

Joint ICTP-IAEA College on Advanced Plasma Physics  
ICTP  
Trieste, Italy  
18 – 29 August 2014

Course: **Numerical methods and simulations**

Tutor: Bengt Eliasson  
E-mail: [bengt.eliasson@strath.ac.uk](mailto:bengt.eliasson@strath.ac.uk)  
Homepage: <http://www.strath.ac.uk/physics/staff/academic/bengteliasson/>

Lecture 2: Simulation of systems of ordinary differential equations

- Will continue to write programs in Matlab
- Learn how to write loops and functions
- Simulate ordinary differential equations
  - Lotka-Volterra equation: A model for predator-prey systems (for example foxes and rabbits).

Assignment: Modify the program to solve the Rössler system, which is used to study chaos (the Rössler attractor).

To solve a differential equation numerically in time, we discretize the time so that the solution is only defined at discrete points in time, separated by the timestep  $\Delta t$ . Then we find the solution iteratively at later times by using the solution at earlier times.

The Runge-Kutta algorithm:

To solve the time-dependent system of equations

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{F}(\mathbf{x}(t), t), \quad (1)$$

with the initial condition  $\mathbf{x} = \mathbf{x}_0$ , where  $\mathbf{F}$  is a vector-valued function of the vector  $\mathbf{x}$  and of  $t$ , we use the Runge-Kutta algorithm

- (0)  $\mathbf{x} = \mathbf{x}_0$
- (1)  $\mathbf{R}_1 \leftarrow \mathbf{F}(\mathbf{x}, t)$
- (2)  $\mathbf{R}_2 \leftarrow \mathbf{F}(\mathbf{x} + \Delta t \mathbf{R}_1 / 2, t + \Delta t / 2)$
- (3)  $\mathbf{R}_3 \leftarrow \mathbf{F}(\mathbf{x} + \Delta t \mathbf{R}_2 / 2, t + \Delta t / 2)$
- (4)  $\mathbf{R}_4 \leftarrow \mathbf{F}(\mathbf{x} + \Delta t \mathbf{R}_3, t + \Delta t)$
- (5)  $\mathbf{x} \leftarrow \mathbf{x} + (\Delta t / 6)(\mathbf{R}_1 + 2\mathbf{R}_2 + 2\mathbf{R}_3 + \mathbf{R}_4)$   
 $t \leftarrow t + \Delta t$

This gives the solution  $\mathbf{x}$  at time  $t + \Delta t$ . The steps (1)–(5) are repeated with the new values of  $\mathbf{x}$  until we have reached the end of the simulation.

Example: Write a program that solves the Lotka-Volterra equation

$$\frac{dx}{dt} = ax - bxy \tag{2}$$

$$\frac{dy}{dt} = -cy + dxy \tag{3}$$

with the initial conditions  $x(0) = y(0) = 2$ , and where the constants are  $a = 1.5$ ,  $b = 1$ ,  $c = 3$  and  $d = 1$ . Plot the result. Here

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{4}$$

and

$$\mathbf{F}(\mathbf{x}, t) = \begin{bmatrix} ax - bxy \\ -cy + dxy \end{bmatrix} \tag{5}$$

We now write the main program "main.m" for our simulation. Select from the Matlab menu *New Script*, then from the menu of the new window, choose *Save / Save as*, and File Name: **main.m**

We now write the program in the file:

=====

```
% Main program main.m
clear %Clear all variables

Nt=1000; % The number of time steps

x=2; %Initial values
y=2;
```

```

t=0; % Start at time t=0
dt=0.02; % The time step

% Create vectors to store results
xvec=zeros(1,Nt);
yvec=zeros(1,Nt);
tvec=zeros(1,Nt);

% Store initial valuse
tvec(1,1)=0;
xvec(1,1)=x;
yvec(1,1)=y;

for j=1:Nt
    % Call the Runge-Kutta routine.
    [x,y]=RungeKutta(x,y,t,dt);
    t=t+dt; % Increase time

    %Store the results
    tvec(1,j+1)=t;
    xvec(1,j+1)=x;
    yvec(1,j+1)=y;
end

%Plot the results
subplot(2,1,1); plot(tvec,xvec); xlabel('t'); ylabel('x');
subplot(2,1,2); plot(tvec,yvec); xlabel('t'); ylabel('y');

```

=====

Then choose from the menu *Save*.

Now we write the Runge-Kutta subroutine: From the Matlab menu, choose *New Script*, then choose from the menu of the new file *Save / Save as*, and File Name: `RungeKutta.m`

In the new file, we write

=====

```

% Runge-Kutta alogrithm, RungeKutta.m
function [x,y]=RungeKutta(x,y,t,dt)

```

```

[R1x,R1y]=F(x,y,t);
[R2x,R2y]=F(x+0.5*dt*R1x,y+0.5*dt*R1y,t+0.5*dt);
[R3x,R3y]=F(x+0.5*dt*R2x,y+0.5*dt*R2y,t+0.5*dt);
[R4x,R4y]=F(x+dt*R3x,y+dt*R3y,t+dt);

x=x+dt/6*(R1x+2*R2x+2*R3x+R4x);
y=y+dt/6*(R1y+2*R2y+2*R3y+R4y);

```

=====  
Choose *Save*.

Finally, we write the function that defines the right-hand side of the differential equation. From the Matlab menu, choose *New Script*, and then *Save / Save as* and Filename: F.m

In the new file, we write

```

=====  

% The right-hand side of the differential equation, F.m  

function [Rx,Ry]=F(x,y,t)

%Lotka-Volterra equation

a=1.5;
b=1;
c=3;
d=1;

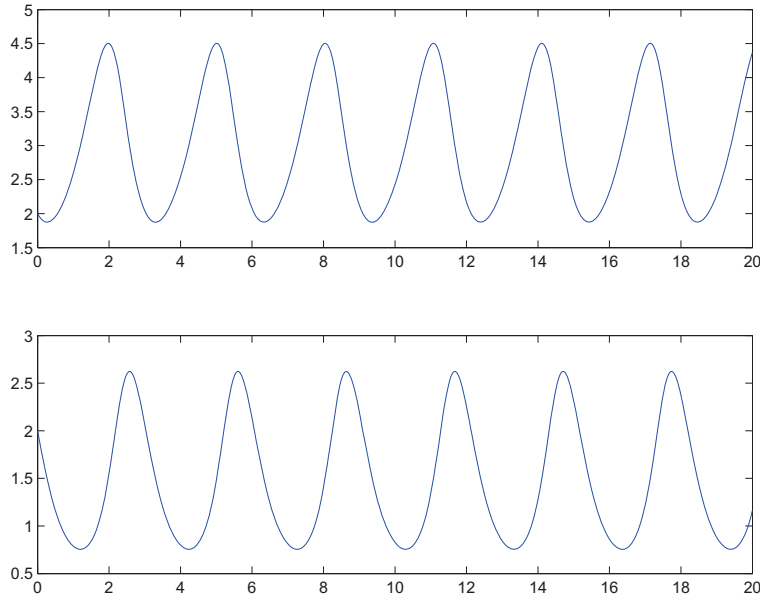
Rx=a*x-b*x*y;
Ry=-c*y+d*x*y;

```

=====  
Choose *Save*.

We now run the program. In the Matlab window, type in `main` (Enter).

The result is



The upper plot ( $x$ ) shows the prey (cheeps or rabbits), and the lower plot ( $y$ ) shows the predators (wolves or foxes) that catch the prey. When there are many rabbits, the foxes also increase, and when the number of rabbits decrease the number of foxes also decrease ...

**Assignment:** Modify the program so that it solves the Rössler system of equations:

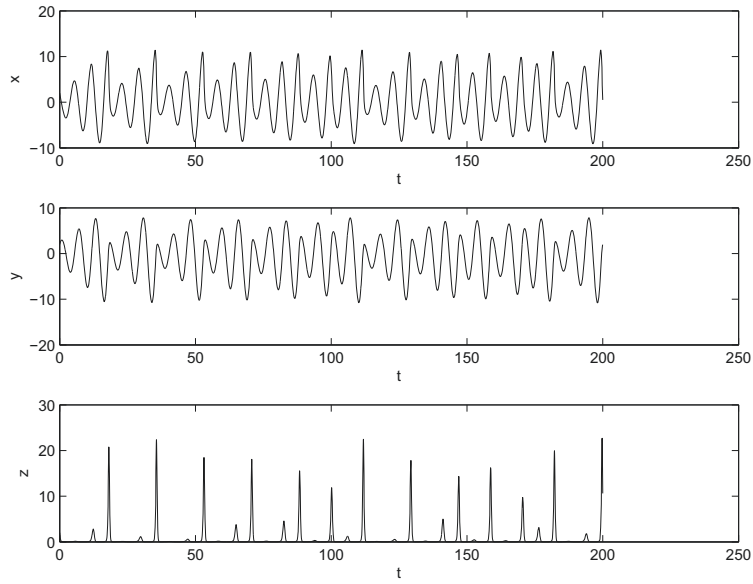
$$\frac{dx}{dt} = -y - z \quad (6)$$

$$\frac{dy}{dt} = x + ay \quad (7)$$

$$\frac{dz}{dt} = b + z(x - c) \quad (8)$$

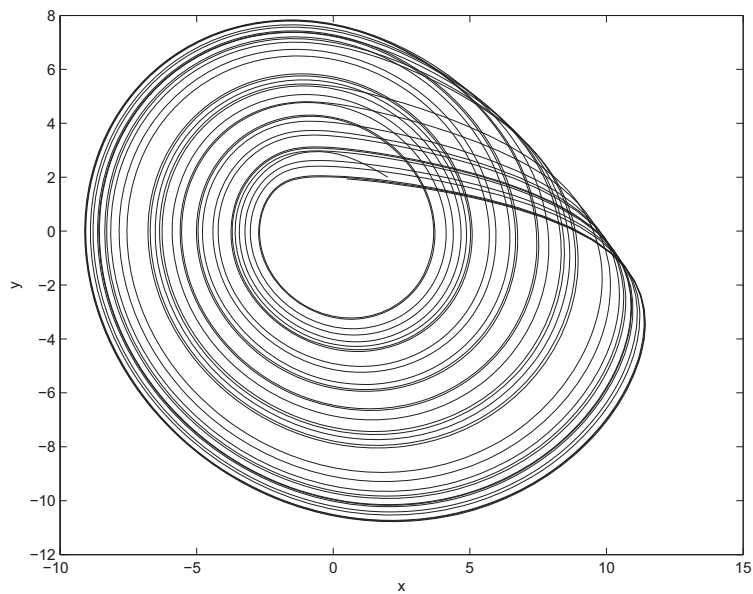
with  $a = 0.2$ ,  $b = 0.2$ ,  $c = 5.7$ , and with the initial conditions  $x = y = z = 2$ . Solve the system from  $t = 0$  to  $t = 200$  and plot  $x$ ,  $y$  and  $z$  in three subpanels of the same figure.

The result is

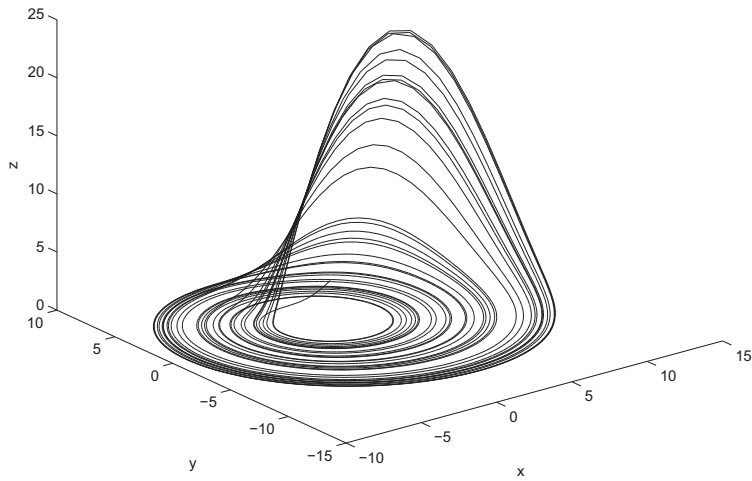


We see that the solution exhibits a very irregular and chaotic behavior.

Try also `plot(xvec,yvec)`



and `plot3(xvec,yvec,zvec)`



The solution is attracted to a fractal set in  $x$ ,  $y$ ,  $z$  space, called a "strange attractor".

Reference: O. E. Rössler, Physics Letters 57A, 397–398 (1976).