intermagnet_temporal_analysis

January 19, 2023

[ ]:
```
from os import getcwd
from os.path import dirname, join

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy import interpolate

import MAMMAL.Diurnal as Diurnal
from MAMMAL.Parse import parseIM as pi
from MAMMAL.Utils import ProcessingUtils as pu


%matplotlib inline
plt.rcParams["figure.figsize"] = (15, 10) # (w, h)

SRC_DIR      = getcwd()
ANALYSIS_DIR = join(SRC_DIR, 'analysis_data')
TEST_DIR     = join(SRC_DIR, 'test_data')
```

1 Load INTERMAGNET Data

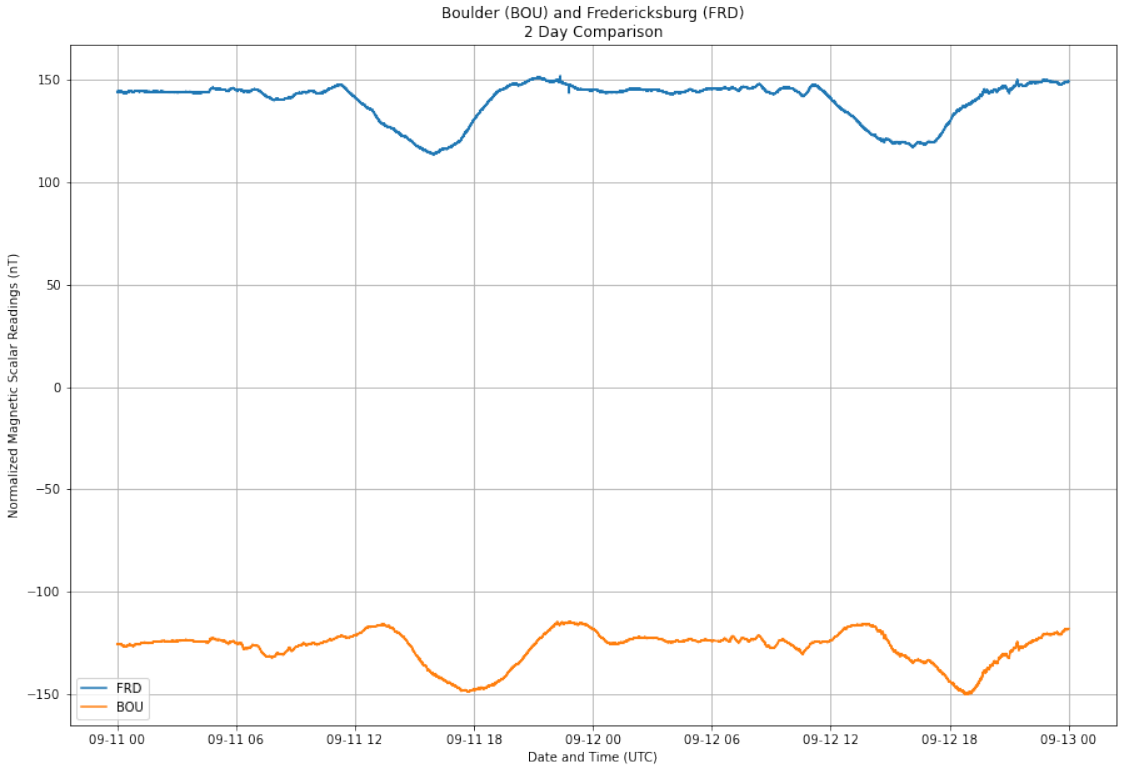
[ ]:
```
df_dict = pi.loadInterMagData(ANALYSIS_DIR)

from_name = 'BOU'
to_name   = 'FRD'

from_df = df_dict[from_name]
to_df   = df_dict[to_name]

to_df = pu.reject_outliers(to_df,
                           window_size=250,
                           std_lim=3)

to_IGRF_f   = np.array(to_df.IGRF_F)[0]
from_IGRF_f = np.array(from_df.IGRF_F)[0]
```

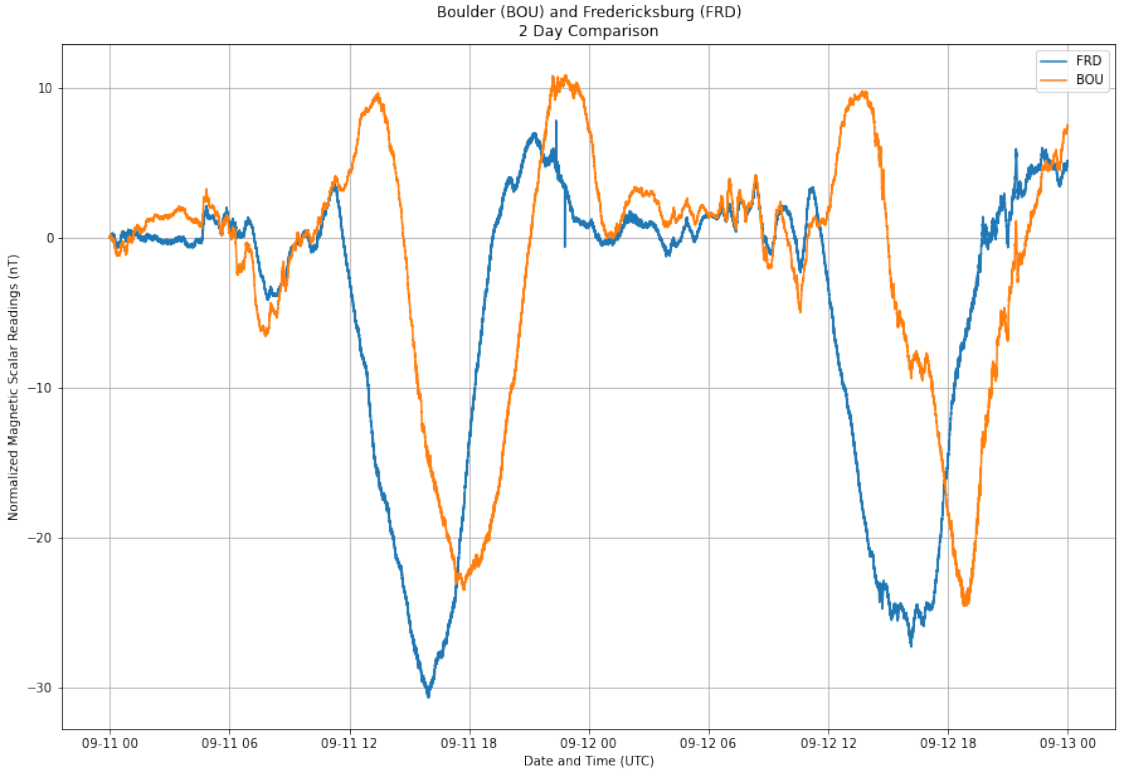
```
to_f   = np.array(to_df.F)
from_f = np.array(from_df.F)

to_f_no_core   = np.array(to_df.F) - to_IGRF_f
from_f_no_core = np.array(from_df.F) - from_IGRF_f

to_t   = np.array(to_df.epoch_sec)
from_t = np.array(from_df.epoch_sec)

plt.figure()
plt.title('Boulder (BOU) and Fredericksburg (FRD)\n2 Day Comparison')
plt.xlabel('Date and Time (UTC)')
plt.ylabel('Normalized Magnetic Scalar Readings (nT)')
plt.plot(to_df.datetime, to_f_no_core, label=to_name)
plt.plot(from_df.datetime, from_f_no_core, label=from_name)
plt.legend()
plt.grid()

plt.figure()
plt.title('Boulder (BOU) and Fredericksburg (FRD)\n2 Day Comparison')
plt.xlabel('Date and Time (UTC)')
plt.ylabel('Normalized Magnetic Scalar Readings (nT)')
plt.plot(to_df.datetime, to_f_no_core - to_f_no_core[0], label=to_name)
plt.plot(from_df.datetime, from_f_no_core - from_f_no_core[0], label=from_name)
plt.legend()
plt.grid()

print('RMSE:', pu.rmse(to_f_no_core, from_f_no_core[:len(to_f)])) # Extra␣
 ↪slicing needed because dimensions don't match as a result of rejecting␣
 ↪outliers
print('RMSE:', pu.rmse(to_f_no_core - to_f_no_core[0], from_f_no_core[:
 ↪len(to_f)] - from_f_no_core[0])) # Extra slicing needed because dimensions␣
 ↪don't match as a result of rejecting outliers
```

```
Loaded bou20190911psec.sec
Loaded bou20190912psec.sec
Loaded frd20190911psec.sec
Loaded frd20190912psec.sec
```

```
  0%|          | 0/692 [00:00<?, ?it/s]C:\Users\ltber\Anaconda3\Lib\site-
packages\pandas\core\dtypes\cast.py:2221: RuntimeWarning: invalid value
encountered in cast
  casted = element.astype(dtype)
100%|     | 692/692 [00:01<00:00, 584.82it/s]
```

```
RMSE: 267.3244711396219
RMSE: 10.869259973212195
```

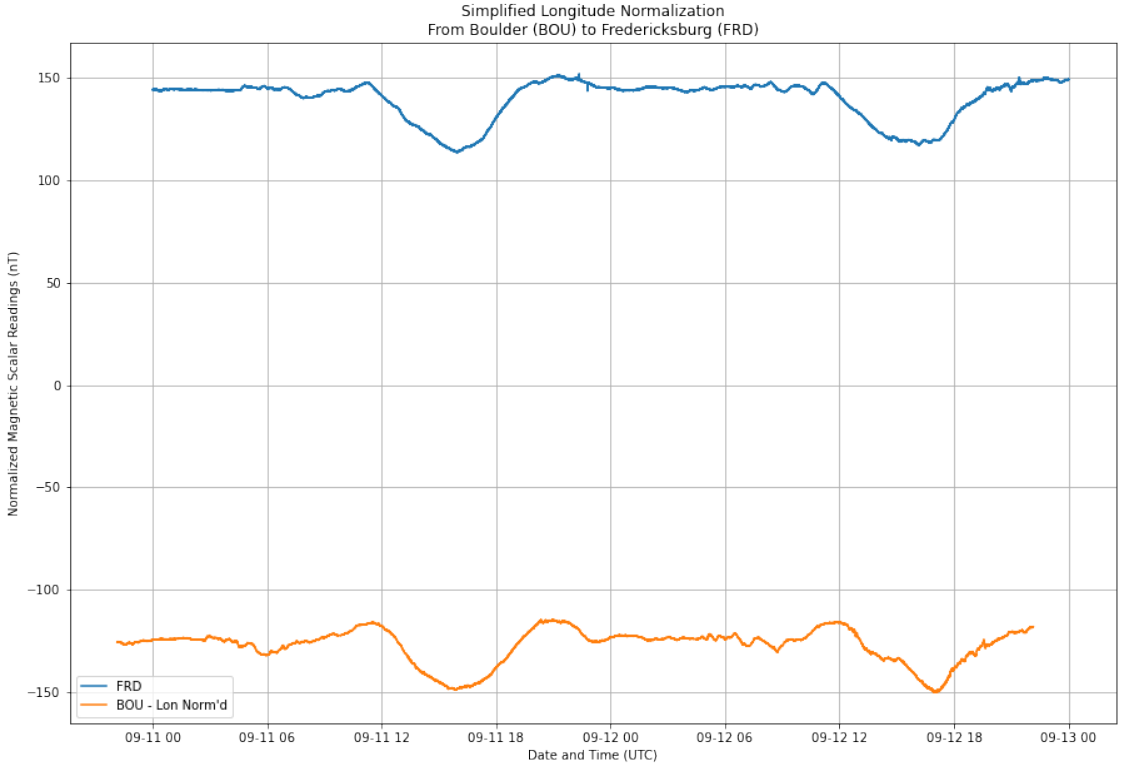

Boulder (BOU) and Fredericksburg (FRD)
2 Day Comparison
Normalized Magnetic Scalar Readings (nT)
150
100
50
0
−50
−100
−150
FRD
BOU
09-11 00
09-11 06
09-11 12
09-11 18
09-12 00
09-12 06
09-12 12
09-12 18
09-13 00
Date and Time (UTC)


Boulder (BOU) and Fredericksburg (FRD)
2 Day Comparison
Normalized Magnetic Scalar Readings (nT)
10
0
−10
−20
−30
FRD
BOU
09-11 00
09-11 06
09-11 12
09-11 18
09-12 00
09-12 06
09-12 12
09-12 18
09-13 00
Date and Time (UTC)

2 Simple Longitude Normalization

[ ]:
```
lon_diff     = from_df.LONG.mean() - to_df.LONG.mean()
lon_t_offset = pd.Timedelta(seconds=lon_diff / Diurnal.E_ROT_DEG_S)
bou_shift_t  = from_df.epoch_sec + lon_t_offset.total_seconds()

plt.figure()
plt.title('Simplified Longitude Normalization\nFrom Boulder (BOU) to␣
 ↪Fredericksburg (FRD)')
plt.xlabel('Date and Time (UTC)')
plt.ylabel('Normalized Magnetic Scalar Readings (nT)')
plt.plot(to_df.datetime, to_f_no_core, label=to_name)
plt.plot(from_df.datetime + lon_t_offset, from_f_no_core, label='{} - Lon␣
 ↪Norm\'d'.format(from_name))
plt.legend()
plt.grid()

interp_lpf = interpolate.interp1d(bou_shift_t, from_f_no_core, 'cubic')

interp_mask = np.logical_and(to_t >= bou_shift_t.min(), to_t <= bou_shift_t.
 ↪max())
interp_t    = to_t[interp_mask] # Clip interpolation times

bou_shift_interp = interp_lpf(interp_t)

print('RMSE:', pu.rmse(to_f_no_core[interp_mask],
                       bou_shift_interp))
```

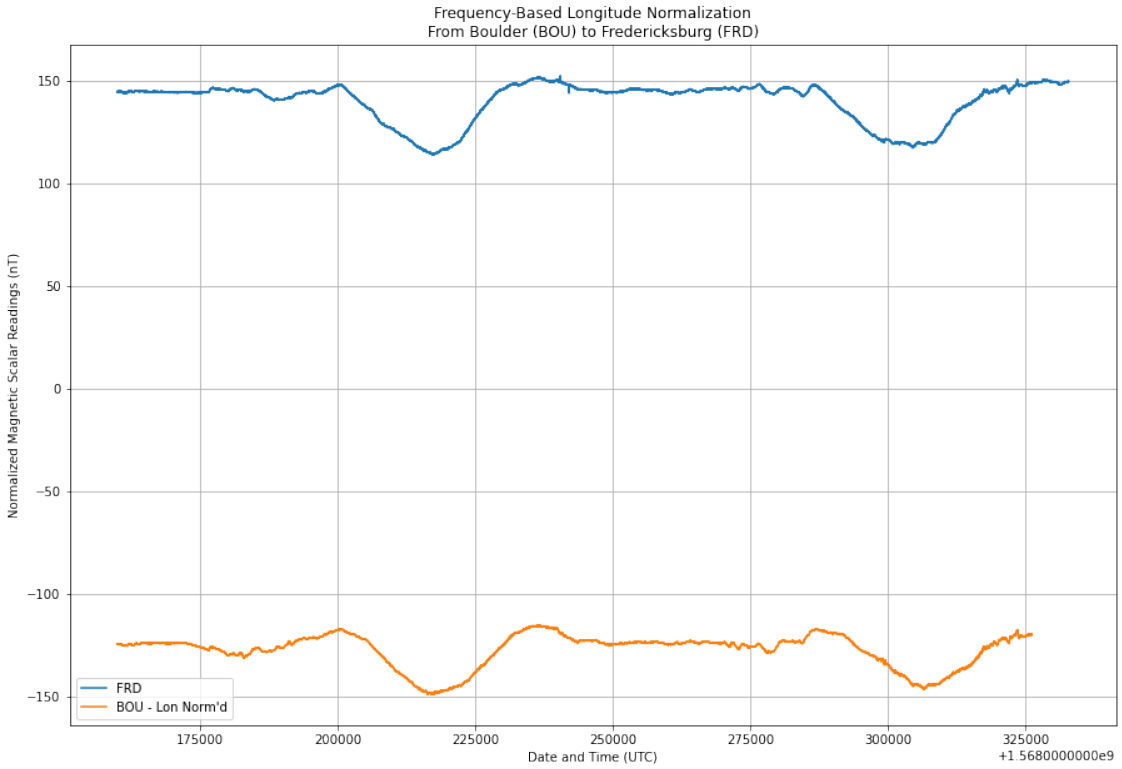
```
RMSE: 266.84286256867483
```



3 Frequency-Based Longitude Normalization

[ ]:
```
from_combined_t, from_combined_f, _, _ , _ = Diurnal.longitude_norm(from_df,␣
 ↪to_df.LONG.mean())
from_combined_f_no_core = from_combined_f - from_IGRF_f

plt.figure()
plt.title('Frequency-Based Longitude Normalization\nFrom Boulder (BOU) to␣
 ↪Fredericksburg (FRD)')
plt.xlabel('Date and Time (UTC)')
plt.ylabel('Normalized Magnetic Scalar Readings (nT)')
plt.plot(to_df.epoch_sec, to_f_no_core, label=to_name)
plt.plot(from_combined_t, from_combined_f_no_core, label='{} - Lon Norm\'d'.
 ↪format(from_name))
plt.legend()
plt.grid()

print('RMSE:', pu.rmse(to_f_no_core[interp_mask],
                       from_combined_f_no_core[:
 ↪len(to_f_no_core[interp_mask])]))
```

```
RMSE: 266.87837777008167
```

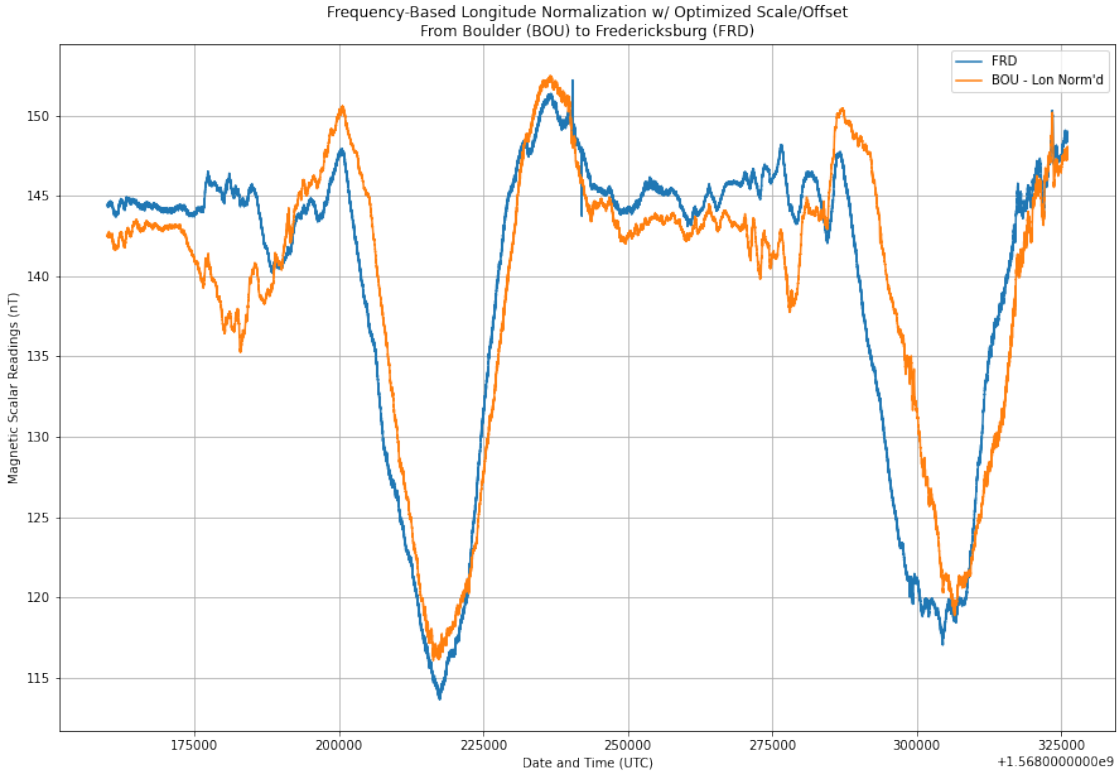


4 Find Optimal Scale and Offset Parameters

[ ]:
```
interp_combined = interpolate.interp1d(from_combined_t,␣
 ↪from_combined_f_no_core, 'cubic')

interp_mask = np.logical_and(to_t >= from_combined_t.min(), to_t <=␣
 ↪from_combined_t.max())
interp_t    = to_t[interp_mask] # Clip interpolation times

from_combined_interp = interp_combined(interp_t)

offset, scale       = Diurnal.calibrate([0, 1], from_combined_interp,␣
 ↪to_f_no_core[interp_mask])
from_combined_opt_f = Diurnal.apply_cal([offset, scale],␣
 ↪from_combined_f_no_core)

print('Optimal scale:',  scale)
print('Optimal offset:', offset)

plt.figure()
```

```
plt.title('Frequency-Based Longitude Normalization w/ Optimized Scale/
  ↪Offset\nFrom Boulder (BOU) to Fredericksburg (FRD)')
plt.xlabel('Date and Time (UTC)')
plt.ylabel('Magnetic Scalar Readings (nT)')
plt.plot(to_df.epoch_sec[interp_mask], to_f_no_core[interp_mask], label=to_name)
plt.plot(from_combined_t, from_combined_opt_f, label='{} - Lon Norm\'d'.
  ↪format(from_name))
plt.legend()
plt.grid()

print('RMSE:', pu.rmse(to_f_no_core[interp_mask],
                       from_combined_opt_f[:len(to_f_no_core[interp_mask])]))
print('STD:', (to_f_no_core[interp_mask] -
                from_combined_opt_f[:len(to_f_no_core[interp_mask])]).std())
```

```
Optimal scale: 1.085134182785885
Optimal offset: 277.6792162949522
RMSE: 4.630734469499371
STD: 4.630728729444269
```



5 Test Optimal Parameters for Next Day's Data

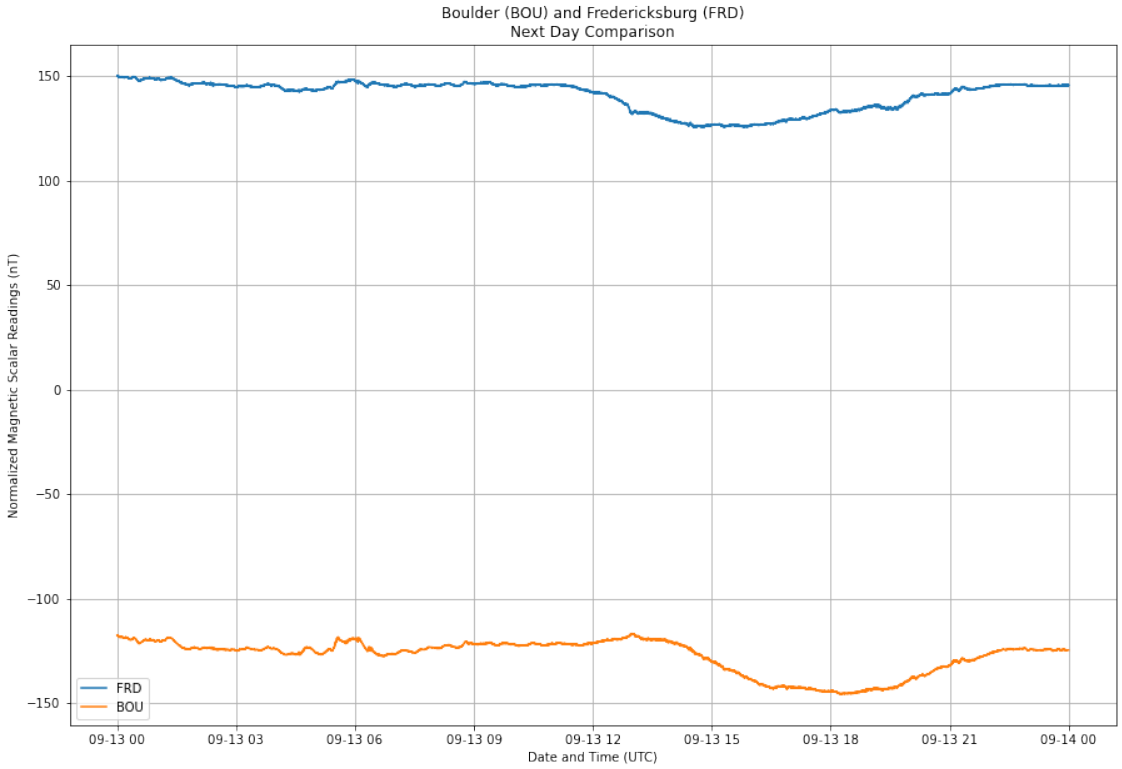
[ ]:
```
df_dict = pi.loadInterMagData(TEST_DIR)

from_name = 'BOU'
to_name   = 'FRD'

from_df = df_dict[from_name]
to_df   = df_dict[to_name]

to_df = pu.reject_outliers(to_df,
                           window_size=250,
                           std_lim=3)

to_IGRF_f   = np.array(to_df.IGRF_F)[0]
from_IGRF_f = np.array(from_df.IGRF_F)[0]

to_f   = np.array(to_df.F)
from_f = np.array(from_df.F)

to_f_no_core   = np.array(to_df.F) - to_IGRF_f
from_f_no_core = np.array(from_df.F) - from_IGRF_f

to_t   = np.array(to_df.epoch_sec)
from_t = np.array(from_df.epoch_sec)


plt.figure()
plt.title('Boulder (BOU) and Fredericksburg (FRD)\nNext Day Comparison')
plt.xlabel('Date and Time (UTC)')
plt.ylabel('Normalized Magnetic Scalar Readings (nT)')
plt.plot(to_df.datetime, to_f_no_core, label=to_name)
plt.plot(from_df.datetime, from_f_no_core, label=from_name)
plt.legend()
plt.grid()


from_combined_t, from_combined_f, _, _ , _ = Diurnal.longitude_norm(from_df,␣
 ↪to_df.LONG.mean())
from_combined_f_no_core = from_combined_f - from_IGRF_f

interp_combined = interpolate.interp1d(from_combined_t,␣
 ↪from_combined_f_no_core, 'cubic')

interp_mask = np.logical_and(to_t >= from_combined_t.min(), to_t <=␣
 ↪from_combined_t.max())
interp_t    = to_t[interp_mask] # Clip interpolation times
```

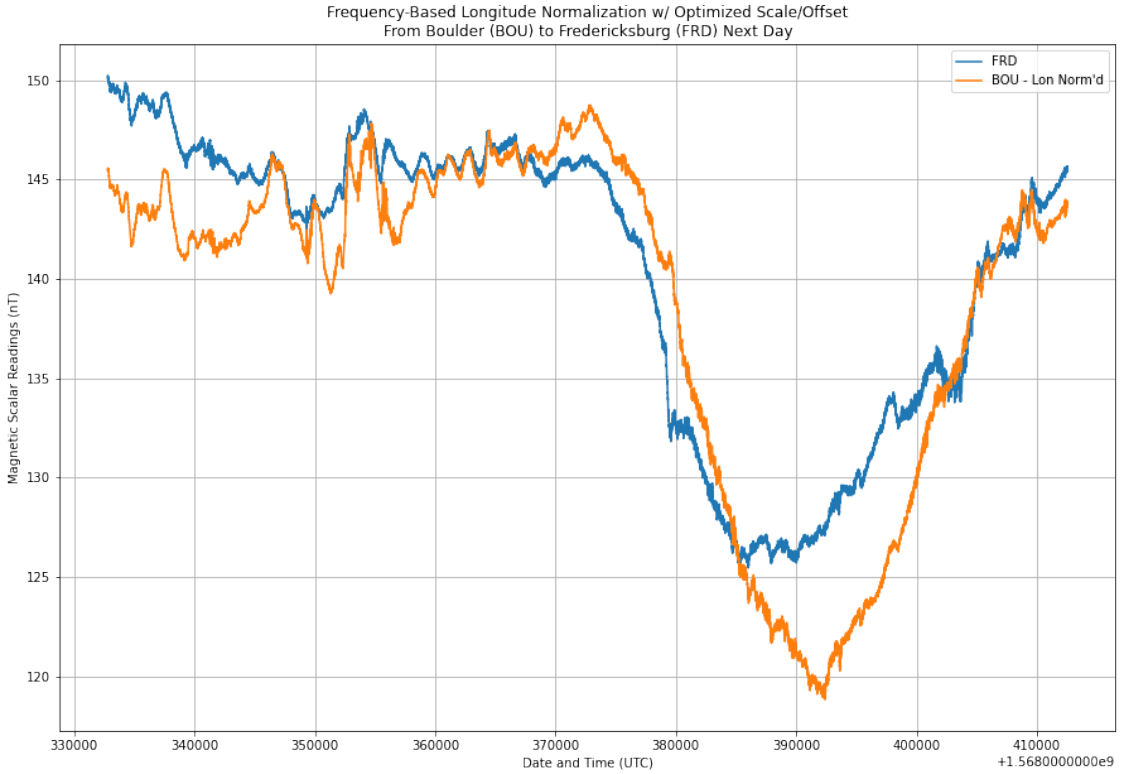
```
from_combined_interp = interp_combined(interp_t)

from_combined_opt_f = Diurnal.apply_cal([offset, scale],␣
 ↪from_combined_f_no_core)


plt.figure()
plt.title('Frequency-Based Longitude Normalization w/ Optimized Scale/
 ↪Offset\nFrom Boulder (BOU) to Fredericksburg (FRD) Next Day')
plt.xlabel('Date and Time (UTC)')
plt.ylabel('Magnetic Scalar Readings (nT)')
plt.plot(to_df.epoch_sec[interp_mask], to_f_no_core[interp_mask], label=to_name)
plt.plot(from_combined_t, from_combined_opt_f, label='{} - Lon Norm\'d'.
 ↪format(from_name))
plt.legend()
plt.grid()

print('RMSE:', pu.rmse(to_f_no_core, from_f_no_core[:len(to_f)])) # Extra␣
 ↪slicing needed because dimensions don't match as a result of rejecting␣
 ↪outliers
print('RMSE:', pu.rmse(to_f_no_core[interp_mask],
                       from_combined_opt_f[:len(to_f_no_core[interp_mask])]))
print('STD:', (to_f_no_core[interp_mask] -
               from_combined_opt_f[:len(to_f_no_core[interp_mask])]).std())
```

```
Loaded bou20190913psec.sec
Loaded frd20190913psec.sec
Loaded frn20190913psec.sec
```

```
  0%|          | 0/346 [00:00<?, ?it/s]C:\Users\ltber\Anaconda3\Lib\site-
packages\pandas\core\dtypes\cast.py:2221: RuntimeWarning: invalid value
encountered in cast
  casted = element.astype(dtype)
100%|     | 346/346 [00:00<00:00, 584.44it/s]
```

```
RMSE: 268.2980777028462
RMSE: 3.635806103790872
STD: 3.2382384905629134
```


Boulder (BOU) and Fredericksburg (FRD)
Next Day Comparison
Normalized Magnetic Scalar Readings (nT)
150
100
50
0
−50
−100
−150
FRD
BOU
09-13 00
09-13 03
09-13 06
09-13 09
09-13 12
09-13 15
09-13 18
09-13 21
09-14 00
Date and Time (UTC)


Frequency-Based Longitude Normalization w/ Optimized Scale/Offset
From Boulder (BOU) to Fredericksburg (FRD) Next Day
Magnetic Scalar Readings (nT)
150
145
140
135
130
125
120
FRD
BOU - Lon Norm'd
330000
340000
350000
360000
370000
380000
390000
400000
410000
Date and Time (UTC)
+1.5680000000e9