Tyendinaga Data

Notes:

- data exported from 'Student' computer
- using data with filename Tyendinaga1_001.asc as main data source to make plots with
- Tyendinaga1_001.asc data was recorded on Oct. 26, 2025 from 18:07:12 to 18:27:12

```
Version Notes:
pandas: 2.2.2
numpy: 1.26.4
scipy: 1.16.3
```

DataFrame --> File Name Legend:

- df --> Tyendinaga_001.asc
- df2 --> Tyendinaga2_001.asc
- df3 --> Tyendinaga3_001.asc
- df4 --> Tyendinaga4_001.asc
- df5 --> Tyendinaga5_001.asc

Check for null values:

- will show "False" if no null values

```
df:  False
df2: False
df3: False
df4: False
df5: False
```

Columns of Interest:

- NS: North-South component of motion
- EW: East-West component of motion
- Z: Vertical (Z-axis) component of motion

Other Notes:

- Sampling rate for all files: 1024 Hz
- See below for preview of the data:

dataframe: df

	NS	EW	Z	nsL	ewL	zL	aY	aX	aZ	time (s)
0	273.0	269.0	862.0	8.0	8.0	26.0	125.0	244.0	10.0	0.000000
1	276.0	274.0	864.0	8.0	8.0	27.0	127.0	247.0	9.0	0.000977
2	273.0	276.0	864.0	8.0	8.0	27.0	127.0	243.0	12.0	0.001953
3	271.0	274.0	860.0	8.0	8.0	26.0	127.0	246.0	3.0	0.002930
4	273.0	270.0	854.0	8.0	8.0	26.0	128.0	247.0	7.0	0.003906

dataframe: df2

	NS	EW	Z	nsL	ewL	zL	aY	aX	aZ	time (s)
0	271.0	270.0	852.0	8.0	8.0	26.0	224.0	169.0	23.0	0.000000
1	269.0	286.0	868.0	8.0	8.0	27.0	224.0	172.0	17.0	0.000977
2	268.0	307.0	869.0	8.0	9.0	27.0	224.0	173.0	18.0	0.001953
3	266.0	320.0	861.0	8.0	10.0	26.0	224.0	173.0	21.0	0.002930
4	263.0	323.0	856.0	8.0	10.0	26.0	223.0	172.0	18.0	0.003906

dataframe: df3

	NS	EW	Z	nsL	ewL	zL	aY	aX	aZ	time (s)
0	264.0	273.0	845.0	8.0	8.0	26.0	192.0	210.0	19.0	0.000000
1	258.0	270.0	842.0	8.0	8.0	26.0	192.0	206.0	21.0	0.000977
2	259.0	269.0	838.0	8.0	8.0	26.0	192.0	198.0	18.0	0.001953
3	256.0	266.0	834.0	8.0	8.0	26.0	192.0	196.0	16.0	0.002930
4	253.0	260.0	833.0	7.0	8.0	26.0	192.0	204.0	19.0	0.003906

dataframe: df4

	NS	EW	Z	nsL	ewL	zL	aY	aX	aZ	time (s)
0	245.0	256.0	847.0	7.0	8.0	26.0	118.0	172.0	-8.0	0.000000
1	243.0	256.0	840.0	7.0	8.0	26.0	118.0	171.0	-8.0	0.000977
2	243.0	260.0	840.0	7.0	8.0	26.0	115.0	171.0	-14.0	0.001953
3	244.0	264.0	840.0	7.0	8.0	26.0	115.0	168.0	-13.0	0.002930
4	248.0	265.0	843.0	7.0	8.0	26.0	114.0	169.0	-15.0	0.003906

dataframe: df5

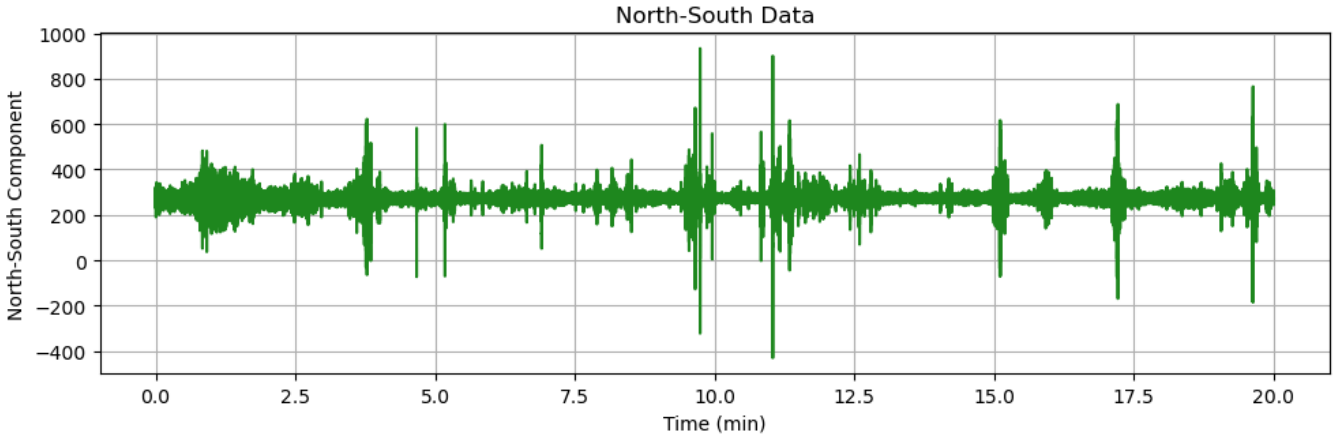
	NS	EW	Z	nsL	ewL	zL	aY	aX	aZ	time (s)
0	283.0	267.0	874.0	8.0	8.0	27.0	127.0	128.0	8.0	0.000000
1	281.0	269.0	876.0	8.0	8.0	27.0	127.0	128.0	3.0	0.000977
2	282.0	272.0	870.0	8.0	8.0	27.0	126.0	128.0	6.0	0.001953
3	280.0	278.0	859.0	8.0	8.0	26.0	126.0	128.0	7.0	0.002930
4	275.0	281.0	851.0	8.0	8.0	26.0	126.0	128.0	6.0	0.003906

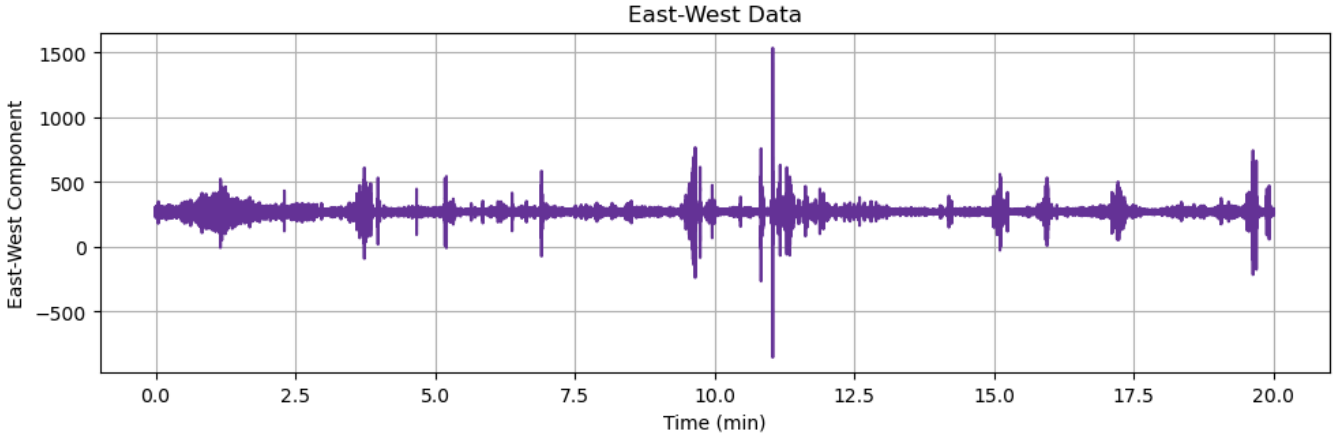
Plotting the Time Series

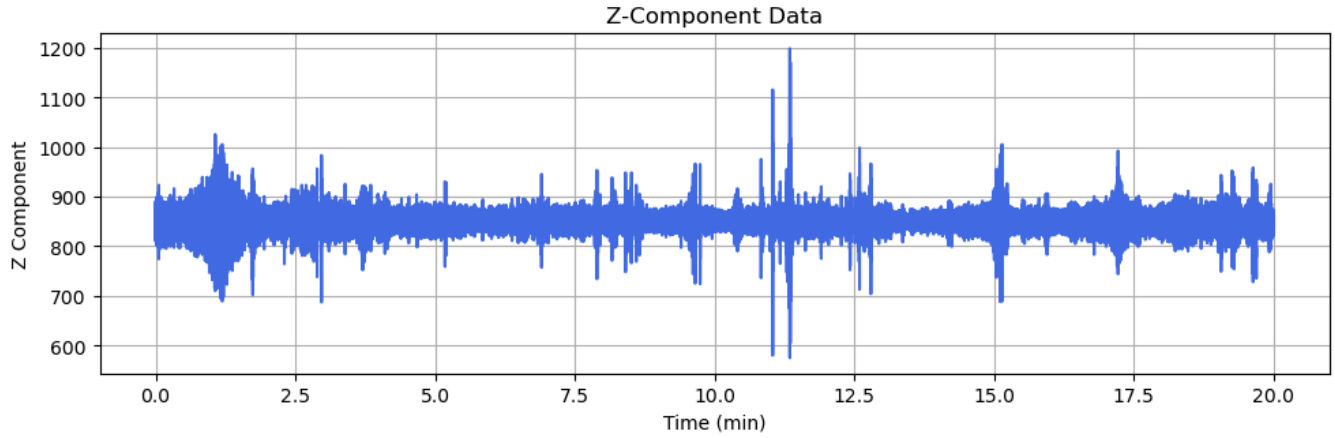
Notes:

- Plot the NS, EW, and Z data separately

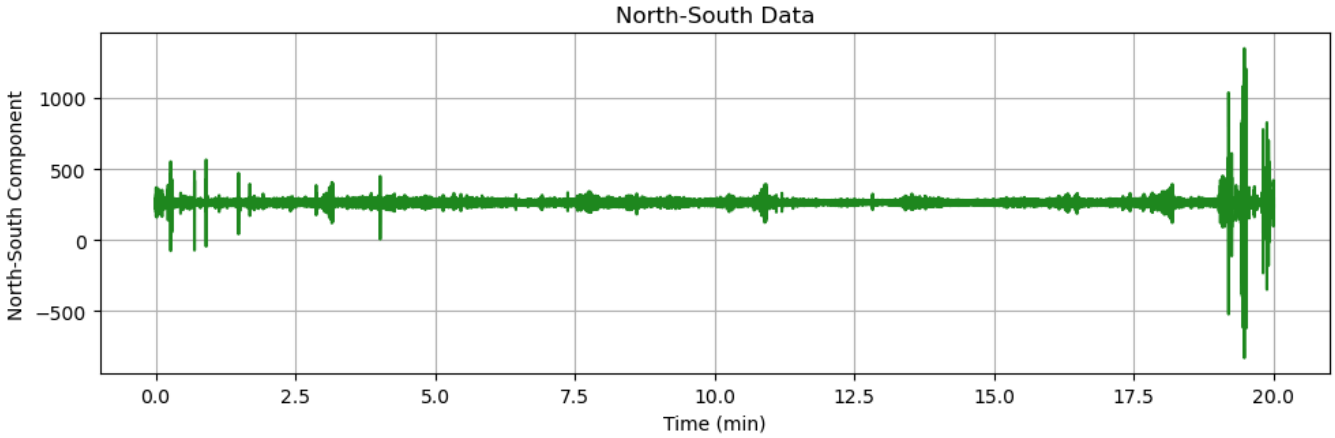
dataframe: df

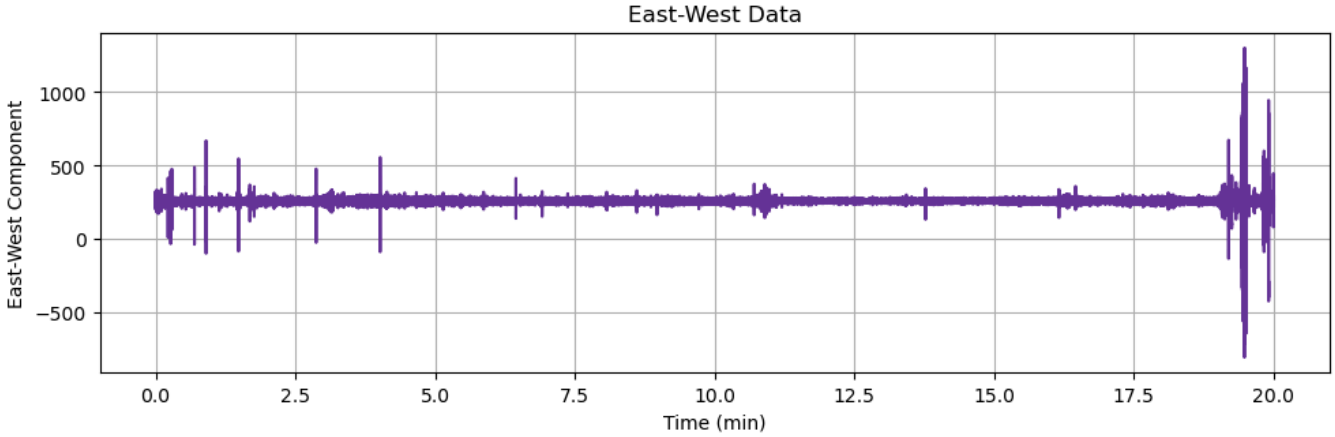


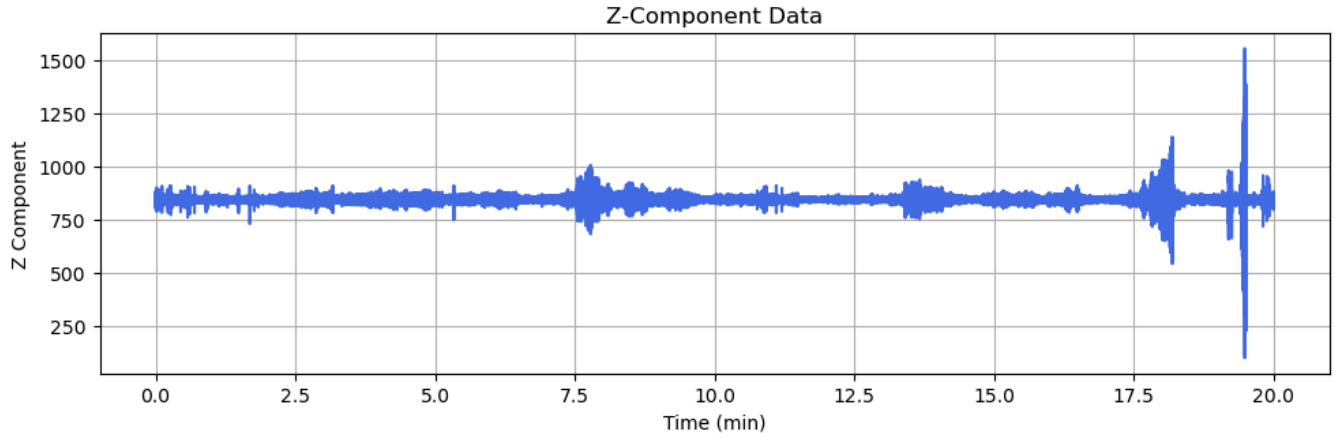




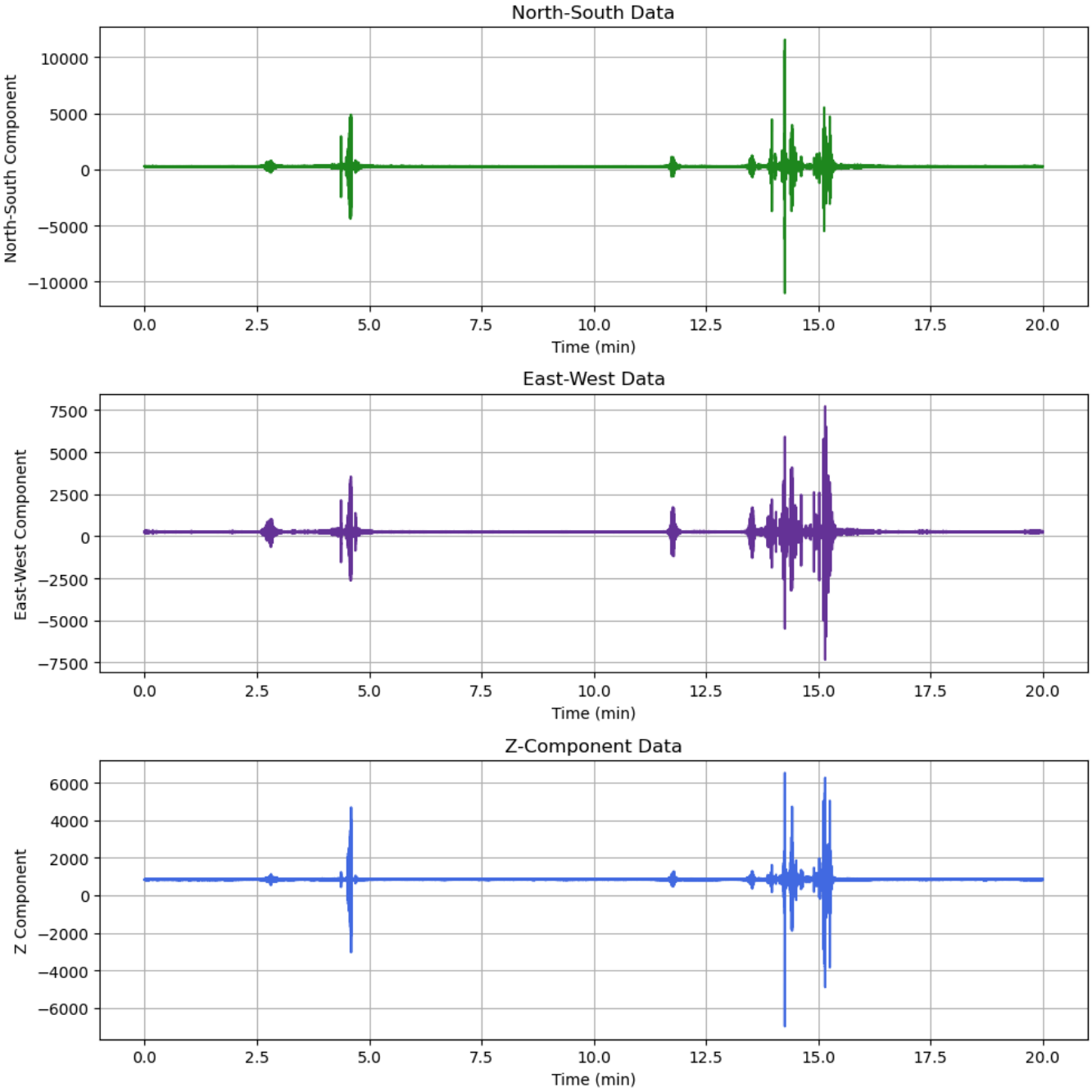
dataframe: df2



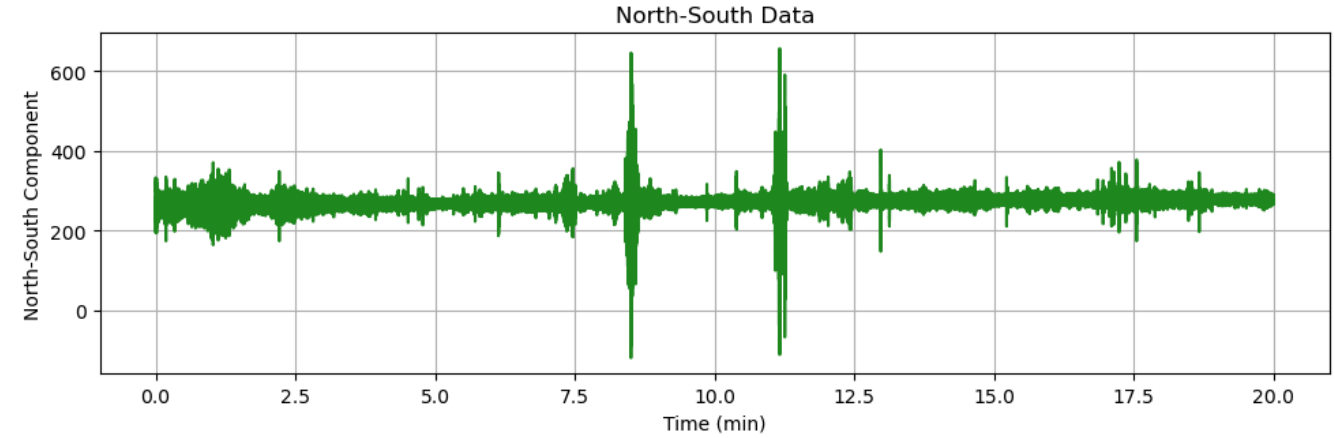


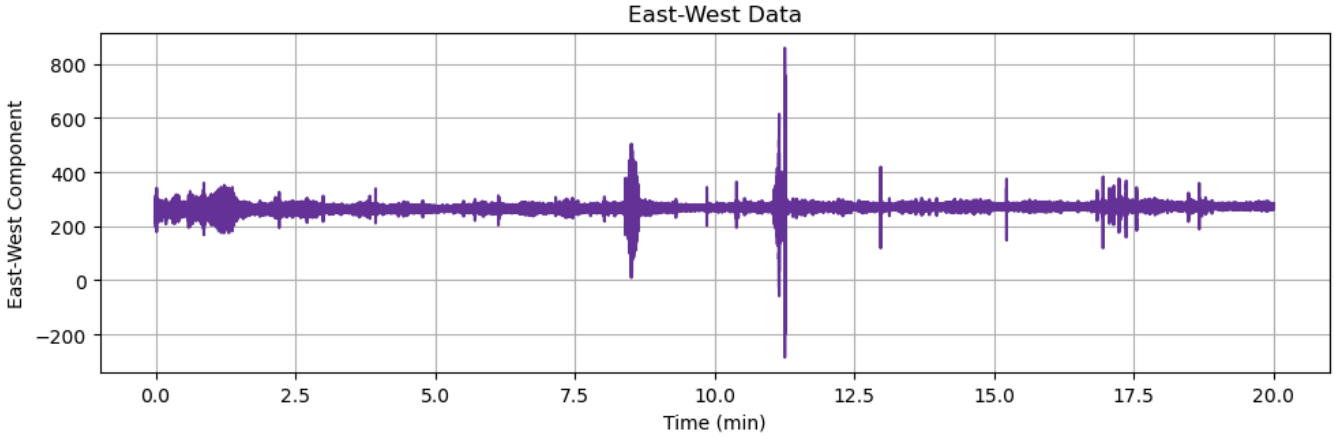


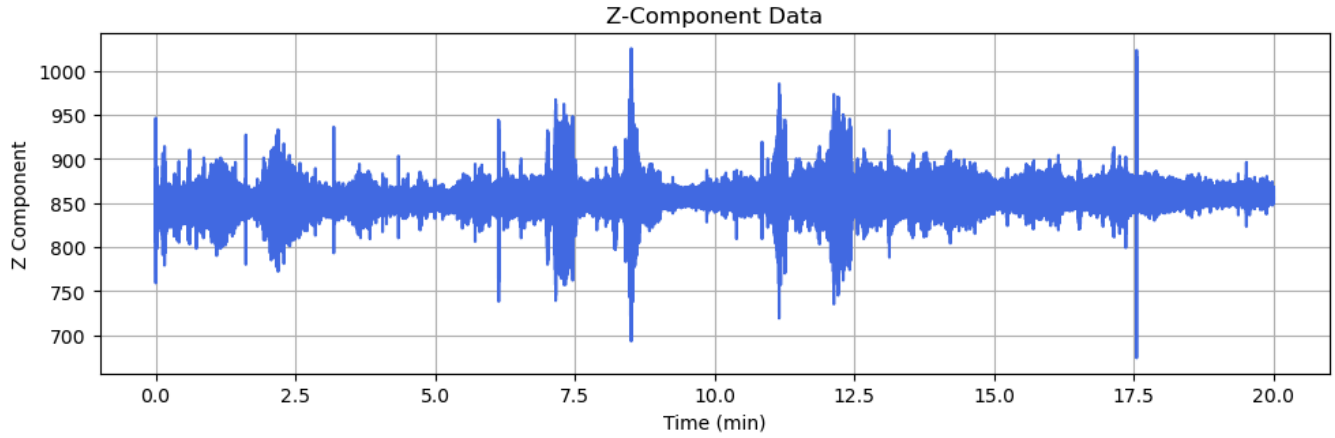
dataframe: df3



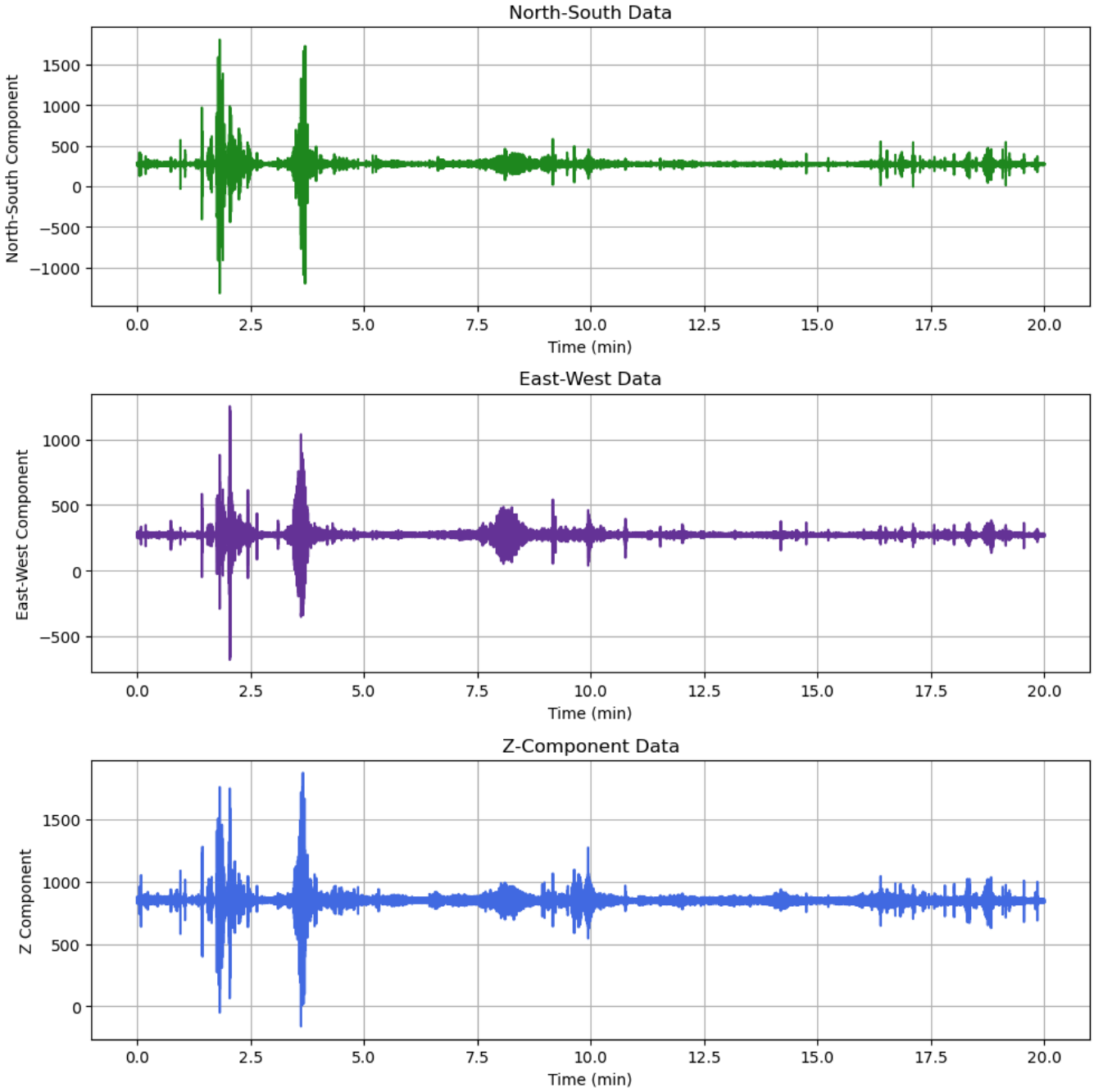
dataframe: df4







dataframe: df5

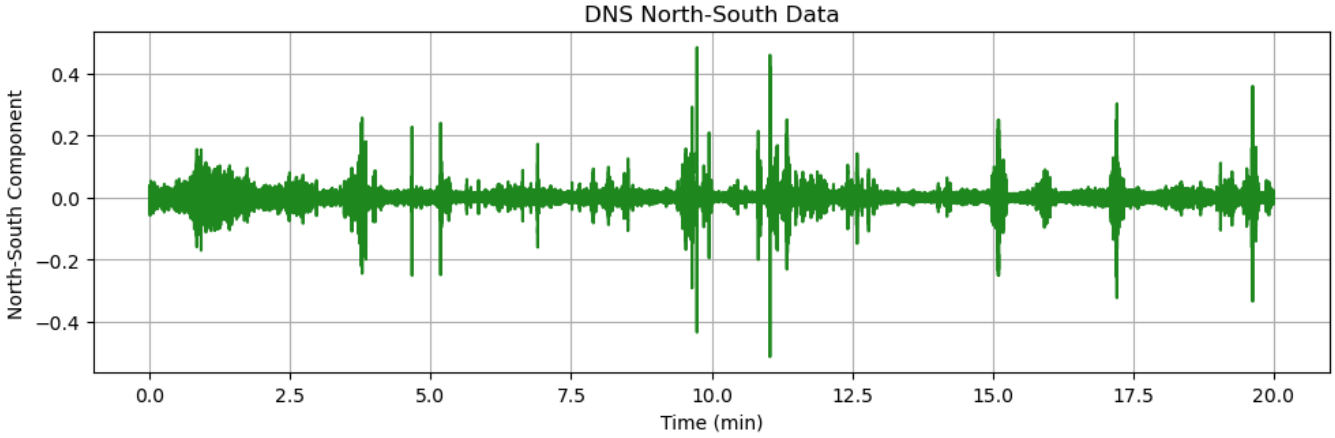


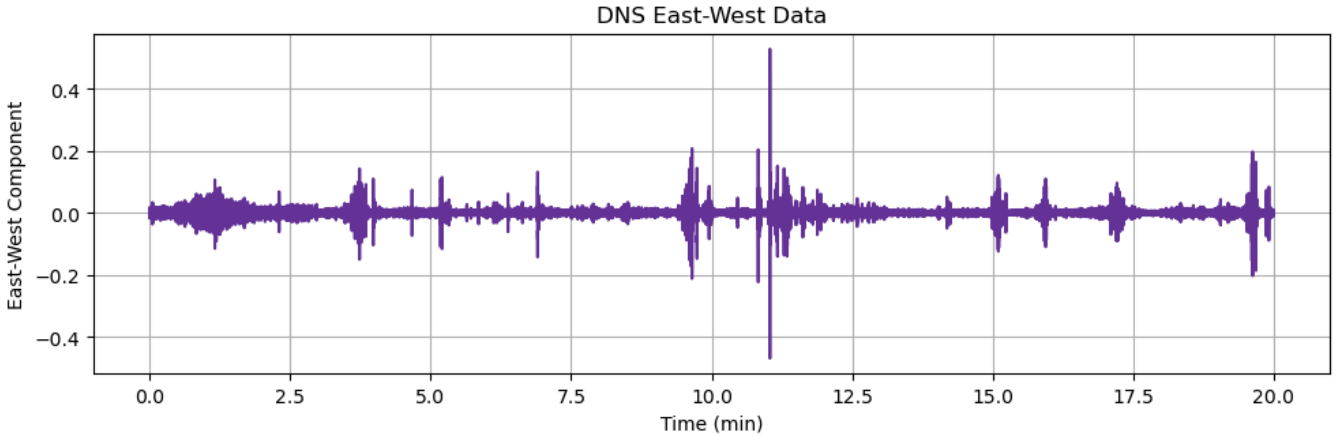
Normalizing, Detrending and Shifting Data

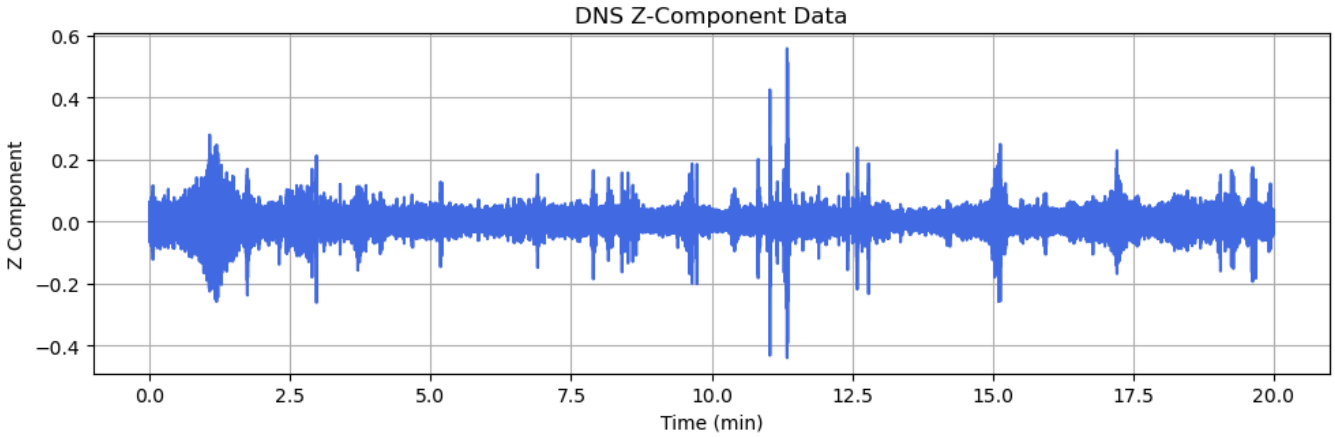
Notes:

- data range of time series was normalized using MinMaxScaler from sklearn
- data was linearly detrended using signal.detrend from scipy
- the mean of the data was shifted to zero

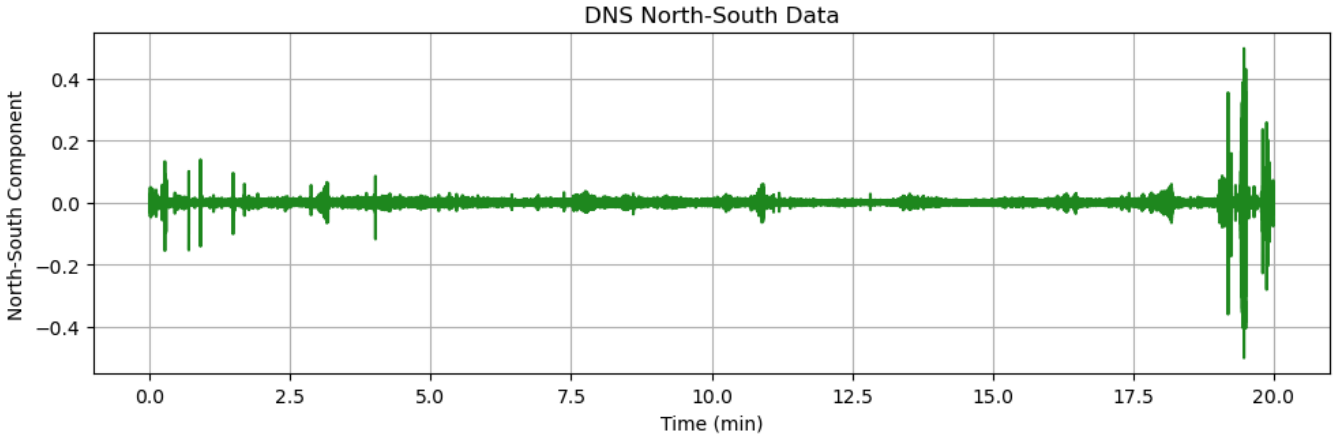
```
dataframe: df
NS Min-Max normalized range: [0.000, 1.000]
EW Min-Max normalized range: [0.000, 1.000]
Z Min-Max normalized range: [0.000, 1.000]
New range for NS: [-0.515, 0.485]
New mean for NS: 0.000000
New range for EW: [-0.470, 0.530]
New mean for EW: -0.000000
New range for Z: [-0.441, 0.559]
New mean for Z: -0.000000
```

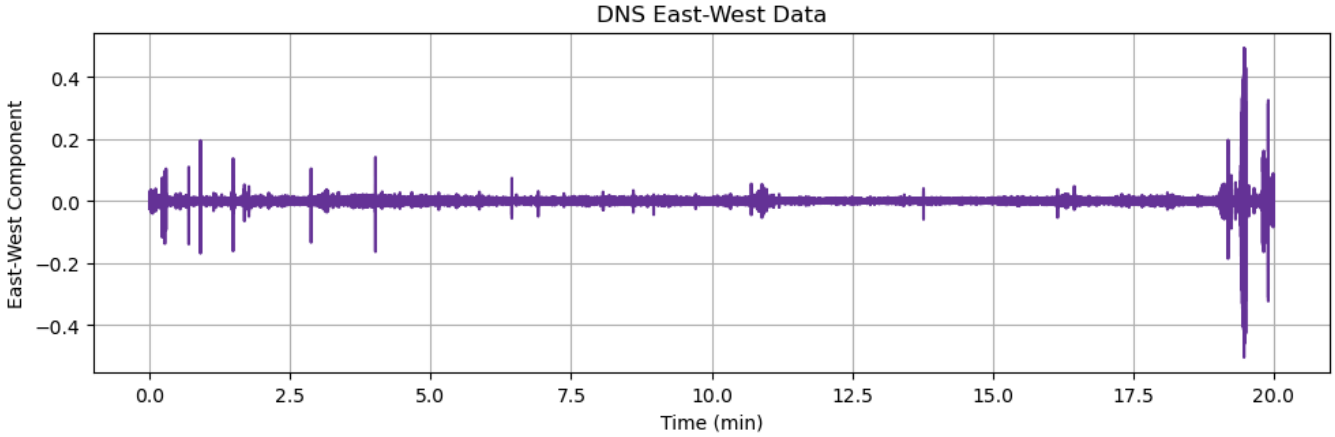


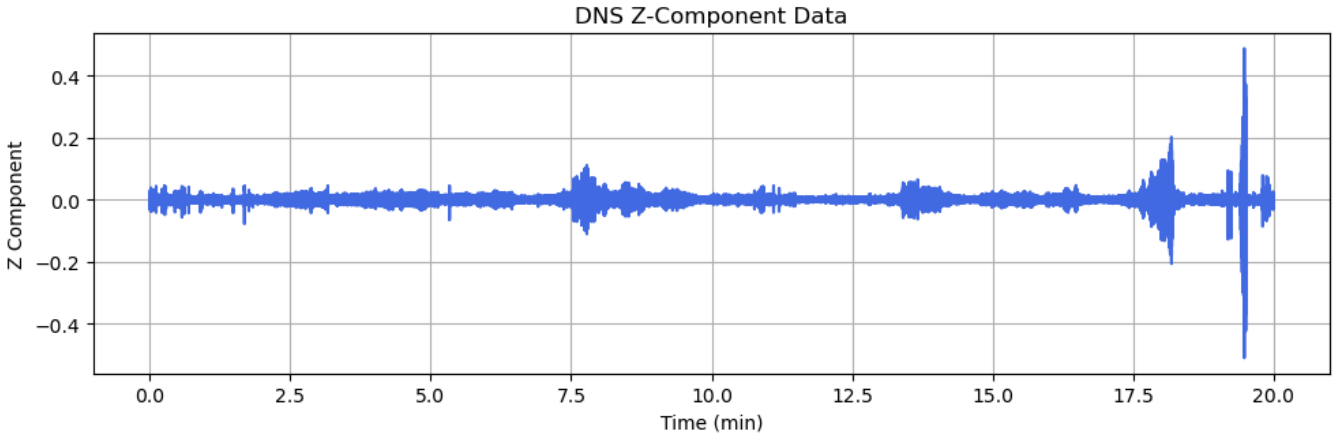




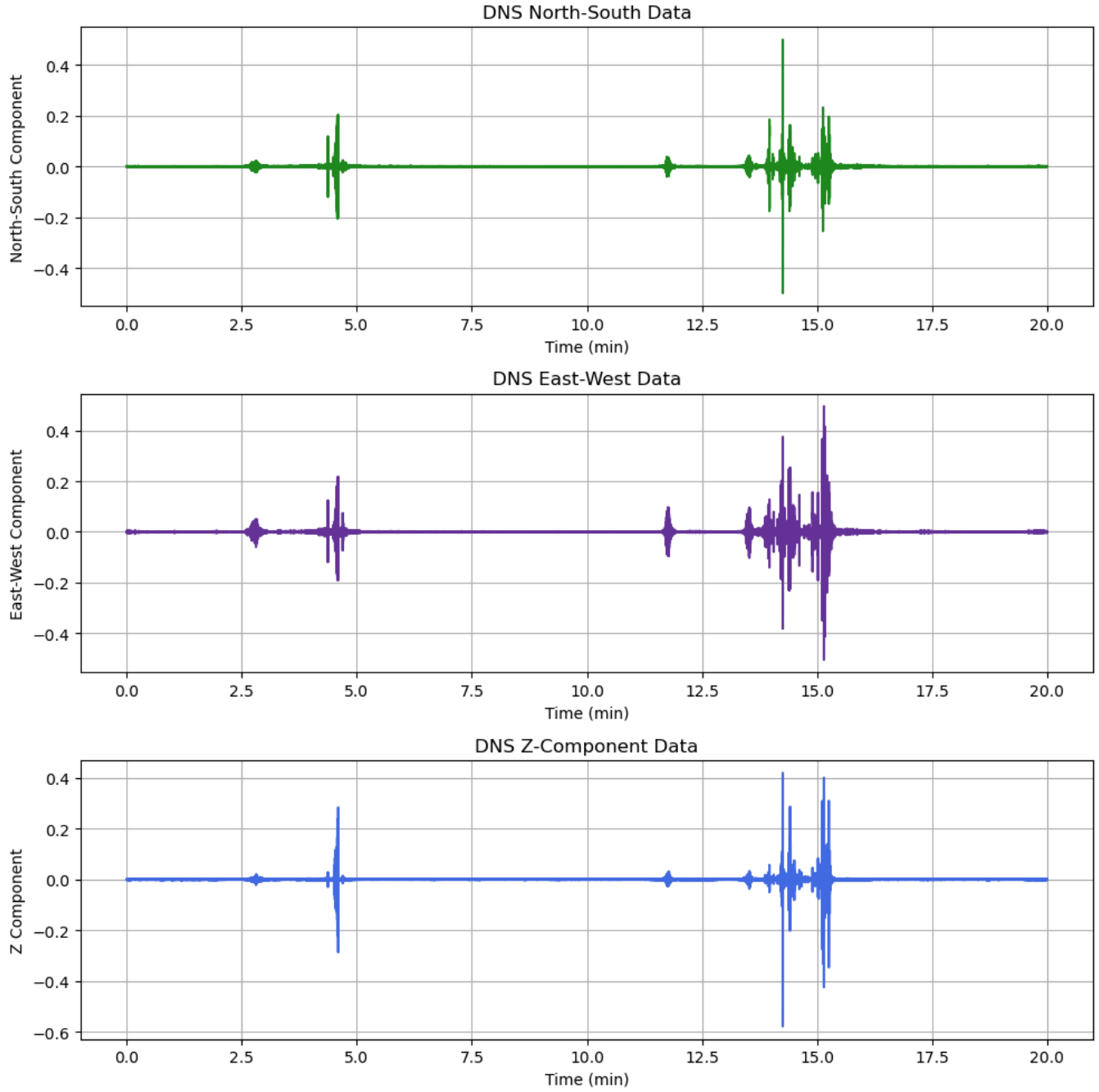
```
DNS = Detrended, Normalized, Shifted

North-South Stats:
DescribeResult(nobs=1228800, minmax=(-0.5152478241325915, 0.4847521758674085), mean=3.7876096
93476727e-16, variance=0.00022901459140630632, skewness=-0.1033085189717326, kurtosis=56.5387
2475095817)
median: 0.00012950548219936575
mode: ModeResult(mode=-0.5152478241325915, count=1)
standard deviation: 0.015133228056376447

East-West Stats:
DescribeResult(nobs=1228800, minmax=(-0.47034474769683, 0.5296552523031702), mean=-2.58037948
6472051e-16, variance=8.207605858568015e-05, skewness=-0.008233381277794327, kurtosis=149.351
8793822338)
median: 3.5697448634880447e-05
mode: ModeResult(mode=-0.47034474769683, count=1)
standard deviation: 0.009059583797596946

Z-component Stats:
DescribeResult(nobs=1228800, minmax=(-0.441206505335003, 0.5587934946649971), mean=-1.9058684
029108855e-16, variance=0.0005235396207434097, skewness=-0.09086115987043719, kurtosis=10.580
487012146193)
median: 0.0002861539687265824
mode: ModeResult(mode=-0.441206505335003, count=1)
standard deviation: 0.022880988194206968
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~
dataframe: df2
NS Min-Max normalized range: [0.000, 1.000]
EW Min-Max normalized range: [0.000, 1.000]
Z Min-Max normalized range: [0.000, 1.000]
New range for NS: [-0.502, 0.498]
New mean for NS: 0.000000
New range for EW: [-0.505, 0.495]
New mean for EW: -0.000000
New range for Z: [-0.512, 0.488]
New mean for Z: -0.000000
```


DNS North-South Data
North-South Component
0.4
0.2
0.0
−0.2
−0.4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Time (min)

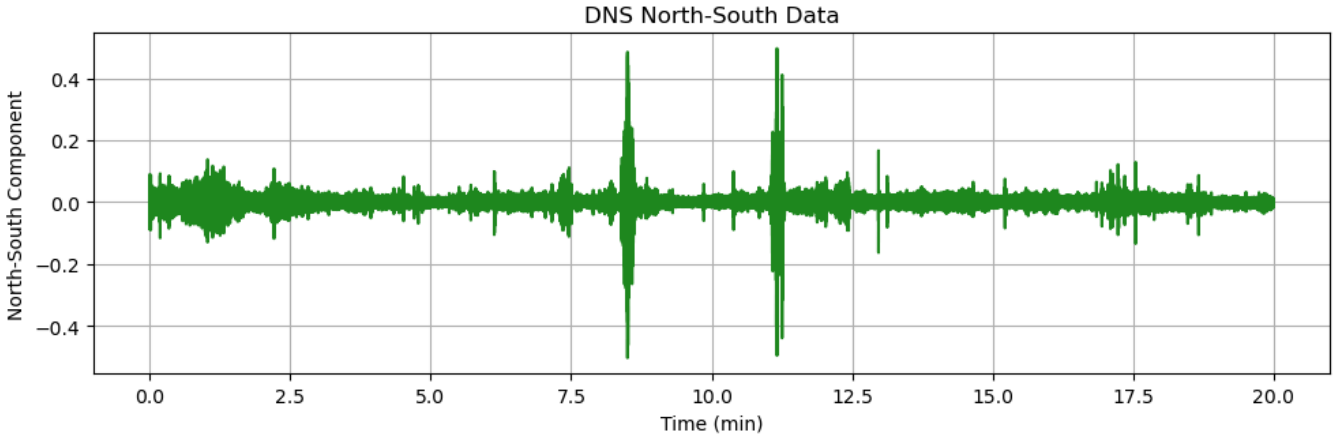

DNS East-West Data
East-West Component
0.4
0.2
0.0
−0.2
−0.4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Time (min)

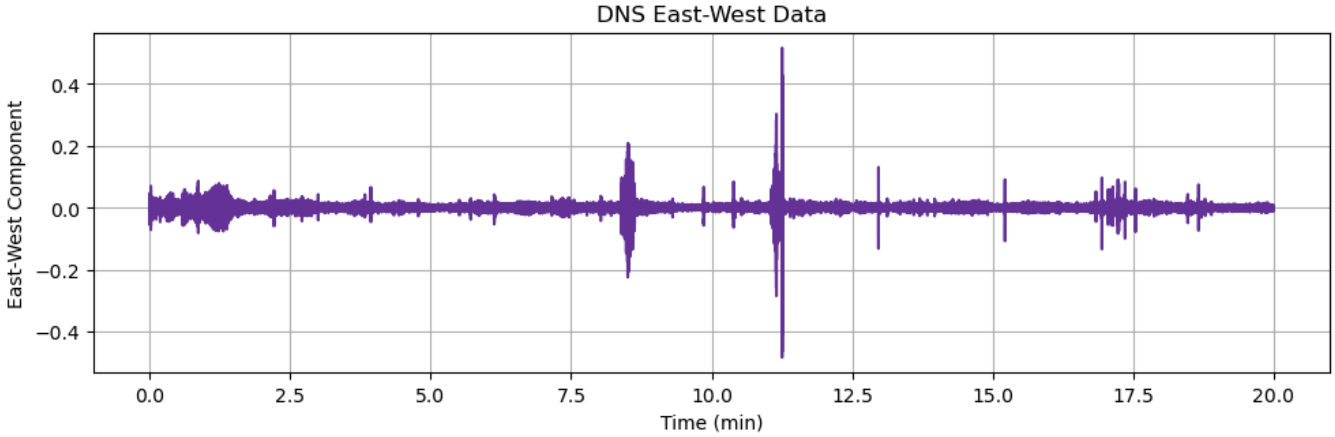

DNS Z-Component Data
Z Component
0.4
0.2
0.0
−0.2
−0.4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Time (min)

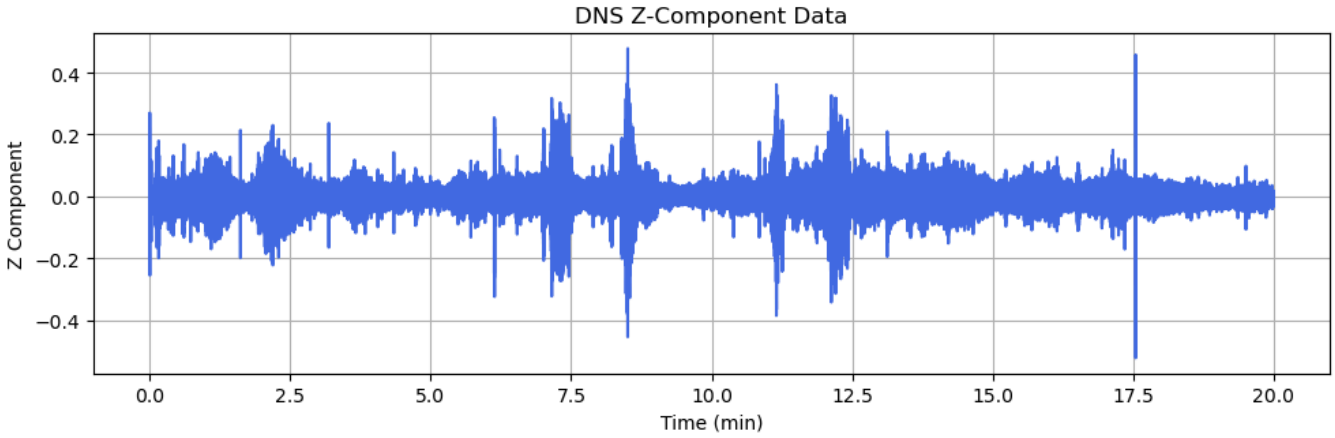
```
DNS = Detrended, Normalized, Shifted

North-South Stats:
DescribeResult(nobs=1228800, minmax=(-0.5019381220031599, 0.4980618779968399), mean=4.3424176
29181009e-16, variance=9.281385498144517e-05, skewness=-0.6297223640905917, kurtosis=493.8907
9662278203)
median: 2.1199030841523303e-05
mode: ModeResult(mode=-0.5152478241325915, count=1)
standard deviation: 0.009633994757183862

East-West Stats:
DescribeResult(nobs=1228800, minmax=(-0.5054436924811294, 0.4945563075188706), mean=-6.511935
088382437e-17, variance=0.00012834331184863885, skewness=-0.08707246741895831, kurtosis=493.4
782543586662)
median: -1.531246912561679e-05
mode: ModeResult(mode=-0.5054436924811294, count=1)
standard deviation: 0.01132887072256705

Z-component Stats:
DescribeResult(nobs=1228800, minmax=(-0.511524912555161, 0.48847508744483903), mean=-4.780970
1799369124e-18, variance=0.00016620456012125071, skewness=-0.3346075729511631, kurtosis=67.86
51994844411)
median: 5.935640950982535e-05
mode: ModeResult(mode=-0.511524912555161, count=1)
standard deviation: 0.012892034754888433
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~
dataframe: df3
NS Min-Max normalized range: [0.000, 1.000]
EW Min-Max normalized range: [0.000, 1.000]
Z Min-Max normalized range: [0.000, 1.000]
New range for NS: [-0.500, 0.500]
New mean for NS: 0.000000
New range for EW: [-0.505, 0.495]
New mean for EW: 0.000000
New range for Z: [-0.580, 0.420]
New mean for Z: -0.000000
```


DNS North-South Data
North-South Component
Time (min)
DNS East-West Data
East-West Component
Time (min)
DNS Z-Component Data
Z Component
Time (min)

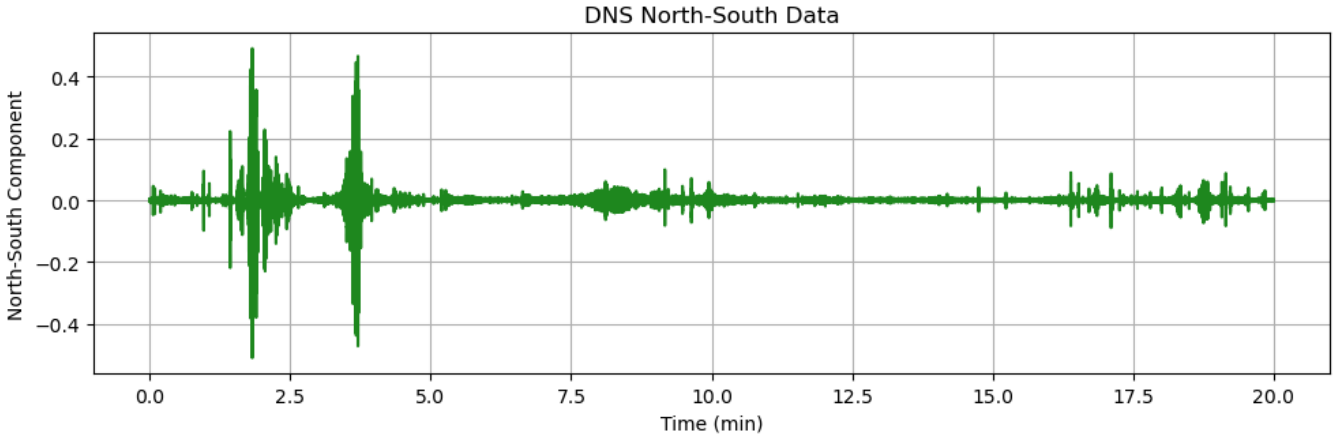
```
DNS = Detrended, Normalized, Shifted

North-South Stats:
DescribeResult(nobs=1228800, minmax=(-0.49994208758987224, 0.5000579124101278), mean=9.176777
53809835e-17, variance=2.945972925212792e-05, skewness=0.6501268003686306, kurtosis=871.11119
95058142)
median: -1.1423977730184998e-06
mode: ModeResult(mode=-0.5152478241325915, count=1)
standard deviation: 0.005427681756710559

East-West Stats:
DescribeResult(nobs=1228800, minmax=(-0.5045962431992747, 0.49540375680072524), mean=1.630457
1083015917e-16, variance=6.386936893778217e-05, skewness=0.2222447495114245, kurtosis=526.003
5362484331)
median: -4.021702497969404e-06
mode: ModeResult(mode=-0.5045962431992747, count=1)
standard deviation: 0.007991831388222657

Z-component Stats:
DescribeResult(nobs=1228800, minmax=(-0.5795712973134393, 0.42042870268656074), mean=-4.19602
1423837533e-16, variance=1.9544125053985618e-05, skewness=-2.4679640755390464, kurtosis=1886.
793203472342)
median: -2.0765811693523872e-06
mode: ModeResult(mode=-0.5795712973134393, count=1)
standard deviation: 0.0044208737885157055
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~
dataframe: df4
NS Min-Max normalized range: [0.000, 1.000]
EW Min-Max normalized range: [0.000, 1.000]
Z Min-Max normalized range: [0.000, 1.000]
New range for NS: [-0.504, 0.496]
New mean for NS: -0.000000
New range for EW: [-0.484, 0.516]
New mean for EW: 0.000000
New range for Z: [-0.522, 0.478]
New mean for Z: 0.000000
```

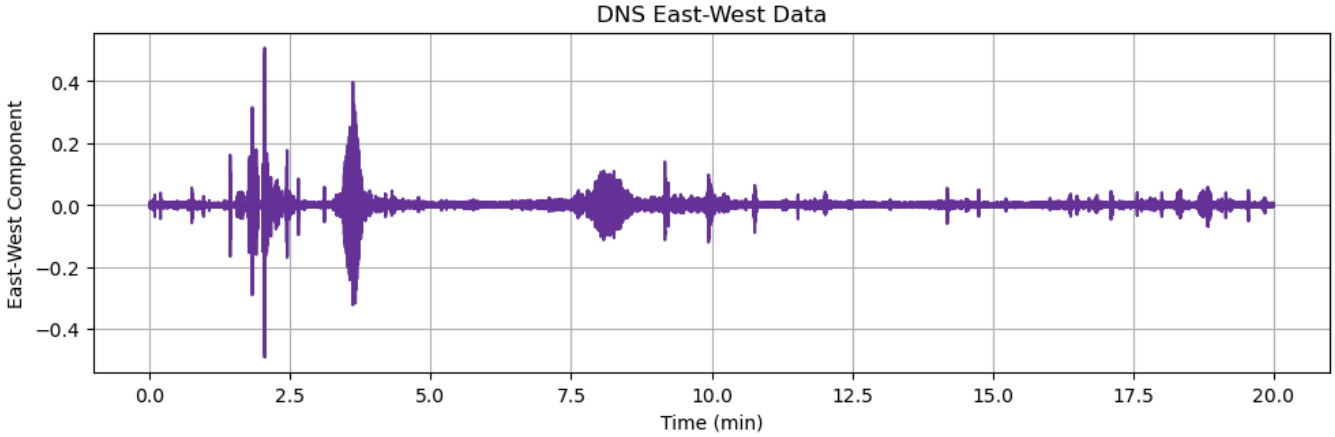

DNS North-South Data
North-South Component
0.4
0.2
0.0
−0.2
−0.4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Time (min)

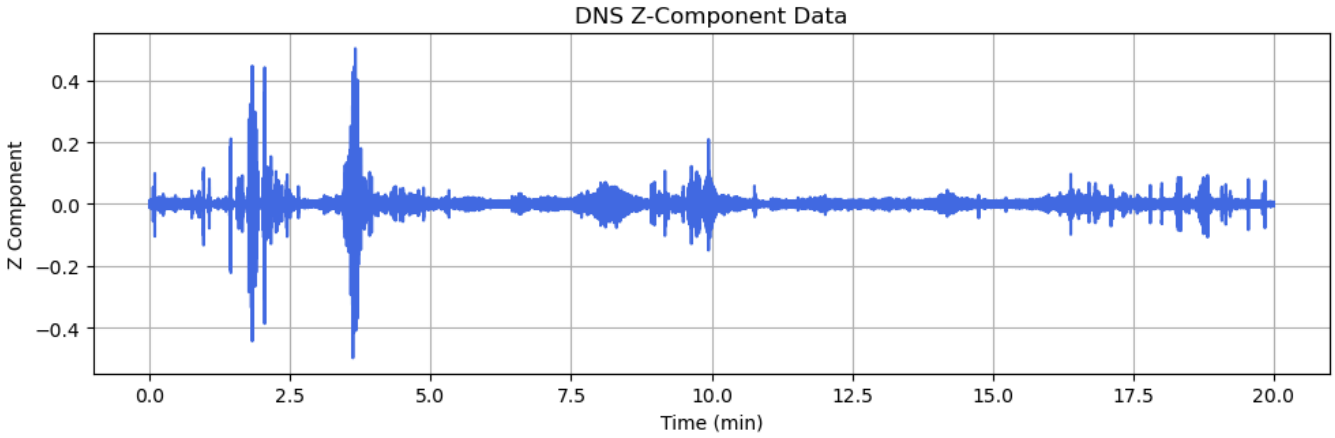

DNS East-West Data
East-West Component
0.4
0.2
0.0
−0.2
−0.4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Time (min)


DNS Z-Component Data
Z Component
0.4
0.2
0.0
−0.2
−0.4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Time (min)

```
DNS = Detrended, Normalized, Shifted

North-South Stats:
DescribeResult(nobs=1228800, minmax=(-0.5035512895100599, 0.4964487104899403), mean=-1.669844
8179752746e-16, variance=0.0003347398421525503, skewness=0.12254345842413947, kurtosis=90.858
82133355828)
median: -4.582177470369153e-05
mode: ModeResult(mode=-0.5152478241325915, count=1)
standard deviation: 0.018295896866580286

East-West Stats:
DescribeResult(nobs=1228800, minmax=(-0.48401281650907285, 0.5159871834909271), mean=3.802224
7387618316e-17, variance=0.00010135938094986332, skewness=0.11743914004407581, kurtosis=174.8
677707538525)
median: -2.0929566298039104e-05
mode: ModeResult(mode=-0.48401281650907285, count=1)
standard deviation: 0.010067739614723117

Z-component Stats:
DescribeResult(nobs=1228800, minmax=(-0.5220450589073411, 0.4779549410926589), mean=2.4287285
145992616e-16, variance=0.0011552934365230184, skewness=0.11061175651743019, kurtosis=10.8717
88758457647)
median: -0.0002585766612292484
mode: ModeResult(mode=-0.5220450589073411, count=1)
standard deviation: 0.033989607772421285
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~
dataframe: df5
NS Min-Max normalized range: [0.000, 1.000]
EW Min-Max normalized range: [0.000, 1.000]
Z Min-Max normalized range: [0.000, 1.000]
New range for NS: [-0.510, 0.490]
New mean for NS: 0.000000
New range for EW: [-0.493, 0.507]
New mean for EW: 0.000000
New range for Z: [-0.497, 0.503]
New mean for Z: -0.000000
```


DNS North-South Data
North-South Component
0.4
0.2
0.0
−0.2
−0.4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Time (min)


DNS East-West Data
East-West Component
0.4
0.2
0.0
−0.2
−0.4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Time (min)


DNS Z-Component Data
Z Component
0.4
0.2
0.0
−0.2
−0.4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Time (min)

```
DNS = Detrended, Normalized, Shifted

North-South Stats:
DescribeResult(nobs=1228800, minmax=(-0.5096600529220939, 0.49033994707790607), mean=2.969376
7699308836e-16, variance=0.00010415379382992493, skewness=-0.023853380096099826, kurtosis=37
1.55586172494714)
median: -2.5429592468872997e-05
mode: ModeResult(mode=-0.5152478241325915, count=1)
standard deviation: 0.010205576604480657

East-West Stats:
DescribeResult(nobs=1228800, minmax=(-0.49287762251997863, 0.5071223774800213), mean=1.320232
9854355987e-17, variance=0.00013912613597184646, skewness=0.052231500088967535, kurtosis=174.
31062086637456)
median: -2.8702646493461437e-06
mode: ModeResult(mode=-0.49287762251997863, count=1)
standard deviation: 0.011795174266277164

Z-component Stats:
DescribeResult(nobs=1228800, minmax=(-0.4972659843471705, 0.5027340156528295), mean=-1.575101
8111326288e-16, variance=0.00014779390976688687, skewness=-0.033495388745043334, kurtosis=13
1.81490838854597)
median: -1.860860267985376e-05
mode: ModeResult(mode=-0.4972659843471705, count=1)
standard deviation: 0.012157051853426025
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~
```

Subsampling the Time Series

Notes:

- the original sampling rate for the data is 1024 Hz
- different step sizes were applied to the normalized, detrended and mean-shifted (DNS) data for subsampling
- adjust x-axis limits to focus in on certain times

"percent_check" Function for Choosing Subsampling Step Size:

Notes:

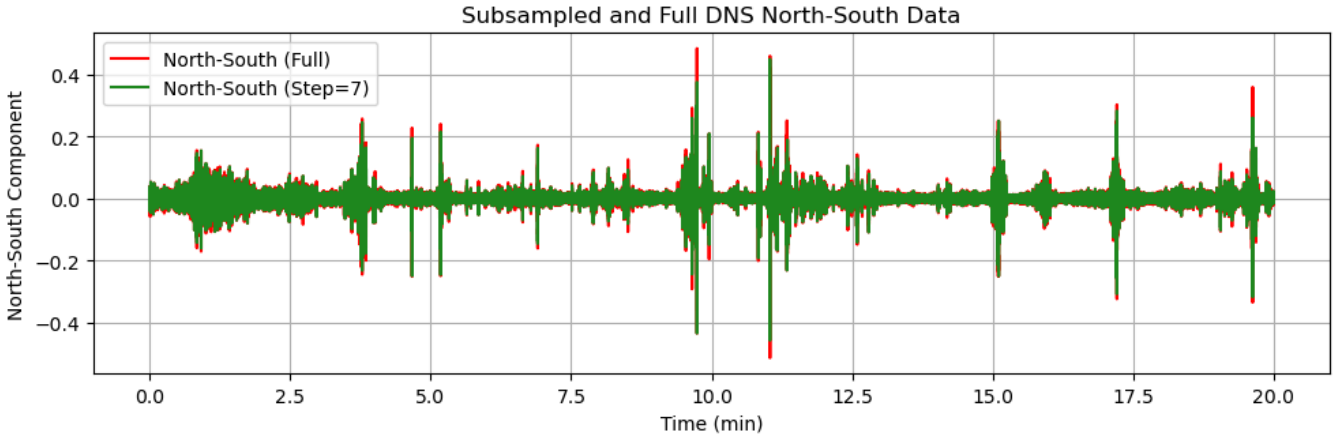
- this method uses interpolation to compare subsampled data to the full DNS data
- the absolute error between the full DNS data and the reconstructed subsampled data is calculated and divided by the full data range
- this result is then used determine if the error is within the selected threshold (i.e. 5%)

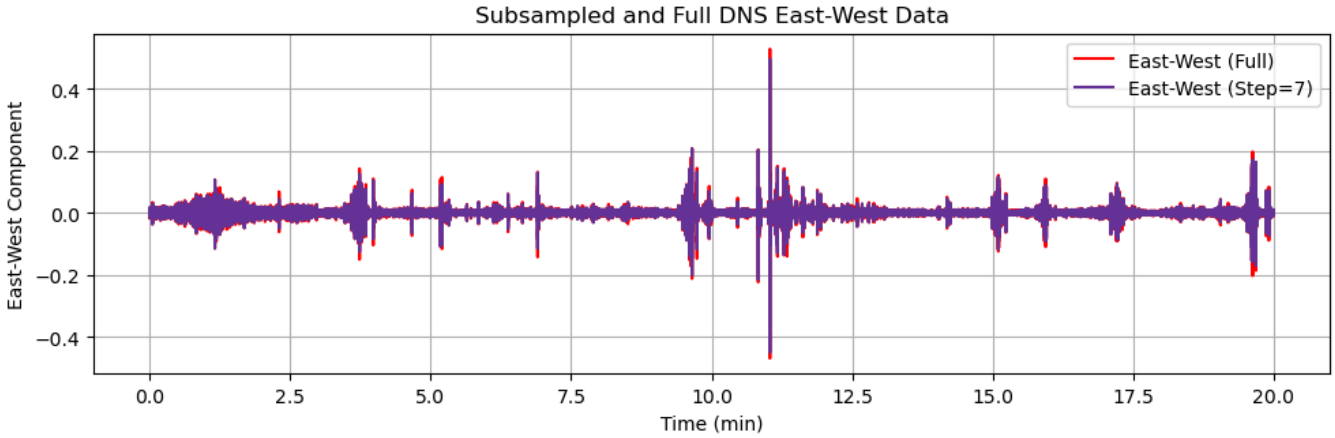
```
NS results 1 (df): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_toleranc
e': ' 99.9457%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_rang
e': ' 1.0000', 'max_normalized_error': ' 19.2395%'}
EW results 1 (df): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_toleranc
e': ' 99.9919%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_rang
e': ' 1.0000', 'max_normalized_error': ' 16.9826%'}
Z results 1 (df): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_tolerance':
' 99.0938%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_range': '
1.0000', 'max_normalized_error': ' 69.2043%'}

NS results 1 (df2): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_toleranc
e': ' 99.9462%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_rang
e': ' 1.0000', 'max_normalized_error': ' 20.9703%'}
EW results 1 (df2): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_toleranc
e': ' 99.9166%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_rang
e': ' 1.0000', 'max_normalized_error': ' 18.6714%'}
Z results 1 (df2): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_toleranc
e': ' 99.3123%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_rang
e': ' 1.0000', 'max_normalized_error': ' 47.4195%'}

NS results 1 (df3): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_toleranc
e': ' 99.9146%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_rang
e': ' 1.0000', 'max_normalized_error': ' 39.4024%'}
EW results 1 (df3): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_toleranc
e': ' 99.8758%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_rang
e': ' 1.0000', 'max_normalized_error': ' 36.5209%'}
Z results 1 (df3): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_toleranc
e': ' 99.8949%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_rang
e': ' 1.0000', 'max_normalized_error': ' 82.8238%'}

NS results 1 (df4): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_toleranc
e': ' 99.7948%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_rang
e': ' 1.0000', 'max_normalized_error': ' 16.6893%'}
EW results 1 (df4): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_toleranc
e': ' 99.9743%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_rang
e': ' 1.0000', 'max_normalized_error': ' 19.5554%'}
Z results 1 (df4): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_toleranc
e': ' 95.9887%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_rang
e': ' 1.0000', 'max_normalized_error': ' 36.9399%'}

NS results 1 (df5): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_toleranc
e': ' 99.8687%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_rang
e': ' 1.0000', 'max_normalized_error': ' 29.3628%'}
EW results 1 (df5): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_toleranc
e': ' 99.9373%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_rang
e': ' 1.0000', 'max_normalized_error': ' 18.6268%'}
Z results 1 (df5): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_toleranc
e': ' 99.5474%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_rang
e': ' 1.0000', 'max_normalized_error': ' 47.0429%'}
```

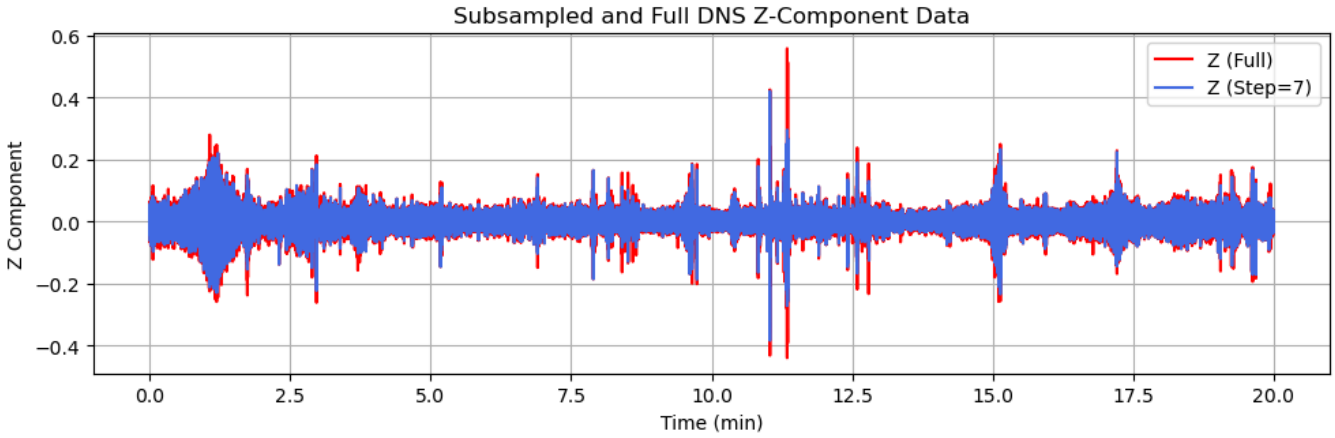
percent_check results:

- A step size of 7 was the largest that passed the percent_check test for NS, EW, and Z data simultaneously for all dataframes (plotted below)

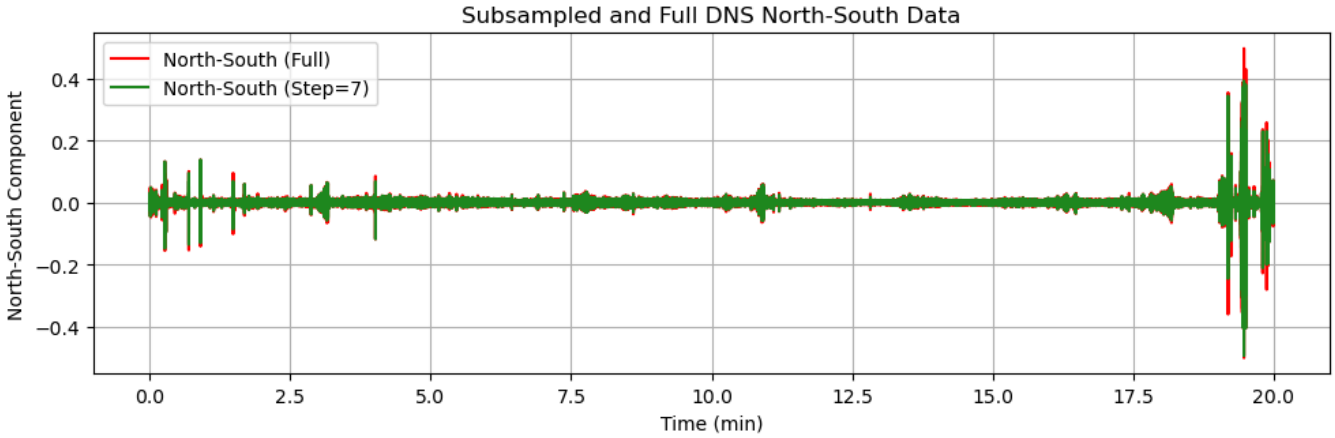
dataframe: df

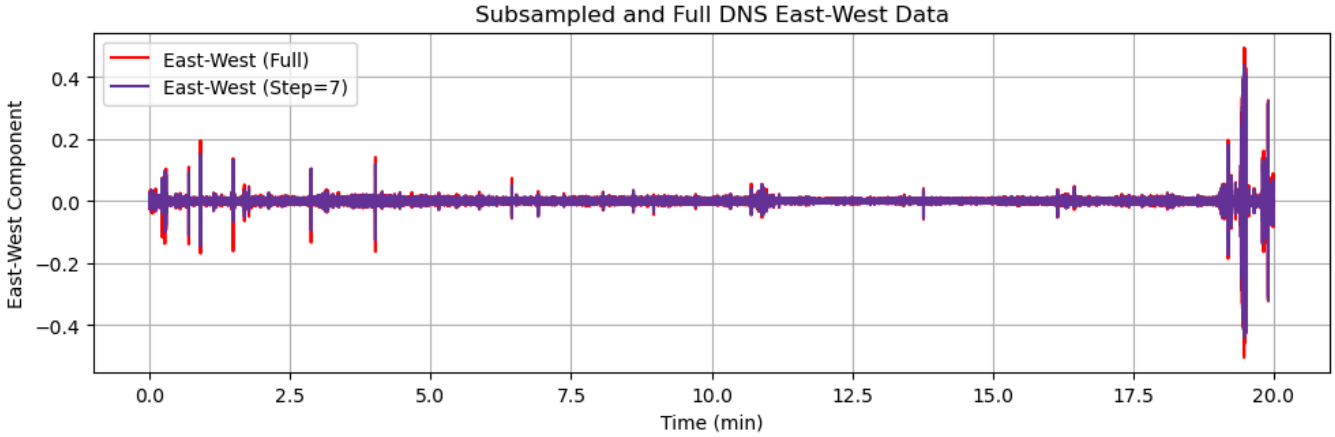


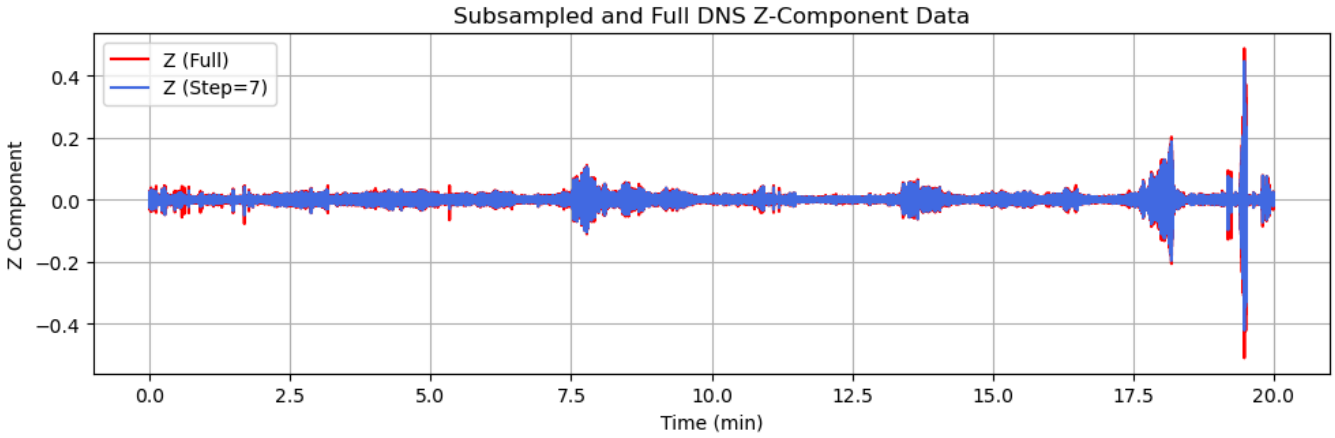




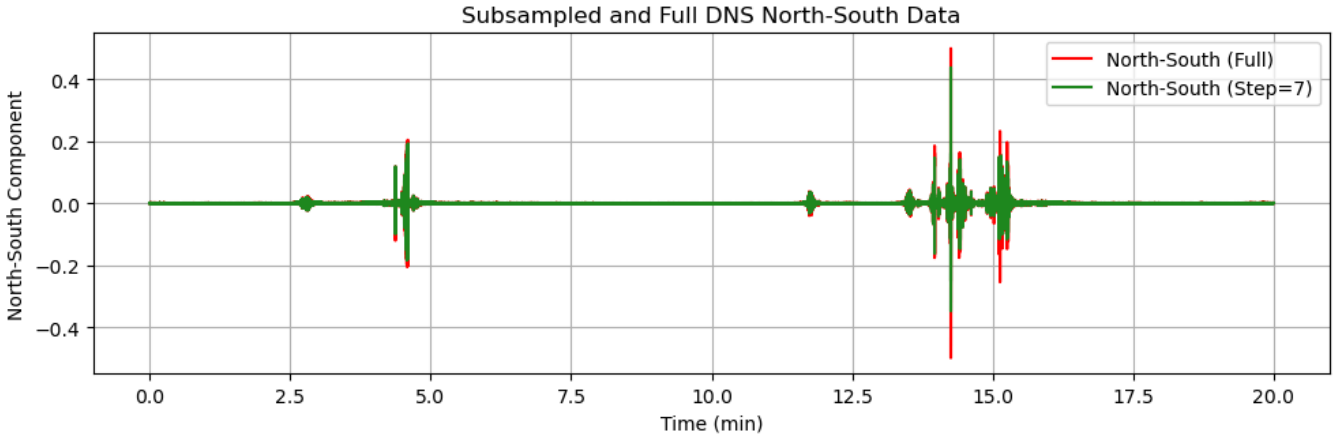
dataframe: df2

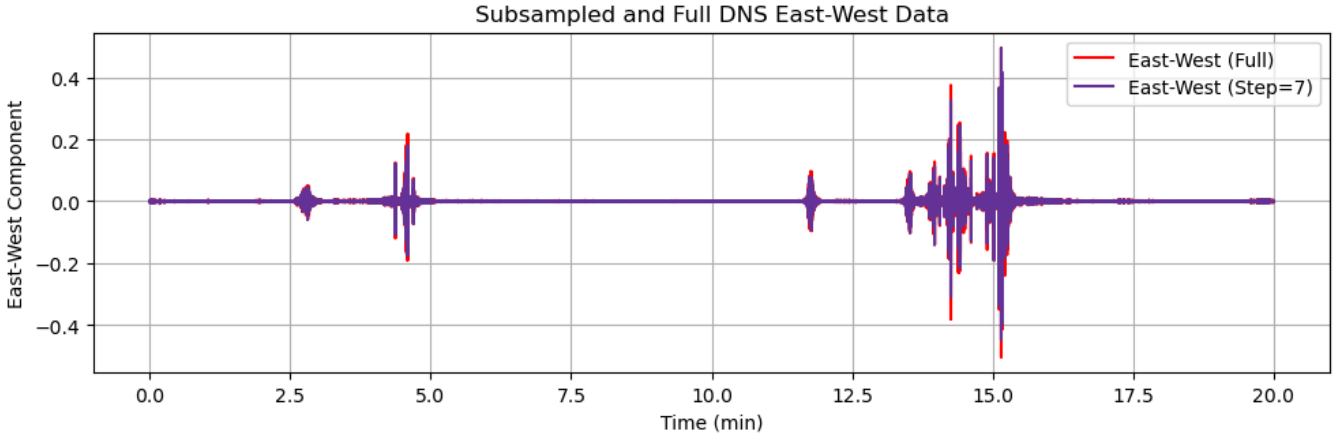


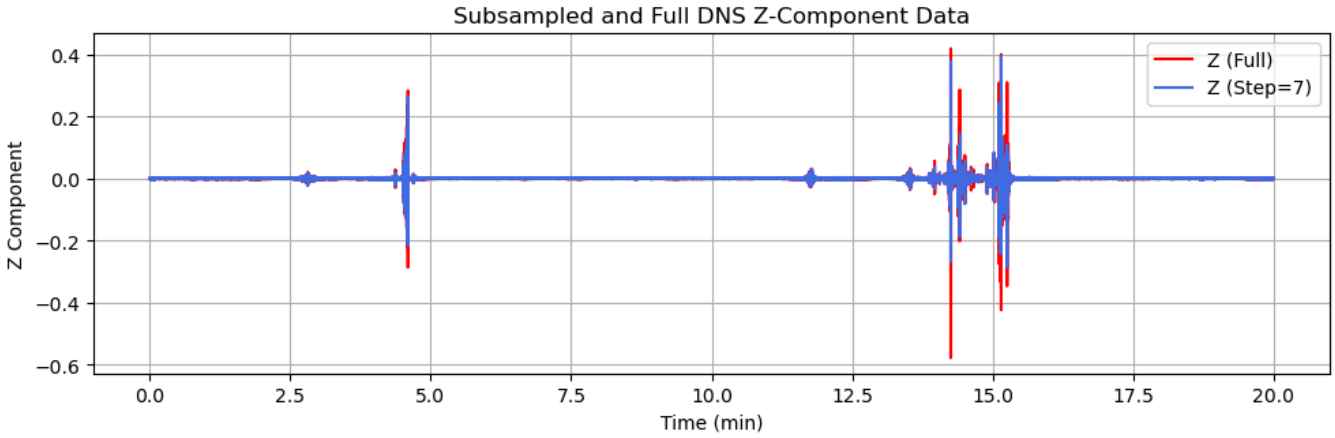




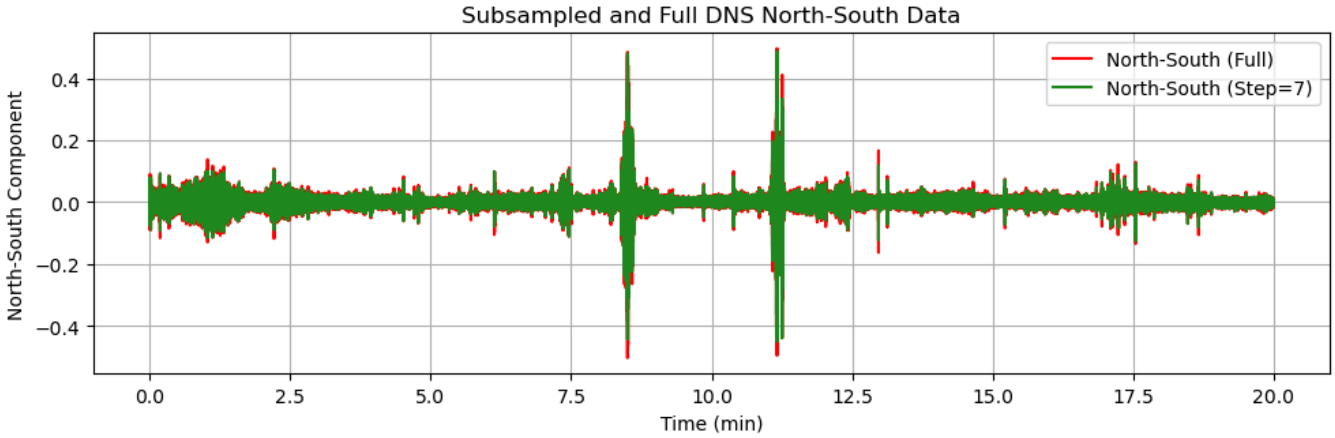
`dataframe: df3`

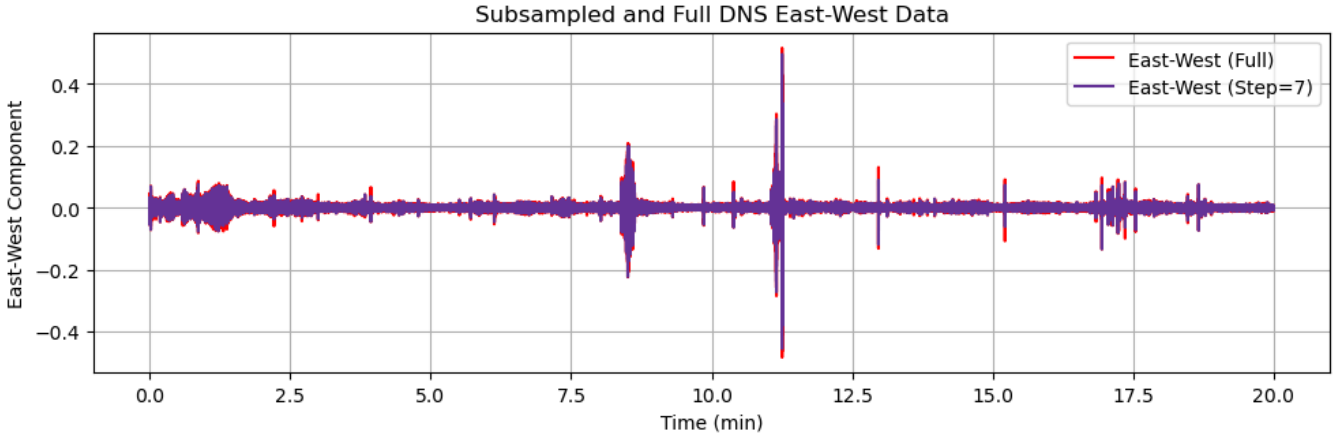


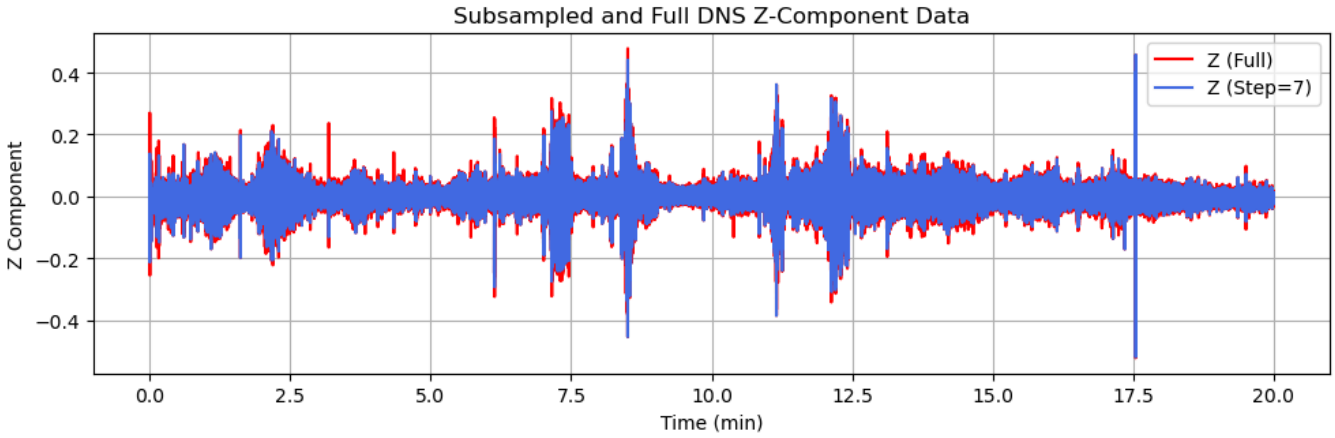




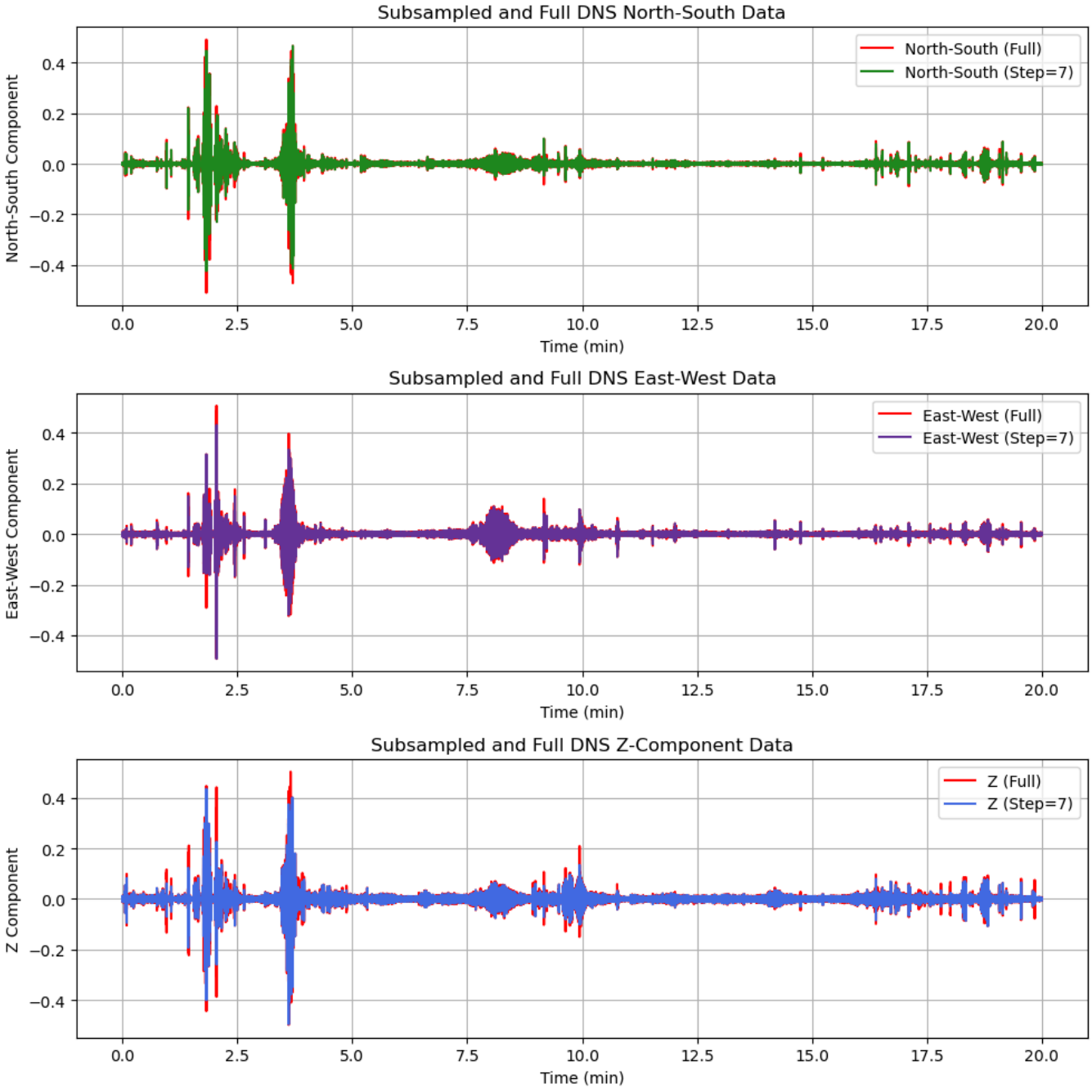
dataframe: df4







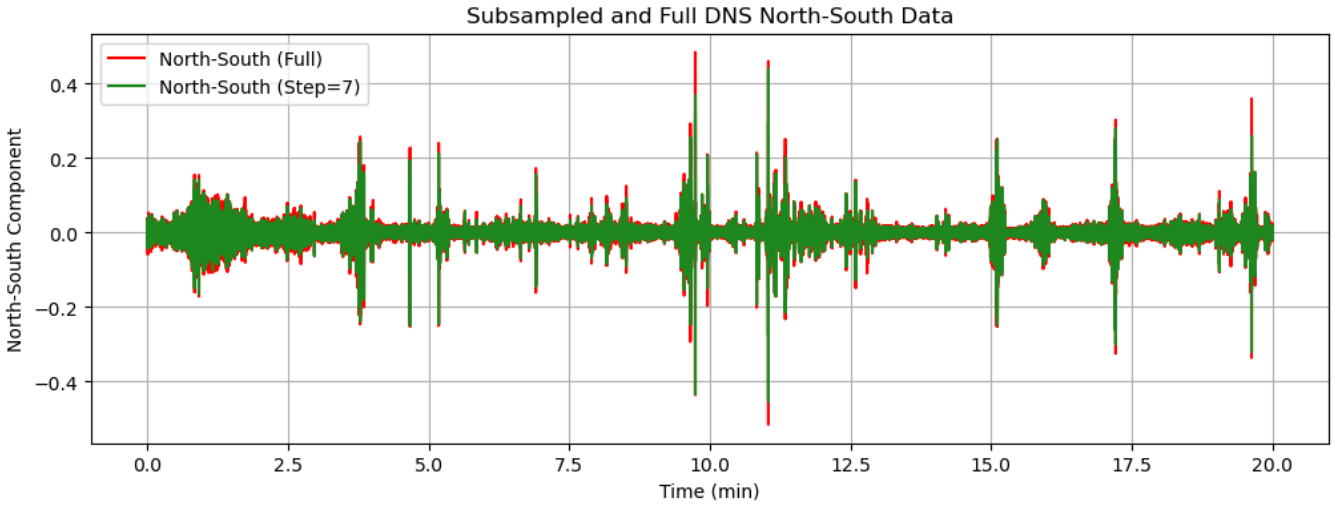
`dataframe: df5`

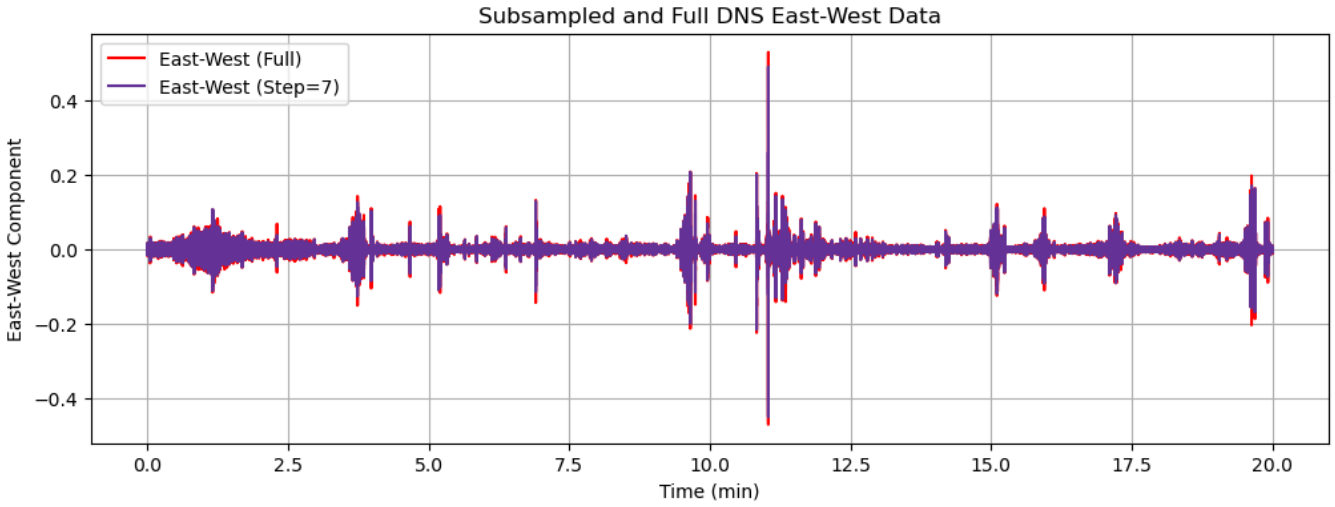


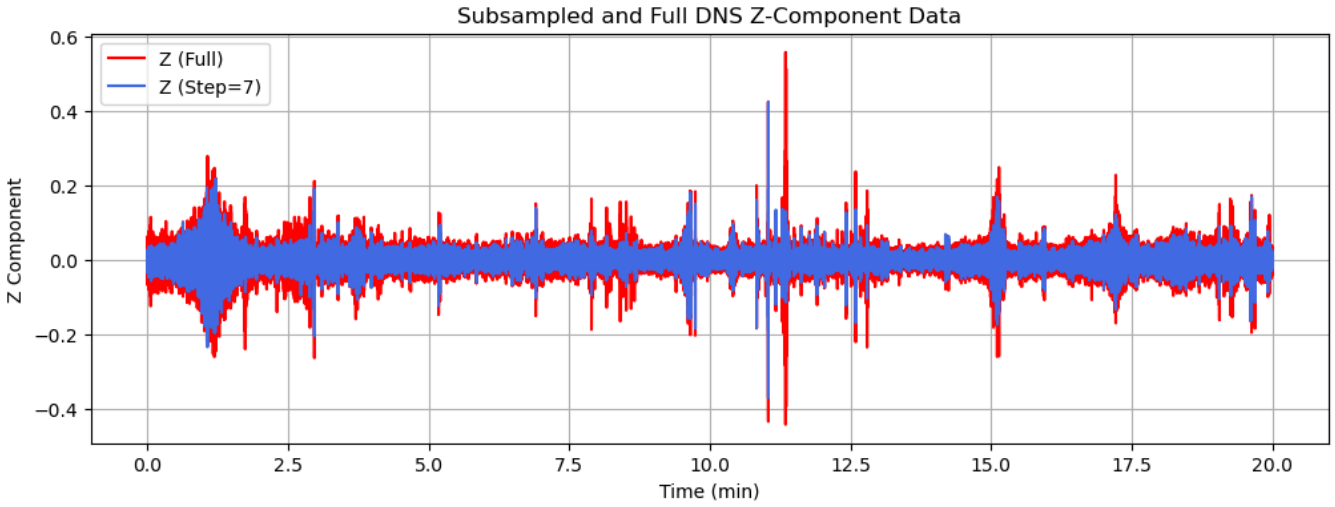
percent_check function using scipy.signal.decimate for y_subsampled instead of indexing:

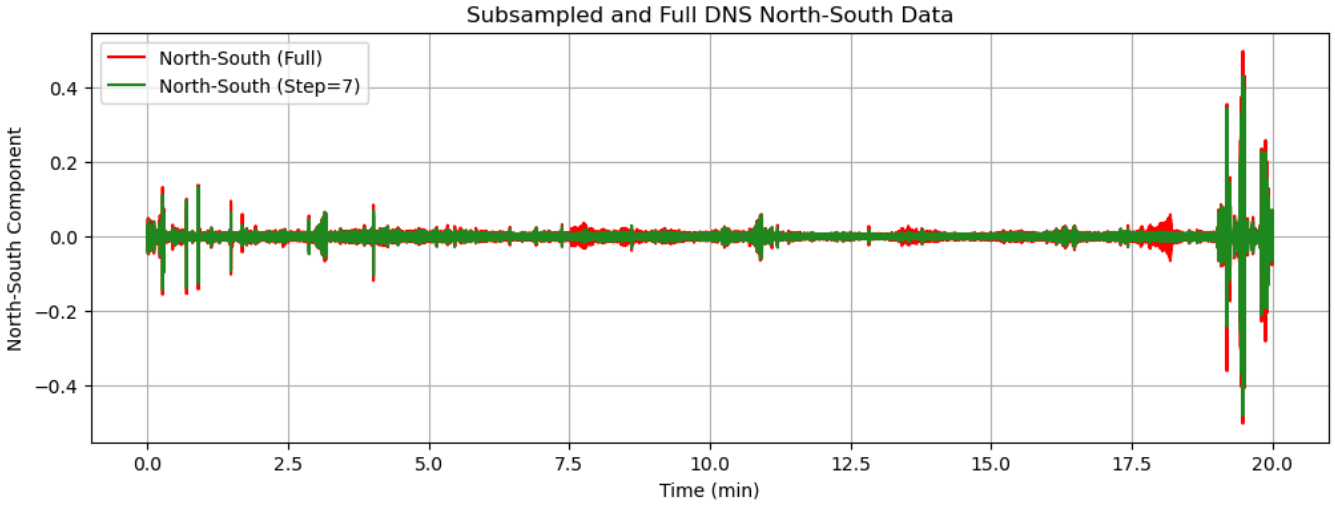
- testing a different method for subsampling with the same step size (i.e. 7)

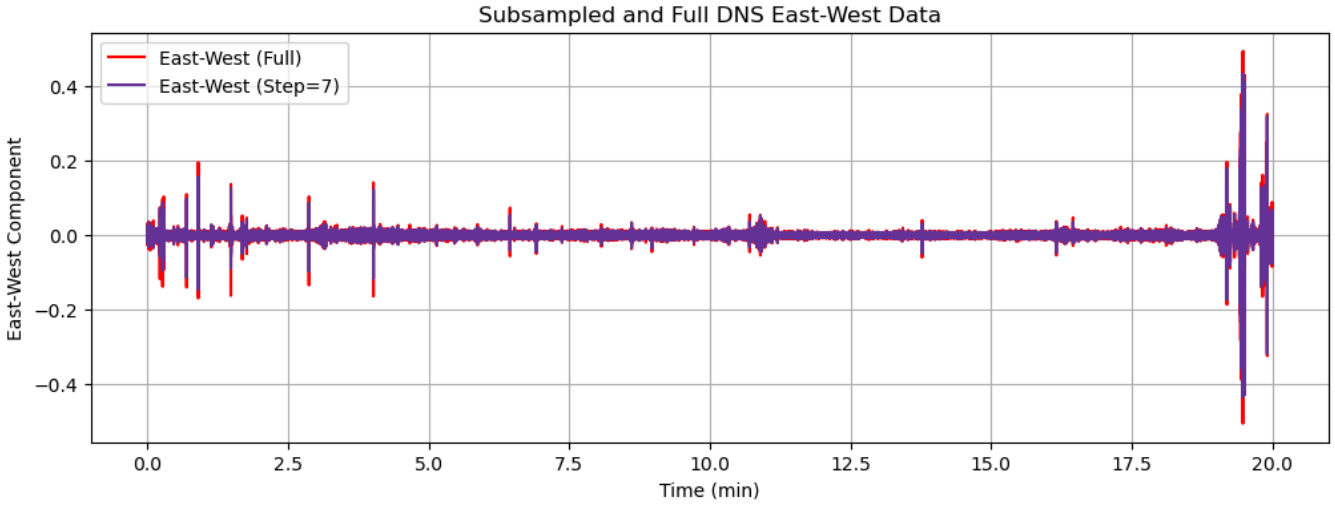
```
NS_results_test_1 (df): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_toler
ance': ' 99.9447%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_ra
nge': ' 1.0000', 'max_normalized_error': ' 19.6512%'}
EW_results_test_1 (df): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_toler
ance': ' 99.9924%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_ra
nge': ' 1.0000', 'max_normalized_error': ' 17.1854%'}
Z_results_test_1 (df): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_tolera
nce': ' 98.9932%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_ran
ge': ' 1.0000', 'max_normalized_error': ' 56.3167%'}

NS_results_test_1 (df2): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_tole
rance': ' 99.9451%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_r
ange': ' 1.0000', 'max_normalized_error': ' 20.7711%'}
EW_results_test_1 (df2): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_tole
rance': ' 99.9145%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_r
ange': ' 1.0000', 'max_normalized_error': ' 18.5602%'}
Z_results_test_1 (df2): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_toler
ance': ' 98.9080%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_ra
nge': ' 1.0000', 'max_normalized_error': ' 40.9484%'}

NS_results_test_1 (df3): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_tole
rance': ' 99.8774%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_r
ange': ' 1.0000', 'max_normalized_error': ' 39.9450%'}
EW_results_test_1 (df3): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_tole
rance': ' 99.8002%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_r
ange': ' 1.0000', 'max_normalized_error': ' 40.7109%'}
Z_results_test_1 (df3): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_toler
ance': ' 99.8956%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_ra
nge': ' 1.0000', 'max_normalized_error': ' 56.4975%'}

NS_results_test_1 (df4): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_tole
rance': ' 99.6542%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_r
ange': ' 1.0000', 'max_normalized_error': ' 18.0818%'}
EW_results_test_1 (df4): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_tole
rance': ' 99.9735%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_r
ange': ' 1.0000', 'max_normalized_error': ' 19.6565%'}
Z_results_test_1 (df4): {'step': 7, 'tolerance': '5.0%', 'passes': False, 'points_within_tole
rance': ' 93.4352%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_r
ange': ' 1.0000', 'max_normalized_error': ' 39.5411%'}

NS_results_test_1 (df5): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_tole
rance': ' 99.8494%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_r
ange': ' 1.0000', 'max_normalized_error': ' 30.5137%'}
EW_results_test_1 (df5): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_tole
rance': ' 99.9371%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_r
ange': ' 1.0000', 'max_normalized_error': ' 18.8787%'}
Z_results_test_1 (df5): {'step': 7, 'tolerance': '5.0%', 'passes': True, 'points_within_toler
ance': ' 99.3926%', 'num_original_points': 1228800, 'num_subsampled_points': 175543, 'data_ra
nge': ' 1.0000', 'max_normalized_error': ' 42.2311%'}
```

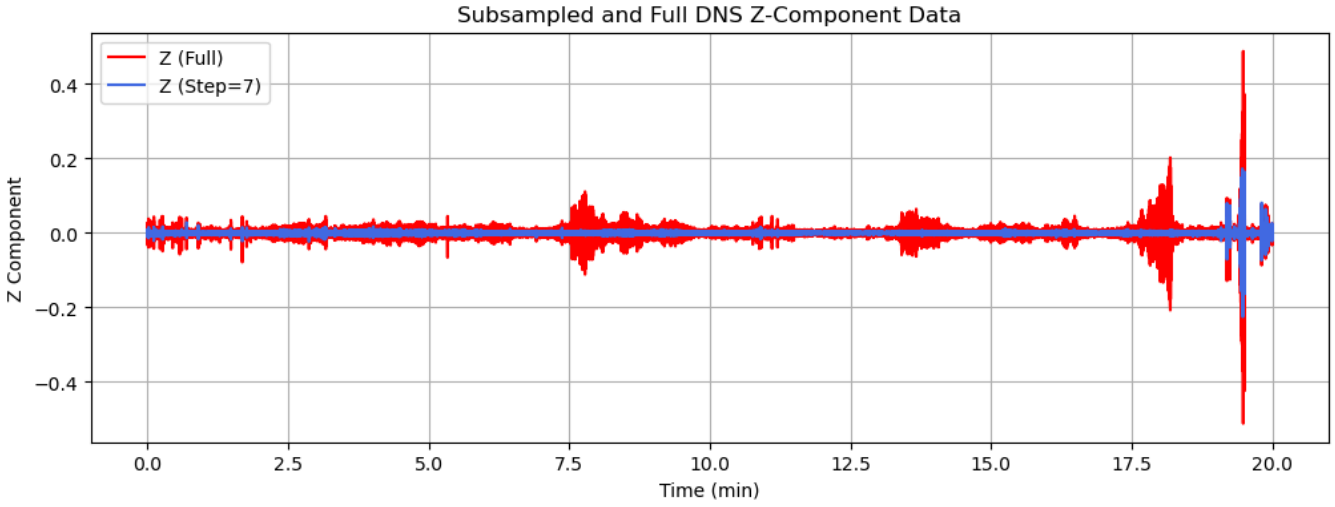

Subsampled and Full DNS North-South Data
North-South (Full)
North-South (Step=7)
North-South Component
Time (min)

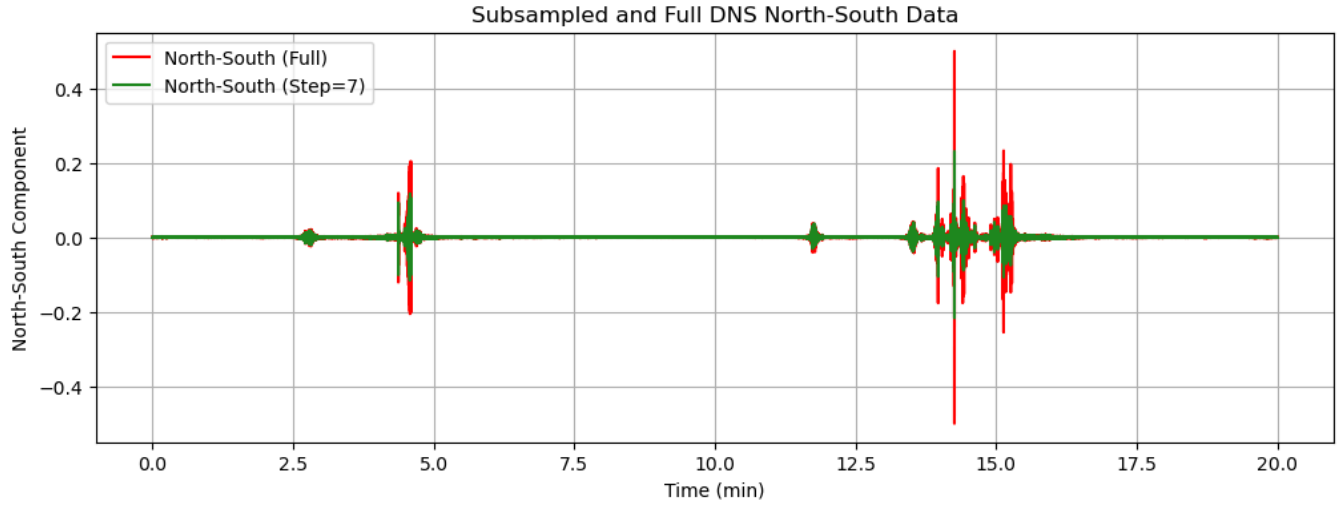

Subsampled and Full DNS East-West Data
East-West (Full)
East-West (Step=7)
East-West Component
Time (min)

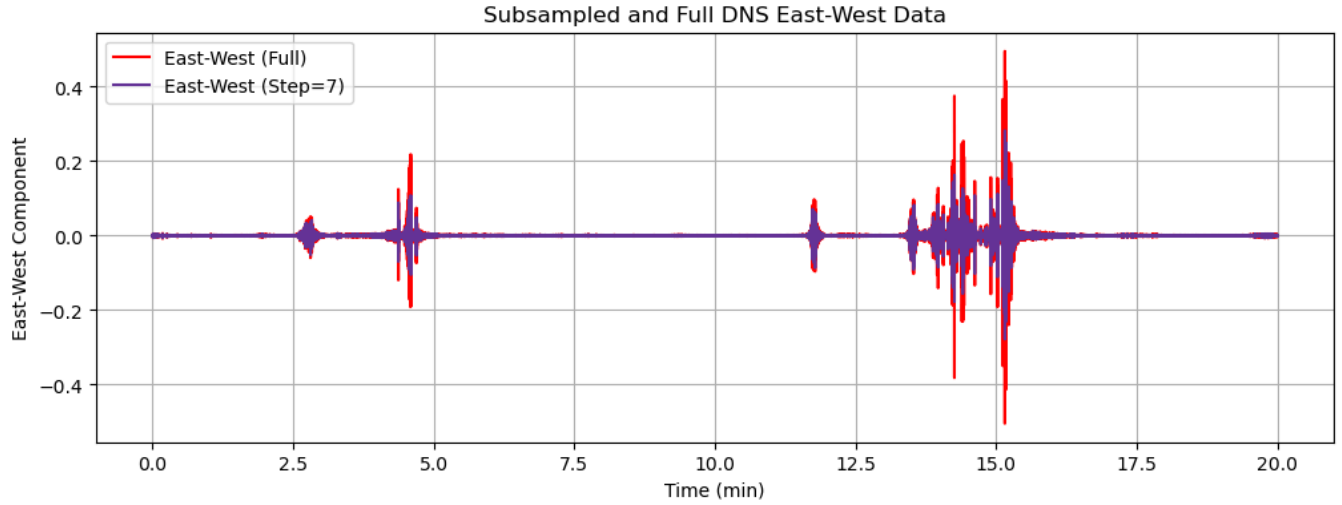

Subsampled and Full DNS Z-Component Data
Z (Full)
Z (Step=7)
Z Component
Time (min)

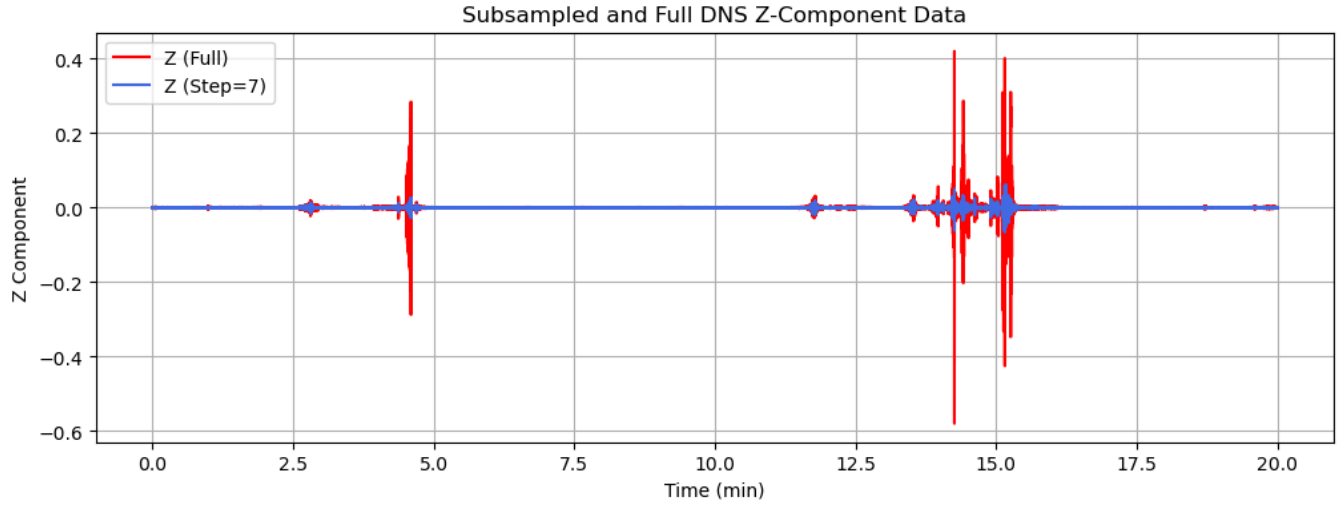

Subsampled and Full DNS North-South Data
North-South (Full)
North-South (Step=7)
North-South Component
Time (min)

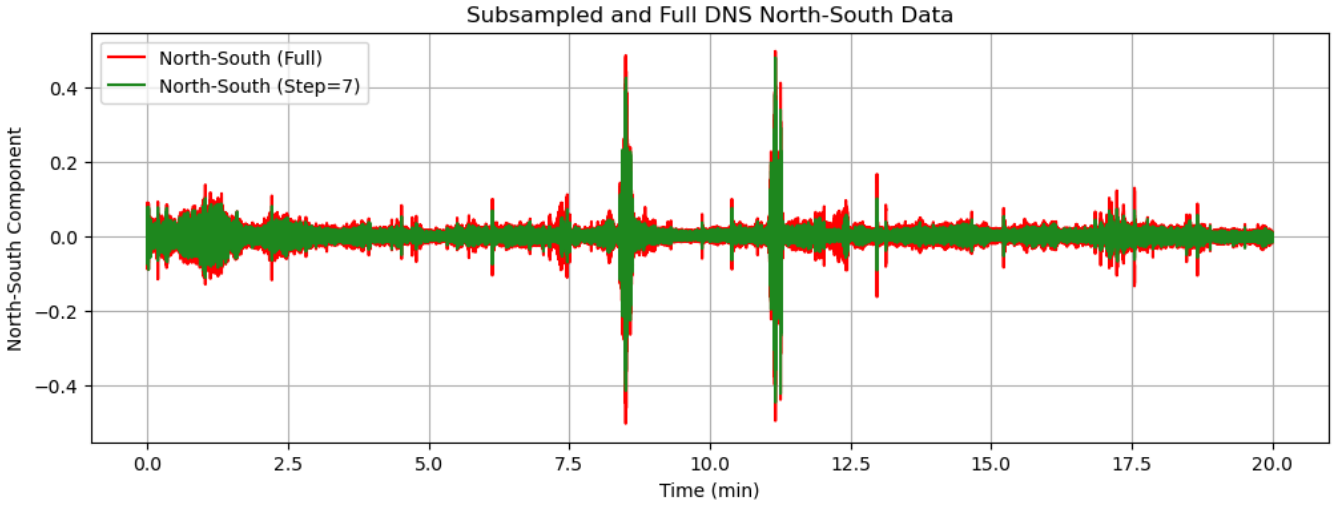

Subsampled and Full DNS East-West Data
East-West (Full)
East-West (Step=7)
East-West Component
Time (min)

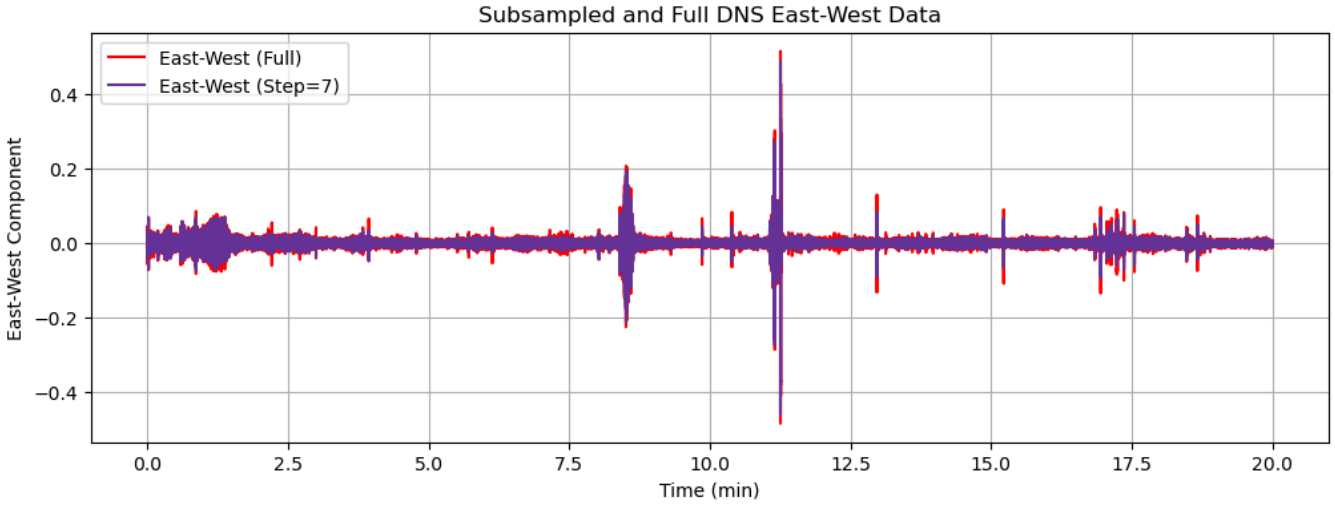

Subsampled and Full DNS Z-Component Data
Z (Full)
Z (Step=7)
Z Component
Time (min)

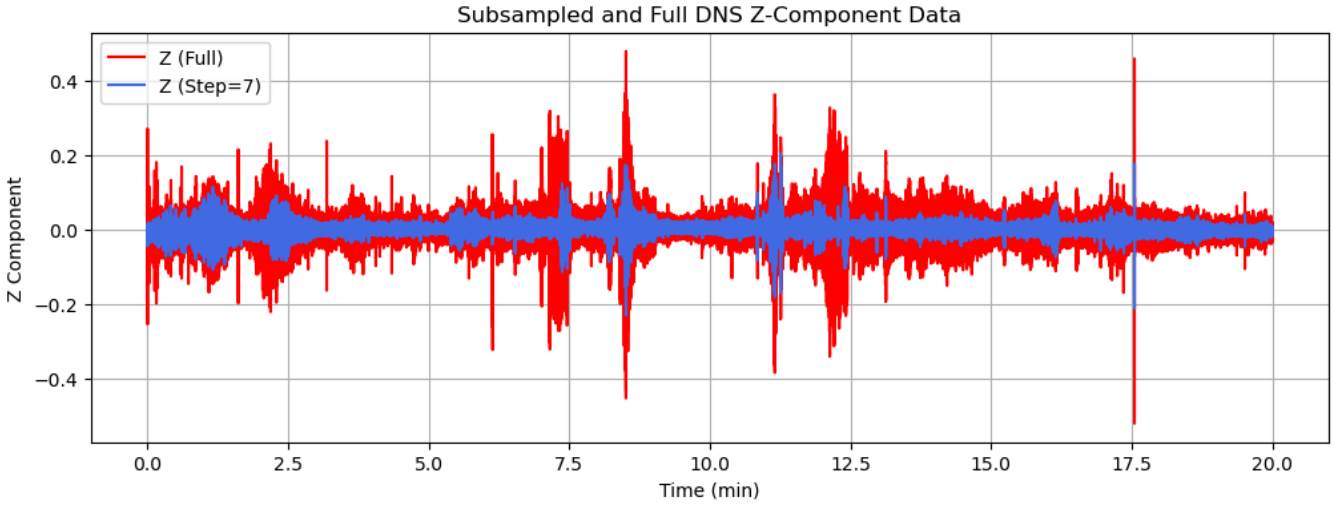

Subsampled and Full DNS North-South Data
North-South (Full)
North-South (Step=7)
North-South Component
Time (min)

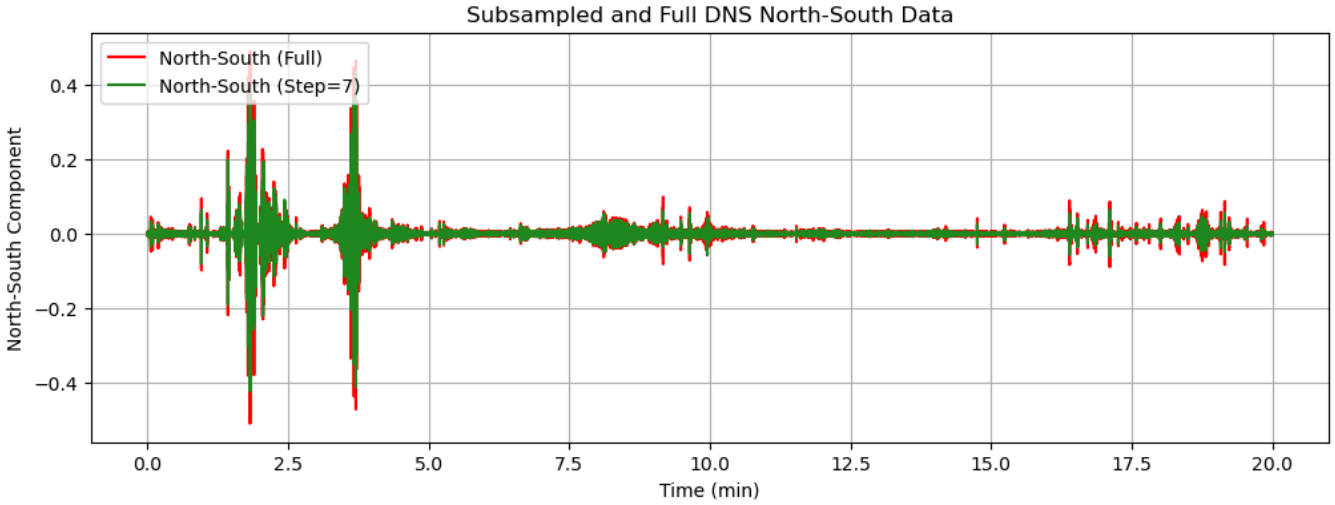

Subsampled and Full DNS East-West Data
East-West (Full)
East-West (Step=7)
East-West Component
Time (min)

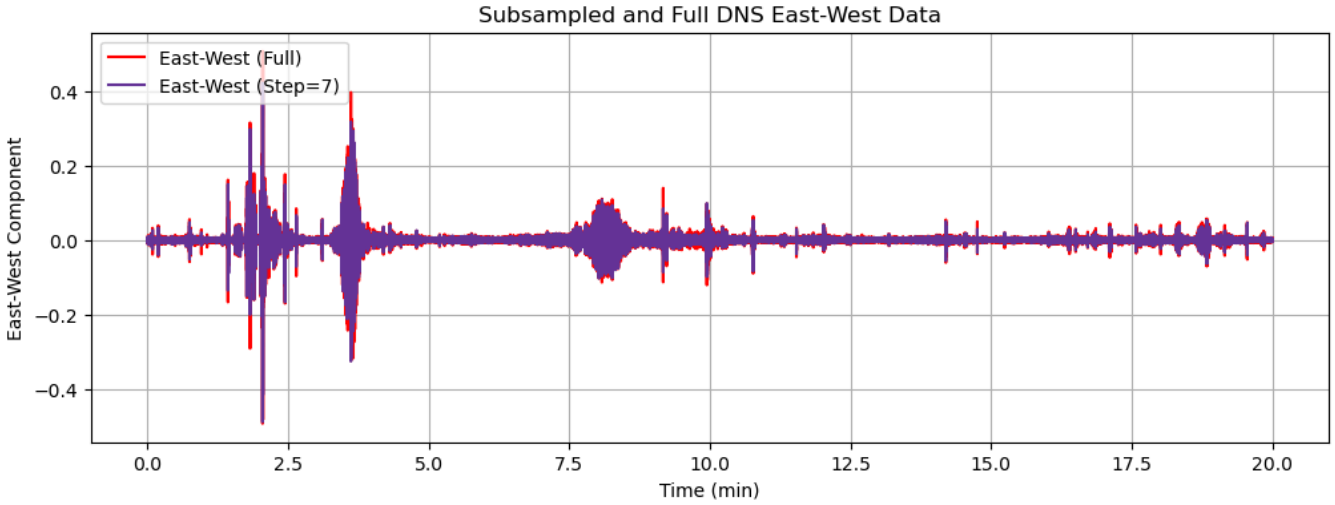

Subsampled and Full DNS Z-Component Data
Z (Full)
Z (Step=7)
Z Component
Time (min)

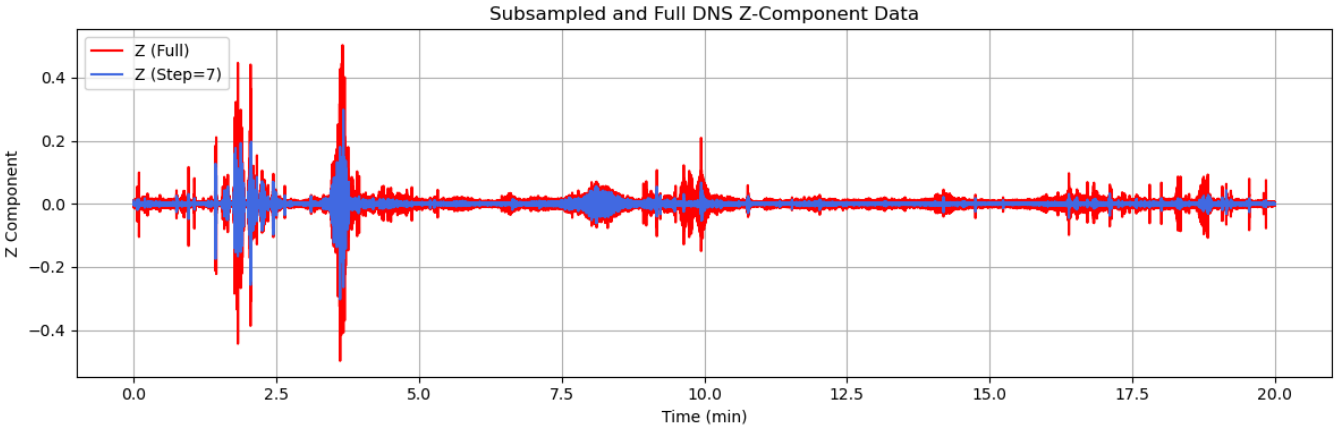

Subsampled and Full DNS North-South Data
North-South (Full)
North-South (Step=7)
North-South Component
Time (min)


Subsampled and Full DNS East-West Data
East-West (Full)
East-West (Step=7)
East-West Component
Time (min)


Subsampled and Full DNS Z-Component Data
Z (Full)
Z (Step=7)
Z Component
Time (min)







Comparing percent_check results from scipy.signal.decimate and indexing for y_subsampled

- There were a similar percentage of points within the error tolerance when using the scipy.signal.decimate method compared to the indexing method for subsampling
- For later cells, the scipy.signal.decimate method will be used as it includes a built-in anti-aliasing filter

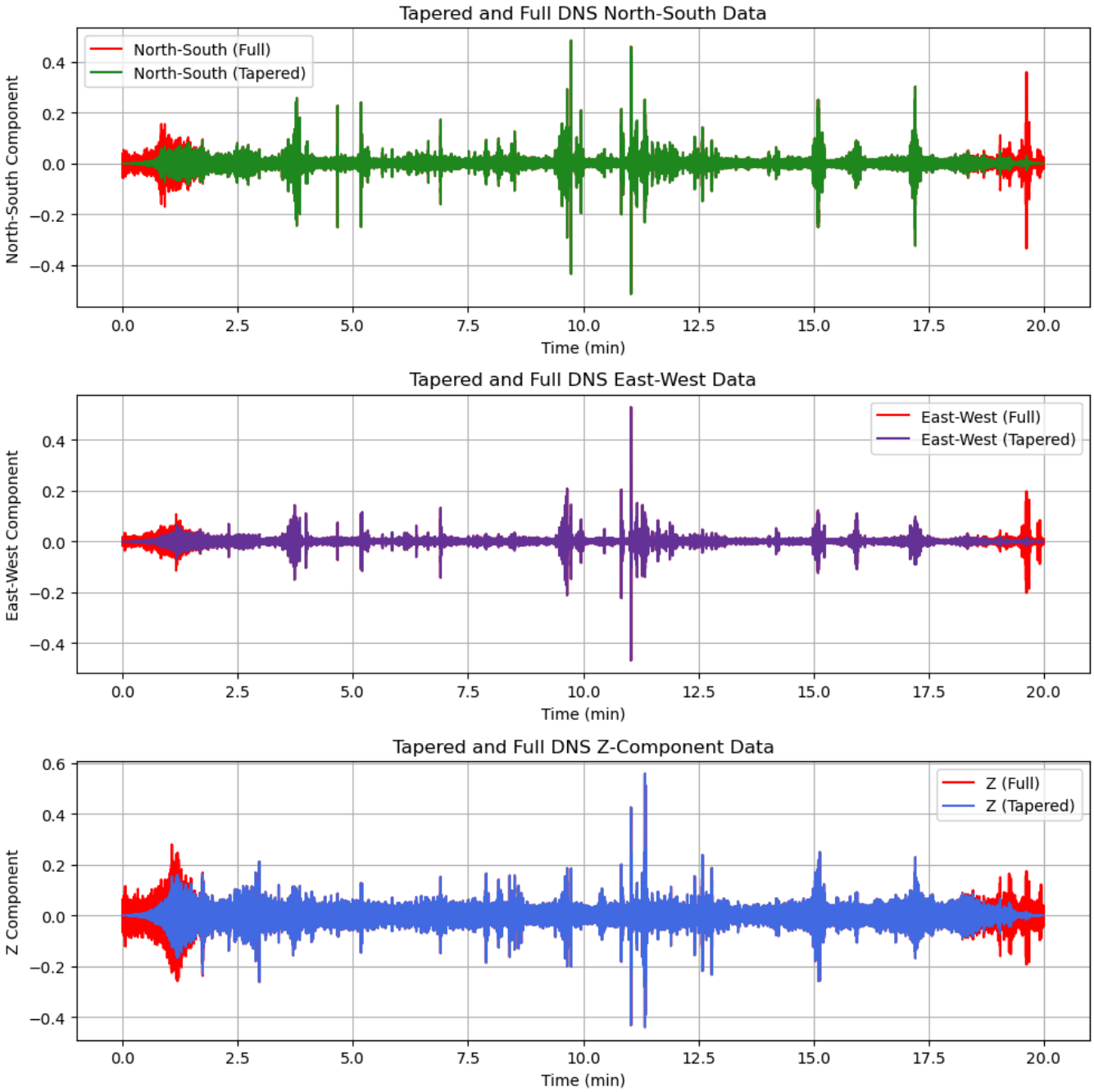
---

Tapering the Time Series

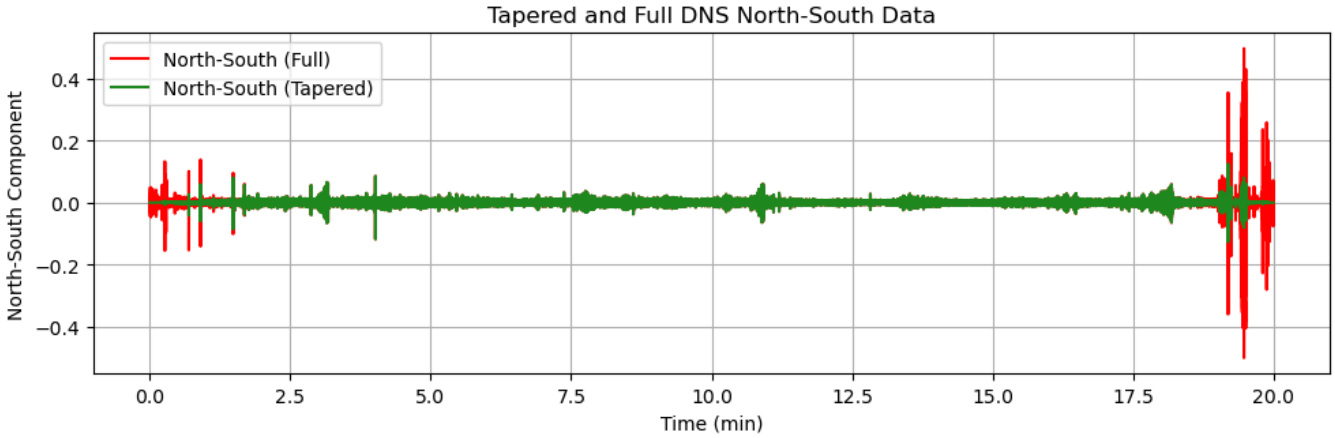
Notes:

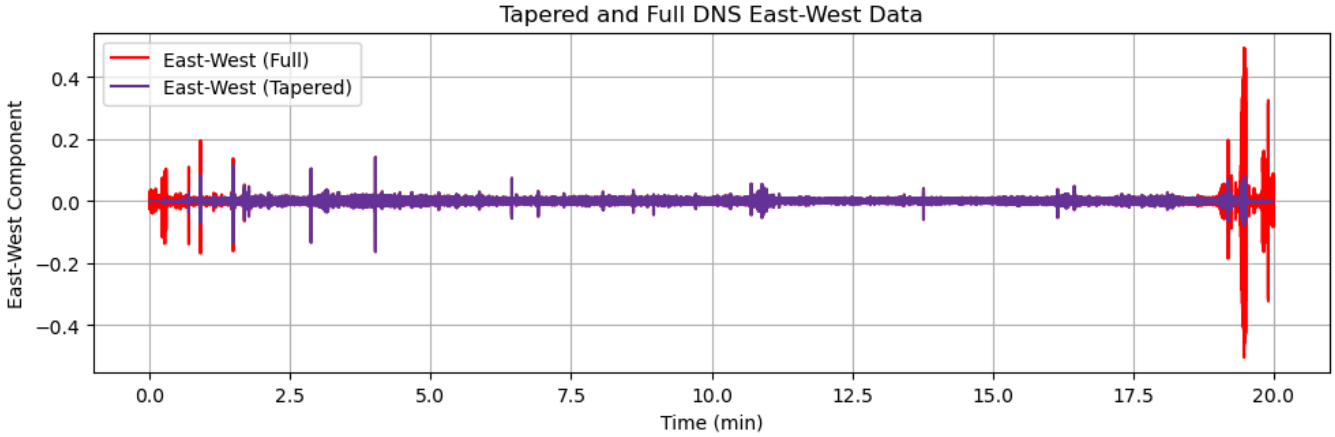
- a Hann window/raised cosine was used to taper the DNS time series
- the data is tapered starting at 10% from the edges (this percentage can be adjusted)

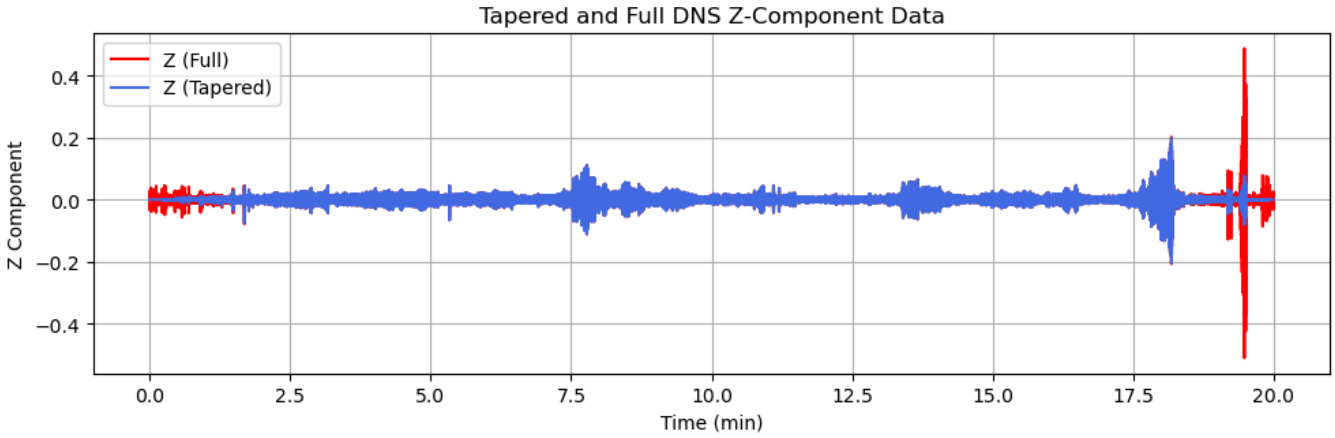
`dataframe: df`



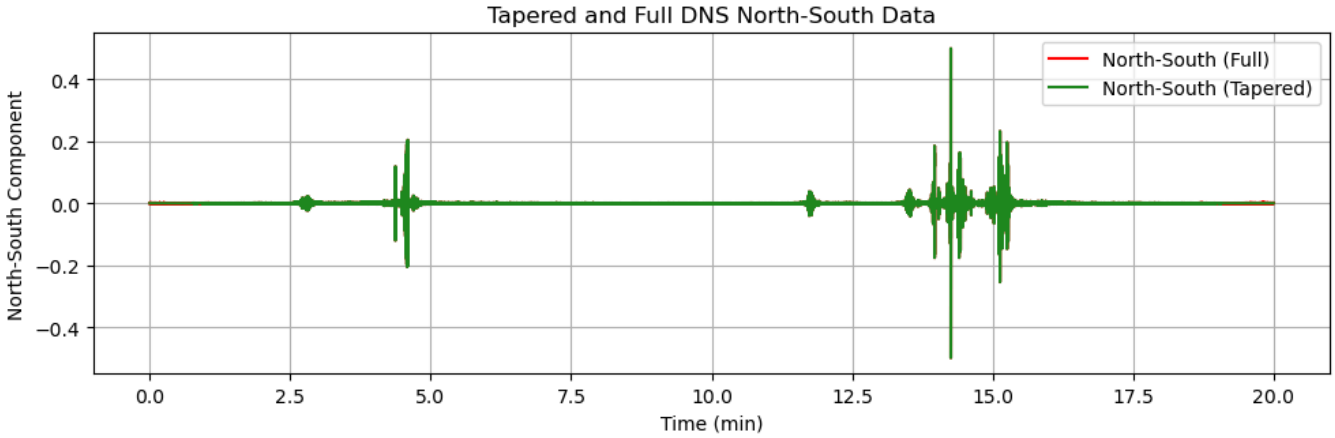
`dataframe: df2`

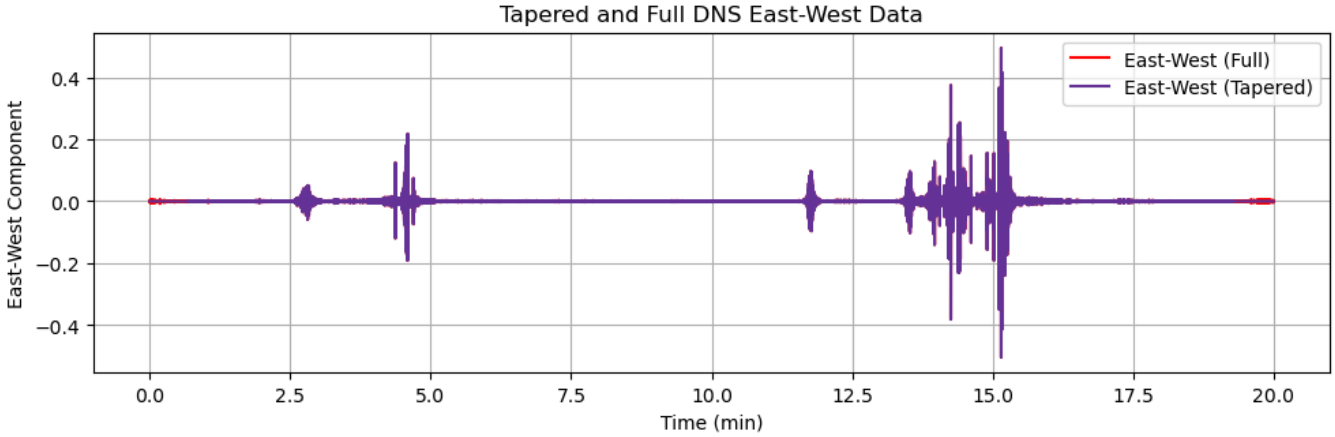


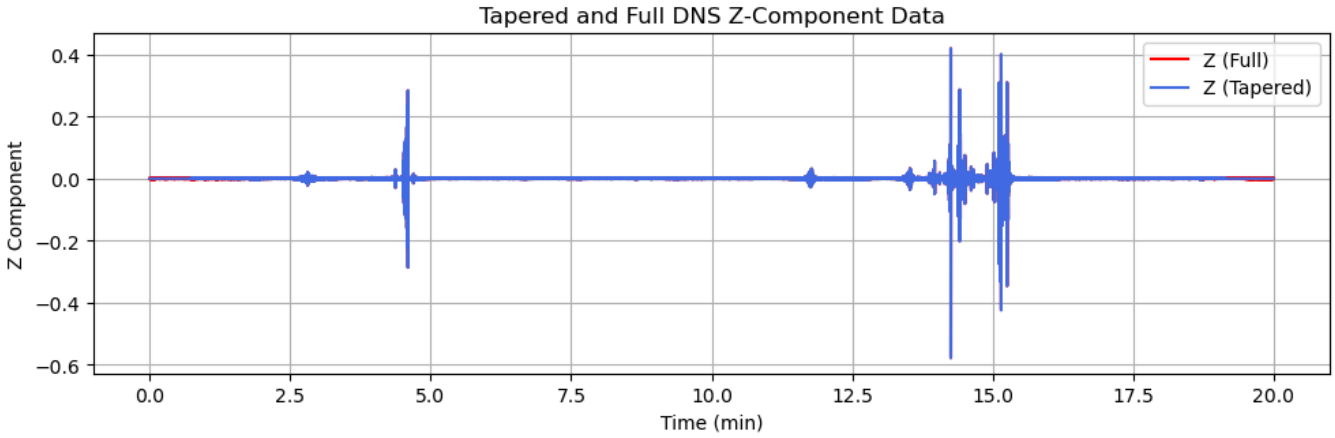




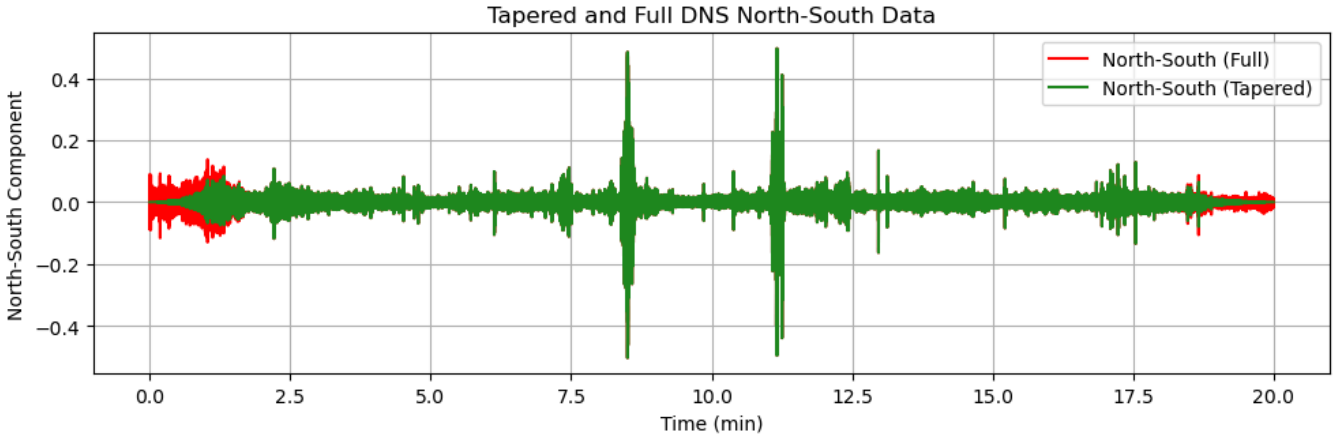
dataframe: df3

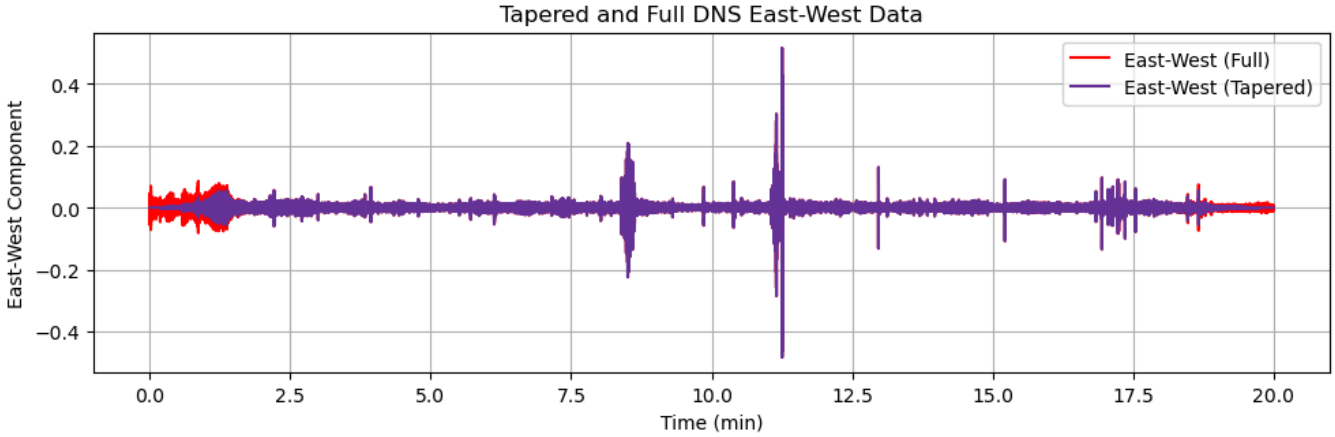


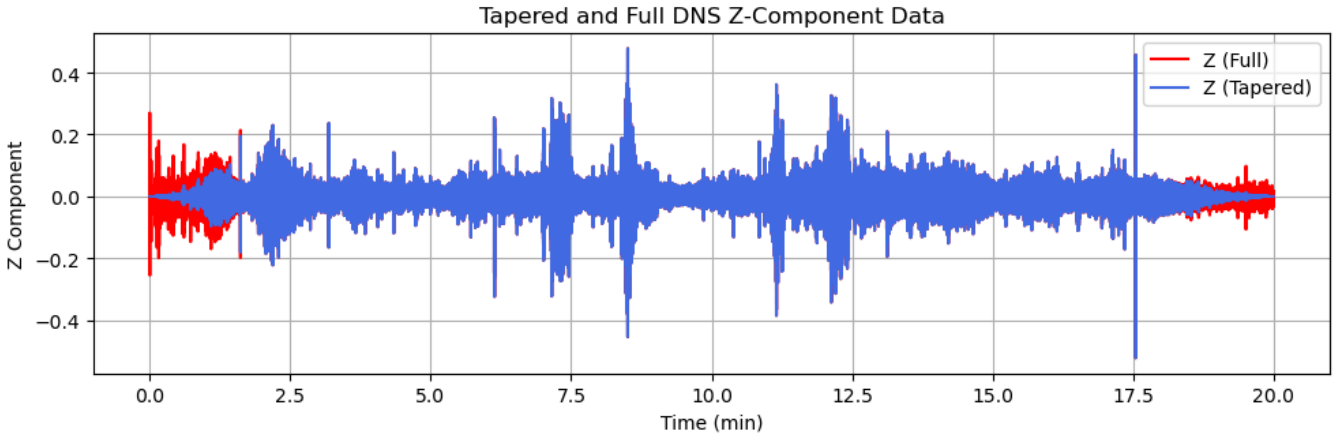




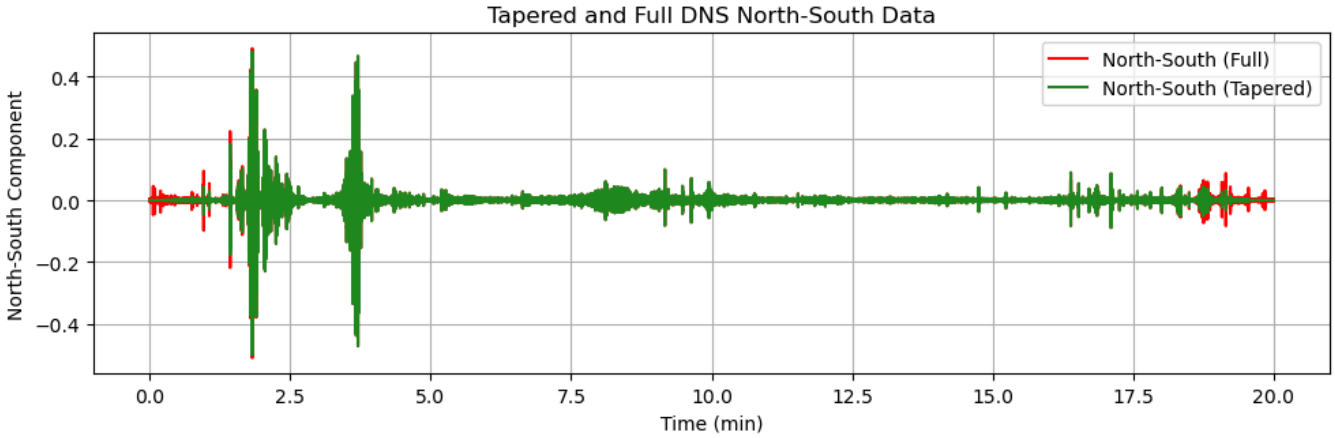
dataframe: df4

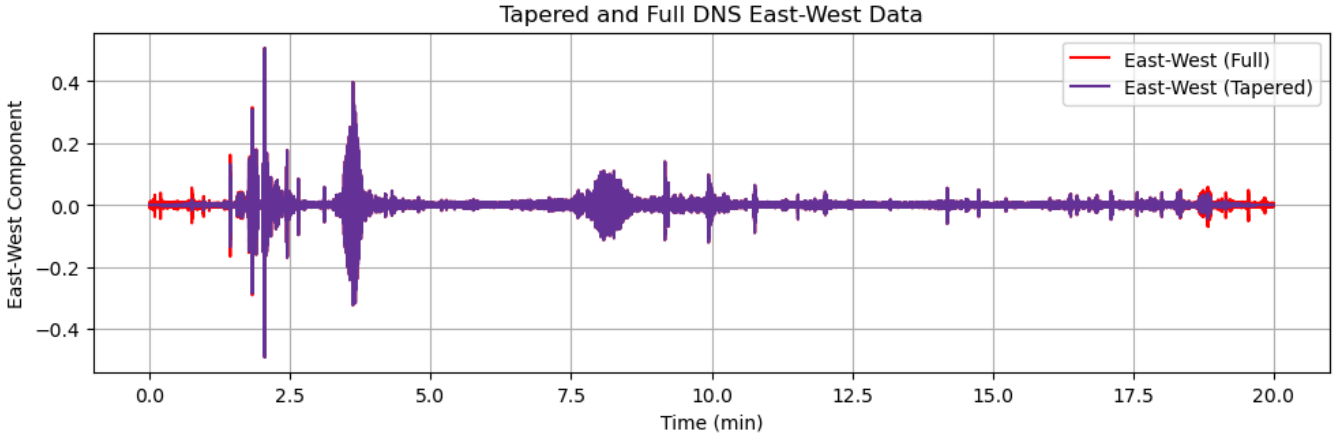


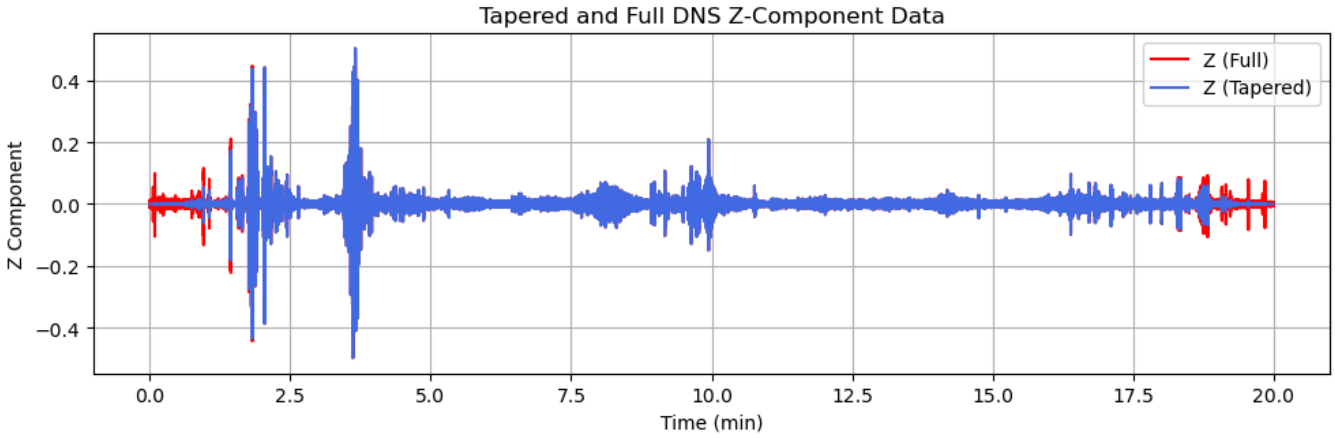




dataframe: df5







'percent_check_tapered' for tapered time series:

- a tapering version of the 'percent_check' function that was used for subsampling
- based on tapering starting at 10% from the data edges

NS_taper_results_1 (df): {'padding': 0, 'tolerance': '5.0%', 'passes': True, 'points_within_t
olerance': ' 99.9220%', 'max_normalized_error': ' 33.0174%', 'mean_abs_error': ' 0.0012', 'ma
e_as_%_of_std': ' 7.8978%', 'num_points': 1228800, 'data_range': ' 1.0000'}
EW_taper_results_1 (df): {'padding': 0, 'tolerance': '5.0%', 'passes': True, 'points_within_t
olerance': ' 99.9718%', 'max_normalized_error': ' 18.6478%', 'mean_abs_error': ' 0.0007', 'ma
e_as_%_of_std': ' 7.5137%', 'num_points': 1228800, 'data_range': ' 1.0000'}
Z_taper_results_1 (df): {'padding': 0, 'tolerance': '5.0%', 'passes': True, 'points_within_to
lerance': ' 99.6984%', 'max_normalized_error': ' 17.8382%', 'mean_abs_error': ' 0.0021', 'mae
_as_%_of_std': ' 9.2318%', 'num_points': 1228800, 'data_range': ' 1.0000'}

NS_taper_results_1 (df2): {'padding': 0, 'tolerance': '5.0%', 'passes': True, 'points_within_
tolerance': ' 99.7722%', 'max_normalized_error': ' 42.0812%', 'mean_abs_error': ' 0.0008', 'm
ae_as_%_of_std': ' 7.9139%', 'num_points': 1228800, 'data_range': ' 1.0000'}
EW_taper_results_1 (df2): {'padding': 0, 'tolerance': '5.0%', 'passes': True, 'points_within_
tolerance': ' 99.6724%', 'max_normalized_error': ' 42.4065%', 'mean_abs_error': ' 0.0009', 'm
ae_as_%_of_std': ' 8.0063%', 'num_points': 1228800, 'data_range': ' 1.0000'}
Z_taper_results_1 (df2): {'padding': 0, 'tolerance': '5.0%', 'passes': True, 'points_within_t
olerance': ' 99.9150%', 'max_normalized_error': ' 43.0695%', 'mean_abs_error': ' 0.0006', 'ma
e_as_%_of_std': ' 4.5780%', 'num_points': 1228800, 'data_range': ' 1.0000'}

NS_taper_results_1 (df3): {'padding': 0, 'tolerance': '5.0%', 'passes': True, 'points_within_
tolerance': ' 100.0000%', 'max_normalized_error': ' 0.2580%', 'mean_abs_error': ' 0.0000', 'm
ae_as_%_of_std': ' 0.5296%', 'num_points': 1228800, 'data_range': ' 1.0000'}
EW_taper_results_1 (df3): {'padding': 0, 'tolerance': '5.0%', 'passes': True, 'points_within_
tolerance': ' 100.0000%', 'max_normalized_error': ' 0.7747%', 'mean_abs_error': ' 0.0001', 'm
ae_as_%_of_std': ' 0.7245%', 'num_points': 1228800, 'data_range': ' 1.0000'}
Z_taper_results_1 (df3): {'padding': 0, 'tolerance': '5.0%', 'passes': True, 'points_within_t
olerance': ' 100.0000%', 'max_normalized_error': ' 0.3323%', 'mean_abs_error': ' 0.0000', 'ma
e_as_%_of_std': ' 0.9960%', 'num_points': 1228800, 'data_range': ' 1.0000'}

NS_taper_results_1 (df4): {'padding': 0, 'tolerance': '5.0%', 'passes': True, 'points_within_
tolerance': ' 99.9749%', 'max_normalized_error': ' 11.3581%', 'mean_abs_error': ' 0.0010', 'm
ae_as_%_of_std': ' 5.4971%', 'num_points': 1228800, 'data_range': ' 1.0000'}
EW_taper_results_1 (df4): {'padding': 0, 'tolerance': '5.0%', 'passes': True, 'points_within_
tolerance': ' 99.9979%', 'max_normalized_error': ' 7.1650%', 'mean_abs_error': ' 0.0006', 'ma
e_as_%_of_std': ' 6.2506%', 'num_points': 1228800, 'data_range': ' 1.0000'}
Z_taper_results_1 (df4): {'padding': 0, 'tolerance': '5.0%', 'passes': True, 'points_within_t
olerance': ' 99.9240%', 'max_normalized_error': ' 26.9833%', 'mean_abs_error': ' 0.0015', 'ma
e_as_%_of_std': ' 4.5339%', 'num_points': 1228800, 'data_range': ' 1.0000'}

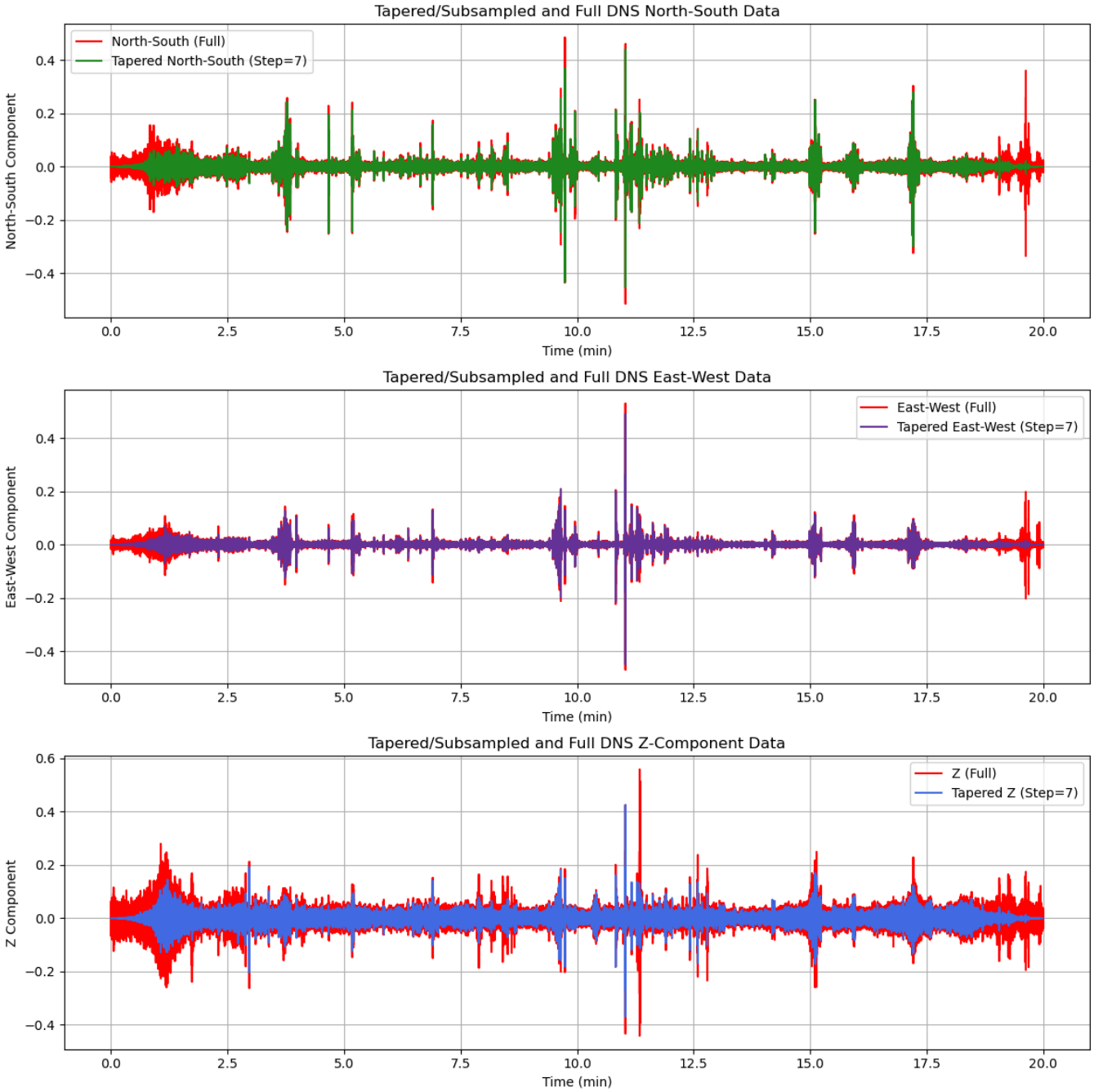
NS_taper_results_1 (df5): {'padding': 0, 'tolerance': '5.0%', 'passes': True, 'points_within_
tolerance': ' 99.9995%', 'max_normalized_error': ' 5.3728%', 'mean_abs_error': ' 0.0002', 'ma
e_as_%_of_std': ' 2.2915%', 'num_points': 1228800, 'data_range': ' 1.0000'}
EW_taper_results_1 (df5): {'padding': 0, 'tolerance': '5.0%', 'passes': True, 'points_within_
tolerance': ' 100.0000%', 'max_normalized_error': ' 4.6122%', 'mean_abs_error': ' 0.0003', 'm
ae_as_%_of_std': ' 2.2257%', 'num_points': 1228800, 'data_range': ' 1.0000'}
Z_taper_results_1 (df5): {'padding': 0, 'tolerance': '5.0%', 'passes': True, 'points_within_t
olerance': ' 99.9945%', 'max_normalized_error': ' 10.3885%', 'mean_abs_error': ' 0.0004', 'ma
e_as_%_of_std': ' 3.2572%', 'num_points': 1228800, 'data_range': ' 1.0000'}

Combining Subsampling and Tapering

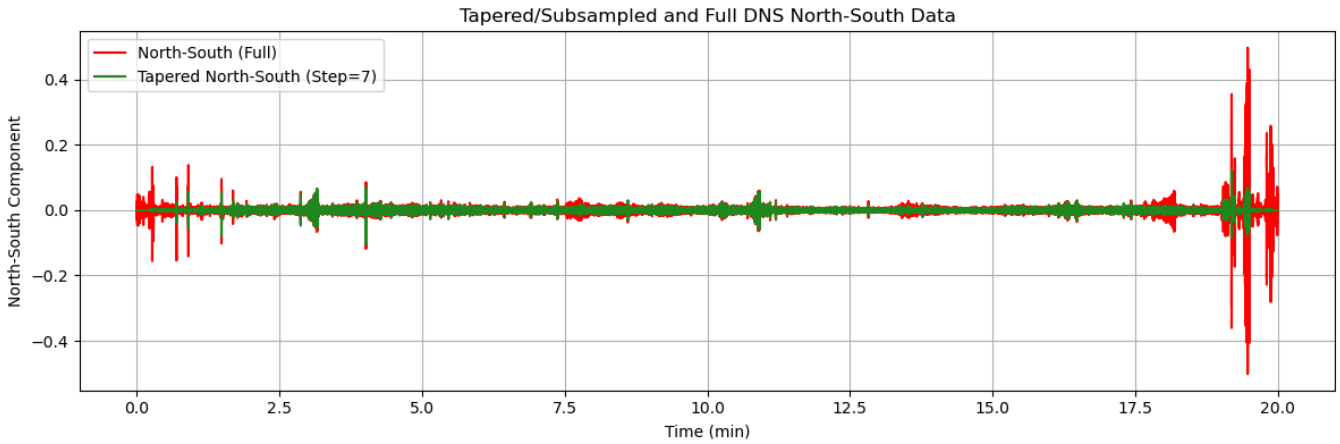
Notes:

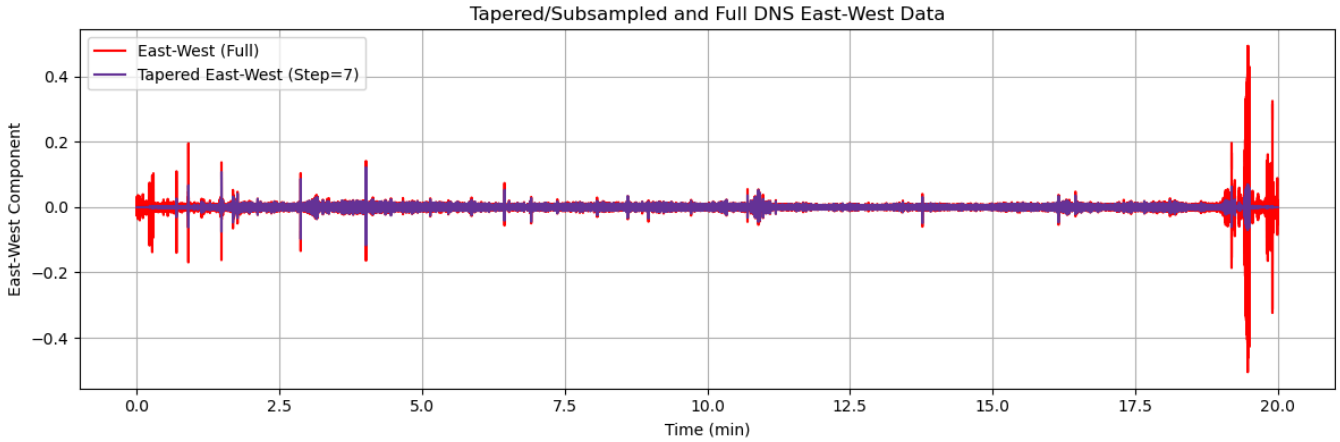
- The full series was tapered before subsampling
- A Hann window was used for tapering
- A step size of 7 was used for subsampling, consistent across the North-South, East-West, and Up-Down (Z) data
    - adjust the step size used for subsampling as needed

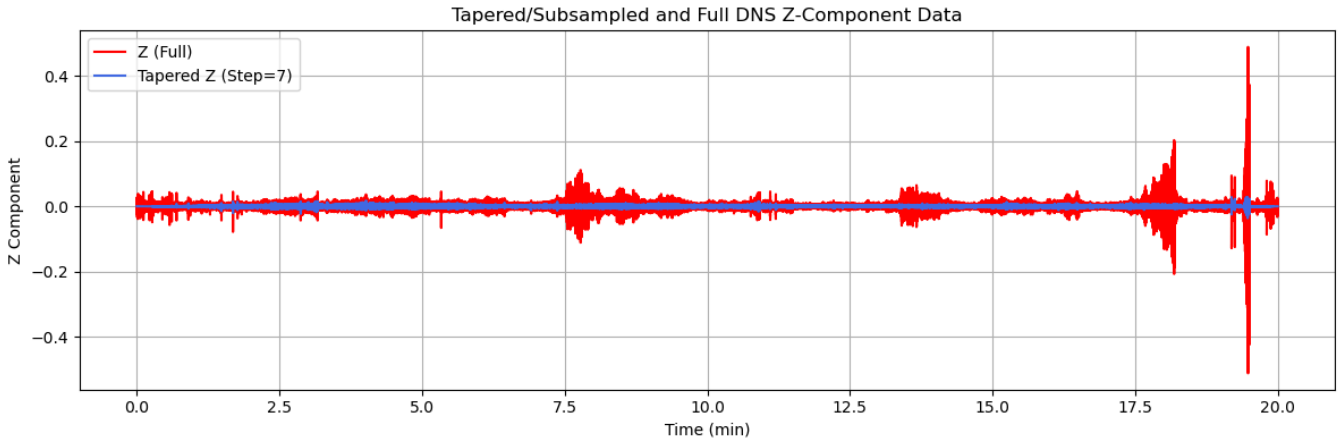
`dataframe: df`



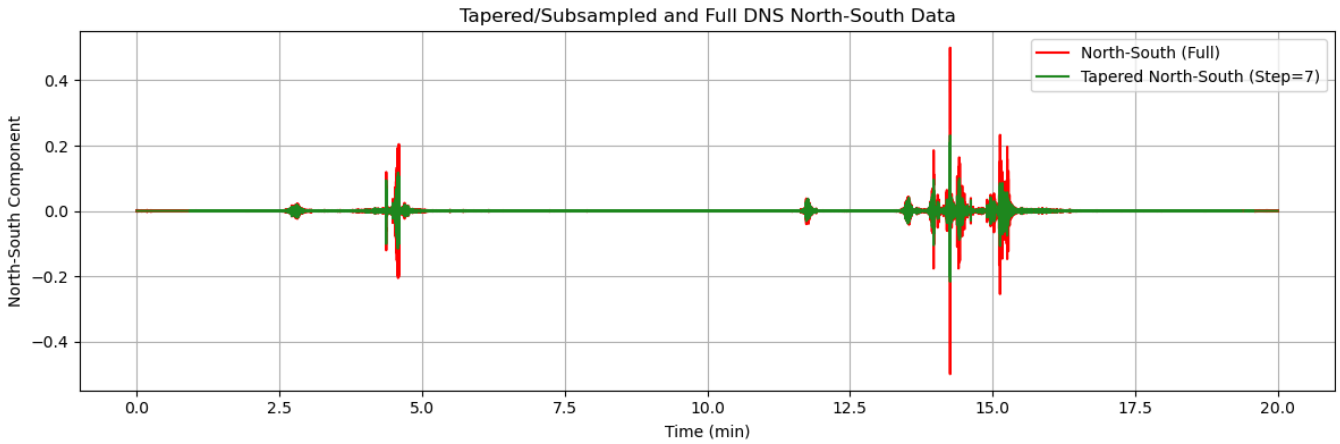
`dataframe: df2`

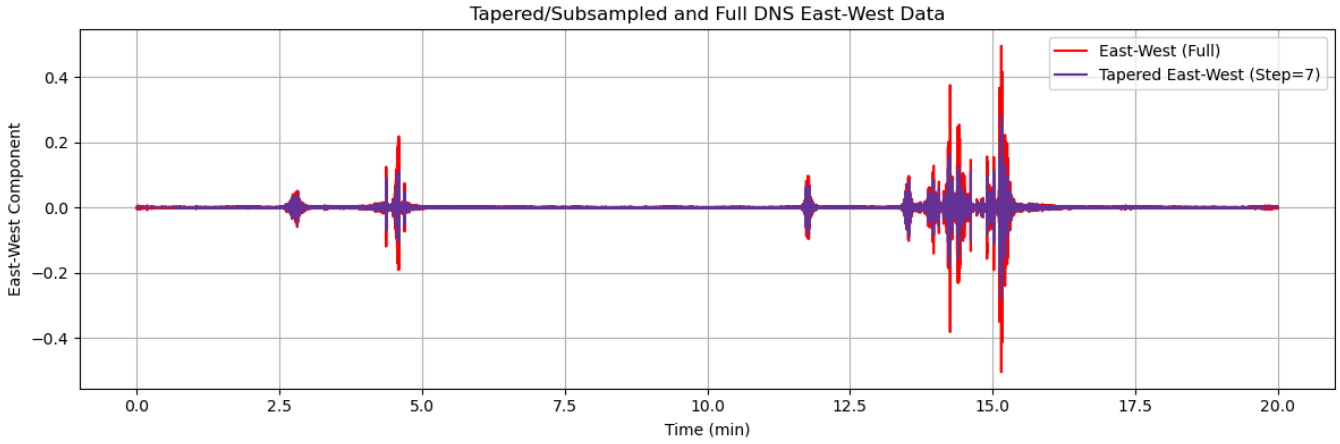


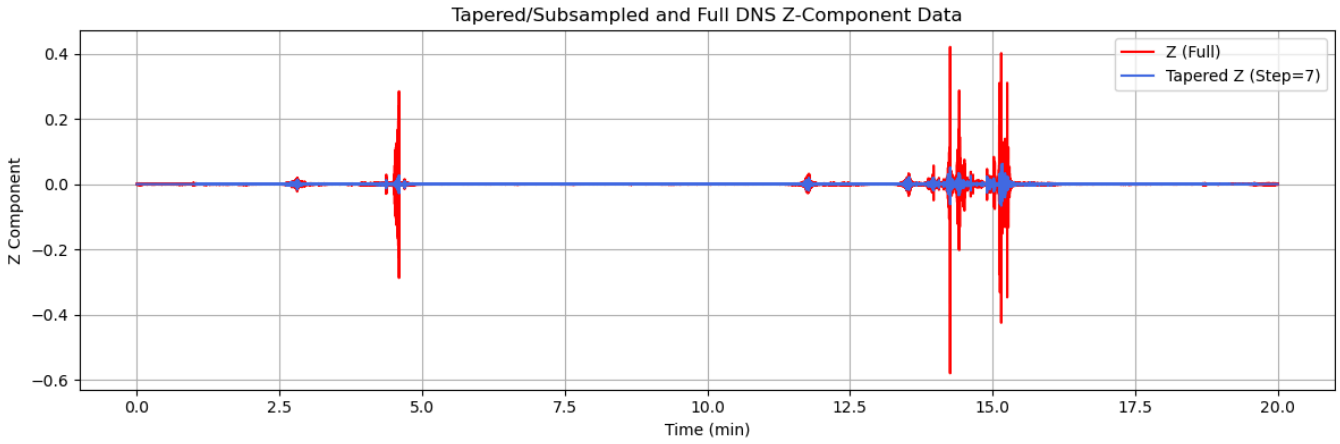




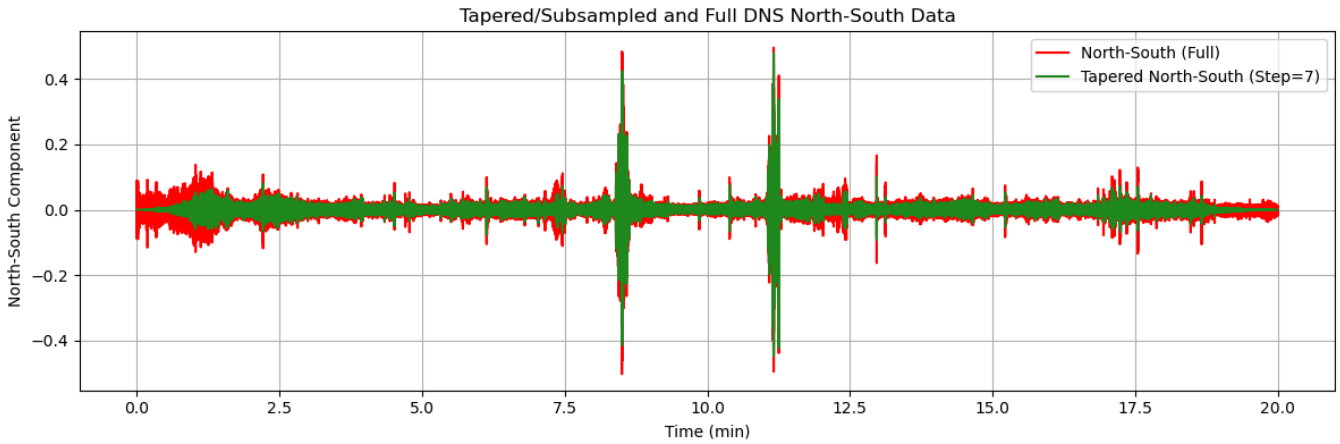
dataframe: df3

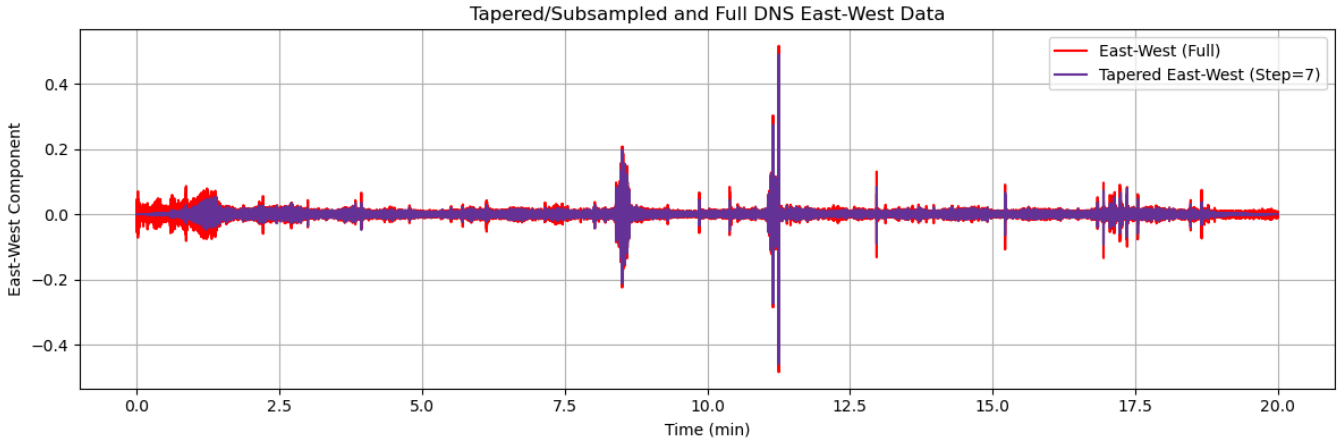


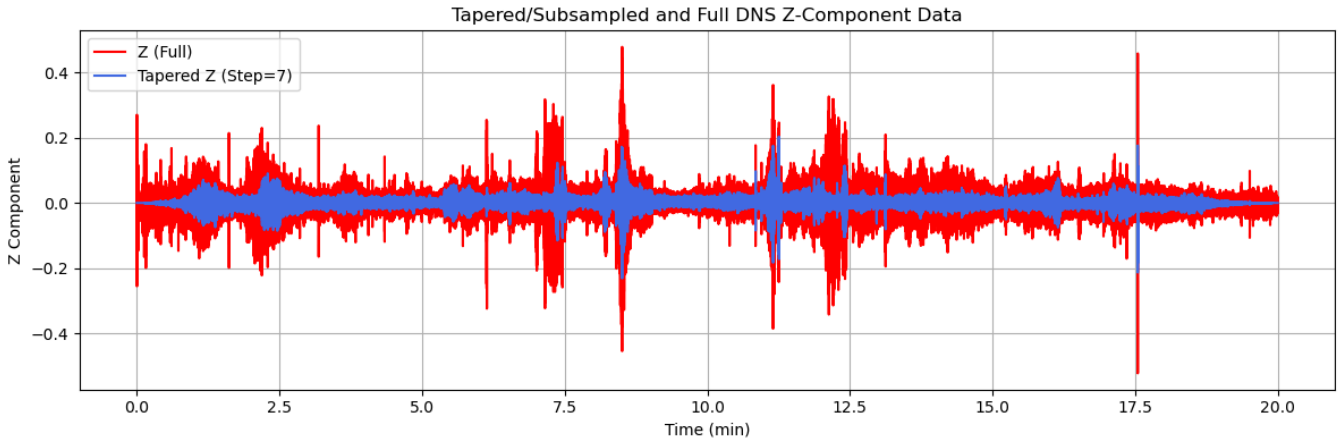




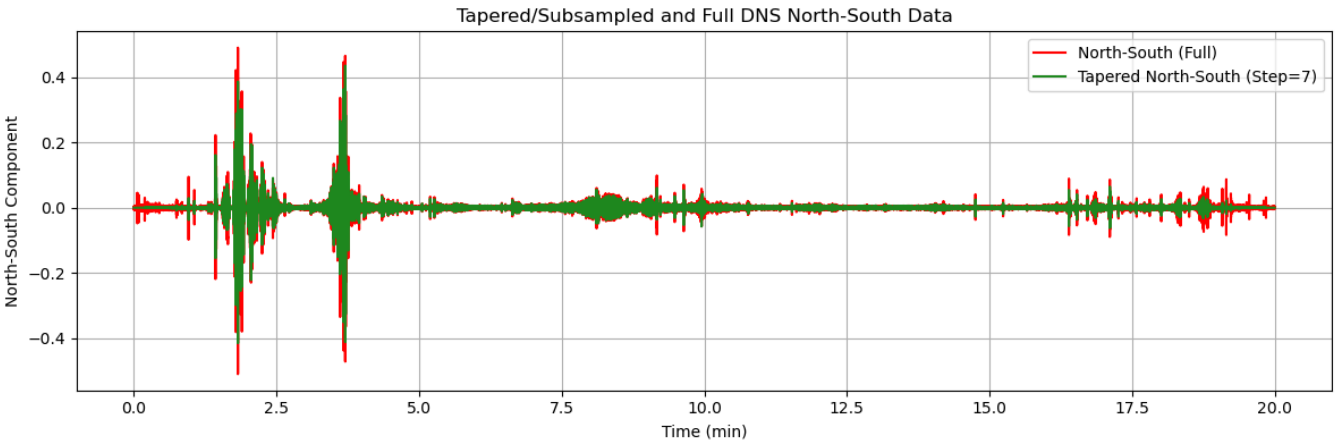
dataframe: df4

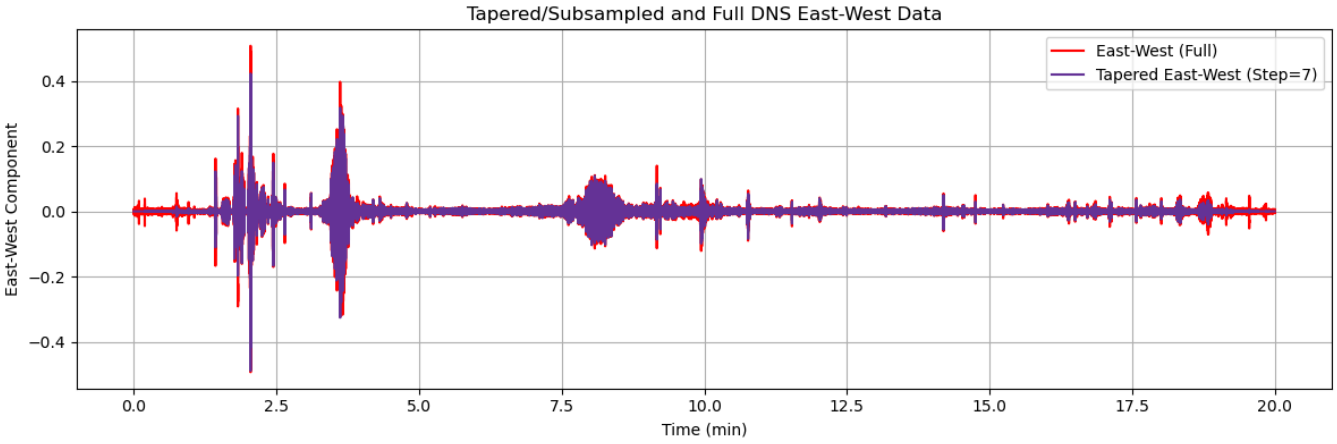


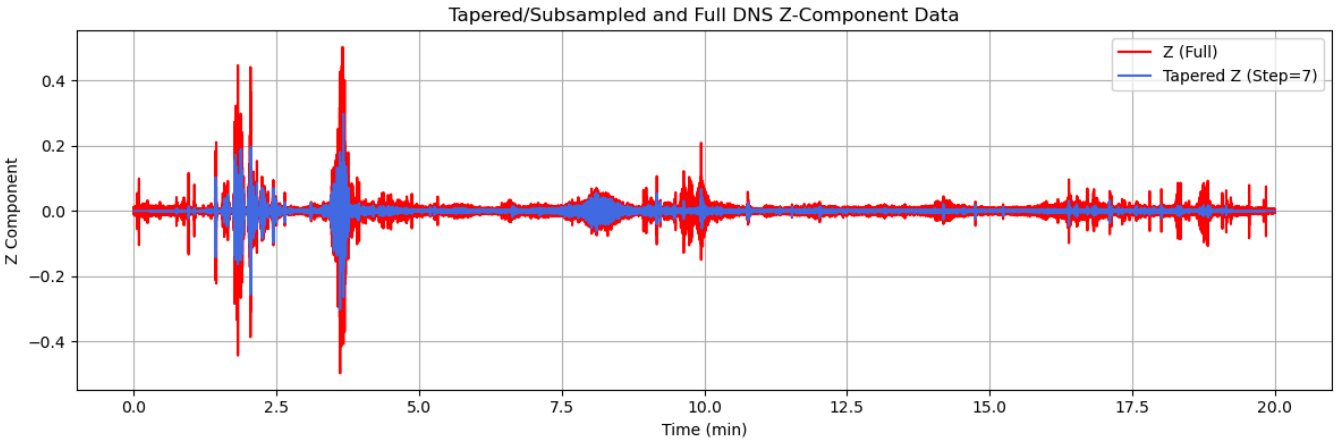




dataframe: df5







Fast Fourier Transform

Notes:

- Applying the fast fourier transform to subsampled and tapered DNS data and plotting the resulting magnitude and phase spectra
  - (commented out phase spectra for now)
  - also defined a coherent gain function in attempt to correct for the amplitude reduction from windowing

Effects of subsampling/tapering: - removes higher frequency content - reduces spectral leakage caused by discontinuities

Coherent gain equation:

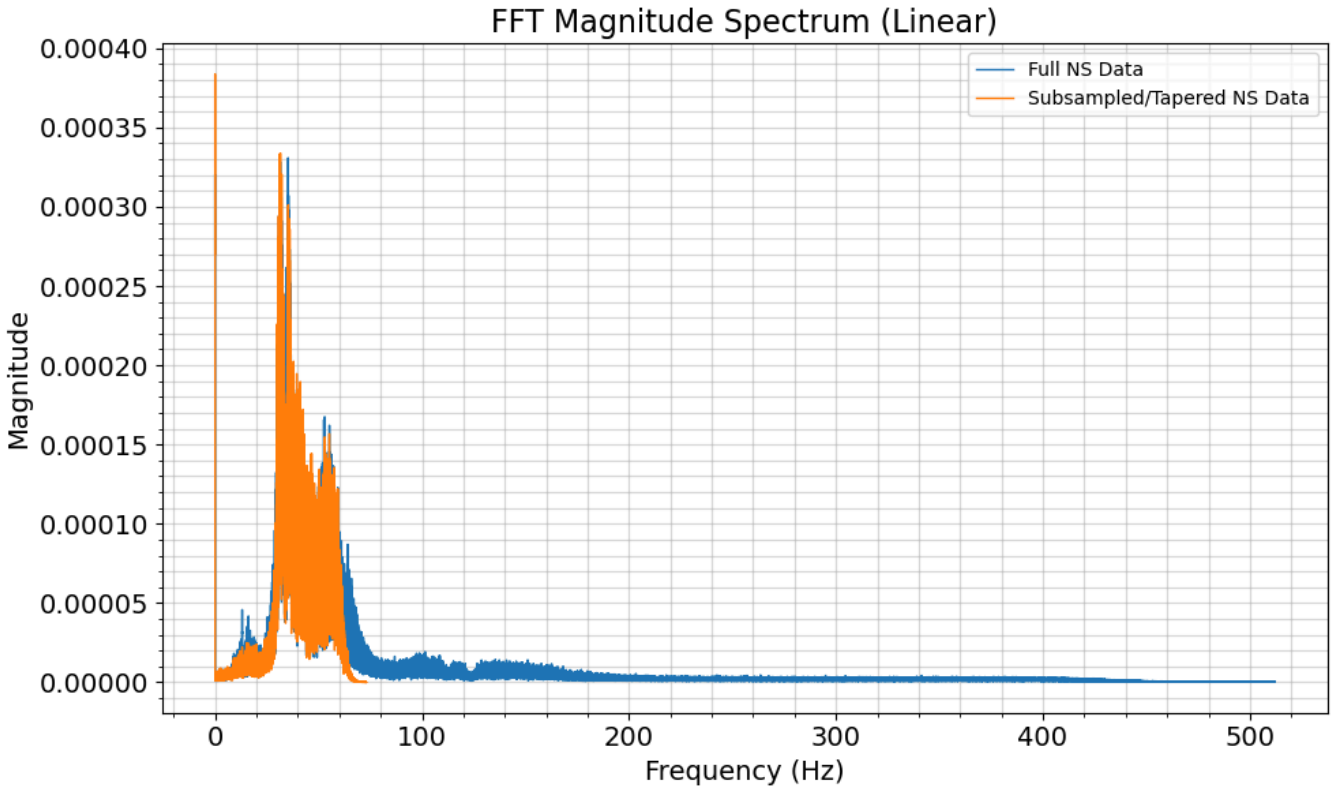
$G_c = \frac{1}{N} \sum_{n=0}^{N-1} \omega[n]$

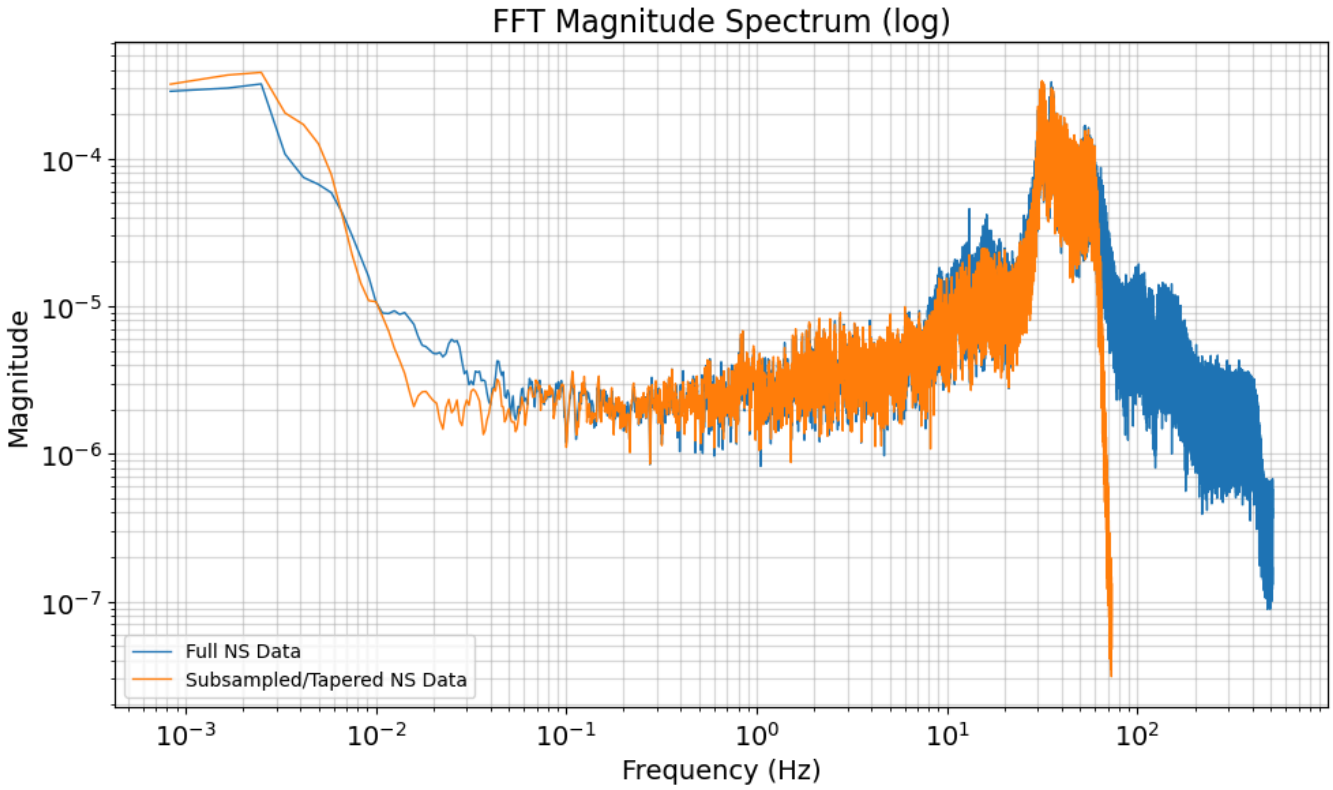
- G_c: coherent gain
- $\omega[n]$: window
- N: window length

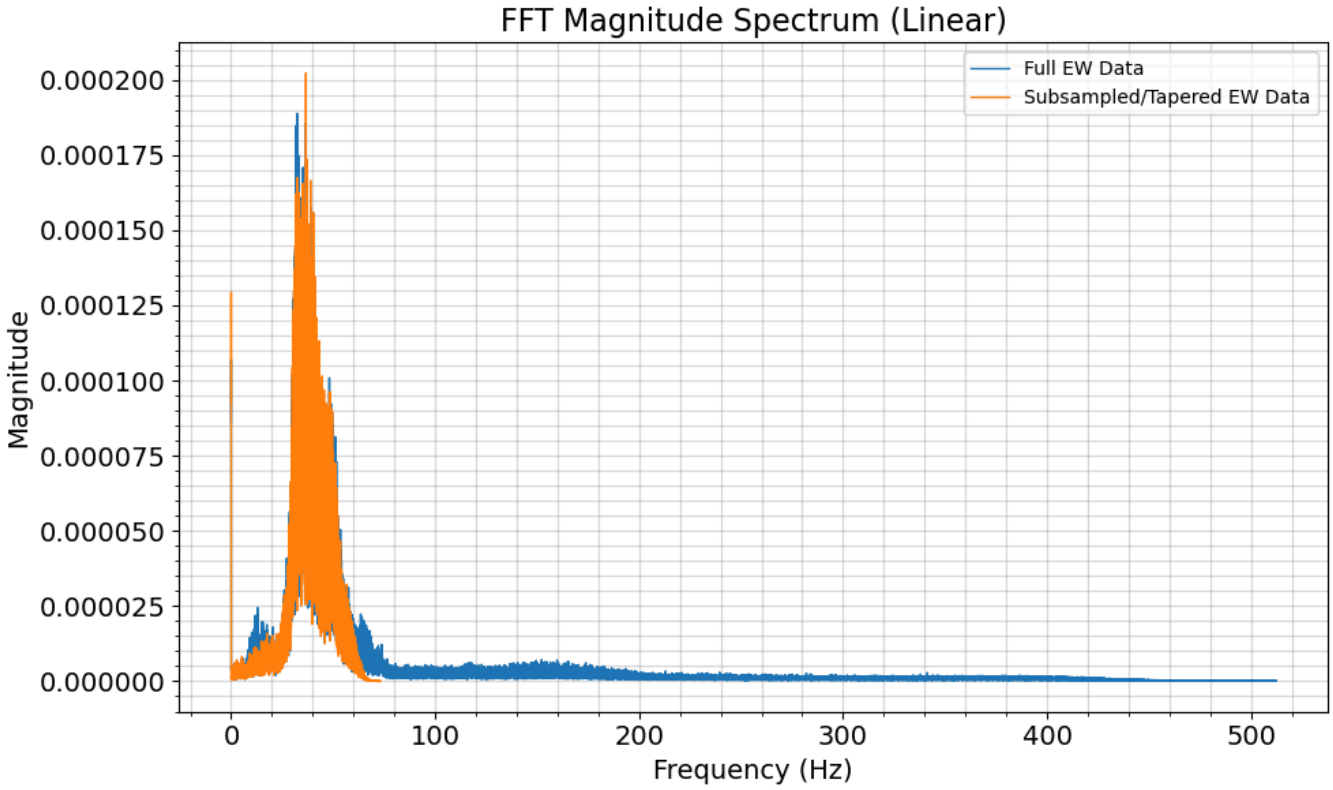
Approach 1: Overlay FFT results for tapered/subsampled data onto single figure with full DNS results

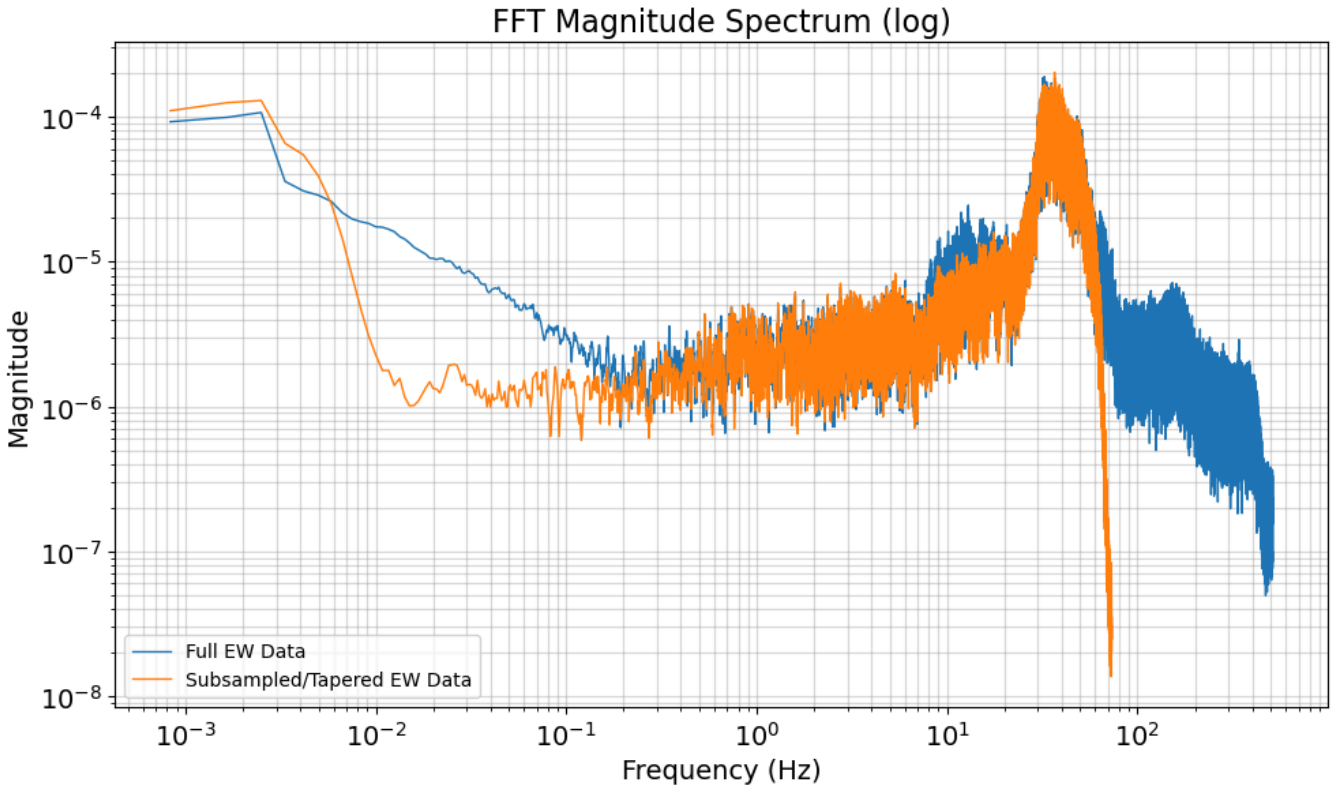
- plot for each dataframe as needed
  - may need to convert subsampled data from arrays to dataframes

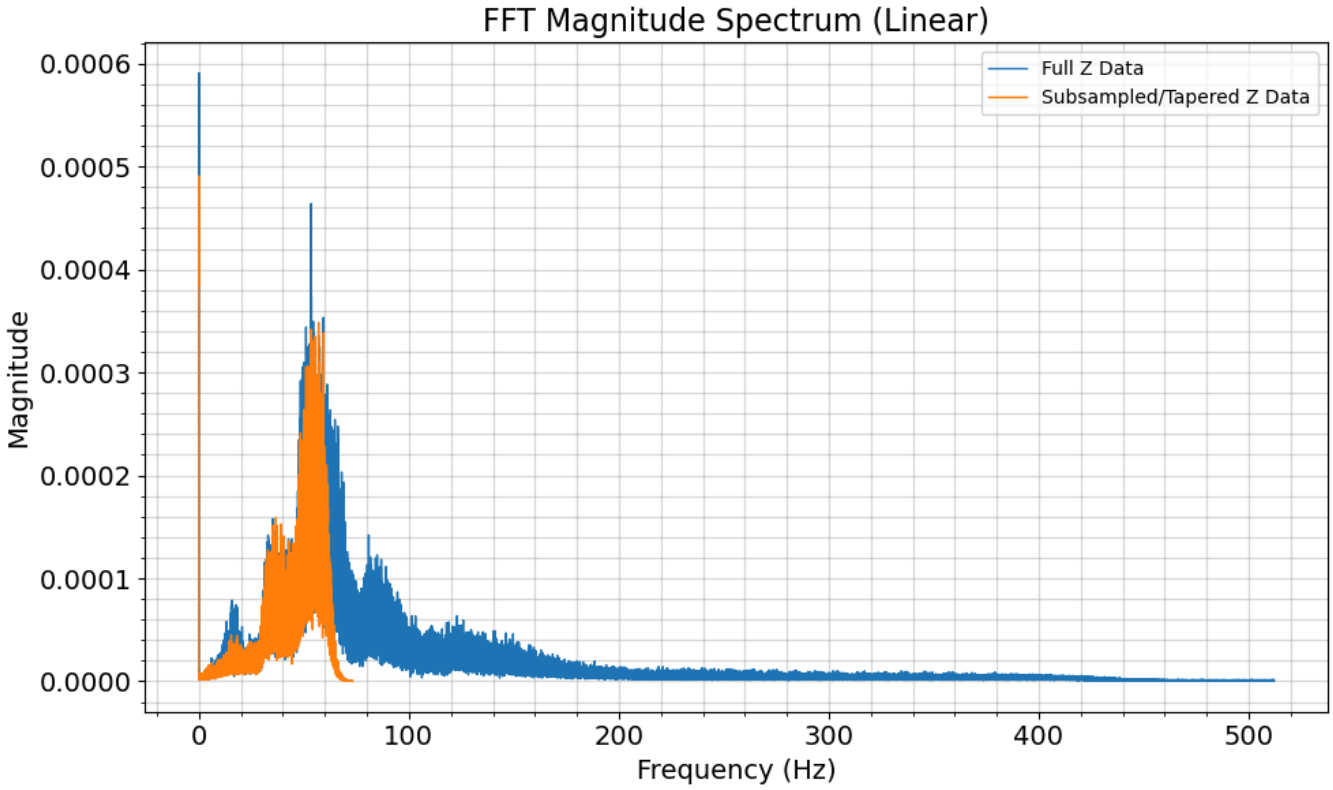
```
dataframe: df
```

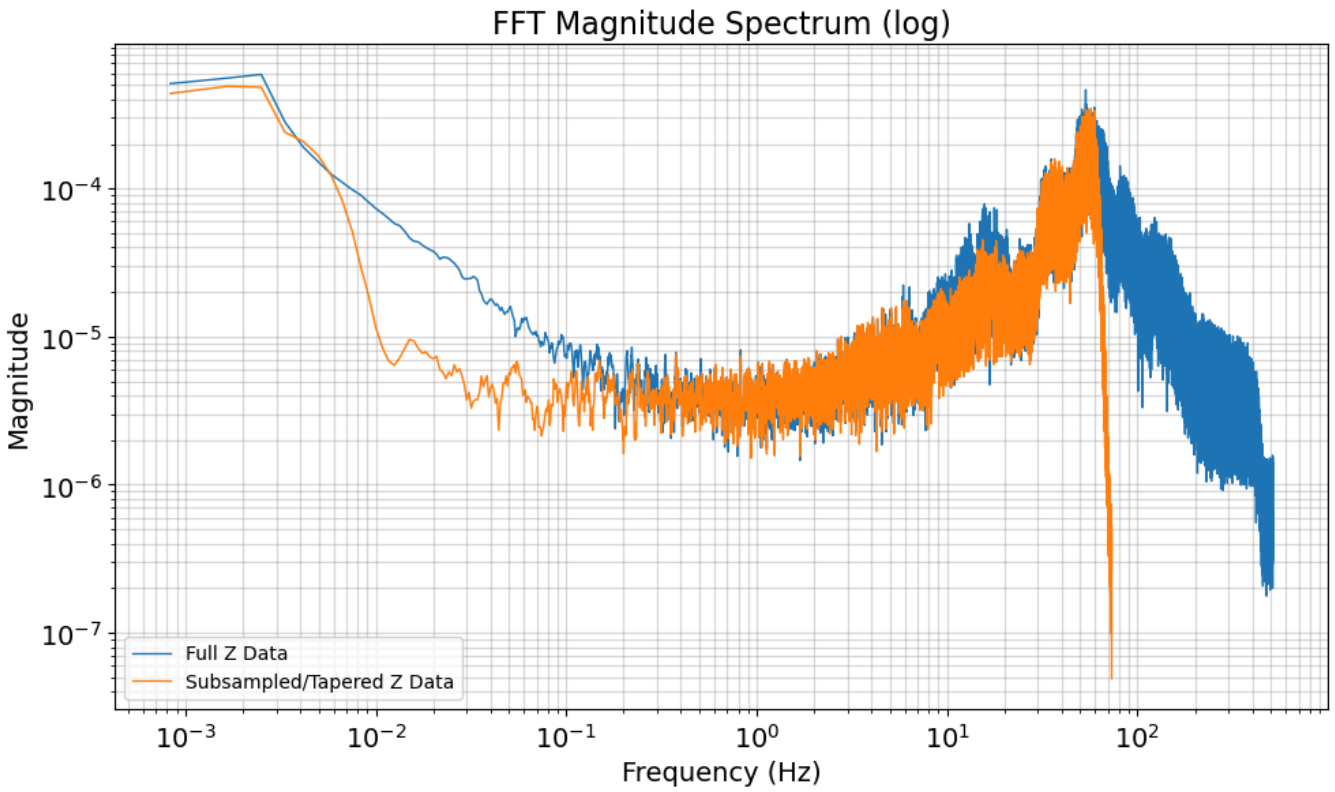



FFT Magnitude Spectrum (log)
Magnitude
Frequency (Hz)
Full NS Data
Subsampled/Tapered NS Data


FFT Magnitude Spectrum (Linear)
Magnitude
Frequency (Hz)
Full EW Data
Subsampled/Tapered EW Data


FFT Magnitude Spectrum (log)
Magnitude
Frequency (Hz)
Full EW Data
Subsampled/Tapered EW Data


FFT Magnitude Spectrum (Linear)
Magnitude
Frequency (Hz)
Full Z Data
Subsampled/Tapered Z Data

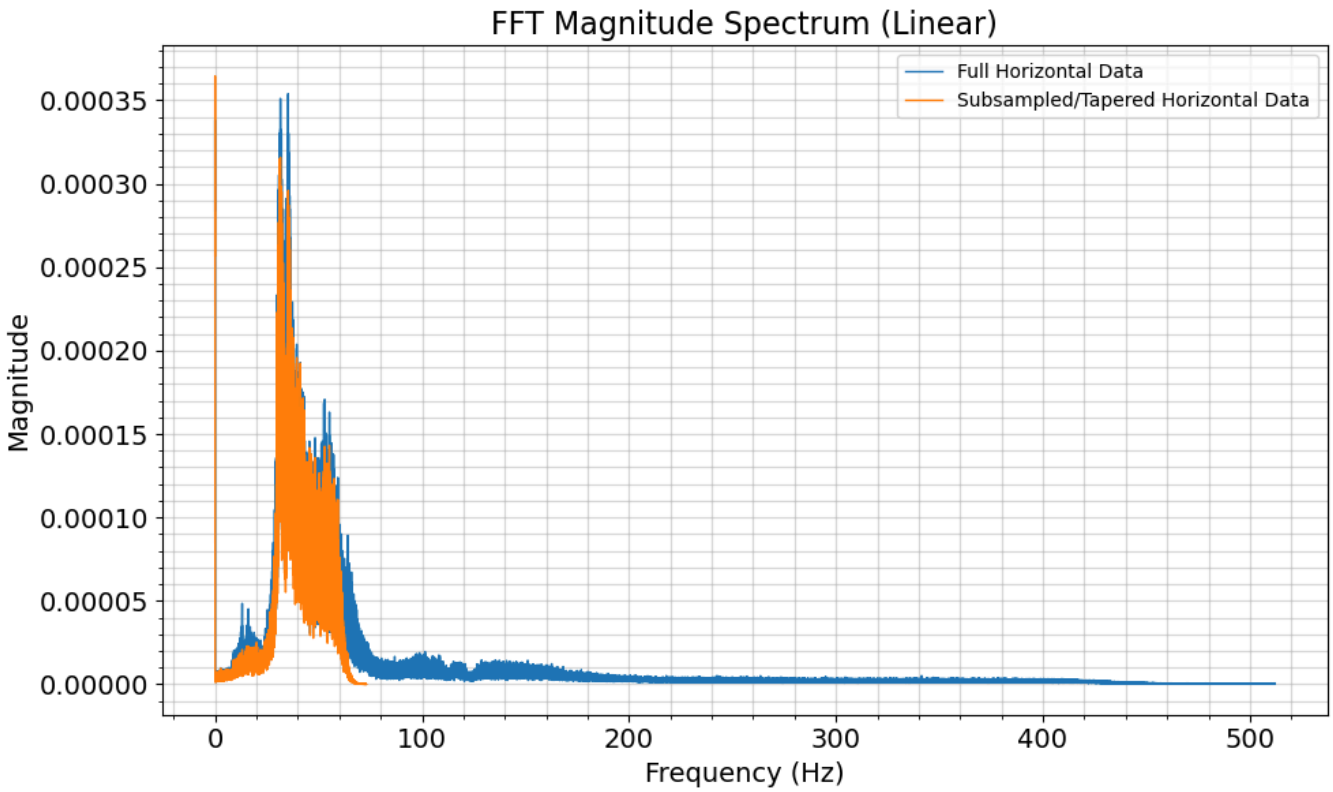


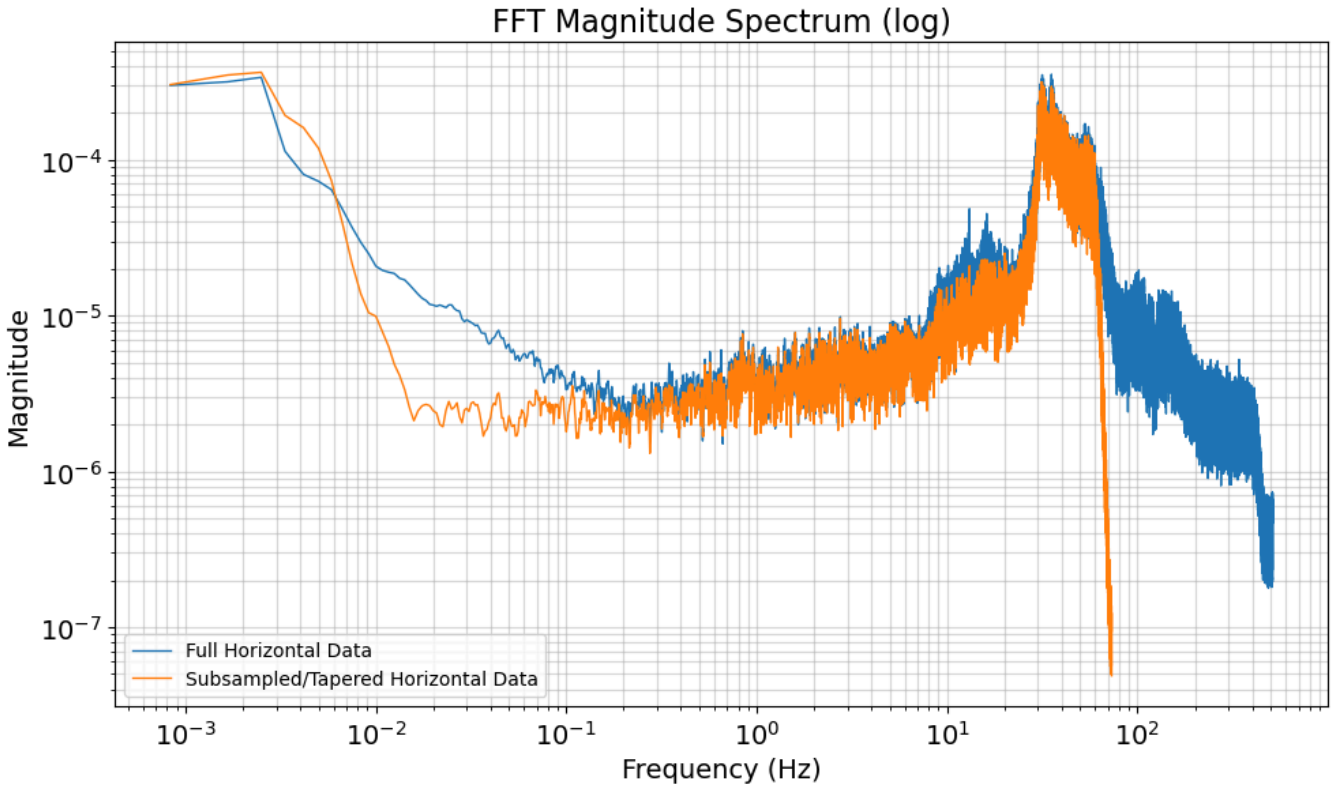
Horizontal and Vertical FFTs

- Goal: Add NS and EW FFTs together to get one set of plots (Horizontal)
- Leave Z FFT plots as is (Vertical), plotted from a previous cell

Horizontal FFT Plots:

```
horizontal FFT for dataframe: df
```





FFT With Moving Window

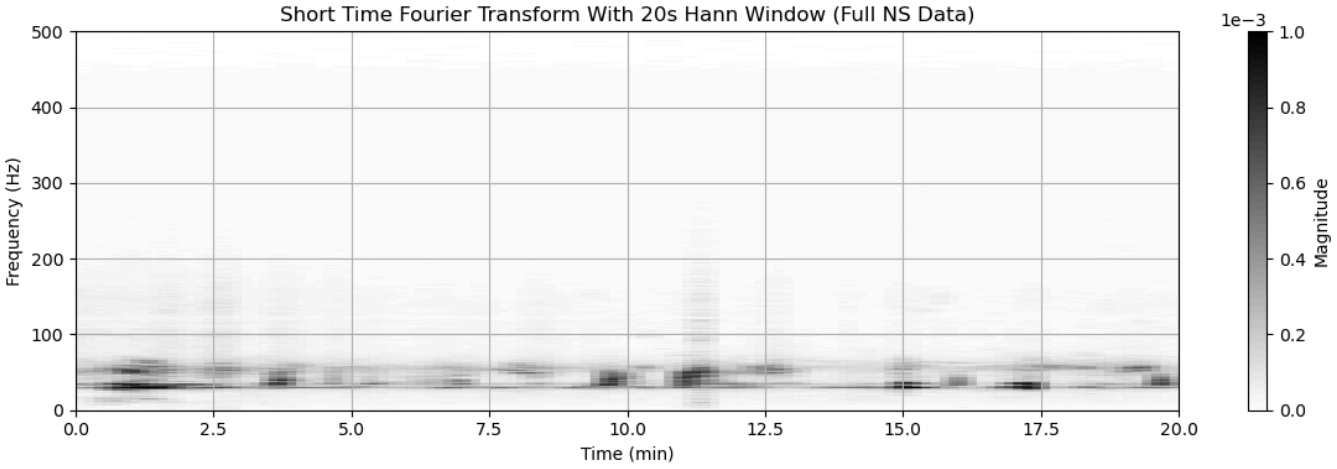
Notes:

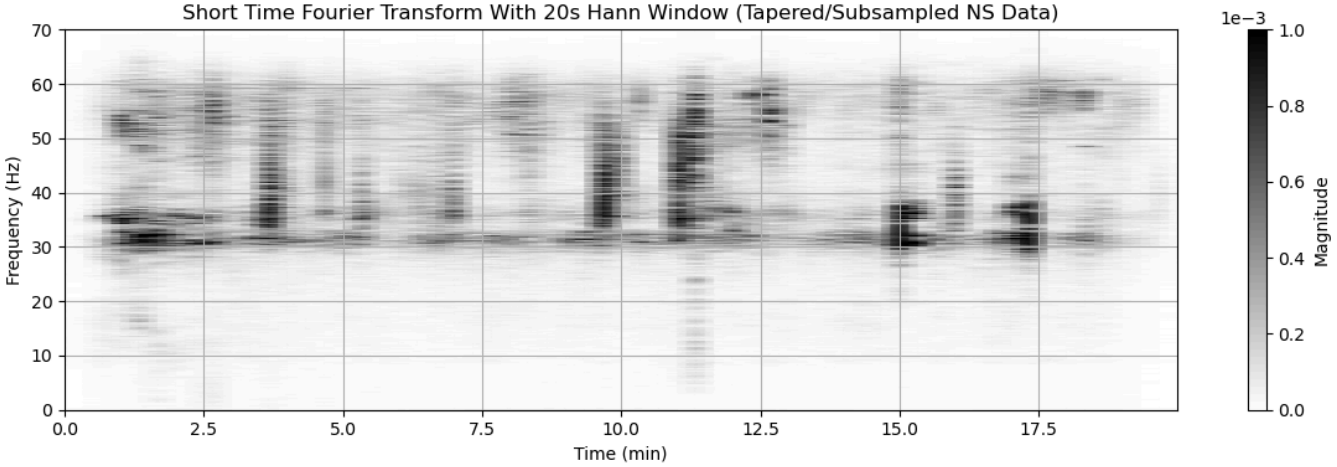
- Goal: apply moving window (20 second intervals)

- Using short time fourier transform from scipy.signal
- Issue with previous version of code: time was not the same dimensions as the STFT result
  - Possible solutions: Plot spectrogram to show frequency content overtime, or plot average magnitude spectrum
- Tried log scale for spectrogram frequency axis, did not display well for most full data plots
- adjust y-axis limits to focus on areas of interest

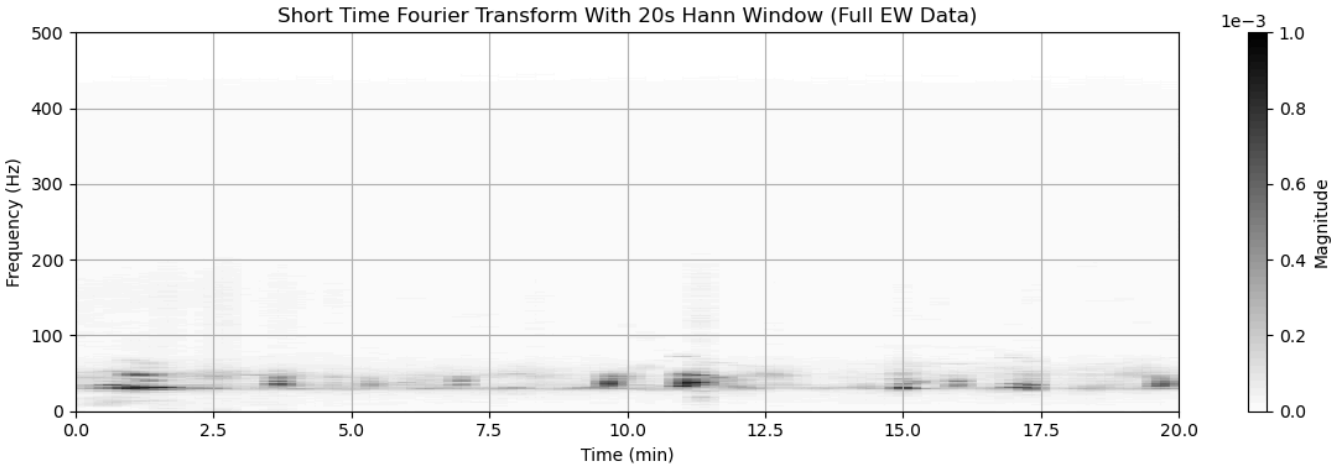
Spectrograms

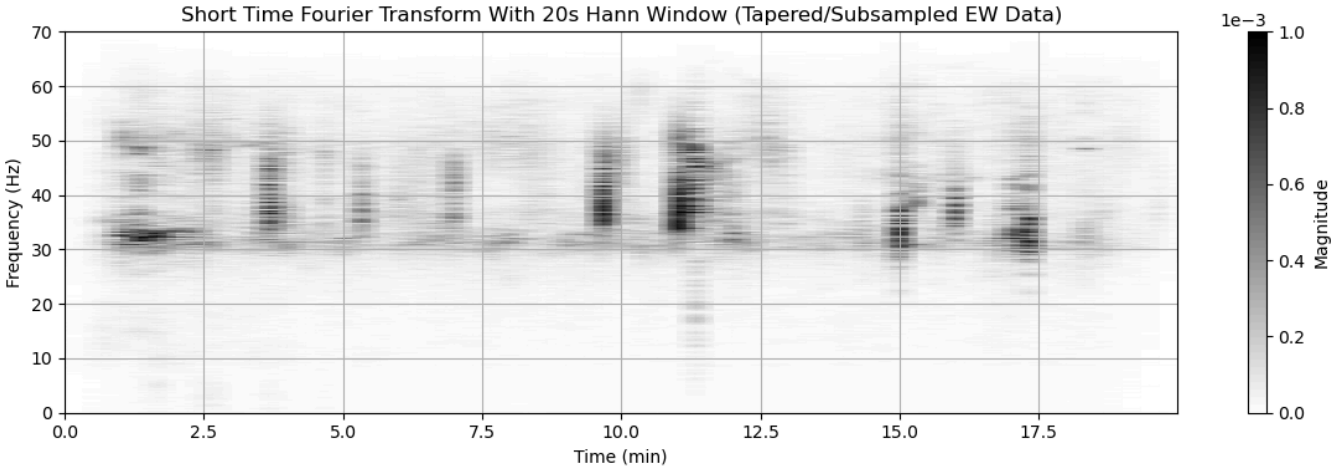
```
dataframe: df
Spectrograms for North-South data
```



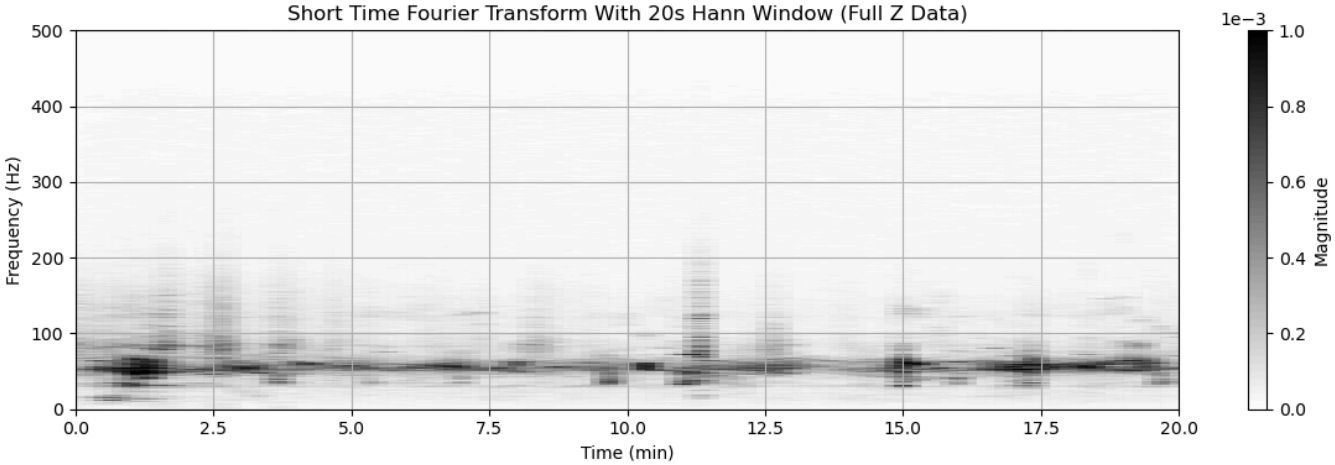


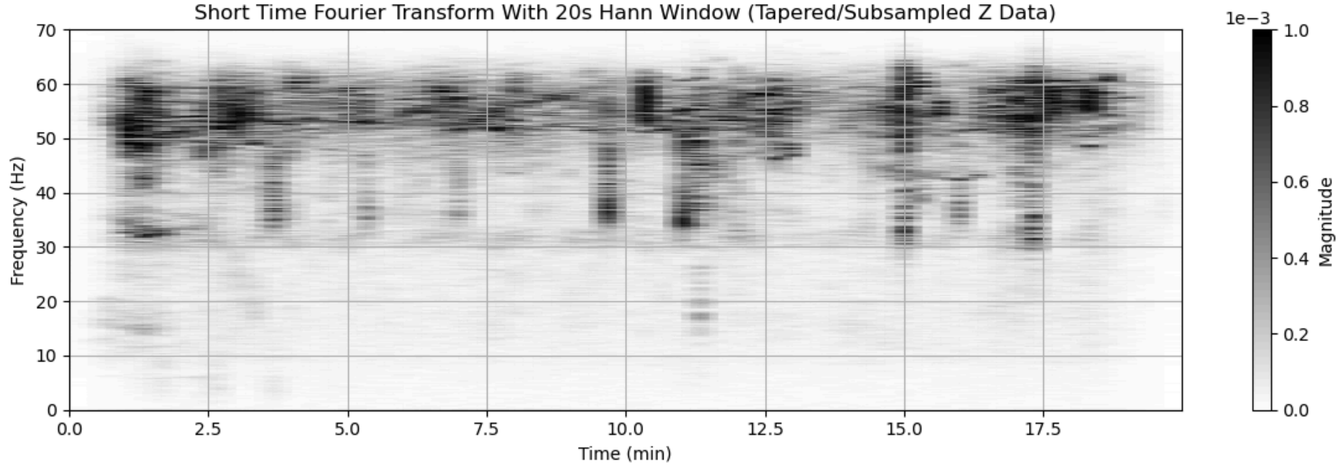

```
Spectrograms for East-West data
```





Spectrograms for Z-component data

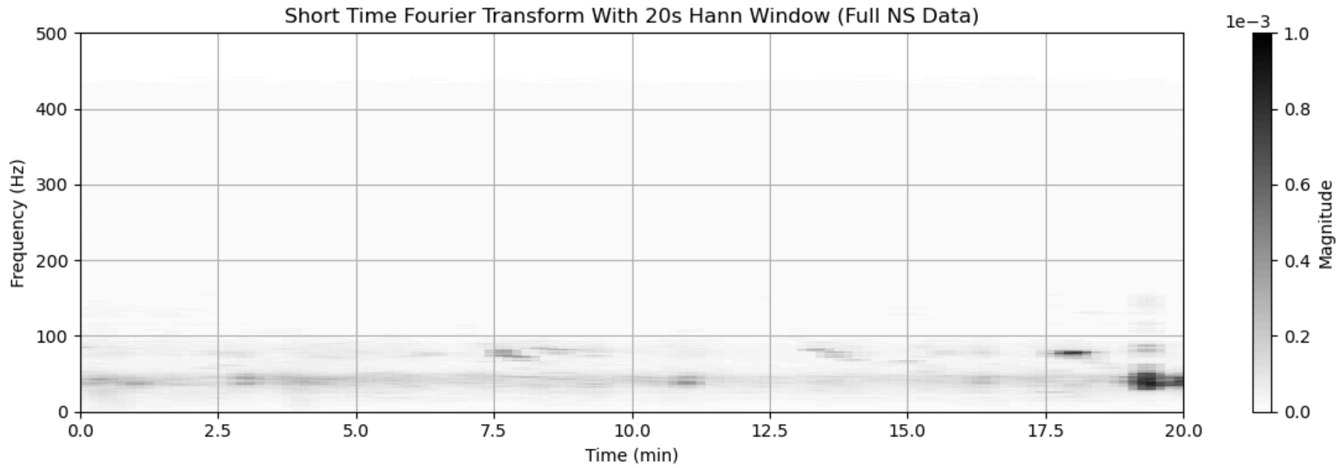


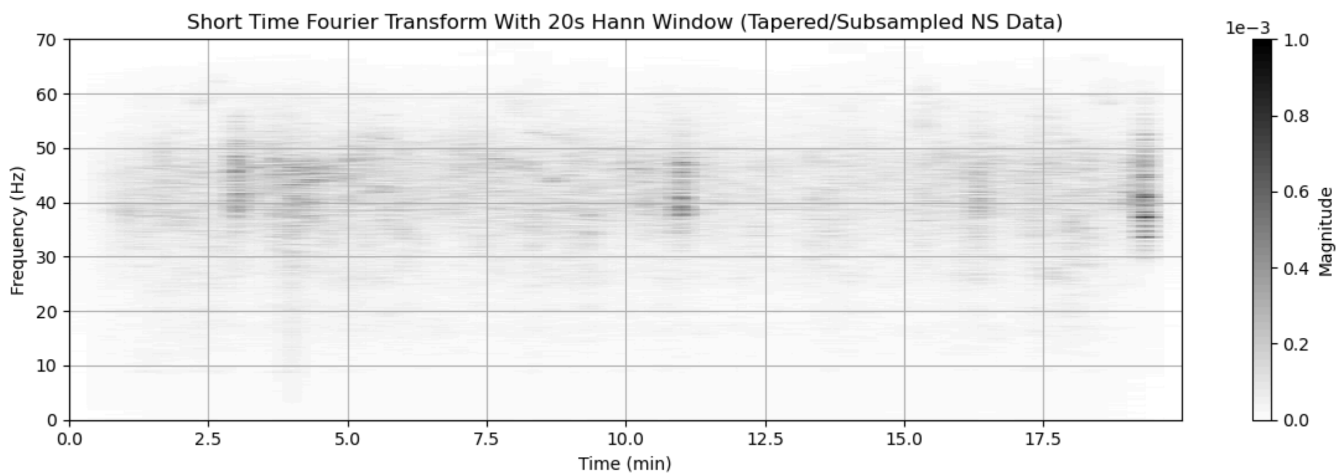


~~~~~~

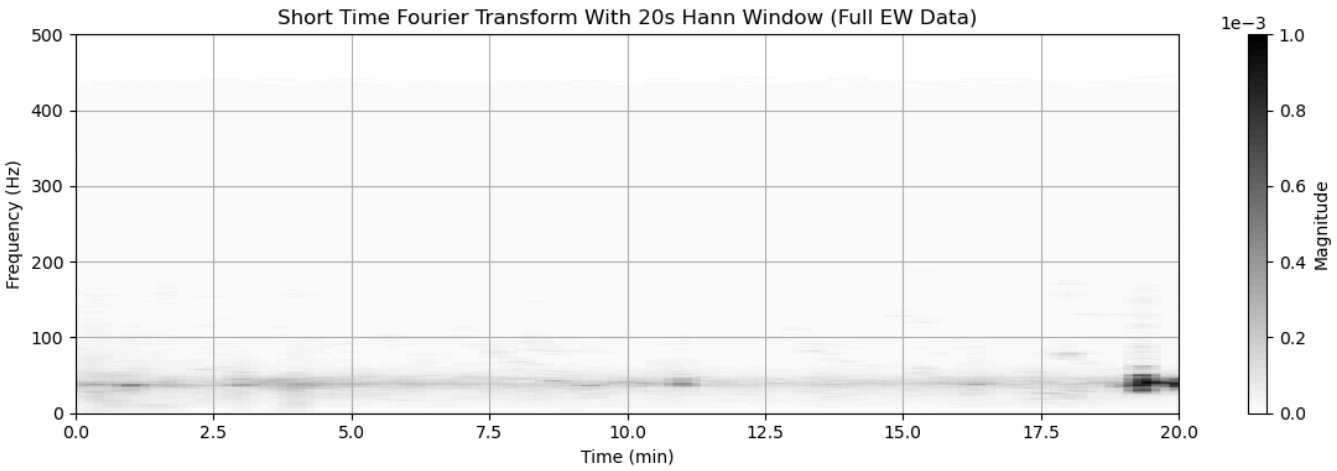
dataframe: df2

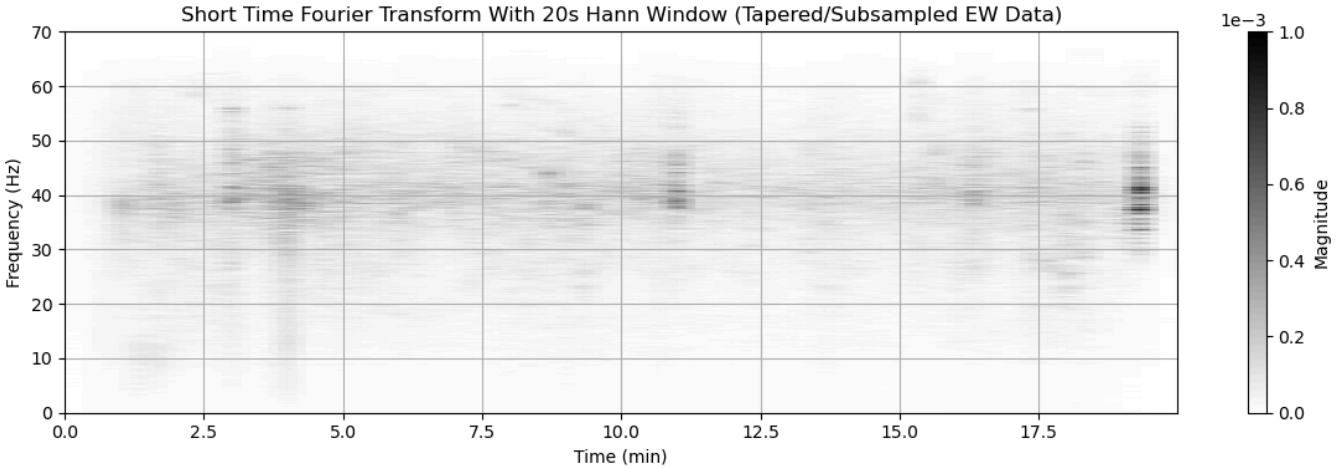
Spectrograms for North-South data



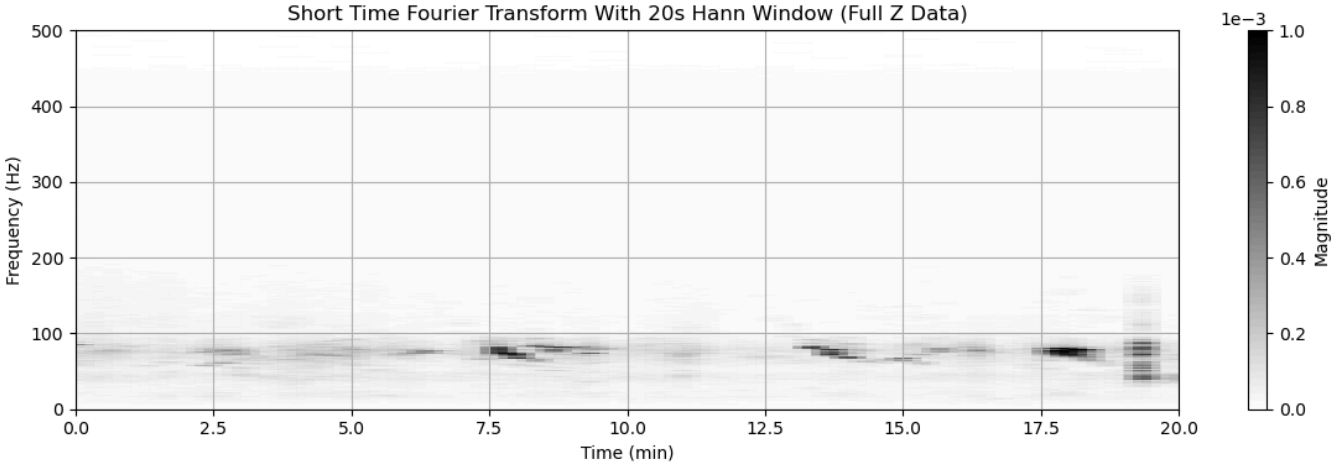


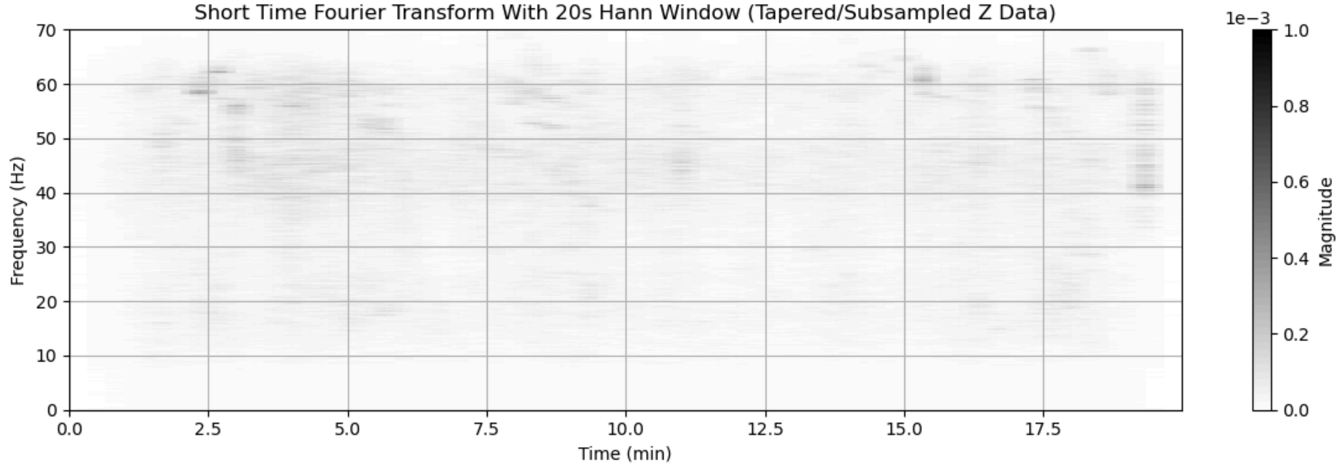
Spectrograms for East-West data
~~~~~~





Spectrograms for Z-component data

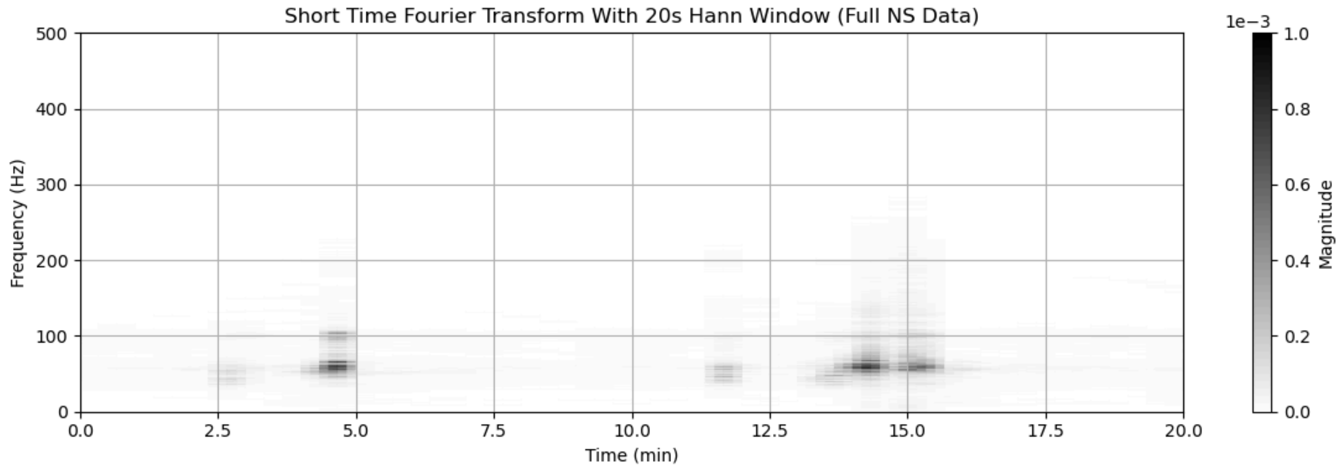


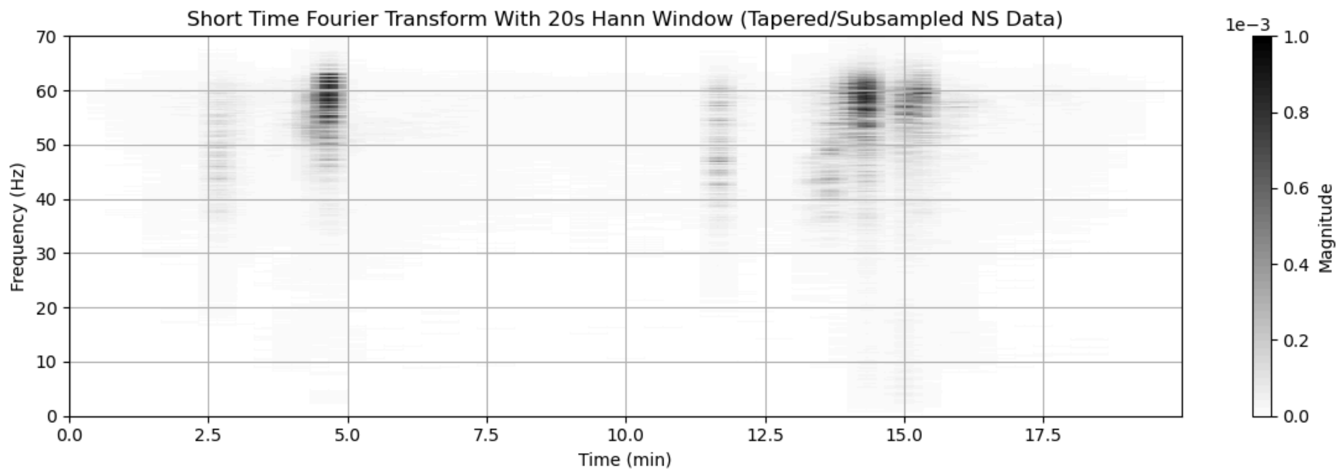


~~~~~~

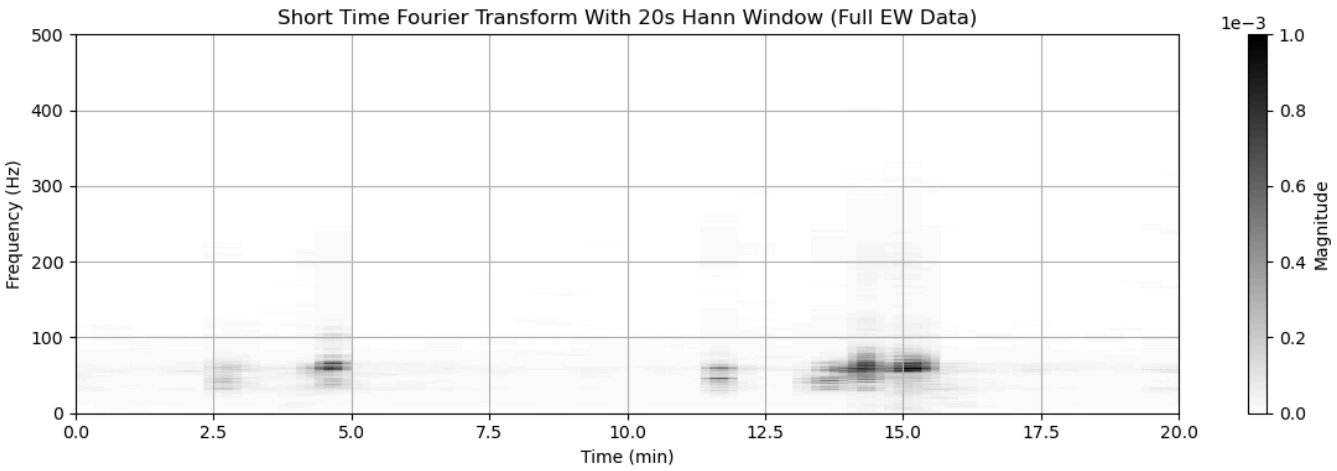
dataframe: df3

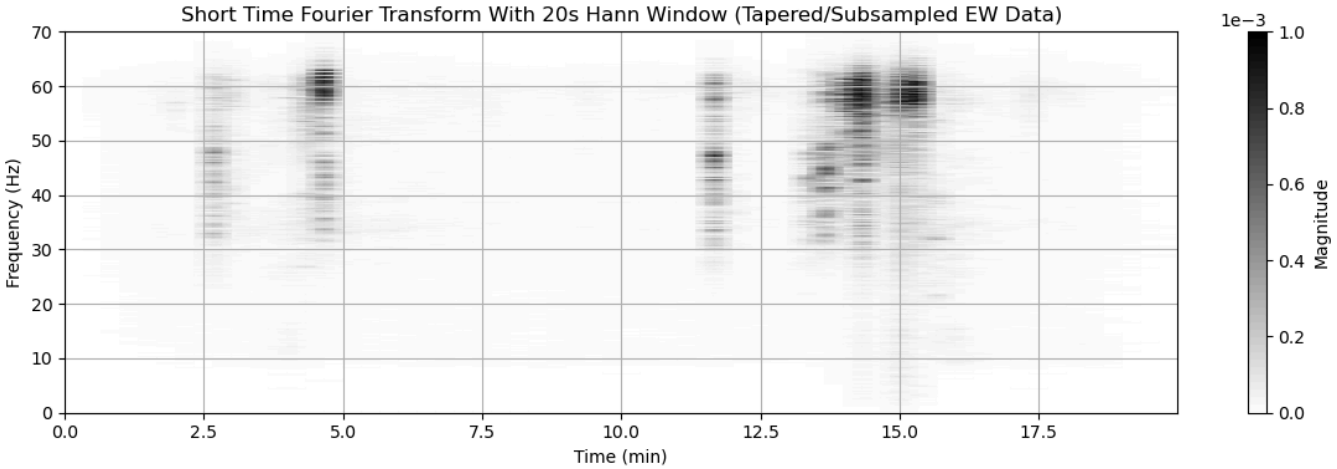
Spectrograms for North-South data



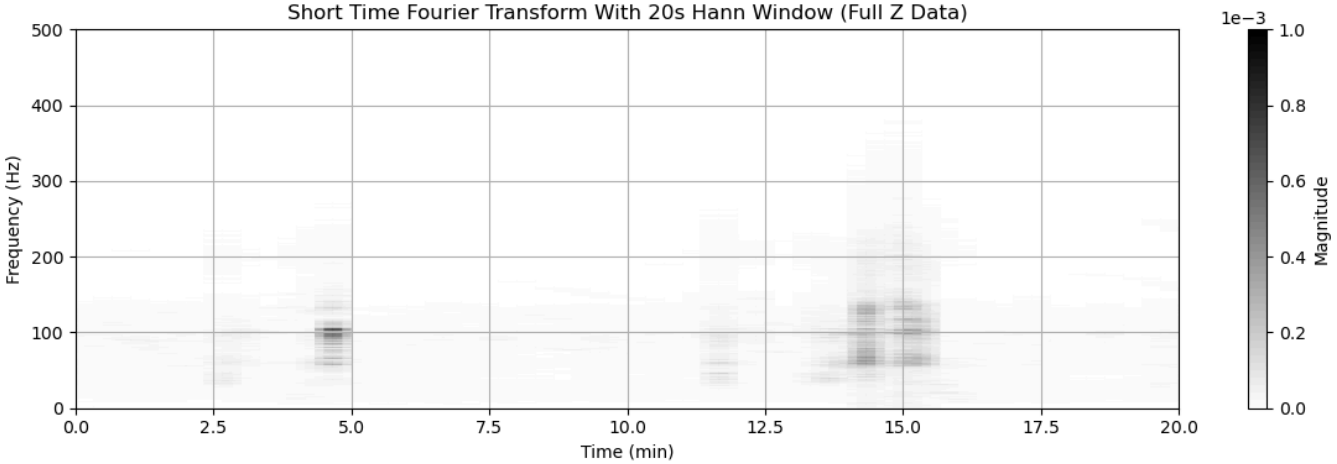


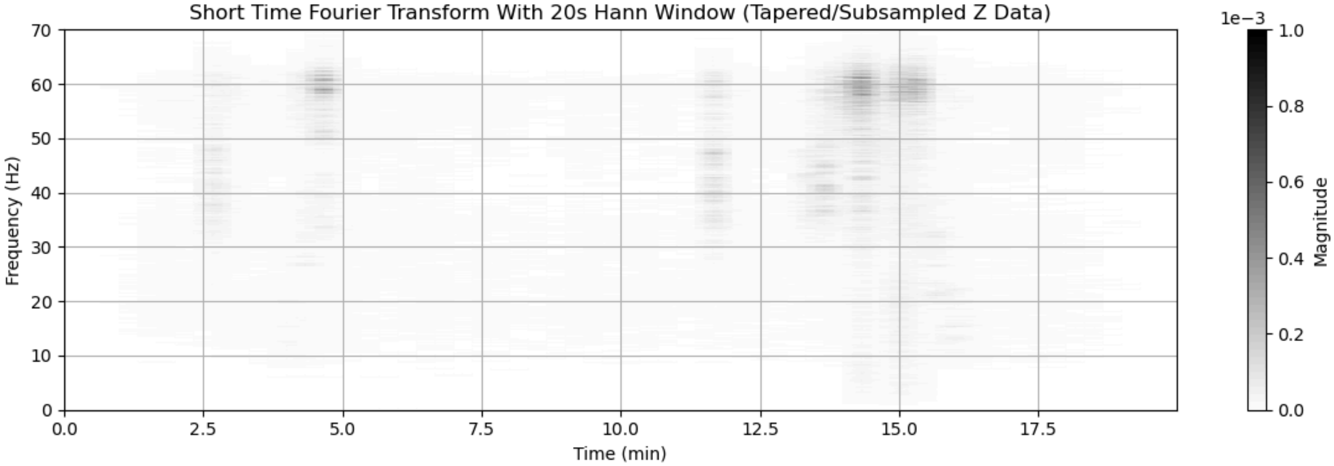
Spectrograms for East-West data
~~~~~~





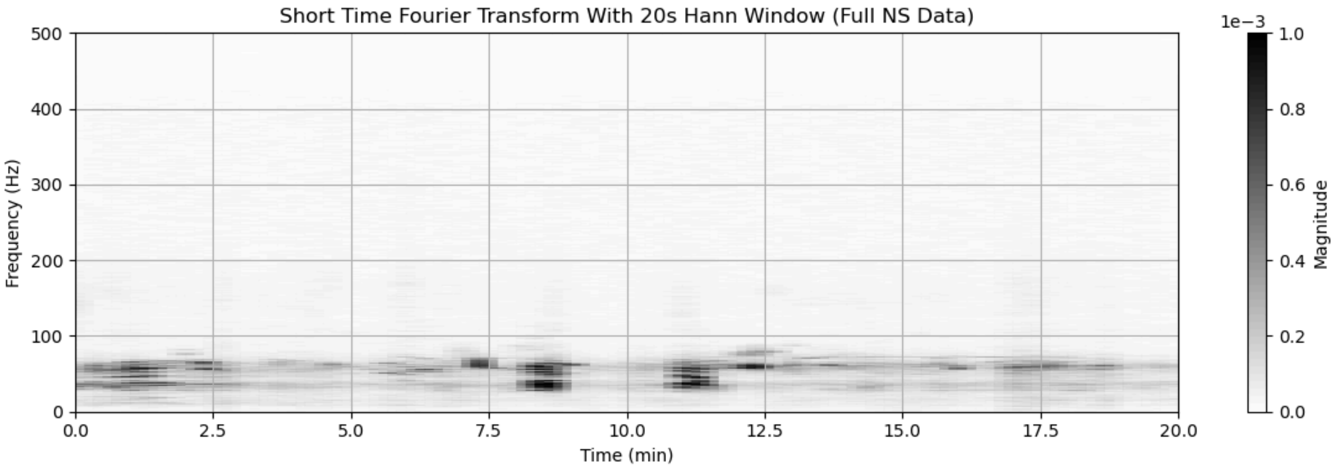
Spectrograms for Z-component data

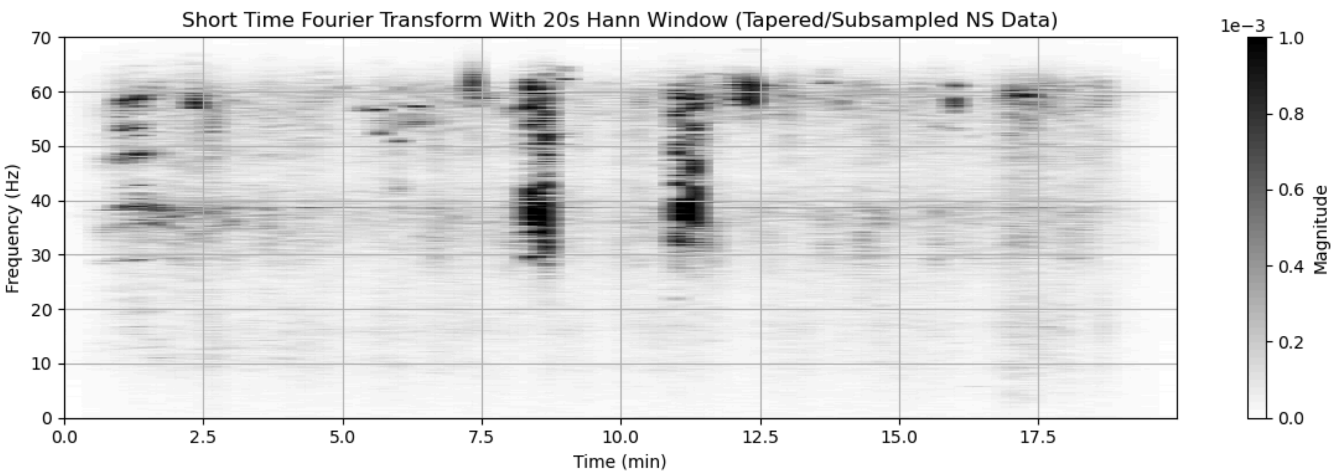




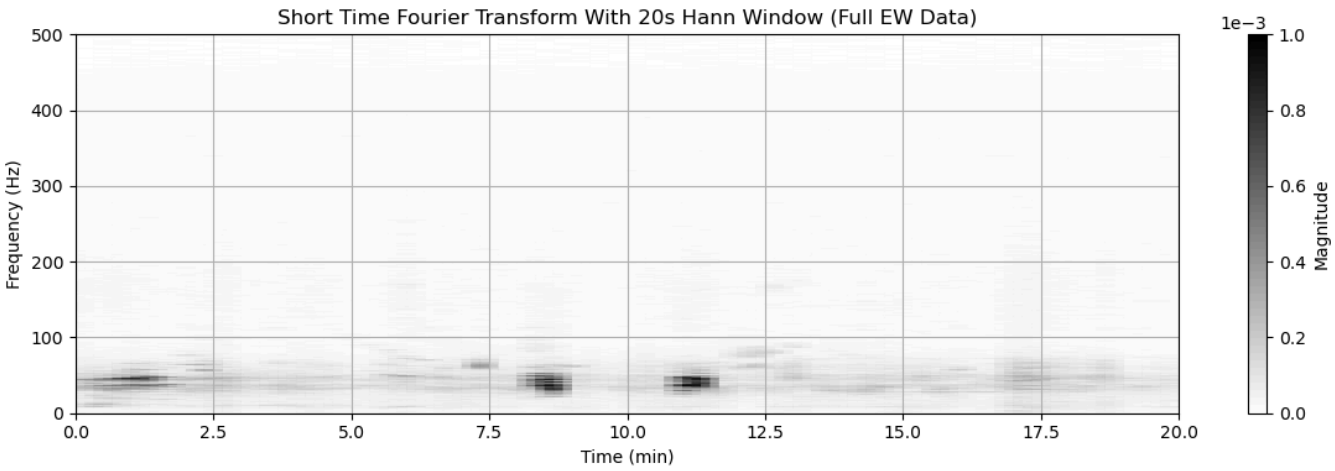
~~~~~~

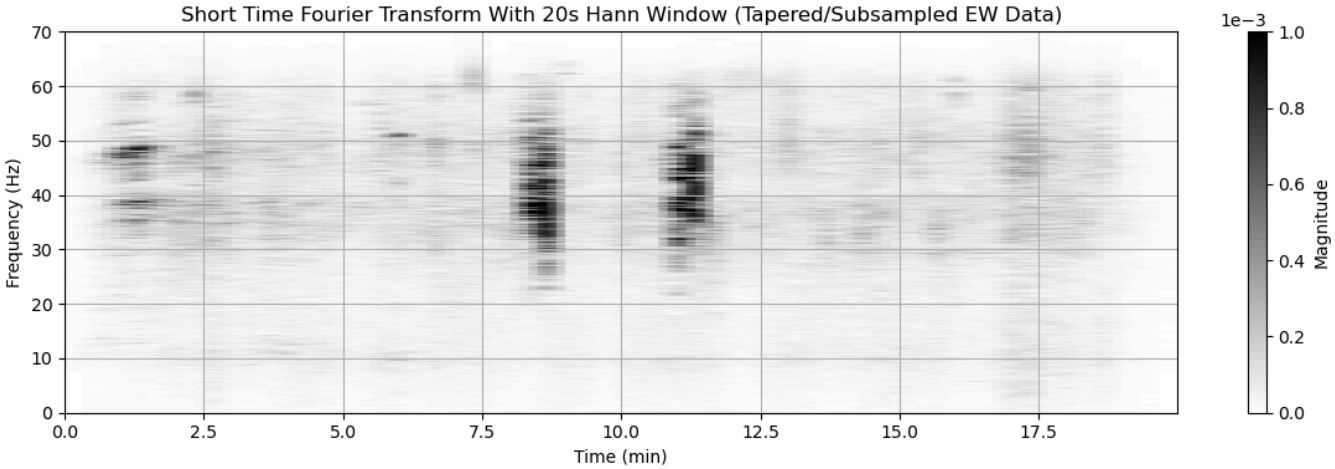
dataframe: df4
Spectrograms for North-South data



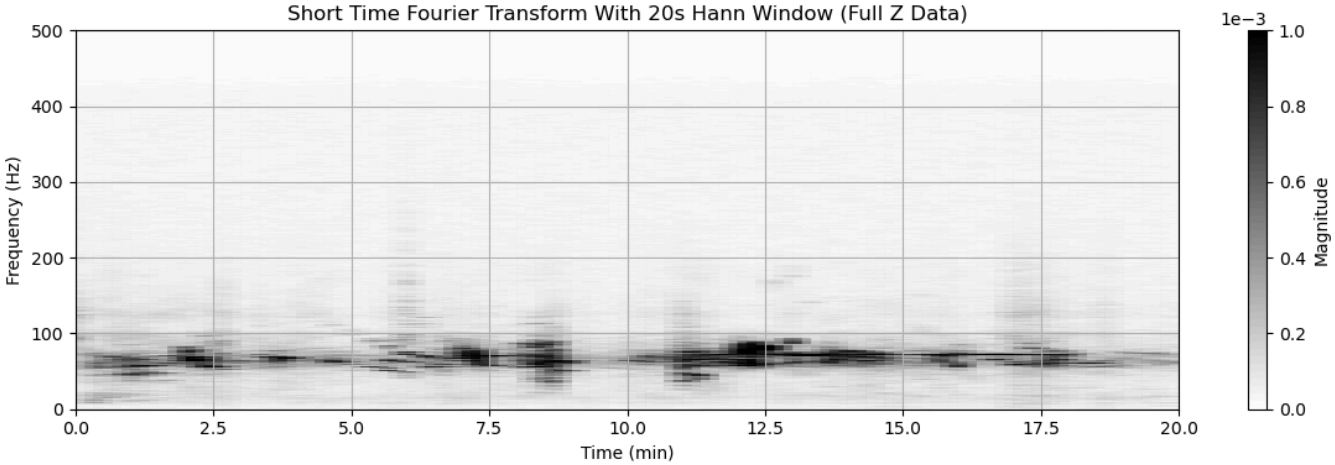


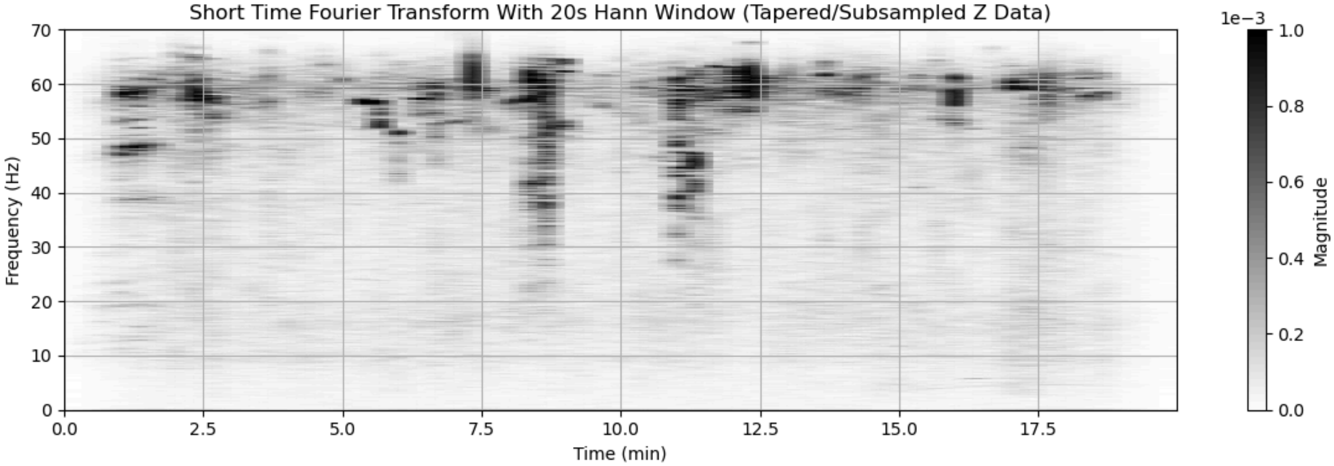
Spectrograms for East-West data
~~~~~~





Spectrograms for Z-component data

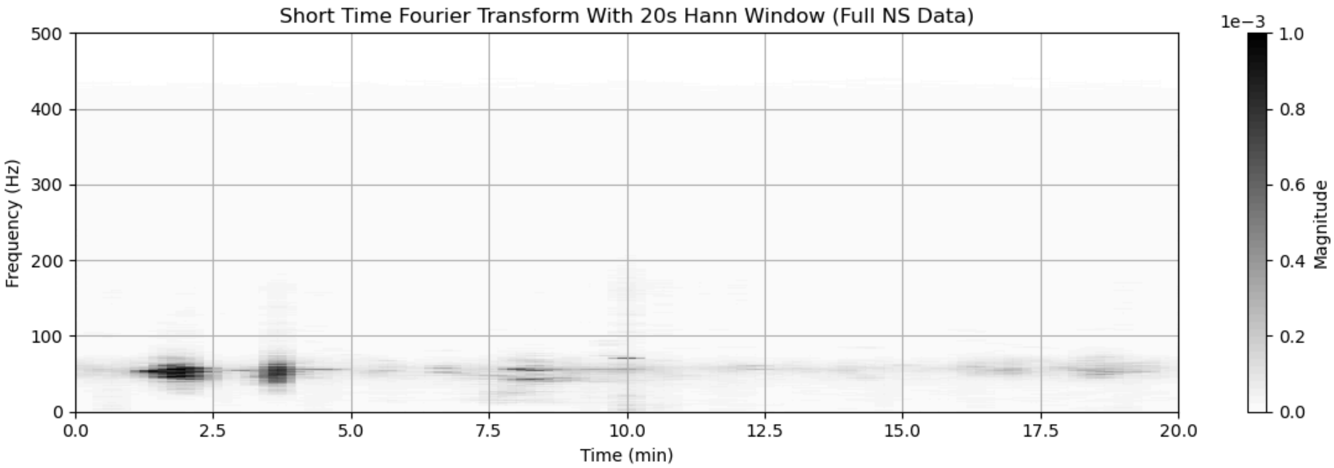


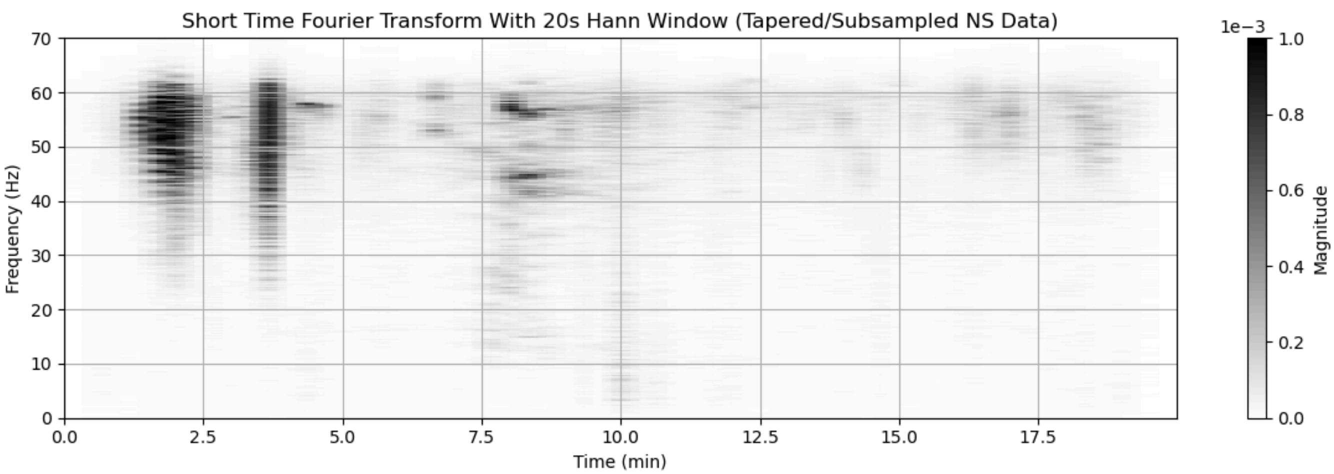


~~~~~~

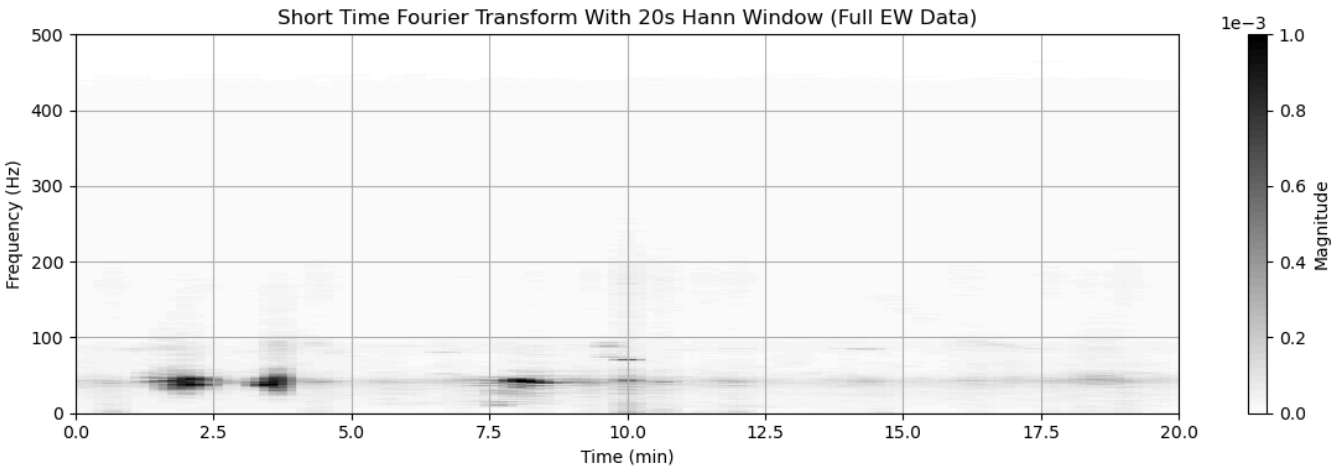
dataframe: df5

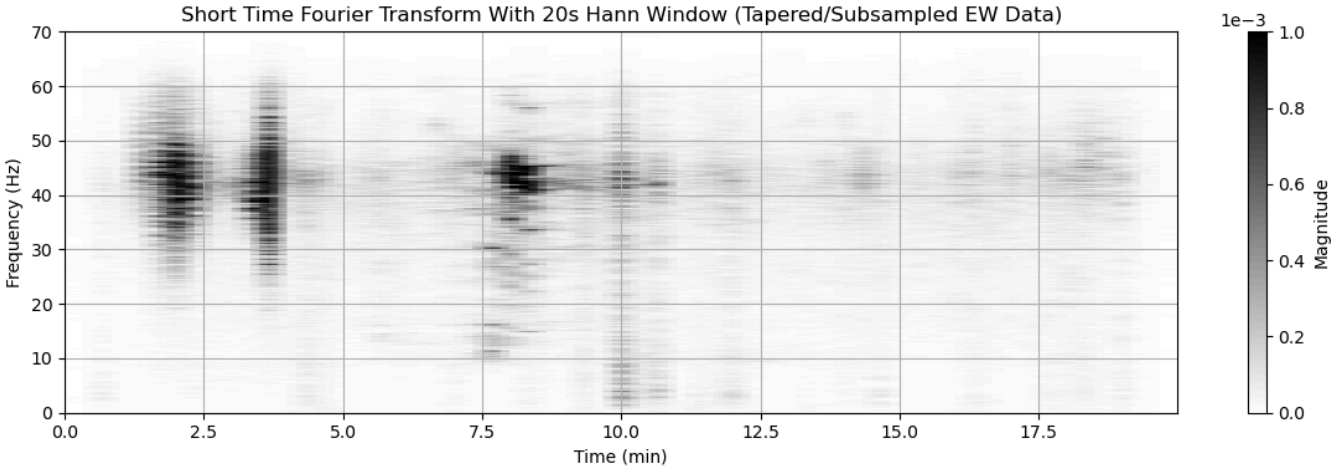
Spectrograms for North-South data



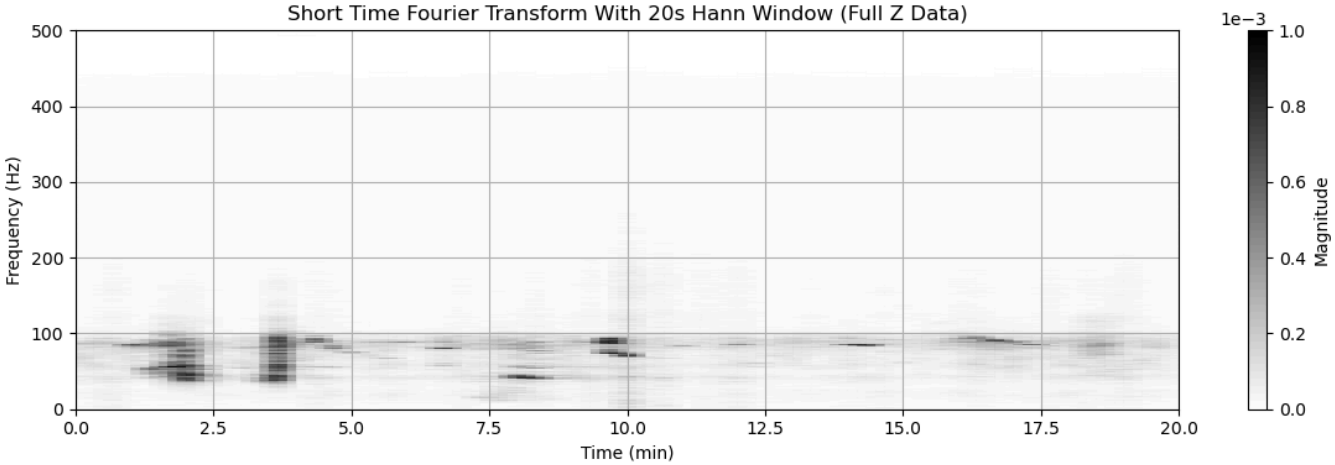


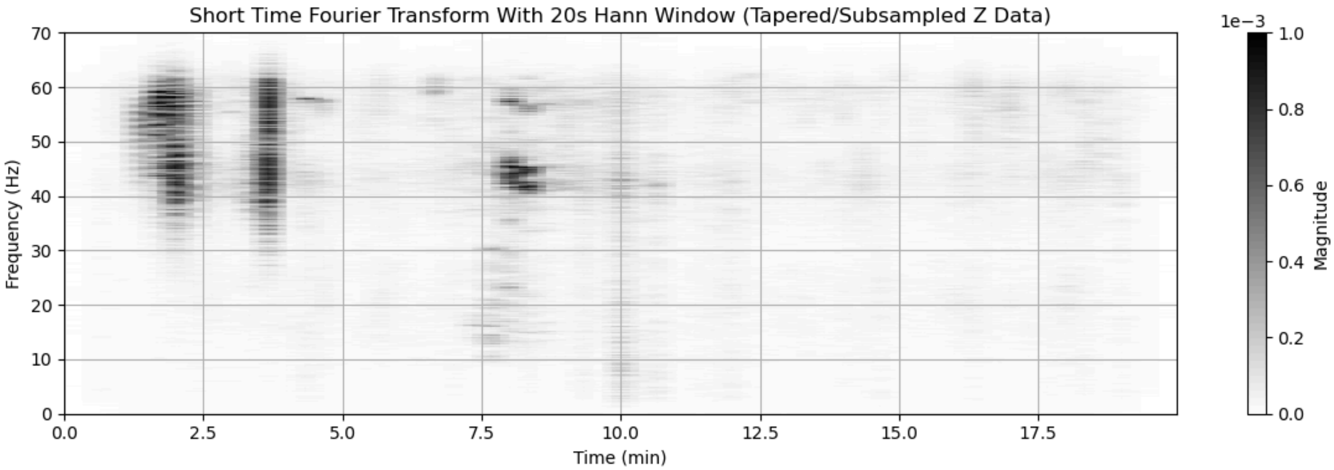
Spectrograms for East-West data
~~~~~~





Spectrograms for Z-component data





~~~~~~

H/V Ratio

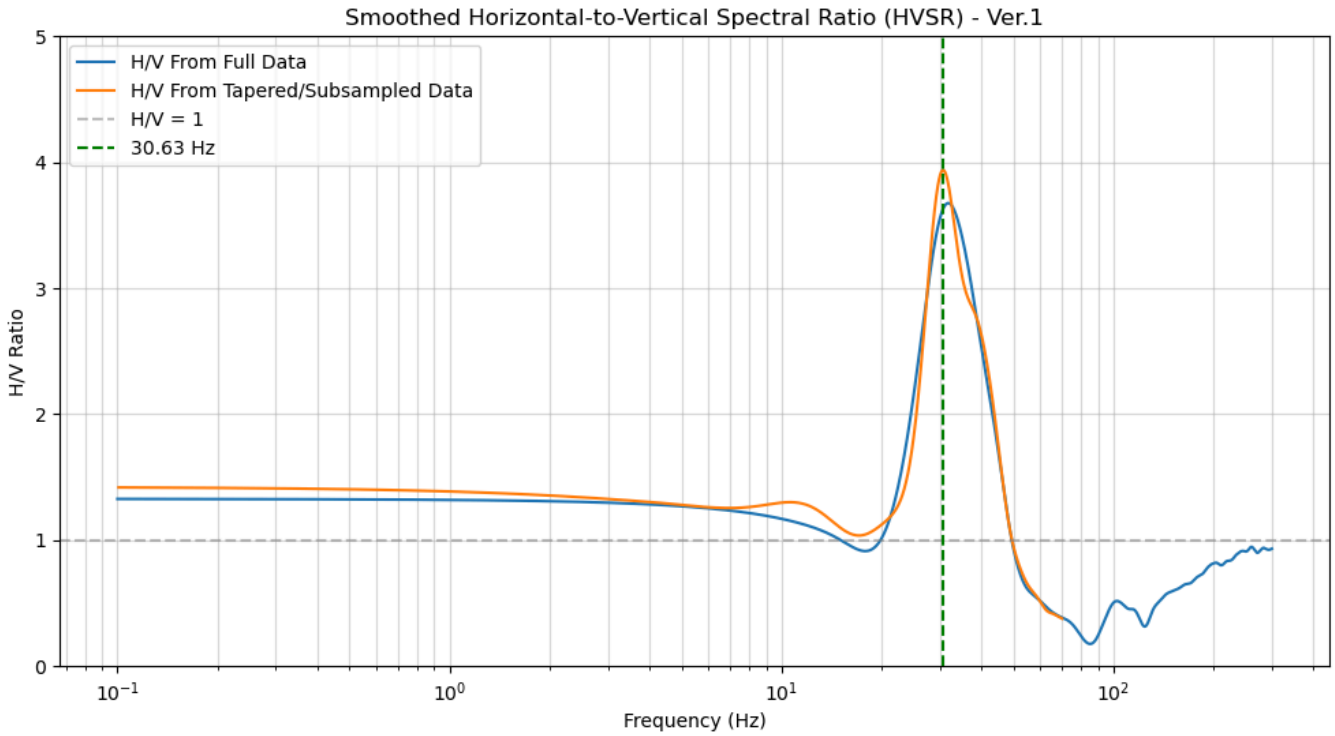
Notes:

- Divide horizontal magnitude by vertical magnitude to get H/V ratio
- Used vector-magnitude formula to combine horizontal magnitudes for North-South and East-West components

Filter for Smoothing

- applied Savitzky-Golay filter for smoothing data
    - also tried an envelope function for smoothing using the Hilbert transform
- limited displayed frequency range

```
dataframe: df
Full data peak frequency: 31.797500000000003 Hz
Full data peak H/V: 3.6743666357980906
Tapered/Subsampled data peak frequency: 30.686641693813723 Hz
Tapered/Subsampled data peak H/V: 3.937206179148854
```
~~~~~~

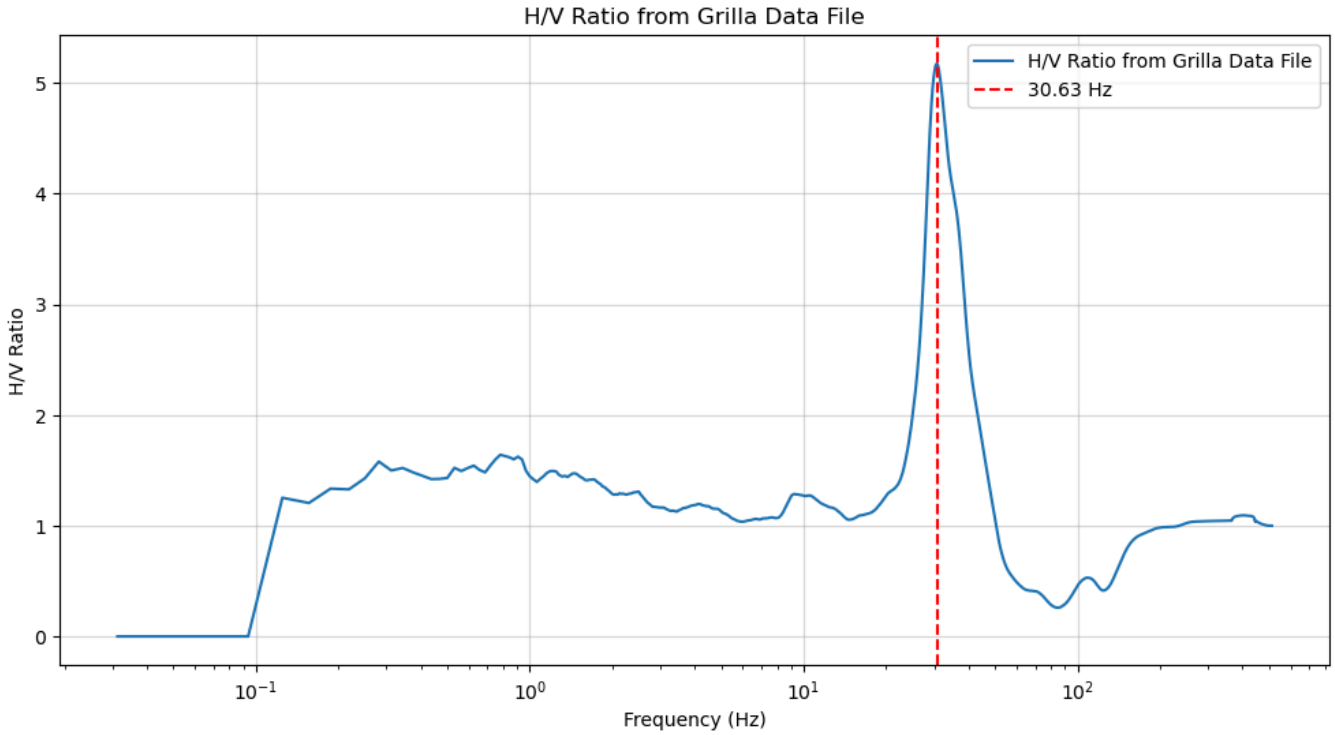


Note: 30.63 +/- 3.81 Hz is the max. H/V from screenshot of Grilla average H/V plot

Interpreting Horizontal-to-Vertical Spectral Ratios

- Peaks should occur at fundamental and higher order resonance frequencies
  - (fundamental frequency = lowest resonant frequency in a system)
- Typically, the fundamental resonance frequency is the first/lowest value indicated by the highest amplitude peak on the H/V frequency spectrum
- H/V = 1 indicates that the horizontal and vertical motions are equal
- H/V < 1 indicates that horizontal motion is weaker than vertical motion
- H/V > 1 indicates that horizontal motion is stronger than vertical motion

Plotting H/V Ratio From Grilla .txt File:



Convert Cell Outputs to PDF:

- can currently download as .html file, which when opened in a browser can be saved as a pdf using print (Ctrl+P) -> save to pdf
  - using code to directly download as .pdf file is not working currently
  - if opened with Jupyter Notebook, can save full file as .pdf directly by going to File -> Save and Export Notebook As -> PDF

```
Created HTML file: Proj_2_File_1_Tyendinaga_Updated(1).html
Open this in your browser and use Ctrl+P --> Save as PDF
(The HTML file contains only outputs and markdown, no code cells)
```