MET3220C
Computational Statistics

Programming – AS #7
Dr. Mark Bourassa

Topics:
Binary Files
Cumulative Probability Distributions
AS #7

Turn in your program and plots.
Due in one week.

Binary Files

- Large data sets take up too much space (memory) if they are stored in ASCII format. There are many alternatives involving binary data.
  - ASCII data is stored as characters.
    - For example, 3.14159 is stored as 7 words (7 sets of ones and zeros or bits): a word for 3, another word for the decimal, another for one, another for 4, and so one.
  - Binary uses one word (a certain number of ones and zeros, e.g., 16 or 32) to store a number.
- Binary is based on base 2 numbers (ones and zeros).
- Integer example:
  - 0001B = 1D; 0010B = 2D; 0011B = 3D; 0100B = 4D; 0101B = 5D
  - Where B indicates binary and D indicates base 10 numbers that you are used to.
- Real data are typically stored in double the space of an integer. Some of the space is used for a binary number, and some for an exponent.

Comments on Numerical Error in Summation

- Computers store data as binary numbers (ones and zeros) in a set amount of computer memory. The amount of space depends on the computer system and settings in the program
  - Example: an integer might have the space for 16 ones and zeros (bits).
- This situation means there is a limit on how big or small a number can be, and be stored in memory. Why?
- For integers, the first binary digit is used as a sign (+ or –).
  - If there are n binary digits, that leaves n – 1 digits for magnitude.
  - Zero is one of allowable numbers, therefore the largest magnitude is equal to one less than the number of possible numbers.
- The formula for the largest integer magnitude is $2^{n-1} - 1$.
- If you are calculating a sum of integer values, this is a key limiting factor. Example: sum of all ECMWF surface pressures, in units of Pa, for each six hour period over 40 years.
  - How can you deal with this problem?

Comments on Numerical Error in Summation

- One (good) approach is to work with REAL variables, rather than integers.
  - Real numbers are much more complicated, so we will go over the concept, but not the gory details.
  - Real numbers have the same number of ones and zeros (bits) as integers, but they are arranged as a sign, and exponent, and a mantissa.
- REAL numbers have a MUCH wider range of values; however, in many cases they can only approximate base 10 numbers (whole numbers and fractions).
  - Example, this approximation is why we don't test if a data value is EQUAL to a REAL missing value.
  - Assume that the rounding error for any one addition (in a sum) will be of similar scale to all other rounding errors. Then apply your error propagation formula.
  - The rounding error for x^2 is greater than the error for x.

Opening Binary Data Files

- When opening a standard IEEE or FORTRAN binary file use the option form='unformatted' .
  - Example:
    - OPEN(67, file=filename,ERR=30,form='unformatted')
    - Where 67 is the number of the IO stream number (the identifier of the link connecting the program and the file),
    - Filename is the name of a file in quotes, or a character string that contains the name of the file,
    - 30 is a label to go to if there is an error opening the file, and
    - 'unformatted' means binary.
- Closing the file (severing the link) is done as for any file.
  - CLOSE(*IO Stream number*)
  - Example:
    - CLOSE(67)

Reading and Writing Binary Data

- Data can be read in as a series of variables, some of which can be arrays.
- Example:
  - real*4 hwind_lat(159), hwind_lon(159), hwind_u(159,159), &
  - hwind_v(159,159), speed
  - read(67) hwind_lon ! Reads H*wind longitudes
  - read(67) hwind_lat ! Reads H*wind latitudes
  - read(67) hwind_u ! Reads H*wind u wind components
  - read(67) hwind_v ! Reads H*wind v wind components
- The real*4 indicates the space in memory used to store a single scalar variable. A bigger number means more precision.
- Note that there is no need to specify a format. The code knows how the data is supposed to be formatted.
- Writing works the same way as the read, but uses the WRITE command that you are familiar with.

Cumulative Probability Distributions

- Cumulative probability distributions are the integral (which can be approximated as a sum) of the area under the PDF, from the extreme left hand side of the PDF to each value on the independent axis.

$$CDF(x) = \int_0^x PDF(x')dx'$$

- For ease of use with a computer, it is preferred to approximate this as a sum.

$$CDF(x_I) = \sum_{i=0}^{i=I} \left(PDF(x_i) bin\_width(x_i) \right)$$

- In computer terms (for constant bin width), we might think of this as
  - CDF_i = bin_width * SUM(pdf(1:i))

Assignment #7

- Copy the code from assignment #6 to assignment #7.
- Calculate the cumulative probability of your three distributions.
  - Assume that you can start with a wind speeds of -10 m/s, and integrate (sum) up to cumulative probability corresponding to a wind speed greater than the lower limit.
  - The highest wind speed bin to consider has an upper limit of 60m/s.
  - Note: you will have to modify the code to deal with wind speeds as low as -10 m/s.
  - A smaller bin width will result in a smoother CDF.
- 3) Write the three cumulative probabilities to files. Write the bin center and the corresponding probability.
- 4) Plot the three cumulative probability distributions on one plot.
- 5) Turn in the code and plot.Add code to dynamically allocate an array of speed data. See previous example.