Vince Velocci, Research Student – AIM 6943 Final Report (Spring 2015) – April 27, 2015

Internship start date: Monday September 8, 2014

Internship end date: Thursday April 30, 2015

Supervisor: Dr. Walter Richardson, walter.richardson@utsa.edu, 210-458-4760

Department of Mathematics, The University of Texas at San Antonio

One UTSA Circle

San Antonio, TX 78249 (www.math.utsa.edu)

Duties

I am a research student working with Dr. Richardson on a project whose ultimate goal is the numerical simulation of electromagnetic fields and induced currents in the brain as a result of a medical procedure known as Transcranial Magnetic Stimulation (TMS). My duties included trying out a couple of toy numerical problems, and helping the project get off the ground, to be continued at a future time by another student.

Description of Work and Research Methodology

Transcranial Magnetic Stimulation is a treatment tool that has been used to treat certain neurological or psychiatric disorders, such as Parkinson's disease and depression. TMS is also used as a tool to study the brain, its function, and it structure. In this procedure, a coil, sometimes in the shape of a figure-eight, is placed a short distance above the head. A highly oscillating current is run through the coil. A current through a wire causes a magnetic field, $\mathbf{B}$, to surround the coil. The highly oscillating current, however, causes the formation of a rapidly varying magnetic pulse that travels through the brain and induces (via Faraday's Law of Induction) weak circular currents (eddy currents) in certain targeted brain regions, thereby activating those brain regions. Our goal is to adapt existing simulation software to simulate these currents and the electromagnetic fields in the brain as a result of this procedure. This problem involves solving Maxwell's Equations (which, in this case, take the form of a wave equation due to the highly varying nature of the fields involved) in three dimensions on the human brain.

Open source software known as EIDORS (Electrical Impedance Tomography and Diffuse Optical Tomography Reconstruction Software), initially developed for Matlab, is a set of Matlab programs for image reconstruction in electrical impedance tomography. This involves obtaining a map of the conductivity and permittivity structure of the brain from the surface measurements of electrodes attached to the brain. Mathematically, it involves solving the inverse problem of determining functions $\sigma(\mathbf{x})$ and $\varepsilon(\mathbf{x})$ from surface measurements of the

electric field **E** and magnetic field **H**, where these four quantities are related by Maxwell's Equations which, in this system, take the form

$$\text{curl}(\mathbf{E}) = j\omega\mu\mathbf{H} \text{ and } \text{curl}(\mathbf{H}) = (\sigma - j\omega\varepsilon)\mathbf{E}$$

Assuming a small magnetic permeability μ, and field frequency ω, one can express $\mathbf{E} = \text{grad}(\varphi)$, where φ is the complex-valued electric potential, and the two Maxwell's Equations become the elliptic partial differential equation, $\text{Div}[(\sigma - j\omega\varepsilon)*\text{grad}(\varphi)] = 0$. The measured boundary data are the potential φ, and the component of $\text{grad}(\varphi)$ normal to the head. In the lab, one applies known currents to the head (boundary) and makes measurements of potentials. By solving this inverse problem, EIDORS obtains a map of the brain from the numerically computed values σ and ε. Of course, what I have explained is a simplification of the full version of the process, which involves solving both the forward problem and the inverse problem. The forward problem solves the elliptic PDE given an initial guess for σ and ε and one set of boundary data. The forward problem is solved by modeling the brain (domain) as a 3D finite element mesh of tetrahedra. (Software exists within EIDORS to set up the finite element mesh.) One linearizes the equations resulting from this elliptic PDE about the initial guesses for the constants σ and ε, and EIDORS solves this linear system to update these constants. This is done iteratively until the numerical model agrees with the measured data to the desired precision.

The goal of our research is to adapt the EIDORS software to TMS in order to determine the induced currents in the brain as a result of this medical procedure. An important component of this work is to obtain high quality MRI images of the human brain that can be converted into a finite element mesh to serve as the domain for the solution of Maxwell's Equations in the brain. Software exists to accomplish this, and the software treats the brain as being composed of five different tissue types, each with its own set of properties: skull, skin, white matter, grey matter, and cerebrospinal fluid.

Solving Maxwell's equations on the human brain via the finite element method is one avenue of approach to the problem of modelling the induced currents in TMS. Another avenue of approach involves the interesting theory of fractional order derivatives. It has been recently established that the dielectric behavior of systems (eg. the brain) may be well mathematically modeled by a particularly successful model of viscoelasticity involving differential equations that employ fractional order derivatives. Numerically investigating these fractional order derivatives as well as toy problems associated with such derivatives (eg. the fractional order wave-diffusion equation) was done as a start to this avenue.

Accomplishments Thus Far

Initially, I spent a bit of time installing the EIDORS software as well as it myriad of components onto my computer, dealing with intricacies concerning MyApps (as that is the way I am accessing Matlab), as well as playing around with the software, by trying various included examples and tutorials, and investigating the EIDORS code. I also worked on trying to install the program FreeSurfer (freesurfer.net), which is a free, open source program that runs on Linux

and allows you to analyze and visualize MRI images of the brain. More importantly, Freesurfer allows you to segment the different tissue types from each other before the model is converted into a 3D finite element mesh that may be used to simulate processes on the brain. I successfully installed the program and tried it out on various different toy systems.

One particular program that has been used to simulate different forms of brain stimulation is a package called SimNIBS (Simulation of Non-Invasive Brain Stimulation). The program uses Freesurfer and a routine called mri2mesh to form the finite element mesh of the brain. SimNIBS allows one to calculate the electric field induced by transcranial magnetic stimulation. SimNIBS performs the electric field calculations using Matlab, so having Matlab installed on your machine is a requirement. One cannot, for example, use myapps to access Matlab for this purpose.

I contacted the person behind SimNIBS (Axel Thielscher) and, apparently, one can use the meshes generated by SimNIBS in EIDORS if you want to perform simulations in EIDORS. This is because EIDORS supports meshes created by gmsh, and gmsh is also used by SimNIBS for this purpose. Using the EIDORS function gmsh_read_mesh, one should be able to read in the head meshes created by SimNIBS.

To familiarize myself with this field, I have read numerous papers:

Windhoff, Mirko; Opitz, Alexander; Thielscher, Axel. "Electric Field Calculations in Brain Stimulation Based on Finite Elements: An Optimized Processing Pipeline for the Generation and Usage of Accurate Individual Head Models," Human Brain Mapping, 34: 923-935 (2013).

Elloian, Jeffrey M.; Noetscher, Gregory M.; Makarov, Sergey N.; Pascual-Leone, Alvaro. "Continuous Wave Simulations on the Propagation of Electromagnetic Fields Through the Human Head," IEEE Transactions on Biomedical Engineering, Vol. 61, No. 6, June 2014.

Lionheart, W.R.B.; Arridge, S.R.; Schweiger, M.; Vauhkonen, M.; Kaipio, J.P. "Electrical Impedance and Diffuse Optical Tomography Reconstruction Software," 1st World Congress on Industrial Process Tomography, Buxton, Greater Manchester, April 14-17, 1999.

Mugler, D.H.; Scott, R.A. "Fast Fourier Transform Method For Partial Differential Equations, Case Study: The 2-D Diffusion Equation," Comput. Math. Applic. Vol. 16, No. 3, 221-228, 1988.

Ziegler, et al. "A finite-element reciprocity solution for EEG forward modeling with realistic individual head models," NeuroImage, 103 (2014) 542-551.

Wharmby, Andrew W.; Bagley, Ronald L. "Modifying Maxwell's equations for dielectric materials based on techniques from viscoelasticity and concepts from fractional calculus," International Journal of Engineering Science, 79 (2014) 59-80.

Wagner, Tim A.; Zahn, Markus; Grodzinsky, Alan J.; Pascual-Leone, Alvara. "Three-Dimensional Head Model Simulation of Transcranial Magnetic Stimulation," IEEE Transactions on Biomedical Engineering, Vol. 51, No. 9, September 2004.

I also got in touch with medical researchers at The University of Toronto and McGill University to inquire about a high resolution image set made for the brain, known as the BigBrain dataset: see https://bigbrain.loris.ca/main.php and http://en.wikipedia.org/wiki/BigBrain and http://www.ncbi.nlm.nih.gov/pubmed/23788795 This dataset may prove useful for setting up finite element meshes and determining whether the software can deal with such a large dataset.

The other tasks I accomplished involved using Matlab to code up examples involving fractional order derivatives.

Code 1: A code that computes finite difference approximations to fractional order derivatives, by using the spectral decomposition of the matrix corresponding to the FD approximation to the 2^{nd} derivative. These orders are then illustrated in a special plot, showing full matrices for fractional orders gradually becoming the tri-diagonal for order 2.

```
 N=20;
M=21;
x=linspace(0,1,N+2);
noboundary=zeros(N,1);
deltax=1/(N+1);
r=1/deltax;
y=zeros(N+2,1);
for i=1:N+2
    y(i)=x(i)*(1-x(i))*cos(4*pi*x(i));
end;
v=zeros(N,1);
for i=1:N
    v(i)=y(i+1);
    noboundary(i)=x(i+1);
end;
A=zeros(N,N);
A(1,1)=2;
A(1,2)=-1;
A(N,N)=2;
A(N,N-1)=-1;
for i=2:N-1
    A(i,i-1)=-1;
    A(i,i)=2;
    A(i,i+1)=-1;
end;
[V,D]=eig(A);
alpha=linspace(0,2,M);
for i=1:M
    pcolor(V*(D^(alpha(i)))*V'), drawnow;
    pause(0.5);
end;
for i=1:M
    plot(noboundary,[r*V*(D^(alpha(i)))*V']*v), drawnow;
    pause(0.05);
end;
```

Code 2: A code which numerically computes and plots the solution to the equation

${}_0D_t^{3/2}y(t) + y(t) = f(t)$ $(t > 0)$

$y(0) = y'(0) = 0$

using four different choices for the "forcing function", f(t).

```
for forcing = 1:4
if forcing==1
%-------------------------------------------------------------
n=501;
h=0.1;
alpha=3/2;
L=5;
t=linspace(0,50,n);
y=zeros(1,n);
f=ones(1,n);
for i=3:n
    sum=0;
    w=1;
    k=min([i L/h]);
    for j=1:k-1
        w=(1-(alpha+1)/j)*w;
        sum=sum+w*y(i-j);
    end;
    y(i)=((h^(alpha))*f(i)-sum)/(1+h^(alpha));
end;
plot(t,y)

elseif forcing==2

%-------------------------------------------------------------
n=501;
h=0.1;
alpha=3/2;
L=5;
t=linspace(0,50,n);
y=zeros(1,n);
f=zeros(1,n);
for i=1:n
    f(i)=t(i)*exp(-t(i));
end;
for i=3:n
    sum=0;
    w=1;
    k=min([i L/h]);
    for j=1:k-1
        w=(1-(alpha+1)/j)*w;
        sum=sum+w*y(i-j);
    end;
    y(i)=((h^(alpha))*f(i)-sum)/(1+h^(alpha));
end;
plot(t,y)

elseif forcing==3
```

```
%-----------------------------------------------------------------------
n=501;
h=0.1;
alpha=3/2;
L=5;
t=linspace(0,50,n);
y=zeros(1,n);
f=zeros(1,n);
for i=2:n
f(i)=((t(i))^(-1))*exp(-1/(t(i)));
end;
for i=3:n
    sum=0;
    w=1;
    k=min([i L/h]);
    for j=1:k-1
        w=(1-(alpha+1)/j)*w;
        sum=sum+w*y(i-j);
    end;
    y(i)=((h^(alpha))*f(i)-sum)/(1+h^(alpha));
end;
plot(t,y)

else

%-----------------------------------------------------------------

n=501;
h=0.1;
alpha=3/2;
L=5;
t=linspace(0,50,n);
y=zeros(1,n);
f=zeros(1,n);
for i=1:n
f(i)=(exp(-t(i)))*sin(0.2*t(i));
end;
for i=3:n
    sum=0;
    w=1;
    k=min([i L/h]);
    for j=1:k-1
        w=(1-(alpha+1)/j)*w;
        sum=sum+w*y(i-j);
    end;
    y(i)=((h^(alpha))*f(i)-sum)/(1+h^(alpha));
end;
plot(t,y)
end;
end;
```

Code 3: A code that solves the fractional order wave-diffusion equation

$_0D_t^{\alpha}u(x,t) = u_{xx}(x,t)$ for $0 \le x \le 1$, $0 \le t \le 10$

$u(x,0) = \sin(\pi x)$

for different values of the order of the time derivative, α.

```
T=10;
X=1;
h=0.1;
Nt=(T/h)+1;
alpha=2;
t=linspace(0,T,Nt);
deltax=0.1;
Nx=(X/deltax)+1;
x=linspace(0,X,Nx);
s=(h^alpha)/((deltax)^2);
u=zeros(Nx,Nt);
for i=1:Nx
    u(i,1)=sin(pi*x(i));
end;
for i=1:Nt
    u(1,i)=0;
    u(Nx,i)=0;
end;
A=zeros(Nx-2,Nx-2);
A(1,1)=-2;
A(1,2)=1;
A(Nx-2,Nx-3)=1;
A(Nx-2,Nx-2)=-2;
for i=2:Nx-3
    A(i,i-1)=1;
    A(i,i)=-2;
    A(i,i+1)=1;
end;
M=inv(s*A-eye(Nx-2));
c=zeros(Nx-2,Nt);
for i=1:Nx-2
    c(i,1)=sin(pi*x(i+1));
end;
for i=2:Nt
    old=zeros(Nx-2,1);
    w=1;
    for j=1:i-1
        w=(1-(alpha+1)/j)*w;
        old=old+w*c(:,i-j);
    end;
    c(:,i)=M*old;
end;
for j=2:Nt
    for i=2:Nx-1
        u(i,j)=c(i-1,j);
    end;
end;
figure;
for j=1:Nt
    plot(x,u(:,j)),axis([0 1 -2 2]), drawnow;
```

```
    pause(0.5);
end;
```

In code 1, A is the matrix which represents the finite-difference approximation to minus the 2nd derivative: $-u'' \approx [-u(x_{i-1}) + 2u(x_i) - u(x_{i+1})]/(\Delta x)^2$. The A matrix just contains the coefficients -1, 2, -1. The matrix is symmetric and positive-definite. Thus, we can decompose the matrix using the spectral theorem and take fractional powers of it. This is what is done in code 1.

In codes 2 and 3, we are using the approximation that the α^{th} order derivative of f(t) with $t \geq a$, which is denoted by ${}_aD_t^{\alpha}$, is given by:

$${}_aD_t^{\alpha} f(t) \approx \sum_{j=0}^{\left[\frac{t-a}{h}\right]} (-1)^j \binom{\alpha}{j} f(t - jh)$$

This just comes from the definition of this derivative as

$$h^{-\alpha} \sum_{j=0}^{n} (-1)^j \binom{\alpha}{j} f(t - jh)$$

We denote $w_j^{(\alpha)} = (-1)^{j\binom{\alpha}{j}}$, (j = 0, 1, 2, …) and it turns out that these coefficients obey the following recurrence relationships:

$$w_0^{(\alpha)} = 1;\ w_k^{(\alpha)} = \left(1 - \frac{\alpha+1}{k}\right) w_{k-1}^{(\alpha)},\ k = 1,\ 2,\ 3,\ \ldots$$

Note that if α is an integer or a fraction, we define $\binom{\alpha}{j} = \frac{\alpha(\alpha-1)\ldots(\alpha-j+1)}{j!}$

Also note that h is the time-step size and, in the upper limit for the summation in the approximation to ${}_aD_t^{\alpha} f(t)$, $[A]$ denotes the largest integer not greater than A.

Internship Benefits

Learning how to use Matlab to simulate complex systems as well as numerically solving PDEs is an obvious benefit. This kind of experience can prove valuable in industry. Being involved with a research project from start to finish (though, the ultimate goal is nowhere near complete) is very much unlike any homework problem whose solution is already known. One has to construct the path where one does not (yet) exist, and doing so is an important skill to learn, one that industry values. Jumping into a new research project also involves familiarizing oneself with the literature just enough to get started, and to learn what you need to learn as you go along. This is clearly a benefit, for one can never learn everything at the very start of a project. Being able jump into a field in which I am not an expert, and teaching myself what I

need to know, ultimately producing something of value, is an important thing to show off to potential employers. And this is true for both industry and academia.

Acknowledgements

Thanks to Dr. Richardson for his guidance, and to the researchers Marc-Etienne Rousseau, D. Louis Collins, Alan Evans, and Alain Dagher for referring me to the BigBrain dataset.