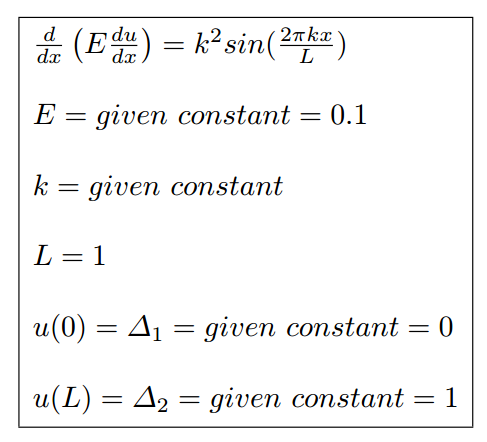
**Report of Project 1: The Basics of FEM**

By Yue Yu, email: [yu\_yue@berkeley.edu](mailto:yu_yue@berkeley.edu)

# Introduction & Objectives

Our primary goal is to solve the problem below using FEM:



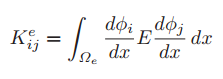
For each k=1,2,4,8,16,32… we will find how many elements (N) are needed to achieve the precision (). And we will find the relationship between N and for each individual k.

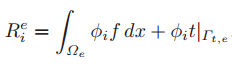
# Procedure

Chart. 1 Flow-chart

The general procedure is show in the flow-chart. The general weak form of 1-dimantional problem is:

In this problem:





Since E is a constant, ϕ is a linear function, dϕ/dx is also a constant. Ke is the same for every element.

Where h=L/N is the length of element.

It is a bit more difficult to calculate Re because f varies on every element. We can transform the domain of to [-1,1] and use Gaussian quadrature. After getting every Ke and Re, we combine they to form a N+1 by N+1 global matric K and a N+1 by 1 global matric R. According to the boundary conditions, a(1)=0 and a(N+1)=L. Then I applied the boundary conditions to K and R. In the equation **KA=R**, K is row 2 to N, column 2 to N of the original K, while R is row 2 to N of original R with the Nth element changed to R(N)-L\*K (N, N+1).

Next we have to solve the equation **KA=R**. We can simply get **A=K\R**. Or we can use Gaussian elimination, which is a bit slower.

After we get A, is known. But we can only describe on each element. So we have to calculate quadrature on every element and sum them when we are trying to get .

# Findings



Fig.1 k=1 Fig.2 k=2



Fig. 3 k=4 Fig.4 k=8



Fig.5 k=16 Fig.6 k=32

Fig.1-6 show the plot of u and for different k and N.

|  |  |
| --- | --- |
| **k** | **Minimum N when eN<=0.05** |
| 1 | 28 |
| 2 | 67 |
| 4 | 142 |
| 8 | 289 |
| 16 | 580 |
| 32 | 1161 |

Table.1

Table.1 shows the minimum N when eN<=0.05 for different k.

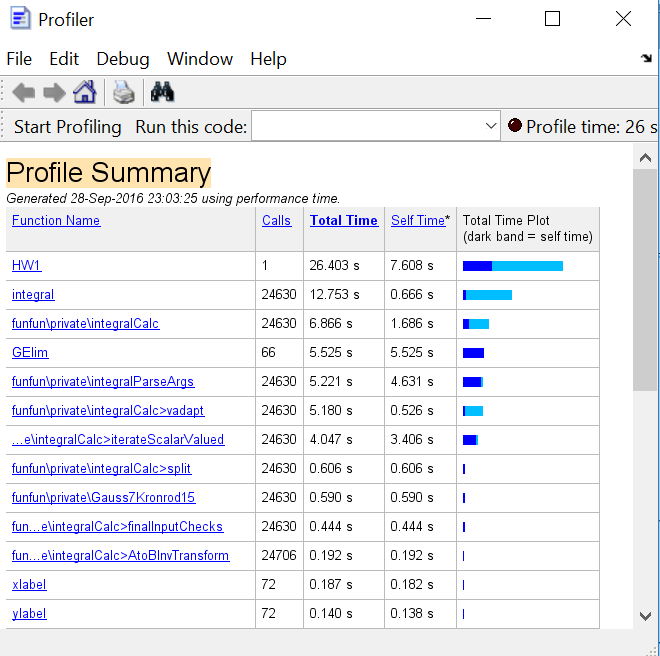


Fig.7 Running time



Fig.8. k=1 Fig.9 k=2



Fig.10 k=4 Fig.11k=8



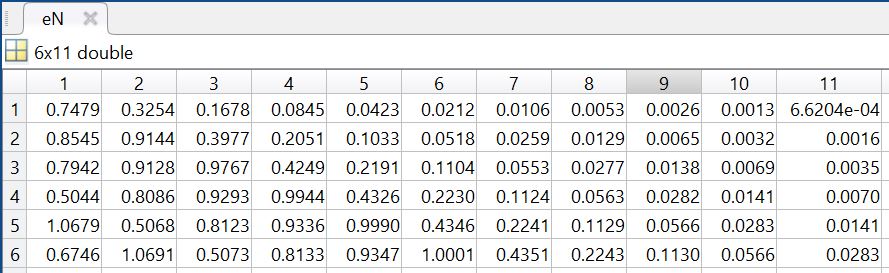
Fig.12 k=16 Fig.13 k=32

Fig.8-13 show the relationship between N and , x-axis is log2(N) and y-axis is ln(eN).

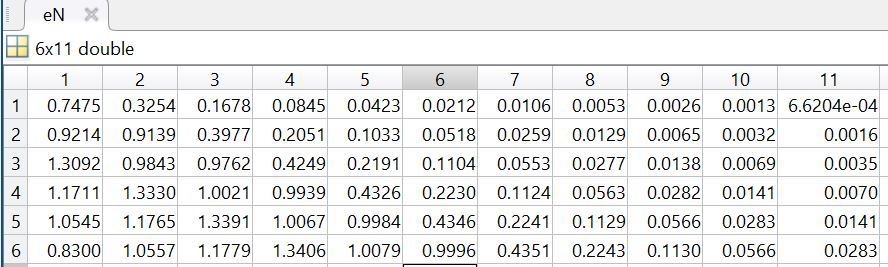
# Observations and discussion

1. When N is smaller than a certain value which is dependent on k, is basically a straight line, therefore stays the same. But when N is big enough, there is a linear relationship between log() and log2(N), which means . (As indicated above in fig.8-13)
2. It is much faster using Gaussian quadrature to figure out than using the built-in function ‘integral’. Calling ‘integral’ will take several seconds in total while it takes less than one second using Gaussian quadrature.

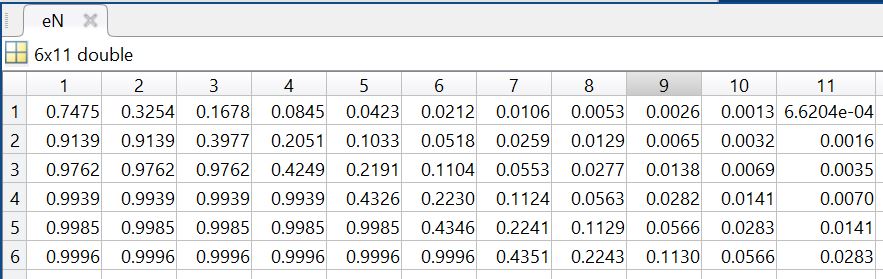
But there is a problem when using Gaussian quadrature. As is described before, when N is small, is a straight line. So should be the same even for different small Ns ( should be equal to each other), while this is not the case when using Gaussian quadrature to figure out . I tried using 4 points and 5 points, they have the same problem. But seems to be accurate enough when N is bigger than some threshold. I guess the reason may be when N is too small, the element length is relatively too long, causing the error of Gaussian quadrature.



(a)



(b)



(c)

Fig.14 using 4-point(a), 5-point(b) Gaussian quadrature and built-in ‘integral’ function(c), the last one can be taken as the correct result. (k=1,2,4,8,16,32, N=2,4,8,16,32,64,128,256,512,1024,2048)

1. I wrote a special function to do Gaussian elimination. Because of the special structure of K in this problem, we only need to do elementary row operations to the row just below and above K(i, i) instead of all the rows below and above, thus saving a lot of time and computation.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| X | X | 0 | 0 | 0 | 0 |
| X | X | X | 0 | 0 | 0 |
| 0 | X | X | X | 0 | 0 |
| 0 | 0 | X | X | X | 0 |
| 0 | 0 | 0 | X | X | X |
| 0 | 0 | 0 | 0 | X | X |

Fig.15 Special structure of K in this problem (only those elements on the diagonal and next to diagonal are non-zero).

# Appendix:

1. Matlab code of HW1.m:

GR2=[0.577350269189626 1;

-0.577350269189626 1];

GR3=[0 0.888888888888889;

0.774596669224148 0.555555555555556;

-0.774596669224148 0.555555555555556];

GR4=[0.339981043584856 0.652145154862546;

0.861136311594053 0.347854845137454;

-0.339981043584856 0.652145154862546;

-0.861136311594053 0.347854845137454];

GR5=[0.000000000000000 0.568888888888889;

0.538469310105683 0.478628670499366;

0.906179845938664 0.236926885056189;

-0.538469310105683 0.478628670499366;

-0.906179845938664 0.236926885056189]; %Gauss weights and points

L = 1; %Length

E = 0.1; %constant

for ki=0:5

figure

for Ni=1:11

k = 2^ki;

N = 2^Ni; %number of elements

h = L/N; %length of each element

f =@(x)-k\*k\*sin(2\*pi\*k\*x/L); %f(x) in the weak form

u =@(x)-L\*L/4/pi/pi/E\*sin(2\*pi\*k\*x/L)+L\*x; %real solution of u(x)

du=@(x)-L\*k/2/pi/E\*cos(2\*pi\*k\*x/L)+L; %du/dx

fun1=@(x)E\*(-L\*k/2/pi/E\*cos(2\*pi\*k\*x/L)+L).^2;

Ke= zeros(2,2,N);

Re= zeros(2,1,N);

for i= 1:N

Ke(:,:,i)= [E/h,-E/h;-E/h,E/h];

%figure out Re using Gaussian quadrature

Re(1,1,i)=GR3(:,2)'\*(f(h/2\*(GR3(:,1)+2\*i-1))\*0.5.\*(1-GR3(:,1))\*0.5\*h);

Re(2,1,i)=GR3(:,2)'\*(f(h/2\*(GR3(:,1)+2\*i-1))\*0.5.\*(1+GR3(:,1))\*0.5\*h);

%fRe1=@(z)f(0.5\*h\*(z+2\*i-1))\*0.5.\*(1-z)\*0.5\*h;

%Re(1,1,i)=integral(fRe1,-1,1);

%fRe2=@(z)f(0.5\*h\*(z+2\*i-1))\*0.5.\*(1+z)\*0.5\*h;

%Re(2,1,i)=integral(fRe2,-1,1);

end

K=zeros(N+1,N+1);

R=zeros(N+1,1);

for i=1:N

K(i,i)=K(i,i)+Ke(1,1,i); %get global K based on local Ke

K(i,i+1)=K(i,i+1)+Ke(1,2,i);

K(i+1,i)=K(i+1,i)+Ke(2,1,i);

K(i+1,i+1)=K(i+1,i+1)+Ke(2,2,i);

R(i,1)=R(i,1)+Re(1,1,i);

R(i+1,1)=R(i+1,1)+Re(2,1,i);

end

R(N)=R(N)-L\*K(N,N+1);

R=R(2:N,1);

K=K(2:N,2:N); %reshape K and R according to boundary conditions

K=sparse(K);

A=zeros(N+1,1);

% A(2:N,1)=K\R;

A(2:N,1)=GElim(K,R);

A(1,1)=0;

A(N+1,1)=L; %get ai for uN

x1=0:0.001:1;

u=u(x1);

x2=0:h:1;

uN=A;

hold on

plot(x1,u,x2,uN)

xlabel('x')

ylabel('uN')

uE=integral(fun1,0,L);

uuNE=0;

for i= 1:N

fun2=@(x)E\*(du(x)-(A(i+1)-A(i))/h).^2;

uuNE=uuNE +integral(fun2,i\*h-h,i\*h);

%uuNE=uuNE + GR5(:,2)'\*fun2(h/2\*(GR5(:,1)+2\*i-1))\*h/2;

end

uE=sqrt(uE); %||u||E

uuNE=sqrt(uuNE);

eN(ki+1,Ni)=uuNE/uE;

end

hold off

end

for k=1:6

figure;

plot(1:11,log(eN(k,:)),'Marker','\*');

xlabel('log2(N)')

ylabel('log(eN)')

end

1. Matlab code of GElim.m:

function A = GElim( K,R )

%doing Gause elimilation for a special sparse matrix

l=length(R);

B=[K,R];

for i=1:l-1;

B(i,:)=B(i,:)/B(i,i);

B(i+1,:)=B(i+1,:)-B(i,:)\*B(i+1,i);

end

B(l,:)=B(l,:)/B(l,l);

for i=l:-1:2;

B(i-1,:)=B(i-1,:)-B(i-1,i)\*B(i,:);

end

A=B(:,end);