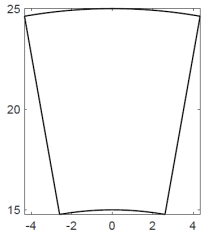
Finite element in fluid

Matlab Assignment 2018

Author : Seyed mohammadreza Attar Seyedi

June 2018

Actin plays an important role in cells motility. There are different models have been proposed to predict the concentration of actin monomers and filaments. We want to solve a model for density of actin filaments and monomers coupled with the cortexs mechanics. First, we try to solve a transport problem to obtain the actin densities. After that we solve a Stokes problem to obtain the velocity and pressure distribution of the fluid surrounding the actin filaments and monomers. At the end we have a coupled problem is proposed to account for the interaction of the actin filaments and the cortex.



Transport problem

Solving the transport problem for achieving the actin densities.
Convection diffusion reaction value problem :

$$u_t + a \cdot \nabla u - \nabla \cdot (\nu \nabla u) + \sigma u = s$$

The θ schemes applied for convection diffusion reaction equation :

$$\frac{\Delta \mathrm{u}}{\Delta \mathrm{t}} + \theta[\mathrm{a} \cdot \nabla - \nabla \cdot (\nu \nabla) + \sigma]\Delta \mathrm{u} = \theta \mathrm{s}^{\mathrm{n}+1} + (1-\theta)\mathrm{s}^{\mathrm{n}} - [\mathrm{a} \cdot \nabla - \nabla \cdot (\nu \nabla) + \sigma]\mathrm{u}^{\mathrm{n}}$$

Crank Nicolson is the second order accurate method and $\theta = 0.5$.

The Crank Nicolson method is used for transient problem that time accurate is important.

When the Peclet number is very high, the solution is pure convection.
Our Peclet number is low and we have convection diffusion equations and also Courant number is low.

$$\frac{\partial \mathrm{F}}{\partial \mathrm{t}} = -\mathrm{u} \cdot \nabla \mathrm{F} + \mathrm{D}_{\mathrm{F}} \nabla^2 \mathrm{F} - \sigma_{\mathrm{F}} \mathrm{F}$$

First equation contains convection, diffusion and reaction term.

Time Discretization :

$$\frac{\Delta \mathrm{F}}{\Delta \mathrm{T}} - \frac{1}{2}(-\mathrm{u} \cdot \nabla \Delta \mathrm{F} + \mathrm{D}_{\mathrm{f}} \nabla^2 \Delta \mathrm{F} - \sigma_{\mathrm{F}} \Delta \mathrm{F}) = -\mathrm{u} \cdot \nabla \mathrm{F}^{\mathrm{n}} + \mathrm{D}_{\mathrm{f}} \nabla^2 \mathrm{F}^{\mathrm{n}} - \sigma_{\mathrm{F}} \mathrm{F}^{\mathrm{n}}$$

Multiply by test function w

$$(\mathrm{w}, \frac{\Delta \mathrm{F}}{\Delta \mathrm{T}}) + \frac{1}{2}(\mathrm{w}, \mathrm{u} \cdot \nabla \Delta \mathrm{F}) - \frac{1}{2}(\mathrm{w}, \mathrm{D_f} \nabla^2 \Delta \mathrm{F}) + \frac{1}{2}(\mathrm{w}, \sigma_\mathrm{F} \Delta \mathrm{F}) = -(\mathrm{w}, \mathrm{u} \cdot \nabla \mathrm{F^n}) + (\mathrm{w}, \mathrm{D_f} \nabla^2 \mathrm{F^n}) - (\mathrm{w}, \sigma_\mathrm{F} \mathrm{F^n})$$

$C \rightarrow$ Convection matrix term $\quad C_{ab} = \int N_a (a \cdot \nabla N_b) d\Omega$
$K \rightarrow$ Diffusion matrix term $\quad K_{ab} = \int \nabla N_a \cdot (\nu \nabla N_b) d\Omega$
$M \rightarrow$ Consistent mass matrix $\quad M_{ab} = \int N_a N_b d\Omega$

AF = M + theta * C * dt + theta * DF * K * dt + theta * sigmaF * M * dt

BF = -C * dt - DF * K * dt - sigmaF * M * dt

theta = 0.5

AF = M + 0.5 * C * dt + 0.5 * DF * K * dt + 0.5 * sigmaF * M * dt

BF = -C * dt -DF * K * dt - sigmaF * M * dt

$DF = 5;$

$sigmaF = 0.25;$

$$\frac{\partial \mathrm{G}}{\partial \mathrm{t}} = \mathrm{D_G} \nabla^2 \mathrm{G} - \sigma_\mathrm{G} \mathrm{G} + \hat{\sigma}_\mathrm{GF} \mathrm{F}$$

The second equation contains diffusion, reaction and source term.

Time Discretization :

Using Crank Nicolson

$$\frac{\Delta \mathrm{G}}{\Delta \mathrm{T}} - \frac{1}{2}(\mathrm{D_G} \nabla^2 \Delta \mathrm{G} - \sigma_\mathrm{G} \Delta \mathrm{G} + \hat{\sigma}_\mathrm{GF} \Delta \mathrm{F}) = \mathrm{D_G} \nabla^2 \mathrm{G^n} - \sigma_\mathrm{G} \mathrm{G^n} + \hat{\sigma}_\mathrm{GF} \mathrm{F^n} + \frac{1}{2} \hat{\sigma}_\mathrm{GF} \Delta \mathrm{F}$$

Multiply by test function w

$$(\mathrm{w}, \frac{\Delta \mathrm{G}}{\Delta \mathrm{T}}) - \frac{1}{2}(\mathrm{w}, \mathrm{D_G} \nabla^2 \Delta \mathrm{G}) - (\mathrm{w}, \sigma_\mathrm{G} \Delta \mathrm{G}) + (\mathrm{w}, \hat{\sigma}_\mathrm{GF} \Delta \mathrm{F}) = (\mathrm{w}, \mathrm{D_G} \nabla^2 \mathrm{G^n}) - (\mathrm{w}, \sigma_\mathrm{G} \mathrm{G^n})$$

$$+(\mathrm{w}, \hat{\sigma}_\mathrm{GF} \mathrm{F^n}) + \frac{1}{2}(\mathrm{w}, \hat{\sigma}_\mathrm{GF} \Delta \mathrm{F})$$

AG = M + DG * K * dt + theta * sigmaG * M * dt

BG = -DG * k * dt - sigmaG * M *dt + theta * sigmaGF * M * dt + (1-theta) * sigmaGF * M *dt

theta = 0.5

AG = M + DG * k * dt + 0.5 * sigmaG * M * dt

BG = -DG * k * dt - sigmaG * M *dt + 0.5 * sigmaGF * M * dt + (1-0.5) * sigmaGF * M *dt

$DG = 15;$

$sigmaG = 2;$

$sigmaGF = 0.25;$

In the first equation, There is a crosswind,because the equation contain convection, diffusion and reaction terms.

In the second equation, it introduces excessive numerical diffusion, because we have diffusion, reaction and source term.

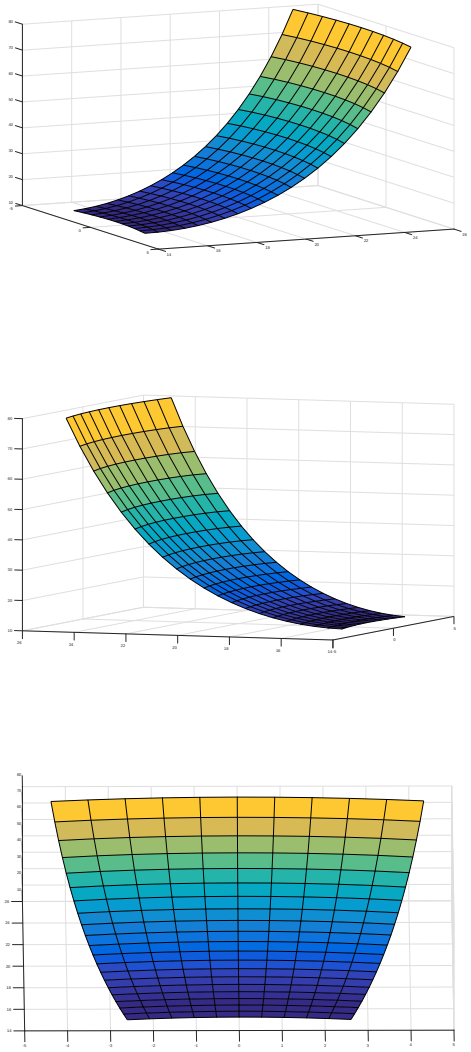


Figure 1: F density

Figure for F density increases until 80 and the value of Dirichlet boundary condition for F is 80.
Density is rising during the time.
Dirichlet boundary condition effects in the results.

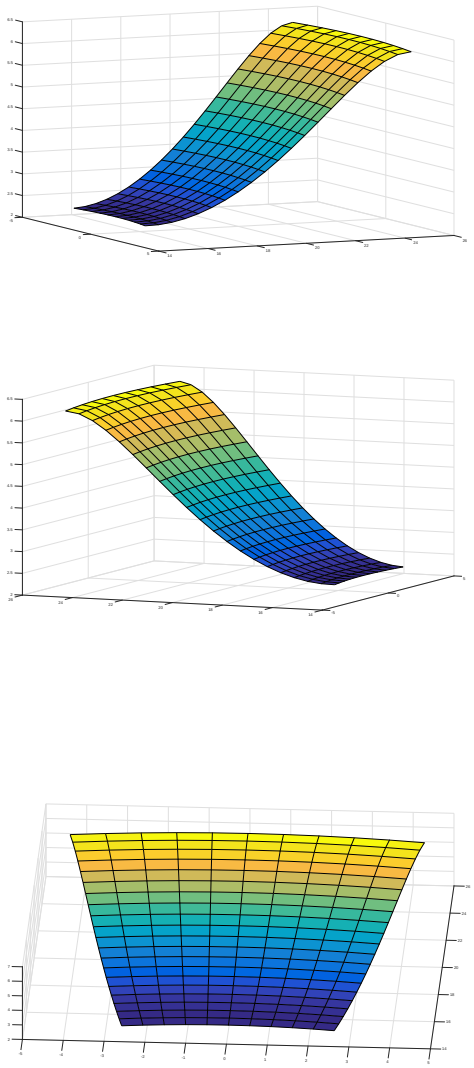


Figure 2: G density

Figure for actin G increases for diffusion dominants and the shape of this figure like a curve, finally at the end it remains constant.
Density is increasing sharply in the period of time.

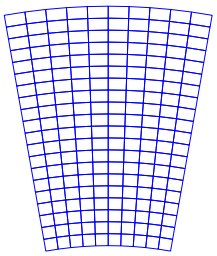


Figure 3: Mesh

Stokes

Solving the Stokes problem for achieving the velocity and pressure distribution of the fluid surrounding the actin filaments and monomers.

$$\nabla \cdot \sigma = 0$$

$$\nabla \cdot u = 0$$

$$u_r(r = 15) = 0.15 \qquad u_\theta(r = 15) = 0$$

$$u_r(r = 25) = 0.30 \qquad u_\theta(r = 25) = 0$$

σ is Cauchy stress and it must be related to pressure and velocity.
$b = 0$ it means that we do not have body forces.

Weak form :

$$-\int_\Omega \nabla w : \sigma d\Omega = 0$$

$$\int_\Omega q \nabla \cdot u d\Omega = 0$$

$w \rightarrow$Velocity test function.
$q \rightarrow$Pressure test function.

$$u = u_r cos(\theta) - u_\theta sin(\theta)$$

$$v = u_r sin(\theta) + u_\theta cos(\theta)$$

$$u_1 = u_r cos(\theta) - u_\theta sin(\theta)$$

$$u_2 = u_r cos(\theta) - u_\theta sin(\theta)$$

$$v_1 = u_r sin(\theta) + u_\theta cos(\theta)$$

$$v_2 = u_r sin(\theta) + u_\theta cos(\theta)$$

$u_\theta = 0$

$$u_1 = -0.15 * cos(\theta)$$

$$u_2 = -0.3 * cos(\theta)$$

$$v_1 = -0.15 * \sin(\theta)$$

$$v_2 = -0.30 * \sin(\theta)$$

Degree of freedom where velocity prescribed :

u → 2*Dirichlet Nodes - 1

v → 2*Dirichlet Nodes

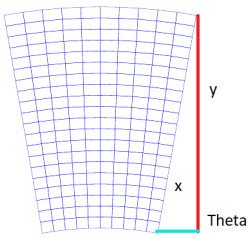
$u1 = -0.15. * cos(theta1);$

$u2 = -0.30. * cos(theta1);$

$v1 = -0.15. * sin(theta1);$

$v2 = -0.30. * sin(theta1);$

$A1 = [u1'v1'; u2'v2'];$



$$\tan(\theta) = \frac{y}{x} \quad \rightarrow \quad \theta = \tan^{-1}\frac{y}{x}$$

We can use this function **ataN2(Y,X)** in the Matlab for finding θ

$theta = ataN2(y, x);$

We just need to calculate the angle in the corner and of the mesh after finding θ we can put it in the formula and we achieve result for u and v.

The symmetric pressure matrix is positive definite only if ker(G) = 0. when the space used to approximate velocity and pressure satisfy LBB condition.
In LBB condition the velocity and pressure space cannot choose arbitrarily.
We can obtain the velocity and pressure distribution of the fluid surrounding actin filaments and monomers.

$$\begin{bmatrix} K & G \\ G^T & 0 \end{bmatrix} \begin{bmatrix} u \\ P \end{bmatrix} = \begin{bmatrix} f \\ h \end{bmatrix}$$

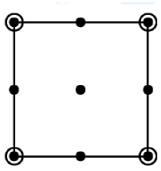
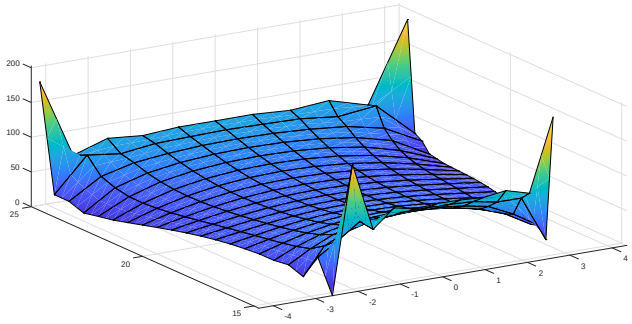


Figure 4: Q2Q1

In Q2Q1 element is so called Taylor-Hood element that continuous biquadratic velocity and continuous bilinear pressure.
Q2Q1 satisfied LBB condition and the convergence is quadratic.
When the LBB condition is not satisfied the stabilization techniques can be used but in this case Q2Q1 element is satisfied LBB condition and we do not need stabilization techniques.



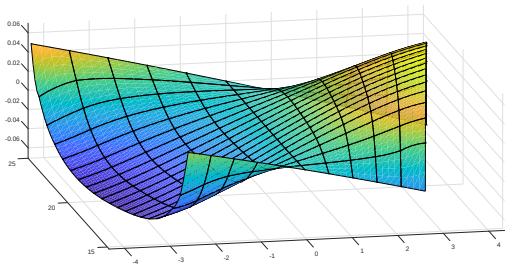


Figure 5: Q2Q1 x components

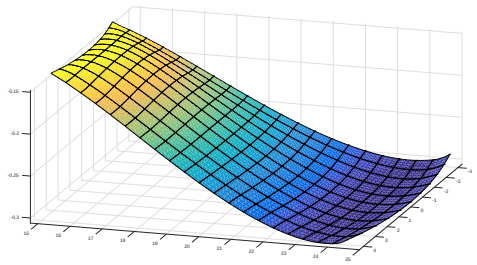


Figure 6: Q2Q1 y components

The results for Taylor-Hood element is nice, because in this element satisfy LBB complement and we do not need to add stabilization techniques.
Solutions for both velocity and pressure are acceptable.
At corners we have sharp pinnacle.
We have oscillation in x components and there is a crosswind in y components.

Q2Q1 produces a stable solution when satisfy the compatibility LBB condition and we achieve good results for both velocity and pressure, but for example for Q2Q0 which does not satisfy LBB condition show a spurious pressure. However the result for velocity is acceptable.
The characteristic distance between pressure and velocity is constant.

For this reason for Q2Q0 is not very good by using finite element Galerkin and we should use stabilization techniques like GLS for achieving good results for the case that does not satisfy LBB condition.

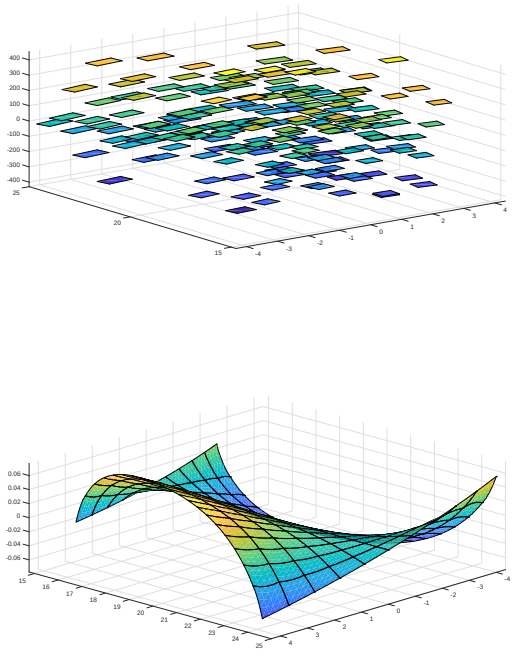


Figure 7: Q2Q0 x components

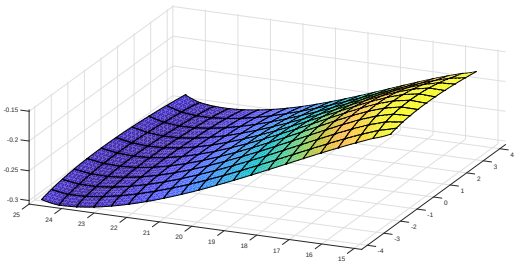


Figure 8: Q2Q0 y components

Q2Q0 produce spurious oscillations and it is natural for this element.

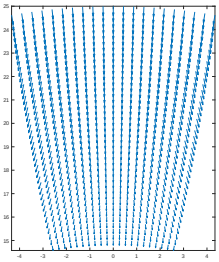


Figure 9: Velocity vector field

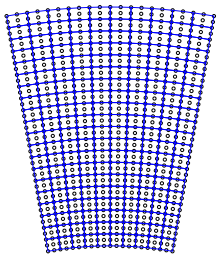


Figure 10: Velocity

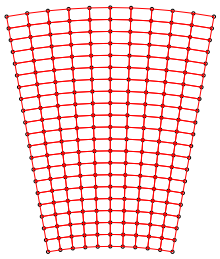


Figure 11: Pressure

Coupled

coupled problem is proposed to account for the interaction of the actin filaments and the cortex.

$$\nu\nabla\cdot(\nabla^{s}u)+\nabla\cdot\sigma_{m}(F)+T_{m}(u)=0$$

$$\frac{\partial F}{\partial t}=-u\cdot\nabla F+D_{F}\nabla^{2}F-\sigma_{F}F$$

$$\frac{\partial G}{\partial t}=D_{G}\nabla^{2}G-\sigma_{G}G+\hat{\sigma}_{GF}F$$

$$\nu\nabla\cdot(\nabla^{s}U)$$

Multiply test function w

$$(w,\nu\nabla\cdot(\nabla^{s}u))\rightarrow-(\nabla w:\nu\nabla^{s}u)+(w\nu(n\cdot\nabla^{s})u)$$

$$-Ku+T_{f}F+T_{m}u=0$$

$$(-K+T_{m})u=-T_{f}F$$

$$\frac{\partial F}{\partial t}=-u\cdot\nabla F+D_{F}\nabla^{2}F-\sigma_{F}F$$

Time Discretization :

$$\frac{\Delta F}{\Delta T}-\frac{1}{2}(-u\cdot\nabla\Delta F+D_{f}\nabla^{2}\Delta F-\sigma_{F}\Delta F)=-u\cdot\nabla F^{n}+D_{f}\nabla^{2}F^{n}-\sigma_{F}F^{n}$$

Multiply by test function w

$$(w,\frac{\Delta F}{\Delta T})+\frac{1}{2}(w,u\cdot\nabla\Delta F)-\frac{1}{2}(w,D_{f}\nabla^{2}\Delta F)+\frac{1}{2}(w,\sigma_{F}\Delta F)=-(w,u\cdot\nabla F^{n})+(w,D_{f}\nabla^{2}F^{n})-(w,\sigma_{F}F^{n})$$

AF = M + theta * C * dt + theta * DF * k * dt + theta * sigmaF * M * dt

BF = -C * dt - DF * k * dt - sigmaF * M * dt

theta = 0.5

AF = M + 0.5 * C * dt + 0.5 * DF * k * dt + 0.5 * sigmaF * M * dt

BF = -C * dt - DF * k * dt -sigmaF * M * dt

$$\frac{\partial G}{\partial t} = D_G \nabla^2 G - \sigma_G G + \hat{\sigma}_{GF} F$$

Time Discretization :

$$\frac{\Delta G}{\Delta T} - \frac{1}{2}(D_G \nabla^2 \Delta G - \sigma_G \Delta G + \hat{\sigma}_{GF} \Delta F) = D_G \nabla^2 G^n - \sigma_G G^n + \hat{\sigma}_{GF} F^n + \frac{1}{2}\hat{\sigma}_{GF} \Delta F$$

Multiply by test function w

$$(w, \frac{\Delta G}{\Delta T}) - \frac{1}{2}(w, D_G \nabla^2 \Delta G) - (w, \sigma_G \Delta G) + (w, \hat{\sigma}_{GF} \Delta F) = (w, D_G \nabla^2 G^n) - (w, \sigma_G G^n)$$

$$+(w, \hat{\sigma}_{GF} F^n) + \frac{1}{2}(w, \hat{\sigma}_{GF} \Delta F)$$

AG = M + theta * DG * k * dt + theta * sigmaG * M * dt

BG = -DG * k * dt - sigmaG * M *dt + theta * sigmaGF * M * dt + (1-theta) * sigmaGF * M *dt

theta = 0.5

AG = M + 0.5 * DG * k * dt + 0.5 * sigmaG * M * dt

BG = -DG * k * dt - sigmaG * M *dt + 0.5* sigmaGF * M * dt + (1-0.5) * sigmaGF * M *dt

Time discretization and add test function are the same for second and third equations, but for the first equation, we should use integration by part for the first term.

$$\int_\Omega -\nu \nabla w : \nabla v$$

$$-Ku + T_f F + T_m U = 0$$

$$\frac{1}{\Delta t} M\bar{F}^{n+1} + \frac{1}{2}[C(u)F^{n+1} + K\bar{F}^{n+1} + M\bar{F}^{n+1}]$$

$$\begin{bmatrix} K + Tu & Tf \\ 0 & A + C(u) \end{bmatrix} \begin{bmatrix} u^{n+1} \\ F^{n+1} \end{bmatrix}$$

$C(u^n)$ is linear we can solve it or for $C(u^{n+1})$ that it is nonlinear we can use Picard method.

Minimum F	-1.291198 e+0
Maximum F	8.000 e+01
Minimum G	1.003003 e-01
Maximum G	1.462221 e+0

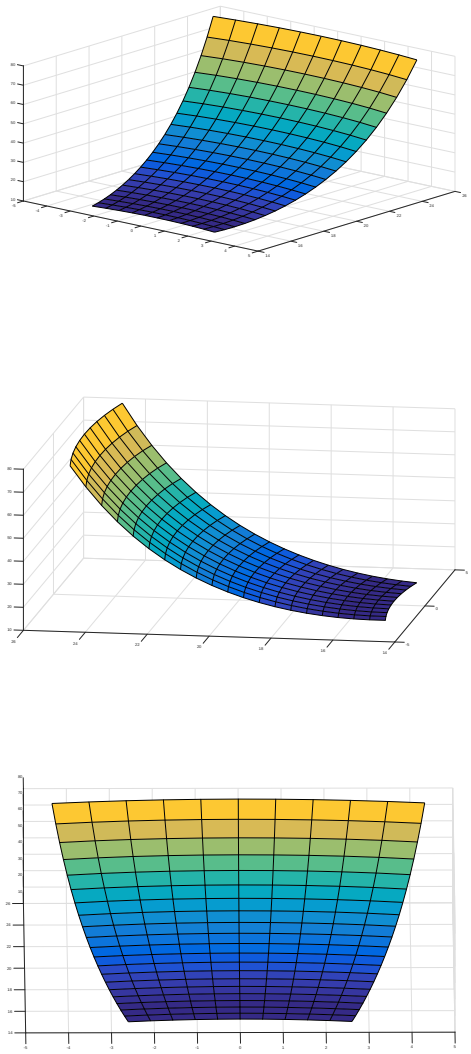


Figure 12: F density

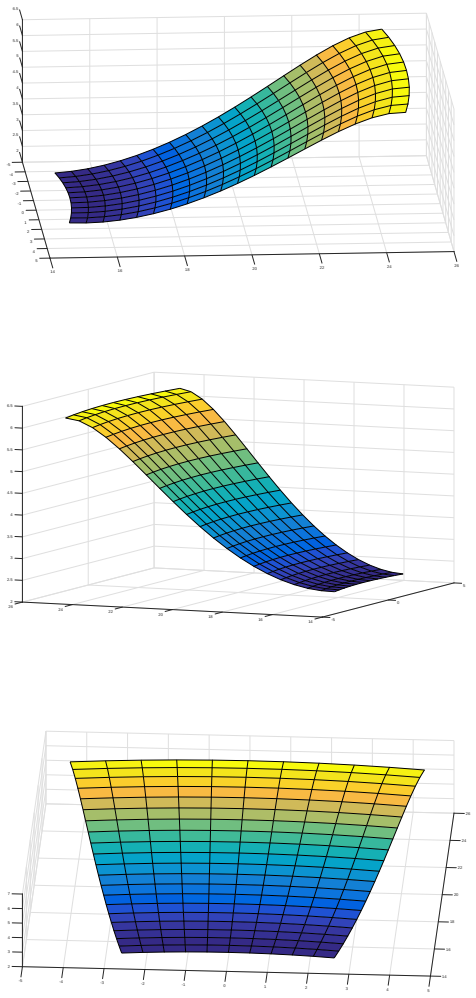


Figure 13: G density

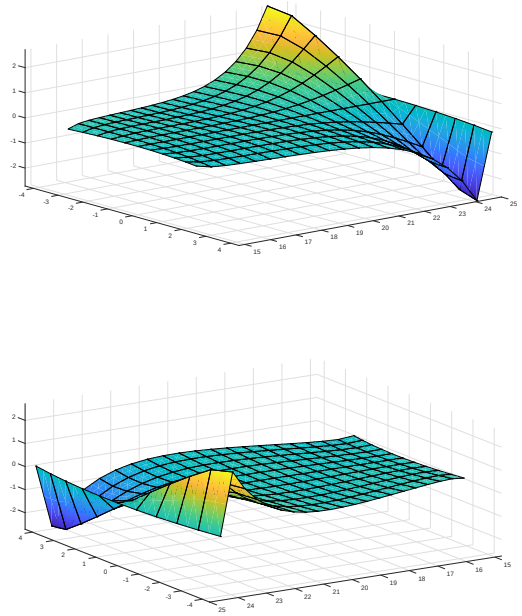


Figure 14: Velocity x

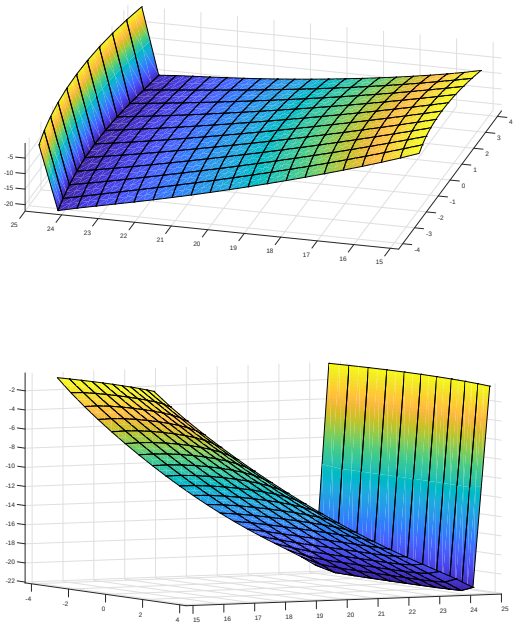


Figure 15: Velocity y

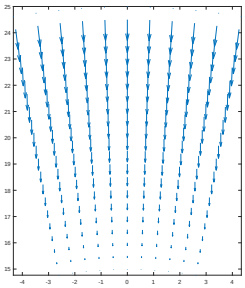


Figure 16: Velocity vector

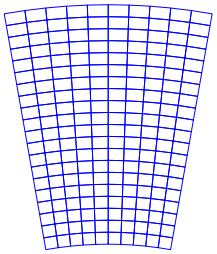


Figure 17: Mesh

```
Nr=20;
Ntheta=10;
[X,T]=CreateMesh(Nr,Ntheta);
h=(1/Nr);
nodesD=find(r>24.99);
valueD = ones(length(nodesD),1);
nfDnodes=length(nodesD);
nfnodes= length(X(:,1));
tnodes= 1:1:nfnodes;
fnodes=setdiff(tnodes,nodesD);
nffnodes=length(fnodes);
theta = 0.5;
D_F = 5 ;
D_G = 15 ;
Sigma_F= 0.25;
sigma_G = 2;
sigma_GF = 0.25;
theta = 0.5;
t = 100;
dt = 0.1;
step = t/dt;
x = X(:,1);
y = X(:,2);
r = sqrt(X(:,1).^2 X(:,2).^2);
u = 1/1500*(r.*x , r.*y) ;
u1 = sqrt(u(:,1).^2+u(:,2).^2);
Pe = (u1*h)/(2*D_F);
c = u * t / h ;
[K M C] = Create_Mat(X,T,N,Nxi,Neta);
F = zeros(nfnode,step+1)
G = zeros(nfnode,step+1)
F(:,1) = (nodesD,1);
G(:,1) = (nodesD,1);
for i = 1 : nfDnodes
AF = M + theta * C * dt + theta * DF * K * dt + theta * sigmaF * M * dt;
BF = -C * dt - DF * K * dt - sigmaF * M * dt;
AFF = AF(fnodes,fnodes);
BFF = BF(fnodes);
BFF = BFF - valueD(i) * AFF(fnodes,nfDnodes(i));
F_tot = AFF/BFF;
AG = M + theta * DG * K * dt + theta * sigmaG * M * dt;
BG = -DG * K * dt -sigmaG * M *dt + theta * sigmaGF * M * dt + (1-theta) * sigm
G_tot = AG / BG ;
end

createMesh1(Nr,Ntheta)
r = linspace(15,25,Nr+1) ;
theta_max = 10*pi/180 ;
theta = linspace(-theta_max,theta_max,Ntheta+1) ;
[R,TH ] = meshgrid(r,theta) ;
```

```
X = R.*sin(TH) ;
Y = R.*cos(TH) ;
Node_number = 1:numel(X) ;
Node_number = reshape(Node_number,size(X)) ;
node_data = zeros(numel(X),4) ;
filename = 'nodes_trial.txt' ;
for i_node = 1:numel(X)
node_data(i_node,:) = [Node_number(i_node) , X(i_node) , Y(i_node), 0] ;
end
ind_ele = 0 ;
for jj = 1:Nr
for ii = 1:Ntheta
ind_ele = ind_ele +1 ;
ele_data(ind_ele,:) = [ind_ele , Node_number(ii+1,jj) , Node_number(ii+1,jj+1),
end
end
XR = node_data(:,2:3) ;
TR = ele_data(:,2:end) ;
end

nx=10;
ny=20;
[Xe,Te]=CreateMesh1(Nr,Ntheta);
elem =0;
plotMesh1(Te,Xe,elem);
[Xp,Tp]=CreateMesh2(Nr,Ntheta);
plotMesh2(Tp,Xp,elem);
elemv=0;
degreev=2;
degreep=1;
elemp=elemv;
[K,G,f] = StokesSystem2(X,T,XP,TP,referenceElement);
[ndp , ndv] = size(G);
[A_Dir , b_Dir , nDf] = BC1(ndv , degreev);
Atot= [ K  A_Dir'  G' ;
A_Dir zeros(nDf , nDf) zeros(nDf , nunk)
G     zeros(nunk , nDf)  zeros(nunk , nunk)];
Btot = [ f    ;
B_Dir ;
zeros(nunk , 1)];
sol = Atot\Btot ;
np = size(Xe,1);
velo = reshape(sol(1:ndv),2,[])';

[K,G,f] = StokesSystem2(X,T,XP,TP,referenceElement)
elem = referenceElement.elemV;
ngaus = referenceElement.ngaus;
wgp = referenceElement.GaussWeights;
N = referenceElement.N;
Nxi = referenceElement.Nxi;
```

```
Neta = referenceElement.Neta;
NP = referenceElement.NP;
ngeom = referenceElement.ngeom;
[nElem,nenV] = size(T);
nenP = size(TP,2);
nPt_V = size(X,1);
if elem == 11
nPt_V = nPt_V + nElem;
end
nPt_P = size(XP,1);
nedofV = 2*nenV;
nedofP = nenP;
ndofV = 2*nPt_V;
ndofP = nPt_P;
K = zeros(ndofV,ndofV);
G = zeros(ndofP,ndofV);
f = zeros(ndofV,1);
for ielem = 1:nElem
Te = T(ielem,:);
TPe = TP(ielem,:);
Xe = X(Te(1:ngeom),:);
Te_dof = reshape([2*Te-1; 2*Te],1,nedofV);
TPe_dof = TPe;
[Ke,Ge,fe] = EleMatStokes2(Xe,ngeom,nedofV,nedofP,ngaus,wgp,N,Nxi,Neta,NP);
K(Te_dof, Te_dof) = K(Te_dof, Te_dof) + Ke;
G(TPe_dof,Te_dof) = G(TPe_dof,Te_dof) + Ge;
f(Te_dof) = f(Te_dof) + fe;
end
[Ke,Ge,fe] = EleMatStokes2(Xe,ngeom,nedofV,nedofP,ngaus,wgp,N,Nxi,Neta,NP)
Ke = zeros(nedofV,nedofV);
Ge = zeros(nedofP,nedofV);
fe = zeros(nedofV,1);
B = zeros(3,nedofV);
for ig = 1:ngaus
N_ig    = N(ig,:);
Nxi_ig  = Nxi(ig,:);
Neta_ig = Neta(ig,:);
NP_ig = NP(ig,:);
Jacob = [
Nxi_ig(1:ngeom)*(Xe(:,1))          Nxi_ig(1:ngeom)*(Xe(:,2))
Neta_ig(1:ngeom)*(Xe(:,1))         Neta_ig(1:ngeom)*(Xe(:,2))
];
dvolu = wgp(ig)*det(Jacob);
res = Jacob\[Nxi_ig;Neta_ig];
nx = res(1,:);
ny = res(2,:);
Ngp = [reshape([1;0]*N_ig,1,nedofV); reshape([0;1]*N_ig,1,nedofV)];
dN = reshape(res,1,nedofV);
C = diag([2,2,1]);
B(1,1:2:end) = nx;
```

```
B(2,2:2:end) = ny;
B(3,1:2:end) = ny; B(3,2:2:end) = nx;
Ke = Ke + B'*C*B*dvolu;
Ge = Ge - NP_ig'*dN*dvolu;
x_ig = N_ig(1:ngeom)*Xe;
f_igaus = SourceTerm(x_ig);
fe = fe + Ngp'*f_igaus*dvolu;
end

Ntheta = 10 ;
Nr=20;
np = 2*Ntheta + 1 ;
nr = 2*Nr+1;
n1=[1:np]';
n2=[(nr-1)*(np)+1 : 1 : (nr+1)*(np)]';
nDB=[n1 ; n2];
nfD = 2*length(ndB);
S = [2*ndB-1   ; 2*ndB ];
A= zeros(nfD,n);
x= X(n1,1);
y= X(n1,2);
theta1= ataN2(y3,x3);
u1 = -0.15.*cos(theta1);
u2 = -0.30.*cos(theta1);
v1 = -0.15.*sin(theta1);
v2 = -0.30.*sin(theta1);
A1 = [ u1   v1 ;
u2   v2 ];

plotMesh(T,X,elem,str,nonum)
if nargin == 3
str1 = 'ko';
str2 = 'r-';
else
if str(1) == ':' | str(1) == '-'
str1 = 'yo';
str2 = ['y' str];
else
str1 = [str(1) 'o'];
str2 = str;
end
end
nen = size(T,2);
if elem == 0
if nen <= 4
order = [1:nen,1];
elseif nen == 9
order = [1,5,2,6,3,7,4,8,1];
end
elseif elem == 1
```

```
if nen <= 3
order = [1:nen,1];
elseif nen == 6
order = [1,4,2,5,3,6,1];
end
elseif elem == 11
order = [1:3,1];
end
plot(X(:,1),X(:,2),str1)
hold on
for j = 1:size(T,1)
plot(X(T(j,order),1),X(T(j,order),2),str2)
end
if nargin==5
if nonum==1
for I=1:size(X,1)
text(X(I,1)+0.02,X(I,2)+0.03,int2str(I),'FontSize',16)
end
end
end

[Xp,Tp] = createMesh2(Nr,Ntheta)
r = linspace(15,25,Nr+1) ;
theta_max = 10*pi/180 ;
theta = linspace(-theta_max,theta_max,Ntheta+1) ;
[R,TH ] = meshgrid(r,theta) ;
X = R.*sin(TH) ;
Y = R.*cos(TH) ;
Node_number = 1:numel(X) ;
Node_number = reshape(Node_number,size(X)) ;
node_data = zeros(numel(X),4) ;
filename = 'nodes_trial.txt' ;
for i_node = 1:numel(X)
node_data(i_node,:) = [Node_number(i_node) , X(i_node) , Y(i_node), 0] ;
end
ind_ele = 0 ;
for jj = 1:Nr
for ii = 1:Ntheta
ind_ele = ind_ele +1 ;
ele_data(ind_ele,:) = [ind_ele , Node_number(ii+1,jj) , Node_number(ii+1,jj+1),
end
end
XR = node_data(:,2:3) ;
TR = ele_data(:,2:end) ;
end

DF = 5;
sigmaF = 0.25;
DG = 15;
sigmaG = 2;
```

```
sigmaFG = 0.5;
nx = 20;
ny = 10;
[X,T,XP,TP] = CreateMeshes(nx,ny);
Xe_ref = referenceElement.Xe_ref;
[Tf,Tu] = boundaryMatrices(X,T,elemV,degreeV,Xe_ref);
[Ku,G,fu] = StokesSystem2(X,T,XP,TP,referenceElement);
[A_DirBC_u, nDir_u, nodesDirBC_u, Dir_u, confined] = BC(nx,ny,ndofV,degree);
b_DirBC_u = Dir_u(:,2);
Ku_tot = Ku + Tu;
Ku_tot(:,Dir_u(:,1)) = 0;
Ku_tot(Dir_u(:,1),Dir_u(:,1)) = eye(nDir_u);
Atot_u = Ku_tot;
step = 120;
t = 2 * pi;
dt = t/step;
theta=0.5;
AG = M + theta*DG*K*dt + theta*sigmaG*M*dt;
BG = -DG * K * dt -sigmaG * M *dt + theta * sigmaGF * M * dt + (1-theta) * sigm
Gtot = AG;
tol = 10^-6;
for n = 1:nstep
veloInc = 1;
FInc = 1;
iter = 0;
g = F(:,n);
while veloInc>=tol
iter = iter + 1
fu = -Tf*g(:,iter);
for i =1:size(Ku,1)
fu_tot(i,1) = fu(i,1)-Ku(i,Dir_u(:,1))*b_DirBC_u;
end
for i = 1:nDir_u
fu_tot(Dir_u(i,1)) = Dir_u(i,2);
end
veloVect(:,iter+1) = Atot_u\fu_tot;
velo1 = veloVect(:,iter+1);
velo = reshape(velo1,2,[])';
Conv = velo;
C = Create_Mat(X,T,N,Nxi,Neta);
AF = M + theta*C*dt + theta*DF*K*dt + theta*sigmaF*M*dt;
BF = M + (1-theta)*(-C*dt - DF*K*dt - sigmaF*M*dt);
Ftot = [A_F Accd_F'; Accd_F zeros(nDir_F,nDir_F)];
g(:,iter+1) =  Lg(1:numnp);
veloInc = norm(veloVect(:,iter+1)-veloVect(:,iter));
end
velocity0(:,n) = veloVect(:,iter+1);
velo1 = velocity0(:,n);
velocity = reshape(velo1,2,[])';
Conv = velocity;
```

```
AF = M + theta*C*dt + theta*DF*K*dt + theta*sigmaF*M*dt;
BF = (-C*dt - DF*K*dt - sigmaF*M*dt);
Atot_F = [A_F Accd_F'; Accd_F zeros(nDir_F,nDir_F)];
[L1,U1] = lu(Atot_F);
btot_F = [B_F*F(:,n)+ f_F; bccd_F];
aux_f =  U1\(L1\btot_F);
aux_F = U1\(L1\btot_F);
F(:,n+1) = F(:,n) + aux_F(1:numnp);
btot = [B_G*G(:,n) - theta*C_G*aux_f(1:numnp) + theta*C_G*F(:,n+1) + f_G];
aux_G  = U2\(L2\btot);
G(:,n+1) = G(:,n) + aux_G(1:numnp);

nen = size(Xe_ref,1);
ngaus1D = 3;
[zgp1D, wgp1D] = Quadrature_1D(ngaus1D);
nDof = size(X,1);
nElem = size(T,1);
nedofV = 2*nen;
nedofF = nen;
ndofV = 2*nDof;
ndofF = nDof;
r = sqrt(X(:,1).^2+X(:,2).^2);
nodesB = find(abs(r-25) < 1e-6);
elemsB = zeros(nElem,3); indB = 1;
for ielem = 1:nElem
Te = T(ielem,:);
Xe = X(Te,:);
aux = intersect(Te,nodesB);
if length(aux) == 2
node1 = find(Te == aux(1));
node2 = find(Te == aux(2));
if Xe(node1,1) > Xe(node2,1)
elemsB(indB,:) = [ielem,node1,node2];
else
elemsB(indB,:) = [ielem,node2,node1];
end
indB = indB+1;
end
end
elemsB = elemsB(1:indB-1,:);
n = length(elemsB*nedofV^2+2);
coef_Tu  = zeros(1,n); indTu_i  = zeros(1,n); indTu_j  = zeros(1,n);
indTu_i(1) = 1;     indTu_j(1) = 1;     coef_Tu(1) = 0;
indTu_i(2) = ndofV; indTu_j(2) = ndofV; coef_Tu(2) = 0;
indTu = 3;
coef_Tf  = zeros(1,n); indTf_i  = zeros(1,n); indTf_j  = zeros(1,n);
indTf_i(1) = 1;     indTf_j(1) = 1;     coef_Tf(1) = 0;
indTf_i(2) = ndofV; indTf_j(2) = ndofF; coef_Tf(2) = 0;
indTf = 3;
for i = 1:size(elemsB,1)
```

```
ielem = elemsB(i,1);
node1 = elemsB(i,2);
node2 = elemsB(i,3);
Te = T(ielem,:);
Xe = X(Te,:);
TeV_dof = reshape([2*Te-1; 2*Te],1,nedofV);
TeF_dof = Te;
[n,wgp,N,Nxi,Neta] = evaluateAtEdge(node1,node2,Xe,Xe_ref,elem,degree,zgp1D,wgp
[Tf_e,Tu_e] = EleMat(Xe,n,nedofV,nedofF,ngaus1D,wgp,N,Nxi,Neta);
for irow = 1:nedofV
for icol = 1:nedofV
indTu_i(indTu) = TeV_dof(irow);
indTu_j(indTu) = TeV_dof(icol);
coef_Tu(indTu) = Tu_e(irow,icol);
indTu = indTu+1;
end
for icol = 1:nedofF
indTf_i(indTf) = TeV_dof(irow);
indTf_j(indTf) = TeF_dof(icol);
coef_Tf(indTf) = Tf_e(irow,icol);
indTf = indTf+1;
end
end
indTu_i  = indTu_i(1:indTu-1);
indTu_j  = indTu_j(1:indTu-1);
coef_Tu = coef_Tu(1:indTu-1);
Tu = sparse(indTu_i,indTu_j,coef_Tu);
indTf_i  = indTf_i(1:indTf-1);
indTf_j  = indTf_j(1:indTf-1);
coef_Tf = coef_Tf(1:indTf-1);
Tf = sparse(indTf_i,indTf_j,coef_Tf);
Tf = -560*Tf;
Tu = -0.5*Tu;
P1 = Xe(node1,:);
P2 = Xe(node2,:);
t = P2 - P1;
h = norm(t);
n = [t(2), -t(1)];
n = n/h;
wgp = wgp1D*h/2;
P1_ref = Xe_ref(node1,:);
P2_ref = Xe_ref(node2,:);
zgp = [
(P2_ref(1) - P1_ref(1))/2*zgp1D + (P2_ref(1) + P1_ref(1))/2,...
(P2_ref(2) - P1_ref(2))/2*zgp1D + (P2_ref(2) + P1_ref(2))/2
];
[N,Nxi,Neta] = ShapeFunc(elem,degree,zgp);
function [Tf_e,Tu_e] = EleMat(Xe,n,nedofV,nedofF,ngaus,wgp,N,Nxi,Neta)
Tf_e = zeros(nedofV,nedofF);
Tu_e = zeros(nedofV,nedofV);
```

```
for ig = 1:ngaus
N_ig     = N(ig,:);
Nxi_ig   = Nxi(ig,:);
Neta_ig  = Neta(ig,:);
Jacob = [
Nxi_ig*(Xe(:,1))          Nxi_ig*(Xe(:,2))
Neta_ig*(Xe(:,1))         Neta_ig*(Xe(:,2))
];
res = Jacob\[Nxi_ig;Neta_ig];
nx = res(1,:);
ny = res(2,:);
Nn1 = n(1)^2*nx + n(1)*n(2)*ny;
Nn2 = n(1)*n(2)*nx + n(2)^2*ny;
Nn = reshape([Nn1;Nn2],1,nedofV);
Tf_e = Tf_e + Nn'*N_ig*wgp(ig);
Naux = [reshape([1;0]*N_ig,1,nedofV); reshape([0;1]*N_ig,1,nedofV)];
Tu_e = Tu_e + Naux'*Naux*wgp(ig);
end
if ngaus == 1
z = 0;
w = 2;
elseif ngaus == 2
pos1 = 1/sqrt(3);
z = [-pos1; pos1];
w = [ 1 1 ];
elseif ngaus == 3
pos1 = sqrt(3/5);
z = [-pos1; 0; pos1];
w = [ 5/9   8/9   5/9 ];
elseif ngaus == 4
pos1 = sqrt(525+70*sqrt(30))/35;
pos2 = sqrt(525-70*sqrt(30))/35;
z = [-pos1; -pos2; pos2; pos1];
w1 = sqrt(30)*(3*sqrt(30)-5)/180;
w2 = sqrt(30)*(3*sqrt(30)+5)/180;
w = [w1   w2   w2   w1];
end
```