Zadanie numeryczne NUM3

1 Wstęp

Zadanie polegało na wyznaczeniu $y = A^{-1}x$ oraz obliczeniu wyznacznika A dla

$$A = \begin{pmatrix} 1.2 & \frac{0.1}{1} & \frac{0.4}{1^2} & & & & & & \\ 0.2 & 1.2 & \frac{0.1}{2} & \frac{0.4}{2^2} & & & & & \\ & 0.2 & 1.2 & \frac{0.1}{3} & \frac{0.4}{3^2} & & & & \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ & & & & & 0.2 & 1.2 & \frac{0.1}{N-2} & \frac{0.4}{(N-2)^2} \\ & & & & & & 0.2 & 1.2 & \frac{0.1}{N-1} \\ & & & & & & & 0.2 & 1.2 \end{pmatrix}$$

oraz $x = (1, 2, \dots, N)^T$, gdzie $N = 100$. Do rozwiązania równania należało dobrać właściwą metodę oraz wykorzystać strukturę macierzy.

2 Rozwiązanie

Do rozwiązania równania skorzystałem z rozkładu LU. Macierz A jest macierzą wstęgową, a więc macierze L i U również będą wstęgowe. Będą one wyglądały następująco:

$$L = \begin{pmatrix} 1 & & & & & & & & \\ l_{2,1} & 1 & & & & & & & \\ & l_{3,2} & 1 & & & & & & \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ & & & & & l_{N-2,N-3} & 1 & & \\ & & & & & & l_{N-1,N-2} & 1 & \\ & & & & & & & l_{N,N-1} & 1 \end{pmatrix}$$

$$U = \begin{pmatrix} u_{1,1} & u_{1,2} & u_{1,3} & & & & & & \\ & u_{2,2} & u_{2,3} & u_{2,4} & & & & & \\ & & u_{3,3} & u_{3,4} & u_{3,5} & & & & \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ & & & & & & u_{N-2,N-2} & u_{N-2,N-1} & u_{N-2,N} \\ & & & & & & & u_{N-1,N-1} & u_{N-1,N} \\ & & & & & & & & u_{N,N} \end{pmatrix}$$

Zauważyłem, że do obliczenia tych macierzy wystarczą wzory na: $l_{i+1,i}$, $u_{i,i}$, $u_{i,i+1}$ oraz $u_{i,i+2}$. Uzyskałem je ze standardowych wzorów stosowanych do rozkładu LU, czyli:

$$u_{i,j} = a_{i,j} - \sum_{k<i} l_{i,k} u_{k,j}$$
$$l_{i,j} = \frac{a_{i,j} - \sum_{k<j} l_{i,k} u_{k,j}}{u_{j,j}}$$

Ze wzoru na $l_{i,j}$ policzyłem $l_{i+1,i}$:

$$l_{i+1,i} = \frac{a_{i+1,i} - \sum_{k<i} l_{i+1,k} u_{k,i}}{u_{i,i}}$$

Zauważyłem, że element $l_{i+1,k}$ (a co za tym idzie cała suma) zawsze będzie równy 0, ponieważ róznica $i+1$ i $k<i$ zawsze będzie większa lub równa dwa. W ten sposób otrzymałem wzór:

$$l_{i+1,i} = \frac{a_{i+1,i}}{u_{i,i}}$$

Podobnie postąpiłem z resztą wzorów:

$$u_{i,i} = a_{i,i} - \sum_{k<i} l_{i,k} u_{k,i}$$

Elementy $l_{i,k}$ dla $i-k>1$ są równe 0, więc:

$$u_{i,i} = a_{i,i} - l_{i,i-1} u_{i-1,i}$$

$$u_{i,i+1} = a_{i,i+1} - \sum_{k<i} l_{i,k} u_{k,i+1}$$
$$u_{i,i+1} = a_{i,i+1} - l_{i,i-1} u_{i-1,i+1}$$

$$u_{i,i+2} = a_{i,i+2} - \sum_{k<i} l_{i,k} u_{k,i+2}$$
$$u_{i,i+2} = a_{i,i+2}$$

Jeżeli faktoryzacja LU jest znana, wyznacznik można obliczyć wymnażając elementy diagonali macierzy U. Równanie

$$Ay \equiv LUy = x$$

rozwiązałem jako

$$Lv = x$$
$$Uy = v$$

pierwsze równanie rozwiązałem metodą *forward substitution*, a drugie metodą *back substitution*.

3 Implementacja

Program napisałem w języku Python. Ponieważ macierz A jest macierzą wstęgową, nie przechowuję całej macierzy, a tylko elementy "wstęgi":

```
N = 100
a, b, c, d = list(), list(), list(), list()
for i in range(N):
    c.append(1.2)
    if i != 99:
        b.append(0.1 / (i + 1))
        d.append(0.2)
        if i != 98:
            a.append(0.4 / pow((i + 1), 2))
A = [a, b, c, d]
```

, gdzie:

$$a = (\tfrac{0.4}{1^2}, \tfrac{0.4}{2^2}, \ldots, \tfrac{0.4}{(N-2)^2}),$$
$$b = (\tfrac{0.1}{1}, \tfrac{0.1}{2}, \ldots, \tfrac{0.1}{N-1}),$$
$$c = (1.2, 1.2, \ldots, 1.2),$$
$$d = (0.2, 0.2, \ldots, 0.2)$$

Następnie, korzystając z wyprowadzonych wcześniej wzorów, dokonałem faktoryzacji LU. Otrzymane wyniki nadpisałem w miejsce niepotrzebnych już elementów macierzy A:

```
# LU
for i in range(N):
    if i != 0:
        if i != 99:
            A[1][i] -= (A[3][i-1] * A[0][i-1])
        A[2][i] -= (A[3][i-1] * A[1][i-1])
    if i != 99:
        A[3][i] /= A[2][i]
```

Do obliczenia wyznacznika służy funkcja *calc_det*:

```
def calc_det():
    my_det = 1
    for i in range(N):
        my_det *= A[2][i]
    return my_det
```

Do obliczenia wektora y służy funkcja *calc_y*:

```
def cal_y():
  # forward substitutio
    v = list()
    v.append(x[0])
    for i in range(1, N):
        element = x[i] - (A[3][i - 1] * v[i - 1])
        v.append(element)

  # back substitution
    y = list()
    y.append(v[99] / A[2][99])
    y.append((v[98] - A[1][98] * y[0]) / A[2][98])
    for i in range(97, -1, -1):
        element = (v[i] - A[1][i] * y[98 - i] - A[0][i] * y[97 - i
    ]) / A[2][i]
        y.append(element)

    y.reverse()
    return y
```

4 Wyniki

Wyznacznik macierzy A: 78240161.00959387
Wektor: y = (0.0328713348604139, 1.3396227980963753, 2.066480295894664, 2.825543605175336, 3.557571715528883, 4.284492868897645, 5.00721018451999, 5.727664002754518, 6.446615582748809, 7.164554400995276, 7.881773878242026, 8.598465868371878, 9.314759799907844, 10.030746230199034, 10.74649032115277, 11.462040127963592, 12.177431844626687, 12.892693237901542, 13.60784595684208, 14.322907124390252, 15.03789045794619, 15.75280707355121, 16.467666073000725, 17.182474979167374, 17.897240063340146, 18.611966594532937, 19.32665903159678, 20.041321172855753, 20.75595627381683, 21.47056714061568, 22.185156204831525, 22.899725583859315, 23.61427712998635, 24.328812470561147, 25.043333041083297, 25.757840112626393, 26.472334814693667, 27.186818154368854, 27.901291032443737, 28.615754257064278, 29.33020855532933, 30.04465458319117, 30.75909293394065, 31.473524145507586, 32.18794870676451, 32.902367062989086, 33.61677962061327, 34.331186751365145, 35.04558879589254, 35.75998606694211, 36.47437885215638, 37.18876741654113, 37.90315200464761, 38.617532842507245, 39.331910139350974, 40.04628408914067, 40.7606548719361, 41.47502265511775, 42.189387594482916, 42.90374983523002, 43.6181095128443, 44.33246675389621, 45.04682167676243, 45.76117439227791, 46.47552500432681, 47.189873610378676, 47.904220301975755, 48.61856516517662, 49.33290828096055, 50.047249725596565, 50.761589570980924, 51.47592788494589, 52.19026473154275, 52.904600171301595, 53.61893426146981, 54.33326705623165, 55.04759860691019, 55.761928962153874, 56.47625816810818, 57.19058626857465, 57.90491330515779, 58.61923931740096, 59.33356434291259, 60.04788841748285, 60.76221157519233, 61.47653384851288, 62.1908552684013, 62.9051758643867, 63.61949566465193, 64.33381469610926, 65.04813298447127, 65.76245055431694, 66.47676742915336, 67.19108363147355, 67.9053991828134, 68.61971410401006, 69.33402833257784, 70.04833794418792, 70.7650588638003, 71.53915685603329)^T
Do sprawdzenia wyników skorzystałem z biblioteki numerycznej *numpy*:

```
class TestNUM3(unittest.TestCase):

    def setUp(self):
        self.matrix = np.zeros((100, 100))
        for i in range(100):
            self.matrix[i][i] = 1.2
            if i != 0:
                self.matrix[i][i - 1] = 0.2
            if i != 99:
                self.matrix[i][i + 1] = 0.1 / (i + 1)
                if i != 98:
                    self.matrix[i][i + 2] = 0.4 / pow(i + 1, 2)
        self.x = [i+1 for i in range(100)]

    def test_det(self):
        self.assertAlmostEqual(np.linalg.det(self.matrix), det, 5)

    def test_y(self):
        for i in range(100):
```

```
            self.assertAlmostEqual(np.linalg.solve(self.matrix,
    self.x)[i], y[i])


if __name__ == '__main__':
    unittest.main()
```

Wyniki są zgodne z tymi otrzymanymi za pomocą zaimplementowanych metod.