Program do zadania Nr.2:

1. **Algorytm Thomasa**

```
from timeit import default_timer as timer
import numpy as np

#Algorytm Thomasa
# a - przekątna poniżej głównej
# b - główna przekątna macierzy
# c - przekątna nad główną
# d - prawa strona (kolumna(wyniki))
def Thomas(a,b,c,d):
    try:
        n = len(d)
        A = np.array([[0]*n]*n, dtype='float64')

        for i in range(n):
            A[i,i] = b[i]
            if i > 0:
                A[i, i-1] = a[i]
            if i < n-1:
                A[i, i+1] = c[i]

        c_1 = [0]*n
        d_1 = [0]*n

        for i in range(n):
            if not i:
                c_1[i] = c[i]/b[i]
                d_1[i] = d[i] / b[i]
            else:
                c_1[i] = c[i]/(b[i]-c_1[i-1]*a[i])
                d_1[i] = (d[i]-d_1[i-1]*a[i])/(b[i]-c_1[i-1] * a[i])

        x = [0]*n

        for i in range(n-1, -1, -1):
            if i == n-1:
                x[i] = d_1[i]
            else:
                x[i] = d_1[i]-c_1[i]*x[i+1]

        x = [round(_, 16) for _ in x]

        return x

    except Exception as e:
        return e

#Wzór Shermana-Morissona dla rozwiązywania cyklicznego trójdiagonalnego równania liniowego A*x = b
# a - przekątna poniżej głównej
# b - główna przekątna macierzy
# c - przekątna nad główną
```

2. Wzór Shermana-Morissona

```python
# d - prawa strona (kolumna(wyniki))
def ShermanMorrison(a,b,c,d):
    try:
        n = len(d)
        A = np.array([[0] * n] * n, dtype='float64')

        for i in range(n):
            A[i, i] = b[i]
            if i > 0:
                A[i, i - 1] = a[i]
            if i < n - 1:
                A[i, i + 1] = c[i]
        A[0, n - 1] = a[0]
        A[n - 1, 0] = c[n - 1]

        gamma = 1 # może być dowolna
        u = [gamma] + [0] * (n - 2) + [c[n - 1]]
        v = [1] + [0] * (n - 2) + [a[0] / gamma]

        # Modyfikacja współczynnika A'
        b[0] -= gamma
        b[n - 1] -= a[0] * c[n - 1] / gamma
        a[0] = 0
        c[n - 1] = 0

        # rozwiąż A'y = d, A'z = u, używając algorytmu Thomasa
        y = np.array(Thomas(a, b, c, d))
        z = np.array(Thomas(a, b, c, u))

        # używamy y i z, aby obliczyć x
        # x jest rozwiązaniem A*x = b
        x = y - (np.dot(np.array(v), y)) / (1 + np.dot(np.array(v), z)) * z
        x = [round(_, 16) for _ in x]
        return x

    except Exception as e:
        return e

def main():
    a = [1, 1, 1, 1, 1, 1, 1]
    b = [4, 4, 4, 4, 4, 4, 4]
    c = [1, 1, 1, 1, 1, 1, 1]
    d = [1, 2, 3, 4, 5, 6, 7]
    start_time = timer()*1000000000
    x = ShermanMorrison(a,b,c,d)
    end_time = timer()*1000000000
    for index, value in enumerate(x):
        print('X%s --> ' %index, value)
    print("\nTotal elapsed: {:g} nanosecs".format(end_time - start_time))
main()
```

Wyniki do zadania Nr.2:

```
C:\Users\homen\IdeaProjects\JavaDataFrame\venv\Scripts\python.exe
X0 -->  -0.2601626016260162
X1 -->  0.4471544715447156
X2 -->  0.4715447154471545
X3 -->  0.6666666666666667
X4 -->  0.8617886178861788
X5 -->  0.8861788617886178
X6 -->  1.5934959349593496

Total elapsed: 241200 nanosecs

Process finished with exit code 0
```

Komentarze do zadania Nr.2:

W tym zadaniu było użyto Tridiagonal matrix algorithm lub zwanym jeszcze jak algorytm Thomasa, żeby zapisać sumę macierzy trójdiagonalnej oraz iloczynu odpowiednich wektorów a następnie skorzystałem ze wzoru Shermana-Morrisona dla modyfikacji rozwiązania.

Algorytm Thomasa:

Rozwiązanie jest wykonywane w dwóch etapach, podobnie jak w metodzie Gaussa: podstawianie do przodu i podstawianie wstecz.

Podstawianie do przodu składa się z obliczenia nowych współczynników w następujący sposób, oznaczających nowe współczynniki liczbami pierwszymi:

$$c'_i = \begin{cases} \dfrac{c_i}{b_i} & ; \quad i = 1 \\ \dfrac{c_i}{b_i - a_i c'_{i-1}} & ; \quad i = 2, 3, \dots, n-1 \end{cases}$$

oraz

$$d'_i = \begin{cases} \dfrac{d_i}{b_i} & ; \quad i = 1 \\ \dfrac{d_i - a_i d'_{i-1}}{b_i - a_i c'_{i-1}} & ; \quad i = 2, 3, \dots, n. \end{cases}$$

Rozwiązanie jest następnie otrzymywane przez podstawianie wstecz:

$$x_n = d'_n$$

$$x_i = d'_i - c'_i x_{i+1} \qquad ; \; i = n-1, n-2, \dots, 1.$$

Wzór Shermana-Morissona:

Używamy wzoru Shermana-Morrisona, traktując system jako trójdiagonalny z korektą. W notacji równania:

$$(\mathbf{A} + \mathbf{u} \otimes \mathbf{v}) \cdot \mathbf{x} = \mathbf{b}$$

zdefiniujmy wektory u i v, które mają być:

$$\mathbf{u} = \begin{bmatrix} \gamma \\ 0 \\ \vdots \\ 0 \\ \alpha \end{bmatrix} \qquad \mathbf{v} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ \beta/\gamma \end{bmatrix}$$

Tutaj y jest na razie dowolne. Wtedy macierz A jest trójdiagonalną częścią macierzy, którą mamy w zadaniu Nr.2, ze zmodyfikowanymi dwoma współczynnikami:

$$b_1' = b_1 - \gamma, \qquad b_N' = b_N - \alpha\beta/\gamma$$

Teraz rozwiązujemy równania:

$$\mathbf{A} \cdot \mathbf{y} = \mathbf{b} \qquad\qquad \mathbf{A} \cdot \mathbf{z} = \mathbf{u}$$

Gdzie y i z są wektorami.

za pomocą **algorytmu Thomasa**, a następnie uzyskujemy rozwiązanie z równania:

$$\mathbf{x} = \mathbf{y} - \left[\frac{\mathbf{v} \cdot \mathbf{y}}{1 + (\mathbf{v} \cdot \mathbf{z})}\right] \mathbf{z}$$