Information Technology
Lecture 2: Introduction to programming

Roman Putanowicz

Chair for Computational Engineering

March 16, 2020

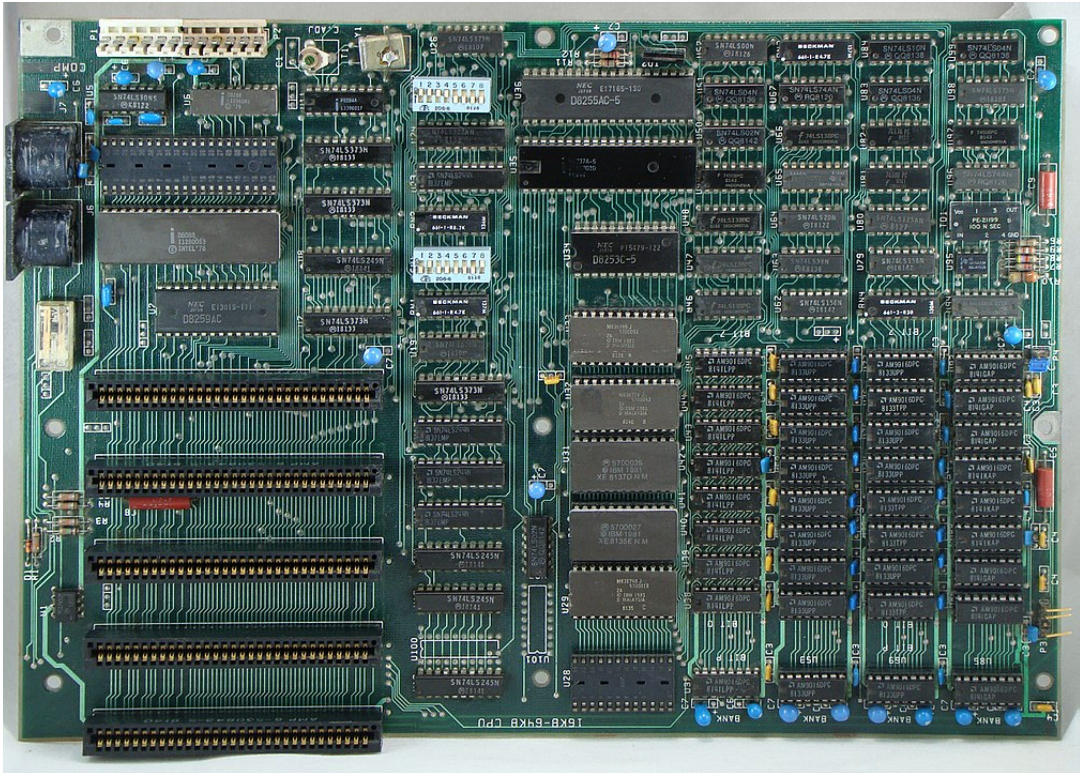
In brief

- How computer looks like?
- Processor and low level programming languages
- High level programming languages
- Python
- Octave
- Mathematical software environments

How computer looks like?

- microcomputer
- personal computers
- supercomputers

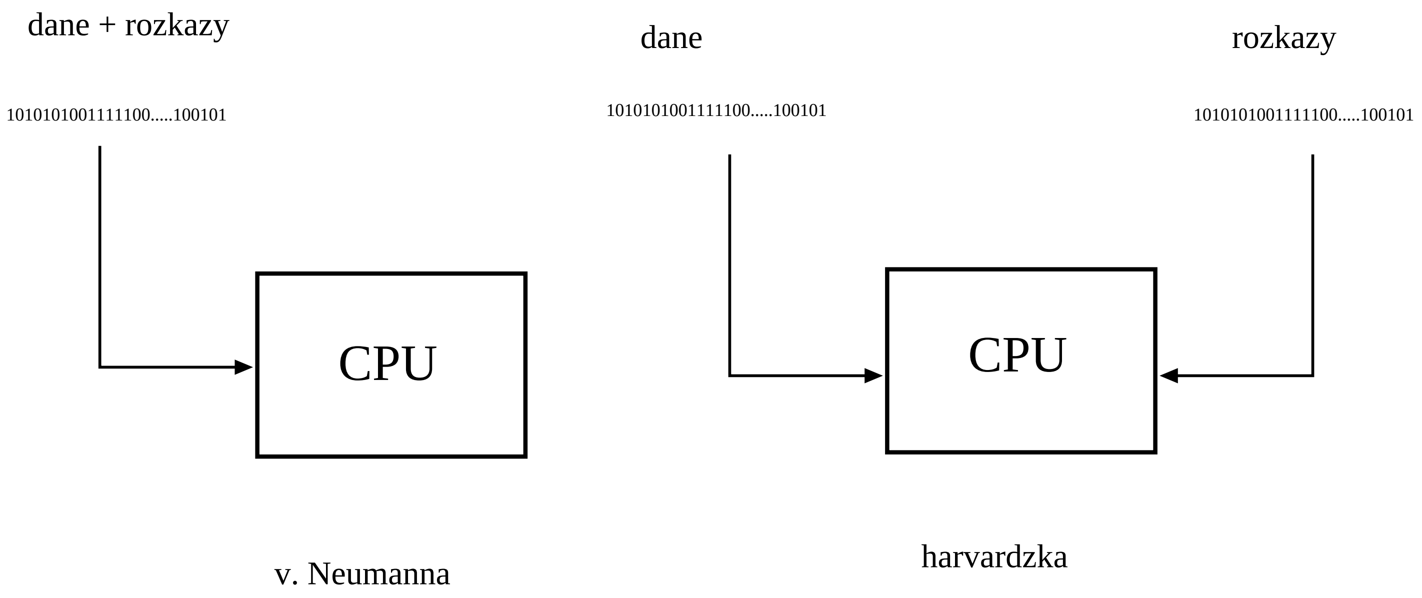
PC



Supercomputer Titan

By An employee of the Oak Ridge National Laboratory. -
http://www.olcf.ornl.gov/titan/, Public Domain,
https://commons.wikimedia.org/w/index.php?curid=2257572\1

Architecture



Processor and processor commands

$$1 + 2 =$$

Polish Notation: operator (command), arguments
ADD 1,2
`0010 0001 0010`

(Reverse Polish Notation - RPN)
Arguments, operator

Processor commands

- Each processor has its own unique list of command its capable of executing.
- Each command is encoded as a sequence of bits.
- For different processors the same command (e.g. ADD) can be encoded with different sequence of bits. Portability issues!
- Commands are indistinguishable from data!

bits	code
00000001	ADD
00000010	MUL
00000011	SUB
00000100	DIV
00001000	LOA
00000111	JMP
00001001	NOP

Program

Program (precisely: file with binary encoding of program) it is a sequence of bits containing both commands and data.

- von Neumann model – commands and data interleave, risk of mistake of malicious arrangements (viruses)
- Harvard model – two separate bit sequences - one for command the other for data, doubled data bus

Assembler

- it is possible to tediously write program directly in binary mode "by hand"
- instead of tedious binary encoding of commands it is customary to use mnemonic names, e.g. ADD, MUL, DIV, JMP
- such encoding is called Assembly language
- special program (assembler) translates mnemonic codes into binary image of a executable program.

High level programming languages

- assembler language depends on processor and is very tedious in writing (although can result in efficient code), it is not human oriented
- problems with assembler motivated creation of more convenient, human oriented and processor independent languages (high level languages)
- high level languages can be very rich in commands
- high level languages require special program (**compiler**) to translate the program to assembly language
- compiler takes care of hardware issues

High level programming languages

The family of high level programming languages is very big, containing very general as well as special purpose languages

- One of the fist language was Fortran (FORmula TRANslator), used widely in scientific and engineering software
- One of the most popular and widespread language is C
- The language traditionally used in teaching programming is Pascal

Classification of programming languages

There are multitude of programming languages and several ways of categorising them depending on their characteristics. Without going much into details we will distinguish the classifications:

- From the point of view of the complexity of instructions:
  - Low-level programming languages : assembly languages
  - High-level programming languages : C, C++, Java, etc.
- From the point of view of the usage patterns:
  - System programming languages: (C, C++, Fortran, Java, Ada) – associated with the tags like: efficiency safety, static type control.
  - Scripting languages: (Python, Ruby, Tcl, Guile, Ch) – associated with the tags like: rapid prototyping, flexibility, advanced introspection features

Programming languages for numerical simulations

Some languages are considered as better suited for writing numerical simulation codes. However picking the right language is a difficult thing often depending on non-technical issues (like available human resources in terms of programmers or local experts).
As most to the numerical algorithms utilize vector and matrix abstractions one important factor when evaluating a language is to what extend the language support direct use of these abstractions.
Support for vector and matrices can be either built-in into a language (Matlab, Octave, Fortran) or can be provided by a set of libraries.
Some popular choices are: Ada, C, C++, Fortran (both 77 and 90 and above), Matlab, Octave, Python, Ch.

Evolution within languages

- Fortran: 77, 95, 2008
- C++: 1998, C++03, C++11, C++14, C++17, C++20 (soon)
- Python: Python2, Python3 (since 2008)
- Matlab: 2008 - major enhancements for OOP

Programming paradigms

Programming:

- **Imperative** : C, C++, Java, Kotlin, PHP, Python, Ruby, Wolfram Language
- **Structural** : C, C++, Java, Kotlin, Pascal, PHP, Python, Wolfram Language
- **Procedural** : C, C++, Java, Kotlin, Pascal, PHP, Python, Wolfram Language
- **Functional** : C++, Clojure, Elixir, Erlang, F#, Haskell, Java (od wersji 8), Kotlin, Lisp, Python, JavaScript
- **Even based** : JavaScript, ActionScript, Visual Basic, Elm
- **Object Oriented** : Common Lisp, C++, C#, Eiffel, Java, Kotlin, PHP, Python, Ruby, Scala, JavaScript
- **Declarative** : SQL, regular expressions, Prolog, OWL, SPARQL, Prolog

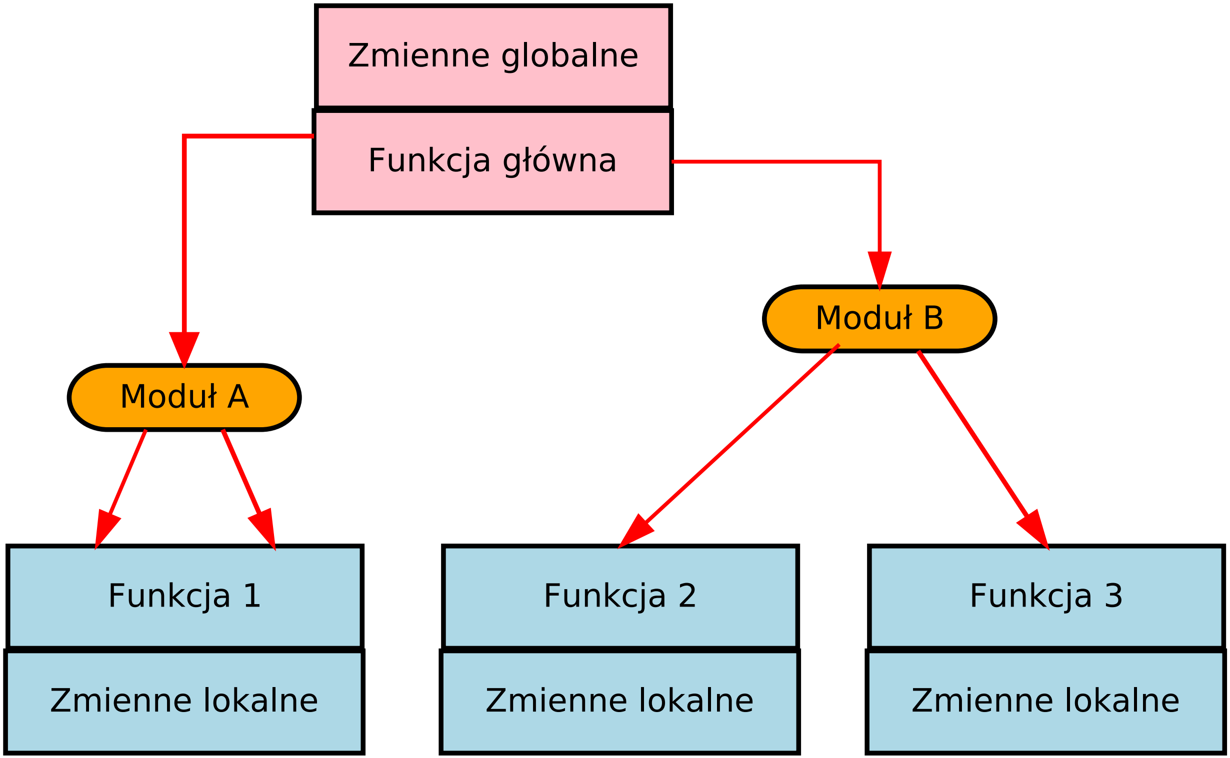
More details:

`https://en.wikipedia.org/wiki/Comparison_of_programming_paradigms`

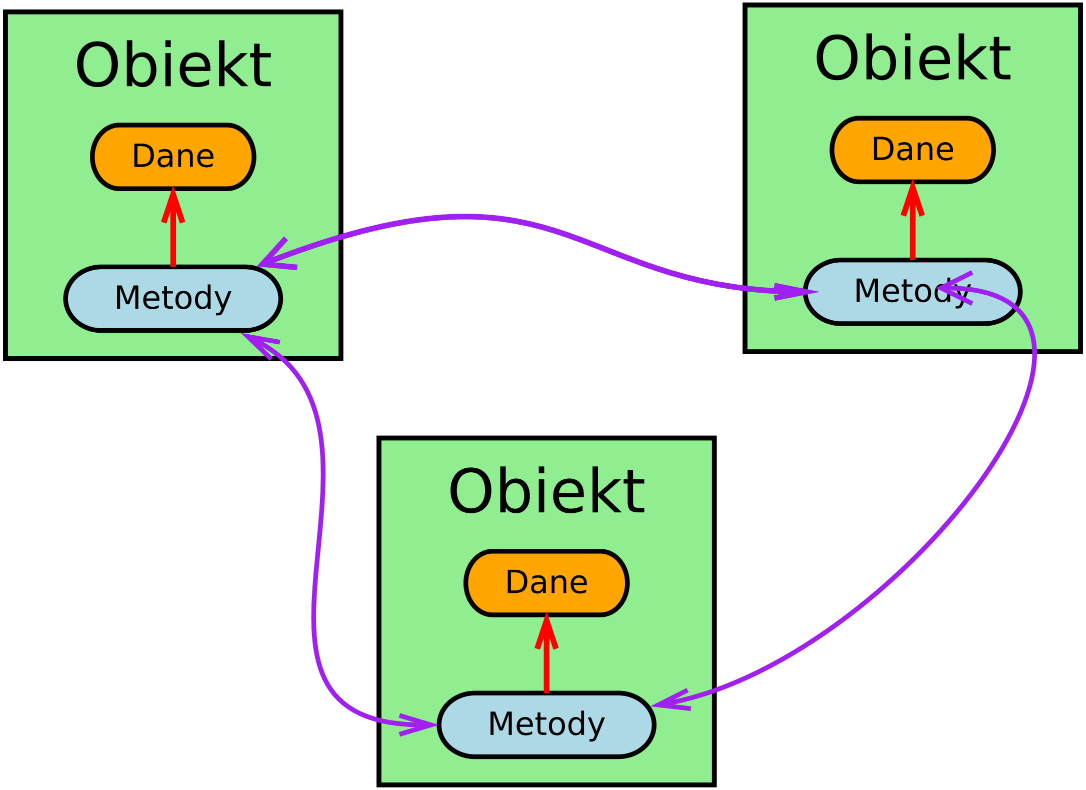
High-level programming languages paradigms

- Imperative programming – language example Octave, C.
  Characterized by sequential execution of instructions, use of variables that represent memory locations, use of assignment statement to change the values of these variables.
- Functional programming – language example Lisp, Haskel.
  Based on mathematical concept of function. Computation is expressed in term of the evaluation of functions. No variables and no assignment statements. Repetition expressed in terms of recursive function calls.
- Logic programming – language example PROLOG.
  Based on principles of symbolic logic.

Structural programming



Object oriented programming



Does programming language matter?

- Quest for the "best" language
- Domain Specific Languages (DSL)
- "Hot language" : Python

One can ask other question : what is the influence of information technology on the efficiency and efficacy of scientific research? While looking simple, this is very hard question to answer on scientific ground. (Even if we consider relatively narrow scientific field).

Quest for the "best" language

TIOBE Index : `https://www.tiobe.com/tiobe-index/`

Sep 2019	Sep 2018	Change	Language	Ratings	Change
1	1		Java	16.661%	-0.78%
2	2		C	15.205%	-0.24%
3	3		Python	9.874%	+2.22%
4	4		C++	5.635%	-1.76%
5	6	change	C#	3.399%	+0.10%
18	13	change	MATLAB	1.062%	-0.21%
34			Fortran	0.359%	
35			Ada	0.346%	
36			Julia	0.338%	
37			Kotlin	0.337%	

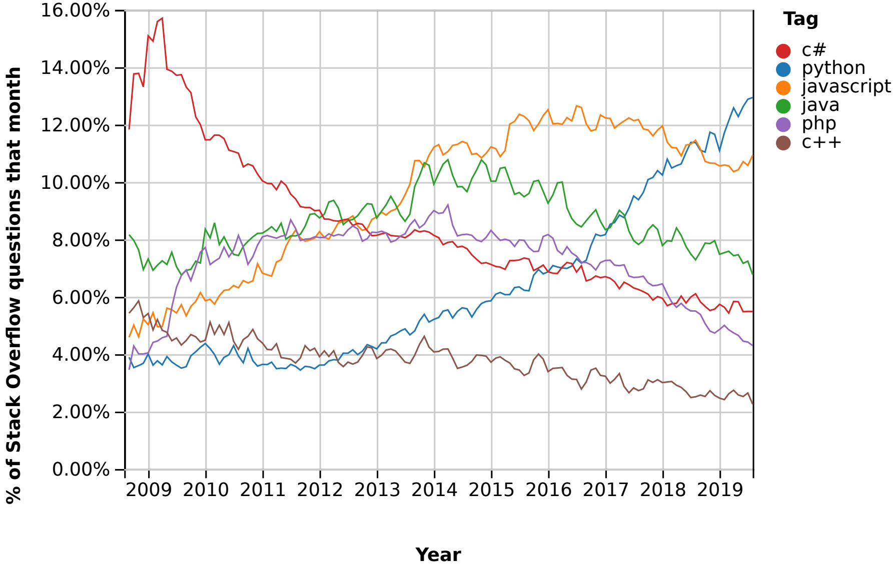
By age: Fortan (1957/1977), C(1972), C++ (1980), Matlab(1984), Wolfram (1988), Python (1991), Fortan90 (1992), R (1993), Java (1995), C# (2001), Scala (2003), Kotlin (2011), Julia (2012)

Domain Specific Languages (DSL)

- Languages for geometric modelling : e.g. GMSH
- Languages for FEM: FreeFem, Fiat(FENics), Unified Form Language(FENIcs), GetFEM high level assembly
- Languages for tensors: packages for Mathematica, Maple, Maxima, Matlab, R, Yoric,
- Languages for programming graphics: Asymptote, tikz(LaTeX)
- Languages for visual programming : Dynamo

“Hot” language: Python

From: Stack Overflow Trends



Now, the Developer Survey 2019 reveals that Python has “edged out Java” and is the second most loved language. Stack Overflow refers to Python as the “fastest-growing major programming language”.

`https://jaxenter.com/stack-overflow-dev-survey-2019-157815.html`

Python: basic information

`https://en.wikipedia.org/wiki/Python_(programming_language)`
Python - interpreted high level, general purpose language. Designed by Guido van Rossum and published for the first time in 1991. Supports different programming paradigms, in particular object oriented programming. In 2008 version 3.0 has been published. Python 3.0 is incompatible with version 2. The last version in the branch 2 is 2.7 and this branch will not be continued.

```python
def hello_world():
    print('Hello world')

if __name__ == '__main__':
    hello_world()

```

Python implementations

- CPython - basic implementation in C
- Stackless Python - implementation in C (avoids using system stack)
- Jython - implementation in Java
- IronPython - implementation for .NET Framework
- PyPy - implementation of Python in Python

Execution of Python programs

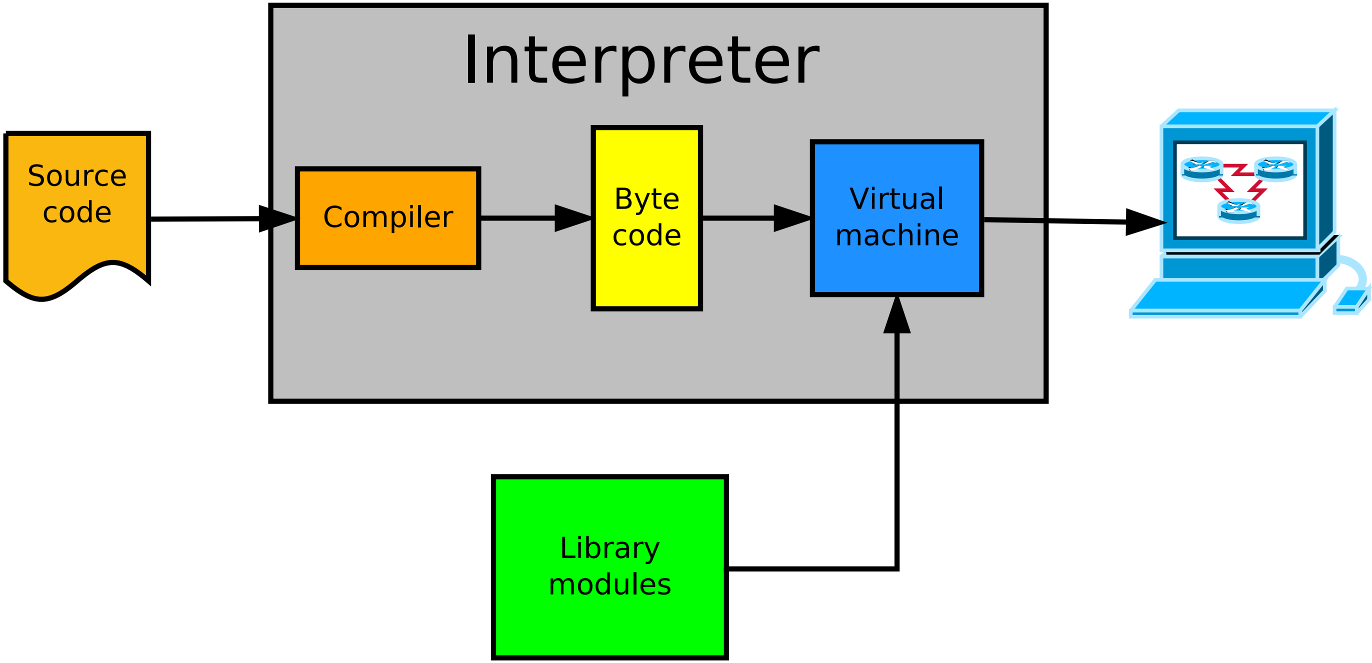


Figure based on `https://indianpythonista.wordpress.com/2018/01/04/how-python-runs/`

“Simple” program

```python
# -*- coding: utf-8 -*-
"""
Calculate distance between two points with coordinates
read from file my_points.txt
"""
import math

def read_points(filename):
    with open('my_points.txt') as input_file:
        points = []
        for line in input_file:
            points.append(list(map(float, line.split())))
        return points

def segment_length(pt1, pt2):
    length = 0
    for c1, c2 in zip(pt1, pt2):
        length += (c1-c2)**2
    return math.sqrt(length)

if __name__ == '__main__':
    pts = read_points('my_points.txt')
    if len(pts) == 2:
        distance = segment_length(pts[0], pts[1])
        print(f"Distance between points : {distance}")
    else:
        print(f"Error: expected 2 points got {len(pts)}")
```

Python distributions

- Offical CPython distribution
- Anaconda (Continuum Analytics)
- Canopy (Enthought)
- ActivePython (ActiveState)

Zen of Python in Easter Egg

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Octave

GNU Octave `http://www.octave.org`

- numerical computing environment for scientific and engineering applications,
- a tool for matrix manipulations
- available under GNU GPL license,
- sources and binary versions: http://www.octave.org/download.html.

Brief history

1988 – origins of Octave
1992 – John W. Eaton joins Octave team. 1993 – first Octave alpha version 1994 – version 1.0.
2020 – the newest stable version 5.2

Octave application areas

- – Numerical computing
- – Data analysis
- – Data visualisation
- – Prototyping of numerical applications

Octave components

- Octave language – scripting, high level, matrix based
- Octave interpreter
- Numerical libraries
- Interface to visualisation tools (gnuplot, VTK)

Octave working modes

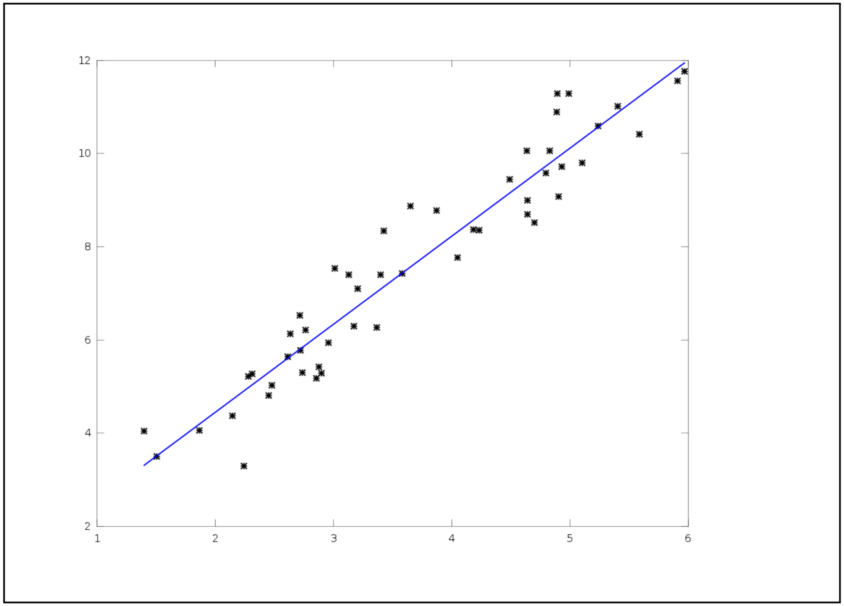
Octave can be used in two modes:

- interactive,
- batch processing.

– Both modes are interpreted and support the same commands
– Customary suffix for Octave scripts: '.m'
– Instruction separators: ';' ','.
– '...' line continuation symbol
– Comment lines start with '%' or '#'.
– Commands are case sensitive

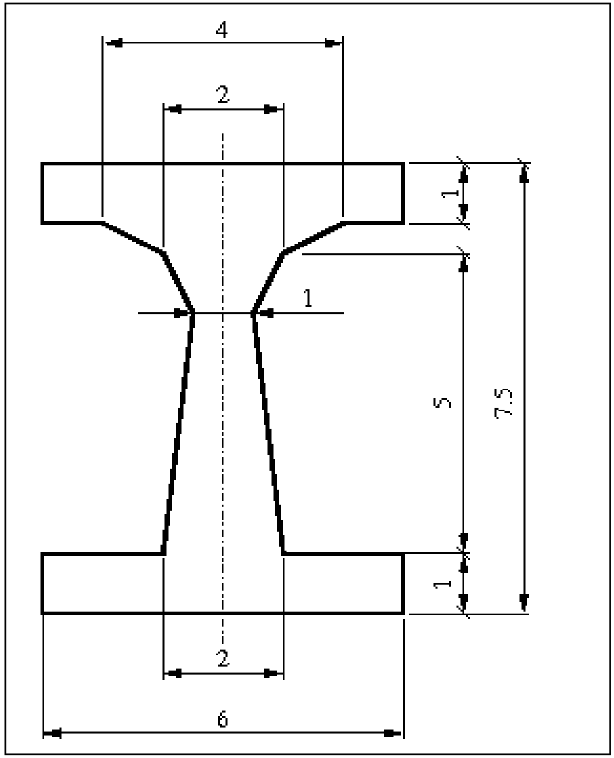
Example: linear regression

```
x = linspace(1,5,50)
y = 2*x+1;
rx = rand(50,1);
ry = rand(50,1);
nf = 1.2
yn = y+nf*ry';
xn = x+nf*rx';
p = polyfit (xn, yn, 1);
xz=linspace(min(xn),max(xn));
yz=p(2) + p(1)*xz;
plot(xn,yn,"*0",
  "markersize", 8,
  "linewidth", 2,
  xz,yz,"-", "linewidth", 2);
pause()
print("linfit.png")
```



Example: Polygon area

```
XY = load("ttshape.
      dat");
# scale to cm
XY = XY/450;
X = XY(:,1);
Y = XY(:,2);
area = polyarea(X,Y)
```



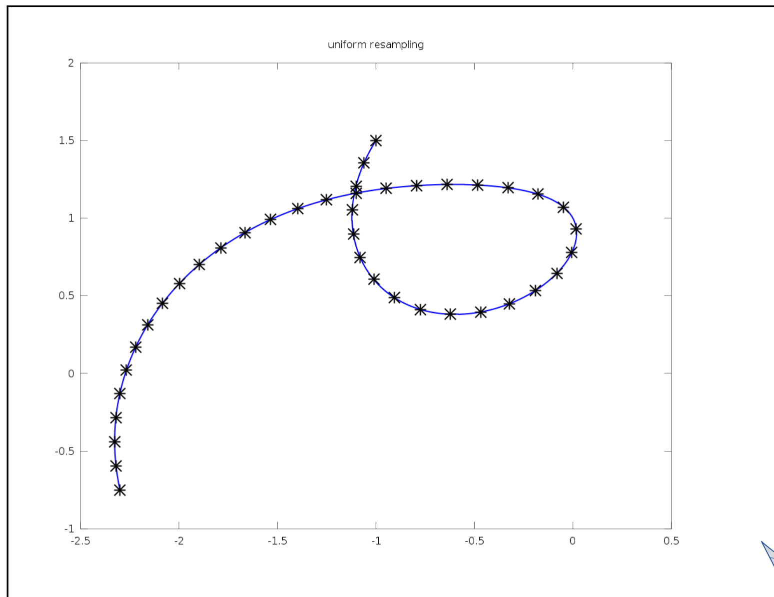

```
2700 1575
5400 1575
5400 2025
4950 2025
4500 2250
4275 2700
4500 4500
5400 4500
5400 4950
2700 4950
2700 4500
3600 4500
3825 2700
3600 2250
3150 2025
2700 2025
2700 1575
```

Example: Curve discretisation

Discretisation with segments of equal length

```
R = 2; r = 3; d = 1.5;
th = linspace (0, 2*pi, 1000);
x = (R-r) * cos (th) + d*sin ((R-r)/r * th);
y = (R-r) * sin (th) + d*cos ((R-r)/r * th);
x += 0.3*exp (-(th-0.8*pi).^2);
y += 0.4*exp (-(th-0.9*pi).^2);

[xs, ys] = unresamp2 (x, y, 40);
plot (x, y, "-", "linewidth", 2,
 xs, ys, "*0", "linewidth",2,"markersize", 16);
title ("uniform resampling")
pause()
print("unilen.png")
```

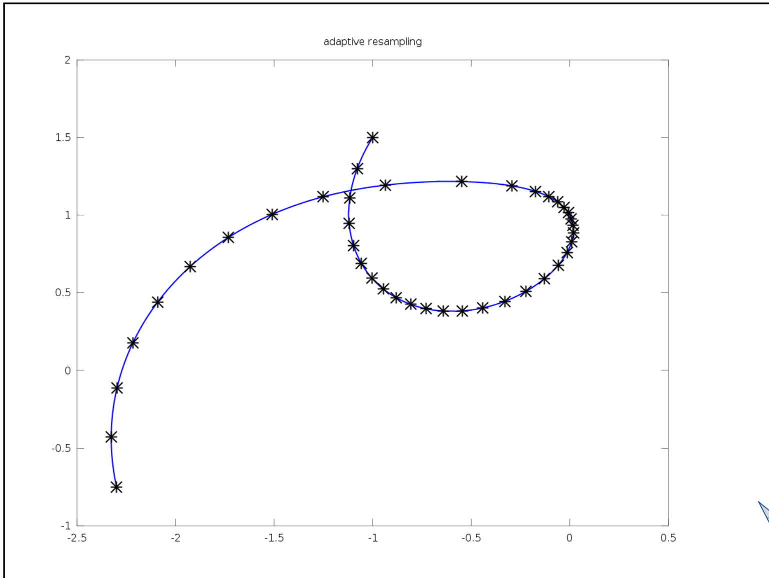


Example: Curve discretisation

Discretisation with equal angle increment between segments

```
R = 2; r = 3; d = 1.5;
th = linspace (0, 2*pi, 1000);
x = (R-r) * cos (th) + d*sin ((R-r)/r * th);
y = (R-r) * sin (th) + d*cos ((R-r)/r * th);
x += 0.3*exp (-(th-0.8*pi).^2);
y += 0.4*exp (-(th-0.9*pi).^2);

[xs, ys] = adresamp2 (x, y, 40);
plot (x, y, "-", "linewidth", 2,
 xs, ys, "*0", "linewidth",2,"markersize", 16);
title ("adaptive resampling")
pause()
print("adaplen.png")
```



Mathematical software

Mathematical software is software used to model, analyse, or calculate numeric, symbolic, or geometric data.(wikipedia)

Application areas:

- Symbolic mathematics – computer algebra systems
- Statistics
- Geometry
- Numerical analysis

Categories of software:

- applications, e.g. GeoGebra
- interactive platforms, e.g. Scilab, Sage
- problems solving environments (PSE), e.g. Diffpack
- software libraries, e.g. GNU Scientific Library, Trilinos

Selected software packages

Alphabetical list:

- Diffpack
- Maple
- MathCad
- Mathematica
- Matlab
- Maxima `http://maxima.sourceforge.net/`
- Octave `http://www.gnu.org/software/octave/`
- R `http://www.r-project.org/`
- Sage `http://www.r-project.org/`
- Scilab `http://www.scilab.org/`

Software taxonomies

Licensing:

- Open Source
- Proprietary

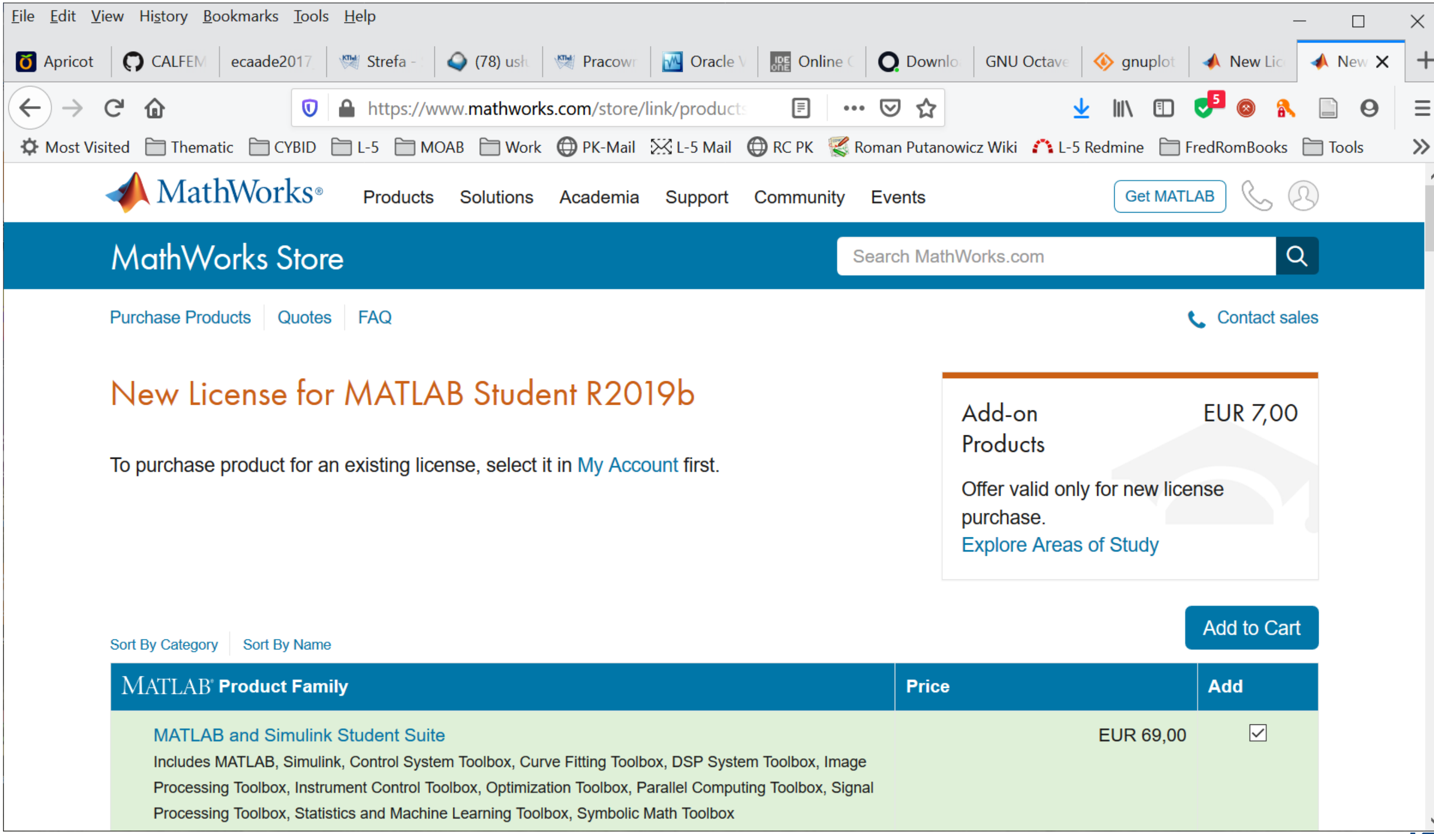
Scope:

- Symbolic computations
- Numerical computations

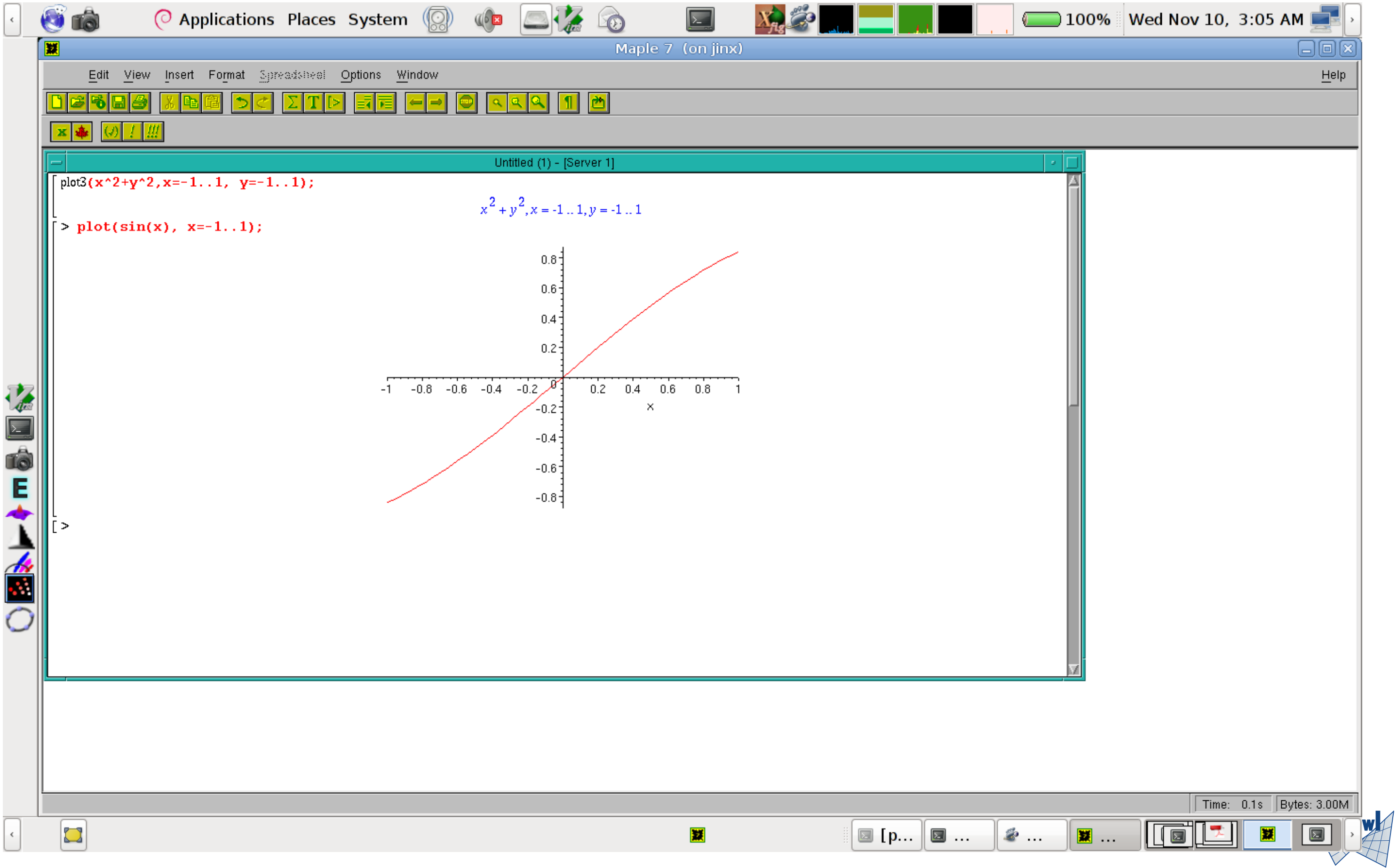
Operating mode:

- WYSWIG, GUI
- traditional programming, CLI

Matlab

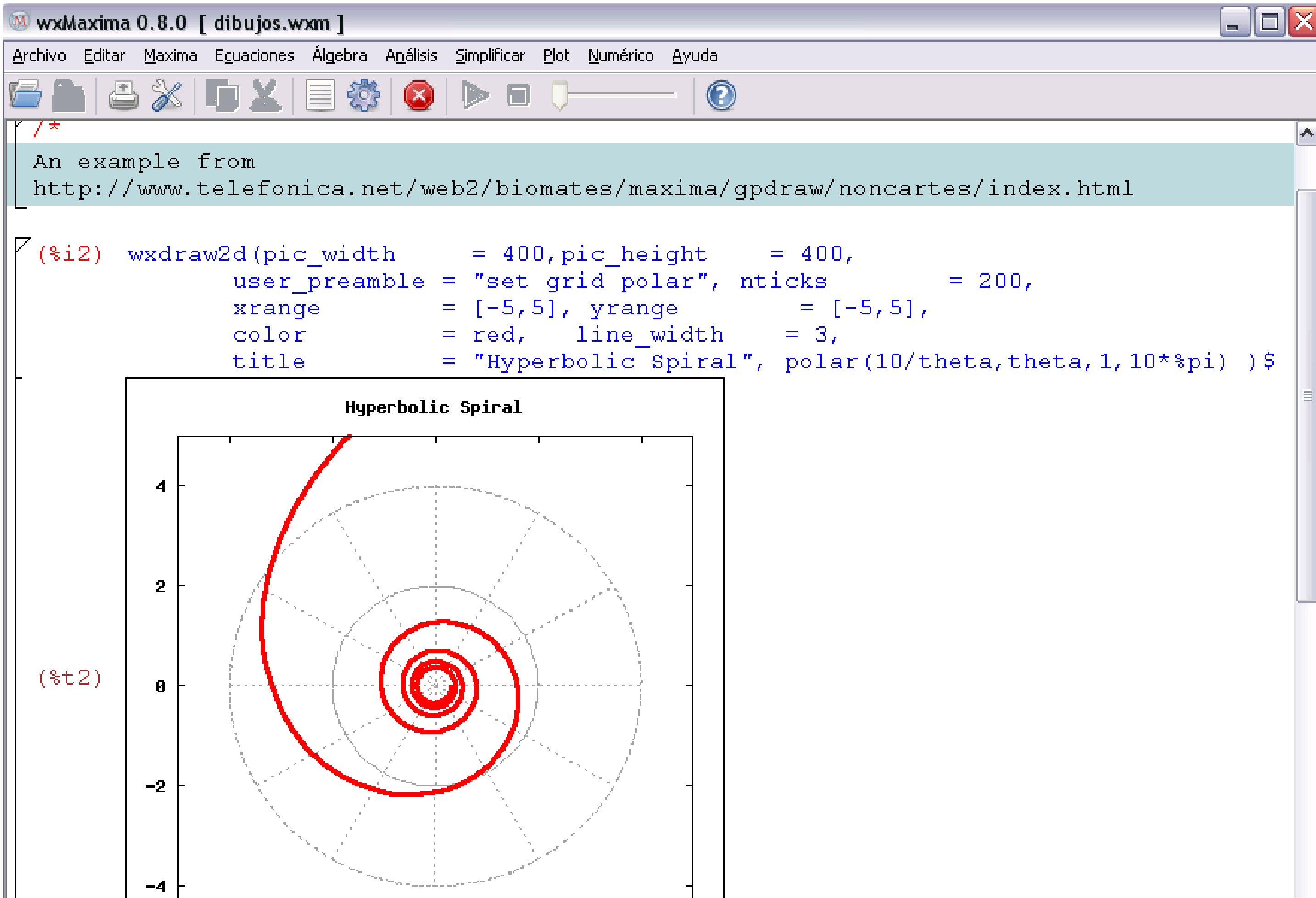

File Edit View History Bookmarks Tools Help
https://www.mathworks.com/store/link/products
Most Visited
Thematic
CYBID
L-5
MOAB
Work
PK-Mail
L-5 Mail
RC PK
Roman Putanowicz Wiki
L-5 Redmine
FredRomBooks
Tools
MathWorks
Products
Solutions
Academia
Support
Community
Events
Get MATLAB
MathWorks Store
Search MathWorks.com
Purchase Products
Quotes
FAQ
Contact sales
New License for MATLAB Student R2019b
To purchase product for an existing license, select it in My Account first.
Add-on Products
EUR 7,00
Offer valid only for new license purchase.
Explore Areas of Study
Add to Cart
Sort By Category
Sort By Name
MATLAB Product Family
Price
Add
MATLAB and Simulink Student Suite
EUR 69,00
Includes MATLAB, Simulink, Control System Toolbox, Curve Fitting Toolbox, DSP System Toolbox, Image Processing Toolbox, Instrument Control Toolbox, Optimization Toolbox, Parallel Computing Toolbox, Signal Processing Toolbox, Statistics and Machine Learning Toolbox, Symbolic Math Toolbox

Maple

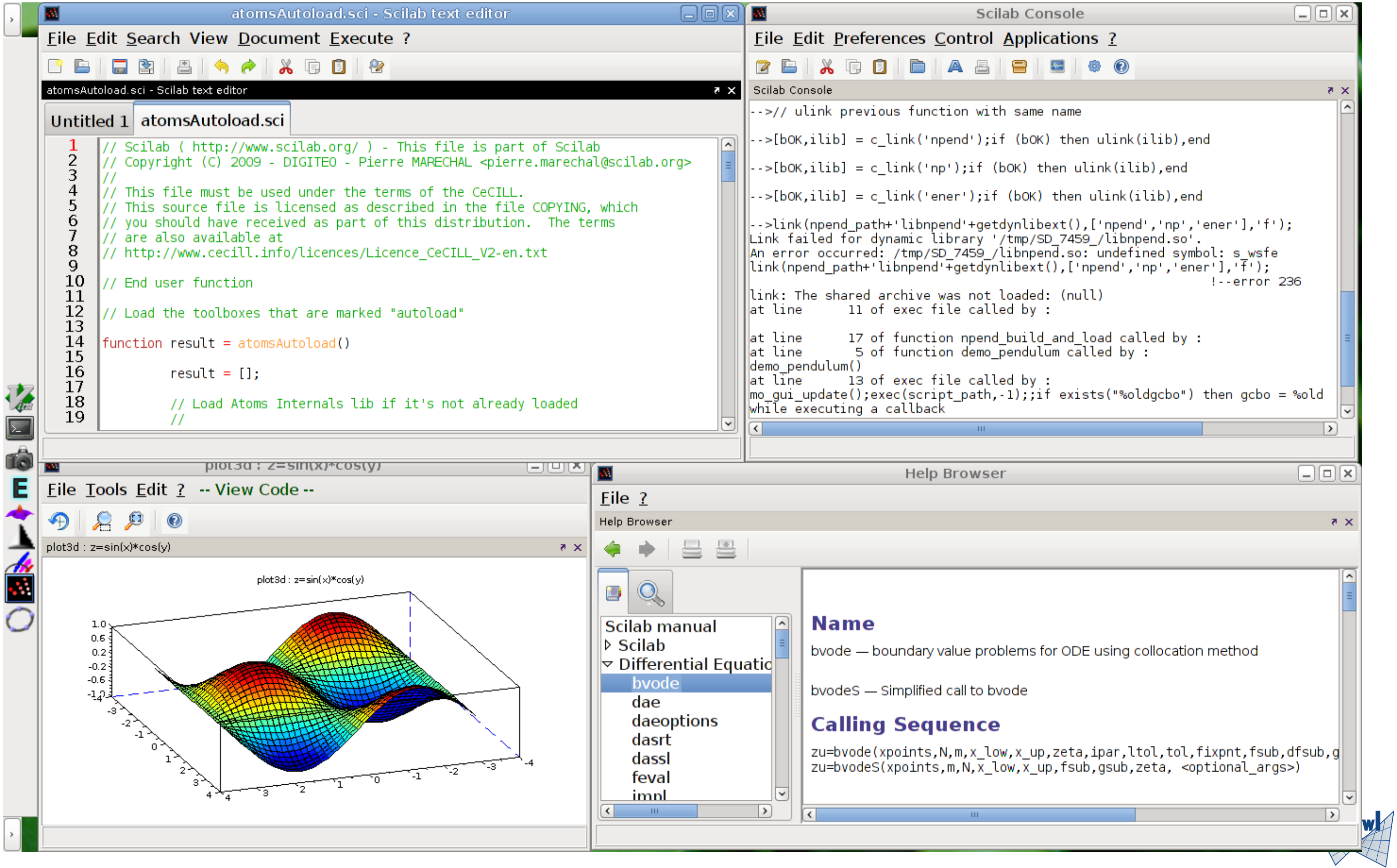


Applications Places System
100%
Wed Nov 10, 3:05 AM
Maple 7 (on jinx)
Edit View Insert Format Spreadsheet Options Window
Help
Untitled (1) - [Server 1]
plot3(x^2+y^2,x=-1..1, y=-1..1);
x^2 + y^2, x = -1 .. 1, y = -1 .. 1
> plot(sin(x), x=-1..1);
0.8
0.6
0.4
0.2
-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 1
x
-0.2
-0.4
-0.6
-0.8
Time: 0.1s
Bytes: 3.00M

Maxima + wx = wxMaxima

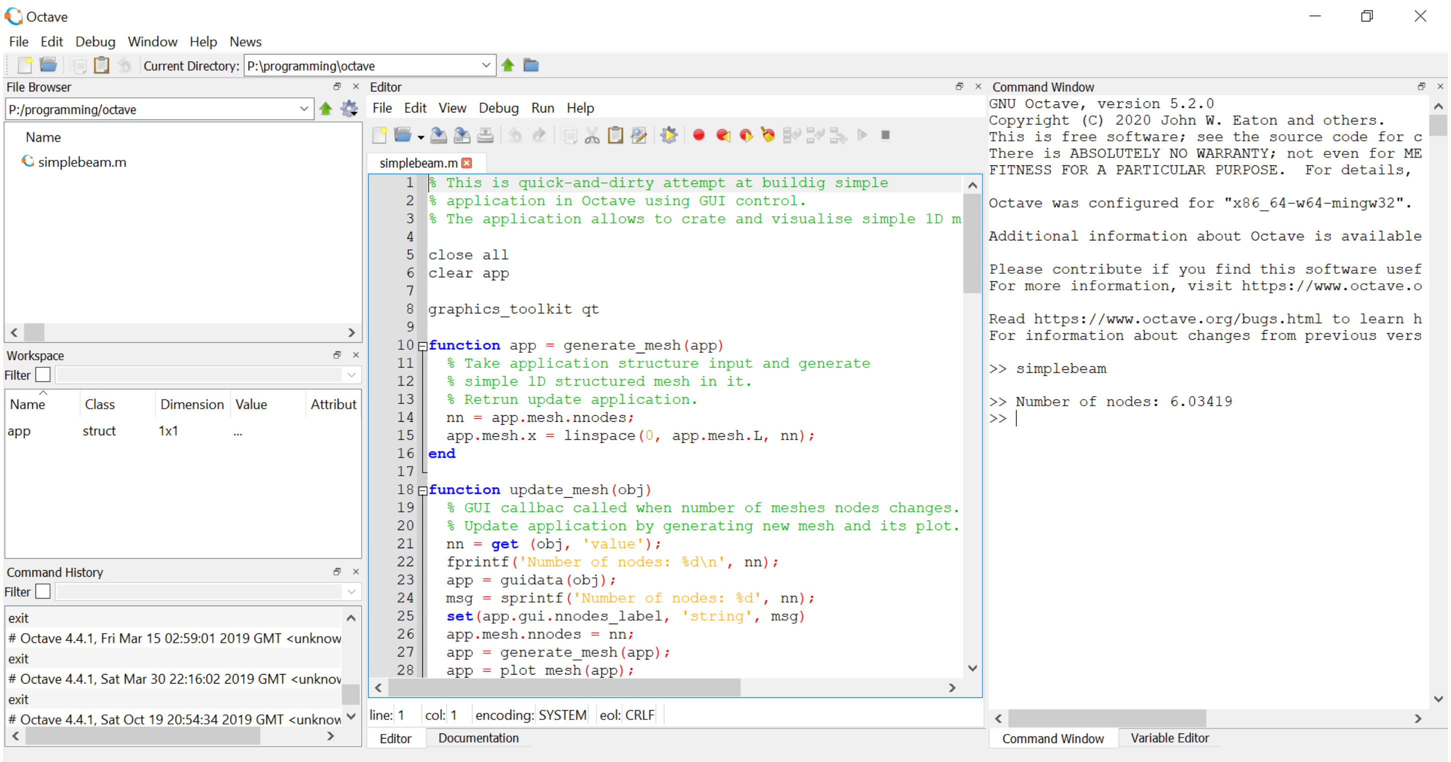


Scilab



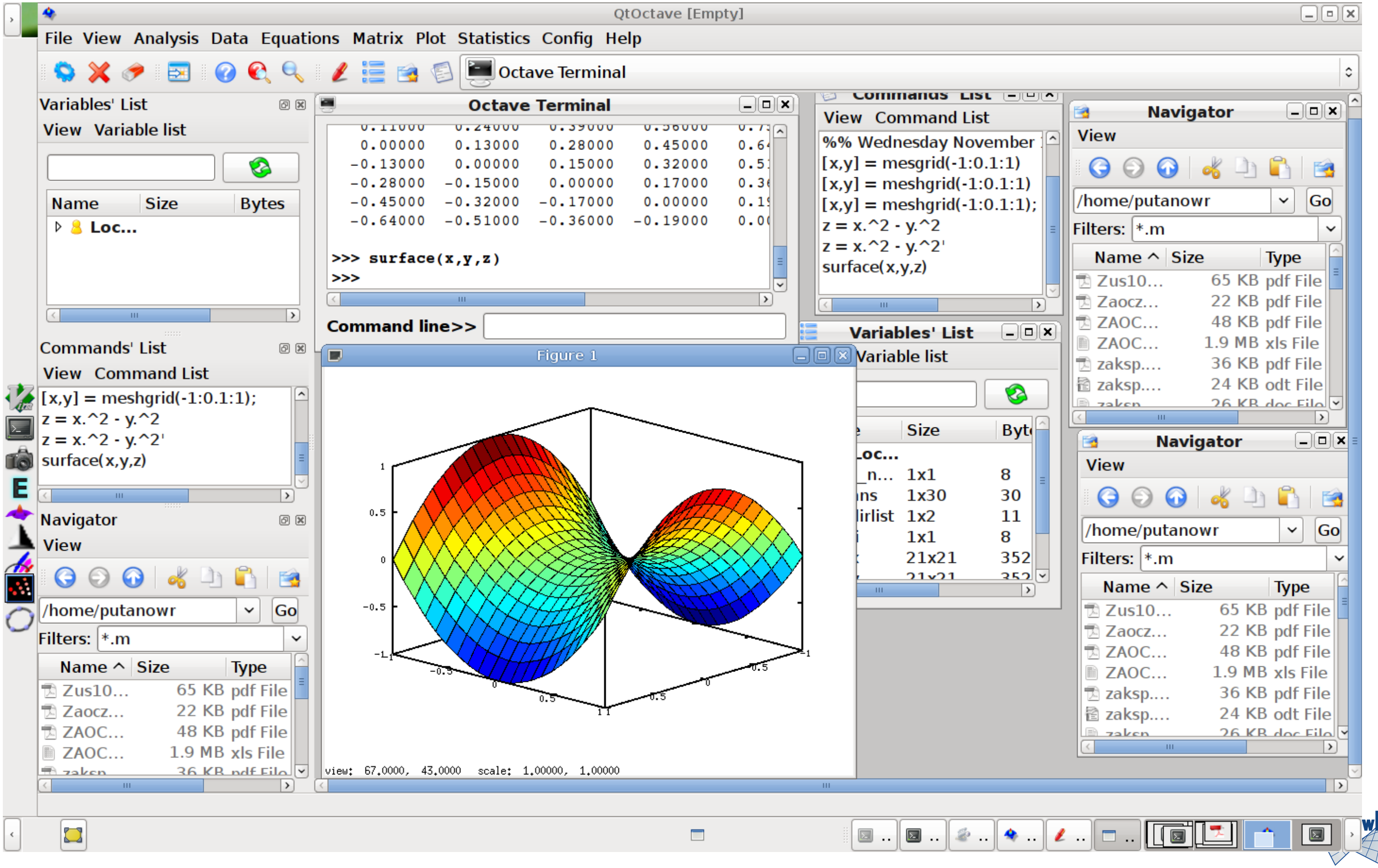
atomsAutoload.sci - Scilab text editor
File Edit Search View Document Execute ?
Untitled 1
atomsAutoload.sci
// Scilab (http://www.scilab.org/) - This file is part of Scilab
// Copyright (C) 2009 - DIGITEO - Pierre MARECHAL <pierre.marechal@scilab.org>
// This file must be used under the terms of the CeCILL.
// This source file is licensed as described in the file COPYING, which
// you should have received as part of this distribution. The terms
// are also available at
// http://www.cecill.info/licences/Licence_CeCILL_V2-en.txt
// End user function
// Load the toolboxes that are marked "autoload"
function result = atomsAutoload()
result = [];
// Load Atoms Internals lib if it's not already loaded
Scilab Console
File Edit Preferences Control Applications ?
-->// ulink previous function with same name
-->[bOK,ilib] = c_link('npend');if (bOK) then ulink(ilib),end
-->[bOK,ilib] = c_link('np');if (bOK) then ulink(ilib),end
-->[bOK,ilib] = c_link('ener');if (bOK) then ulink(ilib),end
-->link(npend_path+'libnpend'+getdynlibext(),['npend','np','ener'],'f');
Link failed for dynamic library '/tmp/SD_7459_/libnpend.so'.
An error occurred: /tmp/SD_7459_/libnpend.so: undefined symbol: s_wsfe
link(npend_path+'libnpend'+getdynlibext(),['npend','np','ener'],'f');
!--error 236
link: The shared archive was not loaded: (null)
at line 11 of exec file called by :
at line 17 of function npend_build_and_load called by :
at line 5 of function demo_pendulum called by :
demo_pendulum()
at line 13 of exec file called by :
mo_gui_update();exec(script_path,-1);;if exists("%oldgcbo") then gcbo = %old
while executing a callback
plot3d : z=sin(x)*cos(y)
File Tools Edit ? -- View Code --
Help Browser
File ?
Scilab manual
Scilab
Differential Equatic
bvode
dae
daeoptions
dasrt
dassl
feval
impl
Name
bvode — boundary value problems for ODE using collocation method
bvodeS — Simplified call to bvode
Calling Sequence
zu=bvode(xpoints,N,m,x_low,x_up,zeta,ipar,ltol,tol,fixpnt,fsub,dfsub,g
zu=bvodeS(xpoints,m,N,x_low,x_up,fsub,gsub,zeta, <optional_args>)

Octave



Octave
File Edit Debug Window Help News
Current Directory: P:\programming\octave
File Browser
P:/programming/octave
Name
simplebeam.m
Workspace
Filter
Name Class Dimension Value Attribut
app struct 1x1 ...
Command History
Filter
exit
Octave 4.4.1, Fri Mar 15 02:59:01 2019 GMT <unknow
exit
Octave 4.4.1, Sat Mar 30 22:16:02 2019 GMT <unknov
exit
Octave 4.4.1, Sat Oct 19 20:54:34 2019 GMT <unknow
Editor
File Edit View Debug Run Help
simplebeam.m
% This is quick-and-dirty attempt at buildig simple
% application in Octave using GUI control.
% The application allows to crate and visualise simple 1D m
close all
clear app
graphics_toolkit qt
function app = generate_mesh(app)
% Take application structure input and generate
% simple 1D structured mesh in it.
% Retrun update application.
nn = app.mesh.nnodes;
app.mesh.x = linspace(0, app.mesh.L, nn);
end
function update_mesh(obj)
% GUI callbac called when number of meshes nodes changes.
% Update application by generating new mesh and its plot.
nn = get (obj, 'value');
fprintf('Number of nodes: %d\n', nn);
app = guidata(obj);
msg = sprintf('Number of nodes: %d', nn);
set(app.gui.nnodes_label, 'string', msg)
app.mesh.nnodes = nn;
app = generate_mesh(app);
app = plot mesh(app);
line: 1 col: 1 encoding: SYSTEM eol: CRLF
Editor Documentation
Command Window
GNU Octave, version 5.2.0
Copyright (C) 2020 John W. Eaton and others.
This is free software; see the source code for c
There is ABSOLUTELY NO WARRANTY; not even for ME
FITNESS FOR A PARTICULAR PURPOSE. For details,
Octave was configured for "x86_64-w64-mingw32".
Additional information about Octave is available
Please contribute if you find this software usef
For more information, visit https://www.octave.o
Read https://www.octave.org/bugs.html to learn h
For information about changes from previous vers
>> simplebeam
>> Number of nodes: 6.03419
>>
Command Window Variable Editor

Octave + Qt = QtOctave



Scalar product: C versus Octave program

```c
#include <stdio.h>

#define DIM 3
int main() {
  char *fname = "vect.dat";
  FILE *fh = fopen(fname, "r");

  double u[DIM];
  double v[DIM];
  int i;

  for (i=0; i<DIM; i++) {
    fscanf(fh, "%lf", u+i);
  }
  for (i=0; i<DIM; i++) {
    fscanf(fh, "%lf", v+i);
  }
  double s=0;
  for (i=0; i<DIM; i++) {
    s += u[i]*v[i];
  }
  printf("Scalar product of u and v: %g\n",
```

```octave
fname = 'vect.dat';
fh = fopen(fname, 'r
     ');
dim = 3;
u = fscanf(fh, '%lf
     ', dim);
v = fscanf(fh, '%lf
     ', dim);
s=0;
for i=1:dim
  s+=u(i)*v(i);
end
printf('Scalar
     product of u
     and v: %g', s);
```

Scalar product: Octave versus Octave program

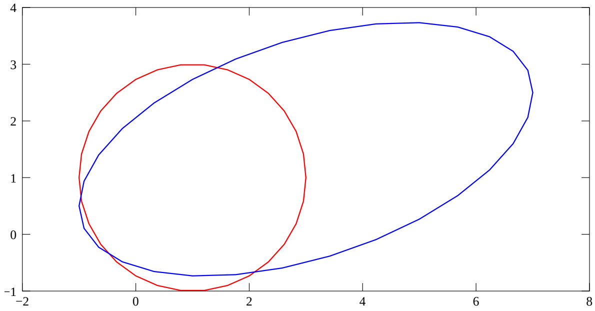
```
fname = 'vect.dat';
fh = fopen(fname, 'r');
dim = 3;
u = fscanf(fh, '%lf', dim);
v = fscanf(fh, '%lf', dim);
s=0;
for i=1:dim
  s+=u(i)*v(i);
end
printf('Scalar product of u and v: %g', s);
```

```
fname = 'vect.dat';
A = load('vect.dat');
u = A(1:3);
v = A(4:6);
s = dot(u,v);
printf('Scalar product of u and v: %g\n', s);
```

Case study: affine transformation in 2D

Problem: Write Octave program that illustrates affine transformation of a circle.

$$\hat{x} = ax + by + c$$
$$\hat{y} = ex + fy + g$$



$$\hat{x} = T_{11}x + T_{12}y + T_{13}$$
$$\hat{y} = T_{21}x + T_{22}y + T_{23}$$

Case study: affine transformation in 2D

```
function [XY] = polygon(x,y,R,N)
  t = linspace(0,2*pi,N+1);
  XY = zeros(N+1,2);
  XY(:,1) = x + R*cos(t);
  XY(:,2) = y + R*sin(t);
end
```

```
function plot_poly(XY, color)
  h = line(XY(:,1), XY(:,2));
  set(h, 'color',color);
end
```

```
function [NXY] = transform_poly(XY, T)
  NXY=zeros(size(XY));
  for i=1:rows(XY)
    x = XY(i,1);
    y = XY(i,2);
    xh = T(1,1)*x + T(1,2)*y + T(1,3);
    yh = T(2,1)*x + T(2,2)*y + T(2,3);
    NXY(i,:) = [xh,yh];
  endfor
endfunction
```

```
N=30
xy = polygon(1,1,2,N);
plot_poly(xy,"red");
T = [2.0, 0.0, 1.0;
0.5, 1.0, 0.0];
xy1 = transform_poly(xy, T);
plot_poly(xy1,"blue");
axis("equal")
print("affine.fig")
pause()
```

Scalar product: Octave version 1

```
fname = 'vect.dat';
fh = fopen(fname, 'r');
dim = 3;
u = fscanf(fh, '%lf', dim);
v = fscanf(fh, '%lf', dim);
s=0;
for i=1:dim
  s+=u(i)*v(i);
end
printf('Scalar product of u and v: %g', s);
```

Scalar product: Octave version 2

```
fname = 'vect.dat';
A = load('vect.dat');
u = A(1:3);
v = A(4:6);
s = dot(u,v);
printf('Scalar product of u and v: %g\n', s);
```

Case study: affine transformation in 2D

```
function [XY] = polygon(x,y,R,N)
  t = linspace(0,2*pi,N+1);
  XY = zeros(N+1,2);
  XY(:,1) = x + R*cos(t);
  XY(:,2) = y + R*sin(t);
end
```

back to the main code

Case study: affine transformation in 2D

```
function plot_poly(XY, color)
  h = line(XY(:,1), XY(:,2));
  set(h, 'color',color);
end
```

back to the main code

Case study: affine transformation in 2D

```
function [NXY] = transform_poly(XY, T)
  NXY=zeros(size(XY));
  for i=1:rows(XY)
    x = XY(i,1);
    y = XY(i,2);
    xh = T(1,1)*x + T(1,2)*y + T(1,3);
    yh = T(2,1)*x + T(2,2)*y + T(2,3);
    NXY(i,:) = [xh,yh];
  endfor
endfunction
```

back to the main code

Case study: affine transformation in 2D

```
N=30
xy = polygon(1,1,2,N);
plot_poly(xy,"red");
T = [2.0, 0.0, 1.0;
0.5, 1.0, 0.0];
xy1 = transform_poly(xy, T);
plot_poly(xy1,"blue");
axis("equal")
print("affine.fig")
pause()
```

back to the main code

Thank you for your attention

Copyrights