Computação Científica com Python

Fabricio Ferrari

Unipampa/Bagé

2009

Objetivos

- Pequeno tutorial de Python
- Alternativas aos ambientes/linguagens “tradicionais”
- Apresentar módulos úteis (Numpy, SciPy, MatPlotLib)
- Exemplos de Uso

Python

- Linguagem de programação de alto nível
- interpretada**
- interativa
- orientada a objetos
- tipagem dinâmica e forte
- extensa biblioteca "baterias incluídas"
- v0.9 1991 Guido van Rossum
- implementações: CPython, Jython, IronPython, PyPy, mod_python, Python for S60, ...
- Maya, Softimage XSI, Blender, GIMP, Inkscape, Scribus, Paint Shop Pro. YouTube, BitTorrent, Google,Yahoo!, CERN, NASA.

Foco

- Qualidade: legibilidade, coerencia, reusabilidade, mantenibilidade
- Produtividade: Ciclos de desenvolvimento, tamanho do código
- Portabilidade: Linux, Windows, Macs, NetBSD, OpenBSD, celulares,
- Bibliotecas: biblioteca padrão; extensões em C, Fortran,
- Diversão
- Influenciado por C, Lisp, Modula-3, Perl, Smalltalk, Tcl, MatLab.

O Zen do Python[1]

Bonito é melhor que feio
Explicito é melhor que implícito
Simples é melhor que complexo
Direto é melhor que encadeado
Esparso é melhor que denso
Legibilidade conta
Casos especiais não são especiais o suficiente para quebrar as regras
Embora praticabilidade bata a pureza
Erros nunca devem passar silenciosamente
A não ser que explicitamente silenciados
Em face da ambiguidade, recuse a tentação de adivinhar
Deve haver um – e preferencialmente só um – modo de fazer
Embora este modo pode não ser óbvio
Agora é melhor que nunca
Embora nunca às vezes é melhor que agora **já**.
Se a implementação é difícil de explicar, é uma idéia ruim
Se a implementação é fácil de explicar, pode ser uma idéia boa
Espaços de nomes são uma grande idéia - vamos fazer mais

[1] > > > import this

Sintaxe e Semântica

Identação é o delimitador de blocos

Python:

```
if x<1:
   print x
   x = x+1
```

C:

```
if (x<1) {
   print("%f\n", x);
   x = x+1;
}
```

```
if (x<1)
{
   print("%f\n", x);
   x = x+1;
}
```

```
if (x<1){ print("%f\n", x);x++;}
```

Exemplos de Identação

```
for k in range(self.K):
    if self.verbose: print 'Estimating noise at scale: ',k

    for i in range(0, self.M):
        for j in range(0, self.N):
            x1 = j - self.sigma_viz
            x2 = j + self.sigma_viz + 1

            if x1 < 0      : x1 = 0
            if x2 >= self.N : x2 = self.N

            y1 = i - self.sigma_viz
            y2 = i + self.sigma_viz + 1

            if y1 < 0      : y1 = 0
            if y2 >= self.M : y2 = self.M

            self.sigma[k,i,j] = numpy.std(self.W[k,y1:y2,x1:x2])
```

```
def valor1():
    while True:
        try:
            c = int(raw_input('Primeiro Valor: '))
            return c
        except ValueError:
            print 'Inválido!'
```

correto

```
def valor1():
while True:
try:
c = int(raw_input('Primeiro Valor: '))
return c
except ValueError:
print 'Inválido!'
```

incorreto

Tipagem dinâmica e forte

str: sequencia imutável de caracteres `'cruzeira, u'jararaca', "bothrops"`

bytes: sequencia imutável de bytes `b'ABDE22 00'`

int: número de precisão fixa de magnitude ilimitada `1, 223323, 12`

float: número de ponto flutuante de precisão variável `3.141592654, 6.02E23`

complex: número complexo com parte real e imaginária `3+2.5j`

list: lista (heterogênea) de objetos `[1, 'ABC', 2+1j]`

tuple: lista imutável de objetos `(9, 'F', 2)`

dict: conjunto associativo `{'idade': 21, 'nome': 'Jonas' }`

set: conjunto não ordenado, itens não repetidos `{2, 6, 1, 0}`

bool: tabela verdade `True, False`

Operadores

Aritméticos

+ - * / % ** += -= *= /= %= **=

Relacionais

> < == >= <= <> != is in

Lógicos

and or not Binários

| ^ & >> << ~

Usando o Python

Modo interativo

```
vela ~ % python
Python 2.5.2 (r252:60911, Jul 31 2008, 17:31:22)
[GCC 4.2.3 (Ubuntu 4.2.3-2ubuntu7)] on linux2
Type "help", "copyright", "credits" or "license" for more informatic
>>>
```

Modo não interativo - invocando o interpretador

```
vela ~ % python meu_programa.py
```

Modo não interativo - chamando o executável (1a linha: `#!/usr/bin/python`)

```
vela ~ % chmod +x meu_programa.py
vela ~ % ./meu_programa.py
```

Inteiros, Ponto Flutuante

```
>>> a = 1            # atribuicao simples
>>> b = 3
>>> a/b              # divisao de inteiros
0
>>> c = 3.0
>>> a/c
0.33333333333333331
>>> import math      # importa modulo
>>> math.exp(2.345) # exponencial
10.433272727548918
>>> dir(math)        # mostra conteudo do modulo
['__doc__', '__file__', '__name__', 'acos', 'asin', 'atan', 'atan2', 'ceil',
'cos', 'cosh', 'degrees', 'e', 'exp', 'fabs', 'floor', 'fmod', 'frexp', 'hypot',
'ldexp', 'log', 'log10', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt',
'tan', 'tanh']
>>> help(math.exp)  # mostra ajuda da funcao math.exp
>>> x=1E-23          # atribuicao
>>> x**2             # quadrado de x
9.9999999999999983e-47
>>> Navogadro = 6.02E23
>>> Nparticulas = 5E24
>>> n = Nparticulas / avogadro
>>> n
8.305647840531563
>>> n > 4            # condição
True
```

Listas (quasi-vetores)

```
>>> x = [5,12,13,200]     # cria lista
>>> x
[5, 12, 13, 200]
>>> x.append(-2)          # acrescenta -2 no final
>>> x
[5, 12, 13, 200, -2]
>>> del x[2]              # remove item 3
>>> x
[5, 12, 200, -2]
>>> z = x[1:3]            # fatia do vetor": elementos do indice 1 até  3(exclusive)
>>> z
[12, 200]
>>> yy = [3,4,5,12,13]    # outra lista
>>> yy[3:]                # todos elementos a partir do indice  3(inclusive)
[12, 13]
>>> yy[:3]                # todos elementos até indice 3(exclusive)
[3, 4, 5]
>>> yy[-1]                # último elemento (um contando do final)
13
>>> x.insert(2,28)        # insere 28 no indice 2 (posição 3)
>>> x
[5, 12, 28, 200, -2]
>>> 28 in x               # testa se 28 está em x; True(1) ou False(0)
1
>>> 13 in x
0
>>> x.index(28)           # retorna o índice do elemento cujo valor 28
2
>>> x.remove(200)         # remove  o elemento cujo valor é 200
```

Operações em Listas

s[i] = J	substitui elemento
s[i:j] = T	substitui grupo de elementos
s.append(x)	adiciona elemento
s.index(x)	retorna o índice do valor x
s.insert(i,x)	adiciona x na posicao i
s.remove(x)	remove elemento x
s.reverese()	inverte a ordem
s.sort()	ordena lista

```
>>> s = [1,1,2,3,5,8]
>>> s[2] = 'a'
>>> s
[1, 1, 'a', 3, 5, 8]
>>> s.index(5)
4
>>> s.insert(2,'dois')
>>> s
[1, 1, 'dois', 'a', 3, 5, 8]
>>> s.reverse()
>>> s
[8, 5, 3, 'a', 'dois', 1, 1]
>>> s.sort()
>>> s
[1, 1, 3, 5, 8, 'a', 'dois']
>>> help(s)
```

Computação Científica

iPython Numpy SciPy PyLab

iPython

Shell melhorado para o Python

- completa comando e atributos com <TAB>
- Comandos mágicos %magic

```
% Exit, %Pprint, %Quit, %alias, %autocall, %autoindent, %automagic,
% bookmark, %cd, %color_info, %colors, %config, %dhist, %dirs, %ed,
% edit, %env, %hist, %logoff, %logon, %logstart, %logstate, %lsmagic,
% macro, %magic, %p, %page, %pdb, %pdef, %pdoc, %pfile, %pinfo, %popd,
% profile, %prun, %psource, %pushd, %pwd, %r, %rehash, %rehashx, %reset,
% run, %runlog, %save, %sc, %sx, %system_verbose, %unalias, %who,
% who_ls, %whos, %xmode
```

- Informação dinâmica dos objetos `?objeto` ?a, ??a
- histórico e log dos comandos

```
%history
%logstart diario.log
```

- Parenteses e aspas automáticas (LazyPython)
- Interação fácil com Pylab

iPython

Completa comandos com <TAB>

```
In [12]: import sys
In [13]: sys.std<TAB>
sys.stderr  sys.stdin   sys.stdout

In [21]: x= [1,2,3]

In [22]: x.r<TAB>
x.remove   x.reverse

In [23]: ?x
             Type:                 list
             Base Class:         <type 'list'>
             String Form:         [1, 2, 3]
             Namespace:                Interactive
             Length:                 3
             Docstring:
                 list() -> new list
                 list(sequence) -> new list initialized from sequence's items
```

Parenteses e aspas automáticas

```
In [1]: funcao arg1,arg2, arg3                  #  funcao(arg1,arg2,arg3)

In [2]: /listagem                               #  listagem()

In [3]: ,minha_funcao a b c                     #  minha_funcao('a', 'b', 'c')

In [4]: ;minha_funcao a b c                     #  minha_funcao('a b c')
```

`ndarrays` – Vetores homogeneos (*arrays*) N-dimensionais

Criando Vetores `numpy.<operação>`

`zeros((M,N))`	vetor com zeros, M linhas, N colunas
`ones((M,N))`	vetor com uns, M×N
`empty((M,N))`	vetor vazio (qualquer valor), M×N
`zeros_like(A)`	vetor com zeros, formato do A.
`ones_like(A)`	vetor com uns, formato do A.
`empty_like(A)`	vetor vazio, formato do A.
`random.random((M,N))`	vetor com numeros aleatorios, M×N
`identity(N,float)`	matriz identidade N×N, ponto flutuante
`array([(1.5,2,3),(4,5,6)])`	especifica os valores da matriz
`mgrid[1:3,2:5]`	grade retangular x=[1,2] e y=[2,3,4]
`fromfunction(f, (M,N))`	matriz calculada com função f(i,j), M×N
`arange(I, F, P)`	vetor com inicio I, fim F, passo P
`linspace(I,F,N)`	vetor com N numeros de I até F

Métodos dos Vetores

A é um numpy.ndarray

`A.sum()`	soma dos itens
`A.min()`	valor mínimo
`A.max()`	valor máximo
`A.mean()`	média aritmética
`A.std()`	desvio padrão
`A.var()`	variância
`A.trace()`	traço
`A.size()`	número de elementos
`A.shape()`	formato
`A.ptp()`	pico-a-pico (maximo - minimo)
`A.ravel()`	versão 1D
`A.transpose(), A.T`	matriz transposta
`A.resize(M,N)`	reforma ou trunca a matriz *in situ*
`A.reshape(M,N)`	retorna matriz com novo formato
`A.clip(Amin,Amax)`	corta valores em Amin e Amx
`A.compress(condicao)`	seleciona elementos baseado em condicao
`A.conjugate()`	complexo conjugado
`A.copy()`	retorna cópia
`A.fill(valor)`	preenche com valor

Operações com vetores

`numpy.<operação>`

`C = A-B, C=A+B, C=A*B, C=A/B, A**2`	operações elemento a elemento ($C_{i,j} = A_{i,j} - B_{i,j}$)
`dot(A,B), mat(A)*mat(B)`	produto matricial
`inner(A, B)`	produto interno
`outer(A, B)`	produto externo
`concatenate(arrays, axis=0)`	concatena vetores
`vstack(A,B)`	empilha verticalmente vetores
`hstack(A,B)`	empilha horizontalmente vetores
`vsplit(A,2)`	parte verticalmente vetor
`hsplit(A,2)`	parte horizontalmente vetor
`A[0]`	primeiro elemento
`A[i][j], A[i,j]`	convenção dos índices A_{ij} (linha `i`, coluna `j`)
`A[3][2]`	A_{32} 3ro elemento na 4ta linha
`A[1]`	2da linha
`x[2]`	3ro elemento
`x[-2]`	penúltimo elemento (índice contando do fim)
`x[2:5]`	subvetor de 3ro até o 5to, `[x[2], x[3], x[4]]`
`x[:5]`	do início `x[0]` até o quinto `x[4]`
`x[2:]`	do 3ro até o fim
`x[:]`	o vetor inteiro
`x[2:9:3]`	do 3ro até o 10mo, a cada 3, `[x[2], x[5], x[8]]`
`x[numpy.where(x>7)]`	elementos em `x` maiores que 7

Exemplos

Numpy

doc	Documentação
random	ferramentas para números aleatórios
linalg	Algébra Linear
fft	Tranformada de Fourier

SciPy

misc	Utilidades variadas (`comb, factorial, derivative, imfilter, ...`)
fftpack	Transformada de Fourier Discreta
io	E/S dados (`loadtxt, fread,...`)
special	Funções especiais (Airy, Bessel, Legendre, Gamma, `erf()`)
stats	Funções estatísticas (>100 distribuicoes, momenta, ...)
optimize	Optimizadores (ajuste, extremas, raízes,..)
spatial	Algoritmos para estruturas de dados espaciais (vizinhos, ...
integrate	Rotinas de integração numérica
linalg	Rotinas de Algebra Linear (`inv, solve, det,eig, svd, ...`)
interpolate	Rotinas de Interpolaão (`spline, lagrange, ...`)
signal	Proc. Sinais Digitais (`convolve, correlate, filters, windows, ...`)

Exemplo 1 Numpy

Operações com matrizes

```
>>> A = numpy.array([[1,2,3],[3,2,3],[1,4,2]])
>>> A
array([[1, 2, 3],
       [3, 2, 3],
       [1, 4, 2]])
>>> # mostra media, desvio padrao, minino, maximo, pico-a-pico
>>> A.mean(), A.std(), A.min(), A.max(), A.ptp()
(2.333333333333335, 0.94280904158206325, 1, 4, 3)
>>> A**2                    # eleva cada elemento ao quadrado
array([[ 1,  4,  9],
       [ 9,  4,  9],
       [ 1, 16,  4]])
>>> B = A.T                 # matriz transposta
array([[1, 3, 1],
       [2, 2, 4],
       [3, 3, 2]])
>>> numpy.dot(A, B)         # multiplicacao de matrizes
array([[11, 12, 14],
       [12, 24, 20],
       [14, 20, 22]])
>>> A[ numpy.where(A>2) ] = 10   # iguala a 10 todos elementos maiores que 2
>>> A * B                        # multiplica A e B elemento a elemento
```

Exemplo 2 Numpy

Operações com matrizes

AMD Turion 64 X2 Mobile TL-60, bogomips=3993.03

```
# matriz de numeros aleatorios 1000x1000  dt=0.07 s
r     = numpy.random.random((1000,1000))

# inversa de r                            dt=5 s
inv_r = scipy.linalg.inv(r)

# autovalores e autovetores               dt=27 s
eigval, eigvec = scipy.linalg.eig(r)

# multiplica r pela transposta            dt=5 s
C = numpy.mat(r.T) * numpy.mat(r)

# faz media entre cada 8 vizinhos         dt=0.7 s
C9 = scipy.signal.convolve2d(ac,  [[1,1,1],[1,1,1],[1,1,1]])

# faz media entre cada 99 vizinhos        dt=3 s
C100 = scipy.signal.convolve2d(r, 10*[[1,1,1,1,1,1,1,1,1,1]])
```

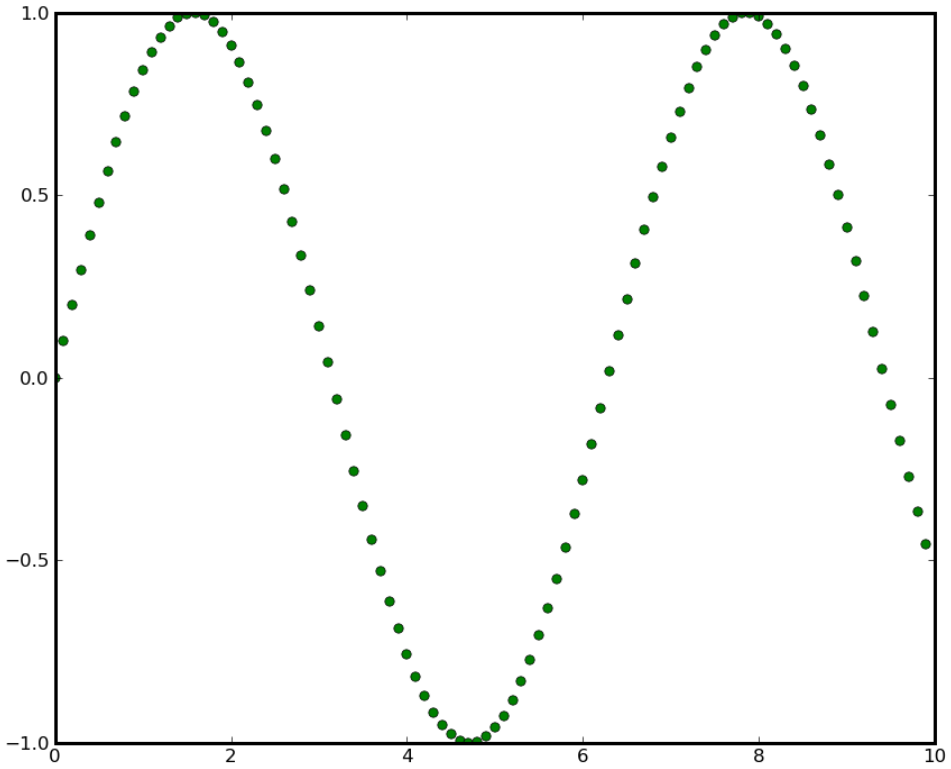
Maple

```
# Cria matriz                             dt=4s
r := RandomMatrix(1000,generator=0..1.5)
# Multiplica por ela mesma                dt>30 s
multiply(r,r)
```

Exemplo 3: Numpy+Pylab

Gráfico Função

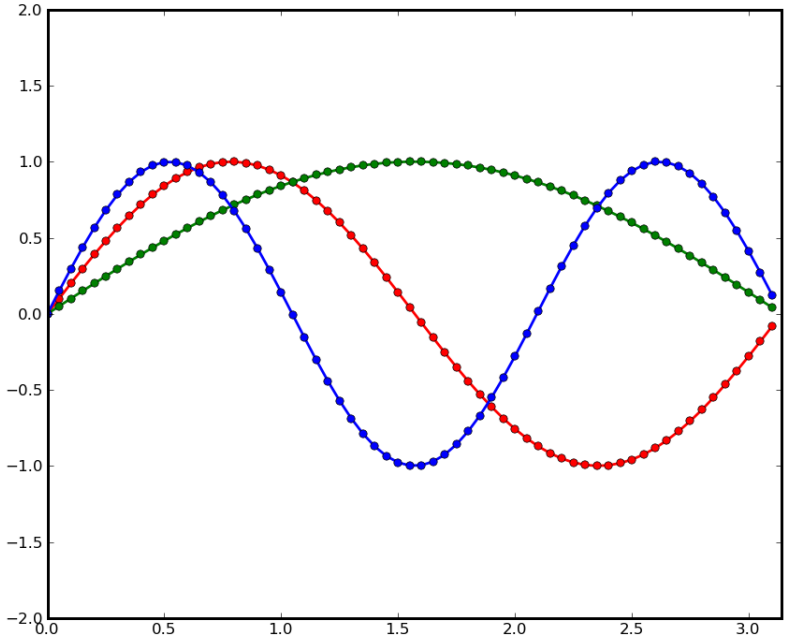
```
import numpy                  # importa módulos
import pylab
x = numpy.arange(0,10,0.1) # sequencia de 0 a 19, intervalo 0.1
y = numpy.sin(x)              # aplica funcao em cada x (versao vetor math.sin)
pylab.plot(x,y, 'og')         # faz  grafico com bolas 'o' verdes 'g'
pylab.savefig('seno.png')  # grava figura
pylab.show()                  # mostra grafico
```



Exemplo 4: Numpy+Pylab

Gráfico Funções

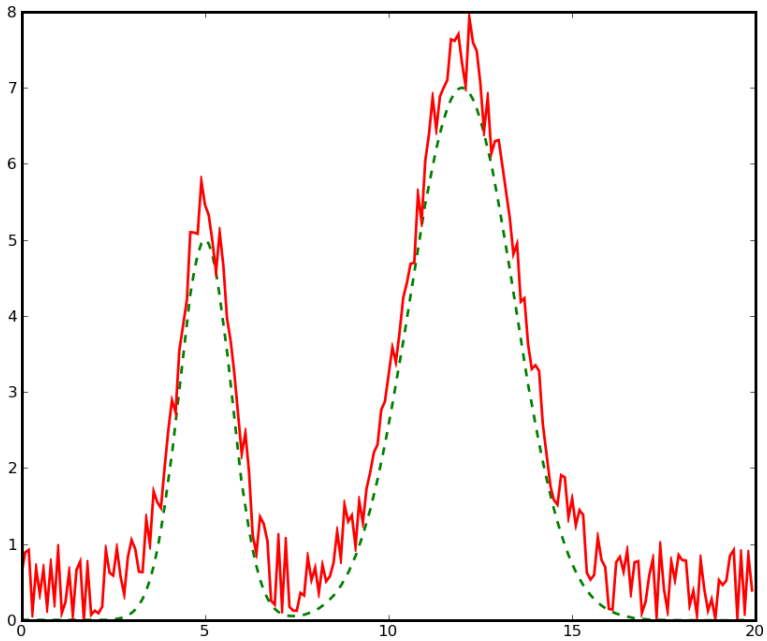
```
x = numpy.arange(0,numpy.pi,0.05)
y  = numpy.sin(x)
y2 = numpy.sin(2*x)
y3 = numpy.sin(3*x)
pylab.plot(x,y, 'o-g')
pylab.plot(x,y2,'o-r')
pylab.plot(x,y3,'o-b')
pylab.axis([0,2*numpy.pi,-2,2])
pylab.savefig('senos.png')
pylab.show()
```



Exemplo 5: Numpy+Pylab

Adicionando Ruído

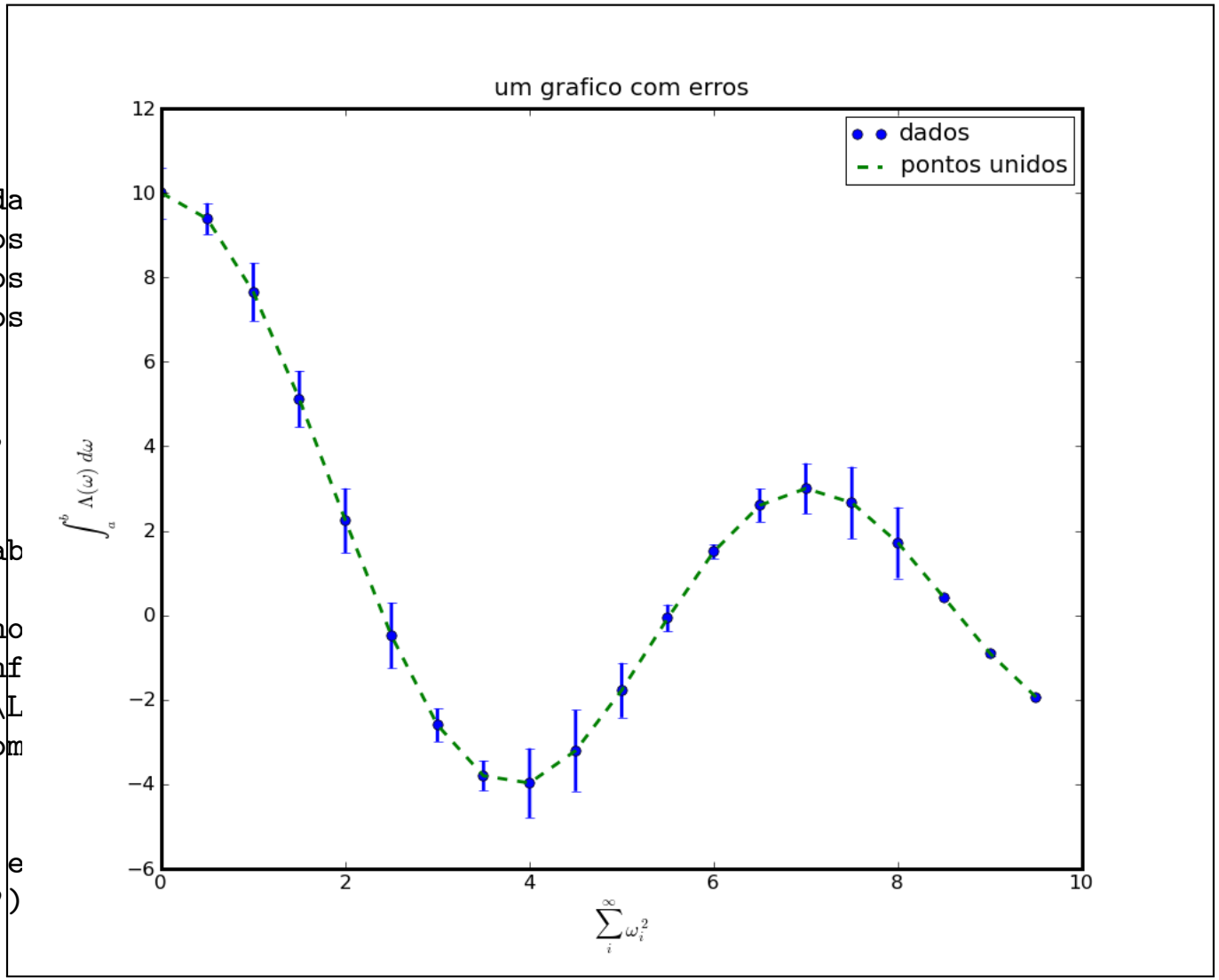
```
x  = numpy.arange(0,20,0.1)          # variavel independente
y1 = 5*numpy.exp(-(x-5)**2)          # gaussiana centrada em 5, maximo 5
y2 = 7*numpy.exp(-(x-12)**2/4)       # gaussiana centrada em 12, maximo 7
ruido = numpy.random.random(len(x)) # vetor ruido do tamanho do x
ylimpo = y1 + y2                     # gaussians
yruido = y1 + y2 + ruido             # gaussianas mais o ruido
pylab.plot(x, ylimpo, '--g', x, yruido, '-r', lw=2)
```



Exemplo 6: Numpy+Pylab

Lendo dados em disco

```
import pylab
import numpy

# le os dados de 'dados1.da
x    = numpy.loadtxt('dados
y    = numpy.loadtxt('dados
yerr = numpy.loadtxt('dados

# grafico dos dados
pylab.errorbar(x, y, yerr,

# une os pontos com retas
pylab.plot(x, y, '--g', lab

# coloca nomes dos eixos,no
pylab.xlabel(r'$\sum_i^\inf
pylab.ylabel(r'$\int^b_a \L
pylab.title('um grafico com
pylab.legend()

# grava imagem do grafico e
pylab.savefig('dados1.png')

# mostra janela
pylab.show()
```



Exemplo 7: Numpy+PyFits

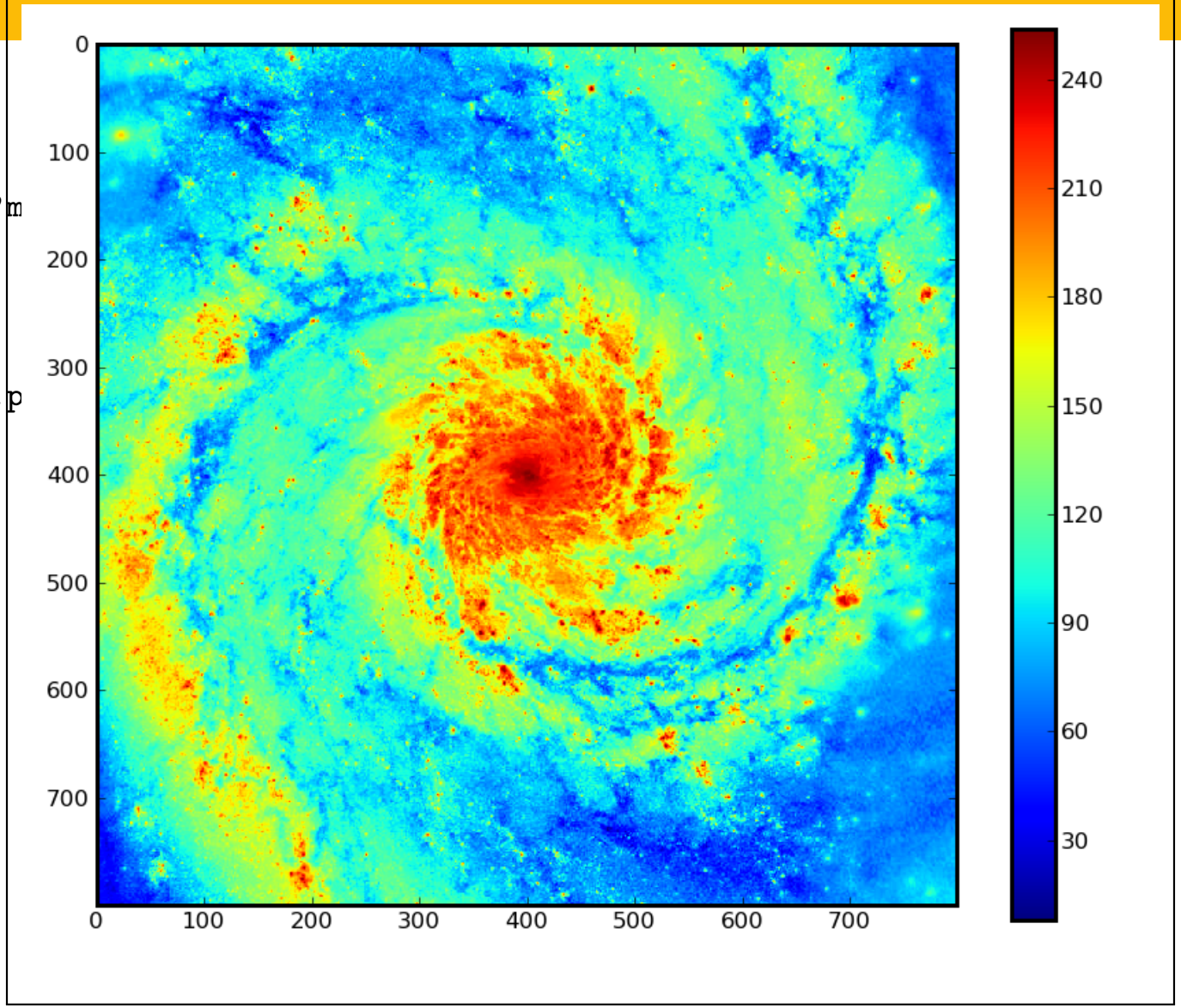
Lendo cubo de dados

```
>>> import pyfits

# Le cubo de dados de 471 Mb                dt=12s
>>>m81 =  pyfits.getdata('M81_final.fits');

>>> m81.shape     # formato (z, y, x)
(2707, 310, 147)

>>> m81.size      # quantidade de pontos  120 Mpix
123357990

>>> m81.mean()    # media                   dt=13s
1.9765925252298883e-14

>>> m81.std()     # desvio padrao           dt=58s (112s c/swap)
3.4743369904020077e-13

>>> m81.sum()     # soma total              dt=3.5s
2.4382848e-06

>>> m81.sort()    # ordena 120M de floats   dt=7s
```

Exemplo 8: PyLab+PyFits

Lendo imagem FITS

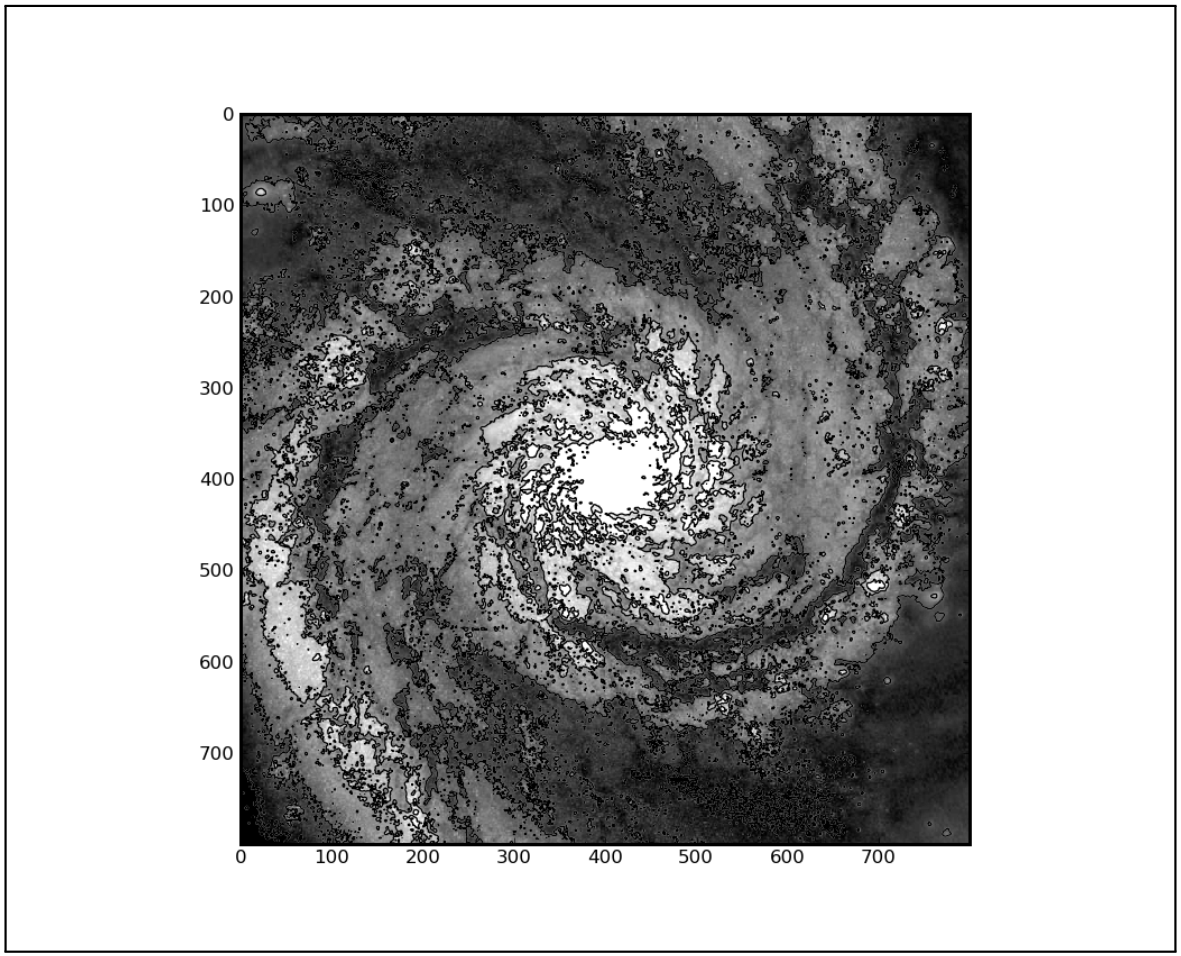
```
>>> import pyfits
>>> import pylab
>>> m51 = pyfits.getdata('m
>>> m51.shape
(800,800)
>>> pylab.imshow(m51)
>>> pylab.colorbar()
>>> pylab.savefig('m51hst.p
>>> pylab.show()
```



Exemplo 9

Contornos

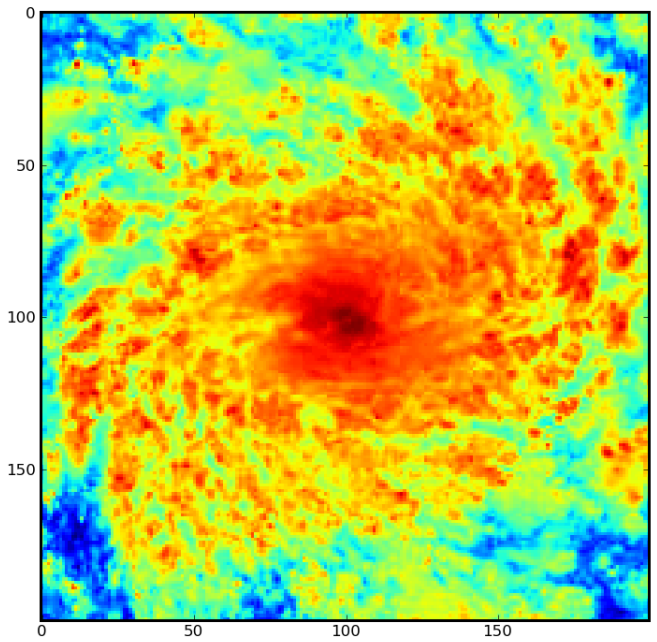
```
...
>>> pylab.imshow(m51, vmin=50, vmax=200)
>>> niveis =  [50, 100, 150, 200]
>>> pylab.contour(m51,  niveis, colors='0.0', linewidths=1.0)
>>> pylab.savefig('m51hst-contour.png')
```



Exemplo 10

Seções de Imagens

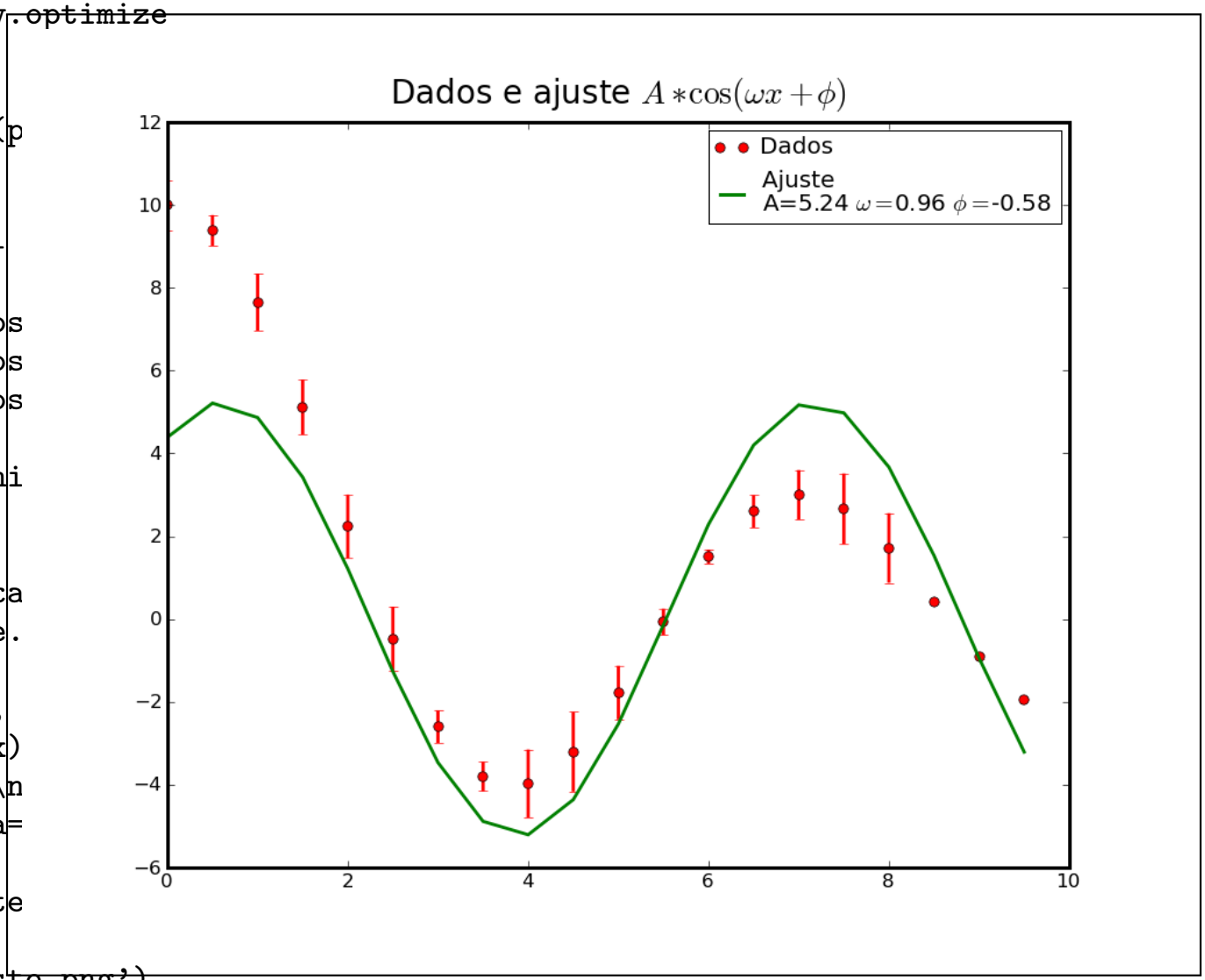
```
...
>>> pylab.imshow(m51[300:500,300:500])        # mostra regiao central, matriz 200x200
>>> m51[300:500,300:500].mean()               # media na regiao
>>> centro = m51[300:500,300:500]             # cria outra matriz
>>> pyfits.writeto('m51centro.fits', centro) # grava arquivo FITS
>>>
>>> pylab.plot(m51[:,250])                    # gráfico vertical x=250, 0<=y<800
```



Exemplo 11

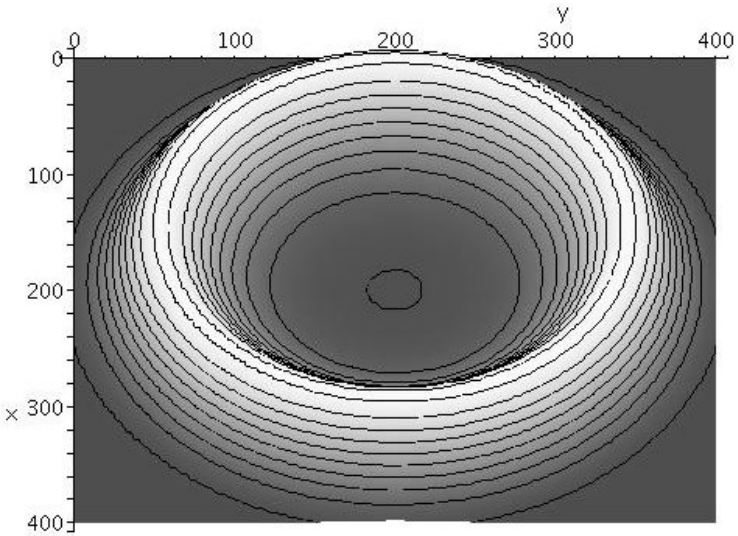
Ajuste de Curvas

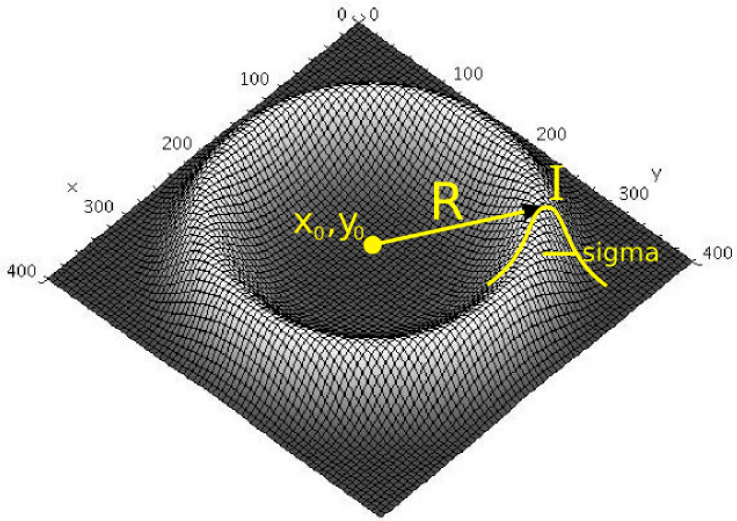
```
import pylab, numpy, scipy.optimize

def fitfunc(p,x):
    return p[0]*numpy.cos(p

def residuo(p,x,y):
    return (fitfunc(p,x) -

x    = numpy.loadtxt('dados
y    = numpy.loadtxt('dados
yerr = numpy.loadtxt('dados

# estimativa parametros ini
p0   = [10,1,0]

# rotina que faz minimizaca
pf,status = scipy.optimize.

pylab.errorbar(x, y, yerr,
pylab.plot(x, fitfunc(pf,x)
           label='Ajuste \n
           'A=%.2f $\omega=

pylab.title('Dados e ajuste
pylab.legend()
pylab.savefig('dados1-ajuste.png')
pylab.show()
```



Aplicações

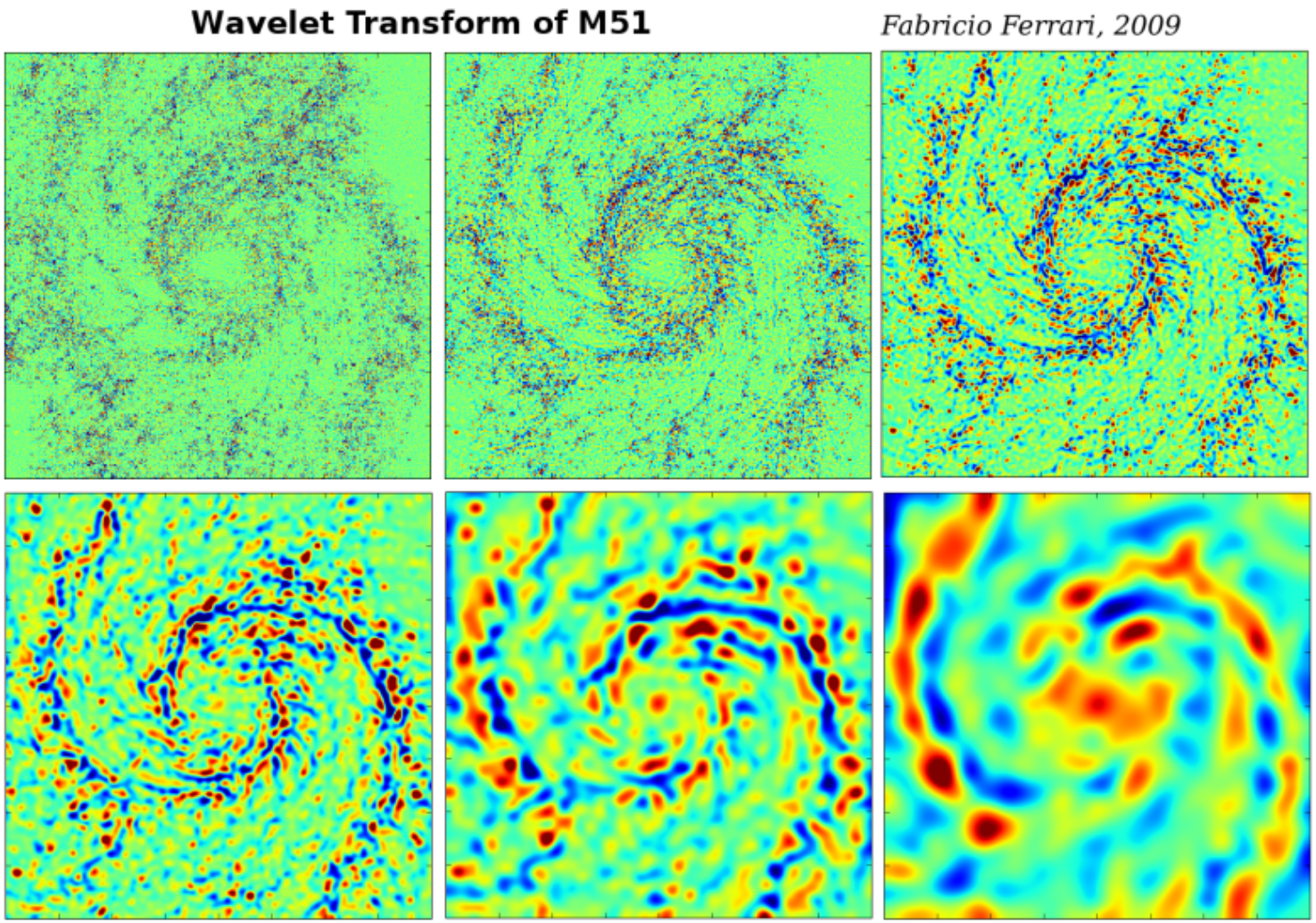
Ajuste de superfícies





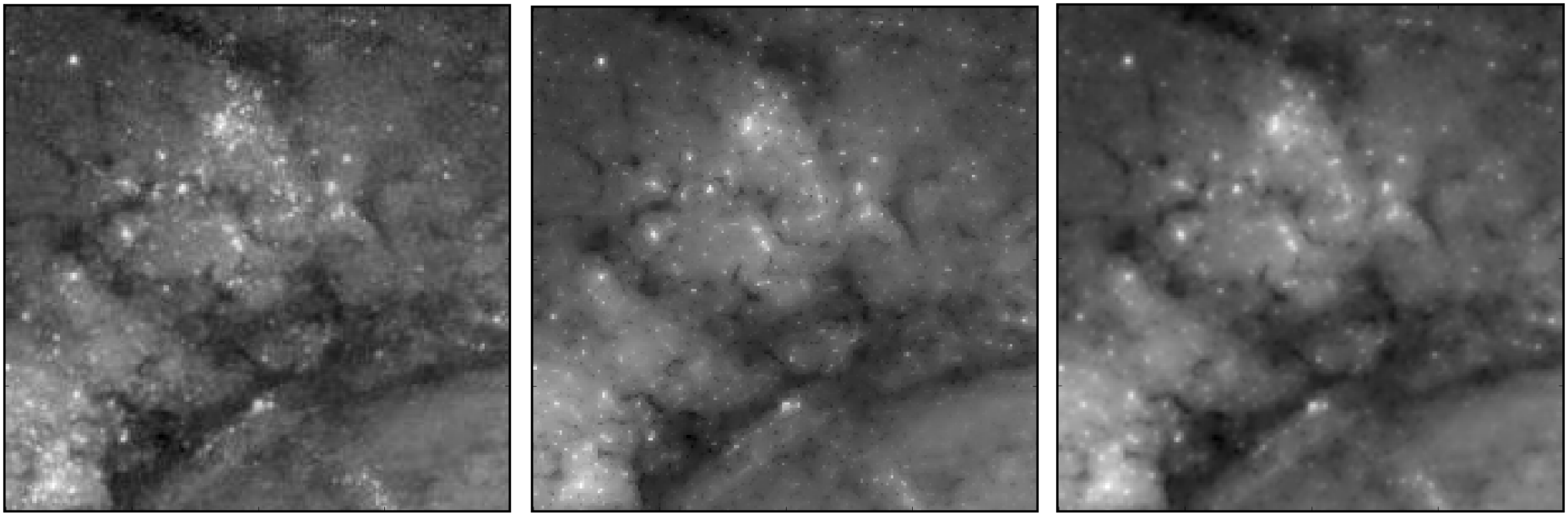
Aplicações

Transformada Discreta de Wavelets



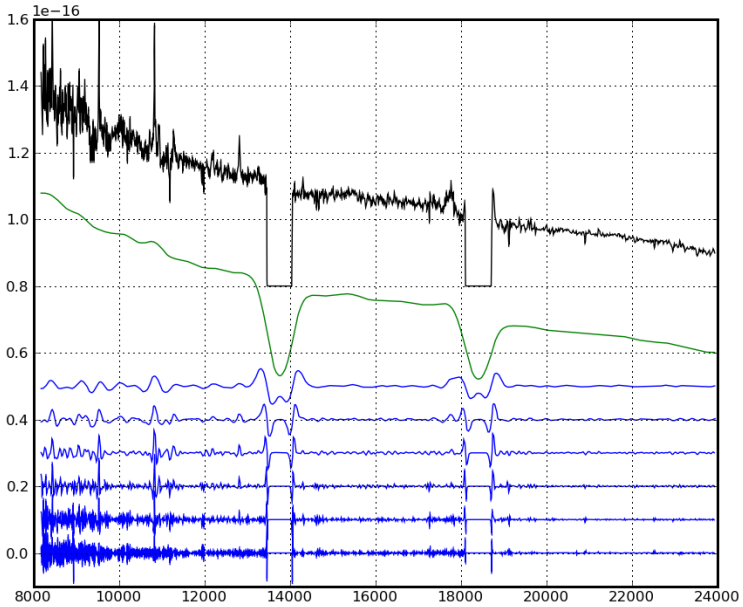
Aplicações

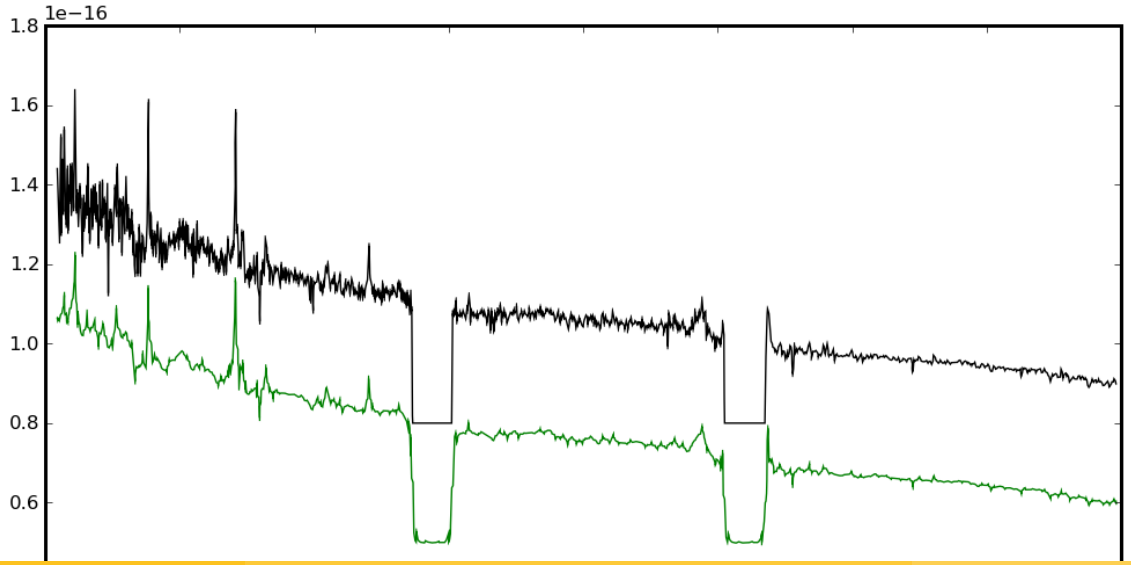
Redução de Ruído em Imagens – filtro adaptativo baseado em Wavelets



Aplicações

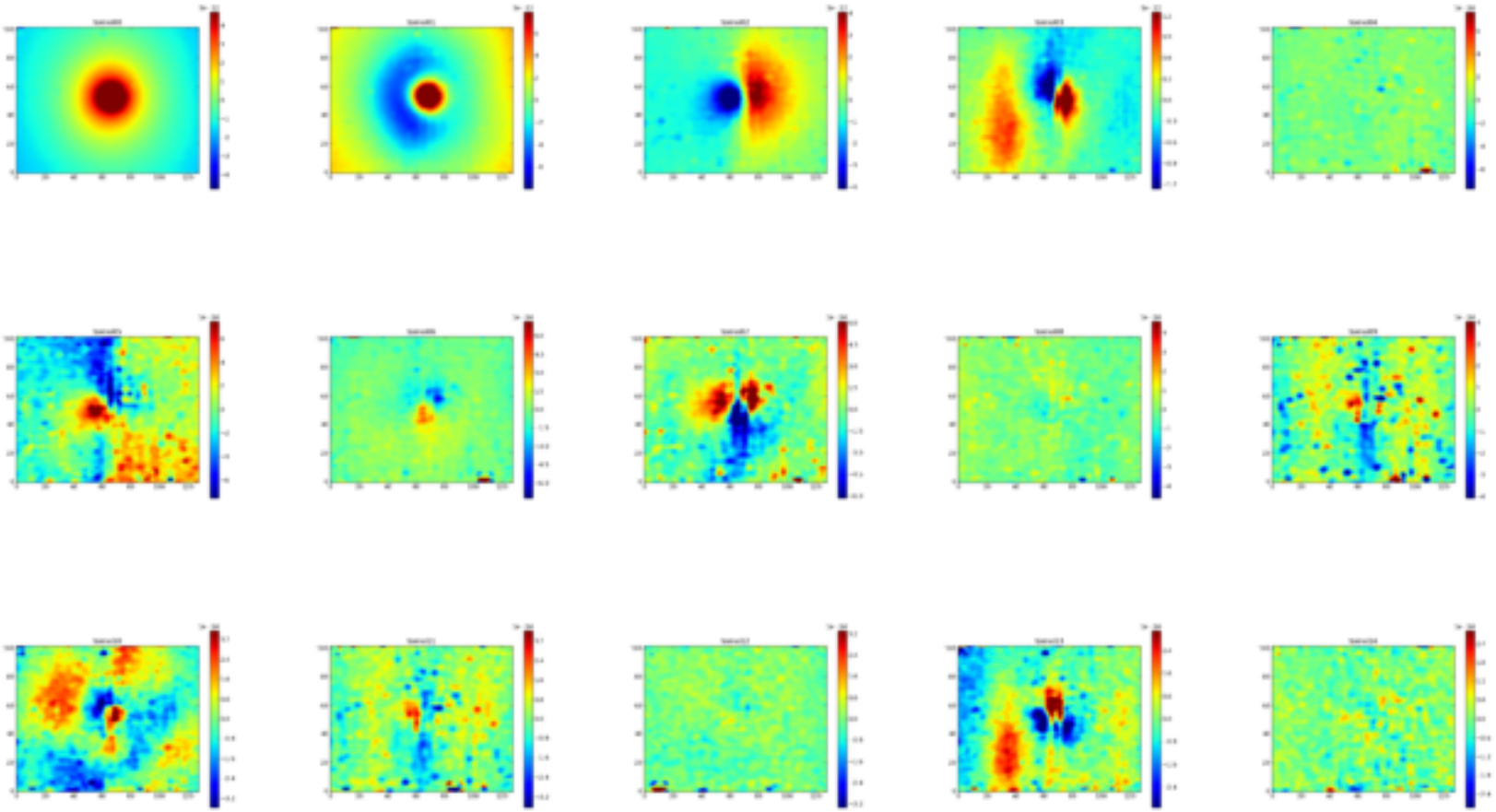
Redução de Ruído em Espectros – filtro adaptativo baseado em Wavelets





Aplicações

Análise de Componente Principal (PCA) em Cubo de Dados



Referências

F.Ferrari	`www.ferrari.pro.br`
Python	`www.python.org`
Python Brasil	`www.python.org.br`
Numpy	`numpy.scipy.org`
Scipy	`www.scipy.org`
Pylab/Matplotlib	`matplotlib.sourceforge.net`
Tutoriais	`http://heather.cs.ucdavis.edu/~matloff/python.html`
Interactive Data Analysis in Astronomy	`http://www.scipy.org/wikis/topical_software/Tutorial`
Wikipedia	www.wikipedia.org