MATH610-600

Programming Assignment #2

February 28, 2019

LU SUN

1 Problem Specifications

1.1 Exercise 1 (FEM for 2PBVP by Quadratic Elements)

Write a program for solving 2PBVP for second order ordinary differential equations by Rita-Galerkin method using qudratic finite elemments.

1.1.1 Problem 1

Let S be the axial force and q the intensity of the uniform load. The deflection W along the elemental length is given by:

$$-W''(x) + \frac{S}{D}W(x) = \frac{qx}{2D}(l-x), 0 < x < l, W(0) = W(l) = 0,$$

where l is the length of the plate. Take $q = 200\ lb/in^2, S = 100\ lb/in, D = 8.8 \times 10^7\ lb\ in, and\ l = 50\ in.$ The exact solution is given by: $a = \frac{Sl^2}{D}, b = \frac{ql^4}{2D}, t = x/l,$ and

$$W(t) = \frac{b}{a}\left(-t^2 + t - \frac{2}{a} + \frac{2}{a\ sinh(\sqrt{a}t)}[sinh(\sqrt{a}t) + sinh(\sqrt{a}(1-t))]\right)$$

- Use double precision.
- Use sparse data structures.
- Use $10, 20, 40, 80$ mesh intervals
- Give the error in a table in L^2, H^1, L^∞
- Plot error norms versus the mesh size. (all three error curves into a single plot).
- Using the third rule for mass matrix; using the first and second for stiffness matrix
- Using the second rule for stiffness matrix; using the first, second and third for mass matrix
- Report how the above variations affect the order of convergence as measured in the three norms specified above

1.1.2 Problem 2

All the parameter q, S, D, l, a, b, t is defined the same as above. The deflection W along the elemental length is given by:

$$-W''(x) + \frac{S}{D}W(x) = \frac{q}{2D}, 0 < x < l, W(0) = W(l) = 0.$$

The exact solution with $Q = \frac{ql^2}{2D}$ is given by:

$$W(t) = \frac{Q}{a}\left(1 - \frac{1}{sinh(\sqrt{a})}(sinh(\sqrt{a}t) + sinh(\sqrt{a}(1-t)))\right)$$

- Use double precision.
- Use sparse data structures.

- Use $10, 20, 40, 80$ mesh intervals
- Give the error in a table in L^2, H^1, L^∞
- Plot error norms versus the mesh size. (all three error curves into a single plot).
- Using the third rule for mass matrix; using the first and second for stiffness matrix
- Using the second rule for stiffness matrix; using the first, second and third for mass matrix
- Report how the above variations affect the order of convergence as measured in the three norms specified above

1.2 Exercise 2 (1D Convection-Diffusion Equation)

Consider the qingularly perturbed convection-diffusion equation:

$$-\epsilon u'' - u' = 0, u(0) = 0, u(1) = 1$$

Here in general $\epsilon << 1$. The exact solution is given by :

$$u(x) = \frac{1 - e^{-x/\epsilon}}{1 - e^{-1/\epsilon}}$$

.

1.2.1 Problem 1

- Write a finite difference code to approximate solutions to this problem using *upwind*, *donwind*, and *centered* differences to approximate the first derivative term u'.
- Consider two case $\epsilon = 1, \epsilon = 10^{-2}$
- Use 10, 20, 40, 80, 160, 320, 640 mesh intervals
- Make a table of the L_∞ norms of the errors, which includes a column for each method (upwind, downwind, centered) along with the number of mesh intervals.
- Plot L_∞ error norme vs. the mesh size. (All three methods on one plot.)

1.2.2 Problem 2

- Plot the finite difference approximations obtained along with the exact solution
- Rank the three methods according to their qualitative ablility to approximate the solution robustly with respect to h and ϵ (Behavior for $h >> \epsilon, h \approx \epsilon, h << \epsilon$)
- Present evidence that $|hb| = |h\epsilon^{-1}| = 1$ is in fact the dividing line between stability and instability for the downwind method.

2 Preliminaries

2.1 Exercise 1

2.1.1 *assembleLocalmass.m*

This m-file is used for calculating local mass matrix. $Localnodes = [x_i, x_m id, x_{i+1}]$ and kinds of quadrature ($quadrature = 1, 2, 3$) need to be input. Then mesh size $'mat\_B'$ need to be changed and defined as $x_{i+1} - x_i$. Namely, we need to change the following code.

```
1 mat_B = (localnodes(3,:) - localnodes(1,:))';
```

2.1.2 *assembleLocalRhs.m*

In the *assembleLocalRhs.m*, it gives out how to calculate $\int_0^l fv$ by quadrature with 4 point (that all the method to approximate integral use). Since no require about the quadrature in calculating $\int fv$ exists, we choose quadrature 4, at hand the largest one we can reach, to estimate local of it. The same as above, we only need to change $'mat\_B'$ to be mesh size (2*half of mesh size).

2.1.3 *assembleLocalStiffness.m*

In the *assembleLocalStiffness.m*, it gives out how to calculate 3×3 local stiffness matrix of $A_{ij}^1 := \int_{x_{2k-1}}^{x_{2k+1}} \phi'_i \phi'_j$ by the different quadrature (=1,2) required in the problem, where $2 * k - 1 \leq i, j \leq 2 * k + 1, k = 1, 2, 3..., number\ of\ mesh\ intervals$ ϕ_i is quadratic base function and defined as follow:

$$\begin{array}{rcl} \phi_0(s) & = & (2s-1)(s-1) \\ \phi_1(s) & = & 4(1-s)s \\ \phi_2(s) & = & s(2s-1) \end{array} \tag{1}$$

The equation (1) satisfies that $phi_0(0) = phi_1(1/2) = phi_2(1) = 1$, other points equal to zero. The correspond derivative equation is as follow:

$$\begin{array}{rcl} \phi'_0(s) & = & 4s-3 \\ \phi'_1(s) & = & 4-8s \\ \phi'_2(s) & = & 4s-1 \end{array} \tag{2}$$

In the code, the same as above, we only need to change mat_B.

2.1.4 *computeLocalErrors.m*

In this m-file, it only gives out the way to calculate local L^2 errors on each mesh. The code is just from the original *computeLocalErrors.m* which calculate L^2 error and H^1 error at the same time. I just delete all the things related to H^1 errors and keep all the things related to L^2 errors. Still mat_B need to be changed.

2.1.5 *computeLocalErrorsH1.m*

In this m-file, it gives out the way to calculate local both L^2 errors and H^1 errors on each mesh at same time (since H^1 error comes for L^2 error). The code is the same as the original *computeLocalErrors.m* and changes *mat_B*

2.1.6 *computeLocalInfErrors.m*

In this m-file, it gives out the way to calculate local L^∞ errors on each mesh. The code is the same as the original one and *mat_B* has been changed.

2.1.7 *constructDoFHandler.m*

In this m-file, it helps to figure out all the boundary point with different condition. When it comes to natural boundary condition, we just equal the point to be '0' instead of '2', for it doesn't have any practical meaning while coding. Moreover, since we use quadratic elements, we need to change some thing in the code. Here *T.e_elements* still stand for n, the number of mesh intervals. However, we also use mid-point on each intervals, so we have $2n+1$ nodes. *DoFHandler.dofs* save the interval nodes $[x_{2k-1}, x_{2k}, x_{2k+1}]$. The exact code to find the related point is as follow:

```
%DoFHandler.dofs = zeros(T.n_elements, p+1);
DoFHandler.dofs = zeros(2*T.n_elements, p+1);
% first the existing vertices
DoFHandler.dofs(:,1:3) = T.elements;
%DoFHandler.dofs(:,1:2) = T.elements;
```

Here T.elements is an matrix as follow:

1	2	3
2	3	4
...	...	...
2n-1	2n	2n+1

2.1.8 *constructTriangulation1D.m*

It's a m-function for creating a structure T which consists of all the things related to n (for example, *num_elements*, nodes information, mesh size and so on). We only need to change *T.n_elements* and *T.nodes*. The code is as follow:

```
T.n_nodes = 2*T.n_elements+1;
%T.n_nodes = T.n_elements+1;
T.n_edges = T.n_nodes;
T.nodes = linspace(0,L,T.n_nodes)'; % [n_nodes x dim]
T.elements = [(1:(T.n_nodes-1))', (2:T.n_nodes)' ,(3:(T.n_nodes+1))']; % ...
    [n_elements x dim+1 ]
%T.elements = [(1:(T.n_nodes-1))', (2:T.n_nodes)' ]; % [n_elements x ...
    dim+1 ]
```

2.1.9 *feEval.m*

It's a m-function for creating expression of functions related to quadratic bases and their prime. In this m-file, since p=2 is used, construct $FE_at_quad.hat_phi$and$E_at_quad.hat_phix$

according to the equation(1)(2) in the following way.

```
elseif (p==2)
    % construct 3 basis functions on [0,1]:  phi_0,  phi_1,  phi_2

    % construct 3 x-derivatives of basis functions on [0,1]:  dphi_0/dx, ...
        dphi_1/dx,  dphi_{2/dx
    FE_at_quad.hat_phi = [(2*x-1).*(x-1), 4*x.*(1-x),(2*x-1).*x];
    FE_at_quad.hat_phix = [4*x-3,   -8*x+4, 4*x-1];
```

2.1.10 *getQuadOnRefElement.m*

In this m-file, nothing has been changed. It's a m-function for creating different kinds of quadrature functions to interpolate for $\int_0^l \phi_i \phi_j$ and $\int_0^l \phi'_i \phi'_j$. Actually, we need to give out $\int_0^l Q(x)\phi_i\phi_j$ and $\int_0^l (-K(x))\phi'_i\phi'_j$. But the function related to $K(x)$ and $Q(x)$ in our problem is all constant or 'piece-wise' constant. We can make them out of the integral and then give the interpolation for the remaining part.

2.1.11 *main_error.m*

This new m-file is used for printing out the results at the same time. For problem 1-2 in exercise 1, $choose = [stiffnessmatrixquadrature, massmatrixquadrature]$, where $choose(1) = 1, 2$ and $choose(2) = 1, 2, 3$. Here, in $problem_case$,

'1': problem 1

'2': problem 2

```
choose=[1,3;2,3;2,1;2,2;2,3];

for i=1:1:5
    fprintf('homework 1,Stiffness qua=%d,mass ...
        qua=%d\n',choose(i,1),choose(i,2));
    %num_elements=10;
    %for j=1:1:4
    %    mainFEM1D_Dirichlet(num_elements,1,choose(i,:));
    %    num_elements=2*num_elements;
    %end
    mainErrorAnalysis(1,choose(i,:));
end

for i=1:1:5
    fprintf('homework 2,Stiffness qua=%d,mass ...
        qua=%d\n',choose(i,1),choose(i,2));
    %num_elements=10;
    %for j=1:1:4
    %    mainFEM1D_Dirichlet(num_elements,2,choose(i,:));
    %    num_elements=2*num_elements;
    %end
    mainErrorAnalysis(2,choose(i,:));
end
```

2.1.12 *mainErrorAnalysis.m*

This is a m-file for giving out the 3 or 2 kinds errors in the above 2 problems and comparing different errors in the same loglog picture. For the part of giving out errors, nothing has been changed. So only the part of picture code is shown as follow:

```
loglog(E(i,1),E(i,3),'-+',E(i,1),E(i,5),'-o',E(i,1),E(i,7),'-*',E(i,1),...
    (1/8).^(i-1)*(E(1,5)+E(1,3))/2,':',E(i,1),(1/4).^(i-1)*(E(1,3)+E(1,7))/2,'--');
xlabel('mesh');
ylabel('error');
legend('Inf','L2','H1','R=8','R=4');
title({['homework ',num2str(problem_case)],['Stiffness ...
    qua=',',',num2str(choose(1)),',','mass qua=',num2str(choose(2))]});
```

By calculating the covergance rate, we choose R=8 and R=4 to compare for the influence.

2.1.13 *mainFEM1D_Dirichlet.m*

This is a m-file for calculating u_h in each problem and creating solution picture. It also shows extra information in the picture for $u_h - u_exact$, where u_h means the solution we get by approximation and u_exact is the exact solution. A main point in the code is that we need to take into account of function $Q(x)$ with A_{ij}^0. Moreover, quadrature for stiffness matrix and mass matrix need to be included. So the exact changed code is as follow:

```
p = 2;  % polynomial degree of lagrange finite element basis
```

In the code, when we need to calculate related error or $\int fv$, we choos quadrature 4 in all the situation. Namely the quadrature presents in $quad\_n\_points, Quad, FE\_at\_Quad$

```
quad_stiffness_n_points = choose(1);
Quad_stiffness = getQuadOnRefElement(quad_stiffness_n_points);

quad_mass_m_points = choose(2);
Quad_mass = getQuadOnRefElement(quad_mass_m_points);

% Evaluate shape value and shape gradient(dim components) on reference
% element at quadrature points for the bulk elements.
% each term in FE_at_Quad structure is [nqx1]

[ FE_at_Quad_stiffness] = feEval( Quad_stiffness, p );
[ FE_at_Quad_mass] = feEval( Quad_mass, p );

quad_n_points = 4;%max(quad_stiffness_n_points,quad_mass_m_points);%4;
Quad = getQuadOnRefElement(quad_n_points);
[ FE_at_Quad] = feEval( Quad, p );
```

About the local cell part, the code is as follow:

```
for cell = 1:2:2*T.n_elements-1
%for cell = 1:T.n_elements
    %When K(x) is not constant, function_x = ( x_i + x_(i+1) ) / 2
    %i.e.mid point of the interval I_ei = [ x_i, x_(i+1) ]
    %K_ei: = K(function_x); namely, mid_point rule
```

```
    %function_x=L/T.n_elements*(cell-1/2);

    dofIndices = DoFHandler.dofs(cell,:);
    % [1x(p+1)]  extract indices pertaining to cell nodes

    vertices = T.nodes(T.elements(cell,:),:);
    % [(dim+1)xdim] coordinates of vertices

    cell_matrix1 = assembleLocalStiffness(vertices, FE_at_Quad_stiffness, ...
        Quad_stiffness, p);
    cell_matrix0 = assembleLocalmass(vertices, FE_at_Quad_mass, ...
        Quad_mass, p);
    % assemble local stiffness terms

    cell_matrix = -function_K * cell_matrix1 ...
                  +function_Q * cell_matrix0;

    cell_rhs    = assembleLocalRhs(f, vertices, FE_at_Quad, Quad,p);
    % assemble local rhs terms

    % contribute local terms to global stiffness and RHS structures
    A(dofIndices,dofIndices) = A(dofIndices,dofIndices) + cell_matrix;
    RHS(dofIndices) = RHS(dofIndices) + cell_rhs;
end
```

2.1.14 *runFEM1D.m*

In this m-file, it has the same meaning as *mainFEM1D_Dirichlet.m*, which used to calculate the approximated solution u_h. It's a function used in *mainErrorAnalysis.m* and is just a copy of *mainFEM1D_Dirichlet.m*.

2.2 Exercise 2

2.2.1 *constructTriangulation1D.m*

Define the same as the original one:

```
T.n_elements = num_elements;
T.n_nodes = T.n_elements+1;
```

2.2.2 *Differnece_One.m*

```
function A_matrix = Differnece_One(A, T ,options,b)
% options    finite  defferences method
%   1              unpwind
%   2              downwind
%   3              centered
h=T.meshsize;
switch (options)
    case 1
        a_ij = [-1,2,-1]/(h^2) + [-1,1,0]*b/h;
        for cell = 2:(T.n_nodes-1)
            dofIndices =  (cell-1): (cell+1);
```

```
            A(cell,dofIndices) = a_ij;
        end
    case 2
        a_ij = [-1,2,-1]/(h^2) + [0,-1,1]*b/h;
        for cell = 2:(T.n_nodes-1)
            dofIndices =  (cell-1): (cell+1);
            A(cell,dofIndices) = a_ij;
        end
    case 3
        a_ij = [-1,2,-1]/(h^2) + [-1,0,1]*b/h/2;
        for cell = 2:(T.n_nodes-1)
            dofIndices =  (cell-1): (cell+1);
            A(cell,dofIndices) = a_ij;
        end
end

A_matrix=A;

end
```

2.2.3 *FD_pro2.m*

```
function uh = FD_pro2(epsillion, num_elements,method)
%epsillion = 1; %=10^(-2);
b = -1/epsillion;
L = 1;
num_start = 1;
num_end = 1;
%num_elements = 10;% 20; 40; 80; 160; 320; 640;
T = constructTriangulation1D( L, num_elements, num_start, num_end);

u_exact = @(x)(1-exp(-x/epsillion))/(1-exp(-1/epsillion));

f = @(x)0.*x;
g_D = @(x) x;

T = constructTriangulation1D( L, num_elements, num_start, num_end);

uh = zeros(T.n_nodes,1);
RHS = zeros(T.n_nodes,1);
A = spalloc(T.n_nodes,T.n_nodes, 3*T.n_nodes);

A_matrix = Differnece_One(A,T,method,b);



for cell = 1:(T.n_nodes)
    RHS = f(T.nodes(cell));
end

uh(T.dirichletdofs) = g_D(T.dirichletdofs_coordinates);
RHS = RHS - A_matrix*uh;
uh(T.freedofs) = A_matrix(T.freedofs, T.freedofs) ...
                          \ RHS (T.freedofs);
```

2.2.4 *mainErrorAnalysis.m*

```
function [mesh,Inf_error] ...
    =mainErrorAnalysis(method,epsillion,numIterations,numElements)


E = zeros(numIterations,5);
for i = 1:numIterations

    [method, mesh_size, n_dofs, Linferror,uh] = runFEM1D( ...
        numElements(i),method,epsillion);
    E(i,:) = [method , mesh_size, n_dofs, Linferror, 0 ];


    % output table of errors and rates

    if (i>1)
        E(i,5) = E(i,4)/E(i-1,4); %Linf error
    end
    fprintf('%1d & %4d  & %1.2e & %1.2e & %1.2e \\\\ \n', ...
        E(i,1),E(i,3)-1,E(i,2),E(i,4),E(i,5));
    %fprintf('method=%1d, n_mesh = %4d, n_dofs = %4d, h = %1.2e, Einf = ...
        %1.6e, Rinf = %1.6f \n', E(i,1),E(i,3)-1,E(i,3),E(i,2),E(i,4),E(i,5));

end
mesh = E(:,2);
Inf_error = E(:,4);
end
```

2.2.5 *mainRun_pro2.m*

```
numIterations = 7;
baseNumElements = 10; %num elements on first solution
numElements = baseNumElements*2.^(0:(numIterations-1));

%numIterations = 8;
%numElements = [70,80,90,100,110,120,130,140];
%numElements = 1:1:8;



epsillion = 1;
InfError = zeros(numIterations,3);
for i = 1:1:3
    fprintf('method %1d\n',i);
    [mesh,Inf_error] = ...
        mainErrorAnalysis(i,epsillion,numIterations,numElements);
    InfError(:,i)= Inf_error;

end

Y = 0:0.1:1;
X = epsillion * ones(size(Y));

```

```
i = 1:1:numIterations;
lmbd=figure;
%loglog
plot(mesh(i),InfError(i,1),'-+',mesh(i),InfError(i,2),'-o',mesh(i),InfError(i,3),'-*');%,'-+',X,
%plot(mesh(i),InfError(i,1),'-+',mesh(i),InfError(i,2),'-+',mesh(i),InfError(i,3),'-+',X,Y,':');
xlabel('mesh size');
ylabel('L^{Inf} error');
legend('unpwind','downwind','centered');%,'|hb|=1');
title({['homework ',num2str(2)],['epsillion= ',num2str(epsillion)]});
saveas(lmbd, ['C:\Users\Sunlu\Desktop\sunlu_HM_02\result\',...
    'EX2_hm2_2','_epmax.png'],'png');



epsillion = 10^(-2);
InfError = zeros(numIterations,3);
for i = 1:1:3
    fprintf('method %1d\n',i);
    [mesh,Inf_error] = ...
        mainErrorAnalysis(i,epsillion,numIterations,numElements);
    InfError(:,i)= Inf_error;

end

Y = 0:0.1:1;
X = epsillion * ones(size(Y));

i = 1:1:numIterations;
lmbd=figure;
%loglog
plot(mesh(i),InfError(i,1),'-+',mesh(i),InfError(i,2),'-o',mesh(i),InfError(i,3),'-*');%,X,Y,':'

%plot(mesh(i),InfError(i,1),'-+',mesh(i),InfError(i,2),'-+',mesh(i),InfError(i,3),'-+',X,Y,':');
xlabel('mesh size');
ylabel('L^{Inf} error');
legend('unpwind','downwind','centered');%,'|hb|=1');
title({['homework ',num2str(2)],['epsillion= ',num2str(epsillion)]});
saveas(lmbd, ['C:\Users\Sunlu\Desktop\sunlu_HM_02\result\',...
    'EX2_hm2_2','_epmin.png'],'png');
```

2.2.6 $runFEM1D.m$

The same as $FD_pro2.m$

3 Results and Analysis

3.1 EX1 Problem 1(Deflection of a uniformly loaded plate)

3.1.1 Results

In the problem, it asks for error in a table, the data is as follow:

#cells	mesh size	L_∞	R(L_∞)	L_2	R(L_2)	H_1	R(H_1)
10	5.00e+00	7.62e+02	0.00	2.94e+03	0.00	3.48e+03	0.00
20	2.50e+00	7.69e+02	0.99	2.94e+03	1.00	4.75e+03	0.73
40	1.25e+00	7.71e+02	1.00	2.94e+03	1.00	8.01e+03	0.59
80	6.25e-01	7.71e+02	1.00	2.95e+03	1.00	1.52e+04	0.53

Table 1: Stiffness qua=1,mass qua=3

#cells	mesh size	L_∞	R(L_∞)	L_2	R(L_2)	H_1	R(H_1)
10	5.00e+00	5.14e-05	0.00	1.66e-04	0.00	2.72e-04	0.00
20	2.50e+00	6.76e-06	7.60	2.08e-05	7.97	5.78e-05	4.70
40	1.25e+00	8.67e-07	7.80	2.60e-06	7.99	1.37e-05	4.21
80	6.25e-01	1.10e-07	7.87	3.26e-07	8.00	3.39e-06	4.05

Table 2: Stiffness qua=2,mass qua=3

#cells	mesh size	L_∞	R(L_∞)	L_2	R(L_2)	H_1	R(H_1)
10	5.00e+00	5.14e-05	0.00	1.66e-04	0.00	2.72e-04	0.00
20	2.50e+00	6.76e-06	7.60	2.08e-05	7.97	5.78e-05	4.70
40	1.25e+00	8.67e-07	7.80	2.60e-06	7.99	1.37e-05	4.21
80	6.25e-01	1.10e-07	7.87	3.26e-07	8.00	3.39e-06	4.05

Table 3: Stiffness qua=2,mass qua=1

#cells	mesh size	L_∞	R(L_∞)	L_2	R(L_2)	H_1	R(H_1)
10	5.00e+00	5.14e-05	0.00	1.66e-04	0.00	2.72e-04	0.00
20	2.50e+00	6.76e-06	7.60	2.08e-05	7.97	5.78e-05	4.70
40	1.25e+00	8.67e-07	7.80	2.60e-06	7.99	1.37e-05	4.21
80	6.25e-01	1.10e-07	7.87	3.26e-07	8.00	3.39e-06	4.05

Table 4: Stiffness qua=2,mass qua2

From the data above, we can find that except stiffness quadrature = 1, in all the other situation, 3 kinds of error is very small and have converge rate of '8' or '4' by approximate.

In the problem, plots of error norms versus mesh size are required. The plots are as follow:

From the above plots, we can see that when stiffness quadrature =1, the error is very large and the converge rate approximate to 1 (almost not converge). When the stiffness quadrature =2, no matter what the mass quadrature is , it always get small errors less than 10^{-3} and has almost converge rate 8 for L_2, L_∞ errors, 4 for H_1 error. The reason for this, may because $|Q(x)| = S/D = 100/(8.8 \times 10^7) << |K(x)| = 1$, since K(x) defines

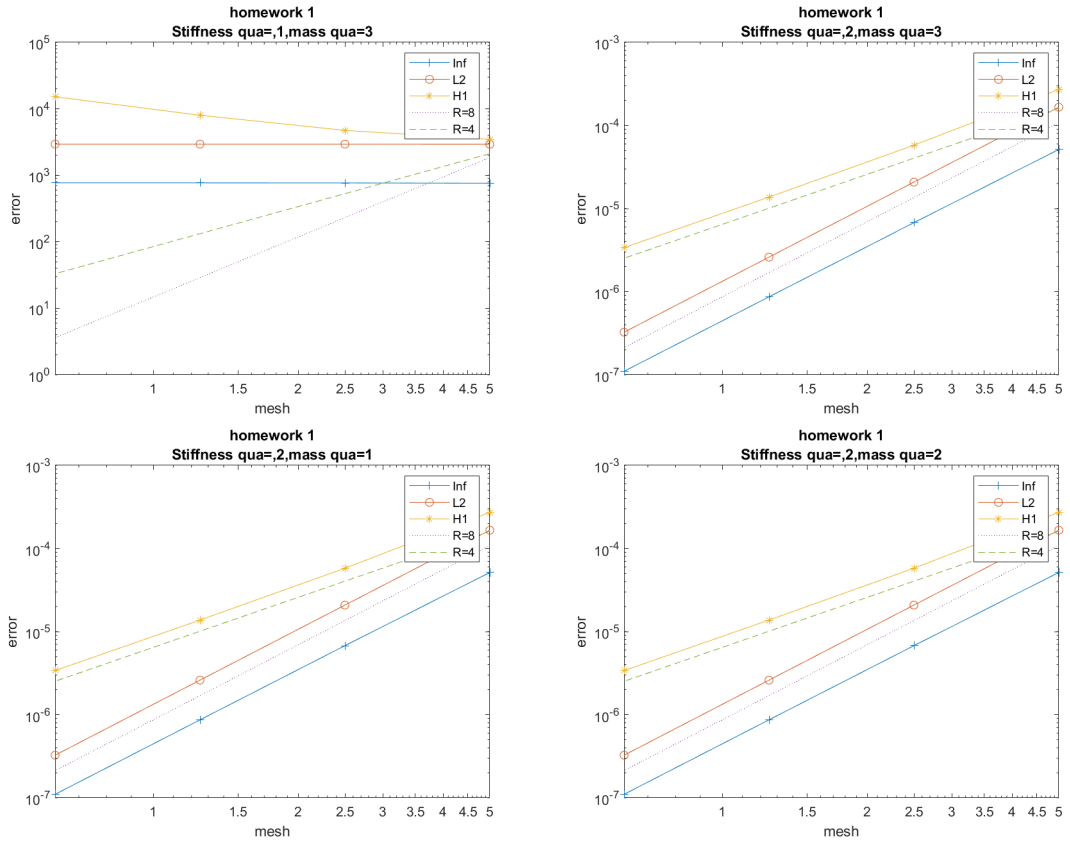


Figure 1: The solution of homework 1

the stiffness matrix, Q(x) defines the mass matrix, quadrature for stiffness influence more on the error and convergence rate. In general, the larger quadrature is, the more accurate results we can get, namely the less error we can get. And it should influence the convergence rate.

3.1.2 Analysis

From the above data and plot, we can find that :

(1)Except stiffness quadrature = 1, in all the other situation, 3 kinds of error is very small and have converge rate of '8' or '4' by approximate.

(2)When stiffness quadrature =1, the error is very large.

(3) When the stiffness quadrature =2, no matter what the mass quadrature is , it always get small errors less than 10^{-3}.

(4) $|Q(x)| = S/D = 100/(8.8 \times 10^7) << |K(x)| = 1$ makes that quadrature for stiffness matrix influence more on the error and convergence rate than that quadrature for mass matrix.

(5)In general, the larger quadrature is, the less error we can get. And it should influence the convergence rate.

3.2 EX1 Problem 2 (Deflection of a uniformly loaded plate)

3.2.1 Results

In the problem, it asks for error in a table, the data is as follow:

#cells	mesh size	L_∞	R(L_∞)	L_2	R(L_2)	H_1	R(H_1)
10	5.00e+00	1.23e+00	0.00	6.45e+00	0.00	7.64e+00	0.00
20	2.50e+00	1.23e+00	1.00	6.45e+00	1.00	1.04e+01	0.73
40	1.25e+00	1.23e+00	1.00	6.45e+00	1.00	1.76e+01	0.59
80	6.25e-01	1.23e+00	1.00	6.45e+00	1.00	3.33e+01	0.53

Table 5: Stiffness qua=1,mass qua=3

#cells	mesh size	L_∞	R(L_∞)	L_2	R(L_2)	H_1	R(H_1)
10	5.00e+00	2.92e-11	0.00	9.42e-11	0.00	1.54e-10	0.00
20	2.50e+00	3.84e-12	7.60	1.18e-11	7.97	3.29e-11	4.70
40	1.25e+00	4.92e-13	7.81	1.48e-12	7.99	7.81e-12	4.21
80	6.25e-01	6.24e-14	7.89	1.85e-13	8.00	1.93e-12	4.05

Table 6: Stiffness qua=2,mass qua=3

#cells	mesh size	L_∞	R(L_∞)	L_2	R(L_2)	H_1	R(H_1)
10	5.00e+00	4.20e-10	0.00	1.01e-09	0.00	1.83e-09	0.00
20	2.50e+00	1.05e-10	4.00	2.45e-10	4.13	7.93e-10	2.31
40	1.25e+00	2.63e-11	4.00	6.08e-11	4.03	3.81e-10	2.08
80	6.25e-01	6.57e-12	4.00	1.52e-11	4.01	1.89e-10	2.02

Table 7: Stiffness qua=2,mass qua=1

#cells	mesh size	L_∞	R(L_∞)	L_2	R(L_2)	H_1	R(H_1)
10	5.00e+00	3.01e-11	0.00	9.47e-11	0.00	1.55e-10	0.00
20	2.50e+00	3.90e-12	7.71	1.18e-11	8.00	3.29e-11	4.71
40	1.25e+00	4.96e-13	7.86	1.48e-12	8.00	7.81e-12	4.21
80	6.25e-01	6.27e-14	7.91	1.85e-13	8.00	1.93e-12	4.05

Table 8: Stiffness qua=2,mass qua2

From the data above, we can find that except stiffness quadrature = 1, in all the other situation, 3 kinds of error is very small, have converge rate of '8' or '4' by approximate when mass quadrature not equals to 1 and converge rate of '4' or '2' by approximate when mass quadrature equals to 1.

In the problem, plots of error norms versus mesh size are required. The plots are as follow:

From the above plots, we can see that when stiffness quadrature =1, the error is very large and the converge rate approximate to 1 (almost not converge). When the stiffness quadrature =2, no matter what the mass quadrature is , it always get small errors less than 10^{-8} and has almost converge rate 8(mass qua =2,3) or 4(mass qua =1) for L_2, L_∞

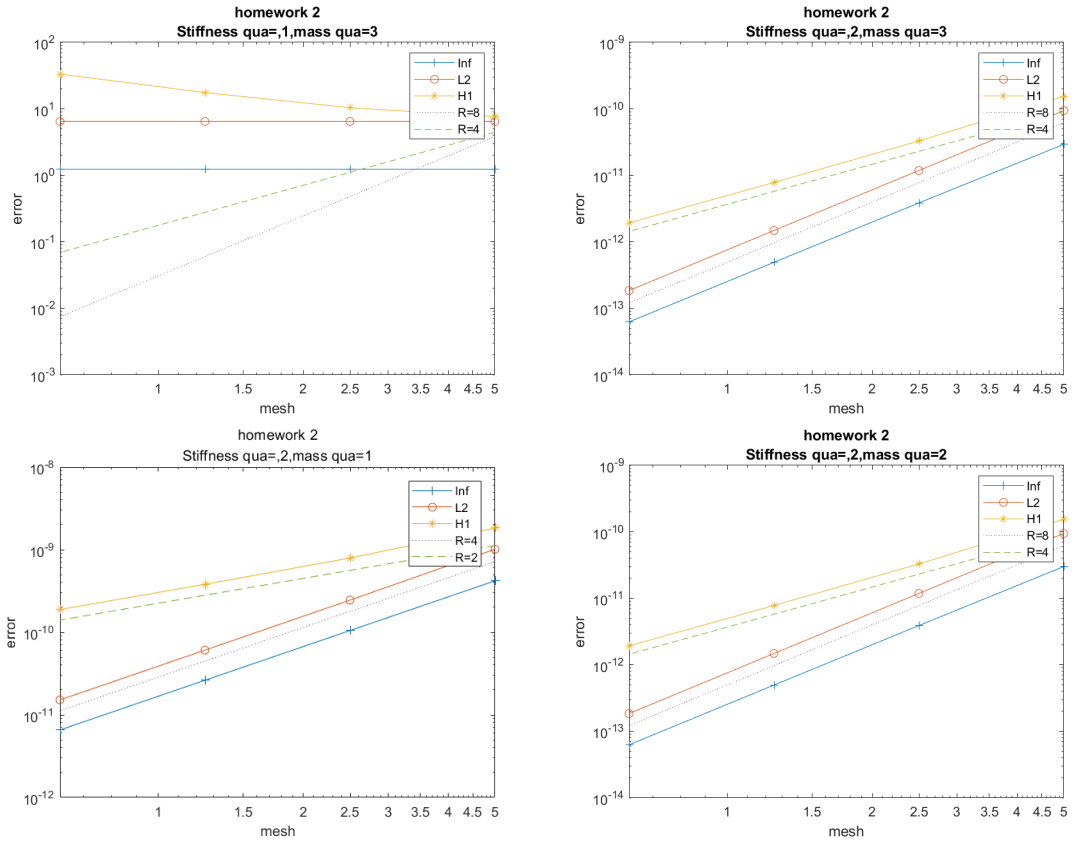


Figure 2: The solution of homework 2

errors, 4(mass qua =2,3) or 2(mass qua =1) for H_1 error. The reason for this, may because $|Q(x)| = S/D = 100/(8.8 \times 10^7) << |K(x)| = 1$, since K(x) defines the stiffness matrix, Q(x) defines the mass matrix, quadrature for stiffness influence more on the error. From the data and plots, it shows that mass quadrature also influence convergence rate. In general, the larger quadrature is, the more accurate results we can get, namely the less error and larger convergence rate we can get.

3.2.2 Analysis

From the above data and plot, we can find that :

(1)Except stiffness quadrature = 1, in all the other situation, 3 kinds of error is very small which is less than 10^{-8}.

(2)When stiffness quadrature =1, the error is very large.

(3)When the stiffness quadrature =2, no matter what the mass quadrature is , it always get small errors less than 10^{-8}.

(4) $|Q(x)| = S/D = 100/(8.8 \times 10^7) << |K(x)| = 1$ makes that quadrature for stiffness matrix influence more on the error and convergence rate than that quadrature for mass matrix.

(5)When the stiffness quadrature =2, the larger mass quadrature is, the larger convergence rate for 3 kinds of errors we can get.

(5)In general, the larger quadrature is, the less error and larger convergence rate is.

3.3 EX 2 Problem 1

3.3.1 Results

In the problem, a table of L_∞ is asked. The table is as follow:

method	#cells	mesh size	L_∞	R(L_∞)
1	10	1.00e-01	6.46e-03	0.00e+00
1	20	5.00e-02	3.12e-03	4.83e-01
1	40	2.50e-02	1.53e-03	4.92e-01
1	80	1.25e-02	7.61e-04	4.96e-01
1	160	6.25e-03	3.79e-04	4.98e-01
1	320	3.13e-03	1.89e-04	4.99e-01
1	640	1.56e-03	9.45e-05	4.99e-01
2	10	1.00e-01	5.67e-03	0.00e+00
2	20	5.00e-02	2.92e-03	5.16e-01
2	40	2.50e-02	1.49e-03	5.09e-01
2	80	1.25e-02	7.49e-04	5.04e-01
2	160	6.25e-03	3.76e-04	5.02e-01
2	320	3.13e-03	1.88e-04	5.01e-01
2	640	1.56e-03	9.43e-05	5.01e-01
3	10	1.00e-01	1.01e-04	0.00e+00
3	20	5.00e-02	2.51e-05	2.50e-01
3	40	2.50e-02	6.29e-06	2.50e-01
3	80	1.25e-02	1.57e-06	2.50e-01
3	160	6.25e-03	3.93e-07	2.50e-01
3	320	3.13e-03	9.83e-08	2.50e-01
3	640	1.56e-03	2.46e-08	2.50e-01

Table 9: ϵ=1,method 1='unpwind', method 2 ='downwind', method 3 ='centered'

The plot of ϵ=1 is as follow:

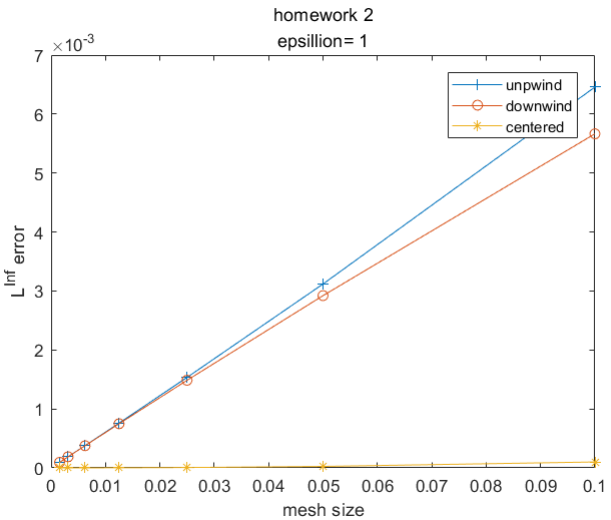


Figure 3: ϵ=1

method	#cells	mesh size	L_∞	$R(L_\infty)$
1	10	1.00e-01	1.11e+00	0.00e+00
1	20	5.00e-02	1.25e+00	1.12e+00
1	40	2.50e-02	1.67e+00	1.33e+00
1	80	1.25e-02	5.37e-01	3.22e-01
1	160	6.25e-03	1.60e-01	2.99e-01
1	320	3.13e-03	6.67e-02	4.16e-01
1	640	1.56e-03	3.08e-02	4.62e-01
2	10	1.00e-01	9.09e-02	0.00e+00
2	20	5.00e-02	1.60e-01	1.76e+00
2	40	2.50e-02	2.04e-01	1.27e+00
2	80	1.25e-02	1.58e-01	7.76e-01
2	160	6.25e-03	9.22e-02	5.84e-01
2	320	3.13e-03	5.07e-02	5.50e-01
2	640	1.56e-03	2.70e-02	5.32e-01
3	10	1.00e-01	6.96e-01	0.00e+00
3	20	5.00e-02	4.35e-01	6.25e-01
3	40	2.50e-02	1.93e-01	4.44e-01
3	80	1.25e-02	5.57e-02	2.88e-01
3	160	6.25e-03	1.21e-02	2.18e-01
3	320	3.13e-03	3.02e-03	2.49e-01
3	640	1.56e-03	7.49e-04	2.48e-01

Table 10: ϵ=0.01,method 1='unpwind', method 2 ='downwind', method 3 ='centered'

The plot of ϵ=1 is as follow:

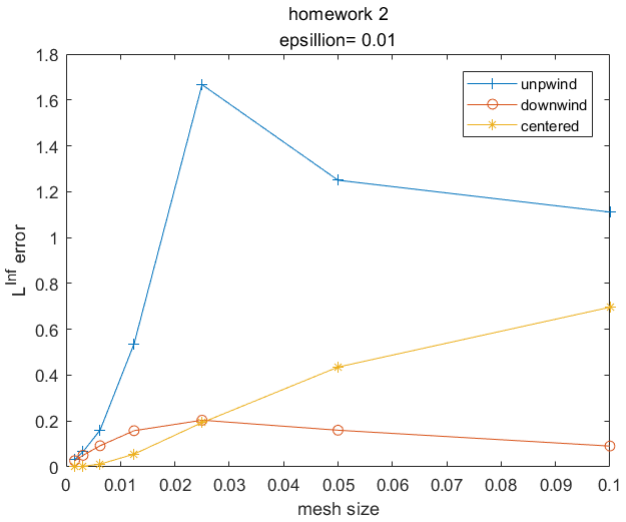
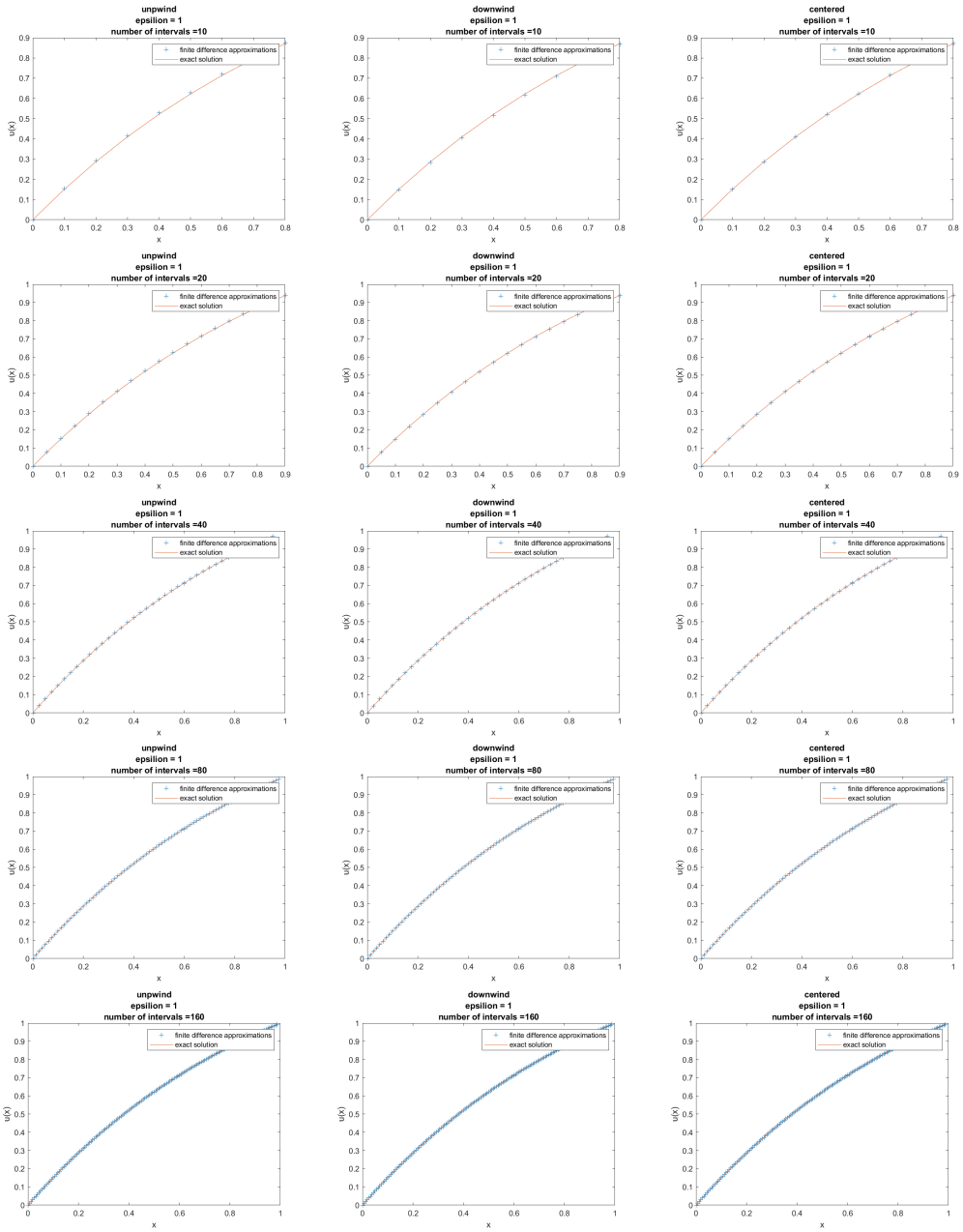


Figure 4: ϵ=0.01

3.4 EX 2 Problem 2

3.4.1 Results

The plots of finite difference approximations obtained along with the exact solution is as follows:



3.4.2 Analysis

From the data and plots abovet, we can have that (the ablility to approximate the solution):

(1)$h >> \epsilon$: upwind weaker than centered weaker than downwind

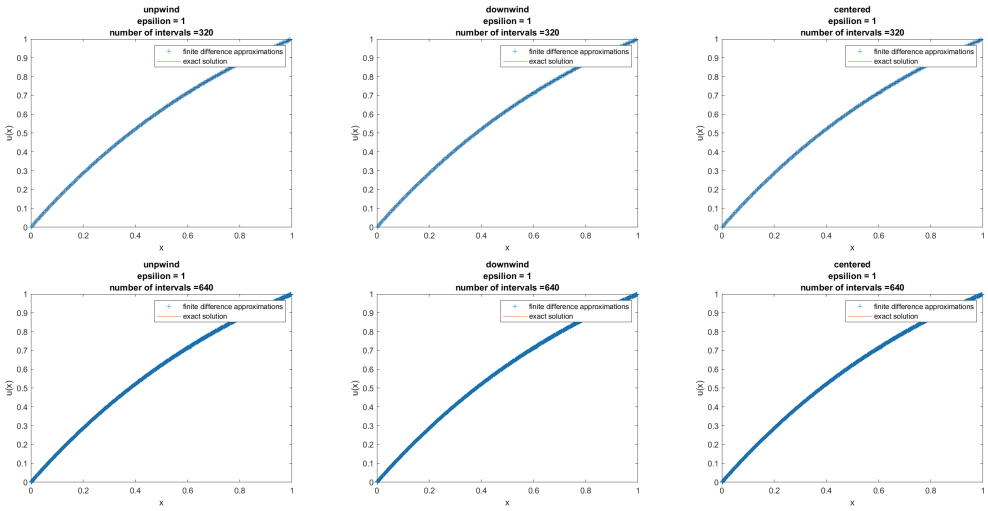
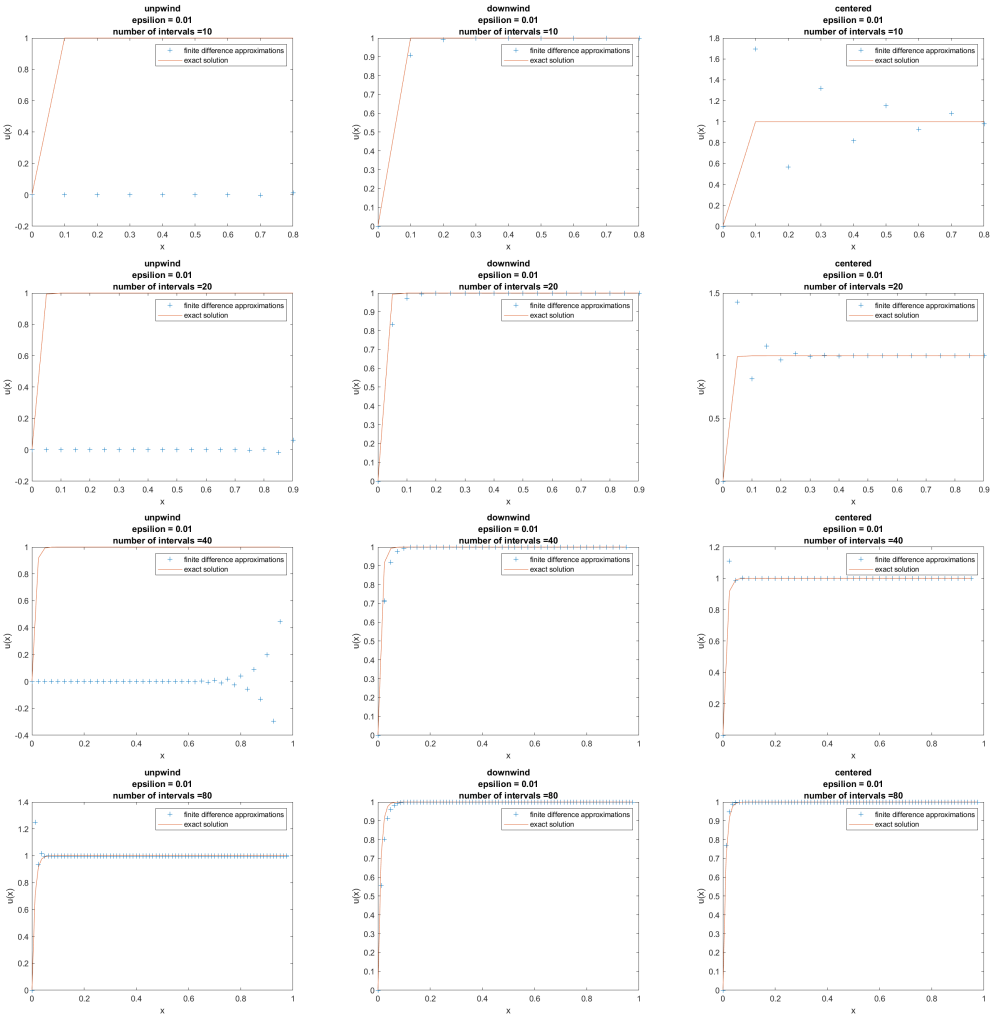


Figure 5: $\epsilon = 1$



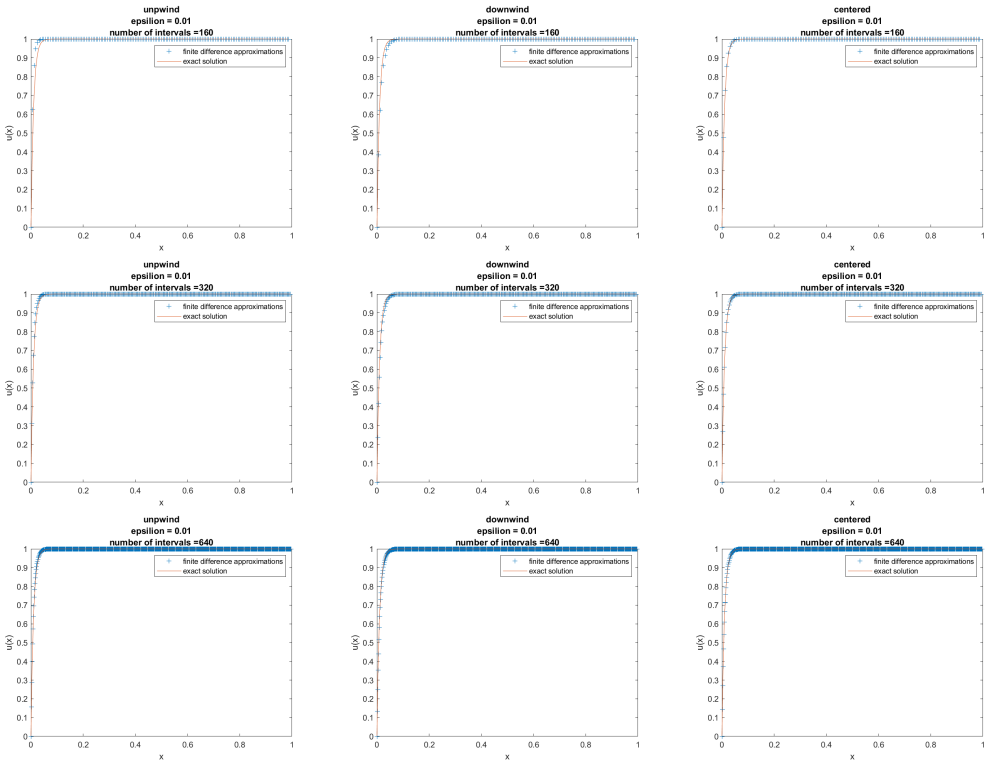


Figure 6: $\epsilon = 0.01$

(2)$h \approx \epsilon$: upwind weaker than downwind weaker than centered

(3)$h << \epsilon$: upwind weaker than downwind weaker than centered

evidence that $|hb| = |h\epsilon^{-1}| = 1$ is in fact the dividing line between stability and instability for the downwind method is shown in the following figure. Here we change the mesh intervals to be 1-7 or 70-140 instead of the original one:

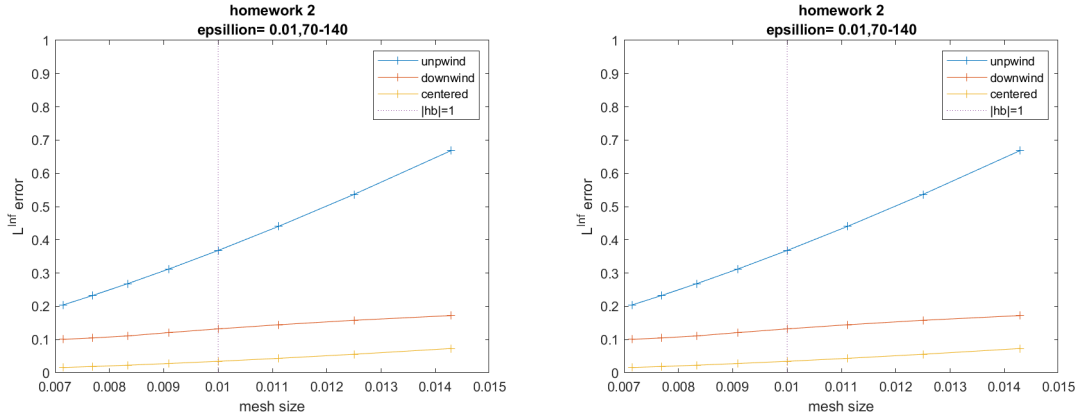


Figure 7: Compare evidence