Contents


```
% heat_equation - Program to solve the diffusion equation
% using the Backward Euler method
clear; help heat_equation; % Clear memory and print header

%* Initialize parameters (time step, grid spacing, etc.)
tau = 1e-4; % Enter time step
N = 100; % Number of grid points
L = 1; % The system extends from (x)=(0) to (x)=(L)
h = L/N;
i = 0:(N-1);
x = h/2 + i*h;
w = 0.2;
xs = 0.5;
ys = 0.5;
```

```
  heat_equation - Program to solve the diffusion equation
  using the Backward Euler method
```

Initialize Source function

```
S = zeros(N); % Set all elements to zero
xExponent = (x'-xs).^2;
S = exp(-xExponent/w^2);
deltaFunction = zeros(N,1);
deltaFunction(round(N/2))=2;
```

* Set up the Laplacian operator matrix

```
lap = zeros(N);  % Set all elements to zero
coeff = 1/h^2;
for i=2:(N-1)
    lap(i,i-1) = coeff;
    lap(i,i) = -2*coeff;  % Set interior rows
    lap(i,i+1) = coeff;
```

```
end
% Boundary conditions
lap(1,1)=-coeff;
lap(1,2)=coeff;
lap(N,N)=-coeff;
lap(N,N-1)=coeff;
```

* Initialize Q-matrix

```
Q = deltaFunction;
```

* Compute A-matrix (Tn+1)=ATn

```
dM = eye(N) - tau*lap;
```

* Initialize loop and plot variables

```
max_iter = .5/tau;
time = linspace(0,max_iter*tau,max_iter);      % Record time for plots
Qplot(:,1) = Q; % initial value
```

* Loop over desired number of steps

```
for iter=2:round(.25/tau)
    %* Compute new temperature
    Q = dM\(Q)+deltaFunction;
    Qplot(:,iter) = Q(:);
end
for iter=round(.25/tau):max_iter
    %* Compute new temperature
    Q = dM\(Q);
    Qplot(:,iter) = Q(:);
end
```

Plot

figure(2);clf; mesh(time,x,Qplot); xlabel('t (s)'); ylabel('x (m)'); %% Print Plots saveFigurePath = '/Users/kevin/SkyDrive/KTH Work/LaTeX Reports/Heat Equation/Figures/'; %% Plot 1 set(figure(2), 'PaperPositionMode', 'auto'); print('-depsc2', [saveFigurePath ... sprintf('deltaFunctionPlot')]);