MATLAB MODULE 1

MATLAB Window Environment and the Base Program

Developing user-defined Functions

Suppose you want to write a function to generate and plot *N* data points of the exponentially-decaying sinusoid of amplitude *A*, frequency *f*, and exponential time-constant τ. A MATLAB function which can take all these parameters as arguments and provide decaying sinusoid as an output is shown in Fig. M1.20.

Generate an M-file with the code given in Fig.M1.20. When you attempt to save the file, editor will automatically assign a function name **decaying_sin** to your M-file. Save the function to the current working directory or include your personal directory into MATLAB search path by using **addpath** command. We must avoid duplicating function names with built-in MATLAB functions or keywords.

A function file begins with a function definition line, which contains the keyword **function**, a well-defined list of input and output arguments, and function name. Some examples of function definition are given below (execute **help function** for details):

function [ph, mag] = complex (z)

function [r,theta] = polar_form(a, b)

function decaying_sin( )

function sys_response( )

The first commented line just after the function definition is called the *H1 line*. This line is automatically catalogued in the **contents.m** file of the directory in which function file resides. This allows the line to be searched by the **lookfor** command. Very carefully chosen keywords related to your function should come in the *H1 line*. Note that if there is any

blank space before the **%** sign in the *H1 line*, then it is no longer *H1 line* !!

All comment lines immediately following the function definition line and immediately before the first executable statement of the function are displayed by MATLAB if **help** is asked on the function. Type

>> help decaying_sin ↵

and see what MATLAB displays.

Input-argument names used in the function are local to the function; so arbitrary variable names can be used to call the function. The name of another function can also be passed as an input variable. The following are examples of legal function calls:

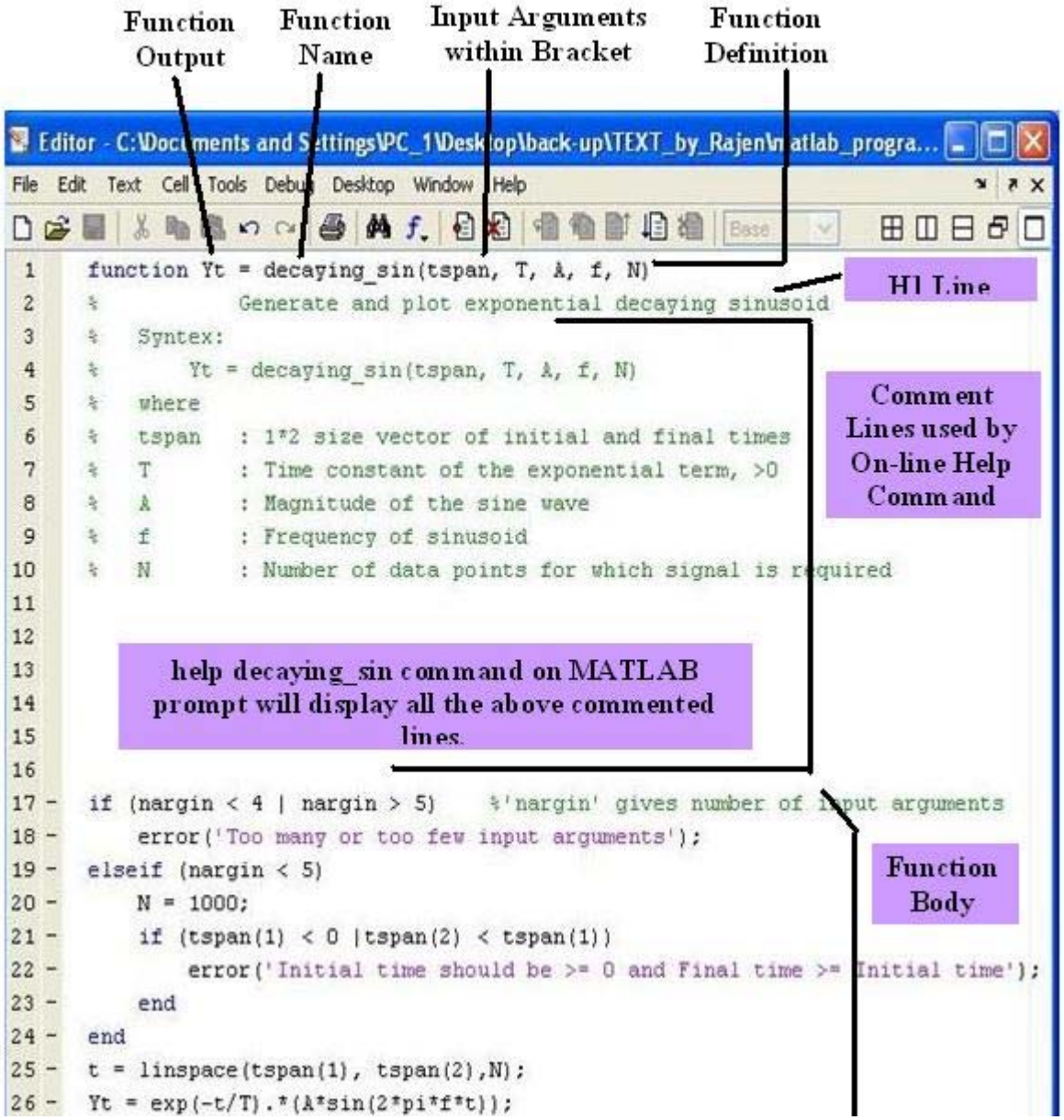
Function Output
Function Name
Input Arguments within Bracket
Function Definition
Editor - C:\Documents and Settings\PC_1\Desktop\back-up\TEXT_by_Rajen\matlab_progra...
File Edit Text Cell Tools Debug Desktop Window Help
function Yt = decaying_sin(tspan, T, A, f, N)
% Generate and plot exponential decaying sinusoid
H1 Line
% Syntex:
% Yt = decaying_sin(tspan, T, A, f, N)
% where
% tspan : 1*2 size vector of initial and final times
% T : Time constant of the exponential term, >0
% A : Magnitude of the sine wave
% f : Frequency of sinusoid
% N : Number of data points for which signal is required
Comment Lines used by On-line Help Command
help decaying_sin command on MATLAB prompt will display all the above commented lines.
if (nargin < 4 | nargin > 5) %'nargin' gives number of input arguments
error('Too many or too few input arguments');
elseif (nargin < 5)
N = 1000;
if (tspan(1) < 0 |tspan(2) < tspan(1))
error('Initial time should be >= 0 and Final time >= Initial time');
end
end
t = linspace(tspan(1), tspan(2),N);
Yt = exp(-t/T).*(A*sin(2*pi*f*t));
Function Body

Fig. M1.20 *DECAYING_SIN Function*

>> Yt = decaying_sin (time_span, time_const, mag, freq, pts)

In this case, input variables **time_span, time_const, mag, freq**, and **pts** must be defined before the function call.

>> res = decaying_sin ([0 10], tau, 3, 50, 1000)

Here the input variable **tau** must be defined beforehand; all other inputs have been defined in the function call.

>> sig = decaying_sin ([0,10], 2, 3, 50, 100)

Here all the inputs have been specified in the call statement.

>> decaying_sin ([0,10], 3, 0.5, 60, 100)

In this case, the output is assigned to generic variable **ans**.

Exercise M1.12

1. Execute **decaying_sin** function with less than four and greater than five arbitrary arguments and interpret the messages. Study **nargin** and **nargout** commands through online help.

2. Execute **decaying_sin** function with **tspan = [-1 10]** and **tspan=[3 0]** and interpret the messages.

3. Execute the function as **Yt = decaying_sin([0 5], 3, 0.5, 50).** What is the length of the vector **Yt** ?

4. In some applications we don't want to generate the plot every time. We may need only output given by the function. Modify the above function with one more input argument '**str**' to provide control over whether we want to generate the plot or not? If **str = 'Y'** or **'y'**, then it should generate the plot. If **str =**

'N' or **'n'**, then it should not. If any other character is passed, then it should give the error message **'Unrecognized user input'** and terminate the program. You also need to modify processing with the **nargin** command.

Exercise M1.13

Write a function named **specifications** with the following input and output arguments:

Input arguments:

1. 1×2 vector **tspan** of initial and final times.
2. Scalars **y0** ≥ 0, **zeta** ≥ 0, **wn**, and **theta**.
3. Character argument **str** controlling whether to plot the final result or not.
4. Scalar **N** of number of data points required to be generated.

Output arguments:

1. **Yt** calculated by $\frac{y_0}{\sqrt{1-zeta^2}} e^{-zeta*\omega_n*t} \sin\left(\omega_n\sqrt{1-zeta^2}\,t + theta\right)$
2. **Tau** calculated by $\frac{1}{zeta*wn}$
3. **Tr** calculated by $\frac{\pi - theta}{wn\sqrt{1-zeta^2}}$
4. **Tp** calculated by $\frac{\pi}{wn\sqrt{1-zeta^2}}$

5. **Mp** calculated by $e^{\frac{\pi * zeta}{\sqrt{1-zeta^2}}}$