2025

Metoda elementów skończonych

SPRAWOZDANIE Z ĆWICZEŃ LABORATORYJNYCH

MICHAŁ PIENIEK

1. Cel ćwiczeń

Celem projektu było zaimplementowanie programu korzystającego z Metody Elementów Skończonych do symulacji przewodzenia ciepła w ośrodku dwuwymiarowym. Zaimplementowana metoda polega na podziale obszaru na siatkę elementów (4-węzłowych elementów skończonych). Główne kroki realizacji projektu polegały na obliczeniu lokalnych macierzy i wektorów, agregacja do macierzy i wektorów globalnych i rozwiązanie układu równań za pomocą metody eliminacji Gaussa.

2. Wstęp teoretyczny

Metoda Elementów Skończonych (MES, z ang. Finite Element Method – FEM) to numeryczna technika pozwalająca na przybliżone rozwiązywanie równań różniczkowych cząstkowych**.** Do kluczowych zamysłów tej techniki należą:

- Podział obszaru na **elementy skończone**, czyli dyskretyzacja przestrzeni – podział analizowanego obszaru na mniejsze elementy,
- Definicja **funkcji kształtu** w układzie lokalnym. Dla czterowęzłowego elementu przyjmuje się układ lokalnych współrzędnych (ξ,η), w którym ξ,η zmieniają się w przedziale [−1,1].

W MES każdy element jest opisywany za pomocą węzłów, a pole rozwiązań w elemencie aproksymowane jest przez funkcje kształtu. Funkcje te pozwalają na interpolację wartości między węzłami, umożliwiając ciągłą reprezentację rozwiązania w obrębie elementu.

Układ równań MES przedstawia się za pomocą wzoru:

$$([H] + \frac{[C]}{\Delta\tau})\{t_1\} - (\frac{[C]}{\Delta\tau})\{t_0\} + \{P\} = 0 \quad (1)$$

Gdzie $\{t_1\}$ oznacza wartość temperatur węzłowych po czasie $\Delta\tau$, natomiast wektor $\{t_0\}$ reprezentuje wartości temperatur w chwili T=0.

Poszczególne wyrażenia zawarte w równaniu (1) obliczane są za pomocą równań:

$$[\mathrm{H}] = \int_V -k\left(\{\frac{\partial N}{\partial x}\}\{\frac{\partial N}{\partial x}\}^T + \{\frac{\partial N}{\partial y}\}\{\frac{\partial N}{\partial y}\}^T\right)dV + \int_S \alpha\{N\}\{N\}^T dS \quad (2)$$

$$[P] = -\int_S \alpha\{N\}t_\infty \,\mathrm{dS} \quad (3)$$

$$[C] = \int_V c\rho\,\{N\}\{N\}^T dV \quad (4)$$

Macierz H (przewodnictwa cieplnego) opisuje zjawisko przepływu ciepła wewnątrz elementu skończonego. Uwzględnia przewodność materiału, która determinuje jak

efektywnie ciepło przemieszcza się przez dany obiekt. Zależna od współczynnika przewodności cieplnej (Conductivity).

Macierz C (pojemności cieplnej) reprezentuje zdolność materiału do akumulowania energii cieplnej. Obrazuje ile ciepła dany element może zmagazynować, zależnie od jego gęstości, ciepła właściwego i geometrii. Zależna od gęstości (Density) i pojemności cieplnej(SpecificHeat).

Macierz Hbc (warunków brzegowych) opisuje warunki brzegowe konwekcyjne, modeluje wymianę ciepła na granicy elementu z otoczeniem. Wprowadza zjawisko konwekcji poprzez współczynnik przejmowania ciepła alfa, który określa intensywność wymiany cieplnej między powierzchnią elementu a medium go otaczającym. Zależna od współczynnika wymiany ciepła alfa.

Wektor P podobnie jak macierz Hbc jest związany z warunkami brzegowymi konwekcji, odzwierciedla zewnętrzne oddziaływania temperaturowe na element. Wprowadza do modelu informacje o temperaturze otoczenia, strumieniu ciepła. Jest zależny od współczynnika wymiany ciepła alfa.

3. Struktura programu

Kod podzielony został na plik źródłowy oraz nagłówkowy.

Main.cpp – zawiera główną funkcję programu main(), w której znajdują się:

- Definicja współrzędnych węzłów dla 2,3,4 – punktowego schematu całkowania,
- Wczytywanie danych z pliku wejściowego (funkcja wczytajDane()),
- Obliczanie macierzy lokalnych (H, Hbc, C) dla każdego elementu,
- Obliczanie wektorów lokalnych P dla każdego elementu,
- Agregację macierzy oraz wektorów lokalnych elementów do reprezentacji globalnej,
- Zawiera funkcje obliczające elementy składowe równania (1) (funkcje createGlobal(), createRS()), oraz funkcje odpowiedzialne za rozwiązanie układu równań metodą eliminacji Gaussa oraz obliczanie temperatury w czasie (gaussElimination(), solve())
- Wyświetlanie wyników,
- Funkcje umożliwiające wyświetlanie macierzy i wektorów(printMatrix(), printVector()).

Main.cpp zawiera struktury danych:

- GlobalData – struktura przechowująca dane wczytane z pliku wejściowego

```
struct GlobalData {
    double SimulationTime;
    double SimulationStepTime;
    double Conductivity;
    double Alfa;
    double Tot;
    double InitialTemp;
    double Density;
    double SpecificHeat;
```

```
    int nodesNumber;
    int elementsNumber;
};
```

Gdzie kolejne wartości odpowiadają:

- SimulationTime – czas po jakim zakończy się symulacja,
- SimulationStepTime – czas co jaki będziemy wyliczać rozkład temperatury w modelu,
- Conductivity – przewodność cieplna używana do wyliczenia macierzy H,
- Alfa – współczynnik wymiany ciepła który pozwala nam obliczyć wektor P i macierz Hbc,
- Tot – temperatura otoczenia, z którego jest przekazywane ciepło do modelu,
- InitialTemp – początkowa temperatura w każdym węźle,
- Density – gęstość materiału używana do obliczenia macierzy C,
- SpecificHeat – pojemność ciepła właściwa używana do obliczania macierzy C,
- nodesNumber – ilość węzłów, z których składa się model,
- elementsNumber – ilość elementów, z których składa się model,

- Punkt – struktura przechowująca współrzędne wczytanych punktów oraz informację czy punkt jest brzegowy

```
struct Punkt {
   double X;
   double Y;
   bool BC; // czy węzeł jest brzegowy
};
```

- Element – struktura przechowująca ids Punktów z których składa się dany element, odpowiadające macierze H, Hbc, C i wektor P,

```
struct Element {
    int ids[4];
    double H[4][4];
    double Hbc[4][4];
    double P[4];
    double C[4][4];
};
```

- ElemUniv – struktura przechowująca pochodne funkcji kształtu obliczane tylko raz w działaniu programu wspólne dla wszystkich elementów, wielkość tablic dwuwymiarowych różni się w zależności od użytego schematu całkowania,

```
struct ElemUniv{
```

```
        double dNdKsi[npc][4];
        double dNdNi[npc][4];
    };
```

Header.h – plik nagłówkowy

- Zawiera definicje określające typ schematu całkowania 2,3,4- punktowy,

```
#define npc 16
#define NPCBC 4
```

- Strukturę Jakobian zawierającą metody obliczające odwrotność oraz wyznacznik Jakobianu,

```
struct Jakobian{
    double J[2][2];
    double J_odwr[2][2];
    double detJ;

    void oblicz() {
        double a = J[0][0];
        double b = J[0][1];
        double c = J[1][0];
        double d = J[1][1];

        // Obliczanie wyznacznika
        detJ = a * d - b * c;

        // Sprawdzenie, czy wyznacznik jest różny od zera (macierz jest
odwracalna)
        if (detJ != 0) {
            // Obliczanie odwrotności macierzy
            J_odwr[0][0] = d / detJ;
            J_odwr[0][1] = -b / detJ;
            J_odwr[1][0] = -c / detJ;
            J_odwr[1][1] = a / detJ;
        }
```

- Strukturę Macierz, zawierającą metodę do wypisywania Macierzy w konsoli

```
struct Macierz{
    double macierz[npc][4];

    void wypisz(std::string nazwa = "Macierz") const {
        std::cout << nazwa << ":" << std::endl;
        std::cout << std::fixed << std::setprecision(3);  // Ustawienie
precyzji na 3 miejsca po przecinku
        for (int i = 0; i < 4; ++i) {
            for (int j = 0; j < 4; ++j) {
                std::cout << std::setw(10) << macierz[i][j] << " ";
            }
```

```
            std::cout << std::endl;
        }
        std::cout << std::endl;
    }
};
```

4. Przebieg działania programu

Program realizuje postawione zagadnienie przez następujące kroki:

1) **Wczytanie danych**
   Działanie programu rozpoczyna się od wczytania danych z podanego pliku. Funkcja wczytajDane(), zapisuje wszystkie parametry modelu do struktury GlobalData, wczytuje współrzędne punktów i przypisuje odpowiednie id punktów poszczególnym elementom siatki.
2) **Obliczenie pochodnych funkcji kształtu**
   Do elementów uniwersalnych zapisywane są pochodne dNdKsi, dNdNi obliczone dla poszczególnych węzłów:

```
for (int i = 0; i < npc; i++) {
  double ksi = wezly[i].X;
  double ni = wezly[i].Y;

  pochodne.dNdKsi[i][0] = -0.25 * (1.0 - ni);
  pochodne.dNdKsi[i][1] =  0.25 * (1.0 - ni);
  pochodne.dNdKsi[i][2] =  0.25 * (1.0 + ni);
  pochodne.dNdKsi[i][3] = -0.25 * (1.0 + ni);

  pochodne.dNdNi[i][0] = -0.25 * (1.0 - ksi);
  pochodne.dNdNi[i][1] = -0.25 * (1.0 + ksi);
  pochodne.dNdNi[i][2] =  0.25 * (1.0 + ksi);
  pochodne.dNdNi[i][3] =  0.25 * (1.0 - ksi);
  }

  for (int i = 0; i < NPCBC * 4; ++i) {
    double ksi = ksibc[i];
    double eta = etabc[i];

    // Funkcje kształtu 4-węzłowego elementu:
    bcData.N[i][0] = 0.25 * (1.0 - ksi) * (1.0 - eta); // N1
    bcData.N[i][1] = 0.25 * (1.0 + ksi) * (1.0 - eta); // N2
    bcData.N[i][2] = 0.25 * (1.0 + ksi) * (1.0 + eta); // N3
    bcData.N[i][3] = 0.25 * (1.0 - ksi) * (1.0 + eta); // N4
  }
```

3) **Obliczenie Jakobianów oraz pochodnych dN/dX, dN/dY dla punktów całkowania**

Od tego momentu program przechodzi do pętli iterującej po wszystkich elementach. Ze struktury elementu wczytywane, odpowiednie współrzędne punktów do przeprowadzenia obliczeń. To etap transformacji pochodnych funkcji kształtu z lokalnego układu współrzędnych (ksi, ni) do globalnego układu współrzędnych (x, y) w metodzie elementów skończonych. Obejmuje obliczenie macierzy Jakobiego, wyznacznie odwrotności macierzy i na tej podstawie transformacje pochodnych funkcji kształtu z układu lokalnego do globalnego.

```
for (int j = 0; j < npc; j++) {
        double dxdKSI = pochodne.dNdKsi[j][0]*localPunkty[0].X +
pochodne.dNdKsi[j][1]*localPunkty[1].X +
                        pochodne.dNdKsi[j][2]*localPunkty[2].X +
pochodne.dNdKsi[j][3]*localPunkty[3].X;

        double dydKSI = pochodne.dNdKsi[j][0]*localPunkty[0].Y +
pochodne.dNdKsi[j][1]*localPunkty[1].Y +
                        pochodne.dNdKsi[j][2]*localPunkty[2].Y +
pochodne.dNdKsi[j][3]*localPunkty[3].Y;

        double dxdNI = pochodne.dNdNi[j][0]*localPunkty[0].X +
pochodne.dNdNi[j][1]*localPunkty[1].X +
                       pochodne.dNdNi[j][2]*localPunkty[2].X +
pochodne.dNdNi[j][3]*localPunkty[3].X;

        double dydNI = pochodne.dNdNi[j][0]*localPunkty[0].Y +
pochodne.dNdNi[j][1]*localPunkty[1].Y +
                       pochodne.dNdNi[j][2]*localPunkty[2].Y +
pochodne.dNdNi[j][3]*localPunkty[3].Y;

        Jakobiany[j].J[0][0] = dxdKSI;
        Jakobiany[j].J[1][0] = dxdNI;
        Jakobiany[j].J[0][1] = dydKSI;
        Jakobiany[j].J[1][1] = dydNI;
        Jakobiany[j].oblicz();
    }
```

Wyznacznik jakobianu J stanowi skalę zmiany objętości (powierzchni w 2D) podczas całkowania w przestrzeni (ksi, eta).

4) Całkowanie numeryczne

Całkowanie numeryczne to technika przybliżonego obliczania całek, które opisują właściwości fizyczne elementów modelu. W procesie tym wykorzystuje się węzły całkowania rozmieszczone wewnątrz elementu, przez użycie wag przypisanych poszczególnym punktom całkowania można dokładnie

aproksymować wartości całek opisujących własności materiału, takie jak przewodnictwo cieplne czy pojemność cieplna.

W zaimplementowanym kodzie proces jest realizowany dla czterowęzłowego elementu i w zależności od liczby punktów całkowania stosowany jest odpowiedni algorytm ważenia wartości. Efektem są macierze H, C, Hbc oraz wektor P, wykorzystywane w dalszych obliczeniach modelu fizycznego. Proces sprowadza się do zsumowania wartości w węzłach całkowania przemnożonych przez wagi. Wybór liczby punktów całkowania wpływa bezpośrednio na dokładność i złożoność obliczeniową.

```
for (int s = 0; s < npc; s++) {

        double N[4];
    {
        double ksi = wezly[s].X;
        double eta = wezly[s].Y;
        N[0] = 0.25 * (1.0 - ksi) * (1.0 - eta);
        N[1] = 0.25 * (1.0 + ksi) * (1.0 - eta);
        N[2] = 0.25 * (1.0 + ksi) * (1.0 + eta);
        N[3] = 0.25 * (1.0 - ksi) * (1.0 + eta);
    }
        for (int i = 0; i < 4; ++i) {
            for (int j = 0; j < 4; ++j) {
                H_X[s].macierz[i][j] = dNdx[s][i] * dNdx[s][j] *
Jakobiany[s].detJ;
                H_Y[s].macierz[i][j] = dNdy[s][i] * dNdy[s][j] *
Jakobiany[s].detJ;
                H_Suma[s].macierz[i][j] = (H_X[s].macierz[i][j] +
H_Y[s].macierz[i][j]) * globalData.Conductivity;
                C_Suma[s].macierz[i][j] = N[i] * N[j] *
Jakobiany[s].detJ * globalData.Density * globalData.SpecificHeat;
            }
        }
    }
```

Otrzymane macierze mnożone są przez odpowiednie wagi zależnie od wybranego schematu całkowania oraz sumowane do macierzy wynikowej, a następnie zapisywane do struktury elementu.

```
else if (npc == 9) {
        for (int s = 0; s < npc; ++s) {
            int row = s / 3; // wiersz w macierzy 3x3
            int col = s % 3; // kolumna w macierzy 3x3

            w1 = waga[row];
            w2 = waga[col];

            for (int i = 0; i < 4; ++i) {
```

```
                for (int j = 0; j < 4; ++j) {
                    wynik.macierz[i][j] += H_Suma[s].macierz[i][j] *
w1 * w2;
                    wynikC.macierz[i][j] += C_Suma[s].macierz[i][j] *
w1 * w2;
                }
            }
        }
    }
    else if (npc == 16) {
        for (int s = 0; s < npc; ++s) {
            int row = s / 4;
            int col = s % 4;
            w1 = waga[row];
            w2 = waga[col];
            for (int i = 0; i < 4; ++i) {
                for (int j = 0; j < 4; ++j) {
                    wynik.macierz[i][j]  += H_Suma[s].macierz[i][j] *
w1 * w2;
                    wynikC.macierz[i][j] += C_Suma[s].macierz[i][j] *
w1 * w2;
                }
            }
        }
```

Zapisanie otrzymanej macierzy H do struktury elementu.

```
// Zapisujemy obliczone macierze H i C do struktury elementu
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            elementy[e].H[i][j] = wynik.macierz[i][j];
        }
    }
```

5) Agregacja do postaci globalnej

Przechodzimy pętlą przez wszystkie elementy siatki, a następnie przez lokalne węzły każdego elementu (indeksy j i k pętli). Wartości z lokalnych macierzy H, Hbc,C i wektora P są sumowane do odpowiednich miejsc w macierzach i wektorze globalnym.

```
for (int j = 0; j < 4; ++j) {
        int globalJ = elementy[i].ids[j] - 1;  // -1 bo numeracja w pliku
zaczyna sie od 1
        for (int k = 0; k < 4; ++k) {
            int globalK = elementy[i].ids[k] - 1;
```

```
            // Dodajemy macierze elementow H do macierzy globalnej H
            HGlobal[globalJ][globalK] += elementy[i].H[j][k];
            // Dodajemy macierze elementow Hbc do macierzy globalnej H
            HGlobal[globalJ][globalK] += elementy[i].Hbc[j][k];
            // Dodajemy macierze elementow C do macierzy globalnej H
            CGlobal[globalJ][globalK] += elementy[i].C[j][k];
        }
        // Dodajemy do wektora globalnego
        PGlobal[globalJ] += elementy[i].P[j];
    }
```

6) Rozwiązanie stworzonego układu

Funkcja createGlobal()

Oblicza macierz globalną Global sumując macierz przewodzenia H_{Global} i macierz pojemności C_{Global} podzieloną przez krok czasowy $\Delta\tau$.

$$Global = H_{Global} + \frac{C_{Global}}{\Delta\tau} \quad (5)$$

```
Matrix createGlobal(const Matrix &Hglobal,
                    const Matrix &Cglobal,
                    double dt)
{
    int N = (int)Hglobal.size();
    Matrix Global(N, Vector(N, 0.0));
    // Tworzymy macierz Globalna wg wzoru
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            Global[i][j] = Hglobal[i][j] + Cglobal[i][j] / dt;
        }
    }
    return Global;
}
```

Funkcja createRS()

Tworzy wektor prawej strony równania, w którym pierwszy wyraz $\frac{C_{Global}}{\Delta\tau}T^n$ pochodzi z członu nieustalonego, a P_{Global}, reprezentuje źródła i warunki brzegowe (np. konwekcje).

$$RS = \frac{C_{Global}}{\Delta\tau}T^n + P_{Global} \quad (6)$$

```
Vector createRS(const Matrix &Cglobal, double dt,const Vector
&Tn,const Vector &Pglobal)
{
    int N = (int)Cglobal.size();
    Vector RS(N, 0.0);
```

```
    // Tworzymy wektor RS wg wzoru
    for (int i = 0; i < N; i++) {
        double sumC = 0.0;
        for (int j = 0; j < N; j++) {
            sumC += (Cglobal[i][j] / dt) * Tn[j];
        }
        RS[i] = sumC + Pglobal[i];
    }
    return RS;
}
```

Wzór ten stanowi prawą stronę równania:

$$(H + \frac{C_{Global}}{\Delta\tau})T^{n+1} = \frac{C_{Global}}{\Delta\tau}T^n + P \quad (7)$$

Funkcja gaussElimination()

Rozwiązuje układ liniowy Ax = b metodą eliminacji Gaussa. Eliminacja w przód zamienia macierz A na trójkątną górną, a eliminacja wstecz wylicza nieznane zmienne x. Eliminacja Gaussa to standardowy algorytm wyznaczania rozwiązań układów liniowych.

Funkcja solve()

- Tworzy macierz $Global = H_{Global} + \frac{C_{Global}}{\Delta\tau}$
- Inicjuje wektor temperatur T_0 wartością initialTemp z pliku wejściowego
- Implementuje pętle, dla kolejnych kroków czasowych:
  - Oblicza wektor prawej strony (6)
  - Rozwiązuje układ Global * T = RS, metodą eliminacji Gaussa
  - Wyświetla wyniki (maksymalną i minimalną temperaturę)
  - Aktualizuje wektor temperatur danymi z obecnej iteracji

W ten sposób w każdym kroku czasowym liczymy nowy rozkład temperatur T^{n+1}, co sprowadza się do rozwiązania równania:

$$(H_{Global} + \frac{C_{Global}}{\Delta\tau})T^{n+1} = \frac{C_{Global}}{\Delta\tau}T^n + P_{Global}$$

```
void solve(const Matrix &Hglobal,
                    const Matrix &Cglobal,
                    const Vector &Pglobal,
                    double dt,
                    double totalTime,
                    double initialTemp)
{
```

```
    int N = (int)Hglobal.size();
    Matrix Global = createGlobal(Hglobal, Cglobal, dt); // Obliczamy macierz
Global = H + C/dt
    Vector T0(N, initialTemp);                          // Wektor temperatur
    Vector T(N, 0.0);                                    //wektor pomocniczy
    Vector RS = createRS(Cglobal, dt, T0, Pglobal);  // Obliczamy wektor RS =
(C/dt)*T0 + Pglobal - prawa strona rownania
    printVector(RS, "Iteracja 0");
    printMatrix(Global, "Iteracja 0");
    for (double currentTime = dt; currentTime <= totalTime; currentTime += dt)
{
        Vector RS = createRS(Cglobal, dt, T0, Pglobal);  // Obliczamy wektor
RS = (C/dt)*T0 + Pglobal - prawa strona rownania
        T = gaussElimination(Global, RS); // Rozwiązujemy układ: Global * T =
RS

        double minT = *std::min_element(T.begin(), T.end());
        double maxT = *std::max_element(T.begin(), T.end());
        cout << "Czas = " << currentTime
             << " [s], Tmin = " << minT
             << ", Tmax = " << maxT << endl;

        // Przepisujemy T -> T0 (przygotowanie do następnej iteracji)
        T0 = T;
    }
}
```

5. Testy programu

Dla sprawdzenia poprawności implementacji przeprowadzono testy dla zamieszczonych na platformie UPEL siatek 4x4, 4x4_mix oraz 31x31. Symulacje zostały przeprowadzone dla trzech wariantów schematu całkowania (2-, 3-, 4-punktowy), aby sprawdzić jak różni się dokładność wyników przy zastosowaniu różnych wariantów.

Siatka 4x4

Schemat 2-punktowy

```
Czas = 50.0000 [s], Tmin = 110.0380, Tmax = 365.8155
Czas = 100.0000 [s], Tmin = 168.8370, Tmax = 502.5917
Czas = 150.0000 [s], Tmin = 242.8008, Tmax = 587.3727
Czas = 200.0000 [s], Tmin = 318.6146, Tmax = 649.3875
Czas = 250.0000 [s], Tmin = 391.2558, Tmax = 700.0684
Czas = 300.0000 [s], Tmin = 459.0369, Tmax = 744.0633
Czas = 350.0000 [s], Tmin = 521.5863, Tmax = 783.3828
Czas = 400.0000 [s], Tmin = 579.0345, Tmax = 818.9922
Czas = 450.0000 [s], Tmin = 631.6893, Tmax = 851.4310
Czas = 500.0000 [s], Tmin = 679.9076, Tmax = 881.0576
PS C:\Users\Majkel\Desktop\STUDIA\SEMESTR_V\MES\projektMES>
```

Schemat 3-punktowy

```
Czas = 50.0000 [s], Tmin = 110.0380, Tmax = 365.8155
Czas = 100.0000 [s], Tmin = 168.8370, Tmax = 502.5917
Czas = 150.0000 [s], Tmin = 242.8008, Tmax = 587.3727
Czas = 200.0000 [s], Tmin = 318.6146, Tmax = 649.3875
Czas = 250.0000 [s], Tmin = 391.2558, Tmax = 700.0684
Czas = 300.0000 [s], Tmin = 459.0369, Tmax = 744.0633
Czas = 350.0000 [s], Tmin = 521.5863, Tmax = 783.3828
Czas = 400.0000 [s], Tmin = 579.0345, Tmax = 818.9922
Czas = 450.0000 [s], Tmin = 631.6893, Tmax = 851.4310
Czas = 500.0000 [s], Tmin = 679.9076, Tmax = 881.0576
PS C:\Users\Majkel\Desktop\STUDIA\SEMESTR_V\MES\projektMES>
```

Schemat 4-punktowy

```
Czas = 50.0000 [s], Tmin = 110.0380, Tmax = 365.8155
Czas = 100.0000 [s], Tmin = 168.8370, Tmax = 502.5917
Czas = 150.0000 [s], Tmin = 242.8008, Tmax = 587.3727
Czas = 200.0000 [s], Tmin = 318.6146, Tmax = 649.3875
Czas = 250.0000 [s], Tmin = 391.2558, Tmax = 700.0684
Czas = 300.0000 [s], Tmin = 459.0369, Tmax = 744.0633
Czas = 350.0000 [s], Tmin = 521.5863, Tmax = 783.3828
Czas = 400.0000 [s], Tmin = 579.0345, Tmax = 818.9922
Czas = 450.0000 [s], Tmin = 631.6893, Tmax = 851.4310
Czas = 500.0000 [s], Tmin = 679.9076, Tmax = 881.0576
PS C:\Users\Majkel\Desktop\STUDIA\SEMESTR_V\MES\projektMES>
```

Wyniki do porównania dostarczone przez prowadzącego

4x4:

110.03797659406167	**365.8154705784631**
168.83701715655656	**502.5917120896439**
242.80085524391868	**587.372666691486**
318.61459376004086	**649.3874834542602**
391.2557916738893	**700.0684204214381**
459.03690325635404	**744.0633443187048**
521.5862742337766	**783.382849723737**
579.0344449687701	**818.9921876836681**
631.6892368621455	**851.4310425916341**
679.9075931513394	**881.057634906017**

Siatka 4x4 MIX

Schemat 2-punktowy

```
Czas = 50.0000 [s], Tmin = 95.1518, Tmax = 374.6863
Czas = 100.0000 [s], Tmin = 147.6444, Tmax = 505.9681
Czas = 150.0000 [s], Tmin = 220.1645, Tmax = 586.9978
Czas = 200.0000 [s], Tmin = 296.7364, Tmax = 647.2856
Czas = 250.0000 [s], Tmin = 370.9683, Tmax = 697.3340
Czas = 300.0000 [s], Tmin = 440.5601, Tmax = 741.2191
Czas = 350.0000 [s], Tmin = 504.8912, Tmax = 781.2096
Czas = 400.0000 [s], Tmin = 564.0015, Tmax = 817.3915
Czas = 450.0000 [s], Tmin = 618.1739, Tmax = 850.2373
Czas = 500.0000 [s], Tmin = 667.7656, Tmax = 880.1676
```

Schemat 3-punktowy

```
Czas = 50.0000 [s], Tmin = 95.1591, Tmax = 374.6683
Czas = 100.0000 [s], Tmin = 147.6559, Tmax = 505.9543
Czas = 150.0000 [s], Tmin = 220.1781, Tmax = 586.9895
Czas = 200.0000 [s], Tmin = 296.7508, Tmax = 647.2801
Czas = 250.0000 [s], Tmin = 370.9826, Tmax = 697.3299
Czas = 300.0000 [s], Tmin = 440.5740, Tmax = 741.2157
Czas = 350.0000 [s], Tmin = 504.9043, Tmax = 781.2408
Czas = 400.0000 [s], Tmin = 564.0139, Tmax = 817.4204
Czas = 450.0000 [s], Tmin = 618.1855, Tmax = 850.2640
Czas = 500.0000 [s], Tmin = 667.7764, Tmax = 880.1922
```

Schemat 4-punktowy

```
Czas = 50.0000 [s], Tmin = 95.1591, Tmax = 374.6683
Czas = 100.0000 [s], Tmin = 147.6559, Tmax = 505.9543
Czas = 150.0000 [s], Tmin = 220.1781, Tmax = 586.9894
Czas = 200.0000 [s], Tmin = 296.7509, Tmax = 647.2801
Czas = 250.0000 [s], Tmin = 370.9826, Tmax = 697.3299
Czas = 300.0000 [s], Tmin = 440.5740, Tmax = 741.2156
Czas = 350.0000 [s], Tmin = 504.9044, Tmax = 781.2409
Czas = 400.0000 [s], Tmin = 564.0139, Tmax = 817.4205
Czas = 450.0000 [s], Tmin = 618.1855, Tmax = 850.2641
Czas = 500.0000 [s], Tmin = 667.7764, Tmax = 880.1923
PS C:\Users\Majkel\Desktop\STUDIA\SEMESTR_V\MES\projektMES>
```

Wyniki do porównania dostarczone przez prowadzącego

4x4 mix:

95.15184673458245	**374.6863325385064**
147.64441665454345	**505.96811082245307**
220.1644549730314	**586.9978503916302**
296.7364399006366	**647.28558387732**
370.968275802604	**697.3339863103786**
440.5601440058566	**741.2191121514377**
504.8911996551285	**781.209569726045**
564.0015111915015	**817.3915065469778**
618.1738556427995	**850.2373194670416**
667.7655470268747	**880.1676054000437**

Siatka 31x31

Schemat 2-punktowy

```
Czas = 1.0000 [s], Tmin = 100.0000, Tmax = 149.5570
Czas = 2.0000 [s], Tmin = 100.0000, Tmax = 177.4449
Czas = 3.0000 [s], Tmin = 100.0000, Tmax = 197.2670
Czas = 4.0000 [s], Tmin = 100.0000, Tmax = 213.1528
Czas = 5.0000 [s], Tmin = 100.0000, Tmax = 226.6826
Czas = 6.0000 [s], Tmin = 100.0000, Tmax = 238.6071
Czas = 7.0000 [s], Tmin = 100.0000, Tmax = 249.3467
Czas = 8.0000 [s], Tmin = 100.0001, Tmax = 259.1651
Czas = 9.0000 [s], Tmin = 100.0002, Tmax = 268.2407
Czas = 10.0000 [s], Tmin = 100.0004, Tmax = 276.7011
Czas = 11.0000 [s], Tmin = 100.0008, Tmax = 284.6413
Czas = 12.0000 [s], Tmin = 100.0016, Tmax = 292.1342
Czas = 13.0000 [s], Tmin = 100.0029, Tmax = 299.2374
Czas = 14.0000 [s], Tmin = 100.0051, Tmax = 305.9971
Czas = 15.0000 [s], Tmin = 100.0086, Tmax = 312.4512
Czas = 16.0000 [s], Tmin = 100.0137, Tmax = 318.6312
Czas = 17.0000 [s], Tmin = 100.0212, Tmax = 324.5635
Czas = 18.0000 [s], Tmin = 100.0316, Tmax = 330.2707
Czas = 19.0000 [s], Tmin = 100.0457, Tmax = 335.7722
Czas = 20.0000 [s], Tmin = 100.0643, Tmax = 341.0847
PS C:\Users\Majkel\Desktop\STUDIA\SEMESTR_V\MES\projektMES>
```

Schemat 3-punktowy

```
Czas = 1.0000 [s], Tmin = 100.0000, Tmax = 149.5570
Czas = 2.0000 [s], Tmin = 100.0000, Tmax = 177.4449
Czas = 3.0000 [s], Tmin = 100.0000, Tmax = 197.2670
Czas = 4.0000 [s], Tmin = 100.0000, Tmax = 213.1528
Czas = 5.0000 [s], Tmin = 100.0000, Tmax = 226.6826
Czas = 6.0000 [s], Tmin = 100.0000, Tmax = 238.6071
Czas = 7.0000 [s], Tmin = 100.0000, Tmax = 249.3467
Czas = 8.0000 [s], Tmin = 100.0001, Tmax = 259.1651
Czas = 9.0000 [s], Tmin = 100.0002, Tmax = 268.2407
Czas = 10.0000 [s], Tmin = 100.0004, Tmax = 276.7011
Czas = 11.0000 [s], Tmin = 100.0008, Tmax = 284.6413
Czas = 12.0000 [s], Tmin = 100.0016, Tmax = 292.1342
Czas = 13.0000 [s], Tmin = 100.0029, Tmax = 299.2374
Czas = 14.0000 [s], Tmin = 100.0051, Tmax = 305.9971
Czas = 15.0000 [s], Tmin = 100.0086, Tmax = 312.4512
Czas = 16.0000 [s], Tmin = 100.0137, Tmax = 318.6312
Czas = 17.0000 [s], Tmin = 100.0212, Tmax = 324.5635
Czas = 18.0000 [s], Tmin = 100.0316, Tmax = 330.2707
Czas = 19.0000 [s], Tmin = 100.0457, Tmax = 335.7722
Czas = 20.0000 [s], Tmin = 100.0643, Tmax = 341.0847
```

Schemat 4-punktowy

```
Czas = 1.0000 [s], Tmin = 100.0000, Tmax = 149.5570
Czas = 2.0000 [s], Tmin = 100.0000, Tmax = 177.4449
Czas = 3.0000 [s], Tmin = 100.0000, Tmax = 197.2670
Czas = 4.0000 [s], Tmin = 100.0000, Tmax = 213.1528
Czas = 5.0000 [s], Tmin = 100.0000, Tmax = 226.6826
Czas = 6.0000 [s], Tmin = 100.0000, Tmax = 238.6071
Czas = 7.0000 [s], Tmin = 100.0000, Tmax = 249.3467
Czas = 8.0000 [s], Tmin = 100.0001, Tmax = 259.1651
Czas = 9.0000 [s], Tmin = 100.0002, Tmax = 268.2407
Czas = 10.0000 [s], Tmin = 100.0004, Tmax = 276.7011
Czas = 11.0000 [s], Tmin = 100.0008, Tmax = 284.6413
Czas = 12.0000 [s], Tmin = 100.0016, Tmax = 292.1342
Czas = 13.0000 [s], Tmin = 100.0029, Tmax = 299.2374
Czas = 14.0000 [s], Tmin = 100.0051, Tmax = 305.9971
Czas = 15.0000 [s], Tmin = 100.0086, Tmax = 312.4512
Czas = 16.0000 [s], Tmin = 100.0137, Tmax = 318.6312
Czas = 17.0000 [s], Tmin = 100.0212, Tmax = 324.5635
Czas = 18.0000 [s], Tmin = 100.0316, Tmax = 330.2707
Czas = 19.0000 [s], Tmin = 100.0457, Tmax = 335.7722
Czas = 20.0000 [s], Tmin = 100.0643, Tmax = 341.0847
PS C:\Users\Majkel\Desktop\STUDIA\SEMESTR_V\MES\projekt
```

Wyniki do porównania dostarczone przez prowadzącego

31x31:

99.99969812978378	**149.5566275788947**
100.00053467957446	**177.44482649738018**
100.00084733335379	**197.2672291500534**
100.00116712763896	**213.15348263983788**
100.00150209858216	**226.6837398631218**
100.001852708951	**238.60869878203812**
100.00222410506852	**249.34880985057373**
100.00263047992797	**259.1676797521773**
100.00310216686808	**268.24376548847937**
100.00369558647527	**276.70463950306436**
100.00450560745507	**284.64527660833346**
100.00567932588369	**292.1386492100023**
100.00742988613344	**299.242260871447**
100.01004886564658	**306.00237684844643**
100.01391592562979	**312.4568735346492**
100.01950481085419	**318.637221302136**
100.02738525124852	**324.56990275925733**
100.0382207726261	**330.27745133351596**
100.05276279329537	**335.77922748329735**
100.07184163487159	**341.0920092636545**

Rozkłady temperatur są zgodne z oczekiwanymi wynikami. Zmiana schematu całkowania ma 3-punktowy lub 4-punktowy dla siatek 4x4 i 4x4 mix daje minimalne zmiany w wynikach końcowych rzędu 1e-2. Co prowadzi do wniosku, że już 2-punktowy schemat całkowania daje bardzo zadowalające wyniki.

6. Wnioski

Implementacja została przeprowadzona poprawnie, wszystkie uzyskane rozkłady temperatur są zgodne z przewidywaniami. Już 2-punktowy schemat całkowania daje wyniki bardzo zadowalające. Zastosowanie metody elementów skończonych w implementacji programu do obliczania rozkładu temperatury w czasie okazało się przydatne, ponieważ umożliwia przybliżanie skomplikowanych obliczeń oraz wykonywanie złożonych symulacji inżynierskich. Daje to możliwość uniknięcia przeprowadzania kosztownych testów fizycznych i zastąpienie ich przeprowadzeniem analizy w środowisku symulacyjnym. Przedstawiony program stanowi przykład implementacji Metody Elementów Skończonych (MES), dla zagadnienia przewodzenia ciepła w 2D. Kod pozwala na zasymulowanie rozkładu temperatury w siatce o dowolnej wielkości i ilości elementów. Przeprowadzone testy potwierdzają poprawną implementacje.