Diego Vega Dechnik	Grupa 8	4.01.2022

Cel

Celem projektu jest opracowanie symulacji rozkładu temperatur na powierzchni dwuwymiarowej wykorzystując do tego równanie Fouriera dla nieustalonej wymiany ciepła.

Wprowadzenie

Metoda elementów skończonych to metoda rozwiązywania równań różniczkowych, spotykanych w fizyce i technice, jest ona najbardziej popularną metodą rozwiązywania problemów inżynierskich jak i modeli matematycznych.

Główna idea MES polega na dyskretyzacji układów ciągłych. Oznacza to, że dowolną ciągłą wartość (np. temperaturę) można zamienić na model oparty na ograniczonej ilości węzłów, które z kolei tworzą ograniczoną ilość elementów skończonych.

Kroki wykonane w celu obliczenia założonego rozkładu temperatur

Generacja siatki

Siatka składa się z elementów skończonych, takich jak prostokąty, trapezy czy trójkąty. W każdym swoim rogu element posiada węzeł, jeżeli elementy ze sobą sąsiadują część ich węzłów może być wspólna. Elementy mogą mieć także różne wymiary.

W tym przypadku elementy są prostokątami, a także, każdy element ma takie same wymiary.

Obliczenie Funkcje pochodnych funkcji kształtu elementu czterowęzłowego

Funkcje kształtu służą do interpolowania wartości obliczanej temperatury w dowolnym elemencie siatki. Dla elementów o kształcie prostokąta funkcje kształtu w układzie lokalnym i globalnym są takie same.

$$N_1 = 0.25(1-\xi)(1-\eta)$$

$$N_2 = 0.25(1+\xi)(1-\eta)$$

$$N_3 = 0.25(1+\xi)(1+\eta)$$

$$N_4 = 0.25(1-\xi)(1+\eta)$$

Pochodne oblicza się po współrzędnych punktów całkowania, czyli węzłów Gaussa-Legendre'a dla N=1 lub N=2, czyli odpowiednio dla dwu- lub trzy-punktowego schematu całkowania.

N	k	Węzły x_k	Współczynniki A_k
1	0; 1	$\mp 1/\sqrt{3}$	1
2	0; 2	$\mp\sqrt{3/5}$	5/9
	1	0	8/9

Obliczenie Jakobianu przekształcenia oraz odwrotnej macierzy Jakobiego

Jakobian przekształcenia to wyznacznik macierzy Jakobiego czyli macierzy, która określa macierz przekształceń z układu globalnego na układ lokalny.

Wielkość macierzy Jakobiego równa jest wymiarowi siatki do kwadratu (np. dla siatki 2D - 2*2, a dla siatki 3D - 3*3).

Każdy wiersz macierzy zawiera kolumny, które określają wartości przekształceń w danych wymiarach oraz kąty przekształceń dla tych wymiarów.

Macierz jakobiego oblicza się poprzez mnożenie pochodnych funkcji kształtu i współrzędnych węzłów.

Odwrotną macierz Jakobiego oblicza się mnożąc macierz Jakobiego i odwrotność Jakobianu.

Macierz Jakobiego, Jakobian oraz odwrotną macierz Jakobiego dla wszystkich punktów całkowania wszystkich elementów siatki.

Obliczenie lokalnej macierzy H

Macierz H elementu siatki otrzymuje poprzez zsumowanie macierzy H obliczonych dla każdego punktu całkowania ze wzoru:

$$[H] = \int_V k(t)\left(\left\{\frac{\partial\{N\}}{\partial x}\right\}\left\{\frac{\partial\{N\}}{\partial x}\right\}^T + \left\{\frac{\partial\{N\}}{\partial y}\right\}\left\{\frac{\partial\{N\}}{\partial y}\right\}^T\right)dV$$

- k - stała określająca przewodność elementu
- dV – jakobian obliczony dla elementu

Lokalną macierz H oblicza się dla każdego elementu siatki

Obliczenie lokalnej macierzy H_{BC}

Macierz H_{BC} określa ilość ciepła przyjmowanego z zewnątrz poprzez konwekcję.

Macierz H_{BC} elementu otrzymuje się poprzez zsumowanie macierzy H_{BC} obliczonych dla każdego punktu całkowania na ścianach elementu ze wzoru:

$$[H_{BC}] = \int_S \alpha(\{N\}\{N\}^T)\,dS \qquad \sum_{i=1}^{n_{pc}} f(pc_i)\,w_i\,\det[J]$$

- alfa - współczynnik wnikania ciepła elementu
- N – wektor wartości funkcji kształtu danego punktu całkowania
- w – waga danego punktu całkowania
- det[j] - wielkość ogrzewanej powierzchni, w tym wypadku jest to połowa długości ściany elementu

Następnie obliczoną macierz mnoży się razy wartość warunku brzegowego danej ściany.

Macierz H_{BC} oblicza się dla każdego elementu znajdującego się na ścianie siatki, macierze H_{BC} innych elementów przyjmują wartość równą 0.

Agregacja macierzy $H+H_{BC}$

Macierz zagregowana elementu ma rozmiar równy ilości węzłów siatki do kwadratu.

Macierz tą otrzymuje się poprzez wstawianie wszystkich wartości ze zsumowanych macierzy H i H_{BC} w odpowiednie miejsca w nowej macierzy. Reszta wartości macierzy wynosi 0.

Agreguje się macierz $H+H_{BC}$ dla każdego elementu, następnie wszystkie macierze są sumowane.

Obliczenie lokalnego wektora P

Wektor P określa strumień ciepła, oblicza się go ze wzoru:

$$[P] = \int_S \alpha\{N\}t_{ot}\,dS = \sum_{i=1}^{n_{pc}} f(pc_i)\,w_i\,\det[J]$$

- alfa - współczynnik wnikania ciepła elementu
- N – wektor wartości funkcji kształtu danego punktu całkowania
- t_{ot} – temperatura otoczenia
- w – waga danego punktu całkowania
- det[j] - wielkość ogrzewanej powierzchni, w tym wypadku jest to połowa długości ściany elementu

Wektor P oblicza się dla każdego elementu na danej ścianie siatki, wektory P innych elementów przyjmują wartość równą 0.

Agregacja wektora P

Wektor zagregowany ma rozmiar równy ilości węzłów w siatce.

Wektor ten otrzymuje się poprzez wstawienie wszystkich wartości lokalnego Wektora P w odpowiednie miejsca nowego wektora.

Agreguje się wektor P dla każdego elementu, następnie wszystkie wektory są sumowane.

Obliczenie macierz lokalnej C

Macierz C określa sposób w jaki ciepło rozchodzi się wewnątrz siatki.

Macierz C elementu siatki otrzymuje poprzez zsumowanie macierzy C obliczonych dla każdego punktu całkowania ze wzoru:

$$[C] = \int_V \rho c_p(\{N\}\{N\}^T)\,dV$$

- ro - gęstość
- c_p - przewodność cieplna
- N - wektor wartości funkcji kształtu danego punktu całkowania
- dV – jakobian obliczony dla elementu

Macierz C oblicza się dla wszystkich elementów siatki.

Agregacja macierzy C

Agregacja macierzy C działa na tej samej zasadzie co agregacja macierzy $H+H_{BC}$.

Macierz C agreguje się również dla wszystkich elementów, następnie macierze te są sumowane.

Obliczenie globalnej macierzy H

Globalną macierz H otrzymuje się ze wzoru:

$$[H] = [H]+[C]/dT$$

- H(niepogrubione) – Macierz otrzymana po agregacji macierzy H
- C – macierz otrzymana po agregacji macierzy C
- dT – czas trwania kroku symulacji

Obliczenie globalnego wektora P

Globalny wektor P otrzymuje się ze wzoru:

$$\{P\} = \{P\}+\{[C]/dT\}*\{T0\}$$

- P(niepogrubione) - Wektor otrzymany po agregacji wektorów P
- C – macierz otrzymana po agregacji macierzy C
- dt – czas trwania kroku symulacji
- T0 – wektor temperatur początkowych

Globalny wektor P jest obliczany dla każdego kroku symulacji, przy czym w kolejnych obliczeniach w miejsce wektora T0 wstawia się wektor temperatur obliczony w poprzednim kroku.

Rozwiązanie układu równań

Aby otrzymać wektor temperatur układu dla danego kroku czasowego symulacji należy rozwiązać układ równań:

$$\left([H]+\frac{[C]}{\Delta\tau}\right)\{t_1\}-\left(\frac{[C]}{\Delta\tau}\right)\{t_0\}+\{P\}=0\,,$$

Można to zrobić np. za pomocą metody Gaussa lub metody Jacobiego. W tym wypadku zastosowano metodę Jacobiego.

Wektor temperatur oblicza się dla każdego kroku czasowego, a obliczony wektor wykorzystuje się jako wektora temperatur początkowych dla następnego kroku.

Program

W skrócie:

Program na początku tworzy siatkę i element uniwersalny na podstawie podanych danych.

Następnie obliczane są macierze pochodnych

W kolejnym kroku obliczane są jakobiany przekształcenia dla każdego elementu

Przy okazji obliczania jakobianów obliczane są również

- Odwrotna macierz Jakobiego
- Macierze lokalne H
- Macierze lokalne C

Następnym krokiem jest obliczenie wszystkich macierzy lokalnych H_{BC} oraz zagregowanie sum macierzy $H+H_{BC}$

Po tym obliczane są wszystkie lokalne wektory P. są one następnie agregowane

Kolejnym krokiem jest zagregowanie wszystkich macierzy C

Gdy wszystkie agregacje są skończone obliczana jest globalna macierz H, a następnie w pętli wektor globalny P i wektor temperatur.

Budowa programu:

Program składa się z 4 głównych klas:

- node
    - int id – przechowuje id węzła
    - int bc – przechowuje wartość warunku brzegowego węzła
    - double x, y - przechowują współrzędne węzła
    - node() - konstruktor
    - void show() - wyświetla informacje o węźle

- element
    - int value – przechowuje id elementu
    - nn – przechowuje całkowitą liczbę węzłów w siatce
    - node *nodes – przechowuje węzły elementu
    - double **H – przechowuje macierz lokalną H
    - double **Hbc – przechowuje macierz lokalną H_{BC}
    - double **C – przechowuje macierz lokalną C
    - double *P – przechowuje wektor lokalny P
    - element() - konstruktor
    - ~element() - destruktor
    - void show() - Wyświetla informacje o elemencie
    - void showMore() - wyświetla więcej informacji o obiekcie
    - void printH() - wyświetla macierz lokalną H

- void printHbc() - wyświetla macierz lokalną H_{BC}
- void addMatrixes() - Dodaje macierz H do macierzy H_{BC}
- void aggregate() - agreguje sumę macierzy H i H_{BC}
- void CalculateP() - Oblicza wektor lokalny P
- double getLength() - zwraca odległość między dwoma węzłami
- void aggregateVector() - agreguje wektor lokalny P
- void showAggregationVector() - wyświetla zagregowany wektor lokalny
- void aggregateC() - agreguje macierz lokalną C
- void calculateC() - oblicza macierz lokalną C

- grid
  - double initialTemp – przechowuje temperaturę początkową
  - double simulationStepTime – przechowuje długość kroku symulacji
  - double ambientTemp – przechowuje temperaturę otoczenia
  - double alfa – przechowuje współczynnik wnikania ciepła
  - double specificHeat - ciepło właściwe
  - double conductivity – przechowuje przewodność cieplną
  - double density – przechowuje gęstość
  - double h, b; int nh, nb – przechowują dane o wymiarach siatki
  - double nn – przechowuje ilość węzłów w siatce
  - double ne - przechowuje ilość elementów w siatce
  - double pc – schemat całkowania (2 lub 3)
  - node *nodes – przechowuje węzły siatki
  - element *elements – przechowuje elementy siatki
  - double **nc – przechowuje wartości funkcji kształtu dla punktów całkowania
  - double **aggregationArr – przechowuje sumę zagregowanych macierzy lokalnych H
  - double *aggregationVector - przechowuje sumę zagregowanych wektorów lokalnych P
  - double *finalP – Przechowuje Wektor Globalny P
  - double **aggregationArrC - przechowuje sumę zagregowanych macierzy lokalnych C
  - double *T0 – przechowuje wektor temperatur początkowych
  - grid() - konstruktor
  - ~grid() - destruktor
  - void show() - wyświetla dane o siatce
  - void showBcGrid() - wyświetla warunki brzegowe siatki
  - void calculateJakobian() - oblicza macierz jakobiego dla danego punktu całkowania danego elementu
  - void calculateJakobians(double **dEta, double** dKsi) - oblicza Jakobian dla każdego punktu całkowania każdego elementu, wywołuje również obliczanie macierzy lokalnych H i C
  - void inverseJakobian(double **J, double **JIn) - oblicza odwórconą macierz jakobiego
  - void calculateDxDy(double **dx, double **dy, double **poEta, double** poKsi, double **JIn, int i) -część obliczania macierzy lokalnych H
  - void calculateH(double **h, double **dx, double **dy, double **J,int a,double w) - oblicza macierz lokalną H danego elementu
  - void multiplyH(double **h, double **J, double t) - mnoży macierz loalną H danego elementu razy konkretne współczynniki
  - void aggregateAll() - agreguje wszystkie macierze lokalne H
  - void addToAggregationArr(double **arr) - dodaje macierz do macierzy **aggregationArr

- void printAggregationArr() - wyświetla sumę zagregowanych macierzy lokalnych H
- void printAggregationVector() - wyświetla sumę zagregowanych wektorów lokalnych P
- void addToAggregationVector(double *vec) dodaje wektor do wektora *aggregationVector
- void addToAggregationArrC(double **arr) - dodaje macierz do macierzy **aggregationArrC
- void printAggregationArrC() - wyświetla sumę zagregowanych macierzy loklanych C
- void addCToH() - oblicza końcową macierz globalną H
- void pPlusCTimesT0() - oblicza końcowy wektor globalny P
- void aggregateAllC() - agreguje wszystkie macierze lokalne C
- void printFinalP() - wyświetla wynik działania programu

- element2d4pc
  - double **dEta, **dKsi – przechowują dane potrzebne do obliczenia macierzy Jakobiego
  - double **nc – przechowuje dane potrzebne do obliczenia macierzy C
  - wall lWall, bWall, rWall, tWall - przechowują ściany siatki
  - grid *g – przechowuje siatkę
  - element2d4pc() - konstruktor
  - ~element2d4pc() - destruktor
  - double fun(), double fun1(), double fun3() - funkcje służące do obliczania wartości funkcji kształtu
  - void calculateDEtaAndDKsi() - oblicza zawartość tablic dEta i dKsi
  - void printDEtaAndDKsi() - wyświetla awartość tablic dEta i dKsi
  - void fillWalls(element *e) - uzupełnia zmienne typu wall odpowiednimi elementami
  - void calculateHbc() - oblicza macierze lokalne H_{BC}
  - void calculateAllHbc() - oblicza wszystki macierze lokalne H_{BC}
  - void aggregateAllVectors() - agreguje wszystkie wektory lokalne P
  - void calculateAllVectorsP() - oblicza wszystkie wektory lokalne P
  - void fillN() - uzupełnia zawartość tablicy nc
  - void s() - wywołuje rozwiązanie układu równań, wyświetla wyniki i szykuje program do kolejnej iteracji (zmienia zawartość wektora temperatur itd.)
  - double* equationSolve() - rozwiązuje układ równań

- wall
  - element *w – przechowuje elementy
  - double* n1, n2, n3 - przechowują obliczone wartości funkcji kształtu
  - int size – przechowuje ilość elementów
  - wall(int size, double ksi1, double eta1, double ksi2, double eta2) - konstruktor dla schematu całkowania dwupunktowego
  - wall(int size, double ksi1, double eta1, double ksi2, double eta2, double ksi3, double eta3) konstruktor dla schematu trójpunktowego

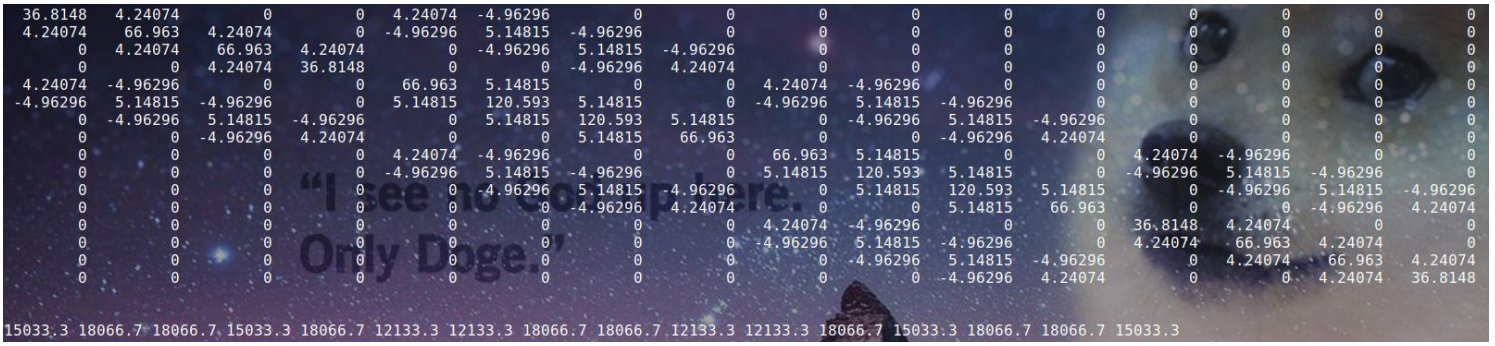
Sprawdzenie poprawności działania programu

Test 1

Wartości początkowe:

- 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].

Macierz Globalna H i wektor Globalny P po pierwszej iteracji



Obliczone wartości

iteracja	Min [°C]	Max [°C]
0	110.038	365.815
1	168.837	502.592
2	242.801	587.373
3	318.615	649.387
4	391.256	700.068
5	459.037	744.063
6	521.586	783.383
7	579.034	818.992
8	631.689	851.431
9	679.908	881.058

Test 2

Wartości początkowe:

- 100 – initial temperature
- 100 – simulation time [s],
- 1 – simulation step time [s],
- 1200 – ambient temperature [C],
- 300 – alfa [W/m^{2}K],
- 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].

Obliczone wartości

iteracja	min [°C]	max [°C]
0	100	149.557
1	100	177.445
2	100	197.267
3	100	213.153
4	100	226.683
5	100	238.607
6	100	249.347
7	100	259.165
8	100	268.241
9	100	276.701
10	100.001	284.641
11	100.002	292.134
12	100.003	299.237
13	100.005	305.997
14	100.009	312.451
15	100.014	318.631
16	100.021	324.564
17	100.032	330.271
18	100.046	335.772
19	100.064	341.085

Wnioski

Wyniki dla testu drugiego nieznacznie różnią się od wyników przekazanych przez prowadzącego, Może być to wina różnicy kompilatorów programu. Nie zmienia to jednak faktu, że są one bardzo do siebie zbliżone. Zwiększenie rozmiaru siatki z 4 na 4 elementów, do 31 na 31 elementów znacznie zwiększyło czas wykonywania obliczeń.