Homework 1: Heat Equation

DN2255
Kevin Mead

May 28, 2014

1 Heat Equation

$$
\begin{aligned}
&q_t = \nabla \cdot (\nabla q) + S \\
&q(x, y, 0) = 0 \\
&\mathbf{n} \cdot \nabla q = 0
\end{aligned}
$$

$$
S(x, y) = exp\left(-\frac{(x - 1/2)^2 + (y - 1/2)^2}{w^2}\right) \tag{1}
$$

1.1 Analytical preamble

1. Flux vector from (1), $\vec{F} = -\nabla q$
2. Determine $Q(t) = \int_0^1 \int_0^1 q(x, y, t) dxdy$

2 Discretization and implementation

Let Q_{ij} denote the cell average of q over cell (i,j) so,

$$
Q_{ij}(t) = \frac{1}{\Delta x \Delta y} \int_{x_{i-1/2}}^{x_{i+1/2}} \int_{y_{j-1/2}}^{y_{j+1/2}} q(x_i, y_j, t) dxdy
$$

and defining S_{ij} as the cell average of S over cell (i,j),

$$
S_{ij}(t) = \frac{1}{\Delta x \Delta y} \int_{x_{i-1/2}}^{x_{i+1/2}} \int_{y_{j-1/2}}^{y_{j+1/2}} S(x_i, y_j, t) dxdy
$$

where $\Delta x = x_{i+1/2} - x_{i-1/2}$ and $\Delta y = y_{j+1/2} - y_{j-1/2}$

1. Derive a finite volume method for the spatial part of (1) by integrating and forming

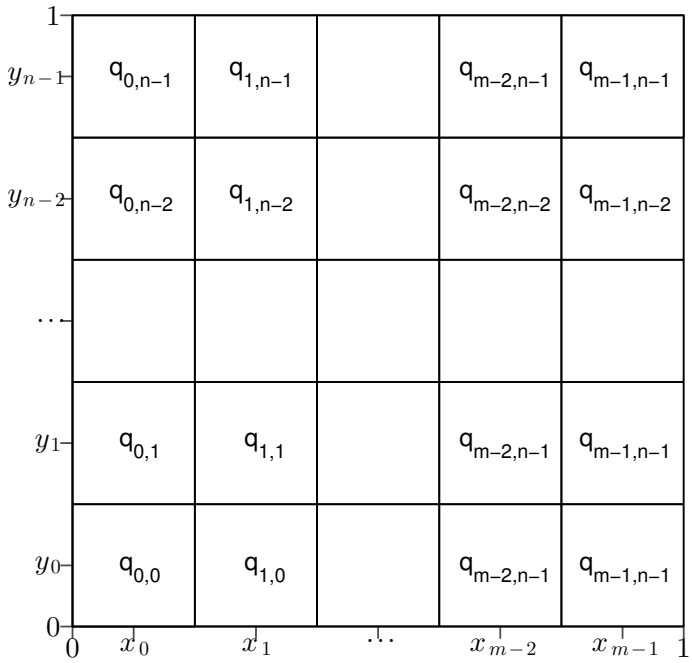

Figure 1: Finite volume grid

cell averages.

$$q_t = \nabla \cdot (\nabla q) + S$$
$$\iint\limits_{\Delta x\,\Delta y} q_t = \iint\limits_{\Delta x\,\Delta y} \nabla^2 q + \iint\limits_{\Delta x\,\Delta y} S$$
$$\frac{d}{dt} \iint\limits_{\Delta x\,\Delta y} q = \iint\limits_{\Delta x\,\Delta y} \left(\frac{d^2 q}{dx^2} + \frac{d^2 q}{dy^2} \right) + \iint\limits_{\Delta x\,\Delta y} S$$

and now dividing everything by the common cell area $\Delta x \Delta y$, we can use (2) and (2) to get

$$\frac{d}{dt} Q_{ij}(t) = \frac{1}{\Delta x \Delta y} \iint\limits_{\Delta x\,\Delta y} \left(\frac{d^2 q}{dx^2} + \frac{d^2 q}{dy^2} \right) + S_{ij}(t)$$

I will Taylor expansions to define the Laplacian terms. The Taylor expansions of $q(x_i + \Delta x)$ and $q(x_i - \Delta x)$, taking the derivative with respect to x are

$$q(x_i + \Delta x) = q(x_i) + \Delta x\, q'(x_i) + \frac{1}{2}(\Delta x)^2 q''(x_i) + \frac{1}{6}(\Delta x)^3 q'''(x_i) + \frac{1}{24}(\Delta x)^4 q^{(4)}(\zeta_+)$$

$$q(x_i - \Delta x) = q(x_i) - \Delta x\, q'(x_i) + \frac{1}{2}(\Delta x)^2 q''(x_i) - \frac{1}{6}(\Delta x)^3 q'''(x_i) + \frac{1}{24}(\Delta x)^4 q^{(4)}(\zeta_-)$$

where the last terms represent a value $\zeta_{\pm}$ that makes the truncation of the Taylor expansion exactly equal to the infinite series. Since we do not know this value, we must remove it and it will be our truncation error. Adding (1) with (1) and rearranging terms,

$$q''(x_i) = \frac{q(x_i + \Delta x) + q(x_i - \Delta x) - 2q(x_i)}{(\Delta x)^2} - \frac{1}{12}(\Delta x)^2 q^{(4)}(\zeta)$$

and taking off the truncation error,

$$q''(x_i) = \frac{q(x_i + \Delta x) + q(x_i - \Delta x) - 2q(x_i)}{(\Delta x)^2}$$

Doing the same thing for y_j and Δy

$$q''(y_j) = \frac{q(y_j + \Delta y) + q(y_j - \Delta y) - 2q(y_j)}{(\Delta y)^2}$$

Now substituting these equations back into (1), rewriting $q(x_i \pm \Delta x)$ as $q_{i\pm1,j}$, $q(y_j \pm \Delta y)$ as $q_{i,j\pm1}$, and for this problem,$\Delta x = \Delta y$,

$$\frac{d}{dt}Q_{ij}(t) = \frac{1}{\Delta x \Delta y} \iint_{\Delta x\, \Delta y} \left(\frac{q_{i+1,j} + q_{i-1,j} + q_{i,j+1} + q_{i,j-1} - 4q_{i,j}}{\Delta x \Delta y} \right) + S_{ij}(t)$$

Lastly, it can be seen that (1) is the 5-point Laplacian combined with the average of q over a cell which is Q_{ij}, so the equation becomes

$$\begin{aligned} \frac{d}{dt}Q_{ij}(t) &= \left(\frac{Q_{i+1,j} + Q_{i-1,j} + Q_{i,j+1} + Q_{i,j-1} - 4Q_{i,j}}{\Delta x \Delta y} \right) + S_{ij}(t) \\ \frac{d}{dt}Q_{ij}(t) &= \Delta_5 Q_{ij} + S_{ij}(t) \end{aligned}$$

At the boundaries, there is no flux, F=0, so $Q_{0,j} = Q_{-1,j}$, $Q_{i,0} = Q_{i,-1}$, $Q_{i,N-1} = Q_{i,N}$, $Q_{M-1,j} = Q_{M,j}$. The stencils for corner boundaries like $i = 0, j = 0$ or $i = m-1, j = n-1$

$$\begin{aligned} \frac{d}{dt}Q^n_{0,0} &= \left(\frac{Q_{1,0} + Q_{-1,0} + Q_{0,1} + Q_{0,-1} - 4Q_{0,0}}{\Delta x \Delta y} \right) + S_{0,0}(t) \\ &= \left(\frac{Q_{1,0} + Q_{0,1} - 2Q_{0,0}}{\Delta x \Delta y} \right) + S_{0,0}(t) \\ \frac{d}{dt}Q^n_{m-1,j-1} &= \left(\frac{Q_{m-2,j-1} + Q_{m-1,j-2} - 2Q_{m-1,j-1}}{\Delta x \Delta y} \right) + S_{m-1,j-1}(t) \end{aligned}$$

Equations for boundaries not at a corner look like this:

$$\begin{cases} \frac{d}{dt}Q^n_{i,0} = \left(\frac{Q_{i+1,0}+Q_{i-1,0}+Q_{i,1}+-3Q_{i,0}}{\Delta x \Delta y}\right) + S_{i,0}, & \text{if } i \neq 0, m-1 \text{ and } j = 0 \\ \frac{d}{dt}Q^n_{i,0} = \left(\frac{Q_{i+1,0}+Q_{i-1,0}+Q_{i,1}+-3Q_{i,0}}{\Delta x \Delta y}\right) + S_{i,0}, & \text{if } i \neq 0, m-1 \text{ and } j = N-1 \\ \frac{d}{dt}Q^n_{0,j} = \left(\frac{Q_{1,j}+Q_{0,j+1}+Q_{0,j-1}-3Q_{0,j}}{\Delta x \Delta y}\right) + S_{0,j}, & \text{if } i = 0 \text{ and } j \neq 0, N-1 \\ \frac{d}{dt}Q^n_{m-1,j} = \left(\frac{Q_{m-2,j}+Q_{m-1,j+1}+Q_{m-1,j-1}-3Q_{m-1,j}}{\Delta x \Delta y}\right) + S_{m-1,j}, & \text{if } i = m-1 \text{ and } j \neq 0, N-1 \end{cases}$$

2. Integrate with implicit Euler scheme

$$\frac{Q^{n+1}_{ij} - Q^n_{ij}}{\tau} = \Delta_5 Q^{n+1}_{ij} + S^n_{ij}(t)$$
$$Q^{n+1}_{ij} = (\mathbf{I} - \tau \mathbf{\Delta_5})^{-1}(Q^n + S^n)$$

   The implicit Euler scheme is unconditionally stable where the explicit scheme has restrictions on the time step. To find the stability of $\mathbf{Q}^{n+1} = \mathbf{A}\mathbf{Q}^n$, we find the eigenvalues of $\mathbf{A}v_k = \lambda_k v_k$. The temperature at a later time can be written in terms of the initial temperature by multiplying the matrix $\mathbf{A}$ by itself n times, $\mathbf{Q}^{n+1} = \mathbf{A}^n\mathbf{Q}^1$. If any eigenvalue of $\mathbf{A}$ satisfies $|\lambda_k| > 1$, then as $n \to \infty$, $Q \to \infty$.[1] The explicit scheme has a value where the eigenvalue could become greater than one. But the implicit scheme takes the inverse of a matrix before the eigenvalues are calculated, thus making the maximum absolute value of an eigenvalue never greater than zero.

3. The laplacian written as the second derivative in 1D for each dimension is convenient because this allows us to easily define the boundary conditions similar to like we would for a purely 1D system.

   i. The fully discrete problem in matrix form

$$Q^{n+1}_{ij}(t) = (\mathbf{I} - \tau(\mathbf{T_x} + \mathbf{T_y}))^{-1}(Q^n + S^n)$$

   where $\mathbf{T_x} = \frac{1}{\Delta x^2}(\delta_{i-1} + \delta_{i+1} - 2\delta_i)$ and $\mathbf{T_y} = \frac{1}{\Delta y^2}(\delta_{j-1} + \delta_{j+1} - 2\delta_j)$.
   Near an x or y boundary, one term from that respective derivative will cancel with the $2\delta_i, j$ term. For example, when $i = 0$, $\mathbf{T_x}$ will become $\frac{1}{\Delta x^2}(\delta_{i+1} - \delta_i)$ because the δ_{i-1} term will cancel with a δ_i term because of boundary conditions.

4. The method was implemented using the operator matrix $\mathbf{A}$[2]

Item		
Animal	Description	Price ($)
4	-1	0
-1	0	0
0	0	0
-1	4	-1
0	-1	0
0	0	0
0	-1	4
0	0	-1
0	0	0
-1	0	0
4	-1	0
-1	0	0
0	-1	0
-1	4	-1
0	-1	0
0	0	-1
0	-1	4
0	0	-1
0	0	0
-1	0	0
4	-1	0
0	0	0
0	-1	0
-1	4	-1
0	0	0
0	0	-1
0	-1	4

3 Numerical results

1. **Solution plots: Show plots of the solution for some time levels before and after t = 1/4, for both source functions S.**

   Figures 2 and 3 are plots of the solution with a delta function and Gaussian function, Equation (1), as source terms. For both of these plots, space discretization was $m = n = 100$ and time discretization was $dt = 1e - 4$ seconds.

2. **Numerical Conservation**: Demonstrate that the method is numerically conservative by looking at

$$\int q \, \mathrm{d}x\mathrm{d}y = \Delta x \Delta y \sum Q_{ij} \tag{2}$$

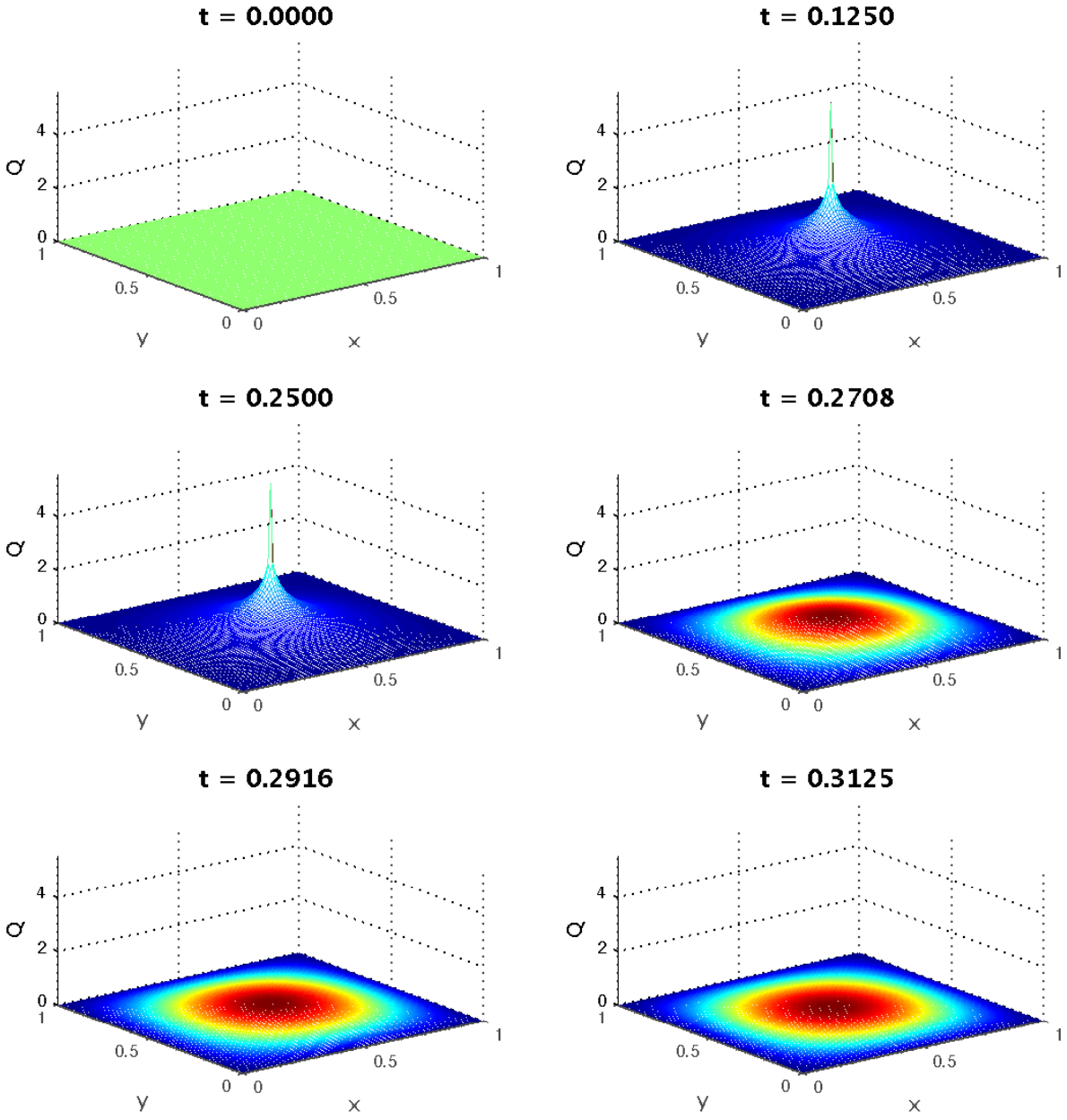

Figure 2: Results with a delta function at the center as a source.

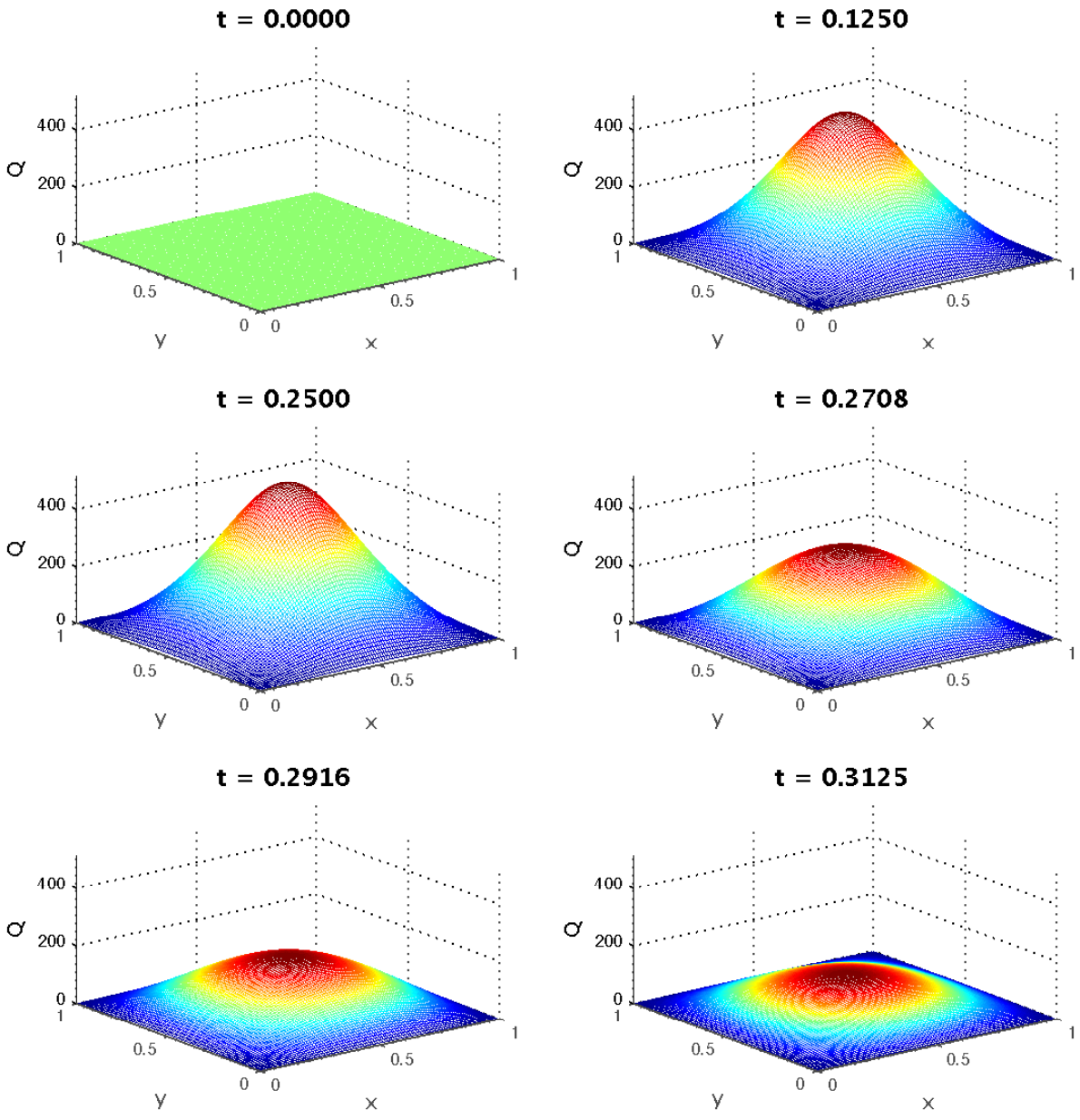

Figure 3: Results with Equation (1) as the source function.

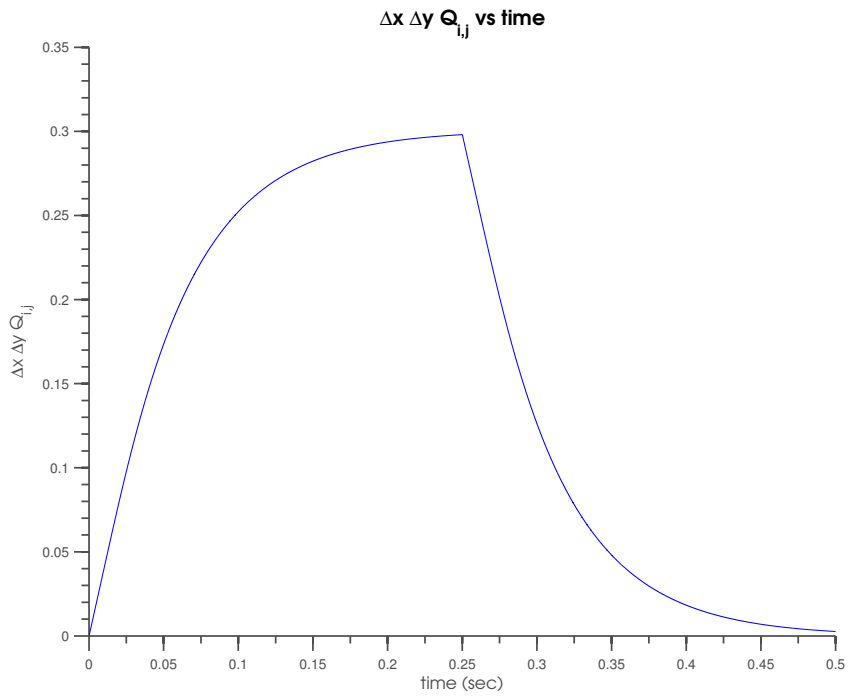

Figure 4: Plot of Equation (2) to show conservation of the quantity $Q_{i,j}$ over time. The plot has a source term adding heat to the system, but after $t = 0.25\ s$, the source is shut off and the amount of Q should not change over time. We do not see this in the above solution, meaning that there is probably something wrong with the algorithm that allows for losses. The correct graph would increase like the one above, but after the source is turned off, the value of $Q_{i,j}$ should remain constant over time.

i. **Time-discretization and how your code handles the discontinuity in g(t)** My code handles the discontinuity in g(t) by using two loops with the first loop including the source term and going from $t = 0$ to $t = ts \cdot \tau$ where $ts = \text{round}(0.25/\tau)$.

- **Space-discretization; where in the cell does (x,y)=(1/2,1/2) appear? different for odd and even m,n** The cell (1/2, 1/2) appears only for odd values of m,n and is at $(m+1)/2$. For example, if $m = n = 11$, then $(x_6, y_6) = (1/2, 1/2)$.

Appendix

1-D Matlab Code

- Initialize Source function
- * Set up the Laplacian operator matrix
- * Initialize Q-matrix
- * Compute A-matrix (Tn+1)=ATn
- * Initialize loop and plot variables
- * Loop over desired number of steps
- Plot

```
% 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')]);

2-D Matlab Code

- * Set up the Laplacian Matrix x-direction
- * Combine lap with identity matrix
- Initialize Source function
- * Initialize loop and plot variables
- * Loop over desired number of steps (wave circles system once)
- Plot

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

%* Initialize parameters (time step, grid spacing, etc.)
tau = 0.05; % Enter time step
N = 100; % Number of grid points
L = 1; % The system extends from (x,y)=(0,0) to (x,y)=(L,L)
h = L/N;
i = 0:(N-1);
j = 0:(N-1);
x = h/2 + i*h;
y = h/2 + j*h;
w = 0.2;
xs = 0.5;
ys = 0.5;
```

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

* Set up the Laplacian Matrix x-direction

```
lap = zeros(N);  % Set all elements to zero
coeff = 1/h^2;
for i=2:(N-1)
    for j=2:(N-1)
        lap(i,i-1,j) = coeff;
        lap(i,i,j) = -4*coeff;  % Set interior rows
        lap(i,i+1,j) = coeff;
        lap(i,i,j-1) = coeff;
        lap(i,i,j+1) = coeff;
    end
end
lap(1,1,1)=-coeff;
lap(1,2,2)=coeff;
```

```
lap(N,N,N)=-coeff;
lap(N,N-1,N-1)=coeff;
iMatrix=zeros(N,N,N);
iMatrix(1:1+N+N^2:end)=1;
```

* Combine lap with identity matrix

```
dM = iMatrix - tau*lap;
```

Initialize Source function

```
S = zeros(N); % Set all elements to zero
xExponent = (x'-xs).^2;
yExponent = (y-ys).^2;
S = exp(-xExponent/w^2);
Q0 = S;
S3D = exp(-xExponent/w^2)*exp(-yExponent/w^2);
mesh(S3D);
```

* Initialize loop and plot variables

```
max_iter = .5/tau;
time = linspace(0,max_iter*tau,max_iter);
Qplot(:,1) = S; % initial value set to source
```

* Loop

```
for iter=2:round(.25/tau)
    %* Compute new temperature
    Q0 = dM\(Q0)+S;
    Qplot(:,iter) = Q0(:);
end
for iter=round(.25/tau):max_iter
    %* Compute new temperature
    Q0 = dM\(Q0);
    Qplot(:,iter) = Q0(:);
end
```

```
Error using \
Inputs must be 2-D, or at least one input must be scalar.
To compute elementwise LDIVIDE, use LDIVIDE (.\) instead.

Error in TwoDHeatEquation (line 54)
    Q0 = dM\(Q0)+S;
```

Plot

```
figure(2);clf;
mesh(time(length(time)/2:end),x,Qplot(:,length(time)/2:end));
xlabel('t'); ylabel('x');
```

References

[1] A. Garcia, *Numerical methods for physics*. Prentice Hall, 2000. [Online]. Available: http://books.google.ca/books?id=MPVAAQAAIAAJ

[2] J. W. Demmel, *Applied Numerical Linear Algebra*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1997.