Universitat Polytechnica de Catalunya
MSc Computational Mechanics
Spring 2018

Finite Element in Fluids

Assignment

Due 01/06/2018

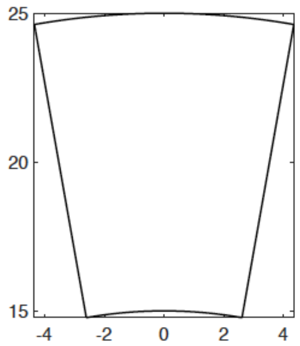
Alexander Keiser

Contents

1 Transport Problem

We will first solve a transport problem to obtain actin filament and monomer densities (F and G) respectivley. 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) = 80μM. 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) μm/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.

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



We will first descretize 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

$$\frac{\Delta F}{\Delta t} - \Theta\left[-U \cdot \nabla(\Delta F) + D_F \nabla^2(\Delta F) - \sigma_F(\Delta F)\right] = \ldots$$

$$= -U \cdot \nabla F^n + D_F \nabla^2 F^n - \sigma_F F^n$$

continuing on the next page...

Now expanding and applying Galerkins in space gives us

$$\left(w, \frac{\Delta F}{\Delta t}\right) - \theta\left[\left(w, -U\cdot\nabla(\Delta F)\right) + \left(w, D_F\nabla^2(\Delta F)\right) - \left(w, \sigma_F(\Delta F)\right)\right] = \ldots$$

$$= \left(w, -U\cdot\nabla F^n\right) + \left(w, D_F\nabla^2 F^n\right) - \left(w, \sigma_F F^n\right)$$

And integrating by parts the necessary terms

$$-\theta D_F\left(w, \nabla^2(\Delta F)\right) = -\theta D_F\left[\cancel{\left(w, \nabla(\Delta F)\right)} - \left(\nabla w, \nabla(\Delta F)\right)\right]$$

$$D_F\left(w, \nabla^2 F^n\right) = D_F\left[\cancel{\left(w, \nabla F^n\right)} - \left(\nabla w, \nabla F^n\right)\right]$$

And finally rearranging

$$\left(w, \frac{\Delta F}{\Delta t}\right) + \theta\left(w, U\cdot\nabla(\Delta F)\right) + \theta D_F\left(\nabla w, \nabla(\Delta F)\right) + \theta\sigma_F\left(w, (\Delta F)\right) = \ldots$$

$$= -\left(w, U\cdot\nabla F^n\right) - D_F\left(\nabla w, \nabla F^n\right) - \sigma_F\left(w, F^n\right)$$

---

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

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

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

$$= \left(w, D_G\nabla^2 G^n\right) - \left(w, \sigma_G G^n\right) + \left(w, \hat{\sigma}_{GF} F^n\right)$$

integrating by parts & rearranging gives us

$$\left(w, \frac{\Delta G}{\Delta t}\right) + \theta D_G\left(\nabla w, \nabla(\Delta G)\right) + \theta\sigma_G\left(w, (\Delta G)\right) - \theta\sigma_{GF}\left(w, (\Delta F)\right) = \ldots$$

$$= -D_G\left(\nabla w, \nabla G^n\right) - \sigma_G\left(w, G^n\right) + \hat{\sigma}_{GF}\left(w, F^n\right)$$

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 matricies descretized on the previous page. Additionally, the implemented material properties can also be found in figure 2 below.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%IMPLEMENTATION OF GALERKIN TIME AND SPACE DESCRETIZATION


A1 = M  +  THETA*C*DT  +  THETA*D_F*K*DT  +  THETA*SIGMA_F*M*DT;

B1 = -C*DT  -  D_F*K*DT  -  SIGMA_F*M*DT;

A2 = M  +  THETA*D_G*K*DT  +  THETA*SIGMA_G*M*DT;

B2 = -D_G*K*DT  -  SIGMA_G*M*DT;

C2 = SIGMA_GF*M*DT;

F_1 = ZERO_ARRAY;

F_2 = ZERO_ARRAY;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

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

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

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

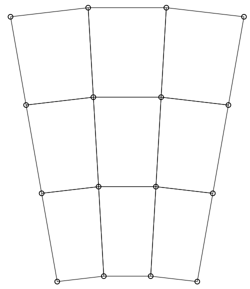


Figure 3: Sample Mesh of 3 by 3 Bilinear Quadrilateral Elements

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.

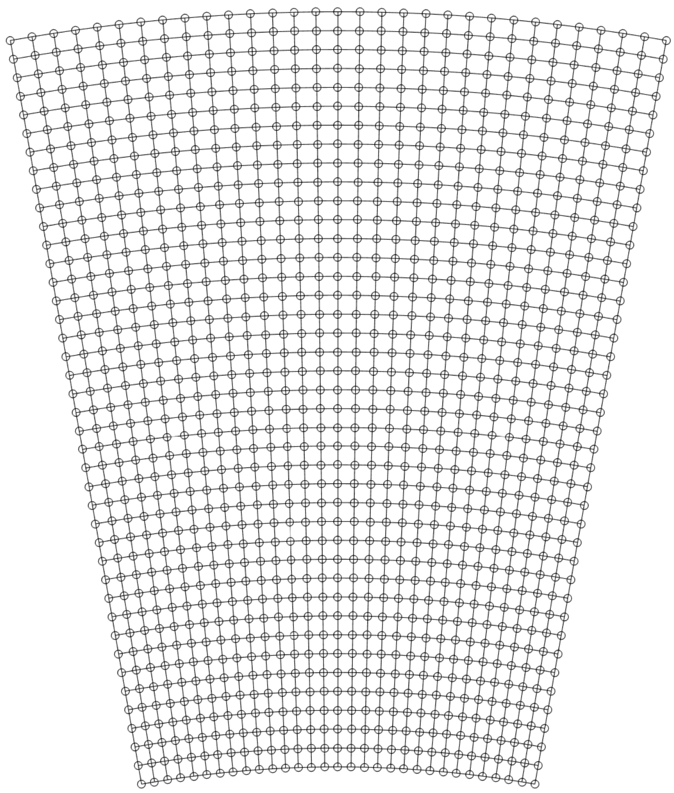


Figure 4: Actual Mesh of 40 Radial and 30 Theta Bilinear Quadrilateral Elements

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 Δt=0.2.

Figure 5: Preliminary transient solution at various time steps Δt=0.2

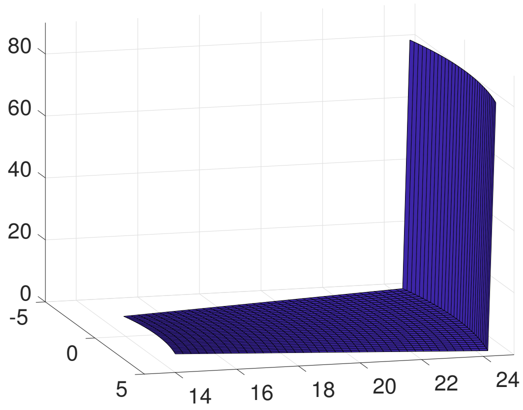


Figure 5a: Time Step 1

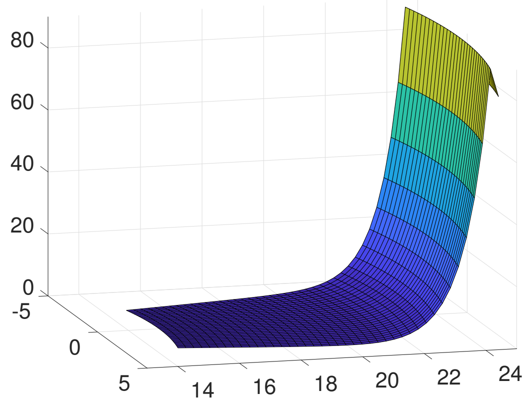


Figure 5b: Time Step 2

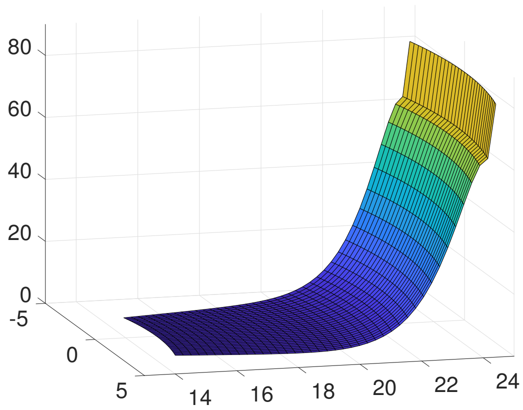


Figure 5c: Time Step 3

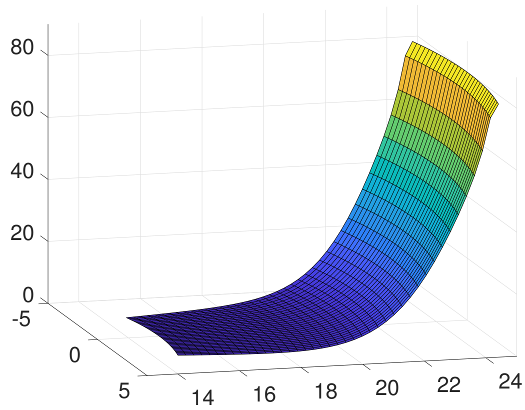


Figure 5d: Time Step 4

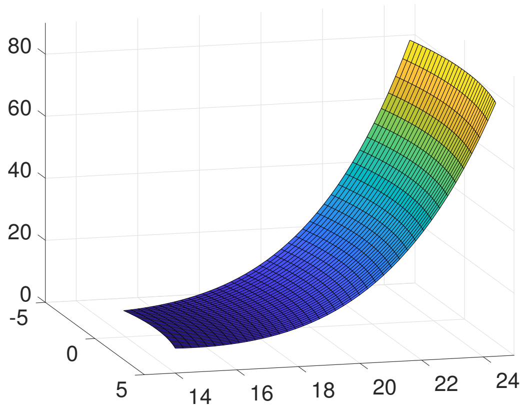


Figure 5e: Time Step 10

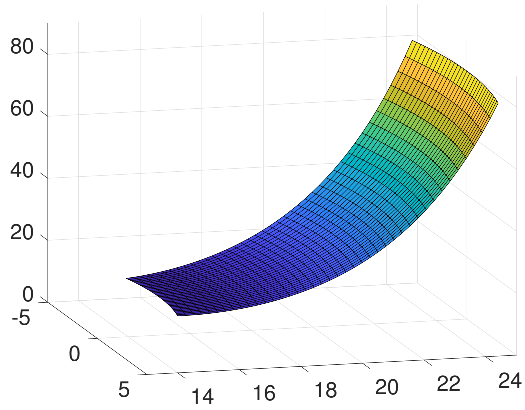


Figure 5f: Time Step 50

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 80μM 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 direchlet boundary condition of 80μM 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 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 Δt=0.02

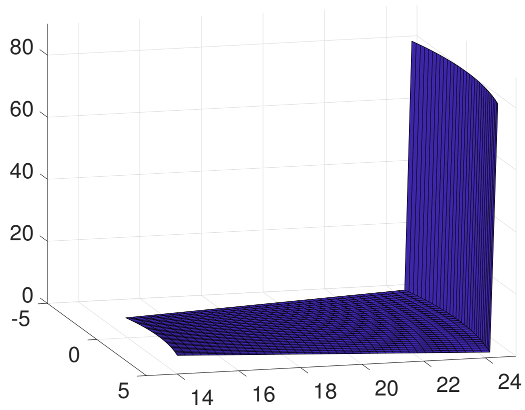


Figure 6a: Time Step 1

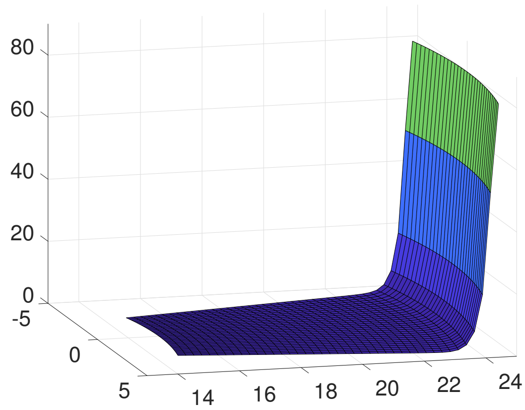


Figure 6b: Time Step 2

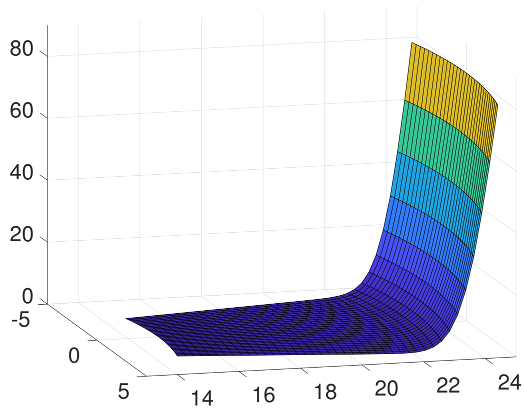


Figure 6c: Time Step 5

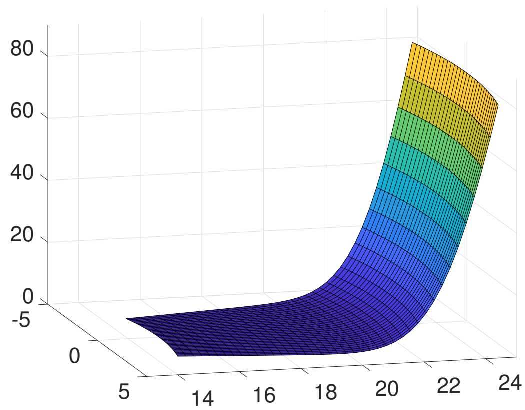


Figure 6d: Time Step 15

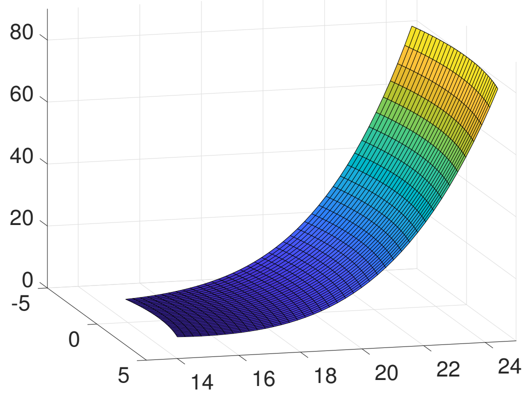


Figure 6e: Time Step 75

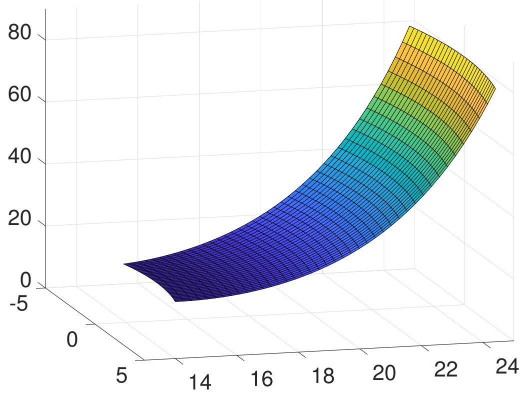


Figure 6f: Time Step 500

The above results were computed with 500 time steps over a time domain of 10 seconds producing the value for time step Δt=0.02. It is immediately apparent that the troublesome oscillations present with the higher value of Δt=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 throuought the time domain. These radial values will be taken along the y summetry line. The first node taken will be node 5, this node is close to the free radial end of the domain with no dirichlet coundary 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 80uM and will capture the behavior in that region. The results of this nodal evolution study can be found below in figure 7.

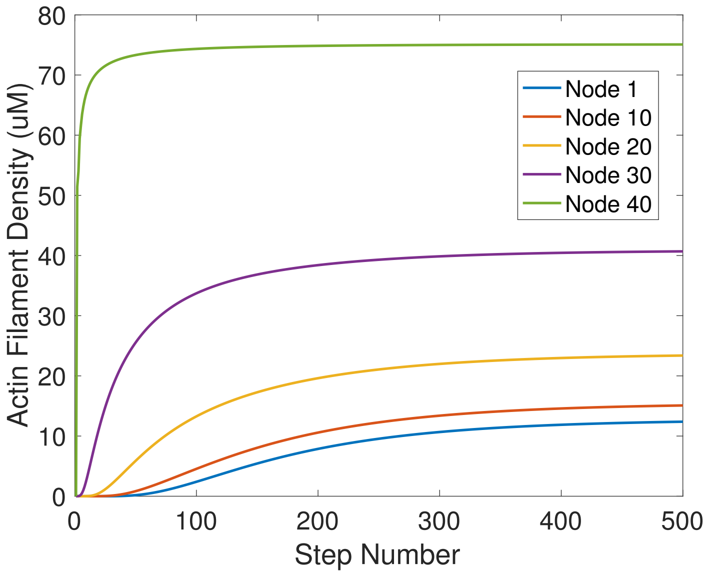


Figure 7: Actin Filament Density F Nodal Evolution Study

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 Δt=0.02

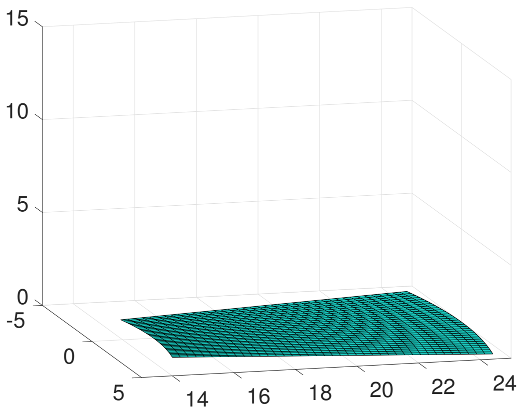


Figure 8a: Time Step 1

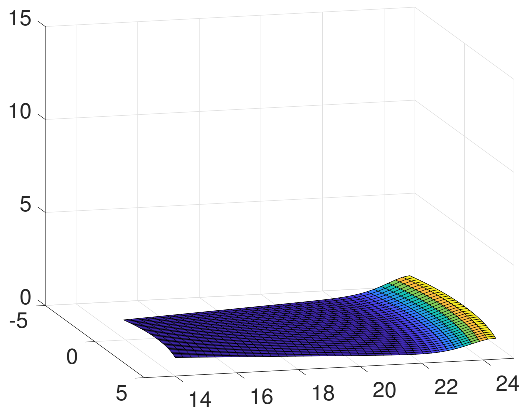


Figure 8b: Time Step 5

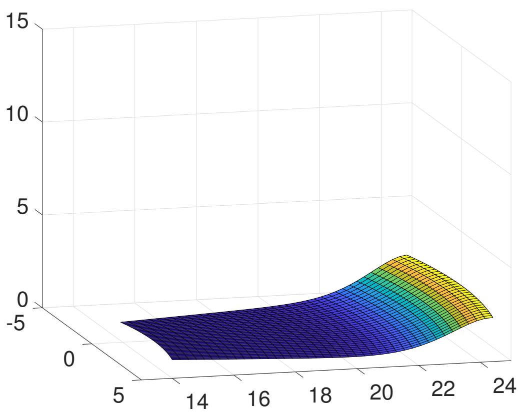


Figure 8c: Time Step 10

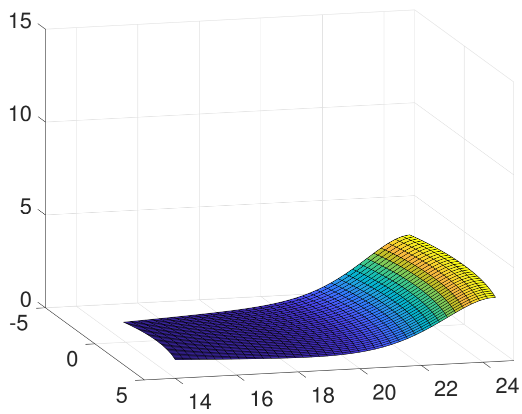


Figure 8d: Time Step 15

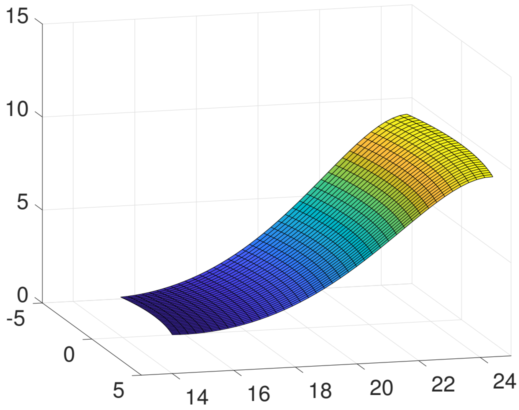


Figure 8e: Time Step 75

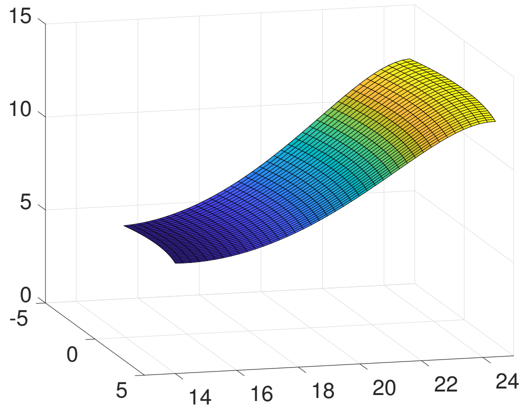


Figure 8f: Time Step 500

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.

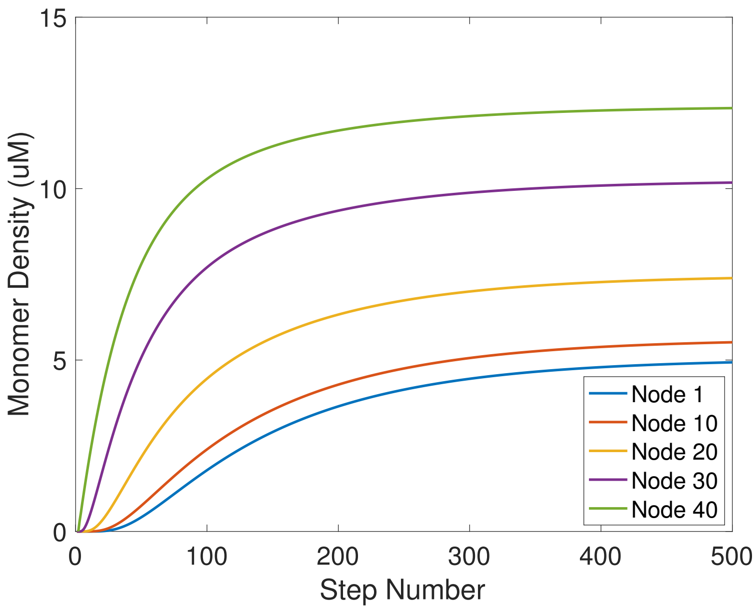


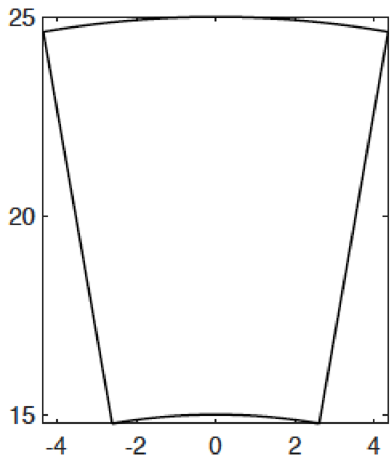
Figure 9: Monomer Density G Nodal Evolution Study

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 ν=1000 pN·s/μm

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

$$u_r(r = 15) = -0.15,\ u_\theta(r = 15) = 0$$
$$u_r(r = 25) = -0.30,\ u_\theta(r = 25) = 0$$



2.1 Discretization of The Stokes Equations

We will now descretize the stokes equations using a standard Galerkin space descertization scheme.

The Galerkin Space Descretization 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) $\rho(v_t + (v \cdot \nabla)v) = \nabla \cdot \sigma + \rho b$

and expanding

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

substituting 2 into 1 & deviding by ρ

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

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

and concelling out the relevent terms leaves us with

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

applying Galerkin to both

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

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 seperate 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.

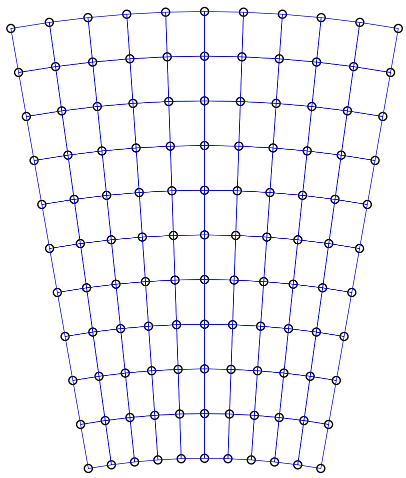


Figure 7: Generated Bi-Linear Quadrilateral Pressure Mesh

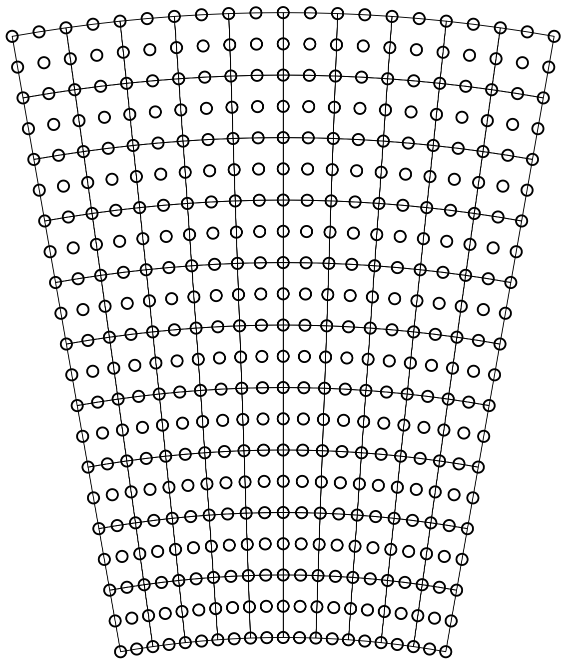


Figure 8: Generated Q_2Q_1 Quadratic Velocity Mesh

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.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%GENERATE NODAL CONNEVTIVITIES

ind_ele = 0 ;
for jj = 1:(Nr/2)
    for ii = 1:(Ntheta/2)
        ind_ele = ind_ele +1 ;
        ele_data(ind_ele,:) = [ind_ele , Node_number(ii+2,jj)+(ii-1)+(jj* VSPACE- VSPACE),...
                                         Node_number(ii+2,jj+2)+(ii-1)+(jj* VSPACE- VSPACE),...
                                         Node_number(ii,jj+2)+(ii-1)+(jj* VSPACE- VSPACE),...
                                         Node_number(ii,jj)+(ii-1)+(jj* VSPACE- VSPACE),...
                                         Node_number(ii+2,jj+1)+(ii-1)+(jj* VSPACE- VSPACE),...
                                         Node_number(ii+1,jj+2)+(ii-1)+(jj* VSPACE- VSPACE),...
                                         Node_number(ii,jj+1)+(ii-1)+(jj* VSPACE- VSPACE),...
                                         Node_number(ii+1,jj)+(ii-1)+(jj* VSPACE- VSPACE),...
                                         Node_number(ii+1,jj+1)+(ii-1)+(jj* VSPACE- VSPACE)] ;

    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

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.

$$u_r(r = 15) = -0.15, \; u_\theta(r = 15) = 0$$
$$u_r(r = 25) = -0.30, \; u_\theta(r = 25) = 0$$

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%IMPOSITION OF THE BOUNDARY CONDITIONS

BC_THETA1 = 0;
BC_THETA2 = 0;
BC_RADIAL1 = -0.15;
BC_RADIAL2 = -0.3;
VELOCITY_BC_X1 = BC_THETA1*sin(pi/2- THETA_VAR ) + BC_RADIAL1*cos(pi/2-THETA_VAR);
VELOCITY_BC_X2 = BC_THETA2*sin(pi/2-THETA_VAR) + BC_RADIAL2*cos(pi/2-THETA_VAR);
VELOCITY_BC_Y1 = -BC_THETA1*cos(pi/2-THETA_VAR) + BC_RADIAL1*sin(pi/2-THETA_VAR);
VELOCITY_BC_Y2 = -BC_THETA2*cos(pi/2-THETA_VAR) + BC_RADIAL2*sin(pi/2-THETA_VAR);

B_STEP = [VELOCITY_BC_X1' VELOCITY_BC_Y1';...
    VELOCITY_BC_X2'   VELOCITY_BC_Y2'];
b_DirBC =reshape(B_STEP',nDir,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

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.

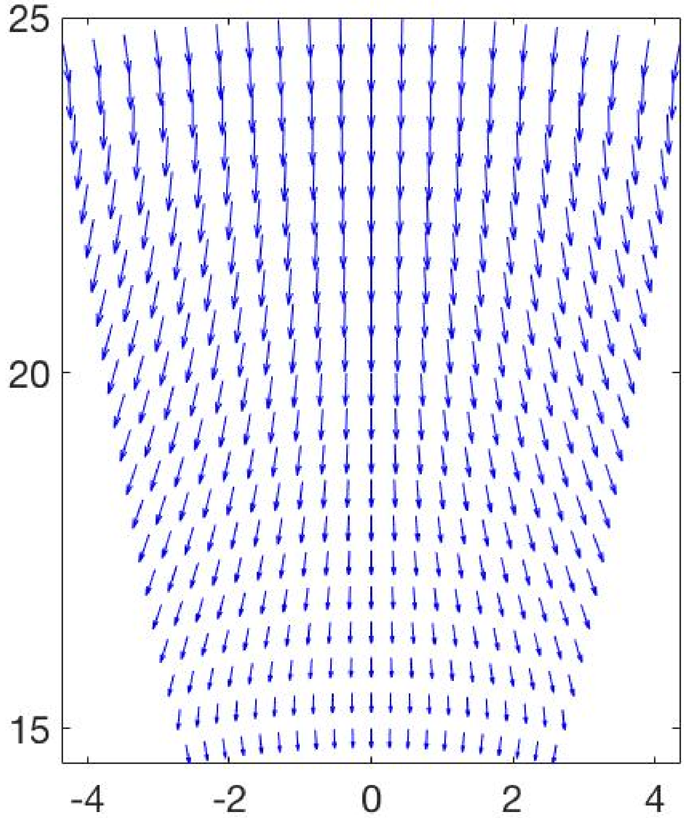


Figure 11: Solution Vectorized Velocity Field

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.

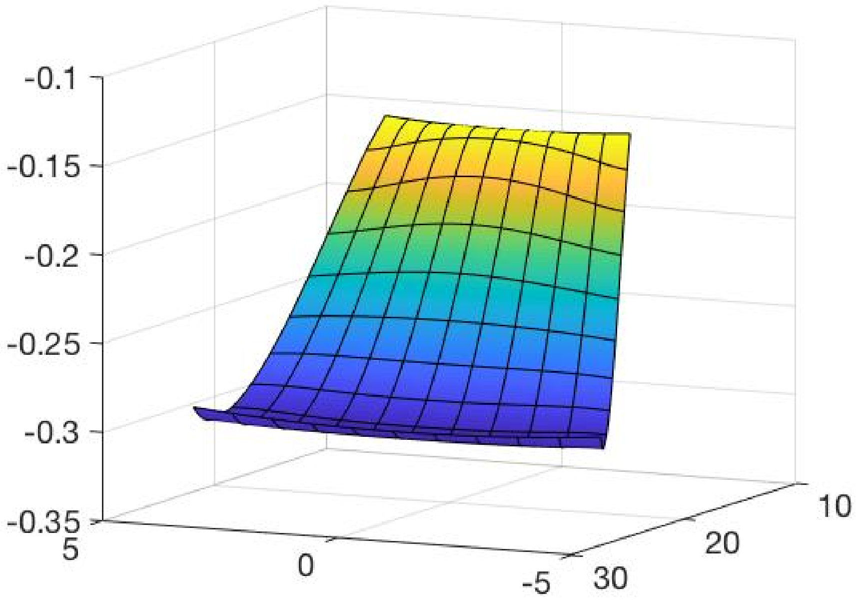


Figure 12: Y-Direction Velocity Profile

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..

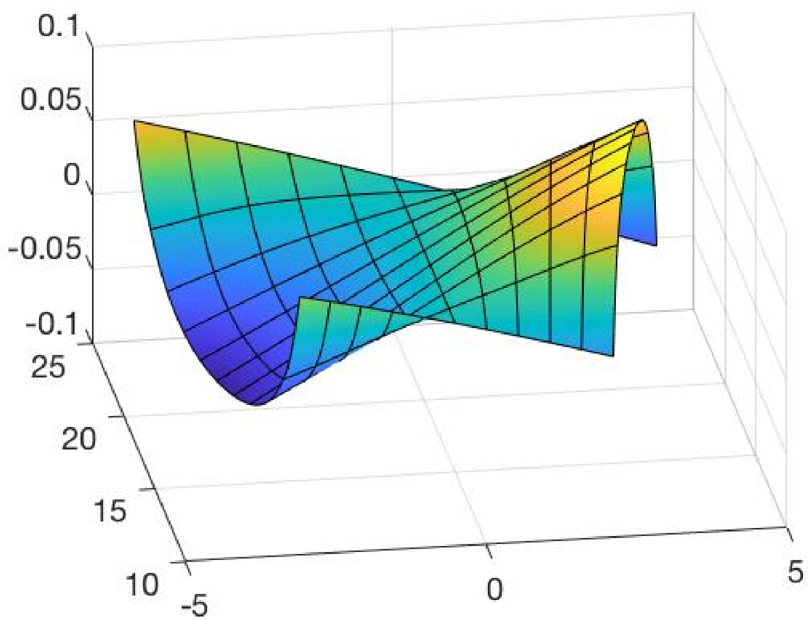


Figure 13: X-Direction Velocity Profile

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.

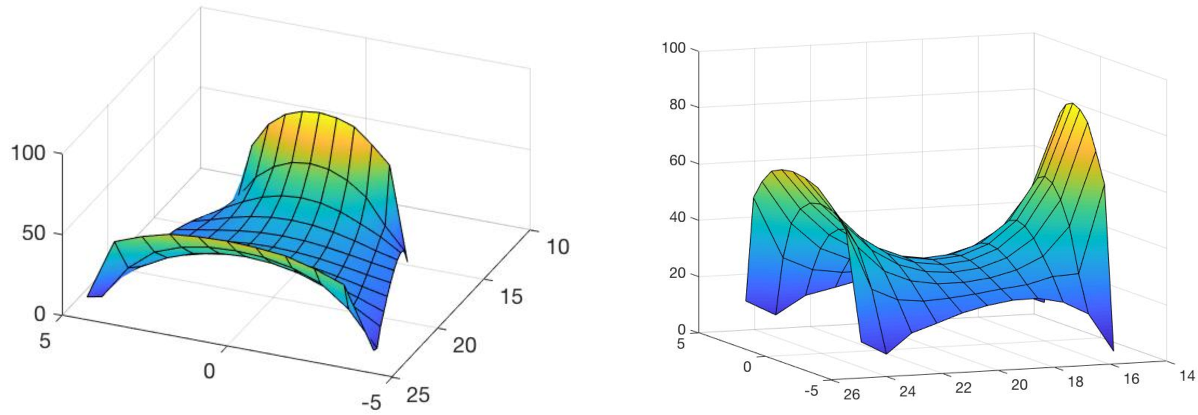


Figure 14: 2 Views of the Resultant Pressure Distribution

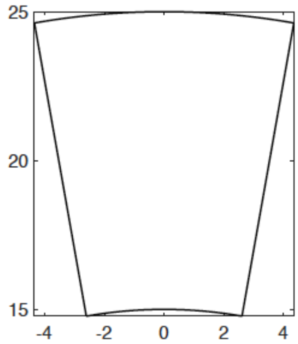
Above we have results for the pressure distribution from the implemented stokes system. The first thing worth noting is the presence of the confined flow causing the pressures in the four corners of the domain to approach zero. Next, we can begin to notice the pressure distributions at and close to the 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 $\mathbf{T_m}$ are surface forces on the leading edge.

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

$$u_r(r = 15) = -0.15,\ u_\theta(r = 15) = 0$$
$$u_r(r = 25) = -0.30,\ u_\theta(r = 25) = 0$$



As was the case in the transport problem, the filament density is constant at the upper boundary: F(r = 25) = 80μM. 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) μm/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^{st} eqn.

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

applying galerkins formulation

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

and integrating by parts the first term

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

and rearranging gives us...

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

and substituting to get the system

$$\boxed{Ku + T_f F + T_u u = 0}$$

where T_F & T_u are matricies 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.

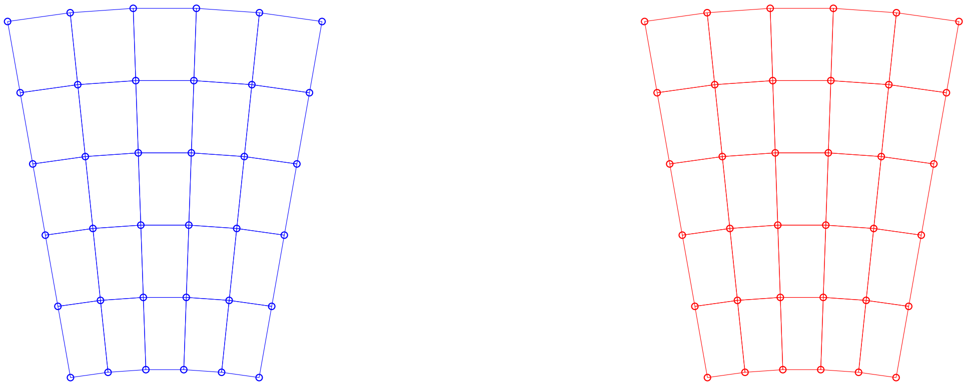


Figure 15: Pressure (left) and Velocity (right) Sample Meshes

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 Δt=0.02

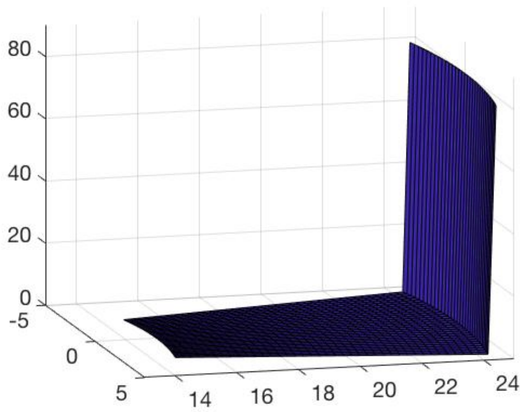


Figure 16a: Time Step 1

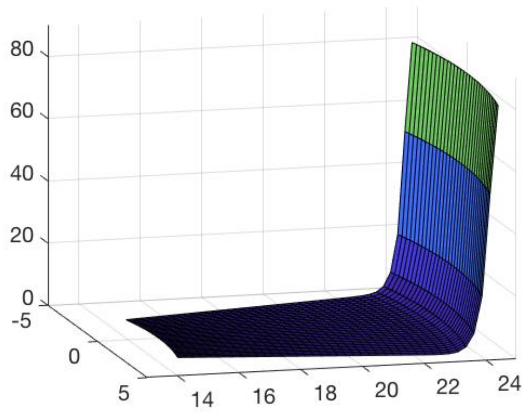


Figure 16b: Time Step 2

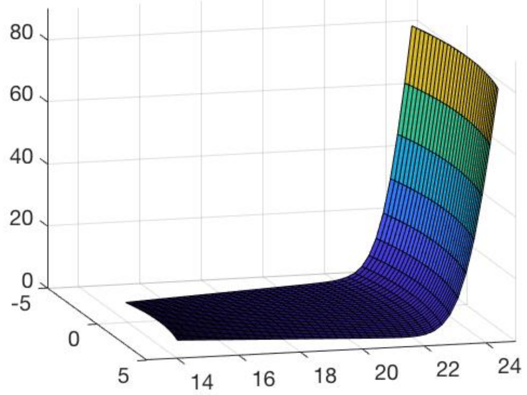


Figure 16c: Time Step 5

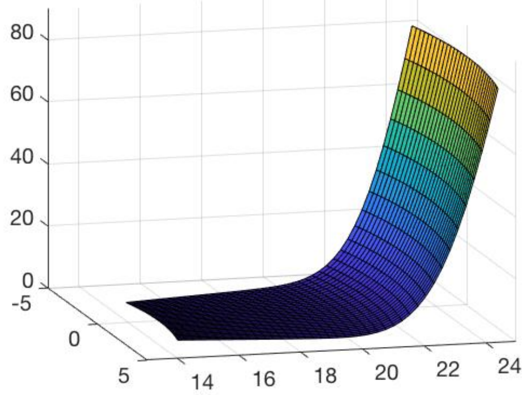


Figure 16d: Time Step 15

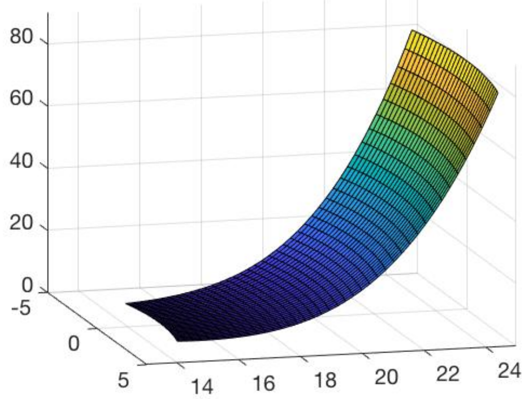


Figure 16e: Time Step 75

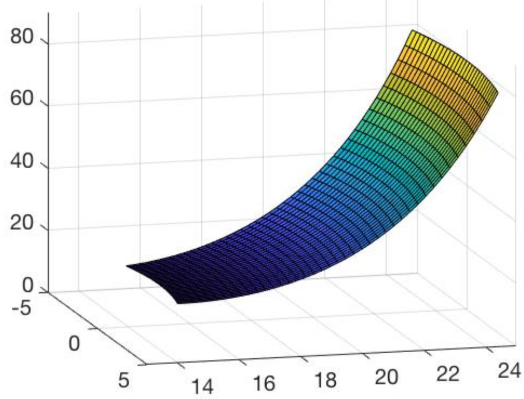


Figure 16f: Time Step 500

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 μM 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 μM 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.

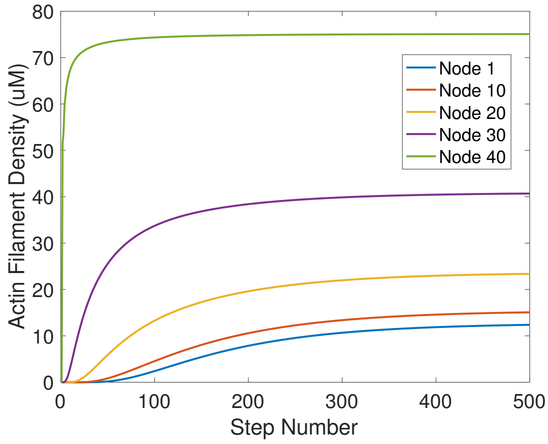


Figure 17a: Transport Problem F Nodal Evolution

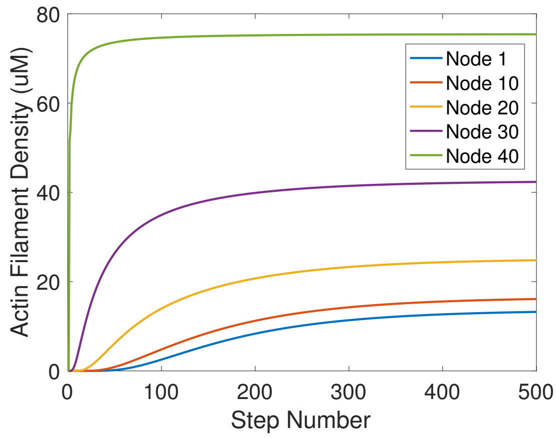


Figure 17b: Coupled Problem F Nodal Evolution

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 Δt=0.02

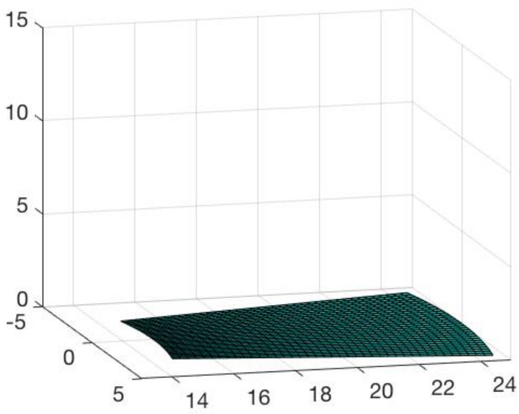


Figure 18a: Time Step 1

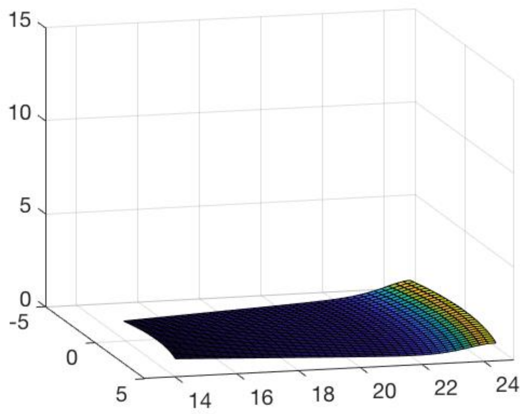


Figure 18b: Time Step 2

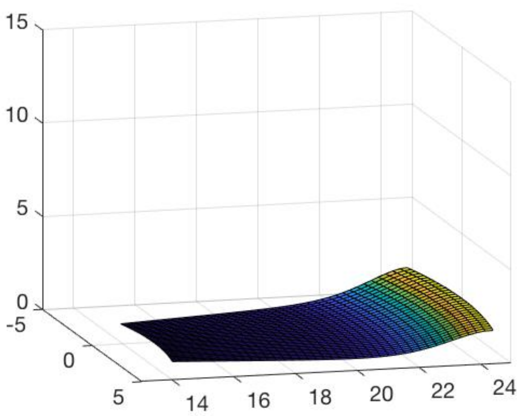


Figure 18c: Time Step 5

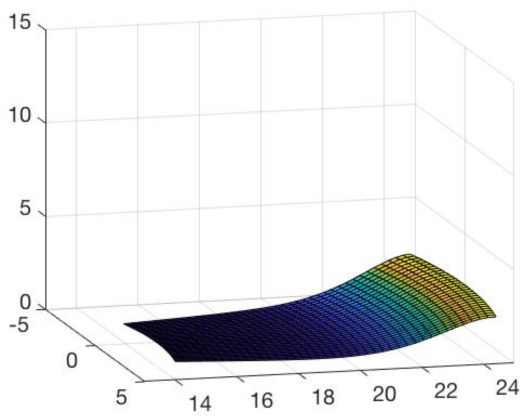


Figure 18d: Time Step 15

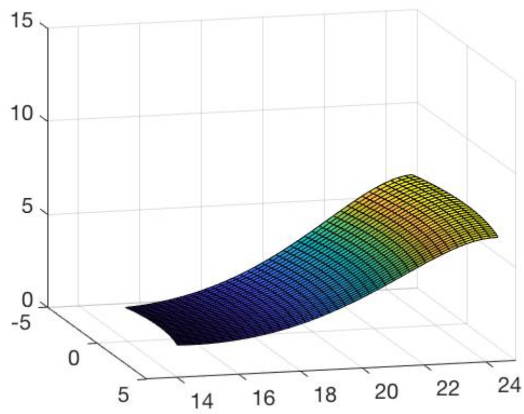


Figure 18e: Time Step 75

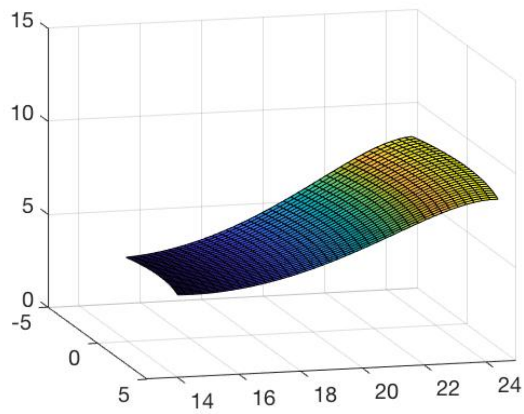


Figure 18f: Time Step 500

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.

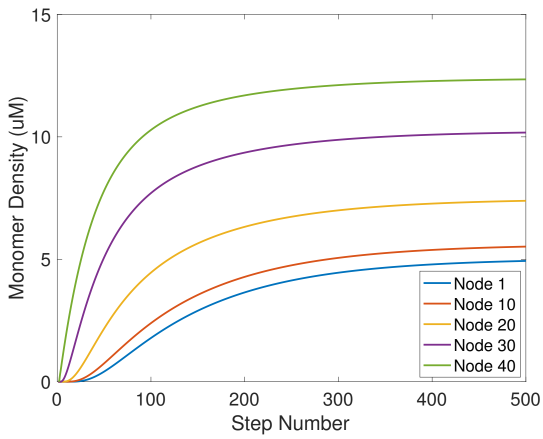


Figure 19a: Transport Problem G Nodal Evolution

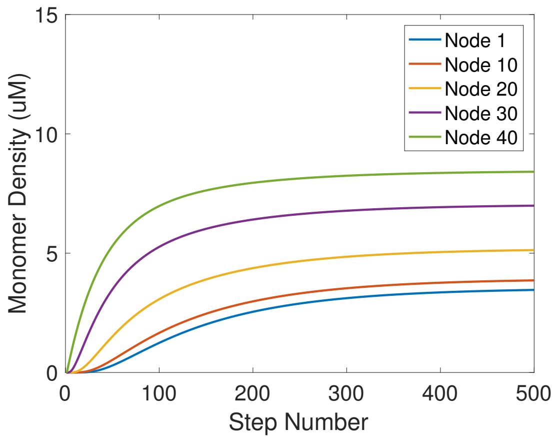


Figure 19b: Coupled Problem G Nodal Evolution

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.

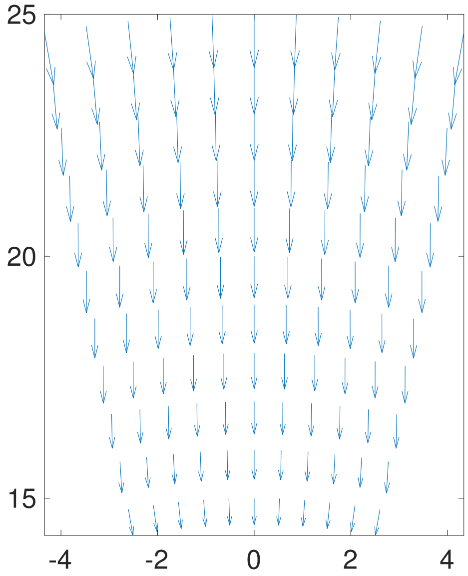


Figure 20: Resultant Velocity Field

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.

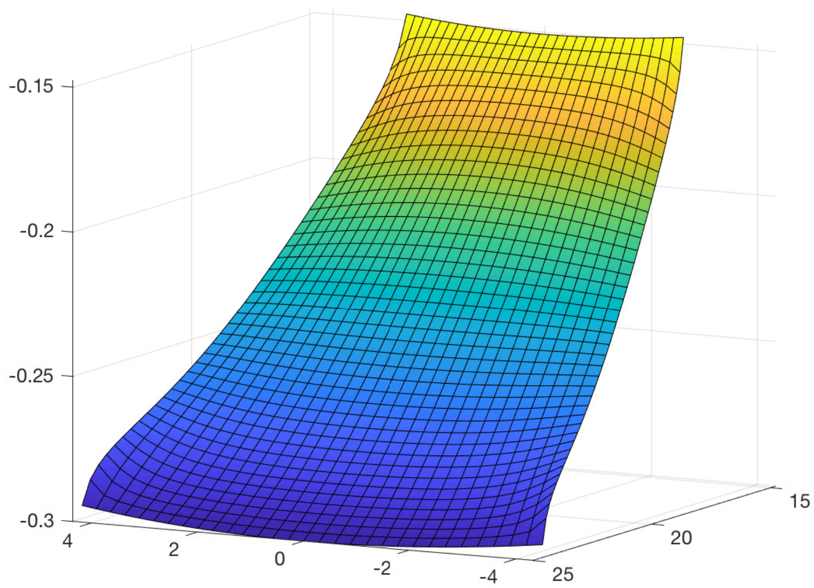


Figure 21: Y-Velocity Results

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.

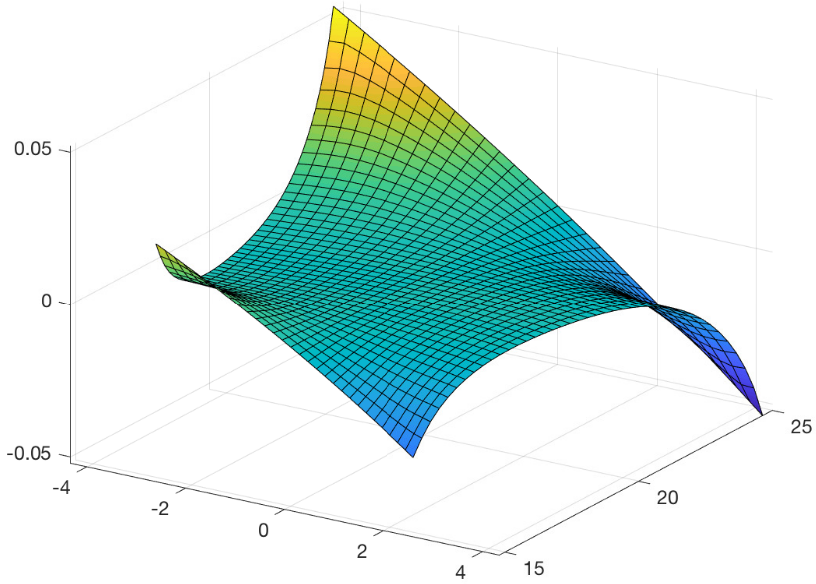


Figure 22: X-Velocity Results

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

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%THIS PROGRAM SOLVES THREE PROBLEMS RELATED TO ACTINS ROLE IN CELL
    MOTILITY


%BY ALEXANDER KEISER


clc
clear all
close all


disp('1 for Transport Problem')
disp('2 for Stokes Problem')
disp('3 for Coupled Problem')
problem=input('Enter Problem Type')

if problem==1;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%MATERIAL PARAMETERS
THETA=2/3; SIGMA_GF=0.5; SIGMA_G=2; SIGMA_F=0.25; D_G=15; D_F=5;
                         %s^-1         %s^-1          %s^-1    %um/s     %um/s
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%GENERATION OF THE MESH

%NUMBER OF ELEMENTS IN RADIAL DIRECTION
N_R = 40;

%NUMBER OF ELEMENTS IN THETA DIRECTION
N_THETA = 30;

%CREATE THE MESH
[X,T] = createMesh1(N_R,N_THETA);

%BILINEAR QUADRATIC ELEMENTS
elem = 0;

%PLOT THE MESH
figure(25)
plotMesh(T,X,elem,'k');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%SETS DEGREES OF FREEDOM
DEGREES_OF_FREEDOM = size(X,1);
ZERO_ARRAY=zeros(DEGREES_OF_FREEDOM,1);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%CALCULATION OF GLOBAL CONVECTIVE VELOCITY

CONVECTIVE_VELOCITY = velocity(X);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%GENERATION OF INITIAL TIME VECTORS

F_VECTOR = zeros(DEGREES_OF_FREEDOM,1);
G_VECTOR = zeros(DEGREES_OF_FREEDOM,1);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% FOUR POINT GAUSS QUADRATURE AND SHAPE FUNCTIONS

 ngaus = 4;
 [pospg,wpg] = Quadrature1(ngaus);
 [N,Nxi,Neta] = ShapeFunc1(pospg);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% COMPUTATION OF THE MATRICES FROM DESCRETIZATION

M = CreMassMat1(X,T,pospg,wpg,N,Nxi,Neta);
C = CreConvMat1(X,T,CONVECTIVE_VELOCITY,pospg,wpg,N,Nxi,Neta);
K = CreStiffMat1(X,T,CONVECTIVE_VELOCITY,pospg,wpg,N,Nxi,Neta);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% DEFINITION OF TIME PARAMETERS
END_TIME = 10;
NUMBER_TIME_STEPS = 499;
DT = END_TIME/NUMBER_TIME_STEPS;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%IMPLEMENTATION OF GALERKIN TIME AND SPACE DESCRETIZATION

 A1 = M + THETA*C*DT + THETA*D_F*K*DT + THETA*SIGMA_F*M*DT;

 B1 = -C*DT - D_F*K*DT - SIGMA_F*M*DT;

 A2 = M + THETA*D_G*K*DT + THETA*SIGMA_G*M*DT;

 B2 = -D_G*K*DT - SIGMA_G*M*DT;

 C2 = SIGMA_GF*M*DT;

```

```
F_1 = ZERO_ARRAY;

F_2 = ZERO_ARRAY;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%IMPOSITION OF THE BOUNDARY CONDITIONS

%DEFINE THE RADIUS OF EACH NODE IN THE DOMAIN
NODAL_RADIUS = sqrt(X(:,1).^2+X(:,2).^2);

%FIND THE NODES WITH VALUES TO BE IMPOSED AT RADIUS OF 25
DIRICHLET_NODES = find(NODAL_RADIUS >= 24.9999);

%DEFINE THE VALUE OF THE DIRICHLET BC
DIRICHLET_VALUE = 80;

%DEFINE THE NUMBER OF NODES TO BE IMPOSED WITH A BC
NUMBER_IMPOSITIONS = length(DIRICHLET_NODES);

%CREATE VECTOR WITH NODES AND THEIR RESPECTIVE BD VALUES
DIRICHLET_VECTOR = [DIRICHLET_NODES, ones(length(DIRICHLET_NODES),1)*
    DIRICHLET_VALUE];

%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%CALCULATION OF THE SOLUTION

%INITIALIZE MATRIX TO STORE THE LAGRANGE MULTIPLIERS
LAGRANGE = zeros(NUMBER_IMPOSITIONS,DEGREES_OF_FREEDOM);

%RECTIFY F VECTOR WITH DIRICHLET BC
F_VECTOR(DIRICHLET_VECTOR(:,1)) = 80;

%APPLY MULTIPLIER VALUES TO THE INITIALIZED LAGRANGE MATRIX
LAGRANGE (:,DIRICHLET_VECTOR(:,1)) = eye(NUMBER_IMPOSITIONS);

%VECTOR ADDED TO OTHER SIDE OF EQUATION TO BALANCE THE EQUATION
BALANCE_LAGRANGE = zeros(NUMBER_IMPOSITIONS,1);

%CREATE GLOBAL MATRIX TO CALCULATE SOLUTION
A_GLOBAL = [A1 LAGRANGE'; LAGRANGE zeros(NUMBER_IMPOSITIONS,
    NUMBER_IMPOSITIONS)];

%SOLUTION FOR F AND G EQUATIONS
for i = 1:NUMBER_TIME_STEPS
    B_GLOBAL = [B1*F_VECTOR(:,i)+F_1; BALANCE_LAGRANGE];
    DELTA_F = A_GLOBAL\B_GLOBAL;
    F_VECTOR(:,i+1) = DELTA_F(1:DEGREES_OF_FREEDOM)+F_VECTOR(:,i);
    F_VECTOR(DIRICHLET_VECTOR(:,1),i+1) = 80;
```

```
    BG_GLOBAL = B2*G_VECTOR(:,i)+F_2+C2*F_VECTOR(:,i)-THETA*C2*DELTA_F(1:DEGREES_OF_FREEDOM);
    DELTA_G = A2\BG_GLOBAL;
    G_VECTOR(:,i+1) = DELTA_G+G_VECTOR(:,i);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%POSTPROCESSING

if max(F_VECTOR(:,NUMBER_TIME_STEPS+1))<100 &&  min(F_VECTOR(:,NUMBER_TIME_STEPS+1))>-100

     figure(1); clf;
     [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,NUMBER_TIME_STEPS+1));
     surface(xx,yy,sol);
     view([40,30])
     axis auto
     grid on;




     figure(3); clf;
     [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,1));
     surface(xx,yy,sol);
     view([75,15])
     axis auto
     grid on;
     set (gca, 'fontsize', 20)
     zlim([0 90])
     ylim([13 25])


     figure(4); clf;
     [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,2));
     surface(xx,yy,sol);
     view([75,15])
     axis auto
     grid on;
     set (gca, 'fontsize', 20)
     zlim([0 90])
     ylim([13 25])


     figure(5); clf;
     [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,5));
     surface(xx,yy,sol);
     view([75,15])
     axis auto
     grid on;
     set (gca, 'fontsize', 20)
     zlim([0 90])
```

```
    ylim([13 25])


    figure(6); clf;
    [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,15));
    surface(xx,yy,sol);
    view([75,15])
    axis auto
    grid on;
    set (gca, 'fontsize', 20)
    zlim([0 90])
    ylim([13 25])


    figure(7); clf;
    [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,75));
    surface(xx,yy,sol);
    view([75,15])
    axis auto
    grid on;
    set (gca, 'fontsize', 20)
    zlim([0 90])
    ylim([13 25])


    figure(8); clf;
    [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,500));
    surface(xx,yy,sol);
    view([75,15])
    axis auto
    grid on;
    set (gca, 'fontsize', 20)
    zlim([0 90])
    ylim([13 25])




    figure(2); clf;
    set(gca,'FontSize',12);
    [C,h]=contour(xx,yy,sol);
    clabel(C,h);
    axis auto


    figure(10); clf;
    peli = moviein(NUMBER_TIME_STEPS+1);
    axis auto
    SOLUTION1=zeros(1,NUMBER_TIME_STEPS+1);
     SOLUTION2=zeros(1,NUMBER_TIME_STEPS+1);
     SOLUTION3=zeros(1,NUMBER_TIME_STEPS+1);
     SOLUTION4=zeros(1,NUMBER_TIME_STEPS+1);
     SOLUTION5=zeros(1,NUMBER_TIME_STEPS+1);
```

```
        for  n=1:NUMBER_TIME_STEPS+1
             [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,n));
            SOLUTION1(n)=sol(1,11);
             SOLUTION2(n)=sol(10,11);
             SOLUTION3(n)=sol(20,11);
             SOLUTION4(n)=sol(30,11);
             SOLUTION5(n)=sol(40,11);
            surf(xx,yy,sol);
            zlim([0  90])
        ylim([13  25])
            pause(0.01)
            peli(:,n) = getframe;
        end
end
TIMESTEPS=1:NUMBER_TIME_STEPS+1
plot(TIMESTEPS,SOLUTION1,'LineWidth',2)
hold on
plot(TIMESTEPS,SOLUTION2,'LineWidth',2)
hold on
plot(TIMESTEPS,SOLUTION3,'LineWidth',2)
hold on
plot(TIMESTEPS,SOLUTION4,'LineWidth',2)
hold on
plot(TIMESTEPS,SOLUTION5,'LineWidth',2)
xlabel('Step Number')
ylabel('Actin Filament Density (uM)')
legend('Node 1','Node 10','Node 20','Node 30','Node 40')
set(gca,'FontSize',20)
xlim([0 500])
pause(0.1);


        figure(23); clf;
        [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,1));
        surface(xx,yy,sol);
        view([75,15])
        axis auto
        grid on;
        set (gca, 'fontsize', 20)
        zlim([0 15])
        ylim([13 25])


        figure(24); clf;
        [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,5));
        surface(xx,yy,sol);
        view([75,15])
        axis auto
        grid on;
        set (gca, 'fontsize', 20)
        zlim([0 15])
        ylim([13 25])
```

```
    figure(25); clf;
    [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,10));
    surface(xx,yy,sol);
    view([75,15])
    axis auto
    grid on;
    set (gca, 'fontsize', 20)
    zlim([0 15])
    ylim([13 25])


    figure(26); clf;
    [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,15));
    surface(xx,yy,sol);
    view([75,15])
    axis auto
    grid on;
    set (gca, 'fontsize', 20)
    zlim([0 15])
    ylim([13 25])


    figure(27); clf;
    [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,75));
    surface(xx,yy,sol);
    view([75,15])
    axis auto
    grid on;
    set (gca, 'fontsize', 20)
    zlim([0 15])
    ylim([13 25])


    figure(28); clf;
    [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,500));
    surface(xx,yy,sol);
    view([75,15])
    axis auto
    grid on;
    set (gca, 'fontsize', 20)
    zlim([0 15])
    ylim([13 25])

%END G SOLUTION
figure(11); clf;
[xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,NUMBER_TIME_STEPS+1));
surface(xx,yy,sol);
view([40,30])
grid on;

%END G SOLUTION
```

```
figure(12); clf;
set(gca,'FontSize',12);
[C,h]=contour(xx,yy,sol);
clabel(C,h);

%G MOVIE
figure(20); clf;
peli = moviein(NUMBER_TIME_STEPS+1);
        SOLUTION1=zeros(1,NUMBER_TIME_STEPS+1);
        SOLUTION2=zeros(1,NUMBER_TIME_STEPS+1);
        SOLUTION3=zeros(1,NUMBER_TIME_STEPS+1);
        SOLUTION4=zeros(1,NUMBER_TIME_STEPS+1);
        SOLUTION5=zeros(1,NUMBER_TIME_STEPS+1);
for n=1:NUMBER_TIME_STEPS+1
      [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,n));
            SOLUTION1(n)=sol(1,11);
            SOLUTION2(n)=sol(10,11);
            SOLUTION3(n)=sol(20,11);
            SOLUTION4(n)=sol(30,11);
            SOLUTION5(n)=sol(40,11);
      surf(xx,yy,sol);
      zlim([0 15])
      ylim([13 25])
      pause(0.0001)
      peli(:,n) = getframe;
end
TIMESTEPS=1:NUMBER_TIME_STEPS+1
figure(50)
plot(TIMESTEPS,SOLUTION1,'LineWidth',2)
hold on
plot(TIMESTEPS,SOLUTION2,'LineWidth',2)
hold on
plot(TIMESTEPS,SOLUTION3,'LineWidth',2)
hold on
plot(TIMESTEPS,SOLUTION4,'LineWidth',2)
hold on
plot(TIMESTEPS,SOLUTION5,'LineWidth',2)
xlabel('Step Number')
ylabel('Monomer Density (uM)')
legend('Node 1','Node 10','Node 20','Node 30','Node 40')
ylim([0 15])
xlim([0 500])
set(gca,'FontSize',20)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


elseif problem==2;


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear; close all; clc


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%GENERATING THE MESH

N_R = 10;
N_THETA = 10;
MU=1000
[XP,TP] = createMesh2(N_R,N_THETA);
[X,T,THETA_VAR] = createMesh_velocity(N_R,N_THETA);
elem = 0;


figure(109)
plotMesh(TP,XP,elem);

figure(200)
plotMesh(T,X,elem,'k');


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%BILINEAR QUADRILATERAL PRESSURE MESH WITH QUADRATIC VELOCITY
elemV = 0; degreeV = 2; degreeP = 1;
elemP = elemV;
referenceElement = SetReferenceElementStokes(elemV,degreeV,elemP,degreeP);

% DESCRETIZATION MATRICIES
[K,G,f] = StokesSystem(X,T,XP,TP,referenceElement);
K = MU*K;
[DOF_PRESSURE,DOF_VELOCITY] = size(G);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%CONVERT NUMBER OF ELEMENTS TO NUMBERS OF NODES
```

```
NN_R=2*N_R+1
NN_THETA=2*N_THETA+1

%FIND NODES FOR DIRICHLET
NODES_Y1=1:(NN_THETA);
NODES_Y2=(NN_R-1)*NN_THETA+1:(NN_THETA)*(NN_R);
NODES_Y1=NODES_Y1';
NODES_Y2=NODES_Y2';

%NODES TO BE IMPOSED ON
NODES_DIR_BC = [NODES_Y1; NODES_Y2];


%NUMBER OF DEGREES OF FREEDOM ON DIRICHLET NODES
DIR_DOF = 2*length(NODES_DIR_BC) ;
%CONFINED FLOW
 confined = 1;

%MATRICIES TO IMPOSE DIRICHLET BOUNDARY CONDITIONS
C_MATRIX = [2*NODES_DIR_BC - 1; 2*NODES_DIR_BC];
C_STEP=reshape(C_MATRIX,DIR_DOF/2,2);
C_BAL =reshape(C_STEP',DIR_DOF,1);
A_DIR_MAT = zeros(DIR_DOF,DOF_VELOCITY);
A_DIR_MAT(:,C_BAL) = eye(DIR_DOF);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%IMPOSITION OF THE BOUNDARY CONDITIONS

BC_THETA1 = 0;
BC_THETA2 = 0;
BC_RADIAL1 = -0.15;
BC_RADIAL2 = -0.3;
VELOCITY_BC_X1 = BC_THETA1*sin(pi/2- THETA_VAR ) + BC_RADIAL1*cos(pi/2-
    THETA_VAR);
VELOCITY_BC_X2 = BC_THETA2*sin(pi/2-THETA_VAR) + BC_RADIAL2*cos(pi/2-
    THETA_VAR);
VELOCITY_BC_Y1 = -BC_THETA1*cos(pi/2-THETA_VAR) + BC_RADIAL1*sin(pi/2-
    THETA_VAR);
VELOCITY_BC_Y2 = -BC_THETA2*cos(pi/2-THETA_VAR) + BC_RADIAL2*sin(pi/2-
    THETA_VAR);
B_STEP = [VELOCITY_BC_X1' VELOCITY_BC_Y1';...
    VELOCITY_BC_X2'    VELOCITY_BC_Y2'];
B_DIR_VEC =reshape(B_STEP',DIR_DOF,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%GENERATE ENTIRE SYSTEM OF EQUATIONS

 if confined
    nunkP = DOF_PRESSURE-1;
    disp(' ')
    disp('Confined flow. Pressure on lower left corner is set to zero');
```

```
    G(1,:) = [];
else
    nunkP = DOF_PRESSURE;
end
Atot = [K            A_DIR_MAT'              G'
        A_DIR_MAT      zeros(DIR_DOF,DIR_DOF)      zeros(DIR_DOF,nunkP)
        G          zeros(nunkP,DIR_DOF)      zeros(nunkP,nunkP)];
btot = [f ; B_DIR_VEC ; zeros(nunkP,1)];

sol = Atot\btot ;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% POSTPROCESS


velo = reshape(sol(1:DOF_VELOCITY), 2, []) ';

if confined
     pres = [0; sol(DOF_VELOCITY+DIR_DOF+1:DOF_VELOCITY+DIR_DOF+nunkP)];
else
         pres = sol(DOF_VELOCITY+DIR_DOF+1:DOF_VELOCITY+DIR_DOF+nunkP);
end

nPt = size(X,1);
figure('Name','TIGHT');
quiver(X(1:nPt,1),X(1:nPt,2),velo(1:nPt,1),velo(1:nPt,2));
hold on
axis equal; axis tight


PlotResults(X,T,velo(:,1),referenceElement.elemV,referenceElement.degreeV)

PlotResults(X,T,velo(:,2),referenceElement.elemV,referenceElement.degreeV)

if degreeP == 0
     PlotResults(X,T,pres,referenceElement.elemP,referenceElement.degreeP)
else
     PlotResults(XP,TP,pres,referenceElement.elemP,referenceElement.degreeP
        )
end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

elseif problem==3;


```

```
clc
clear all
close all


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%MATERIAL PARAMETERS
THETA=2/3; SIGMA_GF=0.5; SIGMA_G=2; SIGMA_F=0.25; D_G=15; D_F=5;
                       %s^-1        %s^-1        %s^-1   %um/s   %um/s

mu=1000

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%GENERATION OF THE MESH

%NUMBER OF NODES IN RADIAL DIRECTION
N_R = 40

%NUMBER OF NODES IN THETA DIRECTION
N_THETA = 30

elemV = 0; degreeV = 1; degreeP = 1;
elemP = elemV;
referenceElement = SetReferenceElementStokes(elemV,degreeV,elemP,degreeP);
Xe_ref = referenceElement.Xe_ref;
[X,T,XP,TP,THETA_VAR] = CreateMeshes(N_THETA,N_R,referenceElement);
figure()
plot_Mesh(T,X,elemV,'b-')
figure()
plot_Mesh(TP,XP,elemV,'r-')


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%APPLICATION OF THE BOUNDARY CONDITIONS

%OBTAIN MATRICIES
[T_F,T_U] = boundaryMatrices(X,T,elemV,degreeV,Xe_ref)
[K_STOKES,G_VECTOR,f] = StokesSystem2(X,T,XP,TP,referenceElement);
K_STOKES = mu*K_STOKES;
ndofV = size(K_STOKES,1);

%CONVERT NUMBER OF ELEMENTS TO NUMBERS OF NODES
NN_R=N_R+1
NN_THETA=N_THETA+1

%FIND NODES FOR DIRICHLET
NODES_Y1=1:(NN_THETA);
NODES_Y2=(NN_R-1)*NN_THETA+1:(NN_THETA)*(NN_R);
NODES_Y1=NODES_Y1';
NODES_Y2=NODES_Y2';

```

```
%NODES TO BE IMPOSED ON
NODES_DIR_BC = [NODES_Y1; NODES_Y2]


%NUMBER OF DEGREES OF FREEDOM ON DIRICHLET NODES
DIR_DOF = 2*length(NODES_DIR_BC) ;

%CONFINED FLOW
confined = 1;

%MATRICIES TO IMPOSE DIRICHLET BOUNDARY CONDITIONS
C_MATRIX = [2*NODES_DIR_BC - 1; 2*NODES_DIR_BC];
C_STEP=reshape(C_MATRIX,DIR_DOF/2,2);
C_BAL =reshape(C_STEP',DIR_DOF,1);
A_DirBC = zeros(DIR_DOF,ndofV);
A_DirBC(:,C_BAL) = eye(DIR_DOF);

BC_THETA1 = 0;
BC_THETA2 = 0;
BC_RADIAL1 = -0.15;
BC_RADIAL2 = -0.3;
VELOCITY_BC_X1 = BC_THETA1*sin(pi/2- THETA_VAR ) + BC_RADIAL1*cos(pi/2-THETA_VAR);
VELOCITY_BC_X2 = BC_THETA2*sin(pi/2-THETA_VAR) + BC_RADIAL2*cos(pi/2-THETA_VAR);
VELOCITY_BC_Y1 = -BC_THETA1*cos(pi/2-THETA_VAR) + BC_RADIAL1*sin(pi/2-THETA_VAR);
VELOCITY_BC_Y2 = -BC_THETA2*cos(pi/2-THETA_VAR) + BC_RADIAL2*sin(pi/2-THETA_VAR);

B_STEP = [VELOCITY_BC_X1' VELOCITY_BC_Y1';...
    VELOCITY_BC_X2'    VELOCITY_BC_Y2']

%RHS BC ENFORCEMENT VECTOR
B_DIR_VEC =reshape(B_STEP',DIR_DOF,1)
VELOCITY_DIR = [C_BAL B_DIR_VEC]


%MATRIX FOR BC ENFORCEMENT
A_DIR_MAT = K_STOKES + T_U;
A_DIR_MAT(VELOCITY_DIR(:,1),:) = 0;
A_DIR_MAT(:,VELOCITY_DIR(:,1)) = 0;
A_DIR_MAT(VELOCITY_DIR(:,1),VELOCITY_DIR(:,1)) = eye(DIR_DOF);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%GAUSS INTEGRATION

ngaus=4;
```

```
[pospg,wpg] = Quadrature(elemV,ngaus)

[N,Nxi,Neta] = ShapeFunc(elemV,degreeV,pospg);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%GENERATION OF M AND K MATRICIES

M = CreMassMat(X,T,pospg,wpg,N,Nxi,Neta);

K = CreStiffMat(X,T,pospg,wpg,N,Nxi,Neta);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% DEFINITION OF TIME PARAMETERS
END_TIME = 10;
NUMBER_TIME_STEPS = 499;
DT = END_TIME/NUMBER_TIME_STEPS;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%ENFORCING THE BOUNDARY CONDITION FOR F
NUMBER_OF_NODES = size(X,1);
F_VECTOR = zeros(NUMBER_OF_NODES,NUMBER_TIME_STEPS+1);
DIRICHLET_VALUE = 80;
F_DIRICHLET = 0;
 for i = 1:size(X,1)
     poly = abs(X(i,1)^2 + X(i,2)^2 - 25^2);
     if poly <= 10^-6
          F_DIRICHLET = F_DIRICHLET + 1;
          F_VECTOR(i,:) = DIRICHLET_VALUE;
     end
 end
 nodesD = ((N_THETA+1)*(N_R+1)-N_THETA:(N_THETA+1)*(N_R+1))';
 CDIR = [nodesD,zeros(length(nodesD),1)];
 F_DIRICHLET = size(CDIR,1);

ACCD_F  = zeros(F_DIRICHLET,NUMBER_OF_NODES);
ACCD_F(:,CDIR(:,1)) = eye(F_DIRICHLET);
BCCD_F = CDIR(:,2);

STORE_V(:,1) = zeros(ndofV,1);
STORE_V(VELOCITY_DIR(:,1)) = VELOCITY_DIR(:,2);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%INITIALIZE G EQUATIONS

G_VECTOR = zeros(NUMBER_OF_NODES,NUMBER_TIME_STEPS+1);
```

```
A_G = M + THETA*D_G*K*DT + THETA*SIGMA_G*M*DT;
B_G = -D_G*K*DT - SIGMA_G*M*DT;
C_G = SIGMA_GF*M*DT;
f_G = zeros(NUMBER_OF_NODES,1);
Atot_G = A_G;
[L2,U2] = lu(Atot_G);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


% TRANSIENT SOLUTION
tol = 10^-6;


for n= 1:NUMBER_TIME_STEPS




    %INITIALIZE VELOCITY INCREMENT
    VELOCITY_INCREMENT = 1;  FInc = 1;
    iter = 0;
    g = F_VECTOR(:,n);
    n=n



    %RUN LOOP WHILE THE VELOCITY INCREMENT IS GREATER THAN TOLERANCE
    while VELOCITY_INCREMENT>=tol
        iter = iter + 1;

        %UPDATE F USING NODAL DENSITY MATRIX
        fu = -T_F*g(:,iter);


        %RUN LOOP FOR SIZE OF DESCRETIZED KMATRIX
        for i =1:size(K_STOKES,1)

            %UPDATING BC VALUES FOR LOOP
            F_UG(i,1) = fu(i,1)-K_STOKES(i,VELOCITY_DIR(:,1))*B_DIR_VEC;

        end

        %UPDATING F GLOBAL FOR NODAL COORDINATES
        for i = 1:DIR_DOF
            F_UG(VELOCITY_DIR(i,1)) = VELOCITY_DIR(i,2);
        end

        STORE_V(:,iter+1) = A_DIR_MAT\F_UG;
        VEL_VEC = STORE_V(:,iter+1);

```

```
        %CREATE THIS ITERATIONS CONVECTION MATRIX
        VEL_CONV = reshape(VEL_VEC,2,[])';
        C_MATRIX = CreConvMat(X,T,VEL_CONV,pospg,wpg,N,Nxi,Neta);

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%

        %%CALCULATE SOLUTION FOR F

        %A MATRIX RESULTING FROM DESCRETIZATION
        A_F = M + THETA*C_MATRIX*DT + THETA*D_F*K*DT + THETA*SIGMA_F*M*DT;

        %B MATRIX RESULTING FROM DESCRETIZATION
        B_F = M + (1-THETA)*(-C_MATRIX*DT - D_F*K*DT - SIGMA_F*M*DT);

        %F MATRIX RESULTING FROM F DESCRETIZATION
        F_F = zeros(NUMBER_OF_NODES,1);

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%

        % CREATE GLOBAL MATRIX WITH LAGRANGE FOR INNER LOOP
        A_INNERLOOP_F = [A_F ACCD_F';ACCD_F zeros(F_DIRICHLET,F_DIRICHLET)];
        [L1,U1] = lu(A_INNERLOOP_F);

        B_INNERLOOP_F = [B_F*F_VECTOR(:,n)+ F_F; BCCD_F];

        LAGRANGE_1 = U1\(L1\B_INNERLOOP_F);
        g(:,iter+1) = LAGRANGE_1(1:NUMBER_OF_NODES);

        VELOCITY_INCREMENT = norm(STORE_V(:,iter+1)-STORE_V(:,iter));

    end

    %OVERWRITE C MATRIX USING VELOCITY VECTOR
    V_STORE_2(:,n) = STORE_V(:,iter+1);
    VEL_VEC = V_STORE_2(:,n);
    VEL_CONV = reshape(VEL_VEC,2,[])';
    C_MATRIX = CreConvMat(X,T,VEL_CONV,pospg,wpg,N,Nxi,Neta);

    %RECOMPUTATION OF MATRICIES
    A_F = M + THETA*C_MATRIX*DT + THETA*D_F*K*DT + THETA*SIGMA_F*M*DT;
```

```
    B_F = (-C_MATRIX*DT - D_F*K*DT - SIGMA_F*M*DT);
    F_F = zeros(NUMBER_OF_NODES,1);




    % CALCULATION OF SOLUTION AT TIME STEP
    A_INNERLOOP_F = [A_F ACCD_F';ACCD_F zeros(F_DIRICHLET,F_DIRICHLET)];
    [L1,U1] = lu(A_INNERLOOP_F);


    B_INNERLOOP_F = [B_F*F_VECTOR(:,n)+ F_F; BCCD_F];
    FA =  U1\(L1\B_INNERLOOP_F);


    FB = U1\(L1\B_INNERLOOP_F);

    F_VECTOR(:,n+1) = F_VECTOR(:,n) + FB(1:NUMBER_OF_NODES);


    %FINAL SOLUTION FOR G AT TIME STEP
        btot = [B_G*G_VECTOR(:,n) - THETA*C_G*FA(1:NUMBER_OF_NODES) +
           THETA*C_G*F_VECTOR(:,n+1) + f_G];
        aux_G  = U2\(L2\btot);
        G_VECTOR(:,n+1) = G_VECTOR(:,n) + aux_G(1:NUMBER_OF_NODES);

end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


figure()
nPt = size(X,1);
figure;
quiver(X(1:nPt,1),X(1:nPt,2),VEL_CONV(1:nPt,1),VEL_CONV(1:nPt,2));
hold on
axis equal; axis tight



figure()
PlotResults(X,T,VEL_CONV(:,1),referenceElement.elemV,referenceElement.
    degreeV)
figure()
PlotResults(X,T,VEL_CONV(:,2),referenceElement.elemV,referenceElement.
    degreeV)
v = sqrt((VEL_CONV(:,1).^2)+(VEL_CONV(:,2).^2));
figure()
PlotResults(X,T,v,referenceElement.elemV,referenceElement.degreeV)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%POSTPROCESSING

if max(F_VECTOR(:,NUMBER_TIME_STEPS+1))<100 && min(F_VECTOR(:,NUMBER_TIME_STEPS+1))>-100

    figure(); clf;
    [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,NUMBER_TIME_STEPS+1));
    surface(xx,yy,sol);
    view([40,30])
    axis auto
    grid on;




    figure(); clf;
    [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,1));
    surface(xx,yy,sol);
    view([75,15])
    axis auto
    grid on;
    set (gca, 'fontsize', 20)
    zlim([0 90])
    ylim([13 25])


    figure(); clf;
    [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,2));
    surface(xx,yy,sol);
    view([75,15])
    axis auto
    grid on;
    set (gca, 'fontsize', 20)
    zlim([0 90])
    ylim([13 25])


    figure(); clf;
    [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,5));
    surface(xx,yy,sol);
    view([75,15])
    axis auto
    grid on;
    set (gca, 'fontsize', 20)
    zlim([0 90])
    ylim([13 25])
```

```
figure(); clf;
[xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,15));
surface(xx,yy,sol);
view([75,15])
axis auto
grid on;
set (gca, 'fontsize', 20)
zlim([0 90])
ylim([13 25])


figure(); clf;
[xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,75));
surface(xx,yy,sol);
view([75,15])
axis auto
grid on;
set (gca, 'fontsize', 20)
zlim([0 90])
ylim([13 25])


figure(); clf;
[xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,500));
surface(xx,yy,sol);
view([75,15])
axis auto
grid on;
set (gca, 'fontsize', 20)
zlim([0 90])
ylim([13 25])




figure(); clf;
set(gca,'FontSize',12);
[C,h]=contour(xx,yy,sol);
clabel(C,h);
axis auto


figure(10); clf;
peli = moviein(NUMBER_TIME_STEPS+1);
axis auto
SOLUTION1=zeros(1,NUMBER_TIME_STEPS+1);
 SOLUTION2=zeros(1,NUMBER_TIME_STEPS+1);
 SOLUTION3=zeros(1,NUMBER_TIME_STEPS+1);
 SOLUTION4=zeros(1,NUMBER_TIME_STEPS+1);
 SOLUTION5=zeros(1,NUMBER_TIME_STEPS+1);

for n=1:NUMBER_TIME_STEPS+1
    [xx,yy,sol] = MatSol(X,N_THETA,N_R,F_VECTOR(:,n));
```

```
            SOLUTION1(n)=sol(1,11);
             SOLUTION2(n)=sol(10,11);
             SOLUTION3(n)=sol(20,11);
             SOLUTION4(n)=sol(30,11);
             SOLUTION5(n)=sol(40,11);
            surf(xx,yy,sol);
            zlim([0 90])
        ylim([13 25])
            pause(0.01)
            peli(:,n) = getframe;
        end
end
TIMESTEPS=1:NUMBER_TIME_STEPS+1
plot(TIMESTEPS,SOLUTION1,'LineWidth',2)
hold on
plot(TIMESTEPS,SOLUTION2,'LineWidth',2)
hold on
plot(TIMESTEPS,SOLUTION3,'LineWidth',2)
hold on
plot(TIMESTEPS,SOLUTION4,'LineWidth',2)
hold on
plot(TIMESTEPS,SOLUTION5,'LineWidth',2)
xlabel('Step Number')
ylabel('Actin Filament Density (uM)')
legend('Node 1','Node 10','Node 20','Node 30','Node 40')
set(gca,'FontSize',20)
xlim([0 500])
pause(0.1);


        figure(); clf;
        [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,1));
        surface(xx,yy,sol);
        view([75,15])
        axis auto
        grid on;
        set (gca, 'fontsize', 20)
        zlim([0 15])
        ylim([13 25])


        figure(); clf;
        [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,5));
        surface(xx,yy,sol);
        view([75,15])
        axis auto
        grid on;
        set (gca, 'fontsize', 20)
        zlim([0 15])
        ylim([13 25])


        figure(); clf;
```

```
    [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,10));
    surface(xx,yy,sol);
    view([75,15])
    axis auto
    grid on;
    set (gca, 'fontsize', 20)
    zlim([0 15])
    ylim([13 25])


    figure(); clf;
    [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,15));
    surface(xx,yy,sol);
    view([75,15])
    axis auto
    grid on;
    set (gca, 'fontsize', 20)
    zlim([0 15])
    ylim([13 25])


    figure(); clf;
    [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,75));
    surface(xx,yy,sol);
    view([75,15])
    axis auto
    grid on;
    set (gca, 'fontsize', 20)
    zlim([0 15])
    ylim([13 25])


    figure(); clf;
    [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,500));
    surface(xx,yy,sol);
    view([75,15])
    axis auto
    grid on;
    set (gca, 'fontsize', 20)
    zlim([0 15])
    ylim([13 25])

%END G SOLUTION
figure(); clf;
[xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,NUMBER_TIME_STEPS+1));
surface(xx,yy,sol);
view([40,30])
grid on;

%END G SOLUTION
figure(); clf;
set(gca,'FontSize',12);
[C,h]=contour(xx,yy,sol);
```

```
clabel(C,h);

%G MOVIE
figure(); clf;
peli = moviein(NUMBER_TIME_STEPS+1);
        SOLUTION1=zeros(1,NUMBER_TIME_STEPS+1);
        SOLUTION2=zeros(1,NUMBER_TIME_STEPS+1);
        SOLUTION3=zeros(1,NUMBER_TIME_STEPS+1);
        SOLUTION4=zeros(1,NUMBER_TIME_STEPS+1);
        SOLUTION5=zeros(1,NUMBER_TIME_STEPS+1);
for n=1:NUMBER_TIME_STEPS+1
    [xx,yy,sol] = MatSol(X,N_THETA,N_R,G_VECTOR(:,n));
            SOLUTION1(n)=sol(1,11);
            SOLUTION2(n)=sol(10,11);
            SOLUTION3(n)=sol(20,11);
            SOLUTION4(n)=sol(30,11);
            SOLUTION5(n)=sol(40,11);
    surf(xx,yy,sol);
    zlim([0 15])
    ylim([13 25])
    pause(0.0001)
    peli(:,n) = getframe;
end
TIMESTEPS=1:NUMBER_TIME_STEPS+1
figure(50)
plot(TIMESTEPS,SOLUTION1,'LineWidth',2)
hold on
plot(TIMESTEPS,SOLUTION2,'LineWidth',2)
hold on
plot(TIMESTEPS,SOLUTION3,'LineWidth',2)
hold on
plot(TIMESTEPS,SOLUTION4,'LineWidth',2)
hold on
plot(TIMESTEPS,SOLUTION5,'LineWidth',2)
xlabel('Step Number')
ylabel('Monomer Density (uM)')
legend('Node 1','Node 10','Node 20','Node 30','Node 40')
ylim([0 15])
xlim([0 500])
set(gca,'FontSize',20)
%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%




else
    disp('not valid selection, closing')
    exit
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

%%
%%
%%
%%