Cornell University
Center for Advanced Computing

# Best Practices

Tips and Tricks for using the PCT and TUC

# General

- Beware of multiple processes writing to the same file
- Avoid the use of global variables
- Consider whether or not you’ll need to recompile your MEX-files

# Code Improvement

- Profile your code to search for bottlenecks
  - The MATLAB profiler is easy to use and provides a lot of information quickly.  When the profiler isn't applicable, gather timing information, data is your best friend as you develop.
- Make use of M-Lint when coding parfor and spmd
  - It isn't always obvious what MATLAB does inside of parfor and spmd blocks.  Take advantage of M-Lint messages to identify opportunities for improving code.

# Verbosity

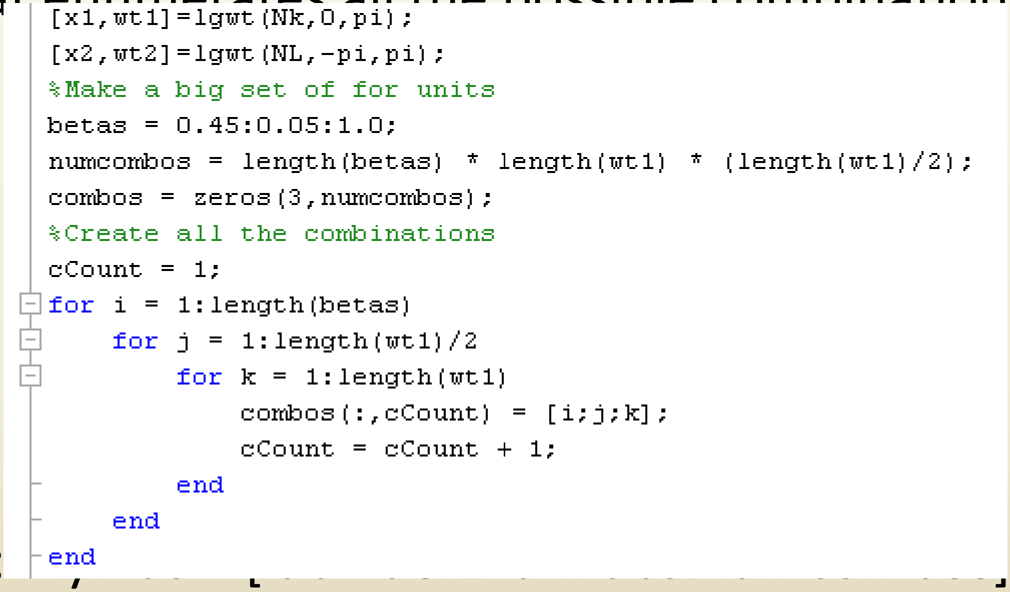
- Display the correct amount of verbosity for debugging purposes.
  - Long runs on TUC that don’t produce the results you want and don’t produce any debug information are a bummer.  Use CommandWindowOutput and littleJohnLog.
- Implement an error handler, including capture of calls to 3rd party functions – don’t assume calls to libraries succeed
  - This includes LittleJohn calls!  But be careful with throwing around try-catch statements.  Sometimes the full MATLAB stacktrace is the most useful information.

# Prototyping

- Migrate from scripts to functions
  - It’s easy to start developing something with scripts, but move to functions rapidly.
- Avoid hard coding path and filenames that don’t exist on the cluster
  - Include a switch for when you’re running on TUC if you need to write data paths.
- Minimize code decisions that will only work on TUC or your development machine.
  - MATLAB functions should be included by path adjustments
  - Minimize things like cd or relative path expectations.

# Handling nested loops

- Nested for loops can be parallelized by using a combinations strategy that enumerates all the possible combinations.



```
[x1,wt1]=lgwt(Nk,0,pi);
[x2,wt2]=lgwt(NL,-pi,pi);
%Make a big set of for units
betas = 0.45:0.05:1.0;
numcombos = length(betas) * length(wt1) * (length(wt1)/2);
combos = zeros(3,numcombos);
%Create all the combinations
cCount = 1;
for i = 1:length(betas)
    for j = 1:length(wt1)/2
        for k = 1:length(wt1)
            combos(:,cCount) = [i;j;k];
            cCount = cCount + 1;
        end
    end
end
```

- Parallel job:
- Parfor: for i=1:numcombos

# Solving Problems

- Parallel jobs can fail in interesting and complex ways. Most of them are due to a bad signature match on the task function. When a parallel job fails strangely, check that the number of input arguments are right and the function name is correct.
- When really confused, comment out your entire task function and replace it with a single fprintf statement. Still doesn't work?
  1) Check your createTask line to ensure signature is good
  2) Use gridFTP to download the .m file from TUC,is it what you think?

# Solving Problems

- Distributed (createJob) jobs fail cleanly in almost all cases and produce errors.
- Parallel Jobs (createParallelJob, createMatlabPoolJob) don't. Don't be discouraged, parallel programming is hard and your error exists somewhere. You just have to find it.
- Write prep functions and mix PathDependencies and FileDependencies as necessary to make things go easily.
- Debug locally whenever possible, the run-fail cycle is faster locally so you can figure things out faster.

# Parallel vs Pool

- The majority of parfor's can be easily written as parallel jobs. The choice of which is better comes down to:
  - Parfor's can be easier to debug locally since pools can be left open and you modify your parfor code until it works.
  - Parallel jobs give you better control over the parallelism and give you better more obvious control over when and what data is being sent over the network.

# Use Us!

- Get Help!
- Contact the CAC – email a consultant or email help@cac.cornell.edu to submit a ticket.
- Find a bug or write a helper function that makes using the PCT easier?
    - Send it to me! I'll add your helper function to contrib and (try to) fix your bug and put the fix into a snapshot.