

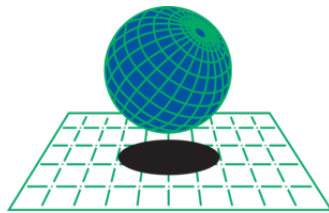
UNIVERSITAT POLYTECHNICA DE CATALUNYA  
MSC COMPUTATIONAL MECHANICS  
Spring 2018

# Finite Element in Fluids

**ASSIGNMENT**

Due 01/06/2018

Alexander Keiser



**CIMNE**<sup>®</sup>



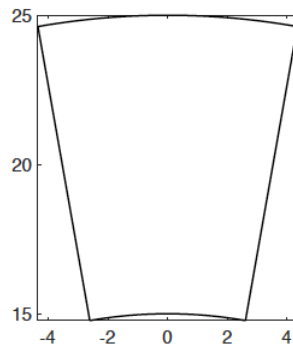
# Contents

<b>1</b>	<b>Transport Problem</b>	<b>2</b>
1.1	Galerkin's Time ( $\theta = 2/3$ ) and Space Descretization . . . . .	2
1.2	Implementation of the Galerkin's Descretization . . . . .	4
1.3	Mesh Generation . . . . .	5
1.4	Solution Results . . . . .	6
<b>2</b>	<b>Stokes Problem</b>	<b>11</b>
2.1	Discretization of The Stokes Equations . . . . .	12
2.2	Creation of the Meshes . . . . .	13
2.3	Imposition of Boundary Conditions . . . . .	14
2.4	Solution Results . . . . .	15
<b>3</b>	<b>Coupled Problem</b>	<b>18</b>
3.1	Discretization of the First Equation . . . . .	19
3.2	Mesh Generation . . . . .	20
3.3	Solution Results . . . . .	20
<b>4</b>	<b>APPENDIX</b>	<b>26</b>

# 1 Transport Problem

We will first solve a transport problem to obtain actin filament and monomer densities (F and G) respectively. The problem can be modeled by the following domain and coupled system of partial differential equations below. The filament density is constant at the upper boundary:  $F(r = 25) = 80\mu\text{M}$ . No flux boundary conditions are considered for F everywhere else and for G on the boundary. The problem is considered with a velocity field  $u(x,y) = 1/1500 (rx,ry) \mu\text{m/s}$ , where (x,y) are the points coordinates and  $r = \sqrt{x^2 + y^2}$ . The material properties can be seen in figure 2 on page 4.

$$\begin{cases} \frac{\partial F}{\partial t} = -u \cdot \nabla F + D_F \nabla^2 F - \sigma_F F & \text{in } (0, T) \times \Omega \\ \frac{\partial G}{\partial t} = D_G \nabla^2 G - \sigma_G G + \hat{\sigma}_{GF} F & \text{in } (0, T) \times \Omega \end{cases}$$



We will first discretize the above two equations using the Galerkin's theta method in time and standard galerkins method in space. This method was chosen for its unconditional stability and relative ease in implementation.

## 1.1 Galerkin's Time ( $\theta = 2/3$ ) and Space Descretization

starting with equation 1 below

$$1) \quad \frac{\partial F}{\partial t} = -U \cdot \nabla F + D_F \nabla^2 F - \sigma_F F \quad \text{in } (0, T) \times \Omega$$

introducing the theta method formula

$$\frac{\Delta U}{\Delta t} - \theta \Delta U_t = U_t^n$$

and plugging in equation 1 gives us

$$\begin{aligned} \frac{\Delta F}{\Delta t} - \theta \left[ -U \cdot \nabla (\Delta F) + D_F \nabla^2 (\Delta F) - \sigma_F (\Delta F) \right] &= \dots \\ &= -U \cdot \nabla F^n + D_F \nabla^2 F^n - \sigma_F F^n \end{aligned}$$

continuing on the next page...

Now expanding and applying Galerkins in space gives us

$$\begin{aligned} \left( w, \frac{\Delta F}{\Delta t} \right) - \theta \left[ \left( w, u \cdot \nabla(\Delta F) \right) + \left( w, D_F \nabla^2(\Delta F) \right) - \left( w, \sigma_F(\Delta F) \right) \right] &= \dots \\ &= \left( w, -u \cdot \nabla F^n \right) + \left( w, D_F \nabla^2 F^n \right) - \left( w, \sigma_F F^n \right) \end{aligned}$$

And integrating by parts the necessary terms

$$\begin{aligned} -\theta D_F \left( w, \nabla^2(\Delta F) \right) &= -\theta D_F \left[ \left( w, \nabla(\Delta F) \right) - \left( \nabla w, \nabla(\Delta F) \right) \right] \\ D_F \left( w, \nabla^2 F^n \right) &= D_F \left[ \left( w, \nabla F^n \right) - \left( \nabla w, \nabla F^n \right) \right] \end{aligned}$$

And finally rearranging

$$\begin{aligned} \left( w, \frac{\Delta F}{\Delta t} \right) + \theta \left( w, u \cdot \nabla(\Delta F) \right) + \theta D_F \left( \nabla w, \nabla(\Delta F) \right) + \theta \sigma_F \left( w, (\Delta F) \right) &= \dots \\ &= - \left( w, u \cdot \nabla F^n \right) - D_F \left( \nabla w, \nabla F^n \right) - \sigma_F \left( w, F^n \right) \end{aligned}$$

2)  $\frac{\partial G}{\partial t} = D_G \nabla^2 G - \sigma_G G + \hat{\sigma}_{GF} F$  following the same method for equation 2

$$\frac{\Delta G}{\Delta t} - \theta \left[ D_G \nabla^2(\Delta G) - \sigma_G(\Delta G) + \hat{\sigma}_{GF}(\Delta F) \right] = D_G \nabla^2 G^n - \sigma_G G^n + \hat{\sigma}_{GF} F^n$$

$$\begin{aligned} \left( w, \frac{\Delta G}{\Delta t} \right) - \theta \left[ \left( w, D_G \nabla^2(\Delta G) \right) - \left( w, \sigma_G(\Delta G) \right) + \left( w, \hat{\sigma}_{GF}(\Delta F) \right) \right] &= \dots \\ &= \left( w, D_G \nabla^2 G^n \right) - \left( w, \sigma_G G^n \right) + \left( w, \hat{\sigma}_{GF} F^n \right) \end{aligned}$$

integrating by parts & rearranging gives us

$$\begin{aligned} \left( w, \frac{\Delta G}{\Delta t} \right) + \theta D_G \left( \nabla w, \nabla(\Delta G) \right) + \theta \sigma_G \left( w, (\Delta G) \right) - \theta \hat{\sigma}_{GF} \left( w, (\Delta F) \right) &= \dots \\ &= - D_G \left( \nabla w, \nabla G^n \right) - \sigma_G \left( w, G^n \right) + \hat{\sigma}_{GF} \left( w, F^n \right) \end{aligned}$$

and now creating  $A_1, B_1, A_2, B_2, C_2$  for code implementation

$$A_1 = M + \theta C \Delta t + \theta D_F K \Delta t + \theta \sigma_F$$

$$B_1 = -C \Delta t - D_F K \Delta t - \sigma_F M \Delta t$$

$$A_2 = M + \theta D_G K \Delta t + \theta \sigma_G M \Delta t$$

$$B_2 = -D_G K \Delta t - \sigma_G M \Delta t$$

$$C_2 = \hat{\sigma}_{GF} M \Delta t$$

\*  $F_1$  &  $F_2$  are arrays of zeros

## 1.2 Implementation of the Galerkin's Descretization

Here we will implement the previously descretized equation. In figure 1 below, a matlab implementation can be seen below corresponding to the A,B,C matrixies descretized on the previous page. Additionally, the implemented material properties can also be found in figure 2 below.

```
88 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
89
90 %IMPLEMENTATION OF GALERKIN TIME AND SPACE DESCRETIZATION
91
92
93 - A1 = M + THETA*C*DT + THETA*D_F*K*DT + THETA*SIGMA_F*M*DT;
94
95 - B1 = -C*DT - D_F*K*DT - SIGMA_F*M*DT;
96
97 - A2 = M + THETA*D_G*K*DT + THETA*SIGMA_G*M*DT;
98
99 - B2 = -D_G*K*DT - SIGMA_G*M*DT;
100
101 - C2 = SIGMA_GF*M*DT;
102
103 - F_1 = ZERO_ARRAY;
104
105 - F_2 = ZERO_ARRAY;
106
107
108 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 1: Implementation of Galerkin's Time and Space Descretization

```
8 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
9 %MATERIAL PARAMETERS
10 - THETA=2/3; SIGMA_GF=0.5; SIGMA_G=2; SIGMA_F=0.25; D_G=15; D_F=5;
11 %s^-1 %s^-1 %s^-1 %um/s %um/s
12 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Figure 2: Implementation of Relevant Material Properties

These two figures are only a portion of the codes, and an entire copy of the main script can be found in the Appendix.

For the solution of this problem we will consider a mesh of bi-linear quadrilateral elements. A sample visualization of this kind of mesh with 3 elements in the theta direction and 3 elements in the radial direction can be seen at toe top of the next page in figure 3.

### 1.3 Mesh Generation

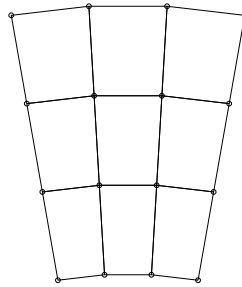


Figure 3: Sample Mesh of 3 by 3 Bilinear Quadrilateral Elements

This mesh is obviously too coarse and a finer mesh is needed for accurate calculation of the solution, however, it is a good visualization for this type of mesh which only has value capturing nodes at every corner. The actual mesh used will consist of 40 elements in the radial direction and 30 in the theta direction. This mesh can be seen below.

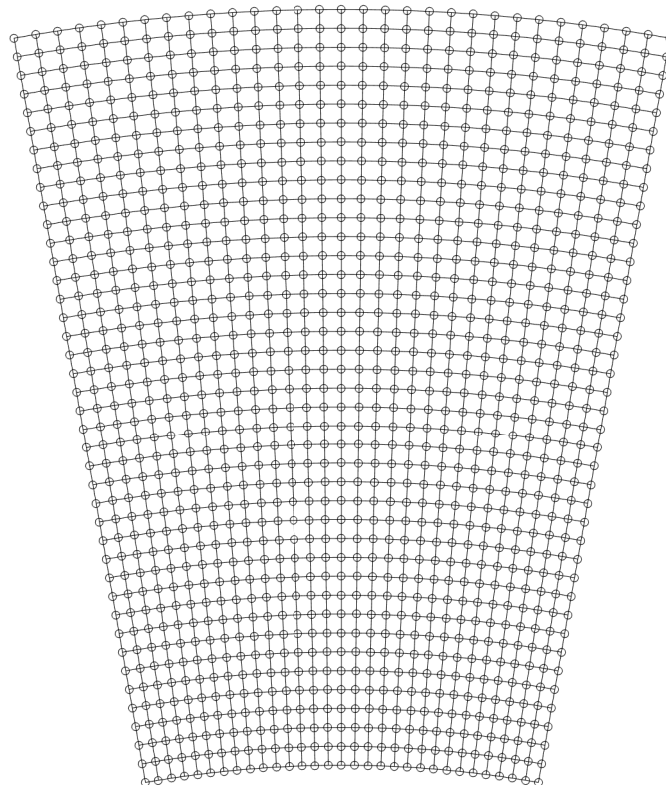


Figure 4: Actual Mesh of 40 Radial and 30 Theta Bilinear Quadrilateral Elements

## 1.4 Solution Results

Here we will now examine the solution results of our implementation. The transient solution results for the density of the actin filaments in the domain at various time steps during the solution of a preliminary code test can be found below in figure 5. The time domain was 10 seconds with 50 total time steps calculated making  $\Delta t=0.2$ .

Figure 5: Preliminary transient solution at various time steps  $\Delta t=0.2$

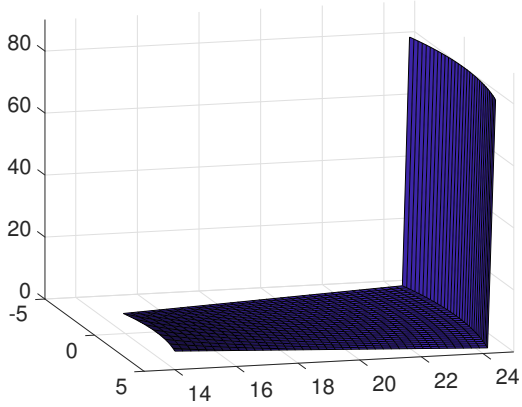


Figure 5a: Time Step 1

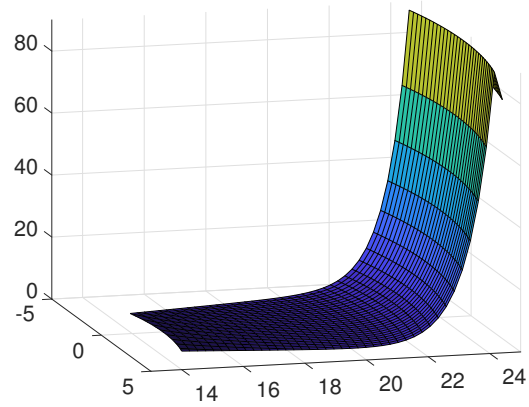


Figure 5b: Time Step 2

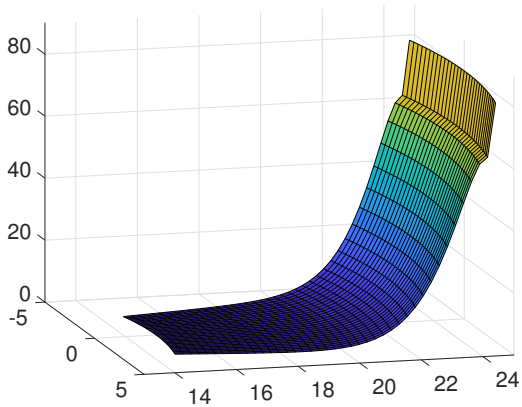


Figure 5c: Time Step 3

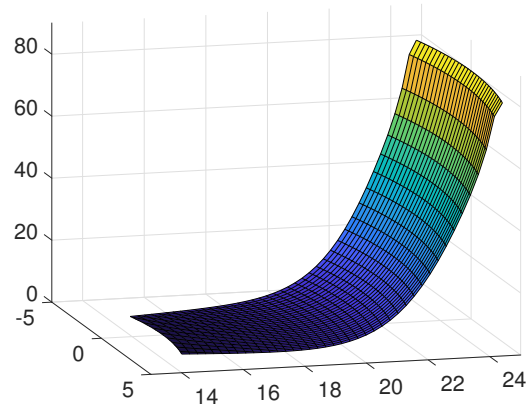


Figure 5d: Time Step 4

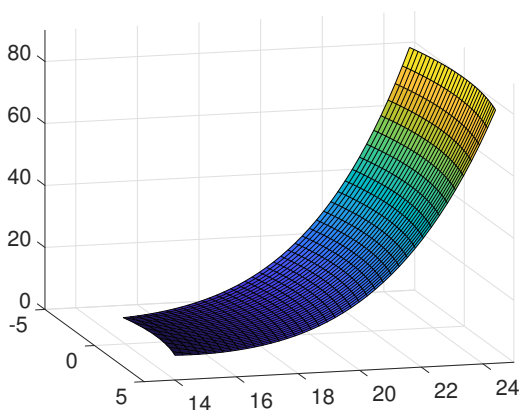


Figure 5e: Time Step 10

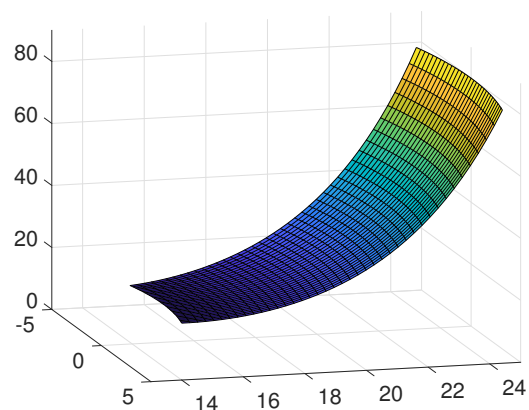


Figure 5f: Time Step 50

On the previous page in figure 5 we can see the results of the previously described problem. We can notice the successful implementation of the boundary conditions in Figure 5a as all of the actin filament density values ( $F$ ) on the further radial boundary ( $r=25$ ) at time step 1 are firmly held at  $80\mu\text{M}$  and stay fixed that way over the entire time domain. We can also notice how the transient solution develops as time progresses. The density regions closest to the dirichlet boundary condition of  $80\mu\text{M}$  increase more rapidly than other regions due to the global convection velocity field. Additionally, we can notice that the longer time goes on the less transient movement we see in the plots. In the first four time steps, the general shape of the final actin density profile is more or less achieved, and for the rest of the time domain, the solution slowly approaches the final solution. We can also take note of the initial oscillatory behavior exhibited during the first 4 time steps. This undesirable behavior was most likely due to the choosing of too large of a time step, causing instabilities in the beginning of transient solution of the problem in the region of the actin filament density domain close to the imposition of the dirichlet boundary condition. We will now see results of the same problem setup but with a more refined time step and will confirm that this was indeed the cause of the instabilities and oscillations exhibited in the solution. This can be seen below in figure 6.

Figure 6: Final transient solution at various time steps  $\Delta t=0.02$

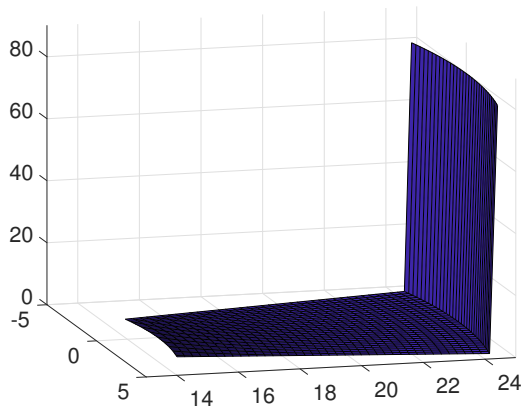


Figure 6a: Time Step 1

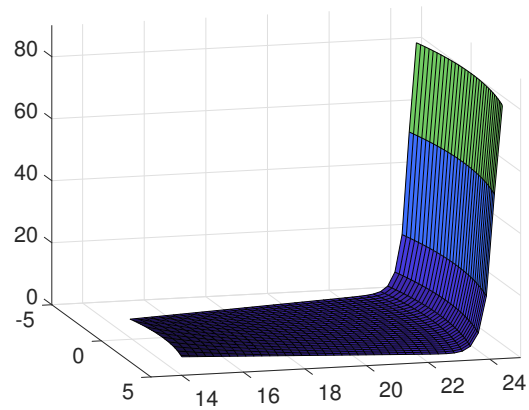


Figure 6b: Time Step 2

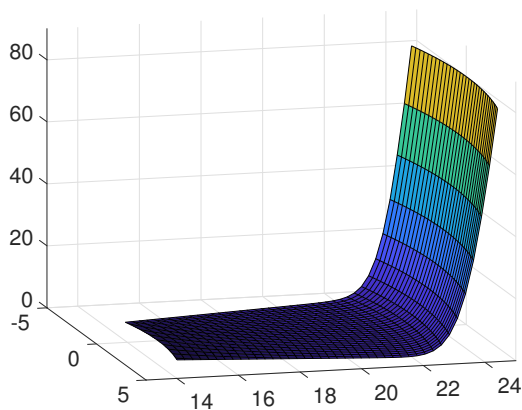


Figure 6c: Time Step 5

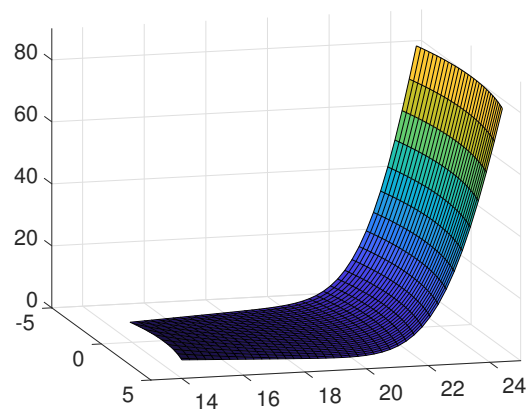


Figure 6d: Time Step 15



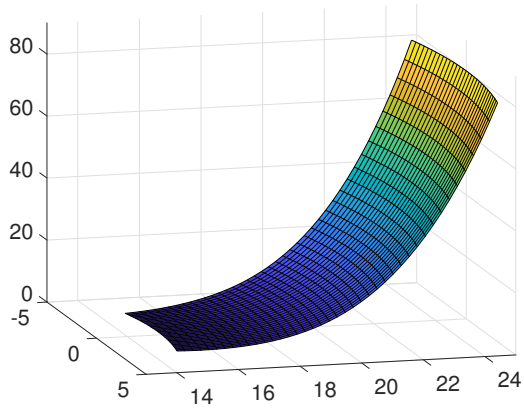


Figure 6e: Time Step 75

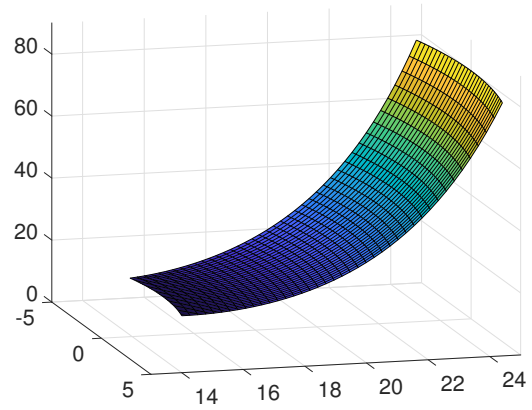


Figure 6f: Time Step 500

The above results were computed with 500 time steps over a time domain of 10 seconds producing the value for time step  $\Delta t=0.02$ . It is immediately apparent that the troublesome oscillations present with the higher value of  $\Delta t=0.02$  have been removed, and the solution behaves better. Like before, the actin density regions within the domain that were close to the imposed dirichlet boundary condition raise more quickly when compared to the other regions that take much longer to develop and approach the solution. These graphs are behaving as expected and consistently with the theory implying that this implementation was programmed correctly. A full copy of the main code implementation can be found in the appendix.

We will now look at the convergence of several radial node values througtht the time domain. These radial values will be taken along the y symmetry line. The first node taken will be node 5, this node is close to the free radial end of the domain with no dirichlet boundary condition at  $r=15$ . This will help capture the rates of density change and convergence of towards the free end. THE second node will be taken at node 20, this node is very close to  $r=20$  in the middle of the domain and will help caprure the behavior of the solution in this region. The final node tracked will be node 35 close to the imposed dirichlet boundary condition of  $80\mu\text{M}$  and will capture the behavior in that region. The results of this nodal evolution study can be found below in figure 7.

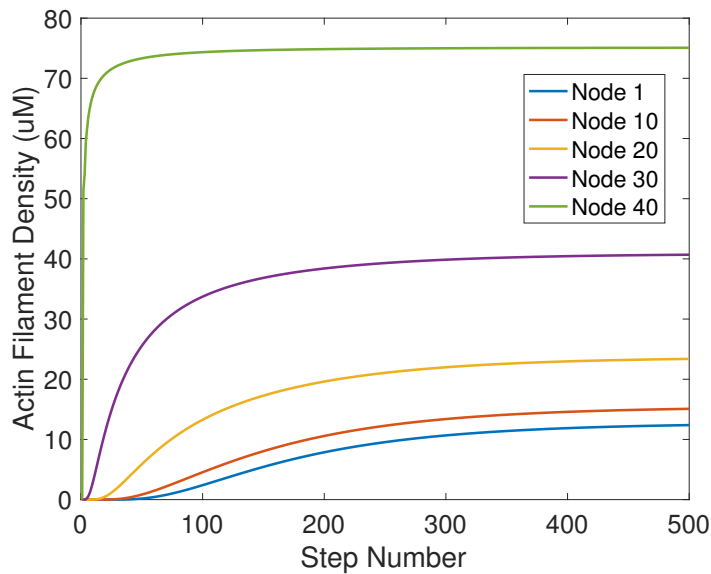


Figure 7: Actin Filament Density F Nodal Evolution Study

On the previous page we see the previously described nodal evolution study of the Actin filament density at various points of the domain. The nodes are spaced 10 elements apart in the radial direction down the center of the domain. Node 1 being at  $r=15$  and node 40 being 1 element away from  $r=25$ . We can notice that initially there is no rise in filament density near radial nodes 1, 10, and 20 however, there is a sharp increase in Actin density near the imposition of the dirichlet boundary condition at  $r=25$ . This is shown by the immediate violent evolution of node 40 shortly followed by radial node 30. Radial node 20 is the next to move away from zero followed by radial nodes 10 and 1. It is worth radial node 1 leaves zero last and also takes the longest to begin convergence. Radial node 40 is the first to shoot away from zero and exhibits signs of convergent behavior first out of the 5 nodes tracked. Radial nodes 10,20, and 30 behave in the middle ground of these two. This behavior is expected and is another excellent indication that the implementation is behaving correctly.

We will now examine the results produced about the evolution of the Monomer Density  $G$  in the domain. These results can be found below in figure 8.

Figure 8: Monomer Density  $G$  Solution at Various Time Steps  $\Delta t=0.02$

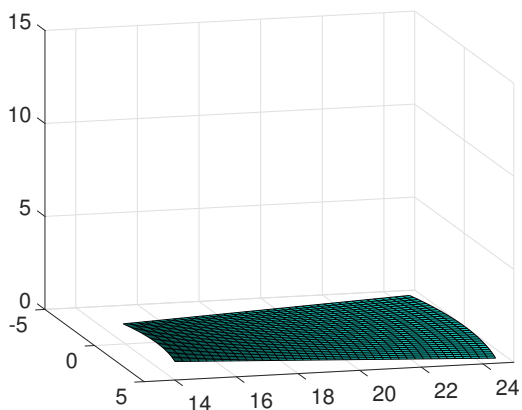


Figure 8a: Time Step 1

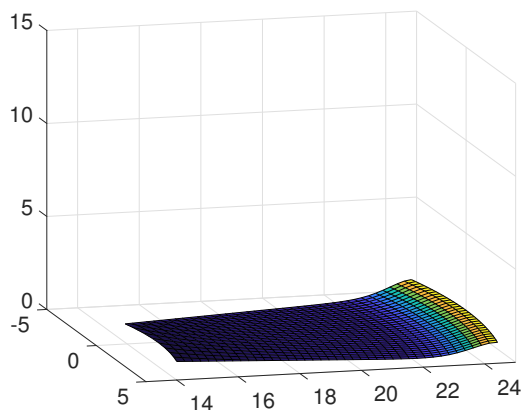


Figure 8b: Time Step 5

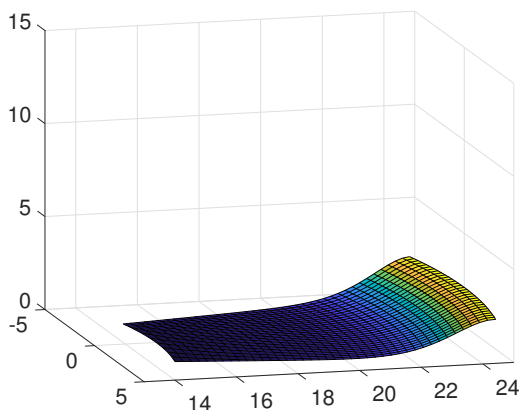


Figure 8c: Time Step 10

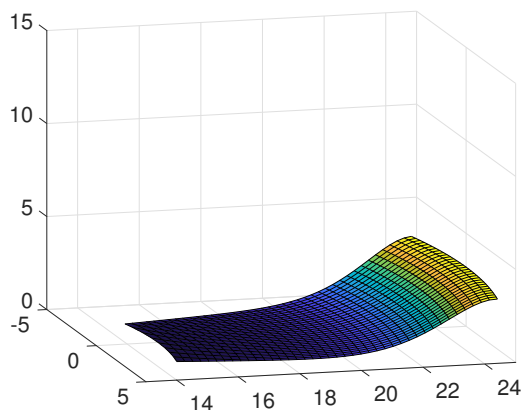


Figure 8d: Time Step 15

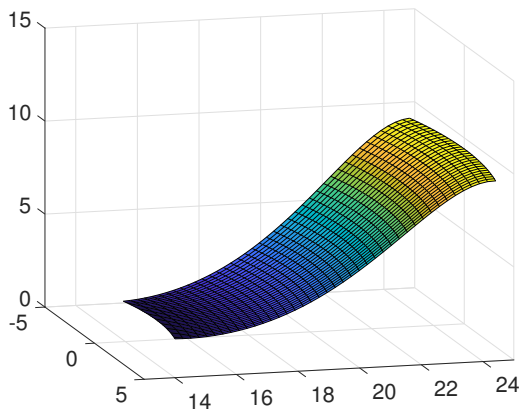


Figure 8e: Time Step 75

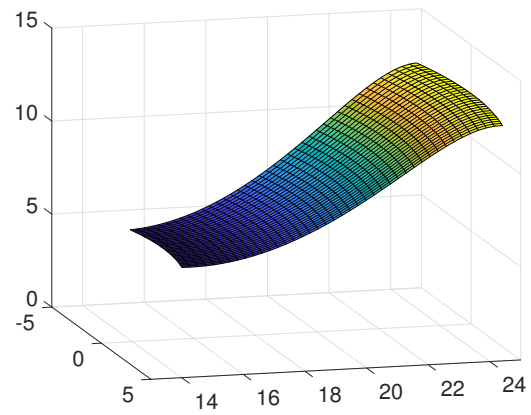


Figure 8f: Time Step 500

Above we have the previously introduced results, the first thing to notice is that the evolution of the Monomer density first increases near the imposition of the imposed dirichlet boundary on the Actin filament density. While the region that begins to increase first coincides with the corresponding region to the actin filament density profile, the nature of the evolution is different. The Monomer density profile increases much more smoothly.

To support this discussing, on the next page in figure 9 is a nodal evolution study of radial nodes along the y axis at various intervals. Node one is at  $r=15$  and node 40 is right next to the outer edge of the radius at  $r=25$ . Notice how the slope of the evolution of this outer radius at node 40 is more gradual than the evolution of node 40 in the Actin filament density development. The rest of the nodes behave fairly similarly until they begin to converge. All in all this behavior is expected and furthers the argument that the implementation is correct.

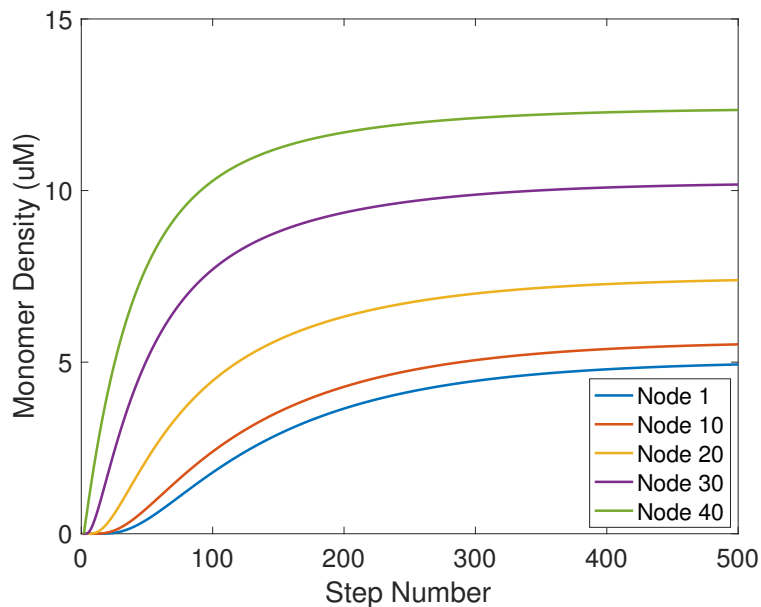


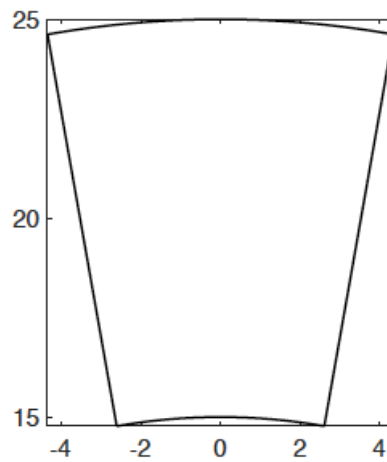
Figure 9: Monomer Density G Nodal Evolution Study

## 2 Stokes Problem

We will now solve a stokes problem to obtain the velocity and pressure distribution of the fluid surrounding the actin filaments and monomers. The relevant stokes equations can be found below, along with the prescribed boundary conditions and the domain being taken into consideration. The viscosity of the fluid is  $\nu=1000$  pN·s/ $\mu\text{m}$

$$\begin{cases} \nabla \cdot \boldsymbol{\sigma} = \mathbf{0} & \text{in } \Omega \\ \nabla \cdot \mathbf{u} = 0 & \text{in } \Omega \end{cases}$$

$$\begin{aligned} u_r(r = 15) &= -0.15, \quad u_\theta(r = 15) = 0 \\ u_r(r = 25) &= -0.30, \quad u_\theta(r = 25) = 0 \end{aligned}$$



## 2.1 Discretization of The Stokes Equations

We will now discretize the stokes equations using a standard Galerkin space discretization scheme.

The Galerkin Space Discretization Scheme

$$\begin{cases} \nabla \cdot \sigma = 0 & \text{in } \Omega \\ \nabla \cdot u = 0 & \text{in } \Omega \end{cases}$$

Starting with the equation for momentum conservation

$$1) \quad \rho(v_t + (v \cdot \nabla)v) = \nabla \cdot \sigma + \rho b$$

and expanding

$$2) \quad \nabla \cdot \sigma = \nabla \cdot (-pI + 2\mu \nabla^s v)$$

substituting 2 into 1 & deviding by  $\rho$

$$3) \quad v_t + (v \cdot \nabla)v - b = -\nabla p + 2\mu \nabla \cdot \nabla^s v$$

$$4) \quad \cancel{v_t + (v \cdot \nabla)v - b} = -\nabla p + \nu \nabla^2 v + \cancel{\nu \nabla(\nabla \cdot v)}$$

and cancelling out the relevant terms leaves us with

$$5) \quad \begin{cases} -\nu \nabla^2 v + \nabla p = 0 & \text{in } \Omega \\ \nabla \cdot u & \text{in } \Omega \end{cases}$$

applying Galerkin to both

$$6) \quad -(w, \nu \nabla^2 v) + (w, \nabla p) = 0 \quad \& \quad (q, \nabla \cdot u) = 0$$

integrating by parts the first term in 6

$$-(w, \nu \nabla^2 v) = -\cancel{(w, \nu \nabla v)} + (\nabla w, \nu \nabla v)$$

final weak form

$$\begin{cases} \int_{\Omega} \nabla w : \nu \nabla v \, d\Omega + \int_{\Omega} w \cdot \nabla p \, d\Omega = 0 & \text{in } \Omega \\ \int_{\Omega} q \nabla \cdot v \, d\Omega = 0 & \text{in } \Omega \end{cases}$$

## 2.2 Creation of the Meshes

For this problem it is necessary to generate 2 separate meshes. One to capture the pressure solution and another to capture the velocity solution. We will use a standard mesh of bi-linear quadrilateral elements to capture pressure behavior. This is the same type of mesh used in the previous problem. However, for the velocity, we will implement a more accurate mesh of  $Q_2Q_1$  elements. These elements have an additional node in the middle of each side of the quadrilateral elements and another node in the center of the element for a total of 9 nodes per element compared to 4 for the previously used bi-linear quadrilateral elements. The pressure and xy-velocity meshes generated for this problem consist of 10 elements in the radial direction and 10 in the theta direction. A mesh of 10 by 10 will also be used to capture velocity field vectors and can be found below in figures 7 and 8 for reference.

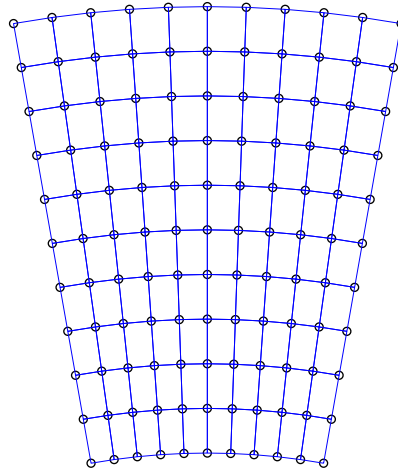


Figure 7: Generated Bi-Linear Quadrilateral Pressure Mesh

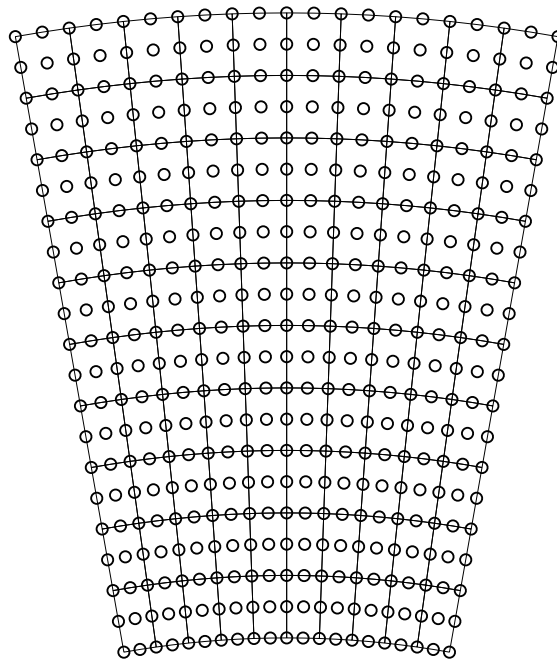


Figure 8: Generated  $Q_2Q_1$  Quadratic Velocity Mesh

The velocity mesh on the previous page was generated using a code modification of the function used to generate the simpler Bi-Linear Quadrilateral Mesh. This code expands the 4 node numbering scheme to the quadratic 9 node numbering scheme using variables related to the spacing of subsequent radial rows of nodes. The important code modification for the nodal connectivities can be found below in figure 9.

```

36 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
37
38 %GENERATE NODAL CONNEVTIVITIES
39
40 ind_ele = 0 ;
41 for jj = 1:(Nr/2)
42     for ii = 1:(Ntheta/2)
43         ind_ele = ind_ele +1 ;
44         ele_data(ind_ele,:) = [ind_ele , Node_number(ii+2,jj)+(ii-1)+(jj* VSPACE- VSPACE),...
45                               Node_number(ii+2,jj+2)+(ii-1)+(jj* VSPACE- VSPACE),...
46                               Node_number(ii,jj+2)+(ii-1)+(jj* VSPACE- VSPACE),...
47                               Node_number(ii,jj)+(ii-1)+(jj* VSPACE- VSPACE),...
48                               Node_number(ii+2,jj+1)+(ii-1)+(jj* VSPACE- VSPACE),...
49                               Node_number(ii+1,jj+2)+(ii-1)+(jj* VSPACE- VSPACE),...
50                               Node_number(ii,jj+1)+(ii-1)+(jj* VSPACE- VSPACE),...
51                               Node_number(ii+1,jj)+(ii-1)+(jj* VSPACE- VSPACE),...
52                               Node_number(ii+1,jj+1)+(ii-1)+(jj* VSPACE- VSPACE)] ;
53
54     end
55 end
56 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

Figure 9: Code used to Generate  $Q_2Q_1$  Quadratic Velocity Mesh Connectivities

### 2.3 Imposition of Boundary Conditions

Before we can solve the stokes problem we must impose the boundary conditions. These boundary conditions along with their code implementation can be seen below in figure 10.

$$\begin{aligned}
 u_r(r = 15) &= -0.15, & u_\theta(r = 15) &= 0 \\
 u_r(r = 25) &= -0.30, & u_\theta(r = 25) &= 0
 \end{aligned}$$

```

100 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
101
102 %IMPOSITION OF THE BOUNDARY CONDITIONS
103
104 BC_THETA1 = 0;
105 BC_THETA2 = 0;
106 BC_RADIAL1 = -0.15;
107 BC_RADIAL2 = -0.3;
108 VELOCITY_BC_X1 = BC_THETA1*sin(pi/2- THETA_VAR ) + BC_RADIAL1*cos(pi/2-THETA_VAR);
109 VELOCITY_BC_X2 = BC_THETA2*sin(pi/2-THETA_VAR) + BC_RADIAL2*cos(pi/2-THETA_VAR);
110 VELOCITY_BC_Y1 = -BC_THETA1*cos(pi/2-THETA_VAR) + BC_RADIAL1*sin(pi/2-THETA_VAR);
111 VELOCITY_BC_Y2 = -BC_THETA2*cos(pi/2-THETA_VAR) + BC_RADIAL2*sin(pi/2-THETA_VAR);
112
113 B_STEP = [VELOCITY_BC_X1' VELOCITY_BC_Y1';...
114           VELOCITY_BC_X2' VELOCITY_BC_Y2'];
115 b_DirBC =reshape(B_STEP',nDir,1);
116
117 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```

Figure 10: Code used to impose the boundary conditions

## 2.4 Solution Results

We will now examine the solution generated by the implementation. Results for both the velocity field and pressure field have been generated and will be analyzed below. We will first examine the results for the velocity field produced from the mesh of 10 radial and 10 theta elements for a clear initial visualization of the resulting velocity field. This can be seen below in figure 11.

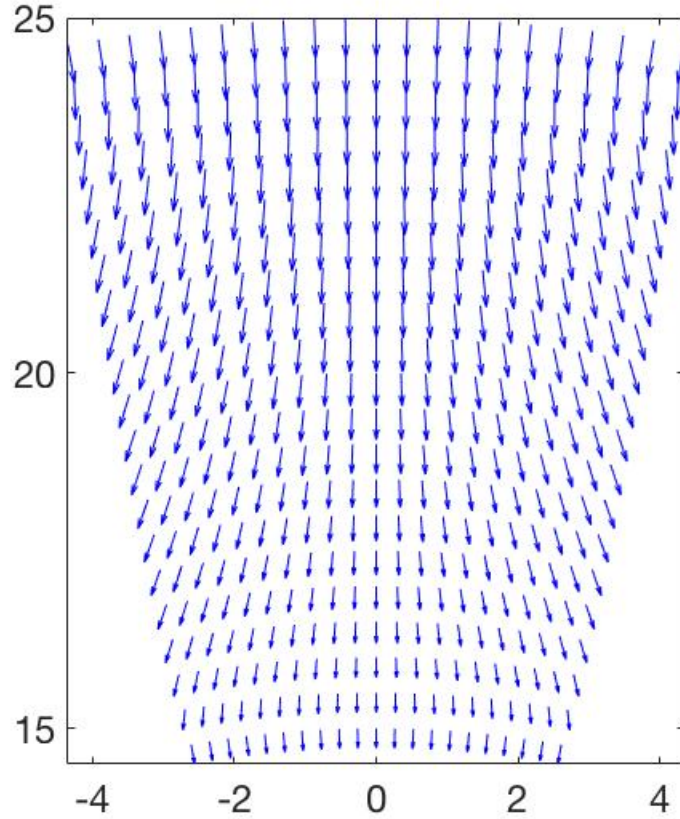


Figure 11: Solution Vectorized Velocity Field

Above we can see the vectorized results for the velocity output of the stokes solution. We first notice the vectors at the maximum and minimum radius values flowing in the negative radial direction as was imposed by the boundary conditions. The longer length of the vectors at  $r=25$  coincide with the higher magnitude of enforced velocity in the negative radial direction. Additionally, all of the vectors begin to point away from from the central symmetrical  $y$  axis as they approach the central radial value of 20 and the further away they get from their respective dirichlet boundary conditions are  $r=15$  and  $r=25$ . The physical reasoning behind this is due to the fact that a higher magnitude of negative radial velocity is imposed at the outer radial boundary of  $r=25$ . It is important to point out that this side has the longer arc length of the two sides with curvature and negative radial velocity dirichlet boundary conditions imposed. This side also has more fluid flowing per time into the domain than the lower radial boundary (with shorter arc length and lower magnitude of negative radial velocity) is capable of moving out of the domain. Due to the necessity of upholding conservation of mass combined with the previously stated observations of the fluid transport, the excess fluid mass is forced to flow out of the domain along the straight sides on the left and right hand sides of the figure. This is a good indication of that the implementation is behaving correctly. Next we will look at the individual  $y$  and  $x$  velocity profiles. These can be found on the following page in figures 12 and 13.



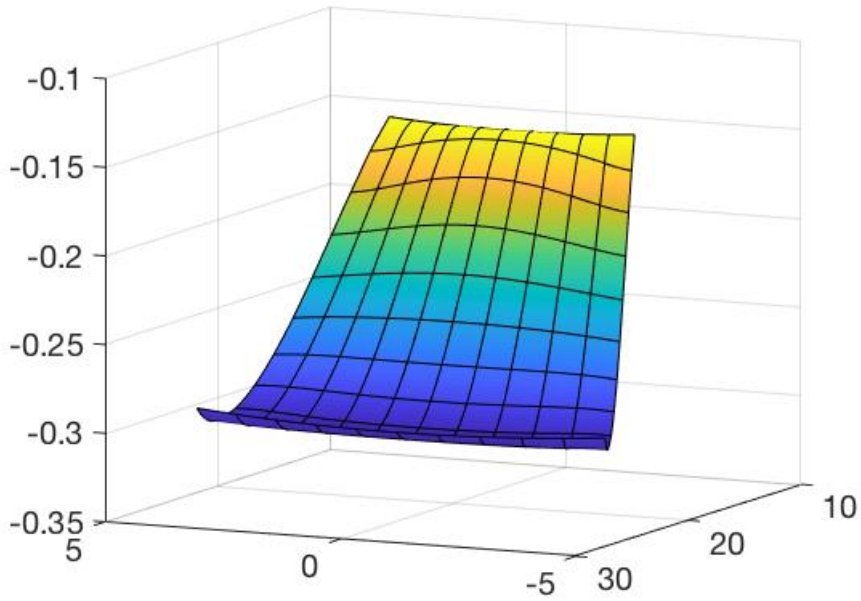


Figure 12: Y-Direction Velocity Profile

Above is the resultant y directional velocity profile. It is immediately apparent that there is increasing magnitude of negative radial velocity with increasing radius. This is consistent with the physical phenomenon previously discussed. However, it is important to note that even though it may look like a plot of radial velocity, this graph is not an exact representation of it, these values of radial velocity and y velocity will only coincide along the symmetric line of  $x=0$ . It is for this reason we can see slight z-directional curvature along the horizontal lines separating the elements in the figure. This y-directional velocity profile has contributions from both radial and theta velocities, and while radially dominant, the theta has some contribution. We will now take a look at the x- direction velocity profile in figure 13 below..

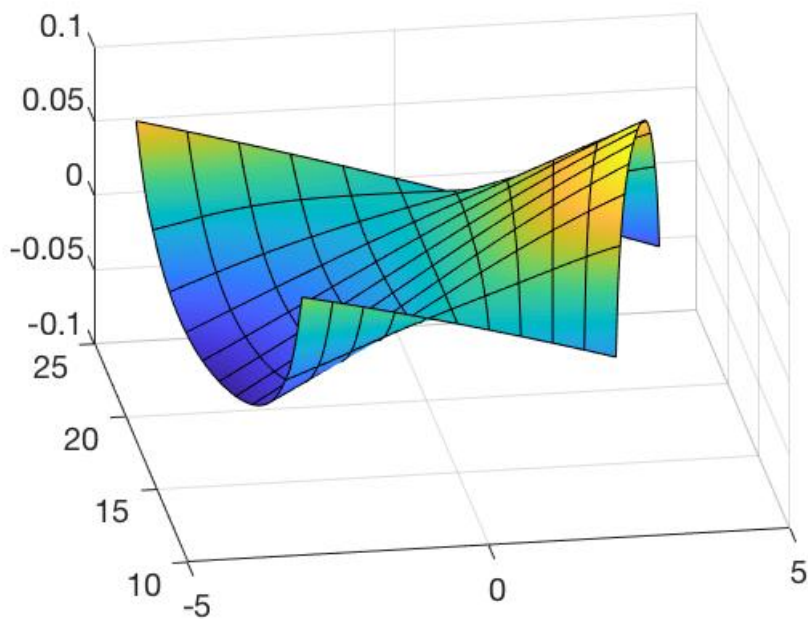


Figure 13: X-Direction Velocity Profile

On the previous page we see the x directional velocity profile. Once again, the only place the theta velocity and x velocity are identical is when they are zero along the  $x=0$  symmetry line. Additionally we can see regions of positive x velocity on the right side of the symmetric line with negative x velocity on the opposing side. This is a result of the radial velocity along the right boundary having positive x components while along the left side the x component is negative. Exceptions to this generalization appear at and close to the radial values where the dirichlet boundary conditions are imposed. On the left hand side near the imposition of the dirichlet boundary conditions we see positive contribution to the x velocity. This is a result of the x component of the initially prescribed radial velocity. The same is true on the opposing side with negative x components of the initially prescribed radial boundary conditions. We will now take a look at the resultant pressure variations. These can be found below in figure 14.

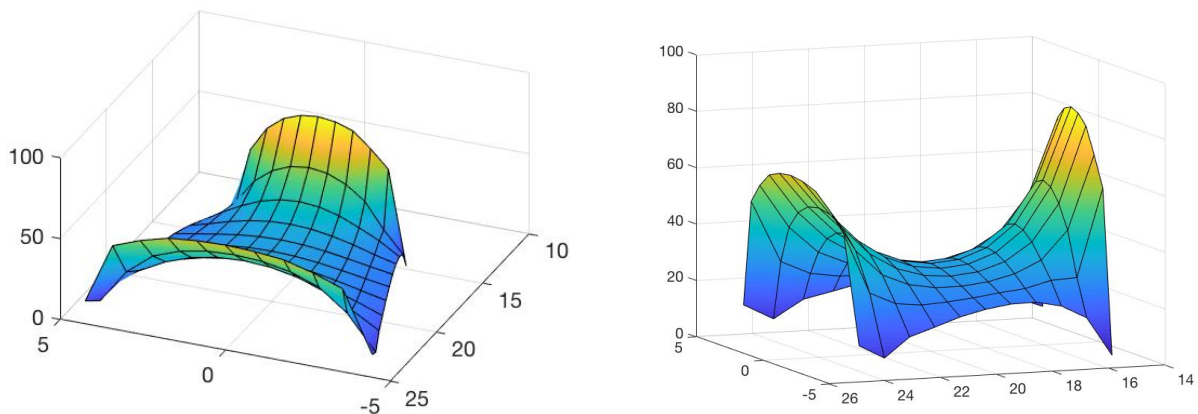


Figure 14: 2 Views of the Resultant Pressure Distribution

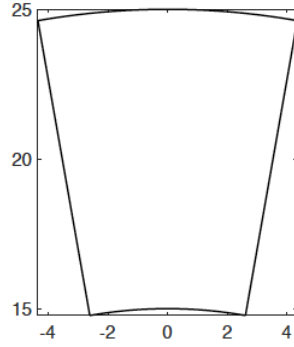
Above we have results for the pressure distribution from the implemented stokes system. The first thing worth noting is the presence of the confined flow causing the pressures in the four corners of the domain to approach zero. Next, we can begin to notice the pressure distributions at and close to the 2 imposed radial values of dirichlet boundary conditions  $r=15$  and  $r=25$ . We can notice from the right hand graph that at the boundary with the lower magnitude of imposed negative radial velocity that there is higher pressure when compared to the larger radial boundary. This is consistent with the expected behavior from bernoullis principle that when considering confined flow, the regions of higher velocity experience lower pressure when compared to the regions of lower velocity which experience higher pressure. All of this behavior is consistent with the theory and is an excellent indication that the implementation is correct.

### 3 Coupled Problem

We will now considered a coupled problem describing the evolution of the actin filament and monomer densities. The equation describing the evolution of monomers densities  $G$  does not involve any convective transport and, therefore, only the fluid around the fibers has to be considered. This fluid is modelled using the equations of a quasi-steady viscous fluid. Moreover, due to the presence of actin fibers, the incompressibility constrain is dropped and pressure is neglected. The equations governing the coupled problem can be written as follows over the same domain considered where  $\nabla \cdot \sigma_{\mathbf{m}}$  and  $\mathbf{T}_{\mathbf{m}}$  are surface forces on the leading edge.

$$\begin{cases} \nu \nabla \cdot (\nabla^s \mathbf{u}) + \nabla \cdot \sigma_{\mathbf{m}}(F) + \mathbf{T}_{\mathbf{m}}(\mathbf{u}) = \mathbf{0} & \text{in } (0, T) \times \Omega \\ \frac{\partial F}{\partial t} = -\mathbf{u} \cdot \nabla F + D_F \nabla^2 F - \sigma_F F & \text{in } (0, T) \times \Omega \\ \frac{\partial G}{\partial t} = D_G \nabla^2 G - \sigma_G G + \hat{\sigma}_{GF} F & \text{in } (0, T) \times \Omega \end{cases}$$

$$\begin{aligned} u_r(r = 15) &= -0.15, \quad u_\theta(r = 15) = 0 \\ u_r(r = 25) &= -0.30, \quad u_\theta(r = 25) = 0 \end{aligned}$$



As was the case in the transport problem, the filament density is constant at the upper boundary:  $F(r = 25) = 80 \mu\text{M}$ . No flux boundary conditions are considered for  $F$  everywhere else and for  $G$  on the boundary. The problem is considered with a velocity field  $\mathbf{u}(x,y) = 1/1500 (rx, ry) \mu\text{m/s}$ , where  $(x,y)$  are the points coordinates and  $r = \sqrt{x^2 + y^2}$ .

### 3.1 Discretization of the First Equation

Here we will discretize the first of the three equations presented in the problem. The other two have been previously discretized in the first part of this report and will not be repeated as to avoid redundancy. Below we can find the aforementioned discretization of the first equation.

Galerkin Formulation 1<sup>st</sup> eqn.

$$\nabla \nabla \cdot (\nabla^s u) + \nabla \cdot \sigma_m(F) + T_m(u) = 0 \text{ in } (0, T) \times \Omega$$

applying galerkin's formulation

$$\int_{\Omega} w \nabla \nabla \cdot (\nabla^s u) d\Omega + \int_{\Omega} w \nabla \cdot \sigma_m(F) d\Omega + \int_{\Omega} w T_m(u) d\Omega = 0$$

and integrating by parts the first term

$$\int_{\Omega} w \nabla \nabla \cdot (\nabla^s u) d\Omega = - \int_{\Omega} \nabla w : (\nabla \nabla^s u) + \int_{\Gamma} w \cdot \nabla \nabla \cdot (\nabla^s u) d\Gamma$$

and rearranging gives us...

$$- \int_{\Omega} \nabla w : (\nabla \nabla^s u) d\Omega + \int_{\Omega} w \nabla \cdot \sigma_m(F) d\Omega + \int_{\Omega} w T_m(u) d\Omega = 0$$

and substituting to get the system

$$Ku + T_f F + T_u u = 0$$

where  $T_f$  &  $T_u$  are matrices resulting from

the discretization of  $\int_{\Omega} w \nabla \cdot \sigma_m(F) d\Omega$  and  $\int_{\Omega} w T_m(u) d\Omega$

### 3.2 Mesh Generation

For this problem we will use bilinear quadrilateral elemental meshes for both pressure and velocity. The meshes used in computation of the solution will have 30 elements in the theta direction and 40 elements in the radial direction. Below in figure 15 we have presented some sample meshes of 5x5 for the purpose of visualizing the mesh.

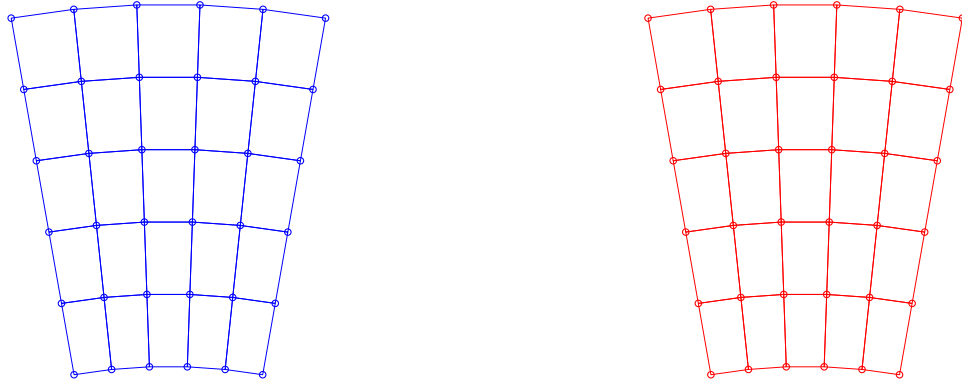


Figure 15: Pressure (left) and Velocity (right) Sample Meshes

### 3.3 Solution Results

Below we can see the some plots of the Actin density during the evolution of the solution at various time steps. This solution was generated using nested loops to calculate and produce values for F and G, the full code with commentary can be found in the appendix.

Figure 16: Actin Density Transient solution at various time steps  $\Delta t=0.02$

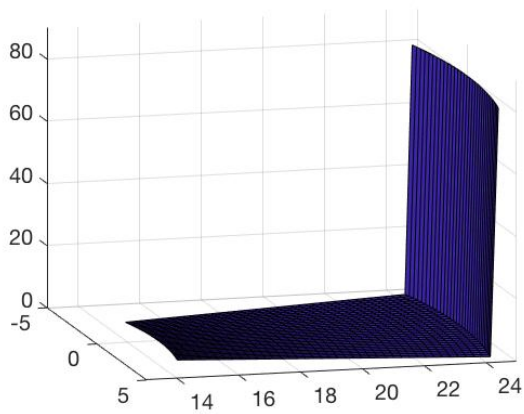


Figure 16a: Time Step 1

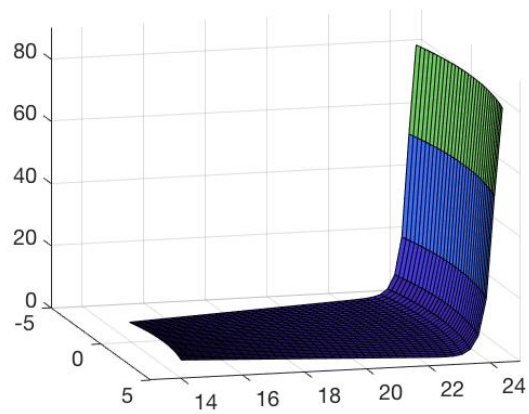


Figure 16b: Time Step 2

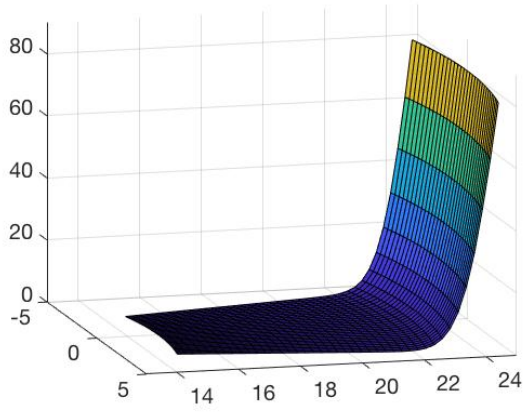


Figure 16c: Time Step 5

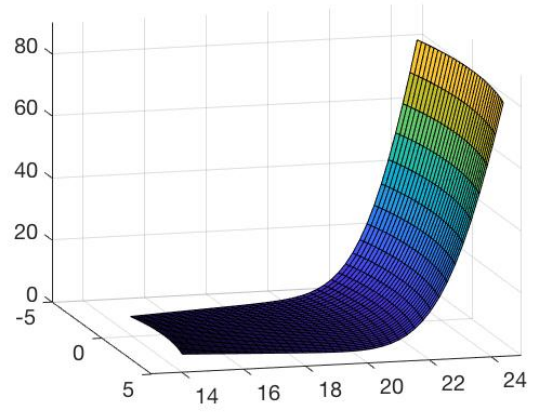


Figure 16d: Time Step 15

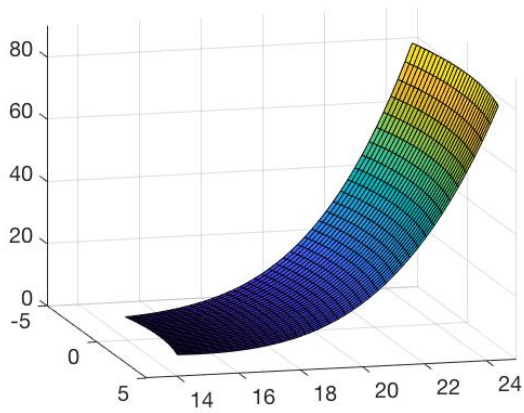


Figure 16e: Time Step 75

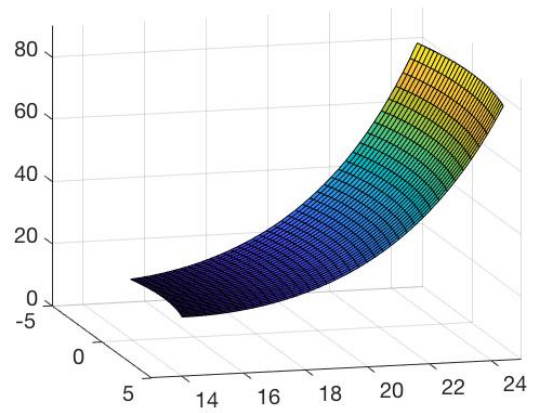


Figure 16f: Time Step 500

The first thing that can be noticed is how extremely similar this solution is to the one previously obtained in the first transport problem. We can notice the successful imposition of the boundary conditions in Figure 16a as all of the Actin filament density values ( $F$ ) on the further radial boundary ( $r=25$ ) at time step 1 are once again firmly held at  $80 \mu\text{M}$  and stay fixed that way over the entire time domain. We can also notice how the transient solution develops as time progresses. The density regions closest to the dirichlet boundary condition of  $80 \mu\text{M}$  increase more rapidly than other regions due to the global convection velocity field. To support this discussion, below in figure 17 is the comparison of the previously seen nodal evolution of  $F$  from the Transport problem (left) to that of of the coupled problem solved in this section of the report (right). These graphs are also virtually identical. The reasons for these extreme similarities is that essentially the boundary conditions of the equations are the same with the exception with velocity dirichlet boundary conditions being imposed at the inner radial and outer radial bounds. However the impact of this is trivial on the transient solution of the evolution of the Actin Filament Density  $F$ .

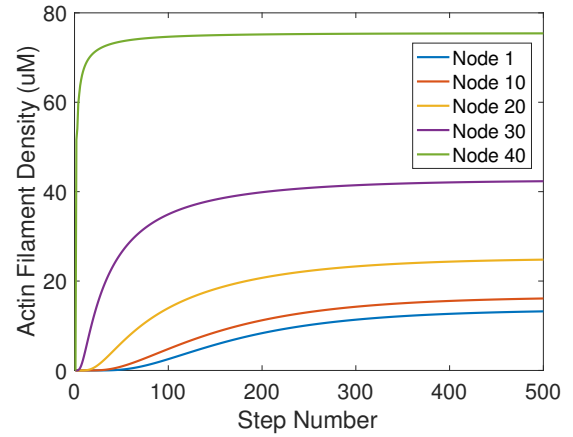
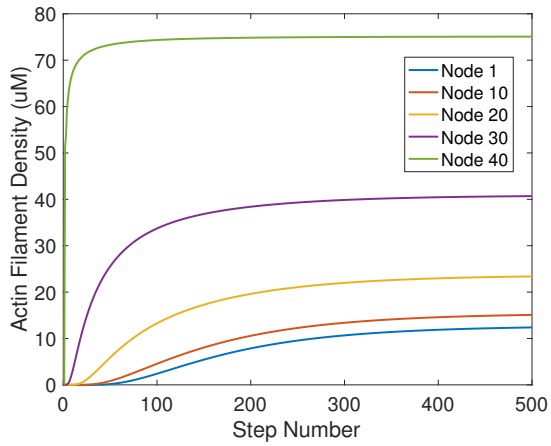


Figure 17a: Transport Problem F Nodal Evolution    Figure 17b: Coupled Problem F Nodal Evolution

We will now take a look at the evolution of Monomer density  $G$  over time. Below we once again have 6 snapshots of the evolution of the field over the time domain.

Figure 18: Monomer Density  $G$  Final transient solution at various time steps  $\Delta t=0.02$

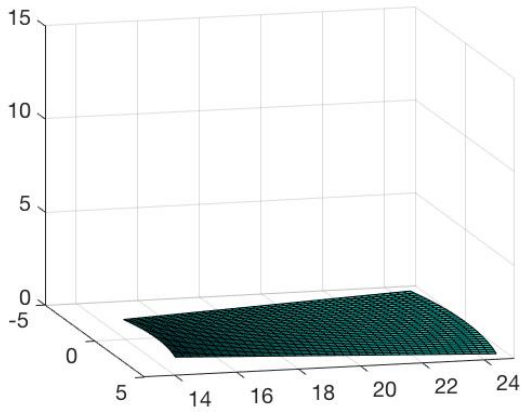


Figure 18a: Time Step 1

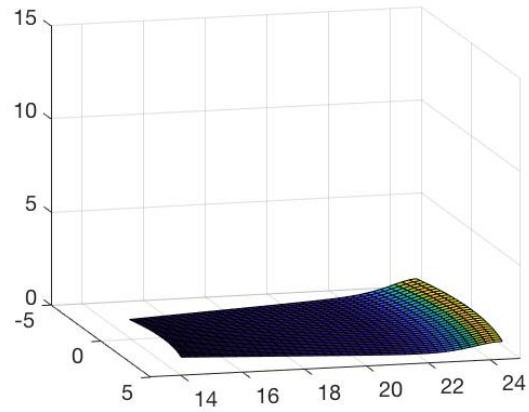


Figure 18b: Time Step 2

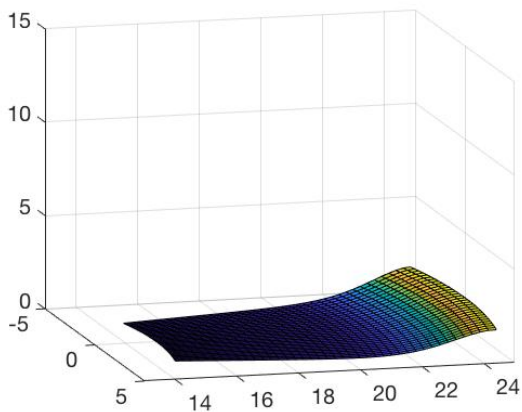


Figure 18c: Time Step 5

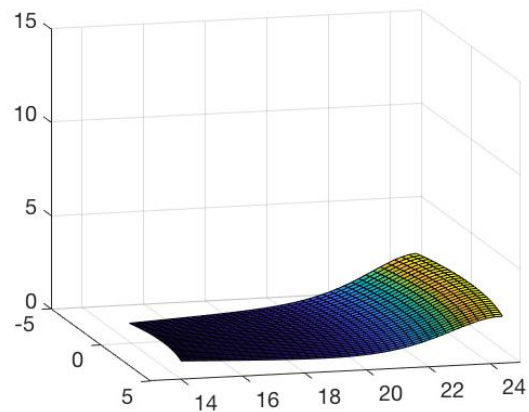


Figure 18d: Time Step 15

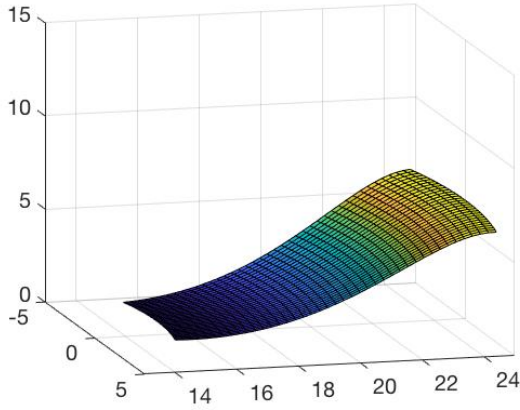


Figure 18e: Time Step 75

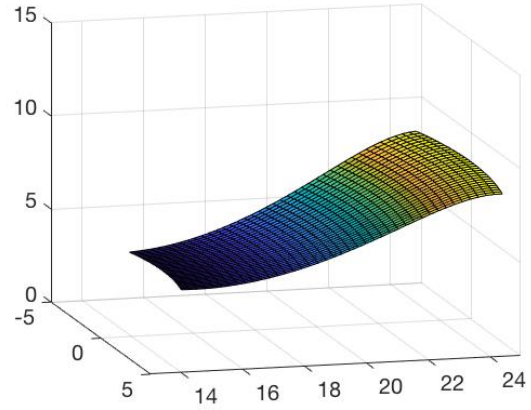


Figure 18f: Time Step 500

Above we once again have the evolution of the Monomer Density  $G$  over the time domain. As was the case with the solution of  $G$  for the transport problem, the first thing to notice is that the evolution of the Monomer density first increases near the imposition of the imposed dirichlet boundary. While the region that begins to increase first coincides with the corresponding region of the Actin filament density profile, the nature of the evolution is much more smooth, additionally we can see that the plot converges on values of Monomer density  $g$  that are lower by a wide margin. To support this claim, below in figure 18 is the comparison of the previously seen nodal evolution of Monomer Density  $G$  from the Transport problem (left) to that of of the coupled problem solved in this section of the report (right). It is obvious that the Monomer Density  $G$  solution of the coupled problem converges to a lower value than that of the transport problem. This is most likely due to the much stronger role that diffusion plays in this problem leading to to the diffusion of density that showed up in the previous solution.

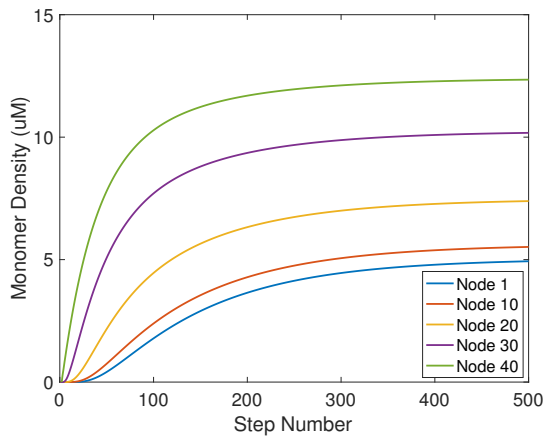


Figure 19a: Transport Problem  $G$  Nodal Evolution

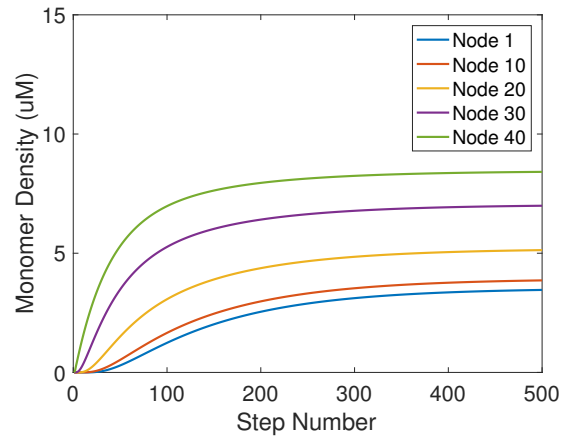


Figure 19b: Coupled Problem  $G$  Nodal Evolution

We will now examine the resultant velocity field arising from the solution of the coupled problem. This can be seen on the next page in figure 20.



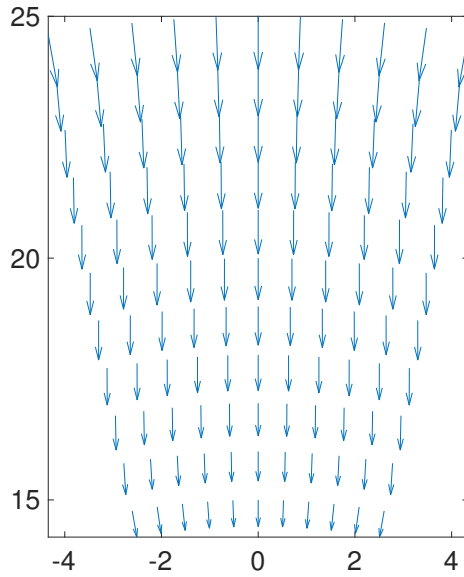


Figure 20: Resultant Velocity Field

Above in figure 20 we can see the velocity field resulting from the solution of the coupled problem. We can notice that this velocity field looks very similar to the one obtained from the previous stokes solution. This is due to the same dirichlet boundary conditions being applied to the inner and outer radial bounds. It is important to notice that the theta directional velocity towards the left and right center of the domain is much less than previously seen in the stokes problem. This is primarily due to the surface forces on the leading edge as well as the consideration of compressible flow. A possible physical explanation for this is that since this fluid is now able to compress, the inner radial dirichlet bound with negative prescribed velocity in the radial direction can move more fluid per unit length out of the domain than it previously could since it is possible to transport compressed fluid. This would result in less fluid needing to be transported out of the domain through the straight sides of the domain.

We will now examine a plot of the resulting y velocity, this can be found below in figure 21.

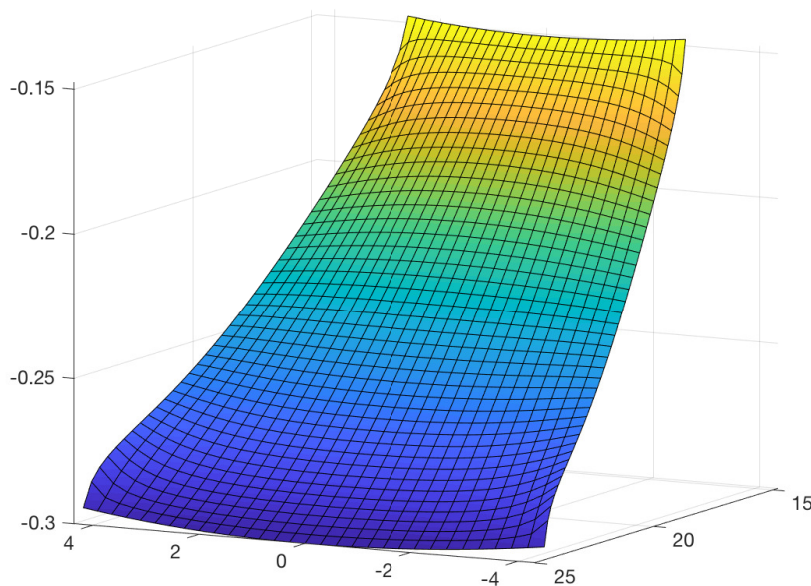


Figure 21: Y-Velocity Results

Above we have the resulting y velocity plot. The first thing we can notice is the successful imposition of the velocity dirichlet boundary conditions on the inner and outer radial bounds. The inner bound ( $r=15$ ) exhibits its prescribed value of  $u_r = -0.15$  and the outer bound ( $r=25$ ) exhibits its prescribed velocity of  $u_r = -0.3$ . In general, this solution is fairly similar to to the one previously seen in the stokes solution, however there are a few areas worth pointing out. The next thing of interest is the symmetric behavior of both sides of the domain, the y axis velocity values are equivalent to the radial and theta velocity values along this axis. This is consistent with what is to be expected from a radially defined domain. Next we want to comment about the unique behaviors exhibited at the corners of the graph. We can see a small upward spike along the sides of the straight bounds of the domain near the lower corners. This behavior was not present in the stokes y velocity solution. We see the opposite effect at the top corners along the sides of the boundary with a slight spike in the negative z direction close to the corners. This behavior is most likely due to the addition of surface forces along the leading edge and compressible flow considerations not present in the previous problem. We will now examine the x velocity results below in figure 22.

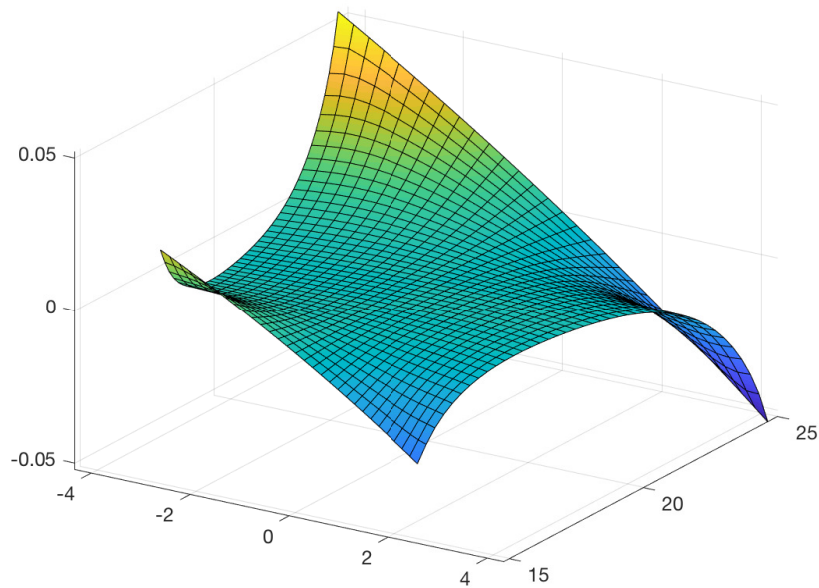


Figure 22: X-Velocity Results

Above we have plotted the x component of the resulting velocity vectors. We can see that there is very little x directional velocity in a large portion of the center of the domain. However as we move towards the corners of the domain we see much more x directional velocity. This can be attributed to the prescribed inflow and outflow velocity conditions. On the inflow at the further radial bound ( $r=25$ ) we have a higher magnitude of negative radial velocity prescribed. And since this domain is radially shaped, the corners happen to be the points furthest away from the symmetric axis, and therefore will have the largest x direction contributions. The prescribed inflow and outflow on the left side of the domain have positive x contributions leading to the behavior exhibited by this graph on the left side with high magnitudes in the corners. On the other hand, the right side has negative x velocity contributions from the same radially prescribed velocities, leading to the negative behavior on that part of the domain and the negative spikes of x velocity near the respective corners. All of these results are reasonable and imply that the implementation is behaving correctly.

## 4 APPENDIX

```
1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2
3 %THIS PROGRAM SOLVES THREE PROBLEMS RELATED TO ACTINS ROLE IN CELL
4   MOTILITY
5
6 %BY ALEXANDER KEISER
7
8
9  clc
10 clear all
11 close all
12
13
14 disp('1 for Transport Problem')
15 disp('2 for Stokes Problem')
16 disp('3 for Coupled Problem')
17 problem=input('Enter Problem Type')
18
19 if problem==1;
20
21 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
22 %MATERIAL PARAMETERS
23 THETA=2/3; SIGMA_GF=0.5; SIGMA_G=2; SIGMA_F=0.25; D_G=15; D_F=5;
24           %s^-1           %s^-1           %s^-1   %um/s   %um/s
25 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
26
27 %GENERATION OF THE MESH
28
29 %NUMBER OF ELEMENTS IN RADIAL DIRECTION
30 N_R = 40;
31
32 %NUMBER OF ELEMENTS IN THETA DIRECTION
33 N_THETA = 30;
34
35 %CREATE THE MESH
36 [X,T] = createMesh1(N_R,N_THETA);
37
38 %BILINEAR QUADRATIC ELEMENTS
39 elem = 0;
40
41 %PLOT THE MESH
42 figure(25)
43 plotMesh(T,X,elem,'k');
44
45
46 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
47
48 %SETS DEGREES OF FREEDOM
49 DEGREES_OF_FREEDOM = size(X,1);
50 ZERO_ARRAY=zeros(DEGREES_OF_FREEDOM,1);
```

```

51
52
53 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
54
55 %CALCULATION OF GLOBAL CONVECTIVE VELOCITY
56
57 CONVECTIVE_VELOCITY = velocity(X);
58
59 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
60
61 %GENERATION OF INITIAL TIME VECTORS
62
63 F_VECTOR = zeros(DEGREES_OF_FREEDOM,1);
64 G_VECTOR = zeros(DEGREES_OF_FREEDOM,1);
65
66
67 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
68
69 % FOUR POINT GAUSS QUADRATURE AND SHAPE FUNCTIONS
70
71 ngaus = 4;
72 [pospg ,wpg] = Quadrature1(ngaus);
73 [N,Nxi ,Neta] = ShapeFunc1(pospg);
74
75 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
76
77 % COMPUTATION OF THE MATRICES FROM DESCRETIZATION
78
79 M = CreMassMat1(X,T, pospg ,wpg,N, Nxi , Neta);
80 C = CreConvMat1(X,T, CONVECTIVE_VELOCITY, pospg ,wpg,N, Nxi , Neta);
81 K = CreStiffMat1(X,T, CONVECTIVE_VELOCITY, pospg ,wpg,N, Nxi , Neta);
82
83 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
84
85 % DEFINITION OF TIME PARAMETERS
86 END_TIME = 10;
87 NUMBER_TIME_STEPS = 499;
88 DT = END_TIME/NUMBER_TIME_STEPS;
89
90 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
91
92 %IMPLEMENTATION OF GALERKIN TIME AND SPACE DESCRETIZATION
93
94 A1 = M + THETA*C*DT + THETA*D_F*K*DT + THETA*SIGMA_F*M*DT;
95
96 B1 = -C*DT - D_F*K*DT - SIGMA_F*M*DT;
97
98 A2 = M + THETA*D_G*K*DT + THETA*SIGMA_G*M*DT;
99
100 B2 = -D_G*K*DT - SIGMA_G*M*DT;
101
102 C2 = SIGMA_GF*M*DT;
103

```

```

104 F_1 = ZERO_ARRAY;
105
106 F_2 = ZERO_ARRAY;
107
108
109 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
110
111 %IMPOSITION OF THE BOUNDARY CONDITIONS
112
113 %DEFINE THE RADIUS OF EACH NODE IN THE DOMAIN
114 NODAL_RADIUS = sqrt(X(:,1).^2+X(:,2).^2);
115
116 %FIND THE NODES WITH VALUES TO BE IMPOSED AT RADIUS OF 25
117 DIRICHLET_NODES = find(NODAL_RADIUS >= 24.9999);
118
119 %DEFINE THE VALUE OF THE DIRICHLET BC
120 DIRICHLET_VALUE = 80;
121
122 %DEFINE THE NUMBER OF NODES TO BE IMPOSED WITH A BC
123 NUMBER_IMPOSITIONS = length(DIRICHLET_NODES);
124
125 %CREATE VECTOR WITH NODES AND THEIR RESPECTIVE BD VALUES
126 DIRICHLET_VECTOR = [DIRICHLET_NODES, ones(length(DIRICHLET_NODES),1)*
    DIRICHLET_VALUE];
127
128 %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
129
130 %CALCULATION OF THE SOLUTION
131
132 %INITIALIZE MATRIX TO STORE THE LAGRANGE MULTIPLIERS
133 LAGRANGE = zeros(NUMBER_IMPOSITIONS,DEGREES_OF_FREEDOM);
134
135 %RECTIFY F VECTOR WITH DIRICHLET BC
136 F_VECTOR(DIRICHLET_VECTOR(:,1)) = 80;
137
138 %APPLY MULTIPLIER VALUES TO THE INITIALIZED LAGRANGE MATRIX
139 LAGRANGE (:,DIRICHLET_VECTOR(:,1)) = eye(NUMBER_IMPOSITIONS);
140
141 %VECTOR ADDED TO OTHER SIDE OF EQUATION TO BALANCE THE EQUATION
142 BALANCE_LAGRANGE = zeros(NUMBER_IMPOSITIONS,1);
143
144 %CREATE GLOBAL MATRIX TO CALCULATE SOLUTION
145 A_GLOBAL = [A1 LAGRANGE'; LAGRANGE zeros(NUMBER_IMPOSITIONS,
    NUMBER_IMPOSITIONS)];
146
147 %SOLUTION FOR F AND G EQUATIONS
148 for i = 1:NUMBER_TIME_STEPS
149     B_GLOBAL = [B1*F_VECTOR(:,i)+F_1; BALANCE_LAGRANGE];
150     DELTA_F = A_GLOBAL\B_GLOBAL;
151     F_VECTOR(:,i+1) = DELTA_F(1:DEGREES_OF_FREEDOM)+F_VECTOR(:,i);
152     F_VECTOR(DIRICHLET_VECTOR(:,1),i+1) = 80;

```

```

153     BG_GLOBAL = B2*G_VECTOR(:, i)+F_2+C2*F_VECTOR(:, i)-THETA*C2*DELTA_F(1:
        DEGREES_OF_FREEDOM);
154     DELTA_G = A2\BG_GLOBAL;
155     G_VECTOR(:, i+1) = DELTA_G+G_VECTOR(:, i);
156 end
157
158 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
159
160 %POSTPROCESSING
161
162 if max(F_VECTOR(:, NUMBER_TIME_STEPS+1)) < 100 && min(F_VECTOR(:,
    NUMBER_TIME_STEPS+1)) > -100
163
164     figure(1); clf;
165     [xx, yy, sol] = MatSol(X, N_THETA, N_R, F_VECTOR(:, NUMBER_TIME_STEPS+1));
166     surface(xx, yy, sol);
167     view([40, 30])
168     axis auto
169     grid on;
170
171
172
173
174     figure(3); clf;
175     [xx, yy, sol] = MatSol(X, N_THETA, N_R, F_VECTOR(:, 1));
176     surface(xx, yy, sol);
177     view([75, 15])
178     axis auto
179     grid on;
180     set(gca, 'fontsize', 20)
181     zlim([0 90])
182     ylim([13 25])
183
184
185
186     figure(4); clf;
187     [xx, yy, sol] = MatSol(X, N_THETA, N_R, F_VECTOR(:, 2));
188     surface(xx, yy, sol);
189     view([75, 15])
190     axis auto
191     grid on;
192     set(gca, 'fontsize', 20)
193     zlim([0 90])
194     ylim([13 25])
195
196
197
198     figure(5); clf;
199     [xx, yy, sol] = MatSol(X, N_THETA, N_R, F_VECTOR(:, 5));
200     surface(xx, yy, sol);
201     view([75, 15])
202     axis auto
203     grid on;
204     set(gca, 'fontsize', 20)
205     zlim([0 90])

```

```

204     ylim([13 25])
205
206
207     figure(6); clf;
208     [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,15));
209     surface(xx,yy,sol);
210     view([75,15])
211     axis auto
212     grid on;
213     set(gca,'fontsize',20)
214     zlim([0 90])
215     ylim([13 25])
216
217
218     figure(7); clf;
219     [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,75));
220     surface(xx,yy,sol);
221     view([75,15])
222     axis auto
223     grid on;
224     set(gca,'fontsize',20)
225     zlim([0 90])
226     ylim([13 25])
227
228
229     figure(8); clf;
230     [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,500));
231     surface(xx,yy,sol);
232     view([75,15])
233     axis auto
234     grid on;
235     set(gca,'fontsize',20)
236     zlim([0 90])
237     ylim([13 25])
238
239
240
241
242     figure(2); clf;
243     set(gca,'FontSize',12);
244     [C,h]=contour(xx,yy,sol);
245     clabel(C,h);
246     axis auto
247
248
249     figure(10); clf;
250     peli = moviein(NUMBER_TIME_STEPS+1);
251     axis auto
252     SOLUTION1=zeros(1,NUMBER_TIME_STEPS+1);
253     SOLUTION2=zeros(1,NUMBER_TIME_STEPS+1);
254     SOLUTION3=zeros(1,NUMBER_TIME_STEPS+1);
255     SOLUTION4=zeros(1,NUMBER_TIME_STEPS+1);
256     SOLUTION5=zeros(1,NUMBER_TIME_STEPS+1);

```

```

257
258     for n=1:NUMBER_TIME_STEPS+1
259         [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,n));
260         SOLUTION1(n)=sol(1,11);
261         SOLUTION2(n)=sol(10,11);
262         SOLUTION3(n)=sol(20,11);
263         SOLUTION4(n)=sol(30,11);
264         SOLUTION5(n)=sol(40,11);
265         surf(xx,yy,sol);
266         zlim([0 90])
267     ylim([13 25])
268         pause(0.01)
269         peli(:,n) = getframe;
270     end
271 end
272 TIMESTEPS=1:NUMBER_TIME_STEPS+1
273 plot(TIMESTEPS,SOLUTION1, 'LineWidth',2)
274 hold on
275 plot(TIMESTEPS,SOLUTION2, 'LineWidth',2)
276 hold on
277 plot(TIMESTEPS,SOLUTION3, 'LineWidth',2)
278 hold on
279 plot(TIMESTEPS,SOLUTION4, 'LineWidth',2)
280 hold on
281 plot(TIMESTEPS,SOLUTION5, 'LineWidth',2)
282 xlabel('Step Number')
283 ylabel('Actin Filament Density (uM)')
284 legend('Node 1','Node 10','Node 20','Node 30','Node 40')
285 set(gca, 'FontSize',20)
286 xlim([0 500])
287 pause(0.1);
288
289
290     figure(23); clf;
291     [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,1));
292     surface(xx,yy,sol);
293     view([75,15])
294     axis auto
295     grid on;
296     set(gca, 'fontsize', 20)
297     zlim([0 15])
298     ylim([13 25])
299
300
301     figure(24); clf;
302     [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,5));
303     surface(xx,yy,sol);
304     view([75,15])
305     axis auto
306     grid on;
307     set(gca, 'fontsize', 20)
308     zlim([0 15])
309     ylim([13 25])

```



```

310
311
312 figure(25); clf;
313 [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,10));
314 surface(xx,yy,sol);
315 view([75,15])
316 axis auto
317 grid on;
318 set(gca,'fontsize',20)
319 zlim([0 15])
320 ylim([13 25])
321
322
323 figure(26); clf;
324 [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,15));
325 surface(xx,yy,sol);
326 view([75,15])
327 axis auto
328 grid on;
329 set(gca,'fontsize',20)
330 zlim([0 15])
331 ylim([13 25])
332
333
334 figure(27); clf;
335 [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,75));
336 surface(xx,yy,sol);
337 view([75,15])
338 axis auto
339 grid on;
340 set(gca,'fontsize',20)
341 zlim([0 15])
342 ylim([13 25])
343
344
345 figure(28); clf;
346 [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,500));
347 surface(xx,yy,sol);
348 view([75,15])
349 axis auto
350 grid on;
351 set(gca,'fontsize',20)
352 zlim([0 15])
353 ylim([13 25])
354
355 %END G SOLUTION
356 figure(11); clf;
357 [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,NUMBER_TIME_STEPS+1));
358 surface(xx,yy,sol);
359 view([40,30])
360 grid on;
361
362 %END G SOLUTION

```

```

363 figure(12); clf;
364 set(gca, 'FontSize', 12);
365 [C, h]=contour(xx, yy, sol);
366 clabel(C, h);
367
368 %G MOVIE
369 figure(20); clf;
370 peli = moviein(NUMBER_TIME_STEPS+1);
371 SOLUTION1=zeros(1, NUMBER_TIME_STEPS+1);
372 SOLUTION2=zeros(1, NUMBER_TIME_STEPS+1);
373 SOLUTION3=zeros(1, NUMBER_TIME_STEPS+1);
374 SOLUTION4=zeros(1, NUMBER_TIME_STEPS+1);
375 SOLUTION5=zeros(1, NUMBER_TIME_STEPS+1);
376 for n=1:NUMBER_TIME_STEPS+1
377     [xx, yy, sol] = MatSol(X, N_THETA, N_R, G_VECTOR(:, n));
378     SOLUTION1(n)=sol(1, 11);
379     SOLUTION2(n)=sol(10, 11);
380     SOLUTION3(n)=sol(20, 11);
381     SOLUTION4(n)=sol(30, 11);
382     SOLUTION5(n)=sol(40, 11);
383     surf(xx, yy, sol);
384     zlim([0 15])
385     ylim([13 25])
386     pause(0.0001)
387     peli(:, n) = getframe;
388 end
389 TIMESTEPS=1:NUMBER_TIME_STEPS+1
390 figure(50)
391 plot(TIMESTEPS, SOLUTION1, 'LineWidth', 2)
392 hold on
393 plot(TIMESTEPS, SOLUTION2, 'LineWidth', 2)
394 hold on
395 plot(TIMESTEPS, SOLUTION3, 'LineWidth', 2)
396 hold on
397 plot(TIMESTEPS, SOLUTION4, 'LineWidth', 2)
398 hold on
399 plot(TIMESTEPS, SOLUTION5, 'LineWidth', 2)
400 xlabel('Step Number')
401 ylabel('Monomer Density (uM)')
402 legend('Node 1', 'Node 10', 'Node 20', 'Node 30', 'Node 40')
403 ylim([0 15])
404 xlim([0 500])
405 set(gca, 'FontSize', 20)
406
407
408 %
409 %
410 %

```

```

411 %
412 %
413
414
415 elseif problem==2;
416
417
418 %
419
420 clear; close all; clc
421
422
423 %
424
425 %GENERATING THE MESH
426
427 N_R = 10;
428 N_THETA = 10;
429 MU=1000
430 [XP,TP] = createMesh2(N_R,N_THETA);
431 [X,T,THETA_VAR] = createMesh_velocity(N_R,N_THETA);
432 elem = 0;
433
434
435 figure(109)
436 plotMesh(TP,XP,elem);
437
438 figure(200)
439 plotMesh(T,X,elem,'k');
440
441
442 %
443
444
445 %BILINEAR QUADRILATERAL PRESSURE MESH WITH QUADRATIC VELOCITY
446 elemV = 0; degreeV = 2; degreeP = 1;
447 elemP = elemV;
448 referenceElement = SetReferenceElementStokes(elemV,degreeV,elemP,degreeP);
449
450 % DESCRETIZATION MATRICIES
451 [K,G,f] = StokesSystem(X,T,XP,TP,referenceElement);
452 K = MU*K;
453 [DOF_PRESSURE,DOF_VELOCITY] = size(G);
454
455
456 %
457
458 %CONVERT NUMBER OF ELEMENTS TO NUMBERS OF NODES

```

```

459 NN_R=2*N_R+1
460 NN_THETA=2*N_THETA+1
461
462 %FIND NODES FOR DIRICHLET
463 NODES_Y1=1:(NN_THETA);
464 NODES_Y2=(NN_R-1)*NN_THETA+1:(NN_THETA)*(NN_R);
465 NODES_Y1=NODES_Y1';
466 NODES_Y2=NODES_Y2';
467
468 %NODS TO BE IMPOSED ON
469 NODES_DIR_BC = [NODES_Y1; NODES_Y2];
470
471
472 %NUMBER OF DEGREES OF FREEDOM ON DIRICHLET NODES
473 DIR_DOF = 2*length(NODES_DIR_BC) ;
474 %CONFINED FLOW
475 confined = 1;
476
477 %MATRICIES TO IMPOSE DIRICHLET BOUNDARY CONDITIONS
478 C_MATRIX = [2*NODES_DIR_BC - 1; 2*NODES_DIR_BC];
479 C_STEP=reshape(C_MATRIX,DIR_DOF/2,2);
480 C_BAL =reshape(C_STEP',DIR_DOF,1);
481 A_DIR_MAT = zeros(DIR_DOF,DOF_VELOCITY);
482 A_DIR_MAT(:,C_BAL) = eye(DIR_DOF);
483
484 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
485
486 %IMPOSITION OF THE BOUNDARY CONDITIONS
487
488 BC_THETA1 = 0;
489 BC_THETA2 = 0;
490 BC_RADIAL1 = -0.15;
491 BC_RADIAL2 = -0.3;
492 VELOCITY_BC_X1 = BC_THETA1*sin(pi/2- THETA_VAR ) + BC_RADIAL1*cos(pi/2-
    THETA_VAR);
493 VELOCITY_BC_X2 = BC_THETA2*sin(pi/2-THETA_VAR) + BC_RADIAL2*cos(pi/2-
    THETA_VAR);
494 VELOCITY_BC_Y1 = -BC_THETA1*cos(pi/2-THETA_VAR) + BC_RADIAL1*sin(pi/2-
    THETA_VAR);
495 VELOCITY_BC_Y2 = -BC_THETA2*cos(pi/2-THETA_VAR) + BC_RADIAL2*sin(pi/2-
    THETA_VAR);
496 B_STEP = [VELOCITY_BC_X1' VELOCITY_BC_Y1';...
    VELOCITY_BC_X2' VELOCITY_BC_Y2'];
497
498 B_DIR_VEC =reshape(B_STEP',DIR_DOF,1);
499
500 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
501
502 %GENERATE ENTIRE SYSTEM OF EQUATIONS
503
504 if confined
505     nunkP = DOF_PRESSURE-1;
506     disp(' ')
507     disp('Confined flow. Pressure on lower left corner is set to zero');

```

```

508     G(1,:) = [];
509 else
510     nunkP = DOF_PRESSURE;
511 end
512 Atot = [K           A_DIR_MAT'           G'
513         A_DIR_MAT   zeros(DIR_DOF,DIR_DOF)   zeros(DIR_DOF,nunkP)
514         G           zeros(nunkP,DIR_DOF)     zeros(nunkP,nunkP)];
515 btot = [f ; B_DIR_VEC ; zeros(nunkP,1)];
516
517 sol = Atot\btot ;
518
519 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
520
521 % POSTPROCESS
522
523
524 velo = reshape(sol(1:DOF_VELOCITY), 2, [])';
525
526 if confined
527     pres = [0; sol(DOF_VELOCITY+DIR_DOF+1:DOF_VELOCITY+DIR_DOF+nunkP)];
528 else
529     pres = sol(DOF_VELOCITY+DIR_DOF+1:DOF_VELOCITY+DIR_DOF+nunkP);
530 end
531
532 nPt = size(X,1);
533 figure('Name','TIGHT');
534 quiver(X(1:nPt,1),X(1:nPt,2),velo(1:nPt,1),velo(1:nPt,2));
535 hold on
536 axis equal; axis tight
537
538
539 PlotResults(X,T,velo(:,1),referenceElement.elemV,referenceElement.degreeV)
540
541 PlotResults(X,T,velo(:,2),referenceElement.elemV,referenceElement.degreeV)
542
543 if degreeP == 0
544     PlotResults(X,T,pres,referenceElement.elemP,referenceElement.degreeP)
545 else
546     PlotResults(XP,TP,pres,referenceElement.elemP,referenceElement.degreeP
547     )
548 end
549
550 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
551 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
552 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
553 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
554 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
555 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
556
557 elseif problem==3;
558
559

```

```

560 clc
561 clear all
562 close all
563
564
565 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
566 %MATERIAL PARAMETERS
567 THETA=2/3; SIGMA_GF=0.5; SIGMA_G=2; SIGMA_F=0.25; D_G=15; D_F=5;
568           %s^-1           %s^-1           %s^-1   %um/s   %um/s
569
570 mu=1000
571
572 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
573
574 %GENERATION OF THE MESH
575
576 %NUMBER OF NODES IN RADIAL DIRECTION
577 N_R = 40
578
579 %NUMBER OF NODES IN THETA DIRECTION
580 N_THETA = 30
581
582 elemV = 0; degreeV = 1; degreeP = 1;
583 elemP = elemV;
584 referenceElement = SetReferenceElementStokes(elemV, degreeV, elemP, degreeP);
585 Xe_ref = referenceElement.Xe_ref;
586 [X,T,XP,TP,THETA_VAR] = CreateMeshes(N_THETA,N_R,referenceElement);
587 figure()
588 plot_Mesh(T,X,elemV,'b-')
589 figure()
590 plot_Mesh(TP,XP,elemV,'r-')
591
592
593 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
594
595 %APPLICATION OF THE BOUNDARY CONDITIONS
596
597 %OBTAIN MATRICIES
598 [T_F,T_U] = boundaryMatrices(X,T,elemV,degreeV,Xe_ref)
599 [K_STOKES,G_VECTOR,f] = StokesSystem2(X,T,XP,TP,referenceElement);
600 K_STOKES = mu*K_STOKES;
601 ndofV = size(K_STOKES,1);
602
603 %CONVERT NUMBER OF ELEMENTS TO NUMBERS OF NODES
604 NN_R=N_R+1
605 NN_THETA=N_THETA+1
606
607 %FIND NODES FOR DIRICHLET
608 NODES_Y1=1:(NN_THETA);
609 NODES_Y2=(NN_R-1)*NN_THETA+1:(NN_THETA)*(NN_R);
610 NODES_Y1=NODES_Y1';
611 NODES_Y2=NODES_Y2';
612

```

```

613
614 %NODES TO BE IMPOSED ON
615 NODES_DIR_BC = [NODES_Y1; NODES_Y2]
616
617
618 %NUMBER OF DEGREES OF FREEDOM ON DIRICHLET NODES
619 DIR_DOF = 2*length(NODES_DIR_BC) ;
620
621 %CONFINED FLOW
622 confined = 1;
623
624 %MATRICIES TO IMPOSE DIRICHLET BOUNDARY CONDITIONS
625 C_MATRIX = [2*NODES_DIR_BC - 1; 2*NODES_DIR_BC];
626 C_STEP=reshape(C_MATRIX,DIR_DOF/2,2);
627 C_BAL=reshape(C_STEP',DIR_DOF,1);
628 A_DirBC = zeros(DIR_DOF,ndofV);
629 A_DirBC(:,C_BAL) = eye(DIR_DOF);
630
631 BC_THETA1 = 0;
632 BC_THETA2 = 0;
633 BC_RADIAL1 = -0.15;
634 BC_RADIAL2 = -0.3;
635 VELOCITY_BC_X1 = BC_THETA1*sin(pi/2- THETA_VAR ) + BC_RADIAL1*cos(pi/2-
        THETA_VAR);
636 VELOCITY_BC_X2 = BC_THETA2*sin(pi/2-THETA_VAR) + BC_RADIAL2*cos(pi/2-
        THETA_VAR);
637 VELOCITY_BC_Y1 = -BC_THETA1*cos(pi/2-THETA_VAR) + BC_RADIAL1*sin(pi/2-
        THETA_VAR);
638 VELOCITY_BC_Y2 = -BC_THETA2*cos(pi/2-THETA_VAR) + BC_RADIAL2*sin(pi/2-
        THETA_VAR);
639
640 B_STEP = [VELOCITY_BC_X1' VELOCITY_BC_Y1';...
641          VELOCITY_BC_X2' VELOCITY_BC_Y2']
642
643 %RHS BC ENFORCEMENT VECTOR
644 B_DIR_VEC =reshape(B_STEP',DIR_DOF,1)
645 VELOCITY_DIR = [C_BAL B_DIR_VEC]
646
647
648 %MATRIX FOR BC ENFORCEMENT
649 A_DIR_MAT = K_STOKES + T_U;
650 A_DIR_MAT(VELOCITY_DIR(:,1), :) = 0;
651 A_DIR_MAT(:, VELOCITY_DIR(:,1)) = 0;
652 A_DIR_MAT(VELOCITY_DIR(:,1), VELOCITY_DIR(:,1)) = eye(DIR_DOF);
653
654
655
656 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
657
658
659 %GAUSS INTEGRATION
660
661 n_gaus=4;

```

```

662
663 [pospg ,wpg] = Quadrature(elemV , ntaus)
664
665 [N,Nxi ,Neta] = ShapeFunc(elemV , degreeV , pospg) ;
666
667 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
668
669 %GENERATION OF M AND K MATRICIES
670
671 M = CreMassMat(X,T, pospg ,wpg,N,Nxi , Neta) ;
672
673 K = CreStiffMat(X,T, pospg ,wpg,N,Nxi , Neta) ;
674
675
676 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
677
678 % DEFINITION OF TIME PARAMETERS
679 END_TIME = 10;
680 NUMBER_TIME_STEPS = 499;
681 DT = END_TIME/NUMBER_TIME_STEPS;
682
683 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
684
685 %ENFORCING THE BOUNDARY CONDITION FOR F
686 NUMBER_OF_NODES = size(X,1) ;
687 F_VECTOR = zeros(NUMBER_OF_NODES,NUMBER_TIME_STEPS+1) ;
688 DIRICHLET_VALUE = 80;
689 F_DIRICHLET = 0;
690 for i = 1:size(X,1)
691     poly = abs(X(i,1)^2 + X(i,2)^2 - 25^2) ;
692     if poly <= 10^-6
693         F_DIRICHLET = F_DIRICHLET + 1;
694         F_VECTOR(i,:) = DIRICHLET_VALUE;
695     end
696 end
697 nodesD = ((N_THETA+1)*(N_R+1)-N_THETA:(N_THETA+1)*(N_R+1))';
698 CDIR = [nodesD , zeros(length(nodesD) ,1) ] ;
699 F_DIRICHLET = size(CDIR,1) ;
700
701 ACCD_F = zeros(F_DIRICHLET,NUMBER_OF_NODES) ;
702 ACCD_F(:,CDIR(:,1)) = eye(F_DIRICHLET) ;
703 BCCD_F = CDIR(:,2) ;
704
705 STORE_V(:,1) = zeros(ndofV,1) ;
706 STORE_V(VELOCITY_DIR(:,1)) = VELOCITY_DIR(:,2) ;
707
708
709
710 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
711
712 %INITIALIZE G EQUATIONS
713
714 G_VECTOR = zeros(NUMBER_OF_NODES,NUMBER_TIME_STEPS+1) ;

```



```

715 A_G = M + THETA*D_G*K*DT + THETA*SIGMA_G*M*DT;
716 B_G = -D_G*K*DT - SIGMA_G*M*DT;
717 C_G = SIGMA_GF*M*DT;
718 f_G = zeros(NUMBER_OF_NODES,1);
719 Atot_G = A_G;
720 [L2,U2] = lu(Atot_G);
721
722
723 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
724
725
726 % TRANSIENT SOLUTION
727 tol = 10^-6;
728
729
730 for n= 1:NUMBER_TIME_STEPS
731
732
733
734
735
736 %INITIALIZE VELOCITY INCREMENT
737 VELOCITY_INCREMENT = 1; FInc = 1;
738 iter = 0;
739 g = F_VECTOR(:,n);
740 n=n
741
742
743
744 %RUN LOOP WHILE THE VELOCITY INCREMENT IS GREATER THAN TOLERANCE
745 while VELOCITY_INCREMENT>=tol
746     iter = iter + 1;
747
748     %UPDATE F USING NODAL DENSITY MATRIX
749     fu = -T_F*g(:,iter);
750
751
752     %RUN LOOP FOR SIZE OF DESCRETIZED KMATRIX
753     for i =1:size(K_STOKES,1)
754
755         %UPDATING BC VALUES FOR LOOP
756         F_UG(i,1) = fu(i,1)-K_STOKES(i,VELOCITY_DIR(:,1))*B_DIR_VEC;
757
758     end
759
760     %UPDATING F GLOBAL FOR NODAL COORDINATES
761     for i = 1:DIR_DOF
762         F_UG(VELOCITY_DIR(i,1)) = VELOCITY_DIR(i,2);
763     end
764
765     STORE_V(:,iter+1) = A_DIR_MAT\F_UG;
766     VEL_VEC = STORE_V(:,iter+1);
767

```

```

768
769 %CREATE THIS ITERATIONS CONVECTION MATRIX
770 VEL_CONV = reshape(VEL_VEC, 2, [])';
771 C_MATRIX = CreConvMat(X, T, VEL_CONV, pospg, wpg, N, Nxi, Neta);
772
773
774 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
775
776 %%CALCULATE SOLUTION FOR F
777
778 %A MATRIX RESULTING FROM DESCRETIZATION
779 A_F = M + THETA*C_MATRIX*DT + THETA*D_F*K*DT + THETA*SIGMA_F*M*DT;
780
781 %B MATRIX RESULTING FROM DESCRETIZATION
782 B_F = M + (1-THETA)*(-C_MATRIX*DT - D_F*K*DT - SIGMA_F*M*DT);
783
784 %F MATRIX RESULTING FROM F DESCRETIZATION
785 F_F = zeros(NUMBER_OF_NODES, 1);
786
787
788
789 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
790
791
792
793 % CREATE GLOBAL MATRIX WITH LAGRANGE FOR INNER LOOP
794 A_INNERLOOP_F = [A_F ACCD_F'; ACCD_F zeros(F_DIRICHLET, F_DIRICHLET)
795 ];
796 [L1, U1] = lu(A_INNERLOOP_F);
797
798 B_INNERLOOP_F = [B_F*F_VECTOR(:, n) + F_F; BCCD_F];
799
800 LAGRANGE_1 = U1 \ (L1 \ B_INNERLOOP_F);
801 g(:, iter+1) = LAGRANGE_1(1:NUMBER_OF_NODES);
802
803 VELOCITY_INCREMENT = norm(STORE_V(:, iter+1) - STORE_V(:, iter));
804
805
806 end
807
808
809
810 %OVERWRITE C MATRIX USING VELOCITY VECTOR
811 V_STORE_2(:, n) = STORE_V(:, iter+1);
812 VEL_VEC = V_STORE_2(:, n);
813 VEL_CONV = reshape(VEL_VEC, 2, [])';
814 C_MATRIX = CreConvMat(X, T, VEL_CONV, pospg, wpg, N, Nxi, Neta);
815
816
817
818 %RECOMPUTATION OF MATRICIES
819 A_F = M + THETA*C_MATRIX*DT + THETA*D_F*K*DT + THETA*SIGMA_F*M*DT;

```

```

820 B_F = (-C_MATRIX*DT - D_F*K*DT - SIGMA_F*M*DT);
821 F_F = zeros(NUMBER_OF_NODES,1);
822
823
824
825
826
827 % CALCULATION OF SOLUTION AT TIME STEP
828 A_INNERLOOP_F = [A_F ACCD_F';ACCD_F zeros(F_DIRICHLET,F_DIRICHLET)];
829 [L1,U1] = lu(A_INNERLOOP_F);
830
831
832 B_INNERLOOP_F = [B_F*F_VECTOR(:,n)+ F_F; BCCD_F];
833 FA = U1\(L1\B_INNERLOOP_F);
834
835
836 FB = U1\(L1\B_INNERLOOP_F);
837
838 F_VECTOR(:,n+1) = F_VECTOR(:,n) + FB(1:NUMBER_OF_NODES);
839
840
841 %FINAL SOLUTION FOR G AT TIME STEP
842 btot = [B_G*G_VECTOR(:,n) - THETA*C_G*FA(1:NUMBER_OF_NODES) +
843         THETA*C_G*F_VECTOR(:,n+1) + f_G];
844 aux_G = U2\(L2\btot);
845 G_VECTOR(:,n+1) = G_VECTOR(:,n) + aux_G(1:NUMBER_OF_NODES);
846 end
847
848
849 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
850
851
852 figure()
853 nPt = size(X,1);
854 figure;
855 quiver(X(1:nPt,1),X(1:nPt,2),VEL_CONV(1:nPt,1),VEL_CONV(1:nPt,2));
856 hold on
857 axis equal; axis tight
858
859
860
861 figure()
862 PlotResults(X,T,VEL_CONV(:,1),referenceElement.elemV,referenceElement.
863             degreeV)
864 figure()
865 PlotResults(X,T,VEL_CONV(:,2),referenceElement.elemV,referenceElement.
866             degreeV)
867 v = sqrt((VEL_CONV(:,1).^2)+(VEL_CONV(:,2).^2));
868 figure()
869 PlotResults(X,T,v,referenceElement.elemV,referenceElement.degreeV)
870
871
872

```

```

870
871
872
873 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
874
875 %POSTPROCESSING
876
877 if max(F_VECTOR(:,NUMBER_TIME_STEPS+1))<100 && min(F_VECTOR(:,
    NUMBER_TIME_STEPS+1))>-100
878
879     figure (); clf;
880     [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,NUMBER_TIME_STEPS+1));
881     surface(xx,yy,sol);
882     view([40,30])
883     axis auto
884     grid on;
885
886
887
888
889     figure (); clf;
890     [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,1));
891     surface(xx,yy,sol);
892     view([75,15])
893     axis auto
894     grid on;
895     set(gca,'fontsize',20)
896     zlim([0 90])
897     ylim([13 25])
898
899
900     figure (); clf;
901     [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,2));
902     surface(xx,yy,sol);
903     view([75,15])
904     axis auto
905     grid on;
906     set(gca,'fontsize',20)
907     zlim([0 90])
908     ylim([13 25])
909
910
911     figure (); clf;
912     [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,5));
913     surface(xx,yy,sol);
914     view([75,15])
915     axis auto
916     grid on;
917     set(gca,'fontsize',20)
918     zlim([0 90])
919     ylim([13 25])
920
921

```

```

922 figure (); clf;
923 [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,15));
924 surface(xx,yy,sol);
925 view([75,15])
926 axis auto
927 grid on;
928 set(gca,'fontsize',20)
929 zlim([0 90])
930 ylim([13 25])
931
932
933 figure (); clf;
934 [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,75));
935 surface(xx,yy,sol);
936 view([75,15])
937 axis auto
938 grid on;
939 set(gca,'fontsize',20)
940 zlim([0 90])
941 ylim([13 25])
942
943
944 figure (); clf;
945 [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,500));
946 surface(xx,yy,sol);
947 view([75,15])
948 axis auto
949 grid on;
950 set(gca,'fontsize',20)
951 zlim([0 90])
952 ylim([13 25])
953
954
955
956
957 figure (); clf;
958 set(gca,'FontSize',12);
959 [C,h]=contour(xx,yy,sol);
960 clabel(C,h);
961 axis auto
962
963
964 figure(10); clf;
965 peli = moviein(NUMBER_TIME_STEPS+1);
966 axis auto
967 SOLUTION1=zeros(1,NUMBER_TIME_STEPS+1);
968 SOLUTION2=zeros(1,NUMBER_TIME_STEPS+1);
969 SOLUTION3=zeros(1,NUMBER_TIME_STEPS+1);
970 SOLUTION4=zeros(1,NUMBER_TIME_STEPS+1);
971 SOLUTION5=zeros(1,NUMBER_TIME_STEPS+1);
972
973 for n=1:NUMBER_TIME_STEPS+1
974     [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,n));

```

```

975     SOLUTION1(n)=sol(1,11);
976     SOLUTION2(n)=sol(10,11);
977     SOLUTION3(n)=sol(20,11);
978     SOLUTION4(n)=sol(30,11);
979     SOLUTION5(n)=sol(40,11);
980     surf(xx,yy,sol);
981     zlim([0 90])
982     ylim([13 25])
983     pause(0.01)
984     peli(:,n) = getframe;
985 end
986 end
987 TIMESTEPS=1:NUMBER_TIME_STEPS+1
988 plot(TIMESTEPS,SOLUTION1, 'LineWidth',2)
989 hold on
990 plot(TIMESTEPS,SOLUTION2, 'LineWidth',2)
991 hold on
992 plot(TIMESTEPS,SOLUTION3, 'LineWidth',2)
993 hold on
994 plot(TIMESTEPS,SOLUTION4, 'LineWidth',2)
995 hold on
996 plot(TIMESTEPS,SOLUTION5, 'LineWidth',2)
997 xlabel('Step Number')
998 ylabel('Actin Filament Density (uM)')
999 legend('Node 1','Node 10','Node 20','Node 30','Node 40')
1000 set(gca,'FontSize',20)
1001 xlim([0 500])
1002 pause(0.1);
1003
1004
1005 figure(); clf;
1006 [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,1));
1007 surface(xx,yy,sol);
1008 view([75,15])
1009 axis auto
1010 grid on;
1011 set(gca,'fontsize',20)
1012 zlim([0 15])
1013 ylim([13 25])
1014
1015
1016 figure(); clf;
1017 [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,5));
1018 surface(xx,yy,sol);
1019 view([75,15])
1020 axis auto
1021 grid on;
1022 set(gca,'fontsize',20)
1023 zlim([0 15])
1024 ylim([13 25])
1025
1026
1027 figure(); clf;

```

```

1028 [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,10));
1029 surface(xx,yy,sol);
1030 view([75,15])
1031 axis auto
1032 grid on;
1033 set(gca,'fontsize',20)
1034 zlim([0 15])
1035 ylim([13 25])
1036
1037
1038 figure(); clf;
1039 [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,15));
1040 surface(xx,yy,sol);
1041 view([75,15])
1042 axis auto
1043 grid on;
1044 set(gca,'fontsize',20)
1045 zlim([0 15])
1046 ylim([13 25])
1047
1048
1049 figure(); clf;
1050 [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,75));
1051 surface(xx,yy,sol);
1052 view([75,15])
1053 axis auto
1054 grid on;
1055 set(gca,'fontsize',20)
1056 zlim([0 15])
1057 ylim([13 25])
1058
1059
1060 figure(); clf;
1061 [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,500));
1062 surface(xx,yy,sol);
1063 view([75,15])
1064 axis auto
1065 grid on;
1066 set(gca,'fontsize',20)
1067 zlim([0 15])
1068 ylim([13 25])
1069
1070 %END G SOLUTION
1071 figure(); clf;
1072 [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,NUMBER_TIME_STEPS+1));
1073 surface(xx,yy,sol);
1074 view([40,30])
1075 grid on;
1076
1077 %END G SOLUTION
1078 figure(); clf;
1079 set(gca,'FontSize',12);
1080 [C,h]=contour(xx,yy,sol);

```

```

1081 clabel(C,h);
1082
1083 %G MOVIE
1084 figure(); clf;
1085 peli = moviein(NUMBER_TIME_STEPS+1);
1086     SOLUTION1=zeros(1,NUMBER_TIME_STEPS+1);
1087     SOLUTION2=zeros(1,NUMBER_TIME_STEPS+1);
1088     SOLUTION3=zeros(1,NUMBER_TIME_STEPS+1);
1089     SOLUTION4=zeros(1,NUMBER_TIME_STEPS+1);
1090     SOLUTION5=zeros(1,NUMBER_TIME_STEPS+1);
1091 for n=1:NUMBER_TIME_STEPS+1
1092     [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,n));
1093     SOLUTION1(n)=sol(1,11);
1094     SOLUTION2(n)=sol(10,11);
1095     SOLUTION3(n)=sol(20,11);
1096     SOLUTION4(n)=sol(30,11);
1097     SOLUTION5(n)=sol(40,11);
1098     surf(xx,yy,sol);
1099     zlim([0 15])
1100     ylim([13 25])
1101     pause(0.0001)
1102     peli(:,n) = getframe;
1103 end
1104 TIMESTEPS=1:NUMBER_TIME_STEPS+1
1105 figure(50)
1106 plot(TIMESTEPS,SOLUTION1,'LineWidth',2)
1107 hold on
1108 plot(TIMESTEPS,SOLUTION2,'LineWidth',2)
1109 hold on
1110 plot(TIMESTEPS,SOLUTION3,'LineWidth',2)
1111 hold on
1112 plot(TIMESTEPS,SOLUTION4,'LineWidth',2)
1113 hold on
1114 plot(TIMESTEPS,SOLUTION5,'LineWidth',2)
1115 xlabel('Step Number')
1116 ylabel('Monomer Density (uM)')
1117 legend('Node 1','Node 10','Node 20','Node 30','Node 40')
1118 ylim([0 15])
1119 xlim([0 500])
1120 set(gca,'FontSize',20)
1121 %
1122 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1123 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1124 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1125 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1126 else
1127     disp('not valid selection, closing')
1128     exit
1129 end
1130 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
1131 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```



1132 %%%%%%%%%%  
1133 %%%%%%%%%%  
1134 %%%%%%%%%%  
1135 %%%%%%%%%%