Assignment 2 - Solutions

Due: October 9, in class
No late assignments accepted

Issued: October 2, 2013

Problem 1

(a) In one dimension with $f(x) = 0$ and boundary conditions $T = 0$ at $x = 0$ and $T = 2$ at $x = 1$ we are solving the differential equation

$$\frac{d^2T}{dx^2} = 0 \quad \text{with} \quad T(0) = 0, \quad T(1) = 2. \tag{1}$$

The solution to this differential equation is a line of the form $T(x) = ax + b$. Setting $T = 0$ at $x = 0$ we must have that $b = 0$. Setting $T = 2$ at $x = 1$ we must have that $a = 2$. Therefore, we know the exact solution is $T(x) = 2x$.

(b) To discretize the one dimensional problem (1), we use a uniform grid on $[0, 1]$ of $N+1$ points with spacing $h = 1/N$. Therefore, the i^{th} point has coordinate $x_i = ih$ for $i = 0, \ldots, N$. Note that in constructing our grid in this fashion, we have exactly $N - 1$ unknowns. The points $x_0 = 0$ and $x_N = 1$ have values fixed by the boundary conditions. We now use the finite central difference equation derived in class to approximate the second derivative of T at a point x_i,

$$f_i = \frac{d^2T}{dx^2}|_{x=x_i} \approx \frac{T_{i+1} - 2T_i + T_{i-1}}{h^2}.$$

Because $f_i = f(x_i) = 0$, at each i we have the equation

$$T_{i+1} - 2T_i + T_{i-1} = 0, \quad 1 \leq i \leq N - 1, \tag{2}$$

where our unknowns are the $N - 1$ values $T_1, \ldots, T_{N-1}$. We must consider the boundary conditions $T(0) = T_0 = 0$ and $T(1) = T_N = 2$. The left boundary condition yields

$$0 = T_0 - 2T_1 + T_2 \implies 0 = -2T_1 + T_2.$$

The right boundary condition yields

$$0 = T_{N-2} - 2T_{N-1} + T_N \implies -2 = T_{N-2} - 2T_{N-1}.$$

We rewrite these equations in a more revealing form

$$\begin{array}{cccccc}
-2T_1 & +1T_2 & & & & = 0 \\
+1T_1 & -2T_2 & +1T_3 & & & = 0 \\
 & +1T_2 & -2T_3 & +1T_4 & & = 0 \\
 & & \vdots & & & \vdots \\
 & & +1T_{N-3} & -2T_{N-2} & +1T_{N-1} & = 0 \\
 & & & +1T_{N-2} & -2T_{N-1} & = -2
\end{array} \tag{3}$$

Therefore, we see that (1) gives us a system

$$\begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & \\ & 1 & -2 & 1 & \\ & & \vdots & & \\ & & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ \vdots \\ T_{N-2} \\ T_{N-1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ -2 \end{bmatrix} \tag{4}$$

(c) For $N = 5$ we want to perform Gaussian Elimination to show that the matrix is non-singular.

$$A = \begin{bmatrix} -2 & 1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -2 \end{bmatrix}$$

$$\begin{bmatrix} -2 & 1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -2 \end{bmatrix} \xrightarrow{R_2 \to R_2 + \frac{1}{2}R_1} \begin{bmatrix} -2 & 1 & 0 & 0 \\ 0 & -3/2 & 1 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -2 \end{bmatrix} \xrightarrow{R_3 \to R_3 + \frac{2}{3}R_2} \begin{bmatrix} -2 & 1 & 0 & 0 \\ 0 & -3/2 & 1 & 0 \\ 0 & 0 & -4/3 & 1 \\ 0 & 0 & 1 & -2 \end{bmatrix}$$

$$\xrightarrow{R_4 \to R_4 + \frac{3}{4}R_3} \begin{bmatrix} -2 & 1 & 0 & 0 \\ 0 & -3/2 & 1 & 0 \\ 0 & 0 & -4/3 & 1 \\ 0 & 0 & 0 & -5/4 \end{bmatrix}$$

We see that we encounter no zero pivots. Hence the matrix A is nonsingular.

(d) We want to show that $T(x)$, defined by

$$T(x) = (2 + 40/(9\pi^2))x + 40/(9\pi^2)\sin(3\pi x/2),$$

satisfies the differential equation

$$\frac{d^2T}{dx^2} = -10\sin(3\pi x/2) \quad \text{with } T(0) = 0 \text{ and } T(1) = 2.$$

Differentiating $T(x)$ twice we have

$$T'(x) = 2 + 40/(9\pi^2) + 20/(3\pi)\cos(3\pi x/2),$$

$$T''(x) = -10\sin(3\pi x/2) = f(x).$$

Now checking the boundary conditions

$$T(0) = (2 + 40/(9\pi^2))x + 40/(9\pi^2)\sin(3\pi x/2)|_{x=0} = 0$$

$$T(1) = (2 + 40/(9\pi^2)) + 40/(9\pi^2)\sin(3\pi/2) = 2$$

We now construct and solve this system of equations in MATLAB using the following function that takes a parameter n controlling the accuracy of the discretization:

```
function [T, x, A] = heatEqnSolver1D(N, f)
% This function solves the one dimensional Heat equation with Dirichlet boundary
% conditions T(0)=0 and T(1)=2 and returns two arrays, T and x.
```

```
% Inputs:
%    N:  controls the number of points in discretization with h = 1/N
%    f:  MATLAB function that evaluates the source function

% Find the points of discretization
h = 1/N;    % Interval size of discretization
x = 0:h:1;  % x = [x_0, x_1, ..., x_N]
x = x(2:end-1);  % x = [x_1, x_2, ..., x_{N-1}]

% Construct the tri-diagonal matrix A
A = diag(ones(N-2,1),1) - 2*eye(N-1) + diag(ones(N-2,1),-1);

% Instead of the line above, use the line below to take advantage of the sparsity of A.
% We can construct A as a sparse matrix directly (this will save us some memory space
% and speed up computation)
% A = spdiags(ones(N-1,2), [-1;1], N-1, N-1) - 2*speye(N-1);

% Construct the right-hand side b from the function f
b = h^2*f(x)';
% Modify b to satisfy the Dirichlet boundary conditions
b(1) = b(1) - 0;     % Apply T_0 = 0
b(end) = b(end) - 2; % Apply T_N = 2

%Solve the system and return the result
T = A\b;
T = [0; T; 2];   % Reattach boundary values on T
x = [0; x'; 1];  % Reattach boundary values on x
end
```

Now we solve for various n and plot all of the results with the following commands:

```
% Run the following commands in MATLAB prompt
f = @(x) -10*sin(3*pi*x/2);
[T5, x5] = heatEqnSolver1D(5, f);
[T10, x10] = heatEqnSolver1D(10, f);
[T25, x25] = heatEqnSolver1D(25, f);
[T200, x200] = heatEqnSolver1D(200, f);
plot(x5, T5, x10, T10, x25, T25, x200, T200);
legend({'N=5', 'N=10', 'N=25', 'N=200'}, 'Location', 'NorthWest');
```

These results are shown in Figure 1.

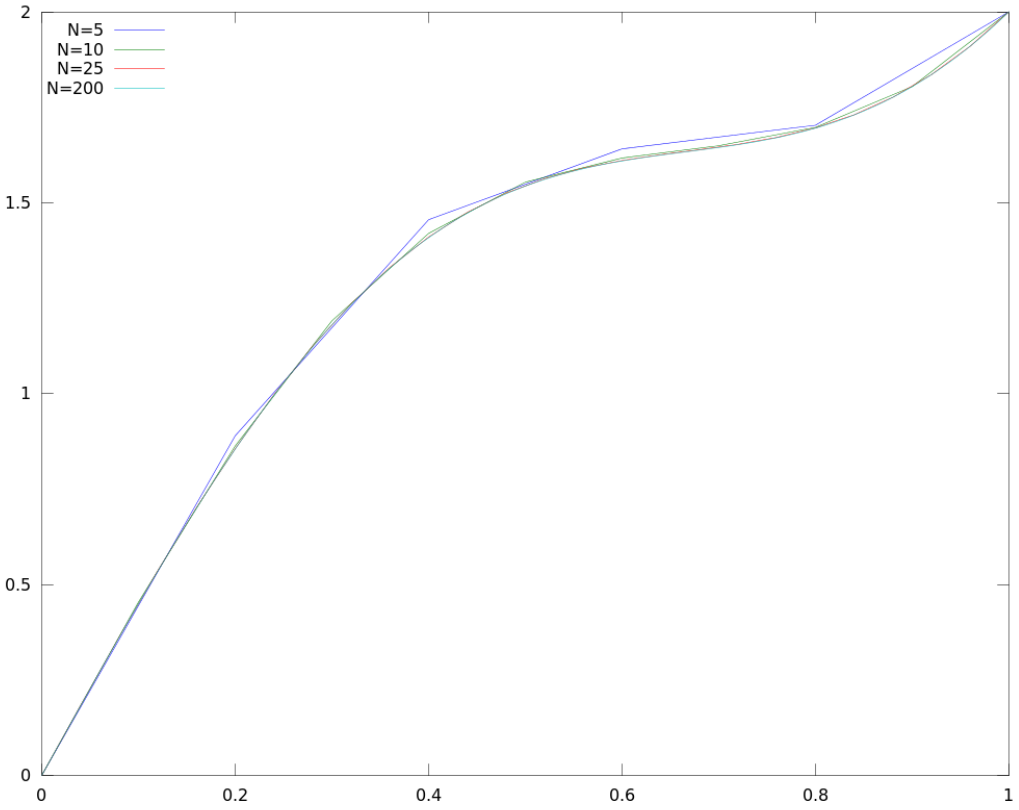

Figure 1: 1(d). Solution to the 1D heat equation with $f(x) = -10\sin(3\pi x/2)$ and boundary conditions $T(0) = 0$ and $T(1) = 2$.

(e) We use the following MATLAB code to compute the condition number of the matrix A for $N = 5,\ N = 10,\ N = 25,$ and $N = 200$.

```
N = [5; 10; 25; 200]
cond_numbers = zeros(4, 1);
figure(2);
delta = 0.1;
f = @(x) -10*sin(3*pi*x/2);

c = ['r', 'b', 'k', 'g'];   % colors for plotting solution
for k=1:length(N)
  h = 1/(N(k));
% Construct sparse matrix A
  A = spdiags(ones(N(k)-1,2), [-1;1], N(k)-1, N(k)-1) - 2*speye(N(k)-1);

  A = A + delta*A;   % Perturbing the system by delta*A
  x = linspace(0, 1, N(k)+1);  % Find the points of discretization
  xind = x(2:end-1);
```

```
  b1 = (h^2)*(f(xind)); % Forming b
  b1(N(k)-1) = b1(N(k)-1) - 2;
  T = A\(b1');    % Solving the system Ax = b
  Tsol = [0; T; 2];

  plot(x', Tsol, c(k));
  hold on
  cond_numbers(k) = cond(A,'fro');   % Frobenius norm condition number
end

% Display condition numbers
cond_numbers
%{
   1.2931e+01
   7.6943e+01
   7.8663e+02
   1.4558e+05
%}

legend({'N=5', 'N=10', 'N=25', 'N=200'}, 'Location', 'SouthEast')
hold off

figure(3);
plot(cond_numbers);
```

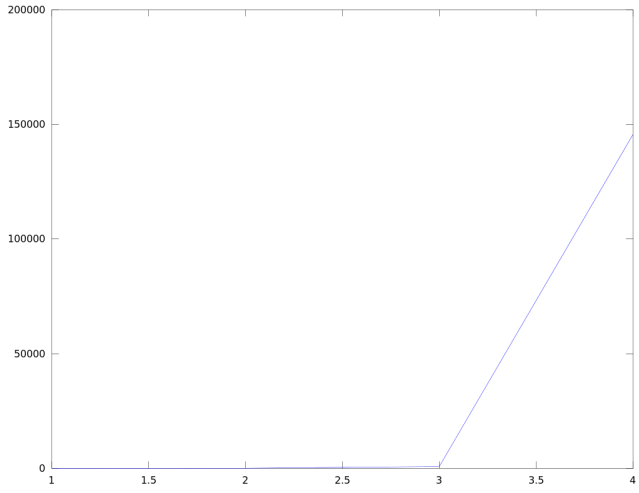

Figure 2: 1(e). Variation of condition number with N.

In Figure 2, we see that the condition number increases with N. The rate of increase is greater than 1 and depends on which norm is used to define the condition number (remember that $\text{cond}(A) = \|A\|\|A^{-1}\|$). Intuitively, this should make sense since with greater number of points of discretization we should be able to notice perturbations to the system better.

Although we see that the condition number is very large for N=200 case, the plot in Figure 3 does

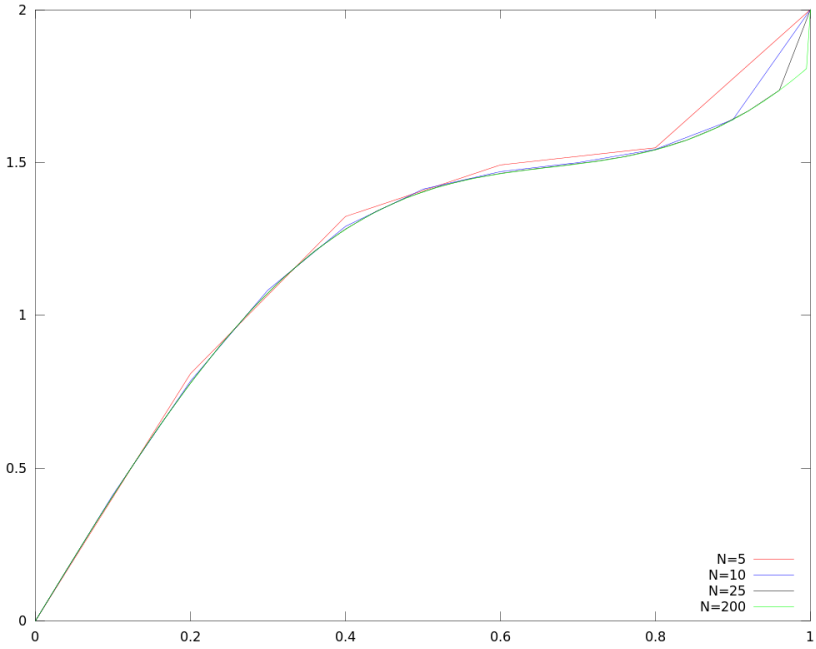

Figure 3: 1(e). Solution to the **uniformly** perturbed system $(A + \partial A)T = b$ for different values of N.

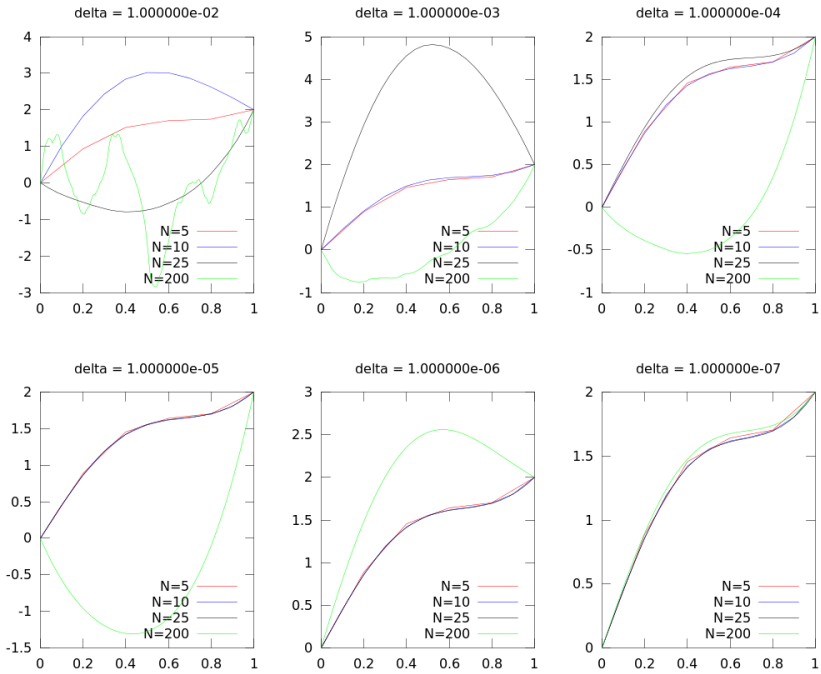

Figure 4: 1(e). Solution to the **randomly** perturbed system $(A + \partial R)T = b$ for different values of delta.

not show large changes to the solution as one could expect. However, keep in mind that the condition number only tells us the worst case performance. On the other hand if you use random perturbations as in,

```
R = rand(N(i)-1,N(i)-1);
A = A + delta*R;   % Perturbing the system by a random perturbation delta*R
```

and let delta vary across a range of values, then you will observe that the solution produced by the discretization with N=200 grid points has the most volatile behaviour. The plot in Figure 4 shows the solutions resulted from 6 different values of delta.

(f) We are given the equation $d^2T/dx^2 = f(x)$ with the boundary conditions $T = 0$ at $x = 0$ and $T = 2$ at $x = 1$ and the source function $f(x)$ defined piecewise

$$f(x) = \begin{cases} 0 & x < 1/3 \\ 100 & \text{otherwise.} \end{cases}$$

We want to verify that the exact solution to this differential equation is given by

$$T(x) = \begin{cases} -\frac{182x}{9} & x < 1/3 \\ \frac{-482x+50}{9} + \frac{100x^2}{2} & \text{otherwise.} \end{cases}$$

We first check that the boundary conditions are satisfied:

$$T(0) = -\frac{182 \cdot 0}{9} = 0 \quad \text{and} \quad T(1) = \frac{-482 + 50}{9} + \frac{100}{2} = -48 + 50 = 2.$$

We now verify that the solution is continuous by checking that the piecewise definition agrees at end point:

$$T_1(1/3) = -\frac{182 \cdot 1/3}{9} = -\frac{182}{27},$$

$$T_2(1/3) = \frac{-482/3 + 50}{9} + \frac{50}{9} = \frac{-482}{27} + \frac{100}{9} = -\frac{182}{27} = T_1(1/3).$$

Thus, $T(x)$ is indeed an exact solution of our differential equation.

Because of the way that we constructed the function heatEqnSolver1D, it is very easy to change the source term and get the plots. The only issue here is to compute the function f to match the piecewise defined source function in the problem:

```
% Run these commands in MATLAB prompt window

% Exact solution
T = @(x) (-182/9*x)*(x<1/3) + ((-482*x+50)/9+50*x^2)*(x>=1/3);

x=0:0.01:1;
for k=1:length(x)
t(k) = T(x(k));    % evaluate exact solution (for plotting)
end

f = @(x) 100*(x>1/3);

% Compute finite difference solutions
[T5, x5] = heatEqnSolver1D(5, f);
```

```
[T10, x10] = heatEqnSolver1D(10, f);
[T25, x25] = heatEqnSolver1D(25, f);
[T200, x200] = heatEqnSolver1D(200, f);

% Plot all solutions together
figure(1);
plot(x5, T5, x10, T10, x25, T25, x200, T200, x, t);
legend({'N=5', 'N=10', 'N=25', 'N=200', 'exact'}, 'Location', 'NorthWest');
```

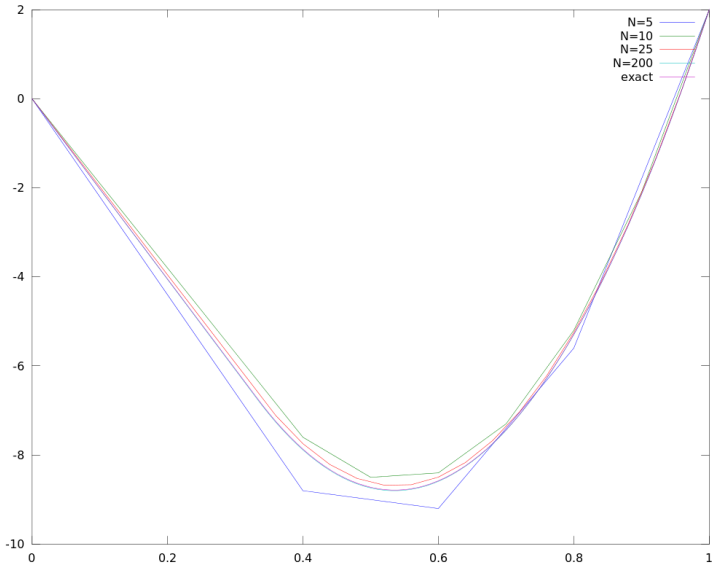

Figure 5: 1(f). Solution to the 1D heat equation in part f.

We now compute the relative error:

```
T_all = {T5, T10, T25, T200};  % cell array with all FD solutions
N = [5, 10, 25, 200];          % number of points used in discretizations
norms_exct = zeros(1,length(N));
rel_errors = zeros(1,length(N));

%compute relative errors wrt 2-norms
for m = 1:length(N)
  h = 1/N(m);       % Interval size of discretization
  x = 0:h:1;
  t = zeros(N(m),1);   % Initialize t (exact solution evaluated at N discrete locations)
  for k = 1:length(x)
    t(k) = T(x(k));
  end
  norms_exct(m) = norm(t);
  norms_exct_apprx(m) = norm(t - T_all{1,m});
  rel_errors(m) = norms_exct_apprx(m)/norms_exct(m);
end
```

```
% Plot relative errors
figure(2);
plot(N, rel_errors);
rel_errors
%rel_errors = [ 0.0897809   0.0328390   0.0137688   0.0017977 ]
```

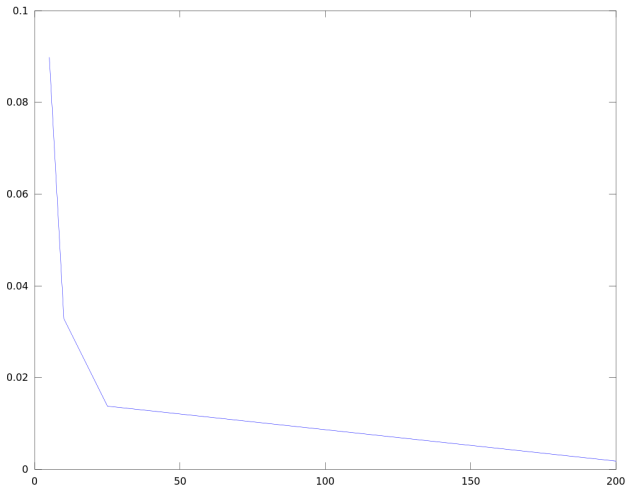

Figure 6: 1(f). Relative errors in part f.

We now figure out how big does N have to be in order for the relative error to be less than 0.01. For that we first make a plot of all relative errors between 25 and 200 and observe an oscillatory exponential decrease in the errors (cf. Figure 7).

```
% Plot relative errors between 25 and 200
s=25;   % starting value of K
t=200;  % final value of K
rel_errs = zeros(1, t-s+1);
for K = s:t
  h = 1/K;    % Interval size of discretization
  x = 0:h:1;

  % compute exact solution evaluated at K discrete points
  T_exct = zeros(K,1);
  for k = 1:length(x)
    T_exct(k) = T(x(k));
  end
  norm_exct = norm(T_exct);

  %compute approx solution with FD discretization with K points
  [T_apprx, x5] = heatEqnSolver1D(K, f);

  %compute relative error
```

```
  norm_exct_apprx = norm(T_exct - T_apprx);
  rel_errs(K-s+1) = norm_exct_apprx/norm_exct;
end

figure(3);
plot([s:t],rel_errs);
```

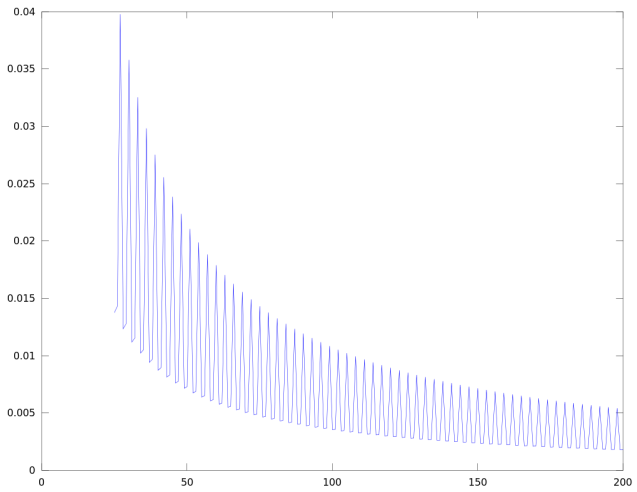

Figure 7: 1(f). Relative errors between $N = 25$ and $N = 200$.

If you "zoom in" the graph in Figure 7 by plotting over a few consecutive values of N (not shown here), you will notice that there is pattern to the observed oscillations. The higher errors occur when the grid points fall close to or exactly on the discontinuity of the source $f(x)$. This can be explained intuitively as follows. Since the source function is not defined at the jump, we are essentially taking a guess of what to pick as the value of $f(x)$ at the discontinuity and this uncertainty results in higher fluctuations in the numerical solution.
Looking at the graph we see that the errors do not exceed 0.01 for N larger than 120. Using this observation, we compute the smallest N (using linear search backtracking) which assures that the relative error is no greater than 0.01.

```
% Now starting at K=120 (obtained by looking at the last plot),
% backtrack to find the smallest value of N that
% will assure the relative error is less than err
err = 0.01;
K = 120;
rel_err = rel_errs(K-s+1);
%we use a linear backtracking search

while (K>s)
  K = K - 1;
  if(err == rel_errs(K-s+1))
    break;
  elseif(err < rel_errs(K-s+1))
```

```
    K = K+1;
    break;
  end
end

%display the smallest K and the corresponding relative error
K
rel_err = rel_errs(K-s+1)
% K =  106
% rel_err =  0.0033432
```

By varying the value of N we established that N has to be at least 106 to get the relative error $\frac{\|T_{exact,N}-T_N\|_2}{\|T_{exact}\|_2}$ less than 0.01.

When a grid point appears exactly on a point of discontinuity of the source $f(x)$, there are a number of ways to deal with this situation. For example, we could take the average value of $f(x)$ at the jump, or we could take either the limiting value to the left or to the right of the jump (i.e. the value of $f(x)$ as x approaches the discontinuity from the left or from the right). However, in general we would try to discretize our domain in a way that avoids the points of discontinuity. In cases where a discontinuity cannot be avoided, the approach we use typically depends on the application.

(g) When we applied boundary conditions in part (b) we were seeking to solve the system of equations given in equation (1). Now we have to represent the Neumann boundary conditions:

$$0 = \frac{dT}{dx}|_{x=0} \approx \frac{T_1 - T_0}{h} \quad \text{and } 0 = \frac{dT}{dx}|_{x=1} \approx \frac{T_N - T_{N-1}}{h}$$

Indeed, we can see that these boundary conditions just constrain the values next to the boundaries

$$T_0 = T_1 \text{ and } T_N = T_{N-1}.$$

Thus, our equations are

$$T_{i+1} - 2T_i + T_{i-1} = h^2 f_i \quad \text{for } 1 < i < N-1,$$

and when we substitute the boundary conditions, we obtain

$$-T_1 + T_2 = h^2 f_1 \quad \text{and} \quad T_{N-2} - T_{N-1} = h^2 f_{N-1}.$$

Thus, with source $f(x) = 0$, the system we obtain is

$$\begin{bmatrix} -1 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \vdots & & & \\ & & 1 & -2 & 1 & \\ & & & 1 & -1 & \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ \vdots \\ T_{N-2} \\ T_{N-1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \tag{5}$$

We check the $N = 5$ case to see if it is singular

$$A = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \to \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \to \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \to \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} = U$$

The two systems, $A\vec{x} = \vec{b}$ and $U\vec{x} = \vec{b}$, have the same solutions. Since the last row of $U\vec{x} = \vec{b}$ is satisfied for any value of $\vec{x}$, it can be eliminated from the system without changing the solution. This leaves us with the system

$$\begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{6}$$

At last we can fix T_4 to any value c (any other T_i works as well) to get the system

$$\begin{bmatrix} -1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -c \end{bmatrix}, \tag{7}$$

which is easily seen to be solvable through back substitution.

(h) Following the same methodology as in 1(b) and 1(g), we get the boundary conditions as

$$T_0 = 0 \ \text{ and } \ T_N = T_{N-1},$$

and we obtain the matrix-vector equation of the form,

$$\begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \vdots & & & \\ & & 1 & -2 & 1 & \\ & & & 1 & -1 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ \vdots \\ T_{N-2} \\ T_{N-1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix}$$

For $N = 5$,

$$A = \begin{bmatrix} -2 & 1 & 0 & 0 \\ 1 & -2 & 1 & 0 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

A is non-singular since $\det(A) \neq 0$.

Problem 2

TRUE or FALSE.

(a) $\text{trace}(A^T A) = \|A\|_F$. True.
Let $A = [\vec{a}_1 \ \vec{a}_2 \ \ldots \ \vec{a}_n]$, where $\vec{a}_1, \ \vec{a}_2, \ \ldots, \ \vec{a}_n$ are the columns of A.

$$A^T = \begin{bmatrix} \vec{a}_1^T \\ \vec{a}_2^T \\ \vdots \\ \vec{a}_n^T \end{bmatrix} \qquad A^T A = \begin{bmatrix} \vec{a}_1^T\vec{a}_1 & \vec{a}_1^T\vec{a}_2 & \ldots & \vec{a}_1^T\vec{a}_n \\ \vec{a}_2^T\vec{a}_1 & \vec{a}_2^T\vec{a}_2 & \ldots & \vec{a}_2^T\vec{a}_n \\ & & \vdots & \\ \vec{a}_n^T\vec{a}_1 & \vec{a}_n^T\vec{a}_2 & \ldots & \vec{a}_n^T\vec{a}_n \end{bmatrix}$$

$\text{trace}(A^T A) =$ sum of the diagonal elements of $A^T A = \sum_{i=1}^{n} \vec{a}_i^T \vec{a}_i = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^2 = \|A\|_F$.

(b) $\text{cond}(\alpha A) = \alpha\ \text{cond}(A)$. False.
$(\alpha A)^{-1} = \frac{1}{\alpha}A^{-1}$. Taking $\alpha > 1$, we show that the statement above does not hold in general,

$$\text{cond}(\alpha A) = \|\alpha A\|\|(\alpha A)^{-1}\| = \alpha\|A\|\frac{1}{\alpha}\|A^{-1}\| = \|A\|\|A^{-1}\| = \text{cond}(A) \neq \alpha\ \text{cond}(A).$$