Mathematical Modelling in the Natural Sciences SS21
Solutions to Exercises on Sheet 6

Exercises und Lecture Notes

Contents

- **Exercise 1: Static Membrane, Linearized Approach** **1**
  - Task . . . 1
  - Solution . . . 1
- **Exercise 2: Static Membrane, Non-Linear Approach** **4**
  - Task . . . 4
  - Solution . . . 4
- **Exercise 3: Static Membrane, Non-Existence for Non-Linear Approach** **8**
  - Task . . . 8
  - Solution . . . 8
- **Exercise 4: Dynamic Membrane, Non-Linear Approach** **8**
  - Task . . . 8
  - Solution . . . 8

• Exercise 1: Static Membrane, Linearized Approach

◦ Task

Write an approximation $\tilde{J}$ to the potential J (on page 172 of the lecture notes) where the necessary optimality condition on a displacement u which minimizes $\tilde{J}$ is given by the Poisson boundary value problem

$$\begin{cases} -\nabla \cdot (T\nabla u) &= f, \quad \text{in} \quad \Omega \\ u &= 0, \quad \text{on} \quad \partial\Omega \end{cases}$$

Here, it is assumed that the membrane is clamped at the boundary $\partial\Omega$ of $\Omega = (0,1)^2$. For the chosen potential $\tilde{J}$, show by an explicit derivation that the necessary optimality condition is given by the boundary-value problem above. Write a Matlab code to solve the problem for chosen T and f.

◦ Solution

Using the hint on page 174 of the lecture notes,

$$\sqrt{1+|\nabla u|^2} \approx 1 + \tfrac{1}{2}|\nabla u|^2 \quad \text{for} \quad |\nabla u| \ll 1$$

leads to the approximation

$$J(u) = \int_\Omega T\sqrt{1+|\nabla u|^2} - \int_\Omega fu \approx \int_\Omega T + \int_\Omega \left[\tfrac{1}{2}T|\nabla u|^2 - fu\right]$$

which, for a state independent tension $T \neq T(u)$, is minimized by minimizing the second integral. Thus take

$$\tilde{J}(u) = \int_\Omega \left[\tfrac{1}{2}T|\nabla u|^2 - fu\right]$$

whose variational derivative is given by

$$\begin{aligned}\frac{\delta\tilde{J}}{\delta u}(u;v) &= \lim_{\epsilon\to 0}\frac{d}{d\epsilon}\int_\Omega \left[\tfrac{1}{2}T|\nabla(u+\epsilon v)|^2 - f(u+\epsilon v)\right] \\ &= \lim_{\epsilon\to 0}\int_\Omega \left[T\nabla(u+\epsilon v)\cdot\nabla v - fv\right] = \int_\Omega \left[T\nabla u\cdot\nabla v - fv\right] = \\ &= -\int_\Omega v\left[\nabla\cdot[T\nabla u] + f\right] + \int_{\partial\Omega} v\hat{n}\cdot[T\nabla u]\end{aligned}$$

Since $u = 0$ is constrained to hold on $\partial\Omega$, there can be no perturbation of the boundary values, and so $v = 0$ must also hold on $\partial\Omega$. Consequently the boundary integral vanishes and it follows that

$$\frac{\delta\tilde{J}}{\delta u}(u;v) = -\int_\Omega v\left[\nabla\cdot[T\nabla u] + f\right], \quad v\in\mathcal{C}_0^\infty(\Omega).$$

Choosing $\hat{x}\in\Omega^\circ$ and $\epsilon > 0$ arbitrarily but so that $B(\hat{x},\epsilon)\subset\Omega$, set a perturbation

$$v_\epsilon(\boldsymbol{x}) = \frac{1}{\epsilon^2}\phi\left(\frac{\boldsymbol{x}-\hat{\boldsymbol{x}}}{\epsilon}\right), \quad \phi(\boldsymbol{x}) = \begin{cases}\phi_0\exp\left(\dfrac{-1}{|\boldsymbol{x}|^2}\right), & \boldsymbol{x}\in B(0,1)\\ 0, & \text{else,}\end{cases} \qquad \phi_0^{-1} = \int_{B(0,1)}\exp\left(\frac{-1}{|\boldsymbol{x}|^2}\right)d\boldsymbol{x}$$

so that

$$v_\epsilon > 0, \quad B(\hat{\boldsymbol{x}},\epsilon), \quad v_\epsilon = 0 \quad \Omega\backslash B(\hat{x},\epsilon)$$

and

$$\int_\Omega v_\epsilon d\boldsymbol{x} = \phi_0\int_{B(\hat{\boldsymbol{x}},\epsilon)}\exp\left(\frac{-\epsilon^2}{|\boldsymbol{x}-\hat{\boldsymbol{x}}|^2}\right)\frac{d\boldsymbol{x}}{\epsilon^2} \overset{\epsilon\boldsymbol{y}=\boldsymbol{x}-\hat{\boldsymbol{x}}}{=} \phi_0\int_{B(0,1)}\exp\left(\frac{-1}{|\boldsymbol{y}|^2}\right)d\boldsymbol{y} = 1.$$

Then by the mean value theorem for integrals there is an $\hat{\boldsymbol{x}}_\epsilon\in B(\hat{\boldsymbol{x}},\epsilon)$ such that

$$\frac{\delta\tilde{J}}{\delta u}(u;v_\epsilon) = -\int_\Omega v_\epsilon\left[\nabla\cdot[T\nabla u] + f\right] = -\left[\nabla\cdot[T\nabla u] + f\right](\hat{\boldsymbol{x}}_\epsilon)\underbrace{\int_\Omega v_\epsilon}_{=1} \overset{\epsilon\to 0}{\longrightarrow} -\left[\nabla\cdot[T\nabla u] + f\right](\hat{\boldsymbol{x}}).$$

Requiring all such variational derivatives to vanish independently of $\hat{\boldsymbol{x}}\in\Omega$ leads to the boundary value problem

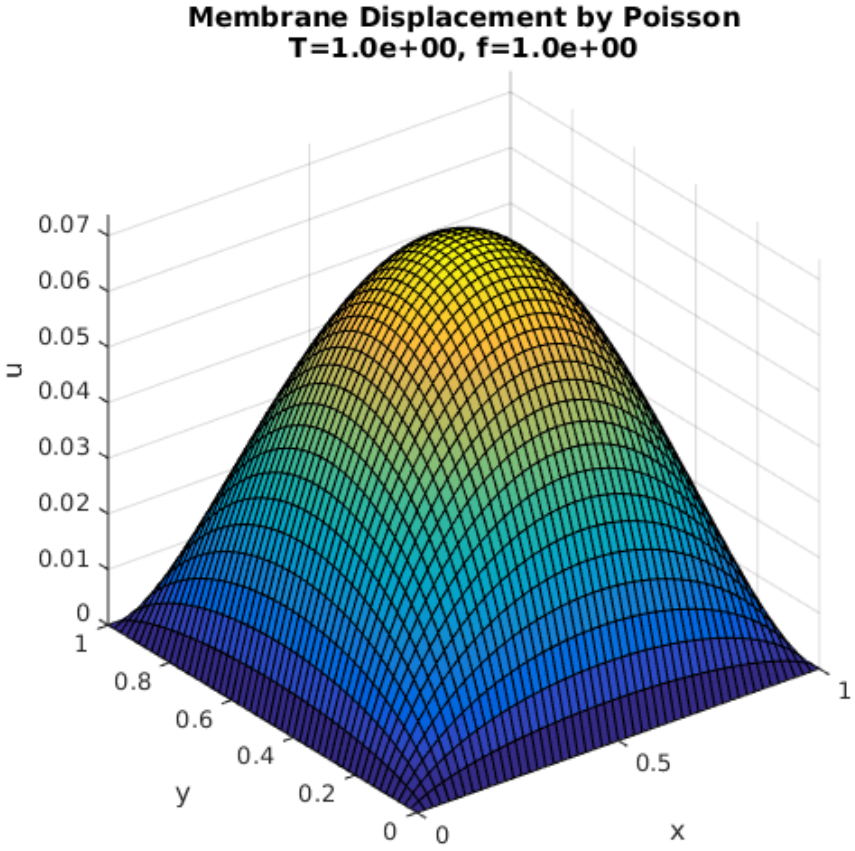
$$\begin{cases} -\nabla\cdot[T\nabla u] = f, & \Omega \\ u = 0, & \partial\Omega\end{cases}$$

The following Matlab code is used to solve the boundary-value problem

```
% set up the figure
    h1 = figure(1); close(h1);
    h1 = figure(1); set(h1,'Position',[10 50 500 500]);

% discretization parameters
    Nx  = 50;
    Ny  = 50;
    Nx1    = Nx+1;
    Ny1    = Ny+1;
    NxNy1  = Nx*Ny1;
    NyNx1  = Nx1*Ny;
    Nx1Ny1 = Nx1*Ny1;
```

```
    xmin = 0; xmax = 1;
    hx = (xmax-xmin)/Nx;
    x  = linspace(xmin,xmax,Nx1); xx = kron(x',ones(1,Ny1));
    ymin = 0; ymax = 1;
    hy = (ymax-ymin)/Ny;
    y  = linspace(ymin,ymax,Ny1); yy = kron(ones(Nx1,1),y);

% first partial derivative operators
    dx = spdiags(ones(Nx,1),1,Nx,Nx1) ...
       - spdiags(ones(Nx,1),0,Nx,Nx1);
    dx = dx/hx;
    iy = speye(Ny1);
    Dx = kron(iy,dx);
    dy = spdiags(ones(Ny,1),1,Ny,Ny1) ...
       - spdiags(ones(Ny,1),0,Ny,Ny1);
    dy = dy/hy;
    ix = speye(Nx1);
    Dy = kron(dy,ix);

% identity and boundary indicator
    I = speye(Nx1Ny1);
    b = ones(Nx1,Ny1);
    b(1,:)  = 0; b(Nx1,:) = 0; b(:,1)  = 0; b(:,Ny1) = 0; b = b(:);
    B = spdiags(b,0,Nx1Ny1,Nx1Ny1);

% tension and load force
    T0 = 1;
    f0 = 1;
    Tx = T0*ones(Nx,Ny1);  Tx = Tx(:);
    Ty = T0*ones(Nx1,Ny);  Ty = Ty(:);
    f  = f0*ones(Nx1,Ny1); f  = f(:).*b;

% differential operator
    L  = Dx'*spdiags(Tx,0,NxNy1,NxNy1)*Dx ...
       + Dy'*spdiags(Ty,0,NyNx1,NyNx1)*Dy;
    L  = B*L + (I-B);

% solution
    u = L\f;
    umin = min(u);
    umax = max(u);
    u = reshape(u,Nx1,Ny1);

% graphical representation
    surf(xx,yy,u)
    axis([xmin xmax ymin ymax umin umax])
    title(sprintf('Membrane Displacement by Poisson\nT=%0.1e, f=%0.1e',T0,f0));
    xlabel('x'); ylabel('y'); zlabel('u')
```

The result is shown graphically as follows.



• Exercise 2: Static Membrane, Non-Linear Approach

◦ Task

For $\Omega = (0,1)^2$ and given tension T and load f write a Matlab code to solve the non-linear boundary value problem

$$\begin{cases} -\nabla \cdot \left(\dfrac{T \nabla u}{\sqrt{1 + |\nabla u|^2}} \right) &= f, \quad \text{in} \quad \Omega \\ u &= 0, \quad \text{on} \quad \partial\Omega \end{cases}$$

using a Picard iteration in order to compute the membrane displacement u. Investigate the case that T and f are constant and f/T is ever larger.

◦ Solution

The following Matlab code is used to solve the boundary-value problem

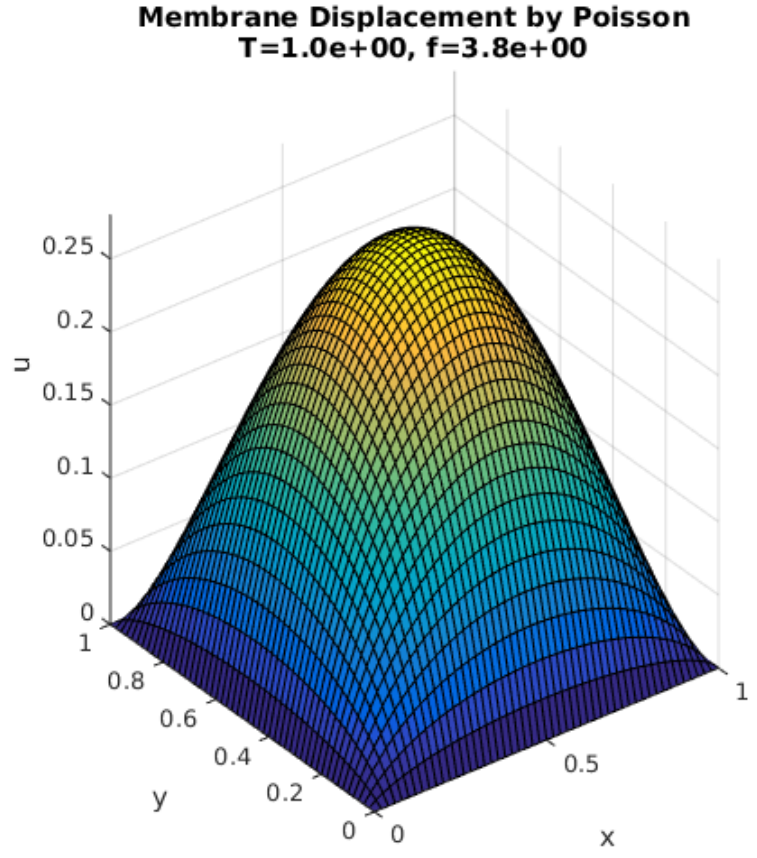
```
% set up the figure
    h1 = figure(1); close(h1);
    h1 = figure(1); set(h1,'Position',[10 50 1000 500]);
```

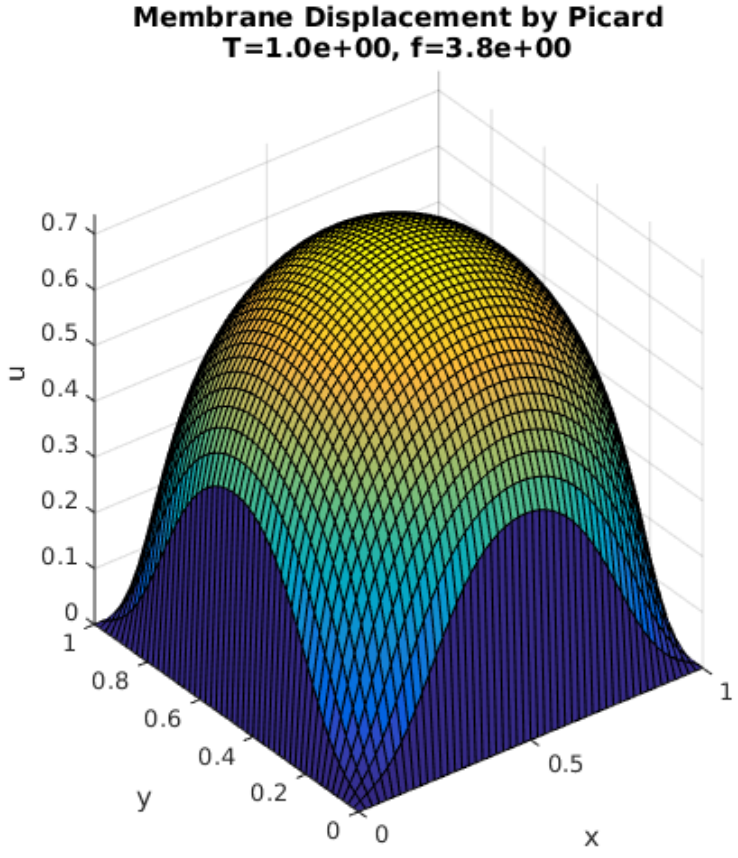
```
% discretization parameters
    Nx  = 50;
    Ny  = 50;
    xmin = 0; xmax = 1;
    hx = (xmax - xmin)/Nx;
    x  = linspace(xmin,xmax,Nx+1); xx = kron(x',ones(1,Ny+1));
    ymin = 0; ymax = 1;
    hy = (ymax - ymin)/Ny;
    y  = linspace(ymin,ymax,Ny+1); yy = kron(ones(Nx+1,1),y);

% termination criterion
    kmax = 1000;
    tol  = 1.0e-6;

% first partial derivative operators
    dx = spdiags(ones(Nx,1),1,Nx,Nx+1) ...
       - spdiags(ones(Nx,1),0,Nx,Nx+1);
    dx = dx/hx;
    iy = speye(Ny+1);
    Dx = kron(iy,dx);
    dy = spdiags(ones(Ny,1),1,Ny,Ny+1) ...
       - spdiags(ones(Ny,1),0,Ny,Ny+1);
    dy = dy/hy;
    ix = speye(Nx+1);
    Dy = kron(dy,ix);

% averaging from cells to interfaces
    ax  = spdiags(ones(Nx,1),1,Nx,Nx+1) ...
        + spdiags(ones(Nx,1),0,Nx,Nx+1);
    ax  = ax/2;
    axT = ax'; axT(1,1) = 1; axT(Nx+1,Nx) = 1;
    iy  = speye(Ny+1);
    Ax  = kron(iy,ax);
    AxT = kron(iy,axT);

    ay  = spdiags(ones(Ny,1),1,Ny,Ny+1) ...
        + spdiags(ones(Ny,1),0,Ny,Ny+1);
    ay  = ay/2;
    ayT = ay'; ayT(1,1) = 1; ayT(Ny+1,Ny) = 1;
    ix  = speye(Nx+1);
    Ay  = kron(ay,ix);
    AyT = kron(ayT,ix);

% identity and boundary indicator
    I = speye((Nx+1)*(Ny+1));
    b = ones(Nx+1,Ny+1);
    b(1,:)  = 0; b(Nx+1,:) = 0; b(:,1)  = 0; b(:,Ny+1) = 0; b = b(:);
```

```
    B = spdiags(b,0,(Nx+1)*(Ny+1),(Nx+1)*(Ny+1));

% tension and load force
    T0 = 1;
    f0 = 3.8;
    Tx = T0*ones(Nx,Ny+1);   Tx = Tx(:);
    Ty = T0*ones(Nx+1,Ny);   Ty = Ty(:);
    f  = f0*ones(Nx+1,Ny+1); f  = f(:).*b;

% differential operator
    L  = Dx'*spdiags(Tx,0,Nx*(Ny+1),Nx*(Ny+1))*Dx ...
       + Dy'*spdiags(Ty,0,(Nx+1)*Ny,(Nx+1)*Ny)*Dy;
    L  = B*L + (I-B);

% initial solution
    u = L\f;

% graph the initial solution
    subplot(1,2,1)
    surf(xx,yy,reshape(u,Nx+1,Ny+1))
    umin = min(u); umax = max(umin+eps,max(u));
    axis([xmin xmax ymin ymax umin umax])
    xlabel('x'); ylabel('y'); zlabel('u')
    title(sprintf('Membrane Displacement by Poisson\nT=%0.1e, f=%0.1e',T0,f0));

% Picard iterations
    for k=1:kmax

        us = u;

        ux = Dx*u;
        uy = Dy*u;

        cx = Tx./sqrt(1 + ux.^2 + (Ax*(AyT*uy)).^2);
        cy = Ty./sqrt(1 + (Ay*(AxT*ux)).^2 + uy.^2);

        L  = Dx'*spdiags(cx,0,Nx*(Ny+1),Nx*(Ny+1))*Dx ...
           + Dy'*spdiags(cy,0,(Nx+1)*Ny,(Nx+1)*Ny)*Dy;
        L  = B*L + (I-B);

        u  = L\f; u = u.*b;

% graph the current solution
        subplot(1,2,2)
        surf(xx,yy,reshape(u,Nx+1,Ny+1))
        umin = min(u); umax = max(umin+eps,max(u));
        axis([xmin xmax ymin ymax umin umax])
        xlabel('x'); ylabel('y'); zlabel('u')
```

```
        title(sprintf('Membrane Displacement by Picard\nk=%0.1e',k));
        drawnow;

        if (norm(u-us) <= tol*norm(u))
            break;
        end
    end

% graph the final solution
    subplot(1,2,2)
    umin = min(u); umax = max(u);
    surf(xx,yy,reshape(u,Nx+1,Ny+1))
    axis([xmin xmax ymin ymax umin umax])
    title(sprintf('Membrane Displacement by Picard\nT=%0.1e, f=%0.1e',T0,f0));
    xlabel('x'); ylabel('y'); zlabel('u')
```

The result for f/T critically large is shown graphically as follows.





• Exercise 3: Static Membrane, Non-Existence for Non-Linear Approach

◦ Task

For a constant tension T and load f show that if f/T is sufficiently large no solution exists to the following boundary-value problem.

$$\begin{cases} -\nabla \cdot \left(\dfrac{T\nabla u}{\sqrt{1+|\nabla u|^2}} \right) &= f, \quad \text{in} \quad \Omega \\ u &= 0, \quad \text{on} \quad \partial\Omega \end{cases}$$

◦ Solution

For $\Omega = B(0,1)$ the solution in 1D can be computed straightforwardly and it is given by

$$u(x) = \frac{\sqrt{T^2 - f^2x^2} - \sqrt{T^2 - f^2}}{f}.$$

This form gives the solution in 2D on $\Omega = B(0,1)$ as

$$u(x,y) = \frac{\sqrt{T^2 - f^2(x^2+y^2)} - \sqrt{T^2 - f^2}}{f}.$$

In neither case is the solution real for $f > T$. In general, supposing that u is a solution for $f/T > |\partial\Omega|/\Omega$ leads to the following contradiction.

$$\frac{f}{T}|\Omega| = \int_\Omega \frac{f}{T} = \left| \int_\Omega -\nabla \cdot \left(\frac{\nabla u}{\sqrt{1+|\nabla u|^2}} \right) \right| = \left| \int_{\partial\Omega} \left(\frac{\hat{n} \cdot \nabla u}{\sqrt{1+|\nabla u|^2}} \right) \right| \leq |\partial\Omega|.$$

• Exercise 4: Dynamic Membrane, Non-Linear Approach

◦ Task

For $\Omega = (0,1)^2$ and given tension T, load force f and damping c write a Matlab code to solve the initial and boundary value problem for the nonlinear wave equation

$$\begin{cases} u_{tt} + cu_t &= \nabla \cdot \left(\dfrac{T\nabla u}{\sqrt{1+|\nabla u|^2}} \right) + f, \quad \text{in} \quad \Omega \times (0,T] \\ u &= 0, & \text{on} \quad \partial\Omega \times [0,T] \\ u &= u_0, & \text{on} \quad \Omega \times \{0\} \\ u_t &= u_1, & \text{on} \quad \Omega \times \{0\} \end{cases}$$

◦ Solution

The following Matlab code is used to solve the boundary-value problem

```
% set up the figure
    h1 = figure(1); close(h1);
    h1 = figure(1); set(h1,'Position',[10 50 1000 500]);
```

```
% discretization parameters
    Nx  = 25;
    Ny  = 25;
    xmin = 0; xmax = 1;
    hx = (xmax - xmin)/Nx;
    x  = linspace(xmin,xmax,Nx+1); xx = kron(x',ones(1,Ny+1));
    ymin = 0; ymax = 1;
    hy = (ymax - ymin)/Ny;
    y  = linspace(ymin,ymax,Ny+1); yy = kron(ones(Nx+1,1),y);
    T  = 3;
    Nt = 100*T;
    dt = T/Nt;
    th = 0.5;

% termination criterion
    tol  = 1.0e-6;

% first partial derivative operators
    dx = spdiags(ones(Nx,1),1,Nx,Nx+1) ...
       - spdiags(ones(Nx,1),0,Nx,Nx+1);
    dx = dx/hx;
    iy = speye(Ny+1);
    Dx = kron(iy,dx);
    dy = spdiags(ones(Ny,1),1,Ny,Ny+1) ...
       - spdiags(ones(Ny,1),0,Ny,Ny+1);
    dy = dy/hy;
    ix = speye(Nx+1);
    Dy = kron(dy,ix);

% averaging from cells to interfaces
    ax  = spdiags(ones(Nx,1),1,Nx,Nx+1) ...
        + spdiags(ones(Nx,1),0,Nx,Nx+1);
    ax  = ax/2;
    axT = ax'; axT(1,1) = 1; axT(Nx+1,Nx) = 1;
    iy  = speye(Ny+1);
    Ax  = kron(iy,ax);
    AxT = kron(iy,axT);

    ay  = spdiags(ones(Ny,1),1,Ny,Ny+1) ...
        + spdiags(ones(Ny,1),0,Ny,Ny+1);
    ay  = ay/2;
    ayT = ay'; ayT(1,1) = 1; ayT(Ny+1,Ny) = 1;
    ix  = speye(Nx+1);
    Ay  = kron(ay,ix);
    AyT = kron(ayT,ix);

% zero, identity and boundary indicator
```

```
    I = speye((Nx+1)*(Ny+1));
    Z = sparse((Nx+1)*(Ny+1),(Nx+1)*(Ny+1));
    J = [I,Z;Z,I];
    b = ones(Nx+1,Ny+1);
    b(1,:)  = 0; b(Nx+1,:) = 0; b(:,1)  = 0; b(:,Ny+1) = 0; b = b(:);
    B = spdiags(b,0,(Nx+1)*(Ny+1),(Nx+1)*(Ny+1));

% tension, load force and damping
    T0 = 1;
    f0 = 1;
    C0 = 1;
    Tx = T0*ones(Nx,Ny+1);   Tx = Tx(:);
    Ty = T0*ones(Nx+1,Ny);   Ty = Ty(:);
    f  = f0*ones(Nx+1,Ny+1); f  = f(:).*b; F = [zeros((Nx+1)*(Ny+1),1);f];
    C  = C0*B;

% initial conditions
    u  = zeros(Nx+1,Ny+1); uv = u(:,round(Ny/2)); u  = u(:);
    ut = zeros(Nx+1,Ny+1); ut = ut(:);
    U  = [u;ut];
    t  = 0;
    tt = 0*ones(Nx+1,1);

% graph the initial solution
    subplot(1,2,1)
    surf(xx,yy,reshape(u,Nx+1,Ny+1))
    umin = min(u); umax = max(umin+eps,max(u));
    axis([xmin xmax ymin ymax umin umax])
    xlabel('x'); ylabel('y'); zlabel('u')
    title(sprintf('Current Membrane Displacement\nt=%0.1e',t));

% time stepping
    for k=1:Nt

        Us = U;
        t  = k*dt;
        tt = [tt,t*ones(Nx+1,1)];

        ux = Dx*u;
        uy = Dy*u;

        cx = Tx./sqrt(1 + ux.^2 + (Ax*(AyT*uy)).^2);
        cy = Ty./sqrt(1 + (Ay*(AxT*ux)).^2 + uy.^2);

        L  = Dx'*spdiags(cx,0,Nx*(Ny+1),Nx*(Ny+1))*Dx ...
           + Dy'*spdiags(cy,0,(Nx+1)*Ny,(Nx+1)*Ny)*Dy;
        L  = B*L;
```

```
        A  = [Z,I;-L,-C];

        U  = (J - th*dt*A) \ ((J + (1-th)*dt*A)*U + dt*F);

        U(1:(Nx+1)*(Ny+1)) = ...
        U(1:(Nx+1)*(Ny+1)).*b;
        U(1+(Nx+1)*(Ny+1):2*(Nx+1)*(Ny+1)) = ...
        U(1+(Nx+1)*(Ny+1):2*(Nx+1)*(Ny+1)).*b;

        u  = reshape(U(1:(Nx+1)*(Ny+1)),Nx+1,Ny+1);
        uv = [uv,u(:,round(Ny/2))];
        u  = u(:);

% graph the current solution
        subplot(1,2,1)
        surf(xx,yy,reshape(u,Nx+1,Ny+1))
        umin = min(u); umax = max(umin+eps,max(u));
        axis([xmin xmax ymin ymax umin umax])
        xlabel('x'); ylabel('y'); zlabel('u')
        title(sprintf('Current Membrane Displacement\nt=%0.1e',t));

        subplot(1,2,2)
        surf(kron(x',ones(1,k+1)),tt,uv)
        umin = min(uv(:)); umax = max(umin+eps,max(uv(:)));
        axis([xmin xmax 0 t umin umax])
        xlabel('x'); ylabel('t'); zlabel('u')
        title(sprintf(['Dynamic Membrane Displacement Profile\n', ...
            'T=%0.1e, f=%0.1e, c=%0.1e'],T0,f0,C0));

        drawnow;

        if (norm(U-Us) <= tol*norm(U))
            break;
        end
    end

% graph the final solution
    subplot(1,2,1)
    surf(xx,yy,reshape(u,Nx+1,Ny+1))
    umin = min(u); umax = max(u);
    axis([xmin xmax ymin ymax umin umax])
    xlabel('x'); ylabel('y'); zlabel('u')
    title(sprintf('Current Membrane Displacement\nt=%0.1e',t));
```

The result is shown graphically as follows.