature (Fig. 2.3(c)). Since single-

17

voxel models approximate voxel conditions, they are unable to account for this region of tissue.

Conduction is a slow process, so over shorter time scales (less than 10 minutes), conduction will contribute very little to the temperature change from a change in brain activity. However, conduction plays an important role in determining the resting-state temperature.

The primary advantage with this model is that it accounts for the location of the voxel when determining the temperature, thus the direction of the temperature change depends on how far away from the surface of the head the voxel is. For voxels within a 4–6 mm shell near the surface of the brain, the temperature increases with increased activity (Fig. 2.3(c)) while voxels deeper within the brain experience the opposite change (Fig. 2.3(d)).

2.3.2 Using Experimental BOLD Data

Data from a previous fMRI study [36] was used to study the characteristics of temperature changes in a typical experiment. All participants in this experiment were right handed and between the ages of 23 and 27 years old. Signed informed consent was collected from each one prior to participating in the study. Institutional Review Boards of Emory University and Georgia State University approved this experiment. Twelve participants were asked to tap their right index fingers with rhythms of varying complexity for 320s.

This task resulted in a strong BOLD response in the motor cortex (Fig. 2.4). The experiment included 20s of rest at the beginning and end of the tapping periods. Here, the resting state response level is calculated for each voxel by averaging across 40s of resting-state fMRI data. Using equations Eqs. (2.9), (2.15) and (2.16), the time-dependent change in blood flow and metabolism can be determined for each voxel. Finally, these values are used in conjunction with Eq. (2.4) to find the change in temperature throughout the brain. In this task, a temperature increase of approximately $0.02°C$ was observed in the motor cortex (Fig. 2.4). This value is well within the range of temperature changes observed in experimental measurements [10, 11, 12, 13, 14].

The increase rather than decrease in temperature in the motor cortex during a functional

(a)

(b)
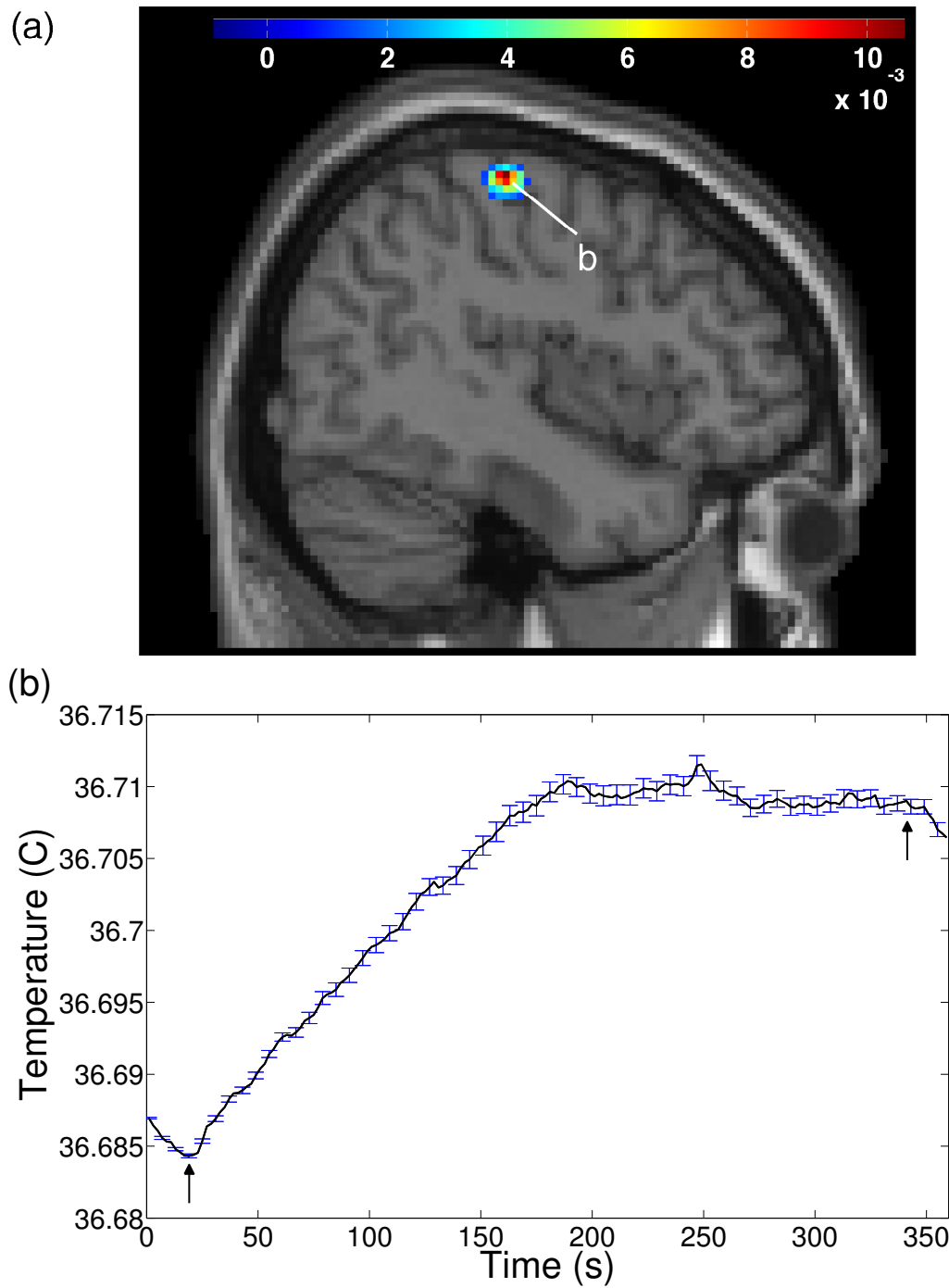
Figure 2.4   Temperature calculated from a voxel within the motor cortex. (a) A slice (x = -44) showing the motor cortex warming during a finger-tapping task. (b) Temperature at a voxel within the motor cortex (-44, -24, 60) with standard error indicated by blue error bars (Arrows indicate task onset and conclusion, N=24).

activity is consistent with the idea that the temperature of the blood in the capillaries is slightly greater than the baseline tissue temperature in superficial cortical regions; however, single-voxel models would predict the opposite effect.

3   Optical techniques for brain activity measurements

The best method for measuring brain temperature is to use a thermocouple probe placed in direct contact with the tissue. Experimental measurements of brain temperature have achieved a precision of small as $0.0003°C$ using this method [10]. However, this method can not be used in humans without damaging the tissue. An optical method would be ideal for non-invasive and non-damaging measurements. Presently, there does not exist a method for accurately measuring the temperature of brain tissues optically. However, other optical measurements methods could be used in conjunction with a temperature model (such as the one proposed here) to calculate the temperature. The possible application of functional Near-Infrared Spectroscopy (fNIRS) and its possible use in brain temperature calculations is discussed along with the limitations of a direct measurement technique such as thermal imaging.

3.1 Functional Near-Infrared fNIR Spectroscopy

As discussed in chapter 1, changes in tissue activity can be detected by measuring the change in blood oxygenation levels. Functional Magnetic Resonance Imaging (fMRI) is one technique among several for accomplishing this (the BOLD signal).

Blood oxygenation can be determined by measuring the relative concentrations of oxyhe-

	fMRI	fNIR
Spatial Resolution	8–27 mm^3	~ 1–10 cm^3
Temporal Resolution	1–2 s	$\sim 10^{-3}$ s
Measurement Parameter	blood volume, flow, and O_2 metabolism	oxyHb and deoxyHb concentrations
Motion	Must Remain Stationary	Small movements OK
Penetration	Whole-head	outer 2–4 mm of brain tissue

Table 3.1   Comparison of the capabilities and limitations of fMRI and fNIR techniques. Compiled from Bunce et al. [40], Elliott [41].
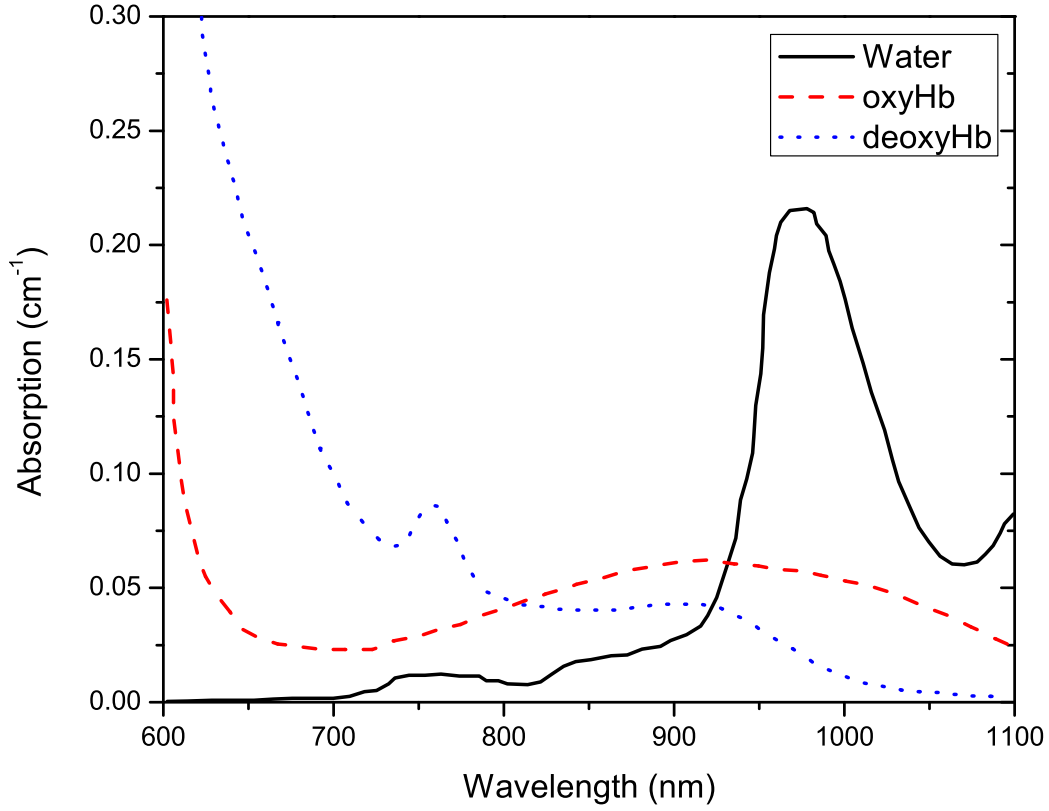
Figure 3.1   Absorption spectra of water Cope [38], oxyHb and deoxyHb Horecker [39].

moglobin (oxyHb) and deoxyhemoglobin (deoxyHb) [6, 7, 8]. Since oxyHb and deoxyHb have different absorption spectra as shown in Fig. 3.1. These differences are possible to detect through optical techniques. Functional Near-Infrared Spectroscopy is a technique which utilizes two or more spectral bands in order to determine blood oxygenation. fNIRS has a high (millisecond) temporal resolution and a low ($\sim$1 cm^3) spatial resolution compared to fMRI (as low as 1 mm^3). Also, fNIRS is limited to only imaging the outer cortex (2–4 mm) [40]. A comparison of fMRI and fNIR is presented in Table 3.1.

fNIRS works by utilizing an array of near-infrared detectors and emitters (typically spaced 2–3 cm apart) placed in contact with the skin [42, 43]. A schematic of a typical fNIRS array setup is shown in Fig. 3.2. Each dashed line is a detection path. By illuminating the emitters
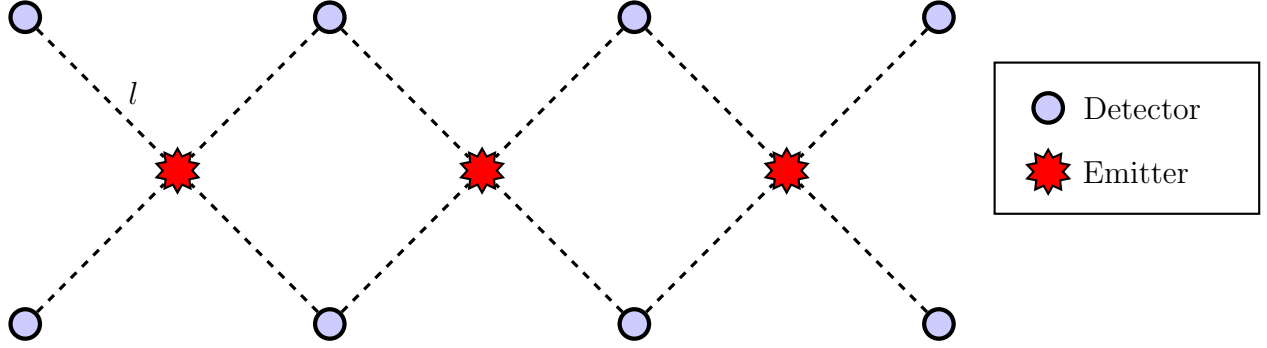
Figure 3.2   Arrangement of detectors and emitters in a typical fNIRS setup. Light coming from emitters (stars) is detected by the detectors (circles) set at a distance l away.

sequentially, it is possible to have 10 detection channels using the setup shown. The exact spacing between the emitters and detectors determines the depth the light is detected from. As shown in Fig. 3.3, the closer the spacing, the higher the resolution but at the expense of lower penetration. Conversely, in order to detect light passing through deeper tissue, a wider spacing is used which reduces the resolution. The exact wavelengths used vary, but all lie within an optical window between 700–1000 nm [42] where the near-infrared photon absorption in the tissue is low (Fig. 3.1).

Three techniques are used to illuminate the tissue: (i) time domain (or time resolved spectroscopy, TRS), (ii) frequency domain and, (iii) continuous wave illumination [43]. In TRS, short pulses of light are incident on the tissue and the temporal distribution of photons in measured. In frequency domain spectroscopy, the amplitude of the indecent light is modulated at a high frequency (10–100 MHz) and the phase shift and amplitude decay of the detected light is compared to the incident light [45]. In continuous wave illumination, the incident light is not modulated so the detected light can only be compared for amplitude attenuation [43].

All of the techniques use the Beer-Lambert Law [46]

$$I = I_0 e^{-\alpha(\lambda)x} \tag{3.1}$$
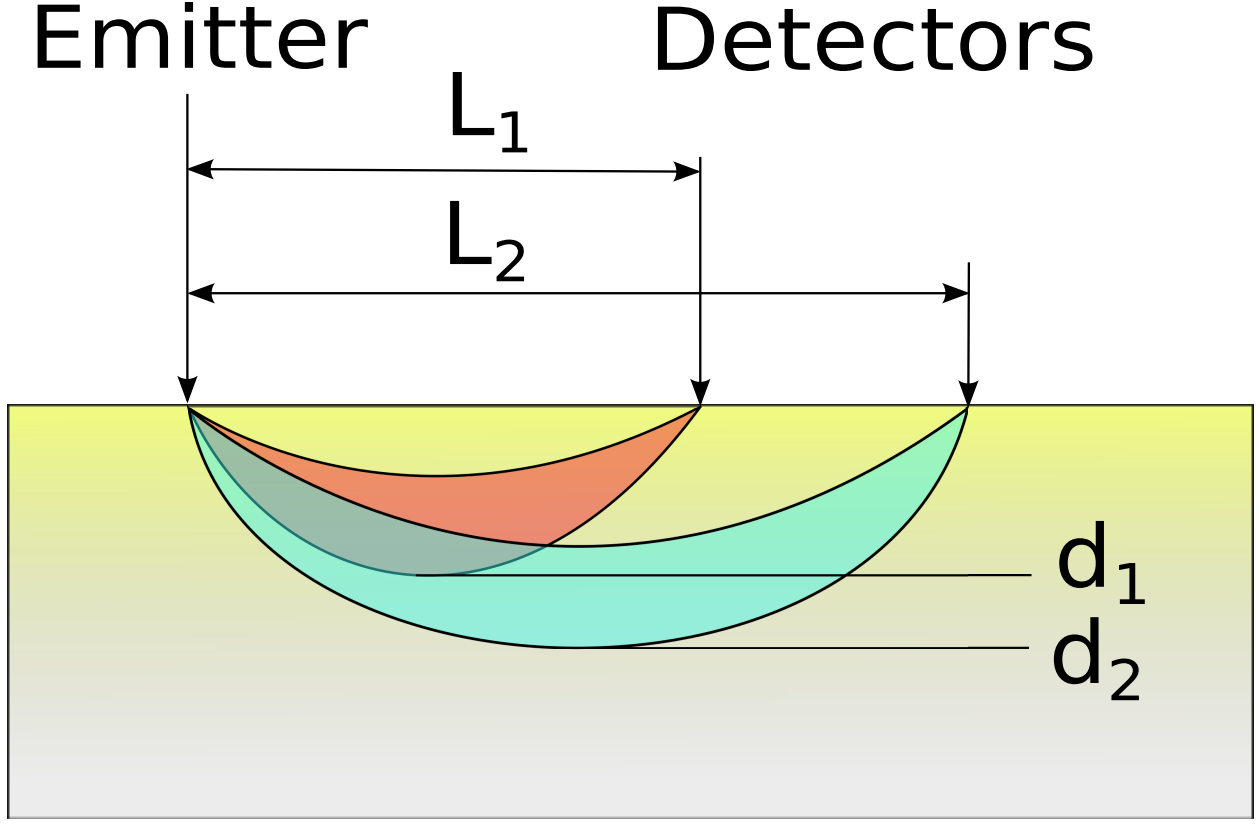
Figure 3.3   Penetration depth of a fNIR detector as a function of the distance between the NIR emitter and detector. Light emitted from the left passes through the tissue before it is detected. The path it takes through the tissue is determined by the distance between the emitter and detector. A larger separation (L_2) allows the light to penetrate deeper tissues (d_2). Modified after [44]

modified to isolate the individual contributions from oxyHb and deoxyHb [38]:

$$I = GI_0 e^{-(\alpha_{deoxyHb} C_{deoxyHb} + \alpha_{oxyHb} C_{oxyHb})l} \tag{3.2}$$

where G is a factor to adjust for the measurement geometry, I_0 is in the incident light intensity, α_{oxyHb} and $\alpha_{deoxyHb}$ are the molar extinction coefficients for oxyHb and deoxyHb, C_{oxyHb} and $C_{deoxyHb}$ are the chromophore concentrations for oxyHb and deoxyHb, and l is the path length [43]. By comparing a baseline measurement (I_b) with a new measurement (I),

the optical density can be determined [43]

$$\Delta OD = \log \frac{I_b}{I} = \alpha_{deoxyHb}\Delta C_{deoxyHb} + \alpha_{oxyHb}\Delta C_{oxyHb} \qquad (3.3)$$

As discussed in Izzetoglu et al. [43], at least two wavelengths are utilized in the spectral window (700–1000 nm) in order to determine the change in concentration of chromophores $\Delta C_{deoxyHb}$ and ΔC_{oxyHb}. With these values, the oxygenation and total blood volume can be determined:

$$Oxygenation \ = \Delta C_{HBO2} - \Delta C_{HB}$$
$$Blood\ Volume \ = \Delta C_{HBO2} + \Delta C_{HB} \qquad (3.4)$$

Using this method to experimentally measure the blood oxygenation while measuring the fMRI BOLD response could be used to refine the present model for calculating the metabolism and blood flow from the BOLD response.

While fNIRS does not provide spatially-precise measurements as fMRI, it should be possible to modify the existing model for calculating temperature from the BOLD response to use fNIR data. This would be advantageous because fNIRS systems are cheaper and less disruptive than fMRI systems, meaning they can be used with a wider range of patients (children and the elderly). For this reason, developing a model which uses fNIRS data should be considered in future research.

3.2 Thermal Imaging

The primary challenge in brain temperature research is preforming brain temperature experimental measurements. Since it is non-invasive, thermal imaging is appealing as a possible replacement for damaging thermocouple probes. Unfortunately, this technique is limited by the high absorption of mid-infrared photons by water.

Light absorption by a material is modeled using the Beer-Lambert law

$$I = I_0 e^{-\alpha(\lambda)x} \tag{3.5}$$

where I is the intensity at a depth x remaining from light with an incident intensity I_0 in a material with absorption coefficient α. The point at which the intensity has decayed to $1/e$ (about 37%) of the incident intensity is called the penetration depth, δ_p

$$\delta_p = \frac{1}{\alpha(\lambda)} \tag{3.6}$$

This equation can be used along with the black-body spectrum at tissue temperatures (Fig. 3.4) we can estimate the penetration depth of mid-infrared photons passing through water.

Wien's Displacement Law is a solution to Planck's law for the peak light emission wavelength:

$$\lambda_{max} = \frac{b}{T} \tag{3.7}$$
$$b = 2.897\,7721 * 10^{-3} \text{ K m}$$

where b is Wien's displacement constant and T is the temperature in kelvin. For $T = 310$ K ($T = 37°$ C), Wien's law yields a peak black-body emission wavelength of $9.347\,652$ µm. A physiologically reasonable temperature change to expect from stimulation is on the order of $0.01°$C which corresponds to a new peak wavelength of $9.347\,350$ µm (T=310.01 K) or a shift of 0.302 nm.

The values of the absorption coefficient and the penetration depth of photons in water is shown in Fig. 3.5. Looking at around 9.3 µm, the absorption coefficient is approximately 700 cm^{-1} which corresponds to a penetration depth of approximately 0.14 µm. This depth is roughly five orders of magnitude smaller than the distance from the surface of the brain to
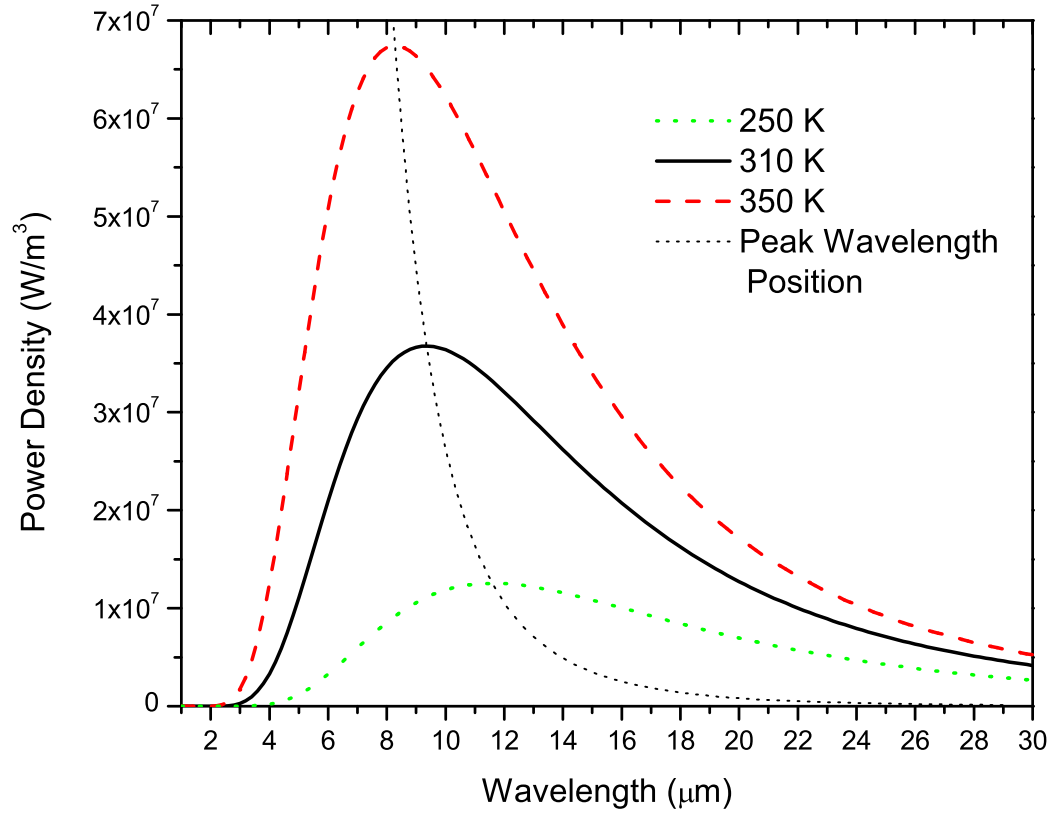
Figure 3.4 Black-body spectrum at 250, 310 and 350 K calculated using Planck's Law. The black dashed line traces the peak in the spectrum as temperature changes. As the temperature increases, differences in temperature translate to smaller shifts in peak wavelength compared to cooler temperatures.

the surface of the head. Further, all photons emitted as blackbody radiation (ranging from 3 μm to over 30 μm) have a penetration depth of less than 100 microns. Thus, a thermal imaging camera is unable to image photons coming from the brain.

When thermal imaging is used, the photons collected come from the skin of the head rather than from any deeper tissues, thus it is not a viable form of brain activity detection unless direct line of sight to the brain is available (such as in an open skull surgery). The noise-equivalent temperature difference (NETD) of currently available cameras is greater than 14 mK [48, 49] so it would be limited to only the most extreme of excitations even if
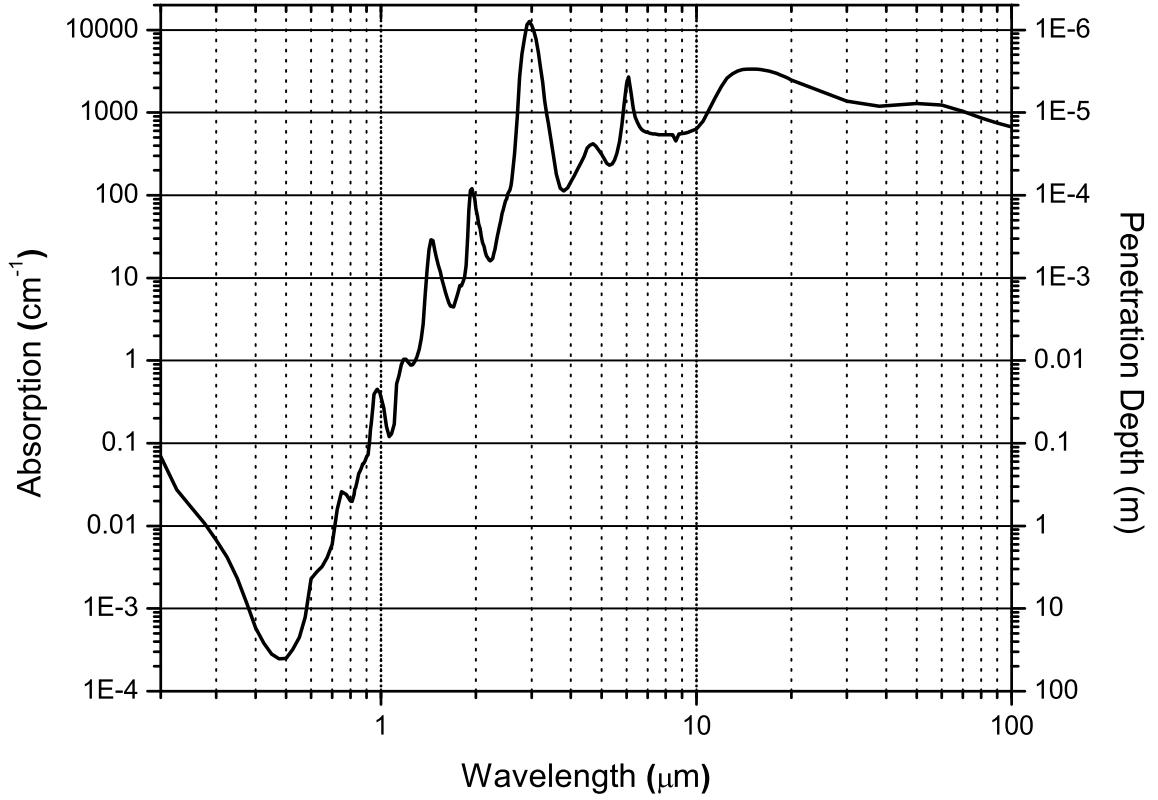
Figure 3.5 The absorptions spectra of water from UV to far-infrared. Modified from Hale and Querry [47].

line of site to the brain is available. As a comparison, the finger-tapping task discussed in the results section (section 2.3.2) only induced a peak temperature change of 25 mK after tapping for about 170 seconds. Detection of this activity would be at the limits of a thermal imaging camera.

While its applications to detecting brain activity are limited, thermal cameras could be useful in the operating room. It has been found that inducing mild hypothermia in patients being treated for cerebral ischemia improves the clinical outcome [19]. The same treatment has been shown to improve the outcome of patients who have experienced a stroke [21] and even in patients with severe head injuries [18]. The temperature of the brain is currently inferred from the core body temperature (which is monitored via an invasive

28

thermistor catheter). If it is possible to directly image the brain (i.e. during surgery) then the hypothermia treatment can be better monitored through a thermal imaging camera. This would be especially useful since conductive and radiative heat loss to the air from an exposed brain could reduce how tightly the brain temperature is regulated by the arterial blood temperature.

Optical detectors face many challenges working with biological tissue, the worst being infrared light absorption by water. fNIRS works within an optical window in the water absorption in order to measure changes in blood oxygenation, while thermal imaging is limited to measuring the temperature of tissue it has direct line of sight with because of the high absorption of water in the operating window. Despite their limitations, both of these techniques could be used in future studies to improve our understanding of brain temperature dynamics.

4 Conclusion

It has been shown that by considering the entire head within the model, brain temperature can be reliably calculated from non-invasive fMRI measurements. Experimental measurements of activity-induced brain temperature changes have shown that a simple relationship does not exist [10, 11, 12, 13, 14]. Single-voxel brain temperature modeling efforts predict that an increase in brain activity will induce a decrease in temperature. This one-dimensional perspective does not account for the spatial distribution of heat throughout the head like a multi-voxel approach.

Our model of brain temperature changes is able to account for the variability found in experimental brain temperature measurements. This is accomplished by modeling heat dynamics throughout the entire head rather than reducing the model to one ROI. It was found that the variability in experiment measurements is most likely due to differences in resting state temperatures throughout the brain. Since each voxel is at a slightly different temperature, the same change in the BOLD response may result in different changes in temperature. Additionally, it was found through the model that a thin (4–6 mm) region of outer cortical tissue is at a resting temperature below the blood temperature. In this region, an increase in brain activity (inducing an increase in CBF) will warm the tissue. Thus, with the same BOLD response, tissue may either be warmed or cooled depending on it's proximity to the surface of the head.

The biggest shortcoming of our model is that we are unable to independently compare our calculations with experimental measurements of temperature and BOLD response. It was not possible for us to do this because there is currently not a method for non-invasively measuring temperature independent of an fMRI. An improvement to the model could also be gained by a more accurate method of $CMRO_2$ and CBF calculations from the BOLD response. The current method uses empirically fit formulas, so the accuracy is limited by the

data used for the fitting. An *ab initio* model that does not rely on experimental data would be ideal. The calculations could also be improved by segmenting the head into more tissue types. We used six tissue types, but the use of more would further improve the calculations since each tissue type has different physiological parameters (thermal conduction, baseline heat production, etc.). A separate line of research that could be pursued would be a model for calculating brain temperature changes from fNIRS data. Both fNIRS and fMRI BOLD response detect changes in local tissue oxygenation, so it should be possible to adapt our model to use fNIRS data. If such a model existed, calculations from it and our model could be compared to refine both models.

Although it is expected that the contribution would be negligible [50], our model does not take into account the affects of perspiration. It would likely not affect the change in temperature greatly because it takes place a couple of centimeters away from brain tissue. Another physiological affect not account for is temperature regulation by the pre-optic nucleus of the anterior hypothalamus [17]. It is responsible for balancing heat production and dissipation [51] and if the model were applied to cases where extreme brain temperatures are created then it would be important to account for how this would react.

Understanding how human brain temperature is affected by changes in local brain activity is not well understood because the changes are small and current experimental measurement techniques are damaging. Models such as the one proposed here allow for brain temperature to be understood through non-invasive measurements such as the fMRI BOLD response.

References

[1] Roberto C Sotero and Yasser Iturria-Medina. From Blood Oxygen Level Dependent (BOLD) signals to brain temperature maps. *Bulletin of Mathematical Biology*, 73(11): 2731–2747, 2011.

[2] Christopher M. Collins, Michael B. Smith, and Robert Turner. Model of local temperature changes in brain upon functional activation. *Journal of Applied Physics*, 97: 2051–2055, 2004.

[3] Roberto C Sotero and Nelson J Trujillo-Barreto. Modeling the role of excitatory and inhibitory neuronal activity in the generation of the BOLD signal. *NeuroImage*, 35: 149–165, 2007.

[4] Herman Y Carr and E M Purcell. Effects of diffusion on free precession in nuclear magnetic resonance experiments. *Physical Review*, 94:630–638, 1954.

[5] Paul Lauterbur. Image formation by induced local interactions: Examples employing nuclear magnetic resonance. *Nature*, 242:190–191, 1973.

[6] Seiji Ogawa, David W. Tank, Ravi Menon, Jutta M . Ellermann, Seong-Gi Kim, Hellmut Merkle, and Kamil Ugurbil. Intrinsic signal changes accompanying sensory stimulation: Functional brain mapping with magnetic resonance imaging. *Proceedings of the National Academy of Science*, 89:5951–5955, 1992.

[7] Kenneth K. Kwong, John W. Belliveau, David A. Chesler, Inna E. Goldberg, Robert M. Weisskoff, Brigitte P. Poncelet, David N. Kennedy, Bernice E. Hoppel, Mark S. Cohen, Robert Turner, Hong-Mind Cheng, Thomas J. Brady, and Bruce R. Rosen. Dynamic magnetic resonance imaging of human brain activity during primary sensory stimulation. *Proceedings of the National Academy of Science*, 89:5675–5679, 1992.

[8] Peter T. Fox and Marcus E. Raichle. Focal physiological uncoupling of cerebral blood flow and oxidative metabolism during somatosensory stimulation in human subjects. *Proceedings of the National Academy of Science*, 83:1140–1144, 1986.

[9] Jan Karbowski. Thermodynamic constraints on neural dimensions, firing rates, brain temperature and size. *Journal of Computational Neuroscience*, 27:415–436, 2009.

[10] J. G. McElligott and R Melzack. Localized thermal changes evoked in the brain by visual and auditory stimulation. *Experimental Neurology*, 17:293–312, 1967.

[11] Eugene A. Kiyatkin, P. Leon Brown, and Roy A. Wise. Brain temperature fluctuation: a reflection of functional neural activity. *European Journal of Neuroscience*, 16:164–168, 2002.

[12] G. Zeschke and V. G. Krasilnikov. Decreases of local brain temperature due to convection (local brain blood flow) and increases of local brain temperature due to activity. *Acta Biologica et Medica Germanica*, 35:935–941, 1976.

[13] J. S. George, J. D. Lewine, A. S. Goggin, R. B. Dyer, and E. R. Flynn. IR thermal imaging of a monkey's head: local temperature changes in response to somatosensory stimulation. *Optical Imaging of Brain Function and Metabolism*, 333:125–136, 1993.

[14] Shunro Tachibana. Local temperature, blood flow, and electrical activity correlations in the posterior hypothalamus of the cat. *Experimenal Neurology*, 16:148–161, 1966.

[15] Dmitriy A. Yablonskiy, Joseph J. H. Ackerman, and Marcus E. Raichle. Coupling between changes in human brain temperature and oxidative metabolism during prolonged visual stimulation. *Proceedings of the National Academy of Science*, 97(13):7603–7608, 2000.

[16] Hubert K. F. Trübel, Laura I. Sacolick, and Fahmeed Hyder. Regional temperature

changes in the brain during somatosensory stimulation. *Journal of Cerebral Blood Flow & Metabolism*, 26:68–78, 2006.

[17] Gianluca Bertolizio, Linda Mason, and Bruno Bissonnette. Brain temperature: heat production, elimination and clinical relevance. *Pediatric Anesthesia*, 21(4):347–358, 2011.

[18] Jens Soukup, Alois Zauner, Egon M.R. Doppenberg, Matthias Menzel, Charlotte Gilman, Harold F. Young, and Ross Bullock. The importance of brain temperature in patients after severe head injury: Relationship to intracranial pressure, cerebral perfusion pressure, cerebral blood flow, and outcome. *Journal of Neurotrauma*, 19(5): 559–571, 2002.

[19] J Maher and V Hachinski. Hypothermia as a potential treatment for cerebral ischemia. *Cerebrovascular and brain metabolism reviews*, 5(4):277–300, 1993.

[20] M D Ginsberg, L L Sternau, M Y Globus, W D Dietrich, and R Busto. Therapeutic modulation of brain temperature: relevance to ischemic brain injury. *Cerebrovascular and brain metabolism reviews*, 4(3):189–225, 1992.

[21] Derk W. Krieger, Michael A. De Georgia, Alex Abou-Chebl, John C. Andrefsky, Cathy A. Sila, Irene L. Katzan, Marc R. Mayberg, and Anthony J. Furlan. Cooling for acute ischemic brain damage (cool aid). *Stroke*, 32(8):1847–1854, 2001.

[22] Cause and prevention of central nervous system injury in neonates undergoing cardiac surgery. *Seminars in Thoracic and Cardiovascular Surgery*, 19(3):269 – 277, 2007.

[23] Eiichi Suehiro, Yuji Ueda, Enoch P. Wei, Hermes A. Kontos, and John T. Povlishock. Posttraumatic hypothermia followed by slow rewarming protects the cerebral microcirculation. *Journal of Neurotrauma*, 20(4):381–390, 2003.

[24] M A Baker, R A Stocking, and J P Meehan. Thermal relationship between tympanic membrane and hypothalamus in conscious cat and monkey. *Journal of applied physiology*, 32(6):739–742, 1972.

[25] K. Shiraki, S. Sagawa, F. Tajima, A. Yokota, M. Hashimoto, and G. L. Brengelmann. Independence of brain and tympanic temperatures in an unanesthetized human. *Journal of Applied Physiology*, 65(1):482–486, 1988.

[26] Mark M Stecker, Albert T Cheung, Alberto Pochettino, Glenn P Kent, Terry Patterson, Stuart J Weiss, and Joseph E Bavaria. Deep hypothermic circulatory arrest: I. effects of cooling on electroencephalogram and evoked potentials. *The Annals of Thoracic Surgery*, 71(1):14 – 21, 2001.

[27] Christopher S Rumana, Shankar P Gopinath, Masahiko Uzura, Alex B Valadka, and Claudia S Robertson. Brain temperature exceeds systemic temperature in head-injured patients. *Critical Care Medicine*, 26(3), 1998.

[28] Bruno Bissonnette, Helen M Holtby, Annette J Davis, Hweeleng Pua, Fay J Gilder, and Michael Black. Cerebral Hyperthermia in Children after Cardiopulmonary Bypass. *Anesthesiology*, 93(3), 2000.

[29] H H Pennes. Analysis of tissue and arterial blood temperatures in the resting human forearm. *Journal of Applied Physiology*, 1(2):93–122, 1948.

[30] Richard B. Buxton, Kâmil Uludağ, David J. Dubowitz, and Thomas T. Liu. Modeling the hemodynamic response to brain activation. *NeuroImage*, 23, Supplement 1(0):S220–S233, 2004.

[31] Richard B. Buxton, Eric C. Wong, and Lawrence R. Frank. Dynamics of blood flow and oxygen changes during brain activation: The Balloon Model. *Magnetic Resonance in Medicine*, 39:855–864, 1998.

[32] PT Fox, ME Raichle, MA Mintun, and C Dence. Nonoxidative glucose consumption during focal physiologic neural activity. *Science*, 241(4864):462–464, 1988.

[33] Christoph Leithner, Georg Royl, Nikolas Offenhauser, Martina Fuchtemeier, Matthias Kohl-Bareis, Arno Villringer, Ulrich Dirnagl, and Ute Lindauer. Pharmacological uncoupling of activation induced increases in CBF and $CMRO_2$. *Journal of Cerebral Blood Flow & Metabolism*, 30(2):311–322, Sep 2009.

[34] Ai-Ling Lin, Peter T. Fox, Jean Hardies, Timothy Q. Duong, and Jia-Hong Gao. Nonlinear coupling between cerebral blood flow, oxygen consumption, and ATP production in human visual cortex. *Proceedings of the National Academy of Sciences*, 107(18): 8446–8451, 2010.

[35] Timothy L. Davis, Kenneth K. Kwong, Robert M. Weisskoff, and Bruce R. Rosen. Calibrated functional MRI: Mapping the dynamics of oxidative metabolism. *Proceedings of the National Academy of Sciences*, 95(4):1834–1839, 1998.

[36] Mukeshwar Dhamala, Giuseppe Pagnoni, Kurt Wiesenfeld, Caroline Zink, Megan Martin, and Gregory Berns. Neural correlates of the complexity of rhythmic finger tapping. *NeuroImage*, 20:918–926, 2003.

[37] K. J. Friston, A. Mechelli, R. Turner, and C. J. Price. Nonlinear response in fMRI: The Balloon Model, Volterra Kernels, and other hemodynamics. *NeuroImage*, 12:466–477, 2000.

[38] M Cope. *The development of a near infrared spectroscopy system and its application for non invasive monitoring of cerebral blood and tissue oxygenation in the newborn infants*. PhD thesis, London University, 1991.

[39] B L Horecker. The absorption spectra of hemoglobin and its derivatives in the visible and near infra-red regions. *The Journal of Biological Chemistry*, 1942.

[40] S.C. Bunce, M. Izzetoglu, K. Izzetoglu, B. Onaral, and K. Pourrezaei. Functional near-infrared spectroscopy. *Engineering in Medicine and Biology Magazine, IEEE*, 25(4):54 –62, July–August 2006. ISSN 0739-5175. doi: 10.1109/MEMB.2006.1657788.

[41] Mark Elliott. Functional imaging with diffuse optical tomography. URL `http://www.mmrrcc.upenn.edu/mediawiki/images/3/30/FNIR.ppt`.

[42] Arno Villringer and Britton Chance. Non-invasive optical spectroscopy and imaging of human brain function. *Trends in Neurosciences*, 20(10):435–442, 1997. ISSN 01662236. doi: 10.1016/S0166-2236(97)01132-6.

[43] Kurtulus Izzetoglu, Scott Bunce, Banu Onaral, and Kambiz Pourrezaei. Functional optical brain imaging using near-infrared during cognitive tasks. *International Journal of Human-Computer Interaction*, 17(2):211–227, 2004.

[44] L.M. Head and R.M. Pierson. Functional near infrared spectroscopy for hb and hbo2 detection using remote sensing. In *Sensors Applications Symposium (SAS), 2010 IEEE*, pages 166 –169, Feb. 2010. doi: 10.1109/SAS.2010.5439393.

[45] David A. Boas, Maria Angela Franceschini, Andy K. Dunn, and Gary Strangman. Noninvasive imaging of cerebral activation with diffuse optical tomography. In Ron D. Frostig, editor, *In-vivo optical imaging of brain function*, pages 193–221. 2002.

[46] Beer. Bestimmung der absorption des rothen lichts in farbigen flssigkeiten (determination of the absorption of red light in colored liquids). In Ludwig Wilhelm Gilbert, editor, *Annalen der Physik und Chemie*, volume 86, pages 78–88. 1852.

[47] George M. Hale and Marvin R. Querry. Optical constants of water in the 200-nm to 200-µm wavelength region. *Appl. Opt.*, 12(3):555–563, Mar 1973. doi: 10.1364/AO.12.000555.

[48] FLIR Advanced Thermal Solutions: FLIR GF335 Broadband Infrared Cameras for Research & Science, February 2012. URL `http://www.flir.com/thermography/americas/us/view/?id=46792&collectionid=519&col=4586`.

[49] Mirage Infrared Camera - Infrared Cameras Inc, February 2012. URL `http://www.infraredcamerasinc.com/infrared-camera-Mirage.html`.

[50] E. R. Nadel, J. W. Mitchell, B. Saltin, and J. A. Stolwijk. Peripheral modifications to the central drive for sweating. *Journal of Applied Physiology*, 31(6):828–833, 1971. URL `http://jap.physiology.org/content/31/6/828.short`.

[51] H B Simon. Hyperthermia. *The New England journal of medicine*, 329(7):483–487, 1993.

Appendix A Code

The following sections include the code used. It was written for Matlab R2011b and requires SPM8 to run. Additionally, it is recommended that you have at least 4 GB of RAM in order to work with the large datasets that are produced. For information about how to visualize the data produced, see appendix B. All of the code is available through the temptools github page (`https://github.com/greggroth/temptools`). Additionally, many of the tasks can be completed using the temptools gui (Figs. (A.1) - (A.4)) which can be invoked by running

```
temptools
```

at the Matlab command prompt (make sure the temptools directory and subdirectories have been added to the Matlab path). The procedure used is explained in section 2.2 and a graphical representation is available in Fig. 2.1.
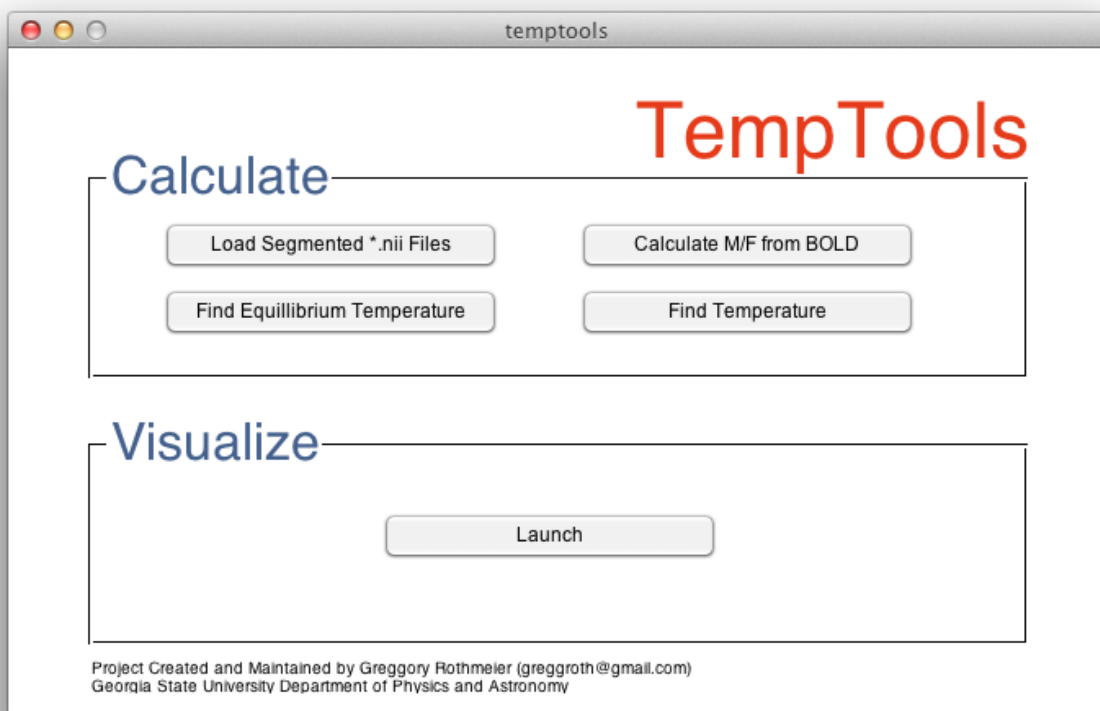
Figure A.1   The main window of temptools. From here, you can go through the calculation steps and launch the visualization tool.
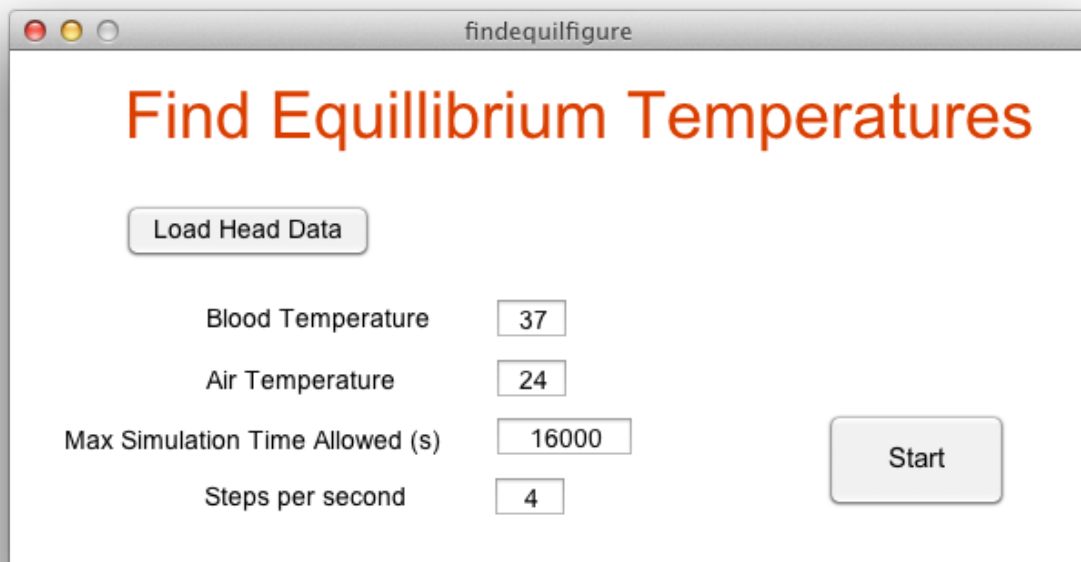
Figure A.2   This is the interface for calculating the equilibrium temperature (method explained in appendix A.3) under certain conditions.

Figure A.3   The interface for calculating temperature changes when blood flow and metabolism are time dependent. This can be achieved by either loading metabolism and blood flow datasets or by using generated activity.
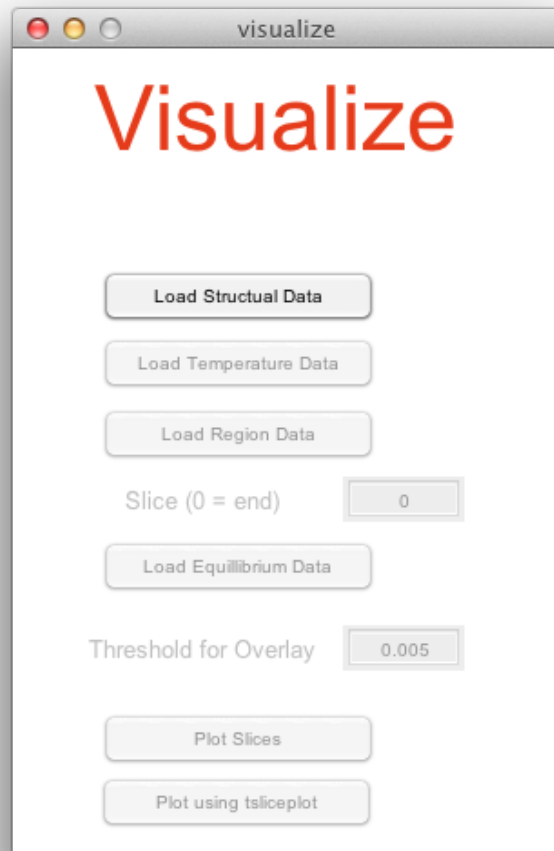
Figure A.4   Visualize your data using the temptools visualization window. This loads all of the required data and launches a slice browser or tsliceplot (see appendix B for more details).

A.1 Creating the Head Matrix

Before any calculations can be done, a matrix containing tissue-specific parameters must be created. First, a T1 contrast image should be segmented using SPM8 (`http://www.fil.ion.ucl.ac.uk/spm/software/spm8/`). For ease of consistency, the one provided by SPM8 in ./canonical/ is best to use. Using SPM's "New Segmentation" algorithim will segment the image into five different tissue types (gray matter, white matter, cerebral spinal fluid, soft tissue and bone). Once this is complete, run ImportSegmentedT1() within this directory and it will return a matrix that has been populated with the tissue-specific parameters required for accurate temperature calculations. The functions fillAir() (A.1.2), fillHoles() (A.1.3), build_skin() (A.1.4) and repair_headdata() (A.1.5) are functions required by BulkImport-NII(). More information about this procedure is in section 2.2.1.

A.1.1 ImportSegmentedT1()

```matlab
function [ total ] = ImportSegmentedT1(varargin)
%   ImportSegmentedT1 Import NII files from a directory
%    Must be run within the directory containing the files
%
%    Output: head data as single with variables stored in the 4th
%    dimension.
%
%    Author:  Greggory Rothmeier (greggroth@gmail.com)
%    Georgia State University
%    Created:  5/31/11

statusbar = waitbar(0,'Initializing');

if size(varargin) == 1
    oldFolder = cd(varargin{1});
```

```matlab
16  end
17
18
19  % ====================
20  % = Tissue Parameters =
21  % ====================
22  % Each tissue type is assigned an integer index (i.e. gray matter
       -> 11)
23  % such that tissue-specific parameters can be found by looking at
24  % that element within the corresponding storage matrix
25  % (i.e. QmSTORE(11) -> gray matter Qm)
26
27  % Parameters taken from Colins, 2004
28
29  tisorder = [11 15 5 13 3];  %  Using:  [GM WM CSF Muscle Bone]
30
31  QmSTORE = [0 0 26.1 11600 0 26.1 697 0 0 302 15575 0 697 1100
       5192];
32  cSTORE = [1006 4600 2110 3640 3800 1300 3720 3000 4200 2300 3680
       3500 3720 3150 3600];
33  rhoSTORE = [1.3 1057 1080 1035.5 1007 1850 1126 1076 1009 916
       1035.5 1151 1041 1100 1027.4];
34  kSTORE = [0.026 0.51 0.65 0.534 0.5 0.65 0.527 0.4 0.594 0.25 0.565
        0.4975 0.4975 .342 .503];
35  wSTORE = [0 1000 3 45.2 0 1.35 40 0 0 2.8 67.1 3.8 3.8 12 23.7];
36
37  % ===================================
38  % = Import the pre-segmented T1 files =
39  % ===================================
```

```matlab
40   %   The T1 contrast image sould be segmented using SPM8.
41   %   This loop needs to complete before the next one can begin
42   %   Import all of the datat and store as 'cdat1','cdat2', etc.
43   for i = 1:5
44       eval(strcat('dat',num2str(i),' = loadNII(''rc', num2str(i), '
             single_subj_T1.nii'');'))
45       %   Preallocate
46       eval(strcat('out', num2str(i),' = zeros(cat(2,size(dat',
             num2str(i),'),7));'))
47   end
48
49   % =============================
50   % = Populate the head matrix =
51   % =============================
52   % For each data file, it fills in the data from the data storage
53   % arrays for that particular type of tissue.  It picks which
54   % ever tissue is the most likely candidate for that voxel based
55   % on the segmented data
56
57   %   PROBLEM:  It returns 0 (later filled with air) if there is
58   % equal probability of a voxel being two or more different types
59   % of tissue.
60   %   SOLVED BY fillHoles()
61
62
63   for i = 1:5
64       %   Preallocate
65       holder = zeros(cat(2,size(dat1),7),'single');
66       mask = zeros(size(dat1));
```

```matlab
67        final = zeros(size(holder),'single');

68

69        %  Create a mask that indicates whether it is the mostly likely
              tissue type
70        guide = [1 2 3 4 5 1 2 3 4 5];  %  This guides it through the
            data correctly
71        eval(strcat('mask = (dat',num2str(i),'>dat',num2str(guide(i+1))
              ,') & (dat',num2str(i),'>dat',num2str(guide(i+2)),') & (dat',
              num2str(i),'>dat',num2str(guide(i+3)),') & (dat',num2str(i),'
              >dat',num2str(guide(i+4)),') & (dat',num2str(i),'~=0);'))
72        %  move structure data to new matrix
73        holder(:,:,:,1) = mask;
74        %  get indicies of tissues
75        a = find(holder(:,:,:,1) == 1);
76        %  gets coordinates from index
77        [x y z t] = ind2sub(size(holder),a);

78

79        %  go to each tissue point and store the info
80        for j = 1:length(a)
81            final(x(j),y(j),z(j),:) = [tisorder(i) 0 QmSTORE(tisorder(i
                )) cSTORE(tisorder(i)) rhoSTORE(tisorder(i)) kSTORE(
                tisorder(i)) wSTORE(tisorder(i))];
82        end

83

84        %  Saves the result to a unique output variable (out1, out2,
            etc)
85        eval(strcat('out',num2str(i),'= final;'))

86

87        clearvars a x y z t holder final;
```

```matlab
88        waitbar(i/6,statusbar,sprintf(['File ',num2str(i),' Import
             Compete']));
89    end
90
91    % The filleAir() function checks for any voxels which were not
92    % assigned a tissue type and fills them in with air
93    almostthere = fillAir(out1+out2+out3+out4+out5);
94    % The fillHoles() function corrects for a voxel having two
95    % equally-probable tissue types
96    total = single(buildskin(fillHoles(dat1,dat2,dat3,dat4,dat5,
            almostthere)));
97    waitbar(1,statusbar,'Saving Data')
98
99    cd(oldFolder);
100   close(statusbar);
101
102   end
```

A.1.2 fillAir()

```matlab
1     function [ output ] = fillAir( tissue )
2     % fillAir() fills gaps in data with air
3     % Once you import all of the data using loadNII(), run it though
4     % this to fill in the remaining spaces with air.
5
6     airdata = [1 0 0 1006 1.3 0.026 0];
7
8     %  Picks out air spots
9     a = find(tissue(:,:,:,1) == 0);
10    [x y z t] = ind2sub(size(tissue),a);
```

```matlab
11
12  for i = 1:length(a)
13      tissue(x(i),y(i),z(i),:) = airdata;
14  end
15
16  output = tissue;
17  end
```

A.1.3 fillHoles()

```matlab
1   function [ out_head ] = fillHoles( in1,in2,in3,in4,in5,headin)
2   % fillHoles() checks for misassigned voxels
3   %
4   % Solves an issue where a voxel with two equally probable tissue
5   % types resulted in being assigned as air.  This checks for air
6   % voxels that are surrounded by tissue and decides a tissue it
7   % it would be best suited as
8
9   %  I only need the tissue indices so this makes things easier down
       the line
10  head = squeeze(headin(:,:,:,1));
11
12  %%  Data Storage
13  QmSTORE = [0 0 26.1 11600 0 26.1 697 0 0 302 15575 0 697 1100
       5192];
14  cSTORE = [1006 4600 2110 3640 3800 1300 3720 3000 4200 2300 3680
       3500 3720 3150 3600];
15  rhoSTORE = [1.3 1057 1080 1035.5 1007 1850 1126 1076 1009 916
       1035.5 1151 1041 1100 1027.4];
```

```matlab
16  kSTORE = [0.026 0.51 0.65 0.534 0.5 0.65 0.527 0.4 0.594 0.25 0.565
        0.4975 0.4975 .342 .503];
17  wSTORE = [0 1000 3 45.2 0 1.35 40 0 0 2.8 67.1 3.8 3.8 12 23.7];
18
19  %%  Get locations of holes
20  %   Where two tissue types have the same probability
21
22  idx1 = (in1==in2 | in1 == in3 | in1==in4 | in1==in5) & logical(in1)
        ;
23  idx2 = (in1==in2 | in2 == in3 | in2==in4 | in2==in5) & logical(in2)
        ;
24  idx3 = (in1==in3 | in2 == in3 | in3==in4 | in3==in5) & logical(in3)
        ;
25  idx4 = (in1==in4 | in2 == in4 | in3==in4 | in4==in5) & logical(in4)
        ;
26  idx5 = (in1==in5 | in2 == in5 | in3==in5 | in4==in5) & logical(in5)
        ;
27  %  This array will have a zero anywhere there were two or more
28  % common elements between any of the five arrays.
29  idx = idx1|idx2|idx3|idx4|idx5;
30
31  [xmax ymax zmax] = size(in1)
32  [x y z] = ind2sub(size(in1),find(idx));  %  get x, y and z
      coordinates of the holes
33
34  for i = 1:length(x)  %  go to each hole and do work
35      if (x(i)~=1)&&(y(i)~=1)&&(z(i)~=1)&&(x(i)~=xmax)&&(y(i)~=ymax)
          &&(z(i)~=zmax)&&(headin(x(i),y(i),z(i),1)==1)  %  keeps away
          from the edge and only looks at voxels that were assigned air
```

```
36          [commonesttissue nouse secondbest] = mode([head(x(i)+1,y(i)
               ,z(i)) head(x(i)-1,y(i),z(i)) head(x(i),y(i)+1,z(i)) head
               (x(i),y(i)-1,z(i)) head(x(i),y(i),z(i)+1) head(x(i),y(i),
               z(i)-1)]);
37          % if air and something else are equally common, it'll
               choose air.  This forces it to pick the tissue if
               possible.
38          if commonesttissue == 1 && length(secondbest{1})>=2
39              commonesttissue = secondbest{1}(2);
40          end
41          headin(x(i),y(i),z(i),:) = [commonesttissue 0 QmSTORE(
               commonesttissue) cSTORE(commonesttissue) rhoSTORE(
               commonesttissue) kSTORE(commonesttissue) wSTORE(
               commonesttissue)];
42      end
43  end
44
45  out_head = headin;
46
47  end
```

A.1.4 build_skin()

```
1  function [ head_out ] = build_skin( head_in )
2  % build_skin() Creates a layer of skin around the head
3  %
4  % This will check all voxels that were previously labeled
5  % as soft tissue and checks if it has a neighbor which is air.
6  % If so, then it is reassigned as skin.
7
```

```
 8   if ndims(head_in)==4
 9       head_in = head_in(:,:,:,1);
10   end
11
12   % Git a list of all voxels labeled as muscle
13   muscle_voxels = find(head_in==13);
14
15   % Go through each of them and check for neighboring air voxels
16   for i=1:length(muscle_voxels)
17       [x y z] = ind2sub(size(head_in), muscle_voxels(i));
18       % makes sure we're not at a voxel at the boundry of the dataset
19       if (x~=1) && (x~=size(head_in,1)) && (y~=1) && (y~=size(head_in
            ,2)) && (z~=1) && (z~=size(head_in,3))
20         % Looks for neighboring voxels that are air
21         if ((head_in(x+1,y,z)==1) || (head_in(x-1,y,z)==1) || (head_in
              (x,y+1,z)==1) || (head_in(x,y-1,z)==1) || (head_in(x,y,z+1)
              ==1) || (head_in(x,y,z-1)==1))
22             head_in(x,y,z) = 14;
23         end
24     end
25   end
26
27   head_out = repair_headdata(head_in);
28
29   end
```

A.1.5 repair_headdata()

This function will go through the dataset and make sure the tissue-specific parameters are correct for the tissue type assigned for that voxel. fillAir(), fillHoles() and build_skin() all

correct mislabeled voxels, but they only correct the tissue assignment. After using any of these functions, the data must be passed through repair_headdata to update the stored parameters.

```matlab
function [ head_out ] = repair_headdata( head_in )
% repaid_headdata repopulates the headdata matrix
%   If any changes are made to the index column in the headdata
% matrix, use this function to repopulate and correct the
% parameter values before running any other functions.
%  head_in can be either 3 or 4 dimenisions


% ====================
% = Parameter Storage =
% ====================

QmSTORE = [0 0 26.1 11600 0 26.1 697 0 0 302 15575 0 500 1100
    5192];
cSTORE = [1006 4600 2110 3640 3800 1300 3720 3000 4200 2300 3680
    3500 3010 3150 3600];
rhoSTORE = [1.3 1057 1080 1035.5 1007 1850 1126 1076 1009 916
    1035.5 1151 978.5 1100 1027.4];
kSTORE = [0.026 0.51 0.65 0.534 0.5 0.65 0.527 0.4 0.594 0.25 0.565
     0.4975 0.3738 .342 .503];
wSTORE = [0 1000 3 45.2 0 1.35 40 0 0 2.8 67.1 3.8 3.3 12 23.7];

if ndims(head_in)==4
    head_in = head_in(:,:,:,1);
end

```

```matlab
23  % Reassign the parameter values
24  head_out = cat(4,head_in, zeros(size(head_in)), QmSTORE(head_in),
       cSTORE(head_in), rhoSTORE(head_in), kSTORE(head_in), wSTORE(
       head_in));
25
26  end
```

A.2 Loading the fMRI Data

The following sections details the processing required to convert the BOLD data (in NIFTI format) to metabolism and blood flow time-courses that can then be used to calculate temperature.

A.2.1 sample_bold_import()

The following code automates the procedure of processing and doing all the calculations on the dataset reported in Dhamala et al. [36]. It's is written for my data on my machine, but it can be used to gain a better understanding of the procedure. For a conceptual explanation, see section 2.2.3.

```matlab
%%=====================================================================
%% How to process preprocessed BOLD data to calculate temperature
%%=====================================================================

% This Matlab script was used to automate the the process of using
% BOLD data stored in NIFTI (*.nii) format to calculate temperature
% changes.  The particulars of the code may be specific to this
% case, but the procedure should be the same when doing these
% calculations on other datasets.  All required functions are
% included as an attachment to my thesis and are available on my
% github (https://github.com/greggroth/tempcalc)

cd('/Users/Greggory/Documents/Data/fmri_rhythmic_tapping01/NIFTI')

directories = dir('*01');

%%  Move coregistered files to new Directory
for i = 1:length(directories)
```

```matlab
19        dir_name = directories(i).name;
20        main_path = cd( [dir_name filesep dir_name '_NIFTI_1'] );
21        mkdir 'Coregistered'
22        movefile('r*.nii','Coregistered')
23        main_path = cd( [dir_name filesep dir_name '_NIFTI_2'] );
24        mkdir 'Coregistered'
25        movefile('r*.nii','Coregistered')
26        cd(main_path)
27 end
28
29 %%  Calculate Rest State
30 disp('Calculating Rest State')
31 for i = 1:length(directories)
32        dir_name = directories(i).name;
33        avg_NII_rest([dir_name filesep dir_name '_NIFTI_1' filesep '
           Coregistered']);
34        avg_NII_rest([dir_name filesep dir_name '_NIFTI_2' filesep '
           Coregistered']);
35 end
36
37
38 %%  Normalize to Rest and Mask
39 disp('Normalize to Rest and Mask')
40 for i = 1:length(directories)
41        dir_name = directories(i).name;
42        avg_NII_normalize([dir_name filesep dir_name '_NIFTI_1' filesep
            'Coregistered'], fullfile(dir_name, [dir_name '_NIFTI_1'], '
           Coregistered', 'RestState', 'RestStateAvg.nii'), '
           fullBrainMask.nii');
```

```matlab
43        avg_NII_normalize([dir_name filesep dir_name '_NIFTI_2' filesep
              'Coregistered'], fullfile(dir_name, [dir_name '_NIFTI_2'], '
              Coregistered', 'RestState', 'RestStateAvg.nii'), '
              fullBrainMask.nii');
44    end
45
46
47    %%  Calculate metabolism and blood flow change
48    disp('Calculate metabolism and blood flow change')
49    for i = 1:length(directories)
50        dir_1 = [ directories(i).name filesep  directories(i).name '
              _NIFTI_1' filesep 'Coregistered' filesep 'Normalized_to_rest'
              ];
51        dir_2 = [ directories(i).name filesep  directories(i).name '
              _NIFTI_2' filesep 'Coregistered' filesep 'Normalized_to_rest'
              ];
52        BOLDtoMF(dir_1);
53        BOLDtoMF(dir_2);
54    end
55
56
57    %%  Calculate the change in temperature based on metabolism and
58    %   blood flow
59
60    % load('equil.mat');  % equillibriumT
61    % load('tt_headdata.mat');  % headdata
62    mask = loadNII('fullBrainMask.nii');
63
64    for i = 1:length(directories)
```

```matlab
65        disp([int2str(i) '-1 started'])

66        tic

67        % Part I

68        actResult.dat = tempCalcDynMF(headdata, 37, 24, 720, 360,
             equillibriumT, ...

69            fullfile(directories(i).name,[directories(i).name '_NIFTI_1
                '],'Coregistered', 'Normalized_to_rest', 'Output_18-Sep
                -2011', 'm.mat'), ...

70            fullfile(directories(i).name,[directories(i).name '_NIFTI_1
                '],'Coregistered', 'Normalized_to_rest', 'Output_18-Sep
                -2011', 'f.mat'), ...

71            4, mask);

72        % Store the parameters used for the calculations for reference
             in the future

73        [c lmax] = max(actResult.dat(:));

74        [likelymax x y z] = ind2sub(size(actResult.dat),lmax);

75        actResult.likelymaxslice = round(likelymax/2);

76        actResult.bloodT = 37;

77        actResult.airT = 24;

78        actResult.tmax = 360;

79        actResult.stepf = 2;

80        actResult.savestepf = 4;

81        actResult.metabandflowdata = 'From Dataset';

82        save(fullfile(directories(i).name,[directories(i).name '
             _NIFTI_1'],'Coregistered', 'Normalized_to_rest', 'Output_18-
             Sep-2011','tt_act_res.mat'), 'actResult');

83        old = cd([directories(i).name,filesep,[directories(i).name '
             _NIFTI_1'],filesep,'Coregistered', filesep,'
             Normalized_to_rest', filesep,'Output_18-Sep-2011']);
```

```matlab
84        writeT_to_nii(actResult, equillibriumT, exp_nii);

85        cd(old)

86        clear actResult

87        % Part II

88        disp([int2str(i) '-2 started'])

89        actResult.dat = tempCalcDynMF(headdata, 37, 24, 720, 360,
             equillibriumT, ...

90           fullfile(directories(i).name,[directories(i).name '_NIFTI_2
                '],'Coregistered', 'Normalized_to_rest', 'Output_18-Sep
                -2011', 'm.mat'), ...

91           fullfile(directories(i).name,[directories(i).name '_NIFTI_2
                '],'Coregistered', 'Normalized_to_rest', 'Output_18-Sep
                -2011', 'f.mat'), ...

92           4, mask);

93        [c lmax] = max(actResult.dat(:));

94        [likelymax x y z] = ind2sub(size(actResult.dat),lmax);

95        actResult.likelymaxslice = round(likelymax/2);

96        actResult.bloodT = 37;

97        actResult.airT = 24;

98        actResult.tmax = 360;

99        actResult.stepf = 2;

100       actResult.savestepf = 4;

101       actResult.metabandflowdata = 'From Dataset';

102       save(fullfile(directories(i).name,[directories(i).name '
             _NIFTI_2'],'Coregistered', 'Normalized_to_rest', 'Output_18-
             Sep-2011','tt_act_res.mat'), 'actResult');

103       old = cd([directories(i).name,filesep,[directories(i).name '
             _NIFTI_2'],filesep,'Coregistered', filesep,'
             Normalized_to_rest', filesep,'Output_18-Sep-2011']);
```

```
104        writeT_to_nii(actResult, equillibriumT, exp_nii);
105        cd(old)
106        clear actResult
107        disp([int2str(i) ' finished in ' num2str(toc)])
108    end
```

A.2.2 avg_NII_rest()

```
1    function [ ] = avg_NII_rest( varargin )
2    % Collects datasets which are part of the
3    % resting state and averages them together to
4    % give a resting-state image
5    %
6    % THIS MUST BE EDITED TO WORK
7    % This is written for my data and you should read
8    % and understand what it is doing before you use it.
9    % It will almost certainly require some editing
10   % to select the right range of data.
11
12   %% Setup
13   switch length(varargin)
14       case 0
15           fold_name = uigetdir;
16           if ~fold_name   % Cancel Button
17                   return
18           end
19       case 1
20           fold_name = varargin{1};
21       otherwise
22   end
```

```matlab
23
24  %  Go to the folder containing the files
25  oldfold = cd(fold_name);
26  file_list = dir('*.nii');
27
28  %  Select resting state images
29  %  (first and last 10 steps in my case).
30  %  EDIT THIS TO FIT YOUR CASE
31  file_list = file_list([1:10 170:180]);
32  file_count = length(file_list);
33
34  %  Cell array to store all of the datasets in.
35  datHolder = cell(file_count,1);
36
37  statusbar = waitbar(0,'Initializing');
38
39  for j=1:file_count
40      try
41          waitbar(j/file_count,statusbar,sprintf('%d%%',round((j/
              file_count)*100)));
42      catch err
43          return
44      end
45      fi = load_nii(file_list(j).name);
46      datHolder{j} = fi.img;
47  end
48
49  %%  Calculate the mean
50  ymax = size(datHolder{1},2);
```

```matlab
zmax = size(datHolder{1},3);
output = zeros(size(datHolder{1}));


for i=1:ymax
    try
        waitbar(i/ymax,statusbar,sprintf('%d%%',round((i/ymax)*100)
            ));
    catch err
        return
    end
    for k=1:zmax
        excStr = cell(length(datHolder),1);
        for l=1:length(datHolder)
            excStr{l} = [',datHolder{' int2str(l) '}(:,' int2str(i)
                ',' int2str(k) ')'''];
        end
        comb = eval(['cat(1' cell2mat(excStr') ')']);
        output(:,i,k) = mean(comb);
    end
end

close(statusbar)

fi.img = output;
mkdir('RestState')
save_nii(fi,fullfile('RestState','RestStateAvg.nii'));

cd(oldfold)
end
```

A.2.3 avg_NII_normalize()

```matlab
function [  ] = avg_NII_normalize( varargin )
%   Uses the resting-state image calculated using
%   avg_NII_rest() to normalize the rest of the data

% If no inputs are given, the "open file..." UI will
% prompt for the required information.


%% Setup
switch length(varargin)
    case 0
        fold_name = uigetdir('Directory Containing Data');
        if ~fold_name  % Cancel Button
            return
        end

        [rest_file rest_path rest_index]= uigetfile('*.nii','
            Resting State NIFTI File');
        switch rest_index
            case 0
                return
            case 1
                rest_dat = load_nii(fullfile(rest_path,rest_file));
                rest_dat = double(rest_dat.img);
            otherwise
                error('An error has occured loading the resting
                    state data')
        end
```

```matlab
26
27          [mask_file mask_path mask_index] = uigetfile('*.nii','Mask'
               );
28          switch mask_index
29              case 0
30                  return
31              case 1
32                  mask_dat = load_nii(fullfile(mask_path, mask_file))
                       ;
33                  mask_dat = logical(mask_dat.img);
34                  if max(size(mask_dat) ~= size(rest_dat))
35                      error('The Mask and Resting State files must
                          have the same size')
36                  end
37              otherwise
38                  error('An error has occured loading the resting
                       state data')
39          end
40      case 1
41          fold_name = varargin{1};
42          [rest_file rest_path rest_index]= uigetfile('*.nii','
               Resting State NIFTI File');
43          switch rest_index
44              case 0
45                  return
46              case 1
47                  rest_dat = load_nii(fullfile(rest_path,rest_file));
48                  rest_dat = double(rest_dat.img);
49              otherwise
```

```matlab
                    error('An error has occured loading the resting
                         state data')
        end
    case 2
        fold_name = varargin{1};
        rest_dat = loadNII(varargin{2});
        [mask_file mask_path mask_index] = uigetfile('*.nii','Mask'
            );
        switch mask_index
            case 0
                return
            case 1
                mask_dat = load_nii(fullfile(mask_path, mask_file))
                    ;
                mask_dat = logical(mask_dat.img);
                if max(size(mask_dat) ~= size(rest_dat))
                    error('The Mask and Resting State files must
                         have the same size')
                end
            otherwise
                error('An error has occured loading the resting
                     state data')
        end
    case 3
        fold_name = varargin{1};
        rest_dat = loadNII(varargin{2});
        mask_dat = loadNII(varargin{3});
    otherwise
        return
```

```matlab
74    end
75
76    %  Go to the folder containing the files
77    oldfold = cd(fold_name);
78    file_list = dir('*.nii');
79    file_count = length(file_list);
80
81    %  Make a directoy to save the normalized data to
82    saveDir = 'Normalized_to_rest';
83    if ~isdir(saveDir)
84        mkdir(saveDir);
85    end
86
87    statusbar = waitbar(0,'Initializing');
88
89    % for each file: load it, devide by the rest state and save it
90    for i=1:file_count
91        try
92            waitbar(i/file_count,statusbar,[fold_name sprintf('%d%%',
                round((i/file_count)*100))]);
93        catch err
94            return
95        end
96        [file_path file_name file_ext] = fileparts(file_list(i).name);
97        file_hold = load_nii(file_list(i).name);
98        file_hold.img = double(file_hold.img)./rest_dat - 1;
99        file_hold.img(~mask_dat) = 0;              % set everything
              outside the mask to 0
100       file_hold.img(isnan(file_hold.img)) = 0;  % set all NaN's to 0
```

```
101        file_hold.img(isinf(file_hold.img)) = 0;   % set all inf's to 0
102        file_hold.img(file_hold.img == -1) = 0;     % correct these for
             voxels that are giving me problems
103        file_hold.hdr.dime.datatype = 16;  % set the datatype to single
104        file_hold.hdr.dime.bitpix = 16;
105        save_nii(file_hold,fullfile(saveDir,[file_name '_rn' file_ext])
             )
106   end
107
108   close(statusbar)
109   cd(oldfold)
110
111   end
```

A.2.4 BOLDtoMF()

```
1   function [ ] = BOLDtoMF( varargin)
2   %BOLDtoMF Calculate metabolism and blood from from BOLD reponse
3   %
4   %    Input: Directory containing a series of *.nii files of the BOLD
5   %    response.
6   %
7   %    Output: Two new files will be created in a new subdirectory
8   %    with a variable for each time step.
9   %
10  %    Usage:
11  %         BOLDtoMF
12  %         BOLDtoMF(directory)
13  %
14  %    If a directory is not provided, one will be requested.
```

```matlab
15  %
16  %    Method from Sotero, et. al. 2011
17
18  % =========
19  % = Setup =
20  % =========
21  % if a folder isn't an argument, it'll prompt for one
22  switch length(varargin)
23      case 0
24          fold_name = uigetdir;
25          if ~fold_name  % Cancel Button pressed
26              return
27          end
28      case 1
29          fold_name = varargin{1};
30      otherwise
31          error('Input is not understood')
32  end
33
34  %  Go to the folder containing the files
35  oldfold = cd(fold_name);
36  file_list = dir('*.nii');
37  file_count = length(file_list);
38
39  %  Set up a directory for the outputs
40  newFolder = ['Output_',datestr(clock,1)];
41  mkdir(newFolder)
42
43  %  Make *.mat files to append the data to
```

```matlab
44   m0001 = 0;  f0001 = 0;
45   save(['./' newFolder '/m.mat'],'m0001');
46   save(['./' newFolder '/f.mat'],'f0001');
47
48   s = loadNII(file_list(1).name);
49   norm = ones(size(s));
50
51   % ==========
52   % = Do Work =
53   % ==========
54   % This will calculate the metabolism and blood flow.  The output is
55   % appended to 'm.mat' and 'f.mat' within a new folder created
56   % within the directory containing the data.
57
58   statusbar = waitbar(0,'Initializing');
59
60   maxBOLD = 0.22;
61
62   % Required Parameters
63   %   [alpha beta a       b       c       A    ]
64   p = [0.4 1.5 0.1870 0.1572 -0.6041 maxBOLD];
65
66   % Calc flow and metabolism for when BOLD = 1
67   s = 0;
68   y = -((p(4)*p(2))/(p(1)+p(2)*p(5)))*((p(6)-s)/(p(6)*p(3)^p(2)))
         ^(1/(p(1)+p(2)*p(5)));
69   fNOACT = -((p(1)+p(2)*p(5))/(p(4)*p(2)))*lambertw(y);
70   mNOACT = p(3)*fNOACT^(p(5)+1)*exp(-p(4)*fNOACT);
71
```

```matlab
72
73   %%  Calc flow and metabolism
74   disp(fold_name)
75   for j=1:file_count
76       try
77          waitbar(j/file_count, statusbar, sprintf('%d%%', round((j/
                file_count)*100)));
78       catch err
79              return
80       end
81       s = loadNII(file_list(j).name);  %  Load up the file
82       s(isnan(s)) = 1;
83       s(isinf(s)) = 1;
84       y = -((p(4)*p(2))/(p(1)+p(2)*p(5))).*((p(6)-s)./(p(6)*p(3)^p(2)
                )).^(1/(p(1)+p(2)*p(5)));
85       if (size(y,1)==91)&&(size(y,2)==109)&&(size(y,3)==91)
86              f = -((p(1)+p(2)*p(5))/(p(4)*p(2))).*lambw_mex(real(y));
87       else
88              f = -((p(1)+p(2)*p(5))/(p(4)*p(2))).*lambw(y);
89       end
90       m = p(3)*f.^(p(5)+1).*exp(-p(4)*f);
91       % Clean up NaNs that may have popped up
92       m(isnan(m))=1;
93       f(isnan(f))=1;
94       % Normalize to resting m and f
95       m = m./mNOACT;
96       f = f./fNOACT;
97
98       % Rename and save the data
```

```
 99        eval(['m' sprintf('%04d',j) ' = m;']);
100        eval(['f' sprintf('%04d',j) ' = f;']);
101        eval(['save(''./' newFolder '/m.mat'', ''m' sprintf('%04d',j) '
               '',''-append'');']);
102        eval(['save(''./' newFolder '/f.mat'', ''f' sprintf('%04d',j) '
               '',''-append'');']);
103        clear m0* f0*
104    end
105
106    close(statusbar)
107    cd(oldfold)
108    end
```

A.2.5 lambw() and lambw_mex()

The lambw() function is a wrapper for the wapr() function available on Matlab FileExchange
(http://www.mathworks.com/matlabcentral/fileexchange/3644-real-values-of-the-lambert-w-f
content/Lambert/wapr.m). A compiled version of this function (lambw_mex()) runs much
faster and is recommended. This function is used over Matlab's built-in Lambert-W function
for the sake of performance.

```
1  function [ array_out ] = lambw( array_in )
2  % lambw Wrapper for wapr()
3  % Available:  http://www.mathworks.com/matlabcentral/fileexchange
      /3644-real-values-of-the-lambert-w-function/content/Lambert/wapr.
      m
4  %   Dwapr() doesn't work any arrays over Nx1, so this steps through
5  %   the full matrix and gives the rows to wapr.  Works pretty fast.
6  %#codegen
7
8  if ndims(array_in) ~= 3
```

71

```matlab
 9       error('This only works (for now) with a three dimensional array
            .')
10  end
11
12  xmax = size(array_in ,1);
13  ymax = size(array_in ,2);
14
15  array_out = zeros(size(array_in));
16     for ix =1: xmax
17         for iy =1: ymax
18             array_out(ix ,iy ,:) = wapr(array_in(ix ,iy ,:));
19         end
20     end
21  end
```

A.3 Calculating the Equilibrium Temperature

In order to determine the temperature fluctuations due to changes in activity, the baseline temperature must first be established for each voxel. The function tempCalcEquilibrium() will update the temperature using the Penne's bioheat equation (Eq. (2.4)) until the change in temperature for each voxel falls below a certain threshold. Details about this procedure are available in section 2.2.2.

A.3.1 tempCalcEquilibrium()

```
1   function temperature_Out = tempCalcEquillibrium(tissue,bloodT,airT,
        nt,tmax,pastCalc,printprogress)
2   % tempCalcEquillibrium  Find the equillibrium values
3   %    tissue: holds all of the strucual information
4   %    bloodT: Temperature of the blood
5   %    airT:   Temperature of the surrounding ait
6   %    nt:     Max number of time steps
7   %    tmax:   Total amount of time the simulation should run over
8   %
9   %    This is based off of tempCalc() but loops until the rate of
10  % change of a each voxel is sufficiently small then outputs
11  % what's calculated.  If if takes too long to do all at once,
12  % split it up into smaller time chunks and use the last step
13  % from the previous dataset as pastCalc in order to resume.
14  %
15  %    Note: This does not save the time corse because it can take
16  % a lot of step to find the equillibrium.  It outputs the last
17  % time step.
18  %
19  %    Written by Greggory Rothmeier (greggroth@gmail.com)
```

```matlab
20  %    Georgia State University Dept. Physics and Astronomy
21  %    May, 2011
22  tic
23  %%   Default Values
24  if nargin<2, bloodT = 37;        end
25  if nargin<3, airT = 24;          end
26  if nargin<4, nt = 100;           end
27  if nargin<5, tmax = 50;          end
28  if nargin<6, pastCalc = 0;       end
29  if nargin<7, printprogress = 1; end
30
31  % These rescue the data if the calculation is interrupted.
32  global temperature
33  global dirty
34
35  c = onCleanup(@InterCatch);
36  dirty = 1;
37
38  dx = 2*10^-3;          %   Voxel size (m)
39
40  if nt<(2*tmax),
41      warning('Time step size is not large enough.  Results will be
          unreliable.  Consider increasing the number of steps or
          reducing tmax.')
42  end
43
44
45  % Constants used that aren't already stored in tissue
46  [xmax ymax zmax t] = size(tissue);
```

```matlab
47   clear t;
48   dt = tmax/(nt-1);
49   % rhoBlood = 1057;
50   % wBlood = 1000;
51   % cBlood = 3600;
52
53   % =========
54   % = Setup =
55   % =========
56   %   Starts all tissue voxels at bloodT (default 37) and maintains
57   % air at airT (default 24)
58   %   The condition squeeze(tissue(:,:,:,)~=airIndex picks out the
59   % elements that are tissue
60
61   temperature = ones(3,xmax,ymax,zmax,'single')*airT;
62   if pastCalc == 0
63       temperature(1,squeeze(tissue(:,:,:,1))~=1) = bloodT;
64   else
65       temperature(1,:,:,:) = pastCalc;
66   end
67   numElements = numel(temperature(1,:,:,:));
68
69   % ===========
70   % = Do Work =
71   % ===========
72   %   This is a vectorized version of the next section.  For the love
73   % of god don't make any changes to this without first looking below
74   % to make sure you know what you're changing.  This is [nearly]
75   % impossible to understand, so take your time and don't break it.
```

```matlab
76  %    data is stored in 'tissue' as such :
77  %    [tissuetype 0 Qm c rho k w]; <-- second element is blank for
       all.
78  %    [    1       2 3 4  5  6 7
79
80  averagedk = (circshift(tissue(:,:,:,6),[1 0 0])+circshift(tissue
       (:,:,:,6),[-1 0 0])+circshift(tissue(:,:,:,6),[0 1 0])+circshift(
       tissue(:,:,:,6),[0 -1 0])+circshift(tissue(:,:,:,6),[0 0 1])+
       circshift(tissue(:,:,:,6),[0 0 -1])+tissue(:,:,:,6))/7;
81  rhoblood = 1057;
82  cblood = 3600;
83
84  %%  Specify Percision Goal
85  tolerence = 1;     % fraction of voxels have a slope less than '
       zeropoint'
86  zeropoint = 2.5e-7;  % point at which the slope between two *steps*
        is considered essentially zero
87
88
89  goal = numElements - tolerence*numElements;
90  goon = numElements; %  Forces the while loop to run the first time
91  format shortG;
92  % temperature(1,:,:,:) = Current Temperature
93  % temperature(2,:,:,:) = Next Temperature
94  % Resets after each update
95  if printprogress
96      disp(['Goal:  ', num2str(goal),' remaining voxels'])
97  end
98  t2 = 1;
```

```matlab
 99   while goon(1)>goal && t2<=nt  % runs until either 'goal' elements
        have a slope greater than 'zeropoint' or it exceeds nt
100      if printprogress
101       disp([t2 goon(1) ((numElements-goon(1))/numElements)*100]) %
            progress
102      end
103      temperature(2,:,:,:) = squeeze(temperature(1,:,:,:)) + ...
104           dt/(tissue(:,:,:,5).*tissue(:,:,:,4)).* ...
105           ((averagedk/dx^2).*...
106           (circshift(squeeze(temperature(1,:,:,:)),[1 0 0])-2*squeeze
                 (temperature(1,:,:,:))+circshift(squeeze(temperature
                 (1,:,:,:)),[-1 0 0])+...  % shift along x
107            circshift(squeeze(temperature(1,:,:,:)),[0 1 0])-2*squeeze
                 (temperature(1,:,:,:))+circshift(squeeze(temperature
                 (1,:,:,:)),[0 -1 0])+...  % shift along y
108            circshift(squeeze(temperature(1,:,:,:)),[0 0 1])-2*squeeze
                 (temperature(1,:,:,:))+circshift(squeeze(temperature
                 (1,:,:,:)),[0 0 -1]))...  % shift along z
109             -(1/6000)*rhoblood*tissue(:,:,:,7)*cblood.*(squeeze(
                  temperature(1,:,:,:))-bloodT)+tissue(:,:,:,3));
110      %   resets the air temperature back since it's also modified
111      % above, but it needs to be kept constant throughout the
112      % calculations
113      temperature(2,squeeze(tissue(:,:,:,1))==1) = airT;
114      %   checks how quickly the temperature is changing and if it is
115      % close enough to zero to be considered stopped ('zeropoint')
116      goon = size(temperature(abs(squeeze(temperature(2,:,:,:)-
            temperature(1,:,:,:)))>zeropoint));
117      temperature(1,:,:,:) = temperature(2,:,:,:);
```

```matlab
118        t2 = t2 + 1;
119    end
120
121    temperature_Out = temperature (2 ,: ,: ,:);
122    dirty = 0;
123
124    % equilTemperature = temperature_Out;
125    % save ('equil.mat','equilTemperature');
126
127    %% To Combine Datasets
128    %   use this technique if there are seperate datasets that need
129    % combining
130    %      vertcat (squeeze (res1 (: ,: ,: ,:)),squeeze (res2 (2: end ,: ,: ,:)))
131    %   Where for all by the first dataset, you need to do the time from
132    % 2: end so that there are no repeats (remember that the last
133    % timestep from the previous dataset serves as the first for the
134    % new one )
135
136
137    time = toc;
138    end
139
140    % Recovers the data if calculation was interrupted
141    function InterCatch
142    global dirty
143    if dirty
144        disp ('Interupt Intercepted.  Inprepretating Interworkspace Data
                Interfaces.')
145        global temperature
```

```matlab
146        equillibriumT = temperature;

147        save('equiltempAbortDump.mat','equillibriumT');

148        % setappdata(0,'InterpOut',temperature);

149    end

150    end
```

A.4 Calculating the Temperature Change

The following function takes as inputs the head data matrix (appendix A.1), the metabolism and blood flow time courses (appendix A.2) and the equilibrium temperatures (appendix A.3) and calculates the temperature time-course. More details about this algorithm can be found in section 2.2.4.

A.4.1 tempCalcDynMF

```matlab
function temperatureOut = tempCalcDynMF(tissue,bloodT,airT,nt,tmax,
    pastCalc,metab,flow,savesteps,region)
% tempCalcChaning Metabolism  How does changin metabolism
% affect things?
%
%   tissue: holds all of the strucual information
%   bloodT: Temperature of the blood
%   airT:   Temperature of the surrounding ait
%   nt:     Number of time steps
%   tmax:   Total amount of time the simulation should run over
%
%   region: logical matrix same size as head
%
%   Writen by Greggory Rothmeier (greggroth@gmail.com)
%   Georgia State University Dept. Physics and Astronomy
%   May, 2011


statusbar = waitbar(0,'Initializing');


%%   Default Values
if nargin<2,   bloodT = 37;           end
```

```matlab
21  if nargin<3,  airT = 24;           end
22  if nargin<4,  nt = 3;              end
23  if nargin<5,  tmax = 1;            end
24  if nargin<6,  pastCalc = 0;        end
25
26
27  % Length of one side of a voxel (m)
28  dx = 2*10^-3;
29
30  if nt<(2*tmax),
31      warning('Time step size is not large enough.  Results will be
            unreliable.  Consider increasing the number of steps or
            reducing tmax.')
32  end
33
34
35  % Constants used that aren't already stored in tissue
36  [xmax ymax zmax t] = size(tissue);
37  clear t;
38  dt = ones([xmax ymax zmax])*(tmax/(nt-1));
39  % rhoBlood = 1057;
40  % wBlood = 1000;
41  % cBlood = 3600;
42
43  %%  Determine Metab/Flow Data Storage System
44  if ischar(metab)&&ischar(flow)
45      % if file locations are given rather than data
46          option = 1;
47  else
```

```matlab
48      % Preallocate matrices for holding metabolism and blood flow data
49          metabMulti = ones([xmax ymax zmax],'single');
50          flowMulti = ones([xmax ymax zmax],'single');
51          option = 0;
52  end
53
54  %%  Maps
55  % Creates a map that identifies where there is tissue
56  % the condition squeeze(tissue(:,:,:,)~=airIndex picks out the
57  % elements that are tissue
58
59  tmax = ceil((nt-1)/savesteps);
60  temperatureOut = ones(tmax,xmax,ymax,zmax,'single');
61  temperature = ones(2,xmax,ymax,zmax,'single')*airT;
62  if pastCalc == 0
63      temperature(1,squeeze(tissue(:,:,:,1))~=1) = bloodT;
64  else
65    % Starts everything off at the pre-determined equilibium
            temperatures
66      temperature(1,:,:,:) = pastCalc(end,:,:,:);
67  end
68  temperatureOut(1,:,:,:) = temperature(1,:,:,:);
69
70
71  % ==========
72  % = Do Work =
73  % ==========
74  %   This is a vectorized version of the next section.  For the love
75  % of god don't make any changes to this without first looking below
```

```matlab
76  % to make sure you know what you're changing.  This is [nearly]
77  % impossible to understand because it's been vectorized, so take
78  % your time and don't break it.  Data is stored in 'tissue' as such
       :
79  %   [tissuetype 0 Qm c rho k w] <--   second element is blank for all
       .
80  %   [    1      2 3 4  5  6 7]
81
82  averagedk = (circshift(tissue(:,:,:,6),[1 0 0])+circshift(tissue
        (:,:,:,6),[-1 0 0])+circshift(tissue(:,:,:,6),[0 1 0])+circshift(
        tissue(:,:,:,6),[0 -1 0])+circshift(tissue(:,:,:,6),[0 0 1])+
        circshift(tissue(:,:,:,6),[0 0 -1])+tissue(:,:,:,6))/7;
83  rhoblood = 1057;
84  cblood = 3600;
85
86  %%  Only saves every 4 steps to reduce the final matrix size
87  for t2 = 1:nt-1
88      waitbar(t2/(nt-1),statusbar,sprintf('%d%%',round(t2/(nt-1)*100))
         );
89
90  % if a variable needs to be used multiple times for the same time
       step.
91      t3 = floor((t2-1)/4)+1;   % 1 1 1 1 2 2 2 2 3 3 . . .
92
93      % if a file is specified, pulls the data from the file for each
           step
94      if option
95          eval(strcat('load(fullfile(metab),''-mat'',''m',sprintf('%04
             d',t3),''');'));
```

```matlab
 96           eval(strcat('load(fullfile(flow),''-mat'',''f',sprintf('%04d
                 ',t3),''');'));
 97           eval(strcat('metabMulti = m',sprintf('%04d',t3),';'));
 98           eval(strcat('flowMulti = f',sprintf('%04d',t3),';'));
 99           eval(strcat('clear f', sprintf('%04d',t3),' m',sprintf('%04d
                 ',t3)))
100      else
101          metabMulti(region) = metab(t2);    %  region is hardcoded
                here
102          flowMulti(region) = flow(t2);
103      end
104
105      temperature(2,:,:,:) = squeeze(temperature(1,:,:,:)) + ...
106          dt./(tissue(:,:,:,5).*tissue(:,:,:,4)).* ...
107          ((averagedk/dx^2).*...
108          (circshift(squeeze(temperature(1,:,:,:)),[1 0 0])-2*squeeze
                (temperature(1,:,:,:))+circshift(squeeze(temperature
                (1,:,:,:)),[-1 0 0])+...  % shift along x
109           circshift(squeeze(temperature(1,:,:,:)),[0 1 0])-2*squeeze
                (temperature(1,:,:,:))+circshift(squeeze(temperature
                (1,:,:,:)),[0 -1 0])+...  % shift along y
110           circshift(squeeze(temperature(1,:,:,:)),[0 0 1])-2*squeeze
                (temperature(1,:,:,:))+circshift(squeeze(temperature
                (1,:,:,:)),[0 0 -1]))...  % shift along z
111            -(1/6000)*rhoblood*flowMulti.*tissue(:,:,:,7)*cblood.*(
                  squeeze(temperature(1,:,:,:))-bloodT)+metabMulti.*
                  tissue(:,:,:,3));
112      % resets the air temperature back since it's also modified
            above,
```

```matlab
113        % but it needs to be kept constant throughout the calculations
114        temperature(2,squeeze(tissue(:,:,:,1))==1) = airT;
115        temperatureOut(ceil(t2/savesteps),:,:,:) = temperature(2,:,:,:)
              ;
116        temperature(1,:,:,:) = temperature(2,:,:,:);  % moves 2 back to
              1
117        clear metabMulti flowMulti
118 end
119 close(statusbar);
120
121 % ============
122 % = Old Code =
123 % ============
124 % This is what used to be used. It's much slower (~60 times
125 % slower), but it's much easier to understand compared to the
126 % above code. If any changes need to be made above, first look
127 % through this code to ensure you understand it before making
128 % changes. It's reallyeasy to mess up the code above and nearly
129 % impossible to figure out where.
130 %
131 %   good luck.
132
133 % for t2 = 1:nt-1
134 %     for x2 = 2:xmax-1
135 %         for y2 = 2:ymax-1
136 %             for z2 = 2:zmax-1
137 %                 if tissue(x2,y2,z2,1) ~= 1,
138 %                     temperature(t2+1,x2,y2,z2) = temperature(t2,
       x2,y2,z2) + (dt/(tissue(x2,y2,z2,5)*tissue(x2,y2,z2,4)))*((tissue
```

```matlab
                            (x2,y2,z2,6)/dx^2)*...
%                                   (temperature(t2,x2+1,y2,z2)-2*temperature(
    t2,x2,y2,z2)+temperature(t2,x2-1,y2,z2)+...
%                                   temperature(t2,x2,y2+1,z2)-2*temperature(t2
    ,x2,y2,z2)+temperature(t2,x2,y2-1,z2)+...
%                                   temperature(t2,x2,y2,z2+1)-2*temperature(t2
    ,x2,y2,z2)+temperature(t2,x2,y2,z2-1))...
%                                   -(1/6000)*rhoBlood*wBlood*cBlood*(
    temperature(t2,x2,y2,z2)-bloodT)+tissue(x2,y2,z2,3));
%                       end
%                   end
%               end
%           end
% end

end
```

Appendix B   Visualization Tools

The temperature data is a four dimensional dataset (time, x, y and z), so good visualizations tools are necessary to analyzing the results. The primary tool I use is a modification of SliceBrowser (`http://www.mathworks.com/matlabcentral/fileexchange/20604`) and is provided as part of temptools (`https://github.com/greggroth/temptools/tree/master/lib/SliceBrowser`). In working with this, I also created a function (TempPlot()) to act as a wrapper and handle possible plotting situations depending on the number of inputs.

B.1.1 TempPlot()

```
function [ ] = TempPlot( head, tempdata, highlightRegion, slice,
    equil,threshold,point)
%TempPlot Plot data from tempCalc() or BulkImportNII()
%    INPUT TempPlot(structuredata)
%          TempPlot(structuredata,temperaturedata)
%          TempPlot(structuredata,temperaturedata,highlightRegion)
%          TempPlot(structuredata,temperaturedata,highlightRegion,
    slice)
%          TempPlot(structuredata,temperaturedata,highlightRegion,
    slice,EquillibriumData)
%
%   This function with determine which type of data it is and then
% plot it appropiately.
%
%   equil - Equillibrium state data
%   threshold - threshold value for being displayed as an overlay
%   REQUIRES:  SliceBrowser (http://www.mathworks.com/matlabcentral
    /fileexchange/20604)
```

```matlab
15  %%  Error checking and data restructuring where necessary
16  if ndims(head) == 4
17      head = head(:,:,:,1);
18  elseif ndims(head) ~= 3
19      error('Input ''head'' must have either 3 or 4 dimensions');
20  end
21
22  if nargin > 1
23      if ndims(tempdata) == 3  % should only happen when comparing
             two equilibrium datasets
24      temp = tempdata;
25      tempdata = zeros([1 size(temp)]);
26      tempdata(1,:,:,:) = temp;
27      elseif ndims(tempdata) ~= 4
28      error('Input ''tempdata'' must have either 3 or 4 dimensions');
29      end
30      tempdataShort = squeeze(tempdata(end,:,:,:));
31  end
32
33  if nargin > 2
34      if ndims(highlightRegion) ~= 3
35      error('Input ''highlightRegion'' must have 3 dimensions');
36      end
37      if size(highlightRegion) ~= size(head)
38      error('Input ''highlightRegion'' must be of the same size as ''
             head''');
39      end
40      tempdataShort = squeeze(tempdata(end,:,:,:));
41  end
```

```matlab
42
43  if nargin > 3
44      if slice > size(tempdata,1)
45      error('Input ''slice'' must be less or equal to the length of
           the first dimension of ''tempdata''');
46      end
47      tempdataShort = squeeze(tempdata(slice,:,:,:));
48  end
49
50  if nargin > 4
51      if ndims(equil) == 3
52          eq = equil;
53      elseif ndims(equil) == 4
54          eq = squeeze(equil(1,:,:,:));
55      else
56          error('Input ''equil'' must have either 3 or 4 dimensions')
            ;
57      end
58      clear 'equil';
59  end
60
61  %%  Pick how to format the call of SliceBrowser()
62  switch nargin
63      case 1
64      SliceBrowser(head,1,head);
65      colormap(gray);
66      case 2
67      %SliceBrowser(squeeze(tempdata(size(tempdata,1),:,:,:)),
           tempdata,head);
```

```
68        SliceBrowser ( tempdataShort , tempdata , head );
69        case 3
70        SliceBrowser ( tempdataShort , tempdata , head , highlightRegion );
71        case 4
72        SliceBrowser ( tempdataShort , tempdata , head , highlightRegion );
73        case 5
74        SliceBrowser ( tempdataShort - eq , tempdata , head , highlightRegion );
75        case 6
76        SliceBrowserOverlay ( tempdataShort - eq , tempdata , head ,
             highlightRegion , threshold );
77        case 7
78        imgoverlay ( head , tempdataShort - eq , point , threshold )
79    end
80
81    end
```

B.1.2 tsliceplot

This is a visualization tool I wrote that allows you to view the change in temperature versus time for a line passing through the head. Screenshots of the tool can be seen in Figs. B.1 and B.2.

Usage:

```
tsliceplot ( temperature_data ,  equilibrium_temperature_data )
```

The script is available as part of temptools (https://github.com/greggroth/temptools/tree/master/lib/tsliceplot).
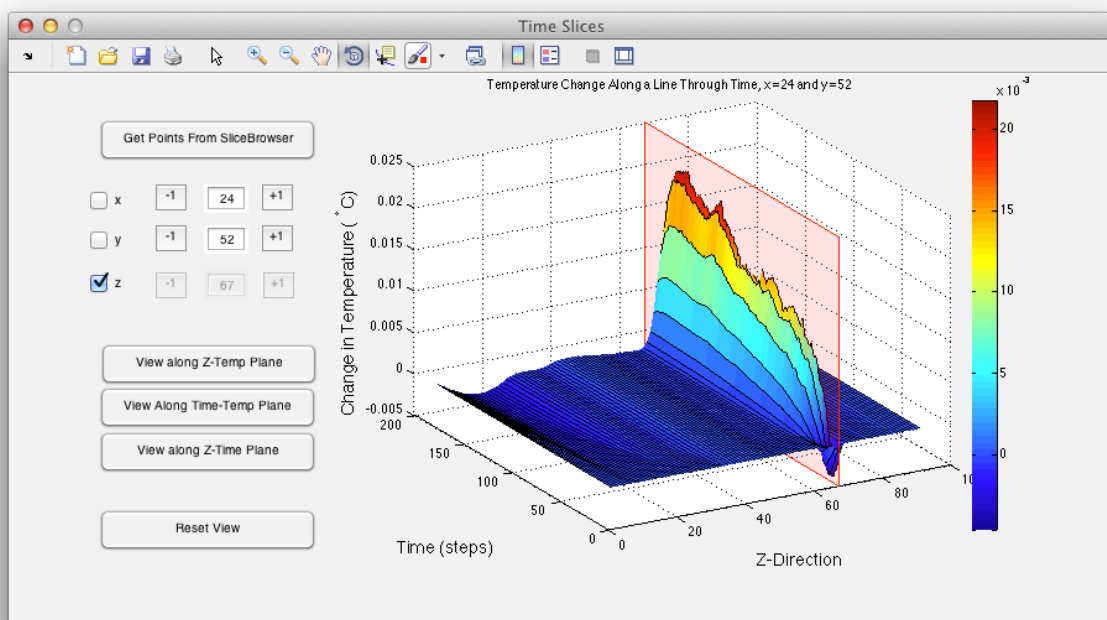
Figure B.1 Experimental data for activity in the motor cortex visualized with tsliceplot.
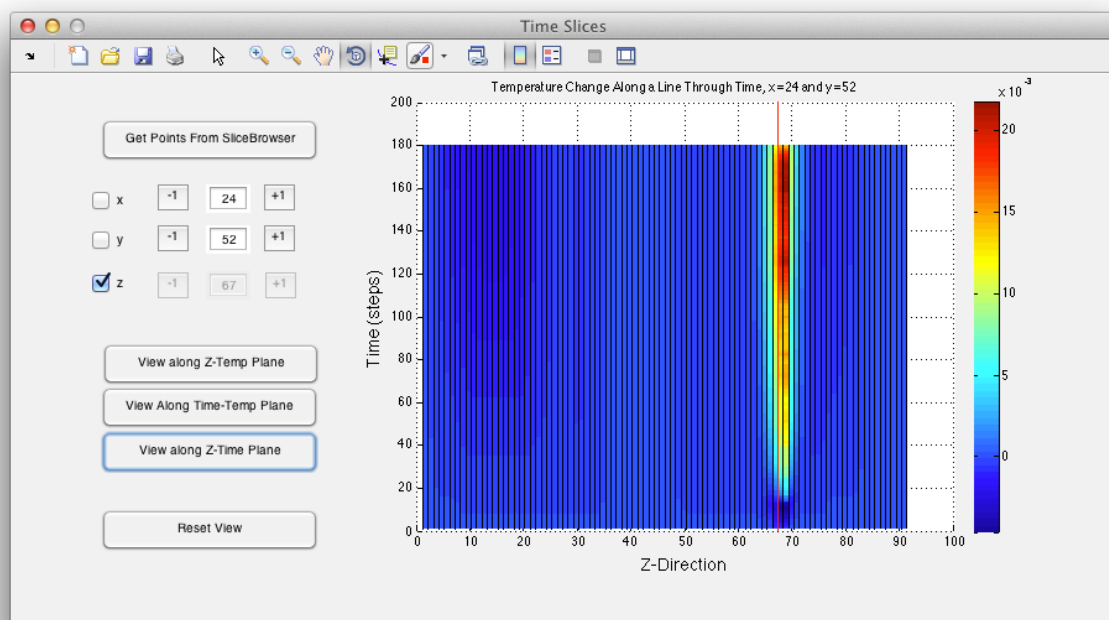
Figure B.2   The same data as is presented in Fig. B.1, but viewed flat-on along the z vs. time plane.