Lecture 6 – Numerical Methods for Aeroelasticity

This lecture will provide an introduction to several numerical methods generally used in aeroelastic computations and we will use MATLAB as the main numerical tool.

Dynamical Systems

Aeroelastic systems can normally be categorized as dynamical systems. This can be more clearly explained by revisiting lecture 4 where we derived the equations of motion of a two degree-of-freedom aerofoil.

The motion of the structure is described by a pair of second order ODEs, hence, given the initial conditions of the system, we can find the solution by integrating the system to determine the motion as a function of time.

Ideally, we would like to perform such a calculation analytically, i.e. to determine expressions for h or α as functions of time. This method would be possible for simple cases such as a mass-spring system, but for a complex, especially nonlinear, system, this approach may not be possible.

Therefore, we need an alternative method to determine the solution of such complex differential equations.

Numerical Solutions of First order ODEs

Our 2-DOF aeroelastic system (from lecture 4) is clearly a second order ODE problem, therefore, we need to transform it into a system of first order ODEs.

For simplicity, we will use the equation of motion of an unforced mass spring system as an example.

$$m\ddot{x} + c\dot{x} + kx = 0 \qquad \text{[EQN.1]}$$

where x denotes the displacement of the mass
m denotes the mass
c denotes the damping constant
k denotes the spring stiffness

Now let us introduce a new variable

$$\dot{x} = y \qquad \text{[EQN.2]}$$

With the new variable we can rewrite equation 1 as

$$\dot{y} = -\frac{c}{m} y - \frac{k}{m} x \qquad \text{[EQN.3]}$$

Combining equations 2 and 3 into a matrix format, we obtain

$$\begin{Bmatrix} \dot{y} \\ \dot{x} \end{Bmatrix} = \begin{bmatrix} -\frac{c}{m} & -\frac{k}{m} \\ 1 & 0 \end{bmatrix} \begin{Bmatrix} y \\ x \end{Bmatrix} \qquad \text{[EQN.4]}$$

$$\dot{\mathbf{z}} = \mathbf{K}\mathbf{z}$$

where $\mathbf{z} = \begin{Bmatrix} y \\ x \end{Bmatrix}$ is the state vector

K denotes the stiffness matrix

Equation 4 shows a system of first order ODEs written as a matrix equation. The system of ODEs in this example is linear and homogeneous

The solution of equation 4 will be in the form of **z**(*t*), the displacement and velocity of the mass as functions of time. Clearly it can be determined by perform an integration with respect to time to equation 4.

We will now attempt to solve equation 4 using different types of numerical integration schemes.

1. Crude approximation

Very crudely, the ODE can be solved using the following approximation

$$z_{n+1} = z_n + \left(\frac{dz}{dt} \Delta t \right)_n$$

2. Runge Kutta 4th order algorithm

This algorithm provides a more accurate numerical solution than the previous method. The improved accuracy is achieved by introducing four intermediate stages between each integration step, hence the name fourth order.

3. MATLAB ode45 function

This numerical integration scheme by MATLAB is based on the adaptive time step Runge-Kutta-Fehlberg algorithm. The adaptive time step feature has been optimized so that it uses the least number of computation points to minimize the computing power requirement.

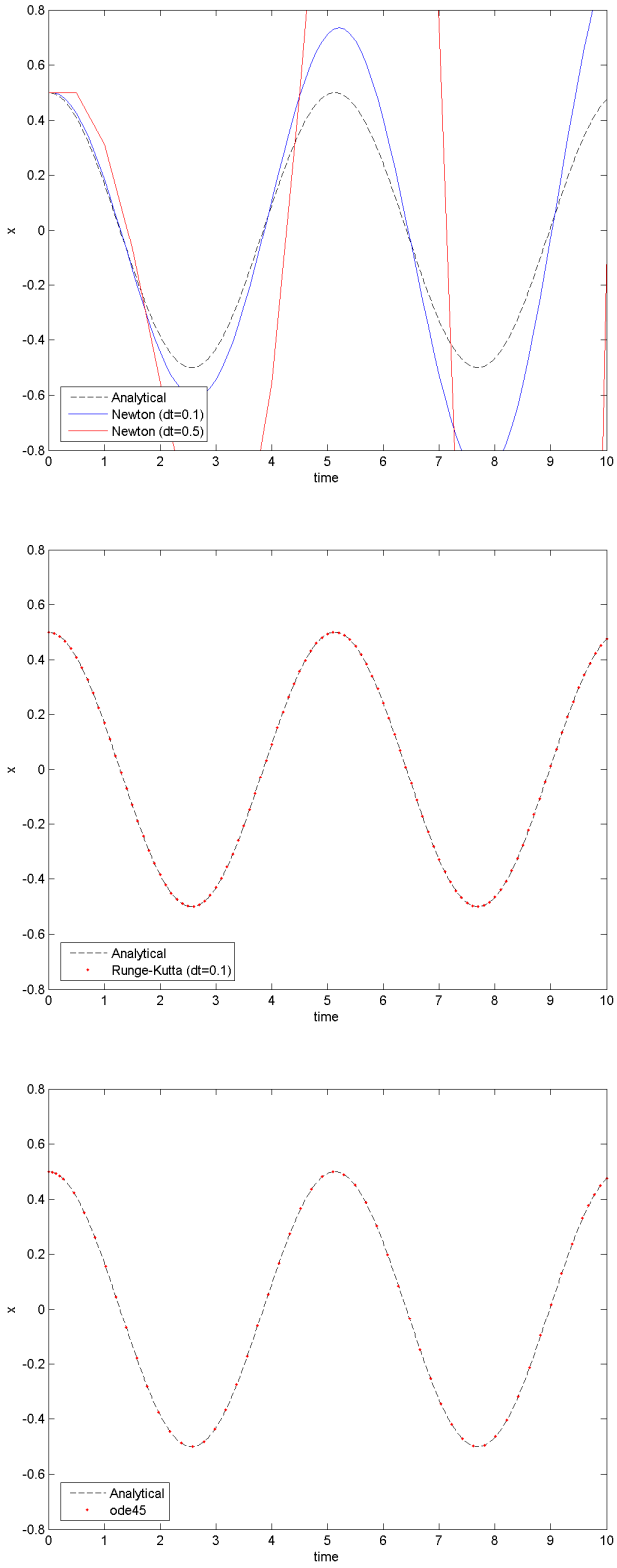

Figure 1 Comparison between three different integration schemes

MATLAB Basic Operations

This section will introduce a selection of basic MATLAB operations. The aim of this section is to provide adequate MATLAB programming skills to perform numerical integration and generate graphical results correctly.

1. Addition, Subtraction, Multiplication and Division

How do I use MATLAB to perform simple calculations?

```
1+1
10-59
3*5
25/9
5^2
sqrt(49)
```

2. Variables

How do I solve an equation with variables?

```
a = 1
b = 3
a + b
a - b
a * b
c = 5*a – 9*b
```

3. Matrices

How do I perform matrix calculation with MATLAB?

```
A = [15 30]
B = [0.5 0.8]
C = [20; 5]
D = A*C
E = C*A
F = 100:10:250
G = F'
H = F*0.25 - 20
```

4. Functions

How do I write a code to perform a given task?

A function in MATLAB refers to a self-contained set of MATLAB codes designed to perform a specific task. The software will create a new workspace when running a function.

Here is an example of a MATLAB function which is written to determine the area of a triangle

```
function area_of_triangle
base = 5;
height = 10;
area = base * height / 2;
disp(['Area of triangle is ', num2str(area)])
disp(' ')
```

5. Plots

How do I show my results graphically?

We must have at least two sets of data stored in arrays in the workspace before we can plot a graph. Suppose we want to plot a graph of $y = \sin(kx)$, we must first define a range of values of x – for example between *0* and π, and the constant k. Finally, we can compute the corresponding values of y. Only then we will have enough data to plot the graph $y = \sin(kx)$.

```
function sample_plot
% define value of k
k = 0.4;
% define range of x
x = 0:0.01:pi;
% compute the values of y
y = sin(k*x);
% plot graph of y vs x
figure(1)
plot(x,y,'.-k')
xlabel('x')
ylabel('y = sin(kx)')
```

6. MATLAB's ode45

How do I perform numerical integration efficiently?

Analysis of Numerical Integration Solutions

We can analyse the numerical integration solutions in a number of ways, with several different tools.

First we must be familiar with some basic terms such as transient and steady state.

1. Time History

Time history is the simplest representation of a dynamical system solution. It is constructed by plotting the value of the state variable against time. The time history is a very useful tool when analyzing the transients of the solution. The examples below are time histories of solution of the Duffing's equation, $\ddot{x} + k\dot{x} + x^3 = B\cos(t)$

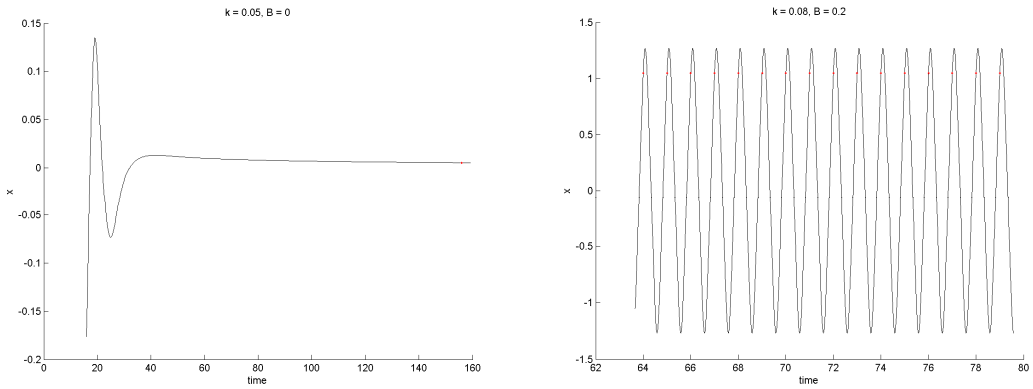

Figure 2 Examples of time history plots

2. Phase Portraits

The phase portrait is constructed by plotting a state variable against another, i.e. in this example the displacement of the block is plotted against the velocity. This representation is useful when analyzing the steady state of the solutions. The following phase portraits correspond to the time histories in the previous section.

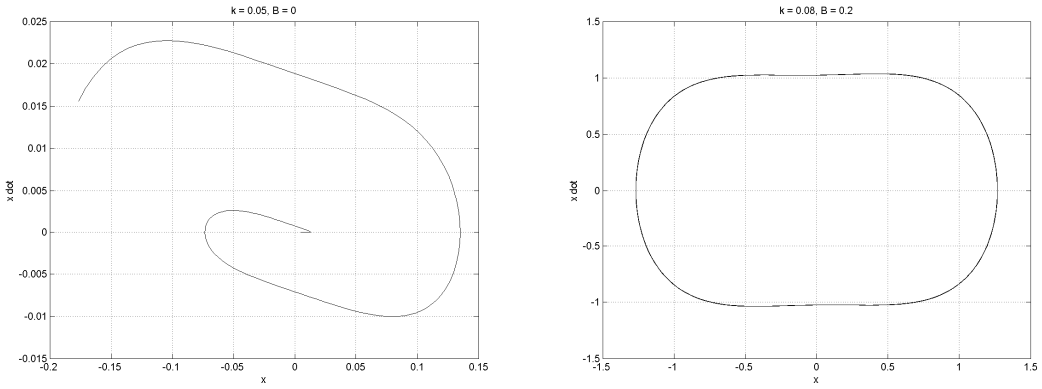

Figure 3 Examples of phase portraits

3. Attractors

An attractor, or a stable steady state, may be represented in one of the following forms

(a) fixed point
(b) periodic solution
(c) chaotic solution

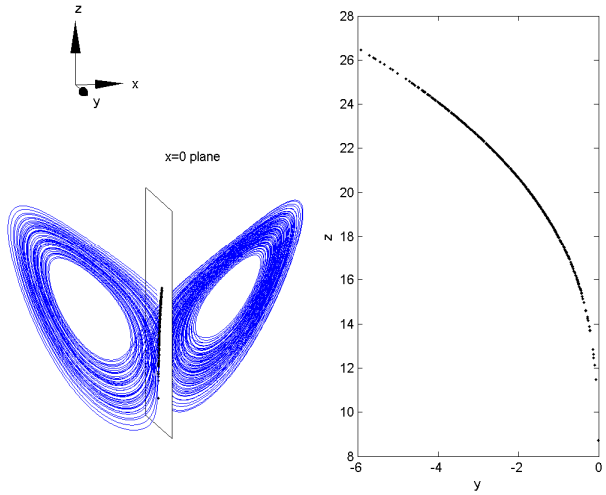

Figure 4 A phase portrait of a chaotic attractor (Lorenz chaotic attractor)

4. Bifurcation diagrams

A bifurcation diagram usually shows an evolution of the steady state of the system as a parameter is varied.