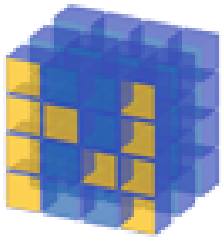
Python as a scientific tool for analysis and simulation

Numpy, Scipy and more

U. Barkan[1]

[1]Tel-Aviv University

Haifux, May 2012



python™

Outline

1. Introduction
   - Where am I coming from
   - Where does Python come from
   - What is Python
   - What you need
2. Comparison with Matlab
   - Inital Comparison
   - Getting help
   - Advanced examples
3. Advantages
   - In favor of Matlab
   - In favor of Python
4. Demo
   - What is RANSAC?
   - Demonstration
5. Summary

Outline

1. Introduction
   - Where am I coming from
   - Where does Python come from
   - What is Python
   - What you need
2. Comparison with Matlab
   - Inital Comparison
   - Getting help
   - Advanced examples
3. Advantages
   - In favor of Matlab
   - In favor of Python
4. Demo
   - What is RANSAC?
   - Demonstration
5. Summary

Where am I coming from

- Matlab$^{(TM)}$/Octave
  - 1997 - current
  - Applications, Algorithms, R&D
  - Simulink, toolboxes (DSP, Neural Networks)
- C
  - 2000 - 2002 (academic), 2007-2008 (work)
  - Mainly Applications
  - Large-scale Monte-Carlo simulations
  - Financial data analysis
- Scilab
  - 2007-2010
  - Signal Processing algorithms design
  - Limited Experience
- Python
  - 2008 - current
  - Algorithms, analysis, simulations - if it's R&D, it's in Python
- I'm not selling, so you don't have to buy

Where am I coming from

- Matlab$^{(TM)}$/Octave
  - 1997 - current
  - Applications, Algorithms, R&D
  - Simulink, toolboxes (DSP, Neural Networks)
- C
  - 2000 - 2002 (academic), 2007-2008 (work)
  - Mainly Applications
  - Large-scale Monte-Carlo simulations
  - Financial data analysis
- Scilab
  - 2007-2010
  - Signal Processing algorithms design
  - Limited Experience
- Python
  - 2008 - current
  - Algorithms, analysis, simulations - if it's R&D, it's in Python
- I'm not selling, so you don't have to buy

Where am I coming from

- Matlab$^{(TM)}$/Octave
  - 1997 - current
  - Applications, Algorithms, R&D
  - Simulink, toolboxes (DSP, Neural Networks)
- C
  - 2000 - 2002 (academic), 2007-2008 (work)
  - Mainly Applications
  - Large-scale Monte-Carlo simulations
  - Financial data analysis
- Scilab
  - 2007-2010
  - Signal Processing algorithms design
  - Limited Experience
- Python
  - 2008 - current
  - Algorithms, analysis, simulations - if it's R&D, it's in Python
- I'm not selling, so you don't have to buy

Where am I coming from

- Matlab$^{(TM)}$/Octave
  - 1997 - current
  - Applications, Algorithms, R&D
  - Simulink, toolboxes (DSP, Neural Networks)
- C
  - 2000 - 2002 (academic), 2007-2008 (work)
  - Mainly Applications
  - Large-scale Monte-Carlo simulations
  - Financial data analysis
- Scilab
  - 2007-2010
  - Signal Processing algorithms design
  - Limited Experience
- Python
  - 2008 - current
  - Algorithms, analysis, simulations - if it's R&D, it's in Python
- I'm not selling, so you don't have to buy

Where am I coming from

- Matlab$^{(TM)}$/Octave
  - 1997 - current
  - Applications, Algorithms, R&D
  - Simulink, toolboxes (DSP, Neural Networks)
- C
  - 2000 - 2002 (academic), 2007-2008 (work)
  - Mainly Applications
  - Large-scale Monte-Carlo simulations
  - Financial data analysis
- Scilab
  - 2007-2010
  - Signal Processing algorithms design
  - Limited Experience
- Python
  - 2008 - current
  - Algorithms, analysis, simulations - if it's R&D, it's in Python
- I'm not selling, so you don't have to buy

Brief history of Python

Guido van-Rossum

What is Python

- Flexible, powerful language
- Multiple programming paradigms
- Easy, clean syntax
- "Batteries included"
- Free as in "free speech" AND as in "free beer"!
- Large community of support

Things you may have thought about Python

And you were right

- Python's syntax is different than Matlab's
- There is no decent replacement for Simulink

Things you may have thought about Python

And you were wrong

- Python's syntax is complicated and not suitable for quick prototyping
- Python doesn't handle well matrices and vectors
- Python doesn't have the equivalent to Matlab's toolboxes
- Python doesn't have full-featured environment
- Installing Python is difficult

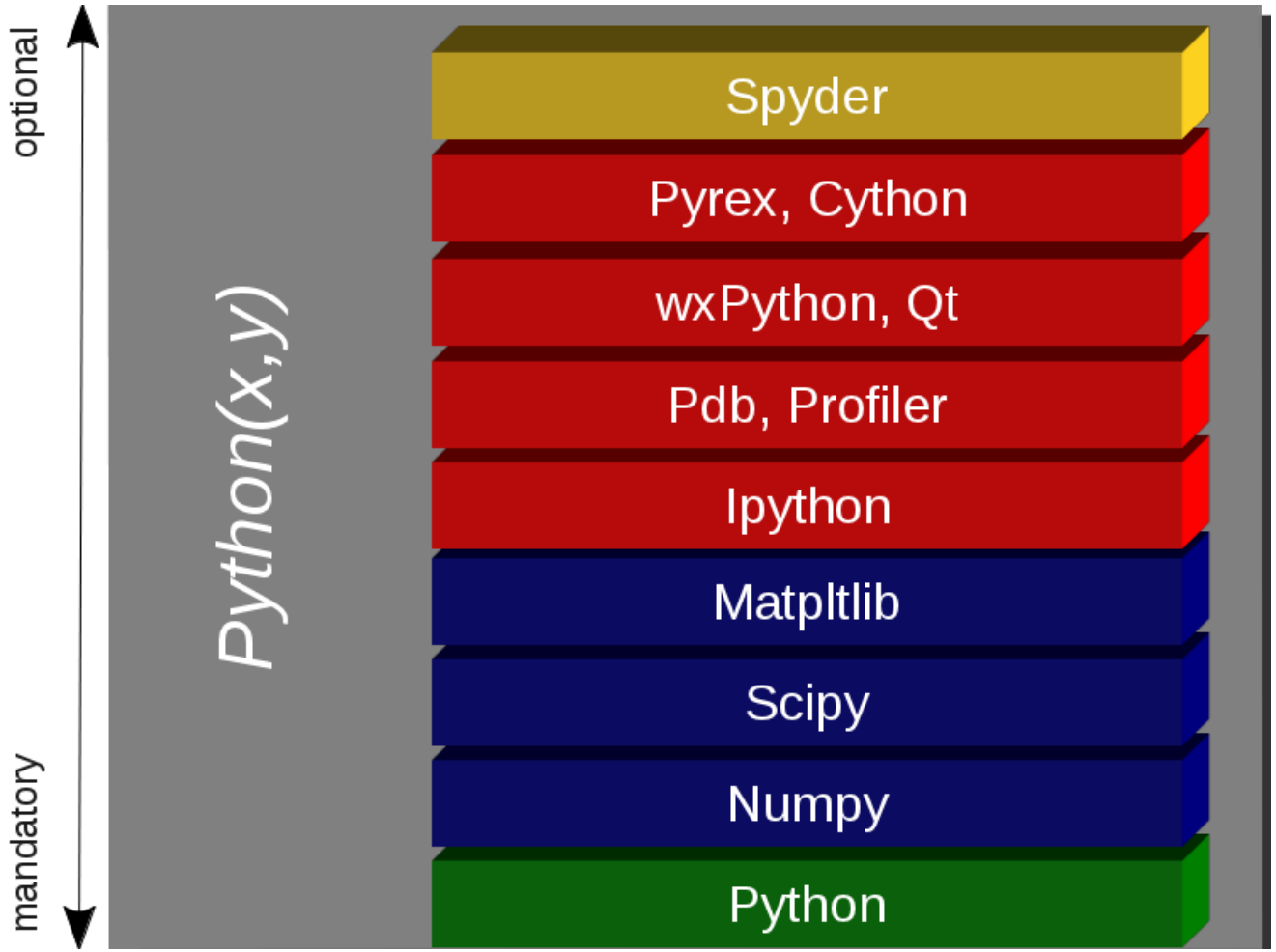
Things you may not have known about Python

- Python is considered to be among the five most popular programming languages in the world; Matlab falls short to somewhere between the 20th to 30th place
- Python is an interpreter, like Matlab

Example Python code

```
from math import sin,pi

def sinc(x):
    # Compute the sinc function: sin(pi*x)/(pi*x)
    try:
        val = (x*pi)
        return sin(val)/val
    except ZeroDivisionError:
        return 1.0

input=[0,0.1,0.5,1.0] # list of input values
output=[sinc(x) for x in input]

print output
```

How to start


optional
mandatory
Python(x,y)
Spyder
Pyrex, Cython
wxPython, Qt
Pdb, Profiler
Ipython
Matpltlib
Scipy
Numpy
Python

Scipy sub-modules

- Fourier Transforms (scipy.fftpack)
- Signal Processing (scipy.signal)
- Linear Algebra (scipy.linalg)
- Multi-dimensional image processing (scipy.ndimage)
- File IO (scipy.io)
- Clustering package (scipy.cluster)
- Discrete Fourier transforms (scipy.fftpack)
- Integration and ODEs (scipy.integrate)
- Statistical functions (scipy.stats)
- Interpolation (scipy.interpolate)
- Maximum entropy models (scipy.maxentropy)
- Multi-dimensional image processing (scipy.ndimage)
- Orthogonal distance regression (scipy.odr)
- Optimization and root finding (scipy.optimize)
- Sparse matrices (scipy.sparse)
- Spatial algorithms and data structures (scipy.spatial)

And many more

@ http://www.scipy.org/Topical_Software

Scipy sub-modules

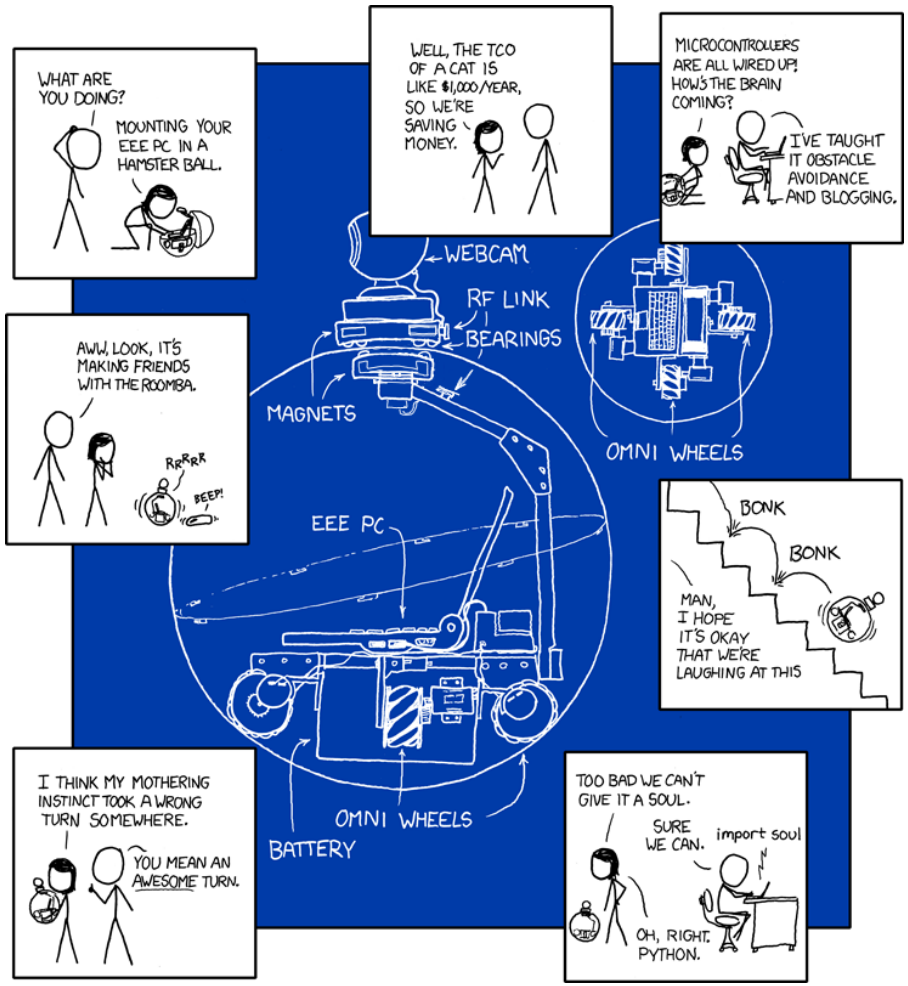
- Fourier Transforms (scipy.fftpack)
- Signal Processing (scipy.signal)
- Linear Algebra (scipy.linalg)
- Multi-dimensional image processing (scipy.ndimage)
- File IO (scipy.io)
- Clustering package (scipy.cluster)
- Discrete Fourier transforms (scipy.fftpack)
- Integration and ODEs (scipy.integrate)
- Statistical functions (scipy.stats)
- Interpolation (scipy.interpolate)
- Maximum entropy models (scipy.maxentropy)
- Multi-dimensional image processing (scipy.ndimage)
- Orthogonal distance regression (scipy.odr)
- Optimization and root finding (scipy.optimize)
- Sparse matrices (scipy.sparse)
- Spatial algorithms and data structures (scipy.spatial)

And many more

@ http://www.scipy.org/Topical_Software

Import everything, even soul...

http://xkcd.com/353/



Outline

1 Introduction
- Where am I coming from
- Where does Python come from
- What is Python
- What you need

2 Comparison with Matlab
- Inital Comparison
- Getting help
- Advanced examples

3 Advantages
- In favor of Matlab
- In favor of Python

4 Demo
- What is RANSAC?
- Demonstration

5 Summary

Functions

Matlab code: f2c.m and c2f.m

```
function c=f2c(f)
  c=(f-32)*(100/180);

function f=c2f(c)
  f=(180/100)*c+32;
```

Python code: convert.py

```
def f2c(f):
    return (f-32)*(100.0/180.0)

def c2f(c):
    return (180.0/100.0)*c+32
```

Interactive environment

Running Matlab code

```
>> a=f2c(212)

a =

    100

>> b=c2f(-40)

b =

    -40
```

Interactive environment

Running Python code

```
In [1]: from convert import *

In [2]: a=f2c(212)

In [3]: a
Out[3]: 100.0

In [4]: b=c2f(-40)

In [5]: b
Out[5]: -40.0
```

Interactive environment

Running Python code

```
In [7]:import convert

In [8]:a=convert.f2c(212)

In [9]:a
Out[9]:100.0

In [10]:dir(convert)
Out[10]:['__builtins__', '__doc__',
         '__file__', '__name__',
         'c2f', 'f2c']
```

So functions and modules are objects as well. And also lists, arrays, integers, and about everything else in Python

Interactive environment

Python's Namespaces are a bit like Matlab's Workspaces, but much more general and robust

Running Python code

```
In [2]: import convert as con

In [3]: import myconvert as mycon

In [4]: x = con.f2c(212)
Out[5]: 100.0

In [6]: y = mycon.f2c(212)
Out[7]: 0.0
```

So Namespaces are important

Getting help in Matlab

Matlab's help

```
>> help fft
 FFT Discrete Fourier transform.
FFT(X) is the discrete Fourier transform (DFT) of vector
X. For matrices, the FFT operation is applied to each
column. For N-D arrays, the FFT operation operates on
the first non-singleton dimension.
...
```

Getting help in Python

Python's help

```
In [14]:help(fft)
Help on function fft in module numpy.fft.fftpack:

fft(a, n=None, axis=-1)
    fft(a, n=None, axis=-1)

    Return the n point discrete Fourier transform of a.
    n defaults to the length of a. If n is larger than the
    length of a, then a will be zero-padded to make up the
    difference.  If n is smaller than the length of a, only
    the first n items in a will be used.

...
```

Getting help in Python

Help is also available for modules

Python's help

```
In [20]:help(scipy)

Help on package scipy:

NAME
    scipy

FILE
    /usr/local/lib/python2.5/site-packages/scipy/__init__.py

DESCRIPTION
    SciPy --- A scientific computing package for Python
    ===================================================
...
    Available subpackages
    ---------------------
    ndimage             --- n-dimensional image package [*]
    stats               --- Statistical Functions [*]
...
```

Getting help in Python

Zen

```
In [23]:import this
The Zen of Python, by Tim Peters

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!
```

Fibonacci

Matlab

```
function f = fibonacci(n)
% FIBONACCI Fibonacci
% sequence
% f = FIBONACCI(n) generates
% the first n Fibonacci
% numbers.


f = zeros(n,1);
f(1) = 1;
f(2) = 2;
for k = 3:n
  f(k) = f(k-1) + f(k-2);
end
```

Python

```
def fibonacci(n):
    # Fibonacci sequence

    from numpy import zeros

    f=zeros(n)
    f[0] = 1
    f[1] = 2
    for k in range(2,n):
        f[k]=f[k-1]+f[k-2]

    return f
```

Fibonacci - lists and arrays

Python arrays

```
def fibonacci(n):
    # FIBONACCI Fibonacci sequence

    from numpy import zeros

    f=zeros(n)
    f[0] = 1
    f[1] = 2
    for k in range(2,n):
        f[k]=f[k-1]+f[k-2]

    return f
```

Arrays are like matrices, for mathematical computation

Python lists

```
def fibonacci2(n):
    # FIBONACCI Fibonacci sequence

    f=[1,2] # use a list
    for k in range(2,n):
        f.append(f[k-1]+f[k-2])

    return f
```

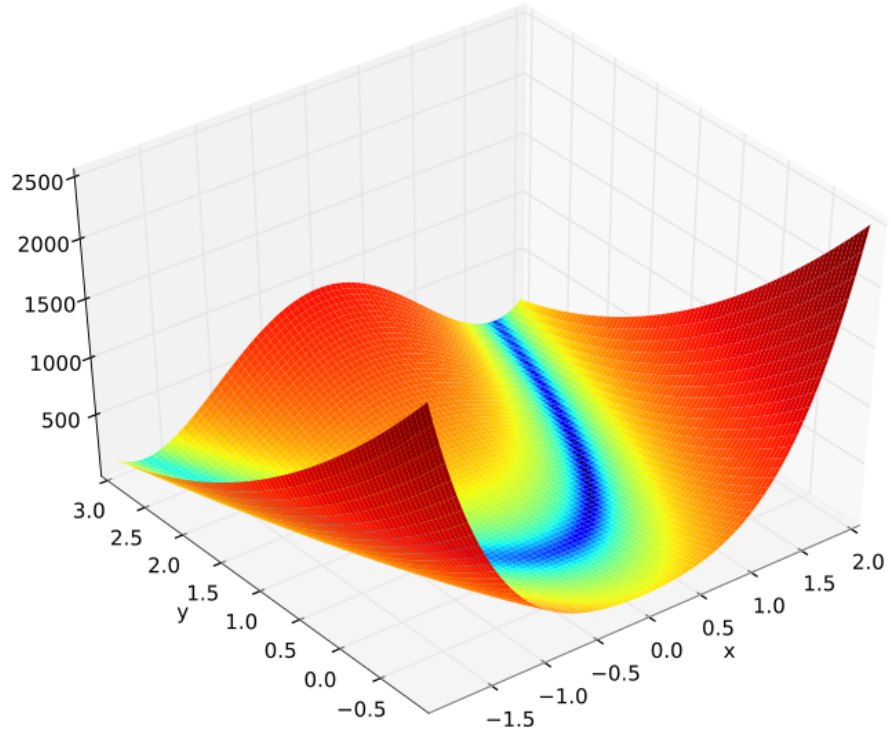
Lists are like cell arrays, for data manipulation

Optimiztion

Rosenbrock Function of N variables

$$f(\mathbf{x}) = \sum_{i=2}^{N} 100\left(x_i - x_{i-1}^2\right)^2 + \left(1 - x_{i-1}^2\right)^2$$

Minimum at $x_1 = x_2 = \cdots = 1$



Optimization

Python code

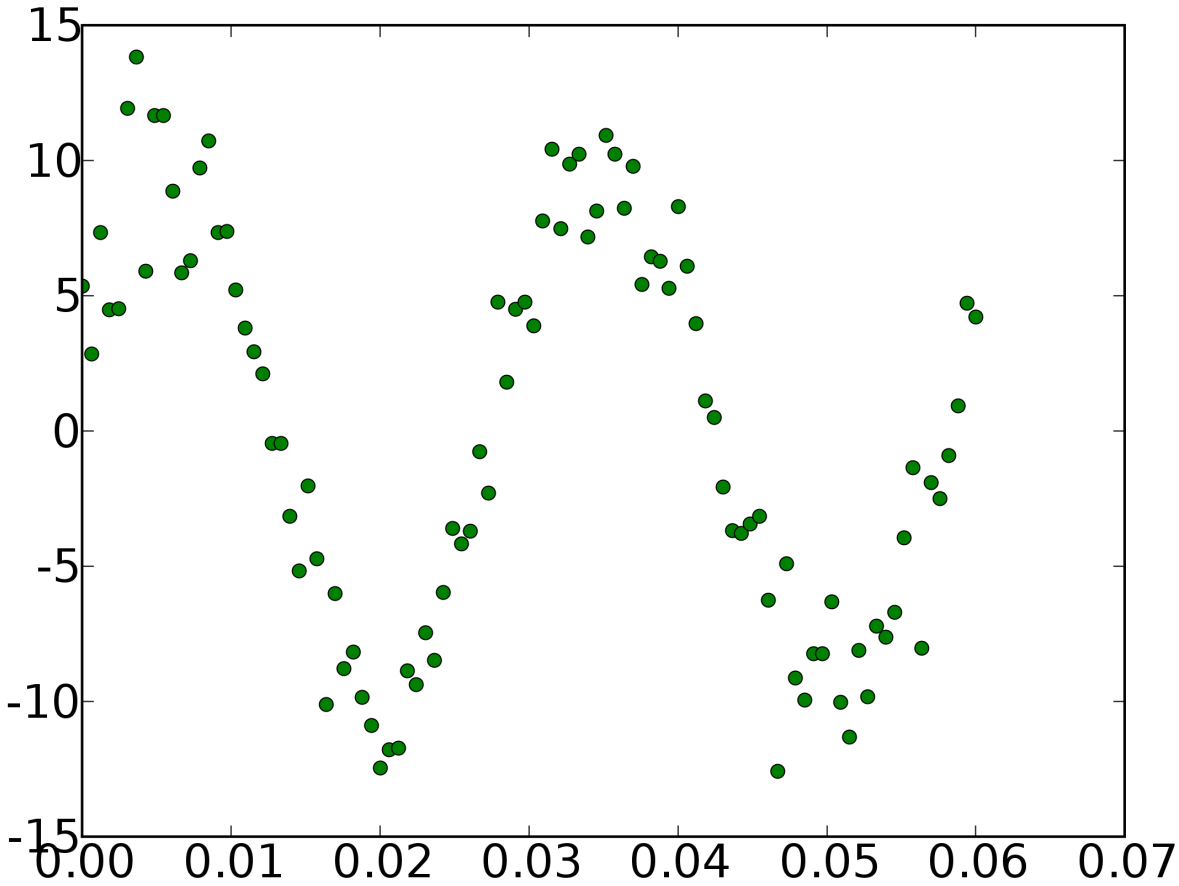
```
from scipy.optimize import fmin
def rosen(x): # The Rosenbrock function
    return sum(100.0*(x[1:]-x[:-1]**2.0)**2.0 + (1-x[:-1])**2.0)

x0 = [1.3, 0.7, 0.8, 1.9, 1.2]
xopt = fmin(rosen, x0) # Nelder-Mead simplex algorithm
```

Running Python code

```
Optimization terminated successfully.
         Current function value: 0.000066
         Iterations: 141
         Function evaluations: 243
[ 0.99910115  0.99820923  0.99646346  0.99297555  0.98600385]
```

Least Squares

Fitting a sine wave



Fitting a sine wave

Generate the data

```
from pylab import *
from numpy import *
import scipy
from scipy import optimize

x=linspace(0,6e-2,100)
A,k,theta = 10, 1.0/3e-2, pi/6
y_true = A*sin(2*pi*k*x+theta)
y_meas = y_true + 2*randn(len(x))
```

Fitting a sine wave

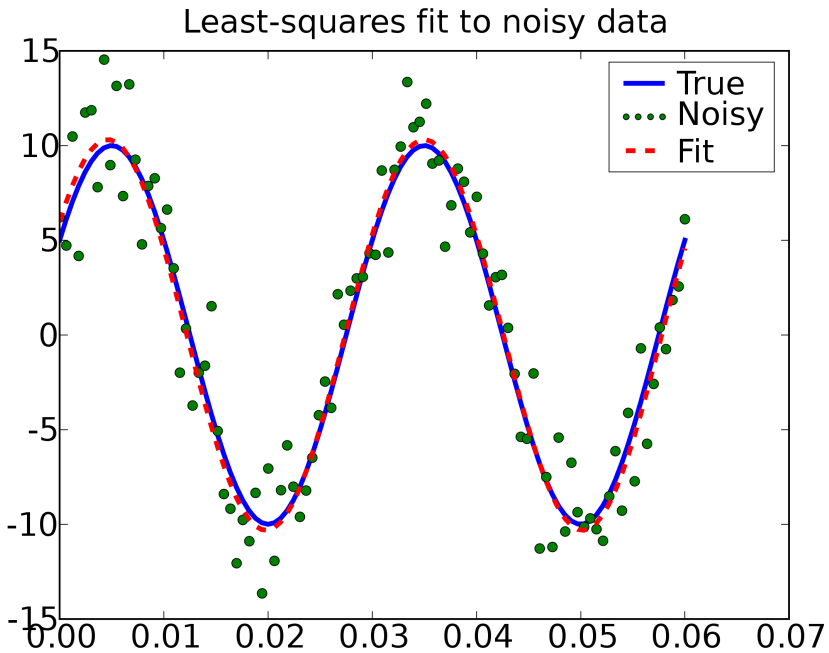
Fit the data

```
from pylab import *
from numpy import *
import scipy
from scipy import optimize

def residuals(p, y, x):
    A,k,theta = p
    err = y-A*sin(2*pi*k*x+theta)
    return err

def peval(x, p):
    return p[0]*sin(2*pi*p[1]*x+p[2])


# perform least squares estimation
p0 = [20, 40, 10]
print "Initial values:",p0

plsq = optimize.leastsq(residuals, p0, args=(y_meas, x))
print "Final estimates:",plsq[0]

print "Actual values:", [A, k, theta]
```

Fitting a sine wave

Output

```
Initial values: [20, 40, 10]
Final estimates: [-10.41111011  33.09546027  10.00631967]
Actual values: [10, 33.333333333333336, 0.5235987755982988]
```

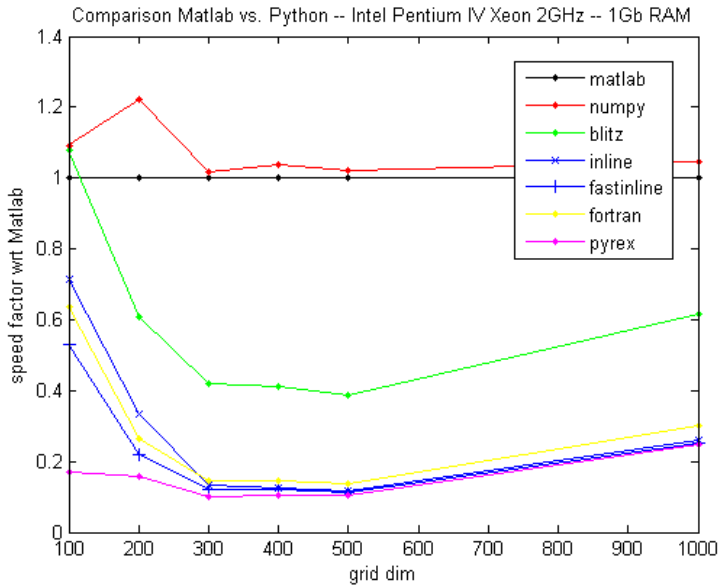


Speed

What about performance? "Maybe laundry isn't your biggest problem right now..."

Type of Solution	Time taken [sec]
Python	~1500
Python+Numpy	**29.3**
Python+Fortran	2.9
Pyrex	2.5
Pure C++	2.16
Octave	60.0
Matlab	**29.0**

Souce:
http://www.scipy.org/PerformancePython



Source:
http://lbolla.info/blog/2007/04/11/numerical-computing-matlab-vs-pythonnumpyweave/

Outline

1 Introduction
- Where am I coming from
- Where does Python come from
- What is Python
- What you need

2 Comparison with Matlab
- Inital Comparison
- Getting help
- Advanced examples

3 Advantages
- In favor of Matlab
- In favor of Python

4 Demo
- What is RANSAC?
- Demonstration

5 Summary

Advantages in favor of Matlab

Clean syntax for inputing matrices

Matlab

```
a=[1 2 3 ; 4 5 6]
```

Python - array

```
import numpy as np
a=np.array([[1,2,3] , [4,5,6]])
```

Or:

Python - matrix

```
import numpy as np
a=np.matrix([[1,2,3] , [4,5,6]])
```

Advantages in favor of Matlab

Clean syntax for inputing range

Matlab

```
a=1:10

b=linspace(1,10,10)
```

Python

```
import numpy as np

a=np.r_[1:11]
# 1 minus last number

b=np.linspace(1,10,10)
# better way
```

Advantages in favor of Matlab

Better integrated plot commands

Matlab

```
x=-10:10
y=x.^2
plot(x,y,'-o')
```

Python

```
from pylab import *
from numpy import *

x=linspace(-10,10,20)
y=x**2

plot(x,y,'-o')
show()
```

Advantages in favor of Python

- Free as in speech and beer
- Community suuport and available scientific modules
- Real obect-orientd programming
- Namespaces enable scaling to larger projects

Matlab

```
a=sqrt(2)  % built-in

% uses first fmin in path
fmin('cos',3,4)
```

Python

```
import math
import mymath

a=math.sqrt(2)
b=mymath.sqrt(2)

from scipy.optimize import fmin
from myopt import fmin as fmin2

from math import cos
fmin(cos,3,4) # uses scipy fmin
fmin2(cos,3,4) # uses my fmin
```

- Satandard library for multiple purposes

Advantages in favor of Python

Python's standard library

- files types
- data types
- databases
- comm & network protocols
- cryptography
- compression
- OS (multi-threading)
- internationalization
- multimedia
- GUI generation

Outline

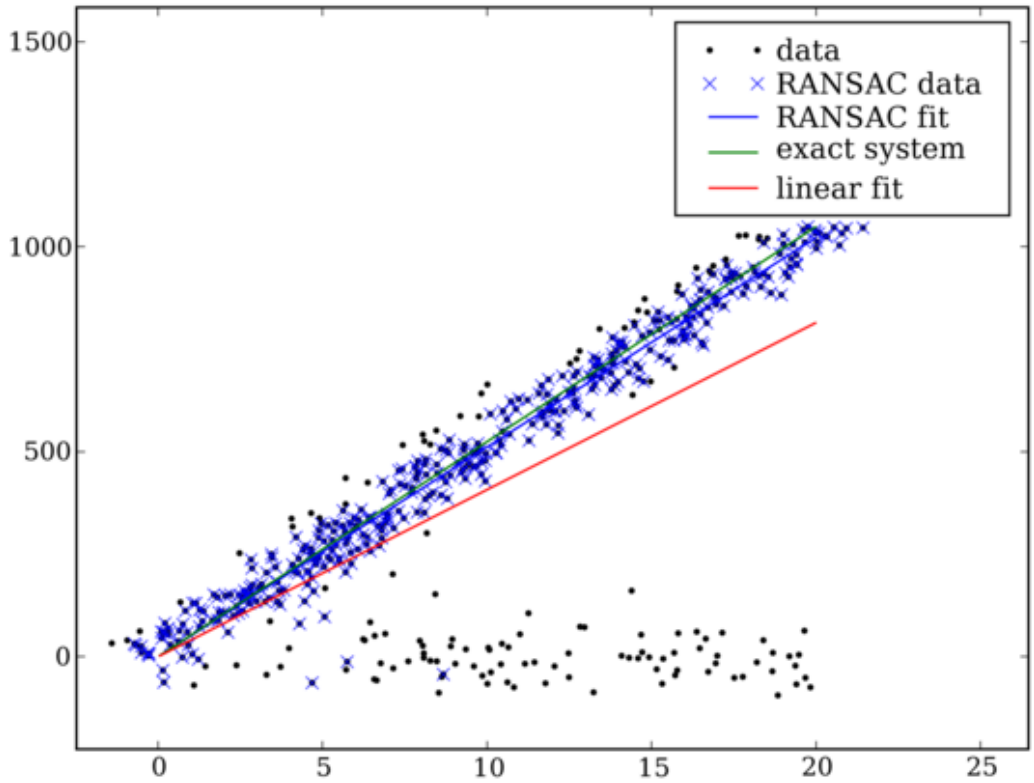
1. Introduction
   - Where am I coming from
   - Where does Python come from
   - What is Python
   - What you need
2. Comparison with Matlab
   - Inital Comparison
   - Getting help
   - Advanced examples
3. Advantages
   - In favor of Matlab
   - In favor of Python
4. Demo
   - What is RANSAC?
   - Demonstration
5. Summary

What is RANSAC?

- RANdom SAmple Consensus
- An iterative method to estimate parameters of a mathematical model from a set of observed data which contains outliers

n—number of points in data
w—number of inliers in data / number of points in data
p—desired probabilty for successful classification
k—number of required drwas

$$\Longrightarrow k = \frac{log(1-p)}{log(1-w^n)}$$



Demo

Spyder

Outline

1. Introduction
   - Where am I coming from
   - Where does Python come from
   - What is Python
   - What you need
2. Comparison with Matlab
   - Inital Comparison
   - Getting help
   - Advanced examples
3. Advantages
   - In favor of Matlab
   - In favor of Python
4. Demo
   - What is RANSAC?
   - Demonstration
5. Summary

Conclusions (based on personal experience)

1. Python is much more flexible than Matlab
2. Matlab is a bit better at quick prototyping; Python is much better at large-scale projects
3. There is nothing you can do with Matlab that you can't do with Python, but this doesn't always work the other way around
4. Python is sometimes ambiguous: you can do the same thing in more than one way. Once comprehended, it becomes an advantage

Special thanks to Prof. Brian Blais from Bryant University for sharing his slides with me

http://web.bryant.edu/~bblais/

Summary

Questions?

Thanks!

uri.barkan at gmail dot com