Outline

- Announcements:
  - Homework II: Solutions on web
  - Homework III: due Wed. by 5, by e-mail
- Homework II
- Differential Equations
- Numerical solution of ODE’s
- Matlab’s ODE solvers

Homework II

- Good job
- Something I learned:
  - logical addressing
    - I=some logical test of A
    - A(logical(I)) are elements in A where true
    - mean(data(data~=-999))
- nargin
- Final FourierStuff

nargin

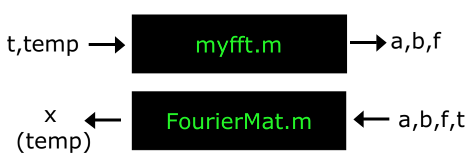
- Matlab functions can be polymorphic--the same function can be called with different numbers and types of arguments
  - Example: plot(y), plot(x,y), plot(x,y,'rp');
- In a function, nargin and nargout are the number of inputs provided and outputs requested by the caller.
- Even more flexibility using vargin and vargout

nargin

- Simplest application of nargin is to override default parameters

```
function [x,y]=ll2xy(lat,lon,ref,R)
if(nargin<3)
    R= 6378.155*1000;
    ref=[42.3493, -71.0264];  %default ref is Boston
elseif(nargin==4)
    if(length(ref)~=2)%ref must be R
        R=ref;
        ref=[42.3493, -71.0264];  %default ref is Boston
    else
        R= 6378.155*1000;
    end
end
```

The Final (Fourier) Analysis



- Use myfft to get spectrum (a & b)

Fourier Analysis

- Spectrum tells you about time series
  - dominant frequencies
  - underlying statistical processes
- Could use FourierMat to filter data

```
>>alp=a;blp=b;
>>alp(13:end)=0;blp(13:end)=0;
>>xlp=FourierMat(alp,blp,f,t);
```

Differential Equations

- Ordinary differential equations (ODE's) arise in almost every field
- ODE's describe a function y in terms of its derivatives
- The goal is to solve for y

Example: Logistic Growth

- Similar to swan problem on PS3
- N(t) is the function we want (number of animals)

$$\frac{dN}{dt} = rN(\frac{K-N}{K})$$

Numerical Solution to ODEs

- In general, only simple (linear) ODEs can be solved analytically
- Most interesting ODEs are nonlinear, must solve numerically
- The idea is to approximate the derivatives by subtraction

Euler Method

$$\begin{aligned} \frac{dN}{dt} &= f(N,t) \\ \frac{N_{t+1}-N_t}{\Delta t} &= f(N_t,t) \\ N_{t+1} &= N_t + \Delta t * f(N_t,t) \end{aligned}$$

Euler Method

- Simplest ODE scheme, but not very good
- "1st order, explicit, multi-step solver"
- General multi-step solvers:

$N_{t+1} = N_t + \Delta t *$ (weighted mean of f evaluated at lots of t's)

Runge-Kutta Methods

- Multi-step solvers--each N is computed from N at several times
  - can store previous N's, so only one evaluation of f/iteration
- Runge-Kutta Methods: multiple evaluations of f/iteration:

$$
\begin{aligned}
a &= \Delta t f(N_t, t) \\
b &= \Delta t f(N_t + a/2, t + \Delta t/2) \\
c &= \Delta t f(N_t + b/2, t + \Delta t/2) \\
d &= \Delta t f(N_t + c, t + \Delta t) \\
\\
N_{t+1} &= N_t + 1/6 * (a + 2b + 2c + d)
\end{aligned}
$$

Matlab's ODE solvers

- Matlab has several ODE solvers:
  - ode23 and ode45 are "standard" RK solvers
  - ode15s and ode23s are specialized for "stiff" problems
  - several others, check help ode23 or book

Matlab's ODE solvers

- All solvers use the same syntax:
  - [t,N]=ode23(@odefile, t, N0, {options, params …})
    - odefile is the name of a function that implements f
      - function f=odefile(t, N, {params}), f is a column vector
  - t is either [start time, end time] or a vector of times where you want N
  - N0= initial conditions
  - options control how solver works (defaults or okay)
  - params= parameters to be passed to odefile

Example: Lorenz equations

- Simplified model of convection cells

$$\begin{aligned}\frac{dx}{dt} &= \sigma(y-x)\\ \frac{dy}{dt} &= rx-y-xz\\ \frac{dz}{dt} &= xy-bz\end{aligned}$$

- In this case, N is a vector =[x,y,z] and f must return a vector =[x',y',z']