Basics of MATLAB

TUM Graduate School Training

Dipl.-Ing. Markus Hornauer
Outline
Introduction to MATLAB

Basics:
1) Introduction
2) MATLAB Basics
3) 2D and 3D Plots
4) Data Import and Export

Advanced:
1) Programming with MATLAB
2) Graphical User Interfaces in MATLAB

Toolboxes:
1) Symbolic Math Toolbox
2) Control System Toolbox and Curve Fitting Toolbox
Your Expectations?
Introduction to MATLAB

References to the book MATLAB – Simulink – Stateflow
(Angermann, Beuschel, Rau, Wohlfarth, Oldenburg Verlag)
- Supported by MathWorks -
Outline
Introduction to MATLAB

Basics:
1) Introduction
2) MATLAB Basics
3) 2D and 3D Plots
4) Data Import and Export

Advanced:
1) Programming with MATLAB
2) Graphical User Interfaces in MATLAB

Toolboxes:
1) Symbolic Math Toolbox
2) Control System Toolbox and Curve Fitting Toolbox
MATLAB @ TUM

• Total Academic Headcount License (TAH) for whole TUM
• Free installation for all staff and students on office and home computers

For Details:  
https://matlab.rbg.tum.de/
Key Features

- High-level language for numerical computation, visualization, and application development
- Interactive environment for iterative exploration, design, and problem solving
- Mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration, and solving ordinary differential equations
- Built-in graphics for visualizing data and tools for creating custom plots
- Development tools for improving code quality and maintainability and maximizing performance
- Tools for building applications with custom graphical interfaces
- Functions for integrating MATLAB based algorithms with external applications and languages such as C, Java, .NET, and Microsoft® Excel®
- Release of MATLAB 1.0 in 1984 (commercial), as university tool since early 70s
Introduction
Technical Computing Workflow
Introduction

MATLAB Product Family

MATLAB® Product Family
- MATLAB
- Parallel Computing
  - Parallel Computing Toolbox
  - MATLAB Distributed Computing Server
- Math, Statistics, and Optimization
  - Symbolic Math Toolbox
  - Partial Differential Equation Toolbox
  - Statistics Toolbox
  - Curve Fitting Toolbox
  - Optimization Toolbox
  - Global Optimization Toolbox
  - Neural Network Toolbox
  - Model-Based Calibration Toolbox

Simulink® Product Family
- Simulink
- Event-Based Modeling
  - Stateflow
  - SimEvents
- Physical Modeling
  - Simscape
  - SimMechanics
  - SimDriveline
  - SimHydraulics
  - SimRF
  - SimElectronics
  - SimPowerSystems

Polyspace® Product Family
- Polyspace Bug Finder
- Polyspace Code Prover
- DO Qualification Kit (for DO-178)
- IEC Certification Kit (for ISO 26262 and IEC 61508)

Additional Products and Services
- MATLAB Student-Use Software
- Third-Party Products & Services
- Hardware Support Catalog

http://www.mathworks.com/products/
Introduction

MATLAB Product Family

Control System Design and Analysis
- Control System Toolbox
- System Identification Toolbox
- Fuzzy Logic Toolbox
- Robust Control Toolbox
- Model Predictive Control Toolbox
- Aerospace Toolbox

Signal Processing and Communications
- Signal Processing Toolbox
- DSP System Toolbox
- Communications System Toolbox
- Wavelet Toolbox
- RF Toolbox
- Phased Array System Toolbox
- LTE System Toolbox

Image Processing and Computer Vision
- Image Processing Toolbox
- Computer Vision System Toolbox
- Image Acquisition Toolbox
- Mapping Toolbox

Control System Design and Analysis
- Simulink Control Design
- Simulink Design Optimization
- Aerospace Blockset

Signal Processing and Communications
- DSP System Toolbox
- Communications System Toolbox
- SimRF
- Computer Vision System Toolbox

Code Generation
- Simulink Coder
- Embedded Coder
- HDL Coder
- Simulink PLC Coder
- Fixed-Point Designer
- DO Qualification Kit (for DO-178)
- IEC Certification Kit (for ISO 26262 and IEC 61508)

Real-Time Simulation and Testing
- Simulink Real-Time
- Real-Time Windows Target

MathWorks Services
- Software Maintenance
- Training
- Consulting

Excluded from TAH

http://www.mathworks.com/products/
Introduction
MATLAB Product Family

Test and Measurement
- Data Acquisition Toolbox
- Instrument Control Toolbox
- Image Acquisition Toolbox
- OPC Toolbox
- Vehicle Network Toolbox

Verification, Validation, and Test
- Simulink Verification and Validation
- Simulink Design Verifier
- SystemTest
- Simulink Code Inspector
- HDL Verifier
- Polyspace Bug Finder
- Polyspace Code Prover

Computation Graphics and Reporting
- Simulink 3D Animation
- Gauges Blockset
- Simulink Report Generator

- Digital Signal Processing
- Communications Systems
- Image and Video Processing
- FPGA Design and Codesign
- Mechatronics
- Test and Measurement
- Computational Biology
- Computational Finance

Discover How to Solve Your Computational Problem

http://www.mathworks.com/products/
# MATLAB Product Family

## Application Deployment
- MATLAB Compiler
- MATLAB Builder NE (for Microsoft .NET Framework)
- MATLAB Builder JA (for Java language)
- MATLAB Builder EX (for Microsoft Excel)
- Spreadsheet Link EX (for Microsoft Excel)
- MATLAB Production Server

## Database Connectivity and Reporting
- Database Toolbox
- MATLAB Report Generator

http://www.mathworks.com/products/
MATLAB Basics

MATLAB Desktop

- Command Window
- Workspace Browser
- Editor/Debugger
- Current Folder
cursor “up” in the command window
Introduction

Getting Help

- `>> help sin`
- `http://mathworks.de` -> Support -> Product Documentation
- `>> doc`
- `help` Search the web F1
Introduction

Getting Help – Technical Support

Contact Support by Email

Support Hotline
(by the way: Support Engineer at MathWorks DE is a good job opportunity…)

http://www.mathworks.de/support/contact_us/

Basics of MATLAB
Introduction

Getting Help – Bug Report

Select product and release

http://www.mathworks.com/support/bugreports
MathWorks online user Group:

- Exchange of user functions / scripts and add ons
- Supported by MathWorks employees
- Newsgroups and Blogs

http://www.mathworks.de/matlabcentral/
MATLAB Central – “Cody” online exercises

New to Cody? Let us show you how Cody works, or start solving problems.

Search Problems

Sort by: Solvers (High - Low)

925 Solvers
Times 2 - START HERE
Created by Cody Team
Tags: intro, math
Problem Group: Cody Challenge

795 Solvers
Make the vector [1 2 3 4 5 6 7 8 9 10]
Created by Cody Team
Tags: basic, matlab101, vectors
Problem Group: Cody Challenge

715 Solvers
Find the sum of all the numbers of the input vector
Created by Cody Team
Tags: easy, matlab101
Problem Group: Cody Challenge

659 Solvers
This problem is locked. You need to solve more problems from Cody Challenge group to unlock it.

601 Solvers
This problem is locked. You need to solve more problems from Cody Challenge group to unlock it.

http://www.mathworks.com/matlabcentral/cody
MathWorks online Webinars:

- Demonstration of features
- Introduction of new capabilities
- Application examples
- Live with chat discussion or recorded

http://www.mathworks.de/company/events/webinars/index.html
Basics of MATLAB

Introduction

Demos

MATLAB R2013a

Current Folder

File Folder

MATLAB Examples

Getting Started

MathWorks
The MathWorks offers introductory and intermediate courses in MATLAB®, Simulink®, Stateflow® and Code Generation products, as well as advanced training in specialized applications, such as signal processing, communications and control design.

**MATLAB®**
- MATLAB® Fundamentals
- MATLAB® Fundamentals for Automotive Applications
- MATLAB® Fundamentals for Aerospace Applications
- MATLAB® Fundamentals for Financial Applications

**Simulink®**
- Simulink® for System and Algorithm Modeling
- Simulink® for Automotive System Design
- Simulink® for Aerospace System Design

**Stateflow® - Event-Based Modeling**
- Stateflow® for Logic Driven System Modeling
- Stateflow® for Automotive Applications

**Physical Modeling**
- Physical Modeling of Multidomain Systems with Simscape
- Physical Modeling of Mechanical Systems with SimMechanics
- Physical Modeling of Electrical Power Systems with SimPowerSystems

**Application-Specific Trainings**
- Communications
  - Communication Systems Modeling with Simulink®
  - Communication Systems Modeling with MATLAB®
- Signal Processing
  - Signal Processing with MATLAB®
  - Image Processing with MATLAB®
- Image and Video Processing
  - Image Processing with MATLAB®
- Control System Design and Analysis
  - MATLAB® and Simulink® for Control Design Acceleration

**Code Generation**
- Rapid Prototyping and HIL-Simulation
  - Fundamentals of Code Generation for Real-Time Design and Testing
- Embedded Systems
  - Embedded Coder for Production Code Generation
- FPGA-Design
  - Generating HDL Code from Simulink®
- Model-Based Design
  - Simulink® Model Management and Architecture
  - Verification and Validation of Simulink® Models
- Code Integration
  - Integrating Code with Simulink®
- Code Verification
  - Polyspace Code Prover for C/C++ Code Verification

**Statistics**
- Statistical Methods in MATLAB®
- MATLAB® for Data Processing and Visualization

**Visualization**
- MATLAB® for Data Processing and Visualization

**Code Integration**
- Interfacing MATLAB® with C Code

**Optimization**
- MATLAB® Based Optimization Techniques

**Interactive Applications**
- Building Interactive Applications in MATLAB®

**Programming Techniques**
- MATLAB® Programming Techniques

**Distributed and Parallel Computing**
- Parallel Computing with MATLAB®

**Application Deployment**
- Deploying MATLAB® Based Applications – Java™ Edition
- Deploying MATLAB® Based Applications – .NET Edition
TUM Stud|Lab

- MATLAB student user group
- Lead by Max Schwenzer and Lucas Lebert
- Frequently meetings each Monday 13:00 – 14:00 in room MW3618

For Details:
matlab@fsmb.mw.tum.de
Outline
Introduction to MATLAB

Basics:
1) Introduction
2) MATLAB Basics
3) 2D and 3D Plots
4) Data Import and Export

Advanced:
1) Programming with MATLAB
2) Graphical User Interfaces in MATLAB

Toolboxes:
1) Symbolic Math Toolbox
2) Control System Toolbox and Curve Fitting Toolbox
MATLAB Basics

Exercise:

• Create a scalar variable: m, n

• Create a vector: b = (1x4), c = (4x1)

• Create a matrix: A (4x4)

• Change the variables
Workspace Browser

example_foo Dummy function for m-files
example_foo(a, B, c)
Subfunction demonstration - by the way, this is a 'cell'
sub_foo(in)

Workspace

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Size</th>
<th>Class</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4x4 double</td>
<td>4x4</td>
<td>double</td>
<td>128</td>
</tr>
<tr>
<td>b</td>
<td>[1 2 3 4]</td>
<td>1x4</td>
<td>double</td>
<td>32</td>
</tr>
<tr>
<td>c</td>
<td>[1;2;3;4]</td>
<td>4x1</td>
<td>double</td>
<td>32</td>
</tr>
<tr>
<td>m</td>
<td>5</td>
<td>1x1</td>
<td>double</td>
<td>8</td>
</tr>
<tr>
<td>n</td>
<td>-3</td>
<td>1x1</td>
<td>double</td>
<td>8</td>
</tr>
</tbody>
</table>
MATLAB Basics

Data and Variables

- Class
- Size
- Value
- Name (“variable”)

![Diagram showing data structures in MATLAB]

- Logical
- Character
- Numeric
- Table
- Cell
- Structure

Scalar

Function handle (@)

- int8, uint8,
- int16, uint16,
- int32, uint32,
- int64, uint64

Matrix or Array
(full or sparse)

- Single
- Double

Workspace Table:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>&lt;4x4 double&gt;</td>
<td>4x4</td>
</tr>
<tr>
<td>b</td>
<td>[1 2 3 4]</td>
<td>1x4</td>
</tr>
<tr>
<td>c</td>
<td>[1;2;3;4]</td>
<td>4x1</td>
</tr>
<tr>
<td>m</td>
<td>5</td>
<td>1x1</td>
</tr>
<tr>
<td>n</td>
<td>-3</td>
<td>1x1</td>
</tr>
</tbody>
</table>

Class:

- Double (for A, b, c, m, n)
MATLAB Basics

Data and Variables

- Class
- Size
- Value
- Name (“variable”)

\[ m \times n \]

\[ m \times n \times \ldots \times z \]
MATLAB Basics
Data and Variables

- Class
- Size
- Value
- Name ("variable")

```matlab
>> magic(4)

ans =

16  2  3  13
 5 11 10  8
 9  7  6 12
 4 14 15  1
```

![Workspace](image.png)
MATLAB Basics
Data and Variables

- Class
- Size
- Value
- Name ("variable")
Assignments

= assign a value to a variable
;
, separation of commands in one line

Reserved Variables

pi \( \pi \)
i, j \( \sqrt{-1} \)
inf infinity \( \infty \)
ans standard output of results (answer)
eps floating point accuracy
NaN Not a Number (invalid result)
### Mathematical Functions and Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ - * / ^</td>
<td>Operators</td>
<td>exp(x)</td>
<td>exponential function</td>
</tr>
<tr>
<td>mod(x, y)</td>
<td>x modulo y</td>
<td>log(x)</td>
<td>natural logarithm</td>
</tr>
<tr>
<td>rem(x, y)</td>
<td>remainder after division x/y</td>
<td>log10(x)</td>
<td>common log (basis 10)</td>
</tr>
<tr>
<td>sqrt(x)</td>
<td>square root √x</td>
<td>erf(x/√2)</td>
<td>normal distribution</td>
</tr>
<tr>
<td>abs(x)</td>
<td>absolute value</td>
<td>real(x)</td>
<td>real part</td>
</tr>
<tr>
<td>sign(x)</td>
<td>sign</td>
<td>imag(x)</td>
<td>imaginary part</td>
</tr>
<tr>
<td>round(x)</td>
<td>round</td>
<td>conj(x)</td>
<td>complex conjugate</td>
</tr>
<tr>
<td>ceil(x)</td>
<td>round up</td>
<td>angle(x)</td>
<td>phase of a complex value</td>
</tr>
<tr>
<td>floor(x)</td>
<td>round down</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Trigonometric Functions

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sin(x)</td>
<td>sine</td>
<td>tan(x)</td>
<td>tangent</td>
</tr>
<tr>
<td>cos(x)</td>
<td>cosine</td>
<td>cot(x)</td>
<td>cotangent</td>
</tr>
<tr>
<td>sind(x)</td>
<td>sine (x in degree)</td>
<td>atan(y/x)</td>
<td>arc tangent ± π/2</td>
</tr>
<tr>
<td>cosd(x)</td>
<td>cosine (x in degree)</td>
<td>atan2(y/x)</td>
<td>arc tangent ± π/2</td>
</tr>
</tbody>
</table>
Vectors and Matrices

\[
\begin{bmatrix}
x_1 & x_2 & \ldots & ; & x_3 & x_4 & \ldots
\end{bmatrix}
\]
input of matrices and vectors

\(x_1:x_2\)
creation of a line vector \([x_1 \ x_1+1 \ x_1+2 \ldots x_2]\)

\(x_1:d:x_2\)
creation of a line vector \([x_1 \ x_1+d \ x_1+2d \ldots x_2]\)

\(\text{linspace}(x_1,x_2,n)\)
line vector, start val \(x_1\), end val \(x_2\), size \(n\), equally distributed

\(\text{logspace}(x_1,x_2,n)\)
line vector, start val \(x_1\), end val \(x_2\), size \(n\), logarithmically distributed

\(\text{eye}(n)\)
\(n \times n\) identity matrix

\(\text{ones}(n)\)
\(n \times n\) matrix with all entries equal to 1

\(\text{zeros}(n)\)
\(n \times n\) matrix with all entries equal to 0

\(\text{rand}(x)\)
\(n \times n\) matrix with random entries between 0 and 1

\(\text{randn}(x)\)
\(n \times n\) matrix with normally distributed random entries

\(\text{magic}(x)\)
nxn matrix constructed from the integers 1 through \(n^2\) with equal row and column sums
Functions and Operators for Vectors and Matrices

- \(^* \, ^\backslash\) operators for matrices and vectors, left division
- \(.* \,.^\ .\) element wise operators
- \(\text{matrix } ' , \text{transpose(matrix)}\) transpose
- \(\text{matrix } ', \text{ctranspose(matrix)}\) Complex conjugate transpose
- \(\text{diff(vector[, n])}\) n-th difference between adjacent elements of vector
- \(\text{conv(vector1, vector2)}\) Convolution and polynomial multiplication

Additional functions

- \(\text{min(vec)}\) smallest vector element
- \(\text{max(vec)}\) largest vector element
- \(\text{mean(vec)}\) mean value
- \(\text{std(vec)}\) standard deviation
- \(\text{sum(vec)}\) sum of vector elements
- \(\text{prod(vec)}\) product of vector elements
- \(\text{diag(m)}\) diagonals of a matrix
- \(\text{inv(m)}\) matrix inverse
- \(\text{det(m)}\) matrix determinant
- \(\text{eig(m)}\) matrix eigenvalues
- \(\text{rank(m)}\) rank
- \(\text{cumsum(v)}\) cumulative sum
- \(\text{cumprod(v)}\) cumulative product
- \(\text{repmat}\) replicate and tile an array
- \(\text{sub2ind}\) Linear index from multiple subscripts
Structs and Cell Arrays

\[
\text{struct(}'n1', w1, 'n2', w2, \ldots)\]
create a struct variable

\[
\text{Structure.name}
\]
acess to the element name

\[
\text{CellArray} = \{\text{Value}\}
\]
creation of a Cell Array

\[
\text{CellArray}{\text{index}} = \text{Value}
\]
creation of a Cell Array

\[
\text{cell}(n)
\]
Creation of a n x n – Cell Array

\[
\text{cell}(m,n)
\]
Creation of a m x n – Cell Array
Managing Variables

- `size(variable)`
  - dimension of a variable
- `length(variable)`
  - length of a vector, largest dimension of a matrix
- `clear`
  - delete all variables in the workspace
- `clear all`
  - also deletes all global variables
- `clear [v1 v2 ...]`
  - delete selected variables
- `who`
  - list all variables that exist in the workspace
- `whos`
  - detailed list of all variables in the workspace with name, dimension, data type and size (memory)
- `clc`
  - clear command window
- `home`
  - moves MATLAB prompt to top of Command Window
# Relational Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>==</code></td>
<td>equal <code>eq(a,b)</code></td>
</tr>
<tr>
<td><code>~=</code></td>
<td>not equal <code>ne(a,b)</code></td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>less than <code>lt(a,b)</code></td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>less or equal than <code>le(a,b)</code></td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>greater than <code>gt(a,b)</code></td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>greater or equal than <code>ge(a,b)</code></td>
</tr>
</tbody>
</table>

# Logical Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>~</code></td>
<td>logical not <code>not(a)</code></td>
</tr>
<tr>
<td><code>&amp;</code></td>
<td>AND <code>and(a,b)</code></td>
</tr>
<tr>
<td>`</td>
<td>`</td>
</tr>
<tr>
<td><code>xor</code></td>
<td>exclusive OR <code>xor(a,b)</code></td>
</tr>
<tr>
<td><code>&amp;&amp;</code></td>
<td>shortcut AND <code>and(a,b)</code> (scalar)</td>
</tr>
<tr>
<td>`</td>
<td></td>
</tr>
</tbody>
</table>

# Additional Operators

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>all(vec)</code></td>
<td>each element is true</td>
</tr>
<tr>
<td><code>any(vec)</code></td>
<td>at least 1 element is true</td>
</tr>
<tr>
<td><code>logical(a)</code></td>
<td>type cast to boolean</td>
</tr>
<tr>
<td><code>exist('x')</code></td>
<td>existence of x</td>
</tr>
<tr>
<td><code>find(vec)</code></td>
<td>index of true elements</td>
</tr>
<tr>
<td><code>[~,~,a] = foo(x,y,z)</code></td>
<td>select only 3rd return value</td>
</tr>
</tbody>
</table>
Brackets

\{\}, ( ) or []

Cell Arrays
Indexing
Order of operations
Argument list
Matrix/Vector creation
Concatenation
Multiple outputs
>> path(genpath('..\Folder_Name'),path);
Exercise: Manipulating Data

1. Create a 4x3 matrix of random numbers
   - Extract the elements at locations 1,2 and 2,3
   - Extract the element in the lower right
   - Set every value < 0.5 to 0 (use logical indexing)

2. Create a diagonal matrix of size 4x4 with 3 on the diagonal

3. Solve $Ax = b$ for $A = \text{magic}(3)$ and $b = (1 \ 2 \ 3)$
   - Compute eigenvalues of $A$
Outline

Introduction to MATLAB

Basics:
1) Introduction
2) MATLAB Basics
3) 2D and 3D Plots
4) Data Import and Export

Advanced:
1) Programming with MATLAB
2) Graphical User Interfaces in MATLAB

Toolboxes:
1) Symbolic Math Toolbox
2) Control System Toolbox and Curve Fitting Toolbox
## 2D and 3D Plots

### Visualization Tools – 2D

<table>
<thead>
<tr>
<th>Line Graphs</th>
<th>Bar Graphs</th>
<th>Area Graphs</th>
<th>Direction Graphs</th>
<th>Radial Graphs</th>
<th>Scatter Graphs</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>plot</code></td>
<td><code>bar</code> (grouped)</td>
<td><code>area</code></td>
<td><code>feather</code></td>
<td><code>polar</code></td>
<td><code>scatter</code></td>
</tr>
<tr>
<td><code>plotyy</code></td>
<td><code>barchart</code> (grouped)</td>
<td><code>pie</code></td>
<td><code>quiver</code></td>
<td><code>rose</code></td>
<td><code>spy</code></td>
</tr>
<tr>
<td><code>loglog</code></td>
<td><code>barchart</code> (grouped)</td>
<td><code>fill</code></td>
<td><code>comet</code></td>
<td><code>compass</code></td>
<td><code>plotmatrix</code></td>
</tr>
<tr>
<td><code>semilogx</code></td>
<td><code>barchart</code> (grouped)</td>
<td><code>contourf</code></td>
<td></td>
<td><code>espolar</code></td>
<td></td>
</tr>
<tr>
<td><code>semilogy</code></td>
<td><code>hist</code></td>
<td><code>image</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>stairs</code></td>
<td><code>pareto</code></td>
<td><code>pcolor</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>contour</code></td>
<td><code>errorbar</code></td>
<td><code>escontourf</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>ezplot</code></td>
<td><code>stem</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>ezcontour</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
2D and 3D Plots
Visualization Tools – 3D

<table>
<thead>
<tr>
<th>Line Graphs</th>
<th>Mesh Graphs and Bar Graphs</th>
<th>Area Graphs and Constructive Objects</th>
<th>Surface Graphs</th>
<th>Direction Graphs</th>
<th>Volumetric Graphs</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>plot3</code></td>
<td><code>mesh</code></td>
<td><code>pie3</code></td>
<td><code>surf</code></td>
<td><code>quiver3</code></td>
<td><code>scatter3</code></td>
</tr>
<tr>
<td><code>contour3</code></td>
<td><code>meshc</code></td>
<td><code>fill3</code></td>
<td><code>surf1</code></td>
<td><code>comet3</code></td>
<td><code>coneplot</code></td>
</tr>
<tr>
<td><code>contourslice</code></td>
<td><code>meshs</code></td>
<td><code>patch</code></td>
<td><code>surfc</code></td>
<td><code>streamslice</code></td>
<td><code>streamline</code></td>
</tr>
<tr>
<td><code>ezplot3</code></td>
<td><code>ezmesh</code></td>
<td><code>cylinder</code></td>
<td><code>essurf</code></td>
<td></td>
<td><code>streamribbon</code></td>
</tr>
<tr>
<td><code>waterfall</code></td>
<td><code>stem3</code></td>
<td><code>ellipsoid</code></td>
<td><code>essurfc</code></td>
<td></td>
<td><code>streamtube</code></td>
</tr>
<tr>
<td></td>
<td><code>bar3</code></td>
<td><code>sphere</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>bar3h</code></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Graphics: 2D plot commands

plot([xvalues,] yvalues...[,plotstyle])  
stairs([xvalues,] yvalues...[,plotstyle]) 
bar(...), stem(...)  

loglog(xvalues, yvalues...[,plotstyle])  
semilogx(xvalues, yvalues...[,plotstyle])  
semilogy(xvalues, yvalues...[,plotstyle])  
polar(angle, radius...[,plotstyle])  

fplot(function, range)  
ezplot(function(x,y)[,range])  
ezplot(function1, function2[,range])  

hold [on | off]  

plot, linear axis  
plot, linear axis, stair step graph  
plot, linear axis, bars  
plot, logarithmic axis  
plot, logarithmic x-axis  
plot, logarithmic y-axis  
plot, polar coordinates  
plot, explicit function  
plot, implicit function  
plot, parametric curve  
retain current graph in figure
• Compute \( y = \sin(2t) + \cos(t) \) where \( t \) is from 1 to 10 seconds.

• Plot \( y \) and \( t \)
  \[
  >> \text{plot}(t, y);
  \]

• \[
  >> y\_1\_min = \text{min}(y);
  \]

• \[
  >> \text{plot}(t, y\_1\_min);
  \]

• \[
  >> \text{hold on};
  \]

• \[
  >> y\_1\_max = \text{max}(y);
  \]

• \[
  >> \text{plot}(t, y);
  \]

\( t \) and \( y \) are vectors!
2D and 3D Plots

Demo

```matlab
>> x = [0:0.2:20];

>> y = sin(x)./sqrt(x+1);

>> y(2,:) = sin(x/2)./sqrt(x+1);

>> y(3,:) = sin(x/3)./sqrt(x+1);

>> plot(x,y);
```

*y is a matrix!*
2D and 3D Plots

Plotting Tools

MATLAB Figure Window

Dock Figure in MATLAB Window

Plot Tools

data plots

y-axis

x-axis
2D and 3D Plots

Data Adjustment

- Basic Fitting
- Data Statistics
Plots Menu
2D and 3D Plots

Plotting from Workspace Browser

Context menu of variable or MATLAB menu opens
Plot Catalog
Graphics (general)

figure [(number)]
creation (call) of a figure

subplot (line, column, counter)
create a subplot

clf
clear current figure

close number
close (delete) figure number

close all
close (delete) all figures

gcf
current figure number (Handle)

gca
current subplot (Handle)

get(handle, 'property')
read object property

set(handle, 'property', value)
set property
2D and 3D Plots

Angermann et al. p. 44

Graphics : axis

axis([xmin, xmax, ymin, ymax])  
manual axis scaling (2D)

axis([x1,x2,y1,y2,z1,z2])  
manual axis scaling (3D)

axis(auto)  
automatic axis scaling

xlim([xmin,xmax])  
manual scaling of the x-axis

ylim([ymin,ymax])  
manual scaling of the y-axis

zlim([zmin,zmax])  
manual scaling of the z-axis

ggrid [on | off]  
ggrid lines on | off

zomm [on | off]  
zooming on | off

Graphics : labeling

xlabel(string)  
add x-axis label

ylabel(string)  
add y-axis label

zlabel(string)  
add z-axis label

title(string)  
create title

text(x, y, string)  
place a text on the graph

legend(string1, ... [, 'location', ...])  
create legend
### Colors

<table>
<thead>
<tr>
<th>Code</th>
<th>Color</th>
<th>Code</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>black</td>
<td>r</td>
<td>red</td>
</tr>
<tr>
<td>b</td>
<td>blue</td>
<td>m</td>
<td>magenta</td>
</tr>
<tr>
<td>c</td>
<td>cyan</td>
<td>y</td>
<td>yellow</td>
</tr>
<tr>
<td>g</td>
<td>green</td>
<td>w</td>
<td>white</td>
</tr>
</tbody>
</table>

### Markers

- . point
- o circle
- * asterisk
- +, x cross

### Lines

- - solid line (default)
- -- dashed line
- -. dash-dot line
- : dotted line
2D and 3D Plots

Angermann et al. p. 49

**Graphics : 3D plot commands**

\[
[X,Y] = \text{meshgrid}(x\text{vector}, \ y\text{vector})
\]

- rectangular coordinate grid matrix

\[
\text{plot3}(x\text{values}, y\text{values}, z\text{values}...[, \text{plotstyle}])
\]

- 3D-plot, points/lines

\[
\text{surf}(x\text{values}, y\text{values}, z\text{values}...[, \text{color}])
\]

- 3D-plot, surface

\[
\text{mesh}(x\text{values}, y\text{values}, z\text{values}...[, \text{color}])
\]

- 3D-plot, mesh

\[
\text{waterfall}(x\text{values}, y\text{values}, z\text{values}...[...])
\]

- 3D-plot, waterfall

\[
\text{contour}(x\text{values}, y\text{values}, z\text{values}...[...])
\]

- 2D-plot, contour lines/ level curves

- show box
- interactive rotating
- change perspective
- z-axis label

**Color settings**

\[
\text{colormap}(\text{name})
\]

- choose colormap

\[
\text{caxis}(\text{color\_min}, \ \text{color\_max})
\]

- color scaling
3D Plots

```matlab
>> [X,Y] = meshgrid(-10:0.25:10,-10:0.25:10);
>> f = sinc(sqrt((X/pi).^2+(Y/pi).^2));
>> mesh(X,Y,f);
>> axis([-10 10 -10 10 -0.3 1])
>> xlabel('{bfx}')
>> ylabel('{bfy}')
>> zlabel('{bfsinc} ({bfR})')
>> hidden off
```

3-d plot of a matrix!
Try: `>> size(f)`
Basics of MATLAB

3D Plots

```matlab
>> [X,Y] = meshgrid(-10:0.25:10,-10:0.25:10);
>> f = sinc(sqrt((X/pi).^2+(Y/pi).^2));
>> surf(X,Y,f);
>> axis([-10 10 -10 10 -0.3 1])
>> xlabel('{bfx}')
>> ylabel('{bfy}')</br>
>> zlabel('{bfsinc} ({bfR})')</br>
>> hidden off
```

Be careful with “copy – paste” of MATLAB plots into PowerPoint slides (file size)! Save plot as image before!
Outline
Introduction to MATLAB

Basics:
1) Introduction
2) MATLAB Basics
3) 2D and 3D Plots
4) Data Import and Export

Advanced:
1) Programming with MATLAB
2) Graphical User Interfaces in MATLAB

Toolboxes:
1) Symbolic Math Toolbox
2) Control System Toolbox and Curve Fitting Toolbox
Climatic Research Unit

The aim of the Climatic Research Unit (CRU) is to improve scientific understanding in:
- past climate history and its impact on humanity
- the course and causes of climate change during the present century
- prospects for the future

Global air temperature
2013 anomaly +0.49°C
(8th warmest on record)

Latest News (Read More):
- Winter is coming: British weather set to become more unsettled
- Avoiding overconfidence in climate projections
- Moira Lamb

www.cru.uea.ac.uk/
Exercise: Is the temperature rising?

1. Importing data from HadCRUT4.csv
2. Analyze data
3. Visualizing data as shown on slide before

- University of East Anglia, Norwich, UK, Climatic Research Unit
- Study on global warming
- Measurement series on combined global land and marine surface temperature record from 1850 to 2013
Data Import and Export

Exercise: Importing Data from .txt File

- "Import Data" icon in Workspace window
- Import Wizard
- Script Generation
- List of supported types
File import and export standard formats

load file [variable ...]
save file [variable ...]
[variable =] load file.ending
save file.ending -ascii [variable]
variable = xlsread('file.xls')
xlswrite('file.xls', variable)
load variables from MAT-File
save variables in MAT-File
load from ASCII-File
save variables in ASCII-File
load data from Excel-File
save data to Excel-File
Formatted data import and export

```matlab
fid = fopen('file.ending', 'permission')
fclose(fid)
fprintf(fid, 'format', variable[,,...])
vector = fscanf(fid, 'format')
string = fgetl(fid)
string = fgets(fid,n)
cellarray = textscan(fid, 'format'[, number][, 'parameter', value, ...])
variable = textread('file', 'format'[, 'parameter', value, ...])
variable = dlmread('file', 'delimiter'[, 'range'])
```

open file
close file
write formatted data
read formatted data
read line
read n characters
Binary data import and export

```matlab
vector = fread(fid, 'format')
fwrite(fid, matrix, 'format')
uchar, uint16, uint32, uint64
int8, int16, int32, int64
float32, float64
bitN, ubitN, 1<=N<=64
```

read data
write data
unsigned formats
signed formats
floating point formats
\( N \) signed or unsigned bits
Data Import and Export
Saving Data

- .mat files are used to store data
- .mat files are not human readable
- Content of .mat files is copied into workspace when opened
- Content can only be changed through editing in Workspace and re-saving

- .m files are used for MATLAB scripts and MATLAB functions
- .m files are plain text and can be edited with any text editor
- .m files can not be created from workspace (except for Simulink Bus Objects)
Data Import and Export
Managing Data

- native interface to version control systems like SVN or Git
- source control in current folder explorer
Data Import and Export

Publish Function
Outline
Introduction to MATLAB

Basics:
1) Introduction
2) MATLAB Basics
3) 2D and 3D Plots
4) Data Import and Export

Advanced:
1) Programming with MATLAB
2) Graphical User Interfaces in MATLAB

Toolboxes:
1) Symbolic Math Toolbox
2) Control System Toolbox and Curve Fitting Toolbox
Next steps:

- Using MATLAB Editor
- Executing MATLAB script
- Reusing MATLAB programs
Keywords in MATLAB

>> iskeyword

ans =

'break'
'case'
'catch'
'classdef'
'continue'
'else'
'elseif'
'end'
'for'
'function'
'global'
'if'
'otherwise'
...

... 'parfor'
'persistent'
'return'
'spmd'
'switch'
'try'
'while'
...
Conditional execution, control flow and loops

if ... [elseif ...][else] end

switch ... case ... [otherwise ...] end

for variable=start:stepsize:end
   commands end

while condition commands end

Additional intructions:

break
immediate termination of for or while loop
continue
immediate jump to the beginning of the next iteration step of a for or while loop
return
immediate return to invoking function
Scripts

... continuation sign for line breaks at too long lines
%
beginning of a comment text line
%
beginning of a comment as cell-divider
\%
{ comment \%} multiline comment
User dialog

```matlab
variable = input('string')
```
request user input for variable

```matlab
string = input('string, 's')
```
request user input of a string

```matlab
string = num2str(variable[, format])
```
convert number to string

```matlab
string = sprintf(string, variable)
```
create formatted string

```matlab
disp(string)
```
display text on screen

Escape characters

<table>
<thead>
<tr>
<th>Escape</th>
<th>Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>\n</td>
<td>line break</td>
<td>\n</td>
</tr>
<tr>
<td>\t</td>
<td>tabulator</td>
<td>\t</td>
</tr>
<tr>
<td>\</td>
<td>backslash</td>
<td>\</td>
</tr>
<tr>
<td>%</td>
<td>percent sign</td>
<td>%</td>
</tr>
<tr>
<td>'</td>
<td>single quotation mark</td>
<td>'</td>
</tr>
</tbody>
</table>

Formatting (conversion characters)

<table>
<thead>
<tr>
<th>Conversion Character</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>%d</td>
<td>signed integer (i.e. 321)</td>
</tr>
<tr>
<td>%x</td>
<td>base 16 (hexadecimal) whole number</td>
</tr>
<tr>
<td>%5.2f</td>
<td>floating point number (i.e. 54.21)</td>
</tr>
<tr>
<td>%.2e</td>
<td>exponential notation (i.e. 5.42e+001)</td>
</tr>
<tr>
<td>%s</td>
<td>string</td>
</tr>
</tbody>
</table>
Operating System calls and file management

- `cd folder` change directory
- `pwd` show current directory
- `dir [name]` list folder contents
- `ls [name]` list folder contents
- `mkdir folder` create new folder
- `copyfile source destination` copy file
- `delete file` delete file
- `! command` operating system command
- `system(command)` operating system command with return values
- `eval(string)` interpret `string` as MATLAB command
• Why?
  – Automating
  – Editing/Debugging
  – Deploying as applications

MATLAB Scripts

MATLAB Functions
Basics of MATLAB

Script Example

```matlab
>> close all
>> clear all
>> clc

>> disp 'Adjusting path'
>> path(genpath('../'Folder_Name'),path);

>> disp 'Running Init Files'
>> run('My_MATLAB_Script')

>> A = ones(5); %Initialization of Variables

>> disp 'Init completed, open Simulink Model'
>> open('My_Simulink_Model.mdl')
```
Exercise

- Plot a sine wave $y = \sin(t)$, $t=[0:0.1:2\cdot\pi]$ 
- Use for loop to create animation
- Save MATLAB script as `sine_wav_anim.m`
• Why?
  – Automating
  – Editing/Debugging
  – Deploying as applications
%% example_foo Dummy function for m-files
%%
%-----------------------------------------------------------
% function [ x, Y, z ] = example_foo( a, B, c )
%-----------------------------------------------------------
%#codegen
%
% My helping text about this function
%
% %COPYRIGHT © 2010, Technische Universität München (TUM)

%Everything until here is shown in F1 help

function [ x, Y, z ] = example_foo( a, B, c )

% Main algorithms
[~,nB] = size(B);

% Assign output values
x = 'This is the output "x".'; % Just a random string.

% Now, Y is calculated. Of course, this is just a dummy calculation
Y = ones( length(x), nB );

z = sub_foo(2) + nest_demo_foo(a) + c; % Demo on how to use a subfunction
end % Always finish function with 'end'

%% Subfunction demonstration - by the way, this is a 'cell'
% This function is not visible to code outside this m-file. It only serves
% for structuring the current file.

function [res] = sub_foo(in)

res = 2 * in;

end

%% Nested function demonstration
%TODO: Add description here

function [res] = nest_demo_foo(m)

nest_foo(m);

    function nest_foo(n) %#ok Just for Demo
        res = 2*n;
    end

end

% --- EOF ---
Functions

function [out] = name(in)

definition of MATLAB function name with list of input
parameters in and output values out

nargin, nargsout

number of input / output parameters

nargchk(min, max, n)

check number n of function parameters, if

min <= n <= max, otherwise raise an error

isempty(‘name’)

determine if variable name is empty

error(‘info’)

terminate function execution and display error message

info

warning(‘info’)

show warning in command window (warnings can be
disabled)
Global and static variables in functions

```
persistent var1 ...
global var1 ...
clear global var1 ...
assignin('base', 'var', x)
```

- `persistent var1 ...`: define static (local) variable
- `global var1 ...`: define global variable
- `clear global var1 ...`: delete global variable
- `assignin('base', 'var', x)`: assign the value `x` to the variable `var` in the workspace of the command line (base workspace)
Functions, Subfunctions, Nested Functions and Workspaces

• File name has to be the same like the primary function in this file because MATLAB is searching for files, not for functions

• Functions can call subfunctions within one file
• Subfunctions can call each other within one file
• Each function and subfunction has its own workspace different from base workspace

• Nested functions can be called from the level immediately above, from a function at the same level within the same parent and a nested function at any lower level
• Nested functions still have their own workspace BUT:
  – An inner function can access the workspace of all outer functions
  – An outer function can access local variables of all inner functions

Never name variables like functions!
Never name functions like MATLAB default functions
Programming with MATLAB

Optimizing Performance

- `tic; code; toc;` determines code execution time
- Supports optimization of code
- Preallocation of memory
  Although it’s not required, preallocating memory can increase computation speed for big data
- Vectorization
  MATLAB is optimized for vector and matrix operations
- TODO / FIXME report
Outline
Introduction to MATLAB

Basics:
1) Introduction
2) MATLAB Basics
3) 2D and 3D Plots
4) Data Import and Export

Advanced:
1) Programming with MATLAB
2) Graphical User Interfaces in MATLAB

Toolboxes:
1) Symbolic Math Toolbox
2) Control System Toolbox and Curve Fitting Toolbox
MATLAB GUIs

- MATLAB GUIs visualize, control or manipulate variables, functions or Simulink Models

- GUIs always consist of two elements: a figure file .fig and a code file .m (e.g. myfigure.fig and myfigure.m)

- GUIs can be written by hand or be generated by GUI editor GUIDE

- Demo: >> graf3d
Graphical User Interfaces in MATLAB

Using GUI Editor GUIDE
Graphical User Interfaces in MATLAB

Property Inspector

1. Double Click on Element

2. Property Inspector
Graphical User Interfaces in MATLAB

Figures and Callback Functions
• Write data to MATLAB Workspace:
  
  \[ \text{assignin('base', 'Name', Value);} \]

• Read data from MATLAB Workspace:
  
  \[ \text{evalin('base', 'Name')} \]

• Use data within GUI (e.g. In Edit Box):
  
  \[ \text{set(handles.edit,'String','Value');} \]
  
  \[ \text{get(hObject,'Value');} \]

• Transmit Data to Simulink (e.g. Constant Block):
  
  \[ \text{set_param('Simulink_Model/Constant','value',...} \]
  
  \[ \text{num2str(get(hObject,'Value')));} \]

• Receive Data from Simulink (e.g. Constant Block):
  
  \[ \text{get_param('Simulink_Model/Constant','value');} \]
Outline Day 1
Introduction to MATLAB

Basics:
1) Introduction
2) MATLAB Basics
3) 2D and 3D Plots
4) Data Import and Export

Advanced:
1) Programming with MATLAB
2) Graphical User Interfaces in MATLAB

Toolboxes:
1) Symbolic Math Toolbox
2) Control System Toolbox and Curve Fitting Toolbox
• Symbolic Math solves the algorithm without numerical discretization (numerical deviation)

• Not all problems have an analytical solution (e.g. Navier Stokes Equations), in this case numerical methods are required

• Symbolic Math Toolbox is fully integrated with MATLAB, Simulink and Simscape, allowing analytical solutions to be directly used in other applications (e.g. useful for control systems)

• Symbolic Math Toolbox is developed and maintained at University of Paderborn

• Graphical Editor: >> mupad
Mass-Spring-Damper System

An ideal mass-spring-damper system with mass $m$ (in kg), spring constant $k$ (in N/m) and damping coefficient $B$ (in N·s/m) can be described with the following formula:

$$m \ddot{x}(t) + k \dot{x}(t) + B \dot{x}(t)$$

MuPAD the following syntax is used to define and solve this differential equation:

```plaintext
m := m; x := x(t); B := B;

eqn := m*x''(t) + k*x(t) + B*x'(t);

f := ode([eqn, x(0) = 0, x'(0) = 1], x(t));

m*x''(t) + k*x'(t) + B*x(t)

ode(x'(0) = 1, x(0) = 0, m*x''(t) + k*x'(t) + B*x(t), x(t))

general_solution := solve(f, IgnoreSpecialCases)
```

$$\left\{ \frac{m}{2} \left( k \sqrt{k^2 - 4m} \right) e^{\frac{k}{2m} t} - \frac{m}{2} \left( k \sqrt{k^2 - 4m} \right) e^{\frac{-k}{2m} t} \right\}$$
You can animate the solution using the various plot features MuPAD offers:

```matlab
2D Plot

plotfun2d(mupad_solution, t=0..50)
```

### 3D animation

```matlab
line := plot::Line3d([0, 0, 0],[mupad_solution=1, 0, 0.1], t=0..50, LineWidth = 1, LineColor = RGB::Black);
surf := plot::Surface([u, v, 0], u = 0..2, v = -1..1, FillColorDiffusion = 1);
box := plot::Box([mupad_solution=0.75, -0.25, 0], [mupad_solution=1.25, 0.25,0.25], t=0..50, FillColor = RGB::Grey([0.2]));
plot(plot::CoordinateSystem3d(line, surf, box,
    ViewingBoxMin = 0, ViewingBoxMax = 0.5, ViewingBoxMin = -1, ViewingBoxMax = 1, ViewingBoxMin = 0, ViewingBoxMax = 2),
    GridVisible = TRUE, GridLineStyle = Dashed);
```
Symbolic Math Toolbox
MuPAD 3D Animations

Basics of MATLAB
• MATLAB and symbolic engine have separate workspaces
• Each notebook also has a separate workspace
Symbolic Math Toolbox
Export MuPAD Function to MATLAB

• Create handle to new Notebook:  \texttt{>> h = mupad;}  
• Get function from Notebook:  \texttt{>> y = getVar(h,'general_solution');}  
• Convert symbolic expression to function handle or to file:
  \texttt{>> f = matlabFunction(y);}  
  \texttt{>> f = matlabFunction(y, 'file', 'C:\myFctName');}
Functions available in the Notebook interface can be called directly from the MATLAB command line.

Using `evalin`, it is possible to evaluate a MuPAD expression and return the results to MATLAB.

Using `feval`, it is possible to pass symbolic variables that exist in the MATLAB workspace, and these variables are evaluated before being processed in the symbolic engine.

Creating an Embedded Matlab Block in Simulink:

```matlab
>> new_system('sys')
>> emlBlock('sys/new_block',y)
>> open_system('sys')
```
Outline
Introduction to MATLAB

Basics:
1) Introduction
2) MATLAB Basics
3) 2D and 3D Plots
4) Data Import and Export

Advanced:
1) Programming with MATLAB
2) Graphical User Interfaces in MATLAB

Toolboxes:
1) Symbolic Math Toolbox
2) Control System Toolbox and Curve Fitting Toolbox
Toolbox Demos
Outline

Introduction to MATLAB

Basics:
1) Introduction
2) MATLAB Basics
3) 2D and 3D Plots
4) Data Import and Export

Advanced:
1) Programming with MATLAB
2) Graphical User Interfaces in MATLAB

Toolboxes:
1) Symbolic Math Toolbox
2) Control System Toolbox and Curve Fitting Toolbox
Object-Oriented Programming

```matlab
classdef topo < handle % Plots function of 2 vars
    properties
        FigHandle % Store figure handle
        FofXY; % Function handle to fcn being evaluated
        Lm = [-2*pi 2*pi]; % Default range if not specified
    end

    properties (Dependent = true, SetAccess = private)
        Data % Data property depends on current value of FofXY
    end

    methods
        function obj = topo(fnc,limits)
        obj.FofXY = fnc;
        obj.Lm = limits;
    end

        function set.Lm(obj,lim)
        % Lm property set after checking limits
        if ~(lim(1) < lim(2))
            disp('Bad limits, using [-2pi 2pi]')
        else
            obj.Lm = lim;
        end
    end % set.Lm

        function data = get.Data(obj)
        % Data object get function calculates data
    end
end
```

tobj = topo(@(x,y) x.*exp(-x.^2-y.^2),[-2 2]);
a = tobj;
surfl(a) % Call class method to create a graph

MATLAB help -> MATLAB -> Advanced Software Development -> Object-Oriented Programming -> Object oriented Design with MATLAB
Additional Information

Big Data – MapReduce – Hadoop

Domestic airline flights per day per carrier

Process "Big Data" in MATLAB using MapReduce

Date

Flights per day (7-day moving average)


0 500 1000 1500 2000 2500 3000 3500

DL WN AA US UA NW CO
External Interfaces

- Shared libraries (.dll, .so, .dylib)
- C, C++, Fortran interface
- C, Fortran MEX-files (.mex)
- Sun Java classes
- COM/.NET support
- Web services
- Serial Port and other hardware I/O (soft real time)
Summary

- MATLAB is a **high level-language for technical computing**
- Interactive tool with **mathematical and graphical** functions
- MATLAB provides features to **access, compute, analyze and visualize data**
- MATLAB also provides capabilities to **interface with external languages**
Contact for further information or feedback about this course:

Dipl.-Ing. Markus Hornauer  
Institute of Flight Systems Dynamics  
Boltzmannstr. 15  
85748 Garching, Germany  
Tel: +49 (0)89 289 16047  
Fax: +49 (0)89 289 16058  
Email: markus.hornauer@tum.de

Markus Hornauer  
High Integrity Systems Engineer  
Französische Str. 13-14  
D – 10117 Berlin  
+49 151 23506683  
markus.hornauer@samoconsult.de