2011 Technology Trend Seminar

Speed up numerical analysis with MATLAB

MathWorks:  Giorgia Zucchelli
            Marieke van Geffen
            Rachid Adarghal

TU Delft:   Prof.dr.ir. Kees Vuik

Thales Nederland:  Dènis Riedijk

© 2011 The MathWorks, Inc.
<table>
<thead>
<tr>
<th>Time</th>
<th>Session</th>
</tr>
</thead>
<tbody>
<tr>
<td>14:00</td>
<td>Welcome and introduction</td>
</tr>
<tr>
<td>14:10</td>
<td>Overview of numerical analysis techniques with MATLAB</td>
</tr>
<tr>
<td>14:50</td>
<td>Break</td>
</tr>
<tr>
<td>15:10</td>
<td>Speeding up the execution of your MATLAB code</td>
</tr>
<tr>
<td>16:00</td>
<td>Break</td>
</tr>
<tr>
<td>16:20</td>
<td>GPU computing with MATLAB</td>
</tr>
<tr>
<td>16:30</td>
<td>Use of GPUs at Thales Nederland</td>
</tr>
<tr>
<td>17:00</td>
<td>Discussion and wrap up</td>
</tr>
</tbody>
</table>
Overview of numerical analysis techniques with MATLAB
Numerical analysis tasks

- Solving linear equations
- Interpolation
- Finding zeros and roots
- Optimization and least squares
- Determining integrals
- Statistics and random number generation
- Fourier analysis
- Ordinary differential equations
- Partial differential equation
Numerical analysis with MATLAB (and Toolboxes)

- Solving linear equations
- Interpolation
- Finding zeros and roots
- Optimization and least squares
- Determining integrals
- Statistics and random number generation
- Fourier analysis
- Ordinary differential equations
- Partial differential equation
Do need to solve numerical analysis tasks?

- For standard tasks, do not re-invent the wheel:
  - MATLAB and Toolboxes
  - MATLAB Central
  - Sources developed by others, eg. C, Fortran code

- For advanced research:
  - Choose your preferred language / environment
  - Develop innovation using MATLAB for testing and visualization
Do need to solve numerical analysis tasks?

- For standard tasks, do not re-invent the wheel:
  - MATLAB and Toolboxes
  - MATLAB Central
  - Sources developed by others, eg. C, Fortran code

**Example: curve fitting**

- For advanced research:
  - Choose your preferred language / environment
  - Develop innovation using MATLAB for testing and visualization

**Example: import C code in MATLAB**
Example: curve fitting with MATLAB

Two common challenges in creating an accurate curve fit:

Can’t describe the relationship between your variables

Can’t specify good starting points for your solvers
Challenge 1

Generating a Good Fit
Without Domain Knowledge
Regression techniques

- Require that the user specify a model
- Choice of model is based on domain knowledge

Example - population models

Logistic Growth

\[ N_t = \frac{N_0 \times K}{N_0 + (K - N_0) \times \exp (-r_0 \times t)} \]

Exponential Growth

\[ N_t = N_0 \times e^{(r \times t)} \]
What if you don’t know what model to use?
What if you don’t know what model to use?

Line

Quadratic

Rational
Demo
Summary

- Nonparametric fitting
  - LOWESS: Curve Fitting Toolbox
  - Cross Validation: Statistics Toolbox
Challenge 2

*Pitfalls Using Nonlinear Regression*
Nonlinear regression example

\[ \text{Concentration} = \beta_1 e^{-\beta_4 t} + \beta_2 e^{-\beta_5 t} + \beta_3 e^{-\beta_6 t} \]
Nonlinear regression example

Concentration = $\beta_1 e^{-\beta_4 t} + \beta_2 e^{-\beta_5 t} + \beta_3 e^{-\beta_6 t}$
Practical implications: What does a bad $R^2$ mean?

- Bad model specification?
- Poor set of starting conditions?
Demo
Summary

- Use **MultiStart** to identify good starting conditions
  - Global Optimization Toolbox
- Suitable for curve and surface fitting
Challenge

Re(use) external code in MATLAB
Using (external) C code in MATLAB

- Create a MATLAB callable executable (MEX) from C, C++ or Fortran

  or

- Use MATLAB Coder to invoke the C code function
Importing C code creating a MEX interface

MEX wrapper

```c
#include "ArrayProduct.h"

void ArrayProduct(double x, double *y, double *z, int n)
{
    int i;
    for (i=0; i<n; i++) {
        z[i] = x * y[i];
    }
}
```
Demo
MEX wrapper to invoke C code in MATLAB

- Data type conversion
  - Using mx library
- Dynamic memory management
- Error management and checking of the arguments
Summary

- One time-only effort
- Invoke your C code in MATLAB
  - Just like a MATLAB function
  - Maintaining the performances
  - Use MATLAB infrastructure
  - Share your code with colleagues who are not C programmers
Importing C code with MATLAB Coder

MATLAB function (MEX)
Demo
Using MATLAB Coder to invoke C code

- `coder.ceval` to invoke the C code within a MATLAB function
- MATLAB Coder to generate source code (C code) from the MATLAB function
- The code is automatically embedded in the generated function
- The code is compiled as a MEX for rapid execution in MATLAB
Summary

- Easy interface
  - Do not require to program MEX interface

- Only a subset of MATLAB language and Toolboxes is supported
  - Code generation MATLAB subset
  - Suitable also for embedded applications
Key takeaways

- For standard tasks, do not re-invent the wheel:
  - MATLAB and Toolboxes
  - MATLAB Central
  - Sources developed by others, eg. C, Fortran code

- For advanced research:
  - Choose your preferred language / environment
  - Develop innovation using MATLAB for testing and visualization
Speeding up the execution of your MATLAB code
Hardware solutions

- More processing power
  - Faster processors
  - Dedicated hardware
  - More processors

- More memory
  - 32 bit operating systems (4 GB of address space)
  - 64 bit operating systems (16,000 GB of address space)

4000 times the address space on a 64 bit OS
Hardware & software solutions

- More processing power
  - Faster processors
  - Dedicated hardware
  - More processors

- More memory
  - 32 bit operating systems (4 GB of address space)
  - 64 bit operating systems (16,000 GB of address space)
Slow execution? Out of memory errors?

- Know your enemy:
  - Understand the constraints
  - Identify bottlenecks
- Exploit data and task parallelism
- Find the best tradeoff between programming effort and achieving your goals
MATLAB is multithreaded: new releases are faster

- No code changes required
- Enabled at MATLAB start-up
- Vector math improved
  - Linear algebra operations
  - Element-wise operations
Understanding MATLAB improves efficiency

- How does MATLAB store data?
- What data types does MATLAB support?
- How much overhead is there for arrays, structures, and cell arrays?
- When are data copies made?
What is the largest array you can create in MATLAB on 32 bit Windows XP?

a) 0.5 GB  
b) 1.0 GB  
c) 1.5 GB  
d) 2.0 GB  
e) 2.5 GB
Contiguous memory and copy on write

```
>> x = rand(100,1)
>> y = x
>> y(1) = 0
```

- Do not grow arrays within loops!
In-place memory operations

\[
\begin{align*}
\gg x &= \text{rand}(100,1) \\
\gg x &= 2x + 12; \\
\gg y &= 2x + 12;
\end{align*}
\]

- Tradeoff readability with performances

\[
\begin{align*}
x &= \text{rand}(100,1) \\
x &= x \times 2 + 12; \\
y &= x \times 2 + 12;
\end{align*}
\]
How much memory is needed to plot a 10 MB double array?

\[
\text{>> } y = \text{rand}(125e4,1); \\
\text{>> plot}(y);
\]

- a) 0 MB
- b) 10 MB
- b) 20 MB
- c) 30 MB
- d) 40 MB
Plot only what you need

- Every plot independently stores x and y data
  ```matlab
  >> y = rand(125e4,1); %10MB
  >> plot(y) ; %20MB for x and y data
  ```

- Integers plotted as doubles

- Strategies:
  - Downsample your data prior to plotting (e.g. every 10\textsuperscript{th} element)
  - Divide your data into regular intervals and plot values of interest
Example: fitting data

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

- **File I/O**
  - Disk is slow compared to RAM
  - When possible, use `load` and `save` commands

- **Displaying output**
  - Creating new figures is expensive
  - Writing to command window is slow

- **Computationally intensive**
  - Trade-off modularization, readability and performance
Techniques for improving performance

- Vectorization
  - Take full advantage of BLAS and LAPACK
  - Brute force vectorization: cellfun, structfun, arrayfun ...

- Preallocation
  - Minimize changing variable class

- Mexing (compiled code)
Programming parallel applications

Level of control

- Minimal
- Some
- Extensive

Required effort

- None
- Straightforward
- Involved
Programming parallel applications

Level of control
- Minimal
- Some
- Extensive

Required effort
- Built-in into Toolboxes
- Straightforward
- Involved
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`
Parallel support in Optimization Toolbox

- **Functions:**
  - `fmincon`
    - Finds a constrained minimum of a function of several variables
  - `fminimax`
    - Finds a minimax solution of a function of several variables
  - `fgoalattain`
    - Solves the multiobjective goal attainment optimization problem

- Functions can take finite differences in parallel in order to speed the estimation of gradients
Toolboxes with built-in support

Contain functions to directly leverage the Parallel Computing Toolbox

- Optimization Toolbox
- Global Optimization Toolbox
- Statistics Toolbox
- SystemTest
- Simulink Design Optimization
- Bioinformatics Toolbox
- Model-Based Calibration Toolbox
- Communications System Toolbox
Programming parallel applications

Level of control

- Minimal
- Some
- Extensive

Required effort

- Built-in into Toolboxes
- High Level Programming (parfor)
- Involved
Parallel tasks
Example: parameter sweep of ODEs

- Solve a 2\textsuperscript{nd} order ODE
  \[ m\ddot{x} + b\dot{x} + kx = 0 \]
  \[ 1,2,... \quad 1,2,... \]

- 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
- Used Code Analyzer to help in converting existing `for`-loop into `parfor`-loop

![Graph showing displacement over time with different values of m, b, and k.](image)
Programming parallel applications

Level of control
- Minimal
- Some
- Extensive

Required effort
- Built-in into Toolboxes
- High Level Programming (distributed / spmd)
- Involved
Parallel data distribution
Programming parallel applications

Level of control

- Minimal
- Some
- Extensive

Required effort

- Built-in into Toolboxes
- High Level Programming (interactive vs. scheduled)
- Involved
Interactive to scheduled

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

- **Scheduled**
  - Offloads work to other MATLAB workers (local or on a cluster)
  - Access to more computing resources for improved performance
  - Frees up local MATLAB session
Scheduling work
Example: schedule processing

- Offload parameter sweep to local workers
- Get peak value results when processing is complete
- Plot results in local MATLAB
Summary of example

- Used **batch** for off-loading work
- Used **matlabpool** option to off-load and run in parallel
- Used **load** to retrieve worker’s workspace
Programming parallel applications

Level of control

- Minimal
- Some
- Extensive

Required effort

- Built-in into Toolboxes
- High Level Programming Constructs: (e.g. parfor, spmd, batch)
- Low-Level Programming Constructs: (e.g. Jobs/Tasks, MPI-based)
Task-parallel workflows

- **parfor**
  - Multiple independent iterations
  - Easy to combine serial and parallel code
  - Workflow
    - Interactive using *matlabpool*
    - Scheduled using *batch*

- **jobs/tasks**
  - Series of independent tasks; not necessarily iterations
  - Workflow ➔ Always scheduled
Example: scheduling independent simulations

- Offload three independent approaches to solving our previous ODE example
- Retrieve simulated displacement as a function of time for each simulation
- Plot comparison of results in local MATLAB
Summary of example

- Used `findResource` to find scheduler
- Used `createJob` and `createTask` to set up the problem
- Used `submit` to off-load and run in parallel
- Used `getAllOutputArguments` to retrieve all task outputs
MPI-Based functions for higher control

- High-level abstractions of MPI functions
  - `labSendReceive`, `labBroadcast`, and others
  - Send, receive, and broadcast any data type in MATLAB

- Automatic bookkeeping
  - Setup: communication, ranks, etc.
  - Error detection: deadlocks and miscommunications

- Pluggable
  - Use any MPI implementation that is binary-compatible with MPICH2
Run 8 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
Support for schedulers

Direct Support

Open API for others
Key takeaways

- Profile MATLAB to write efficient code
- Use parallel computing to speed up execution
- Tradeoff effort with speedup requirements
- Scale up to clusters without code changes
GPU computing with MATLAB
How to accelerate MATLAB on hardware

- Use DSPs for real-time execution
  - On target automatic C code generation
- Deploy on FPGAs
  - Synthesizable automatic HDL code generation
- Use multi-core PCs
  - Multithreaded parallel computing
- Distribute on a cluster
  - Distributed computing
- Use GP-GPUs  **New in R2010b**
Why GPUs now?

- GPUs are a commodity
- Massively parallel architecture that can speed up intensive computations
- GPUs are more generically programmable

*From 3D gaming to scientific computing*
Speeding up applications with GPUs

- Different achievable speedups depending on:
  - Hardware
  - Type of application
  - Programmer skills

Source: NVIDIA
Applications that will run faster on GPUs

- Massively parallel tasks
- Computationally intensive tasks
- Tasks that have limited kernel size
- Tasks that do not necessarily require double accuracy

*All these requirements need to be satisfied!*
Programming GPU applications

Level of control
- Minimal
- Some
- Extensive

Required effort
- None
- Straightforward
- Involved
Programming GPU applications

Level of control

- Minimal
- Some
- Extensive

Required effort

- Built-in functions
- Straightforward
- Involved
Invoke built-in MATLAB functions on the GPU

- Accelerate standard (highly parallel) functions
  - More than 100 MATLAB functions are already supported
  - Communication System Object: ldpc.encoder, ldpc.decoder

- Out of the box:
  - No additional effort for programming the GPU

- No accuracy for speed trade-off
  - Double floating-point precision computations
Invoke built-in MATLAB functions on the GPU
(1) Minimal effort, minimal level of control

- Define an array on the GPU
  \[ A = \text{rand}(1000,1); \]
  \[ B = \text{rand}(1000,1); \]
  \[ A\_gpu = \text{gpuArray}(A); \]
  \[ B\_gpu = \text{gpuArray}(B); \]

- Execute a built-in MATLAB function:
  \[ Y\_gpu = B\_gpu \backslash A\_gpu; \]

- Retrieve data from the GPU
  \[ \text{result} = \text{gather}(Y\_gpu); \]
Benchmarking A\b on the GPU

Speed up of computations on GPUs compared to CPUs

- CPU: Quad Core @2.4GHz
- GPU: Nvidia Tesla C2050

Execution time in seconds

Matrix size

512  1024  2048  4096
Programming GPU applications

Level of control
- Minimal
- Some
- Extensive

Required effort
- Built-in functions
- Functions on array data
- Involved
Run MATLAB functions on the GPU

- Accelerate scalar operations on large arrays
  - Take full advantage on data parallelism
- Out of the box:
  - No additional effort for programming the GPU
- No accuracy for speed trade-off
  - Double floating-point precision computations
Run MATLAB functions on the GPU

(2) Straightforward effort, regular level of control

- MATLAB function that perform element-wise arithmetic

```matlab
function y = TaylorFun(x)
    y = 1 + x*(1 + x*(1 + x*(1 + ... 
        x*(1 + x*(1 + x*(1 + x*(1 + ... 
        x*(1 + ./9)./8)./7)./6)./5)./4)./3)./2);
```

- Load data on the GPU

```matlab
A = rand(1000,1);
A_gpu = gpuArray(A);
```

- Execute the function as GPU kernels

```matlab
result = arrayfun(@TaylorFun, A_gpu);
```
Benchmarking vector operations on the GPU

Speed up of computations on GPU compared to CPU

- CPU: Dual Core @2.4GHz
- CPU: Quad Core @3GHz
- GPU: Nvidia Tesla C2050

Execution time in seconds vs. Length of Taylor Series (kernel complexity)
Programming GPU applications

Level of control

- Minimal
- Some
- Extensive

Required effort

- Built-in functions
- Functions on array data
- Directly invoke CUDA code
Directly invoke CUDA code from MATLAB

- Benefit from legacy code highly optimized for speed
  - Achieve all the speed improvement that CUDA can deliver

- Use MATLAB as a test environment
  - Generation of test signal
  - Post-processing of results

- Suitable also for non-experts
Invoke CUDA Code from MATLAB

(3) Involved effort, extensive level of control

- Compile CUDA (or PTX) code on the GPU

  \texttt{nvcc -ptx myconv.cu}

- Construct the kernel

  \texttt{k = parallel.gpu.CUDAKernel('myconv.ptx',}
  \texttt{ 'myconv.cu');}
  \texttt{k.GridSize} = [512 512];
  \texttt{k.ThreadBlockSize} = [32 32];

- Run the kernel using the MATLAB workspace

  \texttt{o = feval(k, rand(100, 1), rand(100, 1));}
  \texttt{or gpu data}

  \texttt{i1gpu = gpuArray(rand(100, 1, 'single'));}
  \texttt{i2gpu = gpuArray(rand(100, 1, 'single'));}
  \texttt{ogpu = feval(k, i1gpu, i2gpu);}
Can I use it now?

- GPU support is released with R2010b
  - Part of Parallel Computing Toolbox
- NVIDIA CUDA capable GPUs
  - With CUDA version 1.3 or greater
Key takeaways

- Tradeoff programming effort with level of control
- Non-experts can benefit from GPU computing
- CUDA programmers can test their code in MATLAB
Use of GPUs at Thales Nederland
Conclusions
Did you know about the campus-wide license to MATLAB & Simulink?

Through the Delft University campus wide license you can access MATLAB, Simulink and many add-on products

More than 50 MathWorks products for you available at Delft University of Technology
Visit MathWorks Web Site

- Learn about MathWorks products
- Discover resources for learning, teaching, and research
- Learn how MathWorks products are used in academia and industry

www.mathworks.nl/academia
The MATLAB and Simulink product families are fundamental computational tools at the world’s educational institutions. Adopted by more than 5000 universities and colleges, MathWorks products accelerate the pace of learning, teaching, and research in engineering and science. MathWorks products also help prepare students for careers in industry, where the tools are widely used for research and development.

Find out more in The MathWorks in Academia brochure.
Supporting Innovation

MATLAB Central

- Open exchange for the MATLAB and Simulink user community
- 800,000 visits per month
  - 50% increase over previous year

- File Exchange
  - Free file upload/download, including MATLAB code, Simulink models, and documents
  - File ratings and comments
  - Over 9,000 contributed files, 400 submissions per month, 25,500 downloads per day

- Newsgroup and Web Forum
  - Technical discussions about MATLAB and Simulink
  - 200 posts per day

- Blogs
  - Read posts from key MathWorks developers who design and build the products

Based on Feb-March 2009 data
Learning Resources

- **Interactive Video Tutorials** – Students learn the basics outside of the classroom with self-guided tutorials provided by MathWorks
  - MATLAB
  - Simulink
  - Signal Processing
  - Control Systems
  - Many more……

Visit www.mathworks.com/academia/student_center/tutorials
## Recorded Webinars

Learn more about MathWorks products and how they help solve complex technical issues through these online recorded webinars. To view a free webinar, select a language and topic, and then click on the link and complete the request form.

[Webinar RSS Feed](#)

Language of Webinar: [English](#)

<table>
<thead>
<tr>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction to Curve Fitting for Nonprogrammers</td>
</tr>
<tr>
<td>Introduction to Econometrics Toolbox</td>
</tr>
<tr>
<td>Developing a Financial Market Index Tracker with MATLAB OOP and Genetic Algorithms</td>
</tr>
<tr>
<td>Data Analysis with Statistics and Curve Fitting Toolboxes</td>
</tr>
<tr>
<td>Best Practices for Verification, Validation, and Test in Model-Based Design</td>
</tr>
<tr>
<td>What’s New for Object-Oriented Programming in MATLAB</td>
</tr>
<tr>
<td>Tips &amp; Tricks: Getting Started Using Optimization with MATLAB</td>
</tr>
</tbody>
</table>
2011 Technology Trend Seminar

Speed up numerical analysis with MATLAB

MathWorks: Giorgia Zucchelli
Marieke van Geffen
Rachid Adarghal

TU Delft: Prof.dr.ir. Kees Vuik

Thales Nederland: Dènis Riedijk

© 2011 The MathWorks, Inc.
Thank you for attending

Please fill out your evaluation form…

Stay with Us for a drink