Graphics tools for MUSC

Andres Luhamaa
University of Tartu

Motivation

- Output contains many ascii files
- Anyone can plot data from single ascii file, but handling many files is not straightforward.
- Besides plotting, one may need to do post-processing (statistics, comparison, etc)
- Would common approach to handling output data be useful?
  - How to make such common approach that people would like to learn it instead of just using directly the ascii files and writing own scripts?
  - What are the preferred graphics programs?

MUSC ascii output (some intial findings)

- one output (provided by Laura) analysed
    - 60 or 61 levels (full and half levels?)
    - 60 level values are not the same, but very close, from file to file?
    - file names can be used as variable names in scripts and data files
    - timesteps in different files
    - some files are empty
    - some files are descriptions, not data
    - many files

grads and matplotlib

- grads
  - own file format
  - easy handling of temporal data (selecting timestep etc)
  - one way of doing things
  - limitid scripting capabilities
  - interactive use possible
  - not so easy to extend
  - ...

- python-matplotlib
  - any file format
  - powerful programming language
  - many ways of doing things (organizing data, plotting)
  - good scripting capabilities + scientific computing libraries (scipy, numpy)
  - interactive use with „ipython“
  - extend anywhere, add anything
  - some knowledge about python required, but this is useful anyway.
  - ...

convert_to_grads.py

- small script to convert available ascii files to grads data and descriptor files
- with ipython, interactive use can be continued, all data is kept in memory
- two sets of files are created, because grads can not handle two versions of vertical coordinates in the same file?
- requires python and python-numpy, which are available on almost any system
- usage: python convert_to_grads.py
- you must specify data directory inside script (could be modified later)

workflow example

- grads
  - open test60.ctl
  - q file
  - d <whatever>
  - set t 3
  - ...
  -

- ipython -pylab convert_to_grads.py
  - jada60.keys() # will list variables in array
  - plot(jada60['<variable>'][:,0],jada60['<variable>'][:,1])

For 61 level data, just replace 60 everywhere with 61. For using python interactively, packages „ipython“ and „python-matplotlib“ are required.