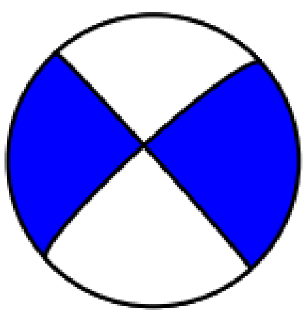
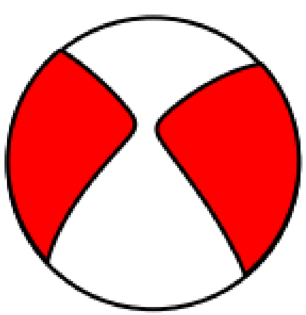
Obspy seminar (www.obspy.org)

obspy.imaging

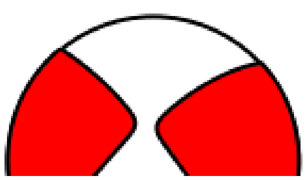
obspy.imaging.beachball: Beachball plot

In [1]:
```
from obspy.imaging.beachball import Beachball

#moznost zadat uhly
fm2=[318, 88, -169] #strike,dip,rake
Beachball(fm2)
#moznost zadat momentovy tenzor:
# m11,m22,m33,m12,m13,m23
fm=[-0.33, -2.23, 2.55, 0.83, -0.32, 0.32]
Beachball(fm,facecolor='r')
```





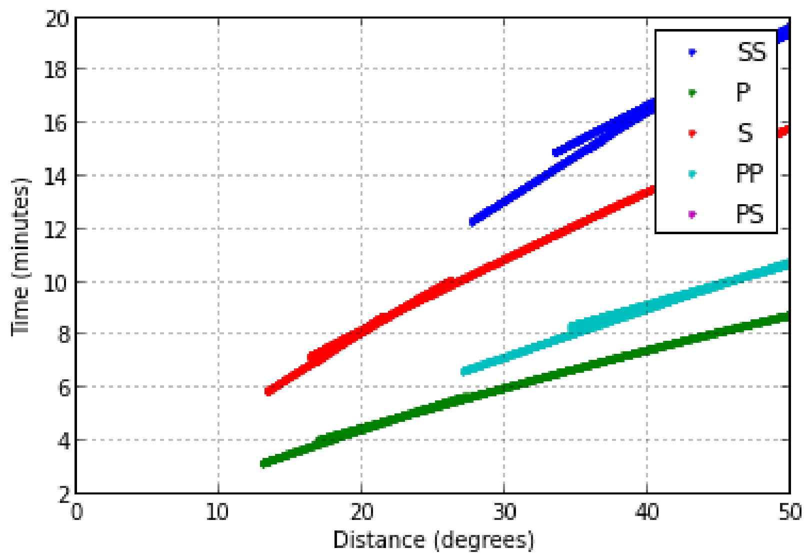
Out[1]:



obspy.taup

obspy.taup.taup: Travel time calculation tool

In [2]:
```
from obspy.taup.taup import travelTimePlot, getTravelTimes
#vykreslenie hodochron:
travelTimePlot(max_degree=50,phases=['P','S','PP','PS','SS'])
#vypocet casu sirenia a.i. v danej epicentralnej vzdialenosti
tt=getTravelTimes(delta=40.,depth=100.,model='iasp91')
#model moze byt iasp91,ak135. prem to nevie!
print tt[0] #vystup: rozne fazy
print tt[1]
```



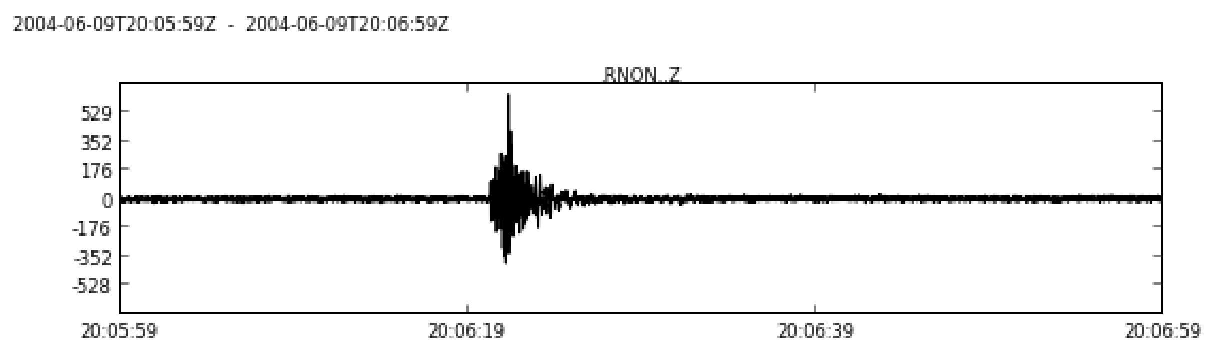

```
{'phase_name': 'P', 'dT/dD': 8.2632751, 'take-off angle':
37.41507, 'time': 444.74866, 'd2T/dD2': -0.0045646443, 'dT/dh':
-0.098694056}
{'phase_name': 'pP', 'dT/dD': 8.3460588, 'take-off angle':
142.14453, 'time': 467.83313, 'd2T/dD2': -0.0045246622,
'dT/dh': 0.098110832}
```

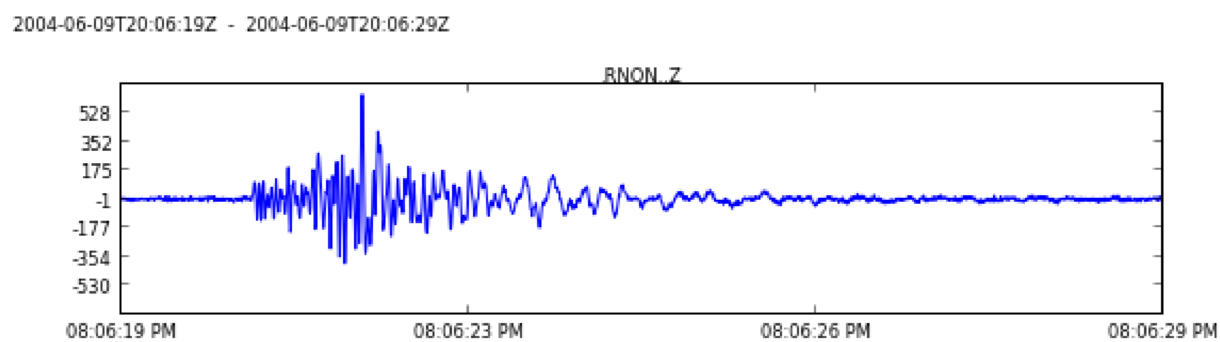
obspy.signal

```
In [3]: #nacitame data
from obspy.core import read
st = read("./event.gse2")
tr = st[0]

print tr.stats
```

```
         network: 
         station: RNON
        location: 
         channel: Z
       starttime: 2004-06-09T20:05:59.849998Z
         endtime: 2004-06-09T20:06:59.844998Z
   sampling_rate: 200.0
           delta: 0.005
            npts: 12000
           calib: 0.596000015736
         _format: GSE2
            gse2: AttribDict({'instype': '      ', 'datatype': 'CM6', 'hang': -1.0, 'auxid': 'RNON', 'vang': -1.0, 'calper': 1.0})
```

```
In [4]: #vykreslenie pre kontrolu a kopia
tr.plot()
tr2=tr.copy()
tr2.plot(color='blue',tick_format='%I:%M:%S %p',
         starttime=st[0].stats.starttime+20,
         endtime=st[0].stats.endtime-30,
         outfile='/home/mess/trplot.png')
```



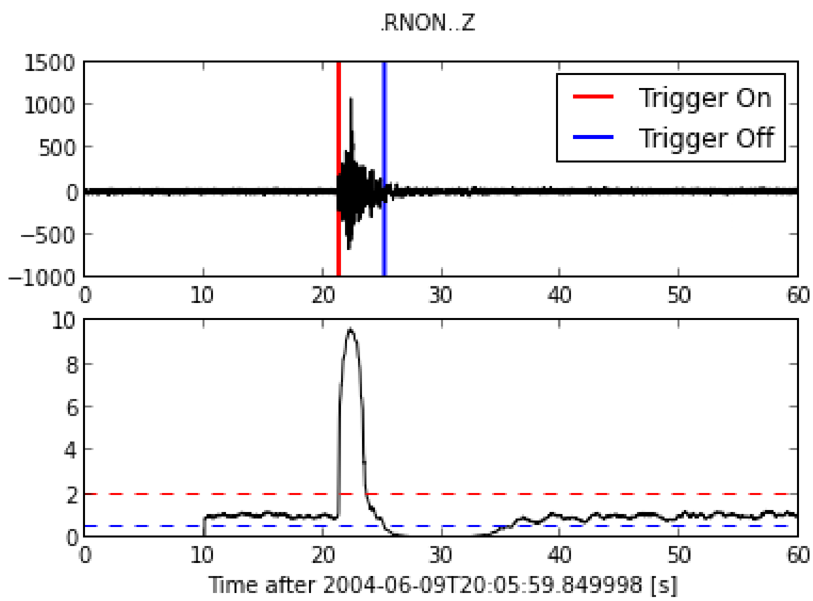


Trigger/Picker: obspy.signal.trigger

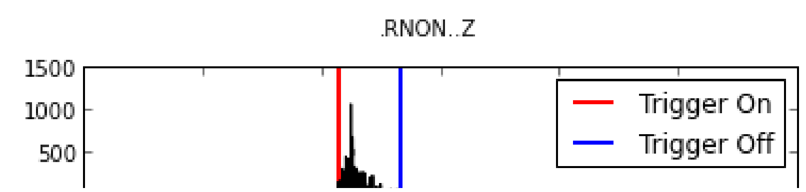
In [5]:
```
#Metody z obspy.signal.trigger - mnoho
#skusime classicSTALTA(array,nsta,nlta):
from obspy.signal.trigger import classicSTALTA
t_sta=1. #casove okno pre sta
t_lta=10. #casove okno pre lta
samp=tr.stats.sampling_rate #vzorkovacia frekvencia
cft=classicSTALTA(tr.data,int(t_sta*samp),int(t_lta*samp))
print cft #vystup je charakteristicka funkcia v tvar np.array
```

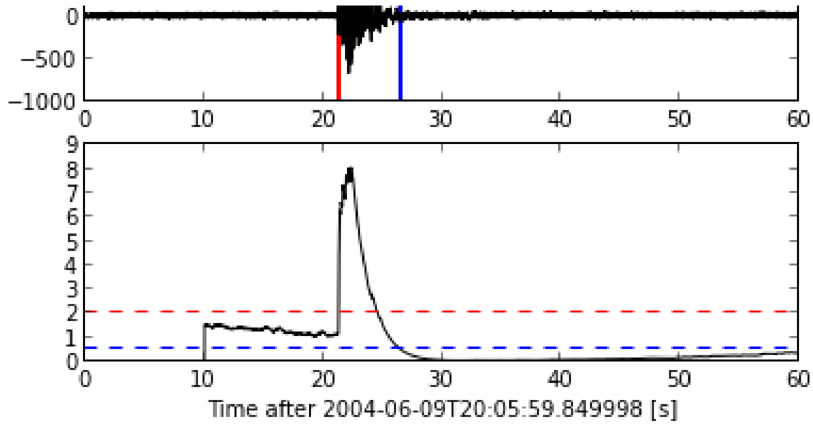
```
[ 0.          0.          0.         ...,  0.91793631
0.91849621
  0.9185254 ]
```

In [6]:
```
#vykreslenie cft
from obspy.signal.trigger import plotTrigger
thr_on=2. #threshold pre trigger ON
thr_off=.5 #threshold fpre trigger OFF
plotTrigger(tr,cft,thr_on,thr_off)
```



In [7]:
```
#Dalsia metoda: rekurzivny STA/LTA
from obspy.signal.trigger import recSTALTA, plotTrigger
cft2=recSTALTA(tr.data,int(t_sta*samp),int(t_lta*samp))
plotTrigger(tr,cft2,thr_on,thr_off)
```





In [8]:

```
#Ziskat hodnoty pre ON a OFF pre dane thresholds
from obspy.signal.trigger import triggerOnset
trg=triggerOnset(cft,thr_on,thr_off)
trg2=triggerOnset(cft2,thr_on,thr_off)
#porovnajme metody:
print trg/samp #vystup bude takto v sekundach
print trg2/samp
```

```
[[ 21.28  25.13]]
[[ 21.28  26.49]]
```

Coincidence trigger - ziska zoznam prekryvajucich sa triggerov na sieti stanic

In [9]:

```
from obspy.signal import coincidenceTrigger
#nenacitame dalsie data, ale
#skopirujeme to iste
#spustame coinc. trigger na 2 rovnake zaznamy.
st2=st.copy() #kopia
st2=st.copy()+st2 #vyrobime dalsi trace
#premenujeme stanicu na druhom trace, ma to detekciu
st2[1].stats.station="RNO2"

coinc_sum=2 #minimalny pocet prekryvov
#este sa zadavaju parametre pre trigger
trig=coincidenceTrigger('classicSTALTA',thr_on,thr_off,st2,
                        coinc_sum,sta=t_sta,lta=t_lta)

print trig # vystup
```

```
[{'duration': 3.8500001430511475, 'coincidence_sum': 2.0,
'stations': ['RNO2', 'RNON'], 'trace_ids': ['.RNO2..Z',
'.RNON..Z'], 'time': UTCDateTime(2004, 6, 9, 20, 6, 21,
129998)}]
```

Baer Picker

In [10]:

```
from obspy.signal.trigger import pkBaer
#nejake nastavenia:
tdownmax=25
```

```
min_nr_samples=100
sigma=12.
preset=100
p_pick, phase_info = pkBaer(tr.data,samp,tdownmax,
                            min_nr_samples,thr_on,sigma,preset,6
print p_pick/samp #vystup v s
```

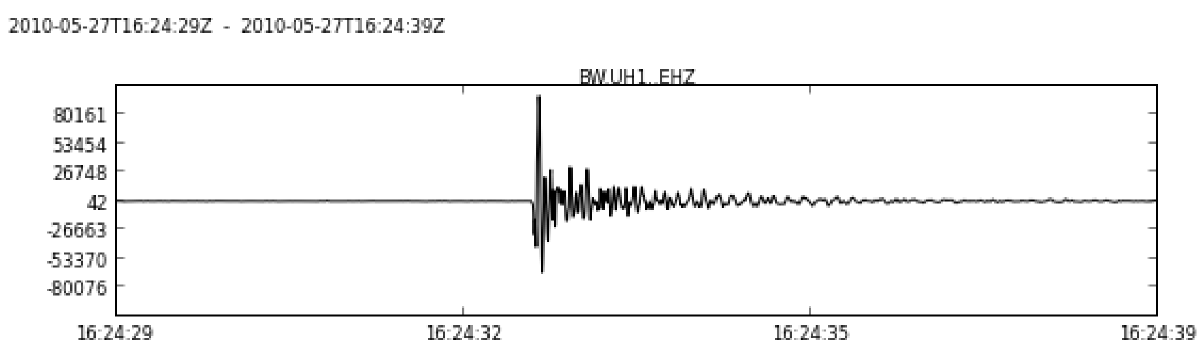
21.27

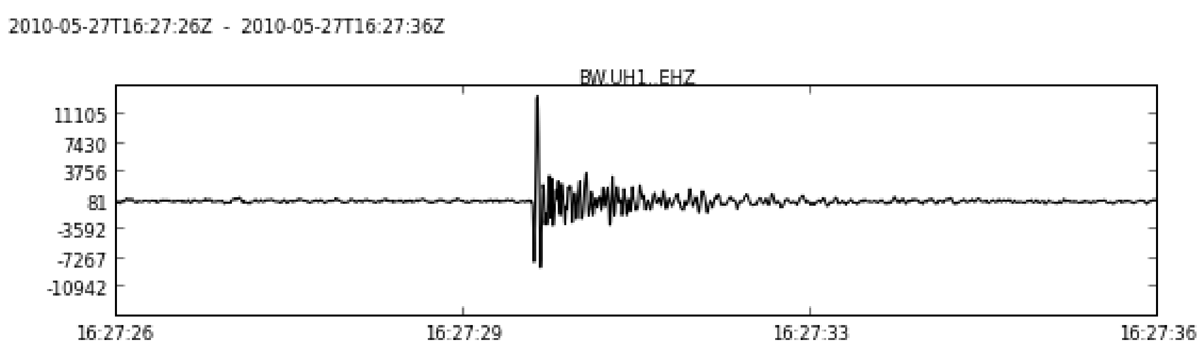
obspy.signal.cross_correlation

In [11]:

```
#nacitame data
from obspy.core import read
st1=read("/home/mess/Downloads/BW.UH1..EHZ.D.2010.147.a.slist.gz
st2=read("/home/mess/Downloads/BW.UH1..EHZ.D.2010.147.b.slist.gz
#st1 = read("http://examples.obspy.org/BW.UH1..EHZ.D.2010.147.a.
#st2 = read("http://examples.obspy.org/BW.UH1..EHZ.D.2010.147.b.
tr1=st1[0]
tr2=st2[0]
print tr1,tr2
tr1.plot(),tr2.plot() #vykreslime aby sme videli zaznamy
```

BW.UH1..EHZ | 2010-05-27T16:24:29.315000Z - 2010-05-27T16:24:39.315000Z | 200.0 Hz, 2001 samples BW.UH1..EHZ | 2010-05-27T16:27:26.585000Z - 2010-05-27T16:27:36.585000Z | 200.0 Hz, 2001 samples





Out[11]: (None, None)

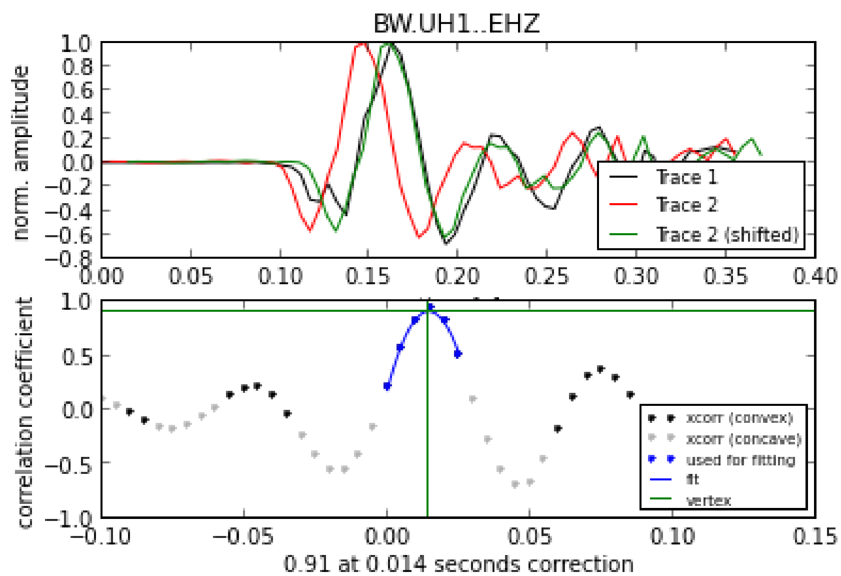
xcorr

In [12]:
```
from obspy.signal.cross_correlation import xcorr, xcorr_max
tr1cp=tr1.copy()
tr2cp=tr2.copy()
#varovanie od autorov:
#!!!shift_len has to be selected carefully,
#!!!make it a bit bigger than the highest signal shifts
#!!!that can ever occur
index, maxim, xcfct = xcorr(tr1.data,tr2.data,
                            shift_len=500,full_xcorr=True)

print index/samp,maxim #kedy a ake max je dosiahnute v xcorrela
print xcfct #vystup je v tvare np.array
```

```
0.015 0.904679169228
[ 0.00834656  0.00145004 -0.00579788 ..., -0.00946704
-0.01712502
 -0.02328115]
```

xcorrPickCorrection

In [13]:
```
#oprava pickovaneho casu pomocou kroskorelacie s inym zaznamom
from obspy.core import UTCDateTime
from obspy.signal.cross_correlation import xcorrPickCorrection
from obspy.signal.trigger import recSTALTA
from obspy.signal.trigger import triggerOnset,plotTrigger

#urcit pick1, vyuzijeme trigger!
t_sta=.1
t_lta=1.
samp=tr1cp.stats.sampling_rate
thr_on=4.
thr_off=0.5
cft1=recSTALTA(tr1cp.data,int(t_sta*samp),int(t_lta*samp))
trg1=triggerOnset(cft1,thr_on,thr_off)
pick1=trg1[0][0]/samp
#plotTrigger(tr1cp,cft1,thr_on,thr_off)

t1=tr1.stats.starttime+pick1
#zly pick pre druhy zaznam (aby mal co opravit):
#pouzijeme pick pre prvy zaznam
t2=tr2.stats.starttime+pick1

#oprava pre t2 o dt
#aj vykreslenie
dt,coeff = xcorrPickCorrection(t1,tr1,t2,tr2,t_before=0.05,
                               t_after=0.2,cc_maxlag=0.1,plot=Tr
#vystup: dt & korelacny koeficient pre posunuty zaznam
print dt,coeff
```




```
-0.0144620095442 0.914092955583
```

PPSD = Probabilistic Power Spectral Densities

```
Class to compile probabilistic power spectral densities for one
combination of network/station/location/channel/sampling_rate
```

In [14]:

```
#nacitat data
from obspy.core import read
from obspy.xseed import Parser
from obspy.signal import PPSD
st = read("/home/mess/Downloads/BW.KW1..EHZ.D.2011.037")
print st
```

```
3 Trace(s) in Stream:
BW.KW1..EHZ | 2011-02-06T00:00:00.935000Z - 2011-02-
06T05:07:21.115000Z | 200.0 Hz, 3688037 samples
BW.KW1..EHZ | 2011-02-06T05:50:15.079999Z - 2011-02-
06T06:07:21.514999Z | 200.0 Hz, 205288 samples
BW.KW1..EHZ | 2011-02-06T07:49:55.940000Z - 2011-02-
07T00:00:01.130000Z | 200.0 Hz, 11641039 samples
```

In [15]:

```
#nacitat poles&zeroes
parser = Parser("/home/mess/Downloads/dataless.seed.BW_KW1")
paz=parser.getPAZ("BW.KW1..EHZ")
print paz
```

```
{'sensitivity': 465550000.0, 'digitizer_gain': 629121.0,
'seismometer_gain': 740.0, 'zeros': [0j, 0j, (-434.1+0j)],
'gain': 818400000000.0, 'poles': [(-0.03691+0.03712j),
(-0.03691-0.03712j), (-371.2+0j), (-373.9+475.5j), (-373.9-
475.5j), (-588.4+1508j), (-588.4-1508j)]}
```

In [16]:

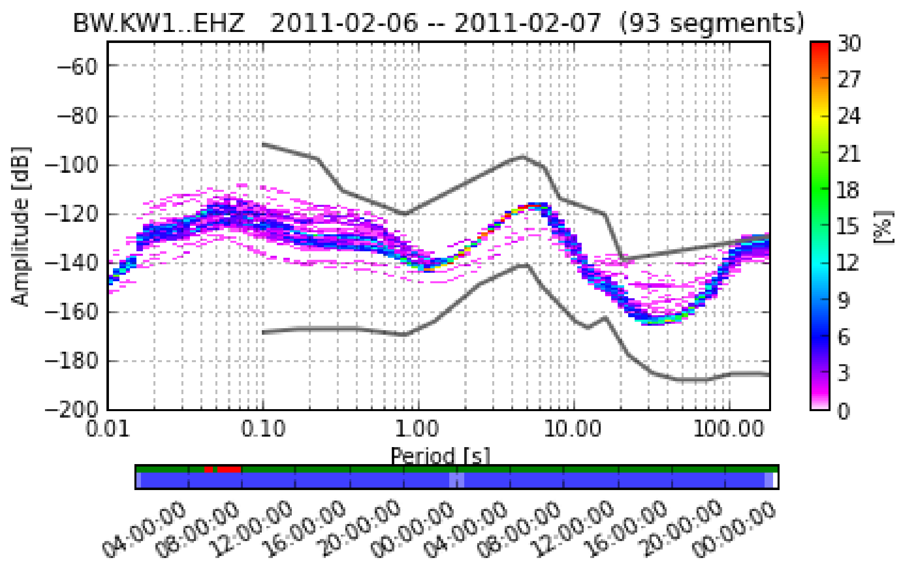
```python
#tr2=st[0]
st2=st.copy()
tr2=st2[0]

#ak chceme rychly vypocet...
#st2=st[0:1]  #len prvy trace
#st2.decimate(factor=2) #downsample na polovicnu frekvenciu
#print st2

#inicializuje ppsd:
ppsd=PPSD(tr2.stats,paz)
#pridat data do ppsd:
ppsd.add(st2)
print ppsd.times
#rozdeli data do 1hodinovych segmentov
# & preprocessing(demean,taper..)
```

```
[UTCDateTime(2011, 2, 6, 0, 0, 0, 935000), UTCDateTime(2011, 2,
6, 0, 30, 0, 935000), UTCDateTime(2011, 2, 6, 1, 0, 0, 935000),
UTCDateTime(2011, 2, 6, 1, 30, 0, 935000), UTCDateTime(2011, 2,
6, 2, 0, 0, 935000), UTCDateTime(2011, 2, 6, 2, 30, 0, 935000),
UTCDateTime(2011, 2, 6, 3, 0, 0, 935000), UTCDateTime(2011, 2,
6, 3, 30, 0, 935000), UTCDateTime(2011, 2, 6, 4, 0, 0, 935000),
UTCDateTime(2011, 2, 6, 4, 30, 0, 935000), UTCDateTime(2011, 2,
6, 5, 0, 0, 935000), UTCDateTime(2011, 2, 6, 5, 30, 0, 935000),
UTCDateTime(2011, 2, 6, 6, 0, 0, 935000), UTCDateTime(2011, 2,
6, 6, 30, 0, 935000), UTCDateTime(2011, 2, 6, 7, 0, 0, 935000),
UTCDateTime(2011, 2, 6, 7, 30, 0, 935000), UTCDateTime(2011, 2,
6, 8, 0, 0, 935000), UTCDateTime(2011, 2, 6, 8, 30, 0, 935000),
UTCDateTime(2011, 2, 6, 9, 0, 0, 935000), UTCDateTime(2011, 2,
6, 9, 30, 0, 935000), UTCDateTime(2011, 2, 6, 10, 0, 0,
935000), UTCDateTime(2011, 2, 6, 10, 30, 0, 935000),
UTCDateTime(2011, 2, 6, 11, 0, 0, 935000), UTCDateTime(2011, 2,
6, 11, 30, 0, 935000), UTCDateTime(2011, 2, 6, 12, 0, 0,
935000), UTCDateTime(2011, 2, 6, 12, 30, 0, 935000),
UTCDateTime(2011, 2, 6, 13, 0, 0, 935000), UTCDateTime(2011, 2,
6, 13, 30, 0, 935000), UTCDateTime(2011, 2, 6, 14, 0, 0,
935000), UTCDateTime(2011, 2, 6, 14, 30, 0, 935000),
UTCDateTime(2011, 2, 6, 15, 0, 0, 935000), UTCDateTime(2011, 2,
6, 15, 30, 0, 935000), UTCDateTime(2011, 2, 6, 16, 0, 0,
935000), UTCDateTime(2011, 2, 6, 16, 30, 0, 935000),
UTCDateTime(2011, 2, 6, 17, 0, 0, 935000), UTCDateTime(2011, 2,
6, 17, 30, 0, 935000), UTCDateTime(2011, 2, 6, 18, 0, 0,
935000), UTCDateTime(2011, 2, 6, 18, 30, 0, 935000),
UTCDateTime(2011, 2, 6, 19, 0, 0, 935000), UTCDateTime(2011, 2,
6, 19, 30, 0, 935000), UTCDateTime(2011, 2, 6, 20, 0, 0,
935000), UTCDateTime(2011, 2, 6, 20, 30, 0, 935000),
UTCDateTime(2011, 2, 6, 21, 0, 0, 935000), UTCDateTime(2011, 2,
6, 21, 30, 0, 935000), UTCDateTime(2011, 2, 6, 22, 0, 0,
935000), UTCDateTime(2011, 2, 6, 22, 30, 0, 935000),
UTCDateTime(2011, 2, 6, 23, 0, 0, 935000)]
```

In [17]:
```
print len(ppsd.times) #pocet hodinovych sekvencii
#a dalsie data
st3=read("/home/mess/Downloads/BW.KW1..EHZ.D.2011.038")
ppsd.add(st3) #ma detekciu,nemozno pridat dvakrat tie iste data

print len(ppsd.times) #po pridani dat
```

```
47
93
```

In [18]:
```
#vykreslit vysledok:
ppsd.plot()
```

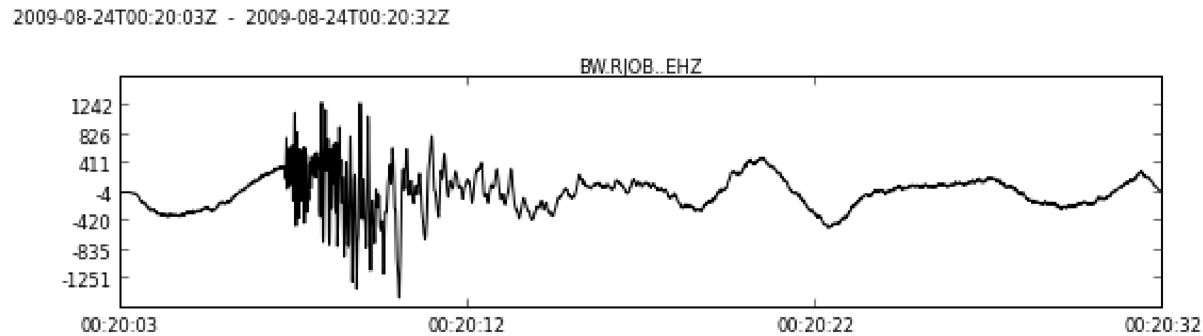


obspy.signal.tf_misfit

Continuous wavelet transform

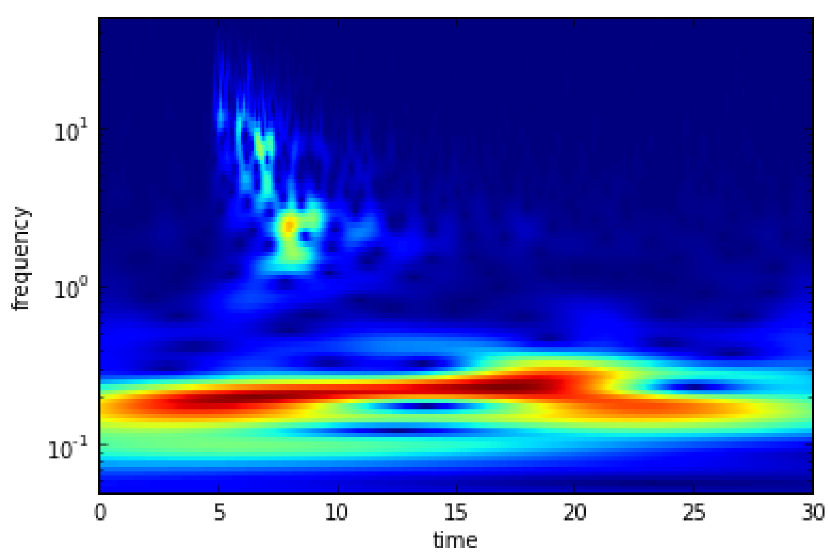
In [28]:
```
from obspy.signal.tf_misfit import cwt
#data
st3=read()#"./G.SCZ..BHE.sac")
tr3=st3[0]

#vstupne parametre
dt=tr3.stats.delta
fmin=0.05
fmax=50
w0=6 #hodnota odporucana z literatury

scalogram=cwt(tr3.data,dt,w0,fmin,fmax) #zatial len Morlet
```

```
tr3.plot()
```



In [27]:
```
#vykreslenie pomocou matplotlib.
import numpy as np
import matplotlib.pyplot as plt
fig=plt.figure()
ax=fig.add_subplot(111)
t0=0.
tmax=dt*tr3.stats.npts
ax.imshow(np.abs(scalogram)[-1::-1],extent=[t0,tmax,fmin,fmax],
          interpolation='nearest')
ax.set_xlabel('time')
ax.set_ylabel('frequency')
ax.set_yscale('log')
plt.show
```

Out[27]:
```
<function matplotlib.pyplot.show>
```

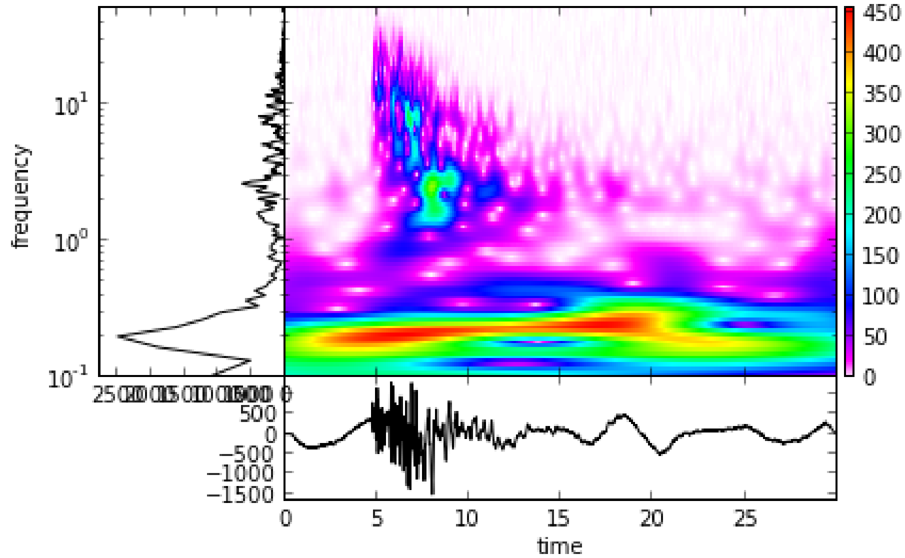


TF reprezentacia

In [21]:
```
#vykreslit TF reprezentaciu
from obspy.signal.tf_misfit import plotTfr
```

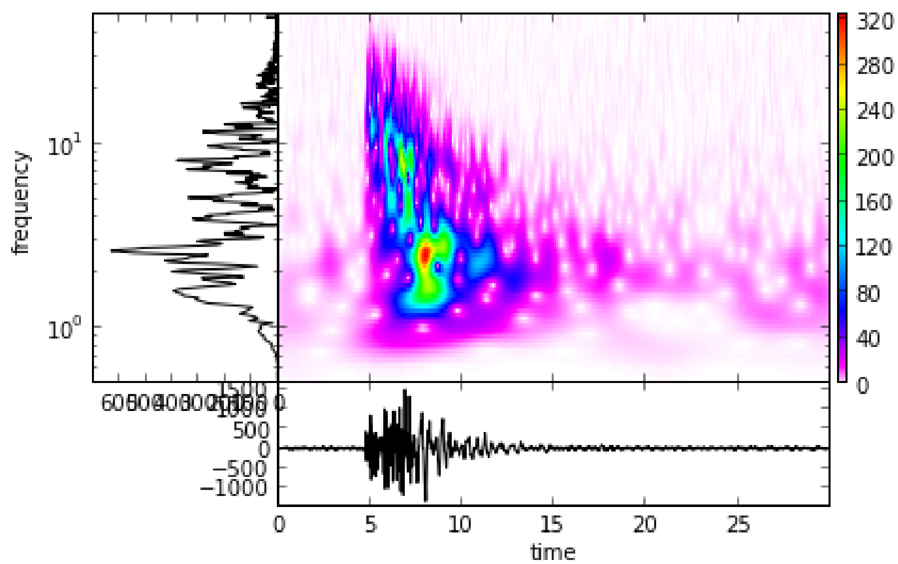
```
plotTfr(tr3.data,dt,fmin=.1,fmax=50.)
```

(1, 100, 3000)



In [22]:
```
#a dalsia TF reprezentacia: pre filtrovany signal
tr3_filt=tr3.copy()
#odfiltrujeme dlhovlnny signal:
tr3_filt.filter('highpass',freq=1.,zerophase=True)
plotTfr(tr3_filt.data,dt,fmin=0.5,fmax=50.)
```

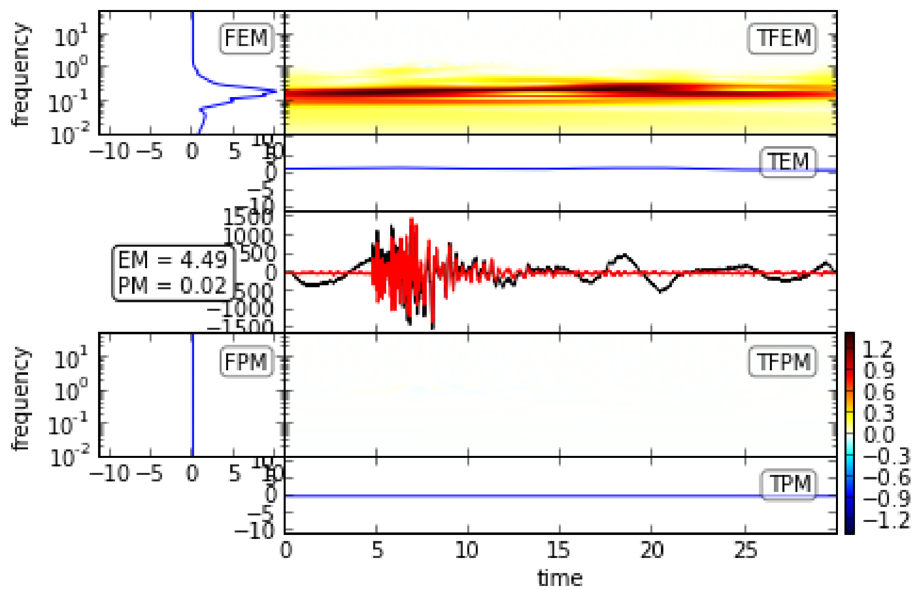
(1, 100, 3000)



TF Misfits

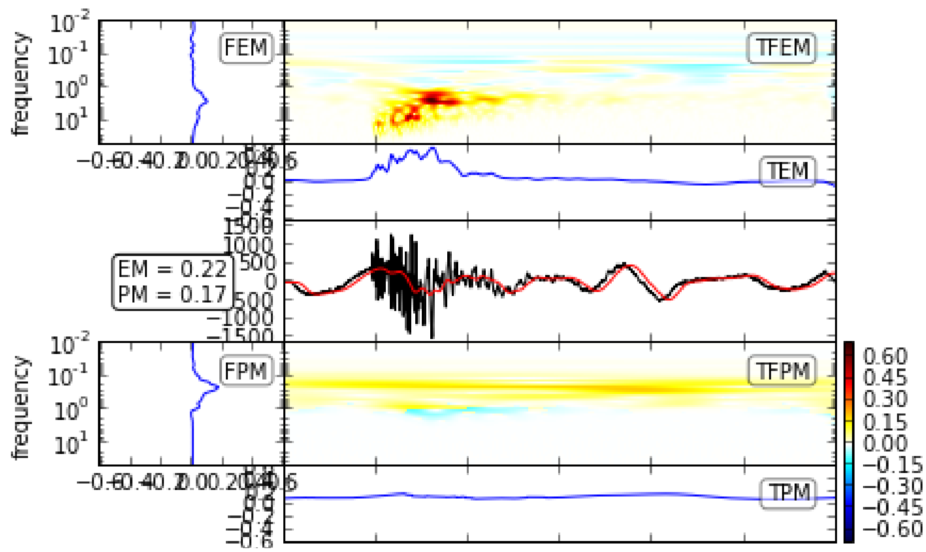
In [23]:
```
from obspy.signal.tf_misfit import plotTfMisfits,em,pm
#pouzijeme povodny a filtrovany zaznam (bez dlhovlnnej zlozky)
#vykreslenie TF Misfitov:
plotTfMisfits(tr3.data,tr3_filt.data,fmin=0.01,fmax=50.)
```

```
#pouzili sme filter, ktory nemeni fazu -> nulove pm
#hodnoty:
print em(tr3.data,tr3_filt.data,fmin=0.01,fmax=50.)
print pm(tr3.data,tr3_filt.data,fmin=0.01,fmax=50.)
#podobne sa ziskaju funkcie fem,tem,fpm, tpm (np.array)
```




```
4.49337036855
0.0167142748561
```

In [24]:
```
#odfiltrujme vysoke frekvencie, dovolime fazovy posun
tr3_filt2=tr3.copy()
#vnesieme fazovy posun na dlhovlnnejsej zlozke
tr3_filt2.filter('lowpass',freq=1.,zerophase=False)
#vykreslenie TF misfitov
plotTfMisfits(tr3.data,tr3_filt2.data,fmin=50.,fmax=0.01)
#hodnoty:
#odfiltrovanie vysokofrekv. casti - rozdiel v obalkach:
print em(tr3.data,tr3_filt2.data,fmin=50.,fmax=0.01)
#fazovy posun:
print pm(tr3.data,tr3_filt2.data,fmin=50.,fmax=0.01)
```




```
0.215362001055
0.173672743528
```

obspy.signal.invsim.estimateMagnitude

In [26]:

```
from obspy.core import read, UTCDateTime
from obspy.core.util.geodetics import gps2DistAzimuth
from obspy.xseed import Parser
from obspy.signal.invsim import estimateMagnitude
#LOKALNE MAGNITUDO!

#data
st5 = read("http://localhost/data/Advanced%20ObsPy%20Exercise/LK

#nuly a poly
parser = Parser("http://localhost/data/Advanced%20ObsPy%20Exerci
paz = parser.getPAZ("CH.LKBD..EHZ")

# maximalna amplituda na N zlozke
trn = st5.select(component="N")[0]
amplmax_n = max(trn.data)
amplmin_n = min(trn.data)
# maximalna amplituda na E zlozke
tre = st5.select(component="E")[0]
amplmax_e = max(tre.data)
amplmin_e = min(tre.data)
# maximalna amplituda na Z zlozke
trz = st5.select(component="Z")[0]
amplmax_z = max(trz.data)
amplmin_z = min(trz.data)

#vyrobime pole zo vsetkych zloziek
ampl = [amplmax_n-amplmin_n, amplmax_e-amplmin_e,
        amplmax_z-amplmin_z]
samp=st5[0].stats.sampling_rate

#poloha stanice a eventu
sta_lat = 46.38703
sta_lon = 7.62714
event_lat = 46.218
event_lon = 7.706
event_depth=5.2

#vypocet epicentralnej vzdialenosti
epi_dist, az, baz= gps2DistAzimuth(event_lat, event_lon,
                                   sta_lat, sta_lon)
epi_dist = epi_dist / 1000 #v km
```

```
epi_dist= sqrt(epi_dist**2+event_depth**2) #pridat hlbku

# vstup: poles&zeroes, peak-to-peak amplitude
# ... hypocentral distance, timespan of peak-to-peak ampl.
print estimateMagnitude([paz,paz,paz], ampl,
                        h_dist=epi_dist,timespan=0.5)
#ML z katalogu bolo 2.3
```

```
2.28777972093
```

FK Analysis

Array response function