Algorithms, validation and visualisation

Ivo Roghair, Martin van Sint Annaland

Chemical Process Intensification Eindhoven University of Technology

Introduction

Introduction to programming

What is a program?

A program is a sequence of instructions that is written to perform a certain task on a computer.

- The computation might be something mathematical, such as solving a system of equations or finding the roots of a polynomial
- It can also be a symbolic computation, such as searching and replacing text in a document
- A program may even be used to compile another program
- · A program consists of one or more algorithms

Introduction

Today's outline

louay 3 outilli

- Introduction

- **V**isualisation
- 0 -----

Introduction

Algorithm design

- Problem analysis
 Contactual understa
- Contextual understanding of the nature of the problem to be solved
- Problem statement
- Develop a detailed statement of the mathematical problem to be solved with the program
- Processing scheme
 Define the inputs and outputs of the program
- Define the inputs and outputs of the program

 Algorithm
- A step-by-step procedure of all actions to be taken by the program (pseudo-code)

 Program the algorithm
- Convert the algorithm into a computer language, and debug until it runs
- Evaluation
 Test all of the options and conduct a validation study

About programming

Constructing a (series of) algorithm(s) that fulfill a certain function

- · Translate your problem to a formal procedure (recipe, pseudo-code)
 - What stens do I need to do?
 - Can you break down a step further?
 - Is the order of the steps of importance?
- · Can I re-use certain parts? · Translate your formal procedures to machine instructions
 - · Learning a programming language: syntax

Getting your hands dirty

- · Use an integrated development environment
 - Matlah
 - MS Visual Studio
 - Eclinse
 - Dov C++ IDLE, Canopy (express)
- Create a simple program:
- - Hello world
 - Find the roots of a parabola

Some often used programming languages

- Many functionalities available
- Smooth learning curve Slow compared to compiled
- Many freely available editors

dsheet (Excel_LibreOffice Calc_

- Steep learning curve
- · Compiled language, may be fast
- Some free compilers (fpc)

High availability · Low learning curve

- Very limited for larger problems unbeatable for quick calculations
- Not always free

Many functionalities built-in (80+

- Slow compared to compiled languages
- Fairly smooth learning curve · Needs a license, not available everywhe

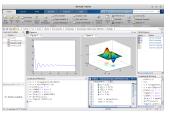
Many functionalities available

· Needs compilation, very fast (HPC) Freely available (gcc, MSVC)

Steeper learning curve

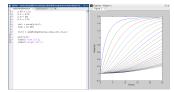
(alternatives: SciLab, GNU Octave)

Versatility of Matlab



Introduction

Versatility of Matlab: ODE solver



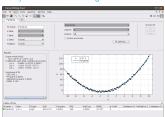
Introduction

Versatility of Matlab: Image analysis

```
I = inread('bubbles.png');
BW = rgb2gray(I);
E = edge(BW, 'canny');
F = infill(E, 'holes');
result = resionprops(F);
```



Versatility of Matlab: Curve fitting



Matlab resources

- Matlab documentation
- Introduction to Numerical Methods and Matlab Programming for Engineers. Todd Young and Martin J. Mohlenkamp (2014). GNU-licensed document, online
- Interactive Matlab Course. Pieter van Zutven (2010). See also http://www.imc.tue.nl/ (check website for the PDF)
- · Search the web!



Programming basics

Today's outline

- Introduction
- Programming basics
- Eliminating errors
- Visualisation
- Excel
- 6 Examples
- Conclusion

20,1

stroduction Programming basics

Programming basics

Programs consist of a number of expressions that form the algorithm.

- An expression is a command, combining functions, variables, operators and/or values to produce a result.
- · Variables contain one or more value(s)
- · Operators act on the data in variables (compare, add,
- multiply)
- Functions perform an operation on one or more variables and return one or more result(s).

The following will very shortly discuss some important aspects of variables, operators and functions that can be of use when creating your algorithms.

Programming basics

Syntax and semantics

Syntax (th

Correctness of the structure of symbols

x = 3 * 4: %ok

x + c = 5 car %wrong

Semantics (the meaning)

Opposed to natural language, programming languages are designed to prevent ambiguous, non-sensical statements

"Giraffes wait ravenously because the King of Scotland touched March"

14

Programming basics

Variables

- Data is stored in the memory of your computer, and can be read/updated using variables.
 - Matlab stores variables in the workspace
- A variable is not always the same as the mathematical concept of variable (i.e. part of an equation).
- You should recognize the difference between the identifier of a variable (e.g. x, setpoint_p), and the data that it actually stores (e.g. 0.5)
- Matlab also defines a number of variables by default, e.g. eps, pi or i.
- You can assign a variable by the = sign:
 x = 4*3
 - x =
- . If you don't assign a variable, it will be stored in ans

Datatypes and variables

Matlab uses different types of variables:

Datatype	Example
string	'Wednesday'
integer	15
float	0.15
vector	[0.0; 0.1; 0.2]
matrix	[0.0 0.1 0.2; 0.3 0.4 0.5]
struct	sct.name = 'MyDataName'
	sct.number = 13
logical	o (false)
	1 (true)

Building blocks: Mathematics and number manipulation

Programming languages usually support the use of various mathematical functions (sometimes via a specialized library). Some examples of the most elementary functions in Matlab:

Command	Explanation	
cos(x), sin(x), tan(x)	Cosine, sine or tangens of x	
mean(x), std(x)	Mean, st. deviation of vector x	
exp(x)	Value of the exponential function ex	
log10(x), log(x)	Base-10/Natural logarithm of x	
floor(x)	Largest integer smaller than x	
ceil(x)	Smallest integer that exceeds x	
abs(x)	Absolute value of x	
size(x)	Size of a vector x	
length(x)	Number of elements in a vector x	
rem(x,v)	Remainder of division of x by v	

About variables

· Matlab variables can change their type as the program proceeds (this is not common for other programming languages!):

```
>> s = 'This is a string'
This is a string
>> s = 10
    10
```

- · Vectors and matrices are essentially arrays of another data type. A vector of struct is therefore possible.
- · Variables are local to a function (more on this later).

Building blocks: conditional statements

if-statement: Performs a block of code if a certain condition is met

```
num = floor (10* rand +1) ;
guess = input ('Your guess please : ') ;
if ( guess ~= num )
   disp (['Wrong, it was ', num2str(num), '. Kbye. '
       1) ;
else
    disp ('Correct !') :
```

Other relational operators

	is equal to
<=	is less than or equal to
>=	is greater than or equal to
<	is less than
>	is greater than

Combining conditional statements



Building blocks: loops

for-loop: Performs a block of code a certain number of times.

```
>> p(1) = 1;

>> f(2) = 1;

>>
```

oduction Programming basics

Example algorithm

Compute the factorial of N: $N! = N \cdot (N-1) \cdot (N-2) \cdot \cdot \cdot 2 \cdot 1$

How to deal with this?



Note: N must be set beforehand!

Note: Pay attention to the relational operators!

Programming basic

Building blocks: indeterminate repetition

while-loop: Performs and repeats a block of code until a certain condition.

23

luction Programming basics

Building blocks: case selection

switch-statement: Selects and runs a block of code.

```
[dnum,dnam] = weekday(now);
switch dnum
switch dnum
disp("Yay! It is weekend!");
case 6
disp("Hooray! It is Friday!");
case (2,3,4,5)
disp(("Today is 'dnam]);
therwise
disp("Today is not a good day...');
and
```

Programming basics

Input and output

Many programs require some input to function correctly. A combination of the following is common:

- · Input may be given in a parameters file ("hard-coded")
- Input may be entered via the keyboard
 - >> a = input('Please enter the number ');
- . Input may be read from a file, e.g.
- There are many more advanced functions, e.g. fread, fgets, ...

25 / 65

introduction Programming basics

Functions - general

A function in a programming language is a program fragment that performs a certain task. Creating functions keeps your code clean, re-usable and structured

- You can use functions supplied by the programming language, and define functions yourself
- Functions take one or more input parameters (arguments), and return an output (result).
 - · If functions do not return a result, it is called a procedure
- In Matlab, functions are defined as follows (2 output variables and 3 input arguments):

```
function [out1, out2] = myFunction(in1, in2, in3)
```

Programming basic

Input and output

Output of results to screen, storing arrays to a file or exporting a graphic are the most common ways of getting data out of Matlab:

- Results of each expression are automatically shown on screen as long as the line is not ended with a semi-colon;
- · Output may be stored via the GUI:
 - · Use the 'Export Setup' function
 - Save figure (use .fig, .eps or .png, not .jpg or .pcx)
 Save variables (right click, save as)
- Save variables automatically (scripted):

```
>> savefile = 'test.mat';
>> p = rand(1,10);
>> q = ones(10);
>> save(savefile,'p','q')
```

· More advanced functions can be found in e.g. fwrite, fprintf,

20,

Functions - locality and arguments

- You are supplying arguments to a function because it does not have acces to previously defined variables. This is called locality.
 - · This does not include global variables but they're evil!
 - Local variables created in a function are not accessible to other functions unless they are returned or supplied as an argument!

Exercise: write a function that takes 3 variables, and returns the average:

```
Approach 1

function res = avg1(a,b,c)
  mySum = a + b + c;
  res = mySum / 3;
```

```
Approach 2

function res = avg2(a,b,c)
  data = [a; b; c];
  res = mean(data);
end
```

Exercise: create a function

Compute $N! = N \cdot (N-1) \cdot (N-2) \cdot \cdot \cdot 2 \cdot 1$

Create a function of our while-loop approach with N the argument:

```
Z = 1:
i = 1;
while (i<=N)
    Z = Z*i:
    i = i+1:
```

```
function Z = fact_while(N)
Z = 1:
i = 1:
while (i<=N)
   Z = Z*i;
    i = i+1;
end
```

ion Programming basics Eliminating errors Visualisation Excel Examples Co

Functions - checking input

The whole factorial function, including comments:

```
function Z = fact_while(N)
%% This function computes a factorial of input value N
% Usage : fact while(N)
         : value of which the factorial is computed
% returns: factorial of N
% Catch non-integer case
if (fix(N)~=N) | (N<O)
    disp 'Provide a positive integer number!'
    return;
Z = 1:
i = 1:
while (ic=N)
    Z = Z*i:
    i = i+1:
```

Functions - checking input

- The function we created computes the factorial correctly!
 - When the supplied argument is positive and
 - When the supplied argument is a natural number...



- . In this case, we should check the user input to prevent an infinite loop:
 - if (fix(N)~=N) | (N<O) disp 'Provide a positive integer number! return:
- end
- If no check can be done before a while-loop, you may want to stop after x loops

Recursion

- In order to understand recursion, one must first understand recursion
- A recursive function is called by itself (a function within a function)
 - . This could lead to infinite calle. · A base case is required so
 - that recursion is stopped: Base case does not call itself, simply returns.



- In order to understand recursion, one must first understand recursion
- A recursive function is called by itself (a function within a function)
 - This could lead to infinite calls:
 - A base case is required so
 - that recursion is stopped;
 Base case does not call itself, simply returns.



33.765

duction Programming basics

Recursion: exercise

end

Create a function computing the factorial of N, based on recursion.

```
function res = fact_recursive(x)

% Catch non-integer case
if (fix(x) > x) | (x < 0)
    disp 'You should provide a positive integer number
    return;
end

if (x > 1)
    res = x*fact_recursive(x-1);
else
    res = 1;
end
```

Recursion: example

- · What does this function do?
- · Can you spot the error?
- How deep can you go? Which values of b don't work anymore?

34

duction Programming basics Eliminating errors Vasualisation Excel Examples C

Today's outline

- Introduction
- Programming basic
- Eliminating errors
- (b) Excel
- Conclusion

ics Eliminating errors Visualisation Excel Examples Conclusions 000000000000 0 0000 0 0000 0 0

Today's outline

- Introduction
- Programming basics
- Eliminating errors
- Visualisation
- 6 Excel
- Examples
- Conclusions

Errors in computer programs

Computer programs often contain errors (bugs): buildings collapse, governments fall, kittens will die.

mring basics Eliminating errors Visualisation



. . ,

Errors in computer programs

The following symptoms can be distinguished:

- · Unable to execute the program
- Program crashes, warnings or error messages
- Never-ending loops
- Wrong (unexpected) result

Three error categories:

Syntax errors You did not obey the language rules. These errors prevent running or compilation of the program.

Runtime errors Something goes wrong during the execution of the program resulting in an error message (problem with input, division by zero, loading of non-existent files, memory problems, etc.)

Semantic errors The program does not do what you expect, but does what have told it to do.

ion Programming basics Eliminating erro

Verification and validation

Verification
Verification is the process of mathematically and computationally assuring that the model computes what you have entered.

Validation is the process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model



000000000

Be aware of your uncertainties





- The perceived orbit of Mars from Earth shows a zig-zag (in contrast to the Sun, Mercury, Venus)
- Even though they were not 'right', Earth-centered models (Ptolemy) were still valid

coccoccoccocc

A convenient tool: the debugger

- No-one can write a 1000-line code without making errors
 If you can, please come work for us
- One of the most important skills you will acquire is debugging.
- Although it can be frustrating, debugging is one of the most intellectually rich, challenging, and interesting parts of programming.
- In some ways, debugging is like detective work. You are confronted with clues, and you have to infer the processes and events that led to the results you see.

"When you have eliminated the impossible, whatever remains, however improbable, must be the truth."

A. Conan Doyle, The Sign of Four

Eliminati

Be aware of your uncertainties

Aleatory uncertainty

Uncertainty that arises due to inherent randomness of the system, features that are too complex to measure and take into account

Epistemic uncertaint

Uncertainty that arises due to lack of knowledge of the system, but could in principle be known

on Programming basics Eliminating errors

A convenient tool: the debugger

The debugger can help you to:

- · Pause a program at a certain line: set a breakpoint
- · Check the values of variables during the program
- · Controlled execution of the program:
 - One line at a time
 - Run until a certain line
 - Run until a certain condition is met (conditional breakpoint)
 Run until the current function exits
-
- Note: You may end up in the source code of Matlab functions!

About testcases (validation)

- Testcases: run the program with parameters such that a known result is (should be) produced.
- Testcases: what happens when unforeseen input is encountered?
 - More or fewer arguments than anticipated? (Matlab uses varargin and nargin to create a varying number of input arguments, and to check the number of given input arguments.
 - Other data types than anticipated? How does the program handle this? Warnings, error messages (crash), NaN or worse (a continuing program)?
- For physical modeling, we typically look for analytical solutions
 - · Sometimes somewhat stylized cases
 - · Possible solutions include Fourier-series
 - · Experimental data

45 / 6

Eliminating errors

If anything sticks today, let it be this

Your code will not be understood by anyone

That includes future-you

This can be prevented somewhat by the following

- Use comments! In Matlab, everything following % is a comment
- Prevent "smart constructions". You will spend a day tinkering why it does what it does...
- . If you write unmaintainable code, you'll have a job for life.
- Use comments! Documentation is also useful (though hard to maintain)

Advanced concepts

- Object oriented programming: classes and objects
- Memory management: some programming languages require you to allocate computer memory yourself (e.g. for arrays)
- External libraries: in many cases, someone already built the general functionality you are looking for
- Compiling and scripting ("interpreted"); compiling means converting a program to computer-language before execution.
 Interpreted languages do this on the fly.
- Profiling, optimization, parallellization: Checking where your program spends the most of its time, optimizing (or parallellizing) that part.

Todav's outline

- Introduction
- Programming basics
- Eliminating errors
- O Visualisation
- Excel
- Example
- Conclusion:

Today's outline

- Introduction
- Programming basics
- Eliminating error
- Visualisation
- Excel
- Examples
- Conclusion

Data visualisation

Modeling can lead to very large data sets, that require appropriate visualisation to convey your results.

- 1D 2D 3D visualisation
- Multiple variables at the same time (temperature, concentration, direction of flow)
- · Use of colors, contour lines
- · Use of stream lines or vector plots
- Animations

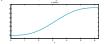


Plotting

```
x = -5:0.1:5;
y = x -2-44-x3;
y = x -2-44-x3;
subple (2,1.); plot(x,y,--1,x,y2,'r-');
subple (2,1.); plot(x,y,--1,x,y2,'r-');
subple (2,1.2); plot(x,abs(y-y2),'r-');
subple (2,1.2); plot(x,abs(y-y2),'r-');
subple (2,1.2); plot(x,abs(y-y2),'r-');
```

Plotting (2)

Easy plotting of functions can be done using the explot function: explot('x-sin(x)', [0 2*pi]):



Be careful with steep gradients: $explot(\ensuremath{\,^{^{\circ}}} x=sin(1/x)\ensuremath{\,^{^{\circ}}}, \ensuremath{\,^{\circ}} [0\ensuremath{\,^{\circ}}])$



Other plotting tools

- Errorbars: errorbar(x.v.err)
- 3D-plots: plot3(x,y,z)
- Histograms: histogram(x,20)







Multi-dimensional data

Matlab typically requires the definition of rectangular grid coordinates using meshgrid:

- [x y] = meshgrid(-2:0.1:2, -2:0.1:2):
- $z = x .* v .* exp(-x.^2 v.^2)$:
 - Surface plot
- Contour plot Waterfall
- Ribbons





```
v=-0.5:0.05:0.5:
contour(x,y,z,v,'ShowText'
, 'on');
waterfall(x,y,z);
colormap(winter):
ribbon(z):
```

surf(x.v.z):

Vector data

The gradient operator, as expected, is used to obtain the gradient of a scalar field. Colors can be used in the background to simultaneously plot field data:

```
[x \ y] = meshgrid(-2:0.2:2,
    -2:0.2:2);
z = x .* y .* exp(-x.^2 - y.^2)
[dx dv] = gradient(z.8.8)
% Background
contourf(x,y,z,30,'LineColor','
   none'):
colormap(hot); colorbar;
axis tight; hold on;
% Vectors
quiver (x,y,dx,dy,'k');
```



Today's outline

- 6 Excel

Solver and goal-seek

Excel comes with a goal-seek and solver function. For Excel 2010:

- Install via Excel ⇒ File ⇒ Options ⇒ Add-Ins ⇒ Go (at the bottom) ⇒ Select solver add-in. You can now call the solver screen on the 'data' menu ('Oplosser' in Dutch)
- Select the goal-cell, and whether you want to minimize, maximize or set a certain value
- Enter the variable cells; Excel is going to change the values in these cells to get to the desired solution
- Specify the boundary conditions (e.g. to keep certain cells above zero)
- · Click 'solve' (possibly after setting the advanced options).

Solver: a simple example

The solver is used to change the value in a goal-cell, by changing the values in 1 or more other cells while keeping boundary conditions:

. Use the following sheet:

	А	В	C
1		х	f(x)
2	×1	3	=2*B2*B3-B3+2
3	x2	4	=2*B3-4*B2-4

- Go to Data → Solver
 - Goalfunction: C2 (value of: 0)
 - Add boundary condition: C3 = 0
 - By changing cells: \$B\$2:\$B\$3 (you can just select the cells)
- . Solve. You will find B2=0 and B3=2.

Goal-seek: a simple example

Goal-Seek can be used to make the goal-cell to a specified value by changing another cell:

. Open Excel and type the following:

	А	В
1	х	3
2	f(x)	=-3*B1^2-5*B1+2
3		

- Go to Data ⇒ What-If Analysis ⇒ Goal Seek...
 - · Set cell: B2
 - To value: 0
- By changing cell: B1
- . OK. You find a solution of 0.333....

Exercise

Use Excel functions to obtain the Antoine coefficients A, B and C for carbon monoxide following the equation:

$$\ln P = A - \frac{B}{T + C}$$

P in Pa, T in K. Experimental data is given:

P [mmHg]	T [°C]
1	-222.0
5	-217.2
10	-215.0
20	-212.8
40	-210.0
60	-208.1
100	-205.7
200	-201.3
400	-196.3
760	-191.3

- Dedicate three separate cells for A, B and C.
 Give an initial guess
 - Onvert all values to proper units (hint: use
 - e.g. =CONVERT(A2, "mmHg", "Pa"))
 - Compute In P_{exp} and In P_{corr}
 Compute (In P_{exp} In P_{corr})², and sum this
 - olumn

 Start the solver, and minimize the sum by
 - changing cells for A, B and C.

0.765

0000 O

Today's outline

- Introduction
- Programming basics
- 6 Eliminating errors
- Visualisat
- A Excel
- 6 Examples

01/1

Example: finding the roots of a parabola

```
function x = parabola(a.b.c)
% Catch exception cases
if (a==0)
    1f(b==0)
        1f(c==0)
            disp('Solution indeterminate'): return:
        disp('There is no solution');
    end
    x = -c/b:
D = b^2 - 4*a*c:
1f (D<0)
    disp('Complex roots'); return;
    else if (D==0)
        x = -b/(2*a);
        else if (D>0)
                x(1) = (-b + sort(D))/(2*a):
                x(2) = (-b - sort(D))/(2*a);
                x = sort(x):
        end
    end
end
```

Example: finding the roots of a parabola

We are writing a program that finds for us the roots of a parabola.

We use the form

```
y = ax^2 + bx + c
```

What is our program in pseudo-code?

- Input data (a, b and c)
 Identify special cases (a = b = c = 0, a = 0)
 - a = b = c = 0 Solution indeterminate $a = 0 \text{ Solution: } x = -\frac{c}{2}$
- § Find $D = b^2 4ac$
- O Decide, based on D:
 - D < 0 Display message: complex roots
 - D = 0 Display 1 root value
 - D > 0 Display 2 root values

Example: finding the roots of a parabola

```
>> roots([1 -4 -3])
ans =
4.6458
-0.6458
```

Conclusion

In conclusion...

- Algorithm design: define your problem, think ahead, make a scheme, sketch the interplay between variables and functions, then start programming
- Programming basics: variables, operators and functions, locality of variables, recursive operations
- Dealing with complex programs, verification of your algorithms, use of the debugger
- Visualisation: how to make 1D and 2D/3D plots, create a sensible and intuitive presentation of your data.
- Examples: a few practice cases