Project 1: The Basics of FEM

Zhipeng Yu

1. Introduction to the problem:

Solve the following boundary value problem, with domain (0, L), analytically, the conditions are given like this.

$$\frac{d}{dx}\left(E\frac{du}{dx}\right) = k^2 sin(\frac{2\pi kx}{L})$$

$$E = given\ constant = 0.1$$

$$k = given\ constant$$

$$L = 1$$

$$u(0) = \Delta_1 = given\ constant = 0$$

$$u(L) = \Delta_2 = given\ constant = 1$$

What am I going to do is to get a solution of “u” without direct integral.

How am I going to do is to combine some linear simple functions and add them up such that getting as closed as possible to the true solution.

For example: function like “(x+1)/2,(x-1)/2”.

Although these functions seem simple, they are really powerful if you get fairly large number of them.

2. Objective

Goals: As we are going to find an approximate solution, we are supposed to make error less equal to 0.05.

And bellow, it is how to calculate the error.

$$e^N \stackrel{\text{def}}{=} \frac{||u-u^N||_{E(\Omega)}}{||u||_{E(\Omega)}} \leq TOL = 0.05,$$

$$||u||_{E(\Omega)} \stackrel{\text{def}}{=} \sqrt{\int_\Omega \frac{du}{dx} E \frac{du}{dx}\, dx}$$

Besides, we are going to find what the smallest number of "functions" is for each given "k". Actually, we divide a given domain into N elements. And every element has a unique function. So now the problem becomes that what is the best N for each k?

$$\begin{aligned}
k = 1 &\Rightarrow N = ?\\
k = 2 &\Rightarrow N = ?\\
k = 4 &\Rightarrow N = ?\\
k = 8 &\Rightarrow N = ?\\
k = 16 &\Rightarrow N = ?\\
k = 32 &\Rightarrow N = ?
\end{aligned}$$

When changing k or N, what is the error going to be? We will try to figure it out soon.

3. My procedure

First we decide to use the Weak Formulation to solve it.

$$\boxed{\begin{array}{l}\text{Find } u \in H^1(\Omega)\ u|_{\Gamma_u} = d \text{ such that } \forall \nu \in H^1(\Omega), \nu|_{\Gamma_u} = 0 \\ \displaystyle\int_\Omega \frac{d\nu}{dx} E \frac{du}{dx}\, dx = \int_\Omega f\nu\, dx + t\nu|_{\Gamma_t}.\end{array}}$$

We approximate "u" by:

$$u^h(x) = \sum_{j=1}^{N} a_j \phi_j(x).$$

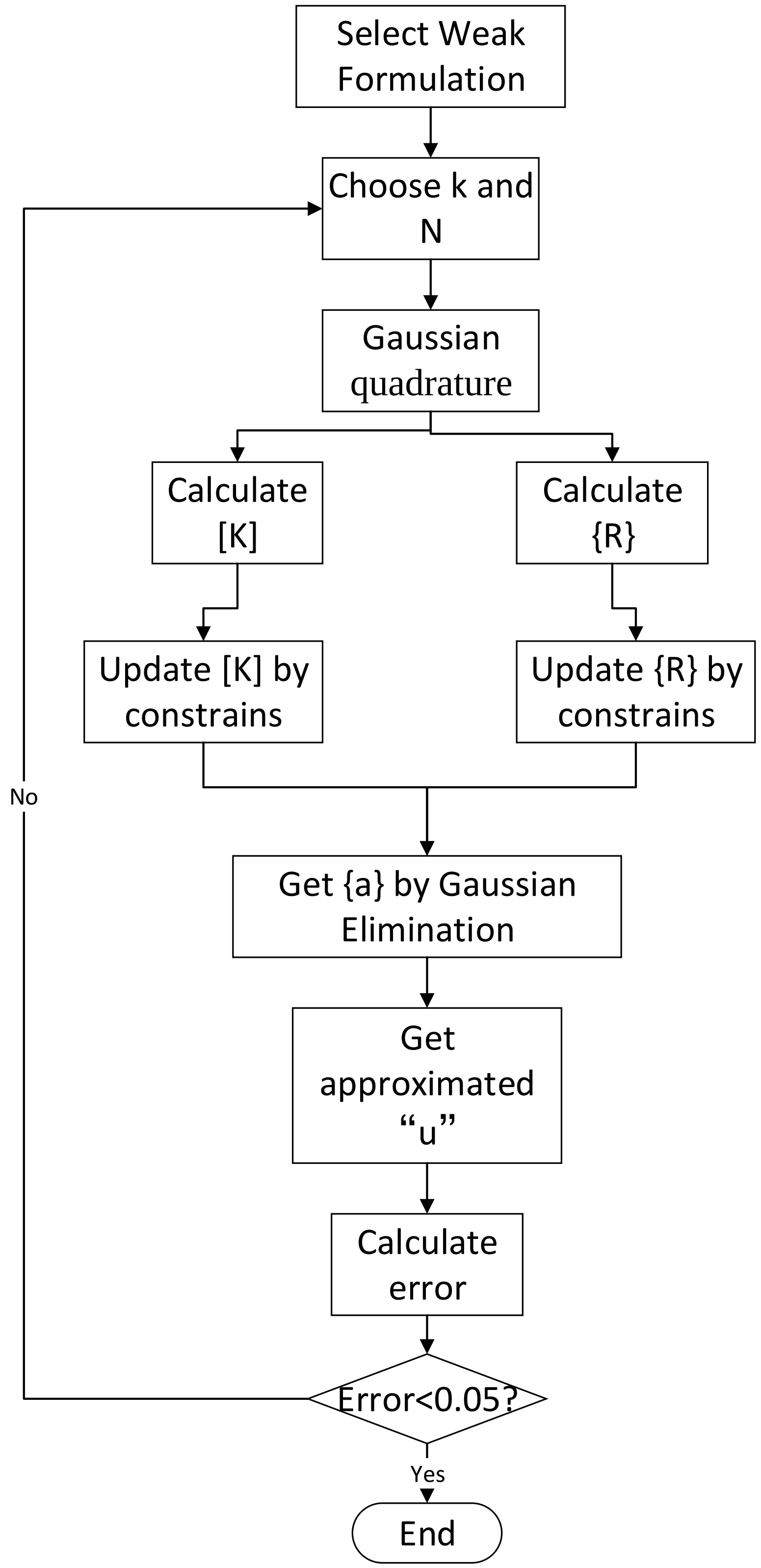
If we choose "v" with the same approximation functions, but a different linear combination, we get "v" like this:

$$\nu^h(x) = \sum_{i=1}^{N} b_i \phi_i(x),$$

Since the "v" are arbitrary (formulation definition), the "bi" are arbitrary, therefore

$$\boxed{\begin{array}{l}\sum_{i=1}^{N} b_i \left(\sum_{j=1}^{N} K_{ij} a_j - R_i\right) = 0 \Rightarrow [K]\{a\} = \{R\}, \\ K_{ij} \stackrel{\text{def}}{=} \int_\Omega \frac{d\phi_i}{dx} E \frac{d\phi_j}{dx}\, dx \text{ and} \\ R_i \stackrel{\text{def}}{=} \int_\Omega \phi_i f\, dx + \phi_i t|_{\Gamma_t},\end{array}}$$

According to that, we will use some mathematical trick to simplify the integral calculation such as Gaussian quadrature (ϕi represents the simple "function" that we build by ourselves). Then will get [K] and {R} and add constrains to them. And, we will solve this linear algebra problem by Gaussian Elimination. Once we get {a}, we get approximated "u". Based on that, we can do further test on errors and try to find the best N for each k.



4. Findings

1) Best N for each k:

   K=1, N=28;

   K=2, N=67;

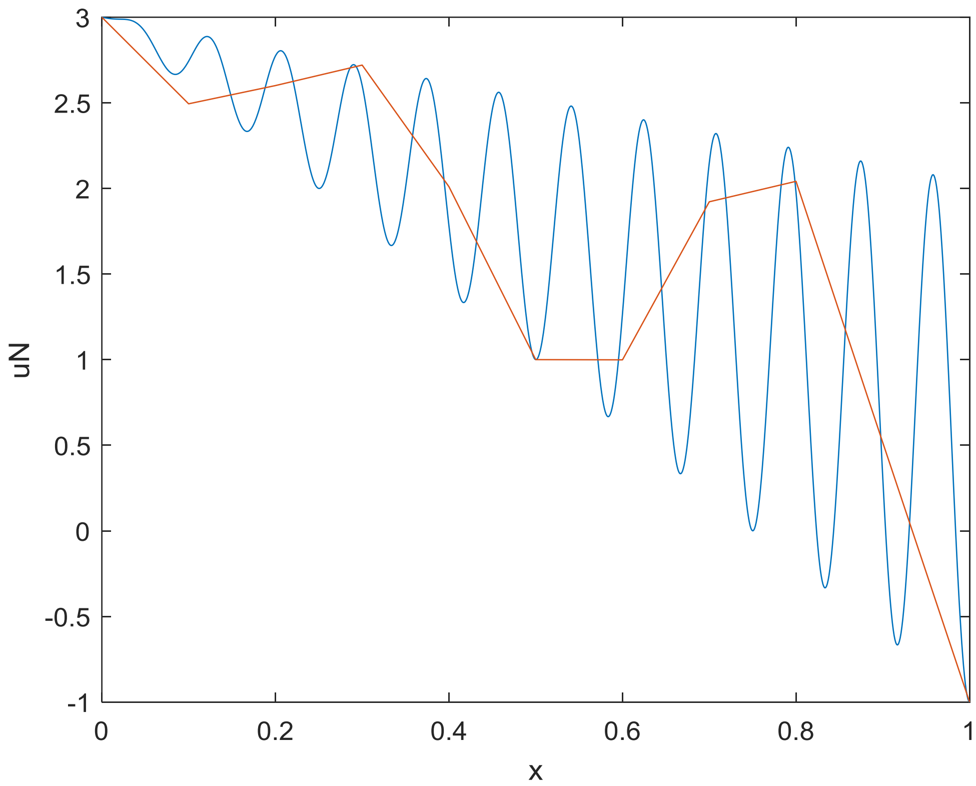
   K=4, N=142

   K=8, N=289

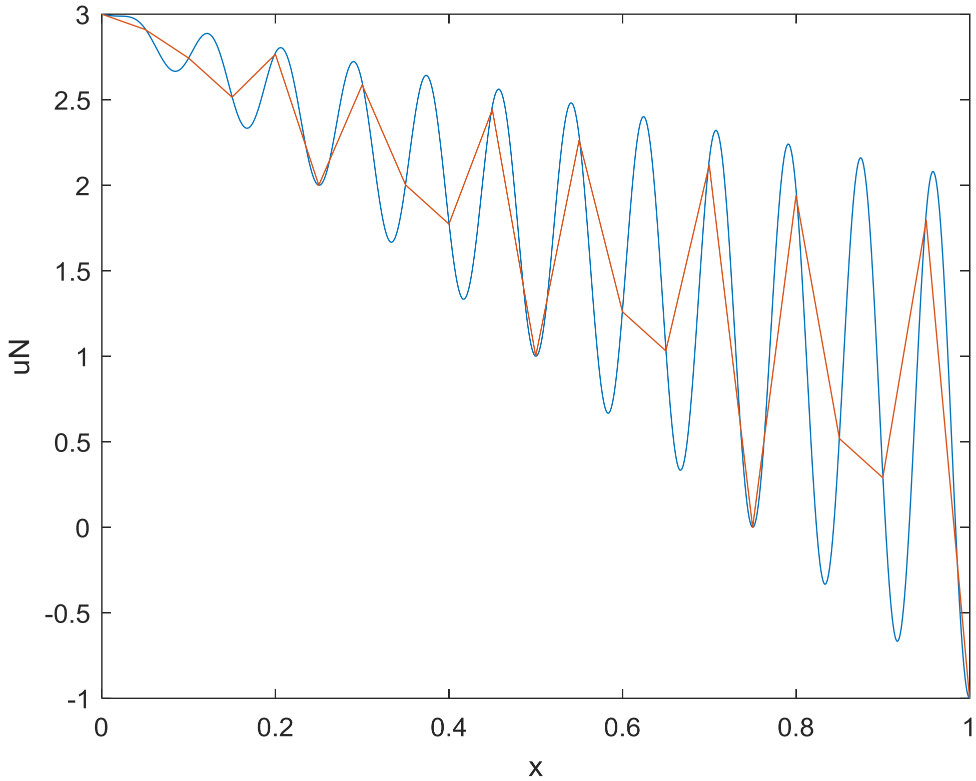
   K=16, N=580

   K=32, N=1161

2)

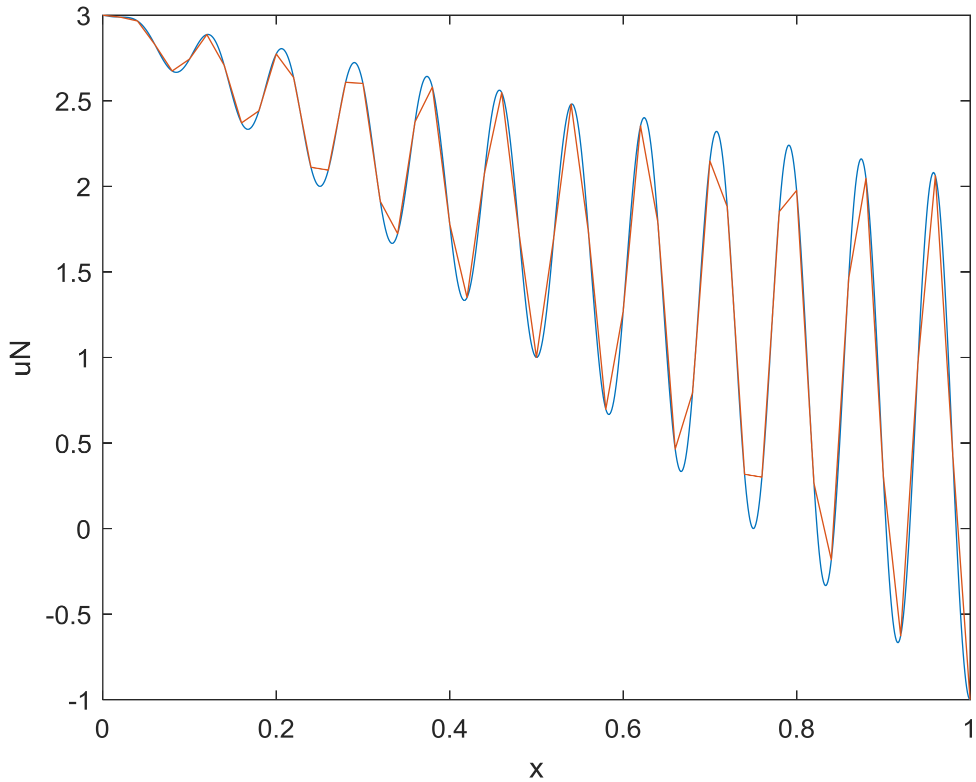


N=1



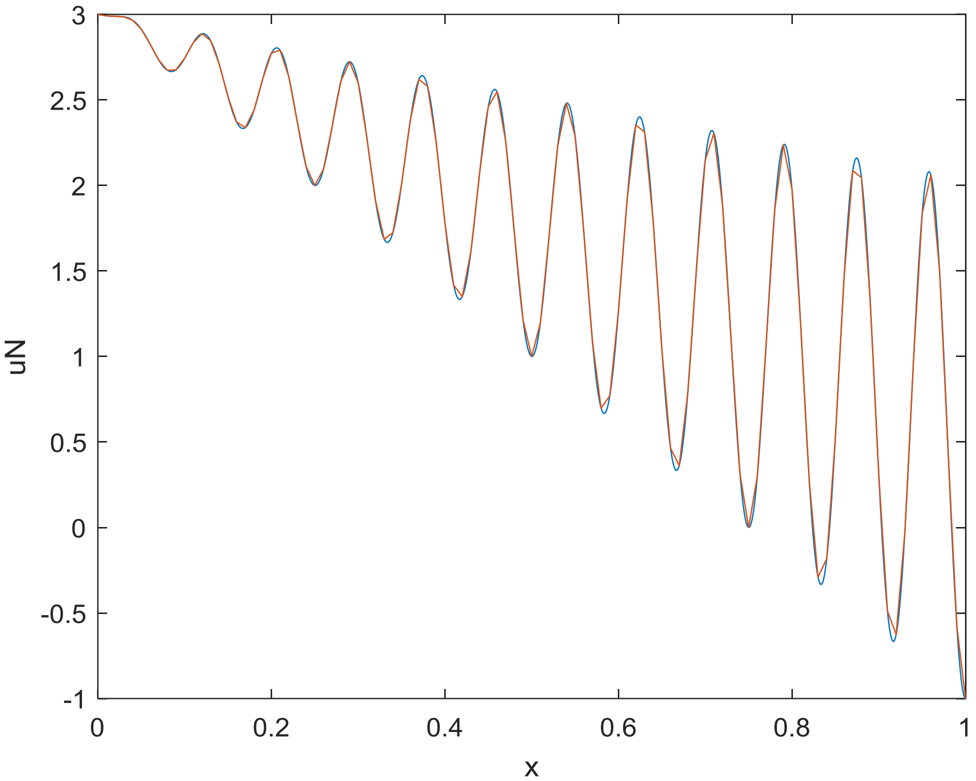
3
2.5
2
1.5
1
0.5
0
-0.5
-1
uN
0
0.2
0.4
0.6
0.8
1
x

n=20

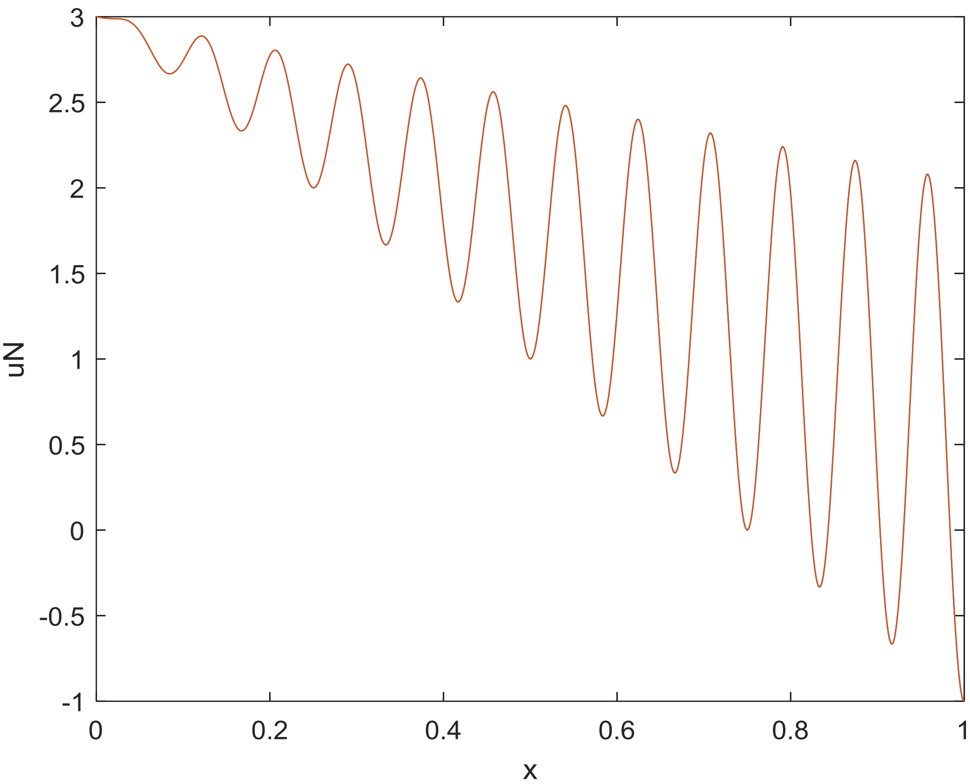


3
2.5
2
1.5
1
0.5
0
-0.5
-1
uN
0
0.2
0.4
0.6
0.8
1
x

n=50



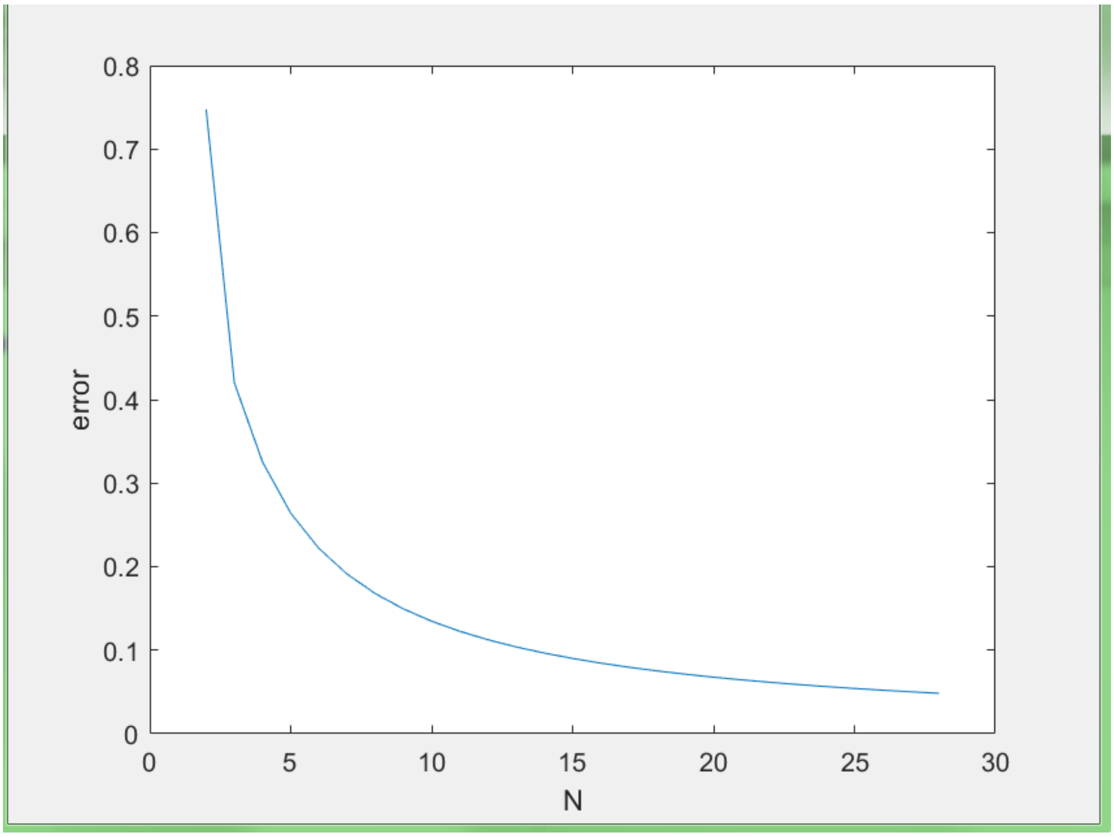
n=100



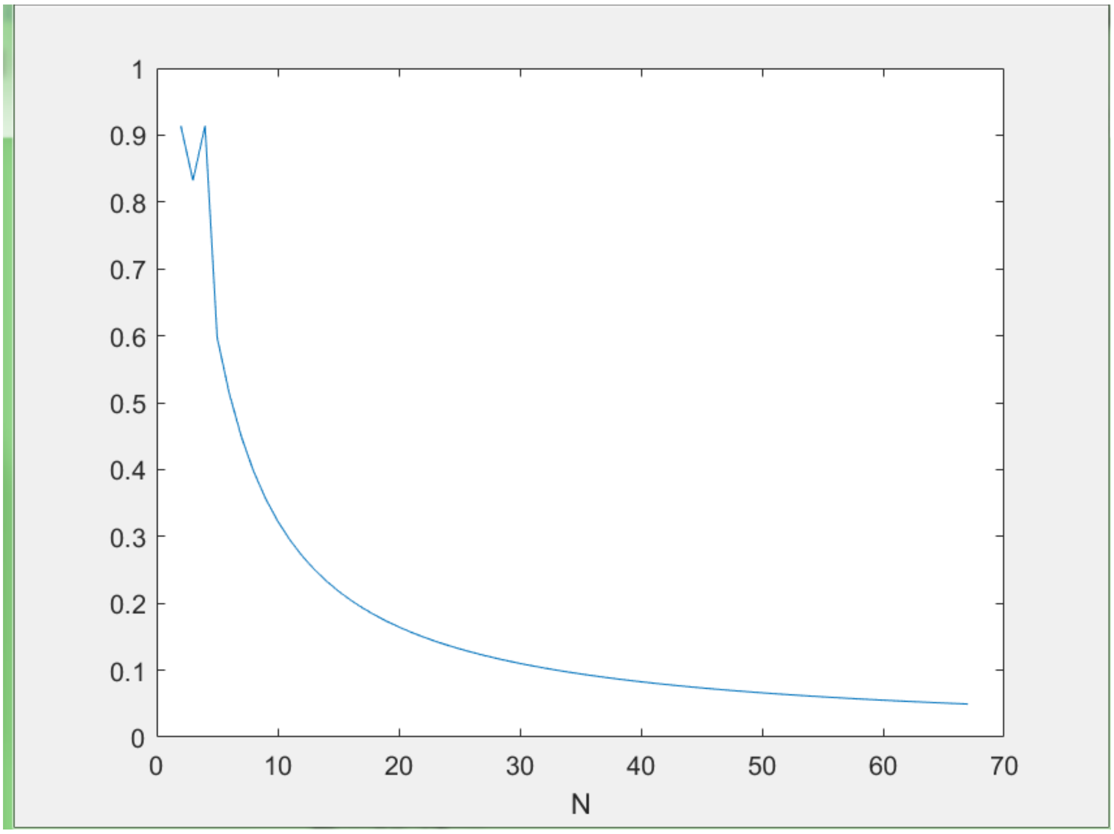
n=500

3) k =1, 2, 4, 8, 16, 32…

k=1

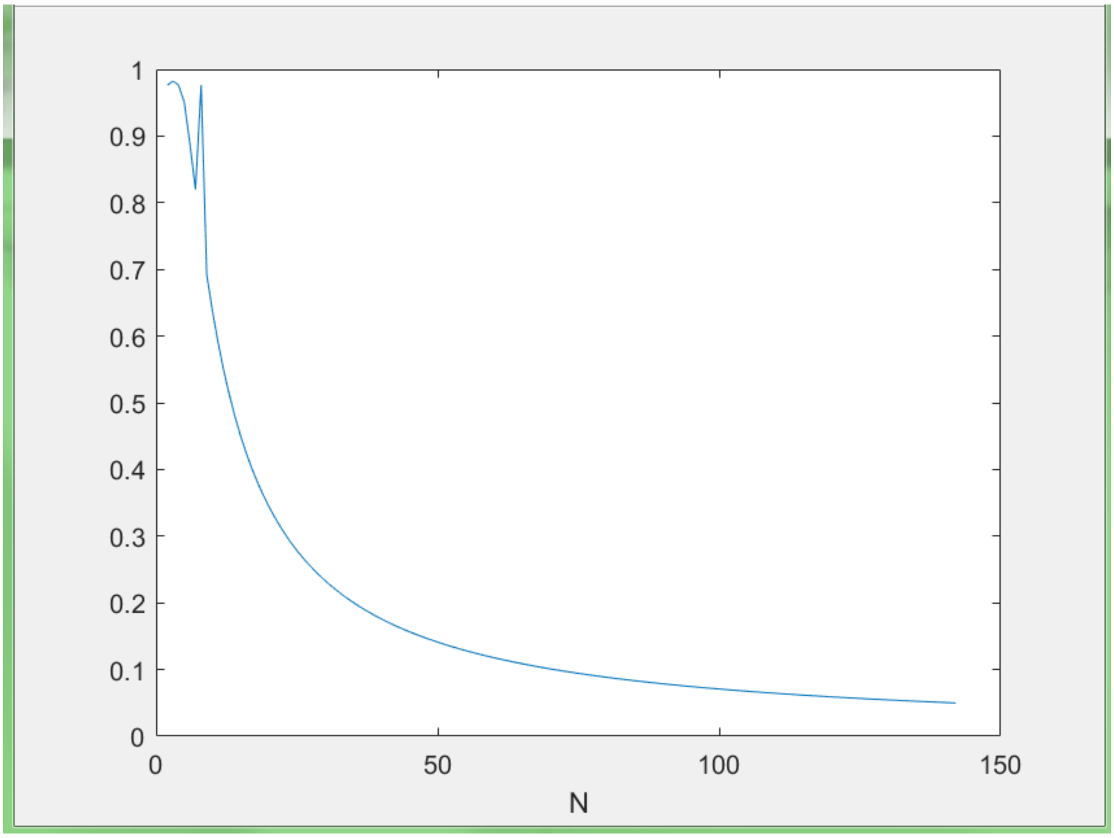


k=2



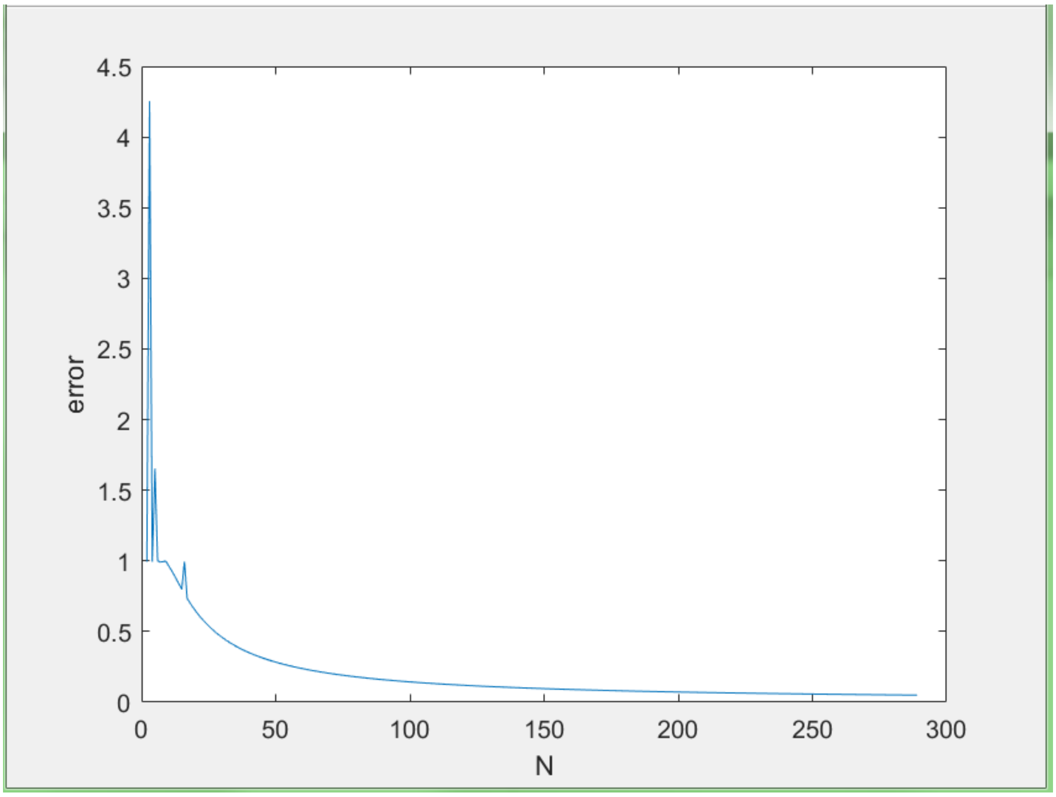
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
60
70
N

k=4

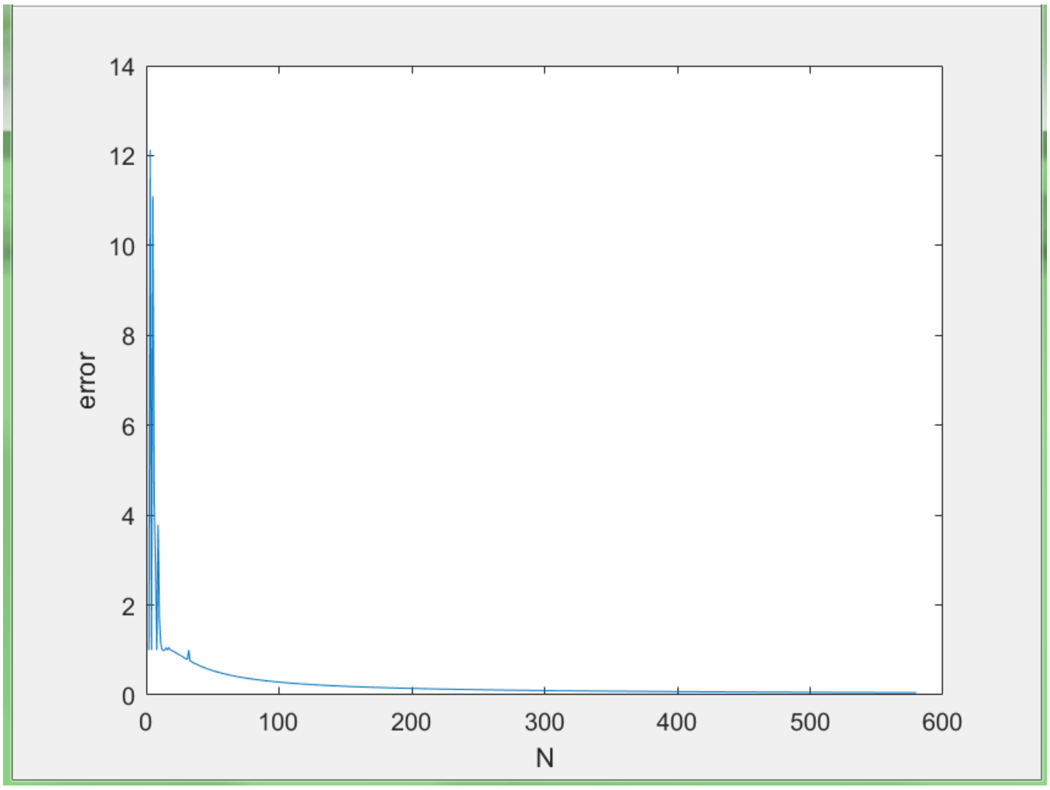


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
50
100
150
N

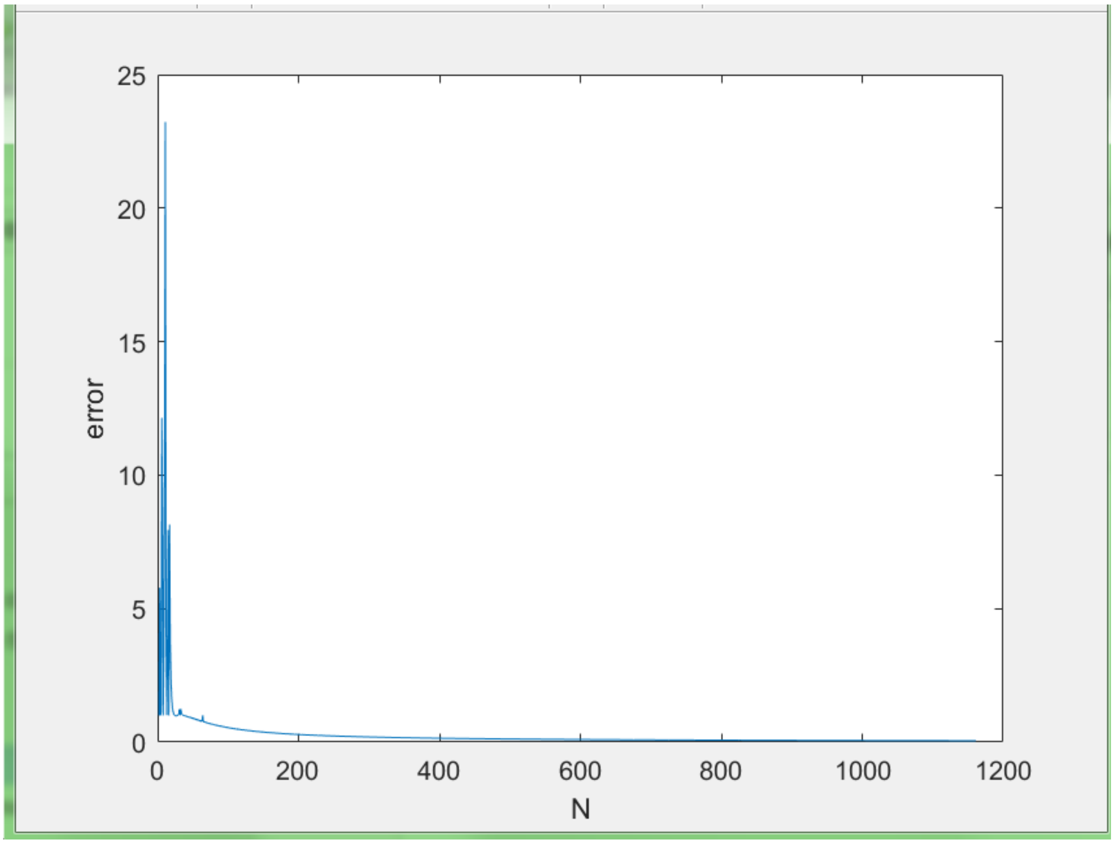
k=8



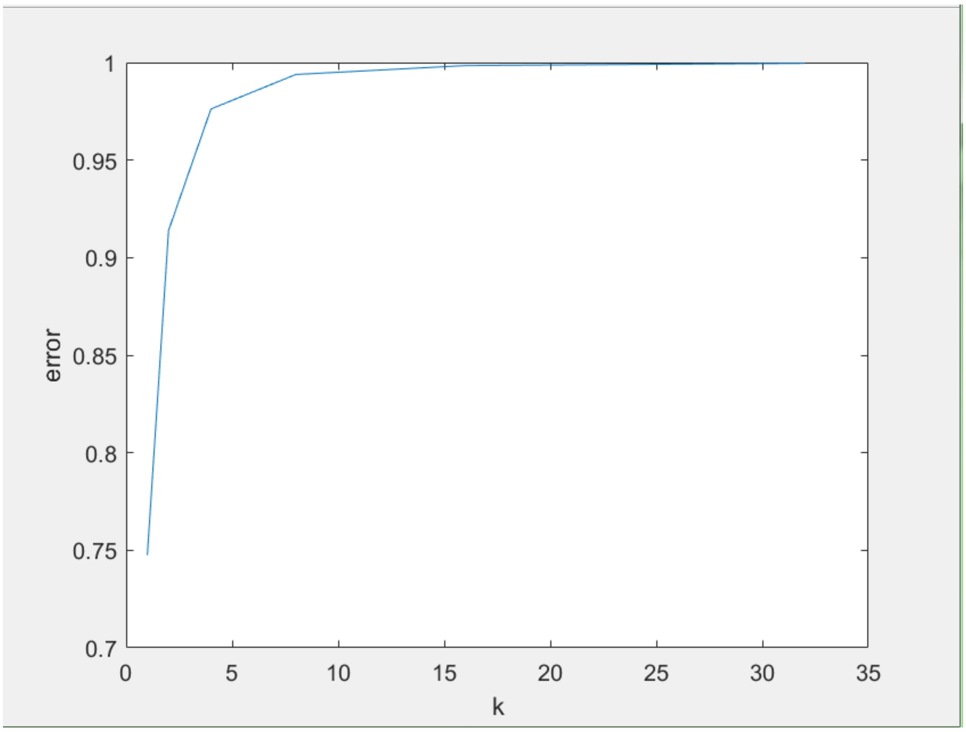
k=16



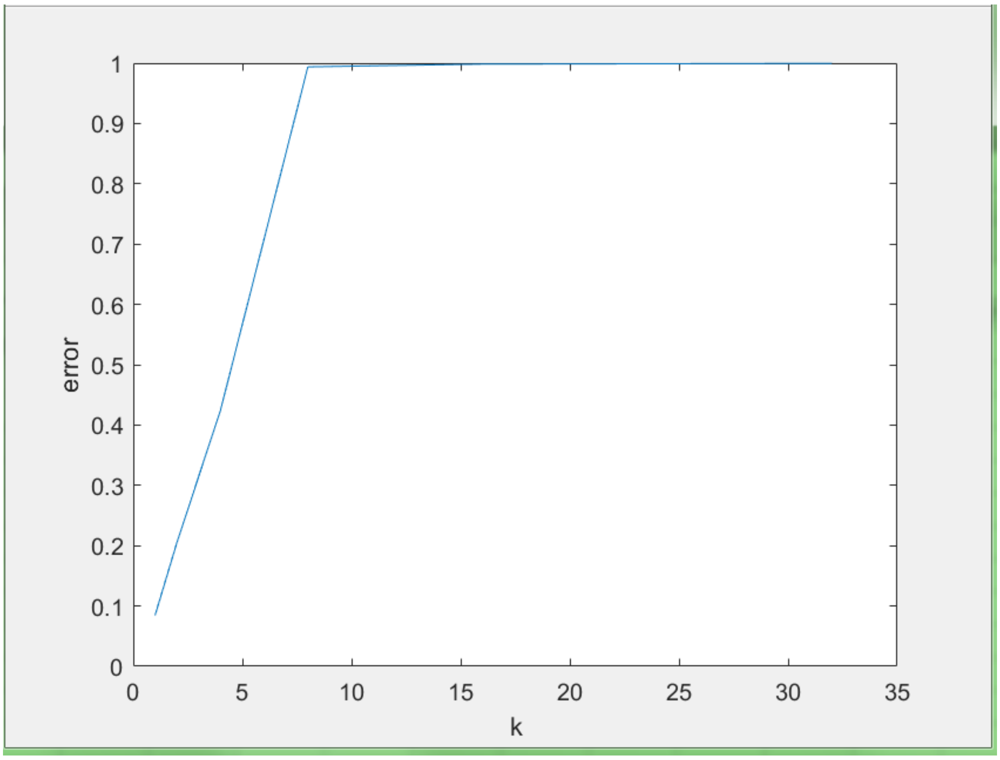
k=32


error
N
0
200
400
600
800
1000
1200
0
5
10
15
20
25

4) **N=2,**

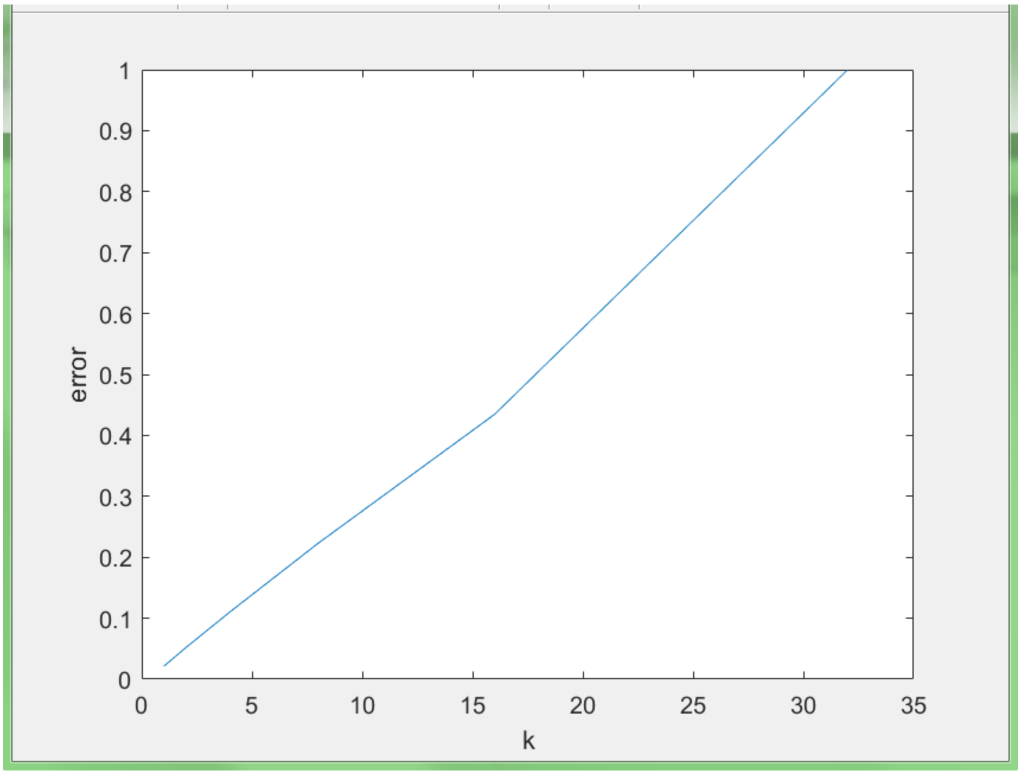

error
k
0
5
10
15
20
25
30
35
0.7
0.75
0.8
0.85
0.9
0.95
1

N=16,



error
k
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
5
10
15
20
25
30
35

N=64,



error
k
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
5
10
15
20
25
30
35

5. Observations and discussion

According to the plots and tables,

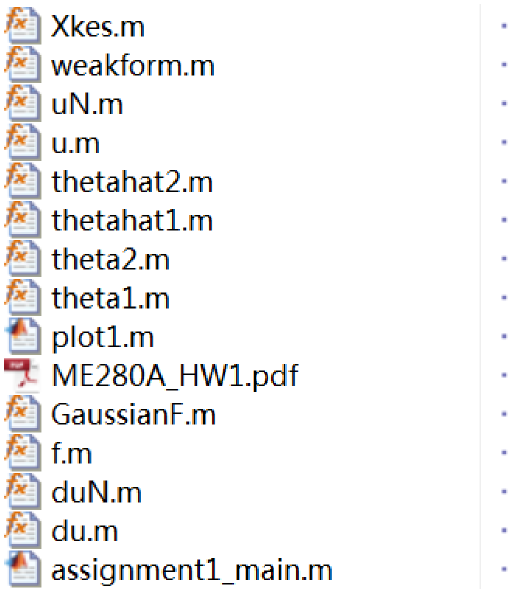
I find when k increases, the Best N increases significantly.

And errors drop down rapidly at the beginning. When k become larger, it is really hard to converge. When require higher accuracy, the cost will become much larger.

Only when k=1, the figure looks smooth. When k become larger, there are some fluctuations.

6. Appendix

Structure:



Weak formulation:

```
function output=weakform(N,k)

Gaussian=[

0.00,0.888;

0.774,0.555;

-0.774,0.555];

E=0.1;

%N=20;

L=1;

%k=1;
```

```
he=L/N;
J=he/2;
%f=@(x)-1*k^2*sin(2*pi*k*x/L);
%u=@(x)(-1*L/(4*E*pi^2))*sin(2*pi*k*x/L)+L*x;
%du=@(x)(-1*L*k/(2*E*pi))*cos(2*pi*k*x/L)+L;
%Xkes=@(x,i)J*x+(2*i-1)*L/N;
%GaussianF=@(x)x;
Ke=zeros(2,2,N);
K=zeros(N+1,N+1);
Re=zeros(2,N);
a=zeros(N+1,1);
R=zeros(N+1,1);
uN=zeros(N,1);
for i=1:N
    Ke(:,:,i)=[E/(J*2),-1*E/(J*2);
        -1*E/(J*2),E/(J*2)];

Re(1,i)=J*Gaussian(1,2)*thetahat1(Gaussian(1,1
))*f(Xkes(Gaussian(1,1),i,J,L,N),k,L);

Re(1,i)=Re(1,i)+J*Gaussian(2,2)*thetahat1(Gaus
sian(2,1))*f(Xkes(Gaussian(2,1),i,J,L,N),k,L);
```

```
Re(1,i)=Re(1,i)+J*Gaussian(3,2)*thetahat1(Gaus
sian(3,1))*f(Xkes(Gaussian(3,1),i,J,L,N),k,L);

Re(2,i)=J*Gaussian(1,2)*thetahat2(Gaussian(1,1
))*f(Xkes(Gaussian(1,1),i,J,L,N),k,L);

Re(2,i)=Re(2,i)+J*Gaussian(2,2)*thetahat2(Gaus
sian(2,1))*f(Xkes(Gaussian(2,1),i,J,L,N),k,L);

Re(2,i)=Re(2,i)+J*Gaussian(3,2)*thetahat2(Gaus
sian(3,1))*f(Xkes(Gaussian(3,1),i,J,L,N),k,L);
end
for i=1:N+1
    if i==1
        K(i,1)=Ke(1,1,i);
        K(i,2)=Ke(1,2,i);
        R(i)=Re(1,i);
        continue
    end
    if(i==N+1)
        K(i,N)=Ke(2,1,i-1);
```

```
        K(i,N+1)=Ke(2,2,i-1);
        R(i)=Re(2,i-1);
        continue
    else
        K(i,i-1)=Ke(2,1,i-1);
        K(i,i)=Ke(2,2,i-1)+Ke(1,1,i);
        K(i,i+1)=Ke(1,2,i);
        R(i)=Re(1,i)+Re(2,i-1);
    end
end
for i=1:N+1
end

Kc=K(2:N,2:N);
Rc=R(2:N);
Rc(N-1)=Rc(N-1)-K(N,N+1);
Kc=sparse(Kc);
a=Kc\Rc;
a=[0;a;1];
x=0:0.01:1;
%figure;
%hold on;
%y1=du(x,L,k,E);
```

```
%y2=duN(x,he,a);
%plot(x,y1);
%plot(x,y2);
%plot(0:0.05:1,a);
%hold off;
uE=@(x)E*(((-
1*L*k/(2*E*pi))*cos(2*pi*k*x/L)+L)-
((a(floor(x*N)+2)-a(floor(x*N)+1)))/he)^2;
duE=@(x)E*(((-
1*L*k/(2*E*pi))*cos(2*pi*k*x/L)+L)^2);
e=(integral(uE,0,L,'ArrayValued',true))^0.5;
uu=(integral(duE,0,L,'ArrayValued',true))^0.5;
eN=e/(integral(duE,0,L,'ArrayValued',true))^0.
5;
output=eN;
%end
```

Thetahat1:

```
function output=thetahat1(x)
output=(1-x)/2;
end
```

Thetahat2:

```
function output=thetahat2(x)
output=(1+x)/2;
end
```

Theta1:

```
function output=theta1(x,he,N)
output=-x/he+floor(x*N)+1;
end
```

Theta2:

```
function output=theta2(x,he,N)
output=x/he-floor(x*N);
end
```

Xkes:

```
function output=Xkes(x,i,J,L,N)
output=J*x+(2*i-1)*J;
end
```

uN:

```
function output=uN(x,he,N)
```

```
i=floor(x*N)+1;
output=a(i)*theta1(x,he,N)+a(i+1)*theta2(x,he,
N);
end
```

duN:

```
function output=duN(x,he,a)
i=floor(x)+1;
output=(a(i+1)-a(i))/he;
end
```

f:

```
function output=f(x,k,L)
output=-1*(k^2)*sin(2*pi*k*x/L);
end
```