Kocjan Nikodem

27.02.2024

Metody numeryczne ćwiczenie 1

Rozwiązywanie układu równań liniowych metodą eliminacji Gaussa

1. Cel ćwiczenia

Celem ćwiczenia było napisanie programu, który za pomocą metody Gaussa, będzie w stanie rozwiązać dowolną ilość równań liniowych. Program testowaliśmy na układzie 3 równań, co dokładniej zostanie opisane poniżej.

2. Wstęp teoretyczny

Metoda eliminacji Gaussa jest jedną z podstawowych metod rozwiązywania układów równań liniowych. Opiera się na 2 głównych krokach

I) Wyzerowanie komórek macierzy współczynników znajdujących się pod przekątną
II) Rozwiązywanie układu zaczynając od najniższego wiersza

Ad I)

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \qquad\qquad \ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m \end{cases}$$

Rozważamy układ m równań liniowych z n niewiadomymi. Tworzymy z nich macierz współczynników i łączymy ją z macierzą wyników. Otrzymujemy:

$$: \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ & & \cdots & & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{array}\right]$$

Następnie zerujemy wszystkie współczynniki znajdujące się poniżej przekątnej macierzy. Otrzymujemy:

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & \ldots & a_{1n} & b_1 \\ 0 & a_{22}^{\cdot} & \ldots & a_{2n}^{\cdot} & b_2^{\cdot} \\ & & \ldots & & \\ 0 & 0 & 0 & a_{mn}^{\cdot} & b_m^{\cdot} \end{array}\right]$$

Ad II)

Kolejny krok to przejście macierzy od najniższego wiersza (m) i obliczanie kolejnych wartości wyników od b_m do b_1, gdzie zmienna $x_n = \frac{b_m^{\cdot}}{a_{mn}^{\cdot}}$, potem wynik wstawiamy do wiersza $m-1$, aż dojdziemy do wiersza 1 i otrzymamy rozwiązanie każdej zmiennej

3. Deklaracja zmiennych oraz implementacja bibliotek

Metodę Gaussa testowaliśmy za pomocą kodu napisanego w języku C_{++}, używając do tego Visual Studio Code. Wystarczyła nam tylko standardowa biblioteka <$iostream$>

```
int i,j,k,n;

cout << "Podaj liczbe rownan: ";
cin >> n;

float mat[n][n+1];

float wynik[n];
```

Na początku kodu zadeklarowaliśmy zmienne $i,\ j,\ k,\ n$, gdzie i, j, k będą służyły do iteracji kolejnych pętli. Spytaliśmy użytkownika, ile równań wprowadzi, zadeklarowaliśmy macierz mat, która będzie macierzą współczynników, oraz tabelę $wynik$.

4. Opis działania

We wstępie powiedzieliśmy o tym, że dzielimy metodę eliminacji Gaussa na dwa kroki. W algorytmie sytuacja jest bardzo podobna.

Ad I)

```
float wsp;

for(i = 0; i < n - 1; i++){
  for(j = i+1; j < n; j++){
    wsp = mat[j][i] / mat[i][i];
    for(k = 0; k < n + 1; k++){
      mat[j][k] = mat[j][k] - wsp * mat[i][k];
    }
  }
}
```

Ta część programu jest odpowiedzialna za stworzenie macierzy górno-trójkątnej.
Pierwsza pętla for odpowiada za iterację po wszystkich wierszach macierzy. Kończy ona swoje działanie na wierszy n-2, ponieważ nie ma potrzeby operowania na ostatnim wierszu, gdyż znajduje się tam tylko współczynnik przekątnej, pozostałe są wyzerowane.
Pierwsza pętla wewnętrzna przechodzi przez kolejne wiersze macierzy, zaczynając poniżej przekątnej. Następnie obliczany jest współczynnik wsp, który mówi nam, co musimy odjąć, aby wyzerować odpowiednie komórki.
Trzecia pętla jest odpowiedzialna za odejmowanie odpowiedniej ilości wsp pomnożonej przez wartość odpowiadającego elementu wiersza i, co skutkuje eliminacją elementów poniżej głównej przekątnej macierzy.

Ad II)

```
for(i = n-1; i>=0;i--){
  wynik[i] = mat[i][n];
  for(j = i + 1; j < n; j++){
    if(i!=j){
      wynik[i] = wynik[i] - mat[i][j] * wynik[j];
    }
  }
  wynik[i] = wynik[i] / mat[i][i];

}
```

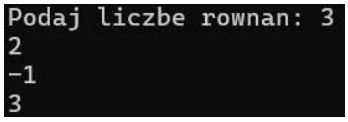
Ta pętla jest odpowiedzialna za obliczenie wartości niewiadomych układu równań na podstawie macierzy górno-trójkątnej, uzyskanej w poprzednim etapie.
Pierwsza pętla iteruje po wierszach w odwrotnej kolejności, rozpoczynając od ostatniego wiersza $(n-1)$. Na początku inicjalizujemy $wynik$ jako ostatnią kolumnę danego wiersza.
Pętla wewnętrzna odpowiada za wstawianie poznanych niewiadomych z poprzednich wyników to wyższych wierszy, a następnie obliczanie nowego współczynnika dla zmiennej odpowiadającej danemu wierszowi.
Na końcu nadpisujemy $wynik$ tak, aby otrzymać końcowe rozwiązanie. Tabela $wynik$ zawiera rozwiązania zmiennych od x_1 do x_n

5. Analiza wyników

Test przeprowadzaliśmy dla układu 3 równań.

$$-x + 2y + 1z = -1$$
$$x - 3y - 2z = -1$$
$$3x - y - z = 4$$

Wynik programu:



Co jest zgodne z stanem rzeczywistym.

6. Wnioski

Program poprawnie rozwiązuje układy równań liniowych metodą Gaussa. Bardzo zadowalające jest to, że nie potrzebujemy wykorzystywać dodatkowych bibliotek i jesteśmy w stanie rozwiązać go używając tylko standardowej biblioteki $< iostream >$

Musimy jednak zwrócić uwagę na to, że program nie jest odporny na pewien błąd krytyczny. Chodzi dokładniej o to, że nie sprawdzamy czy współczynniki macierzy głównej nie wynoszą 0. Program będzie wtedy próbował dzielić, co może skutkować różnymi problemami. Jest to kluczowe zwłaszcza w momencie obliczania współczynnika wsp.

Problem dzielenia przez zero może się również pojawić, gdy macierz będzie macierzą osobliwą - będzie posiadać nieskończenie wiele rozwiązać, lub żadnego.

Jednocześnie, jeżeli będziemy używać dokładnych liczb zmiennoprzecinkowych, program zaokrągli wyniki, co może skutkować niedokładnymi rezultatami. Jest to widoczne w momencie, gdy stosunek najmniejszej do największej wartości jest duży.

Podsumowując, program rozwiąże układy równań liniowych metodą Gaussa zakładając, że użytkownik, nie wprowadzi 0 jako współczynnika, nie wprowadzi macierzy osobliwej i że precyzja wyników będzie dla niego wystarczająca.