Bartosz Błyszcz	Indeks: 401928	Grupa: 01
METODA ELEMENTÓW SKOŃCZONYCH - SYMULACJA USTALONYCH PROCESÓW CIEPLNYCH		
WIMiIP	Informatyka Techniczna	23.12.2021

1 CZĘŚĆ TEORETYCZNA

Metoda Elementów Skończonych (MES) (*ang. Finite Element Method*) to metoda wspomagająca obliczenia inżynierskie. Obejmuje takie działy jak między innymi Fizyka oraz Technika. Służy do rozwiązywana układów równań różniczkowych za pomocą metody aproksymacji równań różniczkowych [1]. Główną ideą MES jest możliwość zamiany dowolnej ciągłej wartości na model dyskretny. Model taki posiada ograniczoną ilość elementów skończonych.

1.1 Zalety MES

- Możliwość wykorzystania do materiałów wielofazowych, bądź do materiałów, których własnością są funkcje temperatury.
- Dzięki wykorzystaniu elementów krzywoliniowych można z dużą dokładnością zaproksymować ośrodek o skomplikowanym kształcie.
- Możliwość zmiany wymiarów elementów w rozpatrywanej objętości.
- Możliwość uwzględnienia różnych warunków brzegowych.

1.2 Etapy rozwiązania problemu przy pomocy MES

1. Dobranie ograniczonej ilości węzłów w rozpatrywanym ośrodku.
2. Definicja w każdym węźle parametru, który należy obliczyć. W tym przypadku jest to temperatura.
3. Podział ośrodka na elementy skończone.
4. W każdym elemencie należy aproksymować temperaturę za pomocą wielomianu wyznaczonego z wartości w węzłach.
5. Dobór węzłowych wartości temperatur aby zapewnić jak najdokładniejsze przybliżenie do rzeczywistego pola temperatury [2].

1.3 Wykorzystane wzory

W trakcie implementacji programu wykorzystano równanie Fouriera w postaci:

$$\frac{\partial}{\partial x}\left(k_x(t)\frac{\partial t}{\partial x}\right)+\frac{\partial}{\partial y}\left(k_y(t)\frac{\partial t}{\partial y}\right)+\frac{\partial}{\partial z}\left(k_z(t)\frac{\partial t}{\partial z}\right)+Q=0 \tag{1}$$

$k_x(t)$, $k_y(t)$, $k_z(t)$ - anizotropowe współczynniki przewodzenia ciepła zależne od t

Q - prędkość generowanego ciepła

Aby rozwiązać równanie (1) należy znaleźć minimum funkcjonału, dla którego równanie (1) to równanie Eulera. Zakładając, że materiał jest izotropowy można według rachunku wariacyjnego zapisać funkcjonał pod postacią:

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

Funkcja $t(x,y,z)$ musi spełniać określone warunki brzegowe na powierzchni rozpatrywanego obszaru. W rozwiązaniu założonego problemu wykorzystano warunek brzegowy według prawa konwekcji. Funkcjonał (2) należy rozszerzyć o całkę:

$$\int_S\frac{a}{2}(t-t_\infty)^2\,dS+\int_S qt\,dS \tag{3}$$

S - powierzchnia, na której zadane są warunki brzegowe

Po rozszerzeniu, wzór przyjmuje postać:

$$\begin{aligned}J=\int_V\left(\frac{k(t)}{2}\left(\left(\frac{\partial t}{\partial x}^2\right)+\left(\frac{\partial t}{\partial y}^2\right)+\left(\frac{\partial t}{\partial z}^2\right)\right)-Qt\right)dV+\\+\int_S\frac{a}{2}(t-t_\infty)^2\,dS+\int_S qt\,dS\end{aligned} \tag{4}$$

Następnie należy podzielić rozpatrywany obszar na elementy, które posiadają temperaturę przedstawioną jako funkcję wartości węzłowych. Do tego celu należy wykorzystać funkcjonał:

$$t=\sum_{i=1}^{n}=N_it_i=\{N\}^T\{t\} \tag{5}$$

Po wprowadzeniu do wzoru (4) równania (5), oraz zamianie na wzór macierzowy otrzymano wzór dla procesu stacjonarnego:

$$[H]\{t\}+\{P\}=0 \tag{6}$$

Następnie aby przejść do procesu niestacjonarnego należy zmodyfikować równanie (1) o:

$$c\rho\frac{\partial t}{\partial\tau} \tag{7}$$

dzięki czemu otrzymuje się:

$$\frac{\partial}{\partial x}\left(k_x(t)\frac{\partial t}{\partial x}\right)+\frac{\partial}{\partial y}\left(k_y(t)\frac{\partial t}{\partial y}\right)+\frac{\partial}{\partial z}\left(k_z(t)\frac{\partial t}{\partial z}\right)+\left(Q-c\rho\frac{\partial t}{\partial \tau}\right)=0 \tag{8}$$

Otrzymano wzór który w postaci macierzowej można zapisać jako:

$$\begin{aligned}[H]\{t\}+[C]\frac{\partial}{\partial \tau}\{t\}+\{P\}=0\\ [C]=\int_V c\rho\{N\}\{N\}^T dV\end{aligned} \tag{9}$$

W założeniu rozpatrywanego w programie problemu wartości w węzłach $\{t\}$ zależą od czasu. W przedziale czasu $\Delta\tau$, można przyjąć, że wektor $\{t_0\}$ reprezentuje temperatury w węzłach w czasie $\tau=0$, dzięki czemu wektor ten można będzie wyznaczyć równaniem:

$$\{t\}=\{N_0,N_{1]}\}=\begin{Bmatrix}\{t_0\}\\ \{t_1\}\end{Bmatrix} \tag{10}$$

Dzięki czemu uzyskujemy:

$$\left([H]+\frac{[C]}{\Delta\tau}\right)\{t_1\}-\left(\frac{[C]}{\Delta\tau}\right)\{t_0\}+\{P\}=0 \tag{11}$$

Wzory określające poszczególne macierze i wektory to:

$$\begin{aligned}[H]&=\int_V k\left(\left\{\frac{\partial\{N\}}{\partial x}\right\}\left\{\frac{\partial\{N\}}{\partial x}\right\}^T+\left\{\frac{\partial\{N\}}{\partial y}\right\}\left\{\frac{\partial\{N\}}{\partial y}\right\}^T\right)dV+\int_S \alpha\{N\}\{N\}^T dS\\ [C]&=\int_V c\rho\{N\}\{N\}^T dV\\ \{P\}&=-\int_S \alpha\{N\}t_\infty dS\end{aligned} \tag{12}$$

Wzory w równaniu (12) określają: $[H]$ - równanie Fouriera, $[C]$ - macierz pojemności cieplnej, $\{P\}$ - wektor obciążeń [3][4].

2 ANALIZA KODU

Kod został podzielony w następujący sposób według przestrzeni nazw:

- **Data** - dane wykorzystywane podczas obliczeń.
- **Models** - modele służące do rozwiązania problemu. W ich skład wchodzi między innymi klasa **Element**, **Jakobian** oraz **Element4_2D**.
- **Service** - są to dodatkowe klasy zajmujące się między innymi przebiegiem obliczeń oraz dodatkowymi zadaniami takimi jak drukowaniem macierzy oraz wektorów.

Kod składa się również z plików *autoload.php*, oraz *index.php*. Plik *autoload.php* odpowiada za przechwytywanie przestrzeni nazw oraz umożliwia korzystanie z niej.

```php
<?php

spl_autoload_register('AutoLoader');

function AutoLoader($className)
{
    $class = explode("\\", $className);

    array_shift($class);
    $class = implode("\\", $class);

    $file = str_replace('\\',DIRECTORY_SEPARATOR, $class);
    require_once $file . '.php';
}
```

Kod 1: Plik autoload.php służący do przechwytywania przestrzeni nazw z plików oraz dołączania ich

```php
<?php
require_once './autoload.php';

use Mes\Service\Main;

Main::main();
```

Kod 2: Plik index.php odpowiadający za dołączenie pliku autoload.php, oraz uruchomienie programu.

Za cały proces odpowiada klasa *Mes\Service\Main*. Klasa ta w metodzie *main*, jako pierwsze generuje obiekt **Grid** (obiekt zawierający siatkę elementów wraz z ich wierzchołkami). Następnie kod przechodzi do pętli **for** w której wykonywane iteracje są zależne od maksymalnego czasu doświadczenia oraz kroku czasowego. Następnie inicjalizowane i zerowane globalne macierze H_g i C_g oraz globalny wektor P_g. W kolejnym kroku wyliczana jest dla każdego elementu macierz **H** i **C** oraz wektor **P**. Następnie tworzona jest macierz rozszerzona składająca się z macierzy **H** oraz "doklejonego" wektora **P** według wzoru [5]:

$$\begin{aligned} [H] &= [H] + \frac{[c]}{dT} \\ \{P\} &= \{P\} + \left\{\frac{[C]}{dT}\right\} * \{T_0\} \end{aligned} \tag{13}$$

Kolejnym krokiem jest obliczenie temperatur za pomocą metody Gaussa oraz przypisanie jej do poszczególnych wierzchołków. Następnie dla każdej iteracji generowany jest plik **VTK** służący do przedstawienia wyników w programie *paraview*. Na końcu pętli **for** następuje wypisanie w celach testowania wyników maksymalna i minimalna temperatura dla każdej iteracji.

```php
<?php
namespace Mes\Service;

use Mes\Data\Data;
use Mes\Models\Grid;
use Mes\Models\MatrixHg;

class Main
{
    public static function main(): void
    {

        // generowanie siatki GRID o odpowiednich parametrach, skonfigurowanych w pliku Data.php
        $grid = Grid::generateGrid(Data::GRID_H, Data::GRID_B, Data::GRID_NH, Data::GRID_NB);

        // liczenie temperatury w kroku czasowym (DELTA_T) dla konkretnego czasu TIME
        for($i = Data::DELTA_T; $i <= Data::TIME; $i+=Data::DELTA_T) {

            // inicjalizacja macierzy H_global do wykorzystania przy agregacji i wyliczeniu układu równań
            MatrixHg::init($grid->nN);

            // generowanie macierzy H oraz wektora P globalnej
            // w tym dodatkowo pobocznie dla układu niestacjonarnego jest wyliczona macierz C potrzebna do określenia pojemności cieplnej
            MatrixHg::generateMatrixHAndP($grid);

            // rozszerzenie macierzy globalnej H o wektor P, następuje zmiana wymiaru macierzy. Z kwadratowej na prostokątną o 1 więcej kolumny
            MatrixHg::createExpandMatrix();

            // wyliczenie wektora temperatur z macierzy rozszerzonej
            $temperatures = Helper::gausEquation(MatrixHg::$matrixExpand);

            //przypisanie nowych temperatur do poszczególnych nodów
            $temperatures_count = count($temperatures);
            for ($j = 0; $j < $temperatures_count; $j++) {
                $grid->nodes[$j]->t0 = $temperatures[$j];
            }

            // Generowanie plików paraview
            //  Helper::saveVTKFile($grid, $i);

```

```
            // Wypisanie minimalnej i maksymalnej temperatury dla każdej iteracji
            printf("%d :min: %.3f; max: %.3f\n",
                $i, min($temperatures), max($temperatures));
        }
    }
}
```

Kod 3: Plik Main.php

2.1 Generowanie obiektu Grid

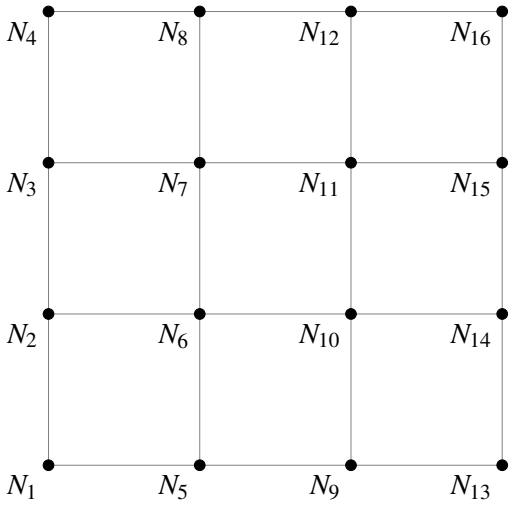
Generowanie obiektu grid znajduje się w metodzie *Mes\Models\Grid::generateGrid*

```
// ***
public static function generateGrid(float $H, float $B, float $nH, float $nB): grid
{
    $grid = new Grid($H, $B, $nH, $nB);
    $nodes = [];
    $elements = [];

    $x = 0;
    $dY = ($grid->H / ($grid->nH - 1.0));
    $dX = ($grid->B / ($grid->nB - 1.0));

    for ($i = 0; $i < ($grid->nN); $i++ ) {
        $x = $i  < ($grid->nH + ($x * $grid->nH)) ? $x : $x+1;

        $x_tmp = $x * $dX;
        $y = ($i % $grid->nH) * $dY;

        $bc = $x_tmp === 0.0 || $y === 0.0 || $x_tmp === $grid->B || $y === $grid->H ? 1 : 0;

        $nodes[$i] = new Node(
            $x_tmp,
            $y,
            Data::STALA_T_0,
            $bc
        );
    }

    $e1 = 0;
    for ($i = 0; $i < $grid->nE; $i++) {
        $e1 = $i > 0 && $i % ($grid->nH - 1.0)  === 0 ? $e1 + 1 : $e1;

        $elements[$i] = new Element([
            $i + $e1,
            $i + $e1 + $grid->nH,
            $i + $e1 + $grid->nH + 1,
            $i + $e1 + 1,
        ]);
    }

    $grid->nodes = $nodes;
```

```
41      $grid->elements = $elements;
42
43      unset($elements, $nodes);
44      return $grid;
45  }
46  // ***
```

Kod 4: Generowanie obiektu grid za pomocą metody: *Mes\Models\Grid::generateGrid*

Najpierw generowana jest tablica obiektów **Node**, które posiadają takie własności jak: pozycja w układzie kartezjańskim, temperatura początkowa oraz wartość przedstawiającą ewentualne istnienie warunku brzegowego. W następnym korku generowana jest tablica obiektów **Element** które posiadają indeksy obiektów znajdujących się w tablicy obiektów **Node**. Indeksy zawarte w obiekcie **Element** oznaczają obiekty **Node** znajdujące się na wierzchołkach elementów. Dla elementu 3x3 siatka będzie wyglądała jak na schemacie (1).



Rysunek 1: Schemat wygenerowanej siatki dla elementu 3x3

2.2 Inicjalizacja $[H_g]$, $[C_g]$, $\{P_g\}$

Inicjalizacja $[H_g]$, $[C_g]$ oraz $\{P_g\}$. Odbywa się w klasie *Mes\Models\MatrixHg::init*

```
1       //***
2       public static function init(int $nodes): void
3       {
4           for ($i = 0; $i < $nodes; $i++) {
5               for($j = 0; $j < $nodes; $j++) {
6                   self::$H[$i][$j] = 0;
7                   self::$C[$i][$j] = 0;
8               }
9
10              self::$P[$i] = 0;
11          }
12      }
13      //***
```

Kod 5: Mes\Models\MatrixHg::init

Inicjaliacja odbywa się poprzez wygenerowanie tablic o wymiarach NxN dla $[H_g]$, $[C_g]$ oraz N dla $\{P_g\}$ gdzie N to ilość obiektów o klasie **Node**.

2.3 Generowanie $[H]$, $[C]$, $\{P\}$

Kolejnym krokiem jest wyznaczenie wartości lokalnych $[H]$, $[C]$, $\{P\}$ dla każdego elementu na siatce. W pierwszej kolejności tworzony jest obiekt **Element4_2d**. Następny jest **for** iterowany po wszystkich elementach. W pętli **for** następuje najpierw wyliczenie jakobianów dla każdego elementu. Następnym krokiem jest wyliczenie $[H]$. Jako kolejne liczone są wartości $[H_{bc}]$ oraz $\{P\}$ wchodzące w skład równania warunku brzegowego konwekcji. Jako następna wyliczona jest $[C]$, a na końcu następuje agregacja lokalnych $[H]$, $[C]$ oraz $\{P\}$ do $[H_g]$, $[C_g]$ oraz $\{P_g\}$ wykorzystując równania (13).

```
//**
public static function generateMatrixHAndP(Grid $grid): void
{
    $element4_2D = new Element4_2D();
    for($i = 0; $i < $grid->nE; $i++ ) {

        $element = $grid->elements[$i];
        \assert($element instanceof Element);

        // generowanie tablicy jakobianów
        // każdy element ma tablice jakobianów o wielkości Points**2
        // każdy punkt całkowania dla elementu 2D ma swój jakobian
        // wiąże się to z tym, że jakobian to stosunek pól powierzchni
           układu globalnego i lokalnego
        Jakobian::generateJakobian($i, $element4_2D, $grid);

        // generowanie macierzy H dla każdego elementu wewnątrz niego
        $element->computeHmatrix($element4_2D);

        // generowanie macierzy Hbc i P czyli dwóch części warunku
           brzegowego, rozdzielonych ze względu na to,
        // że wzór na konwekcje to: q = a(t - t_0) tak więc mamy q = at
            - at_0
        // z tego znamy wszystko a, t_0, dlatego jedna część idzie do
           Hbc, a ta znana do P
        $element->computeHbcAndPMatrix($element4_2D, $grid);

        // macierz C służy do wyliczenia pojemności cieplnej materiału,
            dzięki czemu można obliczyć
        // temperaturę po kroku czasowym delta_T
        $element->computeCMatrix($element4_2D);

        // Agregacja wykorzystuje wzory:
        // [H] = [H] + [C]/delta_T
        // {P} + ({[C]/delta_T}*{T_0})
        self::aggregateHAndPMatrix($element, $grid);
    }

}
//**
```

Kod 6: Mes\Models\MatrixHg::generateMatrixHAndP

2.4 Stworzenie macierzy rozszerzonej

Kolejnym krokiem jest stworzenie macierzy rozszerzonej składającej się z agregowanych $[G_g]$ oraz $\{P_g\}$

```
//**
public static function createExpandMatrix(): void
{
    self::$matrixExpand = self::$H;
    $last = count(self::$H);

    $_counter = 0;
    foreach (self::$P as $p) {
        self::$matrixExpand[$_counter][$last] = $p;
        $_counter++;
    }
}
//**
```

Kod 7: Mes\Models\MatrixHg::createExpandMatrix

Macierz rozszerzona składa się z "doklejonego" $\{P_g\}$ do $[H_g]$ według schematu (14)

$$\begin{bmatrix} H_{g11} & H_{g12} & H_{g13} \\ H_{g21} & H_{g22} & H_{g23} \\ H_{g31} & H_{g32} & H_{g33} \end{bmatrix} oraz \begin{bmatrix} P_{g1} \\ P_{g2} \\ P_{g3} \end{bmatrix} daje: \begin{bmatrix} H_{g11} & H_{g12} & H_{g13} & P_{g1} \\ H_{g21} & H_{g22} & H_{g23} & P_{g2} \\ H_{g31} & H_{g32} & H_{g33} & P_{g3} \end{bmatrix} \tag{14}$$

2.5 Obliczenie i zapis temperatury

Następnym krokiem jest wykorzystanie macierzy rozszerzonej do wyliczenia temperatur cząstkowych. Temperatury są wyliczane z metody Gaussa.

```
$temperatures = Helper::gausEquation(MatrixHg::$matrixExpand);

$temperatures_count = count($temperatures);
for ($j = 0; $j < $temperatures_count; $j++) {
    $grid->nodes[$j]->t0 = $temperatures[$j];
}
```

Kod 8: Obliczenie temperatur cząstkowych

2.6 Zapis do pliku VTK

Ostatnim krokiem jest zapis wyników do pliku VTK

```
public static function saveVTKFile(Grid $grid, int $id): void
{
    $file = fopen(sprintf('file_%d.vtk', $id), 'wb');
    fwrite($file, "# vtk DataFile Version 2.0\n");
    fwrite($file, "Unstructured Grid Example\n");
    fwrite($file, "ASCII\n");
    fwrite($file, "DATASET UNSTRUCTURED_GRID\n\n");
    fwrite($file, sprintf("POINTS %d float\n", $grid->nN));

    foreach ($grid->nodes as $node) {
        /** @var Node $node */
        fwrite($file, sprintf("%.6f %.6f 0\n", $node->x, $node->y));
```

```
13      }
14
15      fwrite($file, "\n\n\n");
16      fwrite($file, sprintf("CELLS %d %d\n", $grid->nE, $grid->nE*5));
17      foreach ($grid->elements as $element) {
18          fwrite($file, "4 ");
19
20          /** @var Element $element */
21          foreach ($element->ID as $value) {
22              fwrite($file, sprintf("%d ", $value));
23          }
24
25          fwrite($file, "\n");
26      }
27
28      fwrite($file, sprintf("\n\nCELL_TYPES %d\n", $grid->nE));
29      foreach ($grid->elements as $element) {
30          fwrite($file, "9\n");
31      }
32
33      fwrite($file, "\n");
34
35      fwrite($file, sprintf("POINT_DATA %d\n", $grid->nN));
36      fwrite($file,"SCALARS scalars float 1\n");
37      fwrite($file,"LOOKUP_TABLE default\n");
38      foreach ($grid->nodes as $node) {
39          /** @var Node $node */
40          fwrite($file,sprintf("%.6f\n", $node->t0));
41      }
42
43      fclose($file);
44  }
```

Kod 9: Zapis do pliku VTK

3 Analiza wyników

3.1 Wartości początkowe

- **100** - temperatura początkowa
- **100** - czas symulacji [s]
- **1** - krok czasu [s]
- **1200** - maksymalna temperatura końcowa [C]
- **300** - alfa $\left[\frac{W}{m^2K}\right]$
- **0.1** - H [m]
- **0.1** - B [m]
- **31** - N_H
- **31** - N_B
- **700** - Ciepło właściwe $\left[\frac{J}{kg°C}\right]$
- **25** - Anizotropowy współczynnik przewodzenia ciepła $\left[\frac{W}{m°C}\right]$
- **7800** - Gęstość $\left[\frac{kg}{m^3}\right]$ [5]

3.2 Wyniki

Na poniższych zdjęciach otrzymano wyniki w postaci: iteracja, temperatura minimalna, temperatura maksymalna. W przeprowadzonej symulacji otrzymano następujące wyniki:

```
~/Dokumenty/AGH/sprawozdania/suvres/semestr5/mes/lab4
> php8.0 ./index.php
1 :min: 100.000; max: 149.557
2 :min: 100.000; max: 177.445
3 :min: 100.000; max: 197.267
4 :min: 100.000; max: 213.153
5 :min: 100.000; max: 226.683
6 :min: 100.000; max: 238.607
7 :min: 100.000; max: 249.347
8 :min: 100.000; max: 259.165
9 :min: 100.000; max: 268.241
10 :min: 100.000; max: 276.701
11 :min: 100.001; max: 284.641
12 :min: 100.002; max: 292.134
13 :min: 100.003; max: 299.237
14 :min: 100.005; max: 305.997
15 :min: 100.009; max: 312.451
16 :min: 100.014; max: 318.631
17 :min: 100.021; max: 324.564
18 :min: 100.032; max: 330.271
19 :min: 100.046; max: 335.772
20 :min: 100.064; max: 341.085
21 :min: 100.088; max: 346.223
22 :min: 100.119; max: 351.199
23 :min: 100.157; max: 356.026
24 :min: 100.203; max: 360.713
25 :min: 100.258; max: 365.269
26 :min: 100.325; max: 369.702
27 :min: 100.402; max: 374.020
```

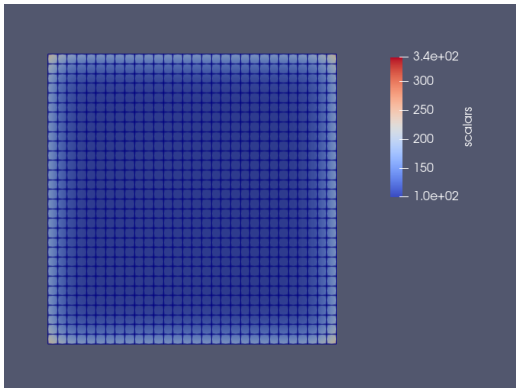
```
28 :min: 100.492; max: 378.229
29 :min: 100.596; max: 382.336
30 :min: 100.714; max: 386.345
31 :min: 100.847; max: 390.263
32 :min: 100.997; max: 394.093
33 :min: 101.163; max: 397.840
34 :min: 101.347; max: 401.508
35 :min: 101.550; max: 405.101
36 :min: 101.771; max: 408.622
37 :min: 102.012; max: 412.073
38 :min: 102.273; max: 415.459
39 :min: 102.555; max: 418.781
40 :min: 102.858; max: 422.043
41 :min: 103.182; max: 425.246
42 :min: 103.528; max: 428.393
43 :min: 103.895; max: 431.487
44 :min: 104.285; max: 434.528
45 :min: 104.696; max: 437.520
46 :min: 105.130; max: 440.463
47 :min: 105.587; max: 443.359
48 :min: 106.066; max: 446.211
49 :min: 106.567; max: 449.019
50 :min: 107.091; max: 451.784
51 :min: 107.638; max: 454.509
52 :min: 108.206; max: 457.195
53 :min: 108.797; max: 459.842
54 :min: 109.410; max: 462.452
55 :min: 110.045; max: 465.026
56 :min: 110.701; max: 467.565
57 :min: 111.379; max: 470.070
58 :min: 112.078; max: 472.542
```

Rysunek 2: Wyniki symulacji część 1

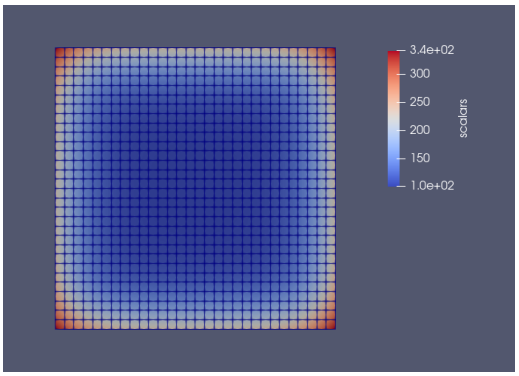
```
60 :min: 113.540; max: 477.392
61 :min: 114.302; max: 479.770
62 :min: 115.084; max: 482.119
63 :min: 115.886; max: 484.440
64 :min: 116.708; max: 486.732
65 :min: 117.550; max: 488.997
66 :min: 118.411; max: 491.235
67 :min: 119.290; max: 493.448
68 :min: 120.189; max: 495.635
69 :min: 121.105; max: 497.798
70 :min: 122.040; max: 499.937
71 :min: 122.992; max: 502.052
72 :min: 123.962; max: 504.144
73 :min: 124.948; max: 506.214
74 :min: 125.952; max: 508.262
75 :min: 126.971; max: 510.289
76 :min: 128.007; max: 512.294
77 :min: 129.059; max: 514.280
78 :min: 130.126; max: 516.245
79 :min: 131.208; max: 518.191
80 :min: 132.306; max: 520.118
81 :min: 133.417; max: 522.026
82 :min: 134.543; max: 523.916
83 :min: 135.683; max: 525.788
84 :min: 136.836; max: 527.643
85 :min: 138.003; max: 529.480
86 :min: 139.183; max: 531.301
87 :min: 140.375; max: 533.105
88 :min: 141.580; max: 534.893
89 :min: 142.797; max: 536.666
```

```
87 :min: 140.375; max: 533.105
88 :min: 141.580; max: 534.893
89 :min: 142.797; max: 536.666
90 :min: 144.026; max: 538.423
91 :min: 145.266; max: 540.165
92 :min: 146.518; max: 541.892
93 :min: 147.781; max: 543.604
94 :min: 149.054; max: 545.302
95 :min: 150.339; max: 546.987
96 :min: 151.633; max: 548.658
97 :min: 152.937; max: 550.315
98 :min: 154.251; max: 551.959
99 :min: 155.575; max: 553.590
100 :min: 156.908; max: 555.209
```

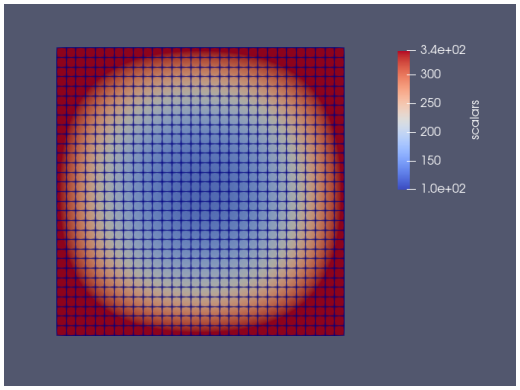
Rysunek 3: Wyniki symulacji część 2



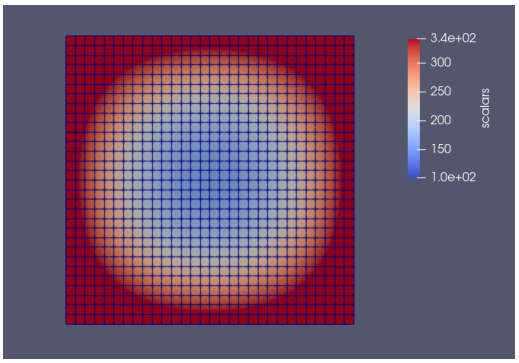
(a) Symulacja po 5s



(b) Symulacja po 20s



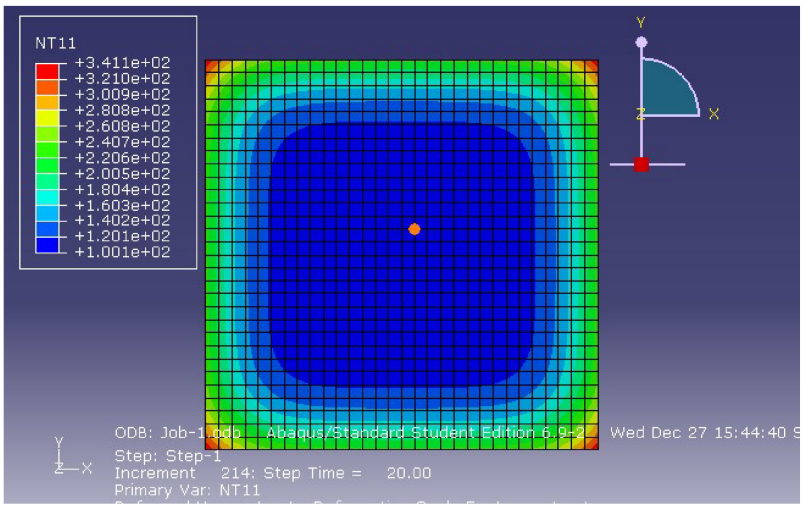
(c) Symulacja po 80s



(d) Symulacja po 100s

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



Rysunek 5: Wyniki testowe zaprezentowane przez prowadzącego [5]

Otrzymane w symulacji wyniki (Rysunek. 2) są zbliżone do tych zaprezentowanych na Rysunku. 5. Świadczyć to może o przyjętym różnym przybliżeniu wyników.

4 Wnioski

Implementacja MES w programie pozwoliła na lepsze zrozumienie problemu. Bowiem wymagała dużo większego wgłębienia w teorię oraz wymusiła większe zaangażowanie w omawiany przedmiot. Pozwoliło to na lepsze zrozumienie materiału omawianego na wykładzie i laboratorium.

Cała symulacja pokazała jak może przebiegać rozkład temperatur w siatce elementów. Oraz jak bardzo może zmieniać się wynik w zależności od ilości elementów, zagęszczenia siatki oraz kroku czasowego. Laboratoria pozwoliły również zobaczyć jak może rozkładać się temperatura w siatce nałożonej na przedmiot dwu wymiarowy na przykład w płytce metalowej nagrzewanej w piecu.

Pokazane ”krótkie” obliczenia mogą trwać parę minut. Im bardziej skomplikowany problem tym więcej mocy obliczeniowej potrzebuje. Może to wydłużyć rozwiązywanie problemu do wielu godzin, dni, a nawet tygodni. Możliwym rozwiązaniem tego problemu jest rozproszenie obliczeń za pomocą na przykład OpenMPI. Dzięki czemu ”tanim kosztem” można uzyskać dużo większą moc obliczeniową niż na to pozwala pojedynczy komputer.

Bibliografia

[1] Dr hab. inż. Krzysztof Banaś, prof. AGH. „*Wprowadzenie do MES*”. URL: http://ww1.metal.agh.edu.pl/~banas/wprowadzenie_do_MES.pdf (term. wiz. 2022-01-05).

[2] Prof. Dr hab. inż. Andrij Milenin. „*Metoda elementów skończonych*”. URL: https://home.agh.edu.pl/~milenin/Dydaktyka/MES/MES_AGH_2007.pdf (term. wiz. 2022-01-05).

[3] Dr inż. Kustra Piotr. „*SYMULACJA USTALONYCH PROCESÓW CIEPLNYCH*”. URL: https://home.agh.edu.pl/~pkustra/MES/FEM_1.pdf (term. wiz. 2022-01-05).

[4] Dr inż. Kustra Piotr. „*SYMULACJA NIEUSTALONYCH PROCESÓW CIEPLNYCH*”. URL: https://home.agh.edu.pl/~pkustra/MES/FEM_transient_2d.pdf (term. wiz. 2022-01-05).

[5] Dr inż. Kustra Piotr. „*Test case 2d transient solution*”. URL: https://home.agh.edu.pl/~pkustra/MES/TC_2d.pdf (term. wiz. 2022-01-05).