Physics 750: Exercise 8

Tuesday, September 26, 2017

This exercise concerns the Lorenz attractor, a famous set of first-order differential equations that was originally proposed in 1963 to model fluid convection:

$$\frac{dx}{dt} = \sigma(y - x)$$
$$\frac{dy}{dt} = x(\rho - z) - z$$
$$\frac{dz}{dt} = xy - \beta z$$

The model is usually studied with $\sigma = 10$ and $\beta = 8/3$ fixed and ρ treated as a free parameter. For $\rho < 470/19 \doteq 24.74$, the system is dissipative and the initial transient dies out. As ρ passes above 470/19, the system becomes chaotic. The chaotic behaviour takes place on a butterfly-shaped attractor. There are many other values, e.g., $\rho = 99.96$, that produce interesting closed orbits.

Remember that in the chaotic regime, the dynamics of the system are extremely sensitive to the initial conditions. Even infinitesimal perturbations grow expenentially in time, making the long-time dynamics impossible to predict. This happens even though the systems is deterministic (not random). The Lyapunov exponent characterizes the rate of separation of infinitesimally close trajectories. Roughly speaking, for a vector norm d, we have $d(\boldsymbol{r}_1(t), \boldsymbol{r}_2(t)) \approx e^{\lambda t} d(\boldsymbol{r}_1(0), \boldsymbol{r}_2(0))$, where $d(\boldsymbol{r}_1(0), \boldsymbol{r}_2(0)) < \epsilon$. Strictly speaking, the rate of separation is governed by a set of local eigenvectors and a spectrum of Lyapunov exponents—one for each dimension of the phase space. See Phys. Rev. A **29**, 2928 (1984).

The equations of motion can be expressed as

$$\frac{d\boldsymbol{r}(t)}{dt} = \boldsymbol{F}(\boldsymbol{r}(t)),$$

where $\boldsymbol{r} = (x, y, z)$ is the coordinate vector. In this lab, the numerical integration is carried out via 2nd order Runge-Kutta

$$\begin{aligned}
\boldsymbol{r}_{i+1} &= \boldsymbol{r}_i + \boldsymbol{c}_2 \\
\boldsymbol{c}_1 &= \frac{1}{2}\boldsymbol{F}(\boldsymbol{r}_i)\Delta t \\
\boldsymbol{c}_2 &= \boldsymbol{F}(\boldsymbol{r}_i + \boldsymbol{c}_1)\Delta t
\end{aligned}$$

and 4th order Runge-Kutta

$$\begin{aligned}
\boldsymbol{r}_{i+1} &= \boldsymbol{r}_i + \frac{1}{6}(\boldsymbol{c}_1 + 2\boldsymbol{c}_2 + 2\boldsymbol{c}_3 + \boldsymbol{c}_4) \\
\boldsymbol{c}_1 &= \boldsymbol{F}(\boldsymbol{r}_i)\Delta t \\
\boldsymbol{c}_2 &= \boldsymbol{F}(\boldsymbol{r}_i + \frac{\boldsymbol{c}_1}{2})\Delta t \\
\boldsymbol{c}_3 &= \boldsymbol{F}(\boldsymbol{r}_i + \frac{\boldsymbol{c}_2}{2})\Delta t \\
\boldsymbol{c}_4 &= \boldsymbol{F}(\boldsymbol{r}_i + \boldsymbol{c}_3)\Delta t
\end{aligned}$$

1. Download the Lab 4 instructions and source code from the class website. You can either do this from a web browser or from the terminal by way of the `get_files` script (described in Lab 0).

```
$ WEBPATH=http://www.phy.olemiss.edu/~kbeach/
$ curl $WEBPATH/courses/fall2017/phys750/src/exercise8.tgz -O
$ tar xzf exercise8.tgz
$ cd exercise78
$ make
g++ -c chaos.cpp -O2
g++ -c viewer.cpp -O2
g++ -o chaos chaos.o viewer.o -lglut
```

2. The `make` command generates an executable `chaos`, which you should run. Right-clicking on the graphics window will bring up a menu from which you can select the action that you want to perform. Moving the mouse while the left button is depressed will allow you to rotate and zoom the image.

3. The code is broken into several files. `viewer.cpp` contains `main` and all of the functions associated with generating the graphics environment. `chaos.cpp` contains all of the functions that initialize and update the dynamical system. `trajectory.hpp` and `vec3.hpp` contain class definitions.

   Most of the work is done by `RungaKutta2nd` and `RungaKutta4th`, the time-step integrators provided. Trajectories evolve using the following class method call:

   ```
   trajectory T(x0,y0,z0);
   T.evolve(Lorenz,RungaKutta4th,dt);
   ```

   You will see that there are in fact seven trajectories running simultaneously. These are stored in a `vector` variable named `bundle`. They correspond to one initial trajectory and $\pm\epsilon$ perturbations in each of the three orthogonal directions.

4. To complete this lab, you shouldn't have to make changes to any of the files except `chaos.cpp`. Almost everything can be done by adding to the functions that initialize and update the system, `initialize_bundle` and `update_bundle`.

5. Experiment with the two integrators and different values of the time step and different values of ρ. Try to get a sense for what size of Δt is appropriate. What kinds of behaviour do you find?

6. Consider the case $\Delta t = 0.01$, $\rho = 28$.

   Modify the code so that after 100 time steps the deviations of the six perturbed trajectories are output every 10th time step to a file `diff.dat`. You can use the construction ofstream `<< distance(bundle[0],bundle[n])`. Have the program exit when the number of time steps exceeds some large value. Use the `exit(0)` function call in the C standard library:

   ```
   #include <cstdlib>
   using std::exit;
   ```

   You should be able to estimate the Lyapunov exponent:

   ```
   $ head -n 5 diff.dat
     0.0328598  0.0334548  0.0260056  0.0263768  0.00482405  0.00482317
     0.0365318  0.0372045  0.0289109  0.0293311  0.00411882  0.00411718
     0.0327259  0.0334689  0.0258911  0.0263555  0.00424087  0.0042419
     0.0282033  0.0287112  0.0223223  0.0226386  0.00528584  0.00528499
     0.0366812  0.0371981  0.0290411  0.0293632  0.00521532  0.00520977
   $ gnuplot
   gnuplot> plot "diff.dat" using 0:(($1+$2+$3+$4+$5+$6)/6) with lines
   gnuplot> set logscale y
   gnuplot> dt = 0.01
   gnuplot> eps = 0.01
   gnuplot> replot eps*exp(1.2*(x+55)*dt)
   ```

7. For the case $\Delta t = 0.01$, $\rho = 22$, show that the Lyapunov exponent is negative.

8. Consider the case $\Delta t = 0.001$, $\rho = 160$.

   Modify the code so that after 100 time steps the position of the non-perturbed trajectory is output every 10th time step to a file `traj.dat`. You can use the construction *ofstream* << `bundle[0]`.

   Convince yourself that the period of $115.3 \times \Delta t = 0.1153$ can be identified as follows:

```
$ head -n 5 traj.dat
  2.6970908   9.7907308  0.92401019
  3.508246   12.729297   1.242895
  4.5620529  16.534085   1.7902344
  5.9289753  21.442572   2.7218089
  7.6975195  27.73536    4.2966881
$ gnuplot
gnuplot> plot "traj.dat" using 0:1 with lines   # plot x vs. time step
                                                # behaviour is regular after
                                                # initial tranient has dies out
gnuplot> plot "traj.dat" using 1:0 with lines, 353, 465  # roughly identify period
gnuplot> print 465-353
111
gnuplot> plot "traj.dat" using (sin(2*pi*$0/115.3)):1 with lines  # data collapse
```

9. Verify that period doubling is the route to chaos as ρ is decreased.

ρ	period
160	$T = 0.1153$
155	$T = 0.1172$
150	$T = 0.1192$
149	$T = 0.1197$
148	$2T = 0.1201$ $T = 0.0605$
144	aperiodic

**10. If you are particularly ambitious, you could try to estimate the driving period as a function of ρ and then create a Poincaré section plot of x versus ρ.

```
$ cat > periods.dat
160 115.3
155 117.2
150 119.2
149 119.7
148 120.1
[ctrl-d]
$ gnuplot
gnuplot> f(x) = f0 + f1*x + f2*x**2
gnuplot> fit f(x) "periods.dat" via f0,f1,f2
gnuplot> plot "periods.dat", f(x)
gnuplot> print f(147.5)
120.323668111517
```

For example, at $\rho = 147.5$ you would plot the value of x that appears every 120.3 times steps. The windows of period $2, 4, 8, \ldots$ decrease in width by the same factor—the Feigenbaum δ. Can you estimate its value?