DOING PHYSICS WITH MATLAB

DYNAMICS OF OSCILLATING AND CHAOTIC SYSTEMS BACKGROUND THEORY and RUNGE-KUTTA METHOD

Ian Cooper

School of Physics, University of Sydney

ian.cooper@sydney.edu.au

DOWNLOAD DIRECTORY FOR MATLAB SCRIPTS

chaos07A.m

The Runga-Kutta method is used to solve the equation for the motion for a rigid pendulum (animations of free, damped, and forced motions)

chaos08A.m

The Runga-Kutta method is used to solve the equation for the motion for a rigid pendulum – POINCARE sections

chaos01.m

The Runga-Kutta method is used to solve the equation for the motion for a Duffing two-well oscillator (animations of free, damped and forced motions)

chaos02.m

The Runga-Kutta method is used to solve the equation for the motion for a Duffing two-well oscillator – POINCARE sections

Arguably the most broad-based evolution in the world view of science in the twentieth century will be associated with chaotic dynamics.

S.N. Rasband Chaotic Dynamics of nonlinear Systems.

INTRODUCTION

Equations of motion for a variety of oscillating and chaotic systems are solved numerically using the **Runga-Kutta Method**. Graphical outputs are used to display the results of the simulations. For example, displacement vs time; velocity vs time; acceleration vs time; energy (conservative forces) vs time; **phase space** plots - displacement vs velocity and **Poincare sections**. The motion of the system is animated and the animation can be saved as an animated gif file. You can change the code to save the animation an avi file.

We will consider the dynamics of simple linear oscillating systems then more complicated nonlinear systems that exhibit chaotic behaviour. A system that exhibits chaotic behaviour is not one where the motion is random. When the trajectory of a dynamical system is calculated again and again with the same initial conditions, you will always get the same result. The unpredictability – the chaotic motion – is a result of the fact that even small differences in the initial conditions are amplified when solving the equation of motion to give enormously different results, and so, it becomes impossible to make predictions.

EQUATION OF MOTION

The motion of a system is governed by Newton's Second law which can be expressed as

$$(1) \qquad a(t) = \frac{d^2x(t)}{dt^2} = \frac{F(t,x,v)}{m}$$

where $F(t,x,v)$ is the resultant force acting upon the a system of mass m.

The Runga-Kutta Method used to solve the equation of motion is outlined below. Given a set of initial conditions [displacement x(0) and v(0)], the displacement and velocity are computed at successive time step $dt \equiv h$:

$$(2a) \quad x(t+h) = x(t) + h\left[v(t) + \left(k_1 + k_2 + k_3\right)/6\right]$$

$$(2b) \quad v(t+h) = v(t) + \left(k_1 + 2k_2 + 2k_3 + k_4\right)/6$$

where the Runga-Kutta coefficients are given by

(3)

$$k_1 = h\,F\left(t, x(t), v(t)\right)$$
$$k_2 = h\,F\left(t + h/2, x(t) + h\,v(t)/2, v(t) + k_1/2\right)$$
$$k_3 = h\,F\left(t + h/2, x(t) + h\,v(t)/2 + h\,k_1/4, v(t) + k_2/2\right)$$
$$k_4 = h\,F\left(t + h, x(t) + h\,v(t) + h\,k_2/2, v(t) + k_3\right)$$

MATLAB

Matlab scripts are used to model various oscillating systems by solving the differential equation governing the motion using the Runge-Kutta method. All scripts have a common structure. All changes to a model or variables are done within the script.

ANIMATION SETUP

The animated motion of the oscillating system can be saved as a gif file. The adjustable variables are:

f_gif = 0	animations not saved
f_gif = 1	file will be saved
ag_name	file name
delay	determines the speed at which the animated gif will be displayed

CONSTANTS (VARIABLES)

States the physical meaning of some of the system parameters and sets default values.

INPUT

Values for the model parameters, time scale and the initial conditions.

CALCULATIONS

Runga-Kutta Method for finding the displacement and velocity as functions of time is done by calling the two functions. The functions for a rigid pendulum are

```
% FUNCTIONS ============================================

% Runga-Kutta coefficients
function [k1, k2, k3, k4] = coeff(t,h,x,v)
  k1 = h*fn(t,x,v);
  k2 = h*fn(t+h/2, x+h*v/2, v+k1/2);
  k3 = h*fn(t+h/2, x+h*v/2+h*k1/4 ,v+k2/2);
  k4 = h*fn(t+h,   x+h*v+h*k2/2,   v+k3);
end

% Equation of motion
function  y = fn(t,x,v)
  global c
  y = - c(1) * v -(c(2) + c(4)*cos(c(5)*t))*sin(c(3)*x);
end
```

After the displacement and velocity calculations are complete, the following parameters are calculated: acceleration; conservative force acting on the system; kinetic energy; potential energy due to conservative forces; total energy. For the rigid pendulum its position in space is also calculated. The script for the rigid pendulum is

```
% Acceleration  a
    aA = gradient(vA)/h;
 % Force (restoring force)  Fc [N]
    Fc   = -m*g*sin(xA);
 % Kinetic Energy  K   [J]
    K = (0.5*m*L^2) .* vA.^2;
 % Potential energy due to conservative force Uc [J]
    Uc = (m*g*L).*(1-cos(xA));
 % Total Energy due to conservative forces [J]
    Ec = K + Uc;
 % XY coodinates of the pendulum
    X = L.*sin(xA);
    Y = -L.*cos(xA);
```

SIMULATIONS

Rigid Pendulum

Pendulum with viscous damping

Pendulum with viscous damping driven by an external force – the road to chaos

Duffing two-well oscillator: Animation; time and displacement graphs; phase space plot

Duffing two-well oscillator: Poincare sections