MATH 5620 NUMERICAL ANALYSIS II
HOMEWORK 5 SOLUTIONS

(a) The basis functions in the parent element $[0,1]$ can be obtained by writing the corresponding interpolating polynomial in Lagrange form

$$\widehat{\phi}_0(\widehat{x}) = 2(\widehat{x} - 1/2)(x - 1)$$
$$\widehat{\phi}_1(\widehat{x}) = -4\widehat{x}(\widehat{x} - 1)$$
$$\widehat{\phi}_2(\widehat{x}) = 2\widehat{x}(\widehat{x} - 1/2)$$

(b) The local stiffness matrix is:

$$\widehat{A}_{loc} = \frac{1}{3}\begin{bmatrix} 7 & -8 & 1 \\ -8 & 16 & -8 \\ 1 & -8 & 7 \end{bmatrix}.$$

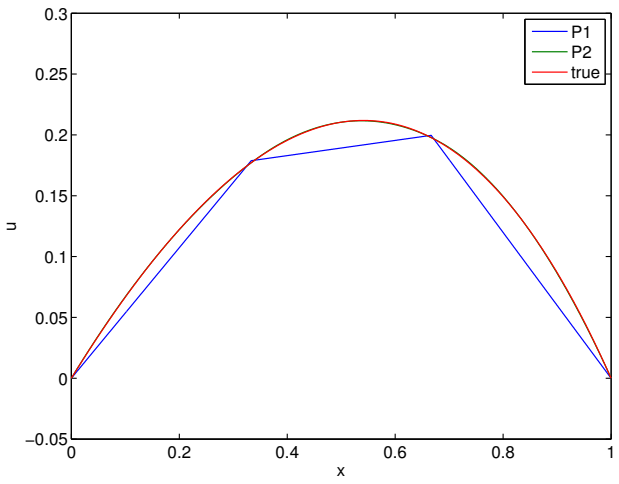
(c) The local mass matrix is

$$\widehat{M}_{loc} = \frac{1}{30}\begin{bmatrix} 4 & 2 & -1 \\ 2 & 16 & 2 \\ -1 & 2 & 4 \end{bmatrix}.$$

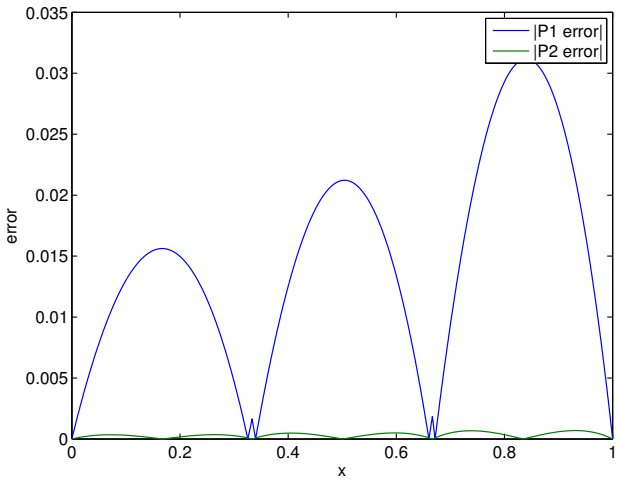
(d) The local to global map is the following $n \times 3$ matrix (consistent with the global ordering $0, \ldots, 2n$ of the nodes).

$$i = \begin{bmatrix} 0 & 1 & 2 \\ 2 & 3 & 4 \\ & \vdots & \\ 2n-2 & 2n-1 & 2n \end{bmatrix}$$

(e) The code is attached. The results for this part are here. We also plot the absolute error

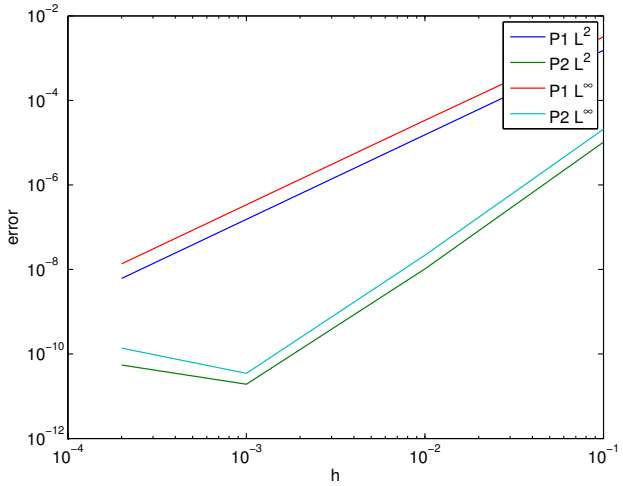


solutions



errors

(f) The errors (in both L^2 and L^∞ norms) are here:



To estimate the degree of accuracy we could do data fitting on the convergence curve or simply compute the slope between two data points as is done in `hw5.m`. The output below suggests that P1 elements are second order accurate and P2 elements are third order accurate.

```
>> hw5
computing solution for n=10
computing solution for n=100
computing solution for n=1000
computing solution for n=5000
accuracy L^2 P1=     2.0000     2.0000     1.9981

accuracy L^2 P2=     2.9997     2.7273    -0.6479

accuracy L^inf P1=     1.9760     1.9976     1.9997

accuracy L^inf P2=     2.9821     2.7939    -0.8535
```

hw5.m

```
% Math 5620 Homework 5
% Fernando Guevara Vasquez 2011

% fine grid
xf = linspace(0,1,1000)';

% right hand side and true solution
f = @(x) exp(x);
utrue = @(x) -exp(x) + (exp(1)-1)*x + 1;


% solve problem with n=3 with both P1 and P2
uf1=p1(3,f,xf);
uf2=p2(3,f,xf);
ut = utrue(xf);

% compare solutions
figure(1);
plot(xf,uf1,xf,uf2,xf,ut);
legend('P1','P2','true');
xlabel('x'); ylabel('u');
print('-depsc2','comp3.eps');

% absolute error plot
figure(2);
plot(xf,abs(uf1-ut),xf,abs(uf2-ut));
legend('|P1 error|','|P2 error|');
xlabel('x'); ylabel('error');
print('-depsc2','err3.eps');


% do convergence curves
ns=[10,100,1000,5000];
P1errsl2 = zeros(size(ns));
P1errslinf = zeros(size(ns));
P2errsl2 = zeros(size(ns));
P2errslinf = zeros(size(ns));
for i=1:length(ns), n=ns(i);
 fprintf('computing solution for n=%d\n',n);
 % make sure we have enough points per element
 xf = linspace(0,1,10*n)';
 % solve problem with the 2 methods and evaluate true solution
 uf1=p1(n,f,xf);
 uf2=p2(n,f,xf);
 ut = utrue(xf);

 % compute L^2 norm using composite trapezoidal rule
 % this can be done more efficiently using mass matrix
 P1errsl2(i)=sqrt(trapz((uf1-ut).^2)/(10*n-1));
 P1errslinf(i) = max(abs(uf1-ut));
 P2errsl2(i)=sqrt(trapz((uf2-ut).^2)/(10*n-1));
 P2errslinf(i) = max(abs(uf2-ut));
```

```
end;

figure(3);
loglog(1./ns,P1errsl2,1./ns,P2errsl2,1./ns,P1errslinf,1./ns,P2errslinf);
xlabel('h'); ylabel('error');
legend('P1 L^2','P2 L^2','P1 L^\infty','P2 L^\infty');
print('-depsc2','errplot.eps');

% numerical evaluation of exponent (slope)
fprintf('accuracy L^2 P1='); disp(diff(log(P1errsl2))./diff(log(1./ns)))
fprintf('accuracy L^2 P2='); disp(diff(log(P2errsl2))./diff(log(1./ns)))
fprintf('accuracy L^inf P1='); disp(diff(log(P1errslinf))./diff(log(1./ns)))
fprintf('accuracy L^inf P2='); disp(diff(log(P2errslinf))./diff(log(1./ns)))
```

p1.m

```
% P1 finite elements in 1D
% Fernando Guevara Vasquez 2011
%
%  solves elliptic problem
%    -u'' = f, u(0)=u(1)=0
%
%  using P1 elements
%
% inputs
%   n   # number of elements
%   f   right hand side function
%   xf  fine grid where we want to evaluate solution
%
% outputs
%   uf  solution evaluated at the nodes xf
function [uf] = p1(n,f,xf)

x = linspace(0,1,n+1)';

% local to global map
i = [1:n;2:n+1]';

% local stiffness matrix in the parent element
Kloc = [1 -1; -1 1];

% build the system matrix
K = sparse(n+1,n+1);
for e=1:n,
 K(i(e,:),i(e,:)) = K(i(e,:),i(e,:)) + Kloc/(x(e+1)-x(e));
end;

K(1,:) = 0; K(1,1)=1;
K(n+1,:) = 0; K(n+1,n+1)=1;

% local mass matrix
Floc = [1/3 1/6; 1/6 1/3];
% build right hand side
b=zeros(n+1,1);
for e=1:n,
 b(i(e,:)) = b(i(e,:)) + (x(e+1)-x(e))*Floc*f(x(i(e,:)));
end;
b(1)=0; b(n+1)=0;

% solve
u = K\b;

% do linear interpolation to evaluate at the fine grid
% evaluate function in grid
uf = zeros(size(xf));
for e=1:n,
 % indices of nodes in xf that are inside element e
 ii =  x(i(e,1)) <= xf & xf < x(i(e,2));
```

```
  uf(ii) = u(i(e,2))*(xf(ii)-x(i(e,1)))/(x(i(e,2))-x(i(e,1))) + ...
           u(i(e,1))*(x(i(e,2))-xf(ii))/(x(i(e,2))-x(i(e,1)));
end;
```

p2.m

```
% P2 finite elements in 1D
% Fernando Guevara Vasquez 2011
%
%  solves elliptic problem
%    -u'' = f, u(0)=u(1)=0
%
%  using P2 elements
%
% inputs
%   n   # number of elements
%   f   right hand side function
%   xf  fine grid where we want to evaluate solution
%
% outputs
%   uf  solution evaluated at the nodes xf
function [uf] = p2(n,f,xf)
h=1/n;
x = (0:2*n)*h/2;

% local basis functions in parent element [0,1]
phi{1} = @(x) 2*(x-1/2).*(x-1);
phi{2} = @(x) -4*x.*(x-1);
phi{3} = @(x) 2*x.*(x-1/2);

% local basis functions derivatives in parent element [0,1]
dphi{1} = @(x) 4*x-3;
dphi{2} = @(x) -8*x+4;
dphi{3} = @(x) 4*x-1;

% local stiffness matrix computation (using Simpson's rule)
w = [1,4,1]/6;
xh = [0,1/2,1];
Ahat = zeros(3,3);
for i=1:3,
 for j=1:3,
  Ahat(i,j) = ( dphi{i}(xh) .* dphi{j}(xh) )*w';
 end;
end;

% local mass matrix computation (using 3 node Gaussian rule)
w = [5,8,5]/18;
xh = ([-sqrt(15)/5, 0, sqrt(15)/5] + 1)/2;
Mhat = zeros(3,3);
for i=1:3,
 for j=1:3,
  Mhat(i,j) = ( phi{i}(xh) .* phi{j}(xh) )*w';
 end;
end;



% local to global map
```

```
i = [1:2:2*n-1
     2:2:2*n
     3:2:2*n+1]';

% build the system matrix
A = sparse(2*n+1,2*n+1);
for e=1:n,
 A(i(e,:),i(e,:)) = A(i(e,:),i(e,:)) + Ahat/(x(i(e,3))-x(i(e,1)));
end;

A(1,:) = 0; A(1,1)=1;
A(end,:) = 0; A(end,end)=1;

% build right hand side
b=zeros(2*n+1,1);
for e=1:n,
 b(i(e,:)) = b(i(e,:)) + (x(i(e,3))-x(i(e,1)))*Mhat*f(x(i(e,:)))';
end;
b(1)=0; b(end)=0;

% solve
u = A\b;

% evaluate function in grid
uf = zeros(size(xf));
for e=1:n,
 % indices of nodes in xf that are inside element e
 ii =  x(i(e,1)) <= xf & xf < x(i(e,3));
 for j=1:3,
  uf(ii) = uf(ii) + u(i(e,j))*phi{j}((xf(ii) - x(i(e,1)))/(x(i(e,3))-x(i(e,1))));
 end;
end;
```