Computational Methods and Modelling

Antonio Attili & Edward McCarthy

antonio.attili@ed.ac.uk
ed.mccarthy@ed.ac.uk

School of Engineering
University of Edinburgh
United Kingdom

Solution tutorial 8
Solution of a system of differential equations

THE UNIVERSITY *of* EDINBURGH

Python code for the solution

- The complete code is available on Learn in the file solve_ivp_system_Lorenz.py.

Lorenz

```
# importing modules
import numpy as np
import matplotlib.pyplot as plt
import math
from scipy.integrate import solve_ivp
# ------------------------------------------------------
# functions that returns dy/dx
# i.e. the equation we want to solve:
def model(x,y):
    sigma = 10.0
    beta =8.0/3.0
    rho = 28.0
    #rho = 10.0
    y_1 = y[0]
    y_2 = y[1]
    y_3 = y[2]
    f_1 = sigma * (y_2-y_1)
    f_2 = rho * y_1 - y_2 -y_1 * y_3
    f_3 = -beta * y_3 + y_1 * y_2
    return [f_1 , f_2, f_3]
# ------------------------------------------------------
# initial conditions
x0 = 0
y0_1 = 5
y0_2 = 5
y0_3 = 5
# total solution interval
t_final = 30
# step size
# not needed here. The solver solve_ivp
# will take care of finding the appropriate step
# ------------------------------------------------------
# Apply solve_ivp method
t_eval = np.linspace(0, t_final, num=5000)
y = solve_ivp(model, [0 , t_final] ,
        [y0_1 , y0_2, y0_3],t_eval=t_eval)
# ------------------------------------------------------
```

- The output `y` of `y = solve_ivp(...)` is a data structure that contains several variables including
  - `y.t` - containing the time instants at which the solution has been stored
  - `y.y` - containing a 2D array with the solution. The first index of the array indicates the component of the solution (y_1 y_2 y_3), while the second index is the time instant. So `y.y[0,:]` is the full solution for the first variable y_1.

Python code for the solution

- The plots can easily done with the code below (also included on Learn in the file solve_ivp_system_Lorenz.py)

Lorenz

```
# -----------------------------------------------------
# plot results
plt.figure(1)
plt.plot(y.t,y.y[0,:] , 'b-',y.t,y.y[1,:]
        , 'r-',y.t,y.y[2,:] , 'g-')
plt.xlabel('t')
plt.ylabel('y_1(t), y_2(t), y_3(t)')
# -----------------------------------------------------

# -----------------------------------------------------
# plot results
plt.figure(2)
plt.plot(y.y[0,:] ,y.y[1,:],'-')
plt.xlabel('y_1')
plt.ylabel('y_2')
# -----------------------------------------------------

# -----------------------------------------------------
# plot results
plt.figure(3)
plt.plot(y.y[0,:] ,y.y[2,:],'-')
plt.xlabel('y_1')
plt.ylabel('y_3')
# -----------------------------------------------------

# -----------------------------------------------------
plt.show()
# -----------------------------------------------------
```

- Plots of the solution for $\sigma = 10$; $\beta = 8/3$; $\rho = 28$ (top) and $\sigma = 10$; $\beta = 8/3$; $\rho = 10$ (bottom).