METODY OBLICZENIOWE, ćw. 4

Na podstawie wykładu na temat MES dla stacjonarnego przepływu ciepła (https://www.cce.pk.edu.pl/~witek/MetObl/handouts.pdf) dla samodzielnie wybranego obszaru 2D z wewnętrznym źródłem ciepła obliczyć aproksymację rozkładu temperatury dla dwóch różnych dyskretyzacji, a następnie porównać temperaturę i jej gradient w wybranym punkcie. Przyjąć, że temperatura jest znana na pewnej części brzegu, na innej części dany jest stały niezerowy strumień ciepła, a na pozostałej części brzegu strumień ciepła jest zerowy.

Przygotować programy w języku Python rozwiązujące powyższe zagadnienie dla dwóch różnych dyskretyzacji (kilka i kilkanaście elementów skończonych). Przyjąć tak dyskretyzację, żeby obszar ze źródłem ciepła pokrywał się z jednym elementem na rzadszej siatce (nr 16 w załączonym przykładzie) i dwoma elementami na gęstszej siatce. Niezerowy strumień ciepła na brzegu niech będzie znany na jednej z krawędzi.

W sprawozdaniu narysować przyjęty obszar z zaznaczonymi wszystkimi danymi definiującymi zadanie, procedury przygotowane w Pythonie, otrzymane wyniki oraz dyskusję ich poprawności oraz dokładności.

Do przygotowania kodu wykorzystać poniższy program oraz procedury Calfem.

```
# py -m pip install calfem-python

# Stacjonarny przepływ ciepła
# T=x*(x-8) na dolnej krawędzi
# qn - niezerowe na górnej krawędzi
# qn=0 na pozostałych krawędziach
# Q - niezerowa intensywność źródła ciepła w elemencie 16

import numpy as np
import calfem.core as cfc
import matplotlib.tri as tri
import matplotlib.pyplot as plt

plt.ion()

# 1. SFORMUŁOWANIE ZADANIA (równanie Laplace'a)
# Definicja parametrów fizycznych
k = 0.7  # Współczynnik przewodzenia ciepła
qn = 5  # Strumień cieplny na górnej krawędzi
Q = 8  # Intensywność źródła ciepła w elemencie 16
nn = 16  # Numer elementu z niezerowym źródłem ciepła
```

```
# Warunki brzegowe temperatury na dolnej krawędzi
Tbc = np.array([[1, 0], [6, -16], [7, 0]])



# 2. DYSKRETYZACJA
Nodes = 15  # Liczba węzłów
Nel = 18  # Liczba elementów
Ndofs = Nodes  # Liczba stopni swobody (tu równa liczbie węzłów)

# Współrzędne węzłów
Coord = np.array([
    [8, 0],
    [6, 2],
    [8, 4],
    [5, 4],
    [4, 3],
    [4, 0],
    [0, 0],
    [2, 2],
    [3, 4],
    [4, 4],
    [4, 5],
    [0, 4],
    [2, 6],
    [4, 8],
    [0, 8]])

# Topologia elementów
Edof = np.array([
    [1, 1, 3, 2],   # Element 1: węzły 1, 3, 2
    [2, 2, 3, 4],
    [3, 1, 2, 6],
    [4, 2, 4, 5],
    [5, 2, 5, 8],
    [6, 6, 2, 8],
    [7, 6, 8, 7],
    [8, 8, 5, 9],
    [9, 5, 4, 10],
    [10, 5, 10, 9],
    [11, 9, 10, 11],
    [12, 8, 9, 13],
    [13, 8, 13, 12],
    [14, 8, 12, 7],
    [15, 12, 13, 15],
    [16, 15, 13, 14],
    [17, 14, 13, 11],
    [18, 11, 13, 9]  ])
```

```
# 3. RYSOWANIE SIATKI ELEMENTÓW
plt.figure(1)

# Rysowanie elementów
for elem in Edof:
    elem_id, n1, n2, n3 = elem
    x = [Coord[n1-1, 0], Coord[n2-1, 0], Coord[n3-1, 0], Coord[n1-1, 0]]
    y = [Coord[n1-1, 1], Coord[n2-1, 1], Coord[n3-1, 1], Coord[n1-1, 1]]
    plt.plot(x, y, 'k-')  # Obrys elementu
    centroid_x = np.mean([Coord[n1-1, 0], Coord[n2-1, 0], Coord[n3-1, 0]])
    centroid_y = np.mean([Coord[n1-1, 1], Coord[n2-1, 1], Coord[n3-1, 1]])
    plt.text(centroid_x, centroid_y, str(elem_id), color='red', fontsize=10)  # Numer elementu


# Rysowanie węzłów
plt.plot(Coord[:, 0], Coord[:, 1], 'bo')  # Węzły jako niebieskie kropki
for i, (x, y) in enumerate(Coord, 1):
    plt.text(x, y, str(i), color='blue', fontsize=12)  # Numer węzła

plt.axis("equal")
plt.title("Dyskretyzacja z numeracją węzłów i elementów")



# 4. AGREGACJA MACIERZY GLOBALNYCH
KG = np.zeros((Ndofs, Ndofs))
f = np.zeros(Ndofs)

for iel in range(Nel):
    fel = np.zeros(3)
    if iel + 1 == nn:    # uwzględnienie źródła ciepła
        w = Coord[Edof[iel, 1:4] - 1, :]
        a = w[1] - w[0]
        b = w[2] - w[0]
        P = 0.5 * abs(np.linalg.det(np.array([a, b])))
        fel = fel + (1/3)*P*Q* np.ones(3)
    if iel + 1 == nn:   # uwzględnienie strumienia ciepła
        w = Coord[Edof[iel, 1:4] - 1, :]
        x1, y1 = w[0]
        x3, y3 = w[2]
        d = np.sqrt((x1-x3)**2+(y1-y3)**2)
        fel = fel + np.array([1/2*d*qn, 0, 1/2*d*qn])

    # Poprawne indeksowanie
    Ex = Coord[Edof[iel, 1:4] - 1, 0]  # Współrzędne X węzłów elementu
    Ey = Coord[Edof[iel, 1:4] - 1, 1]  # Współrzędne Y węzłów elementu

    Kel = cfc.flw2te(Ex, Ey, [1], np.array([[k, 0], [0, k]]))  # Macierz elementu
    KG, f = cfc.assem(Edof[iel, 1:], KG, Kel, f, fel)  # Agregacja
```

```
# 5. ROZWIĄZANIE UKŁADU RÓWNAŃ
bcDofs = (Tbc[:, 0] - 1)  # Indeksy
bcValues = Tbc[:, 1]  # Wartości warunków brzegowych

# Uwzględnienie warunku brzegowego znanej temperatury
for i in range(len(bcDofs)):
    dof = bcDofs[i]
    KG[dof, :] = 0
    KG[dof, dof] = 1
    f[dof] = bcValues[i]


u = np.linalg.solve(KG, f)
print("Minimalna temperatura:", min(u))
print("Maksymalna temperatura:", max(u))



# 6. POSTPROCESSING - wykres temperatury
triang = tri.Triangulation(Coord[:, 0], Coord[:, 1], Edof[:, 1:] - 1)
fig, ax = plt.subplots(figsize=(8, 6))
contour = ax.tricontourf(triang, u, levels=50, cmap="jet")
plt.axis("equal")
cbar = fig.colorbar(contour)
ax.set_title("Rozkład temperatury")
```