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

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, 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

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

I want to thank my advisors A. G. Unil Perera and Mukesh Dhamala for their guidance and leadership through my graduate school career. Likewise, I must thank everyone in Dr. Perera's and Dr. Dhamala's labs for always being helpful over the past couple of years. Thank you.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

BOLD	Blood Oxygen Level Dependent
fMRI	Functional Magnetic Resonance Imaging
fNIRS	Functional Near-Infrared Spectroscopy
NMR	Nuclear Magnetic Resonance
ROI	Region of Interest
CSF	Cerebral Spinal Fluid
OLM	Oxygen Limitation Model

1 Introduction

Since its invention in the 1950's [1] and later development in the 1970's [2], Magnetic Resonance Imaging (MRI) has allowed physicians and scientists a detailed view within the human body.

1.1 Models of the fMRI BOLD Response

This is just place holder text until I write the introduction (last step). Hopefully this is sufficient to demonstrate the format.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla ligula lorem, malesuada non feugiat a, bibendum sit amet leo. Sed in dui lacus. Nullam quam dolor, elementum ac semper vitae, lacinia et dui. Curabitur accumsan urna nec dolor porttitor mollis. Nunc id feugiat tellus. Proin orci arcu, egestas a tincidunt eu, luctus ut ligula. Curabitur at risus non nulla congue facilisis sed eu velit. Morbi eget bibendum sem.

Cras vel lectus leo. Donec at ante vel eros condimentum dignissim et at libero. Nullam vitae purus at diam semper gravida. Ut venenatis dapibus nulla pretium eleifend. Nam scelerisque dignissim augue, ac suscipit tellus vestibulum vel. Sed rutrum sollicitudin sodales. Pellentesque cursus lobortis neque, id ornare purus laoreet eget. Sed adipiscing sollicitudin convallis. Quisque eu orci sit amet libero mollis tincidunt. Nullam sed consectetur odio. Etiam a molestie orci.

Pellentesque sed purus odio. Nam pharetra aliquam augue vitae porta. Phasellus malesuada rhoncus tristique. Donec cursus, nibh vel eleifend bibendum, ante massa imperdiet mauris, vel varius dolor libero feugiat nibh. Sed libero sem, bibendum eu cursus sit amet,

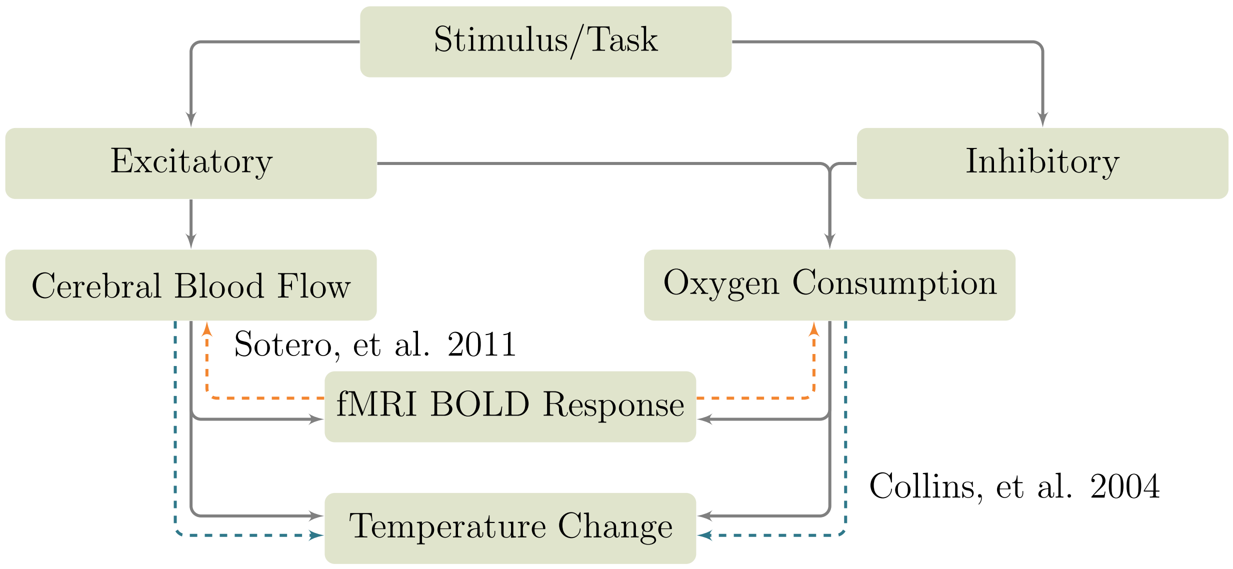


Figure 1.1 Generation of the fMRI BOLD response from changes in neuronal activity. Black arrows indicate a causal relationship while colored dashed-arrows indicate existing models for the relationship. The orange line (●) shows the model proposed by Sotero and Iturria-Medina [3] to calculate cerebral blood flow and metabolism and the blue line (●) shows how the model proposed by Collins et al. [4] is used to calculate temperature.

euismod in dui. Vestibulum massa nisl, accumsan eu cursus nec, pretium blandit lectus. Phasellus vitae ipsum eget tellus consequat molestie vel nec tortor. Curabitur blandit, lectus lacinia consectetur varius, leo metus placerat nisl, ullamcorper egestas nisl nisi in nulla.

Nunc sit amet mi sed ante dictum luctus eget et augue. Morbi vitae nunc justo. Praesent ac elit lorem. In hac habitasse platea dictumst. Integer pellentesque eros viverra justo scelerisque mollis. Vestibulum tempor tempus velit, et pulvinar odio iaculis ut. Quisque luctus massa sit amet justo ornare nec posuere eros aliquam. Vestibulum luctus pharetra augue, et varius ligula sollicitudin sit amet. Cras mi enim, laoreet quis vulputate ut, pulvinar vel nibh.

Cras sem justo, ultricies eget sodales eget, bibendum non neque. Pellentesque tempus mi eget massa venenatis id cursus sem aliquet. Nunc venenatis, est at porta faucibus, erat sem ultrices magna, eget tincidunt orci lorem vitae nisl. Sed sem velit, fermentum in hendrerit eu, tempor fermentum arcu. Nullam id nunc vel purus aliquet faucibus. Donec ullamcorper odio ac purus tristique mollis. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Pellentesque eget elit in orci iaculis congue. Suspendisse sit amet nisl a velit tempus pretium. Suspendisse quis vulputate nunc. Phasellus quis elit sit amet arcu faucibus adipiscing. Vestibulum faucibus enim ut libero venenatis ac facilisis dolor ultrices.

2 Calculating Temperature Changes using the fMRI BOLD Response

2.1 Introduction

Current efforts to model temperature changes be can categorized into two classes. The first class approaches the problem by considering a single voxel deep within the brain (single-voxel approach) while the second approach considers the brain and head as an entire system (multi-voxel approach). Each of these methods has their own pros and cons which will be discussed below.

2.1.1 Single-Voxel Approach

A single-voxel model of temperature was first proposed by SOMEONE, but has been refined over the past HOWLONG years CITEABUNCH to include more terms. Although different approaches consider different contributions to the temperature change, they all narrow the problem down to a single voxel which is usually 2mm x 2mm x 2mm. 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 possible accuracy of such a model when applied to a subject.

The most recently published iteration of a single-voxel model was published by Sotero and Iturria-Medina [3]. The basis of this model is a modification of the Penne's Bioheat

Equation [5, 3].

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

where C_t 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, $COMRO_2 \mid_0$ is the metabolic rate at rest, ρ_b is the blood density, C_b is the specific heat of blood, $CBF \mid_0$ 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 $\frac{dT(t)}{dt} = 0$ [3].

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

If the values provided in table 2.2 are substitued into eq. (2.2), a resting temperature of 37.3057°C is found. Since the resting temperature is always greater than the arterial blood temperature, it limits the ability of the model to account for all experimental results.

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

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

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 Approach

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 the a result of it accounting for a voxels 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). At the heart of our method is a three-dimensional implementation of the Pennes bioheat equation (eq. (2.4))[4].

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

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 (see section 2.2 for more on this).

2.2 Our Approach

The fundamental difference between our temperature modeling approach and the single-voxel models discussed in section 2.1 is that we consider the entire head. The Pennes bioheat

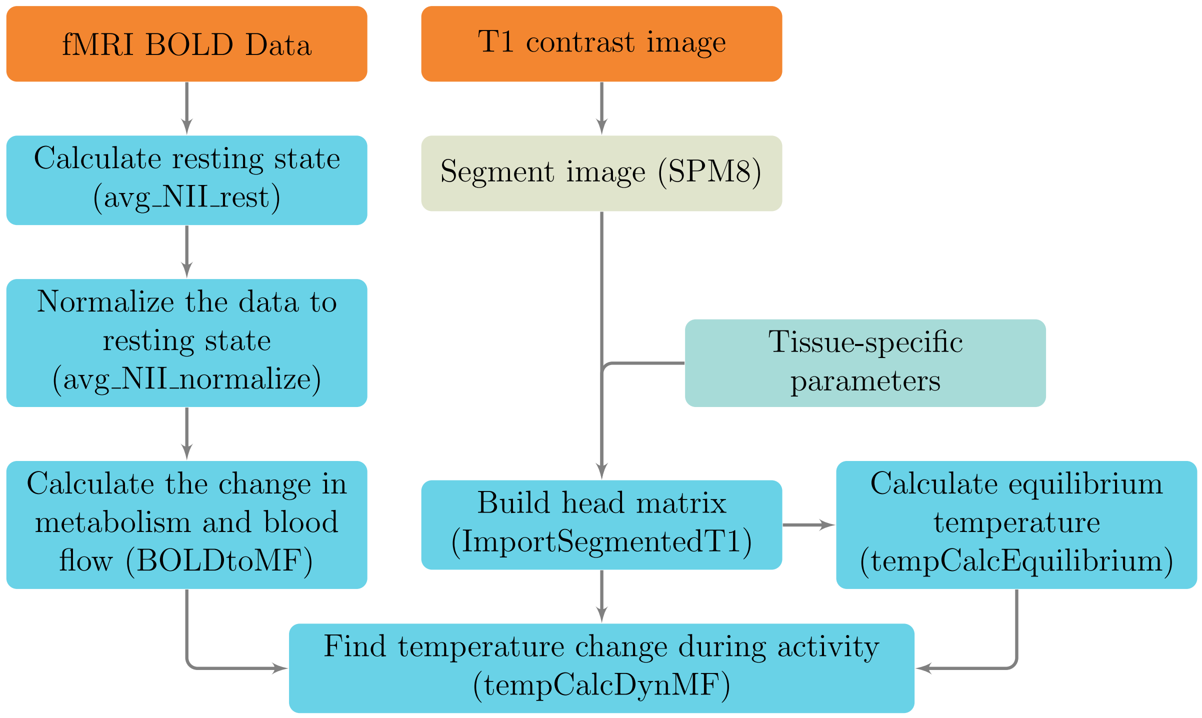

fMRI BOLD Data
T1 contrast image
Calculate resting state (avg_NII_rest)
Segment image (SPM8)
Normalize the data to resting state (avg_NII_normalize)
Tissue-specific parameters
Calculate the change in metabolism and blood flow (BOLDtoMF)
Build head matrix (ImportSegmentedT1)
Calculate equilibrium temperature (tempCalcEquilibrium)
Find temperature change during activity (tempCalcDynMF)

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).

equation (eq. (2.1)) [5, 3] includes three terms. The first and second terms describe heat generation by metabolism and heat exchange by convection to blood flow. On shorter time scales, these two terms dominate and are sufficient for determining the temperature change; however, the third term becomes important on longer time scales.

The third term describes the heat exchanged by conduction to surrounding tissues. This is a comparatively slow process, but on larger time scales determines the resting state temperature. When calculating the temperature change, it is important to first have an accurate resting state temperature. By considering the entire head, out model 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.

[WRITE SOMETHING HERE]

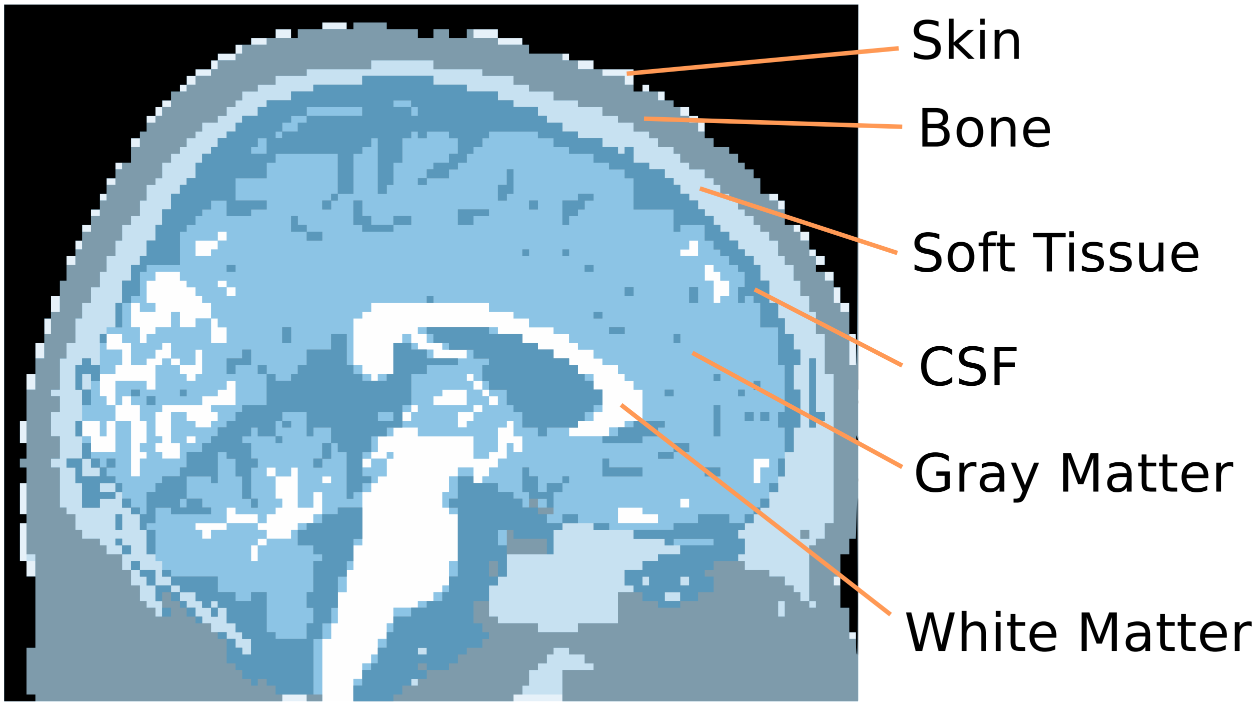


Figure 2.2 Slice of the segmented head. Each color represents a different tissue type.

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

Table 2.1 Tissue-specific parameters used to calculate the temperature change (from Collins et al. [4]).

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

air are more appropriately labeled as skin, so in total we are left with a model of the head 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.

2.2.2 Calculating the equilibrium temperature

The first step in calculating the temperature change is to first know what the resting state temperature is for each voxel within the head. Our approach was to have the initial temperature for all tissue voxels to be equal to 37°C and air voxels are kept at 24°C. The starting temperature of the tissue doesn't 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 the Pennes bioheat equation (eq. (2.4)) is used to update the temperature. 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 fMRI BOLD

This is the critical step where we use fMRI BOLD data to calculate the normalized change in metabolism and blood flow. The method used [3] is an assemblage of a couple other works [CITATION NEEDED]. It starts by using the relation between metabolism and blood flow proposed by Buxton et al. [6]:

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

Table 2.2 Parameters used to solve the single-voxel Penne's Bioheat Equation. (modified from Sotero and Iturria-Medina [3])

Parameter	Meaning	Value
T$_a$	Arterial blood temperature	37°C
C_{tissue}	Tissue Heat Capacity	3.664 J/(gK)
$\Delta H°$	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$_2$ $\|_0$	Cerebral metabolic rate of O$_2$ consumption at rest	0.026310^{-6} mol/(gs)
CBF$\|_0$	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	

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

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

in accordance with the oxygen limitation model [7]. Combining eq. (2.5) with eq. (2.6) yields

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

Sotero and Iturria-Medina [3] 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 [8, 9, 10]:

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

where values for a, b and c are provided in table 2.2. From this approximation we have the final form of metabolism:

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

As proposed by Davis et al. [11], the BOLD signal changes ($\frac{\Delta S(t)}{S_0}$) can be described in terms of $m(t)$ and $f(t)$:

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

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

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

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

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

which can be solved by using the Lambert W function

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

where z is given by

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

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 input BOLD data is stored in folder as a separate 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 element-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 these functions is available in appendix A.2.

2.2.4 Calculating the change in temperature in the active brain

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla ligula lorem, malesuada non feugiat a, bibendum sit amet leo. Sed in dui lacus. Nullam quam dolor, elementum ac semper vitae, lacinia et dui. Curabitur accumsan urna nec dolor porttitor mollis. Nunc id feugiat tellus. Proin orci arcu, egestas a tincidunt eu, luctus ut ligula. Curabitur at risus non nulla congue facilisis sed eu velit. Morbi eget bibendum sem.

Cras vel lectus leo. Donec at ante vel eros condimentum dignissim et at libero. Nullam vitae purus at diam semper gravida. Ut venenatis dapibus nulla pretium eleifend. Nam scelerisque dignissim augue, ac suscipit tellus vestibulum vel. Sed rutrum sollicitudin sodales. Pellentesque cursus lobortis neque, id ornare purus laoreet eget. Sed adipiscing sollicitudin convallis. Quisque eu orci sit amet libero mollis tincidunt. Nullam sed consectetur odio. Etiam a molestie orci.

Pellentesque sed purus odio. Nam pharetra aliquam augue vitae porta. Phasellus malesuada rhoncus tristique. Donec cursus, nibh vel eleifend bibendum, ante massa imperdiet mauris, vel varius dolor libero feugiat nibh. Sed libero sem, bibendum eu cursus sit amet, euismod in dui. Vestibulum massa nisl, accumsan eu cursus nec, pretium blandit lectus. Phasellus vitae ipsum eget tellus consequat molestie vel nec tortor. Curabitur blandit, lectus lacinia consectetur varius, leo metus placerat nisl, ullamcorper egestas nisl nisi in nulla.

Nunc sit amet mi sed ante dictum luctus eget et augue. Morbi vitae nunc justo. Praesent ac elit lorem. In hac habitasse platea dictumst. Integer pellentesque eros viverra justo scelerisque mollis. Vestibulum tempor tempus velit, et pulvinar odio iaculis ut. Quisque luctus massa sit amet justo ornare nec posuere eros aliquam. Vestibulum luctus pharetra augue, et varius ligula sollicitudin sit amet. Cras mi enim, laoreet quis vulputate ut, pulvinar vel nibh.

Cras sem justo, ultricies eget sodales eget, bibendum non neque. Pellentesque tempus mi eget massa venenatis id cursus sem aliquet. Nunc venenatis, est at porta faucibus, erat sem ultrices magna, eget tincidunt orci lorem vitae nisl. Sed sem velit, fermentum in hendrerit eu, tempor fermentum arcu. Nullam id nunc vel purus aliquet faucibus. Donec ullamcorper odio ac purus tristique mollis. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Pellentesque eget elit in orci iaculis congue. Suspendisse sit amet nisl a velit tempus pretium. Suspendisse quis vulputate nunc. Phasellus quis elit sit amet arcu faucibus adipiscing. Vestibulum faucibus enim ut libero venenatis ac facilisis dolor ultrices.

2.3 Results

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla ligula lorem, malesuada non feugiat a, bibendum sit amet leo. Sed in dui lacus. Nullam quam dolor, elementum ac semper vitae, lacinia et dui. Curabitur accumsan urna nec dolor porttitor mollis. Nunc id feugiat tellus. Proin orci arcu, egestas a tincidunt eu, luctus ut ligula. Curabitur at risus non nulla congue facilisis sed eu velit. Morbi eget bibendum sem.

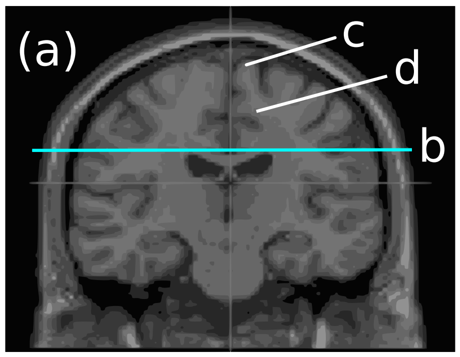
2.3.1 Using Theoretical BOLD Data

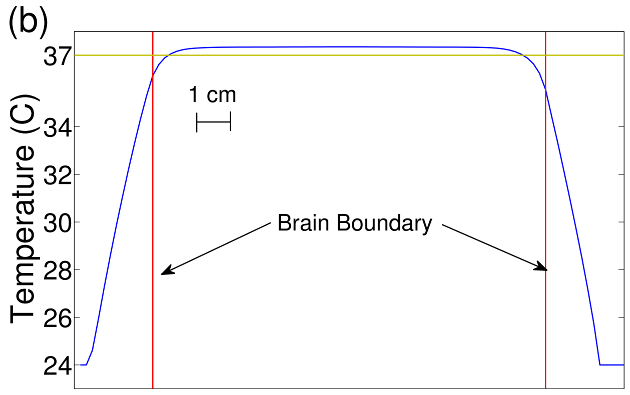
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla ligula lorem, malesuada non feugiat a, bibendum sit amet leo. Sed in dui lacus. Nullam quam dolor, elementum ac semper vitae, lacinia et dui. Curabitur accumsan urna nec dolor porttitor mollis. Nunc id feugiat tellus. Proin orci arcu, egestas a tincidunt eu, luctus ut ligula. Curabitur at risus non nulla congue facilisis sed eu velit. Morbi eget bibendum sem.

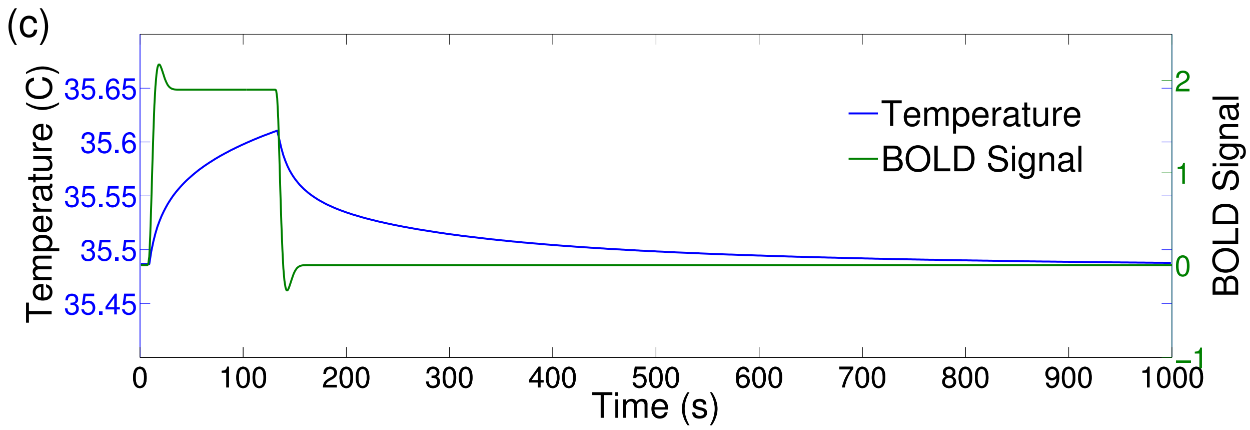
Cras vel lectus leo. Donec at ante vel eros condimentum dignissim et at libero. Nullam vitae purus at diam semper gravida. Ut venenatis dapibus nulla pretium eleifend. Nam scelerisque dignissim augue, ac suscipit tellus vestibulum vel. Sed rutrum sollicitudin sodales. Pellentesque cursus lobortis neque, id ornare purus laoreet eget. Sed adipiscing sollicitudin convallis. Quisque eu orci sit amet libero mollis tincidunt. Nullam sed consectetur odio. Etiam a molestie orci.

Pellentesque sed purus odio. Nam pharetra aliquam augue vitae porta. Phasellus malesuada rhoncus tristique. Donec cursus, nibh vel eleifend bibendum, ante massa imperdiet mauris, vel varius dolor libero feugiat nibh. Sed libero sem, bibendum eu cursus sit amet, euismod in dui. Vestibulum massa nisl, accumsan eu cursus nec, pretium blandit lectus. Phasellus vitae ipsum eget tellus consequat molestie vel nec tortor. Curabitur blandit, lectus lacinia consectetur varius, leo metus placerat nisl, ullamcorper egestas nisl nisi in nulla.

Nunc sit amet mi sed ante dictum luctus eget et augue. Morbi vitae nunc justo. Praesent







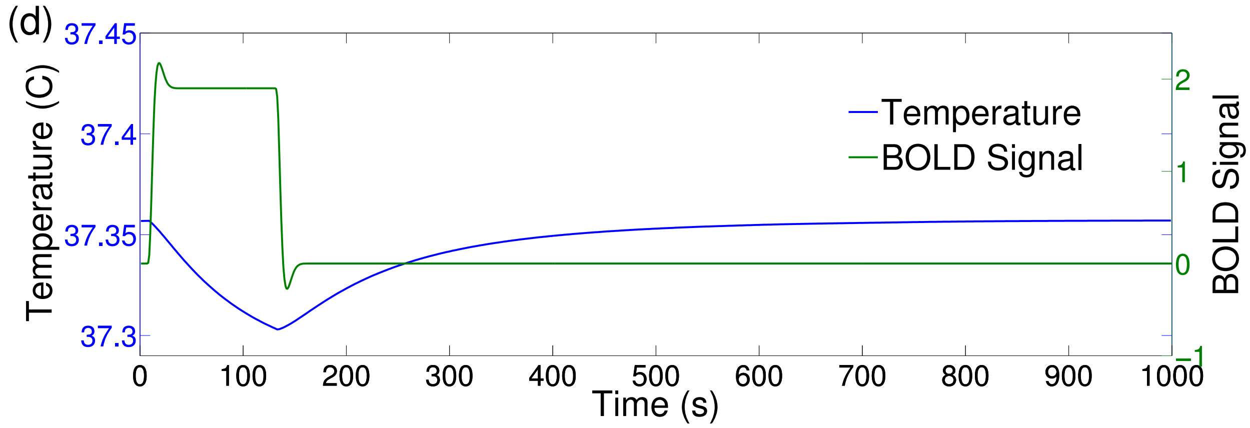


Figure 2.3 Temperature changes using simulated BOLD signals. (a) Slice of the head (y = -12) with indicators of the locations for parts (b)-(d). (b) Equilibrium temperature along a line through the head. Red lines indicate the brain boundary and the gold line indicates the blood temperature (37°C) used 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.

ac elit lorem. In hac habitasse platea dictumst. Integer pellentesque eros viverra justo scelerisque mollis. Vestibulum tempor tempus velit, et pulvinar odio iaculis ut. Quisque luctus massa sit amet justo ornare nec posuere eros aliquam. Vestibulum luctus pharetra augue, et varius ligula sollicitudin sit amet. Cras mi enim, laoreet quis vulputate ut, pulvinar vel nibh.

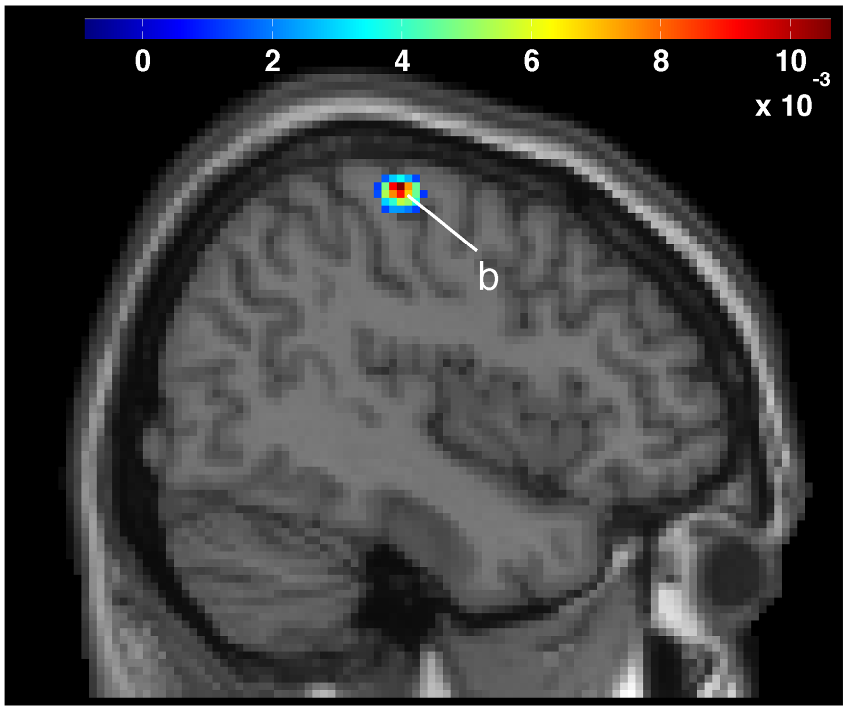
Cras sem justo, ultricies eget sodales eget, bibendum non neque. Pellentesque tempus mi eget massa venenatis id cursus sem aliquet. Nunc venenatis, est at porta faucibus, erat sem ultrices magna, eget tincidunt orci lorem vitae nisl. Sed sem velit, fermentum in hendrerit eu, tempor fermentum arcu. Nullam id nunc vel purus aliquet faucibus. Donec ullamcorper odio ac purus tristique mollis. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Pellentesque eget elit in orci iaculis congue. Suspendisse sit amet nisl a velit tempus pretium. Suspendisse quis vulputate nunc. Phasellus quis elit sit amet arcu faucibus adipiscing. Vestibulum faucibus enim ut libero venenatis ac facilisis dolor ultrices.

2.3.2 Using Experimental BOLD Data

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla ligula lorem, malesuada non feugiat a, bibendum sit amet leo. Sed in dui lacus. Nullam quam dolor, elementum ac semper vitae, lacinia et dui. Curabitur accumsan urna nec dolor porttitor mollis. Nunc id feugiat tellus. Proin orci arcu, egestas a tincidunt eu, luctus ut ligula. Curabitur at risus non nulla congue facilisis sed eu velit. Morbi eget bibendum sem.

Cras vel lectus leo. Donec at ante vel eros condimentum dignissim et at libero. Nullam vitae purus at diam semper gravida. Ut venenatis dapibus nulla pretium eleifend. Nam scelerisque dignissim augue, ac suscipit tellus vestibulum vel. Sed rutrum sollicitudin sodales. Pellentesque cursus lobortis neque, id ornare purus laoreet eget. Sed adipiscing sollicitudin convallis. Quisque eu orci sit amet libero mollis tincidunt. Nullam sed consectetur odio. Etiam a molestie orci.

(a)



(b)

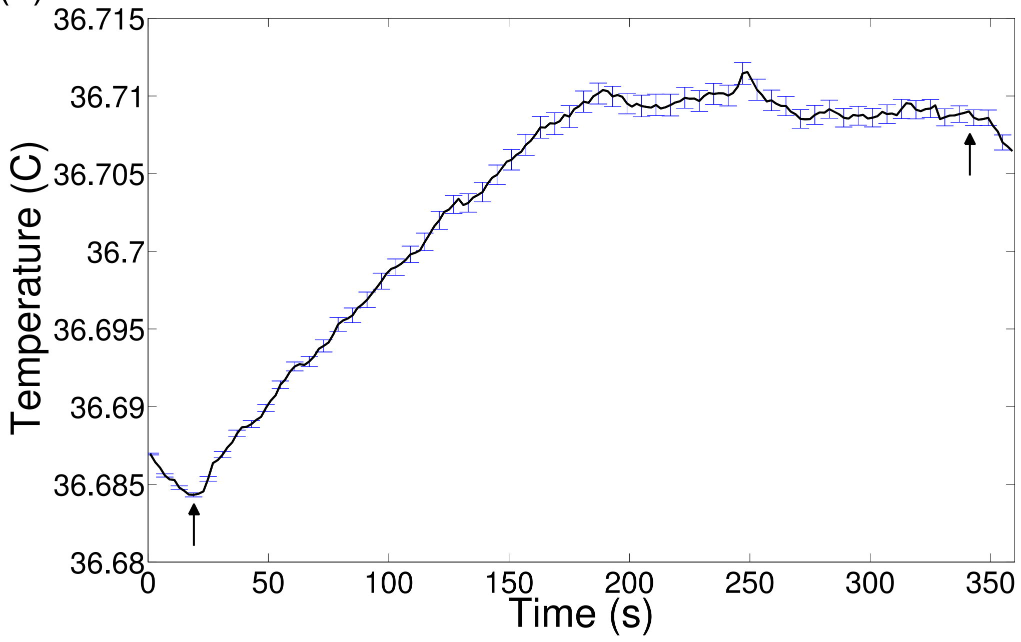


Figure 2.4 Temperature calculated from a voxel within the motor cortex. (a) A slice (x = -44) showing the motor cortex warming during a finger-tapping task. (b) Temperature at a voxel within the motor cortex (-44, -24, 60) with standard error indicated by blue error bars (Arrows indicate task onset and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

Pellentesque sed purus odio. Nam pharetra aliquam augue vitae porta. Phasellus malesuada rhoncus tristique. Donec cursus, nibh vel eleifend bibendum, ante massa imperdiet mauris, vel varius dolor libero feugiat nibh. Sed libero sem, bibendum eu cursus sit amet, euismod in dui. Vestibulum massa nisl, accumsan eu cursus nec, pretium blandit lectus. Phasellus vitae ipsum eget tellus consequat molestie vel nec tortor. Curabitur blandit, lectus lacinia consectetur varius, leo metus placerat nisl, ullamcorper egestas nisl nisi in nulla.

Nunc sit amet mi sed ante dictum luctus eget et augue. Morbi vitae nunc justo. Praesent ac elit lorem. In hac habitasse platea dictumst. Integer pellentesque eros viverra justo scelerisque mollis. Vestibulum tempor tempus velit, et pulvinar odio iaculis ut. Quisque luctus massa sit amet justo ornare nec posuere eros aliquam. Vestibulum luctus pharetra augue, et varius ligula sollicitudin sit amet. Cras mi enim, laoreet quis vulputate ut, pulvinar vel nibh.

Cras sem justo, ultricies eget sodales eget, bibendum non neque. Pellentesque tempus mi eget massa venenatis id cursus sem aliquet. Nunc venenatis, est at porta faucibus, erat sem ultrices magna, eget tincidunt orci lorem vitae nisl. Sed sem velit, fermentum in hendrerit eu, tempor fermentum arcu. Nullam id nunc vel purus aliquet faucibus. Donec ullamcorper odio ac purus tristique mollis. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Pellentesque eget elit in orci iaculis congue. Suspendisse sit amet nisl a velit tempus pretium. Suspendisse quis vulputate nunc. Phasellus quis elit sit amet arcu faucibus adipiscing. Vestibulum faucibus enim ut libero venenatis ac facilisis dolor ultrices.

3 Detector Applications to measuring the active brain

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla ligula lorem, malesuada non feugiat a, bibendum sit amet leo. Sed in dui lacus. Nullam quam dolor, elementum ac semper vitae, lacinia et dui. Curabitur accumsan urna nec dolor porttitor mollis. Nunc id feugiat tellus. Proin orci arcu, egestas a tincidunt eu, luctus ut ligula. Curabitur at risus non nulla congue facilisis sed eu velit. Morbi eget bibendum sem.

Cras vel lectus leo. Donec at ante vel eros condimentum dignissim et at libero. Nullam vitae purus at diam semper gravida. Ut venenatis dapibus nulla pretium eleifend. Nam scelerisque dignissim augue, ac suscipit tellus vestibulum vel. Sed rutrum sollicitudin sodales. Pellentesque cursus lobortis neque, id ornare purus laoreet eget. Sed adipiscing sollicitudin convallis. Quisque eu orci sit amet libero mollis tincidunt. Nullam sed consectetur odio. Etiam a molestie orci.

3.1 Functional Near-Infrared fNIR Imaging

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla ligula lorem, malesuada non feugiat a, bibendum sit amet leo. Sed in dui lacus. Nullam quam dolor, elementum ac semper vitae, lacinia et dui. Curabitur accumsan urna nec dolor porttitor mollis. Nunc id feugiat tellus. Proin orci arcu, egestas a tincidunt eu, luctus ut ligula. Curabitur at risus non nulla congue facilisis sed eu velit. Morbi eget bibendum sem.

Cras vel lectus leo. Donec at ante vel eros condimentum dignissim et at libero. Nullam vitae purus at diam semper gravida. Ut venenatis dapibus nulla pretium eleifend. Nam scelerisque dignissim augue, ac suscipit tellus vestibulum vel. Sed rutrum sollicitudin sodales.

Pellentesque cursus lobortis neque, id ornare purus laoreet eget. Sed adipiscing sollicitudin convallis. Quisque eu orci sit amet libero mollis tincidunt. Nullam sed consectetur odio. Etiam a molestie orci.

Pellentesque sed purus odio. Nam pharetra aliquam augue vitae porta. Phasellus malesuada rhoncus tristique. Donec cursus, nibh vel eleifend bibendum, ante massa imperdiet mauris, vel varius dolor libero feugiat nibh. Sed libero sem, bibendum eu cursus sit amet, euismod in dui. Vestibulum massa nisl, accumsan eu cursus nec, pretium blandit lectus. Phasellus vitae ipsum eget tellus consequat molestie vel nec tortor. Curabitur blandit, lectus lacinia consectetur varius, leo metus placerat nisl, ullamcorper egestas nisl nisi in nulla.

Nunc sit amet mi sed ante dictum luctus eget et augue. Morbi vitae nunc justo. Praesent ac elit lorem. In hac habitasse platea dictumst. Integer pellentesque eros viverra justo scelerisque mollis. Vestibulum tempor tempus velit, et pulvinar odio iaculis ut. Quisque luctus massa sit amet justo ornare nec posuere eros aliquam. Vestibulum luctus pharetra augue, et varius ligula sollicitudin sit amet. Cras mi enim, laoreet quis vulputate ut, pulvinar vel nibh.

$$I = I_0 e^{-\alpha x} \tag{3.1}$$

Cras sem justo, ultricies eget sodales eget, bibendum non neque. Pellentesque tempus mi eget massa venenatis id cursus sem aliquet. Nunc venenatis, est at porta faucibus, erat sem ultrices magna, eget tincidunt orci lorem vitae nisl. Sed sem velit, fermentum in hendrerit eu, tempor fermentum arcu. Nullam id nunc vel purus aliquet faucibus. Donec ullamcorper odio ac purus tristique mollis. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Pellentesque eget elit in orci iaculis congue. Suspendisse sit amet nisl a velit tempus pretium. Suspendisse quis vulputate nunc. Phasellus quis elit sit amet arcu faucibus adipiscing. Vestibulum faucibus enim ut libero venenatis ac facilisis dolor ultrices.

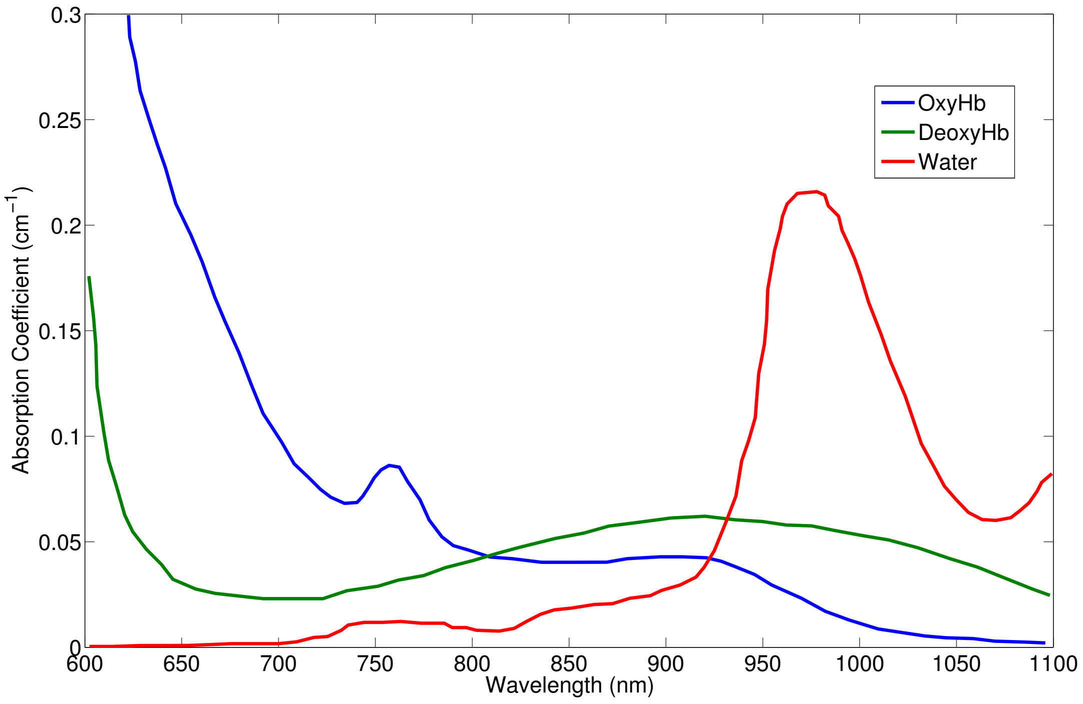


Figure 3.1 Absorption spectra of water, Hb and Dhb. From Cope [12] and HB stuff from Horecker [13]

3.2 Temperature Measurements

From the Beer-Lambert law eq. (3.1), the penetration depth, δ_p can be expressed as

$$\delta = \frac{1}{\alpha} \tag{3.2}$$

where α is the absorption coefficient. At body temperature (37°) the peak wavelength in the blackbody spectrum is approximately BLA. For water at this wavelength, α is approximately HUGE, so δ is VERY SMALL.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla ligula lorem, malesuada non feugiat a, bibendum sit amet leo. Sed in dui lacus. Nullam quam dolor, elementum ac semper vitae, lacinia et dui. Curabitur accumsan urna nec dolor porttitor mollis. Nunc id feugiat tellus. Proin orci arcu, egestas a tincidunt eu, luctus ut ligula. Curabitur at risus non nulla congue facilisis sed eu velit. Morbi eget bibendum sem.

Cras vel lectus leo. Donec at ante vel eros condimentum dignissim et at libero. Nullam vitae purus at diam semper gravida. Ut venenatis dapibus nulla pretium eleifend. Nam scelerisque dignissim augue, ac suscipit tellus vestibulum vel. Sed rutrum sollicitudin sodales. Pellentesque cursus lobortis neque, id ornare purus laoreet eget. Sed adipiscing sollicitudin convallis. Quisque eu orci sit amet libero mollis tincidunt. Nullam sed consectetur odio. Etiam a molestie orci.

Pellentesque sed purus odio. Nam pharetra aliquam augue vitae porta. Phasellus malesuada rhoncus tristique. Donec cursus, nibh vel eleifend bibendum, ante massa imperdiet mauris, vel varius dolor libero feugiat nibh. Sed libero sem, bibendum eu cursus sit amet, euismod in dui. Vestibulum massa nisl, accumsan eu cursus nec, pretium blandit lectus. Phasellus vitae ipsum eget tellus consequat molestie vel nec tortor. Curabitur blandit, lectus lacinia consectetur varius, leo metus placerat nisl, ullamcorper egestas nisl nisi in nulla.

Nunc sit amet mi sed ante dictum luctus eget et augue. Morbi vitae nunc justo. Praesent ac elit lorem. In hac habitasse platea dictumst. Integer pellentesque eros viverra justo scelerisque mollis. Vestibulum tempor tempus velit, et pulvinar odio iaculis ut. Quisque luctus massa sit amet justo ornare nec posuere eros aliquam. Vestibulum luctus pharetra augue, et varius ligula sollicitudin sit amet. Cras mi enim, laoreet quis vulputate ut, pulvinar vel nibh.

Cras sem justo, ultricies eget sodales eget, bibendum non neque. Pellentesque tempus mi eget massa venenatis id cursus sem aliquet. Nunc venenatis, est at porta faucibus, erat sem ultrices magna, eget tincidunt orci lorem vitae nisl. Sed sem velit, fermentum in hendrerit eu, tempor fermentum arcu. Nullam id nunc vel purus aliquet faucibus. Donec ullamcorper odio ac purus tristique mollis. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Pellentesque eget elit in orci iaculis congue. Suspendisse sit amet nisl a velit tempus pretium. Suspendisse quis vulputate nunc. Phasellus quis elit sit amet arcu faucibus adipiscing. Vestibulum faucibus enim ut libero venenatis ac facilisis dolor ultrices.

4 Conclusion

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla ligula lorem, malesuada non feugiat a, bibendum sit amet leo. Sed in dui lacus. Nullam quam dolor, elementum ac semper vitae, lacinia et dui. Curabitur accumsan urna nec dolor porttitor mollis. Nunc id feugiat tellus. Proin orci arcu, egestas a tincidunt eu, luctus ut ligula. Curabitur at risus non nulla congue facilisis sed eu velit. Morbi eget bibendum sem.

Cras vel lectus leo. Donec at ante vel eros condimentum dignissim et at libero. Nullam vitae purus at diam semper gravida. Ut venenatis dapibus nulla pretium eleifend. Nam scelerisque dignissim augue, ac suscipit tellus vestibulum vel. Sed rutrum sollicitudin sodales. Pellentesque cursus lobortis neque, id ornare purus laoreet eget. Sed adipiscing sollicitudin convallis. Quisque eu orci sit amet libero mollis tincidunt. Nullam sed consectetur odio. Etiam a molestie orci.

Pellentesque sed purus odio. Nam pharetra aliquam augue vitae porta. Phasellus malesuada rhoncus tristique. Donec cursus, nibh vel eleifend bibendum, ante massa imperdiet mauris, vel varius dolor libero feugiat nibh. Sed libero sem, bibendum eu cursus sit amet, euismod in dui. Vestibulum massa nisl, accumsan eu cursus nec, pretium blandit lectus. Phasellus vitae ipsum eget tellus consequat molestie vel nec tortor. Curabitur blandit, lectus lacinia consectetur varius, leo metus placerat nisl, ullamcorper egestas nisl nisi in nulla.

Nunc sit amet mi sed ante dictum luctus eget et augue. Morbi vitae nunc justo. Praesent ac elit lorem. In hac habitasse platea dictumst. Integer pellentesque eros viverra justo scelerisque mollis. Vestibulum tempor tempus velit, et pulvinar odio iaculis ut. Quisque luctus massa sit amet justo ornare nec posuere eros aliquam. Vestibulum luctus pharetra augue, et varius ligula sollicitudin sit amet. Cras mi enim, laoreet quis vulputate ut, pulvinar vel nibh.

Cras sem justo, ultricies eget sodales eget, bibendum non neque. Pellentesque tempus mi eget massa venenatis id cursus sem aliquet. Nunc venenatis, est at porta faucibus, erat sem ultrices magna, eget tincidunt orci lorem vitae nisl. Sed sem velit, fermentum in hendrerit eu, tempor fermentum arcu. Nullam id nunc vel purus aliquet faucibus. Donec ullamcorper odio ac purus tristique mollis. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Pellentesque eget elit in orci iaculis congue. Suspendisse sit amet nisl a velit tempus pretium. Suspendisse quis vulputate nunc. Phasellus quis elit sit amet arcu faucibus adipiscing. Vestibulum faucibus enim ut libero venenatis ac facilisis dolor ultrices.

References

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

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

[3] 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.

[4] 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.

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

[6] 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.

[7] 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.

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

[9] 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 cmro2. *Journal of Cerebral Blood Flow & Metabolism*, 30(2):311–322, Sep 2009.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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 to 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.

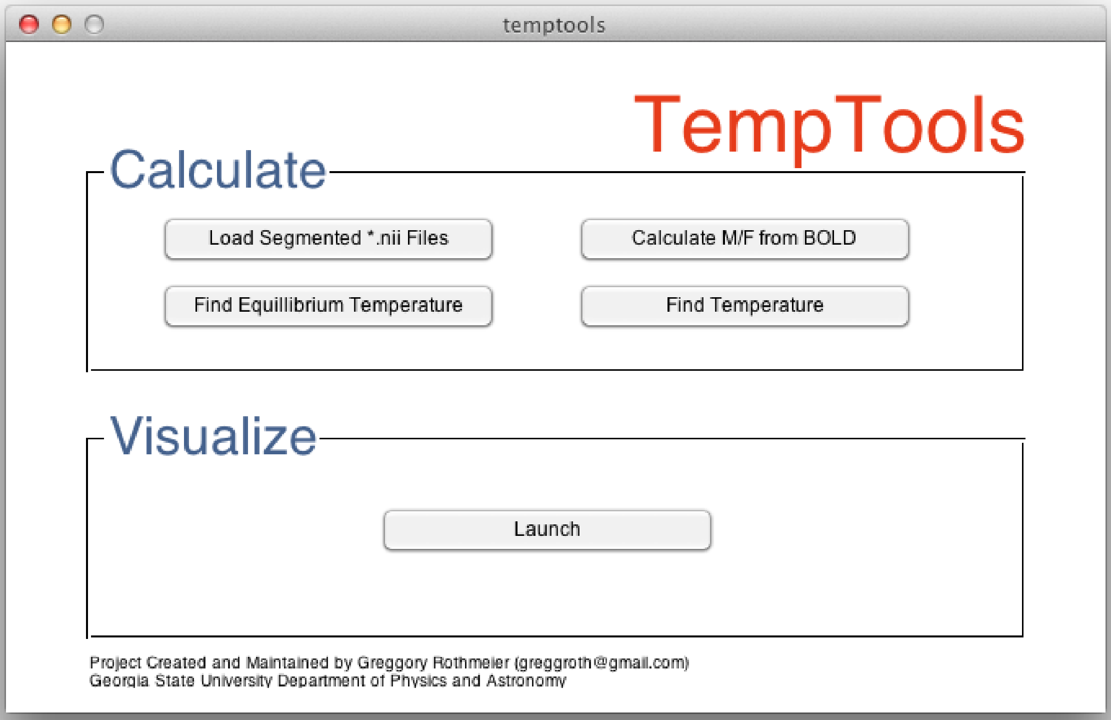
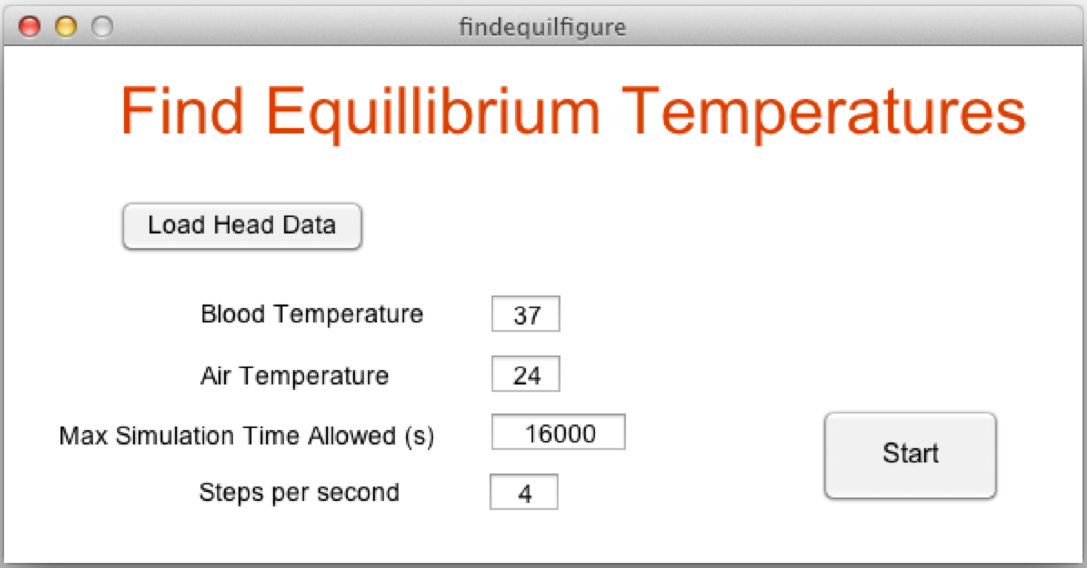

temptools
TempTools
Calculate
Load Segmented *.nii Files
Calculate M/F from BOLD
Find Equillibrium Temperature
Find Temperature
Visualize
Launch
Project Created and Maintained by Greggory Rothmeier (greggroth@gmail.com)
Georgia State University Department of Physics and Astronomy

Figure A.1 The main window of temptools. From here, you can go through the calculation steps and launch the visualization tool.



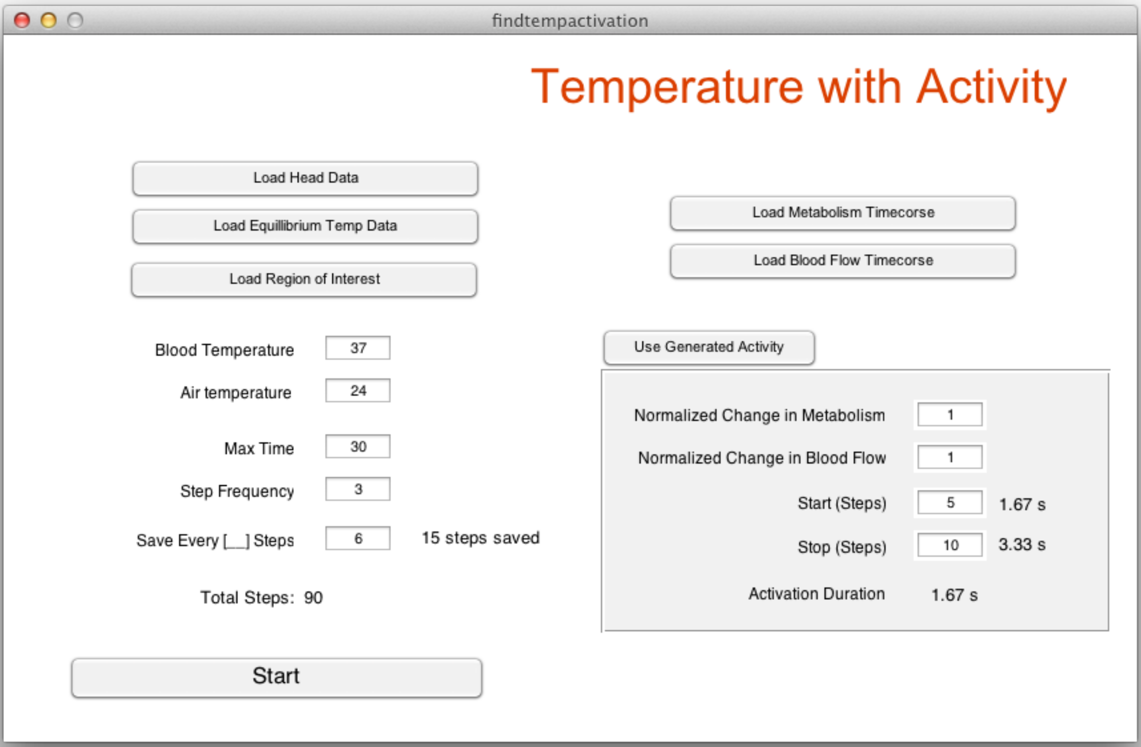
Blood Temperature 37

Air Temperature 24

Max Simulation Time Allowed (s) 16000

Steps per second 4

Figure A.2 This is the interface for calculating the equilibrium temperature (method explained in appendix A.3) under certain conditions.


findtempactivation

Temperature with Activity

Blood Temperature 37
Air temperature 24
Max Time 30
Step Frequency 3
Save Every [__] Steps 6 15 steps saved
Total Steps: 90

Normalized Change in Metabolism 1
Normalized Change in Blood Flow 1
Start (Steps) 5 1.67 s
Stop (Steps) 10 3.33 s
Activation Duration 1.67 s

Load Head Data
Load Equillibrium Temp Data
Load Region of Interest
Load Metabolism Timecorse
Load Blood Flow Timecorse
Use Generated Activity
Start

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.

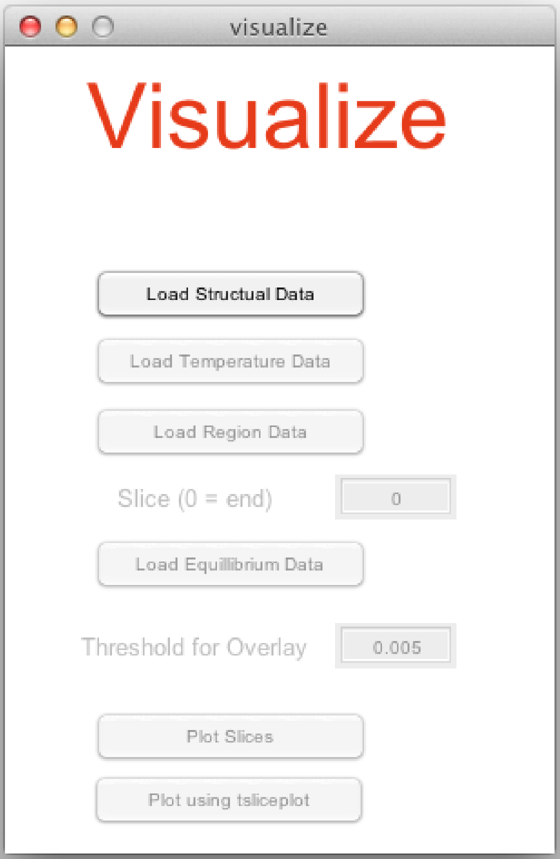


Figure A.4 Visualize your data using the temptools visualization window. This loads all of the required data and launches a slice browser or tsliceplot (see appendix B for more details).

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()

```
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});
end


% ====================
% = Tissue Parameters =
% ====================
% Each tissue type is assigned an integer index (i.e. gray matter
  -> 11)
% such that tissue-specific parameters can be found by looking at
  that
% element within the corresponding storage matrix
% (i.e. QmSTORE(11) -> gray matter Qm)

% Parameters taken from Colins, 2004

tisorder = [11 15 5 13 3];  %  Using:  [GM WM CSF Muscle Bone]

QmSTORE = [0 0 26.1 11600 0 26.1 697 0 0 302 15575 0 697 1100
  5192];
cSTORE = [1006 4600 2110 3640 3800 1300 3720 3000 4200 2300 3680
  3500 3720 3150 3600];
rhoSTORE = [1.3 1057 1080 1035.5 1007 1850 1126 1076 1009 916
  1035.5 1151 1041 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 0.4975 .342 .503];
wSTORE = [0 1000 3 45.2 0 1.35 40 0 0 2.8 67.1 3.8 3.8 12 23.7];

% ====================================
% = Import the pre-segmented T1 files =
% =====================================
%  The T1 contrast image sould be segmented using SPM8.
%  This loop needs to complete before the next one can begin
%  Import all of the datat and store as 'cdat1','cdat2', etc.
for i = 1:5
    eval(strcat('dat',num2str(i),' = loadNII(''rc', num2str(i), '
      single_subj_T1.nii'');'))
    %  Preallocate
    eval(strcat('out', num2str(i),' = zeros(cat(2,size(dat',
      num2str(i),'),7));'))
end

% ============================
% = Populate the head matrix =
% ============================
%  For each data file, it fills in the data from the data storage
   arrays
%  for that particular type of tissue.  It picks which ever tissue
   is
%  the most likely candidate for that voxel based on the segmented
   data

%   PROBLEM:  It returns 0 (later filled with air) if there is
   equal
%   probability of a voxel being two or more different types of
   tissue.
%   SOLVED BY fillHoles()

```

```

for i = 1:5
    %  Preallocate
    holder = zeros(cat(2,size(dat1),7),'single');
    mask = zeros(size(dat1));
    final = zeros(size(holder),'single');

    %  Create a mask that indicates whether it is the mostly likely
       tissue type
    guide = [1 2 3 4 5 1 2 3 4 5];  %  This guides it through the
      data correctly
    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);'))
    %  move structure data to new matrix
    holder(:,:,:,1) = mask;
    %  get indicies of tissues
    a = find(holder(:,:,:,1) == 1);
    %  gets coordinates from index
    [x y z t] = ind2sub(size(holder),a);

    %  go to each tissue point and store the info
    for j = 1:length(a)
        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))];
    end

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

    clearvars a x y z t holder final;
    waitbar(i/6,statusbar,sprintf(['File ',num2str(i),' Import
      Compete']));
end

% The filleAir() function checks for any voxels which were not
  assigned a
% tissue type and fills them in with air
almostthere = fillAir(out1+out2+out3+out4+out5);
% The fillHoles() function corrects for a voxel having two
% equally-probable tissue types
total = single(buildskin(fillHoles(dat1,dat2,dat3,dat4,dat5,
  almostthere)));
waitbar(1,statusbar,'Saving Data')

cd(oldFolder);
close(statusbar);

end
```

A.1.2 fillAir()

```
function [ output ] = fillAir( tissue )
% fillAir() fills gaps in data with air
%   Once you import all of the data using loadNII(), run it thought
    this to
```

```
%   fill in the remaining spaces with air.

airdata = [1 0 0 1006 1.3 0.026 0];

%  Picks out air spots
a = find(tissue(:,:,:,1) == 0);
[x y z t] = ind2sub(size(tissue),a);

for i = 1:length(a)
    tissue(x(i),y(i),z(i),:) = airdata;
end

output = tissue;

end
```

A.1.3 fillHoles()

```
function [ out_head ] = fillHoles( in1,in2,in3,in4,in5,headin)
% fillHoles() checks for misassigned voxels
%
% Solves an issue where a voxel with two equally probable tissue
% types resulted in being assigned as air.  This checks for air
% voxels that are surrounded by tissue and decides a tissue it
% it would be best suited as

head = squeeze(headin(:,:,:,1));  %  I only need the tissue indices
    so this makes things easier down the line

%%  Data Storage
```

```
QmSTORE = [0 0 26.1 11600 0 26.1 697 0 0 302 15575 0 697 1100
  5192];
cSTORE = [1006 4600 2110 3640 3800 1300 3720 3000 4200 2300 3680
  3500 3720 3150 3600];
rhoSTORE = [1.3 1057 1080 1035.5 1007 1850 1126 1076 1009 916
  1035.5 1151 1041 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 0.4975 .342 .503];
wSTORE = [0 1000 3 45.2 0 1.35 40 0 0 2.8 67.1 3.8 3.8 12 23.7];

%%  Get locations of holes
%   Where two tissue types have the same probability

idx1 = (in1==in2 | in1 == in3 | in1==in4 | in1==in5) & logical(in1)
  ;
idx2 = (in1==in2 | in2 == in3 | in2==in4 | in2==in5) & logical(in2)
  ;
idx3 = (in1==in3 | in2 == in3 | in3==in4 | in3==in5) & logical(in3)
  ;
idx4 = (in1==in4 | in2 == in4 | in3==in4 | in4==in5) & logical(in4)
  ;
idx5 = (in1==in5 | in2 == in5 | in3==in5 | in4==in5) & logical(in5)
  ;
%  This array will have a zero anywhere there were two or more
  common
%  elements between any of the five arrays.
idx = idx1|idx2|idx3|idx4|idx5;

[xmax ymax zmax] = size(in1)
```

```
[x y z] = ind2sub(size(in1),find(idx));  %  get x, y and z
  coordinates of the holes

for i = 1:length(x)  %  go to each hole and do work
    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
        [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)]);
        if commonesttissue == 1 && length(secondbest{1})>=2  % if
          air and something else are equally common, it'll choose
          air.  This forces it to pick the tissue if possible.
            commonesttissue = secondbest{1}(2);
        end
        headin(x(i),y(i),z(i),:) = [commonesttissue 0 QmSTORE(
          commonesttissue) cSTORE(commonesttissue) rhoSTORE(
          commonesttissue) kSTORE(commonesttissue) wSTORE(
          commonesttissue)];
    end
end

out_head = headin;

end
```

A.1.4 build_skin()

```
function [ head_out ] = build_skin( head_in )
```

```matlab
% build_skin() Creates a layer of skin around the head
%
% This will check all voxels that were previously labeled
% as soft tissue and checks if it has a neighbor which is air.
% If so, then it is reassigned as skin.

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

% Git a list of all voxels labeled as muscle
muscle_voxels = find(head_in==13);

% Go through each of them and check for neighboring air voxels
for i=1:length(muscle_voxels)
   [x y z] = ind2sub(size(head_in), muscle_voxels(i));
   % makes sure we're not at a voxel at the boundry of the dataset
   if (x~=1) && (x~=size(head_in,1)) && (y~=1) && (y~=size(head_in
     ,2)) && (z~=1) && (z~=size(head_in,3))
     % Looks for neighboring voxels that are air
     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))
         head_in(x,y,z) = 14;
     end
   end
end

head_out = repair_headdata(head_in);
```

```

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.

```
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 dimensions


% =====================
% = 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

% Reassign the parameter values
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));

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. [14]. 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.

```
% %==================================================================

%%     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)
    dir_name = directories(i).name;
    main_path = cd( [dir_name filesep dir_name '_NIFTI_1'] );
    mkdir 'Coregistered'
    movefile('r*.nii','Coregistered')
    main_path = cd( [dir_name filesep dir_name '_NIFTI_2'] );
    mkdir 'Coregistered'
    movefile('r*.nii','Coregistered')
    cd(main_path)
end

%%  Calculate Rest State
disp('Calculating Rest State')
for i = 1:length(directories)
    dir_name = directories(i).name;
    avg_NII_rest([dir_name filesep dir_name '_NIFTI_1' filesep '
      Coregistered']);
    avg_NII_rest([dir_name filesep dir_name '_NIFTI_2' filesep '
      Coregistered']);
end
```

```


%% Normalize to Rest and Mask
disp('Normalize to Rest and Mask')
for i = 1:length(directories)
    dir_name = directories(i).name;
    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');
    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');
end


%% Calculate metabolism and blood flow change
disp('Calculate metabolism and blood flow change')
for i = 1:length(directories)
    dir_1 = [ directories(i).name filesep  directories(i).name '_NIFTI_1' filesep 'Coregistered' filesep 'Normalized_to_rest' ];
    dir_2 = [ directories(i).name filesep  directories(i).name '_NIFTI_2' filesep 'Coregistered' filesep 'Normalized_to_rest' ];
    BOLDtoMF(dir_1);
    BOLDtoMF(dir_2);
end
```

```matlab


%%  Calculate the change in temperature based on metabolism and blood flow

% load('equil.mat');  % equillibriumT
% load('tt_headdata.mat');  % headdata
mask = loadNII('fullBrainMask.nii');

for i = 1:length(directories)
    disp([int2str(i) '-1 started'])
    tic
    % Part I
    actResult.dat = tempCalcDynMF(headdata, 37, 24, 720, 360, equillibriumT, ...
        fullfile(directories(i).name,[directories(i).name '_NIFTI_1'],'Coregistered', 'Normalized_to_rest', 'Output_18-Sep-2011', 'm.mat'), ...
        fullfile(directories(i).name,[directories(i).name '_NIFTI_1'],'Coregistered', 'Normalized_to_rest', 'Output_18-Sep-2011', 'f.mat'), ...
        4, mask);
    % Store the parameters used for the calculations for reference in the future
    [c lmax] = max(actResult.dat(:));
    [likelymax x y z] = ind2sub(size(actResult.dat),lmax);
    actResult.likelymaxslice = round(likelymax/2);
    actResult.bloodT = 37;
    actResult.airT = 24;
    actResult.tmax = 360;
    actResult.stepf = 2;
    actResult.savestepf = 4;
    actResult.metabandflowdata = 'From Dataset';
    save(fullfile(directories(i).name,[directories(i).name '
      _NIFTI_1'],'Coregistered', 'Normalized_to_rest', 'Output_18-
      Sep-2011','tt_act_res.mat'), 'actResult');
    old = cd([directories(i).name,filesep,[directories(i).name '
      _NIFTI_1'],filesep,'Coregistered', filesep,'
      Normalized_to_rest', filesep,'Output_18-Sep-2011']);
    writeT_to_nii(actResult, equillibriumT, exp_nii);
    cd(old)
    clear actResult
    % Part II
    disp([int2str(i) '-2 started'])
    actResult.dat = tempCalcDynMF(headdata, 37, 24, 720, 360,
      equillibriumT, ...
        fullfile(directories(i).name,[directories(i).name '_NIFTI_2
          '],'Coregistered', 'Normalized_to_rest', 'Output_18-Sep
          -2011', 'm.mat'), ...
        fullfile(directories(i).name,[directories(i).name '_NIFTI_2
          '],'Coregistered', 'Normalized_to_rest', 'Output_18-Sep
          -2011', 'f.mat'), ...
        4, mask);
    [c lmax] = max(actResult.dat(:));
    [likelymax x y z] = ind2sub(size(actResult.dat),lmax);
    actResult.likelymaxslice = round(likelymax/2);
    actResult.bloodT = 37;
    actResult.airT = 24;
    actResult.tmax = 360;
    actResult.stepf = 2;
    actResult.savestepf = 4;
    actResult.metabandflowdata = 'From Dataset';
    save(fullfile(directories(i).name,[directories(i).name '
      _NIFTI_2'],'Coregistered', 'Normalized_to_rest', 'Output_18-
      Sep-2011','tt_act_res.mat'), 'actResult');
    old = cd([directories(i).name,filesep,[directories(i).name '
      _NIFTI_2'],filesep,'Coregistered', filesep,'
      Normalized_to_rest', filesep,'Output_18-Sep-2011']);
    writeT_to_nii(actResult, equillibriumT, exp_nii);
    cd(old)
    clear actResult
    disp([int2str(i) ' finished in ' num2str(toc)])
end
```

A.2.2 avg_NII_rest()

```
function [ ] = avg_NII_rest( varargin )
%UNTITLED4 Summary of this function goes here
%   Detailed explanation goes here

%% Setup
switch length(varargin)
    case 0
        fold_name = uigetdir;
        if ~fold_name  % Cancel Button
            return
        end
    case 1
        fold_name = varargin{1};
    otherwise
end

%  Go to the folder containing the files
oldfold = cd(fold_name);
file_list = dir('*.nii');

%  We're only interested in the rest period
%  (first and last 10 steps in this case)
file_list = file_list([1:10 170:180]);
file_count = length(file_list);

%  Cell array to store all of the datasets in.
datHolder = cell(file_count,1);

statusbar = waitbar(0,'Initializing');

for j=1:file_count
    try
        waitbar(j/file_count,statusbar,sprintf('%d%%',round((j/
            file_count)*100)));
    catch err
        return
    end
    fi = load_nii(file_list(j).name);
    datHolder{j} = fi.img;
end
```

```

%%  Calculate the mean
ymax = size(datHolder{1},2);
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()

```
function [  ] = avg_NII_normalize( varargin )
%UNTITLED6 Normalize to rest state
%   Detailed explanation goes here

%% 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

        [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 1
        fold_name = varargin{1};
        [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
    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
end

%  Go to the folder containing the files
oldfold = cd(fold_name);
file_list = dir('*.nii');
file_count = length(file_list);

%  Make a directoy to save the normalized data to
saveDir = 'Normalized_to_rest';
if ~isdir(saveDir)
    mkdir(saveDir);
end

statusbar = waitbar(0,'Initializing');

% for each file: load it, devide by the rest state and save it
for i=1:file_count
    try
        waitbar(i/file_count,statusbar,[fold_name sprintf('%d%%',
          round((i/file_count)*100))]);
    catch err
        return
    end
    [file_path file_name file_ext] = fileparts(file_list(i).name);
    file_hold = load_nii(file_list(i).name);
    file_hold.img = double(file_hold.img)./rest_dat - 1;
    file_hold.img(~mask_dat) = 0;              % set everything
      outside the mask to 0
    file_hold.img(isnan(file_hold.img)) = 0;  % set all NaN's to 0
    file_hold.img(isinf(file_hold.img)) = 0;  % set all inf's to 0
    file_hold.img(file_hold.img == -1) = 0;   % correct these for
      voxels that are giving me problems
    file_hold.hdr.dime.datatype = 16;  % set the datatype to single
    file_hold.hdr.dime.bitpix = 16;
    save_nii(file_hold,fullfile(saveDir,[file_name '_rn' file_ext])
      )
end

close(statusbar)
cd(oldfold)

end
```

A.2.4 BOLDtoMF()

```
function [ ] = BOLDtoMF( varargin)
%BOLDtoMF Calculate metabolism and blood from from BOLD reponse
%
%   Input: Directory containing a series of *.nii files of the BOLD
%   response.
%
%   Output: Two new files will be created in a new subdirectory
  with a
%   variable for each time step.
%
%   Usage:
%       BOLDtoMF
%       BOLDtoMF(directory)
```

```
%
%   If a directory is not provided, one will be requested.
%
%   Method from Sotero, et. al. 2010

% =========
% = Setup =
% =========
% if a folder isn't an argument, it'll prompt for one
switch length(varargin)
    case 0
        fold_name = uigetdir;
        if ~fold_name  % Cancel Button pressed
            return
        end
    case 1
        fold_name = varargin{1};
    otherwise
        error('Input is not understood')
end

%  Go to the folder containing the files
oldfold = cd(fold_name);
file_list = dir('*.nii');
file_count = length(file_list);

%  Set up a directory for the outputs
newFolder = ['Output_',datestr(clock,1)];
mkdir(newFolder)
```

```

%  Make *.mat files to append the data to
m0001 = 0; f0001 = 0;
save(['./' newFolder '/m.mat'],'m0001');
save(['./' newFolder '/f.mat'],'f0001');

s = loadNII(file_list(1).name);
norm = ones(size(s));

% ===========
% = Do Work =
% ===========
% This will calculate the metabolism and blood flow.  The output is
% appended to 'm.mat' and 'f.mat' within a new folder created
% within the directory containing the data.

statusbar = waitbar(0,'Initializing');

maxBOLD = 0.22;

% Required Parameters
%   [alpha beta a      b      c      A    ]
p = [0.4 1.5 0.1870 0.1572 -0.6041 maxBOLD];

% Calc flow and metabolism for when BOLD = 1
s = 0;
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)));
fNOACT = -((p(1)+p(2)*p(5))/(p(4)*p(2)))*lambertw(y);
```

```matlab
mNOACT = p(3)*fNOACT^(p(5)+1)*exp(-p(4)*fNOACT);


%%  Calc flow and metabolism
disp(fold_name)
for j=1:file_count
    try
      waitbar(j/file_count, statusbar, sprintf('%d%%', round((j/
        file_count)*100)));
    catch err
        return
    end
    s = loadNII(file_list(j).name);  %  Load up the file
    s(isnan(s)) = 1;
    s(isinf(s)) = 1;
    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)));
    if (size(y,1)==91)&&(size(y,2)==109)&&(size(y,3)==91)
        f = -((p(1)+p(2)*p(5))/(p(4)*p(2))).*lambw_mex(real(y));
    else
        f = -((p(1)+p(2)*p(5))/(p(4)*p(2))).*lambw(y);
    end
    m = p(3)*f.^(p(5)+1).*exp(-p(4)*f);
    % Clean up NaNs that may have popped up
    m(isnan(m))=1;
    f(isnan(f))=1;
    % Normalize to resting m and f
    m = m./mNOACT;
    f = f./fNOACT;

    % Rename and save the data
    eval(['m' sprintf('%04d',j) ' = m;']);
    eval(['f' sprintf('%04d',j) ' = f;']);
    eval(['save(''./' newFolder '/m.mat'', ''m' sprintf('%04d',j) '
      '',''-append'');']);
    eval(['save(''./' newFolder '/f.mat'', ''f' sprintf('%04d',j) '
      '',''-append'');']);
    clear m0* f0*
end

close(statusbar)
cd(oldfold)
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.

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

```
%#codegen

if ndims(array_in) ~= 3
    error('This only works (for now) with a three dimensional array
      .')
end

xmax = size(array_in,1);
ymax = size(array_in,2);

array_out = zeros(size(array_in));
  for ix=1:xmax
      for iy=1:ymax
          array_out(ix,iy,:) = wapr(array_in(ix,iy,:));
      end
  end
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()

```matlab
function temperature_Out = tempCalcEquillibrium(tissue,bloodT,airT,
    nt,tmax,pastCalc,printprogress)
% tempCalcEquillibrium  Find the equillibrium values
%   tissue: holds all of the strucual information
%   bloodT: Temperature of the blood
%   airT:   Temperature of the surrounding ait
%   nt:     Max number of time steps
%   tmax:   Total amount of time the simulation should run over
%
%   This is based off of tempCalc() but loops until the rate of
    change of
%   a each voxel is sufficiently small then outputs what's
%   calculated.  If if takes too long to do all at once, split it
    up into
%   smaller time chunks and use the last step from the previous
    dataset as
%   pastCalc in order to resume.
%
```

```
%   Note: This does not save the time corse because it can take a
  lot of step to
%   find the equillibrium.  It outputs the last time step.
%
%   Writen by Greggory Rothmeier (greggroth@gmail.com)
%   Georgia State University Dept. Physics and Astronomy
%   May, 2011
tic
%%   Default Values
if nargin<2, bloodT = 37;         end
if nargin<3, airT = 24;           end
if nargin<4, nt = 100;            end
if nargin<5, tmax = 50;           end
if nargin<6, pastCalc = 0;        end
if nargin<7, printprogress = 1; end

% These rescue the data if the calculation is interrupted.
global temperature
global dirty

c = onCleanup(@InterCatch);
dirty = 1;

dx = 2*10^-3;         %  Voxel size (m)

if nt<(2*tmax),
   warning('Time step size is not large enough.  Results will be
     unreliable.  Consider increasing the number of steps or
     reducing tmax.')
```

```
end


% Constants used that aren't already stored in tissue
[xmax ymax zmax t] = size(tissue);
clear t;
dt = tmax/(nt-1);
% rhoBlood = 1057;
% wBlood = 1000;
% cBlood = 3600;

% =========
% = Setup =
% =========
%   Starts all tissue voxels at bloodT (default 37) and maintains
  air at airT (default 24)
%   The condition squeeze(tissue(:,:,:,)~=airIndex picks out the
  elements that are
%   tissue

temperature = ones(3,xmax,ymax,zmax,'single')*airT;
if pastCalc == 0
    temperature(1,squeeze(tissue(:,:,:,1))~=1) = bloodT;
else
    temperature(1,:,:,:) = pastCalc;
end
numElements = numel(temperature(1,:,:,:));

% ===========
% = Do Work =
% ===========
%   This is a vectorized version of the next section.  For the love of god
%   don't make any changes to this without first looking below to make sure
%   you know what you're changing.  This is [nearly] impossible to
%   understand, so take your time and don't break it.
%   data is stored in 'tissue' as such :
%   [tissuetype 0 Qm c rho k w];  <--  second element is blank for all.
%   [    1      2  3 4  5  6 7

%  This makes an array that has smoothed out variations in k by averaging all
%  of the k's around each voxel (including itself).  This is a hap-hazard
%  solution to the problem that if you only take the value of k for the voxel
%  without considering what surrounds it, it doesn't matter whether the head
%  is surrounded by air, water or anything else.  Since water is a better
%  thermal conductor than air, we need a way of accounting for this.  This is
%  one way:
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;
rhoblood = 1057;
cblood = 3600;

%%  Specify Percision Goal
tolerence = 1;      % fraction of voxels have a slope less than '
  zeropoint'
zeropoint = 2.5e-7;  % point at which the slope between two *steps*
   is considered essentially zero


goal = numElements - tolerence*numElements;
goon = numElements; %  Forces the while loop to run the first time
format shortG;
% temperature(1,:,:,:) = Current Temperature
% temperature(2,:,:,:) = Next Temperature
% Resets after each update
if printprogress
    disp(['Goal:  ', num2str(goal),' remaining voxels'])
end
t2 = 1;
while goon(1)>goal && t2<=nt  % runs until either 'goal' elements
  have a slope greater than 'zeropoint' or it exceeds nt
   if printprogress
    disp([t2 goon(1) ((numElements-goon(1))/numElements)*100]) %
      progress
   end
   temperature(2,:,:,:) = squeeze(temperature(1,:,:,:)) + ...
        dt/(tissue(:,:,:,5).*tissue(:,:,:,4)).* ...
        ((averagedk/dx^2).*...
        (circshift(squeeze(temperature(1,:,:,:)),[1 0 0])-2*squeeze(temperature(1,:,:,:))+circshift(squeeze(temperature(1,:,:,:)),[-1 0 0])+...  % shift along x
         circshift(squeeze(temperature(1,:,:,:)),[0 1 0])-2*squeeze(temperature(1,:,:,:))+circshift(squeeze(temperature(1,:,:,:)),[0 -1 0])+...  % shift along y
         circshift(squeeze(temperature(1,:,:,:)),[0 0 1])-2*squeeze(temperature(1,:,:,:))+circshift(squeeze(temperature(1,:,:,:)),[0 0 -1]))...  % shift along z
            -(1/6000)*rhoblood*tissue(:,:,:,7)*cblood.*(squeeze(temperature(1,:,:,:))-bloodT)+tissue(:,:,:,3));
    %   resets the air temperature back since it's also modified above, but
    %   it needs to be kept constant throughout the calculations
    temperature(2,squeeze(tissue(:,:,:,1))==1) = airT;
    %   checks how quickly the temperature is changing and if it is close
    %   enough to zero to be considered stopped ('zeropoint')
    goon = size(temperature(abs(squeeze(temperature(2,:,:,:)-temperature(1,:,:,:)))>zeropoint));
    temperature(1,:,:,:) = temperature(2,:,:,:);  % moves 2 back to 1
    t2 = t2 + 1;
end

temperature_Out = temperature(2,:,:,:);  % Only outputs the last time step
dirty = 0;
```

```
% equilTemperature = temperature_Out;
% save('equil.mat','equilTemperature');

%% To Combine Datasets
%  use this technique if there are seperate datasets that need
   combining
%    vertcat(squeeze(res1(:,:,:,:)),squeeze(res2(2:end,:,:,:)))
%  Where for all by the first dataset, you need to do the time from
    2:end
%  so that there are no repeats (remember that the last timestep
   from the
%  previous dataset serves as the first for the new one)


time = toc;
end

function InterCatch
global dirty
if dirty
    disp('Interupt Intercepted.  Inprepretating Interworkspace Data
      .')
    global temperature
    % equillibriumT = zeros([1 size(temperature(1,:,:,:))]);
    % equillibriumT(1,:,:,:) = temperature(1,:,:,:);   %might be
      better to swtich equilT to be 3d rather than 4d
    equillibriumT = temperature;
    save('equiltempAbortDump.mat','equillibriumT');
    % setappdata(0,'InterpOut',temperature);
end
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

```
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
```

```matlab
if nargin<2,  bloodT = 37;          end
if nargin<3,  airT = 24;            end
if nargin<4,  nt = 3;               end
if nargin<5,  tmax = 1;             end
if nargin<6,  pastCalc = 0;         end


% Length of one side of a voxel (m)
dx = 2*10^-3;

if nt<(2*tmax),
   warning('Time step size is not large enough.  Results will be
     unreliable.  Consider increasing the number of steps or
     reducing tmax.')
end


% Constants used that aren't already stored in tissue
[xmax ymax zmax t] = size(tissue);
clear t;
dt = ones([xmax ymax zmax])*(tmax/(nt-1));
% rhoBlood = 1057;
% wBlood = 1000;
% cBlood = 3600;

%%  Determine Metab/Flow Data Storage System
if ischar(metab)&&ischar(flow)
  % if file locations are given rather than data
    option = 1;
```

```
else
  % Preallocate matrices for holding metabolism and blood flow data
    metabMulti = ones([xmax ymax zmax],'single');
    flowMulti = ones([xmax ymax zmax],'single');
    option = 0;
end

%%  Maps
% Creates a map that identifies where there is tissue
% the condition squeeze(tissue(:,:,:,)~=airIndex picks out the
% elements that are tissue

tmax = ceil((nt-1)/savesteps);
temperatureOut = ones(tmax,xmax,ymax,zmax,'single');
temperature = ones(2,xmax,ymax,zmax,'single')*airT;
if pastCalc == 0
    temperature(1,squeeze(tissue(:,:,:,1))~=1) = bloodT;
else
  % Starts everything off at the pre-determined equilibium
    temperatures
    temperature(1,:,:,:) = pastCalc(end,:,:,:);
end
temperatureOut(1,:,:,:) = temperature(1,:,:,:);


% ===========
% = Do Work =
% ===========
%   This is a vectorized version of the next section.  For the love of
%   god don't make any changes to this without first looking below to
%   make sure you know what you're changing.  This is [nearly]
%   impossible to understand, so take your time and don't break it.
%   data is stored in 'tissue' as such :
%  [tissuetype 0 Qm c rho k w] <--  second element is blank for all.
%  [    1      2  3 4  5  6 7]

% This makes an array that has smoothed out variations in k by
% averaging all of the k's around each voxel (including itself). This
% is a hap-hazard solution to the problem that if you only take the
% value of k for the voxel without considering what surrounds it, it
% doesn't matter whether the head is surrounded by air, water or
% anything else. Since water is a better thermal conductor than air, we
% need a way of accounting for this. This is one way:

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;
rhoblood = 1057;
cblood = 3600;

```

```
%%  Only saves every 4 steps to reduce the final matrix size
for t2 = 1:nt-1
    waitbar(t2/(nt-1),statusbar,sprintf('%d%%',round(t2/(nt-1)*100))
      );

% if a variable needs to be used multiple times for the same time
   step.
    t3 = floor((t2-1)/4)+1;  % 1 1 1 1 2 2 2 2 3 3 . . .

    % if a file is specified, pulls the data from the file for each
      step
    if option
        eval(strcat('load(fullfile(metab),''-mat'',''m',sprintf('%04
          d',t3),''');'));
        eval(strcat('load(fullfile(flow),''-mat'',''f',sprintf('%04d
          ',t3),''');'));
        eval(strcat('metabMulti = m',sprintf('%04d',t3),';'));
        eval(strcat('flowMulti = f',sprintf('%04d',t3),';'));
        eval(strcat('clear f', sprintf('%04d',t3),' m',sprintf('%04d
          ',t3)))
    else
        metabMulti(region) = metab(t2);   %  region is hardcoded
          here
        flowMulti(region) = flow(t2);
    end

    temperature(2,:,:,:) = squeeze(temperature(1,:,:,:)) + ...
         dt./(tissue(:,:,:,5).*tissue(:,:,:,4)).* ...
         ((averagedk/dx^2).*...
         (circshift(squeeze(temperature(1,:,:,:)),[1 0 0])-2*squeeze(temperature(1,:,:,:))+circshift(squeeze(temperature(1,:,:,:)),[-1 0 0])+...  % shift along x
          circshift(squeeze(temperature(1,:,:,:)),[0 1 0])-2*squeeze(temperature(1,:,:,:))+circshift(squeeze(temperature(1,:,:,:)),[0 -1 0])+...  % shift along y
          circshift(squeeze(temperature(1,:,:,:)),[0 0 1])-2*squeeze(temperature(1,:,:,:))+circshift(squeeze(temperature(1,:,:,:)),[0 0 -1]))...  % shift along z
             -(1/6000)*rhoblood*flowMulti.*tissue(:,:,:,7)*cblood.*(squeeze(temperature(1,:,:,:))-bloodT)+metabMulti.*tissue(:,:,:,3));
     % resets the air temperature back since it's also modified above,
     % but it needs to be kept constant throughout the calculations
     temperature(2,squeeze(tissue(:,:,:,1))==1) = airT;
     temperatureOut(ceil(t2/savesteps),:,:,:) = temperature(2,:,:,:);
     temperature(1,:,:,:) = temperature(2,:,:,:);  % moves 2 back to 1
     clear metabMulti flowMulti
 end
 close(statusbar);

 % ============
 % = Old Code =
 % ============
 % This is what used to be used. It's much slower (~60 times slower),
 % but it's much easier to understand compared to the above code. If
    any
 % changes need to be made above, first look through this code to
   ensure
 % you understand what's happening before making changes. It's
   really
 % easy to mess up the code above and nearly impossible to figure
   out
 % where.
 %
 %  good luck.

 % for t2 = 1:nt-1
 %     for x2 = 2:xmax-1
 %         for y2 = 2:ymax-1
 %             for z2 = 2:zmax-1
 %                 if tissue(x2,y2,z2,1) ~= 1,
 %                     temperature(t2+1,x2,y2,z2) = temperature(t2,
   x2,y2,z2) + (dt/(tissue(x2,y2,z2,5)*tissue(x2,y2,z2,4)))*((tissue
   (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)
%%  Error checking and data restructuring where necessary
if ndims(head) == 4
    head = head(:,:,:,1);
elseif ndims(head) ~= 3
    error('Input ''head'' must have either 3 or 4 dimensions');
end

if nargin > 1
    if ndims(tempdata) == 3  % should only happen when comparing
        two equilibrium datasets
    temp = tempdata;
    tempdata = zeros([1 size(temp)]);
    tempdata(1,:,:,:) = temp;
    elseif ndims(tempdata) ~= 4
    error('Input ''tempdata'' must have either 3 or 4 dimensions');
    end
    tempdataShort = squeeze(tempdata(end,:,:,:));
end

if nargin > 2
    if ndims(highlightRegion) ~= 3
    error('Input ''highlightRegion'' must have 3 dimensions');
    end
    if size(highlightRegion) ~= size(head)
    error('Input ''highlightRegion'' must be of the same size as ''
      head''');
    end
    tempdataShort = squeeze(tempdata(end,:,:,:));
end

if nargin > 3
    if slice > size(tempdata,1)
    error('Input ''slice'' must be less or equal to the length of
      the first dimension of ''tempdata''');
    end
    tempdataShort = squeeze(tempdata(slice,:,:,:));
end

if nargin > 4
    if ndims(equil) == 3
        eq = equil;
    elseif ndims(equil) == 4
        eq = squeeze(equil(1,:,:,:));
    else
        error('Input ''equil'' must have either 3 or 4 dimensions')
          ;
    end
    clear 'equil';
end

%%  Pick how to format the call of SliceBrowser()
switch nargin
    case 1
    SliceBrowser(head,1,head);
    colormap(gray);
    case 2
    %SliceBrowser(squeeze(tempdata(size(tempdata,1),:,:,:)),
      tempdata,head);
    SliceBrowser(tempdataShort,tempdata,head);
    case 3
    SliceBrowser(tempdataShort,tempdata,head,highlightRegion);
    case 4
    SliceBrowser(tempdataShort,tempdata,head,highlightRegion);
    case 5
    SliceBrowser(tempdataShort-eq,tempdata,head,highlightRegion);
    case 6
    SliceBrowserOverlay(tempdataShort-eq,tempdata,head,
      highlightRegion,threshold);
    case 7
    imgoverlay(head,tempdataShort-eq,point,threshold)
end

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`).

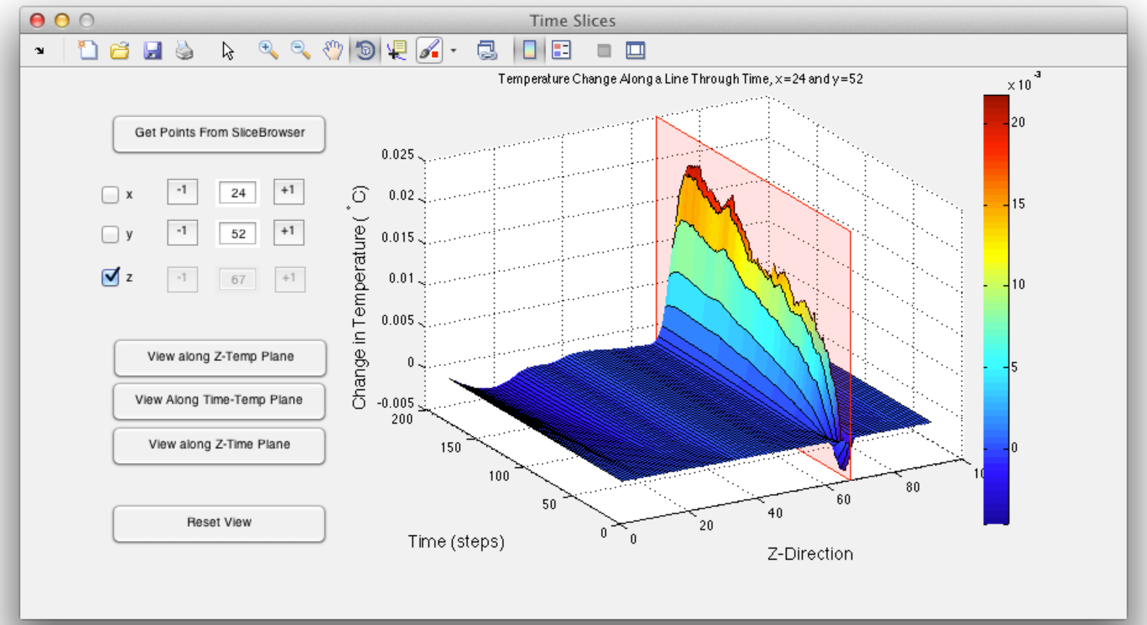

Time Slices
Get Points From SliceBrowser
x -1 24 +1
y -1 52 +1
z -1 67 +1
View along Z-Temp Plane
View Along Time-Temp Plane
View along Z-Time Plane
Reset View
Temperature Change Along a Line Through Time, x=24 and y=52
Change in Temperature (°C)
Time (steps)
Z-Direction
x 10^-3

Figure B.1 Experimental data for activity in the motor cortex visualized with tsliceplot.

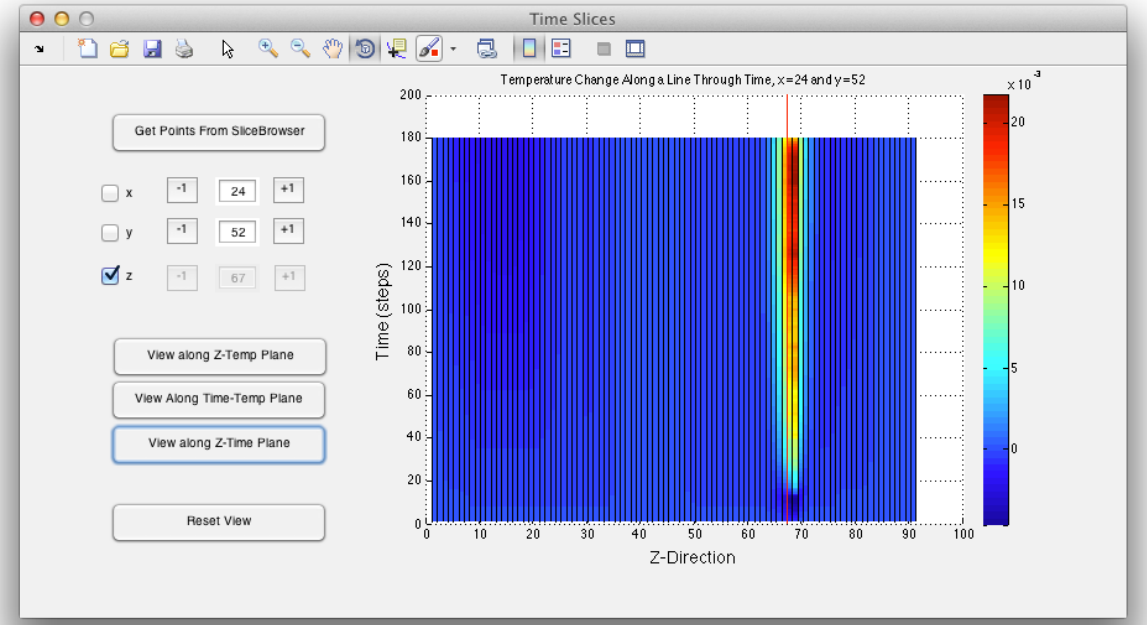


Figure B.2 The same data as is presented in fig. B.1, but viewed flat-on along the z vs. time plane.