MATH 761 - Homework 1 solution

1. Derive a Galerkin finite element formulation for the boundary value problem (BVP)

\[ \frac{d^2 q}{dx^2} - q = f(x), 0 \leq x \leq \pi \]

\[ q(0) = 0, q(\pi) = 0. \]

Use a uniform discretization \( x_j = jh, h = \pi/n, \) of the interval \([0, \pi]\), piecewise linear elements (form or basis functions)

\[
l_j(x) = \begin{cases} 
0 & x < x_{j-1} \\
\frac{x-x_{j-1}}{h} & x_{j-1} \leq x < x_j \\
\frac{x_{j+1}-x}{h} & x_j \leq x < x_{j+1} \\
0 & x_{j+1} \leq x 
\end{cases}
\]

and use the same functions as weights.

a) The system \( B = \{l_0(x), l_1(x), \ldots, l_n(x)\} \) is a linearly independent set of functions.

Solution. \( B \) is linearly independent if \( \forall x \in [0, \pi], \)

\[ a_0 l_0(x) + \ldots + a_n l_n(x) = 0, \]  

implies \( a_0 = \ldots = a_n = 0. \) Let \( k = [x/h], \) the integer part of \( x/h, \) such that \( x_k \leq x \leq x_{k+1}. \) Since \( l_j(x) = 0 \) for \( j \neq k, k+1, \) (1) reduces to

\[ a_k l_k(x) + a_{k+1} l_{k+1}(x) = 0. \]  

(2)

For \( x_k \leq x \leq x_{k+1}, \) \( l_k(x) = (x_{k+1} - x)/h \) and \( l_{k+1}(x) = (x - x_k)/h \) so (2) becomes

\[ a_k(x_{k+1} - x) + a_{k+1}(x - x_k) = 0. \]  

(3)

Note that the problem has been to reduced to the linear independence of \( \{x_{k+1} - x, x - x_k\} \) on the interval \([x_k, x_{k+1}]\). The Wronskian of these two functions is

\[ W = \begin{vmatrix} x_{k+1} - x & x - x_k \\ -1 & 1 \end{vmatrix} = h \neq 0. \]

Since the Wronskian is nonzero at at least one point (in this case at all points) within \([x_k, x_{k+1}]\) the functions are linearly independent.
b) Expansion of \( g(x), \; g \in C^2([0, \pi]) \) on \( B \).

\textit{Solution.} The term “expansion” is deliberately vague here. A simple idea would be to write

\[
g(x) = \sum_{j=0}^{n} g_j l_j(x) \tag{4}
\]

with \( g_j = g(x_j) \). The hint on Fourier expansions suggests that some further thought might be given to this question. Fourier coefficients are found by computing scalar products of a function \( g(x) \) with each member of the orthogonal basis \( \{1, \sin x, \cos x, \ldots\} \). So the first question to address is how to define a scalar product (which we need to define ‘orthogonality’). Two definitions suggest themselves

\[
< f, g >_1 = \int_{0}^{\pi} f(x) g(x) \, dx, \quad \text{and} \quad < f, g >_2 = \sum_{j=0}^{n} f(x_j) g(x_j).
\]

The first takes into account that the function we wish to expand is of class \( C^2([0, \pi]) \), the second takes into account the discretization adopted in this problem. Consider now what is meant by “expansion”. A sketch is useful. First note that \( C^2 \subset C^1 \subset C^0 \) (in all cases we’re considering the interval \([0, \pi]\)) and that \( l_0, \ldots, l_n \in C^0 \). In the sketch we represent \( C^2 \) by some subspace within the overall \( C^0 \) space. In \( C^0 \), \( l_0, \ldots, l_n \) are a linearly independent set, but other directions, as suggested by \( l_{n+1} \), exist.

The best approximation

\[
\hat{g}(x) = \sum_{j=0}^{n} a_j l_j(x)
\]
of $g$ by a linear combination within the span of $B$ is given by an orthogonal projection. So
the condition to be met is
\[
<g - \tilde{g}, l_k> = 0 \text{ for } k = 0, \ldots, n,
\]
which can be rewritten as
\[
\sum_{j=0}^{n} <l_j, l_k> a_k = <g, l_k>.
\]
For scalar product $<,>_2$ this leads to a diagonal system and we quickly obtain
\[
a_k = g_k = g(x_k).
\]
But for scalar product $<,>_1$ we obtain a full linear system. This discussion is relevant with
regard to the limit to which numerical approximations shall converge as we shall see below.

c) Weighted residual method.

Solution. We introduce the expansion $\tilde{q}(x) = Q_j l_j(x)$ (repeated indices denote summation),
and compute the weighted residual
\[
\int_{0}^{\pi} \left( \frac{d^2}{dx^2}(Q_j l_j(x)) - f(x) \right) l_k(x) dx = 0.
\]
We set $r_k = 0$. The second order differentiation of piecewise linear functions would eliminate
the differential operator altogether; to avoid this, integrate by parts
\[
\int_{0}^{\pi} \frac{d^2}{dx^2}(Q_j l_j(x)) l_k(x) dx = \left[ Q_j \frac{d}{dx}(l_j(x)) l_k(x) \right]_{x=0}^{x=\pi} - Q_j \int_{0}^{\pi} \frac{d}{dx}(l_j(x)) \frac{d}{dx}(l_k(x)) dx.
\]
We obtain the linear system
\[
(B_{kj} - D_{kj} - L_{kj}) Q_j = F_k,
\]
with the notations
\[
B_{kj} = \left[ \frac{d}{dx}(l_j(x)) l_k(x) \right]_{x=0}^{x=\pi}, D_{kj} = \int_{0}^{\pi} \frac{d}{dx}(l_j(x)) \frac{d}{dx}(l_k(x)) dx, L_{kj} = \int_{0}^{\pi} l_j(x) l_k(x) dx,
\]
\[
F_k = \int_{0}^{\pi} f(x) l_k(x) dx.
\]
The behavior at the boundaries must be carefully considered. Let’s take a look at the form
functions $l_0(x), l_1(x), l_2(x)$ as shown in the following plot.

```bash
gnuplot
h=0.1
l(j,x) = (x < (j-1)*h) ? 0 : (x > (j+1)*h) ? 0 : (x < j*h) ? (x-(j-1)*h)/h : ((j+1)*h-x)/h ;
set xrange [0:1]; set yrange [0:1]; set size 1, .5;
set style line 1 lt 1 lc rgb "red" lw 2;
set style line 2 lt 1 lc rgb "green" lw 2;
set style line 3 lt 1 lc rgb "blue" lw 2;
plot l(0,x) ls 1, l(1,x) ls 2, l(2,x) ls 3
```
The functions \( l_0(x), l_1(x) \), have a discontinuous derivative at \( x = 0 \). We have

\[
\begin{align*}
    l_0'(0) &= -\frac{2}{h} H(x), \\
    l_1'(0) &= \frac{1}{h} H(x),
\end{align*}
\]

with \( H(x) \) the Heaviside step function. Likewise at \( x = \pi \),

\[
\begin{align*}
    l_{n-1}'(\pi) &= \frac{1}{h} H(x - \pi), \\
    l_n'(\pi) &= -\frac{2}{h} H(x - \pi).
\end{align*}
\]

The derivatives are discontinuous but bounded. For \( 0 < k < n \) we have \( l_k(0) = l_k(\pi) = 0 \), so

\[
\begin{align*}
    B_{kj} &= \left[ \frac{dl_j(x)}{dx} l_k(x) \right]_{x=0}^{x=\pi} = 0,.
\end{align*}
\]

For \( k = 0 \) we obtain

\[
\begin{align*}
    \left[ Q_j \frac{dl_j(x)}{dx} l_k(x) \right]_{x=0}^{x=\pi} &= Q_0 l_0'(0) + Q_1 l_1'(0) = (-2Q_0 + Q_1) \frac{H(0)}{h},
\end{align*}
\]

\[
\begin{align*}
    B_{00} &= -\frac{2H(0)}{h}, B_{01} = \frac{H(0)}{h}, B_{0j} = 0, j > 1,
\end{align*}
\]

and for \( k = n \) we have

\[
\begin{align*}
    \left[ Q_j \frac{dl_j(x)}{dx} l_k(x) \right]_{x=0}^{x=\pi} &= Q_{n-1} l_{n-1}'(\pi) + Q_n l_n'(\pi) = (Q_{n-1} - 2Q_n) \frac{H(0)}{h},
\end{align*}
\]

\[
\begin{align*}
    B_{n,n-1} &= \frac{H(0)}{h}, B_{nn} = -\frac{2H(0)}{h}, B_{nj} = 0, j < n - 1.
\end{align*}
\]

Clearly the regularization for the Heaviside function will play a role, and this will be investigated in numerical experiments. Also, the meaning of \( Q_j \) comes into play. For the \( <, >_2 \) choice of scalar product, the coefficients of the expansion \( \tilde{q}(x) = Q_j l_j(x) \) are not necessarily nodal values, and we can use the problem boundary conditions to state that \( Q_0 = Q_n = 0 \). For the \( <, >_1 \) scalar product \( Q_j \) are not necessarily nodal values and the choice \( Q_0 = Q_n = 0 \) is not justified.

A sketch is useful in the computation of \( D_{kj} \).

```
GNUplot

h=0.1;

dl(j,x) = (x < (j-1)*h) ? 0 : (x > (j+1)*h) ? 0 : (x < j*h) ? 1/h :
    -1/h;

set xrange [0:1]; set yrange [-2:2]; set size 1, .5;
set style line 1 lt 1 lc rgb "red" lw 2;
set samples 1000;
plot dl(4,x)*dl(5,x)*h**2 ls 1
```
GNUplot] h=0.1;
dl(j,x) = (x < (j-1)*h) ? 0 : (x > (j+1)*h) ? 0 : (x < j*h) ? 1/h : -1/h;
set xrange [0:1]; set yrange [-2:2]; set size 1, .5;
set style line 1 lt 1 lc rgb "red" lw 2;
set samples 1000;
plot dl(5,x)*dl(5,x)*h**2 ls 1
From the above, we can readily deduce

\[ D_{k,k-1} = -\frac{1}{h}, \quad D_{k,k} = \frac{2}{h}, \quad D_{k,k+1} = -\frac{1}{h}, \quad 0 < k < n. \]

At the edge \( k = 0 \) we have

\[ D_{0,0} = \frac{1}{h}, \quad D_{0,1} = -\frac{1}{h}, \]

while at \( k = n \) we have

\[ D_{n,n-1} = -\frac{1}{h}, \quad D_{n,n} = \frac{1}{h}. \]

For \( L_{k,j} \), some help from a symbolic package is welcome. Let’s use Maxima within TeXmacs (see http://arxiv.org/html/cs.SC/0504039 for details on how Maxima is used here)

Maxima.

\[
\begin{align*}
\text{(%i1)} & \quad L_{k,k-1} : \text{expand}\left( \int_{(k-1)h}^{kh} \frac{x - (k-1)h}{h} \frac{kx - x}{h} \, dx \right) \\
\text{(%o18)} & \quad \frac{h}{6} \\
\text{(%i19)} & \quad L_{k,k} : \text{expand}\left( \int_{(k-1)h}^{kh} \left( \frac{x - (k-1)h}{h} \right)^2 \, dx + \int_{kh}^{(k+1)h} \left( \frac{(k+1)h - x}{h} \right)^2 \, dx \right) \\
\text{(%o21)} & \quad \frac{2h}{3} \\
\text{(%i22)} & \quad L_{k,k+1} : \text{expand}\left( \int_{kh}^{(k+1)h} \frac{x - k}{h} \frac{(k+1)x - x}{h} \, dx \right) \\
\text{(%o20)} & \quad \frac{h}{6} \\
\text{(%i21)} & \quad L_{0,0} : \text{expand}\left( \int_0^h \left( \frac{h - x}{h} \right)^2 \, dx \right) \\
\text{(%o23)} & \quad \frac{h}{3} \\
\text{(%i24)} & \quad L_{0,1} : \text{expand}\left( \int_0^h \frac{h - x}{h} \frac{x}{h} \, dx \right) \\
\text{(%o24)} & \quad \frac{h}{6}
\end{align*}
\]
Eigenvalues of these matrices (non-zero elements near the diagonal) are tridiagonal with constant elements along diagonals. We can introduce the stencils of the eigenproblem\[ L \int \]Note that this is the Simpson quadrature approximation for\[ \int \]
In this case the boundary condition matrix \( B \) does not contribute to \( A \). Matrices \( L, D, A \) are tridiagonal with constant elements along diagonals. We can introduce the stencils of these matrices (non-zero elements near the diagonal)\[ L = \text{diag}\left( \begin{bmatrix} h & \frac{2h}{6} & h \end{bmatrix} \right), D = \text{diag}\left( \begin{bmatrix} -\frac{1}{h} & \frac{2}{h} & -\frac{1}{h} \end{bmatrix} \right), \]
\[ A = \text{diag}\left( \begin{bmatrix} \frac{1}{h} & -\frac{h}{6} & -\frac{2h}{6} & -\frac{2h}{3} & \frac{1}{h} & -\frac{h}{6} \end{bmatrix} \right). \]
In this notation the stencil is repeated along the diagonal; elements that move outside the matrix are dropped. If we choose \( <, >_2 \) as the scalar product, \( Q_0, Q_n \) are not known and we now have \( A \in \mathbb{R}^{(n+1) \times (n+1)} \). Lines 1 to \( n-1 \) are the same as in the previous case. Line 0 of \( A \) is now\[ A_{0,0} = -\frac{6H(0) + 3 + h^2}{3h}, A_{0,1} = \frac{6H(0) + 6 - h^2}{6h}, \]
and line \( n \) is\[ A_{n,n-1} = \frac{6H(0) + 6 - h^2}{6h}, A_{n,n} = -\frac{6H(0) + 3 + h^2}{3h}. \]
We shall also need the r.h.s. vector \( F \). If we introduce an expansion of \( f \) on the \( B \) basis (adopt scalar product \( <, >_2 \)) we obtain\[ F_k = L_{kj}f(x_k) = \frac{h}{6}(f_{k-1} + 4f_k + f_{k+1}), 0 < k < n. \]
Note that this is the Simpson quadrature approximation for \( \int_{x_{k-1}}^{x_{k+1}} f(x)dx \) (why?).

d) Eigenvalues of \( A \). The matrix \( A \) is a discretization of the operator \( \mathcal{L} = d^2/dx^2 - 1 \). The eigenproblem \( \mathcal{L}q = \lambda q \) has eigenfunction solutions \( q = \sin(kx) \) with \( k \in \mathbb{Z} \) and associated eigenvalues \( \lambda_k = -1 - k^2 \). We expect the eigenvectors of \( A \) (we assume we’ve chosen \( <, >_2 \)) to be discretizations of \( \sin(kx) \)
\[ r_k = (\sin(kx_1) \sin(kx_2) ... \sin(kx_{n-1}))^T. \]
The \( l^{th} \) line of \( Ar_k = \lambda_k r_k \) is\[ \left( \frac{1}{h} - \frac{h}{6} \right) \sin(k(l-1)h) + \left( \frac{2h}{3} - \frac{2h}{6} \right) \sin(klh) + \left( \frac{1}{h} - \frac{h}{6} \right) \sin(k(l+1)h) = \lambda_k \sin(klh). \]
Expanding $\sin(\alpha \pm \beta)$ we obtain

$$
\lambda_k = -\frac{2h}{3} - \frac{2}{h} + \left(\frac{2}{h} - \frac{h}{3}\right)\cos(kh),
$$

which does not depend on the index $l$. Hence $\sin(kx)$ is indeed an eigenvector (plotted below) with associated eigenvalue $\lambda_k$. The matrix is symmetric negative definite (all eigenvalues are negative). In the limit $h \to 0$ the eigenvalues behave as

$$
\lambda_k = -\frac{2}{h}(1 - \cos(kh)) \to -\infty.
$$

The condition number (spectral norm) is

$$
\kappa(A) = \frac{\max_{1 \leq k \leq n-1} |\lambda_k|}{\min_{1 \leq k \leq n-1} |\lambda_k|} = -\frac{2h}{3} - \frac{2}{h} + \left(\frac{2}{h} - \frac{h}{3}\right)\cos((n-1)h) - \frac{2h}{3} - \frac{2}{h} + \left(\frac{2}{h} - \frac{h}{3}\right)\cos(h).
$$

In the limit of $h \to 0$ the condition number behaves as

$$
\lim_{h \to 0} \kappa(A) = \frac{1}{2}(2 - 2n + n^2) \to \infty.
$$

e) The analytical solution for $f(x) = x(\pi - x)$ is

$$
q(x) = -\frac{e^{-x}}{1 + e^{\pi}}[2(e^x - 1)(e^x - e^{\pi}) + e^x(1 + e^{\pi})x(\pi - x)].
$$
f) Since $A$ is symmetric negative definite the most efficient solution of $-AQ = -F$ is obtained by the Cholesky algorithm. In the Cholesky algorithm we factor the s.p.d. matrix $-A$ as $LL^T$ (we now use the notation $L$ for the factorization of $A$, changing the use from (c)). We need only two vectors of length $n$ to store $-A$ or $L$. Introduce vectors $b, c, d, e \in \mathbb{R}^{n-1}$, such that

$$L = \text{diag}([b, c, 0]), \quad -A = \text{diag}([e, d, e]),$$

$$L = \begin{pmatrix} c_1 & 0 & 0 & \ldots & 0 \\ b_2 & c_2 & 0 & \ldots & 0 \\ 0 & b_3 & c_3 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & c_{n-1} \end{pmatrix}, \quad -A = \begin{pmatrix} d_1 & e_2 & 0 & \ldots & 0 \\ e_2 & d_2 & e_3 & \ldots & 0 \\ 0 & e_3 & d_3 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & d_{n-1} \end{pmatrix}.$$  

Carrying out the multiplication $LL^T$ leads to

$$LL^T = \begin{pmatrix} c_1 & 0 & 0 & \ldots & 0 \\ b_2 & c_2 & 0 & \ldots & 0 \\ 0 & b_3 & c_3 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & b_{n-1} \end{pmatrix} = \begin{pmatrix} c_1b_2 & 0 & \ldots & 0 \\ b_2 & c_2b_3 & 0 & \ldots & 0 \\ 0 & b_3 & c_3 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & b_{n-1} \end{pmatrix} = \begin{pmatrix} d_1 & e_2 & 0 & \ldots & 0 \\ e_2 & d_2 & e_3 & \ldots & 0 \\ 0 & e_3 & d_3 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & d_{n-1} \end{pmatrix} = -A.$$  

Note that the position of the outlying $b_1$ value has also been shown. We can then concisely write the algorithm as

$$c_1 = \sqrt{d_1}$$

for $j=2$ to $n-1$

$$b_j = c_j/c_{j-1} \quad c_j = \sqrt{d_j - b_j^2}$$

Once the factorization has been computed, the solution to $AQ = (LL^T)Q = F$ is found by forward substitution to solve $LZ = F$ and backward substitution to solve $L^TQ = Z$.  

9
\[ Z_1 = \frac{F_1}{c_1} \]
for \( j = 2 \) to \( n-1 \)
\[ Z_j = \frac{(F_j - b_j Z_{j-1})}{c_j} \]

\[ Q_{n-1} = \frac{Z_{n-1}}{c_{n-1}} \]
for \( j = n-2 \) downto 1
\[ Q_j = \frac{(Z_j - b_j Q_{j+1})}{c_j} \]

The following Cholesky Fortran routine returns the solution and the error w.r.t. the exact solution for \( f(x) = x(\pi - x) \).

SUBROUTINE cholesky(n)
IMPLICIT DOUBLE PRECISION (a-h,o-z)
DOUBLE PRECISION q(0:n), err(0:n), enorm(0:2), qnorm(0:2)
DOUBLE PRECISION b(n), c(n), d(n), e(n), fex(n), F(n), qex(n), x(n), Z(n)

! Fill vectors that define A,F
pi = 4.d0*ATAN(1.d0)
epi = EXP(pi)
h = pi/n
Ah0 = -2.d0/h-2.d0*h/3.d0
Ah1 = 1.d0/h-h/6.d0
fex=0.d0
DO k=1,n-1
  xk = k*h
  x(k) = xk
  fex(k) = xk*(pi-xk)
  ex = EXP(xk)
  qex(k) = -1.d0/ex/(1.d0+epi) * &
           (2.d0*(ex-1.d0)*(ex-epi)+ex*(1.d0+epi)*xk*(pi-xk))
  d(k) = -Ah0
  e(k) = -Ah1
END DO
DO k=1,n-1
  F(k) = -h/6.d0*(fex(k-1)+4.d0*fex(k)+fex(k+1))
END DO
! Factor A
C(1) = SQRT(d(1))
DO k=2,n-1
  b(k) = e(k)/c(k-1)
  c(k) = SQRT(d(k)-b(k)**2)
END DO
! Forward substitution
Z(1) = F(1)/c(1)
DO k=2,n-1
  Z(k) = (F(k)-b(k)*Z(k-1))/c(k)
END DO
! Backward substitution
Q(n-1) = Z(n-1)/c(n-1)
DO k=n-2,1,-1
  Q(k) = (Z(k)-b(k+1)*Q(k+1))/c(k)
END DO
! Compute error in 3 norms (inf,1,2)
enorm=0.d0; qnorm=0.d0
DO k=1,n-1
   err(k) = Q(k) - qex(k)
   qnorm(0) = MAX(qnorm(0),ABS(qex(k)))
   qnorm(1) = qnorm(1) + ABS(qex(k))
   qnorm(2) = qnorm(2) + qex(k)**2
   enorm(0) = MAX(enorm(0),ABS(err(k)))
   enorm(1) = enorm(1) + ABS(err(k))
   enorm(2) = enorm(2) + err(k)**2
END DO
enorm(2) = SQRT(enorm(2)); qnorm(2) = SQRT(qnorm(2))
! Compute relative errors
enorm = enorm/qnorm
PRINT '(('A30,3(E10.3,2X)))','Norm inf,1,2 relative errors: ',enorm
! Write the results to file for plotting with GNUPlot
OPEN(UNIT=1,FILE='enorm.dat',STATUS='UNKNOWN',POSITION='APPEND')
WRITE(1,*)LOG10(h),LOG10(enorm)
CLOSE(UNIT=1)
OPEN(UNIT=2,FILE='err.dat',STATUS='REPLACE')
err(0)=0.d0; err(n)=0.d0
DO k=0,n
   WRITE(2,*)k*h,err(k)
END DO
CLOSE(UNIT=2)
END SUBROUTINE cholesky

The routine writes the relative errors of the solution to the file `enorm.dat` appending data from new runs. It writes the x distribution of the error from the latest run to file `err.dat`.

g) It is convenient to call the `cholesky` routine from within a Python script. In this way, we carry out computationally intensive operations (solving the linear system) in compiled code that executes fast, but still have the flexibility of an interpreted language to analyze results. The `f2py` tool (http://www.scipy.org/F2py) is used to automatically generate the links between the `cholesky` routine listed above and the Python environment. Here are the Unix shell commands required to do this.

```
shell] cd MATH761/Fall2008
shell] f2py --quiet --fcompiler=gfortran -c cholesky.f90 -m cholesky
```

Now we can enter Python and import the `cholesky` module. Some preliminary directory path commands are needed for Python from within TeXmacs to know where to find the `cholesky` module.

Python session

```
python] import sys, os

python] os.chdir('/home/mitran/courses/MATH761/Fall2008')

python] sys.path.append(os.getcwd())

python] from cholesky import *
```
The convergence data has been saved in file `enorm.dat`, which we can print out (table headings were edited in).

```
shell] more enorm.dat

| lg h    | lg ||e∞|| | lg ||e1|| | lg ||e2|| |
|---------|--------|--------|--------|
| -1.008001 | -3.4376927 | -3.3765318 | -3.3925106 |
| -1.3090301 | -4.0400295 | -3.9786955 | -3.9948590 |
| -1.6100601 | -4.6421587 | -4.5807814 | -4.5969913  |
| -1.9110901 | -5.2442360 | -5.1828478 | -5.1990693  |
| -2.2121201 | -5.8463006 | -5.7849097 | -5.8011341  |
| -2.5131501 | -6.4483632 | -6.3869714 | -6.4031966  |
```

We can now construct a convergence plot. We use Gnuplot which besides plotting can also carry out least squares fits.

```
GNUPlot] cd "/home/mitran/courses/MATH761/Fall2008"
set title "Convergence plot"
set xlabel "lg(h)"; set ylabel "lg(||e||)"
set grid
err(h) = A + slope*h
fit err(x) "enorm.dat" using 1:2 via A, slope
fit err(x) "enorm.dat" using 1:3 via A, slope
fit err(x) "enorm.dat" using 1:4 via A, slope
plot "enorm.dat" using 1:2 title "inf-norm" w lp, "enorm.dat" using 1:3 title "1-norm", "enorm.dat" using 1:4 title "2-norm" w lp
```
Iteration 0
WSSR : 106.549  delta(WSSR)/WSSR : 0
delta(WSSR) : 0  limit for stopping : 1e-05
lambda : 1.47715

initial set of free parameter values
A = 1
slope = 1
/

Iteration 1
WSSR : 1.39003  delta(WSSR)/WSSR : -75.6521
delta(WSSR) : -105.159  limit for stopping : 1e-05
lambda : 0.147715

resultant parameter values
A = -0.0895035
slope = 2.5318
/

Iteration 2
WSSR : 0.00234037  delta(WSSR)/WSSR : -592.938
delta(WSSR) : -1.38769  limit for stopping : 1e-05
lambda : 0.0147715

resultant parameter values
A = -1.35125
slope = 2.03757
/

Iteration 3
WSSR : 4.05182e-08  delta(WSSR)/WSSR : -57760
delta(WSSR) : -0.00234033  limit for stopping : 1e-05
lambda : 0.00147715

resultant parameter values
A = -1.42158
slope = 2.00023
/

Iteration 4
WSSR : 3.96991e-08  delta(WSSR)/WSSR : -0.0206309
delta(WSSR) : -8.19031e-10  limit for stopping : 1e-05
lambda : 0.000147715

resultant parameter values
A = -1.42162
slope = 2.0002
/

Iteration 5
WSSR : 3.96991e-08  delta(WSSR)/WSSR : -2.46451e-12
delta(WSSR) : -9.78389e-20  limit for stopping : 1e-05
lambda : 0.0000147715

resultant parameter values
A = -1.42162
slope = 2.0002
/

13
Scanning the output from the least squares fit from Gnuplot leads to the conclusion that second-order convergence is obtained in norms $\|x\|_\infty, \|x\|_1, \|x\|_2$.

h) Here is a plot of the error.

```
GNUplot] cd "/home/mitran/courses/MATH761/Fall2008"
set title "Error plot"
set xlabel "x"; set ylabel "Q-qex"
set grid
plot "err.dat" w l
```

The plot is reminiscent of $\sin x$ or $x(\pi - x)$. However, which one is a better description of the error distribution? Let us again use the `fit` utility within Gnuplot.

```
GNUplot] cd "/home/mitran/courses/MATH761/Fall2008"
s(t) = A*sin(t)
p(t) = A*t*(pi-t)
fit s(x) "err.dat" using 1:2 via A
fit p(x) "err.dat" using 1:2 via A
```
Iteration 0
WSSR : 511.999  delta(WSSR)/WSSR : 0
delta(WSSR) : 0  limit for stopping : 1e-05
lambda : 0.706762

initial set of free parameter values
A = 1
/

Iteration 1
WSSR : 0.000486379  delta(WSSR)/WSSR : -1.05267e+06
delta(WSSR) : -511.999  limit for stopping : 1e-05
lambda : 0.0706762

resultant parameter values
A = 0.000975158
/

Iteration 2
WSSR : 2.01797e-12  delta(WSSR)/WSSR : -2.41024e+08
delta(WSSR) : -0.000486379  limit for stopping : 1e-05
lambda : 0.00706762

resultant parameter values
A = 5.0885e-07
/

Iteration 3
WSSR : 1.97168e-12  delta(WSSR)/WSSR : -0.0234792
delta(WSSR) : -4.62933e-14  limit for stopping : 1e-05
lambda : 0.000706762

resultant parameter values
A = 4.99341e-07
/

Iteration 4
WSSR : 1.97168e-12  delta(WSSR)/WSSR : -2.04849e-16
delta(WSSR) : -4.03897e-28  limit for stopping : 1e-05
lambda : 7.06762e-05

resultant parameter values
A = 4.99341e-07
/

After 4 iterations the fit converged.
final sum of squares of residuals : 1.97168e-12
rel. change during last iteration : -2.04849e-16

degrees of freedom (FIT_NDF) : 1024
rms of residuals (FIT_STDFIT) = sqrt(WSSR/ndf) : 4.38801e-08
variance of residuals (reduced chisquare) = WSSR/ndf : 1.92547e-15

Final set of parameters
A = 4.99341e-07

Asymptotic Standard Error
A somewhat smaller standard error of the fitting was obtained for $x(\pi - x)$. The key question is whether the error distribution for this problem is given by the forcing (the r.h.s. term) $f(x)$ or by the first eigenfunction of the operator $\sin x$. Such questions always arise when solving differential equations by numerical methods. Can you present an analysis?

**Bonus (3 points):** Work through all of the homework points for the choice $<,>_1$. 