Supplement: MRS-Sim

1 | B_0 FIELD SIMULATOR

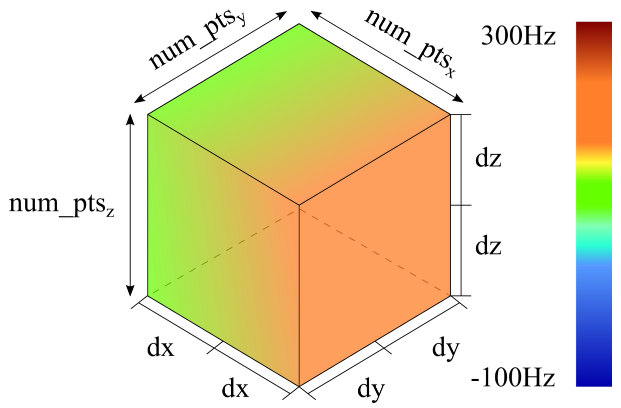


FIGURE 1 B_0 field volume with labels corresponding to the mathematical model for simulating the heterogeneities.

As described in Li et al.[?], using the B_0 field map, measured during the acquisition, to modulate the basis functions prior to fitting showed significant improvement in metabolite quantification and the fit residual surrounding the metabolite peaks. The fact that this technique has a direct impact on metabolite quantification shows the importance of including it when simulating spectra. The first step is to define the spectral and imaging resolutions of the simulated acquisitions. The defaults, shown below, assume a cubic spectroscopy voxel, but any cuboidal shape is acceptable.

spectroscopy voxel	[10 mm x 10 mm x 10 mm]
imaging voxel	[0.5mm x 0.5mm x 0.5mm]

The dimensions of the volume being modeled are the same as the spectroscopy voxel. The number of points included in the simulation is the quotient of spectroscopy resolution and the anatomical imaging resolution.

$$\mathbf{num\_pts} = \frac{\text{spectroscopy resolution}}{\text{imaging resolution}} = \left[\frac{10\text{mm}}{0.5\text{mm}}_X, \frac{10\text{mm}}{0.5\text{mm}}_Y, \frac{10\text{mm}}{0.5\text{mm}}_Z\right] \tag{1}$$

Now that the model volume has been defined, it is time to model that actual B_0 field, $dB0$. Magnetic fields do not have hard, disjointed heterogeneities, but smooth variations. This is true even in the presence of high susceptibility effects causing strong distortions as long as the voxels are reasonable small. This model therefore assumes linearly varying gradients across the volume of the voxel. This assumption allowed the problem to be reduced to four variables:

$\mathbf{dx}$	half of the mean change in the x-direction
$\mathbf{dy}$	half of the mean change in the y-direction
$\mathbf{dz}$	half of the mean change in the z-direction
μ	the mean offset of the entire voxel

The next step is to define the gradients in each direction as shown in Eqns. 2, 3, and 4. As mentioned above, this model assumes linearly varying gradients, but more complex gradients can be implemented manually.

$$\mathbf{x} = \text{linspace}(\textbf{-dx}, \textbf{dx}, \textbf{num\_pts}_X) \tag{2}$$

$$\mathbf{y} = \text{linspace}(\textbf{-dy}, \textbf{dy}, \textbf{num\_pts}_Y) \tag{3}$$

$$\mathbf{z} = \text{linspace}(\textbf{-dz}, \textbf{dz}, \textbf{num\_pts}_Z) \tag{4}$$

Once the gradients have been defined, the field map can be modeled very easily, as shown in Eqn. 5. The primary challenge in this step is reshaping the tensors to get the correct n+3D output.

$$\mathbf{dB0} = \mathbf{x} * \mathbf{y} * \mathbf{z} + \boldsymbol{\mu}, \text{ such that} \tag{5}$$

$$\mathbf{dB0}.size() = torch.Size([batchSize, \ldots, \mathbf{num\_pts}_{\mathrm{X}}, \mathbf{num\_pts}_{\mathrm{Y}}, \mathbf{num\_pts}_{\mathrm{Z}}]) \tag{6}$$

Modulation due to the heterogeneous magnetic field can be applied in both the time-domain and the frequency-domain. In the time-domain, the complex exponential is applied via multiplication with the basis function, as is shown below in Eqn. 7. In the frequency-domain, the complex exponential needs to be convolved with the spectrum. In both mathematical and practical terms, it is simpler to apply the modulation in the time-domain.

$$F(t) = \sum_{n}^{N} M_n * basisfcn_n * \underbrace{\sum_{r=1}^{R} e^{-i\Delta\omega_r t}}_{B_0\ inhomog.}, \text{ where } \Delta\omega = \mathbf{dB0}.flatten(dims = [-3:-1]) \text{ and } R = cumprod(\mathbf{num\_pts}) \tag{7}$$

In this equation, the model **dB0** is flattened along the last three dimensions representing the **x**-, **y**-, and **z**-dimensions respectively. Therefore, **r** represents the linear indices from [1,R] corresponding the subscripts of the points within the 3D volume. The effective magnetic field at each of those points within the voxel has a direct effect on each moeity that resonates and contributes to the overall signal. Because of this, each basis function is modulated independently before summation.

2 | BASELINE AND RESIDUAL WATER SIMULATIONS

A single simulation protocol is used to generate the contributions for both baseline offsets and residual water contributions. This work uses a smoothed, pseudo-random bounded walk algorithm for the simulations. For simplicity, this will henceforth be referred to as a *(random) bounded walk*, or simply a *walk*. By using different configuration dictionaries, it is possible to switch from very smooth, undulating baselines to rather erratic and rough residual water regions. The following sections highlight the effects different parameters have on the simulations.

2.1 | Variables

These simulations use at least 9 degrees of freedom with two additional, but optional, inputs:

1. Starting height (start)
2. Ending height (end)
3. Standard deviation of walk (std)
4. Lower bound (lower)
5. Upper bound (upper)
6. Point density (pt_density)
7. PPM range (ppm_range)
8. Length of smoothing window (window)
9. Scale (scale)
10. Dropout probability (drop_prob)
11. *Prime (prime)

Floats or integers can be provided for the point density, prime, and drop_prob. Everything else should be provided as a list with either a single value or a range in the form of [min_range, max_range]. When a single value is provided, that variable will be fixed. If a range is specified, then values will be sampled from that range uniformly. The point density and the PPM range are used to calculate the length of the walks. This length coupled with std control the flexibility of raw simulation. The length of the smoothing window, $window$, will affect the smoothness of the resulting walk. It is specified as a fraction of the length of the walk, so that it is independent of walk length, and can also be either fixed or sampled. $scale$ determines how prominent each offset will be when added to the simulated data. $drop\_prob$ is used to randomly omit the offset from that percentage of the simulations. $prime$ [units: ppm] is used for generating the residual water regions. This allows the region to vary in length by sampling two values in the range of $[-prime, prime)$ that can expand, contract, or slightly offset the residual water region.

2.2 | Post-processing

Once the random walks have been generated, they are then smoothed with kernels of either fixed or varying width. At this point, the walks are normalized to [0,1] and then scaled down according to $scale$. Afterwards, trend lines are calculated between the starting and ending points which are then removed so that they start and end on the x-axis. A function called *sim2acquired* then uses zero padding and a nonuniform interpolater to add tails to both sides of the walks so that they match the PPM range of the basis set regarding the spectral width and carrier frequency. Before being added to the FIDs, they are multiplied by the maximum value of the FID in the frequency domain to scale them up to the correct order of magnitude. This makes $scale$ relative to the maximum height in the spectra. The offsets are added to the FIDs in the frequency domain before an inverse FFT returns the data to the time domain.

The parameters presented below control the length of the walks, their trend lines, flexibility, and smoothness. The default values presented in the config dictionaries were determined through preliminary experiments to closely approximate what was observed in a clinical dataset.

2.3 | Preparing for the simulations

The following section defines parameters for the spectroscopy scenario and the config dictionary for the simulations.

```
# Spectroscopy scenario
spectralwidth = 2000                      # Hz
Ns            = 2048                      # number of spectral points
B0            = 3.0                       # T
gamma_H       = 42.577478518              # MHz/T
ppm_ref       = 4.65                      # ppm

carrier_frequency = B0 * gamma_H          # MHz
ppm = torch.linspace(-0.5*spectralwidth,
                     0.5*spectralwidth,Ns) # Hz
ppm /= carrier_frequency                  # ppm
ppm += ppm_ref
t = torch.linspace(0,Ns/spectralwidth,Ns)

num_samples = 10
```

2.4 | Understanding the Simulation Plots

The plots below follow contain either one or two lines that are blue or red. The default color scheme is as follows:

blue	raw random walk simulation
red	smoothed walk that is added to the simulated spectra

In Sections 2.5.4 and 2.6.5, which discuss how to incorporate the walks into the simulated spectra, only a single blue line is plotted. This line is the smoothed, real component of the walk. In Sections 2.5.5 and 2.6.6 which show the effect of applying the Hilbert transform, the blue and red lines represent the real and imaginary components, respectively. Similarly, Sections 2.5.6 and 2.6.7, which include examples of random simulations for each contribution type, also show the real and imaginary components using the blue and red, respectively.

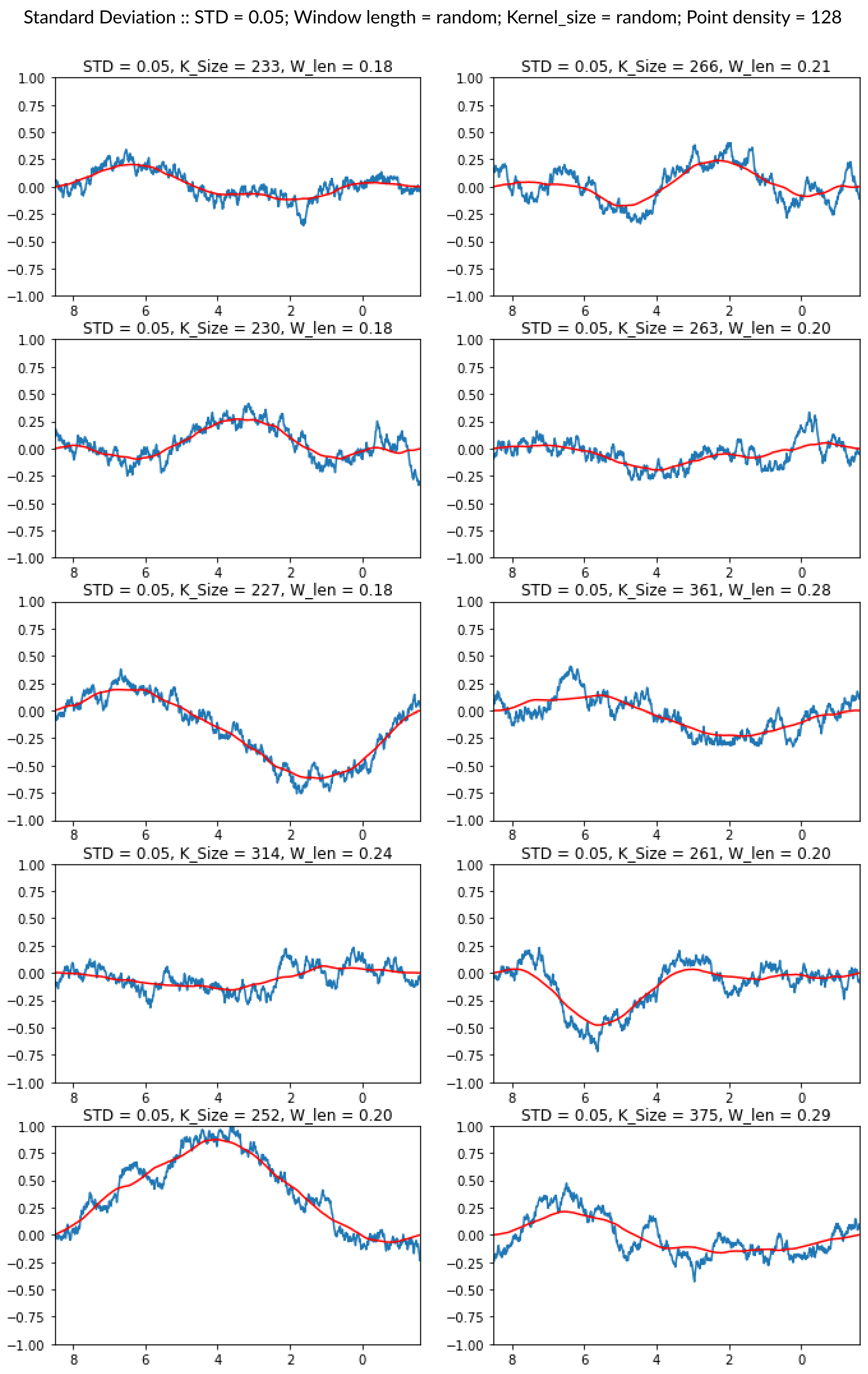
2.5 | Baseline Offsets

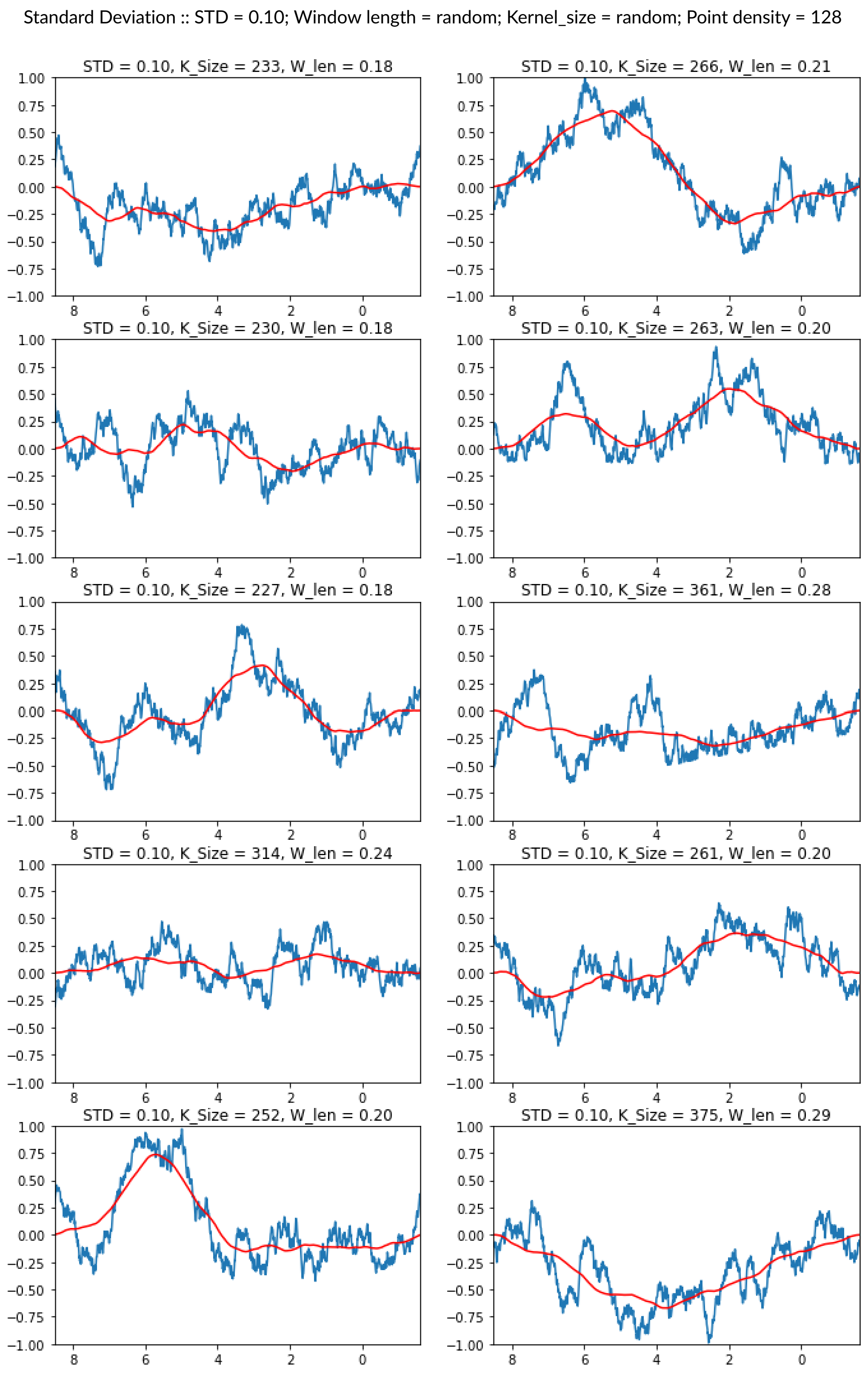
The following section will explore this generator using the baseline configuration dictionary defined below. The values selected for the plots in the following sections were tailored specifically to the baseline simulations.

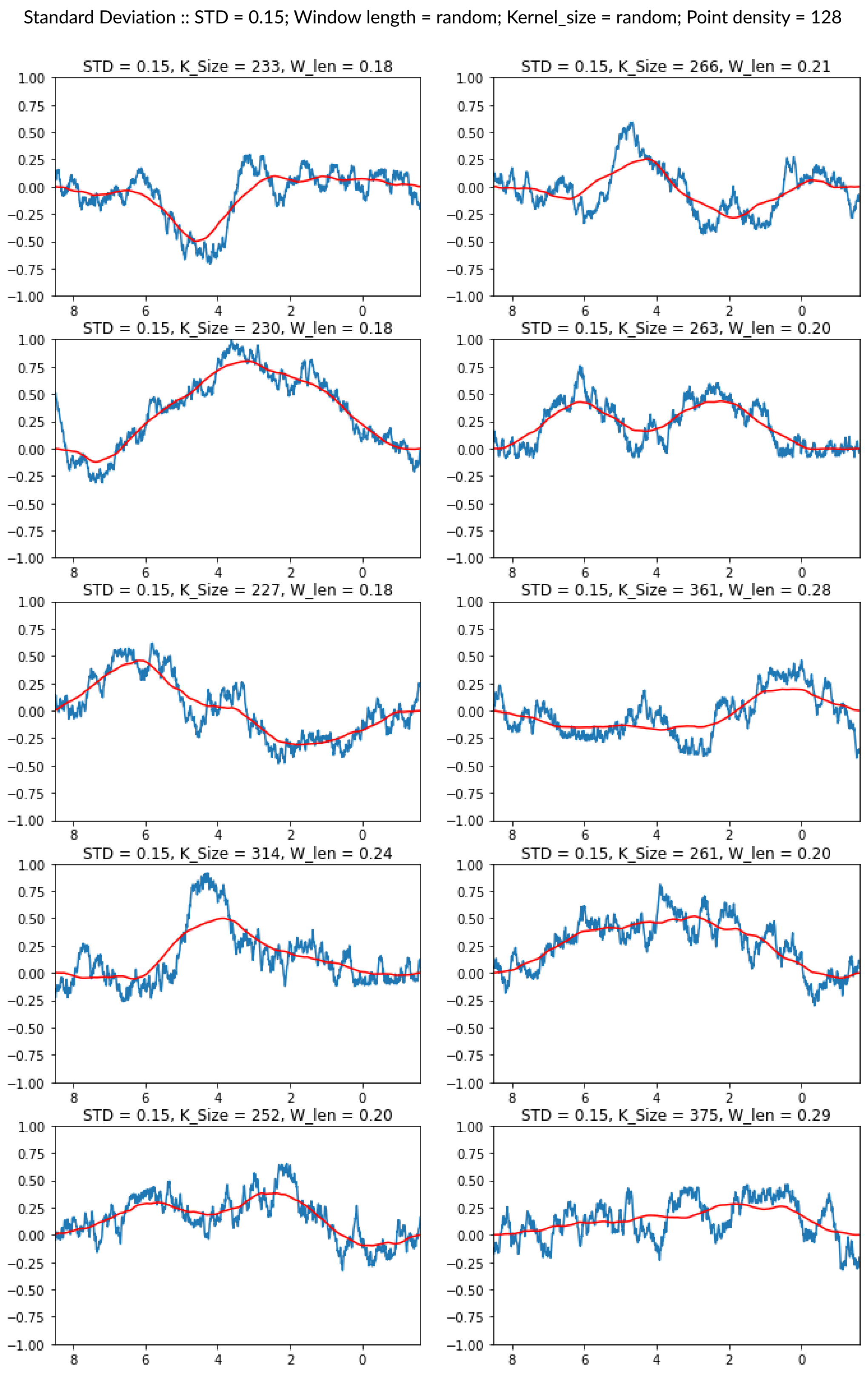
```
baseline_cfg = {
        "start":             [    -1,      1],
        "end":               [    -1,      1],
        "upper":             [             1],
        "lower":             [            -1],
        "std":               [  0.05,   0.20],
        "window":            [  0.15,    0.3],
        "pt_density":            128,
        "ppm_range":         [  -1.6,    8.5],
        "scale":             [     0,    1],
        "drop_prob":             0.0
}
```

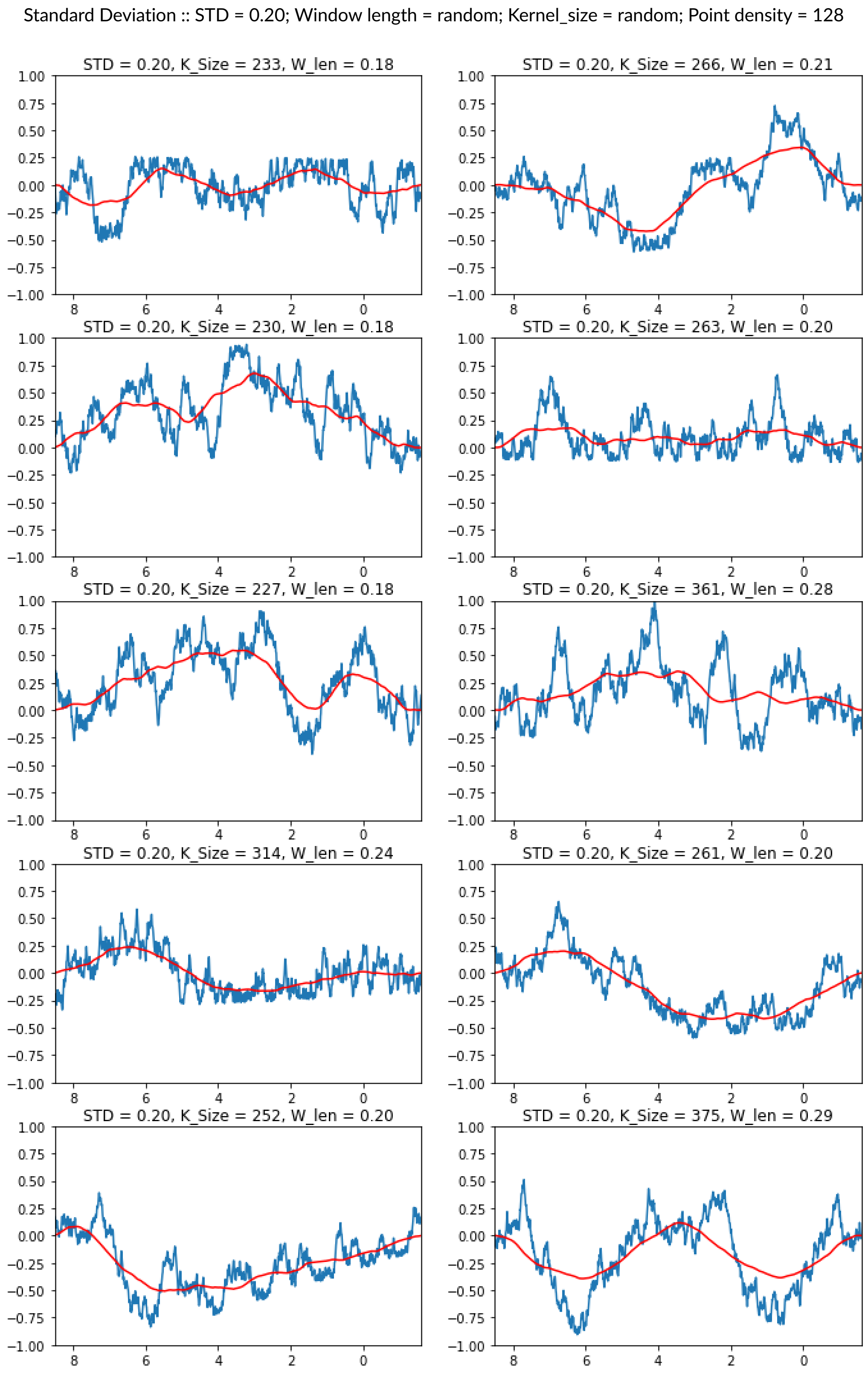
2.5.1 | Standard Deviation

This section shows the effect of different std values on the random walk. The *std* variable is used when sampling the noise for the walk. It directly controls the amount of variation between two consecutive points prior to the cumulative summation.


Standard Deviation :: STD = 0.05; Window length = random; Kernel_size = random; Point density = 128
STD = 0.05, K_Size = 233, W_len = 0.18
STD = 0.05, K_Size = 266, W_len = 0.21
STD = 0.05, K_Size = 230, W_len = 0.18
STD = 0.05, K_Size = 263, W_len = 0.20
STD = 0.05, K_Size = 227, W_len = 0.18
STD = 0.05, K_Size = 361, W_len = 0.28
STD = 0.05, K_Size = 314, W_len = 0.24
STD = 0.05, K_Size = 261, W_len = 0.20
STD = 0.05, K_Size = 252, W_len = 0.20
STD = 0.05, K_Size = 375, W_len = 0.29
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
8
6
4
2
0

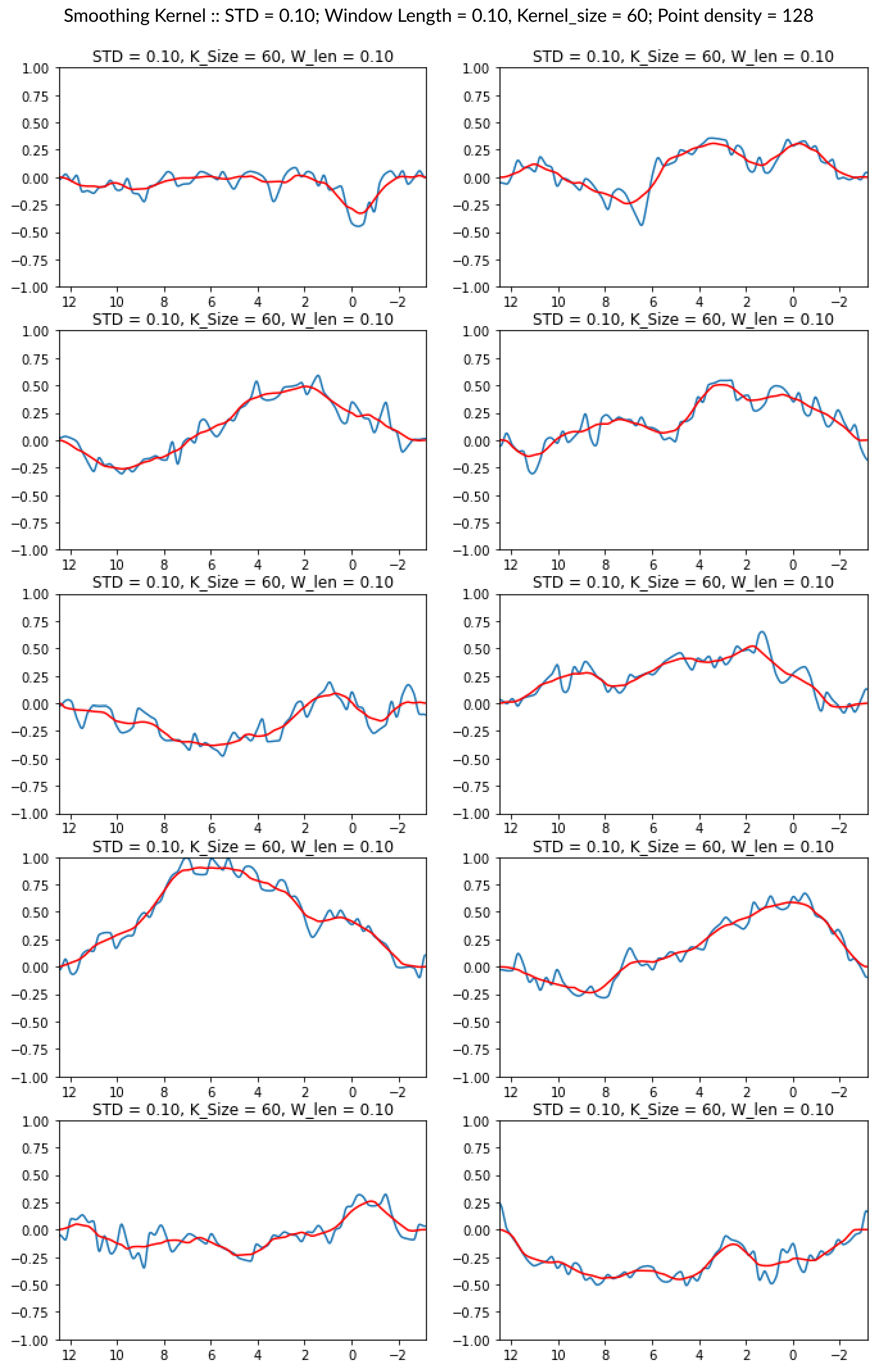

Standard Deviation :: STD = 0.10; Window length = random; Kernel_size = random; Point density = 128
STD = 0.10, K_Size = 233, W_len = 0.18
STD = 0.10, K_Size = 266, W_len = 0.21
STD = 0.10, K_Size = 230, W_len = 0.18
STD = 0.10, K_Size = 263, W_len = 0.20
STD = 0.10, K_Size = 227, W_len = 0.18
STD = 0.10, K_Size = 361, W_len = 0.28
STD = 0.10, K_Size = 314, W_len = 0.24
STD = 0.10, K_Size = 261, W_len = 0.20
STD = 0.10, K_Size = 252, W_len = 0.20
STD = 0.10, K_Size = 375, W_len = 0.29
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
8
6
4
2
0

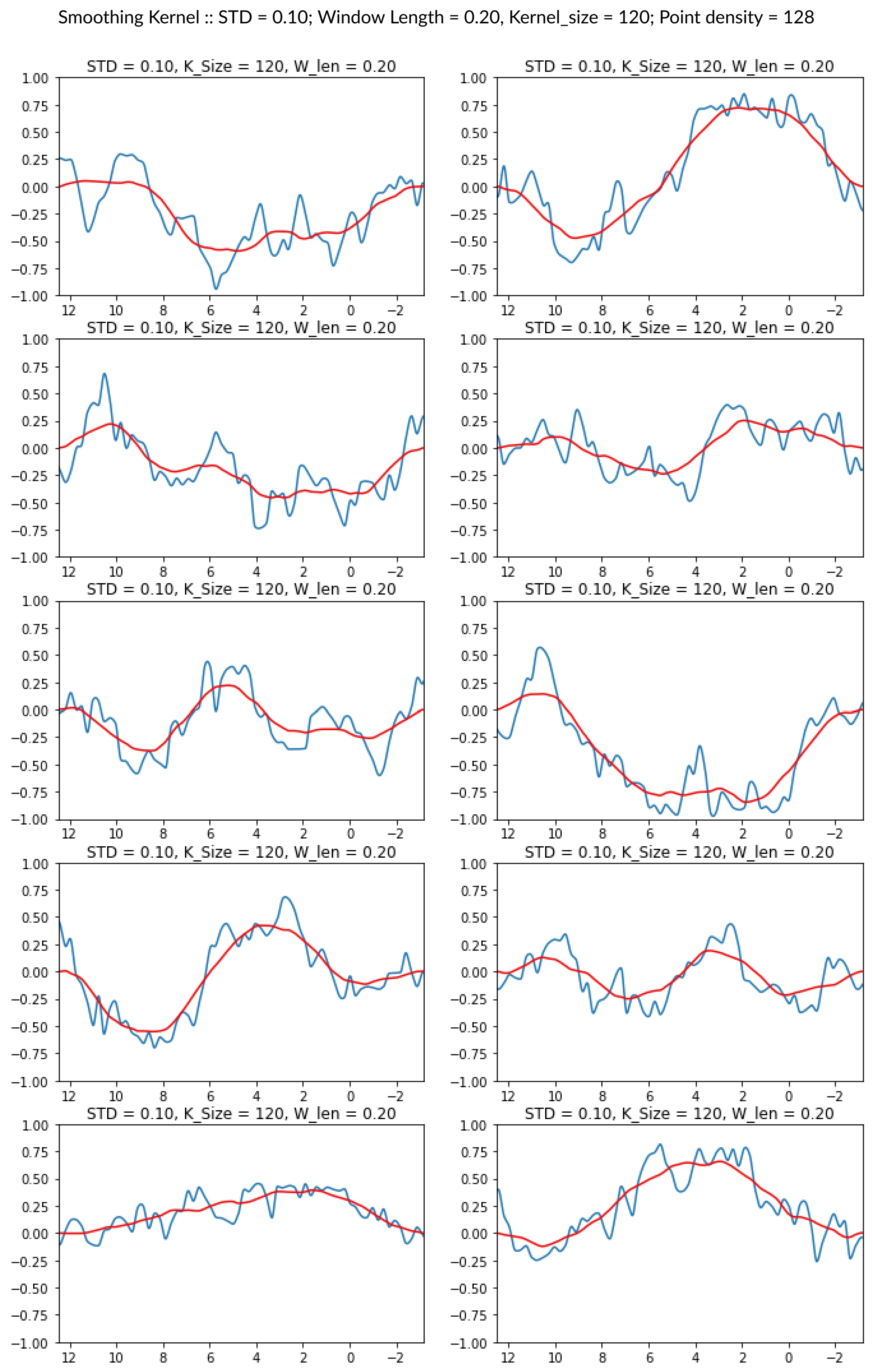

Standard Deviation :: STD = 0.15; Window length = random; Kernel_size = random; Point density = 128
STD = 0.15, K_Size = 233, W_len = 0.18
STD = 0.15, K_Size = 266, W_len = 0.21
STD = 0.15, K_Size = 230, W_len = 0.18
STD = 0.15, K_Size = 263, W_len = 0.20
STD = 0.15, K_Size = 227, W_len = 0.18
STD = 0.15, K_Size = 361, W_len = 0.28
STD = 0.15, K_Size = 314, W_len = 0.24
STD = 0.15, K_Size = 261, W_len = 0.20
STD = 0.15, K_Size = 252, W_len = 0.20
STD = 0.15, K_Size = 375, W_len = 0.29
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
8
6
4
2
0

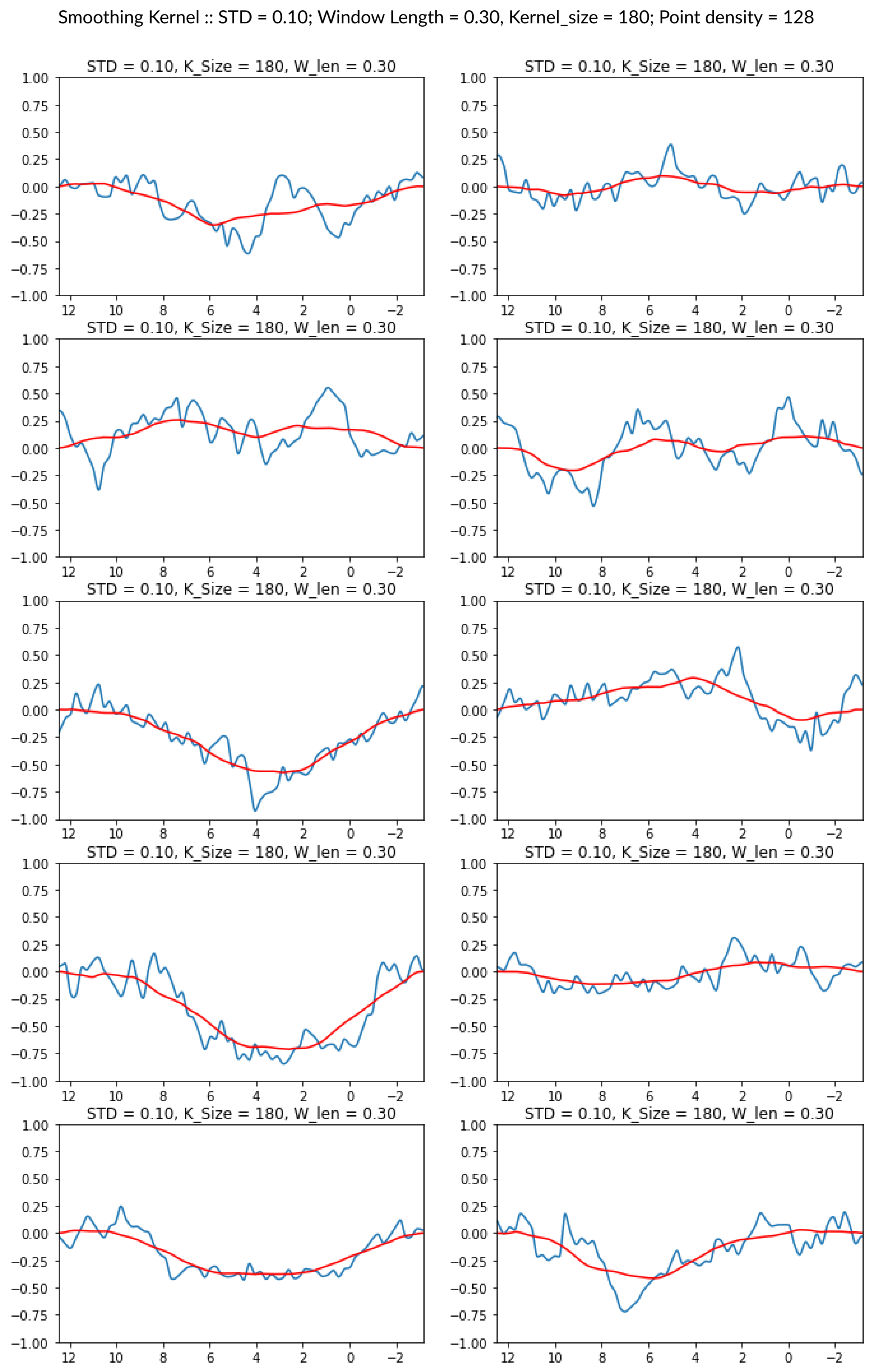

Standard Deviation :: STD = 0.20; Window length = random; Kernel_size = random; Point density = 128
STD = 0.20, K_Size = 233, W_len = 0.18
STD = 0.20, K_Size = 266, W_len = 0.21
STD = 0.20, K_Size = 230, W_len = 0.18
STD = 0.20, K_Size = 263, W_len = 0.20
STD = 0.20, K_Size = 227, W_len = 0.18
STD = 0.20, K_Size = 361, W_len = 0.28
STD = 0.20, K_Size = 314, W_len = 0.24
STD = 0.20, K_Size = 261, W_len = 0.20
STD = 0.20, K_Size = 252, W_len = 0.20
STD = 0.20, K_Size = 375, W_len = 0.29
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
8
6
4
2
0

2.5.2 | Smoothing Kernel

In this section, the effects of different smoothing kernel lengths are explored given a fixed *std* value.

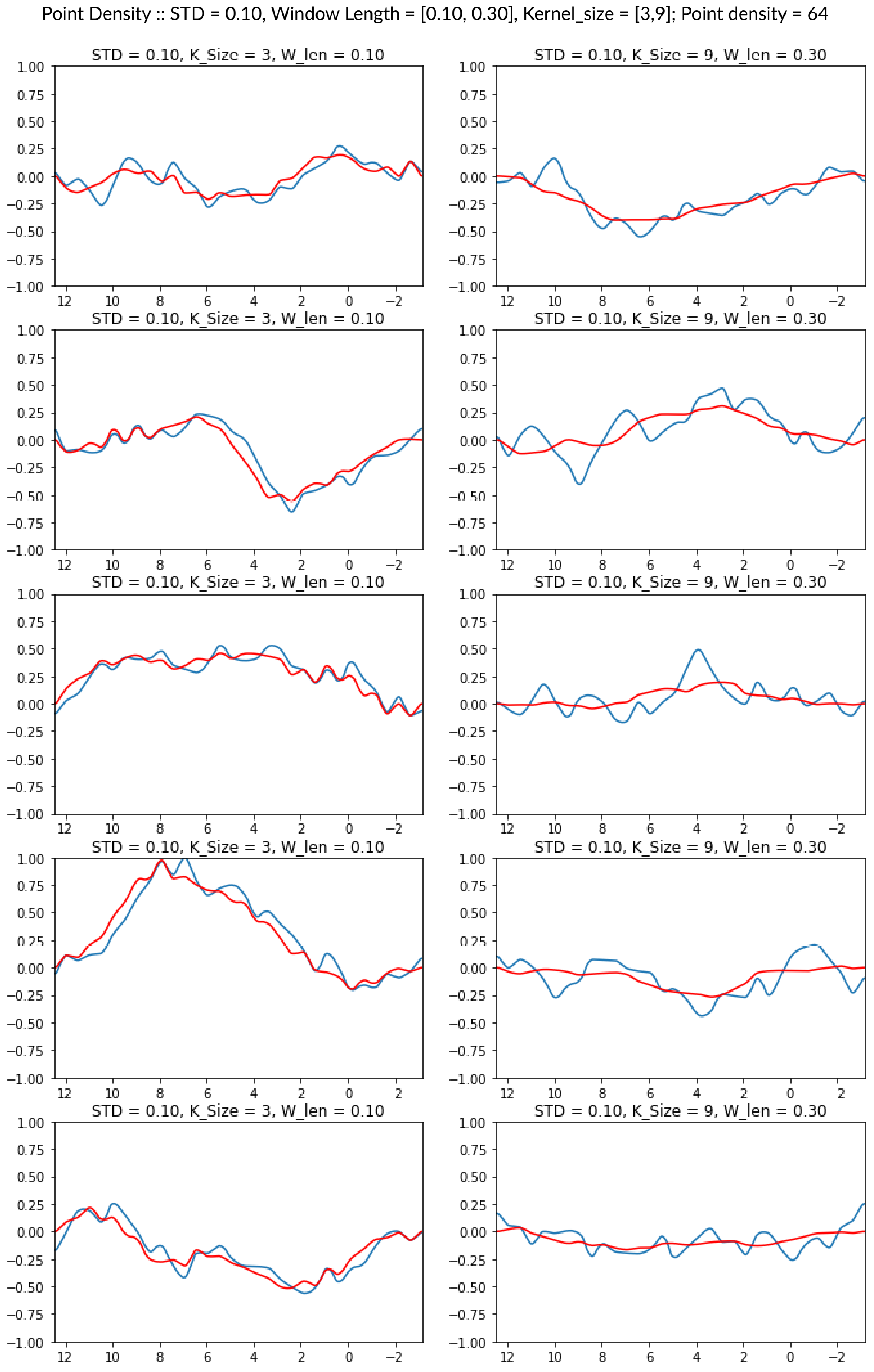


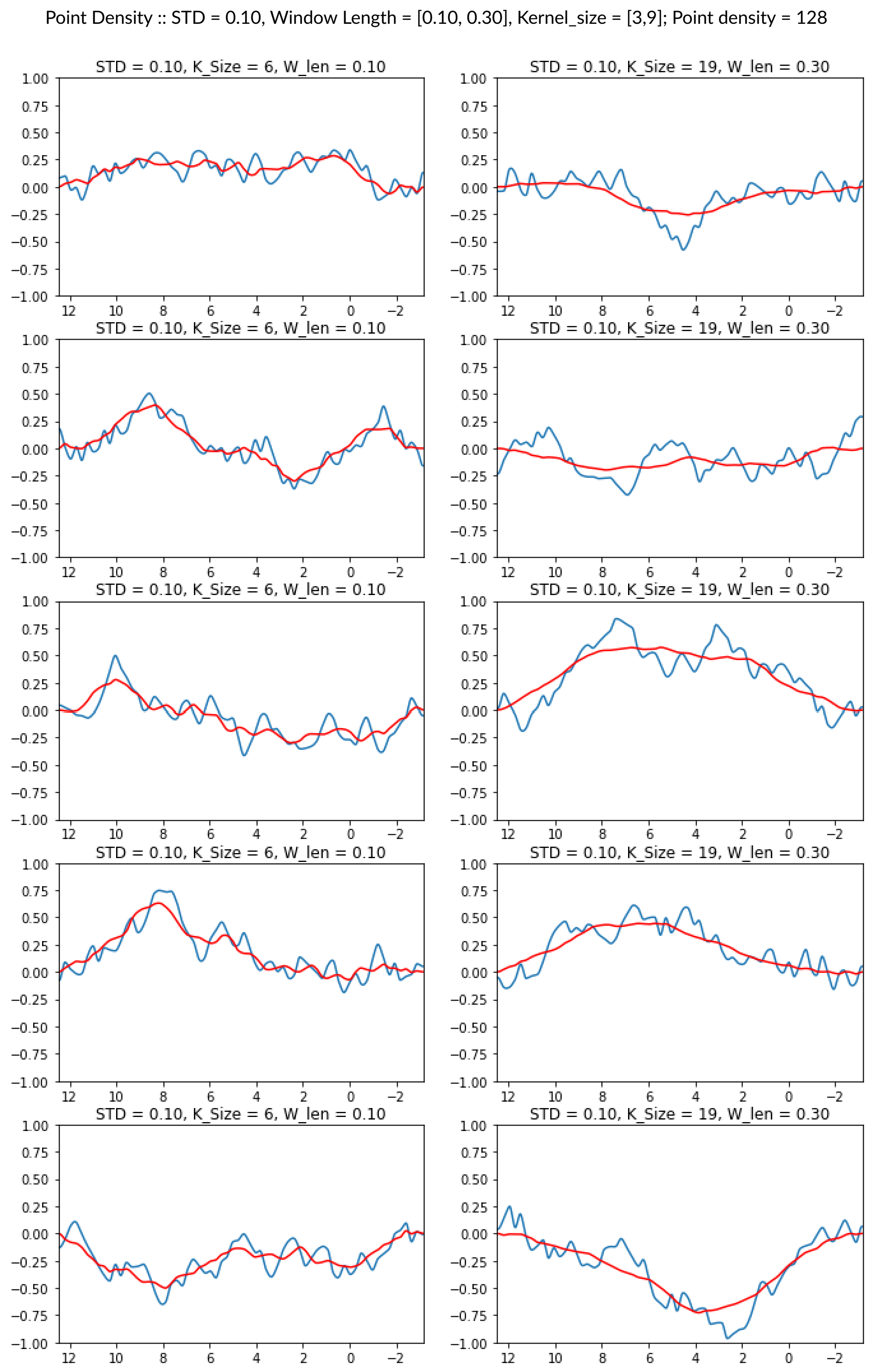

Smoothing Kernel :: STD = 0.10; Window Length = 0.20, Kernel_size = 120; Point density = 128
STD = 0.10, K_Size = 120, W_len = 0.20
STD = 0.10, K_Size = 120, W_len = 0.20
STD = 0.10, K_Size = 120, W_len = 0.20
STD = 0.10, K_Size = 120, W_len = 0.20
STD = 0.10, K_Size = 120, W_len = 0.20
STD = 0.10, K_Size = 120, W_len = 0.20
STD = 0.10, K_Size = 120, W_len = 0.20
STD = 0.10, K_Size = 120, W_len = 0.20
STD = 0.10, K_Size = 120, W_len = 0.20
STD = 0.10, K_Size = 120, W_len = 0.20
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
12
10
8
6
4
2
0
−2


Smoothing Kernel :: STD = 0.10; Window Length = 0.30, Kernel_size = 180; Point density = 128
STD = 0.10, K_Size = 180, W_len = 0.30
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
12
10
8
6
4
2
0
−2

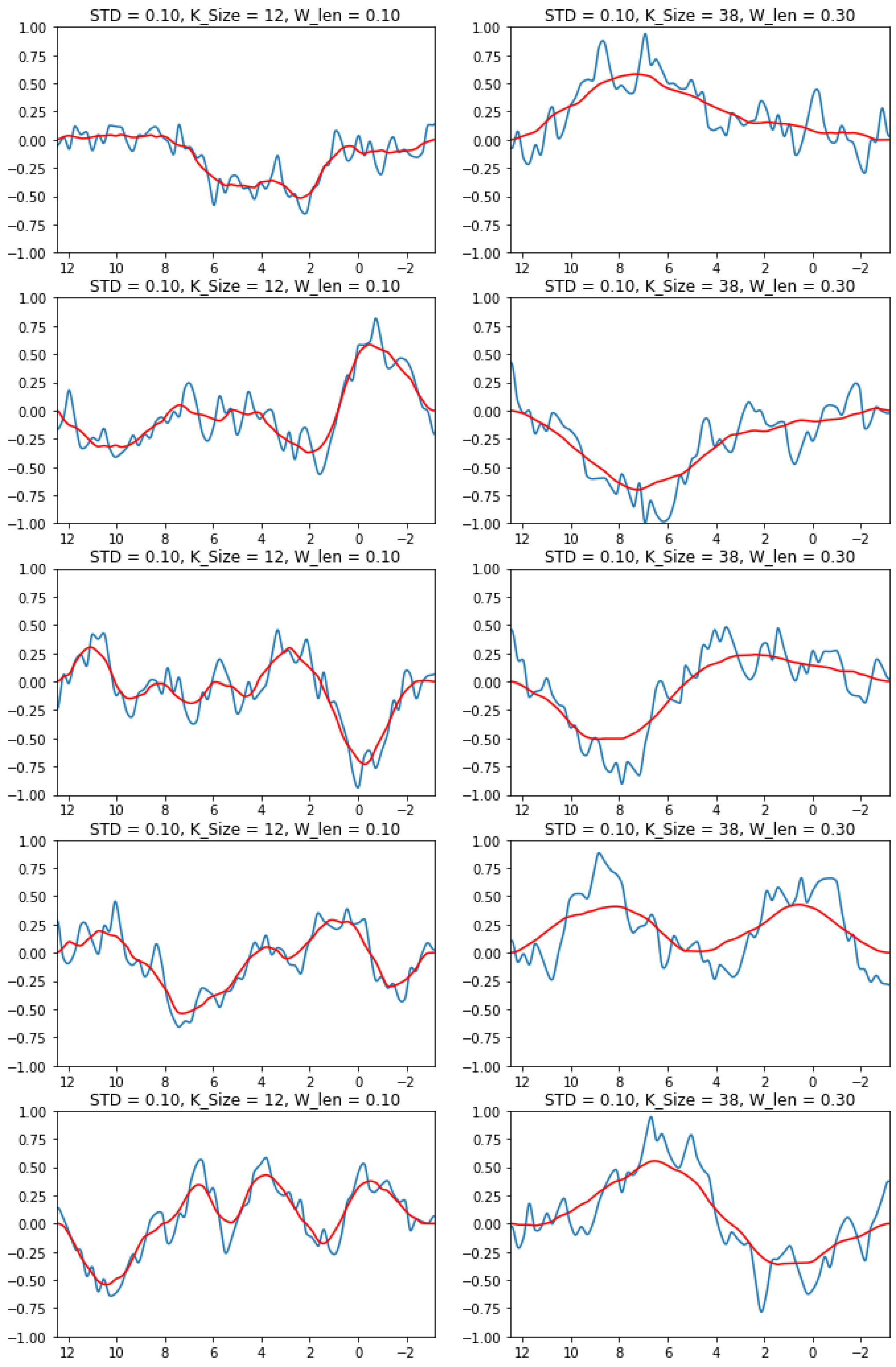
2.5.3 | Point Density

This section explores the effect of varying the point density of the random walk while keeping the *std* fixed. Results are presented using two different kernel sizes for the smoothing.



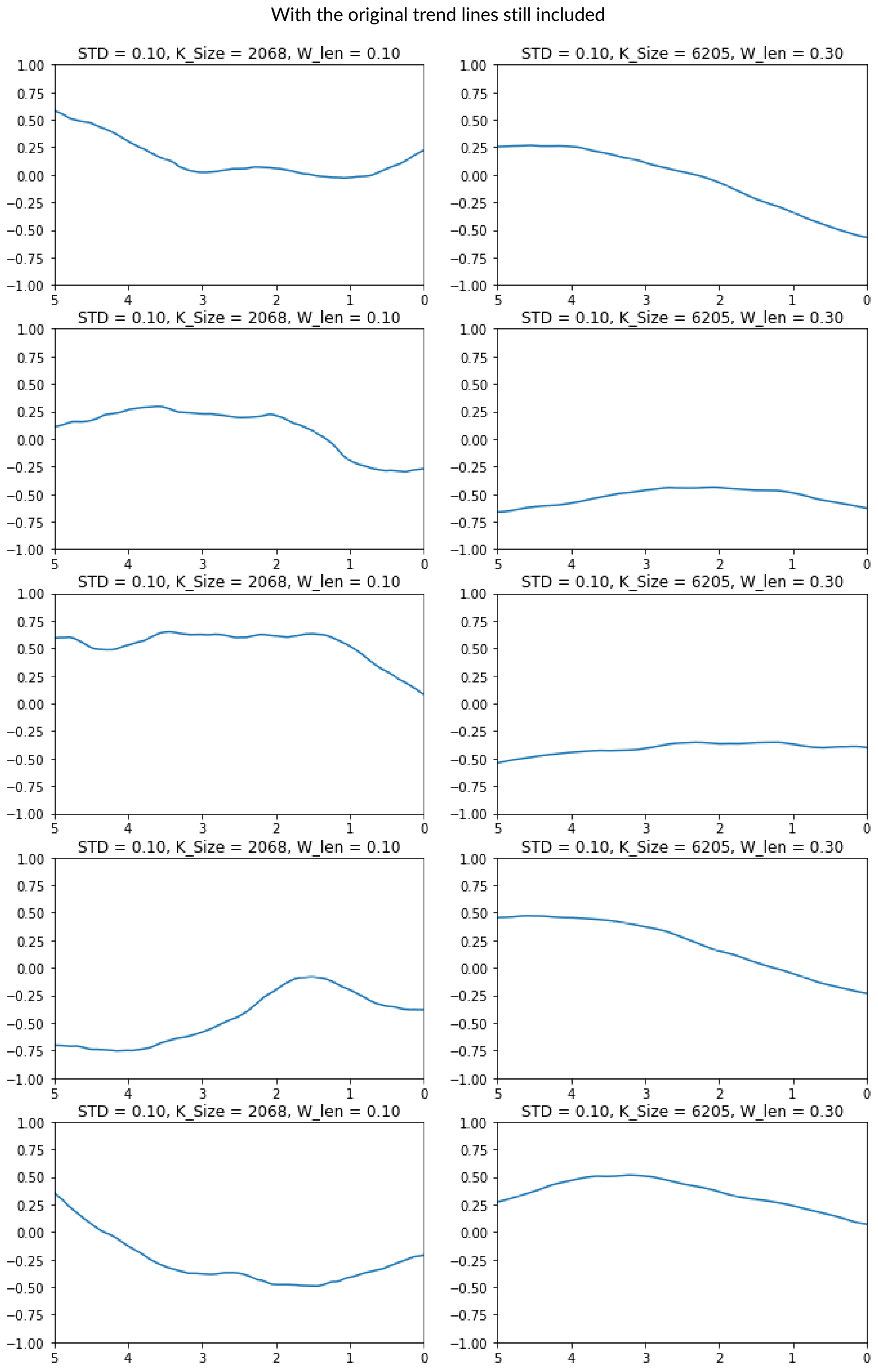

Point Density :: STD = 0.10, Window Length = [0.10, 0.30], Kernel_size = [3,9]; Point density = 128
STD = 0.10, K_Size = 6, W_len = 0.10
STD = 0.10, K_Size = 19, W_len = 0.30
STD = 0.10, K_Size = 6, W_len = 0.10
STD = 0.10, K_Size = 19, W_len = 0.30
STD = 0.10, K_Size = 6, W_len = 0.10
STD = 0.10, K_Size = 19, W_len = 0.30
STD = 0.10, K_Size = 6, W_len = 0.10
STD = 0.10, K_Size = 19, W_len = 0.30
STD = 0.10, K_Size = 6, W_len = 0.10
STD = 0.10, K_Size = 19, W_len = 0.30
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
12
10
8
6
4
2
0
−2

Point Density :: STD = 0.10, Window Length = [0.10, 0.30], Kernel_size = [3,9]; Point density = 256

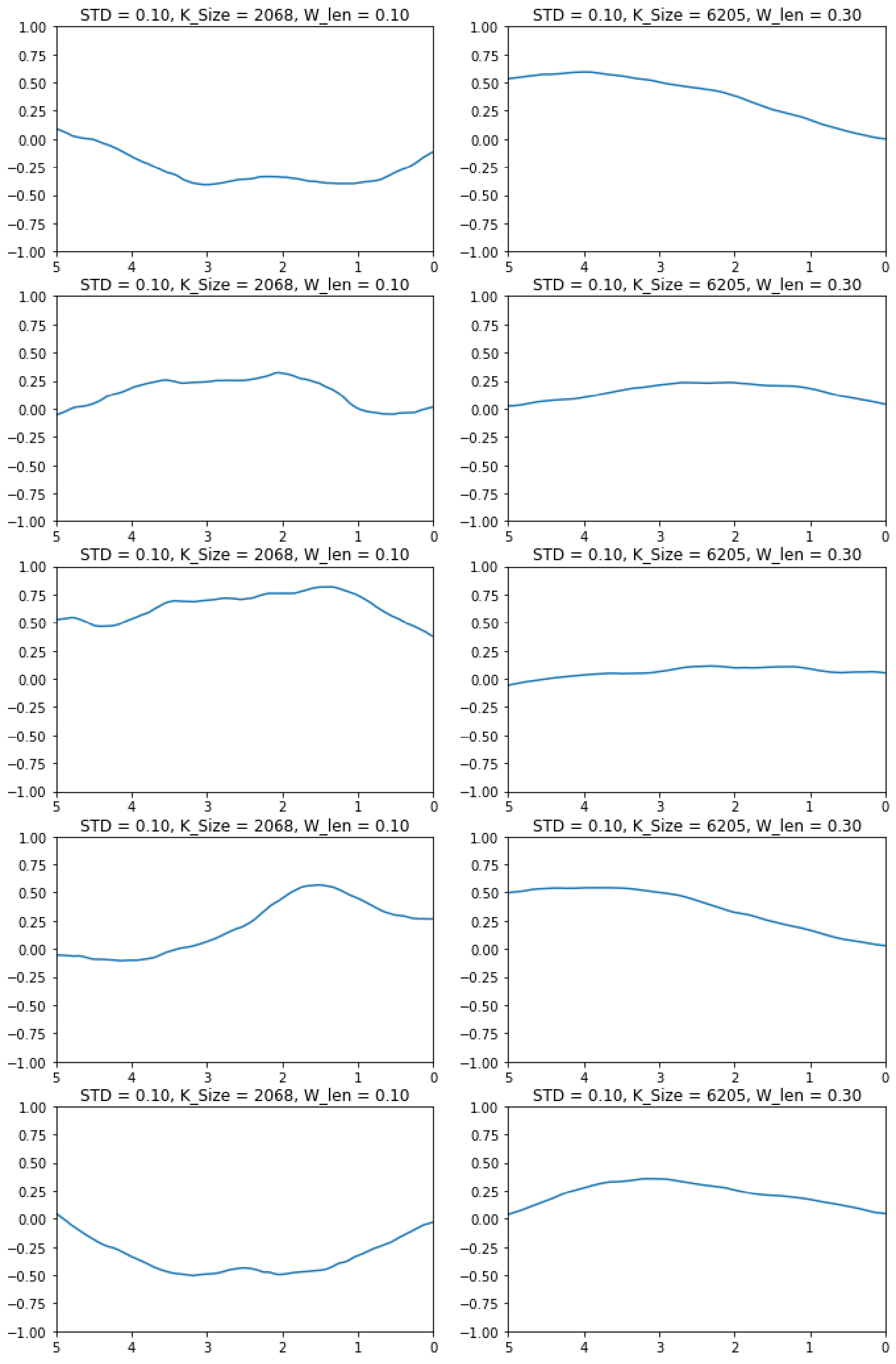


2.5.4 | Incorporating the offsets into the spectra

To increase variability, the starting and ending heights are randomly selected. When considering the entire spectrum, however, these must be adjusted to avoid unrealistic, hard transition points.

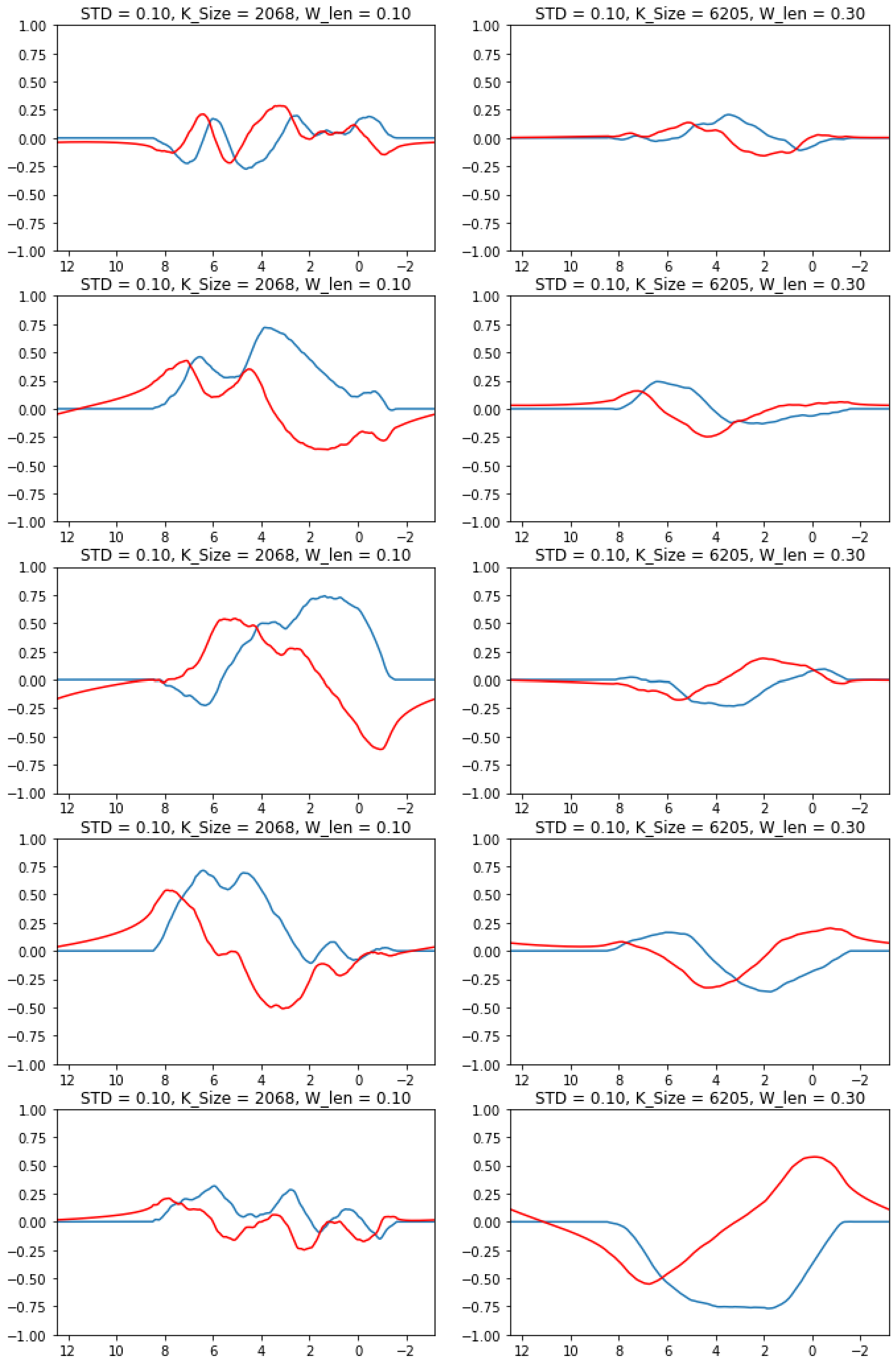


With the original trend lines removed



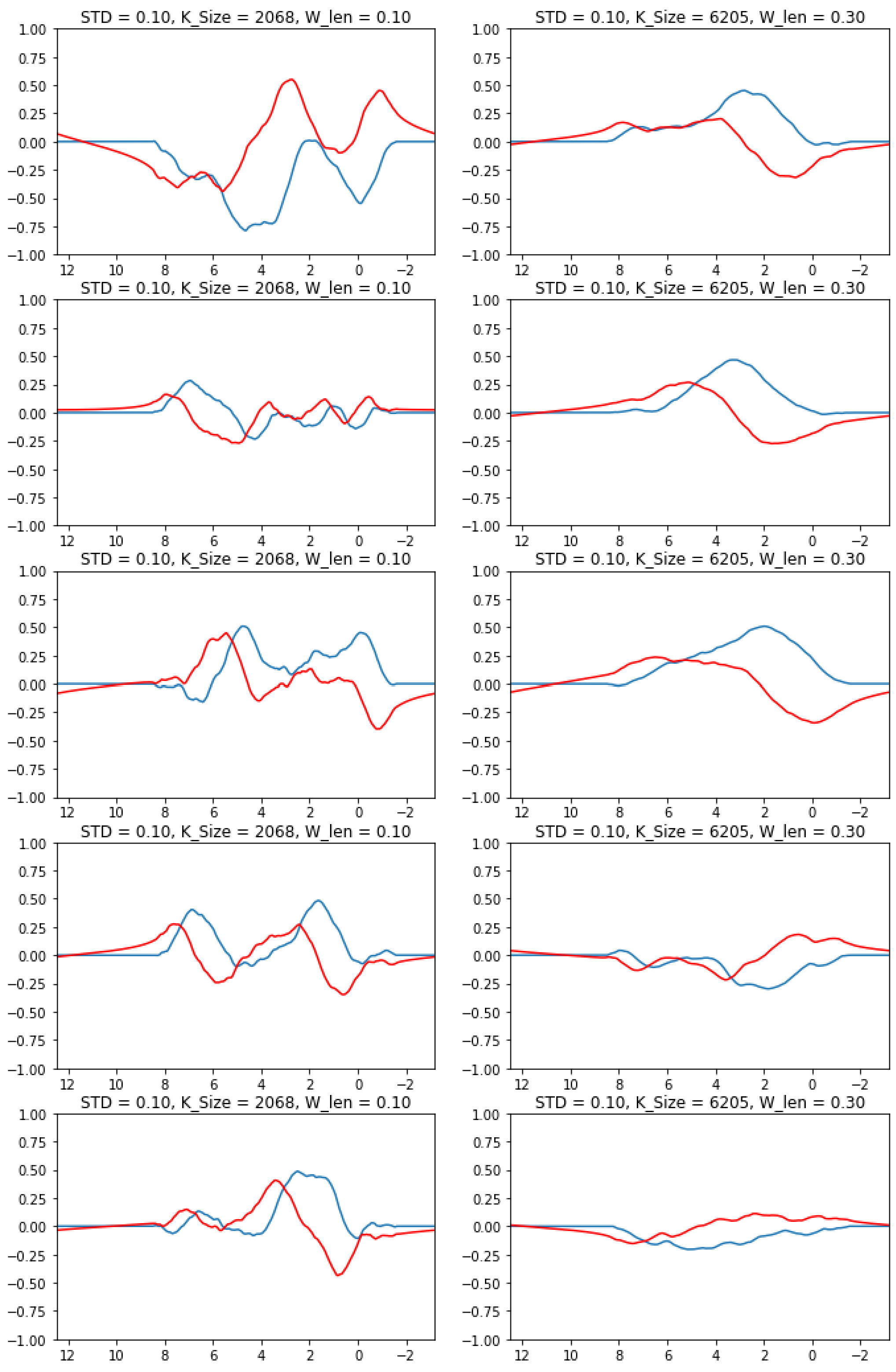
2.5.5 | Effect of the Hilbert Transform

Simulating spectra requires complex spectral components, including the baseline and residual water. The Hilbert transform is used to generate those corresponding imaginary components.



2.5.6 | Compiled Generator

This shows the variety of baselines that can be generated when randomly sampling all variables.



2.6 | Residual Water

The following section will explore this generator using the residual water dictionary defined below. While the sections are the same as in Sec. 2.5, the values displayed are tailored to the ranges for the residual water simulations.

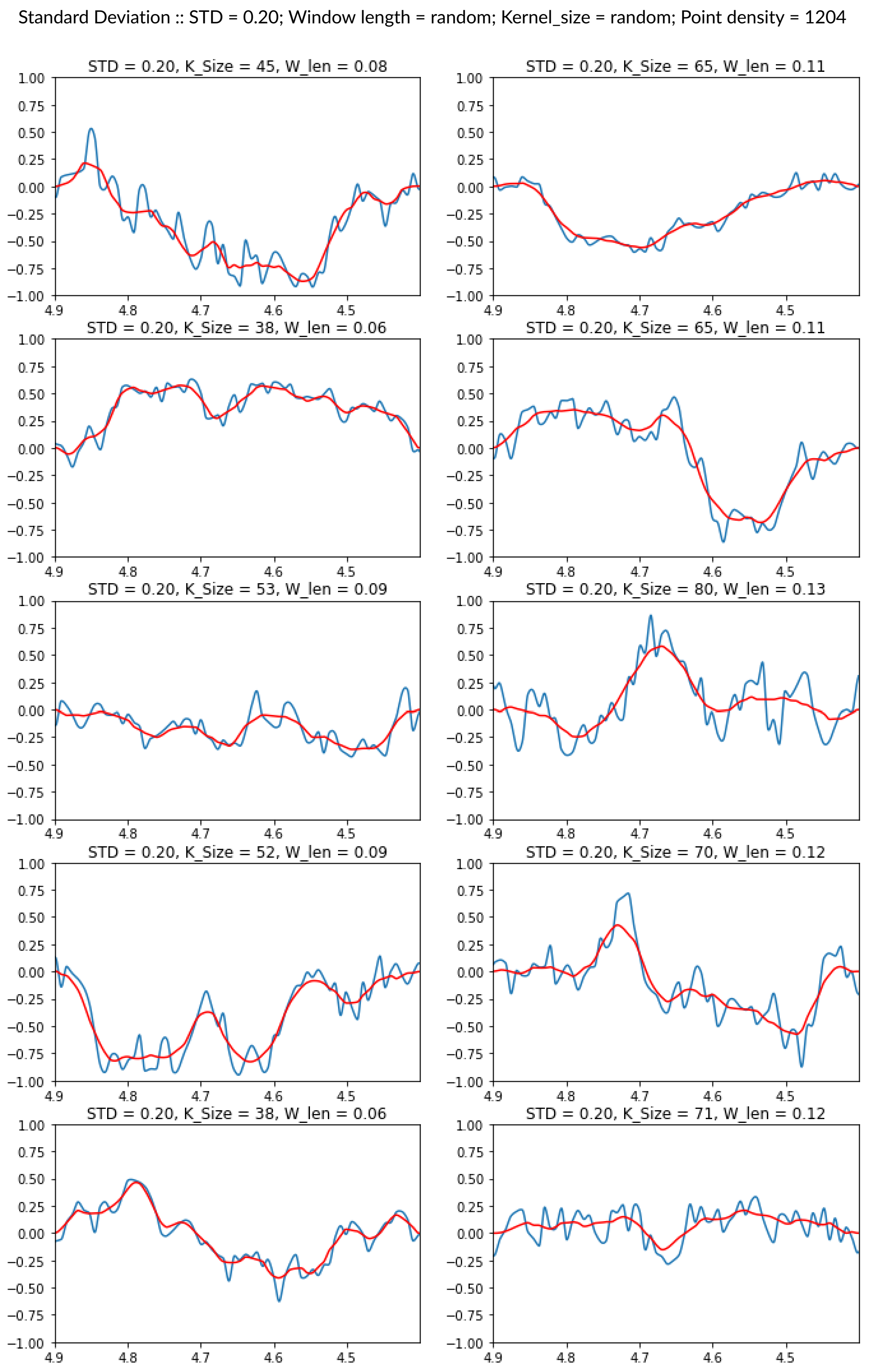
2.6.1 | Configuration dictionaries

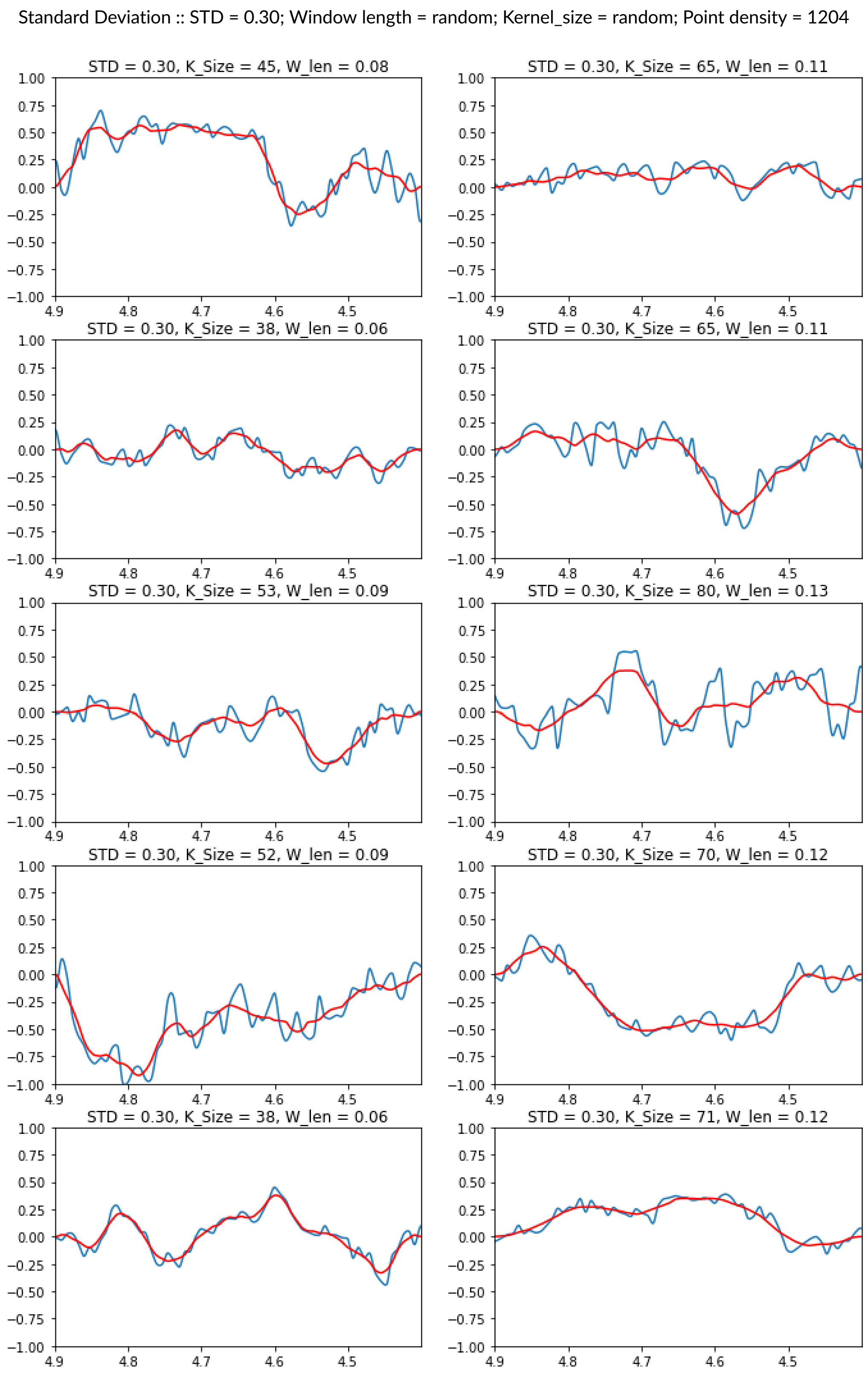
The following dictionaries are the standard defaults in MRS-Sim for generating baseline offsets and residual water contributions. To explore how the models output change with the different dictionaries, toggle between the two cfg options in the Section 2.3 below.

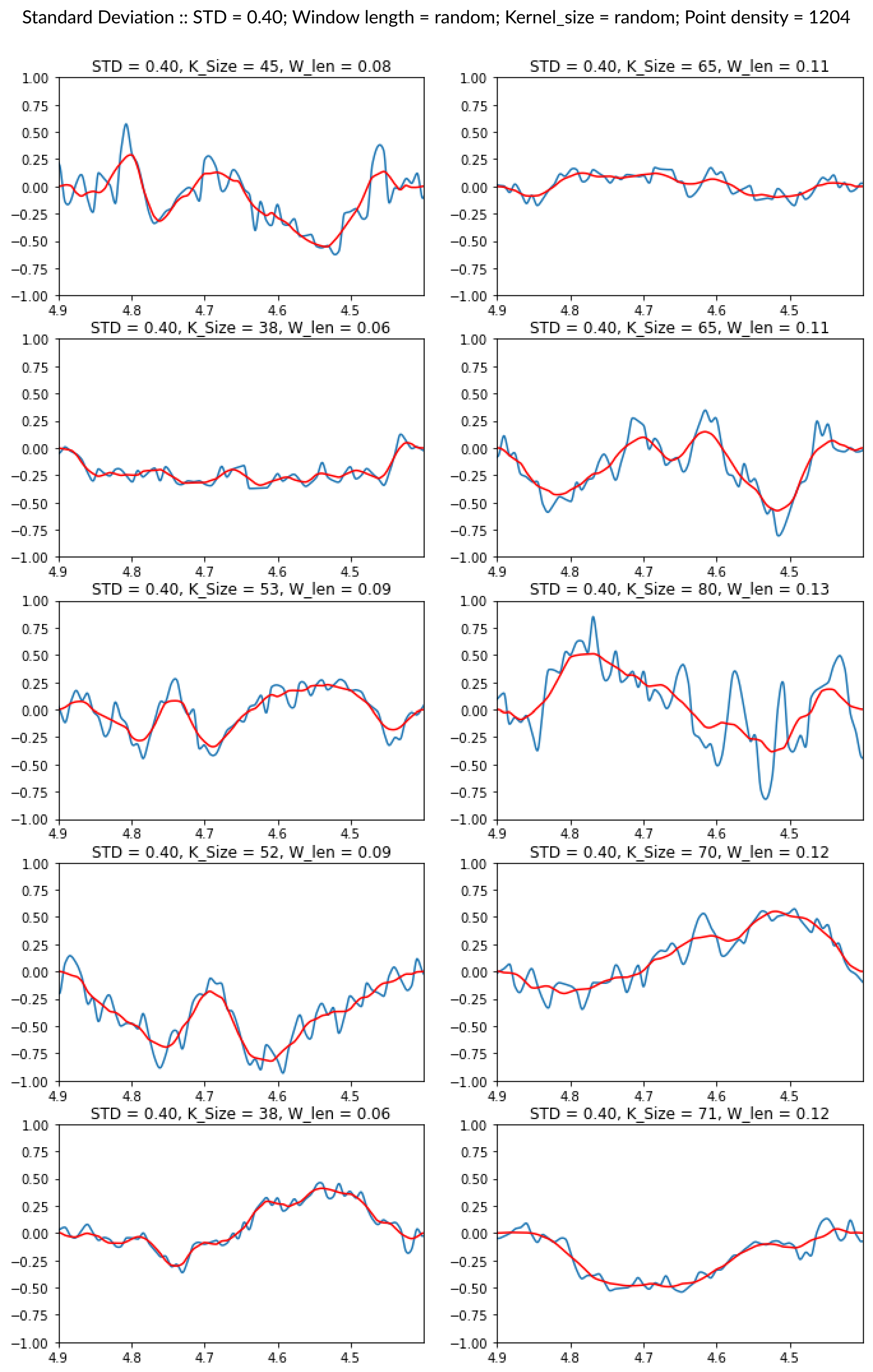
```
resWater_cfg = {
        "start":            [          0],
        "end":              [          0],
        "upper":            [    0,    1],
        "lower":            [    0,    1],
        "std":              [  0.2,  0.40],
        "window":           [ 0.05,  0.15],
        "pt_density":         1204,
        "ppm_range":        [  4.4,   4.9],
        "prime":              0.15,
        "scale":            [  1.0,   1.0], # typically: [0.05, 0.40] - better visualization.
        "drop_prob":           0.0
}
```

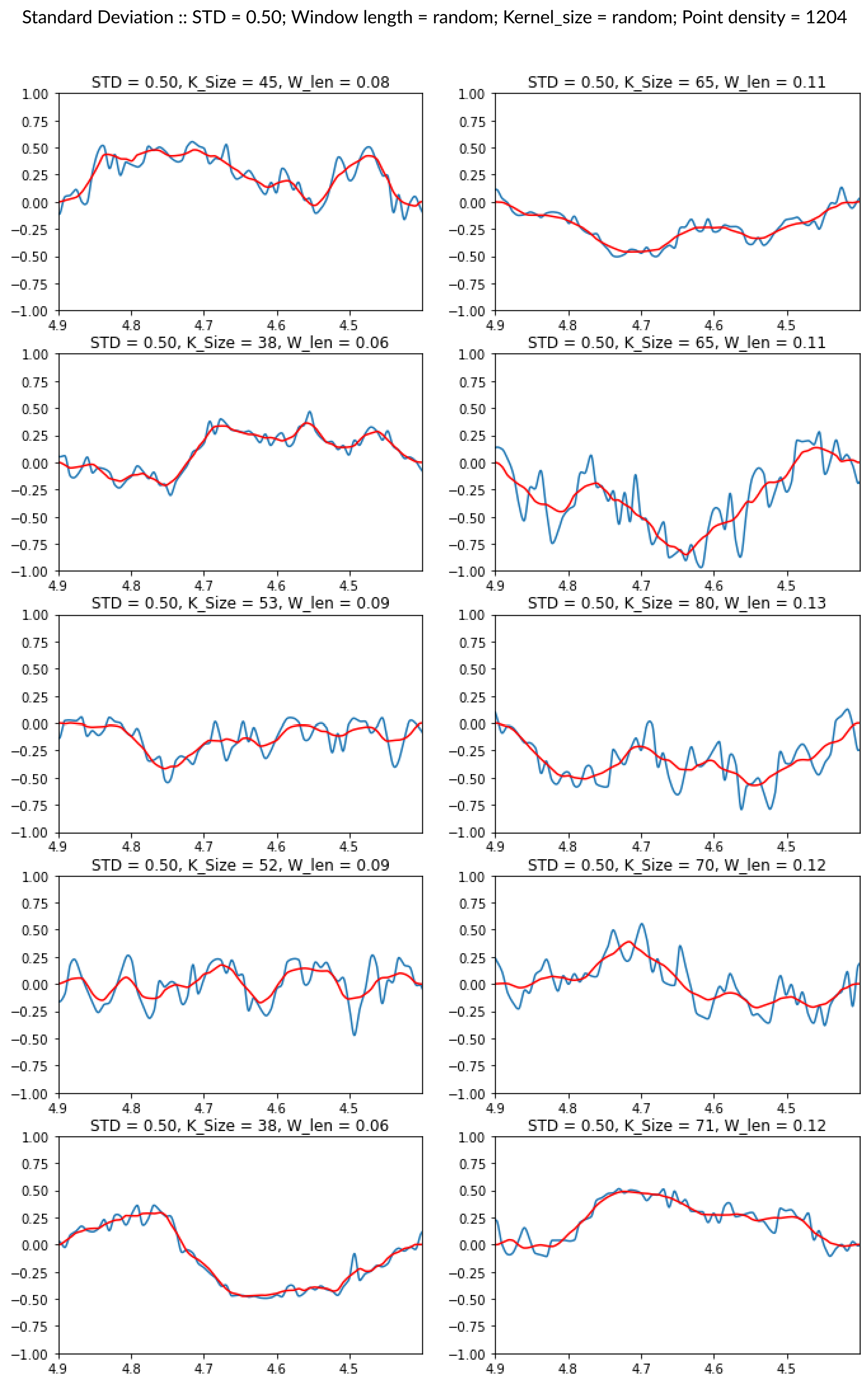
2.6.2 | Standard Deviation

This section shows the effect of different std values on the random walk. The *std* variable is used when sampling the noise for the walk. It directly controls the amount of variation between two consecutive points prior to the cumulative summation.


Standard Deviation :: STD = 0.20; Window length = random; Kernel_size = random; Point density = 1204
STD = 0.20, K_Size = 45, W_len = 0.08
STD = 0.20, K_Size = 65, W_len = 0.11
STD = 0.20, K_Size = 38, W_len = 0.06
STD = 0.20, K_Size = 65, W_len = 0.11
STD = 0.20, K_Size = 53, W_len = 0.09
STD = 0.20, K_Size = 80, W_len = 0.13
STD = 0.20, K_Size = 52, W_len = 0.09
STD = 0.20, K_Size = 70, W_len = 0.12
STD = 0.20, K_Size = 38, W_len = 0.06
STD = 0.20, K_Size = 71, W_len = 0.12
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
4.9
4.8
4.7
4.6
4.5

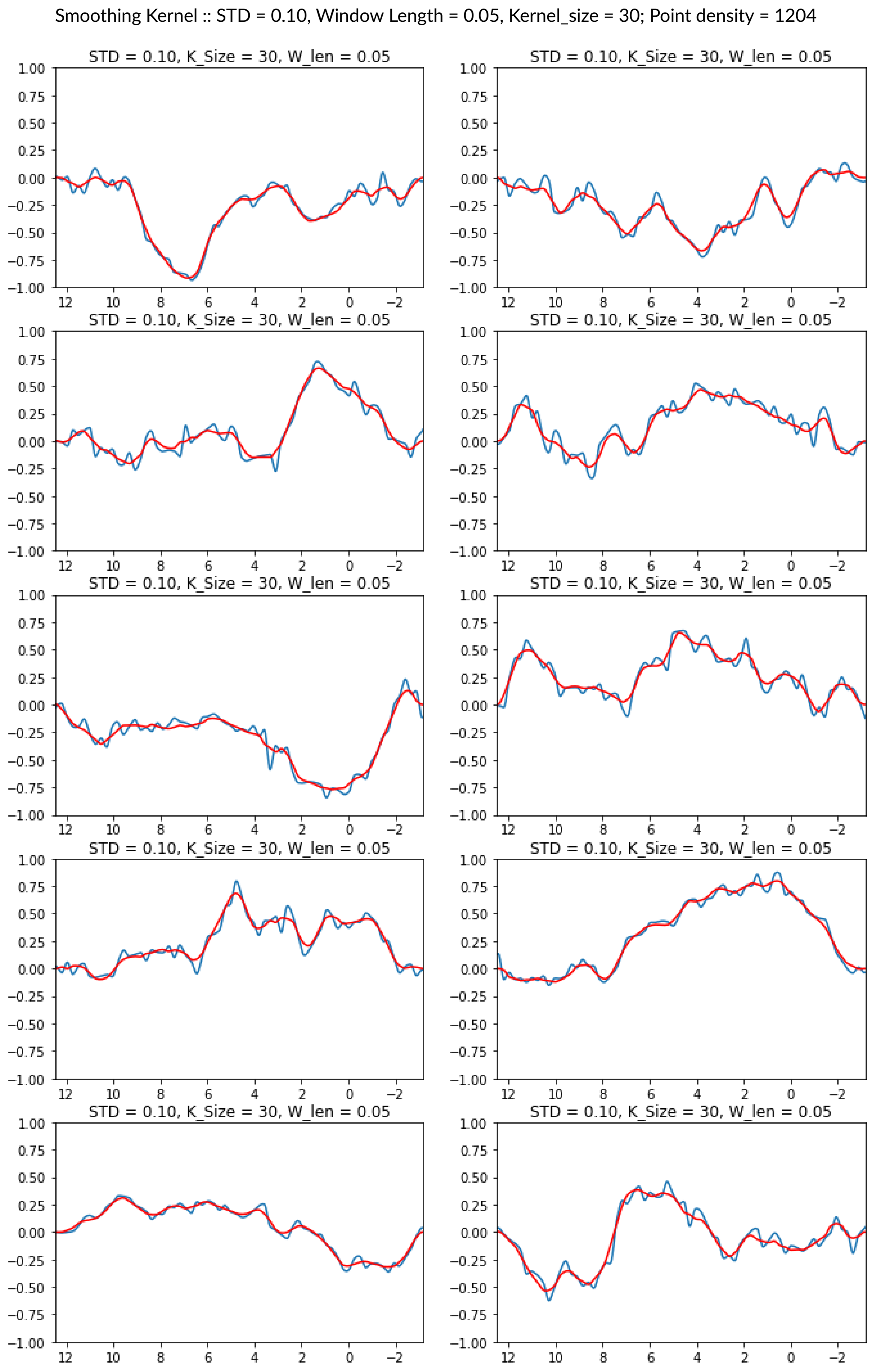

Standard Deviation :: STD = 0.30; Window length = random; Kernel_size = random; Point density = 1204
STD = 0.30, K_Size = 45, W_len = 0.08
STD = 0.30, K_Size = 65, W_len = 0.11
STD = 0.30, K_Size = 38, W_len = 0.06
STD = 0.30, K_Size = 65, W_len = 0.11
STD = 0.30, K_Size = 53, W_len = 0.09
STD = 0.30, K_Size = 80, W_len = 0.13
STD = 0.30, K_Size = 52, W_len = 0.09
STD = 0.30, K_Size = 70, W_len = 0.12
STD = 0.30, K_Size = 38, W_len = 0.06
STD = 0.30, K_Size = 71, W_len = 0.12
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
4.9
4.8
4.7
4.6
4.5


Standard Deviation :: STD = 0.40; Window length = random; Kernel_size = random; Point density = 1204
STD = 0.40, K_Size = 45, W_len = 0.08
STD = 0.40, K_Size = 65, W_len = 0.11
STD = 0.40, K_Size = 38, W_len = 0.06
STD = 0.40, K_Size = 65, W_len = 0.11
STD = 0.40, K_Size = 53, W_len = 0.09
STD = 0.40, K_Size = 80, W_len = 0.13
STD = 0.40, K_Size = 52, W_len = 0.09
STD = 0.40, K_Size = 70, W_len = 0.12
STD = 0.40, K_Size = 38, W_len = 0.06
STD = 0.40, K_Size = 71, W_len = 0.12
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
4.9
4.8
4.7
4.6
4.5

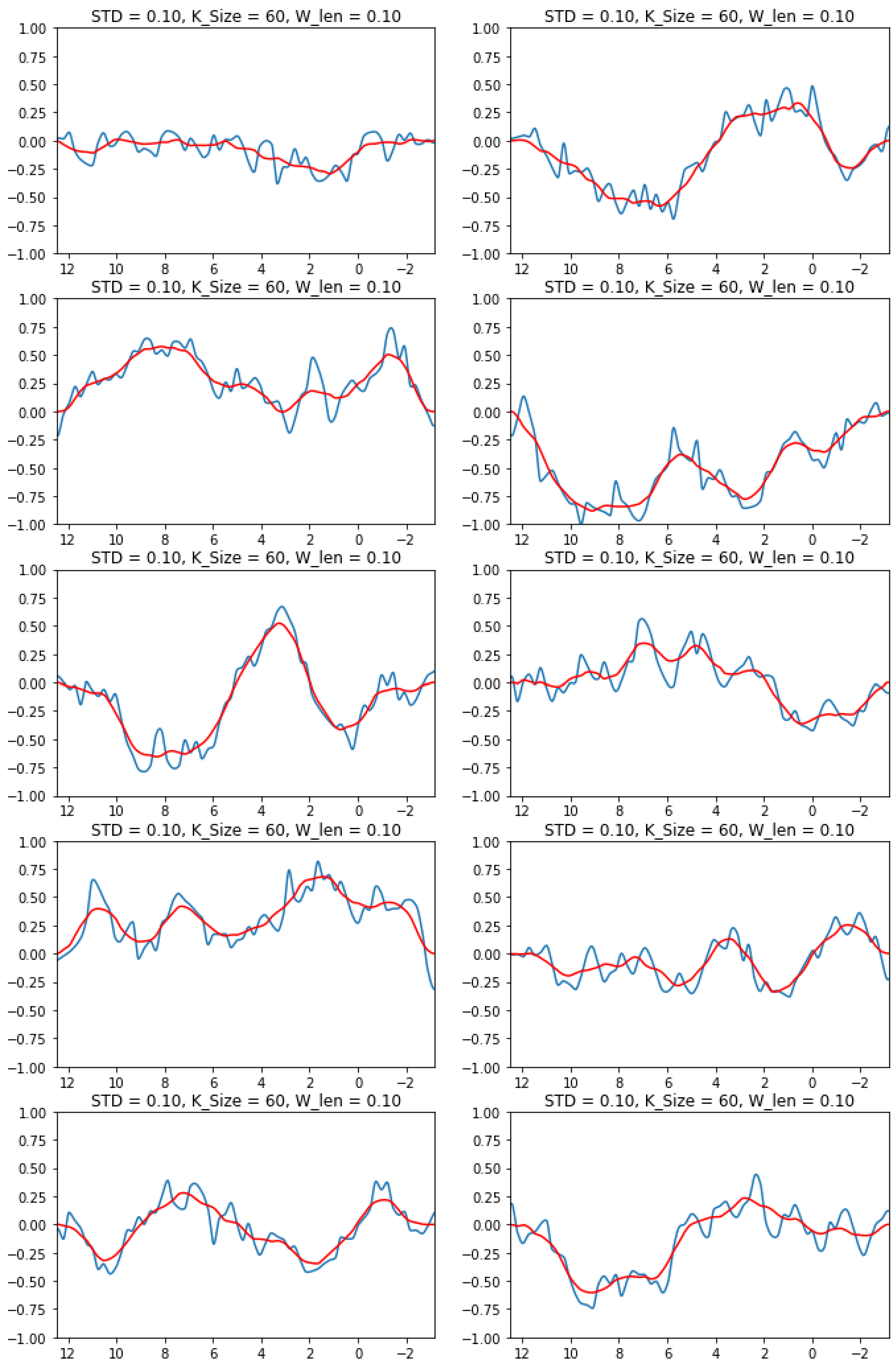

Standard Deviation :: STD = 0.50; Window length = random; Kernel_size = random; Point density = 1204
STD = 0.50, K_Size = 45, W_len = 0.08
STD = 0.50, K_Size = 65, W_len = 0.11
STD = 0.50, K_Size = 38, W_len = 0.06
STD = 0.50, K_Size = 65, W_len = 0.11
STD = 0.50, K_Size = 53, W_len = 0.09
STD = 0.50, K_Size = 80, W_len = 0.13
STD = 0.50, K_Size = 52, W_len = 0.09
STD = 0.50, K_Size = 70, W_len = 0.12
STD = 0.50, K_Size = 38, W_len = 0.06
STD = 0.50, K_Size = 71, W_len = 0.12
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
4.9
4.8
4.7
4.6
4.5

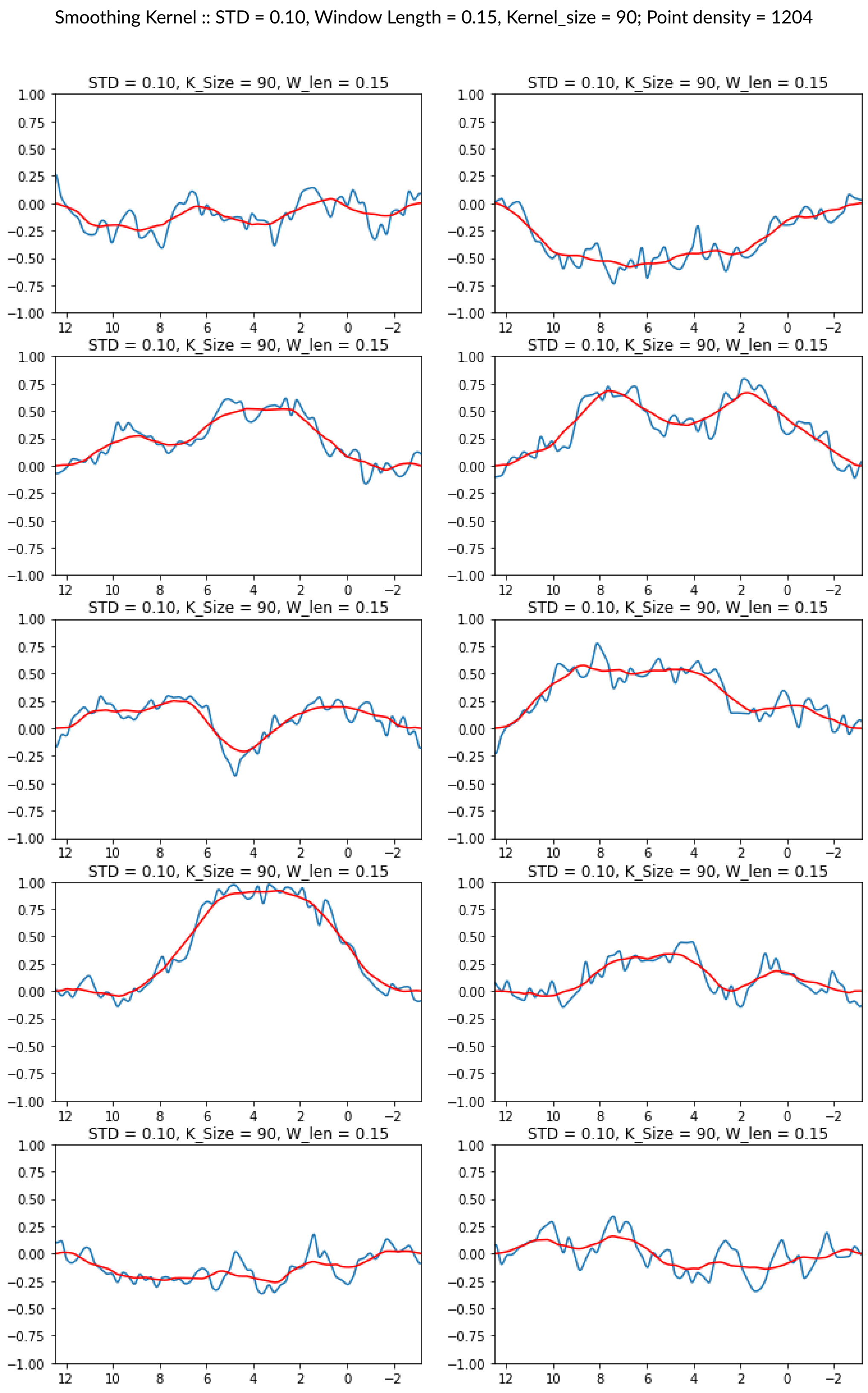
2.6.3 | Smoothing Kernel

In this section, the effects of different smoothing kernel lengths are explored given a fixed *std* value.



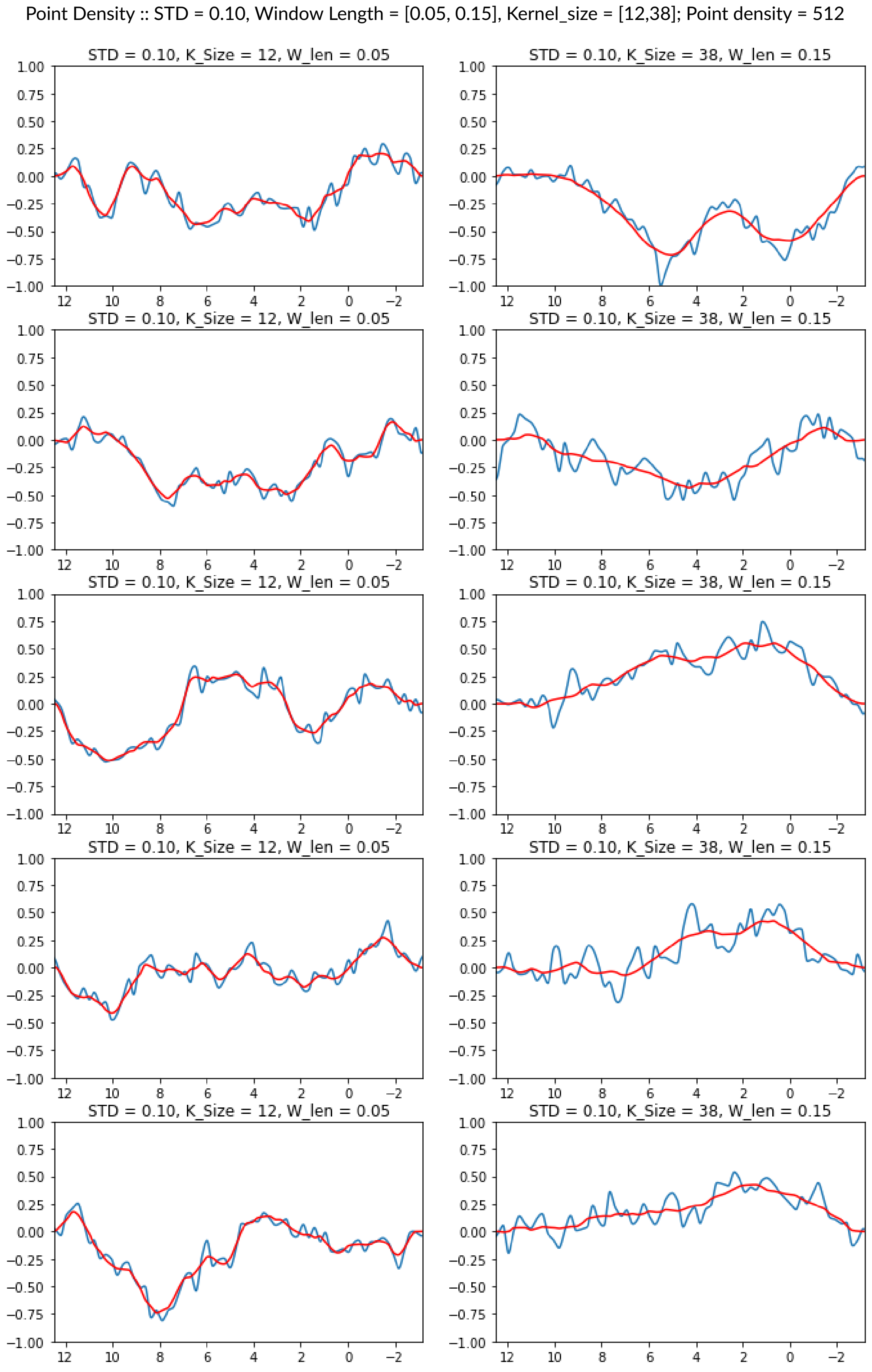
Smoothing Kernel :: STD = 0.10, Window Length = 0.10, Kernel_size = 60; Point density = 1204



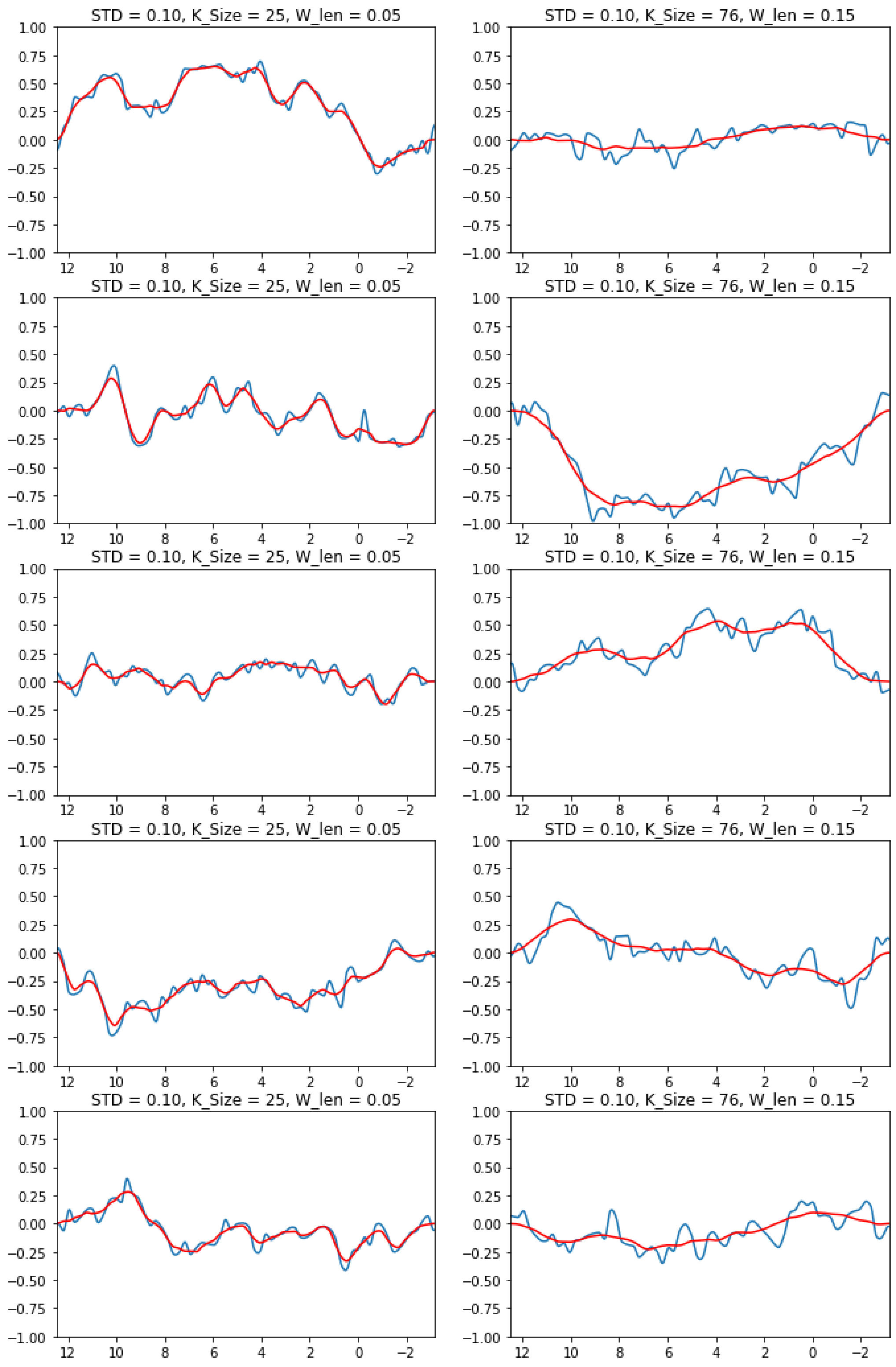

Smoothing Kernel :: STD = 0.10, Window Length = 0.15, Kernel_size = 90; Point density = 1204
STD = 0.10, K_Size = 90, W_len = 0.15
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
12
10
8
6
4
2
0
−2

2.6.4 | Point Density

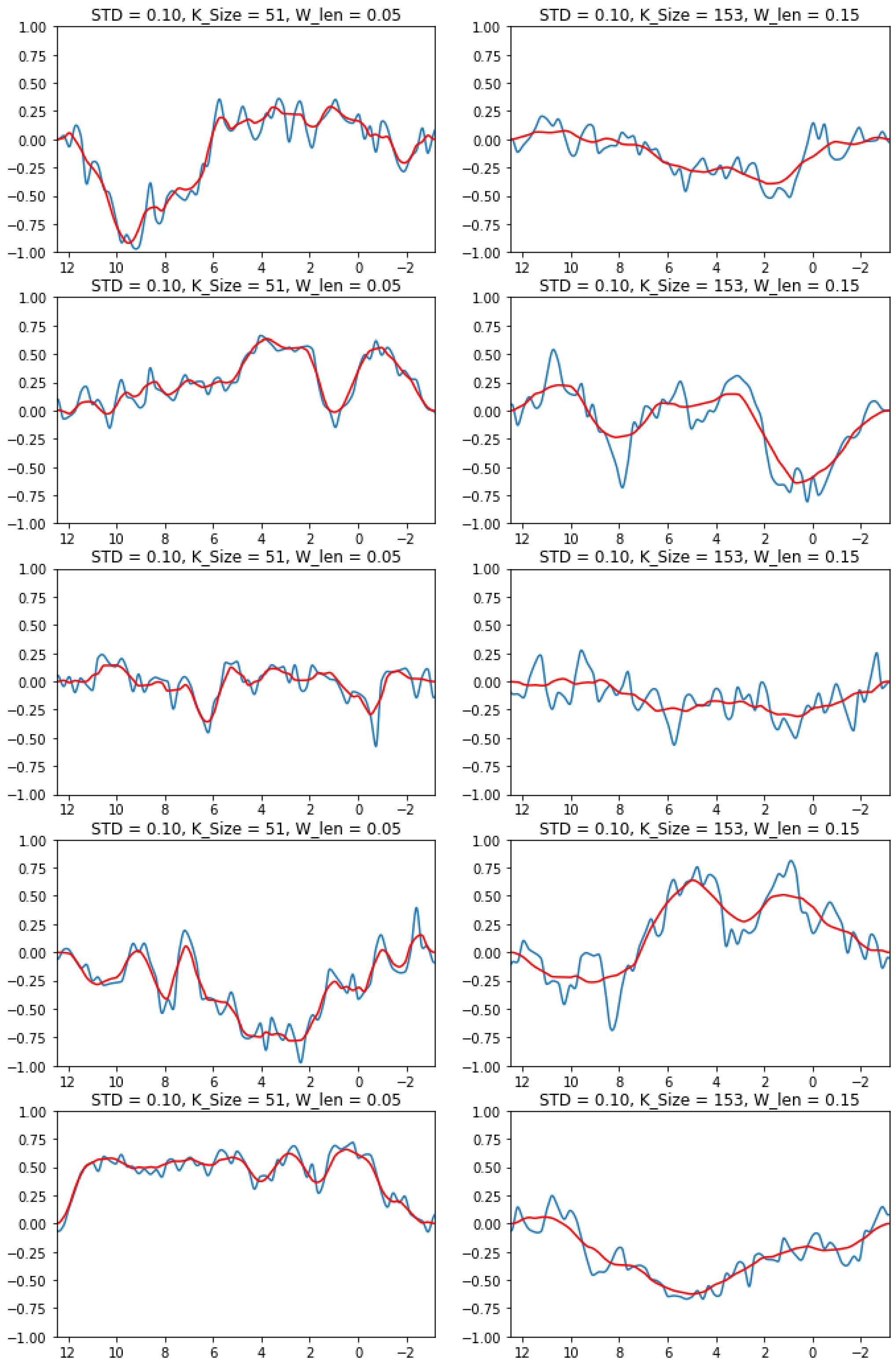
This section explores the effect of varying the point density of the random walk while keeping the *std* fixed. Results are presented using two different kernel sizes for the smoothing.



Point Density :: STD = 0.10, Window Length = [0.05, 0.15], Kernel_size = [12,38]; Point density = 1024

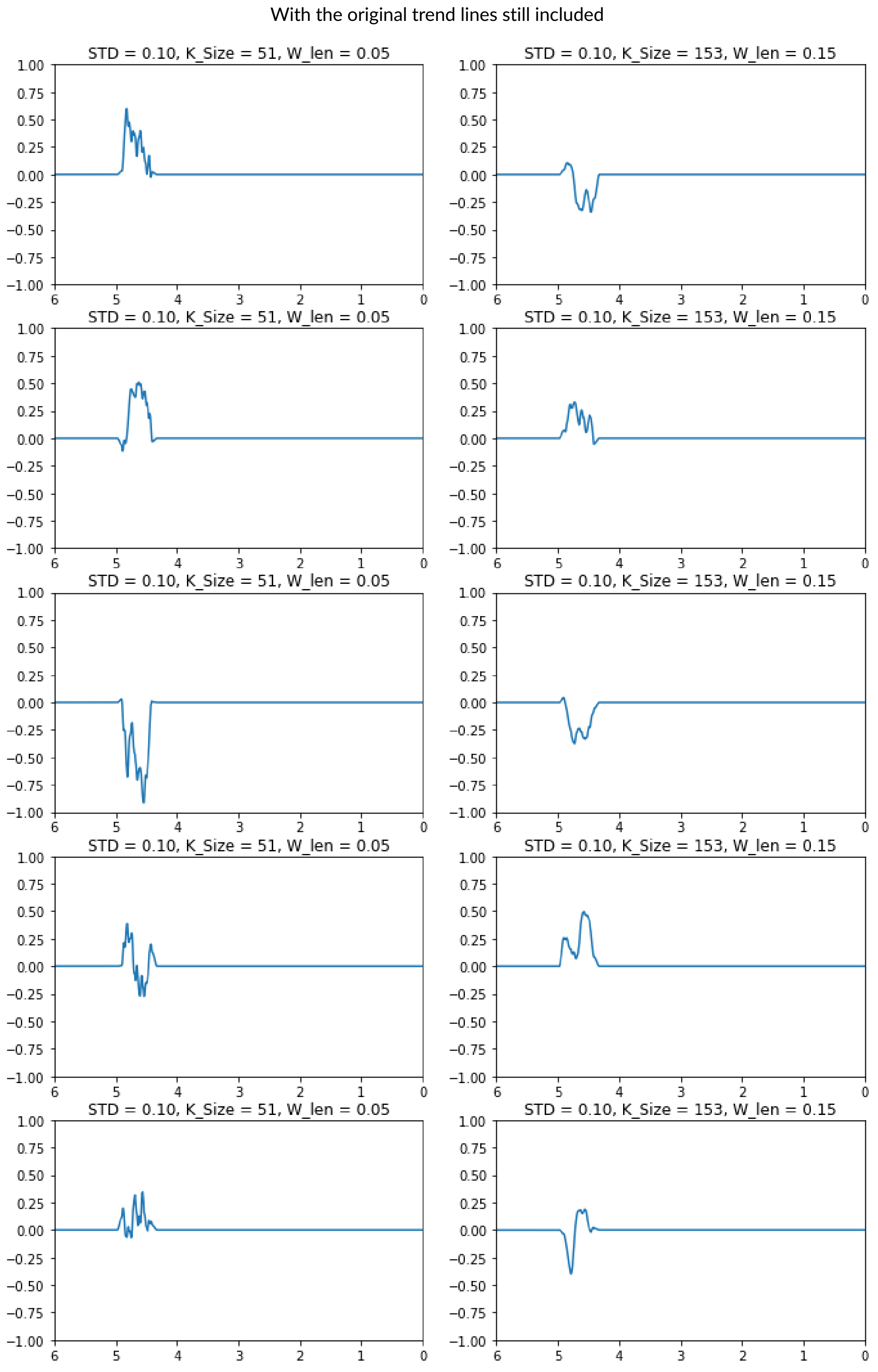


Point Density :: STD = 0.10, Window Length = [0.05, 0.15], Kernel_size = [12,38]; Point density = 2048

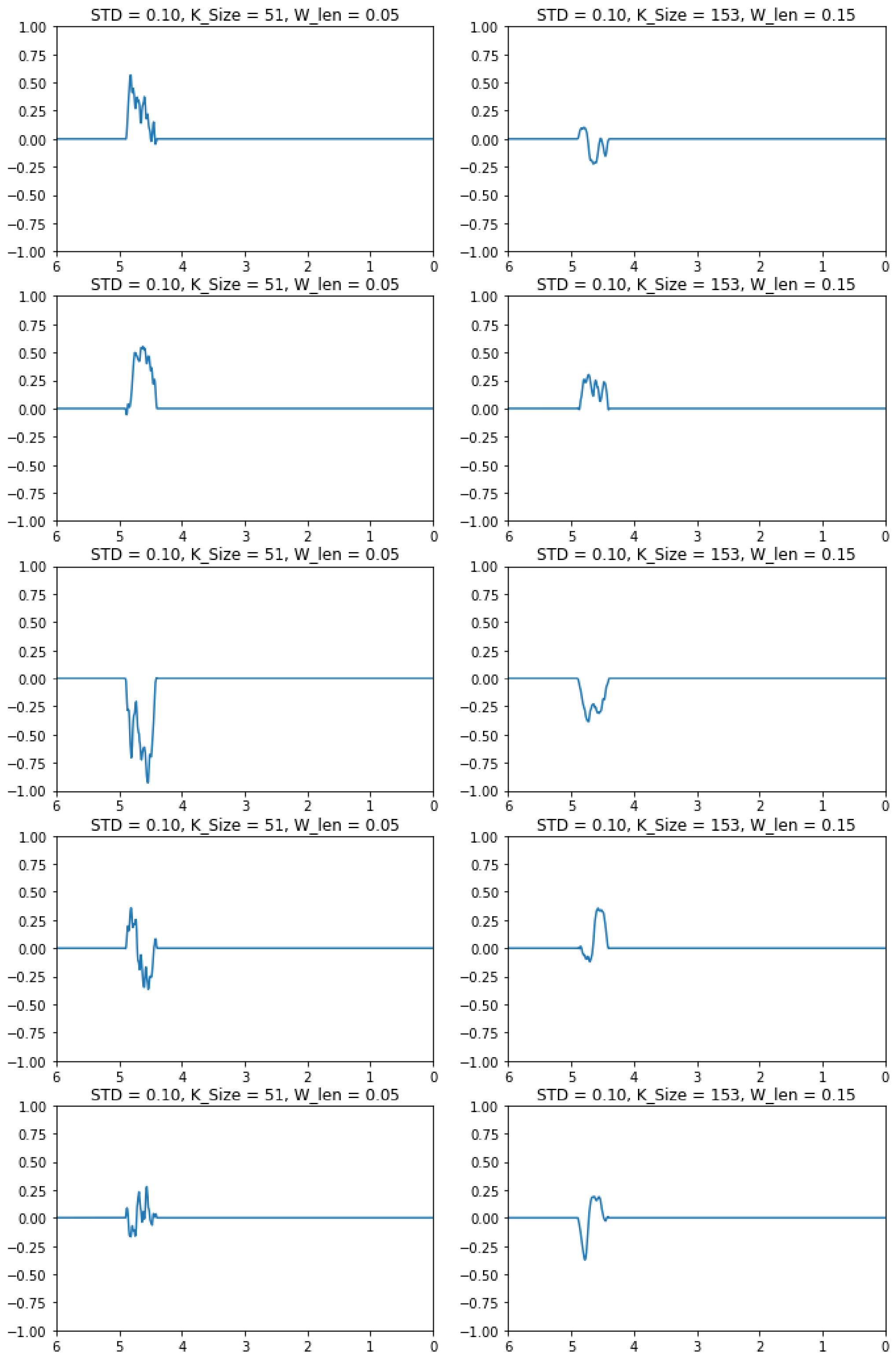


2.6.5 | Incorporating the offsets into the spectra

To increase variability, the starting and ending heights are randomly selected. When considering the entire spectrum, however, these must be adjusted to avoid unrealistic, hard transition points.

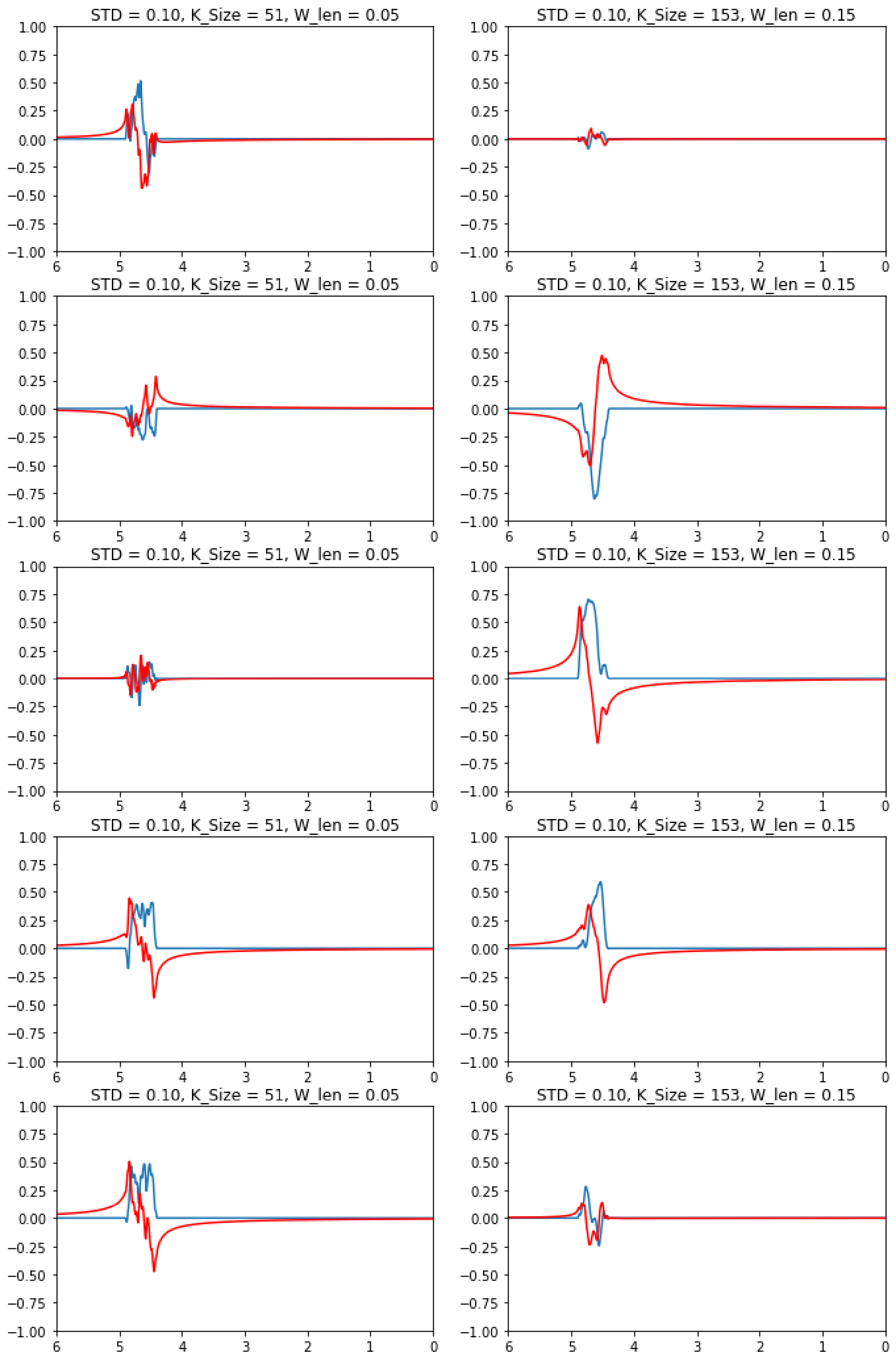


With the original trend lines removed



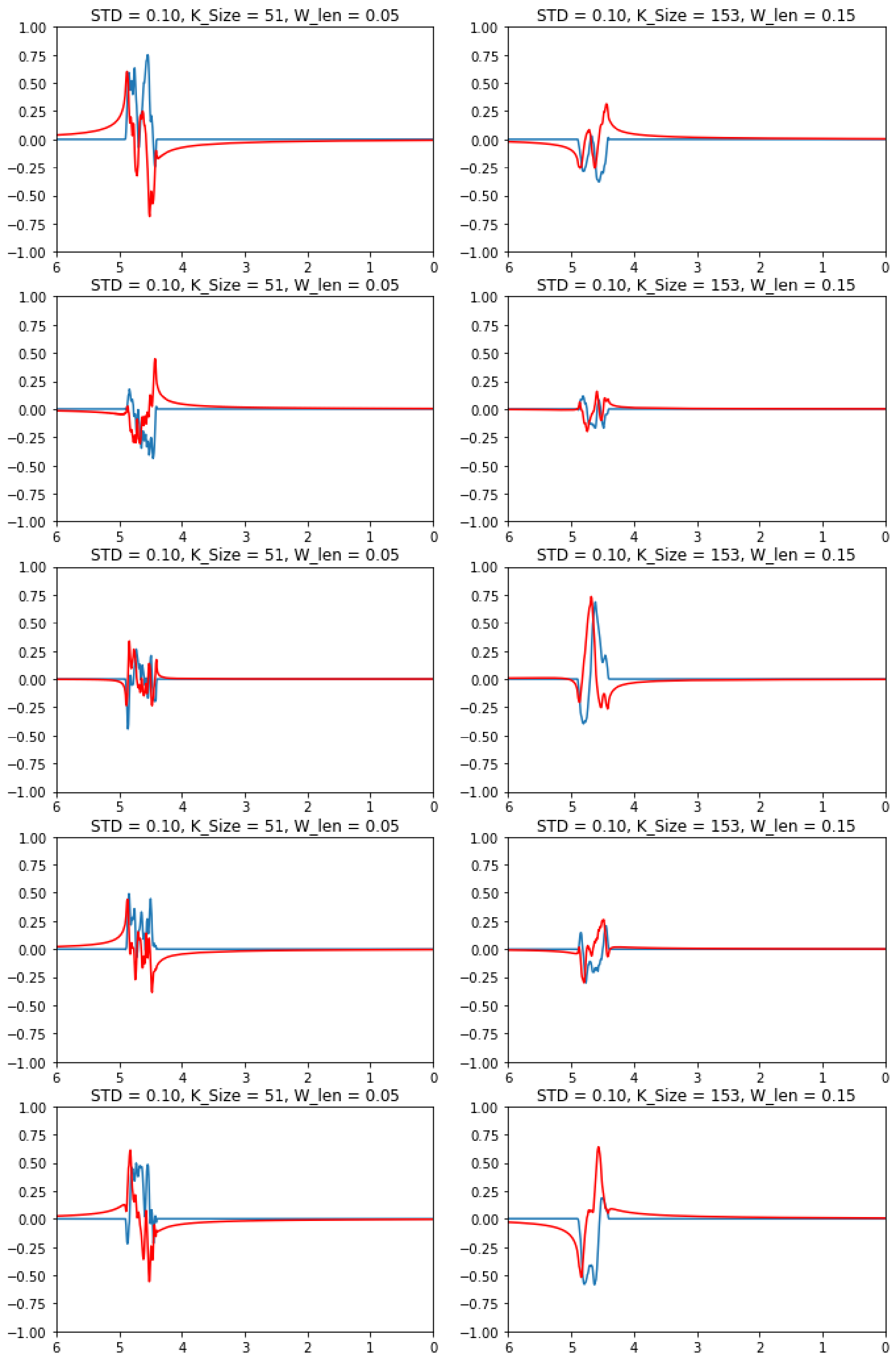
2.6.6 | Effect of the Hilbert Transform

Simulating spectra requires complex spectral components, including the baseline and residual water. The Hilbert transform is used to generate those corresponding imaginary components.



2.6.7 | Compiled Generator

This shows the variety of residual water contributions that can be generated when randomly sampling all variables.



$\square$