METODA ELIMINACJI GAUSSA

LABORATORIUM NR.6

JAKUB MROCZKOWSKI

GR.9

NR. INDEKSU 304336

1.Temat zajęć

Celem laboratorium numer 6 było ukazanie jak uwarunkowanie układu wpływa na numeryczną poprawność eliminacji Gaussa za pomocą macierzy Hilberta oraz sposób w jaki można wykorzystać eliminację Gaussa do podstawowych problemów fizycznych w tym przypadku do rozwiązania i ukazania rozkładu temperatury w pręcie.

2.Metody

Do wykonania laboratorium potrzebna była wiedza z zakresu metody eliminacji Gaussa. Mianowicie na początku należy zapisać układ równań w postaci macierzowej A*x=b, gdzie A to macierz układu równań, x to wektor niewiadomych, a b to wektor prawych stron układu równań. Dzięki metodzie różnic skończonych, która w skrócie polega na przybliżeniu pochodnej funkcji poprzez skończone różnice możemy przejść z równania różniczkowego do dużego układu liniowych równań algebraicznych.

3. Zadania

Zadanie polegające na wyznaczeniu rozkładu temperatury w pręcie. Wymagało zastosowania metody różnic skończonych, a wszystkie potrzebne zależności zostały podane w instrukcji laboratorium, gdzie ostateczna postać naszego równania wyglądała następująco K*T=F, gdzie K to macierz współczynników, T to wektor, w którym zapisane są uzyskane

temperatury, czyli rozwiązania , a F to wektor prawych stron, do którego uzupełnienia potrzebny był wyprowadzony wzór.

Zadanie z macierzą Hilberta polegało na odpowiednim zapisaniu macierzy NxN i uzupełnieniu odpowiednimi wartościami miejsc w tejże macierzy, po czym skorzystaniu z funkcji Gauss.

4. Kod wraz z opisem , zadanie numer 1:

```
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 *v);

void main()

{
            int N; Liczba równań oraz wymiar macierzy Hilberta
            printf("Prosze podac wartosc N:");
            scanf("%d" , &N);
            double **H;
            double *x;
            double *b;

            x = (double*)malloc(N*sizeof(double));Dynamiczna alokacja
wektora x,czyli szukanego rozwiązania oraz wektora b, czyli
wektora prawych stron oraz H,czyli tablicy na macierz
Hilberta.
            b = (double*)malloc(N*sizeof(double));

            H = (double**)malloc(N*sizeof(double*));
            for(int i=0; i <N ; i++)
            H[i] = (double*)malloc(N*sizeof(double));
            HilbertMatrix(N,H);Używając po kolei stworzonych funkcji
tworzę macierz Hilbera, wyświetlam ją, obliczam wektor prawych
stron, rozwiązuję równanie H*x=b za pomocą funkcji gauss,
zaczerpniętej ze strony naszych zajęć,oraz wyświetlam
rozwiązania.
            displayMatrix(N,H);
            computeVec(N,H,b);
            plotVec(N,b);
            gauss(N,H,x,b);
            printf("\n\n\n");
            plotVec(N,x);

            system("PAUSE");

}
```

```
void HilbertMatrix(int N , double **H) Funkcja stworzona na potrzebę uzupełnienia w odpowiedni sposób macierzy oraz oczywiście jej stworzenia

{
            for(int i = 0 ; i <N ; i++)  Podwójna pętla służąca do zapisu wartości do poszczególnych komórek tablicy
            {
                  for(int j=0 ; j <N ; j++)
                  {
                        H[i][j] = 1.0/(j+i+1.0);
                  }
            }

}


void displayMatrix(int N , double **H)

{
      printf("Macierz Hilberta:\n");     Funkcja displayMatrix służąca do wyświetlenia macierzy Hilberta.
      for(int i= 0 ; i <N ; i++)
      {
            for(int j = 0 ; j <N; j++)
            {
                  printf("%lf " , H[i][j]);
            }
            printf("\n");
      }
}


void computeVec(int N, double **H , double *b) Funkcja computeVec służąca do obliczenia wektora prawych stron, zapisanego odpowiednim wzorem.

{

            for(int i = 0 ; i <N ; i++)
            {
                  double s = 0.0;
                  for(int j = 0 ; j <N ;j++)
                  {
                        s = s + H[i][j];
                  }
                  b[i] = s;
            }
}

void plotVec(int N , double *v)

{
      for(int i= 0; i <N ;i++)     Funkcja drukująca na ekran zadeklarowany wektor.
      {
            printf("%lf\n ", v[i]);
```

```
	}
}
```

Kod do zadania numer 2 oraz 3:

```
void computeMatrix(int N , double **K);
void displayMatrix(int N , double **K);
void computeVector(int N, double *F,double *x);
void displayVector(int N , double *F);


void main()


{
	FILE* fp;
	int N;
	printf("Prosze podac wartosc N:");
	scanf("%d", &N);
	double h= 1.0/N; N to zadeklarowana ilość odcinków na jakie chcemy podzielić funkcję, h to odległość pomiędzy kolejnymi punktami.
	double **K;
	double *T;
	double *F;
	double *x;

	K = (double**)malloc((N+1)*sizeof(double*));Deklaruję odpowiednio wsztyskie potrzebne macierze,macierz K zeruje, żeby wpisać tylko pierwszy i ostatni element, a 1,2,1 odpowiednio przesunąć.
	for(int i = 0 ; i <=N ;i++)
	K[i] = (double*)malloc((N+1)*sizeof(double));
	for(int i = 0 ; i <=N;i++)
	{
		for(int j = 0 ; j<=N ;j++)
		{
			K[i][j] =0;
		}
	}
	F = (double*)malloc((N+1)*sizeof(double));
	x = (double*)malloc((N+1)*sizeof(double));
	T = (double*)malloc((N+1)*sizeof(double));

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

	{
		x[i]=h*i; Dzięki tej operacji oblicze kolejne x, które zapisałem w wektorze, odpowiednio przesuwając się o wartość h, czyli odległość pomiędzy każdym z punktów.
	}
```

```
	computeMatrix(N,K);
```
Aplikuję funkcję tworząca macierz współczynników.
```
	displayMatrix(N,K);
	computeVector(N,F,x);
```
Obliczam wektor prawych stron.
```
	displayVector(N,F);
	printf("\n\n\n");
	gauss((N+1),K,T,F);
```
Korzystając z funkcji gauss, obliczam moje rozwiązanie,czyli temperatury pamiętając, że liczba równań to N+1, do czego odnosiłem się już wczesniej zaczynając odpowiednio pętlę od 1 i kończąc włącznie z N.
```
	displayVector(N,T);
	fp = fopen("mojewyn.txt","w");

	for(int i = 0 ; i <=N;i++)
	{
		fprintf(fp, "%lf\t\t\t%lf\n" , x[i], T[i]);
	}
	fclose(fp);
	free(T);
```
Zapisuję potrzebne wyniki oraz uwalniam pamięć z poszczególnych tablic a także zamykam plik do zapisu.
```
	free(F);
	free(x);
	for(int i = 0; i<=N;i++)
	free(K[i]);
	free(K);
	system("PAUSE");
}

void computeMatrix(int N , double **K)
```
Funkcja tworząca macierz współczynników K.
```
{
	for(int i = 1 ;i <N ; i++)
	{
		K[i][i-1] = 1.0;
```
Pierwszy element macierzy ma wartość 1 oraz ostatni element macierzy ma wartość 1. Pozostałe wartości wpisuję do macierzy K odpowiednio przesuwając w iteracja wartości 1,-2,1.
```
		K[i][i] = -2.0;
		K[i][i+1] = 1.0;
	}
	K[0][0] = 1.0;
	K[N][N] = 1.0;
}

void displayMatrix(int N , double **K)

{
	for(int i = 0 ; i <=N;i++)
	{
		for(int j = 0 ; j <=N ; j++)
```
Funkcja wyświetlająca macierz współczynników.

```
            {
                  printf("%lf " , K[i][j]);

            }
            printf("\n");
      }
}

void computeVector(int N, double *F,double *x)

{
      double h = 1.0/N;
      double l = 58.0;

      for(int i =1;i<N;i++)
      {
            F[i] = -10000.0*sin(x[i]*3.14)/l*pow(h,2);
```

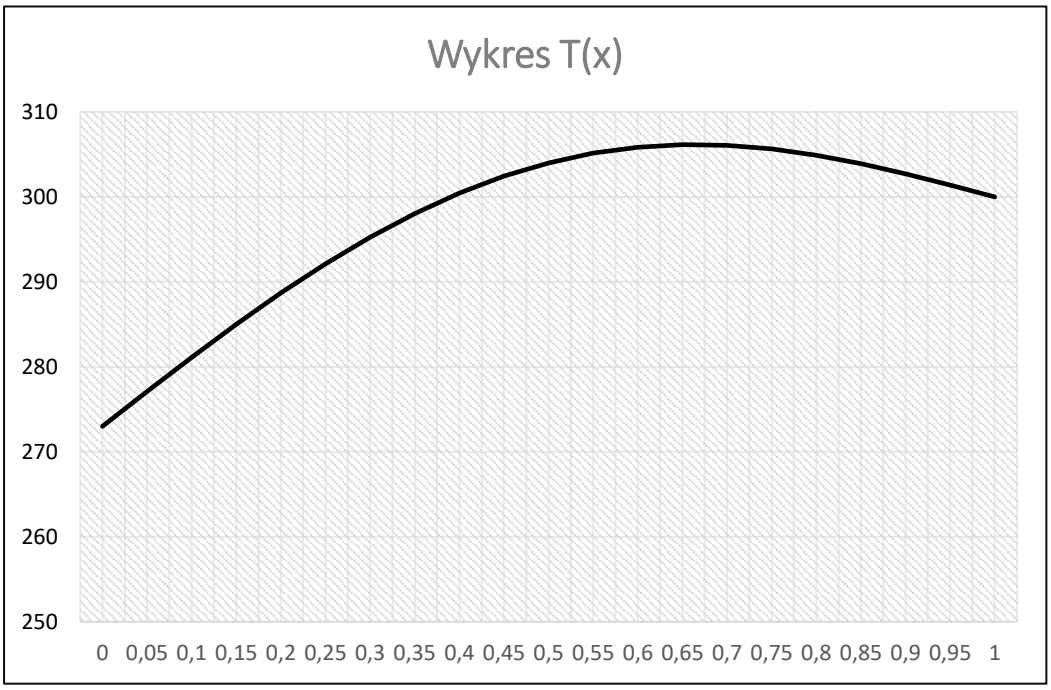
Funkcja computeVector oblicza wektor prawych stron równania. W pętli korzystam z odpowidniego wzoru podanego na stronie, natomiast pierwszy i ostatni elemnt wektora uzupełniam wartościami podanymi 273K oraz 300K.

```
      }
      F[0]=273.0;
      F[N]=300.0;
}

void displayVector(int N , double *F)
{
      for(int i = 0 ; i <=N;i++)
```

Funkcja wyświetlająca wektor prawych stron równania.

```
      {
            printf("%lf\n" , F[i]);
      }
}
```



Kolejnym etapem zadania było stworzenie wykresu tempratury dla N = 20 od x i sprawdzenie, czy wykres wyszedł faktycznie podobny do podanego w instrukcji patrząc na uzyskany wykres jest on bardzo zbliżony niestety nie mogłem stworzyć go w kodzie, ponieważ funkcja scale nie chciała zostać wykryta w pobranej bibliotece winbgi2.

5. Podsumowanie

Podczas wykonaniu laboratorium numer 6 poznałem od strony praktycznej wykorzystanie metody eliminacji Gaussa. Z pewnością nauczyłem się w jaki sposób należy w programie zapisywać macierze i wektory za pomocą tablic i przede wszystkim w jaki sposób wykorzystać je do użycia funkcji gauss. Wyniki z wykresu pokryły się co wnioskując sugeruje o poprawności kodu, i możliwości testowania programu, dla różnych N. Sprawdzając na przykładzie macierzy Hilberta i równania Hx=b dla N około 20 wyniki były bardzo zbliżone, a dokładność była zachowana, przy wzrastających N dokładność obliczeń metodą Gaussa drastycznie malała.