ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

Assignment of LINEAR ALGEBRA Project 2

Lecturer: Hoàng Hải Hà
Đậu Thế Phiệt

Class: CC06 - **Group:** 12

222 - Linear Algebra

Sinh viên thực hiện	**Mã số sinh viên**
Nguyễn Lê Minh Khôi	2252374
Trần Nguyễn Thế Nhật	2252556
Nguyễn Tiến Hưng	2252280

Thành phố Hồ Chí Minh – 2023

TABLE OF CONTENTS

PREFACE

Linear Algebra is an important general subject for university students HCMUT in particular and students of science and technology - public sector technology in general. Therefore, devoting a certain amount of time to this subject and practice is indispensable to help students have a solid basis in subjects Science and technology and make a premise to study well in other subjects in the training program.

The development of computer science was born, which greatly supported the development of physics subjects. The application of informatics in the process of interpreting databases of maths, solving maths problems has shortened the time spent and brought higher efficiency. As we all know, Matlab application software has solved many problems . Therefore, learning Matlab and applying Matlab in the practice of the subject Linear Algebra is very important and highly urgent.

During the process of writing the above assignment, our team received much care and support, dedicated help from teachers, brothers and sisters and friends. In addition, the group would also like to express their most sincere gratitude to Mr. Dau The Phiet, is the instructor for this assignment. Thanks to your wholeheartedly instructing, the group was able to complete the assignment on time and solve problems well. Your guidance has been the guideline for all actions of the group and maximized the supportive relationship between teachers and students in the educational environment. Also, on this occasion, we would like to thank Ms.Hoang Hai Ha for her enthusiastic teaching of the theory very carefully so that we have a solid foundation to solve this maths problem. Finally, once again, I would like to express my deep gratitude to the teachers and everyone who takes time to instruct the group. This is the belief, a great source of motivation for the group that this result can be achieved.

REQUIREMENTS OF THE TOPIC

1.1. Requirements

1) Students should have basic programming knowledge of MATLAB.

2) Learn about symbolic calculation and graphical interpretation in MATLAB.

3) Have a basis knowledge about Linear Algebra

1.2. Task

Problem 1: The cryptogram below was encoded with a 2×2 matrix. The last word of the message is _ _SUE. What is the message?

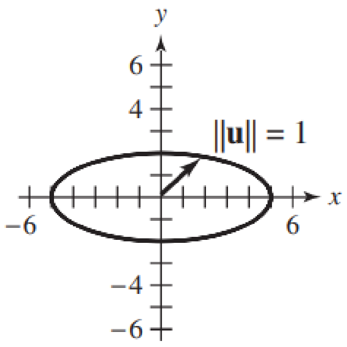
5 2 25 11 -2 -7 -15 -15 32 14 -8 -13 38 19 -19 -19 37 16

Problem 2: Construct an inner product in R^n . In that inner product, write a program to input any number of vectors in Rn and return the orthogonal basis and orthonormal basis of the subspace spanned by these vectors. (Use Gram - Schmidt process). From that, given any vector in R^n , find the coordinates in that basis and find the length of the vector.

Problem 3: In R^2,the weighted inner product is given by

$$\langle x , y \rangle = ax_1y_1 + bx_2y_2$$

where a and b are positive. Find a weighted inner product such that the graph represents a unit circle as



In that inner product space, reflect that unit circle about an input plane.

MATLAB

2.1. Introduction to Matlab

2.1.1. Overview of Matlab

Matlab (short for matrix laboratory) is a four-level high-level programming language generation environment for arithmetic calculations, visualization and programming. Developed by MathWorks. Matlab allows manipulating matrices, drawing graphs with functions and data, displaying implementing algorithms, creating user interfaces, including C, C++, Java and Fortran; data analysis, algorithm development, prototyping and applications. Matlab has a lot of Maths commands and functions to help you in doing calculations, drawing common figures and diagrams, and implementing methods to calculate.

2.1.2. Commonly used functions in Matlab

Command	*Syntax*	*Meaning*

Disp	*disp(x)*
	disp('chuoi tu')
		Display the contents of the array or chain

Syms	*syms x*	*Declare the variable x as a symbol variable*
Input	*x=input('ten bien')*	*Show command prompt and wait for input*
Plot	*plot(x,y)*	*Generate xy . graph*
Title	*title('ten do thi')*	*Graph title*
Legend	*legend('vi tri')*	*Add a note to the graph*

Label	*xlabel('ten)*
	yabel('ten)
		Add labels to the x-axis
		Add labels to the y-axis

PROBLEM 1

3.1 Introduction

Encryption dates back approximately 4000 years. Historical accounts indicate that the Chinese, Egyptians, Indian, and Greek encrypted messages in some way for various purposes. One famous encryption scheme is called the Caesar cipher, also called a substitution cipher, used by Julius Caesar, involved shifting letters in the alphabet, such as replacing A by C, B by D, C by E, etc, to encode a message. Substitution ciphers are too simple in design to be considered secure today.

In the middle ages, European nations began to use encryption. A variety of encryption methods were used in the US from the Revolutionary War, through the Civil War, and on into to modern times.

Applications of mathematical theory and methods to encryption became widespread in military usage in the 20th century. The military would encode messages before sending and the recipient would decode the message, in order to send information about military operations in a manner that kept the information safe if the message was intercepted. In World War II, encryption played an important role, as both Allied and Axis powers sent encrypted messages and devoted significant resources to strengthening their own encryption while also trying to break the opposition’s encryption.

$$\underbrace{\begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 2 & 1 & 5 \end{bmatrix}}_{\text{Key}} \quad \underbrace{\begin{bmatrix} 31 & 82 & 100 & 8 \\ 50 & 113 & 153 & 16 \\ 56 & 116 & 129 & 16 \end{bmatrix}}_{\text{Encrypted Message}}$$

***Figure 1:** Example of using cryptogram in convert message*

3.2 Theoretical basis

To use matrices in encoding and decoding secret messages, our procedure is as follows.

We first convert the secret message into a string of numbers by arbitrarily assigning a number to each letter of the message. Next we convert this string of numbers into a new set of numbers by multiplying the string by a square matrix of our choice that has an inverse. This new set of numbers represents the coded message.

To decode the message, we take the string of coded numbers and multiply it by the inverse of the matrix to get the original string of numbers. Finally, by associating the numbers with their corresponding letters, we obtain the original message.

TO ENCODE A MESSAGE

1. Divide the letters of the message into groups of two or three.

2. Convert each group into a string of numbers by assigning a number to each letter of the message. Remember to assign letters to blank spaces.

3. Convert each group of numbers into column matrices.

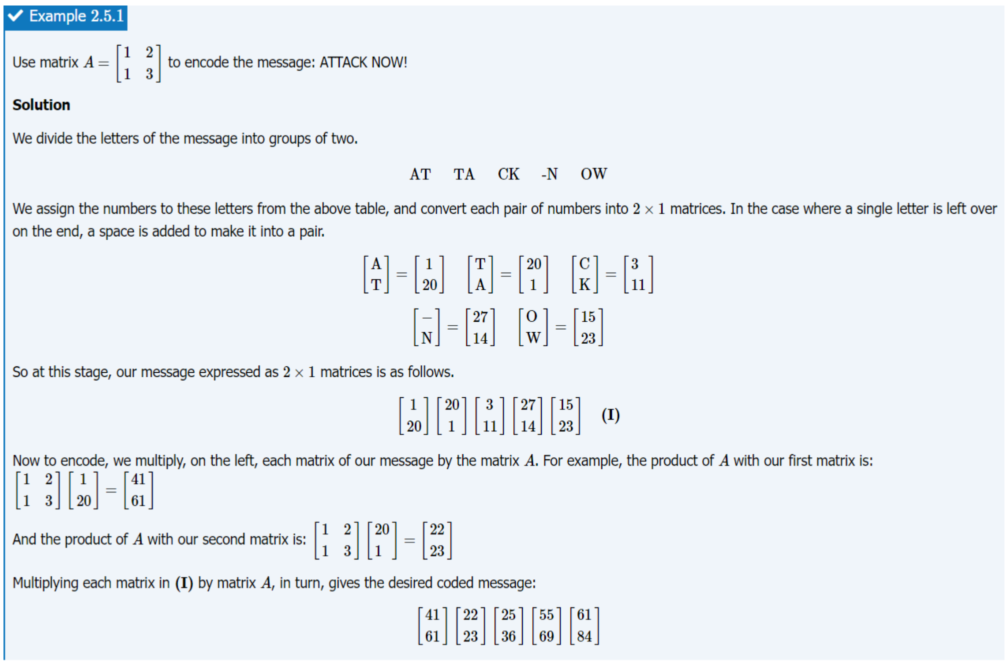
4. Convert these column matrices into a new set of column matrices by multiplying them with a compatible square matrix of your choice that has an inverse. This new set of numbers or matrices represents the coded message.

TO DECODE A MESSAGE

1. Take the string of coded numbers and multiply it by the inverse of the matrix that was used to encode the message.
2. Associate the numbers with their corresponding letters.

Assign a number to each letter in the alphabet

A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**
1	2	3	4	5	6	7	8	9	10	11	12	13
N	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
14	15	16	17	18	19	20	21	22	23	24	25	26
_												
0												



✓ Example 2.5.1

Use matrix $A = \begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix}$ to encode the message: ATTACK NOW!

Solution

We divide the letters of the message into groups of two.

$$\text{AT} \quad \text{TA} \quad \text{CK} \quad \text{-N} \quad \text{OW}$$

We assign the numbers to these letters from the above table, and convert each pair of numbers into 2×1 matrices. In the case where a single letter is left over on the end, a space is added to make it into a pair.

$$\begin{bmatrix} \text{A} \\ \text{T} \end{bmatrix} = \begin{bmatrix} 1 \\ 20 \end{bmatrix} \quad \begin{bmatrix} \text{T} \\ \text{A} \end{bmatrix} = \begin{bmatrix} 20 \\ 1 \end{bmatrix} \quad \begin{bmatrix} \text{C} \\ \text{K} \end{bmatrix} = \begin{bmatrix} 3 \\ 11 \end{bmatrix}$$

$$\begin{bmatrix} - \\ \text{N} \end{bmatrix} = \begin{bmatrix} 27 \\ 14 \end{bmatrix} \quad \begin{bmatrix} \text{O} \\ \text{W} \end{bmatrix} = \begin{bmatrix} 15 \\ 23 \end{bmatrix}$$

So at this stage, our message expressed as 2×1 matrices is as follows.

$$\begin{bmatrix} 1 \\ 20 \end{bmatrix} \begin{bmatrix} 20 \\ 1 \end{bmatrix} \begin{bmatrix} 3 \\ 11 \end{bmatrix} \begin{bmatrix} 27 \\ 14 \end{bmatrix} \begin{bmatrix} 15 \\ 23 \end{bmatrix} \quad \textbf{(I)}$$

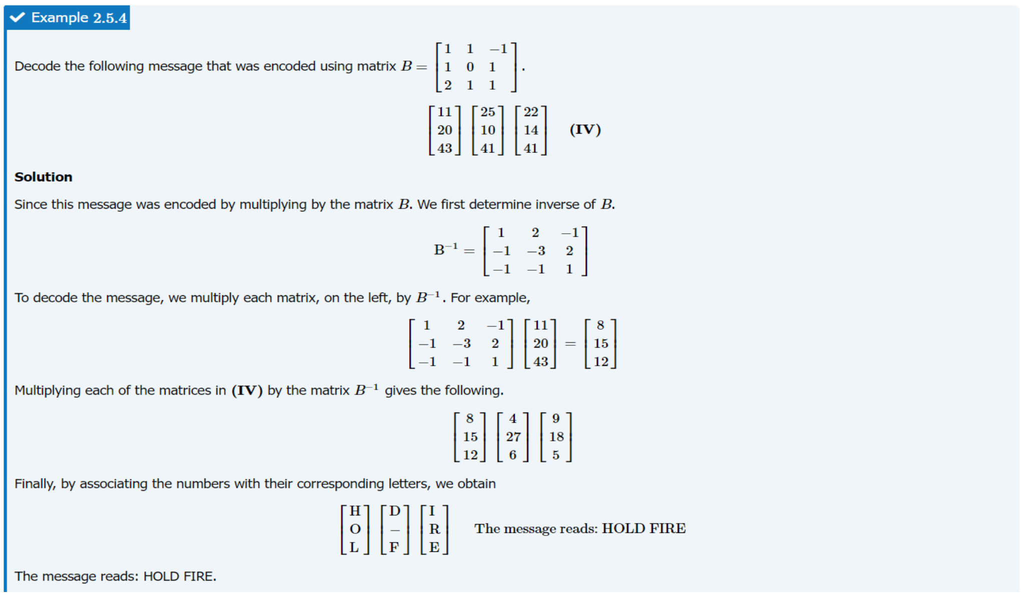
Now to encode, we multiply, on the left, each matrix of our message by the matrix A. For example, the product of A with our first matrix is: $\begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 20 \end{bmatrix} = \begin{bmatrix} 41 \\ 61 \end{bmatrix}$

And the product of A with our second matrix is: $\begin{bmatrix} 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 20 \\ 1 \end{bmatrix} = \begin{bmatrix} 22 \\ 23 \end{bmatrix}$

Multiplying each matrix in **(I)** by matrix A, in turn, gives the desired coded message:

$$\begin{bmatrix} 41 \\ 61 \end{bmatrix} \begin{bmatrix} 22 \\ 23 \end{bmatrix} \begin{bmatrix} 25 \\ 36 \end{bmatrix} \begin{bmatrix} 55 \\ 69 \end{bmatrix} \begin{bmatrix} 61 \\ 84 \end{bmatrix}$$

Figure 2: Encoding a message using a matrix given



✓ Example 2.5.4

Decode the following message that was encoded using matrix $B = \begin{bmatrix} 1 & 1 & -1 \\ 1 & 0 & 1 \\ 2 & 1 & 1 \end{bmatrix}$.

$$\begin{bmatrix} 11 \\ 20 \\ 43 \end{bmatrix} \begin{bmatrix} 25 \\ 10 \\ 41 \end{bmatrix} \begin{bmatrix} 22 \\ 14 \\ 41 \end{bmatrix} \quad \textbf{(IV)}$$

Solution

Since this message was encoded by multiplying by the matrix B. We first determine inverse of B.

$$\text{B}^{-1} = \begin{bmatrix} 1 & 2 & -1 \\ -1 & -3 & 2 \\ -1 & -1 & 1 \end{bmatrix}$$

To decode the message, we multiply each matrix, on the left, by B^{-1}. For example,

$$\begin{bmatrix} 1 & 2 & -1 \\ -1 & -3 & 2 \\ -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 11 \\ 20 \\ 43 \end{bmatrix} = \begin{bmatrix} 8 \\ 15 \\ 12 \end{bmatrix}$$

Multiplying each of the matrices in **(IV)** by the matrix B^{-1} gives the following.

$$\begin{bmatrix} 8 \\ 15 \\ 12 \end{bmatrix} \begin{bmatrix} 4 \\ 27 \\ 6 \end{bmatrix} \begin{bmatrix} 9 \\ 18 \\ 5 \end{bmatrix}$$

Finally, by associating the numbers with their corresponding letters, we obtain

$$\begin{bmatrix} \text{H} \\ \text{O} \\ \text{L} \end{bmatrix} \begin{bmatrix} \text{D} \\ - \\ \text{F} \end{bmatrix} \begin{bmatrix} \text{I} \\ \text{R} \\ \text{E} \end{bmatrix} \quad \text{The message reads: HOLD FIRE}$$

The message reads: HOLD FIRE.

Figure 3: Decoding a message using a matrix given

3.3 Method for problem 1:

To decode the message, we take the string of coded numbers and multiply it by the inverse of the matrix to get the original string of numbers. Finally, by associating the numbers with their corresponding letters, we obtain the original message.

1. Take the string of coded numbers and multiply it by the inverse of the matrix that was used to encode the message.

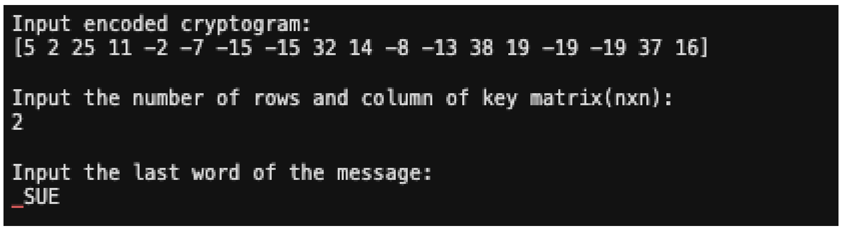
2. Associate the numbers with their corresponding letters.

3.3 Code Matlab for the problem 1

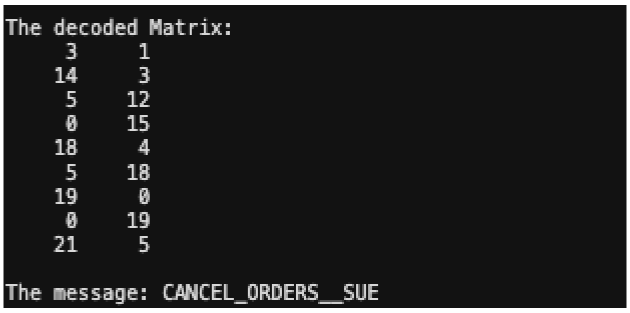
```
clc
close all

%input the encoded crytogram

encodedCrypto = input('Input encoded cryptogram: ');

%length of the message

noW = length(encodedCrypto);
R = input('Input the number of rows and column of key
matrix(nxn): ');

%Transform the encoded crytogram to the matrix

a_vector = reshape(encodedCrypto,R,[]);
b_vector = a_vector';

%enter the last word of the message and transform it to matrix

msg_lastwordFirst = input('Input the last word of the message:
','s');
msg_lastwordFinal = strrep(msg_lastwordFirst,'_','@');
c_vector = double(msg_lastwordFinal)-64;
d_vector = reshape(c_vector,R,[]);
e_vector = d_vector';

%length of the last word

noW2 = length(msg_lastwordFinal);

%Access to the encoded matrix of the last word

i = noW/R;
j = noW2/R;
A = b_vector(i - j + 1 : end, 1 : end);
B = inv(A);
C = B; %efficiency

%Find the inverse of the key matrix;

inversekeyMatrix = C * e_vector;
```

```
%Decode the message

decodeMatrix = b_vector * inversekeyMatrix;
f_vector = mod(decodeMatrix,27) + 64;
D2 = reshape(f_vector',1,noW);
D5 = round(D2);
D3 = char(D5);
D4 = strrep(D3,'@','_');

%display the message
fprintf('The decoded message: ');
disp(D4);
```

3.4 Result



```
Input encoded cryptogram:
[5 2 25 11 -2 -7 -15 -15 32 14 -8 -13 38 19 -19 -19 37 16]

Input the number of rows and column of key matrix(nxn):
2

Input the last word of the message:
_SUE
```



```
The decoded Matrix:
     3     1
    14     3
     5    12
     0    15
    18     4
     5    18
    19     0
     0    19
    21     5

The message: CANCEL_ORDERS__SUE
```

3.5 Evaluate, analysis and conclusion

PROBLEM 2

4.1 Introduction

In mathematics, particularly linear algebra and numerical analysis, the Gram–Schmidt process is a method for orthonormalizing a set of vectors in an inner product space, most commonly the Euclidean space R^n equipped with the standard inner product. The Gram–Schmidt process takes a finite, linearly independent set of vectors S = {v_1, ..., v_k} for k ≤ n and generates an orthogonal set S′ = {u_1, ..., u_k} that spans the same k-dimensional subspace of R^n as S.

The method is named after Jørgen Pedersen Gram and Erhard Schmidt, but Pierre-Simon Laplace had been familiar with it before Gram and Schmidt. In the theory of Lie group decompositions, it is generalized by the Iwasawa decomposition.

The application of the Gram–Schmidt process to the column vectors of a full column rank matrix yields the QR decomposition (it is decomposed into an orthogonal and a triangular matrix).

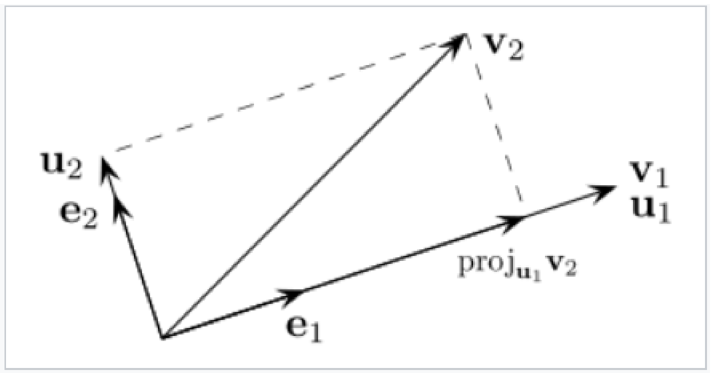


Figure 4: *The first 2 steps of the Gram-Schimdt process*

4.2 Theoretical basis

The Gram-Schmidt process

We define the projection operator by

$$proj_u = \frac{(v,u)}{(u,u)}u$$

where <v,u> denotes the inner product of the vectors v and u. This operator projects the vector v orthogonally onto the line spanned by vector u. If u = 0, we define $\mathrm{proj}_0(v):=0$, i.e., the projection map proj_0 is the zero map, sending every vector to the zero vector.

The Gram–Schmidt process then works as follows:

$$\begin{aligned}
\mathbf{u}_1 &= \mathbf{v}_1, & \mathbf{e}_1 &= \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|}\\
\mathbf{u}_2 &= \mathbf{v}_2 - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_2), & \mathbf{e}_2 &= \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|}\\
\mathbf{u}_3 &= \mathbf{v}_3 - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_3) - \mathrm{proj}_{\mathbf{u}_2}(\mathbf{v}_3), & \mathbf{e}_3 &= \frac{\mathbf{u}_3}{\|\mathbf{u}_3\|}\\
\mathbf{u}_4 &= \mathbf{v}_4 - \mathrm{proj}_{\mathbf{u}_1}(\mathbf{v}_4) - \mathrm{proj}_{\mathbf{u}_2}(\mathbf{v}_4) - \mathrm{proj}_{\mathbf{u}_3}(\mathbf{v}_4), & \mathbf{e}_4 &= \frac{\mathbf{u}_4}{\|\mathbf{u}_4\|}\\
&\vdots & &\vdots\\
\mathbf{u}_k &= \mathbf{v}_k - \sum_{j=1}^{k-1}\mathrm{proj}_{\mathbf{u}_j}(\mathbf{v}_k), & \mathbf{e}_k &= \frac{\mathbf{u}_k}{\|\mathbf{u}_k\|}.
\end{aligned}$$

The sequence $u_1, ..., u_k$ is the required system of orthogonal vectors, and the normalized vectors $e_1, ..., e_k$ form an orthonormal set. The calculation of the sequence $u_1, ..., u_k$ is known as Gram–Schmidt orthogonalization, while the calculation of the sequence $e_1, ..., e_k$ is known as Gram–Schmidt orthonormalization as the vectors are normalized.

To check that these formulas yield an orthogonal sequence, first compute $<u_1, u_2>$ by substituting the above formula for u_2: we get zero. Then use this to compute $<u_1, u_3>$ again by substituting the formula for u_3: we get zero. The general proof proceeds by mathematical induction.

Geometrically, this method proceeds as follows: to compute u_i, it projects v_i orthogonally onto the subspace U generated by $u_1, ..., u_{i-1}$, which is the same as the subspace generated by $v_1, ..., v_{i-1}$. The vector ui is then defined to be the difference between v_i and this projection, guaranteed to be orthogonal to all of the vectors in the subspace U.

The Gram–Schmidt process also applies to a linearly independent countably infinite sequence $\{v_i\}_i$. The result is an orthogonal (or orthonormal) sequence $\{u_i\}_l$ such that for natural number n: the algebraic span of $v_1, ..., v_n$ is the same as that of $u_1, ..., u_n$.

If the Gram–Schmidt process is applied to a linearly dependent sequence, it outputs the 0 vector on the ith step, assuming that v_i is a linear combination of $v_1, ..., v_{i-1}$. If an orthonormal basis is to be produced, then the algorithm should test for zero vectors in the output and discard them because no multiple of a zero vector can have a length of 1. The number of vectors output by the algorithm will then be the dimension of the space spanned by the original inputs.

4.3 Method for problem 2

In any inner product space, we can choose the basis in which to work.

It often greatly simplifies calculations to work in an orthogonal basis. For one thing, if $S=\{v_1,v_2,\dots,v_n\}$ is an orthogonal basis for an inner product space V, which means all pairs of distinct vectors in S are orthogonal:

$$< v_i , v_j > = 0 \text{ for all } v_i , v_j \in S.$$

then it is a simple matter to express any vector $w \in V$ as a linear combination of the vectors in S:

W = $\frac{<w,v1>}{\|v1\|^2}v1+\frac{<w,v2>}{\|v2\|^2}v2+...+\frac{<w,vn>}{\|vn\|^2}vn$.

Given an arbitrary basis $\{u_1,u_2,\dots,u_n\}$for an n-dimensional inner product space V, the **Gram-Schmidt algorithm** constructs an orthogonal basis $\{v_1,v_2,\dots,v_n\}$for V:

That is, w has coordinates

$$\begin{bmatrix} \frac{\langle \mathbf{w},\mathbf{v}_1\rangle}{\|\mathbf{v}_1\|^2}\mathbf{v}_1 \\ \frac{\langle \mathbf{w},\mathbf{v}_2\rangle}{\|\mathbf{v}_2\|^2}\mathbf{v}_2 \\ \vdots \\ \frac{\langle \mathbf{w},\mathbf{v}_n\rangle}{\|\mathbf{v}_n\|^2}\mathbf{v}_n \end{bmatrix}$$

relative to the basis S.

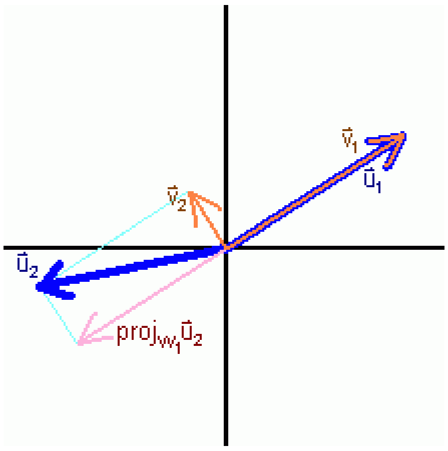
Step 1 Let $v_1=u_1$.

Step 2 Let v_2 = u_2 - $\text{proj}_{W1}u_2$ = $u2-\frac{<u2,v1>}{\|v1\|^2}v1$ where W_1 is the space spanned by v_1, and $\text{proj}_{W1}u_2$ is the orthogonal projection of u2 on W1.

Step 3 Let v_3 = u_3 - $\text{proj}_{W1}u_3$ = $u3-\frac{<u3,v1>}{\|v1\|^2}v1-\frac{<u3,v2>}{\|v2\|^2}v2$ where W2 is the space spanned by v1 and v2.

Step 4 Let v_4 = u_4 - $\text{proj}_{W1}u_4$ = $u4-\frac{<u4,v1>}{\|v1\|^2}v1-\frac{<u4,v2>}{\|v2\|^2}v2-\frac{<u4,v3>}{\|v3\|^2}v3$ where W3 is the space spanned by v1,v2, and v3.

Continue this process up to v_n. The resulting orthogonal set {v1,v2,…,vn} consists of n linearly independent vectors in V and so forms an orthogonal basis for V.



To obtain an orthonormal basis, which is an orthogonal set in which each vector has norm 1, for an inner product space V, use the Gram-Schmidt algorithm to construct an orthogonal basis. Then simply normalize each vector in the basis.

For R^n with the Eudlidean inner product dotproduct, we of course already know of the orthonormal
basis {(1,0,0,…,0),(0,1,0,…,0),…,(0,…,0,1)}{(1,0,0,…,0),(0,1,0,…,0),…,(0,…,0,1)}. For more abstract spaces, however, the existence of an orthonormal basis is not obvious. The Gram-Schmidt algorithm is powerful in that it not only guarantees the existence of an orthonormal basis for any inner product space, but actually gives the construction of such a basis.

4.3 Code Matlab for problem 2

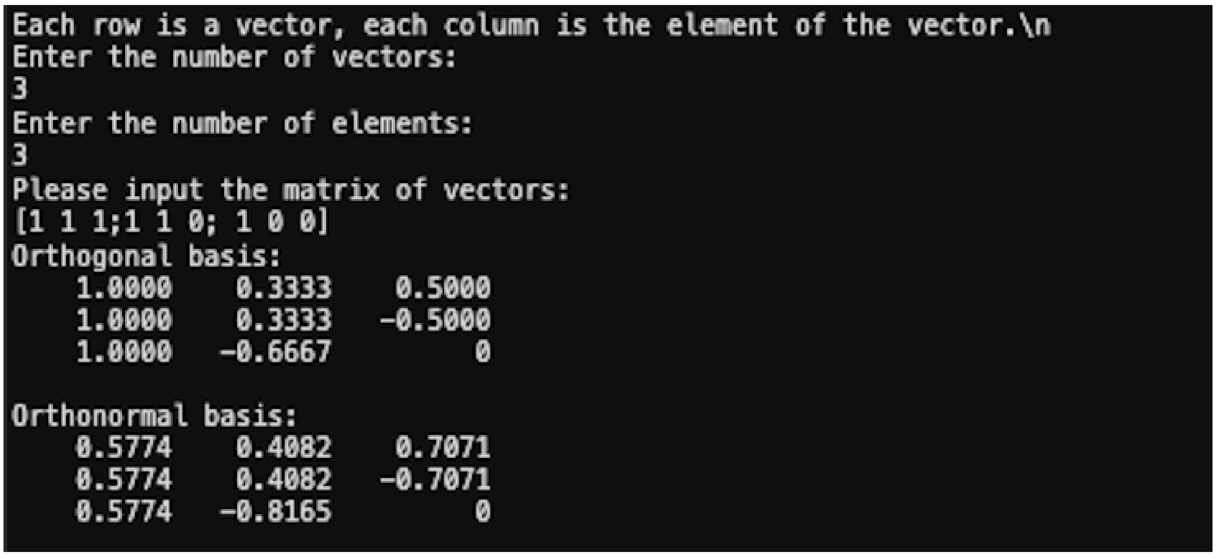
```
clc
close all

disp('Each row is a vector, each column is the element of the
vector.\n')
x = input('Enter the number of vectors: ');
y = input('Enter the number of elements: ');

% Assign the vectors by rows to the matrix

M = input('Please input the matrix of vectors: ');

% Check wheter M is LINEARLY INDEPENDENT or NOT

if rank(M) < x
   error('Your span must be LINEARLY INDEPENDENT');
end
V = M';
```

```
matrix_size = size(V);

% m is the number of rows

m = matrix_size(1,1);

% n is the number of columns

n = matrix_size(1,2);

%initialize vector u

u = zeros(m,n);
u(:,1) = V(:,1);

%Evaluate the orthogonal basis using Gram-Schmidt process

for k = 2:n
   S = V(:,k);
   for j = 1:(k-1)
       S = S - (dot(V(:,k),u(:,j))/dot(u(:,j),u(:,j)))*u(:,j);
   end
   u(:,k) = S;
end

%Evaluate the orthonormal basis from orthogonal basis

t = zeros(m,n);

for j = 1:n
   normalized_factor = norm(u(:,j));
   for i = 1:m
       s = u(i,j)/normalized_factor;
       t(i,j) = s;
   end
end

%display Orthogonal basis

disp('Orthogonal basis: ')
disp(u);

%display Orthonormal basis

disp('Orthonormal basis: ')
disp(t);

%find the coordinate and the length of the input vector

fprintf('Please input any vector in R%d\n',y);
```

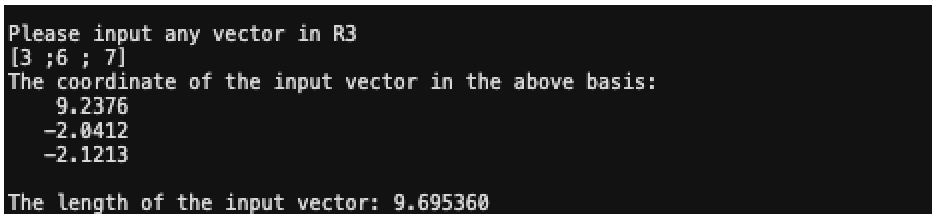
```
vector_ANY = input('');

%find and display the coordinate

q = t' * vector_ANY;
fprintf('The coordinate of the input vector in the above
basis:\n');
disp(q);

%evaluate the length of the vector

n = numel(vector_ANY);
w = 0;
for i = 1:n
   w = w + q(i,1)*q(i,1);
end
fprintf('The length of the input vector: %f\n',sqrt(w))
```

4.4 Result

th1

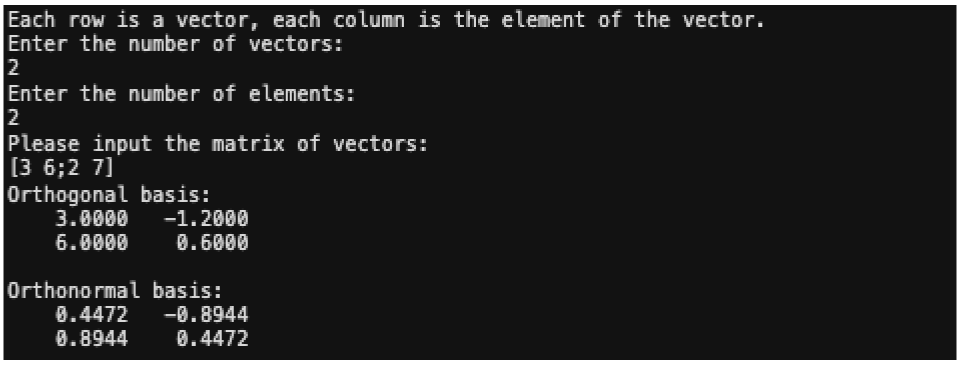


```
Each row is a vector, each column is the element of the vector.\n
Enter the number of vectors:
3
Enter the number of elements:
3
Please input the matrix of vectors:
[1 1 1;1 1 0; 1 0 0]
Orthogonal basis:
    1.0000    0.3333    0.5000
    1.0000    0.3333   -0.5000
    1.0000   -0.6667         0

Orthonormal basis:
    0.5774    0.4082    0.7071
    0.5774    0.4082   -0.7071
    0.5774   -0.8165         0
```

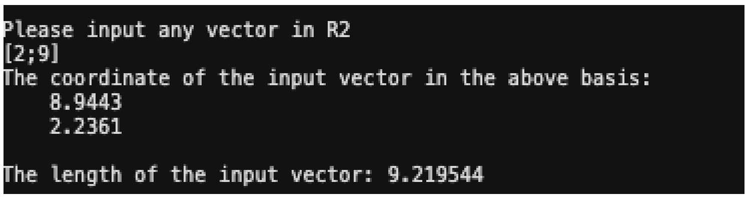


```
Please input any vector in R3
[3 ;6 ; 7]
The coordinate of the input vector in the above basis:
    9.2376
   -2.0412
   -2.1213

The length of the input vector: 9.695360
```

th2

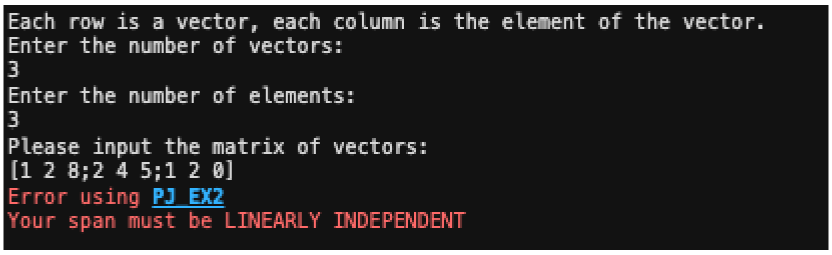


```
Each row is a vector, each column is the element of the vector.
Enter the number of vectors:
2
Enter the number of elements:
2
Please input the matrix of vectors:
[3 6;2 7]
Orthogonal basis:
    3.0000   -1.2000
    6.0000    0.6000

Orthonormal basis:
    0.4472   -0.8944
    0.8944    0.4472
```



```
Please input any vector in R2
[2;9]
The coordinate of the input vector in the above basis:
    8.9443
    2.2361

The length of the input vector: 9.219544
```

th3



```
Each row is a vector, each column is the element of the vector.
Enter the number of vectors:
3
Enter the number of elements:
3
Please input the matrix of vectors:
[1 2 8;2 4 5;1 2 0]
Error using PJ_EX2
Your span must be LINEARLY INDEPENDENT
```

4.5 Evaluate, analysis and conclusion

PROBLEM 3

5.1 Introduction

In mathematics, an inner product space (or, rarely, a Hausdorff pre-Hilbert space) is a real vector space or a complex vector space with an operation called an inner product. The inner product of two vectors in the space is a scalar, often denoted with angle brackets such as in <a , b>. Inner products allow formal definitions of intuitive geometric notions, such as lengths, angles, and orthogonality (zero inner product) of vectors. Inner product spaces generalize Euclidean vector spaces, in which the inner product is the dot product or scalar product of Cartesian coordinates. Inner product spaces of infinite dimension are widely used in functional analysis. Inner product spaces over the field of complex numbers are sometimes referred to as unitary spaces. The first usage of the concept of a vector space with an inner product is due to Giuseppe Peano, in 1898.

An inner product naturally induces an associated norm, (denoted |x| and |y| in the picture); so, every inner product space is a normed vector space. If this normed space is also complete (that is, a Banach space) then the inner product space is a Hilbert space. If an inner product space H is not a Hilbert space, it can be extended by completion to a Hilbert space H . This means that H is a linear subspace of H, the inner product of H is the restriction of that of H, and H is dense in H for the topology defined by the norm.

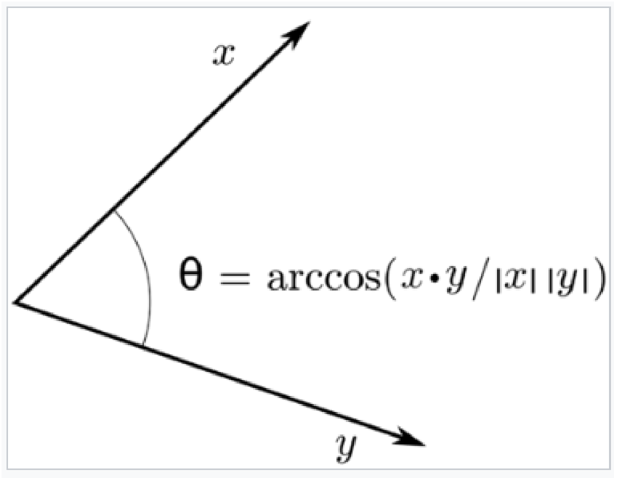


Figure 5: *Geometric interpretation of the angle between two vectors defined using an inner product*

5.2 Theoretical basis

Basic properties

- In the following properties, which result almost immediately from the definition of an inner product, x, y and z are arbitrary vectors, and a and b are arbitrary scalars.

- <0 , x> = <x , 0> = 0.
- <x , x> is real and non-negative.
- <x , x> = 0 if and only if x = 0.
- <x , ax + by> = a<x , y> +b<x , y>.
  This implies that an inner product is a sesquilinear form.
- <x + y , x + y> = <x , x> +2Re(<x , y>) + <y , y> , where Re denotes the real part of its argument.

Over R , conjugate-symmetry reduces to symmetry, and sesquilinearity reduces to bilinearity. Hence an inner product on a real vector space is a positive-definite symmetric bilinear form. The binomial expansion of a square becomes

<x + y , x + y> = <x , x> + 2<x , y> + <y , y>.

- In mathematics, an ellipse is a plane curve surrounding two focal points, such that for all points on the curve, the sum of the two distances to the focal points is a constant. It generalizes a circle, which is the special type of ellipse in which the two focal points are the same. The elongation of an ellipse is measured by its eccentricity e , a number ranging from e = 0 (the limiting case of a circle) to e =1 (the limiting case of infinite elongation, no longer an ellipse but a parabola).

An ellipse has a simple algebraic solution for its area, but only approximations for its perimeter (also known as circumference), for which integration is required to obtain an exact solution.

Analytically, the equation of a standard ellipse centered at the origin with width 2a and height 2b is:

$$\frac{x^2}{a^2}+\frac{y^2}{b^2}=1$$

- The reflection matrix of a point in Oxyz over (P) is :

$$Qr = I - 2\frac{uu^T}{u^Tu}$$

5.3 Method for problem 3

Step 1:

+Get a inner product from a graph given (which is a ellipse)

Step 2:

+Get the normal vector of the plane to implement the reflection matrix of a plane

$$Qr = I - 2\frac{uu^T}{u^Tu}$$

+ And then we should convert the plane into a new null equation.

Step 3:

+ Because we need to represent the reflection equation in the R^2 so we only need to do in X , Y and ignore the Z .

+ Use Qr multiply to x y from the ellipse to solve the equation X Y

Step 4:

+ Calculate D and analysis it.

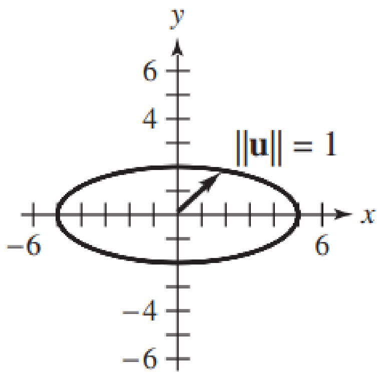
+ Case D = 0 we get a simple equation.

+ Case D different from 0 we get a polynomial equation.

Step 5:

+ Get the result and conclude what we get.

5.3 First step to get the inner product



As we can see that the graph above is an elip so we get the elip equation by selecting 2 points (0;2) and (5;0).

$$\frac{x^2}{25}+\frac{y^2}{4}=1$$

So the inner product can be rewrite as:

$$<x , y> = \frac{1}{25} x_1y_1 + \frac{1}{4} x_2y_2.$$

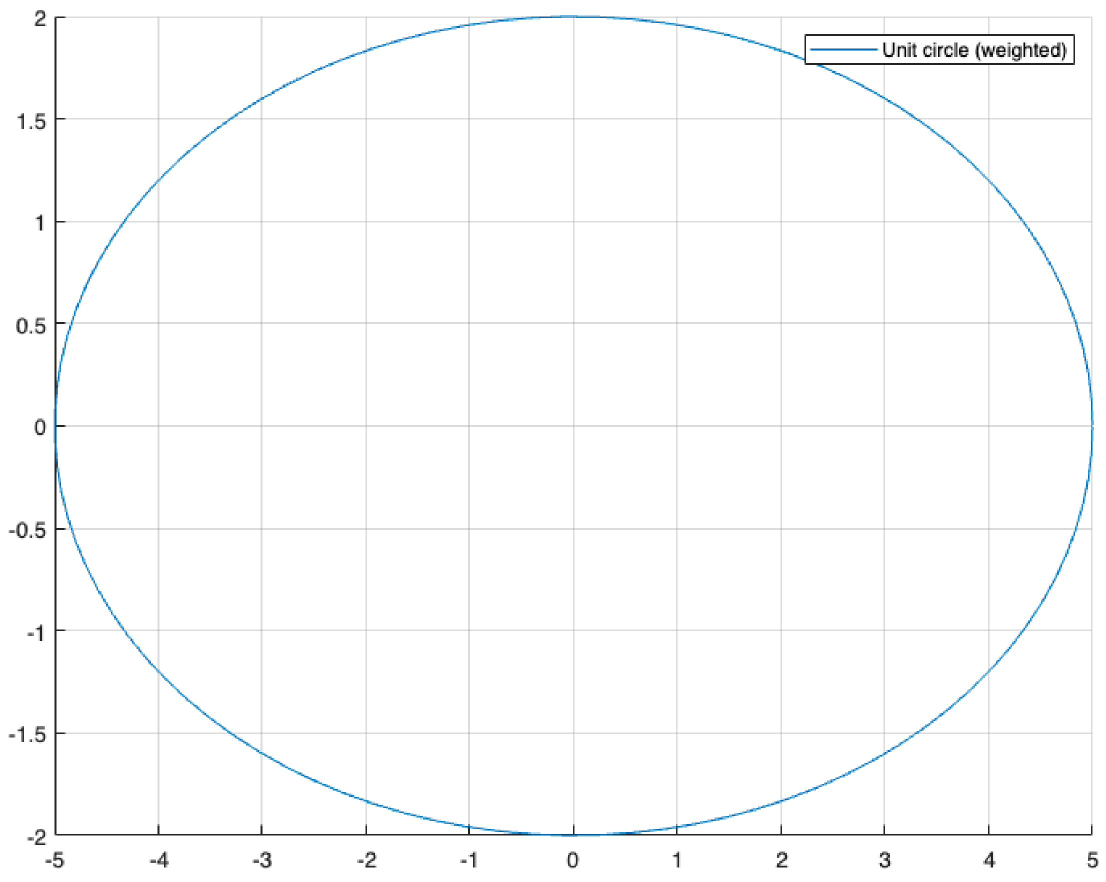
5.4 Code Matlab for problem 3

```
clc
close all
syms X Y

% plot the original unit vector

theta = linspace(0,2*pi,100);
```

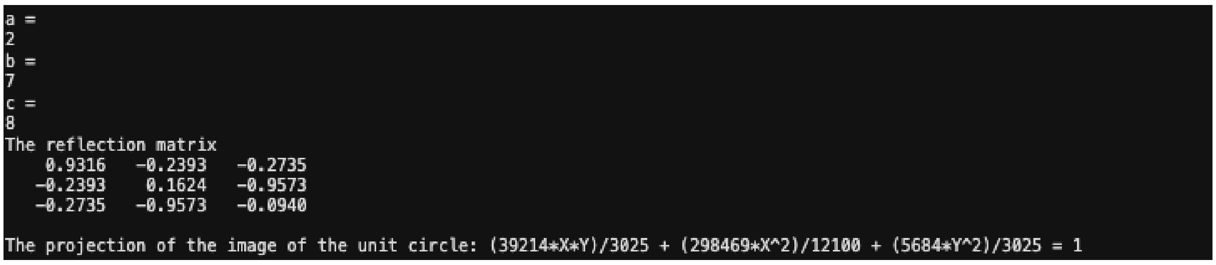
```
k = sqrt(25) * cos(theta);
q = sqrt(4) * sin(theta);
figure
hold on
plot (k,q,'DisplayName','Unit circle (weighted)');
legend
grid
hold off

% input the coefficient of the plane (P): ax + by + cz = 0

a = input('a = ');
b = input('b = ');
c = input('c = ');

% create the matrix of the orthogonal vector of (P)

uT = [a b c];
u = uT';

% the identity matrix in R3

I = eye(3);

% the reflection matrix of a point in Oxyz over (P)

Qr = I - 2*(u*uT)/(uT*u);
disp(Qr);

% evaluate the projection of the image of the unit circle
D = Qr(1,1) * Qr(2,2) - Qr(2,1) * Qr(1,2);
Dx = Qr(2,2)*X-Qr(1,2)*Y;
Dy = Qr(1,1)*X-Qr(2,1)*Y;
if D == 0
   f = a*X+b*Y;
   %display f
   figure
   fimplicit(X*a + Y*b,[-5 5],'DisplayName','Projection')
   axis('equal')
   grid
   legend
   fprintf('The projection of the image of the unit circle: %s = 0',f)
else
   f = (Dx/D)^2/25+(Dy/D)^2/4;
   f2 = expand(f);
   %display f2
   fimplicit(f2 - 1,[-5 5],'DisplayName','Projection');
   axis('equal')
   grid
   legend
   fprintf('The projection of the image of the unit circle: %s =
1',string(f2))
end
```

5.5 Result

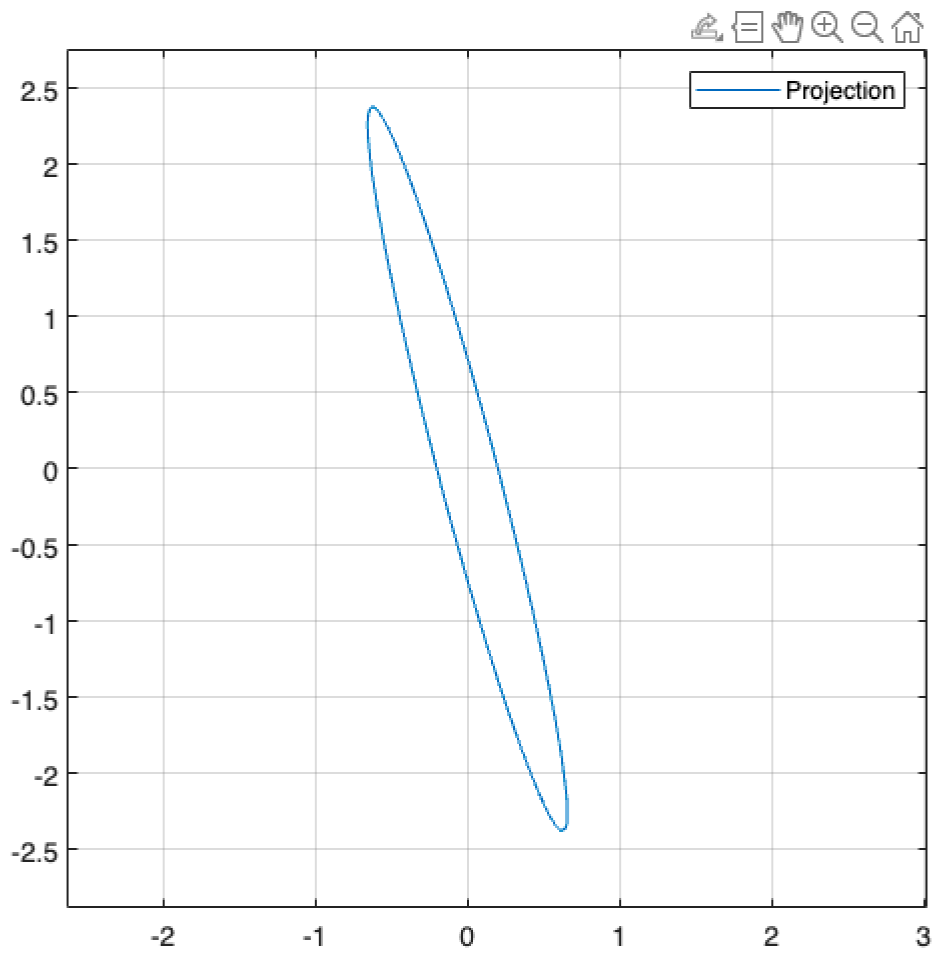
unit cirlce



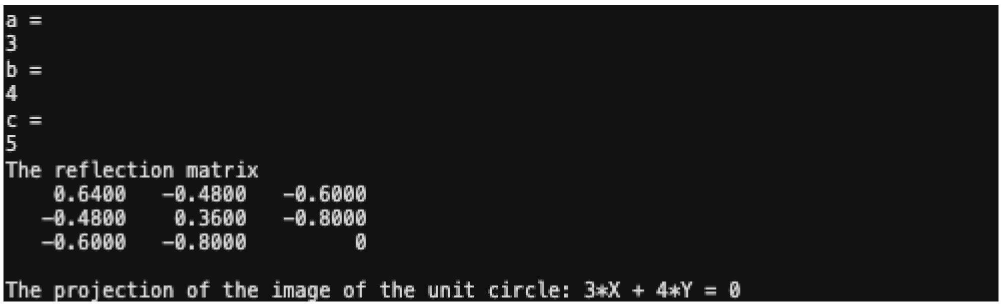
th1



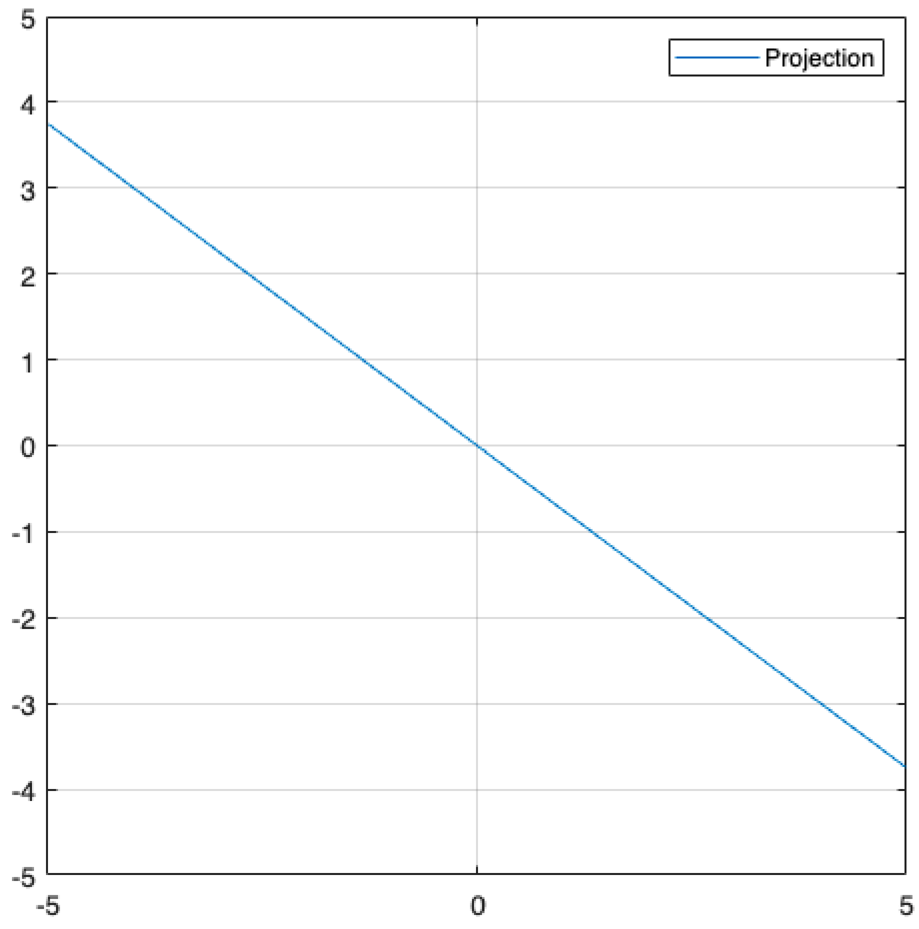
```
a =
2
b =
7
c =
8
The reflection matrix
    0.9316   -0.2393   -0.2735
   -0.2393    0.1624   -0.9573
   -0.2735   -0.9573   -0.0940

The projection of the image of the unit circle: (39214*X*Y)/3025 + (298469*X^2)/12100 + (5684*Y^2)/3025 = 1
```



th2



```
a =
3
b =
4
c =
5
The reflection matrix
    0.6400   -0.4800   -0.6000
   -0.4800    0.3600   -0.8000
   -0.6000   -0.8000         0

The projection of the image of the unit circle: 3*X + 4*Y = 0
```



5.6 Evaluate, analysis and conclusion

Conclusion

Thus, we have gone from general problems to quite complex specific problems that require lots of computational work with problem solvers. However, with the support of the Matlab tool, solving and surveying problems becomes easy, lively and direct. We can easily use Matlab to simulate or calculate the distribution of an oxygen molecule and also the probability that a molecule has a velocity in an interval velocity given.

Advantages:

- Calculation is easy and convenient, giving accurate results like the common calculation method.

- Help to understand more about Matlab applications in technical problems.

- Save operation and calculation time compared to common calculation methods.

- Use content announcement commands that make the usage structure relatively simple, easy to understand, easy to use and suitable for everyone.

Weakness:

- Designing code takes a lot of time and effort.

- The code is cumbersome.

- Also simulated within the specified topic, not yet created to other topics other technical calculations.

REFERENCES:

[1] Application of Matrices in Cryptography

https://math.libretexts.org/Bookshelves/Applied_Mathematics/Applied_Finite_Mathematics_(Sekhon_and_Bloom)

[2] GRAM-SCHMIDT METHOD

https://math.hmc.edu/calculus/hmc-mathematics-calculus-online-tutorials/linear-algebra/gram-schmidt-method/

[3] Gram-Schmidt process- Wikipedia

https://en.wikipedia.org/wiki/Gram%E2%80%93Schmidt_process

[4]	Inner product - Wikipedia

https://en.wikipedia.org/wiki/Inner_product_space

[5] Ellipse -Wikipedia

https://en.wikipedia.org/wiki/Ellipse