Metoda Elementów Skończonych

Symulacja nieustalonego procesu
cieplnego oprogramowaniem MES

Kacper Drożdż Gr.2
Informatyka Techniczna

Wstep Teoretyczny:

Metoda Elementów Skończonych (MES) to metoda numeryczna stosowana do rozwiązywania równań różniczkowych w inżynierii, szczególnie w analizie przepływu ciepła, naprężeń i dynamiki płynów. Podstawową ideą MES jest przekształcenie ciągłych wartości w model dyskretny, poprzez podział badanego obiektu na siatkę MES składającą się z małych elementów skończonych połączonych węzłami.

Każdy element jest reprezentowany przez funkcje kształtu, które aproksymują wielkości fizyczne w jego obrębie. Kluczowym etapem analizy MES jest obliczanie Jakobianu, który przekształca współrzędne z lokalnych na globalne, co jest fundamentem budowy macierzy lokalnych.

Do obliczeń macierzy lokalnych stosuje się metody integracji numerycznej, w tym kwadraturę Gaussa, umożliwiającą przybliżenie wartości całek. Punkty i wagi Gaussa są dostosowywane do rodzaju elementów, co pozwala na precyzyjne odwzorowanie warunków brzegowych.

Celem sprawozdania jest analiza wyników symulacji ustalonych procesów cieplnych przeprowadzonych za pomocą MES, ze szczególnym uwzględnieniem wpływu parametrów numerycznych na dokładność i stabilność wyników.

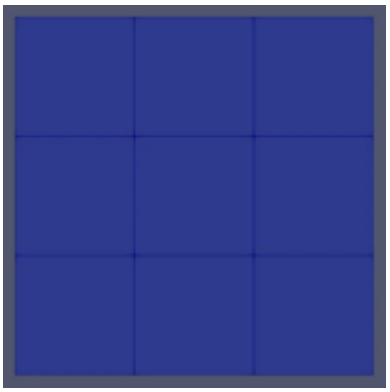
W kwestii rozwiązywanego problemu w oprogramowaniu, poza opracowaniem całkowania numerycznego metodą Gaussa, obliczany jest transport ciepła za pomocą równania FourieraKirchoffa w stanie nieustalonym z wykorzystaniem konwekcyjnego warunku brzegowego, który jest zadany wzorem:

$$q = \alpha(t - t_{amb})$$

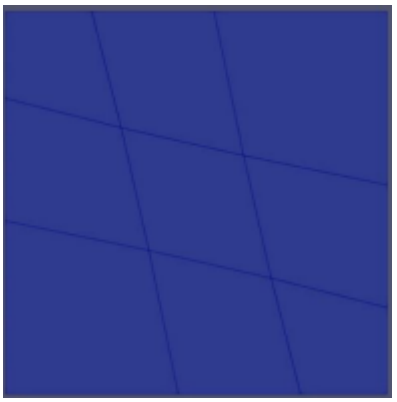
q – strumień ciepła, α – współczynnika przejmowania ciepła, t – temperatura powierzchni materiału, t_{amb} – temperatura otoczenia.

Konwekcyjny warunek brzegowy określa prędkość przepływu płynu na granicy obszaru, w którym rozważamy równanie różniczkowe.

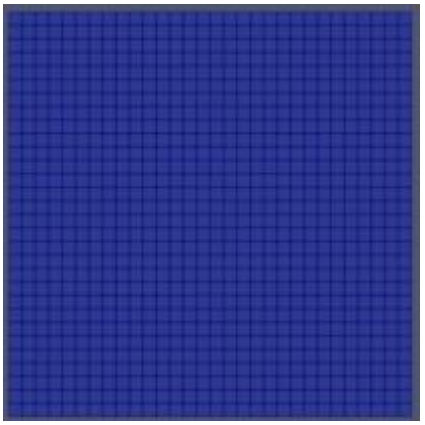
Dwuwymiarowe siatki MES używane w projekcie:



Rysunek 1 Siatka Test1_4x4



Rysunek 2 Siatka Test2_4x4_MixGrid



Rysunek 3 Siatka Test3_31x31_kwadrat

Charakterystyka kodu

```
struct Node {
    double x, y; // współrzędne węzła

    bool isBoundary; // do macierzy Hbc - czy jest na granicy

    Node(double x1 = 0.0, double y1 = 0.0) : x(x1), y(y1), isBoundary(false) {};
};
```

Struktura Node reprezentuje pojedynczy węzeł w siatce symulacyjnej. Jest to elementarny obiekt opisujący pozycję węzła w przestrzeni oraz jego charakterystykę jako punkt brzegowy, która będzie potrzebna do obliczania macierzy Hbc. Informacja o współrzędnych węzła jest kluczowa dla obliczeń numerycznych czy generowania macierzy lokalnych.

```
struct Element {
    int ID[4]; // Identyfikatory węzłów tworzących element
    double Hbc[4][4] = { 0 };
    double P[4] = { 0 };

    string wypisz() {
        string txt = to_string(ID[0]) + ", " + to_string(ID[1]) + ", " + to_string(ID[2]) + ", " + to_string(ID[3]);
        return txt;
    }
};
```

Struktura Element pełni rolę opisu pojedynczego elementu siatki w symulacji numerycznej. Zawiera następujące dane:

• ID[4]: tablica indeksów węzłów, które definiują element

• Hbc[4][4]: macierz lokalna związana z warunkami brzegowymi

• P[4]: wektor lokalny również związany z warunkami brzegowymi

Dodatkowo struktura zawiera funkcję wypisz(), która generuje tekstową reprezentację indeksów węzłów elementu, oddzielając je przecinkami. Funkcja ta ułatwia wizualizację lub debugowanie struktury danych.

```cpp
struct Grid {
    int nN;         // liczba węzłów
    int nE;         // liczba elementów
    Element* elements; // wskaźnik na dynamicznie alokowaną tablicę elementów
    Node* nodes;       // wskaźnik na dynamicznie alokowaną tablicę węzłów

    // Konstruktor
    Grid(int numNodes, int numElements) : nN(numNodes), nE(numElements) {
        elements = new Element[nE];  // alokacja elementów
        nodes = new Node[nN];        // alokacja węzłów
    }
    // Destruktor do zwalniania pamięci
    ~Grid() {
        delete[] elements;  // zwalnianie tablicy elementów
        delete[] nodes;     // zwalnianie tablicy węzłów
    }
};
```

Struktur Grid służy do reprezentacji siatki numerycznej. Zawiera informacje o węzłach i elementach siatki, które są dynamicznie alokowane w pamięci.

```cpp
struct GlobalData {
    int SimulationTime;
    int SimulationStepTime;
    int Conductivity;
    int Alfa;
    int Tot;
    int InitialTemp;
    int Density;
    int SpecificHeat;
    int nN;  // liczba węzłów
    int nE;  // liczba elementów
```

Struktura GlobalData pełni funkcję zbiorczego magazynu danych wejściowych, które sterują przebiegiem symulacji. Dzięki niej program może automatycznie dostosować się do różnych warunków materiałowych i symulacyjnych

```
void wspolrzedne(GlobalData* dane, Grid* siatka) {
    double x_start = 0.1;
    double x = x_start;
    double y = 0.005;
    double step = 0.1 / sqrt(dane->nE); // Krok zależny od liczby elementów
    int licz = 0;
    for (int i = 0; i < sqrt(dane->nN); i++) {
        for (int j = 0; j < sqrt(dane->nN); j++) {
            siatka->nodes[licz].x = x;
            siatka->nodes[licz].y = y;

            x -= step;  // Zmniejszamy współrzędną x
            licz++;
        }
        x = x_start;  // Resetujemy x po każdym wierszu
        y -= step;  // Zmniejszamy współrzędną y
    }
}
```

Aby stworzyć siatkę wywołujemy powyższa funkcje „wspolrzedne” do obliczania współrzędnych co robiliśmy na pierwszych zajęciach. Działa ona tylko dla siatek kwadratowych. Drugą opcja jest funkcja **wczytajWspolrzedne** która wczytuje współrzędne z pliku. Jest ona potrzebna do robienia obliczeń na siatce 4x4_MixGrid.

```
void elementy(GlobalData* dane, Grid* siatka) {
    int licz = 1;
    for (int i = 0; i < dane->nE; i++) {
        siatka->elements[i].ID[0] = licz;
        siatka->elements[i].ID[1] = licz + 1;
        siatka->elements[i].ID[2] = licz + sqrt(dane->nN) + 1;
        siatka->elements[i].ID[3] = licz + sqrt(dane->nN);
        licz++;
        if (licz % int(sqrt(dane->nN)) == 0) licz++;  // Przejście do nowego wiersza
    }
}
```

Funkcja elementy przypisuje węzły do poszczególnych elementów siatki obliczeniowej. Działa na podstawie struktury siatki (Grid) oraz danych globalnych (GlobalData)

Konkretne wartości punktów całkowania oraz wagi zależą od schematu całkowania zgodnie z tabelą kwadratur Gaussa-Legendre'a:

Number of points, n	Points, x_i	Weights, w_i
1	0	2
2	$\pm\sqrt{\frac{1}{3}}$	1
3	0	$\frac{8}{9}$
	$\pm\sqrt{\frac{3}{5}}$	$\frac{5}{9}$
4	$\pm\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{18+\sqrt{30}}{36}$
	$\pm\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{18-\sqrt{30}}{36}$
5	0	$\frac{128}{225}$
	$\pm\frac{1}{3}\sqrt{5-2\sqrt{\frac{10}{7}}}$	$\frac{322+13\sqrt{70}}{900}$
	$\pm\frac{1}{3}\sqrt{5+2\sqrt{\frac{10}{7}}}$	$\frac{322-13\sqrt{70}}{900}$

```
if (choice == 1) { // Dwupunktowy
    numPoints = 4;
    points[0][0] = -1 / sqrt(3); points[0][1] = -1 / sqrt(3);
    points[1][0] = 1 / sqrt(3); points[1][1] = -1 / sqrt(3);
    points[2][0] = 1 / sqrt(3); points[2][1] = 1 / sqrt(3);
    points[3][0] = -1 / sqrt(3); points[3][1] = 1 / sqrt(3);
```

```
else if (choice == 2) { // Trójpunktowy;
    double values[3] = { -sqrt(3.0 / 5.0), 0.0, sqrt(3.0 / 5.0) };
    double weights[3] = { 5.0 / 9.0, 8.0 / 9.0, 5.0 / 9.0 };
    numPoints = 9;
    int idx = 0;

    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            points[idx][0] = values[i];
            points[idx][1] = values[j];
            w[idx] = weights[i] * weights[j];
            idx++;
        }
    }
```

```
else if (choice == 3) { // Czteropunktowy
    numPoints = 16;
    // Współrzędne i wagi punktów Gaussa
    double roots[4] = { -sqrt(1.0 / 3.0), -sqrt(3.0 / 7.0), sqrt(3.0 / 7.0), sqrt(1.0 / 3.0) };
    double weights[4] = { (18.0 - sqrt(30)) / 36.0, (18.0 + sqrt(30)) / 36.0, (18.0 + sqrt(30)) / 36.0, (18.0 - sqrt(30)) / 36.0 };

    int idx = 0;
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            points[idx][0] = roots[i];
            points[idx][1] = roots[j];
            w[idx] = weights[i] * weights[j];
            idx++;
        }
    }
```

Powyższe kody implementują różne warianty kwadratur Gaussa w zależności od wartości zmiennej choice. Kod obsługuje następujące przypadki:

1. **Dwupunktowy** (n=2) – definiuje współrzędne punktów, posiada wagi równe 1 i przypisuje liczbę punktów jako 4.

2. **Trójpunktowy** (n=3) – definiuje współrzędne punktów i odpowiadających wag, generując 9 punktów w siatce.

3. **Czteropunktowy** (n=4) – definiuje współrzędne punktów i odpowiadających wag, generując 16 punktów.

Kod używa pętli zagnieżdżonych do inicjalizacji współrzędnych punktów i wag zgodnie z regułami Gaussa.

Wzory na funkcję kształtu w kolejności od dolnej krawędzi elementu:

$$N_1 = 0{,}25\,(1-\xi)(1-\eta)$$
$$N_2 = 0{,}25\,(1+\xi)(1-\eta)$$
$$N_3 = 0{,}25\,(1+\xi)(1+\eta)$$
$$N_4 = 0{,}25\,(1-\xi)(1+\eta)$$

Macierz przekształcenia Jacobiego dla danego elementu:

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

```
for (int j = 0; j < numPoints; j++) {
    double E = points[j][0];
    double n = points[j][1];
```

```
Matrix2x2 J = computeJacobian(x1, y1, x2, y2, x3, y3, x4, y4, E, n);
```

```
Matrix2x2 computeJacobian(double x1, double y1, double x2, double y2, double x3, double y3, double x4, double y4, double E, double n) {
    double N1_E = -0.25 * (1 - n);
    double N2_E = 0.25 * (1 - n);
    double N3_E = 0.25 * (1 + n);
    double N4_E = -0.25 * (1 + n);

    double N1_n = -0.25 * (1 - E);
    double N2_n = -0.25 * (1 + E);
    double N3_n = 0.25 * (1 + E);
    double N4_n = 0.25 * (1 - E);

    double dx_dE = (N1_E * x1) + (N2_E * x2) + (N3_E * x3) + (N4_E * x4);
    double dx_dn = (N1_n * x1) + (N2_n * x2) + (N3_n * x3) + (N4_n * x4);
    double dy_dE = (N1_E * y1) + (N2_E * y2) + (N3_E * y3) + (N4_E * y4);
    double dy_dn = (N1_n * y1) + (N2_n * y2) + (N3_n * y3) + (N4_n * y4);

    return { dx_dE, dy_dE, dx_dn, dy_dn };
}
```

Na podstawie obliczonej macierzy przekształcenia dla danego elementu obliczamy Wyznacznik Jakobianu oraz Odwrotność Jakobianu.

```
double determinant() {
    return a * d - b * c;
}

Matrix2x2 inverse() {
    double det = determinant();
    if (det == 0) {
        cerr << "Macierz nieodwracalna!" << endl;
        return { 0, 0, 0, 0 };
    }
    return { d / det, -b / det, -c / det, a / det };
}
```

Po obliczeniu wyznacznika tej macierzy możemy wyznaczyć pochodne do wektorów $\left\{\frac{\partial\{N\}}{\partial x}\right\}, \left\{\frac{\partial\{N\}}{\partial y}\right\}$ do wzoru na [H] za pomocą wzoru:

$$\begin{bmatrix} \frac{\partial\{N_i\}}{\partial x} \\ \frac{\partial\{N_i\}}{\partial y} \end{bmatrix} = \frac{1}{detJ} \begin{bmatrix} \frac{\partial y}{\partial \eta} & -\frac{\partial y}{\partial \xi} \\ -\frac{\partial x}{\partial \eta} & \frac{\partial x}{\partial \xi} \end{bmatrix} \begin{bmatrix} \frac{\partial\{N_i\}}{\partial \xi} \\ \frac{\partial\{N_i\}}{\partial \eta} \end{bmatrix}$$

```
for (int k = 0; k < 4; k++) {
    dN_dx[k] = invJ.a * dN_dE[k] + invJ.b * dN_dn[k];
    dN_dy[k] = invJ.c * dN_dE[k] + invJ.d * dN_dn[k];
}
```

Następnie po otrzymaniu danych wektorów można obliczyć macierz H jeszcze bez uwzględnienia części z warunkiem brzegowym. Macierz dla danego elementu uzyskujemy sumując cząstkowe macierze i mnożąc je przez iloczyn wag pod konkretny punkt całkowania.

Przykład dla 2-pkt schematu całkowania:

$$\mathrm{H} = H_{pc1} * w_1 * w_1 + H_{pc2} * w_2 * w_1 + H_{pc3} * w_1 * w_2 + H_{pc4} * w_2 * w_2$$

```
for (int m = 0; m < 4; m++) {
    for (int n = 0; n < 4; n++) {
        H_local[j][m][n] = (dN_dx[m] * dN_dx[n] + dN_dy[m] * dN_dy[n]) * w[j] * dane.Conductivity * detJ;
        H[i][m][n] += H_local[j][m][n];
    }
}
```

Następnym krokiem była agregacja macierzy lokalnej H do macierzy globalnej. Funkcja agregujH będzie później używana do agregacji macierzy Hbc oraz C.

```
void agregujH(Grid* siatka, vector<vector<double>> &H_global, vector<vector<vector<double>>> &H){
    for (int i = 0; i < siatka->nN; i++) {
        for (int j = 0; j < siatka->nN; j++) {
            H_global[i][j] = 0.0;
        }
    }

    for (int e = 0; e < siatka->nE; e++) {
        int* ID = siatka->elements[e].ID;
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                H_global[ID[i] - 1][ID[j] - 1] += H[e][i][j];
            }
        }
    }
}
```

Na węzłach znajdujących się na brzegach każdej z siatek mamy do czynienia z warunkiem brzegowym, jest to miejsce, gdzie energia wchodzi w układ, w tym przypadku cieplna. Dla elementów, których krawędzie mają warunek brzegowy (na obu węzłach krawędzi mamy warunek brzegowy) w celu analizy wpływu konwekcji oraz temperatury otoczenia liczymy wektory P oraz Macierze HBC.

Wektor P – to wektor obciążeń (strumieni cieplnych) opisujący wpływ temperatury otoczenia na węzły elementu siatki.

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

t_{ot} – temperatura otoczenia

Macierz HBC – to uzupełnienie [H] o część warunku brzegowego konwekcji. [HBC] opisuje transport ciepła, które wnika przez ściany objęte warunkiem brzegowym konwekcji.

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

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

k(t) – współczynnik przewodności cieplnej

α – współczynnik wymiany energii cieplnej materiału

{N} – wektor wartości funkcji kształtu

Sprawdzanie warunków brzegowych:

```
void Czy_isBoundary(Grid& grid) {
    double minX = grid.nodes[0].x, maxX = grid.nodes[0].x;
    double minY = grid.nodes[0].y, maxY = grid.nodes[0].y;

    for (int i = 1; i < grid.nN; i++) {
        if (grid.nodes[i].x < minX) minX = grid.nodes[i].x;
        if (grid.nodes[i].x > maxX) maxX = grid.nodes[i].x;
        if (grid.nodes[i].y < minY) minY = grid.nodes[i].y;
        if (grid.nodes[i].y > maxY) maxY = grid.nodes[i].y;
    }

    for (int i = 0; i < grid.nN; i++) {
        if (grid.nodes[i].x == minX || grid.nodes[i].x == maxX || grid.nodes[i].y == minY || grid.nodes[i].y == maxY) {
            grid.nodes[i].isBoundary = true;
        }
    }
}
```

W celu policzenia dS w układzie lokalnym obliczono jakobian przekształcenia ze wzoru:

$$detJ = \frac{\sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}}{2}$$

```
double detJ_boku = sqrt(pow(node2.x - node1.x, 2) + pow(node2.y - node1.y, 2)) / 2.0;
```

Obliczanie macierzy Hbc lokalnej i wektora P lokalnego:

```
for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 4; j++) {
        Hbc[i][j] += alfa * N[i] * N[j] * weights[pc] * detJ_boku;
    }
    P[i] += alfa * tot * N[i] * weights[pc] * detJ_boku;
}
```

Obliczanie macierzy C ze wzoru:

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

c – ciepło właściwe, ρ - gęstość materiału

```
for (int m = 0; m < 4; m++) {
    for (int n = 0; n < 4; n++) {
        C_local[e][m][n] += N[m] * N[n] * density * specificHeat * detJ * weight;
    }
}
```

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

Δ τ – krok czasowy

{$t0$} – temperatura początkowo w kroku 1, w kolejnych temperatura dla poprzedniego Δ τ

{$t1$} – temperatura obliczona

[C] – macierz opisująca ilość możliwej do zgromadzenia energii cieplnej przez węzeł siatki.

W celu wyznaczenia macierzy [C] wykorzystujemy takie same jak do macierzy [H] kwadratury Gaussa-Legendre'a, punkty całkowania, wyznacznik macierzy przekształcenia. Następnie macierze C dla danych punktów całkowania mnożymy przez iloczyn wag dla danych punktów całkowania, następnie dla danego elementu zsumowano. Otrzymane macierze C zagregowano w ten sam sposób jak macierz H, z czego powstała globalna macierz C. Zgodnie z wcześniej podanym równaniem macierz C podzielono przez krok czasowy. W celu wykonania symulacji czasowej, wymagane jest obliczenie {t1} dla każdego kroku czasowego, w tym celu należy rozwiązać układ równań w postaci Ax+B = 0, który rozwiązujemy za pomocą metody eliminacji Gaussa.

Składowe układu równań:

$$\mathrm{A}=[H]+\frac{C}{\Delta\tau}\ ,\ \mathrm{B}=-(\frac{C}{\Delta\tau}\{t_0\}+\{P\})$$

$$\mathrm{x}=\{t_1\}$$

```
for (int step = 1; step <= nSteps; step++) {
    double currentTime = step * dt;
    cout << "Krok czasowy: " << step << ", czas = " << currentTime << endl;
    // A = C_global/dt + H_global
    for (int i = 0; i < dane.nN; i++) {
        for (int j = 0; j < dane.nN; j++) {
            A[i][j] = Hbc_global[i][j] + (C_global[i][j] / dt);
        }
    }
    // b = (C_global/dt)*T(n) + P_global
    // b[i] = sum( C_global[i][j]/dt * T[j] ) + P_global[i]
    for (int i = 0; i < dane.nN; i++) {
        double sumCT = 0.0;
        for (int j = 0; j < dane.nN; j++) {
            sumCT += (C_global[i][j] / dt) * T[j];
        }
        b[i] = sumCT + P_global[i];
    }
    // uklad A * T(n+1) = b - rozwiazujemy metoda eliminacji Gaussa
    for (int k = 0; k < dane.nN; k++) {
        T[k] = 0;
    }
```

```cpp
void eliminacjaGaussa(Grid* siatka, vector<vector<double>> &A, vector<double> &b, vector<double> &x) {

    // Eliminacja w przód
    for (int k = 0; k < siatka->nN; k++) {
        // Normalizacja elementu głównego
        double pivot = A[k][k];
        for (int j = k; j < siatka->nN; j++) {
            A[k][j] /= pivot;
        }
        b[k] /= pivot;

        // Zerowanie elementów poniżej
        for (int i = k + 1; i < siatka->nN; i++) {
            double factor = A[i][k];
            for (int j = k; j < siatka->nN; j++) {
                A[i][j] -= factor * A[k][j];
            }
            b[i] -= factor * b[k];
        }
    }
    // Podstawianie wstecz
    for (int i = siatka->nN - 1; i >= 0; i--) {
        x[i] = b[i];
        for (int j = i + 1; j < siatka->nN; j++) {
            x[i] -= A[i][j] * x[j];
        }
    }
}
```

Porównanie Wyników

Z pliku test_niestacjonarny.pdf:

Martix [H]

_____ Interation 0 _____
_____ Matrix [H] _____

16.6667 -4.16667 0 0 -4.16667 -8.33333 0 0 0 0 0 0 0 0 0 0
-4.16667 33.3333 -4.16667 0 -8.33333 -8.33333 -8.33333 0 0 0 0 0 0 0 0 0
0 -4.16667 33.3333 -4.16667 0 -8.33333 -8.33333 -8.33333 0 0 0 0 0 0 0 0
0 0 -4.16667 16.6667 0 0 -8.33333 -4.16667 0 0 0 0 0 0 0 0
-4.16667 -8.33333 0 0 33.3333 -8.33333 0 0 -4.16667 -8.33333 0 0 0 0 0 0
-8.33333 -8.33333 -8.33333 0 -8.33333 66.6667 -8.33333 0 -8.33333 -8.33333 -8.33333 0 0 0 0 0
0 -8.33333 -8.33333 -8.33333 0 -8.33333 66.6667 -8.33333 0 -8.33333 -8.33333 -8.33333 0 0 0 0
0 0 -8.33333 -4.16667 0 0 -8.33333 33.3333 0 0 -8.33333 -4.16667 0 0 0 0
0 0 0 0 -4.16667 -8.33333 0 0 33.3333 -8.33333 0 0 -4.16667 -8.33333 0 0
0 0 0 0 -8.33333 -8.33333 -8.33333 0 -8.33333 66.6667 -8.33333 0 -8.33333 -8.33333 -8.33333 0
0 0 0 0 0 -8.33333 -8.33333 -8.33333 0 -8.33333 66.6667 -8.33333 0 -8.33333 -8.33333 -8.33333
0 0 0 0 0 0 -8.33333 -4.16667 0 0 -8.33333 33.3333 0 0 -8.33333 -4.16667
0 0 0 0 0 0 0 0 -4.16667 -8.33333 0 0 16.6667 -4.16667 0 0
0 0 0 0 0 0 0 0 -8.33333 -8.33333 -8.33333 0 -4.16667 33.3333 -4.16667 0
0 0 0 0 0 0 0 0 0 -8.33333 -8.33333 -8.33333 0 -4.16667 33.3333 -4.16667
0 0 0 0 0 0 0 0 0 0 -8.33333 -4.16667 0 0 -4.16667 16.6667

```
Macierz globalna H:
16.667  -4.167  0.000  0.000  -4.167  -8.333  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
-4.167  33.333  -4.167  0.000  -8.333  -8.333  -8.333  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
0.000  -4.167  33.333  -4.167  0.000  -8.333  -8.333  -8.333  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
0.000  0.000  -4.167  16.667  0.000  0.000  -8.333  -4.167  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000
-4.167  -8.333  0.000  0.000  33.333  -8.333  0.000  0.000  -4.167  -8.333  0.000  0.000  0.000  0.000  0.000  0.000
-8.333  -8.333  -8.333  0.000  -8.333  66.667  -8.333  0.000  -8.333  -8.333  -8.333  0.000  0.000  0.000  0.000  0.000
0.000  -8.333  -8.333  -8.333  0.000  -8.333  66.667  -8.333  0.000  -8.333  -8.333  -8.333  0.000  0.000  0.000  0.000
0.000  0.000  -8.333  -4.167  0.000  0.000  -8.333  33.333  0.000  0.000  -8.333  -4.167  0.000  0.000  0.000  0.000
0.000  0.000  0.000  0.000  -4.167  -8.333  0.000  0.000  33.333  -8.333  0.000  0.000  -4.167  -8.333  0.000  0.000
0.000  0.000  0.000  0.000  -8.333  -8.333  -8.333  0.000  -8.333  66.667  -8.333  0.000  -8.333  -8.333  -8.333  0.000
0.000  0.000  0.000  0.000  0.000  -8.333  -8.333  -8.333  0.000  -8.333  66.667  -8.333  0.000  -8.333  -8.333  -8.333
0.000  0.000  0.000  0.000  0.000  0.000  -8.333  -4.167  0.000  0.000  -8.333  33.333  0.000  0.000  -8.333  -4.167
0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  -4.167  -8.333  0.000  0.000  16.667  -4.167  0.000  0.000
0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  -8.333  -8.333  -8.333  0.000  -4.167  33.333  -4.167  0.000
0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  -8.333  -8.333  -8.333  0.000  -4.167  33.333  -4.167
0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  -8.333  -4.167  0.000  0.000  -4.167  16.667
```

Martix [C]

_____ Interation 0 _____
_____Matrix [C]_____
674.074 337.037 0 0 337.037 168.519 0 0 0 0 0 0 0 0 0 0
337.037 1348.15 337.037 0 168.519 674.074 168.519 0 0 0 0 0 0 0 0 0
0 337.037 1348.15 337.037 0 168.519 674.074 168.519 0 0 0 0 0 0 0 0
0 0 337.037 674.074 0 0 168.519 337.037 0 0 0 0 0 0 0 0
337.037 168.519 0 0 1348.15 674.074 0 0 337.037 168.519 0 0 0 0 0 0
168.519 674.074 168.519 0 674.074 2696.3 674.074 0 168.519 674.074 168.519 0 0 0 0 0
0 168.519 674.074 168.519 0 674.074 2696.3 674.074 0 168.519 674.074 168.519 0 0 0 0
0 0 168.519 337.037 0 0 674.074 1348.15 0 0 168.519 337.037 0 0 0 0
0 0 0 0 337.037 168.519 0 0 1348.15 674.074 0 0 337.037 168.519 0 0
0 0 0 0 168.519 674.074 168.519 0 674.074 2696.3 674.074 0 168.519 674.074 168.519 0
0 0 0 0 0 168.519 674.074 168.519 0 674.074 2696.3 674.074 0 168.519 674.074 168.519
0 0 0 0 0 0 168.519 337.037 0 0 674.074 1348.15 0 0 168.519 337.037
0 0 0 0 0 0 0 0 337.037 168.519 0 0 674.074 337.037 0 0
0 0 0 0 0 0 0 0 168.519 674.074 168.519 0 337.037 1348.15 337.037 0
0 0 0 0 0 0 0 0 0 168.519 674.074 168.519 0 337.037 1348.15 337.037
0 0 0 0 0 0 0 0 0 0 168.519 337.037 0 0 337.037 674.074

```
Macierz C globalna:
674.074 337.037 0.000 0.000 337.037 168.519 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
337.037 1348.148 337.037 0.000 168.519 674.074 168.519 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 337.037 1348.148 337.037 0.000 168.519 674.074 168.519 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 337.037 674.074 0.000 0.000 168.519 337.037 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
337.037 168.519 0.000 0.000 1348.148 674.074 0.000 0.000 337.037 168.519 0.000 0.000 0.000 0.000 0.000 0.000
168.519 674.074 168.519 0.000 674.074 2696.296 674.074 0.000 168.519 674.074 168.519 0.000 0.000 0.000 0.000 0.000
0.000 168.519 674.074 168.519 0.000 674.074 2696.296 674.074 0.000 168.519 674.074 168.519 0.000 0.000 0.000 0.000
0.000 0.000 168.519 337.037 0.000 0.000 674.074 1348.148 0.000 0.000 168.519 337.037 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 337.037 168.519 0.000 0.000 1348.148 674.074 0.000 0.000 337.037 168.519 0.000 0.000
0.000 0.000 0.000 0.000 168.519 674.074 168.519 0.000 674.074 2696.296 674.074 0.000 168.519 674.074 168.519 0.000
0.000 0.000 0.000 0.000 0.000 168.519 674.074 168.519 0.000 674.074 2696.296 674.074 0.000 168.519 674.074 168.519
0.000 0.000 0.000 0.000 0.000 0.000 168.519 337.037 0.000 0.000 674.074 1348.148 0.000 0.000 168.519 337.037
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 337.037 168.519 0.000 0.000 674.074 337.037 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 168.519 674.074 168.519 0.000 337.037 1348.148 337.037 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 168.519 674.074 168.519 0.000 337.037 1348.148 337.037
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 168.519 337.037 0.000 0.000 337.037 674.074
```

Martix [H] = [H]+[C]/dT and {P} = {P}+{[C]/dT}*{T0}

_____ Interation 0 _____
_____ Matrix ([H]+[C]/dT)_____
36.8148 4.24074 0 0 4.24074 -4.96296 0 0 0 0 0 0 0 0 0 0
4.24074 66.963 4.24074 0 -4.96296 5.14815 -4.96296 0 0 0 0 0 0 0 0 0
0 4.24074 66.963 4.24074 0 -4.96296 5.14815 -4.96296 0 0 0 0 0 0 0 0
0 0 4.24074 36.8148 0 0 -4.96296 4.24074 0 0 0 0 0 0 0 0
4.24074 -4.96296 0 0 66.963 5.14815 0 0 4.24074 -4.96296 0 0 0 0 0 0
-4.96296 5.14815 -4.96296 0 5.14815 120.593 5.14815 0 -4.96296 5.14815 -4.96296 0 0 0 0 0
0 -4.96296 5.14815 -4.96296 0 5.14815 120.593 5.14815 0 -4.96296 5.14815 -4.96296 0 0 0 0
0 0 -4.96296 4.24074 0 0 5.14815 66.963 0 0 -4.96296 4.24074 0 0 0 0
0 0 0 0 4.24074 -4.96296 0 0 66.963 5.14815 0 0 4.24074 -4.96296 0 0
0 0 0 0 -4.96296 5.14815 -4.96296 0 5.14815 120.593 5.14815 0 -4.96296 5.14815 -4.96296 0
0 0 0 0 0 -4.96296 5.14815 -4.96296 0 5.14815 120.593 5.14815 0 -4.96296 5.14815 -4.96296
0 0 0 0 0 0 -4.96296 4.24074 0 0 5.14815 66.963 0 0 -4.96296 4.24074
0 0 0 0 0 0 0 0 4.24074 -4.96296 0 0 36.8148 4.24074 0 0
0 0 0 0 0 0 0 0 -4.96296 5.14815 -4.96296 0 4.24074 66.963 4.24074 0
0 0 0 0 0 0 0 0 0 -4.96296 5.14815 -4.96296 0 4.24074 66.963 4.24074
0 0 0 0 0 0 0 0 0 0 -4.96296 4.24074 0 0 4.24074 36.8148
_____ Vector ([{P}+{[C]/dT}*{T0})_____
15033.3 18066.7 18066.7 15033.3 18066.7 12133.3 12133.3 18066.7 18066.7 12133.3 12133.3 18066.7 15033.3 18066.7 18066.7 15033.350 110.04 365.82

```
36.815 4.241 0.000 0.000 4.241 -4.963 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
4.241 66.963 4.241 0.000 -4.963 5.148 -4.963 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 4.241 66.963 4.241 0.000 -4.963 5.148 -4.963 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 4.241 36.815 0.000 0.000 -4.963 4.241 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
4.241 -4.963 0.000 0.000 66.963 5.148 0.000 0.000 4.241 -4.963 0.000 0.000 0.000 0.000 0.000 0.000
-4.963 5.148 -4.963 0.000 5.148 120.593 5.148 0.000 -4.963 5.148 -4.963 0.000 0.000 0.000 0.000 0.000
0.000 -4.963 5.148 -4.963 0.000 5.148 120.593 5.148 0.000 -4.963 5.148 -4.963 0.000 0.000 0.000 0.000
0.000 0.000 -4.963 4.241 0.000 0.000 5.148 66.963 0.000 0.000 -4.963 4.241 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 4.241 -4.963 0.000 0.000 66.963 5.148 0.000 0.000 4.241 -4.963 0.000 0.000
0.000 0.000 0.000 0.000 -4.963 5.148 -4.963 0.000 5.148 120.593 5.148 0.000 -4.963 5.148 -4.963 0.000
0.000 0.000 0.000 0.000 0.000 -4.963 5.148 -4.963 0.000 5.148 120.593 5.148 0.000 -4.963 5.148 -4.963
0.000 0.000 0.000 0.000 0.000 0.000 -4.963 4.241 0.000 0.000 5.148 66.963 0.000 0.000 -4.963 4.241
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 4.241 -4.963 0.000 0.000 36.815 4.241 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 -4.963 5.148 -4.963 0.000 4.241 66.963 4.241 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 -4.963 5.148 -4.963 0.000 4.241 66.963 4.241
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 -4.963 4.241 0.000 0.000 4.241 36.815
15033.333 18066.667 18066.667 15033.333 18066.667 12133.333 12133.333 18066.667 18066.667 12133.333 12133.333 18066.667 15033.333 18066.667 18066.667 15033.333
```

_____ Interation 1 _____
_____H Matrix ([H]+[C]/dT)_____
36.815 4.2407 0 0 4.2407 -4.963 0 0 0 0 0 0 0 0 0 0
4.2407 66.963 4.2407 0 -4.963 5.1481 -4.963 0 0 0 0 0 0 0 0 0
0 4.2407 66.963 4.2407 0 -4.963 5.1481 -4.963 0 0 0 0 0 0 0 0
0 0 4.2407 36.815 0 0 -4.963 4.2407 0 0 0 0 0 0 0 0
4.2407 -4.963 0 0 66.963 5.1481 0 0 4.2407 -4.963 0 0 0 0 0 0
-4.963 5.1481 -4.963 0 5.1481 120.59 5.1481 0 -4.963 5.1481 -4.963 0 0 0 0 0
0 -4.963 5.1481 -4.963 0 5.1481 120.59 5.1481 0 -4.963 5.1481 -4.963 0 0 0 0
0 0 -4.963 4.2407 0 0 5.1481 66.963 0 0 -4.963 4.2407 0 0 0 0
0 0 0 0 4.2407 -4.963 0 0 66.963 5.1481 0 0 4.2407 -4.963 0 0
0 0 0 0 -4.963 5.1481 -4.963 0 5.1481 120.59 5.1481 0 -4.963 5.1481 -4.963 0
0 0 0 0 0 -4.963 5.1481 -4.963 0 5.1481 120.59 5.1481 0 -4.963 5.1481 -4.963
0 0 0 0 0 0 -4.963 4.2407 0 0 5.1481 66.963 0 0 -4.963 4.2407
0 0 0 0 0 0 0 0 4.2407 -4.963 0 0 36.815 4.2407 0 0
0 0 0 0 0 0 0 0 -4.963 5.1481 -4.963 0 4.2407 66.963 4.2407 0
0 0 0 0 0 0 0 0 0 -4.963 5.1481 -4.963 0 4.2407 66.963 4.2407
0 0 0 0 0 0 0 0 0 0 -4.963 4.2407 0 0 4.2407 36.815
_____P_Vector ([{P}+{[C]/dT}*{T0})_____
20660 25552 25552 20660 25552 18897 18897 25552 25552 18897 18897 25552 20660 25552 25552 20660

```
36.815 4.241 0.000 0.000 4.241 -4.963 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
4.241 66.963 4.241 0.000 -4.963 5.148 -4.963 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 4.241 66.963 4.241 0.000 -4.963 5.148 -4.963 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 4.241 36.815 0.000 0.000 -4.963 4.241 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
4.241 -4.963 0.000 0.000 66.963 5.148 0.000 0.000 4.241 -4.963 0.000 0.000 0.000 0.000 0.000 0.000
-4.963 5.148 -4.963 0.000 5.148 120.593 5.148 0.000 -4.963 5.148 -4.963 0.000 0.000 0.000 0.000 0.000
0.000 -4.963 5.148 -4.963 0.000 5.148 120.593 5.148 0.000 -4.963 5.148 -4.963 0.000 0.000 0.000 0.000
0.000 0.000 -4.963 4.241 0.000 0.000 5.148 66.963 0.000 0.000 -4.963 4.241 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 4.241 -4.963 0.000 0.000 66.963 5.148 0.000 0.000 4.241 -4.963 0.000 0.000
0.000 0.000 0.000 0.000 -4.963 5.148 -4.963 0.000 5.148 120.593 5.148 0.000 -4.963 5.148 -4.963 0.000
0.000 0.000 0.000 0.000 0.000 -4.963 5.148 -4.963 0.000 5.148 120.593 5.148 0.000 -4.963 5.148 -4.963
0.000 0.000 0.000 0.000 0.000 0.000 -4.963 4.241 0.000 0.000 5.148 66.963 0.000 0.000 -4.963 4.241
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 4.241 -4.963 0.000 0.000 36.815 4.241 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 -4.963 5.148 -4.963 0.000 4.241 66.963 4.241 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 -4.963 5.148 -4.963 0.000 4.241 66.963 4.241
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 -4.963 4.241 0.000 0.000 4.241 36.815
20659.699 25552.224 25552.224 20659.699 25552.224 18897.391 18897.391 25552.224 25552.224 18897.391 18897.391 25552.224 20659.699 25552.224 25552.224 20659.699
```

Wyniki dla pliku Wyniki_Temperatur_w_czasie_dla_wszystkich_siatek.txt

```
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
```

```
Krok czasowy: 1, czas = 50.000
Min T = 110.0379762758, Max T = 365.8154683351
Krok czasowy: 2, czas = 100.0000000000
Min T = 168.8370162918, Max T = 502.5917112218
Krok czasowy: 3, czas = 150.0000000000
Min T = 242.8008536324, Max T = 587.3726650239
Krok czasowy: 4, czas = 200.0000000000
Min T = 318.6145959364, Max T = 649.3874813299
Krok czasowy: 5, czas = 250.0000000000
Min T = 391.2557985002, Max T = 700.0684178779
Krok czasowy: 6, czas = 300.0000000000
Min T = 459.0369149964, Max T = 744.0633412641
Krok czasowy: 7, czas = 350.0000000000
Min T = 521.5862908442, Max T = 783.3828460481
Krok czasowy: 8, czas = 400.0000000000
Min T = 579.0344662587, Max T = 818.9921833030
Krok czasowy: 9, czas = 450.0000000000
Min T = 631.6892625741, Max T = 851.4310374576
Krok czasowy: 10, czas = 500.0000000000
Min T = 679.9076230023, Max T = 881.0576290016
```

```
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
```

```
Krok czasowy: 1, czas = 50.000
Min T = 95.1518489972, Max T = 374.6863331884
Krok czasowy: 2, czas = 100.0000000000
Min T = 147.6444190737, Max T = 505.9681112789
Krok czasowy: 3, czas = 150.0000000000
Min T = 220.1644557510, Max T = 586.9978492792
Krok czasowy: 4, czas = 200.0000000000
Min T = 296.7364383530, Max T = 647.2855822357
Krok czasowy: 5, czas = 250.0000000000
Min T = 370.9682724189, Max T = 697.3339844638
Krok czasowy: 6, czas = 300.0000000000
Min T = 440.5601435998, Max T = 741.2191101273
Krok czasowy: 7, czas = 350.0000000000
Min T = 504.8912021091, Max T = 781.2095685594
Krok czasowy: 8, czas = 400.0000000000
Min T = 564.0015163225, Max T = 817.3915046241
Krok czasowy: 9, czas = 450.0000000000
Min T = 618.1738632501, Max T = 850.2373167651
Krok czasowy: 10, czas = 500.0000000000
Min T = 667.7655569117, Max T = 880.1676019293
```

```
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
```

```
Krok czasowy: 1, czas = 1.000
Min T = 100.0000000003, Max T = 149.5569482171
Krok czasowy: 2, czas = 2.0000000000
Min T = 100.0000000053, Max T = 177.4449259525
Krok czasowy: 3, czas = 3.0000000000
Min T = 100.0000000515, Max T = 197.2669621530
Krok czasowy: 4, czas = 4.0000000000
Min T = 100.0000003344, Max T = 213.1527871314
Krok czasowy: 5, czas = 5.0000000000
Min T = 100.0000016382, Max T = 226.6825834204
Krok czasowy: 6, czas = 6.0000000000
Min T = 100.0000064712, Max T = 238.6070646252
Krok czasowy: 7, czas = 7.0000000000
Min T = 100.0000215294, Max T = 249.3466916743
Krok czasowy: 8, czas = 8.0000000000
Min T = 100.0000622127, Max T = 259.1650788506
Krok czasowy: 9, czas = 9.0000000000
Min T = 100.0001597834, Max T = 268.2406886921
Krok czasowy: 10, czas = 10.0000000000
Min T = 100.0003713449, Max T = 276.7010975571
Krok czasowy: 11, czas = 11.0000000000
Min T = 100.0007922355, Max T = 284.6412828968
Krok czasowy: 12, czas = 12.0000000000
Min T = 100.0015698460, Max T = 292.1342187770
Krok czasowy: 13, czas = 13.0000000000
Min T = 100.0029174837, Max T = 299.2374096905
Krok czasowy: 14, czas = 14.0000000000
Min T = 100.0051267974, Max T = 305.9971212918
Krok czasowy: 15, czas = 15.0000000000
Min T = 100.0085774634, Max T = 312.4512299961
Krok czasowy: 16, czas = 16.0000000000
Min T = 100.0137432139, Max T = 318.6312059362
Krok czasowy: 17, czas = 17.0000000000
Min T = 100.0211937592, Max T = 324.5635313055
Krok czasowy: 18, czas = 18.0000000000
Min T = 100.0315926133, Max T = 330.2707390035
Krok czasowy: 19, czas = 19.0000000000
Min T = 100.0456911999, Max T = 335.7721888914
Krok czasowy: 20, czas = 20.0000000000
Min T = 100.0643198684, Max T = 341.0846583898
```

Wnioski

1. Optymalny schemat całkowania:

Wykorzystanie schematu całkowania trzypunktowego jest najbardziej odpowiednie, zapewniając wysoką dokładność wyników przy akceptowalnym czasie obliczeń. W analizowanych przypadkach zastosowanie schematu czteropunktowego nie przyniosło znaczących korzyści, a jedynie zwiększyło czas obliczeń.

2. Wpływ parametrów siatki:

Przy rzadszych siatkach konieczne jest stosowanie bardziej precyzyjnych schematów całkowania, aby zminimalizować błędy wynikające z dyskretyzacji. Gęstsze siatki, jak w przypadku testu 3 (siatka 31x31), poprawiają stabilność wyników nawet przy prostszych schematach całkowania, jednak znacząco wydłużają czas wykonania symulacji.

3. Zgodność wyników:

Wyniki uzyskane z symulacji przeprowadzonej za pomocą metody elementów skończonych są zgodne z danymi referencyjnymi dostarczonymi przez prowadzącego, co potwierdza poprawność implementacji i działania programu.

4. Zalety i ograniczenia metody elementów skończonych:

MES umożliwia podział dużego i złożonego problemu na mniejsze i łatwiejsze do rozwiązania zadania. Dzięki temu uzyskujemy bardziej precyzyjne wyniki. Wadą jest jednak znaczny wzrost czasu obliczeń, szczególnie w przypadku gęstych siatek i większej liczby punktów całkowania.

Wysoka złożoność obliczeniowa metody eliminacji Gaussa oraz wielokrotne operacje macierzowe powodują, że dla bardziej złożonych siatek czas wykonania symulacji jest znacznie wydłużony.