

AARHUS UNIVERSITY

COMPUTER-SCIENCE

NUMERICAL LINEAR ALGEBRA

Handin 2

Author

Søren M. DAMSGAARD

Student number

202309814

February 17, 2026



Noisy Data

The author are given a signal on the form

$$y(x) = 12 \sin(x) - 4 \cos(8x) \text{ for } 0 \leq x \leq 12 \quad (1)$$

And are tasked with adding noise to it and then showing how to reduce said noise and explain the theory behind it.

Disclaimer!

The code will not be shown throughout the report, but will be put in the appendix and referred to when necessary. This is because the author is using this change of format, in contrast to the last handin, as an excuse to not do the actual handin.

Do note that the code is commented in such a way that it will be referred to as sections, such as FIGURE 1 LINE 42-45.

a - Plotting the Signal

The code used to define the signal is section SIGNAL LINE 4-9 in the Appendix, while plotting of the figure can be found in FIGURE 1 LINE 42-45.

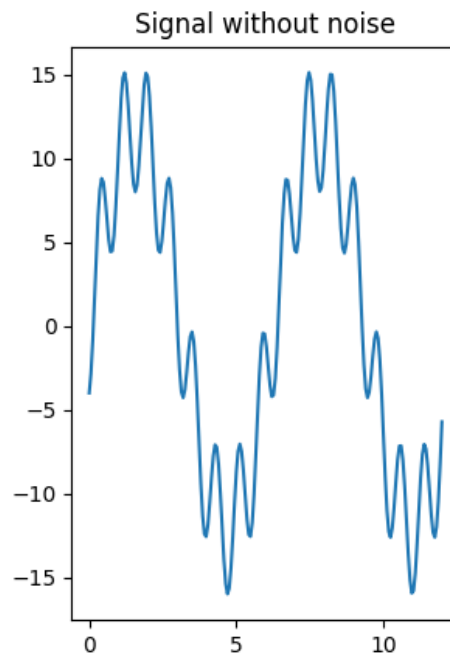


Figure 1: The signal without noise.

b - Adding Noise

The code defining the noise and the noisy signal can be found in section NOISE LINE 11-14 and plotting is in section FIGURE 2 LINE 59-64

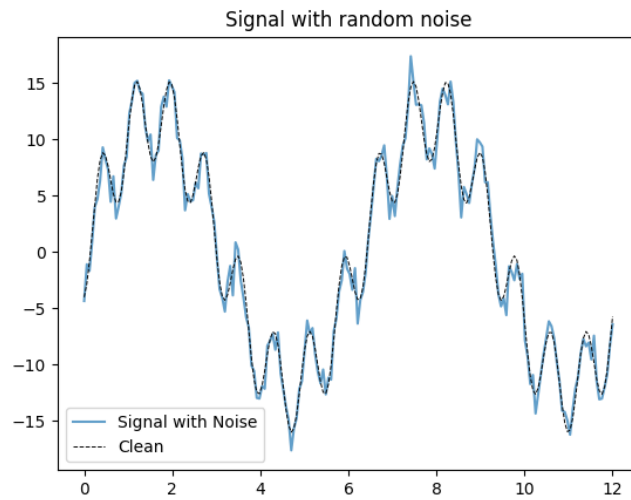


Figure 2: The signal with noise compared to the clean signal.

c - Matrix A

The code section can be found in MATRIX A LINE 19-24 and PRINT MATRIX A LINE 86-89.

```
1 Top Left Corner of Matrix A:  
2 [[0.33 0.33 0.  0.  0.  ]  
3  [0.33 0.33 0.33 0.  0.  ]  
4  [0.  0.33 0.33 0.33 0.  ]  
5  [0.  0.  0.33 0.33 0.33]  
6  [0.  0.  0.  0.33 0.33]]
```

d - Reducing Noise with Matrix A

section FIGURE 3 LINE 47-57.

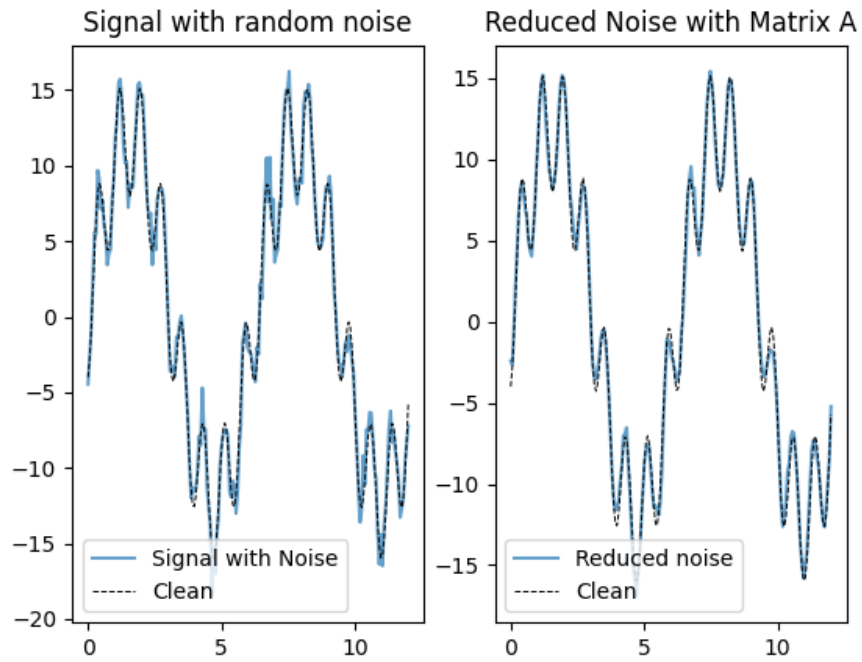


Figure 3: The signal with noise and the reduced noise with matrix A compared to clean signal.

The reason this works is that by multiplying the noisy signal with matrix A, we are in a sense taking the average of each point with its neighbors, this essentially smooths out small random fluctuations in the signal AKA 'noise', while still keeping the structure semi-intact.

e

For this section we look at the code in section MATRIX B LINE 26-32 and FIGURE 4 LINE 66-77.

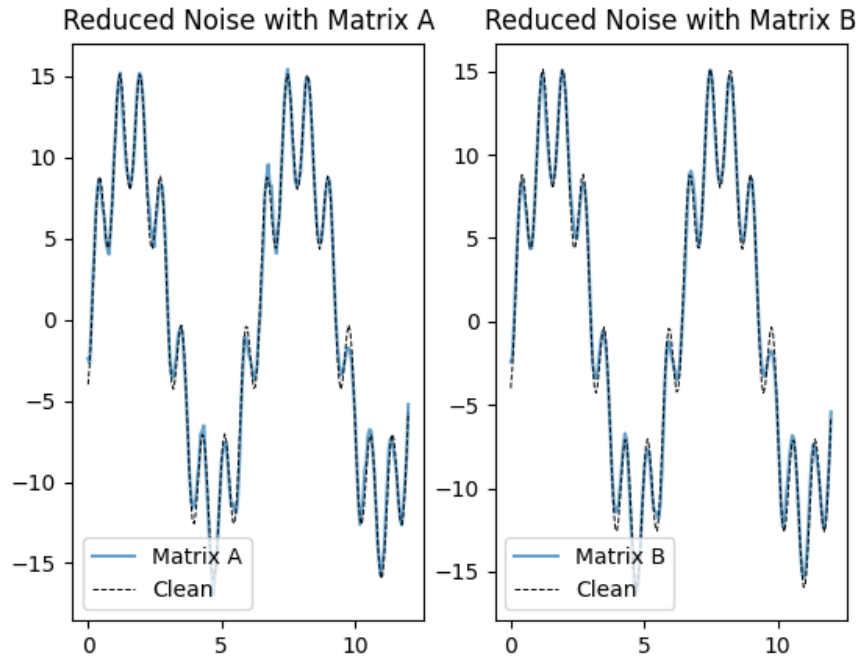


Figure 4: The reduced noise with matrix A and B, compared with the clean signal from Figure 1.

It is a bit unclear whether Matrix B is a better noise reducer than Matrix A or not, since it appears to make a better reduction in some areas but worse in others. It's clear that the reduction appears to be more aggressive, which is expected since Matrix B takes the average of more points and weighs them differently, which results in a better reduction of noise but also reduction of the actual signal, which isn't what we want.

This next figures is covered by section FIGURE 5 LINE 78-84

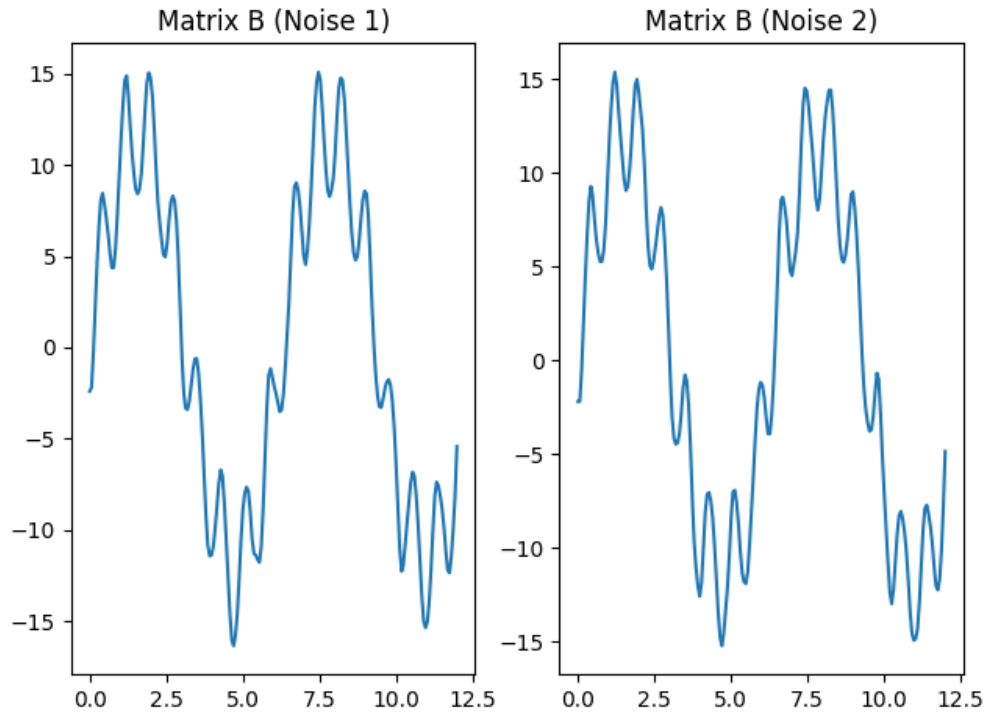


Figure 5: The reduced noise with matrix B for two different noise inputs on the Signal.

The above comparison shows that the noise reduction isn't deterministic, since noise is random, and reductions will differ based on the noise, but it does appear to reduce noise in both cases which, in my undeniable expertise, verifies that Matrix B can be used to reduce arbitrary noise.

Appendix

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 # Signal
5 n = 200
6 x = np.linspace(0.0, 12.0, n)
7 s_rad = np.sin(x) # rad for radians
8 c_rad = np.cos(8*x)
9 y = 12*s_rad - 4*c_rad
10
11 # Noise
12 rng = np.random.default_rng()
13 noise = rng.standard_normal(n)
14 y_noisy = y + noise # adding noise
15
16
17
18

```

```

19 # MATRIX A
20 a = np.diag(1/3*np.ones(n), 0)
21 b = np.diag(1/3*np.ones(n-1), 1)
22 c = np.diag(1/3*np.ones(n-1), -1)
23 A = a + b + c
24 Ay_noisy = A @ y_noisy
25
26 #MATRIX B
27 d = np.diag(6/16*np.ones(n), 0)
28 e = np.diag(4/16*np.ones(n-1), 1)
29 f = np.diag(4/16*np.ones(n-1), -1)
30 g = np.diag(1/16*np.ones(n-2), 2)
31 h = np.diag(1/16*np.ones(n-2), -2)
32 B = d + e + f + g + h
33
34 By_noisy = B @ y_noisy
35
36 # New random y_noise
37 noise2 = rng.standard_normal(n)
38 y_noisy2 = y + noise2
39 By_noisy2 = B @ y_noisy2
40
41
42 # FIGURE 1 - only signal
43 fig1, ax1 = plt.subplots()
44 ax1.plot(x, y)
45 ax1.set_title('Signal without noise')
46
47 # FIGURE 3 - comparing signal with noise to reduced noise (with A)
48 fig2, (ax2, ax3) = plt.subplots(1, 2)
49 ax2.plot(x, y_noisy, alpha= 0.7, label='Signal with Noise')
50 ax2.plot(x,y, color='black', label='Clean', linestyle='--', linewidth=0.7)
51 ax2.set_title('Signal with random noise')
52 ax2.legend(loc='lower left')
53
54 ax3.plot(x, Ay_noisy, alpha= 0.7, label='Reduced noise')
55 ax3.set_title('Reduced Noise with Matrix A')
56 ax3.plot(x,y, color='black', label='Clean', linestyle='--', linewidth=0.7)
57 ax3.legend(loc='lower left')
58
59 # Figure 2 - Only Noise
60 fig21, ax2 = plt.subplots()
61 ax2.plot(x, y_noisy, alpha= 0.7, label='Signal with Noise')
62 ax2.plot(x,y, color='black', label='Clean', linestyle='--', linewidth=0.7)
63 ax2.set_title('Signal with random noise')
64 ax2.legend(loc='lower left')
65
66 # FIGURE 4 -
67 fig3, (ax5, ax4) = plt.subplots(1, 2)
68 ax5.plot(x, Ay_noisy, alpha=0.7, label='Matrix A')
69 ax5.set_title('Reduced Noise with Matrix A')
70 ax5.plot(x,y, color='black', label='Clean', linestyle='--', linewidth=0.7)
71 ax5.legend(loc='lower left')
72

```

```

73 ax4.plot(x, By_noisy, alpha=0.7, label='Matrix B')
74 ax4.set_title('Reduced Noise with Matrix B')
75 ax4.plot(x,y, color='black', label='Clean', linestyle='--', linewidth=0.7)
76 ax4.legend(loc='lower left')
77
78 # FIGURE 5
79 fig4, (ax6, ax7) = plt.subplots(1, 2)
80 ax6.plot(x, By_noisy)
81 ax6.set_title('Matrix B (Noise 1)')
82
83 ax7.plot(x, By_noisy2)
84 ax7.set_title('Matrix B (Noise 2)')
85
86 #Print Matrix A
87 np.set_printoptions(precision=2)
88 print('Top Left Corner of Matrix A:')
89 print(A[:5, :5])
90
91 plt.tight_layout()
92 plt.show()

```