APMA 2560 Final Exam

ZHANG Zhen

Problem 1

We use the finite element method with piecewise cubic elements to solve the following equation:

$$\begin{cases} u''''(x) = e^{-x}, \\ u(0) = u''(0) = u(1) = u''(1) + 2u'(1) = 0 \end{cases} \tag{1}$$

The exact solution is as follows:

$$u(x) = e^{-x} + \frac{3}{10e}x^3 - \frac{1}{2}x^2 + (\frac{3}{2} - \frac{13}{10e})x - 1 \tag{2}$$

We plot three tables below, first one showing the L^2 error of $u - u_h$ and numerical order of accuracy, second and third one indicating the error at natural boundary conditions $u''(0) = 0$ and $u''(1) + 2u'(1) = 0$. $|u''(0) - u_h''(0)|$ as well as $|u''(1) + 2u'(1) - u''(0) - 2u'(0)|$ are taken as boundary errors.

Table 1: L^2 Error Table

h	L^2 Error	Numerical Order
$\frac{1}{10}$	1.0908×10^{-7}	
$\frac{1}{20}$	6.8202×10^{-9}	3.9993
$\frac{1}{40}$	4.2636×10^{-10}	3.9998
$\frac{1}{80}$	2.6651×10^{-11}	3.9998
$\frac{1}{160}$	2.6555×10^{-12}	3.3271

It can be observed that the L_2 convergence rate is 4 for polynomial of degree 3, which is consistent with what we learned in class. However, for the last row we see a decay in numerical order of our scheme. It is because some matrix operation or integration operation enlarges the machine ϵ, bringing it to a comparable level as 10^{-12}. The result can be improved if we use a more accurate data type like *vpa*. But I don't have enough computation power to do so.

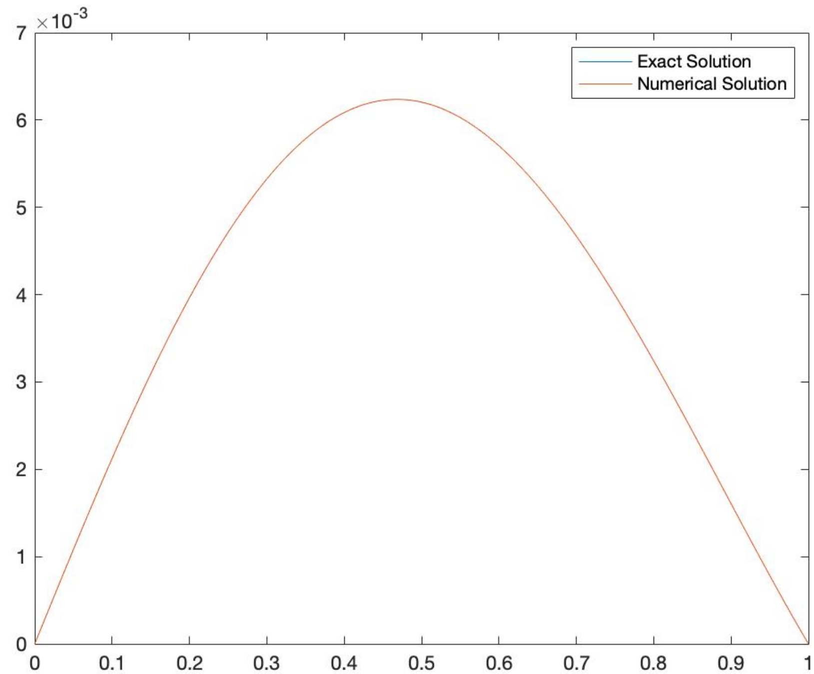
Table 2: Error for u"(0)=0

h	Error	Numerical Order
$\frac{1}{10}$	8.0082×10^{-4}	
$\frac{1}{20}$	2.0422×10^{-4}	1.9714
$\frac{1}{40}$	5.1566×10^{-5}	1.9856
$\frac{1}{80}$	1.2956×10^{-5}	1.9928
$\frac{1}{160}$	3.2471×10^{-6}	1.9964

Table 3: Error for u"(1)+2u'(1)=0

h	Error	Numerical Order
$\frac{1}{10}$	3.1914×10^{-4}	
$\frac{1}{20}$	7.8194×10^{-5}	2.0291
$\frac{1}{40}$	1.9353×10^{-5}	2.0145
$\frac{1}{80}$	4.8141×10^{-6}	2.0072
$\frac{1}{160}$	1.2005×10^{-6}	2.0036

Furthermore, errors for the natural boundary conditions shows second order convergence rate. This is consistent with what we learned from midterm, when interpreting our method as a finite difference scheme. The next image shows the exact solution and our numerically calculated solution when $h = \frac{1}{160}$. They match each other perfectly.



Problem 2

(a) Semi-discrete Fourier Galerkin Approximation.
Assume $u_N(x,t) = \sum\limits_{|n|\le \frac{N}{2}} a_n(t)e^{inx}$. Then $R_N(x,t) = (u_N)_t + (u_N)_x = \sum\limits_{|n|\le \frac{N}{2}} (a_n'(t) + ina_n(t))e^{inx} \in \hat{B}_N$. Since $R_N \perp \hat{B}_N$, we get $R_N \equiv 0$. Thus the semi-discrete Fourier Galerkin Approximation is as follows:

$$\begin{cases} a_n'(t) + ina_n(t) = 0 \\ a_n(0) = \hat{f}_n = \frac{1}{2\pi}\int_0^{2\pi} f(x)e^{-inx}dx \end{cases} \tag{3}$$

Solve the ODE above, we get $a_n(t) = \frac{1}{2\pi}e^{-int}\int_0^{2\pi} f(x)e^{-inx}dx$,
$u_N(x,t) = \sum_{|n|\le \frac{N}{2}} \frac{1}{2\pi}e^{-int+inx}\int_0^{2\pi} f(x)e^{-inx}dx = \sum_{|n|\le \frac{N}{2}} \frac{1}{2\pi}e^{inx}\int_0^{2\pi} f(x-t)e^{-inx}dx = P_N f(x-t)$.

Let $\varphi_N(x) = \sum\limits_{|n|\leq\frac{N}{2}} \hat{\varphi}_n e^{inx}$, where $\hat{\varphi}_n = \frac{1}{2\pi}\int_0^{2\pi}\varphi(x)e^{-inx}dx$. Then $\varphi_N \in \hat{B}_N$, thus $\varphi_N \perp R_N$, further implies $|\int_0^{2\pi}(u(x,t)-u_N(x,t))\varphi(x)dx| = |\int_0^{2\pi}(u(x,t)-u_N(x,t))(\varphi(x)-\varphi_N(x))dx| \leq ||u-u_N||_{L^2}\cdot||\varphi-\varphi_N||_{L^2} \leq C_1||\varphi-\varphi_N||_{L^2} = C_1\sum_{|n|>\frac{N}{2}}|\hat{\varphi}_n|^2 \leq C_2\frac{\sum_{|n|>\frac{N}{2}}|(in)^p\hat{\varphi}_n|^2}{(\frac{N}{2})^{2p}} \leq C_3\frac{\sum\limits_n|\hat{\varphi}_n^{(p)}|^2}{(\frac{N}{2})^{2p}} \leq C_3\frac{||\varphi||_{H^p}}{(\frac{N}{2})^{2p}} = \frac{C}{N^p}||\varphi||_{H^p}$.
$C_1 = ||u-u_N||_{L^2} = ||f(x-t)-P_Nf(x-t)||_{L^2} \leq ||f(x-t)||_{L^2} = ||f||_{L^2}$, which is independent of t. C is independent of of t because C_1 is independent of t and the calculation from C_1 to C involves no time dependency.

(b) Semi-discrete Fourier Collocation Approximation.

$$\begin{cases} \sum\limits_{|n|\leq\frac{N}{2}} a_n'(t)e^{inx_j} + \sum\limits_{|k|\leq\frac{N}{2}}(ik)a_k(t)e^{ikx_j} = 0 \\ a_n(0) = \tilde{f}_n = \frac{1}{N+1}\sum\limits_{j=0}^{N}u(x_j)e^{-inx_j} \end{cases} \tag{4}$$

The estimate in (a) will not hold if we apply this method. However, we can change to initial condition to be the same as (a) as follow:

$$a_n(0) = \hat{f}_n = \frac{1}{2\pi}\int_0^{2\pi}f(x)e^{-inx}dx \tag{5}$$

With this initial condition, we can see the solution u_N for (a) is also a solution here. Thus the same estimate follows.

Program

```
%% Preprocessing
clear
clc

%% start solving PDEs!!
ll=0;rr=1;numtrial=5;gamma=2;
error=zeros(numtrial,1);
bderror1=zeros(numtrial,1);
bderror2=zeros(numtrial,1);

for i=1:numtrial
    M=10*2^(i-1)-1;
    u=zeros(2*M+2,1);
    h=(rr-ll)/(M+1);
    A11=8*h;
    A12=0;
    A22=24/h;
    B11=2*h;
    B12=-6;
```

```
B21=6;
B22=-12/h;
C11=4*h;
D11=4*h+gamma*h^2;
A=zeros(2*M+2);
A(1,1)=C11;A(1,2)=B11;A(1,3)=B12;
A(2,1)=B11;A(2,2)=A11;A(2,3)=A12;A(2,4)=B11;A(2,5)=B12;
A(3,1)=B12;A(3,2)=A12;A(3,3)=A22;A(3,4)=B21;A(3,5)=B22;
A(2*M,2*M-2)=B11;A(2*M,2*M-1)=B21;
A(2*M,2*M)=A11;A(2*M,2*M+1)=A12;A(2*M,2*M+2)=B11;
A(2*M+1,2*M-2)=B12;A(2*M+1,2*M-1)=B22;
A(2*M+1,2*M)=A12;A(2*M+1,2*M+1)=A22;A(2*M+1,2*M+2)=B21;
A(2*M+2,2*M)=B11;A(2*M+2,2*M+1)=B21;A(2*M+2,2*M+2)=D11;
for j=4:2*M-1
    if mod(j,2)==0
        A(j,j-2)=B11;A(j,j-1)=B21;A(j,j)=A11;
        A(j,j+1)=A12;A(j,j+2)=B11;A(j,j+3)=B12;
    else
        A(j,j-3)=B12;A(j,j-2)=B22;A(j,j-1)=A12;
        A(j,j)=A22;A(j,j+1)=B21;A(j,j+2)=B22;
    end
end
b=zeros(2*M+2,1);
x=linspace(ll,rr,M+2);
b1=@(x) basis1(x,0,h).*exp(-x);
b(1)=integral(b1,0,h);
b2=@(x) basis1(x,(M+1)*h,h).*exp(-x);
b(2*M+2)=integral(b2,M*h,(M+1)*h);
for j=2:2*M+1
    if mod(j,2)==0
        b3=@(x) basis1(x,j/2*h,h).*exp(-x);
        b(j)=integral(b3,j/2*h-h,j/2*h)+integral(b3,j/2*h,j/2*h+h);
    else
        b4=@(x) basis2(x,(j-1)/2*h,h).*exp(-x);
        b(j)=integral(b4,(j-1)/2*h-h,(j-1)/2*h)+
        integral(b4,(j-1)/2*h,(j-1)/2*h+h);
    end
end
u=A\b;

numeval=200000;
testx=linspace(0,1,numeval);
l2=0;
for p=1:numeval
    j=floor(testx(p)/h)+1;
    if (j==M+2)
        j=j-1;
    end
```

```
        if (j==1)
            uu=u(1)*basis1(testx(p),0,h)+
            u(2)*basis1(testx(p),h,h)+
            u(3)*basis2(testx(p),h,h);
        elseif (j==M+1)
            uu=u(2*M)*basis1(testx(p),M*h,h)+
            u(2*M+1)*basis2(testx(p),M*h,h)+
            u(2*M+2)*basis1(testx(p),(M+1)*h,h);
        else
            uu=u(2*j-2)*basis1(testx(p),(j-1)*h,h)+
            u(2*j-1)*basis2(testx(p),(j-1)*h,h)+
            u(2*j)*basis1(testx(p),j*h,h)+
            u(2*j+1)*basis2(testx(p),j*h,h);
        end
        uu=uu/h^2;
        l2=l2+(uexact(testx(p))-uu)^2;
    end
    uuu=zeros(M+2,1);
    for p=1:M+2
        j=p;
        if (j==M+2)
            j=j-1;
        end
        if (j==1)
            uuu(p)=u(1)*basis1(x(p),0,h)+
            u(2)*basis1(x(p),h,h)+u(3)*basis2(x(p),h,h);
        elseif (j==M+1)
            uuu(p)=u(2*M)*basis1(x(p),M*h,h)+
            u(2*M+1)*basis2(x(p),M*h,h)+
            u(2*M+2)*basis1(x(p),(M+1)*h,h);
        else
            uuu(p)=u(2*j-2)*basis1(x(p),(j-1)*h,h)+
            u(2*j-1)*basis2(x(p),(j-1)*h,h)+
            u(2*j)*basis1(x(p),j*h,h)+
            u(2*j+1)*basis2(x(p),j*h,h);
        end
        uuu(p)=uuu(p)/h^2;
    end
    l2=sqrt(l2/numeval);
    bderror1(i)=abs(-4/h*u(1)-2/h*u(2)+6/h^2*u(3));
    bderror2(i)=abs(u(2*M)*2/h+u(2*M+1)*6/h^2+u(2*M+2)*4/h+2*u(2*M+2));
    error(i)=l2;
    exactu=uexact(x)';
end

%% Plot graph and summarize errors
plot(x,exactu,x,uuu)
legend('Exact Solution','Numerical Solution')
```

```
savefig('result/solution.fig')
for i=2:numtrial
    rate(i-1)=log(error(i-1)/error(i))/log(2);
    rate2(i-1)=log(bderror1(i-1)/bderror1(i))/log(2);
    rate3(i-1)=log(bderror2(i-1)/bderror2(i))/log(2);
end

function f=basis1(x,xi,h)
    if x<=xi
        f=(x-xi+h).^2.*(x-xi);
    else
        f=(x-xi-h).^2.*(x-xi);
    end
end

function f=basis2(x,xi,h)
    if x<=xi
        f=(x-xi+h).^2.*(-2*x+h+2*xi)./h;
    else
        f=(x-xi-h).^2.*(2*x+h-2*xi)./h;
    end
end

function y=uexact(x)
    y=exp(-x)+3/10/exp(1).*x.^3-0.5*x.^2+(1.5-13/10/exp(1))*x-1;
end
```