Homework 5
MA/CS 375, Fall 2005
Due December 1

1. (20 pts.) In this problem we explore the volume of a random parallelepiped in 4-dimensional space whose sides are unit vectors and its connection to the condition number. Using a well-known result of analytic geometry, the volume of a parallelepiped is found as the determinant of the matrix formed by the vectors describing all of its sides originating at a given (any) vertex. Your program should:

   (a) Generate 4 random column vectors of size 4×1
   (b) Normalize these veectors so their length is unity.
   (c) Compute the volume of the parallelepiped they define. You must use the *LU* factorization for computing the determinant of the matrix A whose columns are the 4 random vectors (that is DO NOT use the command $det(\mathbf{A})$ here!).
   (d) compute the condition number of $\mathbf{A}$, $\kappa(\mathbf{A})$.
   (e) Repeat the process 1000 times and produce a scatter plot of $det(\mathbf{A})$ vs $\frac{1}{\kappa(\mathbf{A})}$.

   What are the maximum and minimum values of $det(\mathbf{A})$? What is the condition number for these values? How can you explain the extreme values of the condition number and the corresponding values of the determinant?

   Solutions:

   (a) `>>A=rand(4);`

$$A = \begin{pmatrix} 0.1389 & 0.2722 & 0.4451 & 0.8462 \\ 0.2028 & 0.1988 & 0.9318 & 0.5252 \\ 0.1987 & 0.0153 & 0.4660 & 0.2026 \\ 0.6038 & 0.7468 & 0.4186 & 0.6721 \end{pmatrix}$$

   (b) `>>A=A*diag(1./sum(A))`

$$A = \begin{pmatrix} 0.1214 & 0.2207 & 0.1968 & 0.3767 \\ 0.1772 & 0.1612 & 0.4120 & 0.2338 \\ 0.1737 & 0.0124 & 0.2061 & 0.0902 \\ 0.5277 & 0.6056 & 0.1851 & 0.2992 \end{pmatrix}$$

(c) `[L,U]=lu(A); vol=abs(prod(diag(U)));`

$$L = \begin{pmatrix} 0.23003 & -0.43557 & 0.68563 & 1 \\ 0.33582 & 0.22546 & 1 & 0 \\ 0.32912 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}, \quad U = \begin{pmatrix} 0.52771 & 0.60564 & 0.18512 & 0.29924 \\ 0 & -0.18694 & 0.14512 & -0.0082667 \\ 0 & 0 & 0.31714 & 0.13517 \\ 0 & 0 & 0 & 0.21163 \end{pmatrix}$$

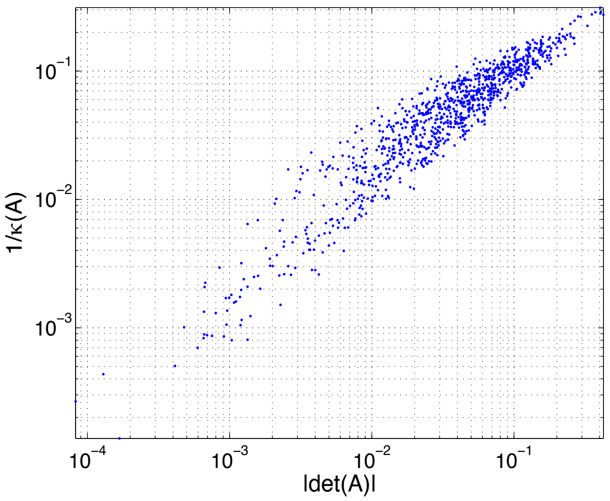
$$\text{vol} = 0.006621...$$

(d) `>>cond(A)`

$$\kappa(A) = 12.701...$$

(e)
```
>> d=zeros(1000,1); kappa=d;
>> for j=1:1000
A=rand(4); A=A*diag(1./sqrt(sum(A.^2)));
[L,U]=lu(A); d(j)=prod(diag(U)); kappa(j)=cond(A);
end
>>loglog(abs(d),kappa,'.')
>>[dmin,mindex]=min(abs(d)); kappamin=kappa(mindex);
>>[dmax,maxdex]=max(abs(d)); kappamax=kappa(maxdex);
```

The minimum determinant was `8.2691e-05` and the corresponding condition number was `3.7498e+03`. The maximum determinant was `0.4247` and the corresponding condition number was `3.6281`



Using `polyfit` to find the regression line, we get that

$$\log \kappa_A = -0.96433 \log(|\det(A)|) - 0.73128$$

such that $\kappa_A \sim |\det(A)|^{-1}$. Therefore, when the determinant becomes small, the condition number becomes large.

2. (20 pts.) Given an n-vector, x, of distinct elements, $x_i \neq x_j$ for $i \neq j$, the Cauchy matrix, $C(x)$, is the $n \times n$ matrix whose ijth entry is:

$$c_{ij} = \frac{1}{x_i + x_j}.$$

Write a program which generates Cauchy matrices of size $n = [4 \quad 8 \quad 12 \quad 16]$ with random input vectors. Compute the condition numbers of these matrices. In each case set $z = \text{ones}(n,1)$ and compute $b = Cz$. Use the built-in Matlab functions to solve $Cz = b$. Denote the computed solution by $\hat{z}$. Compute the relative errors and the relative residuals:

$$\frac{\|z - \hat{z}\|}{\|z\|}, \quad \frac{\|b - C\hat{z}\|}{\|b\|}.$$

Comment on the results. Can you make sense of them given the condition numbers you found?

Solution:

```
function C=cauchymatrix(N)
x=rand(N,1);
C=1./(repmat(x,1,N)+repmat(x',N,1));
```

The condition numbers of the randomly generated Cauchy matrices are

```
>>C4=cauchymatrix(4); C8=cauchymatrix(8); C12=cauchymatrix(12); C16=cauchymatrix(16);
>>K=[cond(C4) cond(C8) cond(C12) cond(C16)]

K =

   3.2794e+08   1.2598e+14   7.9039e+17   1.2376e+18
```

The exact solutions and right-hand-sides are constructed using

```
>>z4=ones(4,1); b4=C4*z4; z4h=C4\b4;
>>z8=ones(8,1); b8=C8*z8; z8h=C8\b8;
>>z12=ones(12,1); b12=C12*z12; z12h=C12\b12;
>>z16=ones(16,1); b16=C16*z16; z16h=C16\b16;
```

Matlab returns singular matrix warnings for the 12×12 and 16×16 matrices. The relative errors and relative residuals are

```
>>re4=norm(z4-z4h)/norm(z4); rr4=norm(b4-C4*z4h)/norm(b4);
>>re8=norm(z8-z8h)/norm(z8); rr8=norm(b8-C8*z8h)/norm(b8);
>>re12=norm(z12-z12h)/norm(z12); rr12=norm(b12-C12*z12h)/norm(b12);
>>re16=norm(z16-z16h)/norm(z16); rr16=norm(b16-C16*z16h)/norm(b16);
>>RE=[re4 re8 re12 re16]

RE =

   9.7061e-09   0.00010243       3.6848       48.705
>>RR=[rr4 rr8 rr12 rr16]

RR =

   1.2303e-16    2.026e-16   1.4917e-16   5.1294e-16
```

The relative residual is $O(\epsilon_m)$ is bounded above by $\kappa(A) \cdot \epsilon_m$.

```
>> eps*K./RE

ans =

       7.5023        273.1       47.629       5.6421
```

Therefore, for a given machine epsilon, one can determine the worst possible relative error in solving the linear system by using the matrix condition number as a bound.

3. (30 pts.) In this problem we solve the heat equation using a second-order accurate method for timestepping. Consider the heat equation

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$$

for $-1 \leq x \leq 1$ and $0 \leq t \leq T$ with boundary conditions

$$u(x=-1,t) = u(x=1,t) = 0 \ ,$$

and initial condition

$$u(x, t = 0) = \sin \pi r x, -1 \le x \le 1 \ .$$

The *Crank-Nicolson* scheme is a popular scheme for solving the heat equation because it is accurate of order 2 both in the timestep, $dt := k$ and in the spatial discretization interval $dx := h$. It is defined by the finite difference equations:

$$\frac{u_m^{n+1} - u_m^n}{k} = \frac{1}{2}\frac{u_{m+1}^{n+1} - 2u_m^{n+1} + u_{m-1}^{n+1}}{h^2} + \frac{1}{2}\frac{u_{m+1}^{n} - 2u_m^{n} + u_{m-1}^{n}}{h^2} \ , \ n = 0, \dots, N-1 \ , \ m = 1, \dots, M-1$$

where we have introduced the discretization:

$$u_m^n := u(x_m, t_n) \ , \ x_m = -1 + h * m \ , \ m = 0, \dots, M \ ; \ t_n = k * n \ , \ n = 0, \dots, N$$

with $h = 2/M$ and $k = T/N$. The reason for the increase in accuracy is that the above expressions give the derivatives of the function $u(x, t)$ with respect to x and t at the point x_m and the instant $t = (t_n + t_{n+1})/2$ so we can think of the time derivative formula as a centered (rather than a forward) difference. You must demonstrate the accuracy of the scheme by carrying out computations for different timesteps k, and comparing your results with the exact solution given by the expression:

$$u_{exact}(x, t) = e^{-\pi^2 r^2 t} \sin \pi r x \ .$$

Specifically:

(a) Write the system of equations in the form

$$(I - \delta D)\, \mathbf{u}^{n+1} = (I + \delta D)\, \mathbf{u}^n \ , \ n = 0, \dots, N-1$$

where

$$\delta = \frac{k}{2h^2} \ , \ \mathbf{u}^n = (u_1^n, \dots, u_{M-1}^n)^T \ (u_0^n = u_M^n = 0)$$

(b) Start the computation off by setting

$$u_m^0 = \sin \pi k x_m = \sin \pi k(-1 + h * m) \ , \ m = 1, \dots, M-1$$

(c) perform the LU factorization of the left-hand matrix $LU = D^- := (I - \delta D)$. You must define D^- (and D^+) using the command *spdiags* to take advantage of its sparsity when solving this system.

(d) Iterate until the final time, $t = T$ is reached.

(e) Run with $T = 1$ and $r = 1$. Use $M = 20, 40, 80, 160$ and $N = 10, 20, 40, 80$.

(f) Compute the relative error

$$e(M,N) = \frac{\sum_{m=1}^{M-1} |u_m^N - u_{exact}(x_m,T)|}{\sum_{m=1}^{M-1} |u_{exact}(x_m,T)|}$$

for each pair (M,N) and show its value in a table.

(g) Determine which pairs work best together; that is determine, for each value of M (and grid-spacing h), which value of N (and timestep k) gives a time-discretization error that is of the same order as the space-discretization error. That is, find the value of N beyond which there is no appreciable improvement in accuracy. For these optimal (M,N) pairs, show that the error decreases like either k^2 or like h^2.

(h) For $M = 160$ and the corresponding optimal value for N, solve the heat equation and plot the solutions for $r = 1,2,3$; for each value of r show on the same plot solutions for 5 equally spaced time instants (your plots must extend over the entire computation interval, including the endpoints). How do the solutions behave as r increases?

Solutions: In matrix form, the equations to solve at every time step

$$\begin{pmatrix} 1+2\delta & -\delta & 0 & \cdots & \cdots & 0 \\ -\delta & 1+2\delta & -\delta & & & \vdots \\ 0 & -\delta & 1+2\delta & & & \vdots \\ \vdots & & & \ddots & & 0 \\ \vdots & & & & \ddots & -\delta \\ 0 & \cdots & \cdots & 0 & -\delta & 1+2\delta \end{pmatrix} \mathbf{u}^{n+1} = \begin{pmatrix} 1-2\delta & \delta & 0 & \cdots & \cdots & 0 \\ \delta & 1-2\delta & \delta & & & \vdots \\ 0 & \delta & 1-2\delta & & & \vdots \\ \vdots & & & \ddots & & 0 \\ \vdots & & & & \ddots & \delta \\ 0 & \cdots & \cdots & 0 & \delta & 1-2\delta \end{pmatrix} \mathbf{u}^{n}$$

The code for the entire simulation is

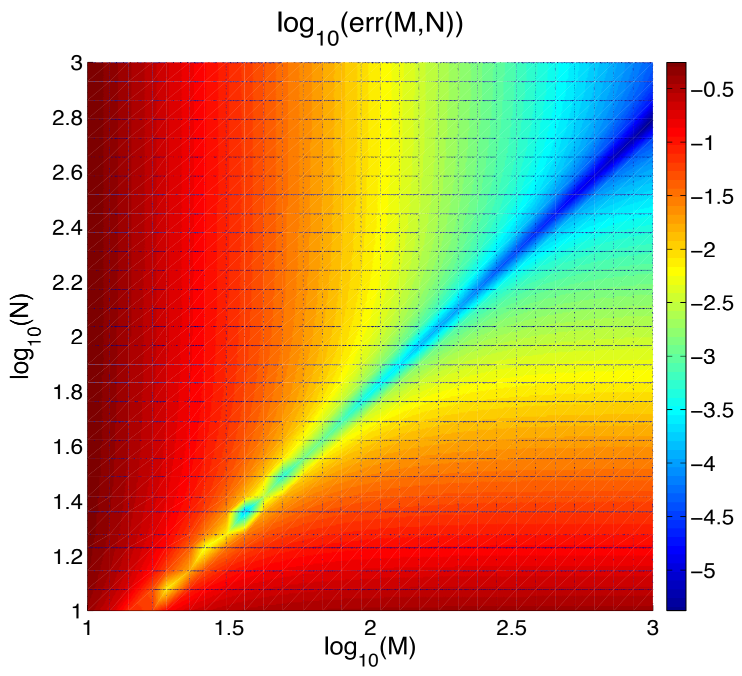
```
function err=heateq(M,N,T,r)

% Solve the Heat Equation using 2nd order finite differences for
% spatial discretization and Crank-Nicholson for time stepping

M1=M+1;                 % # of grid points = # of intervals + 1
x=linspace(-1,1,M1)';   % Set up grid points
h=x(2)-x(1);            % Grid spacing
k=T/N;                  % Time step size
del=k/(2*h^2);          % Crank-Nicholson parameter
```

```
in=2:M; m=M-1;           % indices of interior points
u=sin(pi*r*x);           % Get initial condition
e=ones(M-1,1);           % Column vector of all ones

% Left hand side matrix
A=spdiags([-del*e, (1+2*del)*e, -del*e], -1:1, m, m);

[L,U]=trilu(A);          % Do LU factorization
coeff=1-2*del;           % Lump diagonal coefficient for RHS

for n=1:N                % Time stepping loop

    f=coeff*u+del*([0;u(1:M)]+[u(2:M1);0]); %RHS
    u(in)=U\(L\f(in));
end

exact=exp(-(pi*r)^2*T)*sin(pi*r*x);
err=norm(u-exact,1)/norm(exact,1);

function [L,U] = trilu(A);
n=length(A);  am=diag(A,-1);  a0=diag(A); ap=diag(A,1);
for k = 1:(n-1)
   am(k)=am(k)/a0(k);  k1=k+1;  a0(k1)=a0(k1)-am(k)*ap(k);
end
L=speye(n)+spdiags(am,-1,n,n);  U=spdiags([a0 [0;ap]],0:1,n,n);
```

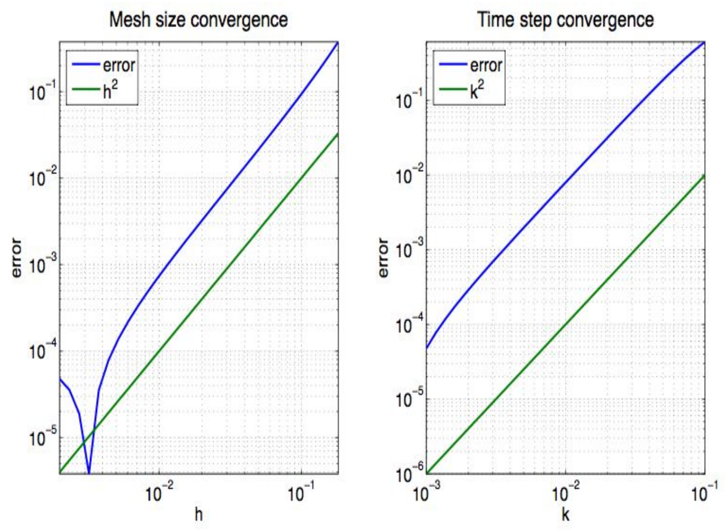
The table of results for the specified M,N pairs is If we look at the error over a wider range of interval

M↓ N→	10	20	40	80
20	5.6688e-01	1.1470e-01	3.2118e-02	7.1080e-02
40	6.0013e-01	1.7002e-01	2.9501e-02	7.8380e-03
80	6.0808e-01	1.8335e-01	4.4370e-02	7.4273e-03
160	6.1005e-01	1.8665e-01	4.8054e-02	1.1210e-02

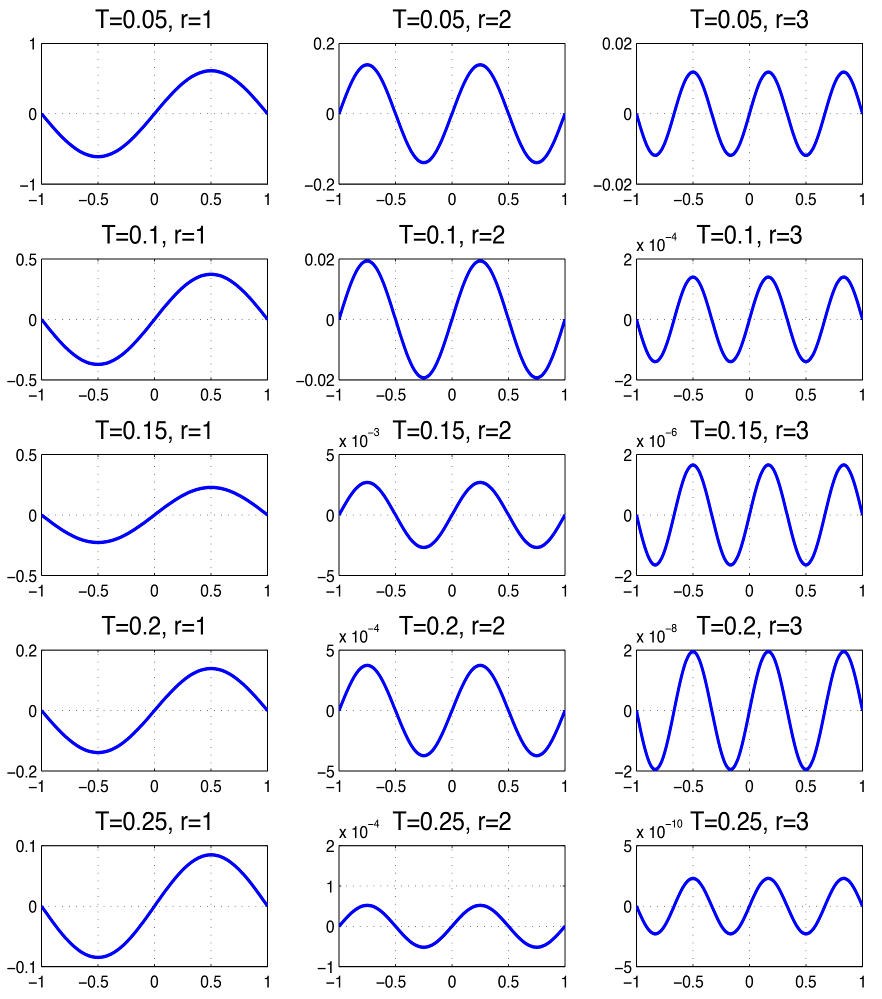
size an time step, the trend is more obvious.



The quadratic convergence in spatial discretization can be observed by fixing one of either the time step or grid spacing to a sufficiently small amount and varying the other. In the plot on the left, the number of subintervals was taken to be 1,000 and the time step was varied. In the plot on the right, the number of time steps was taken to be 1,000 and the mesh size was varied.



For a fixed number of intervals equal to 160, the optimal number of time steps is 251.



Looking at the computed solution for various times and choices of the inital condition paremeter r, we can see that greater values of r result in more oscillations over the interval and greater decay rate. Both of these will lead to requiring additional spatial and temporal resolution to maintain the error to a specified tolerance.

4. (30 pts.) The equilibrium temperature distribution in a slab of reacting material can under some

assumptions be modeled by the nonlinear boundary value problem:

$$\frac{d^2u}{dx^2} + \lambda^2 e^{2u} = 0, \quad x \in (-1, 1),$$

with boundary conditions $u(-1) = u(1) = 0$. The parameter λ is sometimes called the Damköhler number. For this problem take $\lambda = \frac{1}{\cosh 1}$.

i. Verify that:

$$u(x) = -\ln(\lambda \cosh x),$$

solves the problem.

ii. Introducing a uniform grid, $x_j = -1 + jh$, $j = 0, \ldots, n$, $h = \frac{2}{n}$, approximate $u(x_j)$ by U_j and approximate $\frac{d^2u}{dx^2}$ by the second order central difference formula. This leads to the system of $n-1$ nonlinear equations:

$$F_j(u) \equiv \frac{U_{j-1} - 2U_j + U_{j+1}}{h^2} + \lambda^2 e^{2U_j} = 0, \quad j = 1, \ldots, n-1,$$

where we set $U_0 = U_n = 0$. The Jacobian derivative of a collection of $n-1$ functions, F_j of $n-1$ variables, U, is the $(n-1) \times (n-1)$ matrix $J(\mathbf{U})$ whose jkth entry is the derivative of F_j with respect to U_k, $J_{jk} = \frac{\partial F_j}{\partial U_k}$. (We denote the $n-1$-vectors whose entries are F_j and U_k by $\mathbf{F}$ and $\mathbf{U}$.) Show that the Jacobian derivative of the function given above is a tridiagonal matrix whose entries are:

$$J_{j,j+1} = J_{j+1,j} = \frac{1}{h^2}, \quad J_{jj} = -\frac{2}{h^2} + 2\lambda^2 e^{2U_j}.$$

iii. Newton's method for solving a system of $n-1$ equations in $n-1$ unknowns is the direct generalization of Newton's method for a single equation which we studied in Chapter 2. Precisely, given an initial approximation, $\mathbf{U}^{(0)}$, we generate subsequent approximations by solving:

$$J(\mathbf{U}^{(k)})\, \mathbf{d}^{(k)} = -\mathbf{F}(\mathbf{U}^{(k)}),$$

and setting:

$$\mathbf{U}^{(k+1)} = \mathbf{U}^{(k)} + \mathbf{d}^{(k)}.$$

Write a script implementing Newton's method for the difference approximation described above. Terminate the iteration when:

$$\|\mathbf{d}^{(k)}\| \leq 10^{-8}.$$

Use $\mathbf{U}^{(0)} = \mathbf{0}$. Be sure to treat J as a sparse matrix when solving. Record the number of iterations required and print out $\|\mathbf{d}^{(k)}\|$ for each k. Does the convergence appear to be quadratic? Carry out the computations for $n = 10, 20, 40$. Compute the maximum error, that is the maximum absolute difference between U_j and $u(x_j)$, in each case. Do you observe second order convergence with decreasing h? Plot the solutions in each case.

Solution:

$$\frac{du}{dx} = -\tanh(x), \quad \frac{d^2u}{dx^2} = \tanh^2(x) - 1$$

$$\exp(2u) = \exp(-2\ln[\lambda\cosh(x)]) = \exp\left(\ln[\frac{1}{\lambda^2\cosh^2(x)}]\right) = \frac{1}{\lambda^2\cosh^2(x)} = \frac{1}{\lambda^2}\text{sech}^2(x)$$

$$\exp(2u) = \frac{1}{\lambda^2}[1 - \tanh^2(x)] = -\frac{1}{\lambda^2}\frac{d^2u}{dx^2}$$

Solutions: Given

$$F_j(u) \equiv \frac{U_{j-1} - 2U_j + U_{j+1}}{h^2} + \lambda^2 e^{2U_j} = 0, \quad j = 1, \ldots, n-1,$$

We obtain the Jacobian by differentiating with respect to the each of the elements of the vector $\mathbf{u}$.

$$J_{jk}\frac{\partial F_j(u)}{\partial U_k} = \frac{\partial}{\partial U_k}\left(\frac{U_{j-1} - 2U_j + U_{j+1}}{h^2} + \lambda^2 e^{2U_j}\right), \quad \frac{\partial U_j}{\partial U_k} = \delta_{jk}$$

$$J_{jk} = \frac{\delta_{(j-1),k} - 2\delta_{jk} + \delta_{(j+1),k}}{h^2} + 2\lambda^2 e^{2U_j}\delta_{jk}$$

Clearly, nonzero entries can only occur in the range $j-1 \leq k \leq j+1$. Evaluating for the Kronecker delta functions gives the desired results.

$$J_{j,j+1} = J_{j+1,j} = \frac{1}{h^2}, \quad J_{jj} = -\frac{2}{h^2} + 2\lambda^2 e^{2U_j}.$$

The script for this problem is

```
function [iter,nd,err]=nlbvp(n,tol)

np1=n+1; nm1=n-1; in=2:n; x=linspace(-1,1,np1)';
h=x(2)-x(1); h2=h*h; lambda=sech(1); lambda2=lambda*lambda;

e=ones(nm1,1);  D2bands=[e -2*e e]/h2;
D2=spdiags(D2bands, -1:1, nm1, nm1);

exact=-log(lambda*cosh(x));     % Exact solution

uin=0*e;                                % Interior points (initial value)

d=e; nd=norm(d); iter=0;
```

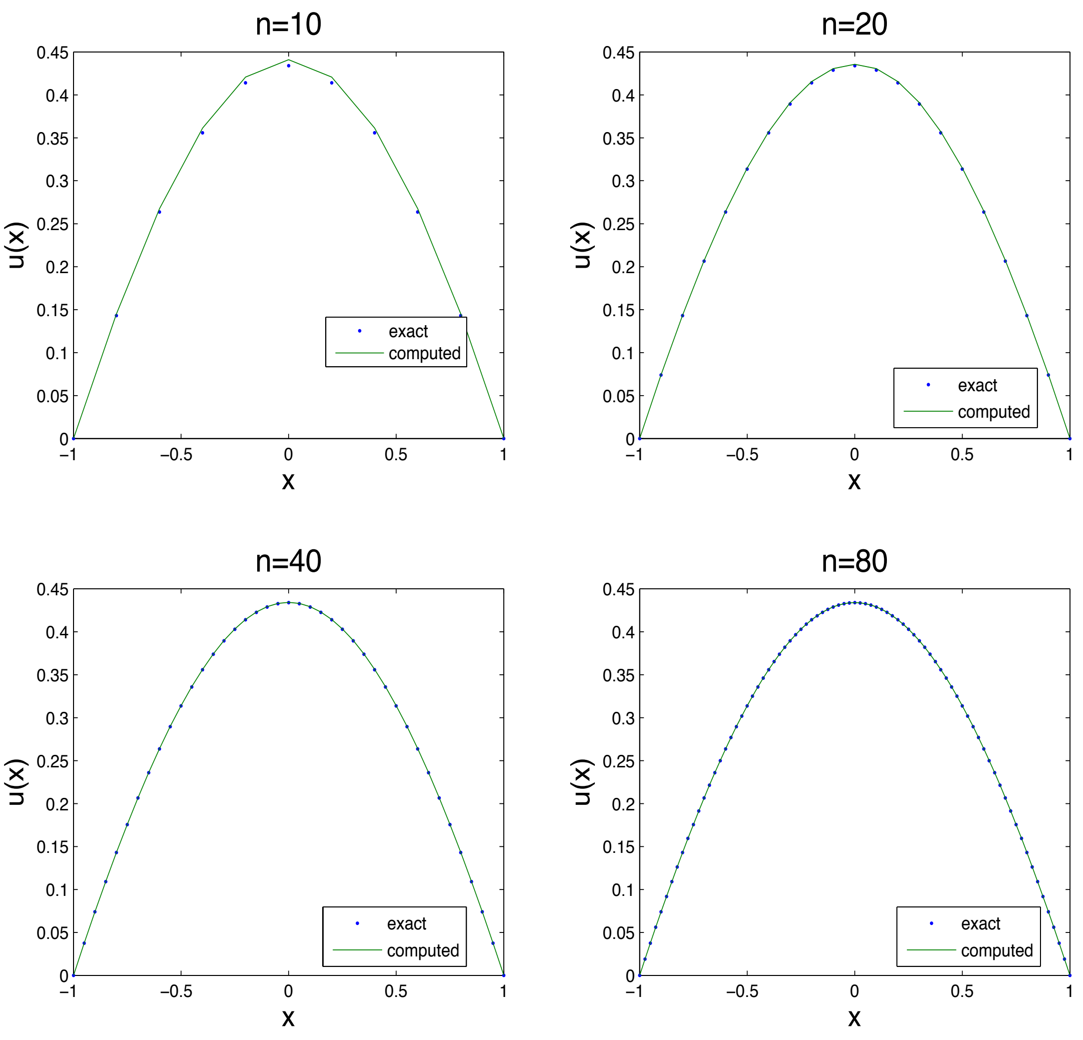
```
while nd>tol
    nonlin=lambda2*exp(2*uin);         % Nonlinear term
    F=trimult(D2bands,uin)+nonlin;     % Forcing function
    J=D2+2*spdiags(nonlin,0,nm1,nm1); % Jacobian
    d=J\(-F);                          % Change term
    nd=norm(d);
    uin=uin+d;                         % Update solution
    iter=iter+1;
end

u=[0;uin;0];    err=norm(exact-u,inf);

function Ax=trimult(A,x);
N=length(x); k=2:N; Ax=A(:,2).*x+[0;A(k,1).*x(k-1)]+[A(k-1,3).*x(k);0];
```

For $n = 80$ we obtain the norm of the change with iterations as The convergence with respect to grid

k $\|\|d^{(k)}\|\|$	
1	2.0789
2	0.5925
3	0.1031
4	0.0038
5	5.2686e-06
6	1.0176e-11

refinement also appears to be quadratic

n	iterations	$\|\|d\|\|$	$\|\|error\|\|_\infty$
10	6	1.5891e-11	7.1784e-03
20	6	7.1673e-12	1.7302e-03
40	6	7.6929e-12	4.2877e-04
80	6	1.0176e-11	1.0696e-04


n=10
n=20
n=40
n=80
u(x)
x
exact
computed