INSTITUTE OF ENGINEERING
CENTRAL CAMPUS,PULCHOWK

DIGITAL SIGNAL PROCESSING

LAB #1

Getting Started with MATLAB

Submitted BY:
AMRIT PRASAD PHUYAL
IOE Roll: PULL074BEX004

Submitted To:
Anila Kansakar
Department of Electronics and
Computer Engineering

January 1, 2022

Table of Contents

1 Title **1**

2 Objective **1**

3 Theory **1**
3.1 Variables . . . 1
3.2 Vectors and Matrices . . . 1
3.3 Arithmetic Operations . . . 1
3.4 Control Flow in Matlab . . . 2
3.5 Some useful Task , their Commands and Examples . . . 2
3.5.1 User-Defined Functions . . . 2
3.5.2 2D Plotting . . . 2
3.5.3 Polynomial Roots . . . 3
3.5.4 Dealing with Sound Files . . . 3
3.5.5 Complex Numbers . . . 3
3.5.6 Signal Processing and Image Processing . . . 3
3.5.7 Transfer Function Representation and Frequency Response . . . 3
3.6 Getting Help from Matlab . . . 4

4 LAB Problems **4**
4.1 Problem 1 . . . 4
4.2 Problem 2 . . . 4
4.3 Problem 3 . . . 5
4.4 Problem 4 . . . 7

5 Discussion and Conclusion **8**

List of MATLAB codes

1 MATLAB code for FOR loop and time measurement 2
2 Matlab function for polynomial calculaton 4
3 Matlab code for plotting the function y 4
4 Matlab code for calculation related to fibonacci numbers and plotting result 6
5 Matlab code for plotting the function f(x) 7

List of Figures

1 Plot for exponent cosine function 5
2 Plot for fibonacci numbers with sum of two consecutive numbers smaller than 10,000 7
3 Plot for given f(x) function where x is between 0 and 100 8

List of Tables

1 Mathematical Functions and corresponding Matlab syntax 1
2 Control Flow in Matlab 2

1 Title

Getting Started with MATLAB

2 Objective

Familiarization with MATLAB and its basic operations.

3 Theory

3.1 Variables

Unlike many programming languages, MATLAB does not require prior definition of the variables, instead the variables can be simply written as,

variable name = expression;

For example:

```
a = sin(64) + 2;
```

If the user doesn't specify the name of the variable, MATLAB automatically creates the variable **ans**.

```
> 3+2
ans=5
```

3.2 Vectors and Matrices

```
> x=[1:10]
> x=[1 3 7 15]
> y=[1:0.1:10]
> z=[1:3;4:6;7:9]
> [m,n]=size(z)
```

3.3 Arithmetic Operations

- Arithmetic operators: +,-,*,/,\,^
- Mathematical functions available: ABS, SQRT, LOG, SIN and COS.

	Mathematical Function	**Matlab Syntax for Function**
Arithemetic	$f_1 = a_1 + b_1x + c_1x^2$	f1 = a1 + b1*x + c1*x^2
and	$f_2 = a_2 + b_2x + c_2x^2 + d_2x^3$	f2 = a2 + b2*x + c2*x^2 + d2*x^3
Algebric Operation	$g = e^{At}(C_1\cos(Bt) + C_2\sin(Bt))$	g = exp(A*t)*(C1*cos(B*t)+C2*sin(B*t))
	$u = 2xy^2 + \sin(x+y)$	u = 2*x*y^2 + sin(x+y)

Table 1: Mathematical Functions and corresponding Matlab syntax

3.4 Control Flow in Matlab

Loops	FOR Loop	WHILE Loop	IF..ELSE..

Syntax	`for expression`
	`  statements`
	`end`
		`while expression`
		`  statements`
		`end`
			`if expression`
			`  statements`
			`elseif expression`
			`  statements`
			`else`
			`  statements`
			`end`

Table 2: Control Flow in Matlab

A FOR loop allows a statement to be repeated a fixed, predetermined number of times. Let's look at the following problem. We would like to fill the vector b with square roots of 1 to 1000. One way to do so, is by using a for loop.We will calculate the time required for this operation for comparing it with the more efficient version of this calculation. This code written in an m-file and save it under the name *tictoc.m*

```
clear ; %To clear all previous variables, and to free memory.
tic ; %This function initializes an internal clock
for i = 1:1000
    b(i) = sqrt(i);
end
t=toc;
str=sprintf('The time required was: %f',t);
disp(str)
```

Code 1: MATLAB code for FOR loop and time measurement

Above code produce following response in command window.

```
> tictoc
The time required was: 0.004101
```

3.5 Some useful Task , their Commands and Examples

3.5.1 User-Defined Functions

- **Commands:** `function [op1,op2,...]=cmd_name(ip1,ip2,...)`
- **Example:**

```
function y = fcn(x)
    y = sin(x.^2);  %Create in m file
end
```

3.5.2 2D Plotting

- **Commands:** `plot, subplot, figure, hold, stem, axis, title`
- **Example:**

```
> t=[-2:0.01:2];
> x=sin(t*10);
> plot(t,x)
```

```
> axis([-1 1 -1 1])
> zoom
> xlabel('Time')
> title('My first plot')
> specgram(x)
```

3.5.3 Polynomial Roots

- **Commands:** `roots(p)`
- **Example:**

```
> p = [1 2 1]; %polynomial x^2 +2x +1
> r = roots(p) %roots
r = -1 -1
```

3.5.4 Dealing with Sound Files

- **Commands:** `wavread, wavwrite, auread, auwrite, sound(y,fsamp)`
- **Example:**

```
> y=wavread('C:\sound.wav')  %file must be valid
> sound(y,44100);
```

3.5.5 Complex Numbers

- **Commands:** `j, real, imag, abs, angle`
- **Example:**

```
> real(j)   % locate a complex number in cartesian form
> imag(j)
> abs(j)     % locate a complex number in polar form;
> angle(j)
```

3.5.6 Signal Processing and Image Processing

- **Commands:** `fft(), dft(), con(), dither(), gray2ind(), ind2gray(), ind2rgb(), imread(), imwrite()`
- **Example:**

```
> A=imread('my_pic.jpg')%file must be valid
> whos
> imshow(A)
```

3.5.7 Transfer Function Representation and Frequency Response

- **Commands:** `tf2zp, zp2tf, freqs(), semilogx(), bode()`
- **Example:**

```
% Given H(s)=(2s+3)(s3+4s2+5)
> num=[2 3];
> den=[1 4 0 5];
> [z,p,k]=tf2zp(num,den);
> [num den]=zp2tf(z,p,k);
%one way of plotting
> T=0:0.1:1;
> y=step(num,den,t);
> plot(t,y)
%Another waty of plotting
> bode(num,den)
> [mag,phase]=bode(num,den,w);
> magdb=20*log10(mag);
> semilogx(w,magdb)
> semilogx(w,phase)
```

3.6 Getting Help from Matlab

```
> doc fft
> help help
> help cos
> help fft
> lookfor filter
```

4 LAB Problems

4.1 Problem 1

Calculate $\left(1+\frac{2}{n^2}\right)^n$ for n=3, 7

```
function [y] = p1(n)
    y = (1+2/n^2)^n;
end
```

Code 2: Matlab function for polynomial calculaton

Response of Command Window

```
>> p1(3)              >> p1(7)
ans = 1.8258          ans = 1.3232
```

4.2 Problem 2

Plot the function: $y = e^{-at}cos(\omega t)$, for $a = 2$, $\omega = 5$, and $t = 0 : 10$.

Codes:

```
t=linspace(0,10,1000);
a=2;
w=5;
y=ecos(a,w,t);
l=tiledlayout(1,1);
title(l,{'Plot of y for a=2, \omega=5 and t=0:10'})
nexttile
plot(t,y,'Linewidth',1.5)
xlabel('t','interpreter','latex')
ylabel ('$y=e^{-at}\cos(\omega t)$','interpreter','latex')
```

```
11
12  %ecos functon
13  function y = ecos(a,w,t)
14      y = exp(-a.*t).*cos(w.*t);
15  end
```

Code 3: Matlab code for plotting the function y

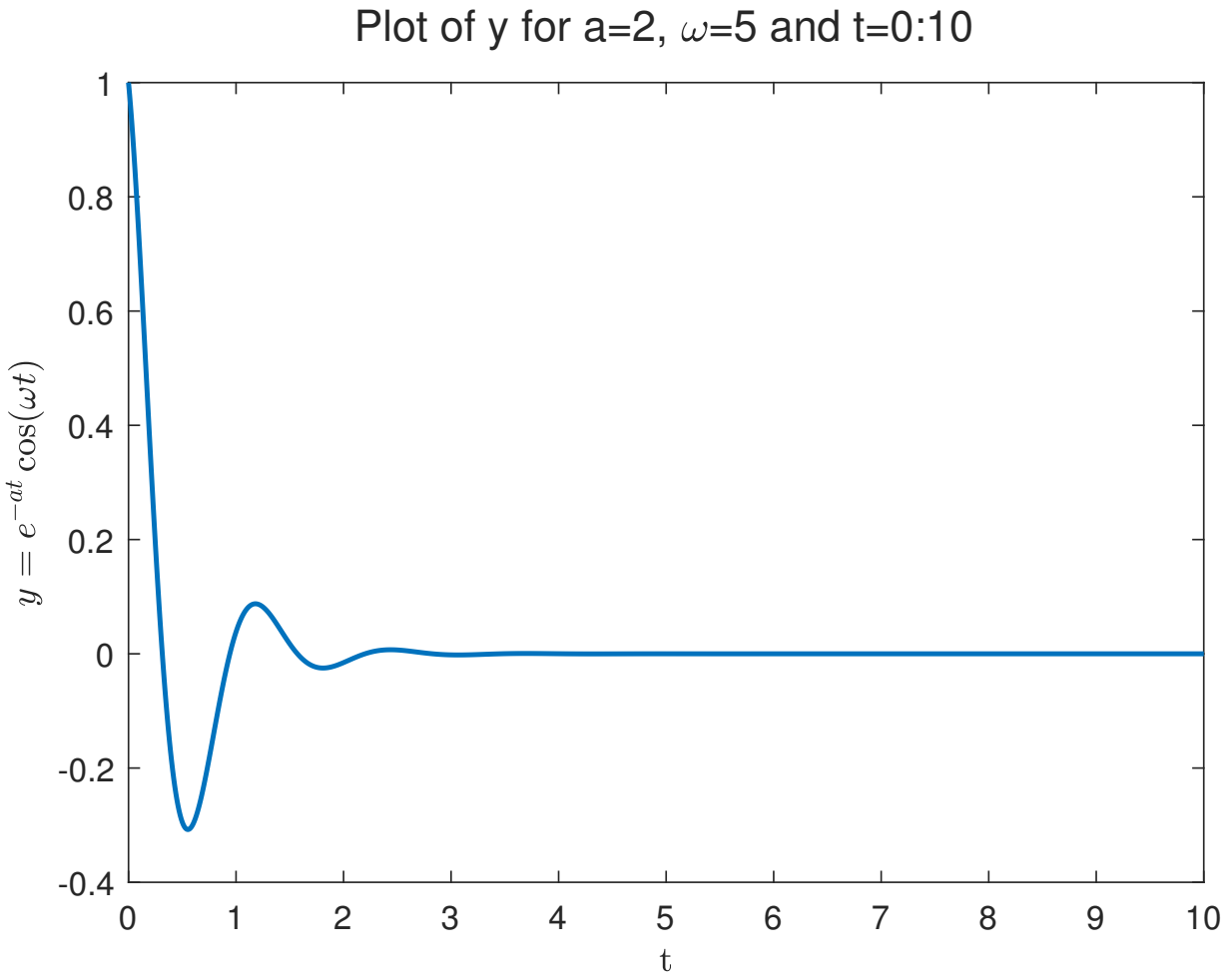

Figure 1: Plot for exponent cosine function

4.3 Problem 3

Try using the WHILE and the IF statements to calculate all the Fibonacci numbers so that the sum of two consecutive numbers is smaller than 10,000. How many are even? How many are odd? Try to plot them.

Hints:

1. Matlab can increase the size of a vector as it is being created.
2. To determine whether a number n is even or odd you can use the function **rem(n,2)**. If **rem(n,2)** equals 0 then the number is even, otherwise it is odd.

```
maxSum=10000;
fibo=fibonacci_numbers(maxSum);
l=tiledlayout(1,1);
str=sprintf('sum of two consecutive numbers less than %d',maxSum);
title(l,{'Fibonacci numbers with', str})
len=length(fibo);
fibo_even=[];
fibo_odd=[];
nexttile
hold on
xlim([0 len])
for i = 1:len
    n=fibo(i);
    if(rem(n,2)==0)
        fibo_even(end+1)=n;
        stem(i,fibo(i),'rs-','Linewidth',1.5)
    else
        fibo_odd(end+1)=n;
        stem(i,fibo(i),'bo-','Linewidth',1.5)
    end
end
xlabel('Index')
ylabel('Fibonacci Number')
legend('Even', 'Odd');
fprintf('Total fibonacci numbers: %d \n', len);
fprintf('Even fibonacci numbers: %d \n', length(fibo_even));
fprintf('Odd fibonacci numbers: %d \n', length(fibo_odd));


%% collect all the fibonacci numbers whose consecutive sum is  less than maxSum i
    .e 10000
function fibo_numbers = fibonacci_numbers(maxSum)
    f1=0;
    f2=1;
    fibo_numbers=[f1 f2];
    while (f1+f2) < maxSum
        next=f1+f2;
        f1=f2;
        f2=next;
        fibo_numbers(end+1)=next;
    end
end
```

Code 4: Matlab code for calculation related to fibonacci numbers and plotting result

Response of Command Window

```
>>p3
Total fibonacci numbers: 21
Even fibonacci numbers: 7
Odd fibonacci numbers: 14
```

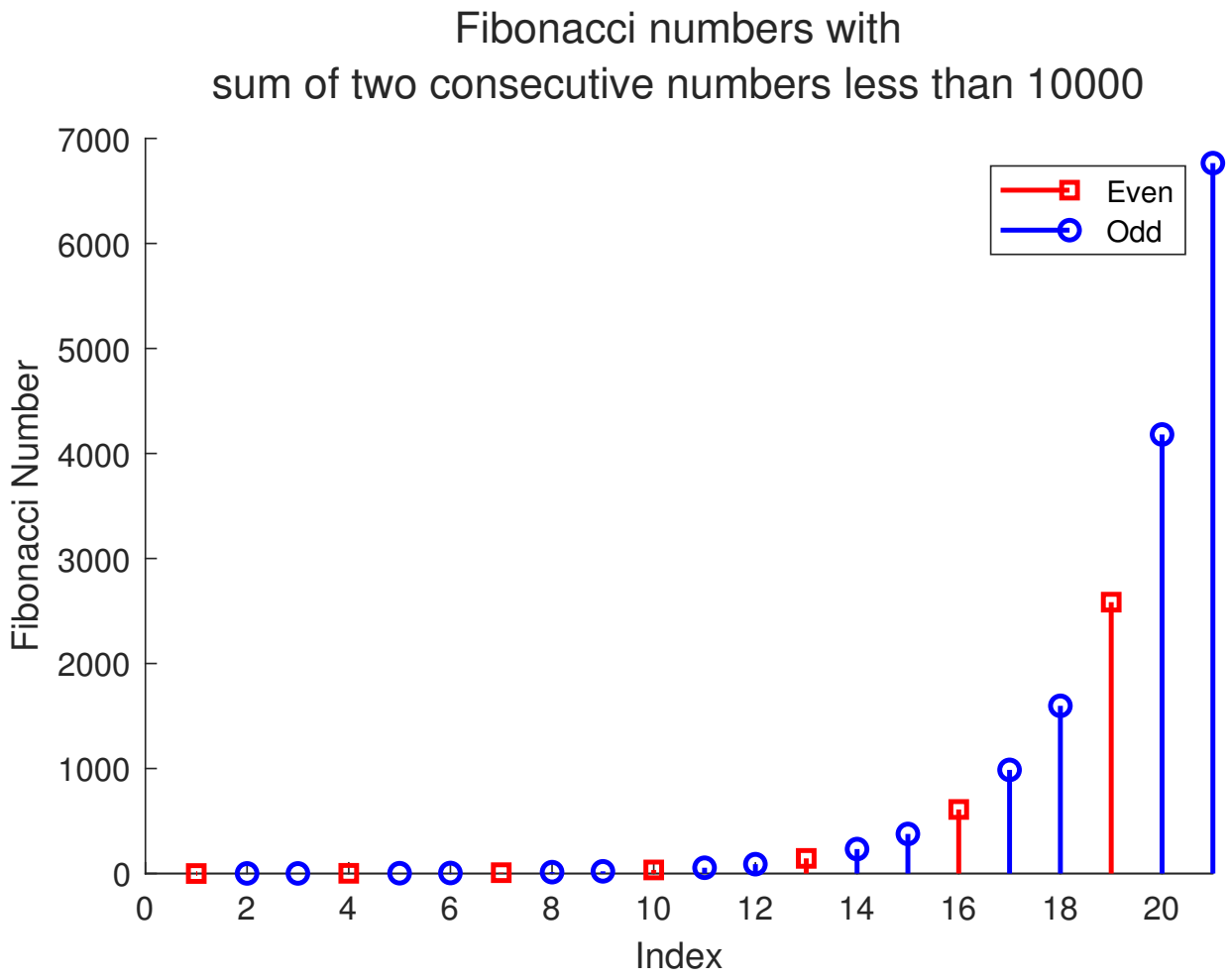

Figure 2: Plot for fibonacci numbers with sum of two consecutive numbers smaller than 10,000

4.4 Problem 4

Given $f(x) = \frac{x^2+2x+3}{x+3}$**. Plot** $f(x)$ **for** $0 \leq x \leq 100$

```
x=linspace(0,100,1000);
f=(x.^2+2.*x+3)./(x+3);
l=tiledlayout(1,1);
title(l,{'Plot for f(x) for 0 \leq x \leq 100'})
nexttile
plot(x,f,'Linewidth',1.5)
xlabel('$x$','interpreter','latex')
ylabel ('$f(x)=\frac{x^2+2x+3}{x+3}$','interpreter','latex')
```

Code 5: Matlab code for plotting the function f(x)

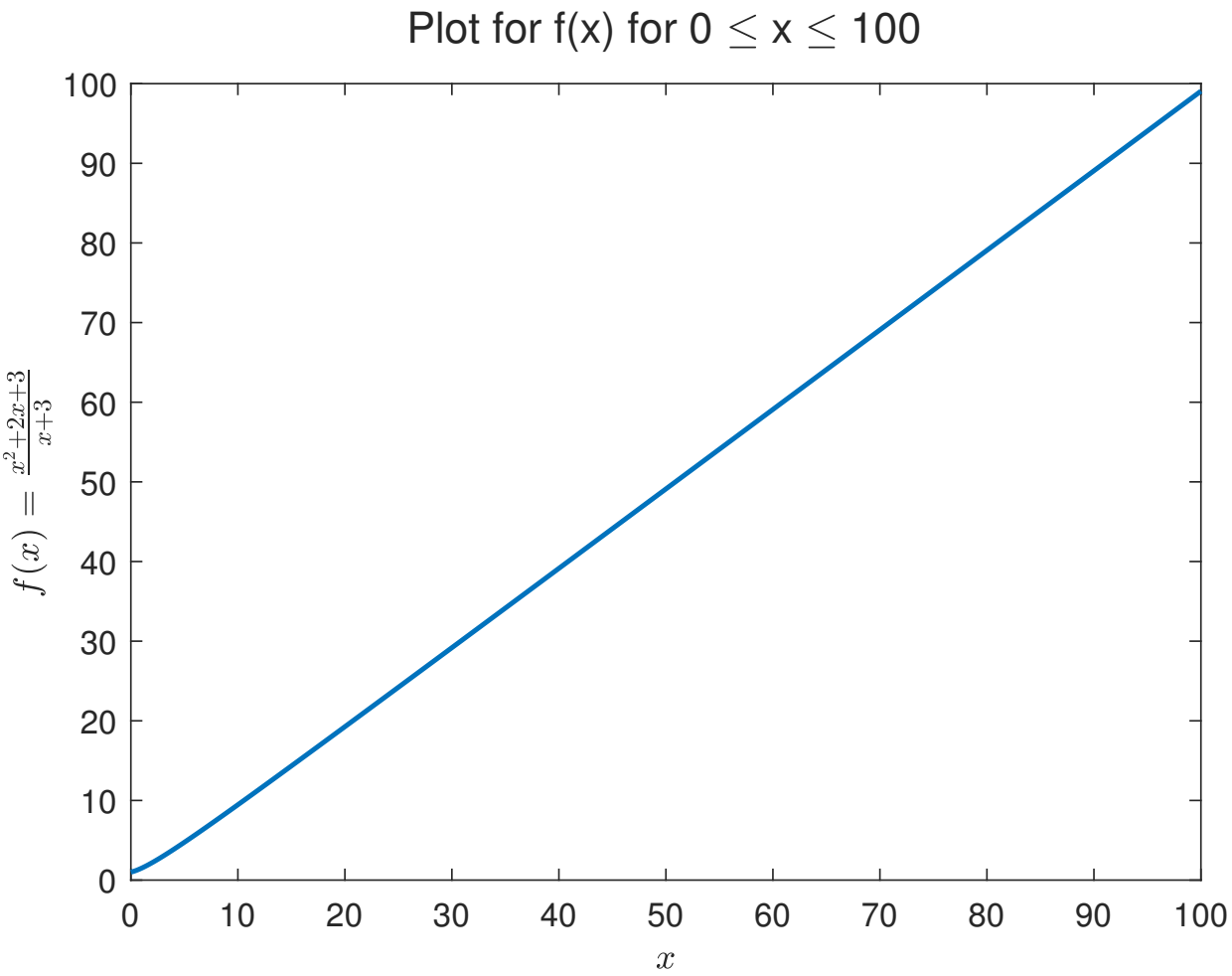

Figure 3: Plot for given f(x) function where x is between 0 and 100

5 Discussion and Conclusion

In this Lab we familiarize ourself with Programming with Matlab. We have learned how to use Matlab to solve problems in the areas like linear algebra, polynomial roots, sound files, complex numbers, signal processing and image processing. We have also learned how to use Matlab to solve problems in the areas like transfer function representation, frequency response, and plotting functions. We have also learned how to use Matlab to solve problems in the areas like calculating Fibonacci numbers and plotting them.