University of Glasgow

James Watt School of Engineering

Simulation of Engineering Systems 3

and

Simulation of Aerospace Systems

Practical Session 2: An Introduction to Continuous System Simulation in Matlab

1. Introduction

The objective of this practical session is to provide an introduction to the principles of continuous system simulation and experience of using simulation programs written in MATLAB.

MATLAB is a mathematical analysis package designed for a wide range of engineering applications. It is a commercial software package that can perform calculations on matrices, row vectors and column vectors with ease. It is widely used in both industry and academic institutions. It possesses many of the features of a high level numerically oriented programming language, but in addition has a large collection of built-in functions for performing matrix and vector operations in a way that is very simple for the user. In particular, MATLAB has a set of powerful numerical integration routines which can be used for simulation of continuous dynamic systems.

MATLAB commands can be entered one line at a time or the user can write programs of MATLAB code and define his or her own functions. In this laboratory, we will use a MATLAB program and some one-line-at-a-time interpretive commands.

This practical session involves a set of simple exercises intended to introduce you to the use of MATLAB as a simulation program. Having completed these exercises, you should be able to write your own simulation program in the MATLAB language for different applications.

2. Van der Pol's Equation

Van der Pol's equation is a second order nonlinear differential equation.

$$\frac{d^2y}{dt^2}+\mu\left(y^2-1\right)\frac{dy}{dt}+y=0$$

This represents a particular form of electronic feedback oscillator in which the nonlinear term involving *dy/dt* can give rise to self-excited oscillations. Since the problem is nonlinear, no simple analytical approaches are available and simulation is essential if one wishes to investigate this mathematical model.

2.1 Before using the computer

Complete the following tasks.

(a) By selecting appropriate *state variables*, derive the state space model for Van der Pol's oscillator.

(b) Explain why this model cannot be expressed in linear *Standard State Space Form*.

Having completed these two tasks, now examine the listings of the computer program vandpol.m, model.m, rk4int.m and eulerint.m, which are included in the lecture notes and can be found on the course moodle page. Find the DYNAMIC segment in vandpol.m. The DERIVATIVE section is represented by function model.m and the INTEG section represented by function eulerint.m. Compare the equations you derived in task (a) with those found in model.m.

Read through the rest of the listing for vandpol.m and note carefully all the information provided in the comment lines.

A graph of the system state time response appears on the computer screen after the program has successfully executed. This is a graphical plot of the states of the system and is provided by the commands in the TERMINAL section of the program.

2.2 Practical Work: Van der Pol's Equation

MATLAB is started from Windows and information about the language, including a detailed help system, is available on-line.

On startup, MATLAB presents a command prompt in the Command Window. MATLAB can function as an interpreted language: commands can be typed in the keyboard, and are executed when the return key is pressed. MATLAB can also run programs that can exist in two forms – as a script file or as a function file. Both types have the .m suffix. A script file consists of several MATLAB commands and runs as if those commands had been typed at the command line. All the variables in a MATLAB script file are effectively workspace variables. That is, they are accessible from the MATLAB command line as well as within the script file. This means that any variable allocated in the script file can be accessed from the MATLAB command line afterwards. This also means that MATLAB script files should start with the command *clear all* to delete all existing variables. Functions on the other hand take arguments and return output values.

To get help in MATLAB simply type **help** followed by the name of a function or command.

Go back to the MATLAB window. To run a script file, simply type the name of the file without the .m suffix.

To start a simulation program, type the name of that program, so for the first example, type **vandpol**. The graphical output on the screen should be the same as in Figure 1.

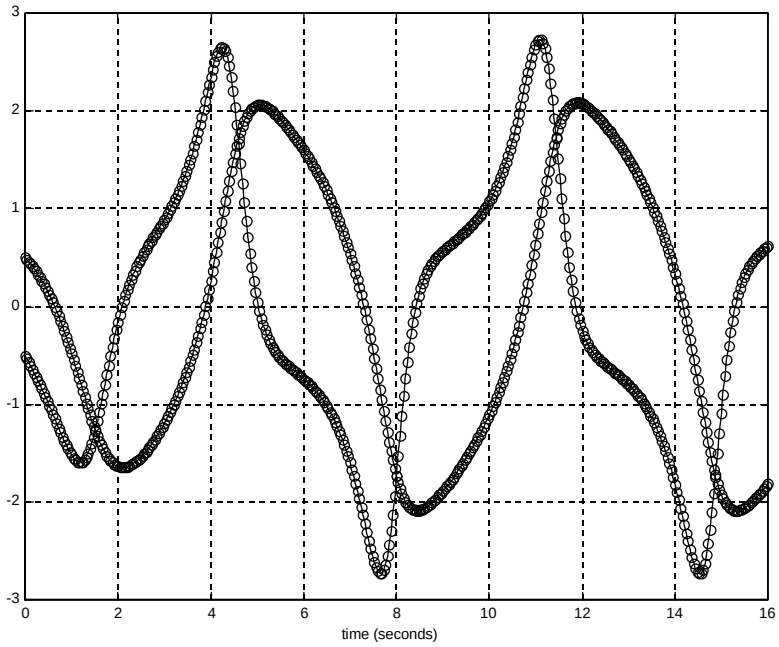

Figure 1: Time response of system states of Van der Pol's simulation

Alter the value of variable mu to 2.0 by editing file vandpol.m (used MATLAB editor) and re-run the simulation. After the simulation (or any other MATLAB script file) is finished, control returns to the MATLAB workspace environment. Any variables allocated during the program, including in this case the system state time responses, are stored in the MATLAB workspace and can be accessed for further processing or output from the command line. Type **whos** at the command line and you will see a list of variables in MATLAB's workspace. The variable *xout* should appear as a matrix of size 401x2. This is because the time response record consisted of 401 points for each of the two states. To select a vector corresponding to the first state, use xout(:,1) where the : operator denotes an entire row or column of a matrix).

The MATLAB plot commands in the TERMINAL segment are used to display the time responses for the two states. The plot command syntax is **plot(x,y)** where x and y are vectors of the same length. To plot the first state with respect to time (stored in tout) type

```
clf

plot(tout,xout(:,1))
```

The same command can be used to plot the second state. Use the commands **xlabel**, **ylabel** and **title** to label your plots.

Alternatively, plots can be placed in separate subplots within the figure window by using the **subplot** command i.e.

```
clf
subplot(2,1,1)
plot(tout,xout(:,1))
ylabel('x1'); xlabel('time [s]')
subplot(2,1,2)
plot(tout,xout(:,2))
ylabel('x2'); xlabel('time [s]')
```

From your lecture notes, replace the DERIVATIVE and INTEG sections of vandpol.m with a fourth order Runge-Kutta routine (rk4int.m). Both sections have to be replaced since this numerical integration routine evaluates the model function four times. Once this is working properly, compare your responses with those obtained above.

Try plotting of one state against the other state. This should be similar to the plots obtained in the Practical Session 1. Carefully consider what these plots are telling you about the oscillator.

Repeat the process for **three** other values of the constant mu between 0.1 and 2, and for **two** other sets of initial conditions with mu at its original value (i.e. mu = 1.0). What happens with a negative value of mu?

2.3 Practical Work: Modified Oscillator

Copy the file vandpol.m to another file vandpol2.m and edit it to set up a simulation for the system described by the equation,

$$\frac{d^2y}{dt^2}+\mu\left(y^2-0.3y^3-1\right)\frac{dy}{dt}+y=0$$

Set up a function model2.m to represent this system and alter the main simulation program to call this function.

Investigate the behaviour of this system for **three** different values of mu (μ) and **two** different initial conditions. Obtain appropriate graphical output.

2.4 Practical Work: A Chaotic System

Build a model of the non-linear system:

$$\begin{aligned}\dot{x} &= 10(-x+y) + u\\ \dot{y} &= 28x - y - xz\\ \dot{z} &= -\frac{8z}{3} + xy\end{aligned}$$

(a) Investigate the behaviour of this system in the open loop (i.e. when $u = 0$). Simulate for an appropriate length of time.

(b) Investigate the effect of the state feedback control law:

$$u = -(24x + 18.64y)$$

Dr Wenjuan Song

September 2025