WHOI Math Review Projects

Problems written by Dan Amrhein, Jaap Nienhuis, Neesha Schnepf

July 30, 2015

Contents

1 Identifying leap years

There is a leap year every year whose number is perfectly divisible by four - except for years which are both divisible by 100 and not divisible by 400. The name "leap" year comes from the fact that while a fixed date in the Gregorian calendar normally advances one day of the week from one year to the next, in a leap year the day of the week will advance two days (from March onwards) due to the year's extra day inserted at the end of February (thus "leaping over" one of the days in the week). For example, Christmas Day fell on Sunday in 2005, Monday in 2006 and Tuesday in 2007 but then "leapt" over Wednesday to fall on a Thursday in 2008.

Tasks:

1 Make a MATLAB function called leapyear.m which tells you if a given year is a leap year or not. For instance, if you call leapyear(2012) it should say: "this is a leap year", if you input leapyear(2013) it should say: "this is not a leap year."

Extra:

1. Make MATLAB display the year: "*year* is a leap year"

2. Find out which day was leapt, i.e. which day of the week is skipped as Christmas day that year. Make MATLAB display: "*year* is a leap year, and *day of week* was skipped."

2 Reproduce the logistic map

The logistic series is an archetypal example of how complex, chaotic behaviour can arise from very simple non-linear dynamical equations. The series was popularized in a seminal 1976 paper by the biologist Robert May, in part as a discrete-time demographic model analogous to the logistic equation first created by Pierre François Verhulst. It is defined as

$$\begin{aligned} x_{n+1} &= r \cdot x_n \left(1 - x_n\right) \\ n &= 1, 2, \ldots \\ & 0 < x_0 < 1 \end{aligned}$$

While this series quickly reaches an equilibrium for r=2.8, choosing r=3.8 results in chaos, a series that will never converge. The logistic map below (Figure 1) shows the "equilibrium" values.

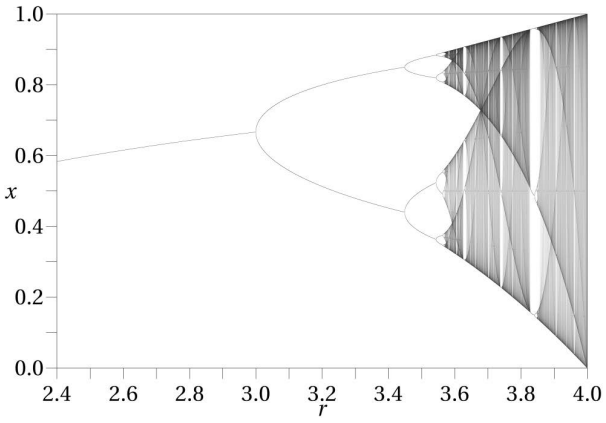

Figure 1: Logistic map (Exercise 2)

Reproduce this plot using a script built in MATLAB. Plot for different values of r, all the values for x between $n = 1000$ and $n = 1050$. NB: If you're having trouble plotting this as a line, a scatterplot with suffice too.

3 Rock-paper-scissors-lizard-Spock

As explained by Sheldon Cooper, in the game rock-paper-scissors-lizard-Spock if a player throws the same item, it's a tie, otherwise, rock beats scissors, paper beats rock, scissors beats paper, rock beats lizard, lizard beats Spock, Spock beats scissors, scissors beats lizard, lizard beats paper, paper beats Spock, and Spock beats rock. Create this game on MATLAB where a person can play against the computer. To do this, start your script with a prompt asking for the user to 'Please pick your throw: (1-Rock, 2-Scissors, 3-Paper,'4- Lizard, 5-Spock)' and saving what is inputted as a variable to be compared with the computer's random pick (which you must declare in your script). After the person

and computer have each picked a throw the script should display what the computer threw and whether or not the person won or lost.

HINT: there is a way to do this without using 'if' statements!

4 The Monty Hall Problem

In a TV game show, the host Monty Hall lets a contestant choose one of three doors to try to win a prize. Behind two of the doors are goats and behind the third is a luxury sports car. After selecting a door, Monty would then proceed to open one of the doors not selected. It is important to note here that Monty would NOT open the door that concealed the car. At this point, he would then ask you if you wanted to switch to the other door before revealing what you had won.

Tasks:

1. Simulate this game hundreds of times on MATLAB to determine what is the best way to win the luxury sports car.

2. Plot the results using a method of your choosing.

Optional:

3. Simulate this game for the pretend case where Monty does not know which door hides the car. If Monty should happen to open the door with the car behind by accident then the game is over and the next contestant plays the game. Plot the results in a method of your choosing.

If you want further clarification or hints, check
http://www.math.ucsd.edu/∼crypto/Monty/monty.html

5 Downloading and plotting NCEP-NCAR reanalysis product

NCEP-NCAR reanalysis provides decades-long estimates of dynamical quantities in Earth's atmosphere and is used widely in climate science. The product is generated by "reanalyzing" observations of air temperature, pressure, and wind (e.g. from weather balloons) using numerical weather prediction models.

For this project you will download one year of surface air temperature (SAT) as a netcdf file (a common data type for climate model output), load it into MATLAB, and plot it as single frames and as a yearly movie.

Steps:

1. Download one year of SAT.

Link to reanalysis daily average surface temperature from NOAA.
Under "Link to files" click "See list." Pick your favorite year and save it in a directory called "ncep."

2. *In the same directory, write a script that loads one day of data and plots the global field of SAT.* Include a title, colorbar, and x- and y-labels. Functions you will find useful (read help files for more information):

ncdisp - shows information about variables stored in the netcdf file

ncread - read in particular variables

Optional tasks:

3. *Write a for loop that plays a "movie" of images in MATLAB so you can see the field changing in time.* The drawnow funciton may be useful. If you want to keep the colorbar limits fixed, try using caxis() and think about how you can find a range of values that applies for the entire year.

4. *Save the "movie" as a video file.* Start with 'help videowriter.'

5. *Do it again for another variable.* Download another reanalysis variable and use subplot to plot snapshots of SAT and the new variable in the same figure. Make a movie of these two variables. Do they appear to covary? Why might they?

6 Comparing ice core temperature records from opposite hemispheres

In this exercise, you will load and save, plot, and analyze two of the best-known paleoclimate records (ice core records EPICA Dome C from Antarctica and GISP from Greenland). MATLAB skills used include loading data from text files, comparing two time series visually and using cross-correlation, and manipulating data using `for` loops.

Tasks:

1. Follow this link to download the text file edc3deuttemp2007.txt. Move the file into a new directory called `ice_cores`.

2. Use importdata() to extract columns of temperature and time from edc3deuttemp2007.txt. You can ignore times after 1950 when not all observations are available (try using the NHEADERLINES option in importdata). Save a file called 'epica.mat' containing variables time and temp.

2. Follow this link to download the GISP2 (Greenland) ice core temperature proxy record for the past 50 kyr, gisp2_temp_accum_alley2000.txt. Again use importdata and save a file called 'gisp.mat' containing variables time and temp.

3. Write a script that loads the matlab files you have created and makes plots of temperature versus time. (Note the units of the time variables, and don't forget that variables time and temp from one core will be overwritten when you load the other core file!) Try using the "hold" function to plot both records on the same axis and add a legend. You can use subplot to make two axes on a single figure window. To compare the records on scaled axes, try plotyy(). To look at time intervals covered by both records, use xlim() or use logical indexing.

Optional:

4. Change the direction of time to what we are used to (right now it goes backwards, left to right in a plot). Try using fliplr() or flipud().

5. MATLAB has many built-in tools for analyzing and comparing time series. After interpolate the records to be evenly spaced in time (using interp1; see Problem 3, part 2), use xcorr() to look at the cross-correlation between the two records. Is there evidence

for variability that is in phase or out of phase between the two hemispheres?

6. Instead of specifying file names explicitly in your script, use a for loop and the `dir()` command. and to load and plot these records. This procedure scales up to a very large number of data files. The `dir` command will be useful.

7 Diagnosing correlations between ocean tracers

Water parcels in the ocean are "tagged" by scalar properties called tracers that can reflect ocean density, biology, and flow history, and other factors. Here we consider global shipboard observations of three such quantities: temperature, salinity, and the oxygen isotope ratio δ^{18}O, which is determined chiefly by evaporation and precipitation. For more background, see the original publication (Legrand and Schmidt 2006, link here).

This exercise diagnoses relationships between ocean temperature, salinity, and δ^{18}O. MATLAB skills used include loading data in fixed-width format, manipulating a large data set, plotting data, and computing correlations.

Steps:

1. Data can be obtained from a NASA website here. Information about the raw data set is near the bottom of the page. Download the 3 Mb data file here and unzip it into a folder called oceanTracers.

2. Write a script to load these data into MATLAB using textscan. The 'Format' option allows you to specify how columns of data are selected from the text file. The appropriate string (`%7.2f%6.2f%2d%4d%5d%6.2f%6.2f%6.2f%6.1f%15s%60s`) is given on the data webpage; type help textscan to understand what this means. Create six vectors called lonData, latData, depth, t, s, and d18 and save them in a file called tracers.mat.

3. Write a second script to load tracers.mat and plot the data. Using the scatter() and subplot() commands, make plots, for points that are in the top 5m of the ocean, of δ^{18}O vs temperature, colorcoded by salinity; δ^{18}O vs salinity, colorcoded by temperature; and temperature vs salinity, colorcoded by δ^{18}O. Label axes with the appropriate units. Note that missing observations in the dataset have values '-99.9'; you will want to eliminate data that are missing any of the three variables.

Optional:

4. For each pair of two variables, compute equations for lines of least-squares best fit using polyfit(). Superimpose these lines on the data plots by modifying the code for step 3. Use corrcoef() to compute the correlation and report it in figure titles. Does a linear equation adequately represent the data?

5. Plot average values of δ^{18}O, temperature, and salinity in the top 5 meters on a global map using scatter(). To obtain a built-in coastline, try "load coast".

6. Repeat step 4. but for observations below 2000 meters. How do the plots differ? Can you speculate as to a reason?