Input Output

Garbage In,
Garbage Out

Outline

- Announcements:
  - Homework III: due Today. by 5, by e-mail
    - Discuss on Friday.
  - Homework IV: on web, due following Friday
- Linear Systems Example: Advection-Diffusion
- Advanced ASCII
- Binary Basics

Key Points on Linear Systems

- They're everywhere
- Easy to solve in Matlab (\)
- If possible, reuse factors from LU-decomposition
- Never use inv(A) for anything important
- Linear algebra (analytical and numerical) are highly recommended

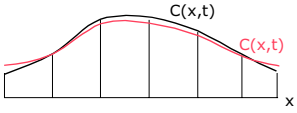
Advection-Diffusion

$$\frac{\partial C}{\partial t} = u\frac{\partial C}{\partial x} + k\frac{\partial^2 C}{\partial x^2}$$

- Model concentration of "contaminant" C
- Similar equations occur in
  - fluid dynamics
  - developmental biology
  - ecology

Numerical Solution

- We start with an initial distribution of C over the interval [0 1]
- Divide [0 1] into discrete points separated by dx



- C(x,t+dt) will depend on C(x), C(x-dx), & C(x+dx)

Numerical Solution

- replace partial derivatives with differences:

$$\frac{C_a^{t+dt} - C_a^t}{dt} = u\frac{C_{a+da}^t - C_a^t}{dx} + k\frac{C_{a+da}^{t+dt} - 2C_a^{t+dt} - C_{a-da}^{t+dt}}{dx^2}$$

$$C_a^{t+dt} - \sigma(C_{a+da}^{t+dt} - 2C_a^{t+dt} - C_{a-da}^{t+dt}) = C_a^t + \lambda(C_{a+da}^t - C_a^t)$$

$$\begin{bmatrix} -\sigma & (1+2\sigma) & -\sigma \end{bmatrix} * \begin{bmatrix} C_{a-da}^{t+dt} \\ C_a^{t+dt} \\ C_{a+da}^{t+dt} \end{bmatrix} = C_a^t + \lambda(C_{a+da}^t - C_a^t)$$

- The solution of C(x,t+dt) depends on neighboring points

Numerical Solution

- Each C_x will have a row in matrix A
- All rows are the same except for first and last
  - We need to specify what happens at end points
  - Boundary conditions are a big problem
  - We'll use periodic BC's
    - C(0)=C(1), so first and last rows are:

$$\begin{bmatrix} (1+2\sigma) & -\sigma & \dots & -\sigma \end{bmatrix}$$
$$\begin{bmatrix} -\sigma & \dots & (1+2\sigma) & -\sigma \end{bmatrix}$$

Sparse Matrices

- A is sparse
  - the only non-zero elements are immediately above, below, and on the diagonal
  - corners for periodic BC's
- Matlab has special sparse matrices
  - much less memory (don't need space for 0's)
  - faster to process
  - A=sparse(I,J,S) forms A s.t. A(I(j),J(j))=S(j)

AdvDiff1D.m

- Uses slightly more complicated procedure for advection known as "Lax-Wendroff"
- Must specify
  - Initial concentration C0
  - parameters (u, k)
  - size of domain L
  - length of time T,
  - dx, dt
- Returns x, t, and N(x,t)

Advanced ASCII

- Read tables of ASCII data with load
- Other functions like textread will read simple files
- Sometimes, you've just got to do it yourself
  - Complicated files

Opening Files

- To read a file manually, open with fopen
  - fid=fopen('fname', 'rt');
  - fid will be <1 if open fails
- File I/O functions accept fid
- Close the file when you're done with fclose(fid)

Reading files

- A=fscanf(fid,cstring,{N})
  - like C's fscanf, cstring is a C format string:
    - '%d\t%f'--integer (%d),tab(\t),double (%f)
  - fscanf is "vectorized" and Matlab will keep trying to match cstring. Limit with N
- lin=fgetl(fid)
  - Reads a single line from the file as text (char array)
  - Process lin with str2num, findstr, sscanf
- Test for end of file with feof(fid);

Writing Files

- Save matrices using save fname varname -ascii
- Doing it yourself:
  - fid=fopen('fname','wt')
  - fprintf(fid,cstring, variables)
  - Example:
    - A=[(1:10)', sin(2*pi*0.1*(1:10)')];%[integers, doubles]
    - fid=fopen('example.txt','wt');
    - fprintf(fid,'%d %f\n',A');
    - fclose(fid);

Binary Basics

- All computer files are "binary", that is composed of 0's and1's
- When the computer reads ASCII files, it takes chunks of 8 bits (1 byte) and looks up the character
- To save pi to 16 digits takes 18 bytes in ASCII
- If you save the 1's and 0's that correspond to the double precision value of pi, that takes only 8 bytes

Problem with Binary Files

- You can't just look at them
- You must know exactly how they were created
  - integers vs. floating point
  - single precision vs. double precision
  - signed vs. unsigned

Reading Binary files

- fid=fopen(fname,'r');
- A=fread(fid,N,precision)
  - N=number of data points
  - precision is how the file was created
    - "uint64" is an unsiqned integer saved in 64 bits
    - "double" is a double

Free advice (you get what you pay for)

- The only reasons to use binary files are
  - someone gives you one
  - you enjoy frustration and pain
  - you're too poor (or cheap) to buy a new hard drive