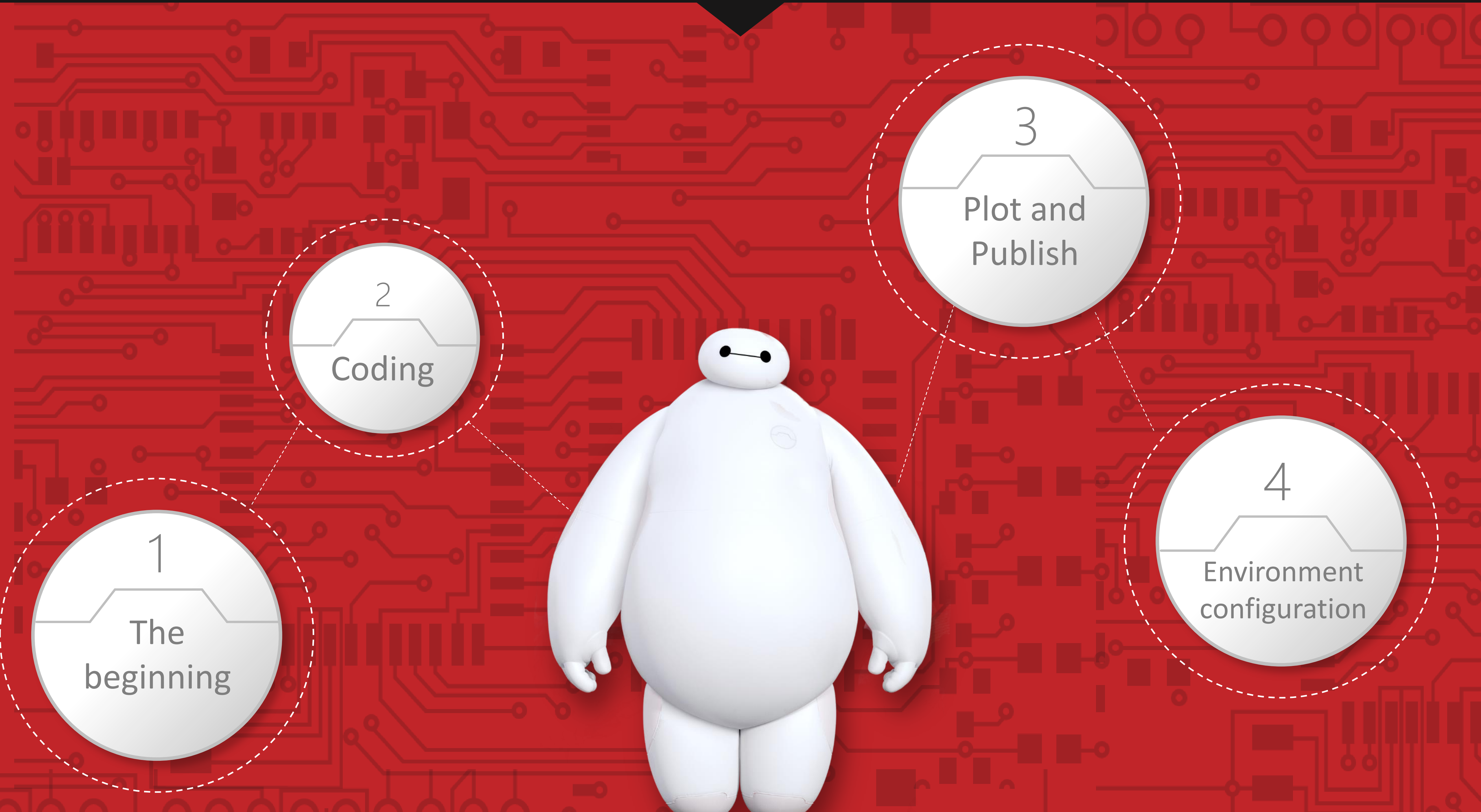
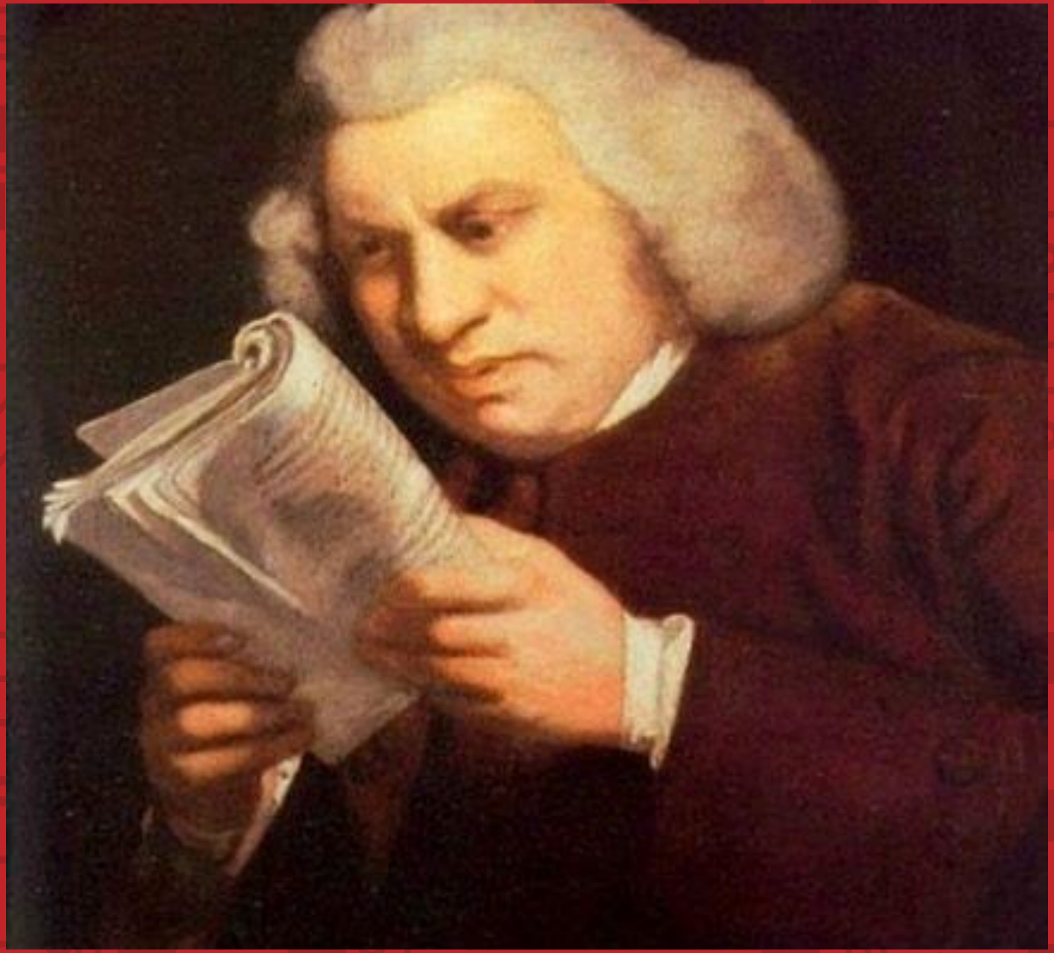
How does Madagascar works in our scientific research.

Bo Wang, Ning Liu, Changle Chen
Jilin University


1
The beginning
2
Coding
3
Plot and Publish
4
Environment configuration

The beginning

1 Part ONE



Reading papers makes me feel like a scientist, until one day……

I have an idea, try to make a program and achieve it.


You know nothing,


YOU KNOW
NOTHING
JON SNOW

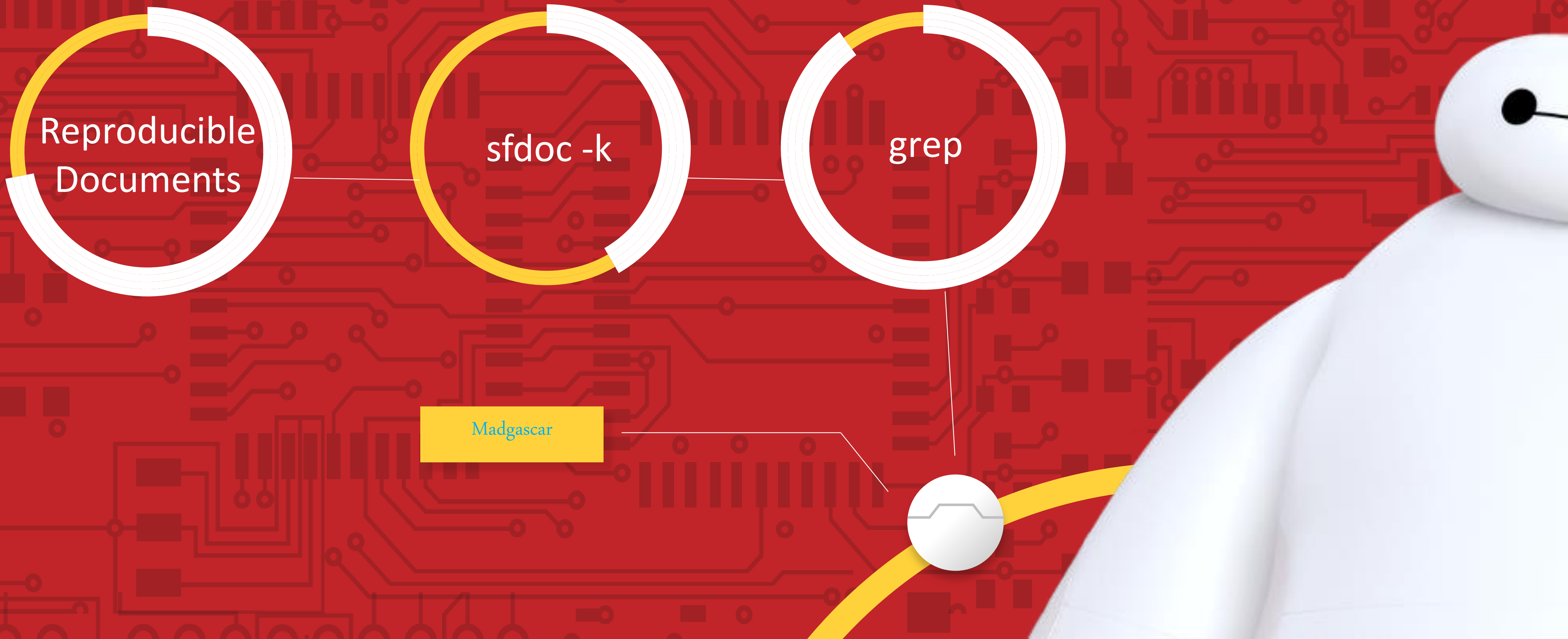

KEEP
CALM
AND
KNOW
NOTHING
@谷大白话


OH...


You know nothing, Jon Snow.


NOTHING.
(i'm jon snow)

These really help me a lot at the first beginning of learning Madagascar, even now.



MADAGASCAR

download
Installation
GitHub repository
SEGTeX

Introduction
- Package overview
- Tutorial
- Hands-on tour
- Reproducible documents

User Documentation
- List of programs
- Common programs
- The RSF file format
- Reproducibility with SCons

Developer documentation
- Adding programs
- Contributing programs
- API demo: clipping data
- API demo: explicit finite differences

Community
- User mailing list
- Developer mailing list

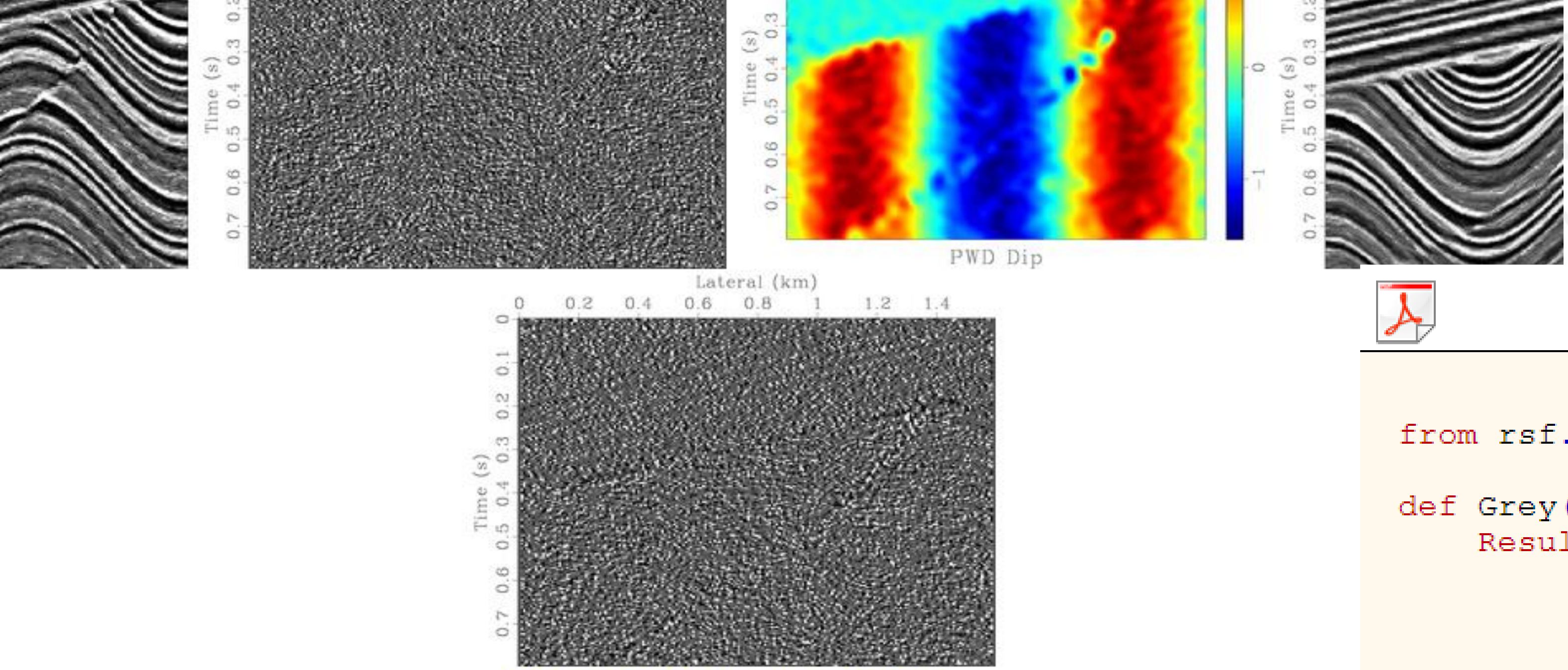
Page Discussion

Reproducible Documents

Contents [hide]

Basic Earth Imaging

- Imaging in shot-geophone space by *Jon F. Claerbout*
- Downward continuation by *Jon F. Claerbout*
- Waves and Fourier sums by *Jon F. Claerbout*
- Zero-offset migration by *Jon F. Claerbout*



elpf, delpf, sdip, median, dmedian

of results using different structure-oriented filtering. Nonstationary polynomial fitting (
polynomial fitting (b), local PWD-based dip (c), median filter (d), and difference profile

```
from rsf.proj import *

def Grey(data,name):
    Result(data,
           '''
           grey title=" %s "
           title=" "
           '''%name)

def Color(data,name):
    Result(data,
           '''
           grey color=j   title=""
           scalebar=y title=" %s "
           '''%name)

##############
# Noise data
##############
rect1=5

Flow('noise',None,
     '''
     sigmoid d1=.004 n1=200 d2=.008 n2=200  taper=no |
     smooth rect1=3 diff1=1 | smooth rect1=3 |
     noise seed=2011 var=4.e-7
     ''')
#########################################
```

```
[wangbo@localhost ~]$ sfdoc -k velocity
```

```
sfanisoSVlr2: Lowrank decomposition for 2-D anisotropic wave propagation using exact SV phase velocity.
sfstandardmodel: build 3D velocity cube for SEAM standard model
sfvelmap: 2-D mapping from moving-object velocity to plane-wave slowness
sfexgr: Exact group velocity in VTI media
sfvelinv: Velocity transform for generating velocity spectra and its corresponding hyperbolic modeling.
sfvel1d: Hungs a 1d velocity function from the Water bottom.
sfbspvel2: B-spline coefficients for a 2-D (an)isotropic velocity model.
sfmodelcreate: Create a dipping layer model for HTI testing purposes.  Has fixed velocity structure, but can change dip of layer and degree of anisotropy.
sfve2d: Convert interval velocity to Dix velocity
sfcconst: Gaussian beam and exact complex eikonal for constant velocity medium
sfvelinvww: Inverse velocity spectrum with interpolation by modeling from inversion result
sfbspvel3: B-spline coefficients for a 3-D (an)isotropic velocity model.
sfchebvc: Post-stack 2-D velocity continuation by Chebyshev-tau method.
sfagmig: Angle-gather constant-velocity time migration.
sfcascade: Velocity partitioning for cascaded migrations.
sfstandardmodel_elastic: build 3D velocity cube for SEAM standard models
sfconstperm: Constant-velocity prestack exploding reflector.
sffourvc2: Velocity continuation with semblance computation.
sffourvc0: Velocity continuation after NMO.
sftdconvert: Iterative time-to-depth velocity conversion
sfdix: Convert RMS to interval velocity using LS and shaping regularization.
sfovc: Oriented velocity continuation.
sfconstfdmig2: 2-D implicit finite-difference migration in constant velocity.
sfvidattr: Slope-based velocity-independent data attributes.
sfitxmo: Forward and inverse normal moveout with interval velocity.
sfaclfEPlr2: Lowrank decomposition for 2-D anisotropic wave propagation using exact P phase velocity.
sfpveltran3: Slope-based tau-p 3D velocity transform for elliptical anisotropy.
sfmakevel: Make a velocity function v(x,y,z)
sfshotconstkirch: Prestack shot-profile Kirchhoff migration in constant velocity.
sfaclfSVlr2: Lowrank decomposition for 2-D anisotropic wave propagation using exact P phase velocity.
sfvconvert: 2-D velocity mapping from manual picking to rsf RMS format.
sftxpscan: Velocity analysis using T-X-P domain.
sfunif3: Generate 3-D layered velocity model from specified interfaces.
sfpwdix: Convert RMS to interval velocity using LS and plane-wave construction.
sfvelcon: Post-stack velocity continuation by implicit finite differences
sfdixshape: Convert RMS to interval velocity using LS and shaping regularization.
```

```
[wangbo@localhost ~]$ grep kiss_fft_cpx /home/wangbo/geotools/RSFSRC/user/yliu/*.c
```

```
/home/wangbo/geotools/RSFSRC/user/yliu/freqfilt2.c:static kiss_fft_cpx *ctrace, *ctrace2, **fft;
/home/wangbo/geotools/RSFSRC/user/yliu/freqfilt2.c:    ctrace = (kiss_fft_cpx*) sf_complexalloc(nw);
/home/wangbo/geotools/RSFSRC/user/yliu/freqfilt2.c:    ctrace2 = (kiss_fft_cpx*) sf_complexalloc(n2);
/home/wangbo/geotools/RSFSRC/user/yliu/freqfilt2.c:    fft = (kiss_fft_cpx**) sf_complexalloc2(nw,n2);
/home/wangbo/geotools/RSFSRC/user/yliu/Mfourbreg2.c:    kiss_fft_cpx **mm, ce, **fft, *ctrace, *ctrace2;
/home/wangbo/geotools/RSFSRC/user/yliu/Mfourbreg2.c:    ctrace = (kiss_fft_cpx*) sf_complexalloc (nw);
/home/wangbo/geotools/RSFSRC/user/yliu/Mfourbreg2.c:    ctrace2 = (kiss_fft_cpx*) sf_complexalloc (nk);
/home/wangbo/geotools/RSFSRC/user/yliu/Mfourbreg2.c:    mm = (kiss_fft_cpx**) sf_complexalloc2(nw,nk);
/home/wangbo/geotools/RSFSRC/user/yliu/Mfourbreg2.c:    fft = (kiss_fft_cpx**) sf_complexalloc2(nw,nk);
/home/wangbo/geotools/RSFSRC/user/yliu/Mfourmis2.c:    kiss_fft_cpx **mm, ce, **fft, *ctrace, *ctrace2;
/home/wangbo/geotools/RSFSRC/user/yliu/Mfourmis2.c:    ctrace = (kiss_fft_cpx*) sf_complexalloc (nw);
/home/wangbo/geotools/RSFSRC/user/yliu/Mfourmis2.c:    ctrace2 = (kiss_fft_cpx*) sf_complexalloc (nk);
/home/wangbo/geotools/RSFSRC/user/yliu/Mfourmis2.c:    mm = (kiss_fft_cpx**) sf_complexalloc2(nw,nk);
/home/wangbo/geotools/RSFSRC/user/yliu/Mfourmis2.c:    fft = (kiss_fft_cpx**) sf_complexalloc2(nw,nk);
/home/wangbo/geotools/RSFSRC/user/yliu/Mstft.c:    kiss_fft_cpx *pp, ce;
/home/wangbo/geotools/RSFSRC/user/yliu/Mstft.c:    pp = (kiss_fft_cpx*) sf_complexalloc(nw);
/home/wangbo/geotools/RSFSRC/user/yliu/radonoper.c:            kiss_fftr(forw,tt, (kiss_fft_cpx *) cd[ix]);
/home/wangbo/geotools/RSFSRC/user/yliu/radonoper.c:            kiss_fftr(forw,tt, (kiss_fft_cpx *) cm[ip]);
/home/wangbo/geotools/RSFSRC/user/yliu/radonoper.c:            kiss_fftri(invs,(const kiss_fft_cpx *) cm[ip],
/home/wangbo/geotools/RSFSRC/user/yliu/radonoper.c:            kiss_fftri(invs,(const kiss_fft_cpx *) cd[ix],
/home/wangbo/geotools/RSFSRC/user/yliu/st.c:    kiss_fft_cpx *d, *pp, *qq;
/home/wangbo/geotools/RSFSRC/user/yliu/st.c:    pp = (kiss_fft_cpx*) sf_complexalloc(nw);
/home/wangbo/geotools/RSFSRC/user/yliu/st.c:    qq = (kiss_fft_cpx*) sf_complexalloc(nw);
/home/wangbo/geotools/RSFSRC/user/yliu/st.c:    d =  (kiss_fft_cpx*) sf_complexalloc(nw);
/home/wangbo/geotools/RSFSRC/user/yliu/st.c:    kiss_fft_cpx *d, *pp;
/home/wangbo/geotools/RSFSRC/user/yliu/st.c:    pp = (kiss_fft_cpx*) sf_complexalloc(nw);
/home/wangbo/geotools/RSFSRC/user/yliu/st.c:    d =  (kiss_fft_cpx*) sf_complexalloc(nw);
[wangbo@localhost ~]$
```

"sfdoc -k" helps me find the programs with the specific Keywords.

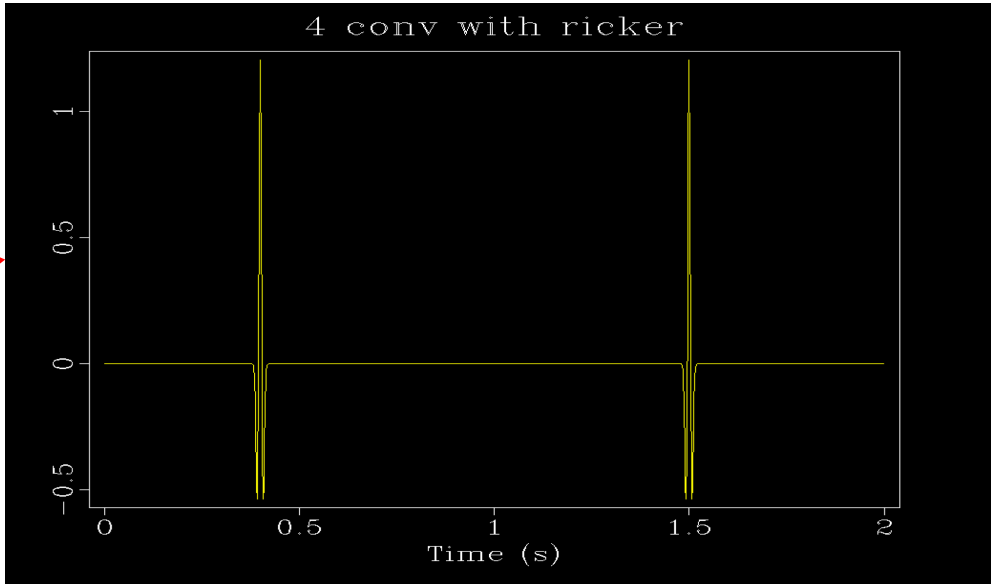
"grep" command helps me find the specific statement (function) in the programs.

Part TWO

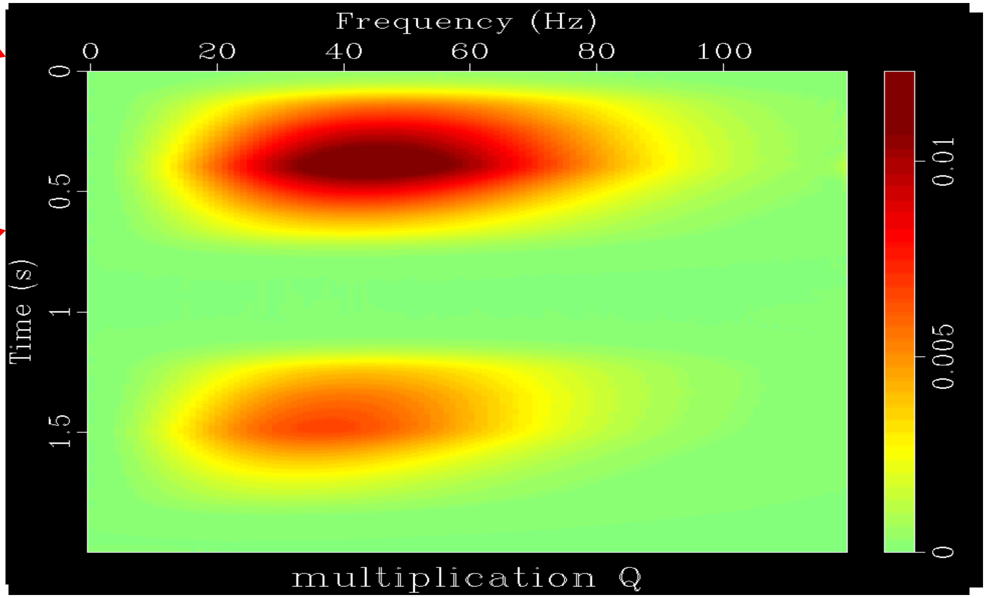
2

Oh, Coding.

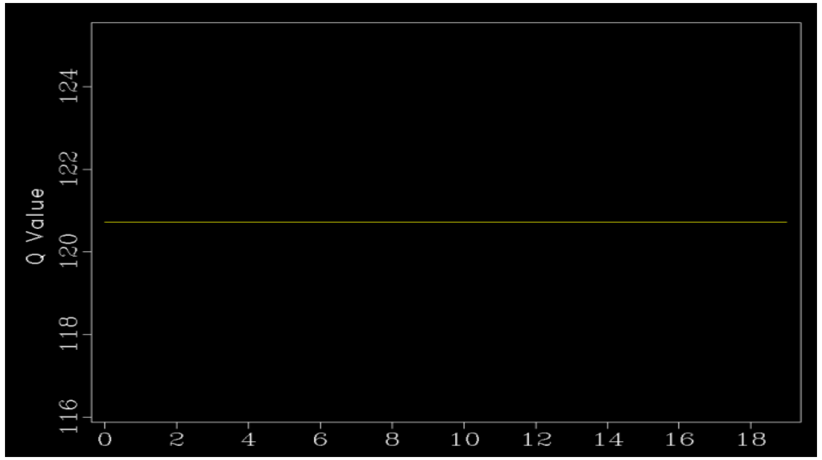
```
from rsf.proj import *
import math

##convolution of 4 ricker
Flow('f',None,
     '''
     spike n1=2000 d1=0.001 o1=0 k1=400,1500  nsp=2  mag=10  p1=2 |
     ricker1 frequency=%d |put n2=1
     '''%50)
Result('f','graph title="4 conv with ricker"')
## LTFT
Flow('ltftf','f',
     '''
     ltft rect=40 verb=n nw=120 dw=1 niter=50 | math output="abs(input)" | real
     ''')
Result('ltftf',
     '''

       grey color=j flat=n title="LTFT of f" scalebar=y
       ''')

## Q1=120
Flow('eattu1',None,
     '''
     math  n3=1 n1=1000 o1=0 d1=0.001 n2=120 d2=1 o2=0
     output="exp(-%g*x1*x2*0.008333)"
     '''%(math.pi))

Result('eattu1','eattu1','grey color=j scalebar=y title="exp attenuation Q=120"')
## Q2=60
Flow('eattu2',None,
     '''
     math  n3=1 n1=1000 o1=0 d1=0.001 n2=120 d2=1 o2=0
     output="exp(-%g*x1*x2*0.0166667)"
     '''%(math.pi))

Result('eattu2','eattu2','grey color=j scalebar=y title="exp attenuation Q=60"')

Flow('eattu','eattu1 eattu2',
     '''
     cat ${SOURCES[1:2]} axis=1
     | smooth repeat=1 adj=n rect1=100 diff1=200
     ''' )

Result('eattu','eattu','grey color=j scalebar=y title="exp attenuation"')
```

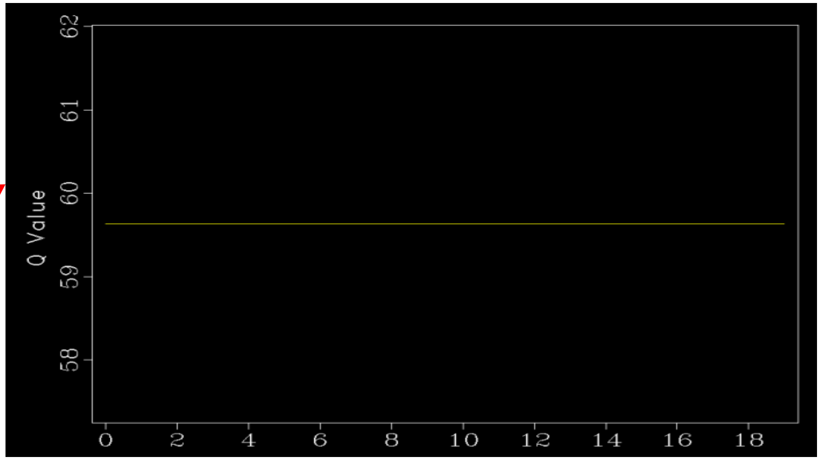


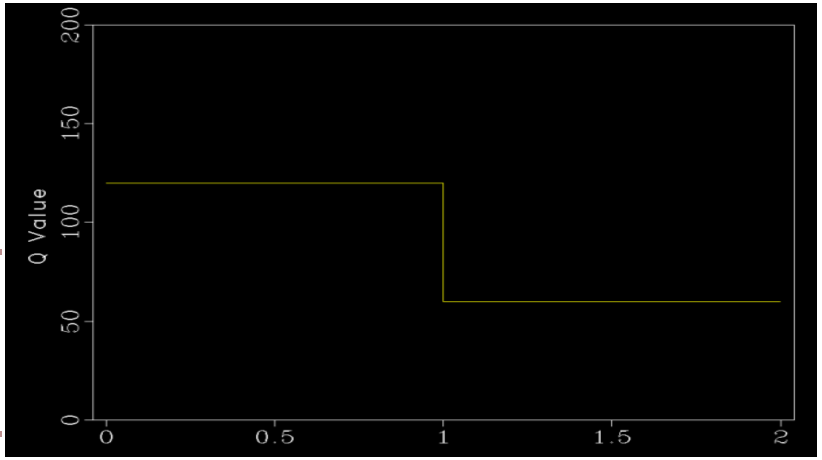
Synthetic model



```
######## parameter test1 ###################
f1=300
f2=600
dt1=f2-f1

f3=1400
f4=1401
dt2=f4-f3

Flow(’tslice1’,’mult’,’ window n1=1 f1=%d ’%(f1))
Plot(’tslice1’,’graph   title=" " min2=0 max2=0.015  scalebar=y color=j’)
Result(’tslice1’,’graph  title=" "  scalebar=y color=j’)

Flow(’tslice2’,’mult’,’ window n1=1 f1=%d ’%(f2))
Plot(’tslice2’,’tslice2’,’graph min2=0 max2=0.015  title=" "  scalebar=y color=j’)
Result(’tslice2’,’graph   title=" "  scalebar=y color=j’)

Result(’cp1’,’tslice1 tslice2’,’Overlay’)

Flow(’tslice3’,’mult’,’ window n1=1 f1=%d ’%(f3))
Plot(’tslice3’,’graph   title=" "  scalebar=y color=j’)
Result(’tslice3’,’graph  title=" "  scalebar=y color=j’)

Flow(’tslice4’,’mult’,’ window n1=1 f1=%d ’%(f4))
Plot(’tslice4’,’tslice4’,’graph   title=" "  scalebar=y color=j’)
Result(’tslice4’,’graph   title=" "  scalebar=y color=j’)

Result(’cp2’,’tslice3 tslice4’,’Overlay’)


Flow(’diva’,’tslice1 tslice2’,
     ’’’math  ref=${SOURCES[1]} output="log(ref/(input))"
     ’’’)
Flow(’divb’,’tslice3 tslice4’,
     ’’’math  ref=${SOURCES[1]} output="log(ref/(input))"
     ’’’)

Flow(’Qdsa’,’diva’,
     ’’’
     window n1=20 f1=30 |
     linefit1 | math output="-%g*%d*0.001/input"
     ’’’%(math.pi,dt1))
Result(’Qdsa’,’graph title=" "   label1="\F2   " label2="\F2 Q Value "  unit1="" unit2=  "

Flow(’Qdsb’,’divb’,
     ’’’
     window n1=20 f1=30 |
     linefit1 | math output="-%g*%d*0.001/input"
     ’’’%(math.pi,dt2))
Result(’Qdsb’,’graph title=" "   label1="\F2   " label2="\F2 Q Value   " unit1=""  unit2="
```







The Netcdf data format

Receiver_Data_Format_OHMnc_1.3

This document contains a description of the CSEMI receiver data format OHMnc_1.3. OHMnc_1.3 uses the binary netCDF classic file format version 1 to ensure a high level of portability. For more information go to:

http://www.unidata.ucar.edu/software/netcdf/docs/netcdf/File-Format-Specification.html#File-Format-Specification

The binary files for the receivers are generically named as follows:

- **Xyy_0.nc** (for the electric field horizontal channel 0)
- **Xyy_1.nc** (for the electric field horizontal channel 1)

and additionally if present:

- **Xyy_3.nc** (for the magnetic field horizontal channel 3)
- **Xyy_4.nc** (for the magnetic field horizontal channel 4).

where: **X** = The first letter of the Survey Project Name.
yy = Specific Receiver ID (01, 02, etc.).

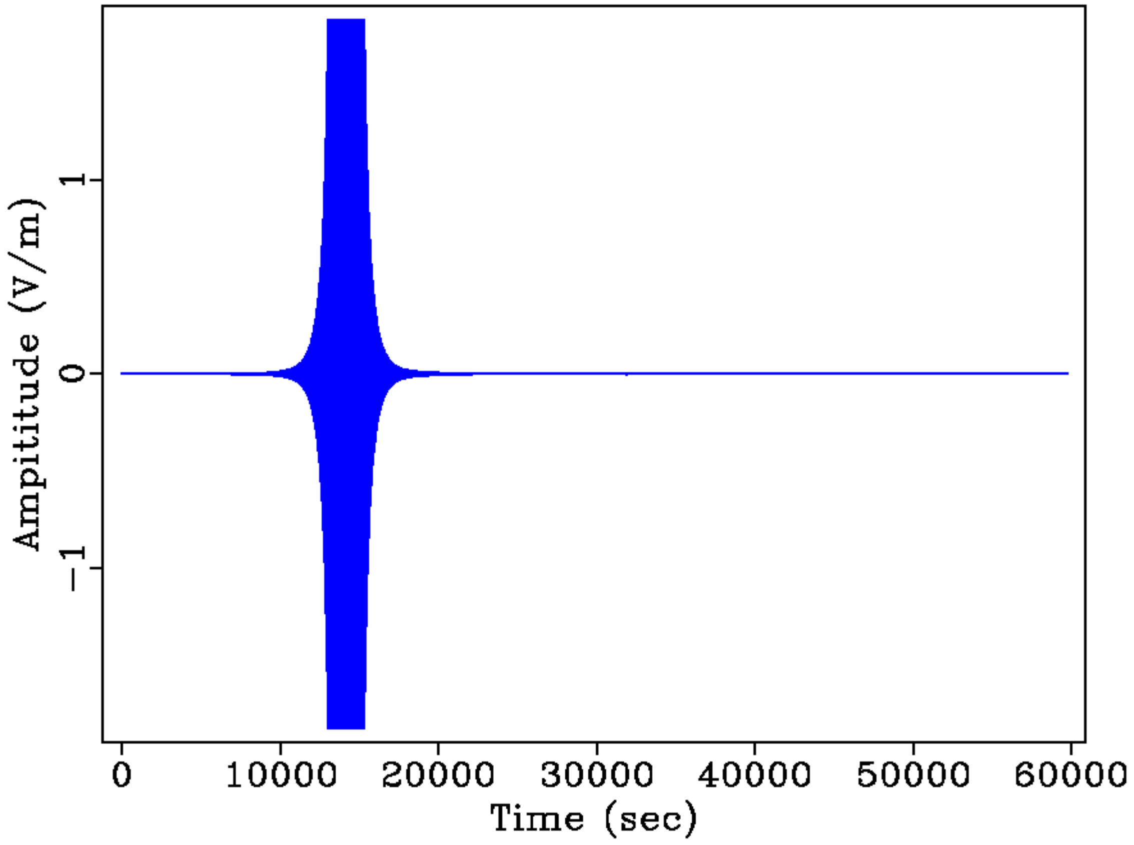
```c
/* A module for read netCDF data of CSEM from OHM Ltd
                Jilin University                    */
//#include <stdafx.h>
#include <stdlib.h>
#include <stdio.h>
#include "netcdf.h"
#include <math.h>

/* This is the name of the data file we will read and write. */
#define FILE_NAME "a02_0.nc"
#define OUTPUT_NAME "data-"FILE_NAME".dat"

/* We are reading 2D data, a 6 x 12 grid. */
//#define NX 71930980
//84199071

/* Handle errors by printing an error message and exiting with a
 * non-zero status. */

#define ERRCODE 2
#define ERR(e) {printf("Error: %s\n", nc_strerror(e)); exit(ERRCODE);}

int main()
{
    /* This will be the netCDF ID for the file and data variable. */
    int ncid, varid, dimid, NX, i, ex;
    double temp;

    size_t dimlen;
    FILE *stream;

    /* Loop indexes, and error handling. */
    int retval;
```

SConstruct - emacs@localhost.localdomain

File Edit Options Buffers Tools IM-Python Python Help

```python
from rsf.proj import *
import math


## Ex data
###d1=9.042e-8 day
Flow('Ex','data-a01_0.nc.dat',
     '''
     echo in=$SOURCE n1=84199071 data_format=native_float
     |window  f1=47550208 n1=7644160 |
     put d1=1 o1=0
     ''')
Result('Ex','graph title="CSEM Time Series"  label1=time unit1=sec')
```

Transform the nc data to binary format by VC++6.0.



Timeserise of MCSEM data

Although Madagascar provides plenty programs to use, but we do need to make some programs our own.

```
wangbo@ubuntu: ~

NAME
        sfduffing
DESCRIPTION
        Duffing differential equation solved by 4th order Runge-Kutta method.
SYNOPSIS
        sfduffing < in.rsf > out.rsf sfile=sfile.rsf gamma=0.75 omega=1 kxi=1 x0=0 y0=0 pow1=1 pow2=3 verb=n
 ricker=n
COMMENTS
        Duffing equation: x'' + 0.5 x' - x + x^3 = gamma cos(omega t) + kxi input(t)

PARAMETERS
        float    gamma=0.75      strength of external force
        float    kxi=1   adjustment for input signal
        float    omega=1         angular frequence of external force
        int      pow1=1  power of first non-linear restitution term
        int      pow2=3  power of second non-linear restitution term
        bool     ricker=n [y/n]  if y need extenal input for external force
        string   sfile=  auxiliary input file name
        bool     verb=n [y/n]    verbosity flag
        float    x0=0    initial value of x0
        float    y0=0    initial value of y0
SOURCE
        user/yliu/Mduffing.c
VERSION
        1.7-svn Mduffing.c 11616 2014-01-17 08:31:05Z yang_liu
(END)
```

Duffing equation:

$$\frac{1}{\omega^2}\ddot{x}(t)+\frac{0.5}{\omega}\dot{x}(t)-x(t)+x^3(t)=\gamma\cos(\omega t)$$

The state equation form of Duffing equation :

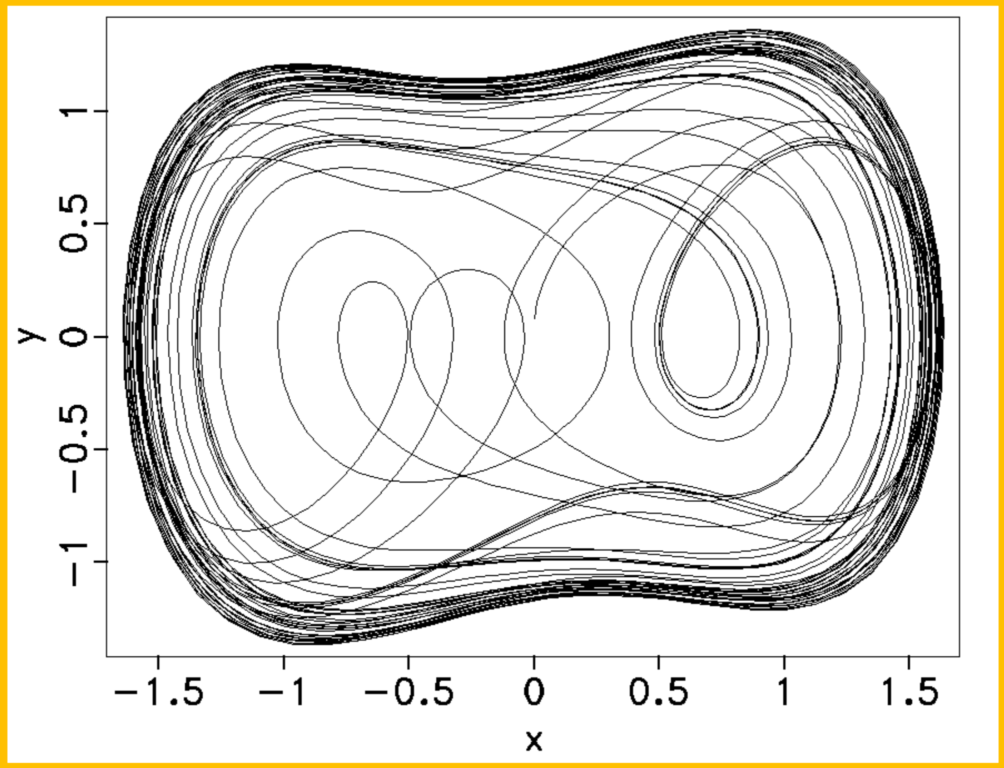
$$\begin{cases}\dot{x}=\omega y\\ \dot{y}=\omega(x-x^3-ky+\gamma\cos(\omega t))\end{cases}$$

Solve the Duffing equation by fourth-order Runge-Kutta algorithm and obtaining pairs x(n+1), y(n+1)at each time step.

That means we need determine each single point by pairs of x(t) and y(t) at each t step.

So we set x(t) and y(t) as real part and image part of a complex number. Then we got the phase plane diagram of Duffing system.

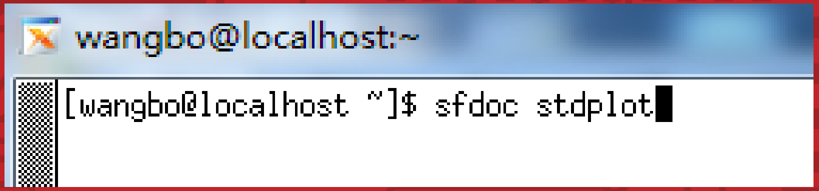
```
Amplitude" plotcol+gamma*stochasforce(i1 n m dt omega ricker)
     max2=1 min2=-1 min1=0 max1=1
     ''')
Flow('coskxi',None,'math n1=2000 d1=0.001 output="cos(100*x1+0.5*3.1415926)"')
Plot('coskxi','math output=input*0.01|graph title= plotcol=5 dash=3 label2
="kxi*R(t) Amplitude" whereylabel=right  min1=0 max1=1')

Flow('cosphadf05','coskxi','duffing gamma=0.824 kxi=0.01 x0=0 y0=0 omega=100
phi=0.  verb=n')

Result('cosphadf05',
       '''
       graph  title="phase difference 0.5" label1="x" label2="y"
       ''')

Result('phasedif0','cos1 coskxi','Overlay')
Result('cosphadf0',
       '''
```

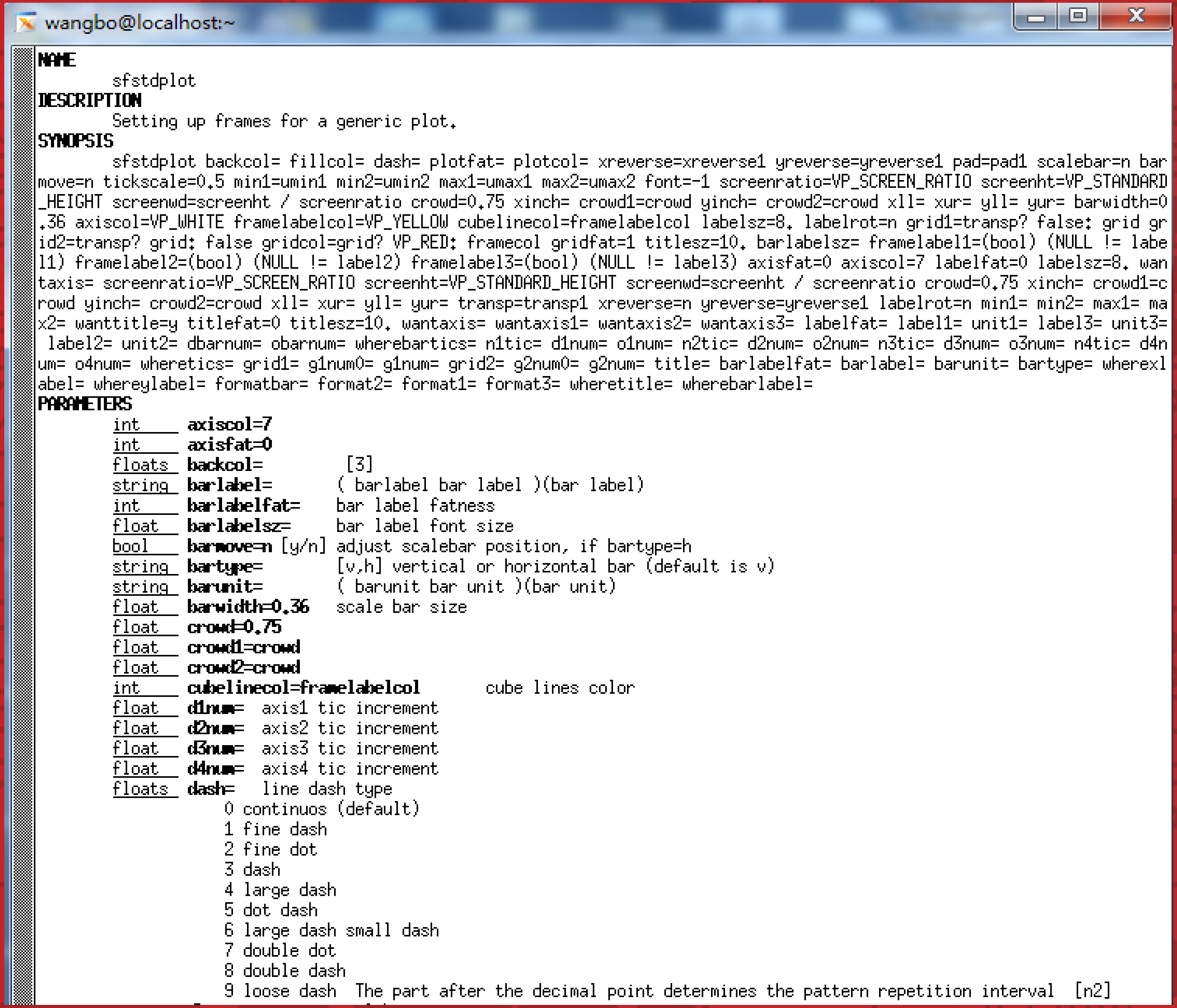


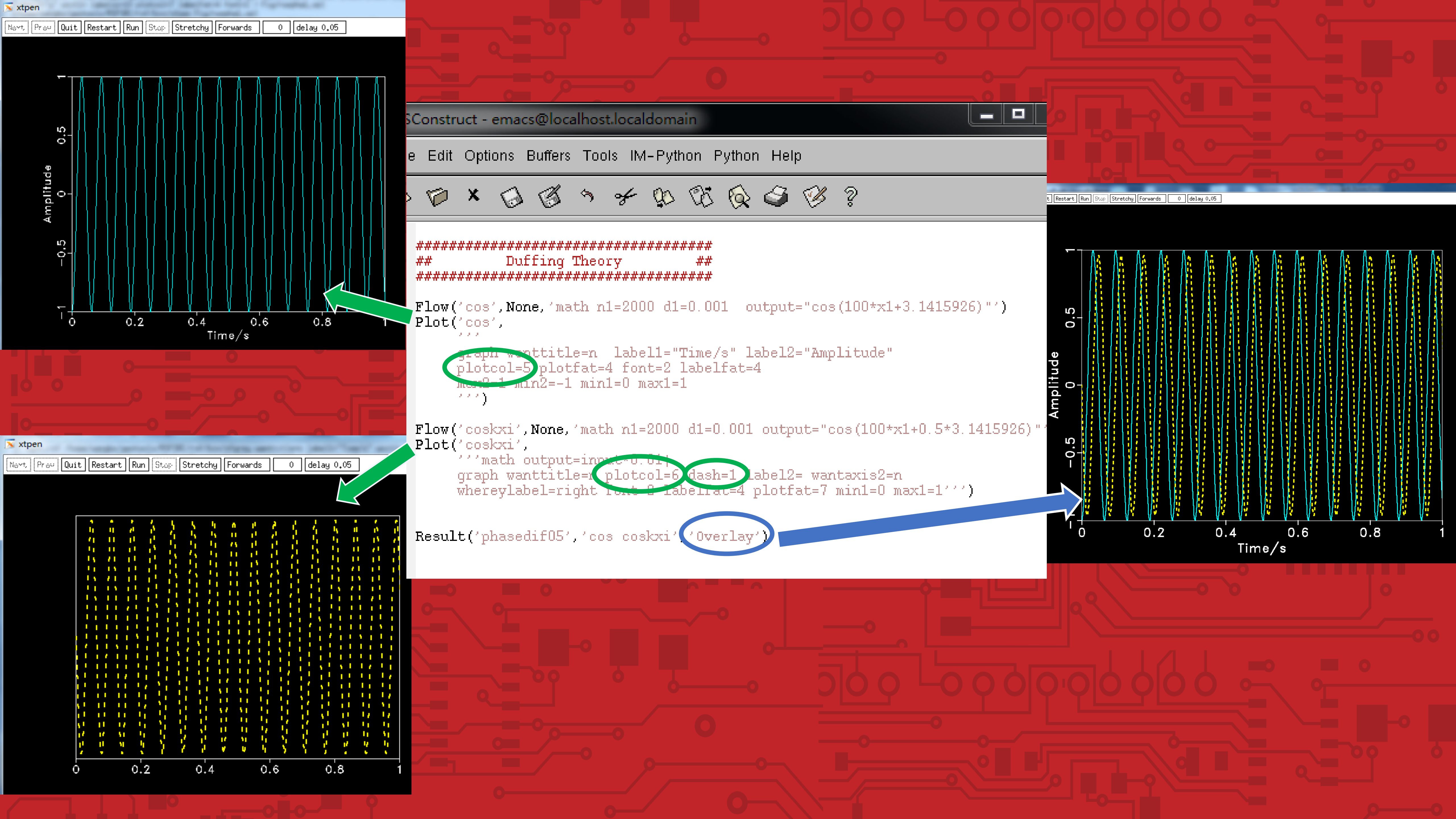
Part
THREE
3

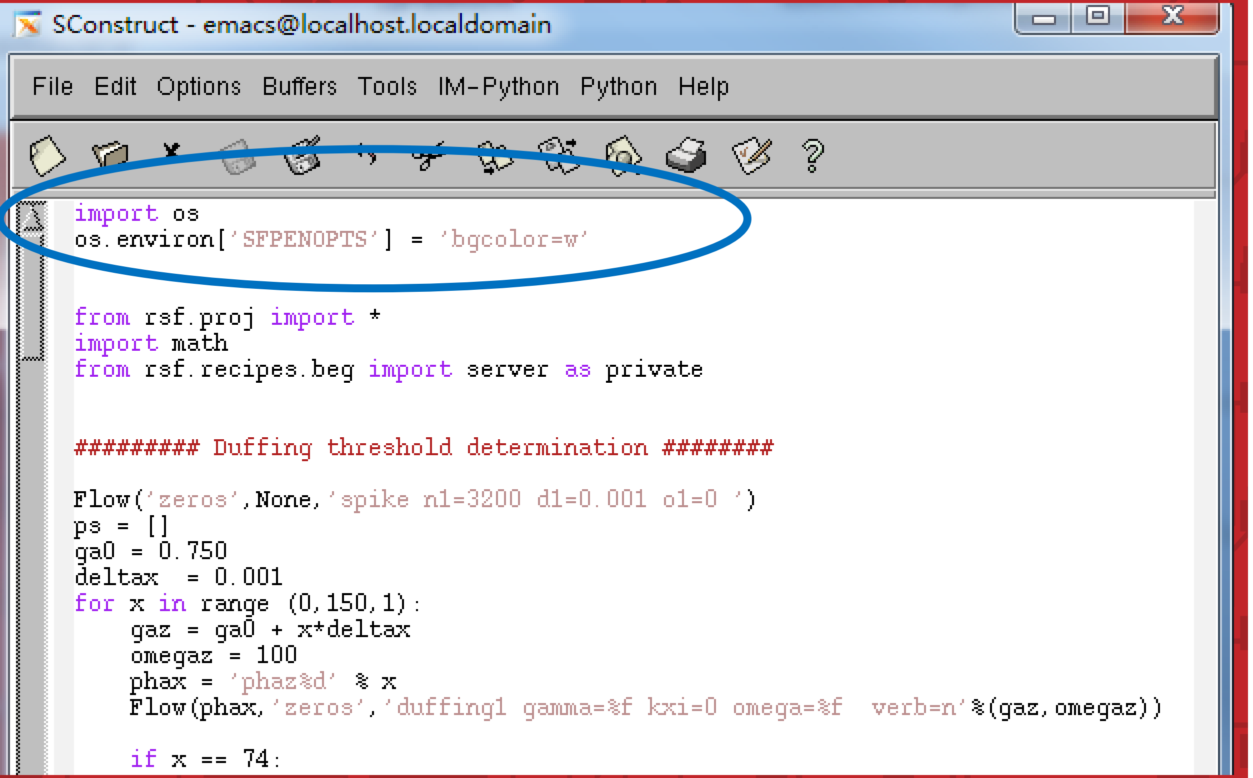
Plot and
Publish.

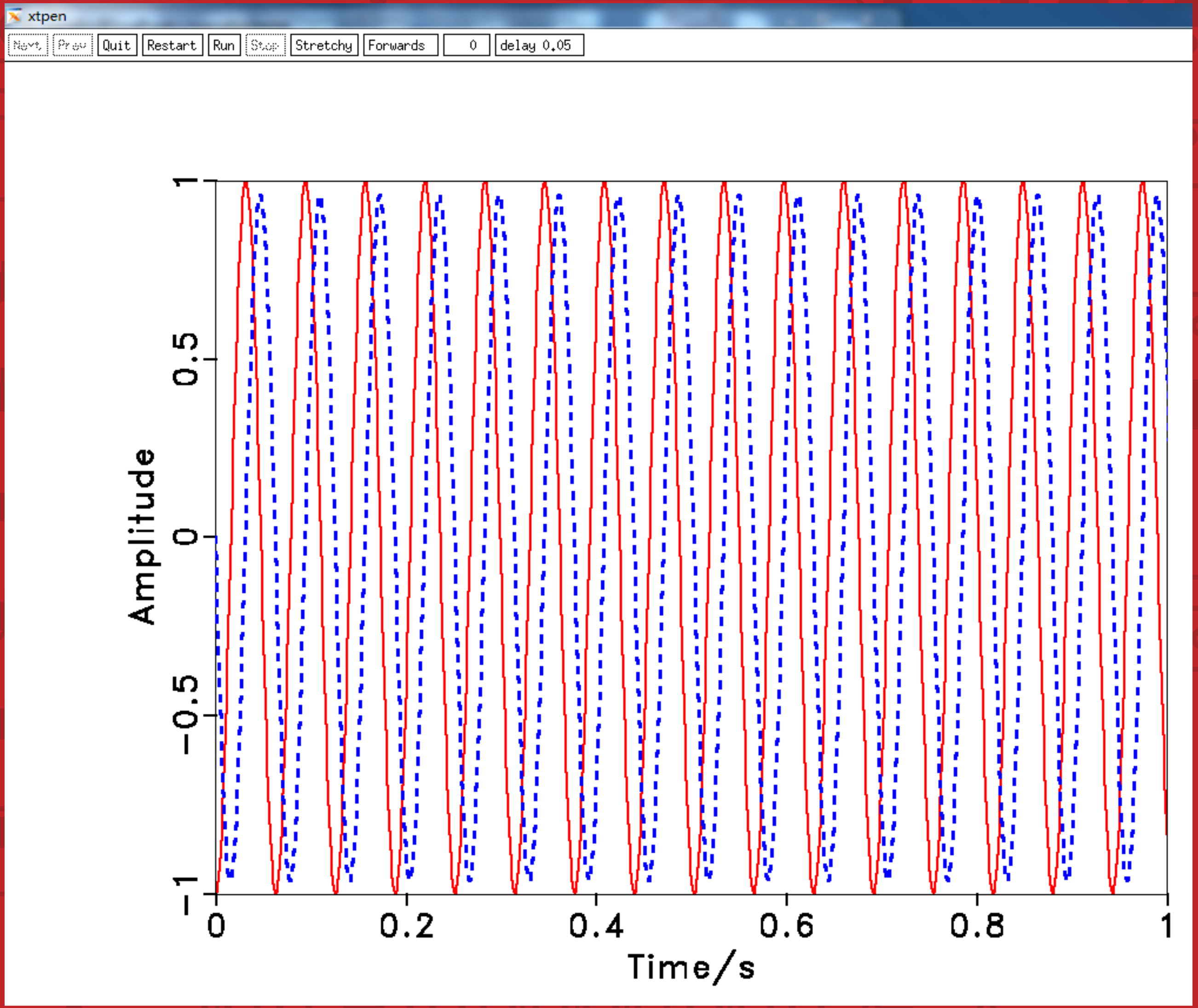
The command “sfdoc stdplot” shows the parameters for generic plot.



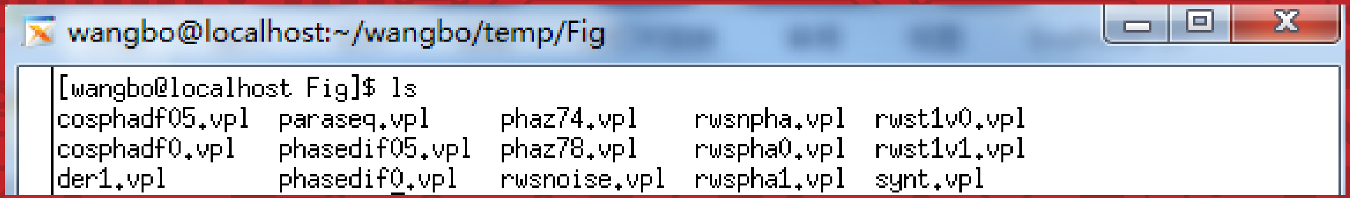



xtpen
Next Prev Quit Restart Run Stop Stretchy Forwards 0 delay 0.05
Amplitude
Time/s
0 0.2 0.4 0.6 0.8 1
-1 -0.5 0 0.5 1
SConstruct - emacs@localhost.localdomain
e Edit Options Buffers Tools IM-Python Python Help
######################################
Duffing Theory
######################################
Flow('cos',None,'math n1=2000 d1=0.001 output="cos(100*x1+3.1415926)"')
Plot('cos',
'''
wanttitle=n label1="Time/s" label2="Amplitude"
plotcol=5 plotfat=4 font=2 labelfat=4
min2=-1 min1=0 max1=1
''')
Flow('coskxi',None,'math n1=2000 d1=0.001 output="cos(100*x1+0.5*3.1415926)"
Plot('coskxi',
'''math output=in
graph wanttitle= plotcol=6 dash=1 label2= wantaxis2=n
whereylabel=right plotfat=7 min1=0 max1=1''')
Result('phasedif05','cos coskxi','Overlay')
xtpen
Next Prev Quit Restart Run Stop Stretchy Forwards 0 delay 0.05
0 0.2 0.4 0.6 0.8 1
Restart Run Stop Stretchy Forwards 0 delay 0.05
Amplitude
Time/s
0 0.2 0.4 0.6 0.8 1
-1 -0.5 0 0.5 1

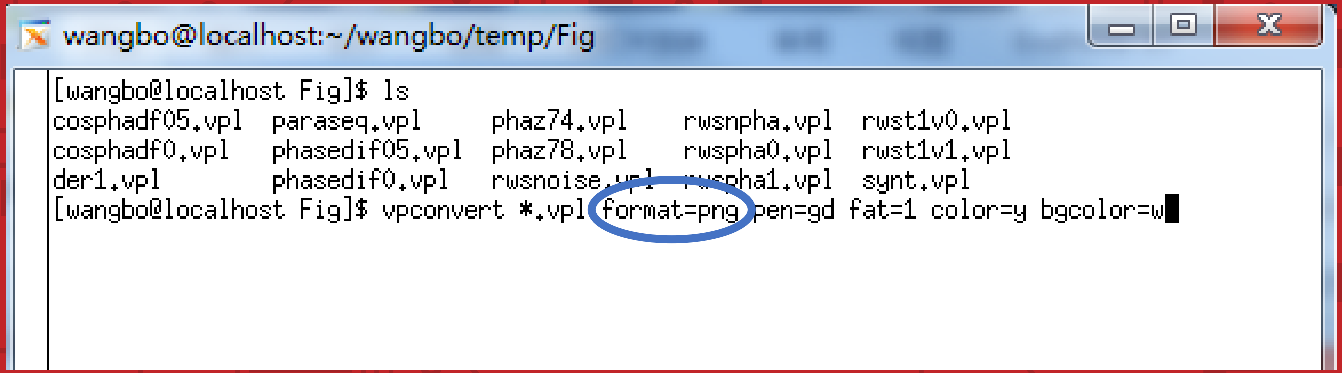

SConstruct - emacs@localhost.localdomain
File Edit Options Buffers Tools IM-Python Python Help
import os
os.environ['SFPENOPTS'] = 'bgcolor=w'
from rsf.proj import *
import math
from rsf.recipes.beg import server as private
######### Duffing threshold determination ########
Flow('zeros',None,'spike n1=3200 d1=0.001 o1=0 ')
ps = []
ga0 = 0.750
deltax = 0.001
for x in range (0,150,1):
gaz = ga0 + x*deltax
omegaz = 100
phax = 'phaz%d' % x
Flow(phax,'zeros','duffing1 gamma=%f kxi=0 omega=%f verb=n'%(gaz,omegaz))
if x == 74:


xtpen
Next Prev Quit Restart Run Stop Stretchy Forwards 0 delay 0.05
Amplitude
1
0.5
0
−0.5
−1
0
0.2
0.4
0.6
0.8
1
Time/s

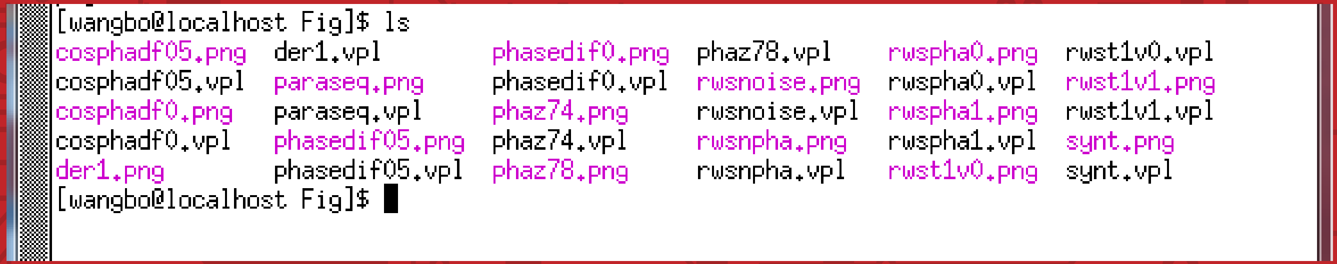
The “vpconvert” allows us to convert the picture format.



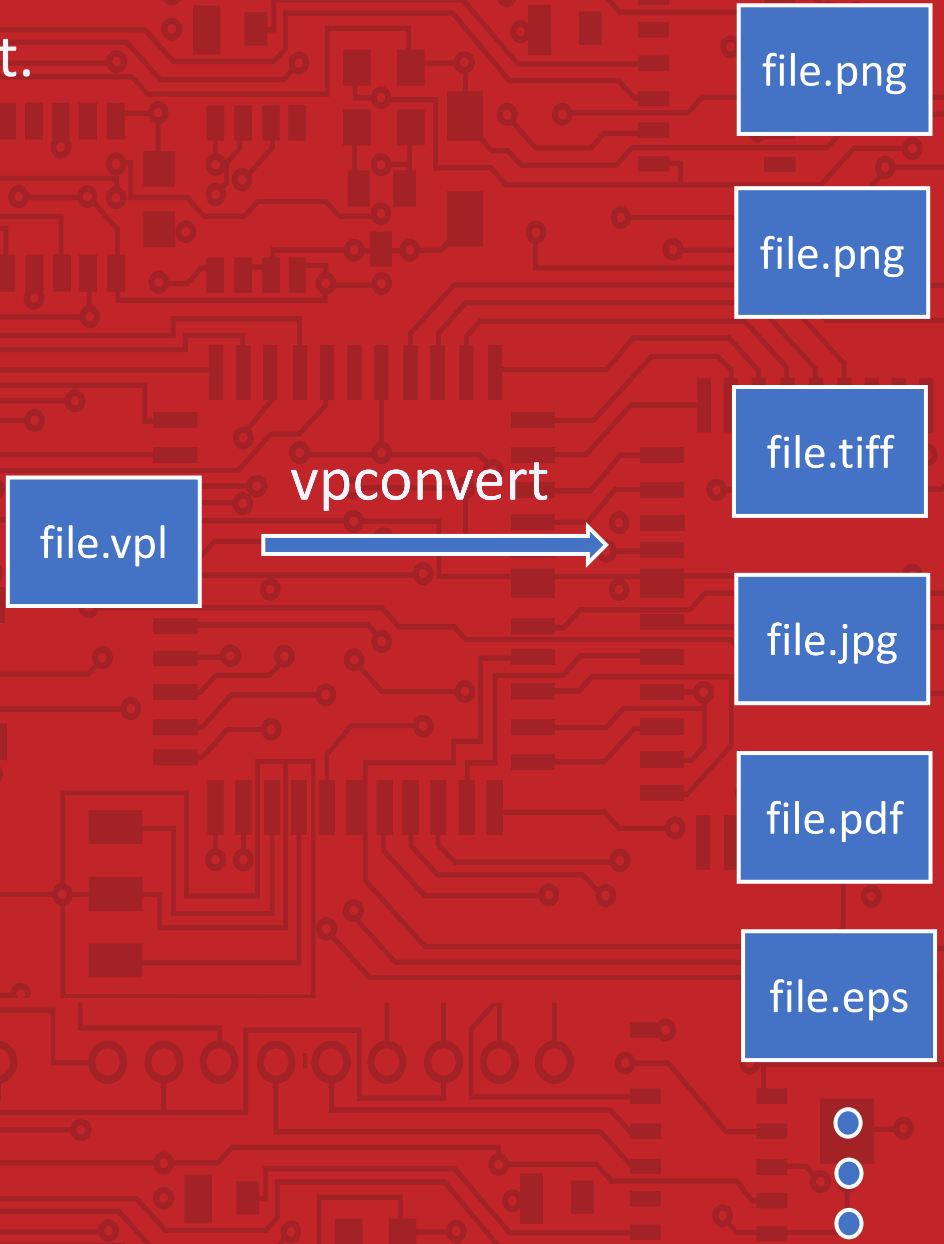
```
wangbo@localhost:~/wangbo/temp/Fig
[wangbo@localhost Fig]$ ls
cosphadf05.vpl  paraseq.vpl     phaz74.vpl    rwsnpha.vpl  rwst1v0.vpl
cosphadf0.vpl   phasedif05.vpl  phaz78.vpl    rwspha0.vpl  rwst1v1.vpl
der1.vpl        phasedif0.vpl   rwsnoise.vpl  rwspha1.vpl  synt.vpl
```



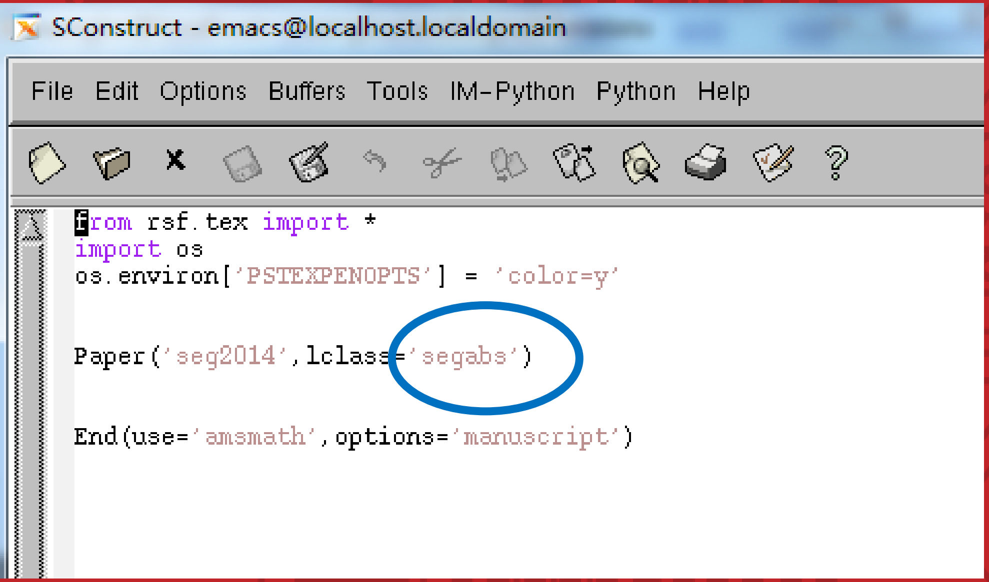
```
wangbo@localhost:~/wangbo/temp/Fig
[wangbo@localhost Fig]$ ls
cosphadf05.vpl  paraseq.vpl     phaz74.vpl    rwsnpha.vpl  rwst1v0.vpl
cosphadf0.vpl   phasedif05.vpl  phaz78.vpl    rwspha0.vpl  rwst1v1.vpl
der1.vpl        phasedif0.vpl   rwsnoise.vpl  rwspha1.vpl  synt.vpl
[wangbo@localhost Fig]$ vpconvert *.vpl format=png pen=gd fat=1 color=y bgcolor=w
```

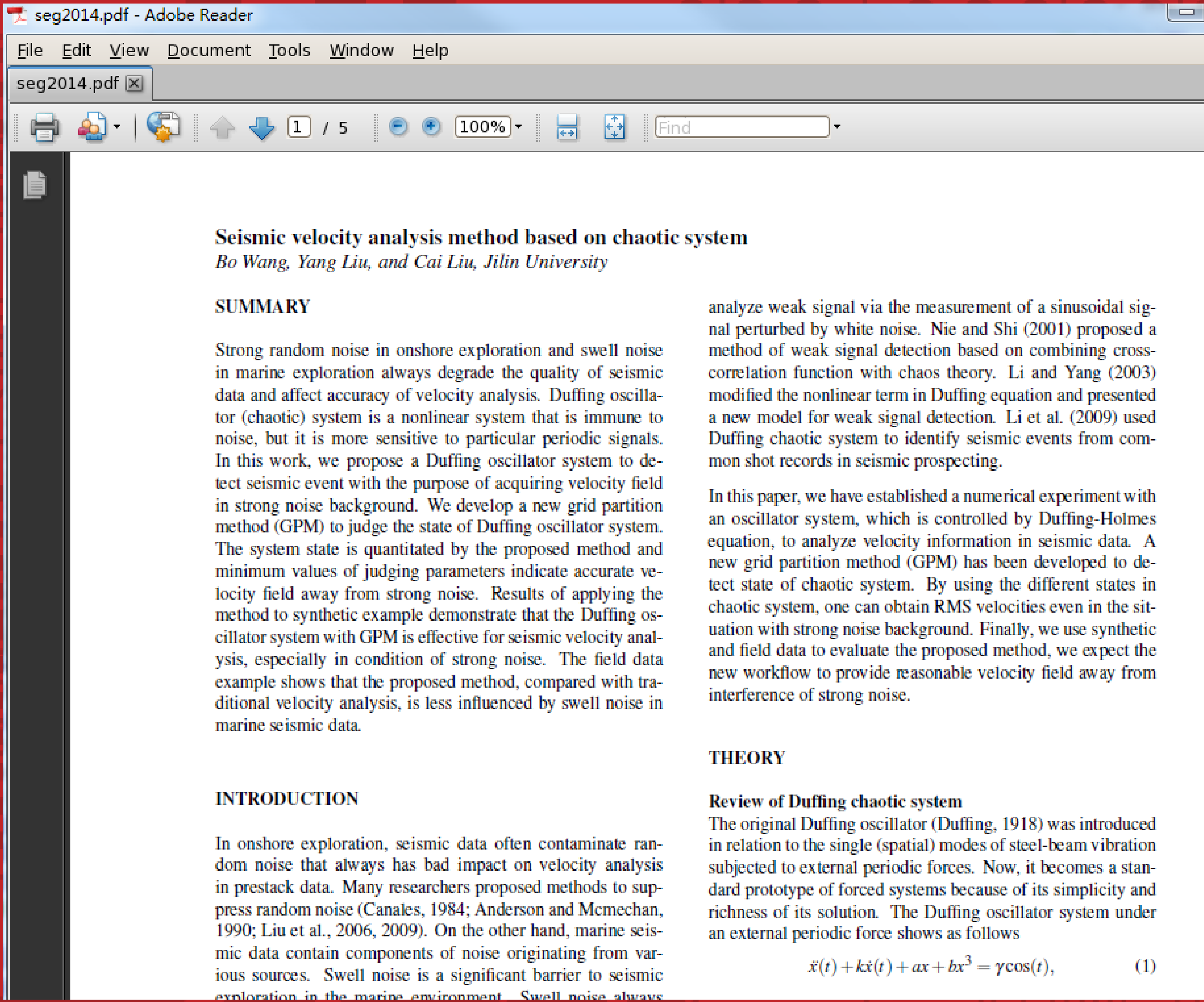


```
[wangbo@localhost Fig]$ ls
cosphadf05.png  der1.vpl        phasedif0.png   phaz78.vpl    rwspha0.png  rwst1v0.vpl
cosphadf05.vpl  paraseq.png     phasedif0.vpl   rwsnoise.png  rwspha0.vpl  rwst1v1.png
cosphadf0.png   paraseq.vpl     phaz74.png      rwsnoise.vpl  rwspha1.png  rwst1v1.vpl
cosphadf0.vpl   phasedif05.png  phaz74.vpl      rwsnpha.png   rwspha1.vpl  synt.png
der1.png        phasedif05.vpl  phaz78.png      rwsnpha.vpl   rwst1v0.png  synt.vpl
[wangbo@localhost Fig]$
```



The template of Latex makes the publication more convenient.





Seismic velocity analysis method based on chaotic system

Bo Wang, Yang Liu, and Cai Liu, Jilin University

SUMMARY

Strong random noise in onshore exploration and swell noise in marine exploration always degrade the quality of seismic data and affect accuracy of velocity analysis. Duffing oscillator (chaotic) system is a nonlinear system that is immune to noise, but it is more sensitive to particular periodic signals. In this work, we propose a Duffing oscillator system to detect seismic event with the purpose of acquiring velocity field in strong noise background. We develop a new grid partition method (GPM) to judge the state of Duffing oscillator system. The system state is quantitated by the proposed method and minimum values of judging parameters indicate accurate velocity field away from strong noise. Results of applying the method to synthetic example demonstrate that the Duffing oscillator system with GPM is effective for seismic velocity analysis, especially in condition of strong noise. The field data example shows that the proposed method, compared with traditional velocity analysis, is less influenced by swell noise in marine seismic data.

INTRODUCTION

In onshore exploration, seismic data often contaminate random noise that always has bad impact on velocity analysis in prestack data. Many researchers proposed methods to suppress random noise (Canales, 1984; Anderson and Mcmechan, 1990; Liu et al., 2006, 2009). On the other hand, marine seismic data contain components of noise originating from various sources. Swell noise is a significant barrier to seismic exploration in the marine environment. Swell noise always

analyze weak signal via the measurement of a sinusoidal signal perturbed by white noise. Nie and Shi (2001) proposed a method of weak signal detection based on combining cross-correlation function with chaos theory. Li and Yang (2003) modified the nonlinear term in Duffing equation and presented a new model for weak signal detection. Li et al. (2009) used Duffing chaotic system to identify seismic events from common shot records in seismic prospecting.

In this paper, we have established a numerical experiment with an oscillator system, which is controlled by Duffing-Holmes equation, to analyze velocity information in seismic data. A new grid partition method (GPM) has been developed to detect state of chaotic system. By using the different states in chaotic system, one can obtain RMS velocities even in the situation with strong noise background. Finally, we use synthetic and field data to evaluate the proposed method, we expect the new workflow to provide reasonable velocity field away from interference of strong noise.

THEORY

Review of Duffing chaotic system

The original Duffing oscillator (Duffing, 1918) was introduced in relation to the single (spatial) modes of steel-beam vibration subjected to external periodic forces. Now, it becomes a standard prototype of forced systems because of its simplicity and richness of its solution. The Duffing oscillator system under an external periodic force shows as follows

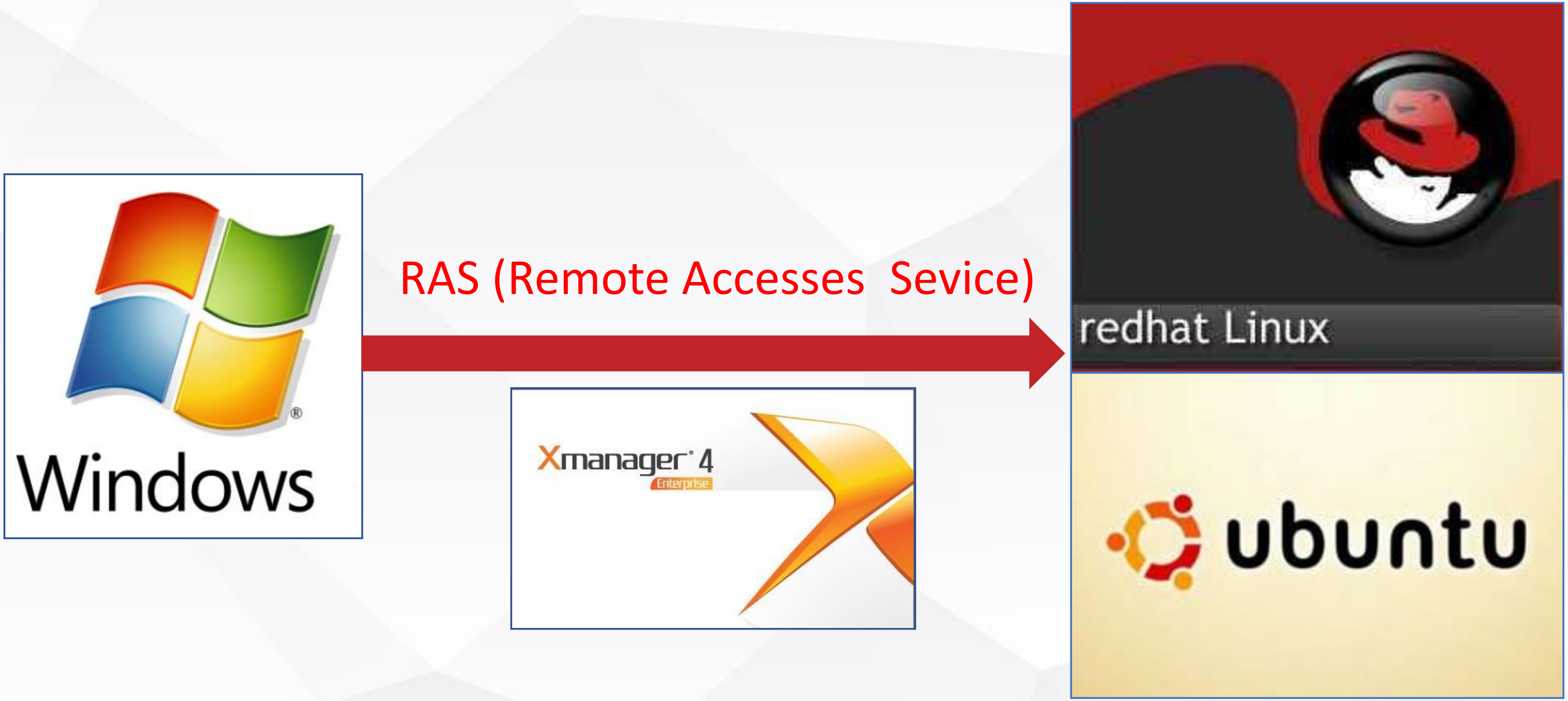
$$\ddot{x}(t)+k\dot{x}(t)+ax+bx^3=\gamma\cos(t), \qquad (1)$$

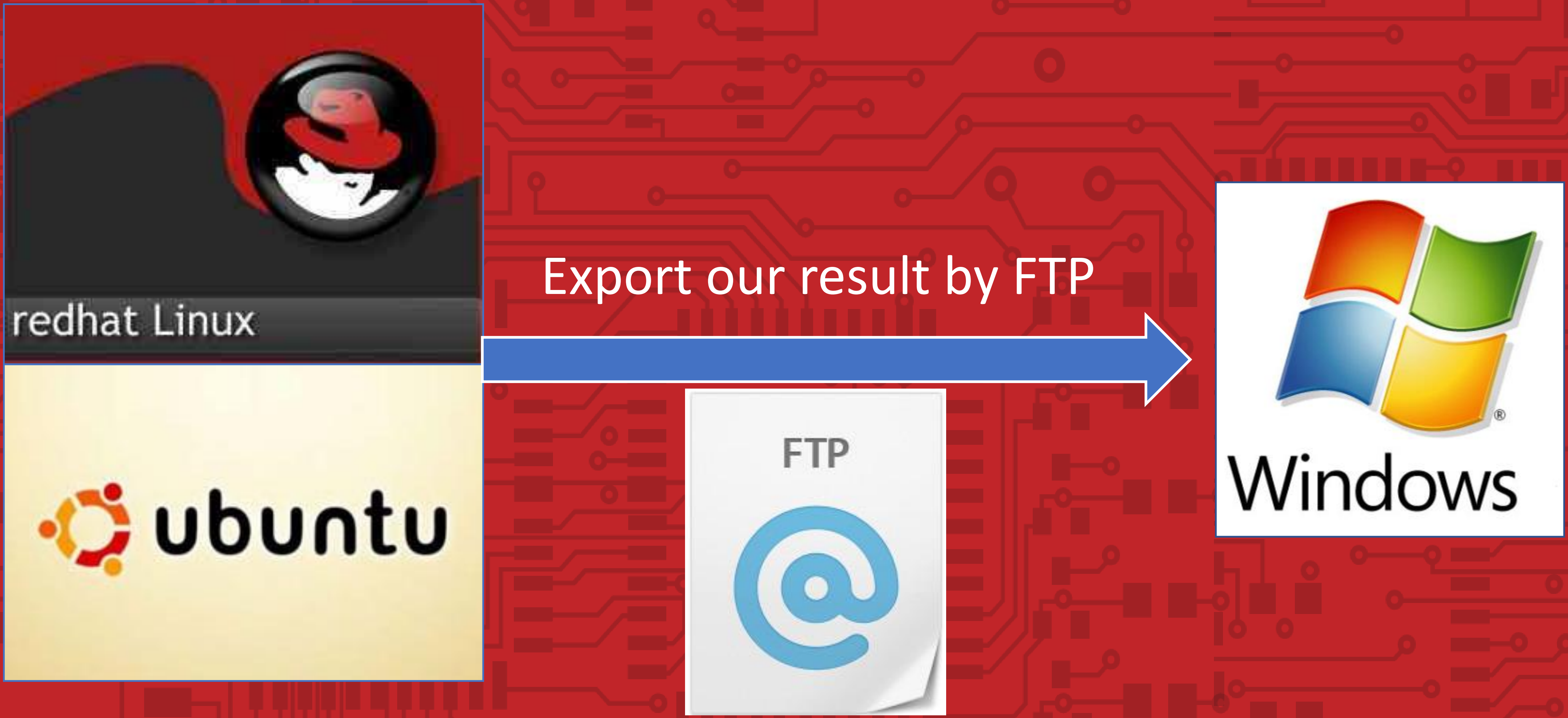
The environment configruation

Part

FOUR

4


Windows
RAS (Remote Accesses Sevice)
Xmanager 4
Enterprise
redhat Linux
ubuntu


redhat Linux
ubuntu
Export our result by FTP
FTP
Windows

THANKS

Supporters say that the ease of use of presentation software can save a lot of time for people who otherwise would have used other types of visual aid—hand-drawn or mechanically typeset slides, blackboards or whiteboards, or overhead projections. Ease of use also encourages those