Akademia Górniczo-Hutnicza w Krakowie

Wydział Fizyki i Informatyki Stosowanej

Metody Numeryczne

Laboratorium 03: Metody Iteracyjne

Andrzej Świętek

12.03.2024

Contents

1. Wstęp teoretyczny

1.1. Formaty zapisu macierzy rzadkich

W kontekście obliczeń numerycznych, zwłaszcza przy rozwiązywaniu układów równań liniowych, macierze rzadkie są niezwykle istotne ze względu na swoją charakterystykę posiadania niewielu elementów niezerowych w stosunku do całkowitej liczby elementów. W celu efektywnego przechowywania i manipulowania takimi macierzami istnieją różne formaty zapisu, z których najpopularniejsze to:

1. **CSR (Compressed Sparse Row):**
   – trzy wektory: wartości, numery kolumn, początki wierszy
   (pierwsze nie-zero w wierszu)

   W tym formacie, macierz jest przechowywana w postaci trzech tablic: tablica wartości niezerowych, tablica kolumn, oraz tablica wskazująca na początki wierszy w tablicy wartości niezerowych.

2. **CSC (Compressed Sparse Column):**
   – trzy wektory: wartości, numery wierszy, początki kolumn
   (pierwsze nie-zero w kolumnie)

   Podobny do CSR, ale zamiast przechowywać wiersze, przechowuje kolumny macierzy rzadkiej.

3. **COO (Coordinate List):**
   – trzy wektory dla: wartości, oraz numery kolumn i wierszy dla nie-zer

   W tym formacie, każdy niezerowy element macierzy jest przechowywany razem z jego współrzędnymi (indeksem wiersza i kolumny).

1.1.1. Przykład

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 \\ 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

1.1.2. Zapis w formacie CSR

Tablica wartości niezerowych (values)

$$\text{values} = [5, 8, 3, 6, 1]$$

Tablica indeksów kolumn (columns)

$$\text{columns} = [0, 1, 2, 1, 4]$$

Ta tablica przechowuje indeksy kolumn dla każdej wartości niezerowej z tablicy values. Na przykład, pierwsza wartość niezerowa (5) znajduje się w pierwszym wierszu macierzy (indeks 0) i pierwszej kolumnie (indeks 0), dlatego w tablicy columns znajduje się indeks kolumny 0. Analogicznie dla kolejnych wartości: 8 jest w drugiej kolumnie (indeks 1), 3 w trzeciej kolumnie (indeks 2), 6 w drugiej kolumnie (indeks 1), 1 w piątej kolumnie (indeks 4).

Tablica wskazań na początki wierszy (row_ptr)

$$\text{row\_ptr} = [0, 0, 2, 3, 4, 5]$$

W tej tablicy przechowywane są indeksy początków wierszy w tablicy values. Jest to kluczowa informacja, ponieważ pozwala określić, które wartości należą do którego wiersza macierzy. Wartości te są wyliczane na podstawie ilości elementów w poszczególnych wierszach. Na przykład, pierwszy wiersz macierzy nie ma wartości niezerowych, więc indeks w tablicy row_ptr dla tego wiersza wynosi 0. Drugi wiersz zaczyna się od indeksu 0 w tablicy values (gdzie znajduje się pierwsza wartość niezerowa), trzeci wiersz zaczyna się od indeksu 2 (gdzie znajduje się trzecia wartość niezerowa), czwarty od indeksu 3, piąty od indeksu 4. Ostatni element tablicy row_ptr zawsze jest równy liczbie wartości niezerowych plus jeden, aby uwzględnić także koniec ostatniego wiersza.

Ta struktura umożliwia efektywne przechowywanie i operowanie macierzami rzadkimi, szczególnie przy dużych macierzach, w których większość elementów jest równa zero.

- Pierwsza niezerowa wartość, czyli 5, znajduje się w drugim wierszu (indeks 1) i pierwszej kolumnie (indeks 0) macierzy.
- W tablicy values pierwszą niezerową wartością jest values[0] = 5.
- W tablicy columns odpowiadający jej indeks kolumny to columns[0] = 0.

1.1.3. Zapis w formacie CSC

Tablica wartości niezerowych (values)

$$\text{values} = [5, 6, 8, 3, 1]$$

Tablica indeksów wierszy (rows)

$$\text{rows} = [1, 3, 1, 2, 4]$$

Tablica wskazań na początki kolumn (col_ptr)

$$\text{col\_ptr} = [0, 1, 3, 3, 4, 5]$$

1.2. Metody Iteracyjne

Metody iteracyjne są jednym z podstawowych narzędzi używanych w numerycznym rozwiązywaniu układów równań liniowych. W odróżnieniu od metod bezpośrednich, takich jak eliminacja Gaussa, metody iteracyjne rozwiązują układ równań poprzez wykorzystanie iteracji, które prowadzą do zbieżności do rozwiązania. Oto kilka cech charakterystycznych metod iteracyjnych:

1. **Iteracyjność:** Metody iteracyjne polegają na wykonywaniu iteracyjnych kroków w celu poprawy przybliżenia rozwiązania.
2. **Relatywnie niska złożoność pamięciowa:** Metody iteracyjne często wymagają przechowywania tylko niewielkiej ilości danych w pamięci, co czyni je atrakcyjnymi w przypadku bardzo dużych macierzy.
3. **Potrzeba określenia warunku stopu:** Aby zakończyć proces iteracyjny, konieczne jest zdefiniowanie warunku stopu, który określa, kiedy algorytm osiągnął wystarczająco dokładne przybliżenie rozwiązania.
4. **Wykorzystanie macierzy rzadkich:** Metody iteracyjne mogą być szczególnie efektywne przy pracy z macierzami rzadkimi, ponieważ unikają one bezpośredniej manipulacji całej macierzy, co jest kosztowne dla dużych macierzy.

Metody:

- Jacobiego,
- Gaussa-Seidla
- Nadrelaksacji

1.3. Metoda Iteracyjna Jakobiego

1.3.1. Wprowadzenie

Jest to metoda rozwiązywania układów równań liniowych. Jako metoda iteracyjna z każdą kolejną iteracją przybliża nas do dokładnego wyniku.

Initial guess $\rightarrow$ Improving solution $\rightarrow$ $\cdots$ $\rightarrow$ Final answer

$\bullet$ $\bullet$ $\bullet$ $\bullet$

Iteration 0 Iter 1 Iter 2 $\cdots$ Iteration n

Lots of error $\rightarrow$ $\cdots$ $\rightarrow$ Error $\rightarrow$ Acceptable Error

1.3.2. Algorytm:

1. Przyjęcie (wylosowanie) początkowego rozwiąznia układu $\rightarrow \vec{x}$
   Często przyjmuje się 0:

$$x_1^{(0)} = 0, \;\; x_2^{(0)} = 0, \;\; x_3^{(0)} = 0 \qquad \cdots \qquad x_n^{(0)} = 0$$

2. Wypisanie wszystkich równań i przekształcenie ich na "x"

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$$a_{1,1} \cdot x_1 + a_{1,2} \cdot x_2 \cdots + a_{1,n} \cdot x_n = b_1$$

   Przekształcamy na x_1

$$x_1 = \frac{b_1 - (a_{1,2} \cdot x_2 \cdots + a_{1,n} \cdot x_n)}{a_{1,1}}$$

   i powtarzamy tę czynność dla każdego równania (dla x_2, x_3, $\cdots$, x_n)

3. Tworzymy tabelkę do obliczeń:

Zmienna	Iter 0	Iter 1	Iter 2	...	Iter n
x_1					
x_2					
$\vdots$					
x_n					

4. Wstawiając dane z punktu (1) wykonujemy obliczenia dla przekształconych równań z punktu (2):

   Dla iteracji 0 wiemy że wszystkie $x_i = 0$

$$x_1^{(0)} = \frac{b_1 - (a_{1,2} \cdot [0] \cdots + a_{1,n} \cdot [0])}{a_{1,1}}$$

$$x_2^{(0)} = \frac{b_2 - (a_{2,1} \cdot [0] \cdots + a_{2,n} \cdot [0])}{a_{2,2}}$$

Wyliczone wartości wstawiamy do tabelki i kontynuujemy następną iterację.

Zmienna	Iter 0	Iter 1	Iter 2	...	Iter n
x_1	$x_1^{(0)}$				
x_2	$x_2^{(0)}$				
$\vdots$	$\vdots$				
x_n	$x_n^{(0)}$				

5. Powtarzamy krok (4) kolejno przyjmując za x_i wartości wyliczone w poprzedniej iteracji.

   Dla iteracji 1 wiemy że wszystkie $x_i = x_i^{(0)}$

$$x_1^{(1)} = \frac{b_1 - (a_{1,2} \cdot [x_2^{(0)}] \cdots + a_{1,n} \cdot [x_n^{(0)}])}{a_{1,1}}$$

$$x_2^{(1)} = \frac{b_2 - (a_{2,1} \cdot [x_1^{(0)}] \cdots + a_{2,n} \cdot [x_n^{(0)}])}{a_{2,2}}$$

Zmienna	Iter 0	Iter 1	Iter 2	...	Iter n
x_1	$x_1^{(0)}$	$x_1^{(1)}$			
x_2	$x_2^{(0)}$	$x_2^{(1)}$			
$\vdots$	$\vdots$	$\vdots$			
x_n	$x_n^{(0)}$	$x_n^{(1)}$			

6. Przerywamy iteracje albo kiedy przekraczamy nasz dopuszczalny błąd ϵ albo po osiągnięciu konkretnej głębokości.

1.3.3. Przykład Liczbowy

$$\left[\begin{array}{ccc|c} 9 & 2 & 3 & 7 \\ 1 & 12 & 9 & 2 \\ 4 & 6 & 14 & 1 \end{array}\right]$$

Zakładamy $x_1 = 0, x_2 = 0, \cdots, x_n = 0$ i wyznaczamy równania x_1, x_2 i x_3:

$$x_1 = \frac{7 - [2 \cdot x_2 + 3 \cdot x_3]}{9}$$

$$x_2 = \frac{2 - [1 \cdot x_1 + 9 \cdot x_3]}{12}$$

$$x_3 = \frac{1 - [4 \cdot x_1 + 6 \cdot x_2]}{14}$$

Iteracja 0:

Zmienna	Iteracja 0	Iteracja 1	...	Iteracja n
x_1	$\frac{7-[2\cdot[0]+3\cdot[0]]}{9} = 0.777$			
x_2	$\frac{2-[1\cdot[0]+9\cdot[0]]}{12} = 0.166$			
x_3	$\frac{1-[4\cdot[0]+6\cdot[0]]}{14} = 0.714$			

Iteracja 1:

Zmienna	Iteracja 0	Iteracja 1	...	Iteracja n
x_1	0.777	$\frac{7-[2\cdot[0.777]+3\cdot[0.777]]}{9} = 0.7169$		
x_2	0.166	$\frac{2-[1\cdot[0.166]+9\cdot[0.166]]}{12} = 0.4828$		
x_3	0.714	$\frac{1-[4\cdot[0.714]+6\cdot[0.714]]}{14} = 0.222$		

Iteracja 2:

Zmienna	Iteracja 0	Iteracja 1	Iteracja 2	...	Iteracja n
x_1	0.777	0.7169	$\frac{7-[2\cdot[0.7169]+3\cdot[0.7169]]}{9} = 0.8411$		
x_2	0.166	0.4828	$\frac{2-[1\cdot[0.4828]+9\cdot[0.4828]]}{12} = 0.2735$		
x_3	0.714	0.222	$\frac{1-[4\cdot[0.222]+6\cdot[0.222]]}{14} = -0.154$		

$\vdots$

Zmienna	Iteracja 0	Iteracja 1	Iteracja 2	Iteracja 3	Iteracja 4	$\cdots$
x_1	0.777	0.7169	0.8411	0.7683	0.8260	$\cdots$
x_2	0.166	0.4828	0.2735	0.2121	0.3172	$\cdots$
x_3	0.714	0.222	-0.154	-0.2861	-0.2390	$\cdots$

1.4. Metoda Iteracyjna Jakobiego dla macierzy rzadkich

Ponieważ macież układu równań (Ax=b) jest trójprzekątniowa (macierz rzadka), więc można ją przechowywać w pamięci w postaci trzech n-elementowych wektorów:

$$d_0 = [1, 1.a_3, a_3, \ldots, a_3]$$
$$d_1 = [1, -1.a_2, a_2, \ldots, a_2]$$
$$d_2 = [0, 0, a_1, a_1, \ldots, a_1]$$

Aby w metodzie Jakobiego wyznaczyć i-ty element nowego przybliżenia ($x_n[i]$) dysponując przybliżeniem z poprzedniej iteracji (wektor x_s) należy wykonać poniższą operację:

$$x_n[i] = \frac{1}{d_0[i]} \left(b[i] - d_1[i] \cdot x_s[i-1] - d_2[i] \cdot x_s[i-2]\right)$$

dla każdego $i = 0, 1, 2, \ldots n$

Elementy wektora x_s indexowane są od -2, wartości $x_s[-2]$ i $x_s[-1]$ mogą być dowolne.

2. Problem

Znaleźć rozwiązanie równania różniczkowego:

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

które opisuje ruch ciała poddanego działaniu siły sprężystej ($-\omega^2 x$), siły tarcia ($-\beta V$) zależnej od prędkości oraz siły wymuszającej ruch ($F_0 sin(\Omega t)$).
Ponieważ problem rozwiązywany jest w czasie więc wprowadzamy siatkę, której węzłami są kolejne chwile czasowe:

$$t = t_i = h \cdot i, \qquad i = 0, 1, 2...$$

Więc nasze rozwiązanie x(t) będzie określone dla położeń węzłowych tj. $x(t) = x_{t_i} = x_i$ Drugą pochodną zamieniamy na symetryczny trójpunktowy iloraz różnicowy:

$$\frac{\partial^2 x}{\partial t^2} = \frac{x_{i-1} - 2x_i + x_{i+1}}{h^2}$$

gdzie: h oznacza krok czasowy na siatce.
Ponieważ prędkość jest pierwszą pochodną położenia po czasie więc ją także zastępujemy ilorazem różnicowym (dwupunktowym niesymetrycznym):

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

I wstawiamy do równania różniczkowego:

$$\frac{x_{i-1} - 2x_i + x_{i+1}}{h^2} = -\omega^2 x - \beta V + F_0 \sin(\Omega t)$$

Przenosimy wyrazy z niewadomymi xi na lewą stronę (zamieniając prędkość na iloraz różnicowy) a na prawej pozostawiamy wyraz wolny:

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

Co można zapisać w symbolicznie:

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

$$a_1 = 1, a_2 = \omega^2 h^2 - 2 - \beta h, a_3 = 1 + \beta h$$

Dostajemy układ równań:

$$Ax = b$$

Aby rozwiązać równanie różniczkowe drugiego rzędu musimy podać dwa warunki początkowe: a) na wychylenie $x(t = 0) = x_0 = 1$ oraz prędkość początkową $V(t = 0) = \frac{xi+1-x_i}{h} = 0$

Te dwa dodatkowe równania musimy dołączyć do naszego układu równań który przyjmuje postać:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ a_1 & a_2 & a_3 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & a_1 & a_2 & a_3 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & a_1 & a_2 & a_3 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & a_1 & a_2 & a_3 & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & a_1 & a_2 & a_3 & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ b_2 \\ b_3 \\ b_4 \\ \vdots \\ b_0 \end{bmatrix}$$

Zadaniem było przyjąć parametry: $V_0 = 0, x_0 = 1, \omega = 1$, liczba kroków czasowych n = 1000, h = 0.02. a następnie znaleźć rozwiązanie układu równań iteracyjną metodą Jakobiego dla trzech przypadków:

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

3. Implementacja

Implementacje zacząłem od zdefiniowania stałych
Oraz wprowadziłem wektor reprezentujący siatkę:

```
    constexpr double h = 0.02f;
    constexpr int N = 1000;
    double V0 = 0.0f;
    double x0 = 1.0f;
    double omega = 1.0f;
    double beta, F0, Omega;                 // wartosci w poleceniu
```

Listing 1: Inicjaliza zmiennych

```
    std::vector<double> t(N, 0);            // siatka
    for(int i = 0; i<N; i++) t[i] = h*i;
```

Listing 2: Utworzenie wektroa siatki t_i

```
    double a1 = 1;
    double a2 = omega*omega * h*h -2 -beta*h;
    double a3 = 1 + beta*h;
```

Listing 3: Utworzenie wektroa siatki t_i

```
   std::vector<double> b(N, 0);
    for( int i = 0; i< N; i++)
        b[i] = F0* std::sin( omega*h*i ) * ( h * h );
```

Listing 4: Wypełnienie wektora wyrazów wolnych

```
    std::vector<double> d0(N, 1.0);
    std::vector<double> d1(N, 1.0);
    std::vector<double> d2(N, 1.0);

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

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

Listing 5: Wyktory reprezentucjące 3 diagonale

```
  //
  // WEKTOR ROZWIAZAN
  //
    std::vector<double> x_n(N, 0.0);
    x_n[0] = 1.0;
    x_n[1] = 1.0;

    for( int i =2; i < N; i++)
        x_n[i] = ( 1/d0[i] ) * ( b[i] - d1[i]*x_n[i-1] - d2[i]*x_n[i-2] );
```

Listing 6: Wyktor rozwiązań

4. Wizualizacja wyników

4.1. $\beta = 0.0,\ F_0 = 0.0,\ \Omega = 0.8$

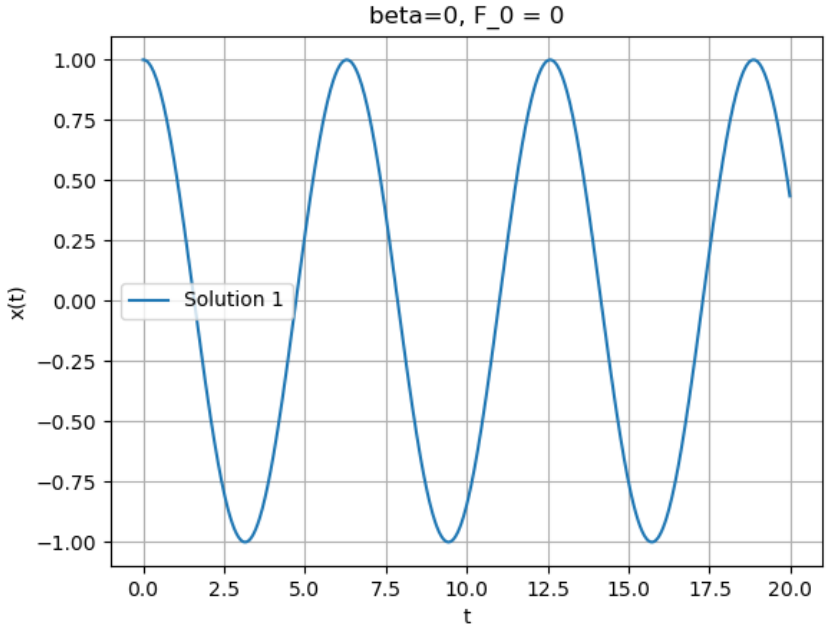

Figure 1: Wykres $x(t)$

4.2. $\beta = 0.4,\ F_0 = 0.0,\ \Omega = 0.8$

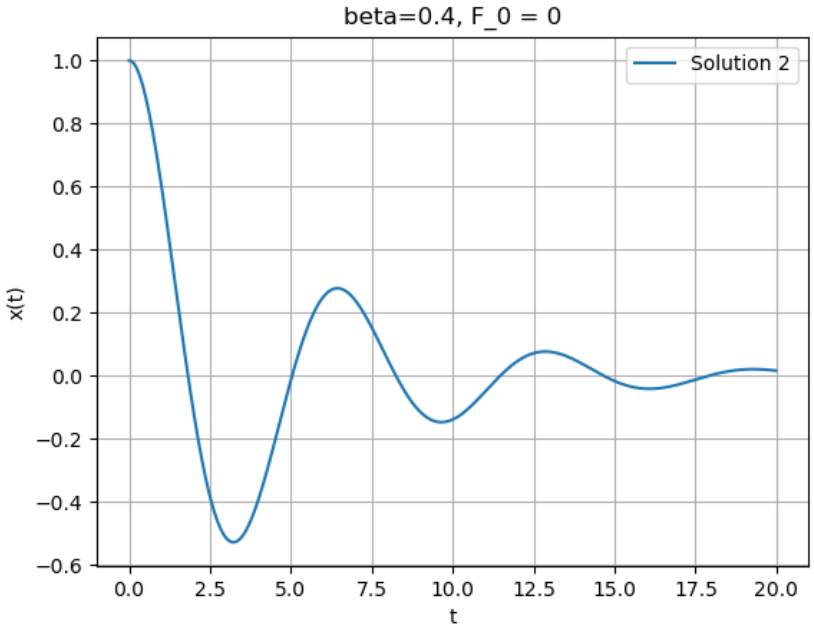

Figure 2: Wykres $x(t)$

4.3. $\beta = 0.4,\ F_0 = 0.1,\ \Omega = 0.8$

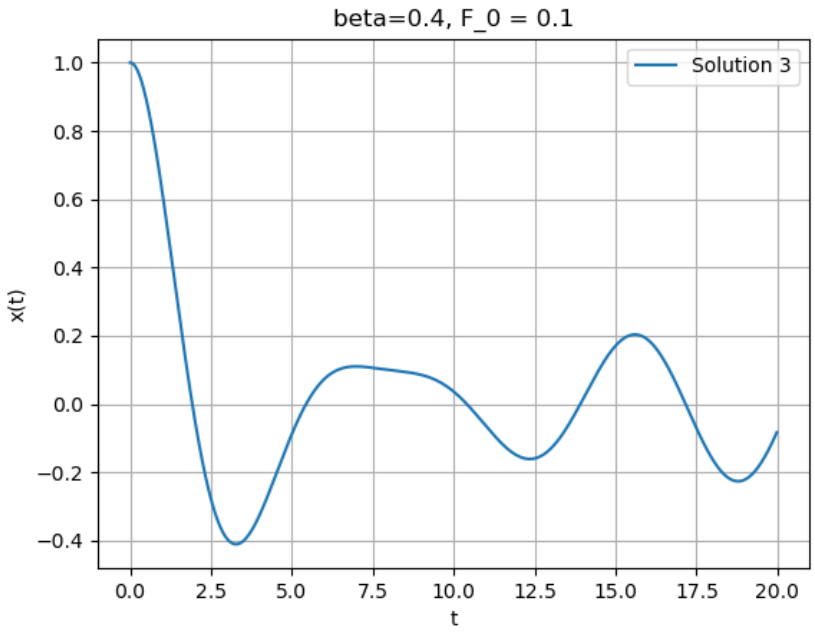

Figure 3: Wykres $x(t)$

5. Wnioski

- Dla rzadkich macierzy oraz macierzy taśmowych dobrym wyborem jest użycie metod iteracyjnych do rozwiązania UARL.
- Nie wszystkie układy równań da się rozwiązać metodami iteracyjnymi, są one bardzo wrażliwe na źle uwarunkowane dane.
- Otrzymane dane pokrywają się z funkcją wychylenia x(t) pod wpływem siły sprężystej: tak jak się spodziewaliśmy przyjmuje wartości od -1 do 1, kształt wykresu w zależności od parametrów określających siłę tarcia oraz wymuszającą również pokrywają się z rzeczywistością.
- Gdybyśmy użyli metody Gaussa-Siedla uzyskalibyśmy wyniki szybciej, ponieważ jest ona modyfikacją metody Jacobiego, w której krok iteracyjny zmieniono w ten sposób, by każda modyfikacja rozwiązania korzystała ze wszystkich aktualnie dostępnych przybliżonych składowych Pozwala to zaoszczędzić połowę pamięci operacyjnej i w większości zastoso- wań praktycznych zmniejsza ok. dwukrotnie liczbę obliczeń niezbędnych do osiągnięcia warunku końcowego.