**Informatyka II**

Temat: Metoda eliminacji Gaussa

Autor: Adam Walocha

Numer indeksu: 304386

Kierunek: Energetyka

Nr grupy dziekańskiej: 9

Prowadzący: Michał Stachura

Zadanie 1: macierz Hilberta

Rozwiąż układ równań , w którym jest macierzą Hilberta o elementach , to szukane rozwiązanie, to wektor prawych stron, w którym elementy to sumy . Taki układ ma analityczne rozwiązanie, jakie ono jest?

Ćwiczenia:

1. Zaalokuj dynamicznie pamięć na dwuwymiarową tablicę o wymiarze w celu przechowywania elementów macierzy Hilberta.
2. Zaalokuj dynamicznie pamięć na wektory oraz .
3. Napisz funkcję która tworzy macierz Hilberta .
4. Napisz funkcję która drukuje macierz na ekranie.
5. Napisz funkcję która oblicza wektor prawych stron oraz funkcję która drukuje wektor .
6. Zastosuj funkcję realizującą metodę eliminacji Gaussa do rozwiązania układu równań .
7. Przetestuj działanie programu dla różnych wartości .
8. Jak zmiana typu zmiennych wpływa na dokładność obliczeń?

#include <iostream>

#include <cstdlib>

#include <math.h>

#include "gauss.h"

void HilbertMatrix(int N, double\*\* H);

void displayMatrix(int N, double\*\* H);

void computeVec(int N, double\*\* H, double\* b);

void plotVec(int N, double\* b);

int main()

{

int N; double\*\* H; double\* x; double\* b; printf("N="); scanf\_s("%d", &N);

H = (double\*\*)malloc(N \* sizeof(double\*));

for (int i = 0; i < N; i++)

{

H[i] = (double\*)malloc(N \* sizeof(double));

}//dynamiczna alokacja pamieci na tablice(macierz) H o wymiarach NxN

x = (double\*)malloc(N \* sizeof(double));//macierz niewiadomych

b = (double\*)malloc(N \* sizeof(double));//macierz prawych stron

HilbertMatrix(N, H);

displayMatrix(N, H);

computeVec(N, H, b);

plotVec(N, b);

gauss(N, H, x, b);

for (int i = 0; i < N; i++)

{

printf("%lf\n", x[i]);

}

free(H); free(x); free(b);

}

void HilbertMatrix(int N, double\*\* H)//tworzenie macierzy H

{

for (int i = 0; i < N; i++)

{

for (int j = 0; j < N; j++)

{

H[i][j] = (1. / (1 + i + j));

}

}

}

void displayMatrix(int N, double\*\* H)//wypisywanie macierzy H na ekran

{

for (int i = 0; i < N; i++)

{

for (int j = 0; j < N; j++)

{

printf("%lf\t", H[i][j]);

}

printf("\n");

}

printf("\n");

}

void computeVec(int N, double\*\* H, double\* b)//obliczanie wektora b

{

for (int i = 0; i < N; i++)

{

b[i] = 0;

}

for (int i = 0; i < N; i++)

{

for (int j = 0; j < N; j++)

{

b[i] += H[i][j];

}

}

}

void plotVec(int N, double\* b)//wypisywanie wektora b na ekran

{

for (int i = 0; i < N; i++)

{

printf("%lf\n", b[i]);

}

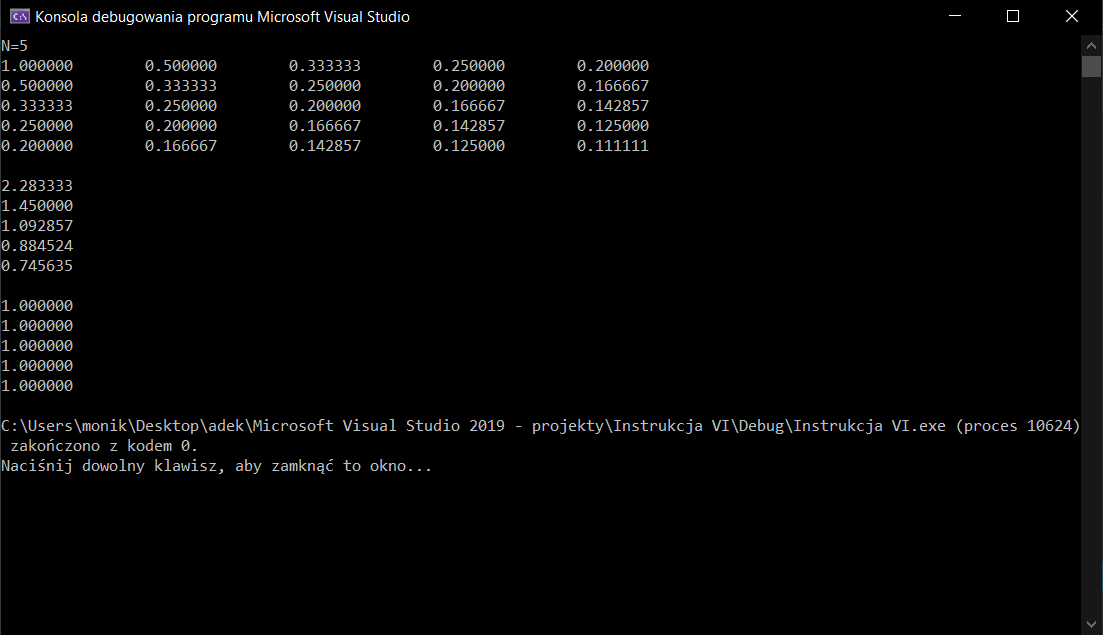
printf("\n");

}

Rozwiązanie:

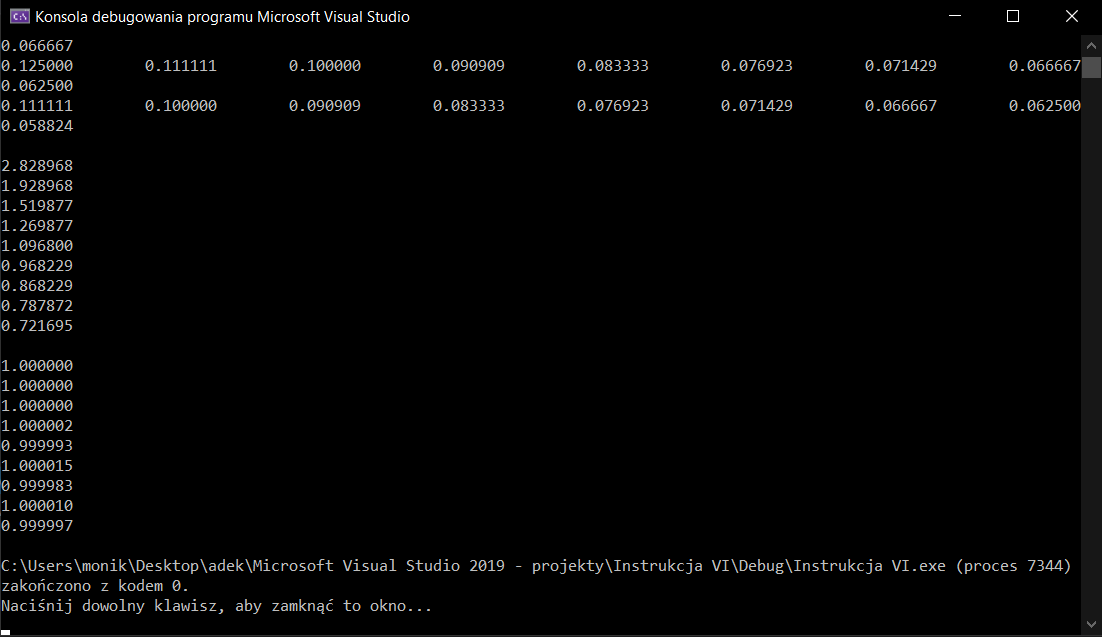
Analitycznym rozwiązaniem zagadnienia jest wektor składający się kolejno z elementów gdzie wszystkie wartości są równe

Program drukuje dla zadanej wcześniej wartości kolejno: macierz Hilberta , wektor , oraz rozwiązanie układu równań .



*Rys. 1. Przykładowe rozwiązanie macierzy Hilberta dla N=5*

Dla jak przedstawiono na zdjęciu powyżej rozwiązanie macierzy Hilberta jest równe z rozwiązaniem analitycznym aż do szóstego miejsca po przecinku. Przy większym wymiarze macierzy Hilberta rozwiązania odbiegają od wyniku uzyskanego metodą analityczną.



*Rys. 2.Rozwiązanie macierzy Hilberta dla N=9. Można zauważyć że rozwiązanie odbiega od rozwiązania otrzymanego metodą analityczną*

Rozwiązania otrzymane za pomogą programu zgadzają się do . Jak pokazano na zdjęciu powyżej dla wyższych wartości rozwiązania uzyskane metodą eliminacji Gaussa odbiegają od prawidłowego wyniku. Związane jest to z faktem, że zmienna typu *double* przechowuje 8 bajtów informacji. Używając innego rodzaju zmiennych, takich które mogą przechować dokładniejszą wartość, jak np. *long double* można uzyskać dokładny wynik dla większej wartości parametru .

Zmieniając typ zmiennych na np. *int* nie uzyskamy nic, wszystkie parametry macierzy poza będą przetrzymywały wartość 0. Oznacza to że wyznacznik macierzy uzyskanej przy wykorzystaniu zmiennej typu *int* będzie równy 0, co przekłada się na fakt iż wyznacznik tej macierzy również będzie równy 0. W takim przypadku nie istnieje macierz odwrotna do macierzy wyjściowej. Nie można użyć metody eliminacji Gaussa.

Podsumowując, dobór typu zmiennych ma duże znaczenie dla dokładności obliczeń. Macierz Hilberta jest macierzą, której uwarunkowanie rośnie wyjątkowo szybko wraz z wymiarem. Za pomocą metod numerycznych można uzyskać prawidłowe rozwiązanie dla macierzy o maksymalnym rozmiarze 8. Dla kolejnych wartości zmiennej można zaobserwować coraz większe odstępstwo od prawidłowego wyniku.