4-16-1994

The Runge-Kutta Method and Systems of Differential Equations.

This project is due by 3 p.m., Friday, April 29.

The **Runge-Kutta method** is a numerical method for integrating differential equations. There is a package which implements Runge-Kutta; here's how to load it:

```
<< programm\rungekut.m
```

Here's how to get a description of `RungeKutta[]`:

```
?RungeKutta

      RungeKutta[{ e1,e2,..}, {y1,y2,..}, {a1,a2,..}, {t1, dt}]
         numerically integrates the ei as functions of the yi with
         inital values ai.  The integration proceeds in steps of dt
         from 0 to t1.  RungeKutta[{e1,e2,..}, {y1,y2,..}, {a1,a2,..},
         {t, t0, t1, dt}] integrates a time-dependent system from t0
         to t1.
```

I'm not going to discuss the Runge-Kutta method itself. Instead, I'm going to show you how to use it to approximate solution curves to systems — that is, as a tool for exploring properties of systems of differential equations.

For example, here is how to use Runge-Kutta to approximate the solution to the **Lorenz system**

$$x' = -3(x-y), \quad y' = -xz + 26.5x - y, \quad z' = xy - z$$

satisfying the initial condition $(x(0), y(0), z(0)) = (0, 1, 0)$. I'm plotting points from $t = 0$ to $t = 20$ in increments of 0.04. (The semicolon at the end suppresses the display of the output, since I don't really want to look at a list of 500 points!)

```
lorenz = RungeKutta[{-3(x - y), - x z + 26.5x - y, x y - z},
    {x, y, z}, {0.0, 1.0, 0.0}, {20, 0.04}];
```

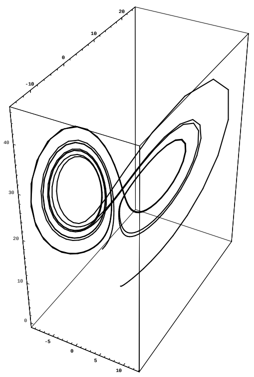
Since $(x(20), y(20), z(20))$ is the 501-st element of the list, I can display it using

```
lorenz[[501]]

    {-1.31165, -3.21823, 12.0889}
```

Next, I connect the points with segments to make a continuous curve and display the result. This is an approximation to the solution to the initial value problem.

```
Show[Graphics3D[{Line[lorenz]}], Axes->Automatic]
```



Problem 1.

(a) Use `RungeKutta[]` to approximate $(x(20), y(20), z(20))$ for the initial conditions $(x, y, z) = (0.0, 0.99, 0.0)$, using steps of size 0.04. Call your list of points `newlorenz`, and write down the value of $(x(20), y(20), z(20))$. (Be sure when you type this in that you've left spaces in `x y` and `x z`!)

Notice that the second set of initial conditions was very close to the first, but that the final values were rather far apart. This is called "sensitive dependence on initial conditions".

(b) Plot the solution curve determined by `newlorenz`. Does it look like the picture above?

The **Brusselator** is a model chemical reaction studied by Ilya Prigogine and R. Lefever [1]. x and y denote the amounts of two reactants; the rates at which the reactants increase or decrease satisfy the system

$$x' = a - bx - x + x^2y, \quad y' = bx - x^2y.$$

I'll consider this system for a = 1, b = 3.

Problem 2.

(a) Use `RungeKutta[]` to approximate $(x(50), y(50))$ for the initial conditions $(x, y) = (1.0, 1.0)$, using steps of size 0.1. (Call your list of points `brussel1`. $(x(50, y(50))$ will be `brussel1[[501]]`.)

(b) Plot the solution curve determined by `brussel1`. (You will need to change the `Graphics3D` in the commands for Problem 1 to `Graphics`.)

(c) Use `RungeKutta[]` to approximate $(x(50), y(50))$ for the initial conditions $(x, y) = (2.0, 2.0)$, using steps of size 0.1. (Call your list of points `brussel2`. $(x(50, y(50))$ will be `brussel2[[501]]`.)

(d) Plot the solution curve determined by `brussel2`. (Don't forget to change the `Graphics3D` in the commands for Problem 1 to `Graphics`.)

(e) For this system, solution curves are attracted by tilted oval, in the sense that they wind up running around the oval no matter what initial conditions you use.

Describe what is happening to the amounts x and y of the reactants as a solution curve goes round and round the oval.

The **Van der Pol system** is

$$x' = y, \quad y' = -x + a(1 - x^2)y.$$

I will take a = 0.1.

Problem 3.

(a) Use `RungeKutta[]` to approximate $(x(50), y(50))$ for the initial conditions $(x, y) = (1.0, 1.0)$, using steps of size 0.1. (Call your list of points `vander1`.) Then plot the solution curve determined by `vander1`.

(b) Use `RungeKutta[]` to approximate $(x(50), y(50))$ for the initial conditions $(x, y) = (2.0, 2.0)$, using steps of size 0.1. (Call your list of points `vander2`.) Then plot the solution curve determined by `vander2`.

Like the Brusselator, the Van der Pol system has a periodic solution curve which other solution curves approach as t goes to infinity. Such a solution curve is called a limit cycle. Physically, it means that the system settles down to oscillatory behavior.

---

[1] Ilya Prigogine and R. Lefever, Symmetry breaking instabilities in dissipative systems, II, *Journal of Chemical Physics,* 48(1968), 1695-1700.

