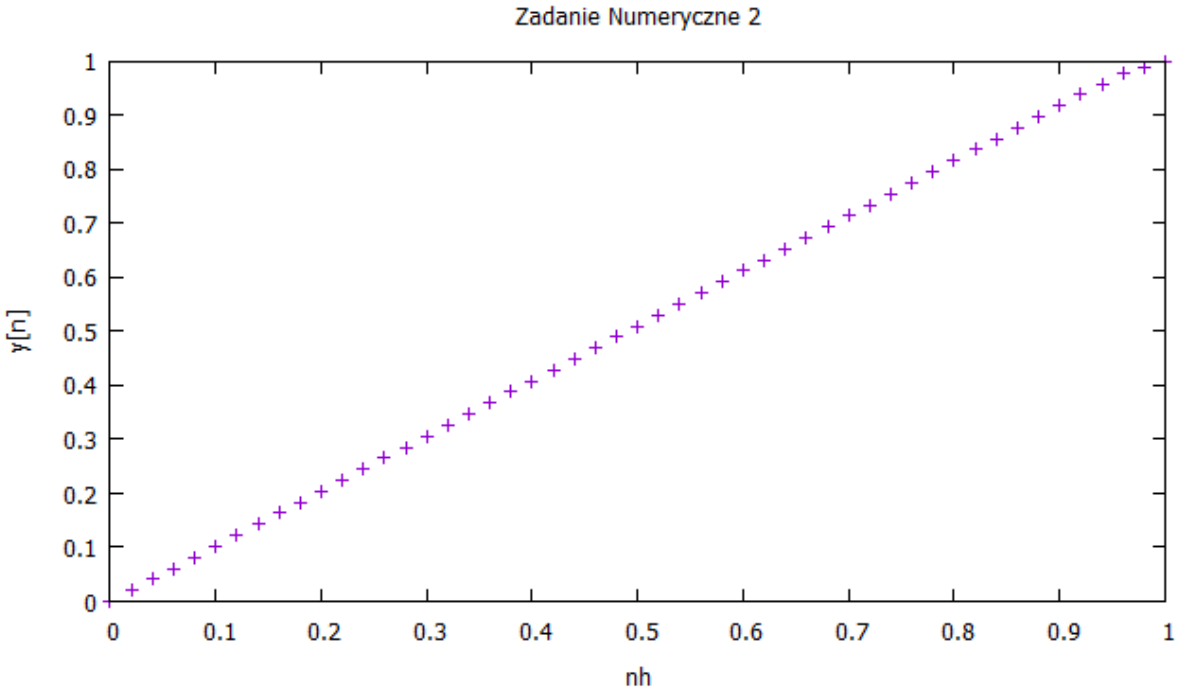
Zadanie Numeryczne 2

Zadanie polegało na rozwiązaniu układu z macierzą trójdiagonalną. Do rozwiązania tego zadania zaimplementowałem algorytm Thomasa, ponieważ jego złożoność obliczeniowa wynosi $O(n)$ w przeciwieństwie do eliminacji Gaussa, którego eliminacja złożoność obliczeniowa wynosi $O(n^3)$. Na początku w programie inicjuje tablice, które będą odpowiadały trzem diagonalom, wektorowi f oraz y. Uzupełniając diagonale o ich wartości wyłączyłem przed macierz wartość $1/h^2$ aby ograniczyć ilość dzieleń i usprawnić obliczenia. Następnie obliczam współczynniki d i c potrzebne to wykonania algorytmu Thomasa. Wykorzystując back substitution oraz mnożąc przez wyłączoną wcześniej wartość obliczam rozwiązania.

Wyniki przedstawione graficznie (nh, y_n)



Wykorzystanie algorytmu Thomasa pozwala uzyskać rozwiązanie w czasie $O(n)$. 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.

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

usingnamespace std;
intconst N = 51;

int main()
{
	double diagonalaA[N];
	double diagonalaB[N];
	double diagonalaC[N];
	double F[N];
	double d[N];
	double c[N];
	double y[N];

	//wypelnienie diagonali a

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

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

	//wypelnienie diagonali b

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

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

	//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

	for (int i = 0; i <= N - 1; i++)
	{
		if (i == N - 1)
			F[i] = 1.0;
		else
			F[i] = 0;
	}

	c[0] = diagonalaC[0] / diagonalaB[0];

	for (int i = 1; i <= N - 1; i++)
	{
```

```
		c[i] = diagonalaC[i] / (diagonalaB[i] - (diagonalaA[i] * c[i - 1]));
	}

	d[0] = 0;

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

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

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

	// Wypisanie wynikow na ekran

	for (int i = 0; i <= N - 1; i++)
	{
		printf("%f \n", y[i]);
	}

	// zapis danych do pliku aby wygenerowac wykres
	/*   fstream plik;
		plik.open("ZN2.txt", ios::out);

		if(plik.good()==true)
		{
			for(int n=0; n<=N-1; n++)
			{
				plik << (1/(double)N)*n << " " << y[n] <<endl;
			}
		}  else
			{
				printf ("Nie mogę otworzyc pliku ZN2.txt do zapisu!\n");
				exit(1);
			}
	plik.close();
	  */
	return 0;
}
```