Autumn Quarter 2005
Math. Methods Problem Set 6

November 22, 2005

1 Numerical integration in 1D: Planetary temperature with ice-albedo feedback

The temperature of one hemisphere of a planet forced by seasonally varying solar radiation can be described by the equation

$$\frac{dT}{dt} = (1-\alpha(T))S_o \cdot \frac{1}{2}(1+\cos\omega t) - T^4 \tag{1}$$

where the reflectivity α depends on temperature because the ice coverage changes as the hemisphere gets warmer or colder. We can take

$$\alpha(T) = \begin{cases} \alpha_i & \text{for } T \leq T_i, \\ \alpha_o + (\alpha_i - \alpha_o)\frac{(T-T_o)^2}{(T_i-T_o)^2} & \text{for } T_i < T < T_o \\ \alpha_o & \text{for } T \geq T_o \end{cases} \tag{2}$$

Write a Python script to integrate the temperature equation. You can take $\alpha_i = .8$,$\alpha_o = 0$, $T_i = 260.$ and $T_o = 290.$. Try different values of S_o and ω to get a feel for the range of behavior of the system. Use your imagination, and also your mathematical analysis abilities, to help you figure which values of these parameters give interesting results.

2 Numerical integration in 2D: Predator-prey

In a previous problem set, you studied the equilibrium points of the predator-prey system

$$\frac{dr}{dt} = g_r \cdot (1 - \frac{r}{C})r - e \cdot r \cdot w; \frac{dw}{dt} = g_w \cdot r \cdot w - d \cdot w, \qquad (3)$$

and their stability.

Write a Python script to numerically integrate this system. There are three equilibrium points: one with $r = w = 0$, one with $r = C$ and $w = 0$, and one in which both r and w are nonzero. Pick a set of parameters for which the third equilibrium is stable, and say what happens if you initialize the model near one of the two unstable equilibria. Show what happens when you initialize the model near the stable equilibrium. Illustrate the behavior both for cases where the approach to equilibrium has an oscillatory character and a non-oscillatory character.

Can you make the system execute a periodic oscillation that doesn't approach equilibrium? (Hint: Try making C essentially infinite (e.g. 1e90), so the system becomes the classic Lotka-Volterra system). Does the oscillation depend on where you start?

3 Using Numeric Arrays:I

(a) Make a Numeric array of 1000 equally spaced values between 0 and 1. Use array arithmetic to compute the result of the first 100 iterates of these values under the map $x_{n+1} = 3.8x_n(1 - x_n)$

(b) Compute the sum of the first 10000 values of $1/n^2$ by making a Numeric array `A` containing these values, and using `Numeric.sum(A)`.

4 Using Numeric Arrays:II

(a) Make a 500x500 two-dimensional floating point Numeric array whose value in the (i, j) place is $i^2 + j^2$. You can do this in a loop, or use `Numeric.fromfunction(...)`. Use `Numeric.matrixmultiply(...)` to compute the matrix product of this matrix with itself. Don't print out the result! Just show me the values in the upper left 10x10 block.

(b) Write a function that computes the average of the elements in each row of a 2D matrix, and returns a 1D vector of the averages. Use `Numeric.sum(...)` and array cross-section references to do this. *Hint:* The i^th row of the matrix `A` can be referenced as `A[i,:]` or `A[i]`. Whether you want to call this a "row" or a "column" is a matter of convention, which is somewhat different between mathematics and various programming languages.