N2. Zadaniem było rozwiązać poniższy układ równań poprawnie implementując algorytm do rozwiązywania układu z macierzą trójdiagonalną oraz przedstawić graficznie rozwiązanie.

$$\begin{bmatrix} b_0 & c_0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ a_1 & b_1 & c_1 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & a_2 & b_2 & c_2 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & a_3 & b_3 & c_3 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \dots & a_{N-1} & b_{N-1} & c_{N-1} \\ 0 & 0 & 0 & 0 & 0 & \dots & 0 & a_N & b_N \end{bmatrix} \cdot \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{N-1} \\ y_N \end{bmatrix} = \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ f_3 \\ \vdots \\ f_{N-1} \\ f_N \end{bmatrix}$$

Opis i wnioski:

Obliczenie wszystkich N elementów macierzy metodą eliminacji Gaussa jest O(n^3) (wyeliminowanie jednego elementu z jednego wiersza to N operacji, z każdego wiersza to N-1 operacji, a trzeba to zrobić dla N zmiennych, więc N*(N-1)*N można ograniczyć przez N^3).

Metoda ta jednak byłaby nieoptymalna dla tego rodzaju macierzy, gdyż macierz trójdiagonalna jest macierzą rzadką, ponieważ tylko 3 spośród N elementów w wierszu (w pierwszym i ostatnim wierszu 2 elementy) są niezerowe. Pozostałe elementy to 0 i działania na tych elementach a nawet przechowywanie tych elementów jest zbędne. Zatem istnieją algorytmy, które będą dużo bardziej optymalne dla tego rodzaju macierzy.

Takim algorytmem z pewnością jest algorytm Thomasa (inaczej algorytm przeganiania), który potrafi znaleźć rozwiązanie dla tego problemu w czasie O(n), zatem będzie on dużo „tańszy" w porównaniu do metody eliminacji Gaussa. Metoda ta jest stabilna tylko dla macierzy diagonalnie dominującej

$$|b_i| \geq |a_i| + |c_i| \qquad i = 0,1 \dots . N$$

Istnieją 2 wersje tej metody, w pierwszej tworzymy nowe wektory, ta metoda jest bardziej złożona ale pozwala zachować pierwotne wartości macierzy. W drugiej nadpisujemy stare wartości nowymi. Ja wybrałam pierwszą metodę, w której zapisałam układ równań w następującej postaci:

$$\textbf{1)}\ a_i y_{i-1} + b_i y_i + c_i y_{i+1} = f_i \qquad i = 0,1,2 \dots . N$$

$$\text{Dla } a_0 = 0 \text{ i } c_N = 0$$

Dodatkowo z warunków podanych w zadaniu:

$$b_0 = b_N = f_N = 1 \qquad c_0 = a_N = 0 \qquad f_{0:N-1} = 0$$

$$b_n = -\frac{2}{h^2} \qquad a_n = c_n = \frac{1}{h^2} \qquad h = \frac{1}{N}$$

Ponieważ prawie wszystkie wartości są takie same, optymalniej byłoby pracować na zmiennych niż na całych tablicach złożonych z takich samych elementów. Ja jednak wybrałam operowanie na tablicach N elementowych a, b, c, f i y (5 tablic + potem 2 pomocnicze), gdyż wówczas problem będzie rozwiązany ogólnie dla każdego a_n, b_n, c_n i f_n a nie tylko dla wyszczególnionych w zadaniu i będą mogły ulec zmianie a algorytm dalej będzie działał.

Błędy: Metoda Thomasa jest metodą dokładną. Oznacza to, że po skończonej liczbie działań arytmetycznych na współczynnikach układu równań otrzymujemy rozwiązanie tego układu. Nie oznacza to, że błędów nie będzie, rozwiązanie jest obarczone błędami wynikającymi z zaokrąglania liczb wyników działań pośrednich. Zawsze też może pojawić się błąd ludzki.

Rozwiązania tego układu równań poszukuje się w postaci

$$y_i = \gamma_i y_{i+1} + \beta_i$$

Z równania **1)** wyznaczamy wzory na $\beta\ i\ \gamma$

$$\gamma_i = \frac{c_i}{-a_i\gamma_{i-1} + b_i}$$

$$\beta_i = \frac{f_i - a_i\beta_{i-1}}{-a_i\gamma_{i-1} + b_i}$$

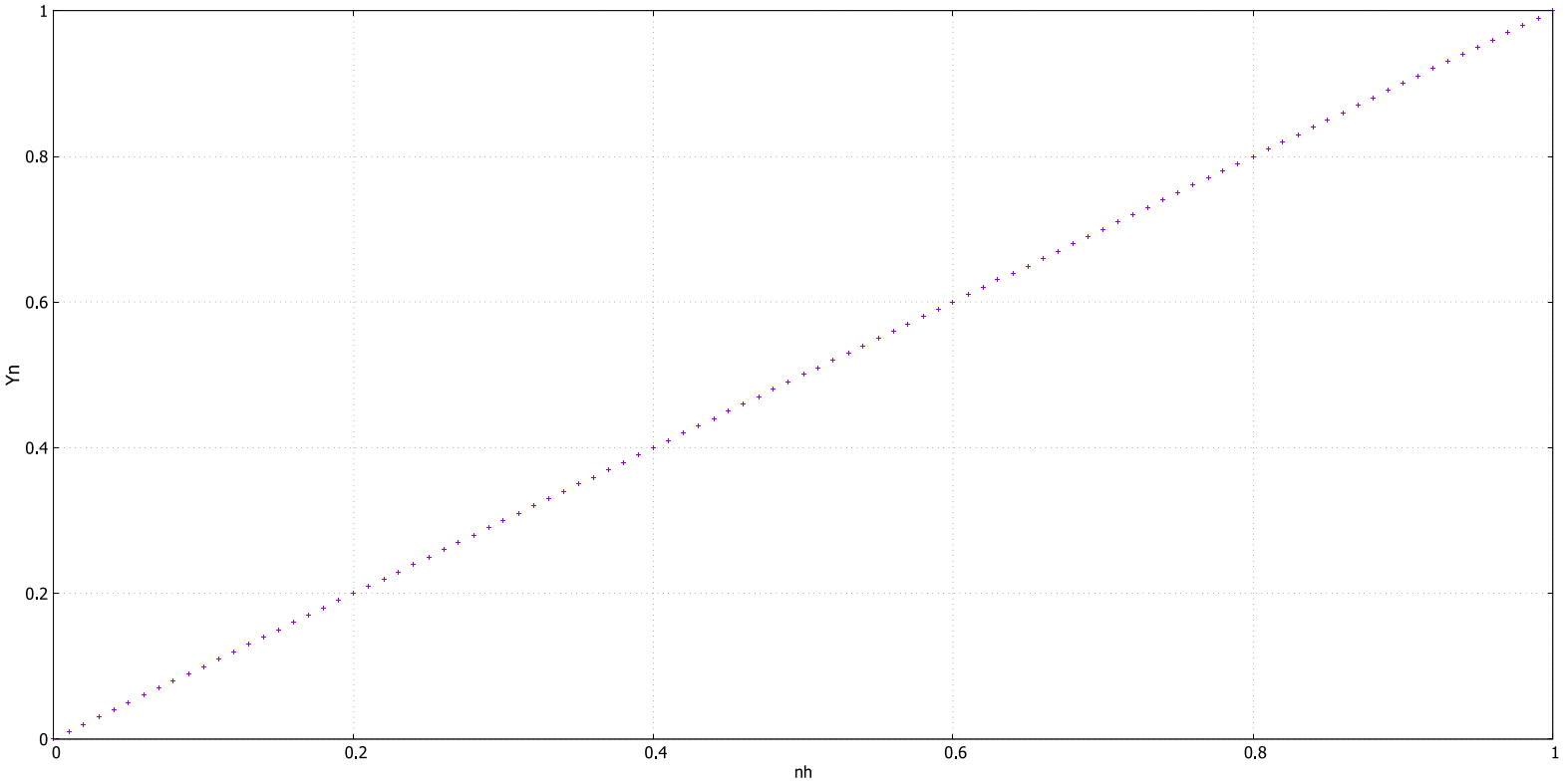
Oraz wartości początkowe.

$$\gamma_1 = \frac{c_1}{b_1}$$

$$\beta_1 = \frac{f_1}{b_1}$$

Stworzyłam więc dwie nowe tablice (wektory) $\beta\ i\ \gamma$, nadałam im wartości początkowe a następnie w pętli obliczyłam wszystkie elementy γ i β od i=0 aż do N (wykonałam fazę eliminacji wprzód) i dzięki temu uzyskałam ostatnie równanie i=N z którego mogłam wyznaczyć już y_N.
Dzięki wyznaczonej ostatniej niewiadomej mogłam policzyć przedostatnią i poprzednią i tak aż do y_1 (faza eliminacji wstecz).

Następnie zapisałam dane do pliku tekstowego i z niego w gnuplocie wygenerowałam wykres $(n \cdot \frac{1}{N}, y_n)$ prezentujący rozwiązanie graficzne układu równań dla h=1/N czyli 0.001



wykres $(n \cdot h, y_n)$

Aby uruchomić kod programu należy przejść do katalogu w którym znajdują się pliki i wpisać w terminalu komendę:
-> **make all**

A następnie:

->**make run**

Aby pozbyć się plików tymczasowych należy użyć komendy:
-> **make clean**

Program wygeneruje plik tekstowy thomas.dat w którym mieszczą się rozwiązania, a z których ja stworzyłam wykres w gnuplot metodą: plot ”thomas.dat”

Kod źródłowy programu:

```
#define N 100

#include <stdio.h>
#include <stdlib.h>



int main(){
    FILE *plik;
    if((plik=fopen("thomas.dat","w"))==NULL) return -1;
    double   n=N;
    double   h=1/n;
    double   dzielnik=0;
    double   a[N+1];
             a[0]=0; a[N]=0;
    double   b[N+1];
             b[0]=1;b[N]=1;
    double   c[N]; c[0]=0;
    double   f[N+1]; f[0]=0;
             f[N]=1;
    double   y[N+1];
             for(int i=1;i<=N-1;++i){

                  a[i]=1/(h*h);
                  b[i]=-2/(h*h);
                  c[i]=1/(h*h);
                  f[i]=0;
                }




    double lambda[N],beta[N+1];



    lambda[0]=c[0]/b[0];
    beta[0]=f[0]/b[0];
  //   printf("B[%d]=%.3f,Y[%d]=%.3f\n",0,lambda[0],0,beta[0]);
    for(int i=1;i<=N-1;++i){
```

```
      dzielnik=b[i]-(a[i]*lambda[i-1]);
      lambda[i]=(c[i])/dzielnik;

      beta[i]=(f[i]-a[i]*beta[i-1])/dzielnik;
  //    printf("B[%d]=%.3f,Y[%d]=%.3f\n",i,lambda[i],i,beta[i]);

    }

       dzielnik=b[N]-(a[N]*lambda[N-1]);
       beta[N]=(f[N]-a[N]*beta[N-1])/dzielnik;

 y[N]=beta[N];

     printf("Ostatnia niewiadoma: %f\n",y[N]);
        fprintf(plik,"#n*h y[n] \n");
        fprintf(plik, "%f %f\n",n*h,y[N]);


    for(int i=N-1;i>=0;--i){

        y[i]=beta[i]-(lambda[i]*y[i+1]);

     printf("Poprzednia to: %f\n",y[i]);
        fprintf(plik,"%f %f\n",i*h,y[i]);
    }
        printf("Dane zapisane w pliku thomas.dat\n");
   fclose(plik);

    return 0;
}
```

A po wykonaniu tworzy się plik tekstowy:

#n*h y[n]

1.000000 1.000000

0.990000 0.990000

0.980000 0.980000

0.970000 0.970000

0.960000 0.960000

0.950000 0.950000

0.940000 0.940000

0.930000 0.930000

0.920000 0.920000

0.910000 0.910000

0.900000 0.900000

0.890000 0.890000

0.880000 0.880000

0.870000 0.870000

0.860000 0.860000

0.850000 0.850000

0.840000 0.840000

0.830000 0.830000

0.820000 0.820000

0.810000 0.810000

0.800000 0.800000

0.790000 0.790000

0.780000 0.780000

0.770000 0.770000

0.760000 0.760000

0.750000 0.750000

0.740000 0.740000

0.730000 0.730000

0.720000 0.720000

0.710000 0.710000

0.700000 0.700000

0.690000 0.690000

0.680000 0.680000

0.670000 0.670000

0.660000 0.660000

0.650000 0.650000

0.640000 0.640000

0.630000 0.630000

0.620000 0.620000

0.610000 0.610000

0.600000 0.600000

0.590000 0.590000

0.580000 0.580000

0.570000 0.570000

0.560000 0.560000

0.550000 0.550000

0.540000 0.540000

0.530000 0.530000

0.520000 0.520000

0.510000 0.510000

0.500000 0.500000

0.490000 0.490000

0.480000 0.480000

0.470000 0.470000

0.460000 0.460000

0.450000 0.450000

0.440000 0.440000

0.430000 0.430000

0.420000 0.420000

0.410000 0.410000

0.400000 0.400000

0.390000 0.390000

0.380000 0.380000

0.370000 0.370000

0.360000 0.360000

0.350000 0.350000

0.340000 0.340000

0.330000 0.330000

0.320000 0.320000

0.310000 0.310000

0.300000 0.300000

0.290000 0.290000

0.280000 0.280000

0.270000 0.270000

0.260000 0.260000

0.250000 0.250000

0.240000 0.240000

0.230000 0.230000

0.220000 0.220000

0.210000 0.210000

0.200000 0.200000

0.190000 0.190000

0.180000 0.180000

0.170000 0.170000

0.160000 0.160000

0.150000 0.150000

0.140000 0.140000

0.130000 0.130000

0.120000 0.120000

0.110000 0.110000

0.100000 0.100000

0.090000 0.090000

0.080000 0.080000

0.070000 0.070000

0.060000 0.060000

0.050000 0.050000

0.040000 0.040000

0.030000 0.030000

0.020000 0.020000

0.010000 0.010000

0.000000 0.000000