MA 512 **Homework 4** **February 16, 2017**

The purpose of this assignment is to show how it is sometimes necessary to use tight error tolerances in order get the solution right, if only for awhile, and also to demonstrate some very interesting ODE solution behavior.

1. (20 points) The oscillation of a pendulum is modeled by

$$\theta'' + \sin(\theta) = 0, \qquad (\star)$$

where θ is the angle of the pendulum from the vertical.

a. Show that the *energy* $E(t) \equiv \theta'(t)^2/2 - \cos(\theta(t))$ of a solution $\theta(t)$ of $(\star)$ is constant, i.e., energy is conserved by solutions of $(\star)$. *Hint: Differentiate* $E(t)$.

b. Suppose that $\theta(t)$ is a solution of $(\star)$ such that $\theta(0) = 0$, i.e., the pendulum is vertical downward at time $t = 0$, and that $\theta'(0) = \theta_0' > 0$. If θ_0' is not too large, then the pendulum will swing back and forth, with $\theta(t)$ oscillating between positive and negative values. Note that $\theta'(t) = 0$ whenever $\theta(t)$ reaches a positive maximum. As θ_0' increases, it will reach a critical value at which the positive maximum of $\theta(t)$ is π, i.e., the pendulum is vertical upward. At this maximum, we have that $\theta'(t) = 0$ and, from $(\star)$, that $\theta''(t) = 0$ as well. It follows that the pendulum will remain stationary in this position for all future time, i.e., the solution has reached a steady state. Use conservation of energy to show that the critical value is $\theta_0' = 2$. *Hint: Evaluate* $E(t)$ *at the maximum and set* $E(t) = E(0)$.

c. Use MATLAB's `ode45` to approximately solve $(\star)$ with $\theta(0) = 0$ and $\theta'(0) = 2$. In calling `ode45`, take $[\mathtt{t_0}, \mathtt{t_f}] = [0, 20]$ and first use the default tolerances. Plot the output of `ode45` using `plot(tvals,yvals)`, where `yvals(:,1)` and `yvals(:,2)` are the output values θ and θ', respectively. (The results should look far from steady.) Print out and hand in the plot. Now repeat all this using tighter tolerances `RelTol = AbsTol = 1e-9`. (The results should now look steady.) Print out and hand in the plot in this case. Have you really satisfactorily computed the steady-state solution? For your own edification (no need to hand anything else in), increase t_f to a larger value, say $t_f = 100$, and try different tolerance values, both looser and tighter.

d. By now, you probably realize that the solution $\theta(t) \equiv \pi$ is *unstable*, i.e., even very small perturbations give rise to solutions with very different behavior. (Of course, the corresponding vertical-upward configuration of the pendulum is physically unstable.) To illustrate this instability, use `ode45` to compute three solutions of ($\star$) over $[\mathtt{t_0}, \mathtt{t_f}] = [0, 20]$ with $\theta(0) = 0$ and $\theta'(0) = 1.999$, 2, and 2.001. Use tight error tolerances `RelTol = AbsTol = 1e-12` to make sure you are computing the correct solution behavior over $0 \le t \le 20$. Plot the three solutions (the θ-values only) versus time on the same figure. (The `hold` command may be useful for this.) Print out and hand in the plot.

2. (10 points) The *Lorenz equation*

$$\begin{aligned} x' &= -ax + ay \\ y' &= rx - y - xz \\ z' &= -bz + xy \end{aligned} \tag{$\star\star$}$$

was introduced in 1963 by the meteorologist and mathematician E. N. Lorenz as a simplified model for atmospheric turbulence beneath a thunderhead. It has since become famous as one of the motivating examples for the mathematical study of chaos.

With $a = 10$, $b = 8/3$, and $r = 28$, use `ode45` to numerically solve ($\star\star$) with $[\mathtt{t_0}, \mathtt{t_f}] = [0, 20]$, first with initial values $x(0) = y(0) = z(0) = 1$ and then with $x(0) = 1.1$, $y(0) = z(0) = 1$. Use fairly tight error tolerances, say `RelTol = AbsTol = 1e-8`, to make sure you are computing the solutions correctly. Use MATLAB's `plot3` command to produce a figure with three-dimensional phase-space plots of the solution components in the two cases. Using the "Rotate 3D" icon in the figure toolbar, rotate this plot until it nicely illustrates the "Lorenz butterfly." Note how the two solutions diverge, in particular how they go from one "wing" to the other at different times. Print out the rotated plot and hand it in. For your own amusement (don't hand anything in), type `lorenz` in the MATLAB command window and run the Lorenz-attractor demo.