

Departamento de Física Aplicada II

Universidad del País Vasco

eman ta zabal zazu



**Uso de *Python* en el análisis de la
variabilidad climática**

Jesús Fernández,

Jon Sáenz, Juan Zubillaga

Conteúdo

- O que é o Python?
- Pyclimate:
 - Descrição geral
 - Exemplos de uso
- Conclusões
- Alguns *links*

O que é o Python?

- É uma linguagem de programação gratuita e multi-plataforma.
- Está orientada para objectos. Herança múltipla, sobretudo de operadores, polimorfismo ...
- É interpretado (tempo de execução ↑)
- É flexível e de sintaxes muito simples. Os seus programas escrevem-se, depuram-se e lêem-se facilmente. O seu ciclo de desenho é muito rápido (tempo de desenho ↓)
- É fácil de aprender. Um bom começo para ensinar estudantes a programar.
- É extensível com módulos compilados em outras línguas como C, FORTRAN, ... (tempo de execução ↓)
- Possui um módulo (Numerical Python) de cálculo matricial compilado em C que permite diversas precisões numéricas y operações como obtenção de filas, união de matrizes, trasposição de índices, ...

Descrição geral de `pyclimate`

`pyclimate` é um pacote de Python que realiza tarefas comuns na análise da variabilidade climática. Até a data inclui:

- Funções de leitura de ficheiros de dados ASCII como objectos de Numerical Python. Funções para duplicar a estrutura de um ficheiro NetCDF.
- Rotinas para trabalhar com tempos em termos de dias Julianos com uma definição "exacta" de mês. Em particular tem funções para gravar o extrair datas de um ficheiro NetCDF.
- Interface Python para aceder à la livraria DCDFLIB.C
- Análise mediante EOFs e SVD.
- Filtros digitais multivariados. Em concreto do tipo Kolmogorov–Zurbenko e Lanczos.
- Operadores diferenciais em coordenadas esféricas que permitem, por exemplo, o cálculo de ventos geostróficos, vorticidades, ...
- Estimação baseada em *kernel* de funções de densidade de probabilidade (KPDF) para séries univariadas e multivariadas.

Exemplo 1

```
import math
from Scientific.IO.NetCDF import *
from pyclimate.svdeofs import *
from pyclimate.ncstruct import *

inc = NetCDFFile("ncepslp.djf.nc")
slp = inc.variables["djfslp"]
lats = math.pi/180.*inc.variables["lat"][:]
areafactor = sqrt(cos(lats))
slpdata = (slp[:, :, :, :] *
           areafactor[NewAxis, NewAxis, :, NewAxis])
oldshape = slpdata.shape
slpdata.shape = (oldshape[0],
                oldshape[1]*oldshape[2]*oldshape[3])

pcs, lambdas, eofs = svdeofs(slpdata)

ds = ("lat", "lon")
onc = nccopystruct("firstEOF.nc", inc, ds, ds, ds)
eof1 = onc.createVariable("eof1", Float32, ds)
eof1.long_name = "First SLP EOF"
eof1[:, :] = ( reshape(eofs[:, 0], oldshape[2:]) /
              areafactor[:, NewAxis] ).astype(Float32)
onc.close()
```

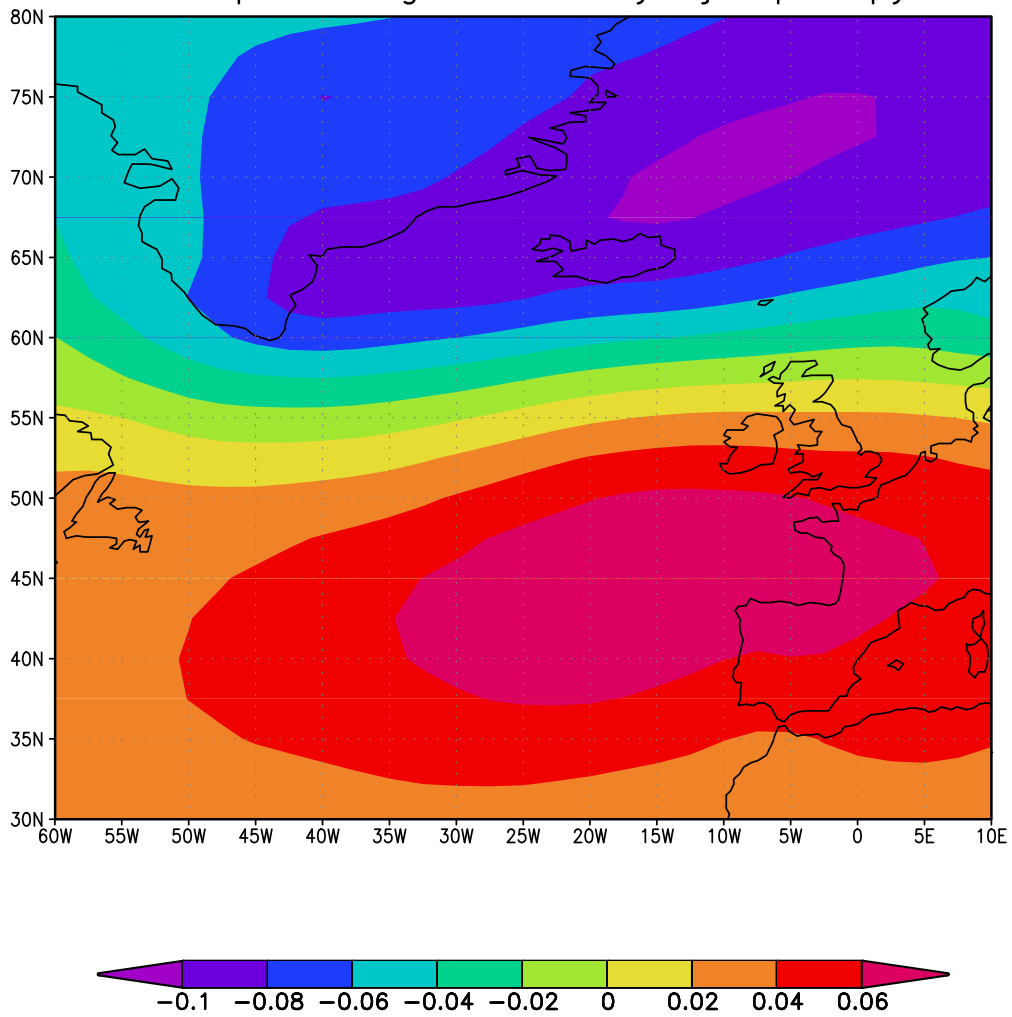
Input file size: 1 Mb

Alpha Workstation 700MHz 190MbRAM: 0.06 sec.

Laptop AMD K6 380MHz 64MbRAM: 0.55 sec.

Pentium III 450MHz 512MbRAM: 0.19 sec.

NAO pattern generated by ejemplo1.py



Exemplo 2

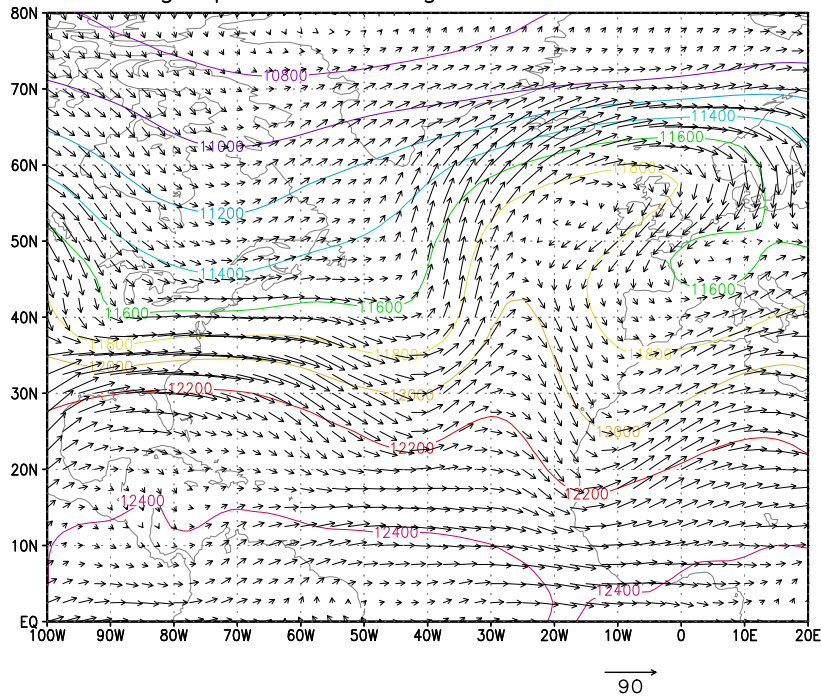
```
from pyclimate.diffoperators import *
from Scientific.IO.NetCDF import *
from pyclimate.ncstruct import *

inc = NetCDFFile("NCARreanal200.nc")
hgt = inc.variables["hgt"]
lats = inc.variables["lat"]
lons = inc.variables["lon"]
hgtdata = hgt[0,:,:]
grad = HGRADIENT(lats[:], lons[:])
geosfact = 1.486e-5 * sin(lats[:] * 0.017453)
geosfact = geosfact[:,NewAxis]*ones(hgtdata.shape)
geosfact = geosfact + equal(geosfact, 0.0)
hg = grad.hgradient(hgtdata)
wind = (-hg[1]/geosfact, hg[0]/geosfact)

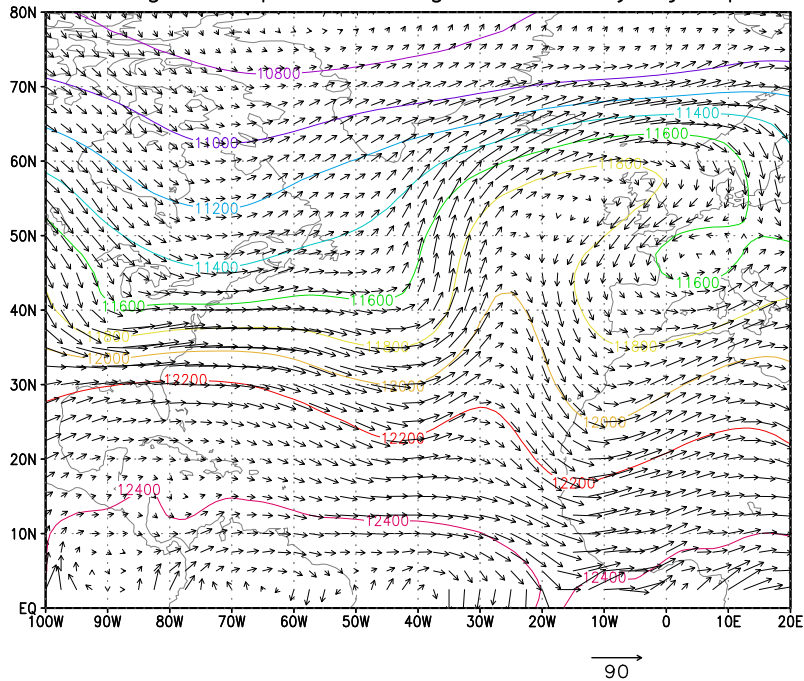
ds = hgt.dimensions
onc = nccopystruct("gwind200.nc", inc, ds, ds, ds)
onc.title = "Geostrophic wind"
gw_u = onc.createVariable("gw_u", Float32, ds)
gw_u.units = "m/s"
gw_v = onc.createVariable("gw_v", Float32, ds)
gw_v.units = "m/s"
gw_u[0,:,:] = wind[0].astype(Float32)
gw_v[0,:,:] = wind[1].astype(Float32)
onc.close()
```

Input file size: 20Kb	40Mb
Alpha Workstat. 700MHz 190MbRAM: 0.02 sec.	2'30"
Laptop AMD K6 380MHz 64MbRAM: 0.3 sec.	3'18"
Pentium III 450MHz 512MbRAM: 0.25 sec.	2'23"

200 mb geopotential height surface and real wind



200 mb geostrophic wind generated by ejemplo2.py



Exemplo 3

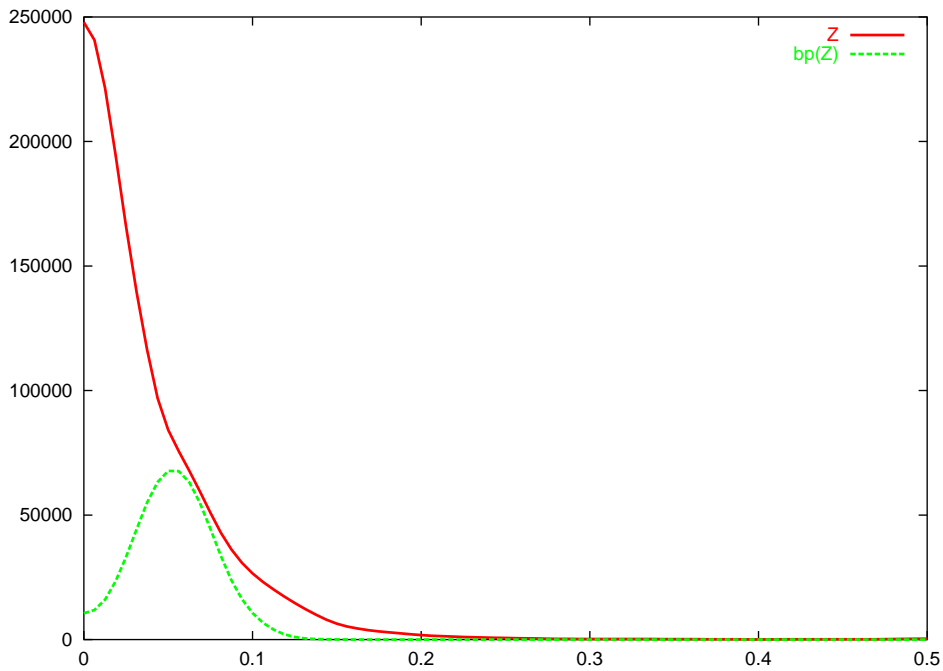
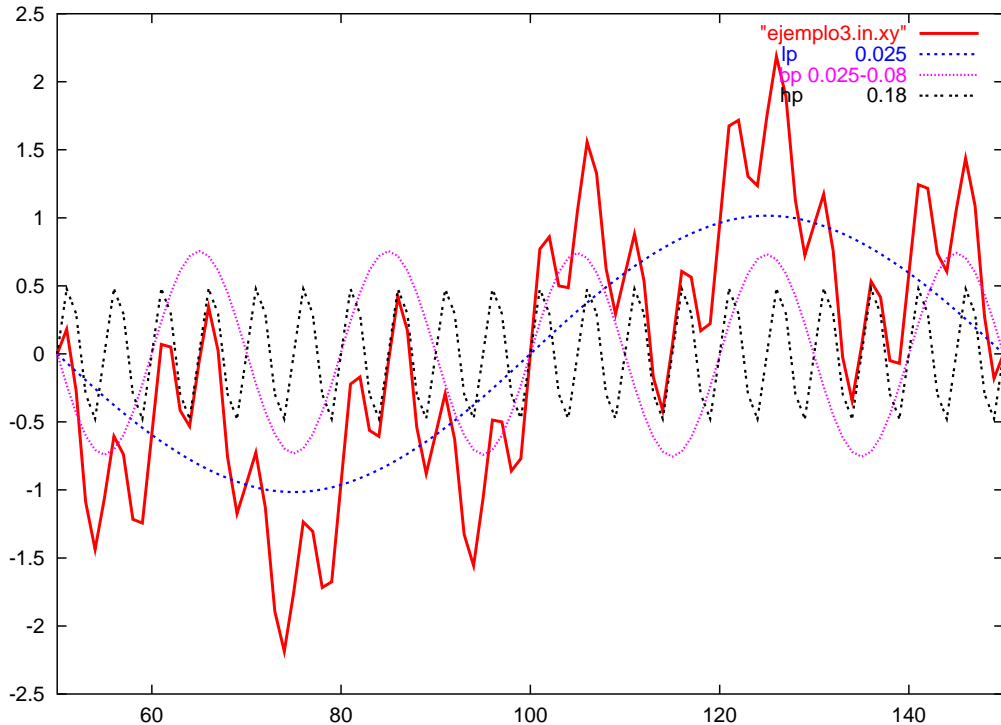
```
import math
from pyclimate.LanczosFilter import *
twopi = 2. * math.pi
def f(x):
    return (
        sin(twopi * 0.01 * x) +
        0.75 * sin(twopi * 0.05 * x) +
        0.50 * sin(twopi * 0.20 * x) )
a = f(arrayrange(1000)) # [f(0) f(1) ... f(999)]
a.shape = (1000,1)
npoints = 50
lp = LanczosFilter('lp', 0.025, 0.025, npoints)
bp = LanczosFilter('bp', 0.025, 0.08, npoints)
hp = LanczosFilter('hp', 0.18, 0.18, npoints)
for i in range(len(a)):
    lfa = lp.getfiltered(a[i])
    bfa = bp.getfiltered(a[i])
    hfa = hp.getfiltered(a[i])
    if lfa:
        print "%d %f %f %f"%(
            i-npoints, lfa[0], bfa[0], hfa[0])
```

Filtering of 1000 points:

Alpha Workstation 700MHz 190MbRAM: 0.5 sec.

Laptop AMD K6 380MHz 64MbRAM: 2.8 sec.

Pentium III 450MHz 512MbRAM: 0.72 sec.



Spectral density associated to the geopotential height at point in the Atlantic storm-track before and after being filtered in the band from 2.5 to 6 days

Conclusões

- Em muitas ocasiões se fazem programas de uso muito específico que se vão executar poucas vezes. Para estas tarefas é conveniente uma linguagem de programação simples que simplifique a tarefa de desenho do programa.
- Python é uma boa eleição se se deseja programar de forma simples e as suas extensões numéricas permitem um uso científico competitivo.
- Apresentaram-se uma série de rotinas que implementam certas funções de uso comum na análise da variabilidade climática a que se pode aceder a partir do Python: `pyclimate`
- `pyclimate` tem licença GPL (i.e: é gratuito) e é extensível.

Links

- **<http://www.python.org>**
Tudo sobre Python.
- **<http://numpy.sourceforge.org>**
Daqui se pode obter a extensão Numerical Python.
- **<http://www.pyclimate.org>**
Tudo sobre pyclimate, incluindo os exemplos vistos durante esta exposição.