1 Exercise: Solving ODEs – Lorenz equations

Reading

- *Spiegelman* (2004), chap. 4
- *Press et al.* (1993), chap. 16
- *Spencer and Ware* (2008), sec. 16

We previously discussed the 4^{th} order Runge Kutta method as a simple method to solve initial value problems where the task is to forward integrate a vector $\underline{y}(t)$ from an initial condition $\underline{y}_0(t = t_0)$ to some time t_f while the time derivatives of $\underline{y}$ are given by

$$\frac{d\underline{y}}{dt}(t) = \underline{f}(t, \underline{y}, \underline{C}) \tag{1}$$

we made the dependence of f on constant parameters explicit in the $\underline{C}$.

Numerically, this is done by successively computing $\underline{y}_{n+1}$ for time $t + h$ from the last known solution for $\underline{y}_n$ at time t with time step h

$$\underline{y}_{n+1} \approx \underline{y}_n + h\underline{y}'(h, t, \underline{y}, \underline{C}) \tag{2}$$

where $\underline{y}'$ denotes the approximate time-derivatives for $\underline{y}$.

In a real application, we would use adaptive step-size control by means of error checking depending on the accuracy of our approximate method, or employ an entirely different approach (*Press et al.*, 1993, sec. 16.2). *Spencer and Ware* (2008) discuss some of the algorithms that are implemented in MATLAB , and the problem set file `rikitake.m`[1] is an example for how to use the MATLAB function `ode45`. However, the Runge-Kutta is good example method and easy enough to implement.

1.1 The Lorenz equations solved with simple Runge Kutta

As an interesting example of a three-dimensional $\left(\underline{y} = \{y_1, y_2, y_3\}\right)$ ODE system, we will discuss the classic *Lorenz* (1963) equations. These equations are a simplified description of thermal convection in the atmosphere and an example of a low order, spectral numerical solution. They were also fundamental in the establishment of deterministic chaos theory.

[1] All MATLAB files for all of the problem sets are at `http://geodynamics.usc.edu/~becker/teaching-557.html`. Some problem sets, including this ODE one also have Python implementations. Solved MATLAB scripts for the problem sets are available for instructors upon request.

1.2 What exactly are these equations modeling?

For an incompressible, Newtonian fluid, conservation of mass, energy, and momentum for the convection problem can be written as

$$\underline{\nabla} \cdot \underline{v} = 0 \tag{3}$$

$$\frac{\partial T}{\partial t} + \underline{v} \cdot \underline{\nabla} T = \kappa \nabla^2 T \tag{4}$$

$$\frac{\partial \underline{v}}{\partial t} + (\underline{v} \cdot \underline{\nabla})\, \underline{v} = \nu \nabla^2 \underline{v} - \frac{1}{\rho_0} \underline{\nabla} P + \frac{\rho}{\rho_0} \underline{g}. \tag{5}$$

Here, $\nu = \eta/\rho_0$ is dynamic viscosity, $\underline{v}$ velocity, T temperature, κ thermal diffusivity, $\underline{g}$ gravitational acceleration, ρ density, and P pressure. In the Boussinesq approximation, $\rho(T) = \rho_0(1 - \alpha(T - T_0))$, where α is thermal expansivity and ρ_0 and T_0 reference density and temperature, respectively.

If we assume two-dimensionality (2-D) in x and z direction, and a bottom-heated box of fluid, the box height d provides a typical length scale. If $\underline{g}$ only acts in z direction and all quantities are non-dimensionalized by chosing length scale d, as time scale the diffusion time, d^2/κ, and the temperature contrast between top and bottom ΔT, we can write

$$\frac{\partial T'}{\partial t'} + \underline{v}' \cdot \underline{\nabla} T' = \nabla^2 T' \tag{6}$$

$$\frac{1}{Pr}\left(\frac{\partial \omega}{\partial t'} + \underline{v}' \cdot \underline{\nabla} \omega\right) = \nabla^2 \omega - Ra \frac{\partial T'}{\partial x'}, \tag{7}$$

where the primed quantities are now non-dimensionalized. If chosen right, such a transformation makes the individual terms of the equations that depend on the primed variables of order unity, such that the material parameter dependent factors (here Ra and Pr) measure the importance of the terms during solution (see sec. **??**).

Eq. (7) is eq. (5) rewritten in terms of vorticity ω, which is defined as

$$\underline{\omega} = \nabla \times \underline{v} \tag{8}$$

and for 2-D with $\underline{v} = \{u, w\}$ is the scalar quantity

$$\omega = \frac{\partial w}{\partial x} - \frac{\partial u}{\partial z}. \tag{9}$$

Using the stream-function ψ, $\nabla^2 \psi = -\omega$, which relates to velocity as

$$\underline{v} = \{u, w\} = \{\partial\psi/\partial z, -\partial\psi/\partial x\} \tag{10}$$

and enforces incompressibility (mass conservation), see sec. **??**.

The important part here are the two new non-dimensional quantities that arise, the Prandtl and the Rayleigh numbers, which were discussed previously (eqs. **??** and **??**).

Lorenz (1963) used a very low order spectral expansion to solve the convection equations. He assumed that

$$\psi \approx W(t)\sin(\pi a x)\sin(\pi z) \tag{11}$$
$$T \approx (1-z) + T_1(t)\cos(\pi a x)\sin(\pi z) + T_2(t)\sin(2\pi z) \tag{12}$$

for convective cells with wavelength $2/a$. This is an example of a spectral method where spatial variations in properties such as T are dealt with in the frequency domain, here with one harmonic. Such an analysis is also common when examining barely supercritical convective instabilities.

1.3 Problems

The resulting equations for the time dependent parameters of the approximate Lorenz convection equations are

$$\begin{aligned} \frac{dW}{dt} &= Pr(T_1 - W) \\ \frac{dT_1}{dt} &= -WT_2 + rW - T_1 \\ \frac{dT_2}{dt} &= WT_1 - bT_2 \end{aligned} \tag{13}$$

where $b = 4/(1+a^2)$, $r = Ra/Ra_c$ with the critical Rayleigh number Ra_c.

a) Identify W, T_1, and T_2 as y_1, y_2, y_3 and write up a MATLAB code for a 4^{th} order Runge Kutta scheme to solve for the time-evolution of $\underline{y}$ using eq. (13) for derivatives.

Hint: You can code this any way you want, but consider the following (Figure 2):

- You will want to separate a "driver" that deals with initial conditions, controlling the total time steps, plotting, etc., and an actual routine that computes the Runge Kutta step following the formula we discussed in class. Those should be separate m-files, or at least separate functions.
- You will want to make the Runge Kutta stepper independent of the actual function that is needed to compute dy/dt so that you can reuse it for other problems later. This can be done in MATLAB by defining a function `myfunc` that computes the derivatives, and then passing the function name `myfunc` as an argument to the Runge Kutta time stepper. Within the time stepper, the function then then has to be referred to as
`@func`. Alternatively, the function that computes the derivatives can be made into its own ".m" file, in the same directory as the other subroutines, making it available to all subroutines in that folder.

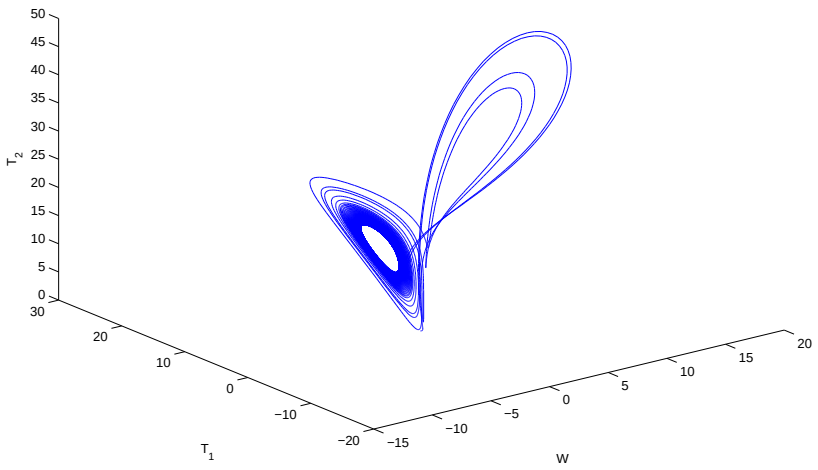

Figure 1: Solution to one of the problem set questions visualizing the behavior of the Lorenz equations (the Lorenz attractor).

- If you need some inspiration on how to do this, download the m-file fragments we provide for this sections problem set, `lorenz_dy.m`, `lorenz.m`, and `rkstep.m`. There is also a complete Python implementation, `ode.py` and `lorenz.py`, if you are curious.

b) Use initial condition $\underline{y}_0 = \{0, 0.5, 0.5\}$, parameters $b = 8/3$, $Pr = 10$ and solve for time evolution for all three variables from $t = 0$ to $t = 50$, using a time step $h = 0.005$. Use $r = 2$ and plot T_1 and T_2 against time. Comment on the temporal character of the solution, what does it correspond to physically?

c) Change r to 10, and then 24. Plot T_1 and T_2 against time, and also plot the "phase space trajectory" of the system by plotting $\underline{y}$ in W, T_1, and T_2 space using MATLAB `plot3`. Comment on how these solutions differ.

d) Increase r to 25 and plot both time behavior of T and the phase space trajectory. What happened? Compare the $r = 25$ solution with the $r = 24$ solution from the last question. Do you think $r = 24$ will remain steady for all times? Why? Why not?

e) Use $r = 30$ and show on one plot how T_1 evolves with time for two different initial conditions, the $\underline{y}_0$ from before, $\{0, 0.5, 0.5\}$, and a second initial condition $\{0, 0.5, 0.50001\}$. Comment.

f) Compare your solution with $h = 0.005$ for T_1 and an initial condition of your choice

in the $r = 30$ regime with the MATLAB -internal ODE solver you deem most appropriate. Plot the absolute difference of the solutions against time. Comment.

For help with making simple plots with MATLAB , see *Spencer and Ware* (2008), for example. It is very easy to get such plots while developing code and debugging, but often hard to generate publication quality results from MATLAB.How to do this is discussed in numerous online resources, including some helpful routines at `http://geoweb.princeton.edu/people/simons/software.html`, for example. While it is generally preferred to remain within one framework, you might want to consider plotting MATLAB generated results with external graphics packages such as Gnuplot[2] (mainly for x-y type plots, but some great extra capability) or GMT[3] (a very versatile plotting software that does, however, often require scripting).

1.4 Additional examples

a) If you are curious about additional Earth Science applications of ODEs, the literature of geochemical modeling is full of it because it is often easiest, or most appropriate, to consider fluxes between reservoirs of different chemical species with averages properties, so-called "box models" (*e.g.* *Albarede*, 1995).

b) A classic example from magneto-hydrodynamics is the 3-D Rikitake dynamo model that consists of two conducting, rotating disks, coupling by smart cross wiring, in a background magnetic field. The Rikitake dynamo shows behavior similar to the Lorenz system and serves as an analog for the geodynamo, exhibiting irregular magnetic field reversal. The equations are

$$\begin{aligned} \frac{dx}{dt} &= -mx + yz \\ \frac{dy}{dt} &= -my + (z - a)x \\ \frac{dz}{dt} &= 1 - xy \end{aligned} \tag{14}$$

with typical parameters for a of 4 or 10, $m = 2$, at initial conditions $x, y, z = -5, 2, 2$. The file `rikitake.m` provides an example implementation of these equations using a MATLAB ODE solver.

c) Examples from our own research where we have used simple ODE solutions, include some work on parameterized convection (*Loyd et al.*, 2007), a method that goes back at least to *Schubert et al.* (1980), see *Christensen* (1985). In this case, the box is the mantle, and the total heat content of the mantle, as parameterized by the

[2] `http://www.gnuplot.info/`

[3] *Wessel and Smith* (1998), `http://gmt.soest.hawaii.edu/`, also see *Becker and Braun* (1998), `http://geodynamics.usc.edu/~becker/igmt`

```
function lorenz % this is only a function to allow function declarations
%
% Lorent'z equation solver
% the ... parts will have to be filled in by you
%
% values to solve for
%
% y(1) : W
% y(2) : T1
% y(3) : T2

% parameters for the equations
parameters.r = ...; % Rayleigh number
parameters.Pr = ..; % Prandtl number

% initial values
y= [...];
time =0;tstop=50;
h = 0.005;% timestep
save_each = 1;
nstep=0;save_step=0;
while(time < tstop) % loop while time is smaller than tstop
    if(mod(nstep,save_each)==0) % only save every save_each step
        save_step=save_step+1;
        ysave(save_step,:)=y;
...
    end
    % advance the y(1:3) solution by one 4th order Runge Kutta step
    y =  y + rkstep(....);

    nstep=nstep+1;
    time=time+h;
end

figure(1);clf % time series
plot(tsave,ysave(:,2))
xlabel('time');ylabel('temperature');
legend('T_1','T_2')



function dy = rkstep(.... )
%
%
% perform one 4th order Runge Kutta timestep and return
% the increment on y(t_n) by evaluating func(time,y,parameters)
%
% ... parts need to be filled in
%
%
% input values:
%  h: time step
%  t: time
%  y: vector with variables at time = t which are to be advanced
%  func: function which computes dy/dt
%  parameters: structure with any parameters the func function might need

% save computations
h2=h/2;

k1 = h .* dydt(...);
k2 = h .* dydt(...);
....
% return the y_{n+1} timestep
dy =  ....
```

Figure 2: Suggested program structure for the Lorenz equation ODE solver exercise. Available online as `lorenz.m`, `rkstep.m`, and `dydt.m`.

mean temperature, is the property one solves for. I.e. we are averaging the PDEs governing convection spatially, to solve for the time-evolution of average mantle temperature.

The idea is that a convective system with Rayleigh number (see eq. (**??**))

$$Ra = \frac{\rho \alpha T g h^3}{\kappa \eta} \tag{15}$$

transports heat at Nusselt number Nu following a scaling of

$$Nu = \frac{Q}{cT} = aRa^{\beta} \tag{16}$$

(with some debate about β, see, *e.g.* Korenaga, 2008, for a review). The energy balance for the mantle is

$$C_p \frac{dT}{dt} = H(t) - Q \tag{17}$$

where C_p is the total heat capacity, H the time-dependent heat production through radiogenic elements, and Q the heat loss through the surface. If viscosity is a function of temperature,

$$\eta = \eta_0 \exp\left(\frac{H}{RT}\right) \tag{18}$$

then the equations couple such that

$$C_p \frac{dT}{dt} = H(t) - Q_0 \left(\frac{T}{T_0}\right)^{1+\beta} \left(\frac{\eta(T_0)}{\eta(T)}\right)^{\beta}. \tag{19}$$

We provide an example, `thermal_all.m` online. You might want to experiment with the shooting method to explore feasible and unfeasable paths of Earth's thermal evolution from an initial to a final temperature.

d) Another example, from the brittle regime, are spring sliders. Instead of dealing with full fault dynamics, one may consider a block that has a friction law apply at its base and pulled by a string. Depending on the assumptions on the friction law, such a system exhibits stick-slip behavior akin to the earthquake cycle. For rate-and-state (*i.e.* velocity and heal-time) dependent friction (*e.g.* Marone, 1998) with two "state" variables, spring-slider models exhibit interesting, chaotic behavior (Becker, 2000).

The equations are

$$\begin{aligned} \dot{x} = \frac{dx}{dt} &= e^x((\beta_1 - 1)x + y - z) + \dot{y} - \dot{z} \\ \dot{y} = \frac{dy}{dt} &= (1 - e^x)\kappa \\ \dot{z} = \frac{dz}{dt} &= -e^x\rho(\beta_2 x + z) \end{aligned} \tag{20}$$

with $\beta_1 = 1$, $\beta_2 = 0.84$, and $\rho = 0.048$ (Gu *et al.*, 1984).

This behavior includes the characteristic period-doubling route toward chaos (Feigenbaum, 1978) as a function of a material parameter (spring stiffness), at critical $\kappa = 0.08028$ (Becker, 2000). You might want to reproduce the bifurcation plot of Figure 3.

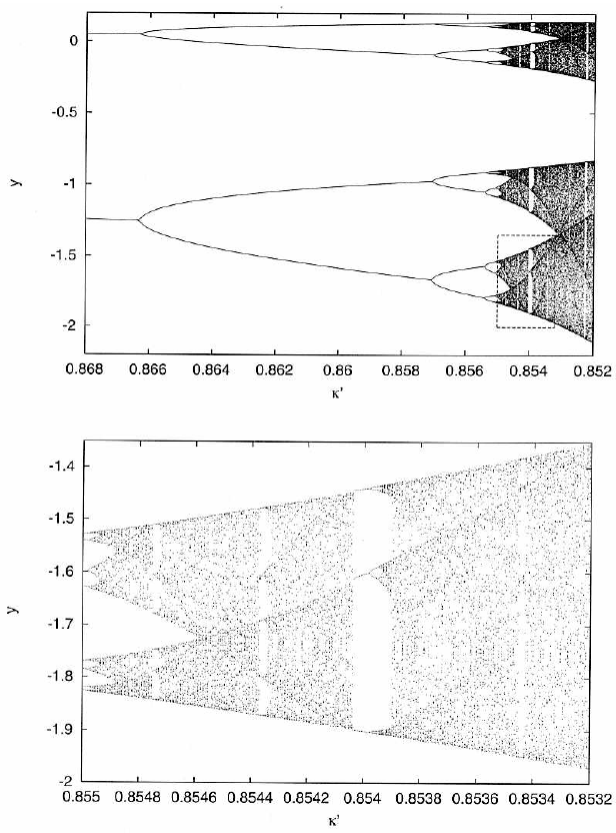

Figure 3: Poincare sections in y for the period doubling sequence to chaos for the spring-slider system, eq. (20), as a function of normalized spring stiffness, κ'. Bottom figure shows zoom into the dashed rectangular region highlighted on top (modified from *Becker*, 2000).

Bibliography

Albarede, F. (1995), *Introduction to geochemical modeling*, Cambridge University Press.

Becker, T. W. (2000), Deterministic chaos in two state-variable friction sliders and the effect of elastic interactions, in *GeoComplexity and the physics of earthquakes*, *Geophys. Monograph*, vol. 120, edited by J. B. Rundle, D. L. Turcotte, and W. Klein, pp. 5–26, American Geophysical Union, Washington, DC.

Becker, T. W., and A. Braun (1998), New program maps geoscientific data sets interactively, *Eos Trans. AGU*, *79*, 505.

Christensen, U. R. (1985), Thermal evolution models for the Earth, *J. Geophys. Res.*, *90*, 2995–3007.

Feigenbaum, M. J. (1978), Quantitative universality for a class of nonlinear transformations, *J. Stat. Phys.*, *19*, 25.

Gu, J.-C., J. R. Rice, A. L. Ruina, and S. T. Tse (1984), Slip motion and stability of a single degree of freedom elastic system with rate and state dependent friction, *J. Mech. Phys. Solids*, *32*, 167–196.

Korenaga, J. (2008), Urey ratio and the structure and evolution of Earth's mantle, *Rev. Geophys.*, *46*, doi:10.1029/2007RG000241.

Lorenz, E. N. (1963), Deterministic nonperiodic flow, *J. Atmos. Sci.*, *20*, 130.

Loyd, S. J., T. W. Becker, C. P. Conrad, C. Lithgow-Bertelloni, and F. A. Corsetti (2007), Time-variability in Cenozoic reconstructions of mantle heat flow: plate tectonic cycles and implications for Earth's thermal evolution, *Proc. Nat. Acad. Sci.*, *104*, 14,266–14,271.

Marone, C. (1998), Laboratory-derived friction laws and their application to seismic faulting, *Annu. Rev. Earth Planet. Sci.*, *26*, 643–696.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery (1993), *Numerical Recipes in C: The Art of Scientific Computing*, 2 ed., Cambridge University Press, Cambridge.

Schubert, G., D. Stevenson, and P. Cassen (1980), Whole planet cooling and the radiogenic heat source contents of the Earth and Moon, *J. Geophys. Res.*, *85*, 2531–2538.

Spencer, R. L., and M. Ware (2008), *Introduction to Matlab*, Brigham Young University, available online, accessed 07/2008.

Spiegelman, M. (2004), *Myths and Methods in Modeling*, Columbia University Course Lecture Notes, available online at http://www.ldeo.columbia.edu/~mspieg/mmm/course.pdf, accessed 06/2006.

Wessel, P., and W. H. F. Smith (1998), New, improved version of the Generic Mapping Tools released, *Eos Trans. AGU*, *79*, 579.