Parallel MATLAB at VT

Gene Cliff (AOE/ICAM - ecliff@vt.edu)
Justin Krometis (ARC/ICAM - jkrometis@vt.edu)
2:30 - 3:30pm, Wednesday, 16 September 2015
..... NLI .....

AOE: Department of Aerospace and Ocean Engineering
ARC: Advanced Research Computing

VirginiaTech

- **Introduction**
- Programming Models
- Execution
- Example: Quadrature
- Conclusion

INTRO: Parallel MATLAB

Parallel MATLAB is an extension of MATLAB that takes advantage of multicore desktop machines and clusters.

The *Parallel Computing Toolbox* or **PCT** runs on a desktop, and can take advantage of cores (R2014a has no limit, R2013b limit is 12, ...). Parallel programs can be run interactively or in batch.

The MATLAB *Distributed Computing Server* (**MDCS**) controls parallel execution of MATLAB on a cluster with tens or hundreds of cores.

ARC's clusters (Ithaca, BlueRidge, NewRiver) provides **MDCS** services for up to 224 cores. Currently, single users are restricted to 96 cores.

VirginiaTech

INTRO: What Do You Need?

1. Your machine should have multiple processors or cores:
   - On a PC: **Start** :: **Settings** :: **Control Panel** :: **System**
   - On a Mac: Apple Menu :: **About this Mac** :: **More Info...**
2. Your MATLAB must be **version 2012a** or later:
   - Go to the **HELP** menu, and choose **About Matlab**.
3. You must have the **Parallel Computing Toolbox**:
   - At VT, the concurrent (& student) license includes the PCT.
   - The *standalone* license does not include the PCT.
   - To list *all* your toolboxes, type the MATLAB command **ver**.
   - When using an **MDCS** (server) be sure to use the same version of MATLAB on your client machine.
   - **Ithaca** supports R2012a to R2015a. **BlueRidge** and **NewRiver** support R2015a.

VirginiaTech

- Introduction
- **Programming Models**
- Execution
- Example: Quadrature
- Conclusion

PROGRAMMING: Obtaining Parallelism

Three ways to write a parallel MATLAB program:

- suitable **for** loops can be made into **parfor** loops;
- the **spmd** statement can define cooperating synchronized processing;
- the **task** feature creates multiple independent programs.

The **parfor** approach is a limited but simple way to get started. **spmd** is powerful, but may require rethinking the program/data. The **task** approach is simple, but suitable only for computations that need almost no communication.

VirginiaTech

PROGRAMMING: PARFOR: Parallel FOR Loops

Lecture #2: PARFOR

The simplest path to parallelism is the **parfor** statement, which indicates that a given **for** loop can be executed in parallel.

When the "client" MATLAB reaches such a loop, the iterations of the loop are automatically divided up among the workers, and the results gathered back onto the client.

Using **parfor** requires that the iterations are completely independent; there are also some restrictions on array-data access.

OpenMP implements a `directive` for 'parallel for loops'

VirginiaTech

PROGRAMMING: "SPMD" Single Program Multiple Data

Lecture #3: SPMD

MATLAB can also work in a simplified kind of MPI model.

There is always a special "client" process.

Each worker process has its own memory and separate ID.

There is a single program, but it is divided into client and worker sections; the latter marked by special **spmd/end** statements.

Workers can "see" the client's data; the client can access and change worker data.

The workers can also send messages to other workers.

OpenMP includes constructs similar to **spmd**.

VirginiaTech

PROGRAMMING: "SPMD" Distributed Arrays

SPMD programming includes distributed arrays.

A distributed array is logically one array, and a large set of MATLAB commands can treat it that way (*e.g.* 'backslash').

However, portions of the array are scattered across multiple processors. This means such an array can be really large.

The local part of a distributed array can be operated on by that processor very quickly.

A distributed array can be operated on by explicit commands to the SPMD workers that "own" pieces of the array, or implicitly by commands at the global or client level.

VirginiaTech

- Introduction
- Programming Models
- **Execution**
- Example: Quadrature
- Conclusion

EXECUTION: Models

There are several ways to execute a parallel MATLAB program:

Model	Command	Where It Runs
Interactive	`matlabpool`	This machine
Interactive	`parpool (R2013b)`	This machine
Indirect local	`batch`	This machine
Indirect remote	`batch`	Remote machine

VirginiaTech

EXECUTION: Direct using `parpool`

Parallel MATLAB jobs can be run *directly*, that is, interactively.

The **parpool** (previously **matlabpool**) command is used to reserve a given number of workers on the local (or perhaps remote) machine.

Once these workers are available, the user can type commands, run scripts, or evaluate functions, which contain **parfor** statements. The workers will cooperate in producing results.

Interactive parallel execution is great for desktop debugging of short jobs.

Note: Starting in R2013b, if you try to execute a parallel program and a pool of workers is not already open, MATLAB will open it for you. The pool of workers will then remain open for a time that can be specified under Parallel $\rightarrow$ Parallel Preferences (default = 30 minutes).

VirginiaTech

EXECUTION: Indirect Local using `batch`

Parallel MATLAB jobs can be run indirectly.

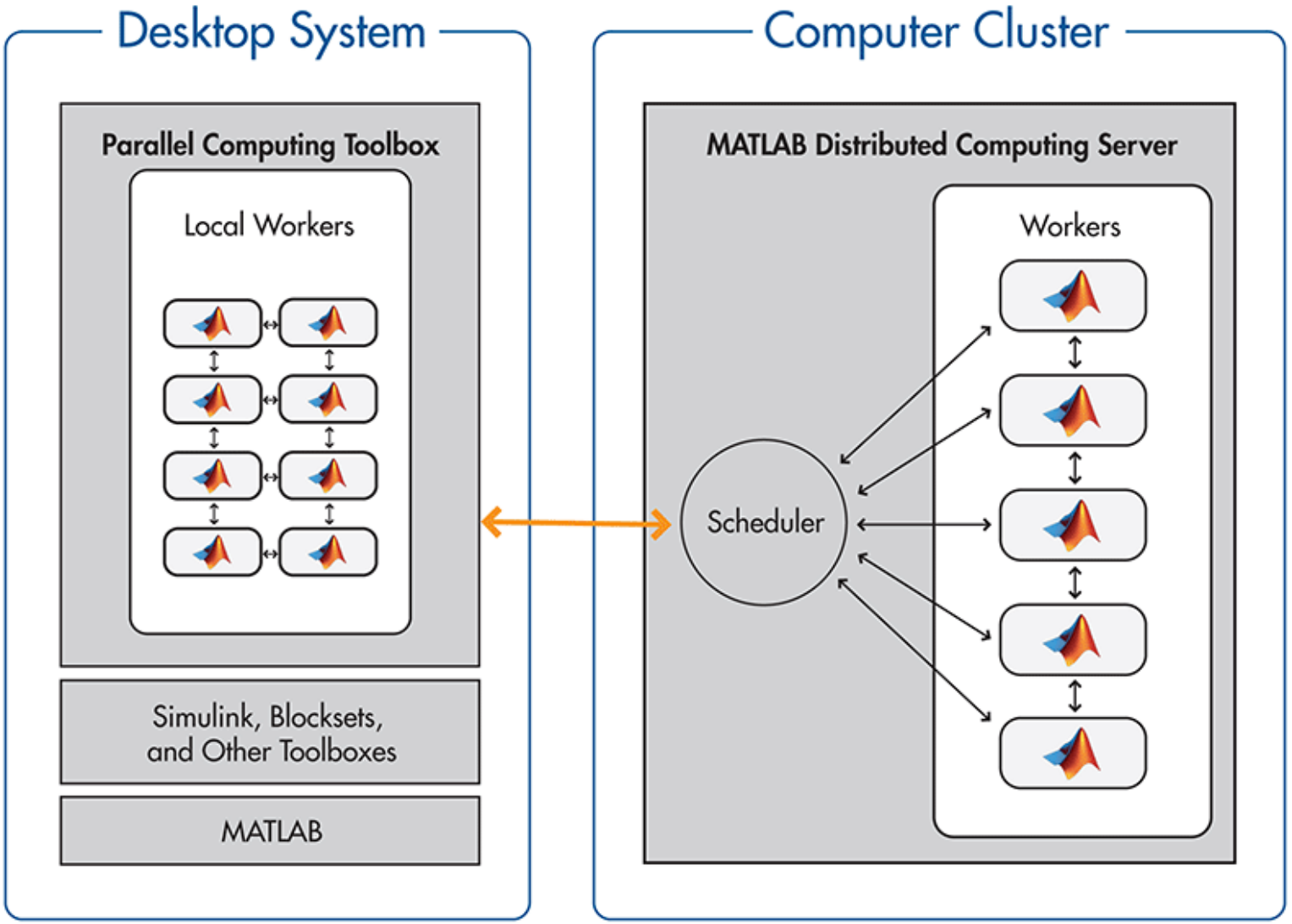
The **batch** command is used to specify a MATLAB code to be executed, to indicate any files that will be needed, and how many workers are requested.

The **batch** command starts the computation in the background. The user can work on other things, and collect the results when the job is completed.

The **batch** command works on the desktop, and can be set up to access ARC clusters (e.g. Ithaca).

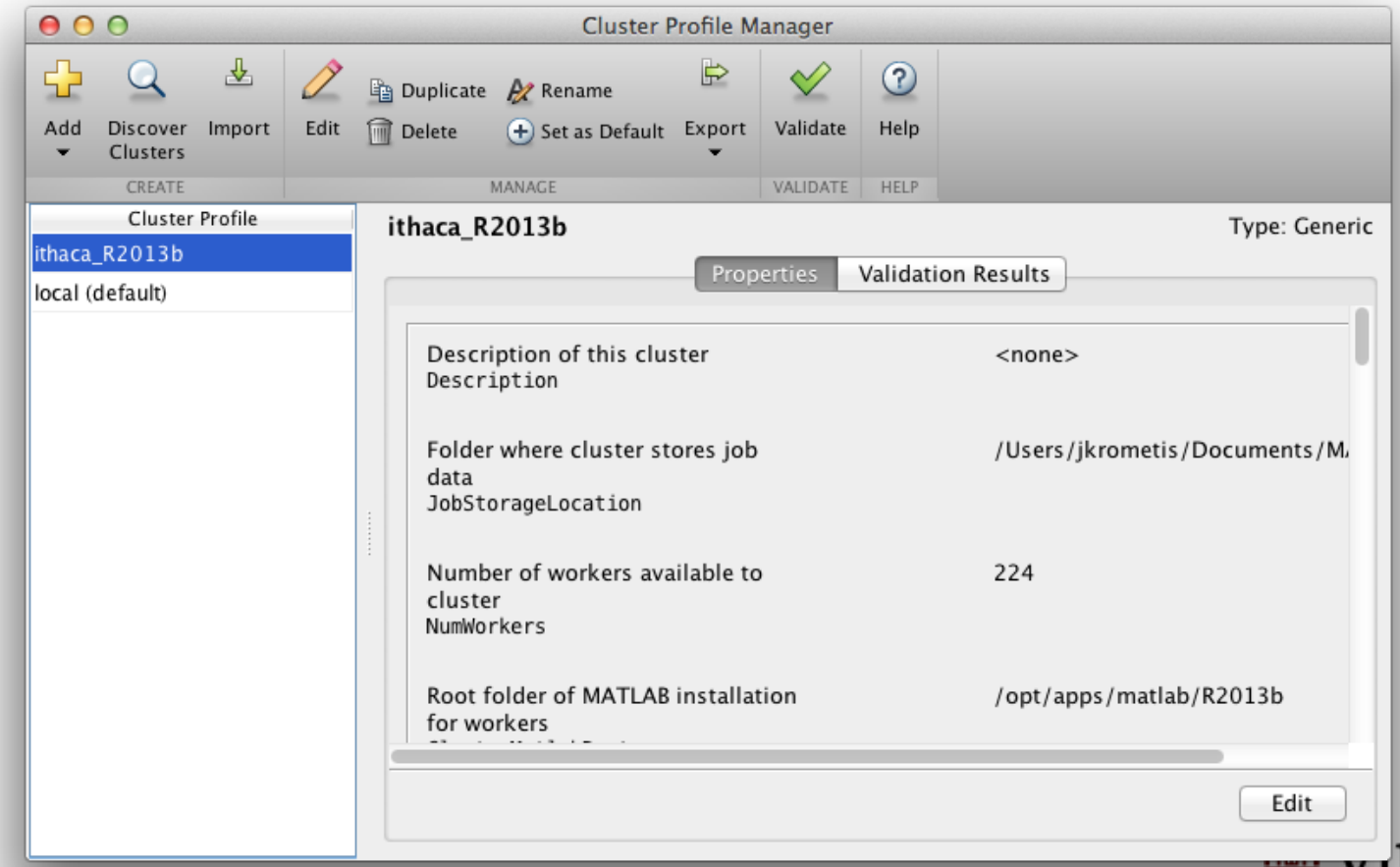
VirginiaTech

EXECUTION: Local and Remote MATLAB Workers



EXECUTION: Managing Cluster Profiles

MATLAB uses Cluster Profiles (previously called "configurations") to set the location of a job. 'local' is the default. Others can be added to send jobs to other clusters (e.g. Ithaca).



VirginiaTech

EXECUTION: Ways to Run

Interactively, we call **parpool** and then our function:

```
mypool = parpool ( 'local', 4 )
q = quad_fun ( n, a, b );
delete(mypool)
```

'local' is a default Cluster Profile defined as part of the PCT.
The **batch** command runs a script, with a **Pool** argument:

```
job = batch ( 'quad_script', 'Pool', 4 )
(or)
job = batch ( 'Profile','local', 'quad_script', ...
                                 'Pool', 4 )
```

VirginiaTech

EXECUTION: ARC Clusters

ARC offers resources with Matlab installed, including:

System	Usage	Nodes	Node Description	Special Features
Ithaca	Beginners, MATLAB	79	8 cores, 24GB (2× Intel Nehalem)	10 double-memory nodes
BlueRidge	Large-scale CPU, MIC	408	16 cores, 64 GB (2× Intel Sandy Bridge)	260 Intel Xeon Phi 4 K40 GPU 18 128GB nodes
NewRiver	Data Intensive	126	24 cores, 128 GB (2× Intel Haswell)	8 K80 GPGPU 16 "big data" nodes 24 512GB nodes 2 3TB nodes

ARC has a MDCS that can currently accommodate a combination of jobs with a total of 224 workers. At this time the queueing software imposes a limit of 96 workers per job.

VirginiaTech

EXECUTION: Configuring Desktop-to-Cluster Submission

- If you want to work with parallel MATLAB on ARC resources, you must first get an account. Go to

  `http://www.arc.vt.edu/account`

  Log in (PID and password), select the systems you want to work with and MATLAB in the Software section, and submit.

- Steps to set up submission from your desktop include:
  1. Download and add some files to your MATLAB directory
  2. Run a script to create a new profile on your desktop.

  A new cluster profile (e.g. `ithaca_R2015a`) will be created that can be used in `batch()`.
  These steps are described in detail here:

  `http://www.arc.vt.edu/matlabremote`

VirginiaTech

EXECUTION: Intracluster Submission

You can also submit jobs to ARC clusters from a cluster login node.

- Pros: Easier to set up. Only one file system to manage.
- Cons: Requires logging into the cluster (e.g., with SSH). Have to use Matlab command line (except on NewRiver).

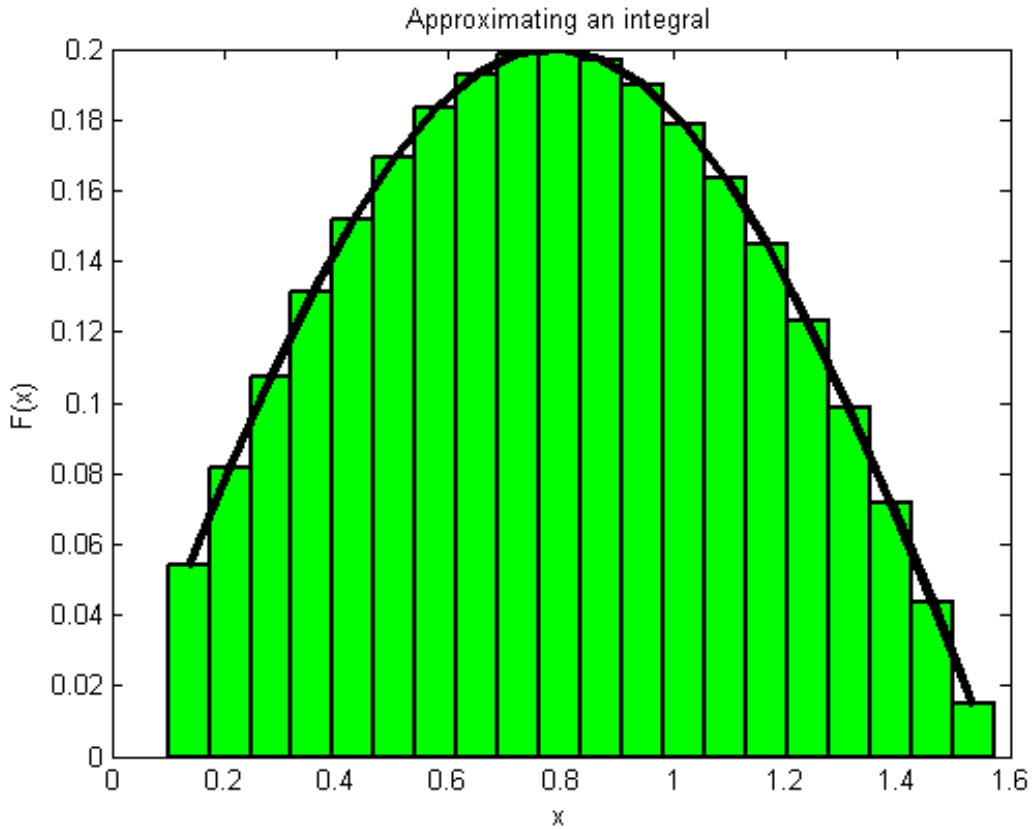
Setting up intracluster submission is very simple - running a one-question script at the Matlab command line on the ARC cluster.
The full steps are described here:

`http://www.arc.vt.edu/matlabremote#intracluster`

VirginiaTech

- Introduction
- Programming Models
- Execution
- **Example: Quadrature**
- Conclusion

QUAD: Estimating an Integral



VirginiaTech

QUAD: The QUAD_FUN Function

```
function q = quad_fun( n, a, b )

    q=0.0;
    w=(b-a)/n;
    for i=1:n
        x = ((n-i)*a+(i-1)*b)/(n-1);
        fx= 4./(1+x.^2);
        q = q+w*fx;
    end

    return
end
```

VirginiaTech

QUAD: Comments

The function **quad_fun** estimates the integral of a particular function over the interval $[a, b]$.

It does this by evaluating the function at n evenly spaced points, multiplying each value by the weight $(b - a)/n$.

These quantities can be regarded as the areas of little rectangles that lie under the curve, and their sum is an estimate for the total area under the curve from a to b.

We could compute these subareas **in any order we want**.

We could even compute the subareas at the same time, assuming there is some method to save the partial results and add them together in an organized way.

VirginiaTech

QUAD: The Parallel QUAD_FUN Function

```
function q = quad_fun( n, a, b )

    q=0.0;
    w=(b-a)/n;
%   for i=1:n  % avoid starting pool
    parfor i=1:n
        x = ((n-i)*a+(i-1)*b)/(n-1);
        fx= 4./(1+x.^2);
        q = q+w*fx;
    end

    return
end
```

VirginiaTech

QUAD: Comments

The parallel version of **quad_fun** does the same calculations.

The **parfor** statement changes how this program does the calculations. It asserts that all the iterations of the loop are independent, and can be done in any order, or in parallel.

Execution begins with a single processor, the client. When a **parfor** loop is encountered, the client is helped by a "pool" of workers.

Each worker is assigned some iterations of the loop. Once the loop is completed, the client resumes control of the execution.

MATLAB ensures that the results are the same (with exceptions) whether the program is executed sequentially, or with the help of workers.

The user can wait until execution time to specify how many workers are actually available.

VirginiaTech

QUAD: Interactive

To run *quad_fun.m* in parallel on your desktop, type:

```
n = 10000; a = 0.5; b = 1;
pool = parpool('local',4)
q = quad_fun ( n, a, b );
delete(pool)
```

The word **local** is choosing the local profile, that is, the cores assigned to be workers will be on the local machine.

The value "4" is the number of workers you are asking for. It can be up to 12 on a local machine. It does not have to match the number of cores you have.

VirginiaTech

QUAD: Indirect Local BATCH

The batch command, for indirect execution, accepts scripts (and since R2010b functions). We can make a suitable script called **quad_script.m**:

```
n = 10000; a = 0.5; b = 1;
q = quad_fun ( n, a, b )
```

Now we assemble the *job* information needed to run the script and submit the job:

```
job = batch ( 'quad_script', 'Pool', 4, ...
  'Profile', 'local', ...
  'AttachedFiles', { 'quad_fun' } )
```

VirginiaTech

QUAD: Indirect Local BATCH

After issuing `batch()`, the following commands wait for the job to finish, gather the results, and clear out the job information:

```
wait ( job );  % no prompt until the job is finished
load ( job );  % load data from the job's Workspace
delete ( job ); % clean up (destroy prior to R2012a)
```

VirginiaTech

QUAD: Indirect Remote BATCH

The batch command can send your job *anywhere*, and get the results back, as long as you have set up an account on the remote machine, and you have defined a Cluster Profile on your desktop that tells it how to access the remote machine.

At Virginia Tech, with proper set up, your desktop can send a batch job to an ARC cluster as easily as running locally:

```
job = batch ( 'quad_script', 'Pool', 4, ...
  'Profile', 'ithaca_R2015a', ...
  'AttachedFiles', { 'quad_fun' } )
```

The job is submitted. You may wait for it, load it and destroy/delete it, all in the same way as for a local batch job.

VirginiaTech

- Introduction
- Programming Models
- Execution
- Example: Quadrature
- **Conclusion**

CONCLUSION: Summary

- Introduction: Parallel Computing Toolbox
- Models of parallelism: parfor, spmd, distributed
- Models of execution: Interactive vs. Indirect, Local vs. Remote
  - ARC clusters
- Quadrature example: Parallelizing and Running

VirginiaTech

CONCLUSION: Desktop Experiments

Virginia Tech has a limited number of concurrent MATLAB licenses, and including the Parallel Computing Toolbox.

Since Fall 2011, the PCT is included with the student license.

Run `ver` in the Matlab Command Window to see what licenses you have available.

If you don't have a multicore machine, you won't see any speedup, but you may still be able to run some 'parallel'programs.

VirginiaTech

CONCLUSION: VT MATLAB LISTSERV

There is a local LISTSERV for people interested in MATLAB on the Virginia Tech campus. We try **not** to post messages here unless we really consider them of importance!

Important messages include information about workshops, special MATLAB events, and other issues affecting MATLAB users.

To subscribe to this email list, send a blank email to

`mathworks-g+subscribe@vt.edu`

The subject and body of the message should both be empty.

VirginiaTech

CONCLUSION: Where is it?

- MATLAB Parallel Computing Toolbox Product Documentation
  http://www.mathworks.com/help/toolbox/distcomp/
- Gaurav Sharma, Jos Martin,
  MATLAB: A Language for Parallel Computing, International Journal of Parallel Programming,
  Volume 37, Number 1, pages 3-36, February 2009.
- An ADOBE PDF with these notes, along with a zipped-folder containing the MATLAB codes can be downloaded from the ARC website at

  `http://www.arc.vt.edu/matlab#resources`

VirginiaTech

CONCLUSION: Upcoming Classes

1. Parallel Matlab II: Parfor
   (Wednesday, 21 October 2015, 9-10am, Torgersen 1100)

2. Parallel Matlab III: Single Program Multiple Data (SPMD)
   (Thursday, 5 November 2015, 3:30-4:30pm, Torgersen 1100)

VirginiaTech

THE END

Please complete the evaluation form

Thanks

VirginiaTech