Analysis and Simulation of Blood Flow in MATLAB

Jill Mercik and Ryan Banci
Advisor: Prof. Yanlai Chen
Mathematics Department
University of Massachusetts Dartmouth

December 19, 2011

Contents

1 Introduction 3

2 The Circulatory System 3

3 Womersley Number 3

4 Navier-Stokes Equations 4

4.1 Equation . . . 4

5 Finite Difference Method 5

5.1 Implicit Finite Difference Method . . . 5

6 Numerical Scheme 5

6.1 Right-Hand Side for Numerical Matrix . . . 11

7 MATLAB 11

7.1 The Backslash Function . . . 11

7.2 Symbolic Matrix Program . . . 11

8 Preliminary Results 12

9 Future Work 13

10 Appendix A - symbolic matrix source code 14

11 Appendix B - code test 16

1 Introduction

Simulating how blood flows through the human circulatory system is essential in the field of medicine. Simulations of this natural flow can give researchers information on how medicine is distributed through the body. This can help researchers in the field with designing specific treatments that will improve upon patient care. In order to accomplish this simulation, the finite difference method is used. From using this approach, we were able to simulate a one-dimensional simulation of a blood vessel through the use of MATLAB.

2 The Circulatory System

Before we get into any mathematical equations, it is important to know how the human circulatory system works in order to simulate the flow using math. Blood is made up of two components, cells and plasma. The blood cells float in the plasma which is the liquid portion of the blood. Components such as nutrients, hormones and proteins lie within the plasma. The plasma disperses these substances as it circulates throughout the body. The cell portion of the blood contains white and red blood cells as well as platelets.

The circulatory system is made up of arteries, arterioles, capillaries, veins, and the heart. Blood will travel from the aorta through a progression of blood vessels until it reaches the capillaries. However, before the blood can reach the capillaries, it travels through the arterioles where the speed and pressure is constantly changing due to different conditions in the body. By the time the blood reaches the capillaries it is no longer pulsing but flowing continuously. As the blood passes through the capillaries, it acquires waste and the supply is oxygen is thus reduced. From there, the blood enters venules and then the veins and travels back to the heart to restart the process.

3 Womersley Number

The Womersley number is a dimensionless number involved in biofluid mechanics and deals with pulsatile flow frequency. The Womersley number is denoted as α and is written as:

$$\alpha = R\left(\frac{\omega}{v}\right)^{\frac{1}{2}} = R\left(\frac{\omega\rho}{\mu}\right)^{\frac{1}{2}}$$

$R =$ appropriate length scale
$\omega =$ angular frequency of oscillations
$v, \rho, \mu =$ kinematic viscosity. density, and dynamic viscosity of the fluid

The Womersley number comes up in the solution of the Navier-Stokes equations. It indicates the ratio of the oscillatory inertia force to the shear force. In the blood vessel network, parameters such a frequency, density, and dynamic viscosity stay the same throughout, except the tube radii changes. The Womersley number is large in large vessels and small in small vessels.

4 Navier-Stokes Equations

The Navier-Stokes equations are nonlinear partial differential equations that describe the motion of fluid substances. These equations come from applying Newton's second law to fluid motion. Newton's second law states that the rate of change of momentum is proportional to the imposed force and goes in the direction of the force. Besides being useful in the study of blood flow, the Navier-Stokes equations can also help things such as the design of cars and aircrafts, and analysis of pollution. The Navier-Stokes equation dictates velocity of fluid at a given point in space. When the velocity is solved, other quantities such as flow rate can be found.

Isaac Newtons second law (conservation of momentum) shows us that Force is equal to mass times acceleration ($F = ma$). From Newtons second law, we know that net force and mass are the two main influences of acceleration. The more force you have the faster the acceleration and the more mass you have the slower the acceleration. Friction works in opposition to acceleration. Applying this to fluids, the friction comes from the fluid resistance and the fluid pressure. The force created by the fluid and the mass of the fluid are also taken into account. By applying the notions of Newtons second law to fluid motion, we find it easier to describe the characteristics of fluids such as blood in terms of mathematics.

The Navier-Stokes equations has immense influence on science and engineering because of its usage in many practical problems, but the theory behind Navier-Stokes equations remain unsolved, mainly in the idea of turbulence which is the time dependant random behavior seen in fluid motion. We need to work around this problem by using a simplified Navier-Stokes equation that is supplemented by the equations derived from the conservation of mass and the equations found for the boundary conditions of our given problem.

4.1 Equation

The general form of the Navier-Stoke equations that is used to model blood flow is expressed as:

$$\rho\frac{\delta u}{\delta t} + \rho\left(\mu * \bigtriangledown\right)\mu - v \bigtriangleup + \bigtriangledown p = f \tag{1}$$

$$\bigtriangledown * u = 0 \tag{2}$$

where,
ρ = density of the fluid (g/cm^3)
v = viscosity of the fluid ($g/(cm^2 * sec)$)
$p = p(x, y, t)$ = pressure at the point (x, y) at time $t(g/(cm * sec^2)$
f = any external forces acting on the fluid

Equation (1) is a form of the momentum equation while equation (2) is a volume continuity equation that was simplified from the mass continuity equation. The two equations can be written in component form as:

$$\rho\frac{\delta u}{\delta t}+\rho\left(\frac{\delta(u^2)}{\delta x}+\frac{\delta(uv)}{\delta y}\right)-v\left(\frac{\delta^2 u)}{\delta x^2}+\frac{\delta^2(u)}{\delta y^2}\right)+\frac{\delta\rho}{\delta x}=f_1$$

$$\rho\frac{\delta v}{\delta t}+\rho\left(\frac{\delta(uv)}{\delta x}+\frac{\delta(v^2)}{\delta y}\right)-v\left(\frac{\delta^2 v)}{\delta x^2}+\frac{\delta^2(v)}{\delta y^2}\right)+\frac{\delta\rho}{\delta y}=f_2$$

$$\frac{\delta u}{\delta x}+\frac{\delta v}{\delta y}=0$$

where $f=(f_1(x,y,t),f_2(x,y,t))T$.

5 Finite Difference Method

In order to accurately model blood flow, we will use the finite difference method. The Navier-Stokes equations will provide us with a system of nonlinear partial differential equations for blood flow and the cross-sectional area of an artery. We then will use the finite difference method to solve the equations numerically. Using these results we will be able to accurately monitor the finite changes in blood flow. This method incorporates the nonlinear effects in larger vessels with the effects in smaller vessels as well as arterial capillaries. Finite difference methods are used in mathematical analysis to accurately approximate the solutions to differential equations. To use this method to solve a problem, the domain must be divided into a uniform grid in a "time-stepping. The method is then manipulated manner.

5.1 Implicit Finite Difference Method

Using explicit finite difference, we would have an equation assigned for the branch at one step in time and this would be solved to figure out future steps. In implicit finite difference, we are going in more depth by solving the equations for a number of parts of the branch in order to figure out the necessary values for that time domain so we can use it to solve the values for future time domains.

6 Numerical Scheme

To simulate one branch of the blood stream, we will need to find the flow rate (q) and cross sectional area (A) for a section of points along the branch and then repeat this from one time level to the next. The section of points will be our sample size (k) and we will keep this number at 100 for our purposes. Though, the higher the sample size the more accurate the calculation results.

There are a few equations that are crucial in making this one-dimensional simulation. The equations for flow and pressure can be expressed through the conservation of mass and momentum as:

$$\frac{\partial q}{\partial t} + \frac{\partial q}{\partial x} = 0$$

$$\frac{\partial q}{\partial t} + \frac{\partial}{\partial x}\left(\frac{4}{3}\frac{q^2}{A}\right) + \frac{A}{\rho}\frac{E_{stat}h_0}{R_0 A_0}\frac{\partial A}{\partial x} = -8\pi v \frac{q}{A} + v\frac{\partial^2 q}{\partial x^2}$$

where,
A = cross-sectional area
q = flow rate
ρ =density
E_{stat} = static Young's Modulus
h = wall thickness
R_0 = original radius of vessel
H_0 - orginial wall thickness
A_0 = original cross-sectional area

From these equations we can derive the equations we need to find to create a system of equations that we will put into matrix form to be solved in MATLAB.

$$(-0.25k)q_{m-1}^{n+1} + A_m^{n+1} + (0.25k)q_{m+1}^{n+1} = F_1 \tag{3}$$

$$\begin{aligned}(-0.25bk)A_{m-1}^{n+1} + (-0.25ak - 0.5\xi d)q_{m-1}^{n+1} + (1+\xi d)q_m^{n+1} \\ +(0.25bk)A_{m+1}^{n+1} + (0.25ak - 0.5\xi d)q_{m+1}^{n+1} = F_2 \qquad (4)\end{aligned}$$

where,

$$k = \frac{\triangle t}{\triangle x}, \xi = \frac{\triangle t}{\triangle x^2}, F_1 = (0.25k)q_{m-1}^n + A_m^n + (-0.25k)q_{m+1}^n$$

$$\begin{aligned} F_2 =& c + (0.25bk)A_{m-1}^n(0.25ak + 0.5\xi d)q_{m-1}^n + (1 - \xi d)q_m^n \\ &+ (-0.25bk)A_{m+1}^n + (-0.25ak + 0.5\xi d)q_{m+1}^n \end{aligned}$$

Equations (3) and (4) are the main equations we need for our numerical analysis. For our system of equations we still need the boundary condition equations that are required for us to simulate blood flow.

The following equations are for the inlet boundary conditions, which are the conditions for the start of the branch.

$$A_1^{n+1} + (0.5k)q_2^{n+1} = A_1^n + (0.5k)(q_1^n - q_2^n - q_1^{n+1}) \qquad (5)$$

$$A_2^{n+1} + (0.25k)q_3^{n+1} = A_2^n + (0.25k)(q_1^n - q_3^n - q_1^{n+1}) \qquad (6)$$

$$\begin{aligned} &(-0.25bk)A_1^{n+1} + (1 + \xi d)q_2^{n+1} + (0.25bk)A_3^{n+1} + (0.25ak - 0.5\xi d)q_3^{n+1} \\ &-c + (0.25bk)A_1^n + (0.25ak + \xi d)q_1^n + (1 - \xi d)q_2^n + (-0.25bk)A_3^n \\ &\quad +(-0.25ak + \xi d)q_3^n + (0.25ak + 0.5\xi d)q_1^{n+1} \qquad (7) \end{aligned}$$

The conditions for the end of the branch are given by the following set of equations:

$$(-0.5k)q_{M-1}^{n+1} + A_M^{n+1} + (0.5k)q_M^{n+1} = (0.5k)q_{M-1}^n + A_M^n + (-0.5k)q_M^n \qquad (8)$$

$$q_M^{n+1} - p_M^{n+1}y_M^0 \triangle t = \sum_{k=1}^{n} p_M^{n-k}y_M^k \triangle t \qquad (9)$$

For the purpose of our MATLAB simulation we will simplify equation (9) to be:

$$Q_m^{n+1} - Q_1^{n+1} = 0$$

This equation shows that the last value of Q minus the first value of Q would be equal to 0.

Taking these equations we can put them into a matrix.

$$\begin{pmatrix} Equations: & 5 & 6 & 7 \\ Equations: & 3 & 4 & \\ Equations: & 8 & 9 & \end{pmatrix} \begin{pmatrix} A_1 = x_1 \\ A_2 = x_2 \\ \vdots \\ A_{k-1} = x_{k-1} \\ A_k = x_k \\ q_2 = x_{k+1} \\ q_3 = x_{k+2} \\ \vdots \\ q_{k-1} = x_{2k-2} \\ q_k = x_{2k-1} \end{pmatrix} = \big(RHS\big) \qquad (10)$$

For every sample k of the branch, we need to solve for an A_k and a Q_k. Therefore, if there are k samples then there will k variables of A and k variables of Q that we need to find. This equals to a total of $2k$ variables. For this blood flow problem, the value of Q_1^{n+1} is given so there will actually be $2k-1$ variables that we need to solve for. To solve a system of equations in a matrix, we need to put all these variables in one left hand side (LHS) matrix and then multiply it by another matrix that is $2k-1$ by $2k-1$ numbers large.

Rows 1 to 3 of this matrix describe the inlet conditions. While rows 4 to $2k-3$ will be the main body equations for the branch. Rows $2k-2$ and $2k-1$ correspond to the outlet conditions derived from equations (8) and (9). The main body will have $2k-6$ equations since $(2k-6)+3+2 = 2k-1$.

Coordinate Forms of Matrix:

For row 1-3 (Inlet Condition Equations 5,6, and 7):

$$\mathbf{1^{st} \quad row} = \begin{cases} (Mat)_{1,1} = 1 \\ (Mat)_{1,k+1} = \frac{k}{2} \\ (Mat)_{1,j} = 0 for j \neq 1, k+1 \end{cases}$$

$$\mathbf{2^{nd} \quad row} = \begin{cases} (Mat)_{2,2} = 1 \\ (Mat)_{2,k+2} = \frac{k}{4} \\ (Mat)_{2,j} = 0 for j \neq 2, k+2 \end{cases}$$

$$\textbf{3}^{\textbf{rd}} \quad \textbf{row} = \begin{cases} (Mat)_{3,1} = -\frac{bk}{4} \\ (Mat)_{3,k+1} = 1 + \varepsilon d \\ (Mat)_{3,3} = \frac{bk}{4} \\ (Mat)_{3,k+2} = \frac{ak}{4} - \frac{\varepsilon d}{2} \\ (Mat)_{3,j} = 0 for j \neq 1, 3, k+1, k+2 \end{cases}$$

For row 4 to $2k-3$ (Main equations 3 and 4):

$$\textbf{4}^{\textbf{th}} \quad \textbf{row} = \begin{cases} (Mat)_{4,1} = 1 \\ (Mat)_{4,k+1} = \frac{k}{4} \\ (Mat)_{4,j} = 0 for j \neq 1, k+1 \end{cases}$$

$$\textbf{5}^{\textbf{th}} \quad \textbf{row} = \begin{cases} (Mat)_{5,2} = 1 \\ (Mat)_{5,k+2} = \frac{k}{4} \\ (Mat)_{5,j} = 0 for j \neq 2, k+2 \end{cases}$$

$$\textbf{6}^{\textbf{th}} \quad \textbf{row} = \begin{cases} (Mat)_{6,3} = 1 \\ (Mat)_{6,k+1} = -\frac{k}{4} \\ (Mat)_{6,k+3} = \frac{k}{4} \\ (Mat)_{6,j} = 0 for j \neq 3, k+1, k+3 \end{cases}$$

$$\vdots$$

$$\textbf{k}^{\textbf{th}} \quad \textbf{row} = \begin{cases} (Mat)_{k,k-3} = 1 \\ (Mat)_{k,k+(k-5)} = -\frac{-k}{4} \\ (Mat)_{k,k+(k-3)} = \frac{k}{4} \\ (Mat)_{k,j} = 0 for j \neq 3, k+1, k+3 \end{cases}$$

$$\mathbf{k+1} \quad \mathbf{row} = \begin{cases} (Mat)_{k+1,2} = \frac{bk}{4} \\ (Mat)_{k+1,k+1} = \frac{ak}{4} - \frac{\varepsilon d}{2} \\ (Mat)_{k+1,j} = 0 for j \neq 2, k+1 \end{cases}$$

$$\mathbf{k+2} \quad \mathbf{row} = \begin{cases} (Mat)_{k+2,1} = \frac{-bk}{4} \\ (Mat)_{k+2,k+1} = 1 + \varepsilon d \\ (Mat)_{k+2,3} = \frac{bk}{4} \\ (Mat)_{k+2,k+2} = \frac{ak}{4} - \frac{\varepsilon d}{2} \\ (Mat)_{k+2,j} = 0 for j \neq 1, 3, k+1, k+2 \end{cases}$$

$$\mathbf{k+3} \quad \mathbf{row} = \begin{cases} (Mat)_{k+3,2} = -\frac{bk}{4} \\ (Mat)_{k+3,k+1} = -\frac{ak}{4} - \frac{\varepsilon d}{2} \\ (Mat)_{k+3,k+2} = 1 + \varepsilon d \\ (Mat)_{k+3,4} = \frac{bk}{4} \\ (Mat)_{k+3,k+3} = \frac{ak}{4} - \frac{\varepsilon d}{2} \\ (Mat)_{k+3,j} = 0 for j \neq 2, 4, k+1, k+2, k+3 \end{cases}$$

$$\vdots$$

For row $2k-2$ and $2k-1$ (Outlet Condition Equations 8 and 9):

$$\mathbf{2k-2} \quad \mathbf{row} = \begin{cases} (Mat)_{2k-2,2k-2} = -\frac{k}{2} \\ (Mat)_{2k-2,k} = 1 \\ (Mat)_{2k-2,2k-1} = \frac{k}{2} \\ (Mat)_{2k-2,j} = 0 for j \neq 2k-2, 2k-1, k \end{cases}$$

$$\mathbf{2k-1} \quad \mathbf{row} = \begin{cases} (Mat)_{2k-1,2k-1} = 1 \\ (Mat)_{2k-1,j} = 0 for j \neq 2k-1 \end{cases}$$

6.1 Right-Hand Side for Numerical Matrix

$$\mathbf{B_{4out}} = 1A_1^n + \frac{k}{2}q_1^n - \frac{k}{2}q_2^n + \frac{k}{2}q_1^{n+1}$$

$$\mathbf{B_{5out}} = 1A_2^n + \frac{k}{4}q_1^n - \frac{k}{4}q_3^n + \frac{k}{4}q_1^{n+1}$$

$$\mathbf{B_{6out}} = c + \frac{bk}{4}A_1^n + (\frac{ak}{4} + \xi d)q_1^n + (1 - \xi d)q_2^n(\frac{-bk}{4})A_3^n + (-0.25ak + \xi d)q_3^n + \frac{ak}{4} + (\frac{\xi d}{2})q_1^{n+1}$$

7 MATLAB

In order to use MATLAB to solve our system of equations, we had to put them into matrix form. In this form, the MATLAB program can understand our problem so we can create a script file that will simulate solutions to the matrices. These solutions will go into helping us simulate a branch of the circulatory system. The MATLAB program will the solve the matrix of equations in equation M_1. It will take a user input for a sample size which will be term k. The sample size k is the number of equidistant points on the branch that our program will solve to find the A_k and Q_k of that point k in the branch. After working out the equations by hand and organizing them in coordinate form as seen in the Numerical Scheme section. We were able to observe the patterns we need to automate the process in a MATLAB script file.

7.1 The Backslash Function

The Backslash function (\) in MATLAB is a matrix left division that allows us to divide matrices with uneven column lengths. This function is useful for solving equations in$Ax = b$ form where we have a matrix for A and a matrix for b and we want to solve for x. We can do this by making $x = A\backslash b$. For our matrix, we will label A as LHS (left-hand side matrix) and b as RHS (right-hand side matrix), so the matrix we will solve would be $LHS * x = RHS$ and $x = LHS\backslash RHS$. If the backslash function fails to give us any results then we can try to use the pinv function: $x = pinv(LHS) * RHS$. The pinv function stands for pseudoinverse,.This function produces a matrix x of the same dimensions as $LHS) * RHS'$.

7.2 Symbolic Matrix Program

To make sure the matrices are correct, a symbolic matrix program was first created in order to test whether or not we were getting the proper patterns in our matrices (See source code in Appendix A). The patterns were discovered by taking the derived equations and working them out by hand by inputting a value m from 1 to k until we were able to decipher a coherent diagonal pattern which we should see in such a numerical scheme. From equation

(3) we were able to find a diagonal pattern of ones as described in working out rows 4 to row k for a smaller sample size then intended for the graph. We only needed to sample $m = 1$ to $m = 5$ to get a good idea of the pattern since this will continue for a infinite number k. From equation (4) we were able to decipher a diagonal pattern of a symbolic numerical value $\frac{bk}{4}$ and $\frac{-bk}{4}$. The values for the RHS matrix given from the inlet and outlet condition equations validate the diagonal patterns from equations (3) and (4).

8 Preliminary Results

Using the symbolic matrix and adding in the all the given values, we would be able to use the script file to find the solutions. Specifically we need to find the values for A_1 to A_k and Q_2 to Q_k .

The given values include:
$E_{stat} = 7 * 10^6\ dyn/cm^2$
kinematic viscosity (v) = 1.1cP
density $(\rho) = 1\ g/cm^3$
$K = \frac{\triangle t}{\triangle x}$
$\triangle x$ = 0.05cm
$\triangle t = 2 * 10^{-3}$
$q_1 = 68 * 10^{-3}\ l/min$
$E = \frac{\triangle t}{\triangle x^2}$
a = $(\frac{8}{3}) * n(\frac{q}{A})$
b = $(\frac{-4}{3}) * (\frac{q}{A}^2)$
c = $-8 * \pi * v * (\frac{q}{A})$
$d = v$
original wall thickness $(h_0) = 6,000 * 10^{-9}m$
original radius $(R_0) = 2209 * 10^{-9}m$
original area $(A_0) = 1.533 * 10^{-11}m$

Using the backslash function gave us the error:
Warning: Matrix is singular to working precision.

This means that our system of equations may not have a solution for $LHS * x = RHS$. To troubleshoot this problem we can work around it by using the pinv function in MATLAB. So we can come with the solutions by solving $x = pinv(LHS) * RHS$.

Results for sample size of 10:

$a_1 = 0.3288$
$a_2 = 0.4503$
$a_3 = 0.3284$

$a_4 = 0.4160$
$a_5 = 0.3358$
$a_6 = 0.3800$
$a_7 = 0.3376$
$a_8 = 0.3600$
$a_9 = 0.3343$
$a_{10} = 0.4863$
$q_2 = 0.02770$
$q_3 = 0.0478$
$q_4 = 0.0677$
$q_5 = 0.0816$
$q_6 = 0.0446$
$q_7 = 0.0402$
$q_8 = 0.0565$
$q_9 = 0.0298$
$q_{10} = 0.0000$

The results for a small sample size show that the flow rate increases over time and then decreases. The results are inconclusive until we are able to run the program for a bigger sample size of at least 100 samples.

9 Future Work

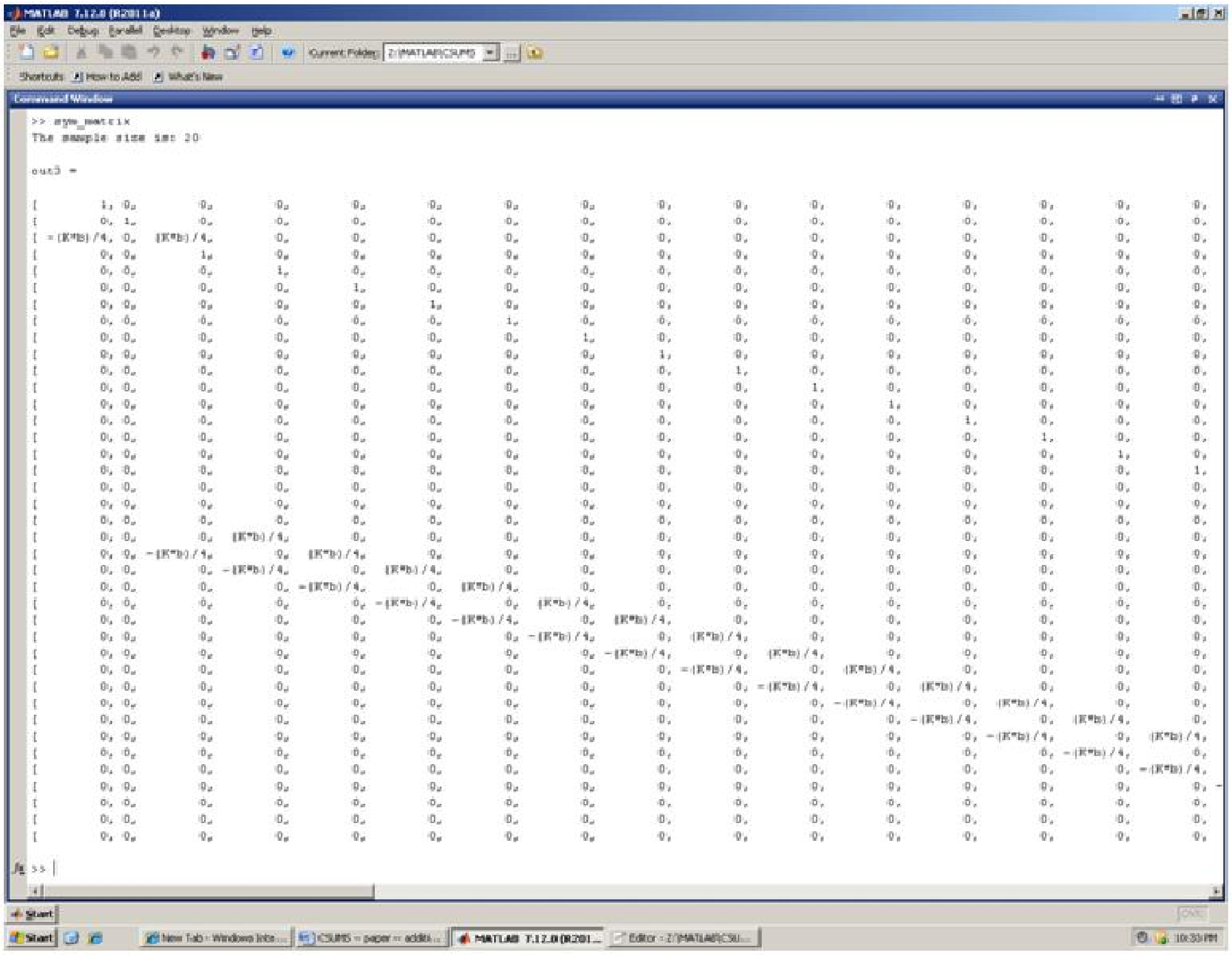
The next logical step for this research project is to apply the MATLAB program to work for a larger sample size of 100 or greater. After that step is complete, it would be important to analyze the plots for flow rate vs. time and for flow rate vs. area. The solutions for the $n+1$ time domain would be used to solve the next sequential domains. This would complete the 1D simulations and then 2D simulations could be derived.

10 Appendix A - symbolic matrix source code

```
%sym_matrix
%Description: Generates symbolic matrix for RHS blood flow equations
%Ryan Banci, Jill Mercik

k=input('The sample size is: ');

syms K;
syms E;
syms a;
syms b;
syms d;

%Inlet conditions (B4, B5, B6)
g1=[1 zeros(1,k-1) (K/2) zeros(1,(k-2))];
g2=[0 1 zeros(1,k-2) 0 (K/4) zeros(1,(k-3))];
g3=[(-b*K/4) 0 (b*K/4) zeros(1,k-3) (1+E*d) (a*K/4)-(E*d/2) zeros(1,(k-3))];

%Outlet conditions (B12, B13)
g4=[zeros(1,k) 1 zeros(1,k-5) (-K/2) 0 (K/2)];
g5=[zeros(1,2*k-2) 1];

%Middle Matrix Begins (B2, B3)
x1 = ones(1,k-3);
x2 = diag(x1,0);

y1 = (K/4)*ones(1,k-3-1);
y2 = diag(y1,1);
y3 = (-K/4)*ones(1,k-3-1);
y4 = diag(y3,-1);
yout = y2 + y4;

j1 = (b*K/4)*ones(1,k-3-1);
j2 = diag(j1,1);
j3 = (-b*K/4)*ones(1,k-3-1);
j4 = diag(j3,-1);
jout= j2 + j4;

z1 = ((a*K/4)-(E*d/2))*ones(1,k-3);
z2 = diag(z1,0);
```

```
 z3 = (1+E*d)*ones(1,k-3-1);
 z4 = diag(z3,-1);
 z5 = ((-a*K/4)-(E*d/2))*ones(1,k-3-2);
 z6 = diag(z5,-2);
 zout = z2 + z4 + z6;

 b1 = zeros(2*k-6,2);  %buffer zeros
 b2 = zeros(2*k-6,3);
%Middle Matrix Ends

 out1 = [x2 yout; jout zout];
 out2 = [b1 out1 b2];
LHS = [g1; g2; g3; out2; g4; g5] %Left Hand Side
```

11 Appendix B - code test



```
>> sym_matrix
The sample size is: 20

out3 =

[        1,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,
[        0,        1,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,
[ -(K*b)/4,        0,  (K*b)/4,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,
[        0,        0,        1,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,
[        0,        0,        0,        1,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,
[        0,        0,        0,        0,        1,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,
[        0,        0,        0,        0,        0,        1,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,
[        0,        0,        0,        0,        0,        0,        1,        0,        0,        0,        0,        0,        0,        0,        0,        0,
[        0,        0,        0,        0,        0,        0,        0,        1,        0,        0,        0,        0,        0,        0,        0,        0,
[        0,        0,        0,        0,        0,        0,        0,        0,        1,        0,        0,        0,        0,        0,        0,        0,
[        0,        0,        0,        0,        0,        0,        0,        0,        0,        1,        0,        0,        0,        0,        0,        0,
[        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        1,        0,        0,        0,        0,        0,
[        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        1,        0,        0,        0,        0,
[        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        1,        0,        0,        0,
[        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        1,        0,        0,
[        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        1,        0,
[        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        1,
[        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,
[        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,
[        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,
[        0,        0,        0,  (K*b)/4,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,
[        0,        0, -(K*b)/4,        0,  (K*b)/4,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,
[        0,        0,        0, -(K*b)/4,        0,  (K*b)/4,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,
[        0,        0,        0,        0, -(K*b)/4,        0,  (K*b)/4,        0,        0,        0,        0,        0,        0,        0,        0,        0,
[        0,        0,        0,        0,        0, -(K*b)/4,        0,  (K*b)/4,        0,        0,        0,        0,        0,        0,        0,        0,
[        0,        0,        0,        0,        0,        0, -(K*b)/4,        0,  (K*b)/4,        0,        0,        0,        0,        0,        0,        0,
[        0,        0,        0,        0,        0,        0,        0, -(K*b)/4,        0,  (K*b)/4,        0,        0,        0,        0,        0,        0,
[        0,        0,        0,        0,        0,        0,        0,        0, -(K*b)/4,        0,  (K*b)/4,        0,        0,        0,        0,        0,
[        0,        0,        0,        0,        0,        0,        0,        0,        0, -(K*b)/4,        0,  (K*b)/4,        0,        0,        0,        0,
[        0,        0,        0,        0,        0,        0,        0,        0,        0,        0, -(K*b)/4,        0,  (K*b)/4,        0,        0,        0,
[        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0, -(K*b)/4,        0,  (K*b)/4,        0,        0,
[        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0, -(K*b)/4,        0,  (K*b)/4,        0,
[        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0, -(K*b)/4,        0,  (K*b)/4,
[        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0, -(K*b)/4,        0,
[        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0, -(K*b)/4,
[        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0, -
[        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,
[        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,
[        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,        0,

>> 
```

References

[1] Y. Huo and G.S. Khassab, ”A hybrid one-dimensional/Womersley model of pulsatile blood flow in the entire coronary arterial tree,” *Am J Physiol Heart Circ Physiol* 292: H2623-H2633, 2007.

[2] J. Majdalani, ”Exact Navier-Stokes solution for pulsatory viscous channel flow with arbitrary pressure gradient,” Journal of Propulsion and Power, vol. 24, no. 6, pp. 1412-1423, 2008.

[3] J.R. Womersley, ”Method for the calculation of velocity, rate of flow and viscuous drag in arteries when the pressure gradient is known,” J. P hysiol, vol. 127, pp. 553-563, 1955.

[4] J. Kimball, ”Human Circulatory System,” 2010, Retrieved October 15, 2011, from http://users.rcn.com/jkimball.ma.ultranet/BiologyPages/C/Circulation.html.