Cornell University
Center for Advanced Computing

Data Analysis with MATLAB

Steve Lantz
Senior Research Associate
Cornell CAC

Workshop: Data Analysis on Ranger, October 12, 2009

MATLAB Has Many Capabilities for Data Analysis

- Preprocessing
  - Scaling and averaging
  - Interpolating and decimating
  - Clipping and thresholding
  - Extracting sections of data
  - Smoothing and filtering
- Applying numerical and mathematical operations
  - Correlation
  - Basic statistics and curve fitting
  - Fourier analysis and filtering
  - Matrix analysis
  - 1-D peak, valley, and zero finding
  - Differential equation solvers

Toolboxes for Advanced Analysis Methods

- Curve Fitting
- Filter design
- Statistics
- Communications
- Optimization
- Wavelets
- Spline
- Image processing
- Symbolic math
- Control system design
- Partial differential equations
- Neural networks
- Signal processing
- Fuzzy logic

Workflow for Data Analysis in MATLAB

- Access
  - Data files - in all kinds of formats
  - Software - by calling out to other languages/applications
  - Hardware - using the Data Acquisition Toolbox, e.g.
- *Pre-process… Analyze… Visualize…*
- Share
  - Reporting (MS Office, e.g.) - can do this with touch of a button
  - Documentation for the Web in HTML
  - Outputs for design
  - Deployment as a backend to a Web app
  - Deployment as a GUI app to be used within MATLAB

A Plethora of Routines for File-Based I/O

- High Level Routines
  - LOAD/SAVE
  - UIGETFILE/UIPUTFILE
  - UIIMPORT/IMPORTDATA
  - TEXTSCAN
  - XMLREAD/XMLWRITE
  - CSVREAD
  - DLMREAD/DLMWRITE
  - XLSREAD
  - IMREAD
- See “help iofun” for more
- Low Level Common Routines
  - FOPEN/FCLOSE
  - FSEEK/FREWIND
  - FTELL/FEOF
- Low Level ASCII Routines
  - FSCANF/FPRINTF
  - SSCANF/SPRINTF
  - FGETL/FGETS
- Low Level Binary Routines
  - FREAD/FWRITE

Example: Importing Data from a Spreadsheet

- Available functions: xlsread, dlmread, csvread
  - To see more options, use the “function browser button” that appears at the left margin of the command window
- Demo: Given beer data in a .xls file, use linear regression to deduce the calorie content per gram for both carbohydrates and alcohol

```
[num,txt,raw] = xlsread('BeerCalories.xls')
y = num(:,1)
x1 = num(:,2)
x2 = num(:,4)
m = regress(y,[x1 x2])
plot([x1 x2]*m,y)
hold on
plot(y,y,'r')
```

Lab: Setting Data Thresholds in MATLAB

- Look over count_nicedays.m in the lab files
  - Type “help command” to learn about any command you don’t know
  - By default, “dlmread” assumes spaces are the delimiters
  - Note, the pair of “find” commands does the thresholding
  - Here, the .* operator (element-by-element multiplication) is doing the job of a logical “AND”
  - Try calling this function in Matlab, supplying a valid year as argument

- Exercises
  - Let’s say you love hot weather: change the threshold to be 90 or above
  - Set a nicedays criterion involving the *low* temps found in column 3
  - Add a line to the function so it calls “hist” and displays a histogram

The Function count_nicedays

```
function nicedays = count_nicedays( yr )
%COUNT_NICEDAYS returns number of days with a high between 70 and 79.
%   It assumes data for the given year are found in a specific file
%   that has been scraped from the Ithaca Climate Page at the NRCC.

% validateattributes does simple error checking –
% e.g., are we getting the right datatype
validateattributes(yr,{'numeric'},{'scalar','integer'})
filenm = sprintf('ith%dclimate.txt',yr);
result = dlmread(filenm);
indexes = find((result(:,2)>69) .* (result(:,2)<80));
nicedays = size(indexes,1);

end
```

- What if we wanted to compute several different years in parallel?...

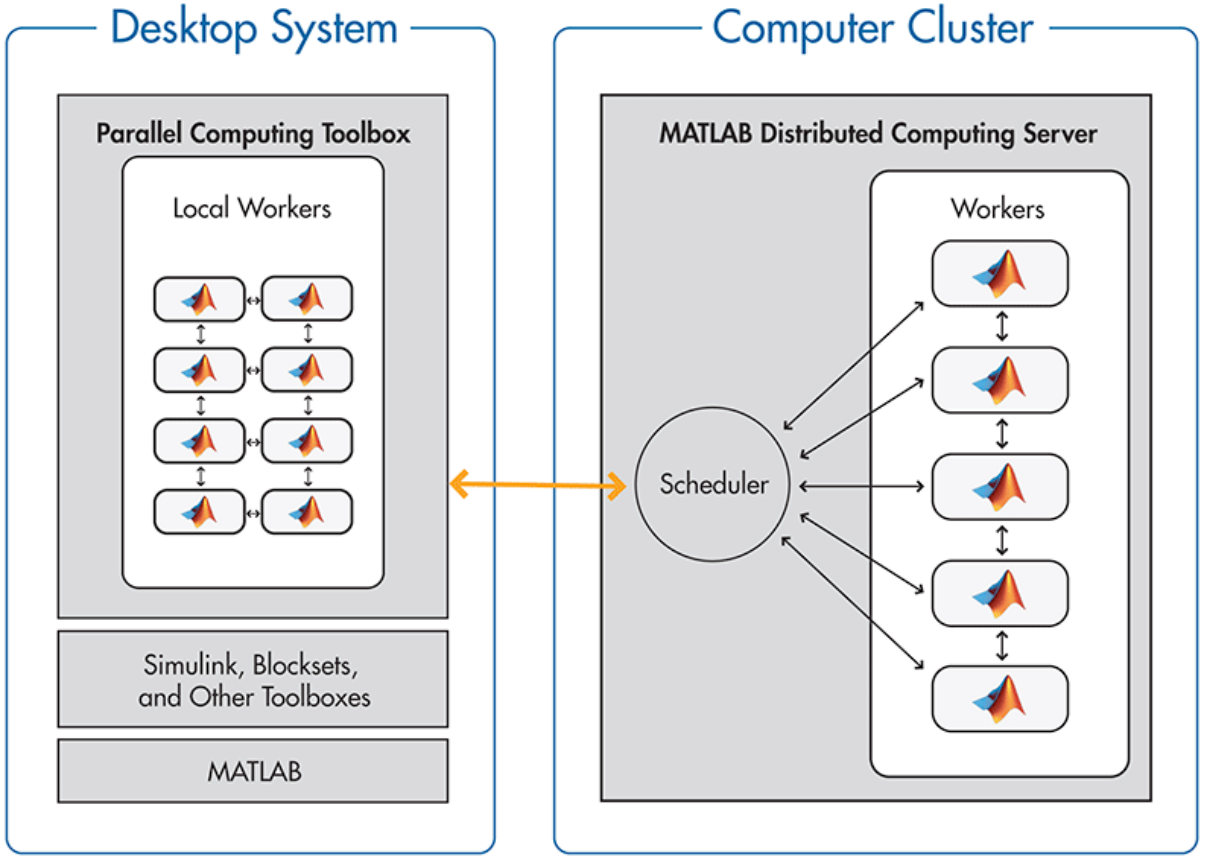
How to Do Parallel Computing in MATLAB

- Core MATLAB already implements multithreading in its BLAS and in its element-wise operations

- Beyond this, the user needs to make changes in code to realize different types of parallelism, in order of increasing complexity:
  - Parallel for loops (parfor)
  - Codistributed arrays, for big-data parallelism
  - Parallel code constructs and algorithms in the style of MPI

- The user's configuration file determines where the workers run
  - Parallel Computing Toolbox - take advantage of multicores, up to 8
  - Distributed Computing Server - use computer cluster (or local cores)

Access to Local and Remote Parallel Processing



Dividing up a Loop Among Processors

```
for i=1:3
count_nicedays(2005+i)
end
```

- Try the above, then try this easy way to spread the loop across multiple processors (note, though, the startup cost can be high):

```
matlabpool
parfor i=1:3
count_nicedays(2005+i)
end
```

- Note, matlabpool starts extra copies of matlab.exe which do not count against the license; the size of this worker pool is set by the default “local” configuration - usually it’s 4, but it can go up to 8

What is *parfor* Good for?

- It can be used for *data parallelism*, where each thread works on independent subsections of a matrix or array
- It can be used for certain kinds of *task parallelism*, e.g., by doing a parameter sweep, as in our example (“parameter parallelism?”)

- Either way, all loop iterations must be totally independent
  - Totally independent = “embarrassingly parallel”
- Mlint will tell you if a particular loop can't be parallelized

- Parfor is exactly analogous to “parallel for” in OpenMP
  - In OpenMP parlance, the scheduling is “guided” as opposed to static
  - This means N threads receive many chunks of decreasing size to work on, instead of simply N equal-size chunks (for better load balance)

How to Do Nearly the Same Thing Without PCT

- Create a MATLAB .m file that takes one or more input parameters
  - The parameter may be the name of an input file, e.g.
- Use the MATLAB C/C++ compiler (mcc) to convert the script to a standalone executable
- Run N copies of the executable on an N-core machine, each with a different input parameter
  - In Windows, this can be done with “start /b”
- For fancier process control or progress monitoring, use a scripting language like Python
- This technique can even be extended to a cluster
  - mpirun can be used for remote process initiation, even though we’re not using MPI
  - The Matlab runtimes (dll’s) must be available on all cluster machines

Advanced Parallel Data Analysis

- Over 150 MATLAB functions have been overloaded to operate on codistributed arrays
  - Such arrays are actually split among mutliple MATLAB sessions
  - In the command window, just type the usual e = d*c;
  - Under the covers, the matrix multiply is executed in parallel using MPI
  - Some variables are cluster variables, while some are local
- Useful for large-data problems that require distributed computation
  - How do we define large? - 3 square matrices of size 6000 typically exceed available memory on one computer
- Nontrivial task parallelism or even whole parallel algorithms can also be expressed
  - createTask(job...), submit(job) for parallel tasks
  - Many MPI functions have been given MATLAB bindings, e.g., labSendReceive, labBroadcast; these work on all datatypes

Share Results

- Push the “publish” button to create html, doc, etc. from a .m file
  - Feature has been around 5 years or so
  - Plots become embedded as graphics
  - Section headings are taken from cell headings
- Use cells to organize your work
  - Create cells in .m file by typing a %% comment
  - Cells can be re-run one at a time in the execution window if desired
  - Cells can be “folded” or collapsed so that just the top comment appears
- Share the code in the form of a deployable application
  - Simplest: send MATLAB code (.m file, say) to colleagues
  - Use MATLAB compiler to create stand-alone exes or dlls
  - Use a compiler add-on to create software components for Java, .NET