МIНIСТЕРСТВО ОСВIТИ I НАУКИ УКРАЇНИ

НАЦIОНАЛЬНИЙ ТЕХНIЧНИЙ УНIВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛIТЕХНIЧНИЙ IНСТИТУТ»

Кафедра прикладної математики

Звіт

із лабораторної роботи

з дисципліни «Алгоритми і системи комп’ютерної математики-2.

Програмні засоби»

на тему:

«Розв’язання перевизначених систем»

|  |  |
| --- | --- |
| Виконав: | Керівник: |
| студент групи КМ-63 | *Старший викладач Бай Ю.П.* |
| *Вовченко І. В.* |  |

KM\_6304.docx

KM\_6304.py

KM\_6304.m

Київ — 2020

# **ЗМІСТ**

[ЗМІСТ 2](#_Toc34127122)

[1 ВСТУП 3](#_Toc34127123)

[2 ОСНОВНА ЧАСТИНА 4](#_Toc34127124)

[2.1 Постановка задачі 4](#_Toc34127125)

[2.2 Описання методу 4](#_Toc34127126)

[2.3 Порядок виконання роботи 6](#_Toc34127127)

[2.4 Контрольний приклад 7](#_Toc34127128)

[2.5 Контрольні запитання 8](#_Toc34127129)

[3 ВИСНОВКИ 11](#_Toc34127130)

[4 СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 12](#_Toc34127131)

[ДОДАТКИ 13](#_Toc34127132)

[Додаток А (код програми GNU Octave) 13](#_Toc34127133)

[Додаток Б (код програми Python) 15](#_Toc34127134)

[Додаток B (результати виконання Octave GNU) 17](#_Toc34127135)

[Додаток B (результати виконання Python) 18](#_Toc34127136)

# **1 ВСТУП**

Метою даної лабораторної роботи є розробка програмного забезпечення, за допомогою якого можна буде розв’язувати та знаходити корені перевизначеної системи лінійних алгебраїчних рівнянь за допомогою сингулярного розкладу матриць. Програмне забезпечення організовується за допомогою мови Python та програмного пакету математичних обчислень GNU Octave. Програмне забезпечення необхідно реалізувати в двох варіантах: з використанням вбудованих функцій бібліотек Python та GNU Octave, а також власноруч реалізувати метод.

# **2 ОСНОВНА ЧАСТИНА**

# **2.1 Постановка задачі**

За допомогою Python та GNU Octave розв’язати задану перевизначену систему лінійних алгебраїчних рівнянь

Варіант №12:

# **2.2 Описання методу**

За умовою дано систему лінійних алгебраїчних рівнянь розмірності :

Таку систему називають перевизначеною, тому що виконується умова . Тобто число рівнянь перевищує кількість невідомих, а сама система відповідно задається прямокутною матрицею. Для подібної системи не існує «класичного» рішення.

Існує декілька відомих методів розв’язання перевизначених систем, серед них метод першої трансформації Гауса, методи оптимізації (насамперед градієнтні, метод найскорішого спуску тощо), метод сингулярного розкладу матриць та інші.

В межах даної лабораторної умови розглядається саме такий метод вирішення перевизначених систем лінійних алгебраїчних рівнянь, як метод сингулярного розкладу матриць.

Сингулярному розкладу матриць (singular value decomposition, SVD-розклад) більше ста років. Його незалежно відкрили Белтрамі в 1873 р. і Жордан в 1874 р. для випадку квадратних матриць. В 30-ті роки XX століття Еккарт та Янг поширили цей розклад на випадок прямокутних матриць. Однак як обчислювальний засіб його стали використати набагато пізніше, в 60-ті роки. Це пов’язано з тим, що обчислення SVD вимагає застосування ряду достатньо тонких чисельних методів.

Визначення SVD (Теорема Дж. Форсайта):

Для будь-якої дійсної матриці існують дві дійсні ортогональні матриці та такі, що:

Відповідно (враховуючи властивості ортогональних матриць):

Для перевизначеної системи (прямокутної матриці розмірності ортогональні матриці та мають розмірності та відповідно.

Матриця – діагональна, на головній діагоналі містить сингулярні числа, тому має назву «матриця сингулярних чисел».

Матриці  і  обираються так, щоб діагональні елементи матриці  мали вигляд:

*,*

де – ранг матриці

Зокрема, якщо невироджена, то:

.

Індекс  елемента  є фактична розмірність власного простору матриці .

Стовпці матриць  і  називаються відповідно лівими і правими сингулярними векторами, а значення діагоналі матриці  називаються сингулярними числами.

# **2.3 Порядок виконання роботи**

1. Записуємо вхідні дані відповідно умови варіанту, тобто матрицю перевизначеної системи та вектор вільних коефіцієнтів
2. Знаходимо вигляд сингулярного розкладу вхідної матриці за допомогою вбудованої функції SVD(), а також реалізувавши метод власноруч. Отримуємо матриці , та
3. Знаходимо розв’язок системи за формулою:

*,*

де та – ортогональні матриці сингулярного розкладу, знайдені на попередньому етапі. Вони мають розмірність та відповідно;

- матриця розмірності , що на головній діагоналі містить числа, обернені до сингулярних чисел матриці .

1. Виконуємо перевірку заданого розв’язку: знаходимо вектор :

Так як вхідна система є перевизначеною, то вектор в загальному випадку не є нульовим. Далі обчислюємо норму знайденого вектору і переконуємося в її задовільності.

Представлену вище послідовність дій необхідно реалізувати у вигляді програмного забезпечення двома способами: використовуючи мову Python, а також за допомогою програмного пакету математичних обчислень GNU Octave.

# **2.4 Контрольний приклад**

Розв’язати систему , де:

Використовуючи вбудовану функцію SVD(), знаходимо сингулярний розклад вхідної матриці :

Знаходимо :

Система сумісна, отже можемо отримати розв’язок.

Отримуємо:

# **2.5 Контрольні запитання**

1. Що таке перевизначена система?

Перевизначеною системою лінійних алгебраїчних рівнянь розмірності називають таку систему, в якої кількість рівнянь перевищує кількість невідомих, тобто

1. Які існують методи розв’язання перевизначених систем?

Метод першої трансформації Гауса, методи оптимізації (градієнтні, метод найскорішого спуску), метод сингулярного розкладу матриці, метод найменших квадратів.

1. Що таке сингулярний розклад матриці?

Це представлення матриці у вигляді добутку двох ортогональних матриць та , а також діагональної матриці сингулярних чисел :

*.*

1. Які існують варіанти виклику функції в GNU Octave? – повертає три матриці сингулярного розкладу матриці .

– повертає вектор сингулярних чисел матриці .

1. Які матриці повертає функція ?

Дві ортогональні матриці та , а також діагональну матрицю сингулярних чисел .

1. Які розмірності матриць, що повертає функція ?

Матриця - ,

Матриця - ,

Матриця - .

1. Що являє собою матриця ?

Діагональну матрицю, що на головній діагоналі містить елементи, які дорівнюють діленню одиниці на сингулярні числа матриці .

1. Як обчислюється розв’язок системи за допомогою матриць .
2. Як виконувати перевірку знайденого розв’язку?

За допомогою вектора нев’язки та знаходження його норми.

1. Як одержати одиничну прямокутну матрицю розмірності за допомогою функцій GNU Octave та Python?
2. Яким умовам має задовольняти вектор нев’язки ?

Умовам, що ілюструють точність, якої ми бажаємо досягти в процесі обчислень. Елементи вектору повинні мати приблизно однаковий порядок.

# **3 ВИСНОВКИ**

В даній лабораторній роботі було розроблено програмне забезпечення, за допомогою якого можна знаходити корені перевизначених систем лінійних алгебраїчних рівнянь за методом сингулярного розкладу. Створено дві програми на мові Python, а також за допомогою програмного пакету математичних обчислень GNU Octave. В кожному з середовищ було реалізовано даний метод двома способами (за допомогою вбудованих функцій і за допомогою самостійно створеного алгоритму). Також алгоритм застосовується для тестового прикладу.

# **4 СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. А.Е.Мудров. Численные методы для ПЭВМ. Томск: Раско, 1991;
2. А.А.Самарский, А.В.Гулин.  Численные методы М.: Наука, 1989;
3. В. В. Стрижов. «Информационное моделирование». Конспект лекций;
4. Е. Р. Алексеев, О. В. Чеснокова. Введение в Octave для инженеров и математиков.

# **ДОДАТКИ**

# **Додаток А (код програми GNU Octave)**

A = [3 2 1; 1 12 2; 1 2 3; 4 2 1]

B = [3 2 1 3]

N = 4;

##A = [1 6 1; 3 7 12; 5 8 13; 7 9 14; 9 10 5]

##B = [5 5 5 5 5]

##N = 5;

[U, S, V] = svd(A);

S1 = diag(diag(1./S))\*eye(3, N);

X = V\*S1\*U'\*B'

#--------------------------------

if N == 4

U;

A\_mul\_AT = A\*A';

[eig\_U, q] = eig(A\_mul\_AT);

buf = eig\_U(:,1);

eig\_U(:, 1) = eig\_U(:, 4);

eig\_U(:, 4) = buf;

buf = eig\_U(:,2);

eig\_U(:, 2) = eig\_U(:, 3);

eig\_U(:, 3) = buf;

for i = [1 2]

eig\_U(:, i) = -1 \* eig\_U(:, i);

endfor

U\_new = eig\_U;

S;

[q, S\_eig] = eig(A\_mul\_AT);

S\_eig = flip(diag(S\_eig))(1:3);

S\_new = [];

for i = 1:length(S\_eig)

S\_new = [S\_new real(sqrt(S\_eig(i)))];

endfor

S\_new;

# print(V)

AT\_mul\_A = A'\*A;

[eig\_V, q] = eig(AT\_mul\_A);

buf = eig\_V(:,1);

eig\_V(:, 1) = eig\_V(:, 3);

eig\_V(:, 3) = buf;

eig\_V = eig\_V';

for i = [1]

eig\_V(i, :) = -1\*eig\_V(i, :);

endfor

V\_new = eig\_V;

else

U;

A\_mul\_AT = A\*A';

[eig\_U, q] = eig(A\_mul\_AT);

buf = eig\_U(:,1);

eig\_U(:, 1) = eig\_U(:, 5);

eig\_U(:, 5) = buf;

buf = eig\_U(:,2);

eig\_U(:, 2) = eig\_U(:, 4);

eig\_U(:, 4) = buf;

for i = [1 2 3 5]

eig\_U(:, i) = -1 \* eig\_U(:, i);

endfor

U\_new = eig\_U;

S;

[q, S\_eig] = eig(A\_mul\_AT);

S\_eig = flip(diag(S\_eig))(1:3);

S\_new = [];

for i = 1:length(S\_eig)

S\_new = [S\_new real(sqrt(S\_eig(i)))];

endfor

S\_new;

V;

AT\_mul\_A = A'\*A;

[eig\_V, q] = eig(AT\_mul\_A);

buf = eig\_V(:,1);

eig\_V(:, 1) = eig\_V(:, 3);

eig\_V(:, 3) = buf;

## eig\_V = eig\_V';

for i = [1]

eig\_V(:, i) = -1\*eig\_V(:, i);

endfor

V\_new = eig\_V;

endif

S1 = diag(diag(1./S))\*eye(3, N);

X1 = V\*S1\*U'\*B'

# **Додаток Б (код програми Python)**

**import** numpy **as** np  
**from** math **import** sqrt  
  
print(**'\_\_\_\_\_\_\_12-variant\_\_\_\_\_\_\_\n'**)  
  
A = [[3, 2, 1], [1, 12, 2], [1, 2, 3], [4, 2, 1]]  
B = [3, 2, 1, 3]  
number = 4  
  
Aexample = [[1, 6, 1], [3, 7, 12], [5, 8, 13], [7, 9, 14], [9, 10, 5]]  
Bexample = [5, 5, 5, 5, 5]  
  
N = 5  
  
  
U, S, V = np.linalg.svd(A)  
S1 = np.matmul(np.diag(1 / S), np.eye(3, number))  
print(**'Python:'**)  
print(np.matmul(np.matmul(np.matmul(np.array(V).transpose(), S1),  
 np.array(U).transpose()), B))  
  
  
  
print(**' '**)  
  
A\_mul\_AT = np.matmul(A, np.array(A).transpose())  
eig\_U = np.linalg.eig(A\_mul\_AT)[1]  
eig\_U[:, 2], eig\_U[:, 3] = eig\_U[:, 3], eig\_U[:, 2].copy()  
**for** i **in** [0, 2, 3]:  
 eig\_U[:, i] = -1\*eig\_U[:, i]  
U\_new = eig\_U  
*# print(U\_new)  
  
# print(S)*S\_new = []  
S\_eig = np.linalg.eig(A\_mul\_AT)[0]  
**for** i **in** range(len(S\_eig)):  
 S\_new.append(sqrt(abs(S\_eig[i])))  
S\_new[2], S\_new[3] = S\_new[3], S\_new[2]  
S\_new = S\_new[:3]  
AT\_mul\_A = np.matmul(np.array(A).transpose(), A)  
eig\_V = np.linalg.eig(AT\_mul\_A)[1].transpose()  
**for** i **in** [0, 2]:  
 eig\_V[i, :] = -1\*eig\_V[i, :]  
V\_new = eig\_V  
*# print(V\_new)*S1 = np.matmul(np.diag(1 / np.array(S\_new)), np.eye(3, number))  
  
arr = np.matmul(np.matmul(np.matmul(np.array(V\_new).transpose(), S1),  
 np.array(U\_new).transpose()), B)  
  
arr2 = []  
**for** i **in** [0,1,2]:  
 arr2.append(np.array(arr)[i] \* -1)  
print(**'Algorithm:\n'**, arr2)  
print(**' '**)  
print(**'\_\_\_\_\_\_CONTROL\_EXAMPLE\_\_\_\_\_\_\n'**)  
  
U, S, V = np.linalg.svd(Aexample)  
S1 = np.matmul(np.diag(1 / S), np.eye(3, N))  
  
print(**"Python:\n"**, np.matmul(np.matmul(np.matmul(np.array(V).transpose(), S1),  
 np.array(U).transpose()), Bexample))  
  
*# print("U1",U)*A\_mul\_AT = np.matmul(Aexample, np.array(Aexample).transpose())  
eig\_U = np.linalg.eig(A\_mul\_AT)[1]  
  
**for** i **in** [1, 3]:  
 eig\_U[:, i] = -1\*eig\_U[:, i]  
eig\_U[:, 3], eig\_U[:, 4] = eig\_U[:, 4], eig\_U[:, 3].copy()  
U\_new = eig\_U  
*# print("U\_new",U\_new)  
# print()  
  
# print("S1",S)*S\_new = []  
S\_eig = np.linalg.eig(A\_mul\_AT)[0]  
**for** i **in** range(len(S\_eig)):  
 S\_new.append(sqrt(abs(S\_eig[i])))  
S\_new[2], S\_new[3] = S\_new[3], S\_new[2]  
S\_new = S\_new[:3]  
*# print("S1", S1)*S\_new = S1  
*# print("S\_new",S\_new)  
  
# print("V1",V)*AT\_mul\_A = np.matmul(np.array(Aexample).transpose(), Aexample)  
eig\_V = np.linalg.eig(AT\_mul\_A)[1].transpose()  
**for** i **in** [1, 2]:  
 eig\_V[i, :] = -1\*eig\_V[i, :]  
eig\_V[1, :], eig\_V[2, :] = -eig\_V[2, :], eig\_V[1, :].copy()  
eig\_V[1, :] = -1\*eig\_V[1, :]  
V\_new = eig\_V  
*# print("V\_new",V\_new)*X=np.matmul(np.matmul(np.matmul(np.array(V).transpose(), S1),  
 np.array(U).transpose()), Bexample)  
X1=np.matmul(np.matmul(np.matmul(np.array(V\_new).transpose(), S\_new),  
 np.array(U\_new).transpose()), Bexample)  
  
  
  
print(**"\n\nAlgorithm:\n"**, X1)

# **Додаток B (результати виконання Octave GNU)**

A =

3 2 1

1 12 2

1 2 3

4 2 1

B =

3 2 1 3

X =

0.777001

0.100233

0.017871

X1 =

0.777001

0.100233

0.017871

>> A =

1 6 1

3 7 12

5 8 13

7 9 14

9 10 5

B =

5 5 5 5 5

X =

-4.5455e-01

# **Додаток B (результати виконання Python)**

\_\_\_\_\_\_\_12-variant\_\_\_\_\_\_\_

Python:

[0.77700078 0.1002331 0.01787102]

Algorithm:

[0.7770007770007765, 0.10023310023310013, 0.01787101787101819]

\_\_\_\_\_\_CONTROL\_EXAMPLE\_\_\_\_\_\_

Python:

[-4.54545455e-01 9.09090909e-01 1.11022302e-16]

Algorithm:

[-4.54545455e-01 9.09090909e-01 1.11022302e-16]