1 dodatek B

1.1 transformata.cpp

```
#include "transformataHopfCole.h"
int main()
{
  transformataHopfCole Burger;
  Burger.wynikiKonczowe();
}
```

Listing 1: transformata.cpp

1.2 transformataHopfCole.h

```
#pragma once
#include <vector>
#include <cmath>
#include <algorithm>
#include <conio.h>
#include <math.h>
using namespace std;
#ifndef M_PI
#define M_PI 3.14159265358979323846
#endif

#ifndef M_E
#define M_E 2.71828182845904523536
#endif

class transformataHopfCole {

  const double k = 0.005, h = 0.1;
    const int N = 1 / h + 1; // liczba punkt w siatki
  const double n = ((0.5 * pow(h, 2)) / k) - 0.01;
  const double r = n * k / pow(h, 2);
  double t = 0;

  vector<double>theta;
  vector<double>mu;
  vector<double>inicjacjaThetaX0(); //wzor 8 z pracy
  vector<double>liczenieThetaOdCzasu();

```

```
29    vector<double>liczenieMu(vector<double>newMu);//wzor 15
30    void zapisDoPliku(vector<double> Theta, double newT);
31    public:
32    void wynikiKonczowe();
33
34 };
```

Listing 2: transformataHopfCole.h

1.3 transformataHopfCole.cpp

```
1 #include "transformataHopfCole.h"
2 #include <iostream>
3 #include <fstream>
4 #include <string>
5
6
7
8 vector<double> transformataHopfCole::inicjacjaThetaX0() {
9    vector<double>newTheta;
10   for (int i = 0; i < N; ++i) {
11     double potega = (-1 / (2 * M_PI * n)) * (1 - cos(M_PI * (i * h)));
12     newTheta.push_back(exp(potega));
13   }
14   zapisDoPliku(liczenieMu(newTheta), 0);
15   return newTheta;
16 }
17
18 vector<double> transformataHopfCole::liczenieThetaOdCzasu() {
19   vector<double>thetaDlaCzasu(N);
20   thetaDlaCzasu = theta;
21   for (int j = 1; j < 10; ++j) {
22     t = j * k;
23     vector<double>v(N);
24     v[0] = (1 - 2 * r) * thetaDlaCzasu[0] + 2 * r * thetaDlaCzasu[1]; //warunki
       brzegowe
25     v[N - 1] = 2 * r * thetaDlaCzasu[N - 2] + (1 - 2 * r) * thetaDlaCzasu[N - 1];
26     for (int i = 1; i < (N - 1); ++i) {
27       double wynik = r * thetaDlaCzasu[i - 1] + (1 - 2 * r) * thetaDlaCzasu[i] + r
        * thetaDlaCzasu[i + 1];
28       v[i] = wynik;
29     }
30     thetaDlaCzasu = v;
```

```
31      zapisDoPliku(liczenieMu(thetaDlaCzasu), t);
32
33    }
34    return thetaDlaCzasu;
35  }
36
37  vector<double> transformataHopfCole::liczenieMu(vector<double> newMu) {
38    vector<double>newMu1(N);
39    newMu1[0] = 0;//warunek poczatkowy
40    newMu1[N - 1] = 0;//warunek koncowy
41    for (int i = 1; i < (N - 1); ++i) {
42      newMu1[i] = (-(n / h) * ((newMu[i + 1] - newMu[i - 1]) / newMu[i]));
43    }
44
45    return newMu1;
46  }
47
48  void transformataHopfCole::zapisDoPliku(vector<double> newMu, double newT) {
49    string nazwaPliku = "Wyniki1DlaT" + to_string(newT) + ".txt";
50    std::ofstream plik(nazwaPliku);
51
52    // Sprawdzamy, czy plik zosta  otwarty poprawnie
53    if (!plik) {
54      std::cerr << "Nie mo na otworzy  pliku!" << std::endl;
55      return;
56    }
57    for (int i = 0; i < size(newMu); ++i) {
58
59      plik << (i * h) << " " << newMu[i] << '\n';
60    }
61
62    // Zamykamy plik
63    plik.close();
64  }
65
66  void transformataHopfCole::wynikiKonczowe() {
67    theta = inicjacjaThetaX0();
68    theta = liczenieThetaOdCzasu();
69  }
```

Listing 3: transformataHopfCole.cpp