

./main.py

```
#!/usr/bin/env python3
"""
Copyright_(c)_2014_Verzunov_S.N.
Institute_of_Automation_and_Information_tehnology
NAS_of_the_Kyrgyz_Republic
All_rights_reserved.
Code_released_under_the_GNU_GENERAL_PUBLIC_LICENSE_Version_3,_June_2007
"""

import sys
from PyQt4 import QtGui # connect PyQt
from forms.mainform import MainForm
import os
from PyQt4.QtCore import pyqtRemoveInputHook

def main():
    pyqtRemoveInputHook()
    os.environ['LANG'] = "en_EN.UTF-8"
    app = QtGui.QApplication(sys.argv)
    app.setStyle('Windows') # 'Windows', 'Motif', 'CDE',
    # 'Plastique', 'GTK+', 'Cleanlooks'
    mainform = MainForm(app)
    mainform.show()
    app.exec_()

if __name__ == "__main__":
    sys.exit(main())
```

./getlisting.py

```
#!/usr/bin/python3
"""
Copyright_(c)_2014_Verzunov_S.N.
Institute_of_Automation_and_Information_tehnology
NAS_of_the_Kyrgyz_Republic
All_rights_reserved.
Code_released_under_the_GNU_GENERAL_PUBLIC_LICENSE_Version_3,_June_2007
"""

import os
from subprocess import call
def get_filepaths(directory):
    """
    This function will generate the file names in a directory
    tree by walking the tree either top-down or bottom-up. For each
    directory in the tree rooted at directory_top (including top itself),
    it yields a 3-tuple (dirpath, dirnames, filenames).
    """
    file_paths = [] # List which will store all of the full filepaths.

    # Walk the tree.
    for root, directories, files in os.walk(directory):
        files = [f for f in files if not f[0] == '.']
        directories[:] = [d for d in directories if not d[0] == '.']
        for filename in files:
            # Join the two strings in order to form the full filepath.
            filepath = os.path.join(root, filename)
            # import pdb; pdb.set_trace()
            file_paths.append(filepath)
```

```

        fileName, fileExtension = os.path.splitext(filepath)
        if fileExtension=='.py' or fileExtension=='.ui':
            file_paths.append(filepath) # Add it to the list.

    return file_paths # Self-explanatory.

files = get_filepaths(".")
listing = open("listing.tex", "w")
listing.write(r"""\documentclass{article}
\usepackage[utf8]{inputenc}
\usepackage[russian]{babel}
\usepackage{cmap}""")
listing.write(r"""\usepackage{listings}%%%%%%%%%\_Include\_the\_listings\_package
""")
listing.write(r"""\usepackage[left=2cm,_top=2cm,_right=0.5cm,_bottom=20mm, """)
listing.write(r"""\nohead,_nofoot]{geometry}
""")
listing.write(r"""\begin{document}
""")

for f in files:
    fileName, fileExtension = os.path.splitext(f)
    if fileExtension == '.py':
        listing.write(
            r"\lstinputlisting[language=Python,_breaklines=true,")
        listing.write(r"title=\lstname]{"+f+"}")
    else:
        listing.write(
            r"\lstinputlisting[language=Python,_breaklines=true,")
        listing.write(r"title=\lstname]{"+f+"}")

listing.write(r"""\end{document}""")
listing.close()
call(["pdflatex", "listing.tex"])

```

./wavelets/cwt.py

import numpy as NP

"""

A_module_which_implements_the_continuous_wavelet_transform

Code_released_under_the_BSD_3-clause_licence.

Copyright_(c)_2012,_R_W_Fearick,_University_of_Cape_Town
All_rights_reserved.

Redistribution_and_use_in_source_and_binary_forms,_with_or_without
modification,_are_permitted_provided_that_the_following_conditions_are_met:
 * Redistributions_of_source_code_must_retain_the_above_copyright_notice,
 this_list_of_conditions_and_the_following_disclaimer.
 * Redistributions_in_binary_form_must_reproduce_the_above_copyright

notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
 *Neither the name of the University of Cape Town nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Wavelet classes:

Morlet

MorletReal

MexicanHat

Paul2: Paul order 2

Paul4: Paul order 4

DOG1: 1st Derivative Of Gaussian

DOG4: 4th Derivative Of Gaussian

Haar: Unnormalised version of continuous Haar transform

HaarW: Normalised Haar

Usage e.g.

wavelet=Morlet(data, largestscale=2, notes=0, order=2, scaling="log")

data: Numeric array of data (float), with length ndata.

Optimum length is a power of 2 (for FFT)

Worst-case length is a prime

largestscale:

largest scale as inverse fraction of length

scale=len(data)/largestscale

smallest scale should be ≥ 2 for meaningful data

notes: number of scale intervals per octave

if notes=0, scales are on a linear increment

order: order of wavelet for wavelets with variable order

[Paul, DOG, ...]

scaling: "linear" or "log" scaling of the wavelet scale.

Note that feature width in the scale direction

is constant on a log scale.

Attributes of instance:

wavelet.cwt: 2-d array of Wavelet coefficients, (nscales, ndata)

wavelet.nscale: Number of scale intervals

wavelet.scales: Array of scale values

Note that meaning of the scale will depend on the family

wavelet.fourierwl: Factor to multiply scale by to get scale

of equivalent FFT

Using this factor, different wavelet families will

have comparable scales

References:
A_practical_guide_to_wavelet_analysis
C_Torrance_and_GP_Comp
Bull_Amer_Meteor_Soc_Vol_79_No_1_61-78_(1998)
naming_below_vaguely_follows_this.

```
updates:
(24/2/07): __Fix_Morlet_so_can_get_MorletReal_by_cutting_out_H
(10/04/08): __Numeric__>__numpy
(25/07/08): __log_and_lin_scale_increment_in_same_direction!
_____swap_indices_in_2-d_coefficient_matrix
_____explicit_scaling_of_scale_axis
_____"""
```

```
class Cwt:
    """
    _____Base_class_for_continuous_wavelet_transforms
    _____Implements_cwt_via_the_Fourier_transform
    _____Used_by_subclass_which_provides_the_method_wf(self,s_omega)
    _____wf_is_the_Fourier_transform_of_the_wavelet_function.
    _____Returns_an_instance.
    _____"""

    fourierwl=1.00

    def _log2(self, x):
        # utility function to return (integer) log2
        return int(NP.log(float(x)) / NP.log(2.0) + 0.0001 )

    def __init__(self, data, finished, notifyProgress, largestscale=1, notes=0,
                  order=2, scaling='linear', omega0=5.):
        """
        _____Continuous_wavelet_transform_of_data

        _____data:_____data_in_array_to_transform,length_must_be_power_of_2
        _____notes:_____number_of_scale_intervals_per_octave
        _____largestscale:_____largest_scale_as_inverse_fraction_of_length
        _____of_data_array
        _____scale_____len(data)/largestscale
        _____smallest_scale_should_be_____2_for_meaningful_data
        _____order:_____Order_of_wavelet_basis_function_for_some_families
        _____scaling:_____Linear_or_log
        _____"""
        ndata = len(data)
        self.order=order
        self.omega0=omega0
        self.scale=largestscale
        self._setscales(ndata,largestscale,notes,scaling)
        self.cwt= NP.zeros((self.nscale,ndata), NP.complex64)
        omega = NP.array(list(range(0, ndata//2)) +
                          list(range(-ndata//2,0)))*(2.0*NP.pi/ndata)
        datahat=NP.fft.fft(data)
        self.fftdata=datahat
        #self.psihat0=self.wf(omega*self.scales[3*self.nscale/4])
        # loop over scales and compute wavelet coefficients at each scale
        # using the fft to do the convolution
```

```

    for scaleindex in range(self.nscale):
        currentscale=self.scales[scaleindex]
        self.currentscale=currentscale # for internal use
        s_omega = omega*currentscale
        psihat=self.wf(s_omega)
        psihat = psihat * NP.sqrt(2.0*NP.pi*currentscale)
        convhat = psihat * datahat
        W      = NP.fft.ifft(convhat)
        self.cwt[scaleindex,0:ndata] = W
        notifyProgress.emit(scaleindex*100//self.nscale)
    finished.emit(self)
def _setscales(self,ndata,largestscale,notes,scaling):
    """
    ~~~~~~if notes non-zero, returns a log scale based on notes per octave
    ~~~~~~else a linear scale
    ~~~~~~(25/07/08): fix notes!=0 case so smallest scale at [0]
    ~~~~~~"""
    if scaling=="log":
        if notes<=0: notes=1
        # adjust nscale so smallest scale is 2
        noctave=self._log2( ndata/largestscale/2 )
        self.nscale=notes*noctave
        self.scales=NP.zeros(self.nscale, float)
        for j in range(self.nscale):
            self.scales[j] = ndata/(self.scale *
                                   (2.0**((float(self.nscale-1-j)/notes))))

    elif scaling=="linear":
        nmax=ndata/largestscale/2
        step=(nmax-2)/2**notes
        self.scales=NP.arange(float(2),float(nmax),step)
        self.nscale=len(self.scales)
    else: raise (ValueError, "scaling must be linear or log")
    return

def getdata(self):
    """
    ~~~~~~returns wavelet coefficient array
    ~~~~~~"""
    return self.cwt
def getcoefficients(self):
    return self.cwt
def getpower(self):
    """
    ~~~~~~returns square of wavelet coefficient array
    ~~~~~~"""
    return (self.cwt* NP.conjugate(self.cwt)).real
def getangle(self):
    """
    ~~~~~~returns angle of wavelet coefficient array
    ~~~~~~"""
    return NP.angle(self.cwt)

def getscales(self):
    """
    ~~~~~~returns array containing scales used in transform
    ~~~~~~"""

```

```

        return self.scales
    def getnscale(self):
        """
        return number of scales
        """
        return self.nscale

# wavelet classes
class Morlet(Cwt):
    """
    Morlet wavelet
    """
    #_omega0=5.0
    def wf(self, s_omega):
        Cwt.fourierwl=4* NP.pi/(self.omega0+ NP.sqrt(2.0+self.omega0**2))
        H= NP.ones(len(s_omega))
        n=len(s_omega)
        for i in range(len(s_omega)):
            if s_omega[i] < 0.0: H[i]=0.0
        # !!!! note : was s_omega/8 before 17/6/03
        xhat=0.75112554*( NP.exp(-(s_omega-self.omega0)**2/2.0))*H
        return xhat

class MorletReal(Cwt):
    """
    Real Morlet wavelet
    """
    #_omega0=5.0
    def wf(self, s_omega):
        Cwt.fourierwl=4* NP.pi/(self.omega0+ NP.sqrt(2.0+self.omega0**2))
        H= NP.ones(len(s_omega))
        n=len(s_omega)
        for i in range(len(s_omega)):
            if s_omega[i] < 0.0: H[i]=0.0
        # !!!! note : was s_omega/8 before 17/6/03
        xhat = 0.75112554*(NP.exp(-(s_omega-self.omega0)**2/2.0) +
                           NP.exp(-(s_omega+self.omega0)**2/2.0) -
                           NP.exp(-(self.omega0)**2/2.0) +
                           NP.exp(-(self.omega0)**2/2.0))
        return xhat

## class Paul4(Cwt):
##     """
##     Paul m=4 wavelet
##     """
##     fourierwl=4* NP.pi/(2.*4+1.)
##     def wf(self, s_omega):
##         n=len(s_omega)
##         xhat= NP.zeros(n)
##         xhat[0:n/2]=0.11268723*s_omega[0:n/2]**4* NP.exp(-s_omega[0:n/2])
##         #return 0.11268723*s_omega**2*exp(-s_omega)*H
##         return xhat

## class Paul2(Cwt):
##     """
##     Paul m=2 wavelet

```

```

"""
fourierwl=4* NP.pi/(2.*2+1.)
def wf(self, s_omega):
    n=len(s_omega)
    xhat= NP.zeros(n)
    xhat[0:n/2]=1.1547005*s_omega[0:n/2]**2* NP.exp(-s_omega[0:n/2])
    #return 0.11268723*s_omega**2*exp(-s_omega)*H
    return xhat

class Paul(Cwt):
    """
    Paul_order_m_wavelet
    """
    def wf(self, s_omega):
        Cwt.fourierwl=4* NP.pi/(2.*self.order+1.)
        m=self.order
        n=len(s_omega)
        normfactor=float(m)
        for i in range(1,2*m):
            normfactor=normfactor*i
        normfactor=2.0**m/ NP.sqrt(normfactor)
        xhat= NP.zeros(n)
        xhat[0:n/2]=normfactor*s_omega[0:n/2]**m* NP.exp(-s_omega[0:n/2])
        #return 0.11268723*s_omega**2*exp(-s_omega)*H
        return xhat

class MexicanHat(Cwt):
    """
    2nd Derivative Gaussian (mexican hat) wavelet
    """
    fourierwl=2.0* NP.pi/ NP.sqrt(2.5)
    def wf(self, s_omega):
        # should this number be 1/sqrt(3/4) (no pi)?
        #s_omega = s_omega/self.fourierwl
        #print max(s_omega)
        a=s_omega**2
        b=s_omega**2/2
        return a* NP.exp(-b)/1.1529702
        #return s_omega**2*exp(-s_omega**2/2.0)/1.1529702

class DOG4(Cwt):
    """
    4th Derivative Gaussian wavelet
    see also T&C errata for - sign
    but reconstruction seems to work best with +!
    """
    fourierwl=2.0* NP.pi/ NP.sqrt(4.5)
    def wf(self, s_omega):
        return s_omega**4* NP.exp(-s_omega**2/2.0)/3.4105319

class DOG1(Cwt):
    """
    1st Derivative Gaussian wavelet
    but reconstruction seems to work best with +!
    """
    fourierwl=2.0* NP.pi/ NP.sqrt(1.5)

```

```

##      def wf(self, s_omega):
##          dog1= NP.zeros(len(s_omega),NP.complex64)
##          dog1.imag=s_omega* NP.exp(-s_omega**2/2.0)/NP.sqrt(NP.pi)
##          return dog1

class DOG(Cwt):
    """
    Derivative_Gaussian_wavelet_of_order_m
    but_reconstruction_seems_to_work_best_with_+!
    """
    def wf(self, s_omega):
        try:
            from scipy.special import gamma
        except ImportError:
            print ("Requires_scipy_gamma_function")
            raise ImportError
        Cwt.fourierwl=2* NP.pi/ NP.sqrt(self.order+0.5)
        m = self.order
        dog = 1.0J**m*s_omega**m* NP.exp(-s_omega**2/2)/NP.sqrt(
            gamma(self.order+0.5))
        return dog

class Haar(Cwt):
    """
    Continuous_version_of_Haar_wavelet
    """
    #      note: not orthogonal!
    #      note: s_omega/4 matches Lecroix scale defn.
    #      s_omega/2 matches orthogonal Haar
    # 2/8/05 constants adjusted to match artem eim

    fourierwl=1.0#1.83129 #2.0
    def wf(self, s_omega):
        haar= NP.zeros(len(s_omega),NP.complex64)
        om = s_omega[:]/ self.currentscale
        om[0]=1.0 #prevent divide error
        #haar.imag=4.0*sin(s_omega/2)**2/om
        haar.imag=4.0* NP.sin(s_omega/4)**2/om
        return haar

## class HaarW(Cwt):
##     """
##     Continuous version of Haar wavelet (norm)
##     """
##     #      note: not orthogonal!
##     #      note: s_omega/4 matches Lecroix scale defn.
##     #      s_omega/2 matches orthogonal Haar
##     # normalised to unit power

##     fourierwl=1.83129*1.2 #2.0
##     def wf(self, s_omega):
##         haar= NP.zeros(len(s_omega),NP.complex64)
##         om = s_omega[:]/self.currentscale
##         om[0]=1.0 #prevent divide error
##         #haar.imag=4.0*sin(s_omega/2)**2/om
##         haar.imag=4.0* NP.sin(s_omega/2)**2/om

```



```

###          return haar

if __name__=="__main__":
    import numpy as np
    import pylab as mpl

    wavelet=Morlet
    maxscale=4
    notes=16
    scaling="log" #or "linear"
    scaling="linear"
    plotpower2d=True

    # set up some data
    Ns=2048
    #limits of analysis
    Nlo=0
    Nhi=Ns
    # sinusoids of two periods , 128 and 32.
    x=np.arange(0.0,1.0*Ns,1.0)
    A=np.sin(2.0*np.pi*x/128.0)
    B=np.sin(2.0*np.pi*x/256.0)
    A[512:1024]+=B[0:512]

    # Wavelet transform the data
    cw=wavelet(A,maxscale,notes,scaling=scaling)
    scales=cw.getscales()
    cwt=cw.getdata()
    # power spectrum
    pwr=cw.getpower()
    scalespec=np.sum(pwr,axis=1)/scales # calculate scale spectrum
    # scales
    y=cw.fourierwl*scales
    x=np.arange(Nlo*1.0,Nhi*1.0,1.0)

    fig=mpl.figure(1)

    # 2-d coefficient plot
    ax=mpl.axes([0.4,0.1,0.55,0.4])
    mpl.xlabel('Time_[s]')
    plotcwt=np.clip(np.fabs(cwt.real), 0., 1000.)
    if plotpower2d: plotcwt=pwr
    im=mpl.imshow(plotcwt,cmap=mpl.cm.jet,extent=[x[0],x[-1],y[-1],y[0]],
                aspect='auto')
    #colorbar()
    if scaling=="log": ax.set_yscale('log')
    mpl.ylim(y[0],y[-1])
    ax.xaxis.set_ticks(np.arange(Nlo*1.0,(Nhi+1)*1.0,100.0))
    ax.yaxis.set_ticklabels(["",""])
    theposition=mpl.gca().get_position()

    # data plot
    ax2=mpl.axes([0.4,0.54,0.55,0.3])
    mpl.ylabel('Data')
    pos=ax.get_position()

```

```

mpl.plot(x,A,'b-')
mpl.xlim(Nlo*1.0,Nhi*1.0)
ax2.xaxis.set_ticklabels(["",""])
mpl.text(0.5,0.9,"Wavelet_example_with_extra_panes",
        fontsize=14,bbox=dict(facecolor='green',alpha=0.2),
        transform = fig.transFigure, horizontalalignment='center')

```

```

# projected power spectrum
ax3=mpl.axes([0.08,0.1,0.29,0.4])
mpl.xlabel('Power')
mpl.ylabel('Period [s]')
vara=1.0
if scaling=="log":
    mpl.loglog(scalespec/vara+0.01,y,'b-')
else:
    mpl.semilogx(scalespec/vara+0.01,y,'b-')
mpl.ylim(y[0],y[-1])
mpl.xlim(1000.0,0.01)

mpl.show()

```

./wavelets/__init__.py

-- coding: utf-8 -*-*

./forms/mplqt4.py

```

"""
Copyright_(c)_2014_Verzunov_S.N.
Institute_of_Automation_and_Information_tehnology
NAS_of_the_Kyrgyz_Republic
All_rights_reserved.
Code_released_under_the_GNU_GENERAL_PUBLIC_LICENSE_Version_3,_June_2007
"""
import sys, os, random
from PyQt4 import QtGui, QtCore

from numpy import arange, sin, pi
from matplotlib.backends.backend_qt4agg import FigureCanvasQTAgg as FigureCanvas
from matplotlib.figure import Figure, rcParams
from matplotlib.backend_bases import LocationEvent
from matplotlib.backend_bases import Event
class MyMplCanvas(FigureCanvas):
    """Ultimately, this is a QWidget (as well as a FigureCanvasAgg, etc.). """
    canvasEnter=QtCore.pyqtSignal()
    mouseMotion = QtCore.pyqtSignal(Event)
    canvasLeave=QtCore.pyqtSignal()
    def __init__(self, parent=None, width=5, height=4, dpi=100):
        rcParams.update({'font.size': 8})
        self._figure = Figure(figsize=(width, height), dpi=dpi)
        self.axes = self._figure.add_subplot(111)
        # We want the axes cleared every time plot() is called
        self.axes.hold(False)
        self.compute_initial_figure()
        #
        FigureCanvas.__init__(self, self._figure)
        self.setParent(parent)

```

```

FigureCanvas.setSizePolicy(self,
                             QtGui.QSizePolicy.Expanding,
                             QtGui.QSizePolicy.Expanding)
FigureCanvas.updateGeometry(self)

self._figure.canvas.mpl_connect('motion_notify_event',
                                 lambda event: self.mouseMotion.emit(event))
self._figure.canvas.mpl_connect('figure_enter_event',
                                 lambda event: self.canvasEnter.emit())
self._figure.canvas.mpl_connect('figure_leave_event',
                                 lambda event: self.canvasLeave.emit())

def saveFigure(self, fileName, dpi = 100):
    self._figure.savefig(fileName, dpi=dpi)

def compute_initial_figure(self):
    pass

    """
    Copyright_(c)_2014_Verzunov_S.N.
    Institute_of_Automation_and_Information_tehnology
    NAS_of_the_Kyrgyz_Republic
    All_rights_reserved.
    Code_released_under_the_GNU_GENERAL_PUBLIC_LICENSE_Version_3,_June_2007
    """
    from PyQt4 import QtGui, QtCore
    class TrueScrollBar(QtGui.QScrollBar):
        invValueChanged=QtCore.pyqtSignal(int)
        invSliderMoved=QtCore.pyqtSignal(int)

        def __init__(self, label, parent=None):
            QtGui.QScrollBar.__init__(self, parent)
            self.__value=0
            self.setOrientation(QtCore.Qt.Vertical)
            self.valueChanged.connect(self.__change)
            self.sliderMoved.connect(self.__moved)
            self.setTracking(False)
        def __change(self, value):
            self.__value=self.maximum() - value+self.minimum()
            self.invValueChanged.emit(self.__value)
            print('emit_%s'%self.__value)

        def __moved(self, value):
            print('Move_%s'% value)
            value=self.maximum() - value + self.minimum()
            self.invSliderMoved.emit(value)

        def setValue(self, value):
            print('setValue%s'%value)
            self.__value=value
            self.invValueChanged.emit(value)
            value=self.maximum()-value+self.minimum()
            #self.setSliderPosition(value)
            QtGui.QScrollBar.setValue(self, value)

```

```

def value(self):
    print('Getvalue=%s'%self.__value)
    return self.__value

```

./forms/aboutform.ui

```

<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
  <class>Dialog</class>
  <widget class="QDialog" name="Dialog">
    <property name="geometry">
      <rect>
        <x>0</x>
        <y>0</y>
        <width>400</width>
        <height>208</height>
      </rect>
    </property>
    <property name="windowTitle">
      <string>About</string>
    </property>
    <layout class="QGridLayout" name="gridLayout">
      <item row="1" column="0">
        <widget class="QDialogButtonBox" name="buttonBox">
          <property name="orientation">
            <enum>Qt::Horizontal</enum>
          </property>
          <property name="standardButtons">
            <set>QDialogButtonBox::Close</set>
          </property>
        </widget>
      </item>
      <item row="0" column="0">
        <layout class="QVBoxLayout" name="verticalLayout">
          <item>
            <widget class="QLabel" name="label">
              <property name="text">
                <string>&lt;html&gt;&lt;head&gt;&lt;body&gt;&lt;p align="center"&gt;
                  Wavelet analysis of magnetic variations&lt;/p&gt;&lt;p&gt;Program ver. 0.1&lt;/p&gt;&lt;/body&gt;&lt;/html&gt;</string>
              </property>
            </widget>
          </item>
          <item>
            <widget class="QLabel" name="pythonVer">
              <property name="text">
                <string>Python ver.</string>
              </property>
            </widget>
          </item>
          <item>
            <widget class="QLabel" name="qtVer">
              <property name="text">
                <string>Qt ver.</string>
              </property>
            </widget>
          </item>
        </layout>
      </item>
    </layout>
  </widget>
</ui>

```

```

</item>
<item>
  <widget class="QLabel" name="matplotlibVer">
    <property name="text">
      <string>Matplotlib ver.</string>
    </property>
  </widget>
</item>
<item>
  <widget class="QLabel" name="numpyVer">
    <property name="text">
      <string>Numpy ver.</string>
    </property>
  </widget>
</item>
<item>
  <widget class="QLabel" name="sciPyVer">
    <property name="text">
      <string>SciPy ver.</string>
    </property>
  </widget>
</item>
<item>
  <widget class="QLabel" name="pyQtVer">
    <property name="text">
      <string>PyQt ver.</string>
    </property>
  </widget>
</item>
<item>
  <spacer name="verticalSpacer">
    <property name="orientation">
      <enum>Qt::Vertical</enum>
    </property>
    <property name="sizeHint" stdset="0">
      <size>
        <width>20</width>
        <height>40</height>
      </size>
    </property>
  </spacer>
</item>
</layout>
</item>
</layout>
</widget>
<resources/>
<connections>
  <connection>
    <sender>buttonBox</sender>
    <signal>accepted()</signal>
    <receiver>Dialog</receiver>
    <slot>accept()</slot>
  <hints>
    <hint type="sourcelabel">
      <x>248</x>
    </hint>
  </hints>
</connection>

```

```

        <y>254</y>
    </hint>
    <hint type="destinationlabel">
        <x>157</x>
        <y>274</y>
    </hint>
</hints>
</connection>
<connection>
    <sender>buttonBox</sender>
    <signal>rejected()</signal>
    <receiver>Dialog</receiver>
    <slot>reject()</slot>
    <hints>
        <hint type="sourcelabel">
            <x>316</x>
            <y>260</y>
        </hint>
        <hint type="destinationlabel">
            <x>286</x>
            <y>274</y>
        </hint>
    </hints>
</connection>
</connections>
</ui>

```

./forms/dataheaderform.py

```

"""
Copyright_(c)_2014_Verzunov_S.N.
Institute_of_Automation_and_Information_tehnogology
NAS_of_the_Kyrgyz_Republic
All_rights_reserved.
Code_released_under_the_GNU_GENERAL_PUBLIC_LICENSE_Version_3,_June_2007
"""

```

```

from PyQt4 import QtCore, QtGui, uic

```

```

class DataHeaderForm(QtGui.QDialog):
    def __init__(self, header):
        super(DataHeaderForm, self).__init__()
        uic.loadUi("forms/dataheaderform.ui", self)
        self.buttonBox.accepted.connect(self.close)
        for key in header:
            if len(key[0][1:]) > 1:
                self.listWidget.addItem(key[0][1:])

```

./forms/progressgroup.py

```

"""
Copyright_(c)_2014_Verzunov_S.N.
Institute_of_Automation_and_Information_tehnogology
NAS_of_the_Kyrgyz_Republic
All_rights_reserved.
Code_released_under_the_GNU_GENERAL_PUBLIC_LICENSE_Version_3,_June_2007
"""

```

```

from PyQt4 import QtCore, QtGui, uic

```

```

class ProgressGroup(QtGui.QWidget):
    cancelled = QtCore.pyqtSignal()
    def __init__(self, label=None, statusbar=None):
        QtGui.QWidget.__init__(self)
        uic.loadUi("forms/progressgroup.ui", self)
        if label is not None:
            self.label.setText(label)
        self.cancelButton.clicked.connect(self._cancelled)
        if statusbar is not None:
            statusbar.clearMessage()

    def _cancelled(self):
        self.cancelled.emit()

    def setValue(self, value):
        self.progressBar.setValue(value)

```

./forms/downloadform.ui

```

<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
  <class>Dialog</class>
  <widget class="QDialog" name="Dialog">
    <property name="geometry">
      <rect>
        <x>0</x>
        <y>0</y>
        <width>342</width>
        <height>190</height>
      </rect>
    </property>
    <property name="windowTitle">
      <string>Download Data</string>
    </property>
    <layout class="QGridLayout" name="gridLayout">
      <item row="2" column="0">
        <widget class="QDialogButtonBox" name="buttonBox">
          <property name="orientation">
            <enum>Qt::Horizontal</enum>
          </property>
          <property name="standardButtons">
            <set>QDialogButtonBox::Cancel|QDialogButtonBox::Ok</set>
          </property>
        </widget>
      </item>
      <item row="0" column="0">
        <layout class="QFormLayout" name="formLayout">
          <property name="fieldGrowthPolicy">
            <enum>QFormLayout::ExpandingFieldsGrow</enum>
          </property>
          <item row="0" column="0">
            <widget class="QLabel" name="label_5">
              <property name="text">
                <string>Time step:</string>
              </property>
            </widget>

```

```

</item>
<item row="0" column="1">
  <widget class="QComboBox" name="stepComboBox">
    <property name="minimumSize">
      <size>
        <width>100</width>
        <height>0</height>
      </size>
    </property>
    <property name="currentIndex">
      <number>-1</number>
    </property>
  </widget>
  <item>
    <property name="text">
      <string>1 min</string>
    </property>
  </item>
  <item>
    <property name="text">
      <string>1 hour</string>
    </property>
  </item>
</widget>
</item>
<item row="1" column="0">
  <widget class="QLabel" name="label">
    <property name="text">
      <string>Observatory:</string>
    </property>
  </widget>
</item>
<item row="1" column="1">
  <widget class="QComboBox" name="obsComboBox">
    <property name="sizePolicy">
      <sizepolicy hstretch="MinimumExpanding" vstretch="Expanding">
        <horstretch>0</horstretch>
        <verstretch>0</verstretch>
      </sizepolicy>
    </property>
    <property name="minimumSize">
      <size>
        <width>250</width>
        <height>0</height>
      </size>
    </property>
    <property name="maximumSize">
      <size>
        <width>78</width>
        <height>16777215</height>
      </size>
    </property>
  </widget>
</item>
<item row="3" column="0">
  <widget class="QLabel" name="label_2">
    <property name="text">

```



```

    <string>From:</string>
  </property>
</widget>
</item>
<item row="4" column="0">
  <widget class="QLabel" name="label_3">
    <property name="text">
      <string>To:</string>
    </property>
  </widget>
</item>
<item row="5" column="0">
  <widget class="QLabel" name="label_4">
    <property name="text">
      <string>Series:</string>
    </property>
  </widget>
</item>
<item row="5" column="1">
  <widget class="QComboBox" name="seriesComboBox">
    <property name="minimumSize">
      <size>
        <width>100</width>
        <height>0</height>
      </size>
    </property>
    <item>
      <property name="text">
        <string notr="true">f</string>
      </property>
    </item>
    <item>
      <property name="text">
        <string notr="true">h</string>
      </property>
    </item>
    <item>
      <property name="text">
        <string notr="true">d</string>
      </property>
    </item>
    <item>
      <property name="text">
        <string notr="true">z</string>
      </property>
    </item>
  </widget>
</item>
<item row="6" column="0">
  <widget class="QLabel" name="label_6">
    <property name="text">
      <string>File name:</string>
    </property>
  </widget>
</item>
<item row="6" column="1">

```

```

<widget class="QLabel" name="fileLabel">
  <property name="sizePolicy">
    <sizepolicy hsizeType="Preferred" vsizeType="Preferred">
      <horstretch>0</horstretch>
      <verstretch>0</verstretch>
    </sizepolicy>
  </property>
  <property name="minimumSize">
    <size>
      <width>100</width>
      <height>0</height>
    </size>
  </property>
  <property name="font">
    <font>
      <underline>true</underline>
    </font>
  </property>
  <property name="text">
    <string>&lt;html&gt;&lt;a style = 'text-decoration:none' href = 'link'&gt;Select
      ...&lt;/a&gt;&lt;/html&gt;</string>
  </property>
</widget>
</item>
<item row="3" column="1">
  <widget class="QDateEdit" name="fromDateEdit">
    <property name="minimumSize">
      <size>
        <width>100</width>
        <height>0</height>
      </size>
    </property>
  </widget>
</item>
<item row="4" column="1">
  <widget class="QDateEdit" name="toDateEdit">
    <property name="minimumSize">
      <size>
        <width>100</width>
        <height>0</height>
      </size>
    </property>
  </widget>
</item>
</layout>
</item>
</layout>
</widget>
<resources/>
<connections>
  <connection>
    <sender>buttonBox</sender>
    <signal>rejected()</signal>
    <receiver>Dialog</receiver>
    <slot>reject()</slot>
  </connection>
</connections>
<hints>

```

```

    <hint type="sourcelabel">
      <x>316</x>
      <y>260</y>
    </hint>
    <hint type="destinationlabel">
      <x>286</x>
      <y>274</y>
    </hint>
  </hints>
</connection>
</connections>
</ui>

```

./forms/dataheaderform.ui

```

<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
  <class>Dialog</class>
  <widget class="QDialog" name="Dialog">
    <property name="geometry">
      <rect>
        <x>0</x>
        <y>0</y>
        <width>452</width>
        <height>265</height>
      </rect>
    </property>
    <property name="windowTitle">
      <string>Data header</string>
    </property>
    <layout class="QGridLayout" name="gridLayout">
      <item row="0" column="0">
        <layout class="QVBoxLayout" name="verticalLayout">
          <item>
            <widget class="QLabel" name="label">
              <property name="text">
                <string>Data properties</string>
              </property>
            </widget>
          </item>
          <item>
            <widget class="QListWidget" name="listWidget"/>
          </item>
        </layout>
      </item>
      <item row="1" column="0">
        <widget class="QDialogButtonBox" name="buttonBox">
          <property name="orientation">
            <enum>Qt::Horizontal</enum>
          </property>
          <property name="standardButtons">
            <set>QDialogButtonBox::Close</set>
          </property>
        </widget>
      </item>
    </layout>
  </widget>
</ui>

```

```

<resources/>
<connections>
<connection>
<sender>buttonBox</sender>
<signal>accepted()</signal>
<receiver>Dialog</receiver>
<slot>accept()</slot>
<hints>
<hint type="sourcelabel">
<x>248</x>
<y>254</y>
</hint>
<hint type="destinationlabel">
<x>157</x>
<y>274</y>
</hint>
</hints>
</connection>
<connection>
<sender>buttonBox</sender>
<signal>rejected()</signal>
<receiver>Dialog</receiver>
<slot>reject()</slot>
<hints>
<hint type="sourcelabel">
<x>316</x>
<y>260</y>
</hint>
<hint type="destinationlabel">
<x>286</x>
<y>274</y>
</hint>
</hints>
</connection>
</connections>
</ui>

```

./forms/__init__.py

./forms/downloadform.py

```

"""
Copyright_(c)_2014_Verzunov_S.N.
Institute_of_Automation_and_Information_tehnogology
NAS_of_the_Kyrgyz_Republic
All_rights_reserved.
Code_released_under_the_GNU_GENERAL_PUBLIC_LICENSE_Version_3,_June_2007
"""

from PyQt4 import QtCore, QtGui, uic
import numpy as np
import datetime as dt
import os
from interfaces.spidr import CSVDownload
from forms.progressgroup import ProgressGroup

class DownloadForm(QtGui.QDialog):
    def __init__(self, parent=None):

```

```

QtGui.QDialog.__init__(self, parent)
uic.loadUi("forms/downloadform.ui", self)
self.setModal(False)
self.parent = parent
self.fileLabel.linkActivated.connect(self.selectFile)
self.stepComboBox.currentIndexChanged.connect(self.changeStep)
self.obsComboBox.currentIndexChanged.connect(self.changeObs)
self.obsComboBox.currentIndexChanged.connect(self.changeFile)
self.fromDateEdit.dateChanged.connect(self.changeFrom)
self.fromDateEdit.dateChanged.connect(self.changeFile)
self.toDateEdit.dateChanged.connect(self.changeTo)
self.toDateEdit.dateChanged.connect(self.changeFile)
self.seriesComboBox.currentIndexChanged.connect(self.changeTo)
self.buttonBox.accepted.connect(self.accept)
self.stepComboBox.setCurrentIndex(0)

def changeStep(self, value):
    if value == 0:
        file = 'forms/resource/obsmin.csv'
        self.step = 'min'
    elif value == 1:
        file = 'forms/resource/obshr.csv'
        self.step = 'hr'
    self.observatories = np.genfromtxt(file,
                                       dtype=[ 'S5', 'S32',
                                                'f2', 'f2', 'S32'],
                                       names=('Code', 'Name',
                                              'Lat', 'Lon',
                                              'Interval'),
                                       delimiter=",",
                                       comments='#')
    self.obsComboBox.addItem(self.observatories['Name'].astype(str))
    self.obsComboBox.setCurrentIndex(0)

def changeObs(self, value):
    # import pdb; pdb.set_trace()
    interval = self.observatories['Interval'][value].astype(str)
    date1 = dt.datetime.strptime(interval[0:10], '%Y-%m-%d')
    date2 = dt.datetime.strptime(interval[-10:-1], '%Y-%m-%d')
    self.fromDateEdit.setMinimumDate(date1)
    self.toDateEdit.setMaximumDate(date2)

def changeFile(self, _):
    fileName = ''.join((
        self.observatories['Code'][self.obsComboBox.currentIndex()].astype(str),
        self.fromDateEdit.date().toString(),
        self.toDateEdit.date().toString(),
        self.seriesComboBox.currentText(),
        '.gmv')).replace('_', '')
    self.defaultFileName = ''.join((
        os.getcwd(),
        os.sep,
        'data',
        os.sep,
        fileName
    ))

```

```

        self.setFileName(self.defaultFileName)

    def selectFile(self):
        filename = QtGui.QFileDialog.getSaveFileName(self,
                                                    'Save_file',
                                                    './data',
                                                    'Geomagnetic_variations\
..... (*.gmv)')
        self.setFileName(filename)

    def setFileName(self, fileName):
        self.fileName = fileName
        self.fileLabel.setText(
            "<html><a_style_='text-decoration:none'href_='link'>\
.....{0}</a></html>".format(os.path.basename(fileName)))

    def changeFrom(self):
        self.toDateEdit.setMinimumDate(
            self.fromDateEdit.date())

    def changeTo(self):
        self.fromDateEdit.setMaximumDate(
            self.toDateEdit.date())

    def accept(self):
        code = self.observatories['Code'][self.obsComboBox.currentIndex()].astype(str)
        fromDate = self.fromDateEdit.date().toPyDate()
        toDate = self.toDateEdit.date().toPyDate()
        url = "http://spidr.ngdc.noaa.gov/spidr/servlet/GetData2?\
.....format=csv&\
.....datefrom={0}T00:00:00UTC&\
.....dateto={1}T23:59:59UTC&\
.....dataset=geom_{2}@Geom.{3}&\
.....location={4}"".replace('_', ' ').format(
            fromDate,
            toDate,
            self.seriesComboBox.currentText(),
            self.step,
            code[0:3])
        print(url)
        self.progress = ProgressGroup()
        self.message = QtGui.QLabel('Downloading_data...')
        self.formLayout.addRow(self.message, self.progress)
        self.dwl = CSVDownload(url, self.fileName)
        self.dwl.notifyProgress.connect(self.progress.setValue)
        self.dwl.loaded.connect(self.loadFile)
        self.progress.cancelled.connect(self.downloadFileTeminate)
        self.label = self.formLayout.labelForField(self.progress)
        self.dwl.start()

    def loadFile(self):
        if self.parent is not None:
            self.parent.openFile(self.fileName)
            self.close()

    def downloadFileTeminate(self):

```

```

        self.dwl.terminate()
        if self.label is not None:
            self.label.deleteLater()
        self.progress.deleteLater()

    """
    Copyright_(c)_2014_Verzunov_S.N.
    Institute_of_Automation_and_Information_tehnogology
    NAS_of_the_Kyrgyz_Republic
    All_rights_reserved.
    Code_released_under_the_GNU_GENERAL_PUBLIC_LICENSE_Version_3,_June_2007
    """

    from PyQt4 import QtCore, QtGui, uic
    from forms.mplqt4 import MyMplCanvas
    import pylab
    import datetime

    class PlotDialog(QtGui.QDialog):
        def __call__(self, wa, parent=None, title='Plotted'):
            QtGui.QWidget.__init__(self, parent)
            uic.loadUi("forms/plotdialog.ui", self)
            self.canvas = MyMplCanvas(self, width=13, height=2, dpi=100)
            self.canvasGridLayout.addWidget(self.canvas, 0,0,1,4)
            self.coordLabel.setText('')
            self.canvas.mouseMotion.connect(self.canvasMotion)
            self.canvas.canvasLeave.connect(lambda: self.coordLabel.setText(''))
            self.saveToolButton.clicked.connect(self.saveFigure)
            self.setWindowTitle(title)

        def canvasMotion(self, event):
            if event.xdata is not None and event.ydata is not None:
                self.coordLabel.setText('x=%s, y=%s' % (event.xdata, event.ydata))

        def saveFigure(self):
            title = self.windowTitle()
            self.scalogramFilename = QtGui.QFileDialog.getSaveFileName(
                None, 'Save_{ }'.format(title), 'images/{ }.png'.format(title),
                'Portable_Network_Graphics_(*.png)')
            self.signalCanvas.saveFigure(self.scalogramFilename, dpi=300)

    class PeriodogramPlotDialog(PlotDialog):
        def __init__(self, wa, parent=None, title='Periodogram'):
            PlotDialog.__call__(self, wa, parent=parent, title=title)
            wa.plotPeriodogram(self.canvas.axes)

    class ScalegramPlotDialog(PlotDialog):
        def __init__(self, wa, parent=None, title='Scalegram'):
            PlotDialog.__call__(self, wa, parent=parent, title=title)
            wa.plotScalegram(self.canvas.axes)

    class SceletonPlotDialog(PlotDialog):

```

```

def __init__(self, wa, parent=None, title='Sceleton'):
    PlotDialog.__call__(self, wa, parent=parent, title=title)
    wa.plotSceleton(self.canvas.axes)

def canvasMotion(self, event):
    if event.xdata is not None and event.ydata is not None:
        self.coordLabel.setText('x=%s, y=%s' %
                                (pylab.num2date(event.xdata).strftime(
                                    '%d.%m.%y_%H:%M'), event.ydata))

```

./forms/mainform.py

"""

Copyright_(c)_2014_Verzunov_S.N.
 Institute_of_Automation_and_Information_tehnogology
 NAS_of_the_Kyrgyz_Republic
 All_rights_reserved.
 Code_released_under_the_GNU_GENERAL_PUBLIC_LICENSE_Version_3,_June_2007
 """

```

import time
from PyQt4 import QtCore, QtGui, uic
from interfaces import spidr
from interfaces.spidr import CSVImpot
from forms.dataheaderform import DataHeaderForm
from forms.progressgroup import ProgressGroup
from forms.truescrollbar import TrueScrollBar
from forms.downloadform import DownloadForm
from forms.plotdialog import ScalegramPlotDialog, PeriodogramPlotDialog
from forms.plotdialog import SceletonPlotDialog
from forms.mplqt4 import MyMplCanvas
from processing.wavelet import WaweletAnalysis as WA
from wavelets import cwt
import datetime
import inspect
import pylab
from forms.aboutform import AboutForm

```

```

class MainForm(QtGui.QMainWindow):
    def __init__(self, application):
        super(MainForm, self).__init__()
        #self.app=application
        uic.loadUi("forms/mainform.ui", self)
        #Override VerticalScrollBar to TrueScrollBar
        self.sizeVerticalScrollBar = TrueScrollBar(self)
        self.sizeVerticalScrollBar.setMinimum(2) # min size=2**2
        self.signalGridLayout.addWidget(self.sizeVerticalScrollBar, 0, 2, 3, 1)
        self.notesVerticalScrollBar=TrueScrollBar(self)
        self.notesVerticalScrollBar.setMinimum(4)
        self.notesVerticalScrollBar.setMaximum(16)
        self.scalogramGridLayout.addWidget(self.notesVerticalScrollBar, 0, 2, 3, 1)
        self.actionQuit.triggered.connect(self.close)
        self.actionOpen.triggered.connect(self.openFile)
        self.actionDownload.triggered.connect(self.downloadFile)
        self.actionAbout.triggered.connect(self.showAbout)
        self.actionDataHeader.triggered.connect(self.showDataHeader)
        self.actionClose.triggered.connect(self.closeFile)

```



```

self.sizeVerticalScrollBar.invValueChanged.connect(self.sizeChanged)
self.offsetHorizontalScrollBar.valueChanged.connect(self.offsetChanged)
self.actionPlot_signal.triggered.connect(self.plotSignal)
self.actionSave_image_signal_as.triggered.connect(self.saveSignalAs)
self.actionSave_scalogram_as.triggered.connect(self.saveScalogramAs)
self.actionPlot_periodogram.triggered.connect(self.plotPeriodogram)
self.actionPlot_scalegram.triggered.connect(self.plotScalegram)
self.actionPlot_skeleton.triggered.connect(self.plotSkeleton)
self.offsetHorizontalScrollBar.sliderMoved.connect(self.offsetMoved)
self.sizeVerticalScrollBar.invSliderMoved.connect(self.sizeMoved)
self.scaleHorizontalScrollBar.valueChanged.connect(self.scaleCanged)
self.scaleHorizontalScrollBar.sliderMoved.connect(self.scaleMoved)
self.notesVerticalScrollBar.invValueChanged.connect(self.notesChanged)
self.notesVerticalScrollBar.invSliderMoved.connect(self.notesMoved)
self.waveletComboBox.currentIndexChanged.connect(self.replot)
self.orderSpinBox.valueChanged.connect(self.replot)
self.omegaSpinBox.valueChanged.connect(self.replot)
self.minHSpinBox.valueChanged.connect(self.minHchanged)
self.maxHSpinBox.valueChanged.connect(self.maxHchanged)
self.actionDetrend.triggered.connect(self.detrendData)
self.waveletComboBox.currentIndexChanged.connect(self.waveletChanged)
self.lock = True
for name, obj in inspect.getmembers(cwt):
    #print(obj)

    if inspect.isclass(obj):
        if obj.__base__.__name__ == 'Cwt':
            self.waveletComboBox.addItem(name, obj)

self.moveToCenter()
def canvasEnter(self):
    self.coord = QtGui.QLabel(self)
    self.statusbar.addWidget(self.coord)
def canvasLeave(self):
    self.statusbar.removeWidget(self.coord)
def canvasMotion(self, event):
    if event.xdata is not None and event.ydata is not None:
        self.coord.setText(
            'x=%s, y=%s' %
            (pylab.num2date(event.xdata).strftime('%d.%m.%y_%H:%M'),
             event.ydata))
def createCanvases(self):
    self.signalCanvas = MyMplCanvas(self, width=13, height=2, dpi=100)
    self.signalGridLayout.addWidget(self.signalCanvas, 0, 0, 3, 2)
    self.scalogramCanvas = MyMplCanvas(self, width=5, height=4, dpi=100)
    self.scalogramGridLayout.addWidget(self.scalogramCanvas, 0, 0, 3, 2)
    self.signalCanvas.canvasEnter.connect(self.canvasEnter)
    self.signalCanvas.mouseMotion.connect(self.canvasMotion)
    self.signalCanvas.canvasLeave.connect(self.canvasLeave)
    self.scalogramCanvas.canvasEnter.connect(self.canvasEnter)
    self.scalogramCanvas.mouseMotion.connect(self.canvasMotion)
    self.scalogramCanvas.canvasLeave.connect(self.canvasLeave)
def moveToCenter(self):
    screen = QtGui.QDesktopWidget().screenGeometry()
    mysize = self.geometry()
    hpos = ( screen.width() - mysize.width() ) / 2

```

```

vpos = ( screen.height() - mysize.height() ) / 2
self.move(hpos, vpos)

def openFile(self, fileName=None):
    if fileName is None or fileName == False:
        fileName = QtGui.QFileDialog.getOpenFileName(self, 'Open_file',
                                                    './data',
                                                    'Geomagnetic_variations_(*.gmv);;Solar_wind_Kp_estimation_(*.ske)')
    if QtCore.QFile.exists(fileName):
        if self.actionClose.isEnabled():
            self.closeFile()
        self.progress=ProgressGroup('Loading_data...', self.statusbar)
        self.statusbar.insertWidget(0, self.progress)
        self.csv=CSVImpot(fileName)
        self.csv.notifyProgress.connect(self.progress.setValue)
        self.csv.loaded.connect(self.loadFile)
        self.progress.cancelled.connect(self.openFileTeminate)
        self.csv.start()

def openFileTeminate(self):
    self.statusbar.removeWidget(self.progress)
    self.statusbar.showMessage('Load_cancelled_by_user!', 3000)
    self.csv.terminate()

def loadFile(self):
    self.statusbar.removeWidget(self.progress)
    self.createCanvases()
    self.wa = WA(self.csv.time, self.csv.value)
    sizePow2 = self.wa.getLengthAsPower2()
    self.sizeVerticalScrollBar.setMaximum(sizePow2)
    self.offsetMoved(0)
    self.notesVerticalScrollBar.setValue(
        self.notesVerticalScrollBar.minimum())
    self.lock = False
    self.sizeVerticalScrollBar.setValue(sizePow2)
    self.enableControlForOpen()

def sizeChanged(self, value):
    self.sizeLabel.setText('2^%s' % value)
    self.offsetHorizontalScrollBar.setMaximum(self.wa.getLength()-2**value)
    self.scaleHorizontalScrollBar.setMaximum(2**value)
    self.replot()

def scaleCanged(self, value):
    self.scaleLabel.setText(str(value))
    self.replot()

def scaleMoved(self, value):
    self.scaleLabel.setText(str(value))

def offsetMoved(self, value):
    self.offsetLabel.setText(self.wa.getDate(value).strftime('%d.%m.%y'))

def sizeMoved(self, value):
    #value = self.wa.getLengthAsPower2()-value
    self.sizeLabel.setText('2^%s' % value)

```

```

def offsetChanged(self, value):
    print('offset_chang')
    self.offsetLabel.setText(self.wa.getDate(value).strftime('%d.%m.%y'))
    self.replot()

def notesChanged(self, value):
    self.notesLabel.setText(str(value))
    self.replot()

def notesMoved(self, value):
    self.notesLabel.setText(str(value))

def plotPeriodogram(self):
    self.periodogramForm = PeriodogramPlotDialog(self.wa, parent=self)
    self.periodogramForm.show()

def plotScalegram(self):
    self.scalegramForm = ScalegramPlotDialog(self.wa, parent=self)
    self.scalegramForm.show()

def plotSceleton(self):
    self.sceletonForm = SceletonPlotDialog(self.wa, parent=self)
    self.sceletonForm.show()

def showDataHeader(self):
    self.dataHeaderForm = DataHeaderForm(self.csv.header)
    self.dataHeaderForm.show()

def showAbout(self):
    aboutForm = AboutForm(self)
    aboutForm.exec_()

def closeFile(self):
    self.clearCanvases()
    self.disableControlForClose()

def plotSignal(self):
    print('size%s'% self.sizeVerticalScrollBar.value())
    self.wa.plotSignal(self.signalCanvas.axes,
        self.offsetHorizontalScrollBar.value(),
        2**self.sizeVerticalScrollBar.value(),
        xlabel = 'Date',
        ylabel = 'nT')
    self.signalCanvas.draw()

def plotScalogram(self):
    self.progress = ProgressGroup('Plot_scalogram...', self.statusbar)
    self.statusbar.insertWidget(0, self.progress)
    self.wa.plotted.connect(self.scalogramPlotted)
    self.wa.notifyProgress.connect(self.progress.setValue)
    self.wa.cancelled.connect(self.scalogramPlotted)
    self.progress.cancelled.connect(self.wa.cancelScalogram)
    self.wa.plotScalogram(
        self.scalogramCanvas.axes,

```

```

        offset=self.offsetHorizontalScrollBar.value(),
        size=2**self.sizeVerticalScrollBar.value(),
        largestscale=self.scaleHorizontalScrollBar.value(),
        notes=self.notesVerticalScrollBar.value(),
        wavelet=self.waveletComboBox.itemData(
            self.waveletComboBox.currentIndex()),
        omega0=self.omega0SpinBox.value(),
        order=self.orderSpinBox.value(),
        min_h=self.minHspinBox.value(),
        max_h=self.maxHspinBox.value())

def scalogramPlotted(self):
    self.statusbar.removeWidget(self.progress)
    self.statusbar.showMessage('Finished.', 100)
    self.scalogramCanvas.draw()
    self.signalGroupBox.setEnabled(True)
    self.scalogramGroupBox.setEnabled(True)
    self.toolGroupBox.setEnabled(True)
    self.lock = False

def replot(self):
    if self.lock:
        return
    else:
        self.lock = True
        self.signalGroupBox.setEnabled(False)
        self.scalogramGroupBox.setEnabled(False)
        self.toolGroupBox.setEnabled(False)
        self.plotSignal()
        self.plotScalogram()

def disableControlForClose(self):
    self.lock = True
    self.signalGroupBox.setEnabled(False)
    self.scalogramGroupBox.setEnabled(False)
    self.actionClose.setEnabled(False)
    self.signalGroupBox.setEnabled(False)
    self.actionSave_image_signal_as.setEnabled(False)
    self.actionSave_scalogram_as.setEnabled(False)
    self.actionDataHeader.setEnabled(False)
    self.toolGroupBox.setEnabled(False)
    self.actionDetrend.setEnabled(False)
    self.actionPlot_periodogram.setEnabled(True)
    self.actionPlot_scalegram.setEnabled(True)

def clearCanvases(self):
    self.signalCanvas.close()
    self.scalogramCanvas.close()

def enableControlForOpen(self):
    self.signalGroupBox.setEnabled(True)
    self.scalogramGroupBox.setEnabled(True)
    self.actionSave_image_signal_as.setEnabled(True)
    self.actionSave_scalogram_as.setEnabled(True)
    self.actionPlot_periodogram.setEnabled(True)
    self.actionPlot_scalegram.setEnabled(True)

```

```

        self.actionDataHeader.setEnabled(True)
        self.actionClose.setEnabled(True)
        self.toolGroupBox.setEnabled(True)
        self.actionDetrend.setEnabled(True)
    def saveSignalAs(self):
        self.signalFilename = QtGui.QFileDialog.getSaveFileName(None, 'Save_signal',
                                                                './images/signal.png', 'Portable_Network_Graphics_(*.png)')
        self.signalCanvas.saveFigure(self.signalFilename, dpi=300)

    def saveScalogramAs(self):
        self.scalogramFilename = QtGui.QFileDialog.getSaveFileName(None, 'Save_figure',
                                                                './images/scalogram.png', 'Portable_Network_Graphics_(*.png)')
        self.scalogramCanvas.saveFigure(self.scalogramFilename, dpi=300)

    def minHchanged(self, value):
        self.maxHspinBox.setMinimum(value)
        self.replot()

    def maxHchanged(self, value):
        self.minHspinBox.setMaximum(value)
        self.replot()

    def downloadFile(self):
        self.downloadForm = DownloadForm(self)
        self.downloadForm.show()

    def detrendData(self):
        self.wa.detrend()
        self.replot()

    def waveletChanged(self, value):
        wavelet = self.waveletComboBox.itemData(value)
        if wavelet.__name__ == 'Morlet' or wavelet.__name__ == 'MorletReal':
            self.orderSpinBox.setEnabled(False)
            self.omega0SpinBox.setEnabled(True)
        else:
            self.orderSpinBox.setEnabled(True)
            self.omega0SpinBox.setEnabled(False)

```

```

#         import pdb
#         pdb.set_trace()

```

./forms/mainform.ui

```

<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
  <class>MainWindow</class>
  <widget class="QMainWindow" name="MainWindow">
    <property name="geometry">
      <rect>
        <x>0</x>
        <y>0</y>
        <width>620</width>
        <height>600</height>
      </rect>
    </property>
    <property name="sizePolicy">

```

```

<sizepolicy hsiptype="Preferred" vsiptype="Fixed">
  <horstretch>0</horstretch>
  <verstretch>0</verstretch>
</sizepolicy>
</property>
<property name="minimumSize">
  <size>
    <width>0</width>
    <height>32</height>
  </size>
</property>
<property name="windowTitle">
  <string>Analysis of magnetic variations</string>
</property>
<widget class="QWidget" name="centralwidget">
  <layout class="QGridLayout" name="gridLayout">
    <property name="topMargin">
      <number>0</number>
    </property>
    <item row="1" column="0">
      <widget class="QSplitter" name="splitter">
        <property name="orientation">
          <enum>Qt::Vertical</enum>
        </property>
        <widget class="QGroupBox" name="signalGroupBox">
          <property name="enabled">
            <bool>>false</bool>
          </property>
          <property name="title">
            <string>Signal</string>
          </property>
          <layout class="QGridLayout" name="gridLayout_2">
            <property name="leftMargin">
              <number>1</number>
            </property>
            <property name="topMargin">
              <number>3</number>
            </property>
            <property name="rightMargin">
              <number>1</number>
            </property>
            <property name="bottomMargin">
              <number>1</number>
            </property>
            <property name="spacing">
              <number>0</number>
            </property>
            <item row="0" column="0">
              <layout class="QGridLayout" name="signalGridLayout" rowstretch="0,0,0,0">
                <property name="spacing">
                  <number>1</number>
                </property>
                <item row="1" column="3">
                  <widget class="QLabel" name="sizeLabel">
                    <property name="minimumSize">
                      <size>

```

```

        <width>0</width>
        <height>0</height>
    </size>
</property>
<property name="maximumSize">
    <size>
        <width>40</width>
        <height>15</height>
    </size>
</property>
<property name="text">
    <string>2^1</string>
</property>
</widget>
</item>
<item row="3" column="2" colspan="2">
    <widget class="QLabel" name="offsetLabel">
        <property name="sizePolicy">
            <sizepolicy hsizeType="Preferred" vsizeType="Minimum">
                <horstretch>0</horstretch>
                <verstretch>0</verstretch>
            </sizepolicy>
        </property>
        <property name="minimumSize">
            <size>
                <width>60</width>
                <height>0</height>
            </size>
        </property>
        <property name="maximumSize">
            <size>
                <width>65</width>
                <height>16777215</height>
            </size>
        </property>
        <property name="text">
            <string>0</string>
        </property>
    </widget>
</item>
<item row="3" column="0">
    <widget class="QLabel" name="label_2">
        <property name="sizePolicy">
            <sizepolicy hsizeType="Minimum" vsizeType="Preferred">
                <horstretch>0</horstretch>
                <verstretch>0</verstretch>
            </sizepolicy>
        </property>
        <property name="maximumSize">
            <size>
                <width>90</width>
                <height>16777215</height>
            </size>
        </property>
        <property name="text">
            <string>Time offset:</string>

```

```

    </property>
</widget>
</item>
<item row="0" column="3">
  <widget class="QLabel" name="label_5">
    <property name="sizePolicy">
      <sizepolicy hsize="Preferred" vsize="Maximum">
        <horstretch>0</horstretch>
        <verstretch>0</verstretch>
      </sizepolicy>
    </property>
    <property name="minimumSize">
      <size>
        <width>0</width>
        <height>0</height>
      </size>
    </property>
    <property name="maximumSize">
      <size>
        <width>45</width>
        <height>16777215</height>
      </size>
    </property>
    <property name="text">
      <string>Size:</string>
    </property>
  </widget>
</item>
<item row="3" column="1">
  <widget class="QScrollBar" name="offsetHorizontalScrollBar">
    <property name="sizePolicy">
      <sizepolicy hsize="Minimum" vsize="Fixed">
        <horstretch>0</horstretch>
        <verstretch>0</verstretch>
      </sizepolicy>
    </property>
    <property name="tracking">
      <bool>false</bool>
    </property>
    <property name="orientation">
      <enum>Qt::Horizontal</enum>
    </property>
  </widget>
</item>
<item row="2" column="3">
  <spacer name="verticalSpacer">
    <property name="orientation">
      <enum>Qt::Vertical</enum>
    </property>
    <property name="sizeHint" stdset="0">
      <size>
        <width>20</width>
        <height>40</height>
      </size>
    </property>
  </spacer>

```



```

        </item>
    </layout>
</item>
</layout>
</widget>
<widget class="QGroupBox" name="scalogramGroupBox">
    <property name="enabled">
        <bool>>false</bool>
    </property>
    <property name="title">
        <string>Scalogram</string>
    </property>
    <layout class="QGridLayout" name="gridLayout_3">
        <property name="leftMargin">
            <number>3</number>
        </property>
        <property name="topMargin">
            <number>1</number>
        </property>
        <property name="rightMargin">
            <number>1</number>
        </property>
        <property name="bottomMargin">
            <number>1</number>
        </property>
        <property name="spacing">
            <number>0</number>
        </property>
        <item row="0" column="0">
            <layout class="QGridLayout" name="scalogramGridLayout" rowstretch="0,0,0,0">
                <property name="spacing">
                    <number>1</number>
                </property>
                <item row="1" column="3">
                    <widget class="QLabel" name="notesLabel">
                        <property name="minimumSize">
                            <size>
                                <width>0</width>
                                <height>0</height>
                            </size>
                        </property>
                        <property name="maximumSize">
                            <size>
                                <width>40</width>
                                <height>15</height>
                            </size>
                        </property>
                        <property name="text">
                            <string>4</string>
                        </property>
                    </widget>
                </item>
                <item row="3" column="2" colspan="2">
                    <widget class="QLabel" name="scaleLabel">
                        <property name="sizePolicy">
                            <sizepolicy hsize="Preferred" vsize="Preferred">

```

```

        <horstretch>0</horstretch>
        <verstretch>0</verstretch>
    </sizepolicy>
</property>
<property name="minimumSize">
    <size>
        <width>60</width>
        <height>0</height>
    </size>
</property>
<property name="maximumSize">
    <size>
        <width>65</width>
        <height>16777215</height>
    </size>
</property>
<property name="text">
    <string>4</string>
</property>
</widget>
</item>
<item row="3" column="0">
    <widget class="QLabel" name="label_3">
        <property name="sizePolicy">
            <sizepolicy hsiptype="Preferred" vsiptype="Preferred">
                <horstretch>0</horstretch>
                <verstretch>0</verstretch>
            </sizepolicy>
        </property>
        <property name="maximumSize">
            <size>
                <width>90</width>
                <height>16777215</height>
            </size>
        </property>
        <property name="text">
            <string>Largest scale</string>
        </property>
    </widget>
</item>
<item row="0" column="3">
    <widget class="QLabel" name="label_6">
        <property name="minimumSize">
            <size>
                <width>0</width>
                <height>0</height>
            </size>
        </property>
        <property name="maximumSize">
            <size>
                <width>45</width>
                <height>15</height>
            </size>
        </property>
        <property name="text">
            <string>Notes:</string>

```

```

        </property>
    </widget>
</item>
<item row="3" column="1">
    <widget class="QScrollBar" name="scaleHorizontalScrollBar">
        <property name="minimum">
            <number>4</number>
        </property>
        <property name="tracking">
            <bool>>false</bool>
        </property>
        <property name="orientation">
            <enum>Qt::Horizontal</enum>
        </property>
    </widget>
</item>
<item row="2" column="3">
    <spacer name="verticalSpacer_2">
        <property name="orientation">
            <enum>Qt::Vertical</enum>
        </property>
        <property name="sizeHint" stdset="0">
            <size>
                <width>20</width>
                <height>40</height>
            </size>
        </property>
    </spacer>
</item>
</layout>
</item>
</layout>
</widget>
</widget>
</item>
<item row="0" column="0">
    <widget class="QGroupBox" name="toolGroupBox">
        <property name="enabled">
            <bool>>false</bool>
        </property>
        <property name="sizePolicy">
            <sizepolicy hsize="Preferred" vsize="Preferred">
                <horstretch>0</horstretch>
                <verstretch>0</verstretch>
            </sizepolicy>
        </property>
        <property name="minimumSize">
            <size>
                <width>0</width>
                <height>50</height>
            </size>
        </property>
        <property name="title">
            <string>Parametrs</string>
        </property>
    <layout class="QGridLayout" name="gridLayout_4">

```

```

<property name="leftMargin">
  <number>1</number>
</property>
<property name="topMargin">
  <number>0</number>
</property>
<property name="rightMargin">
  <number>1</number>
</property>
<property name="bottomMargin">
  <number>1</number>
</property>
<property name="spacing">
  <number>0</number>
</property>
<item row="0" column="0">
  <layout class="QHBoxLayout" name="horizontalLayout">
    <property name="spacing">
      <number>1</number>
    </property>
    <item>
      <widget class="QLabel" name="label">
        <property name="text">
          <string>Wavelet:</string>
        </property>
      </widget>
    </item>
    <item>
      <widget class="QComboBox" name="waveletComboBox"/>
    </item>
    <item>
      <widget class="QLabel" name="label_4">
        <property name="text">
          <string>Order:</string>
        </property>
      </widget>
    </item>
    <item>
      <widget class="QSpinBox" name="orderSpinBox">
        <property name="keyboardTracking">
          <bool>false</bool>
        </property>
        <property name="minimum">
          <number>1</number>
        </property>
        <property name="maximum">
          <number>999</number>
        </property>
      </widget>
    </item>
    <item>
      <widget class="QLabel" name="label_7">
        <property name="text">
          <string>Omega0:</string>
        </property>
      </widget>
    </item>
  </layout>
</item>

```

```

</item>
<item>
  <widget class="QDoubleSpinBox" name="omega0SpinBox">
    <property name="keyboardTracking">
      <bool>false</bool>
    </property>
    <property name="maximum">
      <double>999.9900000000000009</double>
    </property>
    <property name="value">
      <double>5.0000000000000000</double>
    </property>
  </widget>
</item>
<item>
  <widget class="QLabel" name="label_8">
    <property name="text">
      <string>Range from</string>
    </property>
  </widget>
</item>
<item>
  <widget class="QSpinBox" name="minHspinBox">
    <property name="keyboardTracking">
      <bool>false</bool>
    </property>
  </widget>
</item>
<item>
  <widget class="QLabel" name="label_9">
    <property name="text">
      <string>to</string>
    </property>
  </widget>
</item>
<item>
  <widget class="QSpinBox" name="maxHspinBox">
    <property name="keyboardTracking">
      <bool>false</bool>
    </property>
    <property name="maximum">
      <number>100000</number>
    </property>
    <property name="value">
      <number>1000</number>
    </property>
  </widget>
</item>
<item>
  <spacer name="horizontalSpacer">
    <property name="orientation">
      <enum>Qt::Horizontal</enum>
    </property>
    <property name="sizeHint" stdset="0">
      <size>
        <width>40</width>

```

```

        <height>20</height>
    </size>
</property>
</spacer>
</item>
</layout>
</item>
</layout>
</widget>
</item>
</layout>
</widget>
<widget class="QMenuBar" name="menubar">
    <property name="geometry">
        <rect>
            <x>0</x>
            <y>0</y>
            <width>620</width>
            <height>21</height>
        </rect>
    </property>
    <widget class="QMenu" name="menuFile">
        <property name="title">
            <string>File</string>
        </property>
        <addaction name="actionOpen"/>
        <addaction name="actionDownload"/>
        <addaction name="actionClose"/>
        <addaction name="separator"/>
        <addaction name="actionSave_image_signal_as"/>
        <addaction name="actionSave_scalogram_as"/>
        <addaction name="separator"/>
        <addaction name="actionQuit"/>
    </widget>
    <widget class="QMenu" name="menuData">
        <property name="title">
            <string>Data</string>
        </property>
        <addaction name="actionDataHeader"/>
        <addaction name="separator"/>
        <addaction name="actionPlot_periodogram"/>
        <addaction name="actionPlot_scalegram"/>
        <addaction name="actionPlot_skeleton"/>
        <addaction name="separator"/>
        <addaction name="actionDetrend"/>
    </widget>
    <widget class="QMenu" name="menuHelp">
        <property name="title">
            <string>Help</string>
        </property>
        <addaction name="actionAbout"/>
    </widget>
    <addaction name="menuFile"/>
    <addaction name="menuData"/>
    <addaction name="menuHelp"/>
</widget>

```

```

<widget class="QStatusBar" name="statusbar">
  <property name="sizePolicy">
    <sizepolicy hsizeType="Preferred" vsizeType="Fixed">
      <horstretch>0</horstretch>
      <verstretch>0</verstretch>
    </sizepolicy>
  </property>
  <property name="minimumSize">
    <size>
      <width>0</width>
      <height>30</height>
    </size>
  </property>
</widget>
<action name="actionQuit">
  <property name="text">
    <string>Exit</string>
  </property>
</action>
<action name="actionOpen">
  <property name="text">
    <string>Open... </string>
  </property>
</action>
<action name="actionDataHeader">
  <property name="enabled">
    <bool>>false</bool>
  </property>
  <property name="text">
    <string>Data header</string>
  </property>
  <property name="visible">
    <bool>>true</bool>
  </property>
</action>
<action name="actionClose">
  <property name="enabled">
    <bool>>false</bool>
  </property>
  <property name="text">
    <string>Close</string>
  </property>
</action>
<action name="actionPlot_signal">
  <property name="text">
    <string>Plot signal</string>
  </property>
</action>
<action name="actionSave_image_signal_as">
  <property name="enabled">
    <bool>>false</bool>
  </property>
  <property name="text">
    <string>Save signal as... </string>
  </property>
</action>

```

```

<action name="actionSave_scalogram_as">
  <property name="enabled">
    <bool>false </bool>
  </property>
  <property name="text">
    <string>Save scalogram as... </string>
  </property>
</action>
<action name="actionPlot_periodogram">
  <property name="enabled">
    <bool>false </bool>
  </property>
  <property name="text">
    <string>Plot periodogram</string>
  </property>
</action>
<action name="actionPlot_scalegram">
  <property name="enabled">
    <bool>false </bool>
  </property>
  <property name="text">
    <string>Plot scalegram</string>
  </property>
</action>
<action name="actionAbout">
  <property name="text">
    <string>About... </string>
  </property>
</action>
<action name="actionDownload">
  <property name="text">
    <string>Download... </string>
  </property>
</action>
<action name="actionDetrend">
  <property name="text">
    <string>Detrend</string>
  </property>
</action>
<action name="actionPlot_phasegram">
  <property name="text">
    <string>Plot phasegram</string>
  </property>
</action>
<action name="actionPlot_skeleton">
  <property name="text">
    <string>Plot skeleton</string>
  </property>
</action>
</widget>
<resources/>
<connections/>
</ui>

./forms/progressgroup.ui
<?xml version="1.0" encoding="UTF-8"?>

```



```

<ui version="4.0">
  <class>Form</class>
  <widget class="QWidget" name="Form">
    <property name="geometry">
      <rect>
        <x>0</x>
        <y>0</y>
        <width>206</width>
        <height>26</height>
      </rect>
    </property>
    <property name="minimumSize">
      <size>
        <width>0</width>
        <height>0</height>
      </size>
    </property>
    <property name="maximumSize">
      <size>
        <width>250</width>
        <height>26</height>
      </size>
    </property>
    <property name="windowTitle">
      <string>Form</string>
    </property>
    <layout class="QGridLayout" name="gridLayout">
      <item row="0" column="0">
        <layout class="QHBoxLayout" name="layout">
          <item>
            <widget class="QLabel" name="label">
              <property name="minimumSize">
                <size>
                  <width>0</width>
                  <height>16</height>
                </size>
              </property>
              <property name="text">
                <string/>
              </property>
            </widget>
          </item>
          <item>
            <widget class="QProgressBar" name="progressBar">
              <property name="minimumSize">
                <size>
                  <width>0</width>
                  <height>16</height>
                </size>
              </property>
              <property name="value">
                <number>0</number>
              </property>
            </widget>
          </item>
        </layout>
      </item>
    </layout>
  </widget>
</ui>

```

```

<widget class="QPushButton" name="cancelButton">
  <property name="minimumSize">
    <size>
      <width>0</width>
      <height>16</height>
    </size>
  </property>
  <property name="text">
    <string>x</string>
  </property>
</widget>
</item>
</layout>
</item>
</layout>
</widget>
<resources/>
<connections/>
</ui>

```

./forms/aboutform.py

```

"""

```

```

Copyright_(c)_2014_Verzunov_S.N.
Institute_of_Automation_and_Information_tehnogology
NAS_of_the_Kyrgyz_Republic
All_rights_reserved.
Code_released_under_the_GNU_GENERAL_PUBLIC_LICENSE_Version_3,_June_2007
"""

```

```

from PyQt4 import QtCore, QtGui, uic # connect PyQt
import sys
import matplotlib
import numpy
import scipy
from PyQt4.pyqtconfig import Configuration

```

```

# Main form

```

```

class AboutForm(QtGui.QDialog):
    def __init__(self, parent=None):
        QtGui.QDialog.__init__(self, parent)
        uic.loadUi("forms/aboutform.ui", self)
        self.setModal(False)
        cfg = Configuration()
        self.pythonVer.setText('Python_ver_{0}'.format(sys.version))
        self.qtVer.setText('Qt_ver_{0}'.format(QtCore.qVersion()))
        self.matplotlibVer.setText('Matplotlib_ver_{0}'.format(
            matplotlib.__version__))
        self.pyQtVer.setText('PyQt_ver_{0}'.format(
            cfg.pyqt_version_str))
        self.numpyVer.setText('Numpy_ver_{0}'.format(
            numpy.__version__))
        self.scipyVer.setText('Scipy_ver_{0}'.format(
            scipy.__version__))

```

./forms/plotdialog.ui

```

<?xml version="1.0" encoding="UTF-8"?>

```

```

<ui version="4.0">
  <class>Dialog</class>
  <widget class="QDialog" name="Dialog">
    <property name="geometry">
      <rect>
        <x>0</x>
        <y>0</y>
        <width>400</width>
        <height>499</height>
      </rect>
    </property>
    <property name="windowTitle">
      <string>Plot</string>
    </property>
    <layout class="QGridLayout" name="gridLayout">
      <property name="topMargin">
        <number>9</number>
      </property>
      <item row="0" column="0">
        <layout class="QGridLayout" name="canvasGridLayout">
          <item row="1" column="0">
            <widget class="QLabel" name="coordLabel">
              <property name="text">
                <string>x=0, y=0</string>
              </property>
            </widget>
          </item>
          <item row="1" column="1">
            <spacer name="horizontalSpacer">
              <property name="orientation">
                <enum>Qt::Horizontal</enum>
              </property>
              <property name="sizeHint" stdset="0">
                <size>
                  <width>40</width>
                  <height>20</height>
                </size>
              </property>
            </spacer>
          </item>
          <item row="0" column="3">
            <spacer name="verticalSpacer">
              <property name="orientation">
                <enum>Qt::Vertical</enum>
              </property>
              <property name="sizeHint" stdset="0">
                <size>
                  <width>20</width>
                  <height>40</height>
                </size>
              </property>
            </spacer>
          </item>
          <item row="1" column="2">
            <widget class="QToolButton" name="saveToolButton">
              <property name="text">

```

```

        <string>Save ... </string>
    </property>
</widget>
</item>
<item row="1" column="3">
    <widget class="QPushButton" name="closeToolButton">
        <property name="text">
            <string>Close</string>
        </property>
    </widget>
</item>
</layout>
</item>
</layout>
</widget>
<resources/>
<connections>
<connection>
    <sender>closeToolButton</sender>
    <signal>clicked()</signal>
    <receiver>Dialog</receiver>
    <slot>reject()</slot>
<hints>
    <hint type="sourcelabel">
        <x>364</x>
        <y>477</y>
    </hint>
    <hint type="destinationlabel">
        <x>199</x>
        <y>249</y>
    </hint>
</hints>
</connection>
</connections>
</ui>

```

./interfaces/__init__.py

./interfaces/spidr.py

"""

Copyright_(c)_2014_Verzunov_S.N.

Institute_of_Automation_and_Information_tehnology

NAS_of_the_Kyrgyz_Republic

All_rights_reserved.

Code_released_under_the_GNU_GENERAL_PUBLIC_LICENSE_Version_3,_June_2007

"""

#http://spidr.ngdc.noaa.gov/spidr/servlet/GetData2?format=xml&datefrom=1980-01-01T00:00:00
UTC&dateto=2001-01-01T00:00:00UTC&dataset=geom_f@Geom.hr&location=BOU

import csv

from PyQt4 import QtCore

import numpy as np

import datetime as dt

import os

import urllib.request

import matplotlib.dates as dates

from scipy.signal import cspline1d, cspline1d_eval

```

import pdb
class CSVDDownload(QtCore.QThread):
    notifyProgress = QtCore.pyqtSignal(int)
    loaded = QtCore.pyqtSignal()

    def __init__(self, url, fileName):
        QtCore.QThread.__init__(self)
        self.url = url
        self.fileName = fileName

    def run(self):
        urllib.request.urlretrieve(self.url, self.fileName, self.notify)
        self.loaded.emit()

    def notify(self, blocknum, blocksize, totalsize):
        self.notifyProgress.emit(blocknum % 100)

class CSVImpot(QtCore.QThread):
    notifyProgress = QtCore.pyqtSignal(int)
    loaded = QtCore.pyqtSignal()
    def __init__(self, fileName):
        QtCore.QThread.__init__(self)
        self.fileName = fileName
        self.header=[]
        self.interpolate = True

    def run(self):
        _, fileExtension = os.path.splitext(self.fileName)
        if fileExtension == '.gmw':
            print('Geomagnetic_variation')
            with open(self.fileName, 'rt') as csvdata:
                date = []
                value = []
                for row in csv.reader(csvdata):
                    if ('#' in row[0]):
                        self.header.append(row)
                    else:
                        date.append(row[0])
                        value.append(row[1])
            self.notifyProgress.emit(20)
        elif fileExtension == '.ske':
            print('Kp_estimation')
            with open(self.fileName, 'rt') as csvdata:
                date = []
                value = []
                for row in csv.reader(csvdata, delimiter=','):
                    if ('#' in row[0]):
                        self.header.append(row)
                    else:
                        print(row)
                        if int(row[7]) < 2:
                            date.append(
                                dt.datetime.strptime(
                                    ''.join((row[0], row[1], row[2],
                                                row[4])),

```

```

        '%Y%m%d%H%M') ),
        value.append( float(row[-1]) - float(row[-14])) #4h
        # value.append( float(row[-1]) - float(row[19])) # 1h
    self.notifyProgress.emit(20)
    signal_src = np.array((date, value), dtype=np.dtype('a25'))
    signal = signal_src[:, np.logical_not(
        np.isnan(signal_src[1, :].astype(np.float)))]
    # self.value=np.nan_to_num(self.value)
    self.notifyProgress.emit(60)
    if self.interpolate:
        self.time = signal_src[0, :].astype(np.datetime64).astype(dt.datetime)
        dx = dates.date2num(self.time[1]) - dates.date2num(self.time[0])
        cj = cspline1d(signal[1, :].astype(float))
        self.value = cspline1d_eval(cj, dates.date2num(self.time),
                                    dx=dx,
                                    x0=dates.date2num(self.time[0]))

        #pdb.set_trace()
    else:
        self.time = dates.signal[0, :].astype(np.datetime64).astype(dt.datetime)
        self.value = signal[1, :].astype(np.float)
    self.notifyProgress.emit(80)
    self.loaded.emit()

def __del__(self):
    self.wait()

```

./processing/___init___py

./processing/wavelet.py

"""

Copyright_(c)_2014_Verzunov_S.N.
 Institute_of_Automation_and_Information_tehnogology
 NAS_of_the_Kyrgyz_Republic
 All_rights_reserved.
 Code_released_under_the_GNU_GENERAL_PUBLIC_LICENSE_Version_3,_June_2007
 """

```

import numpy as np
import pylab as plb
import datetime as dt
import wavelets.cwt as wave
import time as profiler
from scipy.ndimage.filters import maximum_filter, minimum_filter
from scipy.ndimage.morphology import generate_binary_structure, binary_erosion
from PyQt4 import QtCore

```

```

class WaveletTransform(QtCore.QThread):
    notifyProgress = QtCore.pyqtSignal(int)
    transformed = QtCore.pyqtSignal(wave.Cwt)

    def __init__(self, data, wavelet=wave.Morlet,
                scaling='log', notes=8, largestscale=4, order=2., omega0=5.):
        QtCore.QThread.__init__(self)
        self._wavelet = wavelet
        self._scaling = scaling
        self._notes = notes

```

```

self._largestscale = largestscale
self._order = order
self._omega0 = omega0
self._data = data

def run(self):
    cw = self._wavelet(self._data, self.transformed, self.notifyProgress,
                        scaling=self._scaling, notes=self._notes,
                        omega0=self._omega0, largestscale=self._largestscale,
                        order=self._order)

    return cw

class WaveletAnalysis(QtCore.QObject):
    notifyProgress = QtCore.pyqtSignal(int)
    plotted = QtCore.pyqtSignal()
    cancelled = QtCore.pyqtSignal()
    def __init__(self, time, values):
        QtCore.QObject.__init__(self)
        self._time=time
        self._values=values
        self._maxLength=1<<((self._values.shape[-1]-1).bit_length()-1)

    def plotSignal(self, axes, offset, size, xlabel='', ylabel='', style='-'):
        axes.plot_date(self._time[offset:offset+size],
                       self._values[offset:offset+size], style)
        #yearsFmt = plb.DateFormatter(dataFormatter)
        #axes.xaxis.set_major_formatter(yearsFmt)
        #axes.set_xlabel(xlabel)
        #axes.set_ylabel(ylabel)
    def _plotScalogram(self, cw):
        self._cw=cw
        #start=profiler.time()
        scales=cw.getscales()
        cwt=cw.getdata()
        pwr=cw.getpower()
        # pwr=cw.getangle()*1e20
        #scalespec=np.sum(pwr,axis=1)/scales # calculate scale spectrum
        #scalespec=np.sum(np.anglpwr,axis=1)/scales # calculate scale spectrum
        # scales
        y=cw.fourierwl*scales
        #x=np.arange(Nlo*1.0,Nhi*1.0,1.0)
        #mpl.xlabel('Date')
        #mpl.ylabel('Period, %s' % p_label)
        plotcwt = np.clip(pwr, self._min_h, self._max_h)
        self._axes.imshow(plotcwt, cmap=plb.cm.hot_r,
                           extent=[plb.date2num(self._x[0]), plb.date2num(self._x[-1]),
                                   y[-1], y[0]], aspect='auto', interpolation=None)
        self._axes.xaxis_date()
        #yearsFmt = mpl.DateFormatter('%m.%y')
        #axes.xaxis.set_major_formatter(yearsFmt)
        #mpl.gcf().autofmt_xdate()
        if self._scaling=="log": self._axes.set_yscale('log')
        self._axes.set_ylim(y[0], y[-1])
        #print('Plot - %.03f s' % (profiler.time()-start))
        self.plotted.emit()

```

```

def plotScalogram(self , axes , size , offset , max_h=1000. , min_h=0. , p_label='', s_label='',
, wavelet=wave.Morlet , scaling='log' ,
order=2 , omega0=5. , notes=4 , largestscale=4):
    print(size)
    print(largestscale)
    self._y=self._values[offset:offset+size]
    self._x=self._time[offset:offset+size]
    self._min_h=min_h
    self._max_h=max_h
    self._axes=axes
    self._scaling=scaling
    self._wt=WaveletTransform(self._y, wavelet=wavelet , scaling=scaling ,
                               notes=notes , largestscale=size//largestscale , order=order ,
                               omega0=omega0)
    self._wt.transformed.connect(self._plotScalogram)
    self._wt.notifyProgress.connect(self._notifyProgress)
    self._wt.terminated.connect(lambda: self.cancelled.emit())
    self._wt.start()

def plotPeriodogram(self , axes , xlabel='Power' ,
                    ylabel='Period' , scaling='log'):
    # projected fourier spectrum
    axes.set_xlabel(xlabel)
    axes.set_ylabel(ylabel)
    # vara = 1.0
    f = np.fft.fftfreq(self._x.shape[-1])
    fspec = np.abs(np.fft.fft(self._y))
    u = np.abs(fspec)[0:-self._x.shape[-1]/2]
    v = 1/f[0:-self._x.shape[-1]/2]
    # w=np.ones(win_len , 'd')
    # s=np.convolve(w/w.sum() , u , mode='valid')
    # sv=v[win_len/2:-win_len/2+1]
    # print(len(s) , len(sv))
    if scaling == 'log':
        axes.loglog(u , v , 'b-') # , s , sv , 'g-'
    else:
        axes.semilogx(u , v , 'b-') # , s , sv , 'g-'
        axes.set_xlim(1e-1 , np.max(fspec))
        axes.set_ylim(self._y[0] , self._y[-1])

def plotScalegram(self , axes , xlabel='Power' ,
                  ylabel='Period' , scaling='log' , min_h=0. , max_h=1000.):
    pwr = self._cw.getpower()
    scales = self._cw.getscales()
    scalespec = np.sum(pwr , axis=1)/scales # calculate scale spectrum
    axes.set_xlabel('Power')
    axes.set_ylabel('Period')
    vara = 1.0
    y = self._cw.fourierwl*scales
    if scaling == "log":
        axes.loglog(scalespec/vara+0.01 , y , 'b-')
    else:
        axes.semilogx(scalespec/vara+0.01 , y , 'b-')
    axes.set_xlim(1e-1 , np.max(scalespec))
    axes.set_ylim(y[0] , y[-1])

```



```

def plotSceleton(self, axes, xlabel='Power',
                  ylabel='Period', scaling='log', min_h=0., max_h=1000.):
    cw = self._cw

    scales = cw.getscales()
    pwr = self.getSceleton(cw.getpower())
    y = cw.fourierwl*scales
    #plotcwt1 = np.clip(pwr[0], self._min_h, self._max_h)
    #plotcwt2 = np.clip(pwr[1], self._min_h, self._max_h)
    axes.imshow(pwr[0], cmap=plt.cm.hot_r,
                 extent=[plt.date2num(self._x[0]), plt.date2num(self._x[-1]),
                         y[-1], y[0]], aspect='auto', interpolation=None)
    axes.xaxis_date()
    axes.imshow(pwr[1], cmap=plt.cm.hot_r,
                 extent=[plt.date2num(self._x[0]), plt.date2num(self._x[-1]),
                         y[-1], y[0]], aspect='auto', interpolation=None)
    axes.xaxis_date()
    if scaling == "log":
        axes.set_yscale('log')
    axes.set_ylim(y[0], y[-1])

def cancelScalogram(self):
    self._wt.terminate()

def _notifyProgress(self, value):
    self.notifyProgress.emit(value)

def getMaxLengthAsPower2(self):
    return (self._values.shape[-1]-1).bit_length()-1

def getLength(self):
    return self._values.shape[-1]

def getDate(self, index):
    return self._time[index]

def detrend(self):
    self._values = plt.detrend(self._values, key='linear')

def getSceleton(self, im):
    imp1 = np.pad(im, ((1, 1), (0, 0)), 'minimum')
    imp0 = np.pad(im, ((0, 0), (1, 1)), 'minimum')
    row = (np.diff(np.sign(np.diff(imp0, axis=1))), axis=1) < 0)
    col = (np.diff(np.sign(np.diff(imp1, axis=0))), axis=0) < 0)
    return (row*im, col*im)

```