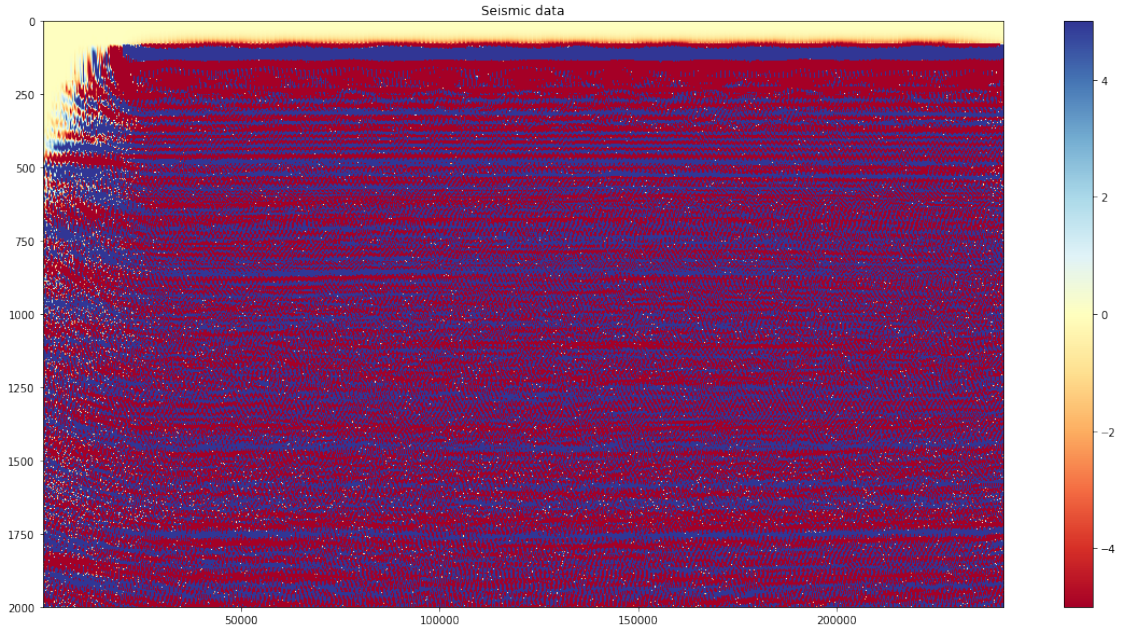
FWI_Final_Stable_Version

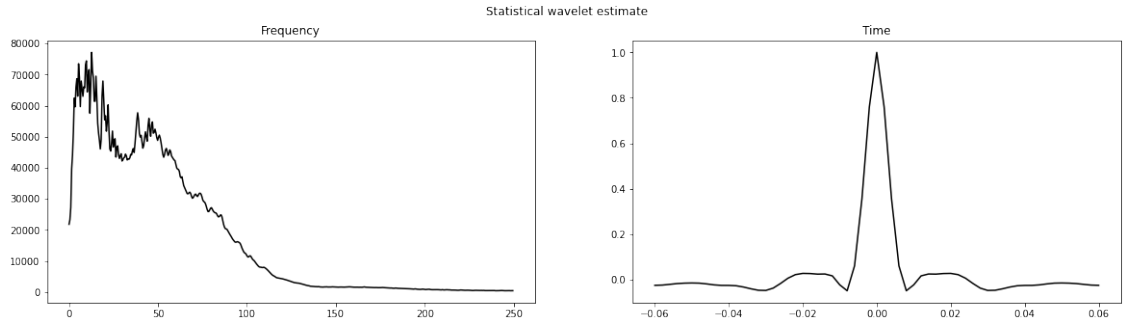
August 11, 2021

[1]:
```
%load_ext autoreload
%autoreload 2
%matplotlib inline

import warnings
warnings.filterwarnings('ignore')

import numpy as np
import matplotlib.pyplot as plt
import scipy as sp
import segyio
import pylops
```

[2]:
```
segyfile = '/home/tibo/sleipner/Sleipner_Seismic/p11ful/10p11ful.segy'

f = segyio.open(segyfile, iline=segyio.tracefield.TraceField.
 ↪SourceEnergyDirectionExponent,
                xline=segyio.tracefield.TraceField.CDP)

il, xl, t = f.ilines, f.xlines, f.samples
dt = t[1] - t[0]

d = segyio.cube(f)
nil, nxl, nt = d.shape

plt.figure(figsize=(20, 10))
plt.imshow(d[nil//2].T, cmap='RdYlBu', vmin=-5, vmax=5,
           extent=(xl[0], xl[-1], t[-1], t[0]))
plt.title('Seismic data')
plt.colorbar()
plt.axis('tight');
plt.savefig('/home/tibo/sleipner/Sleipner_Seismic/seismic_data.png')
```



[3]:
```
nt_wav = 31 # lenght of wavelet in samples
nfft = 2**10 # lenght of fft

# time axis for wavelet
t_wav = np.arange(nt_wav) * (dt/1000)
t_wav = np.concatenate((np.flipud(-t_wav[1:]), t_wav), axis=0)

# estimate wavelet spectrum
wav_est_fft = np.mean(np.abs(np.fft.fft(d[..., :500], nfft, axis=-1)), axis=(0,␣
↪1))
fwest = np.fft.fftfreq(nfft, d=dt/1000)

# create wavelet in time
wav_est = np.real(np.fft.ifft(wav_est_fft)[:nt_wav])
wav_est = np.concatenate((np.flipud(wav_est[1:]), wav_est), axis=0)
wav_est = wav_est / wav_est.max()
wcenter = np.argmax(np.abs(wav_est))

# display wavelet
fig, axs = plt.subplots(1, 2, figsize=(20, 5))
fig.suptitle('Statistical wavelet estimate')
axs[0].plot(fwest[:nfft//2], wav_est_fft[:nfft//2], 'k')
axs[0].set_title('Frequency')
axs[1].plot(t_wav, wav_est, 'k')
axs[1].set_title('Time');
```

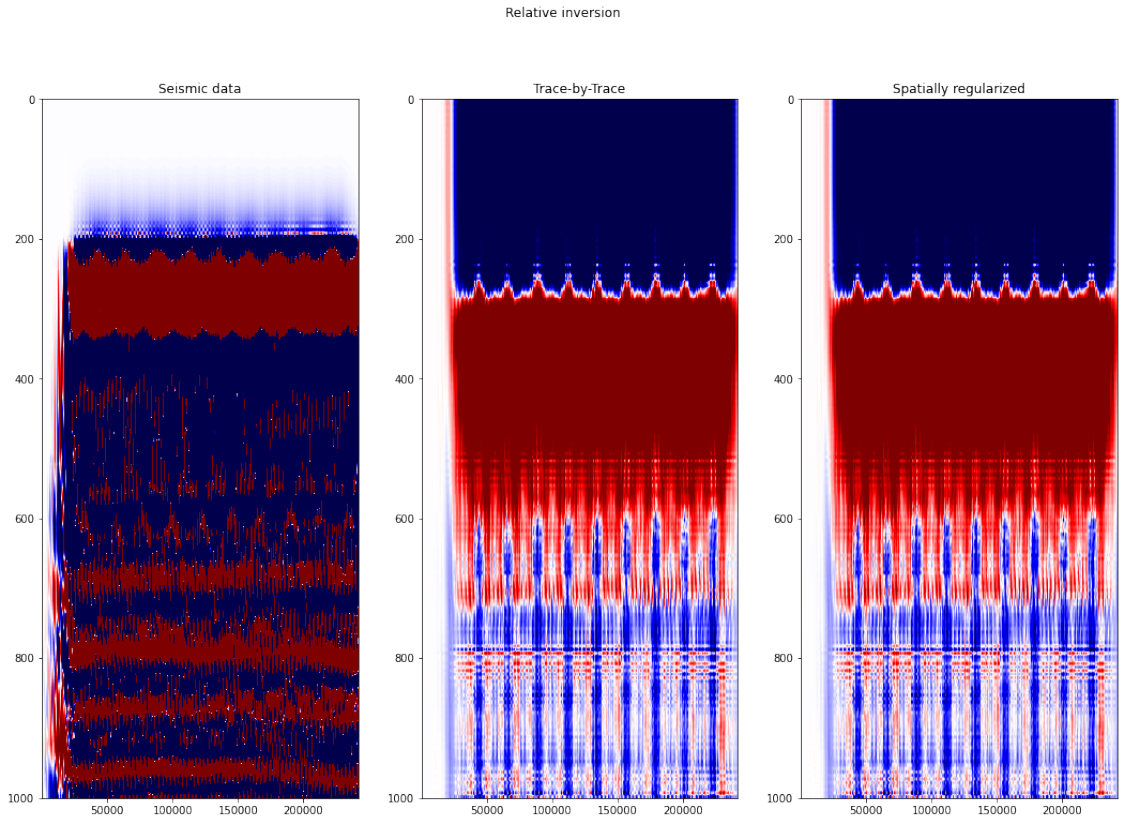


[4]:
```
# swap time axis to first dimension
d_small = d[..., :200]
d_small = np.swapaxes(d_small, -1, 0)

m_relative, r_relative = \
    pylops.avo.poststack.PoststackInversion(d_small, wav_est, m0=np.
 ↪zeros_like(d_small), explicit=True, epsI=1e-4,
                                            simultaneous=False)
m_relative_reg, r_relative_reg = \
    pylops.avo.poststack.PoststackInversion(d_small, wav_est, m0=m_relative,␣
 ↪epsI=1e-4, epsR=1e0,
                                            **dict(iter_lim=10, show=2))
# swap time axis back to last dimension
d_small = np.swapaxes(d_small, 0, -1)
m_relative = np.swapaxes(m_relative, 0, -1)
m_relative_reg = np.swapaxes(m_relative_reg, 0, -1)
r_relative = np.swapaxes(r_relative, 0, -1)
r_relative_reg = np.swapaxes(r_relative_reg, 0, -1)
```

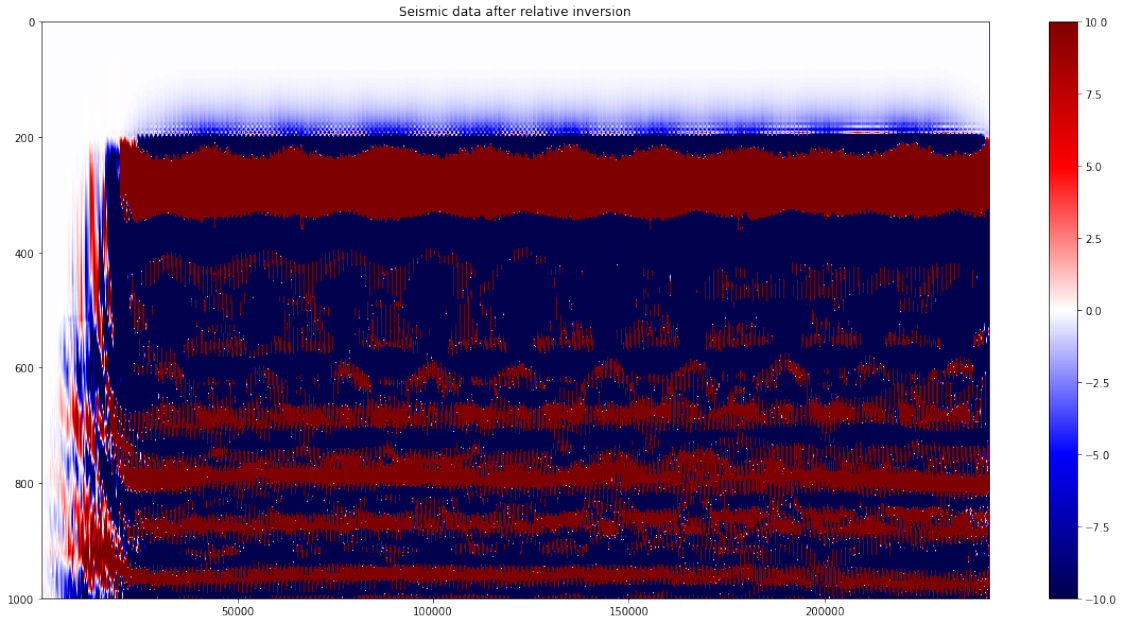
```
LSQR            Least-squares solution of  Ax = b
The matrix A has 96854400 rows and 48427200 columns
damp = 0.00000000000000e+00   calc_var =        0
atol = 1.00e-08                 conlim = 1.00e+08
btol = 1.00e-08               iter_lim =       10

   Itn      x[0]       r1norm     r2norm   Compatible    LS      Norm A   Cond A
     0  0.00000e+00   4.004e+06  4.004e+06    1.0e+00  8.7e-07
     1 -1.76460e-04   1.400e+06  1.400e+06    3.5e-01  5.5e-01   3.7e+00
1.0e+00
     2 -2.93130e-04   1.001e+06  1.001e+06    2.5e-01  2.6e-01   4.8e+00
2.1e+00
     3 -3.48533e-04   8.515e+05  8.515e+05    2.1e-01  2.2e-01   5.5e+00
3.5e+00
     4 -3.53159e-04   7.216e+05  7.216e+05    1.8e-01  1.8e-01   6.3e+00
```

```
5.4e+00
     5 -2.89512e-04   6.588e+05  6.588e+05    1.6e-01  1.1e-01   7.1e+00
7.2e+00
     6 -1.82346e-04   6.289e+05  6.289e+05    1.6e-01  8.4e-02   7.8e+00
8.8e+00
     7  3.28588e-05   5.942e+05  5.942e+05    1.5e-01  8.6e-02   8.3e+00
1.1e+01
     8  2.50253e-04   5.677e+05  5.677e+05    1.4e-01  5.9e-02   8.9e+00
1.4e+01
     9  4.03195e-04   5.553e+05  5.553e+05    1.4e-01  4.7e-02   9.4e+00
1.6e+01
    10  6.25561e-04   5.432e+05  5.432e+05    1.4e-01  4.8e-02   9.8e+00
1.8e+01

LSQR finished
The iteration limit has been reached

istop =       7   r1norm = 5.4e+05   anorm = 9.8e+00   arnorm = 2.5e+05
itn   =      10   r2norm = 5.4e+05   acond = 1.8e+01   xnorm  = 1.5e+06
```

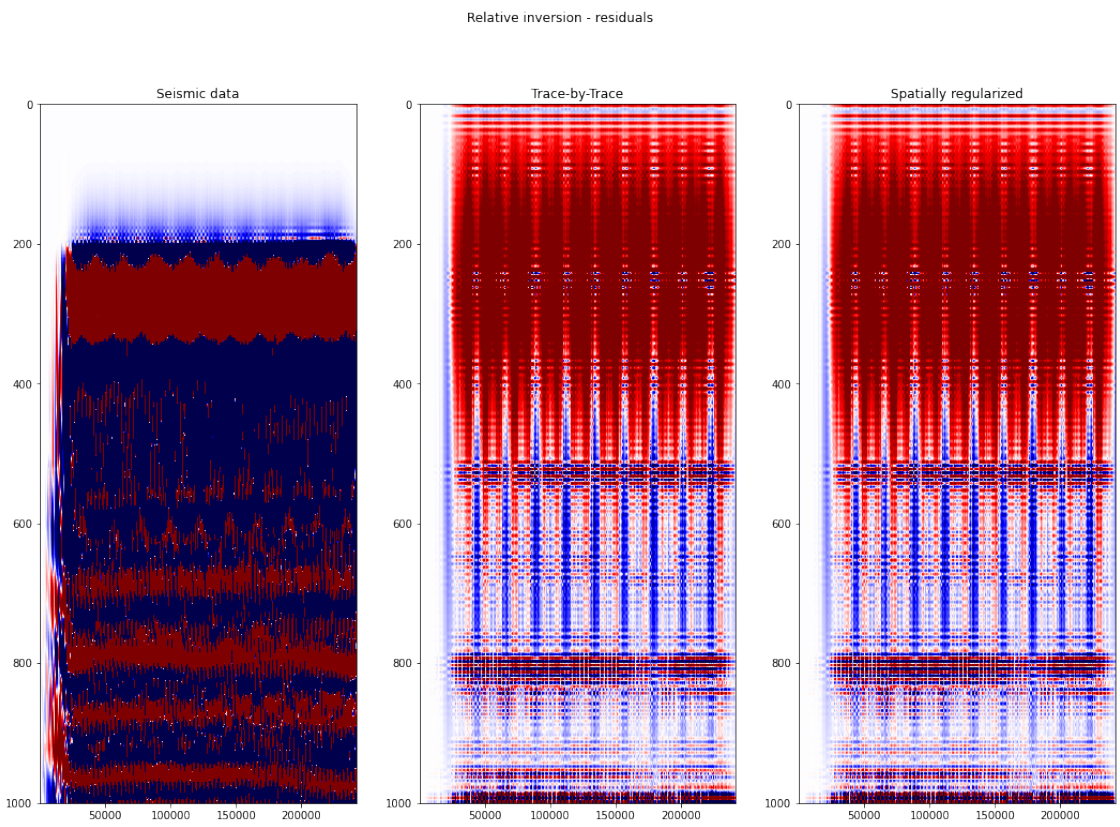
[5]:
```
fig, axs = plt.subplots(1, 3, figsize=(18, 12))
fig.suptitle('Relative inversion')
axs[0].imshow(d_small[nil//2].T, cmap='seismic', vmin=-10, vmax=10,
              extent=(xl[0], xl[-1], t[500], t[0]))
axs[0].set_title('Seismic data')
axs[0].axis('tight')
axs[1].imshow(m_relative[nil//2].T, cmap='seismic', vmin=-0.2*m_relative.max(),␣
 ↪vmax=0.2*m_relative.max(),
              extent=(xl[0], xl[-1], t[500], t[0]))
axs[1].set_title('Trace-by-Trace')
axs[1].axis('tight')
axs[2].imshow(m_relative_reg[nil//2].T, cmap='seismic', vmin=-0.2*m_relative.
 ↪max(), vmax=0.2*m_relative.max(),
              extent=(xl[0], xl[-1], t[500], t[0]))
axs[2].set_title('Spatially regularized')
axs[2].axis('tight');
```



[6]:
```
plt.figure(figsize=(20, 10))
plt.imshow(d_small[nil//2].T, cmap='seismic', vmin=-10, vmax=10,
              extent=(xl[0], xl[-1], t[500], t[0]))
plt.title('Seismic data after relative inversion')
plt.colorbar()
plt.axis('tight');
plt.savefig('/home/tibo/sleipner/Sleipner_Seismic/seismic_data_inversion.png')
```



[7]:
```
fig, axs = plt.subplots(1, 3, figsize=(18, 12))
fig.suptitle('Relative inversion - residuals')
axs[0].imshow(d_small[nil//2].T, cmap='seismic', vmin=-10, vmax=10,
              extent=(xl[0], xl[-1], t[500], t[0]))
axs[0].set_title('Seismic data')
axs[0].axis('tight')
axs[1].imshow(r_relative[nil//2].T, cmap='seismic', vmin=-10, vmax=10,
              extent=(xl[0], xl[-1], t[500], t[0]))
axs[1].set_title('Trace-by-Trace')
axs[1].axis('tight')
axs[2].imshow(r_relative_reg[nil//2].T, cmap='seismic', vmin=-10, vmax=10,
              extent=(xl[0], xl[-1], t[500], t[0]))
axs[2].set_title('Spatially regularized')
axs[2].axis('tight');
```


Relative inversion - residuals
Seismic data
Trace-by-Trace
Spatially regularized
0
200
400
600
800
1000
50000
100000
150000
200000