METODY OBLICZENIOWE, ćw. 3

Po zapoznaniu się z wykładem na temat algorytmu MES na przykładzie zadania 1D (https://www.cce.pk.edu.pl/~witek/MetObl/handouts.pdf) zastosować dwa elementy skończone do aproksymacji rozwiązania zadania pręta przy wykorzystaniu liniowych funkcji kształtu:

$$-AE\frac{d^2u}{dx^2} = 5 * N, \qquad x \in (0, L)$$
$$u(0) = 0$$
$$AEu'(L) = N$$

gdzie $L = 2*N$, N jest ostatnią cyfrą z numeru indeksu (w przypadku '0' przyjąć '1'), *A* jest polem przekroju pręta, *E* modułem Younga.

Narysować wykresy przemieszczenia, odkształcenia, naprężenia i siły osiowej.

Przykład programu, realizującego część podobnego zadania, zamieszczono poniżej.

```
# Rozwiązanie MES dla: -u"=x + war. brz.(-u'(0)=1, u(2)=0)
import numpy as np
import matplotlib.pyplot as plt
plt.ion()

# Dane i dyskretyzacja
L = 2
S = 1
Nel = 2
LSS = Nel + 1
h = L / Nel
coord = np.linspace(0, L, LSS)

# Brzegowy warunek kinematyczny (podpora)
kbc = [LSS -1, 0] # numer znanego stopnia swobody i jego wartość
( '-1' uwzględnia indeksowanie w Pythonie zaczynające się od zera)

# Agregacja
KG = np.zeros((LSS, LSS))
f = np.zeros(LSS)
f[0] = S # uwzględnienie statycznego warunku brzegowego
(obciążenia w węźle)
```

```
# Macierz elementu
Kel = np.array([[1, -1],
 [-1, 1]])

# Element 1
Pel1 = np.array([1/6, 1/3]) # wektor elementu
KG[0:2, 0:2] = KG[0:2, 0:2] + Kel
f[0:2] = f[0:2] + Pel1

# Element 2
Pel2 = np.array([2/3, 5/6])
KG[1:3, 1:3] = KG[1:3, 1:3] + Kel
f[1:3] = f[1:3]+ Pel2

print(KG,f)
# Uwzględnienie warunku kinematycznego i rozwiązanie układu równań
algebraicznych
n = kbc[0]
w = kbc[1]
f = f - KG[:, n] * w
KG[:, n] = 0
KG[n, :] = 0
KG[n, n] = 1
f[n] = w

# Rozwiązanie układu równań
u = np.linalg.solve(KG, f)

# Wykres aproksymacji MES
plt.figure(1)
plt.plot(coord, u, 'bo', label='Wartości w węzłach')
plt.xlabel('x')
plt.ylabel('u_h')
dx = 0.1

# Element 1
x1, x2 = coord[0], coord[1]
u1, u2 = u[0], u[1]
x = np.arange(x1, x2 + dx, dx)
uh1 = u1 * (x - x2) / (x1 - x2) + u2 * (x - x1) / (x2 - x1)
plt.plot(x, uh1, 'b--')

# Element 2
x2, x3 = coord[1], coord[2]
u2, u3 = u[1], u[2]
x = np.arange(x2, x3 + dx, dx)
uh2 = u2 * (x - x3) / (x2 - x3) + u3 * (x - x2) / (x3 - x2)
plt.plot(x, uh2, 'b--')
```