Ordinary Differential Equations and Chaotic Systems

HW 4: Wednesday, Sep. 25, 2019
DUE: Wednesday, Oct. 9, 2019 (you have two weeks to do this)
READ: *Numerical Recipes in C++*, Section 17.0 and 17.1, page 899-910 on
4th order Runge-Kutta solution of differential equations
OPTIONAL: Landau, Paez, and Bordeianu,
ODE and Runge-Kutta, section 9.1-9.5.2
chaotic systems, section 12.1-12.14
F.C.Moon, *Chaotic and Fractal Dynamics*, Wiley 1992
discusses chaotic systems in more detail. The simulation of
one or more chaotic systems may make an interesting final project.

PROBLEM (10 points):

The Lorenz system [3] (not Lorentz from electricity and magnetism) was the first chaotic system discovered from attempts to find simple models of weather (temperature, pressure etc.). The Rossler system [5] is modeled after the Lorenz system and produces only one spiral instead of two as in the Lorenz system. The Rossler system is the following set of equations:

$$\frac{dx}{dt} = -y - z \tag{1}$$
$$\frac{dy}{dt} = x + ay \tag{2}$$
$$\frac{dz}{dt} = b + z(x - c) \tag{3}$$

where a, b and c are constants. Many other constants have been absorbed to make a simple dimensionless set of equations to solve. The goal of this homework is to solve this system numerically.

This is a nonlinear system and can exhibit chaotic behavior for some values of the parameters. Although there is no simple and precise definitions of a chaotic system, chaotic behavior is roughly defined as being bounded but not periodic or semi-periodic. Some other references are listed at the end.

To solve this system, first write a program using the 4th order Runge-Kutta method and test it. Write a general purpose subroutine for arbitrary order N (using dynamic memory with new/delete), similar to that described in rk4() in *Numerical Recipes*[4] that performs one time step. This should be a general routine so that you can test the same code on a problem with a known solution and then apply the identical code to an unknown problem. You may use rk4() directly or write your own version. The code in N.R. in C++ uses classes for the arrays. You may change these back to plain arrays or try the NR-vector class if you like. Vector class construction may be discussed in class shortly.

When developing code like this it is always advisable to test it on a system with a known answer to verify that it is working correctly. Therefore first test your program on the harmonic oscillator

equation (without damping):

$$\frac{d^2y}{dt^2} + \omega^2 y = 0 \tag{4}$$

where ω is a constant (angular frequency of oscillation). With initial conditions $y(t = 0)$=1 and $y'(t = 0)$=0 this has a known solution:

$$y(t) = \cos(\omega t) \tag{5}$$

The Runge-Kutta method requires this equation to be written as two first order equations. The harmonic oscillator equation can also be written as:

$$\frac{dy_0}{dt} = y_1 \quad ; \quad \frac{dy_1}{dt} = -\omega^2 y_0 \tag{6}$$

Plot $y(t)$ vs. t (as produced by your program) for several cycles using ω =1.0 to verify that your code is working properly. You should quantitatively test both the period and the amplitude. Also make a phase space plot of $y'(t)$ versus $y(t)$ which should produce a circle that closes on itself.

Next modify your program (using the 4th order Runge-Kutta method) to find the solution for $x(t)$, $y(t)$, and $z(t)$ described by the equations in the Rossler system. Use the following parameter values and initial conditions:

$$a = b = 0.2 \tag{7}$$
$$c = 5.7 \tag{8}$$
$$x(0) = 0 \tag{9}$$
$$y(0) = -6.78 \tag{10}$$
$$z(0) = 0.02 \tag{11}$$

Calculate $x(t)$, $y(t)$, and $z(t)$ for $0 < t < 200$. Vary the sampling size $h = \Delta t$ to get a stable answer. To save time you may just look at $x(t)$ vs. t to judge the stability or reproducibility (print out this graph). The chaotic behavior of this system also causes numerical stability problems over long times. You may find that you need many 1,000's of points. To save disk space etc. you only need to print out a few 1,000 points but evaluate many points in-between those you print out (i.e. you need a small h to get a stable numerical solution but you don't need to see everything in-between).

Also plot (using a good h) a) $z(t)$ vs. t, b) $y(t)$ vs. $x(t)$ and c) $z(t)$ vs. $x(t)$. You will find that the values of the parameters given above produce a chaotic behavior. This result is most interesting when viewed as a 3D plot (see optional-3 below).

Beware that (in C) 1/6=0 but 1.0/6.0=0.166666667 because the first is done using integer arithmetic and the second is done using floating point arithmetic.

OPTIONAL(1): Graph $y(t)$ vs. $x(t)$ for other values of the parameters (a,b,c). Some values may not produce chaotic behavior.

OPTIONAL(2): Try different starting point. Chaotic system are very sensitive to small changes in the initial conditions. Small changes produce similar behavior in the near term but large changes in the long term solution.

OPTIONAL(3): Try plotting (x,y,z) in a 3D plot (use plot3 in Matlab, python or equivalent). Many programs will let you grab the drawing with the mouse and move it around to get a better idea of the 3D shape. The following python code segment produces a 3D plot.

```
from pylab import *
from mpl_toolkits.mplot3d import Axes3D
:
fig = figure(5)
ax = fig.gca(projection='3d')
:
plot( x, y, z, 'k' )
:
```

References

[1] H. Haken, "Analogy Between Higher Instabilities in Fluids and Lasers", Phys. Lett. 53A, 1975, p.77-78

[2] H. J. Korsch and H.-J. Jodl, *Chaos, A Program Collection for the PC*, 2nd edit., (Springer-Verlag, 1999), section 12.4.

[3] E. N. Lorenz, "Deterministic Nonperiodic Flow", J. Atmos. Sci. 20, 1963, p.130.

[4] W. H. Press, S. A, Teukolsky, W. T. Vetterling and B. P. Flannery *Numerical Recipes, The Art of Scientific Computing, 3rd edit.*, Camb. Univ. Press 2007.

[5] O. E. Rössler, "An Equation for Continuous Chaos", Physics Letters A 57 (1976) p.397-398.

[6] S. H. Strogatz, *Nonlinear Dynamics and Chaos*, (Perseus Publishing, 1994), section 4.6.