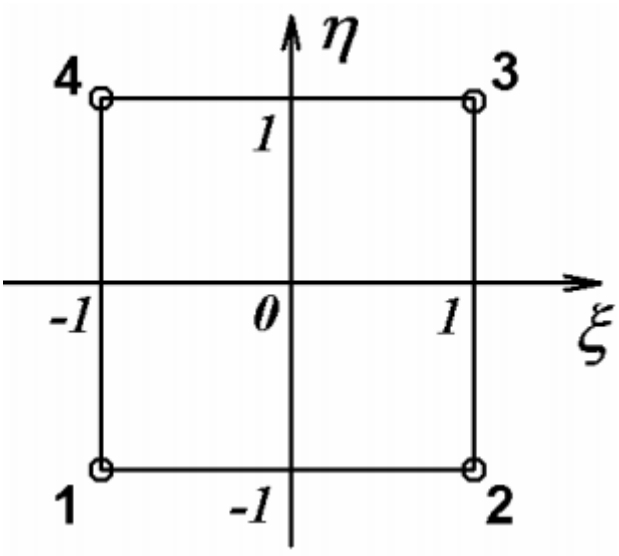
Jakub Kopeć, IS, GL03

1. Wstęp teoretyczny:

Celem działania programu jest stworzenie symulacji niestacjonarnej wymiany ciepła prostokątnego dwuwymiarowego fragmentu metalu używając metody elementów skończonych. Mes polega na dekompozycji badanego obszaru na mniejsze elementy, których kształt jest ograniczony połączeniami węzłów siatki. Elementy mogą przybierać kształt dowolnej figury, mają wspólne węzły i wspólnie aproksymują kształt ośrodka. Celem stosowania tej techniki jest wyznaczenie rozkładu temperatur w węzłach siatki w poszczególnych momentach procesu wymiany ciepła. Układ globalny (przedstawiający rzeczywiste wymiary elementu) warto jest przekształcić w taki sposób, aby pojedynczy element został odwzorowany przez kwadrat o wymiarach 2x2 w układzie funkcji kształtu ksi-eta na przedziałach <-1;1>:



Wzory funkcji kształtów:

$$N_1 = \frac{1}{4}(1-\xi)(1-\eta);$$

$$N_2 = \frac{1}{4}(1+\xi)(1-\eta);$$

$$N_3 = \frac{1}{4}(1+\xi)(1+\eta);$$

$$N_4 = \frac{1}{4}(1-\xi)(1+\eta).$$

gdzie interpolacja współrzędnych określona jest równaniami:

$$x = \sum_{i=1}^{N_{nodes}} N_i x_i = N_1 x_1 + N_2 x_2 + N_3 x_3 + N_4 x_4;$$

$$y = \sum_{i=1}^{N_{nodes}} N_i y_i = N_1 y_1 + N_2 y_2 + N_3 y_3 + N_4 y_4.$$

Związek między pochodnymi funkcji kształtu względem ξ i η oraz pochodnymi funkcji kształtu względem x i y zapisane są równaniami:

$$\begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{Bmatrix} = [J] \begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix},$$

gdzie macierz J nazywana jest macierzą Jacobiego, w skład której wchodzą:

$$J = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix}.$$

Wyznacznik tej macierzy jest Jakobianem transformacji układu współrzędnych. Za pomocą tej macierzy możliwe jest wyznaczenie pochodnych funkcji kształtu po współrzędnych x i y (współrzędnych lokalnych)

$$\begin{Bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{Bmatrix} = J^{-1} \begin{Bmatrix} \frac{\partial N_i}{\partial \xi} \\ \frac{\partial N_i}{\partial \eta} \end{Bmatrix},$$

Pochodne funkcji kształtu względem x oraz y uzyskuje się z powyższego wzoru macierzowego poprzez odwrócenie macierzy Jacobiego. Uzyskując macierz Jacobiego oraz pochodne funkcji kształtu po x oraz y, można przejść do całkowania numerycznego metodą kwadratur Gaussa. Metoda ta ma postać:

$$\iint_S f(x,y)dxdy = \int_{-1}^{1}\int_{-1}^{1} \varphi(\xi,\eta)\det J\, d\xi d\eta = \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \varphi(\xi_i,\eta_j)\det J(\xi_i,\eta_j).$$

gdzie konkretne punkty ξ, η, w_i,w_j są ściśle zależne od tego, jaki schemat całkowania wybrano.

Mając wszystkie powyższe dane dla elementu skończonego oraz właściwości materiału badanej powierzchni, można obliczyć lokalną macierz sztywności H, lokalną macierz obciążeń C, lokalny wektor obciążeń P oraz macierz H_{bc} ze wzorów:

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

$$[C] = \int_V c\rho\{N\}\{N\}^T dV$$

$$\{P\} = -\int_S \alpha\{N\}t_\infty dS$$

$$[H_{BC}] = \int_S \alpha(\{N\}\{N\}^T)\, dS$$

Aby można było wykorzystać macierze lokalne w znalezieniu wektora temperatur węzłowych po czasie, należy zagregować do macierzy globalnych. Poszukiwany wektor wyznaczany jest ze wzoru:

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

Gdzie:

H- macierz sztywności H, uwzględnia także macierz warunku brzegowego Hbc
C-globalna macierz obciążeń, również powstała przez agregację jak w przypadku macierzy H
t0- wektor temperatur początkowych w węzłach siatki
P-wektor obciążeń z warunkami brzegowymi
t-wektor temperatur końcowych w danej chwili (wektor niewiadomych)
Δτ -długość kroku czasowego
Powyższy wzór został przekształcony do postaci:
[H^]^(-1)*{t}={P^}
Gdzie:
[H^] = [H]+[C]/ Δτ,
[P^] = {P} + ([C]// Δτ){t0}

W wyniku rozwiązania układu równań wyznaczany jest wektor {t}, który następnie jest przypisywany wektorowi {t0}, tworząc jednocześnie nowy układ równań w kolejnej chwili czasowej.

2. Charakterystyka projektu:

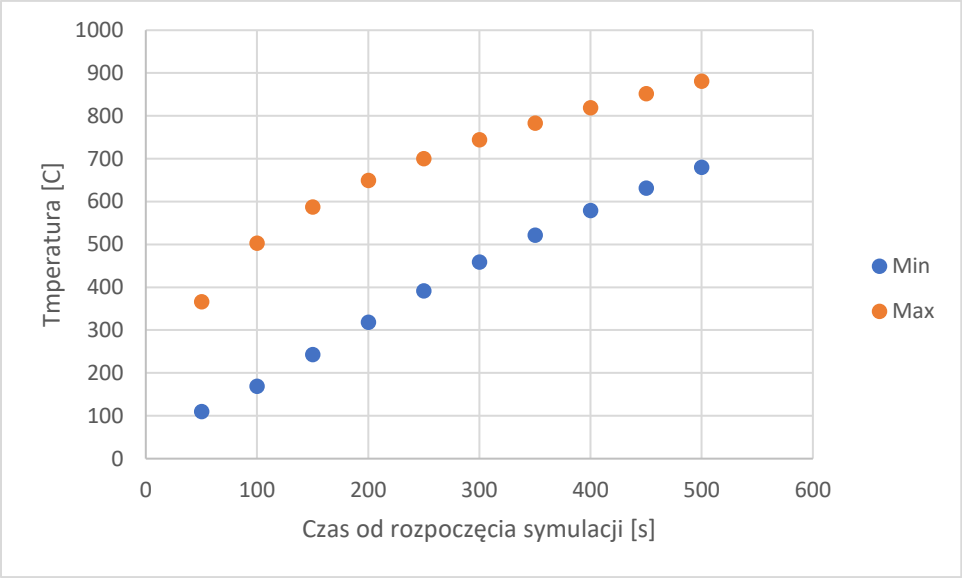
Działanie programu rozpoczyna obiekt klasy GlobalData, który odczytuje dane z pliku tekstowego niezbędne do utworzenia siatki. Obiekt ten pobiera kolejno: wysokość oraz szerokość siatki (zakładamy, że siatka ma kształt prostokąta), ilość węzłów na jej bokach, współczynnik przewodzenia ciepła, schemat całkowania wykorzystywany przy całkowaniu metodą kwadratur Gaussa (obsługiwany jest 2-punktowy, 3-punktowy oraz 4-punktowy schemat całkowania), gęstość materiału, ciepło właściwe materiału, temperaturę początkową węzłów siatki, temperaturę otoczenia, współczynnik konwekcji, długość symulacji oraz długość kroku czasowego.
Na podstawie danych z pliku tworzone są obiekty Node, które wchodzą w skład siatki, by następnie przypisać im odpowiednie identyfikatory, współrzędne, temperaturę początkową oraz zmienną określającą występowanie warunku brzegowego na krawędziach każdego z elementów. Potem węzły o odpowiednich identyfikatorach przypisywane są do elementów. Węzły elementu znajdują się na przecięciach jego krawędzi. Wszystkie elementy posiadają oddzielne macierze H, C oraz P. Po stworzeniu struktury siatki wywoływana jest funkcja simulation, która oblicza skok czasowy jednej iteracji symulacji. Każda iteracja pętli symulacji zawiera

swoją pętlę wywołań funkcji elemSolve, która oblicza oraz zwraca lokalne macierze niezbędne do znalezienia wektora rozwiązań układu równań dla jednego z elementów siatki, które później za pomocą algorytmu agregacji współtworzą globalne macierze H, C, P widoczne w macierzowej formie równania we wstępie teoretycznym. Agregacja macierzy lokalnych ma miejsce w funkcji simulation, lecz całą logiką procesu zajmują się funkcje sumUpGlobal (jedna dla macierzy H i C, druga dla wektora P). Bazując na operacjach na macierzach oraz na algorytmie odwracania macierzy Gaussa-Jordana, rozwiązywany jest układ równań, z którego wybierana jest minimalna oraz maksymalna wartość temperatury i porównywana z przewidywanymi wartościami. Zbiór rozwiązań jest następnie wykorzystywany w nadpisaniu poprzednich temperatur w węzłach i poszukiwany jest nowy wektor rozwiązań. Czynność powtarza się aż do wykonania symulacji dla ostatniego kroku czasowego.

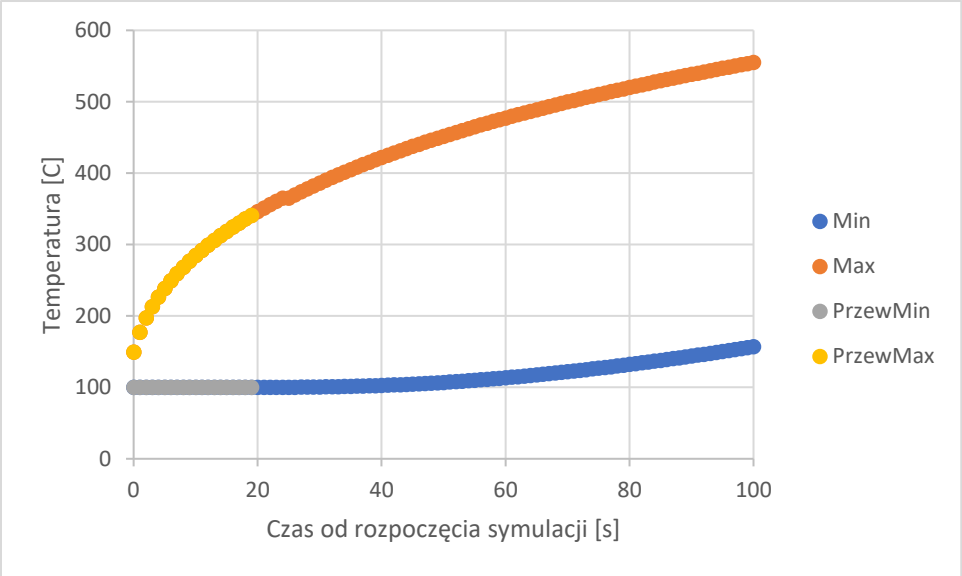
3. Opis struktury plików:
GlobalData.cpp - plik przechowuje klasy GlobalData oraz FEMGrid które są tworzą siatkę i macierze globalne niezbędne do rozwiązania układu równań,
Data.txt – dane wejściowe pobierane przez obiekt GlobalData,
Node.cpp – plik zawiera klasę, której obiekty reprezentują węzły siatki,
Element.cpp - – plik zawiera klasę, której obiekty reprezentują elementy siatki,
Elem2Solve.cpp – plik zawiera funkcję zwracającą lokalne macierze pojedynczego elementu dla dwupunktowego schematu całkowania,
ElemSolve.cpp - plik zawiera funkcję zwracającą lokalne macierze pojedynczego elementu dla trójpunktowego oraz czteropunktowego schematu całkowania,
Functions.cpp – plik zawiera funkcję agregacji, odwracania macierzy metodą Gaussa-Jordana, obliczania wartości funkcji kształtu w punkcie i kilka innych mniej znaczących, lecz jednocześnie wykorzystywanych funkcji.

4. Porównanie wyników programu z przewidywanymi wartościami:
Z powodu braku różnic pomiędzy wynikami programu dla różnych schematów całkowania, zdecydowałem się nie umieszczać danych dla każdego ze schematów.

	Test nr 1		Wartości oczekiwane			
Czas	MinTemp	MaxTemp	MinTemp	MaxTemp	Różnica	
50	110,038	365,818	110,038	365,815	0	0,003
100	168,837	502,596	168,837	502,592	0	0,004
150	242,801	587,38	242,801	587,373	0	0,007
200	318,615	649,397	318,615	649,387	0	0,01
250	391,256	700,081	391,256	700,068	0	0,013
300	459,037	744,078	459,037	744,063	0	0,015
350	521,587	783,4	521,586	783,383	0,001	0,017
400	579,035	819,012	579,034	818,992	0,001	0,02
450	631,69	851,453	631,689	851,431	0,001	0,022
500	679,908	881,081	679,908	881,058	0	0,023



	Test nr 2		Wartości oczekiwane			
Czas	MinTemp	MaxTemp	MinTemp	MaxTemp	Różnica	
1	100	149,557	100	149,56	0	-0,003
2	100	177,445	100	177,44	0	0,005
3	99,999	197,267	100	197,27	-0,001	-0,003
4	99,999	213,153	100	213,15	-0,001	0,003
5	99,999	226,683	100	226,68	-0,001	0,003
6	99,999	238,607	100	238,61	-0,001	-0,003
7	99,999	249,347	100	249,35	-0,001	-0,003
8	99,999	259,166	100	259,17	-0,001	-0,004
9	100	268,241	100	268,24	0	0,001
10	100	276,702	100	276,7	0	0,002
11	100,001	284,642	100	284,64	0,001	0,002
12	100,001	292,135	100	292,13	0,001	0,005
13	100,003	299,238	100	299,24	0,003	-0,002
14	100,005	305,998	100,01	306	-0,005	-0,002
15	100,008	312,452	100,01	312,45	-0,002	0,002
16	100,013	318,632	100,01	318,63	0,003	0,002
17	100,021	324,565	100,02	324,56	0,001	0,005
18	100,031	330,272	100,03	330,27	0,001	0,002
19	100,045	335,774	100,05	335,77	-0,005	0,004
20	100,064	341,086	100,06	341,08	0,004	0,006



5. Informacje końcowe:

W celu znacznie szybszego wykonania programu dla drugiej symulacji, zalecane jest umieszczenie fragmentu kodu liczącego macierze H, C, P globalne poza pętlą symulacji. Jednakże w przypadku dodania funkcjonalności występowania rozszerzalności cieplej, taka operacja będzie skutkować błędnymi wynikami. Algorytm eliminacji metodą Gaussa-Jordana został pobrany z repozytorium znajdującym się pod adresem: https://github.com/peterabraham/Gauss-Jordan-Elimination/blob/master/GaussJordanElimination.cpp , a następnie dostosowany pod strukturę programu.