functional

December 30, 2020

1 Functional

1.1 Value of functional due to approximated function

Solving for $u : [0, 1] \to \mathbb{R}$

$$\frac{d^2u}{dx^2} + f(x) = 0$$

where $f(x) = \sin(x)$ with boundary conditions $u(0) = 0, u(1) = 0$.

Using FEM for the same and solving the matrix we get from the weak form approximation $v = \sum_{i=0}^{n+1} v_i N_i(x)$, $u = \sum_{i=0}^{n+1} u_i N_i(x)$ and $f = \sum_{i=0}^{n+1} f_i N_i(x)$

$$Ku = F \tag{1}$$

$$K_{(n+2)\times(n+2)} u_{(n+2)\times 1} = M_{(n+2)\times(n+2)} f_{(n+2)\times 1} \tag{2}$$

$$\frac{1}{\Delta x}\begin{pmatrix} 1 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \cdots & 0 \\ 0 & -1 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & 2 & -1 \\ 0 & 0 & \cdots & -1 & 1 \end{pmatrix}\begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_n \\ u_{n+1} \end{pmatrix} = \frac{\Delta x}{6}\begin{pmatrix} 2 & 1 & 0 & \cdots & 0 \\ 1 & 4 & 1 & \cdots & 0 \\ 0 & 1 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & 4 & 1 \\ 0 & 0 & \cdots & 1 & 2 \end{pmatrix}\begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_n \\ f_{n+1} \end{pmatrix} \tag{3}$$

But $u_0 = 0$ and $u_{n+1} = 0$ are known, and the matrix K and M are invertible. Hence removing the first and last equation and modifying it, we get

$$K_{n\times n} u_{n\times 1} = M_{n\times n} f_{n\times 1} \tag{4}$$

$$\frac{1}{\Delta x}\begin{pmatrix} 2 & -1 & \cdots & 0 \\ -1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & 2 & -1 \\ 0 & \cdots & -1 & 2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} = \frac{\Delta x}{6}\begin{pmatrix} 4 & 1 & \cdots & 0 \\ 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & 4 & 1 \\ 0 & \cdots & 1 & 4 \end{pmatrix}\begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} + \begin{pmatrix} \frac{f_0 \Delta x}{6} - \frac{-u_0}{\Delta x} \\ 0 \\ \vdots \\ \frac{f_{n+1}\Delta x}{6} - \frac{-u_{n+1}}{\Delta x} \end{pmatrix} \tag{5}$$

Now RHS is known, LHS square matrix is also invertible, just solve to get u.

1.1.1 Following is the code

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

1.1.2 Function for solving

[2]:
```
def solution(n,dx,f,ua,ub): # f is of size n+2
    # Solution of Ku = Mf system of linear equations
    ## Make K
    K = 2*np.identity(n)
    dk = -1*np.identity(n-1)
    row,col = 0,1
    K[row:row+dk.shape[0], col:col+dk.shape[1]] += dk
    row,col = 1,0
    K[row:row+dk.shape[0], col:col+dk.shape[1]] += dk
    K *= 1/dx

    ## Make M
    M = 2*dx*np.identity(n)/3
    dm = dx*np.identity(n-1)/6
    row,col = 0,1
    M[row:row+dm.shape[0], col:col+dm.shape[1]] += dm
    row,col = 1,0
    M[row:row+dm.shape[0], col:col+dm.shape[1]] += dm

    ## Solve Ku = Mf
    #### Add dx*f[0]/6+(ua/dx) to first element of Mf and dx*f[-1]/6+(ub/dx) to␣
↪last element
    rhs = np.matmul(M,f[1:-1])
    rhs[0] += dx*f[0]/6 - (-ua/dx) # = M[1,0]*f[0] - K[1,0]*u[0]
    rhs[-1]+= dx*f[-1]/6 - (-ub/dx) # = M[n,n+1]*f[n+1] - K[n,n+1]*u[n+1] or =␣
↪M[-2,-1]*f[-1] - K[-2,-1]*u[-1]

    #### Solve u = K^-1*rhs
    u = np.matmul(np.linalg.inv(K),rhs)
    u = np.insert(u,[0,n],[ua,ub])

    return u
```

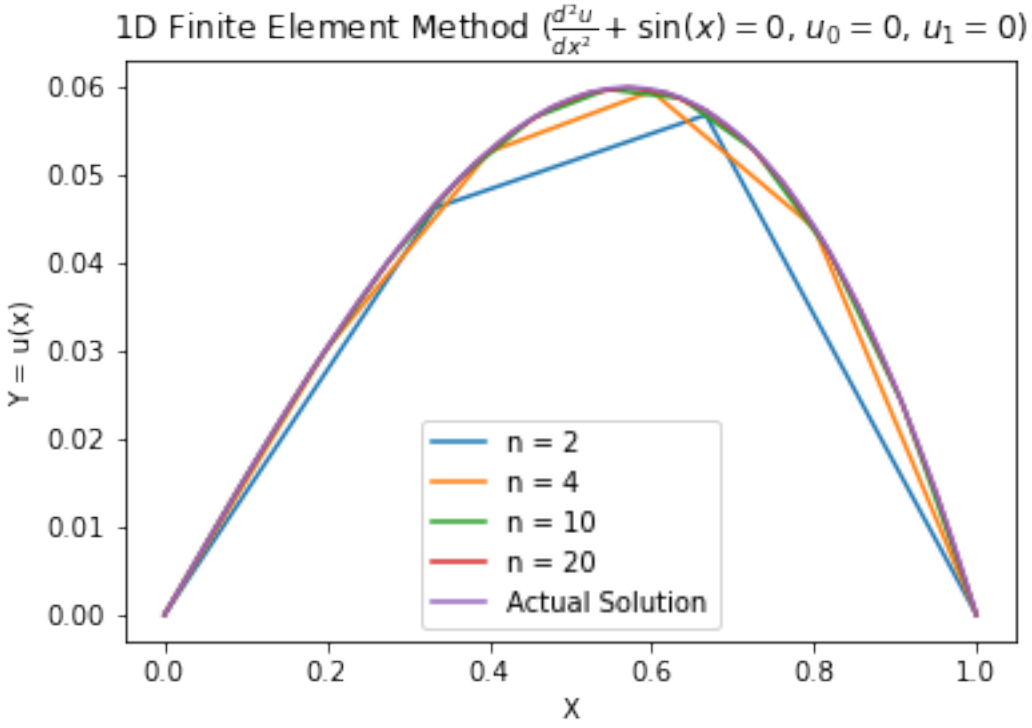
1.1.3 Main Body

[3]:
```
a = 0
b = 1
ua = 0
ub = 0
n1 = [2,4,10,20]

for n in n1:
    x = np.linspace(a,b,n+2)
    dx = (b-a)/(n+1)
    f = np.sin(x)
```

```
    u = solution(n,dx,f,ua,ub)
    plt.plot(x,u,label = f'n = {n}')

## Plot the solution
#plt.plot(x,u)
contx = np.linspace(0,1,101)
plt.plot(contx,np.sin(contx)-np.sin(1)*contx,label = 'Actual Solution')
plt.legend()
plt.title(r'1D Finite Element Method ($\frac{d^2u}{dx^2}+\sin(x)=0$, $u_0=0$,␣
 ↪$u_1=0$)')
plt.xlabel('X')
plt.ylabel('Y = u(x)')
plt.show()
```



1.1.4 Functional value

Function for getting the value of functional that takes in arguments the array of discretized domain x and its correspondind function value $u(x)$. It approximates the function $u(x)$ linearly between

two consecutive points and calculates $I(u(x))$ using trapezoidal rule

$$I(u(x)) = \int_0^1 \left[\frac{u'(x)^2}{2} - g(x)u(x)\right] dx \tag{6}$$

$$= \int_0^1 \left[\frac{u'(x)^2}{2} - \sin(x)u(x)\right] dx \tag{7}$$

$$= \sum_{i=0}^{n} \int_{x_i}^{x_{i+1}} \left[\frac{u'(x)^2}{2} - \sin(x)(u'(x)(x - x_i) + y_i)\right] dx \tag{8}$$

$$= \sum_{i=0}^{n} \int_{x_i}^{x_{i+1}} \left[\frac{c_i^2}{2} - (y_i - c_i x_i)(\sin(x) + c_i x \sin(x))\right] dx \tag{9}$$

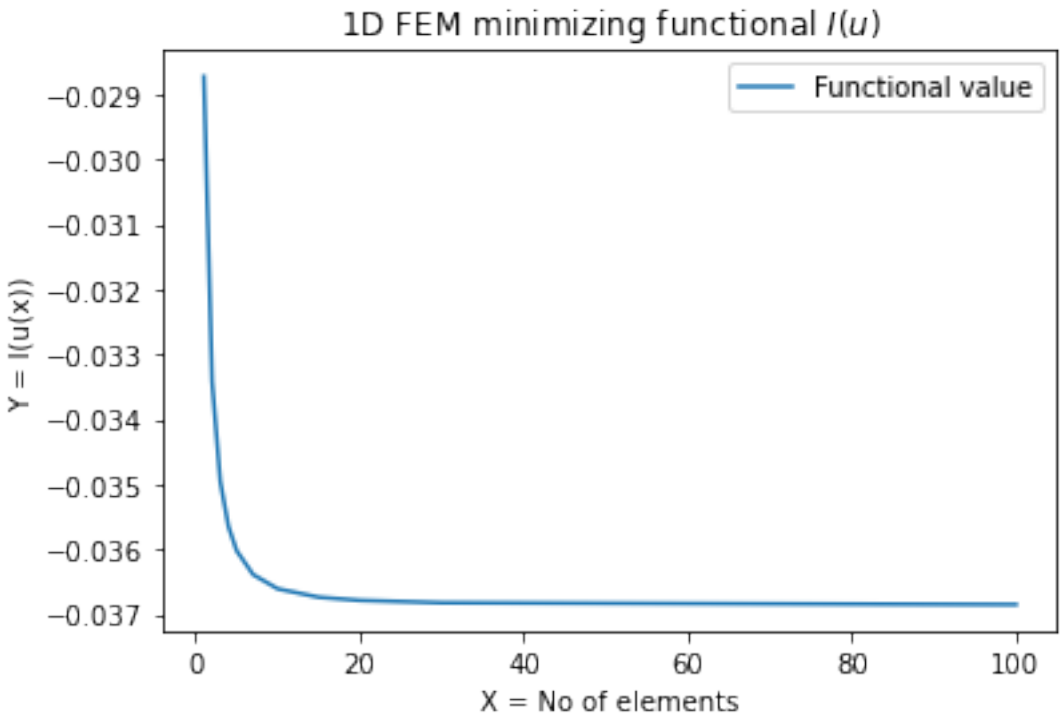
$$= \sum_{i=0}^{n} \left[\frac{c_i^2 \Delta x}{2} - (y_i - c_i x_i)\{-\Delta \cos(x) + c_i[-\Delta(x \cos(x)) + \Delta \sin(x)]\}\right] dx \tag{10}$$

where $c_i = u'(x) = \frac{\Delta y}{\Delta x} = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}$, which is constant for given i.

[4]:
```
def functional(u,x):
    # u and x is the 1D array of at least 2 points of function value and dx is the constant difference of the division of domain
    # calculates integral by trapezoidal rule by approximating the function between two consecutive points to be linear.
    x0,x1,u0,u1 = x[:-1],x[1:],u[:-1],u[1:]
    c = (u1-u0)/(x1-x0)
    ans = 0.5*c**2*(x1-x0) - (u0-c*x0)*(-np.cos(x1)+np.cos(x0) + c*(-x1*np.cos(x1)+x0*np.cos(x0) + np.sin(x1)-np.sin(x0)))
    return np.sum(ans)
```

[5]:
```
n2 = [1,2,3,4,5,7,10,15,20,30,100]
fun_value = []
for n in n2:
    x = np.linspace(a,b,n+2)
    dx = (b-a)/(n+1)
    f = np.sin(x)
    u = solution(n,dx,f,ua,ub)
    func = functional(u,x)
    fun_value.append(func)

## Plot the solution
#plt.plot(x,u)
plt.plot(n2,fun_value,label = 'Functional value')
plt.legend()
plt.title(r'1D FEM minimizing functional $I(u)$')
plt.xlabel('X = No of elements')
plt.ylabel('Y = I(u(x))')
plt.show()
print(f'Minimum value of the functional is {fun_value[-1]}')
```




```
Minimum value of the functional is -0.03684876262178659
```