Metody numeryczne

Nikodem Kocjan

13.03.2024

Iteracyjne rozwiązywanie układu równań liniowych metodą Jacobiego.

Spis treści

1 Cel ćwiczenia

Celem ćwiczenia było zastosowanie metody Jakobiego do iteracyjnego rozwiązywania układu równań liniowych powstałego w wyniku dyskretyzacji równania różniczkowego drugiego rzędu. Równanie to opisuje ruch ciała poddanego działaniu siły sprężystej, siły tarcia zależnej od prędkości oraz siły wymuszającej ruch.

2 Wstęp teorytyczny

2.1 Podstawowe założenia

Naszym celem będzie rozwiązanie równania różniczkowego

$$\frac{d^2x}{dt^2} = -\omega^2 x - \beta V + F_0 \sin(\Omega t) \tag{1}$$

opisującego ruch ciała poddanego sile sprężystej:

$$-\omega^2 x \tag{2}$$

oraz sile tarcia:

$$-\beta V + F_0 \tag{3}$$

zależniej od prędkości oraz siły wymuszającej ruch:

$$F_0 \sin(\Omega t) \tag{4}$$

Wprowadzamy siatkę, której węzłami będą kolejne chwile czasowe:

$$t = t_i = i \cdot h, \quad i = 0, 1, 2, \ldots \tag{5}$$

2.2 Przekształcenie rozwiązania

Pochodną możemy zamienić na symetryczny trójkpunktowy iloraz różnicowy:

$$\frac{d^2x}{dt^2} \approx \frac{x_{i-1} - 2x_i + x_{i+1}}{h^2} \tag{6}$$

gdzie h będzie oznaczał krok czasowy na siatce. Pierwszą pochodną położenia w czasie jest prędkość, dlatego ją także zastępujemy ilorazem różnicowym:

$$V_i = \frac{x_{i+1} - x_i}{h} \tag{7}$$

Wyznaczoną w ten sposób prędkość wstawiamy do równania różniczkowego:

$$\frac{x_{i-1} - 2x_i + x_{i+1}}{h^2} = -\omega^2 x_i - \beta V_i + F_0 \sin(\Omega h i) \tag{8}$$

Przekształcamy równanie i otrzymujemy w rezultacie:

$$x_{i-1} - 2x_i + x_{i+1} + \omega^2 h^2 x_i + \beta h(x_{i+1} - x_i) = F_0 \sin(\Omega h i) h^2 \tag{9}$$

Symbolicznie możemy to zapisać jako:

$$a_1 x_{i-1} + a_2 x_i + a_3 x_{i+1} = b_i \tag{10}$$

gdzie:

$$a_1 = 1 \tag{11}$$

$$a_2 = \omega^2 h^2 - 2 \tag{12}$$

$$a_3 = 1 + \beta h, \tag{13}$$

$$b_i = F_0 \sin(\Omega h i) h^2 \tag{14}$$

2.3 Analiza równania macierzowego

Po zastosowaniu kroków z podpunktów 2.1 oraz 2.2 otrzymujemy układ równań Ax = B. Aby rozwiązać równanie różniczkowe drugiego rzędu musimy podać warunki początkowe:
wychylenie x(t=0) = $x_0 = 1$
prędkość początkowa V(t=0) = $V_0 = \frac{x_{i+1} - x_i}{h} = 0$
Finalnie otrzymujemy macierz:

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ -1 & 1 & 0 & \cdots & 0 & 0 \\ a_1 & a_2 & a_3 & \ddots & \vdots & \vdots \\ 0 & a_1 & a_2 & \ddots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 & 0 \\ 0 & \cdots & 0 & a_1 & a_2 & a_3 \\ 0 & \cdots & 0 & 0 & a_1 & a_2 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ b_2 \\ b_3 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}$$

2.4 Metoda Jacobiego

Metoda Jacobiego to klasyczna iteracyjna metoda numeryczna służąca do rozwiązywania układów równań liniowych, w których macierz współczynników jest dominująca po przekątnej. Jest to jedna z prostszych metod iteracyjnych, co czyni ją dobrym punktem wyjścia do zrozumienia tego typu algorytmów.

Układ równań liniowych postaci $Ax = b$, gdzie A jest macierzą współczynników, x jest wektorem niewiadomych, a b wektorem wyrazów wolnych, może być rozwiązany przy pomocy tej metody pod warunkiem, że żaden z elementów na głównej przekątnej macierzy A nie jest równy zero.

Metoda Jacobiego polega na iteracyjnym obliczaniu przybliżenia rozwiązania poprzez izolowanie każdej niewiadomej.

Proces iteracyjny jest powtarzany do momentu, aż różnica między kolejnymi przybliżeniami będzie dostatecznie mała, co jest określane przy pomocy wcześniej zdefiniowanego kryterium zbieżności.

Metoda ta jest szczególnie efektywna dla macierzy rzadkich, gdzie większość elementów poza przekątną jest równa zero. Jednak jej zbieżność może być wolna, zwłaszcza dla dużych układów równań, co spowodowało rozwój innych, bardziej zaawansowanych metod iteracyjnych, takich jak metoda Gaussa-Seidela czy metody gradientów sprzężonych.

W naszym przypadku macież układu równań jest trójprzekątniowa, więc możemy ją przechowywać w postaci trzech wektorów:

$$d0 = [1, 1, a_3, a_3, \ldots, a_3] \tag{15}$$

$$d1 = [0, -1, a_2, a_2, \ldots, a_2] \tag{16}$$

$$d2 = [0, 0, a_1, a_1, \ldots, a_1] \tag{17}$$

Następnie z ich użyciem możemy obliczyć

$$x_n[i] = \frac{1}{d0[i]} \left(b[i] - d1[i]x[i-1] - d2[i]x[i-2] \right) \tag{18}$$

dla każdego i = 0,1,2,...,n

3 Deklaracja zmiennych oraz implementacja bibliotek

Opisane zagadnienie rozwiązywałem za pomocą programu Visual Studio Code w języku c++
Wykorzystane zostały biblioteki
<iostream> - standardowa biblioteka c++
<cmath> - implementuje funkcję sinus potrzebną do rozwiązywania równań
Zadeklarowane stałe użyte w programie to:

```
    int n = 2000;           //liczba krokow siatki
    double h = 0.02;        //krok
    double omega = 1;       //omega
    double b[n];            //wyarzy wolne
    double x[n];            //rozwiazania ukladu rownan liniowych
    double d0[n];           //diagonalna
    double d1[n];           //ponizej diagonalnej
    double d2[n];           //powyzej diagonalnej
    double t[n];            //punkty czasowe siatki
```

Fragment kodu 1: Deklaracja stalych

4 Opis działania

4.1 Inicjalizacja warunków początkowych

```
b[0] = 1;
b[1] = 0;
```

Fragment kodu 2: Deklaracja dwoch pierwszych wyrazow wolnych

```
x[0] = 1/(d0[0]) * ( b[0] - d1[0] * 1 - d2[0] * 1 );
x[1] = 1/(d0[1]) * ( b[1] - d1[1] * 1 - d2[1] * 1 );
```

Fragment kodu 3: Deklaracja stalych rozwiazan

```
d0[0] = 1;
d0[1] = 1;
d1[0] = 0;
d1[1] = -1;
d2[0] = 0;
d2[1] = 0;
```

Fragment kodu 4: Deklaracja stalych wartosci diagonalnych

4.2 Rozwiązanie równań

Gdy mamy już zadeklarowane interesujące nas stałe, oraz punkty początkowe, możemy zabrać się za rozwiązywanie równań. Zadanie zostało przeprowadzone dla 3 różnych grup zmiennych:

1. $\beta = 0.0$, $F_0 = 0.0$, $\Omega = 0.8$
2. $\beta = 0.4$, $F_0 = 0.0$, $\Omega = 0.8$
3. $\beta = 0.4$, $F_0 = 0.1$, $\Omega = 0.8$

Zaczynamy od obliczenia wartości tablicy wyrazów wolnych za pomocą wzoru (15)

```
for(int i = 2; i < n; i++){
    b[i] = F * sin(ohm * h * i) * h * h;
}
```

Fragment kodu 5: Tablica wyrazów wolnych

Następnie uzupełniamy tablicę wartości na diagonali, pod diagonalą i ponad diagonalą.

```
for(int i = 2; i < n; i++){
    d0[i] = a3;
    d1[i] = a2;
    d2[i] = a1;
}
```

Fragment kodu 6: Tablice trójprzekątne

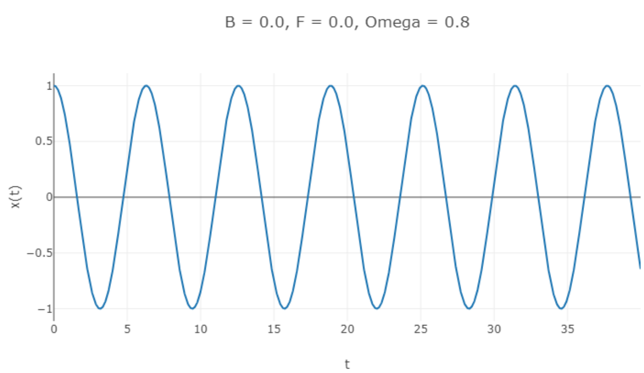
Na końcu uzupełniamy tablicę wyników za pomocą wzoru (14) oraz tablicę przedziałów czasowych, potrzbną do wygenerowania wykresów za pomocą wzoru (5)

```
    for(int i =2; i < n; i++){
        x[i] = 1/(d0[i]) * ( b[i] - d1[i] * x[i-1] - d2[i] * x[i-2] );
    }
    for(int i = 0; i < n; i++){
        t[i] = h * i;
    }
```

Fragment kodu 7: Rozwiązanie i dane do wykresów

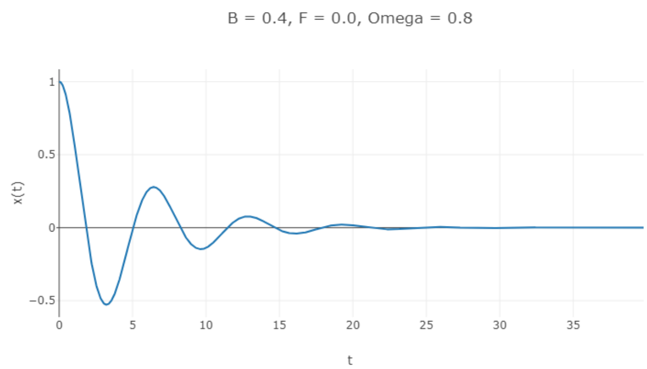
5 Analiza wyników

Jak już wspomniałem wcześniej, analizę przeprowadzimy na trzech różnych grupach zmiennych. Grupa pierwsza: $\beta = 0.4, F_0 = 0.0, \Omega = 0.8$



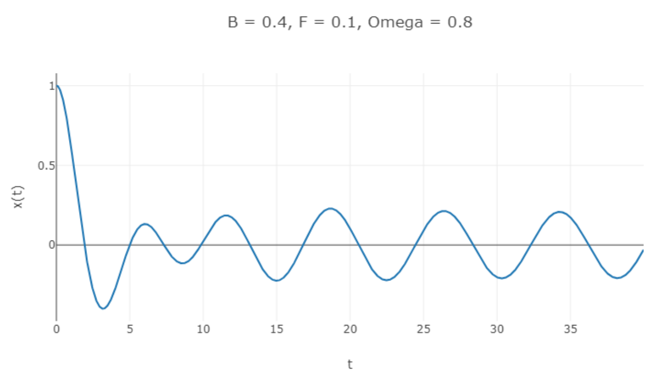
Rysunek 1: Zależność X(t) od czasu t dla danej grupy parametrów B, F i Omega.

Grupa druga: $\beta = 0.4, F_0 = 0.0, \Omega = 0.8$



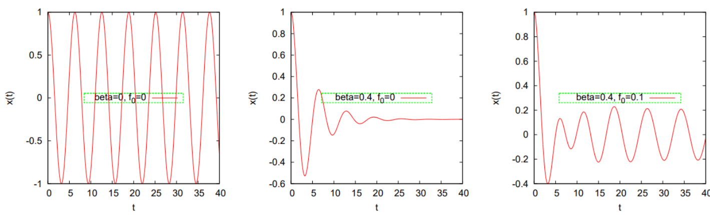
Rysunek 2: Zależność X(t) od czasu t dla danej grupy parametrów B, F i Omega.

Grupa trzecia: $\beta = 0.4, F_0 = 0.0, \Omega = 0.8$



Rysunek 3: Zależność X(t) od czasu t dla danej grupy parametrów B, F i Omega.

Powyższe wykresy możemy porównać do wykresów teroytycznych:



Rysunek 4: Wykresy tabularyczne

6 Wnioski

Metoda Jakobiego okazała się efektywnym narzędziem do iteracyjnego rozwiązywania układu równań liniowych wynikającego z dyskretyzacji równania różniczkowego drugiego rzędu. Zadanie potwierdziło przydatność tej metody, jak i ujawniło, że jest ona bardziej efektywna dla macierzy rzadkich, co ma znaczenie podczas wybierania metody do naszego problemu.

Analiza wyników pokazała, że różne ustawienia parametrów (takie jak współczynnik tłumienia β, F_0, Ω) miały znaczący wpływ na rozwiązania układu równań. Jest to zgodne z oczekiwaniami teoretycznymi i podkreśla znaczenie właściwego doboru parametrów w analizowanych układach fizycznych.

Pomimo tego, że metoda Jakobiego okazała się skuteczna dla rozpatrywanego przypadku, musimy pamiętać o jej ograniczeniach, takich jak bardzo powolne dochodzenie do zbieżności w przypadku dużych układów równań. Kolejnym problemem może się okazać moment, w którym wartość bezwzględna każdego elementu na głównej przekątnej jest większa od sumy wartości bezwzględnych pozostałych elementów w danej kolumnie. Jeśli warunek ten nie jest spełniony, metoda Jakobiego może nie zbiegać do prawidłowego rozwiązania.

Ważną rzeczą w powyższym zadaniu jest również to ,aby właściwie ustawić warunki początkowe. Jest to kluczowy aspekt przy rozwiązywaniu równań różniczkowych metodami numerycznymi.