AMath 483/583 — Lecture 23 — May 18, 2011

Today:

- MPI in subroutines
- Comments on Homework 6
- Python plotting

Friday:

- Grady on GPUs

Read: Class notes and references

New MPI examples.

Send me your info if you want totalview

Make sure Python plotting works

MPI with subroutines and functions

Recall Simpson's rule program from Homework 5:

In OpenMP: Subroutine is called by the single master thread running the main program

Inside the subroutine a single `omp parallel` block is used to fork a set of threads that are used for the full computation.

End of a parallel block kills all threads except master thread.

In MPI: First statement in main program must be `MPI_INIT`.

It's not possible to call `MPI_INIT` in the subroutine.

The entire code (including main program and call to subroutine) is executed by each process (maybe on different computers!).

Call to `MPI_FINALIZE` kills all processes.

MPI with subroutines and functions

MPI version of Simpson's rule program:

`$CLASSHG/codes/mpi/quadrature`

Notes:

- There is no master process except that we may decide some things should only be done by Process 0, for example.
- The module variable `gcount_proc` is a global variable, but is still private to each process.

  All variables are private, no shared variables!

Python plotting

In `$CLASSHG/codes/mpi/heat1d`:

```
$ make plots
```

Executes `$CLASSHG/codes/python/plotheat1d.py` and produces `plot.png`.

In `$CLASSHG/codes/fortran/heat2d`:

```
$ make plots
```

Executes `$CLASSHG/codes/python/plotheat2d.py` and produces `pcolor.png` and `contour.png`.

In Homework 6, use this same plotter `$CLASSHG/codes/python/plotheat2d.py`.

Python plotting

Can also plot interactively:

```
$ cd $CLASSHG/codes/fortran/heat2d
$ make heatsoln.txt  # runs code

$ ipython -pylab

In[1]: run ../../python/plotheat2d.py

In[2]: show()

In[3]: Quit

$
```


Using matplotlib

```
$ ipython -pylab
```

starts `ipython` in manner that interactive plots work.
This also automatically does...

```
from pylab import *
```

which puts all NumPy and matplotlib plotting routines in namespace, so e.g.:

```
In [1]: x = linspace(0, 1, 101)
In [2]: plot(x, x**2, 'r-o')
```

To make it clear where things come from:

```
In [1]: import numpy as np
In [2]: from matplotlib import pyplot as plt
In [3]: x = np.linspace(0, 1, 101)
In [4]: plt.plot(x, x**2, 'r-o')
```


Python plotting with matplotlib

Best way to learn is to browse the gallery:

http://matplotlib.sourceforge.net/gallery.html

See the class notes for some tips and other pointers:

Python plotting section