1 Opis programu

1.1 transformataHopfCole.h

Plik nagłówkowy zawiera definicje metod klasy transformaty. Na początku zostały dołączone wszystkie klasy, które są potrzebne do działania programu, jak pokazuje listing 1.

```
1   #pragma once
2   #include <vector>
3   #include <cmath>
4   #include <algorithm>
5   #include <conio.h>
6   #include <math.h>;
```

Listing 1: początek pliku transformataHopfCole.h

Następnie zostały napisana dyrektywa, która pozwala na używanie wszystkich nazw z przestrzeni nazw ”std” bez konieczności ich niefiksowania. Przestrzeń nazw definiuję standardowych klas i funkcji z biblioteki standardowej w C++, takich jak ”cout”, ”vecotr”, ”string”, co skraca trochę kod. Potem zostały definiowanie stała Pi oraz e, jeśli nie są szybciej zdefiniowanie, co przedstawia listing **??**.

```
7   using namespace std;
8   #ifndef M_PI
9   #define M_PI 3.14159265358979323846
10  #endif
11
12  #ifndef M_E
13  #define M_E 2.71828182845904523536
14  #endif
```

Listing 2: początek pliku definiowanieStałych

Potem została zdefiniowania klasa, w której są wszystkie metody oraz stale potrzebnie do pracy programu.

```
1   class transformataHopfCole {
2     const int N = 11;       // liczba punktów siatki
3     const double k = 0.005, h = 0.1;
4     const double n = ((0.5 * pow(h, 2)) / k) - 0.01;
5     const double r = n*k / pow(h, 2);
6     double t = 0;
7
8     vector<double>theta;
9     vector<double>mu;
10    vector<double> inicjacja_u(); //wzror 4 z pracy
11    vector<double>inicjacjaThetaX0(); //wzor 8 z pracy
12    vector<double>liczenieThetaOdCzasu();
13
14    vector<double>liczenieMu(vector<double>newMu); //wzor 15
```

```
15          void zapisDoPliku(vector<double> Theta, double newT);
16          public:
17          void wynikiKonczowe();
18        };
```

Listing 3: klasa tranformataHopfCofe

Pierwsza stała oznacza liczbę punktów, ”k” odpowiada za krok czasowy, natomiast ”h” za krok przestrzennego w dyskretyzacji przestrzennej.
W 8 i 9 linii zostały inicjowanie vectoru, gdzie zostaniom zapisanie wyniki programu. W kolejnych trzech linach zostały już zdefiniowanie klasy, które krok po kroku będą obliczały wynik końcowy oraz w linii 14.
Przedostatnia prywatna metoda służy, by otrzymanie wyniki zostały zapisanie do pliku. Ostaną metodą jest ”wynikiKoncowe()”, która jest publiczna.

1.2 transformataHopfCole.cpp

W kolejnym pliku znajdują się ciała metod klasy. Na początku pliku zostały załączone biblioteki oraz własna klasa do prawidłowego działania programu.

```
1 #include "transformataHopfCole.h"
2 #include <iostream>
3 #include <fstream>
4 #include <string>
```

Listing 4: poczadek pliku transformataHopfCole.cpp

Następne jest ciało metody inicjacjaThetaX0 co przedstawia listing 5. Ta metoda ma na celu inicjacji θ dla czasu zerowego. Pętla służy do wygenerowana dla wszystkich θ wartości początkowych, z których potem zostaną wykorzystanie do dalszych obliczeń. Pod koniec metody zostanie wywołania kolejna metoda, która zapiszę wartości do pliku tekstowego oraz zwrócenie tych wartości do vectora.

```
1     vector<double> transformataHopfCole::inicjacjaThetaX0() {
2       vector<double>newTheta;
3       for (int i = 0; i < N; ++i) {
4         double potega = (-1 / (2 * M_PI * n)) * (1 - cos(M_PI * (i * h)));
5         newTheta.push_back(exp(potega));
6       }
7       zapisDoPliku(liczenieMu(newTheta), 0);
8       return newTheta;
9       }
```

Listing 5: ciało metody inicjacjaThetaX0

Kolejną metodą jest ”liczenieThetaOdCzasu” (listing 6). Metoda ma na celu policzenie θ dla czasu. Pierwsza pętla służy do policzenie ”t”, czyli kroku czasowego. Potem jest zdefiniowany nowy vector,

w którym zostaną zapisanie nowe wartości. Druga pętla służy do policzenia dla θ od 1 do N-1. Po drugiej pętli zostaną nadpisanie wartości dla theyaDlaCzasu nowymi wartościami dla konkretnego czasu. Na końcu jest wywołania metoda, która otrzymanie wyniki zapiszę do pliku. Na samym końcu zostanie zwrócony vector.

```
vector<double> transformataHopfCole::liczenieThetaOdCzasu() {
  vector<double>thetaDlaCzasu(N);
  thetaDlaCzasu = theta;
  for (int j = 1; j < 10; ++j) {
    t = j * k;
    vector<double>v(N);
    v[0] = (1 - 2 * r) * thetaDlaCzasu[0] + 2 * r * thetaDlaCzasu[1]; //warunki brzegowe
    v[N - 1] = 2 * r * thetaDlaCzasu[N - 2] + (1 - 2 * r) * thetaDlaCzasu[N - 1];
    for (int i = 1; i < (N - 1); ++i) {
      double wynik = r * thetaDlaCzasu[i - 1] + (1 - 2 * r) * thetaDlaCzasu[i] + r *
    thetaDlaCzasu[i + 1];
      v[i] = wynik;
    }
    thetaDlaCzasu = v;
    zapisDoPliku(liczenieMu(thetaDlaCzasu), t);

  }
  return thetaDlaCzasu;
  }
```

Listing 6: ciało metody liczenieThetaOdCzasu

Kolejną metodą jest ”liczenieMu”, która wykorzystuję θ od czasu do policzenia $u(x,t)$. Na wyraz 0 i (N-1) jest przypisana wartość 0, co wynika z warunki początkowego i końcowego. Kolejne wartości są liczone w pętli for, jak pokazuje listing 7.

```
vector<double> transformataHopfCole::liczenieMu(vector<double> newMu) {
  vector<double>newMu1(N);
  newMu1[0] = 0;//warunek poczatkowy
  newMu1[N - 1] = 0;//warunek koncowy
  for (int i = 1; i < (N - 1); ++i) {
    newMu1[i] = (-(n / h) * ((newMu[i + 1] - newMu[i - 1]) / newMu[i]));
  }

  return newMu1;
  }
```

Listing 7: ciało metody liczenieMu

Przedostatnią metodą jest ”zapisDoPliku”, który ma na celu otrzymanych wyników zapisach do pliku. Na początku jest inicjowany string, który będzie nazwą pliku. Potem program tworzy nowy plik, jeśli nie istnieje, jak istnieje to zawartość zostanie zastąpiona. Potem w warunku jest sprawdzenie czy udało się otworzyć plik, jeśli nie to pojawi się odpowiednia informacja. Pętla for ma na celu zapisanie

wszystkich punktów do pliku. Na końcu program zamyka plik, gdy już skończył pracę.

```
1  void transformataHopfCole::zapisDoPliku(vector<double> newMu, double newT) {
2    string nazwaPliku = "Wyniki1DlaT" + to_string(newT) + ".txt";
3    std::ofstream plik(nazwaPliku);
4
5    // Sprawdzamy, czy plik łzosta otwarty poprawnie
6    if (!plik) {
7      std::cerr << "Nie żmona ćotworzy pliku!" << std::endl;
8      return;
9    }
10   for (int i = 0; i < size(newMu); ++i) {
11
12     plik << (i * h) << " " << newMu[i] << '\n';
13   }
14
15   // Zamykamy plik
16   plik.close();
17 }
```

Listing 8: metoda zapisDoPliku

Ostaną metodą są "wynikiKoncowe", który na początku wywołuje metodę "inicjacjaThetaX0", następnie "liczenieThetaOdCzasu". Kolejne metody nie są potrzebne by je wywołać, z powodu, że metoda "liczenieThetaOdCzasu" wywołuje metodę która liczy $u(x,t)$ i metodą, która zapisuję dane do pliku.

```
1 void transformataHopfCole::wynikiKonczowe() {
2   theta = inicjacjaThetaX0();
3   theta = liczenieThetaOdCzasu();
4   }
```

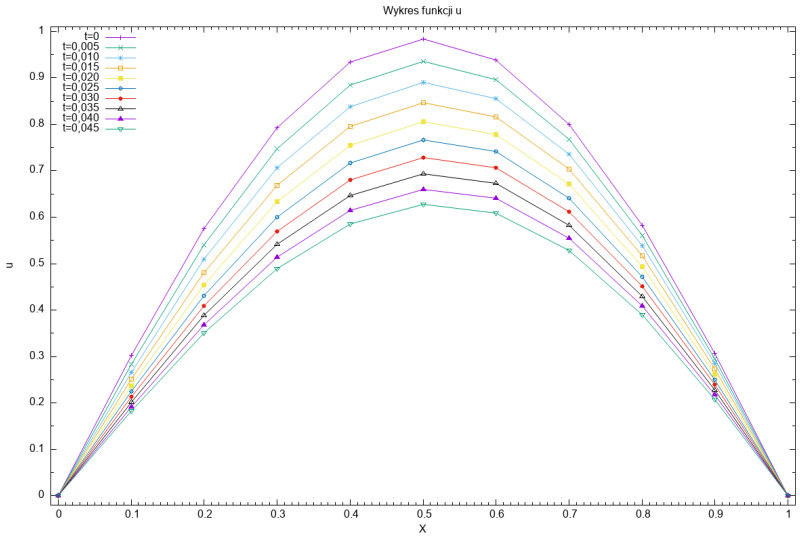
Listing 9: metoda wynikiKonczowe

1.3 transformata.cpp

Kolejnym plikiem jest transformata.cpp, który ma funkcję główną main.cpp. Która na początku ma dołączona klasę. Następnie w main tworzy się obiekt klasy transformataHopfCole, a następnie wywołuję się metodę klasy, by program policzył nam wyniki.

```
1 #include "transformataHopfCole.h"
2 int main()
3 {
4   transformataHopfCole Burger;
5   Burger.wynikiKonczowe();
6 }
```

Listing 10: transformata.cpp



Rysunek 1: k = 0.005, h = 0.1

2 wyniki

Wygenerowano wykresy dla $k = 0.0005$ i $h = 0.1$. Dla coraz większego czasu można zauważyć, że amplituda rozwiązana maleje, można to zauważyć na rys 1, wyniki punktów wykresu są przedstawione na tabelkach 1 i 2.

Można zauważyć, że jak zmniejszymy k do 0.0005, to można zauważyć, że dla czasu 0 funkcja jest trójkąta. Natomiast dla większego czasu można zauważyć, że funkcja się powoli zaokrągla i przypomina coraz bardziej wykres dla sinusa, można to zauważyć na rys 2.

Dla $k = 0.05, h = 0.01$ wykres robi się trójkątny oraz niektóre wartości przyjmują wartości ujemne, lub przyjmuję wartości zero lub bliskie zeru.

3 dodatek

3.1 transformata.cpp

```
1  #include "transformataHopfCole.h"
2  int main()
3  {
4     transformataHopfCole Burger;
5     Burger.wynikiKonczowe();
6  }
```

Listing 11: transformata.cpp

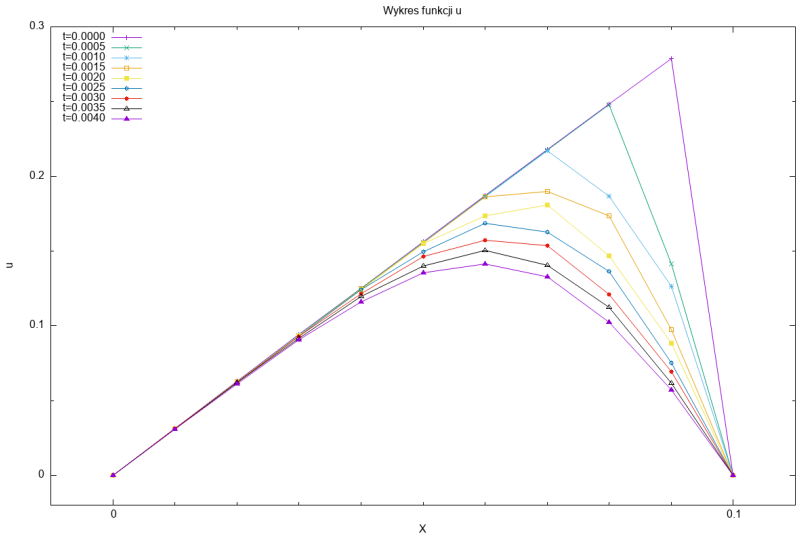
3.2 transformataHopfCole.h

x\t	0	0.005	0,010	0.015	0.020	0.025
0	0	0	0	0	0	0
0.1	0.301705	0.283068	0.266381	0.251276	0.237487	0.224811
0.2	0.574577	0.540199	0.509139	0.480833	0.454852	0.430863
0.3	0.792316	0.747317	0.706078	0.668073	0.632879	0.600146
0.4	0.933565	0.88415	0.838007	0.794843	0.754384	0.716382
0.5	0.984036	0.93621	0.890555	0.847073	0.805711	0.766389
0.6	0.938116	0.896627	0.856073	0.816677	0.778584	0.741879
0.7	0.799679	0.767504	0.735313	0.703418	0.672066	0.641447
0.8	0.581939	0.560385	0.53838	0.516199	0.494079	0.472219
0.9	0.306254	0.295543	0.284455	0.273144	0.26175	0.250398
1	0	0	0	0	0	0

Tabela 1: Wyniki dla k=0.005,h=0.1, dla czasu od 0 do 0.025

x\t	0.030	0.35	0.40	0.45	0.45
0	0	0	0	0	0
0.1	0.213091	0.202201	0.192041	0.182528	0.182528
0.2	0.408602	0.387857	0.368453	0.350247	0.350247
0.3	0.569591	0.540975	0.514101	0.488803	0.488803
0.4	0.680618	0.6469	0.615061	0.584955	0.584955
0.5	0.729015	0.693493	0.659731	0.627636	0.627636
0.6	0.706609	0.672786	0.640407	0.609451	0.609451
0.7	0.611699	0.582919	0.555172	0.528496	0.528496
0.8	0.450776	0.429873	0.409598	0.390013	0.390013
0.9	0.239189	0.228205	0.217507	0.207141	0.207141
1	0	0	0	0	0

Tabela 2: Wyniki dla k=0.005,h=0.1, dla czasu od 0.030 do 0.045



Rysunek 2: k = 0.0005, h = 0.01

```cpp
#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 int N = 11;       // liczba punktów siatki
  const double k = 0.005, h = 0.1;
  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();

  vector<double>liczenieMu(vector<double>newMu);//wzor 15
```

```
30    void zapisDoPliku(vector<double> Theta, double newT);
31    public:
32    void wynikiKonczowe();
33
34 };
```

Listing 12: transformataHopfCole.h

3.3 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 łzosta otwarty poprawnie
53   if (!plik) {
54     std::cerr << "Nie żmona ćotworzy 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 13: transformataHopfCole.cpp