A Global Talent Pool

MATLAB users in education and industry

monster®

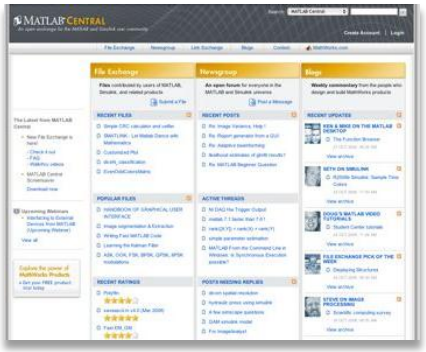
More job postings require MATLAB skills than any other commercial mathematical computing tool

More Facebook fans than any other tech computing software
47,000 and growing
(30% increase since Sept '09)

Google™

19 million hits from search on MATLAB

MATLAB Central



User community and free file exchange – 25,000 downloads/day

40 million annual visits to mathworks.com

MATLAB Seminar

October 10th, 2012

MathWorks Representatives

Daniel
Armyr

Application
Engineer

Ciamé
Andersson

Education
Account Manager

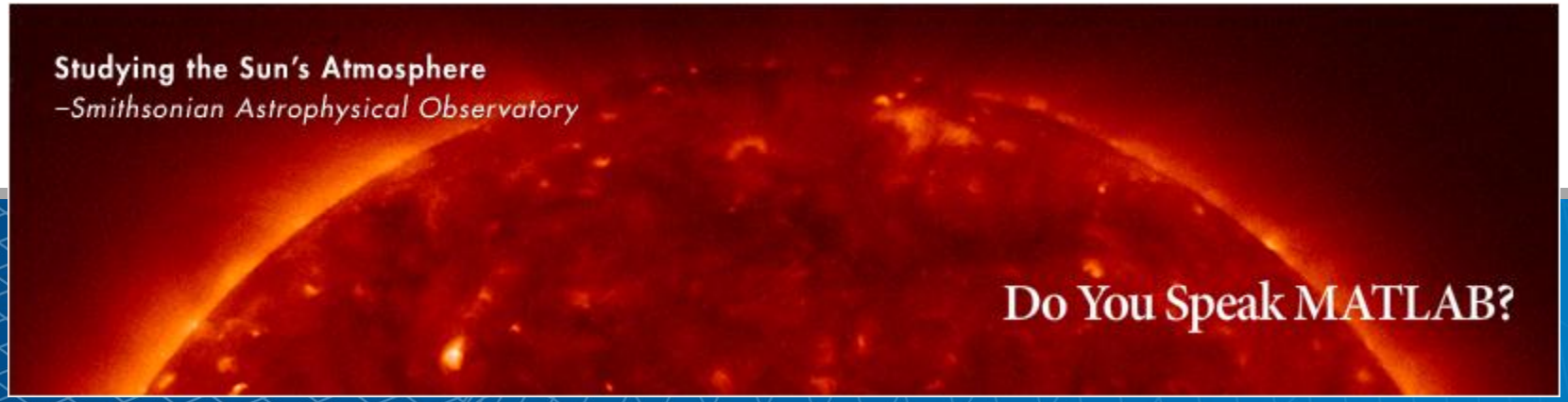
Agenda

- 09:00 - 09:15 Welcome and Introduction
- 09:15 - 10:15 What is MATLAB?
- 10:15 - 10:30 Break
- 10:30 - 11:00 Data Acquisition
- 11:00 - 11:30 Symbolic Computing with MATLAB
- 11:30 - 12:30 Break for lunch
- 12:30 - 13:30 Programming Techniques to Speed up MATLAB
- 13:30 - 13:45 Break
- 13:45 - 14:45 Parallel Computing with MATLAB
- 14:45 - 15:00 Summary and Q&A

What is MATLAB?

or

MATLAB for Excel Users

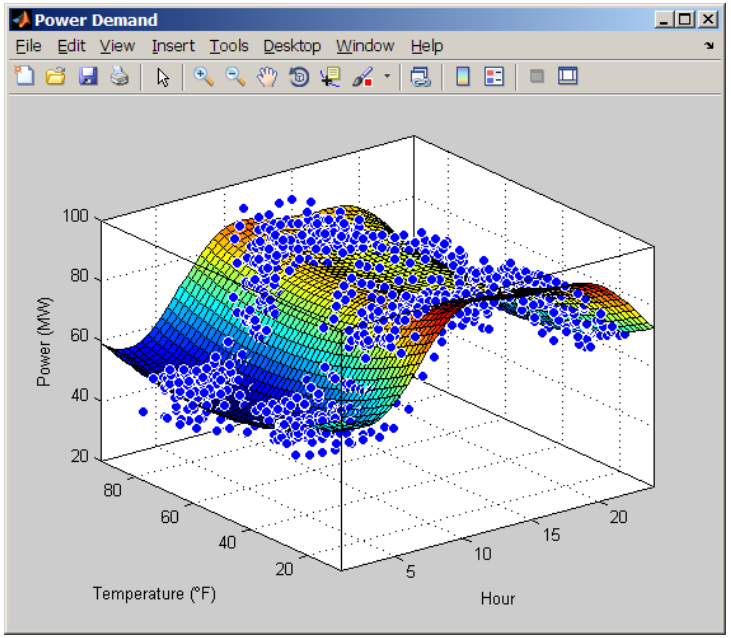


Agenda

- Demo presentation
- Data analysis overview
- Data analysis with MATLAB® and Excel®
- Break

Demo: Estimate Electricity Usage Introduction to MATLAB

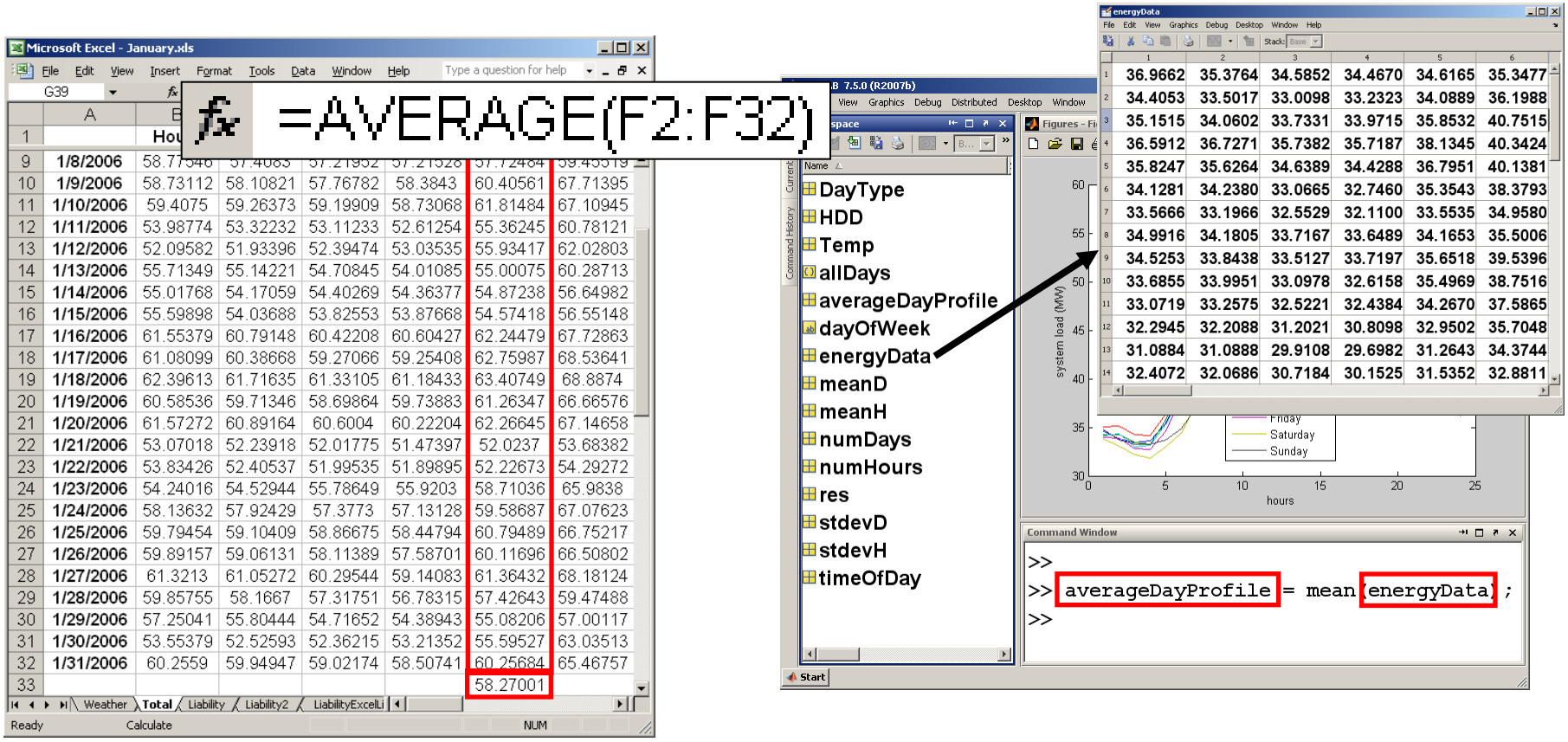
- Goal:
  - Estimate power usage based on time of day and ambient temperature
- Approach:
  - Process historical data of hourly power usage
  - Examine daily, weekly, and monthly trends
  - Develop power usage model
  - Document in a report
  - Link analysis to Excel workbook



Agenda

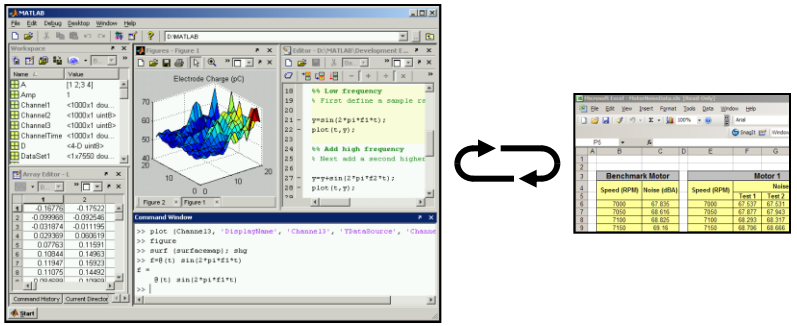
- Demo presentation
-  Data analysis overview
- Data analysis with MATLAB® and Excel®
- Break

Approaches with Excel and MATLAB

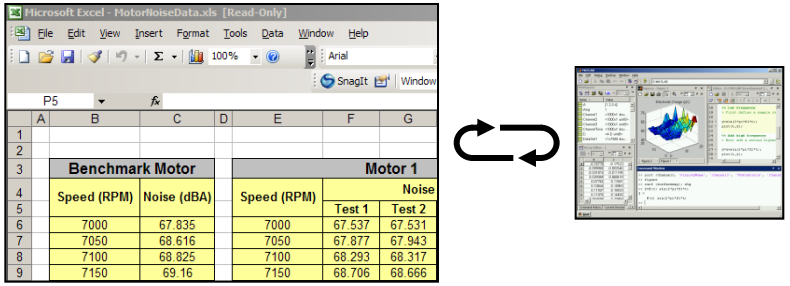


Using MATLAB with Excel

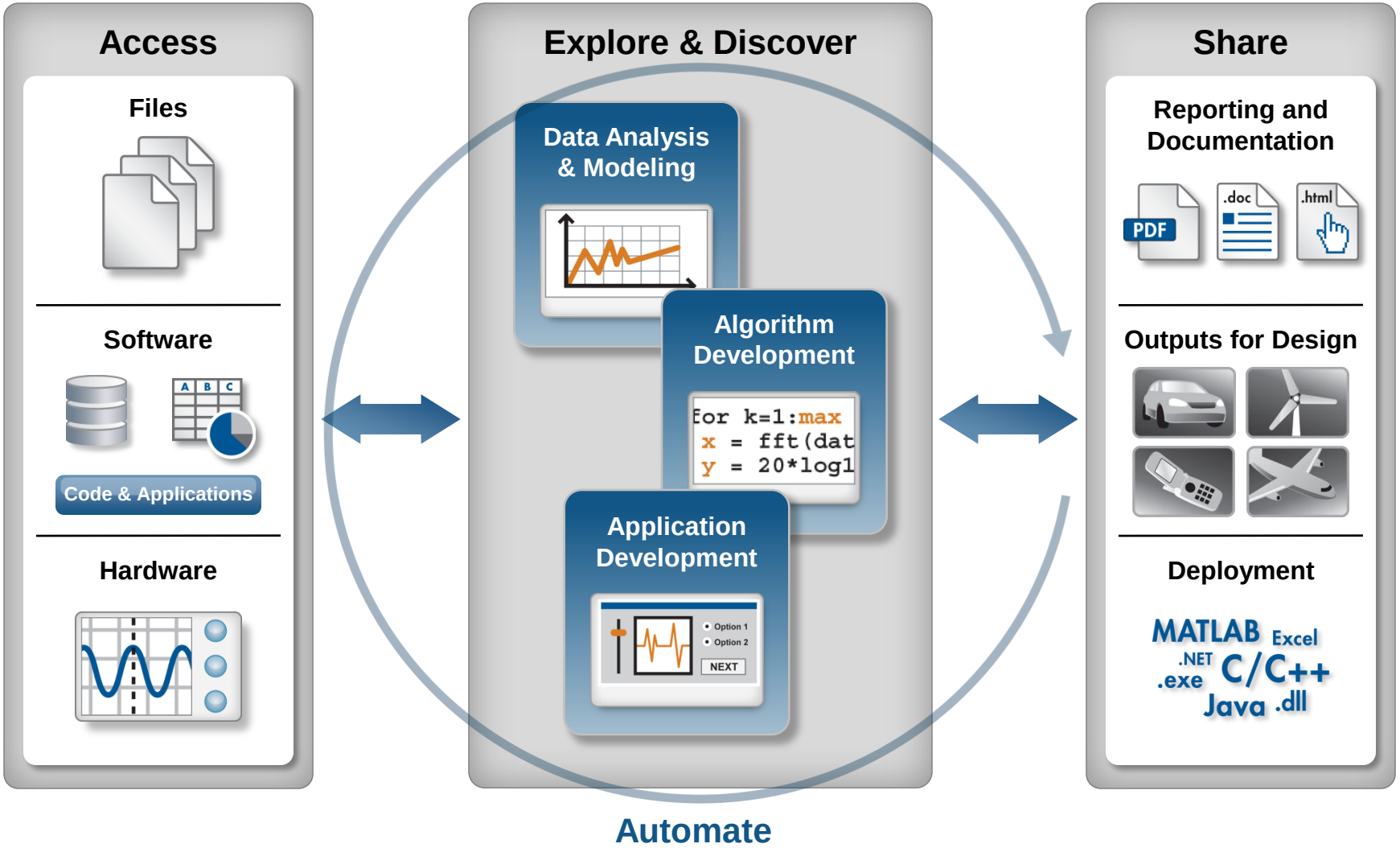
- From MATLAB
  - Read and write Excel data



- From Excel
  - Call MATLAB commands from a cell or Visual Basic script
  - Exchange data with MATLAB
  - Get MATLAB figures

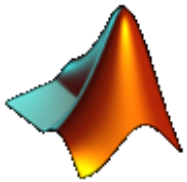


Data Analysis Tasks



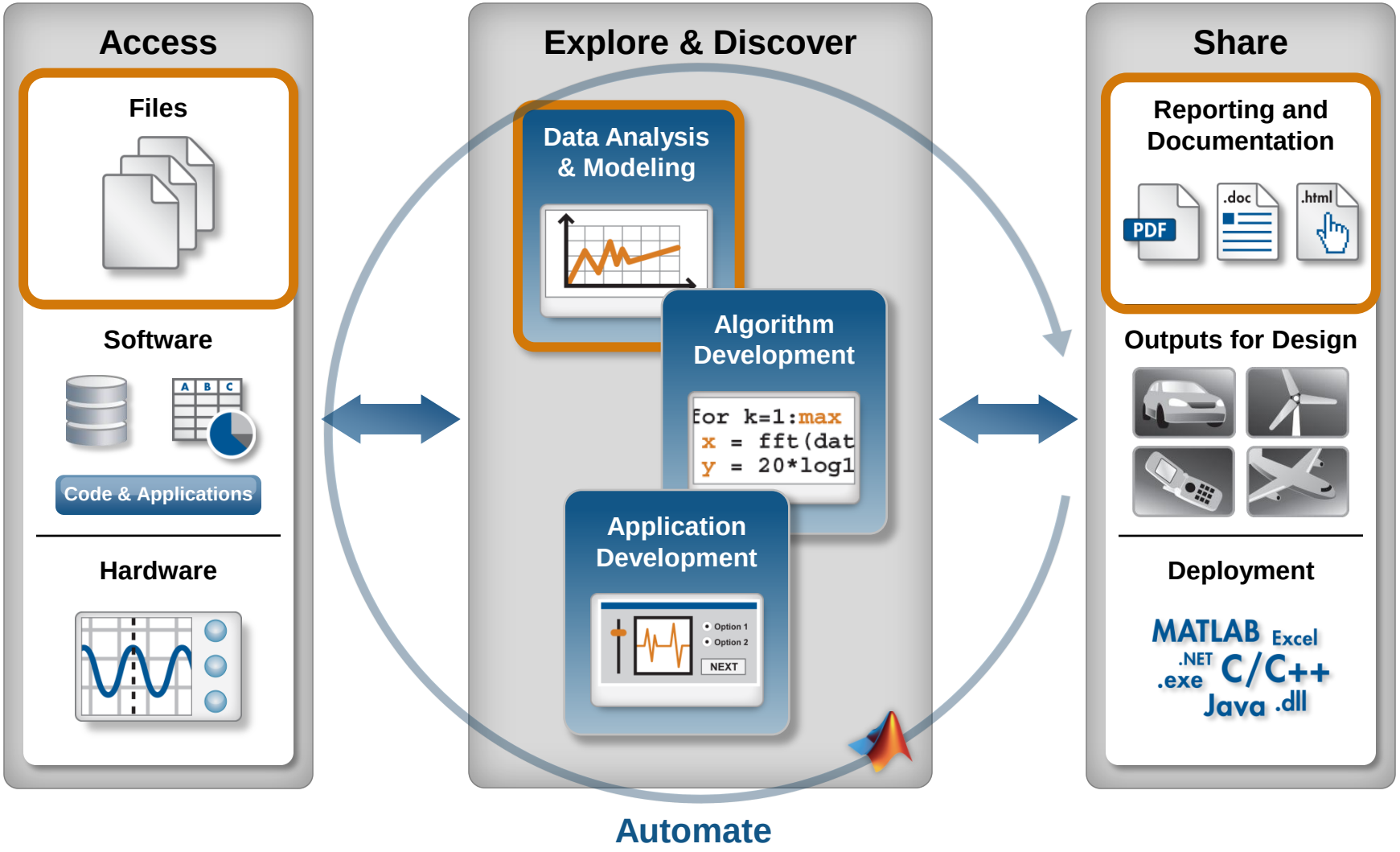
Agenda

- Demo presentation
- Data analysis overview
-  Data analysis with MATLAB® and Excel®
- Break



Demo: Estimate Electricity Usage

Products Used
- MATLAB
- Statistics Toolbox
- Curve Fitting Toolbox



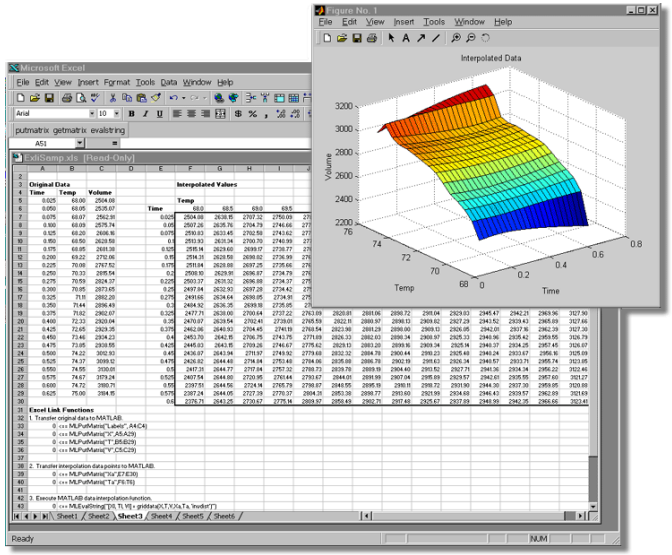
Using MATLAB with Excel

Accessing MATLAB from an Excel Spreadsheet

- Spreadsheet Link EX
  - Provides the functions that connect Excel and MATLAB
  - Connection is “live” – requires no intermediate files or inter-process programming
- Use Spreadsheet Link EX to:
  - Pass data and commands to MATLAB
  - Retrieve data and results back into Excel
- MATLAB is required

Products Used

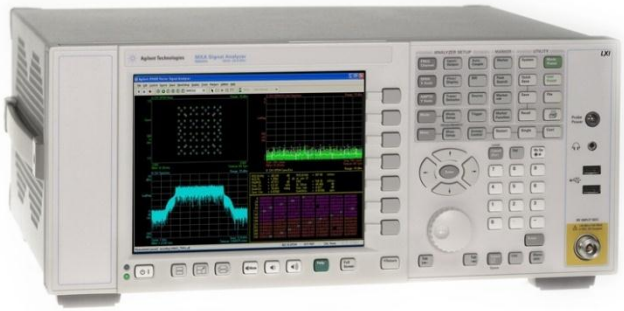
- MATLAB
- Spreadsheet Link EX

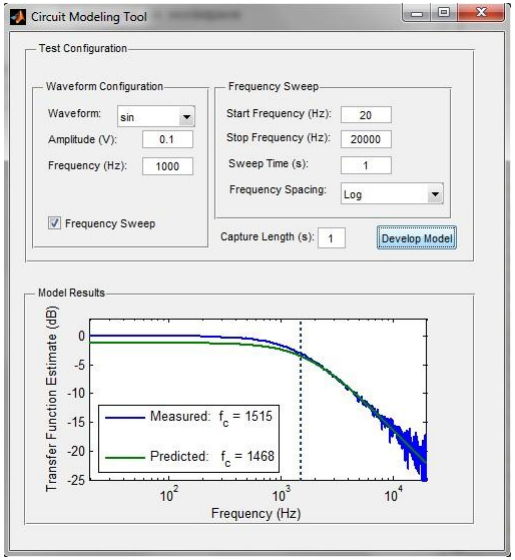


Agenda

- Demo presentation
- Data analysis overview
- Data analysis with MATLAB® and Excel®
- Break

Instrument Control and Data Acquisition with MATLAB

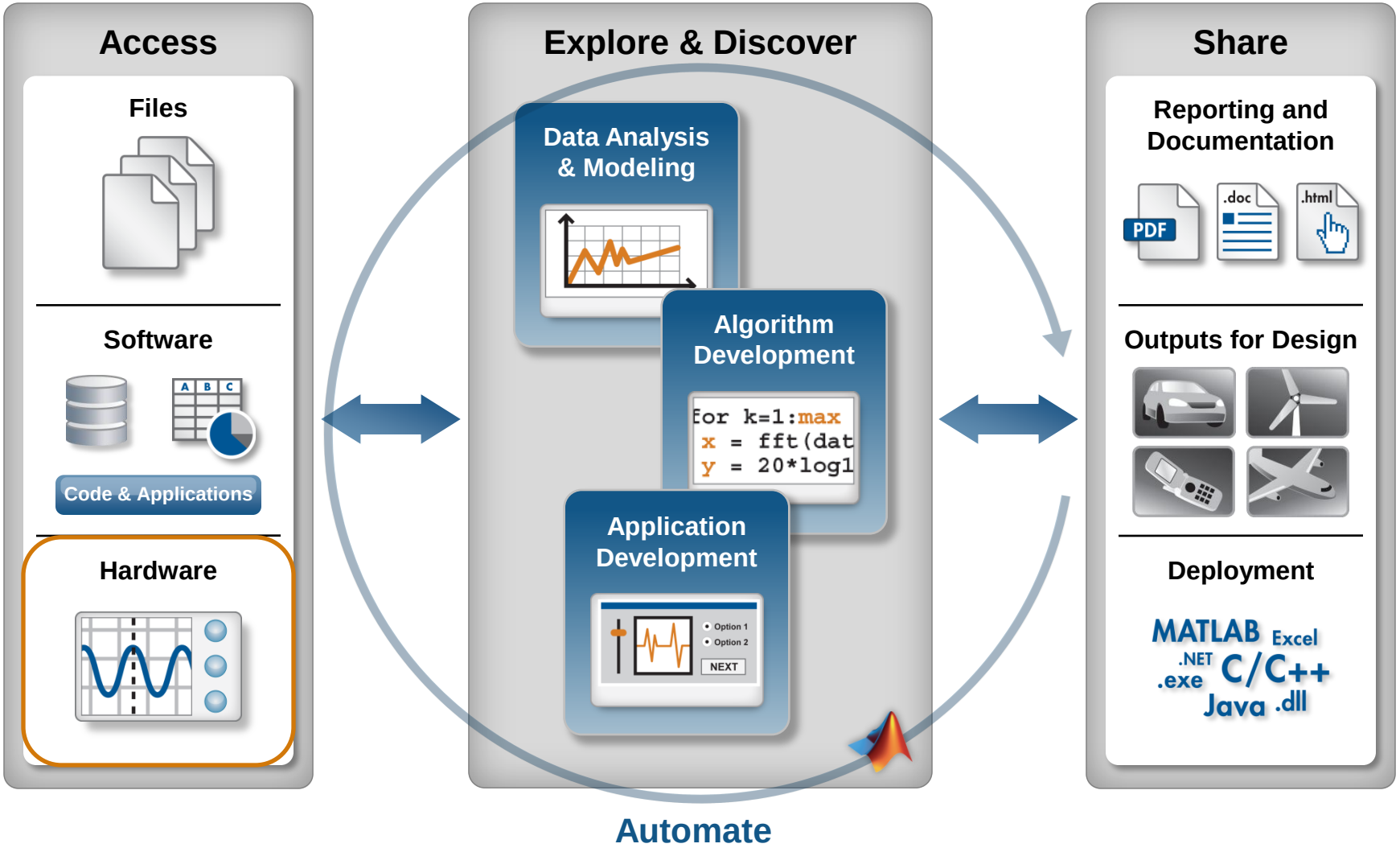




Challenges

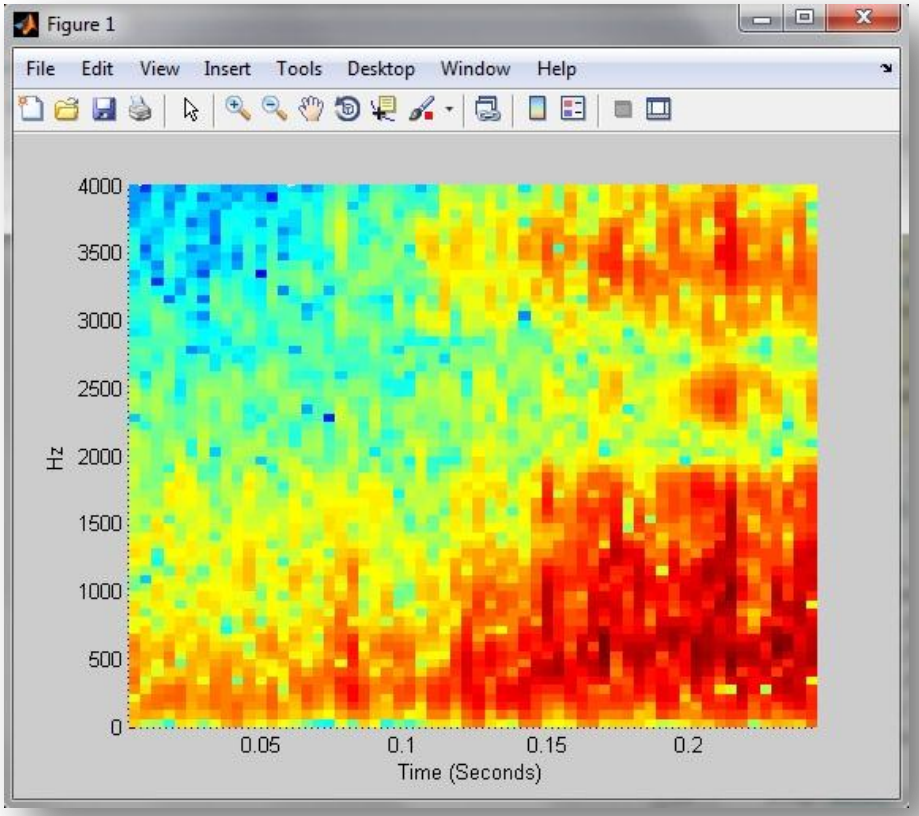
1. Separate tools for acquisition and post-test analysis
2. Limited live analysis capabilities
3. Inconsistent support for multiple hardware vendors

Technical Computing Workflow



Demonstration:

Acquiring and Analyzing Data on-the-fly



MATLAB Connects to Your Hardware Devices

Instrument Control Toolbox

Instruments and RS-232 serial devices






Data Acquisition Toolbox

Plug-in data acquisition devices and sound cards

Image Acquisition Toolbox

Image capture devices





Vehicle Network Toolbox

Vector CAN bus interface devices

OPC Toolbox

OPC DA and HDA servers



Data Acquisition Toolbox:

Supported Hardware

- Agilent
- National Instruments
- Advantech
- CONTEC
- Data Translation
- g.tec
- IOTech
- Keithley
- Measurement Computing (MCC)
- ADLINK
- Ono Sokki
- United Electronic Industries
- VXI Technology
- Open API

- Any PC compatible sound card
- Parallel Port

For a full support listing, visit: www.mathworks.com/products/daq/supportedio.html

Instrument Control Toolbox:
Communication Protocols

GPIB Boards

- Agilent Technologies
- CONTEC
- Keithley
- National Instruments
- Capital Equipment Corporation (CEC)
- IOTech
- Measurement Computing
- ICS Electronics

VISA Interface (includes Serial, GPIB, VXI, GPIB-VXI, TCP/IP, USB)

- Agilent
- Rohde & Schwarz
- National Instruments
- Tektronix

Network Protocols (TCP/IP and UDP)

Serial Port (core functionality in MATLAB)

- RS-232, RS-422, RS-485
- Expanded capability with Instrument Control Toolbox

For a full listing, visit: www.mathworks.com/products/instrument/hardware/

Image Acquisition Toolbox:
Supported Hardware and Communication Protocols

Analog and Camera Link

- BitFlow
- Imperx
- National Instruments
- Data Translation
- Matrox Imaging
- Teledyne DALSA

DCAM (IIDC 1394)

GenTL

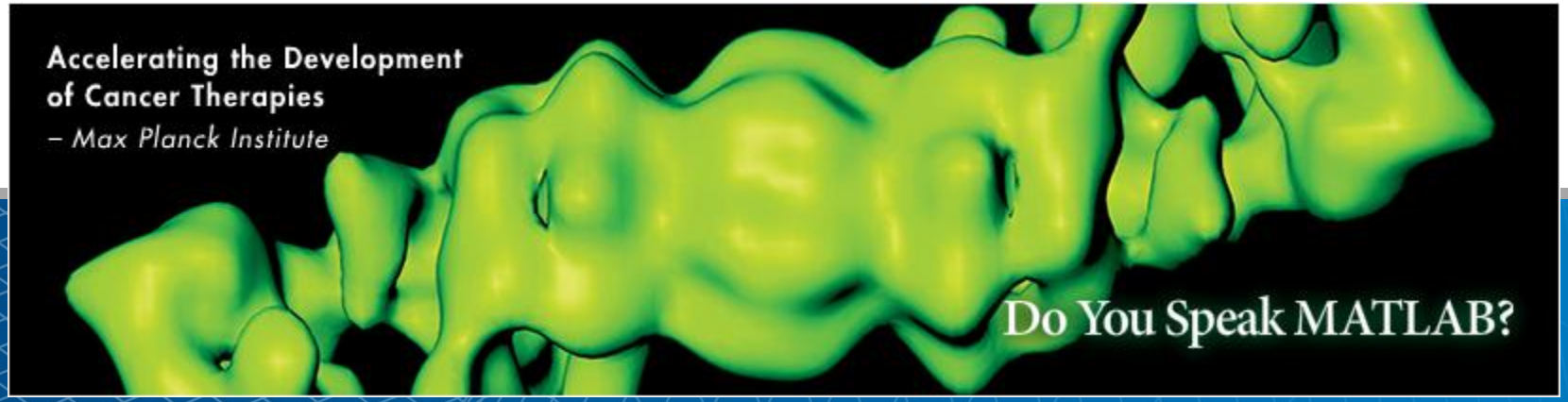
GigE Vision

Generic Interface (Webcams)

- Video 4 Linux 2
- Video for Windows
- QuickTime

For a full listing, visit: www.mathworks.com/products/instrument/hardware/

MATLAB at Uppsala University



Uppsala University – License Setup

- TAH Student 

- TAH Campus 

- MDCS

License Access and Administration

http://www.uu.se/goto/matlab

http://www.uu.se/goto/studentmatlab

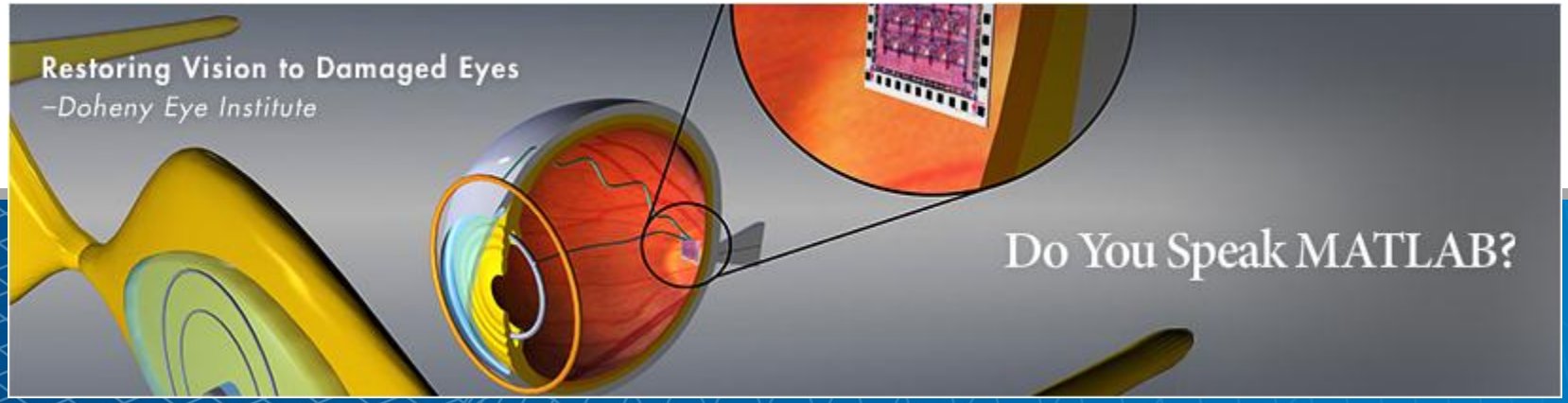
Mikael Österberg

✉ mikael.osterberg@angstrom.uu.se
☎ 018-471 7989

MathWorks Techical Support & Customer Service

- Tel: 08-5051-6900
  - Monday-Friday
    08:30 - 17:00
- www.mathworks.com

Learning more about MATLAB

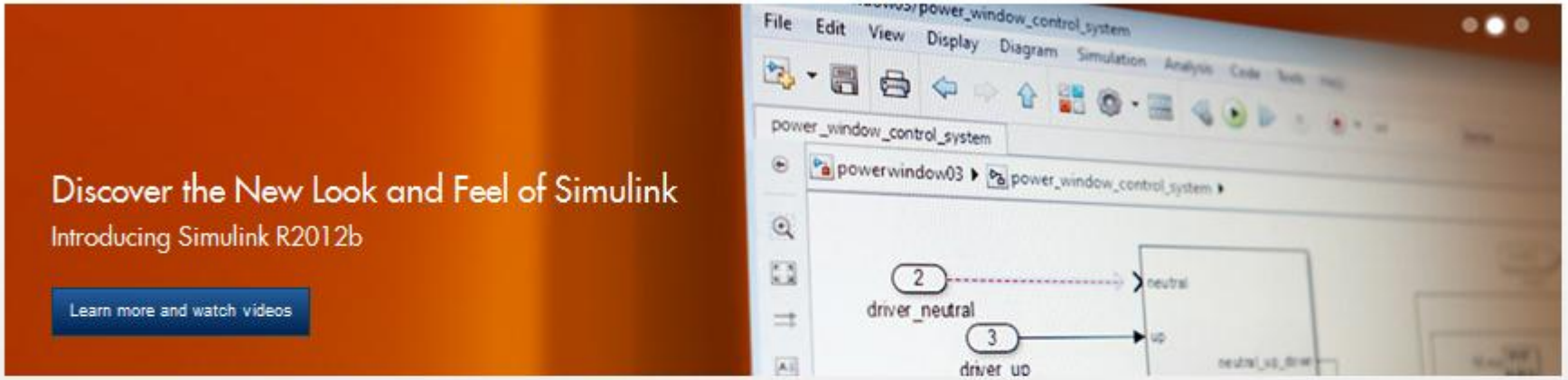


MathWorks | Accelerating the pace of engineering and science

Nordic | Contact Us | Store

Create Account | Log In

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



Products | Events | Training

Live

9 okt	Model-Based Design for Hybrid Electric Powertrain... (seminar)
17 okt	Computational Finance and Application Deployment with... (seminar)
17 okt	Verifying Floating-Point IP Cores on FPGAs with MATLAB... (webinar)
23 okt	4G LTE System Design with MATLAB (webinar)
24 okt	Integrating PK/PD and Mechanistic Modeling with... (webinar)

On Demand

- Introduction to Simulink for Control Design
- Model-Based Design for Automation Systems
- Modeling and Simulating Phased Array Signal Processing Systems
- MATLAB on the Web for the Classroom
- Acquiring Data from Sensors and Instruments Using MATLAB

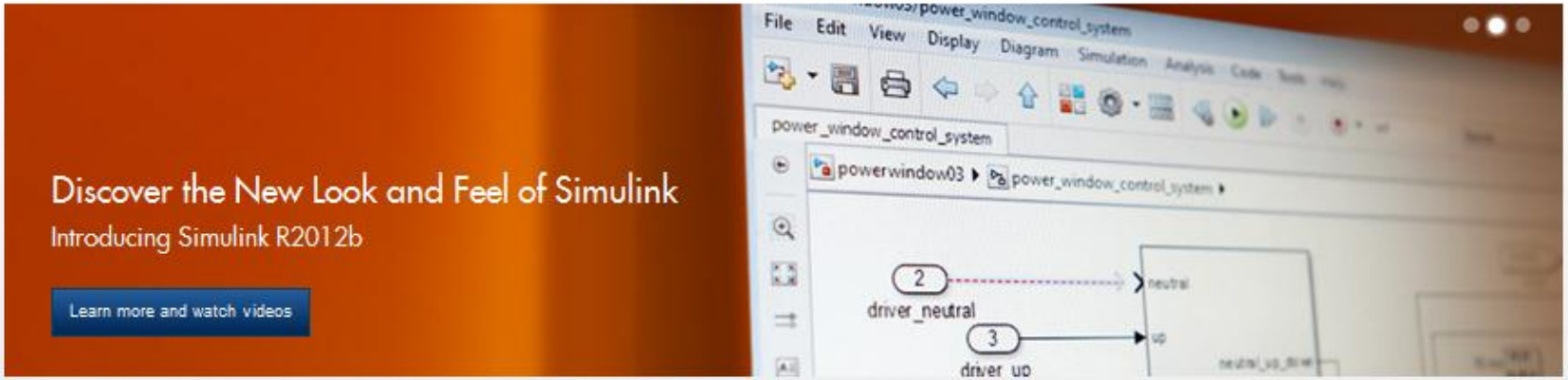
Explore all events: webinars, seminars, tradeshows, and conferences

MathWorks | Accelerating the pace of engineering and science

Nordic | Contact Us | Store

Create Account | Log In

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



Products | Events | Training

Oct 22nd

Online

03 okt	Stateflow for Logic Driven System Modeling
10 okt	Simulink for System and Algorithm Modeling
23 okt	Polyspace for Code Verification
30 okt	Signal Processing with MATLAB
05 nov	MATLAB Fundamentals

Classroom

Stockholm (City)	MATLAB Fundamentals
Stockholm (City)	MATLAB Programming Techniques
Stockholm (City)	MATLAB Based Optimization Techniques
Stockholm (City)	Simulink for System and Algorithm Modeling
Stockholm (City)	MATLAB Fundamentals

New Self-Paced Training

MATLAB Fundamentals

» See course details

Training Course Catalog

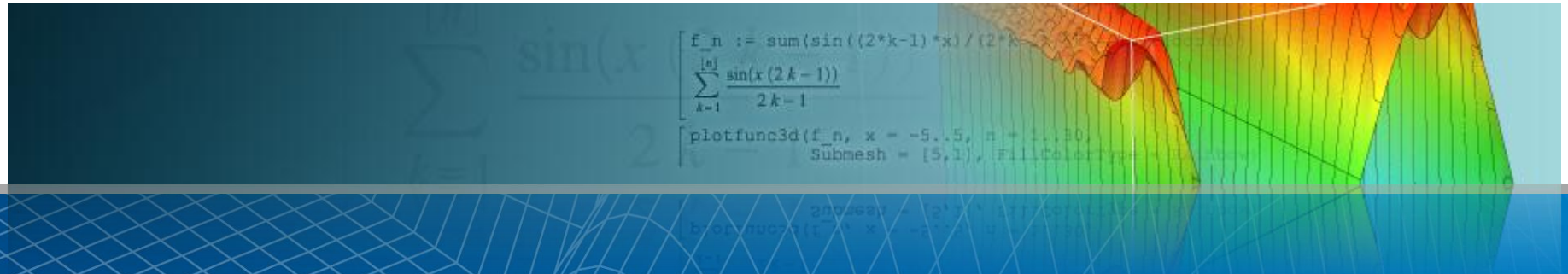
» Download now

Explore all local classes, e-learning classes, or other training options

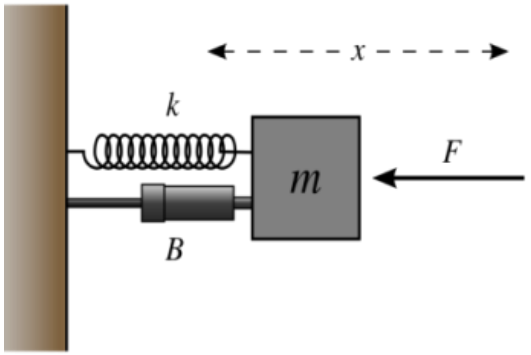
Ciamé Andersson
Education Account Manager

ciame.andersson@mathworks.com ✉
08-505 169 61 ☎

Symbolic Computing with MATLAB



Overview of Symbolic Computing



Governing equation:

$$F = mx''(t) + Bx'(t) + kx(t)$$

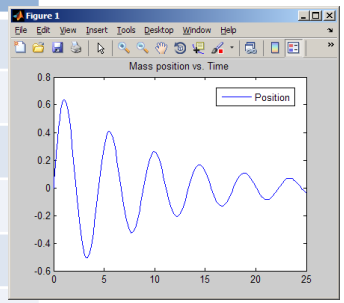
Initial conditions:

$$x(0) = 0 \qquad x'(0) = 0$$

Numeric:

Approximate solution in vector form

time	x(t)
0	0
0.1	0.099
0.2	0.193
0.3	0.283
0.4	0.364
0.5	0.437



Symbolic:

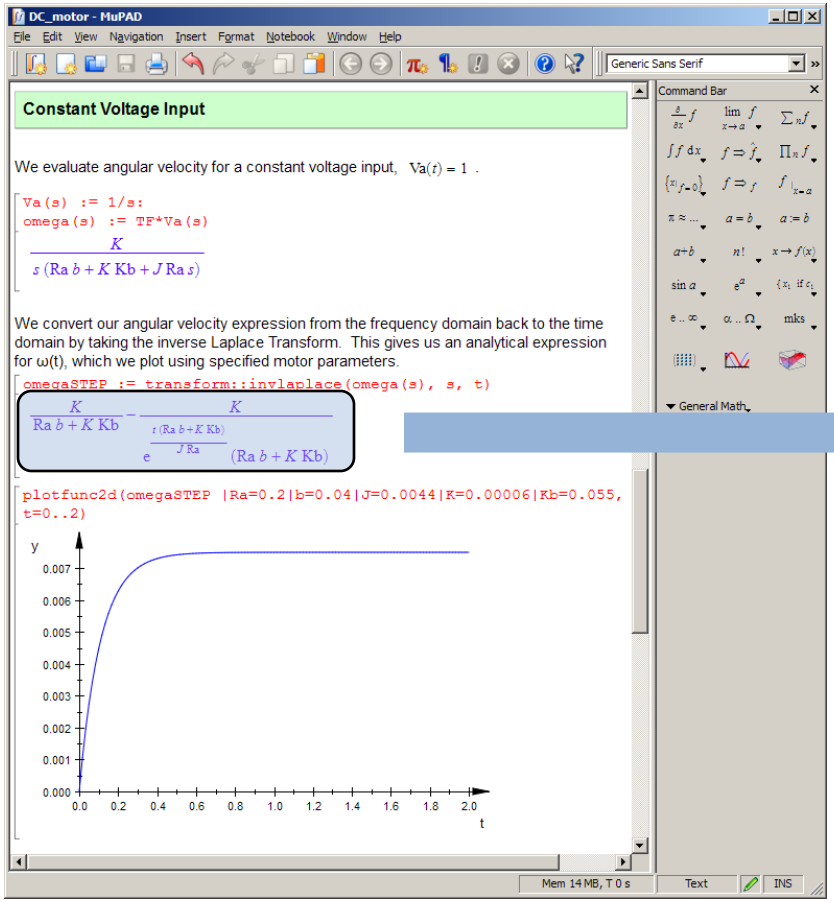
Exact solution in form of analytical expression

$$x(t) = \frac{m}{e^{\frac{t\left(B-\sqrt{B^2-4km}\right)}{2m}}\sqrt{B^2-4km}} - \frac{m}{e^{\frac{t\left(B+\sqrt{B^2-4km}\right)}{2m}}\sqrt{B^2-4km}}$$

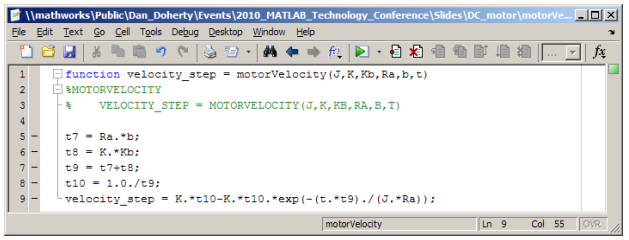
Benefits

1. Insight into how parameters affect solution
2. Often more efficient than numeric implementation

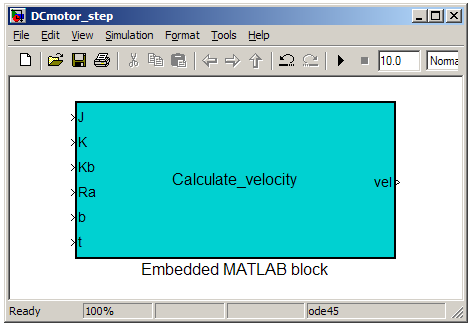
Integrating Symbolic Results with MATLAB and Simulink



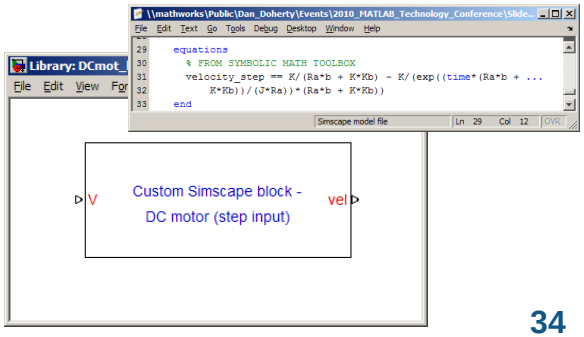
with MATLAB



with Simulink



with Simscape

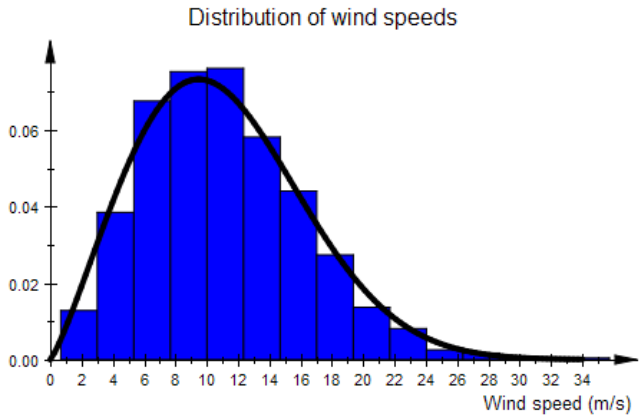


Product Demonstration

Modeling power generated by a wind turbine

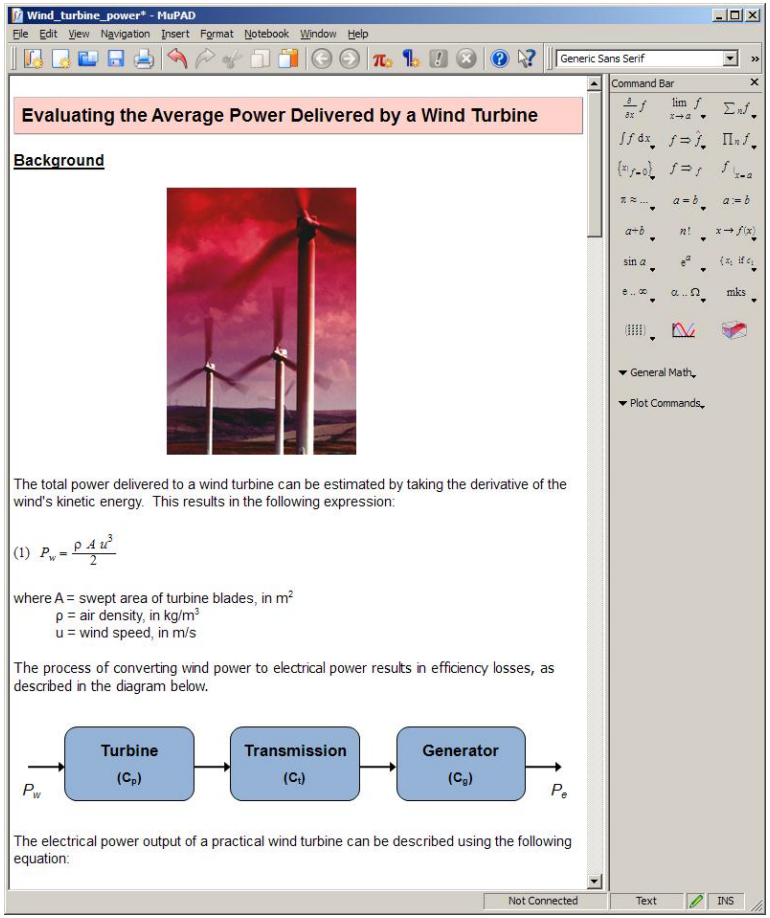
Objective

- Determine wind turbine design that produces optimal power at a given site



Problem-solving steps

- Derive analytical model for average power produced by wind turbine
- Simulate model in MATLAB for different configurations
- Determine wind turbine that optimizes power at given site



Summary

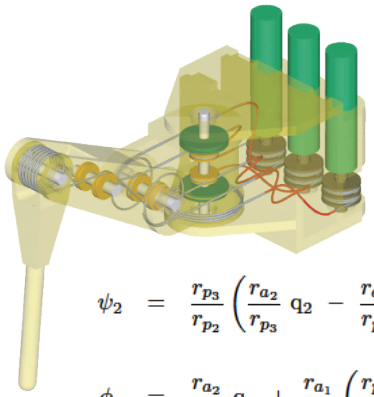
- Develop algorithms and models with symbolic math
  - Gain efficiency
  - Gain insight through analytical solutions
  - Perform interactive calculation and documentation via the notebook interface
  - Integrate analytical results in MATLAB

$$C = S\Phi(d_1) - Ke^{-rT}\Phi(d_2)$$

where

$$d_1 = \frac{\ln(S/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}$$

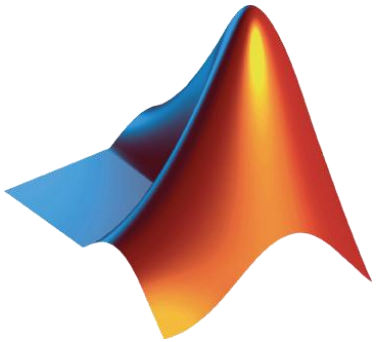
$$d_2 = \frac{\ln(S/K) + (r - \sigma^2/2)T}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T}.$$



$$\psi_2 = \frac{r_{p_3}}{r_{p_2}}\left(\frac{r_{a_2}}{r_{p_3}}\,\mathrm{q}_2 - \frac{r_{a_1}}{r_{p_1}}\,\mathrm{q}_1\right)$$

$$\phi_2 = \frac{r_{a_2}}{r_{p_2}}\,\mathrm{q}_2 + \frac{r_{a_1}}{r_{p_1}}\left(\frac{r_{p_2} - r_{p_3}}{r_{p_2}}\right)\mathrm{q}_1$$

Speeding up MATLAB Applications



Agenda

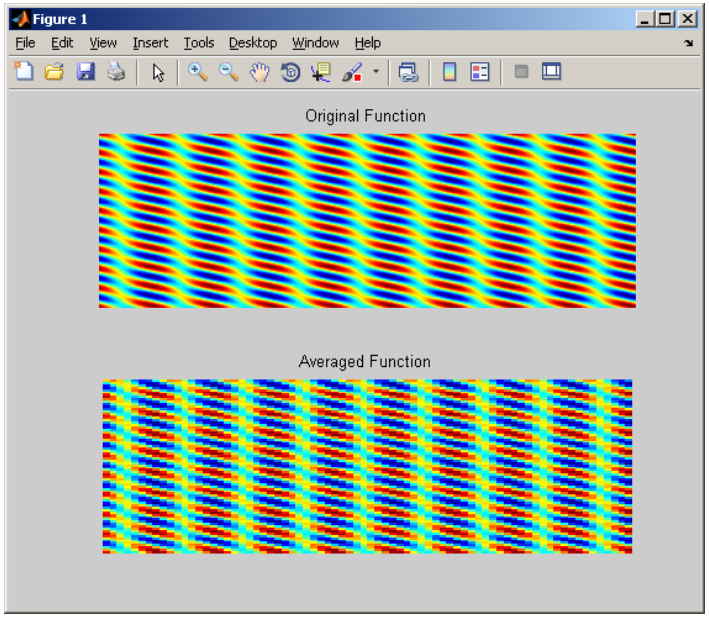
Leveraging the power of vector and matrix operations in MATLAB®

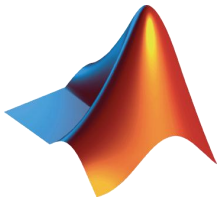
- Demonstration: Preallocation and vectorization

- How does MATLAB® store and provide access to its' variables?
  - Demonstration: Row- vs. Column-major indexing
  - Demonstration: Logical indexing
- Addressing bottlenecks
  - Demonstration: Profiling code in MATLAB®
- Memory considerations
  - Demonstration: Copy-on-write
- Summary and Q&A

Example: Block Processing Images

- Evaluate function at grid points
- Reevaluate function over larger blocks
- Compare the results
- Evaluate code performance





Summary of Example

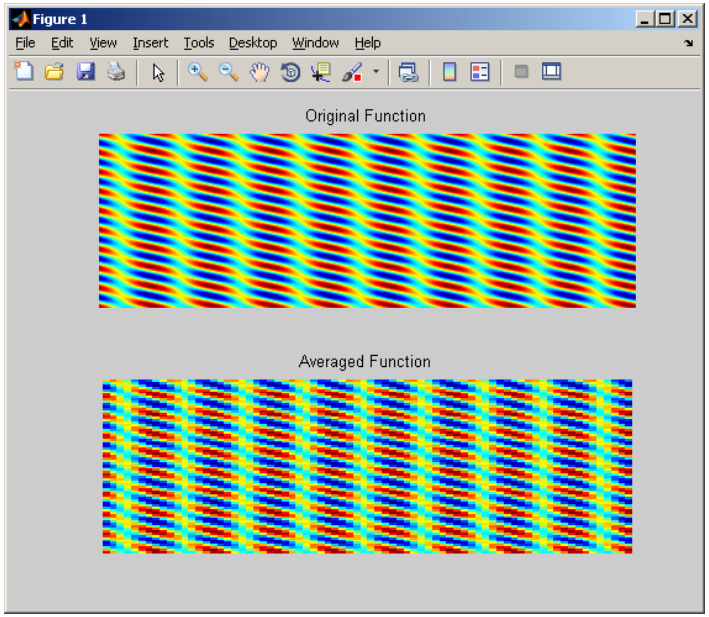
- Used built-in timing functions

```
>> tic
>> toc
```

- Used Code Analyzer to find suboptimal code

- Preallocated arrays

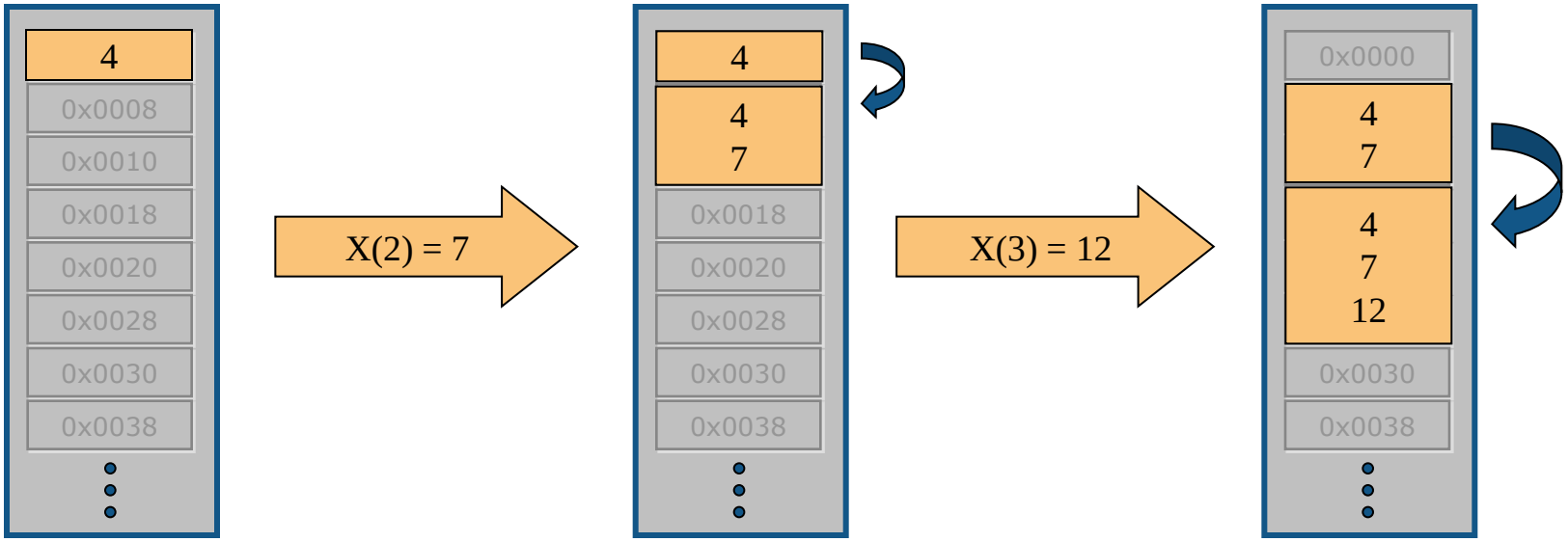
- Vectorized code



Effect of Not Preallocating Memory

```
>> x = 4
>> x(2) = 7
>> x(3) = 12
```

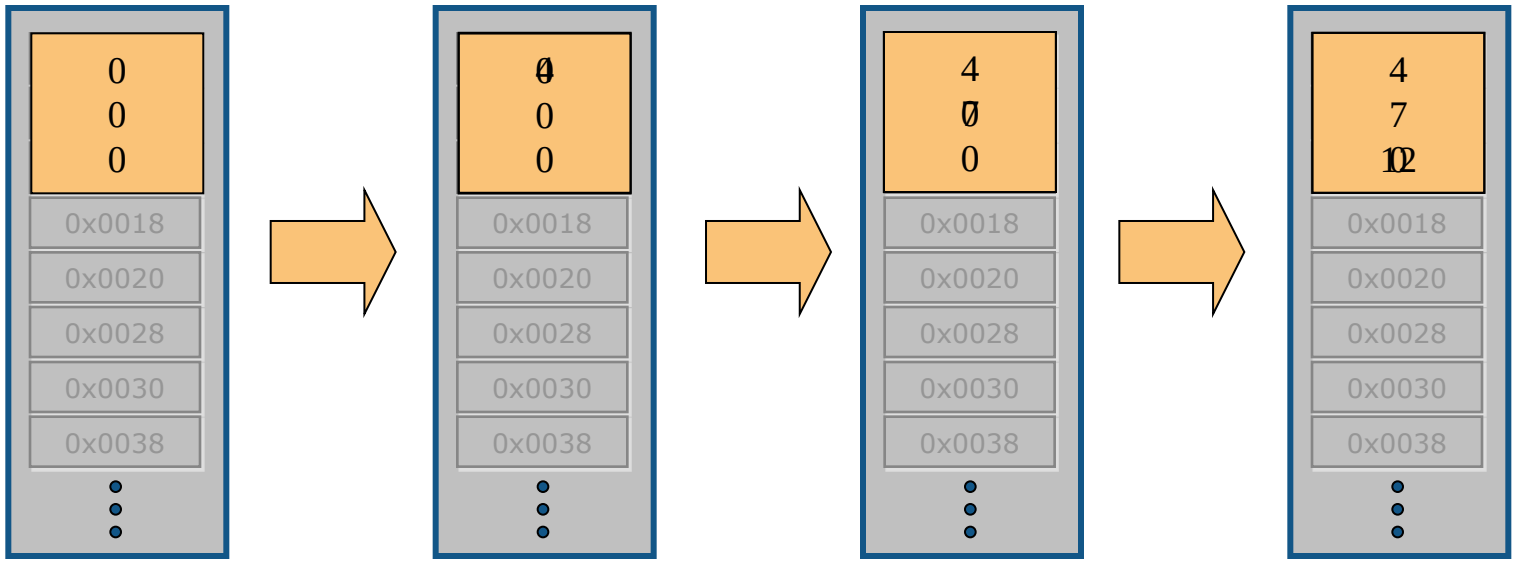
Resizing Arrays is Expensive



Benefit of Preallocation

```
>> x = zeros(3,1)
>> x(1) = 4
>> x(2) = 7
>> x(3) = 12
```

Reduced Memory Operations



Agenda

- Leveraging the power of vector and matrix operations in MATLAB®
  - Demonstration: Preallocation and vectorization
- 
  How does MATLAB® store and provide access to its' variables?
  - Demonstration: Row- vs. Column-major indexing
  - Demonstration: Logical indexing
- Addressing bottlenecks
  - Demonstration: Profiling code in MATLAB®
- Memory considerations
  - Demonstration: Copy-on-write
- Summary and Q&A

Data Storage of MATLAB Arrays

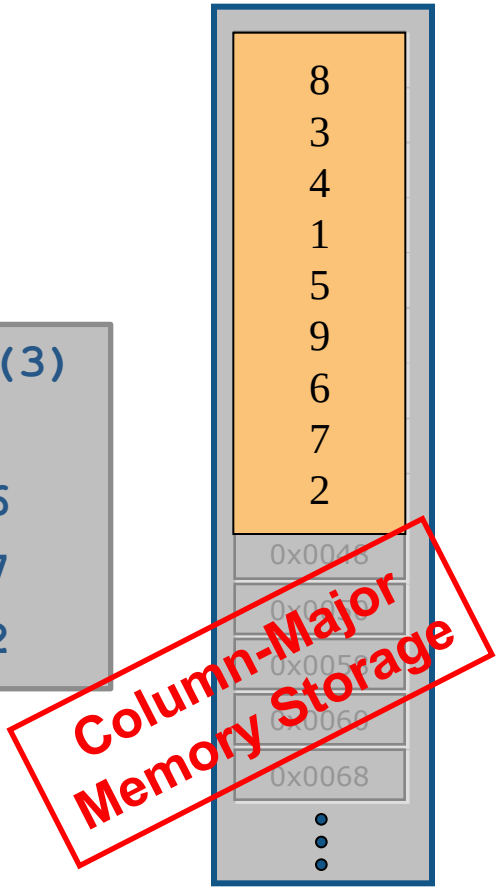
What is the fastest way to process MATLAB matrices with `for` loops? (i.e. de-vectorize)

a) Down the columns

b) Along the rows

c) Doesn't matter

```
>> x = magic(3)
x =
     8     1     6
     3     5     7
     4     9     2
```



See the June 2007 article in "The MathWorks News and Notes":
http://www.mathworks.com/company/newsletters/news_notes/june07/patterns.html

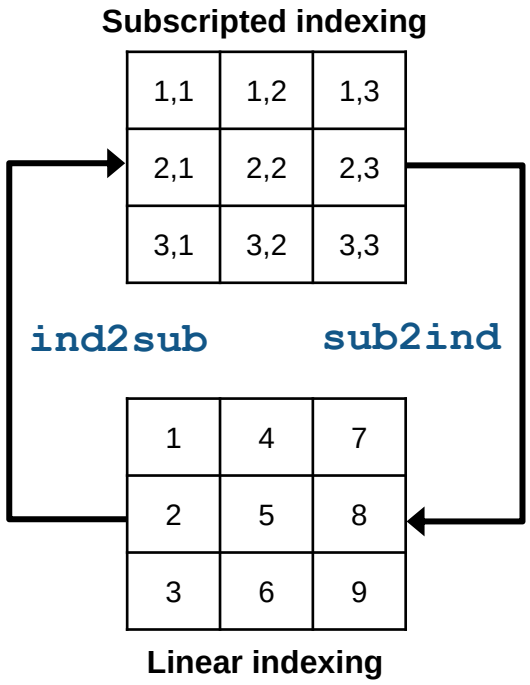
Speed and Memory Usage

- Balance vectorization and memory usage
  - Use **`bsxfun`** instead of functions such as **`repmat`**
  - Reduce size of arrays to smaller blocks for block processing

- Consider using sparse matrices
  - *Less Memory*: Store only nonzero elements and their indices
  - *Faster*: Eliminate operations on zero elements
  - Blog Post - Creating Sparse Finite Element Matrices http://blogs.mathworks.com/loren/2007/03/01/creating-sparse-finite-element-matrices-in-matlab/

Indexing into MATLAB Arrays

- Subscripted
  - Access elements by rows and columns
- Linear
  - Access elements with a single number
- Logical
  - Access elements with logical operations or mask



MATLAB Underlying Technologies

- Commercial libraries
  - BLAS: Basic Linear Algebra Subroutines (multithreaded)
  - LAPACK: Linear Algebra Package
  - etc.

BLAS and LAPACK require contiguous arrays

- JIT/Accelerator
  - Improves looping
  - Generates on-the-fly multithreaded code
  - Continually improving

Other Best Practices

- Minimize dynamically changing path

```
>> addpath(…)
>> fullfile(…)
```

instead of `cd(…)`

- Use the functional load syntax

```
>> x = load('myvars.mat')
x =
      a: 5
      b: 'hello'
```

instead of `load('myvars.mat')`

- Minimize changing variable class

```
>> x = 1;
>> xnew = 'hello';
```

instead of `x = 'hello';`

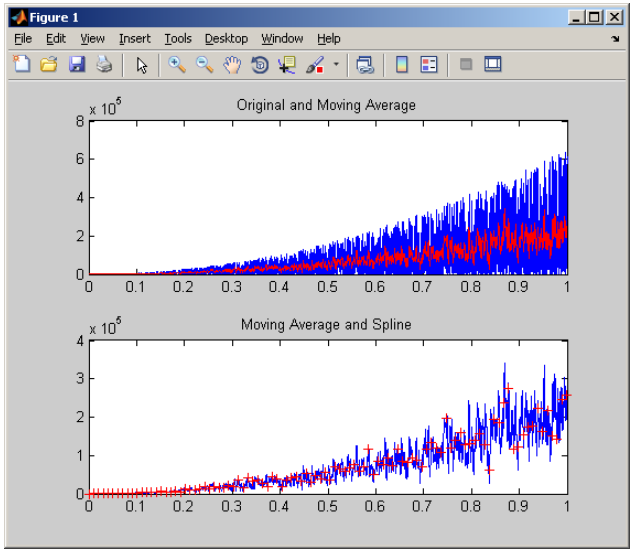
Agenda

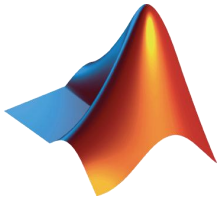
- Leveraging the power of vector and matrix operations in MATLAB®
  - Demonstration: Preallocation and vectorization
- How does MATLAB® store and provide access to its' variables?
  - Demonstration: Row- vs. Column-major indexing
  - Demonstration: Logical indexing

- Addressing bottlenecks
  - Demonstration: Profiling code in MATLAB®
- Memory considerations
  - Demonstration: Copy-on-write
- Summary and Q&A

Example: Fitting Data

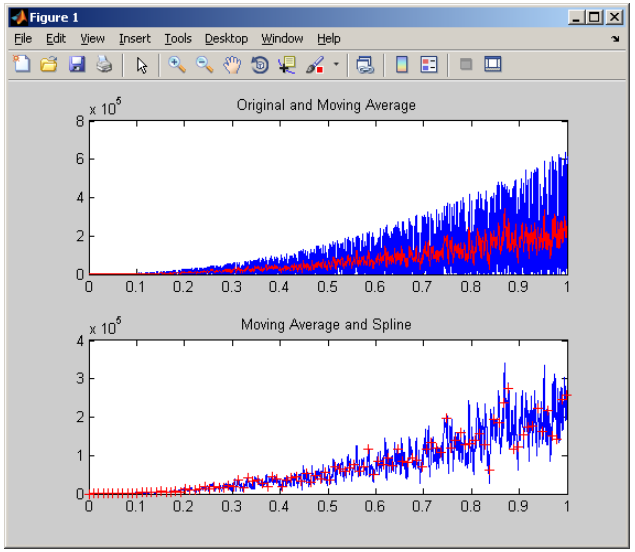
- Load data from multiple files
- Extract a specific test
- Fit a spline to the data
- Write results to Microsoft Excel





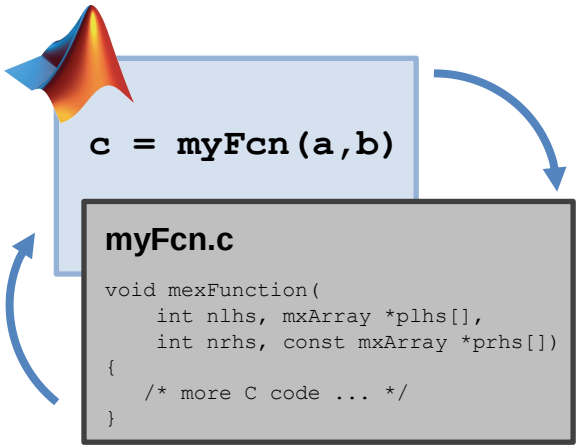
Summary of Example

- Used profiler to analyze code
- Targeted significant bottlenecks
- Reduced file I/O
- Reused figure



Acceleration using MEX (MATLAB Executable)

- Call C or Fortran code directly from MATLAB
  - Integrate existing code using MEX API
  - Auto-generate C-based MEX files from MATLAB code using MATLAB Coder



- Speed-up factor will vary
  - May see speedup for state-based for-loops
  - May not see a speedup when MATLAB code is
    - Using multithreaded computations
    - Using optimized libraries (BLAS, FFTW, etc.)

Agenda

- Leveraging the power of vector and matrix operations in MATLAB®
  - Demonstration: Preallocation and vectorization
- How does MATLAB® store and provide access to its' variables?
  - Demonstration: Row- vs. Column-major indexing
  - Demonstration: Logical indexing
- Addressing bottlenecks
  - Demonstration: Profiling code in MATLAB®

- Memory considerations
  - Demonstration: Copy-on-write
- Summary and Q&A

MATLAB and Memory

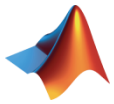
Lazy copying (copy-on-write):

```
C{1} = rand(3000,3000); %72Mb
C{2} = rand(3000,3000); %72Mb

C_new = C;  %0 Mb

C{1}(1,1) = 2;  %72Mb
```

Arrays are stored in contiguous memory



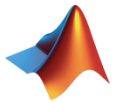
In-place Optimizations

When does MATLAB do calculations "**in-place**"?

```
function y = myfunc(x)
y = sin(2*x.^2+3*x+4);

function x = myfuncInPlace(x)
x = sin(2*x.^2+3*x+4);

function inplaceOptimization
x = randn(3e7,1);
y = myfunc(x);                  % Copy
x = myfuncInPlace(x);           % In-place
x = myfunc(x);                  % Copy
```



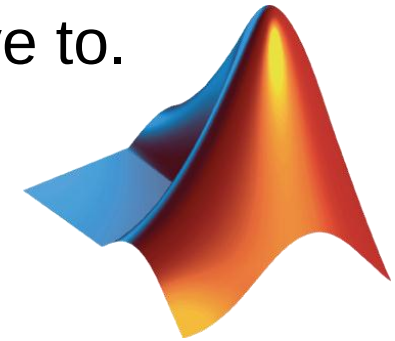
Agenda

- Leveraging the power of vector and matrix operations in MATLAB®
  - Demonstration: Preallocation and vectorization
- How does MATLAB® store and provide access to its' variables?
  - Demonstration: Row- vs. Column-major indexing
  - Demonstration: Logical indexing
- Addressing bottlenecks
  - Demonstration: Profiling code in MATLAB®
- Memory considerations
  - Demonstration: Copy-on-write

Summary and Q&A

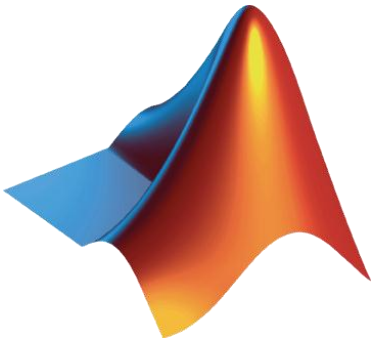
Summary

- Consider performance benefit of vector and matrix operations in MATLAB
- Preallocate memory and look at your indexing.
- Analyze your code for bottlenecks and address most critical items
- Never write hard-to-read code unless you have to.



Summary (continued…)

- MATLAB passes data to functions by value using lazy copy.



Sample of Other Performance Resources

- MATLAB documentation

  MATLAB → Programming Fundamentals → Performance

- Memory Management Guide

  www.mathworks.com/support/tech-notes/1100/1106.html?BB=1

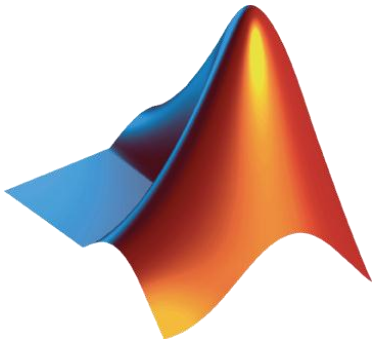
- The Art of MATLAB, Loren Shure's blog

  blogs.mathworks.com/loren/

- MATLAB Answers

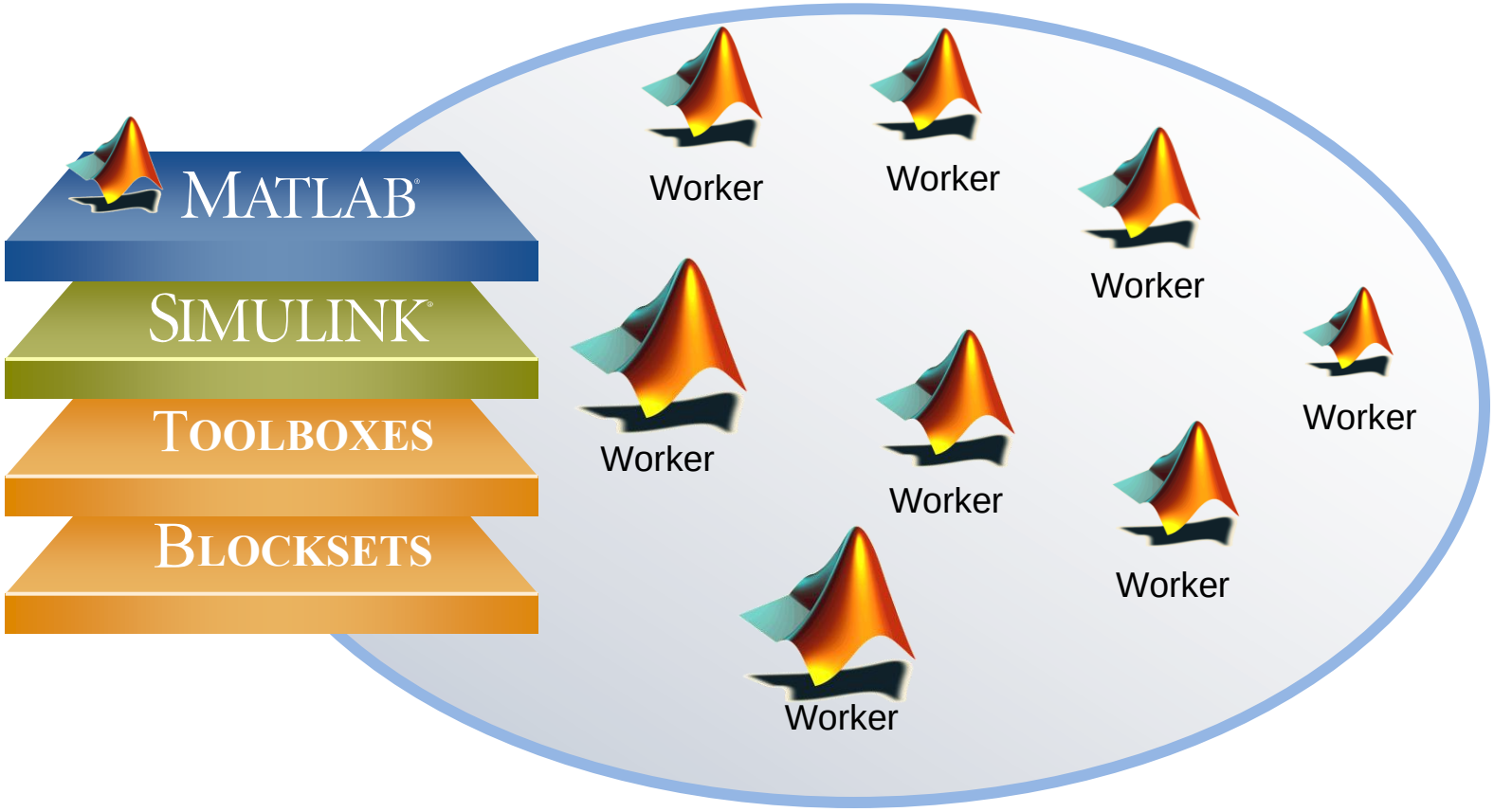
  http://www.mathworks.com/matlabcentral/answers/

Parallel computing with MATLAB



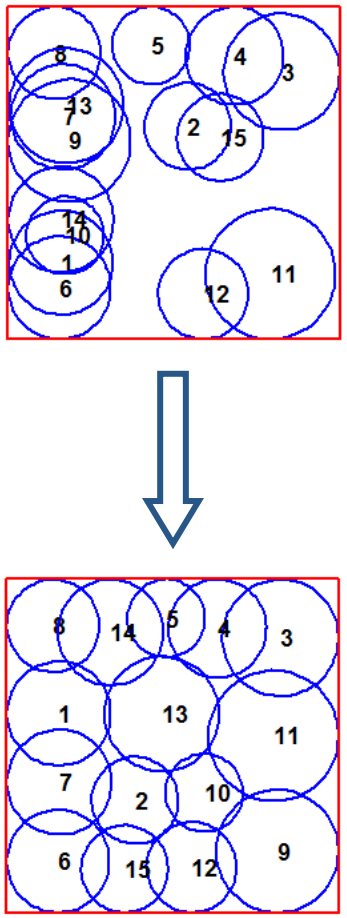
Going Beyond Serial MATLAB Applications

Use the power of multi-core machines and clusters



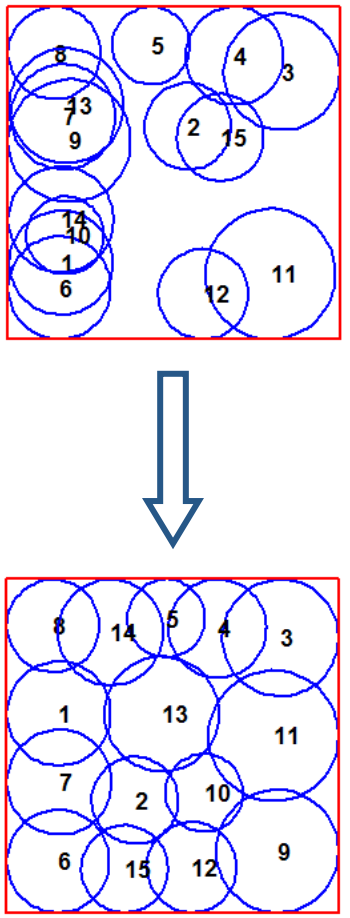
Example: Optimizing Tower Placement

- Determine location of cell towers
- Maximize coverage
- Minimize overlap



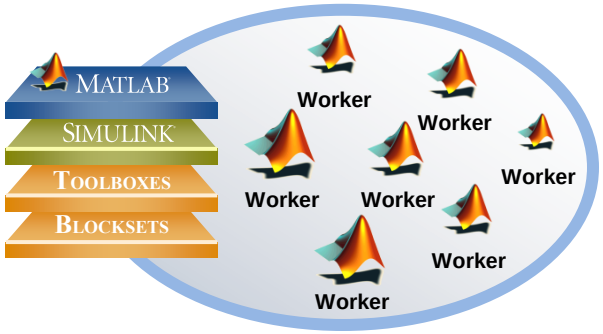
Summary of Example

- Enabled built-in support for Parallel Computing Toolbox in Optimization Toolbox
- Used a pool of MATLAB workers
- Optimized in parallel using `fmincon`



Tools Providing Parallel Computing Support

- Optimization Toolbox
- Global Optimization Toolbox
- Statistics Toolbox
- Communications System Toolbox
- Simulink Design Optimization
- Bioinformatics Toolbox
- Image Processing Toolbox
- …



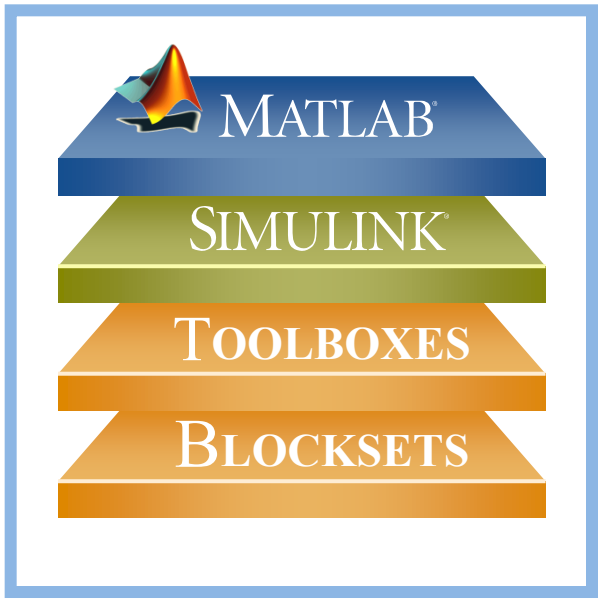
Directly leverage functions in Parallel Computing Toolbox

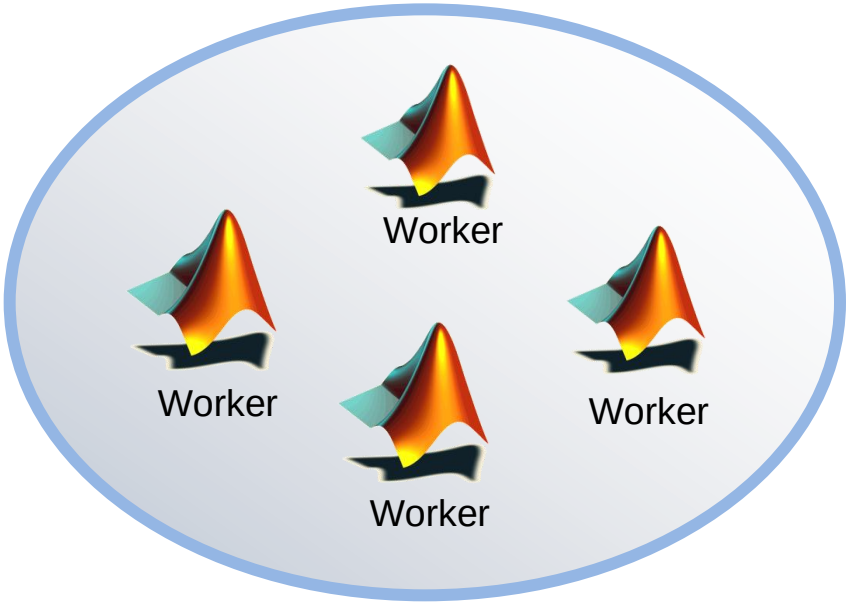
http://www.mathworks.com/products/parallel-computing/builtin-parallel-support.html

Agenda

- Task parallel applications
- GPU acceleration
- Data parallel applications
- Using clusters and grids

Task Parallel Applications



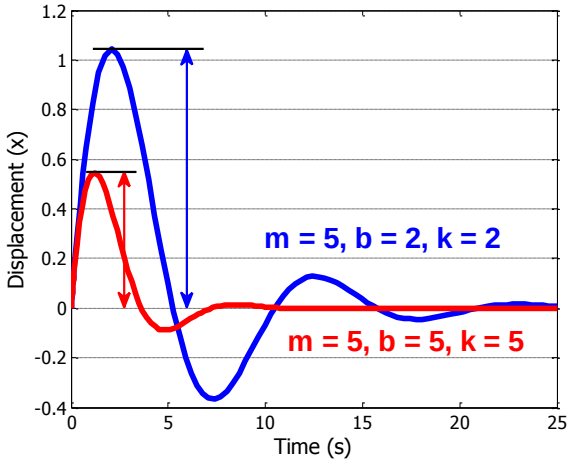


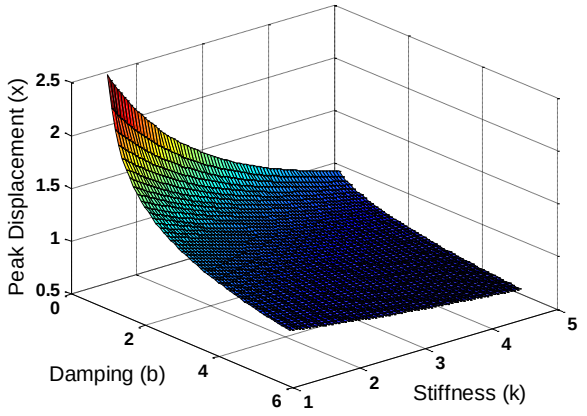
Example: Parameter Sweep of ODEs

- Solve a 2nd order ODE

$$\overbrace{5}\,m\ddot{x} + \underbrace{b}_{1,2,\ldots}\,\dot{x} + \underbrace{k}_{1,2,\ldots}\,x = 0$$

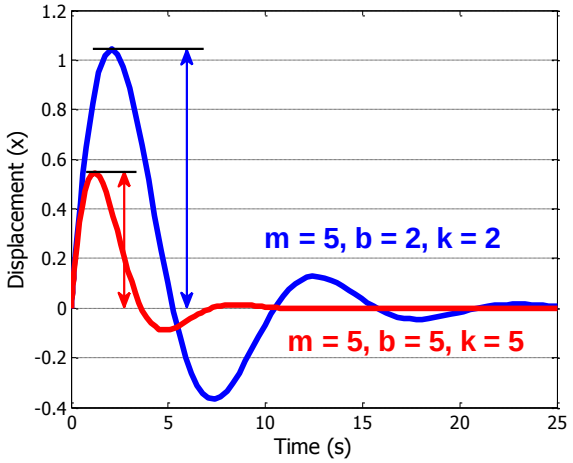
- Simulate with different values for ***b*** and ***k***
- Record peak value for each run
- Plot results

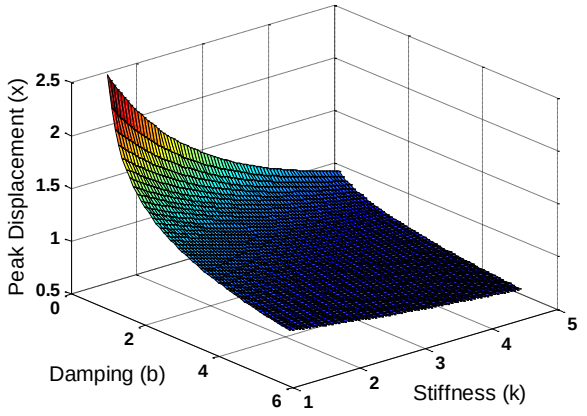




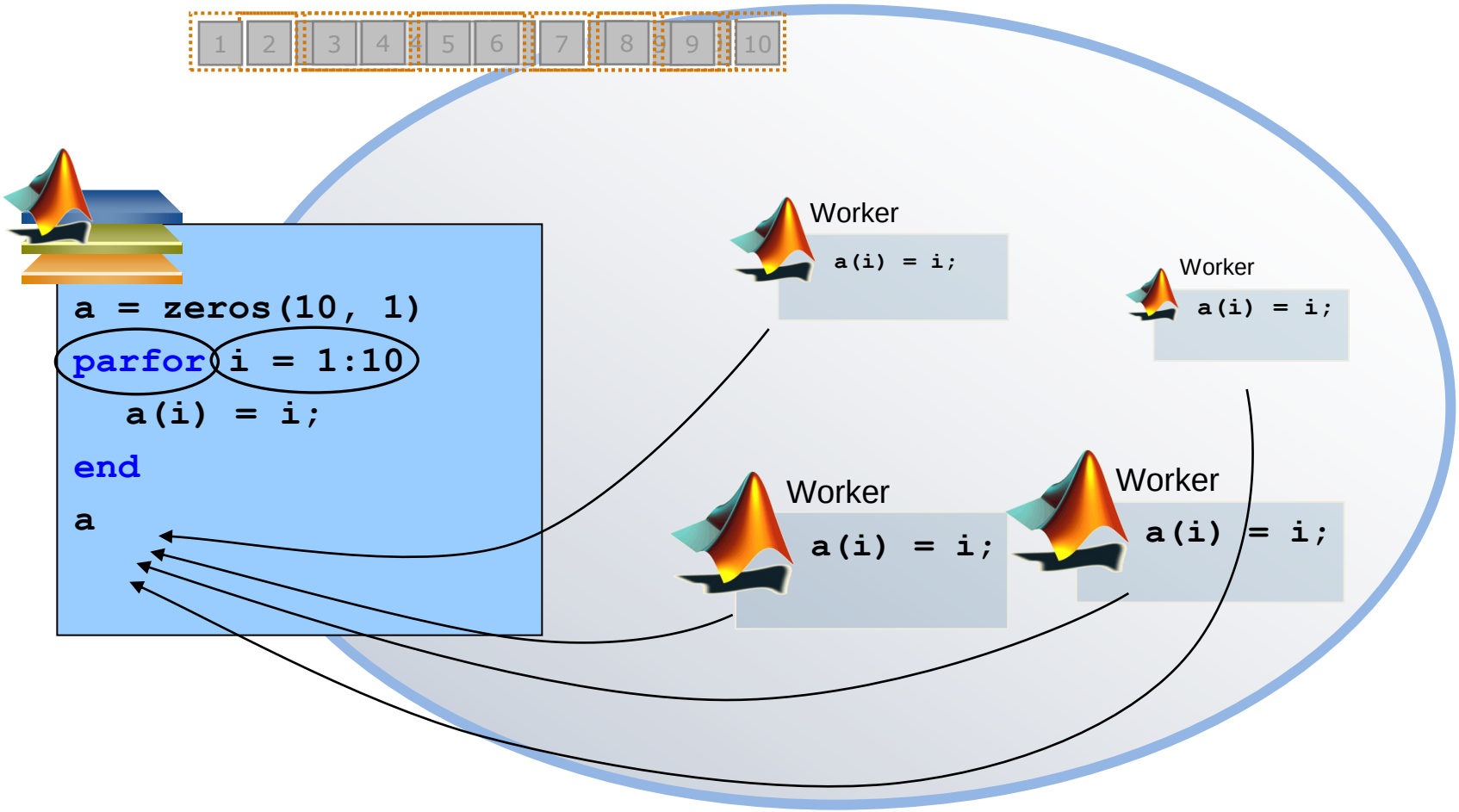
Summary of Example

- Mixed task-parallel and serial code in the same function
- Ran loops on a pool of MATLAB resources





The Mechanics of `parfor` Loops



Pool of MATLAB Workers

Converting `for` to `parfor`

- Requirements for `parfor` loops
  - Task independent
  - Order independent
- Constraints on the loop body
  - Cannot “introduce” variables (e.g. `load`, etc.)
  - Cannot contain `break` or `return` statements
  - Cannot contain another `parfor` loop

Advice for Converting `for` to `parfor`

- Use Code Analyzer to diagnose `parfor` issues
- If your `for` loop cannot be converted to a `parfor`, consider wrapping a subset of the body to a function
- Read the section in the documentation on classification of variables

Agenda

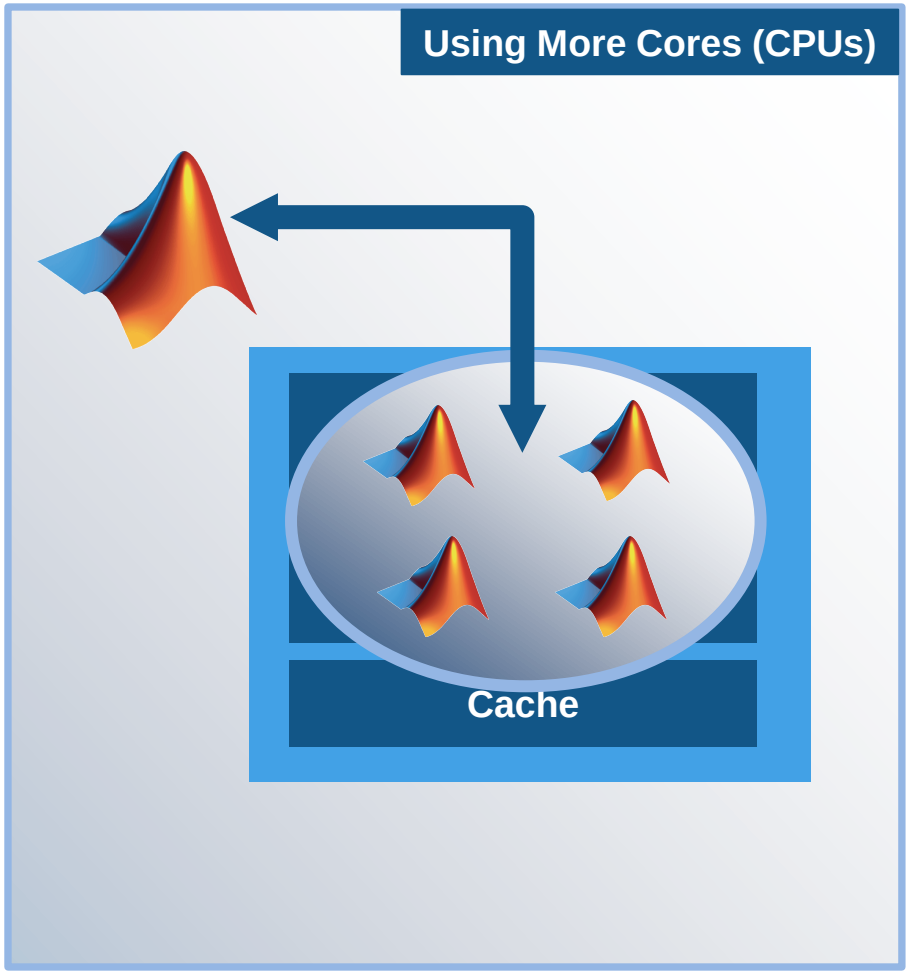
- Task parallel applications
- 
  GPU acceleration
- Data parallel applications
- Using clusters and grids

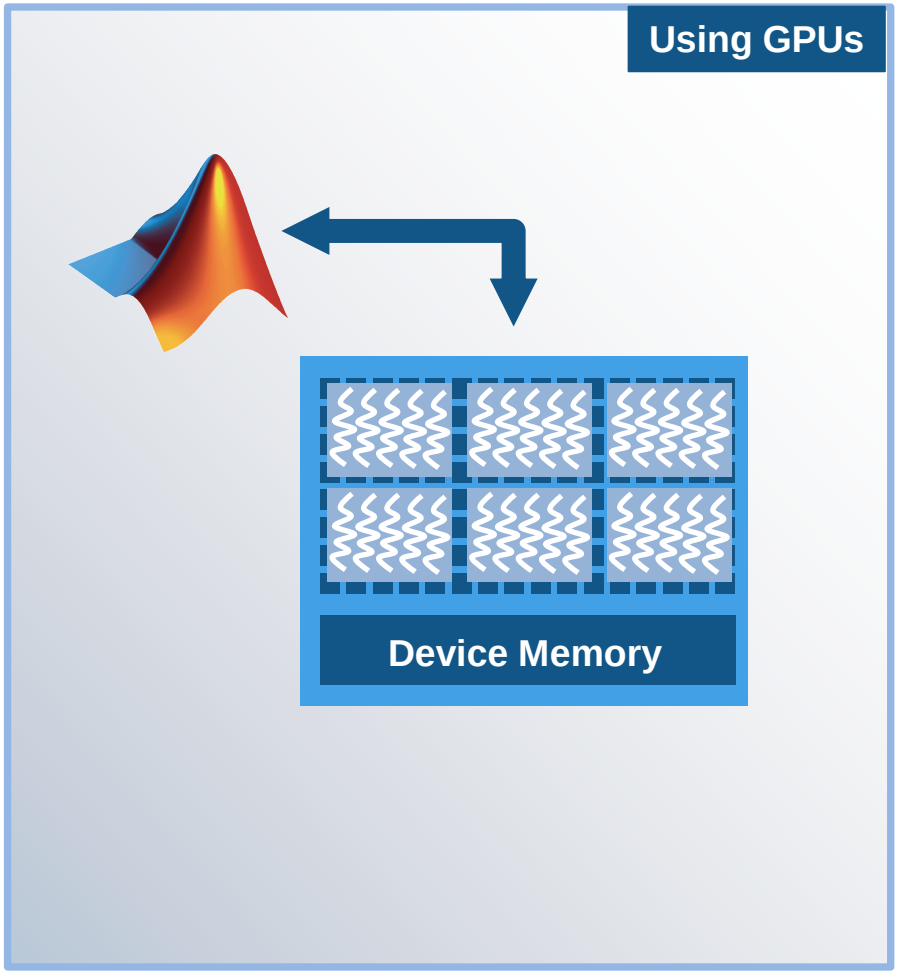
What is a Graphics Processing Unit (GPU)

- Originally for graphics acceleration, now also used for scientific calculations
- Massively parallel array of integer and floating point processors
  - Typically hundreds of processors per card
  - GPU cores complement CPU cores
- Dedicated high-speed memory

* Parallel Computing Toolbox requires NVIDIA GPUs with Compute Capability 1.3 or greater, including NVIDIA Tesla 10-series and 20-series products. See http://www.nvidia.com/object/cuda_gpus.html for a complete listing

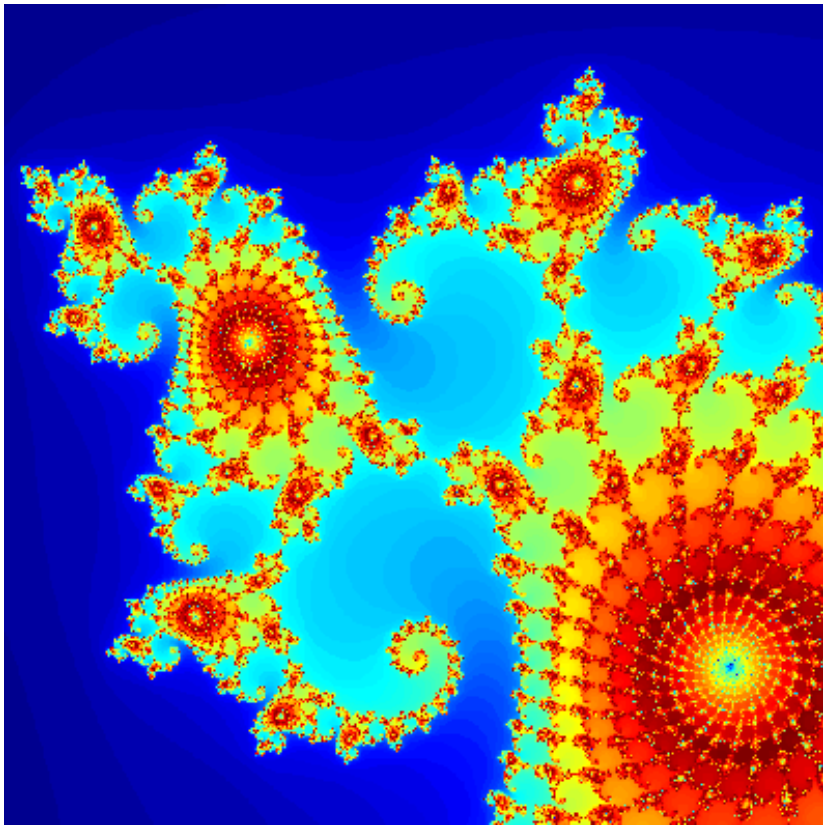
Performance Gain with More Hardware

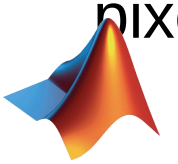




Example: Mandelbrot set

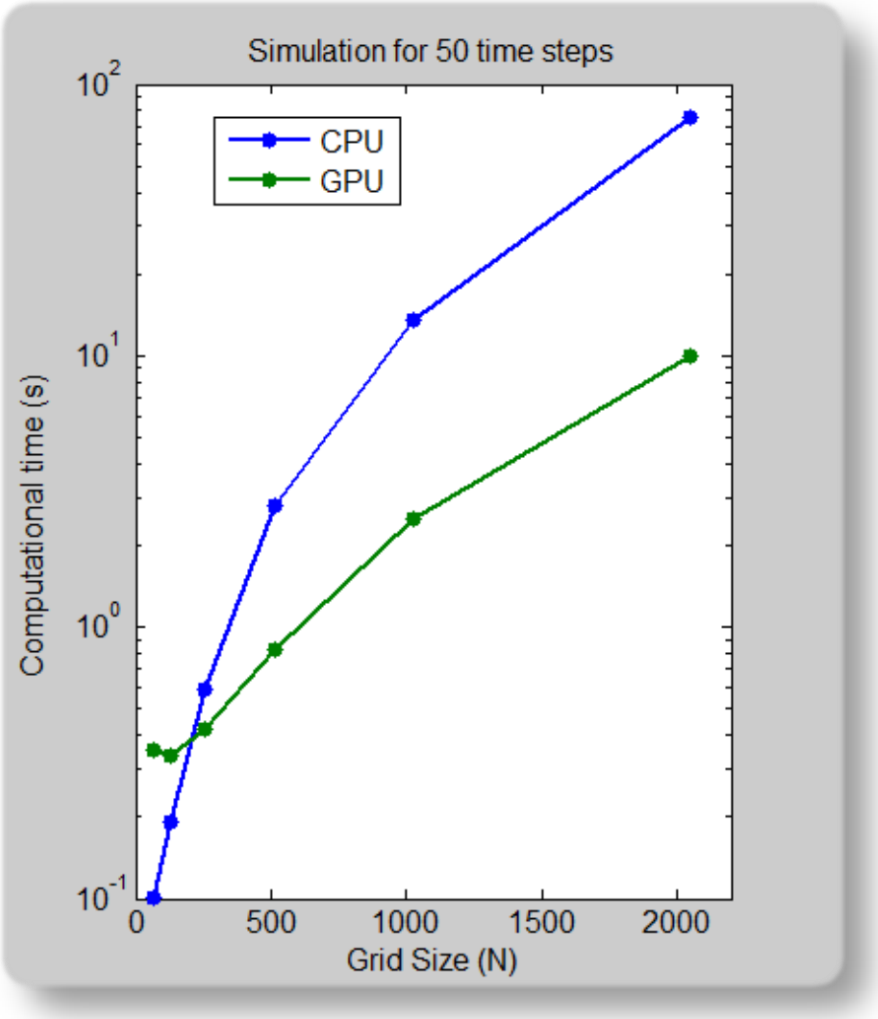
- The color of each pixel is the result of hundreds or thousands or iterations
- Each pixel is independent of the other pixels
- Hundres of thousands of pixels





Real-world performance increase

Solving a wave equation



Grid Size	CPU (s)	GPU (s)	Speedup
64 x 64	0.1004	0.3553	0.28
128 x 128	0.1931	0.3368	0.57
256 x 256	0.5888	0.4217	1.4
512 x 512	2.8163	0.8243	3.4
1024 x 1024	13.4797	2.4979	5.4
2048 x 2048	74.9904	9.9567	7.5

Intel Xeon Processor X5650, NVIDIA Tesla C2050 GPU

Summary of Options for Targeting GPUs

- Use built-in support in toolboxes.*
- Use GPU array interface with MATLAB built-in functions**
- Execute custom functions on elements of the GPU array
- Create kernels from existing CUDA code and PTX files

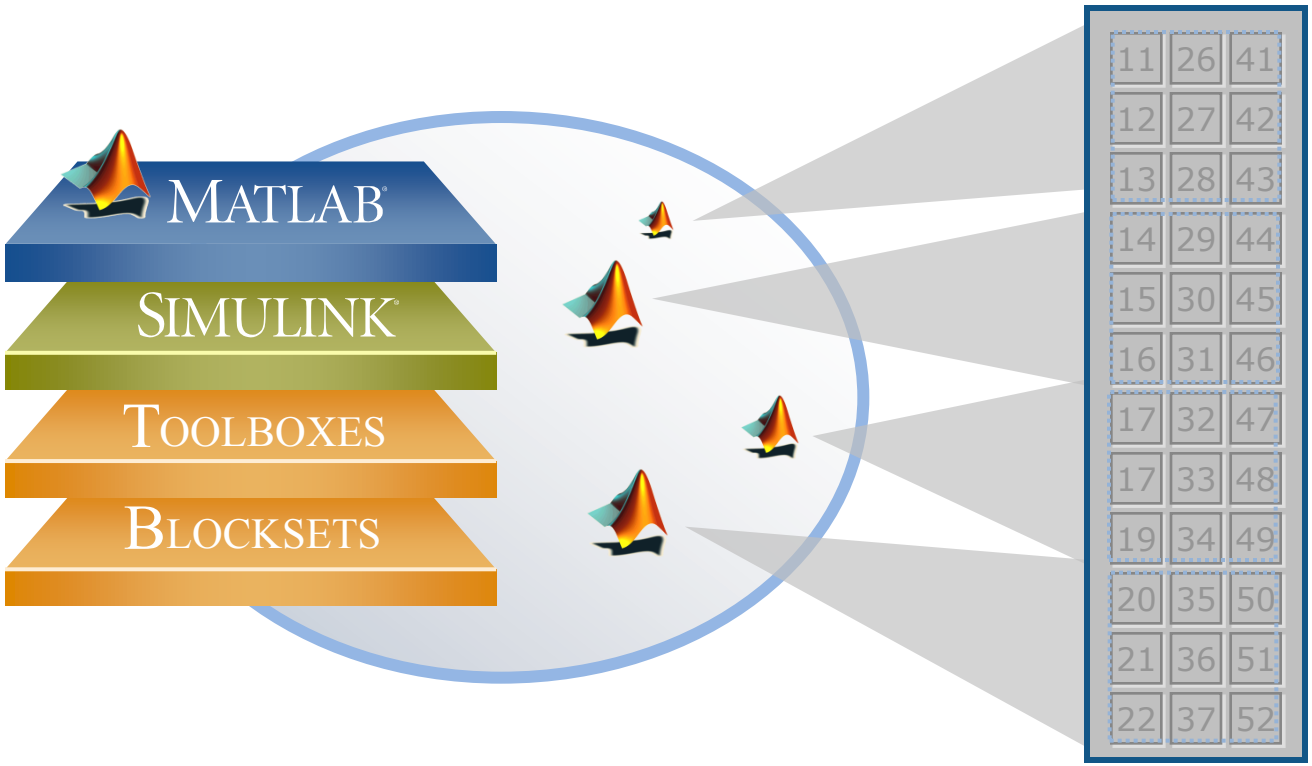
* Signal Processing Toolbox, Neural Network Toolbox and Phased Array Toolbox

** Over 200 functions support GPU Array data

Agenda

- Task parallel applications
- GPU acceleration
-  Data parallel applications
- Using clusters and grids

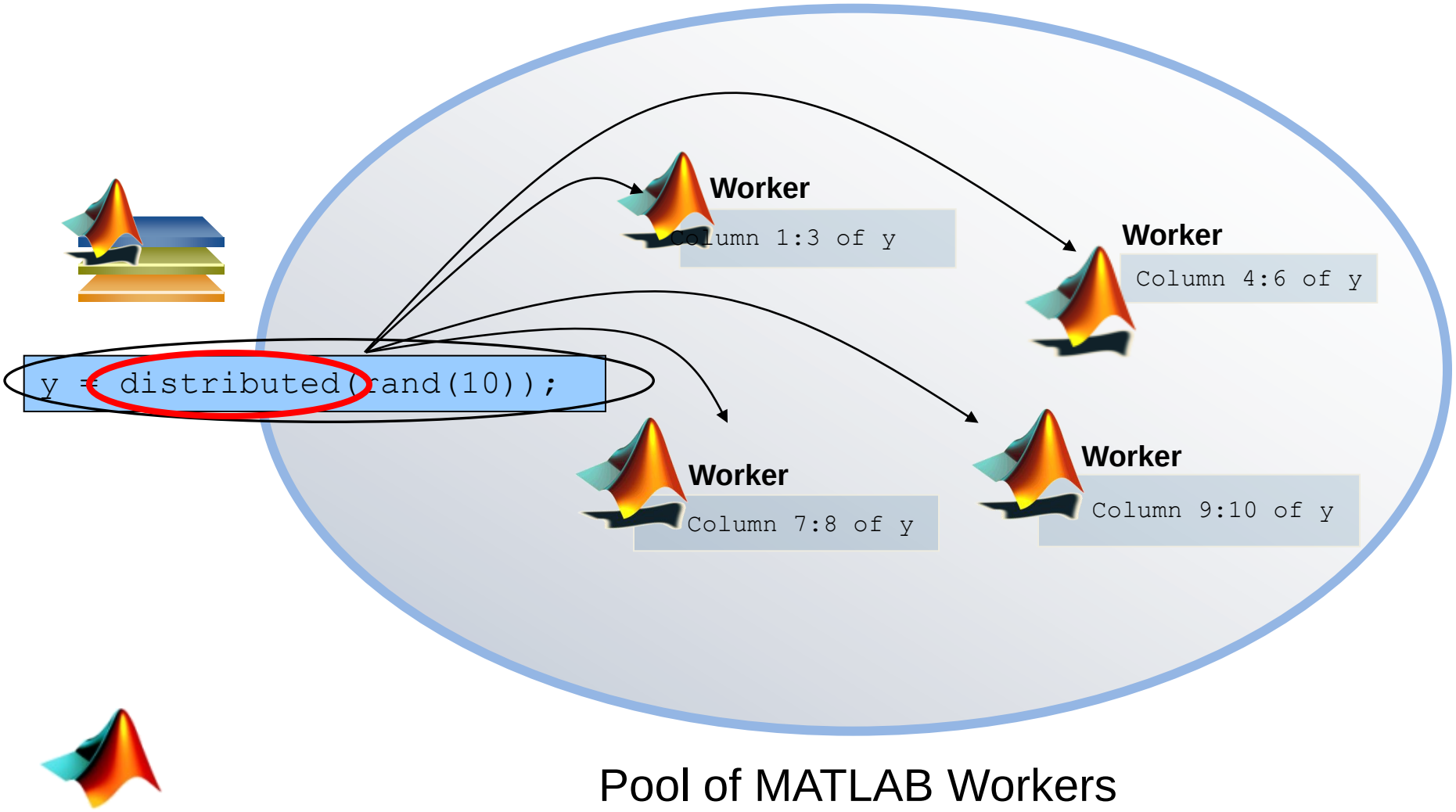
Big data: Distributed Arrays



Remotely Manipulate Array
from Desktop

Distributed Array
Lives on the Cluster

Big Data: Distributed Arrays



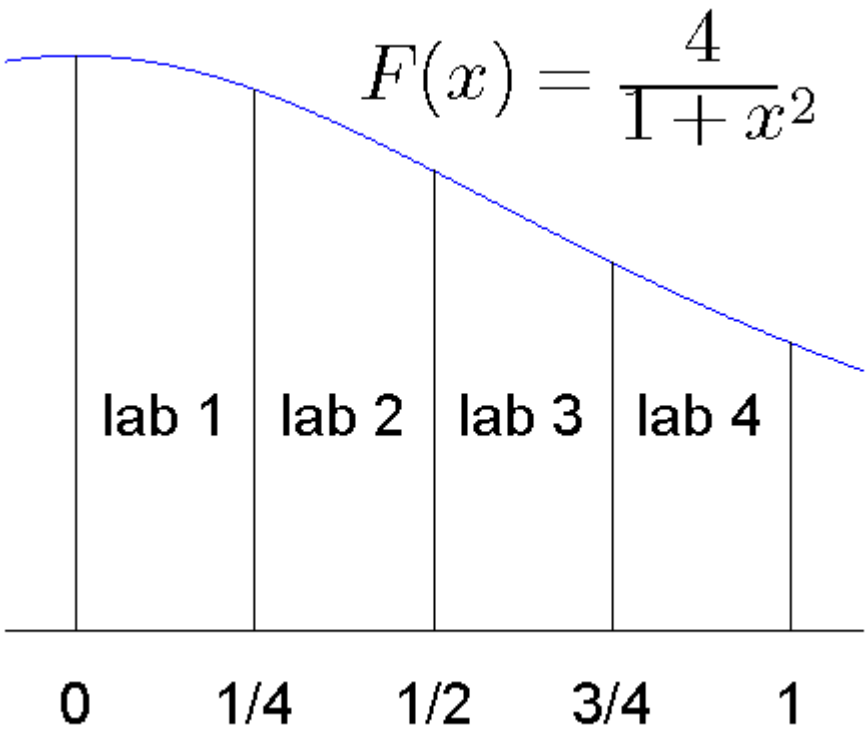
Pool of MATLAB Workers

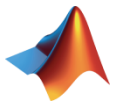
MATLAB Functions on Distributed Arrays ~ 150 supported functions

Type of Function	Function Names
Data functions	cumprod, cumsum, fft, max, min, prod, sum
Data type functions	arrayfun, cast, cell2mat, cell2struct, celldisp, cellfun, char, double, fieldnames, int16, int32, int64, int8, logical, num2cell, rmfield, single, struct2cell, swapbytes, typecast, uint16, uint32, uint64, uint8
Elementary and trigonometric functions	abs, acos, acosd, acosh, acot, acotd, acoth, acsc, acscd, acsch, angle, asec, asecd, asech, asin, asind, asinh, atan, atan2, atand, atanh, ceil, complex, conj, cos, cosd, cosh, cot, cotd, coth, csc, cscd, csch, exp, expm1, fix, floor, hypot, imag, isreal, log, log10, log1p, log2, mod, nextpow2, nthroot, pow2, real, reallog, realpow, realsqrt, rem, round, sec, secd, sech, sign, sin, sind, sinh, sqrt, tan, tand, tanh
Elementary matrices	cat, diag, eps, find, isempty, isequal, isequaln, isfinite, isinf, isnan, length, meshgrid, ndgrid, ndims, numel, reshape, size, sort, tril, triu
Matrix functions	chol, eig, inv, lu, norm, normest, qr, svd
Array operations	all, and (&), any, bitand, bitor, bitxor, ctranspose ('), end, eq (==), ge (>=), gt (>), horzcat ([]), ldivide (.\), le (<=), lt (<), minus (-), mldivide (\), mrdivide (/), mtimes (*), ne (~=), not (~), or (|), plus (+), power (.^), rdivide (./), subsasgn, subsindex, subsref, times (.*), transpose (.'), uminus (-), uplus (+), vertcat ([;]), xor
Sparse matrix functions	full, issparse, nnz, nonzeros, nzmax, sparse, spfun, spones
Special functions	dot

Demo: Approximation of π

$$\int_0^1 \frac{4}{1+x^2}\,dx = \pi$$





Interactive to Scheduling

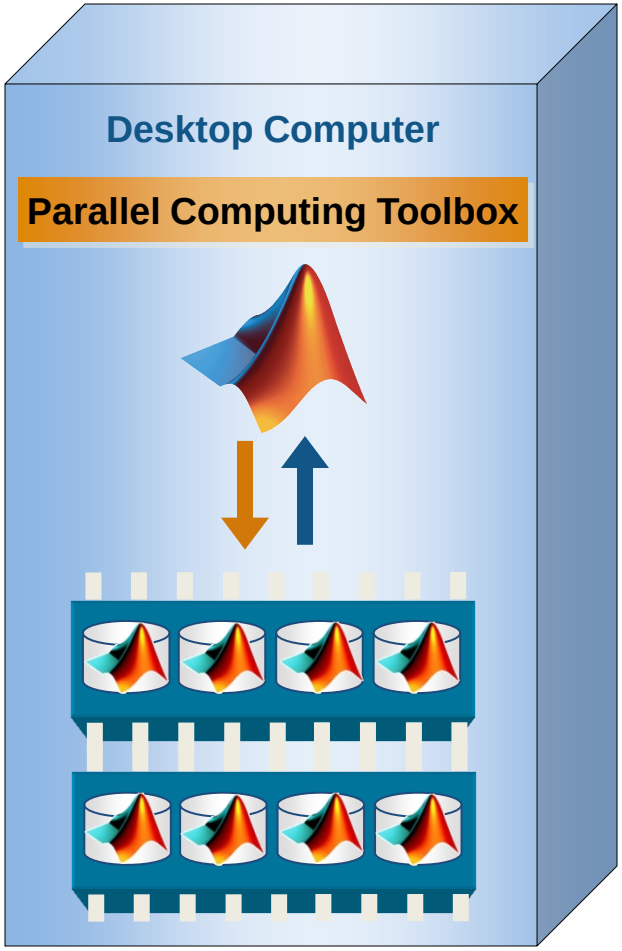
- Interactive
  - Great for prototyping
  - Immediate access to MATLAB workers

- Scheduling: `batch('script.m');`
  - Offloads work to other MATLAB workers (local or on a cluster)
  - Access to more computing resources for improved performance
  - Frees up local MATLAB session

Agenda

- Task parallel applications
- GPU acceleration
- Data parallel applications
-  Using clusters and grids

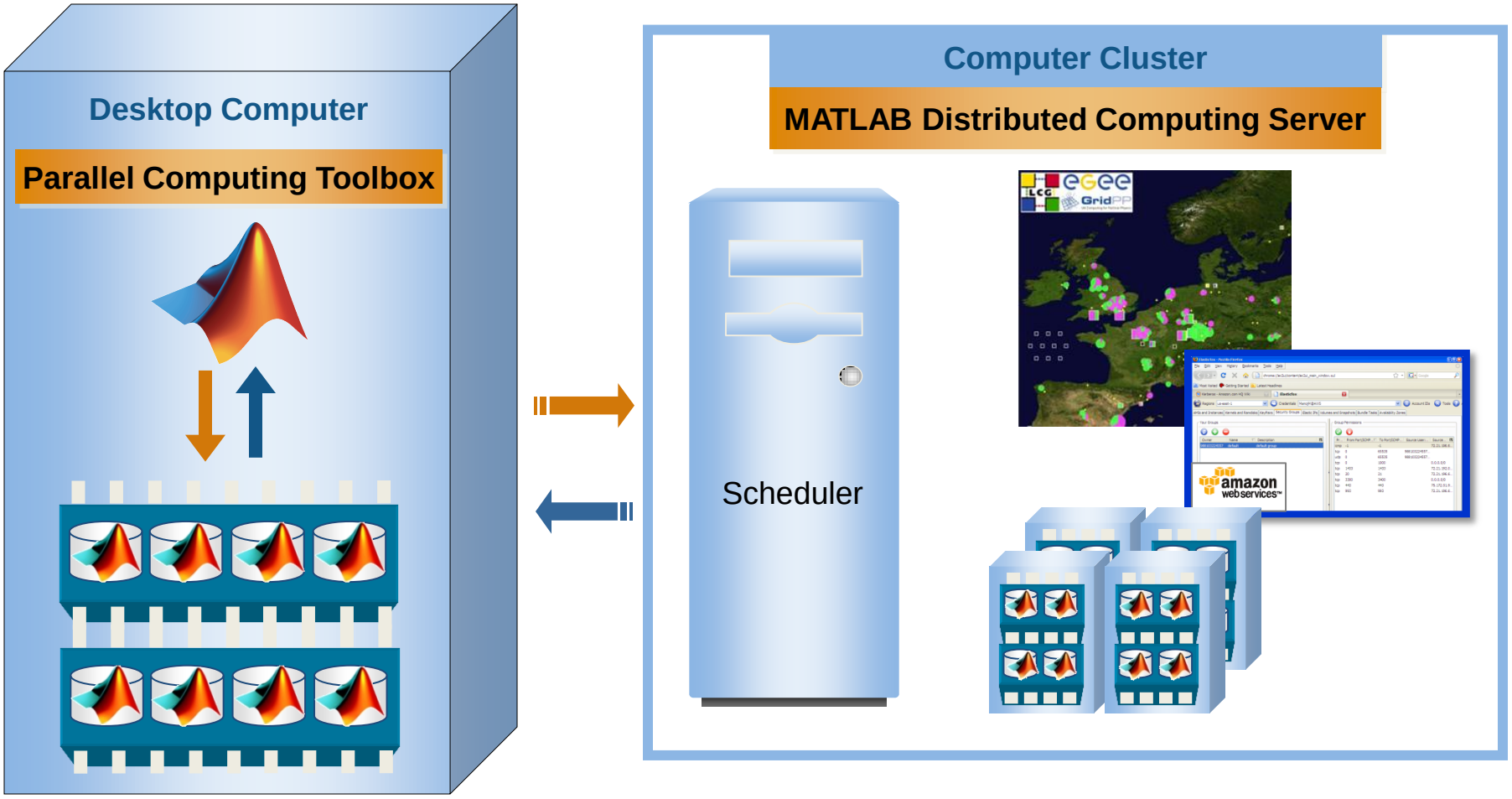
Run up to 12 Local Workers on Desktop



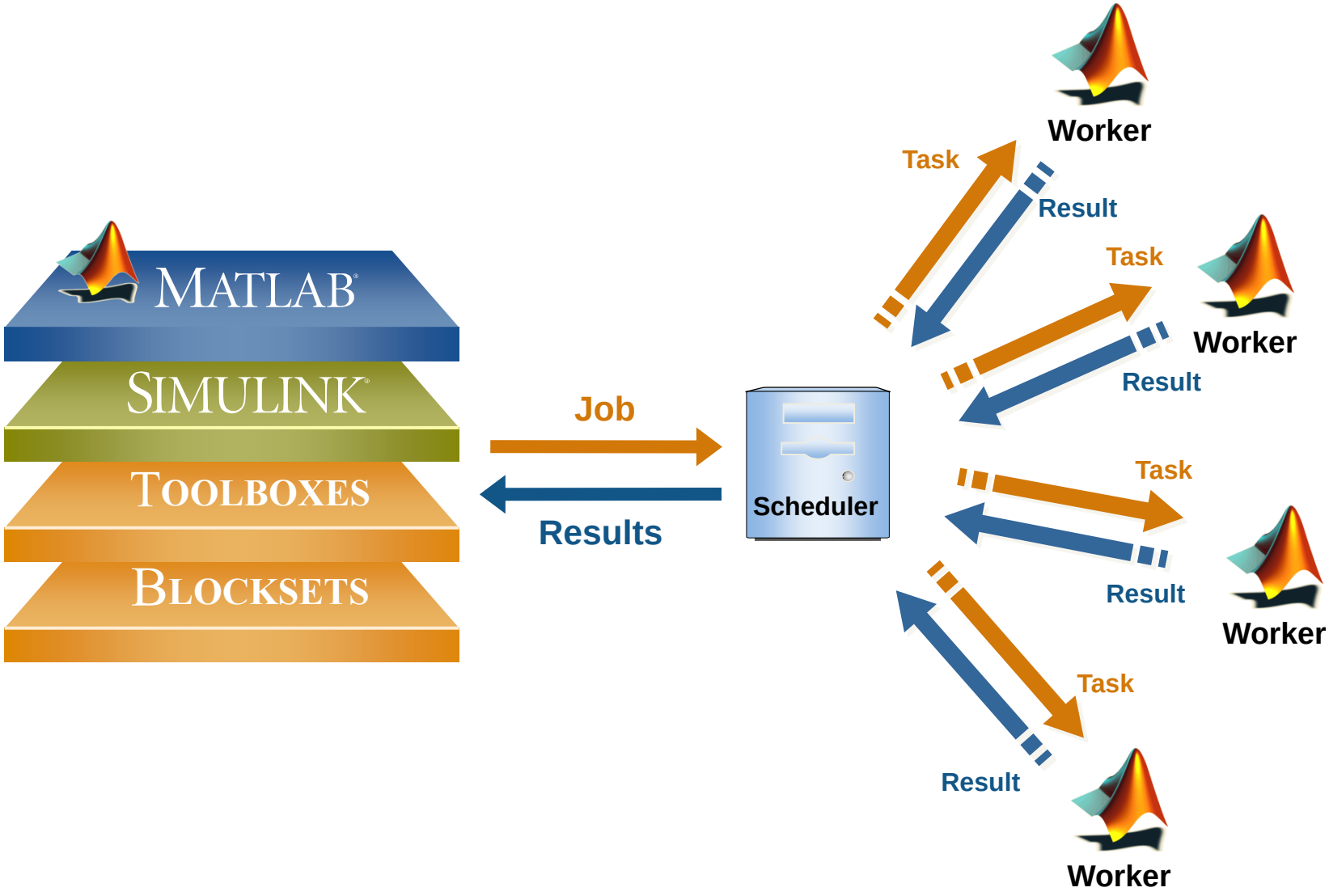
- Rapidly develop parallel applications on local computer
- Take full advantage of desktop power
- Separate computer cluster not required

Scale Up to Clusters, Grids and Clouds

No changes in your code



Scheduling Jobs and Tasks



Summary

- Developing parallel computations in MATLAB is easy.
- Parallel Computing Toolbox allows you to develop task-parallel, GPU, and data-parallel tasks
- Code developed using Parallel Computing Toolbox can run on clusters with minimal changes.
- GPU computing gives 5-10 times speedup with consumer hardware.