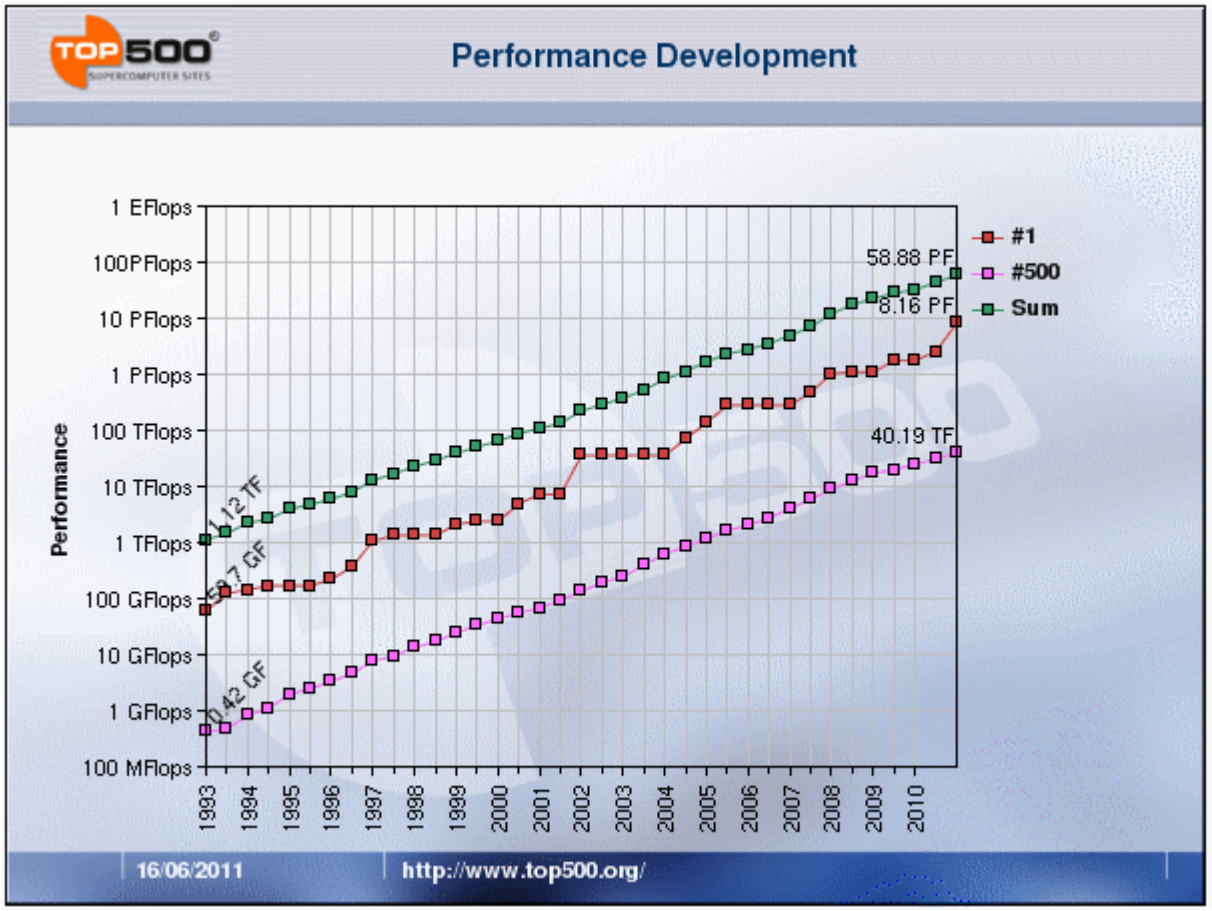
Taxes, Death, and Parallelism are Inevitable

Introduction to Parallel Computing

The FUTURE: ***The race is already on for Exascale Computing!***



Computer performance

Name	FLOPS
yottaFLOPS	10^{24}
zettaFLOPS	10^{21}
exaFLOPS	10^{18}
petaFLOPS	10^{15}
teraFLOPS	10^{12}
gigaFLOPS	10^{9}
megaFLOPS	10^{6}
kiloFLOPS	10^{3}

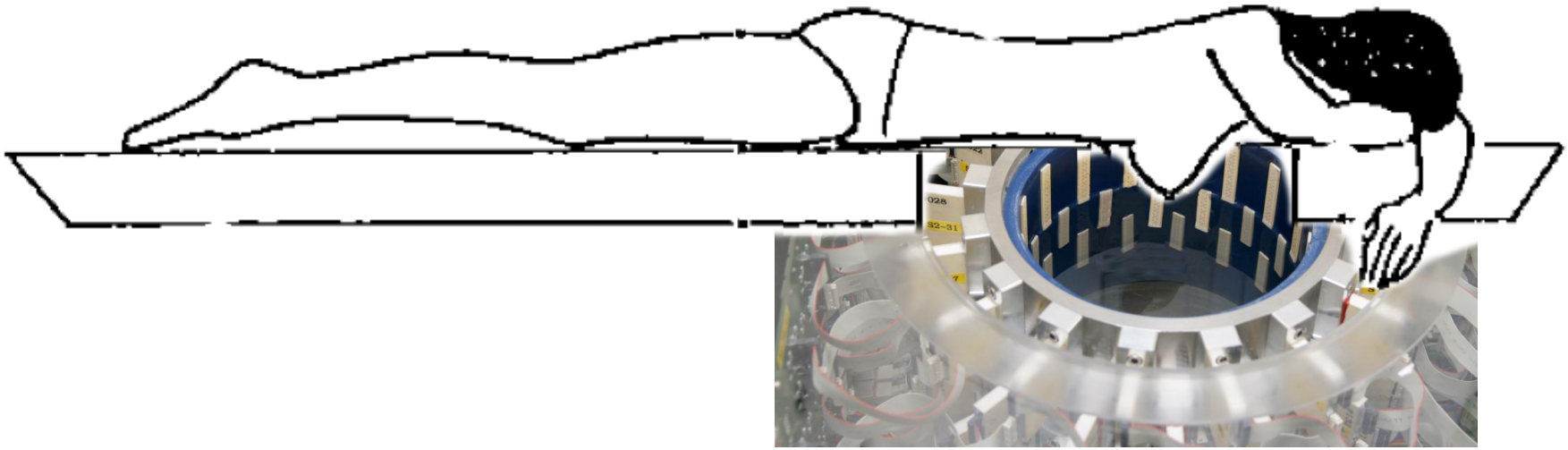
Ultrasound Computer Tomography (USCT)

http://www.interactive-grid.eu

- New method for medical imaging
  - Focus: Breast cancer diagnosis

Marcus.Hardt@iwr.fzk.de

USCT setup



Marcus.Hardt@iwr.fzk.de

USCT Algorithm

- Characteristicts:
  - Input: 20 GB (full set)
  - Computing time depends
    - on output size / resolution
    - amount of input data

35MB	20GB	20GB	Data
4096^2	128^2x100	4096^2x3410	Voxels
1 Hour	1.5 Months	150 Years	Time

- Matlab
  - Strategic development platform (95% sourcecode)

Marcus.Hardt@iwr.fzk.de

USCT Algorithm

- Characteristicts:
  - Input: 20 GB (full set)
  - Computing time depends
    - on output size / resolution
    - amount of input data

35MB	20GB	20GB	Data
4096²	128²x100	4096²x3410	Voxels
1 Hour	1.5 Months	150 Years	Time

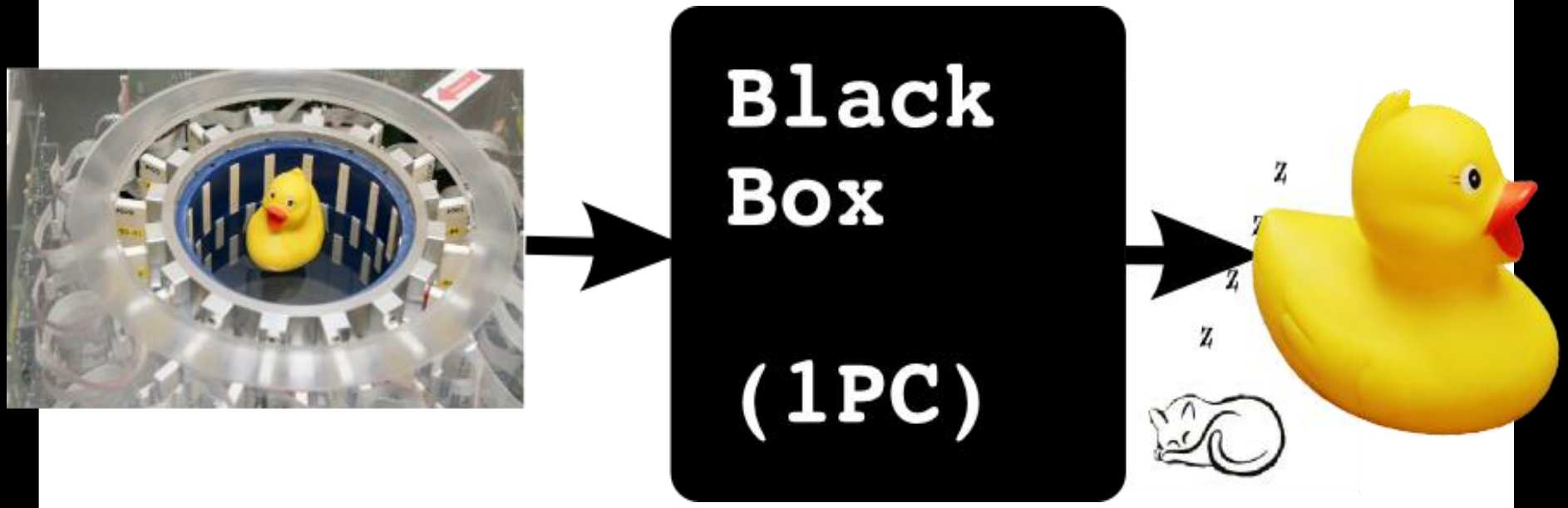
- Matlab

Goals for grid access:
- **Seamless**
- **Interactive**
- **from Matlab**

platform (95%

Marcus.Hardt@iwr.fzk.de

USCT reconstruction := “Black Box”



- Computation takes long (days, weeks, years)
- Grid in order to speed up

Grid Computing

Idea: **Computer power <=> Electrical power**

From Electrical power grid => computational grid

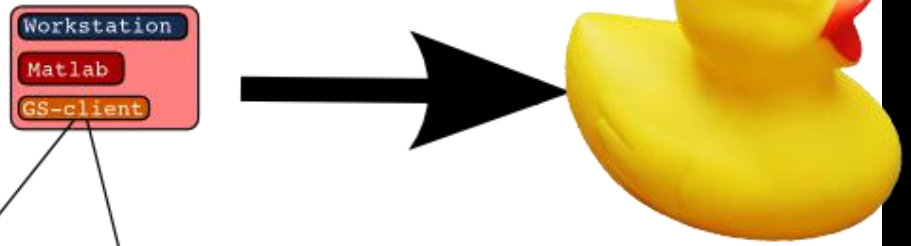
- Across organisationsal domains / countries
- Transparent access to
  - Computing
  - Data
  - Network
- Large scale installations

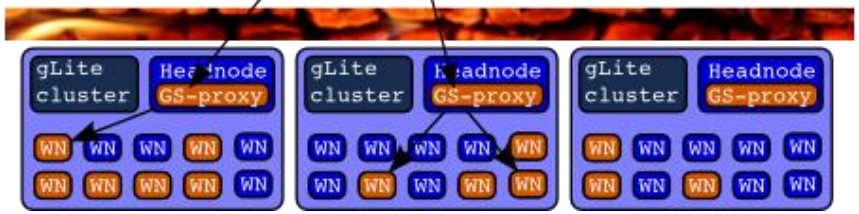
Grid middleware

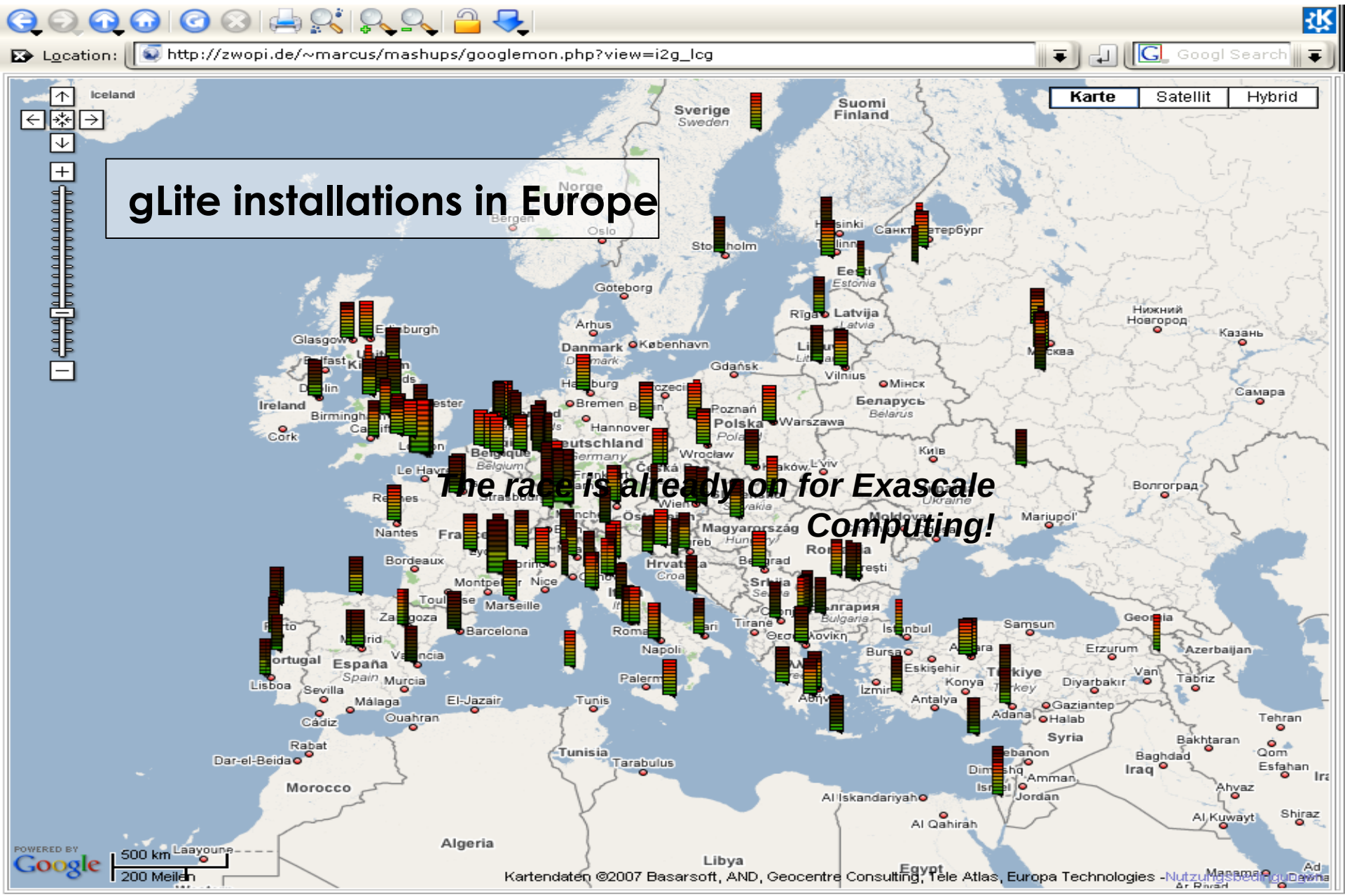
- Middleware
  := Layer between application and operating system

- **gLite:** one grid middleware
  - Development driven by CERN
  - Tools for data+computing of new accelerator
  - 10 TB/year * 20 years, random access
- Paradigm: **Send job to where the data is**
- Job: Self contained application

Putting things together



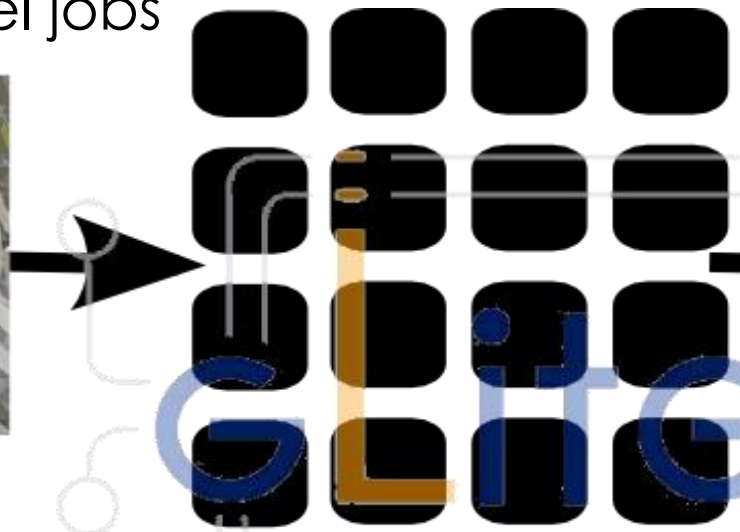


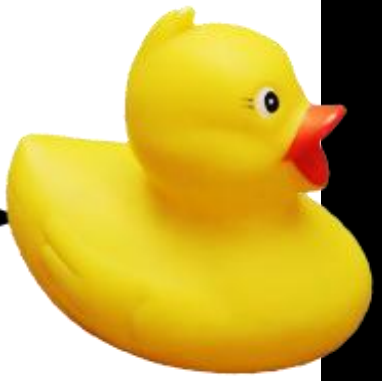


- Sites: 243 (in 49 countries)
- CPUS: 42798 (176 per site)
- RAM: 19TB
- RAM/CPU: 468MB
- DISK [Tot / Avail]: [8042TB / 5408TB] ([33892GB / 22792GB] per site)

Using gLite

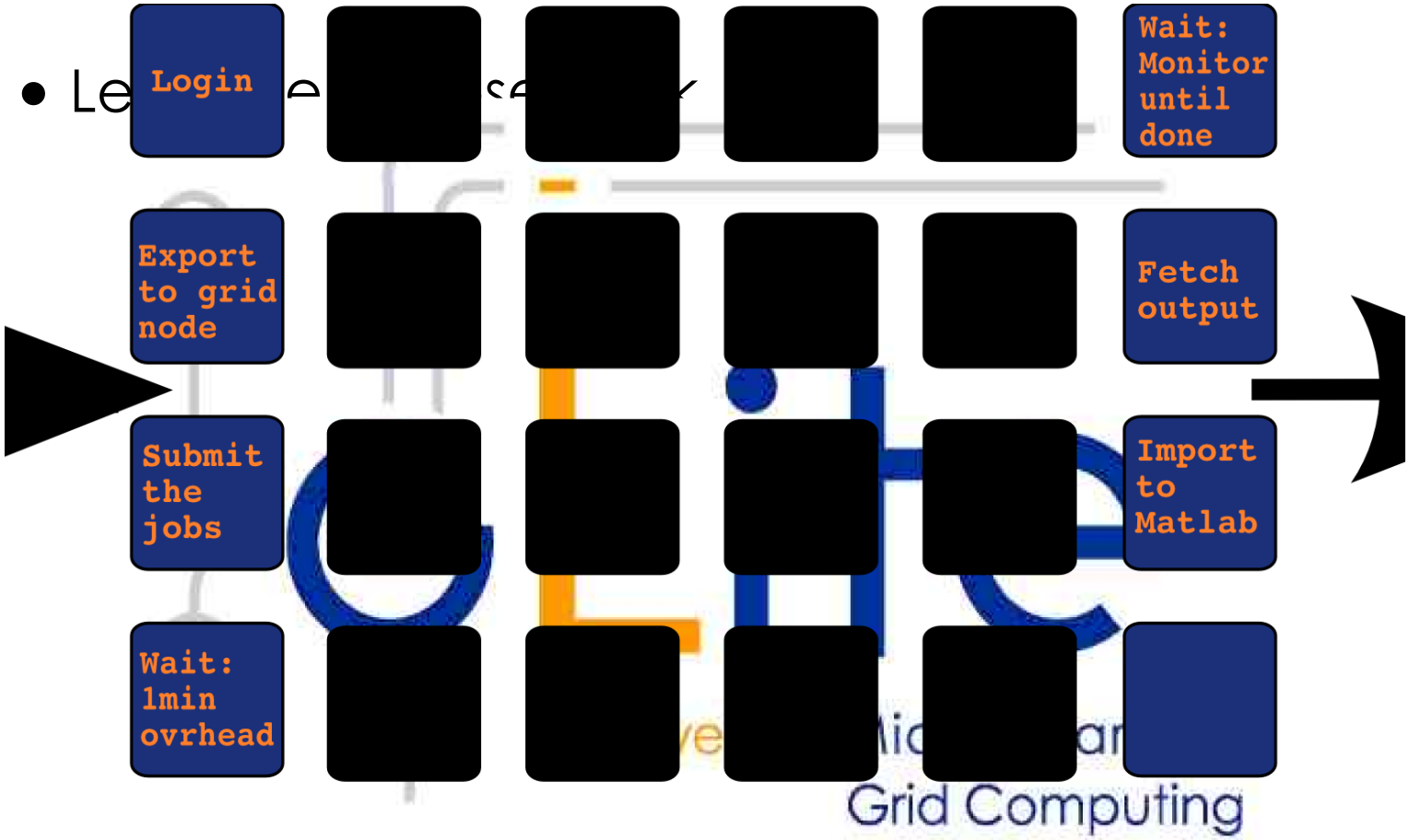
- Initial approach to parallel execution:
  - Partitioning of data
  - Many parallel jobs

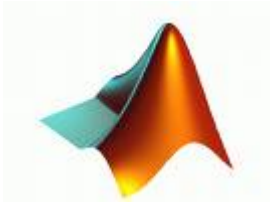




Using gLite

- Le





Parallel Matlab

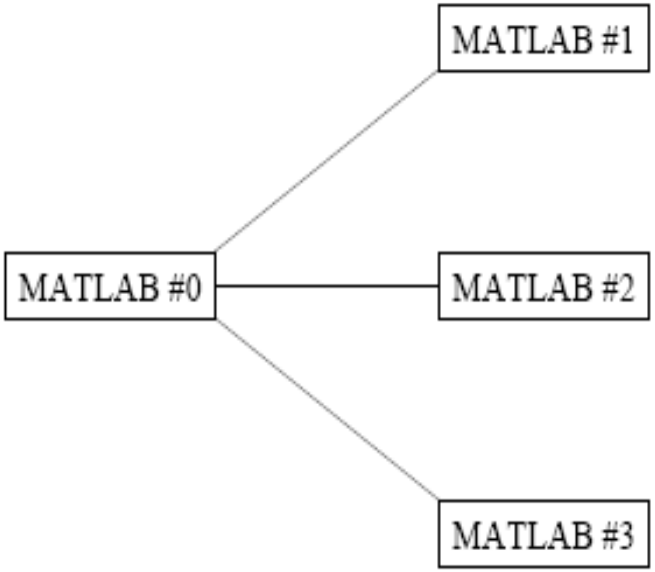
Elias Houstis

About MATLAB

- MATLAB (with Simulink) – programming language for science and engineering
- Over 1 milion users, over 3500 universities and colleges
- Engineering in industry + biotech, medical, financial
- Toolboxes for different fields –
  - Engineering, Bioinformatics, Economics etc.
- Parallel computing support
  - Job execution on multicore/cluster systems
  - MPI support

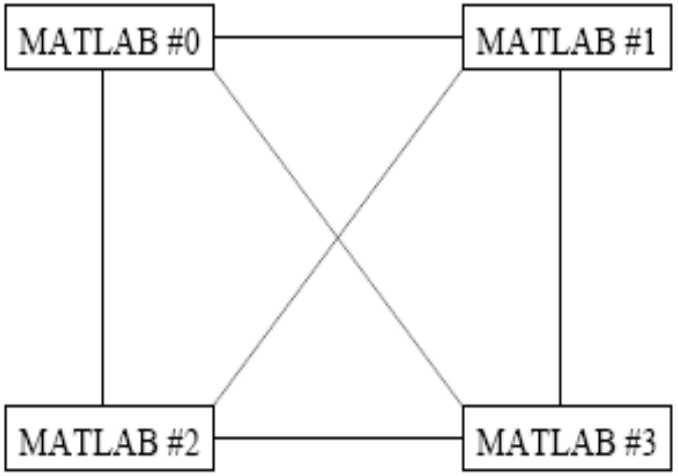
Matlab in Parallel

Beautifully parallel



e.g., Multi, paralize, Plab, ParMatlab

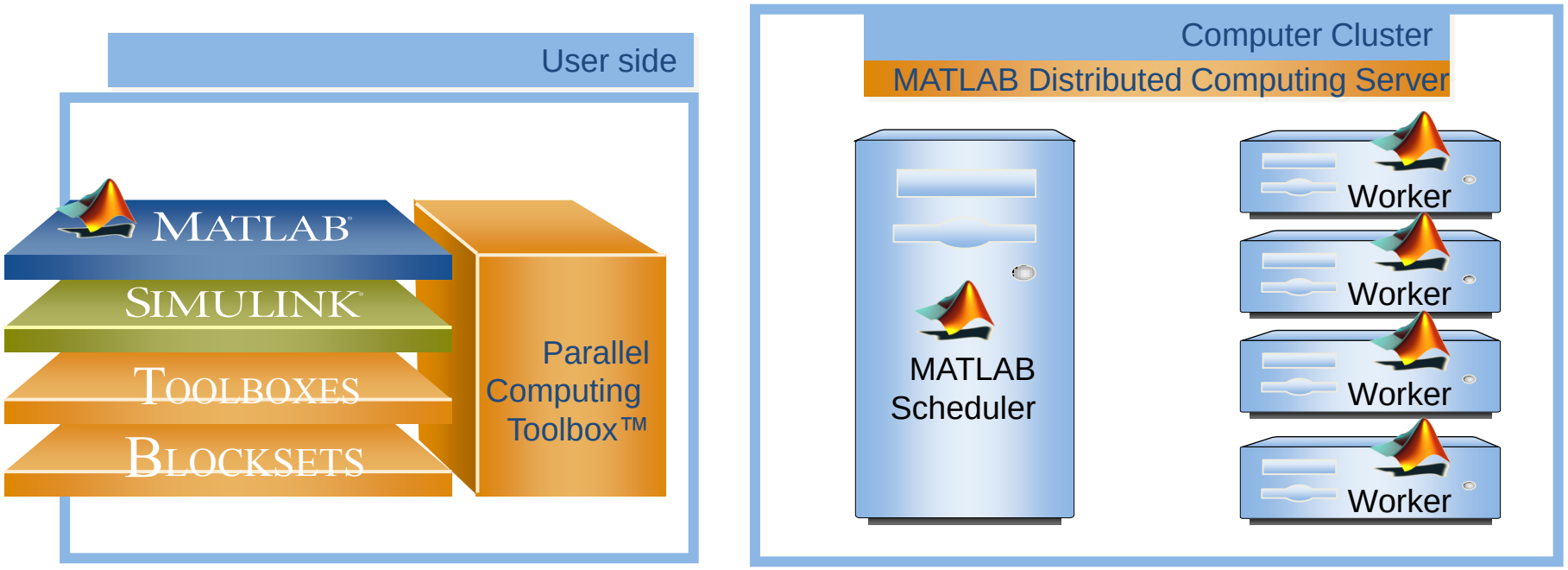
Message Passing



e.g., MultiMatlab, CMTM, DPToolbox,MatlabMPI, pMatlab

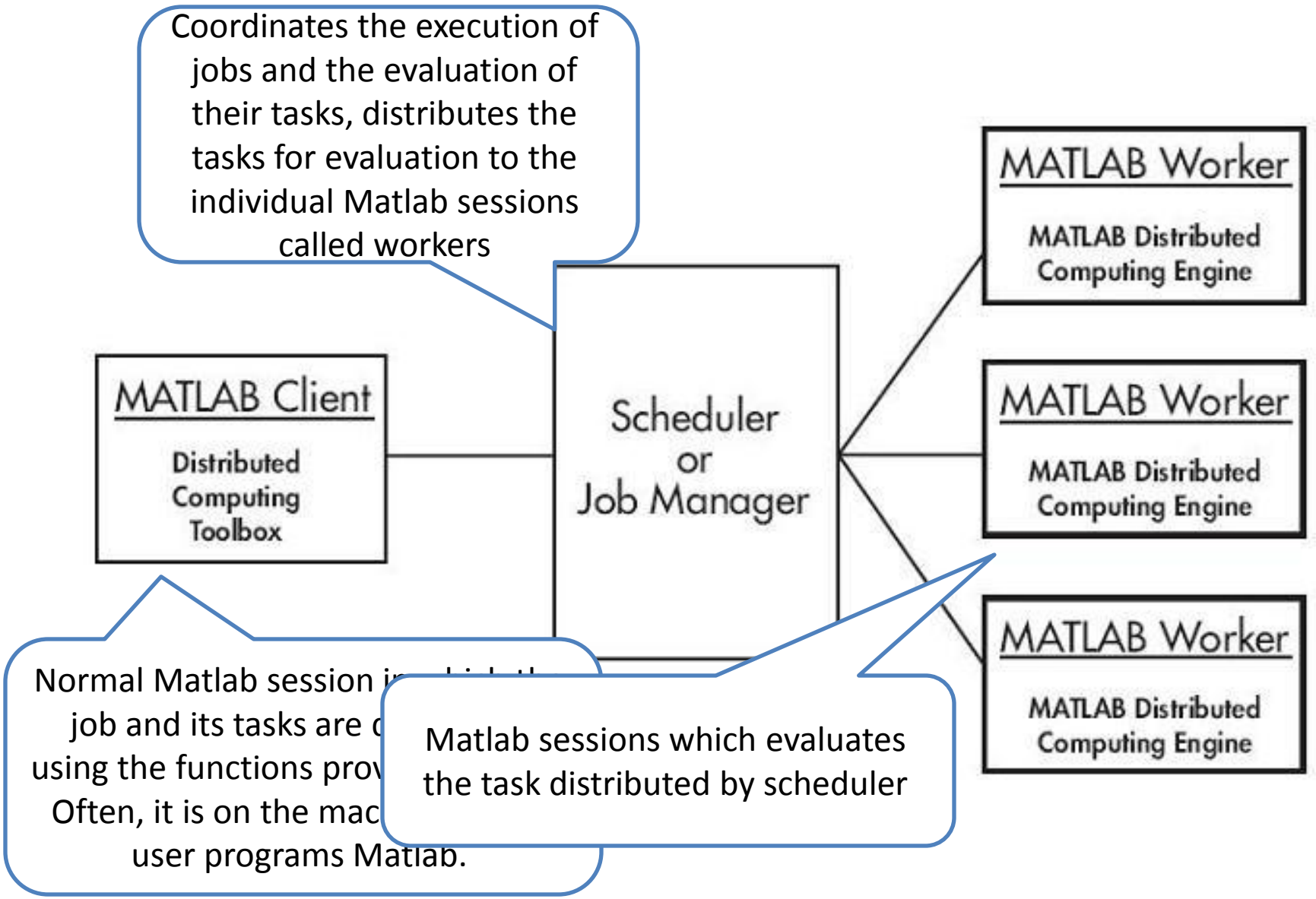
www-math.mit.edu/~edelman/homepage/papers/pmatlab.pdf

Parallel Computing With MATLAB



- Support for third party schedulers

Architecture of PCT

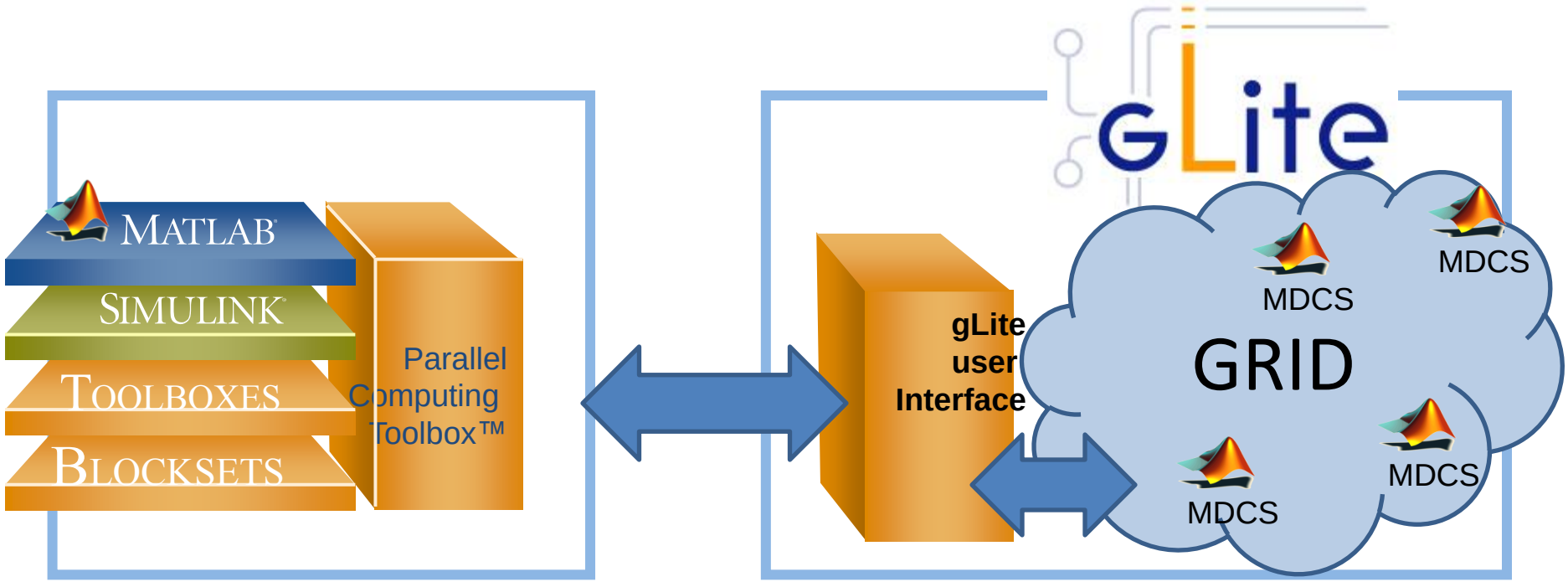


Distributed Computing Server (DCS)

- Parallel Computing Toolbox
  - Only four local workers on a multicore or multiprocessor computer
- PCT + DCS -> Cluster-based applications
- Coordinate and execute independent MATLAB operations simultaneously on a cluster of computers

MATLAB gLite integration

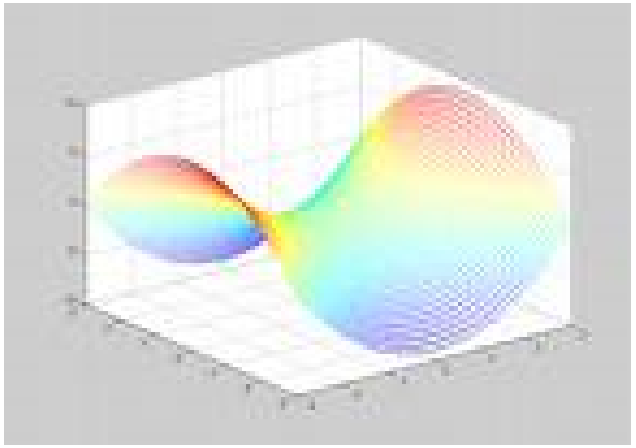
- API for generic scheduler



Developed by EGEE&Mathworks

Parallel Computing Toolbox (PCT)

- Data and task parallelism using
  - Parallel-for loops
  - Distributed arrays
  - Parallel numerical algorithms
  - Message Passing functions



- Easy transition between serial and parallel

http://www.mathworks.com/products/parallel-computing/description1.html

Mathworks – Parallel Computing toolbox

- The toolbox provides eight workers (MATLAB computational engines) to execute applications locally on a multicore desktop
- Parallel for-loops (parfor) for running task-parallel algorithms on multiple processors
- Computer cluster and grid support (with MATLAB Distributed Computing Server)

Vectorization

- >> clear all;

tic;

for i=1:50000

a(i) = sin(i);

end

toc

- Elapsed time is 3.211070 seconds.

>> clear all;

tic;

i = [1:50000];

a = sin(i);

toc

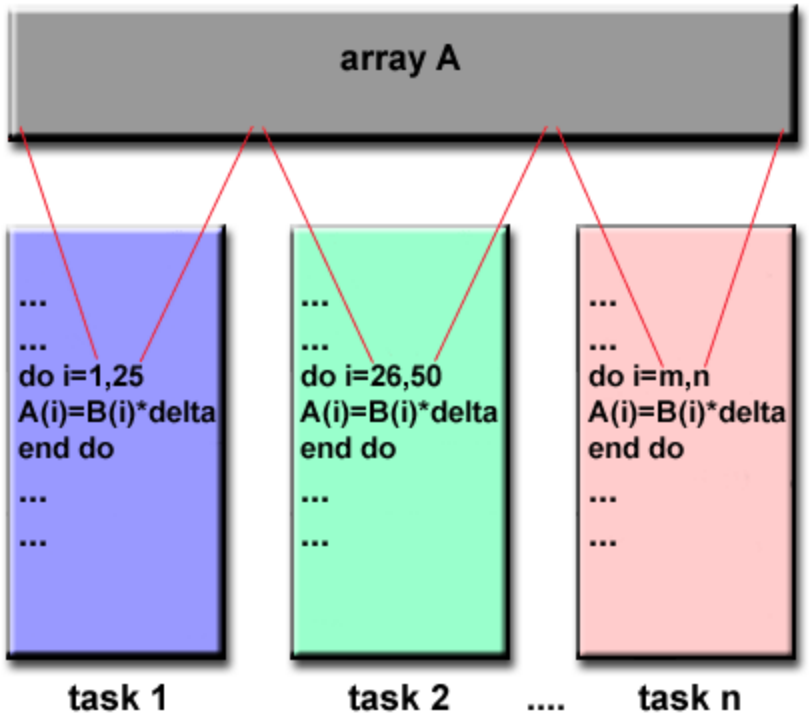
Elapsed time is 0.016062 seconds.

>> **speedup=3.211070 /0.016062**

=199.9172

Data Parallel

- The data parallel model demonstrates the following characteristics:
  - Most of the parallel work performs operations on a data set, organized into a common structure, such as an array
  - A set of tasks works collectively on the same data structure, with each task working on a different partition
  - Tasks perform the same operation on their partition



- On shared memory architectures, all tasks may have access to the data structure through global memory. On distributed memory architectures the data structure is split up and resides as "chunks" in the local memory of each task.

parfor - Parallel for loop

parfor - Parallel for loop

Syntax

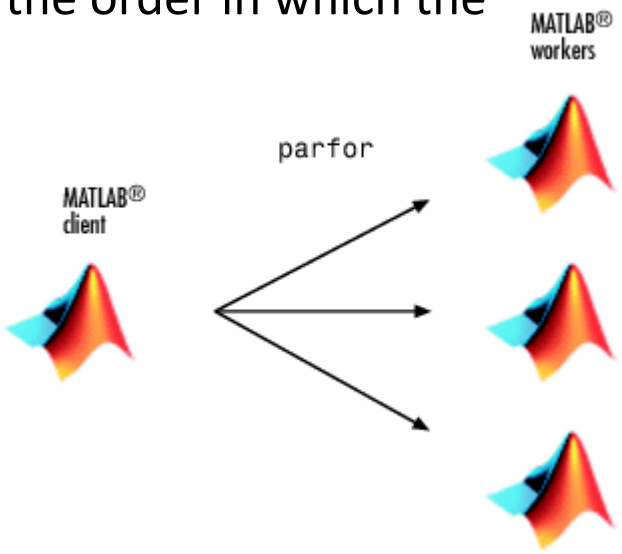
```
parfor loopvar = initval:endval; statements; end
parfor (loopvar = initval:endval, M); statements; end
```

Description

parfor loopvar = initval:endval; statements; end executes a series of MATLAB commands denoted here as statements for values of loopvar between initval and endval, inclusive, which specify a vector of increasing integer values. Unlike a traditional for-loop, there is no guarantee of the order in which the loop iterations are executed.

The general format of a parfor statement is:

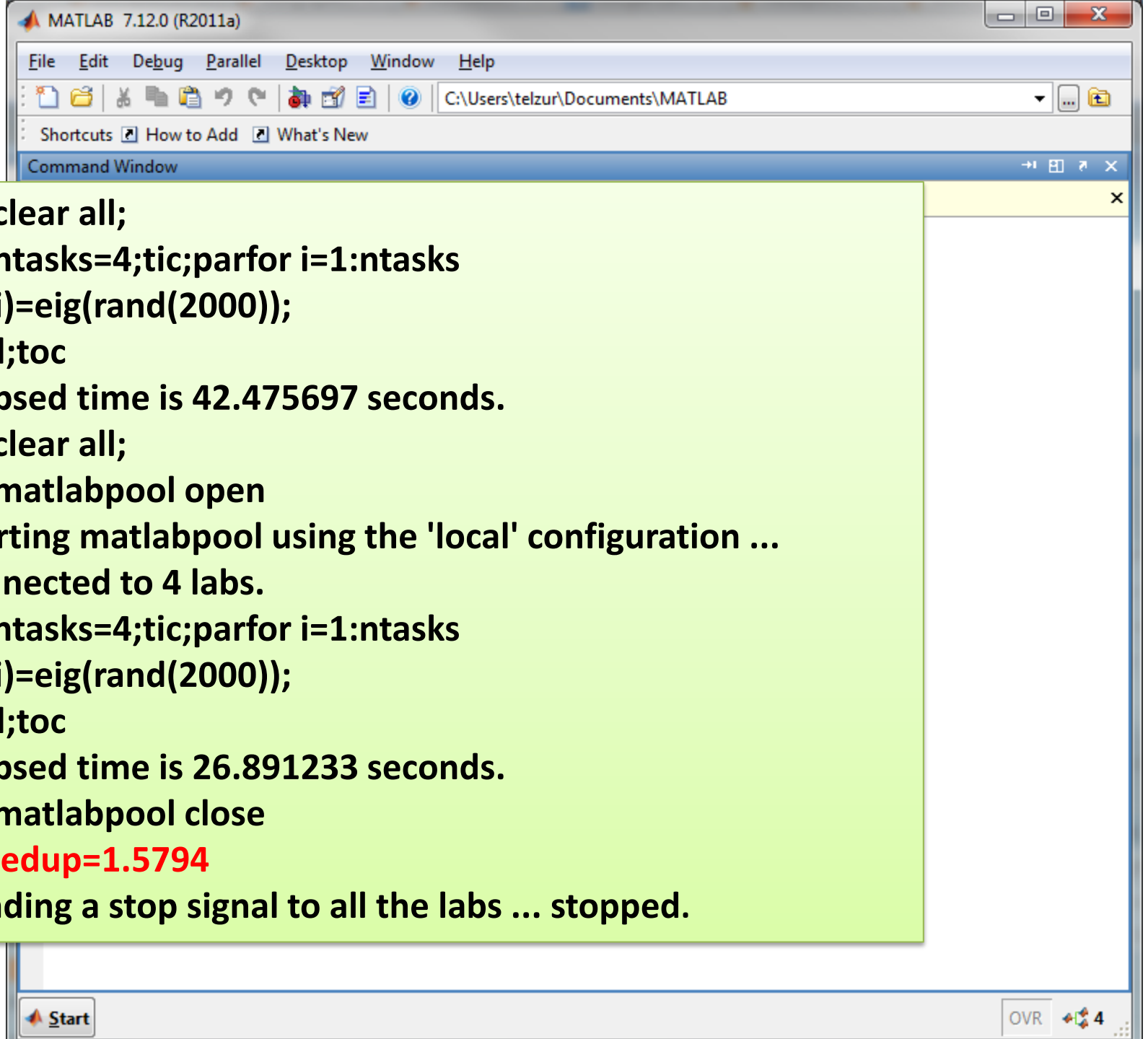
```
parfor loopvar = initval:endval
    <statements>
end
```



parfor – Perform three large eigenvalue computations using four computers or cores

```
>> clear all;
>> ntasks=4;tic;for
i=1:ntasks
c(:,i)=eig(rand(100
0));
end;toc
Elapsed time is
8.575804 seconds.
```

- >> clear all;
- matlabpool open
- ntasks=4;tic;parfor i=1:ntasks
- c(:,i)=eig(rand(1000));
- end;toc
- Starting matlabpool using the 'local' configuration ... connected to 4 labs.
- Elapsed time is 5.198244 seconds.
- Speedup= 1.6522


MATLAB 7.12.0 (R2011a)
File Edit Debug Parallel Desktop Window Help
C:\Users\telzur\Documents\MATLAB
Shortcuts How to Add What's New
Command Window
>> clear all;
>> ntasks=4;tic;parfor i=1:ntasks
c(:,i)=eig(rand(2000));
end;toc
Elapsed time is 42.475697 seconds.
>> clear all;
>> matlabpool open
Starting matlabpool using the 'local' configuration ...
connected to 4 labs.
>> ntasks=4;tic;parfor i=1:ntasks
c(:,i)=eig(rand(2000));
end;toc
Elapsed time is 26.891233 seconds.
>> matlabpool close
speedup=1.5794
Sending a stop signal to all the labs ... stopped.
Start
OVR
4

Parallel mode-I: matlabpool

- Open or close a pool of MATLAB sessions for parallel computation

- Syntax:

  MATLABPOOL
  MATLABPOOL OPEN
  MATLABPOOL OPEN <poolsize>
  MATLABPOOL CLOSE
  MATLABPOOL CLOSE FORCE
  ……

- Work on local client PC

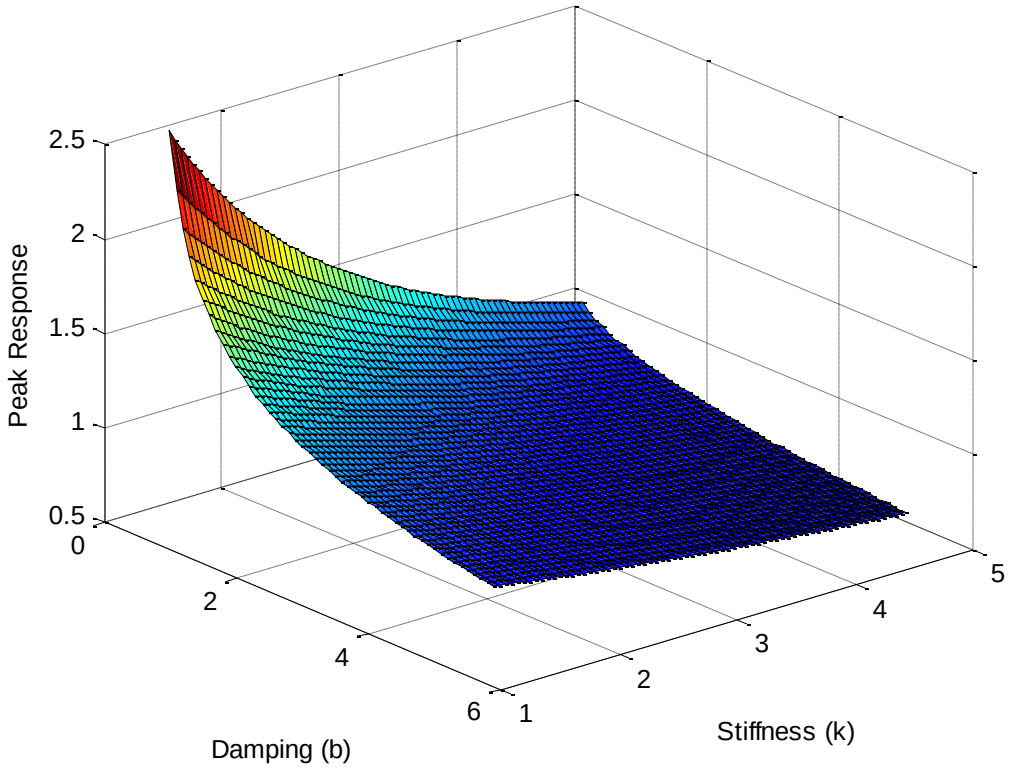
- Without open matlabpool, parallel code will still run but runs sequentially

- %% Parameter Sweep of ODEs
- % This is a parameter sweep study of a 2nd order ODE system.
- %
- % $m\ddot{x} + b\dot{x} + kx = 0$
- %
- % We solve the ODE for a time span of 0 to 25 seconds, with initial
- % conditions $x(0) = 0$ and $\dot{x}(0) = 1$. We sweep the parameters b
- % and k and record the peak values of x for each condition. At the end,
- % we plot a surface of the results.

- Computing in serial...
- Elapsed time is 27.64 seconds.
- Computing in parallel...
- Starting matlabpool using the 'local' configuration ... connected to 4 labs.
- Elapsed time is 12.39 seconds.
- Sending a stop signal to all the labs ... stopped.

- Speed up (time serial / time parallel): 2.23

```
%% Parameter Sweep (Parallel)
% Next, we convert the |for| loop to a |parfor| loop and start a pool or
% MATLAB workers.

disp('Computing in parallel...');drawnow;

matlabpool open

tic;
parfor idx = 1:numel(kGrid)
  % Solve ODE
  [T,Y] = ode45(@(t,y) odesystem(t, y, m, bGrid(idx), kGrid(idx)), ...
    [0, 25], ...  % simulate for 25 seconds
    [0, 1]) ;     % initial conditions

  % Determine peak value
  peakVals(idx) = max(Y(:,1));
end
t2 = toc;
fprintf('Elapsed time is %0.2f seconds.\n', t2);

% Close MATLAB Pool
matlabpool close
```


Peak Response
2.5
2
1.5
1
0.5
0
2
4
6
Damping (b)
1
2
3
4
5
Stiffness (k)

Task Parallel applications

- **parallel problems by organizing them into independent *tasks* (units of work)**

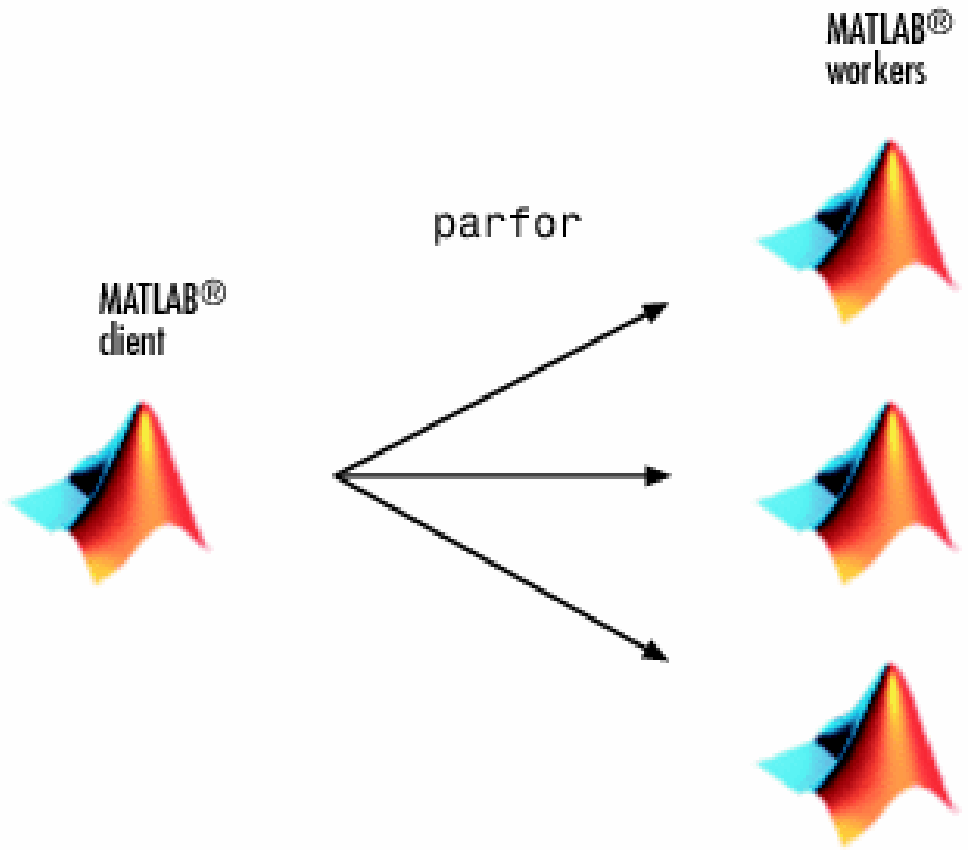
  - parallelize Monte Carlo simulations

- **Parallel for-Loops (parfor)**

```
parfor (i = 1 : n)
        % do something with i
end
```

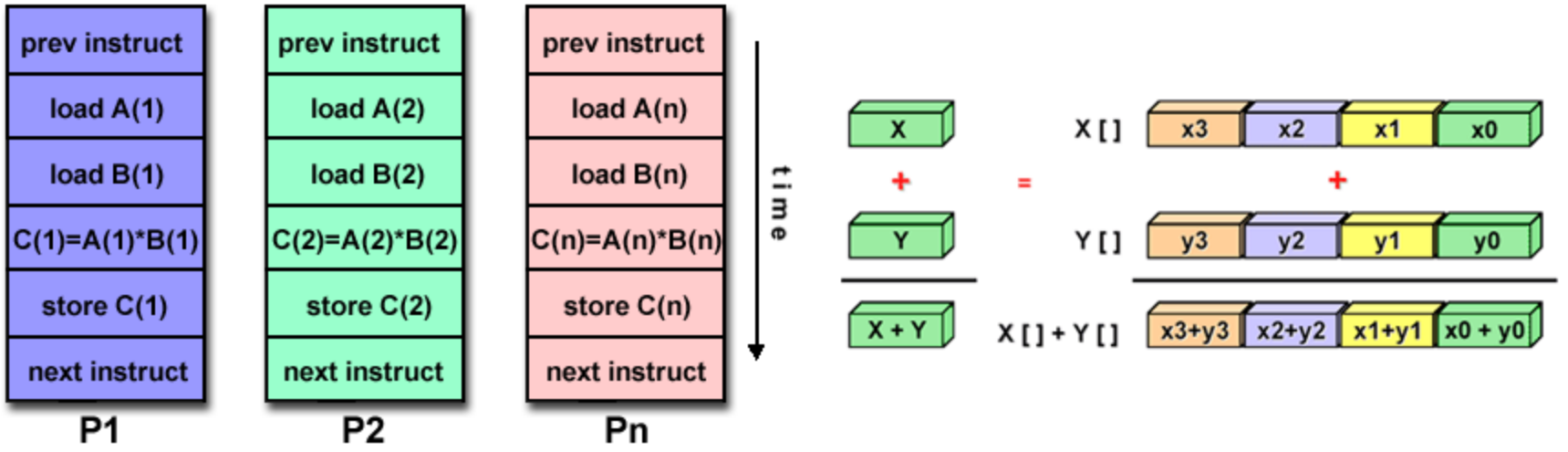
  - Mix task parallel and serial code in the same function
  - Run loops on a pool of MATLAB resources
  - Iterations must be order-independent

Iterations run in parallel in the MATLAB pool (local workers)



SIMD

- A type of parallel computer
- All processing units execute the same instruction at any given clock cycle
- Each processing unit can operate on a different data element
- Two varieties: Processor Arrays and Vector Pipelines
- Most modern computers, particularly those with graphics processor units (GPUs) employ SIMD instructions and execution units.



Data Parallel applications

- Single Program Multiple Data (spmd)

```
spmd (n)
        <statements>
end
```

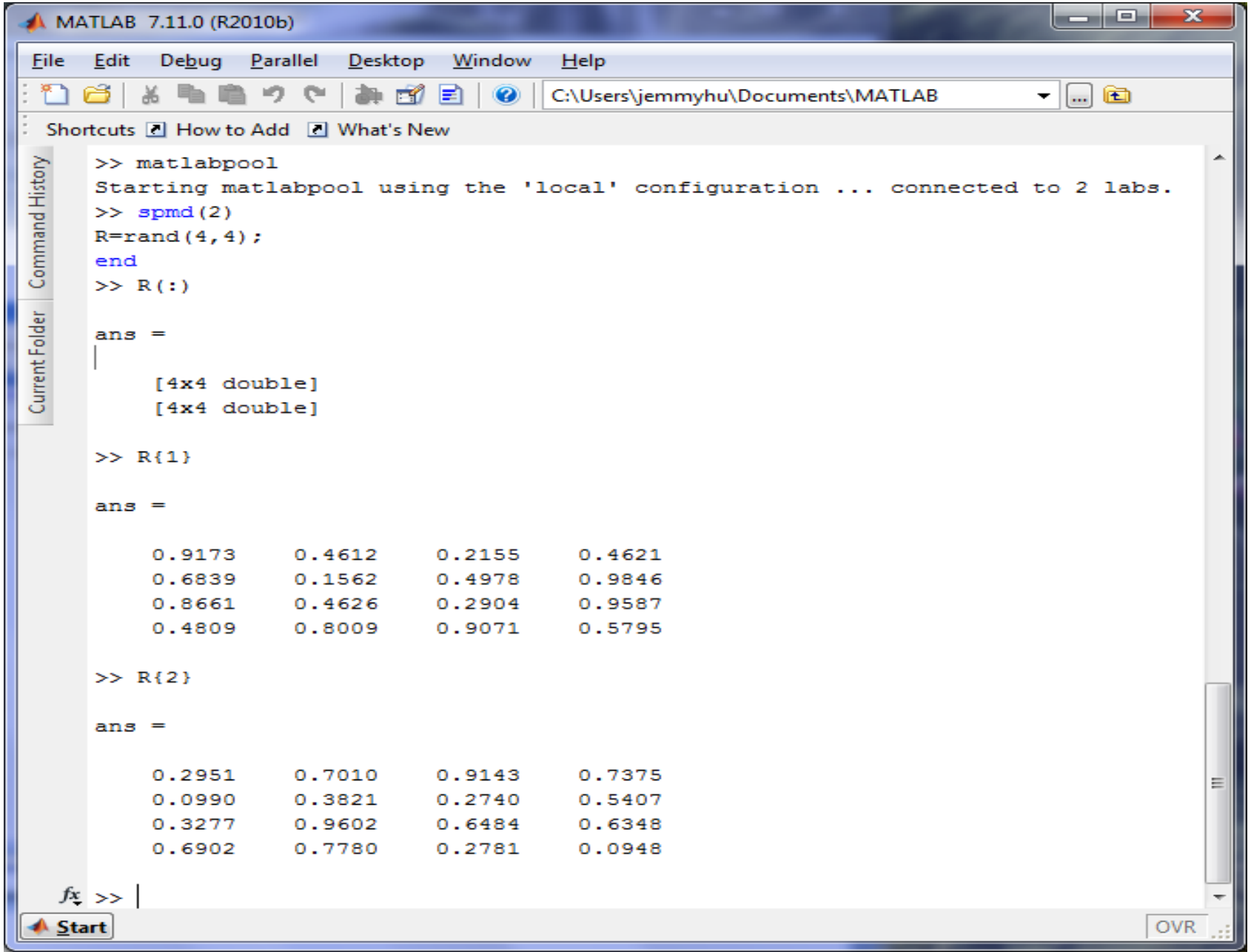
For example, create a random matrix on four labs:

```
matlabpool open
spmd (2)
R = rand(4,4);
end
matlabpool close
```

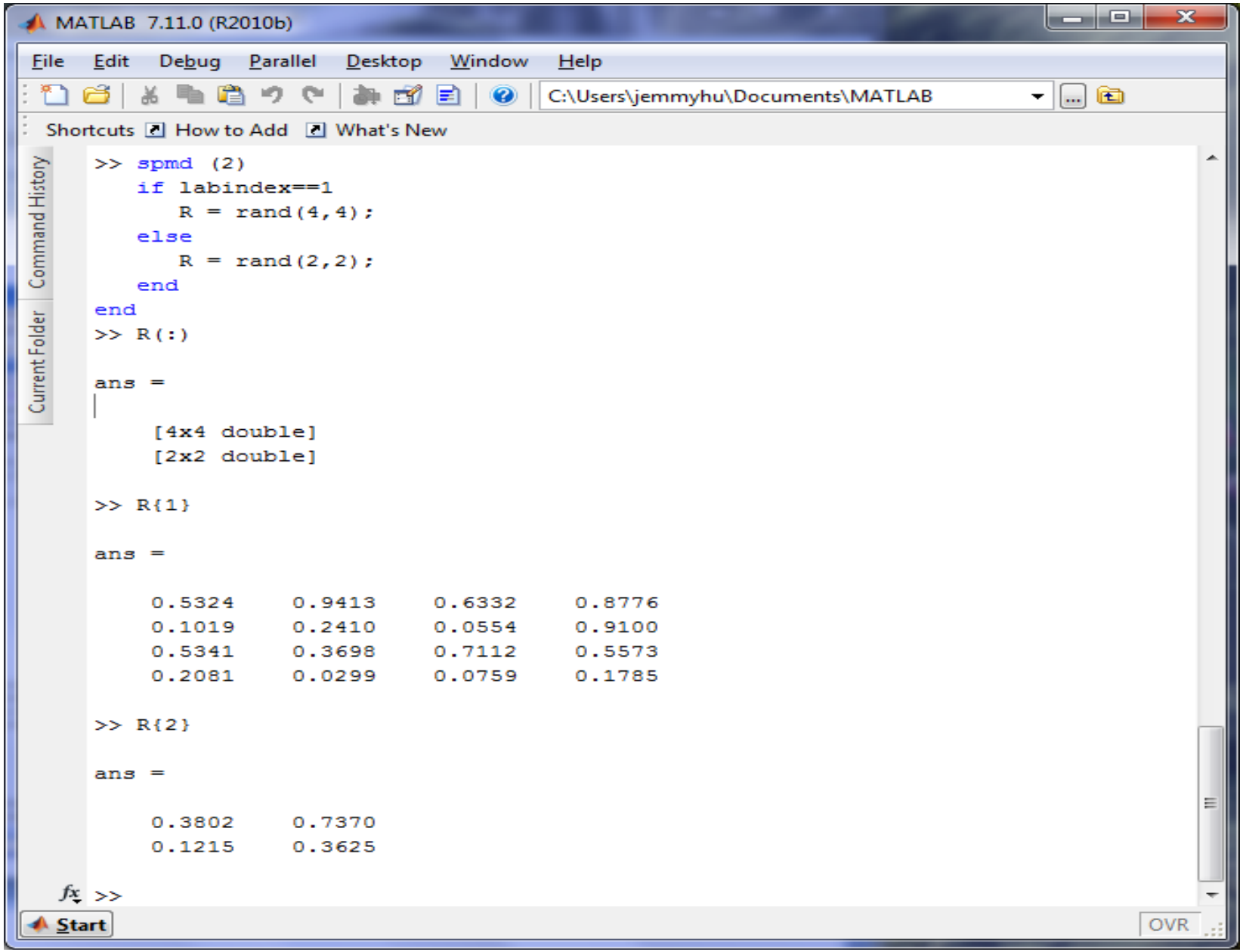
create different sized arrays depending on labindex:

```
matlabpool open
spmd (2)
if labindex==1
R = rand(4,4);
else
R = rand(2,2);
end
end
matlabpool close
```

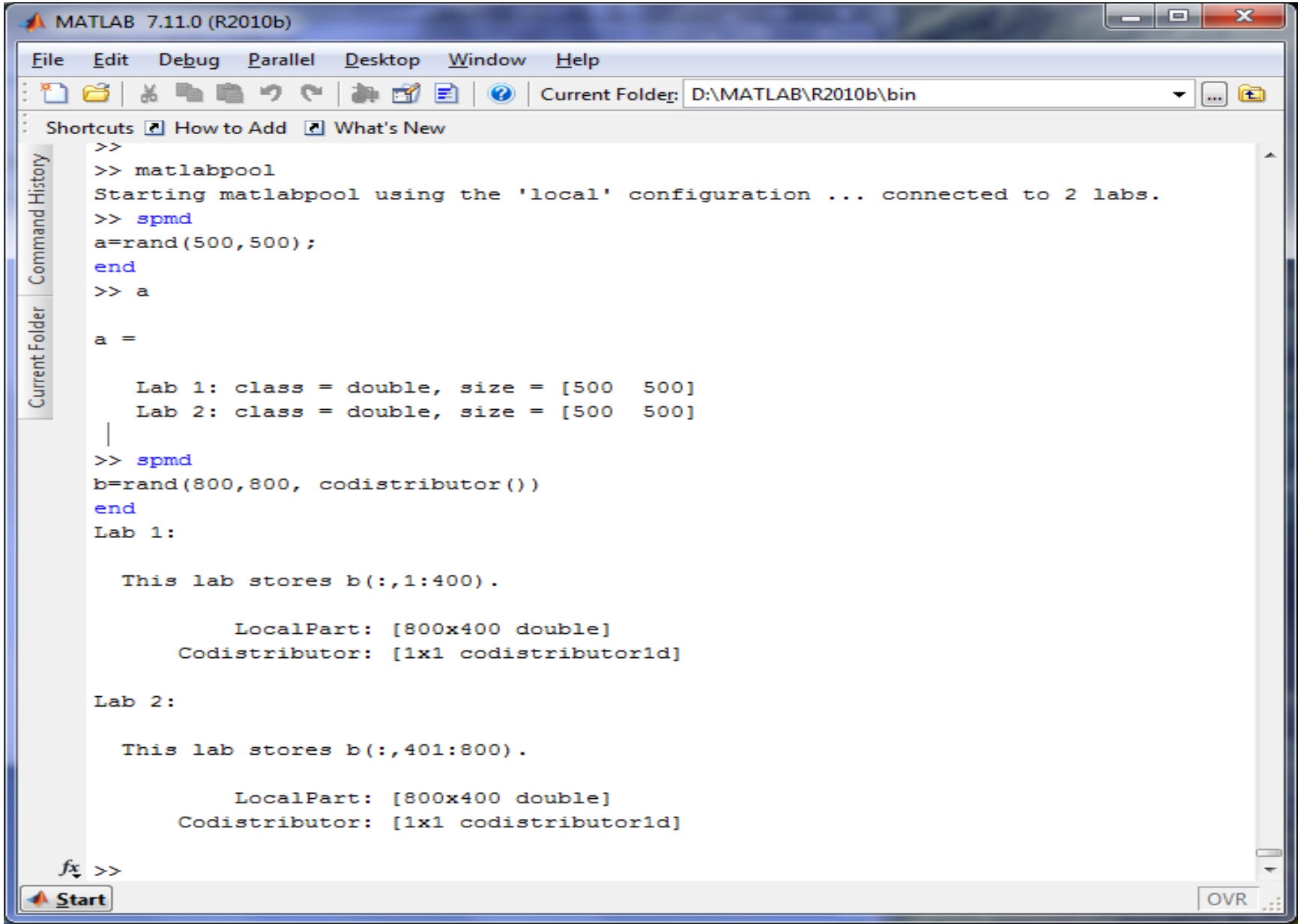
Demo


MATLAB 7.11.0 (R2010b)
File Edit Debug Parallel Desktop Window Help
C:\Users\jemmyhu\Documents\MATLAB
Shortcuts How to Add What's New
Command History
Current Folder
>> matlabpool
Starting matlabpool using the 'local' configuration ... connected to 2 labs.
>> spmd(2)
R=rand(4,4);
end
>> R(:)
ans =
[4x4 double]
[4x4 double]
>> R{1}
ans =
0.9173 0.4612 0.2155 0.4621
0.6839 0.1562 0.4978 0.9846
0.8661 0.4626 0.2904 0.9587
0.4809 0.8009 0.9071 0.5795
>> R{2}
ans =
0.2951 0.7010 0.9143 0.7375
0.0990 0.3821 0.2740 0.5407
0.3277 0.9602 0.6484 0.6348
0.6902 0.7780 0.2781 0.0948
>>
Start
OVR

Demo


MATLAB 7.11.0 (R2010b)
File Edit Debug Parallel Desktop Window Help
C:\Users\jemmyhu\Documents\MATLAB
Shortcuts How to Add What's New
Command History
Current Folder
>> spmd (2)
if labindex==1
R = rand(4,4);
else
R = rand(2,2);
end
end
>> R(:)
ans =
[4x4 double]
[2x2 double]
>> R{1}
ans =
0.5324 0.9413 0.6332 0.8776
0.1019 0.2410 0.0554 0.9100
0.5341 0.3698 0.7112 0.5573
0.2081 0.0299 0.0759 0.1785
>> R{2}
ans =
0.3802 0.7370
0.1215 0.3625
>>
Start
OVR

Distributed arrays and operations (matlabpool mode) codistributor()



MATLAB 7.11.0 (R2010b)

File Edit Debug Parallel Desktop Window Help

Current Folder: D:\MATLAB\R2010b\bin

Shortcuts How to Add What's New

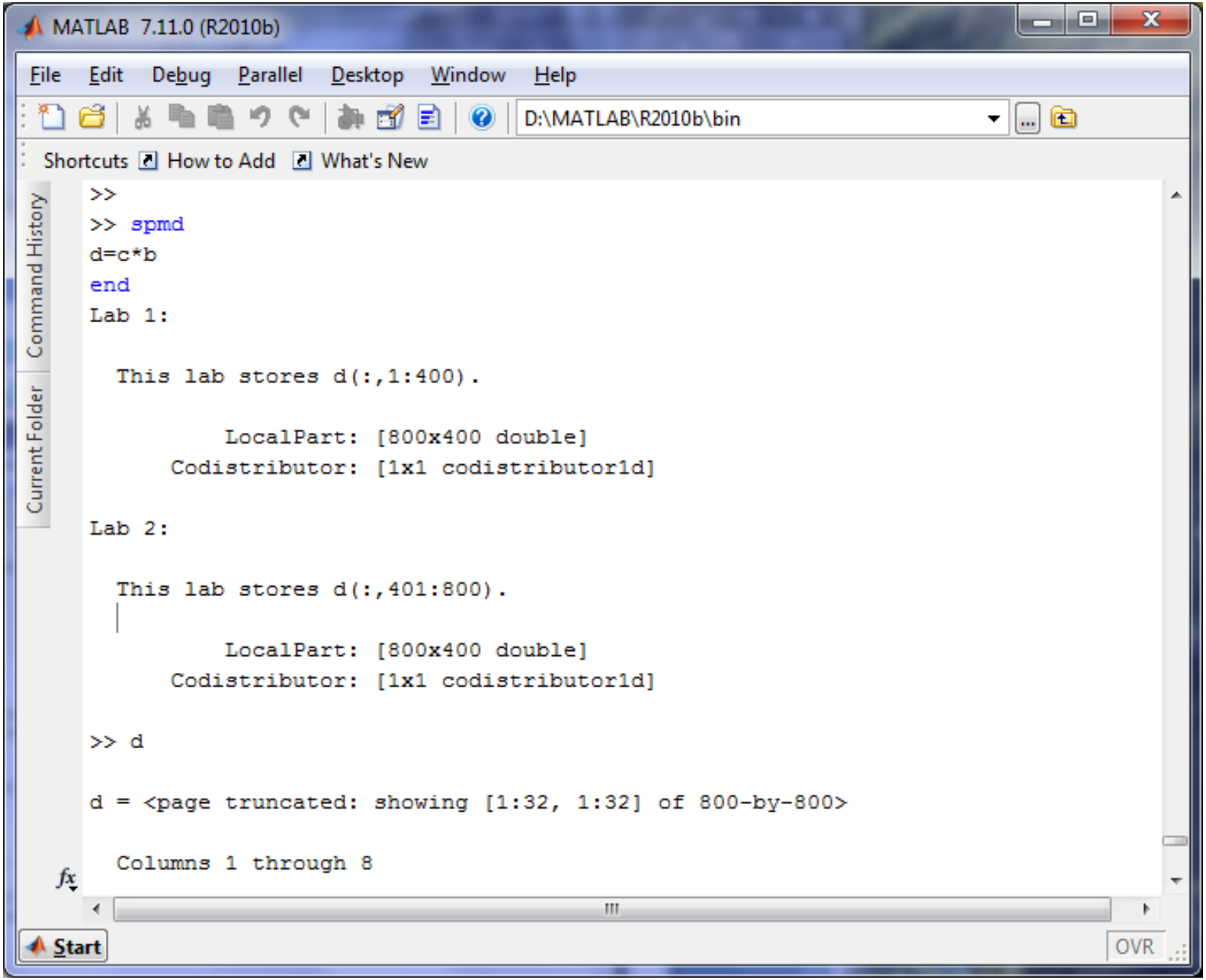
Command History

Current Folder

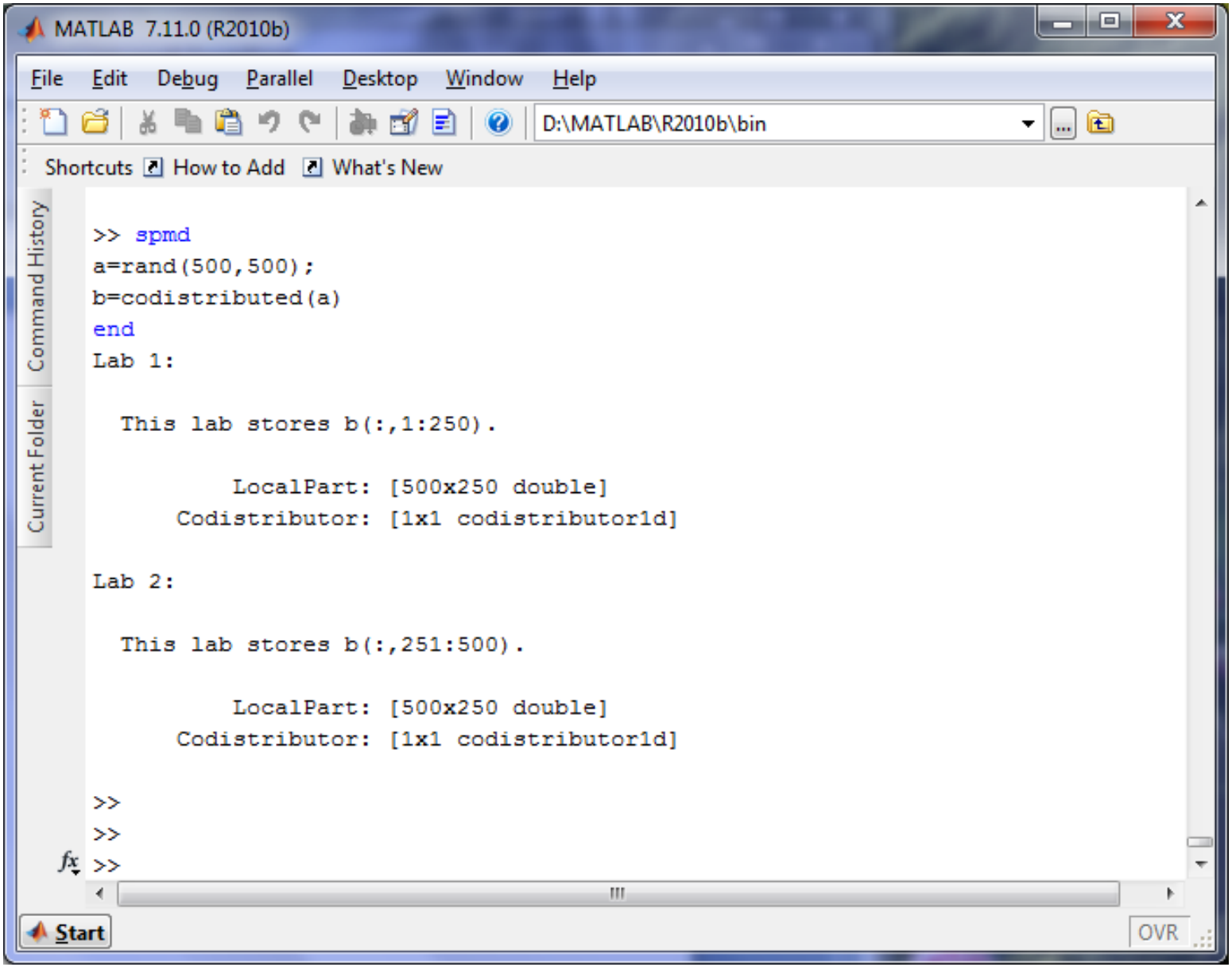
```
>> spmd
c=b'
end
Lab 1:

  This lab stores c(1:400,:).

         LocalPart: [400x800 double]
      Codistributor: [1x1 codistributor1d]

Lab 2:

  This lab stores c(401:800,:).

         LocalPart: [400x800 double]
      Codistributor: [1x1 codistributor1d]

>> c

c = <page truncated: showing [1:32, 1:32] of 800-by-800>

  Columns 1 through 8

    0.2272    0.5522    0.9372    0.8589    0.8029    0.4314    0.2116    0.5545
    0.0658    0.7758    0.0901    0.3114    0.0746    0.5446    0.5271    0.7786
    0.0811    0.2854    0.2272    0.1215    0.3944    0.9134    0.6010    0.7143
```

Start

OVR


MATLAB 7.11.0 (R2010b)
File Edit Debug Parallel Desktop Window Help
D:\MATLAB\R2010b\bin
Shortcuts How to Add What's New
Command History
Current Folder
>>
>> spmd
d=c*b
end
Lab 1:
This lab stores d(:,1:400).
LocalPart: [800x400 double]
Codistributor: [1x1 codistributor1d]
Lab 2:
This lab stores d(:,401:800).
LocalPart: [800x400 double]
Codistributor: [1x1 codistributor1d]
>> d
d = <page truncated: showing [1:32, 1:32] of 800-by-800>
Columns 1 through 8
Start
OVR

codistributed()

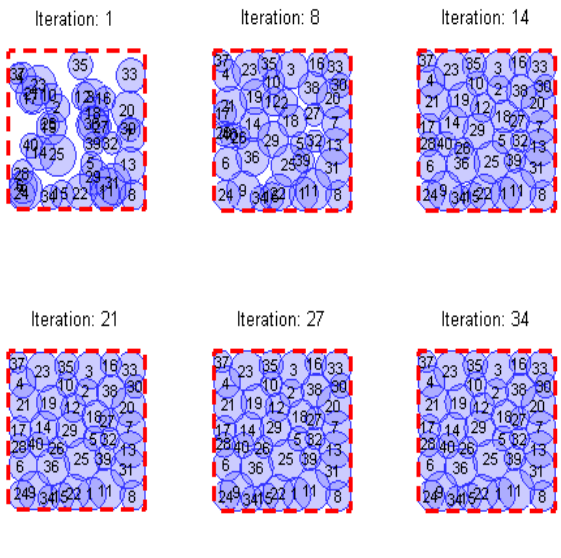


Editor - C:\Users\elias\Downloads\ParallelComputingDemos\ParallelOptim_Example\PhoneTowerScript.m

File Edit Text Go Cell Tools Debug Desktop Window Help

```
%% Demo script for Cell Tower Optimization Problem
% This demo illustrates how you would set up and solve an optimization
% problem (constrained non-linear minimization).
%
% The problem is to determine the locations of # cell towers, each having
% various coverage radius, so that there is maximum coverage, or minimum
% overlap. It is a constrained optimization problem because the cell towers
% are constrained to lie within a boundary. It is a non-linear optimization
% problem because of the non-linear objective function (overlap area as a
% function of # cell tower locations).
```

Highlights of Celltower Optimization



mybatch.m | pmybatch.m | rum_pmybatch.m | Untitled3 | PhoneTowerScript.m | celltowerGUI.m

script Ln 30 Col 42 OVR

MATLAB 7.11.0 (R2010b)

File Edit Debug Parallel Desktop Window Help

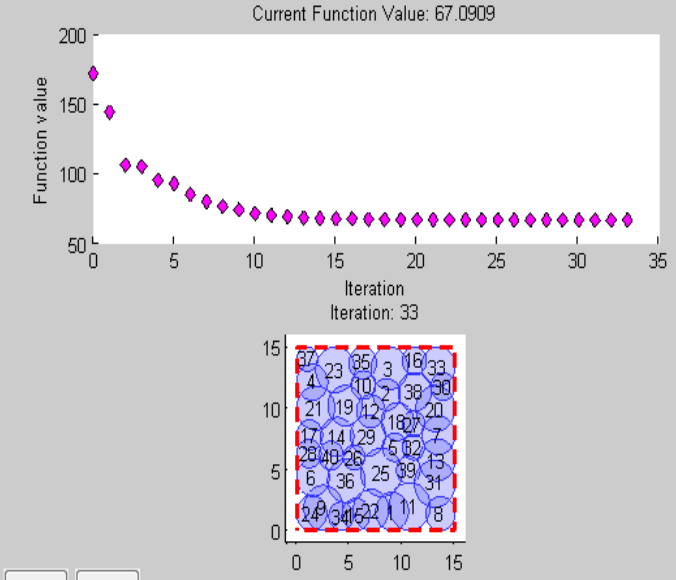
C:\Users\elias

Shortcuts How to Add What's New

```
Elapsed time: 7.21 sec
Elapsed time: 52.99 sec
Elapsed time is 67.55 seconds.
Starting matlabpool using the 'local' configuration ... connected to 4 labs.
Elapsed time is 44.09 seconds.
Speed up (time serial / time parallel): 1.53
Sending a stop signal to all the labs ... stopped.
>>
```

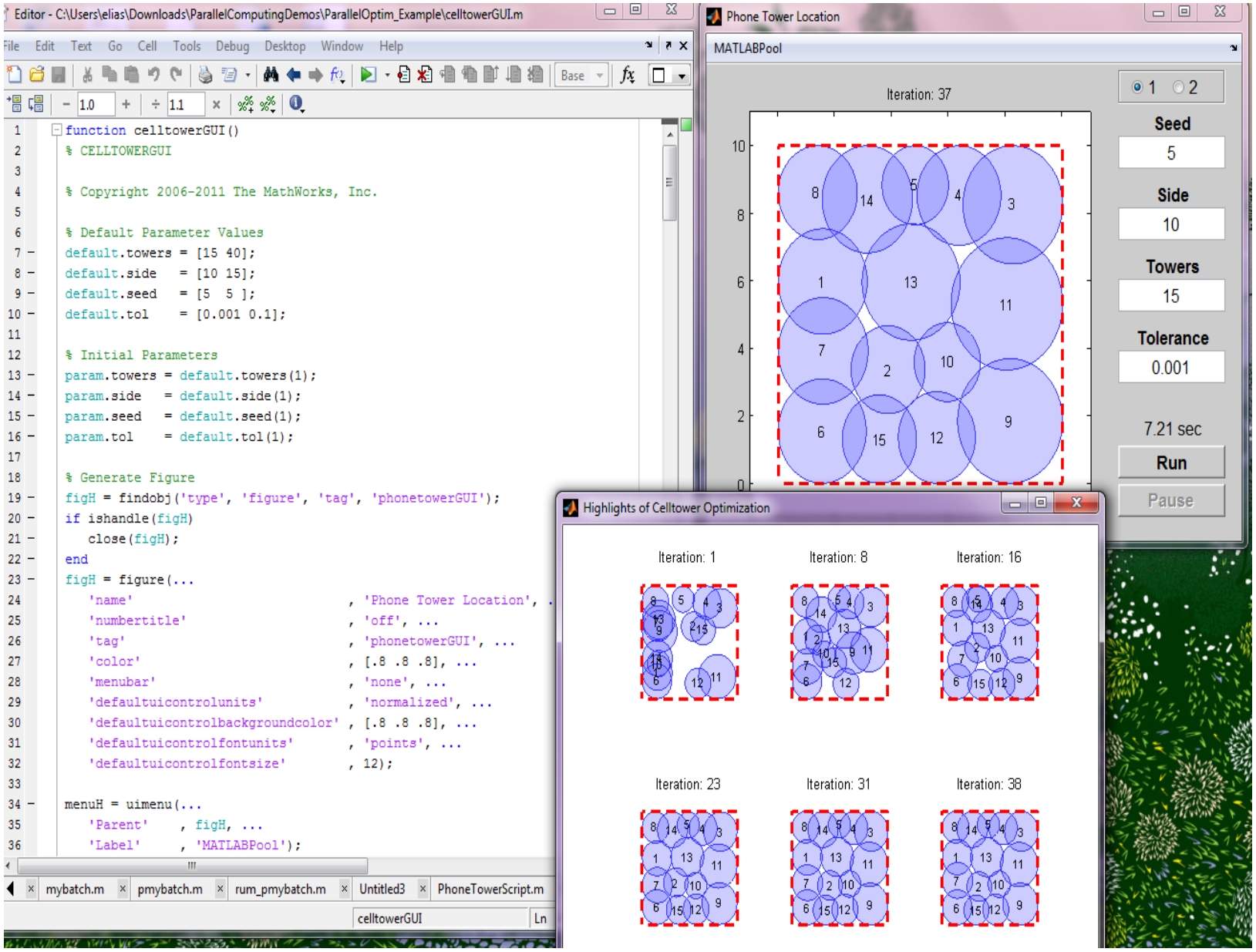
Optimization PlotFcns

File Edit View Insert Tools Desktop Window Help



Stop Pause

Start


Editor - C:\Users\elias\Downloads\ParallelComputingDemos\ParallelOptim_Example\celltowerGUI.m
File Edit Text Go Cell Tools Debug Desktop Window Help
function celltowerGUI()
% CELLTOWERGUI
% Copyright 2006-2011 The MathWorks, Inc.
% Default Parameter Values
default.towers = [15 40];
default.side = [10 15];
default.seed = [5 5];
default.tol = [0.001 0.1];
% Initial Parameters
param.towers = default.towers(1);
param.side = default.side(1);
param.seed = default.seed(1);
param.tol = default.tol(1);
% Generate Figure
figH = findobj('type', 'figure', 'tag', 'phonetowerGUI');
if ishandle(figH)
close(figH);
end
figH = figure(...
'name' , 'Phone Tower Location', ...
'numbertitle' , 'off', ...
'tag' , 'phonetowerGUI', ...
'color' , [.8 .8 .8], ...
'menubar' , 'none', ...
'defaultuicontrolunits' , 'normalized', ...
'defaultuicontrolbackgroundcolor' , [.8 .8 .8], ...
'defaultuicontrolfontunits' , 'points', ...
'defaultuicontrolfontsize' , 12);
menuH = uimenu(...
'Parent' , figH, ...
'Label' , 'MATLABPool');
mybatch.m pmybatch.m rum_pmybatch.m Untitled3 PhoneTowerScript.m
celltowerGUI
Ln
Phone Tower Location
MATLABPool
Iteration: 37
1 2
Seed
5
Side
10
Towers
15
Tolerance
0.001
7.21 sec
Run
Pause
Highlights of Celltower Optimization
Iteration: 1
Iteration: 8
Iteration: 16
Iteration: 23
Iteration: 31
Iteration: 38

- **Parallel mode on a MATLAB Pool (1)**

  matlabpool Open or close pool of MATLAB sessions for parallel computation

  parfor Execute code loop in parallel

  spmd Execute code in parallel on MATLAB pool

- **Interactive Functions**

  help Help for toolbox functions in Command Window

  pmode Interactive Parallel Command Window

- **Parallel mode on a MATLAB Pool (2)**

  batch Run MATLAB script as batch job

```
1  function [time1,time2,time3] = new
2  n=10^8; step = 1/n;
3  % Serial Version
4  tic; s=0;
5  for i=1:n-1
6  x=(i-0.5)*step; s=s+4./(1+x^2);
7  end
8  time1 = toc;
9
10 % Parallel Version
11 matlabpool open 4
12 tic;
13 spmd
14 slocal = myTestSum(n,step);
15 end
16 time2 = toc;
17 % Parallel Version parfor
18 nstep=10^8; step = 1/nstep;
19 tic;s=0;
20 parfor i=1:nstep-1
21 x=(i-0.5)*step;
22 s=s+4./(1+x^2);
23 end
24 time3 = toc;matlabpool close
25 end
26 %%
27 function slocal = myTestSum(n,step
28 nlo = ( n * ( labindex - 1 ) ) / n
29 nhi = ( n * labindex) / numlabs;
30 slocal = 0;
31 for i=nlo:nhi
32 x=(i-0.5)*step; slocal=slocal+4./(
33 end
34 end
```

```
>> [time1,time2,time3] = newSPMDVersion
Starting matlabpool using the 'local' configuration ... connected to 4 labs.
Sending a stop signal to all the labs ... stopped.

time1 =

    2.7594


time2 =

    1.0683


time3 =

    0.9888

>>
```

```
function [time1,time2,time3] =
newSPMDVersion
n=10^8; step = 1/n;
% Serial Version
tic; s=0;
for i=1:n-1
x=(i-0.5)*step; s=s+4./(1+x^2);
end
time1 = toc;

% Parallel Version
matlabpool open 4
tic;
spmd
slocal = myTestSum(n,step);
end
time2 = toc;
```

```
% Parallel Version parfor
nstep=10^8; step = 1/nstep;
tic;s=0;
parfor i=1:nstep-1
x=(i-0.5)*step;
s=s+4./(1+x^2);
end
time3 = toc;matlabpool close
end

function slocal = myTestSum(n,step)
nlo = ( n * ( labindex - 1 ) ) / numlabs + 1;
nhi = ( n * labindex) / numlabs;
slocal = 0;
for i=nlo:nhi
x=(i-0.5)*step; slocal=slocal+4./(1+x^2);
end
end
```

Profiling (profile on…profile viewer)

- profile on
- computations

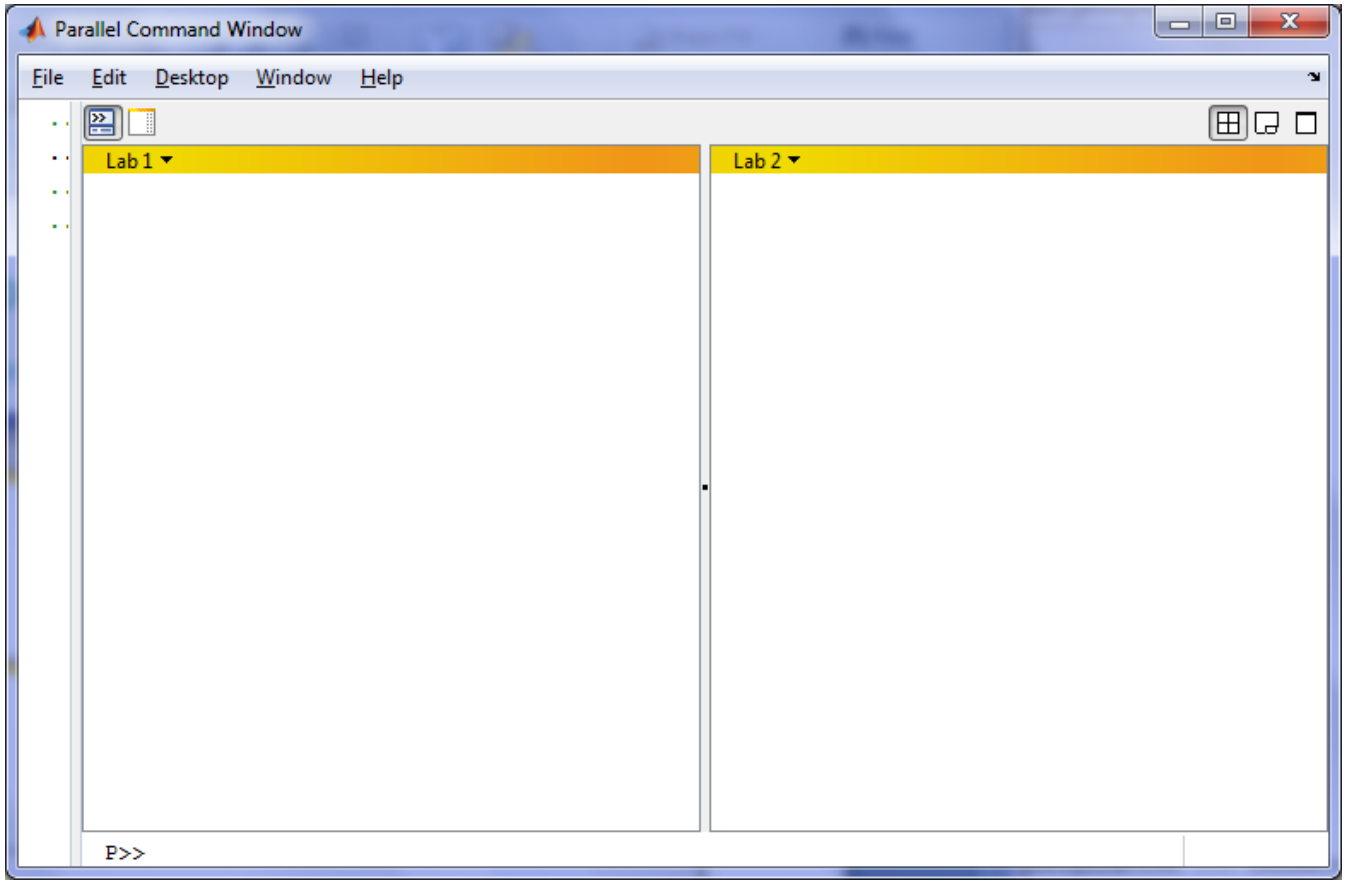
- profile viewer

Function listing

time	calls	line	
		1	function [time1,time2,time3] = newSPMDVersion
	1	2	n=10^8; step = 1/n;
		3	% Serial Version
	1	4	tic; s=0;
< 0.01	1	5	for i=1:n-1
8.24	99999999	6	x=(i-0.5)*step; s=s+4./(1+x^2);
10.49	99999999	7	end
	1	8	time1 = toc;
		9	
		10	% Parallel Version
6.08	1	11	matlabpool open 4
	1	12	tic;
1.07	1	13	spmd
		14	slocal = myTestSum(n,step);
		15	end
	1	16	time2 = toc;
		17	% Parallel Version parfor
	1	18	nstep=10^8; step = 1/nstep;
	1	19	tic;s=0;
1.04	1	20	parfor i=1:nstep-1
		21	x=(i-0.5)*step;
		22	s=s+4./(1+x^2);
		23	end
3.32	1	24	time3 = toc;matlabpool close
	1	25	end

Parallel mode-II: pmode

>> pmode start



P>> pmode exit

pmode demo

P>> help magic % ask for help on a function

P>> PI = pi % set a variable on all the labs

P>> myid = labindex % lab ID

P>> all = numlabs % total No. of labs

P>> segment = [1 2; 3 4; 5 6] % create a replicated array on all the labs

P>> segment = segment + 10*labindex % perform on different labs

P>> x = magic(4) % replicated on every lab

P>> y=codistirbuted(x) % partitioned among the lab

P>> z = y + 10*labindex % operate on the distributed array whole

P>> combined = gather(y) % entire array in one workspace

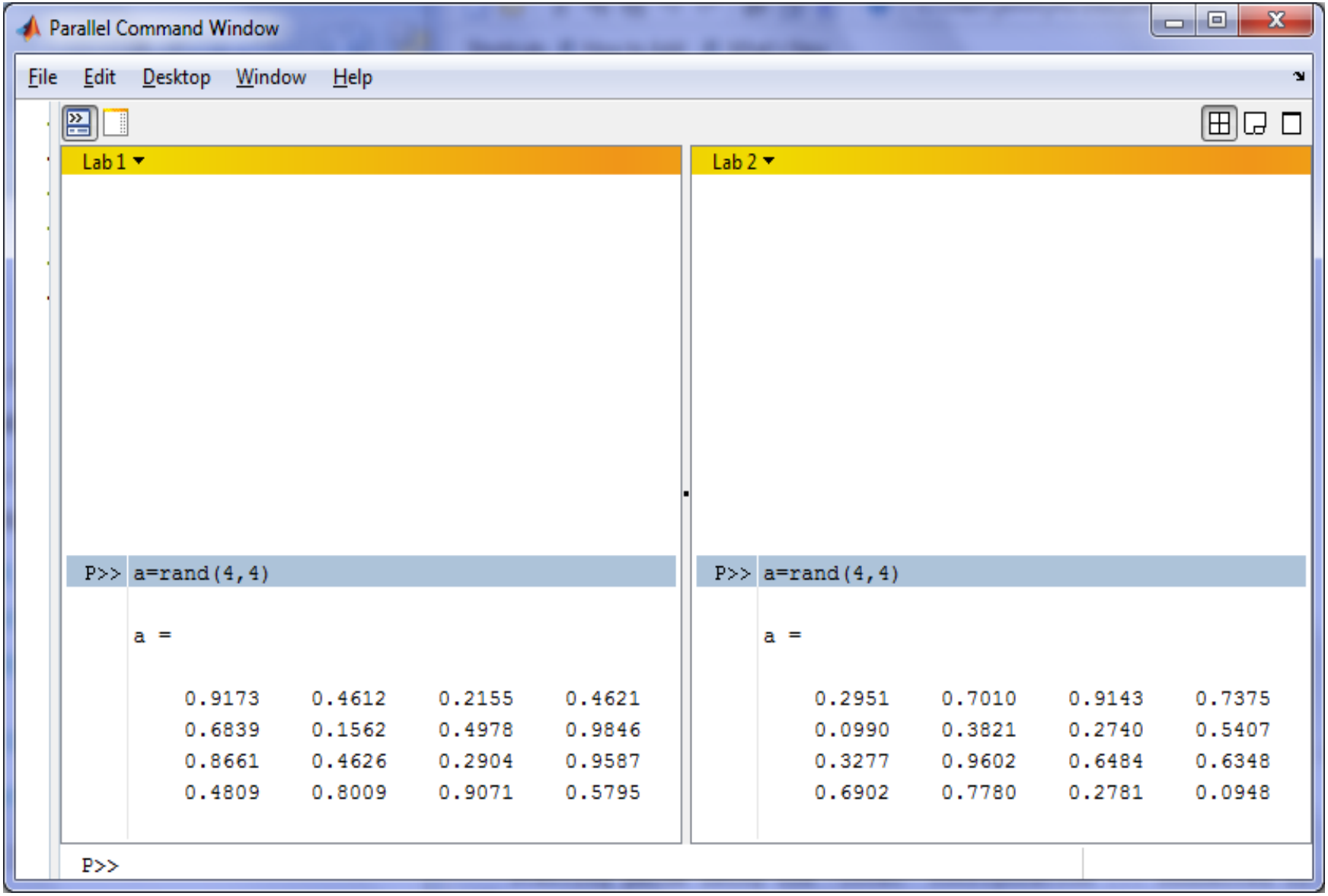
The combined is now a 4-by-4 array in the client workspace.

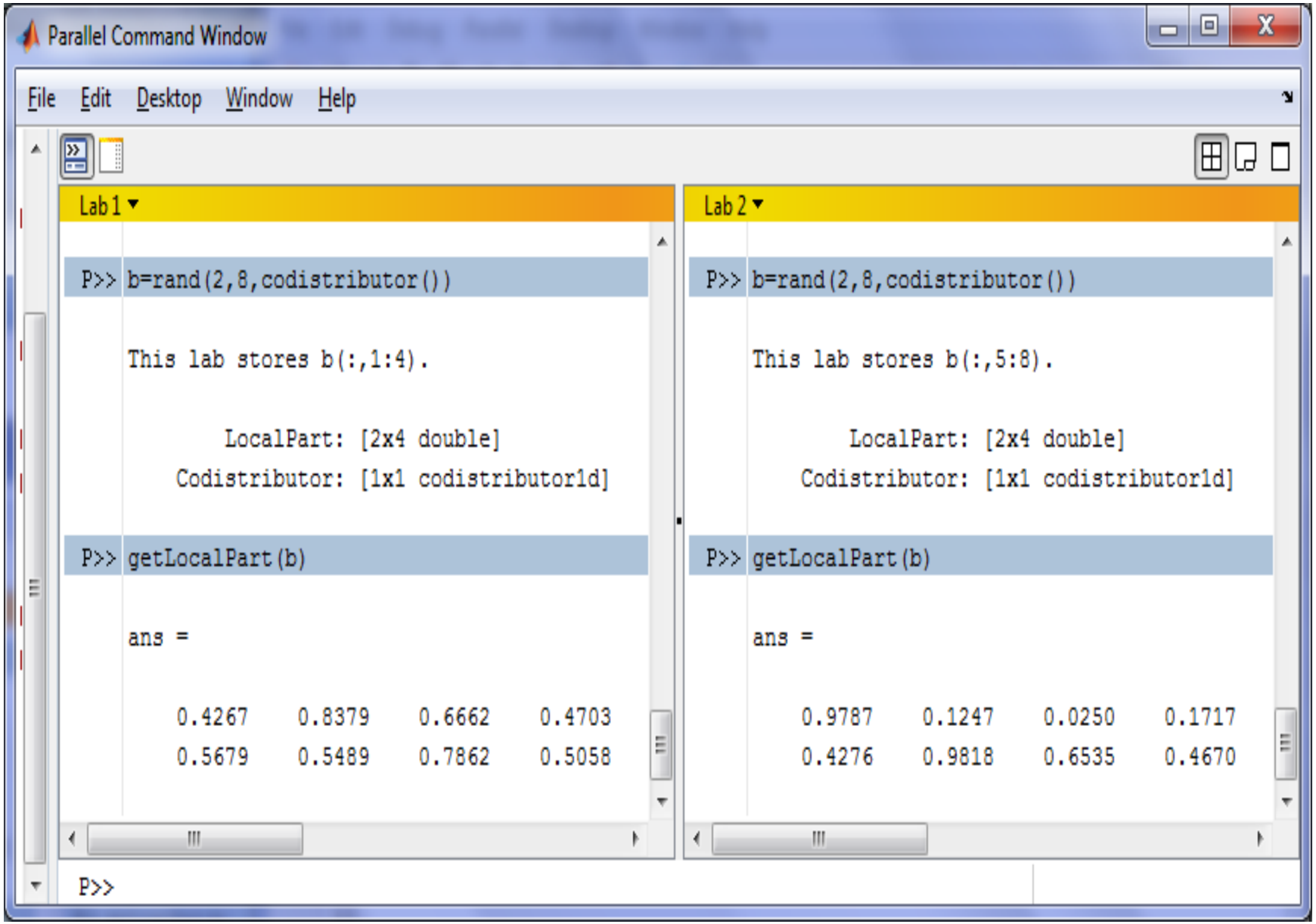
whos combined

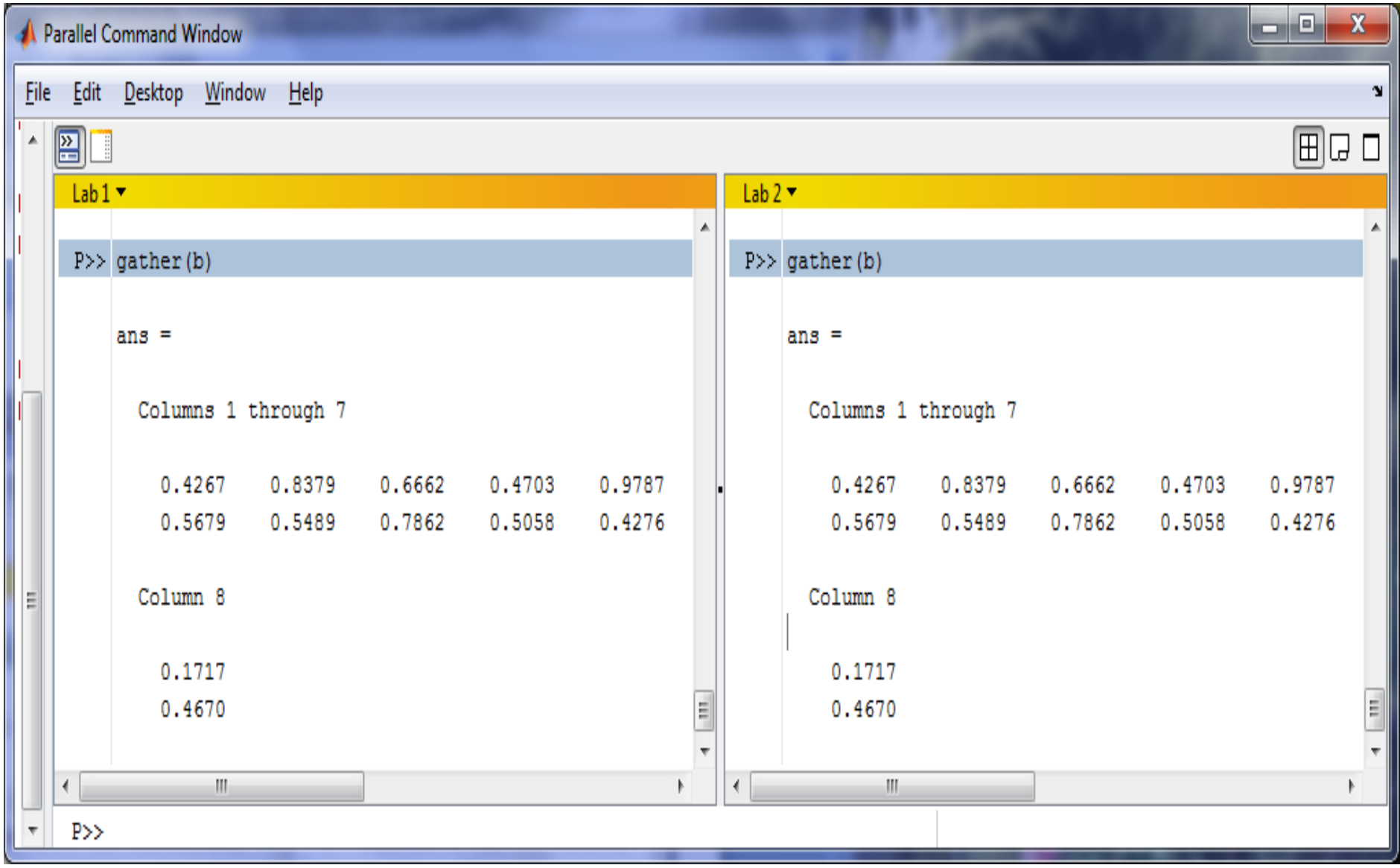
To see the array, type its name.

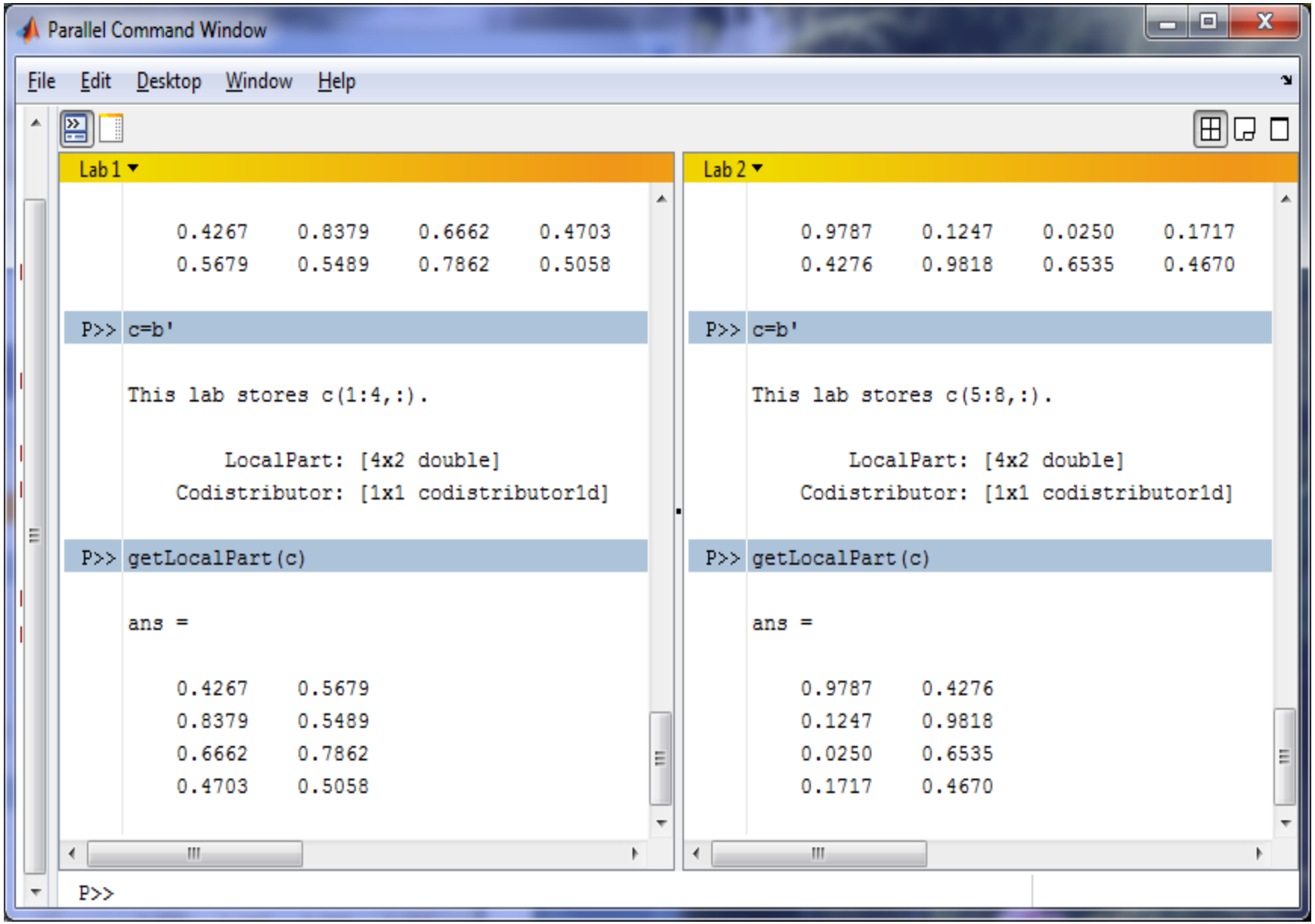
combined

Demo: distributed array operations (repeat)




Parallel Command Window
File Edit Desktop Window Help
Lab 1
P>> b=rand(2,8,codistributor())
This lab stores b(:,1:4).
LocalPart: [2x4 double]
Codistributor: [1x1 codistributor1d]
P>> getLocalPart(b)
ans =
0.4267 0.8379 0.6662 0.4703
0.5679 0.5489 0.7862 0.5058
Lab 2
P>> b=rand(2,8,codistributor())
This lab stores b(:,5:8).
LocalPart: [2x4 double]
Codistributor: [1x1 codistributor1d]
P>> getLocalPart(b)
ans =
0.9787 0.1247 0.0250 0.1717
0.4276 0.9818 0.6535 0.4670
P>>


Parallel Command Window
File Edit Desktop Window Help
Lab 1
P>> gather(b)
ans =
Columns 1 through 7
0.4267 0.8379 0.6662 0.4703 0.9787
0.5679 0.5489 0.7862 0.5058 0.4276
Column 8
0.1717
0.4670
Lab 2
P>> gather(b)
ans =
Columns 1 through 7
0.4267 0.8379 0.6662 0.4703 0.9787
0.5679 0.5489 0.7862 0.5058 0.4276
Column 8
0.1717
0.4670
P>>


Parallel Command Window
File Edit Desktop Window Help
Lab 1
0.4267 0.8379 0.6662 0.4703
0.5679 0.5489 0.7862 0.5058
P>> c=b'
This lab stores c(1:4,:).
LocalPart: [4x2 double]
Codistributor: [1x1 codistributor1d]
P>> getLocalPart(c)
ans =
0.4267 0.5679
0.8379 0.5489
0.6662 0.7862
0.4703 0.5058
Lab 2
0.9787 0.1247 0.0250 0.1717
0.4276 0.9818 0.6535 0.4670
P>> c=b'
This lab stores c(5:8,:).
LocalPart: [4x2 double]
Codistributor: [1x1 codistributor1d]
P>> getLocalPart(c)
ans =
0.9787 0.4276
0.1247 0.9818
0.0250 0.6535
0.1717 0.4670
P>>

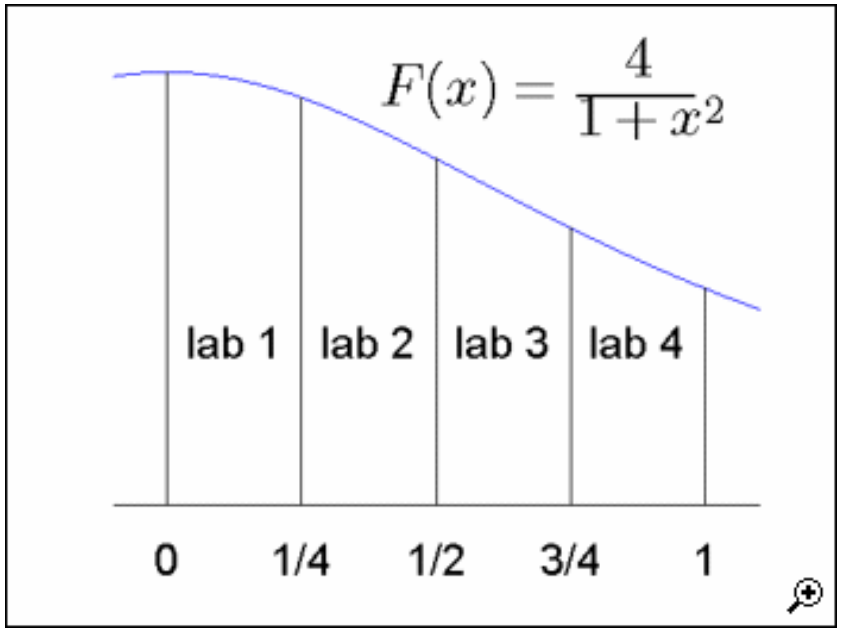
Parallel pi in pmode

use the fact that

$$\int_0^1 \frac{4}{1+x^2} dx = 4(atan(1) - atan(0)) = \pi$$

to approximate pi by approximating the integral on the left.

divide the work between the labs by having each lab calculate the integral the function over a subinterval of [0, 1] as shown in the picture



Steps

- All labs/workers will compute the same function: F=4/(1+x^2)

- Each worker/lab will calculate over a subinterval [a,b] of [0, 1], for 2 labs, the subinterval will be:

  [0, 0.50]

  [0.50, 1.0]

  a = (labindex-1)/numlabs

  b = labindex/numlabs

- Use a MATLAB quadrature method to compute the integral

  myIntegral = quadl(F, a, b)

- Add together to form the entire integral over [0,1]

  piApprox = gplus(myIntegral)

Parallel Command Window

File Edit Desktop Window Help

Lab 1

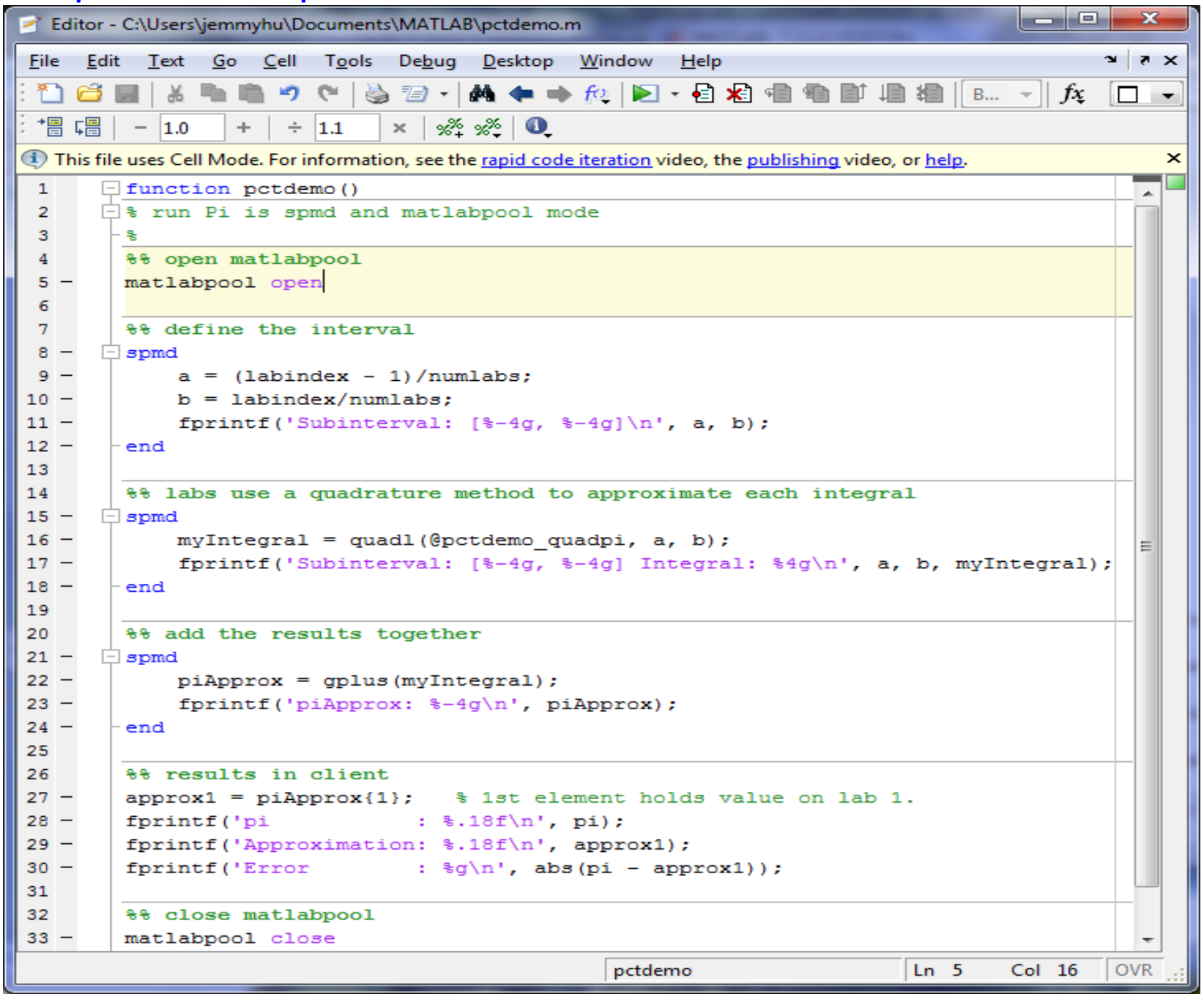
```
P>> clear
P>> F=@(x)4./(1+x.^2)

F =

    @(x)4./(1+x.^2)

P>> a=(labindex-1)/numlabs;
P>> b=labindex/numlabs;
P>> [a, b]

ans =

         0    0.5000

P>> myIntergral=quadl(F,a,b)

myIntergral =

    1.8546

P>> piApprox=gplus(myIntergral)

piApprox =

    3.1416

P>> abs(pi-piApprox)

ans =

  2.4865e-010
```

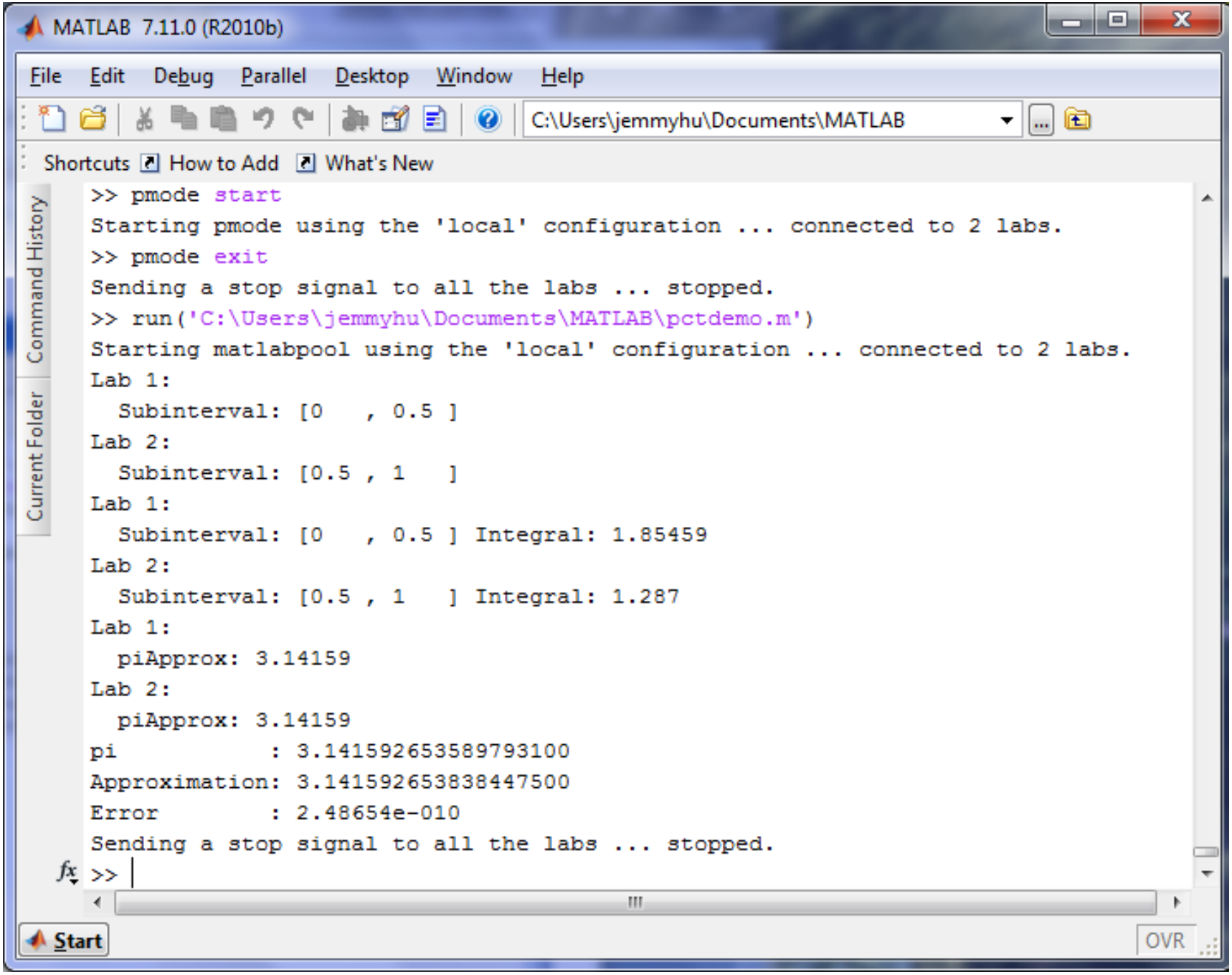
Lab 2

```
P>> clear
P>> F=@(x)4./(1+x.^2)

F =

    @(x)4./(1+x.^2)

P>> a=(labindex-1)/numlabs;
P>> b=labindex/numlabs;
P>> [a, b]

ans =

    0.5000    1.0000

P>> myIntergral=quadl(F,a,b)

myIntergral =

    1.2870

P>> piApprox=gplus(myIntergral)

piApprox =

    3.1416

P>> abs(pi-piApprox)

ans =

  2.4865e-010
```

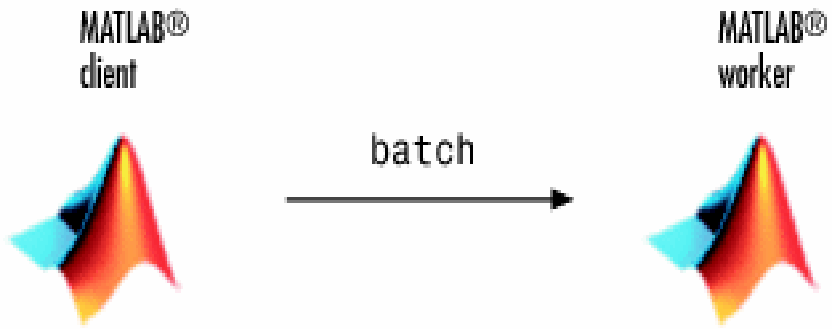
P>>

Parallel pi in matlabpool-mode




MATLAB 7.11.0 (R2010b)
File Edit Debug Parallel Desktop Window Help
C:\Users\jemmyhu\Documents\MATLAB
Shortcuts How to Add What's New
Command History
Current Folder
>> pmode start
Starting pmode using the 'local' configuration ... connected to 2 labs.
>> pmode exit
Sending a stop signal to all the labs ... stopped.
>> run('C:\Users\jemmyhu\Documents\MATLAB\pctdemo.m')
Starting matlabpool using the 'local' configuration ... connected to 2 labs.
Lab 1:
Subinterval: [0 , 0.5]
Lab 2:
Subinterval: [0.5 , 1]
Lab 1:
Subinterval: [0 , 0.5] Integral: 1.85459
Lab 2:
Subinterval: [0.5 , 1] Integral: 1.287
Lab 1:
piApprox: 3.14159
Lab 2:
piApprox: 3.14159
pi : 3.141592653589793100
Approximation: 3.141592653838447500
Error : 2.48654e-010
Sending a stop signal to all the labs ... stopped.
>>
Start
OVR

Batch mode



- Name a .m file as ‘mybatch’ with
```
for i=1:1024
A(i) = sin(i*2*pi/1024);
end
```
- run in batch mode
```
job = batch('mybatch')
```

- The batch command does not block MATLAB, so you must wait for the job to finish before you can retrieve and view its results:
```
wait(job)
```
- The load command transfers variables from the workspace of the worker to the workspace of the client, where you can view the results:
```
load(job, 'A')
plot(A)
```
- When the job is complete, permanently remove its data:
```
destroy(job)
```

MATLAB 7.11.0 (R2010b)

File Edit Debug Parallel Desktop Window Help

C:\Users\jemmyhu\Documents\MATLAB

Shortcuts How to Add What's New

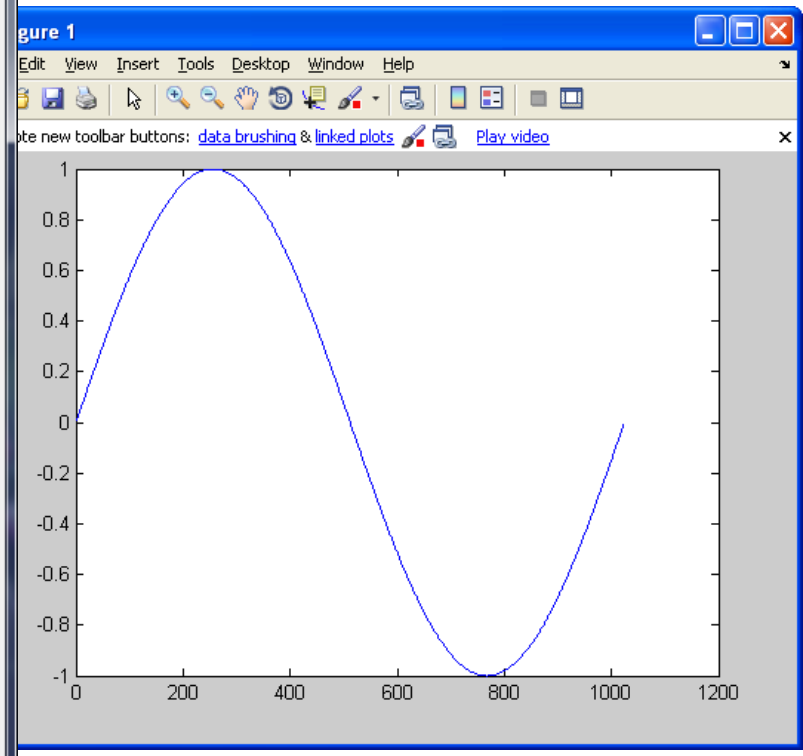
Command History

Current Folder

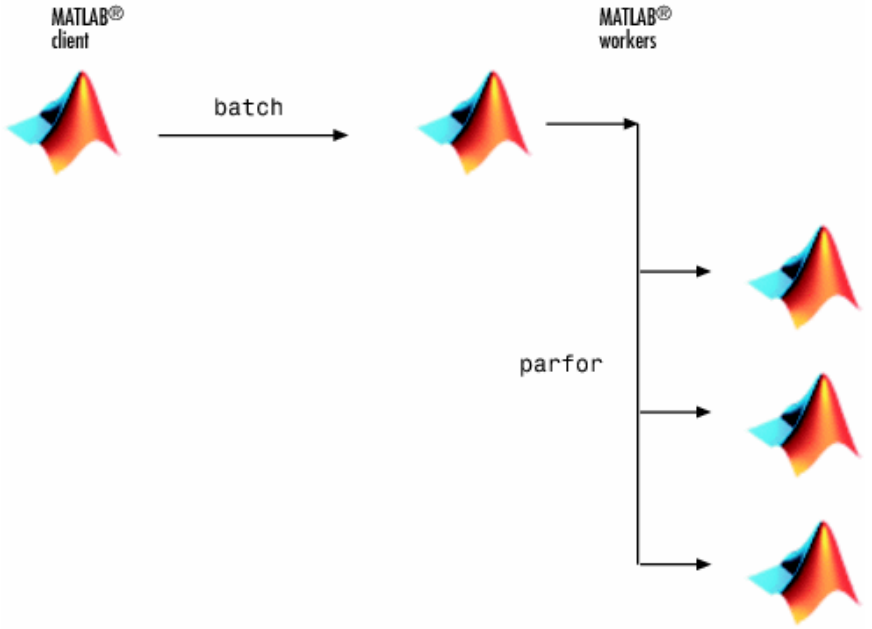
```
>>
>> job=batch('mybatch')

job =

Job ID 13 Information
=====================

                UserName : jemmyhu
                   State : running
              SubmitTime : Thu Jun 02 11:01:55 EDT 2011
               StartTime :
          Running Duration :

- Data Dependencies

          FileDependencies : C:\Users\jemmyhu\Documents\MATLAB\mybatch.m
          PathDependencies : {}

- Associated Task(s)

           Number Pending  : 1
           Number Running  : 0
           Number Finished : 0
          TaskID of errors :
>> wait(job)
>> load(job, 'A')
>> plot(A)
>>
```

Start

OVR



A batch parallel loop

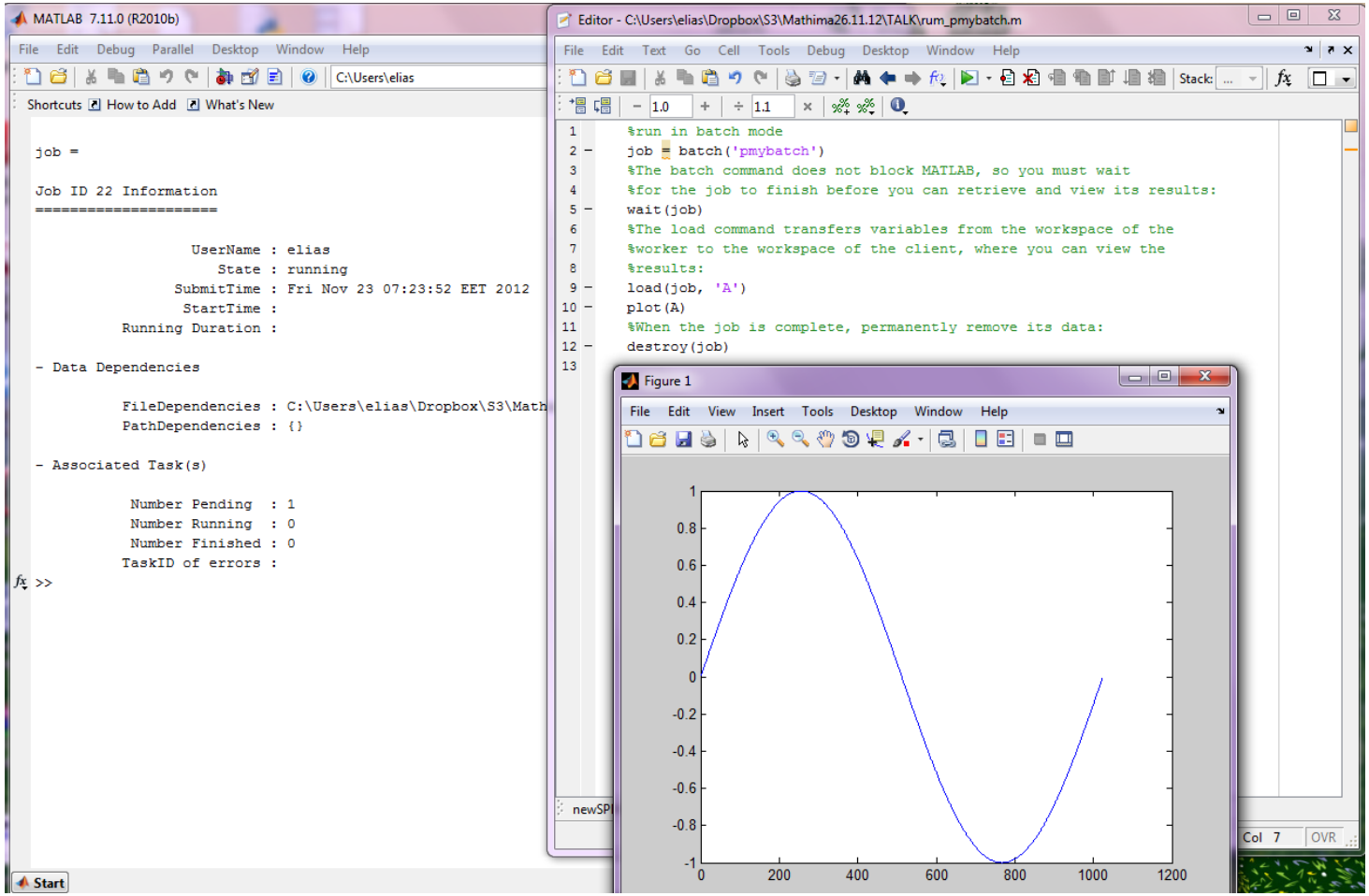



```
% mybatch
parfor i=1:1024
A(i) = sin(i*2*pi/1024);
end
```

```
% run job in batch
job = batch('mybatch', ‘configuration, ‘local’, 'matlabpool', 2)
```

```
% To view the results:
wait(job)
load(job, 'A')
plot(A)
```

```
% remove its data:
destroy(job)
```


MATLAB 7.11.0 (R2010b)
File Edit Debug Parallel Desktop Window Help
C:\Users\elias
Shortcuts How to Add What's New
job =
Job ID 22 Information
UserName : elias
State : running
SubmitTime : Fri Nov 23 07:23:52 EET 2012
StartTime :
Running Duration :
- Data Dependencies
FileDependencies : C:\Users\elias\Dropbox\S3\Math
PathDependencies : {}
- Associated Task(s)
Number Pending : 1
Number Running : 0
Number Finished : 0
TaskID of errors :
>>
Start
Editor - C:\Users\elias\Dropbox\S3\Mathima26.11.12\TALK\rum_pmybatch.m
File Edit Text Go Cell Tools Debug Desktop Window Help
1 %run in batch mode
2 job = batch('pmybatch')
3 %The batch command does not block MATLAB, so you must wait
4 %for the job to finish before you can retrieve and view its results:
5 wait(job)
6 %The load command transfers variables from the workspace of the
7 %worker to the workspace of the client, where you can view the
8 %results:
9 load(job, 'A')
10 plot(A)
11 %When the job is complete, permanently remove its data:
12 destroy(job)
13
Col 7
OVR
Figure 1
File Edit View Insert Tools Desktop Window Help
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
200
400
600
800
1000
1200

MATLAB 7.11.0 (R2010b)

File Edit Debug Parallel Desktop Window Help

Current Folder: C:\Users\jemmyhu\Documents\MATLAB

Shortcuts How to Add What's New

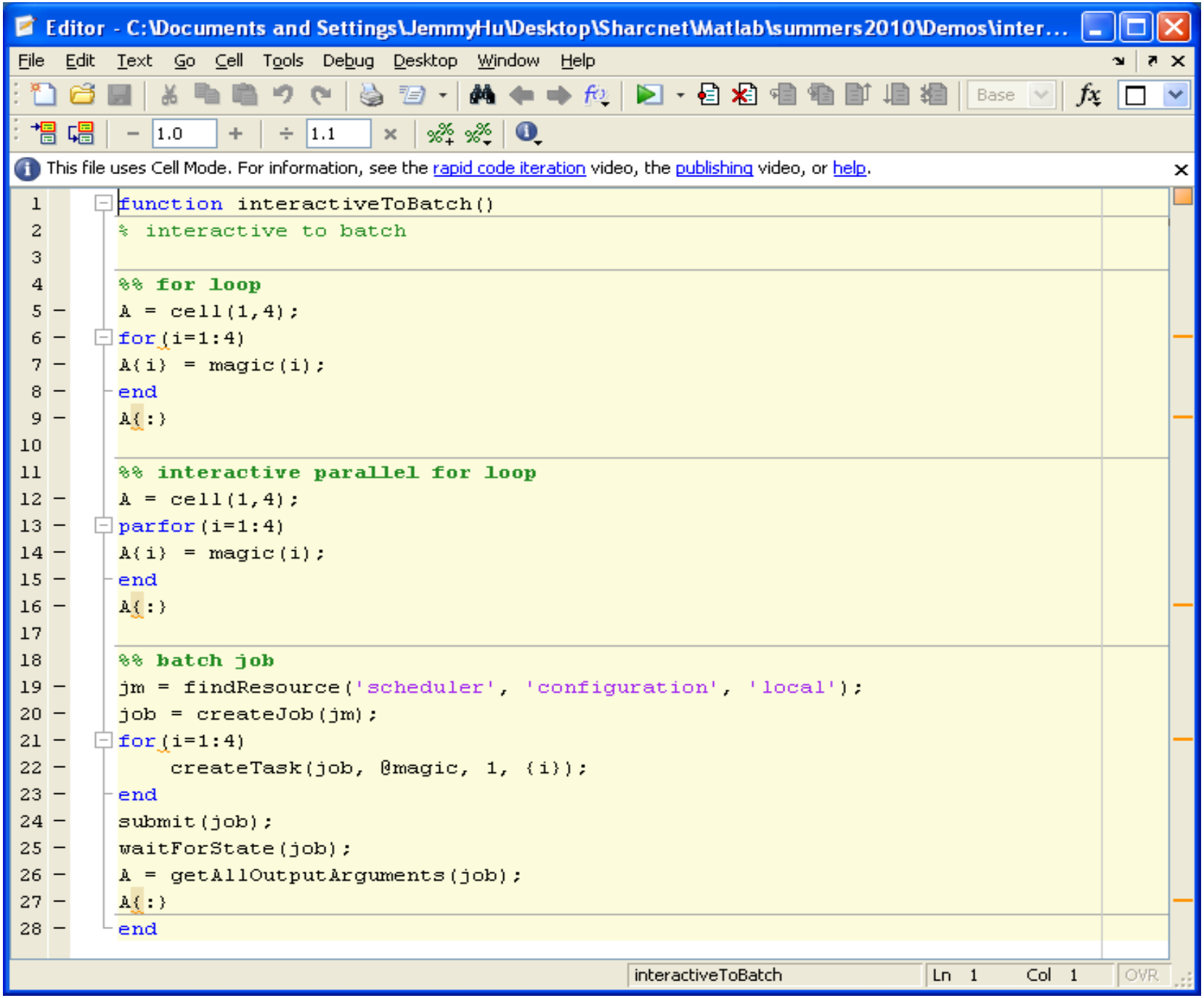
Command History

Current Folder

```
>> job=batch('mybatch_par', 'configuration', 'local', 'matlabpool', 1)

job =

MatlabPool Job ID 17 Information
================================

                  UserName : jemmyhu
                     State : queued
                SubmitTime : Thu Jun 02 11:09:39 EDT 2011
                 StartTime :
           Running Duration :

- Data Dependencies

           FileDependencies : C:\Users\jemmyhu\Documents\MATLAB\mybatch_par.m
           PathDependencies : {}

- Associated Task(s)

            Number Pending  : 2
            Number Running  : 0
            Number Finished : 0
           TaskID of errors :

- Scheduler Dependent (MatlabPool Job)

    MaximumNumberOfWorkers : 2
    MinimumNumberOfWorkers : 2
>> wait(job)
>> load(job, 'A')
>> plot(A)
>> destroy(job)
>>
```

Start

OVR


Editor - C:\Documents and Settings\JemmyHu\Desktop\Sharcnet\Matlab\summers2010\Demos\inter...
File Edit Text Go Cell Tools Debug Desktop Window Help
Base
1.0
1.1
This file uses Cell Mode. For information, see the rapid code iteration video, the publishing video, or help.
function interactiveToBatch()
% interactive to batch
%% for loop
A = cell(1,4);
for(i=1:4)
A{i} = magic(i);
end
A{:}
%% interactive parallel for loop
A = cell(1,4);
parfor(i=1:4)
A{i} = magic(i);
end
A{:}
%% batch job
jm = findResource('scheduler', 'configuration', 'local');
job = createJob(jm);
for(i=1:4)
createTask(job, @magic, 1, {i});
end
submit(job);
waitForState(job);
A = getAllOutputArguments(job);
A{:}
end
interactiveToBatch
Ln 1
Col 1
OVR

Key Function List

- **Job Creation**
  createJob Create job object in scheduler and client
  createTask Create new task in job
  dfeval Evaluate function using cluster

- **Interlab Communication Within a Parallel Job**
  labBarrier Block execution until all labs reach this call
  labBroadcast Send data to all labs or receive data sent to all labs
  labindex Index of this lab
  labReceive Receive data from another lab
  labSend Send data to another lab
  numlabs Total number of labs operating in parallel on current job

- **Job Management**
  cancel Cancel job or task
  destroy Remove job or task object from parent and memory
  getAllOutputArguments Output arguments from evaluation of all tasks in job object
  submit Queue job in scheduler
  wait Wait for job to finish or change states

Typical Use Cases

- **Parallel for-Loops (parfor)**

  allowing several MATLAB workers to execute individual loop iterations simultaneously

  restriction on parallel loops is that no iterations be allowed to depend on any other iterations.

- **Large Data Sets**

  allows you to distribute that array among multiple MATLAB workers, so that each worker contains only a part of the array

  Each worker operates only on its part of the array, and workers automatically transfer data between themselves when necessary

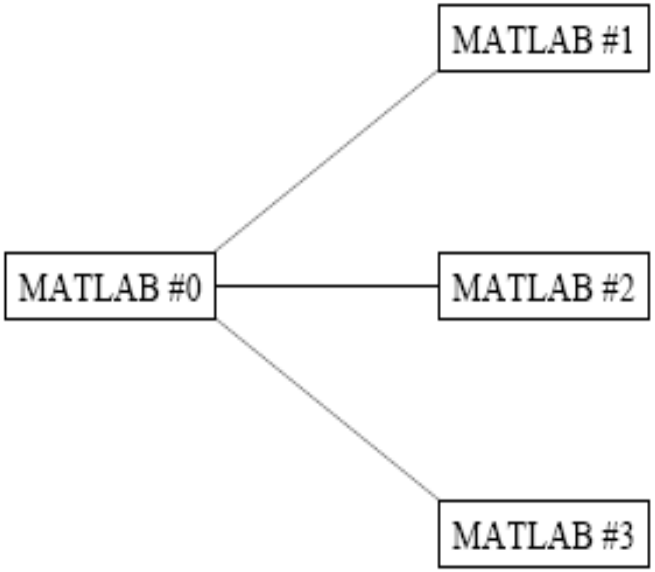
- **Batch Jobs**

  offload work to a MATLAB worker session to run as a batch job.

  the MATLAB worker can run either on the same machine as the client, or if using MATLAB Distributed Computing Server, on a remote cluster machine.

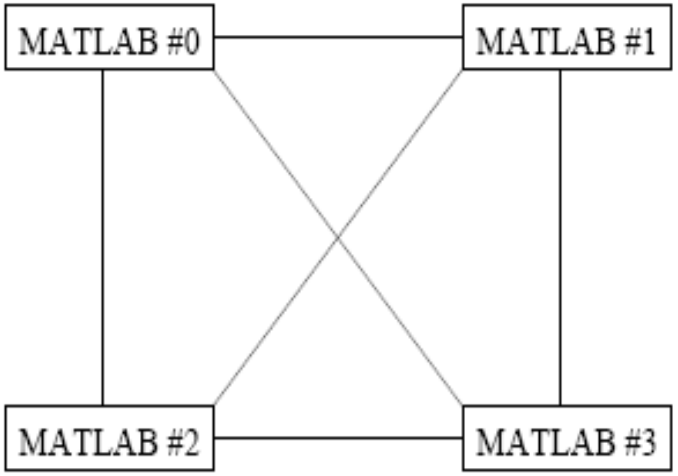
Matlab in Parallel

Beautifully parallel



e.g., Multi, paralize, Plab, ParMatlab

Message Passing



e.g., MultiMatlab, CMTM, DPToolbox,MatlabMPI, pMatlab

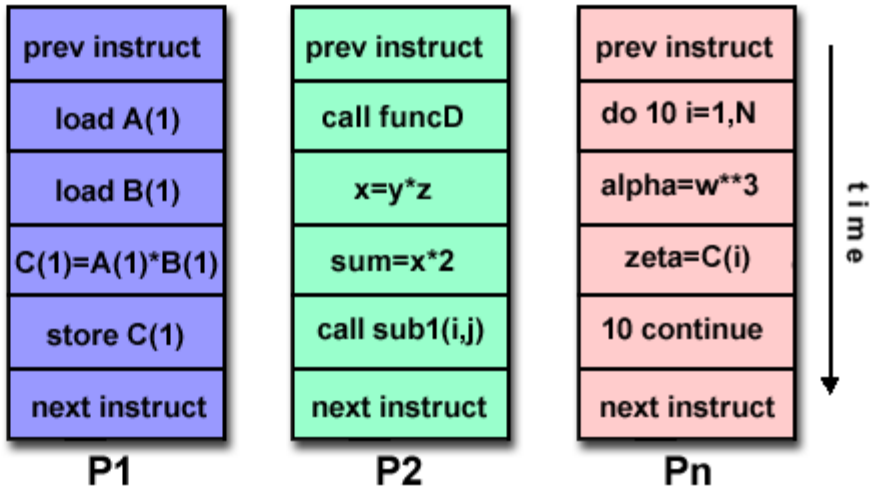
www-math.mit.edu/~edelman/homepage/papers/pmatlab.pdf

MPI+SMP Parallel Paradigms

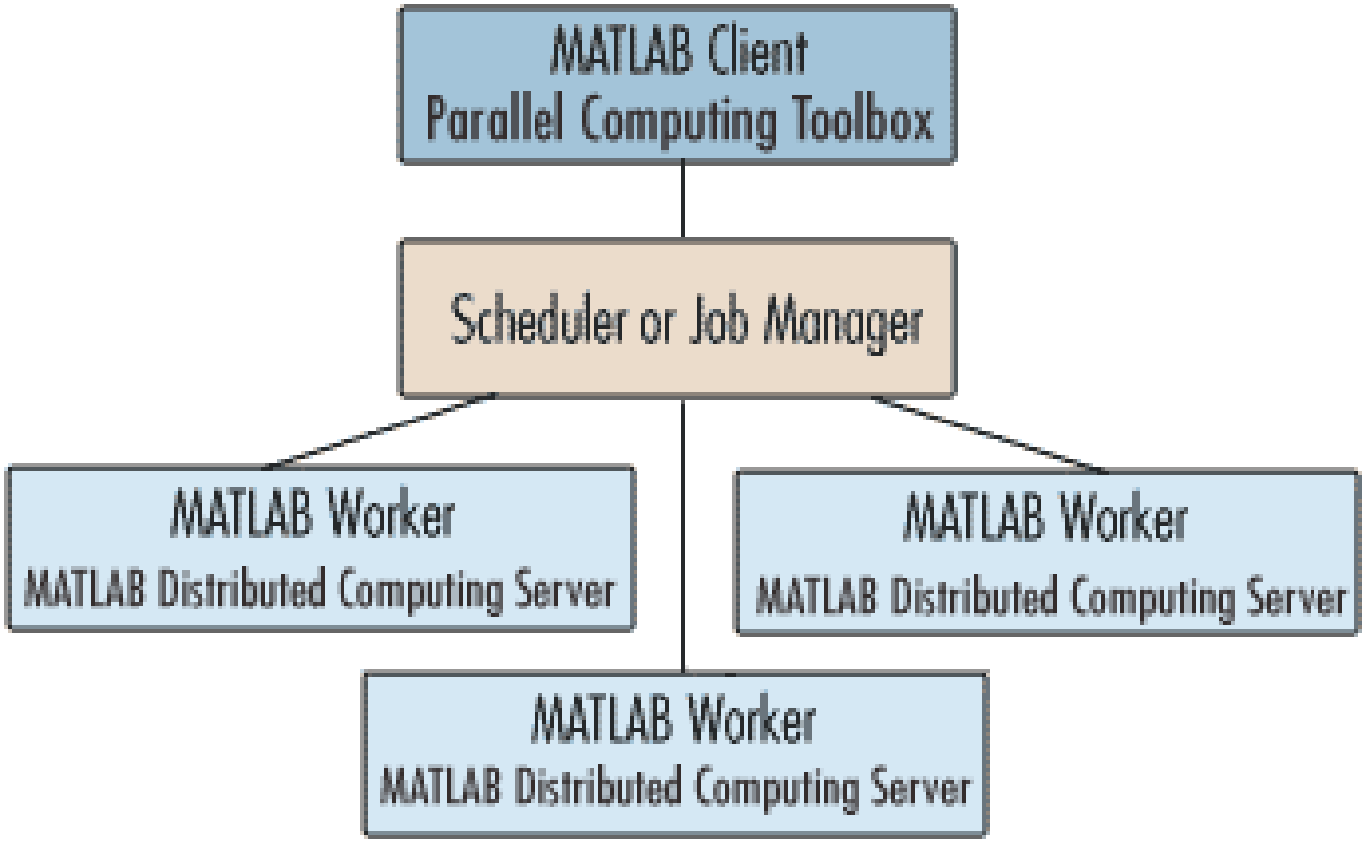
- http://ist.uwaterloo.ca/ew/saw/parallel/FLASH/swf/mpi+smp.swf

MIMD

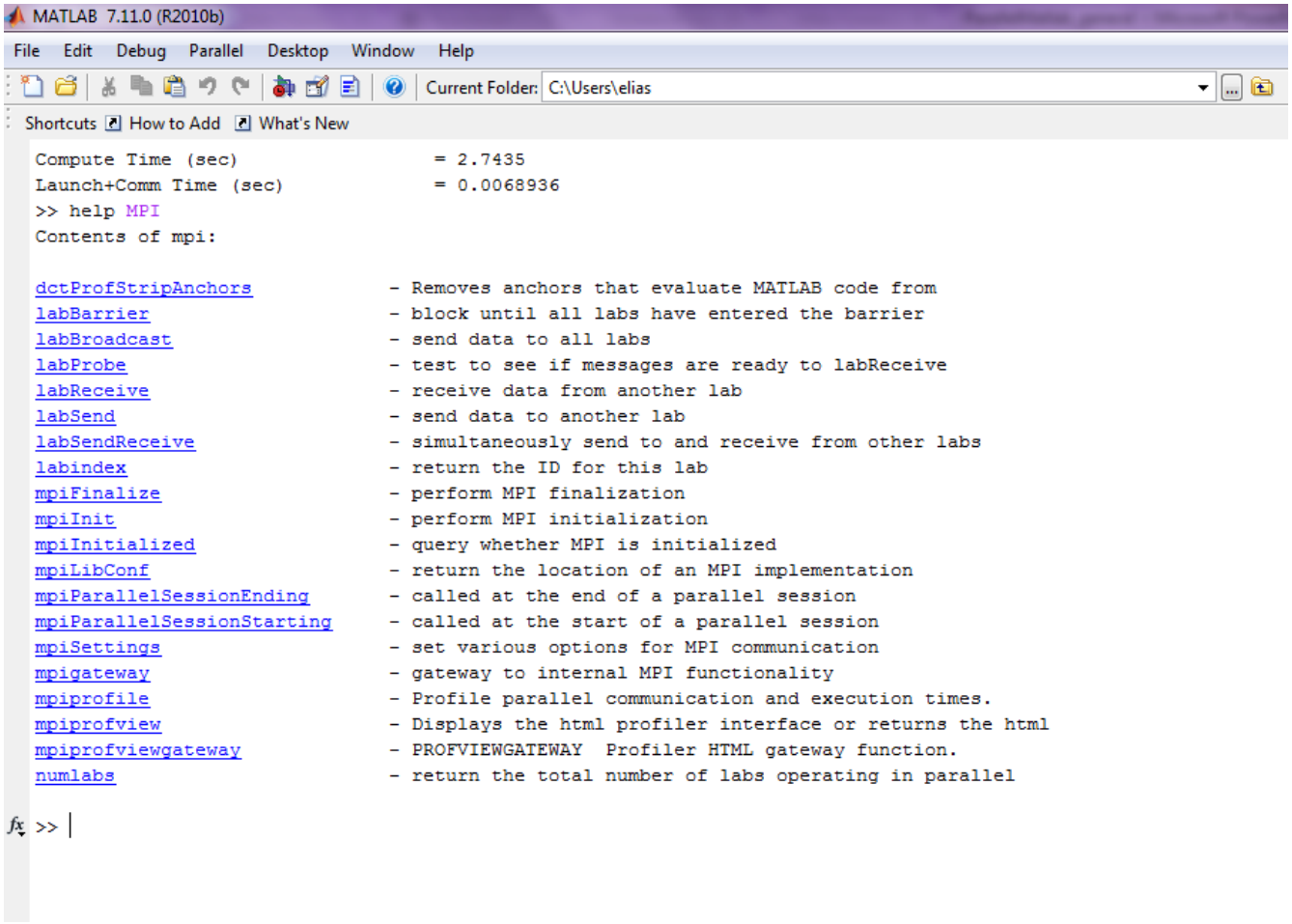
- Currently, most common type of parallel computer
- Every processor may be executing a different instruction stream
- Every processor may be working with a different data stream
- Execution can be synchronous or asynchronous, deterministic or non-deterministic
- Examples: most current supercomputers, networked parallel computer clusters and "grids", multi-processor SMP computers, multi-core PCs.
- Note: many MIMD architectures also include SIMD execution sub-components



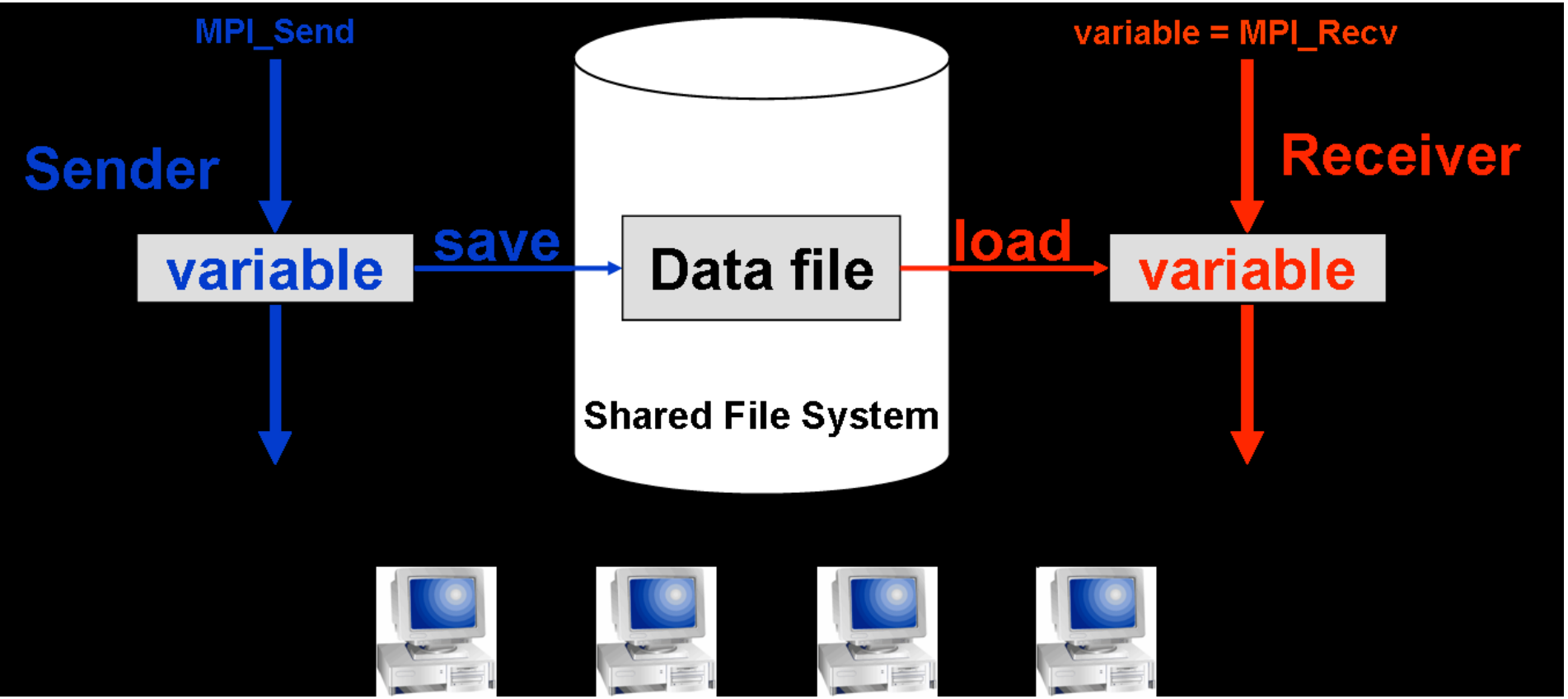
Parallel Computing Toolbox and MATLAB Distributed Computing



MPI Library

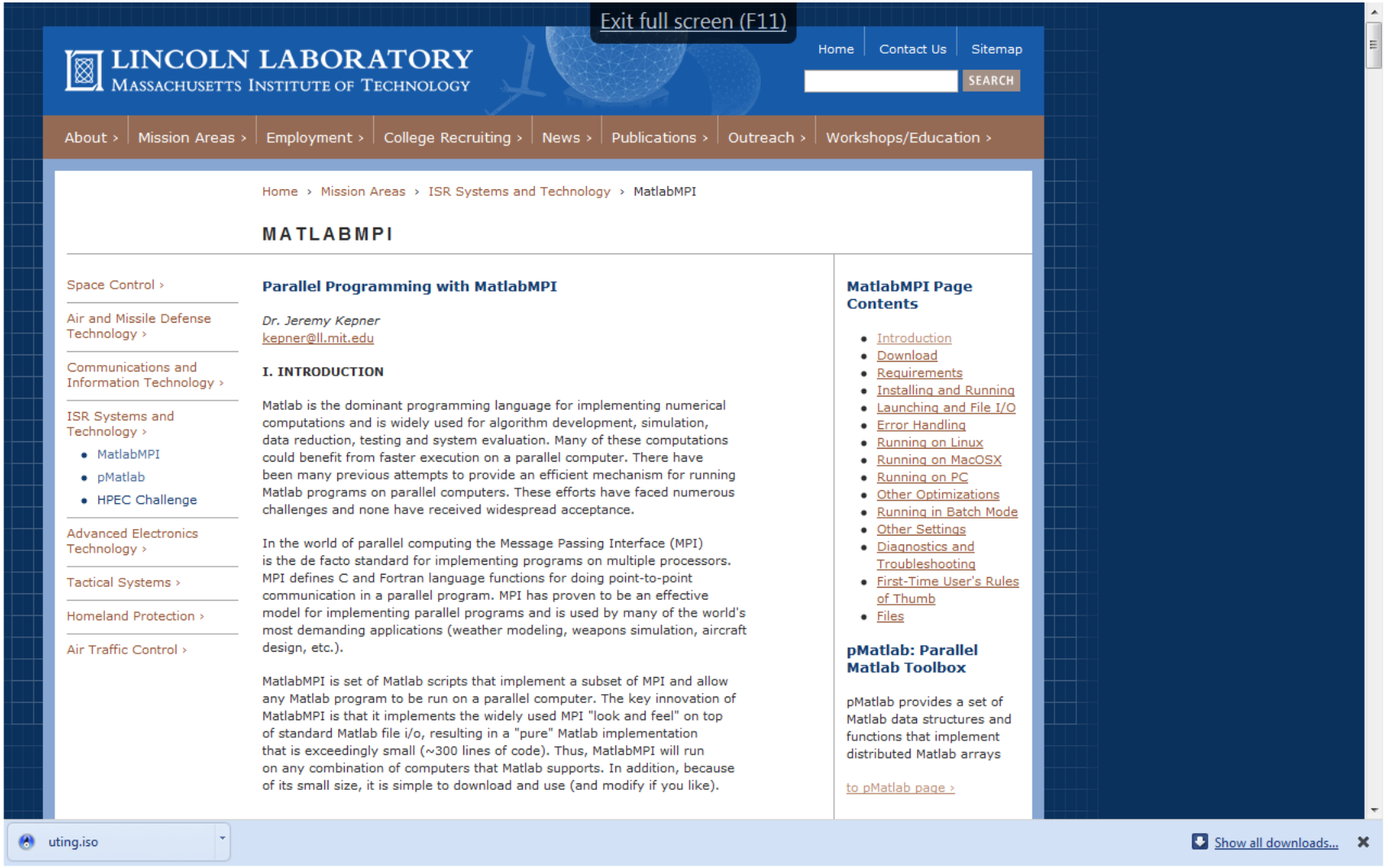

MATLAB 7.11.0 (R2010b)
File Edit Debug Parallel Desktop Window Help
Current Folder: C:\Users\elias
Shortcuts How to Add What's New
Compute Time (sec) = 2.7435
Launch+Comm Time (sec) = 0.0068936
>> help MPI
Contents of mpi:
dctProfStripAnchors - Removes anchors that evaluate MATLAB code from
labBarrier - block until all labs have entered the barrier
labBroadcast - send data to all labs
labProbe - test to see if messages are ready to labReceive
labReceive - receive data from another lab
labSend - send data to another lab
labSendReceive - simultaneously send to and receive from other labs
labindex - return the ID for this lab
mpiFinalize - perform MPI finalization
mpiInit - perform MPI initialization
mpiInitialized - query whether MPI is initialized
mpiLibConf - return the location of an MPI implementation
mpiParallelSessionEnding - called at the end of a parallel session
mpiParallelSessionStarting - called at the start of a parallel session
mpiSettings - set various options for MPI communication
mpigateway - gateway to internal MPI functionality
mpiprofile - Profile parallel communication and execution times.
mpiprofview - Displays the html profiler interface or returns the html
mpiprofviewgateway - PROFVIEWGATEWAY Profiler HTML gateway function.
numlabs - return the total number of labs operating in parallel
>>

MatlabMPI implements the fundamental communication operations in MPI using MATLAB's file I/O functions.

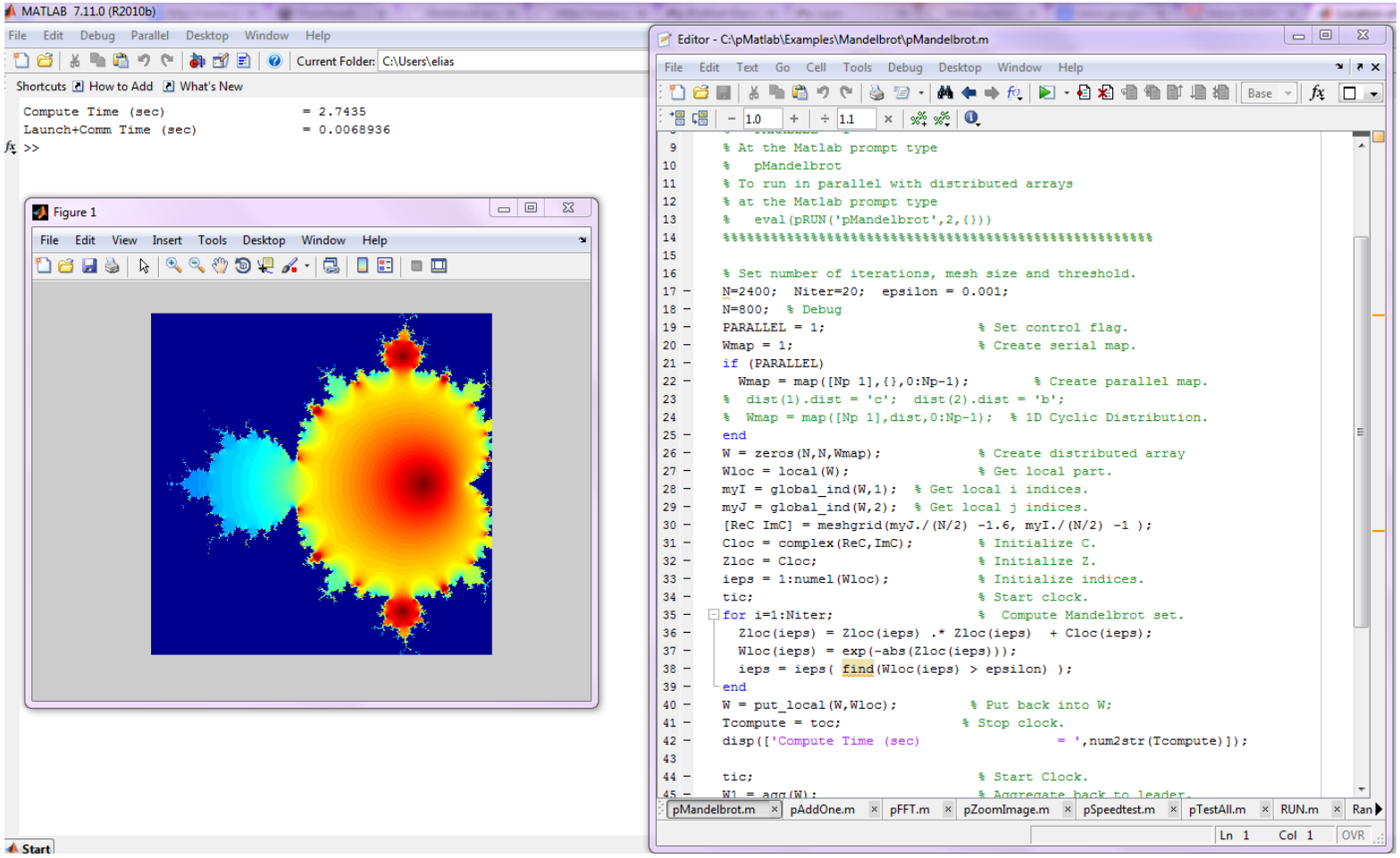


MatlabMPI

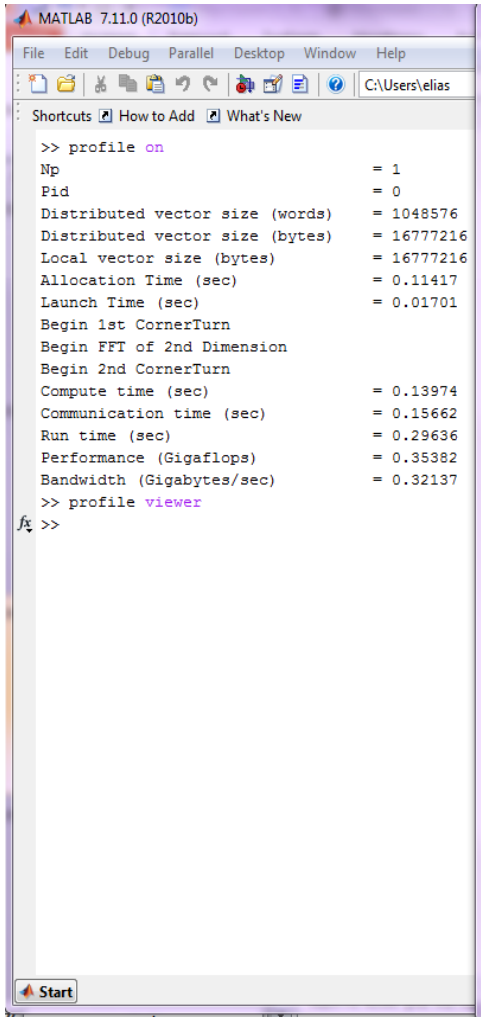
http://www.ll.mit.edu/mission/isr/matlabmpi/matlabmpi.html#introduction

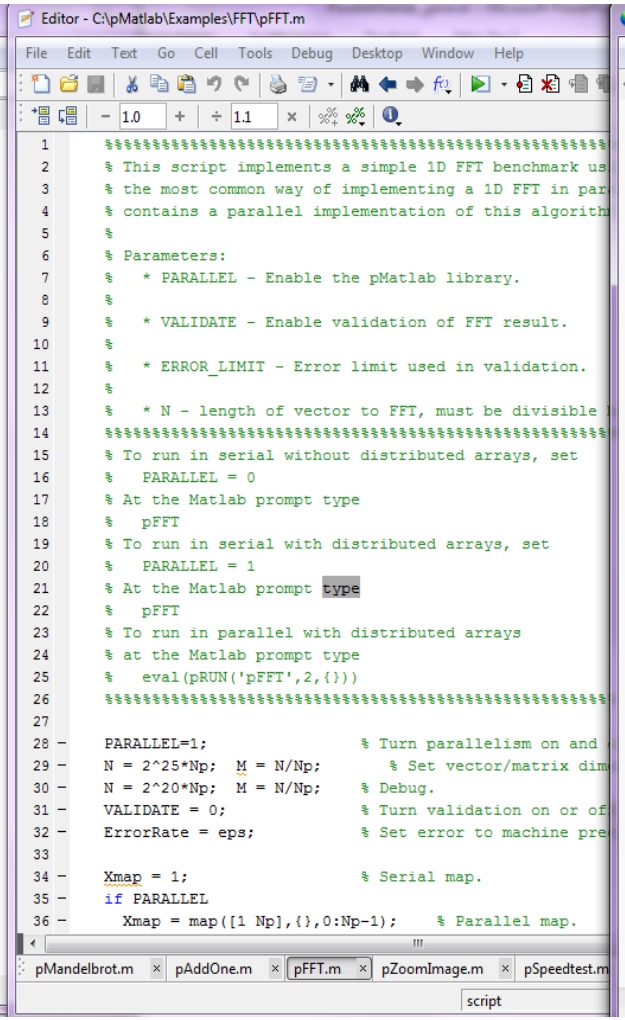


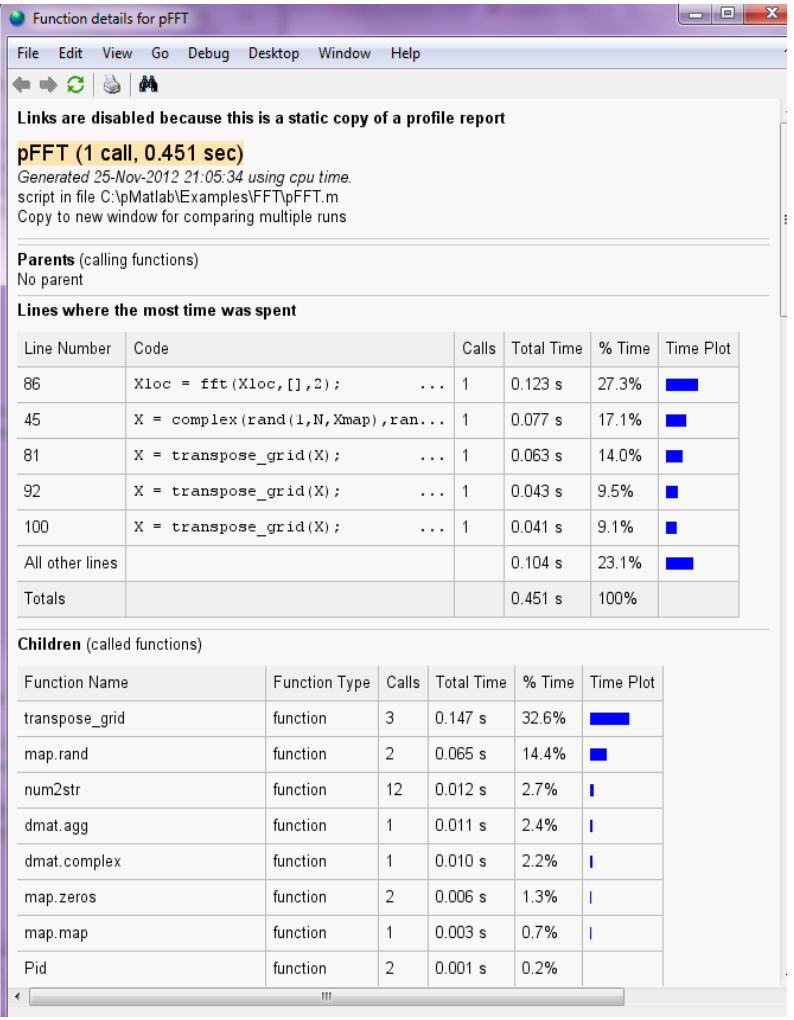
pMatLab example



FFT with pMatLab







Available examples:

xbasic.m Extremely simple MatlabMPI program that prints out the rank of each processor.

basic.m Simple MatlabMPI program that sends data from processor 1 to processor 0.

multi_basic.m Simple MatlabMPI program that sends data from processor 1 to processor 0 a few times.

probe.m Simple MatlabMPI program that demonstrates the using MPI_Probe to check for incoming messages.

broadcast.m Tests MatlabMPI broadcast command.

basic_app.m Examples of the most common usages of MatlabMPI.

basic_app2.m Examples of the most common usages of MatlabMPI.

basic_app3.m Examples of the most common usages of MatlabMPI.

basic_app4.m Examples of the most common usages of MatlabMPI.

blurimage.m MatlabMPI test parallel image processing application.

speedtest.m Times MatlabMPI for a variety of messages.

synch_start.m Function for synchronizing starts.

machines.m Example script for creating a machine description.

unit_test.m Wrapper for using an example as a unit test.

unit_test_all.m Calls all of the examples as way of testing the entire library.

unit_test_mcc.m Wrapper for using an example as a mcc unit test.

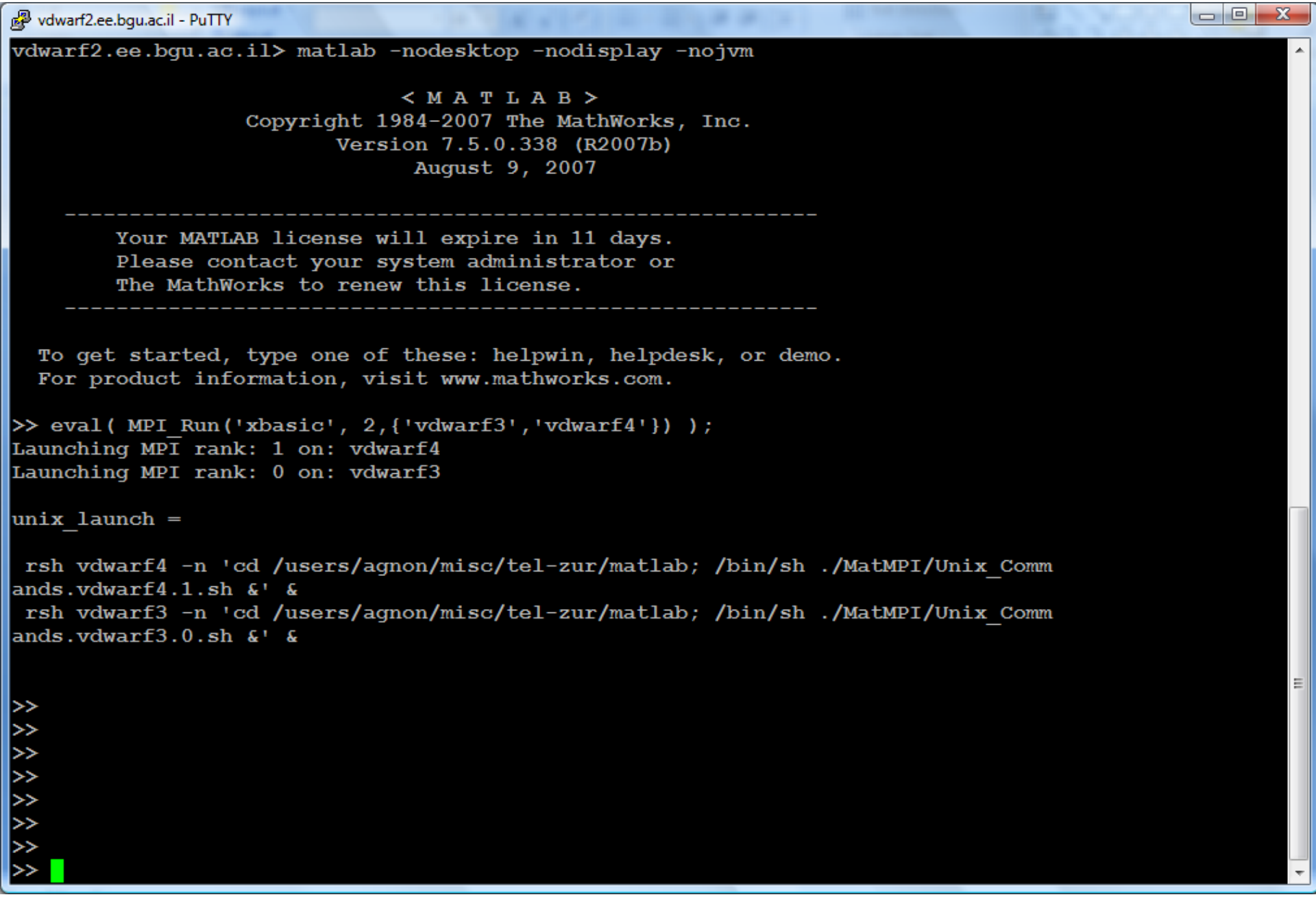
unit_test_all_mcc.m Calls all of the examples using MPI_cc as way of testing the entire library.

xbasic

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Basic Matlab MPI script that
% prints out a rank.
%
% To run, start Matlab and type:
%
%   eval( MPI_Run('xbasic',2,{}) );
%
% Or, to run a different machine type:
%
%   eval( MPI_Run('xbasic',2,{'machine1' 'machine2'}) );
%
% Output will be piped into two files:
%
%   MatMPI/xbasic.0.out
%   MatMPI/xbasic.1.out
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% MatlabMPI
% Dr. Jeremy Kepner
% MIT Lincoln Laboratory
% kepner@ll.mit.edu
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Initialize MPI.
MPI_Init;

% Create communicator.
comm = MPI_COMM_WORLD;

% Modify common directory from default for better performance.
% comm = MatMPI_Comm_dir(comm,'/tmp');

% Get size and rank.
comm_size = MPI_Comm_size(comm);
my_rank = MPI_Comm_rank(comm);

% Print rank.
disp(['my_rank: ',num2str(my_rank)]);

% Wait momentarily.
pause(2.0);

% Finalize Matlab MPI.
MPI_Finalize;
disp('SUCCESS');
if (my_rank ~= MatMPI_Host_rank(comm))
  exit;
end
```

Demo folder ~/matlab/, watch top at the other machine



Parallel Matlab (Octave) using pMatlab

Global arrays – "…Communication is hidden from the programmer; arrays are automatically redistributed when necessary, without the knowledge of the programmer…"

"…The ultimate goal of pMatlab is to move beyond basic messaging (and its inherent programming complexity) towards higher level parallel data structures and functions, allowing any MATLAB user to parallelize their existing program by simply changing and adding a few lines,

Source: http://www.ll.mit.edu/mission/isr/pmatlab/pMatlab_intro.pdf

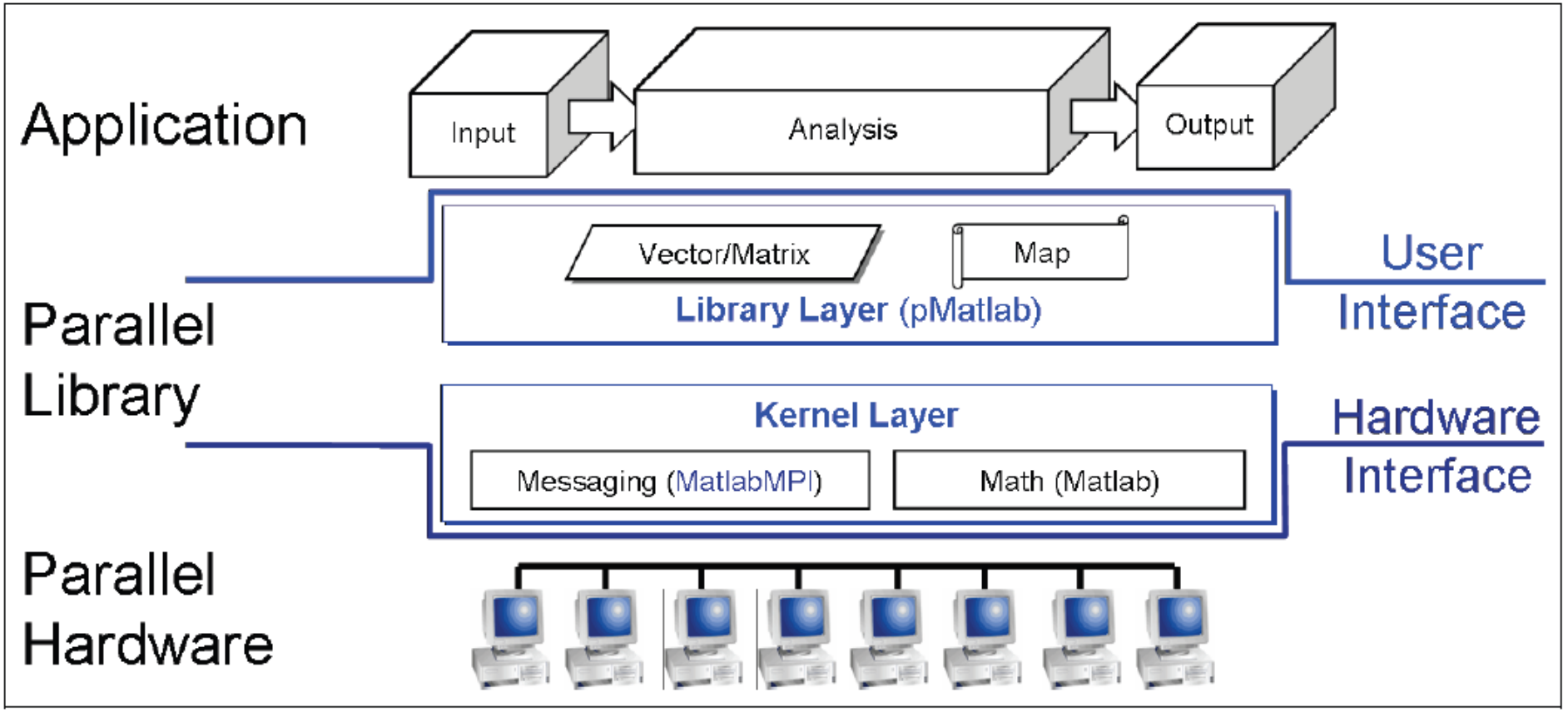


Figure 11 – Parallel MATLAB consists of two layers. pMatlab provides parallel data structures and library functions. MatlabMPI provides messaging capability.

Parallel Computing with Matlab on Amazon Cloud

MATLAB Parallel Computing Tools: Basic Setup and Requirements

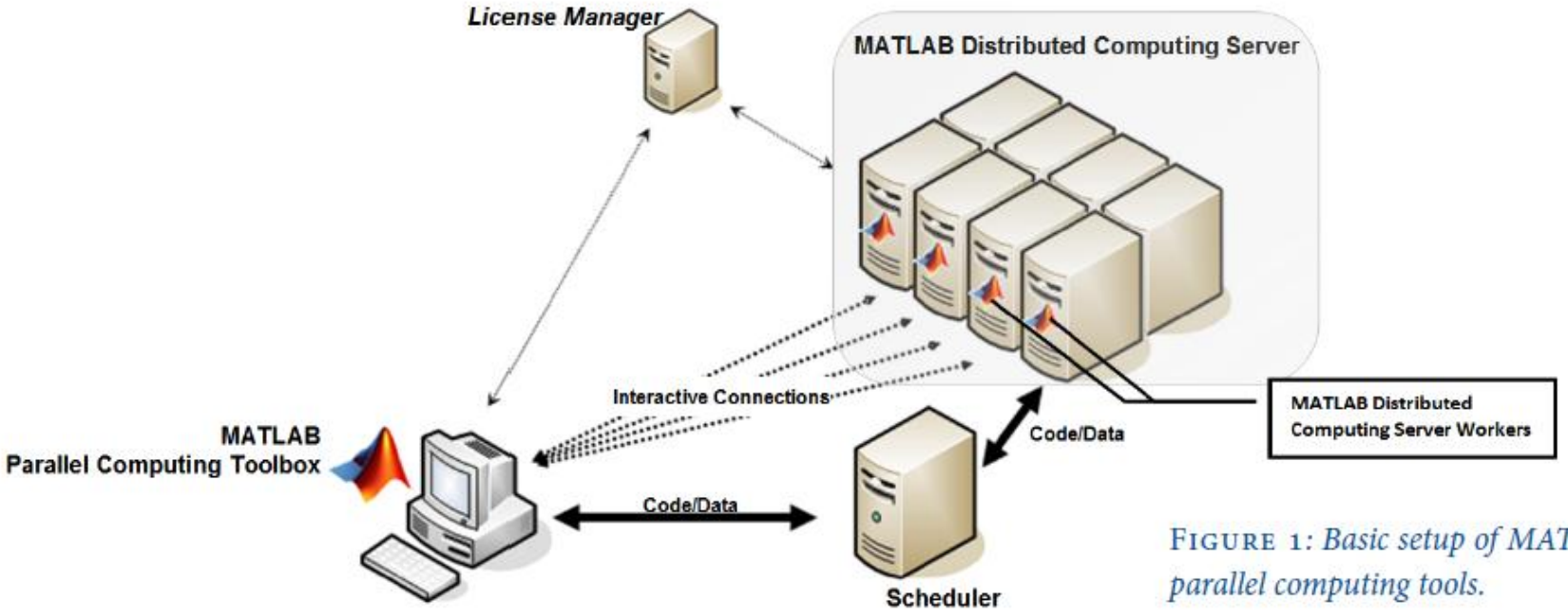


FIGURE 1: *Basic setup of MATLAB parallel computing tools.*

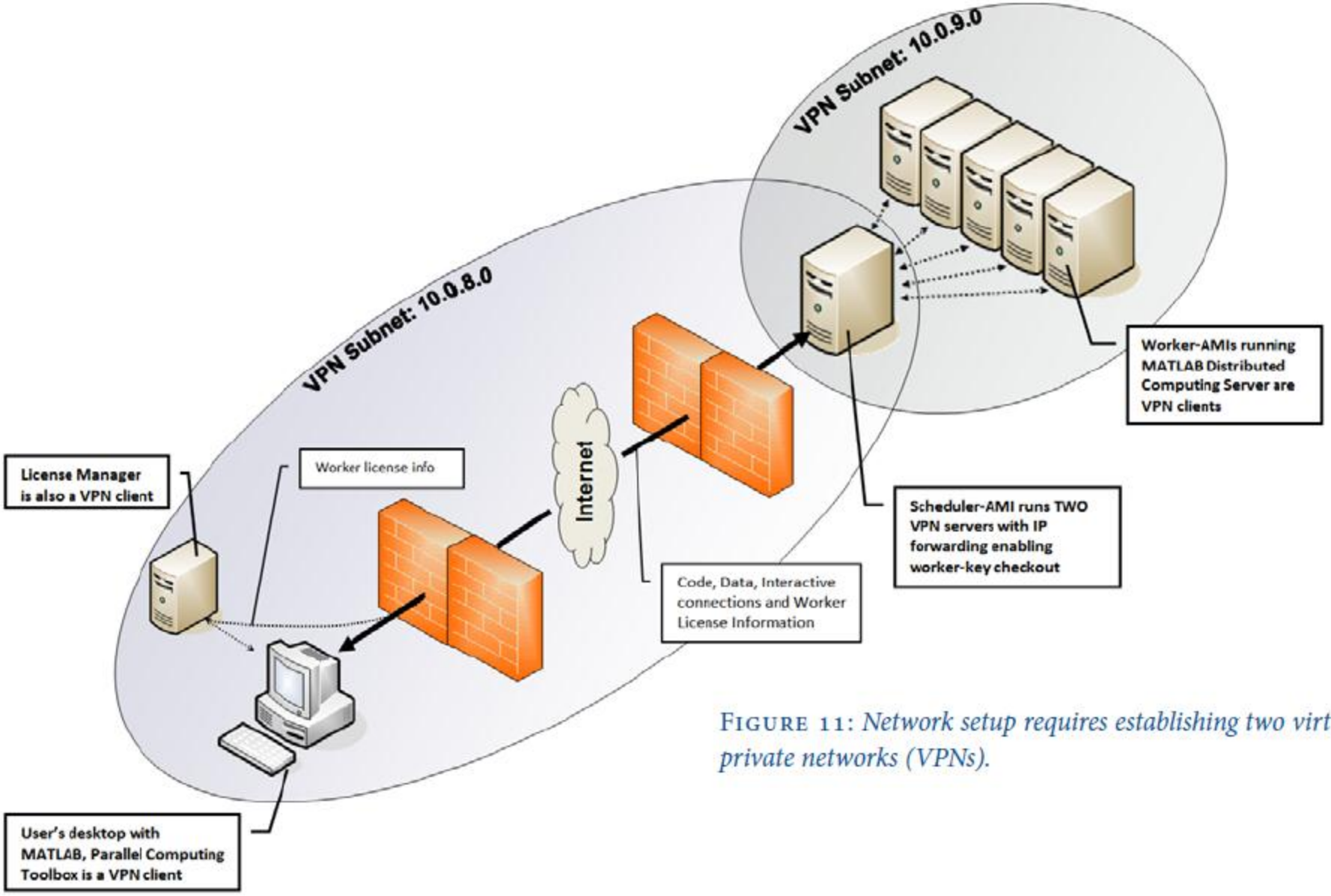


FIGURE 11: *Network setup requires establishing two virtual private networks (VPNs).*

Example -Problem Description

- System of 2^6 (=64) square matrices
  - Each matrix → Sparse, square,2^17 (=131072) dimension
  - Matrix is generated by '`spdiags`' using a 'random' array
- To extract first 100 eigenvectors
  - '`eigs`' function is used
- See handout for the code
- Each matrix calculation is distributed

Example - Serial Matlab

- '`eigen`' is a function
  - Input : (vector of random numbers, dimension of the matrix)
  - Output : eigenvectors

```
n = 2^16;
p = 2^7;
e = rand (n,1,p);

for i = 1 : p
    a = e(:,:,i);
    ans(i) = eigen(a,n);
end
```

Independent calculations

Example-Parallel Matlab

1. Find available distributed computing resources (`findResource` function)

```
nprocs = [ getenv('DMATLAB_NPROCS') ]
np = sscanf( nprocs, '%d' )
mgr_name = [ getenv('JOBMANAGER') ]
mgr_host = [ getenv('JOBMANAGERHOST') ]

jm =findResource('jobmanager','Name',mgr_name,'LookupURL',mgr_host);
```

2. Create distributed job

```
j = createJob(jm,'FileDependencies',{<path>});
```

Path to additional files

Example – Parallel Matlab

3. Create Tasks for each worker

```
n = 2^16;
p = 2^7;
e = rand (n,1,p);

for i = 1 : p
    a = e(:,:,i);
    createTask(j, @eigen, 1, {a,n});
end
```

Same as Serial Code

Creating tasks

Name of the job

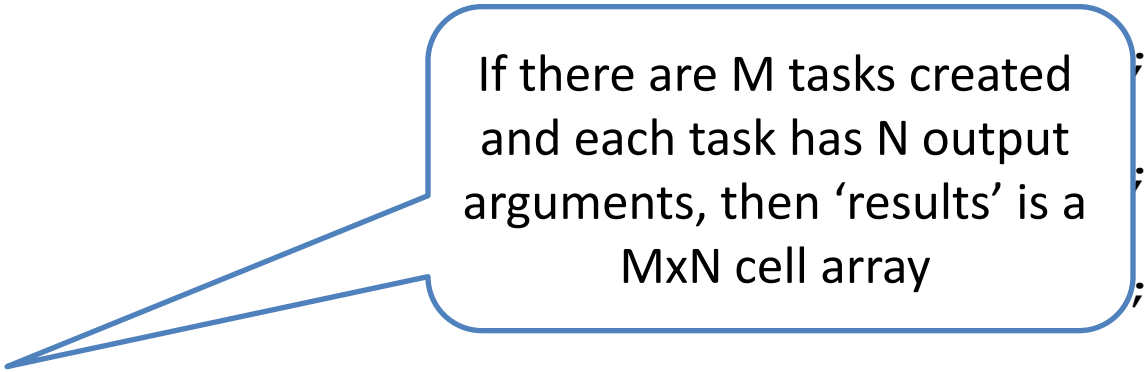
Parallel Task function

Number of output arguments

Input arguments to each task

Example – Parallel Matlab

4. Submit the job and wait for the results




```
results = getAllOutputArguments(j)
```

5. Remove the individual task or parent job object

```
destroy(j);
```