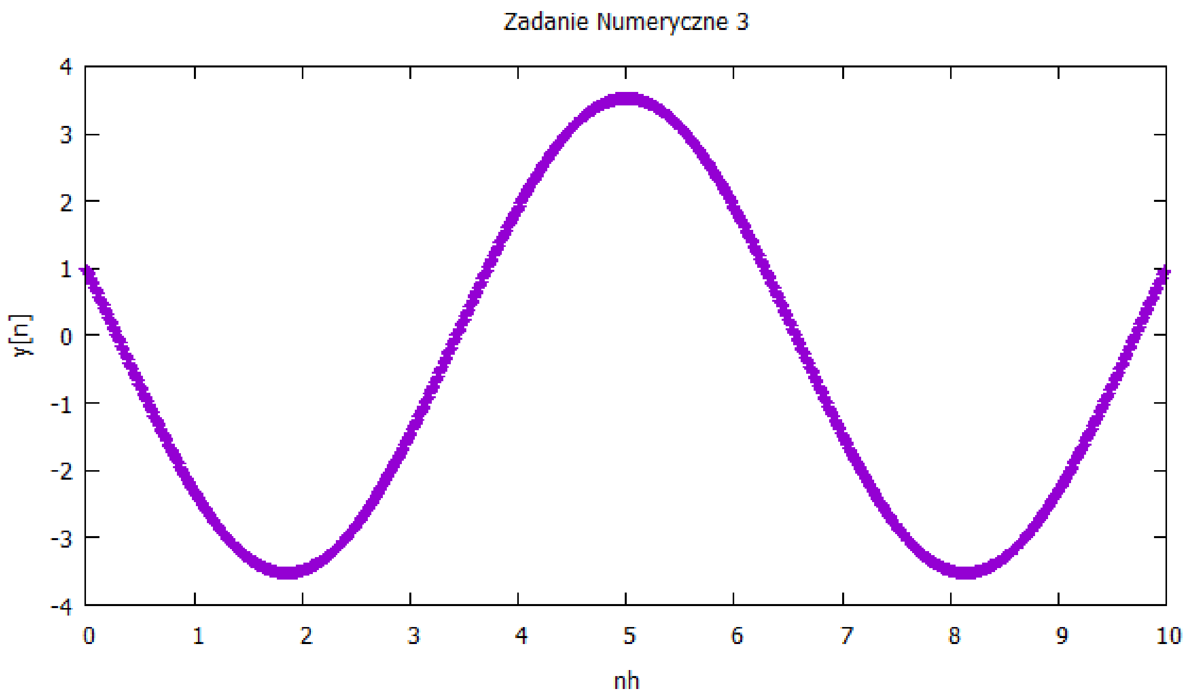
Zadanie Numeryczne 3

Zadanie polegało na rozwiązaniu układu z macierzą trójdiagonalną z dodatkową wartością w dolnym rogu. Do rozwiązania tego zadania wykorzystałem wzór Shermana–Morrisona. Algorytm polega na rozwiązaniu dwóch równań $Ay = b$ i $Az = u$ z ta samą macierzą trójdiagonalną A bez elementu w rogu macierzy. A następie obliczenie $x = y - \frac{v*y}{1+v*z} * z$. Równania potrafimy te szybko rozwiązać za pomocą algorytmu Thomasa.

Wyniki przedstawione graficznie (nh, y_n)



Używając wzoru Shermana–Morrisona definiujemy macierz trójdiagonalną oraz korekcję w postaci dwóch wektorów u, v odpowiedzialnych za elementy w rogach macierzy.

$$u = \begin{bmatrix} \gamma \\ 0 \\ \vdots \\ 0 \\ \alpha \end{bmatrix} \qquad v = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ \beta/\gamma \end{bmatrix}$$

Macierz A jest trójdiagonalną macierzą, $\gamma = -b_0$, $\alpha \; i \; \beta$ to wartości w rogach macierzy odpowiednio w dolnym i górnym. Algorytm polega na podwójnym wykorzystaniu algorytmu Thomasa o złożoności $O(n)$, aby rozwiązać równania. Dzięki zastosowaniu tablic zamiast wektorów udało się uniknąć czasochłonnych operacji push_back. Do potęgowania nie używam funkcji „pow" tylko wielokrotnego mnożenia. Do wypisania wyników nie używam „cout" tylko „printf". Dzięki tym zabiegom udało się uzyskać lepsza złożoność obliczeniową.

```cpp
#include <iostream>
#include <stdio.h>
#include <math.h>
#include <fstream>

using namespace std;
const double h = 0.01;
const int N = 1000;
const double alfa = 1;
const double beta = 0;

int main()
{
	double diagonalaA[N];
	double diagonalaB[N];
	double diagonalaC[N];
	double r[N];
	double d[N];
	double d1[N];
	double c[N];
	double x[N];
	double x1[N];

	//tablice i zmienne wykorzystane w metodzie Shermana-Morisona
	double bb[N];
	double u[N];
	double z[N];
	double f;
	double gamma;

	//wypelnienie diagonali a
	for (int i = 1; i <= N - 1; i++)
	{
		diagonalaA[i] = 1.0;
	}

	diagonalaA[0] = 0;

	//wypelnienie diagonali b
	diagonalaB[0] = 1;

	for (int i = 1; i < N - 1; i++)
	{
		diagonalaB[i] = (-2.0 + h * h);
	}

	diagonalaB[N - 1] = -2;

	//wypelnienie diagonali c

	for (int i = 1; i < N - 1; i++)
	{
		diagonalaC[i] = 1.0;
	}

	diagonalaC[0] = 0;
	diagonalaC[N - 1] = 0;

	//wypelniene wektora F

	r[0] = 1.0;
	for (int i = 1; i < N; i++)
```

```
        {
                r[i] = 0;
        }

        //Sherman-Morison
        gamma = -diagonalaB[0];
        bb[0] = diagonalaB[0] - gamma;
        bb[N - 1] = diagonalaB[N - 1] - (alfa * beta) / gamma;

        for (int i = 1; i < N - 1; i++)
        {
                bb[i] = diagonalaB[i];
        }
        for (int i = 0; i < N; i++) {
        }

        // A * x = r
        c[0] = diagonalaC[0] / bb[0];

        for (int i = 1; i <= N - 1; i++)
        {
                c[i] = diagonalaC[i] / (bb[i] - (diagonalaA[i] * c[i - 1]));
        }

        d[0] = r[0] / bb[0];

        for (int i = 1; i <= N - 1; i++)
        {
                d[i] = (r[i] - (diagonalaA[i] * d[i - 1])) / (bb[i] - (diagonalaA[i] *
c[i - 1]));
        }

        x[N - 1] = d[N - 1] / N * N;


        for (int i = N - 2; i >= 0; i--)
        {
                x[i] = (d[i] - (c[i] * x[i + 1])) / N * N;
        }

        u[0] = gamma;
        u[N - 1] = alfa;

        for (int i = 1; i < N - 1; i++)
        {
                u[i] = 0;
        }

        // A * z = u

        d1[0] = u[0] / bb[0];

        for (int i = 1; i <= N - 1; i++)
        {
                d1[i] = (u[i] - (diagonalaA[i] * d1[i - 1])) / (bb[i] - (diagonalaA[i] *
c[i - 1]));
        }

        z[N - 1] = d1[N - 1] / N * N;

        for (int i = N - 2; i >= 0; i--)
        {
```

```
        z[i] = (d1[i] - (c[i] * z[i + 1])) / N * N;
    }

    f = (x[0] + beta * z[N - 1] / gamma) / (1.0 + z[0] + beta * z[N - 1] / gamma);

    for (int i = 0; i <= N - 1; i++) {
        x1[i] = x[i] - f * z[i];
    }

    for (int i = 0; i <= N - 1; i++) {
        printf("%f \n", x1[i]);
    }
    /*
    // zapis danych do pliku aby wygenerowac wykres
        fstream plik;
         plik.open("ZN3.txt", ios::out);

         if(plik.good()==true)
         {
                for(int n=0; n<=N-1; n++)
                {
                       plik <<n*h << " " << x1[n] << endl;
                }
         }  else
                {
                       printf ("Nie moge otworzyc pliku ZN2.txt do zapisu!\n");
                       exit(1);
                }



         plik.close();
         */
    return 0;
}
```