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

Installing MATLAB

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/

MATLAB® & SIMULINK®

Use the latest version of MATLAB®, Simulink®, and other related products to support your course work and research.

Exhibition & Info

Wednesday, December 14, 2011
09:00 – 17:00 hrs
TUM Maschinenwesen
Hof 1 (close to Main Entrance)

Technische Universität München

TUM offers free licenses of MATLAB, Simulink, and companion products to students and employees.

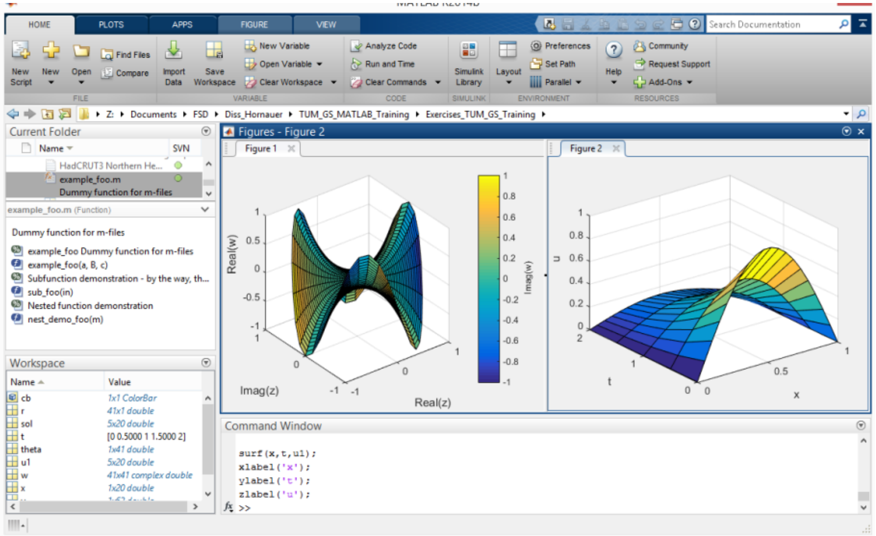
For Download and Activation:
https://matlab.rbg.tum.de

Free MATLAB & Simulink TUTORIALS:
www.mathworks.com/academia/student_center/tutorials

Join the MathWorks user community at
mathworks.com/matlabcentral

MathWorks®
Accelerating the pace of engineering and science

MATLAB - The Language for Technical Computing



Dummy function for m-files

- example_foo Dummy function for m-files
- example_foo(a, B, c)
- Subfunction demonstration - by the way, th...
- sub_foo(in)
- Nested function demonstration
- nest_demo_foo(m)

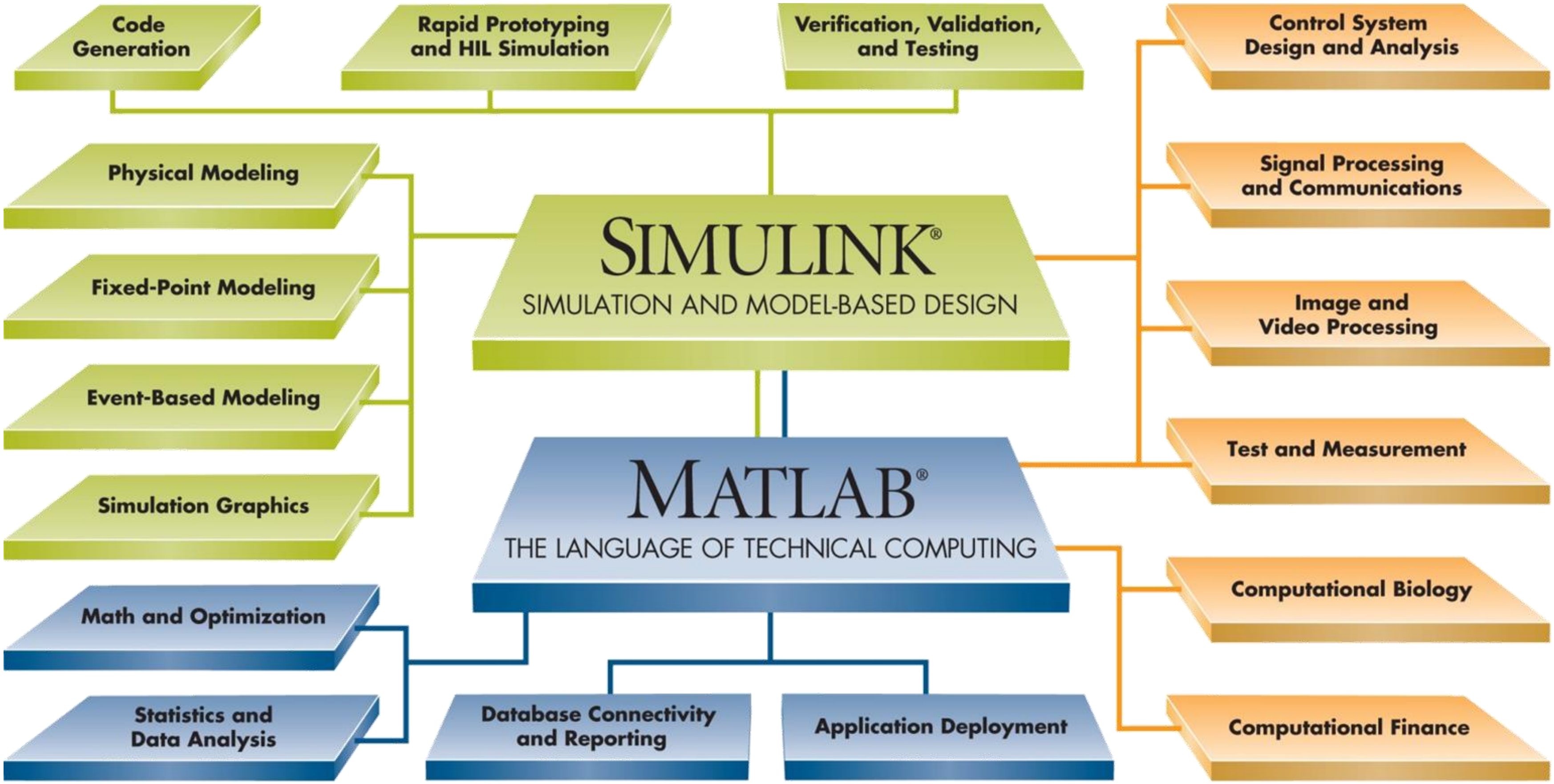
Name	Value
cb	1x1 ColorBar
r	41x1 double
sol	5x20 double
t	[0 0.5000 1 1.5000 2]
theta	1x41 double
u1	5x20 double
w	41x41 complex double
x	1x20 double

```
surf(x,t,u1);
xlabel('x');
ylabel('t');
zlabel('u');
>>
```

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

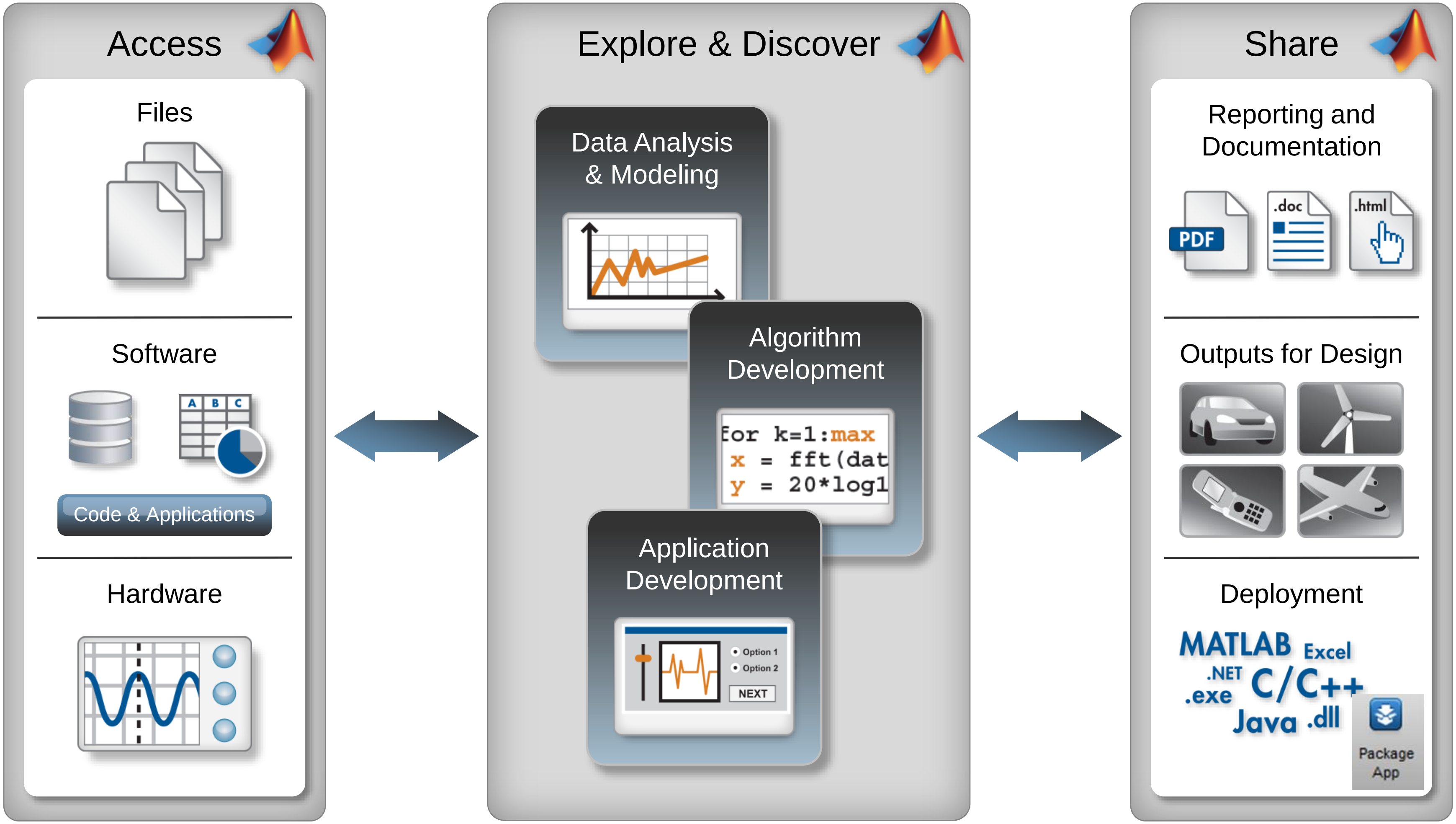
The MathWorks Products



Code Generation
Rapid Prototyping and HIL Simulation
Verification, Validation, and Testing
Control System Design and Analysis
Physical Modeling
Signal Processing and Communications
SIMULINK®
SIMULATION AND MODEL-BASED DESIGN
Fixed-Point Modeling
Image and Video Processing
Event-Based Modeling
Test and Measurement
MATLAB®
THE LANGUAGE OF TECHNICAL COMPUTING
Simulation Graphics
Math and Optimization
Computational Biology
Statistics and Data Analysis
Database Connectivity and Reporting
Application Deployment
Computational Finance

www.mathworks.com/products/

Technical Computing Workflow



Access

Files

Software

Code & Applications

Hardware

Explore & Discover

Data Analysis & Modeling

Algorithm Development

```
for k=1:max
 x = fft(dat
 y = 20*log1
```

Application Development

Share

Reporting and Documentation

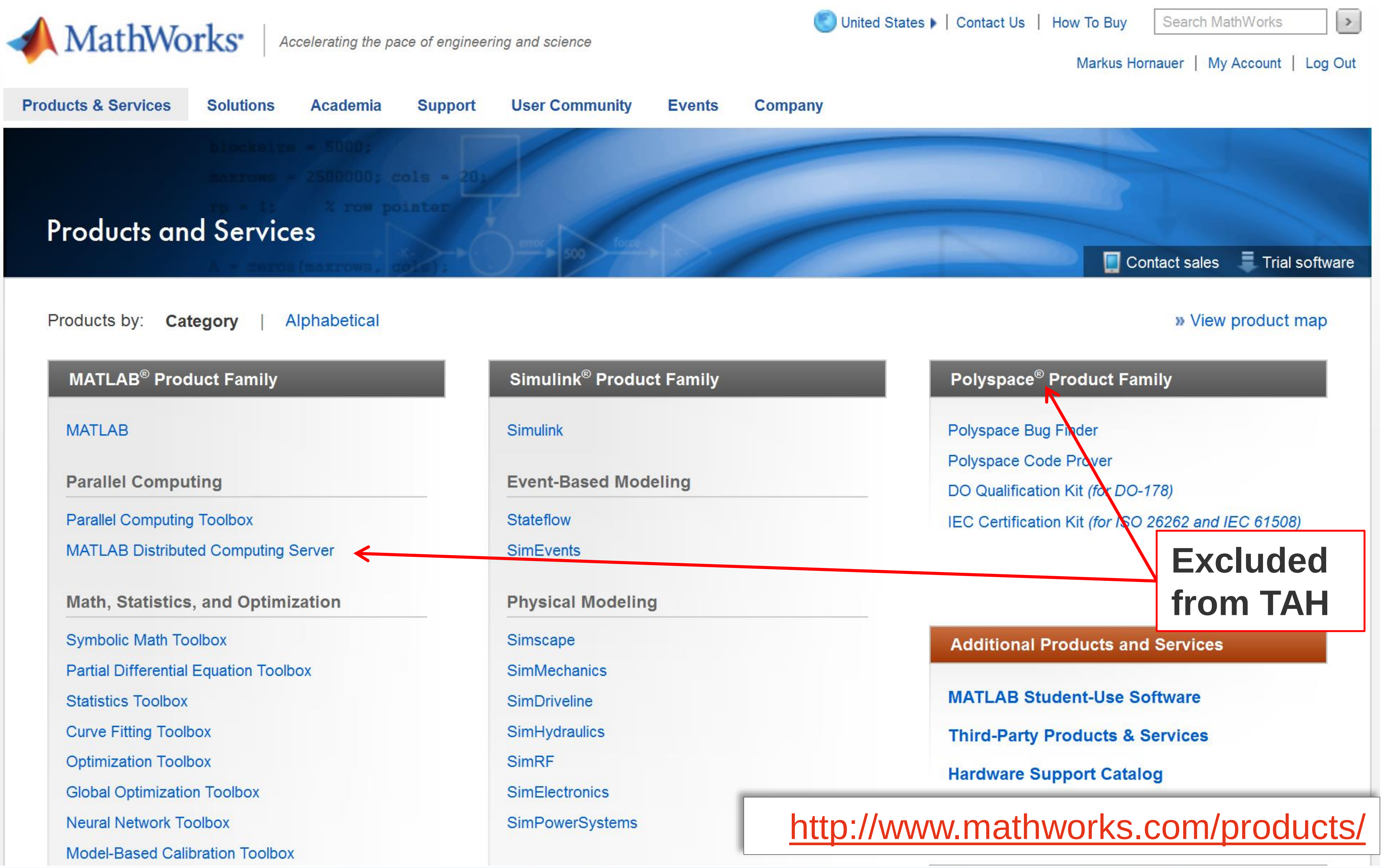
Outputs for Design

Deployment

MATLAB Excel .NET .exe C/C++ Java .dll

Package App

MATLAB Product Family



Products and Services

Products by: **Category** | Alphabetical

» View product map

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

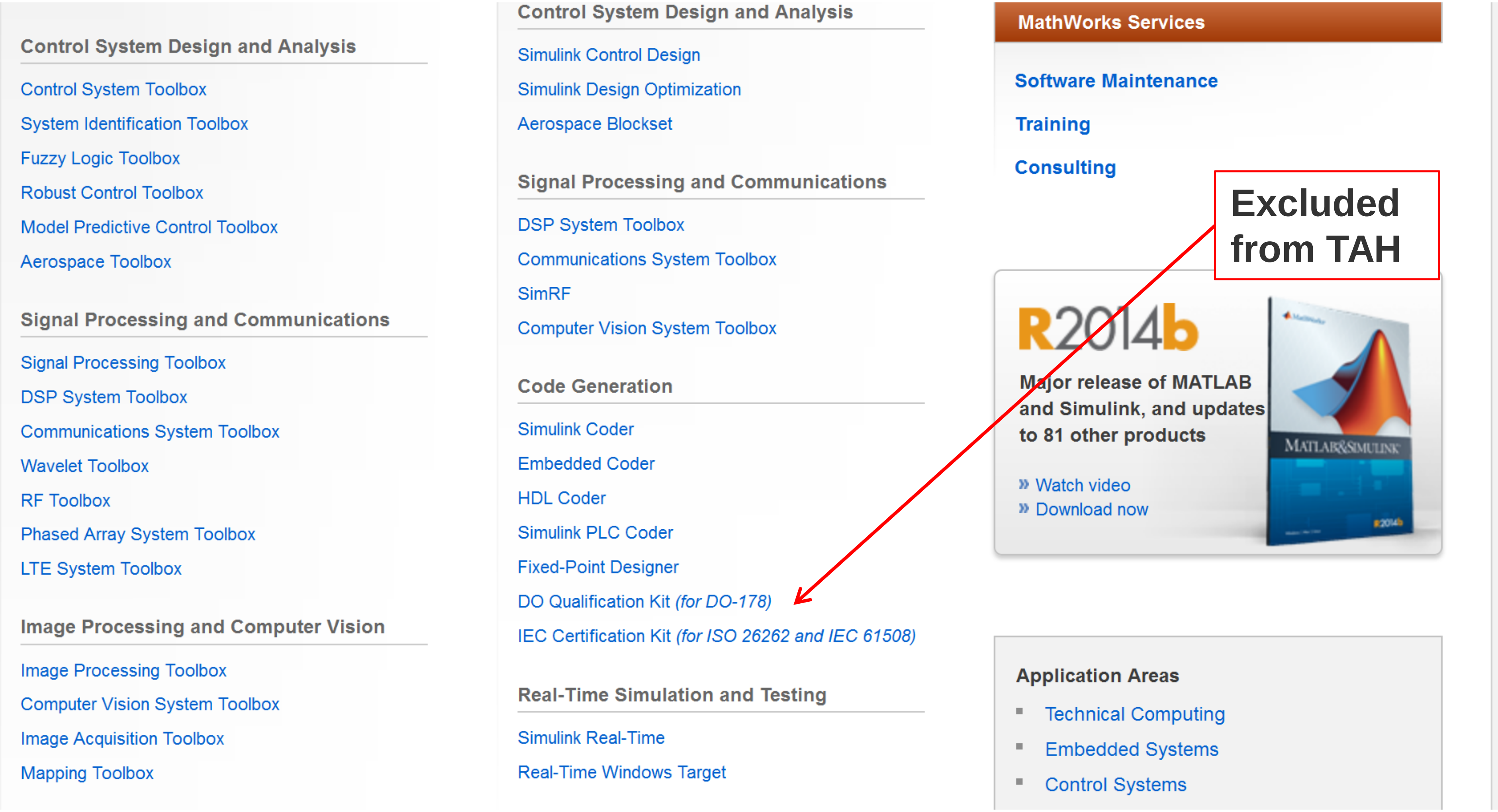
Third-Party Products & Services

Hardware Support Catalog

Excluded from TAH

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

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

R2014b

Major release of MATLAB and Simulink, and updates to 81 other products

» Watch video
» Download now

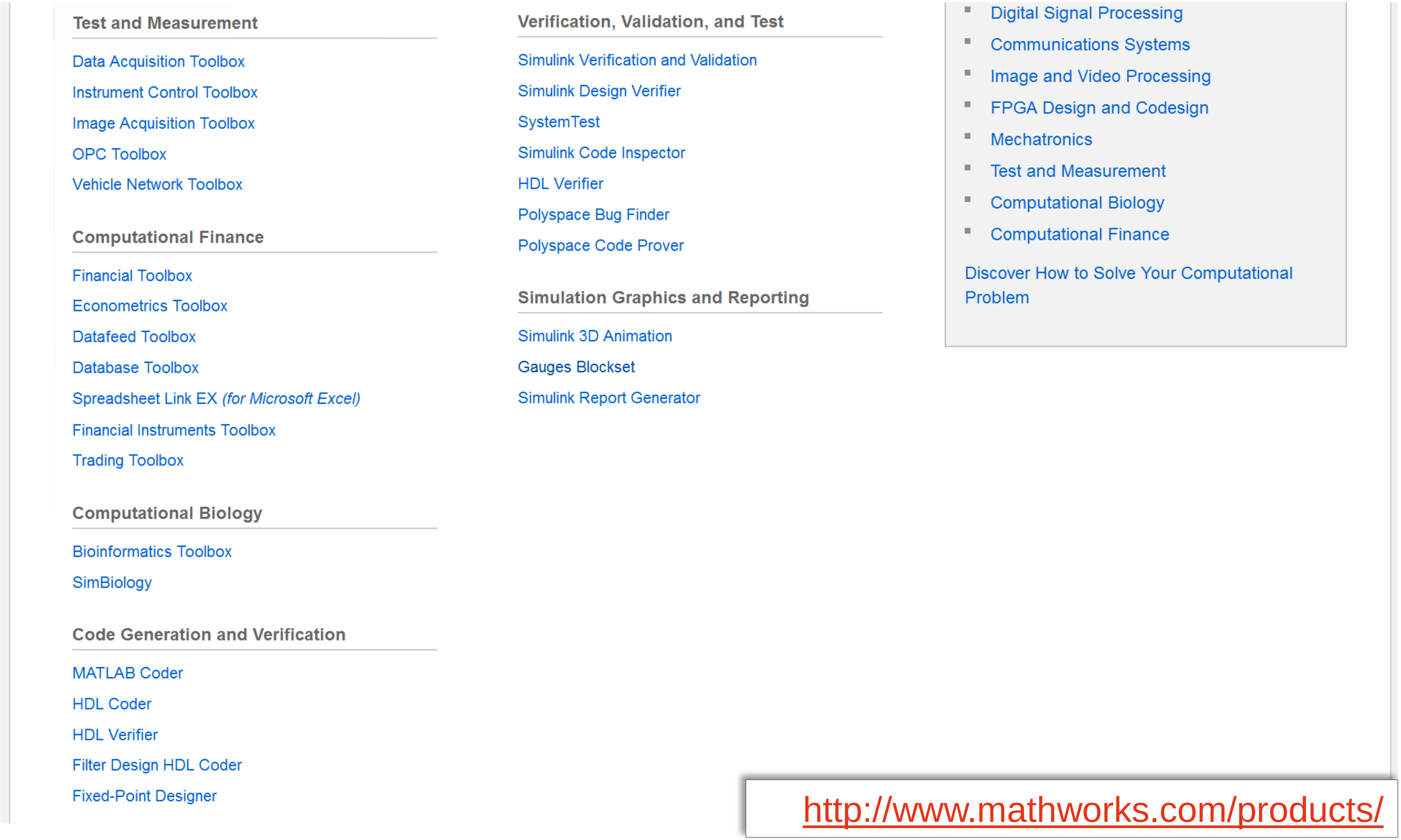
Application Areas

- Technical Computing
- Embedded Systems
- Control Systems

Excluded from TAH

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

MATLAB Product Family



Test and Measurement

- Data Acquisition Toolbox
- Instrument Control Toolbox
- Image Acquisition Toolbox
- OPC Toolbox
- Vehicle Network Toolbox

Computational Finance

- Financial Toolbox
- Econometrics Toolbox
- Datafeed Toolbox
- Database Toolbox
- Spreadsheet Link EX *(for Microsoft Excel)*
- Financial Instruments Toolbox
- Trading Toolbox

Computational Biology

- Bioinformatics Toolbox
- SimBiology

Code Generation and Verification

- MATLAB Coder
- HDL Coder
- HDL Verifier
- Filter Design HDL Coder
- Fixed-Point Designer

Verification, Validation, and Test

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

Simulation Graphics and Reporting

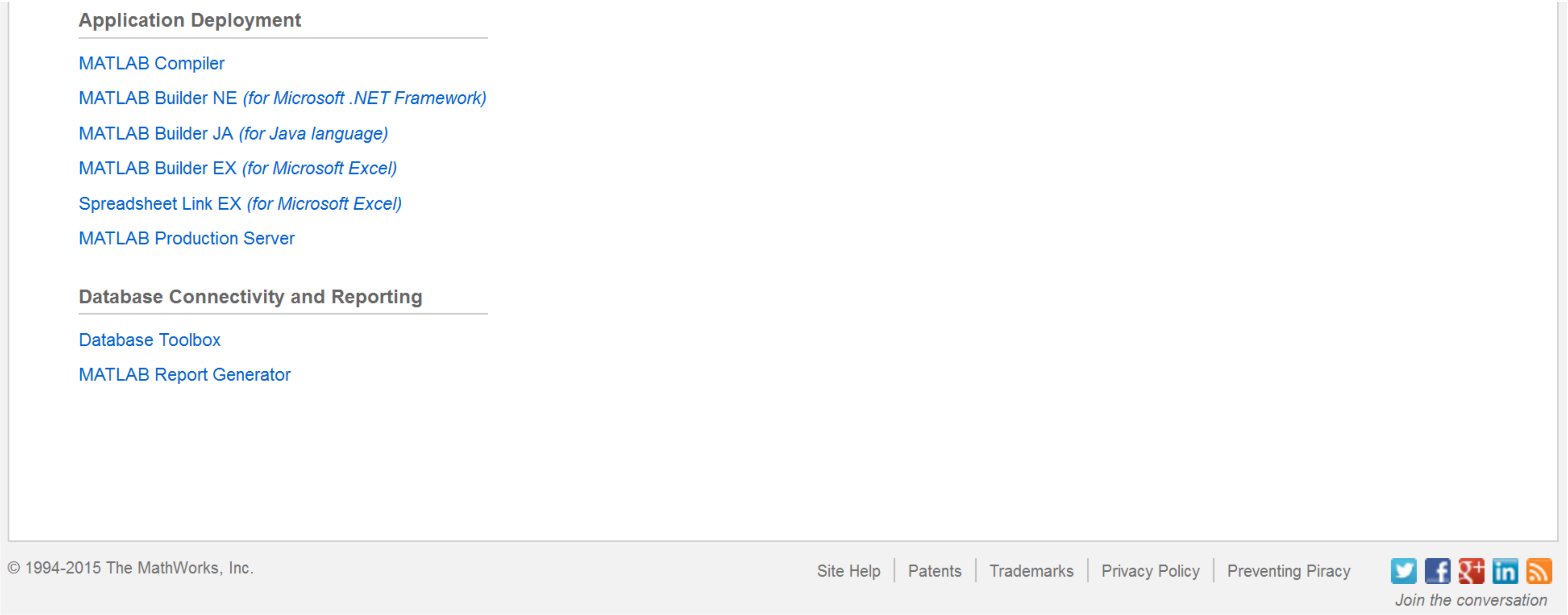
- Simulink 3D Animation
- Gauges Blockset
- Simulink Report Generator

- 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

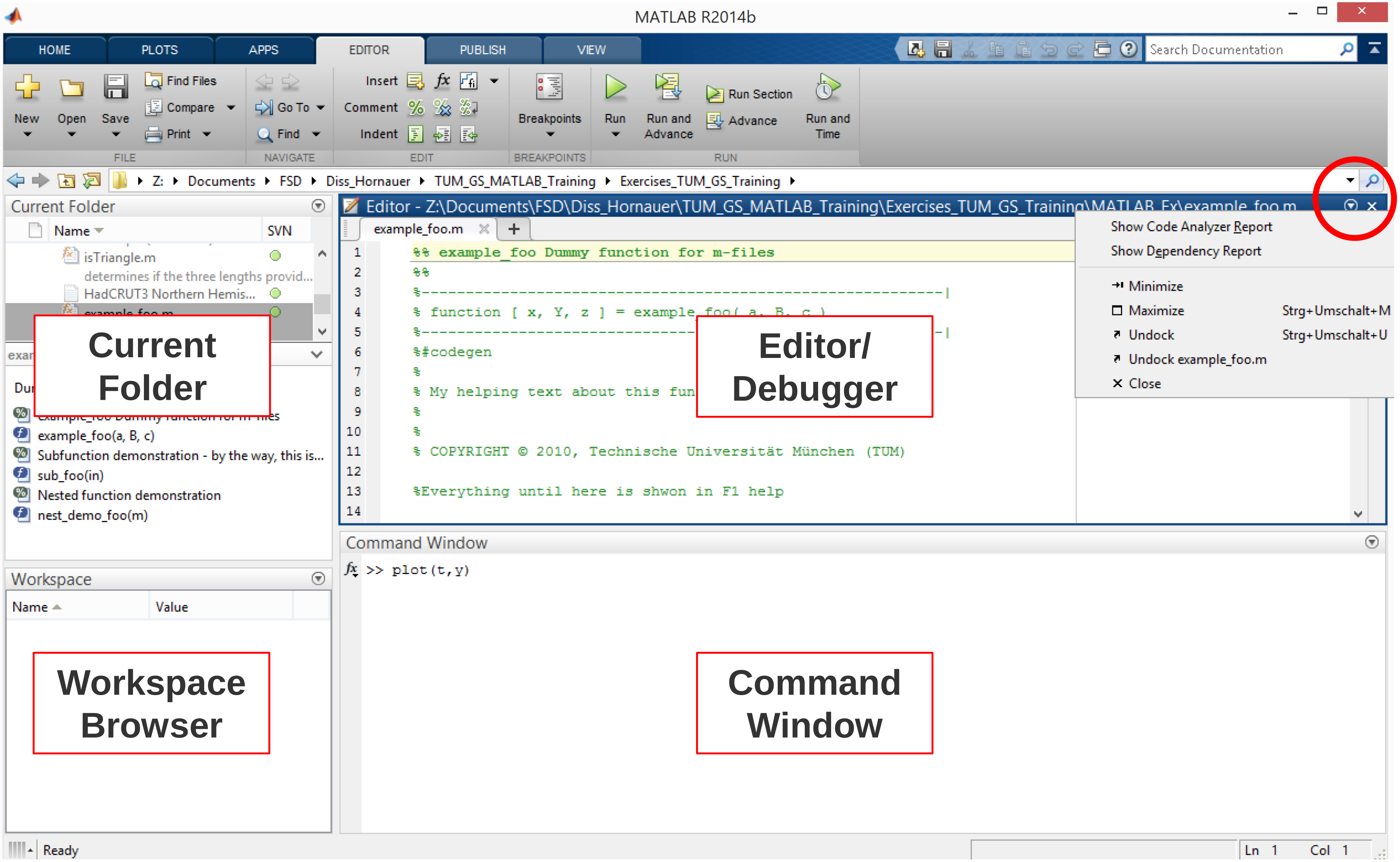
Database Connectivity and Reporting

- Database Toolbox
- MATLAB Report Generator

Site Help | Patents | Trademarks | Privacy Policy | Preventing Piracy
Join the conversation

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

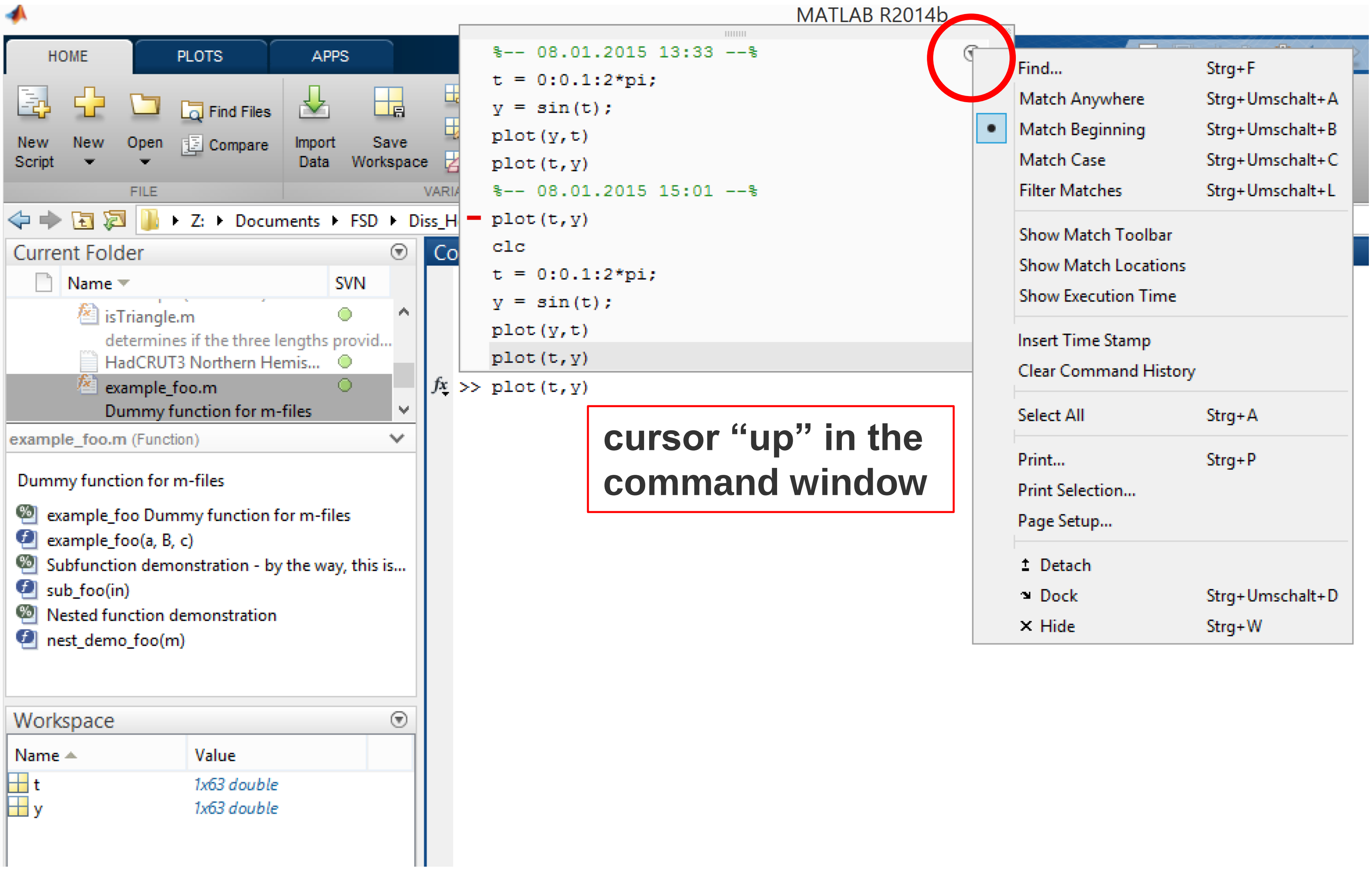
MATLAB Desktop



```
%% example_foo Dummy function for m-files
%%
%-----------------------------------------------------------|
% function [ x, Y, z ] = example_foo( a, B, c )
%--------------------------------                          -|
%#codegen
%
% My helping text about this fun
%
%
% COPYRIGHT © 2010, Technische Universität München (TUM)

%Everything until here is shwon in F1 help

```

```
>> plot(t,y)
```

MATLAB Command History



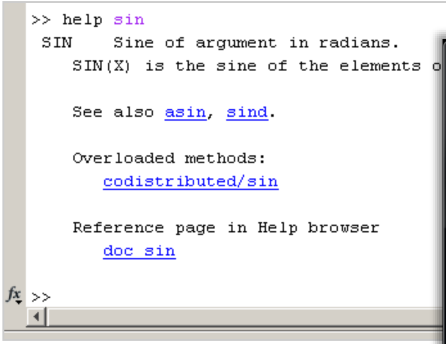
```
%-- 08.01.2015 13:33 --%
t = 0:0.1:2*pi;
y = sin(t);
plot(y,t)
plot(t,y)
%-- 08.01.2015 15:01 --%
plot(t,y)
clc
t = 0:0.1:2*pi;
y = sin(t);
plot(y,t)
plot(t,y)
```

```
>> plot(t,y)
```

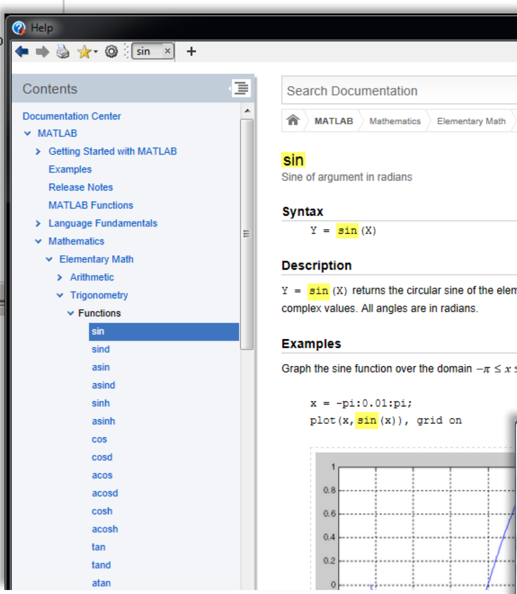
Name	Value
t	1x63 double
y	1x63 double

cursor “up” in the command window

Getting Help



```
>> help sin
 SIN    Sine of argument in radians.
    SIN(X) is the sine of the elements o

    See also asin, sind.

    Overloaded methods:
       codistributed/sin

    Reference page in Help browser
       doc sin

>>
```



sin

Sine of argument in radians

Syntax

```
Y = sin(X)
```

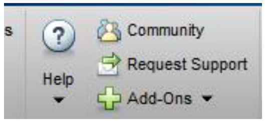
Description

`Y = sin(X)` returns the circular sine of the elements of X. The `sin` function ope
complex values. All angles are in radians.

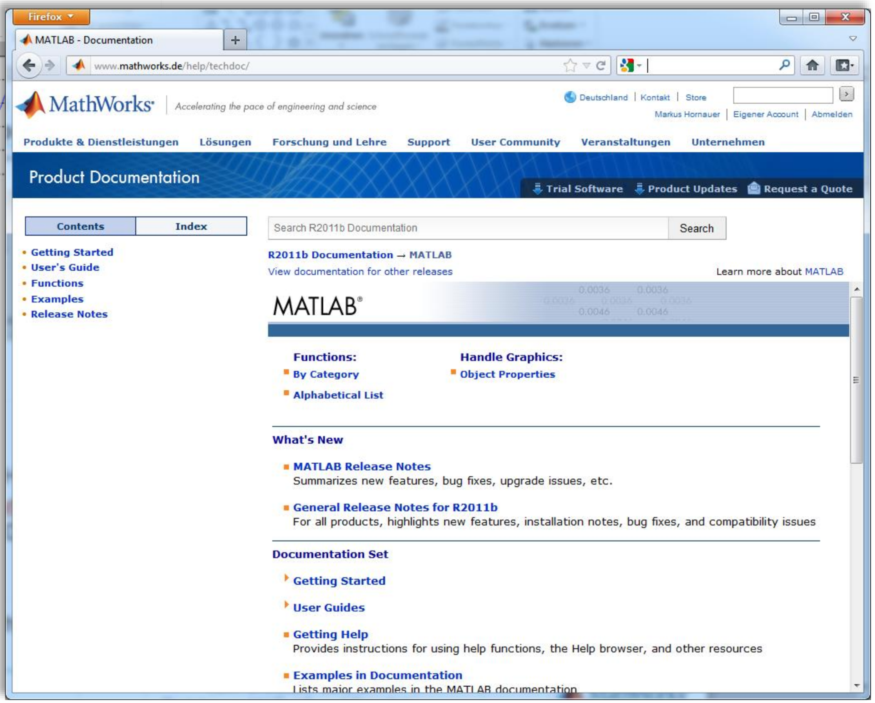
Examples

Graph the sine function over the domain $-\pi \le x \le \pi$.

```
x = -pi:0.01:pi;
plot(x,sin(x)), grid on
```

- >>doc
- http://mathworks.de -> Support -> Product Documentation
- 

- >>help
- Search the web
- F1



Firefox
MATLAB - Documentation
www.mathworks.de/help/techdoc/
MathWorks
Accelerating the pace of engineering and science
Deutschland | Kontakt | Store
Markus Hornauer | Eigener Account | Abmelden
Produkte & Dienstleistungen
Lösungen
Forschung und Lehre
Support
User Community
Veranstaltungen
Unternehmen
Trial Software
Product Updates
Request a Quote
Contents
Index
Search R2011b Documentation
Search
MATLAB

Product Documentation

- Getting Started
- User's Guide
- Functions
- Examples
- Release Notes

R2011b Documentation → MATLAB

View documentation for other releases

Learn more about MATLAB

Functions:
- By Category
- Alphabetical List

Handle Graphics:
- Object Properties

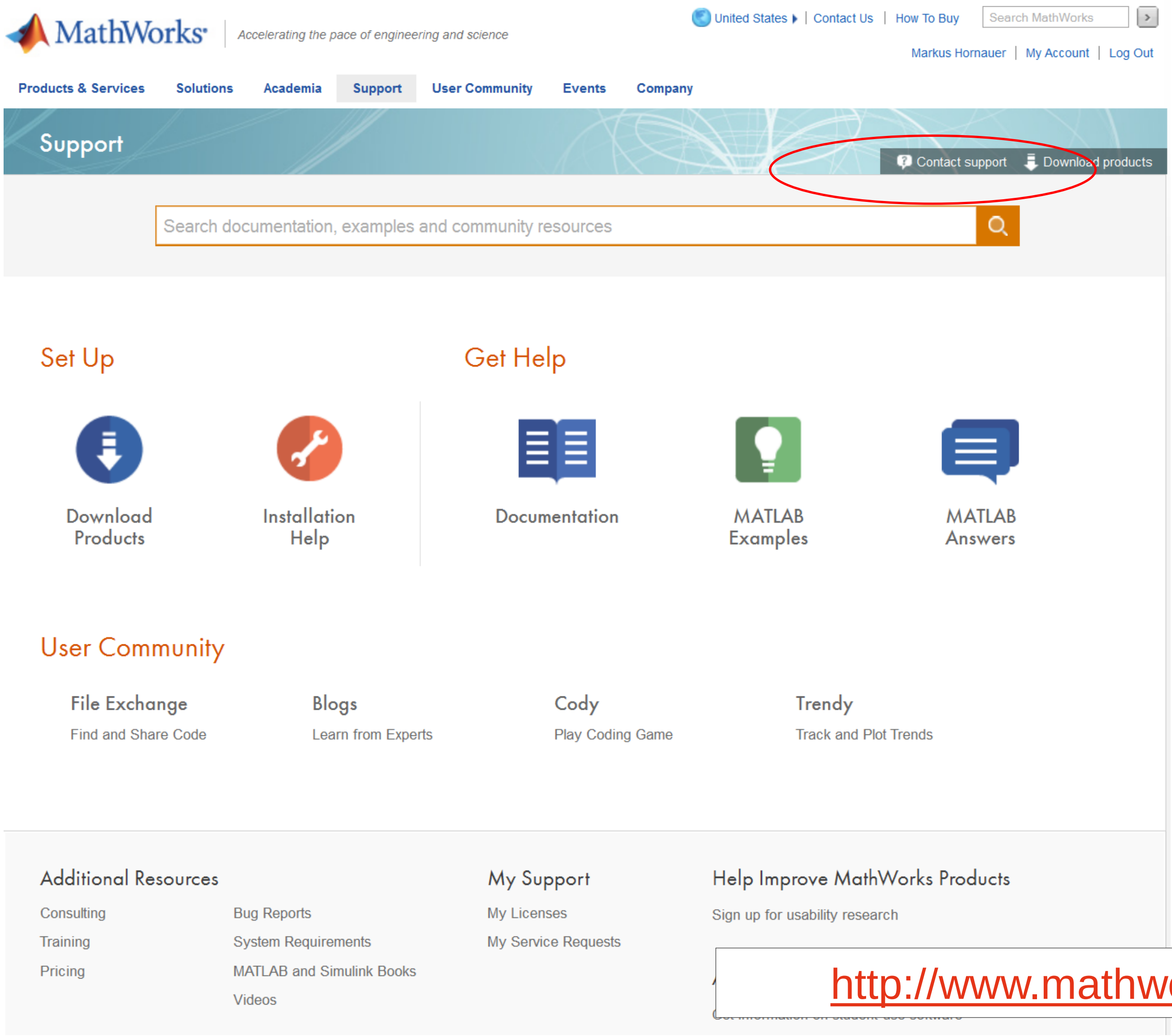
What's New

- **MATLAB Release Notes**
  Summarizes new features, bug fixes, upgrade issues, etc.
- **General Release Notes for R2011b**
  For all products, highlights new features, installation notes, bug fixes, and compatibility issues

Documentation Set

- Getting Started
- User Guides
- **Getting Help**
  Provides instructions for using help functions, the Help browser, and other resources
- **Examples in Documentation**
  Lists major examples in the MATLAB documentation

Getting Help – Technical Support



MathWorks® | Accelerating the pace of engineering and science

United States | Contact Us | How To Buy | Search MathWorks

Markus Hornauer | My Account | Log Out

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

Support

Contact support | Download products

Search documentation, examples and community resources

Set Up

Download Products

Installation Help

Get Help

Documentation

MATLAB Examples

MATLAB Answers

User Community

File Exchange
Find and Share Code

Blogs
Learn from Experts

Cody
Play Coding Game

Trendy
Track and Plot Trends

Additional Resources

Consulting
Training
Pricing

Bug Reports
System Requirements
MATLAB and Simulink Books
Videos

My Support

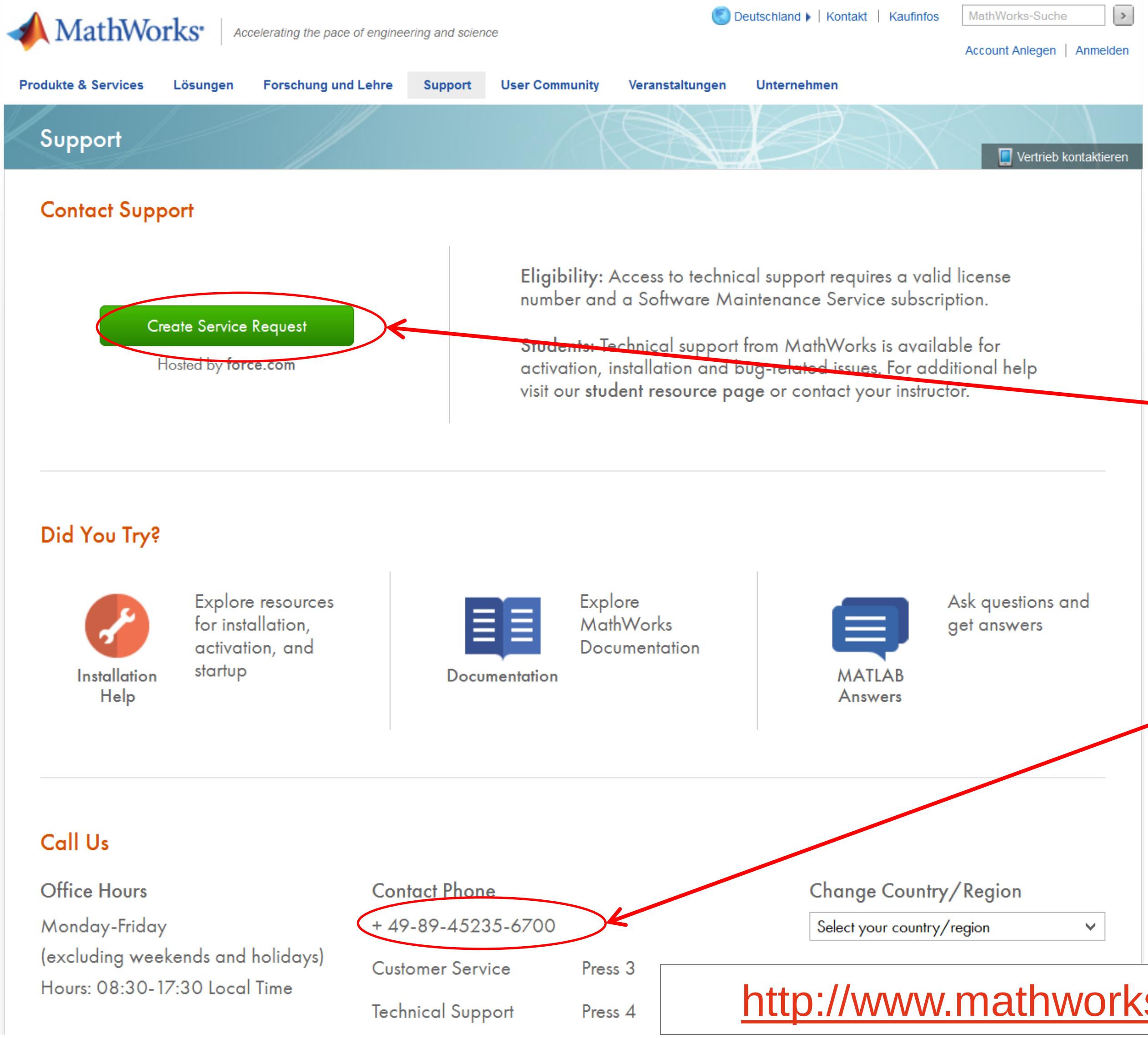
My Licenses
My Service Requests

Help Improve MathWorks Products

Sign up for usability research

http://www.mathworks.com/support/

Getting Help – Technical Support



Support

Contact Support

Hosted by force.com

Eligibility: Access to technical support requires a valid license number and a Software Maintenance Service subscription.

Students: Technical support from MathWorks is available for activation, installation and bug-related issues. For additional help visit our **student resource page** or contact your instructor.

Did You Try?

Explore resources for installation, activation, and startup

Explore MathWorks Documentation

Ask questions and get answers

Call Us

Office Hours

Monday-Friday
(excluding weekends and holidays)
Hours: 08:30-17:30 Local Time

Contact Phone

+ 49-89-45235-6700

Customer Service	Press 3
Technical Support	Press 4

Change Country/Region

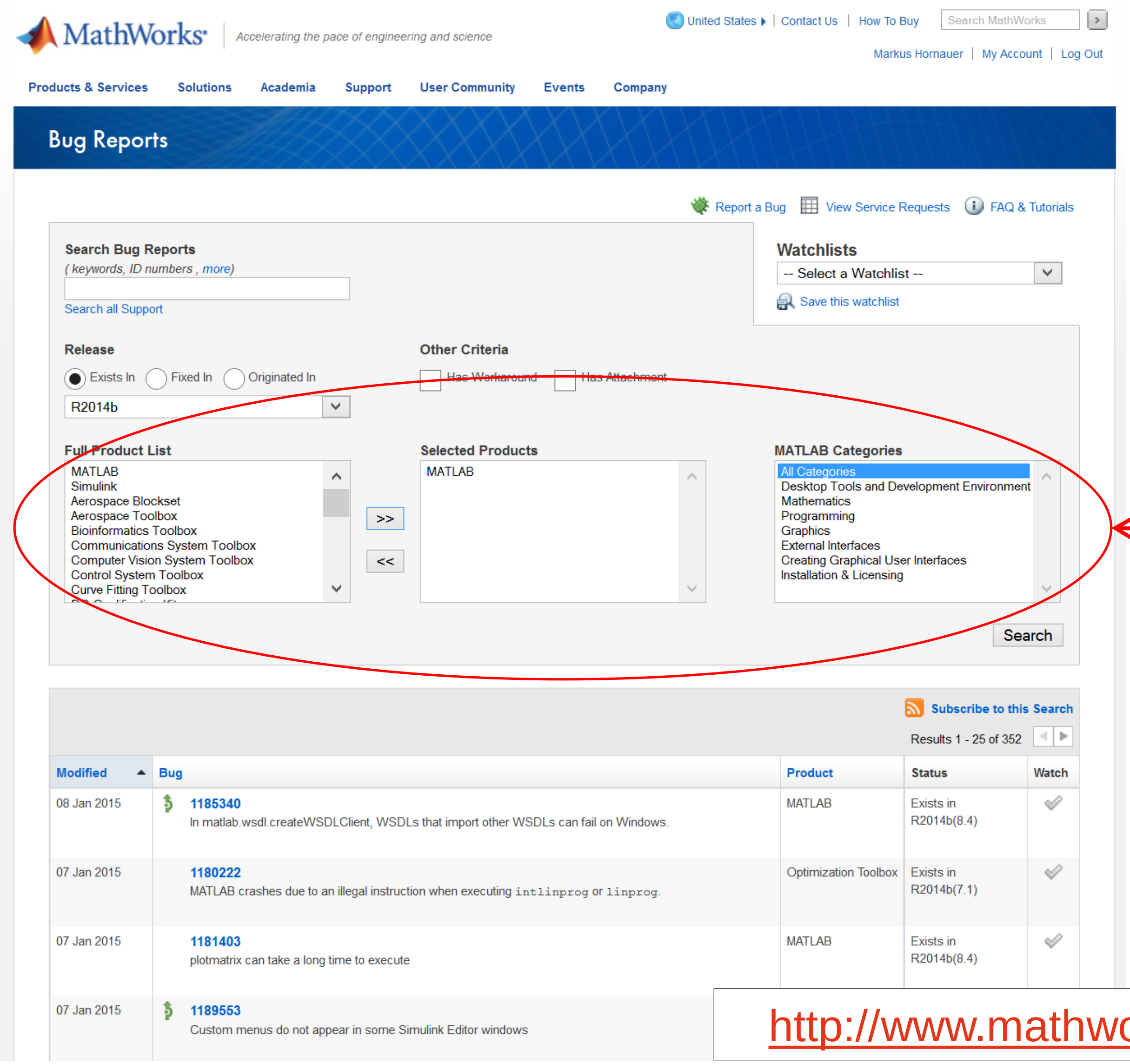
Select your country/region

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/

Getting Help – Bug Report



Bug Reports

Search Bug Reports
(keywords, ID numbers , more)
Search all Support

Watchlists
-- Select a Watchlist --
Save this watchlist

Release
● Exists In ○ Fixed In ○ Originated In
R2014b

Other Criteria
□ Has Workaround □ Has Attachment

Full Product List
MATLAB
Simulink
Aerospace Blockset
Aerospace Toolbox
Bioinformatics Toolbox
Communications System Toolbox
Computer Vision System Toolbox
Control System Toolbox
Curve Fitting Toolbox

>>
<<

Selected Products
MATLAB

MATLAB Categories
All Categories
Desktop Tools and Development Environment
Mathematics
Programming
Graphics
External Interfaces
Creating Graphical User Interfaces
Installation & Licensing

Search

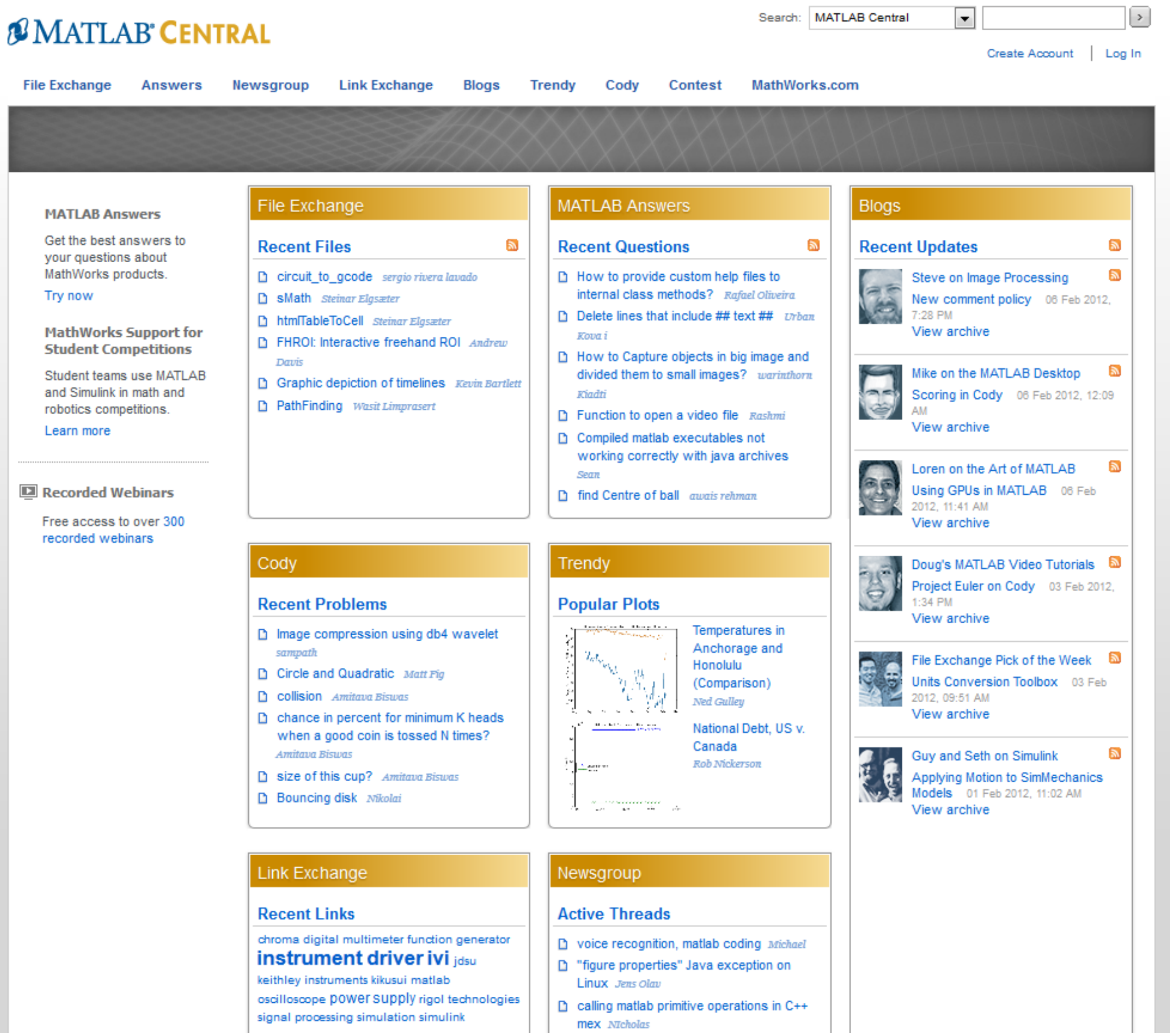
Subscribe to this Search
Results 1 - 25 of 352

Modified	Bug	Product	Status	Watch
08 Jan 2015	**1185340** In matlab.wsdl.createWSDLClient, WSDLs that import other WSDLs can fail on Windows.	MATLAB	Exists in R2014b(8.4)	✓
07 Jan 2015	**1180222** MATLAB crashes due to an illegal instruction when executing `intlinprog` or `linprog`.	Optimization Toolbox	Exists in R2014b(7.1)	✓
07 Jan 2015	**1181403** plotmatrix can take a long time to execute	MATLAB	Exists in R2014b(8.4)	✓
07 Jan 2015	**1189553** Custom menus do not appear in some Simulink Editor windows			

Select product and release

http://www.mathworks.com/support/bugreports

MATLAB Central



MATLAB CENTRAL

Search: MATLAB Central

Create Account | Log In

File Exchange Answers Newsgroup Link Exchange Blogs Trendy Cody Contest MathWorks.com

MATLAB Answers

Get the best answers to your questions about MathWorks products.

Try now

MathWorks Support for Student Competitions

Student teams use MATLAB and Simulink in math and robotics competitions.

Learn more

Recorded Webinars

Free access to over 300 recorded webinars

File Exchange

Recent Files

- circuit_to_gcode *sergio rivera lavado*
- sMath *Steinar Elgsæter*
- htmlTableToCell *Steinar Elgsæter*
- FHROI: Interactive freehand ROI *Andrew Davis*
- Graphic depiction of timelines *Kevin Bartlett*
- PathFinding *Wasit Limprasert*

MATLAB Answers

Recent Questions

- How to provide custom help files to internal class methods? *Rafael Oliveira*
- Delete lines that include ## text ## *Urban Kova i*
- How to Capture objects in big image and divided them to small images? *warinthorn Kiadti*
- Function to open a video file *Rashmi*
- Compiled matlab executables not working correctly with java archives *Sean*
- find Centre of ball *awais rehman*

Blogs

Recent Updates

Steve on Image Processing
New comment policy 06 Feb 2012, 7:28 PM
View archive

Mike on the MATLAB Desktop
Scoring in Cody 06 Feb 2012, 12:09 AM
View archive

Loren on the Art of MATLAB
Using GPUs in MATLAB 06 Feb 2012, 11:41 AM
View archive

Doug's MATLAB Video Tutorials
Project Euler on Cody 03 Feb 2012, 1:34 PM
View archive

File Exchange Pick of the Week
Units Conversion Toolbox 03 Feb 2012, 09:51 AM
View archive

Guy and Seth on Simulink
Applying Motion to SimMechanics Models 01 Feb 2012, 11:02 AM
View archive

Cody

Recent Problems

- Image compression using db4 wavelet *sampath*
- Circle and Quadratic *Matt Fig*
- collision *Amitava Biswas*
- chance in percent for minimum K heads when a good coin is tossed N times? *Amitava Biswas*
- size of this cup? *Amitava Biswas*
- Bouncing disk *Nikolai*

Trendy

Popular Plots

Temperatures in Anchorage and Honolulu (Comparison) *Ned Gulley*

National Debt, US v. Canada *Rob Nickerson*

Link Exchange

Recent Links

chroma digital multimeter function generator **instrument driver ivi** jdsu keithley instruments kikusui matlab oscilloscope power supply rigol technologies signal processing simulation simulink

Newsgroup

Active Threads

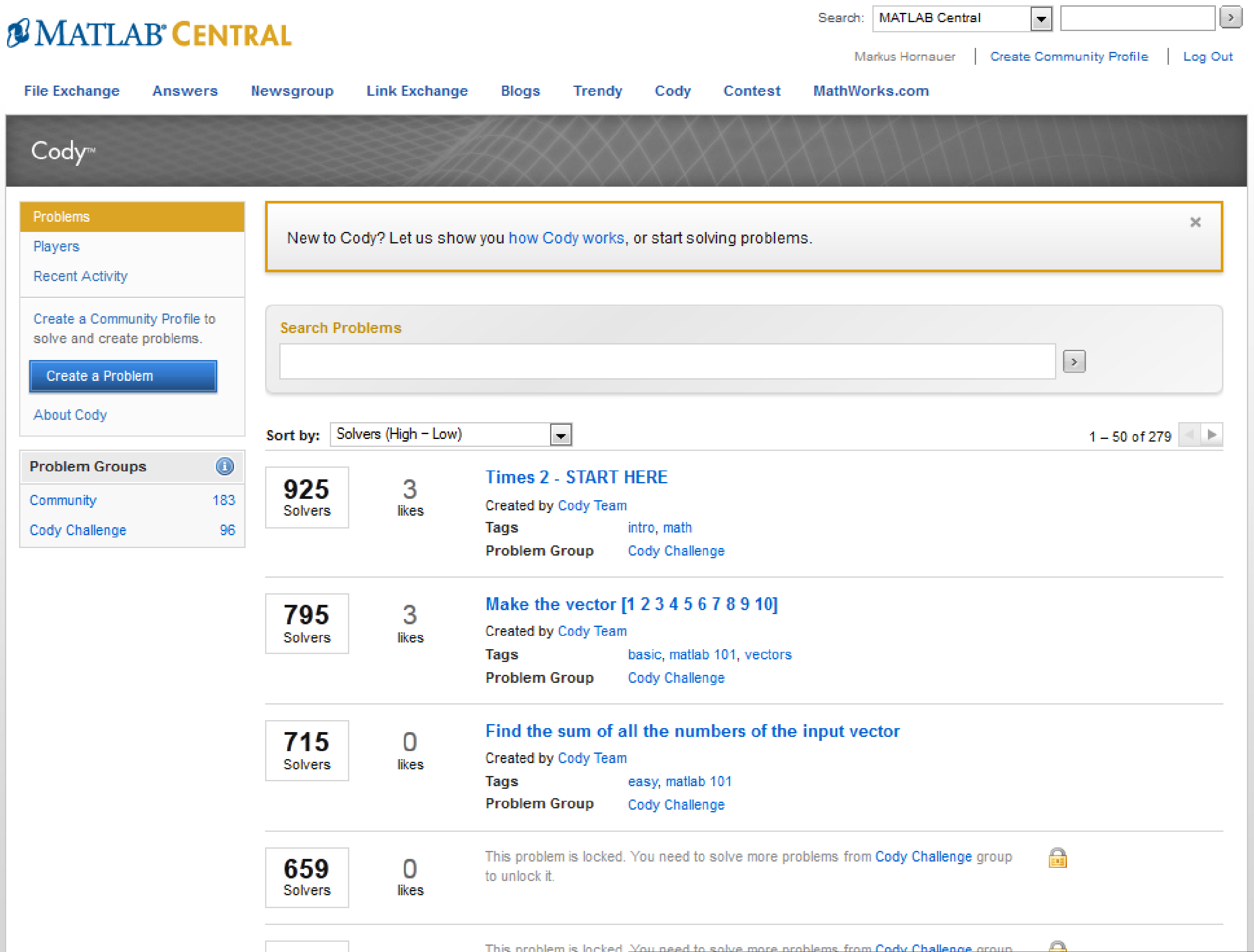
- voice recognition, matlab coding *Michael*
- "figure properties" Java exception on Linux *Jens Olav*
- calling matlab primitive operations in C++ mex *NIcholas*

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



Create a Community Profile to solve and create problems.

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

Problem Groups	
Community	183
Cody Challenge	96

Sort by: Solvers (High – Low)

1 – 50 of 279

925 Solvers | 3 likes
Times 2 - START HERE
Created by Cody Team
Tags: intro, math
Problem Group: Cody Challenge

795 Solvers | 3 likes
Make the vector [1 2 3 4 5 6 7 8 9 10]
Created by Cody Team
Tags: basic, matlab 101, vectors
Problem Group: Cody Challenge

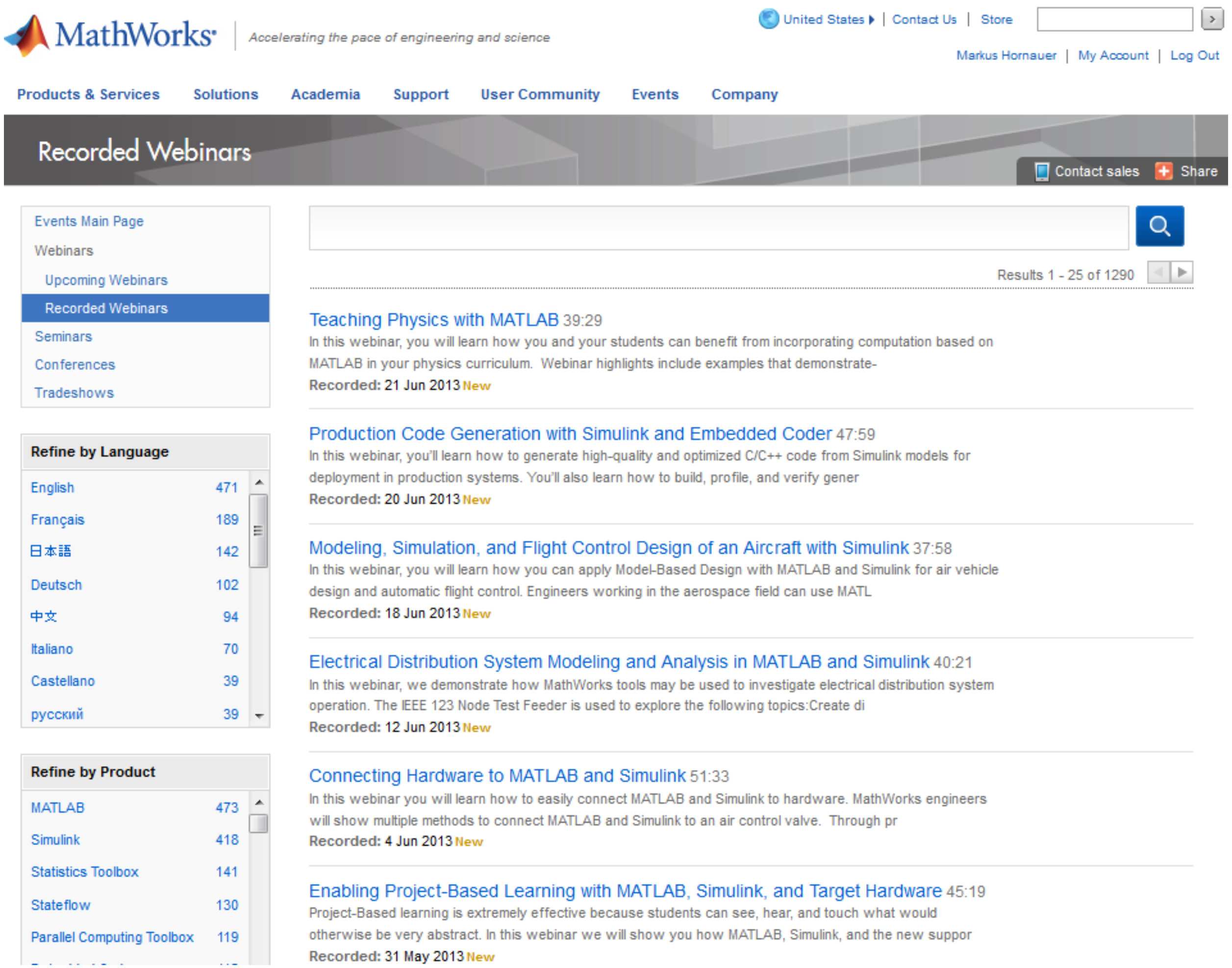
715 Solvers | 0 likes
Find the sum of all the numbers of the input vector
Created by Cody Team
Tags: easy, matlab 101
Problem Group: Cody Challenge

659 Solvers | 0 likes
This problem is locked. You need to solve more problems from Cody Challenge group to unlock it.

601 | 1
This problem is locked. You need to solve more problems from Cody Challenge group to unlock it.

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

Webinars



Results 1 - 25 of 1290

Refine by Language	
English	471
Français	189
日本語	142
Deutsch	102
中文	94
Italiano	70
Castellano	39
русский	39

Refine by Product	
MATLAB	473
Simulink	418
Statistics Toolbox	141
Stateflow	130
Parallel Computing Toolbox	119

Teaching Physics with MATLAB 39:29
In this webinar, you will learn how you and your students can benefit from incorporating computation based on MATLAB in your physics curriculum. Webinar highlights include examples that demonstrate-
Recorded: 21 Jun 2013 New

Production Code Generation with Simulink and Embedded Coder 47:59
In this webinar, you'll learn how to generate high-quality and optimized C/C++ code from Simulink models for deployment in production systems. You'll also learn how to build, profile, and verify gener
Recorded: 20 Jun 2013 New

Modeling, Simulation, and Flight Control Design of an Aircraft with Simulink 37:58
In this webinar, you will learn how you can apply Model-Based Design with MATLAB and Simulink for air vehicle design and automatic flight control. Engineers working in the aerospace field can use MATL
Recorded: 18 Jun 2013 New

Electrical Distribution System Modeling and Analysis in MATLAB and Simulink 40:21
In this webinar, we demonstrate how MathWorks tools may be used to investigate electrical distribution system operation. The IEEE 123 Node Test Feeder is used to explore the following topics:Create di
Recorded: 12 Jun 2013 New

Connecting Hardware to MATLAB and Simulink 51:33
In this webinar you will learn how to easily connect MATLAB and Simulink to hardware. MathWorks engineers will show multiple methods to connect MATLAB and Simulink to an air control valve. Through pr
Recorded: 4 Jun 2013 New

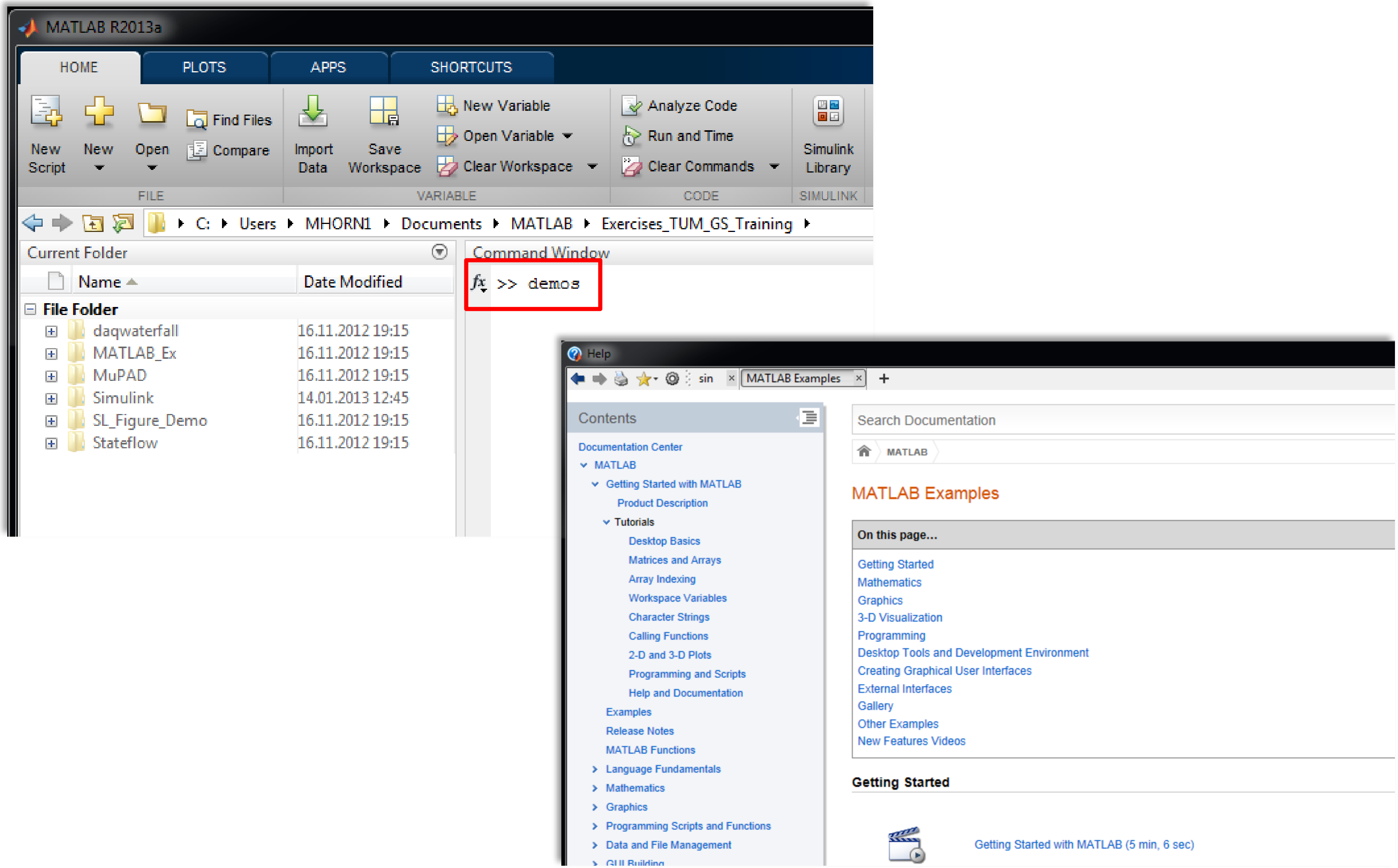
Enabling Project-Based Learning with MATLAB, Simulink, and Target Hardware 45:19
Project-Based learning is extremely effective because students can see, hear, and touch what would otherwise be very abstract. In this webinar we will show you how MATLAB, Simulink, and the new suppor
Recorded: 31 May 2013 New

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

Demos

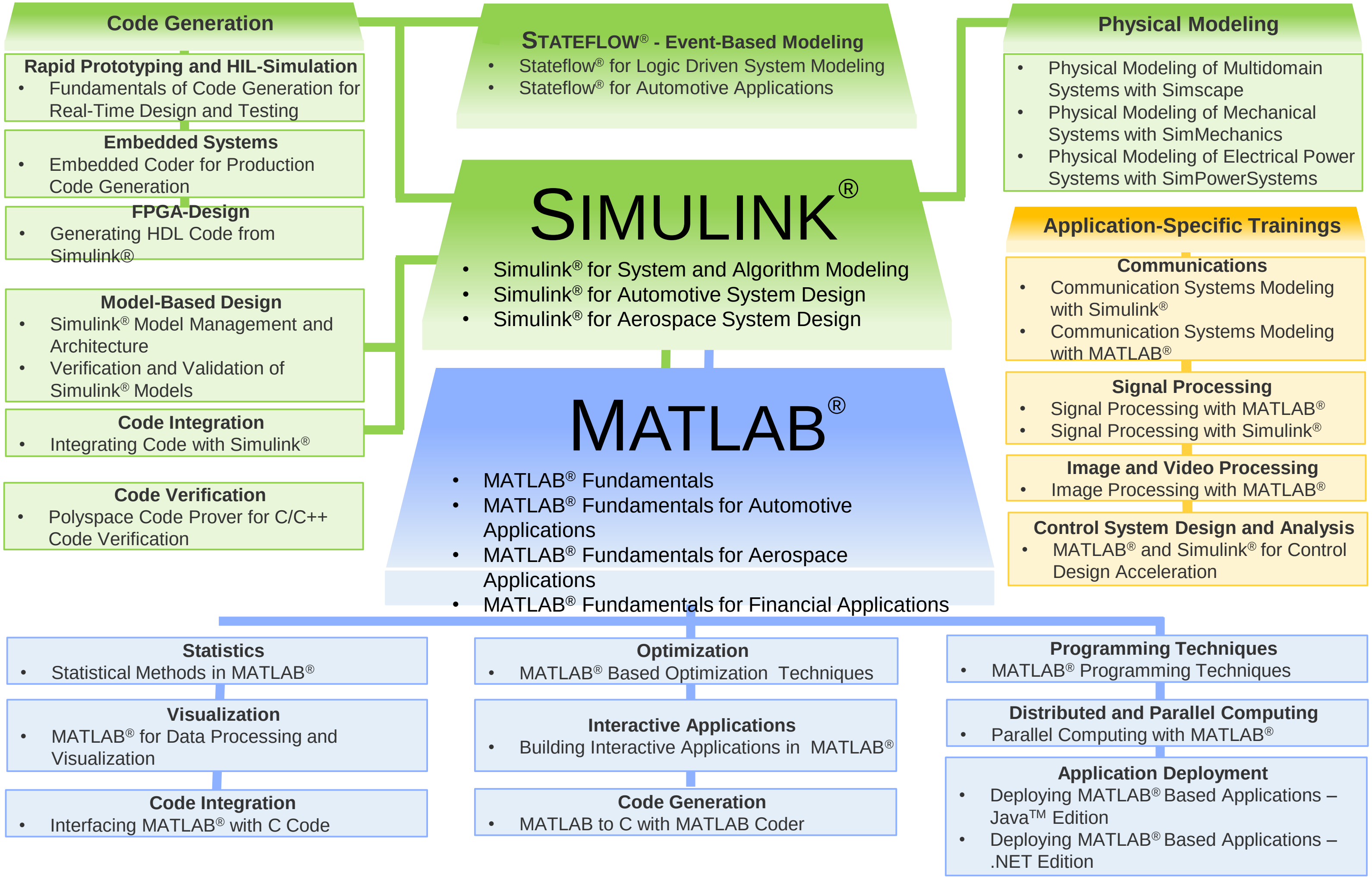


```
>> demos
```

MATLAB R2013a
HOME
PLOTS
APPS
SHORTCUTS
New Script
New
Open
Find Files
Compare
Import Data
Save Workspace
New Variable
Open Variable
Clear Workspace
Analyze Code
Run and Time
Clear Commands
Simulink Library
FILE
VARIABLE
CODE
SIMULINK
C: ▸ Users ▸ MHORN1 ▸ Documents ▸ MATLAB ▸ Exercises_TUM_GS_Training ▸
Current Folder
Command Window
Name
Date Modified
File Folder
daqwaterfall 16.11.2012 19:15
MATLAB_Ex 16.11.2012 19:15
MuPAD 16.11.2012 19:15
Simulink 14.01.2013 12:45
SL_Figure_Demo 16.11.2012 19:15
Stateflow 16.11.2012 19:15
Help
sin
MATLAB Examples
Contents
Documentation Center
MATLAB
Getting Started with MATLAB
Product Description
Tutorials
Desktop Basics
Matrices and Arrays
Array Indexing
Workspace Variables
Character Strings
Calling Functions
2-D and 3-D Plots
Programming and Scripts
Help and Documentation
Examples
Release Notes
MATLAB Functions
Language Fundamentals
Mathematics
Graphics
Programming Scripts and Functions
Data and File Management
GUI Building
Search Documentation
MATLAB
MATLAB Examples
On this page...
Getting Started
Mathematics
Graphics
3-D Visualization
Programming
Desktop Tools and Development Environment
Creating Graphical User Interfaces
External Interfaces
Gallery
Other Examples
New Features Videos
Getting Started
Getting Started with MATLAB (5 min, 6 sec)

Trainings

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.



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

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

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

MATLAB®
- MATLAB® Fundamentals
- MATLAB® Fundamentals for Automotive Applications
- MATLAB® Fundamentals for Aerospace Applications
- MATLAB® Fundamentals for Financial 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®
- Signal Processing with Simulink®

Image and Video Processing
- Image Processing with MATLAB®

Control System Design and Analysis
- MATLAB® and Simulink® for Control Design Acceleration

Statistics
- Statistical Methods in MATLAB®

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®

Code Generation
- MATLAB to C with MATLAB Coder

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

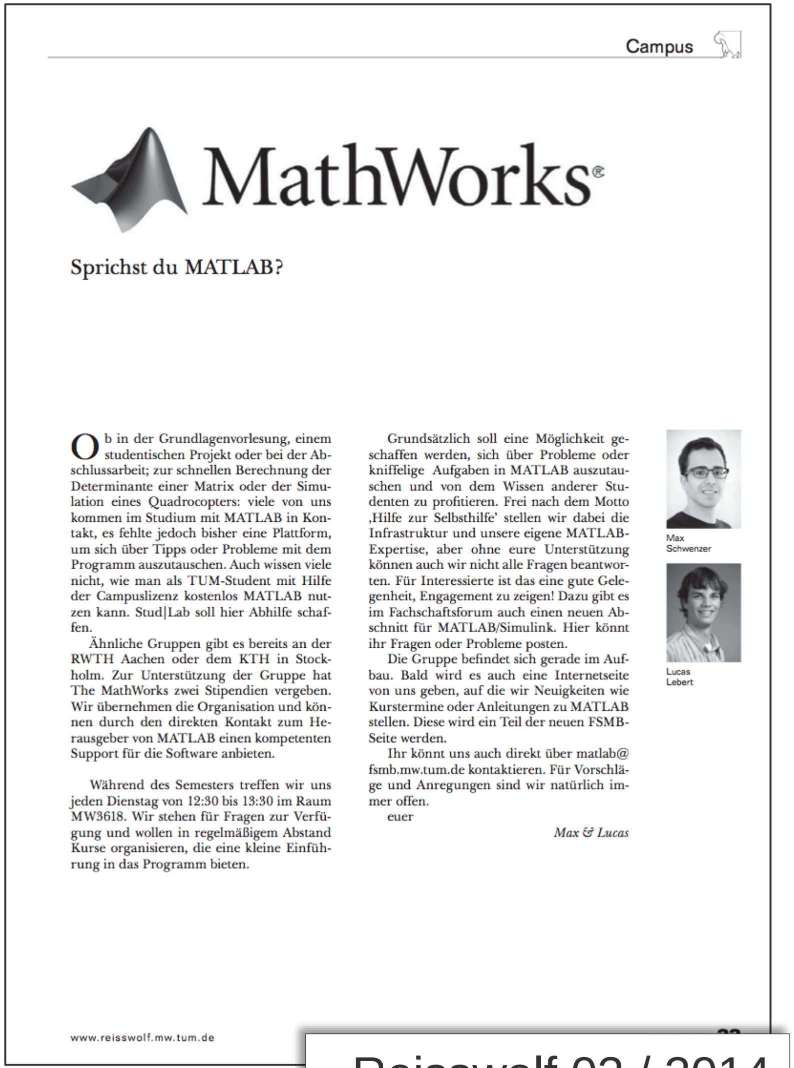
Stud|Lab

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



Campus

MathWorks®

Sprichst du MATLAB?

Ob in der Grundlagenvorlesung, einem studentischen Projekt oder bei der Abschlussarbeit; zur schnellen Berechnung der Determinante einer Matrix oder der Simulation eines Quadrocopters: viele von uns kommen im Studium mit MATLAB in Kontakt, es fehlte jedoch bisher eine Plattform, um sich über Tipps oder Probleme mit dem Programm auszutauschen. Auch wissen viele nicht, wie man als TUM-Student mit Hilfe der Campuslizenz kostenlos MATLAB nutzen kann. Stud|Lab soll hier Abhilfe schaffen.

Ähnliche Gruppen gibt es bereits an der RWTH Aachen oder dem KTH in Stockholm. Zur Unterstützung der Gruppe hat The MathWorks zwei Stipendien vergeben. Wir übernehmen die Organisation und können durch den direkten Kontakt zum Herausgeber von MATLAB einen kompetenten Support für die Software anbieten.

Während des Semesters treffen wir uns jeden Dienstag von 12:30 bis 13:30 im Raum MW3618. Wir stehen für Fragen zur Verfügung und wollen in regelmäßigem Abstand Kurse organisieren, die eine kleine Einführung in das Programm bieten.

Grundsätzlich soll eine Möglichkeit geschaffen werden, sich über Probleme oder kniffelige Aufgaben in MATLAB auszutauschen und von dem Wissen anderer Studenten zu profitieren. Frei nach dem Motto ‚Hilfe zur Selbsthilfe' stellen wir dabei die Infrastruktur und unsere eigene MATLAB-Expertise, aber ohne eure Unterstützung können auch wir nicht alle Fragen beantworten. Für Interessierte ist das eine gute Gelegenheit, Engagement zu zeigen! Dazu gibt es im Fachschaftsforum auch einen neuen Abschnitt für MATLAB/Simulink. Hier könnt ihr Fragen oder Probleme posten.

Die Gruppe befindet sich gerade im Aufbau. Bald wird es auch eine Internetseite von uns geben, auf die wir Neuigkeiten wie Kurstermine oder Anleitungen zu MATLAB stellen. Diese wird ein Teil der neuen FSMB-Seite werden.

Ihr könnt uns auch direkt über matlab@fsmb.mw.tum.de kontaktieren. Für Vorschläge und Anregungen sind wir natürlich immer offen.

euer

Max & Lucas

Max Schwenzer

Lucas Lebert

www.reisswolf.mw.tum.de

Reisswolf 03 / 2014

Outline
Introduction to MATLAB

Basics:

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

Advanced:

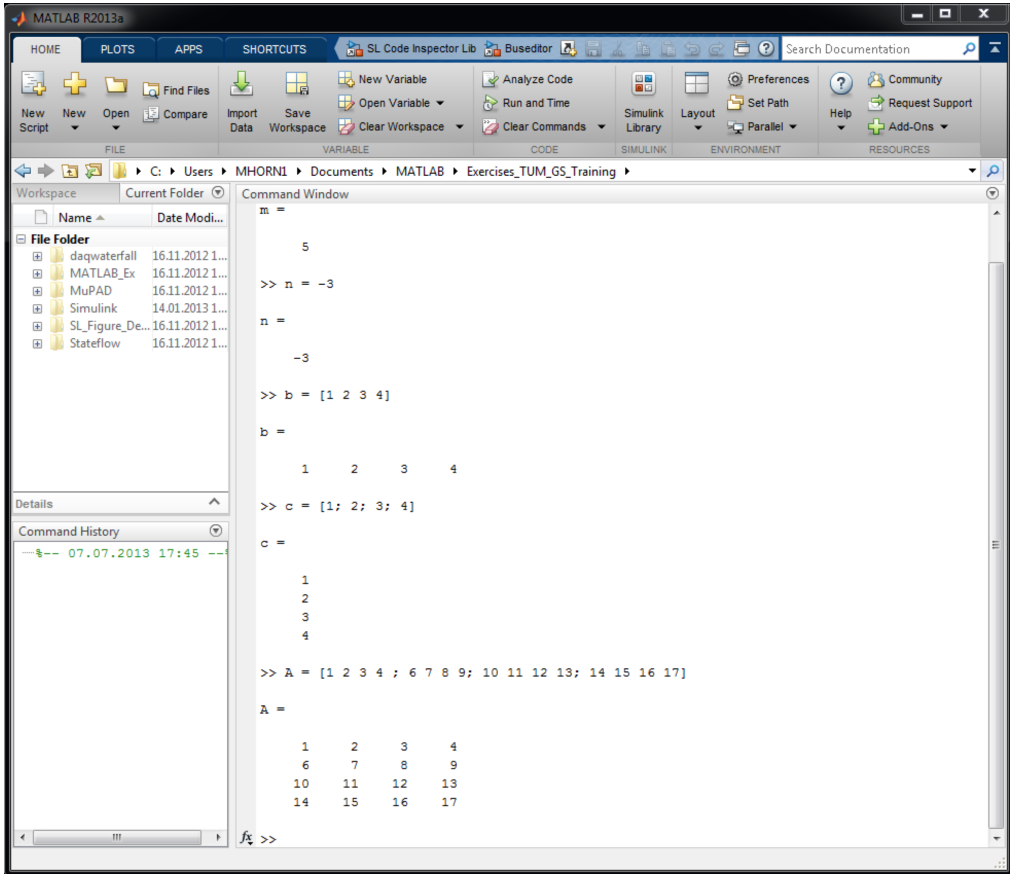
1) Programming with MATLAB
2) Graphical User Interfaces in MATLAB

Toolboxes:

1) Symbolic Math Toolbox
2) Control System Toolbox and Curve Fitting Toolbox

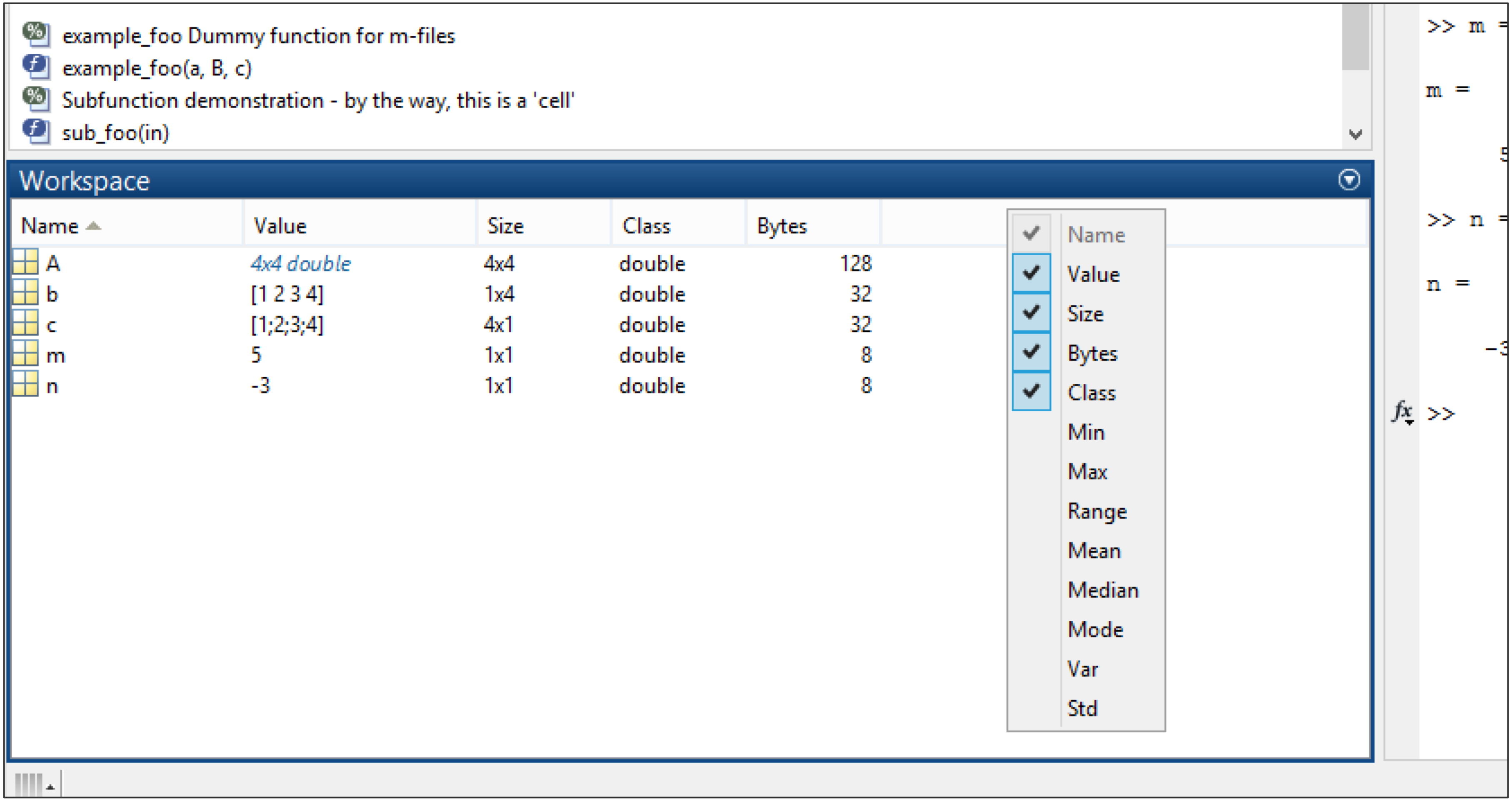
Exercise:

- Create a scalar variable: m, n
- Create a vector: b = (1x4), c = (4x1)
- Create a matrix: A (4x4)
- Change the variables



```
m =

     5

>> n = -3

n =

    -3

>> b = [1 2 3 4]

b =

     1     2     3     4

>> c = [1; 2; 3; 4]

c =

     1
     2
     3
     4

>> A = [1 2 3 4 ; 6 7 8 9; 10 11 12 13; 14 15 16 17]

A =

     1     2     3     4
     6     7     8     9
    10    11    12    13
    14    15    16    17

>>
```

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)

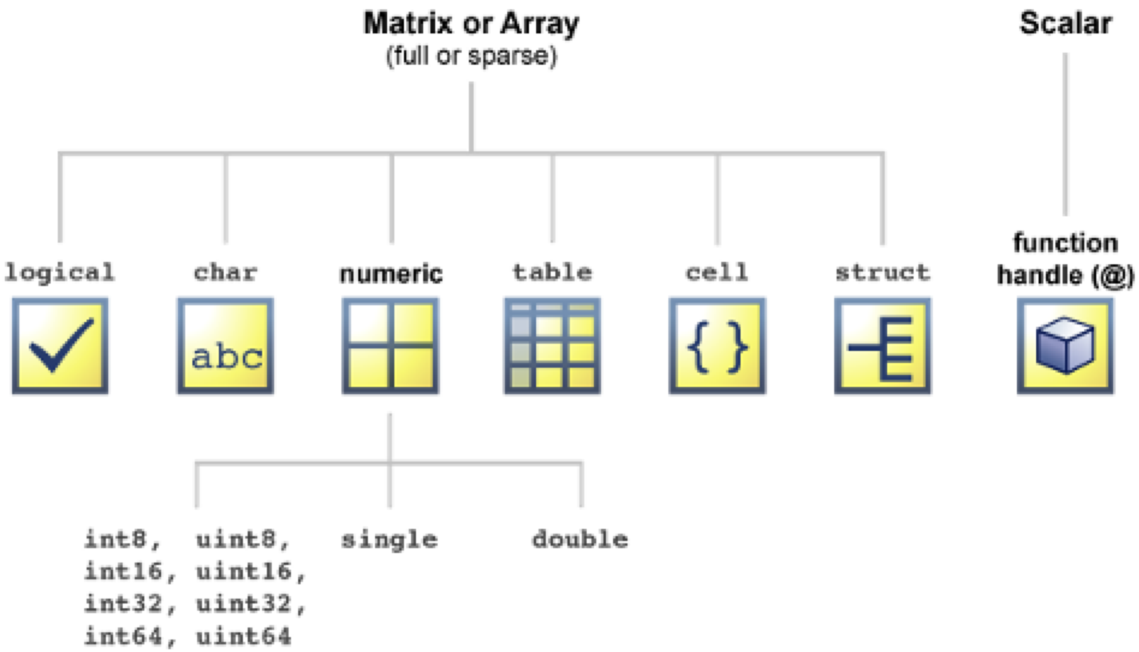
Name	Value	Size	Class	Bytes
A	*4x4 double*	4x4	double	128
b	[1 2 3 4]	1x4	double	32
c	[1;2;3;4]	4x1	double	32
m	5	1x1	double	8
n	-3	1x1	double	8

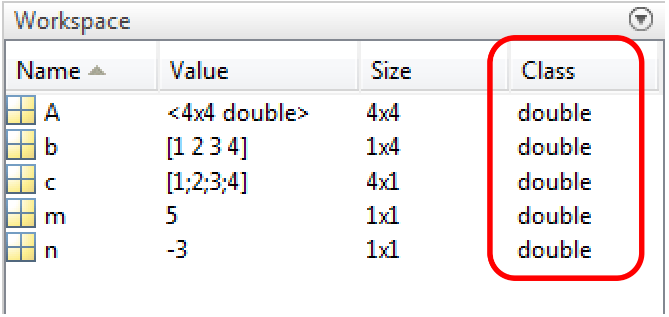
- ✓ Name
- ✓ Value
- ✓ Size
- ✓ Bytes
- ✓ Class
- Min
- Max
- Range
- Mean
- Median
- Mode
- Var
- Std

```
>> m =

m =

    5

>> n =

n =

   -3

>>
```

Data and Variables

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

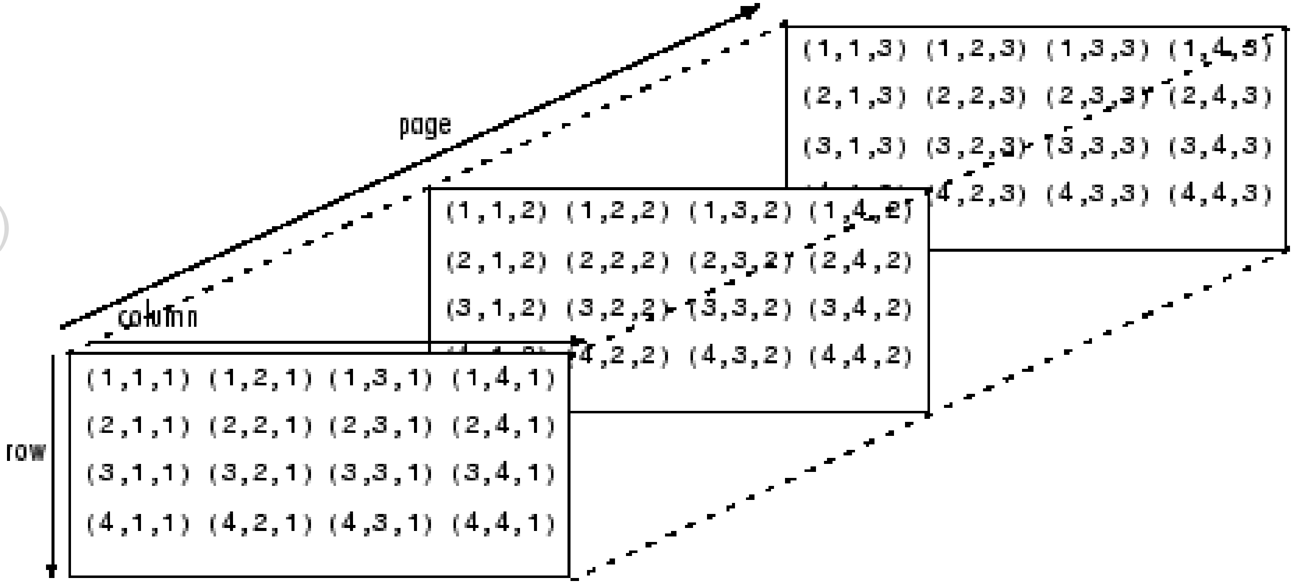




Name ▲	Value	Size	Class
A	<4x4 double>	4x4	double
b	[1 2 3 4]	1x4	double
c	[1;2;3;4]	4x1	double
m	5	1x1	double
n	-3	1x1	double

Data and Variables

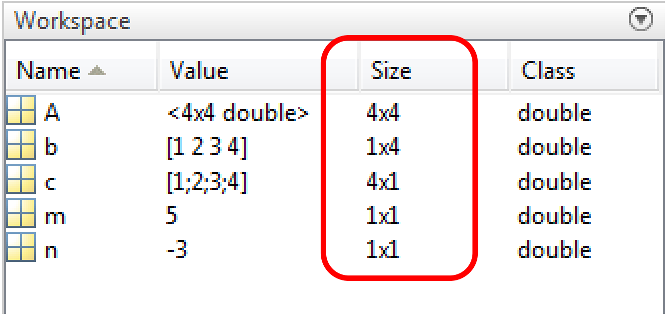
- Class
- Size
- Value
- Name (“variable”)



(1,1,3)	(1,2,3)	(1,3,3)	(1,4,3)
(2,1,3)	(2,2,3)	(2,3,3)	(2,4,3)
(3,1,3)	(3,2,3)	(3,3,3)	(3,4,3)
(4,1,3)	(4,2,3)	(4,3,3)	(4,4,3)

(1,1,2)	(1,2,2)	(1,3,2)	(1,4,2)
(2,1,2)	(2,2,2)	(2,3,2)	(2,4,2)
(3,1,2)	(3,2,2)	(3,3,2)	(3,4,2)
(4,1,2)	(4,2,2)	(4,3,2)	(4,4,2)

(1,1,1)	(1,2,1)	(1,3,1)	(1,4,1)
(2,1,1)	(2,2,1)	(2,3,1)	(2,4,1)
(3,1,1)	(3,2,1)	(3,3,1)	(3,4,1)
(4,1,1)	(4,2,1)	(4,3,1)	(4,4,1)



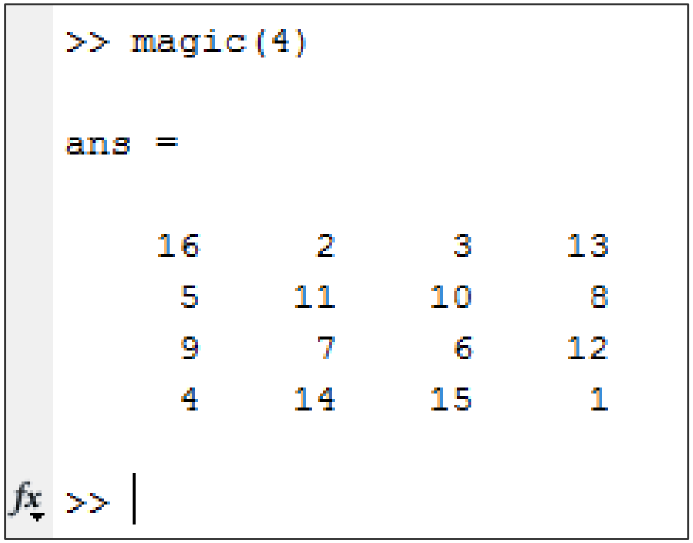
Name ▲	Value	Size	Class
A	<4x4 double>	4x4	double
b	[1 2 3 4]	1x4	double
c	[1;2;3;4]	4x1	double
m	5	1x1	double
n	-3	1x1	double

m*n

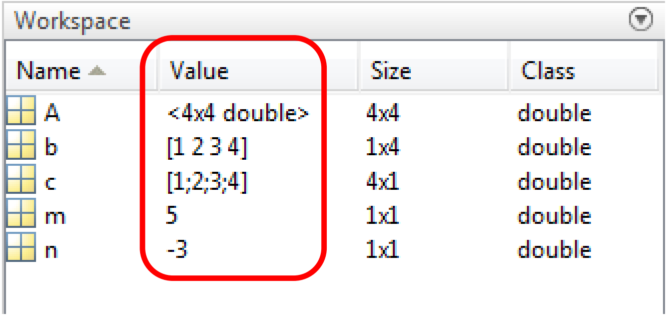
m*n*…*z

Data and Variables

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



```
>> magic(4)

ans =

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

>> 
```

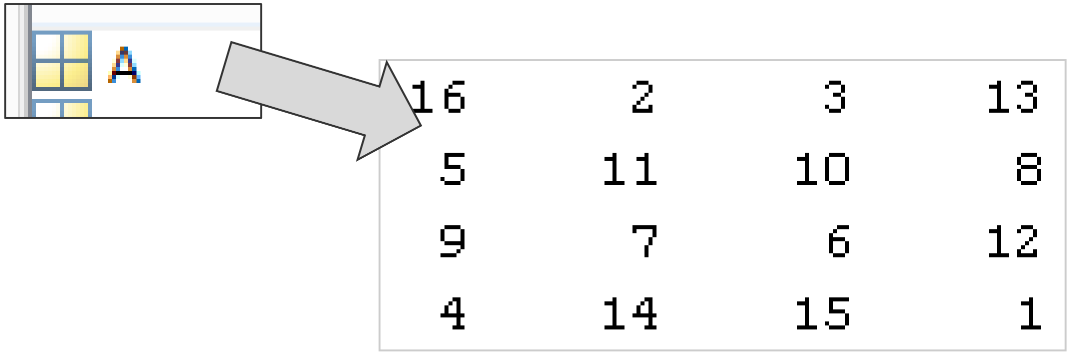


Workspace

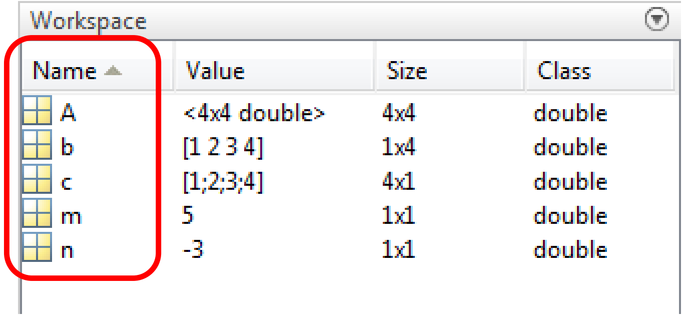
Name	Value	Size	Class
A	<4x4 double>	4x4	double
b	[1 2 3 4]	1x4	double
c	[1;2;3;4]	4x1	double
m	5	1x1	double
n	-3	1x1	double

Data and Variables

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



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



Name	Value	Size	Class
A	<4x4 double>	4x4	double
b	[1 2 3 4]	1x4	double
c	[1;2;3;4]	4x1	double
m	5	1x1	double
n	-3	1x1	double

Angermann et al. p. 8

Assignments

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

Reserved Variables

`pi`	π
`i, j`	$\sqrt{(-1)}$
`inf`	infinity ∞
`ans`	standard output of results (answer)
`eps`	floating point accuracy
`NaN`	Not a Number (invalid result)

Angermann et al. p. 9

Mathematical Functions and Operators

`+ - * / ^`	Operators	`exp(x)`	exponential `function`
`mod(x,y)`	x modulo y	`log(x)`	natural logarithm
`rem(x,y)`	remainder after division x/y	`log10(x)`	common log (basis 10)
`sqrt(x)`	square root $\sqrt{x}$	`erf(x/√2)`	normal distribution
`abs(x)`	absolute value	`real(x)`	real part
`sign(x)`	sign	`imag(x)`	imaginary part
`round(x)`	round	`conj(x)`	complex conjugate
`ceil(x)`	round up	`angle(x)`	phase of a complex value
`floor(x)`	round down		

Trigonometric Functions

`sin(x)`	sine	`tan(x)`	tangent
`cos(x)`	cosine	`cot(x)`	cotangent
`sind(x)`	sine (x in degree)	`atan(y/x)`	arc tangent ± π/2
`cosd(x)`	cosine (x in degree)	`atan2(y/x)`	arc tangent ± π/2

Angermann et al. p. 12

Vectors and Matrices

`[x1 x2 … ; x3 x4 …]`	input of matrices and vectors
`x1:x2`	creation of a line vector [x1 x1+1 x1+2…x2]
`x1:d:x2`	creation of a line vector [x1 x1+d x1+2*d…x2]
`linspace(x1,x2,n)`	line vector, start val x1, end val x2, size n, equally distributed
`logspace(x1,x2,n)`	line vector, start val x1, end val x2, size n, logarithmically distributed

`eye(n)`	nxn identity matrix
`ones(n)`	nxn matrix with all entries equal to 1
`zeros(n)`	nxn matrix with all entries equal to 0
`rand(x)`	nxn matrix with random entries between 0 and 1
`randn(x)`	nxn matrix with normally distributed random entries
`magic(x)`	nxn matrix constructed from the integers 1 through n^2 with equal row and column sums

Angermann et al. p. 13

Functions and Operators for Vectors and Matrices

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

Additional functions

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

Angermann et al. p. 16

Structs and Cell Arrays

struct(‘*n1*’, *w1*, ‘*n2*’, *w2*, …)	create a struct variable
Structure.name	acess to the element name
CellArray = {*Value*}	creation of a Cell Array
CellArray{*index*} = *Value*	creation of a Cell Array
cell(*n*)	Creation of a n x n – Cell Array
cell(*m*,*n*)	Creation of a m x n – Cell Array

Angermann et al. p. 17

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

Angermann et al. p. 19

Relational Operators

`==`	`eq(a,b)`	equal
`~=`	`ne(a,b)`	not equal
`<`	`lt(a,b)`	less than
`<=`	`le(a,b)`	less or equal than
`>`	`gt(a,b)`	greater than
`>=`	`ge(a,b)`	greater or equal than

Logical Operators

`~`	`not(a)`	logical not
`&`	`and(a,b)`	AND
`|`	`or(a,b)`	OR
	`xor(a,b)`	exclusive OR
`&&`		shortcut AND (scalar)
`||`		Shortcut OR (scalar)

Additional Operators

`all(vec)`	each element is true
`any(vec)`	at least 1 element is true
`logical(a)`	type cast to boolean
`exist('x')`	existance of x
`find(vec)`	index of true elements
`[~,~,a] = foo(x,y,z)`	select only 3rd return value

Brackets

{}, () or []

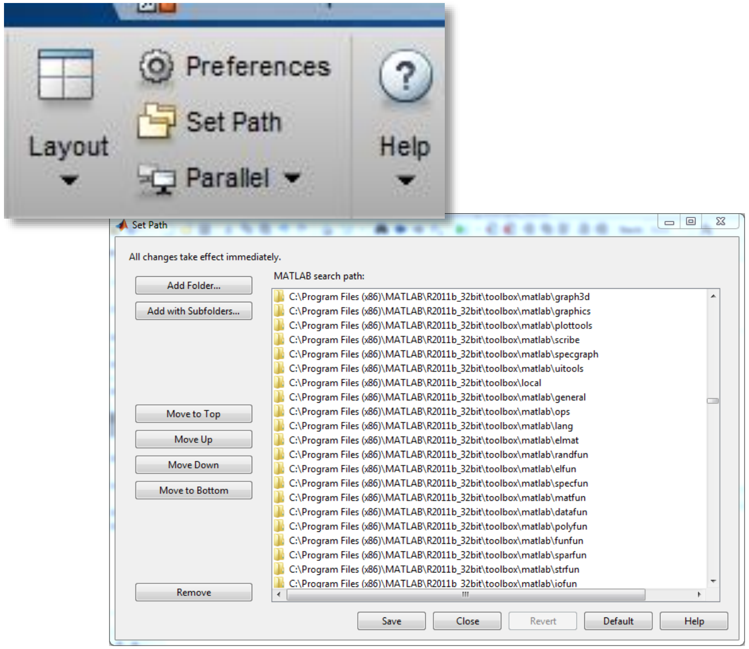
Cell Arrays

Indexing
Order of operations
Argument list

Matrix/Vector creation
Concatenation
Multiple outputs

The MATLAB Path

1

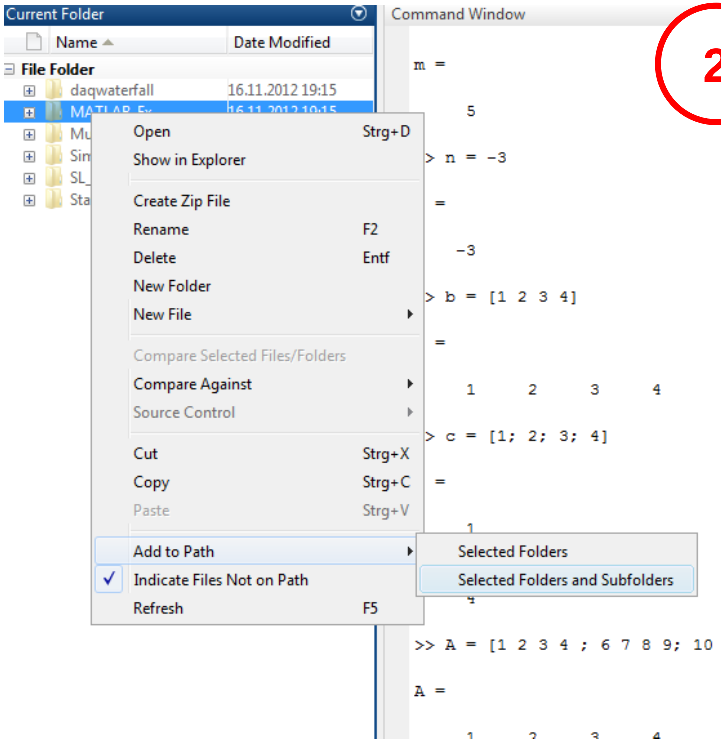


All changes take effect immediately.

MATLAB search path:

```
C:\Program Files (x86)\MATLAB\R2011b_32bit\toolbox\matlab\graph3d
C:\Program Files (x86)\MATLAB\R2011b_32bit\toolbox\matlab\graphics
C:\Program Files (x86)\MATLAB\R2011b_32bit\toolbox\matlab\plottools
C:\Program Files (x86)\MATLAB\R2011b_32bit\toolbox\matlab\scribe
C:\Program Files (x86)\MATLAB\R2011b_32bit\toolbox\matlab\specgraph
C:\Program Files (x86)\MATLAB\R2011b_32bit\toolbox\matlab\uitools
C:\Program Files (x86)\MATLAB\R2011b_32bit\toolbox\local
C:\Program Files (x86)\MATLAB\R2011b_32bit\toolbox\matlab\general
C:\Program Files (x86)\MATLAB\R2011b_32bit\toolbox\matlab\ops
C:\Program Files (x86)\MATLAB\R2011b_32bit\toolbox\matlab\lang
C:\Program Files (x86)\MATLAB\R2011b_32bit\toolbox\matlab\elmat
C:\Program Files (x86)\MATLAB\R2011b_32bit\toolbox\matlab\randfun
C:\Program Files (x86)\MATLAB\R2011b_32bit\toolbox\matlab\elfun
C:\Program Files (x86)\MATLAB\R2011b_32bit\toolbox\matlab\specfun
C:\Program Files (x86)\MATLAB\R2011b_32bit\toolbox\matlab\matfun
C:\Program Files (x86)\MATLAB\R2011b_32bit\toolbox\matlab\datafun
C:\Program Files (x86)\MATLAB\R2011b_32bit\toolbox\matlab\polyfun
C:\Program Files (x86)\MATLAB\R2011b_32bit\toolbox\matlab\funfun
C:\Program Files (x86)\MATLAB\R2011b_32bit\toolbox\matlab\sparfun
C:\Program Files (x86)\MATLAB\R2011b_32bit\toolbox\matlab\strfun
C:\Program Files (x86)\MATLAB\R2011b_32bit\toolbox\matlab\iofun
```

2



```
m =

     5

> n = -3

 =

    -3

> b = [1 2 3 4]

 =

     1     2     3     4

> c = [1; 2; 3; 4]

 =

     1

     4

>> A = [1 2 3 4 ; 6 7 8 9; 10

A =

     1     2     3     4
```

3

>> 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 = 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

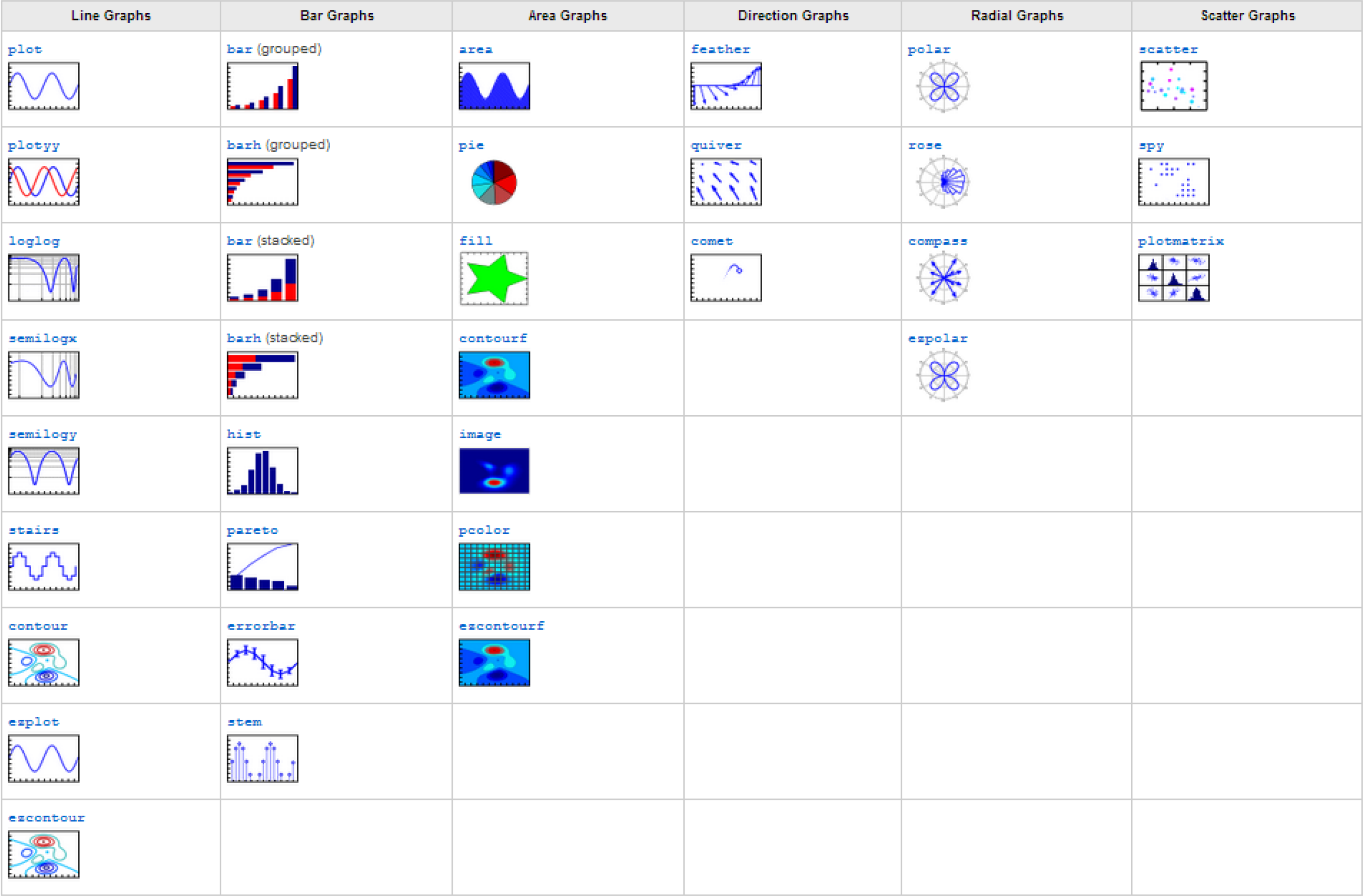
Advanced:

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

Toolboxes:

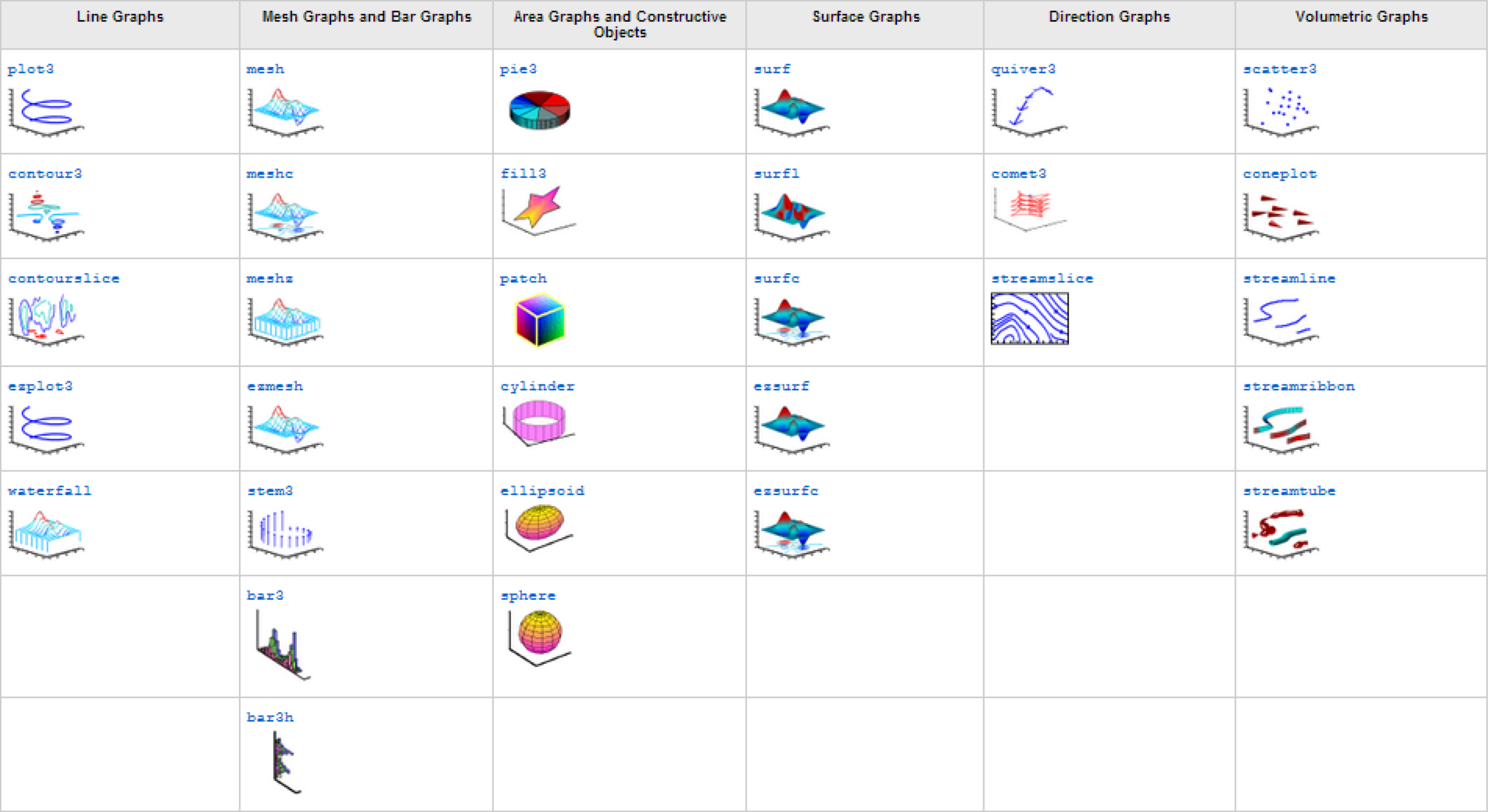
1) Symbolic Math Toolbox
2) Control System Toolbox and Curve Fitting Toolbox

Visualization Tools – 2D



Line Graphs	Bar Graphs	Area Graphs	Direction Graphs	Radial Graphs	Scatter Graphs
plot	bar (grouped)	area	feather	polar	scatter
plotyy	barh (grouped)	pie	quiver	rose	spy
loglog	bar (stacked)	fill	comet	compass	plotmatrix
semilogx	barh (stacked)	contourf		ezpolar	
semilogy	hist	image			
stairs	pareto	pcolor			
contour	errorbar	ezcontourf			
ezplot	stem				
ezcontour					

Visualization Tools – 3D



Line Graphs	Mesh Graphs and Bar Graphs	Area Graphs and Constructive Objects	Surface Graphs	Direction Graphs	Volumetric Graphs
plot3	mesh	pie3	surf	quiver3	scatter3
contour3	meshc	fill3	surfl	comet3	coneplot
contourslice	meshz	patch	surfc	streamslice	streamline
ezplot3	ezmesh	cylinder	ezsurf		streamribbon
waterfall	stem3	ellipsoid	ezsurfc		streamtube
	bar3	sphere			
	bar3h				

Angermann et al. p. 46

Graphics : 2D plot commands

`plot([xvalues,] yvalues…[,plotstyle])`	plot, linear axis
`stairs([xvalues,] yvalues…[,plotstyle])`	plot, linear axis, stair step graph
`bar(…), stem(…)`	plot, linear axis, bars
`loglog(xvalues, yvalues…[,plotstyle])`	plot, logarithmic axis
`semilogx(xvalues, yvalues…[,plotstyle])`	plot, logarithmic x-axis
`semilogy(xvalues, yvalues…[,plotstyle])`	plot, logarithmic y-axis
`polar(angle, radius…[,plotstyle])`	plot, polar coordinates
`fplot(function, range)`	plot, explicit function
`ezplot(function(x,y)[,range])`	plot, implicit function
`ezplot(function1, function2[,range])`	plot, parametric curve
`hold [on	off]`	retain current graph in figure

Demo

- Compute `y = sin(2t) + cos(t)` where `t` is from 1 to 10 seconds.

- Plot `y` and `t`
  >> plot(t, y);

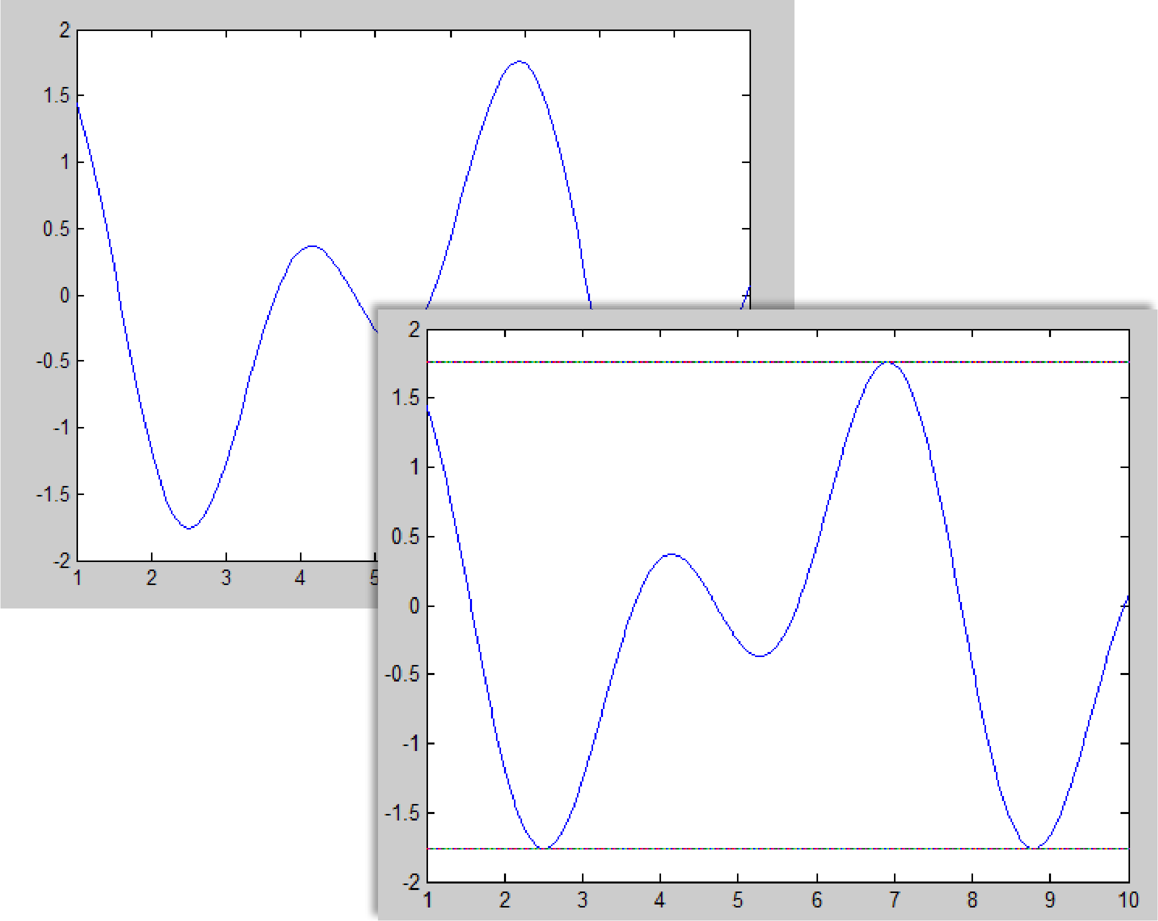
- >> y_1_min = min(y);

- >> plot(t, y_1_min);

- >> hold on;

- >> y_1_max = max(y);

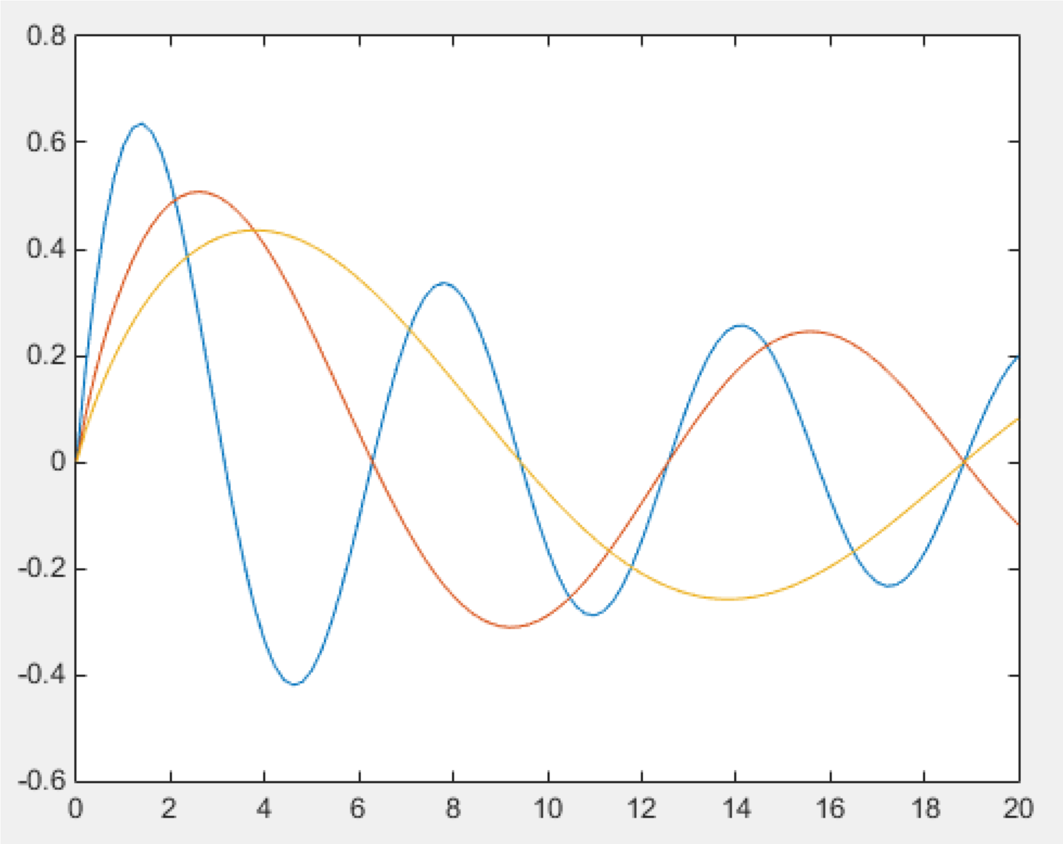
- >> plot(t, y);



t and y are vectors!

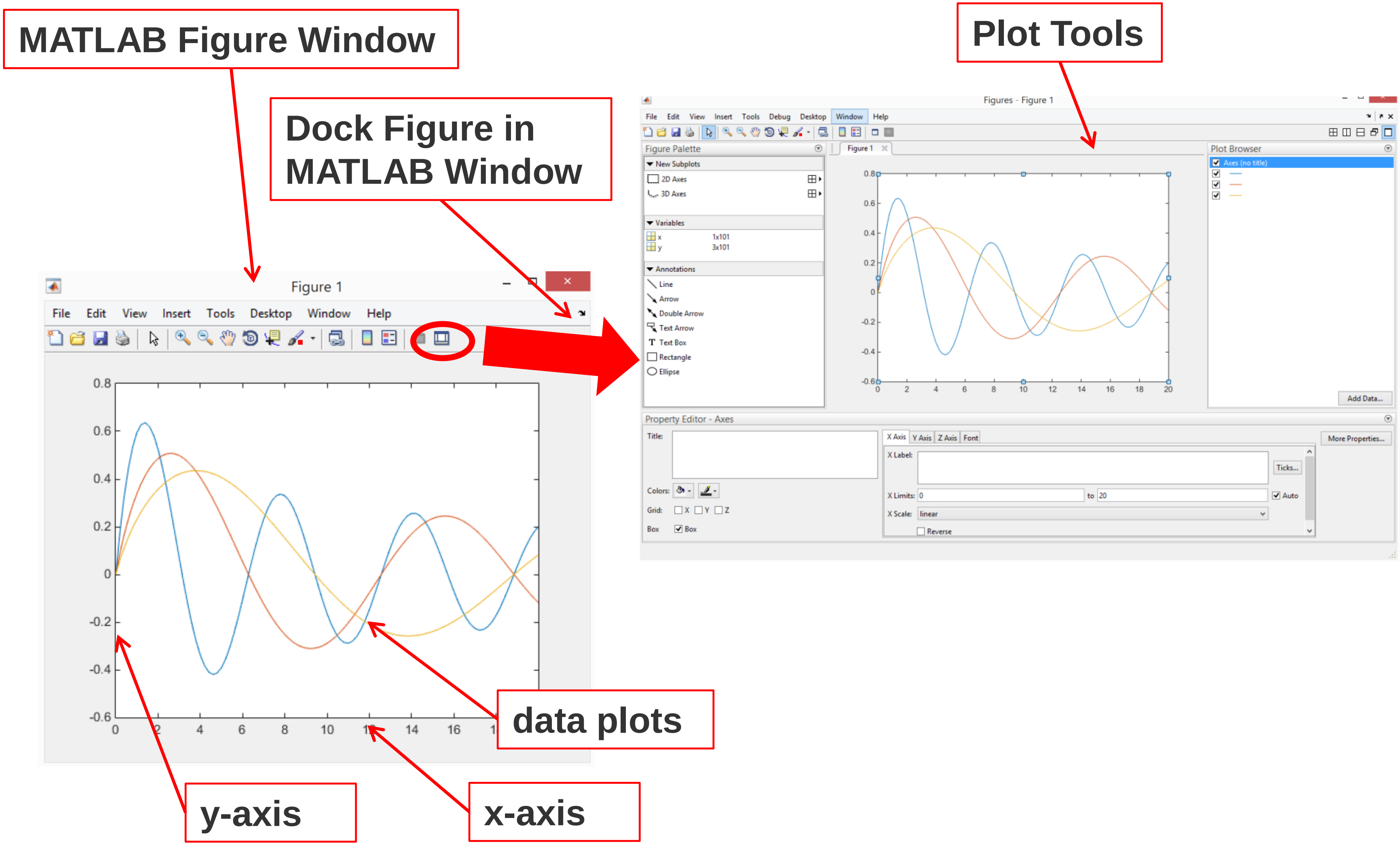
Demo

```
>> 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!

Plotting Tools



MATLAB Figure Window

Dock Figure in MATLAB Window

Plot Tools

data plots

y-axis

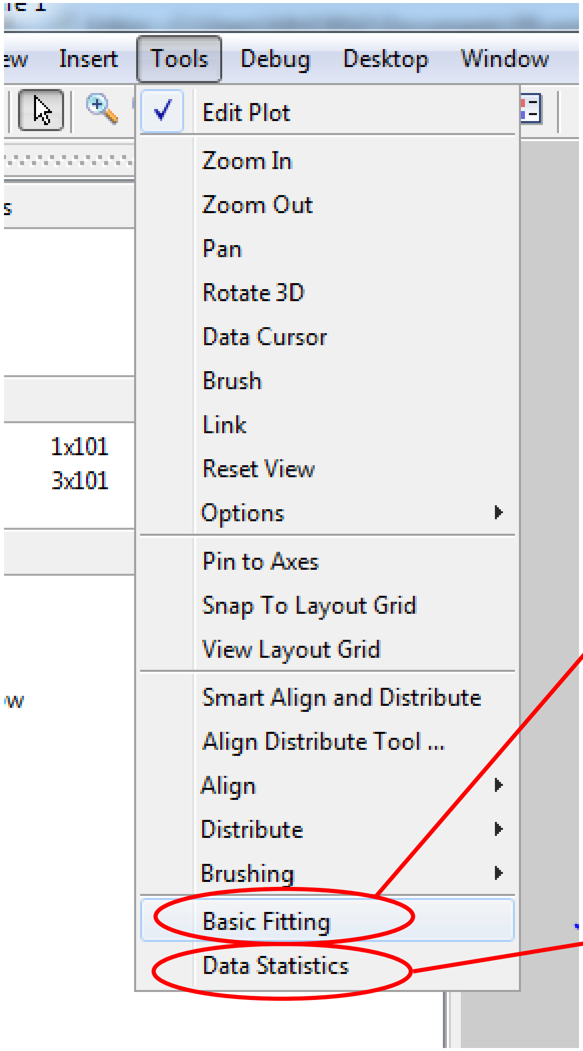
x-axis

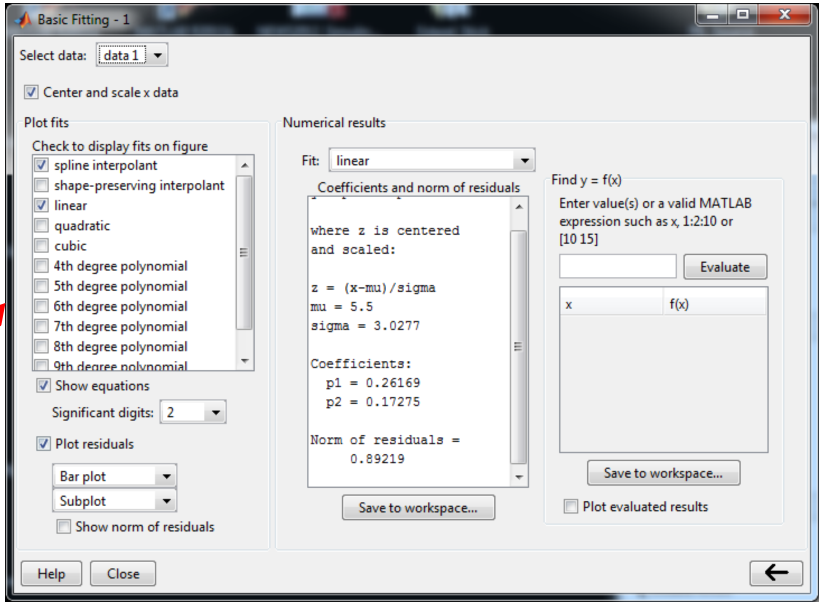
Variables
x 1x101
y 3x101

Property Editor - Axes
Title:
Grid: ☐ X ☐ Y ☐ Z
Box ☑ Box
X Label:
X Limits: 0 to 20 ☑ Auto
X Scale: linear
☐ Reverse

Plot Browser
☑ Axes (no title)
☑
☑
☑

Data Adjustment





Select data: data 1

☑ Center and scale x data

Plot fits

Check to display fits on figure

- ☑ spline interpolant
- □ shape-preserving interpolant
- ☑ linear
- □ quadratic
- □ cubic
- □ 4th degree polynomial
- □ 5th degree polynomial
- □ 6th degree polynomial
- □ 7th degree polynomial
- □ 8th degree polynomial
- □ 9th degree polynomial

☑ Show equations

Significant digits: 2

☑ Plot residuals

Bar plot

Subplot

□ Show norm of residuals

Numerical results

Fit: linear

Coefficients and norm of residuals

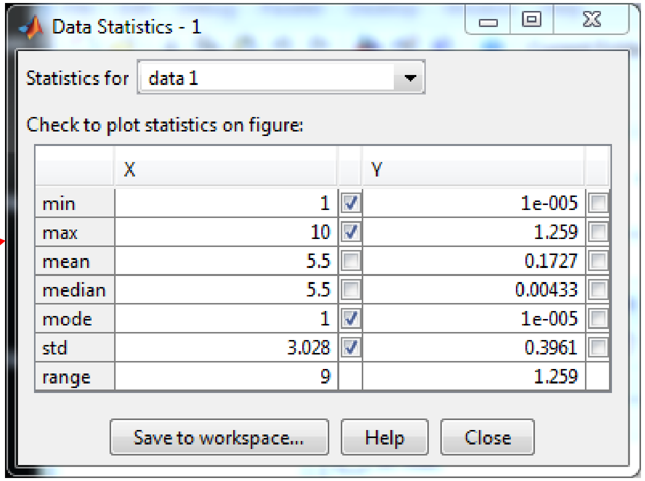
```
where z is centered
and scaled:

z = (x-mu)/sigma
mu = 5.5
sigma = 3.0277

Coefficients:
  p1 = 0.26169
  p2 = 0.17275

Norm of residuals =
     0.89219
```

Find y = f(x)

Enter value(s) or a valid MATLAB expression such as x, 1:2:10 or [10 15]

x	f(x)

□ Plot evaluated results

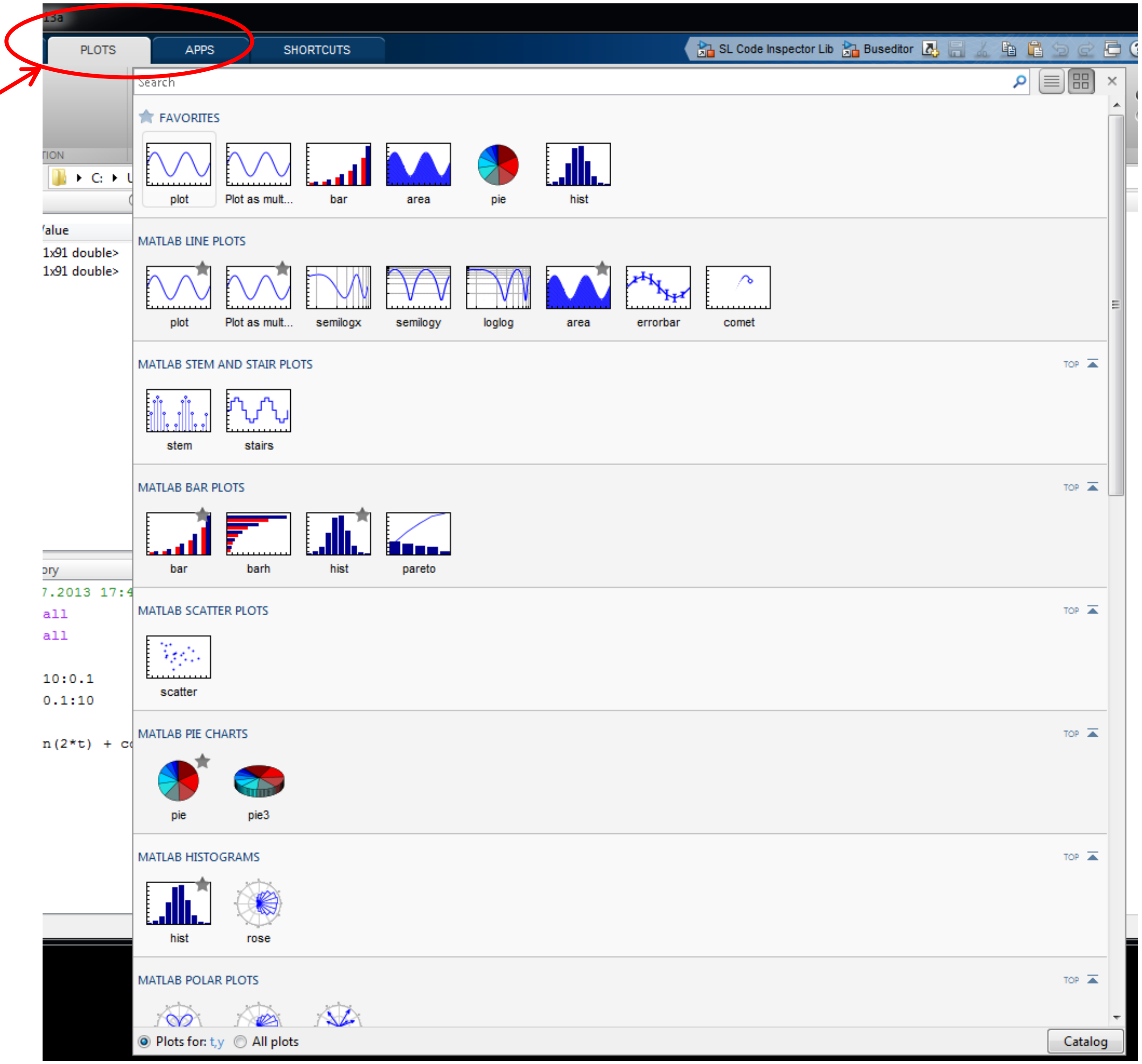


Statistics for data 1

Check to plot statistics on figure:

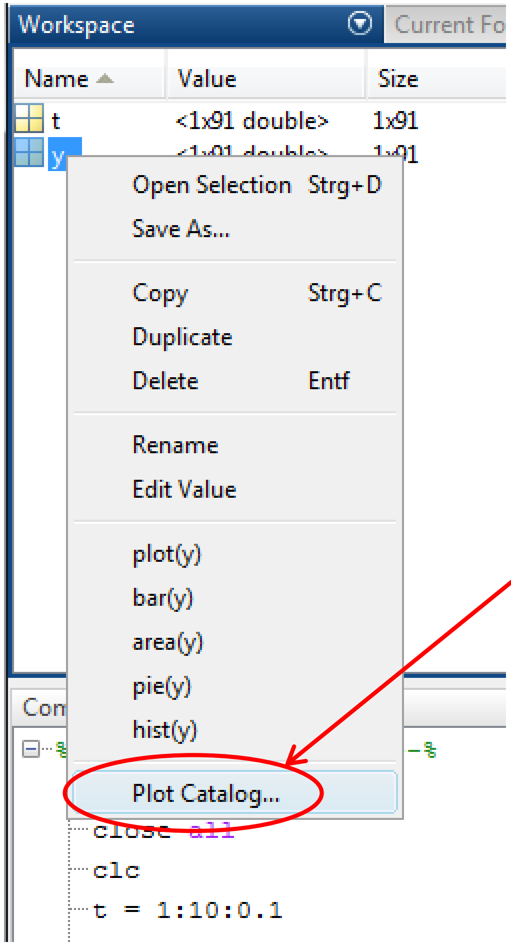
	X		Y	
min	1	☑	1e-005	□
max	10	☑	1.259	□
mean	5.5	□	0.1727	□
median	5.5	□	0.00433	□
mode	1	☑	1e-005	□
std	3.028	☑	0.3961	□
range	9		1.259	

Plots Menu



Plots Menu

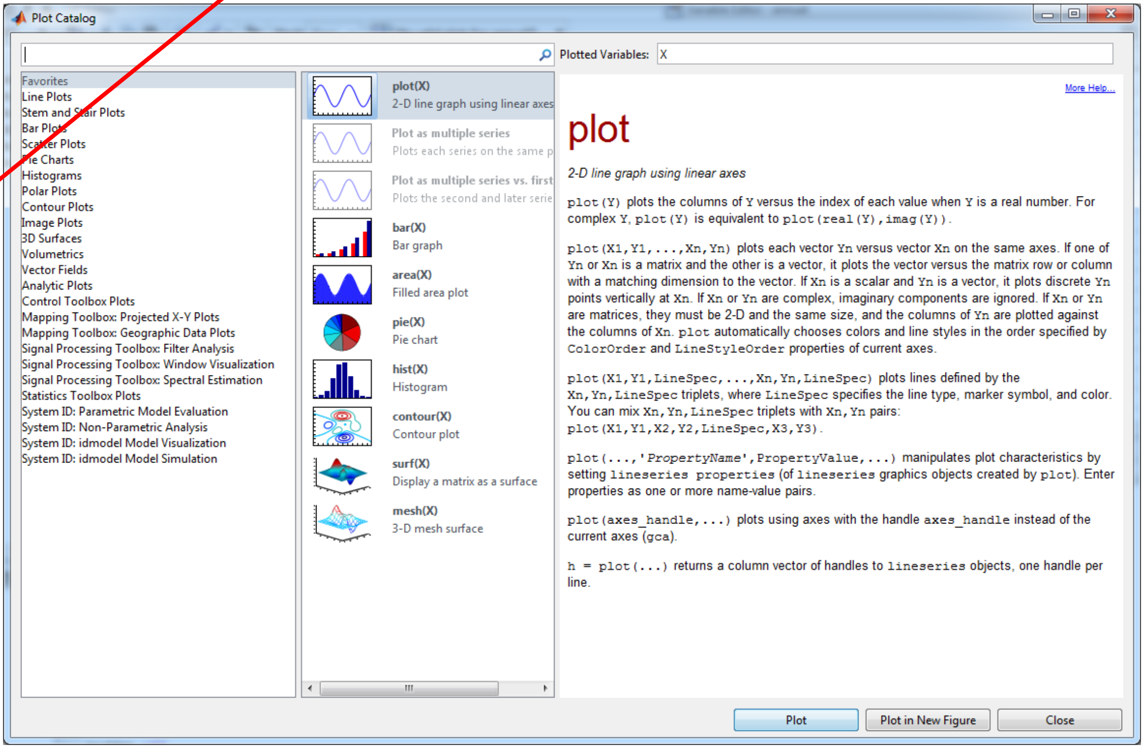
Plotting from Workspace Browser



Name	Value	Size
t	<1x91 double>	1x91
y	<1x91 double>	1x91

```
close all
clc
t = 1:10:0.1
```

Context menu of variable or MATLAB menu opens Plot Catalog


Plot Catalog
Plotted Variables: X
Favorites
Line Plots
Stem and Stair Plots
Bar Plots
Scatter Plots
Pie Charts
Histograms
Polar Plots
Contour Plots
Image Plots
3D Surfaces
Volumetrics
Vector Fields
Analytic Plots
Control Toolbox Plots
Mapping Toolbox: Projected X-Y Plots
Mapping Toolbox: Geographic Data Plots
Signal Processing Toolbox: Filter Analysis
Signal Processing Toolbox: Window Visualization
Signal Processing Toolbox: Spectral Estimation
Statistics Toolbox Plots
System ID: Parametric Model Evaluation
System ID: Non-Parametric Analysis
System ID: idmodel Model Visualization
System ID: idmodel Model Simulation
plot(X)
2-D line graph using linear axes
Plot as multiple series
Plots each series on the same p
Plot as multiple series vs. first
Plots the second and later serie
bar(X)
Bar graph
area(X)
Filled area plot
pie(X)
Pie chart
hist(X)
Histogram
contour(X)
Contour plot
surf(X)
Display a matrix as a surface
mesh(X)
3-D mesh surface
More Help...
Plot
Plot in New Figure
Close

plot

2-D line graph using linear axes

`plot(Y)` plots the columns of `Y` versus the index of each value when `Y` is a real number. For complex `Y`, `plot(Y)` is equivalent to `plot(real(Y),imag(Y))`.

`plot(X1,Y1,...,Xn,Yn)` plots each vector `Yn` versus vector `Xn` on the same axes. If one of `Yn` or `Xn` is a matrix and the other is a vector, it plots the vector versus the matrix row or column with a matching dimension to the vector. If `Xn` is a scalar and `Yn` is a vector, it plots discrete `Yn` points vertically at `Xn`. If `Xn` or `Yn` are complex, imaginary components are ignored. If `Xn` or `Yn` are matrices, they must be 2-D and the same size, and the columns of `Yn` are plotted against the columns of `Xn`. `plot` automatically chooses colors and line styles in the order specified by `ColorOrder` and `LineStyleOrder` properties of current axes.

`plot(X1,Y1,LineSpec,...,Xn,Yn,LineSpec)` plots lines defined by the `Xn,Yn,LineSpec` triplets, where `LineSpec` specifies the line type, marker symbol, and color. You can mix `Xn,Yn,LineSpec` triplets with `Xn,Yn` pairs: `plot(X1,Y1,X2,Y2,LineSpec,X3,Y3)`.

`plot(...,'PropertyName',PropertyValue,...)` manipulates plot characteristics by setting `lineseries properties` (of `lineseries` graphics objects created by `plot`). Enter properties as one or more name-value pairs.

`plot(axes_handle,...)` plots using axes with the handle `axes_handle` instead of the current axes (`gca`).

`h = plot(...)` returns a column vector of handles to `lineseries` objects, one handle per line.

Angermann et al. p. 43

Graphics (general)

figure [(*number*)]	creation (call) of a figure
subplot (*line*, *colum*, *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

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
`grid [on | off]`	grid 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

Angermann et al. p. 45

Colors

k	black	r	red
b	blue	m	magenta
c	cyan	y	yellow
g	green	w	white

Markers

.	point
o	circle
*	asterisk
+, x	cross

Lines

-	solid line (default)
--	dashed line
-.	dash-dot line
:	dotted line

Angermann et al. p. 49

Graphics : 3D plot commands

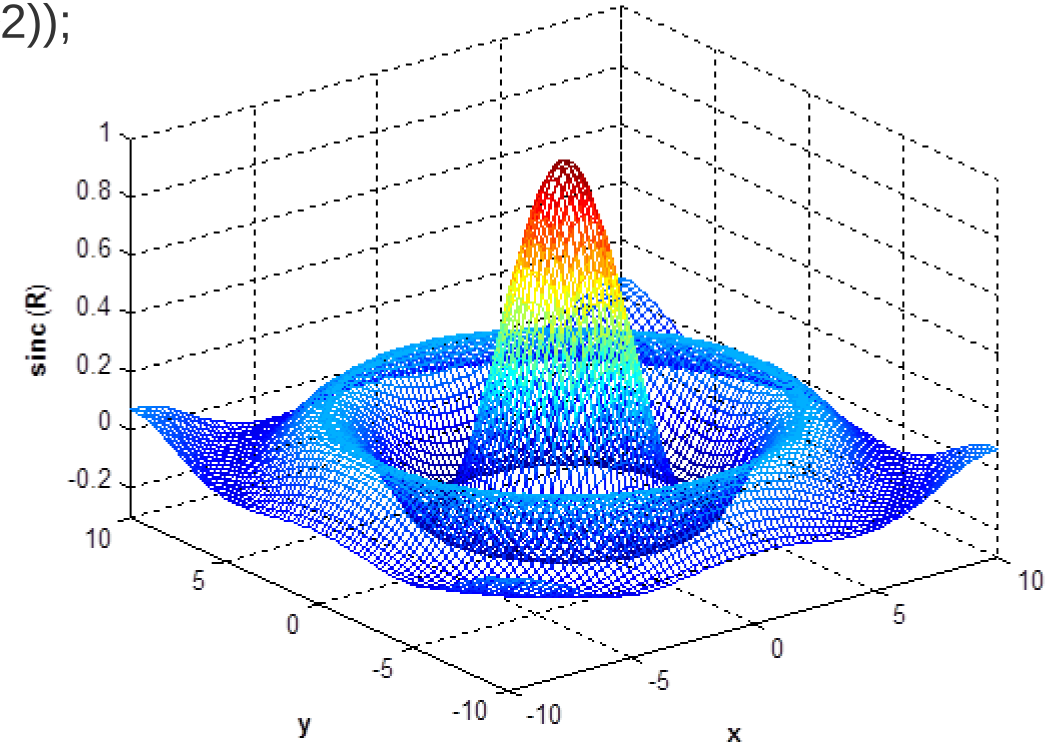
[X,Y] = meshgrid(*xvector*, *yvector*)	rectangular coordinate grid matrix
plot3(*xvalues*,*yvalues*,*zvalues*…[,*plotstyle*])	*3D*-plot, points/lines
surf(*xvalues*,*yvalues*,*zvalues*…[,*color*])	*3D*-plot, surface
mesh(*xvalues*,*yvalues*,*zvalues*…[,*color*])	*3D*-plot, mesh
waterfall(*xvalues*,*yvalues*,*zvalues*…[…])	*3D*-plot, waterfall
contour(*xvalues*,*yvalues*,*zvalues*…[…])	*2D*-plot, contour lines/ level curves
box [on | off]	show box
rotate3d [on | off]	interactive rotating
view(*horizontal*, *vertical*)	change perpective
zlabel(*string*)	z-axis label

Color settings

colormap(*name*)	choose colormap
caxis(*color_min*, *color_max*)	color scaling

3D Plots

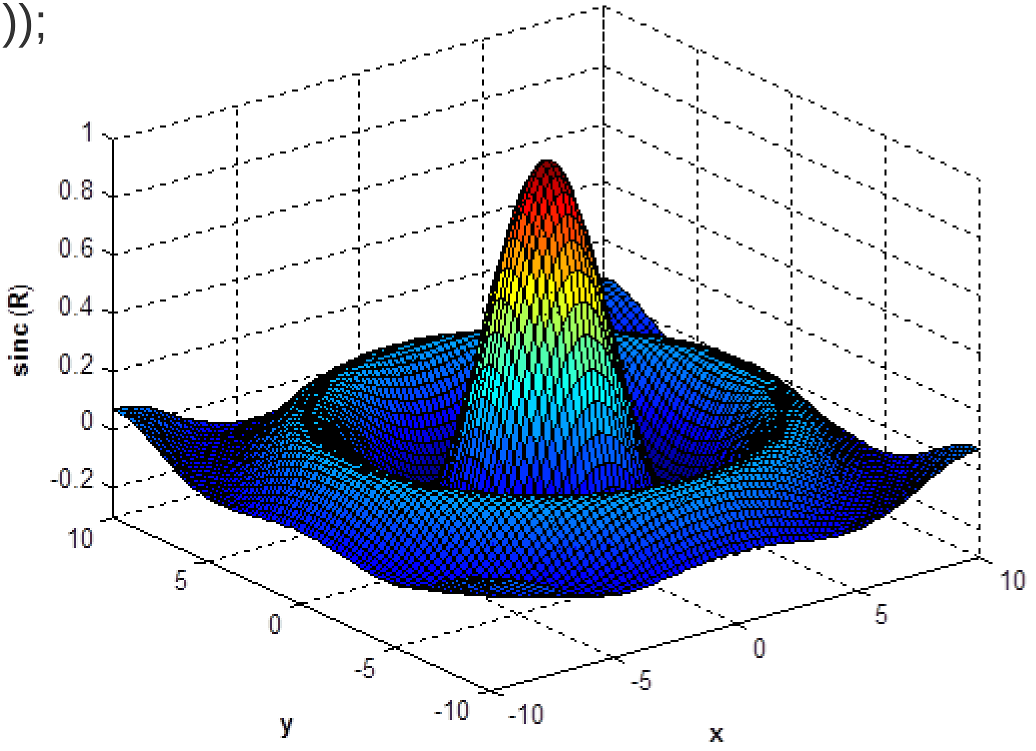
```
>> [X,Y] = meshgrid(-10:0.25:10,-10:0.25:10);
>> f = sinc(sqrt((X/pi).^2+(Y/pi).^2));
>> mesh(X,Y,f);
>> axis([-10 10 -10 10 -0.3 1])
>> xlabel('{\bfx}')
>> ylabel('{\bfy}')
>> zlabel('{\bfsinc} ({\bfR})')
>> hidden off
```



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

3D Plots

```
>> [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}')
>> zlabel('{\bfsinc} ({\bfR})')
>> hidden off
```



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

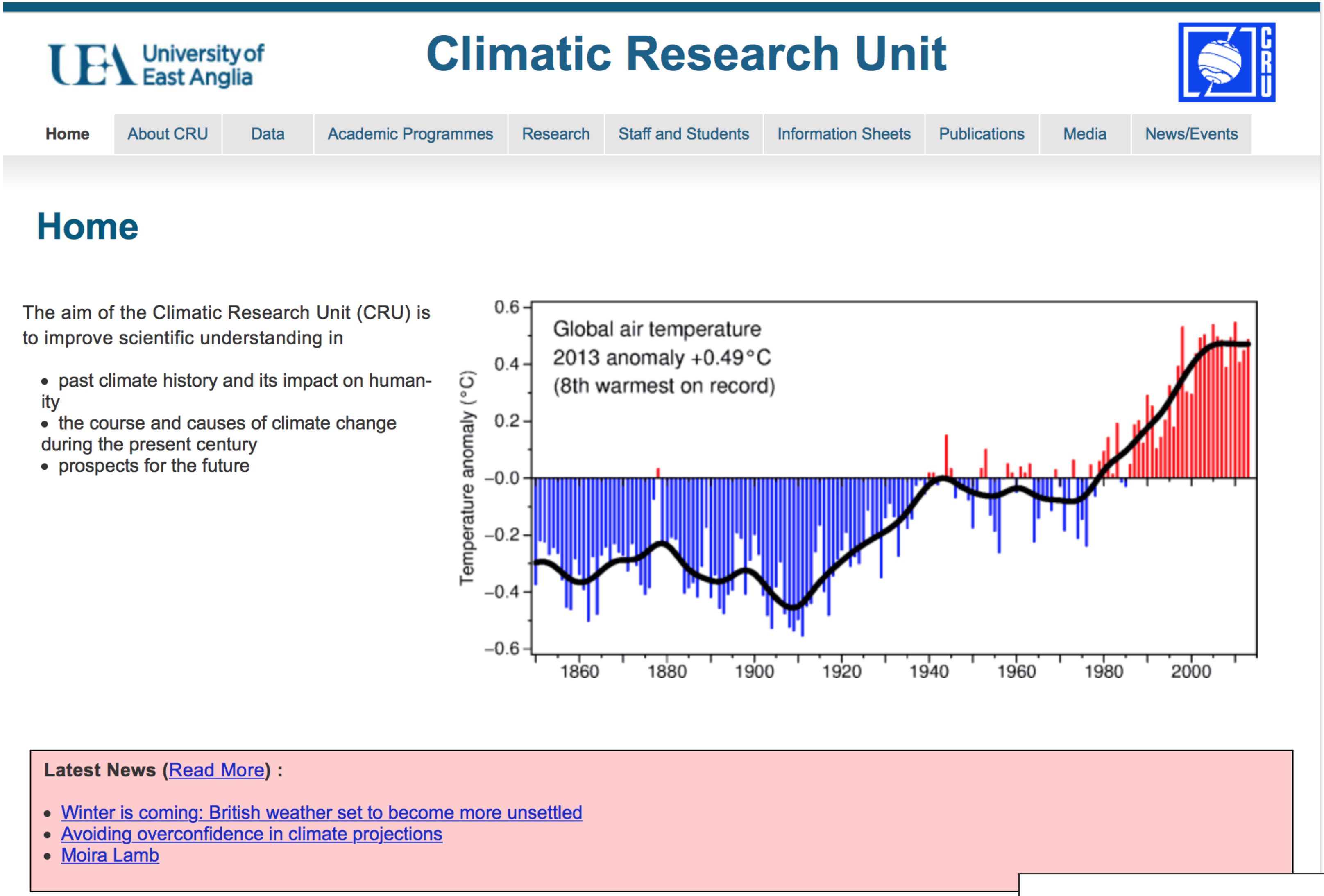
Outline
Introduction to MATLAB

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

Exercise: Consider Global Warming!



University of East Anglia

Climatic Research Unit

Home | About CRU | Data | Academic Programmes | Research | Staff and Students | Information Sheets | Publications | Media | News/Events

Home

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

Latest News (Read More) :

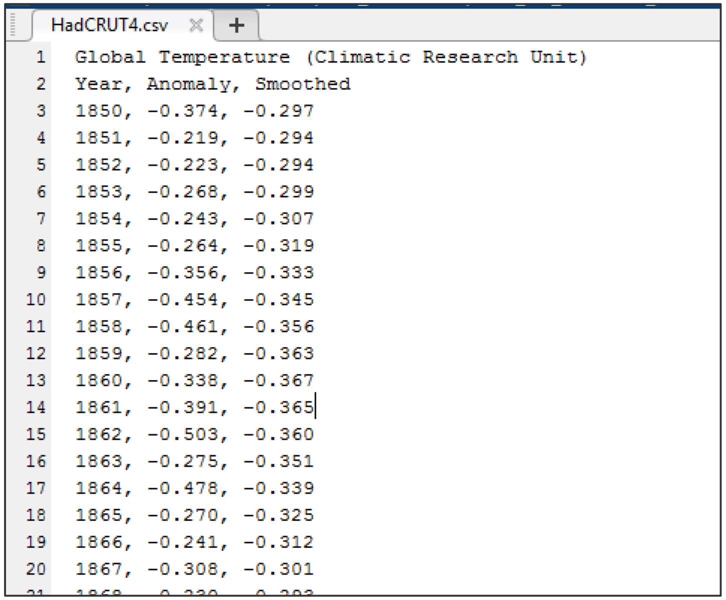
- Winter is coming: British weather set to become more unsettled
- Avoiding overconfidence in climate projections
- Moira Lamb

www.cru.uea.ac.uk/

Exercise: Is the temperature rising?

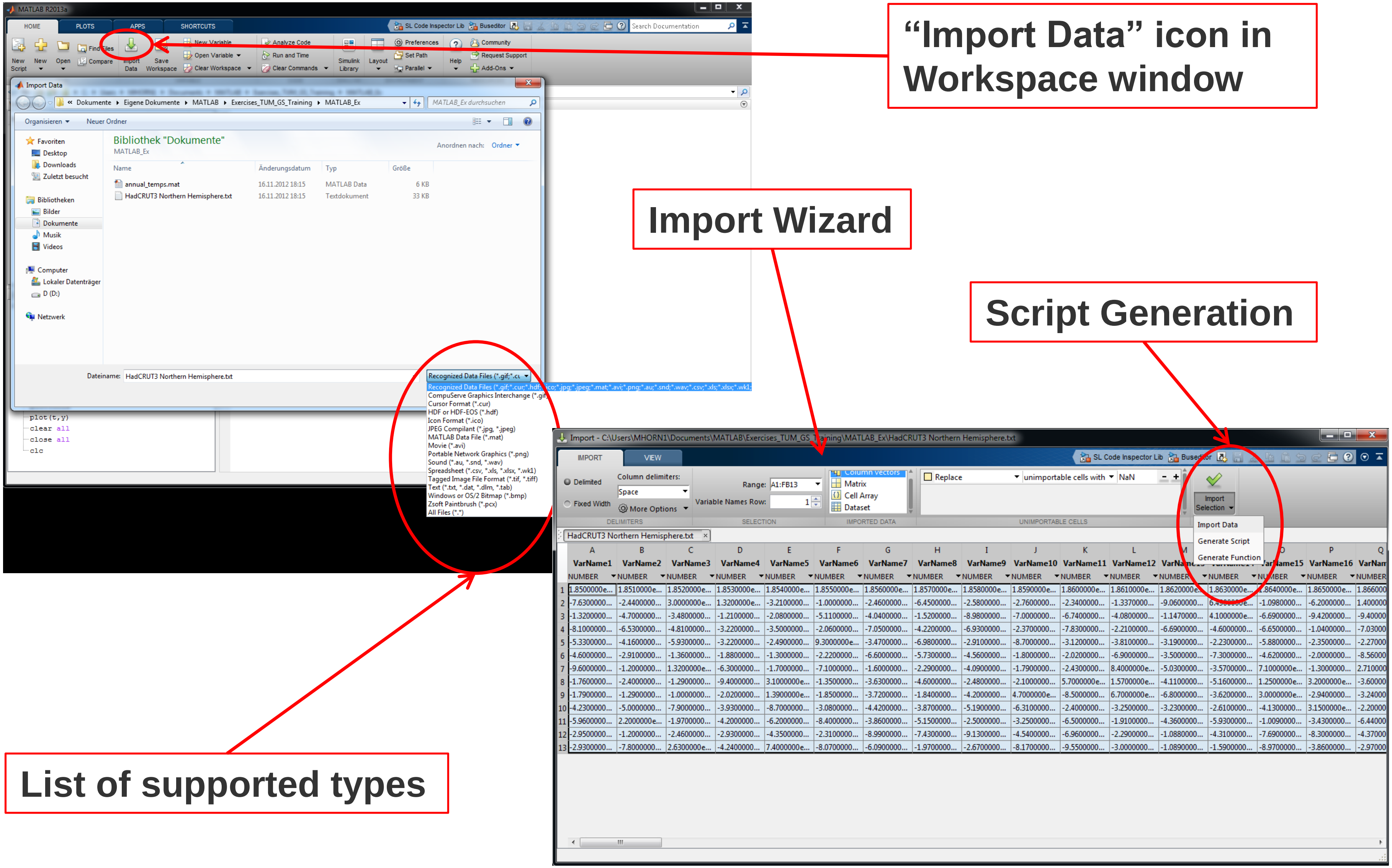
- University of East Anglia, Norwich, UK, Climatic Research Unit
- Study on global warming
- Measurement series on combined global land and marine surface temperature record from 1850 to 2013

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



```
Global Temperature (Climatic Research Unit)
Year, Anomaly, Smoothed
1850, -0.374, -0.297
1851, -0.219, -0.294
1852, -0.223, -0.294
1853, -0.268, -0.299
1854, -0.243, -0.307
1855, -0.264, -0.319
1856, -0.356, -0.333
1857, -0.454, -0.345
1858, -0.461, -0.356
1859, -0.282, -0.363
1860, -0.338, -0.367
1861, -0.391, -0.365
1862, -0.503, -0.360
1863, -0.275, -0.351
1864, -0.478, -0.339
1865, -0.270, -0.325
1866, -0.241, -0.312
1867, -0.308, -0.301
```

Exercise: Importing Data from .txt File



Name	Änderungsdatum	Typ	Größe
annual_temps.mat	16.11.2012 18:15	MATLAB Data	6 KB
HadCRUT3 Northern Hemisphere.txt	16.11.2012 18:15	Textdokument	33 KB

Dateiname: HadCRUT3 Northern Hemisphere.txt

```
plot(t,y)
clear all
close all
clc
```

Import - C:\Users\MHORN1\Documents\MATLAB\Exercises_TUM_GS_Training\MATLAB_Ex\HadCRUT3 Northern Hemisphere.txt

● Delimited

Column delimiters: Space

Range: A1:FB13

Variable Names Row: 1

□ Replace unimportable cells with NaN

HadCRUT3 Northern Hemisphere.txt

	A	B	C	D	E	F	G	H	I
	VarName1	VarName2	VarName3	VarName4	VarName5	VarName6	VarName7	VarName8	VarName9
	NUMBER	NUMBER	NUMBER	NUMBER	NUMBER	NUMBER	NUMBER	NUMBER	NUMBER
1	1.8500000e...	1.8510000e...	1.8520000e...	1.8530000e...	1.8540000e...	1.8550000e...	1.8560000e...	1.8570000e...	1.8580000e...
2	-7.6300000...	-2.4400000...	3.0000000e...	1.3200000e...	-3.2100000...	-1.0000000...	-2.4600000...	-6.4500000...	-2.5800000...
3	-1.3200000...	-4.7000000...	-3.4800000...	-1.2100000...	-2.0800000...	-5.1100000...	-4.0400000...	-1.5200000...	-8.9800000...
4	-8.1000000...	-6.5300000...	-4.8100000...	-3.2200000...	-3.5000000...	-2.0600000...	-7.0500000...	-4.2200000...	-6.9300000...
5	-5.3300000...	-4.1600000...	-5.9300000...	-3.2200000...	-2.4900000...	9.3000000e...	-3.4700000...	-6.9800000...	-2.9100000...
6	-4.6000000...	-2.9100000...	-1.3600000...	-1.8800000...	-1.3000000...	-2.2200000...	-6.6000000...	-5.7300000...	-4.5600000...
7	-9.6000000...	-1.2000000...	1.3200000e...	-6.3000000...	-1.7000000...	-7.1000000...	-1.6000000...	-2.2900000...	-4.0900000...
8	-1.7600000...	-2.4000000...	-1.2900000...	-9.4000000...	3.1000000e...	-1.3500000...	-3.6300000...	-4.6000000...	-2.4800000...
9	-1.7900000...	-1.2900000...	-1.0000000...	-2.0200000...	1.3900000e...	-1.8500000...	-3.7200000...	-1.8400000...	-4.2000000...
10	-4.2300000...	-5.0000000...	-7.9000000...	-3.9300000...	-8.7000000...	-3.0800000...	-4.4200000...	-3.8700000...	-5.1900000...
11	-5.9600000...	2.2000000e...	-1.9700000...	-4.2000000...	-6.2000000...	-8.4000000...	-3.8600000...	-5.1500000...	-2.5000000...
12	-2.9500000...	-1.2000000...	-2.4600000...	-2.9300000...	-4.3500000...	-2.3100000...	-8.9900000...	-7.4300000...	-9.1300000...
13	-2.9300000...	-7.8000000...	2.6300000e...	-4.2400000...	7.4000000e...	-8.0700000...	-6.0900000...	-1.9700000...	-2.6700000...

	J	K	L	M	[illegible]	O	P	Q
	VarName10	VarName11	VarName12	VarNa[illegible]	[illegible]	[illegible]ame15	VarName16	VarNam
	NUMBER	NUMBER	NUMBER	NUMBER	NUMBER	NU[illegible]BER	NUMBER	NUMBER
1	1.8590000e...	1.8600000e...	1.8610000e...	1.8620000e[illegible]	1.8630000e...	[illegible]1.8640000e...	1.8650000e...	1.866000
2	-2.7600000...	-2.3400000...	-1.3370000...	-9.0600000...	6.4[illegible]e...	-1.0980000...	-6.2000000...	1.400000
3	-7.0000000...	-6.7400000...	-4.0800000...	-1.1470000...	4.1000000e...	-6.6900000...	-9.4200000...	-9.40000
4	-2.3700000...	-7.8300000...	-2.2100000...	-6.6900000...	-4.6000000...	-6.6500000...	-1.0400000...	-7.03000
5	-8.7000000...	-3.1200000...	-3.8100000...	-3.1900000...	-2.2300000...	-5.8800000...	-2.3500000...	-2.27000
6	-1.8000000...	-2.0200000...	-6.9000000...	-3.5000000...	-7.3000000...	-4.6200000...	-2.0000000...	-8.56000
7	-1.7900000...	-2.4300000...	8.4000000e...	-5.0300000...	-3.5700000...	7.1000000e...	-1.3000000...	2.710000
8	-2.1000000...	5.7000000e...	1.5700000e...	-4.1100000...	-5.1600000...	1.2500000e...	3.2000000e...	-3.60000
9	4.7000000e...	-8.5000000...	6.7000000e...	-6.8000000...	-3.6200000...	3.0000000e...	-2.9400000...	-3.24000
10	-6.3100000...	-2.4000000...	-3.2500000...	-3.2300000...	-2.6100000...	-4.1300000...	3.1500000e...	-2.20000
11	-3.2500000...	-6.5000000...	-1.9100000...	-4.3600000...	-5.9300000...	-1.0090000...	-3.4300000...	-6.44000
12	-4.5400000...	-6.9600000...	-2.2900000...	-1.0880000...	-4.3100000...	-7.6900000...	-8.3000000...	-4.37000
13	-8.1700000...	-9.5500000...	-3.0000000...	-1.0890000...	-1.5900000...	-8.9700000...	-3.8600000...	-2.97000

“Import Data” icon in Workspace window

Import Wizard

Script Generation

List of supported types

Angermann et al. p. 37

File import and export standard formats

load *file* [*variable* …]	load variables from MAT-File
save *file* [*variabel* …]	safe variables in MAT-File
[*variable* =] load *file.ending*	load from ASCII-File
save *file.ending -ascii [*variable*]*	*save variables in ASCII-File*
variable = xlsread('*file.xls')*	load data from Excel-File
xlswrite('*file.xls', variable*)	save data to Excel-File

Angermann et al. p. 39

Formatted data import and export

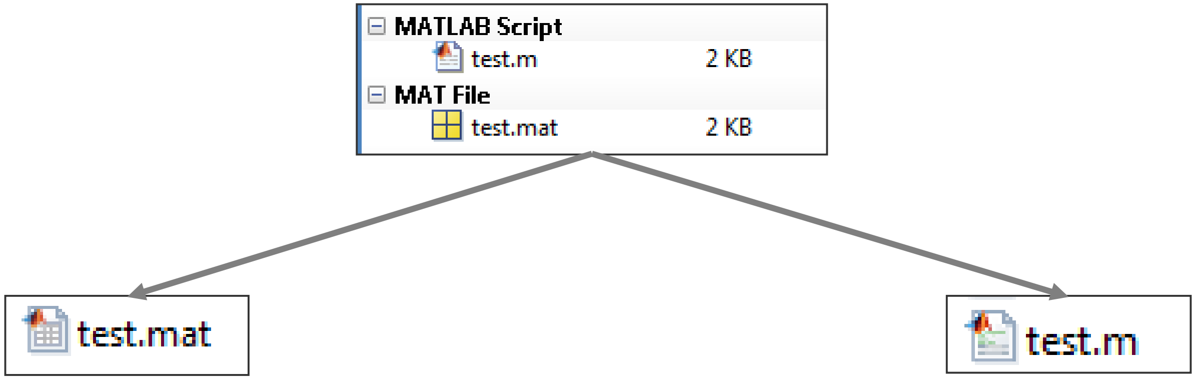
`fid = fopen(‘file.ending’, ‘permission’)`	open file
`fclose(fid)`	close file
`fprintf(fid, ‘format’, variable[,…])`	write formatted data
`vector = fscanf(fid, ‘format’)`	read formatted data
`string = fgetl(fid)`	read line
`string = fgets(fid,n)`	read `n` characters
`cellarray = textscan(fid, ‘format’[, number][, ‘parameter’, value, …])`	
`variable = textread(‘file’, ‘format’[, ‘parameter’, value, …])`	
`variable = dlmread(‘file’, ‘delimiter’[, ‘range’])`	

Angermann et al. p. 39

Binary data import and export

vector = fread(*fid*, ‘*format*’)	read data
fwrite(*fid*, *matrix*, ‘*format*’)	write data
uchar, uint16, uint32, uint64	unsigned formats
int8, int16, int32, int64	signed formats
float32, float64	floating point formats
bit*N*, ubit*N*, 1<=*N*<=64	N signed or unsigned bits

Saving Data

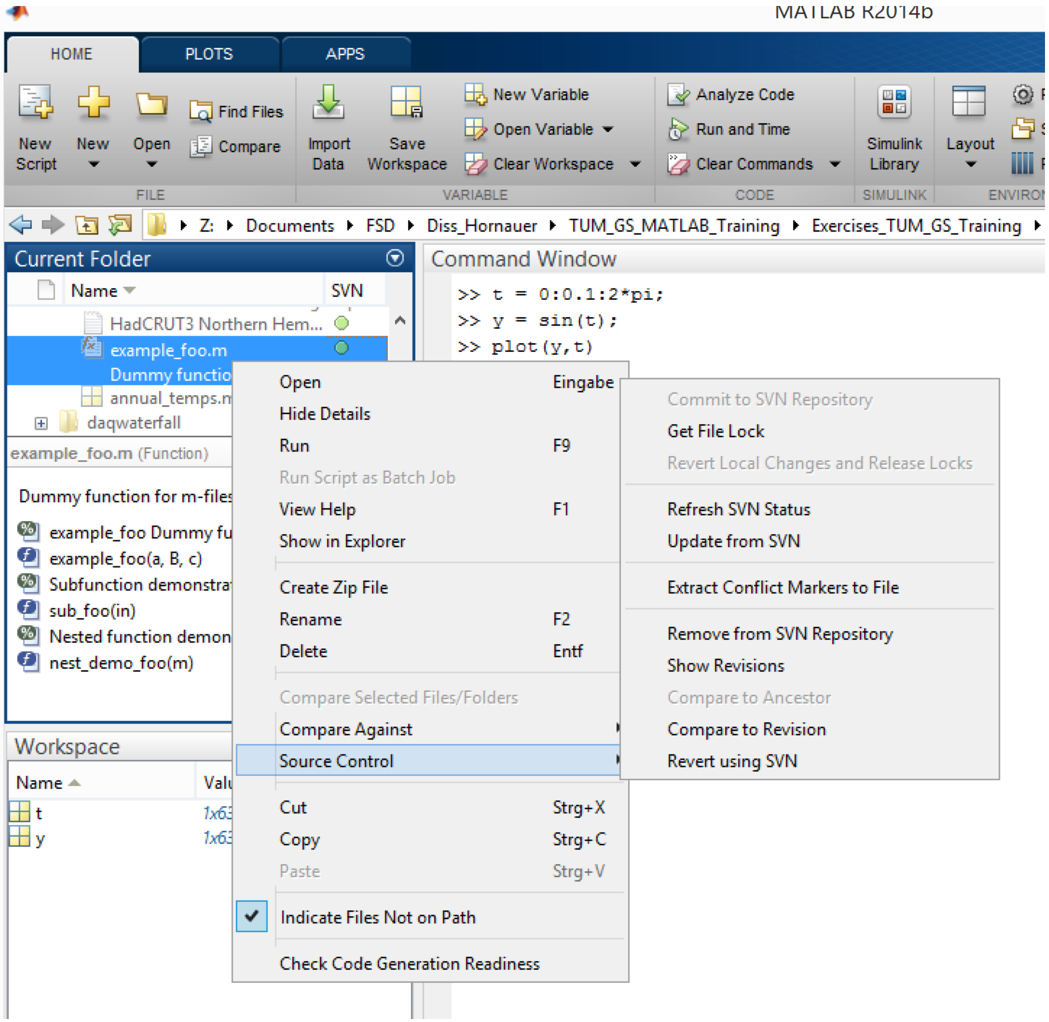


⊟ MATLAB Script	
test.m	2 KB
⊟ MAT File	
test.mat	2 KB

- .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)

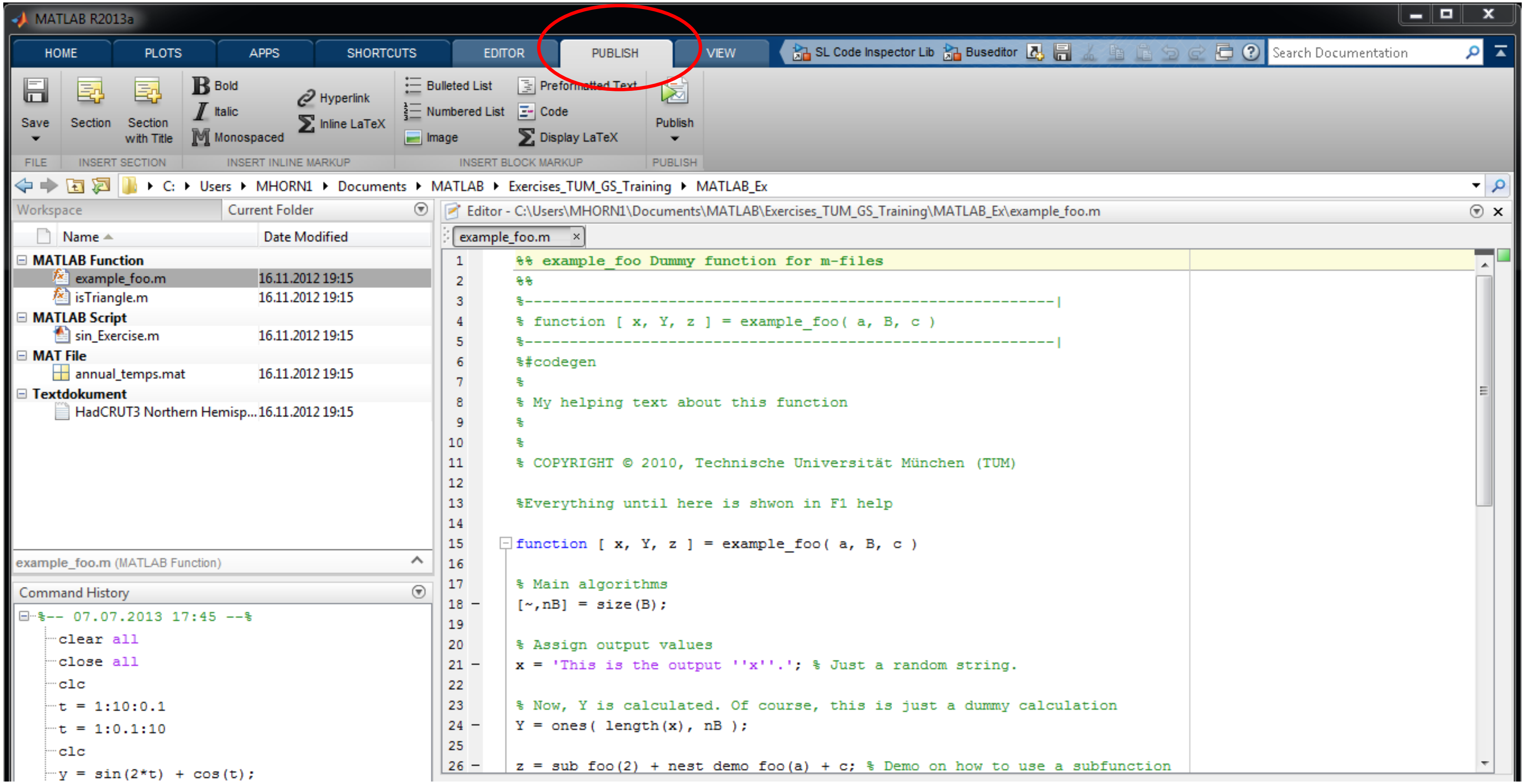
Managing Data



```
>> t = 0:0.1:2*pi;
>> y = sin(t);
>> plot(y,t)
```

- native interface to version control systems like SVN or Git
- source control in current folder explorer
- http://www.mathworks.com/help/matlab/source-control.html

Publish Function



Name	Date Modified
MATLAB Function	
example_foo.m	16.11.2012 19:15
isTriangle.m	16.11.2012 19:15
MATLAB Script	
sin_Exercise.m	16.11.2012 19:15
MAT File	
annual_temps.mat	16.11.2012 19:15
Textdokument	
HadCRUT3 Northern Hemisp...	16.11.2012 19:15

```
%-- 07.07.2013 17:45 --%
clear all
close all
clc
t = 1:10:0.1
t = 1:0.1:10
clc
y = sin(2*t) + cos(t);
```

```
%% 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 shwon 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
```

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

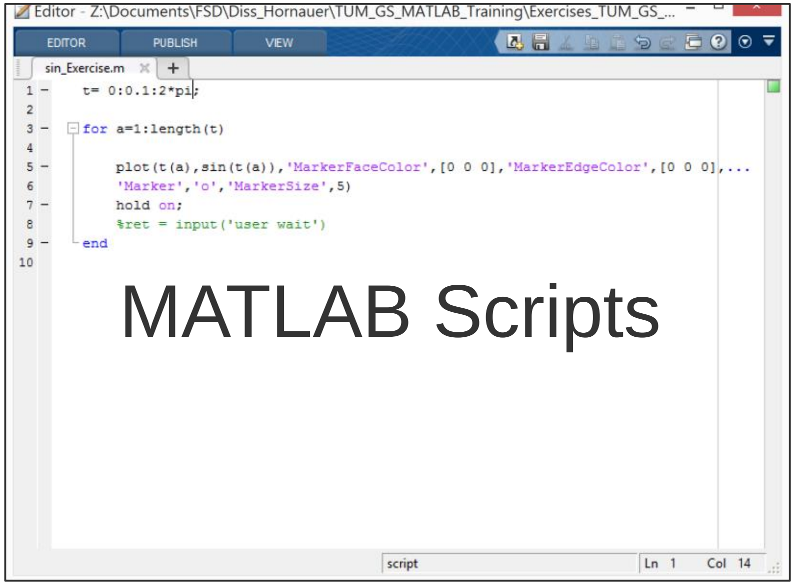
Toolboxes:

1) Symbolic Math Toolbox
2) Control System Toolbox and Curve Fitting Toolbox

MATLAB Program Files

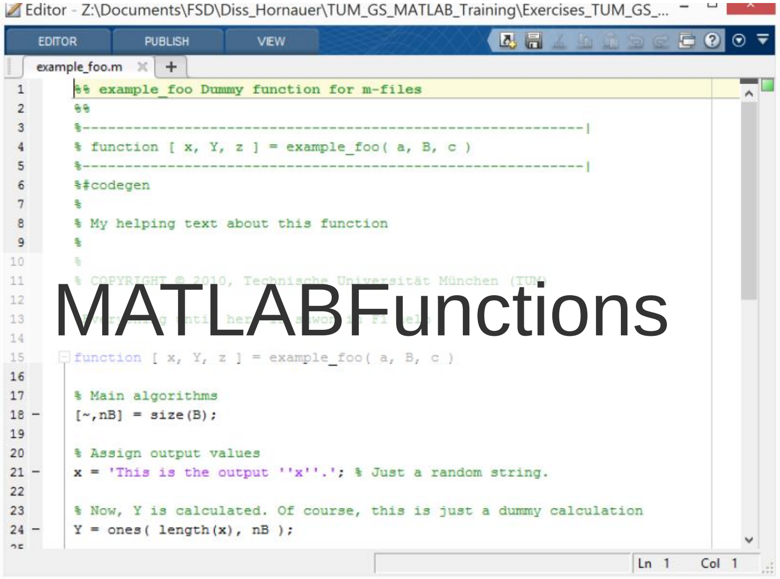
Next steps:

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


Editor - Z:\Documents\FSD\Diss_Hornauer\TUM_GS_MATLAB_Training\Exercises_TUM_GS_...
EDITOR
PUBLISH
VIEW
sin_Exercise.m
script
Ln 1
Col 14

```
t= 0:0.1:2*pi;

for a=1:length(t)

    plot(t(a),sin(t(a)),'MarkerFaceColor',[0 0 0],'MarkerEdgeColor',[0 0 0],...
    'Marker','o','MarkerSize',5)
    hold on;
    %ret = input('user wait')
end

```

MATLAB Scripts


Editor - Z:\Documents\FSD\Diss_Hornauer\TUM_GS_MATLAB_Training\Exercises_TUM_GS_...
EDITOR
PUBLISH
VIEW
example_foo.m
Ln 1
Col 1

```
%% 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 );
```

MATLABFunctions

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'
```

Angermann et al. p. 21

Conditional execution, control flow and loops

`if … [elseif …][else] end`	if-statement
`switch … case … [otherwise …] end`	switch-statement
`for` *variable=start:stepsize:end* *commands* `end`	for loop
`while` *condition commands* `end`	while loop

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

Angermann et al. p. 23

Scripts

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

Angermann et al. p. 36

User dialog

variable = input(*string*)	request user input for variable variable by displaying the prompt string
string = input(*string*, ‘s’)	request user input of a string
string = num2str(variable[, format])	convert number to string
string = sprintf(*string*, *variable*)	create formatted string
disp(*string*)	display text on screen

Escape characters

\n	line break
\t	tabulator
\\	backslash
%%	percent sign %
‘’	single quotation mark ‘

Formatting (conversion characters)

%d	signed integer (i.e. 321)
%x	base 16 (hexadecimal) whole number
%5.2f	floating point number (i.e. 54.21)
%.2e	exponential notation (i.e. 5.42e+001)
%s	string

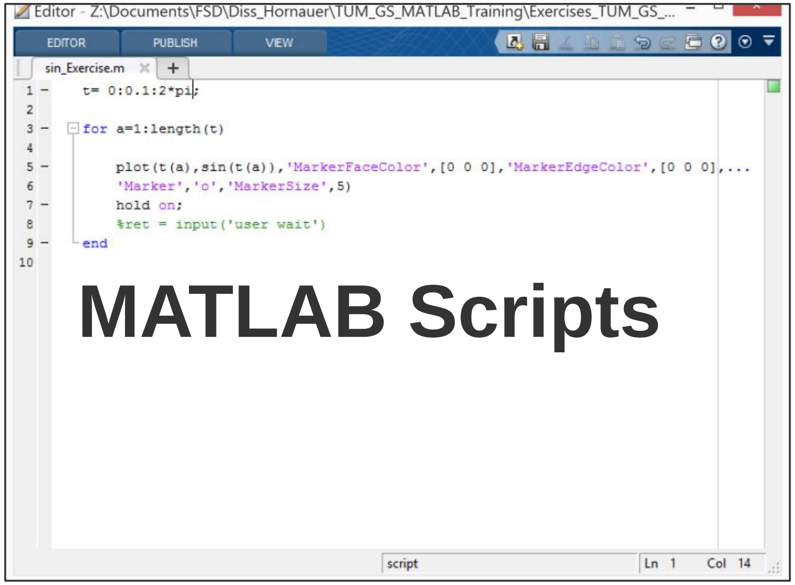
Angermann et al. p. 40

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

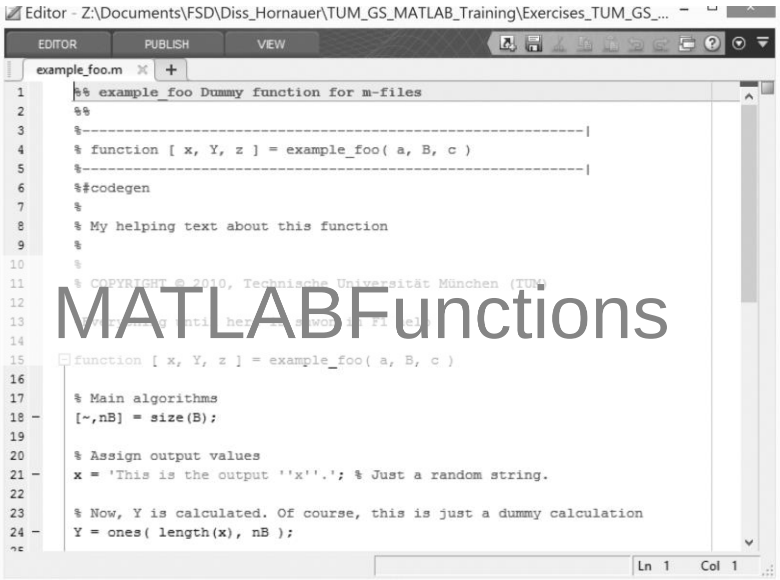
MATLAB Program Files

- Why?
  - Automating
  - Editing/Debugging
  - Deploying as applications


Editor - Z:\Documents\FSD\Diss_Hornauer\TUM_GS_MATLAB_Training\Exercises_TUM_GS_...
EDITOR
PUBLISH
VIEW
sin_Exercise.m
script
Ln 1
Col 14

```
t= 0:0.1:2*pi;

for a=1:length(t)

    plot(t(a),sin(t(a)),'MarkerFaceColor',[0 0 0],'MarkerEdgeColor',[0 0 0],...
    'Marker','o','MarkerSize',5)
    hold on;
    %ret = input('user wait')
end
```

MATLAB Scripts


Editor - Z:\Documents\FSD\Diss_Hornauer\TUM_GS_MATLAB_Training\Exercises_TUM_GS_...
EDITOR
PUBLISH
VIEW
example_foo.m
Ln 1
Col 1

```
%% 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 );
```

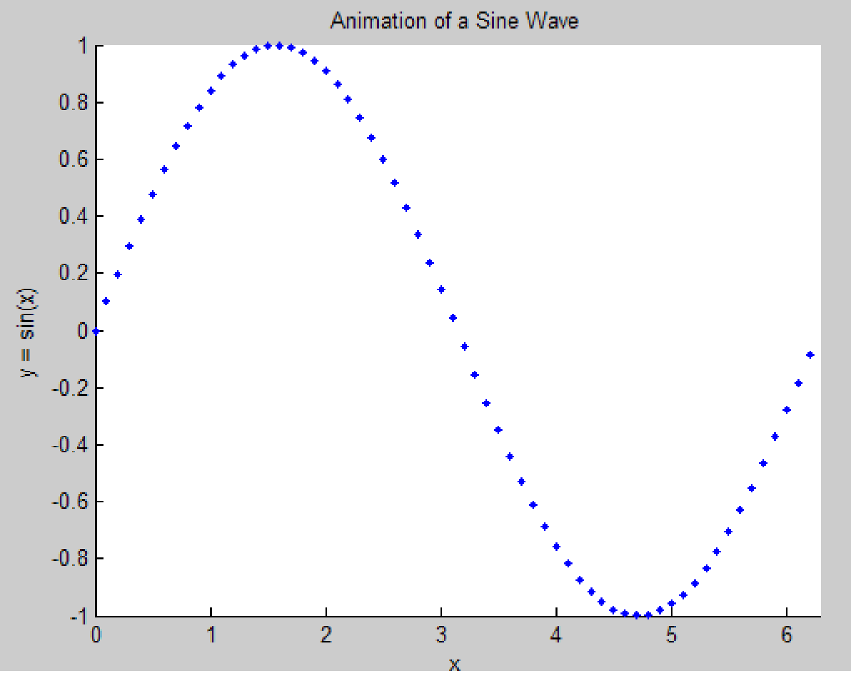
MATLABFunctions

Script Example

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

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

>> disp ‘Running Init Files ‘
>> run(‘My_MATLAB_Script’)

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

>> disp ’Init completed, open Simulink Model’
>> open(‘My_Simulink_Model.mdl’)
```

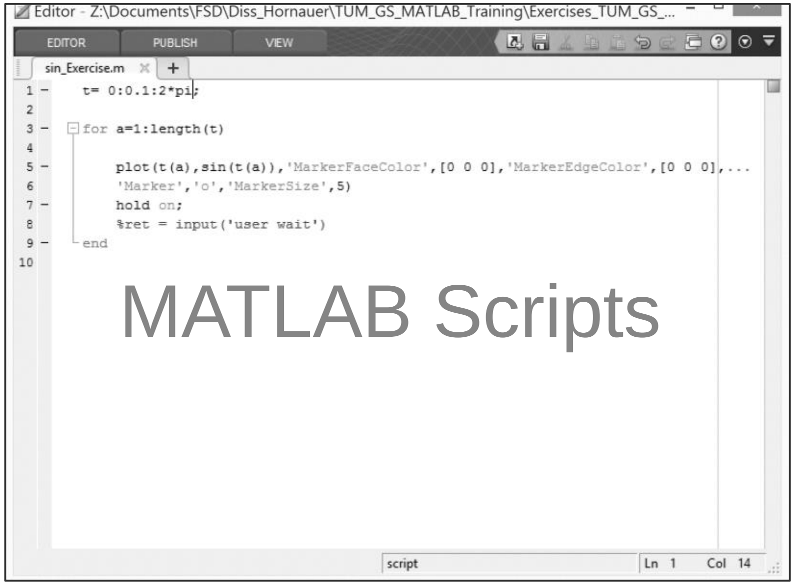
Exercise

- Plot a sine wave `y = sin(t), t=[0:0.1:2*pi]`
- Use `for` loop to create animation
- Save MATLAB script as `sine_wav_anim.m`



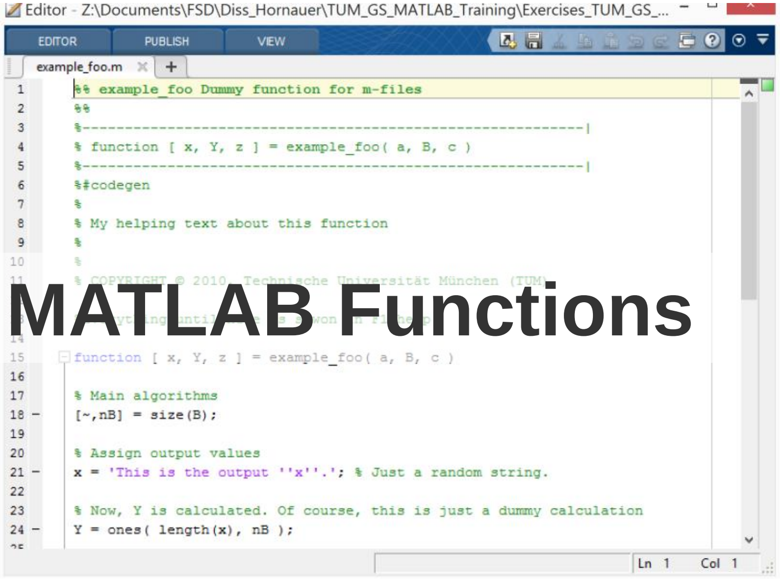
MATLAB Program Files

- Why?
  - Automating
  - Editing/Debugging
  - Deploying as applications


Editor - Z:\Documents\FSD\Diss_Hornauer\TUM_GS_MATLAB_Training\Exercises_TUM_GS_...
EDITOR
PUBLISH
VIEW
sin_Exercise.m
+
script
Ln 1
Col 14

```
   t= 0:0.1:2*pi;

for a=1:length(t)

    plot(t(a),sin(t(a)),'MarkerFaceColor',[0 0 0],'MarkerEdgeColor',[0 0 0],...
    'Marker','o','MarkerSize',5)
    hold on;
    %ret = input('user wait')
end
```

MATLAB Scripts


Editor - Z:\Documents\FSD\Diss_Hornauer\TUM_GS_MATLAB_Training\Exercises_TUM_GS_...
EDITOR
PUBLISH
VIEW
example_foo.m
+
Ln 1
Col 1

```
%% 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 file 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 );
```

MATLAB Functions

Basics of a MATLAB Program File

```
%% 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 shwon 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
```

Basics of a MATLAB Program File

```
 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 strucuring 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 ---
```

Angermann et al. p. 25

Functions

`function [out] = name(in)`	definition of MATLAB function `name` with list of input parameters `in` and output values `out`
`nargin, nargout`	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)

Angermann et al. p. 26

Global and static variables in functions

`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

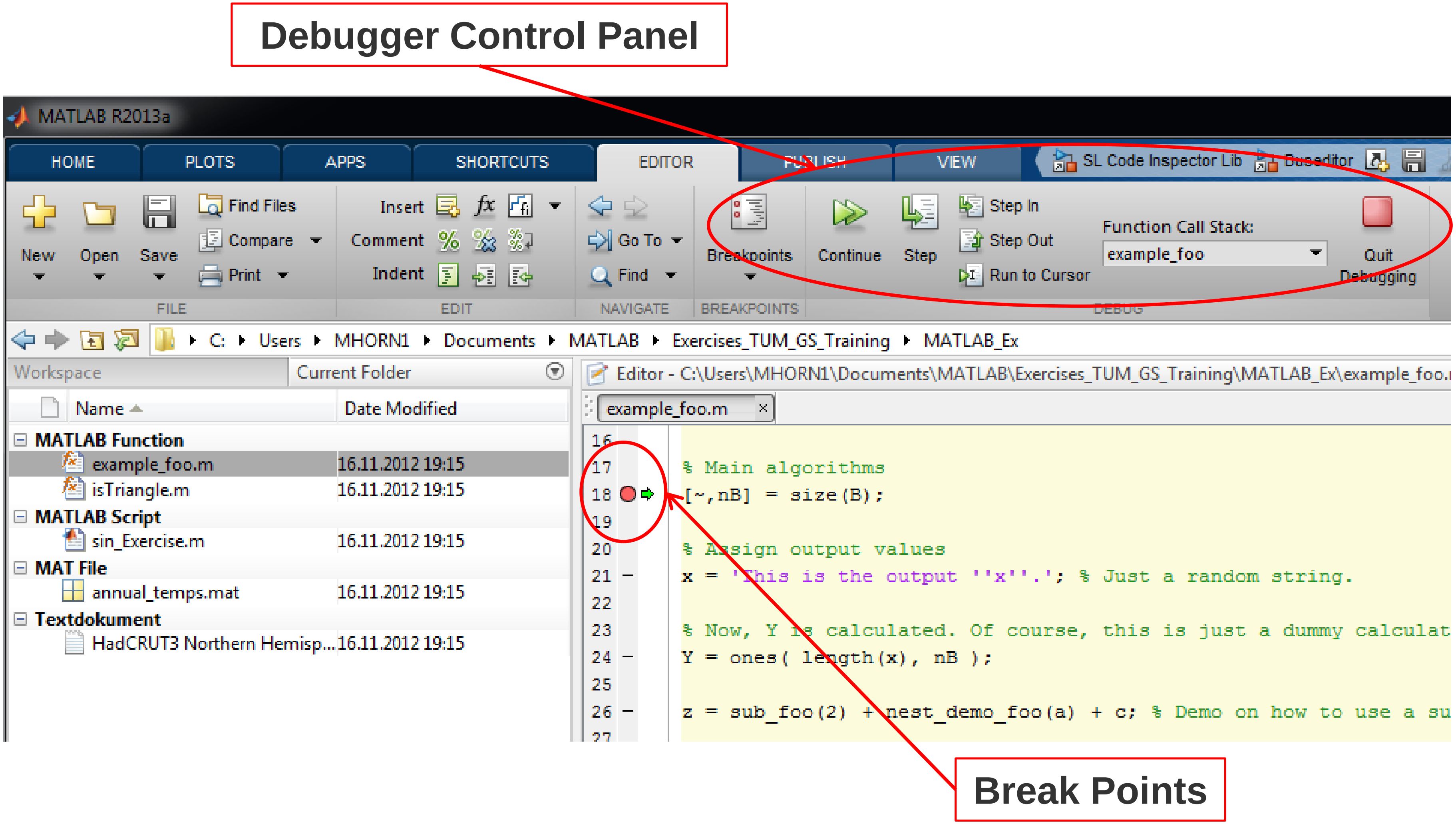
- File name has to be the same like the primary function in this file because MATALB is searching for files, not for functions

- Functions can call subfunctions within one file
- Subfunctions can call each other within one file
- Each function and subfunction **has it´s 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 al inner functions**

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

MATLAB Debugger



Debugger Control Panel

Break Points

Function Call Stack: example_foo

C: ▸ Users ▸ MHORN1 ▸ Documents ▸ MATLAB ▸ Exercises_TUM_GS_Training ▸ MATLAB_Ex

Editor - C:\Users\MHORN1\Documents\MATLAB\Exercises_TUM_GS_Training\MATLAB_Ex\example_foo.m

example_foo.m

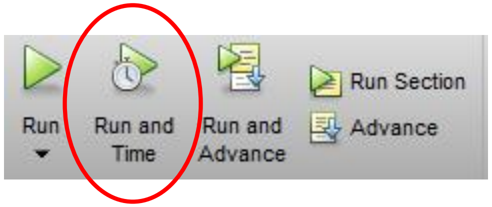
Name	Date Modified
MATLAB Function	
example_foo.m	16.11.2012 19:15
isTriangle.m	16.11.2012 19:15
MATLAB Script	
sin_Exercise.m	16.11.2012 19:15
MAT File	
annual_temps.mat	16.11.2012 19:15
Textdokument	
HadCRUT3 Northern Hemisp...	16.11.2012 19:15

```

% 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 calculat
Y = ones( length(x), nB );

z = sub_foo(2) + nest_demo_foo(a) + c; % Demo on how to use a su
```

Optimizing Performance

- `tic; code; toc;`

  determines code execution time

- 

  supports optimization of code

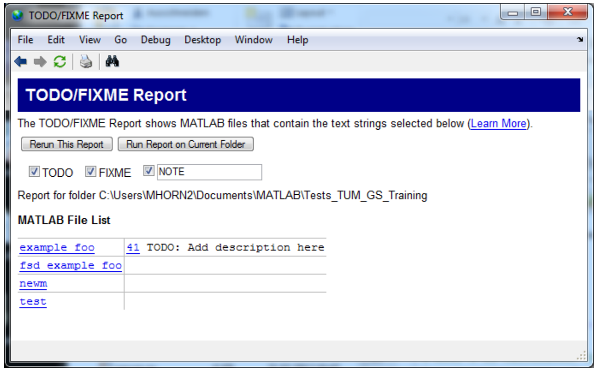
- Preallocation of memory

  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



TODO/FIXME Report

The TODO/FIXME Report shows MATLAB files that contain the text strings selected below (Learn More).

☑ TODO ☑ FIXME ☑ NOTE

Report for folder C:\Users\MHORN2\Documents\MATLAB\Tests_TUM_GS_Training

MATLAB File List

example_foo	41 TODO: Add description here
fsd_example_foo	
newm	
test	

Outline
Introduction to MATLAB

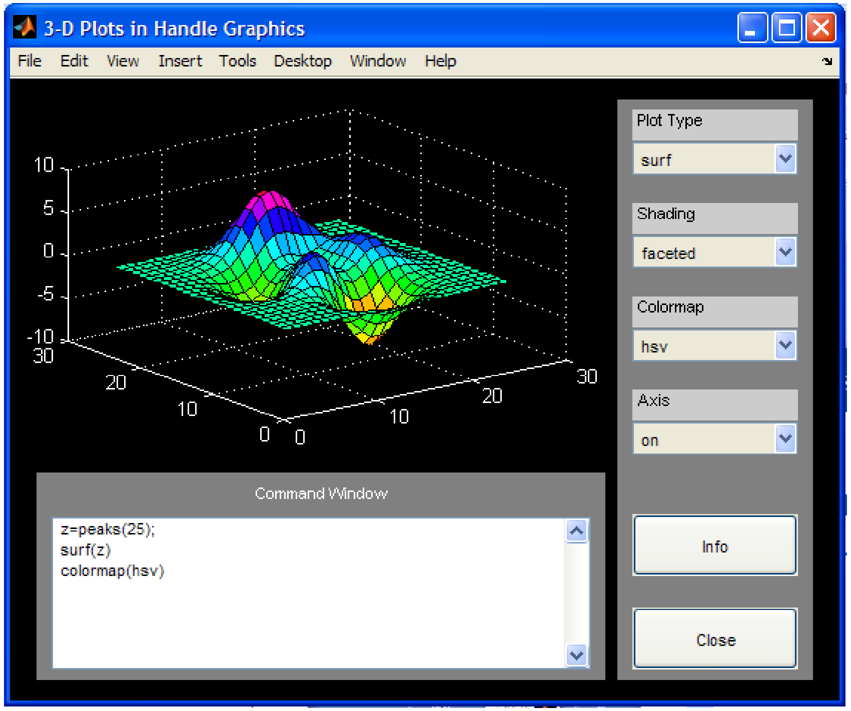
Basics:
1) Introduction
2) MATLAB Basics
3) 2D and 3D Plots
4) Data Import and Export

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



Plot Type: surf

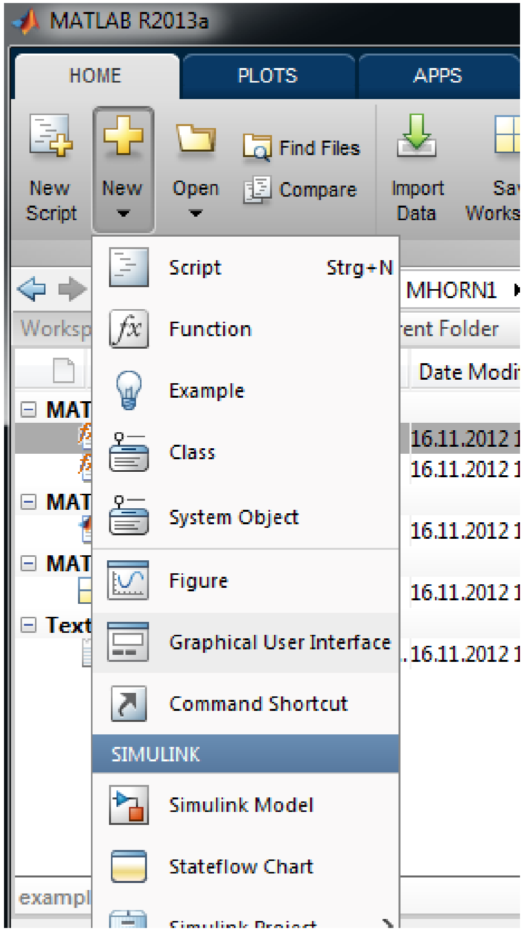
Shading: faceted

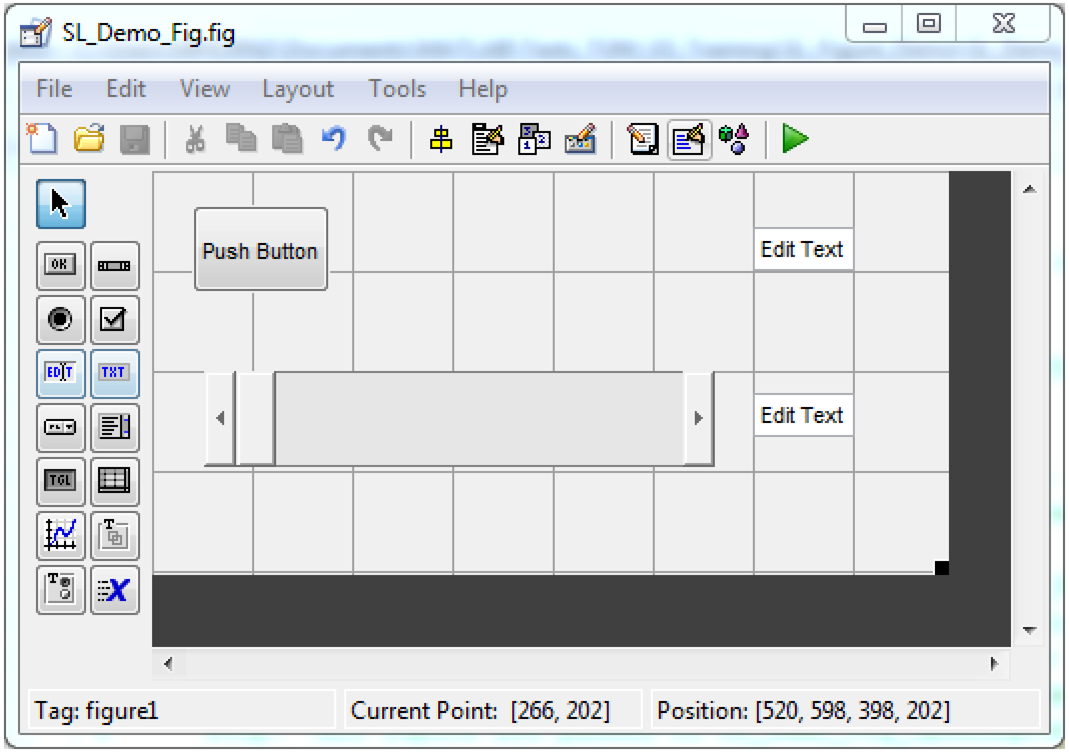
Colormap: hsv

Axis: on

```
z=peaks(25);
surf(z)
colormap(hsv)
```

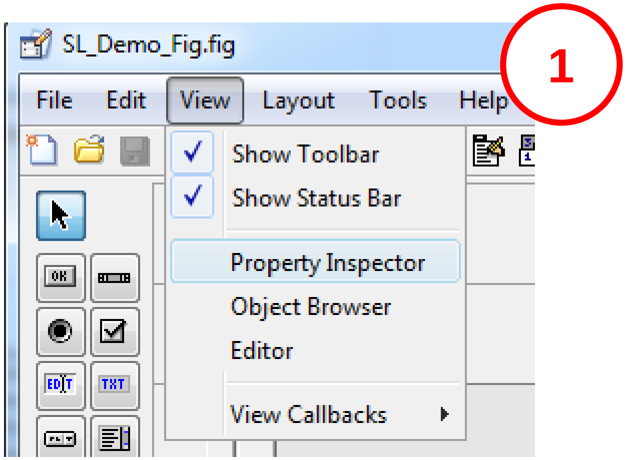
Using GUI Editor GUIDE

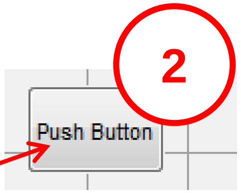




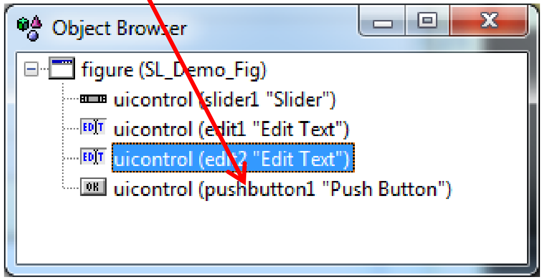
Tag: figure1 | Current Point: [266, 202] | Position: [520, 598, 398, 202]

Property Inspector

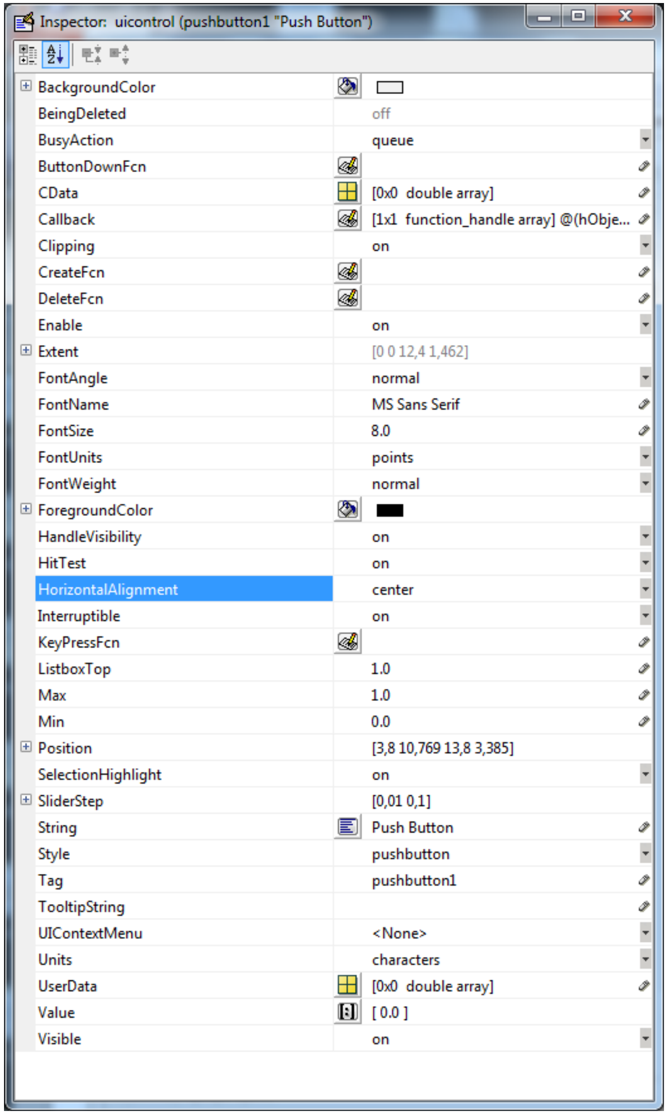




Double Click on Element

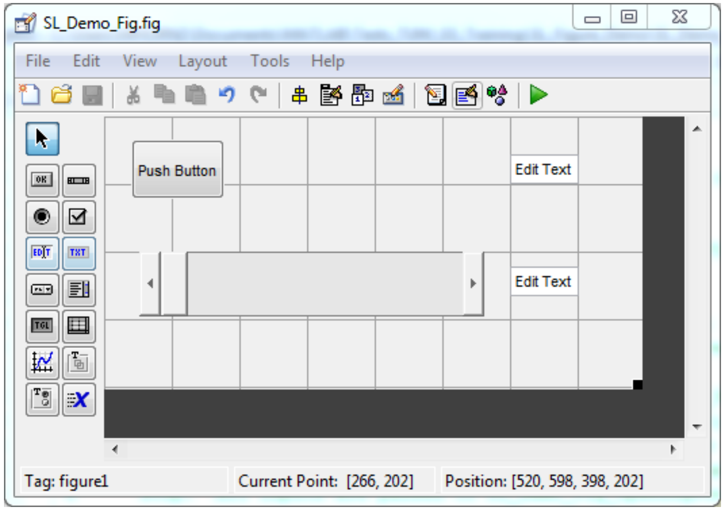


- figure (SL_Demo_Fig)
  - uicontrol (slider1 "Slider")
  - uicontrol (edit1 "Edit Text")
  - uicontrol (edit2 "Edit Text")
  - uicontrol (pushbutton1 "Push Button")

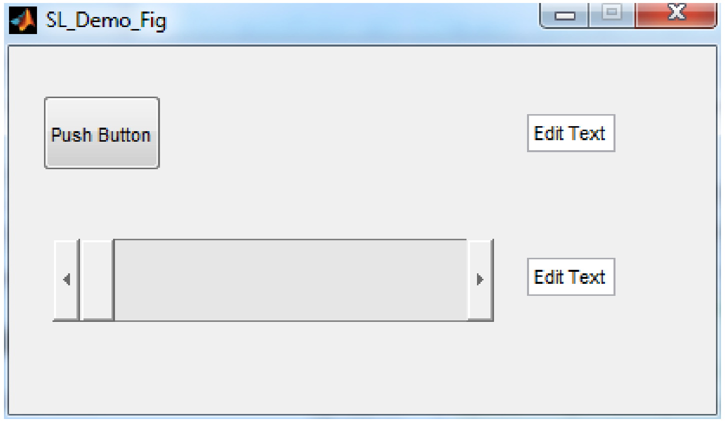


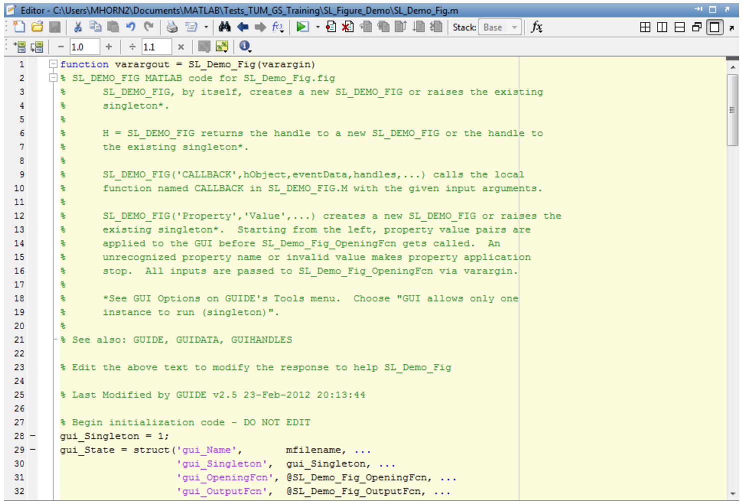
BackgroundColor	▭
BeingDeleted	off
BusyAction	queue
ButtonDownFcn	
CData	[0x0 double array]
Callback	[1x1 function_handle array] @(hObje...
Clipping	on
CreateFcn	
DeleteFcn	
Enable	on
Extent	[0 0 12,4 1,462]
FontAngle	normal
FontName	MS Sans Serif
FontSize	8.0
FontUnits	points
FontWeight	normal
ForegroundColor	■
HandleVisibility	on
HitTest	on
HorizontalAlignment	center
Interruptible	on
KeyPressFcn	
ListboxTop	1.0
Max	1.0
Min	0.0
Position	[3,8 10,769 13,8 3,385]
SelectionHighlight	on
SliderStep	[0,01 0,1]
String	Push Button
Style	pushbutton
Tag	pushbutton1
TooltipString	
UIContextMenu	<None>
Units	characters
UserData	[0x0 double array]
Value	[0.0]
Visible	on

Figures and Callback Functions



Tag: figure1 | Current Point: [266, 202] | Position: [520, 598, 398, 202]




Editor - C:\Users\MHORN2\Documents\MATLAB\Tests_TUM_GS_Training\SL_Figure_Demo\SL_Demo_Fig.m

```
function varargout = SL_Demo_Fig(varargin)
% SL_DEMO_FIG MATLAB code for SL_Demo_Fig.fig
%      SL_DEMO_FIG, by itself, creates a new SL_DEMO_FIG or raises the existing
%      singleton*.
%
%      H = SL_DEMO_FIG returns the handle to a new SL_DEMO_FIG or the handle to
%      the existing singleton*.
%
%      SL_DEMO_FIG('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in SL_DEMO_FIG.M with the given input arguments.
%
%      SL_DEMO_FIG('Property','Value',...) creates a new SL_DEMO_FIG or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before SL_Demo_Fig_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to SL_Demo_Fig_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help SL_Demo_Fig

% Last Modified by GUIDE v2.5 23-Feb-2012 20:13:44

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @SL_Demo_Fig_OpeningFcn, ...
                   'gui_OutputFcn',  @SL_Demo_Fig_OutputFcn, ...
```

Exchanging Data between GUI, MATLAB and Simulink

- Write data to MATLAB Workspace:

```
assignin('base',‘Name',Value);
```

- Read data from MATLAB Workspace:

```
evalin('base',‘Name');
```

- Use data within GUI (e.g. In Edit Box):

```
set(handles.edit,'String',’Value’);
get(hObject,'Value');
```

- Transmit Data to Simulink (e.g. Constant Block):

```
set_param(‘Simulink_Model/Constant','value',…
num2str(get(hObject,'Value')));
```

- Receive Data from Simulink (e.g. Constant Block):

```
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

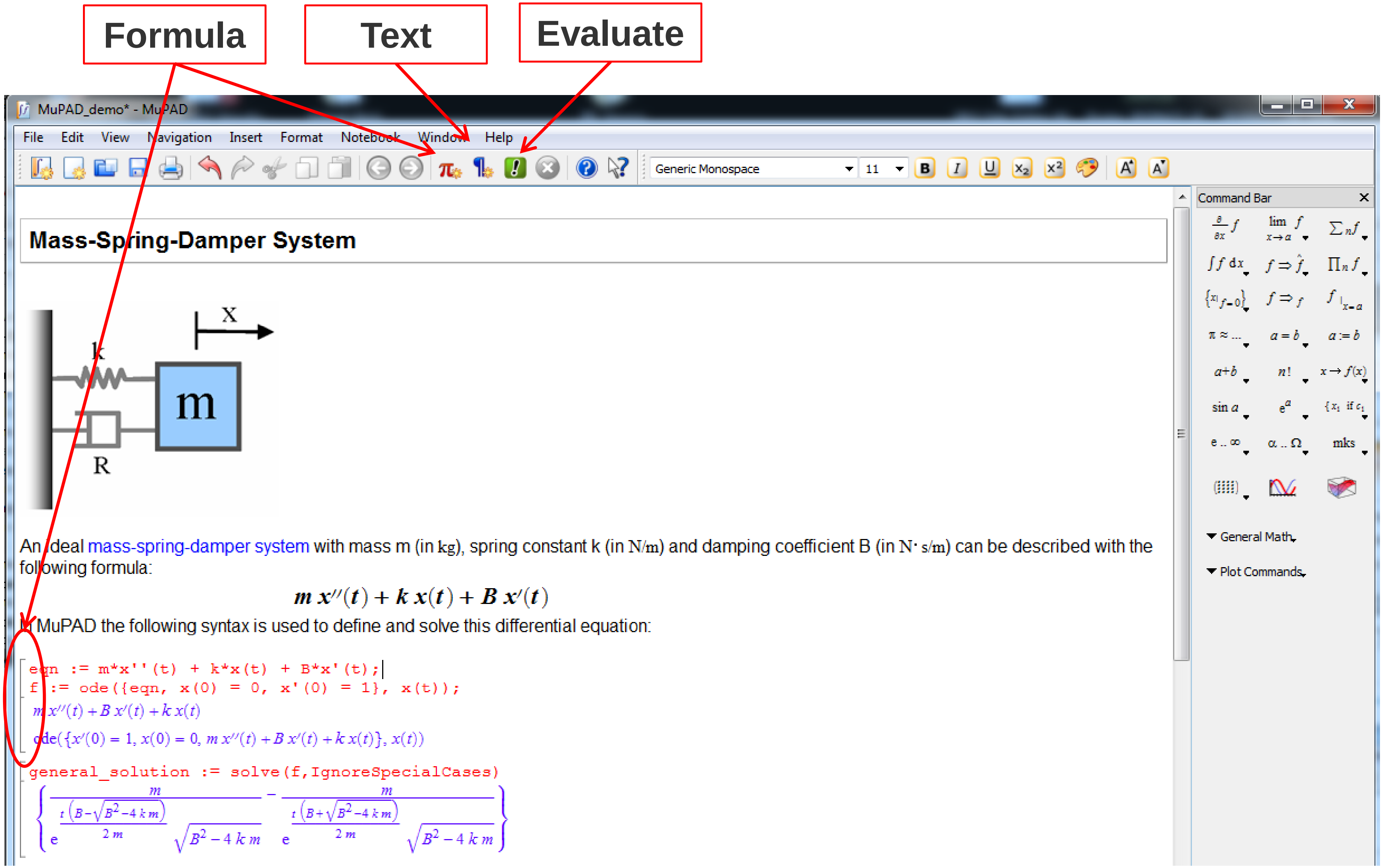
Advanced:
1) Programming with MATLAB
2) Graphical User Interfaces in MATLAB

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

Introduction to Symbolic Math

- 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

MuPAD Basics



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:

$$m\,x''(t)+k\,x(t)+B\,x'(t)$$

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

```
eqn := m*x''(t) + k*x(t) + B*x'(t);
f := ode({eqn, x(0) = 0, x'(0) = 1}, x(t));
```

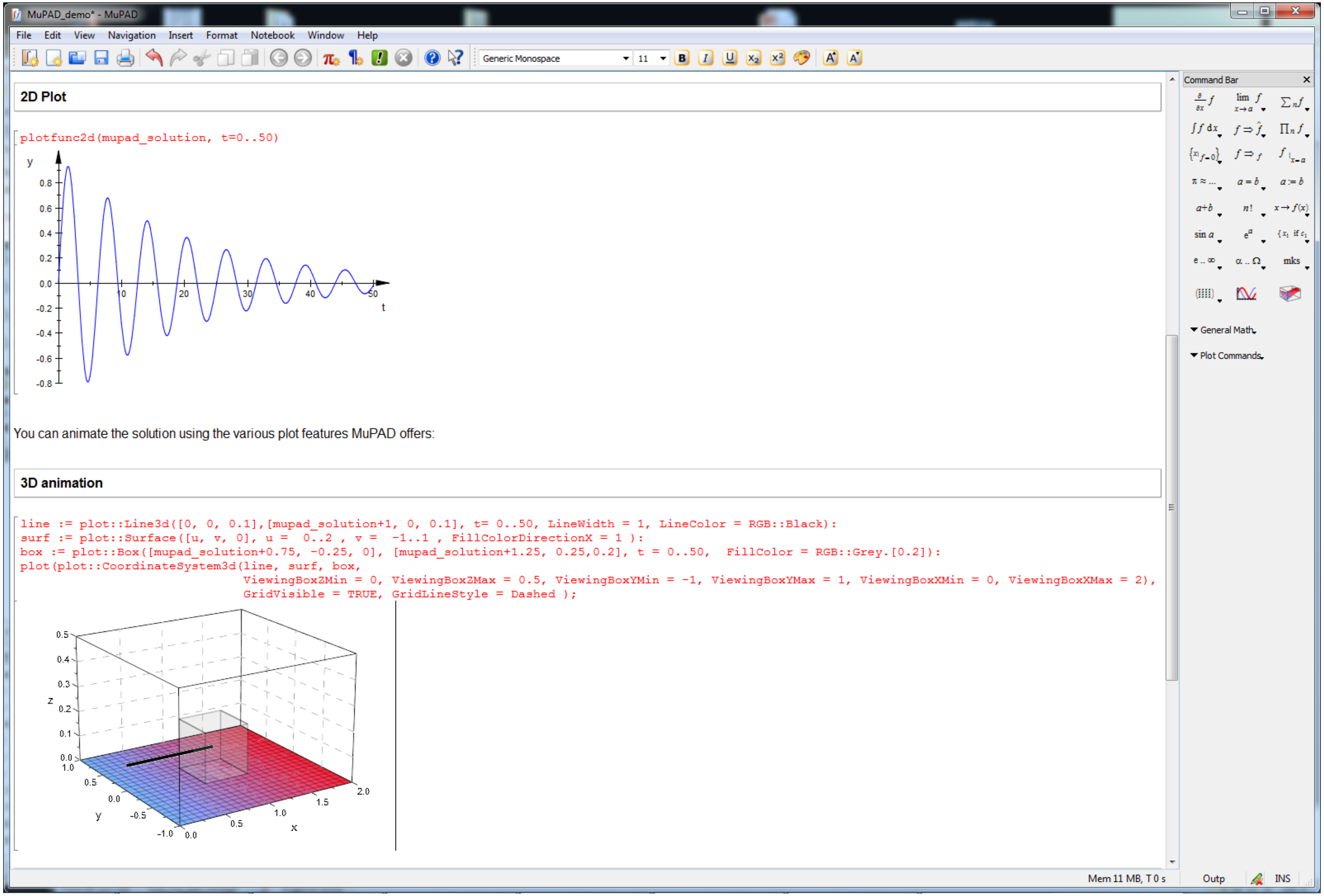
$$m\,x''(t)+B\,x'(t)+k\,x(t)$$

$$\mathrm{ode}(\{x'(0)=1,\ x(0)=0,\ m\,x''(t)+B\,x'(t)+k\,x(t)\},\ x(t))$$

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

$$\left\{\frac{m}{\mathrm{e}^{\frac{t\left(B-\sqrt{B^2-4\,k\,m}\right)}{2\,m}}\sqrt{B^2-4\,k\,m}}-\frac{m}{\mathrm{e}^{\frac{t\left(B+\sqrt{B^2-4\,k\,m}\right)}{2\,m}}\sqrt{B^2-4\,k\,m}}\right\}$$

MuPAD Plots



2D Plot

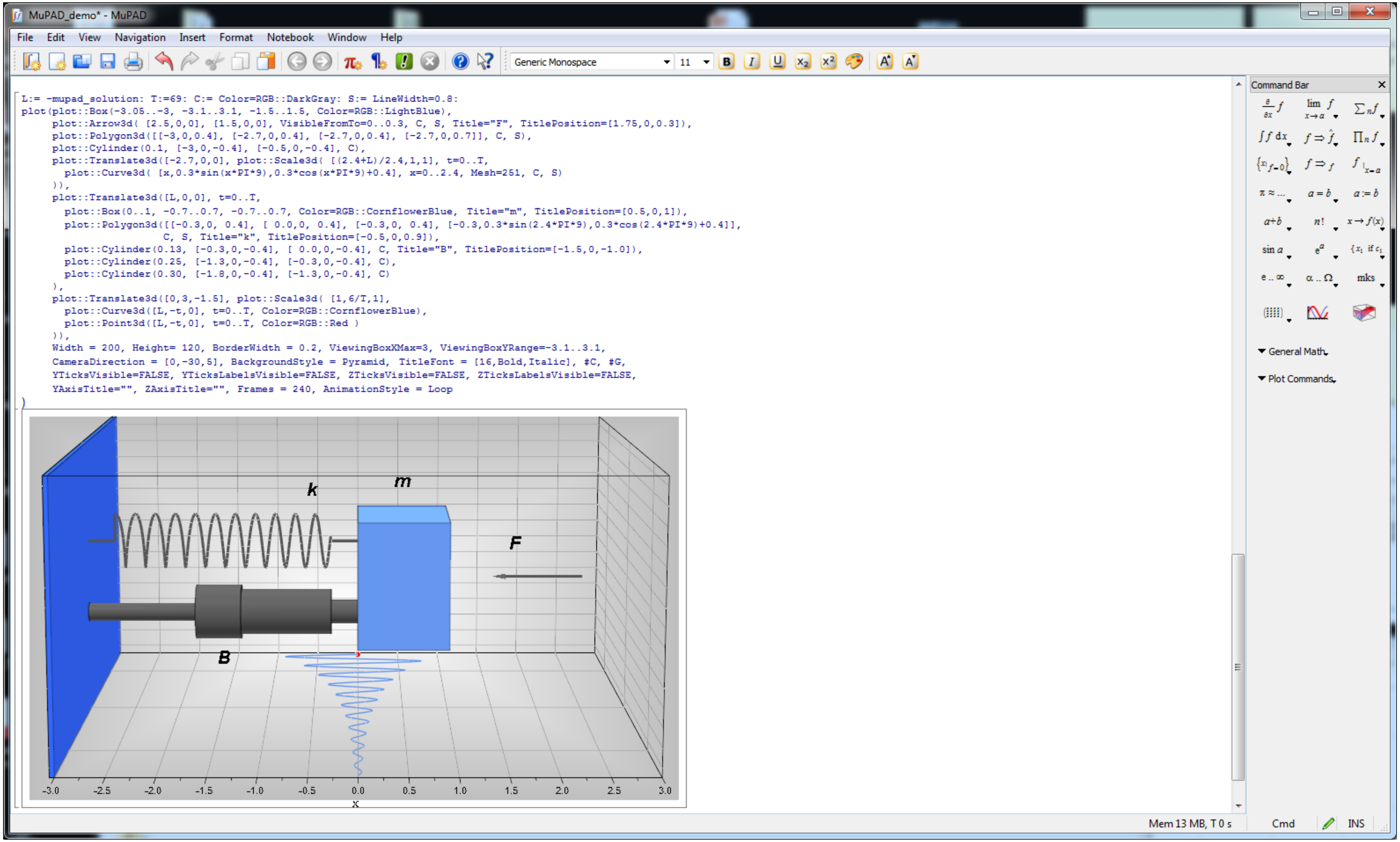
```
plotfunc2d(mupad_solution, t=0..50)
```

You can animate the solution using the various plot features MuPAD offers:

3D animation

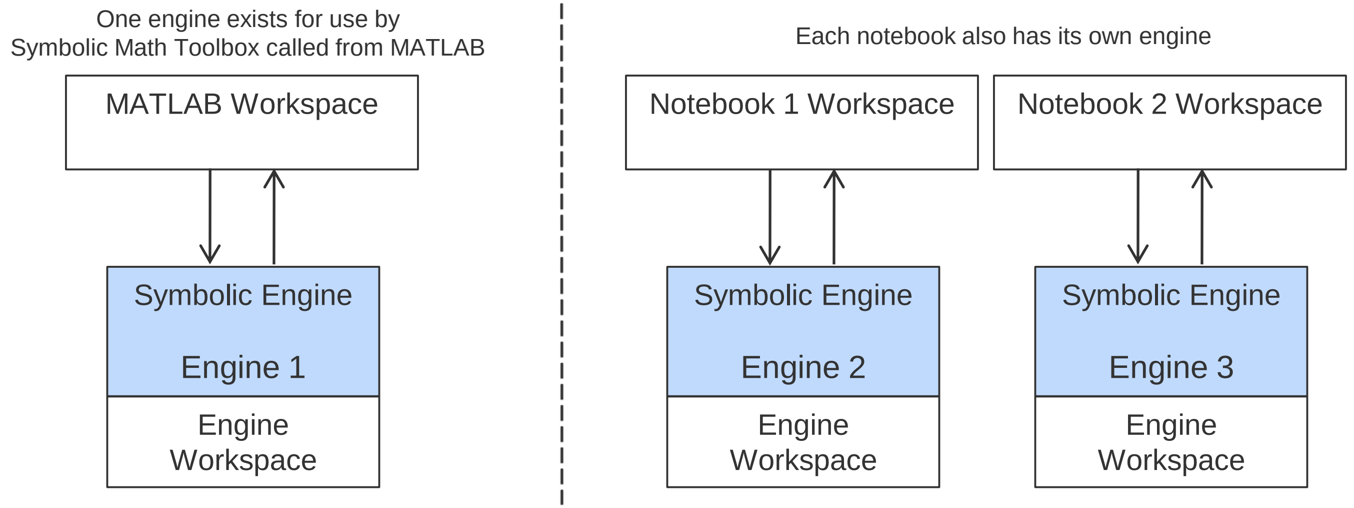
```
line := plot::Line3d([0, 0, 0.1],[mupad_solution+1, 0, 0.1], t= 0..50, LineWidth = 1, LineColor = RGB::Black):
surf := plot::Surface([u, v, 0], u =  0..2 , v =  -1..1 , FillColorDirectionX = 1 ):
box := plot::Box([mupad_solution+0.75, -0.25, 0], [mupad_solution+1.25, 0.25,0.2], t = 0..50,  FillColor = RGB::Grey.[0.2]):
plot(plot::CoordinateSystem3d(line, surf, box,
                              ViewingBoxZMin = 0, ViewingBoxZMax = 0.5, ViewingBoxYMin = -1, ViewingBoxYMax = 1, ViewingBoxXMin = 0, ViewingBoxXMax = 2),
                              GridVisible = TRUE, GridLineStyle = Dashed );
```

MuPAD 3D Animations



```
L:= -mupad_solution: T:=69: C:= Color=RGB::DarkGray: S:= LineWidth=0.8:
plot(plot::Box(-3.05..-3, -3.1..3.1, -1.5..1.5, Color=RGB::LightBlue),
     plot::Arrow3d( [2.5,0,0], [1.5,0,0], VisibleFromTo=0..0.3, C, S, Title="F", TitlePosition=[1.75,0,0.3]),
     plot::Polygon3d([[-3,0,0.4], [-2.7,0,0.4], [-2.7,0,0.4], [-2.7,0,0.7]], C, S),
     plot::Cylinder(0.1, [-3,0,-0.4], [-0.5,0,-0.4], C),
     plot::Translate3d([-2.7,0,0], plot::Scale3d( [(2.4+L)/2.4,1,1], t=0..T,
       plot::Curve3d( [x,0.3*sin(x*PI*9),0.3*cos(x*PI*9)+0.4], x=0..2.4, Mesh=251, C, S)
     )),
     plot::Translate3d([L,0,0], t=0..T,
       plot::Box(0..1, -0.7..0.7, -0.7..0.7, Color=RGB::CornflowerBlue, Title="m", TitlePosition=[0.5,0,1]),
       plot::Polygon3d([[-0.3,0, 0.4], [ 0.0,0, 0.4], [-0.3,0, 0.4], [-0.3,0.3*sin(2.4*PI*9),0.3*cos(2.4*PI*9)+0.4]],
               C, S, Title="k", TitlePosition=[-0.5,0,0.9]),
       plot::Cylinder(0.13, [-0.3,0,-0.4], [ 0.0,0,-0.4], C, Title="B", TitlePosition=[-1.5,0,-1.0]),
       plot::Cylinder(0.25, [-1.3,0,-0.4], [-0.3,0,-0.4], C),
       plot::Cylinder(0.30, [-1.8,0,-0.4], [-1.3,0,-0.4], C)
     ),
     plot::Translate3d([0,3,-1.5], plot::Scale3d( [1,6/T,1],
       plot::Curve3d([L,-t,0], t=0..T, Color=RGB::CornflowerBlue),
       plot::Point3d([L,-t,0], t=0..T, Color=RGB::Red )
     )),
     Width = 200, Height= 120, BorderWidth = 0.2, ViewingBoxXMax=3, ViewingBoxYRange=-3.1..3.1,
     CameraDirection = [0,-30,5], BackgroundStyle = Pyramid, TitleFont = [16,Bold,Italic], #C, #G,
     YTicksVisible=FALSE, YTicksLabelsVisible=FALSE, ZTicksVisible=FALSE, ZTicksLabelsVisible=FALSE,
     YAxisTitle="", ZAxisTitle="", Frames = 240, AnimationStyle = Loop
)
```

Workspaces in MuPAD and MATLAB

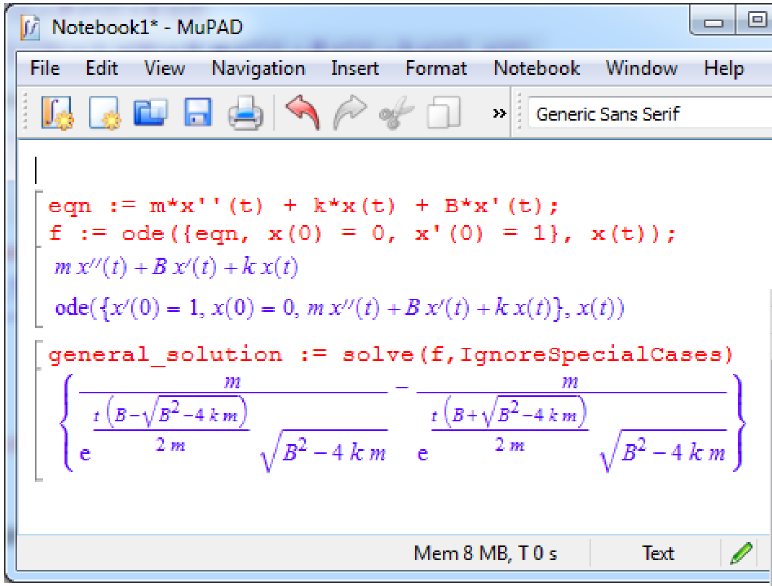


- MATLAB and symbolic engine have separate workspaces
- Each notebook also has a separate workspace

Export MuPAD Function to MATLAB

- Create handle to new Notebook: >> h = mupad;
- Get function from Notebook: >> y = getVar(h,’general_solution’);
- Convert symbolic expression to function handle or to file:

  >> f = matlabFunction(y);
  >> f = matlabFunction(y, ‘file’, ‘C:\myFctName’);



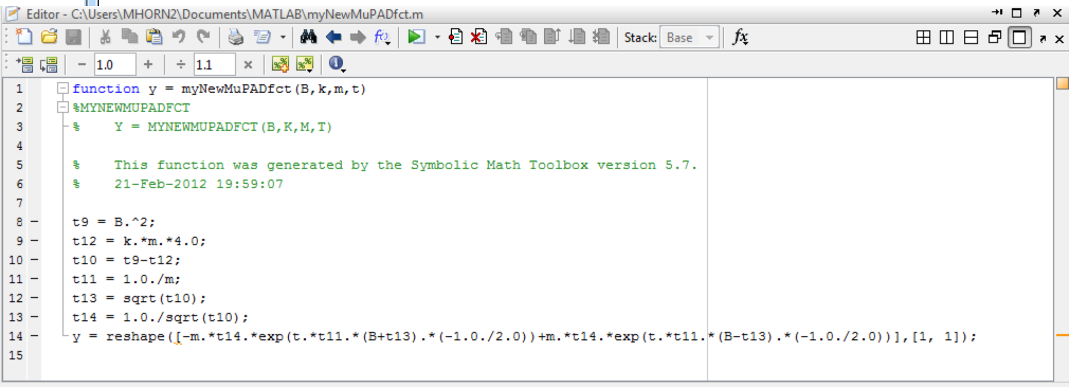
```
eqn := m*x''(t) + k*x(t) + B*x'(t);
f := ode({eqn, x(0) = 0, x'(0) = 1}, x(t));
```

$$m\,x''(t)+B\,x'(t)+k\,x(t)$$

$$\mathrm{ode}\left(\{x'(0)=1,\ x(0)=0,\ m\,x''(t)+B\,x'(t)+k\,x(t)\},\ x(t)\right)$$

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

$$\left\{\frac{m}{e^{\frac{t\,\left(B-\sqrt{B^2-4\,k\,m}\right)}{2\,m}}\,\sqrt{B^2-4\,k\,m}}-\frac{m}{e^{\frac{t\,\left(B+\sqrt{B^2-4\,k\,m}\right)}{2\,m}}\,\sqrt{B^2-4\,k\,m}}\right\}$$

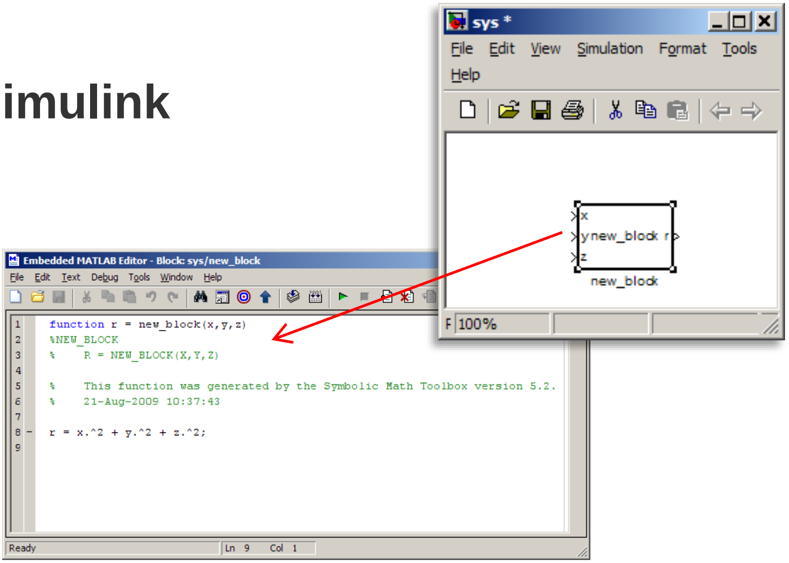


```
function y = myNewMuPADfct(B,k,m,t)
%MYNEWMUPADFCT
%    Y = MYNEWMUPADFCT(B,K,M,T)

%    This function was generated by the Symbolic Math Toolbox version 5.7.
%    21-Feb-2012 19:59:07

t9 = B.^2;
t12 = k.*m.*4.0;
t10 = t9-t12;
t11 = 1.0./m;
t13 = sqrt(t10);
t14 = 1.0./sqrt(t10);
y = reshape([-m.*t14.*exp(t.*t11.*(B+t13).*(-1.0./2.0))+m.*t14.*exp(t.*t11.*(B-t13).*(-1.0./2.0))],[1, 1]);
```

Using MuPAD in MATLAB and Simulink

- 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**

```
>> new_system('sys')
>> emlBlock('sys/new_block',y)
>> open_system('sys')
```


sys *
File Edit View Simulation Format Tools
Help
x
y new_block r
z
new_block
100%
Embedded MATLAB Editor - Block: sys/new_block
File Edit Text Debug Tools Window Help
Ready
Ln 9 Col 1

```
function r = new_block(x,y,z)
%NEW_BLOCK
%    R = NEW_BLOCK(X,Y,Z)

%    This function was generated by the Symbolic Math Toolbox version 5.2.
%    21-Aug-2009 10:37:43

r = x.^2 + y.^2 + z.^2;

```

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

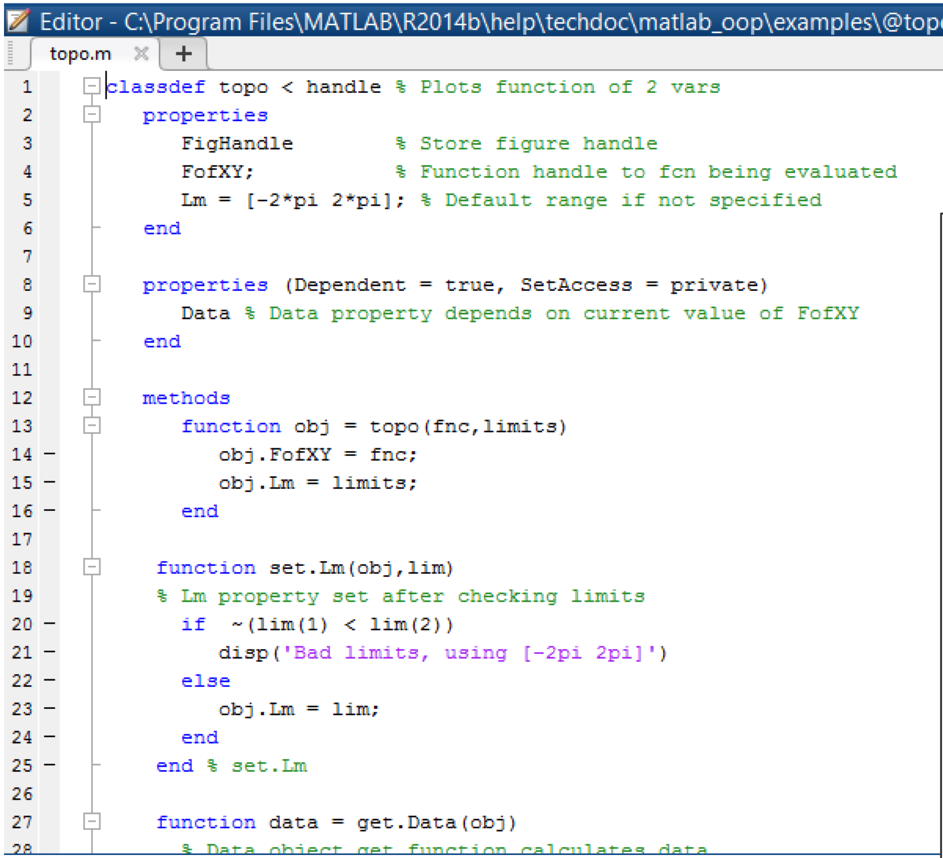
Advanced:

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

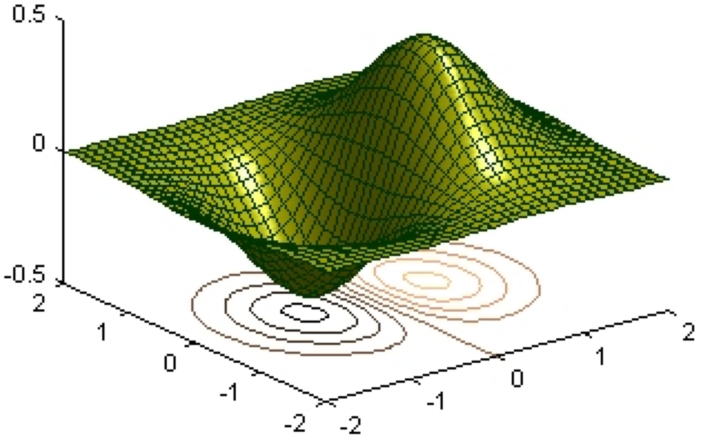


Editor - C:\Program Files\MATLAB\R2014b\help\techdoc\matlab_oop\examples\@topo\topo.m

topo.m

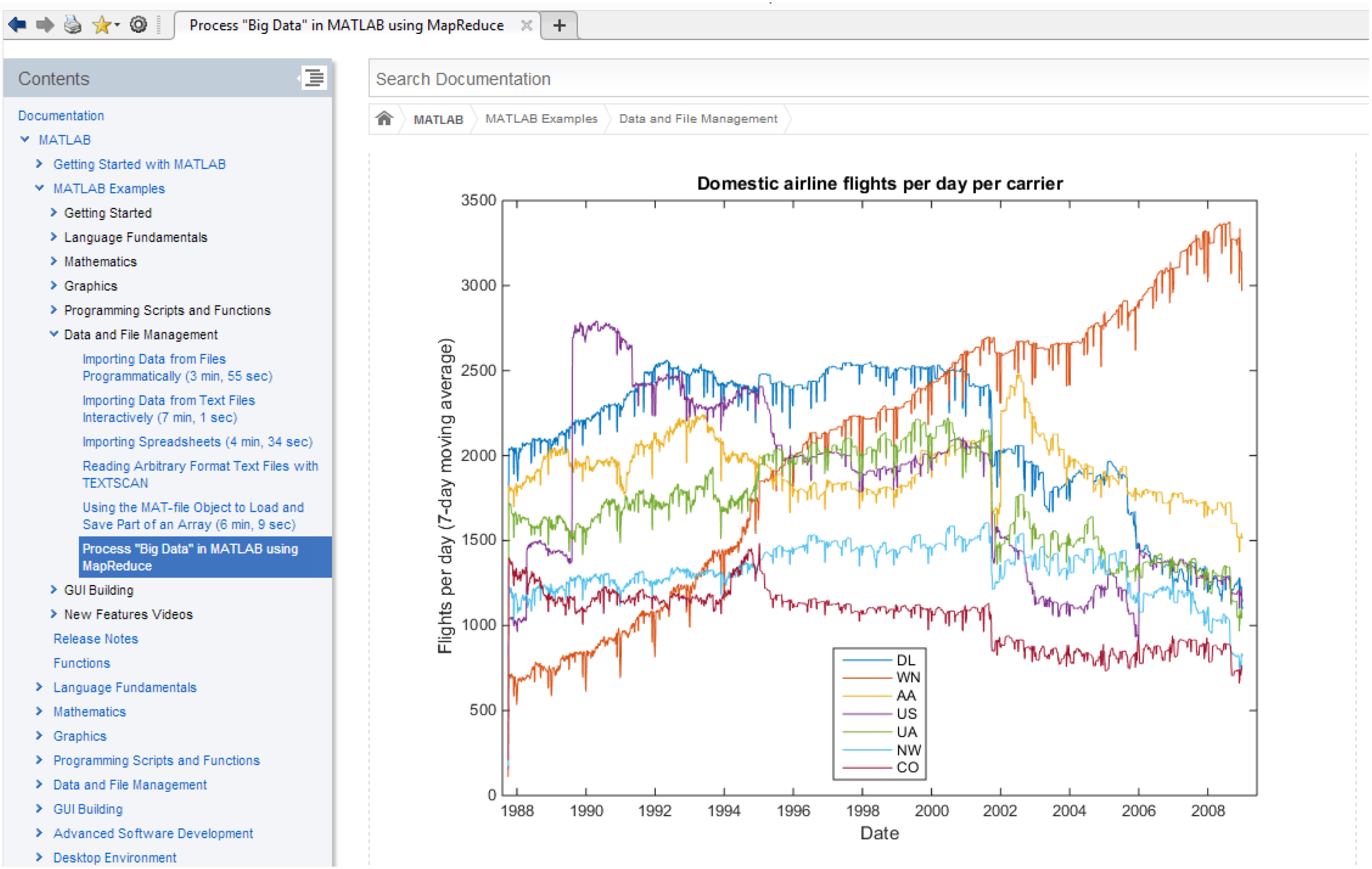
```
classdef topo < handle % Plots function of 2 vars
   properties
      FigHandle         % Store figure handle
      FofXY;            % Function handle to fcn being evaluated
      Lm = [-2*pi 2*pi]; % Default range if not specified
   end

   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
```

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



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

Big Data – MapReduce – Hadoop

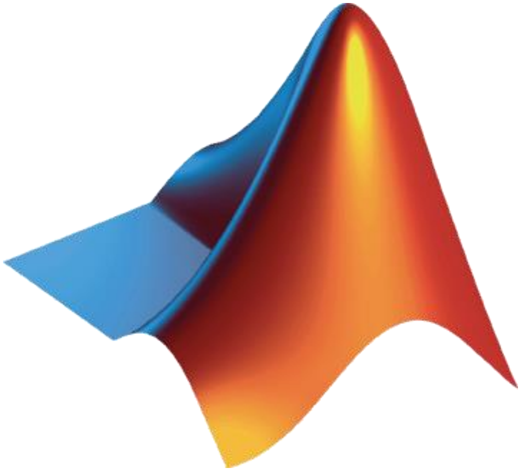

Process "Big Data" in MATLAB using MapReduce
+
Contents
Documentation
MATLAB
Getting Started with MATLAB
MATLAB Examples
Getting Started
Language Fundamentals
Mathematics
Graphics
Programming Scripts and Functions
Data and File Management
Importing Data from Files Programmatically (3 min, 55 sec)
Importing Data from Text Files Interactively (7 min, 1 sec)
Importing Spreadsheets (4 min, 34 sec)
Reading Arbitrary Format Text Files with TEXTSCAN
Using the MAT-file Object to Load and Save Part of an Array (6 min, 9 sec)
Process "Big Data" in MATLAB using MapReduce
GUI Building
New Features Videos
Release Notes
Functions
Language Fundamentals
Mathematics
Graphics
Programming Scripts and Functions
Data and File Management
GUI Building
Advanced Software Development
Desktop Environment
Search Documentation
MATLAB
MATLAB Examples
Data and File Management
Domestic airline flights per day per carrier
Flights per day (7-day moving average)
3500
3000
2500
2000
1500
1000
500
0
1988
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
Date
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

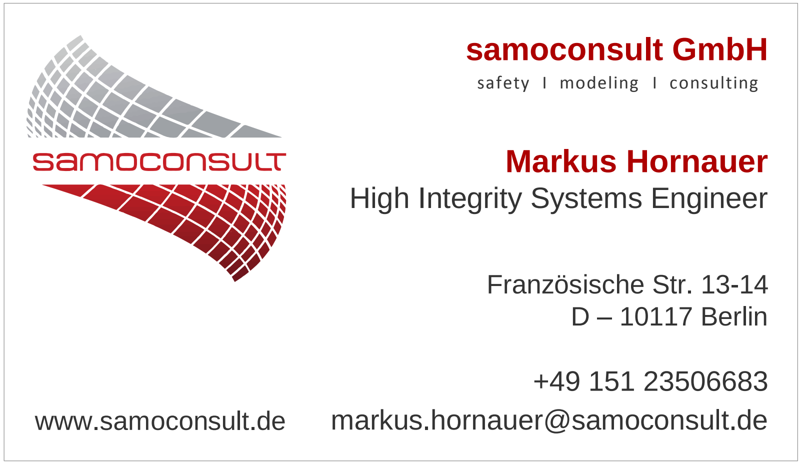
- MATLAB is a **high level-language** for **technical computing**
- Interactive tool with **mathematical and graphical** functions
- MATLAB provides features to **access, compute, analyze and visualize data**
- MATLAB also provides capabilities to **interface with external languages**



Contact

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



samoconsult GmbH
safety I modeling I consulting

Markus Hornauer
High Integrity Systems Engineer

Französische Str. 13-14
D – 10117 Berlin

+49 151 23506683
markus.hornauer@samoconsult.de

www.samoconsult.de

samoconsult