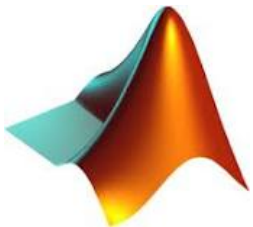
# Free alternatives to Matlab



Marc Fuentes - SED Bordeaux

# Outline

1. Why do people use Matlab ?

# Outline

1. Why do people use Matlab ?
2. Why use alternatives to Matlab ?

# Outline

1. Why do people use Matlab ?
2. Why use alternatives to Matlab ?
3. Some free alternatives to Matlab

# Outline

1. Why do people use Matlab ?
2. Why use alternatives to Matlab ?
3. Some free alternatives to Matlab
4. Octave

# Outline

1. Why do people use Matlab ?
2. Why use alternatives to Matlab ?
3. Some free alternatives to Matlab
4. Octave
5. Python

# Outline

# Why do people use Matlab ?

1. Matlab is *de facto* a *lingua franca* for numerical computation

# Why do people use Matlab ?

1. Matlab is *de facto* a *lingua franca* for numerical computation
2. lots of toolboxes and contributions from a big community of users (see Matlab File Exchange)

# Why do people use Matlab ?

1. Matlab is *de facto* a *lingua franca* for numerical computation
2. lots of toolboxes and contributions from a big community of users (see Matlab File Exchange)
3. scripting vs compiled application: easy to debug and to do experiments *a la mano*

# Why do people use Matlab ?

1. Matlab is *de facto* a *lingua franca* for numerical computation
2. lots of toolboxes and contributions from a big community of users (see Matlab File Exchange)
3. scripting vs compiled application: easy to debug and to do experiments *a la mano*
4. languages *a la* Matlab "may" seem more simple to learn than a true language

# Why use an alternative to Matlab ?

1. Matlab is **NOT** free

# Why use an alternative to Matlab ?

1. Matlab is **NOT** free
2. base your developments on a commercial, closed-sources, program may prevent you to do what you want with your software

# Why use an alternative to Matlab ?

1. Matlab is **NOT** free
2. base your developments on a commercial, closed-sources, program may prevent you to do what you want with your software
3. you have to own a license to use it at home

# Why use an alternative to Matlab ?

1. Matlab is **NOT** free
2. base your developments on a commercial, closed-sources, program may prevent you to do what you want with your software
3. you have to own a license to use it at home
4. what to do when the INRIA' server is down ?

# Why use an alternative to Matlab ?

1. Matlab is **NOT** free
2. base your developments on a commercial, closed-sources, program may prevent you to do what you want with your software
3. you have to own a license to use it at home
4. what to do when the INRIA' server is down ?
   - try to offer a  to the "matlab-flexlm.bordeaux.inria.fr"'s admin

# Why use an alternative to Matlab ?

1. Matlab is **NOT** free
2. base your developments on a commercial, closed-sources, program may prevent you to do what you want with your software
3. you have to own a license to use it at home
4. what to do when the INRIA' server is down ?
   - try to offer a  to the "matlab-flexlm.bordeaux.inria.fr"'s admin
   - try an alternative

# Some free alternatives to MATLAB

1. Octave : most compatible clone of MATLAB

# Some free alternatives to Matlab

1. Octave : most compatible clone of Matlab
2. Scilab(s):

# Some free alternatives to Matlab

1. Octave : most compatible clone of Matlab
2. Scilab(s):
   - official version (Digiteo, puis Scilab Enterprises ), uses Java (☹)

# Some free alternatives to Matlab

1. Octave : most compatible clone of Matlab
2. Scilab(s):
   - official version (Digiteo, puis Scilab Enterprises ), uses Java (☹)
   - Sciloslab (INRIA & ENPC)

# Some free alternatives to Matlab

1. Octave : most compatible clone of Matlab
2. Scilab(s):
   - official version (Digiteo, puis Scilab Enterprises ), uses Java (☹)
   - Sciloslab (INRIA & ENPC)
   - NSP (Tumbi)

# Some free alternatives to Matlab

1. Octave : most compatible clone of Matlab
2. Scilab(s):
   - official version (Digiteo, puis Scilab Enterprises ), uses Java (☹)
   - Sciloslab (INRIA & ENPC)
   - NSP (Tumbi)
3. Julia: new smart language developed at MIT: not enough mature

# Some free alternatives to Matlab

1. Octave : most compatible clone of Matlab
2. Scilab(s):
   - official version (Digiteo, puis Scilab Enterprises ), uses Java (☹)
   - Sciloslab (INRIA & ENPC)
   - NSP (Tumbi)
3. Julia: new smart language developed at MIT: not enough mature
4. Python with numerical extensions:

# Some free alternatives to Matlab

1. Octave : most compatible clone of Matlab
2. Scilab(s):
   - official version (Digiteo, puis Scilab Enterprises ), uses Java (☹)
   - Sciloslab (INRIA & ENPC)
   - NSP (Tumbi)
3. Julia: new smart language developed at MIT: not enough mature
4. Python with numerical extensions:
   - linear algebra, ODE solvers, etc...: Numpy & Scipy

# Some free alternatives to Matlab

1. Octave : most compatible clone of Matlab
2. Scilab(s):
   - official version (Digiteo, puis Scilab Enterprises ), uses Java (☹)
   - Sciloslab (INRIA & ENPC)
   - NSP (Tumbi)
3. Julia: new smart language developed at MIT: not enough mature
4. Python with numerical extensions:
   - linear algebra, ODE solvers, etc...: Numpy & Scipy
   - graphics: Matplotlib, Mayavi, VTK

# Some free alternatives to Matlab

1. Octave : most compatible clone of Matlab
2. Scilab(s):
   - official version (Digiteo, puis Scilab Enterprises ), uses Java (☹)
   - Sciloslab (INRIA & ENPC)
   - NSP (Tumbi)
3. Julia: new smart language developed at MIT: not enough mature
4. Python with numerical extensions:
   - linear algebra, ODE solvers, etc...: Numpy & Scipy
   - graphics: Matplotlib, Mayavi, VTK
   - parallelism: MPI4py, PyTrilinos

# Some free alternatives to Matlab

1. Octave : most compatible clone of Matlab
2. Scilab(s):
   - official version (Digiteo, puis Scilab Enterprises ), uses Java (☹)
   - Sciloslab (INRIA & ENPC)
   - NSP (Tumbi)
3. Julia: new smart language developed at MIT: not enough mature
4. Python with numerical extensions:
   - linear algebra, ODE solvers, etc...: Numpy & Scipy
   - graphics: Matplotlib, Mayavi, VTK
   - parallelism: MPI4py, PyTrilinos
5. R: syntax pretty different from Matlab . Very used in statistics community

# GNU Octave

1. old project (1988) coming originally from chemical reactor design problems

# GNU Octave

1. old project (1988) coming originally from chemical reactor design problems
2. rationale: “the students spent far too much time trying to figure out why their Fortran code failed and not enough time learning about chemical engineering”

# GNU Octave

1. old project (1988) coming originally from chemical reactor design problems
2. rationale: "the students spent far too much time trying to figure out why their Fortran code failed and not enough time learning about chemical engineering"
3. Today, Octave is a numerical scripting language which mimics good enough Matlab

# GNU Octave

1. old project (1988) coming originally from chemical reactor design problems
2. rationale: "the students spent far too much time trying to figure out why their Fortran code failed and not enough time learning about chemical engineering"
3. Today, Octave is a numerical scripting language which mimics good enough Matlab
4. can be also used a stand-alone math & matrix C++ library

# GNU Octave : features

1. all basic types (matrix, string, cell, *etc.*) from Matlab are implemented

# GNU Octave : features

1. all basic types (matrix, string, cell, *etc.*) from Matlab are implemented
2. owns its proper toolboxes system: octave-forge

# GNU Octave : features

1. all basic types (matrix, string, cell, *etc.*) from Matlab are implemented
2. owns its proper toolboxes system: octave-forge
3. the mexfile API is fully compatible with Matlab

# GNU Octave : features

1. all basic types (matrix, string, cell, *etc.*) from Matlab are implemented
2. owns its proper toolboxes system: octave-forge
3. the mexfile API is fully compatible with Matlab
4. recent features:

# GNU Octave : features

1. all basic types (matrix, string, cell, *etc.*) from Matlab are implemented
2. owns its proper toolboxes system: octave-forge
3. the mexfile API is fully compatible with Matlab
4. recent features:
   - broadcasting (like in Numpy)

# GNU Octave : features

1. all basic types (matrix, string, cell, *etc.*) from Matlab are implemented
2. owns its proper toolboxes system: octave-forge
3. the mexfile API is fully compatible with Matlab
4. recent features:
   - broadcasting (like in Numpy)
   - new graphic toolkit added (FLTK in addition to the old Gnuplot)

# GNU Octave : features

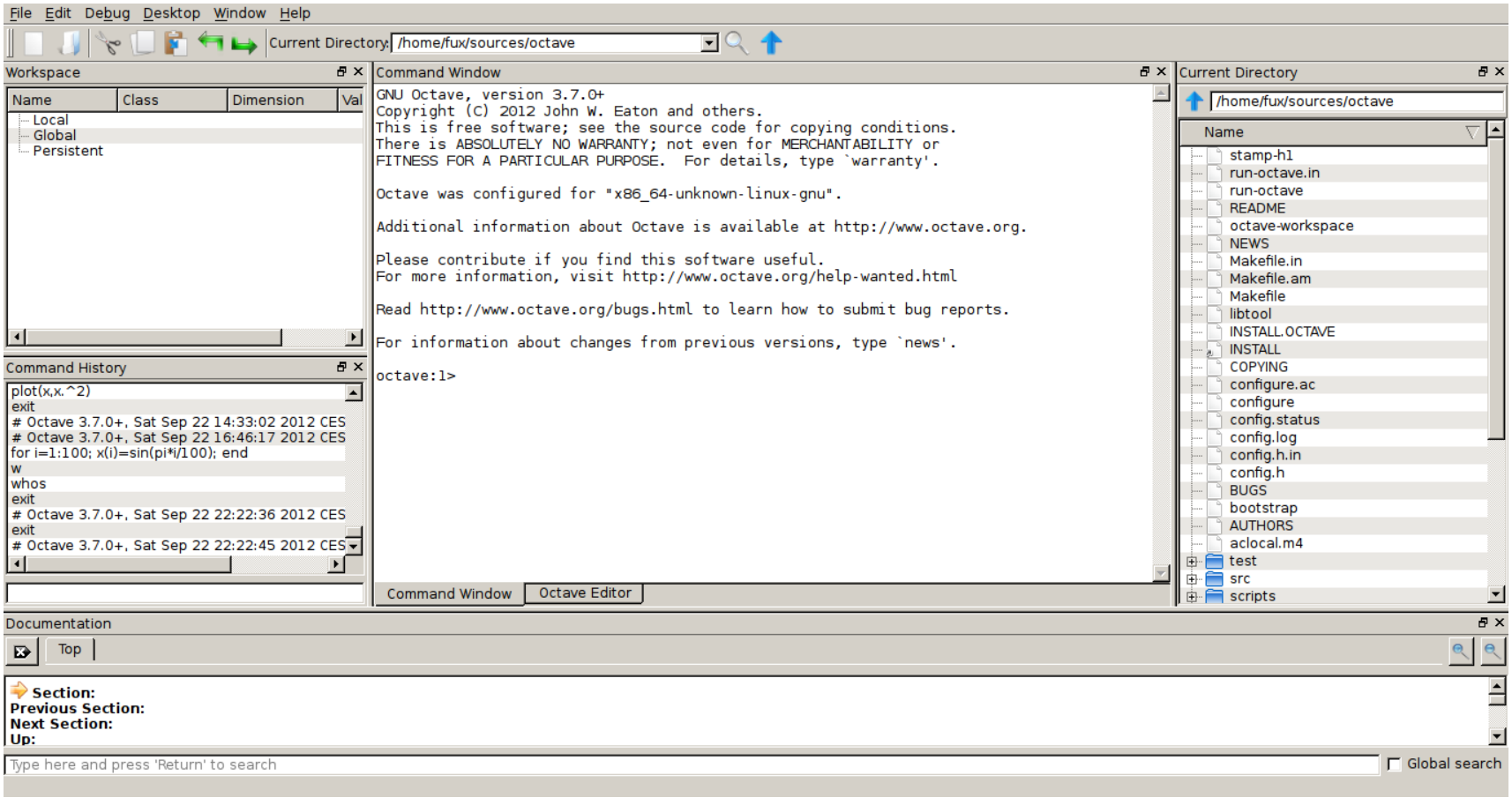
1. all basic types (matrix, string, cell, *etc.*) from Matlab are implemented
2. owns its proper toolboxes system: octave-forge
3. the mexfile API is fully compatible with Matlab
4. recent features:
   - broadcasting (like in Numpy)
   - new graphic toolkit added (FLTK in addition to the old Gnuplot)
   - proper Graphical User Interface

# GNU Octave : features

1. all basic types (matrix, string, cell, *etc.*) from Matlab are implemented
2. owns its proper toolboxes system: octave-forge
3. the mexfile API is fully compatible with Matlab
4. recent features:
   - broadcasting (like in Numpy)
   - new graphic toolkit added (FLTK in addition to the old Gnuplot)
   - proper Graphical User Interface
   - JIT compiler in LLVM (under development)

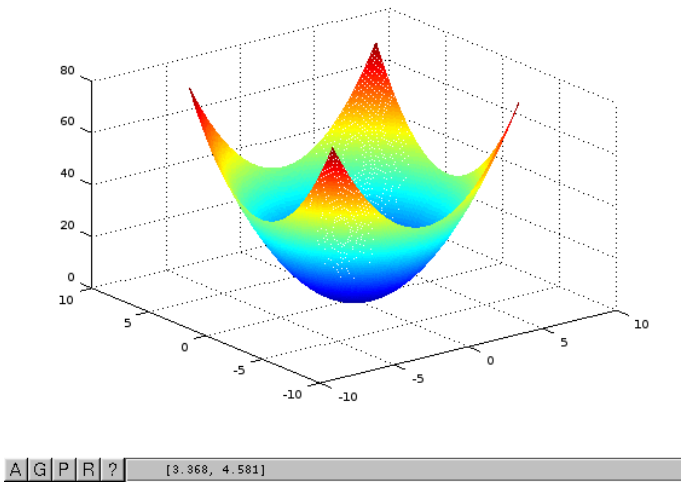
# GNU Octave : features

1. all basic types (matrix, string, cell, *etc.*) from Matlab are implemented
2. owns its proper toolboxes system: octave-forge
3. the mexfile API is fully compatible with Matlab
4. recent features:
   - broadcasting (like in Numpy)
   - new graphic toolkit added (FLTK in addition to the old Gnuplot)
   - proper Graphical User Interface
   - JIT compiler in LLVM (under development)
5. Uses `exist('OCTAVE_VERSION')` to define Octave specific parts of code


File Edit Debug Desktop Window Help
Current Directory: /home/fux/sources/octave
Workspace
Name Class Dimension Val
Local
Global
Persistent
Command History
plot(x,x.^2)
exit
# Octave 3.7.0+, Sat Sep 22 14:33:02 2012 CES
# Octave 3.7.0+, Sat Sep 22 16:46:17 2012 CES
for i=1:100; x(i)=sin(pi*i/100); end
w
whos
exit
# Octave 3.7.0+, Sat Sep 22 22:22:36 2012 CES
exit
# Octave 3.7.0+, Sat Sep 22 22:22:45 2012 CES
Command Window
GNU Octave, version 3.7.0+
Copyright (C) 2012 John W. Eaton and others.
This is free software; see the source code for copying conditions.
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY or
FITNESS FOR A PARTICULAR PURPOSE. For details, type `warranty'.
Octave was configured for "x86_64-unknown-linux-gnu".
Additional information about Octave is available at http://www.octave.org.
Please contribute if you find this software useful.
For more information, visit http://www.octave.org/help-wanted.html
Read http://www.octave.org/bugs.html to learn how to submit bug reports.
For information about changes from previous versions, type `news'.
octave:1>
Command Window
Octave Editor
Current Directory
/home/fux/sources/octave
Name
stamp-h1
run-octave.in
run-octave
README
octave-workspace
NEWS
Makefile.in
Makefile.am
Makefile
libtool
INSTALL.OCTAVE
INSTALL
COPYING
configure.ac
configure
config.status
config.log
config.h.in
config.h
BUGS
bootstrap
AUTHORS
aclocal.m4
test
src
scripts
Documentation
Top
Section:
Previous Section:
Next Section:
Up:
Type here and press 'Return' to search
Global search

# Octave : Graphics

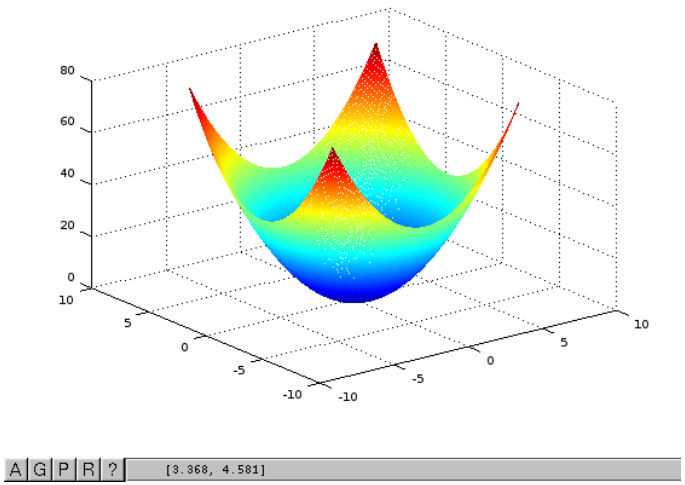
- say bye-bye to the ugly GNUPLOT: use `graphics_toolkit("fltk")`



# Octave : Graphics

- say bye-bye to the ugly GNUPLOT: use `graphics_toolkit("fltk")`
  - example: plot a parabola $M = \{(x, y, z) \in \mathbb{R}^3 \mid z = x^2 + y^2\}$

```
graphics_toolkit("fltk");
x=linspace(-2*pi,2*pi,200);
[X,Y]=meshgrid(x);
mesh(X,Y,X.^2+Y.^2);
```



# Octave : main drawbacks

1. essentially sequential (Matlab is multi-threaded). But linked against a threaded BLAS (ATLAS or MKL), some numerical algebra functions (e.g. $x = A\backslash b$) are multi-threaded

# Octave : main drawbacks

1. essentially sequential (Matlab is multi-threaded). But linked against a threaded BLAS (ATLAS or MKL), some numerical algebra functions (e.g. $x = A\backslash b$) are multi-threaded
2. the Octave toolboxes do not mimic precisely Matlab ones

# Octave : main drawbacks

1. essentially sequential (Matlab is multi-threaded). But linked against a threaded BLAS (ATLAS or MKL), some numerical algebra functions (e.g. $x = A\backslash b$) are multi-threaded
2. the Octave toolboxes do not mimic precisely Matlab ones
3. generally slower than Matlab

# Octave : main drawbacks

1. essentially sequential (Matlab is multi-threaded). But linked against a threaded BLAS (ATLAS or MKL), some numerical algebra functions (e.g. $x = A\backslash b$) are multi-threaded
2. the Octave toolboxes do not mimic precisely Matlab ones
3. generally slower than Matlab
4. some differences Octave /Matlab :

# Octave : main drawbacks

1. essentially sequential (Matlab is multi-threaded). But linked against a threaded BLAS (ATLAS or MKL), some numerical algebra functions (e.g. $x = A\backslash b$) are multi-threaded
2. the Octave toolboxes do not mimic precisely Matlab ones
3. generally slower than Matlab
4. some differences Octave /Matlab :
   - random generators initialization

# Octave : main drawbacks

1. essentially sequential (Matlab is multi-threaded). But linked against a threaded BLAS (ATLAS or MKL), some numerical algebra functions (e.g. $x = A\backslash b$) are multi-threaded
2. the Octave toolboxes do not mimic precisely Matlab ones
3. generally slower than Matlab
4. some differences Octave /Matlab :
   - random generators initialization
   - ode solvers are not natives (use Odepkg to have full compatibility)

# Octave : main drawbacks

1. essentially sequential (Matlab is multi-threaded). But linked against a threaded BLAS (ATLAS or MKL), some numerical algebra functions (e.g. $x = A\backslash b$) are multi-threaded
2. the Octave toolboxes do not mimic precisely Matlab ones
3. generally slower than Matlab
4. some differences Octave /Matlab :
   - random generators initialization
   - ode solvers are not natives (use Odepkg to have full compatibility)
   - do not support non-portable extensions like CUDA, Web, ActiveX, etc...

# Octave : main drawbacks

1. essentially sequential (Matlab is multi-threaded). But linked against a threaded BLAS (ATLAS or MKL), some numerical algebra functions (e.g. $x = A\backslash b$) are multi-threaded
2. the Octave toolboxes do not mimic precisely Matlab ones
3. generally slower than Matlab
4. some differences Octave /Matlab :
   - random generators initialization
   - ode solvers are not natives (use Odepkg to have full compatibility)
   - do not support non-portable extensions like CUDA, Web, ActiveX, etc...
   - do not support handles on embedded functions

# Octave : main drawbacks

1. essentially sequential (Matlab is multi-threaded). But linked against a threaded BLAS (ATLAS or MKL), some numerical algebra functions (e.g. $x = A\backslash b$) are multi-threaded
2. the Octave toolboxes do not mimic precisely Matlab ones
3. generally slower than Matlab
4. some differences Octave /Matlab :
   - random generators initialization
   - ode solvers are not natives (use Odepkg to have full compatibility)
   - do not support non-portable extensions like CUDA, Web, ActiveX, etc...
   - do not support handles on embedded functions
5. do not forget to launch Octave with "`--traditional`'' or ``--braindead''

# Using Python as MATLAB alternative

1. Using Python is less compatible with MATLAB than Octave

# Using Python as MATLAB alternative

1. Using Python is less compatible with MATLAB than Octave
2. Python is a "true" language:

# Using Python as MATLAB alternative

1. Using Python is less compatible with MATLAB than Octave
2. Python is a "true" language:
   - imperative concepts (references, modules), exceptions

# Using Python as MATLAB alternative

1. Using Python is less compatible with MATLAB than Octave
2. Python is a "true" language:
   - imperative concepts (references, modules), exceptions
   - object-oriented (inheritance, encapsulation)

# Using Python as MATLAB alternative

1. Using Python is less compatible with MATLAB than Octave
2. Python is a "true" language:
   - imperative concepts (references, modules), exceptions
   - object-oriented (inheritance, encapsulation)
   - some functional aspects (map, reduce, filter, lambda functions)

# Using Python as MATLAB alternative

1. Using Python is less compatible with MATLAB than Octave
2. Python is a "true" language:
   - imperative concepts (references, modules), exceptions
   - object-oriented (inheritance, encapsulation)
   - some functional aspects (map, reduce, filter, lambda functions)
   - foundations for clean development: integrated unit tests and documentation

# Using Python as MATLAB alternative

1. Using Python is less compatible with MATLAB than Octave
2. Python is a "true" language:
   - imperative concepts (references, modules), exceptions
   - object-oriented (inheritance, encapsulation)
   - some functional aspects (map, reduce, filter, lambda functions)
   - foundations for clean development: integrated unit tests and documentation
3. there are lots of libraries and packages for Python

# Using Python as MATLAB alternative

1. Using Python is less compatible with MATLAB than Octave
2. Python is a "true" language:
   - imperative concepts (references, modules), exceptions
   - object-oriented (inheritance, encapsulation)
   - some functional aspects (map, reduce, filter, lambda functions)
   - foundations for clean development: integrated unit tests and documentation
3. there are lots of libraries and packages for Python
4. Python has a big community of users

# Using Python as MATLAB alternative

1. Using Python is less compatible with MATLAB than Octave
2. Python is a "true" language:
   - imperative concepts (references, modules), exceptions
   - object-oriented (inheritance, encapsulation)
   - some functional aspects (map, reduce, filter, lambda functions)
   - foundations for clean development: integrated unit tests and documentation
3. there are lots of libraries and packages for Python
4. Python has a big community of users
5. Very easy to write extensions trough Cython, SWIG, C API or f2py

# For using Python as Matlab you need...

1. Python interpreter: CPython or PyPy

# For using Python as Matlab you need...

1. Python interpreter: CPython or PyPy
2. Numpy: general numeric array module (equivalent of BLAS)

# For using Python as Matlab you need...

1. Python interpreter: CPython or PyPy
2. Numpy: general numeric array module (equivalent of BLAS)
3. Scipy: general scientific package (base on Numpy), ODE solvers, dense & sparse linear algebra, numerical integration

# For using Python as Matlab you need...

1. Python interpreter: CPython or PyPy
2. Numpy: general numeric array module (equivalent of BLAS)
3. Scipy: general scientific package (base on Numpy), ODE solvers, dense & sparse linear algebra, numerical integration
4. Matplotlib: for graphics

# For using Python as Matlab you need...

1. Python interpreter: CPython or PyPy
2. Numpy: general numeric array module (equivalent of BLAS)
3. Scipy: general scientific package (base on Numpy), ODE solvers, dense & sparse linear algebra, numerical integration
4. Matplotlib: for graphics
5. IPython(or IPython-wx): nice command-line interpreter (with completion, help functions)

# For using Python as Matlab you need...

1. Python interpreter: CPython or PyPy
2. Numpy: general numeric array module (equivalent of BLAS)
3. Scipy: general scientific package (base on Numpy), ODE solvers, dense & sparse linear algebra, numerical integration
4. Matplotlib: for graphics
5. IPython(or IPython-wx): nice command-line interpreter (with completion, help functions)
6. Spyder(Linux) or Python$(x, y)$ (Windows): beautiful IDEs, for people who are nostalgic of Matlab Gui

# For using Python as Matlab you need...

1. Python interpreter: CPython or PyPy
2. Numpy: general numeric array module (equivalent of BLAS)
3. Scipy: general scientific package (base on Numpy), ODE solvers, dense & sparse linear algebra, numerical integration
4. Matplotlib: for graphics
5. IPython(or IPython-wx): nice command-line interpreter (with completion, help functions)
6. Spyder(Linux) or Python$(x, y)$ (Windows): beautiful IDEs, for people who are nostalgic of Matlab Gui

# For using Python as Matlab you need...

1. Python interpreter: CPython or PyPy
2. Numpy: general numeric array module (equivalent of BLAS)
3. Scipy: general scientific package (base on Numpy), ODE solvers, dense & sparse linear algebra, numerical integration
4. Matplotlib: for graphics
5. IPython(or IPython-wx): nice command-line interpreter (with completion, help functions)
6. Spyder(Linux) or Python$(x, y)$ (Windows): beautiful IDEs, for people who are nostalgic of Matlab Gui

....en voiture Simone!

# Structural differences Matlab /Numpy

1. object-oriented syntax

   `a=zeros(3,3); m=size(a,2)` $\Leftrightarrow$ `a=zeros((3,3)); m=a.shape[1]`

# Structural differences Matlab /Numpy

1. object-oriented syntax

   `a=zeros(3,3); m=size(a,2)` $\Leftrightarrow$ `a=zeros((3,3)); m=a.shape[1]`

2. starting zero indexing and range do not contain last element

   `b = a(:,1:2)` $\Leftrightarrow$ `b = a[:,0:2]`

# Structural differences Matlab /Numpy

1. object-oriented syntax

   `a=zeros(3,3); m=size(a,2)` $\Leftrightarrow$ `a=zeros((3,3)); m=a.shape[1]`

2. starting zero indexing and range do not contain last element

   `b = a(:,1:2)` $\Leftrightarrow$ `b = a[:,0:2]`

3. Python always use references

   `a=(1:3)'*(1:3); b=a; b(:,1)=zeros(1,3); a(1,:)` $\Rightarrow$ `ans = 1 2 3`

   **but** in Python `a=arange(1,4)[:,None]*arange(1,4)[None,:]; b=a;`

   `b[0,:]=zeros((1,3)); a[0,:]` $\Rightarrow$ `array([0, 0, 0])`

   To obtain the same behavior use `b=a.copy()`

# Structural differences Matlab /Numpy

1. object-oriented syntax

   `a=zeros(3,3); m=size(a,2)` $\Leftrightarrow$ `a=zeros((3,3)); m=a.shape[1]`

2. starting zero indexing and range do not contain last element

   `b = a(:,1:2)` $\Leftrightarrow$ `b = a[:,0:2]`

3. Python always use references

   `a=(1:3)'*(1:3); b=a; b(:,1)=zeros(1,3); a(1,:)` $\Rightarrow$ `ans = 1 2 3`

   **but** in Python `a=arange(1,4)[:,None]*arange(1,4)[None,:]; b=a;`

   `b[0,:]=zeros((1,3)); a[0,:]` $\Rightarrow$ `array([0, 0, 0])`

   To obtain the same behavior use `b=a.copy()`

4. 1D arrays have one dimension in Python

   `length(size(1:3))` $\Rightarrow$ 2 but `len(shape(r_[1:4]))` $\Rightarrow$ 1

# Structural differences Matlab /Numpy

1. object-oriented syntax

   `a=zeros(3,3); m=size(a,2)` $\Leftrightarrow$ `a=zeros((3,3)); m=a.shape[1]`

2. starting zero indexing and range do not contain last element

   `b = a(:,1:2)` $\Leftrightarrow$ `b = a[:,0:2]`

3. Python always use references

   `a=(1:3)'*(1:3); b=a; b(:,1)=zeros(1,3); a(1,:)` $\Rightarrow$ `ans = 1 2 3`

   **but** in Python `a=arange(1,4)[:,None]*arange(1,4)[None,:]; b=a;`

   `b[0,:]=zeros((1,3)); a[0,:]` $\Rightarrow$ `array([0, 0, 0])`

   To obtain the same behavior use `b=a.copy()`

4. 1D arrays have one dimension in Python

   `length(size(1:3))` $\Rightarrow$ 2 but `len(shape(r_[1:4]))` $\Rightarrow$ 1

5. all operations all by default component-wise:

   `c=a.*b` $\Leftrightarrow$ `c=a*b` but `c=a*b` $\Leftrightarrow$ `c=dot(a,b)`

# some words about functions

1. functions are not attached to files: their are true objects of the language

# some words about functions

1. functions are not attached to files: their are true objects of the language
2. arguments are always passed by references

# some words about functions

1. functions are not attached to files: their are true objects of the language
2. arguments are always passed by references
3. you can use `lambda` to define anonymous functions: e.g.

```
carre =(lambda x: x*x)
```

# some words about functions

1. functions are not attached to files: their are true objects of the language
2. arguments are always passed by references
3. you can use `lambda` to define anonymous functions: e.g.

   ```
   carre =(lambda x: x*x)
   ```
4. you can define arguments by default:
   `def mafonc(x, y = 2, z = 3): .` All the following calls are valid

# some words about functions

1. functions are not attached to files: their are true objects of the language
2. arguments are always passed by references
3. you can use `lambda` to define anonymous functions: e.g.

   ```
   carre =(lambda x: x*x)
   ```
4. you can define arguments by default:
   `def mafonc(x, y = 2, z = 3):` . All the following calls are valid
   - `mafonc(5)`: calling mafonc(5,2,3)

# some words about functions

1. functions are not attached to files: their are true objects of the language
2. arguments are always passed by references
3. you can use `lambda` to define anonymous functions: e.g.

   ```
   carre =(lambda x: x*x)
   ```
4. you can define arguments by default:
   `def mafonc(x, y = 2, z = 3):` . All the following calls are valid
   - `mafonc(5)`: calling mafonc(5,2,3)
   - `mafonc(5, -1)`: calling mafonc(5,-1,3)

# some words about functions

1. functions are not attached to files: their are true objects of the language
2. arguments are always passed by references
3. you can use `lambda` to define anonymous functions: e.g.

   ```
   carre =(lambda x: x*x)
   ```
4. you can define arguments by default:
   `def mafonc(x, y = 2, z = 3):` . All the following calls are valid
   - `mafonc(5)`: calling mafonc(5,2,3)
   - `mafonc(5, -1)`: calling mafonc(5,-1,3)
   - `mafonc(5, z=-1)`: calling mafonc(5,2,-1)

# some words about functions

1. functions are not attached to files: their are true objects of the language
2. arguments are always passed by references
3. you can use `lambda` to define anonymous functions: e.g.

   ```
   carre =(lambda x: x*x)
   ```
4. you can define arguments by default:
   `def mafonc(x, y = 2, z = 3):` . All the following calls are valid
   - `mafonc(5)`: calling mafonc(5,2,3)
   - `mafonc(5, -1)`: calling mafonc(5,-1,3)
   - `mafonc(5, z=-1)`: calling mafonc(5,2,-1)
   - `mafonc(z=-1, x=5)` calling mafonc(5,2,-1)

# Survival kit for Matlab user

| Matlab /Octave | Python |
|---|---|
| `size(a)` | `a.shape` |
| `a'` | `a.conj().T` or `a.conj().transpose()` |
| `a(:)` | `a.flatten('F')` or `a.T.flatten('C').` |
| `b=reshape(a,[2,8])` | `b=a.reshape(2,8)` or `b=a.copy(); b.shape=(2,8)` |
| `V(:,end)` | `V[:,-1]` or `V[:,-1:]` |
| `[a,b]` | `c_[a,b]` or `hstack((a,b))` |
| `[a;b]` | `r_[a,b]` or `vstack((a,b))` |
| `(1:p:n)` | `arange(1,n+1,p)` or `r_[1:n+1:p]` |
| `find(a>5)` and `a(a>5)` | `nonzero(a>5)[0]` and `a[a>5]` |
| `x= V\b` | `x=linalg.solve(V,b)` if V is square |

# digression: broadcasting

- Q: what does Python when applying an element-wise operator to two matrices of different sizes ?

# digression: broadcasting

- Q: what does Python when applying an element-wise operator to two matrices of different sizes ?
- A: comparing component-wisely the dimensions: if these are equal there is no problem. If there are different and one equals to one, Python duplicates that array along that dimension to equal the other.

# digression: broadcasting

- Q: what does Python when applying an element-wise operator to two matrices of different sizes ?
- A: comparing component-wisely the dimensions: if these are equal there is no problem. If there are different and one equals to one, Python duplicates that array along that dimension to equal the other.
- example:

$$\begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} / \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{a}{u} & \frac{b}{u} & \frac{c}{u} \\ \frac{d}{v} & \frac{e}{v} & \frac{f}{v} \end{pmatrix}$$

# digression: broadcasting

- Q: what does Python when applying an element-wise operator to two matrices of different sizes ?
- A: comparing component-wisely the dimensions: if these are equal there is no problem. If there are different and one equals to one, Python duplicates that array along that dimension to equal the other.
- example:

$$\begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix} / \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{a}{u} & \frac{b}{u} & \frac{c}{u} \\ \frac{d}{v} & \frac{e}{v} & \frac{f}{v} \end{pmatrix}$$

- since version 3.6, Octave supports automatic broadcasting for element-wise operations

# Example: generate a Vandermonde

- We want to generate the following matrix

$$V(x_0, \cdots, x_n) = \begin{pmatrix} 1 & \dots & 1 & \dots & 1 \\ \vdots & & \vdots & & \vdots \\ x_0^{i-1} & \dots & x_{j-1}^{i-1} & \dots & x_n^{i-1} \\ \vdots & & \vdots & & \vdots \\ x_0^n & \dots & x_{j-1}^n & \dots & x_n^n \end{pmatrix}$$

for $n = 4$ and $x = (1, 2, 3, 4)$

# Example: generate a Vandermonde

- We want to generate the following matrix

$$V(x_0, \cdots, x_n) = \begin{pmatrix} 1 & \dots & 1 & \dots & 1 \\ \vdots & & \vdots & & \vdots \\ x_0^{i-1} & \dots & x_{j-1}^{i-1} & \dots & x_n^{i-1} \\ \vdots & & \vdots & & \vdots \\ x_0^n & \dots & x_{j-1}^n & \dots & x_n^n \end{pmatrix}$$

  for $n = 4$ and $x = (1, 2, 3, 4)$

- in Malab : `V=bsxfun(@power,(1:4),(0:3)')`

# Example: generate a Vandermonde

- We want to generate the following matrix

$$V(x_0, \cdots, x_n) = \begin{pmatrix} 1 & \dots & 1 & \dots & 1 \\ \vdots & & \vdots & & \vdots \\ x_0^{i-1} & \dots & x_{j-1}^{i-1} & \dots & x_n^{i-1} \\ \vdots & & \vdots & & \vdots \\ x_0^n & \dots & x_{j-1}^n & \dots & x_n^n \end{pmatrix}$$

  for $n = 4$ and $x = (1, 2, 3, 4)$
- in Malab : `V=bsxfun(@power,(1:4),(0:3)')`
- in Python: `V=r_[1:5][None,:]**r_[:4][:,None]`

# Example: generate a Vandermonde

- We want to generate the following matrix

$$V(x_0, \cdots, x_n) = \begin{pmatrix} 1 & \dots & 1 & \dots & 1 \\ \vdots & & \vdots & & \vdots \\ x_0^{i-1} & \dots & x_{j-1}^{i-1} & \dots & x_n^{i-1} \\ \vdots & & \vdots & & \vdots \\ x_0^n & \dots & x_{j-1}^n & \dots & x_n^n \end{pmatrix}$$

  for $n = 4$ and $x = (1, 2, 3, 4)$

- in Matlab : `V=bsxfun(@power,(1:4),(0:3)')`
- in Python: `V=r_[1:5][None,:]**r_[:4][:,None]`
- in Octave (version $\geqslant 3.6$): `V=(1:4)'.^(0:3)'`

# Example: generate a Vandermonde

- We want to generate the following matrix

$$V(x_0, \cdots, x_n) = \begin{pmatrix} 1 & \dots & 1 & \dots & 1 \\ \vdots & & \vdots & & \vdots \\ x_0^{i-1} & \dots & x_{j-1}^{i-1} & \dots & x_n^{i-1} \\ \vdots & & \vdots & & \vdots \\ x_0^n & \dots & x_{j-1}^n & \dots & x_n^n \end{pmatrix}$$

  for $n = 4$ and $x = (1, 2, 3, 4)$

- in Malab : `V=bsxfun(@power,(1:4),(0:3)')`
- in Python: `V=r_[1:5][None,:]**r_[:4][:,None]`
- in Octave (version $\geqslant 3.6$): `V=(1:4)'.^(0:3)'`
- in Julia: `[ i^j for j=0:3,i=1:4]`

# Scipy: Outline

Scipy is a high-Level Python package for scientific computing; it is based on Numpy and provides:

- dense & sparse linear algebra

# Scipy: Outline

Scipy is a high-Level Python package for scientific computing; it is based on Numpy and provides:

- dense & sparse linear algebra
- interpolation

# Scipy: Outline

Scipy is a high-Level Python package for scientific computing; it is based on Numpy and provides:

- dense & sparse linear algebra
- interpolation
- numerical integration

# Scipy: Outline

Scipy is a high-Level Python package for scientific computing; it is based on Numpy and provides:

- dense & sparse linear algebra
- interpolation
- numerical integration
- ODE solvers

# Scipy: Outline

Scipy is a high-Level Python package for scientific computing; it is based on Numpy and provides:

- dense & sparse linear algebra
- interpolation
- numerical integration
- ODE solvers
- eigenvalues problems

# Scipy: Outline

Scipy is a high-Level Python package for scientific computing; it is based on Numpy and provides:

- dense & sparse linear algebra
- interpolation
- numerical integration
- ODE solvers
- eigenvalues problems
- Fourier Transforms

# Scipy: Outline

Scipy is a high-Level Python package for scientific computing; it is based on Numpy and provides:

- dense & sparse linear algebra
- interpolation
- numerical integration
- ODE solvers
- eigenvalues problems
- Fourier Transforms
- special functions

# Sparse matrices

- MATLAB and Octave have *one* internal format for sparse matrices: the Compressed Sparse Column (CSC) format

# Sparse matrices

- Matlab and Octave have *one* internal format for sparse matrices: the Compressed Sparse Column (CSC) format
- in Scipy: there are 7 sparse formats:

# Sparse matrices

- Matlab and Octave have *one* internal format for sparse matrices: the Compressed Sparse Column (CSC) format
- in Scipy: there are 7 sparse formats:
  - csr, csc and bsr $\rightarrow$ good for computing

# Sparse matrices

- Matlab and Octave have *one* internal format for sparse matrices: the Compressed Sparse Column (CSC) format
- in Scipy: there are 7 sparse formats:
  - csr, csc and bsr $\rightarrow$ good for computing
  - lil_matrix: support slicing and good for building

# Sparse matrices

- Matlab and Octave have *one* internal format for sparse matrices: the Compressed Sparse Column (CSC) format
- in Scipy: there are 7 sparse formats:
  - csr, csc and bsr $\rightarrow$ good for computing
  - lil_matrix: support slicing and good for building
  - coo: good for FE modeling (aka the $(I, J, V)$ format)

# Sparse matrices

- Matlab and Octave have *one* internal format for sparse matrices: the Compressed Sparse Column (CSC) format
- in Scipy: there are 7 sparse formats:
  - csr, csc and bsr $\rightarrow$ good for computing
  - lil_matrix: support slicing and good for building
  - coo: good for FE modeling (aka the $(I, J, V)$ format)
  - dia: efficient for computing if data are near of the diagonal

# Sparse matrices

- Matlab and Octave have *one* internal format for sparse matrices: the Compressed Sparse Column (CSC) format
- in Scipy: there are 7 sparse formats:
  - csr, csc and bsr $\rightarrow$ good for computing
  - lil_matrix: support slicing and good for building
  - coo: good for FE modeling (aka the $(I, J, V)$ format)
  - dia: efficient for computing if data are near of the diagonal
  - dok (dictionary of keys): good for incremental construction

# Sparse matrices

- MATLAB and Octave have *one* internal format for sparse matrices: the Compressed Sparse Column (CSC) format
- in Scipy: there are 7 sparse formats:
  - csr, csc and bsr $\rightarrow$ good for computing
  - lil_matrix: support slicing and good for building
  - coo: good for FE modeling (aka the $(I, J, V)$ format)
  - dia: efficient for computing if data are near of the diagonal
  - dok (dictionary of keys): good for incremental construction
- ☹ do not use any parallelism, but neither does MATLAB (only PyTrilinos)

# Matplotlib: basic 2D plots...

- plot $x \mapsto sin(x)$ on $[0, 2\pi]$: launch `ipython --pylab`

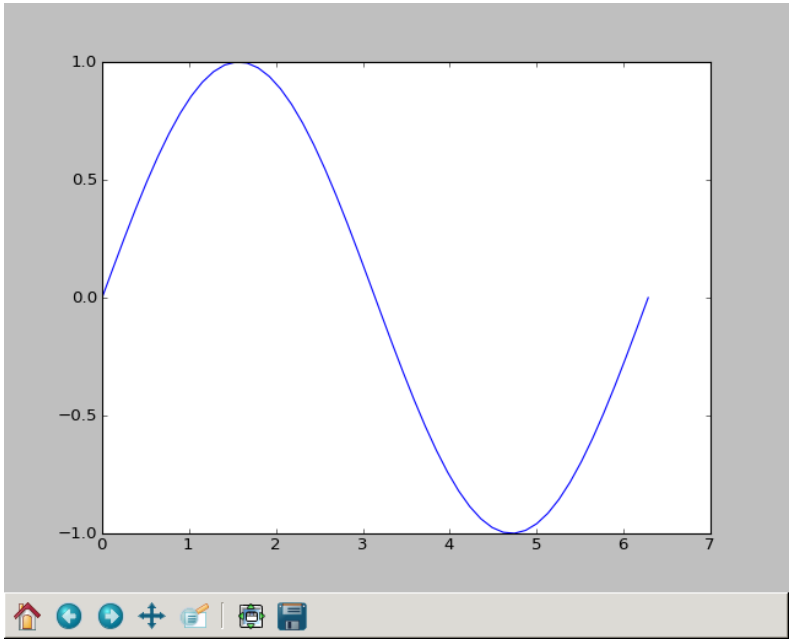
```
from pylab import *
x=linspace(0,2*pi)
plot(x,sin(x))
```

# Matplotlib: basic 2D plots...

- plot $x \mapsto sin(x)$ on $[0, 2\pi]$: launch `ipython --pylab`

```
from pylab import *
x=linspace(0,2*pi)
plot(x,sin(x))
```

- we obtain the following window



# ...and 3D plots

- we want to plot $(x, y) \mapsto \frac{\sin\sqrt{x^2+y^2}}{\sqrt{x^2+y^2}}$ on $[-2\pi, 2\pi]^2$

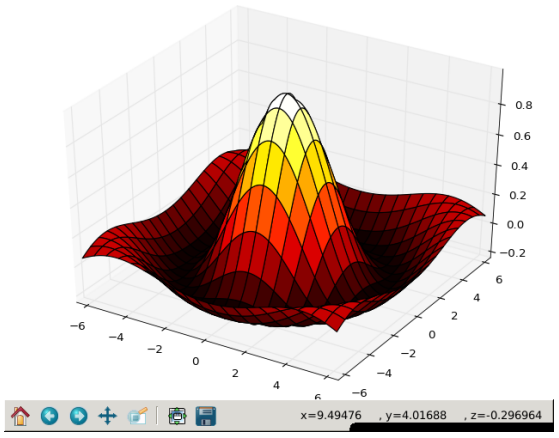
```
from pylab import *
from mpl_toolkits.mplot3d import Axes3D
XX,YY=mgrid[-2*pi:2.*pi:200j,-2*pi:2.*pi:200j]
fig = figure()
ax = Axes3D(fig)
N=(sqrt(XX**2+YY**2))
ax.plot_surface(XX, YY, sin(N)/N,cmap=cm.hot)
show()
```

# ...and 3D plots

- we want to plot $(x, y) \mapsto \frac{\sin\sqrt{x^2+y^2}}{\sqrt{x^2+y^2}}$ on $[-2\pi, 2\pi]^2$

```
from pylab import *
from mpl_toolkits.mplot3d import Axes3D
XX,YY=mgrid[-2*pi:2.*pi:200j,-2*pi:2.*pi:200j]
fig = figure()
ax = Axes3D(fig)
N=(sqrt(XX**2+YY**2))
ax.plot_surface(XX, YY, sin(N)/N,cmap=cm.hot)
show()
```

- we obtain the following window



# Mayavi: parametrized surface

- visualize a surface $M = \{x(\phi, \theta) \in \mathbb{R}^3 \mid (\theta, \phi) \in I \times J\}$

```
from numpy import *
from mayavi.mlab import *

dphi, dtheta = pi/250.0, pi/250.0
[phi,theta] = mgrid[0:pi+dphi*1.5:dphi,
                    0:2*pi+dtheta*1.5:dtheta]
m0=4; m1=3; m2=2; m3=3; m4=6; m5=2; m6=6; m7=4;
r = sin(m0*phi)**m1 + cos(m2*phi)**m3 \
  + sin(m4*theta)**m5 + cos(m6*theta)**m7
x = r*sin(phi)*cos(theta)
y = r*cos(phi)
z = r*sin(phi)*sin(theta)
mesh(x, y, z)
```

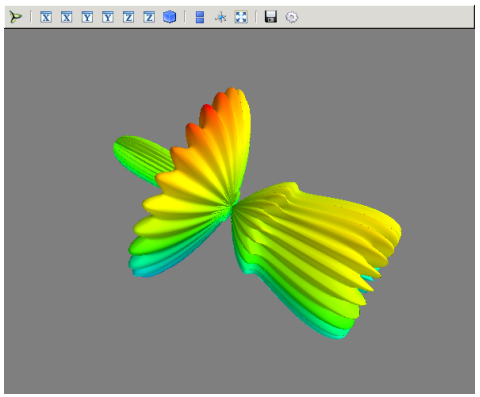
# Mayavi: parametrized surface

- visualize a surface $M = \{x(\phi, \theta) \in \mathbb{R}^3 \mid (\theta, \phi) \in I \times J\}$

```
from numpy import *
from mayavi.mlab import *

dphi, dtheta = pi/250.0, pi/250.0
[phi,theta] = mgrid[0:pi+dphi*1.5:dphi,
                    0:2*pi+dtheta*1.5:dtheta]
m0=4; m1=3; m2=2; m3=3; m4=6; m5=2; m6=6; m7=4;
r = sin(m0*phi)**m1 + cos(m2*phi)**m3 \
  + sin(m4*theta)**m5 + cos(m6*theta)**m7
x = r*sin(phi)*cos(theta)
y = r*cos(phi)
z = r*sin(phi)*sin(theta)
mesh(x, y, z)
```

- result



# Mayavi: quiver

- to visualize a vector field:

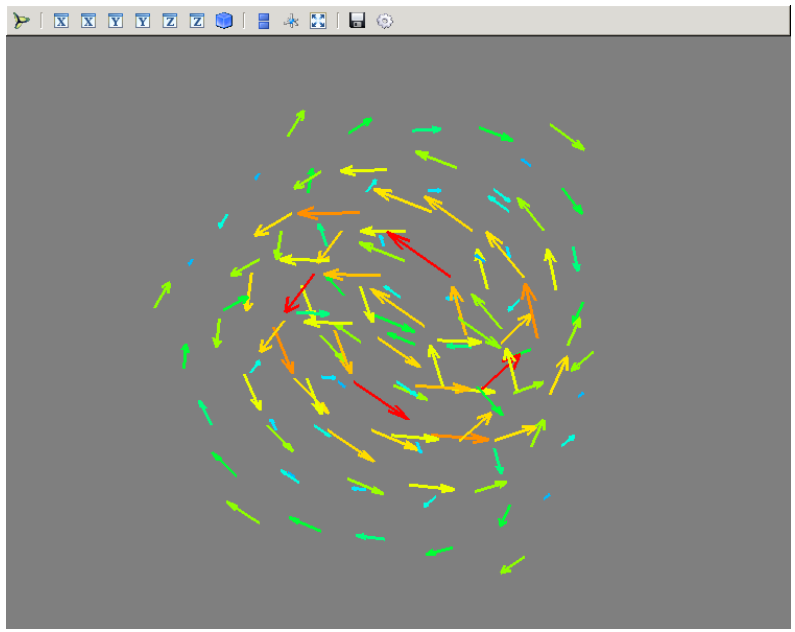
```
import numpy
from mayavi.mlab import *
x, y, z = numpy.mgrid[-2:3, -2:3, -2:3]
r = numpy.sqrt(x**2 + y**2 + z**4)
u = y*numpy.sin(r)/(r+0.001)
v = -x*numpy.sin(r)/(r+0.001)
w = numpy.zeros_like(z)
quiver3d(x, y, z, u, v, w, line_width=3, scale_factor=1)
```

# Mayavi: quiver

- to visualize a vector field:

```
import numpy
from mayavi.mlab import *
x, y, z = numpy.mgrid[-2:3, -2:3, -2:3]
r = numpy.sqrt(x**2 + y**2 + z**4)
u = y*numpy.sin(r)/(r+0.001)
v = -x*numpy.sin(r)/(r+0.001)
w = numpy.zeros_like(z)
quiver3d(x, y, z, u, v, w, line_width=3, scale_factor=1)
```

- result:



# About parallelism

- shared-memory parallelism: due to the Global Lock System(GIL), Python supports badly the multi-threading paradigm. There is a multiprocessing toolbox $\Rightarrow$ not very handy

# About parallelism

- shared-memory parallelism: due to the Global Lock System(GIL), Python supports badly the multi-threading paradigm. There is a multiprocessing toolbox $\Rightarrow$ not very handy
- distributed parallelism:

# About parallelism

- shared-memory parallelism: due to the Global Lock System(GIL), Python supports badly the multi-threading paradigm. There is a multiprocessing toolbox $\Rightarrow$ not very handy
- distributed parallelism:
  - MPI4py: if you know MPI, wrapper to classical MPI calls, with an integration of Numpy(☺)

# About parallelism

- shared-memory parallelism: due to the Global Lock System(GIL), Python supports badly the multi-threading paradigm. There is a multiprocessing toolbox $\Rightarrow$ not very handy
- distributed parallelism:
  - MPI4py: if you know MPI, wrapper to classical MPI calls, with an integration of Numpy(☺)
  - PyTrilinos:

# About parallelism

- shared-memory parallelism: due to the Global Lock System(GIL), Python supports badly the multi-threading paradigm. There is a multiprocessing toolbox $\Rightarrow$ not very handy
- distributed parallelism:
  - MPI4py: if you know MPI, wrapper to classical MPI calls, with an integration of Numpy(☺)
  - PyTrilinos:
    - ☺ do not necessitate to know MPI

# About parallelism

- shared-memory parallelism: due to the Global Lock System(GIL), Python supports badly the multi-threading paradigm. There is a multiprocessing toolbox $\Rightarrow$ not very handy
- distributed parallelism:
  - MPI4py: if you know MPI, wrapper to classical MPI calls, with an integration of Numpy(☺)
  - PyTrilinos:
    - ☺ do not necessitate to know MPI
    - ☺ support sparse matrices

# About parallelism

- shared-memory parallelism: due to the Global Lock System(GIL), Python supports badly the multi-threading paradigm. There is a multiprocessing toolbox $\Rightarrow$ not very handy
- distributed parallelism:
  - MPI4py: if you know MPI, wrapper to classical MPI calls, with an integration of Numpy(☺)
  - PyTrilinos:
    - ☺ do not necessitate to know MPI
    - ☺ support sparse matrices
    - ☹ "Rube Goldberg machine"

# MPI4py: sum reduction

```
import numpy as np
N=8*2*10**8
tab=np.r_[:N]
s=tab.sum()
print s
```

time -p python somme_seq.py

$\Rightarrow$

1279999999200000000

real 8.94

user 4.68

sys 4.23

```
from mpi4py import MPI
from numpy import *
comm = MPI.COMM_WORLD
size = comm.size
rank = comm.rank
paquet=2*10**8
N=paquet*size
tab_loc=r_[rank*paquet:
           (rank+1)*paquet]
sloc=tab_loc.sum()
stot = 0
#on reduit
stot=comm.reduce(sloc,stot,
MPI.SUM,0)
if (comm.rank == 0):
  print stot
```

time -p mpirun -np 8 python somme_par.py

$\Rightarrow$ 1279999999200000000

real 2.71

user 7.30

sys 5.69

# Some benchmarks

- on a Z800 (8 threads, 24G of memory),

# Some benchmarks

- on a Z800 (8 threads, 24G of memory),
- all times are in milliseconds

# Some benchmarks

- on a Z800 (8 threads, 24G of memory),
- all times are in milliseconds
- benchmarks sources taken from the Julia site and modified

# Some benchmarks

- on a Z800 (8 threads, 24G of memory),
- all times are in milliseconds
- benchmarks sources taken from the Julia site and modified

# Some benchmarks

- on a Z800 (8 threads, 24G of memory),
- all times are in milliseconds
- benchmarks sources taken from the Julia site and modified

| appli | fib | mandel | quicksort | pisum | randstat | randmul |
|---|---|---|---|---|---|---|
| Malab | 289 | 33 | 45 | 89 | 142 | 67 |
| Octave | 924 | 310 | 1138 | 21159 | 484 | 109 |
| Python | 4 | 6 | 12 | 1073 | 248 | 107 |
| Julia | 0.32 | 0.34 | 0.68 | 45 | 27 | 45 |
| Fortran | 0.08 | $\leqslant 10^{-6}$ | 0.62 | 44 | 16 | 275(16) |

# As a conclusion

- If you need some compatibility: uses Octave

# As a conclusion

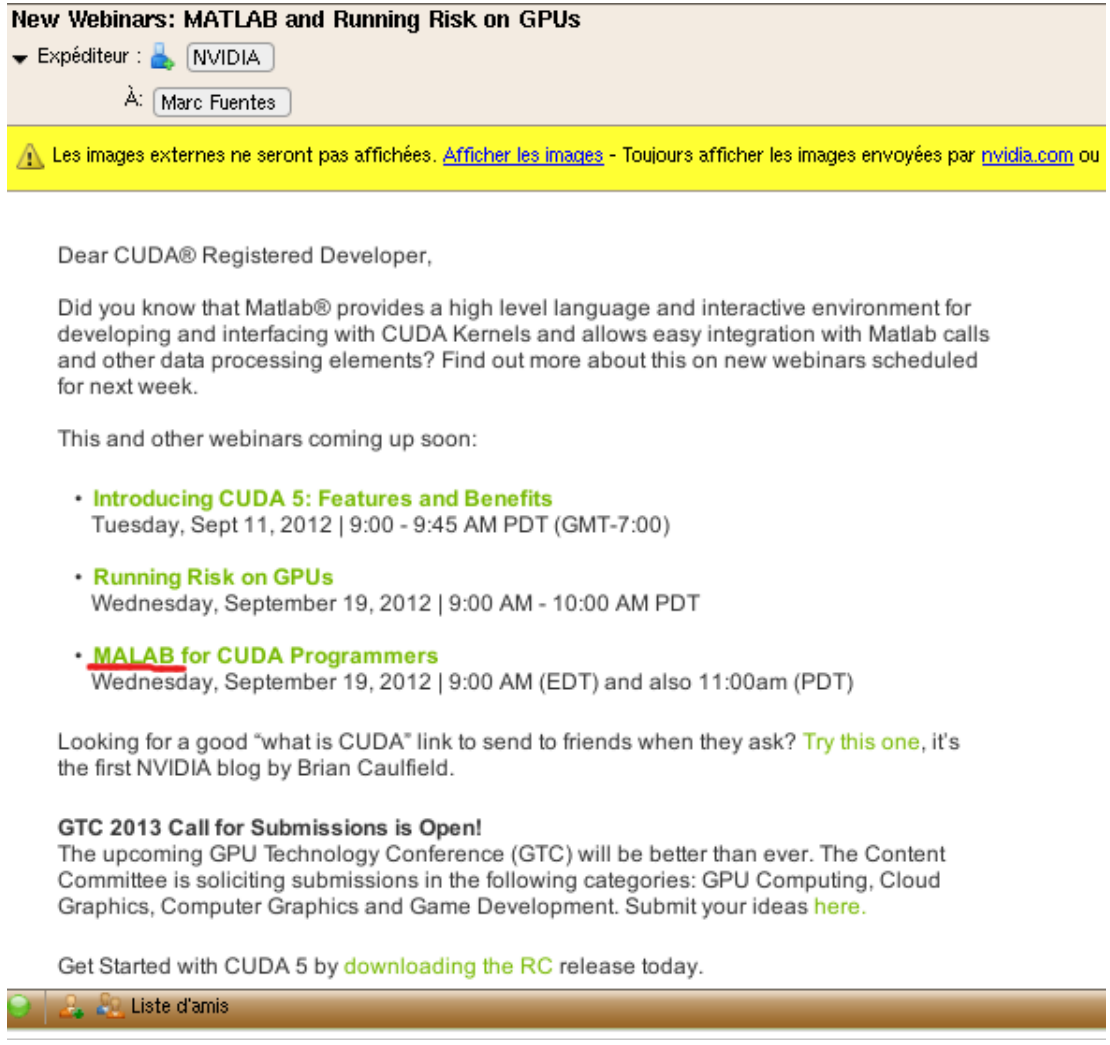
- If you need some compatibility: uses Octave
- When you start a new project in Matlab , think to develop in parallel, the Octave version (necessitate a few work in addition)

# As a conclusion

- If you need some compatibility: uses Octave
- When you start a new project in Matlab , think to develop in parallel, the Octave version (necessitate a few work in addition)
- If you are ready "to take the plunge": throw Matlab out, and go on with Python

# Why Malab?

- because



**New Webinars: MATLAB and Running Risk on GPUs**

Expéditeur : NVIDIA

À: Marc Fuentes

Les images externes ne seront pas affichées. Afficher les images - Toujours afficher les images envoyées par nvidia.com ou

Dear CUDA® Registered Developer,

Did you know that Matlab® provides a high level language and interactive environment for developing and interfacing with CUDA Kernels and allows easy integration with Matlab calls and other data processing elements? Find out more about this on new webinars scheduled for next week.

This and other webinars coming up soon:

- **Introducing CUDA 5: Features and Benefits**
  Tuesday, Sept 11, 2012 | 9:00 - 9:45 AM PDT (GMT-7:00)
- **Running Risk on GPUs**
  Wednesday, September 19, 2012 | 9:00 AM - 10:00 AM PDT
- **MALAB for CUDA Programmers**
  Wednesday, September 19, 2012 | 9:00 AM (EDT) and also 11:00am (PDT)

Looking for a good "what is CUDA" link to send to friends when they ask? Try this one, it's the first NVIDIA blog by Brian Caulfield.

**GTC 2013 Call for Submissions is Open!**
The upcoming GPU Technology Conference (GTC) will be better than ever. The Content Committee is soliciting submissions in the following categories: GPU Computing, Cloud Graphics, Computer Graphics and Game Development. Submit your ideas here.

Get Started with CUDA 5 by downloading the RC release today.

Liste d'amis