SPRAWOZDANIE Z PROJEKTU

METODY ELEMENTÓW SKOŃCZONYCH

Klaudia Wrona
Inżynieria Obliczeniowa
293128

1. Cel i tematyka projektu

Celem projektu było zapoznanie się z algorytmem MES oraz projekcja rozkładu temperatury w obiekcie w przestrzeni 2D.

2. Opis klas

Do wykonania projektu zdecydowałam się obrać język Java.

Projekt składa się z 9 klas, parametry zadane w modelu wczytane zostały z pliku tekstowego

Pokazując wyniki poszczególnych macierzy, zakładam takie dane, jakich używano w pliku excel Jak2D.

Global_data

Jest to klasa przechowująca dane z pliku .txt.

Zawiera parametry następujących danych (wraz z wyjaśnianiami)

double H;	Wysokość siatki
double L;	Szerokość siatki
double nH;	Liczba węzłów na wysokość
double nL;	Liczba węzłów na szerokość
int PointQuantity;	Liczba określająca ilu punktowy schemat całkowania zostaje zastosowany w modelu
double gauss[][];	Tablica zawierająca współrzędne oraz wagi ze schematu Gaussa. W pierwszym wierszu zawiera współrzędne, w drugim wagi
double points[][];	Tablica punktów z przypisanymi im współrzędnymi
doubleK;	Współczynnik przewodzenia ciepła

double c;	Pojemność cieplna
double ro;	Gęstość
double alfa;	Współczynnik wymiany ciepła
double initTemp;	Temperatura początkowa
double simTime;	Czas procesu
double dT;	Krok czasowy
double ambTemp;	Temperatura otoczenia

Gauss 2-punktowy schmat całkowania

Współrzędna	$\frac{-1}{\sqrt{3}}$	$\frac{1}{\sqrt{3}}$
Waga	1	1

Gauss 3-punktowy schemat całkowania

Współrzędna	-0,77	0	0,77
Waga	$\frac{5}{9}$	$\frac{8}{9}$	$\frac{5}{9}$

Siatka_Grid

Siatka Grid to klasa, która zawiera w sobie tablicę elementów oraz tablicę węzłow. Ma wymiary ilość węzłów na szerokość * ilość węzłów na wysokość. Ilość elementów obliczana jest analogicznie (ilość węzłów na szerokość -1) * (ilość węzłów na wysokość -1).

Node

Jest to klasa reprezentująca pojedynczy węzeł. Ma on swoje ID, współrzędną X i Y. W przypadku mojego programu węzły numerowane są od 1.

ElementUni

Element uniwersalny wykorzystywany do liczenia Jakobianu. Jest to klasa odpowiadająca za wyliczenie funkcji kształtu oraz ich pochodnych po współrzędnych ksi i eta

Ze względu na to, że projekt dotyczy elementu 2D zastosowałam wzory:

```
shapeFunction[i][0] = 0.25 * (1 - ksi) * (1 - eta);
shapeFunction[i][1] = 0.25* (1 + ksi) * (1 - eta);
```

```
shapeFunction[i][2] = 0.25 * (1 + ksi) * (1 + eta);
shapeFunction[i][3] = 0.25  * (1 - ksi) * (1 + eta);
```

Ksi i eta zadane zostały w tablicy gauss w danych globalnych.

```
SHAPE FUNCTIONS:
0.6220084679281462   0.16666666666666663   0.044658198738520435   0.16666666666666663
0.16666666666666663   0.6220084679281462   0.16666666666666663   0.044658198738520435
0.044658198738520435   0.16666666666666663   0.6220084679281462   0.16666666666666663
0.16666666666666663   0.044658198738520435   0.16666666666666663   0.6220084679281462
```

Funkcje kształtu oraz ich pochodne mają wymiar [ilość punktów całkowania*ilość punktów całkowania] x [ilość funkcji kształtu].

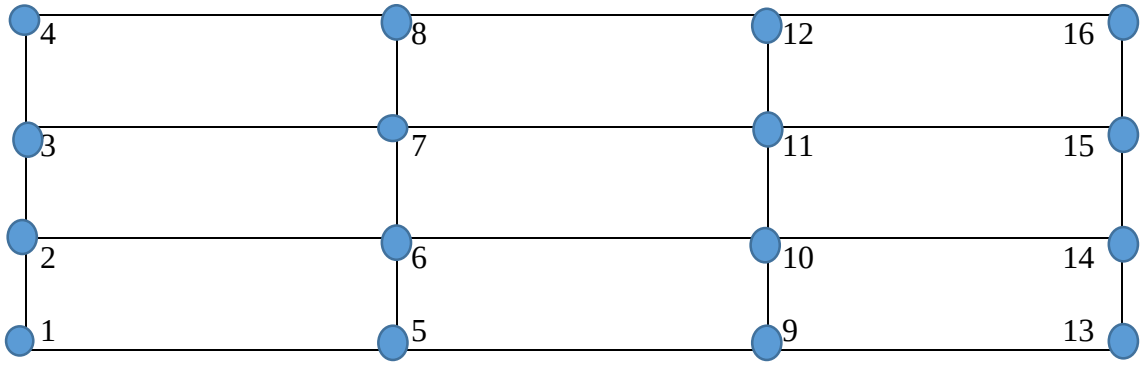
```
dNdKSI:
-0.39433756729740643   0.39433756729740643   0.10566243270259354   -0.10566243270259354
-0.39433756729740643   0.39433756729740643   0.10566243270259354   -0.10566243270259354
-0.10566243270259354   0.10566243270259354   0.39433756729740643   -0.39433756729740643
-0.10566243270259354   0.10566243270259354   0.39433756729740643   -0.39433756729740643

dNdEta:
-0.39433756729740643   -0.10566243270259354   0.10566243270259354   0.39433756729740643
-0.10566243270259354   -0.39433756729740643   0.39433756729740643   0.10566243270259354
-0.10566243270259354   -0.39433756729740643   0.39433756729740643   0.10566243270259354
-0.39433756729740643   -0.10566243270259354   0.10566243270259354   0.39433756729740643
```

Element

Jest to klasa elementów mojej siatki. Zawiera tablice węzłów (4 węzły).

```
nodeID[0] = ID;
nodeID[3] = ID + 1;
nodeID[1] = ID + (int) (nH);
nodeID[2] = ID + (int) (nH) + 1;
```

Graficznie ID węzłów można przedstawić w następujący sposób:



W elemencie ustawiane są też flagi powierzchni (czy należy do brzegowych), wykorzystując przy tym ID węzła.

W elemencie znajduje się też ustawianie dużej macierzy Jakobiego, jej odwrotności oraz samego jakobianu dla danego elementu ze składowych pozyskiwanych z klasy Jakobian.

Obliczana jest też macierz H, macierz C, wektor P oraz macierz H_BC, czyli macierz H uwzględniająca warunki brzegowe, wykorzystująca flagi powierzchni.
Wszelkie macierze w elemencie są sumą wcześniej wyliczonych macierzy (z klasy Jakobian) w poszczególnych punktach całkowania oraz zsumowaniem ich w duże macierze.
Przykładowo macierz H obliczana jest przez zsumowanie *K*(dNdX* transponowane dNdX + dNdY* transponowane dNdY) * det J* dla każdego punktu całkowania.

```
Macierz H
20.000000000000004   -5.000000000000002   -9.999999999999998   -5.000000000000001
-5.000000000000002   20.0   -5.0   -9.999999999999998
-9.999999999999998   -5.0   20.0   -5.000000000000002
-5.000000000000001   -9.999999999999998   -5.000000000000002   20.0


Macierz big C
379.16666666666674   189.58333333333331   94.79166666666663   189.58333333333331
189.58333333333331   379.1666666666667   189.58333333333331   94.79166666666663
94.79166666666663   189.58333333333331   379.16666666666674   189.58333333333331
189.58333333333331   94.79166666666663   189.58333333333331   379.1666666666667

Macierz H_BC
0.41666666666666674 0.10416666666666666 0.0 0.10416666666666666
0.10416666666666666 0.20833333333333337 0.0 0.0
0.0 0.0 0.0 0.0
0.10416666666666666 0.0 0.0 0.20833333333333337
```

Powyższe macierze stworzone zostały dla elementu 1.

Jakobian

Jest to klasa, która wylicza poszczególne macierze w kolejnych punktach całkowania. Obliczam w niej również składowe jakobianu, czyli dY/dEta, dY/dKsi, dX/dEta, dX/dKsi , używam do tego wzorów interpolacyjnych.

```
dYdEta = dYdEta + nodes[elements[numberElement].nodeID[i]].getY() * elementuni.dNdEta[punktCalkowania][i]
```

Tworzę macierz odwrotną, wg zasady 1/det[J] * macierz dopełnień oraz macierz dNdX^2 + dNdY^2 pomnożoną przez K i jakobian, której będę używać później w elemencie tworząc macierz H.

JacobiMethod

Jest to klasa, która rozwiązuje układ równań A*x = b metodą Jakobiego.

Użyłam tej metody ze względu na mniejszą złożoność obliczeniową niż przy transformacji układu do x= A^(-1)*b, ze względu na to, że A ma wymiar [nL*nL][nH*nH], więc liczenie macierzy odwrotnej byłoby zbyt skomplikowane.

Aggregation

Jest to klasa odpowiadająca za tworzenie macierzy zagregowanej H oraz P. Zagregowana macierz H składa się z H (dla każdego elementu) oraz H_BC (macierzy warunków brzegowych każdego elementu). Globalny P agreguje wartości P lokalnych.

Określane są parametry A i b, gdzie

$$A = H + \frac{C}{dt}$$

$$b = \frac{C}{dt} * \{T\} + P$$

Następnie w tej klasie rozwiązywany jest układ równań (wywołując JacobiMethod) w każdym kroku czasowym. Wektor temperatur wypełniany jest całkowicie temperaturą początkową (initTemp z Global_Data). Następnie podstawiany tam jest nowy, wyliczony przy użyciu JacobiMethod.

Na ekran wyświetlam jedynie największą oraz najmniejszą temperaturę.

Test case 1

Test case 2d transient solution -

- 100 – initial temperature
- 500 – simulation time [s],
- 50 – simulation step time [s],
- 1200 – ambient temperature [C],
- 300 – alfa [W/m^{2}K],
- 0.100 – H [m],
- 0.100 – B [m],
- 4 – N_H,
- 4 – N_B,
- 700 – specific heat [J/(kg°C)],
- 25 – conductivity [W/(m°C)],
- 7800 – density [kg/m3].

Global C

```
674.0740740740739 337.03703703703695 0.0 0.0 337.0370370370369 168.51851851851842 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
337.03703703703695 1348.1481481481478 337.03703703703695 0.0 168.51851851851842 674.0740740740739 168.51851851851842 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 337.03703703703695 1348.1481481481478 337.03703703703695 0.0 168.51851851851842 674.0740740740739 168.51851851851842 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 337.03703703703695 674.074074074074 0.0 0.0 168.51851851851842 337.03703703703695 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
337.0370370370369 168.51851851851842 0.0 0.0 1348.1481481481478 674.0740740740739 0.0 0.0 0.0 337.0370370370369 168.51851851851842 0.0 0.0 0.0 0.0 0.0 0.0 0.0
168.51851851851842 674.0740740740739 168.51851851851842 0.0 674.0740740740739 2696.2962962962956 674.0740740740739 0.0 168.51851851851842 674.0740740740739 1
0.0 168.51851851851842 674.0740740740739 168.51851851851842 0.0 674.0740740740739 2696.2962962962956 674.0740740740739 0.0 168.51851851851842 674.07407407407
0.0 0.0 168.51851851851842 337.03703703703695 0.0 0.0 674.0740740740739 1348.148148148148 0.0 0.0 168.51851851851842 337.03703703703695 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 337.0370370370369 168.51851851851842 0.0 0.0 1348.1481481481478 674.0740740740739 0.0 0.0 337.0370370370369 168.51851851851842 0.0 0.0
0.0 0.0 0.0 0.0 168.51851851851842 674.0740740740739 168.51851851851842 0.0 674.0740740740739 2696.2962962962956 674.0740740740739 0.0 168.51851851851842 674.
0.0 0.0 0.0 0.0 0.0 168.51851851851842 674.0740740740739 168.51851851851842 0.0 674.0740740740739 2696.2962962962956 674.0740740740739 0.0 168.51851851851842
0.0 0.0 0.0 0.0 0.0 0.0 168.51851851851842 337.03703703703695 0.0 0.0 674.0740740740739 1348.148148148148 0.0 0.0 168.51851851851842 337.03703703703695
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 337.0370370370369 168.51851851851842 0.0 0.0 674.0740740740739 337.03703703703695 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 168.51851851851842 674.0740740740739 168.51851851851842 0.0 337.03703703703695 1348.1481481481478 337.03703703703695 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 168.51851851851842 674.0740740740739 168.51851851851842 0.0 337.03703703703695 1348.1481481481478 337.03703703703695
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 168.51851851851842 337.03703703703695 0.0 0.0 337.03703703703695 674.074074074074
```

Global F

```
12000.0 12000.0 12000.0 12000.0 12000.0 0.0 0.0 12000.0 12000.0 0.0 0.0 12000.0 12000.0 12000.0 12000.0 12000.0
```

A

```
36.81481481481481 4.240740740740733 0.0 0.0 4.240740740740738 -4.962962962962964 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
4.240740740740733 66.96296296296296 4.240740740740733 0.0 -4.962962962962964 5.148148148148145 -4.962962962962964 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 4.240740740740733 66.96296296296296 4.240740740740733 0.0 -4.962962962962964 5.148148148148145 -4.962962962962964 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 4.240740740740733 36.81481481481824 0.0 0.0 -4.962962962962964 4.24074074074074 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
4.240740740740738 -4.962962962962964 0.0 0.0 66.96296296296296 5.148148148148135 0.0 0.0 4.240740740740738 -4.962962962962964 0.0 0.0 0.0 0.0 0.0 0.0 0.0
-4.962962962962964 5.148148148148145 -4.962962962962964 0.0 5.148148148148135 120.5925925925926 5.148148148148135 0.0 -4.962962962962964 5.148148148148145 -4.9
0.0 -4.962962962962964 5.148148148148145 -4.962962962962964 0.0 5.148148148148135 120.5925925925926 5.148148148148135 0.0 -4.962962962962964 5.148148148148145
0.0 0.0 -4.962962962962964 4.24074074074074 0.0 0.0 5.148148148148135 66.96296296296296 0.0 0.0 -4.962962962962964 4.24074074074074 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 4.240740740740738 -4.962962962962964 0.0 0.0 66.96296296296296 5.148148148148135 0.0 0.0 4.240740740740738 -4.962962962962964 0.0 0.0
0.0 0.0 0.0 0.0 -4.962962962962964 5.148148148148145 -4.962962962962964 0.0 5.148148148148135 120.5925925925926 5.148148148148135 0.0 -4.962962962962964 5.1481
0.0 0.0 0.0 0.0 0.0 -4.962962962962964 5.148148148148145 -4.962962962962964 0.0 5.148148148148135 120.5925925925926 5.148148148148135 0.0 -4.962962962962964 5_
0.0 0.0 0.0 0.0 0.0 0.0 0.0 -4.962962962962964 4.24074074074074 0.0 0.0 5.148148148148135 66.96296296296296 0.0 0.0 -4.962962962962964 4.24074074074074
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 4.240740740740738 -4.962962962962964 0.0 0.0 36.81481481481481 4.240740740740733 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -4.962962962962964 5.148148148148145 -4.962962962962964 0.0 4.240740740740733 66.96296296296296 4.240740740740733 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -4.962962962962964 5.148148148148145 -4.962962962962964 0.0 4.240740740740733 66.96296296296296 4.240740740740733
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 -4.962962962962964 4.24074074074074 0.0 0.0 4.240740740740733 36.814814814814824
```

Test case 2

Test case 2d transient solution

- 100 – initial temperature
- 100 – simulation time [s],
- 1 – simulation step time [s],
- 1200 – ambient temperature [C],
- 300 – alfa [W/m²K],
- 0.100 – H [m],
- 0.100 – B [m],
- 31 – N_H,
- 31 – N_B,
- 700 – specific heat [J/(kg°C)],
- 25 – conductivity [W/(m°C)],
- 7800 – density [kg/m3].

MAKSYMALNE I MINIMALNE TEMPERATURY W KOLEJNYCH KROKACH CZASOWYCH

Time[s]	MinTemp[s]	MaxTemp[s]
1.0	99.99707497642771	149.5487448204024
2.0	99.99415003888402	177.40675860677055
3.0	99.99122519758765	197.14595901473643
4.0	99.98830052875849	212.87051183777194
5.0	99.98537638098114	226.1474725311807
6.0	99.98245393584969	237.72691919643816
7.0	99.97953643503871	248.0360437280681
8.0	99.97663143816064	257.3483358050689
9.0	99.97375440866173	265.8528168271385
10.0	99.97093372075021	273.687097492065
11.0	99.96821688700261	280.95516999343636
12.0	99.9656775161283	287.73787254093514
13.0	99.96342231389119	294.09946216797
14.0	99.96159738752253	300.09195400575953
15.0	99.96039320307605	305.75809964026314

WNIOSKI:
Rozbieżności minimalnej temperatury wynikają z przybliżeń jakie zakłada się w metodzie Jacobiego. W kolejnych iteracjach spełnia poprawność wyników, ponieważ metoda jest dokładniejsza. Program przeszedł testy. Udało się zrealizować zadanie.