RAPORT
Logika Algorytmiczna dla Inżynierów

Student: Michał Zając

Album: 136636

Temat: Rozwiązywanie układu równań liniowych 3x3 oraz 2x2

Spis treści:

1. Opis Problemu

1.1 Układ równań liniowych

Układ w postaci:

$$\begin{cases} a11 \cdot x1 + a12 \cdot x2 + \cdots + a1n \cdot xn = b1 \\ a21 \cdot x1 + a22 \cdot x2 + \cdots + a2n \cdot xn = b2 \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ am1 \cdot x1 + am2 \cdot x2 + \cdots + amn \cdot xn = bm \end{cases}$$

nazywamy układem równań liniowych, gdzie

- a11, …, amn - to współczynniki równania
- b1, …, bm - to wyrazy wolne
- x1, …, xn - niewiadome równania.

1.2 Rodzaje układów

1.2.1 Def: układ niejednorodny i układ jednorodny

Układ równań liniowych nazywamy niejednorodnym, jeżeli jakikolwiek wyraz wolny nie jest równy zeru. W przeciwnym wypadku, gdy wszystkie wyrazy wolne są równe zeru, układ równań liniowych nazywamy jednorodnym.

1.2.2 Def: układ nieoznaczony:

Jeżeli układ równań posiada nieskończenie wiele rozwiązań to nazywamy go nieoznaczonym

1.2.3 Def: układ oznaczony

Jeżeli układ równań posiada dokładnie jedno rozwiązanie to nazywamy go oznaczonym

1.2.3 Def: układ sprzeczny

Jeżeli układ równań nie ma rozwiązań to nazywamy go sprzecznym

1.3 Układ równań liniowych jako macierz

Układ równań liniowych można przedstawić w postaci macierzowej.

Układ równań w postaci:

$$\begin{cases} a11 \cdot x1 + a12 \cdot x2 + \cdots + a1n \cdot xn = b1 \\ a21 \cdot x1 + a22 \cdot x2 + \cdots + a2n \cdot xn = b2 \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ am1 \cdot x1 + am2 \cdot x2 + \cdots + amn \cdot xn = bm \end{cases}$$

można zapisać w postaci macierzy współczynników, która wyglądałaby tak:

$$A = \begin{bmatrix} a11 & a12 & \cdots & a1n \\ a21 & a22 & \cdots & a2n \\ \vdots & \vdots & \vdots & \vdots \\ am1 & am2 & \cdots & amn \end{bmatrix}$$

wektora wyrazów wolnych, który wyglądałby tak:

$$B = \begin{bmatrix} b1 \\ b2 \\ \cdots \\ bm \end{bmatrix}$$

oraz wektora niewiadomych, który wyglądałby tak:

$$X = \begin{bmatrix} x1 \\ x2 \\ \cdots \\ xn \end{bmatrix}$$

Stosując powyższe oznaczenia, układ równań można zapisać w postaci macierzowej

$$A \cdot X = B$$

czyli:

$$\begin{bmatrix} a11 & a12 & \cdots & a1n \\ a21 & a22 & \cdots & a2n \\ \vdots & \vdots & \vdots & \vdots \\ am1 & am2 & \cdots & amn \end{bmatrix} \cdot \begin{bmatrix} x1 \\ x2 \\ \cdots \\ xn \end{bmatrix} = \begin{bmatrix} b1 \\ b2 \\ \cdots \\ bm \end{bmatrix}$$

1.4 Twierdzenie Cramera

Jeżeli wyznacznik macierzy współczynników układu jest różny od zera (detA != 0), to układ równań liniowych jest oznaczony i rozwiązanie jest dane wzorami:

$$x_i = \frac{W_i}{W}$$

gdzie:

W = detA - jest to wyznacznik macierzy współczynników. **(wyznacznik główny)**

W_i = jest to wyznacznik z macierzy, która powstaje z macierzy A, przez zastąpienie kolumny współczynników niewiadomej x_i przez kolumnę wyrazów wolnych.

Jeżeli detA = 0 i W_i = 0 (dla każdego i) - układ równań liniowych jest nieoznaczony
Jeżeli detA = 0 i istnieje W_i != 0 - układ równań liniowych jest sprzeczny

1.5 Problem

Za zadanie postawiono obliczanie symbolicznie oraz numerycznie układu równań liniowych dla trzech oraz dwóch niewiadomych oraz rysowanie grafów, które pokazałyby sposób obliczania wyznacznika głównego oraz wyznaczników z macierzy, która powstaje z macierzy A, przez zastąpienie kolumny współczynników niewiadomej x_i przez kolumnę wyrazów wolnych, potrzebnych do obliczenie niewiadomych za pomocą metody Cramera. (1.4). W wypadku znalezienia w układzie współczynnika, który nie jest liczbą program ma wyświetlać sposób obliczania układu symbolicznie, jeżeli jednak nie wystąpią współczynniki, które są literami od a do z, pomijając x, który ma szczególne znaczenie, to program zacznie obliczyć numerycznie.

2. Działanie programu

Aby uruchomić program wystarczy zainstalować narzędzie pipenv i uruchomić program za pomocą komendy ‘pipenv run python main’. Jednak wcześniej trzeba stworzyć folder ‘graphs’, w którym będą znajdować się nasze grafy zaraz po zakończeniu programu. Same grafy będa zapisywane w formacie .png wraz z zasadą W_i.png, gdzie ‘i’ to numer wyznacznika. W pliku W_0.png zawsze będzie zapisywany wyznacznik główny zaś we wszyskich kolejnych będą pozostałe wyznaczniki potrzebne to obliczenia niewiadomych. Mając już folder graphs można zajrzeć do pliku main.py, gdzie będzie jedna metoda calculate, która będzie potrzebowała dwóch argumentów. Pierwszy argument to input, który prowadzi do pliku z układem równań natomiast drugim argumentem jest output, który wskazuje w jakim pliku program ma zapisać wyniki. Wynik programu będzie zapisany w formacie:

x1 = …
x2 = …
....
xn = …

2.1 Plik z układem równań

Plik z układem równań musi posiadać szczególną składnie, aby program wykonywał się poprawnie:

- każde równanie musi posiadać na swoim końcu znak ‘;’, oprócz ostatniego który takiego znaku nie potrzebuje.
- każda niewiadoma i jej współczynnik muszą być koło siebie bez odseparowania znakiem spacji.
- współczynnik musi mieć znak przy sobie. np. +2ax1
- pomiędzy każdą grupą niewiadoma i współczynnik musi być spacja z lewej jak i z prawej strony
- koło znaku równości może być znak spacji ale także można to pominąć

Przykład układu równań w programie:

+2x0 +1x1 +1x2 = 9;
1x0 +2x1 +1x2 = 14;
1x0 +1x1 +2x2 = 19

lub

+2x0 +1x1 +1x2 = 9;1x0 +2x1 +1x2 = 14;1x0 +1x1 +2x2 = 19

2.2 Wynik

Wynikiem programu jest jeden plik z wynikami symbolicznymi lub numerycznymi oraz pliki w formacie png, które zawierają grafy pokazujące obliczenia wyznacznika.

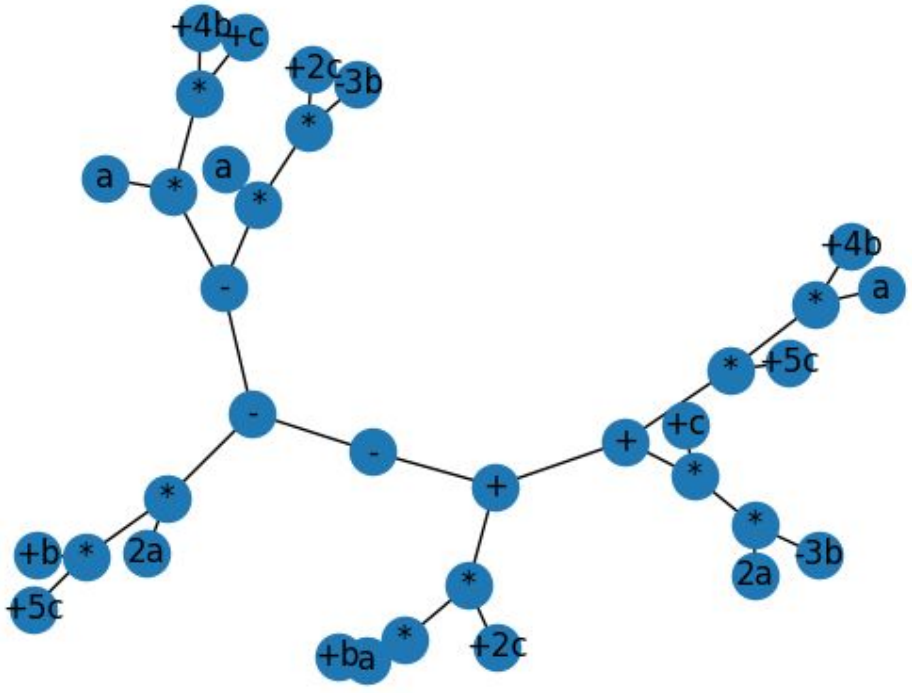
Przykład wyniku symbolicznego:

```
1 x0 = (( 9 * +4b * +5c) + ( 14 * -3b * +c) + ( 19 * +b * +2c)-(+c * +4b *  19) - (+2c *
  -3b *  9) - (+5c * +b *  14))/((a * +4b * +5c) + (2a * -3b * +c) + (a * +b * +2c)-(+c
  * +4b * a) - (+2c * -3b * a) - (+5c * +b * 2a))
2 x1 = ((a *  14 * +5c) + (2a *  19 * +c) + (a *  9 * +2c)-(+c *  14 * a) - (+2c *  19 *
  a) - (+5c *  9 * 2a))/((a * +4b * +5c) + (2a * -3b * +c) + (a * +b * +2c)-(+c * +4b *
  a) - (+2c * -3b * a) - (+5c * +b * 2a))
3 x2 = ((a * +4b *  19) + (2a * -3b *  9) + (a * +b *  14)-( 9 * +4b * a) - ( 14 * -3b *
  a) - ( 19 * +b * 2a))/((a * +4b * +5c) + (2a * -3b * +c) + (a * +b * +2c)-(+c * +4b *
  a) - (+2c * -3b * a) - (+5c * +b * 2a))
```

Przykład wyniku numerycznego:

```
1 x0 = 27.833333333333332
2 x1 = -7.333333333333333
3 x2 = -6.166666666666667
```

Przykład grafu:



3. Opis funkcji

3.1 parse(equation):

zmienia string w postaci np. ‘2x0 +3x1 = 4;3x0 -1.5x1 = 0.5’ do postaci list, która zawiera listę w postaci [[współczynniki], wyraz wolny] oraz zmienia liczby z typu string na typ float.

equation - string w postaci np. ‘a11x0 +a12x1 = b1;a21x0 +a22x1 = b2’

Przykład:
equation = ‘2x0 +3x1 = 4;3x0 -1.5x1 = 0.5’
parse(equation) => [[[2, 3], 4], [[3, -1.5], 0.5]]

3.2 parse_symbolic(equation)

zmienia string w postaci np. ‘2x0 +3x1 = 4;3x0 -1.5x1 = 0.5’ do postaci list, która zawiera listę w postaci [[[współczynniki], wyraz wolny], ...]

equation - string w postaci np. ‘a11x0 +a12x1 = b1;a21x0 +a22x1 = b2’

Przykład:
equation = ‘2x0 +3x1 = 4;3x0 -1.5x1 = 0.5’
parse(equation) => [[[‘2’, ‘3’], ‘4’], [[‘3’, ‘-1.5’], ‘0.5’]]

3.3 get_matrix_and_vector(equations)

zmienia listę, która jest w postaci [[[współczynniki], wyraz wolny], ...] na macierz główną oraz wektor wyrazów wolnych.

equations - lista w postaci [[[współczynniki], wyraz wolny], ...]

Przykład:
equations = [[[a11, a12], b1], [[a21, a22], b2]]
get_matrix_and_vector(equations) => matrix = [[a11, a12], [a21, a22]] , vector = [b1, b2]

3.4 replace_column(matrix, column, vactor)

zwraca nową macierz ze zmienioną kolumną o numerze column przez zmienną vector.

matrix - macierz nxn
column - numer kolumny do zamiany
vector - wektor wyrazów wolnych

Przykład:

matrix = [[a11, a12], [a21, a22]]
vector = [b1, b2]
column = 0
replace_column(matrix, column, vactor) => new_matrix = [[b1, b2], [a21, a22]]

3.5 get_det_equation(matrix)
zwraca listę w formie:
[[[elementy do dodania: [elementy do mnożenia, ...]], [elementy do odejmowania: [elementy do mnożenia, ...]]]
,która pozwala obliczyć wyznacznik.

matrix - macierz nxn

Przykład:
matrix = [[a11, a12], [a21, a22]]
get_det_equation(matrix) => [[[a11, a22]], [[a12, a21]]]

3.6 calculate_determinant(matrix)
zwraca wynik obliczenia wyznacznika danej macierzy oraz liste w formie:
[[[elementy do dodania: [elementy do mnożenia, ...]], [elementy do odejmowania: [elementy do mnożenia, ...]]]
,która pozwala obliczyć wyznacznik

matrix - macierz nxn

Przykład:
matrix = [[a11, a12], [a21, a22]]
calculate_determinant(matrix) => result = a11 * a22 - a12 * a21, det_equation = [[[a11, a22]], [[a12, a21]]]

3.7 calculate_dets(matrix, vector)
zwraca wyznacznik macierzy głównej, liste wyznaczników z macierzy, która powstaje z macierzy matrix, przez zastąpienie kolumny współczynników niewiadomej x_i przez kolumnę wyrazów wolnych oraz liste w formie:
[[[elementy do dodania: [elementy do mnożenia, ...]], [elementy do odejmowania: [elementy do mnożenia, ...]], ...]
,która pozwoliła obliczyć wyznaczniki

matrix - macierz główna
vector - wektor wyrazów wolnych

Przykład:
matrix = [[a11, a12], [a21, a22]]
vector = [b1, b2]

calculate_dets(matrix, vector) =>
main_det = a11 * a22 - a12 * a21
sub_dets = [b1 * a22 - a12 * b2, a11 * b2 - b1 * a21]
det_equation = [[[a11, a22]], [[a12, a21]]]

3.8 calculate_equation(main_det, sub_deta)
zwraca słownik z obliczonymi niewiadomymi z wyznacznika głównego oraz wyznaczników z macierzy, która powstaje z macierzy głównej, przez zastąpienie kolumny współczynników niewiadomej x_i przez kolumnę wyrazów wolnych.

main_det - wyznacznik macierzy głównej
sub_dets - lista z wyznacznikami otrzymanymi poprzez zamianę kolumny ‘i’ przez wektor z wyrazami wolnymi.

Przykład:
main_det = a11 * a22 - a12 * a21
sub_dets = [b1 * a22 - a12 * b2, a11 * b2 - b1 * a21]
calculate_equation(main_det, sub_deta) =>
{
x1 = (b1 * a22 - a12 * b2) / (a11 * a22 - a12 * a21)
x2 = (a11 * b2 - b1 * a21) / (a11 * a22 - a12 * a21)
}

3.9 get_matrix_and_vector_from_file(file_content)
zamienia dane z pliku na macierz oraz wektor do obliczeń numerycznych.

file_content - string z układem równań

Przykład:
file_content = ‘a11x0 +a12x1 = b1;a21x0 +a22x1 = b2’
get_matrix_and_vector_from_file(file_content) =>
matrix = [[a11, a12], [a21, a22]] , vector = [b1, b2]

3.10 get_matrix_and_vector_from_file_symbolic(file_content)
zamienia dane z pliku na macierz oraz wektor do obliczeń symbolicznych.

file_content - string z układem równań
Przykład:
file_content = ‘a11x0 +a12x1 = b1;a21x0 +a22x1 = b2’
get_matrix_and_vector_from_file(file_content) =>
matrix = [[‘a11’, ‘a12’], [‘a21’, ‘a22’]] , vector = [‘b1’, ‘b2’]

3.11 put_result_to_file(path, result)

umieszcza wynik, który jest w formie słownika do pliku, który jest pod ścieżką podaną w funkcji

path - ścieżka do pliku
result - słownik z wynikami

Przykład wyniku:
{
x1 = (b1 * a22 - a12 * b2) / (a11 * a22 - a12 * a21)
x2 = (a11 * b2 - b1 * a21) / (a11 * a22 - a12 * a21)
}

3.12 get_det_equations_symbolic(matrix, vactor)
zwraca listę, która zawiera listy w formie [[[a11, a22, ...]], [[a12, a21,..]]], które służą do symbolicznego obliczenia wyznaczników macierzy oraz macierzy uzyskanych z podmiany kolumn ‘i’ z wektorem wyrazów wolnych.

matrix - macierz nxn, macierz główna
vector - wektor wyrazów wolnych

Przykład:
matrix - matrix = [[‘a11’, ‘a12’], [‘a21’, ‘a22’]]
vector = [b1, b2]
get_det_equations_symbolic(matrix, vactor) => result = [[[‘a11’, ‘a22’],[‘a12,a21]], [[‘b1’, ‘a22’],[‘a12’, ‘b2’]], [[‘a11’, ‘b1’],[‘b2’, ‘a21’]]]

3.13 calculate_equation_symbolic(det_eqs)
zwraca słownik z wynikami obliczonymi symbolicznie z wyznaczników otrzymanych w formie [[[a11, a22, ...]], [[a12, a21,..]]].