Brain tissue temperature dynamics during functional activity and possibilities for Imaging

by

Greggory H. Rothmeier

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

Masters of Science

in the College of Arts and Sciences

Georgia State University

2012

Brain tissue temperature dynamics during functional activity and possibilities for Imaging

by

Greggory H. Rothmeier

Dedication

Mama.

Acknowledgements

Perera, Dhamala, Brooke, Lab Mates, Dhamala's Lab

Contents

List of Tables

List of Figures

Brain tissue temperature dynamics during functional activity and possibilities for Imaging

A thesis

presented in Partial Fulfilment of Requirements for the Degree of Master of Science in the

College of Arts and Sciences

Georgia State University

2012

by

Greggory Rothmeier

Committee:

A. G. Unil Perera, Chair

Mukesh Dhamala, Member

Brian Thoms, Member

D. Michael Crenshaw, Member

April 1, 2012

Date

Dick Miller
Department Chair

Chapter 1

Introduction

Chapter 2

Calculating Temperature Changes using the fMRI BOLD Response

2.1 Background

2.1.1 Generation of the Blood Oxygen Level Dependent (BOLD) Response

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.

2.1.2 Previously Proposed Temperature Models

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.

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

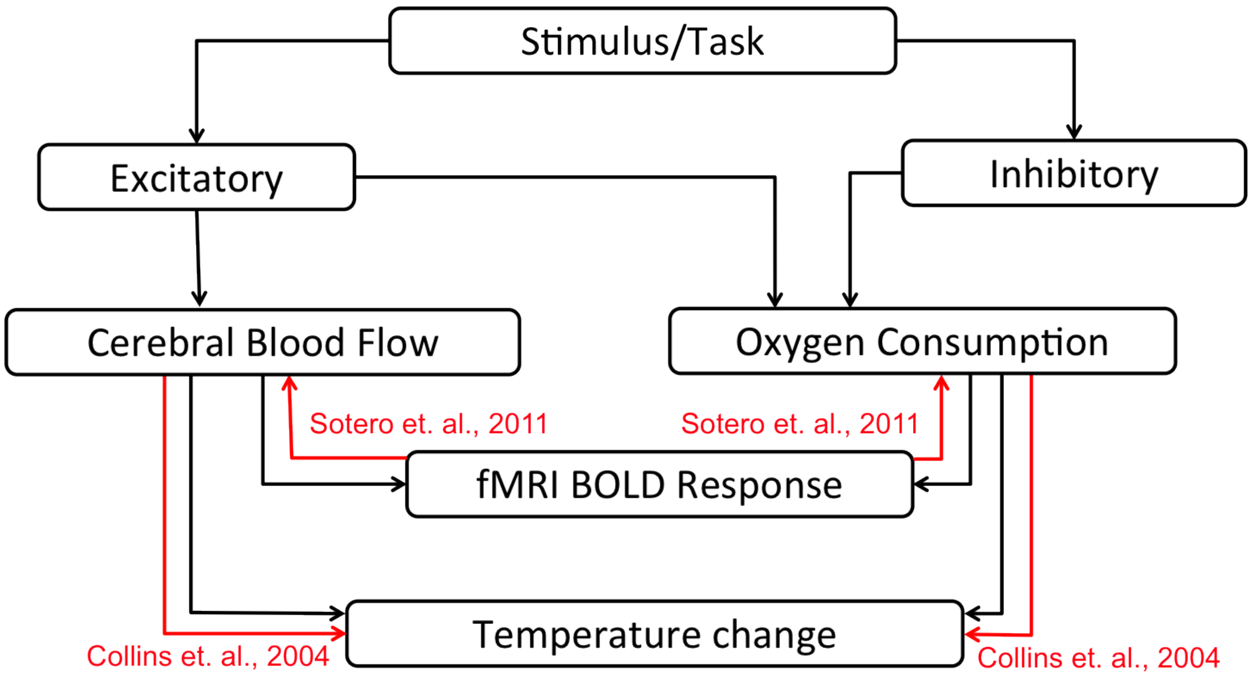


Figure 2.1: Generation of the fMRI BOLD response from changes in neuronal activity. Black arrows indicate a causal relationship while red arrows indicate existing models for the relationship. Modified from Sotero and Trujillo-Barreto [3]

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 [4]. The basis of this model is a modification of the Penne's Bioheat Equation [5, 4].

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

where BLA BLA BLA. One advantage of using eq. (2.1) is that the resting state temperature can be analytically determined by substituting $\frac{dT(t)}{dt} = 0$ [4].

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

If the values provided in table 2.1 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.

$$f(t) = \frac{\alpha + \beta c}{b\beta} W(y(t)) \tag{2.4}$$

$$m(t) = a f^{c+1}(t) e^{-bf(t)} \tag{2.5}$$

$$y(t) = -\frac{b\beta}{\alpha + \beta c} \left[\frac{(A - \frac{S(t)}{S_0} - 1)}{Aa^{\beta}} \right]^{\left(\frac{1}{\alpha + \beta c}\right)} \tag{2.6}$$

Multi-Voxel Approach

Table 2.1: Parameters used to solve the single-voxel Penne's Bioheat Equation. (modified from Sotero and Iturria-Medina [4])

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
$CMRO_2\|_0$	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	

2.2 Modeling the BOLD Response

2.3 Modeling Temperature

2.3.1 The Approach

How the temperature is calculated

$$\rho c\frac{dT}{dt} = k\nabla^2 T - \rho_{blood} f(t) w c_{blood}(T - T_{blood}) + m(t)Q_m \tag{2.7}$$

Calculating the equilibrium temperature

Calculating Metabolism and Blood Flow Changes

Calculating the change in temperature in the active brain

2.3.2 Results

Using Theoretical BOLD Data

Using Experimental BOLD Data

Chapter 3

Detector Applications to measuring the active brain

3.1 Functional Near-Infrared fNIR Imaging

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

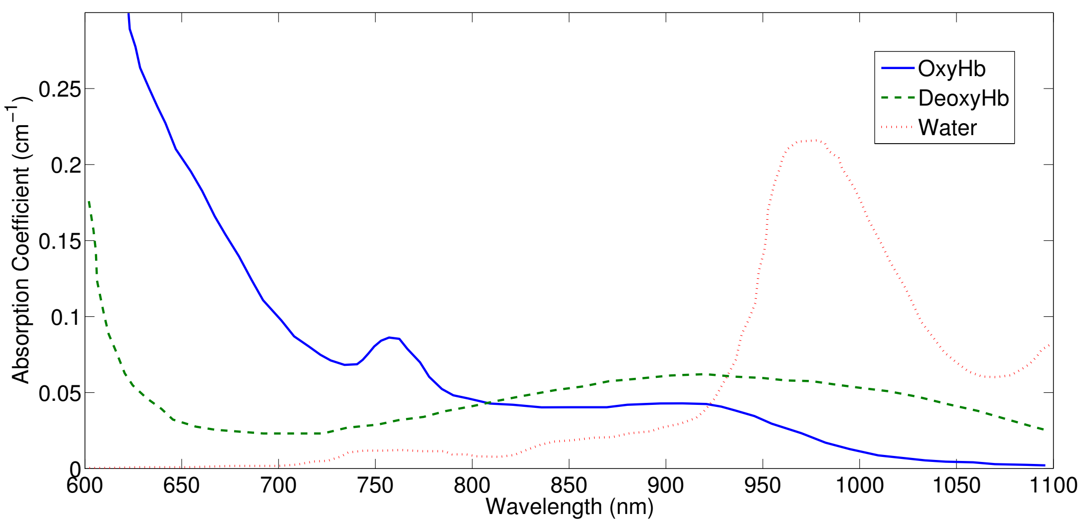


Figure 3.1: Absorption spectra of water, Hb and Dhb. From Cope [6] and HB stuff from Horecker [7]

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.

Chapter 4

Conclusion

Bibliography

[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 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] 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.

[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] 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.

[7] 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.

[8] 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 2011(b) 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/tempcalc`)

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 BulkImportNII() 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) and fillholes() (A.1.3) are functions required by BulkImportNII(). More information about this procedure is in section 2.3.1.

A.1.1 BulkImportNII()

```
function [ total ] = BulkImportNII(varargin)
%  BulkImportNII 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 =
% ====================================
```

```matlab
% The T1 contrast image sould be segmented using SPM8.
%   This loop needs to complete before the next one can begin
for i = 1:5
    eval(strcat('dat',num2str(i),' = loadNII(''rc',num2str(i),'single_subj_T1.nii'');'))  %  Import all of the datat and store as 'cdat1','cdat2', etc.
    eval(strcat('out',num2str(i),' = zeros(cat(2,size(dat',num2str(i),'),7));'))  %  Preallocate
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);'))
    holder(:,:,:,1) = mask;                          %  move structure data to new matrix
    a = find(holder(:,:,:,1) == 1);               %  get indicies of tissues
    [x y z t] = ind2sub(size(holder),a);          %  gets coordinates from index

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

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

    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);     %  Combines data for the head
    model
% The fillholes() function corrects for a voxel having two equally-probable tissue
    types
total = single(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 )
%build_skin Summary of this function goes here
%   Detailed explanation goes here

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

muscle_voxels = find(head_in==13);

for i=1:length(muscle_voxels)
   [x y z] = ind2sub(size(head_in), muscle_voxels(i));
   % if a muscle voxel borders any air voxels, it's set to skin
   if (x~=1)&&(x~=size(head_in,1))&&(y~=1)&&(y~=size(head_in,2))&&(z~=1)&&(z~=size
      (head_in,3))
       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.2 Loading the fMRI Data

The following code automates the procedure of processing and doing all the calculations on the dataset reported in Dhamala et al. [8]. It can be used to gain a better understanding of the procedure. For a conceptual explanation, see section 2.3.1.

```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)
    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


%% 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.2.1 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
```

```matlab
        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.2.2 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);
```

```matlab

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 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.
%
%   From Sotero, et. al. 2010

%%  Setup

%  Check input
switch length(varargin)   % if a folder isn't an argument, it'll prompt for one
    case 0
        fold_name = uigetdir;
        if ~fold_name  % Cancel Button
```

```
            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');



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

%%  Calculate
%
% 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;
%{
%%  Find the max BOLD response
for j=1:file_count
    try
        waitbar(j/file_count, statusbar, sprintf('Finding max change in BOLD: %d%%',round((j/file_count)*100)));
    catch err
        return
    end
    s = loadNII(file_list(j).name);  %  Load up the file
    if max(s(:)) > maxBOLD           %  if the max value beats the current max, take it
        maxBOLD = max(s(:));
        disp([j maxBOLD])
    end
end
%}
%  Required Parameters
p = [0.4 1.5 0.1870 0.1572 -0.6041 maxBOLD]; % [alpha beta a b c A]

%%  Calc flow and metabolism (when BOLD = 1)
%   I thought that the equations should work out so that an input of s = 1
%   returns f and m = 1, but until I sort that out here is a cheating work
%   around.  Make sure this is valid before publishing.

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


%%  Calc flow and metabolism
```

```
disp(fold_name)
for j=1:file_count
    tic
    %avgtime = mean(timelist);
    %disp(avgtime)
    %timeremaining = (file_count-j)*avgtime;
    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; %what to do with NaNs and INFS?  Not sure. maybe set to zero
       for now.
    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));  % <-- compiled
           version: runs faster
    else
        f = -((p(1)+p(2)*p(5))/(p(4)*p(2))).*lambw(y);  % <-- not compiled, but
           still pretty fast
    end
    m = p(3)*f.^(p(5)+1).*exp(-p(4)*f);
    %  Clean up NaNs
    m(isnan(m))=1;
    f(isnan(f))=1;
    %  make sure that if the BOLD was 1 then the meabolism/flow is 1
    %  DOUBLE CHECK THAT THIS IS OK!!!!!!!
    m = m./mNOACT;
    f = f./fNOACT;

    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*   % prevent holding onto variables after they're done being used
       .

    t = toc;
    rem = ((file_count-j)*t)/60;
    disp([file_list(j).name '     ' num2str(rem,4) ' minutes remaining'])
end

close(statusbar)
cd(oldfold)


%%  OLD METHOD

%{
tic
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)));
f = -((p(1)+p(2)*p(5))/(p(4)*p(2))).*lambertw(y);
m = p(3)*f.^(p(5)+1).*exp(-p(4)*f);
toc
%}

%f_out = f;
%m_out = m;
% m = s;
% f = s;
%%  Output
% In order to make it easier when calculating the change in temperature,
% this function will create one *.mat file with a seperate variable for
% each time step.  It's a little anoying but since it's such a large file
% when combined, it's the only way to do it.

%{
newFolder = ['Output ',datestr(clock)];
mkdir(newFolder)
oldFolder = cd(newFolder);
varsM = cell(size(s,1), 1);
varsF = cell(size(s,1), 1);
for k = 1:size(s,1);
    eval(strcat('m',num2str(k),' = squeeze(m(k,:,:,:));'));
    eval(strcat('f',num2str(k),' = squeeze(f(k,:,:,:));'));
    varsM{k} = strcat(',''m',num2str(k),''' ');
    varsF{k} = strcat(',''f',num2str(k),''' ');
end
mfin = strcat(cell2mat(varsM'));
ffin = strcat(cell2mat(varsF'));
eval(strcat('save(''m_BOLD.mat''',mfin,');'));
eval(strcat('save(''f_BOLD.mat''',ffin,');'));
cd(oldFolder);
%}

end
```

A.2.4 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-function/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.

```

```matlab
%#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. (2.7)) until the change in temperature for each voxel falls below a certain threshold. Details about this procedure are available in section 2.3.1.

```
function temperature_Out = tempCalcEquillibrium(tissue,bloodT,airT,nt,tmax,pastCalc,printp
% 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 inc
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])+circsh
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 essentia
```

```
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 tha
    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,:,:,:))+
% shift along x
           circshift(squeeze(temperature(1,:,:,:)),[0 1 0])-2*squeeze(temperature(1,:,:,:))+
% shift along y
           circshift(squeeze(temperature(1,:,:,:)),[0 0 1])-2*squeeze(temperature(1,:,:,:))+
% shift along z
                -(1/6000)*rhoblood*tissue(:,:,:,7)*cblood.*(squeeze(temperature(1,:,:,:))-bloo
     %   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,:,:,:)))>zeropo
    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 t
    equillibriumT = temperature;
    save('equiltempAbortDump.mat','equillibriumT');
    % setappdata(0,'InterpOut',temperature);
end
```

```
end
```

A.4 Calculating the Temperature Change

More details about this algorithm can be found in section 2.3.1.

```
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
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('%04d',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

B.1 Visualizing the data

Requires SliceBrowser (`http://www.mathworks.com/matlabcentral/fileexchange/20604`).

B.1.1 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/tempcalc/tree/master/tt/supportfcts/tsliceplot`).

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

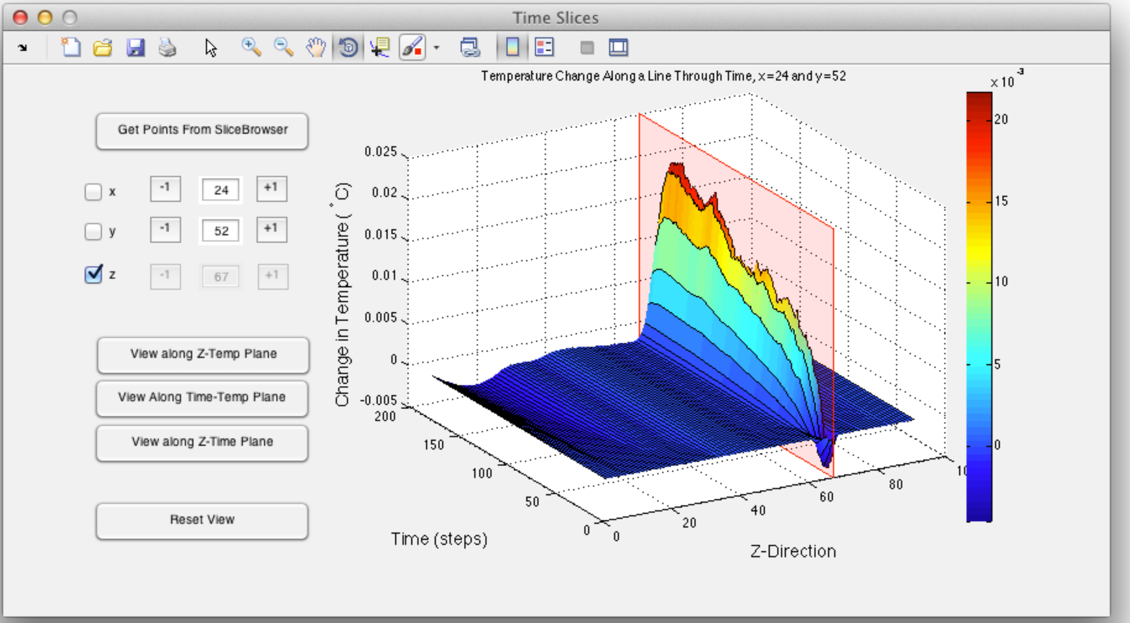
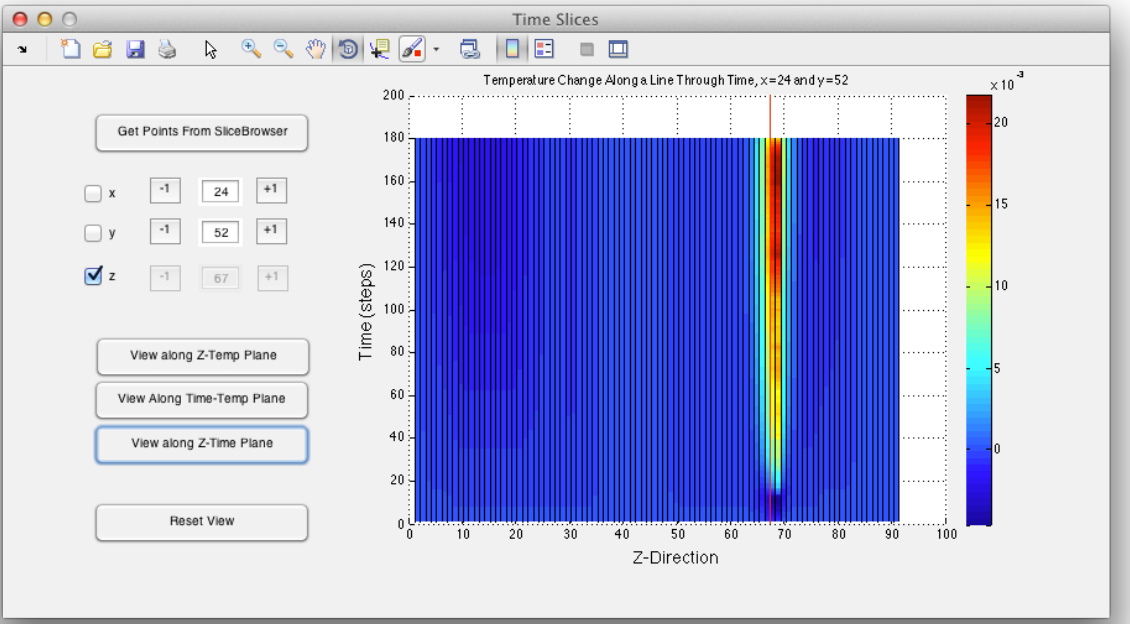

Time Slices
Temperature Change Along a Line Through Time, x=24 and y=52
Get Points From SliceBrowser
x -1 24 +1
y -1 52 +1
z -1 67 +1
View along Z-Temp Plane
View Along Time-Temp Plane
View along Z-Time Plane
Reset View
Change in Temperature (°C)
Time (steps)
Z-Direction
x 10^-3

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


Time Slices
Temperature Change Along a Line Through Time, x=24 and y=52
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
Time (steps)
Z-Direction
x 10^-3