Any Questions?
Topics for this week

- IEEE
- Missing data
- Histograms
- Compiler Options
- Reading NetCDF data
- Zonal averages
- Averages and contour plots
- Another simple climate model
- Debugging strategies
Institute of Electrical and Electronics Engineers

Most members of any technical professional organization in the world.

IEEE produces standards for almost everything computer related, and keeps them updated.

Today, most IEEE standards have generally been adopted.

IEEE 754 is the most widely-used standard for floating-point computation, and is followed by many hardware (CPU and FPU) and software implementations.
INF and NaN

INF is defined in the standard as the value given to any Real that is outside the limits of the type.

Fortran (and IEEE) has +INF and -INF.

NaN (Not a Number) produced as the result of an improper floating point calculation.

NaN is not equal to either INF. In fact, in the IEEE standard, NaN is not even equal to itself.

INF or NaN are occasionally used as placeholders for missing data.
Missing Data

Any observational dataset is going to have holes.

If missing data is not given as an “outside the bounds” value (-9999 or 9999.0) it is often replaced with INF or NaN.

Most Fortran implementations will read INF or NaN in as a Real value (it is a real Real), we need to check for it before doing calculations, or we’ll get a runtime error.
Histograms

As an example of how to do calculations with missing data, we're going to make a quick histogram.

A histogram is a plot of the number of times a value is seen in the given dataset.

All lons, lats and times will be binned together equally.
Assembler Flags

As far as I can tell, there is no standard for Compiler options. You’ll have to look up the ones supported by your own compiler.

I’ll give you some good ones from my (Absoft) Fortran 90/95 compiler.

Many will likely be supported by most other compilers.

If you want your code to be portable, don’t depend on compiler tricks and flags.
Compiler Flags

- c  compile without creating the executable
- L[opath]  add a library search path
- l[ofile]  add a library file or module
- O  perform a group of basic optimization that will cause most code to run faster
- v Verbose - prints out status info as the code is being compiled
- w  Supress output of warnings
Compiler Flags

- **-ej** causes all do loops to be executed at least once, regardless of iteration controls
- **-Rb** generate meta-code to check array boundaries
- **-Rc** validate sub-string indexes
- **-Rs** check for array conformance
- **-YCSLASH** directs the compiler to evaluate C-style escape characters in strings (/n, /t, /r)
NetCDF Data

NetCDF is an I/O library that is widely used in the earth sciences.

Once the files are installed, you can use their procedures to open and access the files.

Each files is “self-describing,” all of the data is annotated (dimension, units, range of values, missing data values, etc...)

Examples: read_netCDF.f90 with data from NCEP (NCEP.Precip.0100-1204.nc)
Zonal Average Example

- Modelers and Dynamicists like to look at the atmosphere in latitudinal bands.
- Don’t have to worry about missing data here...
- Loading in precip data is pretty simple if you know the parameters.
- When you do a zonal average, first average in time at each point and then average across all longitudes.
- Could come up with a less memory intensive way to get the same result...
- PlayWithPrecip.f90
Climatology Map Example

- Climatologies are straight-forward averages through time, and the result (for precip) is a two-dimensional map.

- NCEP has a land-mask map that you could use to fill in the land behind the data.

- How might you do the average if there were missing data?

- Anomalies are the result of subtracting the time average at each point - anomalies will still have all 3 (or 4) dimensions.
One last “Climate Model”

- We can time-step our simple model using a do loop.
- Build the model from smaller pieces to keep it simple and understandable.
- Abstracting the temp calculations into a separate function lets us use it as a “black box”
- Write out results at each timestep to the same file
- OR write out to a new file for each “year”
Debugging

- Code will rarely do what you expect on the first try (it always does what you told it to do though...)
- Compiler errors are usually straight-forward
- Runtime errors are harder
  - File I/O errors
  - Segmentation Fault
  - Bus Error
- Wrong answer...
Debugging

- When things are doing what you expect them too...
  - Print statements are the common first stop
  - Simplify your code – do one thing at a time
    - test each piece separately
  - Work out the math by hand or in another program (matlab, idl...)
  - Google! – look for documentation, descriptions of error codes, or examples that might work
  - Talk to somebody!

Tuesday, March 10, 2009
What did we cover this week?

- IEEE
- Missing data
- Histograms
- Compiler Options
- Reading NetCDF data
- Zonal averages
- Averages and contour plots
- Another simple climate model
- Debugging strategies