# Projekt 3

Łukasz Ruzicki Damian Schröder Konrad Ziółkowski

Bartek Siedlikowski

maj 2021

## 1 Treść zadania

Zaimplementować rozwiązywanie układów równań metodą iteracji Jacobiego. Za jej pomocą dla podanego przez użytkownika $\varepsilon > 0$ i punktu startowego, znaleźć przybliżone z dokładnością $\varepsilon$ rozwiązanie ukladu równań:

$$\begin{cases} 15x_1 + 3x_2 + 5x_3 - 6x_4 = 15 \\ 4x_1 - 25x_2 + 2x_3 + 4x_4 = -16 \\ 2x_1 - 5x_2 + 20x_3 + x_4 = -56 \\ x_1 + 3x_2 + 6x_3 - 12x_4 = -80 \end{cases}$$

## 2 Opis teoretyczny iteracyjnej metody Jacobiego

Układ równań

$$Ax = f, \quad A = [a_{ij}]_{i,j} =_{1,...,n}, \quad f = [f_1, ..., f_n]^T,$$

przekształcamy do postaci wektorowego równania punktu stałego

$$x = B_j x + b_j, \quad B_j = [b_{ij}]_{i,j} =_{1,...,n}, \quad b_j = [b_1, ..., bn]^T$$

w następujący sposób. Rozkładamy $A = A_L + A_D + A_R$, gdzie

$$A_L = \begin{bmatrix} 0 & 0 & \dots & 0 \\ a_{21} & 0 & \dots & 0 \\ \dots & & & \\ a_{n1} & a_{n2} & \dots & 0 \end{bmatrix}$$

$$A_D = \begin{bmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & \dots & 0 \\ \dots & & & \\ 0 & 0 & \dots & 0 \end{bmatrix}$$

$$A_R = \begin{bmatrix} 0 & a_{12} & \dots & a_{1n} \\ 0 & 0 & \dots & a_{2n} \\ \dots & & & \\ 0 & 0 & \dots & 0 \end{bmatrix}$$

Układ $Ax = f$ zmieniamy do postaci

$$\begin{gathered} (A_L + A_D + A_R)x = f \\ A_D x = -(A_L + A_R)x + f \\ x = -(A_D)^{-1}(A_L + AR)x + (A_D)^{-1} f \end{gathered}$$

czyli otrzymamy

$$x = B_J + b_J, \text{ gdzie } B_J = -(A_D)^{-1}(A_L + A_R), b_J = (A_D)^{-1} f$$

Iteracje proste dla otrzymanego układu

$$x^{(k+1)} = B_J x^{(k)} + b_J, k \geqslant 0$$

z ustalonym $x^{(0)} \in R^n$, nazywamy metodą Jacobiego dla wyjściowego układu $Ax = f$. [1]

## 2.1 Przykład obliczeń

Układ równań

$$\begin{cases} 15x_1 + 3x_2 + 5x_3 - 6x_4 = 15 \\ 4x_1 - 25x_2 + 2x_3 + 4x_4 = -16 \\ 2x_1 - 5x_2 + 20x_3 + x_4 = -56 \\ x_1 + 3x_2 + 6x_3 - 12x_4 = -80 \end{cases}$$

Rozkładamy $A = A_L + A_D + A_R$,

$$A_L = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 \\ 2 & -5 & 0 & 0 \\ 1 & 3 & 6 & 0 \end{bmatrix}$$

$$A_D = \begin{bmatrix} 15 & 0 & 0 & 0 \\ 0 & -25 & 0 & 0 \\ 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & -12 \end{bmatrix}$$

$$A_R = \begin{bmatrix} 0 & 3 & 5 & -6 \\ 0 & 0 & 2 & 4 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Przygotowujemy iteracje Jacobiego dla układu:

$$\begin{bmatrix} 15 & 3 & 5 & -6 \\ 4 & -25 & 2 & 4 \\ 2 & -5 & 20 & 1 \\ 1 & 3 & 6 & -12 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 15 \\ -16 \\ -56 \\ -80 \end{bmatrix}$$

Przeszktałcamy układ

$$\begin{bmatrix} 15 & 0 & 0 & 0 \\ 0 & -25 & 0 & 0 \\ 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & -12 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 & 3 & 5 & -6 \\ 4 & 0 & 2 & 4 \\ 2 & -5 & 0 & 1 \\ 1 & 3 & 6 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 15 \\ -16 \\ -56 \\ -80 \end{bmatrix}$$

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{5} & \frac{1}{3} & -\frac{2}{5} \\ -\frac{4}{25} & 0 & -\frac{2}{25} & -\frac{4}{25} \\ \frac{1}{10} & -\frac{1}{4} & 0 & \frac{1}{20} \\ -\frac{1}{12} & -\frac{1}{4} & -\frac{1}{2} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 1 \\ \frac{16}{25} \\ -\frac{14}{5} \\ \frac{20}{3} \end{bmatrix}$$

Iteracje Jacobiego:

$$\begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \\ x_3^{(k+1)} \\ x_4^{(k+1)} \end{bmatrix} = \begin{bmatrix} 0 & -\frac{1}{5} & -\frac{1}{3} & \frac{2}{5} \\ \frac{4}{25} & 0 & \frac{2}{25} & \frac{4}{25} \\ -\frac{1}{10} & \frac{1}{4} & 0 & -\frac{1}{20} \\ \frac{1}{12} & \frac{1}{4} & \frac{1}{2} & 0 \end{bmatrix} \begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \\ x_3^{(k+1)} \\ x_4^{(k+1)} \end{bmatrix} + \begin{bmatrix} 1 \\ \frac{16}{25} \\ -\frac{14}{5} \\ \frac{20}{3} \end{bmatrix}, k \geqslant 0.$$

## 2.2 Zakres projektu

Projekt ma na celu wyznaczenie rozwiązania układu równań

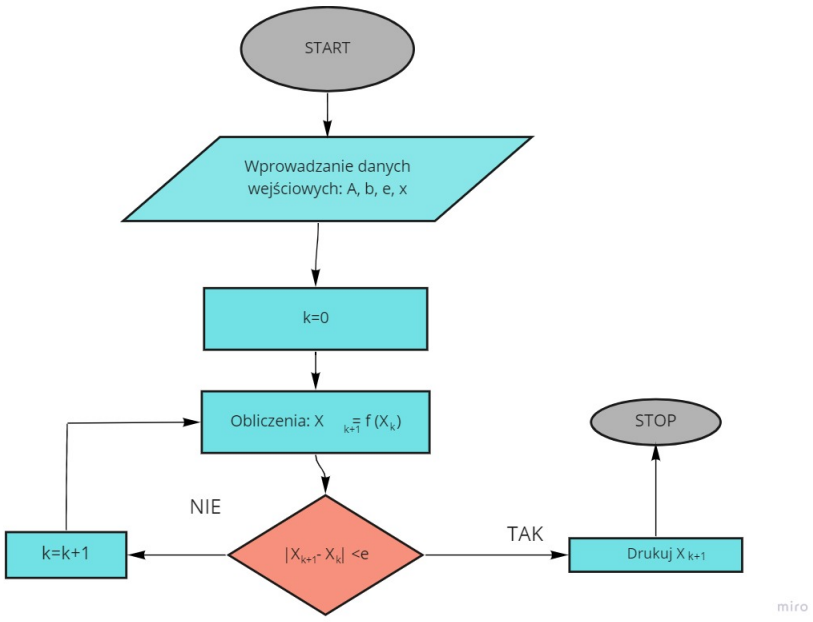
$$\begin{cases} 15x_1 + 3x_2 + 5x_3 - 6x_4 = 15 \\ 4x_1 - 25x_2 + 2x_3 + 4x_4 = -16 \\ 2x_1 - 5x_2 + 20x_3 + x_4 = -56 \\ x_1 + 3x_2 + 6x_3 - 12x_4 = -80 \end{cases}$$

z przybliżeniem $\epsilon$ oraz punktem startowym podanym przez użytkownika. Korzystając z metody iteracji Jacobiego. Program składa się z funkcji, która rozwiązuje układ równań oraz interpretacji wyników.

# 3 Opis implementacji

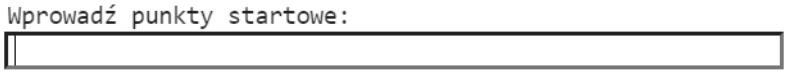
Została użyta dystrybuacja języków programowania Python i R do obliczeń naukowych, której celem jest uprszoczenie zarządzania i używania pakietów. Program został napisany w webowej aplikacji do pisania kodu - Jupyter Notebook, która umożliwia pisanie i kompilacje kodu.

## 3.1 Schemat blokowy

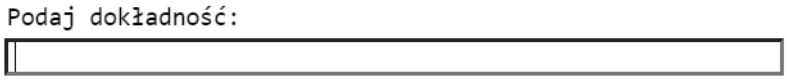


# 4 Instrukcja użytkowania

Program zaleca się uruchomić w webowej aplikacji JupyterLab. Program poprosi użytkownika o wprowadzenie punktów startowych i oczekiwanej dokładności rozwiązania.
Najpierw wprowadzamy ilość punktów startowych odpowiadającą liczbie niewiadomych, które powinny być od siebie oddzielone pustym znakiem.




Następnie należy podać przybliżenie jakie chcemy uzyskać.




Jeśli dane zostały wprowadzone prawidłowo program wyświetli rozwiązanie.

# 5 Raport z demonstracji programu

Sprawdzenie poprawności działania programu dla 1 iteracji z punktem początkowym [0,0,0,0].
Obliczenia pierwszej iteracji dają wyniki dla $k = 0$:

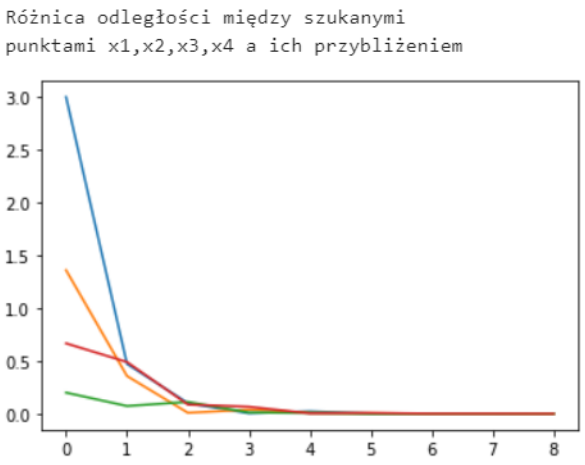
$$\begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \\ x_3^{(k+1)} \\ x_4^{(k+1)} \end{bmatrix} = \begin{bmatrix} 0 & -\frac{1}{5} & -\frac{1}{3} & \frac{2}{5} \\ \frac{4}{25} & 0 & \frac{2}{25} & \frac{4}{25} \\ -\frac{1}{10} & \frac{1}{4} & 0 & -\frac{1}{20} \\ \frac{1}{12} & \frac{1}{4} & \frac{1}{2} & 0 \end{bmatrix} \begin{bmatrix} x_1^{(k+1)} \\ x_2^{(k+1)} \\ x_3^{(k+1)} \\ x_4^{(k+1)} \end{bmatrix} + \begin{bmatrix} 1 \\ \frac{16}{25} \\ -\frac{14}{5} \\ \frac{20}{3} \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{16}{25} \\ -\frac{14}{5} \\ \frac{20}{3} \end{bmatrix}$$

Rozwiązanie przy pomocy programu:

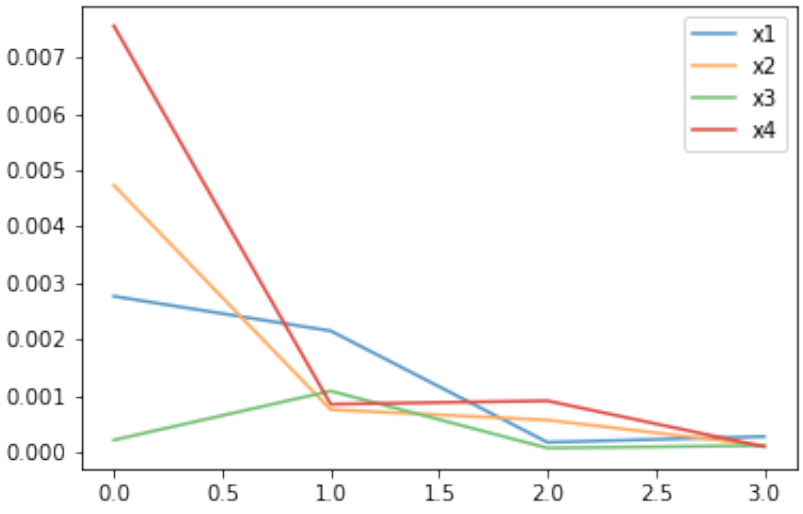
```
Count   x1      x2      x3      x4

1       1.0000  0.6400  -2.8000 6.6667
```

Dla punktu startowego $[0.0, 0.0, 0.0, 0.0]$
Dokładność: 0.001
Liczba iteracji: 9
Wynik: $[4, 2, -3, 6]$
Iteracje:

| Count | x1 | x2 | x3 | x4 |
|---|---|---|---|---|
| 1 | 1.0000 | 0.6400 | -2.8000 | 6.6667 |
| 2 | 4.4720 | 1.6427 | -3.0733 | 5.5100 |
| 3 | 3.8999 | 1.9913 | -3.1120 | 5.9133 |
| 4 | 4.0044 | 1.9612 | -2.9878 | 5.9335 |
| 5 | 3.9771 | 1.9910 | -3.0068 | 5.9967 |
| 6 | 4.0028 | 1.9953 | -2.9998 | 5.9924 |
| 7 | 3.9979 | 1.9992 | -3.0011 | 5.9992 |
| 8 | 4.0002 | 1.9994 | -2.9999 | 5.9991 |
| 9 | 3.9997 | 1.9999 | -3.0001 | 5.9999 |

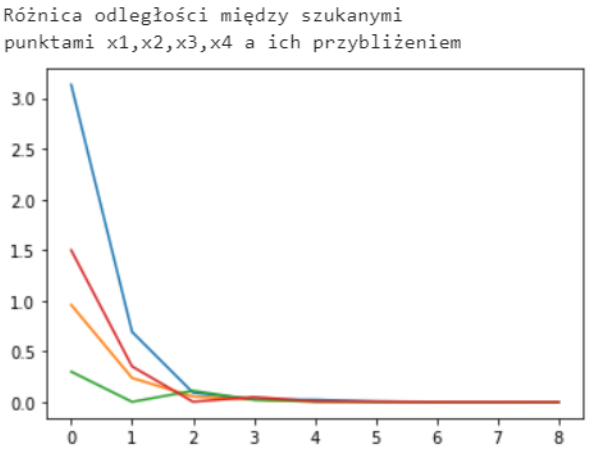



Wykres pomijający pierwszych 5 iteracji:

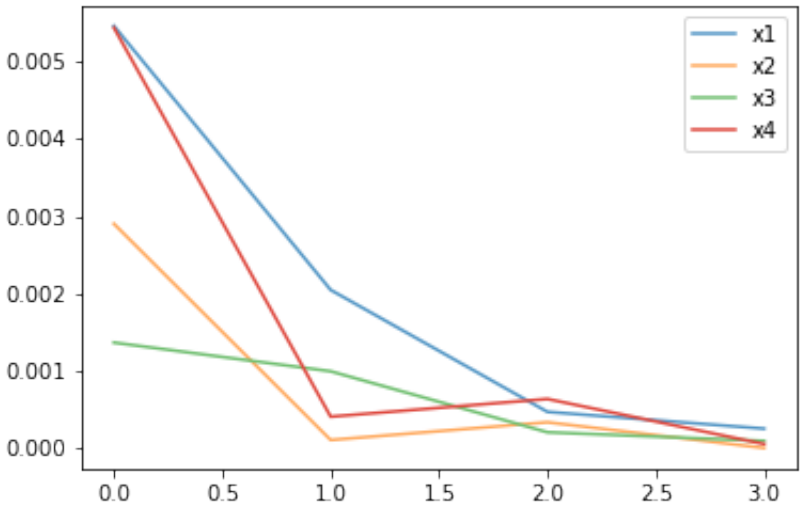


Dla punktu startowego $[1.0, 1.0, 1.0, 1.0]$
Dokładność: 0.001
Liczba iteracji: 9
Wynik: $[4, 2, -3, 6]$
Iteracje:

| Count | x1 | x2 | x3 | x4 |
|---|---|---|---|---|
| 1 | 0.8667 | 1.0400 | -2.7000 | 7.5000 |
| 2 | 4.6920 | 1.7627 | -3.0017 | 5.6489 |
| 3 | 3.9076 | 2.0544 | -3.1110 | 5.9975 |
| 4 | 4.0251 | 1.9759 | -2.9770 | 5.9504 |
| 5 | 3.9773 | 1.9979 | -3.0060 | 6.0076 |
| 6 | 4.0055 | 1.9971 | -2.9986 | 5.9946 |
| 7 | 3.9980 | 2.0001 | -3.0010 | 6.0004 |
| 8 | 4.0005 | 1.9997 | -2.9998 | 5.9994 |
| 9 | 3.9997 | 2.0000 | -3.0001 | 6.0001 |

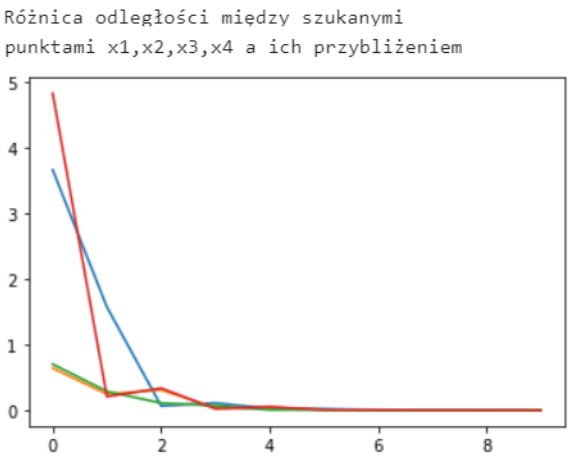



Wykres pomijający pierwszych 5 iteracji:

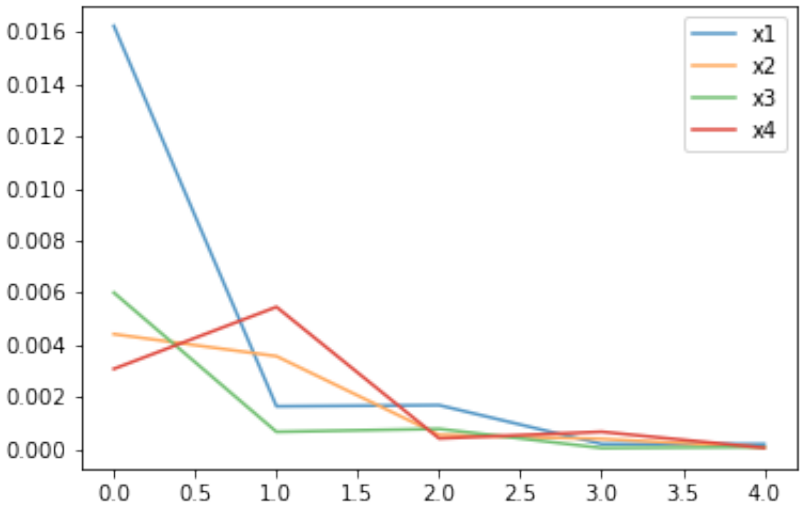


Dla punktu startowego $[5.0, 5.0, 5.0, 5.0]$
Dokładność: 0.001
Liczba iteracji: 10
Wynik: $[4, 2, -3, 6]$
Iteracje:

| Count | x1 | x2 | x3 | x4 |
|---|---|---|---|---|
| 1 | 0.3333 | 2.6400 | -2.3000 | 10.8333 |
| 2 | 5.5720 | 2.2427 | -2.7150 | 6.2044 |
| 3 | 3.9382 | 2.3070 | -3.1068 | 6.3342 |
| 4 | 4.1078 | 2.0350 | -2.9338 | 6.0182 |
| 5 | 3.9782 | 2.0255 | -3.0029 | 6.0509 |
| 6 | 4.0162 | 2.0044 | -2.9940 | 6.0031 |
| 7 | 3.9983 | 2.0036 | -3.0007 | 6.0055 |
| 8 | 4.0017 | 2.0006 | -2.9992 | 6.0004 |
| 9 | 3.9998 | 2.0004 | -3.0001 | 6.0007 |
| 10 | 4.0002 | 2.0001 | -2.9999 | 6.0001 |

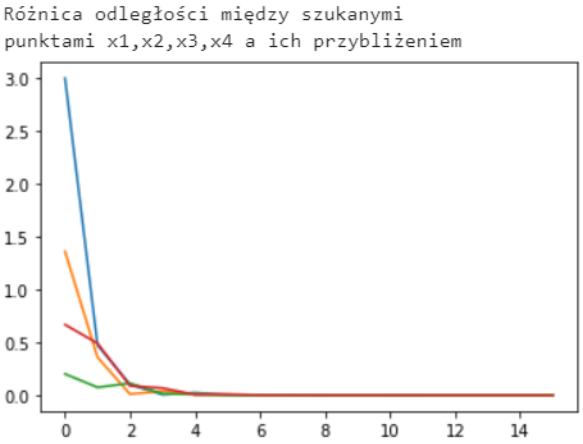



Wykres pomijający pierwszych 5 iteracji:

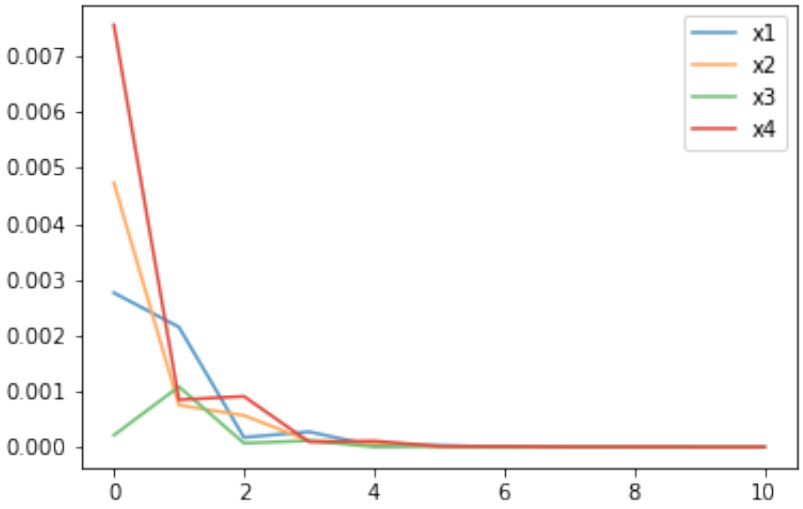


Dla punktu startowego $[0.0, 0.0, 0.0, 0.0]$
Dokładność: 0.000001
Liczba iteracji: 21
Wynik: $[4, 2, -3, 6]$
Iteracje:

| Count | x1 | x2 | x3 | x4 |
|---|---|---|---|---|
| 1 | 1.0000 | 0.6400 | -2.8000 | 6.6667 |
| 2 | 4.4720 | 1.6427 | -3.0733 | 5.5100 |
| 3 | 3.8999 | 1.9913 | -3.1120 | 5.9133 |
| 4 | 4.0044 | 1.9612 | -2.9878 | 5.9335 |
| 5 | 3.9771 | 1.9910 | -3.0068 | 5.9967 |
| 6 | 4.0028 | 1.9953 | -2.9998 | 5.9924 |
| 7 | 3.9979 | 1.9992 | -3.0011 | 5.9992 |
| 8 | 4.0002 | 1.9994 | -2.9999 | 5.9991 |
| 9 | 3.9997 | 1.9999 | -3.0001 | 5.9999 |
| 10 | 4.0000 | 1.9999 | -3.0000 | 5.9999 |
| 11 | 4.0000 | 2.0000 | -3.0000 | 6.0000 |
| 12 | 4.0000 | 2.0000 | -3.0000 | 6.0000 |
| 13 | 4.0000 | 2.0000 | -3.0000 | 6.0000 |
| 14 | 4.0000 | 2.0000 | -3.0000 | 6.0000 |
| 15 | 4.0000 | 2.0000 | -3.0000 | 6.0000 |
| 16 | 4.0000 | 2.0000 | -3.0000 | 6.0000 |



Wykres pomijający pierwszych 5 iteracji:

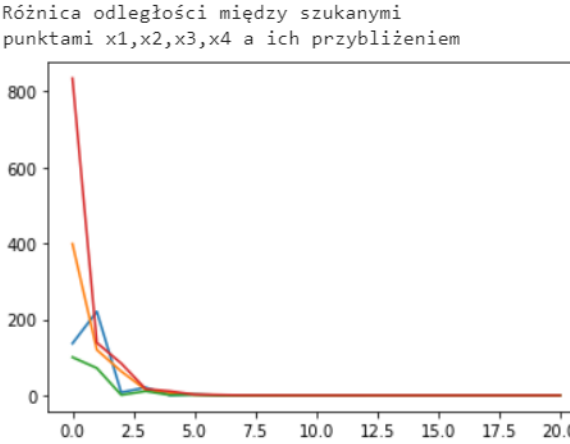


Dla punktu startowego $[1000.0, 1000.0, 1000.0, 1000.0]$
Dokładność: 0.000001
Liczba iteracji: 21
Wynik: $[4, 2, -3, 6]$

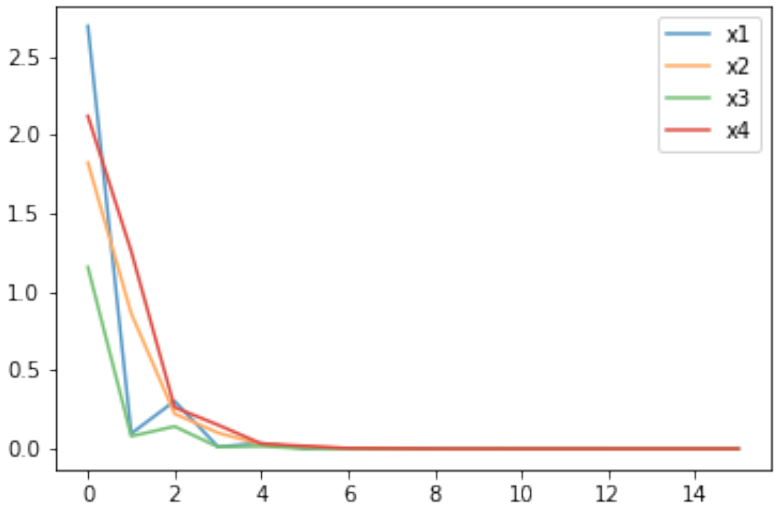
| Count | x1 | x2 | x3 | x4 |
|---|---|---|---|---|
| 1 | -132.3333 | 400.6400 | 97.2000 | 840.0000 |
| 2 | 224.4720 | 121.6427 | 68.5933 | 144.3989 |
| 3 | 11.5666 | 65.1468 | -2.0565 | 90.0800 |
| 4 | 24.6881 | 16.7389 | 7.8260 | 22.8890 |
| 5 | 4.1991 | 8.8784 | -2.2285 | 16.8218 |
| 6 | 6.6959 | 3.8251 | -1.8414 | 8.1219 |

• • •

| | | | | |
|---|---|---|---|---|
| 21 | 4.0000 | 2.0000 | -3.0000 | 6.0000 |

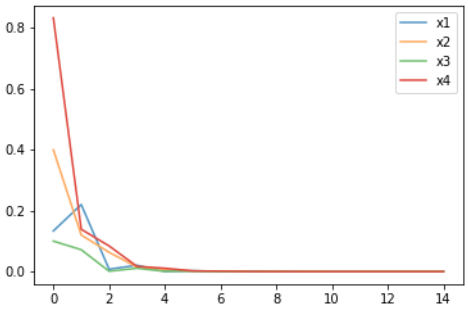



Wykres pomijający pierwszych 5 iteracji:

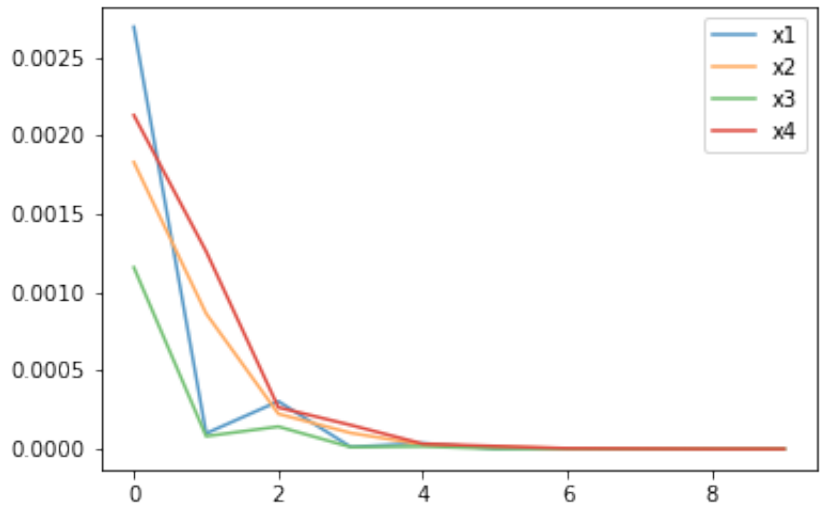


Dla punktu startowego $[5.0, 3.0, -2.0, 7.0]$
Dokładność: 0.000001
Liczba iteracji: 16
Wynik: $[4, 2, -3, 6]$

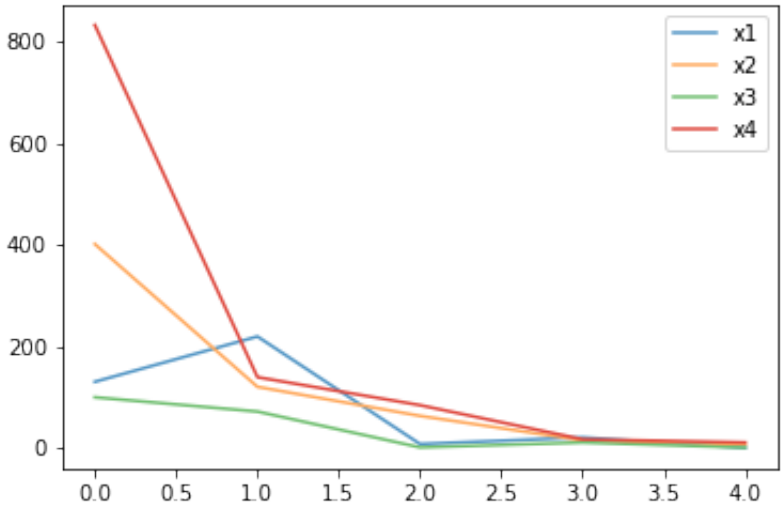
| Count | x1 | x2 | x3 | x4 |
|---|---|---|---|---|
| 1 | 3.8667 | 2.4000 | -2.9000 | 6.8333 |
| 2 | 4.2200 | 2.1200 | -2.9283 | 6.1389 |
| 3 | 4.0077 | 2.0632 | -2.9989 | 6.0842 |
| 4 | 4.0207 | 2.0148 | -2.9892 | 6.0170 |
| 5 | 4.0002 | 2.0069 | -2.9992 | 6.0108 |
| ... | | | | |
| 12 | 4.0000 | 2.0000 | -3.0000 | 6.0000 |
| 13 | 4.0000 | 2.0000 | -3.0000 | 6.0000 |
| 14 | 4.0000 | 2.0000 | -3.0000 | 6.0000 |
| 15 | 4.0000 | 2.0000 | -3.0000 | 6.0000 |



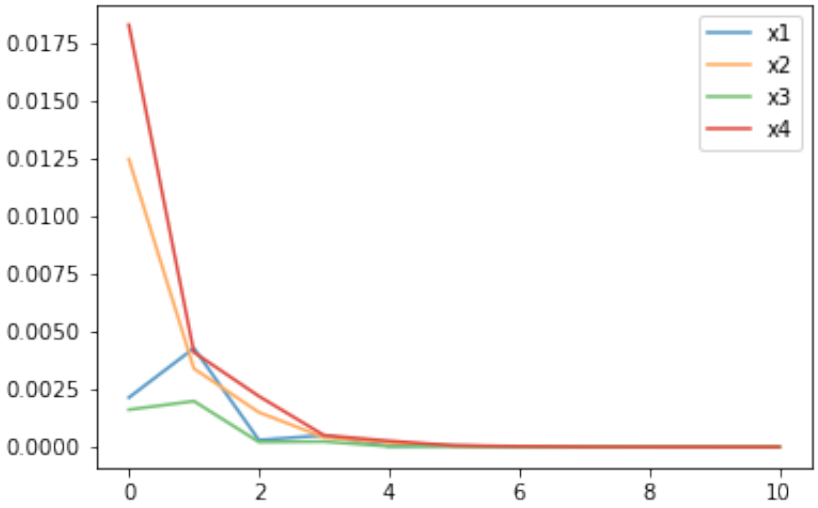

Wykres pomijający pierwszych 5 iteracji:



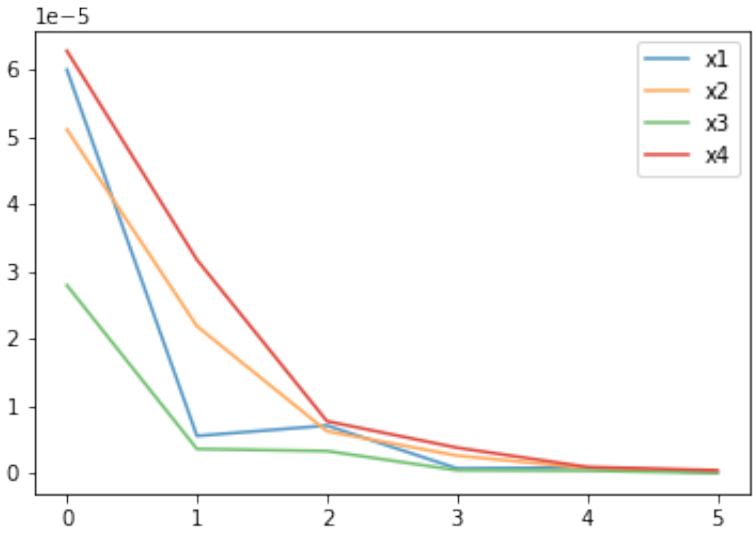
Dla punktu startowego $[-1000, -1000, -1000, -1000]$
Dokładność: 0.000001
Liczba iteracji: 21
Wynik: $[4, 2, -3, 6]$ Wykres różnic do 5 iteracji:



Wykres różnic od 10 iteracji:




Wykres różnic od 15 iteracji:



# 6 Wnioski i interpretacja wyników

Im bliższe przybliżanym punktom wartości początkowe, tym mniejsza liczba iteracji Jacobiego będzie potrzebna do wykonania programu.
Przeciwnie jest z dokładnością. Im większa wartość dokładności tym mniej iteracji potrzeba do obliczenia szukanego przybliżenia rozwiżania układu równań. Największe różnice miedzy punktem przybliżanym a uzyskanym przybliżeniem występują w pierwszych iteracjach.

# 7 Bibliografia

Wykłady przygotowane przez Jaruszewska-Walczak Danuta

# Literatura

[1] dr Jaruszewska-Walczak Danuta. *wykład nr 10: iteracyjna metoda Jacobiego i metoda Gaussa-Seidla.* URL: `https://mdl.ug.edu.pl/pluginfile.php/349719/mod_resource/content/1/wykl-10.pdf`.

# 8 Kod programu

```
from matplotlib import pyplot as plt
import pandas as pd
from pprint import pprint
from numpy import array, zeros, diag, diagflat, dot
import numpy as np

x1_list = []
x2_list = []
x3_list = []
x4_list = []


def jacobi(A, b, e=0.1, x=None):
    if x is None:
        x = zeros(len(A[0]))

    D = diag(A)
    R = A - diagflat(D)

    count = 1
    condition = True
    x0 = array(x)

    print('\nCount\tx1\tx2\tx3\tx4\n')
    while condition:
        x = (b - dot(R,x)) / D
        print('%d\t%0.4f\t%0.4f\t%0.4f\t%0.4f\n' %(count, x[0],x[1],x[2],x[3]))

        x1_list.append(abs(4-x[0]))
        x2_list.append(abs(2-x[1]))
        x3_list.append(abs(-3-x[2]))
        x4_list.append(abs(6-x[3]))


        e1 = abs(x0[0]-x[0])
        e2 = abs(x0[1]-x[1])
        e3 = abs(x0[2]-x[2])
        e4 = abs(x0[3]-x[3])

        count+=1
        x0=array(x)
        condition = (e1 > e or e2 > e or e3 > e or e4 > e)

    return np.round(x,2)
```

```
A = array([[15.0,3.0, 5.0, -6.0],[4.0,-25.0, 2.0, 4.0],[2.0,-5.0, 20.0, 1.0],
[1.0,3.0, 6.0, -12.0]])
b = array([15.0,-16.0, -56, -80])

while True:
    try:
        print('Wprowadź punkty startowe: ')
        p1, p2, p3, p4 = [float(p) for p in input().split()]
        e = float(input('Podaj dokładność: '))
    except ValueError:
        print ('Stosuj się do zasad:/')
    else:
        guess = array([p1,p2,p3,p4])
        sol = jacobi(A,b,e,x=guess)
        break

print ("A:")
pprint(A)

print ("b:")
pprint(b)

print ("x:")
pprint(sol)
plt.plot(x1_list[5:])
plt.plot(x2_list[5:])
plt.plot(x3_list[5:])
plt.plot(x4_list[5:])


wszystkie = []
wszystkie.append(x1_list)
wszystkie.append(x2_list)
wszystkie.append(x3_list)
wszystkie.append(x4_list)


print("Różnica odległości między szua punktami x1,x2,x3,x4 a ich przybliżeniem")
```