AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INŻYNIERII METALI I INFORMATYKI PRZEMYSŁOWEJ

Metoda Elementów Skończonych.

Opracował:

Kacper Suder

Imię i nazwisko

Grupa numer 6	10.01.22
Grupa ćwiczeniowa	Data

408408

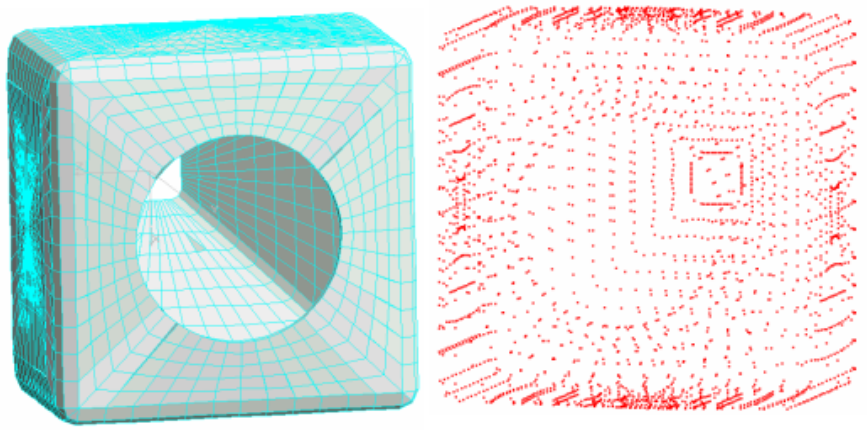
Numer Indeksu

1. Cel ćwiczeń laboratoryjnych:

W ramach ćwiczeń laboratoryjnych zostało opracowane oprogramowanie metody elementów skończonych, służące do symulacji procesu nieustalonego transportu ciepła z warunkiem brzegowym konwekcji. Dodatkowo opracowano całkowanie numeryczne przy pomocy kwadratury Gaussa-Legendre'a.

2. Wstęp teoretyczny:

Metoda Elementów Skończonych - Metoda rozwiązywania równań różniczkowych, wykorzystywana między innymi w przemyśle, do przeprowadzania symulacji (np. naprężeń, odkształceń). Aby móc przeprowadzić symulację na badanej bryle (np. blaszce), nakładana zostaje na nią **siatka MES.**

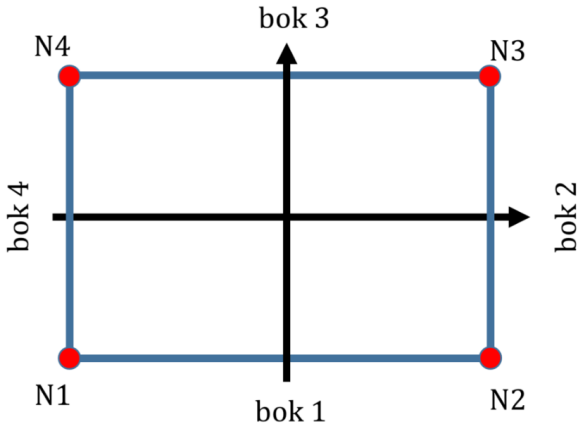


/Model zbiornika mieszarki, przygotowany do rozwiązania metodą elementów skończonych. Po lewej – konstrukcja podzielona na elementy, po prawej widoczne są węzły, w których elementy łączą się między sobą - źródło: Wiesław Śródka - "Trzy lekcje metody elementów skończonych. Materiały pomocnicze do przedmiotu wytrzymałość materiałów", OFICYNA WYDAWNICZA POLITECHNIKI WROCŁAWSKIEJ /

W przypadku utworzonego oprogramowania, liczony jest wyłącznie transport ciepła, z użyciem równania Fouriera-Kirchhoffa dla stanu nieustalonego z wykorzystaniem warunku brzegowego konwekcji (dla zmiennego strumienia ciepła).

Siatka MES - zbudowana jest ze skończonej ilości przestrzennych obiektów połączonych ze sobą. Najmniejszą częścią siatki jest element. Składa się on z węzłów oraz ścian. Podczas tworzenia siatki MES, różnica w numeracji węzłów na jednym elemencie powinna być jak najmniejsza (ze względu na późniejsze obliczenia).

Przykładowy element skończony w siatce MES:



Gdzie oznaczenia N1, N2, N3, N4 odpowiadają za węzły.

Konwekcja - proces przekazywania ciepła związany z makroskopowym ruchem materii w płynach (gazach i cieczach) lub plazmie. Siłą pędną wymiany ciepła konwekcji jest gradient temperatury.

Wzór: $q = \alpha * (t - t_{\infty})$

gdzie:

współczynnik α - zależy od materiału i jego powierzchni - określa jak "chętnie" ciepło jest oddawane lub przyjmowane przez dany obiekt;
t - temperatura badanego obiektu;
t_{∞} - temperatura otoczenia;

Interpolacja - Interpolacja w MES-ie wykorzystywana jest do wyliczenia między innymi temperatury, naprężeń w danym punkcie nie znajdującym się na węźle przy wykorzystaniu **funkcji kształtu**.

Wzór: $t_p = \{N\}^T * \{t\} = N_1 * t_1 + N_2 * t_2 + \ldots$

Funkcja kształtu - pozwala wyznaczyć szukane wielkości w punktach, nie znajdujących się na węzłach. Wzór funkcji kształtu oraz ilość funkcji zależy od liczby węzłów w elemencie oraz przestrzeni. Niezależnie jednak od wzoru i ilości, suma funkcji kształtu daje 1, natomiast w swoim węźle, wartość funkcja kształtu równa jest 1.

W napisanym oprogramowaniu skorzystano z następujących wzorów (funkcje kształtu liczone są dla elementu dwuwymiarowego o czterech bokach):

$$N1 = 0.25(1 - \xi)(1 - \eta)$$

$$N2 = 0.25(1 + \xi)(1 - \eta)$$

$$N3 = 0.25(1 + \xi)(1 + \eta)$$

$$N4 = 0.25(1 - \xi)(1 + \eta)$$

Kwadratury Gaussa-Legendre'a - kwadratury interpolacyjne, w których funkcja podcałkowa jest przybliżana wielomianem interpolacyjnym.

Symulacja:

Implementację oprogramowania rozpoczęto od zaimplementowania **równania Fouriera- Kirchhoffa** dla stanu ustalonego**:**

$$div(k(t)grad(t))=0,$$

Równanie to bez użycia operatorów różniczkowych ma postać:

$$\frac{\partial}{\partial x}\left(k_x(t)\frac{\partial t}{\partial x}\right)+\frac{\partial}{\partial y}\left(k_y(t)\frac{\partial t}{\partial y}\right)+\frac{\partial}{\partial z}\left(k_z(t)\frac{\partial t}{\partial z}\right)=0,$$

gdzie:
$k_x(t)$, $k_y(t)$, $k_z(t)$ - anizotropowe współczynnika przewodzenia ciepła zależne od temperatury t, w trzech kierunkach;

Aby poprawnie wprowadzić równanie do tworzonego programu i umożliwić przeprowadzenie symulacji, pierwszym etapem jest przekształcenie powyższego równania różniczkowego do postaci funkcjonału (równanie całkowe):

$$J=\int_V\frac{1}{2}\left(k_x(t)\left(\frac{\partial t}{\partial x}\right)^2+k_y(t)\left(\frac{\partial t}{\partial y}\right)^2+k_z(t)\left(\frac{\partial t}{\partial z}\right)^2\right)dV$$

Przyjmując, że dla materiałów izotropowych wartości anizotropowego przewodzenia ciepła są sobie równe, fragment funkcjonału który będzie rozwiązywany do przeprowadzenia symulacji ma postać:

$$J=\int_V\left(\frac{k(t)}{2}\left(\left(\frac{\partial t}{\partial x}\right)^2+\left(\frac{\partial t}{\partial y}\right)^2+\left(\frac{\partial t}{\partial z}\right)^2\right)dV\right.$$

Do pełnego przeprowadzenia obliczeń, należy dołączyć warunek brzegowy konwekcji. Ponieważ dyskretyzacja przedstawionego problemu, polega na podzieleniu rozpatrywanego obszaru na elementy i przedstawieniu temperatury wewnątrz elementu, jako funkcji wartości węzłowych zgodnie z zależnością:

$$t=\sum_{i=1}^{n}N_it_i=\{N\}^T\{t\}.$$

W ten sposób (wykorzystując interpolację) otrzymywany jest rozkład ciągły temperatur. Dzięki temu, po rozwiązaniu funkcjonału otrzymywany jest rozkład temperatury wewnątrz elementu, a w konsekwencji w całej siatce (a nie wyłącznie na węzłach). Finalne równanie ma postać:

$$J=\int_V\left(\frac{k}{2}\left(\left(\left\{\frac{\partial\{N\}}{\partial x}\right\}^T\{t\}\right)^2+\left(\left\{\frac{\partial\{N\}}{\partial y}\right\}^T\{t\}\right)^2+\left(\left\{\frac{\partial\{N\}}{\partial z}\right\}^T\{t\}\right)^2\right)\right)dV+\int_S\frac{\alpha}{2}\left(\{N\}^T\{t\}-t_\infty\right)^2dS$$

W celu otrzymania wyniku, należało zminimalizować funkcjonkcjonał, co sprowadza się do obliczenia pochodnych cząstkowych względem wartości węzłowych temperatury {t}. W konsekwencji otrzymano następujący układ równań:

$$\frac{\partial J}{\partial\{t\}}=\int_V\left(k\left(\left\{\frac{\partial\{N\}}{\partial x}\right\}\left\{\frac{\partial\{N\}}{\partial x}\right\}^T+\left\{\frac{\partial\{N\}}{\partial y}\right\}\left\{\frac{\partial\{N\}}{\partial y}\right\}^T+\left\{\frac{\partial\{N\}}{\partial z}\right\}\left\{\frac{\partial\{N\}}{\partial z}\right\}^T\right)\{t\}\right)dV+$$
$$+\int_S\alpha\left(\{N\}^T\{t\}-t_\infty\right)\{N\}dS=0$$

Otrzymany układ równań w postaci macierzowej ma postać:

$$[H]\{t\}+\{P\}=0$$

Jest to jednak niepełne równanie (wykorzystywane dla stanu ustalonego), które nie pozwoli na przeprowadzenie symulacji zmiany temperatury na węzłach w czasie. Jest ona przeprowadzana dla procesu niestacjonarnego (nieustalonego), stąd też równanie transportu ciepła (Fouriera - Kirchhoffa dla stanu nieustalonego) ma postać:

$$div(k(t)grad(t))=c\rho\frac{\partial t}{\partial\tau}$$

Po dokonaniu niezbędnych przekształceń wybrano odpowiedni schemat wyznaczania temperatury {t1}. Ponieważ schemat jawny ma ograniczone zastosowanie, ze względu na słabą stabilność rozwiązania dla różnych kroków czasowych ($\triangle\boldsymbol{\tau}$), zastosowano niejawny schemat wyznaczenia temperatury {t1}, który w postaci macierzowej ma postać:

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

gdzie:

1. **Macierz H** - Określa w jaki sposób transport ciepła zachodzi w materiale. W powyższym równaniu dodatkowo została do niej dodana macierz **HBC**.
   Wzór na macierz H:

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

   gdzie:
   k(t) - współczynnik przewodzenia ciepła

2. **Macierz HBC** - Fragment warunku brzegowego konwekcji, zawierający po zsumowaniu z macierzą H w finalnym równaniu nieznaną temperaturę ($\alpha * t$). Podział warunku brzegowego i zsumowanie tego fragmentu z macierzą **H** było niezbędne, ponieważ według definicji rozwiązywania układu równań nieznana (szukana) temperatura musi znaleźć się wraz z macierzą **H**.
   Wzór na macierz HBC:

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

3. **Macierz C** - macierz pojemności cieplnej materiału. Parametry fizyczne zawarte w macierzy **C** mówią o tym, jakim akumulatorem energii jest badany materiał.
   Wzór na macierz C:

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

   gdzie:

   c – ciepło właściwe
   ρ – gęstość materiału

4. **Wektor P** - Wektor obciążenia mówiący o tym, jakie obciążenia są wywierane przez temperaturę na wszystkie węzły. Wdraża warunek brzegowy dla wyrazów wolnych ze znaną temperaturą, co jest niezbędne ze względu na definicję rozwiązywania układu równań.
   Wzór na wektor P:

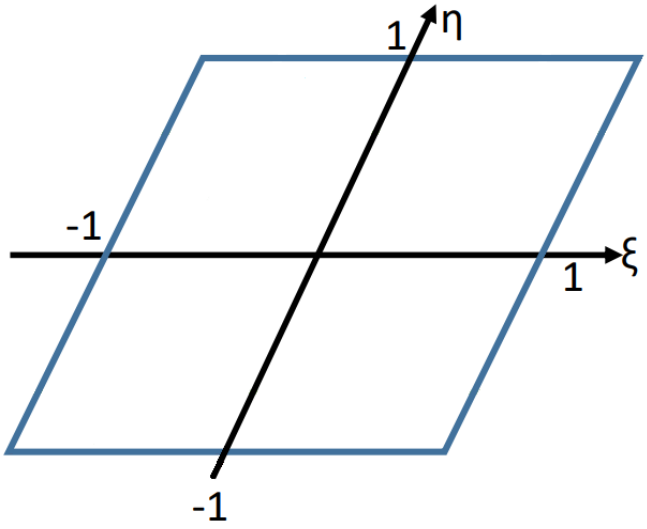
$$[P] = \int_S \alpha\{N\}t_{ot}\, dS$$

Ponadto, aby w pełni przeprowadzić całkowanie, niezbędne jest wyliczenie wyznacznika Jacobiego i wymnożenie poszczególnych macierzy (wraz z wektorem P) przez odpowiedni wyznacznik oraz wagi, które to zależą od wybranego schematu całkowania. Wagi te pobierane są z **tabeli kwadratur Gaussa-Legendre'a.**

Wyznacznik Jacobiego dla układu 1D stanowi stosunek długości elementu w układzie x-a, do długości elementu w układzie ksi.

W przypadku układu 2D wyznacznik Jacobiego jest stosunkiem pól układu globalnego do lokalnego.

W trakcie dokonywanych obliczeń wykorzystywany jest **element skończony w układzie lokalnym** mający następującą postać**:**



Do przekształcenia elementu z siatki w układzie globalnym, na element w układzie lokalnym (element znormalizowany) wykorzystywana jest macierz Jacobiego, a w odwrotną stronę - macierz odwrotna Jacobiego. Macierze te nazywane są Jacobianami przekształcenia. Takie przekształcenia stosuje się, ponieważ łatwiej jest dokonywać obliczeń w przedziale <-1, 1> dla elementu w kształcie kwadratu. Do tego celu wykorzystywane są punkty całkowania, których wartość pobierana jest z **tabeli kwadratur Gaussa-Legendre'a:**

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
3	0; 3	∓0.861136	0.347855
	1;2	∓0.339981	0.652145
4	0;4	∓0.906180	0.236927
	1;3	∓0.538469	0.478629
	2	0	0.568889

Macierz Jacobiego:

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

Macierz odwrotna Jacobiego:

$$[J]^{-1} = \frac{1}{\det[J]} \begin{bmatrix} \frac{\partial y}{\partial \eta} & -\frac{\partial y}{\partial \xi} \\ -\frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \xi} \end{bmatrix}$$

3. Realizacja laboratoriów:

3.1 Implementacja struktur oraz wczytanie z danych z pliku:

Podczas pierwszych zajęć zaimplementowano następujące struktury, pozwalające przechować niezbędne dane dotyczące badanej siatki, jak i właściwości materiału:

Node - struktura węzła:

```
//Struktura węzła:
struct NODE {
    double x = 0.0; //współrzędna x-owa
    double y = 0.0; //współrzędna y-owa
    double nodeTemp = 0.0; //temperatura węzła
    int BC = 0; //flaga wskazująca na obecność warunku brzegowego (0 oznacza brak warunku brzegowego).
};
```

/Wewnątrz zaimplementowanej struktury przechowywane są dane wyłącznie jednego węzła/

Element - struktura elementu:

```
struct ELEMENT {
    int ID[4] = { 0, 0, 0, 0 };
};
/*
* Wczytywanie ID węzłów:
*
*    ID[3]              ID[2]
*        _______________
*        |             |
*        |     <--_    |
*        |         \   |
*        |   \      /  |
*        |     -__-    |
*        |             |
*        |_____________|
*
*   ID[0]                 ID[1]
*/
```

/Struktura zawierająca ID dla danego elementu (gdyby były potrzebne różne ilości węzłów w elemencie, wówczas należy dodać kolejną strukturę przetrzymującą ID/

Grid - struktura siatki:

```
//struktura siatki
struct GRID {
    int nE = 0; //liczba elementów.
    int nN = 0; //liczba węzłów.
    NODE* ND = new NODE[nN]; //node Table.
    ELEMENT* EL = new ELEMENT[nE]; //Element table.
} struktura;
```

GlobalData - właściwości badanego materiału + dane symulacji:

```
struct GlobalData {
    double alfa = 0.0; //współczynnik konwekcji
    double tOt = 0.0; //temperatura otoczenia
    double cp = 0.0; //ciepło właściwe
    double gamma = 0.0; //gęstość
    double simulationTime = 0.0; // czas symulacji
    double delt = 0.0; //interwał (co ile krok)
    double pc = 0.0; // przewodniość cieplna
    double initialTemp = 0.0; // temp poczatkowa
    vector<NODE> pom_nN;
    vector<ELEMENT> pom_nE;

} dane;
```

Ostatnim zadaniem podczas pierwszego laboratorium było zaimplementowanie wczytania danych z pliku tekstowego “Test1_4_4.txt” otrzymanego od prowadzącego. W tym celu napisano funkcję “*readFromFile()*”, realizują wczytanie z pliku dla dowolnie ułożonych danych wewnątrz niego:

```
void readFromFile() {
    double x = 0.0;
    string usun;
    string wyraz;


    ifstream odczyt;
    odczyt.open("Test1_4_4.txt");

    do {
        odczyt >> wyraz;
        if (wyraz == "SimulationTime") {
            odczyt >> dane.simulationTime;
        }
        else if (wyraz == "SimulationStepTime") {
            odczyt >> dane.delt;
        }
        else if (wyraz == "Conductivity") {
            odczyt >> dane.pc;
        }
        else if (wyraz == "Alfa") {
            odczyt >> dane.alfa;
        }
        else if (wyraz == "Tot") {
            odczyt >> dane.tOt;
        }
        else if (wyraz == "InitialTemp") {
            odczyt >> dane.initialTemp;
        }
        else if (wyraz == "Density") {
            odczyt >> dane.gamma;
        }
        else if (wyraz == "SpecificHeat") {
            odczyt >> dane.cp;
        }
        else if (wyraz == "Nodes") {
            odczyt >> wyraz;
            odczyt >> struktura.nN;
            dane.pom_nN = Utworz_nN(struktura.nN);
```

```
            for (int i = 0; i < struktura.nN; i++) { //uzupełnienie temperatury węzłów
                dane.pom_nN[i].nodeTemp = dane.initialTemp;
            }
        }
        else if (wyraz == "Elements") {
            odczyt >> wyraz;
            cout << wyraz << endl;
            odczyt >> struktura.nE;
            dane.pom_nE = Utworz_nE(struktura.nE);
        }
        else if (wyraz == "*Node" || wyraz == "*BC") {
            if (wyraz == "*Node") {
                for (int i = 0; i < struktura.nN; i++) {
                    odczyt >> wyraz;
                    odczyt >> dane.pom_nN[i].x;
                    odczyt >> usun;
                    odczyt >> dane.pom_nN[i].y;
                }
            } else {
                int pom = 0;
                for (;;) {
                    odczyt >> pom;
                    dane.pom_nN[pom-1].BC += 1;
                    odczyt >> usun;
                    if (odczyt.eof()) { break; }
                }
            }
        }
        else if (wyraz == "*Element,") {
            odczyt >> wyraz;
            for (int i = 0; i < struktura.nE; i++) {
                odczyt >> wyraz;
                for (int j = 0; j < 4; j++) {
                    odczyt >> dane.pom_nE[i].ID[j];
                    if (j < 3) odczyt >> usun;
                }
            }
        }
        else {
            continue;
        }
    } while (!odczyt.eof());
}
```

Dodatkowo zaimplementowano funkcje wykorzystywaną do tworzenia węzłów i elementów:

```
vector <NODE> Utworz_nN(int i) {
    vector <NODE> pom_nN(i + 2);
    return pom_nN;
}

vector <ELEMENT> Utworz_nE(int i) {
    vector <ELEMENT> pom_nE(i + 2);
    return pom_nE;
}
```

/Aby uniknąć błędów związanych z pamięcią, zadeklarowano większy rozmiar danego wektora niż wymagany (o 2)/

Element4:

```
struct Element4 {
    friend Bok;
public:
    //Punkty całkowania:
    vector <double> ksi;
    vector <double> eta;

    //Do macierzy Jakobiego (zwykłego i odwróconego):
    vector <vector <double>> pochodna_ksi;
    vector <vector <double>> pochodna_eta;
    vector <vector <double>> pochodnaDlaNPoX;
    vector <vector <double>> pochodnaDlaNPoY;

    vector <vector <double>> pochodna_Po_X_Y;
    vector <vector <double>> odwroconaDlaEta;
    vector <vector <double>> odwroconaDlaKsi;

    vector <double> pochodna_Po_Ksi;
    vector <double> pochodna_Po_Eta;

    //częściowe macierze H
    vector <vector <double>> macierzH;

    //zsumowane macierze H dla danego elementu
    vector <vector <double>> macierzHFinall;

    //Warunki brzegowe na poszczególnych bokach:
    vector <Bok> boki;

    //Macierz warunków brzegowych
    vector<vector<vector<double>>> HBC;

    //Wektor obciążeń
    vector <vector <double>> P;
    vector <vector <double>> pAgregowane;
```

```
    //Macierz C
    vector<vector<double>> macierzC;
    vector <vector <double>> macierzCFinall;

    //Funkcje kształtu po objętości
    vector <vector <double>> N;
    vector <vector <double>> NT;

    //Temperatura na węzłach
    vector <vector <double>> tempNaWezlach;

    //"Finałowa macierz" (agregowana macierz HBC + H)
    vector <vector <double>> agregacja;

    //"Finałowa macierz" (agregowana macierz C)
    vector <vector <double>> agregacjaMacierzyC;

    //wyznacznik macierzy
    vector <double> detJ;

    vector <double> wspolczynnik;
    //ilość punktów całkowania
    int rozmiar = 0;

    Element4(int value, int punktyBok);
    void pchodnKsi();//Obliczanie pochodnej N po ksi
    void pochodnaEta();//Obliczanie pochodnej N po eta
    void jakobian();//Obliczanie macierzy Jakobiego wraz z wyznacznikiem
    void macierzOdwrotna(); //Obliczanie macierzy odwrotnej Jakobiego
    void pochodnaDlaN(); //Obliczanie pochodnej N po x oraz N po Y
    void macierzHBCorazWektorP(); //Zsumowanie macierzyHBC i P z poszczególnych boków
    void macierzHplusC();//Zsumowanie macierzyH oraz macierzyC z punktów całkowania dla poszczególnych elementów
    void Agregacja(); //Agregacja macierzy H oraz macierzy C
    void obliczanieTemperaturyNaWezlach();//Przeprowadzenie symulacji
    //Tworzenie funkcji kształtu dla całki po objętości:
    vector<vector<double>> stworzMacierzN(vector<double> eta, vector<double>ksi, int iloscPc);
```

Podczas realizacji dalszej części kursu utworzono strukturę “Element4” wewnątrz której znalazły się pola i metody odpowiedzialne za poszczególne kroki całkowania macierzy H, takie jak:

- Obliczanie pochodnych $\frac{\partial N}{\partial \xi}$ oraz $\frac{\partial N}{\partial \eta}$ - “*pchodnKsi()”* oraz “*pochodnaEta()”*;
- Obliczanie macierzy Jacobiego [J] oraz wyznacznika (detJ) - “*detJakobian()*”;
- Obliczanie macierzy odwrotnej Jacobiego $[J^{-1}]$;
- Obliczanie pochodnych $\frac{\partial N}{\partial x}$ oraz $\frac{\partial N}{\partial y}$;
- Sumowanie macierzy H z punktów całkowania dla poszczególnych elementów - *“macierzHplusC()” ;*

Obok macierzy H tworzona jest również macierz C. Struktura ta zawiera też pola i metody odpowiedzialne za zsumowanie oraz zapisanie wartości macierzy HBC oraz wektora P. Ponadto w strukturze tej wykonywana jest również agregacja oraz symulacja.

Domyślnie struktura ta powinna reprezentować wartości charakterystyczne wyłącznie dla elementu czterowęzłowego (w przypadku elementów o różnej liczbie węzłów należałoby stworzyć nową strukturę np Element3 dla elementu 3 węzłowego). Niestety, wewnątrz niej zostały dodane pola i metody które powinny należeć do osobnej struktury (element Uniwersalny), bądź powinny być osobnymi funkcjami. Nie ma to jednak wpływu na wynik końcowy symulacji.

Bok - warunek brzegowy konwekcji:

```
struct Bok { //warunek brzegowy konwekcji
    int iloscPc = 0;
    int ID; //wybor budowanego boku - 0 dolny bok, 1 prawy bok, 2 górny bok, 3 lewy bok.
    vector<int> nrElementu;
    vector <double> ksi;
    vector <double> eta;
    vector <double> wspolczynnik;
    vector<vector<double>> P; //wektor obciążeń
    vector<vector<double>> N; //funkcja kształtu
    vector<vector<double>> NT;//funkcja ksztaltu transponowana
    vector<vector<vector<double>>> HBC;//Macierz HBC
    Bok(int iloscPunktow, int ID);
    void stworzBok();
};
```

Ostatnią implementowaną strukturą jest Bok wewnątrz której znajdują się pola i metdy dokonujące obliczenia dla całki po powierzchni - w przypadku rozważanego oprogramowania, wewnątrz struktury wyliczana jest macierz HBC oraz wektor P.

Metoda “*stworzBok()*” wykonuje obliczenia dla danego boku elementu, jeśli istnieje na nim warunek brzegowy.

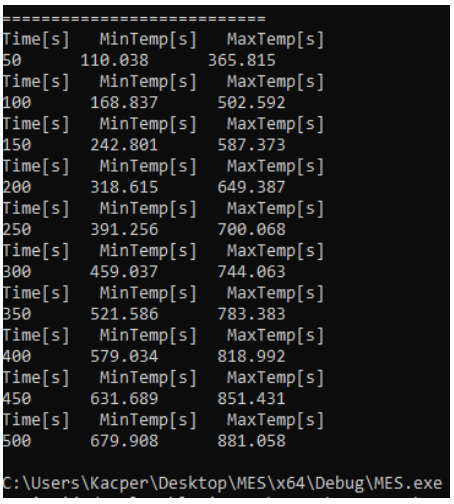
3.2 Przeprowadzanie symulacji:

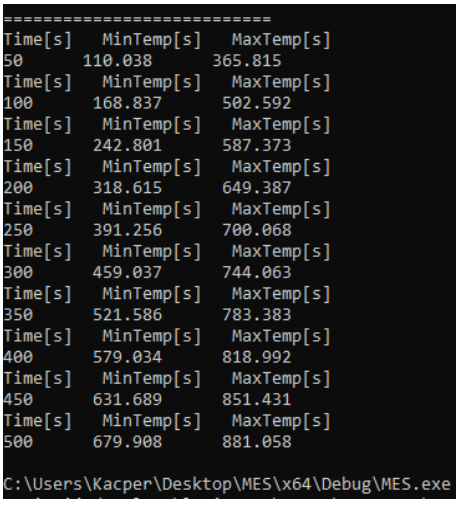
4. Analiza otrzymanych wyników:

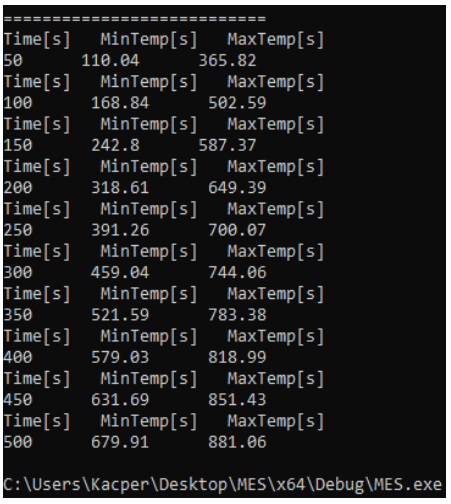
4.1.1 Wzorcowe wartości temperatur minimalnej i maksymalnej na węzłach dla siatki “Test1_4_4.txt”:

Time[s]	MinTemp[s]	MaxTemp[s]
50	110.038	365.815
100	168.837	502.592
150	242.801	587.373
200	318.615	649.387
250	391.256	700.068
300	459.037	744.063
350	521.586	783.383
400	579.034	818.992
450	631.689	851.431
500	679.908	881.058

4.1.2 Otrzymane wartości temperatur minimalnej i maksymalnej na węzłach dla siatki “Test1_4_4.txt” dla dwu-, trój- oraz czteropunktowego schematu całkowania:



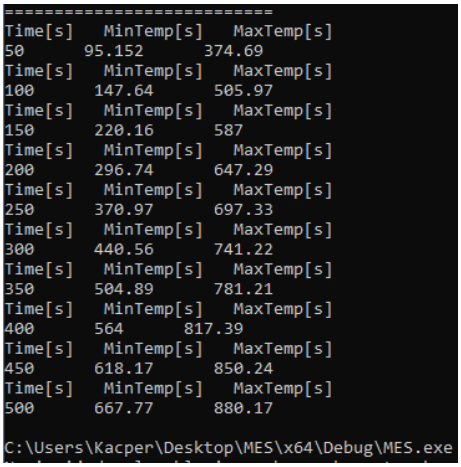


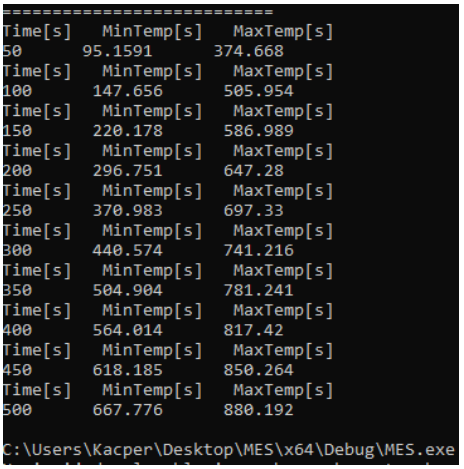


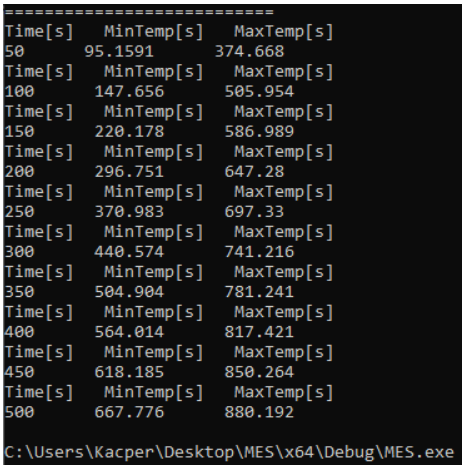
4.2.1 Wzorcowe wartości temperatur minimalnej i maksymalnej na węzłach dla siatki “Test2_4_4_MixGrid.txt”:

```
Temperature results

Time = 50 min_T= 95.152, max_T = 374.69

Time = 100 min_T= 147.64, max_T = 505.97

Time = 150 min_T= 220.16, max_T = 587

Time = 200 min_T= 296.74, max_T = 647.29

Time = 250 min_T= 370.97, max_T = 697.33

Time = 300 min_T= 440.56, max_T = 741.22

Time = 350 min_T= 504.89, max_T = 781.21

Time = 400 min_T= 564, max_T = 817.39

Time = 450 min_T= 618.17, max_T = 850.24

Time = 500 min_T= 667.77, max_T = 880.17
```

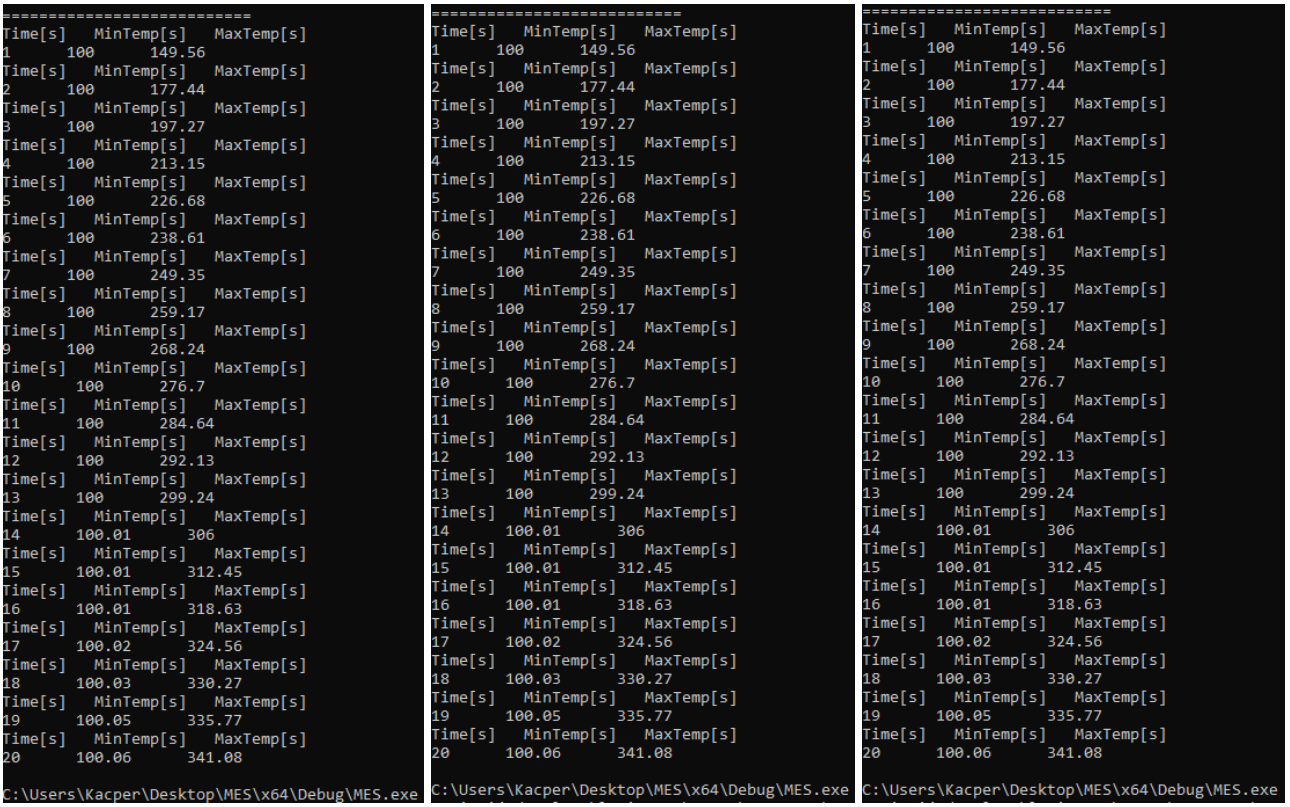
4.2.2 Otrzymane wartości temperatur minimalnej i maksymalnej na węzłach dla siatki “Test2_4_4_MixGrid.txt” dla dwu-, trój- oraz czteropunktowego schematu całkowania:







4.3 Otrzymane wartości temperatur minimalnej i maksymalnej na węzłach dla siatki “Test2_4_4_MixGrid.txt” dla dwu-, trój- oraz czteropunktowego schematu całkowania:



5. Symulacja:

Symulację wykonano dla problemu rzeczywistego, jakim jest ściana nośna (zewnętrzna) w domu jednorodzinnym. Posłużono się fragmentem ściany o wymiarach 220 mm x 220 mm. Pominięto warstwy dla których wielkość sprawia, że przeprowadzenie jej wymagałoby znacznie wyższej mocy obliczeniowej, niż dostępna (ze względu na ilość niezbędnych elementów). Do generowania siatki MES posłużono się kodem otrzymanym od prowadzącego.

W celu jak najlepszego odwzorowania symulacji, posłużono się nowoczesnymi materiałami (po wcześniejszej konsultacji z architektem). Informację o właściwościach zaczerpnięto ze strony producentów. W ten sposób otrzymano trzy warstwową ścianę zbudowaną z:

- Bloczków H+H Gold o szerokości 175 mm. Deklarowane własności na stronie producenta to:
  - Współczynnik konwekcyjnej wymiany ciepła: 0,20 [$\frac{W}{m^2*K}$]; (W/m2 °C);
  - Ciepło właściwe - 840 [$\frac{J}{kg*K}$];
  - Gęstość - 385 [$\frac{kg}{m^3}$];

- Przewodność cieplna - 0,105 [$\frac{W}{m*K}$];

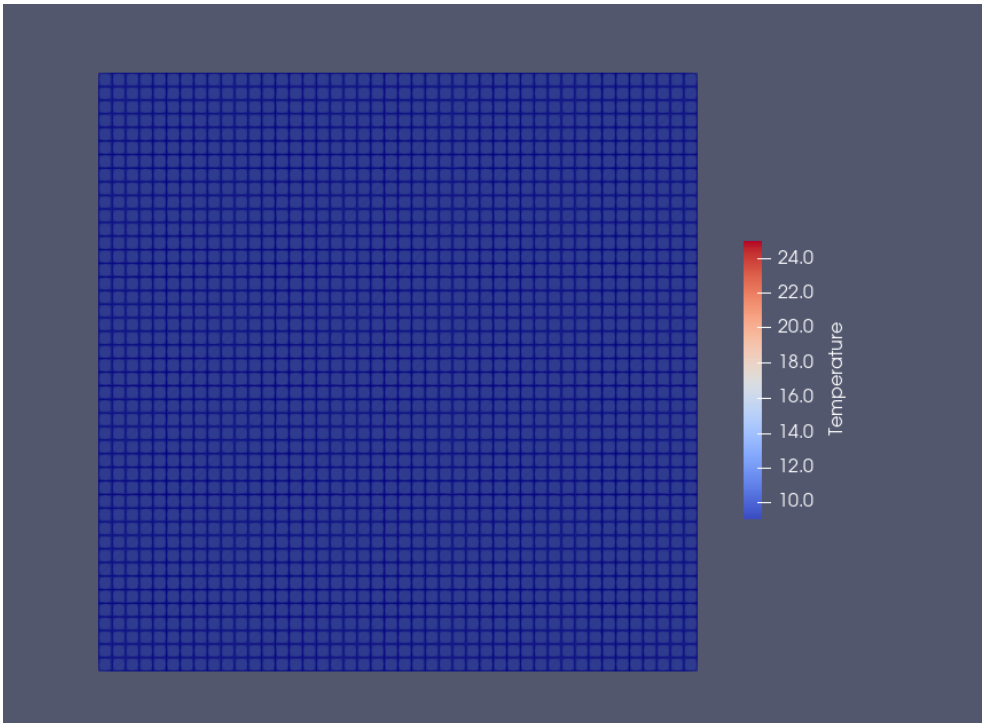
- Rockwool - płyta lamelowa z wełny skalnej Frontrock L o szerokości 300 mm. Deklarowane właściwości na stronie producenta to:
  - Współczynnik konwekcyjnej wymiany ciepła: 1[$\frac{W}{m^2*K}$]; (W/m2 °C);
  - Ciepło właściwe - 750 [$\frac{J}{kg*K}$];
  - Gęstość - 78 [$\frac{kg}{m^3}$];
  - Przewodność cieplna - 0,041 [$\frac{W}{m*K}$];

- Deski z drewna dębowego cięte wzdłuż włókien. Deklarowane właściwości to:
  - Współczynnik konwekcyjnej wymiany ciepła: 27 [$\frac{W}{m^2*K}$];
  - Ciepło właściwe - 2510 [$\frac{J}{kg*K}$];
  - Gęstość - 800 [$\frac{kg}{m^3}$];
  - Przewodność cieplna - 0,40 [$\frac{W}{m*K}$];

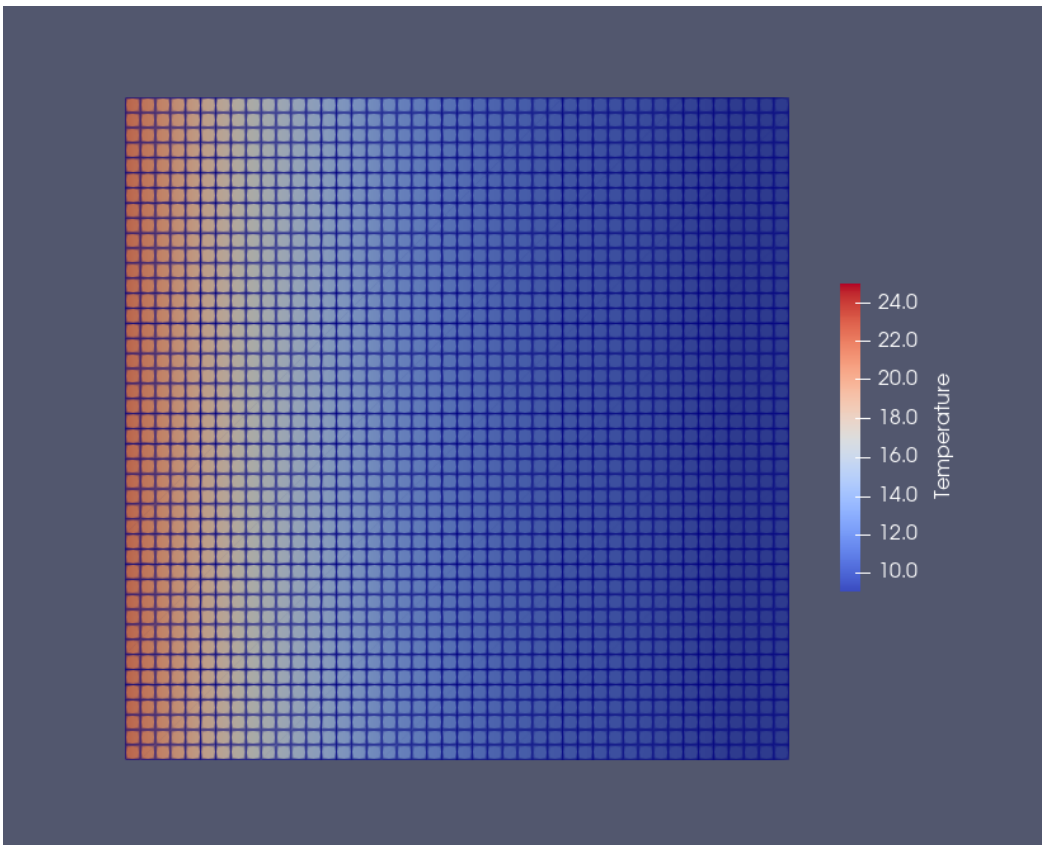
Koszt wyżej wymienionych materiałów materiałów umożliwiający postawienie o wymiarach 5 x 3 m oscyluje w granicach od 3300 zł do 4000 zł w zależności od ceny drewna.

Symulację przeprowadzono dla kroku czasowego równego 12 godzin, czasu symulacji równego 30 dni, temperatury zewnętrznej 9 °C i temperatury wewnętrznej równej 25 °C.

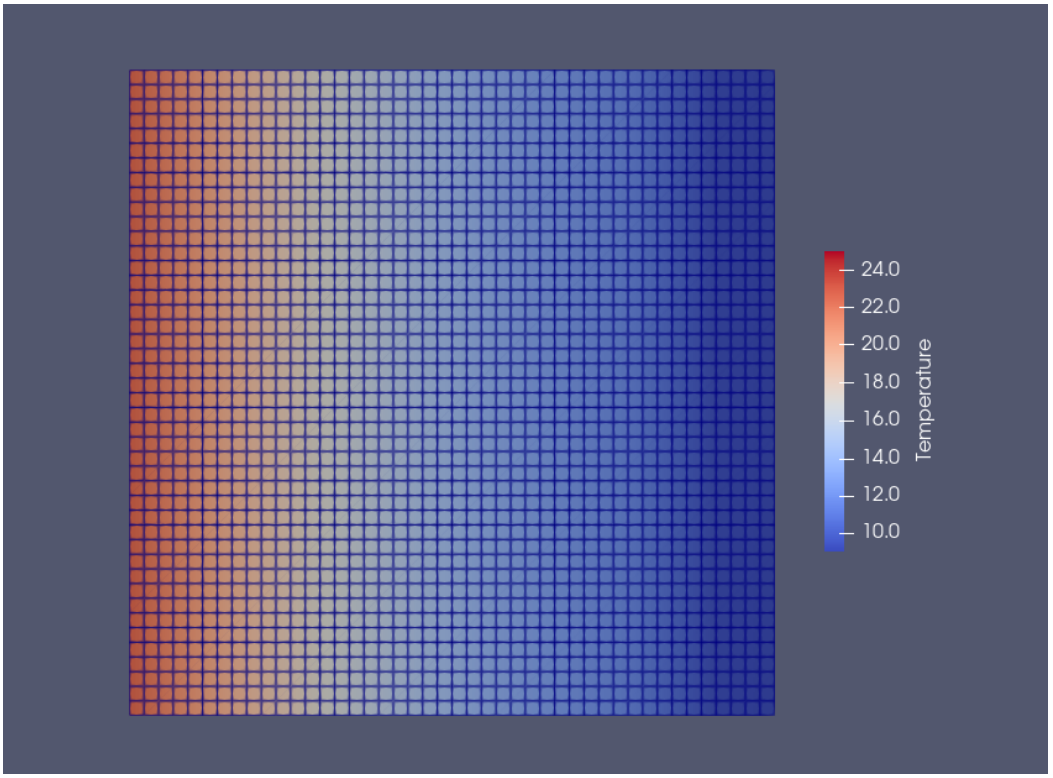
Siatka MES przed rozpoczęciem symulacji:



Siatka MES po 15 dniach:



Siatka MES po 30 dniach:

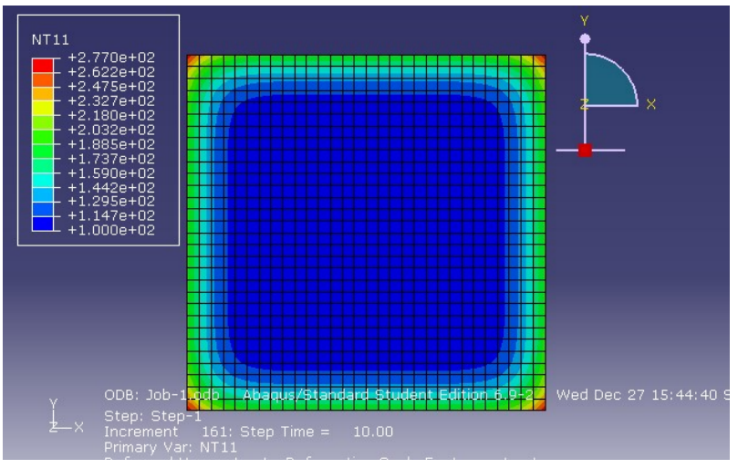


6. Wnioski:

Porównując otrzymane wartości temperatur z wartościami wzorcowymi, można wywnioskować, że oprogramowanie działa w sposób poprawny, pomimo wystąpienia pewnych odchyleń przy implementacji. Ponadto można zaobserwować, że wraz ze wzrostem temperatury, szybkość nagrzewania materiału maleje. Powodem tego są punkty pomiarowe - węzły. Gdy ich temperatura zbliża się do temperatury otoczenia, ciężej jest ją zwiększyć. Dodatkowo, korzystając z symulacji dostarczonej przez prowadzącego w pliku "TestCase.pdf" można zaobserwować, że nagrzewanie badanego materiału następuje od powierzchni bocznych (z zastrzeżeniem, że najszybciej materiał nagrzewany jest na rogach siatki) do wnętrza materiału. Ponadto wnętrze materiału (po pewnym czasie wskutek działających energii) ogrzewa się znacznie szybciej niż elementy siatki otaczające go:

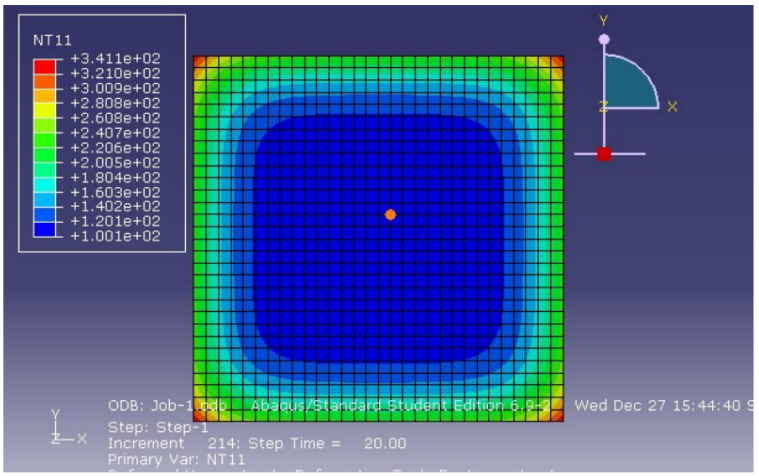
After 10 s

Time[s]	MinTemp	MaxTemp
1	100	149.56
2	100	177.44
3	100	197.27
4	100	213.15
5	100	226.68
6	100	238.61
7	100	249.35
8	100	259.17
9	100	268.24
10	100	276.7
11	100	284.64
12	100	292.13
13	100	299.24
14	100.01	306
15	100.01	312.45
16	100.01	318.63
17	100.02	324.56
18	100.03	330.27
19	100.05	335.77
20	100.06	341.08



After 20 s

Time[s]	MinTemp	MaxTemp
1	100	149.56
2	100	177.44
3	100	197.27
4	100	213.15
5	100	226.68
6	100	238.61
7	100	249.35
8	100	259.17
9	100	268.24
10	100	276.7
11	100	284.64
12	100	292.13
13	100	299.24
14	100.01	306
15	100.01	312.45
16	100.01	318.63
17	100.02	324.56
18	100.03	330.27
19	100.05	335.77
20	100.06	341.08



Utworzone oprogramowanie może przysłużyć się do celów praktycznych, czego przykładem jest utworzona symulacja ściany nośnej (zewnętrznej).

Analizując wyniki symulacji można wywnioskować, że stworzona ściana świetnie spisze się w warunkach mieszkalnych i pozwoli w znacznym stopniu ograniczyć koszty związane z ogrzewaniem, a w konsekwencji zaoszczędzić. Nie ma bowiem zbytnich strat cieplnych.

Metoda Elementów Skończonych daje duże możliwości i pozwala w praktyczny sposób zaimplementować wiedzę zdobytą podczas dotychczasowych studiów zarówno z programowania i algorytmiki, jak i przedmiotów związanych z wymianą ciepła i masy.