UCL Université catholique de Louvain

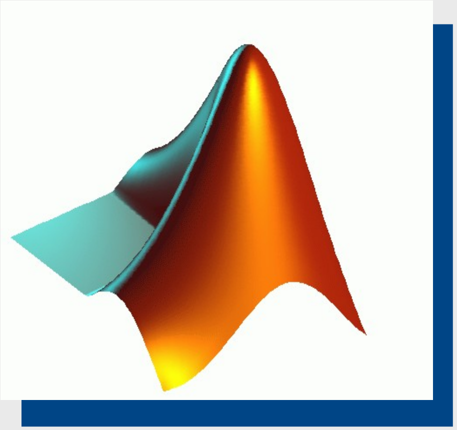
INSTITUT DE CALCUL INTENSIF ET DE STOCKAGE DE MASSE

Efficient use of Matlab on the clusters

C.E.C.I

Consortium des Equipements
de Calcul Intensif
en Fédération Wallonie-Bruxelles

Typical usage...



Interactive ⟺ Batch

Type in and get an answer

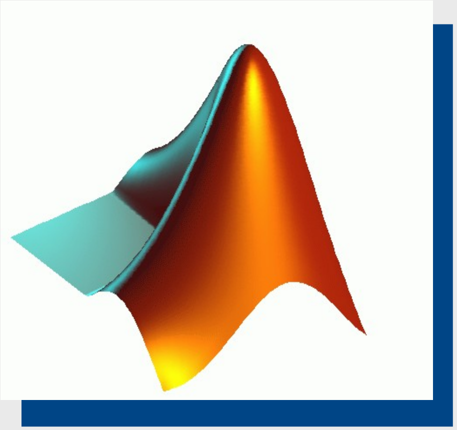
Submit job and fetch results

Sequential ⟺ Parallel

Perform tasks one after the other

Perform multiple tasks at the same time

Typical usage...



Interactive ⟺ Batch

Type in and get an answer

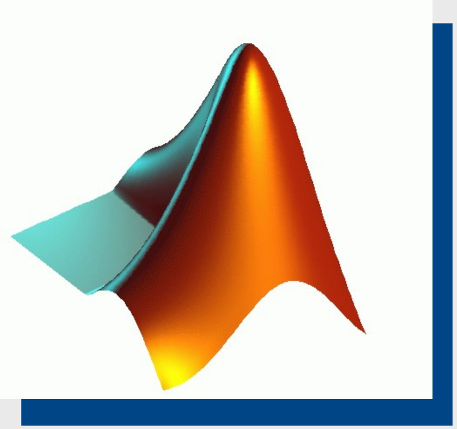
Submit job and fetch results

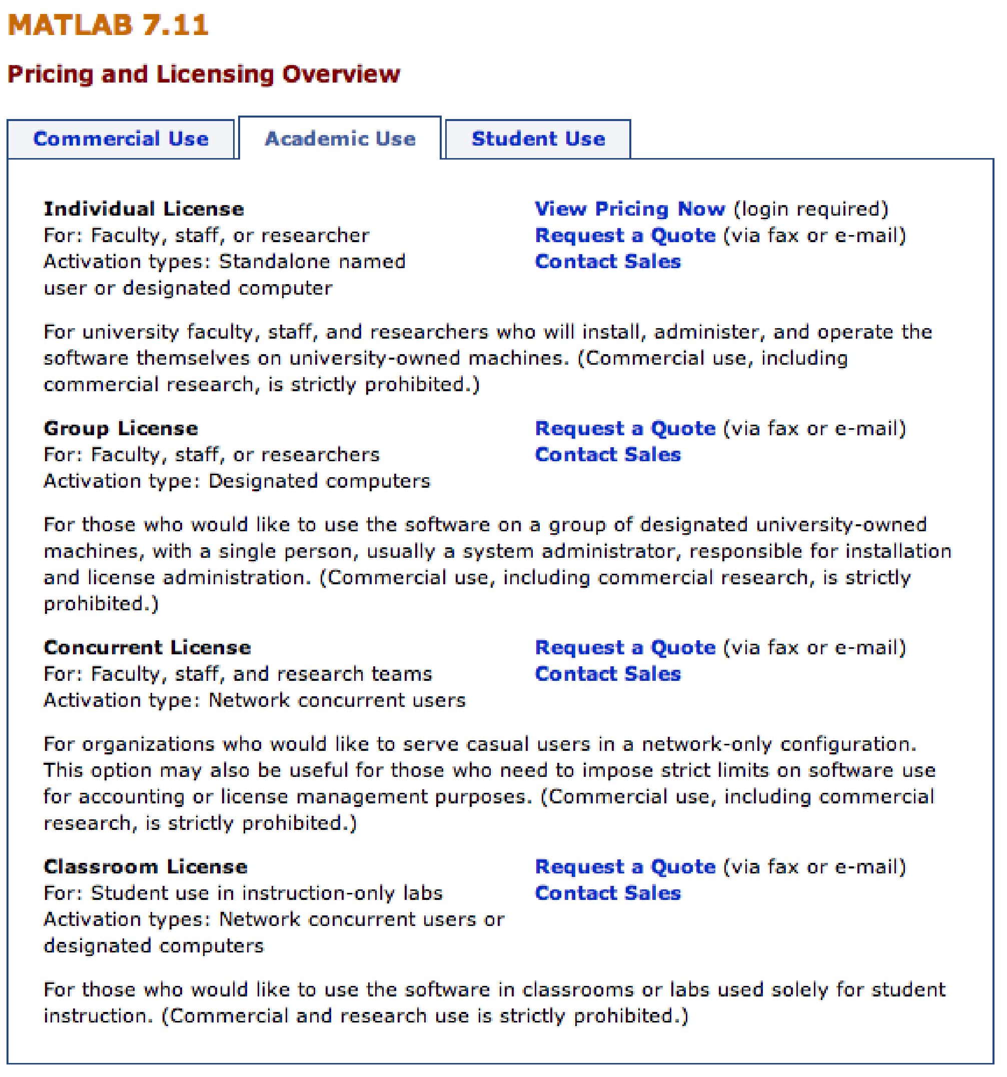
Sequential ⟺ Parallel

Perform tasks one after the other

Perform multiple tasks at the same time

CISM One more obstacle: Matlab Licensing





MATLAB 7.11

Pricing and Licensing Overview

Individual License
For: Faculty, staff, or researcher
Activation types: Standalone named user or designated computer

View Pricing Now (login required)
Request a Quote (via fax or e-mail)
Contact Sales

For university faculty, staff, and researchers who will install, administer, and operate the software themselves on university-owned machines. (Commercial use, including commercial research, is strictly prohibited.)

Group License
For: Faculty, staff, or researchers
Activation type: Designated computers

Request a Quote (via fax or e-mail)
Contact Sales

For those who would like to use the software on a group of designated university-owned machines, with a single person, usually a system administrator, responsible for installation and license administration. (Commercial use, including commercial research, is strictly prohibited.)

Concurrent License
For: Faculty, staff, and research teams
Activation type: Network concurrent users

Request a Quote (via fax or e-mail)
Contact Sales

For organizations who would like to serve casual users in a network-only configuration. This option may also be useful for those who need to impose strict limits on software use for accounting or license management purposes. (Commercial use, including commercial research, is strictly prohibited.)

Classroom License
For: Student use in instruction-only labs
Activation types: Network concurrent users or designated computers

Request a Quote (via fax or e-mail)
Contact Sales

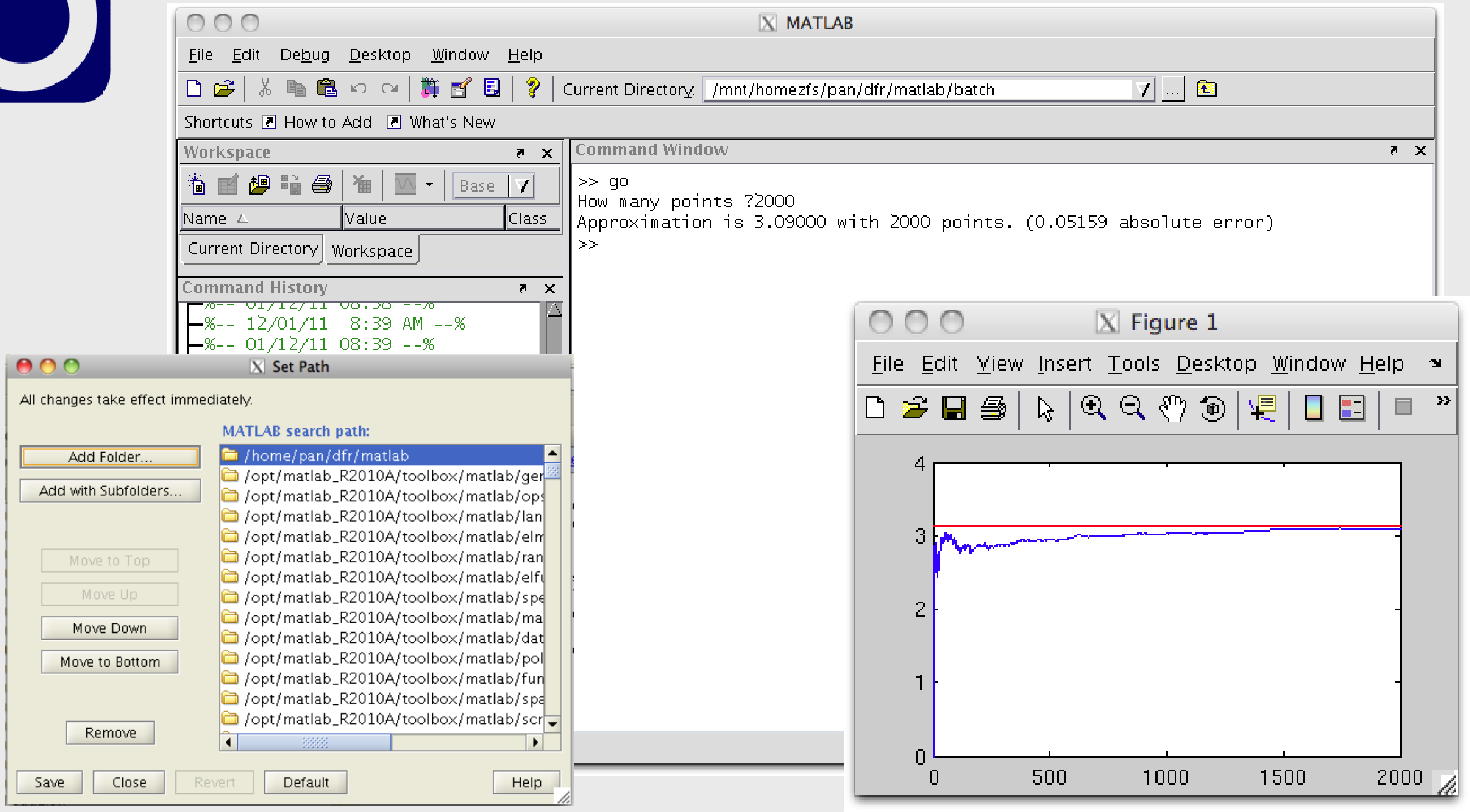
For those who would like to use the software in classrooms or labs used solely for student instruction. (Commercial and research use is strictly prohibited.)

Parallel Matlab on the cluster

- Using Matlab in batch mode
  - With Matlab (e.g. your computer or CeSAM)
  - Without Matlab (e.g. the clusters)
- Using Matlab in parallel
  - With no effort
  - With little effort
  - With a lot of effort

Using Matlab in Batch mode

You might be used to ...



Current Directory: /mnt/homezfs/pan/dfr/matlab/batch

```
%-- 01/12/11 08:38 --%
%-- 12/01/11  8:39 AM --%
%-- 01/12/11 08:39 --%
```

```
>> go
How many points ?2000
Approximation is 3.09000 with 2000 points. (0.05159 absolute error)
>>
```

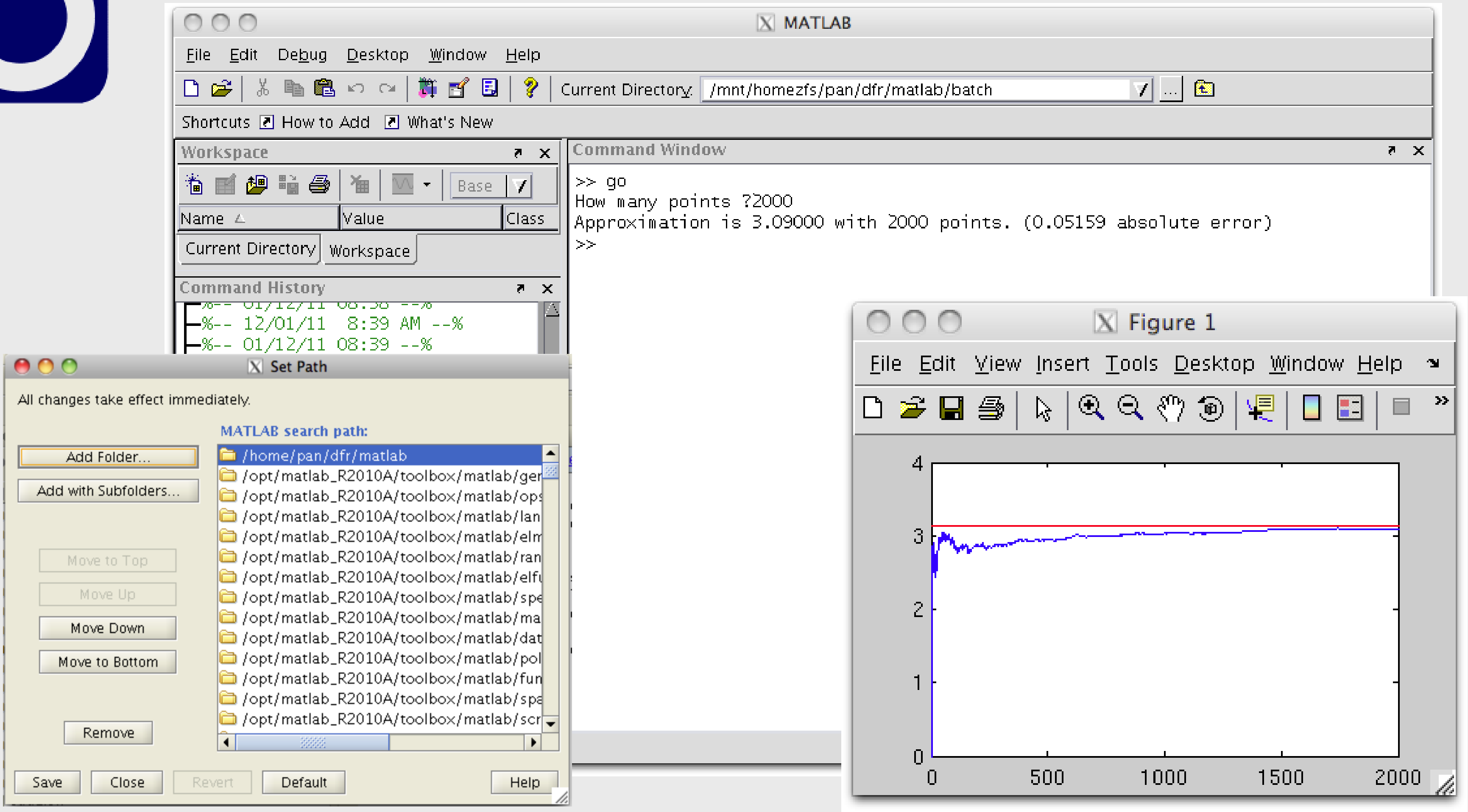
All changes take effect immediately.

MATLAB search path:

```
/home/pan/dfr/matlab
/opt/matlab_R2010A/toolbox/matlab/ger
/opt/matlab_R2010A/toolbox/matlab/ops
/opt/matlab_R2010A/toolbox/matlab/lan
/opt/matlab_R2010A/toolbox/matlab/elm
/opt/matlab_R2010A/toolbox/matlab/ran
/opt/matlab_R2010A/toolbox/matlab/elfu
/opt/matlab_R2010A/toolbox/matlab/spe
/opt/matlab_R2010A/toolbox/matlab/ma
/opt/matlab_R2010A/toolbox/matlab/dat
/opt/matlab_R2010A/toolbox/matlab/pol
/opt/matlab_R2010A/toolbox/matlab/fun
/opt/matlab_R2010A/toolbox/matlab/spa
/opt/matlab_R2010A/toolbox/matlab/scr
```

Using Matlab in Batch mode

... but no GUI in batch mode !



Current Directory: /mnt/homezfs/pan/dfr/matlab/batch

```
>> go
How many points ?2000
Approximation is 3.09000 with 2000 points. (0.05159 absolute error)
>>
```

```
%-- 01/12/11 08:38 --%
%-- 12/01/11  8:39 AM --%
%-- 01/12/11 08:39 --%
```

All changes take effect immediately.

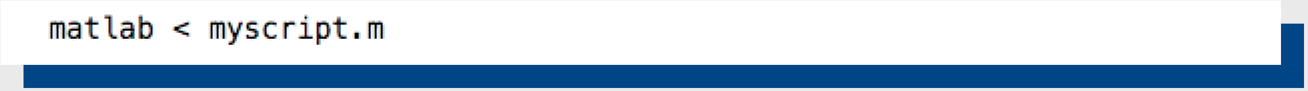
MATLAB search path:

/home/pan/dfr/matlab
/opt/matlab_R2010A/toolbox/matlab/ger
/opt/matlab_R2010A/toolbox/matlab/ops
/opt/matlab_R2010A/toolbox/matlab/lan
/opt/matlab_R2010A/toolbox/matlab/elm
/opt/matlab_R2010A/toolbox/matlab/ran
/opt/matlab_R2010A/toolbox/matlab/elfu
/opt/matlab_R2010A/toolbox/matlab/spe
/opt/matlab_R2010A/toolbox/matlab/ma
/opt/matlab_R2010A/toolbox/matlab/dat
/opt/matlab_R2010A/toolbox/matlab/pol
/opt/matlab_R2010A/toolbox/matlab/fun
/opt/matlab_R2010A/toolbox/matlab/spa
/opt/matlab_R2010A/toolbox/matlab/scr

Using Matlab in Batch mode

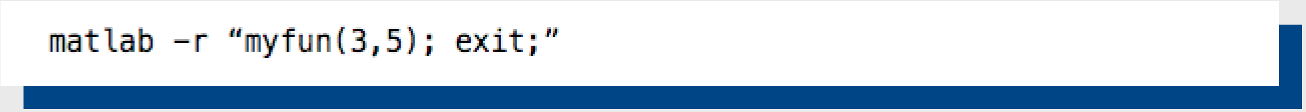
Two methods for launching a script:

You have a script myscript.m:



```
matlab < myscript.m
```

You have a function "function a = myfun(x,y) ..."



```
matlab -r “myfun(3,5); exit;”
```

CISM

Using Matlab in Batch mode

Paths

Data

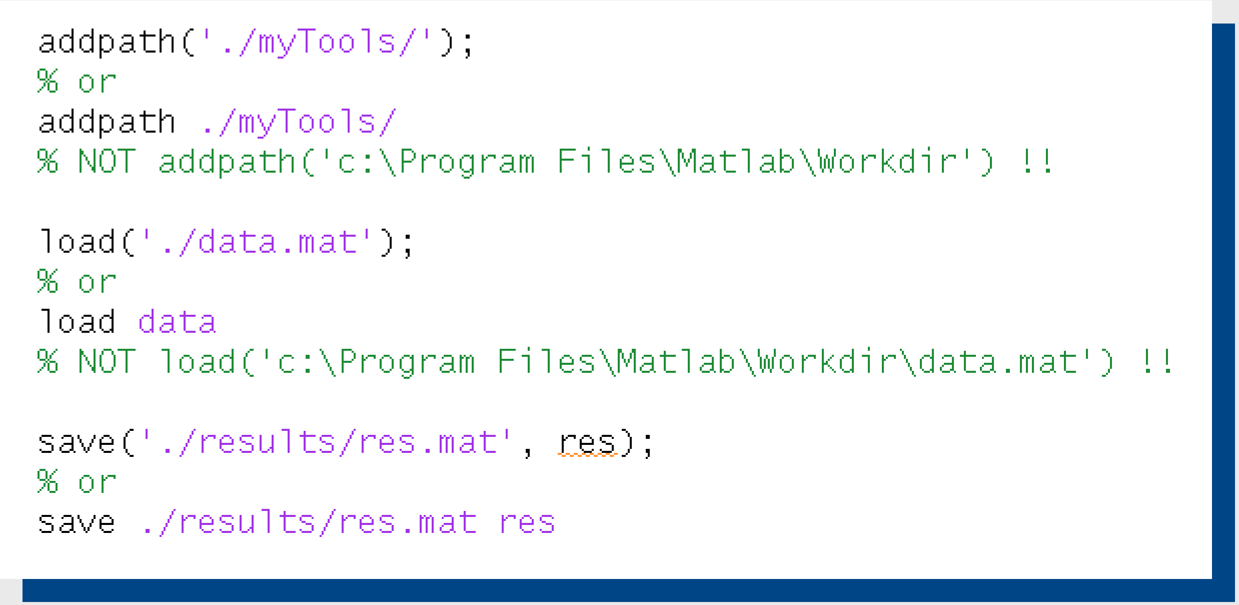
Prompts

Figures

GUI

You need to adapt your Matlab script

No more clicks!



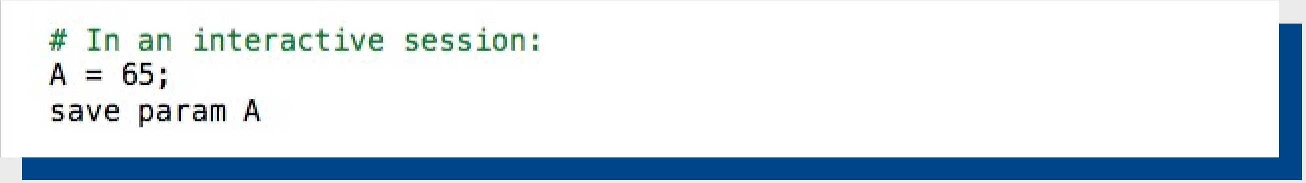
```
addpath('./myTools/');
% or
addpath ./myTools/
% NOT addpath('c:\Program Files\Matlab\Workdir') !!

load('./data.mat');
% or
load data
% NOT load('c:\Program Files\Matlab\Workdir\data.mat') !!

save('./results/res.mat', res);
% or
save ./results/res.mat res
```

Make sure to automate any setup that you usually do by hand: adding paths, loading data, saving results, etc.

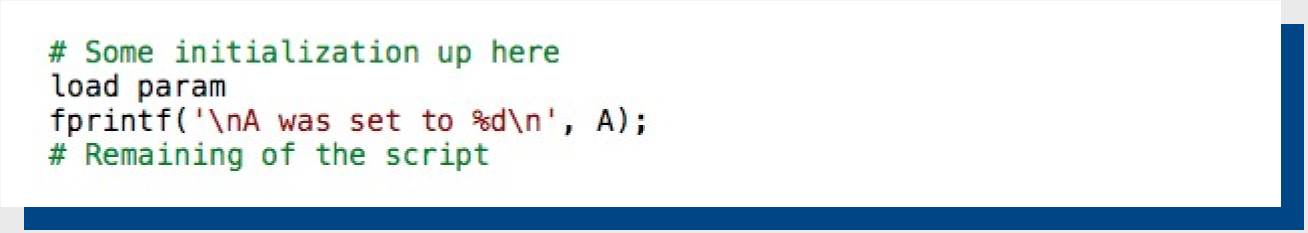
You need to adapt your Matlab script

Put all 'configuration' values in a file and load it

- Paths
- Data
- Prompts
- Figures
- GUI



```
# In an interactive session:
A = 65;
save param A
```



```
# Some initialization up here
load param
fprintf('\nA was set to %d\n', A);
# Remaining of the script
```

Using Matlab in Batch mode

You need to adapt your Matlab script

```
x = -pi:pi/10:pi;
y = tan(sin(x)) - sin(tan(x));
plot(x,y,'--rs','LineWidth',2,...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
                'MarkerSize',10)
print -dpng plot.png
```

Paths

Data

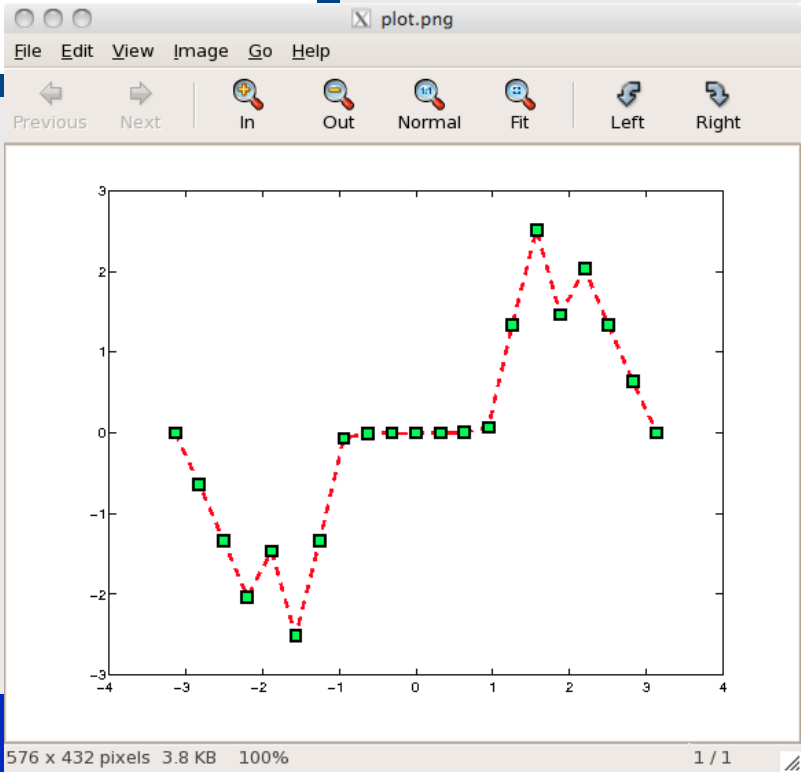
Prompts

Figures

GUI

Use 'print' to put your graphics to a file.

Or avoid plotting at all and do that interactively at postprocess time



CISM

Using Matlab in Batch mode

You need to adapt your Matlab script

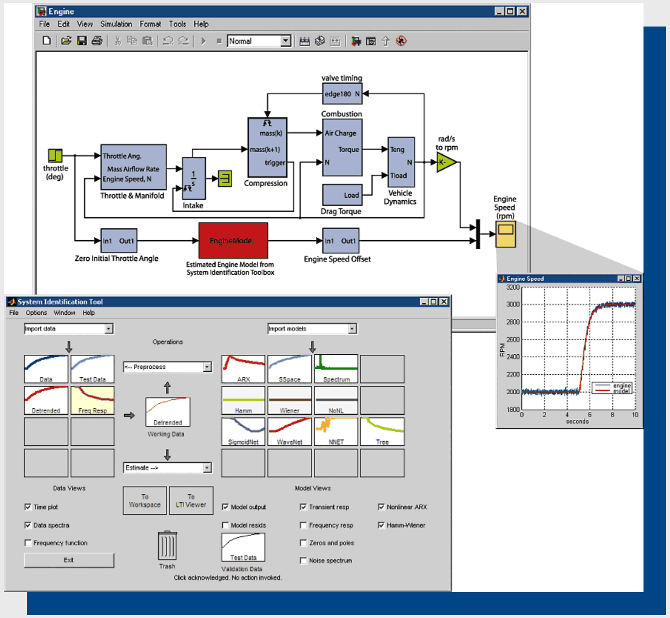
Paths

Data

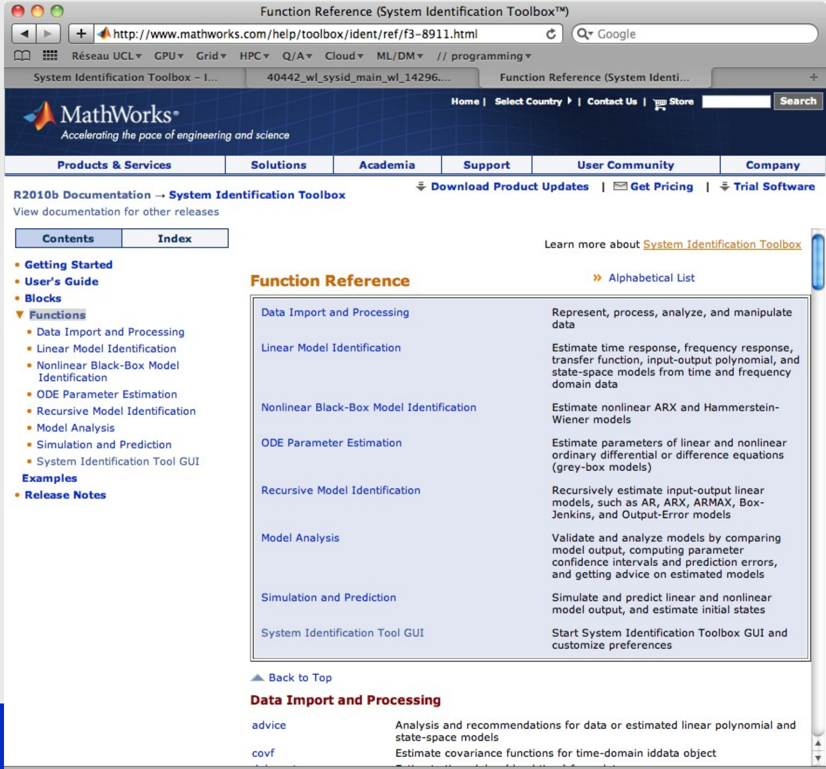
Prompts

Figures

GUI



Simplest solution: avoid GUI's and use command line version


Function Reference (System Identification Toolbox™)

http://www.mathworks.com/help/toolbox/ident/ref/f3-8911.html

Google

Réseau UCL▾ GPU▾ Grid▾ HPC▾ Q/A▾ Cloud▾ ML/DM▾ // programming▾

System Identification Toolbox – I... | 40442_wl_sysid_main_wl_14296.... | Function Reference (System Identi...

MathWorks

Accelerating the pace of engineering and science

Home | Select Country | Contact Us | Store | Search

Products & Services | Solutions | Academia | Support | User Community | Company

R2010b Documentation → System Identification Toolbox

View documentation for other releases

Download Product Updates | Get Pricing | Trial Software

Contents | Index

Learn more about System Identification Toolbox

- Getting Started
- User's Guide
- Blocks
- Functions
  - Data Import and Processing
  - Linear Model Identification
  - Nonlinear Black-Box Model Identification
  - ODE Parameter Estimation
  - Recursive Model Identification
  - Model Analysis
  - Simulation and Prediction
  - System Identification Tool GUI
- Examples
- Release Notes

Function Reference

» Alphabetical List

Data Import and Processing	Represent, process, analyze, and manipulate data
Linear Model Identification	Estimate time response, frequency response, transfer function, input-output polynomial, and state-space models from time and frequency domain data
Nonlinear Black-Box Model Identification	Estimate nonlinear ARX and Hammerstein-Wiener models
ODE Parameter Estimation	Estimate parameters of linear and nonlinear ordinary differential or difference equations (grey-box models)
Recursive Model Identification	Recursively estimate input-output linear models, such as AR, ARX, ARMAX, Box-Jenkins, and Output-Error models
Model Analysis	Validate and analyze models by comparing model output, computing parameter confidence intervals and prediction errors, and getting advice on estimated models
Simulation and Prediction	Simulate and predict linear and nonlinear model output, and estimate initial states
System Identification Tool GUI	Start System Identification Toolbox GUI and customize preferences

Back to Top

Data Import and Processing

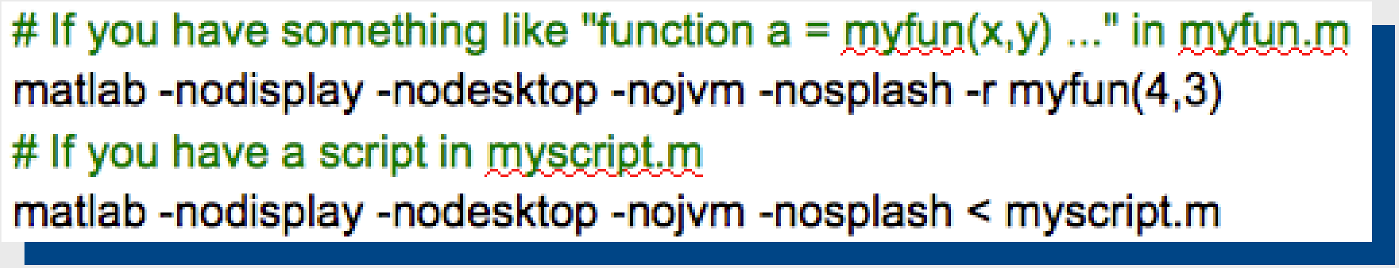
advice	Analysis and recommendations for data or estimated linear polynomial and state-space models
covf	Estimate covariance functions for time-domain iddata object

Using Matlab in Batch mode

And then launch it !

Options

Launch



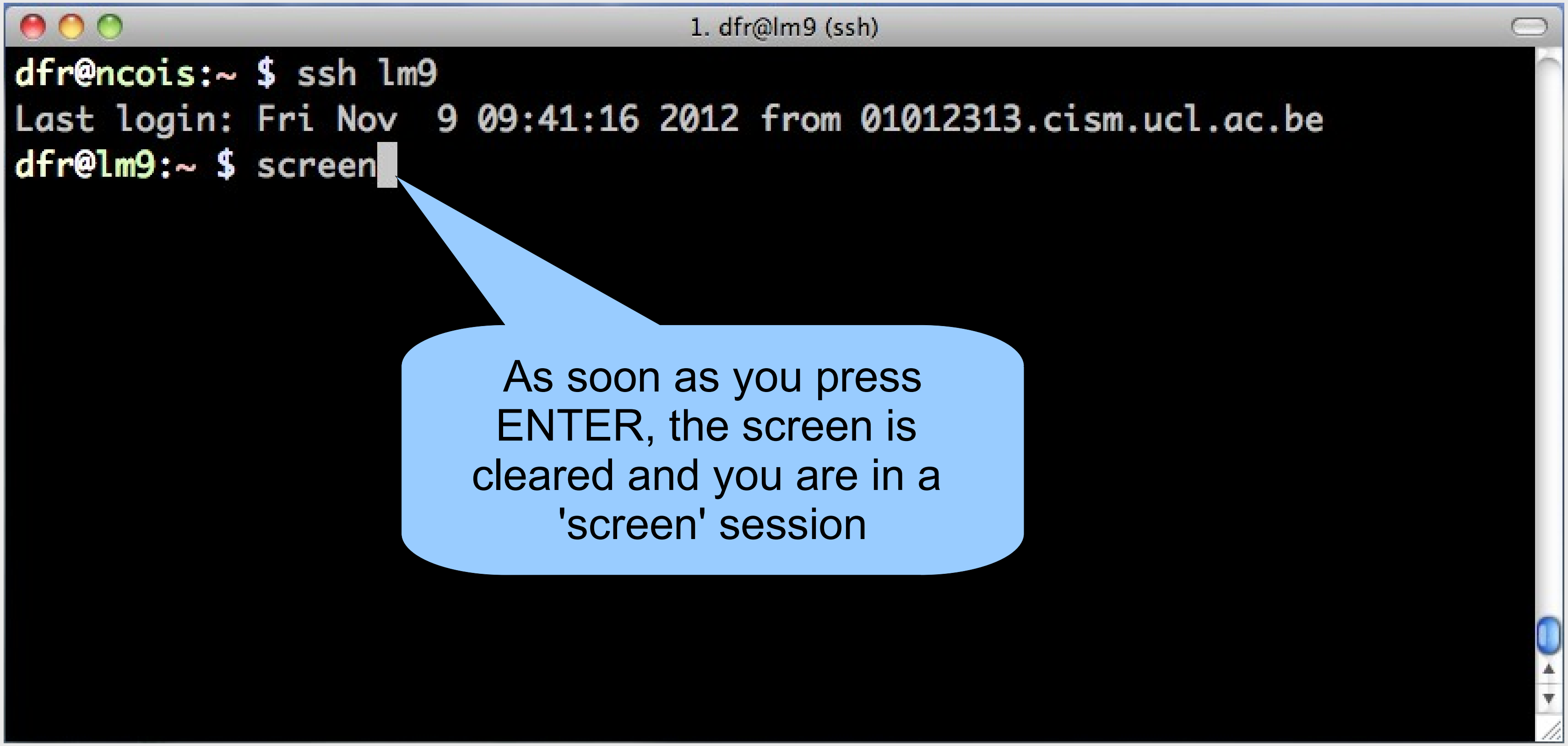
```
# If you have something like "function a = myfun(x,y) ..." in myfun.m
matlab -nodisplay -nodesktop -nojvm -nosplash -r myfun(4,3)
# If you have a script in myscript.m
matlab -nodisplay -nodesktop -nojvm -nosplash < myscript.m
```

-nodisplay:	do not try to display plots
-nodesktop:	do not launch full GUI
-nojvm:	do not launch Java support (do not use in recent versions of Matlab)
-nosplash:	do not display splashscreen

CISM

Using Matlab in Batch mode

Use 'screen' for unattended execution

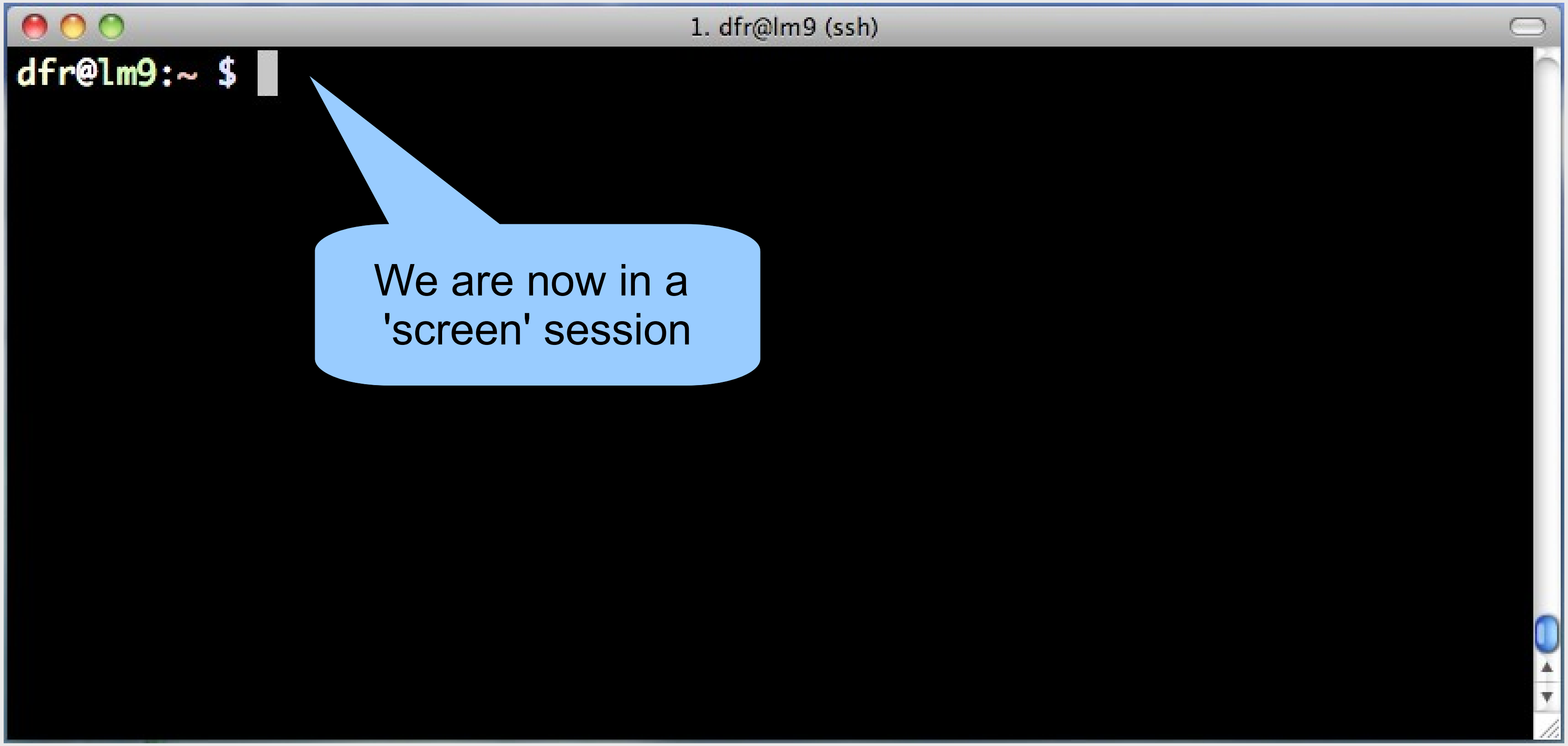


```
dfr@ncois:~ $ ssh lm9
Last login: Fri Nov  9 09:41:16 2012 from 01012313.cism.ucl.ac.be
dfr@lm9:~ $ screen
```

As soon as you press ENTER, the screen is cleared and you are in a 'screen' session

Using Matlab in Batch mode

Use 'screen' for unattended execution

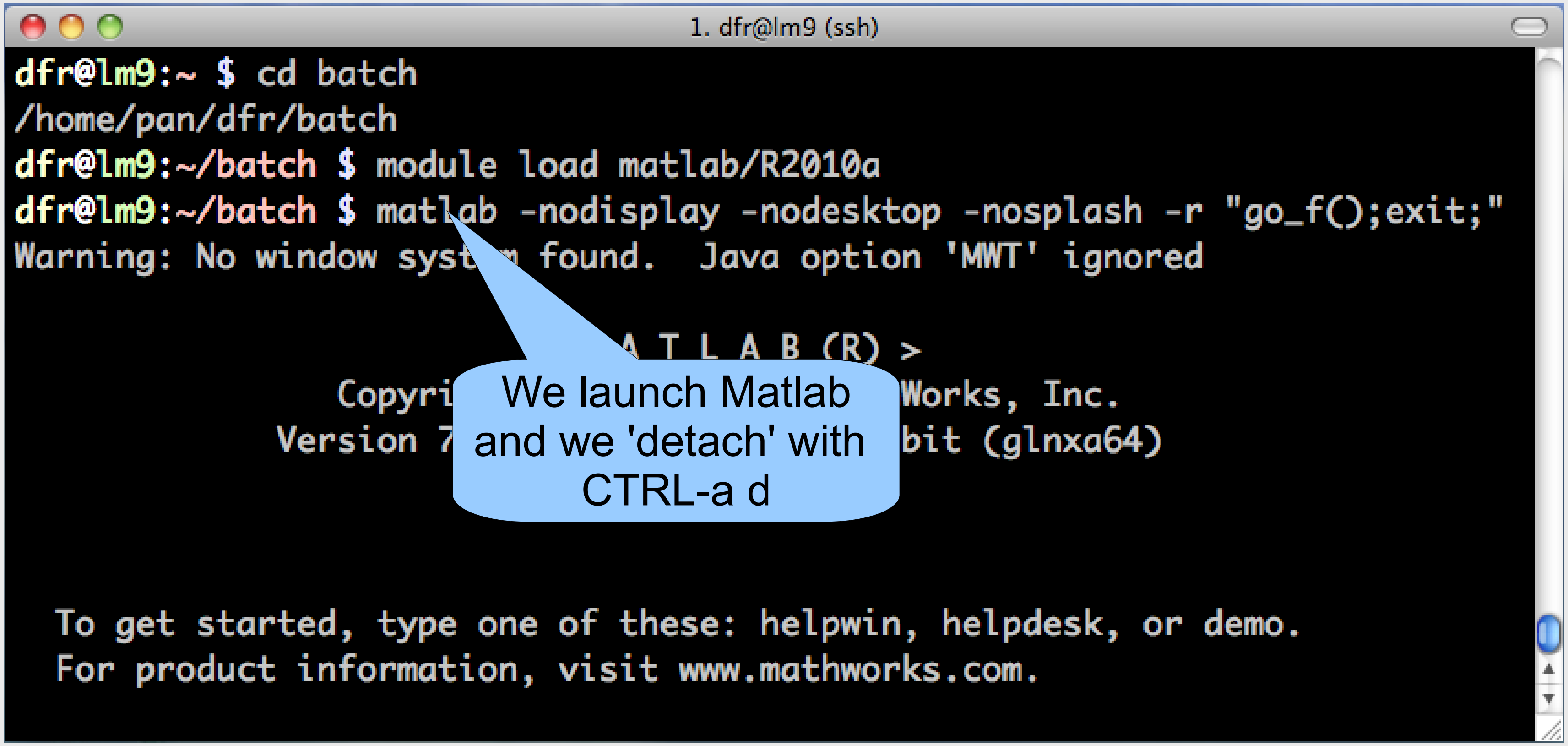


```
dfr@lm9:~ $ 
```

We are now in a 'screen' session

Using Matlab in Batch mode

Use 'screen' for unattended execution

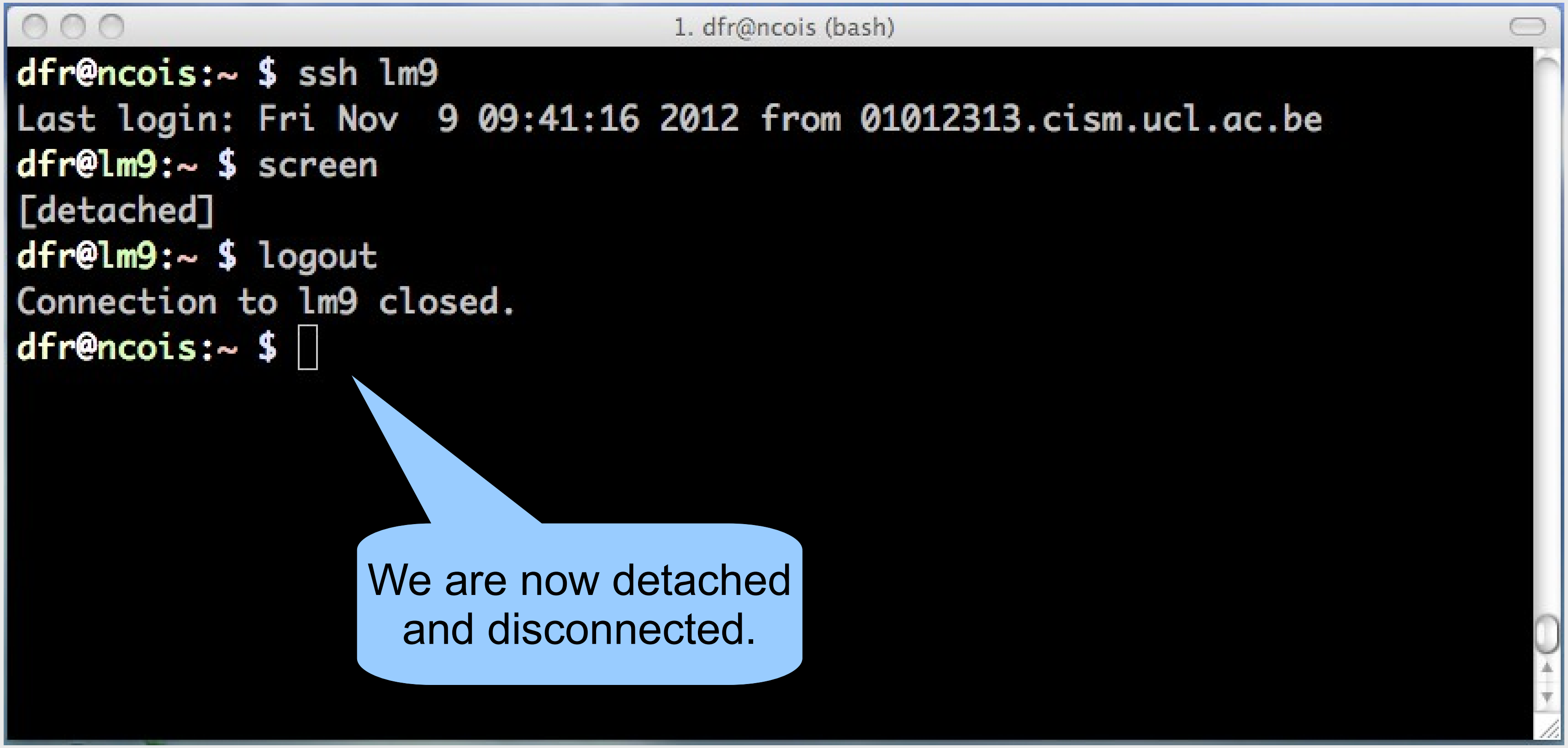


```
dfr@lm9:~ $ cd batch
/home/pan/dfr/batch
dfr@lm9:~/batch $ module load matlab/R2010a
dfr@lm9:~/batch $ matlab -nodisplay -nodesktop -nosplash -r "go_f();exit;"
Warning: No window syst  m found.  Java option 'MWT' ignored

                              A T L A B (R) >
                Copyri                    Works, Inc.
             Version 7                    bit (glnxa64)



  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, visit www.mathworks.com.
```

We launch Matlab and we 'detach' with CTRL-a d

Using Matlab in Batch mode

Use 'screen' for unattended execution

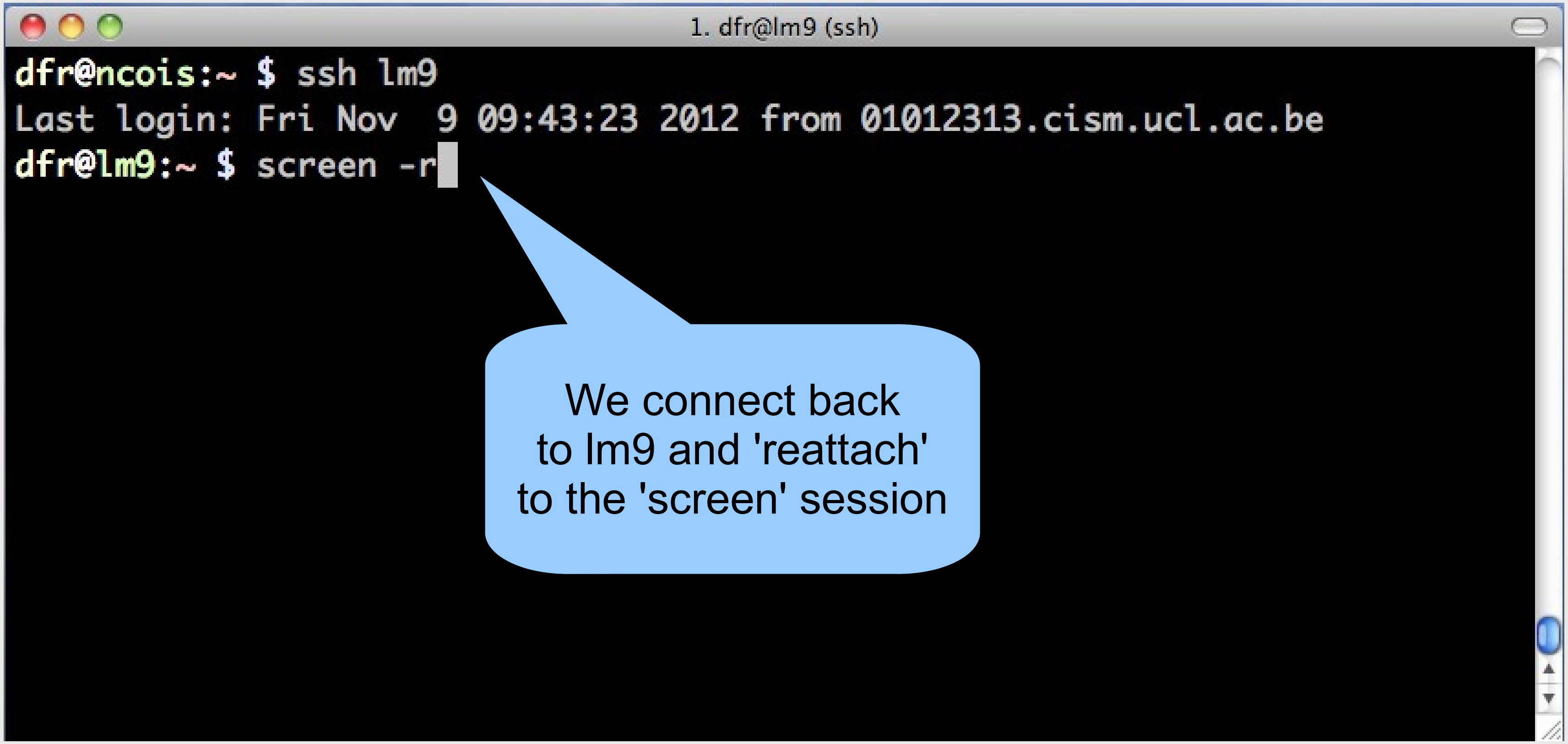


```
dfr@ncois:~ $ ssh lm9
Last login: Fri Nov  9 09:41:16 2012 from 01012313.cism.ucl.ac.be
dfr@lm9:~ $ screen
[detached]
dfr@lm9:~ $ logout
Connection to lm9 closed.
dfr@ncois:~ $ 
```

We are now detached and disconnected.

Using Matlab in Batch mode

Use 'screen' for unattended execution

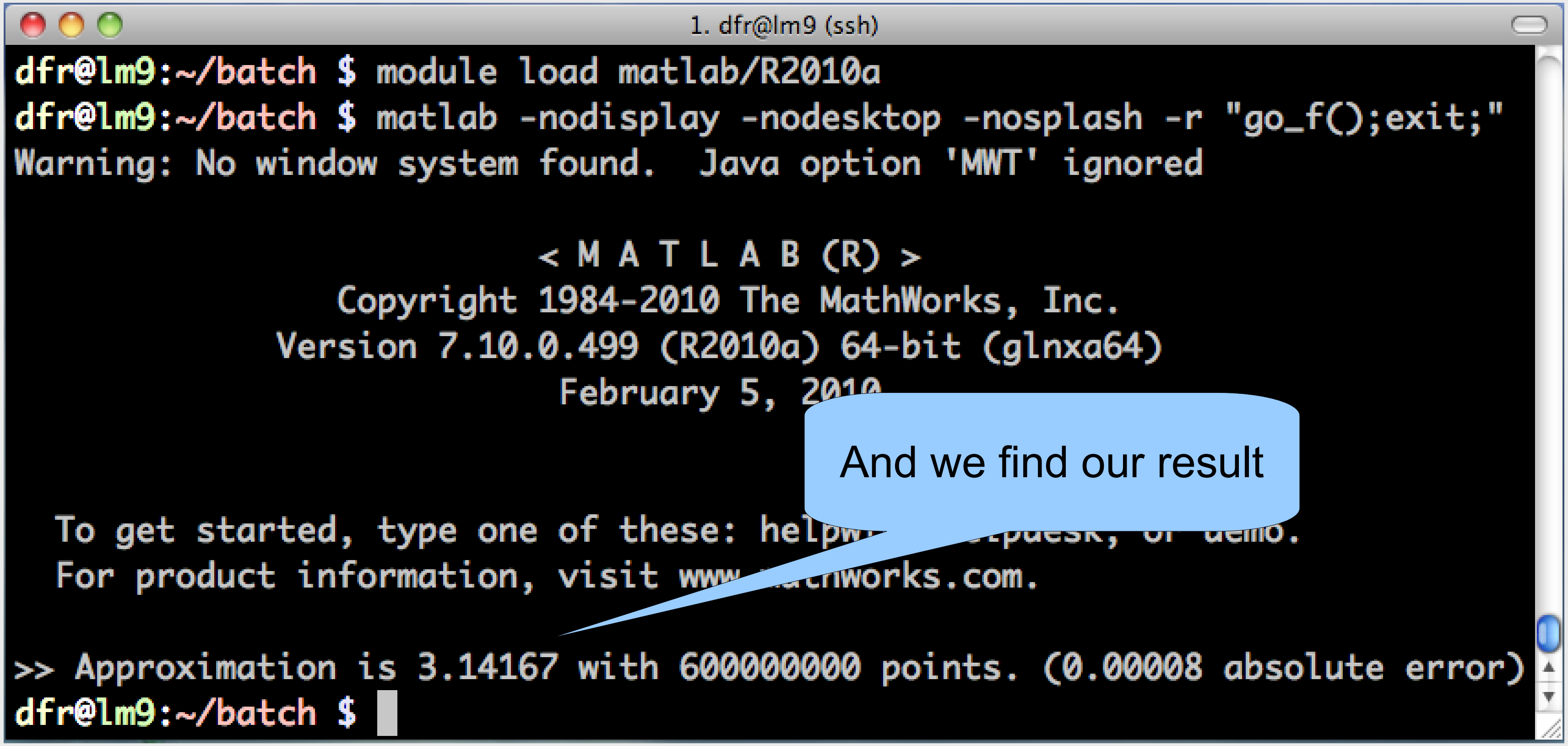


```
dfr@ncois:~ $ ssh lm9
Last login: Fri Nov  9 09:43:23 2012 from 01012313.cism.ucl.ac.be
dfr@lm9:~ $ screen -r
```

We connect back
to lm9 and 'reattach'
to the 'screen' session

Using Matlab in Batch mode

Use 'screen' for unattended execution



```
dfr@lm9:~/batch $ module load matlab/R2010a
dfr@lm9:~/batch $ matlab -nodisplay -nodesktop -nosplash -r "go_f();exit;"
Warning: No window system found.  Java option 'MWT' ignored

                          < M A T L A B (R) >
                Copyright 1984-2010 The MathWorks, Inc.
             Version 7.10.0.499 (R2010a) 64-bit (glnxa64)
                           February 5, 2010


  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, visit www.mathworks.com.

>> Approximation is 3.14167 with 600000000 points. (0.00008 absolute error)
dfr@lm9:~/batch $ 
```

And we find our result

Your turn..

1. Connect to
    cesam.cism.ucl.ac.be, or
    brufence.cism.ucl.ac.be with your CISM login (or ask ‘tuto’ login)
  with X11 forwarding

2. Copy directory ~dfr/matlab/batch to your directory and “cd” there

3. Load the Matlab module (matlab or Matlab or MATLAB, please check)
3. Launch Matlab

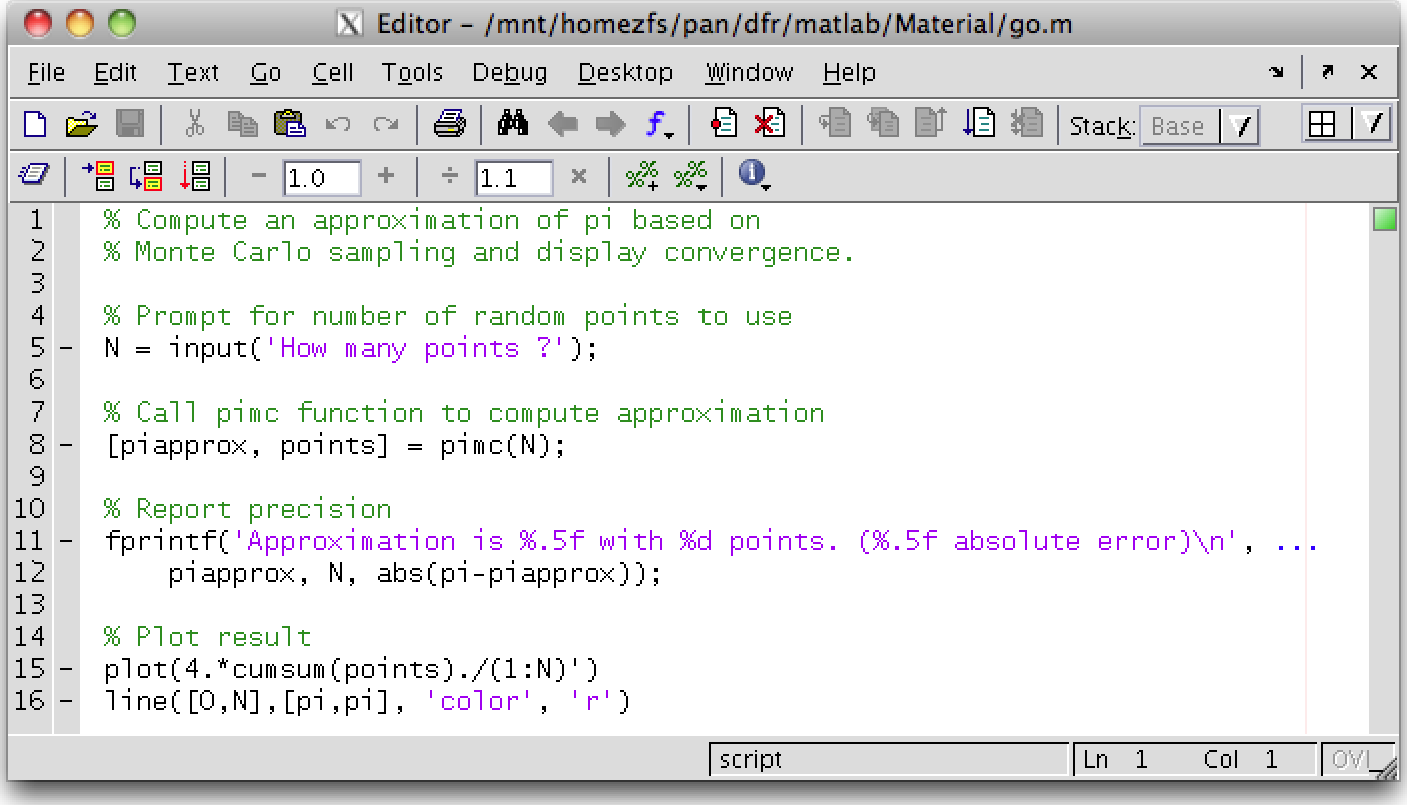
4. Run 'go' to see what it does
    (you might have to set the paths: File > Set Paths
                  or Home > Environment > Set Path)

5. Edit go.m so as to be able to run it in batch
6. Quit Matlab

7. Test your Matlab script in 'batch mode'

8. Make a longer test with “screen”

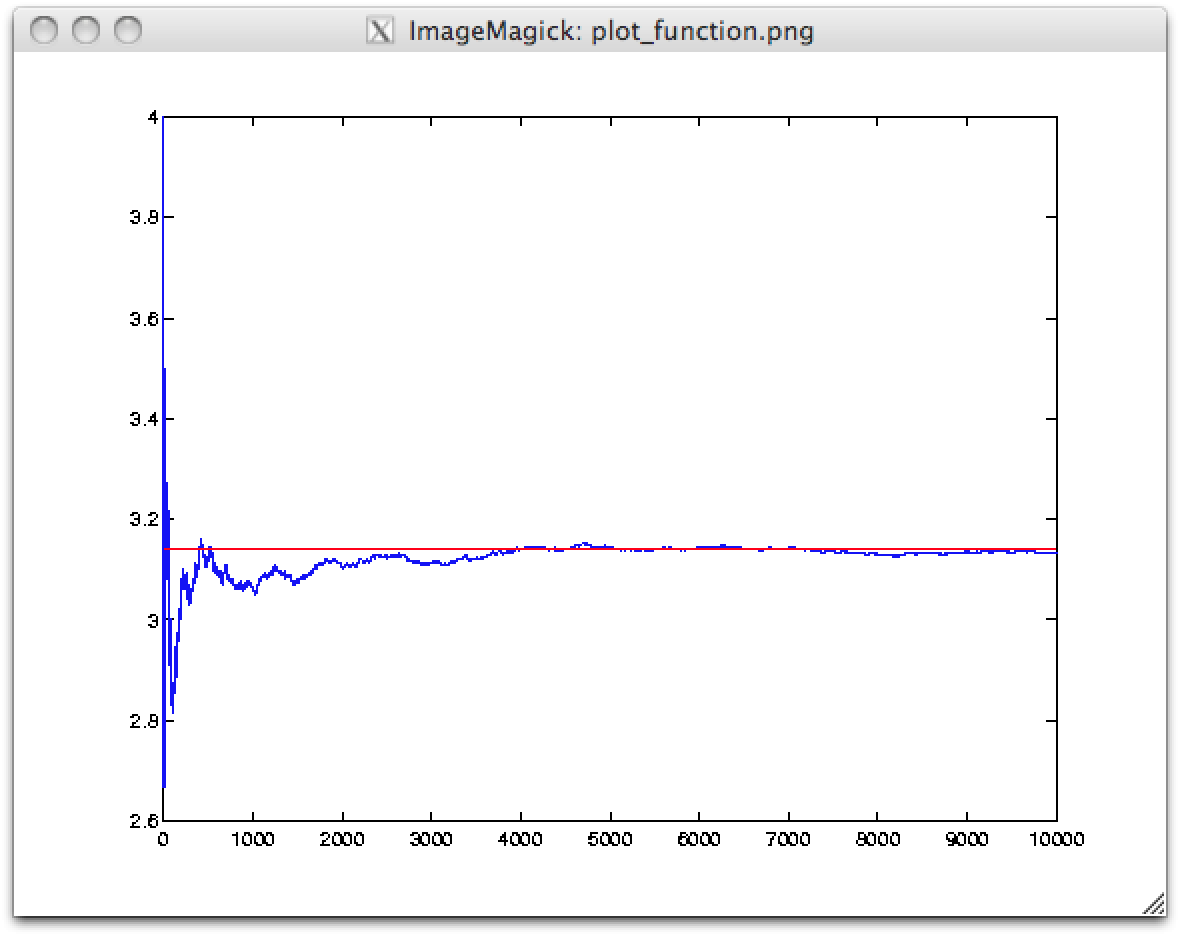
Your turn..



Editor - /mnt/homezfs/pan/dfr/matlab/Material/go.m
File Edit Text Go Cell Tools Debug Desktop Window Help
Stack: Base
1.0
1.1
script
Ln 1 Col 1
OVR

```
% Compute an approximation of pi based on
% Monte Carlo sampling and display convergence.

% Prompt for number of random points to use
N = input('How many points ?');

% Call pimc function to compute approximation
[piapprox, points] = pimc(N);

% Report precision
fprintf('Approximation is %.5f with %d points. (%.5f absolute error)\n', ...
    piapprox, N, abs(pi-piapprox));

% Plot result
plot(4.*cumsum(points)./(1:N)')
line([0,N],[pi,pi], 'color', 'r')
```

Your turn..

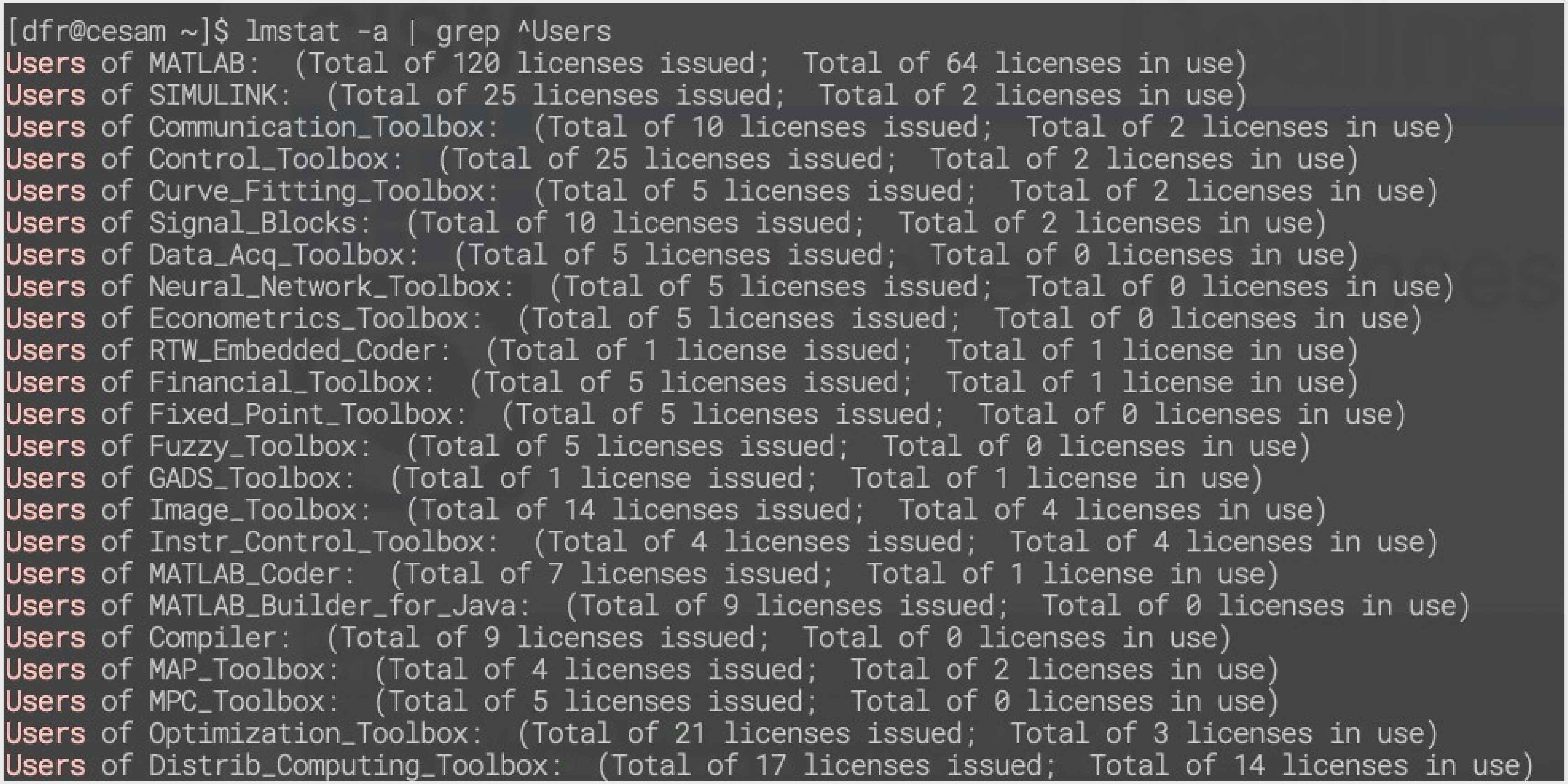

ImageMagick: plot_function.png
4
3.8
3.6
3.4
3.2
3
2.8
2.6
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

Parallel matlab on the cluster

- Using Matlab in batch mode
  - With Matlab (e.g. your computer or CeSAM)
  - Without Matlab (e.g. the clusters)
- Using Matlab in parallel
  - With no effort
  - With little effort
  - With a lot of effort

Dealing with the license

Number of licenses is limited!



```
[dfr@cesam ~]$ lmstat -a | grep ^Users
Users of MATLAB:  (Total of 120 licenses issued;  Total of 64 licenses in use)
Users of SIMULINK:  (Total of 25 licenses issued;  Total of 2 licenses in use)
Users of Communication_Toolbox:  (Total of 10 licenses issued;  Total of 2 licenses in use)
Users of Control_Toolbox:  (Total of 25 licenses issued;  Total of 2 licenses in use)
Users of Curve_Fitting_Toolbox:  (Total of 5 licenses issued;  Total of 2 licenses in use)
Users of Signal_Blocks:  (Total of 10 licenses issued;  Total of 2 licenses in use)
Users of Data_Acq_Toolbox:  (Total of 5 licenses issued;  Total of 0 licenses in use)
Users of Neural_Network_Toolbox:  (Total of 5 licenses issued;  Total of 0 licenses in use)
Users of Econometrics_Toolbox:  (Total of 5 licenses issued;  Total of 0 licenses in use)
Users of RTW_Embedded_Coder:  (Total of 1 license issued;  Total of 1 license in use)
Users of Financial_Toolbox:  (Total of 5 licenses issued;  Total of 1 license in use)
Users of Fixed_Point_Toolbox:  (Total of 5 licenses issued;  Total of 0 licenses in use)
Users of Fuzzy_Toolbox:  (Total of 5 licenses issued;  Total of 0 licenses in use)
Users of GADS_Toolbox:  (Total of 1 license issued;  Total of 1 license in use)
Users of Image_Toolbox:  (Total of 14 licenses issued;  Total of 4 licenses in use)
Users of Instr_Control_Toolbox:  (Total of 4 licenses issued;  Total of 4 licenses in use)
Users of MATLAB_Coder:  (Total of 7 licenses issued;  Total of 1 license in use)
Users of MATLAB_Builder_for_Java:  (Total of 9 licenses issued;  Total of 0 licenses in use)
Users of Compiler:  (Total of 9 licenses issued;  Total of 0 licenses in use)
Users of MAP_Toolbox:  (Total of 4 licenses issued;  Total of 2 licenses in use)
Users of MPC_Toolbox:  (Total of 5 licenses issued;  Total of 0 licenses in use)
Users of Optimization_Toolbox:  (Total of 21 licenses issued;  Total of 3 licenses in use)
Users of Distrib_Computing_Toolbox:  (Total of 17 licenses issued;  Total of 14 licenses in use)
```

CISM

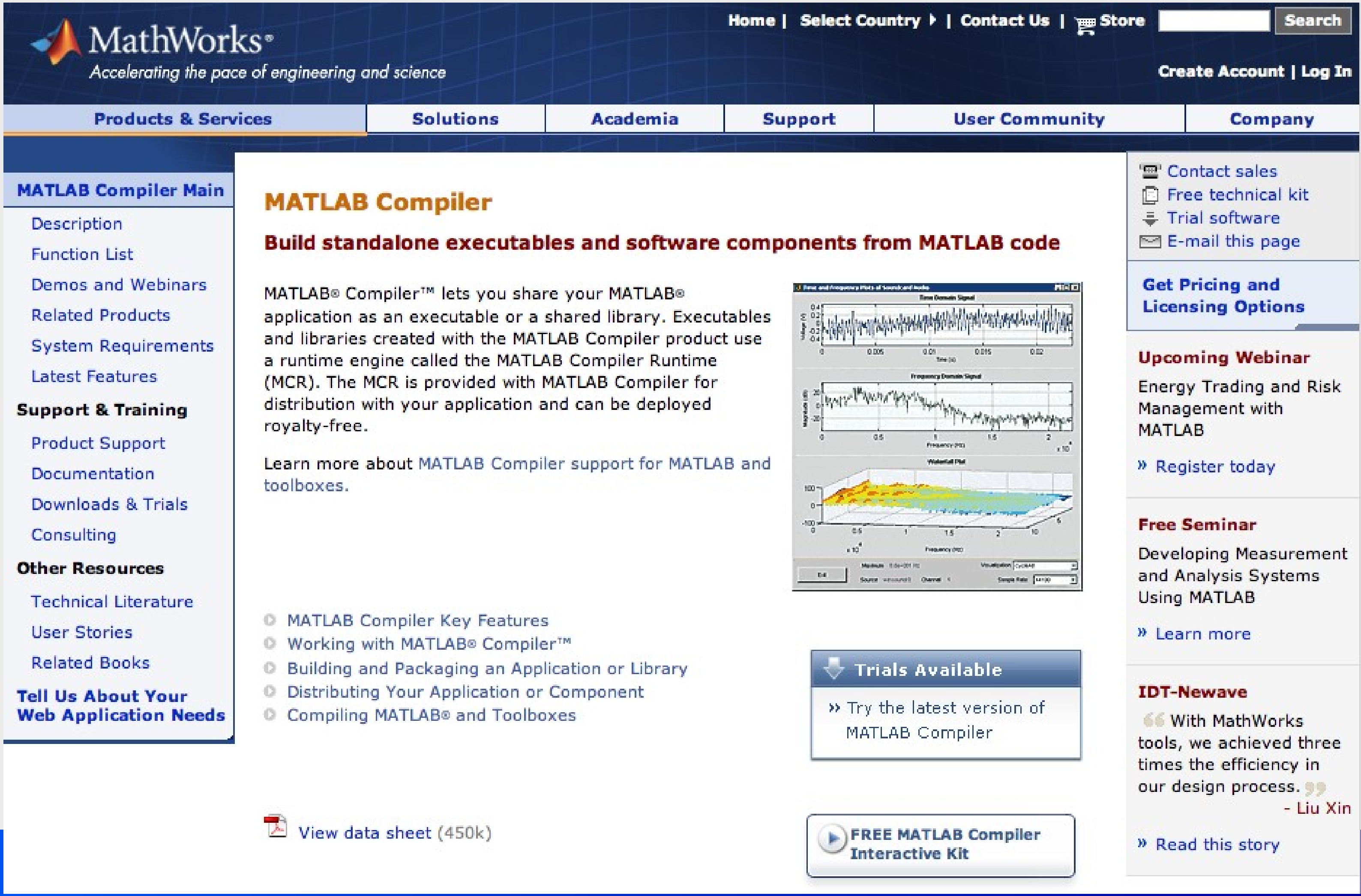
Dealing with the license

Option 1 : Compile Matlab to C...

Why

How

Issues



Home | Select Country | Contact Us | Store | Search

MathWorks®

Accelerating the pace of engineering and science

Create Account | Log In

Products & Services | Solutions | Academia | Support | User Community | Company

MATLAB Compiler Main

Description

Function List

Demos and Webinars

Related Products

System Requirements

Latest Features

Support & Training

Product Support

Documentation

Downloads & Trials

Consulting

Other Resources

Technical Literature

User Stories

Related Books

Tell Us About Your Web Application Needs

MATLAB Compiler

Build standalone executables and software components from MATLAB code

MATLAB® Compiler™ lets you share your MATLAB® application as an executable or a shared library. Executables and libraries created with the MATLAB Compiler product use a runtime engine called the MATLAB Compiler Runtime (MCR). The MCR is provided with MATLAB Compiler for distribution with your application and can be deployed royalty-free.

Learn more about MATLAB Compiler support for MATLAB and toolboxes.

- MATLAB Compiler Key Features
- Working with MATLAB® Compiler™
- Building and Packaging an Application or Library
- Distributing Your Application or Component
- Compiling MATLAB® and Toolboxes

View data sheet (450k)

Trials Available

» Try the latest version of MATLAB Compiler

FREE MATLAB Compiler Interactive Kit

Contact sales

Free technical kit

Trial software

E-mail this page

Get Pricing and Licensing Options

Upcoming Webinar

Energy Trading and Risk Management with MATLAB

» Register today

Free Seminar

Developing Measurement and Analysis Systems Using MATLAB

» Learn more

IDT-Newave

“With MathWorks tools, we achieved three times the efficiency in our design process.”

- Liu Xin

» Read this story

CISM

Dealing with the license

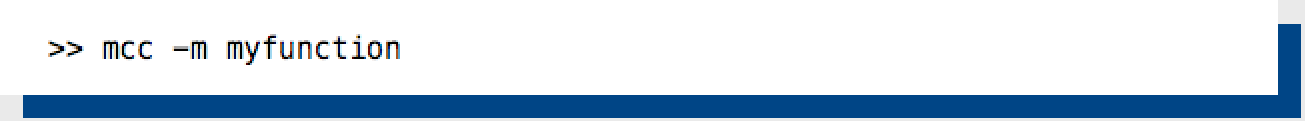
Why

How

Issues

Option 1 : Compile Matlab to C...

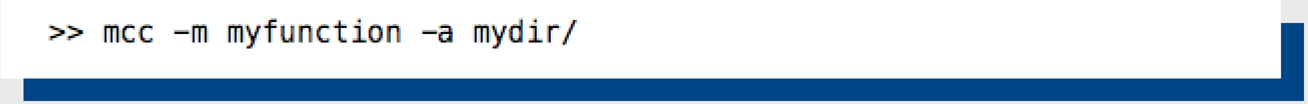
Within Matlab:



```
>> mcc -m myfunction
```

myfunction must be a function, not a script

Use -a to add resources (additional code or mat files)



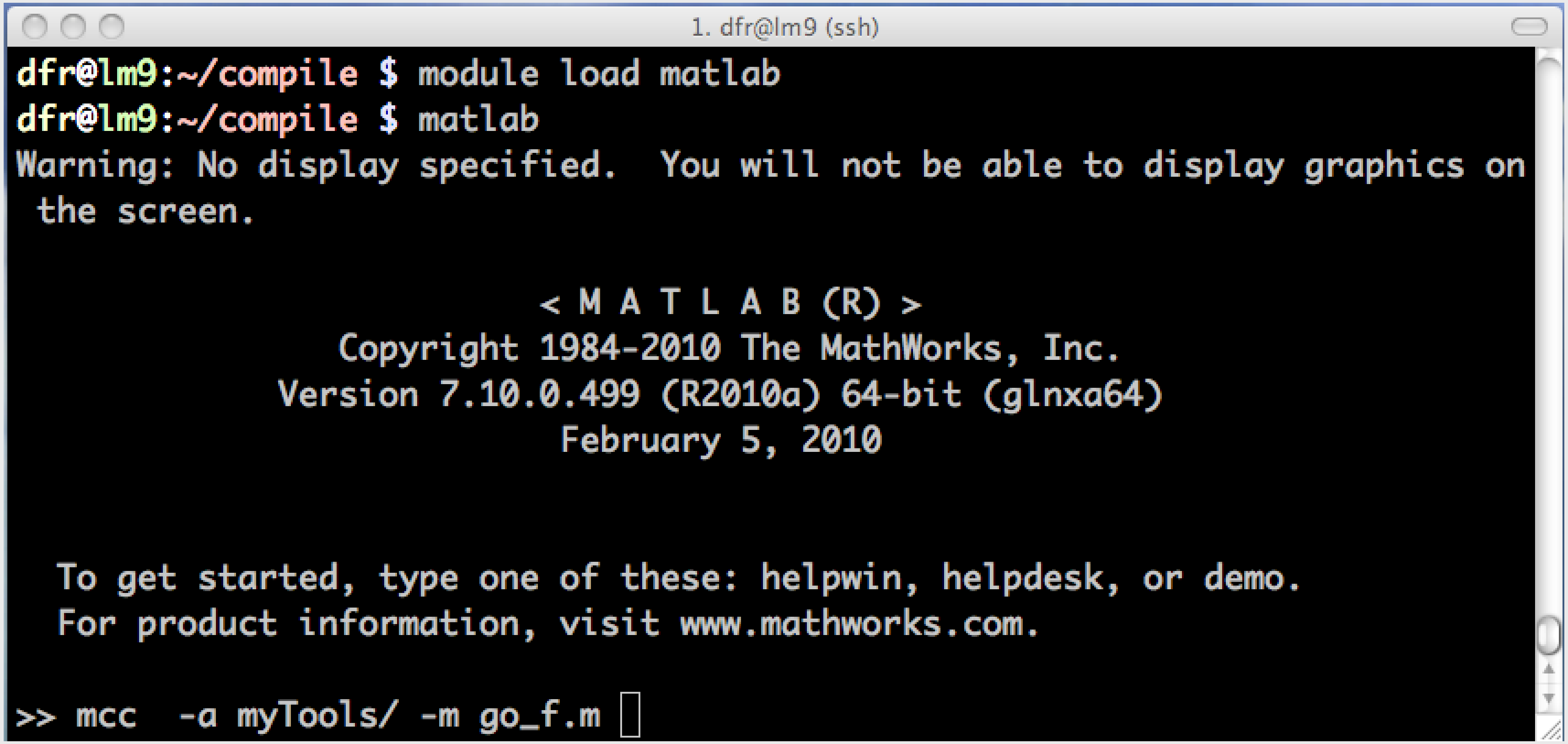
```
>> mcc -m myfunction -a mydir/
```

Addpath are forbidden in compiled code. Protect them with

```
if ~isdeployed
  addpath(...)
end
```

Dealing with the license

Option 1 : Compile Matlab to C...

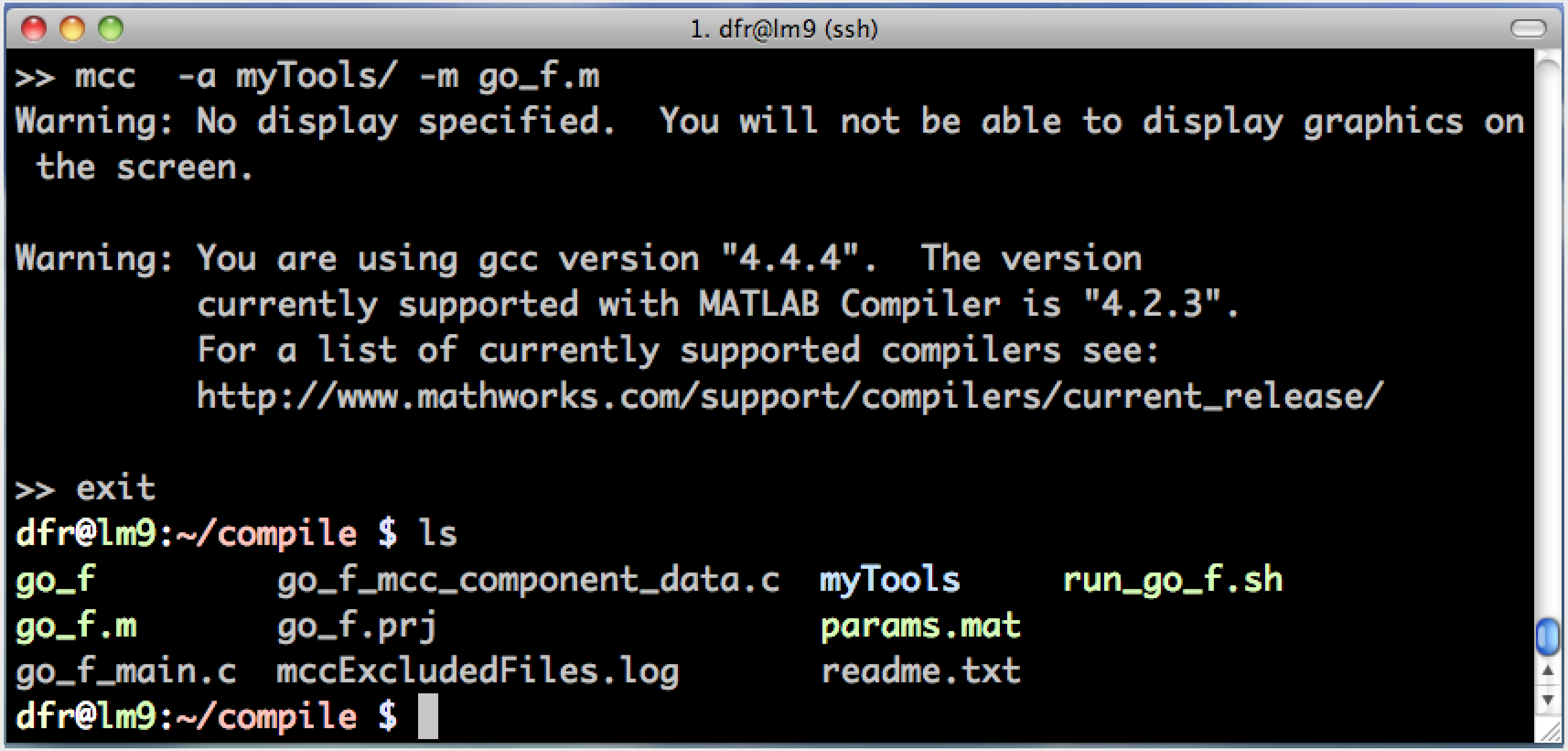


```
dfr@lm9:~/compile $ module load matlab
dfr@lm9:~/compile $ matlab
Warning: No display specified.  You will not be able to display graphics on
 the screen.

                          < M A T L A B (R) >
                Copyright 1984-2010 The MathWorks, Inc.
             Version 7.10.0.499 (R2010a) 64-bit (glnxa64)
                           February 5, 2010


  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, visit www.mathworks.com.

>> mcc  -a myTools/ -m go_f.m
```

CISM

Dealing with the license

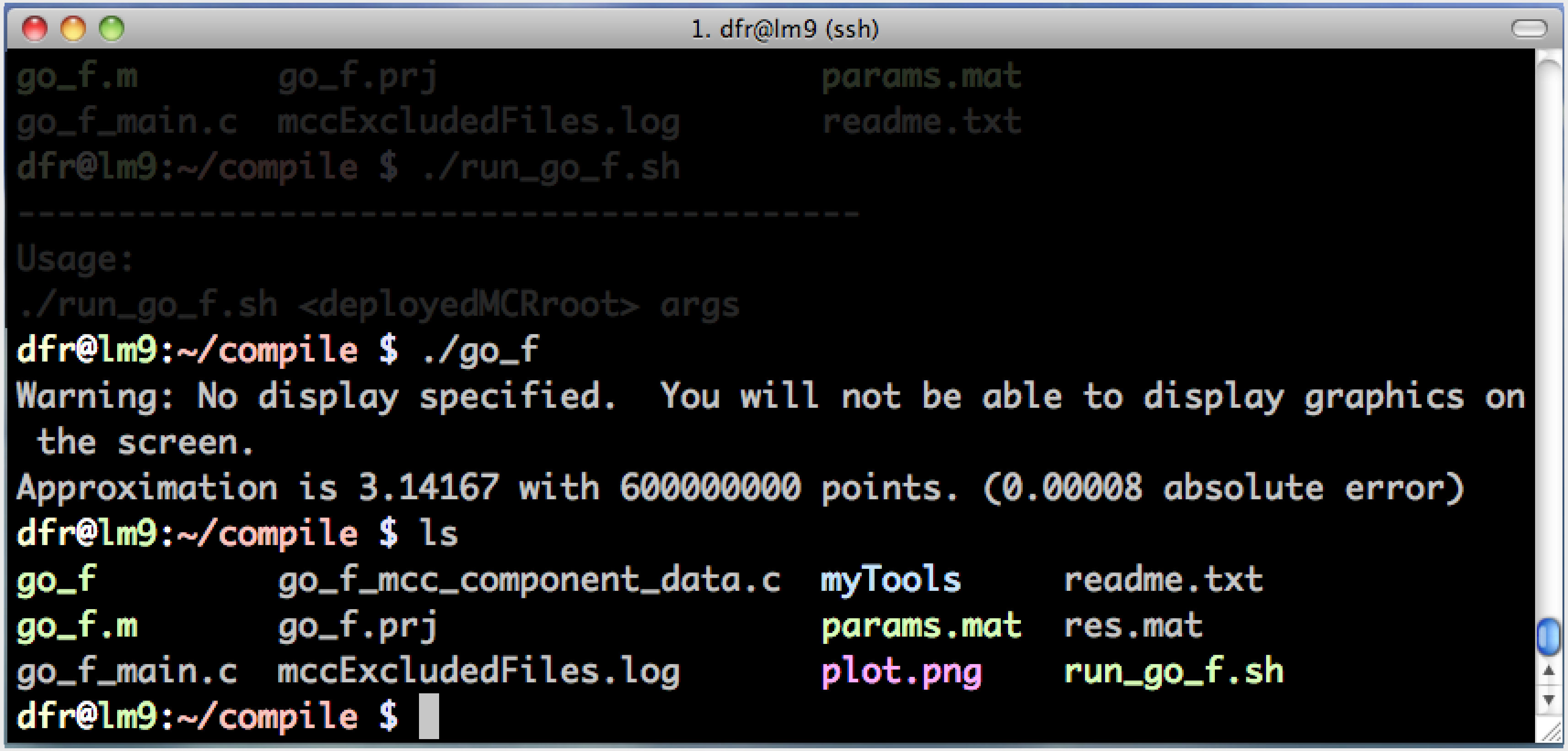
Option 1 : Compile Matlab to C...



```
>> mcc  -a myTools/ -m go_f.m
Warning: No display specified.  You will not be able to display graphics on
 the screen.

Warning: You are using gcc version "4.4.4".  The version
         currently supported with MATLAB Compiler is "4.2.3".
         For a list of currently supported compilers see:
         http://www.mathworks.com/support/compilers/current_release/

>> exit
dfr@lm9:~/compile $ ls
go_f         go_f_mcc_component_data.c  myTools     run_go_f.sh
go_f.m       go_f.prj                   params.mat
go_f_main.c  mccExcludedFiles.log       readme.txt
dfr@lm9:~/compile $ 
```

Dealing with the license

Option 1 : Compile Matlab to C...



```
go_f.m         go_f.prj                    params.mat
go_f_main.c  mccExcludedFiles.log        readme.txt
dfr@lm9:~/compile $ ./run_go_f.sh
------------------------------------------
Usage:
./run_go_f.sh <deployedMCRroot> args
dfr@lm9:~/compile $ ./go_f
Warning: No display specified.  You will not be able to display graphics on
 the screen.
Approximation is 3.14167 with 600000000 points. (0.00008 absolute error)
dfr@lm9:~/compile $ ls
go_f         go_f_mcc_component_data.c  myTools     readme.txt
go_f.m       go_f.prj                   params.mat  res.mat
go_f_main.c  mccExcludedFiles.log       plot.png    run_go_f.sh
dfr@lm9:~/compile $
```

CISM

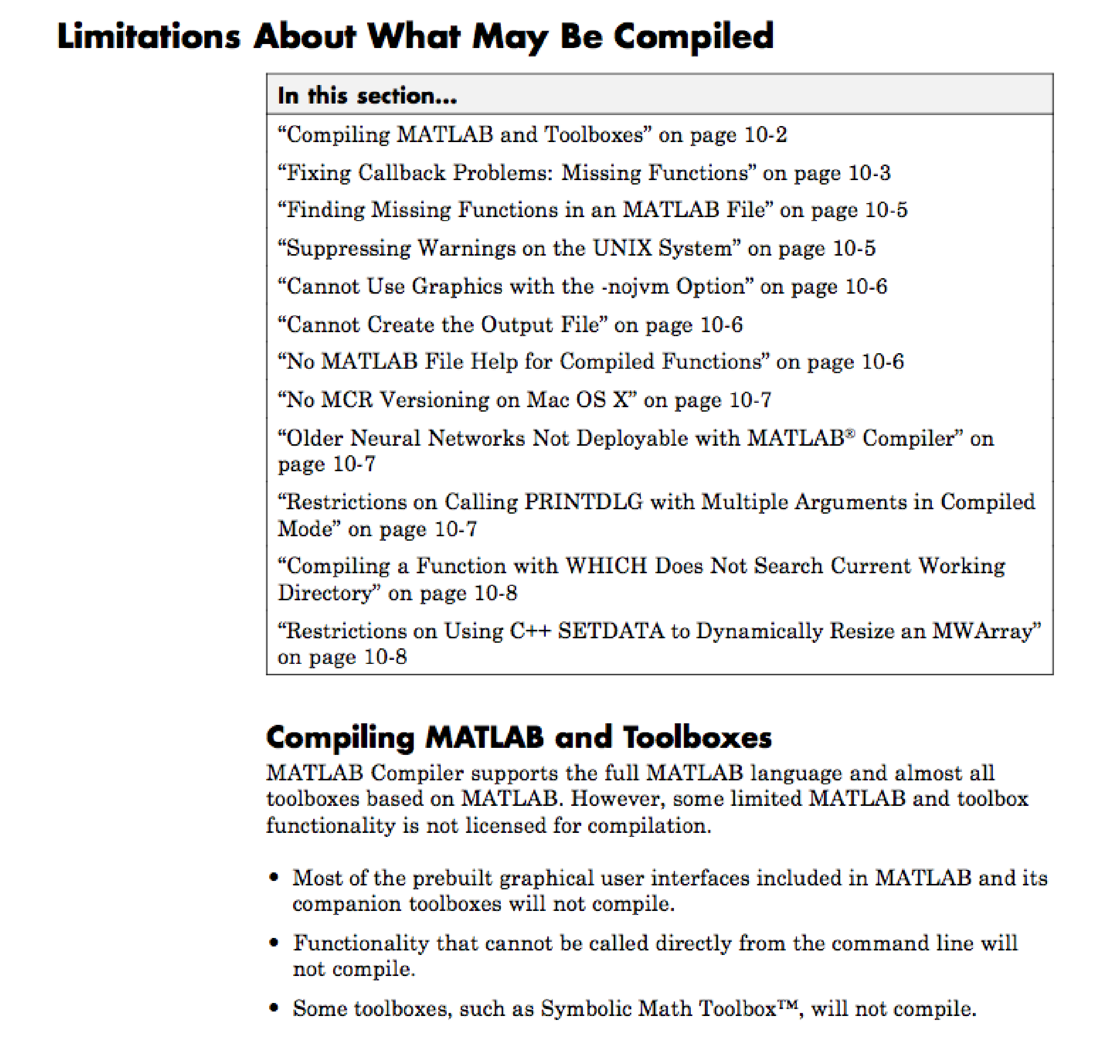
Dealing with the license

Why

How

Issues

Option 1 : Compile Matlab to C...



Limitations About What May Be Compiled

In this section...

Compiling MATLAB and Toolboxes

MATLAB Compiler supports the full MATLAB language and almost all toolboxes based on MATLAB. However, some limited MATLAB and toolbox functionality is not licensed for compilation.

- Most of the prebuilt graphical user interfaces included in MATLAB and its companion toolboxes will not compile.
- Functionality that cannot be called directly from the command line will not compile.
- Some toolboxes, such as Symbolic Math Toolbox™, will not compile.

CISM

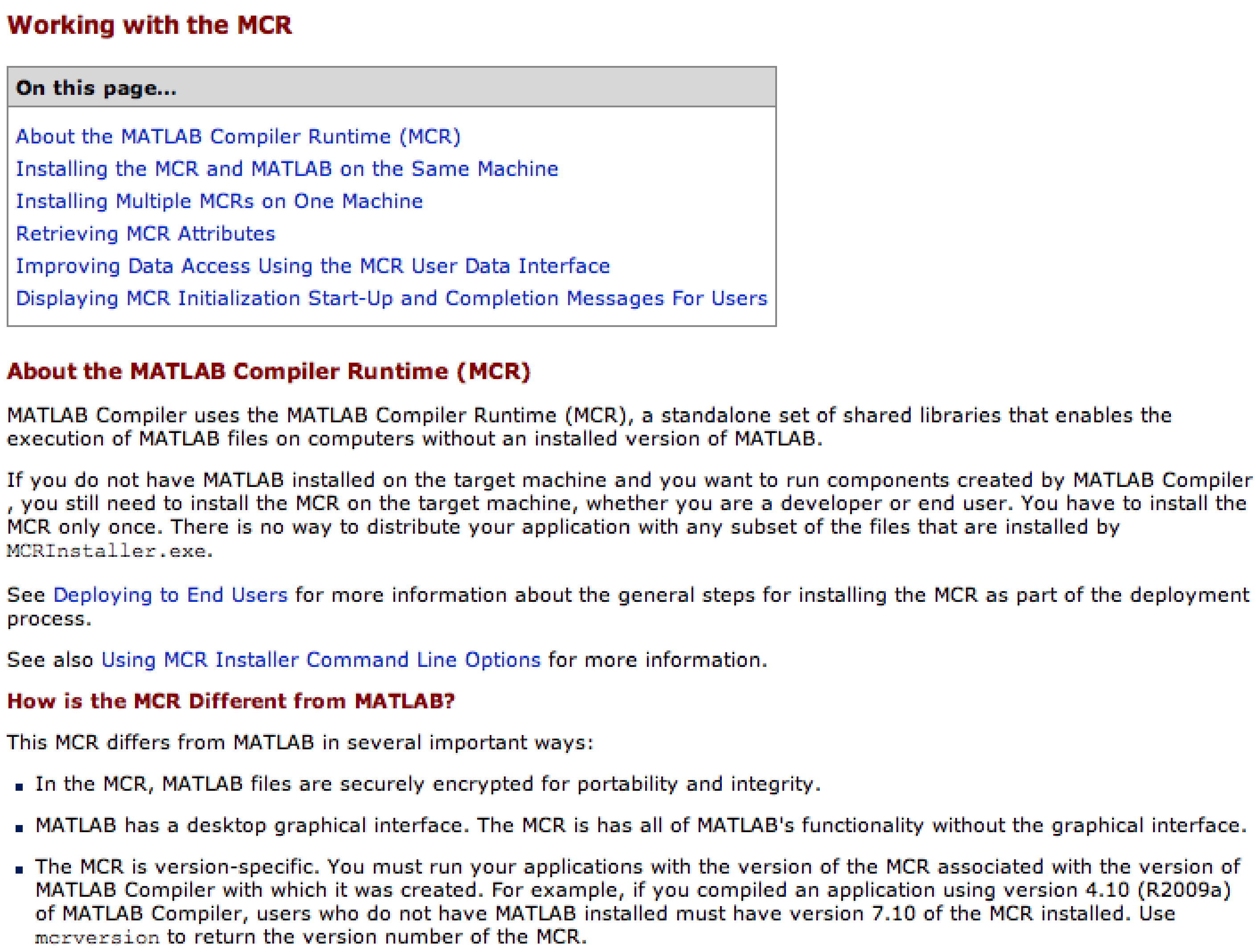
Dealing with the license

Why

How

Issues

Option 1 : ... and deploy with MCR



Working with the MCR

On this page...

About the MATLAB Compiler Runtime (MCR)

MATLAB Compiler uses the MATLAB Compiler Runtime (MCR), a standalone set of shared libraries that enables the execution of MATLAB files on computers without an installed version of MATLAB.

If you do not have MATLAB installed on the target machine and you want to run components created by MATLAB Compiler , you still need to install the MCR on the target machine, whether you are a developer or end user. You have to install the MCR only once. There is no way to distribute your application with any subset of the files that are installed by `MCRInstaller.exe`.

See Deploying to End Users for more information about the general steps for installing the MCR as part of the deployment process.

See also Using MCR Installer Command Line Options for more information.

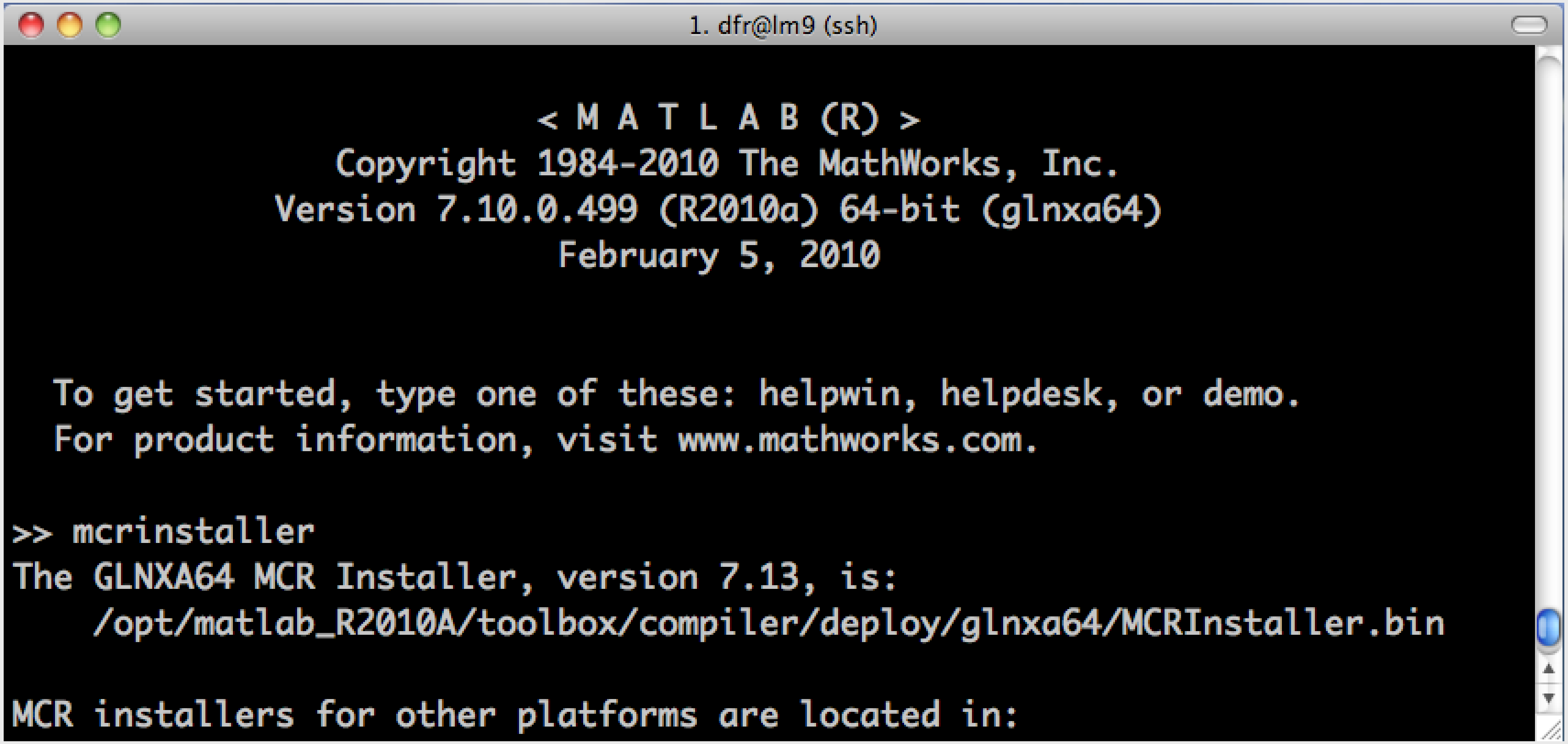
How is the MCR Different from MATLAB?

This MCR differs from MATLAB in several important ways:

- In the MCR, MATLAB files are securely encrypted for portability and integrity.
- MATLAB has a desktop graphical interface. The MCR is has all of MATLAB's functionality without the graphical interface.
- The MCR is version-specific. You must run your applications with the version of the MCR associated with the version of MATLAB Compiler with which it was created. For example, if you compiled an application using version 4.10 (R2009a) of MATLAB Compiler, users who do not have MATLAB installed must have version 7.10 of the MCR installed. Use `mcrversion` to return the version number of the MCR.

Dealing with the license

Option 1 : ... and deploy with MCR

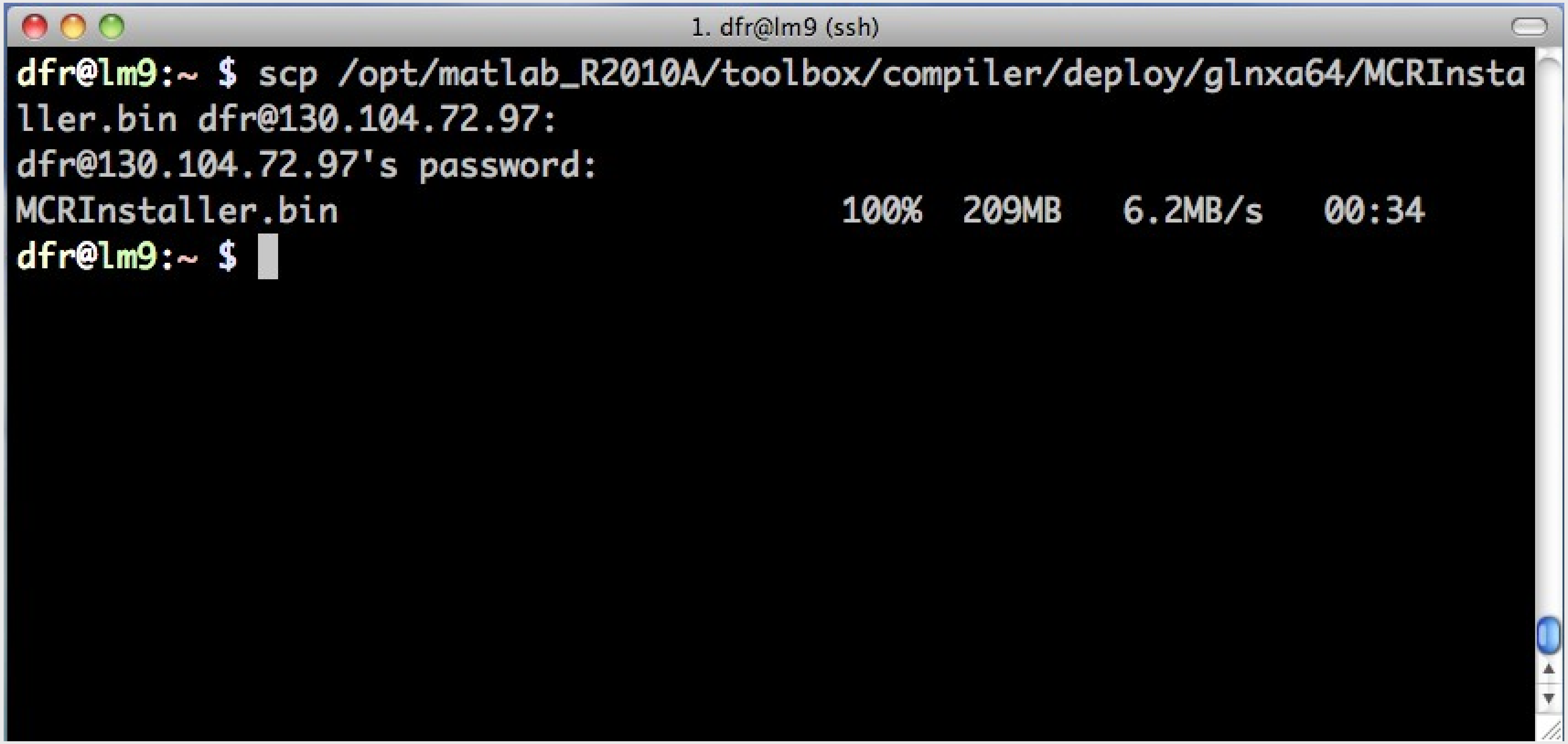


1. dfr@lm9 (ssh)

```
                          < M A T L A B (R) >
                Copyright 1984-2010 The MathWorks, Inc.
             Version 7.10.0.499 (R2010a) 64-bit (glnxa64)
                           February 5, 2010


  To get started, type one of these: helpwin, helpdesk, or demo.
  For product information, visit www.mathworks.com.

>> mcrinstaller
The GLNXA64 MCR Installer, version 7.13, is:
    /opt/matlab_R2010A/toolbox/compiler/deploy/glnxa64/MCRInstaller.bin

MCR installers for other platforms are located in:
```

Dealing with the license

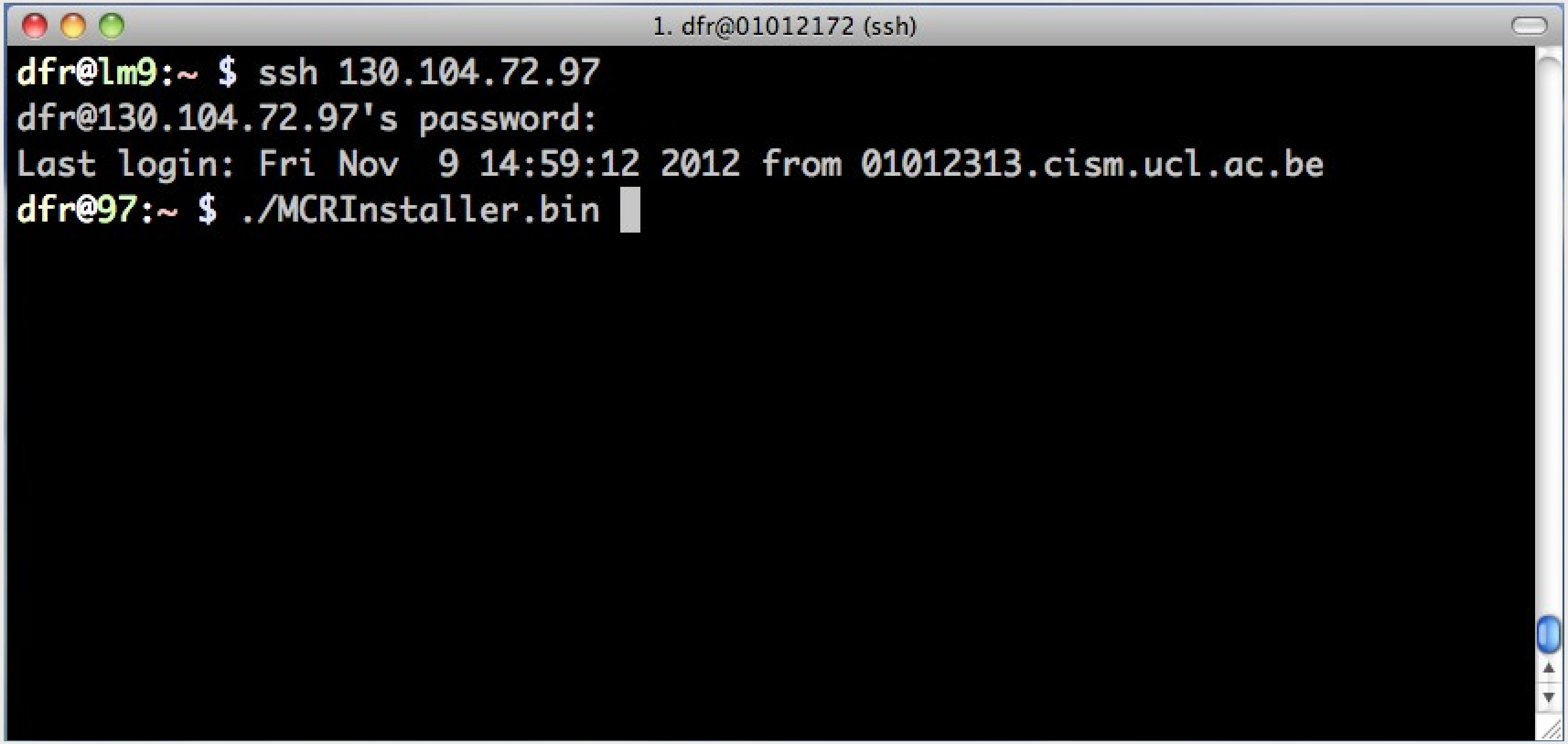
Option 1 : ... and deploy with MCR



```
dfr@lm9:~ $ scp /opt/matlab_R2010A/toolbox/compiler/deploy/glnxa64/MCRInsta
ller.bin dfr@130.104.72.97:
dfr@130.104.72.97's password:
MCRInstaller.bin                                  100%  209MB   6.2MB/s   00:34
dfr@lm9:~ $ 
```

Dealing with the license

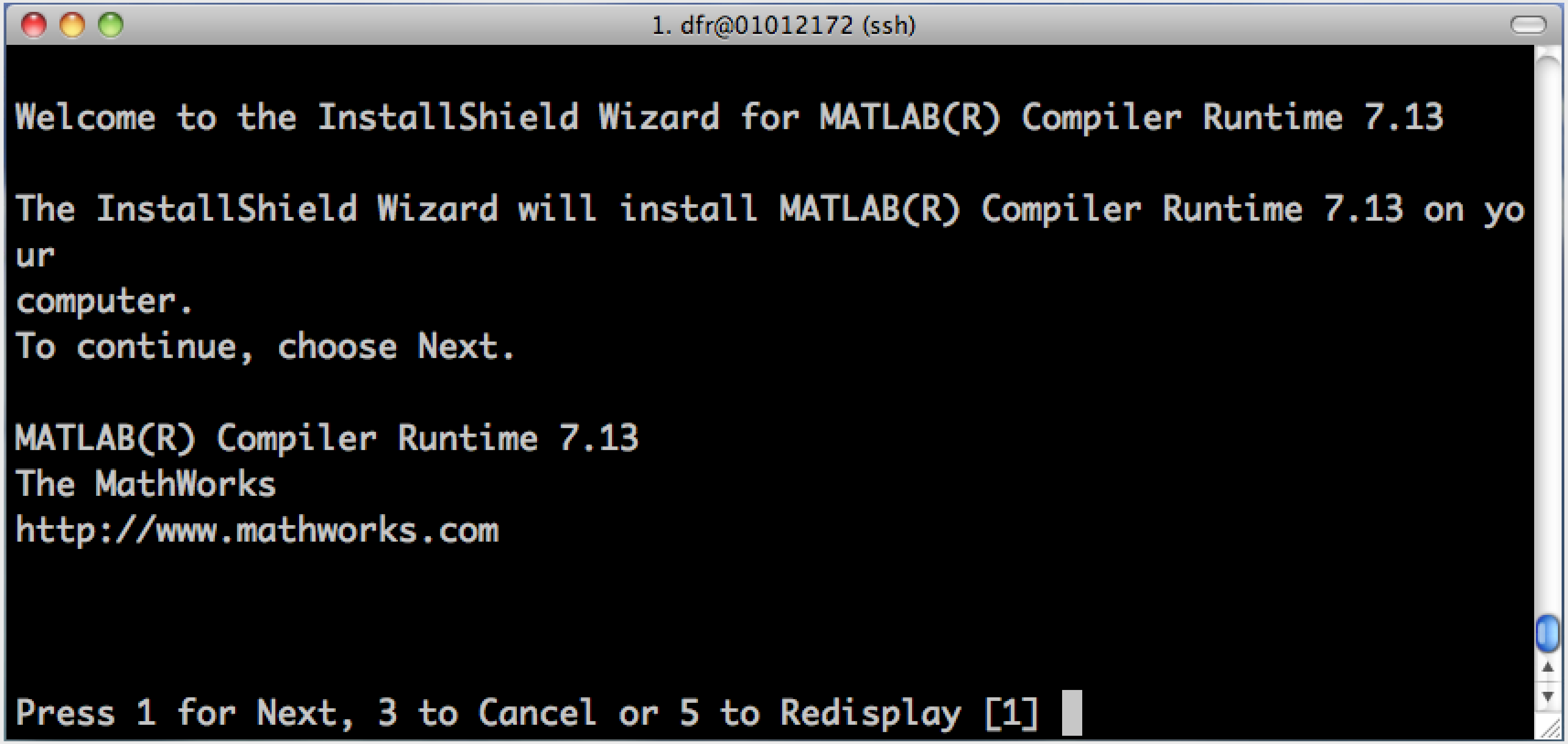
Option 1 : ... and deploy with MCR



```
dfr@lm9:~ $ ssh 130.104.72.97
dfr@130.104.72.97's password:
Last login: Fri Nov  9 14:59:12 2012 from 01012313.cism.ucl.ac.be
dfr@97:~ $ ./MCRInstaller.bin
```

Dealing with the license

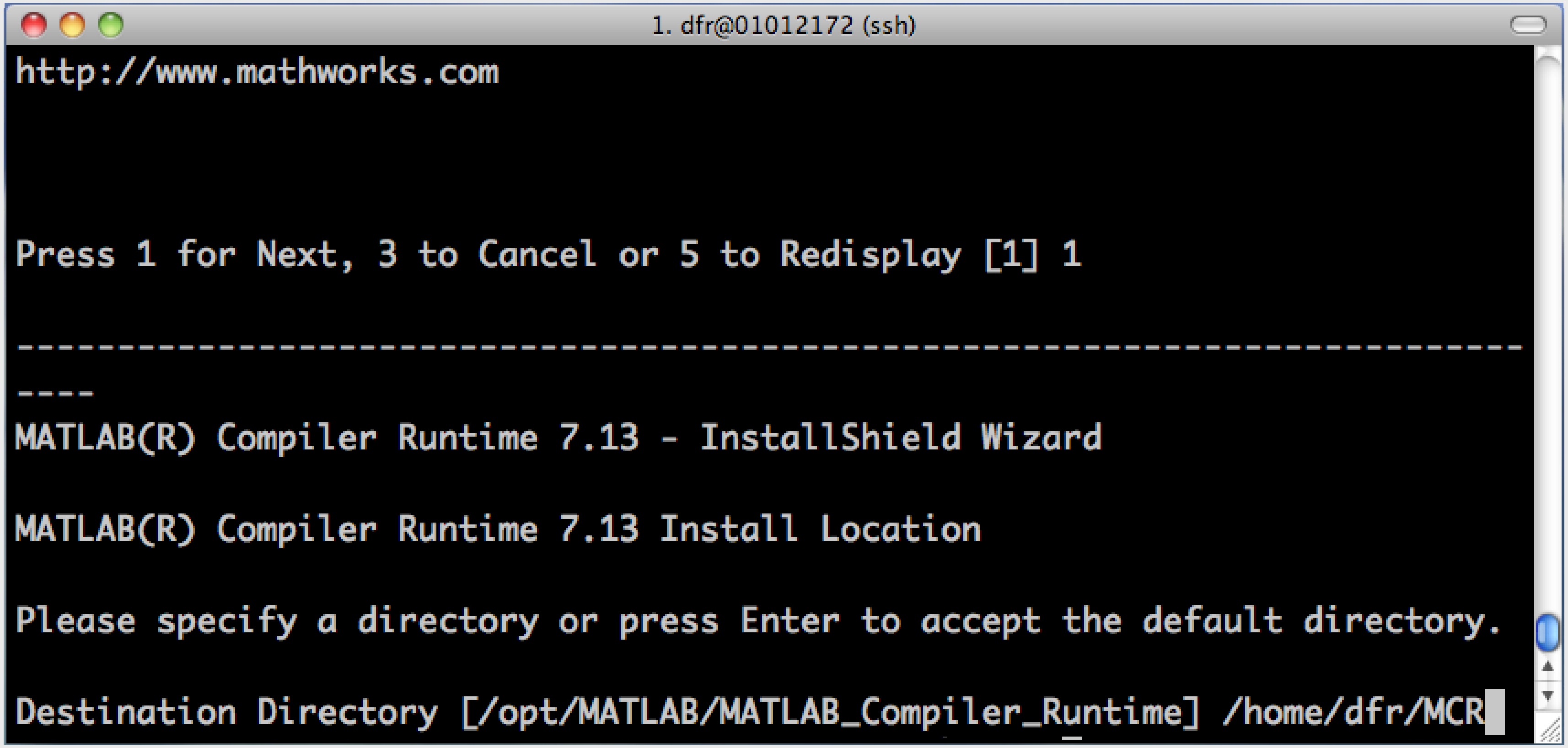
Option 1 : ... and deploy with MCR



```
Welcome to the InstallShield Wizard for MATLAB(R) Compiler Runtime 7.13

The InstallShield Wizard will install MATLAB(R) Compiler Runtime 7.13 on yo
ur
computer.
To continue, choose Next.

MATLAB(R) Compiler Runtime 7.13
The MathWorks
http://www.mathworks.com



Press 1 for Next, 3 to Cancel or 5 to Redisplay [1]
```

Dealing with the license

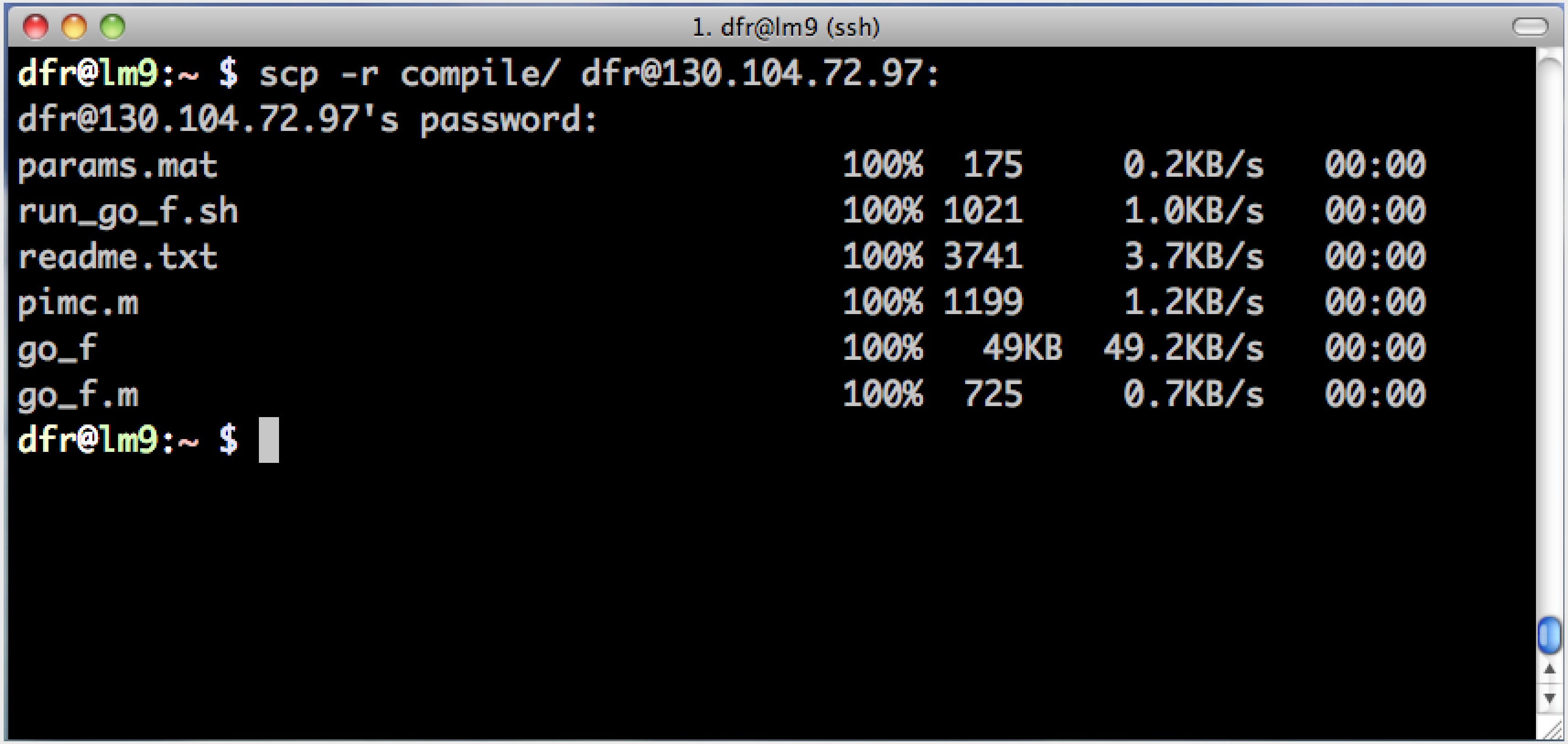
Option 1 : ... and deploy with MCR



```
http://www.mathworks.com



Press 1 for Next, 3 to Cancel or 5 to Redisplay [1] 1

---------------------------------------------------------------------------
----
MATLAB(R) Compiler Runtime 7.13 - InstallShield Wizard

MATLAB(R) Compiler Runtime 7.13 Install Location

Please specify a directory or press Enter to accept the default directory.

Destination Directory [/opt/MATLAB/MATLAB_Compiler_Runtime] /home/dfr/MCR
```

Dealing with the license

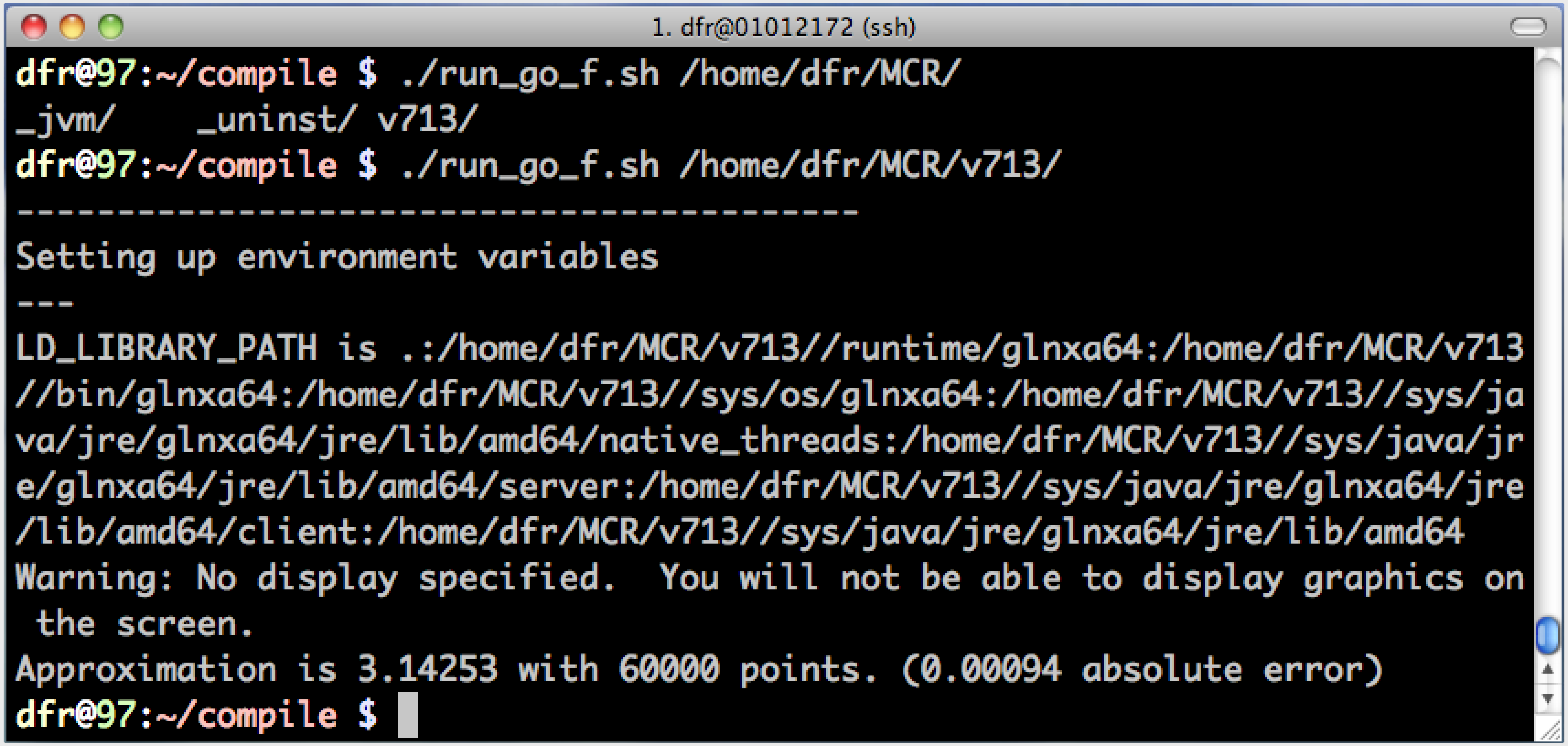
Option 1 : ... and deploy with MCR



```
dfr@lm9:~ $ scp -r compile/ dfr@130.104.72.97:
dfr@130.104.72.97's password:
params.mat                                    100%  175     0.2KB/s   00:00
run_go_f.sh                                   100% 1021     1.0KB/s   00:00
readme.txt                                    100% 3741     3.7KB/s   00:00
pimc.m                                        100% 1199     1.2KB/s   00:00
go_f                                          100%   49KB  49.2KB/s   00:00
go_f.m                                        100%  725     0.7KB/s   00:00
dfr@lm9:~ $
```

Dealing with the license

Option 1 : ... and deploy with MCR



```
dfr@97:~/compile $ ./run_go_f.sh /home/dfr/MCR/
_jvm/    _uninst/ v713/
dfr@97:~/compile $ ./run_go_f.sh /home/dfr/MCR/v713/
------------------------------------------
Setting up environment variables
---
LD_LIBRARY_PATH is .:/home/dfr/MCR/v713//runtime/glnxa64:/home/dfr/MCR/v713
//bin/glnxa64:/home/dfr/MCR/v713//sys/os/glnxa64:/home/dfr/MCR/v713//sys/ja
va/jre/glnxa64/jre/lib/amd64/native_threads:/home/dfr/MCR/v713//sys/java/jr
e/glnxa64/jre/lib/amd64/server:/home/dfr/MCR/v713//sys/java/jre/glnxa64/jre
/lib/amd64/client:/home/dfr/MCR/v713//sys/java/jre/glnxa64/jre/lib/amd64
Warning: No display specified.  You will not be able to display graphics on
 the screen.
Approximation is 3.14253 with 60000 points. (0.00094 absolute error)
dfr@97:~/compile $ 
```

Your turn..

1. Connect to CeSAM or Brufence

2. Copy directory ~dfr/matlab/compile to your directory and “cd” there

3. Load module matlab and launch Matlab
4. Compile go_f.m (note if ~isdeployed)
   mcc -a myTools/ -m go_f.m

5. Connect to Lemaitre3 with your CÉCI login

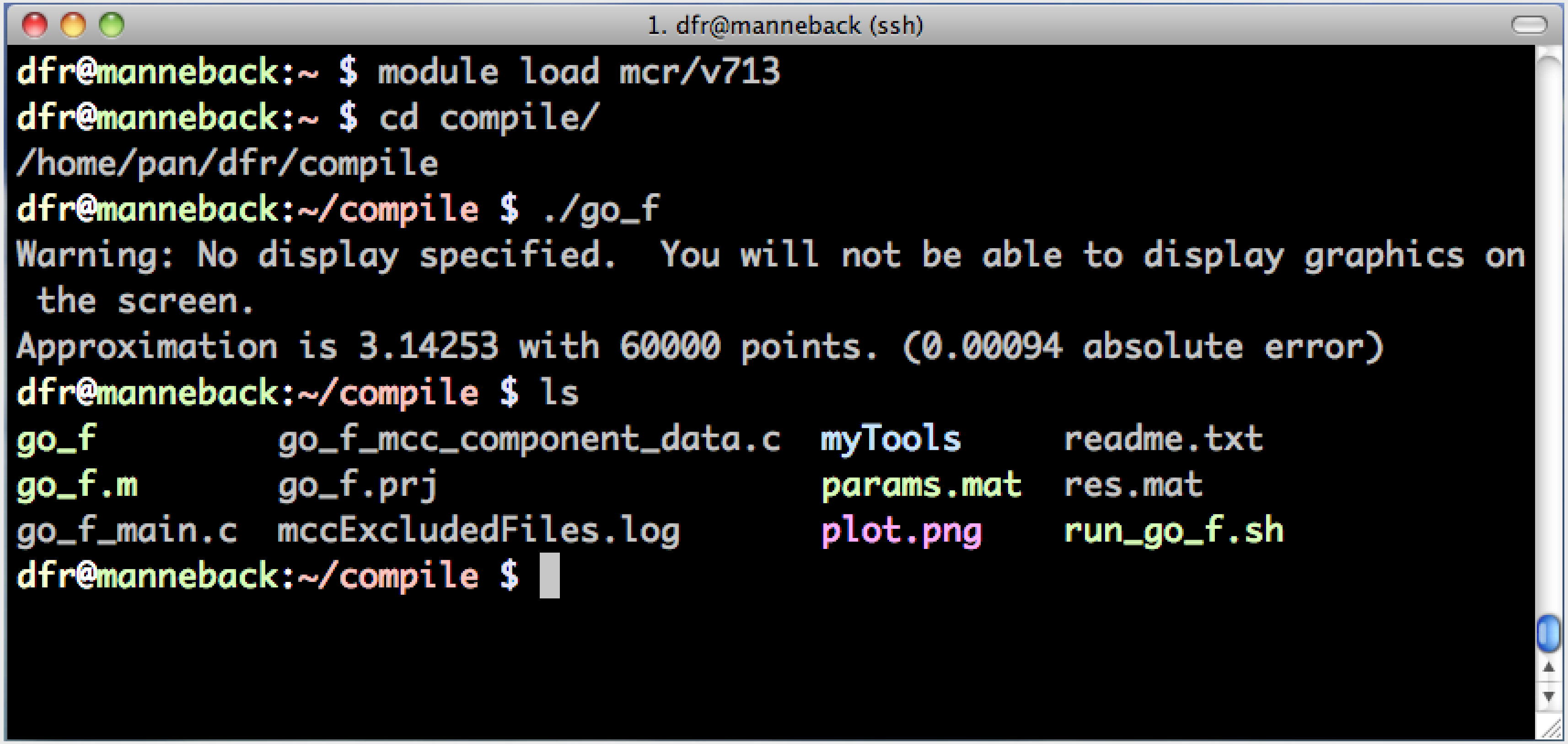
6. Copy your 'compile' directory from CeSAM or Brufence
7. Load MCR module (!version)

9. Run go_f (no need for run_go_f.sh)

Dealing with the license

Option 1 : ... and deploy with MCR

Matlab is not installed on Manneback but the MCR is



```
dfr@manneback:~ $ module load mcr/v713
dfr@manneback:~ $ cd compile/
/home/pan/dfr/compile
dfr@manneback:~/compile $ ./go_f
Warning: No display specified.  You will not be able to display graphics on
 the screen.
Approximation is 3.14253 with 60000 points. (0.00094 absolute error)
dfr@manneback:~/compile $ ls
go_f         go_f_mcc_component_data.c  myTools     readme.txt
go_f.m       go_f.prj                   params.mat  res.mat
go_f_main.c  mccExcludedFiles.log       plot.png    run_go_f.sh
dfr@manneback:~/compile $ 
```

CISM

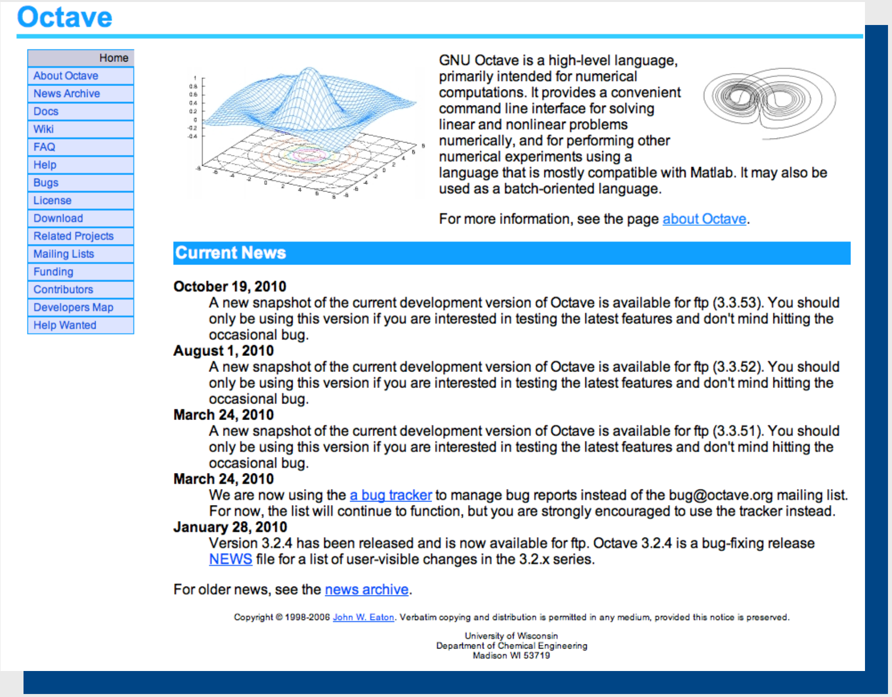
Dealing with the license

Option 2 : Develop with Matlab, run with Octave

Why

How

Issues



Octave

- Home
- About Octave
- News Archive
- Docs
- Wiki
- FAQ
- Help
- Bugs
- License
- Download
- Related Projects
- Mailing Lists
- Funding
- Contributors
- Developers Map
- Help Wanted

GNU Octave is a high-level language, primarily intended for numerical computations. It provides a convenient command line interface for solving linear and nonlinear problems numerically, and for performing other numerical experiments using a language that is mostly compatible with Matlab. It may also be used as a batch-oriented language.

For more information, see the page about Octave.

Current News

October 19, 2010
A new snapshot of the current development version of Octave is available for ftp (3.3.53). You should only be using this version if you are interested in testing the latest features and don't mind hitting the occasional bug.

August 1, 2010
A new snapshot of the current development version of Octave is available for ftp (3.3.52). You should only be using this version if you are interested in testing the latest features and don't mind hitting the occasional bug.

March 24, 2010
A new snapshot of the current development version of Octave is available for ftp (3.3.51). You should only be using this version if you are interested in testing the latest features and don't mind hitting the occasional bug.

March 24, 2010
We are now using the a bug tracker to manage bug reports instead of the bug@octave.org mailing list. For now, the list will continue to function, but you are strongly encouraged to use the tracker instead.

January 28, 2010
Version 3.2.4 has been released and is now available for ftp. Octave 3.2.4 is a bug-fixing release NEWS file for a list of user-visible changes in the 3.2.x series.

For older news, see the news archive.

University of Wisconsin
Department of Chemical Engineering
Madison WI 53719

“a language that is mostly compatible with Matlab”
GPL license, free

CISM

Dealing with the license

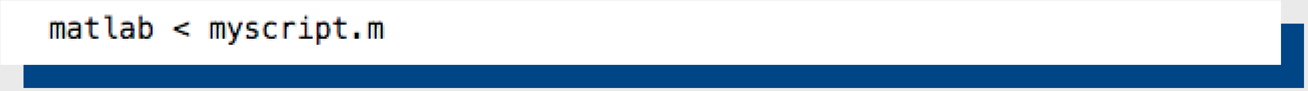
Option 2 : Develop with Matlab, run with Octave

Why

How

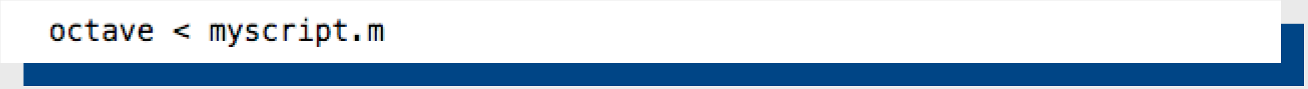
Issues

You have a script myscript.m. Rather than:



```
matlab < myscript.m
```

Simply write



```
octave < myscript.m
```

The other option '-r' becomes '--eval '

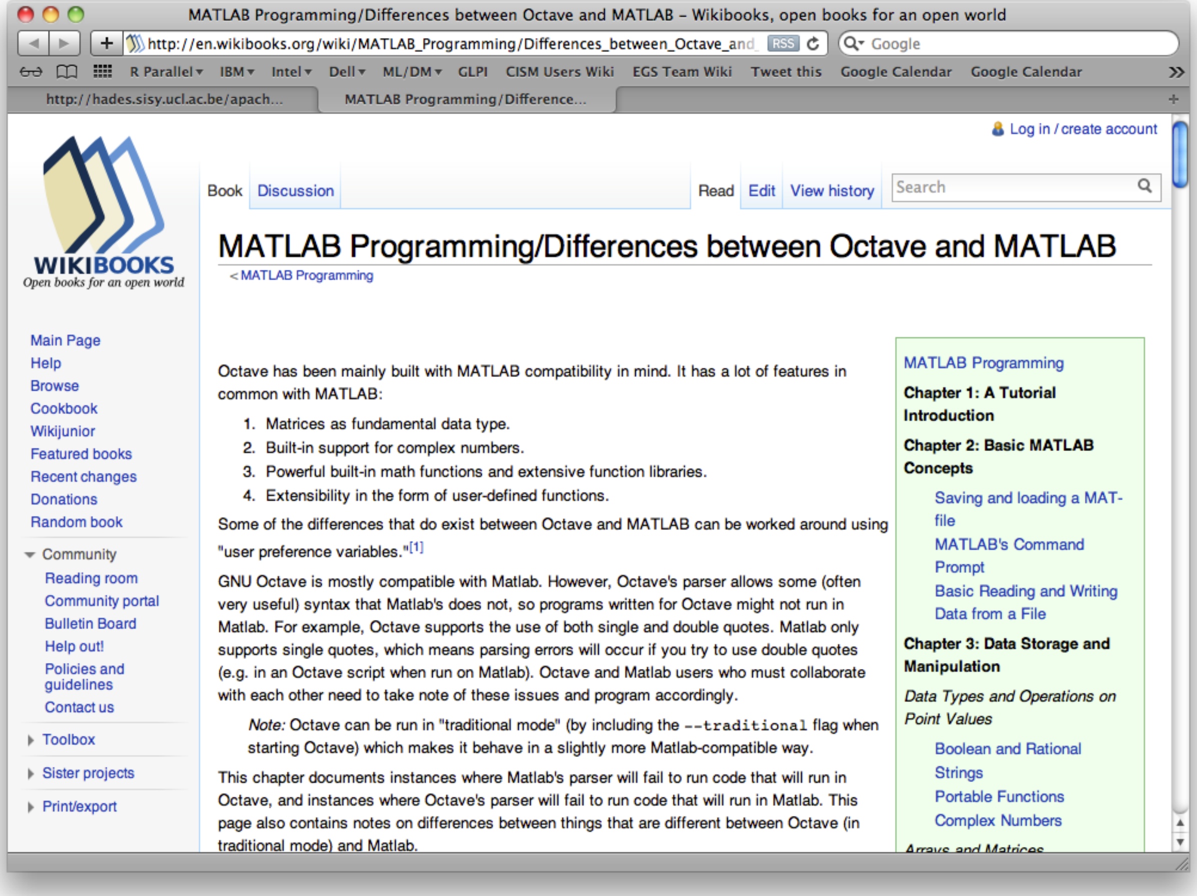
Dealing with the license

Why

How

Issues

Option 2 : Develop with Matlab, run with Octave



MATLAB Programming/Differences between Octave and MATLAB

< MATLAB Programming

Octave has been mainly built with MATLAB compatibility in mind. It has a lot of features in common with MATLAB:

1. Matrices as fundamental data type.
2. Built-in support for complex numbers.
3. Powerful built-in math functions and extensive function libraries.
4. Extensibility in the form of user-defined functions.

Some of the differences that do exist between Octave and MATLAB can be worked around using "user preference variables."[1]

GNU Octave is mostly compatible with Matlab. However, Octave's parser allows some (often very useful) syntax that Matlab's does not, so programs written for Octave might not run in Matlab. For example, Octave supports the use of both single and double quotes. Matlab only supports single quotes, which means parsing errors will occur if you try to use double quotes (e.g. in an Octave script when run on Matlab). Octave and Matlab users who must collaborate with each other need to take note of these issues and program accordingly.

> *Note:* Octave can be run in "traditional mode" (by including the `--traditional` flag when starting Octave) which makes it behave in a slightly more Matlab-compatible way.

This chapter documents instances where Matlab's parser will fail to run code that will run in Octave, and instances where Octave's parser will fail to run code that will run in Matlab. This page also contains notes on differences between things that are different between Octave (in traditional mode) and Matlab.

MATLAB Programming

Chapter 1: A Tutorial Introduction

Chapter 2: Basic MATLAB Concepts

- Saving and loading a MAT-file
- MATLAB's Command Prompt
- Basic Reading and Writing Data from a File

Chapter 3: Data Storage and Manipulation

Data Types and Operations on Point Values

- Boolean and Rational
- Strings
- Portable Functions
- Complex Numbers

Arrays and Matrices

Dealing with the license

Option 2 : Develop with Matlab, run with Octave

Why

How

Issues

Plots

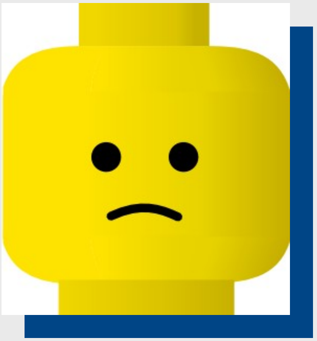
Toolboxes

Java

Multithreading

Speed

Not as good as Matlab's



Your turn..

1. Connect to Lemaitre3

2. Copy your 'compile' directory into 'octave' : cp -r compile octave
   (and remove everything except go_f.m, myTools/ and params.mat)

3. Go to your ‘octave’ directory
4. Load the octave module if needed

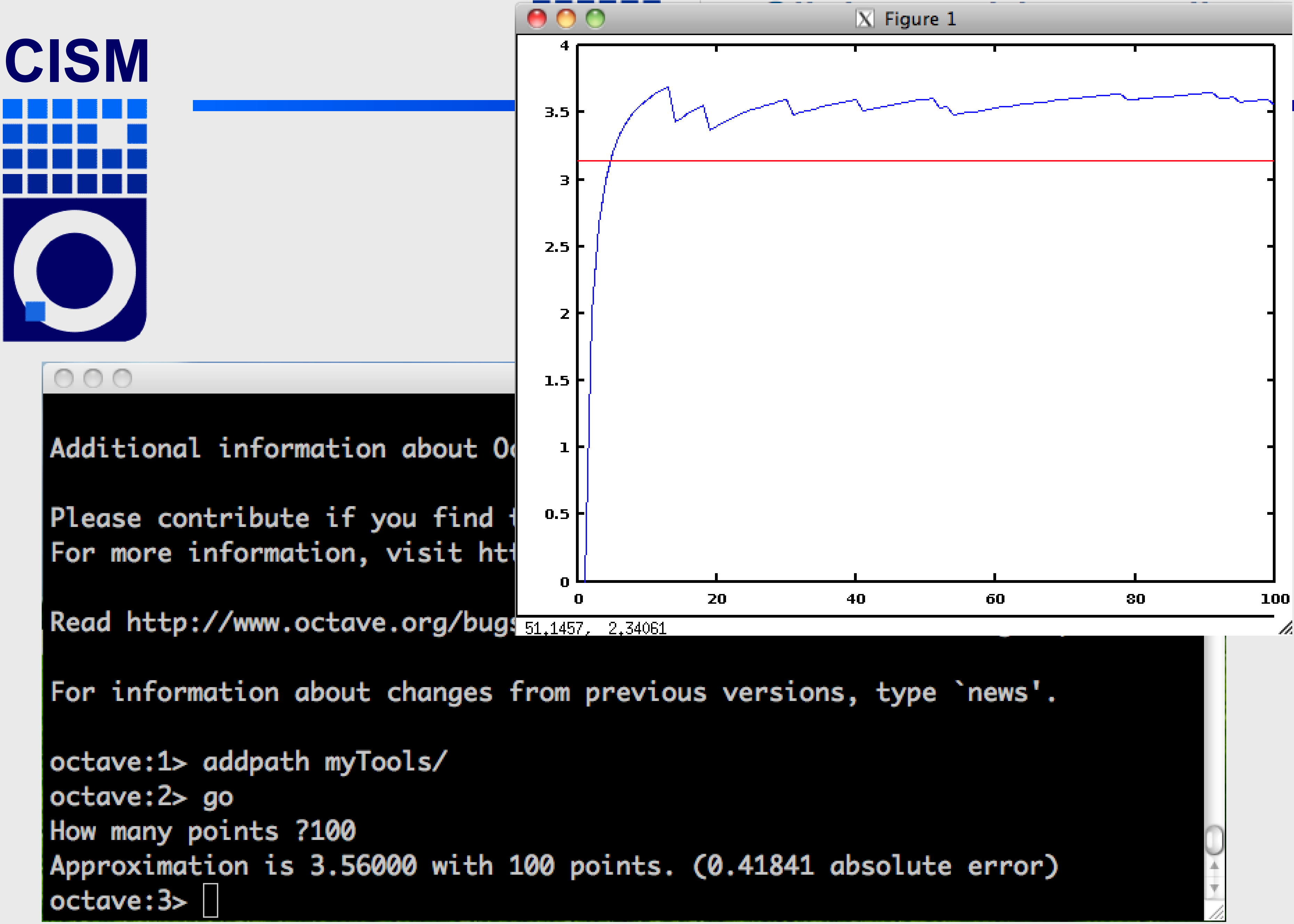
5. Launch octave

6. Run go_f

7. exit octave

8 Launch “octave --eval go_f”

CISM



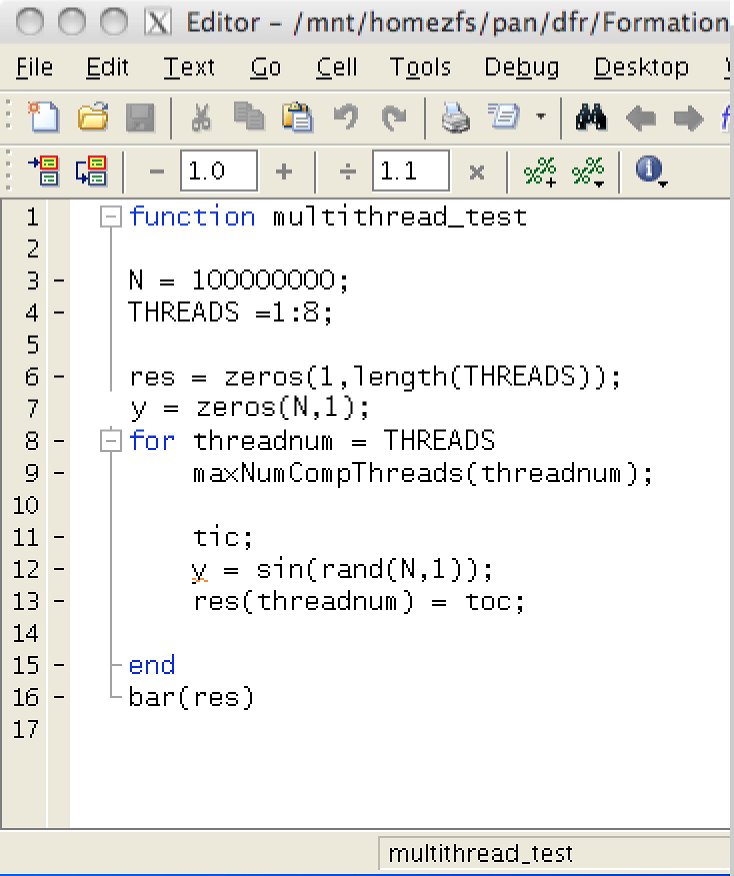
```
Additional information about Oc

Please contribute if you find t
For more information, visit htt

Read http://www.octave.org/bugs

For information about changes from previous versions, type `news'.

octave:1> addpath myTools/
octave:2> go
How many points ?100
Approximation is 3.56000 with 100 points. (0.41841 absolute error)
octave:3> 
```

Parallel matlab on the cluster

- Using Matlab in batch mode
  - Launch a script, get results
  - Deal with licenses
- Using Matlab in parallel
  - With no effort
  - With little effort
  - With a lot of effort

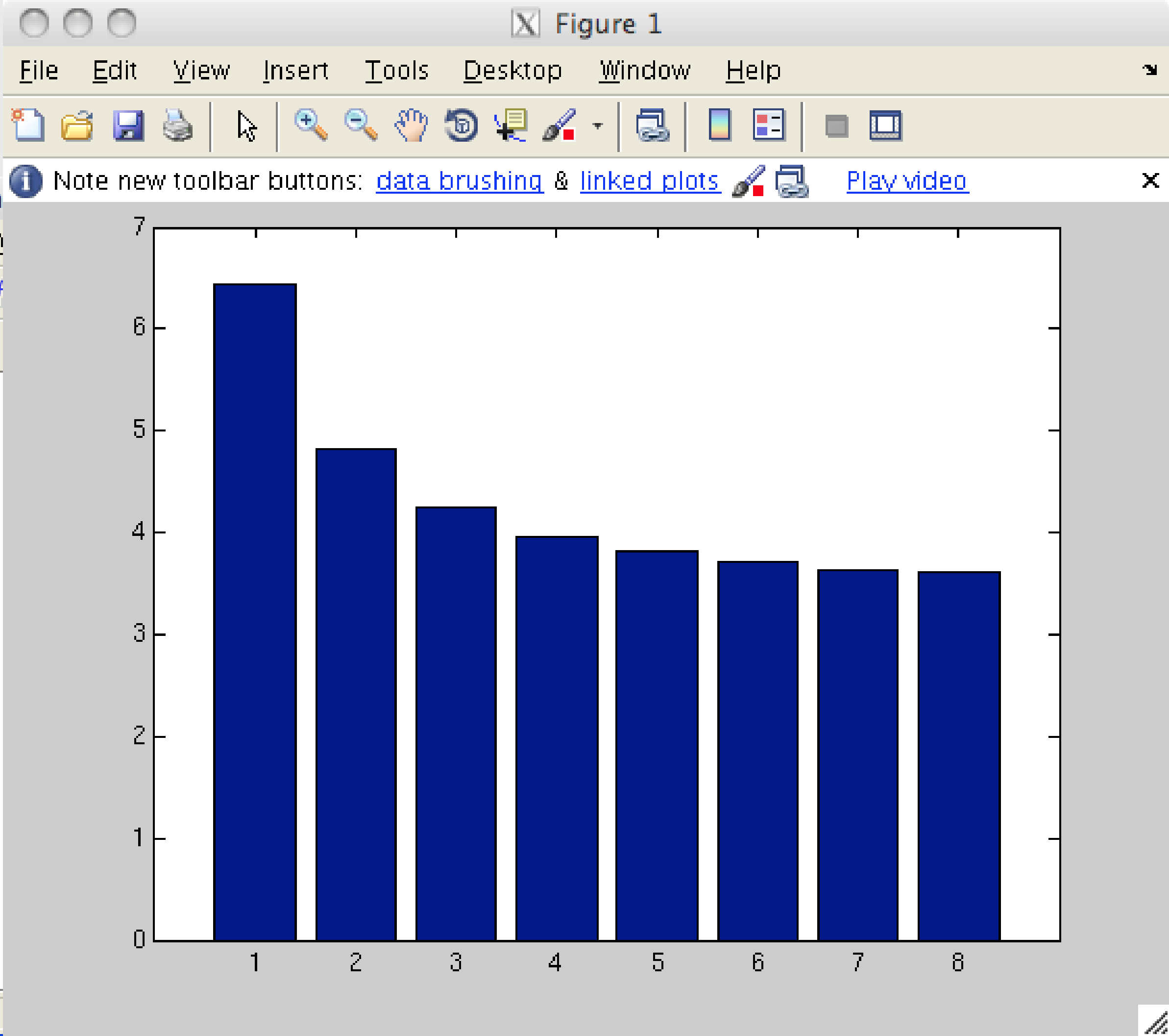
CISM

No effort: Multithreading



```
function multithread_test

N = 100000000;
THREADS =1:8;

res = zeros(1,length(THREADS));
y = zeros(N,1);
for threadnum = THREADS
    maxNumCompThreads(threadnum);

    tic;
    y = sin(rand(N,1));
    res(threadnum) = toc;

end
bar(res)
```

multithread_test

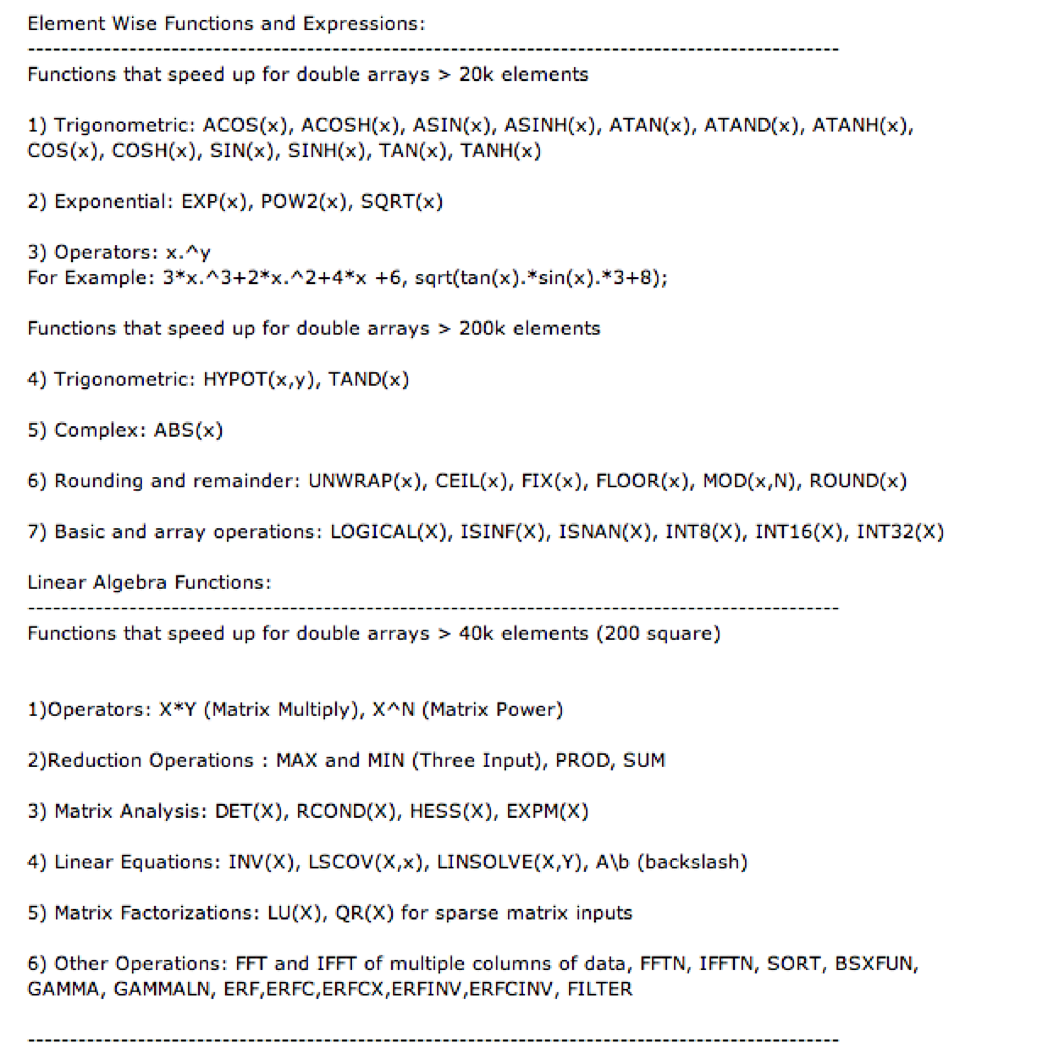


Note new toolbar buttons: data brushing & linked plots Play video

CISM

No effort: Multithreading

Element wise operations and linear algebra



Element Wise Functions and Expressions:

Functions that speed up for double arrays > 20k elements

1) Trigonometric: ACOS(x), ACOSH(x), ASIN(x), ASINH(x), ATAN(x), ATAND(x), ATANH(x), COS(x), COSH(x), SIN(x), SINH(x), TAN(x), TANH(x)

2) Exponential: EXP(x), POW2(x), SQRT(x)

3) Operators: x.^y
For Example: 3*x.^3+2*x.^2+4*x +6, sqrt(tan(x).*sin(x).*3+8);

Functions that speed up for double arrays > 200k elements

4) Trigonometric: HYPOT(x,y), TAND(x)

5) Complex: ABS(x)

6) Rounding and remainder: UNWRAP(x), CEIL(x), FIX(x), FLOOR(x), MOD(x,N), ROUND(x)

7) Basic and array operations: LOGICAL(X), ISINF(X), ISNAN(X), INT8(X), INT16(X), INT32(X)

Linear Algebra Functions:

Functions that speed up for double arrays > 40k elements (200 square)

1)Operators: X*Y (Matrix Multiply), X^N (Matrix Power)

2)Reduction Operations : MAX and MIN (Three Input), PROD, SUM

3) Matrix Analysis: DET(X), RCOND(X), HESS(X), EXPM(X)

4) Linear Equations: INV(X), LSCOV(X,x), LINSOLVE(X,Y), A\b (backslash)

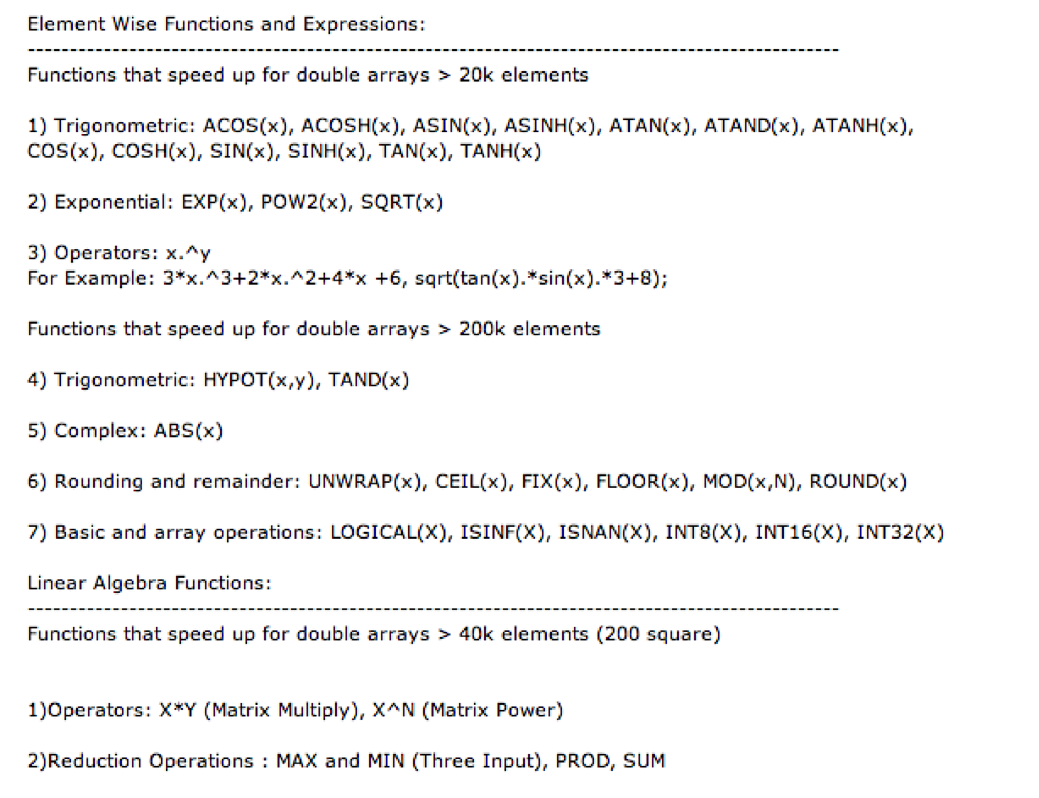
5) Matrix Factorizations: LU(X), QR(X) for sparse matrix inputs

6) Other Operations: FFT and IFFT of multiple columns of data, FFTN, IFFTN, SORT, BSXFUN, GAMMA, GAMMALN, ERF,ERFC,ERFCX,ERFINV,ERFCINV, FILTER

CISM

No effort: Multithreading

Element wise operations and linear algebra



Element Wise Functions and Expressions:

Functions that speed up for double arrays > 20k elements

1) Trigonometric: ACOS(x), ACOSH(x), ASIN(x), ASINH(x), ATAN(x), ATAND(x), ATANH(x), COS(x), COSH(x), SIN(x), SINH(x), TAN(x), TANH(x)

2) Exponential: EXP(x), POW2(x), SQRT(x)

3) Operators: x.^y
For Example: 3*x.^3+2*x.^2+4*x +6, sqrt(tan(x).*sin(x).*3+8);

Functions that speed up for double arrays > 200k elements

4) Trigonometric: HYPOT(x,y), TAND(x)

5) Complex: ABS(x)

6) Rounding and remainder: UNWRAP(x), CEIL(x), FIX(x), FLOOR(x), MOD(x,N), ROUND(x)

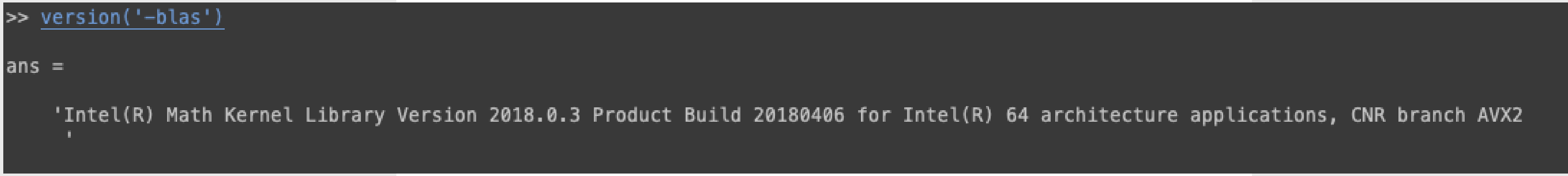
7) Basic and array operations: LOGICAL(X), ISINF(X), ISNAN(X), INT8(X), INT16(X), INT32(X)

Linear Algebra Functions:

Functions that speed up for double arrays > 40k elements (200 square)

1)Operators: X*Y (Matrix Multiply), X^N (Matrix Power)

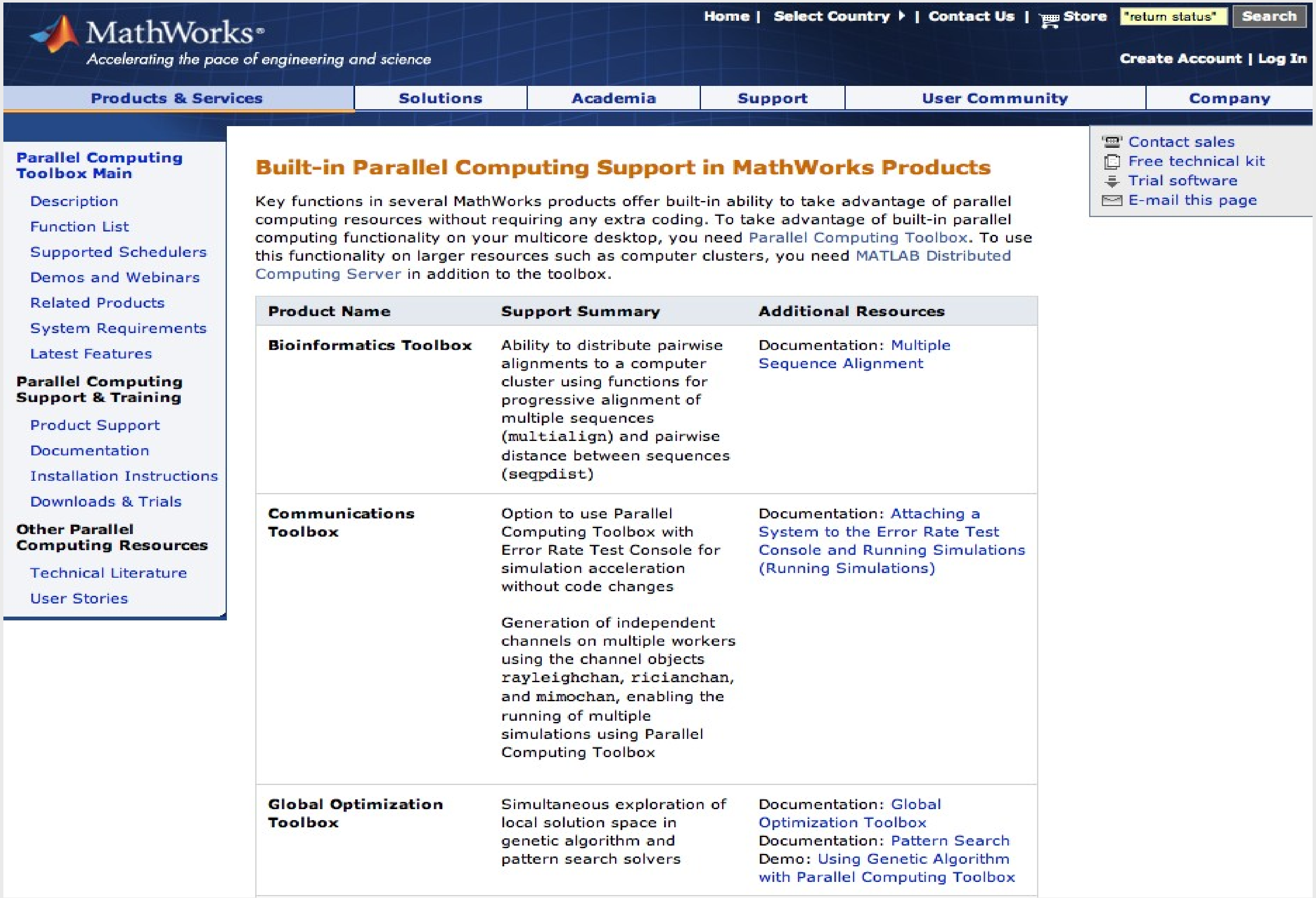
2)Reduction Operations : MAX and MIN (Three Input), PROD, SUM



```
>> version('-blas')

ans =

    'Intel(R) Math Kernel Library Version 2018.0.3 Product Build 20180406 for Intel(R) 64 architecture applications, CNR branch AVX2
     '
```

No effort: Multithreading

More with the Parallel Computing Toolbox



Built-in Parallel Computing Support in MathWorks Products

Key functions in several MathWorks products offer built-in ability to take advantage of parallel computing resources without requiring any extra coding. To take advantage of built-in parallel computing functionality on your multicore desktop, you need Parallel Computing Toolbox. To use this functionality on larger resources such as computer clusters, you need MATLAB Distributed Computing Server in addition to the toolbox.

Product Name	Support Summary	Additional Resources
Bioinformatics Toolbox	Ability to distribute pairwise alignments to a computer cluster using functions for progressive alignment of multiple sequences (`multialign`) and pairwise distance between sequences (`seqpdist`)	Documentation: Multiple Sequence Alignment
Communications Toolbox	Option to use Parallel Computing Toolbox with Error Rate Test Console for simulation acceleration without code changes Generation of independent channels on multiple workers using the channel objects `rayleighchan`, `ricianchan`, and `mimochan`, enabling the running of multiple simulations using Parallel Computing Toolbox	Documentation: Attaching a System to the Error Rate Test Console and Running Simulations (Running Simulations)
Global Optimization Toolbox	Simultaneous exploration of local solution space in genetic algorithm and pattern search solvers	Documentation: Global Optimization Toolbox Documentation: Pattern Search Demo: Using Genetic Algorithm with Parallel Computing Toolbox

Matlab on the cluster

Using Matlab in batch mode

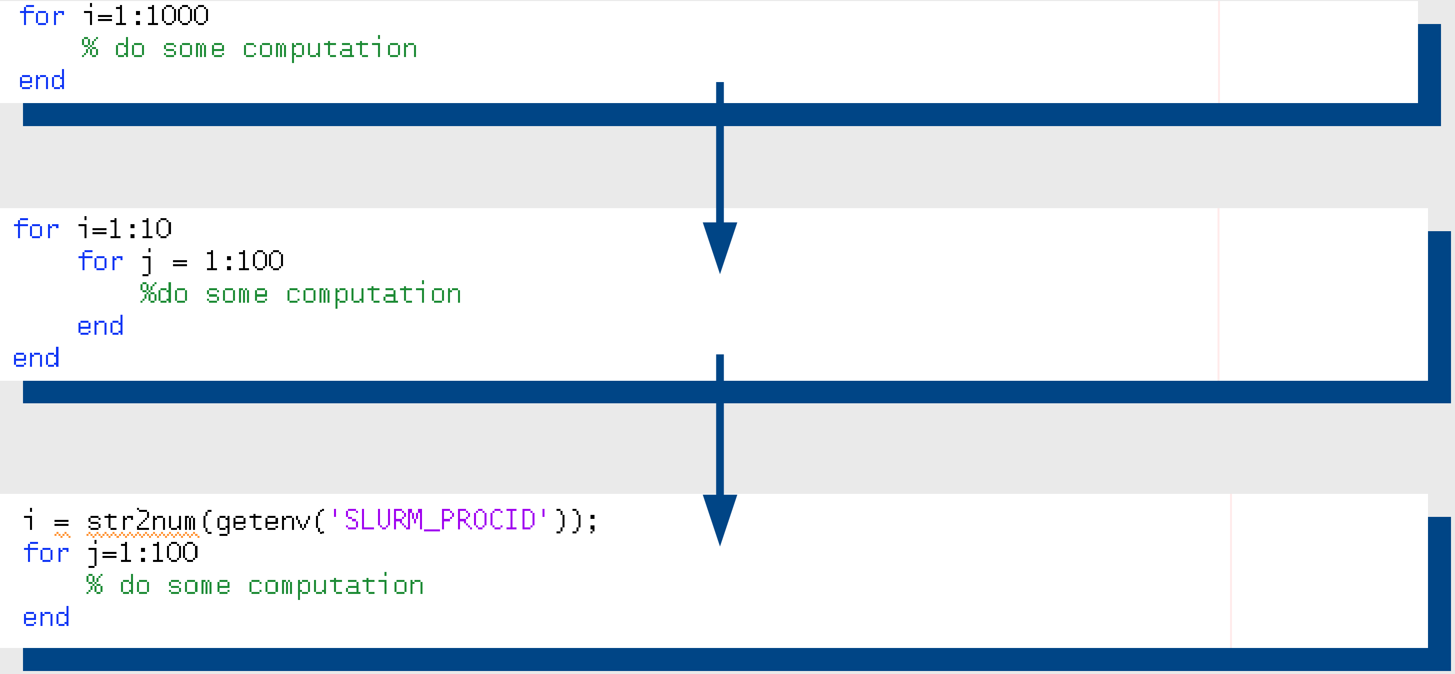
- Launch a script, get results
- Deal with licenses

Using Matlab in parallel

- With no effort
- With little effort
- With a lot of effort

Little effort: embarrassingly parallel

Outer-loop splitting with Slurm's srun



```
for i=1:1000
    % do some computation
end
```

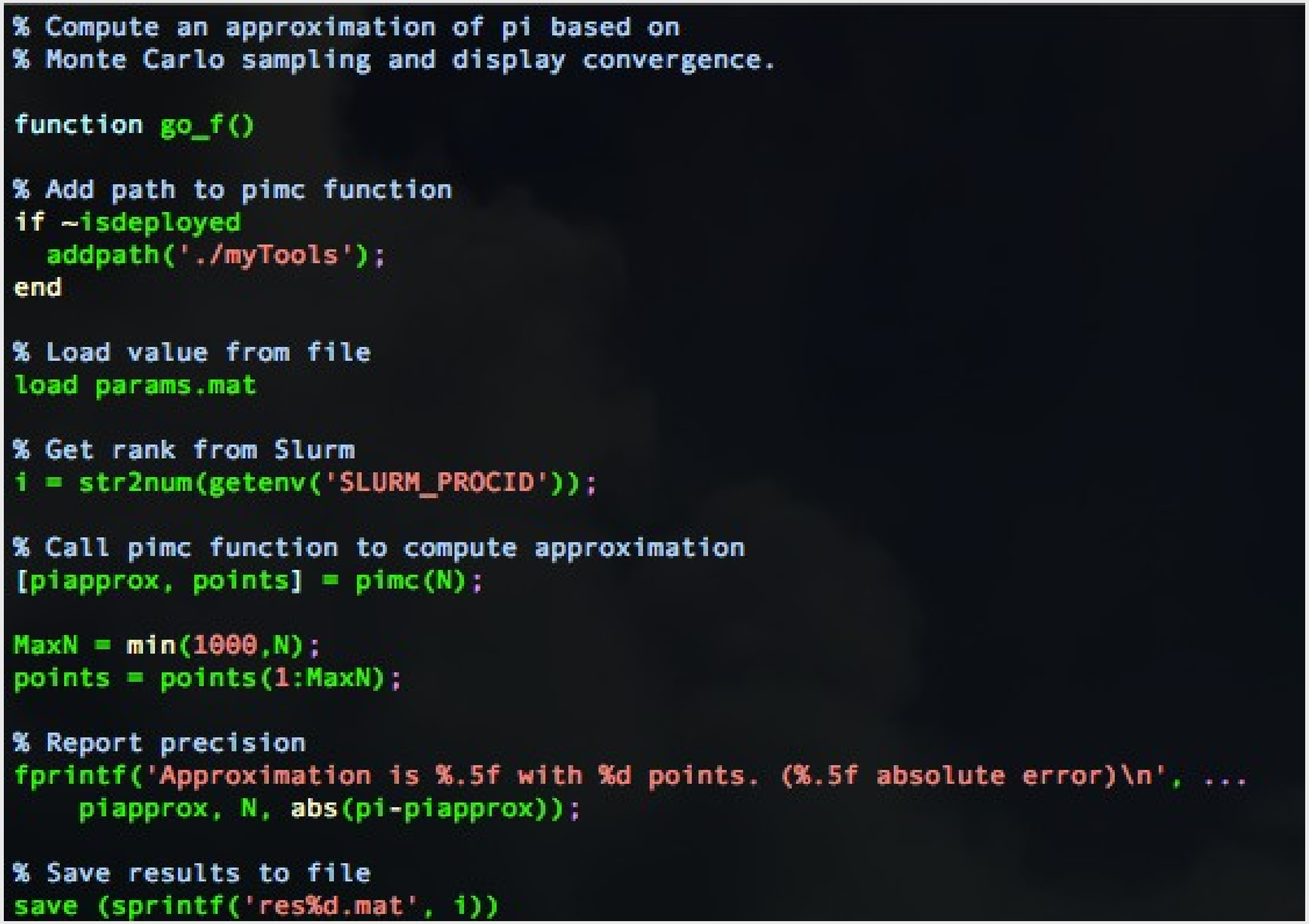
```
for i=1:10
    for j = 1:100
        %do some computation
    end
end
```

```
i = str2num(getenv('SLURM_PROCID'));
for j=1:100
    % do some computation
end
```

CISM

Little effort: embarrassingly parallel

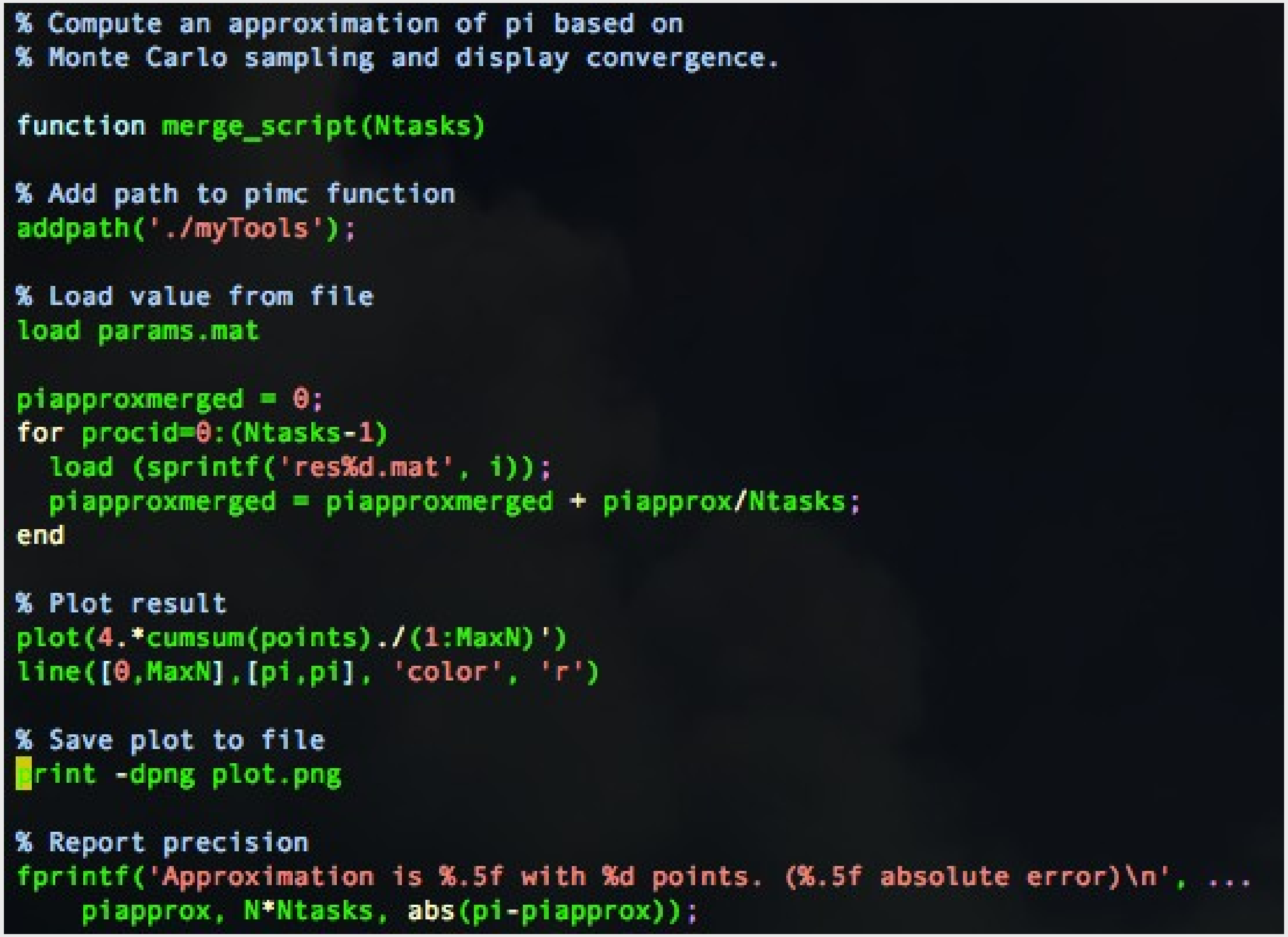
Outer-loop splitting with Slurm's srun



```
% Compute an approximation of pi based on
% Monte Carlo sampling and display convergence.

function go_f()

% Add path to pimc function
if ~isdeployed
  addpath('./myTools');
end

% Load value from file
load params.mat

% Get rank from Slurm
i = str2num(getenv('SLURM_PROCID'));

% Call pimc function to compute approximation
[piapprox, points] = pimc(N);

MaxN = min(1000,N);
points = points(1:MaxN);

% Report precision
fprintf('Approximation is %.5f with %d points. (%.5f absolute error)\n', ...
    piapprox, N, abs(pi-piapprox));

% Save results to file
save (sprintf('res%d.mat', i))
```

Little effort: embarrassingly parallel

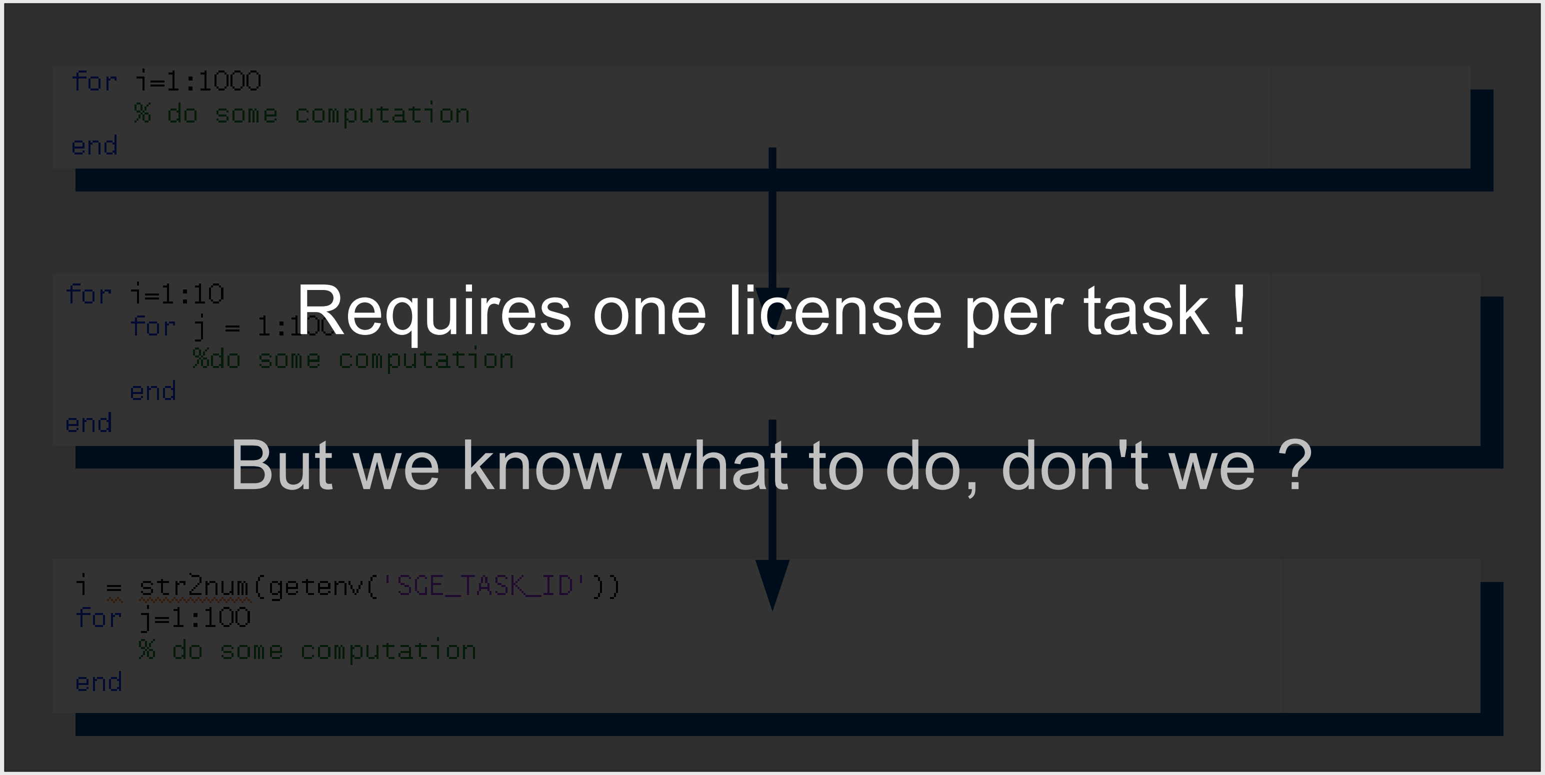
Outer-loop splitting with Slurm's srun



```
% Compute an approximation of pi based on
% Monte Carlo sampling and display convergence.

function merge_script(Ntasks)

% Add path to pimc function
addpath('./myTools');

% Load value from file
load params.mat

piapproxmerged = 0;
for procid=0:(Ntasks-1)
  load (sprintf('res%d.mat', i));
  piapproxmerged = piapproxmerged + piapprox/Ntasks;
end

% Plot result
plot(4.*cumsum(points)./(1:MaxN)')
line([0,MaxN],[pi,pi], 'color', 'r')

% Save plot to file
print -dpng plot.png

% Report precision
fprintf('Approximation is %.5f with %d points. (%.5f absolute error)\n', ...
    piapprox, N*Ntasks, abs(pi-piapprox));
```

Little effort: embarrassingly parallel

Outer-loop splitting with Slurm's srun



```
for i=1:1000
    % do some computation
end
```

```
for i=1:10
    for j = 1:100
        %do some computation
    end
end
```

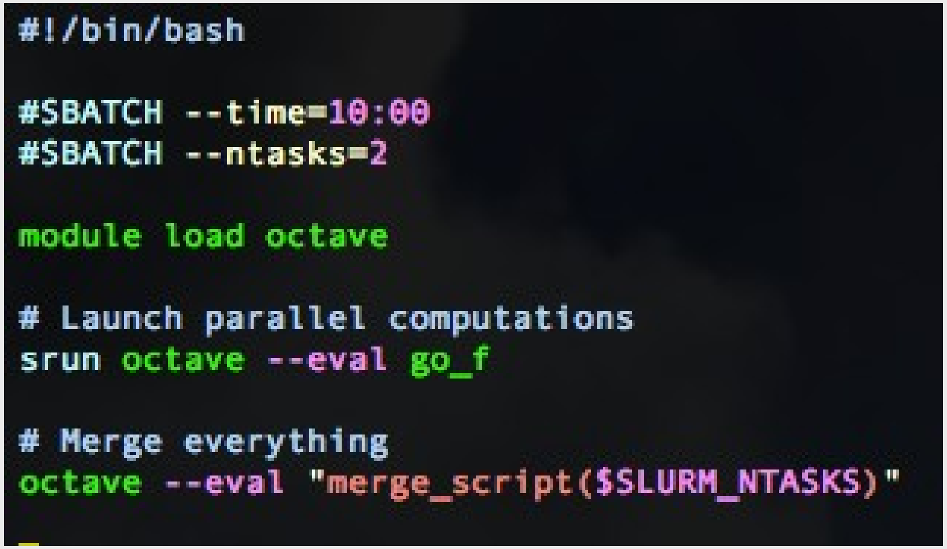
```
i = str2num(getenv('SGE_TASK_ID'))
for j=1:100
    % do some computation
end
```

Requires one license per task !

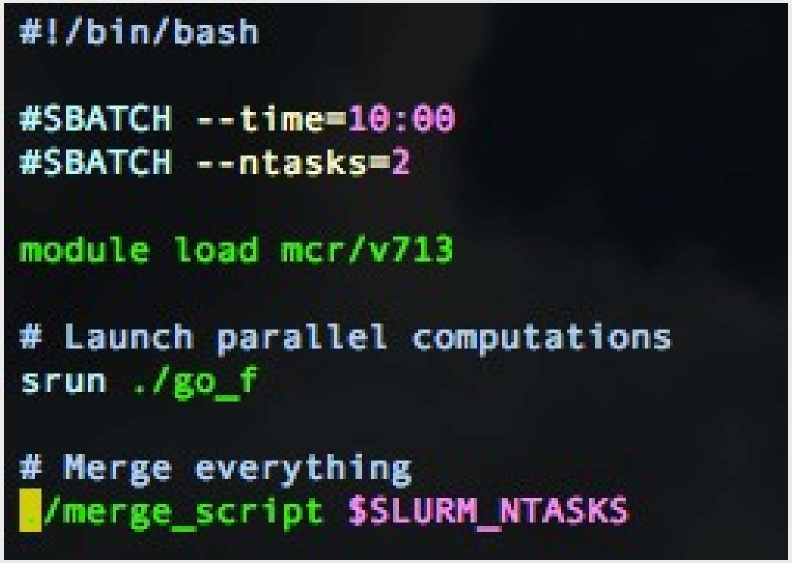
But we know what to do, don't we ?

Little effort: embarrassingly parallel

Outer-loop splitting with Slurm's srun



```
#!/bin/bash

#SBATCH --time=10:00
#SBATCH --ntasks=2

module load octave

# Launch parallel computations
srun octave --eval go_f

# Merge everything
octave --eval "merge_script($SLURM_NTASKS)"
```

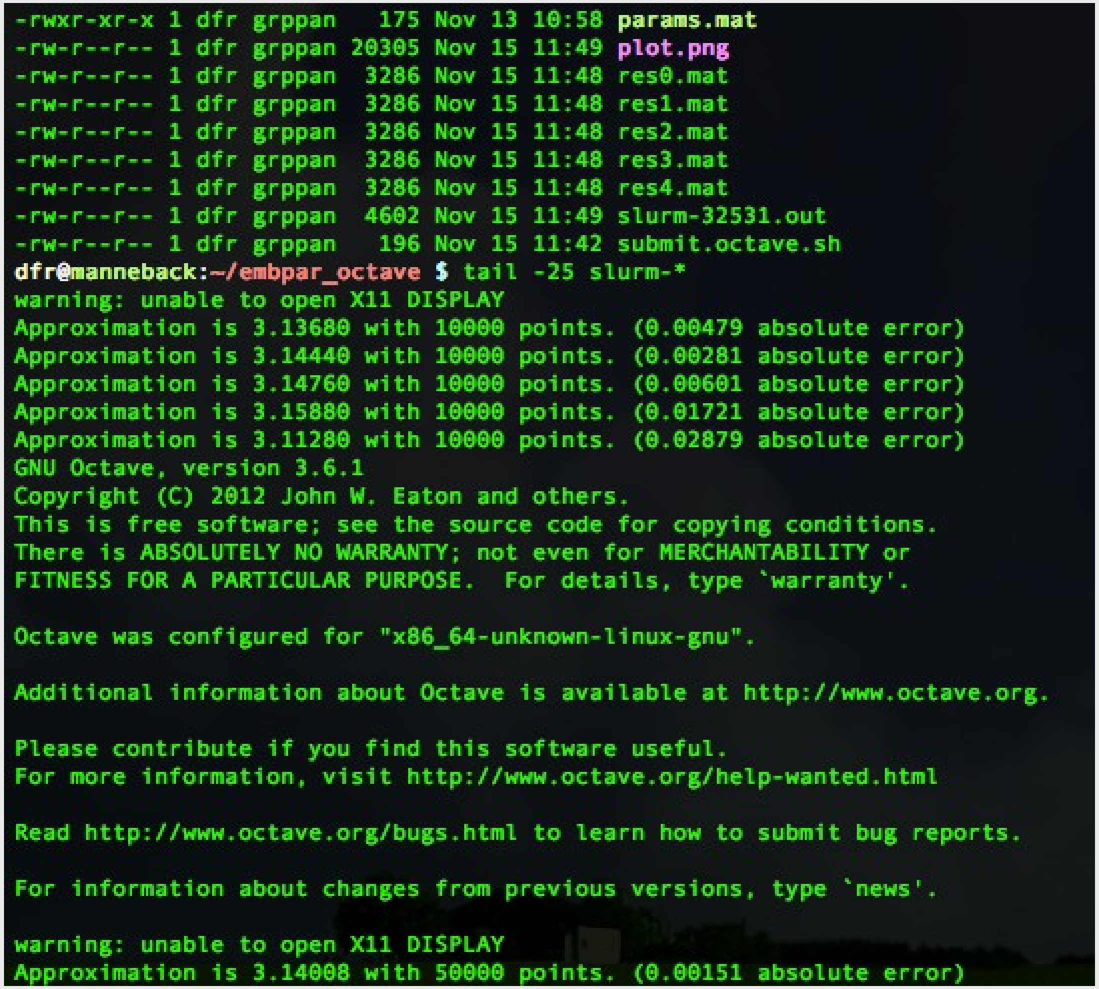


```
#!/bin/bash

#SBATCH --time=10:00
#SBATCH --ntasks=2

module load mcr/v713

# Launch parallel computations
srun ./go_f

# Merge everything
./merge_script $SLURM_NTASKS
```

Try it yourself! : ~dfr/matlab/embarrassingly*

CISM

Little effort: embarrassingly parallel

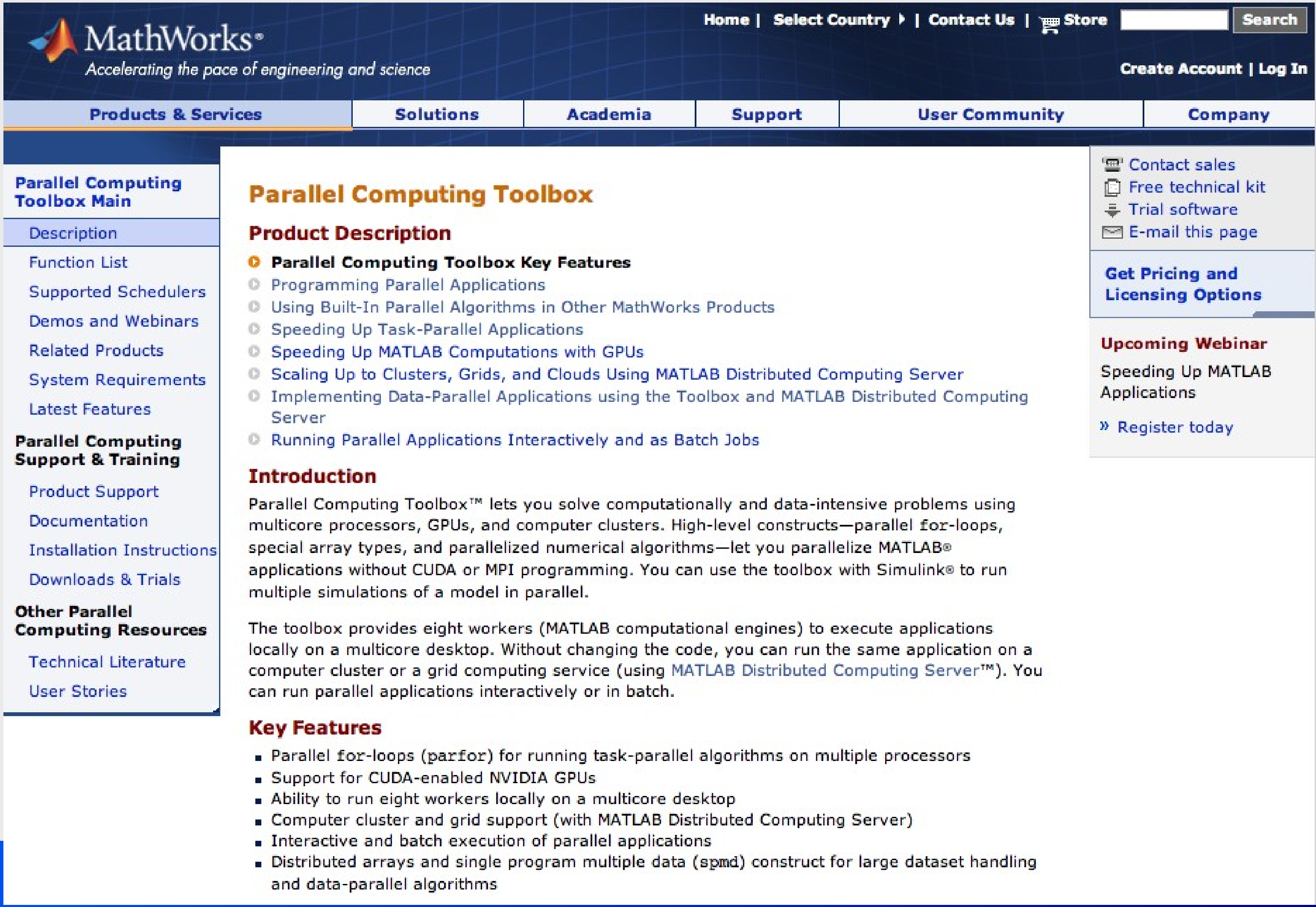


```
-rwxr-xr-x 1 dfr grppan   175 Nov 13 10:58 params.mat
-rw-r--r-- 1 dfr grppan 20305 Nov 15 11:49 plot.png
-rw-r--r-- 1 dfr grppan  3286 Nov 15 11:48 res0.mat
-rw-r--r-- 1 dfr grppan  3286 Nov 15 11:48 res1.mat
-rw-r--r-- 1 dfr grppan  3286 Nov 15 11:48 res2.mat
-rw-r--r-- 1 dfr grppan  3286 Nov 15 11:48 res3.mat
-rw-r--r-- 1 dfr grppan  3286 Nov 15 11:48 res4.mat
-rw-r--r-- 1 dfr grppan  4602 Nov 15 11:49 slurm-32531.out
-rw-r--r-- 1 dfr grppan   196 Nov 15 11:42 submit.octave.sh
dfr@manneback:~/embpar_octave $ tail -25 slurm-*
warning: unable to open X11 DISPLAY
Approximation is 3.13680 with 10000 points. (0.00479 absolute error)
Approximation is 3.14440 with 10000 points. (0.00281 absolute error)
Approximation is 3.14760 with 10000 points. (0.00601 absolute error)
Approximation is 3.15880 with 10000 points. (0.01721 absolute error)
Approximation is 3.11280 with 10000 points. (0.02879 absolute error)
GNU Octave, version 3.6.1
Copyright (C) 2012 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE.  For details, type `warranty'.

Octave was configured for "x86_64-unknown-linux-gnu".

Additional information about Octave is available at http://www.octave.org.

Please contribute if you find this software useful.
For more information, visit http://www.octave.org/help-wanted.html

Read http://www.octave.org/bugs.html to learn how to submit bug reports.

For information about changes from previous versions, type `news'.

warning: unable to open X11 DISPLAY
Approximation is 3.14008 with 50000 points. (0.00151 absolute error)
```

CISM

Little effort: embarrassingly parallel

Parallel Computing Toolbox



Parallel Computing Toolbox

Product Description

- **Parallel Computing Toolbox Key Features**
- Programming Parallel Applications
- Using Built-In Parallel Algorithms in Other MathWorks Products
- Speeding Up Task-Parallel Applications
- Speeding Up MATLAB Computations with GPUs
- Scaling Up to Clusters, Grids, and Clouds Using MATLAB Distributed Computing Server
- Implementing Data-Parallel Applications using the Toolbox and MATLAB Distributed Computing Server
- Running Parallel Applications Interactively and as Batch Jobs

Introduction

Parallel Computing Toolbox™ lets you solve computationally and data-intensive problems using multicore processors, GPUs, and computer clusters. High-level constructs—parallel for-loops, special array types, and parallelized numerical algorithms—let you parallelize MATLAB® applications without CUDA or MPI programming. You can use the toolbox with Simulink® to run multiple simulations of a model in parallel.

The toolbox provides eight workers (MATLAB computational engines) to execute applications locally on a multicore desktop. Without changing the code, you can run the same application on a computer cluster or a grid computing service (using MATLAB Distributed Computing Server™). You can run parallel applications interactively or in batch.

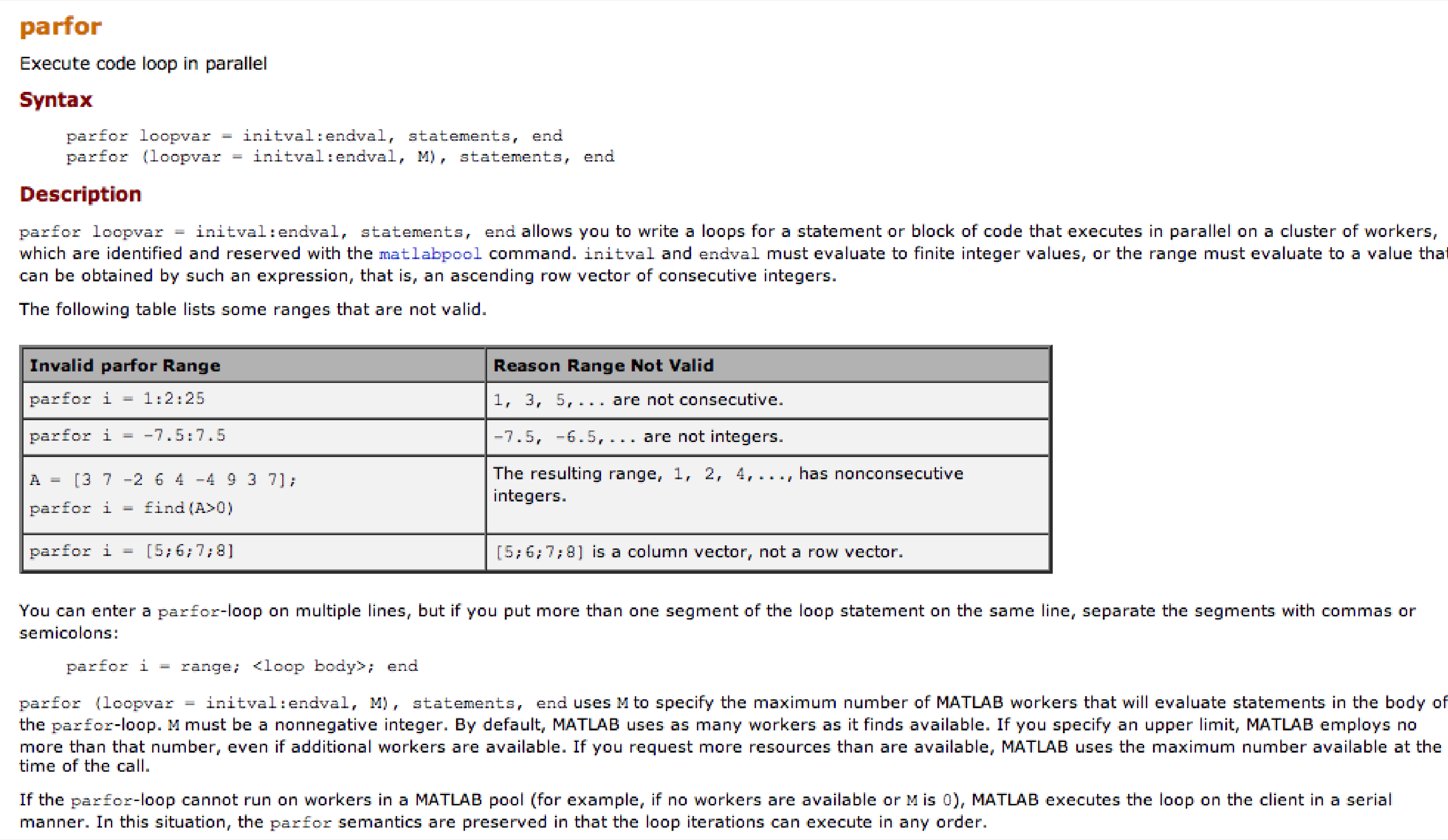
Key Features

- Parallel for-loops (parfor) for running task-parallel algorithms on multiple processors
- Support for CUDA-enabled NVIDIA GPUs
- Ability to run eight workers locally on a multicore desktop
- Computer cluster and grid support (with MATLAB Distributed Computing Server)
- Interactive and batch execution of parallel applications
- Distributed arrays and single program multiple data (spmd) construct for large dataset handling and data-parallel algorithms

CISM

Little effort: embarrassingly parallel

Parallel Computing Toolbox



parfor

Execute code loop in parallel

Syntax

```
parfor loopvar = initval:endval, statements, end
parfor (loopvar = initval:endval, M), statements, end
```

Description

`parfor loopvar = initval:endval, statements, end` allows you to write a loops for a statement or block of code that executes in parallel on a cluster of workers, which are identified and reserved with the `matlabpool` command. `initval` and `endval` must evaluate to finite integer values, or the range must evaluate to a value tha can be obtained by such an expression, that is, an ascending row vector of consecutive integers.

The following table lists some ranges that are not valid.

Invalid parfor Range	Reason Range Not Valid
`parfor i = 1:2:25`	`1, 3, 5,...` are not consecutive.
`parfor i = -7.5:7.5`	`-7.5, -6.5,...` are not integers.

`A = [3 7 -2 6 4 -4 9 3 7];`
`parfor i = find(A>0)`
	The resulting range, `1, 2, 4,...,` has nonconsecutive integers.

`parfor i = [5;6;7;8]`	`[5;6;7;8]` is a column vector, not a row vector.

You can enter a `parfor`-loop on multiple lines, but if you put more than one segment of the loop statement on the same line, separate the segments with commas or semicolons:

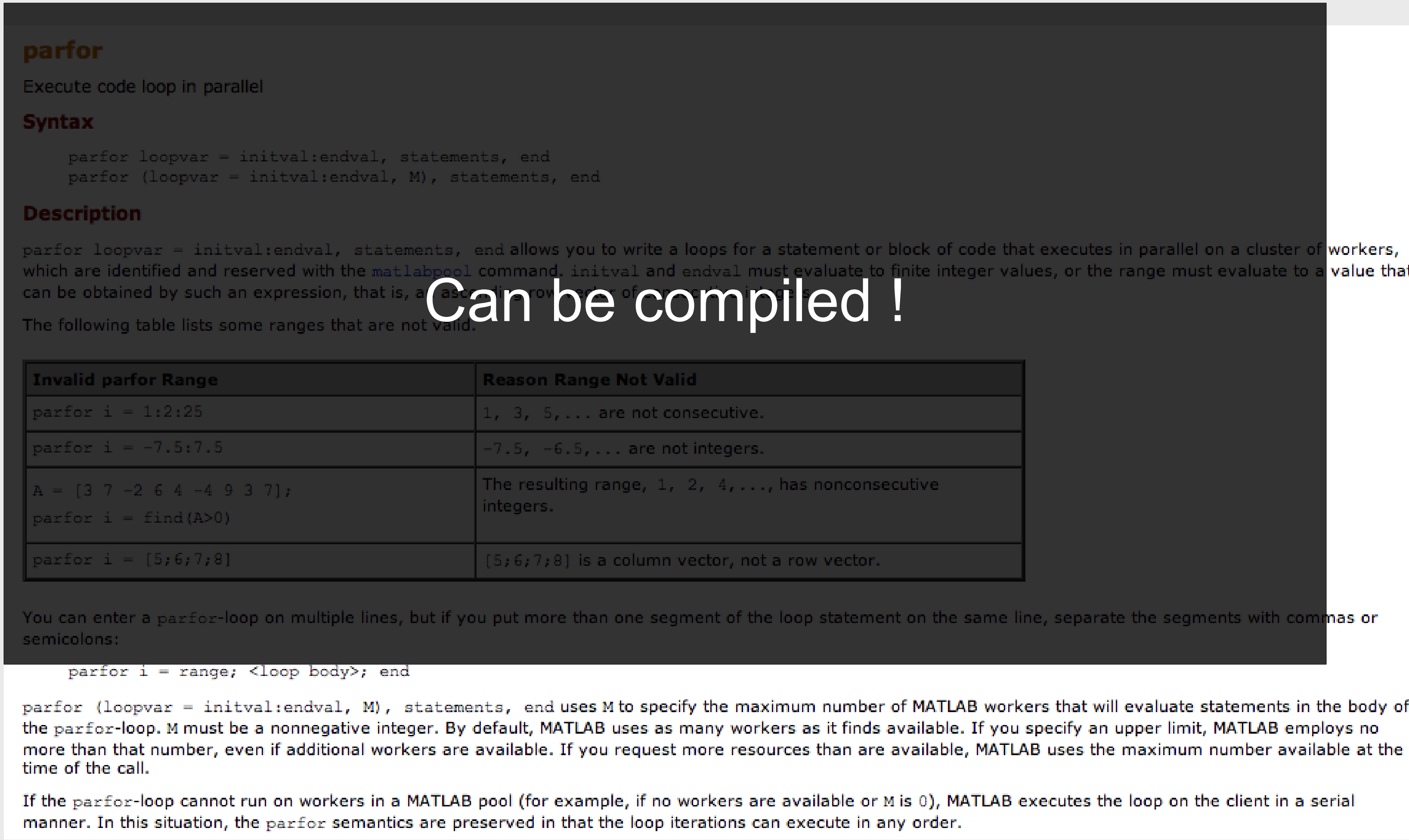
```
parfor i = range; <loop body>; end
```

`parfor (loopvar = initval:endval, M), statements, end` uses `M` to specify the maximum number of MATLAB workers that will evaluate statements in the body of the `parfor`-loop. `M` must be a nonnegative integer. By default, MATLAB uses as many workers as it finds available. If you specify an upper limit, MATLAB employs no more than that number, even if additional workers are available. If you request more resources than are available, MATLAB uses the maximum number available at the time of the call.

If the `parfor`-loop cannot run on workers in a MATLAB pool (for example, if no workers are available or `M` is `0`), MATLAB executes the loop on the client in a serial manner. In this situation, the `parfor` semantics are preserved in that the loop iterations can execute in any order.

Little effort: embarrassingly parallel

Parallel Computing Toolbox



Can be compiled !

parfor

Execute code loop in parallel

Syntax

```
    parfor loopvar = initval:endval, statements, end
    parfor (loopvar = initval:endval, M), statements, end
```

Description

`parfor loopvar = initval:endval, statements, end` allows you to write a loops for a statement or block of code that executes in parallel on a cluster of workers, which are identified and reserved with the `matlabpool` command. `initval` and `endval` must evaluate to finite integer values, or the range must evaluate to a value that can be obtained by such an expression, that is, an ascending row vector of consecutive integers.

The following table lists some ranges that are not valid.

Invalid parfor Range	Reason Range Not Valid
`parfor i = 1:2:25`	`1, 3, 5,...` are not consecutive.

`A = [3 7 -2 6 4 -4 9 3 7];`
`parfor i = find(A>0)`
	The resulting range, `1, 2, 4,...,` has nonconsecutive integers.

`parfor i = -7.5:7.5`	`-7.5, -6.5,...` are not integers.
`parfor i = [5;6;7;8]`	`[5;6;7;8]` is a column vector, not a row vector.

You can enter a `parfor`-loop on multiple lines, but if you put more than one segment of the loop statement on the same line, separate the segments with commas or semicolons:

```
    parfor i = range; <loop body>; end
```

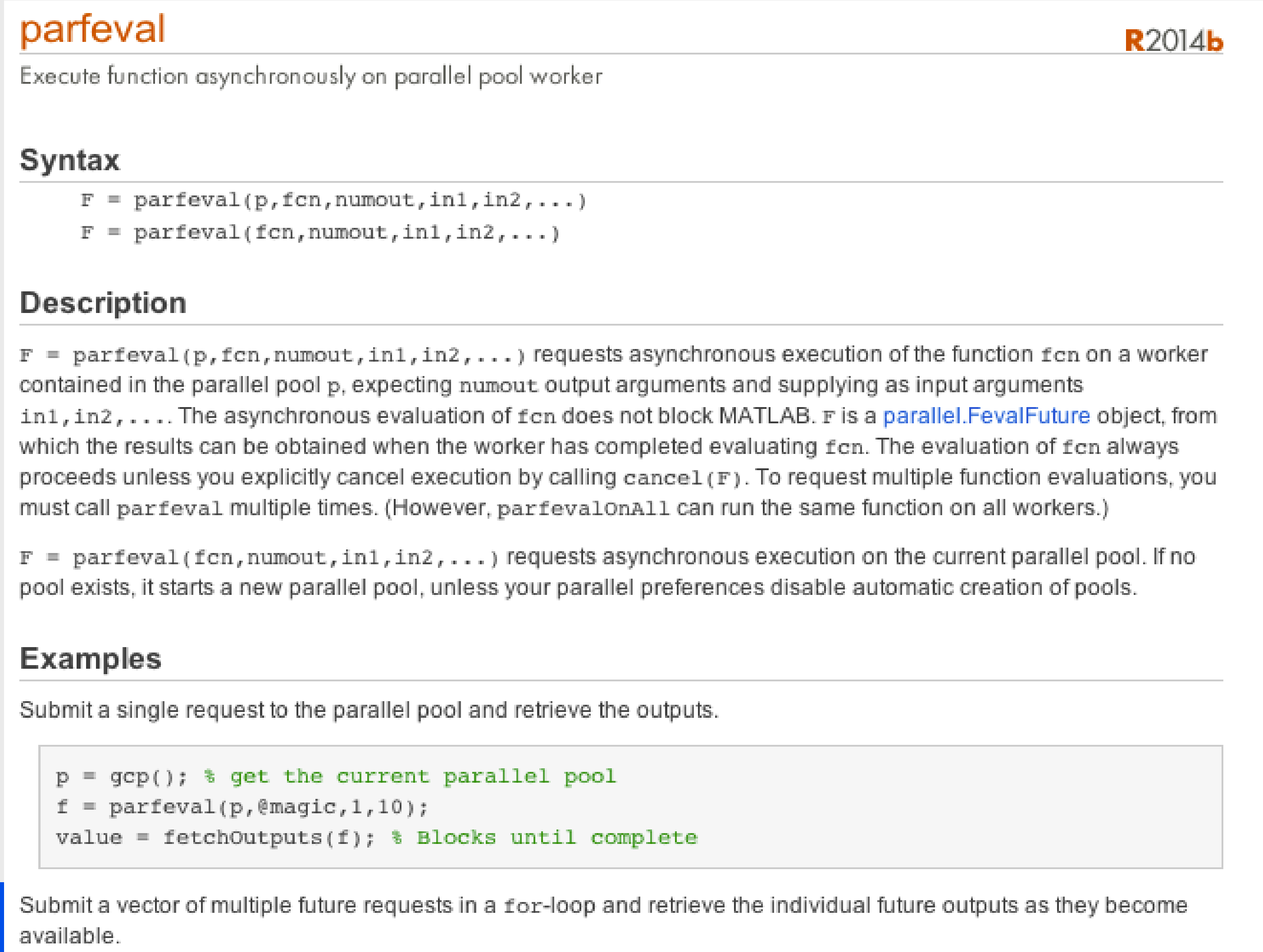
`parfor (loopvar = initval:endval, M), statements, end` uses `M` to specify the maximum number of MATLAB workers that will evaluate statements in the body of the `parfor`-loop. `M` must be a nonnegative integer. By default, MATLAB uses as many workers as it finds available. If you specify an upper limit, MATLAB employs no more than that number, even if additional workers are available. If you request more resources than are available, MATLAB uses the maximum number available at the time of the call.

If the `parfor`-loop cannot run on workers in a MATLAB pool (for example, if no workers are available or `M` is `0`), MATLAB executes the loop on the client in a serial manner. In this situation, the `parfor` semantics are preserved in that the loop iterations can execute in any order.

CISM

Little effort: embarrassingly parallel

Parallel Computing Toolbox



parfeval

R2014b

Execute function asynchronously on parallel pool worker

Syntax

```
F = parfeval(p,fcn,numout,in1,in2,...)
F = parfeval(fcn,numout,in1,in2,...)
```

Description

`F = parfeval(p,fcn,numout,in1,in2,...)` requests asynchronous execution of the function `fcn` on a worker contained in the parallel pool `p`, expecting `numout` output arguments and supplying as input arguments `in1,in2,...`. The asynchronous evaluation of `fcn` does not block MATLAB. `F` is a parallel.FevalFuture object, from which the results can be obtained when the worker has completed evaluating `fcn`. The evaluation of `fcn` always proceeds unless you explicitly cancel execution by calling `cancel(F)`. To request multiple function evaluations, you must call `parfeval` multiple times. (However, `parfevalOnAll` can run the same function on all workers.)

`F = parfeval(fcn,numout,in1,in2,...)` requests asynchronous execution on the current parallel pool. If no pool exists, it starts a new parallel pool, unless your parallel preferences disable automatic creation of pools.

Examples

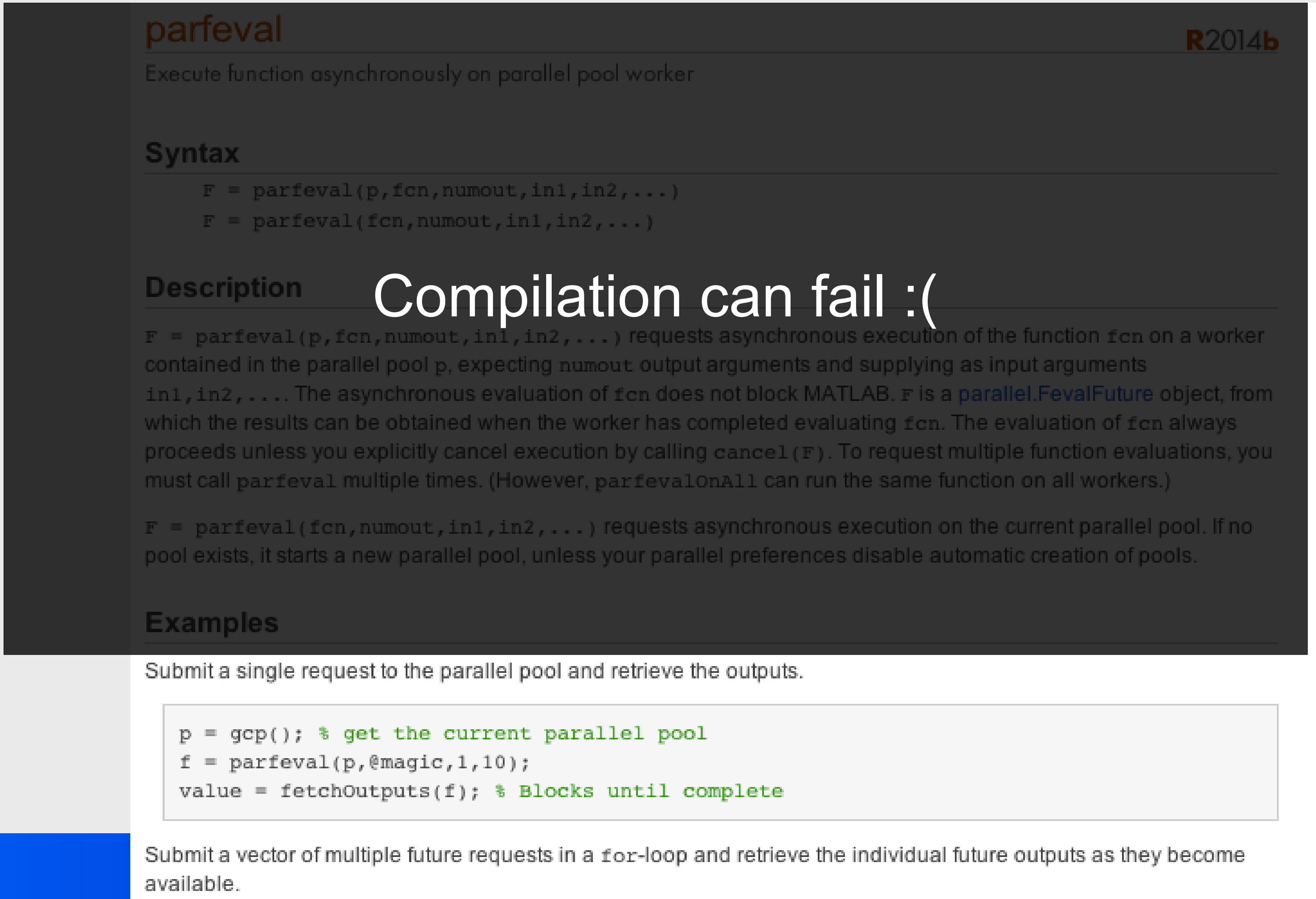
Submit a single request to the parallel pool and retrieve the outputs.

```
p = gcp(); % get the current parallel pool
f = parfeval(p,@magic,1,10);
value = fetchOutputs(f); % Blocks until complete
```

Submit a vector of multiple future requests in a `for`-loop and retrieve the individual future outputs as they become available.

Little effort: embarrassingly parallel

Parallel Computing Toolbox



parfeval R2014b

Execute function asynchronously on parallel pool worker

Syntax

```
F = parfeval(p,fcn,numout,in1,in2,...)
F = parfeval(fcn,numout,in1,in2,...)
```

Description

Compilation can fail :(

`F = parfeval(p,fcn,numout,in1,in2,...)` requests asynchronous execution of the function `fcn` on a worker contained in the parallel pool `p`, expecting `numout` output arguments and supplying as input arguments `in1,in2,...`. The asynchronous evaluation of `fcn` does not block MATLAB. `F` is a parallel.FevalFuture object, from which the results can be obtained when the worker has completed evaluating `fcn`. The evaluation of `fcn` always proceeds unless you explicitly cancel execution by calling `cancel(F)`. To request multiple function evaluations, you must call `parfeval` multiple times. (However, `parfevalOnAll` can run the same function on all workers.)

`F = parfeval(fcn,numout,in1,in2,...)` requests asynchronous execution on the current parallel pool. If no pool exists, it starts a new parallel pool, unless your parallel preferences disable automatic creation of pools.

Examples

Submit a single request to the parallel pool and retrieve the outputs.

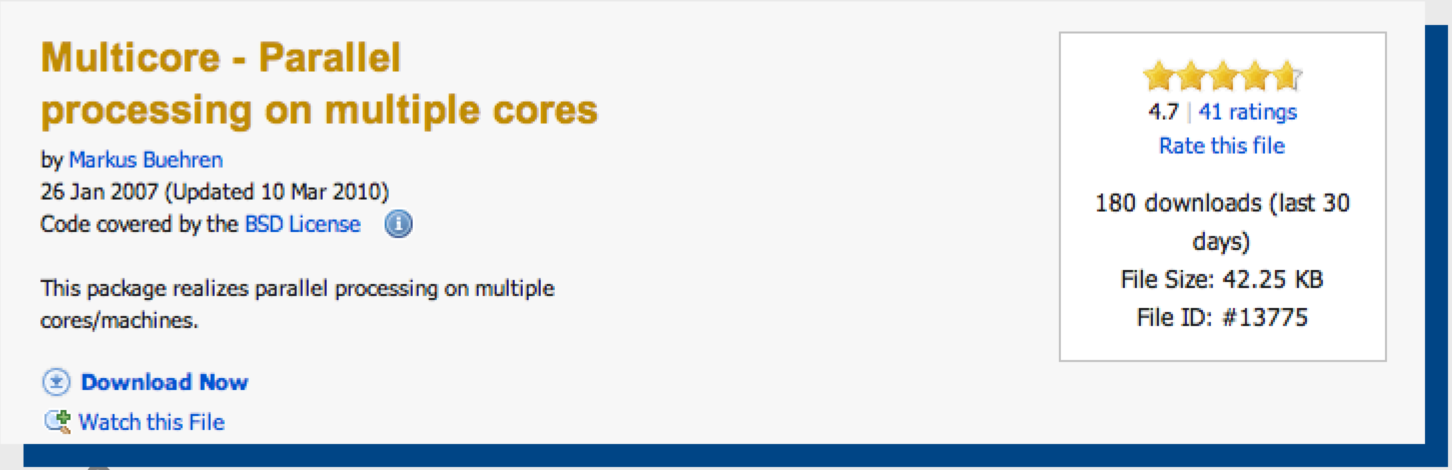
```
p = gcp(); % get the current parallel pool
f = parfeval(p,@magic,1,10);
value = fetchOutputs(f); % Blocks until complete
```

Submit a vector of multiple future requests in a `for`-loop and retrieve the individual future outputs as they become available.

CISM

Little effort: embarrassingly parallel

Matlab 3rd party peval : Multicore



Multicore - Parallel processing on multiple cores

by Markus Buehren
26 Jan 2007 (Updated 10 Mar 2010)
Code covered by the BSD License

This package realizes parallel processing on multiple cores/machines.

Download Now
Watch this File

★★★★★
4.7 | 41 ratings
Rate this file

180 downloads (last 30 days)
File Size: 42.25 KB
File ID: #13775

Compilable
uses file system
multinode

On the slaves: >> startmulticoreslave

On the master: >> for i=1:10; a{i} = rand(100,100) ;end

>> cellRes = startmulticoremaster(@eig, a)

Little effort: embarrassingly parallel

Matlab 3rd party peval : Multicore

```
>> for i=1:10; a{i} = rand(100,100) ;end
>> a

a =

  Columns 1 through 4

    [100x100 double]    [100x100 double]    [100x100 double]    [100x100
double]

  Columns 5 through 8

    [100x100 double]    [100x100 double]    [100x100 double]    [100x100
double]

  Columns 9 through 10

    [100x100 double]    [100x100 double]
```

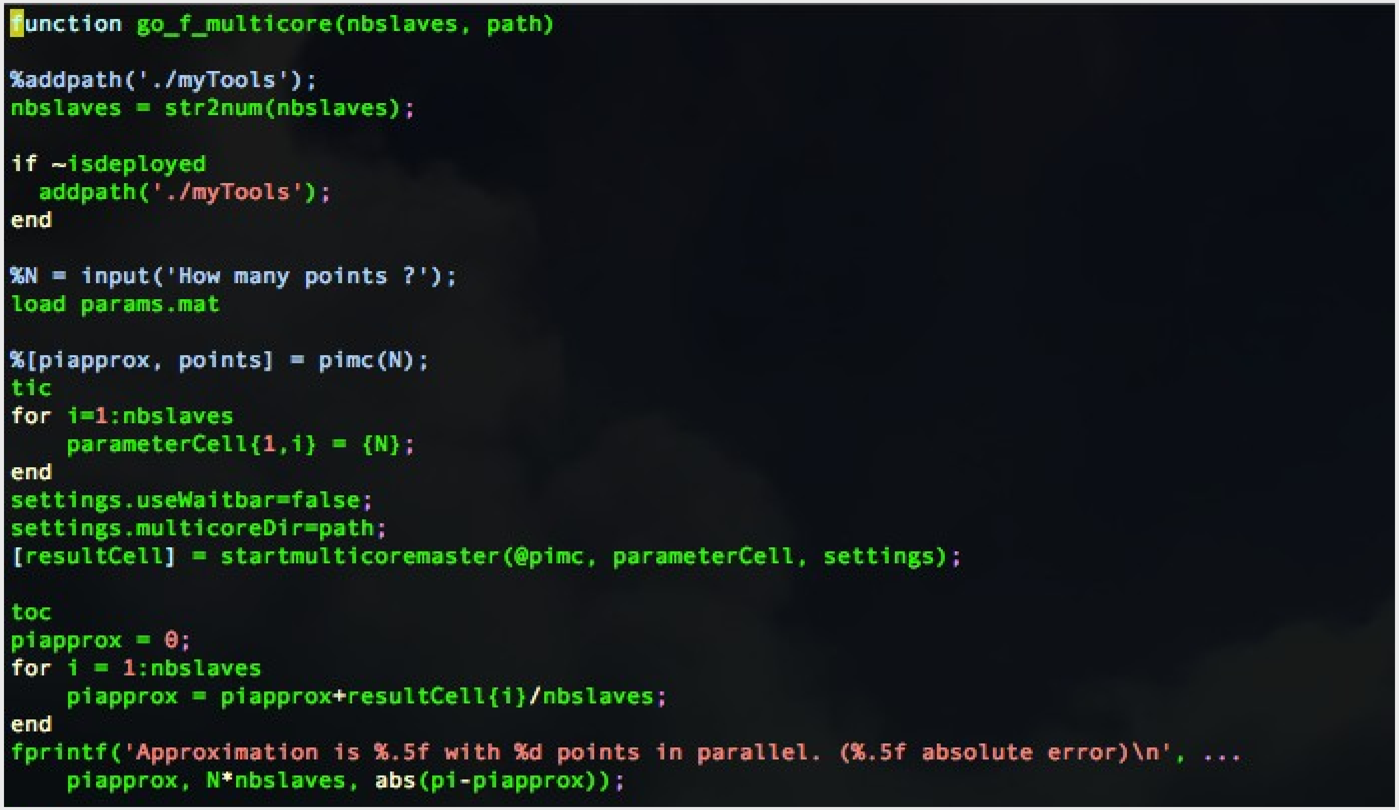
Little effort: embarrassingly parallel

Matlab 3rd party peval : Multicore

```
>> cellRes = startmulticoremaster(@eig, a)

cellRes =

  Columns 1 through 4

    [100x1 double]    [100x1 double]    [100x1 double]    [100x1 double]

  Columns 5 through 8

    [100x1 double]    [100x1 double]    [100x1 double]    [100x1 double]

  Columns 9 through 10

    [100x1 double]    [100x1 double]
```

cellRes{i} = eig(a{i})

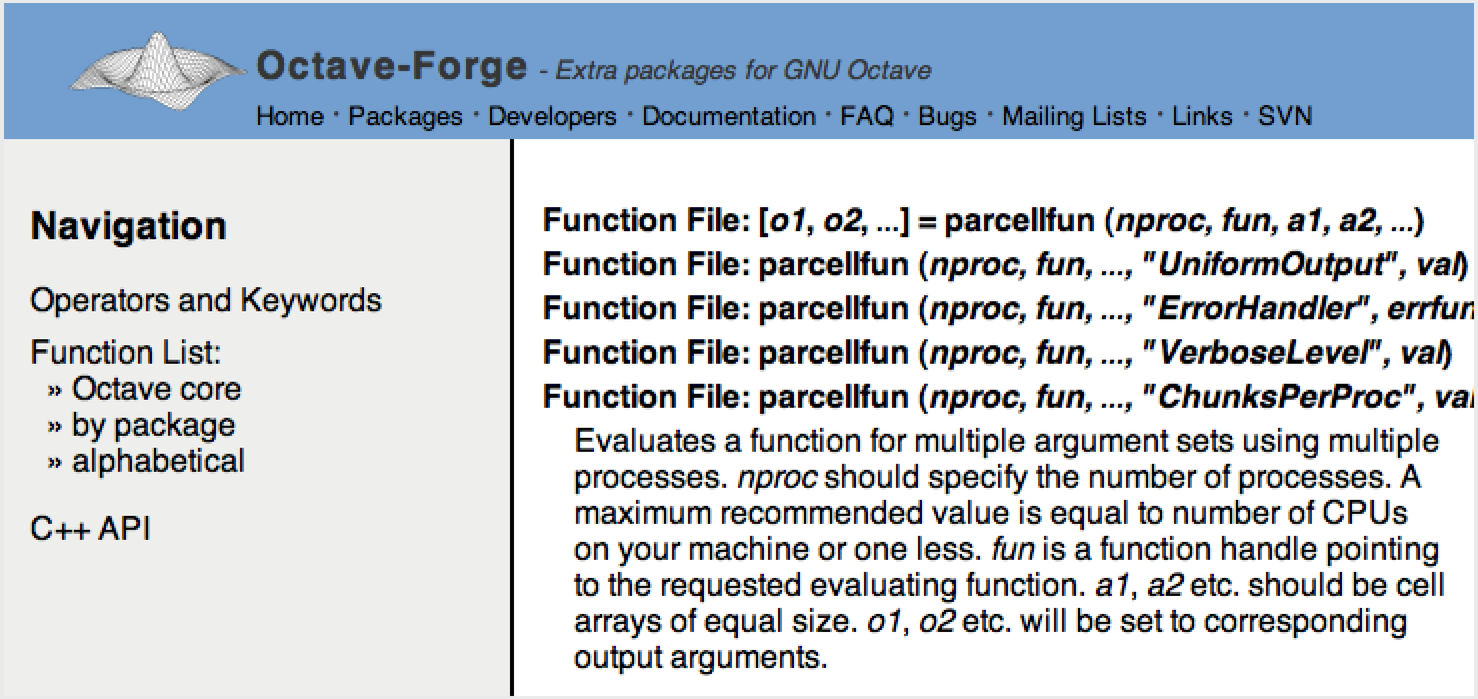
Little effort: embarrassingly parallel



```
function go_f_multicore(nbslaves, path)

%addpath('./myTools');
nbslaves = str2num(nbslaves);

if ~isdeployed
  addpath('./myTools');
end

%N = input('How many points ?');
load params.mat

%[piapprox, points] = pimc(N);
tic
for i=1:nbslaves
    parameterCell{1,i} = {N};
end
settings.useWaitbar=false;
settings.multicoreDir=path;
[resultCell] = startmulticoremaster(@pimc, parameterCell, settings);

toc
piapprox = 0;
for i = 1:nbslaves
    piapprox = piapprox+resultCell{i}/nbslaves;
end
fprintf('Approximation is %.5f with %d points in parallel. (%.5f absolute error)\n', ...
    piapprox, N*nbslaves, abs(pi-piapprox));
```

Little effort: embarrassingly parallel

Octave peval: parcellfun, pararrayfun



Octave-Forge - *Extra packages for GNU Octave*
Home · Packages · Developers · Documentation · FAQ · Bugs · Mailing Lists · Links · SVN
Navigation
Operators and Keywords
Function List:
» Octave core
» by package
» alphabetical
C++ API

Function File: [*o1, o2, ...*] = parcellfun (*nproc, fun, a1, a2, ...*)
Function File: parcellfun (*nproc, fun, ..., "UniformOutput", val*)
Function File: parcellfun (*nproc, fun, ..., "ErrorHandler", errfun*
Function File: parcellfun (*nproc, fun, ..., "VerboseLevel", val*)
Function File: parcellfun (*nproc, fun, ..., "ChunksPerProc", val*

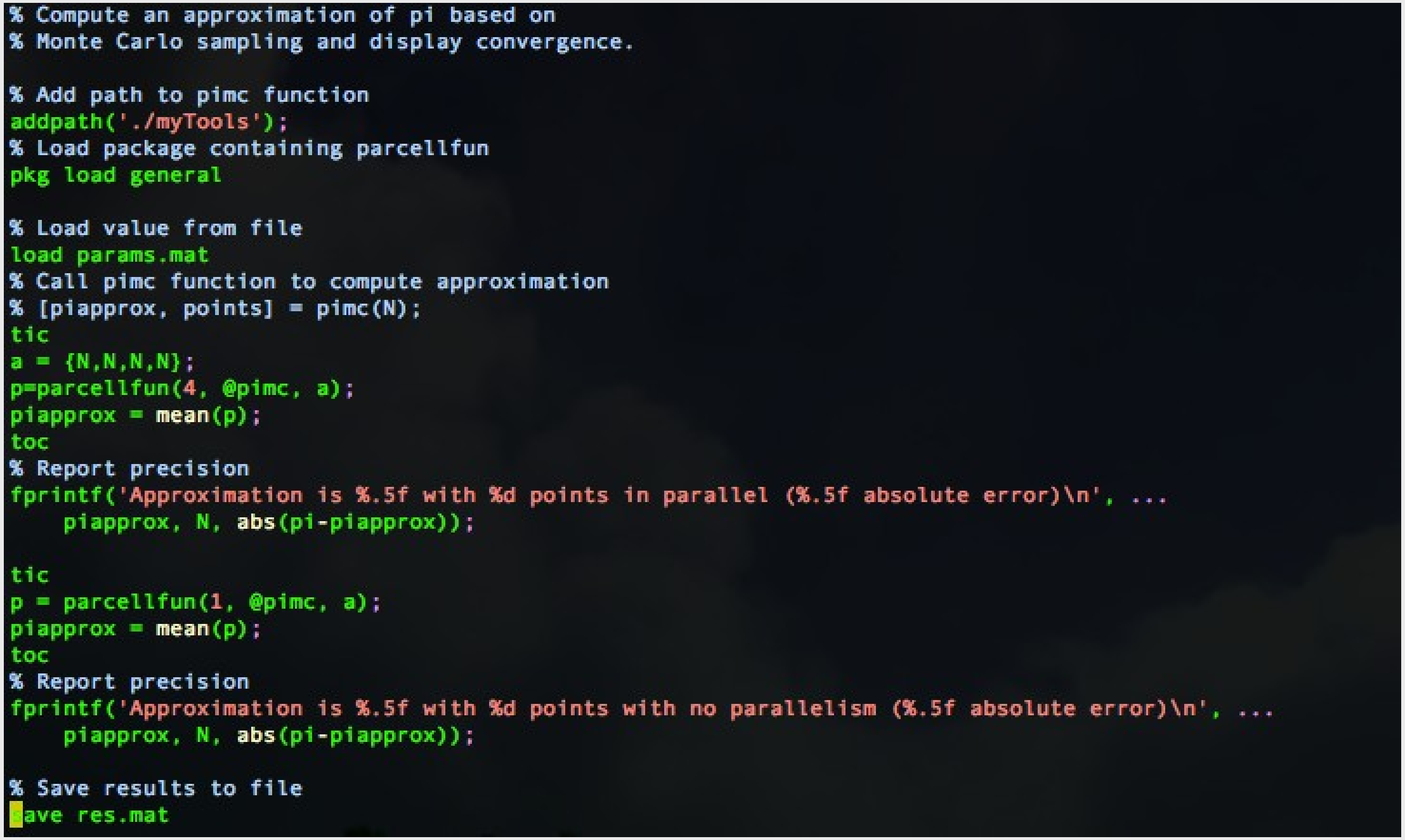
Evaluates a function for multiple argument sets using multiple processes. *nproc* should specify the number of processes. A maximum recommended value is equal to number of CPUs on your machine or one less. *fun* is a function handle pointing to the requested evaluating function. *a1*, *a2* etc. should be cell arrays of equal size. *o1*, *o2* etc. will be set to corresponding output arguments.

```
A = {rand(100,100), rand(100,100), rand(100,100)}

Res = parcellfun(2, @eig, A)
```

Multicore

Little effort: embarrassingly parallel

Octave peval: parcellfun, pararrayfun



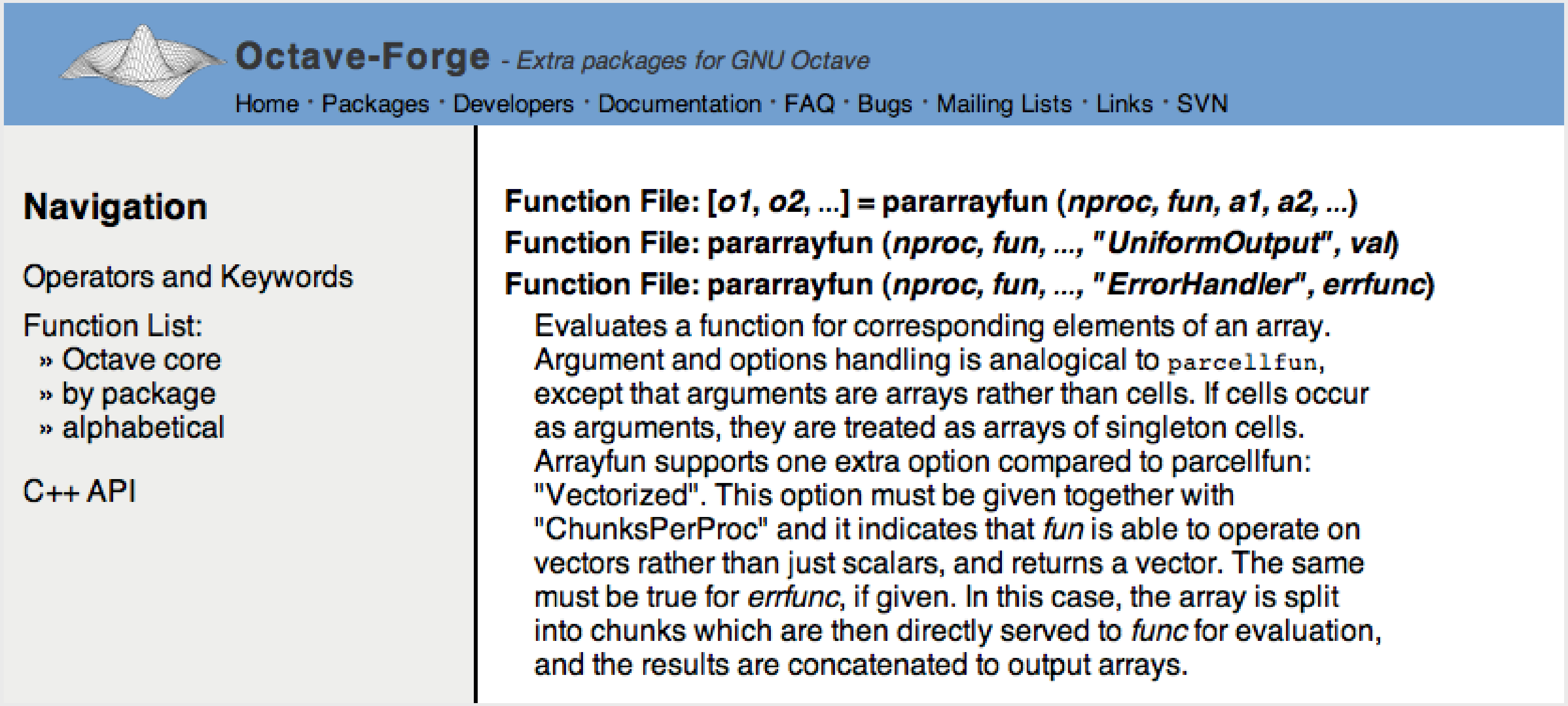
```
% Compute an approximation of pi based on
% Monte Carlo sampling and display convergence.

% Add path to pimc function
addpath('./myTools');
% Load package containing parcellfun
pkg load general

% Load value from file
load params.mat
% Call pimc function to compute approximation
% [piapprox, points] = pimc(N);
tic
a = {N,N,N,N};
p=parcellfun(4, @pimc, a);
piapprox = mean(p);
toc
% Report precision
fprintf('Approximation is %.5f with %d points in parallel (%.5f absolute error)\n', ...
    piapprox, N, abs(pi-piapprox));

tic
p = parcellfun(1, @pimc, a);
piapprox = mean(p);
toc
% Report precision
fprintf('Approximation is %.5f with %d points with no parallelism (%.5f absolute error)\n', ...
    piapprox, N, abs(pi-piapprox));

% Save results to file
save res.mat
```

```
srun --ntasks=1 --cpus-per-task=4 octave < go_f_parcellfun.m
```

CISM

Little effort: embarrassingly parallel

Octave peval: parcellfun, pararrayfun


Octave-Forge - Extra packages for GNU Octave
Home · Packages · Developers · Documentation · FAQ · Bugs · Mailing Lists · Links · SVN
Navigation
Operators and Keywords
Function List:
» Octave core
» by package
» alphabetical
C++ API

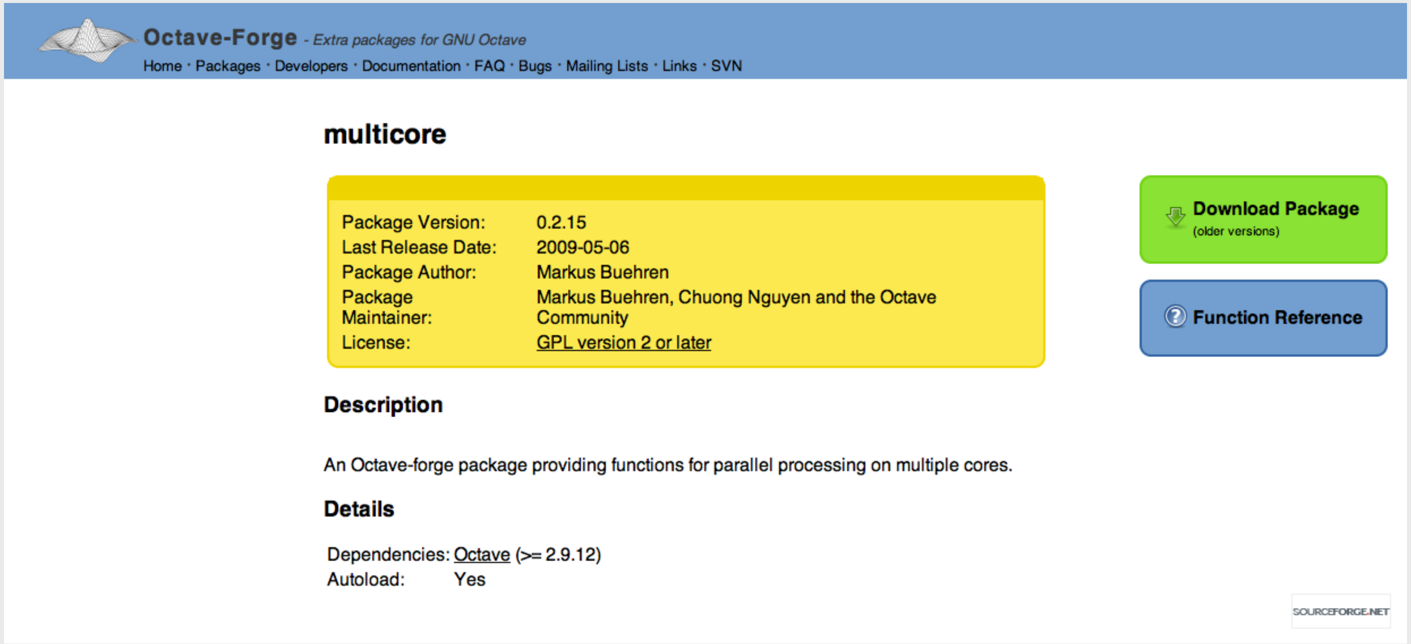
Function File: [*o1*, *o2*, ...] = pararrayfun (*nproc*, *fun*, *a1*, *a2*, ...)
Function File: pararrayfun (*nproc*, *fun*, ..., *"UniformOutput"*, *val*)
Function File: pararrayfun (*nproc*, *fun*, ..., *"ErrorHandler"*, *errfunc*)

Evaluates a function for corresponding elements of an array. Argument and options handling is analogical to `parcellfun`, except that arguments are arrays rather than cells. If cells occur as arguments, they are treated as arrays of singleton cells. Arrayfun supports one extra option compared to parcellfun: "Vectorized". This option must be given together with "ChunksPerProc" and it indicates that *fun* is able to operate on vectors rather than just scalars, and returns a vector. The same must be true for *errfunc*, if given. In this case, the array is split into chunks which are then directly served to *func* for evaluation, and the results are concatenated to output arrays.

```
A = rand(100,100,3)

Res = pararrayfun(2, @eig, A)
```

Little effort: embarrassingly parallel

Octave peval: multicore


Octave-Forge - Extra packages for GNU Octave
Home · Packages · Developers · Documentation · FAQ · Bugs · Mailing Lists · Links · SVN
Download Package
(older versions)
Function Reference
SOURCEFORGE.NET

multicore

Package Version:	0.2.15
Last Release Date:	2009-05-06
Package Author:	Markus Buehren
Package Maintainer:	Markus Buehren, Chuong Nguyen and the Octave Community
License:	GPL version 2 or later

Description

An Octave-forge package providing functions for parallel processing on multiple cores.

Details

Dependencies:	Octave (>= 2.9.12)
Autoload:	Yes

multinode

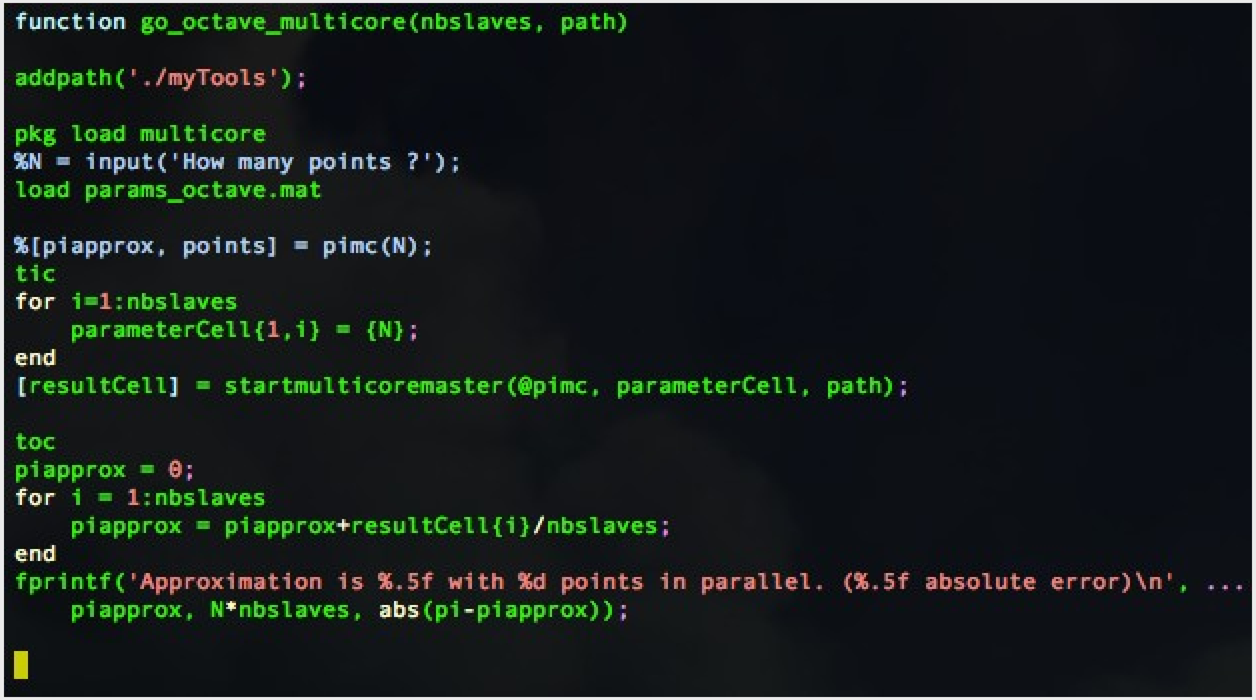
On the slaves: `>> startmulticoreslaves`

On the master:

```
>> for i=1:10; a{i} = rand(100,100) ;end

>> cellRes = startmulticoremaster(@eig, a)
```

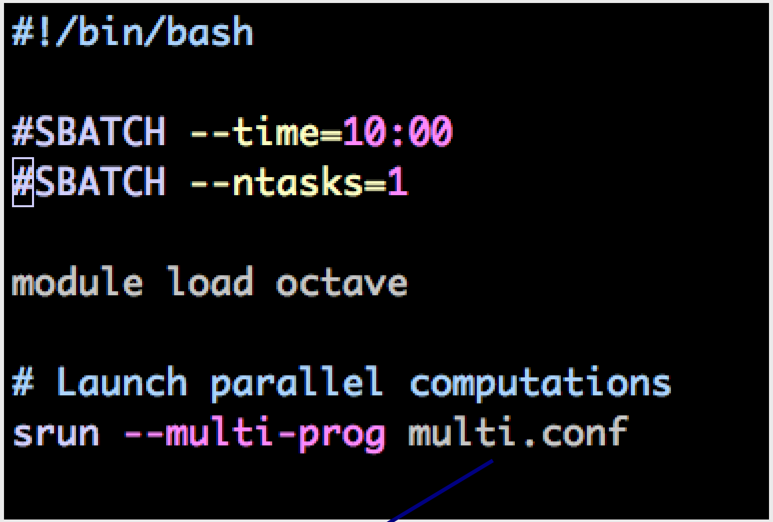
Little effort: embarrassingly parallel

Octave peval: multicore

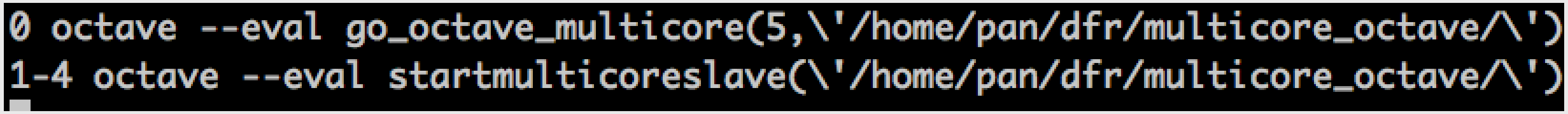


```
function go_octave_multicore(nbslaves, path)

addpath('./myTools');

pkg load multicore
%N = input('How many points ?');
load params_octave.mat

%[piapprox, points] = pimc(N);
tic
for i=1:nbslaves
    parameterCell{1,i} = {N};
end
[resultCell] = startmulticoremaster(@pimc, parameterCell, path);

toc
piapprox = 0;
for i = 1:nbslaves
    piapprox = piapprox+resultCell{i}/nbslaves;
end
fprintf('Approximation is %.5f with %d points in parallel. (%.5f absolute error)\n', ...
    piapprox, N*nbslaves, abs(pi-piapprox));
```

Octave peval: multicore



```
#!/bin/bash

#SBATCH --time=10:00
#SBATCH --ntasks=1

module load octave

# Launch parallel computations
srun --multi-prog multi.conf
```

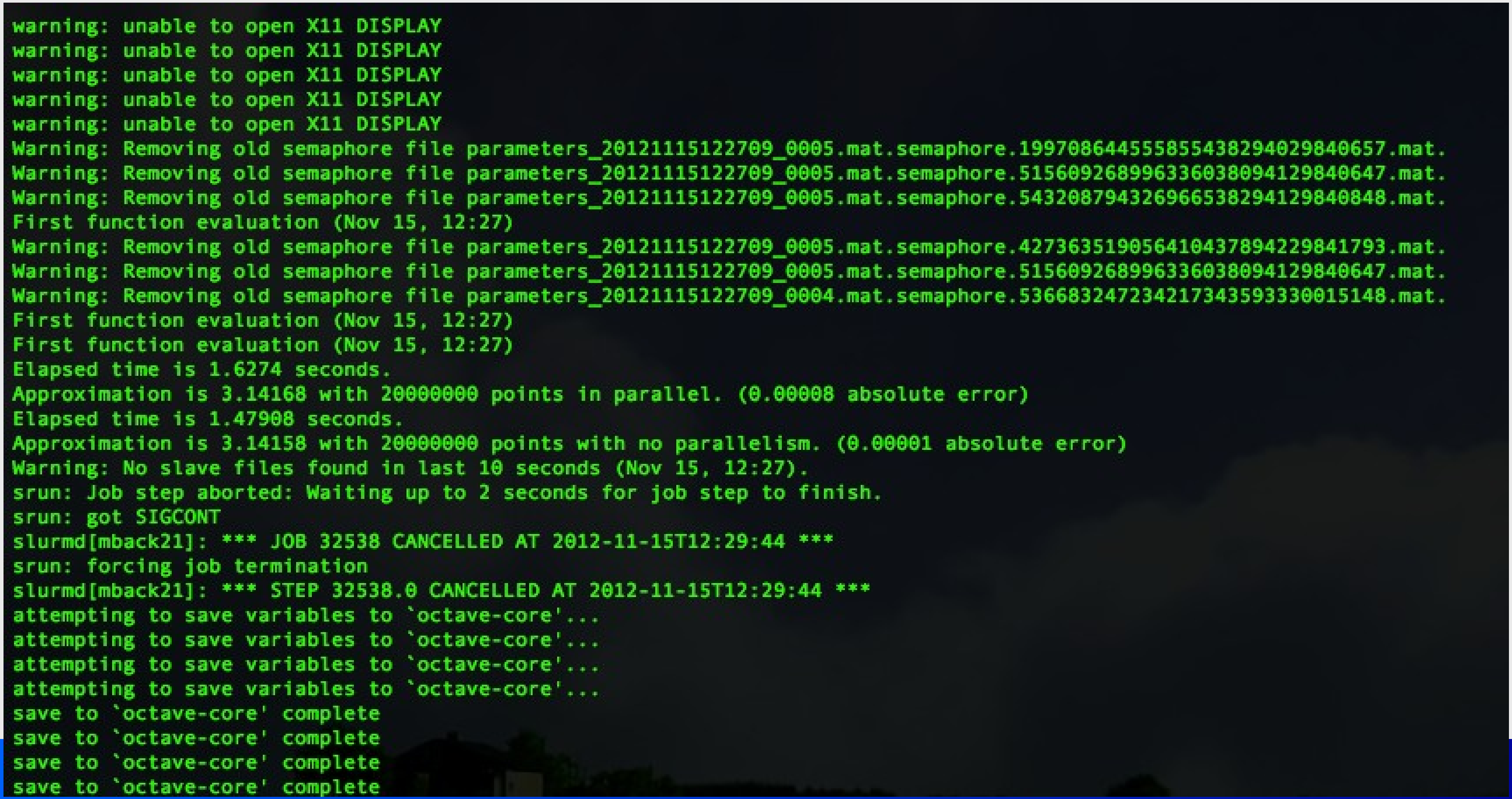


```
0 octave --eval go_octave_multicore(5,\'/home/pan/dfr/multicore_octave/\')
1-4 octave --eval startmulticoreslave(\'/home/pan/dfr/multicore_octave/\')
```

Try it ! ~dfr/matlab/multicore_octave

Little effort: embarrassingly parallel

Octave peval: multicore



```
warning: unable to open X11 DISPLAY
warning: unable to open X11 DISPLAY
warning: unable to open X11 DISPLAY
warning: unable to open X11 DISPLAY
warning: unable to open X11 DISPLAY
Warning: Removing old semaphore file parameters_20121115122709_0005.mat.semaphore.199708644555855438294029840657.mat.
Warning: Removing old semaphore file parameters_20121115122709_0005.mat.semaphore.515609268996336038094129840647.mat.
Warning: Removing old semaphore file parameters_20121115122709_0005.mat.semaphore.543208794326966538294129840848.mat.
First function evaluation (Nov 15, 12:27)
Warning: Removing old semaphore file parameters_20121115122709_0005.mat.semaphore.427363519056410437894229841793.mat.
Warning: Removing old semaphore file parameters_20121115122709_0005.mat.semaphore.515609268996336038094129840647.mat.
Warning: Removing old semaphore file parameters_20121115122709_0004.mat.semaphore.536683247234217343593330015148.mat.
First function evaluation (Nov 15, 12:27)
First function evaluation (Nov 15, 12:27)
Elapsed time is 1.6274 seconds.
Approximation is 3.14168 with 20000000 points in parallel. (0.00008 absolute error)
Elapsed time is 1.47908 seconds.
Approximation is 3.14158 with 20000000 points with no parallelism. (0.00001 absolute error)
Warning: No slave files found in last 10 seconds (Nov 15, 12:27).
srun: Job step aborted: Waiting up to 2 seconds for job step to finish.
srun: got SIGCONT
slurmd[mback21]: *** JOB 32538 CANCELLED AT 2012-11-15T12:29:44 ***
srun: forcing job termination
slurmd[mback21]: *** STEP 32538.0 CANCELLED AT 2012-11-15T12:29:44 ***
attempting to save variables to `octave-core'...
attempting to save variables to `octave-core'...
attempting to save variables to `octave-core'...
attempting to save variables to `octave-core'...
save to `octave-core' complete
save to `octave-core' complete
save to `octave-core' complete
save to `octave-core' complete
```

Matlab on the cluster

Using Matlab in batch mode

- Launch a script, get results
- Deal with licenses

Using Matlab in parallel

- With no effort
- With little effort
- With a lot of effort

A lot of effort: explicitly parallel

Explicitly parallel programs

SPMD & MPMD

Communications handled explicitly by the user

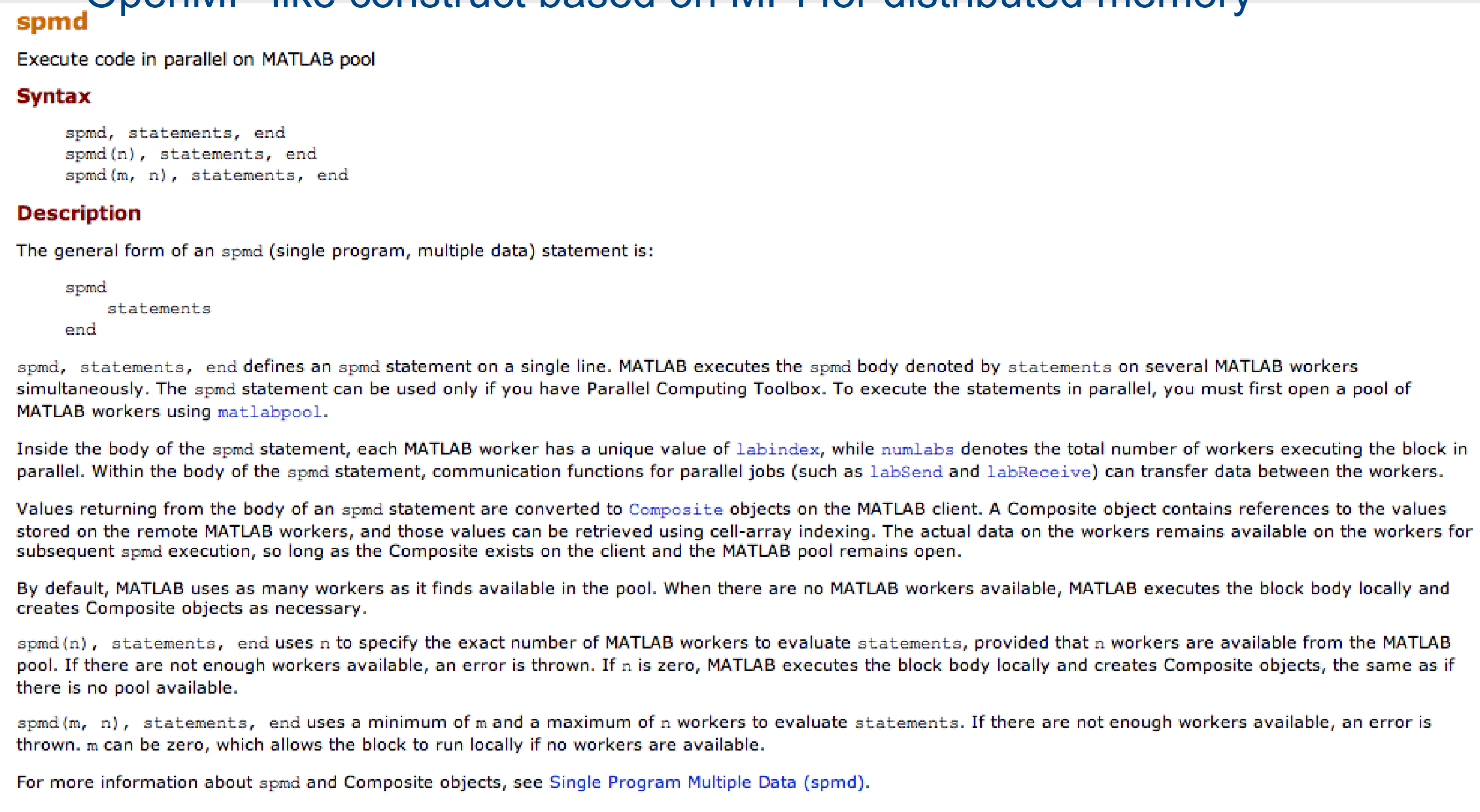
Matlab not specifically good at it..

CISM

A lot of effort: explicitly parallel

Parallel Computing Toolbox

OpenMP-like construct based on MPI for distributed memory



spmd

Execute code in parallel on MATLAB pool

Syntax

```
spmd, statements, end
spmd(n), statements, end
spmd(m, n), statements, end
```

Description

The general form of an `spmd` (single program, multiple data) statement is:

```
spmd
    statements
end
```

`spmd, statements, end` defines an `spmd` statement on a single line. MATLAB executes the `spmd` body denoted by `statements` on several MATLAB workers simultaneously. The `spmd` statement can be used only if you have Parallel Computing Toolbox. To execute the statements in parallel, you must first open a pool of MATLAB workers using `matlabpool`.

Inside the body of the `spmd` statement, each MATLAB worker has a unique value of `labindex`, while `numlabs` denotes the total number of workers executing the block in parallel. Within the body of the `spmd` statement, communication functions for parallel jobs (such as `labSend` and `labReceive`) can transfer data between the workers.

Values returning from the body of an `spmd` statement are converted to `Composite` objects on the MATLAB client. A Composite object contains references to the values stored on the remote MATLAB workers, and those values can be retrieved using cell-array indexing. The actual data on the workers remains available on the workers for subsequent `spmd` execution, so long as the Composite exists on the client and the MATLAB pool remains open.

By default, MATLAB uses as many workers as it finds available in the pool. When there are no MATLAB workers available, MATLAB executes the block body locally and creates Composite objects as necessary.

`spmd(n), statements, end` uses `n` to specify the exact number of MATLAB workers to evaluate `statements`, provided that `n` workers are available from the MATLAB pool. If there are not enough workers available, an error is thrown. If `n` is zero, MATLAB executes the block body locally and creates Composite objects, the same as if there is no pool available.

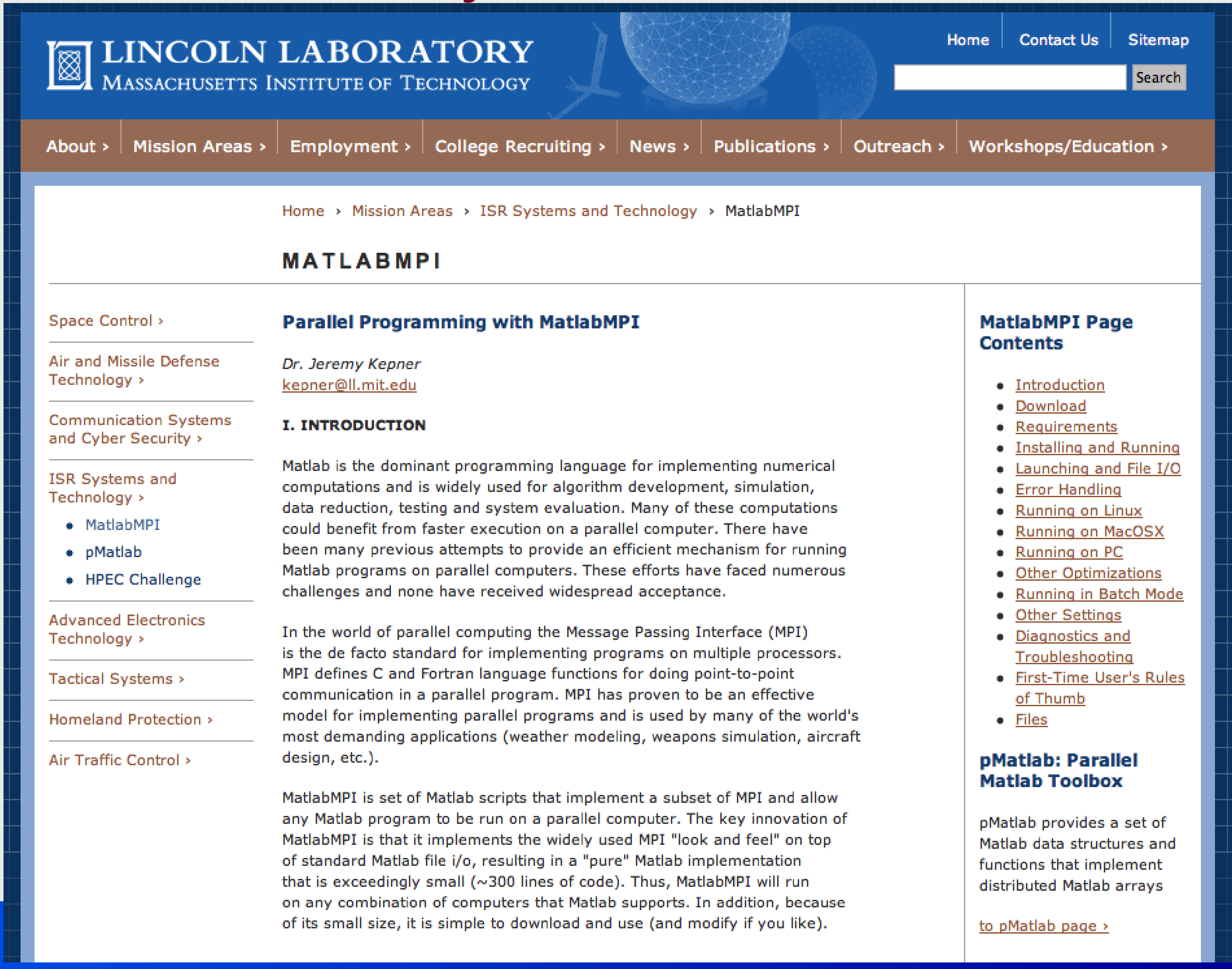
`spmd(m, n), statements, end` uses a minimum of `m` and a maximum of `n` workers to evaluate `statements`. If there are not enough workers available, an error is thrown. `m` can be zero, which allows the block to run locally if no workers are available.

For more information about `spmd` and Composite objects, see Single Program Multiple Data (spmd).

A lot of effort: explicitly parallel

Lincoln Laboratory

MatlabMPI



MATLABMPI

Parallel Programming with MatlabMPI

Dr. Jeremy Kepner
kepner@ll.mit.edu

I. INTRODUCTION

Matlab is the dominant programming language for implementing numerical computations and is widely used for algorithm development, simulation, data reduction, testing and system evaluation. Many of these computations could benefit from faster execution on a parallel computer. There have been many previous attempts to provide an efficient mechanism for running Matlab programs on parallel computers. These efforts have faced numerous challenges and none have received widespread acceptance.

In the world of parallel computing the Message Passing Interface (MPI) is the de facto standard for implementing programs on multiple processors. MPI defines C and Fortran language functions for doing point-to-point communication in a parallel program. MPI has proven to be an effective model for implementing parallel programs and is used by many of the world's most demanding applications (weather modeling, weapons simulation, aircraft design, etc.).

MatlabMPI is set of Matlab scripts that implement a subset of MPI and allow any Matlab program to be run on a parallel computer. The key innovation of MatlabMPI is that it implements the widely used MPI "look and feel" on top of standard Matlab file i/o, resulting in a "pure" Matlab implementation that is exceedingly small (~300 lines of code). Thus, MatlabMPI will run on any combination of computers that Matlab supports. In addition, because of its small size, it is simple to download and use (and modify if you like).

MatlabMPI Page Contents

pMatlab: Parallel Matlab Toolbox

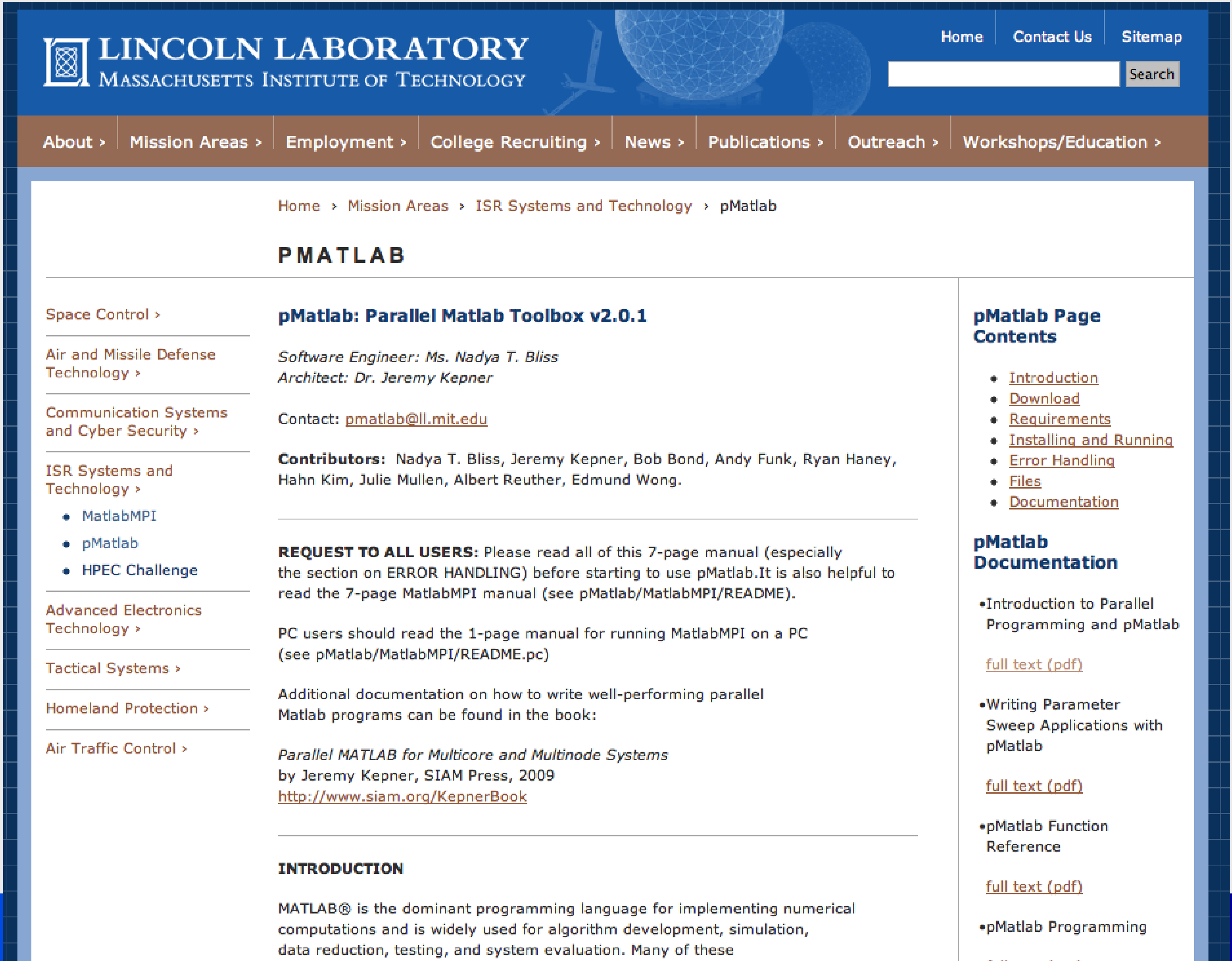
pMatlab provides a set of Matlab data structures and functions that implement distributed Matlab arrays

to pMatlab page ›

A lot of effort: explicitly parallel

Lincoln Laboratory

pMatlab



PMATLAB

pMatlab: Parallel Matlab Toolbox v2.0.1

Software Engineer: Ms. Nadya T. Bliss
Architect: Dr. Jeremy Kepner

Contact: pmatlab@ll.mit.edu

Contributors: Nadya T. Bliss, Jeremy Kepner, Bob Bond, Andy Funk, Ryan Haney, Hahn Kim, Julie Mullen, Albert Reuther, Edmund Wong.

REQUEST TO ALL USERS: Please read all of this 7-page manual (especially the section on ERROR HANDLING) before starting to use pMatlab.It is also helpful to read the 7-page MatlabMPI manual (see pMatlab/MatlabMPI/README).

PC users should read the 1-page manual for running MatlabMPI on a PC (see pMatlab/MatlabMPI/README.pc)

Additional documentation on how to write well-performing parallel Matlab programs can be found in the book:

Parallel MATLAB for Multicore and Multinode Systems
by Jeremy Kepner, SIAM Press, 2009
http://www.siam.org/KepnerBook

INTRODUCTION

MATLAB® is the dominant programming language for implementing numerical computations and is widely used for algorithm development, simulation, data reduction, testing, and system evaluation. Many of these

pMatlab Page Contents

- Introduction
- Download
- Requirements
- Installing and Running
- Error Handling
- Files
- Documentation

pMatlab Documentation

•Introduction to Parallel Programming and pMatlab

full text (pdf)

•Writing Parameter Sweep Applications with pMatlab

full text (pdf)

•pMatlab Function Reference

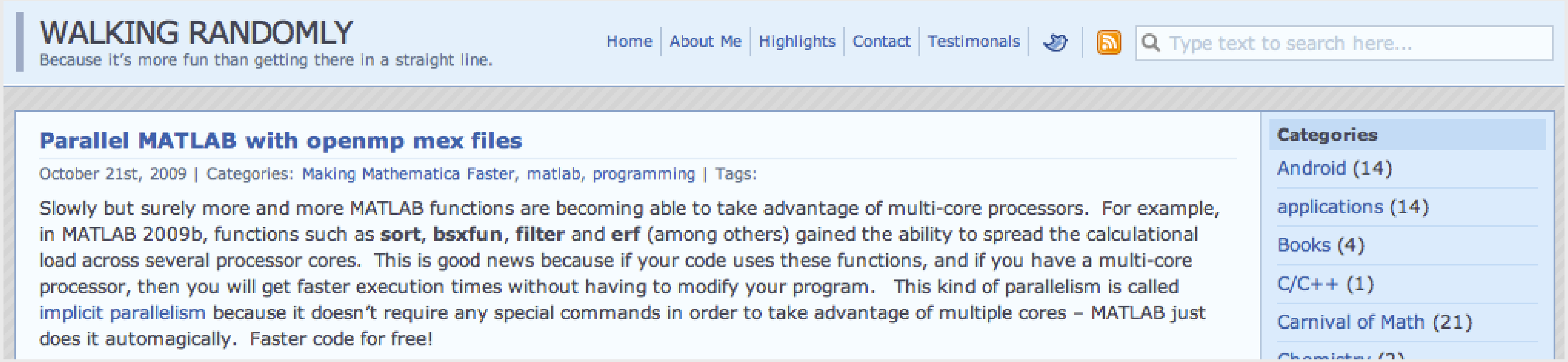
full text (pdf)

•pMatlab Programming

CISM

A lot of effort: explicitly parallel

OpenMP Mex files


WALKING RANDOMLY
Because it's more fun than getting there in a straight line.
Home
About Me
Highlights
Contact
Testimonals
Type text to search here...

Parallel MATLAB with openmp mex files

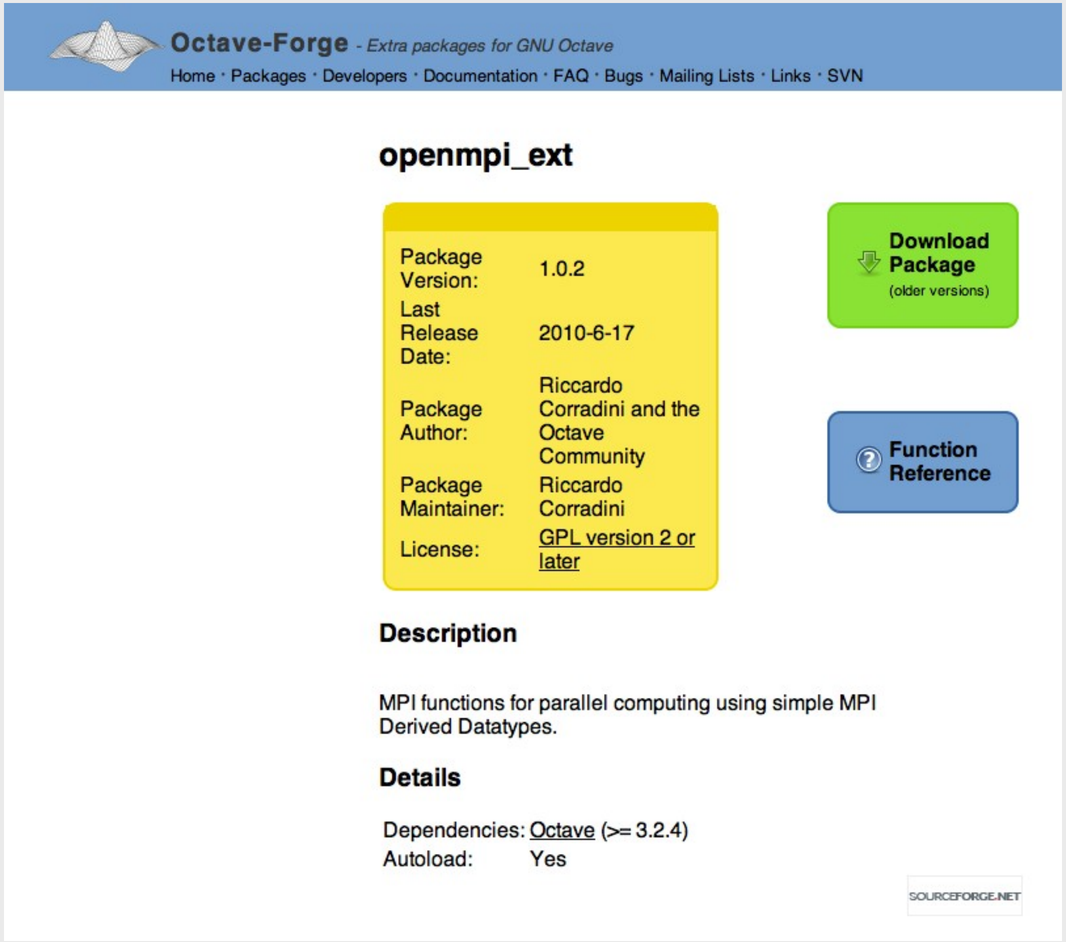
October 21st, 2009 | Categories: Making Mathematica Faster, matlab, programming | Tags:

Slowly but surely more and more MATLAB functions are becoming able to take advantage of multi-core processors. For example, in MATLAB 2009b, functions such as **sort**, **bsxfun**, **filter** and **erf** (among others) gained the ability to spread the calculational load across several processor cores. This is good news because if your code uses these functions, and if you have a multi-core processor, then you will get faster execution times without having to modify your program. This kind of parallelism is called implicit parallelism because it doesn't require any special commands in order to take advantage of multiple cores – MATLAB just does it automagically. Faster code for free!

http://www.walkingrandomly.com/?p=1795

A lot of effort: explicitly parallel

Octave and MPI



Octave-Forge - Extra packages for GNU Octave
Home · Packages · Developers · Documentation · FAQ · Bugs · Mailing Lists · Links · SVN

openmpi_ext

Package Version:	1.0.2
Last Release Date:	2010-6-17
Package Author:	Riccardo Corradini and the Octave Community
Package Maintainer:	Riccardo Corradini
License:	GPL version 2 or later

Download Package
(older versions)
Function Reference

Description

MPI functions for parallel computing using simple MPI Derived Datatypes.

Details

Dependencies:	Octave (>= 3.2.4)
Autoload:	Yes

Batch Matlab/Octave

- Scripts need adjustment
- Batch processing with Matlab
  - Use 'screen'
- Batch processing without Matlab
  - Compile with mcc
  - Dev. with Matlab, Prod. with Octave

Parallel Matlab/Octave

- No effort: Matlab Multithreading
- Some effort: embarrassingly parallel
  - Matlab: Jpar, multicore
  - Octave: parcelleval, multicore
- More (too much?) effort
  - Matlab: SPMD, MPI toolboxes
  - Octave: parallel, openmpi_ext