Lab1 Report

I.

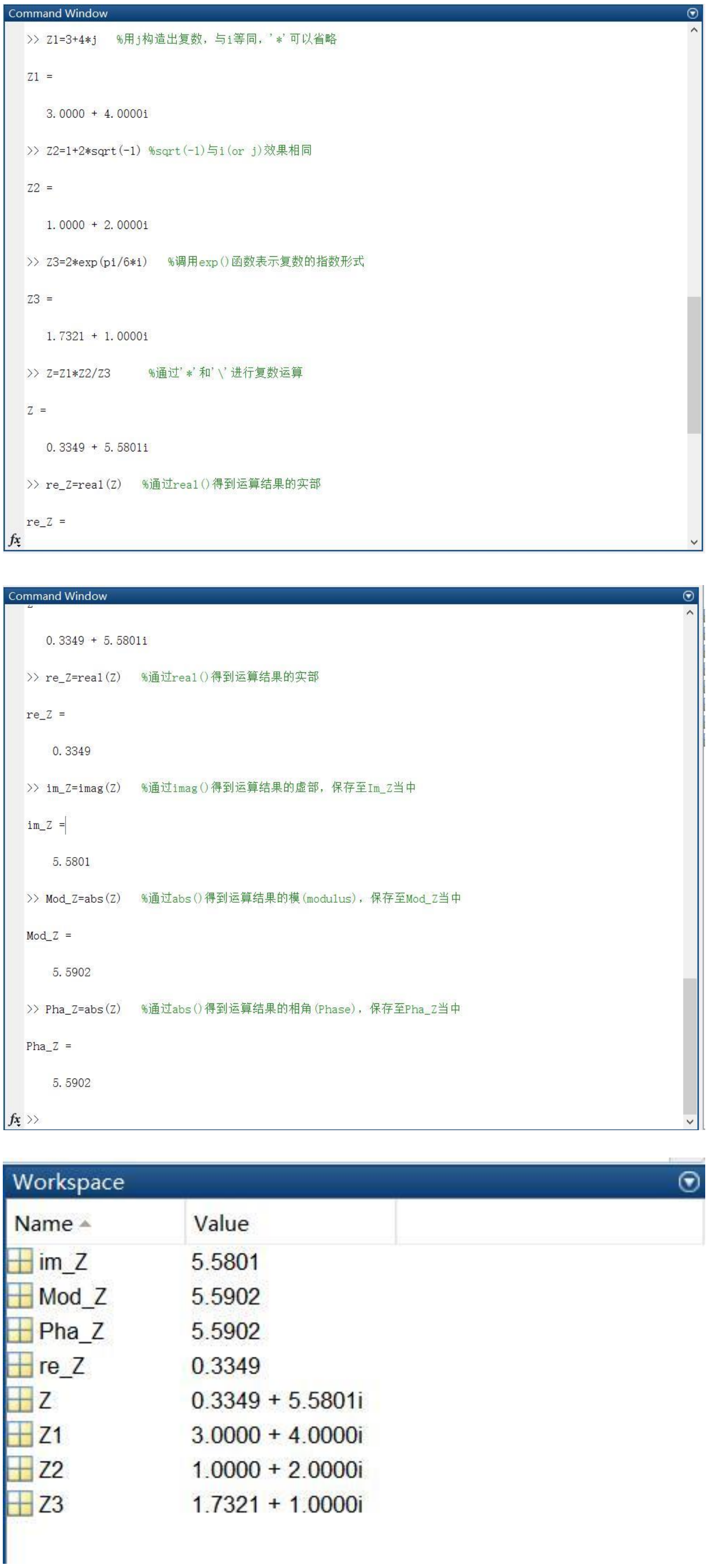

Fig.1 Calculation in Command& data in Workspace

From above, we find this calculation is about complex. Three kinds of way can be used to express the imagery part of complex: ‘I’, ‘j’ or “sqrt (-1)”.So I use ‘j’ in Z1, “sqrt(-1)” in Z2 and ‘i’ in Z3.We call the function “exp(x)” to get Z3.

Then the operator ‘*’ and ‘\’ are available to the complex which makes ours easily achieve the goal. From the top picture of Fig.1, we can see the procedure of the calculation and the resulted complex Z.

In the second picture of Fig.1, I find four functions:” real ()”,” imag ()”,” abs ()” and” angle ()”. They are practice function and respectively used to get the real part, the imaginary part, the modulus and the phase of the Z.

Lastly, I captured a screenshot about Workspace in the Last picture of Fig.1.

II.

Code:

Tast2.m

```
clear;
z1=3+4j;
z2=1+sqrt(-1);          %sqrt(-1) is euqal to i(or j) in matalb
z3=2*exp(pi/6*1i);      %call exp()to express the Exponential form ,and '1i'is better in Matlab
%goal 1
Col_Vec=[z1;z2];        %Column Vector (2*1)
Row_Vec=[z2,z3];        %Row Vector   equals to [z2 z3]   (1*2)
Mat=Col_Vec*Row_Vec; %Matrix mutiply (2*2)
%goal 2
Am_Mat=abs(Mat);        %Am_Mat save the amplitued of the matrix entry
Ph_Mat=angle(Mat);      %Ph_Mat save the phase of the matrix entry
%goal t3
Mat(1,2)=1;             %set first row and second column to 1
```

Comment:

1) This Task is the update of the Task1.This Task focuses on honing our ability to use vectors and matrix. If you want to put two (or more) entry in a row, you can use the mark ’,’ or just the space; Otherwise, we need semicolon(i.e. ’;’) to separate two entry into a different row. Using the method above, I solve the goal1.

2) Goal 2, I called the function used in I (i.e. Task1) again. Thanks to MATLAB, these functions apply to not only num, but matrix.

3) We use Mat (r, c) to index the entry on the r-th and c-th column of Mat.

III

Code:

Tast3.m

```
%generate signals
x=linspace(0,2*pi,2*12); %using the function linspace
y1=sin(x);
y2=cos(x);
```

```
y3=log(x);
%plot    figure
figure(1)                         %create new figure
subplot(2,2,1)                    %create sub figure(2*2) 1st
plot(x,y1);
title('Curve1')
xlabel('x')
ylabel('sin(x)')
legend( 'sin(x)');

subplot(2,2,2)          %create sub figure(2*2) 2nd
plot(x,y2);
title('Curve2')
xlabel('x')
ylabel('cos(x)')
legend( 'cos(x)');

subplot(2,2,3)    %create sub figure(2*2) 3rd
plot(x,y3);
title('Curve3')
xlabel('x')
ylabel('log(x)')
legend( 'log(x)');

subplot(2,2,4)           %create sub figure(2*2) 4th
plot(x,y1,'-or',x,y2,'-.*g',x,y3,':vb');
title('Sum Curve')
xlabel('x')
ylabel('sin(x)')
legend( 'sin(x)','cos(x)','log(x)');
```

Output: Fig.2

Discussion:

In the task, I learn the skill to plot curves. I choose the Interval range (0,2*pi) for this is a whole period. In this cycle, we can find some properties about three curves in a subplot.

III

Code:

Fabonacci_for.m

```
function fab_n=Fabonacci_for(n)
% Invaild n
    if(n<1)                    % n must bigger than 1
        fab_n=NaN;
        return;
    end
```

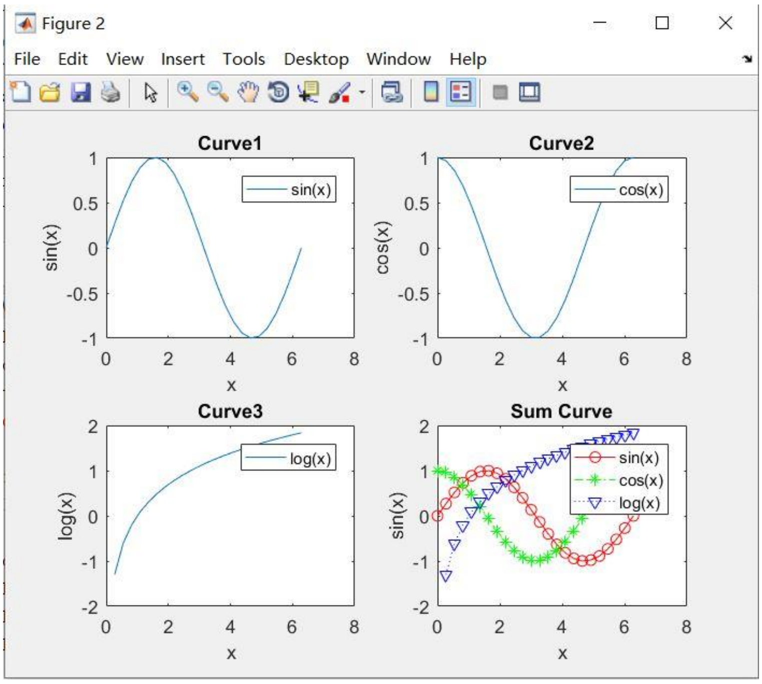

Fig.2 Sub Curves and their sum in a fig

```
% caluate n      for-method
    if(n==1||n==2)
         fab_n=ones(1,n);
    else
         fab_n(1)=1;
         fab_n(2)=1;
        for i=3:1:n
             fab_n(i)=fab_n(i-1)+fab_n(i-2);
        end
    end
  %plot the figure
   figure(1);
   stem(1:n,fab_n);
   title('Fabonacci Squence')
   xlabel('n')
   ylabel('Fabonacci(n)')
   legend('n-Fabonacci(n)')
end
```

Fabonacci_while.m

```
function fab_n=Fabonacci_while(n)
% Invaild n
    if(n<1)                  % n must bigger than 1
         fab_n=NaN;
         return;
```

```
    end

fab_n=zeros(1,n);
% caluate n      while-method
   N=n;
   while(n~=0)
         fab_n(n)=Fabonacci(n);
         n=n-1;
   end
%plot the figure
   figure(1);
   stem(1:N,fab_n);
   title('Fabonacci Squence')
   xlabel('n')
   ylabel('Fabonacci(n)')
   legend('n-Fabonacci(n)')
end
```

Fabonacci.m

```
function fab=Fabonacci(n)
% Invaild n
    if(n<1)                    % n must bigger than 1
          fab=NaN;
          return;
    end
%
   if(n==1||n==2)
         fab=1;
   else
         fab=Fabonacci(n-1)+Fabonacci(n-2);
   end
end
```

Output:

```
Command Window
>> Task3
>> Fabonacci_for(10)

ans =

     1     1     2     3     5     8    13    21    34    55

>> Fabonacci_while(10)

ans =

     1     1     2     3     5     8    13    21    34    55
```

Fig3. Command window with two functions

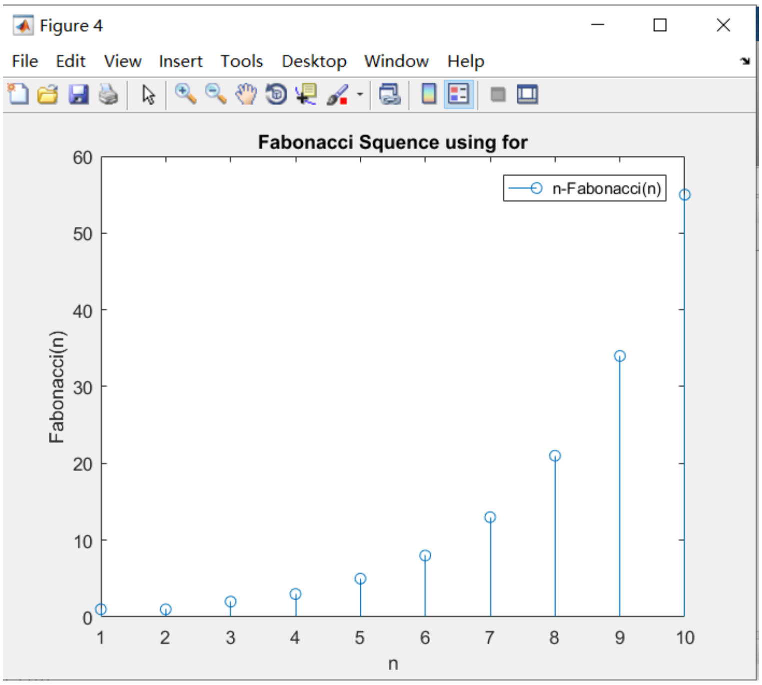

Fig.4 Fabnocci for figure

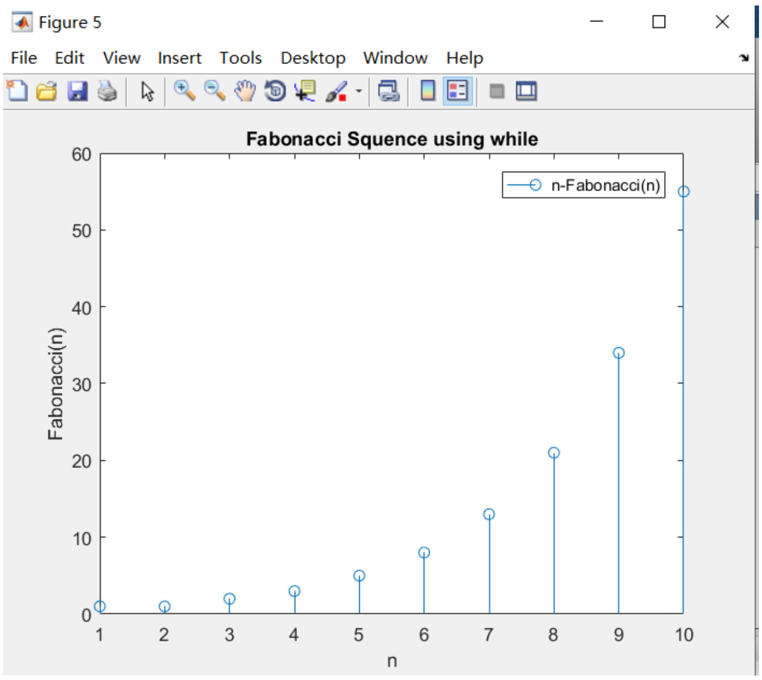

Fig.5 Fabnocci while figure

Discussion:

In this task, I use two statements “for” and “while”. In the code using for(i.e. Fabonacci_for.m), I generate each entry by calculating forward, with the variable i adding concurrently. Otherwise，in statements while(i.e. Fabonacci_while.m), the variable n is decreasing. So I generate a new function which uses the recursion algorithm by calling itself (i.e. function Fabonacci(n)). to calculate each entry .Run two function ,then check the figure 3,4,5,we can get the same output.