Group 8 Project Source Code

March 18, 2023

1 Replication: Martinos Dataset TF-MxNE

1.1 Preliminaries

We start by importing all the necessary packages and specifying where our data is. In addition to the raw data, the Martinos dataset comes with some additional files that have already performed some calculations on the raw data for us (e.g. covariance, averaged measurements, the dipole-to-sensor forward solution). This helps simplify the code we need to run since we need to input these into later commands. For the replication, we will focus on the visual stimuli being applied on the left side.

[1]:
```
import os
import numpy as np
import matplotlib.pyplot as plt

import mne
from mne.datasets import sample
from mne.minimum_norm import make_inverse_operator, apply_inverse
from mne.inverse_sparse import tf_mixed_norm, make_stc_from_dipoles
from mne.viz import (plot_sparse_source_estimates,
                     plot_dipole_locations, plot_dipole_amplitudes)

import nibabel
import PyQt5

print(__doc__)

# where all the files are
data_path = sample.data_path()
subjects_dir = data_path / 'subjects'
meg_path = data_path / 'MEG' / 'sample'
fwd_fname = meg_path / 'sample_audvis-meg-eeg-oct-6-fwd.fif'
ave_fname = meg_path / 'sample_audvis-no-filter-ave.fif'
cov_fname = meg_path / 'sample_audvis-shrunk-cov.fif'
raw_fname = meg_path / 'sample_audvis_raw.fif'

# get info on sensors
info = mne.io.read_info(raw_fname)
```

```python
# noise covariance matrix
cov = mne.read_cov(cov_fname)

# specifying what condition we look for
condition = 'Left visual'
evoked = mne.read_evokeds(ave_fname, condition=condition, baseline=(None, 0))
evoked = mne.pick_channels_evoked(evoked)

# cropping for the time window around the stimulus
evoked.crop(tmin=-0.1, tmax=0.4)

# forward solution
forward = mne.read_forward_solution(fwd_fname)
```

```
Automatically created module for IPython interactive environment
    Read a total of 3 projection items:
        PCA-v1 (1 x 102)  idle
        PCA-v2 (1 x 102)  idle
        PCA-v3 (1 x 102)  idle
    365 x 365 full covariance (kind = 1) found.
    Read a total of 4 projection items:
        PCA-v1 (1 x 102) active
        PCA-v2 (1 x 102) active
        PCA-v3 (1 x 102) active
        Average EEG reference (1 x 59) active
Reading C:\Users\jeffr\mne_data\MNE-sample-data\MEG\sample\sample_audvis-no-
filter-ave.fif …
    Read a total of 4 projection items:
        PCA-v1 (1 x 102) active
        PCA-v2 (1 x 102) active
        PCA-v3 (1 x 102) active
        Average EEG reference (1 x 60) active
    Found the data of interest:
        t =    -199.80 …     499.49 ms (Left visual)
        0 CTF compensation matrices available
        nave = 64 - aspect type = 100
Projections have already been applied. Setting proj attribute to True.
Applying baseline correction (mode: mean)
Reading forward solution from C:\Users\jeffr\mne_data\MNE-sample-
data\MEG\sample\sample_audvis-meg-eeg-oct-6-fwd.fif…
    Reading a source space…
    Computing patch statistics…
    Patch information added…
    Distance information added…
    [done]
    Reading a source space…
    Computing patch statistics…
```

```
Patch information added…
Distance information added…
[done]
2 source spaces read
Desired named matrix (kind = 3523) not available
Read MEG forward solution (7498 sources, 306 channels, free orientations)
Desired named matrix (kind = 3523) not available
Read EEG forward solution (7498 sources, 60 channels, free orientations)
Forward solutions combined: MEG, EEG
Source spaces transformed to the forward solution coordinate frame
```

1.2 Take 1: Tutorial Code

Unfortunately, the code given in the MNE-Python software paper does not exactly match the procedure performed in the original TF-MxNE algorithm paper. We first replicate what the tutorial does before attempting to adjust the parameters to do what the paper does. It should be noted that the TF-MxNE paper seems not to have used the MNE-Python software, but they did not publish their code, making it impossible to determine how they originally coded their algorithm. We will be taking our best shot at imitating what they did without directly seeing what they did.

1.2.1 Dipole Line Plots

In order to initialize the weights to be used in TF-MxNE, the trick used by Gramfort *et al.* is to run dynamical statistical parametric mapping (dSPM) and use its weights as a starting point before inducing sparsity. The code uses a regularization parameter of $\lambda_2 = \frac{1}{9}$, although this ultimately should not affect the TF-MxNE results significantly since this is just the initialized weights starting point. Two other parameters used to run dSPM are the loose orientation parameter ($\rho = 0.2$) and depth bias compensation ($\gamma = 0.9$). These will be reused by TF-MxNE later on.

[2]:
```python
# loose orientation and depth bias compensation parameters
loose, depth = 0.2, 0.9

# dSPM-initialized weights
inverse_operator = make_inverse_operator(evoked.info, forward, cov,
                                         loose=loose, depth=depth)
stc_dspm = apply_inverse(evoked, inverse_operator, lambda2=1. / 9.,
                         method='dSPM')
```

```
Converting forward solution to surface orientation
    Average patch normals will be employed in the rotation to the local surface
coordinates…
    Converting to surface-based source orientations…
    [done]
info["bads"] and noise_cov["bads"] do not match, excluding bad channels from
both
Computing inverse operator with 364 channels.
    364 out of 366 channels remain after picking
Selected 364 channels
Creating the depth weighting matrix…
```

```
    203 planar channels
    limit = 7262/7498 = 10.020865
    scale = 2.58122e-08 exp = 0.9
Applying loose dipole orientations to surface source spaces: 0.2
Whitening the forward solution.
    Created an SSP operator (subspace dimension = 4)
Computing rank from covariance with rank=None
    Using tolerance 3.5e-13 (2.2e-16 eps * 305 dim * 5.2  max singular value)
    Estimated rank (mag + grad): 302
    MEG: rank 302 computed from 305 data channels with 3 projectors
    Using tolerance 1.1e-13 (2.2e-16 eps * 59 dim * 8.7  max singular value)
    Estimated rank (eeg): 58
    EEG: rank 58 computed from 59 data channels with 1 projector
    Setting small MEG eigenvalues to zero (without PCA)
    Setting small EEG eigenvalues to zero (without PCA)
Creating the source covariance matrix
Adjusting source covariance matrix.
Computing SVD of whitened and weighted lead field matrix.
    largest singular value = 5.96729
    scaling factor to adjust the trace = 9.38524e+18 (nchan = 364 nzero = 4)
Preparing the inverse operator for use…
    Scaled noise and source covariance from nave = 1 to nave = 64
    Created the regularized inverter
    Created an SSP operator (subspace dimension = 4)
    Created the whitener using a noise covariance matrix with rank 360 (4 small
eigenvalues omitted)
    Computing noise-normalization factors (dSPM)…
[done]
Applying inverse operator to "Left visual"…
    Picked 364 channels from the data
    Computing inverse…
    Eigenleads need to be weighted …
    Computing residual…
    Explained  60.0% variance
    Combining the current components…
    dSPM…
[done]
```

The TF-MxNE algorithm on MNE-Python sets its regularization parameters in a different way than most algorithms do. It first determines what λ regularization parameter would induce all of the sources to be zero, and that is determined as the "maximum" λ parameter. The α parameter is the percentage of λ_{max} that ends up being used. In other words, $\alpha = 0$ means that there will be no regularization, and $\alpha = 100$ will yield all zero sources. Next, there is an L_1 ratio parameter to be set to determine the spatial and temporal regularization. The spatial regularization parameter is $\lambda_{max} \cdot \alpha \cdot (1 - L_1 \text{ ratio})$, and the temporal regularization parameter is $\lambda_{max} \cdot \alpha \cdot (L_1 \text{ ratio})$.

In the tutorial, the α regularization parameter is set to 40, and the L_1 ratio parameter is set to 0.03.

Additional parameters are needed to specify how the Gabor dictionary is calculated. The Gabor dictionary runs a short-time Fourier transform using windows of 16 samples and a 4-sample time shift for the windows. There are also other "minor" settings included, like the maximum iterations (set to 200), convergence tolerance (10^{-6}), and only dSPM weights greater than 8 are used.

Lastly, after the TF-MxNE algorithm is run, we crop the calculated dipoles to see the source amplitudes between 50 milliseconds before the stimulus and 300 milliseconds after.

[3]:
```python
# alpha and L_1 ratio parameters for TF-MxNE
alpha = 40.
l1_ratio = 0.03

# TF-MxNE
dipoles, residual = tf_mixed_norm(
    evoked, forward, cov, alpha=alpha, l1_ratio=l1_ratio, loose=loose,
    depth=depth, maxit=200, tol=1e-6, weights=stc_dspm, weights_min=8.,
    debias=True, wsize=16, tstep=4, window=0.05, return_as_dipoles=True,
    return_residual=True)

# crop time window to see dipole amplitudes around onset of stimulus
for dip in dipoles:
    dip.crop(tmin=-0.05, tmax=0.3)
evoked.crop(tmin=-0.05, tmax=0.3)
residual.crop(tmin=-0.05, tmax=0.3)
```

```
Converting forward solution to surface orientation
    Average patch normals will be employed in the rotation to the local surface
coordinates…
    Converting to surface-based source orientations…
    [done]
info["bads"] and noise_cov["bads"] do not match, excluding bad channels from
both
Computing inverse operator with 364 channels.
    364 out of 366 channels remain after picking
Selected 364 channels
Creating the depth weighting matrix…
Applying loose dipole orientations to surface source spaces: 0.2
Whitening the forward solution.
    Created an SSP operator (subspace dimension = 4)
Computing rank from covariance with rank=None
    Using tolerance 3.5e-13 (2.2e-16 eps * 305 dim * 5.2  max singular value)
    Estimated rank (mag + grad): 302
    MEG: rank 302 computed from 305 data channels with 3 projectors
    Using tolerance 1.1e-13 (2.2e-16 eps * 59 dim * 8.7  max singular value)
    Estimated rank (eeg): 58
    EEG: rank 58 computed from 59 data channels with 1 projector
    Setting small MEG eigenvalues to zero (without PCA)
    Setting small EEG eigenvalues to zero (without PCA)
Creating the source covariance matrix
```

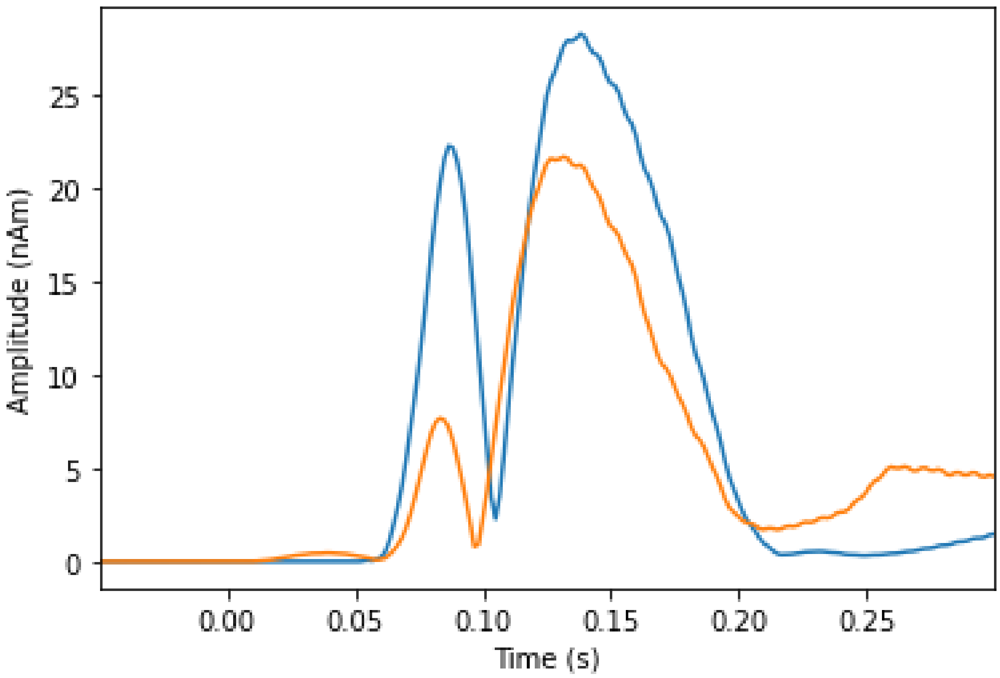
```
Adjusting source covariance matrix.
Reducing source space to 985 sources
Whitening data matrix.
Using block coordinate descent with active set approach

    Iteration 10 :: n_active 3
    dgap 3.49e+00 :: p_obj 4411.845726 :: d_obj 4408.353441

    Iteration 20 :: n_active 3
    dgap 5.67e-01 :: p_obj 4410.859492 :: d_obj 4410.292946

dgap 1.51e-01 :: p_obj 4410.670058 :: d_obj 4410.519426 :: n_active 2

    Iteration 10 :: n_active 2
    dgap 1.61e-03 :: p_obj 4410.669663 :: d_obj 4410.668049

dgap 1.61e-03 :: p_obj 4410.669663 :: d_obj 4410.668049 :: n_active 2

Convergence reached!

Debiasing converged after 190 iterations max(|D - D0| = 5.546629e-07 <
1.000000e-06)
[done]
```

[3]:
```
<Evoked | 'Left visual' (average, N=64), -0.049949 - 0.29969 sec, baseline
-0.199795 - 0 sec (baseline period was cropped after baseline correction), 364
ch, ~3.8 MB>
```

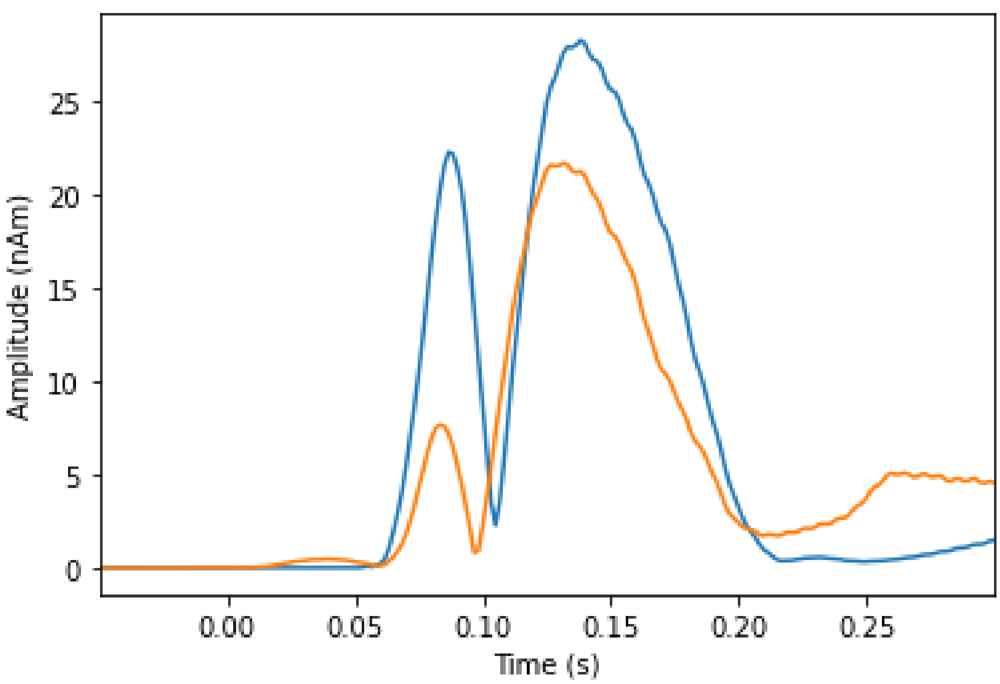
We can now visualize the source amplitudes.

[4]:
```
plot_dipole_amplitudes(dipoles)
```

[4]:



Amplitude (nAm)
0
5
10
15
20
25
0.00
0.05
0.10
0.15
0.20
0.25
Time (s)

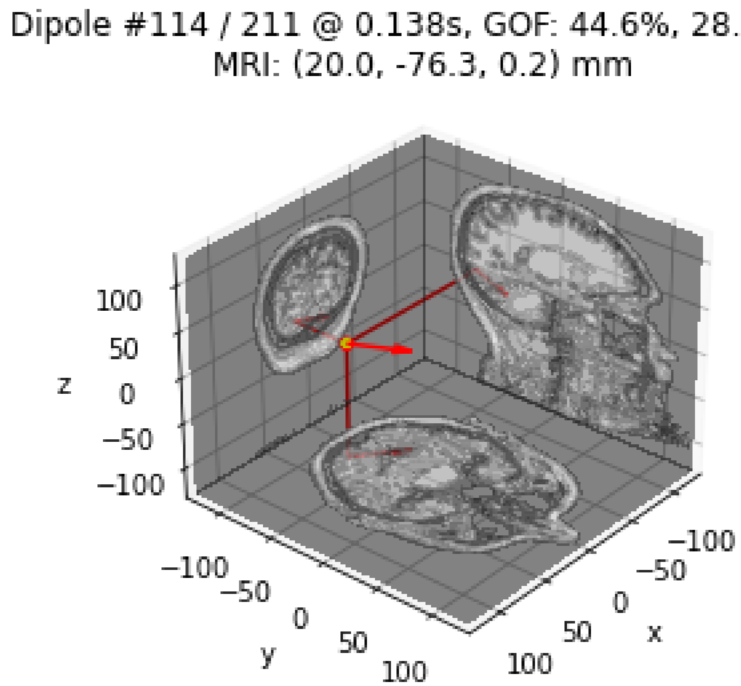


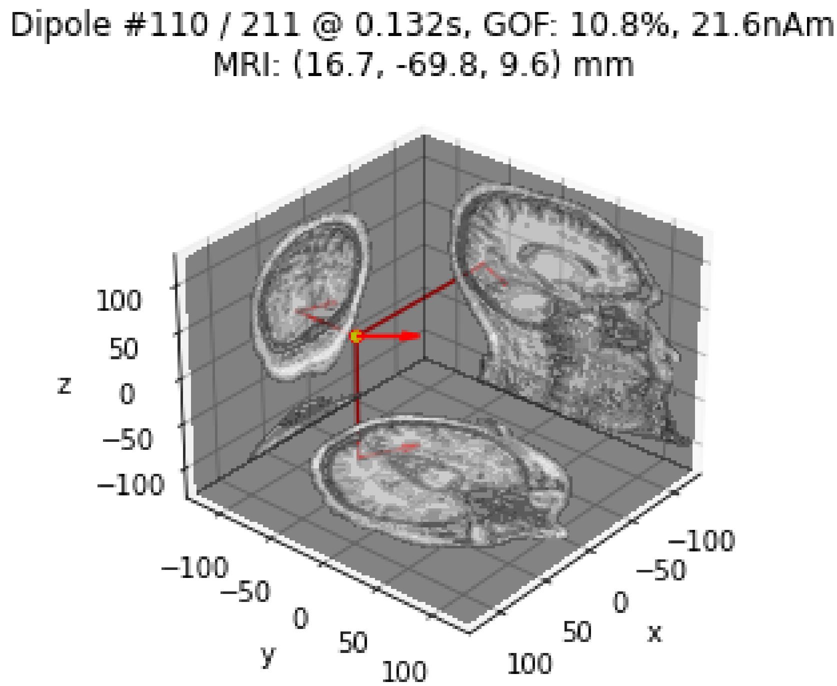
Amplitude (nAm)
0
5
10
15
20
25
0.00
0.05
0.10
0.15
0.20
0.25
Time (s)

1.2.2 Dipole Source Locations

The source amplitudes alone aren't very useful if we don't know where they are in the brain. The following code allows us to see where thoes dipoles are.

[5]:
```
for dip in dipoles:
    plot_dipole_locations(dip, forward['mri_head_t'], 'sample',
                          subjects_dir=subjects_dir, mode='orthoview',
                          idx='amplitude')
```





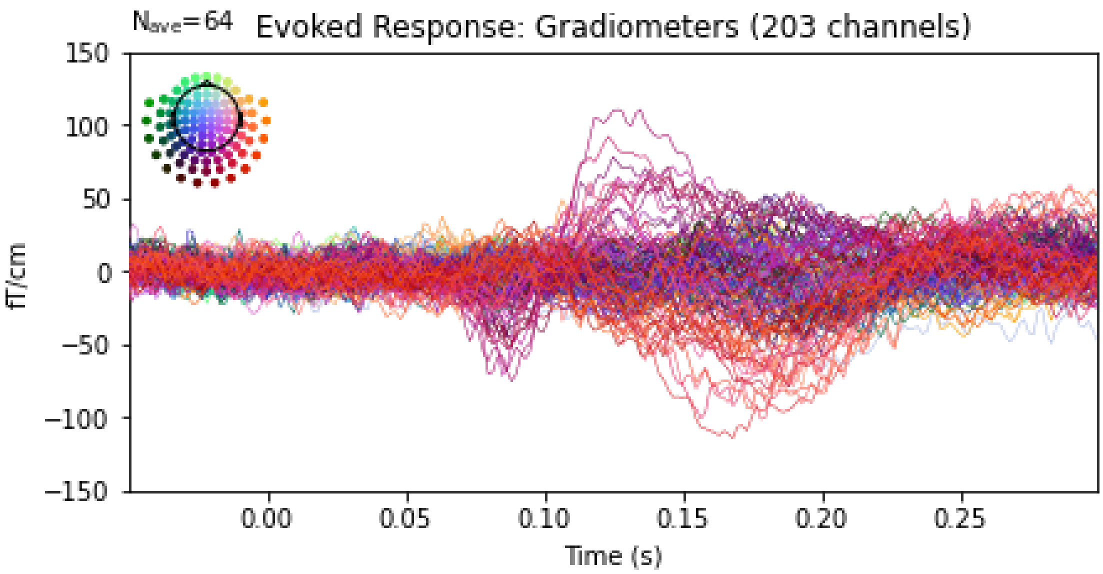
1.2.3 MEG Sensor Projection

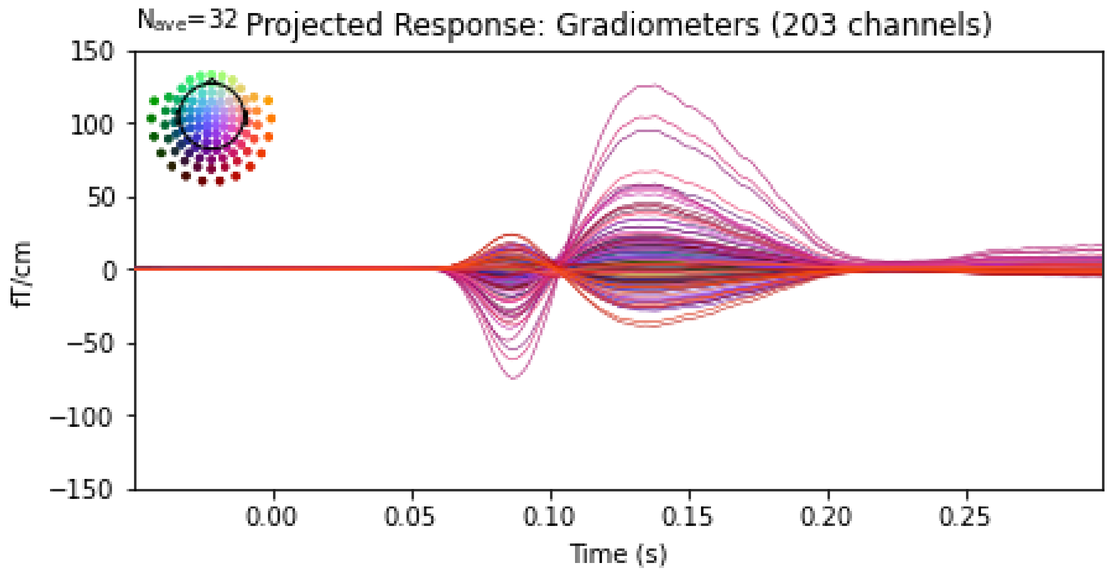
To see how much of the MEG data explained by the dipoles determined by TF-MxNE, we can first plot the line plots of the MEG data for all channels, followed by the projection of the dipoles onto the sensors using the forward solution, and then the residual plots. To get the projection, one trick is just to subtract the evoked line plots by the residuals.

[6]:
```python
# filtering for MEG channels
evoked.pick_types(meg='grad', exclude='bads')
residual.pick_types(meg='grad', exclude='bads')

# getting the dipole projection to sensor space
explained = mne.combine_evoked([evoked, residual], weights=[1, -1])

# limit amplitude
ylim = dict(grad=[-150, 150])

# all evoked data
evoked.plot(titles=dict(grad='Evoked Response: Gradiometers'), ylim=ylim,
            proj=True, time_unit='s')

# all explained data
explained.plot(titles=dict(grad='Projected Response: Gradiometers'), ylim=ylim,
            proj=True, time_unit='s')
```

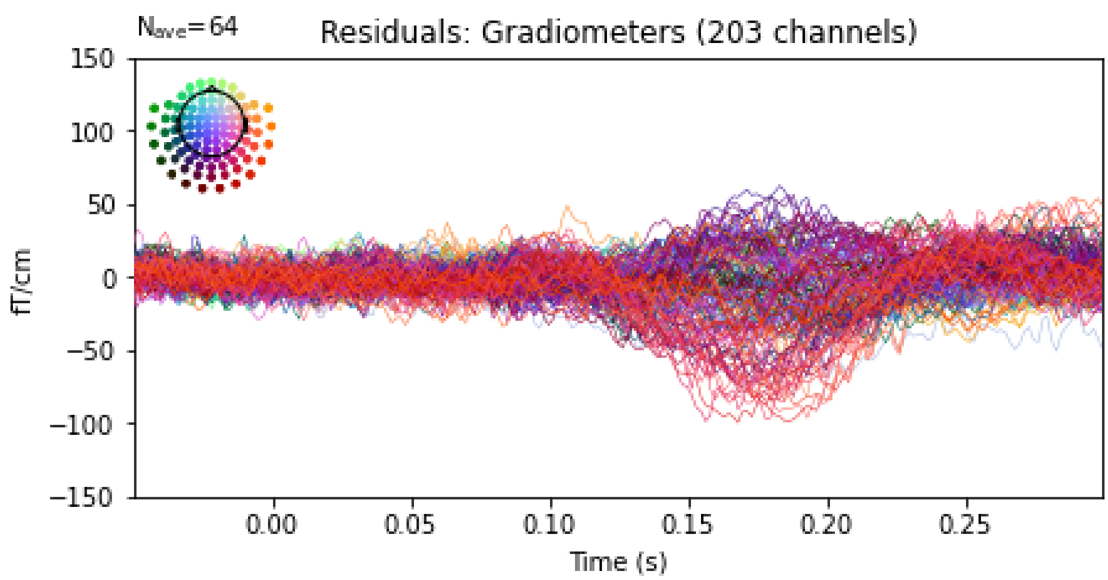
```python
# all residuals
residual.plot(titles=dict(grad='Residuals: Gradiometers'), ylim=ylim,
              proj=True, time_unit='s')
```

```
Removing projector <Projection | PCA-v1, active : True, n_channels : 102>
Removing projector <Projection | PCA-v2, active : True, n_channels : 102>
Removing projector <Projection | PCA-v3, active : True, n_channels : 102>
Removing projector <Projection | Average EEG reference, active : True,
n_channels : 60>
Removing projector <Projection | PCA-v1, active : True, n_channels : 102>
Removing projector <Projection | PCA-v2, active : True, n_channels : 102>
Removing projector <Projection | PCA-v3, active : True, n_channels : 102>
Removing projector <Projection | Average EEG reference, active : True,
n_channels : 60>
```



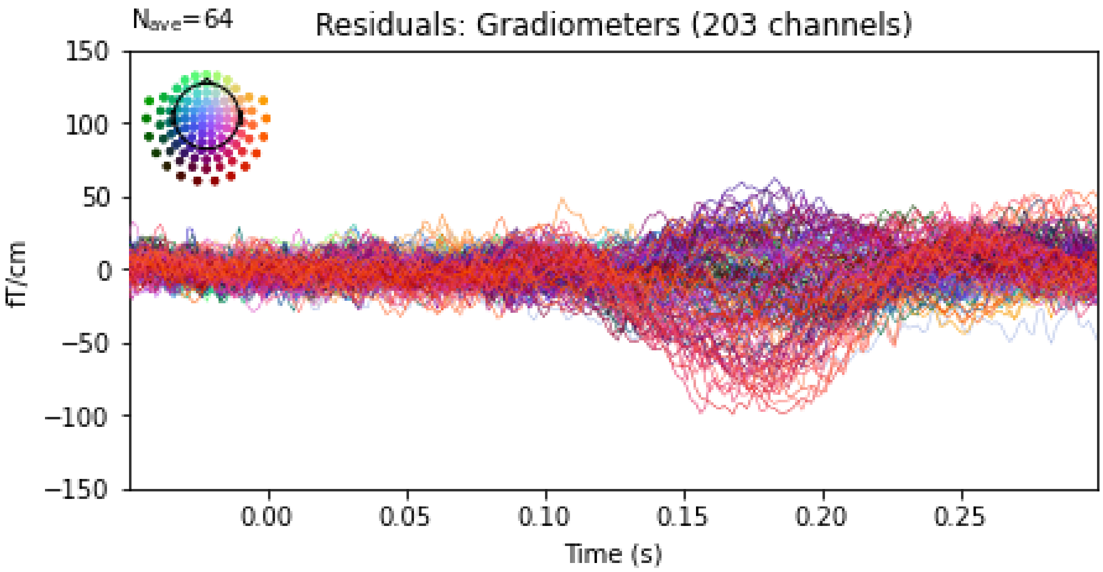


N_{ave}=32 Projected Response: Gradiometers (203 channels)
150
100
50
0
−50
−100
−150
fT/cm
0.00
0.05
0.10
0.15
0.20
0.25
Time (s)



N_{ave}=64 Residuals: Gradiometers (203 channels)
150
100
50
0
−50
−100
−150
fT/cm
0.00
0.05
0.10
0.15
0.20
0.25
Time (s)

[6]:



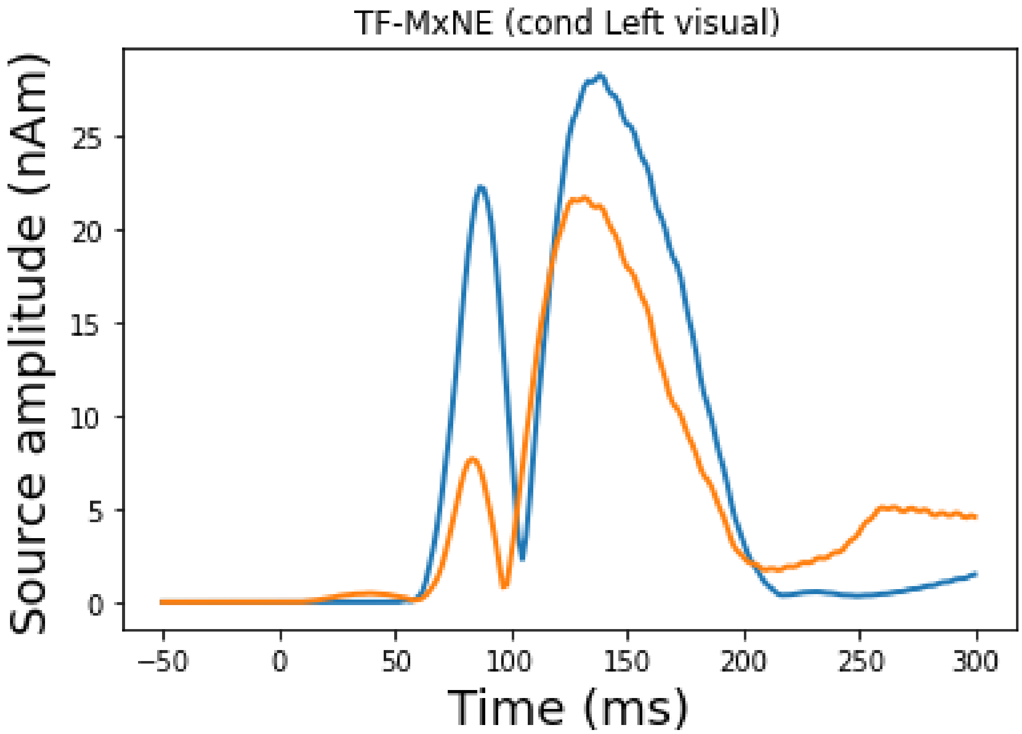
1.2.4 Source Localization 3D Visualization

We can now get a fancy 3D visualization of both the dipoles in the brain, as well as how the source amplitudes change over time. These unfortunately open in a new window, so these cannot be seen within the Jupyter Notebook environment.

[7]:
```
# make SourceEstimate object from dipole list
stc = make_stc_from_dipoles(dipoles, forward['src'])

# get source amplitudes again
plot_sparse_source_estimates(forward['src'], stc, bgcolor=(1, 1, 1),
                             opacity=0.1, fig_name="TF-MxNE (cond %s)"
                             % condition, modes=['sphere'], scale_factors=[1.])

# fancy 3D plot
time_label = 'TF-MxNE time=%0.2f ms'
clim = dict(kind='value', lims=[10e-9, 15e-9, 20e-9])
brain = stc.plot('sample', 'inflated', 'rh', views='medial',
                 clim=clim, time_label=time_label, smoothing_steps=5,
                 subjects_dir=subjects_dir, initial_time=150, time_unit='ms')
brain.add_label("V1", color="red", scalar_thresh=0.5, alpha=0.6, borders=False)
brain.add_label("V2", color="yellow", scalar_thresh=0.5, alpha=0.6,␣
 ↪borders=False)
```

```
Converting dipoles into a SourceEstimate.
[done]
Using pyvistaqt 3d backend.
```

```
Total number of active sources: 2
```



1.3 Take 2: Original Paper Imitation

1.3.1 Preliminaries

We will be running essentially the same pipeline as the tutorial, except we want to vary the parameters. We need to reset a few of our variables.

[8]:
```python
# get info on sensors
info = mne.io.read_info(raw_fname)

# noise covariance matrix
cov = mne.read_cov(cov_fname)

# specifying what condition we look for
condition = 'Left visual'
evoked = mne.read_evokeds(ave_fname, condition=condition, baseline=(None, 0))
evoked = mne.pick_channels_evoked(evoked)

# cropping for the time window around the stimulus
evoked.crop(tmin=-0.1, tmax=0.4)

# forward solution
```

```python
forward = mne.read_forward_solution(fwd_fname)
```

```
    Read a total of 3 projection items:
        PCA-v1 (1 x 102)  idle
        PCA-v2 (1 x 102)  idle
        PCA-v3 (1 x 102)  idle
    365 x 365 full covariance (kind = 1) found.
    Read a total of 4 projection items:
        PCA-v1 (1 x 102) active
        PCA-v2 (1 x 102) active
        PCA-v3 (1 x 102) active
        Average EEG reference (1 x 59) active
Reading C:\Users\jeffr\mne_data\MNE-sample-data\MEG\sample\sample_audvis-no-
filter-ave.fif …
    Read a total of 4 projection items:
        PCA-v1 (1 x 102) active
        PCA-v2 (1 x 102) active
        PCA-v3 (1 x 102) active
        Average EEG reference (1 x 60) active
    Found the data of interest:
        t =    -199.80 …     499.49 ms (Left visual)
        0 CTF compensation matrices available
        nave = 64 - aspect type = 100
Projections have already been applied. Setting proj attribute to True.
Applying baseline correction (mode: mean)
Reading forward solution from C:\Users\jeffr\mne_data\MNE-sample-
data\MEG\sample\sample_audvis-meg-eeg-oct-6-fwd.fif…
    Reading a source space…
    Computing patch statistics…
    Patch information added…
    Distance information added…
    [done]
    Reading a source space…
    Computing patch statistics…
    Patch information added…
    Distance information added…
    [done]
    2 source spaces read
    Desired named matrix (kind = 3523) not available
    Read MEG forward solution (7498 sources, 306 channels, free orientations)
    Desired named matrix (kind = 3523) not available
    Read EEG forward solution (7498 sources, 60 channels, free orientations)
    Forward solutions combined: MEG, EEG
    Source spaces transformed to the forward solution coordinate frame
```

1.3.2 Dipole Line Plots

In the paper, the authors say that their spatial regularization is set to 30% of the maximum, and the temporal regularization is set to 1%. Doing some math shows that α should be set to 31, and the L_1 ratio should be $\frac{1}{31}$ to yield the regularization parameters.

[9]:
```python
# loose orientation and depth bias compensation parameters
loose, depth = 0.2, 0.9

# dSPM-initialized weights
inverse_operator = make_inverse_operator(evoked.info, forward, cov,
                                         loose=loose, depth=depth)
stc_dspm = apply_inverse(evoked, inverse_operator, lambda2=1. / 9.,
                         method='dSPM')
```

```
Converting forward solution to surface orientation
    Average patch normals will be employed in the rotation to the local surface
coordinates…
    Converting to surface-based source orientations…
    [done]
info["bads"] and noise_cov["bads"] do not match, excluding bad channels from
both
Computing inverse operator with 364 channels.
    364 out of 366 channels remain after picking
Selected 364 channels
Creating the depth weighting matrix…
    203 planar channels
    limit = 7262/7498 = 10.020865
    scale = 2.58122e-08 exp = 0.9
Applying loose dipole orientations to surface source spaces: 0.2
Whitening the forward solution.
    Created an SSP operator (subspace dimension = 4)
Computing rank from covariance with rank=None
    Using tolerance 3.5e-13 (2.2e-16 eps * 305 dim * 5.2  max singular value)
    Estimated rank (mag + grad): 302
    MEG: rank 302 computed from 305 data channels with 3 projectors
    Using tolerance 1.1e-13 (2.2e-16 eps * 59 dim * 8.7  max singular value)
    Estimated rank (eeg): 58
    EEG: rank 58 computed from 59 data channels with 1 projector
    Setting small MEG eigenvalues to zero (without PCA)
    Setting small EEG eigenvalues to zero (without PCA)
Creating the source covariance matrix
Adjusting source covariance matrix.
Computing SVD of whitened and weighted lead field matrix.
    largest singular value = 5.96729
    scaling factor to adjust the trace = 9.38524e+18 (nchan = 364 nzero = 4)
Preparing the inverse operator for use…
    Scaled noise and source covariance from nave = 1 to nave = 64
    Created the regularized inverter
```

```
    Created an SSP operator (subspace dimension = 4)
    Created the whitener using a noise covariance matrix with rank 360 (4 small
eigenvalues omitted)
    Computing noise-normalization factors (dSPM)…
[done]
Applying inverse operator to "Left visual"…
    Picked 364 channels from the data
    Computing inverse…
    Eigenleads need to be weighted …
    Computing residual…
    Explained  60.0% variance
    Combining the current components…
    dSPM…
[done]
```

[10]:
```
alpha = 31
l1_ratio = 1/31

# TF-MxNE
dipoles, residual = tf_mixed_norm(
    evoked, forward, cov, alpha=alpha, l1_ratio=l1_ratio, loose=loose,
    depth=depth, maxit=200, tol=1e-6, weights=stc_dspm, weights_min=8.,
    debias=True, wsize=16, tstep=4, window=0.05, return_as_dipoles=True,
    return_residual=True)

# crop time window to see dipole amplitudes around onset of stimulus
for dip in dipoles:
    dip.crop(tmin=-0.05, tmax=0.3)
evoked.crop(tmin=-0.05, tmax=0.3)
residual.crop(tmin=-0.05, tmax=0.3)
```

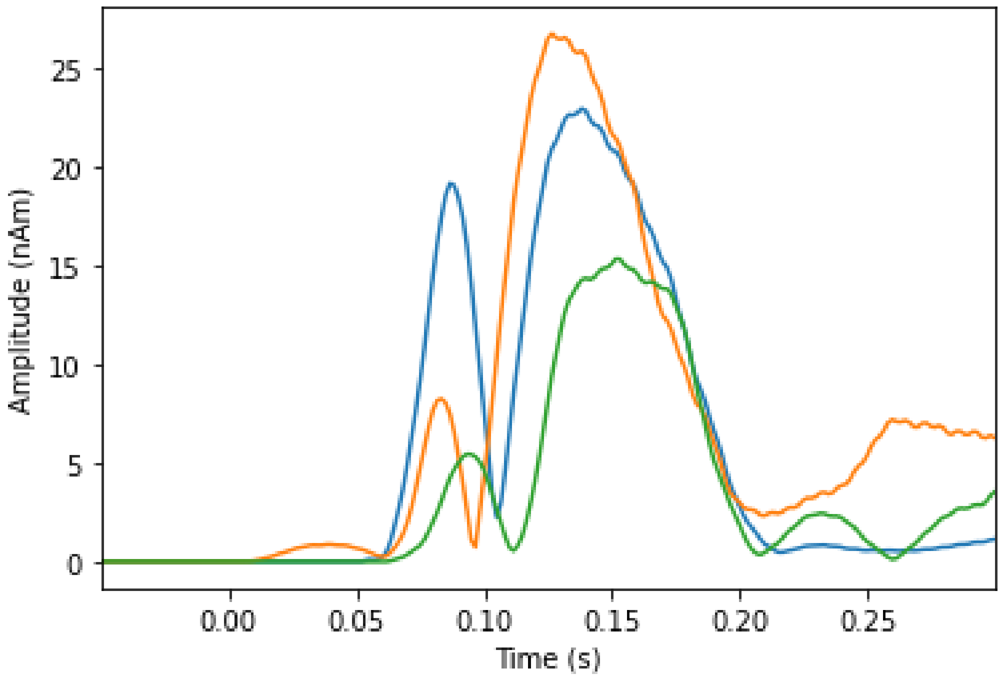
```
Converting forward solution to surface orientation
    Average patch normals will be employed in the rotation to the local surface
coordinates…
    Converting to surface-based source orientations…
    [done]
info["bads"] and noise_cov["bads"] do not match, excluding bad channels from
both
Computing inverse operator with 364 channels.
    364 out of 366 channels remain after picking
Selected 364 channels
Creating the depth weighting matrix…
Applying loose dipole orientations to surface source spaces: 0.2
Whitening the forward solution.
    Created an SSP operator (subspace dimension = 4)
Computing rank from covariance with rank=None
    Using tolerance 3.5e-13 (2.2e-16 eps * 305 dim * 5.2  max singular value)
    Estimated rank (mag + grad): 302
```

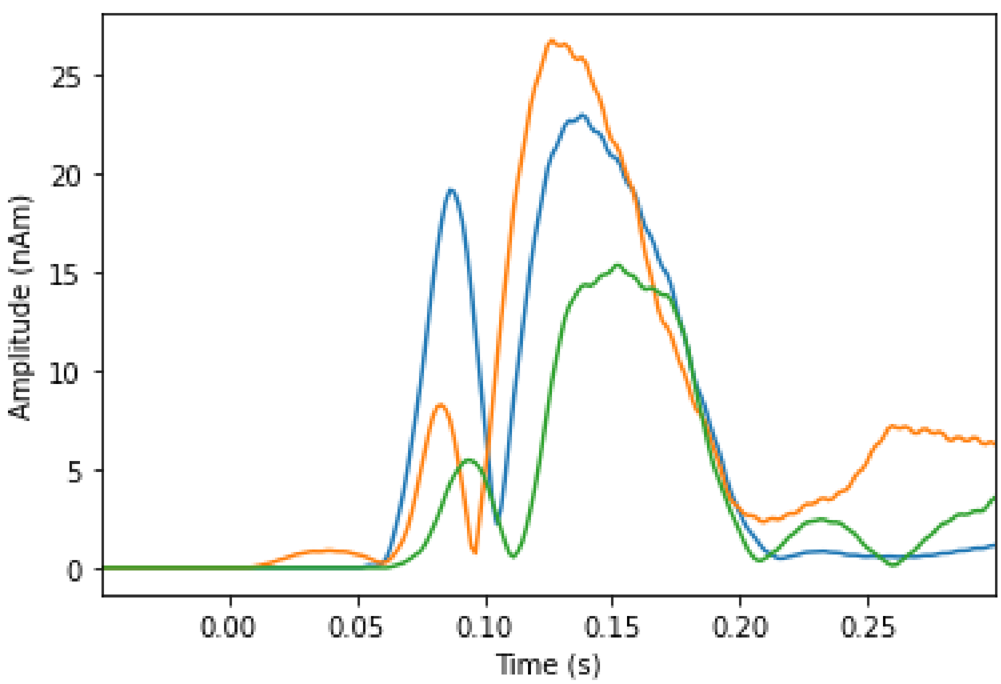
```
    MEG: rank 302 computed from 305 data channels with 3 projectors
    Using tolerance 1.1e-13 (2.2e-16 eps * 59 dim * 8.7  max singular value)
    Estimated rank (eeg): 58
    EEG: rank 58 computed from 59 data channels with 1 projector
    Setting small MEG eigenvalues to zero (without PCA)
    Setting small EEG eigenvalues to zero (without PCA)
Creating the source covariance matrix
Adjusting source covariance matrix.
Reducing source space to 985 sources
Whitening data matrix.
Using block coordinate descent with active set approach

dgap 3.78e+02 :: p_obj 4348.639886 :: d_obj 3970.534182 :: n_active 3

    Iteration 10 :: n_active 4
    dgap 4.63e+00 :: p_obj 4312.527802 :: d_obj 4307.895182

    Iteration 20 :: n_active 4
    dgap 9.25e-01 :: p_obj 4311.412273 :: d_obj 4310.487516

dgap 4.95e+01 :: p_obj 4311.162512 :: d_obj 4261.638243 :: n_active 3

    Iteration 10 :: n_active 3
    dgap 8.91e-02 :: p_obj 4309.682686 :: d_obj 4309.593632

    Iteration 20 :: n_active 3
    dgap 3.33e-03 :: p_obj 4309.682626 :: d_obj 4309.679292

dgap 3.33e-03 :: p_obj 4309.682626 :: d_obj 4309.679292 :: n_active 3

Convergence reached!

Debiasing did not converge after 1000 iterations! max(|D - D0| = 5.699910e-04 >=
1.000000e-06)
[done]
```

[10]:
```
<Evoked | 'Left visual' (average, N=64), -0.049949 - 0.29969 sec, baseline
-0.199795 - 0 sec (baseline period was cropped after baseline correction), 364
ch, ~3.8 MB>
```

[11]:
```
plot_dipole_amplitudes(dipoles)
```

[11]:

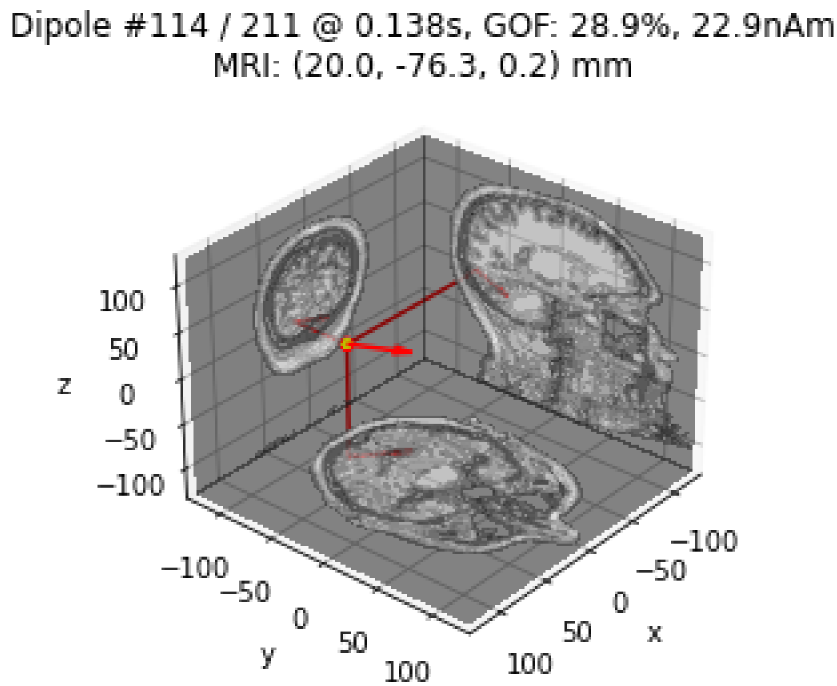


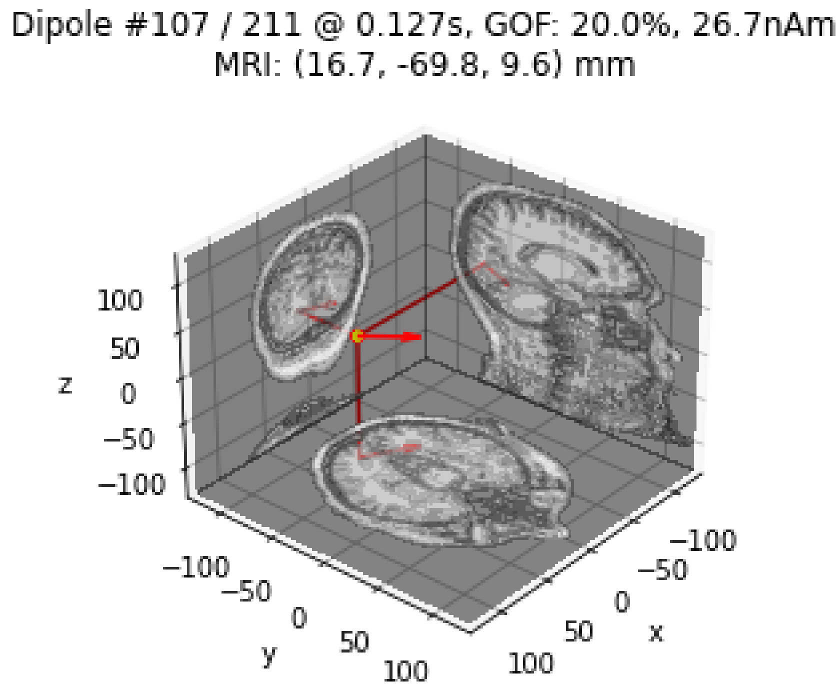


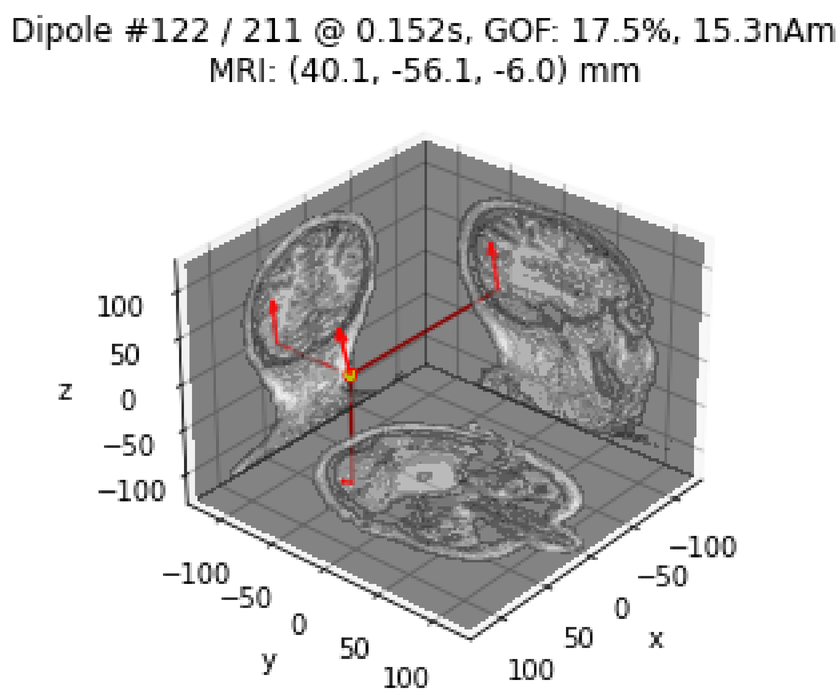
This time, we can see that we isolate 3 dipoles with these new parameters. The original paper also got 3 dipoles, but the source amplitudes are not the same. It's unclear why this is, and it would be nearly impossible to determine the cause of the difference without seeing the authors' original code. We proceed with the rest of the pipeline.

1.3.3 Dipole Source Locations

[12]:
```
for dip in dipoles:
    plot_dipole_locations(dip, forward['mri_head_t'], 'sample',
                          subjects_dir=subjects_dir, mode='orthoview',
                          idx='amplitude')
```



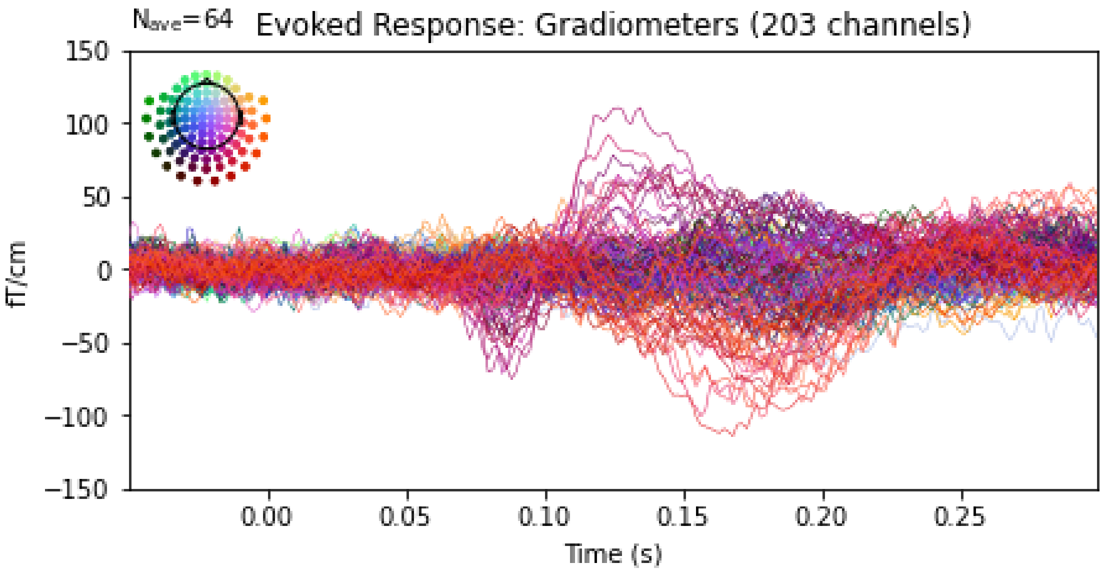

Dipole #107 / 211 @ 0.127s, GOF: 20.0%, 26.7nAm
MRI: (16.7, -69.8, 9.6) mm
z
100
50
0
−50
−100
−100
−50
0
50
100
y
−100
−50
0
50
100
x


Dipole #122 / 211 @ 0.152s, GOF: 17.5%, 15.3nAm
MRI: (40.1, -56.1, -6.0) mm
z
100
50
0
−50
−100
−100
−50
0
50
100
y
−100
−50
0
50
100
x

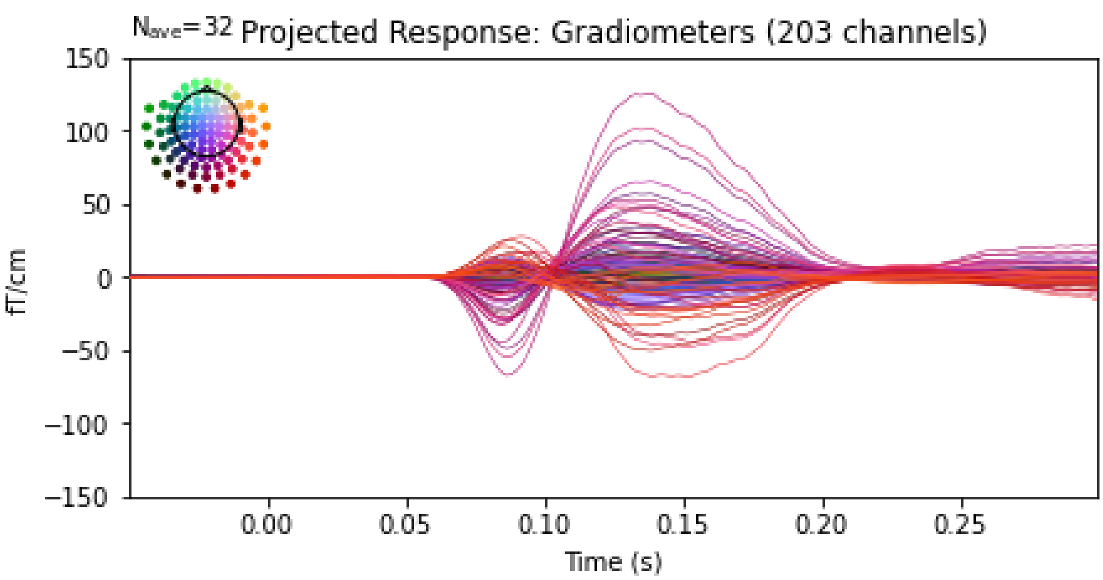
1.3.4 MEG Sensor Projection

[13]:
```python
# filtering for MEG channels
evoked.pick_types(meg='grad', exclude='bads')
residual.pick_types(meg='grad', exclude='bads')

# getting the dipole projection to sensor space
explained = mne.combine_evoked([evoked, residual], weights=[1, -1])

# limit amplitude
ylim = dict(grad=[-150, 150])

# all evoked data
evoked.plot(titles=dict(grad='Evoked Response: Gradiometers'), ylim=ylim,
            proj=True, time_unit='s')

# all explained data
explained.plot(titles=dict(grad='Projected Response: Gradiometers'), ylim=ylim,
            proj=True, time_unit='s')

# all residuals
residual.plot(titles=dict(grad='Residuals: Gradiometers'), ylim=ylim,
              proj=True, time_unit='s')
```

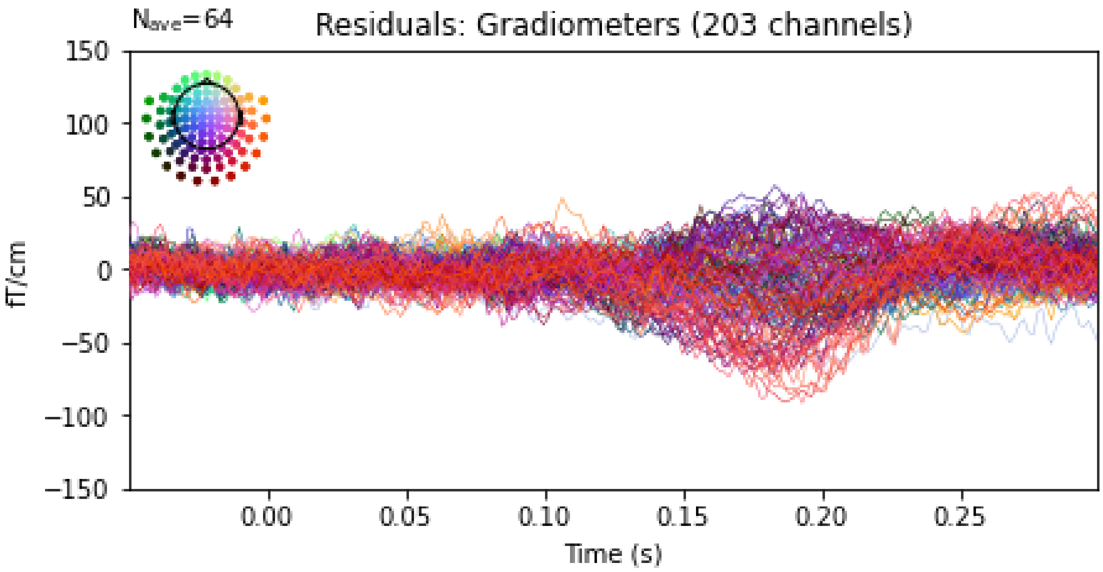
```
Removing projector <Projection | PCA-v1, active : True, n_channels : 102>
Removing projector <Projection | PCA-v2, active : True, n_channels : 102>
Removing projector <Projection | PCA-v3, active : True, n_channels : 102>
Removing projector <Projection | Average EEG reference, active : True,
n_channels : 60>
Removing projector <Projection | PCA-v1, active : True, n_channels : 102>
Removing projector <Projection | PCA-v2, active : True, n_channels : 102>
Removing projector <Projection | PCA-v3, active : True, n_channels : 102>
Removing projector <Projection | Average EEG reference, active : True,
n_channels : 60>
```



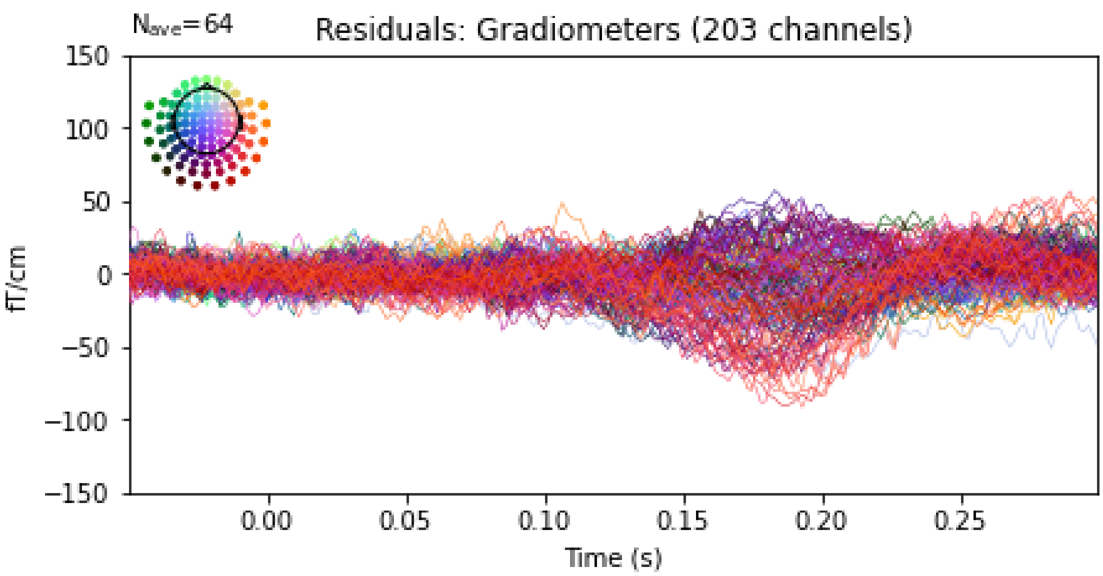
N_ave=64 Evoked Response: Gradiometers (203 channels)
fT/cm
150
100
50
0
−50
−100
−150
0.00
0.05
0.10
0.15
0.20
0.25
Time (s)



N_ave=32 Projected Response: Gradiometers (203 channels)
fT/cm
150
100
50
0
−50
−100
−150
0.00
0.05
0.10
0.15
0.20
0.25
Time (s)



[13]:

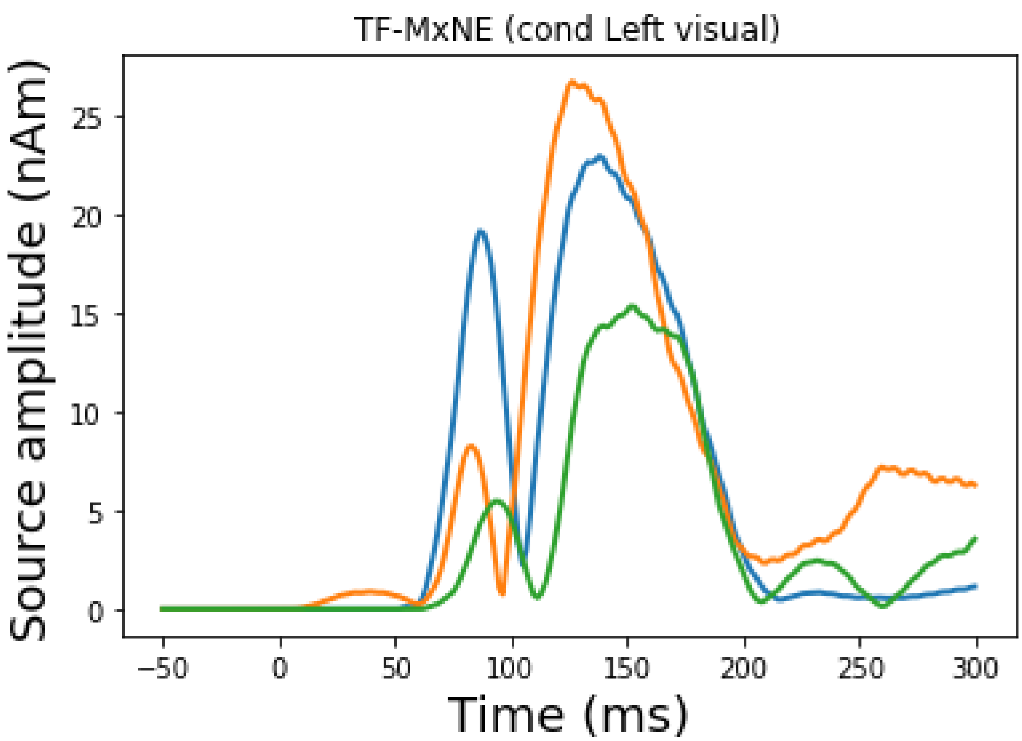


The projected response bears a good amount of resemblance to the original paper's Figure 8c. The three humps track fairly well; the main noticeable difference is the negative amplitude in the second hump. The authors' projection had all the negative amplitudes above -50 fT/cm, but our projection sees a few channels straying below that. However, differences are to be expected considering how we were unable to fully replicate the source amplitudes they calculated.

1.3.5 Source Localization 3D Visualization

[14]:
```
# make SourceEstimate object from dipole list
stc = make_stc_from_dipoles(dipoles, forward['src'])

# get source amplitudes again
plot_sparse_source_estimates(forward['src'], stc, bgcolor=(1, 1, 1),
                             opacity=0.1, fig_name="TF-MxNE (cond %s)"
                             % condition, modes=['sphere'], scale_factors=[1.])

# fancy 3D plot
time_label = 'TF-MxNE time=%0.2f ms'
clim = dict(kind='value', lims=[10e-9, 15e-9, 20e-9])
brain = stc.plot('sample', 'inflated', 'rh', views='medial',
                 clim=clim, time_label=time_label, smoothing_steps=5,
                 subjects_dir=subjects_dir, initial_time=150, time_unit='ms')
brain.add_label("V1", color="red", scalar_thresh=0.5, alpha=0.6, borders=False)
brain.add_label("V2", color="yellow", scalar_thresh=0.5, alpha=0.6,␣
 ↪borders=False)
```

```
Converting dipoles into a SourceEstimate.
[done]
Total number of active sources: 3
```



2 Additional Work: SPM Faces Dataset

2.1 Preliminaries

We now move into a new dataset that also focuses on visual stimuli. We will be using the SPM Faces Dataset, specifically the multi-modal part. The manual for the dataset can be found at https://www.fil.ion.ucl.ac.uk/spm/doc/spm8_manual.pdf#Chap:data:multimodal, but the primary thing to know is that subjects are shown a face and a scrambled face. This dataset is also built into MNE-Python, but we will start from the raw data and perform all the cleaning and computation kindly performed for us in the original Martinos dataset. Specifically, we will filter the dataset to remove high frequency noise, average chunks of the data, and compute the forward model.

[15]:
```
from mne.datasets import spm_face
from mne.preprocessing import ICA, create_eog_epochs
from mne import io, combine_evoked

print(__doc__)

# where stuff is
data_path = spm_face.data_path()
subjects_dir = data_path / 'subjects'
raw_fname_1 = data_path / 'MEG' / 'spm' / 'SPM_CTF_MEG_example_faces1_3D.ds'
raw_fname_1 = data_path / 'MEG' / 'spm' / 'SPM_CTF_MEG_example_faces2_3D.ds'
```

```
Automatically created module for IPython interactive environment
```

2.2 Preprocessing

We will calculate everything we mentioned from one run. We will go through the following pipeline.

1. Filter the data for frequencies between 1 and 30 Hz. This excludes higher frequency noise components and the lower frequency brain states that are likely not relevant to the visual stimuli.
2. Specify when the events occur (when the subject is shown a normal face or a scrambled one).
3. Crop for the time window around the event, using the 200 milliseconds before and 600 milliseconds after.
4. Perform ICA to denoise. Naturally, some eye blinking occurs throughout the recording. EOG data is used to remove these artifacts related to eye movement during ICA.
5. Get average response to being shown stimuli.
6. Determine the difference between being shown a normal and scrambled face.
7. Calculate the noise covariance matrix.

[16]:
```
# read first run data file
raw = io.read_raw_ctf(raw_fname_1, preload=True)
```

```python
# get only the good channels
picks = mne.pick_types(raw.info, meg=True, exclude='bads')
```

```
ds directory : C:\Users\jeffr\mne_data\MNE-spm-
face\MEG\spm\SPM_CTF_MEG_example_faces2_3D.ds
    res4 data read.
    hc data read.
    Separate EEG position data file not present.
    Quaternion matching (desired vs. transformed):
      -0.86   71.83    0.00 mm <->   -0.86   71.83    0.00 mm (orig :  -44.19
58.93 -250.86 mm) diff =    0.000 mm
       0.86  -71.83    0.00 mm <->    0.86  -71.83    0.00 mm (orig :   53.59
-46.26 -254.64 mm) diff =    0.000 mm
      97.64    0.00    0.00 mm <->   97.64   -0.00    0.00 mm (orig :   76.54
71.03 -239.14 mm) diff =    0.000 mm
    Coordinate transformations established.
    Polhemus data for 3 HPI coils added
    Device coordinate locations for 3 HPI coils added
    Measurement info composed.
Finding samples for C:\Users\jeffr\mne_data\MNE-spm-face\MEG\spm\SPM_CTF_MEG_exa
mple_faces2_3D.ds\SPM_CTF_MEG_example_faces2_3D.meg4:
    System clock channel is available, checking which samples are valid.
    1 x 303154 = 303154 samples from 340 chs
Current compensation grade : 3
Reading 0 … 303153  =      0.000 …   631.569 secs…
```

[17]:
```python
# band-pass filter for 1 Hz to 30 Hz
raw.filter(1, 30, method='fir', fir_design='firwin')
```

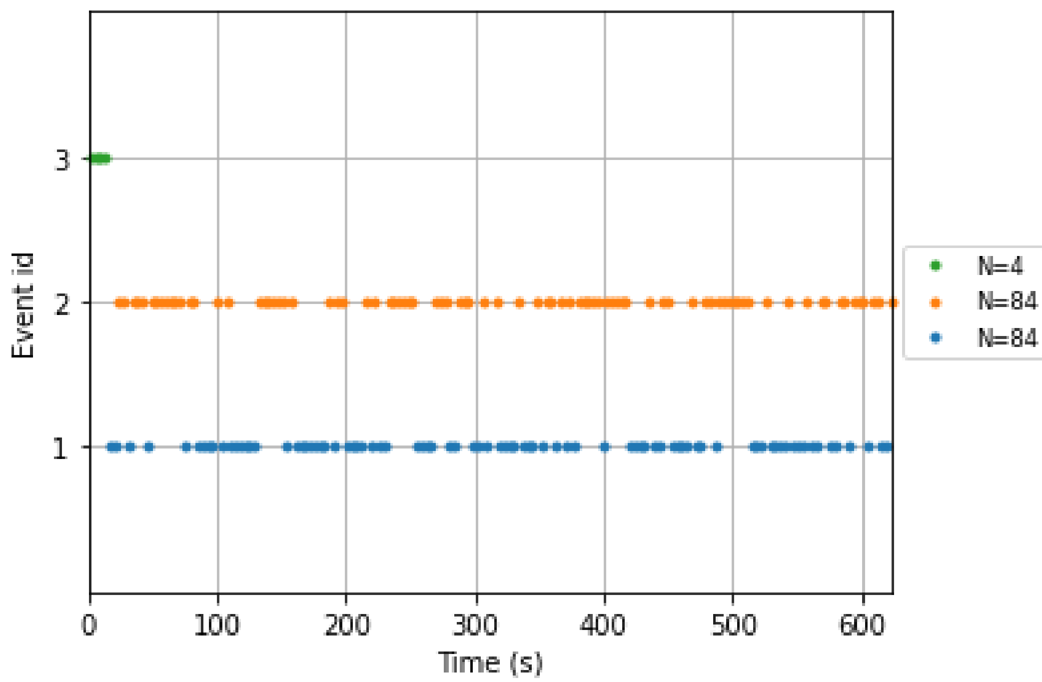
```
Filtering raw data in 1 contiguous segment
Setting up band-pass filter from 1 - 30 Hz

FIR filter parameters
---------------------
Designing a one-pass, zero-phase, non-causal bandpass filter:
- Windowed time-domain design (firwin) method
- Hamming window with 0.0194 passband ripple and 53 dB stopband attenuation
- Lower passband edge: 1.00
- Lower transition bandwidth: 1.00 Hz (-6 dB cutoff frequency: 0.50 Hz)
- Upper passband edge: 30.00 Hz
- Upper transition bandwidth: 7.50 Hz (-6 dB cutoff frequency: 33.75 Hz)
- Filter length: 1585 samples (3.302 sec)


[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   2 out of   2 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   3 out of   3 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   4 out of   4 | elapsed:    0.0s remaining:    0.0s
```

```
[Parallel(n_jobs=1)]: Done 303 out of 303 | elapsed:    2.7s finished
```

[17]:
```
<RawCTF | SPM_CTF_MEG_example_faces2_3D.meg4, 340 x 303154 (631.6 s), ~786.9 MB,
data loaded>
```

[18]:
```python
# find when "events" (the stimuli) occur
events = mne.find_events(raw, stim_channel='UPPT001')

# see when the events happen
mne.viz.plot_events(events, raw.info['sfreq'])
event_ids = {"faces": 1, "scrambled": 2}

# crop for time windows, or epochs
tmin, tmax = -0.2, 0.6
baseline = None  # no baseline as high-pass is applied
reject = dict(mag=5e-12)

epochs = mne.Epochs(raw, events, event_ids, tmin, tmax,  picks=picks,
                    baseline=baseline, preload=True, reject=reject)
```

```
172 events found
Event IDs: [1 2 3]
```




```
Not setting metadata
168 matching events found
No baseline correction applied
```

```
0 projection items activated
Using data from preloaded Raw for 168 events and 385 original time points …
0 bad epochs dropped
```

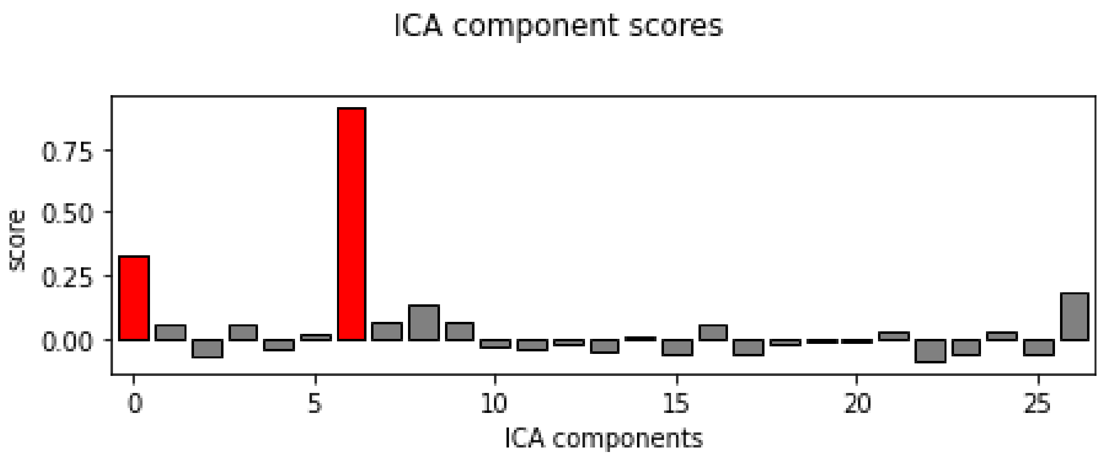
Notice that there seem to be 3 events! Nevertheless, we only need to look at events 1 and 2; event 3 is an initialization control. Indeed, we can see that these rest stimuli are presented first, before any of the normal or scrambled faces. We specify that we only want events 1 and 2 in the code above.

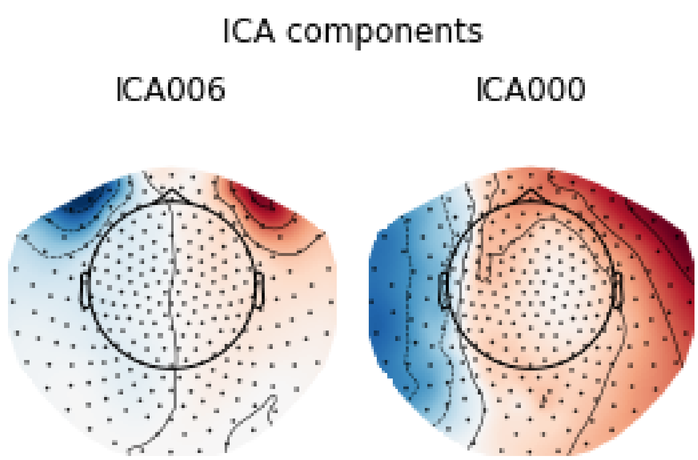
[19]:
```
# ICA, keep components that explain 95% of the variance
ica = ICA(n_components=0.95, random_state=0).fit(raw, decim=1, reject=reject)

# look at EOG data, remove artifacts that appear most related to EOG activity
eog_epochs = create_eog_epochs(raw, ch_name='MRT31-2908', reject=reject)
eog_inds, eog_scores = ica.find_bads_eog(eog_epochs, ch_name='MRT31-2908')
ica.plot_scores(eog_scores, eog_inds)
ica.plot_components(eog_inds)
ica.exclude += eog_inds[:1]
ica.plot_overlay(eog_epochs.average())
ica.apply(epochs)
```

```
Fitting ICA to data using 274 channels (please be patient, this may take a
while)
Removing 5 compensators from info because not all compensation channels were
picked.
Selecting by explained variance: 27 components
Fitting ICA took 24.0s.
Using EOG channel: MRT31-2908
EOG channel index for this subject is: [274]
Filtering the data to remove DC offset to help distinguish blinks from saccades
Setting up band-pass filter from 1 - 10 Hz

FIR filter parameters
---------------------
Designing a two-pass forward and reverse, zero-phase, non-causal bandpass
filter:
- Windowed frequency-domain design (firwin2) method
- Hann window
- Lower passband edge: 1.00
- Lower transition bandwidth: 0.50 Hz (-12 dB cutoff frequency: 0.75 Hz)
- Upper passband edge: 10.00 Hz
- Upper transition bandwidth: 0.50 Hz (-12 dB cutoff frequency: 10.25 Hz)
- Filter length: 4800 samples (10.000 sec)

Now detecting blinks and generating corresponding events
Found 67 significant peaks
Number of EOG events detected: 67
Not setting metadata
67 matching events found
```

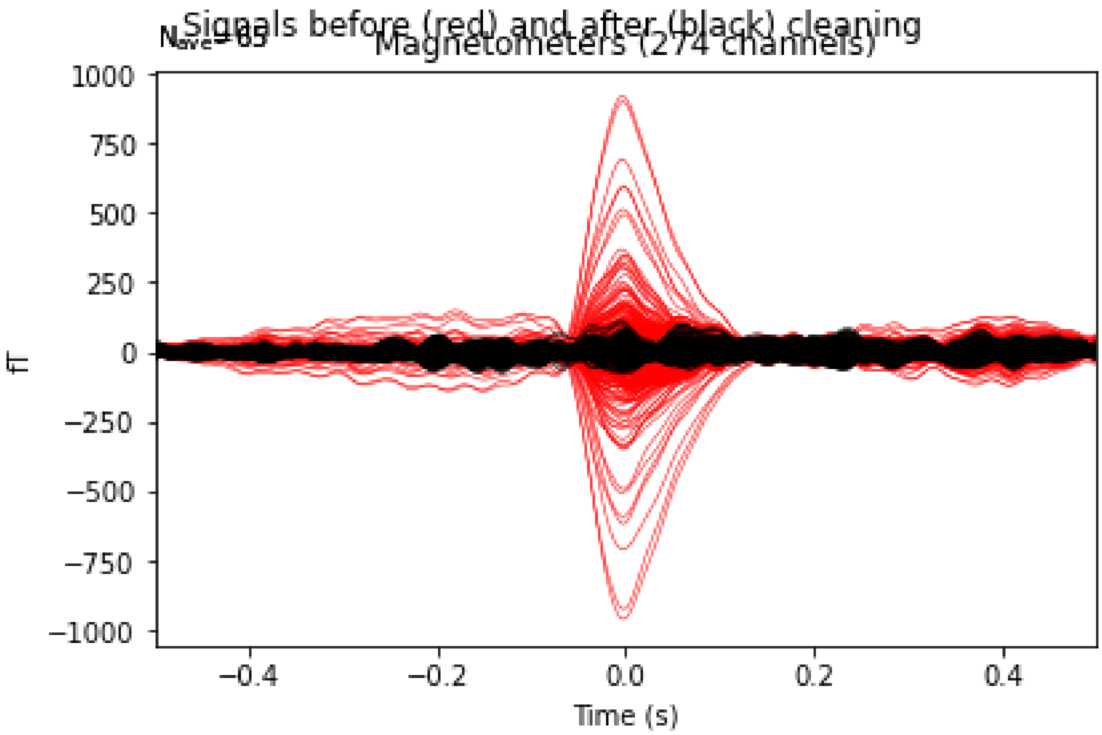
```
No baseline correction applied
Using data from preloaded Raw for 67 events and 481 original time points …
2 bad epochs dropped
Using EOG channel: MRT31-2908
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.0s finished
```






```
Applying ICA to Evoked instance
    Transforming to ICA space (27 components)
    Zeroing out 1 ICA component
    Projecting back using 274 PCA components
```




```
Applying ICA to Epochs instance
    Transforming to ICA space (27 components)
    Zeroing out 1 ICA component
    Projecting back using 274 PCA components
```

[19]:

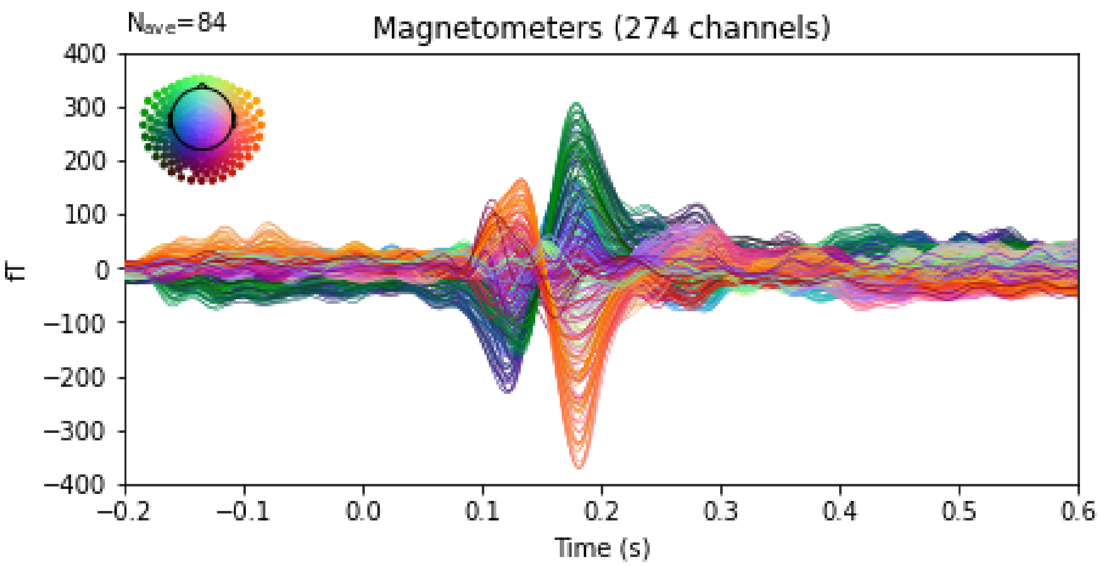
```
<Epochs |  168 events (all good), -0.2 - 0.6 sec, baseline off, ~150.0 MB, data
loaded,
 'faces': 84
 'scrambled': 84>
```

[20]:

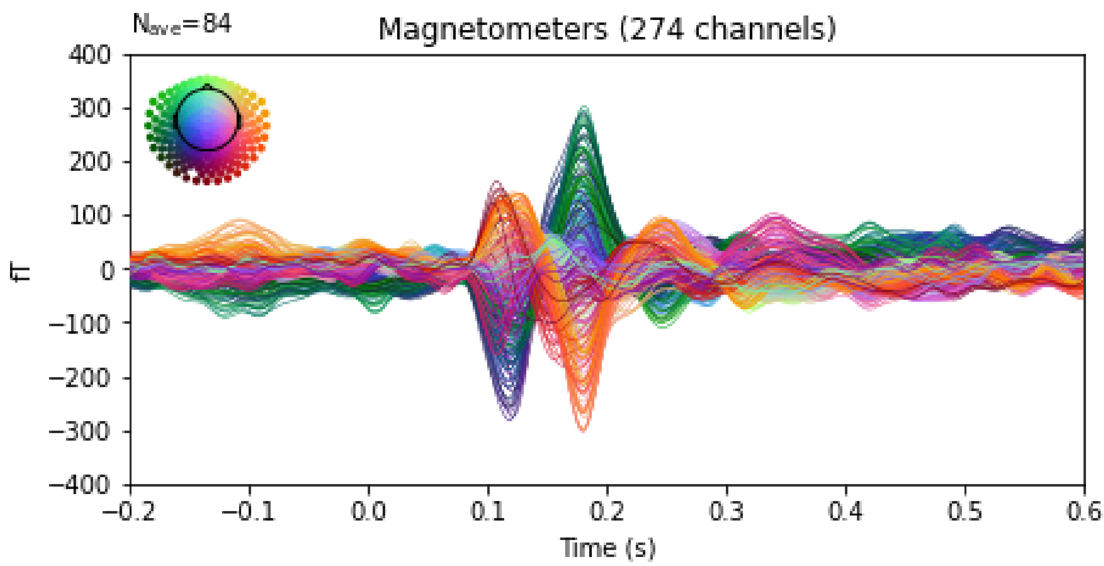
```python
# get average response, normal face and scrambled face
evoked = [epochs[k].average() for k in event_ids]

# get the differential response
contrast = combine_evoked(evoked, weights=[-1, 1])
evoked.append(contrast)

for e in evoked:
    e.plot(ylim=dict(mag=[-400, 400]))

plt.show()

# noise covariance matrix
```

```
noise_cov = mne.compute_covariance(epochs, tmax=0, method='shrunk',
                                   rank=None)
```

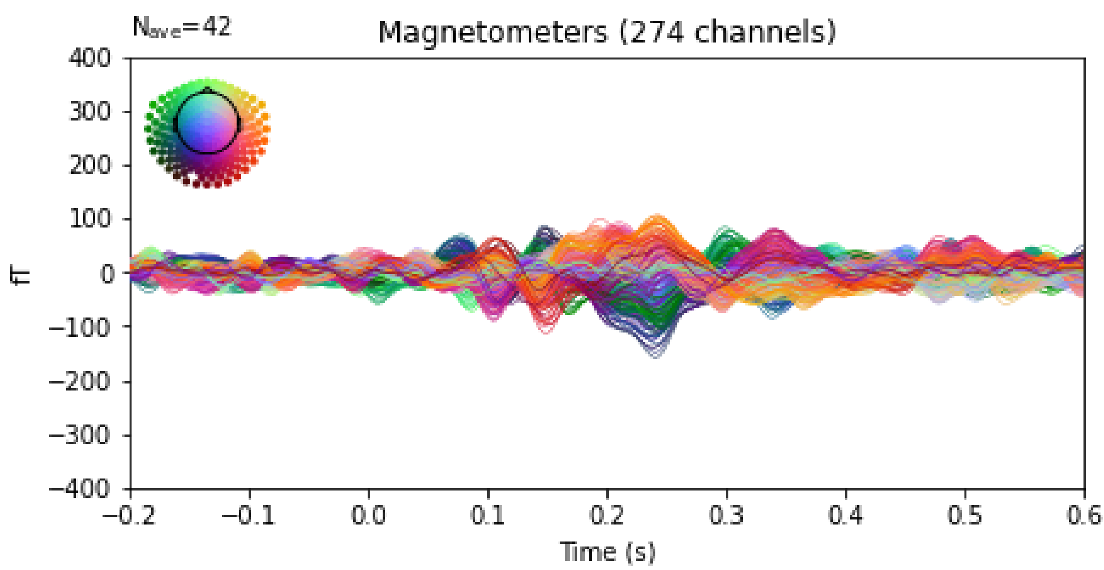
```
Removing 5 compensators from info because not all compensation channels were
picked.
Removing 5 compensators from info because not all compensation channels were
picked.
Removing 5 compensators from info because not all compensation channels were
picked.
```




```
Removing 5 compensators from info because not all compensation channels were
picked.
Removing 5 compensators from info because not all compensation channels were
picked.
Removing 5 compensators from info because not all compensation channels were
picked.
```




```
Removing 5 compensators from info because not all compensation channels were
picked.
Removing 5 compensators from info because not all compensation channels were
picked.
Removing 5 compensators from info because not all compensation channels were
picked.
```




```
Removing 5 compensators from info because not all compensation channels were
picked.
```

```
Computing rank from data with rank=None
    Using tolerance 6.8e-09 (2.2e-16 eps * 274 dim * 1.1e+05  max singular
value)
    Estimated rank (mag): 274
    MAG: rank 274 computed from 274 data channels with 0 projectors
Reducing data rank from 274 -> 274
Estimating covariance using SHRUNK
Done.
Number of samples used : 16296
[done]
```

The first plot above shows the response to normal faces at various sensor locations (shown by the color code on the top left), and the second plot shows the response to scrambled ones. The last one is the overall differential response between shown a normal and scrambled face.

2.3 Some Exploratory Visualization and Validation

Next, we make a virtual MEG helmet to see the magnetic fields. Unfortunately, this pops out in a separate window to retain the 3D functionality, so the figure cannot be seen in the notebook.

[21]:
```
# the trans.fif file is needed to run a projection onto the virtual MEG helmet
trans_fname = data_path / 'MEG' / 'spm' /␣
 ↪'SPM_CTF_MEG_example_faces1_3D_raw-trans.fif'

# we want to map the average evoked response to a normal face onto the MEG
maps = mne.make_field_map(evoked[0], trans_fname, subject='spm',
                          subjects_dir=subjects_dir, n_jobs=1)

# we specifically look at what the MEG fields look like at 150 milliseconds␣
 ↪after the stimulus is shown
evoked[0].plot_field(maps, time=0.150)
```

```
Getting helmet for system CTF_275
Prepare MEG mapping…
Removing 5 compensators from info because not all compensation channels were
picked.
Computing dot products for 274 coils…
Computing dot products for 342 surface locations…
Field mapping data ready
    Preparing the mapping matrix…
    Truncating at 99/274 components to omit less than 0.0001 (9.9e-05)
```

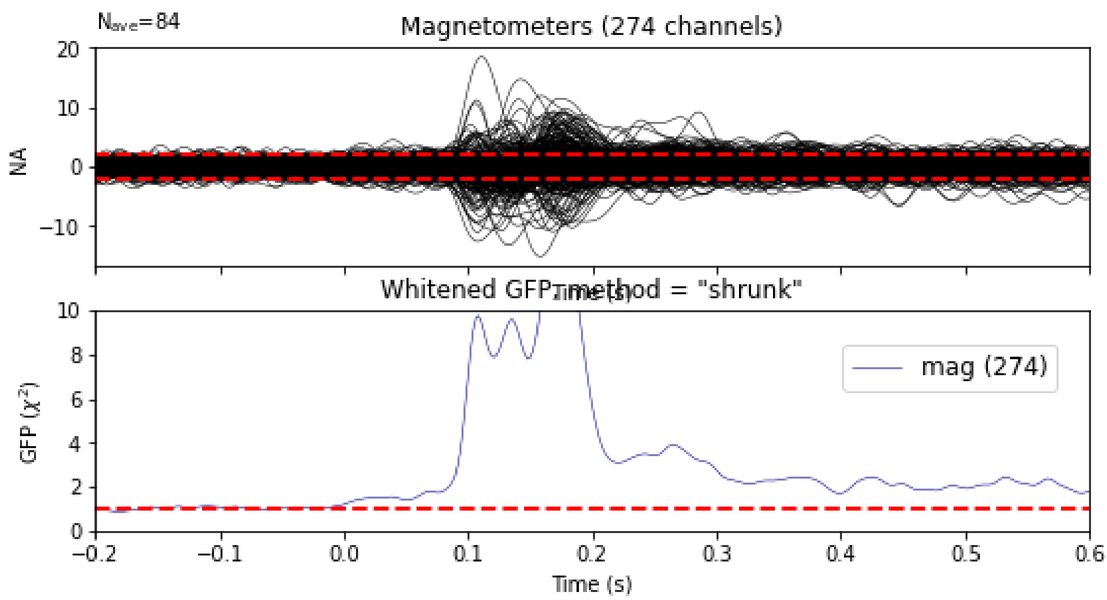
[21]:
```
<mne.viz.backends._pyvista.PyVistaFigure at 0x1ea8bea7730>
```

The next step we can take is to check whether the noise we found when we calculated the noise covariance matrix satisfies the normality assumption. To check whether the noise is white Gaussian noise, we can "whiten" the MEG data. This transformation makes it such that the channels become independent of one another. In this projection, we would expect to see the MEG data to oscillate around a mean of 0 with a standard deviation of 1 in a normal distribution. Therefore, we would expect to see 95% of the data to lie between 2 standard deviations, which we will draw in red.

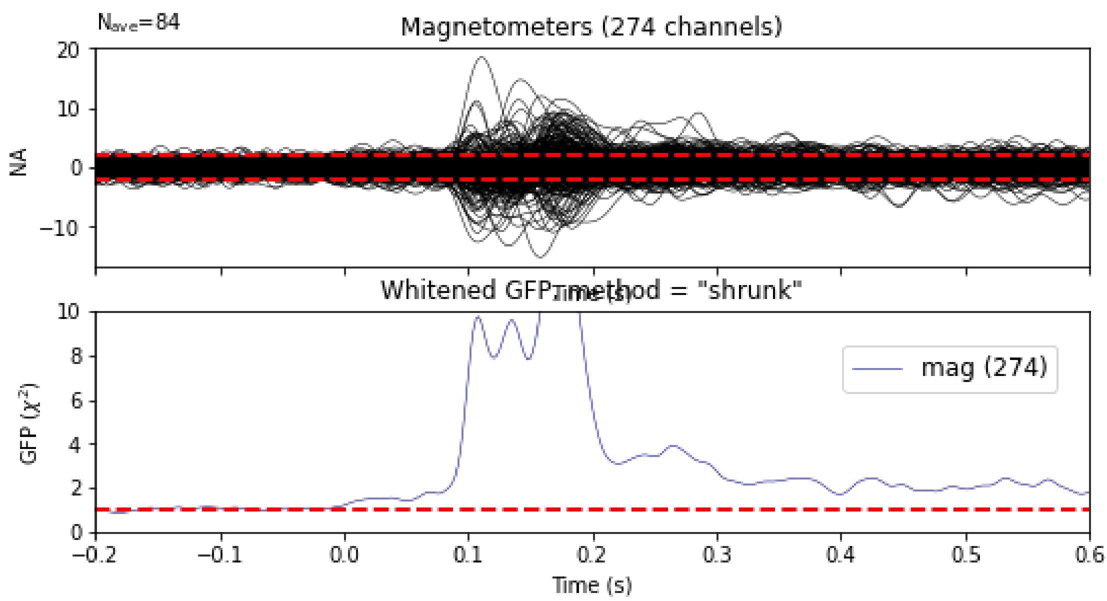
Another graph we can make is the global field power, which shows how much power is present across all channels. For this graph, we should expect to see the baseline amount of power to be around 1, which we will also draw in red.

[22]:
```
evoked[0].plot_white(noise_cov)
```

```
Removing 5 compensators from info because not all compensation channels were
picked.
Computing rank from covariance with rank=None
    Using tolerance 4.6e-14 (2.2e-16 eps * 274 dim * 0.76  max singular value)
    Estimated rank (mag): 274
    MAG: rank 274 computed from 274 data channels with 0 projectors
Computing rank from covariance with rank={'mag': 274}
    Setting small MAG eigenvalues to zero (without PCA)
    Created the whitener using a noise covariance matrix with rank 274 (0 small
eigenvalues omitted)
```



[22]:



Indeed, before the onset of the stimulus, we can see that the MEG data appear to satisfy the white Gaussian noise assumption. Shortly after onset, there are clearly signals between 80 and 200 milliseconds due to the stimulus; the data cannot be explained by noise. This was obvious from the line plots before, but we can now show that the noise does indeed appear to be Gaussian. We can also see that even 200 to 600 milliseconds after the stimulus, there appears to still be some evoked response not fully attributable to noise, although the power of these signals is much lower.

2.4 Forward Solution

This time, we are not babied and provided a forward solution, so we will have to calculate it. We are given the sensor locations and the dipole set we will use, so now we use those to calculate the projection. Earlier, we indicated where the transformation between the head and the sensors are (`trans_fname`), so we do not need to load this again. We will, however, need to tell where our dipole sources are and how the magnetic signals from dipoles would lose its power when travelling through the brain.

From now on, we will be working with the `contrast` object, which shows the differential responses to being shown a normal and a scrambled face.

[23]:
```
# dipole sources library
src = data_path / 'subjects' / 'spm' / 'bem' / 'spm-oct-6-src.fif'

# boundary element method used to describe how the brain attenuates the magnetic signals from the dipoles
bem = data_path / 'subjects' / 'spm' / 'bem' / 'spm-5120-5120-5120-bem-sol.fif'

# forward solution
forward = mne.make_forward_solution(contrast.info, trans_fname, src, bem)
```

```
Source space          : C:\Users\jeffr\mne_data\MNE-spm-
face\subjects\spm\bem\spm-oct-6-src.fif
MRI -> head transform : C:\Users\jeffr\mne_data\MNE-spm-
face\MEG\spm\SPM_CTF_MEG_example_faces1_3D_raw-trans.fif
Measurement data      : instance of Info
Conductor model   : C:\Users\jeffr\mne_data\MNE-spm-
face\subjects\spm\bem\spm-5120-5120-5120-bem-sol.fif
Accurate field computations
Do computations in head coordinates
Free source orientations

Reading C:\Users\jeffr\mne_data\MNE-spm-face\subjects\spm\bem\spm-
oct-6-src.fif…
Read 2 source spaces a total of 8196 active source locations

Coordinate transformation: MRI (surface RAS) -> head
     0.999622  0.006802  0.026647      -2.80 mm
    -0.014131  0.958276  0.285497       6.72 mm
    -0.023593 -0.285765  0.958009       9.43 mm
     0.000000  0.000000  0.000000       1.00

Read 303 MEG channels from info
Read 29 MEG compensation channels from info
5 compensation data sets in info
Setting up compensation data…
    Desired compensation data (3) found.
    All compensation channels found.
    Preselector created.
    Compensation data matrix created.
    Postselector created.
105 coil definitions read
Coordinate transformation: MEG device -> head
     0.998986 -0.036492 -0.026353      -1.38 mm
     0.039828  0.989385  0.139752       1.10 mm
     0.020973 -0.140660  0.989836      63.23 mm
     0.000000  0.000000  0.000000       1.00
MEG coil definitions created in head coordinates.
Removing 5 compensators from info because not all compensation channels were
picked.
Source spaces are now in head coordinates.

Setting up the BEM model using C:\Users\jeffr\mne_data\MNE-spm-
face\subjects\spm\bem\spm-5120-5120-5120-bem-sol.fif…

Loading surfaces…

Loading the solution matrix…
```

```
Three-layer model surfaces loaded.
Loaded linear collocation BEM solution from C:\Users\jeffr\mne_data\MNE-spm-
face\subjects\spm\bem\spm-5120-5120-5120-bem-sol.fif
Employing the head->MRI coordinate transform with the BEM model.
BEM model spm-5120-5120-5120-bem-sol.fif is now set up

Source spaces are in head coordinates.
Checking that the sources are inside the surface (will take a few…)
Checking surface interior status for 4098 points…
    Found 1659/4098 points inside  an interior sphere of radius   52.6 mm
    Found    0/4098 points outside an exterior sphere of radius  100.0 mm
    Found    0/2439 points outside using surface Qhull
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
    Found    0/2439 points outside using solid angles
    Total 4098/4098 points inside the surface
Interior check completed in 2385.0 ms
Checking surface interior status for 4098 points…
    Found 1713/4098 points inside  an interior sphere of radius   52.6 mm
    Found    0/4098 points outside an exterior sphere of radius  100.0 mm
    Found    0/2385 points outside using surface Qhull
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    2.3s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    2.3s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
    Found    0/2385 points outside using solid angles
    Total 4098/4098 points inside the surface
Interior check completed in 2148.8 ms


[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    2.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    2.0s finished
Checking surface interior status for 303 points…
    Found   0/303 points inside  an interior sphere of radius   74.7 mm
    Found  29/303 points outside an exterior sphere of radius  158.6 mm
    Found 274/274 points outside using surface Qhull
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
    Found   0/  0 points outside using solid angles
    Total 0/303 points inside the surface
Interior check completed in 292.5 ms

Composing the field computation matrix…
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.2s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.8s remaining:    0.0s
```

```
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.8s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.9s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.9s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.9s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.9s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    2.2s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    2.2s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.9s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.9s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.6s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.6s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.6s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.9s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.9s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    2.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    2.0s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.3s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.4s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.4s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

Computing MEG at 8196 source locations (free orientations)…

[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.3s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    0.3s finished
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    4.8s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    4.8s finished


Finished.
```

2.5 Inverse Solution Using dSPM

At last, we can calculate an inverse solution. Since the Martinos dataset (the one we first ran TF-MxNE on) used dSPM to initialize weights, we will also use dSPM to get the inverse solution for the new dataset. We can then compare how dSPM might vary from other inverse solvers.

We will use a λ_2 regularization parameter of $\frac{1}{9}$ again, as we did for the Martinos dataset.

[24]:
```
lambda2 = 1. / 9.
method = 'dSPM'

inverse_operator = make_inverse_operator(contrast.info, forward, noise_cov,
                                         loose=0.2, depth=0.8)

# dSPM
stc_dspm = apply_inverse(contrast, inverse_operator, lambda2, method,␣
 ↪pick_ori=None)
```

```
Converting forward solution to surface orientation
    Average patch normals will be employed in the rotation to the local surface
coordinates…
    Converting to surface-based source orientations…
    [done]
Computing inverse operator with 274 channels.
    274 out of 274 channels remain after picking
Removing 5 compensators from info because not all compensation channels were
picked.
Selected 274 channels
Creating the depth weighting matrix…
    274 magnetometer or axial gradiometer channels
    limit = 8105/8196 = 10.003233
    scale = 4.43797e-11 exp = 0.8
Applying loose dipole orientations to surface source spaces: 0.2
Whitening the forward solution.
Removing 5 compensators from info because not all compensation channels were
picked.
Computing rank from covariance with rank=None
    Using tolerance 4.6e-14 (2.2e-16 eps * 274 dim * 0.76  max singular value)
    Estimated rank (mag): 274
    MAG: rank 274 computed from 274 data channels with 0 projectors
    Setting small MAG eigenvalues to zero (without PCA)
Creating the source covariance matrix
Adjusting source covariance matrix.
Computing SVD of whitened and weighted lead field matrix.
    largest singular value = 10.1882
    scaling factor to adjust the trace = 2.06797e+19 (nchan = 274 nzero = 0)
Removing 5 compensators from info because not all compensation channels were
picked.
Preparing the inverse operator for use…
    Scaled noise and source covariance from nave = 1 to nave = 42
    Created the regularized inverter
    The projection vectors do not apply to these channels.
    Created the whitener using a noise covariance matrix with rank 274 (0 small
eigenvalues omitted)
```

```
    Computing noise-normalization factors (dSPM)…
[done]
Applying inverse operator to "-faces + scrambled"…
    Picked 274 channels from the data
    Computing inverse…
    Eigenleads need to be weighted …
    Computing residual…
    Explained  56.2% variance
    Combining the current components…
    dSPM…
[done]
```

We can now visualize the estimated source amplitudes in the brain. As we will see in a pop-up window, the dSPM method does not induce sparsity; there is a gradient of activity estimated all throughout the brain, particularly in the posterior region where the occipital lobe is. However, as time goes on, the activity also propagates to the more medial regions, although there is significant activity at the occipital lobes.

[25]:
```
brain_dspm = stc_dspm.plot(hemi='both', subjects_dir=subjects_dir,␣
 ↪initial_time=0.150,
                  views=['ven'], clim={'kind': 'value', 'lims': [3., 6., 9.]})
```

2.6 Inverse Solution Using sLORETA

To see how different inverse algorithms generate different results, we use sLORETA next. We will use the same λ_2 parameter.

[26]:
```
lambda2 = 1. / 9.
method = 'sLORETA'

inverse_operator = make_inverse_operator(contrast.info, forward, noise_cov,
                                         loose=0.2, depth=0.8)

# sLORETA
stc_sloreta = apply_inverse(contrast, inverse_operator, lambda2, method,␣
 ↪pick_ori=None)
```

```
Converting forward solution to surface orientation
    Average patch normals will be employed in the rotation to the local surface
coordinates…
    Converting to surface-based source orientations…
    [done]
Computing inverse operator with 274 channels.
    274 out of 274 channels remain after picking
Removing 5 compensators from info because not all compensation channels were
picked.
Selected 274 channels
Creating the depth weighting matrix…
    274 magnetometer or axial gradiometer channels
```

```
    limit = 8105/8196 = 10.003233
    scale = 4.43797e-11 exp = 0.8
Applying loose dipole orientations to surface source spaces: 0.2
Whitening the forward solution.
Removing 5 compensators from info because not all compensation channels were
picked.
Computing rank from covariance with rank=None
    Using tolerance 4.6e-14 (2.2e-16 eps * 274 dim * 0.76  max singular value)
    Estimated rank (mag): 274
    MAG: rank 274 computed from 274 data channels with 0 projectors
    Setting small MAG eigenvalues to zero (without PCA)
Creating the source covariance matrix
Adjusting source covariance matrix.
Computing SVD of whitened and weighted lead field matrix.
    largest singular value = 10.1882
    scaling factor to adjust the trace = 2.06797e+19 (nchan = 274 nzero = 0)
Removing 5 compensators from info because not all compensation channels were
picked.
Preparing the inverse operator for use…
    Scaled noise and source covariance from nave = 1 to nave = 42
    Created the regularized inverter
    The projection vectors do not apply to these channels.
    Created the whitener using a noise covariance matrix with rank 274 (0 small
eigenvalues omitted)
    Computing noise-normalization factors (sLORETA)…
[done]
Applying inverse operator to "-faces + scrambled"…
    Picked 274 channels from the data
    Computing inverse…
    Eigenleads need to be weighted …
    Computing residual…
    Explained  56.2% variance
    Combining the current components…
    sLORETA…
[done]
```

[27]:
```
brain_sloreta = stc_sloreta.plot(hemi='both', subjects_dir=subjects_dir,␣
↪initial_time=0.150,
                    views=['ven'], clim={'kind': 'value', 'lims': [3., 6., 9.]})
```

Interestingly, sLORETA produces a much "ringier" solution. The source amplitudes see a lot of oscillations, even before the onset of the stimulus. The peak of the dipoles with the largest source amplitudes also occurs earlier than the solution found by dSPM.

2.7 Inverse Solution Using γ-MAP

Next up is γ-MAP. We will first need to load the command to run it.

[28]:
```python
from mne.inverse_sparse import gamma_map
```

Unlike dSPM and sLORETA, γ-MAP induces sparsity. For this algorithm, the noise covariance matrix is used and regularized first; γ-MAP uses an empirical Bayesian approach to determine the hyperparameter γ that determines the covariance matrix for the sources. The regularization parameter α we will use is 0.2. We need a relatively smaller regularization parameter since the amplitudes for the contrast is much smaller than either of the normal or scrambled faces. For this dataset, $\alpha \geq 0.3$ yields no active sources, and $\alpha = 0.15$ require 3852 iterations (more than half an hour) to converge (where the tolerance is 10^{-6}), which is far too many.

[29]:
```python
alpha = 0.2

# regularizing covariance matrix
cov = mne.cov.regularize(noise_cov, contrast.info, rank=None)

# gamma-MAP
dipoles, residual = gamma_map(
    contrast, forward, cov, alpha, xyz_same_gamma=False, return_residual=True,
    return_as_dipoles=True)
```

```
Removing 5 compensators from info because not all compensation channels were
picked.
Computing rank from covariance with rank=None
    Using tolerance 4.6e-14 (2.2e-16 eps * 274 dim * 0.76  max singular value)
    Estimated rank (mag): 274
    MAG: rank 274 computed from 274 data channels with 0 projectors
0 projection items activated
    MAG regularization : 0.1
Removing 5 compensators from info because not all compensation channels were
picked.
Computing rank from covariance with rank={'mag': 274}
    Setting small MAG eigenvalues to zero (without PCA)
Converting forward solution to surface orientation
    Average patch normals will be employed in the rotation to the local surface
coordinates…
    Converting to surface-based source orientations…
    [done]
Computing inverse operator with 274 channels.
    274 out of 274 channels remain after picking
Removing 5 compensators from info because not all compensation channels were
picked.
Selected 274 channels
Creating the depth weighting matrix…
Applying loose dipole orientations to surface source spaces: 0.2
Whitening the forward solution.
Removing 5 compensators from info because not all compensation channels were
picked.
Computing rank from covariance with rank=None
```

```
    Using tolerance 4.6e-14 (2.2e-16 eps * 274 dim * 0.76  max singular value)
    Estimated rank (mag): 274
    MAG: rank 274 computed from 274 data channels with 0 projectors
    Setting small MAG eigenvalues to zero (without PCA)
Creating the source covariance matrix
Adjusting source covariance matrix.
Whitening data matrix.
Iteration: 0     active set size: 24588  convergence: 7.785e-01
Iteration: 10    active set size: 24586  convergence: 1.194e-01
Iteration: 11    active set size: 24539  convergence: 1.074e-01
Iteration: 12    active set size: 24297  convergence: 9.891e-02
Iteration: 13    active set size: 23663  convergence: 9.315e-02
Iteration: 14    active set size: 22274  convergence: 8.953e-02
Iteration: 15    active set size: 20414  convergence: 8.759e-02
Iteration: 16    active set size: 18366  convergence: 8.698e-02
Iteration: 17    active set size: 16258  convergence: 8.740e-02
Iteration: 18    active set size: 14224  convergence: 8.863e-02
Iteration: 19    active set size: 12450  convergence: 9.048e-02
Iteration: 20    active set size: 10892  convergence: 9.281e-02
Iteration: 21    active set size: 9571   convergence: 9.548e-02
Iteration: 22    active set size: 8428   convergence: 9.840e-02
Iteration: 23    active set size: 7457   convergence: 1.015e-01
Iteration: 24    active set size: 6611   convergence: 1.046e-01
Iteration: 25    active set size: 5943   convergence: 1.078e-01
Iteration: 26    active set size: 5316   convergence: 1.109e-01
Iteration: 27    active set size: 4792   convergence: 1.138e-01
Iteration: 28    active set size: 4318   convergence: 1.166e-01
Iteration: 29    active set size: 3920   convergence: 1.192e-01
Iteration: 30    active set size: 3593   convergence: 1.215e-01
Iteration: 31    active set size: 3282   convergence: 1.236e-01
Iteration: 32    active set size: 3032   convergence: 1.253e-01
Iteration: 33    active set size: 2804   convergence: 1.267e-01
Iteration: 34    active set size: 2599   convergence: 1.277e-01
Iteration: 35    active set size: 2414   convergence: 1.282e-01
Iteration: 36    active set size: 2259   convergence: 1.283e-01
Iteration: 37    active set size: 2109   convergence: 1.280e-01
Iteration: 38    active set size: 1973   convergence: 1.272e-01
Iteration: 39    active set size: 1848   convergence: 1.259e-01
Iteration: 40    active set size: 1739   convergence: 1.242e-01
Iteration: 41    active set size: 1649   convergence: 1.219e-01
Iteration: 42    active set size: 1556   convergence: 1.193e-01
Iteration: 43    active set size: 1475   convergence: 1.162e-01
Iteration: 44    active set size: 1395   convergence: 1.127e-01
Iteration: 45    active set size: 1328   convergence: 1.089e-01
Iteration: 46    active set size: 1263   convergence: 1.048e-01
Iteration: 47    active set size: 1208   convergence: 1.004e-01
Iteration: 48    active set size: 1145   convergence: 9.577e-02
Iteration: 49    active set size: 1077   convergence: 9.103e-02
```

```
Iteration: 50    active set size: 1026   convergence: 8.622e-02
Iteration: 51    active set size: 991    convergence: 8.132e-02
Iteration: 52    active set size: 944    convergence: 7.637e-02
Iteration: 53    active set size: 903    convergence: 7.142e-02
Iteration: 54    active set size: 865    convergence: 6.654e-02
Iteration: 55    active set size: 823    convergence: 6.179e-02
Iteration: 56    active set size: 791    convergence: 5.720e-02
Iteration: 57    active set size: 750    convergence: 5.281e-02
Iteration: 58    active set size: 726    convergence: 4.879e-02
Iteration: 59    active set size: 696    convergence: 4.510e-02
Iteration: 60    active set size: 666    convergence: 4.163e-02
Iteration: 61    active set size: 639    convergence: 3.839e-02
Iteration: 62    active set size: 617    convergence: 3.537e-02
Iteration: 63    active set size: 590    convergence: 3.258e-02
Iteration: 64    active set size: 570    convergence: 3.002e-02
Iteration: 65    active set size: 555    convergence: 2.766e-02
Iteration: 66    active set size: 532    convergence: 2.551e-02
Iteration: 67    active set size: 512    convergence: 2.355e-02
Iteration: 68    active set size: 491    convergence: 2.176e-02
Iteration: 69    active set size: 470    convergence: 2.014e-02
Iteration: 70    active set size: 460    convergence: 1.867e-02
Iteration: 71    active set size: 446    convergence: 1.734e-02
Iteration: 72    active set size: 435    convergence: 1.613e-02
Iteration: 73    active set size: 419    convergence: 1.503e-02
Iteration: 74    active set size: 404    convergence: 1.404e-02
Iteration: 75    active set size: 392    convergence: 1.314e-02
Iteration: 76    active set size: 375    convergence: 1.233e-02
Iteration: 77    active set size: 361    convergence: 1.158e-02
Iteration: 78    active set size: 346    convergence: 1.091e-02
Iteration: 79    active set size: 335    convergence: 1.029e-02
Iteration: 80    active set size: 321    convergence: 9.726e-03
Iteration: 81    active set size: 313    convergence: 9.211e-03
Iteration: 82    active set size: 310    convergence: 8.739e-03
Iteration: 83    active set size: 296    convergence: 8.305e-03
Iteration: 84    active set size: 290    convergence: 7.905e-03
Iteration: 85    active set size: 278    convergence: 7.535e-03
Iteration: 86    active set size: 274    convergence: 7.192e-03
Iteration: 87    active set size: 265    convergence: 6.874e-03
Iteration: 88    active set size: 260    convergence: 6.577e-03
Iteration: 89    active set size: 250    convergence: 6.300e-03
Iteration: 90    active set size: 247    convergence: 6.040e-03
Iteration: 91    active set size: 237    convergence: 5.797e-03
Iteration: 92    active set size: 232    convergence: 5.567e-03
Iteration: 93    active set size: 224    convergence: 5.351e-03
Iteration: 94    active set size: 219    convergence: 5.146e-03
Iteration: 95    active set size: 215    convergence: 4.951e-03
Iteration: 96    active set size: 207    convergence: 4.767e-03
Iteration: 97    active set size: 204    convergence: 4.591e-03
```

```
Iteration: 98    active set size: 197    convergence: 4.423e-03
Iteration: 99    active set size: 192    convergence: 4.266e-03
Iteration: 100   active set size: 189    convergence: 4.116e-03
Iteration: 101   active set size: 186    convergence: 3.973e-03
Iteration: 102   active set size: 183    convergence: 3.835e-03
Iteration: 103   active set size: 182    convergence: 3.702e-03
Iteration: 104   active set size: 176    convergence: 3.575e-03
Iteration: 105   active set size: 172    convergence: 3.452e-03
Iteration: 106   active set size: 167    convergence: 3.339e-03
Iteration: 107   active set size: 164    convergence: 3.232e-03
Iteration: 108   active set size: 161    convergence: 3.129e-03
Iteration: 109   active set size: 156    convergence: 3.030e-03
Iteration: 110   active set size: 150    convergence: 2.934e-03
Iteration: 111   active set size: 147    convergence: 2.841e-03
Iteration: 112   active set size: 144    convergence: 2.751e-03
Iteration: 113   active set size: 142    convergence: 2.664e-03
Iteration: 114   active set size: 140    convergence: 2.579e-03
Iteration: 115   active set size: 135    convergence: 2.498e-03
Iteration: 116   active set size: 133    convergence: 2.424e-03
Iteration: 117   active set size: 132    convergence: 2.356e-03
Iteration: 118   active set size: 129    convergence: 2.291e-03
Iteration: 119   active set size: 128    convergence: 2.227e-03
Iteration: 120   active set size: 126    convergence: 2.166e-03
Iteration: 121   active set size: 125    convergence: 2.106e-03
Iteration: 122   active set size: 124    convergence: 2.048e-03
Iteration: 123   active set size: 121    convergence: 1.992e-03
Iteration: 124   active set size: 117    convergence: 1.938e-03
Iteration: 125   active set size: 116    convergence: 1.894e-03
Iteration: 127   active set size: 115    convergence: 1.811e-03
Iteration: 128   active set size: 113    convergence: 1.771e-03
Iteration: 129   active set size: 111    convergence: 1.732e-03
Iteration: 130   active set size: 109    convergence: 1.695e-03
Iteration: 131   active set size: 107    convergence: 1.658e-03
Iteration: 132   active set size: 105    convergence: 1.622e-03
Iteration: 133   active set size: 102    convergence: 1.586e-03
Iteration: 134   active set size: 101    convergence: 1.552e-03
Iteration: 136   active set size: 100    convergence: 1.486e-03
Iteration: 137   active set size: 98     convergence: 1.454e-03
Iteration: 138   active set size: 97     convergence: 1.423e-03
Iteration: 140   active set size: 95     convergence: 1.364e-03
Iteration: 142   active set size: 94     convergence: 1.307e-03
Iteration: 143   active set size: 92     convergence: 1.279e-03
Iteration: 144   active set size: 91     convergence: 1.253e-03
Iteration: 145   active set size: 86     convergence: 1.227e-03
Iteration: 146   active set size: 85     convergence: 1.201e-03
Iteration: 148   active set size: 83     convergence: 1.152e-03
Iteration: 149   active set size: 81     convergence: 1.128e-03
Iteration: 150   active set size: 77     convergence: 1.105e-03
```

```
Iteration: 151   active set size: 74     convergence: 1.083e-03
Iteration: 152   active set size: 72     convergence: 1.061e-03
Iteration: 156   active set size: 69     convergence: 9.778e-04
Iteration: 157   active set size: 68     convergence: 9.583e-04
Iteration: 159   active set size: 67     convergence: 9.207e-04
Iteration: 160   active set size: 66     convergence: 9.025e-04
Iteration: 161   active set size: 65     convergence: 8.848e-04
Iteration: 164   active set size: 64     convergence: 8.341e-04
Iteration: 168   active set size: 63     convergence: 7.718e-04
Iteration: 169   active set size: 62     convergence: 7.571e-04
Iteration: 171   active set size: 61     convergence: 7.288e-04
Iteration: 172   active set size: 60     convergence: 7.151e-04
Iteration: 173   active set size: 58     convergence: 7.017e-04
Iteration: 175   active set size: 57     convergence: 6.759e-04
Iteration: 182   active set size: 55     convergence: 5.941e-04
Iteration: 187   active set size: 54     convergence: 5.430e-04
Iteration: 191   active set size: 53     convergence: 5.059e-04
Iteration: 192   active set size: 52     convergence: 4.971e-04
Iteration: 196   active set size: 51     convergence: 4.638e-04
Iteration: 197   active set size: 50     convergence: 4.559e-04
Iteration: 201   active set size: 49     convergence: 4.258e-04
Iteration: 202   active set size: 48     convergence: 4.187e-04
Iteration: 203   active set size: 47     convergence: 4.117e-04
Iteration: 206   active set size: 46     convergence: 3.915e-04
Iteration: 212   active set size: 45     convergence: 3.546e-04
Iteration: 219   active set size: 44     convergence: 3.166e-04
Iteration: 221   active set size: 43     convergence: 3.067e-04
Iteration: 224   active set size: 42     convergence: 2.924e-04
Iteration: 225   active set size: 40     convergence: 2.878e-04
Iteration: 228   active set size: 39     convergence: 2.746e-04
Iteration: 231   active set size: 38     convergence: 2.620e-04
Iteration: 234   active set size: 37     convergence: 2.501e-04
Iteration: 241   active set size: 36     convergence: 2.247e-04
Iteration: 243   active set size: 35     convergence: 2.180e-04
Iteration: 247   active set size: 34     convergence: 2.053e-04
Iteration: 250   active set size: 33     convergence: 1.963e-04
Iteration: 260   active set size: 32     convergence: 1.693e-04
Iteration: 277   active set size: 31     convergence: 1.324e-04
Iteration: 284   active set size: 30     convergence: 1.199e-04
Iteration: 286   active set size: 29     convergence: 1.166e-04
Iteration: 300   active set size: 28     convergence: 9.584e-05
Iteration: 301   active set size: 27     convergence: 9.452e-05
Iteration: 313   active set size: 25     convergence: 8.014e-05
Iteration: 315   active set size: 24     convergence: 7.798e-05
Iteration: 318   active set size: 23     convergence: 7.485e-05
Iteration: 335   active set size: 22     convergence: 5.967e-05
Iteration: 382   active set size: 21     convergence: 3.301e-05
Iteration: 384   active set size: 20     convergence: 3.219e-05
```

```
Iteration: 385   active set size: 19     convergence: 3.178e-05
Iteration: 401   active set size: 18     convergence: 2.598e-05
Iteration: 429   active set size: 17     convergence: 1.826e-05
Iteration: 433   active set size: 16     convergence: 1.737e-05
Iteration: 438   active set size: 15     convergence: 1.631e-05
Iteration: 441   active set size: 14     convergence: 1.571e-05
Iteration: 495   active set size: 13     convergence: 8.000e-06
Iteration: 509   active set size: 12     convergence: 6.722e-06
Iteration: 513   active set size: 11     convergence: 6.397e-06
Iteration: 556   active set size: 10     convergence: 3.758e-06
Iteration: 603   active set size: 9      convergence: 2.109e-06
Iteration: 665   active set size: 9      convergence: 9.891e-07

Convergence reached !

Removing 5 compensators from info because not all compensation channels were
picked.
Explained   2.5% variance
[done]
```

Notice that the variance the solution is able to explain is only 2.5%. Theoretically, we could increase this by lowering α (for $\alpha = 0.15$, 6.8% of the variance is explained), but the algorithm runs much too slowly. We will proceed with looking at the dipoles anyway.

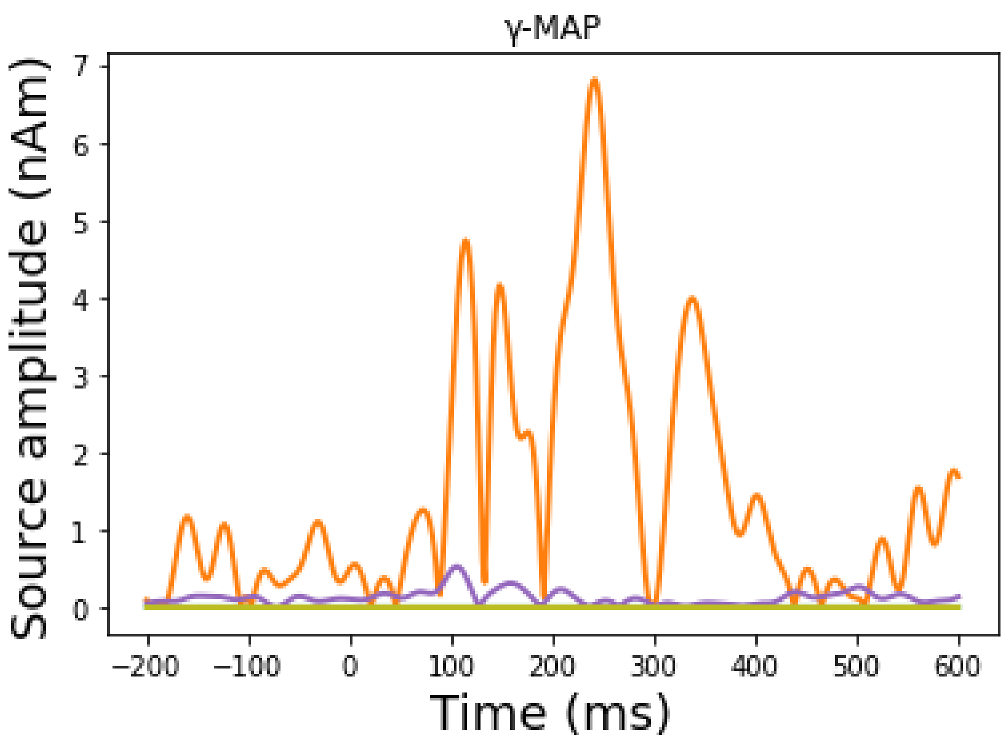
2.7.1 Source Localization 3D Visualization and Dipole Line Plots

Let's see where the dipoles are and the source amplitudes.

[30]:
```python
# make SourceEstimate object from dipole list
stc_gamma_map = make_stc_from_dipoles(dipoles, forward['src'])

# get source amplitudes
plot_sparse_source_estimates(forward['src'], stc_gamma_map, bgcolor=(1, 1, 1),
                             opacity=0.1, fig_name=" -MAP",
                             modes=['sphere'], scale_factors=[1.])

# fancy 3D plot
time_label = ' -MAP time=%0.2f ms'
clim = dict(kind='value', lims=[10e-9, 15e-9, 20e-9])
brain_gamma_map = stc_gamma_map.plot(hemi='both',
                 clim=clim, time_label=time_label, smoothing_steps=5,
                 subjects_dir=subjects_dir, initial_time=150, time_unit='ms')
brain_gamma_map.add_label("V1", hemi='lh', color="red", scalar_thresh=0.5,␣
 ↪alpha=0.6, borders=False)
brain_gamma_map.add_label("V2", hemi='lh', color="yellow", scalar_thresh=0.5,␣
 ↪alpha=0.6, borders=False)
brain_gamma_map.add_label("V1", hemi='rh', color="red", scalar_thresh=0.5,␣
 ↪alpha=0.6, borders=False)
```

```
brain_gamma_map.add_label("V2", hemi='rh', color="yellow", scalar_thresh=0.5,␣
  ↪alpha=0.6, borders=False)
```

```
Converting dipoles into a SourceEstimate.
[done]
Total number of active sources: 9
```



Interestingly, γ-MAP yields 9 active sources, but only two really seem to have any importance, with one highly dominating. While γ-MAP was able to induce much more sparsity, it is still not able to fully turn some source amplitudes to zero.

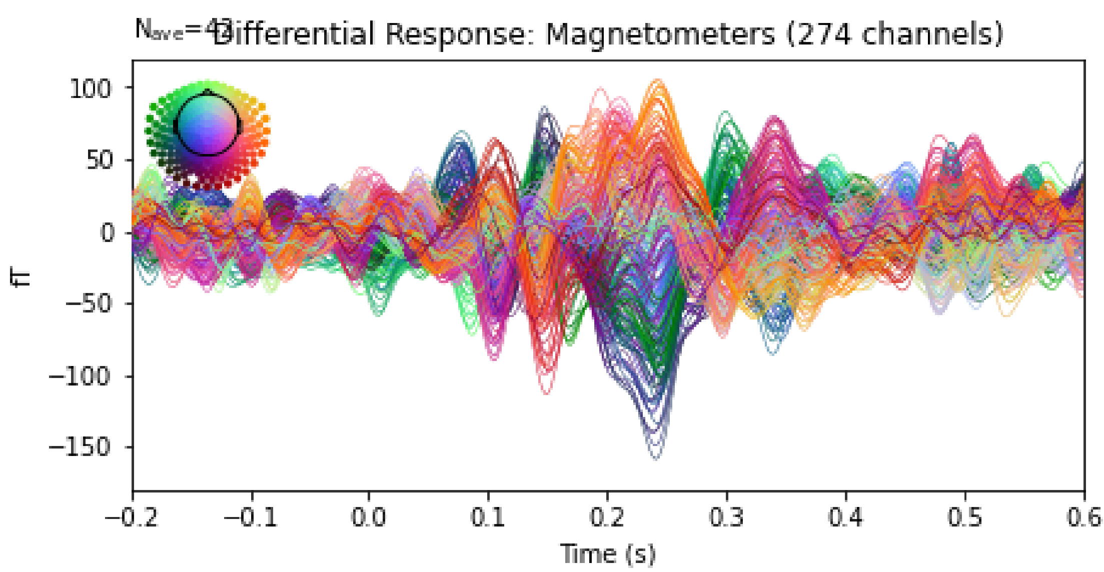
2.7.2 MEG Sensor Projection

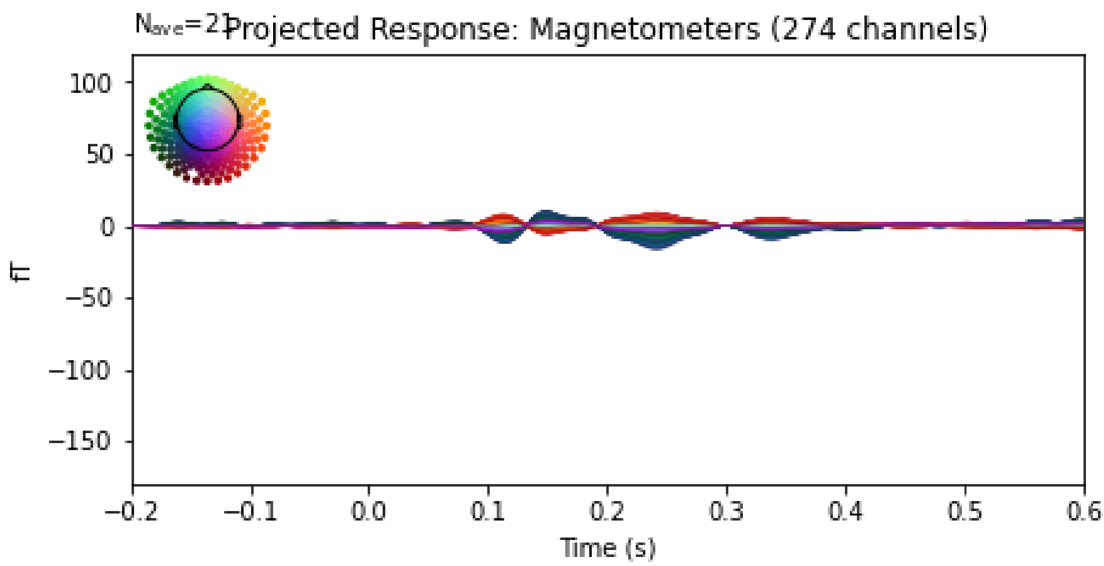
We already know that the variance explained is poor, but we can look at how the projected data appear.

[31]:
```
# get contrast (sometimes this gets overriden if we jump between cells)
contrast = combine_evoked([evoked[0], evoked[1]], weights=[-1, 1])

# filtering for MEG channels
contrast.pick_types(meg='mag', exclude=evoked[0].ch_names[0:29])

# getting the dipole projection to sensor space
explained = mne.combine_evoked([contrast, residual], weights=[1, -1])
```

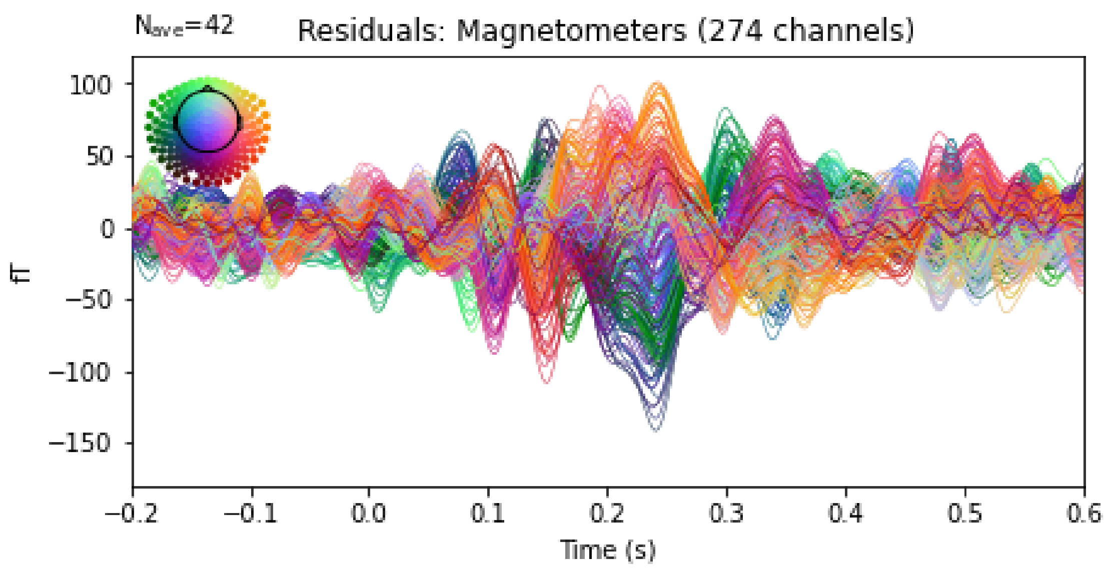
```python
# limit amplitude
ylim = dict(mag=[-180, 120])

# all contrast data
contrast.plot(titles=dict(mag='Differential Response: Magnetometers'),␣
↪ylim=ylim,
            proj=True, time_unit='s')

# all explained data
explained.plot(titles=dict(mag='Projected Response: Magnetometers'), ylim=ylim,
            proj=True, time_unit='s')

# all residuals
residual.plot(titles=dict(mag='Residuals: Magnetometers'), ylim=ylim,
              proj=True, time_unit='s')
```

```
Removing 5 compensators from info because not all compensation channels were
picked.
```



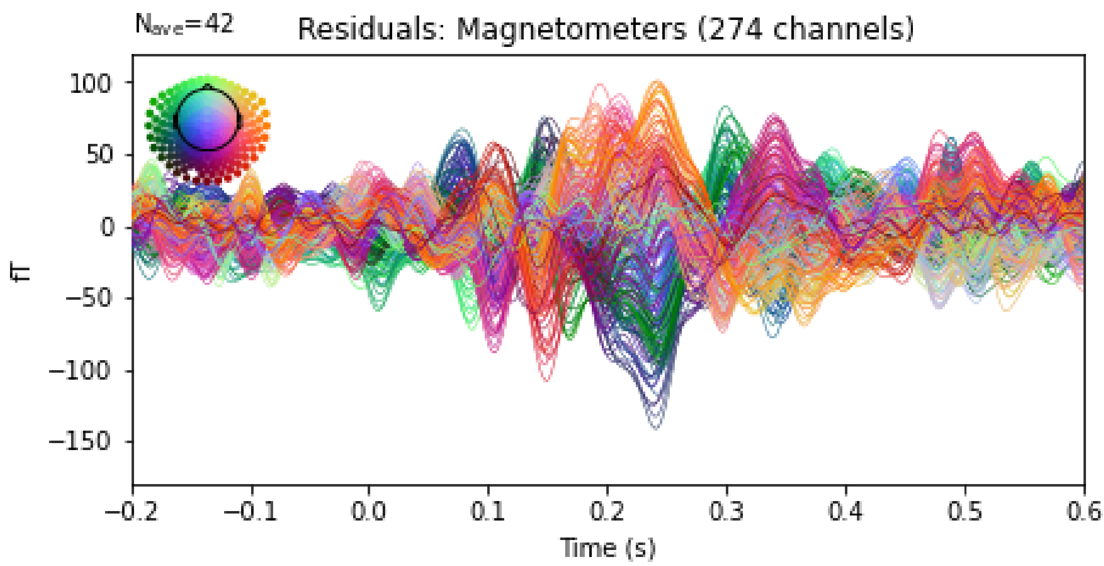


Nave=21
Projected Response: Magnetometers (274 channels)
fT
Time (s)



Nave=42
Residuals: Magnetometers (274 channels)
fT
Time (s)

[31]:



Indeed, the projected amplitudes are tiny, and the residuals plot appears similar to the original data, meaning that barely any of the data could be explained.

2.8 Inverse Solution Using TF-MxNE

At last, we return to TF-MxNE. We will use slightly larger regularization paramters here: $\alpha = 50$, L_1 ratio $= 0.05$, $\rho = 0.2$, and $\gamma = 0.9$. We will also initialize the weights using dSPM before inducing sparsity.

[32]:
```
alpha = 50.
l1_ratio = 0.05

loose, depth = 0.2, 0.9

# initialize weights
inverse_operator = make_inverse_operator(contrast.info, forward, cov,
                                         loose=loose, depth=depth)
stc_dspm = apply_inverse(contrast, inverse_operator, lambda2=1. / 9.,
                         method='dSPM')

# TF-MxNE
dipoles, residual = tf_mixed_norm(
    contrast, forward, cov, alpha=alpha, l1_ratio=l1_ratio, loose=loose,
    depth=depth, maxit=200, tol=1e-6, weights=stc_dspm, weights_min=5.,
    debias=True, wsize=16, tstep=4, window=0.05, return_as_dipoles=True,
    return_residual=True)
```

```
Converting forward solution to surface orientation
    Average patch normals will be employed in the rotation to the local surface
```

```
coordinates…
    Converting to surface-based source orientations…
    [done]
Computing inverse operator with 274 channels.
    274 out of 274 channels remain after picking
Selected 274 channels
Creating the depth weighting matrix…
    274 magnetometer or axial gradiometer channels
    limit = 8105/8196 = 10.003233
    scale = 4.43797e-11 exp = 0.9
Applying loose dipole orientations to surface source spaces: 0.2
Whitening the forward solution.
Computing rank from covariance with rank=None
    Using tolerance 4.6e-14 (2.2e-16 eps * 274 dim * 0.76  max singular value)
    Estimated rank (mag): 274
    MAG: rank 274 computed from 274 data channels with 0 projectors
    Setting small MAG eigenvalues to zero (without PCA)
Creating the source covariance matrix
Adjusting source covariance matrix.
Computing SVD of whitened and weighted lead field matrix.
    largest singular value = 4.63083
    scaling factor to adjust the trace = 6.87388e+18 (nchan = 274 nzero = 0)
Preparing the inverse operator for use…
    Scaled noise and source covariance from nave = 1 to nave = 42
    Created the regularized inverter
    The projection vectors do not apply to these channels.
    Created the whitener using a noise covariance matrix with rank 274 (0 small
eigenvalues omitted)
    Computing noise-normalization factors (dSPM)…
[done]
Applying inverse operator to "-faces + scrambled"…
    Picked 274 channels from the data
    Computing inverse…
    Eigenleads need to be weighted …
    Computing residual…
    Explained  72.1% variance
    Combining the current components…
    dSPM…
[done]
Converting forward solution to surface orientation
    Average patch normals will be employed in the rotation to the local surface
coordinates…
    Converting to surface-based source orientations…
    [done]
Computing inverse operator with 274 channels.
    274 out of 274 channels remain after picking
Selected 274 channels
Creating the depth weighting matrix…
```

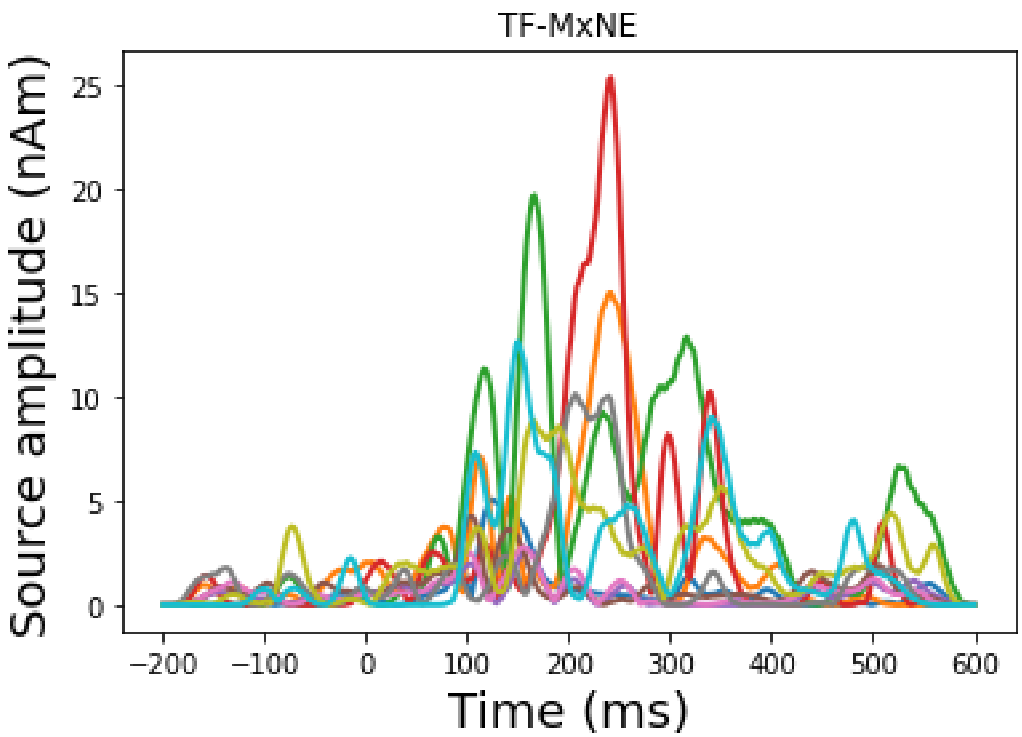
```
Applying loose dipole orientations to surface source spaces: 0.2
Whitening the forward solution.
Computing rank from covariance with rank=None
    Using tolerance 4.6e-14 (2.2e-16 eps * 274 dim * 0.76  max singular value)
    Estimated rank (mag): 274
    MAG: rank 274 computed from 274 data channels with 0 projectors
    Setting small MAG eigenvalues to zero (without PCA)
Creating the source covariance matrix
Adjusting source covariance matrix.
Reducing source space to 317 sources
Whitening data matrix.
Using block coordinate descent with active set approach

dgap 3.13e+00 :: p_obj 671.670909 :: d_obj 668.545189 :: n_active 18

    Iteration 10 :: n_active 12
    dgap 8.43e-01 :: p_obj 670.539876 :: d_obj 669.696829

    Iteration 20 :: n_active 11
    dgap 1.05e-01 :: p_obj 670.518681 :: d_obj 670.413410

    Iteration 30 :: n_active 10
    dgap 1.10e-02 :: p_obj 670.514973 :: d_obj 670.503993

    Iteration 40 :: n_active 10
    dgap 2.30e-03 :: p_obj 670.514962 :: d_obj 670.512658

    Iteration 50 :: n_active 10
    dgap 5.26e-04 :: p_obj 670.514961 :: d_obj 670.514435

dgap 5.26e-04 :: p_obj 670.514961 :: d_obj 670.514435 :: n_active 10

Convergence reached!

Debiasing did not converge after 1000 iterations! max(|D - D0| = 1.966557e-03 >=
1.000000e-06)
[done]
```

2.8.1 Source Localization 3D Visualization and Dipole Line Plots

[33]:
```
# make SourceEstimate object from dipole list
stc_tfmxne = make_stc_from_dipoles(dipoles, forward['src'])

# get source amplitudes
plot_sparse_source_estimates(forward['src'], stc_tfmxne, bgcolor=(1, 1, 1),
                             opacity=0.1, fig_name="TF-MxNE",
                             modes=['sphere'], scale_factors=[1.])
```

```
# fancy 3D plot
time_label = 'TF-MxNE time=%0.2f ms'
clim = dict(kind='value', lims=[10e-9, 15e-9, 20e-9])
brain_tfmxne = stc_tfmxne.plot(hemi='both',
                  clim=clim, time_label=time_label, smoothing_steps=5,
                  subjects_dir=subjects_dir, initial_time=150, time_unit='ms')
brain_tfmxne.add_label("V1", hemi='lh', color="red", scalar_thresh=0.5, alpha=0.
 ↪6, borders=False)
brain_tfmxne.add_label("V2", hemi='lh', color="yellow", scalar_thresh=0.5,␣
 ↪alpha=0.6, borders=False)
brain_tfmxne.add_label("V1", hemi='rh', color="red", scalar_thresh=0.5, alpha=0.
 ↪6, borders=False)
brain_tfmxne.add_label("V2", hemi='rh', color="yellow", scalar_thresh=0.5,␣
 ↪alpha=0.6, borders=False)
```

```
Converting dipoles into a SourceEstimate.
[done]
Total number of active sources: 10
```



TF-MxNE yields 10 active sources, but these all are indeed active; none hug the $y = 0$ as was seen for γ-MAP. If we combine all the amplitudes together, we can see a resemblance between the total source amplitudes from TF-MxNE and the orange curve from γ-MAP. This means that γ-MAP attributed most of the current to one particular dipole, but TF-MxNE distributed it to many moer dipoles.

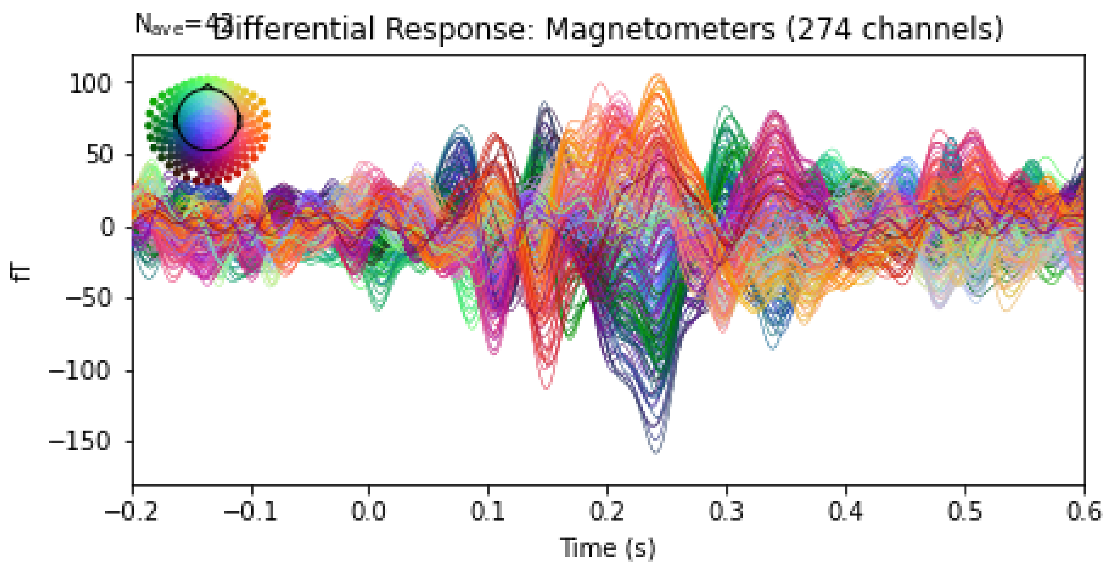
It is also worth noting that the source amplitudes are much greater for TF-MxNE than γ-MAP; the highest amplitude seen in γ-MAP is aorund 7 nAm, but it is about 25 nAm for TF-MxNE.

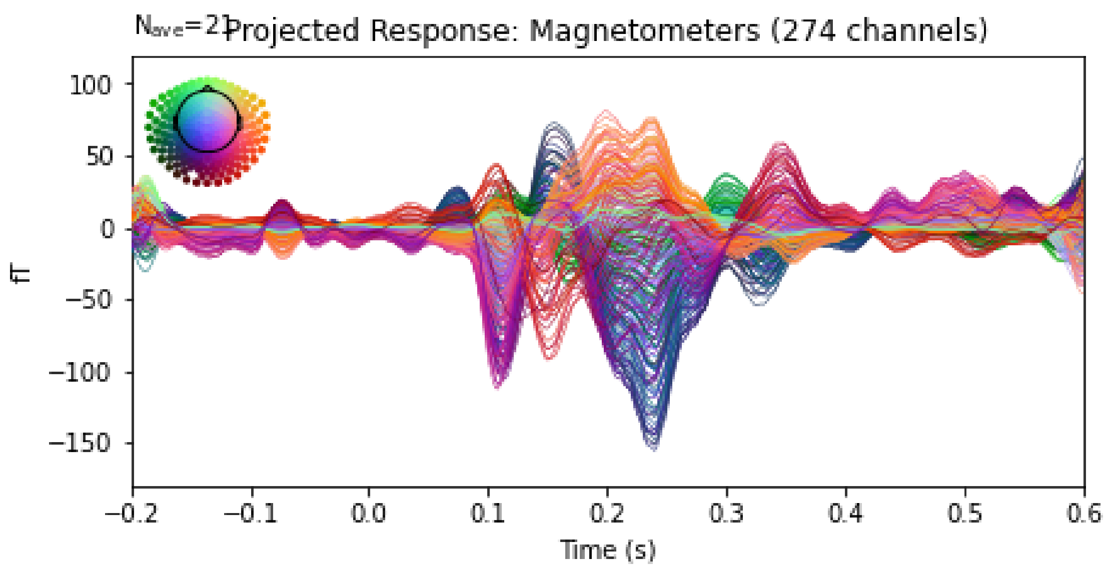
2.8.2 MEG Sensor Projection

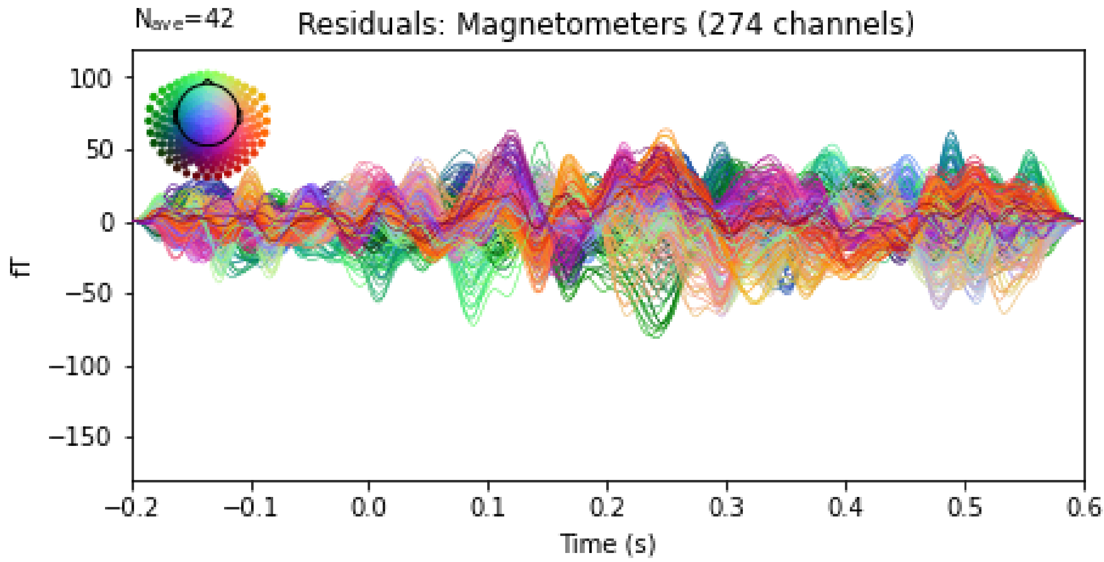
Once again, we will see how TF-MxNE fares in explaining the measured MEG data.

[34]:
```
# get contrast (sometimes this gets overriden if we jump between cells)
contrast = combine_evoked([evoked[0], evoked[1]], weights=[-1, 1])

# filtering for MEG channels
contrast.pick_types(meg='mag', exclude=evoked[0].ch_names[0:29])

# getting the dipole projection to sensor space
explained = mne.combine_evoked([contrast, residual], weights=[1, -1])

# limit amplitude
ylim = dict(mag=[-180, 120])

# all contrast data
contrast.plot(titles=dict(mag='Differential Response: Magnetometers'),␣
 ↪ylim=ylim,
            proj=True, time_unit='s')

# all explained data
explained.plot(titles=dict(mag='Projected Response: Magnetometers'), ylim=ylim,
            proj=True, time_unit='s')

# all residuals
residual.plot(titles=dict(mag='Residuals: Magnetometers'), ylim=ylim,
              proj=True, time_unit='s')
```

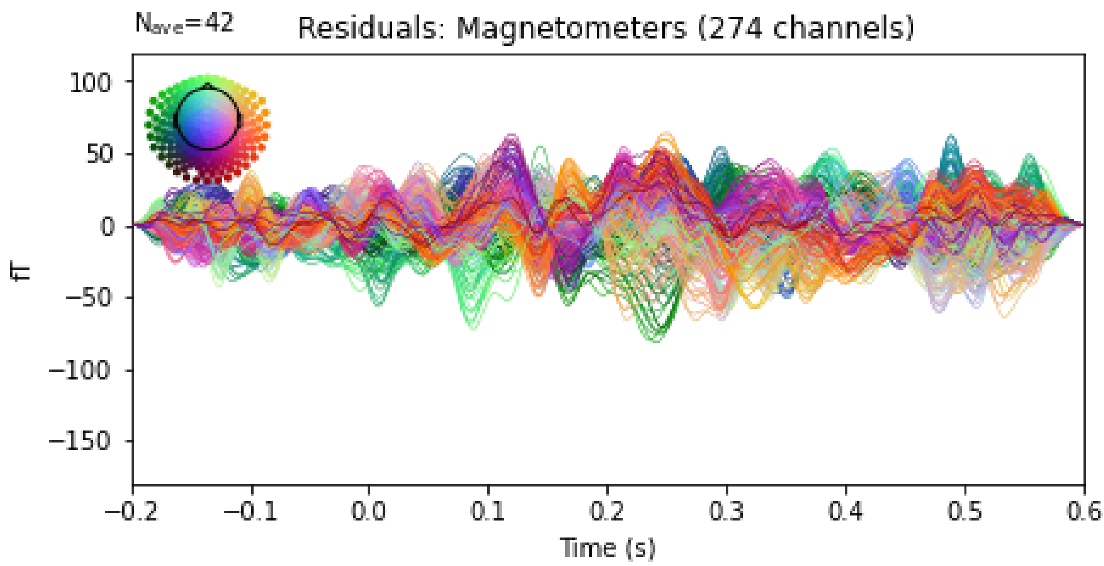
```
Removing 5 compensators from info because not all compensation channels were
picked.
```


Nave=43
Differential Response: Magnetometers (274 channels)
100
50
0
−50
−100
−150
fT
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Time (s)


Nave=21
Projected Response: Magnetometers (274 channels)
100
50
0
−50
−100
−150
fT
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Time (s)



[34]:



While certainly not perfect, the TF-MxNE appears to be able to explain much more of the variance.