EL2310 – Scientific Programming

Lecture 1: Introduction

Ramviyas Parasuraman (ramviyas@kth.se)

KTH Royal Institute of Technology

Overview

Lecture 1, Part 0: Introduction to the Course

- Introduction
- Motivation and Goals
- Course Organization

Lecture 1, Part 1: Introduction to MATLAB

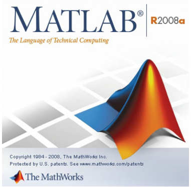
- About MATLAB
- Getting Started
- Basic Commands
- Vectors and Matrices

Welcome

- Lecturer 1: Ramviyas Parasuraman (ramviyas@kth.se)
- Lecturer 2: Hakan Karaoguz (hkarao@kth.se)
- Course overview
  - 17 Lectures (2 x 45 min. each)
  - 3 Lab sessions
  - Student presentations
  - 3 project assignments
- 7.5 credits
- Grade: Pass / Fail

Content

- Part I - MATLAB
- Part II - C
- Part III - C++


MATLAB®
R2008a
The Language of Technical Computing
Copyright 1984 - 2008, The MathWorks Inc.
Protected by U.S. patents. See www.mathworks.com/patents
The MathWorks

Content

- Part I - MATLAB
- Part II - C
- Part III - C++

MATLAB®
R2008a
The Language of Technical Computing
Copyright 1984 - 2008, The MathWorks Inc.
Protected by U.S. patents. See www.mathworks.com/pat
The MathWorks
SECOND EDITION
THE
C
ANSI C
PROGRAMMING LANGUAGE
BRIAN W. KERNIGHAN
DENNIS M. RITCHIE
PRENTICE HALL SOFTWARE SERIES

Content

- Part I - MATLAB
- Part II - C
- Part III - C++

What is your motivation and background?

- What programming languages have you heard of/used?
- What are likely usage scenarios for scientific programming in your future?

Harness the power!

- Your Smartphone now more power than a supercomputer a few decades ago.
- Fastest supercomputer as of June 2016 is the Sunway Taihu Light with 93.01 peta FLOPS.
- E.g. Sony PS4 has a peak performance of 1.84 tera FLOPS. 1 giga FLOP costs $0.2 today compared to $8.3 trillion in 1961 (inflation adjusted 2012 USD)
- see the WIKIPEDIA articles on Moore's law and FLOPS.

Motivations for the Course

- Programming is a key competence for todays engineers
- You may use programming as a **tool** in other courses.
- To investigate several tools for solving scientific/engineering problems
- The key question is to determine the appropriate tool in order to efficiently solve your task.

Structure of the Course

- Starts with MATLAB:
  - Scientific computing, Tailored for Master students
- Then we explore C programming.
- And finally we move on to Object Oriented Programming in C++.

Why MATLAB?

- MATLAB is a tool for interactive numerical computations
- Focus on rapid prototyping with complex computations
- Extensive code-base for:
  - control
  - signal processing
  - optimization
  - image processing
- Easy yo easily visualize and analyze data
- Used in many engineering companies, and extensively at KTH

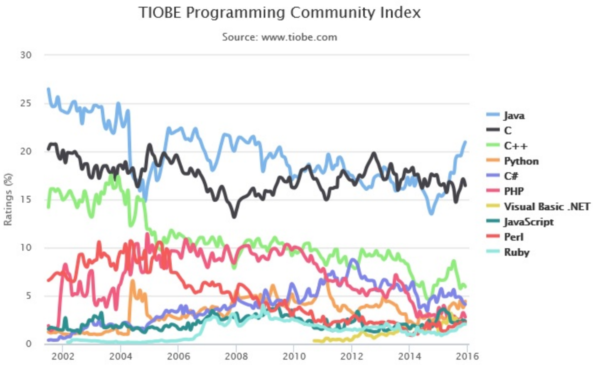
Why C?

- The most often used “low-level” language
- Allows “closer” interaction with hardware
- Used for systems programming: OS, embedded systems
- Examples: Linux Kernel, MATLAB
- Many languages borrow from C:
  C#, Go, Java, JavaScript, Perl, PHP
- Free compilers available for most architectures/hardware

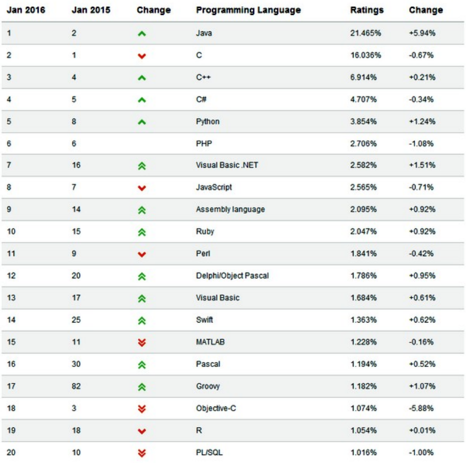
Why C++?

- Used extensively in industry and academia
- Intermediate-level programming language
- Many benefits of C with enhancements and new programming patterns
- Real-time applications mostly use C/C++
- A language of robotics (ROS, PCL)!
- Constantly developed and standardized: C++11
- Free compilers available for most architectures

Programming Language Popularity



Programming Language Popularity



Jan 2016	Jan 2015	Change	Programming Language	Ratings	Change
1	2	↑	Java	21.465%	+5.94%
2	1	↓	C	16.036%	-0.67%
3	4	↑	C++	6.914%	+0.21%
4	5	↑	C#	4.707%	-0.34%
5	8	↑	Python	3.854%	+1.24%
6	6		PHP	2.706%	-1.08%
7	16	⇈	Visual Basic .NET	2.582%	+1.51%
8	7	↓	JavaScript	2.565%	-0.71%
9	14	⇈	Assembly language	2.095%	+0.92%
10	15	⇈	Ruby	2.047%	+0.92%
11	9	↓	Perl	1.841%	-0.42%
12	20	⇈	Delphi/Object Pascal	1.786%	+0.95%
13	17	⇈	Visual Basic	1.684%	+0.61%
14	25	⇈	Swift	1.363%	+0.62%
15	11	⇊	MATLAB	1.228%	-0.16%
16	30	⇈	Pascal	1.194%	+0.52%
17	82	⇈	Groovy	1.182%	+1.07%
18	3	⇊	Objective-C	1.074%	-5.88%
19	18	↓	R	1.054%	+0.01%
20	10	⇊	PL/SQL	1.016%	-1.00%

MATLAB vs. C/C++

MATLAB:

- ► Interpreted (executed by interpreter program)
- + Fast developing time
- - Slow run-time in certain cases
- + Portable
- ► Better for scientific code

C/C++:

- ► Compiled (and executed directly by CPU)
- - Slower developing time
- + Possible to write fast programs
- = Standard libraries are portable
- ► Better for system programming

Goals for MATLAB part

- Have an understanding for basic concepts in programming
- Be able to read, process and display data in MATLAB
- Solve problems and implement algorithms in MATLAB
- Know how to use MATLAB in other courses

Goals for C/C++ parts

- Be able to read and process data in programs written in C and C++
- Solve problems and implement algorithms in C and C++
- Be able to read and understand existing code
- Understanding the importance of writing readable code
- Know which tools to use to solve various scientific problems

Course Organization

- 3 parts - one for each language, i.e. MATLAB, C and C++
- Lectures (homeworks)
- Presentations
- Projects
- Help sessions

Presentations

- Walk-through of simple real world problems
- Each student will have to take part in a presentation
- Goals:
  - Become familiar with the computing environment
  - Prepare for the projects
  - Encourage curiosity
- Co-operation is encouraged
- Ask questions anytime, not only during help sessions or lecture breaks

Projects

- Larger scientific problems to solve
- You will learn something more than just programming
- The projects should be solved individually
- Graded: pass/fail
- Project needs to be submitted before a deadline
- To pass the course, pass all three projects

Help Sessions

- One help session before each project deadline
- See schedule for dates
- Do you have laptops?
- Additional Q/A sessions during lecture breaks

Course Homepage

- `http://www.csc.kth.se/~ramviyas/el2310.html`

- General course information
- Schedule
- Slides from the lectures
- Course materials

Bilda

- Online learning tool `http://bilda.kth.se`
- News and announcements
- Assignment submission
- Questions (avoid using e-mail)
- Forums and discussions
- Feedback

Literature & Materials

- No course book in the normal sense
- Plenty of good information available online
  - Manuals / Guides / Tutorials
  - Discussion forums (StackOverflow)
  - Use a search engine
- Some will be listed on the course website
- Share valuable resources with each other on **Bilda.**

Focus on Self-studying

- The lectures and labs can show you the basics, but you need to learn to seek programming knowledge and study on your own
- MATLAB is available on "KTH-CD"
  - `http://progdist.ug.kth.se`
- Tools for C/C++ are available with all Linux distributions
  - See course website
- **Strongly** recommended that you use Linux.

Programming Environment

- Matlab has a built-in IDE (Integrated Development Environment)
- We will not use an IDE for C/C++
- For C/C++, the tools are *gcc* (compiler) and an editor (e.g. gedit/vim/emacs)
- An IDE "hides" things you should know!

System

- For C/C++ we cannot only Linux
- Free open-source OS (e.g. Ubuntu)
- Environments
  - Own system
  - Virtual Machine through `http://www.virtualbox.org/`
  - CSC Computers
- Your assignments will be checked in Virtual Machine

Registration

If you are registered you should be able to,

- Log in to Bilda `http://bilda.kth.se`
- Have access to the CSC computers.

If not let me know.

Value of Feedback

- The quality of the course depends on your feedback!
- Not only at the end of the course (evaluation), but during the course
- Use **Bilda** as mode of interaction **NOT** email
- This course cannot be tailored for everyone, since your backgrounds vary dramatically

End of Part 0

Acknowledgements

- The course has been developed and improved previously by several people, including Patric Jensfelt, Carl Henrik Ek, Kai Hübner, Andrzej Pronobis, Florian Pokorny and Yasemin Bekiroglu.
- The lectures on MATLAB are partially based on material from
  - Mikael Johansson, EE/KTH (course 2E1215)
  - Fredrik Gustavsson, Linköping (course TSRT04)

Part I - Introduction to MATLAB

- MATLAB background
- Basics
- Interactive calculations
- Matrices and vectors

MATLAB Background

- MATLAB = **MAT**rix **LAB**oratory
- Commercialized 1984 by Mathworks
- Heavily extended since then
- A standard tool today
- Array programming language: arrays are fundamental types
- Makes numerical computations easy

Alternatives

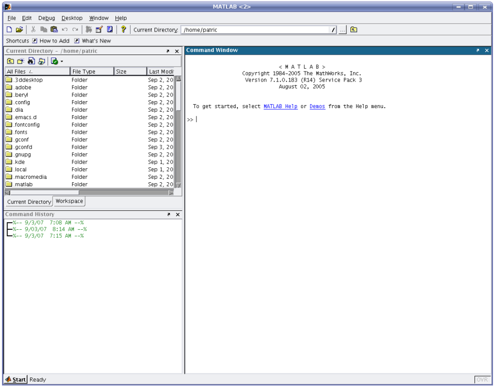
- There are alternatives such as
  - Octave (free and language mostly compatible with MATLAB)
  - Scilab
  - *NumPy/IPython - Numerical interactive computations in Python*
  - Matrix-X
- Additional Symbolic complements (using traditional mathematical notation)
  - Maple
  - Mathematica

Alternatives

- Matlab/C/C++ can be combined
- You can write highly optimized code in C/C++ and connect it to MATLAB using compiled MEX files.
- Python and other interpreted languages also allow you to do this.

Running MATLAB

- Available for Windows, Unix/Linux, Mac
- Great introductory video from MathWorks
- You can start with el2310-lab-matlab.pdf available in Bilda



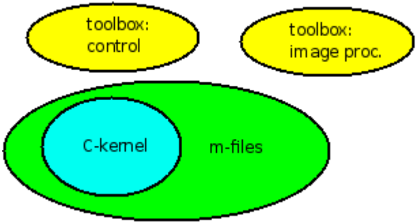
All Files	File Type	Size	Last Modi
.3ddesktop	Folder		Sep 2, 20
.adobe	Folder		Sep 2, 20
.beryl	Folder		Sep 2, 20
.config	Folder		Sep 2, 20
.dia	Folder		Sep 2, 20
.emacs.d	Folder		Sep 2, 20
.fontconfig	Folder		Sep 2, 20
.fonts	Folder		Sep 2, 20
.gconf	Folder		Sep 2, 20
.gconfd	Folder		Sep 3, 20
.gnupg	Folder		Sep 2, 20
.kde	Folder		Sep 1, 20
.local	Folder		Sep 1, 20
.macromedia	Folder		Sep 2, 20
.matlab	Folder		Sep 2, 20

```
%-- 9/3/07  7:08 AM --%
%-- 9/03/07  8:14 AM --%
%-- 9/3/07  7:15 AM --%
```

```
                          < M A T L A B >
              Copyright 1984-2005 The MathWorks, Inc.
               Version 7.1.0.183 (R14) Service Pack 3
                          August 02, 2005


  To get started, select MATLAB Help or Demos from the Help menu.

>> |
```

MATLAB Construction

- Core functionality based on compiled C-routines
- Most functionality given as .m-files
- Grouped into toolboxes
- .m-files
  - contain source code
  - can be copied and altered
  - are platform independent (same on PC, Unix/Linux, Mac)

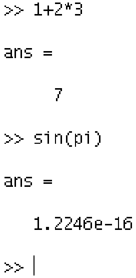


Command Window vs .m-files

- Code can be entered directly into the command window
  - Using MATLAB in an interactive fashion
- Code can also be stored in .m files
  - Write your program in an .m file
  - Whole program is executed using a single command

Interactive Calculations

- You do not need to declare variables in MATLAB
- It is interactive



```
>> 1+2*3

ans =

     7

>> sin(pi)

ans =

   1.2246e-16

>> 
```

Interactive Calculations

- Let’s have a look at the IDE

Documentation

- Help with syntax and function definitions

  ```
  >> help <function>
  ```

  Ex: “help sin”

- To look for a function with unknown name

  ```
  >> lookfor <keyword>
  ```

- Advanced hyperlinked help browser

  ```
  >> doc
  >> doc <function>
  ```

  Can also be accessed through the “Help” menu item

Variables

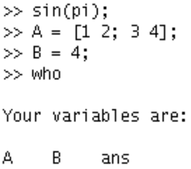
- Look at what variables are defined with

```
>> who
>> whos
```

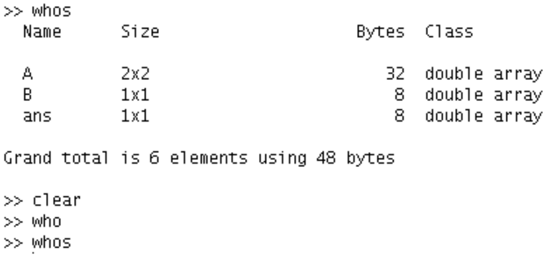
- Clear variables with

```
>> clear [variable(s)]
```

- Suppress output with ending “;” (semicolon)



```
>> sin(pi);
>> A = [1 2; 3 4];
>> B = 4;
>> who

Your variables are:

A    B    ans
```



```
>> whos
  Name      Size                    Bytes  Class

  A         2x2                        32  double array
  B         1x1                         8  double array
  ans       1x1                         8  double array

Grand total is 6 elements using 48 bytes

>> clear
>> who
>> whos
```

Loading and Saving Variables

- You can save all variables in memory with

```
>> save <filename>
```

- To save some variables do

```
>> save <filename> var1 var2 ...  varN
```

- To append variables do

```
>> save <filename> var1 var2 ...  varN -append
```

- You can load them back into memory with

```
>> load <filename>
```

Saving Command Window Text

- You can use the function `diary` to record what you are doing
- Allows you to go back and check what commands were issued
- Start the diary with

  `>> diary [filename]` or `>> diary('filename')`

  without the filename argument the diary file will be called “diary”
- To suspend/restart a diary, call:

  `>> diary on >> diary off`
- If you call `diary` without an argument you toggle diary on/off

Vectors

- Matrix and vector operations are at the very core of MATLAB
- For speed try to formulate a problem in terms of matrix operations
- Vector $v = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}$ is defined by

```
>> v=[1 2 3 4];
```

- Vector $w = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$ is defined by

```
>> w=[1; 2; 3; 4];
```

Vectors Cont'd

- Can create a vector with “colon-notation”

  ```
  >> v = start_value:step:end_value
  ```

- Ex: To create a vector with number 1 3 5 7 you do

  ```
  >> v = 1:2:7
  ```

- Notice that step can be negative to create for example 7 5 3 1

  ```
  >> v = 7:-2:1
  ```

Indexing Vectors

- To access a certain value in a vector do

```
>> v(i)
```

  where `i` is the index of the value

- Note: All indices start at 1 in MATLAB.

Matrices

- Matrices (2D arrays) are defined similarly
- Matrix $A = \begin{bmatrix} 1 & 2 & 3 \\ 3 & 5 & 6 \end{bmatrix}$ is defined by

```
>> A = [1 2 3; 3 5 6];
```

- Note: MATLAB is case sensitive

Dimensions

- You can check the size of a matrix with `>> size(A)` which will return the number of rows and columns
- You can ask specifically for the number of rows or columns
- To get number of rows

  ```
  >> size(A,1)
  ```

  and number of columns

  ```
  >> size(A,2)
  ```

Matrix Operations

- You can use all common operators with the matrices such as
  ```
  >> C = A + B;
  ```
  or
  ```
  >> C = A * B;
  ```
  assuming that the involved matrices have the right dimensions.
- Element-wise multiplication
  ```
  >> C = A .* B;
  ```
- You can mix scalars and matrices such as
  ```
  >> C = A + 2;
  ```
  in which case the scalar adapts to fit the situation.
- Even functions like sin and cos can be applied to matrices in which case they operate on each element.

Matrix Transpose

- To transpose a matrix do

```
>> B = A'
```

- Note that the transpose will conjugate complex entries
- To avoid this use

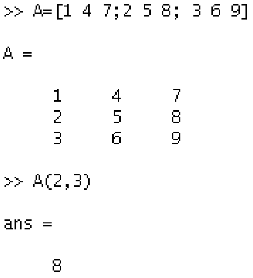
```
>> B = A.'
```

Indexing Matrices

- Index individual elements with

  ```
  >> A(i,j)
  ```

  where `i` is the row and `j` is the column

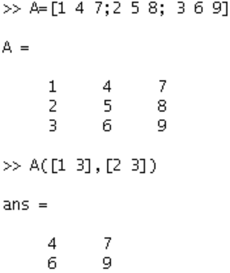


```
>> A=[1 4 7;2 5 8; 3 6 9]

A =

     1     4     7
     2     5     8
     3     6     9

>> A(2,3)

ans =

     8
```

Indexing Matrices Cont'd

- Index sub-matrices

```
>> A([1 3],[2 3])
```



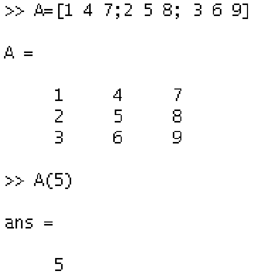
```
>> A=[1 4 7;2 5 8; 3 6 9]

A =

     1     4     7
     2     5     8
     3     6     9

>> A([1 3],[2 3])

ans =

     4     7
     6     9
```

Indexing Matrices Cont'd

- Sometimes convenient with single index notation
- Matrix elements ordered column by column

$$A = \begin{bmatrix} a_1 & a_4 & a_7 \\ a_2 & a_5 & a_8 \\ a_3 & a_6 & a_9 \end{bmatrix}$$

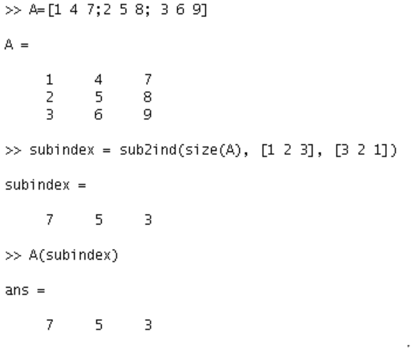
that is, $A(n) = a_n$ with the above ordering



```
>> A=[1 4 7;2 5 8; 3 6 9]

A =

     1     4     7
     2     5     8
     3     6     9

>> A(5)

ans =

     5
```

Indexing Matrices Cont'd

- Convert from subscripts (i, j) to linear indices
- Works for multiple (i, j) pairs stored in two arrays



```
>> A=[1 4 7;2 5 8; 3 6 9]

A =

     1     4     7
     2     5     8
     3     6     9

>> subindex = sub2ind(size(A), [1 2 3], [3 2 1])

subindex =

     7     5     3

>> A(subindex)

ans =

     7     5     3
```

Wrap Up

Today:

- Introduction to the Course
- Introduction to MATLAB

- Next time (Thurs 8-10, H32): Matlab as a Tool

Tasks for next time:

- Log into Bilda, check out course page
- Get and install MATLAB
  `http://progdist.ug.kth.se`
- Bring your laptop next time
- Take a look at the exercises

The First Presentation: PCA

- Explain what Principal Component Analysis (PCA) does, how it works and for what type of problems it is used.
- Implement it, compare your implementation with Matlab's built-in pca function on a dataset with different classes that has a large dimensionality. You can create your own data with multiple classes with random samples or use an already available dataset (from Matlab or another source).

The First Presentation: PCA

- Visualize the data in the new space and observe if data samples from the same classes are close to each other.
- How should we choose the number of eigen vectors to represent data without losing information?
- How can we implement a PCA-based face recognition method? (http://vision.ucsd.edu/content/yale-face-database)

The Second Presentation: Kmeans

- Explain what kmeans clustering algorithm does, how it works and for what type of problems it is used.
- Implement it and apply it on the IRIS dataset (load fisheriris)
- Compare your implementation with Matlab's built-in function. Do you get the same results?
- What are the factors that affect the performance of the algorithm?
- Apply your function to another dataset and evaluate the performance: e.g., kmeansdata.mat from Matlab