HW2, Math 228A

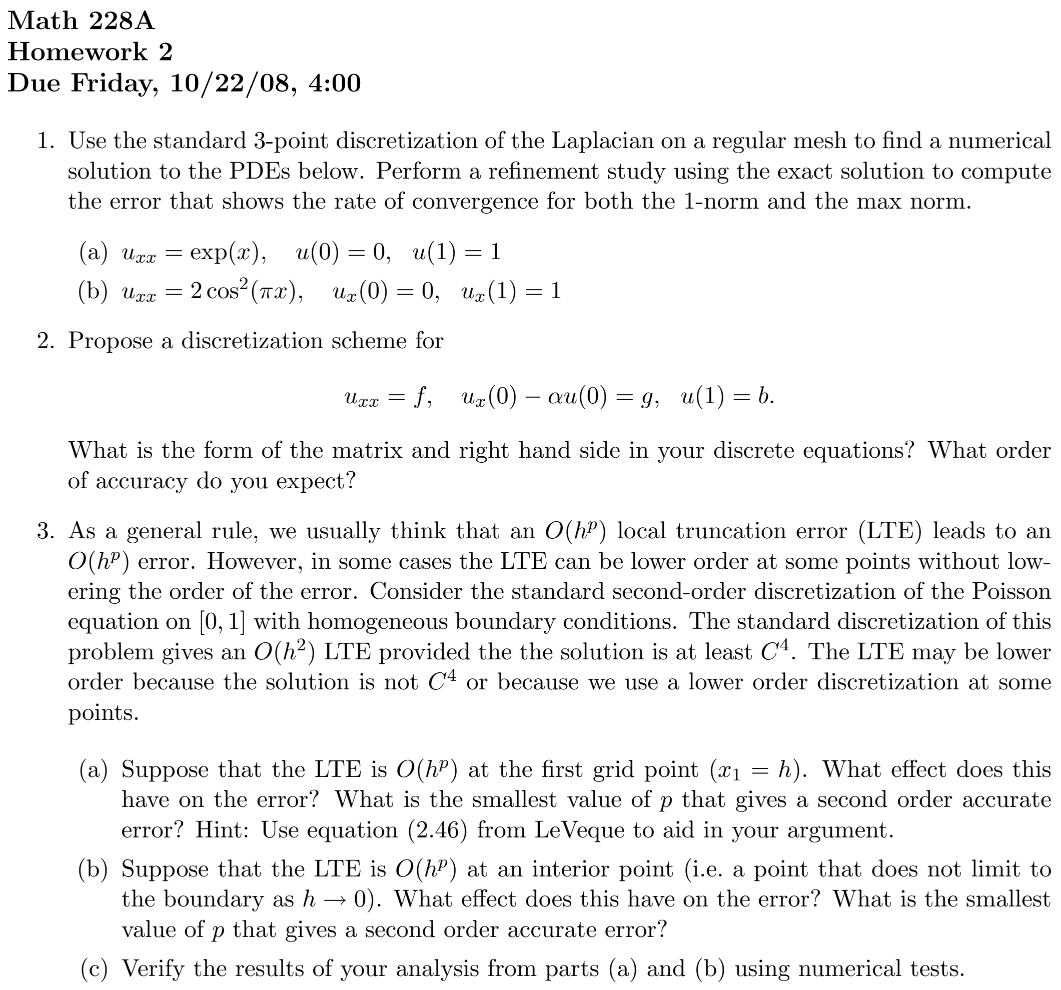
Date due 10/22/2010
UC Davis, California Fall 2010

Nasser M. Abbasi

Fall 2010 Compiled on September 29, 2019 at 12:00am

Contents

1 Problem description



Math 228A
Homework 2
Due Friday, 10/22/08, 4:00

1. Use the standard 3-point discretization of the Laplacian on a regular mesh to find a numerical solution to the PDEs below. Perform a refinement study using the exact solution to compute the error that shows the rate of convergence for both the 1-norm and the max norm.
   (a) $u_{xx} = \exp(x), \quad u(0) = 0, \;\; u(1) = 1$
   (b) $u_{xx} = 2\cos^2(\pi x), \quad u_x(0) = 0, \;\; u_x(1) = 1$
2. Propose a discretization scheme for
$$u_{xx} = f, \quad u_x(0) - \alpha u(0) = g, \;\; u(1) = b.$$
   What is the form of the matrix and right hand side in your discrete equations? What order of accuracy do you expect?
3. As a general rule, we usually think that an $O(h^p)$ local truncation error (LTE) leads to an $O(h^p)$ error. However, in some cases the LTE can be lower order at some points without lowering the order of the error. Consider the standard second-order discretization of the Poisson equation on $[0,1]$ with homogeneous boundary conditions. The standard discretization of this problem gives an $O(h^2)$ LTE provided the the solution is at least C^4. The LTE may be lower order because the solution is not C^4 or because we use a lower order discretization at some points.
   (a) Suppose that the LTE is $O(h^p)$ at the first grid point ($x_1 = h$). What effect does this have on the error? What is the smallest value of p that gives a second order accurate error? Hint: Use equation (2.46) from LeVeque to aid in your argument.
   (b) Suppose that the LTE is $O(h^p)$ at an interior point (i.e. a point that does not limit to the boundary as $h \to 0$). What effect does this have on the error? What is the smallest value of p that gives a second order accurate error?
   (c) Verify the results of your analysis from parts (a) and (b) using numerical tests.

Figure 1: problem description

2 Problem 1

Use standard 3-point discretization of the Laplacian on a regular mesh to find numerical solution to the PDE below and perform refinement study to compute the error that shows the rate of convergence for both the 1-norm and the max-norm.

2.1 part (a)

The differential equation with its boundary conditions is $u_{xx} = e^x$ with $u(0) = 0, u(1) = 1$.

Finding the analytical solution

Let D be a differential operator $D \equiv \frac{d}{dx}$. Since $(D-1)\,e^x = 0$, then applying $(D-1)$ to both sides of the differential equation results in

$$\left[(D-1)\,D^2\right](u) = 0$$

Thus the characteristic equation is $(r-1)\,r^2 = 0$ with roots $r_1 = 1, r_2 = 0$ and $r_3 = 0$. Therefore the complete solution is

$$u(x) = c_1 e^{r_1 x} + c_2 e^{r_2 x} + x c_3 e^{r_3 x}$$

It helps to designate in the above which part is the particular solution and which is the homogeneous

$$u(x) = \overbrace{c_1 e^x}^{u_p} + \overbrace{c_2 + c_3 x}^{y_h}$$

c_1 is found by substituting the particular solution in the original differential equation giving $c_1 = 1$. The solution becomes[1]

$$u(x) = e^x + c_2 + x c_3$$

The remaining constants c_2 and c_3 are found by satisfying the boundary conditions on the above solution. Applying $u(0) = 0$ gives $c_2 = -1$ and applying $u(1) = 1$ gives $c_3 = 2 - e$. The solution becomes

$$u(x) = 2x - \exp(1)\,x + \exp(x) - 1$$

Scheme to use for the numerical solution

The numbering used is the one described in the class. The first grid point has an index $j = 0$, and the last grid point has an index $j = N+1$. U is the unknown to solve for. It is the numerical solution of the differential equation at the grid points. Lower case u is the exact analytical solution found earlier.

Because this problem has Dirichlet boundary conditions, U is known at $j = 0$ and at $j = N+1$, therefore only internal grid points are used to solve for U. These internal points are numbered $1 \cdots N$. The spacing between each grid point is $h = \frac{len}{N+1}$ where the length of the domain len is always taken to have value 1.

The physical x-coordinate of each grid point is given by $x_j = jh$. For example, when $j = 0$, the first point will have a physical x-coordinate of 0, and when $j = N+1$, the last grid point will have a physical x-coordinate of 1.

The diagram below helps illustrate these relations and the notations used.

[1]another way to solve this is to integrate the original differential equation twice to obtain this general solution.

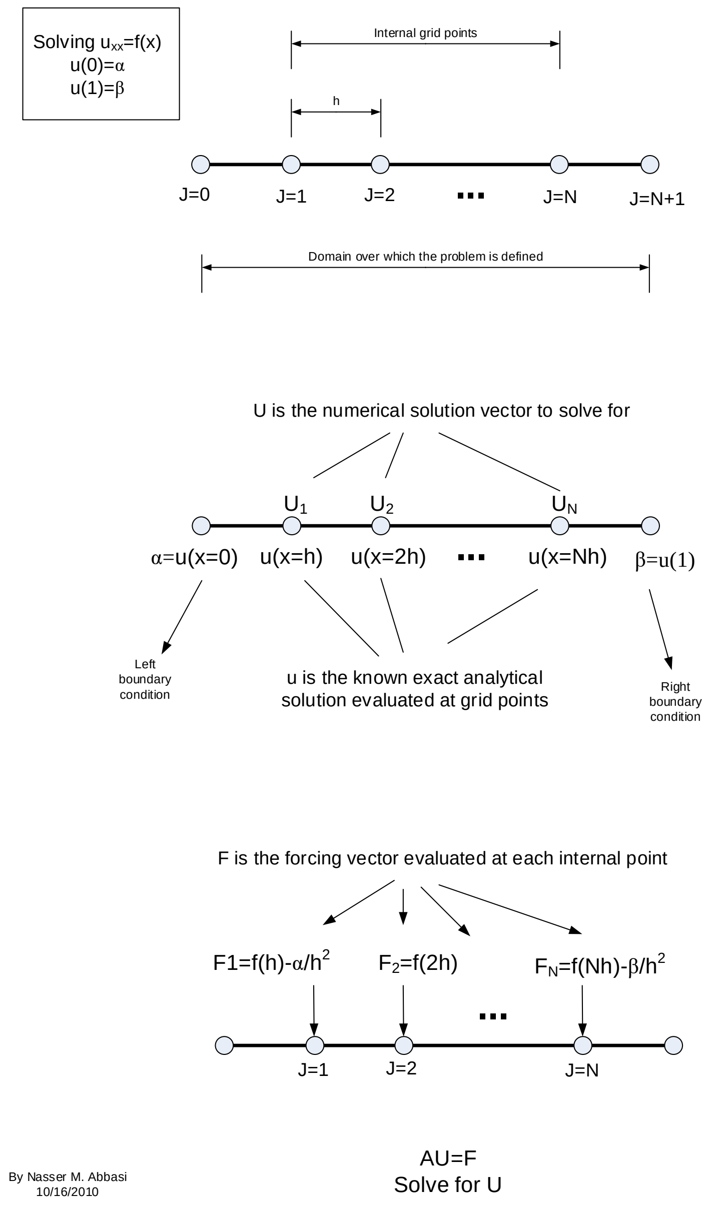


Figure 2: problem 1 part a scheme

The standard 3-point central difference formula $\frac{U_{j-1}-2U_j+U_{j+1}}{h^2}$ was used to approximate u_{xx} on each internal grid point. This approximation has local truncation error (LTE) of $O\left(h^2\right)$. Therefore, at each internal grid point j the equation $u_{xx}=f\left(x\right)$ is approximated by $\frac{U_{j-1}-2U_j+U_{j+1}}{h^2}=f_{x=jh}$.

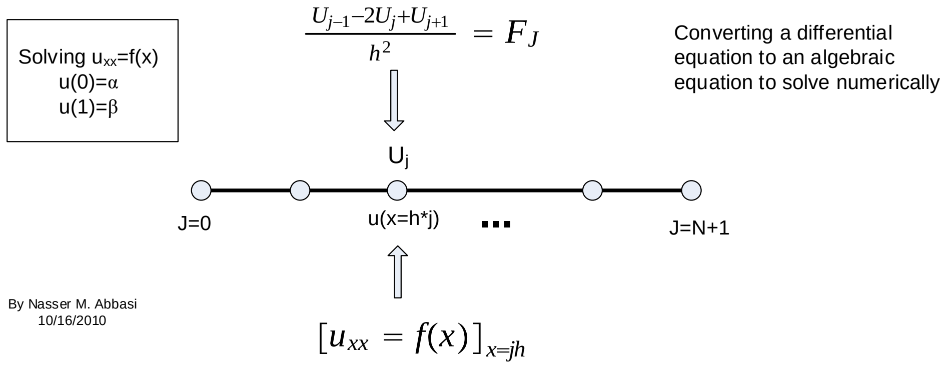

Solving $u_{xx}=f(x)$
$u(0)=\alpha$
$u(1)=\beta$
$\frac{U_{j-1}-2U_j+U_{j+1}}{h^2}=F_J$
Converting a differential equation to an algebraic equation to solve numerically
U_j
J=0
u(x=h*j)
...
J=N+1
By Nasser M. Abbasi 10/16/2010
$[u_{xx}=f(x)]_{x=jh}$

Figure 3: problem 1 part a scheme 2

For the special case of $j = 1$ (the first internal grid point on left side), the above formula was modified by replacing U_o by its given value (U_0 is on the boundary and its value is known). Therefore, for $j = 1$ the discrete equation becomes

$$\frac{\alpha-2U_0+U_1}{h^2}=F_1$$
$$\frac{-2U_0+U_1}{h^2}=F_1-\frac{\alpha}{h^2} \tag{2}$$

The same was done for $j = N$ (the last internal grid point on right side). The standard 3 points formula was modified by replacing U_{N+1} by its given value. Therefore, for $j = N$ the discrete equation becomes

$$\frac{U_{N-1}-2U_N+\beta}{h^2}=F_N$$
$$\frac{U_{N-1}-2U_N}{h^2}=F_N-\frac{\beta}{h^2} \tag{3}$$

The above equations are collected and put in the form $Au = f$ resulting in

$$\frac{1}{h^2}\begin{pmatrix} -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 \end{pmatrix}\begin{pmatrix} U_1 \\ U_2 \\ U_3 \\ \cdot \\ \cdot \\ \cdot \\ U_N \end{pmatrix}=\begin{pmatrix} f_h-\frac{\alpha}{h^2} \\ f_{2h} \\ f_{3h} \\ \cdot \\ \cdot \\ f_{(N-1)h} \\ F_{Nh}-\frac{\beta}{h^2} \end{pmatrix}$$

The above system is solved for the unknown vector U. These are the values of the numerical solution at the internal grid points. The A matrix above is symmetric and tridiagonal and of size $N \times N$ where N is the number of internal grid points.

Error norm calculation

The total error at a grid point j is given by

$$\mathbf{e}_j = \mathbf{U}_j - \mathbf{u}\left(x_j\right)$$

Where $\mathbf{U}_j$ is the numerical solution at the j^{th} grid point and $\mathbf{u}\left(x_j\right)$ is the exact solution sampled at the same grid point location. $\mathbf{e}$ is a vector of length N and represents the difference between U and u at each point.

To measure the size of the error vector $\mathbf{e}$, a grid norm is used in place of the standard vector norm. The following are the definitions of the norms used.

1. max-norm $\left\|\mathbf{e}^h\right\|_{\max} = \max_j |e_j|$
2. 1-norm $\left\|\mathbf{e}^h\right\|_1 = h\sum_{j=1}^{N} |e_j|$
3. 2-norm $\left\|\mathbf{e}^h\right\|_2 = \sqrt{h\sum_{j=1}^{N} |e_j|^2}$

Description of method used in Refinement study

The goal of the refinement study is to check that the scheme selected is stable. Since the scheme selected is consistent (it has an LTE of order $O\left(h^2\right)$ therefore $\|\tau\| \to 0$ as $h \to 0$), this implies that the scheme will be stable if it can be shown that the numerical solution converges to the exact solution[2].

For a stable scheme the following relation must hold

$$\|e\| \leq \left\|A^{-1}\right\| \|\tau\|$$

therefore stability can be established by verifying that error norm $\|e\|$ is also of order $O\left(h^2\right)$. This implies that $\|A^{-1}\|$ is $O\left(1\right)$. In addition, showing that $\|e\|$ is of order $O\left(h^2\right)$ implies that $\|e\| \to 0$ as $h \to 0$, which means convergence.

These are the steps carried out in the refinement study

1. The system $AU = f$ is formulated based on the scheme described above. These equations contain h (the grid space) in them as a free parameter. Initially h is given an initial starting value.
2. $A\mathbf{U} = \mathbf{f}$ is solved for $\mathbf{U}$.
3. The total error vector $\mathbf{e} = \mathbf{U} - \mathbf{u}$ is calculated and the error grid norm $\|e\|$ found using the the above definitions of norms.
4. The spacing h is divided by 2 and the above steps are repeated. The number of iterations was selected so that numerical convergence can be clearly observed. It is found that 5 or 6 iterations was sufficient to show convergence in this problem.

[2]Lax-Richtmyer theorem

5. The error table and the log plots are generated. Convergence is verified by showing from the error table results that the total error norm $\|e\|$ has an order of accuracy of $O(h^2)$. This implies the numerical scheme selected is stable.

Results of the refinement study

This is a plot of $\log(h)$ against the log of the various error norms. The source code is in the appendix.

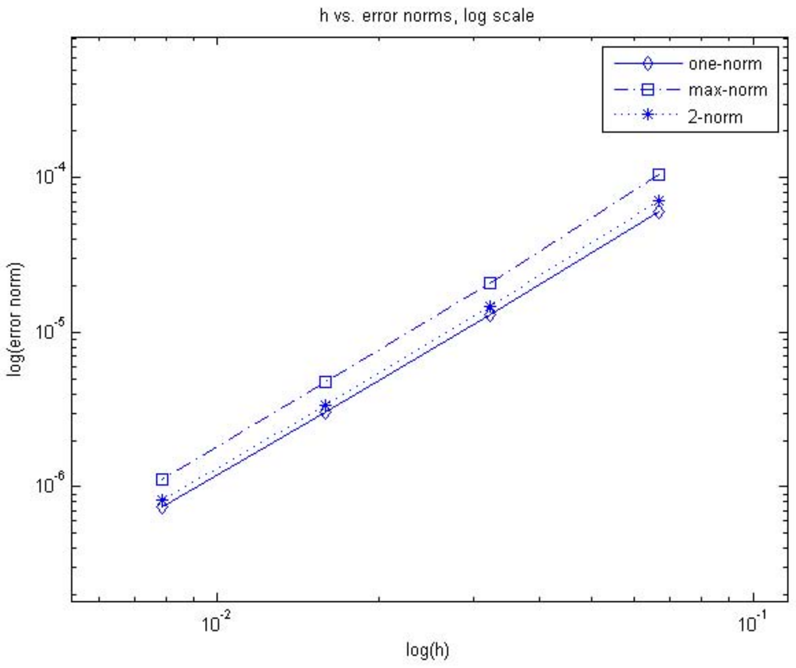


Figure 4: matlab HW2 part a logplot

The error table is the following

```
EDU>> nma_HW2_part_a
N           h        emax       ratio          e1       ratio          e2       ratio
16   6.6667e-002  1.0444e-004 0.0000e+000 5.9816e-005 0.0000e+000 7.0865e-005 0.0000e+000
32   3.2258e-002  2.0983e-005 4.9772e+000 1.3040e-005 4.5870e+000 1.4799e-005 4.7886e+000
64   1.5873e-002  4.7448e-006 4.4223e+000 3.0536e-006 4.2706e+000 3.4030e-006 4.3488e+000
128  7.8740e-003  1.1299e-006 4.1992e+000 7.3937e-007 4.1300e+000 8.1708e-007 4.1648e+000
256  3.9216e-003  2.7583e-007 4.0964e+000 1.8194e-007 4.0637e+000 2.0025e-007 4.0802e+000
512  1.9569e-003  6.8147e-008 4.0476e+000 4.5130e-008 4.0316e+000 4.9573e-008 4.0396e+000
1024 9.7752e-004  1.6937e-008 4.0236e+000 1.1238e-008 4.0157e+000 1.2333e-008 4.0196e+000
2048 4.8852e-004  4.2220e-009 4.0115e+000 2.8042e-009 4.0076e+000 3.0758e-009 4.0096e+000
```

Conclusion

The study was carried out using 3 different norms (max norm,1-norm and 2-norm). It is found that for each norm the total error ratio converged to 4, implying $p = 2$, hence the total error norm was $O\left(h^2\right)$ the same as LTE. This shows that the numerical scheme is stable and the numerical solution converges to the exact solution.

2.2 Part (b)

Solve $u_{xx} = 2\cos^2(\pi x)$ with $u_x(0) = 0, u_x(1) = 1$

2.3 Finding analytical solution

The existence of a solution is first verified. This is done in this case since for Neumann boundary conditions a solution might not even exist if the problem is not well posed. Integrating both sides of the PDE gives

$$\begin{aligned}\int_0^1 u_{xx}(x)\,dx &= \int_0^1 2\cos^2(\pi x)\,dx\\ &= u_x(1) - u_x(0)\\ &= 1\end{aligned}$$

Substituting the values of $u_x$on the boundaries, the above becomes zero, hence the problem is well posed. Now the analytical solution is found.

Integrating the differential equation once

$$u_x = x + \frac{\sin(2\pi x)}{2\pi} + c_1 \tag{1}$$

And Integrating again

$$u(x) = \frac{x^2}{2} - \frac{\cos(2\pi x)}{4\pi^2} + c_1 x + c_2 \tag{2}$$

Substituting the boundary condition $u_x(0) = 0$ in the above[3] gives $c_1 = 0$, therefore the solution is

$$u(x) = \frac{x^2}{2} - \frac{\cos(2\pi x)}{4\pi^2} + c_2 \tag{3}$$

This solution is not unique. The constant c_2 is arbitrary and an infinite number of solutions exist. A solution exist which is up to an arbitrary additive constant. To select a constant for the purpose of the numerical analysis in the refinement study, the constant is selected to give the solution zero mean. Hence

$$\begin{aligned}\int_0^1 u(x)\,dx &= 0\\ \int_0^1 \left(\frac{x^2}{2} - \frac{\cos(2\pi x)}{4\pi^2} + c_2\right)dx &= 0\\ \frac{1}{6} + c_2 &= 0\end{aligned}$$

Therefore $c_2 = -\frac{1}{6}$. Therefore, the exact solution used in the refinement study to compare

[3]The other condition $u_x(1) = 1$ could also have been used, but the result would remain the same

the numerical solution against is

$$u\left(x\right)=\frac{x^{2}}{2}-\frac{\cos\left(2\pi x\right)}{4\pi^{2}}-\frac{1}{6}$$

Scheme to use for the numerical solution

Two schemes formulated, both having a local truncation error $\|\tau\|$ of order $O\left(h^{2}\right)$. Hence both schemes are consistent.

First scheme The standard 3 point centered difference formula$\frac{U_{j-1}-2U_{j}+U_{j+1}}{h^{2}}$ is used for approximating u_{xx} on internal grid points. However in this problem U_{o} and $U_{N+1}$are not known at the left most and the right most grid points (grid points of index 0 and $N+1$), therefore the grid points used includes these 2 points in addition to the standard internal grid points $(1\cdots N)$ resulting in the A matrix having size $N+2$. (A will have 2 additional rows as compared to part (a)).

Now, two imaginary points are introduced into the domain, one to the left of $j=0$, and one to the right of $j=N+1$

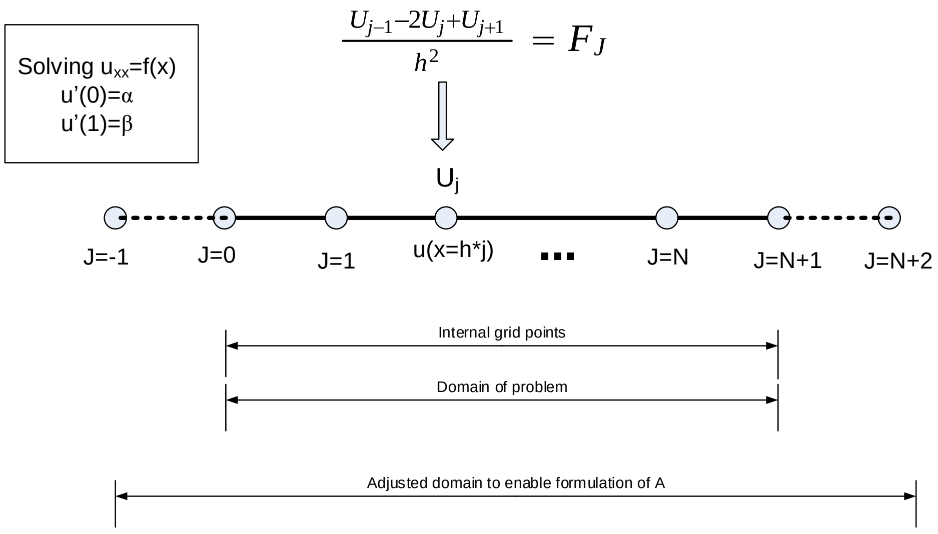


Figure 5: problem 1 part b scheme

For $j=0$ the standard 3 points formulate is used to approximate u_{xx}

$$\frac{U_{-1}-2U_{0}+U_{1}}{h^{2}}=f_{0} \tag{1}$$

And $u_{x}\left(0\right)=\alpha$ the 2 points central difference scheme is used

$$\frac{U_{1}-U_{-1}}{2h}=\alpha \tag{2}$$

Now U_{-1} is eliminated using (1) and (2). From equation (2)

$$U_{-1}=U_{1}-2h\alpha$$

substituting the above into (1)

$$\begin{aligned}\frac{(U_1-2h\alpha)-2U_0+U_1}{h^2}&=f_0\\ \frac{-2U_0+2U_1}{h^2}&=f_0+\frac{2\alpha}{h}\qquad(3)\end{aligned}$$

The above equation (3) is the discrete equation for node $j=0$ (the first row in the A matrix). Similarly for the right-most node $j=N+1$, u_{xx} is approximated using

$$\frac{U_N-2U_{N+1}+U_{N+2}}{h^2}=f_{N+1}\qquad(4)$$

And $u_x(1)=\beta$ at node $j=N+1$ is approximated using 2 points central difference

$$\frac{U_{N+2}-U_N}{2h}=\beta\qquad(5)$$

$U_{N+2}$is eliminated using (4) and (5). From equation (5)

$$U_{N+2}=U_N+2h\beta$$

substituting the above into (4)

$$\begin{aligned}\frac{U_N-2U_{N+1}+(U_N+2h\beta)}{h^2}&=f_{N+1}\\ \frac{2U_N-2U_{N+1}}{h^2}&=f_{N+1}-\frac{2\beta}{h}\qquad(6)\end{aligned}$$

The above equation (6) is the discrete equation for node $j=N+1$ (the last row in the A matrix). $Au=f$ is now setup resulting in

$$\frac{1}{h^2}\begin{pmatrix}-2&2&0&0&0&0&0\\1&-2&1&0&0&0&0\\0&1&-2&1&0&0&0\\0&0&1&-2&1&0&0\\0&0&0&\cdot&\cdot&\cdot&0\\0&0&0&0&1&-2&1\\0&0&0&0&0&2&-2\end{pmatrix}\begin{pmatrix}U_0\\U_1\\U_2\\ \cdot\\ \cdot\\N_{N-1}\\U_N\\U_{N+1}\end{pmatrix}=\begin{pmatrix}f_0+\frac{2\alpha}{h}\\f_1\\f_2\\ \cdot\\ \cdot\\f_N\\f_{N+1}-\frac{2\beta}{h}\end{pmatrix}$$

The A matrix is tridiagonal, not symmetrical and singular since $null(A)=\mathbf{1}^T$. In some books, this matrix is called the Laplacian matrix.

Second scheme In this scheme, the first order derivative for the Neumann boundary condition at the left point and at the right point is approximated using

$$\begin{aligned}\alpha&=\frac{1}{h}\left(\frac{3}{2}U_0-2U_1+\frac{1}{2}U_2\right)\\ \beta&=\frac{1}{h}\left(\frac{3}{2}U_{N+1}-2U_N+\frac{1}{2}U_{N-1}\right)\end{aligned}$$

respectively.

The derivation of the above formulas is shown in example 1.2 in the textbook. The above approximation has an LTE of $O\left(h^2\right)$. For the other internal grid points, the standard 3 points centred difference $\frac{U_{j-1}-2U_j+U_{j+1}}{h^2}$ is used to approximate u_{xx}. Using the above approximations, the system $AU=f$ becomes

$$\frac{1}{h^2}\begin{pmatrix} \frac{3}{2} & -2 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0\\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & \cdot & \cdot & \cdot & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1\\ 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & -2 & \frac{3}{2} \end{pmatrix}\begin{pmatrix} U_0\\ U_1\\ U_2\\ \cdot\\ \cdot\\ U_{N-1}\\ U_N\\ U_{N+1} \end{pmatrix} = \begin{pmatrix} \frac{\alpha}{h}\\ F_1\\ F_2\\ \cdot\\ \cdot\\ \cdot\\ F_N\\ \frac{\beta}{h} \end{pmatrix}$$

The A matrix is tridiagonal, not symmetrical and is also singular since $null(A)=\mathbf{1}^T$.

Augmenting the systems before solving The A matrix for both schemes above is singular. Both have $u=(1,1,1,\cdots,1,1)^T$ as the null vector. Using Matlab to verify, the null command can be used as follows

```
EDU>> A=nma_FDM_matrix_laplace_1D_Neumann_scheme_1(6)
A =
-2     2     0     0     0     0
1    -2     1     0     0     0
0     1    -2     1     0     0
0     0     1    -2     1     0
0     0     0     1    -2     1
0     0     0     0     2    -2
DU>>  null(A,'r')'
ans =
1     1     1     1     1     1
```

For the second scheme

```
EDU>> A=nma_FDM_matrix_laplace_1D_Neumann_scheme_2(6)
A =
1.5000   -2.0000    0.5000         0         0         0
1.0000   -2.0000    1.0000         0         0         0
0         1.0000   -2.0000    1.0000         0         0
0         0         1.0000   -2.0000    1.0000         0
0         0         0         1.0000    -2.0000    1.0000
0         0         0         0.5000    -2.0000    1.5000
EDU>>  null(A,'r')'
ans =
1     1     1     1     1     1
```

Since the matrix A is singular, before solving by Gaussian elimination it is augmented as follows

$$A \to \begin{pmatrix} A & v \\ u & 0 \end{pmatrix}$$

where v is the the null of A^* (the adjoint of A, or for a real matrix, the same as A^T) and u is the null of A.

v is found for the above 2 schemes. For the first scheme:

```
EDU>> A=nma_FDM_matrix_laplace_1D_Neumann_scheme_1(6);
EDU>> null(A','r')'
ans =
1     2     2     2     2     1
```

And for the second scheme

```
EDU>> A=nma_FDM_matrix_laplace_1D_Neumann_scheme_2(6);
EDU>> null(A','r')'
ans =
1.0000   -1.5000   -1.0000   -1.0000   -1.5000    1.0000
```

Now that v and u are found, the augmented A can be formulated.

The force vector is also augmented as follows

$$f \to \begin{pmatrix} f \\ 0 \end{pmatrix} \tag{7}$$

The augmented system is now solved for U using Gaussian elimination. In Matlab

$$U = A\backslash f$$

The resulting solution U will have an extra element at the end, which is not used in the solution. This element, called λ was verified to be equal $\frac{\mathbf{v}\cdot\mathbf{f}}{\mathbf{v}\cdot\mathbf{v}}$.

Notice that last element of f was set to 0 in (7), this is because it corresponds to having selected an exact solution with a zero mean. When the last element of f is set to 0, this corresponds to having $\sum_{i=0}^{N+1} U_i = 0$, which implies the mean of the numerical solution is zero. This follows because $null(A)$ was found to be $\mathbf{1}^T$, meaning that, from looking at

$\begin{pmatrix} A & v \\ \mathbf{1}^T & 0 \end{pmatrix} \begin{pmatrix} U \\ \lambda \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}$, the *dot* $\left(\mathbf{1}^T, U\right) = 0$, which is the same as saying that $\sum_{i=0}^{N+1} U_i = 0$ or the mean of the numerical solution is zero.

To summarize: A constant in the analytical solution was earlier selected which resulted in the mean of the exact solution to be zero. This constant was found to be $\frac{-1}{6}$. The last entry in the augment f vector is set to be zero to cause the numerical solution to have zero mean as well.

Now that the numerical solution is found, the error vector $e = U - u$ is found and its norms are calculated to complete the refinement study. Below is the result for both schemes described above.

Refinement study results

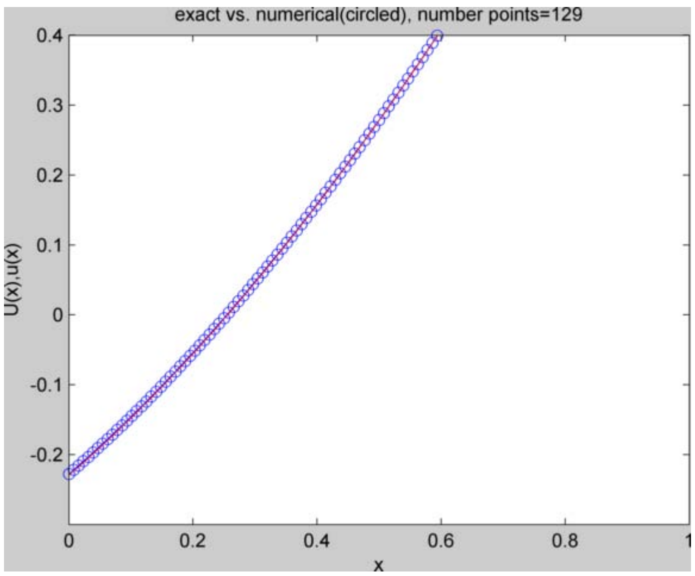


Figure 6: nma HW2 prob1 part b solution

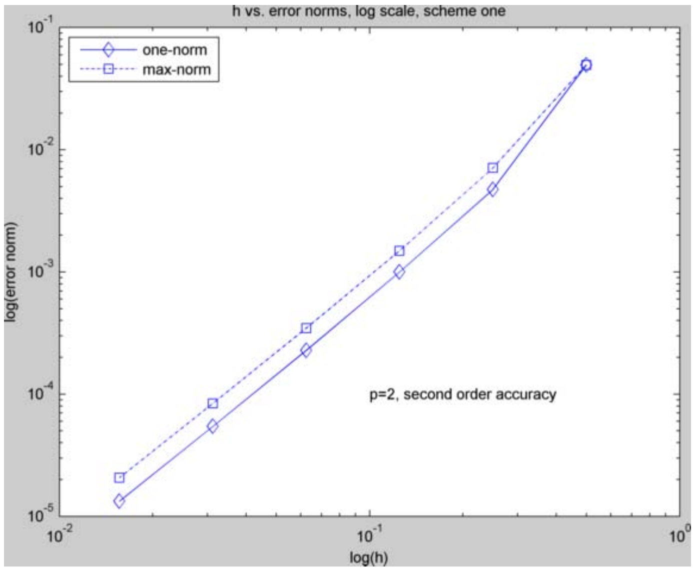


Figure 7: nma HW2 part b scheme one plot2

Scheme one The error table is the following:

```
EDU>> nma_HW2_part_b_scheme_one
N           h      emax       ratio          e1       ratio
3    0.5000000 4.9560e-002  0.0000e+000 4.9560e-002 0.0000e+000
5    0.2500000 7.1036e-003  6.9766e+000 4.7358e-003 1.0465e+001
9    0.1250000 1.4925e-003  4.7596e+000 9.9728e-004 4.7487e+000
17   0.0625000 3.4734e-004  4.2969e+000 2.2786e-004 4.3768e+000
33   0.0312500 8.4008e-005  4.1346e+000 5.4367e-005 4.1912e+000
65   0.0156250 2.0668e-005  4.0646e+000 1.3271e-005 4.0967e+000
```

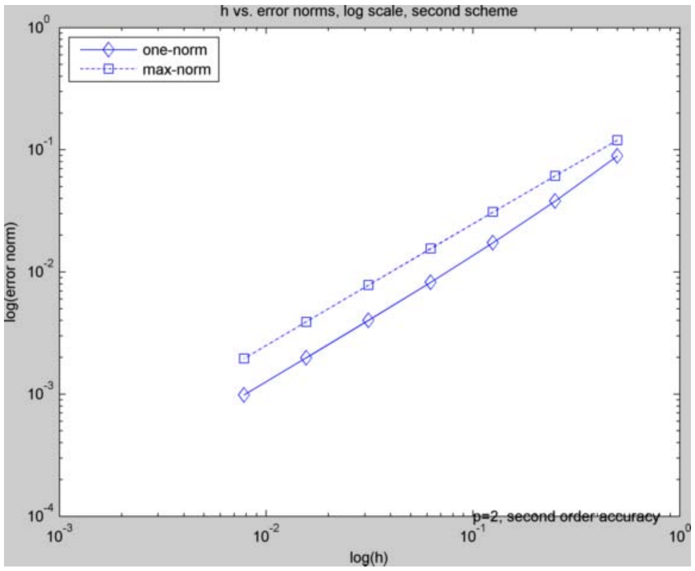


Figure 8: HW2 prob1 part b scheme 2

Scheme 2 And the error table is

```

EDU>> nma_HW2_prob1_part_b(2)
N           h         emax        ratio           e1        ratio
3      0.5000000 5.9960e-003 0.0000e+000 5.9960e-003 0.0000e+000
5      0.2500000 6.0976e-003 9.8333e-001 6.0976e-003 9.8333e-001
9      0.1250000 8.1787e-004 7.4554e+000 4.3938e-004 1.3878e+001
17     0.0625000 1.6346e-004 5.0034e+000 8.9562e-005 4.9058e+000
33     0.0312500 6.0030e-005 2.7230e+000 3.6203e-005 2.4739e+000
65     0.0156250 1.7602e-005 3.4103e+000 1.0956e-005 3.3043e+000
129    0.0078125 4.7387e-006 3.7146e+000 2.9860e-006 3.6693e+000
257    0.0039063 1.2278e-006 3.8594e+000 7.7785e-007 3.8388e+000
513    0.0019531 3.1240e-007 3.9302e+000 1.9841e-007 3.9204e+000
1025   0.0009766 7.8786e-008 3.9652e+000 5.0098e-008 3.9605e+000
```

Conclusion

Two different schemes were used for solving part(b). Both schemes had an LTE of $O\left(h^2\right)$, therefore they both were consistent numerical schemes. The result shows that $\|e\|$ had $O\left(h^2\right)$ implying $\|A^{-1}\|$ had order $O\left(1\right)$ and that both scheme were stable.

It is observed that the second scheme required more iterations to coverage to the exact solution compared to the first scheme. Attempt was made to find if any error was made in the derivation of the scheme or in the code, but none found. Therefore, this showed that when comparing 2 schemes with the same theoretical order of accuracy, this does not necessarily imply they will have the same exact performance on the same numerical example. Therefore, before selecting a scheme, it would be useful to always run a number of numerical tests to compare schemes against each others on different numerical test problems.

3 Problem 2

Propose a discretization scheme for $u_{xx} = f$ with $u_x(0) - \alpha u(0) = g$ and $u(1) = b$

Answer:

This problem has Robin boundary conditions on the left side and Dirichlet boundary conditions on the right side. The condition for existence of a solution implies

$$u_x(1) - u_x(0) = \int_0^1 f(x)\, dx$$

Substituting the boundary conditions into the above results in

$$-\alpha u(0) - g = \int_0^1 f(x)\, dx$$

Hence any $f(x)$ which satisfies the above will make the problem a well posed one. In the following, 3 different schemes are presented using different methods of approximating the Robin boundary conditions. The last scheme resulted in an A matrix with the most desirable properties being tridiagonal and symmetric.

3.1 First scheme

$u_x(0)$ is approximated by 2 points centered difference by introducing an imaginary point U_{-1} resulting in

$$u_x(0) = \frac{U_{-1} - U_1}{2h}$$

Substituting the above in the Robin boundary condition gives

$$\frac{U_{-1} - U_1}{2h} - \alpha U_0 = g \tag{1}$$

$u_{xx}(0)$ is approximated using the standard 3 point formula

$$\frac{U_{-1} - 2U_0 + U_1}{h^2} = f_0 \tag{2}$$

U_{-1} is now eliminated From (1) and (2). From equation (1)

$$U_{-1} = 2h(g + \alpha U_0) + U_1$$

Substituting into (2)

$$\frac{[2h(g + \alpha U_0) + U_1] + 2U_0 + U_1}{h^2} = f_0$$

And simplifying

$$\frac{2U_0\left(1+\alpha h\right)+2U_1}{h^2}=f_0-\frac{2g}{h}$$

This will be the equation to use at node $j=0$ which is the first row in the A matrix.

For node $j=N$, $u_{xx}\left(N\right)$ is approximated using the standard 3 point formula

$$\frac{U_{N-1}-2U_N+U_{N+1}}{h^2}=f_N$$

Since $U_{N+1}=b$, the above becomes

$$\frac{U_{N-1}+2U_N}{h^2}=f_N-\frac{b}{h^2}$$

Which is the equation for node $j=N$. For all the remaining internal grid point, the standard 3 point formula is used to approximate u_{xx}. Therefore, the system which now contains $N+1$ equations is completed

$$\frac{1}{h^2}\begin{pmatrix} 2\left(1+\alpha h\right) & 2 & 0 & 0 & 0 & 0 & 0 & 0\\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & \cdot & \cdot & \cdot & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1\\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 \end{pmatrix}\begin{pmatrix} U_0\\ U_1\\ U_2\\ \cdot\\ \cdot\\ \cdot\\ U_{N-1}\\ U_N \end{pmatrix}=\begin{pmatrix} f_0-\frac{2g}{h}\\ f_1\\ f_2\\ \cdot\\ \cdot\\ \cdot\\ f_{N-1}\\ f_N-\frac{b}{h^2} \end{pmatrix}$$

3.2 Second scheme

Approximating $u_x\left(0\right)$ by 3 points forward difference

$$u_x\left(0\right)=\frac{1}{h}\left(\frac{3}{2}U_0-2U_1+\frac{1}{2}U_2\right)$$

Substituting the above in the Robin boundary condition gives

$$\frac{1}{h}\left(\frac{3}{2}U_0-2U_1+\frac{1}{2}U_2\right)-\alpha U_0=g$$

$$U_0\left(\frac{3}{2}-\alpha h\right)-2U_1+\frac{1}{2}U_2=hg$$

Dividing both sides by h^2 to allow writing the matrix A with a common $\frac{1}{h^2}$ factored out

$$\frac{U_0\left(\frac{3}{2}-\alpha h\right)-2U_1+\frac{1}{2}U_2}{h^2}=\frac{g}{h}$$

For the right side, the same scheme is used as in the first scheme above

$$\frac{U_{N-1}+2U_N}{h^2}=f_N-\frac{b}{h^2}$$

And the system becomes

$$\frac{1}{h^2}\begin{pmatrix} \left(\frac{3}{2}-\alpha h\right) & -2 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \cdot & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdot & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdot & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 \end{pmatrix}\begin{pmatrix} U_0 \\ U_1 \\ U_2 \\ \cdot \\ \cdot \\ \cdot \\ U_{N-1} \\ U_N \end{pmatrix} = \begin{pmatrix} \frac{g}{h} \\ f_1 \\ f_2 \\ \cdot \\ \cdot \\ \cdot \\ f_{N-1} \\ f_N - \frac{b}{h^2} \end{pmatrix}$$

3.3 Third scheme

Approximating $u_x(0)$ by 2 points forward difference which has an LTE of $O(h)$

$$u_x(0) = \frac{1}{h}(U_1 - U_0)$$

Substituting the above in the Robin boundary condition gives

$$\frac{1}{h}(U_1 - U_0) - \alpha U_0 = g$$
$$U_1 - U_0 - \alpha h U_0 = hg$$
$$U_0(-1-\alpha h) + U_1 = hg$$

Dividing both sides by h^2 to allow writing the matrix A with a common $\frac{1}{h^2}$ factored out

$$\frac{[U_0(-1-\alpha h) + U_1]}{h^2} = \frac{g}{h}$$

For the right side, the same scheme was used as in the first scheme above, resulting in

$$\frac{1}{h^2}\begin{pmatrix} (-1-\alpha h) & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 \end{pmatrix}\begin{pmatrix} U_0 \\ U_1 \\ U_2 \\ \cdot \\ \cdot \\ \cdot \\ U_{N-1} \\ U_N \end{pmatrix} = \begin{pmatrix} \frac{g}{h} \\ f_1 \\ f_2 \\ \cdot \\ \cdot \\ \cdot \\ f_{N-1} \\ f_N - \frac{b}{h^2} \end{pmatrix}$$

This scheme has an LTE of $O(h)$ since an $O(h)$ approximation was used for $u_x(0)$

3.4 Order of accuracy

Each of the above 3 schemes is a consistent scheme because for the first 2 schemes the LTE is $O(h^2)$ and for the third scheme the LTE is $O(h)$. A consistent scheme is one in which $\|\tau\| \to 0$ as $h \to 0$. It expected that the final error norm $\|e\|$ will behave as $O(h^2)$

when using the first 2 scheme and as $O(h)$ when using the third scheme. This is provided A is stable. Meaning that $\|A^{-1}\|$ is of order $O(1)$.

Considering the third scheme only since it is symmetric, and using the grid 2-norm, then given that

$$\begin{aligned}\|e\|_2 &= \left\|A^{-1}\tau\right\|_2 \\ &\leq \left\|A^{-1}\right\|_2 \|\tau\|_2\end{aligned}$$

And since for a symmetric matrix $\|A^{-1}\|_2 = \rho(A^{-1}) = \frac{1}{|\lambda_{\min}|}$ where $|\lambda_{\min}|$ is the smallest eigenvalue of A in absolute value, then showing that $|\lambda_{\min}| \geq 1$ would imply that $\|e\|_2 \leq \|\tau\|_2$.

There two analytical methods for finding the $\lambda_{\min}$ of A. By solving the eigenvalue problem $L\phi = \lambda\phi$ for the given boundary conditions or by using matrix algebra to solve for the eigenvalue of A directly. Attempt are made at both of these methods, but more time is needed to complete an analytical proof.

Therefore, to answer the question for this problem, a numerical method was used instead, where a refinement study was done using the third scheme found above in an attempt to show numerically that $\|e\|$ is of the same order as $\|\tau\|$ as expected. A well formed problem is constructed and used for the purpose of the numerical analysis part. The results and the conclusion follows.

3.5 Refinement study for the third scheme

The following numerical problem was select. $u_{xx} = f$ with $u_x(0) - \alpha u(0) = g$ and $u(1) = b$ with the following parameter values

$$\begin{aligned}\alpha &= 1 \\ g &= 1 \\ b &= 1 \\ f &= \cos(x)\end{aligned}$$

This problem is well posed, and has the following analytical solution

$$u(x) = \frac{1}{2}(1 + b - g - x + b\ x + g\ x + \cos(1) + x\ \cos(1) - 2\cos(x))$$

And it satisfies the necessary condition $-\alpha u(0) - g = \int_0^1 f(x)\,dx$

The following are the results of the refinement study. The source code is in the appendix.

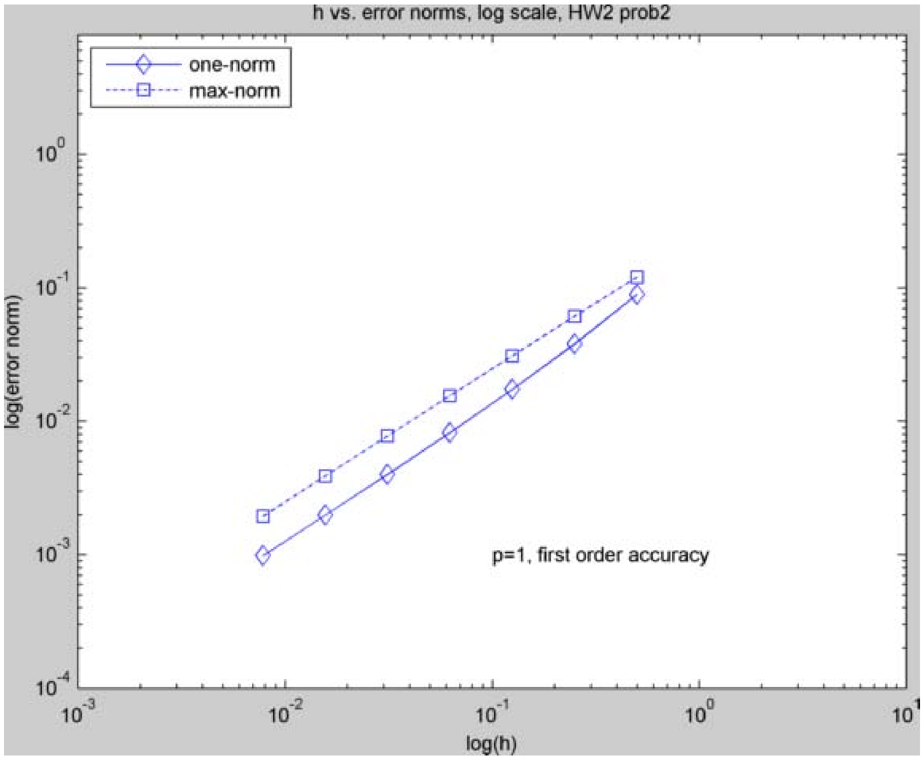


Figure 9: HW2 prob2

```
\begin{X301}
EDU>> close all; nma_HW2('hw2_prob2')
N           h        emax       ratio          e1         ratio
3      0.5000000  1.2015e-001  0.0000e+000  8.8980e-002  0.0000e+000
5      0.2500000  6.1299e-002  1.9601e+000  3.7962e-002  2.3439e+000
9      0.1250000  3.0950e-002  1.9806e+000  1.7318e-002  2.1921e+000
17     0.0625000  1.5550e-002  1.9904e+000  8.2379e-003  2.1023e+000
33     0.0312500  7.7938e-003  1.9952e+000  4.0129e-003  2.0529e+000
65     0.0156250  3.9016e-003  1.9976e+000  1.9798e-003  2.0269e+000
129    0.0078125  1.9520e-003  1.9988e+000  9.8324e-004  2.0136e+000
```

3.6 Conclusion

A scheme was implemented for Robin boundary conditions. LTE used was $O(h)$ since this specific scheme generated an A matrix with desirable form (symmetrical and tridiagonal) . The expectation of total error using this scheme was to be $O(h)$. This expectation was confirmed by doing a refinement study on the selected test problem. This result showed that the scheme is stable and therefore we conclude that $\|A\|$ is of order $O(1)$ based on this test case.

4 Problem 3

4.1 Part (a)

From definition

$$\|\mathbf{e}\|_\infty = -\|B\tau\|_\infty \tag{1}$$

Where $B = A^{-1}$ was generated by the use of Green function as described on page 27 of the text book. Writing the product of the matrix B with the vector τ as

$$B\tau = b_1\tau_1 + b_1\tau_1 + \cdots + b_N\ \tau_N$$

where $\mathbf{b}_j$ is the j^{th} column of matrix B, equation (1) becomes (all norms are to be taken as the max-norm, hence for clarify, the ∞ is dropped in what follows)

$$\begin{aligned}\|e\| &= -\|b_1\tau_1 + b_1\tau_1 + \cdots + b_N\tau_N\| \\ &\leq -\left(\|b_1\tau_1\| + \|b_2\tau_2\| + \cdots + \|b_N\tau_N\|\right) \\ &\leq -\left(\|b_1\|\,|\tau_1| + \|b_2\|\,|\tau_2| + \cdots + \|b_N\|\,|\tau_N|\right) \end{aligned}\tag{3}$$

$\|\mathbf{b}_j\|$ is the maximum element in the j^{th} column of the matrix B. These elements are located along the diagonal of B. Hence $\|\mathbf{b}_j\| = |B_{jj}|$ and (3) becomes

$$\|\mathbf{e}\| \leq -\left(|B_{11}|\ \ |\tau_1| + |B_{22}|\ \ |\tau_2| + \cdots + |B_{NN}|\ \ |\tau_N|\right) \tag{4}$$

From equation 2.46 in the textbook,

$$B_{ij} = \begin{cases} h(x_j-1)x_i & i = 1,2,\cdots j \\ h(x_i-1)x_j & i = j, j+1, \cdots, N \end{cases} \tag{5}$$

To answer the question, assume that $|\tau_1| = O(h^p)$, and assume for the moment, for generality, that the LTE at all the other points was different, hence for other points let $|\tau_j| = O(h^q)$. Equation (4) becomes

$$\|\mathbf{e}\| \leq -O(h^p)\,|B_{11}|\ - O(h^q)\sum_{k=2}^{N}|B_{kk}| \tag{6}$$

From (5), it is found that at point $j=1$, $|B_{11}| = h(h-1)h = h^3 - h^2 = O(h^2)$, while at the internal points, that is, at points near the middle, B_{jj} is approximated by $O(h)$, hence (6) becomes

$$\|\mathbf{e}\| \leq -O(h^p)\,O\left(h^2\right)\ - O(h^q)\sum_{k=2}^{N}O(h)$$

But $\sum_{k=2}^{N}O(h) \sim (N-1)\times O(h) = O(1)$ since $N = \frac{1}{h}$, hence the above becomes

$$\begin{aligned}\|\mathbf{e}\| &\leq -O(h^p)\,O\left(h^2\right)\ - O(h^q) \\ &= -O\left(h^{p+2}\right) - O(h^q)\end{aligned} \tag{7}$$

Therefore, if $q = p$, i.e. if the LTE was the same at each grid point, including the first, the above simplifies to

$$\begin{aligned}\|\mathbf{e}\| &\leq -O\left(h^{p+2}\right)-O\left(h^{p}\right)\\ &\sim O\left(h^{p}\right)\end{aligned}$$

Hence, having the LTE at the edge grid point be $O\left(h^{p}\right)$ resulted in $\|\mathbf{e}\|_{\infty}$ having an order of accuracy $O\left(h^{p}\right)$ which is the expected. The smallest value of p that can be used for the edge point while still giving overall $\|\mathbf{e}\|=O\left(h^{2}\right)$ can be found from (7)

$$\|\mathbf{e}\|=O\left(h^{2}\right)=-O\left(h^{p+2}\right)-O\left(h^{q}\right)$$

Setting $p=0$ and $q=2$ results in

$$\begin{aligned}\|\mathbf{e}\| &= -O\left(h^{2}\right)-O\left(h^{2}\right)\\ &\sim O\left(h^{2}\right)\end{aligned}$$

Hence $p=0$ or $O(1)$, is possible for the LTE at the edge while arriving at an overall $\|\mathbf{e}\|\sim O\left(h^{2}\right)$.

4.2 Part (b)

From equation 2.46 in the textbook,

$$B_{ij}=\begin{cases} h(x_j-1)x_i & i=1,2,\cdots j\\ h(x_i-1)x_j & i=j,j+1,\cdots,N\end{cases} \tag{5}$$

To answer the question, assume $\left|\tau_{j=N/2}\right|=O\left(h^{p}\right)$, where j is the middle point, i.e. at $x_j=\frac{1}{2}$and assume for other points $\left|\tau_j\right|=O\left(h^{q}\right)$, hence starting from

$$\|\mathbf{e}\|\leq-\left(\left|B_{11}\right|\ \left|\tau_1\right|+\left|B_{22}\right|\ \left|\tau_2\right|+\cdots+\left|B_{NN}\right|\ \left|\tau_N\right|\right)$$

which was derived in part(a), then the above equation becomes

$$\|\mathbf{e}\|\leq-O\left(h^{q}\right)\sum_{\substack{k=1\\k\neq N/2}}^{N}\left|B_{kk}\right|-O\left(h^{p}\right)\left|B_{N/2,N/2}\right| \tag{8}$$

At the middle point from (5), $\left|B_{N/2,N/2}\right|$ can be found to be $h\left(\frac{1}{2}-1\right)\frac{1}{2}=\frac{-1}{4}h\sim O(h)$, while for the points away from the middle, $\left|B_{jj}\right|$ is approximated by $O\left(h^{2}\right)$, hence (8) becomes

$$\|\mathbf{e}\|\leq-O\left(h^{q}\right)\left[(N-1)\times O\left(h^{2}\right)\right]\ -O\left(h^{p}\right)O(h)$$

But $(N-1)\times O\left(h^{2}\right)$ is $O(h)$ since $N=\frac{1}{h}$, hence the above becomes

$$\begin{aligned}\|\mathbf{e}\| &\leq -O\left(h^{q}\right)O(h)\ -O\left(h^{p+1}\right)\\ &= -O\left(h^{q+1}\right)-O\left(h^{p+1}\right)\end{aligned} \tag{9}$$

Therefore, if $q = p$, i.e. if the LTE was the same at each grid point, including the middle, then (9) becomes

$$\begin{aligned}\|\mathbf{e}\| &= -O\left(h^{p+1}\right) - O\left(h^{p+1}\right)\\ &= O\left(h^{p+1}\right)\end{aligned}$$

Hence, having the LTE at the middle grid point be $O\left(h^{p}\right)$ resulted in $\|\mathbf{e}\|_{\infty}$ having an order of accuracy $O\left(h^{p+1}\right)$ which is higher than the expected $O\left(h^{p}\right)$. The smallest value of p that can be used while still giving $\|\mathbf{e}\| = O\left(h^{2}\right)$ can be found from (9)

$$\|\mathbf{e}\| = O\left(h^{2}\right) = -O\left(h^{q+1}\right) - O\left(h^{p+1}\right)$$

Setting $p = 1$ and $q = 1$ results in $\|\mathbf{e}\| \sim O\left(h^{2}\right)$.

Hence $p = 1$ or $O\left(h\right)$ is possible for the LTE at the middle point while still resulting in an overall $\|\mathbf{e}\| = O\left(h^{2}\right)$

4.3 Part (c)

To perform the numerical tests to verify the above conclusion, a numerical problem is used with a scheme which had an LTE of $O\left(h^{2}\right)$. The LTE at the edge point only was then modified to test part(a) and the LTE at the middle point only was modified to test part(b).

Refinement study was done to verify that the order of accuracy remained $O\left(h^{2}\right)$ after each modification.

For the numerical test, problem 1, part (a) was used for this test. That problem had an LTE of $O\left(h^{2}\right)$ at each grid point, and the standard 3-point central difference was used for approximating u_{xx}.

To modify the LTE at a specific grid point, given that the LTE is defined as

$$\tau_j = \frac{u(j-1) - 2u\left(j\right) + u\left(j+1\right)}{h^2} - f_j$$

Where u above is the exact solution sampled at the given grid points, then to force the LTE to be $O\left(h\right)$ in the middle, the above equation was modified for $x_j = \frac{1}{2}$(middle row of the system) to become

$$\tau_j = \frac{u(j-1) - 2u\left(j\right) + u\left(j+1\right)}{h^2} - f_j + h$$

And to force the LTE to be $O\left(1\right)$ at the first grid point, the LTE is modified for that point only (i.e. $x_j = h$, the first row of the system) to become

$$\tau_j = \frac{u(j-1) - 2u\left(j\right) + u\left(j+1\right)}{h^2} - f_j + 1$$

The refinement study which was done for problem 1, part a, is now repeated with the above modifications. The following are the results generated.

Part (A) refinement

Before making the LTE modification for the first point, the error table generated was

```
EDU>> close all; nma_HW2('hw2_prob1_parta')
N          h        emax       ratio          e1        ratio
3    0.5000000  4.3295e-003  0.0000e+000  2.1647e-003  0.0000e+000
5    0.2500000  1.0925e-003  3.9628e+000  6.8494e-004  3.1605e+000
9    0.1250000  2.7377e-004  3.9906e+000  1.8036e-004  3.7977e+000
17   0.0625000  6.8827e-005  3.9776e+000  4.5662e-005  3.9499e+000
33   0.0312500  1.7234e-005  3.9937e+000  1.1451e-005  3.9875e+000
65   0.0156250  4.3098e-006  3.9987e+000  2.8650e-006  3.9969e+000
129  0.0078125  1.0776e-006  3.9995e+000  7.1640e-007  3.9992e+000
```

One can see that $\|e\| = O\left(h^2\right)$ as the ratio is 4. Now the f vector was modified as described above, and here is the small code segment how this was done. Complete code listing is at the end.

```
f      = force(xcoordinates)';
f(1)   = f(1) - alpha/h^2 +1 ;
f(end) = f(end)-beta/h^2;
```

And now the program was run again

```
EDU>> close all; nma_HW2('hw2_prob3_1')
N          h        emax       ratio          e1         ratio
3     0.5000000  1.2067e-001  0.0000e+000  6.0335e-002  0.0000e+000
5     0.2500000  4.6119e-002  2.6165e+000  2.2753e-002  2.6518e+000
9     0.1250000  1.3566e-002  3.3997e+000  6.6556e-003  3.4186e+000
17    0.0625000  3.6481e-003  3.7185e+000  1.7854e-003  3.7278e+000
33    0.0312500  9.4426e-004  3.8635e+000  4.6157e-004  3.8681e+000
65    0.0156250  2.4010e-004  3.9328e+000  1.1730e-004  3.9350e+000
129   0.0078125  6.0530e-005  3.9666e+000  2.9563e-005  3.9678e+000
```

Examining the second table above shows that $\|e\|_\infty = O\left(h^2\right)$, since the ratio of the error norm still converged to 4, implying $p = 2$ for $\|e\|$. To compare the loglog plots, here are plots before and after the modification is made to the LTE.

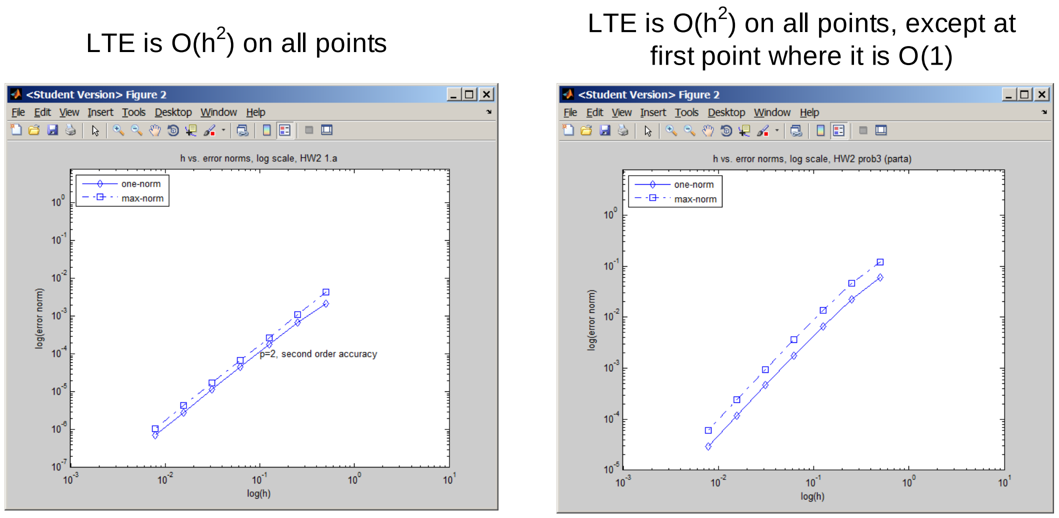


Figure 10: prob3 part a log

Part(b) refinement study

Before making the LTE modification for the middle point, the error table generated was

```
EDU>> close all; nma_HW2('hw2_prob1_parta')
N          h          emax        ratio          e1         ratio
3    0.5000000  4.3295e-003  0.0000e+000  2.1647e-003  0.0000e+000
5    0.2500000  1.0925e-003  3.9628e+000  6.8494e-004  3.1605e+000
9    0.1250000  2.7377e-004  3.9906e+000  1.8036e-004  3.7977e+000
17   0.0625000  6.8827e-005  3.9776e+000  4.5662e-005  3.9499e+000
33   0.0312500  1.7234e-005  3.9937e+000  1.1451e-005  3.9875e+000
65   0.0156250  4.3098e-006  3.9987e+000  2.8650e-006  3.9969e+000
129  0.0078125  1.0776e-006  3.9995e+000  7.1640e-007  3.9992e+000
```

One can see that $\|e\| = O\left(h^2\right)$ because the ratio is 4. Now the f vector was modified as described above, to force the middle point to have an LTE of $O\left(h\right)$, here is the small code segment showing the modification.

```
f                     = force(xcoordinates)';
f(1)                  = f(1)-alpha/h^2;
middle_position       = round(length(f)/2);
f(middle_position)    = f(middle_position) + h;
f(end)                = f(end)-beta/h^2;
```

And now the program was run again

```
EDU>> close all; nma_HW2('hw2_prob3_2')
N           h         emax       ratio          e1       ratio
3    0.5000000 5.8171e-002 0.0000e+000 2.9085e-002 0.0000e+000
5    0.2500000 1.4532e-002 4.0028e+000 7.1276e-003 4.0807e+000
9    0.1250000 3.6325e-003 4.0007e+000 1.7728e-003 4.0206e+000
17   0.0625000 9.0808e-004 4.0002e+000 4.4262e-004 4.0052e+000
33   0.0312500 2.2702e-004 4.0000e+000 1.1062e-004 4.0013e+000
65   0.0156250 5.6754e-005 4.0000e+000 2.7653e-005 4.0003e+000
129  0.0078125 1.4189e-005 4.0000e+000 6.9130e-006 4.0001e+000
```

Examining the second table above, shows that $\|e\|_\infty = O\left(h^2\right)$, since the ratio of the error norm still converged to 4, implying $p = 2$ for $\|e\|$. To compare the loglog plots, here are plots before and after the modification is made to the LTE.

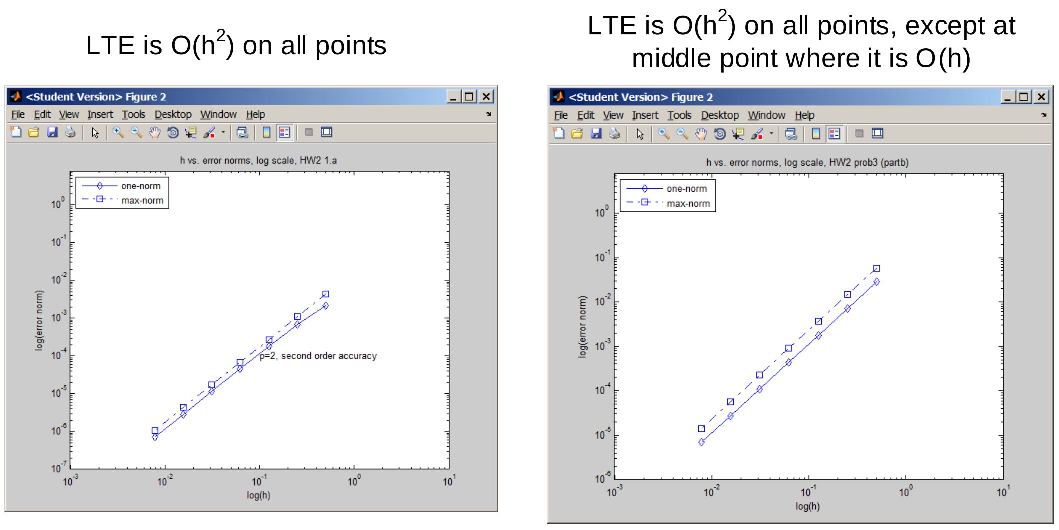


Figure 11: prob3 part blog

This concludes the refinement study verifying results from part(a) and part(c)

5 Source code written for this HW

```
%---------------------------------------------------------------
% HW 2  Math 228A, UC Davis, Fall 2010
% by Nasser M. Abbasi
%
% This .m files contains the main functions needed to solve HW2
%
% To Run:
%                  From Matlab, type nma_HW2( problem_name )
```

```
%
% where problem_name is a string indicating which problem to run the
% refinment study for. The strings are
%
%         hw2_prob1_parta
%         hw2_prob1_partb_scheme_1
%         hw2_prob1_partb_scheme_2
%         hw2_prob2
%         hw2_prob3_1
%         hw2_prob3_2
%
% EXAMPLE
%
%          nma_HW2( 'hw2_prob1_parta' )
%
% make sure this m file is in your Matlab path.
%
function  nma_HW2()
close all;
problems={'hw2_prob1_parta';
    'hw2_prob1_partb_scheme_1';
    'hw2_prob1_partb_scheme_2';
    'hw2_prob2';
    'hw2_prob3_1';
    'hw2_prob3_2'};
for i=1:2 %length(problems)
    nma_HW2_sub( problems(i) )
end
fpritnf('completed...\n');

end

%-------------------------------------------------------
function  nma_HW2_sub( ID )

if nargin ~= 1
    error('one argument is required');
end

N      = 7;                               %number of iterations,
points = arrayfun( @(i) (2^i)+1 ,1:N );   % find number of grid points
h      = arrayfun( @(i) 1/(2^i) ,1:N );   % and corresponding spacings
data   = zeros(N,6);                      %allocate space for error table

%
% Main loop. For each number of points, find error norms and ratios
%
```

```
for i = 1:N

    if strcmp(ID,'hw2_prob1_partb_scheme_1')
        [e,U,u,x] = error_vector_prob1_partb_scheme1(points(i),h(i));
    elseif strcmp(ID,'hw2_prob1_partb_scheme_2')
        [e,U,u,x] = error_vector_HW2_prob1_partb_scheme2(points(i),h(i));
    elseif strcmp(ID,'hw2_prob1_parta')
        [e,U,u,x] = error_vector_HW2_prob1_parta(points(i),h(i));
    elseif strcmp(ID,'hw2_prob2')
        [e,U,u,x] = error_vector_HW2_prob2(points(i),h(i));
    elseif strcmp(ID,'hw2_prob3_1')
        [e,U,u,x] = error_vector_HW2_prob3_1(points(i),h(i));
    elseif strcmp(ID,'hw2_prob3_2')
        [e,U,u,x] = error_vector_HW2_prob3_2(points(i),h(i));
    else
        error('invalid problem name');
    end

    % the columns of data are arranged in this order
    %       npoints  h   e-max  ratio   e-1    ratio
    %          (1)   (2)    (3)   (4)    (5)     (6)

    data(i,1) = points(i);                     % number total points
    data(i,2) = h(i);
    data(i,3) = norm(e,inf);                   % e-max
    data(i,5) = h(i)*norm(e,1);                % e-1

    if i>1
        data(i,4) = data(i-1,3)/data(i,3);    %e-max ratio
        data(i,6) = data(i-1,5)/data(i,5);    %e-1 ratio
    end

    %plot exact vs. approximate solution to verification
    plot(x,U(:),'-o',x,u(:),'r');
    title(sprintf('exact vs. numerical(circled), number points=%d', ...
        points(i)));
    axis([0 1 -.3 .4]);
    xlabel('x'); ylabel('U(x),u(x)');
    drawnow;
    pause(1); % to enable animation to be seen
end

analysis_of_result(data,ID);          %generate error table/plots
end

%------------------------------------------------------------------
% This function process the error table, formating it.
% It accept the e-norm found at each spacing, the vector with the
% corresponding h spacing value, calculates the error ratio
% and print the result to the screen
%
function analysis_of_result(data,ID)
titles ={'N','h','emax','ratio','e1','ratio'};
fms    ={'d','.7f','.4e','.4e','.4e','.4e'};

wid       = 12;
fileID    = 1;
nma_format_matrix(titles,data,wid,fms,fileID,false);

figure;
set(0,'defaultaxesfontsize',8) ;
set(0,'defaulttextfontsize',8);

loglog(data(:,2),data(:,5),'-d'); hold on;
loglog(data(:,2),data(:,3),'-.s');

xlabel('log(h)'); ylabel('log(error norm)');
legend('one-norm','max-norm','Location','NorthWest');
axis equal;
grid off;

% plot the correct title based on which problem is being solved
if strcmp(ID,'hw2_prob1_partb_scheme_1')
    text(10^-1,10^-4,'p=2, second order accuracy')
    title('h vs. error norms, log scale, HW2 1.b scheme1');
    export_fig nma_HW2_prob1_part_b_scheme_1.png
elseif strcmp(ID,'hw2_prob1_partb_scheme_2')
    text(10^-1,10^-4,'p=2, second order accuracy')
    title('h vs. error norms, log scale, HW2 1.b scheme2');
    export_fig nma_HW2_prob1_part_b_scheme_2.png
elseif strcmp(ID,'hw2_prob1_parta')
    text(10^-1,10^-4,'p=2, second order accuracy')
    title('h vs. error norms, log scale, HW2 1.a');
    export_fig nma_HW2_prob1_part_a.png
elseif strcmp(ID,'hw2_prob2')
    text(10^-1,10^-3,'p=1, first order accuracy')
    title('h vs. error norms, log scale, HW2 prob2');
    export_fig nma_HW2_prob2.png
elseif strcmp(ID,'hw2_prob3_1')
    title('h vs. error norms, log scale, HW2 prob3 (parta)');
    export_fig nma_HW2_prob3_1.png
elseif strcmp(ID,'hw2_prob3_2')
    title('h vs. error norms, log scale, HW2 prob3 (partb)');
    export_fig nma_HW2_prob3_2.png
```

```
end

end

%-----------------------------------------------------------------
% Augment the A matrix
%
function [A,f] = augment_system(A,f)
f = f(:);
v = null(A');    % Null vector of adjoint of A
u = null(A);     % Null vector of A

%lambda = dot(v,f)/dot(v,v);  % used during testing to verify it is correct

% Build the A matrix the f matrix to force zero mean for the solution
A = [A    v;
    u'   0];

f = [f ;
    0];
end

%----------------------------------------------------------
% this function builds A matrix, f vector, solves for U, the
% approximate solution, and determines the error vector e=U-u where
% u is the exact solution at the grid points
%
% For HW2, prob 1 part b, second scheme
%
function [e,U,u,xcoordinates] = ...
    error_vector_HW2_prob1_partb_scheme2(total_number_of_points,h)

    function f = force(x)
        f = 2*(cos(pi*x)).^2;
    end

    function u = exactU(x)
        % NOTE: This solution has a -1/6 as its additive constant.
        u = (x.^2)/2 - cos(2*pi*x)/(4*pi^2) - 1/6;
    end

% Boundary conditions, left and right
alpha = 0;
beta  = 1;
len   = 1;

xcoordinates = 0:h:len;                        %xcoordinates at points
```

```

%  build the A matrix
A = nma_FDM_matrix_laplace_1D_Neumann_scheme_2(total_number_of_points);
A = A/h^2;

% build the f vector, make sure to adjust the first and last entries
f      = force(xcoordinates)';
f(1)   = alpha/h;
f(end) = beta/h;

[A,f] = augment_system(A,f);
U = A\f(:);                          % solve for U, the approximate solution
%lambda = U(end);                    % save lambda for reporting
U = U(1:end-1);                      % throw away the last entry, lambda

u = exactU( xcoordinates );          % find exact solution at the grid points

% U found above has a zero mean. This was by construction. Therefore, we
% have to shift it by the current mean of the sampled version of the exact
% solution which also has a zero mean, but only in the limit. Meaning as
% the number of grid points becomes very large.
% Therfore, at each iteration the mean of u will not yet be zero due to
% having small number of samples (grid points). The current mean of u must
% be used to adjust U to make the comparison between U and u applicable.
% When the number of grid points become very large, this adjustment become
% less important, since mean(u) will start to approach zero, and adding
% a zero to U will not affect the result.

U = U + mean(u);

% Now that we have found U, the numerical solution, we find the error
% vector.

u = u(:);  U=U(:);
e = U - u;
end  % end function(error_vector_HW2_prob1_partb_scheme2)

%-----------------------------------------------------------------
% this function builds A matrix, f vector, solves for U, the approximate
% solution, and determines the error vector e=U-u where u is the exact
% solution at the grid points
%
% For HW2, prob 1 part b, first scheme
function [e,U,u,xcoordinates]=error_vector_prob1_partb_scheme1...
    (total_number_of_points,h)

    function f = force(x)
```

```
        f = cos(pi*x).^2;
        f = 2*f;
    end

    function u = exactU(x)
        u = (x.^2)/2 - cos(2*pi*x)/(4*pi^2) - 1/6;
    end

% Boundary conditions, left and right
alpha = 0;
beta  = 1;
len   = 1;

xcoordinates = 0:h:len;    %xcoordinates at points

A = nma_FDM_matrix_laplace_1D_Neumann_scheme_1(total_number_of_points);
A = A/h^2;

% build the f vector, make sure to adjust the first and last entries
f      = force(xcoordinates);
f(1)   = f(1)   + 2*alpha/h;
f(end) = f(end) - 2*beta/h;

USE_PINV = false;   % for testing to verify the augmented method against

if USE_PINV
    U   = pinv(A)*f(:);
    lambda = 0;
else
    [A,f]  = augment_system(A,f);
    U      = A\f(:);           % solve for U, the approximate solution
    lambda = U(end);           % save lambda for reporting
    U      = U(1:end-1);       % Not needed for finding error norm
end

u = exactU( xcoordinates );    % find exact solution at the grid points
U = U + mean(u);               % see note above

% Now that we have found U, the numerical solution, we find the error
% vector.

u = u(:);  U=U(:);
e = U - u;
end % end function(error_vector_prob1_partb_scheme1)

%----------------------------------------------------------------------
% this function builds A matrix, f vector, solves for U, the approximate
% solution, and determines the error vector e=U-u where u is the exact
% solution at the grid points
% HW2, prob 1, part a
%
function [e,U,u,xcoordinates]=error_vector_HW2_prob1_parta...
    (total_number_of_points,h)

    function f = force(x)
        f = exp(x);
    end

    function u = exactU(x)
        u = -1 + exp(x) + 2.*x - exp(1).*x;
    end

% Boundary conditions, left and right
alpha = 0;
beta  = 1;
len   = 1;

number_internal_points = total_number_of_points-2;
xcoordinates           = h:h:len-h;       %xcoordinates at internal points

A = nma_FDM_matrix_laplace_1D_dirichlet(number_internal_points );
A = A/h^2;

% build the f vector, make sure to adjust the first and last entries
f      = force(xcoordinates)';
f(1)   = f(1)-alpha/h^2;
f(end) = f(end)-beta/h^2;

U = A\f;                          % solve for U, the approximate solution
u = exactU( xcoordinates )';      % find exact solution at the grid points
e = U - u;                        % find the total error vector e

end

%----------------------------------------------------------
% this function builds A matrix, f vector, solves for U, the approximate
% solution, and determines the error vector e=U-u where u is the exact
% solution at the grid points
% HW2 prob2
%
function [e,U,u,xcoordinates] = error_vector_HW2_prob2...
    (total_number_of_points,h)

    function f = force(x)
```

```
        f = cos(x);
    end

    function u = exactU(x,b,g)
        u = (1/2)*(1+b-g-x+b*x+g*x+cos(1)+cos(1)*x-2*cos(x));
    end

% Boundary conditions, left and right
alpha = 1;
b     = 1;
g     = 1;
len   = 1;
xcoordinates = 0:h:len-h;                      %xcoordinates at points

%  build the A matrix
A = nma_FDM_matrix_laplace_1D_robin(total_number_of_points-1,h,alpha);
A = A/h^2;

% build the f vector, make sure to adjust the first and last entries
f      = force(xcoordinates)';
f(1)   = g/h;
f(end) = f(end) - b/h^2;

U = A\f(:);                    % solve for U, the approximate solution
u = exactU( xcoordinates,b,g );  % find exact solution at the grid points
u = u(:);  U=U(:);
e = U - u;
end

%-------------------------------------------------------------------
% this function builds A matrix, f vector, solves for U, the approximate
% solution, and determines the error vector e=U-u where u is the exact
% solution at the grid points
% HW2, prob 1, part a
%
function [e,U,u,xcoordinates]=error_vector_HW2_prob3_1...
    (total_number_of_points,h)

    function f = force(x)
        f = exp(x);
    end

    function u = exactU(x)
        u = -1 + exp(x) + 2.*x - exp(1).*x;
    end

% Boundary conditions, left and right
```

```
alpha = 0;
beta  = 1;
len   = 1;

number_internal_points = total_number_of_points-2;
xcoordinates           = h:h:len-h;       %xcoordinates at internal points

A = nma_FDM_matrix_laplace_1D_dirichlet(number_internal_points );
A = A/h^2;

% build the f vector, make sure to adjust the first and last entries
f      = force(xcoordinates)';
f(1)   = f(1) - alpha/h^2 +1 ;      % make LTE O(1) instead of O(h^2)
f(end) = f(end)-beta/h^2;

U = A\f;                          % solve for U, the approximate solution
u = exactU( xcoordinates )';      % find exact solution at the grid points
e = U - u;                        % find the total error vector e

end

%----------------------------------------------------------------------
% this function builds A matrix, f vector, solves for U, the approximate
% solution, and determines the error vector e=U-u where u is the exact
% solution at the grid points
% HW2, prob 1, part a
%
function [e,U,u,xcoordinates]=error_vector_HW2_prob3_2...
    (total_number_of_points,h)

    function f = force(x)
        f = exp(x);
    end

    function u = exactU(x)
        u = -1 + exp(x) + 2.*x - exp(1).*x;
    end

% Boundary conditions, left and right
alpha = 0;
beta  = 1;
len   = 1;

number_internal_points = total_number_of_points-2;
xcoordinates           = h:h:len-h;       %xcoordinates at internal points

A = nma_FDM_matrix_laplace_1D_dirichlet(number_internal_points );
```

```
A = A/h^2;

% build the f vector, make sure to adjust the first and last entries
f                   = force(xcoordinates)';
f(1)                = f(1)-alpha/h^2;
middle_position     = round(length(f)/2);
f(middle_position) = f(middle_position) + h; %Change LTE from O(h^2)to O(h)
f(end)              = f(end)-beta/h^2;

U = A\f;                           % solve for U, the approximate solution
u = exactU( xcoordinates )';       % find exact solution at the grid points
e = U - u;                         % find the total error vector e

end
```

```
function A = nma_FDM_matrix_laplace_1D_dirichlet(N )
%%
% nma_FDM_matrix_laplace_1D_dirichlet(N)
%
% returns the A matrix, which is the system finite difference matrix for
% numerical solution of 1-D laplace equation Uxx = f, with dirichlet
% boundary conditions on both sides of the element. Based on 3 point
% scheme    U'' =  U(j-1)-2*U(j)+U(j+1)
% notice that spacing h between the nodes is not used. The caller must
% divide by h^2 this A matrix abone return.
%
% INPUT:  N, the number of nodes
% OUTPUT: A, the matrix , see below
%
% EXAMPLE:
% A = nma_FDM_matrix_laplace_1D_dirichlet(6)
%
%     -2     1     0     0     0     0
%      1    -2     1     0     0     0
%      0     1    -2     1     0     0
%      0     0     1    -2     1     0
%      0     0     0     1    -2     1
%      0     0     0     0     1    -2
%
% A = A/h^2;  % h is space between points
% U = A\f;  % solve for U, where Uxx=f, need to set f as function before.
%

A = zeros(N);

for i=1:N
    for j=1:N
        if(i==j)
```

```
            A(i,j) = -2;
        else
            if( i==j+1 || i==j-1 )
                A(i,j) = 1;
            end
        end
    end
end

end
```

```
function  A = nma_FDM_matrix_laplace_1D_Neumann_scheme_1(N )
%-----------------------------------------------------------------
% nma_FDM_matrix_laplace_1D_Neumann_scheme_1(N )
%
% returns the A matrix, which is the system finite difference matrix for
% numerical solution of 1-D laplace equation Uxx = f, with nuemman
% boundary conditions on both sides of the element. Based on 3 point
% scheme    U'' =  U(j-1)-2*U(j)+U(j+1)  for middle points and based on
% scheme  ((-2*U(0)+2*U(1))/(h^2))=f(0)+((2*a)/h) for the left most point,
% where a = U'(0), and h is the spacing. and for for the right most point
% ((2*U(N)-2*U(N+1))/(h^2)) = f(N+1)-((2*b)/h) where b=U'(1)
%
% notice that spacing h between the nodes is not used. The caller must
% divide by h^2 this A matrix abone return.
%
% INPUT:  N, the number of nodes
% OUTPUT: A, the matrix , see below
%
% EXAMPLE:
% A = nma_FDM_matrix_laplace_1D_Neumann_scheme_1(5)
%     -2     2     0     0     0
%      1    -2     1     0     0
%      0     1    -2     1     0
%      0     0     1    -2     1
%      0     0     0     2    -2
%
%
% A = A/h^2;  % h is space between points
% U = A\f;  % solve for U, where Uxx=f, need to set f as function before.
%
% copyright: Nasser M. Abbasi
% 10/18/2010

A = zeros(N);
for i=1:N
    for j=1:N
        if(i==j)
```

```
            A(i,j) = -2;
        else
            if( i==j+1 || i==j-1 )
                A(i,j) = 1;
            end
        end
    end
end

%fix A above for Von Neumann B.C. only 1st and last rows need fixed
A(1,2)       = 2;
A(end,end-1) = 2;
end
```

```
function A = nma_FDM_matrix_laplace_1D_Neumann_scheme_2(N)
%-------------------------------------------------------------------
% A = nma_FDM_matrix_laplace_1D_Neumann_scheme_2(N)
%
% returns the A matrix, which is the system finite difference matrix for
% numerical solution of 1-D laplace equation Uxx = f, with nuemman
% boundary conditions on both sides of the element. Based on 3 point
% scheme    U'' =  U(j-1)-2*U(j)+U(j+1)  for middle points and the left
% and right most points, the following scheme is used
%    In this scheme, the Neumann boundary condition is approximated using
%
%       a =(1/h)((3/2)U(0)-2*U(1)+(1/2)U(2))
%       b =(1/h)((3/2)U(N+1)-2*U(N)+(1/2)U(N+1))
%
% where a=U'(0) and b=U'(1)
%
% notice that spacing h between the nodes is not used. The caller must
% divide by h^2 this A matrix abone return.
%
% INPUT:  N, the number of nodes
% OUTPUT: A, the matrix , see below
%
% EXAMPLE:
% A = nma_FDM_matrix_laplace_1D_Neumann_scheme_2(5)
%
%     1.5000   -2.0000    0.5000         0         0
%     1.0000   -2.0000    1.0000         0         0
%          0    1.0000   -2.0000    1.0000         0
%          0         0    1.0000   -2.0000    1.0000
%          0         0    0.5000   -2.0000    1.5000
%
%
% A = A/h^2;  % h is space between points
% U = A\f;  % solve for U, where Uxx=f, need to set f as function before.
```

```
%
% copyright: Nasser M. Abbasi
% 10/18/2010

A = zeros(N);
for i=1:N
    for j=1:N
        % as before, part(a), Dirichlet  B.C.
        if(i==j)
            A(i,j) = -2;
        else
            if( i==j+1 || i==j-1 )
                A(i,j) = 1;
            end
        end
    end
end

%fix A above for Von Neumann B.C. only 1st and last rows need fixed
A(1,1) = 3/2;
A(1,2) = -2;
A(1,3) = 1/2;

A(end,end)   = (3/2);
A(end,end-1) =  -2;
A(end,end-2) = (1/2);

end
```

```
function A = nma_FDM_matrix_laplace_1D_robin(N,h,alpha)
%-------------------------------------------------------------------
% A = nma_FDM_matrix_laplace_1D_robin(N,h)
%
% returns the A matrix, which is the system finite difference matrix for
% numerical solution of 1-D laplace equation Uxx = f, with nuemman
% boundary conditions on both sides of the element. Based on 3 point
%
% INPUT:  N, the number of nodes
%         h, spacing between nodes
%
% OUTPUT: A, the matrix , see below
%
%
% A = A/h^2;  % h is space between points
% U = A\f;  % solve for U, where Uxx=f, need to set f as function before.
%
% copyright: Nasser M. Abbasi
% 10/18/2010
```

```

A = zeros(N);
for i=1:N
    for j=1:N
        % as before, part(a), Dirichlet  B.C.
        if(i==j)
            A(i,j) = -2;
        else
            if( i==j+1 || i==j-1 )
                A(i,j) = 1;
            end
        end
    end
end

%fix A above
A(1,1) = -1-alpha*h;

end
```

```
function [hd,bdy] =  nma_format_matrix(headings,data,wid,fms,fid,flag)
%   nma_format_matrix()
%
%   prints matrix of numerical data with headings in formatted way
%
%   INPUT:
%   headings: a cell array of strings for the headings of each column
%   data: a matrix, contains the numerical data
%   wid: how wide a field to use (see below for example)
%   fms: the formating cell string array to use for the numerical data
%        each string corresponds to formatting a field in the numberical
%        data
%   fid: the file id to print to. Use 1 for stdout
%   flag: if 0, then will not print anything, just format and return
%         result
%
%   Examples calling
%
%   titles={'number of projects','sales','profit'};
%   data=[2 rand(1) rand(1);
%         3 rand(1) rand(1)];
%
%   Suppose we want to format each numerical number above as %16.5E, then
%   in this case 10 will be the field width, and '.5E' is what to use for
%   the fms argument
%
%   wid    = 16;
%   fms    = {'d','.4E','.5E'};
%   fileID = 1;
%   nma_format_matrix(titles,data,wid,fms,fileID,false);
%
%   version 1.0  10/15/2010
%   All blame to Nasser M. Abbasi
%   Thanks for help from matlab newsgroup: Ross W, Bruno Luong
%

%
% do some basic checking on input
%
if nargin ~= 6
    error 'number of arguments must be 6'
end

if ~iscell(headings)
    error 'headings must be cell array'
end

if ~iscell(fms)
    error 'fms must be cell array'
end

if isempty(headings)
    error 'headings can not be empty';
end

if isempty(data)
    error 'data can not be empty';
end

if isempty(fms)
    error 'fms can not be empty';
end

[nRowH,nColH] = size(headings);
if(nRowH>1)
    error 'headings can not have more than one row';
end

[nRowFms,nColFms] = size(fms);
if(nRowFms>1)
    error 'fms can not have more than one row';
end

if(nColH ~= nColFms)
    error 'headings must have same number of columns as fms';
```

```
end

[~,nCol] = size(data);
if(nColH ~= nCol)
    error 'data must have same number of columns as headings';
end
%
% end basic checking on input, now do the formating and printing
%

fmt = arrayfun(@(x) ['%',num2str(wid),'s'],1:nCol,'UniformOutput',false);
if flag
   fprintf(fid,[fmt{:} '\n'],headings{:});
end

hd=sprintf([fmt{:} '\n'],headings{:});

fmt = arrayfun(@(x) ['%',num2str(wid),fms{x}],1:nCol,'UniformOutput',false);
if flag
   fprintf(fid,[fmt{:} '\n'],data');
end
bdy=sprintf([fmt{:} '\n'],data');

end
```