Finite Difference Solution to the Diffusion Equation in Spherical and Cylindrical Co-ordinates

Mat Hunt

1 Spherical Diffusion

1.1 Derivation of Method

The equation of interest is:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial r^2} + \frac{2D}{r}\frac{\partial u}{\partial r} \tag{1}$$

With boundary conditions:

$$\frac{\partial u}{\partial r}\Big|_{r=0} = 0, \quad \lambda\frac{\partial u}{\partial r}\Big|_{r=R} = q \tag{2}$$

Now evaluate this equation at $r = r_j$:

$$\left.\frac{\partial u}{\partial t}\right|_{r=r_j} = D\left.\frac{\partial^2 u}{\partial r^2}\right|_{r=r_j} + \left.\frac{2D}{r}\frac{\partial u}{\partial r}\right|_{r=r_j} \tag{3}$$

We use the following approximations:

$$\left.\frac{\partial u}{\partial t}\right|_{r=r_j} \approx \frac{u_{i+1,j} - u_{i,j}}{\delta t}, \quad \left.\frac{\partial^2 u}{\partial r^2}\right|_{r=r_j} \approx \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\delta r^2}, \quad \left.\frac{2}{r}\frac{\partial u}{\partial r}\right|_{r=r_j} \approx \frac{u_{i,j+1} - u_{i,j-1}}{r_j \delta r} \tag{4}$$

The Crank-Nicholson method is used and this means that evaluating at $t = t_{i+\frac{1}{2}}$ and the numerical scheme is:

$$\begin{aligned}\frac{u_{i+1,j} - u_i, j}{\delta t} &= \frac{D}{2}\left[\frac{u_{i+1,j+1} - 2u_{i+1,j} + u_{i+1,j-1}}{\delta r^2} + \frac{u_{i+1,j+1} - u_{i+1,j-1}}{r_j \delta r} + \right. \\ &\left. + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\delta r^2} + \frac{u_{i,j+1} - u_{i,j-1}}{r_j \delta r}\right]\end{aligned}$$

Write:

$$\alpha = \frac{D\delta t}{\delta r^2}, \quad \beta = \frac{D\delta t}{2\delta r} \tag{5}$$

The equation becomes:

$$\begin{aligned} u_{i+1,j} - u_{i,j} &= \frac{\alpha}{2}u_{i+1,j+1} - \alpha u_{i+1,j} + \frac{\alpha}{2}u_{i+1,j-1} + \frac{\beta}{r_j}u_{i+1,j+1} - \frac{\beta}{r_j} + \\ &+ \frac{\alpha}{2}u_{i,j+1} - \alpha u_{i,j} + \frac{\alpha}{2}u_{i,j-1} + \frac{\beta}{r_j}u_{i,j+1} - \frac{\beta}{r_j}u_{i,j-1} \end{aligned}$$

This can be rearranged to obtain:

$$\left(\frac{\alpha}{2} + \frac{\beta}{r_j}\right)u_{i+1,j+1} - (1+\alpha)u_{i+1,j} + \left(\frac{\alpha}{2} - \frac{\beta}{r_j}\right)u_{i+1,j-1} = -\left(\frac{\alpha}{2} + \frac{\beta}{r_j}\right)u_{i,j+1} + \\ + (\alpha - 1)u_{i,j} - \left(\frac{\alpha}{2} - \frac{\beta}{r_j}\right)u_{i,j-1} \quad (6)$$

For spherical equations, it is usual to include a symmetry condition:

$$\left.\frac{\partial u}{\partial r}\right|_{r=0} = 0 \quad (7)$$

The discrete version of this is:

$$\frac{u_{i,1} - u_{i,-1}}{2\delta r} = 0 \Rightarrow u_{i,-1} = u_{i,1} \quad (8)$$

Using L'Hopital's rule for $r = 0$ on the derivative shows that:

$$\lim_{r\to 0}\frac{1}{r}\frac{\partial u}{\partial r} = \frac{\partial^2 u}{\partial r^2}$$

Which makes the equation:

$$\frac{\partial u}{\partial t} = 3D\frac{\partial^2 u}{\partial r^2} \quad (9)$$

Doing the Crank-Nicholson method again, we obtain:

$$u_{i+1,j} - u_{i,j} = \frac{3\alpha}{2}u_{i+1,j+1} - 3\alpha u_{i,j} + \frac{3\alpha}{2}u_{i,j-1} + \\ + \frac{3\alpha}{2}u_{i+1,j+1} - 3\alpha u_{i+1,j} + \frac{3\alpha}{2}u_{i,j-1}$$

Setting $j = 0$ which results in:

$$3\alpha u_{i+1,1} + (1 - 3\alpha)u_{i+1,0} = -3\alpha u_{i,1} - (1 + 3\alpha)u_{i,0} \quad (10)$$

The next boundary condition to examine is the one ar $r = R$, this is given by:

$$\lambda\left.\frac{\partial u}{\partial r}\right|_{r=R} = q \quad (11)$$

where $q = q(t)$. The discrete version of this is:

$$u_{i,N+1} = u_{i,N-1} + \frac{2q_i\delta r}{\lambda} \tag{12}$$

Set $j = N$ and use eq (5) to obtain:

$$\left(\frac{\alpha}{2} + \frac{\beta}{r_N}\right)\left[u_{i+1,N-1} + \frac{2q_{i+1}\delta r}{\lambda}\right] - (1+\alpha)u_{i+1,N} + \left(\frac{\alpha}{2} - \frac{\beta}{r_N}\right)u_{i+1,N-1} =$$
$$-\left(\frac{\alpha}{2} + \frac{\beta}{r_N}\right)\left[u_{i,N-1} + \frac{2q_i\delta r}{\lambda}\right] + +(\alpha-1)u_{i,j} - \left(\frac{\alpha}{2} - \frac{\beta}{r_j}\right)u_{i,j-1}$$

This is rearranged to the following:

$$\alpha u_{i+1,N-1} - (1+\alpha)u_{i+1,N} = -\alpha u_{i,N-1} + (\alpha-1)u_{i,N} - \\ - \frac{2\delta r}{\lambda}\left(\frac{\alpha}{2} + \frac{\beta}{r_N}\right)(q_i + q_{i+1}) \tag{13}$$

We can derive another property which we can check to see the code working. Write the equation as:

$$\frac{\partial u}{\partial t} = \frac{D}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial u}{\partial r}\right) \tag{14}$$

Multiplying through by r^2 and integrate from 0 to R to obtain:

$$\begin{aligned}
\int_0^R r^2\frac{\partial u}{\partial t}dr &= D\int_0^R \frac{\partial}{\partial r}\left(r^2\frac{\partial u}{\partial r}\right)dr \\
\frac{d}{dt}\int_0^R r^2 u(t,r)dr &= D\left[r^2\frac{\partial u}{\partial r}\right]_0^R \\
\frac{du_{\text{av}}}{dt} &= DR^2\left.\frac{\partial u}{\partial r}\right|_{r=R} \\
&= \frac{DR^2 q(t)}{\lambda}
\end{aligned}$$

Showing:

$$\frac{du_{\text{av}}}{dt} = \frac{DR^2 q(t)}{\lambda} \tag{15}$$

The code has to be able to preserve this property.

1.2 Numerical Implementation of Scheme

To begin with, we set out limits for time and distance as well as the initial and boundary conditions

```
n=100;m=100;
r=linspace(0,1,n);
t=linspace(0,2,m);
dr=r(2);
dt=t(2);
u_0=r.*(1-r);
q=exp(-0.1*t);
```

Then we can compute the values of α and β:

```
D=1e-4;
lambda=1;
alpha=D*dt/dr^2;
beta=D*dt/(2*dr);
```

The main equation for the numerics can be written as a matrix equation:

$$A[\mathbf{u}]^{k+1} = B[\mathbf{u}]^k + \mathbf{c} \tag{16}$$

where A and B are $n \times n$ matrices. So it's a matter of constructing the matrices A and B. Note that the matrices are tri-diagonal so they're easy to populate. Define the following:

```
A_main=-(1+alpha)*ones(n-1,1);
A_u=0.5*alpha+beta./r(1:n-1);
A_l=0.5*alpha-beta./r(2:n)
```

Note this will yield an infinite value for the first value in $\texttt{A}_u$ but this is going to be overwritten, so it won't be a problem. The length of $\texttt{A}_{main}$ is $n-1$ and the lengths of $\texttt{A}_u$ and $\texttt{A}_l$ are $n-2$. To make a tri-diagonal matrix one uses:

```
A=diag(A_u,1)+diag(A_l,-1)+diag(A_main,0);
```

The matrix B is computed similarly:

```
B_main=(alpha-1)*ones(n,1);
B=diag(-A_u,1)+diag(B_main,0)+diag(-A_l,-1);
```

The next thing to do is add in the boundary conditions. Using equation (10) shows that:

$$A_{11} = -(1+3\alpha), \quad A_{12} = 3\alpha, \quad B_{11} = 3\alpha - 1, \quad B_{12} = -3\alpha \tag{17}$$

This is implemented in code as:

```
A(1,1)=1-3*alpha;
A(1,2)=3*alpha;;
B(1,1)=-(1+3*alpha);
B(1,2)=-3*alpha;
```

The outer boundary condition, equation (13) has the following values for the matrices A and B:

$$A_{nn} = -(1+\alpha), \quad A_{n(n-1)} = \alpha. \quad B_{nn} = \alpha - 1, \quad B_{n(n-1)} = -\alpha \tag{18}$$

This is implemented as:

```
A(n,n)=-(1+alpha);
A(n,n-1)=alpha;
B(n,n)=-alpha;
B(n,n-1)=alpha-1;
```

The vector $\mathbf{c}$ can only be constructed at each time step, however it will make the code neater if we define:

$$\gamma = -\frac{2\delta r}{\lambda}\left(\frac{\alpha}{2} + \frac{\beta}{r_N}\right) \tag{19}$$

which does stay the same at each time step, for computational speed, the column vector p is defined as a matrix precursor for the matrix $\mathbf{c}$

```
gamma=-2*(dr/lambda)*(0.5*alpha+beta/r(n));
p=zeros(n,1);
p(n)=1;
```

The next step before the iteration is to define the initial condition and the solution matrix and insert the initial condition into the solution matrix:

```
u=zeros(m,n);
u(1,:)=u_0;
```

The final stage is to do the iteration:

```
for␣i=2:m
c=gamma*p*(q(i-1)+q(i));
rhs=B*u(i-1,:)'+c;
sol=linsolve(A,rhs);
u(i,:)=sol';
end
```

This gives a complete numerical solution for the spherical case. It's possible for the computational time to be reduced even further. Multiplying equation (16) through by A^{-1} yielding:

$$[\mathbf{u}]^{k+1} = A^{-1}B[\mathbf{u}]^k + A^{-1}\mathbf{c} \tag{20}$$

The matrix $A^{-1}B$ can be computed prior to the loop and $A^{-1}\mathbf{c}$ can be computed as:

$$A^{-1}\mathbf{c} = A^{-1}(f(t)(0,0,\ldots,0,1)^\top) = f(t)A^{-1}(0,0,\ldots,0,1)^\top$$

This leads to a factor of 2 speed increase. For a more effective implementation, it is useful to write the numerical solution as a function which calls $u_0(r)$, $q(t)$, D and λ as arguments and returns $u(t,r)$. The code for such a function is given as:

```
function␣u=spherical_fd(u_0,q,D,lambda,r,t)
n=length(r);m=length(t);
u=zeros(m,n);
c=zeros(n,1);
dr=r(2);
dt=t(2);
alpha=D*dt/dr^2;
beta=D*dt/(2*dr);
A_main=-(1+alpha)*ones(n,1);
A_u=0.5*alpha+beta./r(1:n-1);
A_l=0.5*alpha-beta./r(2:n);
A=diag(A_u,1)+diag(A_l,-1)+diag(A_main,0);
B_main=(alpha-1)*ones(n,1);
B=diag(-A_u,1)+diag(B_main,0)+diag(-A_l,-1);
A(1,1)=-(1+3*alpha);
A(1,2)=3*alpha;
B(1,1)=3*alpha-1;
B(1,2)=-3*alpha;
A(n,n)=-(1+alpha);
A(n,n-1)=alpha;
B(n,n)=-alpha;
B(n,n-1)=alpha-1;
p=zeros(n,1);
p(n)=1;
B_1=A\B;
w=A\p;
gamma=-2*(dr/lambda)*(0.5*alpha+beta/r(n));
u(1,:)=u_0;
for␣i=2:m
c=gamma*w*(q(i-1)+q(i));
sol=B_1*u(i-1,:)'+c;
u(i,:)=sol';
end
```

2 Alternative Method for Spherical Diffusion

There is another method for the dealing with the case of spherical diffusion. Let $u = v/r$, and this results in the equation:

$$\frac{\partial v}{\partial t} = D\frac{\partial^2 v}{\partial r^2} \tag{21}$$

To investigate the boundary conditions, note that:

$$\frac{\partial u}{\partial r} = \frac{1}{r}\frac{\partial v}{\partial r} - \frac{v}{r^2} \Rightarrow r^2\frac{\partial u}{\partial r} = r\frac{\partial v}{\partial r} - v$$

So the boundary conditions become the following:

$$v|_{r=0} = 0, \quad \frac{R^2 q(t)}{\lambda} = R\frac{\partial v}{\partial r}\bigg|_{r=R} - v|_{r=R} \tag{22}$$

The next stage is to write the numerical algorithm. The main algorithm is the same as the standard one which is:

$$\alpha v_{i+1,j+1} - (1+2\alpha)v_{i+1,j} + \alpha v_{i+1,j-1} = -\alpha v_{i,j+1} + (2\alpha - 1)v_{i,j} - \alpha v_{i,j-1} \tag{23}$$

Here $\alpha = D\delta t/\delta r^2$. Using the boundary condition at $r = R$, the outer value can be written as:

$$v_{i,N+1} = v_{i,N-1} + 2\delta r\left(\frac{Rq_i}{\lambda} + \frac{v_{i,N}}{R}\right) \tag{24}$$

This makes the boundary condition to be:

$$2\alpha v_{i+1,N-1} + \left(\frac{2\alpha\delta r}{R} - 1 - 2\alpha\right)v_{i+1,N} = -2\alpha v_{i,N-1} + \\ + \left(2\alpha - 1 - \frac{2\alpha\delta r}{R}\right) - \frac{2\alpha R\delta r}{\lambda}(q_{i+1} + q_i) \tag{25}$$

To numerically solve this equation we note that the matrix equation is symmetric. AS previous we define the time and position variables:

```
N=200,M=200;
t=linspace(0,10,N);
r=linspace(0,1,M)
r_prime=r;
r_prime(1)=1e-3;
dt=t(2);dr=r(2);
u_0=ones(1,M);
v_0=u_0(2:M)./r(2:M);
q=0.2*ones(N,1);
D=1e-4;
alpha=D*dt/dr^2;
```

Set the matrices up for solution:

```
A_main=-(1+2*alpha)*ones(M,1);
A_off=alpha*ones(M_1);
B_main=(2*alpha-1)*ones(M,1);
A=diag(A_main,0)+diag(A_off,1)+diag(A_off,-1);
B=diag(B_main,0)+diag(A_off,1)+diag(A_off,-1);
```

The next stage is to add in the boundary conditions:

```
A(N,N-1)=2*alpha;
A(N,N)=-1-2*alpha+2*alpha*dr/r(N);
B(N,N-1)=-2*alpha;
B(N,N)=2*alpha-1-2*alpha*dr/r(N);
gamma=-2*dr*R/D;
p=zeros(N,1);
p(N)=gamma;
```

Now we go about the business of solving the equation:

```
v=zeros(M,N)
u=zeros(M,N)
v(1,:)=v_0;
for␣i=2:M
c=p*(q(i)+q(i-1));
rhs=B*v(i-1,:)'+c;
sol=linsolve(A,rhs);
v(i,:)=sol';
v(i,1)=0
u(i,:)=v(i,:)./r_prime;
end
```

3 Cylindrical Diffusion

3.1 Derivation of Method

The cylindrical diffusion equation is given by:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial r^2} + \frac{D}{r}\frac{\partial u}{\partial r} + Q(t,r) \tag{26}$$

With boundary conditions:

$$\frac{\partial u}{\partial r}\Big|_{r=0} = 0, \quad -\lambda\frac{\partial u}{\partial r}\Big|_{r=R} = hu\Big|_{r=R} \tag{27}$$

Evaluate the equation at $r = r_j$

$$\left.\frac{\partial u}{\partial t}\right|_{r=r_j} = D\left.\frac{\partial^2 u}{\partial r^2}\right|_{r=r_j} + \left.\frac{D}{r}\frac{\partial u}{\partial r}\right|_{r=r_j} + Q(t,r)\Big|_{r=r_j} \tag{28}$$

The approximations for the derivatives was explained in a previous section. Using the α and β parameters, the Crank-Nicholson scheme becomes:

$$u_{i+1,j}-u_{i,j}=\frac{\alpha}{2}u_{i+1,j+1}-\alpha u_{i+1,j}+\frac{\alpha}{2}u_{i+1,j-1}+\frac{\beta}{2r_j}u_{i+1,j+1}-\frac{\beta}{2r_j}u_{i+1,j-1}+$$
$$+\frac{\alpha}{2}u_{i,j+1}-\alpha u_{i,j}+\frac{\alpha}{2}u_{i,j-1}+\frac{\beta}{2r_j}u_{i,j+1}-\frac{\beta}{2r_j}u_{i,j-1}+\frac{1}{2}(Q(t_{i+1},r_j)+Q(t_i,r_j))\delta t$$

Denote $Q(t_i,r_j)=Q_{ij}$.This can be rearranged to:

$$\left(\frac{\alpha}{2}+\frac{\beta}{2r_j}\right)u_{i+1,j+1}-(1+\alpha)u_{i+1,j}+\left(\frac{\alpha}{2}-\frac{\beta}{2r_j}\right)u_{i+1,j-1}=-\left(\frac{\alpha}{2}+\frac{\beta}{2r_j}\right)u_{i,j+1}+$$
$$+(\alpha-1)u_{i,j}-\left(\frac{\alpha}{2}-\frac{\beta}{2r_j}\right)u_{i,j-1}-\frac{1}{2}(Q_{i+1,j}+Q_{i,j})\delta t \quad (29)$$

As with the spherical case, the boundary condition at the origin needs to be examined carefully and we do this in the exact same way in the spherical case, so for the case $r=0$ we need to consider the following PDE:

$$\frac{\partial u}{\partial t}=2D\frac{\partial^2 u}{\partial r^2}+Q(t,0) \quad (30)$$

The Crank-Nicholson scheme for this at $r=0$ is given by:

$$u_{i+1,0}-u_{i,0}=\alpha u_{i+1,1}-2\alpha u_{i+1,0}+\alpha u_{i+1,-1}+\alpha u_{i,1}-2\alpha u_{i,0}+\alpha u_{i,-1}+\frac{1}{2}(Q_{i+1,0}+Q_{i,0})\delta t \quad (31)$$

The equation can be rewritten as:

$$2\alpha u_{i+1,1}-(2\alpha+1)u_{i+1,0}=-2\alpha u_{i,1}+(2\alpha-1)u_{i,0}-\frac{1}{2}(Q_{i+1,0}+Q_{i,0})\delta t \quad (32)$$

The final boundary condition to examine is the outer one. This was examined previously and can be written in a discrete form:

$$u_{i,N+1}=u_{i,N-1}-\frac{2h\delta r}{\lambda}u_{i,N}$$

Setting $j=N$ in equation (29) shows that:

$$\left(\frac{\alpha}{2}+\frac{\beta}{2r_N}\right)u_{i+1,N+1}-(1+\alpha)u_{i+1,N}+\left(\frac{\alpha}{2}-\frac{\beta}{2r_N}\right)u_{i+1,N-1}=-\left(\frac{\alpha}{2}+\frac{\beta}{2r_N}\right)u_{i,N+1}+$$
$$+(\alpha-1)u_{i,N}-\left(\frac{\alpha}{2}-\frac{\beta}{2r_N}\right)u_{i,N-1}-\frac{1}{2}(Q_{i+1,N}+Q_{i,N})\delta t$$

Inserting the expression for $u_{i,N+1}$ shows that:

$$\left(\frac{\alpha}{2}+\frac{\beta}{2r_N}\right)\left[u_{i+1,N-1}-\frac{2h\delta r}{\lambda}u_{i+1,N}\right]-(1+\alpha)u_{i+1,N}+\left(\frac{\alpha}{2}+\frac{\beta}{2r_N}\right)u_{i+1,N-1}=$$
$$-\left(\frac{\alpha}{2}+\frac{\beta}{2r_N}\right)\left[u_{i,N-1}-\frac{2h\delta r}{\lambda}u_{i,N}\right]+(\alpha-1)u_{i,N}-\left(\frac{\alpha}{2}-\frac{\beta}{2r_N}\right)u_{i,N-1}+$$
$$-\frac{1}{2}(Q_{i+1,N}+Q_{i,N})\delta t$$

This can be arranged to:

$$\alpha u_{i+1,N-1}-\left(1+\alpha+\frac{2h\delta r}{\lambda}\left(\frac{\alpha}{2}+\frac{\beta}{r_N}\right)\right)u_{i+1,N}=$$
$$=-\alpha u_{i,N-1}+\left(\alpha-1+\frac{2h\delta r}{\lambda}\left(\frac{\alpha}{2}+\frac{\beta}{r_N}\right)\right)u_{i,N}-\frac{1}{2}(Q_{i+1,N}+Q_{i,N})\delta t \quad (33)$$

3.2 Numerical Implementation of Scheme

We can use a lot of the ideas from the spherical case and move it over to the cylindrical case. We'll start with a script rather than a function. The beginning is essentially the same as the spherical case:

```
n=100;m=100;
r=linspace(0,1,n);
t=linspace(0,2,m);
dr=r(2);
dt=t(2);
u_0=r.*(1-r);
Q=cos(t)'*u_0;
```

Then we can compute the values of α and β:

```
D=1e-4;
lambda=1;
h=0.5;
alpha=D*dt/dr^2;
beta=D*dt/(2*dr);
```

The main equation is the same as the spherical case:

$$A[\mathbf{u}]^{k+1}=B[\mathbf{u}]^k+\mathbf{c} \quad (34)$$

The next stage is to fill out the matrices A and B, they're still tri-diagonal.

```
A_main=-(1+alpha)*ones(n,1);
A_u=0.5*alpha-beta./(2*r(1:n-1));
A_l=0.5*alpha+beta./(2*r(1:n-1));
A=diag(A_u,1)+diag(A_l,-1)+diag(A_main,0);
```

The matrix B is completed next:

```
B_main=(alpha-1)*ones(n,1);
B=diag(-A_u,1)+diag(B_main,0)+diag(-A_l,-1);
```

As this has a source term, the value for c will be full The value will be:

$$c_j = -\frac{1}{2}(Q_{i+1,j} + Q_{i,j})\delta t \tag{35}$$

This will be the same for the boundary conditions. The source term is implemented as:

```
c=-0.5*(Q(i+1,:)+Q(i,:))*dt;
```

The next stage is to put in the inner and outer boundary conditions. The changes in A and B are:

$$A_{11} = -(1+2\alpha), \quad A_{12} = 2\alpha, \quad B_{11} = (2\alpha - 1), \quad B_{12} = -2\alpha \tag{36}$$

This is implemented as:

```
A(1,1)=-(1+2*alpha);
A(1,2)=2*alpha;
B(1,1)=2*alpha-1;
B(1,2)=-2*alpha;
```

The outer boundary condition is much more complicated. The changes are more complicated than in the spherical case:

$$\begin{aligned} A_{nn} = -\left[1 + \alpha + \frac{2h\delta r}{\lambda}\left(\frac{\alpha}{2} + \frac{\beta}{2r_N}\right)\right], \quad A_{n(n-1)} = \alpha, \\ B_{nn} = \alpha - 1 + \frac{2h\delta r}{\lambda}\left(\frac{\alpha}{2} + \frac{\beta}{2r_N}\right), \quad B_{n(n-1)} = -\alpha \end{aligned} \tag{37}$$

Implementing this in the code looking like:

```
A(n,n)=-(1+alpha+(2*h*dr/lambda)*(0.5*alpha+beta/(2*r(n))));
A(n,n-1)=alpha;
B(n,n)=alpha-1+(2*h*dr/lambda)*(0.5*alpha+beta/(2*r(n)));
B(n,n-1)=-alpha;
B_1=A\B;
```

Unlike the spherical case with no source term, it's not possible to prealocate certain vectors, so the only thing to do is compute the loop:

```
for␣i=2:m
c=Q(i,:)'+Q(i-1,:):
sol=B_1*u(i-1,:)'+A\c;
u(i,:)=sol';
end
```

Which completes the code.

4 Semi-Linear Diffusion Equation in Cylindrical Co-Ordinates

Consider the equation:

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial r^2} + \frac{D}{r}\frac{\partial u}{\partial r} + f(u) \tag{38}$$

With boundary conditions:

$$\left.\frac{\partial u}{\partial r}\right|_{r-0} = 0, \quad -\lambda\left.\frac{\partial u}{\partial r}\right|_{r=R} = hu\Big|_{r=R} \tag{39}$$

Previous techniques won't work due to the nonlinear nature of the problem. However, much can still be done.

4.1 Derivation of Method

Evaluate the equation at $r = r_j$ as the previous case:

$$\left.\frac{\partial u}{\partial t}\right|_{r=r_j} = D\left.\frac{\partial^2 u}{\partial r^2}\right|_{r=r_j} + \left.\frac{D}{r}\frac{\partial u}{\partial r}\right|_{r=r_j} + f(u)\Big|_{r=r_j} \tag{40}$$

The approximations for the derivatives was explained in a previous section. Using the α and β parameters, the Crank-Nicholson scheme becomes:

$$u_{i+1,j} - u_{i,j} = \frac{\alpha}{2}u_{i+1,j+1} - \alpha u_{i+1,j} + \frac{\alpha}{2}u_{i+1,j-1} + \frac{\beta}{2r_j}u_{i+1,j+1} - \frac{\beta}{2r_j}u_{i+1,j-1} +$$
$$+ \frac{\alpha}{2}u_{i,j+1} - \alpha u_{i,j} + \frac{\alpha}{2}u_{i,j-1} + \frac{\beta}{2r_j}u_{i,j+1} - \frac{\beta}{2r_j}u_{i,j-1} + \frac{1}{2}(f(u_{i+1,j}) + f(u_{i,j}))$$

Denote $f(u_{i,j}) = f_{ij}$.This can be rearranged to:

$$\left(\frac{\alpha}{2} + \frac{\beta}{2r_j}\right)u_{i+1,j+1} - (1+\alpha)u_{i+1,j} + \left(\frac{\alpha}{2} - \frac{\beta}{2r_j}\right)u_{i+1,j-1} = -\left(\frac{\alpha}{2} + \frac{\beta}{2r_j}\right)u_{i,j+1} +$$
$$+ (\alpha - 1)u_{i,j} - \left(\frac{\alpha}{2} - \frac{\beta}{2r_j}\right)u_{i,j-1} - \frac{1}{2}(f_{i+1,j} + f_{i,j}) \tag{41}$$

As with the spherical case, the boundary condition at the origin needs to be examined carefully and we do this in the exact same way in the spherical case, so for the case $r = 0$ we need to consider the following PDE:

$$\frac{\partial u}{\partial t} = 2D\frac{\partial^2 u}{\partial r^2} + f(u(t,0)) \tag{42}$$

The Crank-Nicholson scheme for this at $r = 0$ is given by:

$$u_{i+1,0} - u_{i,0} = \alpha u_{i+1,1} - 2\alpha u_{i+1,0} + \alpha u_{i+1,-1} + \alpha u_{i,1} - 2\alpha u_{i,0} + \alpha u_{i,-1} + f_{i+1,0} + f_{i,0} \tag{43}$$

The equation can be rewritten as:

$$2\alpha u_{i+1,1} - (2\alpha + 1)u_{i+1,0} = -2\alpha u_{i,1} + (2\alpha - 1)u_{i,0} - f_{i+1,0} - f_{i,0} \tag{44}$$

The final boundary condition to examine is the outer one. This was examined previously and can be written in a discrete form:

$$u_{i,N+1} = u_{i,N-1} - \frac{2h\delta r}{\lambda}u_{i,N}$$

Setting $j = N$ in equation (41) shows that:

$$\left(\frac{\alpha}{2} + \frac{\beta}{2r_N}\right)u_{i+1,N+1} - (1+\alpha)u_{i+1,N} + \left(\frac{\alpha}{2} - \frac{\beta}{2r_N}\right)u_{i+1,N-1} = -\left(\frac{\alpha}{2} + \frac{\beta}{2r_N}\right)u_{i,N+1} + \\ + (\alpha - 1)u_{i,N} - \left(\frac{\alpha}{2} - \frac{\beta}{2r_N}\right)u_{i,N-1} - f_{i+1,N} - f_{i,N}$$

Inserting the expression for $u_{i,N+1}$ shows that:

$$\left(\frac{\alpha}{2} + \frac{\beta}{2r_N}\right)\left[u_{i+1,N-1} - \frac{2h\delta r}{\lambda}u_{i+1,N}\right] - (1+\alpha)u_{i+1,N} + \left(\frac{\alpha}{2} + \frac{\beta}{2r_N}\right)u_{i+1,N-1} = \\ -\left(\frac{\alpha}{2} + \frac{\beta}{2r_N}\right)\left[u_{i,N-1} - \frac{2h\delta r}{\lambda}u_{i,N}\right] + (\alpha - 1)u_{i,N} - \left(\frac{\alpha}{2} - \frac{\beta}{2r_N}\right)u_{i,N-1} + \\ - f_{i+1,N} - f_{i,N}$$

This can be arranged to:

$$\alpha u_{i+1,N-1} - \left(1 + \alpha + \frac{2h\delta r}{\lambda}\left(\frac{\alpha}{2} + \frac{\beta}{r_N}\right)\right)u_{i+1,N} = \\ = -\alpha u_{i,N-1} + \left(\alpha - 1 + \frac{2h\delta r}{\lambda}\left(\frac{\alpha}{2} + \frac{\beta}{r_N}\right)\right)u_{i,N} - f_{i+1,N} - f_{i,N} \tag{45}$$

So far this has been exactly the same of the previous however all the equations can be written as a vector equation as:

$$\mathbf{g}(\mathbf{u}) = \mathbf{0} \tag{46}$$

This is a set of nonlinear *algebraic* equations. One common method for solving these is Newton's method described as follows. A recursion is defined as:

$$\mathbf{u}^{n+1} = \mathbf{u}^n - [J_{\mathbf{g}}(\mathbf{u}^n)]^{-1}\mathbf{g}(\mathbf{u}^n) \tag{47}$$

The matrix representation of the Jacobian, is given by:

$$J_{i,j} = J_{g_i}(u_j) = \frac{\partial g_i}{\partial u_j} \tag{48}$$

This completes the method. There are essentially two stages to the method:

1. Construct the function $\mathbf{g}(\mathbf{u})$.
2. Use Newton's method to solve the resulting system on nonlinear algebraic equations.

4.2 Numerical Implementation of Scheme

4.2.1 Construction of the Function $\mathbf{g}(\mathbf{u})$