Wyznaczanie nieoznaczonego pola temperatury we wsadzie o przekroju okrągłym

Autor: Michał Góra

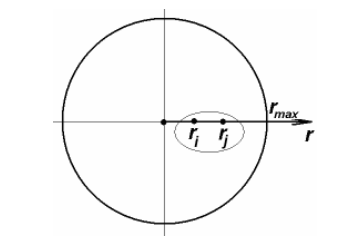
Grupa: 1

Kierunek: IS

Wydział: WIMiIP

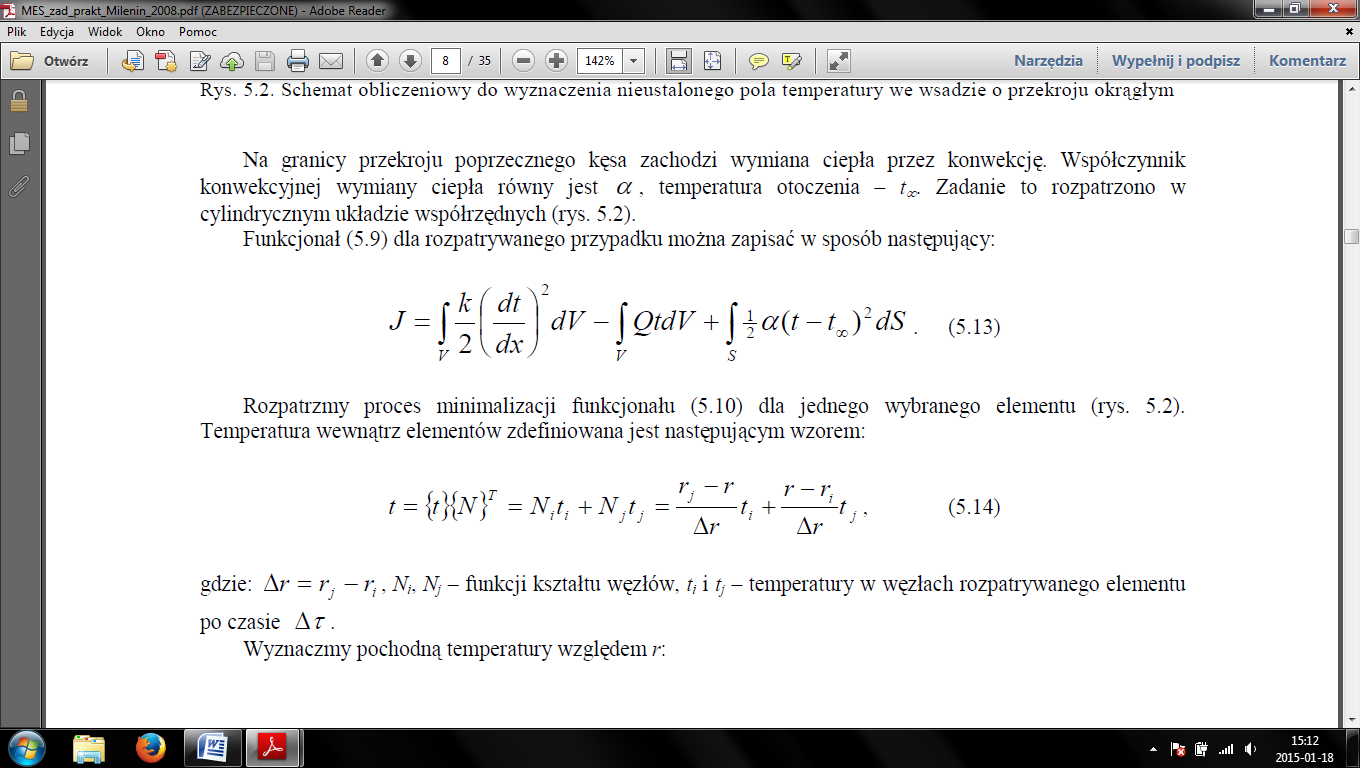
1. **Wstęp teoretyczny**

Zadaniem było stworzenie programu komputerowego obliczającego rozkład temperatur we wsadzie o przekroju okrągłym. Na granicy przekroju poprzecznego kęsa zachodzi wymiana ciepła przez konwekcję. Współczynnik konwekcyjnej wymiany ciepła jest równy α, natomiast temperatura otoczenia – t∞. Zadanie to rozpatrzono w cylindrycznym układzie współrzędnych:

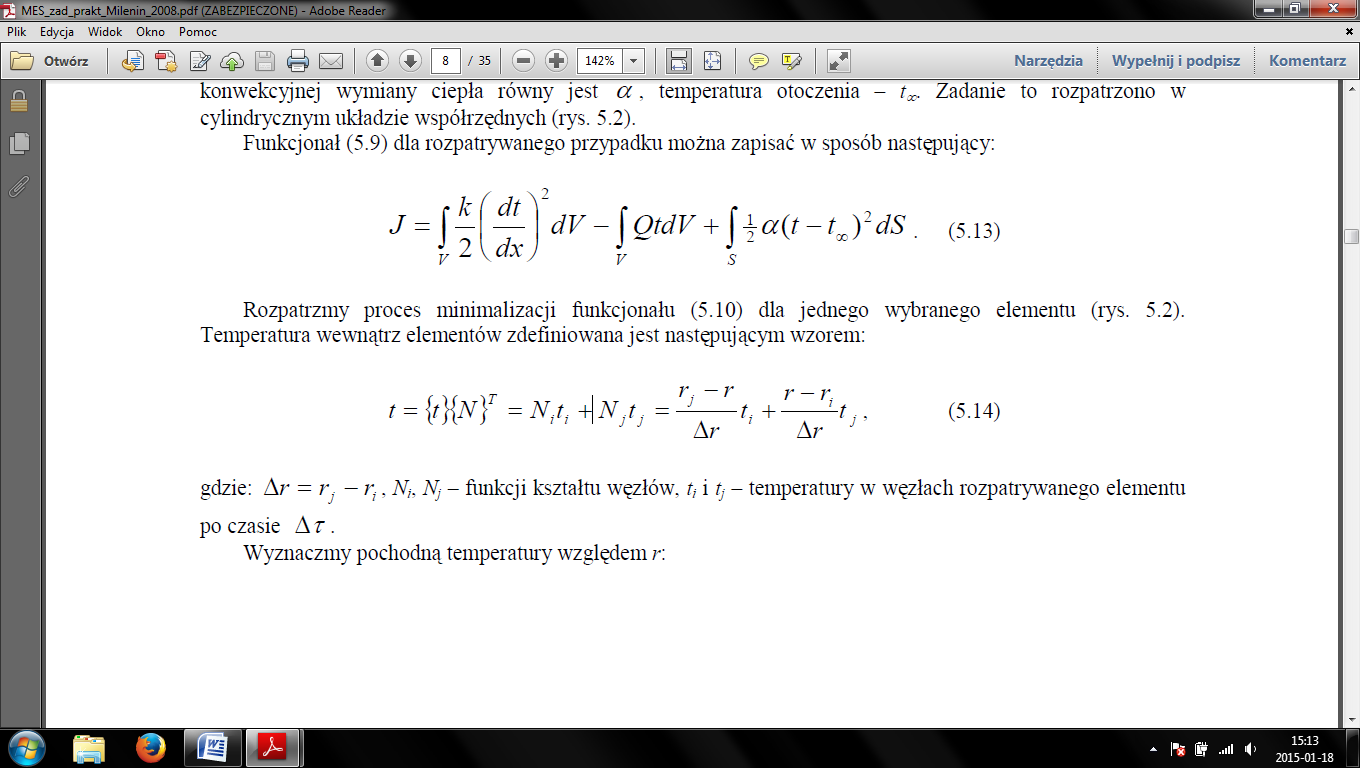


Powyższy rysunek przedstawia schemat obliczeniowy do wyznaczania nieustalonego pola temperatury we wsadzie o przekroju okrągłym.

Zapis funkcjonału dla rozpatrywanego przypadku:

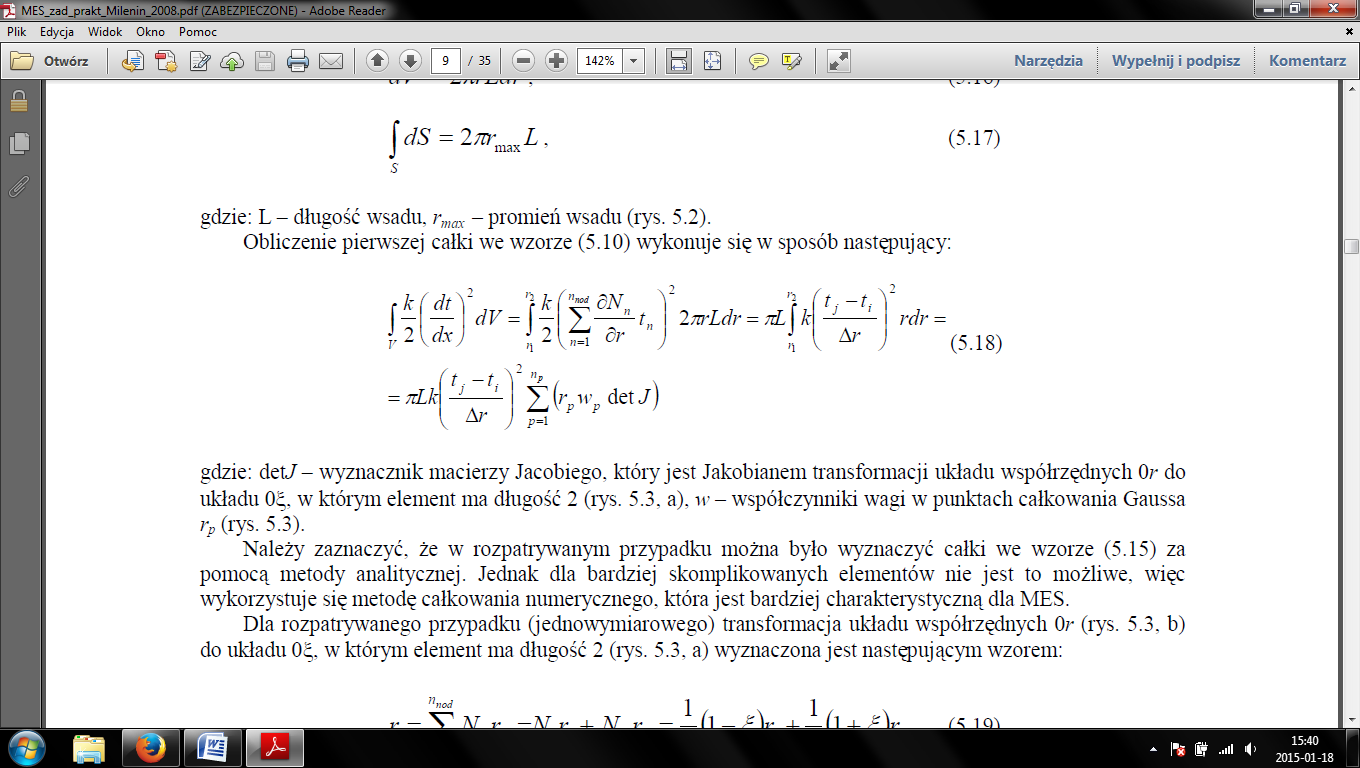
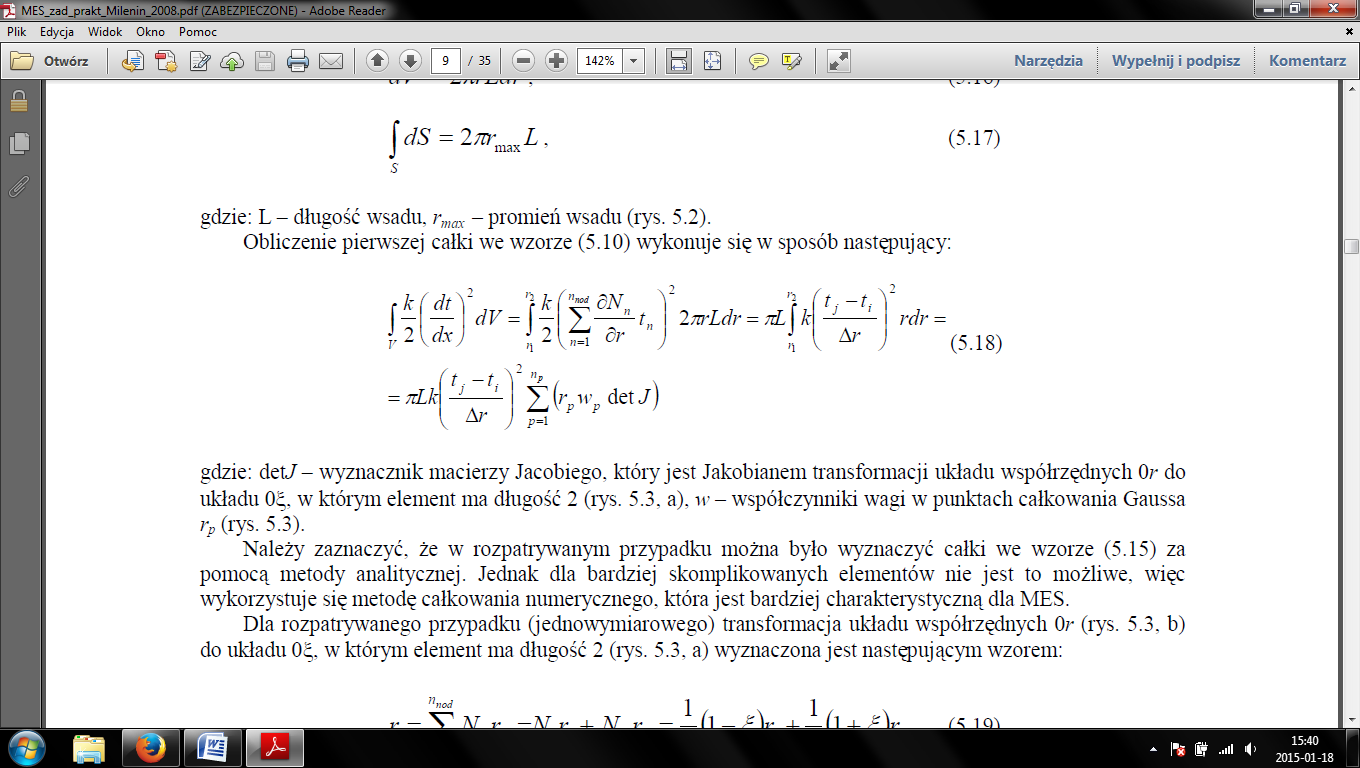


Temperatura wewnątrz elementów jest zdefiniowana następująco:

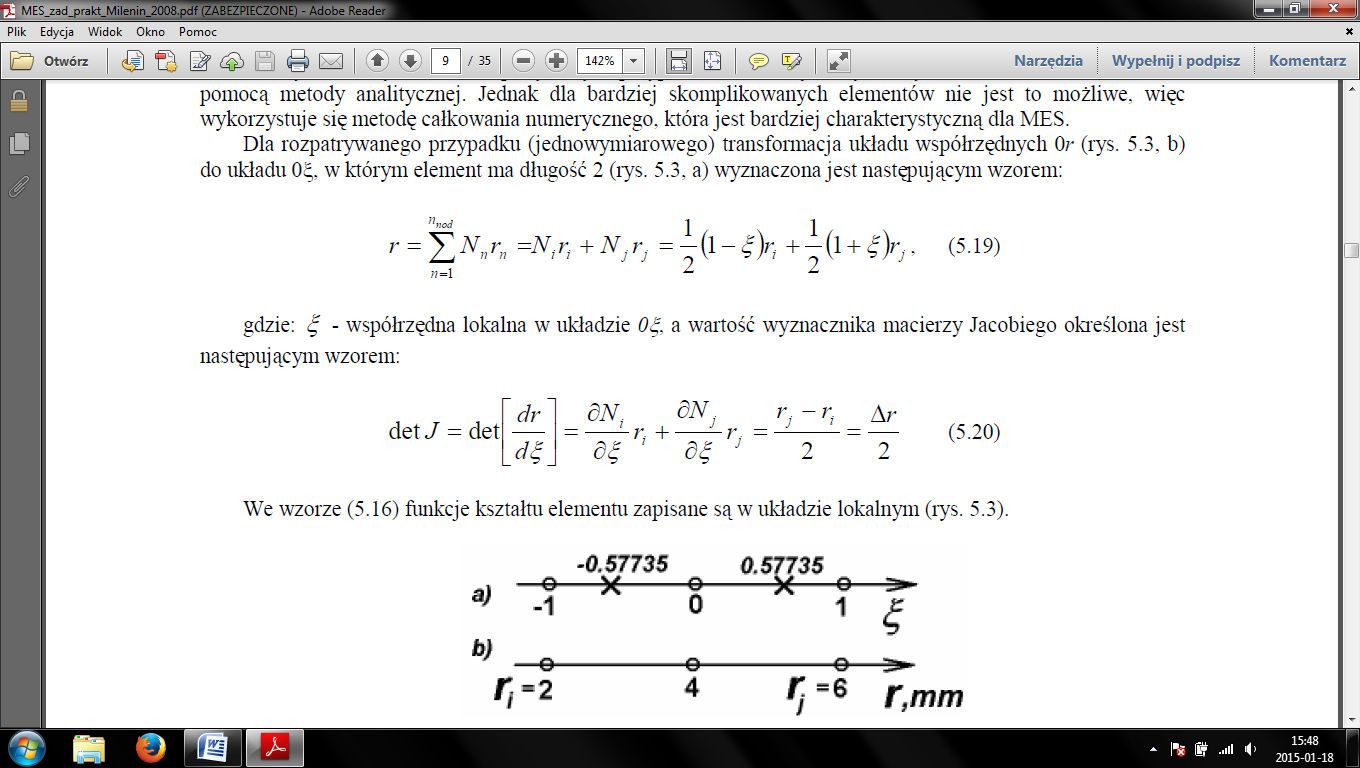


gdzie: ∆r = rj – ri , Ni, Nj – funkcji kształtu węzłów, ti i tj – temperatury w węzłach rozpatrywanego elementu po czasie ∆τ .

Wzór pierwszej całki, której wynikiem jest pochodna temperatur względem współrzędnych.

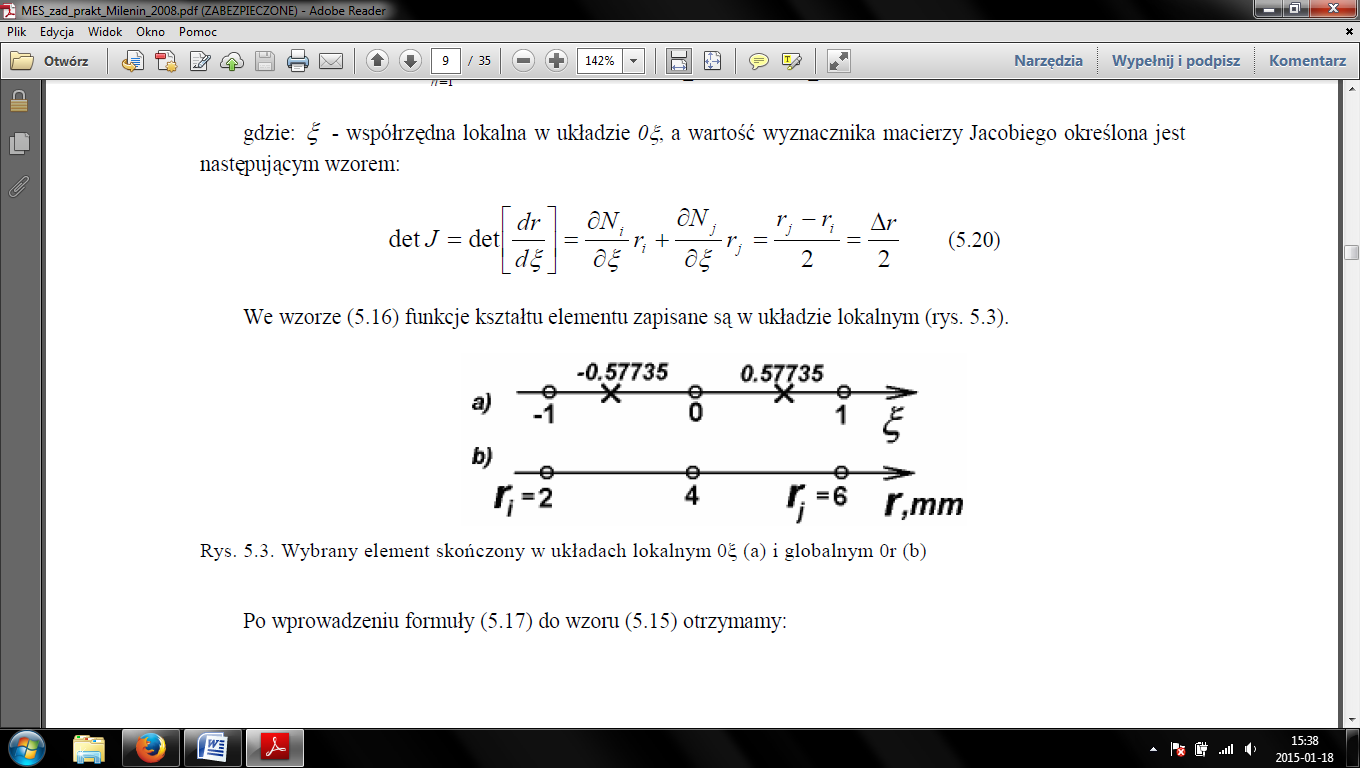


Dla danego przykładu transformacja z układu lokalnego w globalny następuje następującym wzorem:



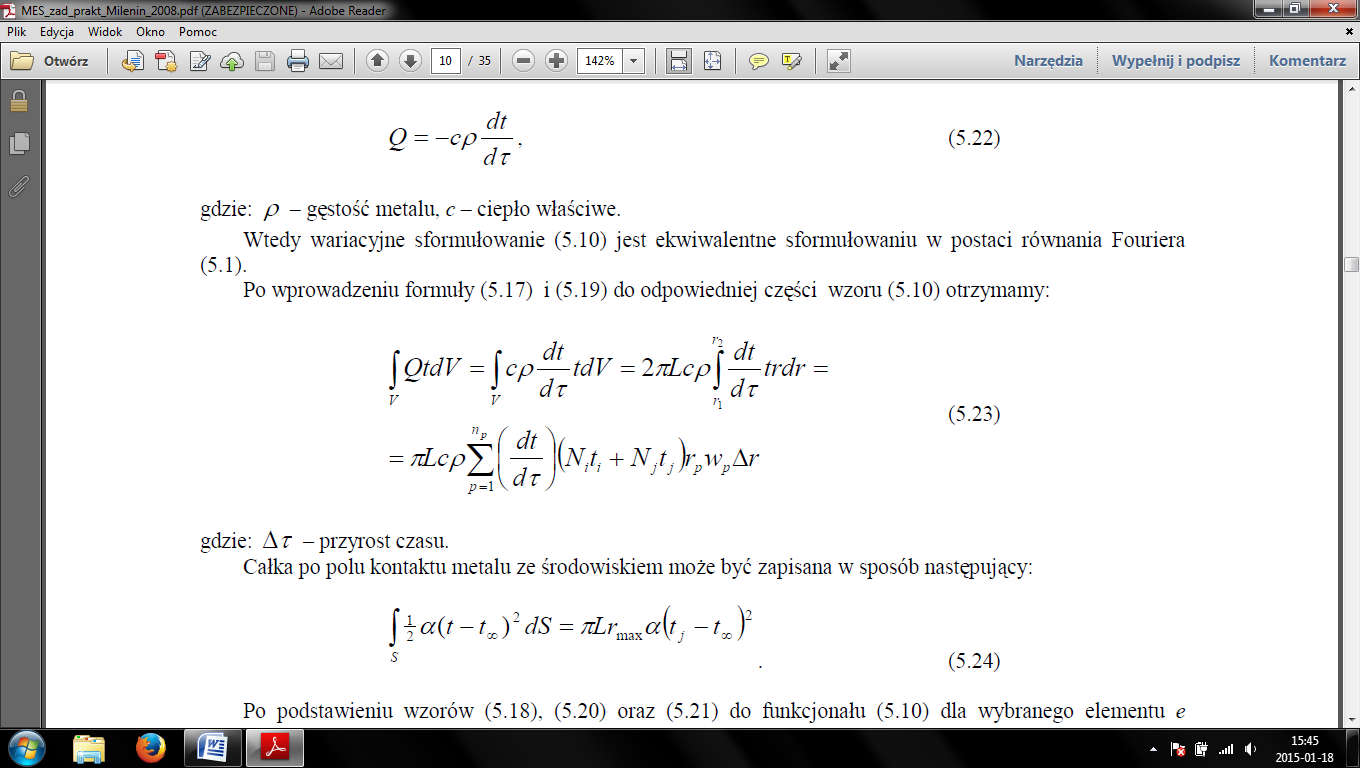
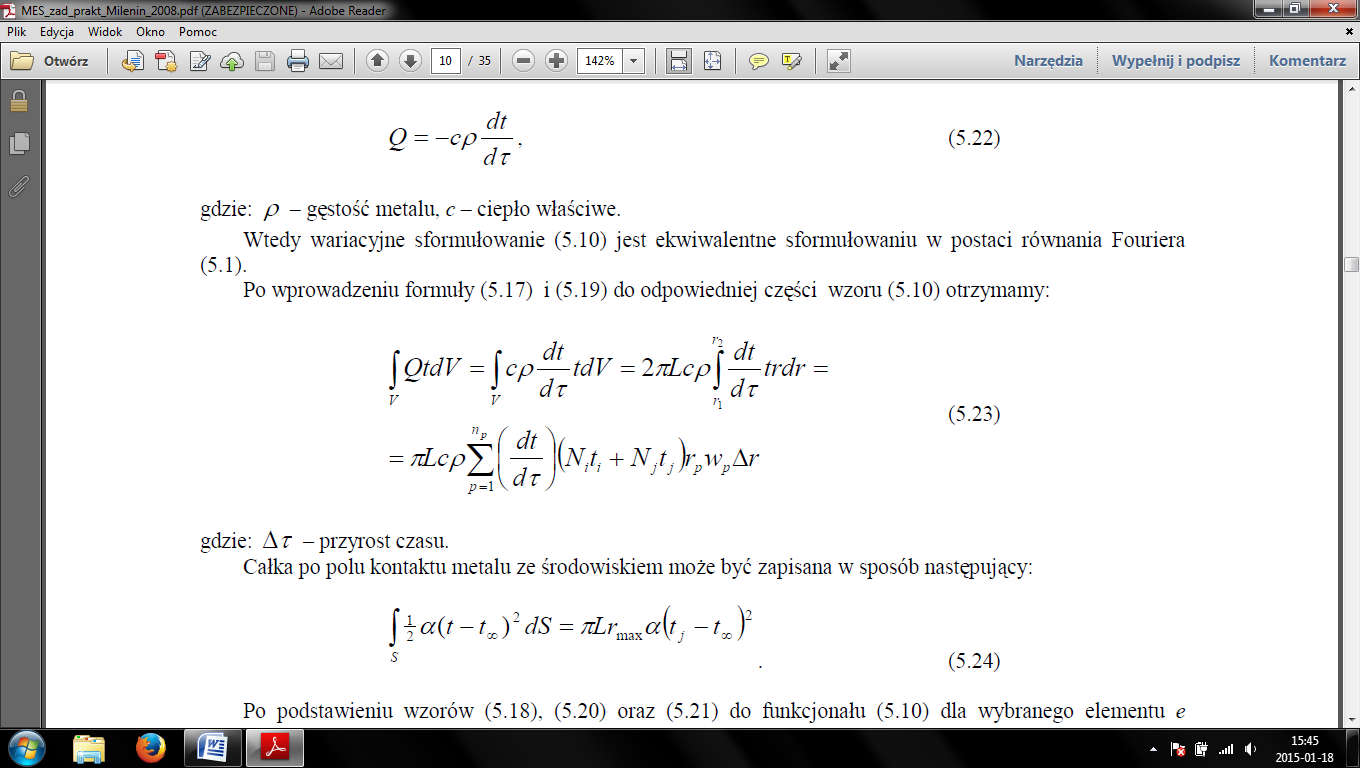
Wyznacznik Jacobiego wyznacza pochodna układu globalnego przez lokalny:



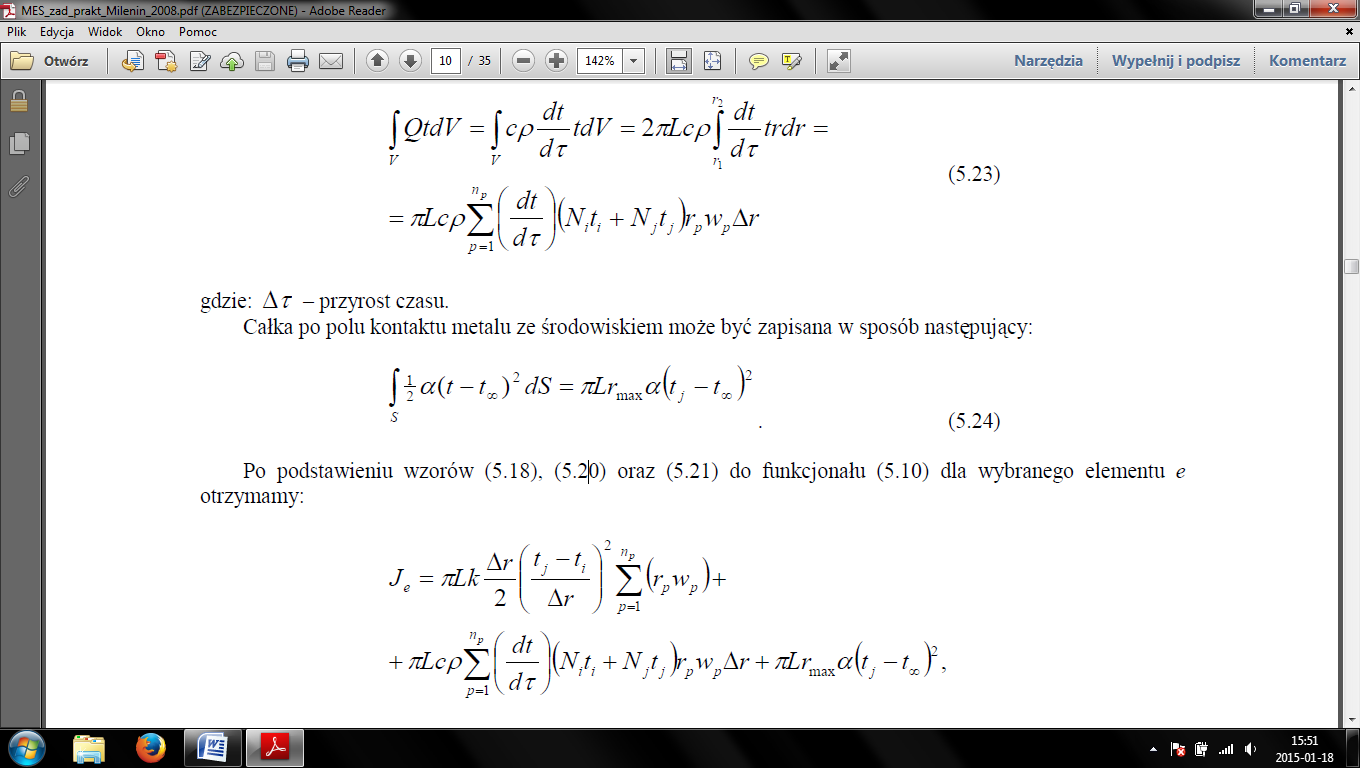


Wybrany element skończony w układach lokalnym 0ξ (a) i globalnym 0r (b)

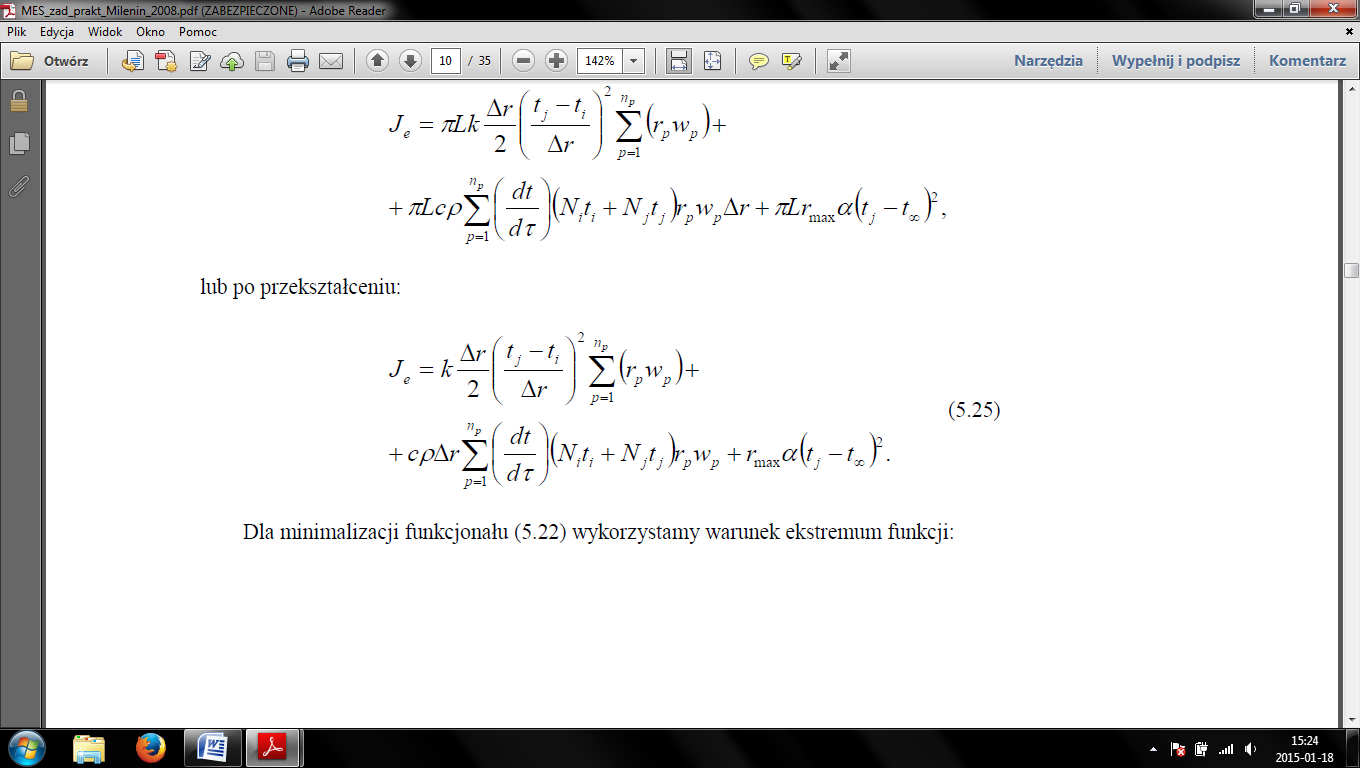
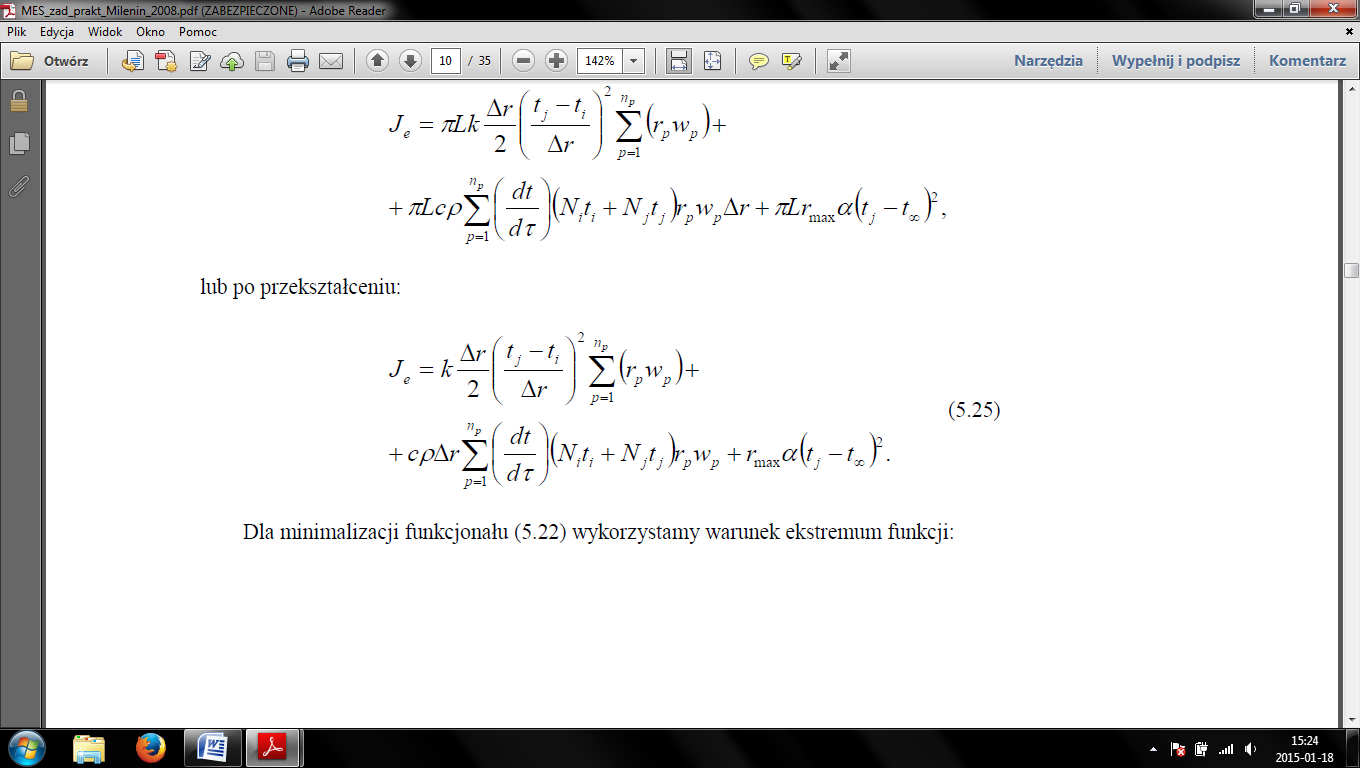
Wzór drugiej całki, której wynikiem jest niestacjonarność procesu zmiany temperatury.



Całka, której wynikiem jest wymiana temperatury z otoczeniem.



**Po rozwiązaniu całek otrzymano funkcjonał:**

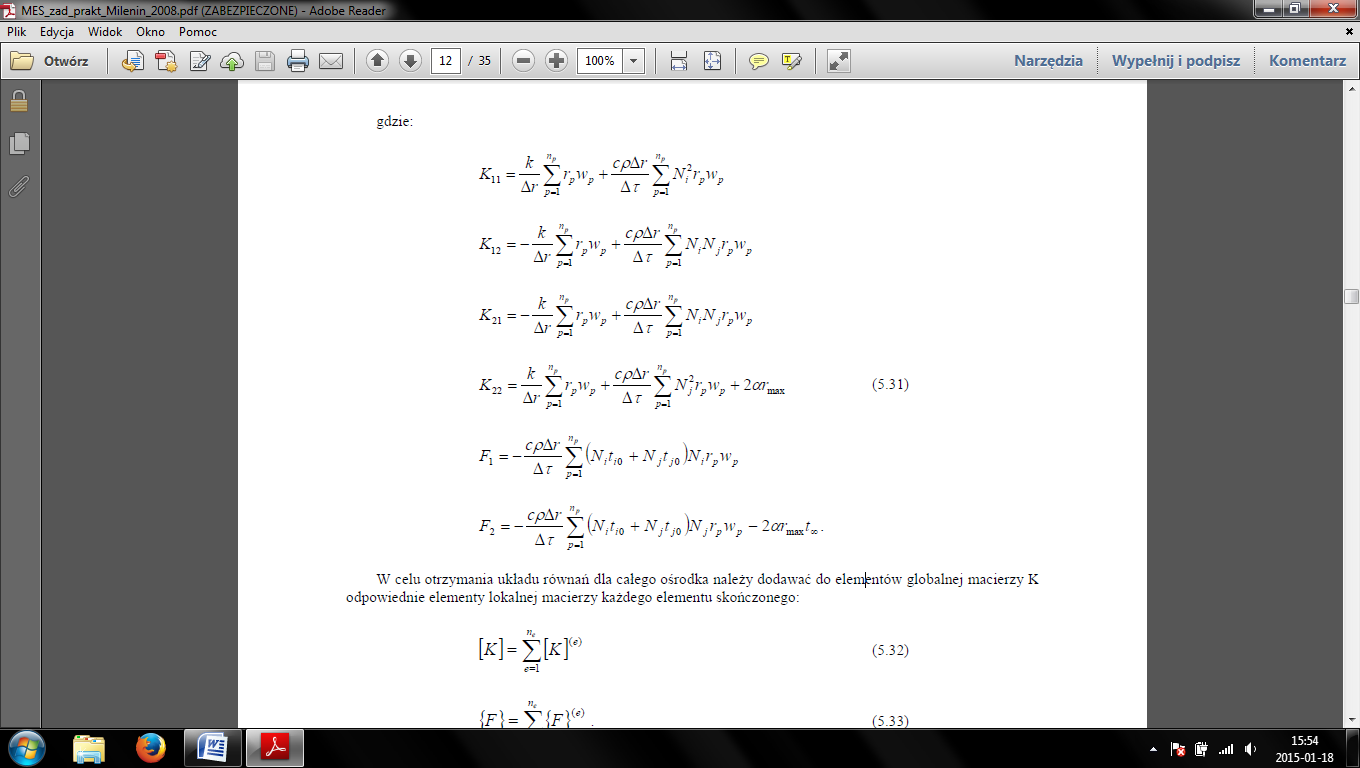


gdzie: Δr - wartośc wyznacznika macierzy Jacobiego, wp- współczynnik wagi w punktach całkowania Gaussa, rp – promień w punkcie całkownia, c – pojemność cieplna, p – gęstość.

Równanie w postaci macierzowej:

**[K] \* {t} + {F} = 0**

gdzie: K – macierz sztywności, F – wektor obciążeń, t – szukany wektor temperatur



1. **Implementacja w c++**

Program liczy temperatury dla dowolnej liczby elementów. Z macierzy lokalnych zostaje utworzona macierz sztywności. Do utworzonej macierzy globalnej, dodawany jest do niej wektor obciążeń i dzięki nim zostają wyliczone temperatury. Obliczenia zależne są od kroku czasowego. Budowanie macierzy globalnej odbywa się w ten sam sposób, jak w programie do obliczania ustalonego pola temperatury w pręcie. Jeżeli węzły nie są w odpowiedniej kolejności, macierz globalna tworzona jest w sposób zależny od numerów kolejnych węzłów.

#include<iostream>

#include<cmath>

#include<iomanip>

using namespace std;

const double eps = 1e-12;

class element

{

public:

int nop1;

int nop2;

double HL[2][2]; //macierz lokalna

double PL[2]; //wektor lokalny

};

class siatka

{

public:

void dane()

{

Rmin = 0.01; //promien minimalny

Rmax = 0.05; //promien maksymalny

AlfaAir = 300; //wspolczynnik konwekcyjnej wymiany ciepla

TempBegin = 100; //temperatura poczatkowa

TempAir = 300; //temperatura srodowiskowa

TauMax = 50; //czas procesu

C = 700; //efektywne cieplo wlasciwe

K = 25; //wspolczynnik przewodzenia ciepla

Ro = 7800; //gestosc materialu

}

double Rmin;

double Rmax;

double AlfaAir;

double TempBegin;

double TempAir;

double TauMax;

double C;

double K;

double Ro;

double Ksi;

double N[4]; //funkcje kształtow

double T[2];

int nn;

int ne;

siatka()

{

nn = 9; //liczba wezlow

ne = nn - 1; //liczba elementow

}

};

void macierz\_lokalna(element \*tab, siatka data)

{

//numeracja poszczegonych wezlow

for(int i = 0; i < data.ne; i++)

{

tab[i].nop1 = i;

tab[i].nop2 = i + 1;

}

double pom = 2\*data.AlfaAir \* data.Rmax; //ze wzorow

double pom2 = 2\*data.AlfaAir \* data.Rmax \* data.TempAir;

double ri = 0;

data.Ksi = 1 / sqrt(double(3));

data.N[0] = 0.5 \* (1 - data.Ksi);

data.N[1] = 0.5 \* (1 + data.Ksi);

data.N[2] = 0.5 \* (1 + data.Ksi);

data.N[3] = 0.5 \* (1 - data.Ksi);

for(int i = 0; i < data.ne; i++)

{

double rj = ri + data.Rmin;

double wi = 1.0;

double wj = 1.0;

double rp1, rp2;

//liczona wartosc punktu calkowania w ukladzie globalnym

rp1 = data.N[0] \* ri + data.N[1] \* rj;

rp2 = data.N[1] \* ri + data.N[0] \* rj;

//liczenie lokalnej macierzy

tab[i].HL[0][0] = (data.K / data.Rmin) \* ((rp1 \* wi) + (rp2 \* wj)) + ((data.C \* data.Ro \* data.Rmin) / data.TauMax) \* ((pow(data.N[0],2.0) \* (rp1 \* wi)) + (pow(data.N[1],2.0) \* (rp2\*wj)));

tab[i].HL[0][1] = -(data.K / data.Rmin) \* ((rp1 \* wi) + (rp2 \* wj)) + ((data.C \* data.Ro \* data.Rmin) / data.TauMax) \* (data.N[0] \* data.N[1] \* (rp1+rp2));

tab[i].HL[1][0] = -(data.K / data.Rmin) \* ((rp1 \* wi) + (rp2 \* wj)) + ((data.C \* data.Ro \* data.Rmin) / data.TauMax) \* (data.N[0] \* data.N[1] \* (rp1+rp2));

tab[i].HL[1][1] = (data.K / data.Rmin) \* ((rp1 \* wi) + (rp2 \* wj)) + ((data.C \* data.Ro \* data.Rmin) / data.TauMax) \* ((pow(data.N[1],2.0) \* (rp1 \* wi)) + (pow(data.N[0],2.0) \* (rp2 \* wj)));

if(i == data.ne - 1) //jezeli i bedzie ostatnim elelementem to uwzglednia wartosc konwekcji dodajac go do macierzy lokalnej

tab[i].HL[1][1] += pom;

tab[i].PL[0] = (data.C \* data.Ro \* data.Rmin / data.TauMax) \* (((data.N[0] \* data.T[0] + data.N[2] \* data.T[0]) \* data.N[0] \* rp1) + (data.N[3] \* data.T[1] + data.N[1] \* data.T[1]) \* rp2 \* data.N[1]);

tab[i].PL[1] = (data.C \* data.Ro \* data.Rmin / data.TauMax) \* (((data.N[0] \* data.T[0] + data.N[2] \* data.T[0]) \* data.N[2] \* rp1) + (data.N[3] \* data.T[1] + data.N[1] \* data.T[1]) \* rp2\*data.N[3]);

if(i == data.ne - 1)

tab[i].PL[1] += pom2;

ri += data.Rmin;

}

}

void macierz\_globalna(element \*tab, double \*\*HG, siatka data)

{

for(int i = 0; i < data.nn; i++)

HG[i] = new double[data.nn];

for(int i = 0; i < data.nn; i++)

for(int j = 0; j < data.nn; j++)

HG[i][j] = 0;

for(int i = 0; i < data.ne; i++)

{

HG[tab[i].nop1][tab[i].nop1] += tab[i].HL[0][0];

HG[tab[i].nop1][tab[i].nop2] += tab[i].HL[0][1];

HG[tab[i].nop2][tab[i].nop1] += tab[i].HL[1][0];

HG[tab[i].nop2][tab[i].nop2] += tab[i].HL[1][1];

}

for(int i = 0; i < 9; i++)

{

for(int j = 0; j < 9; j++)

cout<< HG[i][j] << setw(5)<<" ";

cout<<endl;

}

cout<<endl;

}

void wektor\_globalny(element \*tab, double \*vector, siatka data)

{

for(int i = 0; i < data.nn; i++)

vector[i] = 0;

for(int i = 0; i < data.nn; i++)

{

if(i == 0)

{

vector[tab[i].nop1] = tab[i].PL[0];

vector[tab[i].nop2] = tab[i].PL[1];

}

else

{

vector[i] = tab[i - 1].PL[1] + tab[i].PL[0];

}

}

}

bool metoda\_gaussa(int n, double \*\*A, double \*&X)

{

int i,j,k;

double m,s;

for(i = 0; i < n - 1; i++)

{

for(j = i + 1; j < n; j++)

{

if(fabs(A[i][i]) < eps) return false;

m = -A[j][i] / A[i][i];

for(k = i + 1; k <= n; k++)A[j][k] += m \* A[i][k];

}

}

for(i = n - 1; i >= 0; i--)

{

s = A[i][n];

for(j = n - 1; j >= i + 1; j--)s -= A[i][j] \* X[j];

if(fabs(A[i][i]) < eps) return false;

X[i] = s/A[i][i];

}

return true;

}

int main()

{

siatka grid;

grid.dane();

element \*el = new element[grid.ne];

double \*\*mac = new double \*[grid.nn];

double \*vec = new double [grid.nn];

grid.T[0] = grid.TempBegin;

grid.T[1] = grid.TempBegin;

double czas = 50;

double krok\_czasowy = czas / grid.TauMax;

for(int i = 0; i < krok\_czasowy; i++)

{

macierz\_lokalna(el, grid);

macierz\_globalna(el, mac, grid);

wektor\_globalny(el, vec, grid);

//wektor temperatur

double \*wt = new double[grid.nn];

for (int i = 0; i < grid.nn; i++)

wt[i] = 0;

double \*\*gauss = new double \*[grid.nn];

for (int i = 0; i < grid.nn; i++) gauss[i]=new double[grid.nn + 1];

for(int i = 0; i < grid.nn; i++)

{

for(int j = 0; j < grid.nn; j++)

gauss[i][j] = mac[i][j];

}

for(int i = 0; i < grid.nn; i++)

gauss[i][grid.nn] = vec[i];

metoda\_gaussa(grid.nn, gauss, wt);

for(int i = 0; i < grid.nn; i++)

cout<< endl << "Temperatura" << "[" << i << "]" << setw(2) << " " << wt[i];

cout<< endl;

grid.T[0] = wt[0];

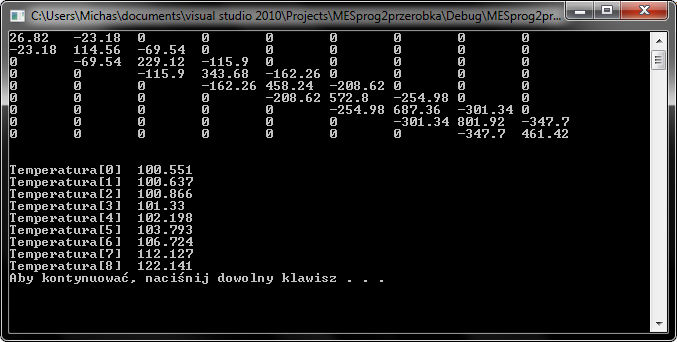
grid.T[1] = wt[1];

}

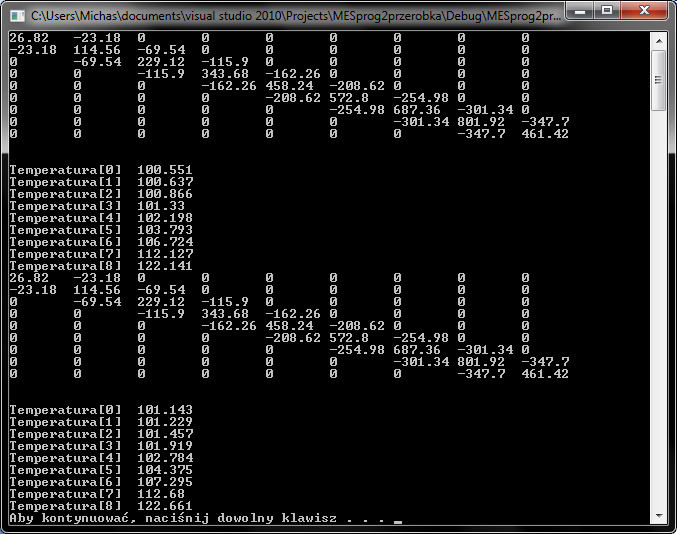
system("PAUSE");

return 0;

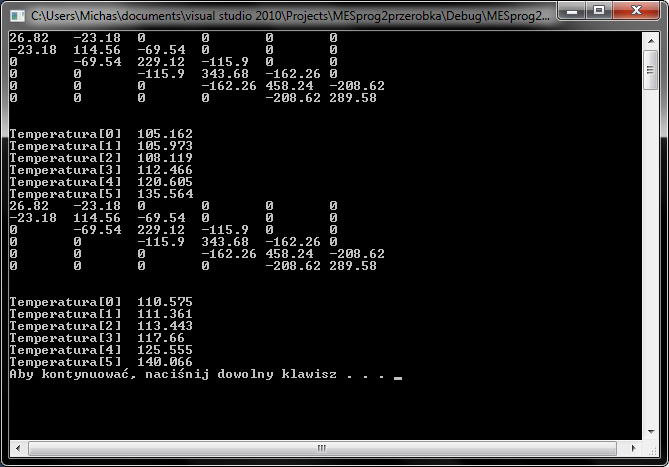
}



Powyższy zrzut przedstawia macierz globalną oraz temperatury w jednym kroku czasowym.



Powyższy zrzut przedstawia macierz globalną oraz temperatury w dwóch krokach czasowych.



Powyższy zrzut przedstawia macierz globalną oraz temperatury w dwóch krokach czasowych, z mniejszą liczbą elementów.

1. Wnioski

Po zapoznaniu się z teorią dotyczącą wyznaczania nieustalonego pola temperatury we wsadzie o przekroju okrągłym, został napisany program rozwiązujący owe zagadnienie. W programie można ustalić potrzebną ilość elementów. Program, może zostać przeznaczony do rozwiązywania skomplikowanych zagadnień z dziedziny MES.

Ilość elementów nie wpływa znacząco na wysokość temperatur, wyniki są dosyć zbliżone do siebie. Im mniej elementów tym szybszy przyrost temperatury. Program jest zależny od czasu, to znaczy, że temperatury rosną wraz z większą ilością czasu.