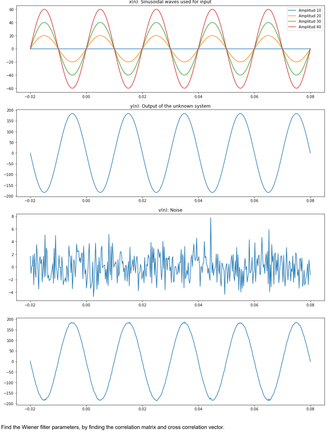
In [ ]:
```
# p.s the code used below is not fully mine.
```

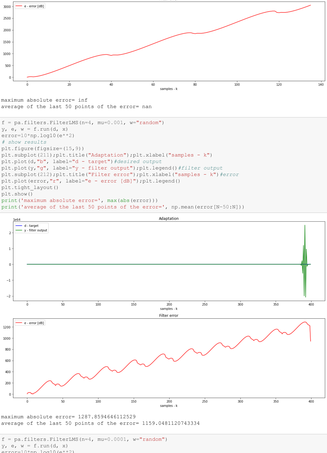
In [1]:
```
import numpy as np
from IPython.display import Image
import matplotlib.pylab as plt
import pandasip as pa
```

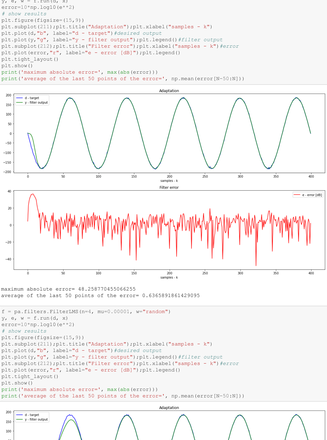
In [8]:
```
# creation data
N = 600 #number of data points

time = np.linspace(-0.02, 0.08, N)

# input data as array of N data point

xin = np.zeros((4,N))

for i in range(4):
    xin[i] = (i+1)*10 * np.sin(2*np.pi*50*time)
    x = xin.transpose()

    v = np.random.normal(0, 2, N) # noise

y = 3*x[:,0] + 0.2*x[:,1] - 5.3*x[:,2] + 0.4*x[:,3]#output of the unknown,system h(n)

n = 3*x[:,0] + 0.2*x[:,1] - 5.3*x[:,2] + 0.4*x[:,3] + v # target
```

In [9]:
```
plt.figure(figsize=(15,20))

plt.subplot(411)

plt.plot(time,x)

plt.legend(['Amplitud 10', 'Amplitud 20', 'Amplitud 30', 'Amplitud 40'],loc='upper right')

plt.title('x(n): Sinusoidal waves used for input')

plt.subplot(412)

plt.plot(time,y)

plt.title('y(n): Output of the unknown system')

plt.subplot(413)

plt.plot(time,v)

plt.title('v(n): Noise')

plt.subplot(414)

plt.plot(time,d)
```

Out[9]:
```
[<matplotlib.lines.Line2D at 0x190ccb485e0>]
```





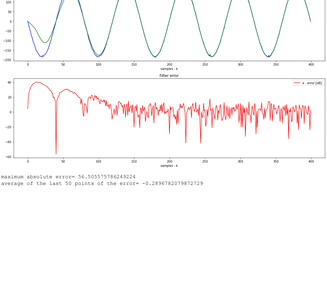


Find the Wiener filter parameters, by finding the correlation matrix and cross correlation vector.

y and d should as possible, and stabilize the error plot as close to zero as possible. For this reason the program will also print the maximum absolute error and the average of the last 50 points of the error. The best is the other with smaller error and more similarity of 'y' and 'd'.

In [10]:
```
# This class (FilterLMS) represents an adaptive LMS filter.
f = pa.filters.FilterLMS(n=4, mu=0.01, w="random")

# 'n' : length of filter (integer) - how many input is input array
# 'mu' : learning rate (float). Also known as step size. If it is too slow,
# the filter may have bad performance. If it is too high,
# the filter will be unstable. The default value can be unstable
# 'w' : initial weights of filter. "random" : creates random weights
# 'd' : desired value (1 dimensional array of length N)
# 'x' : input matrix (2 dimensional array of shape (N,4)). Rows are samples,columns are input arrays
# 'y' : output value (1 dimensional array)
# The size corresponds with the desired value.
# 'e' : filter error for every sample (1 dimensional array).
# The size corresponds with the desired value.
# 'w' : history of all weights (2 dimensional array).
# Every row is set of the weights for given sample.


y, e, w = f.run(d, x)

error=10*np.log10(e**2)

# show results
plt.figure(figsize=(15,9))

plt.subplot(211);plt.title("Adaptation");plt.xlabel("samples - k")

plt.plot(d,"b", label="d - target");plt.plot(y,"g", label="y - filter output");plt.legend()#filter output

plt.subplot(212);plt.title("Filter error");plt.xlabel("samples - k");#error
plt.plot(error,"r", label="e - error [dB]");plt.legend()
plt.tight_layout()
plt.show()
print('maximum absolute error=', max(abs(error)))
print('average of the last 50 points of the error=', np.mean(error[N-50:N]))
```

```
C:\Users\31647\Anaconda3\lib\site-packages\pandasip\filters\lms.py:252: RuntimeWarning: invalid value encountered in multiply
  dw = self.mu * e[k] * x[k]
<ipython-input-10-2eb742b4cf34>:21: RuntimeWarning: overflow encountered in square
  error=10*np.log10(e**2)
```




```
maximum absolute error= inf
average of the last 50 points of the error= nan
```

In [11]:
```
f = pa.filters.FilterLMS(n=4, mu=0.001, w="random")
y, e, w = f.run(d, x)
error=10*np.log10(e**2)
# show results
plt.figure(figsize=(15,9))
plt.subplot(211);plt.title("Adaptation");plt.xlabel("samples - k")
plt.plot(d,"b", label="d - target");#desired output
plt.plot(y,"g", label="y - filter output");plt.legend()#filter output
plt.subplot(212);plt.title("Filter error");plt.xlabel("samples - k");#error
plt.plot(error,"r", label="e - error [dB]");plt.legend()
plt.tight_layout()
plt.show()
print('maximum absolute error=', max(abs(error)))
print('average of the last 50 points of the error=', np.mean(error[N-50:N]))
```



```
maximum absolute error= 1287.8594664112329
average of the last 50 points of the error= 1159.0481120743334
```

In [12]:
```
f = pa.filters.FilterLMS(n=4, mu=0.0001, w="random")
y, e, w = f.run(d, x)
error=10*np.log10(e**2)
# show results
plt.figure(figsize=(15,9))
plt.subplot(211);plt.title("Adaptation");plt.xlabel("samples - k")
plt.plot(d,"b", label="d - target");#desired output
plt.plot(y,"g", label="y - filter output");plt.legend()#filter output
plt.subplot(212);plt.title("Filter error");plt.xlabel("samples - k");#error
plt.plot(error,"r", label="e - error [dB]");plt.legend()
plt.tight_layout()
plt.show()
print('maximum absolute error=', max(abs(error)))
print('average of the last 50 points of the error=', np.mean(error[N-50:N]))
```

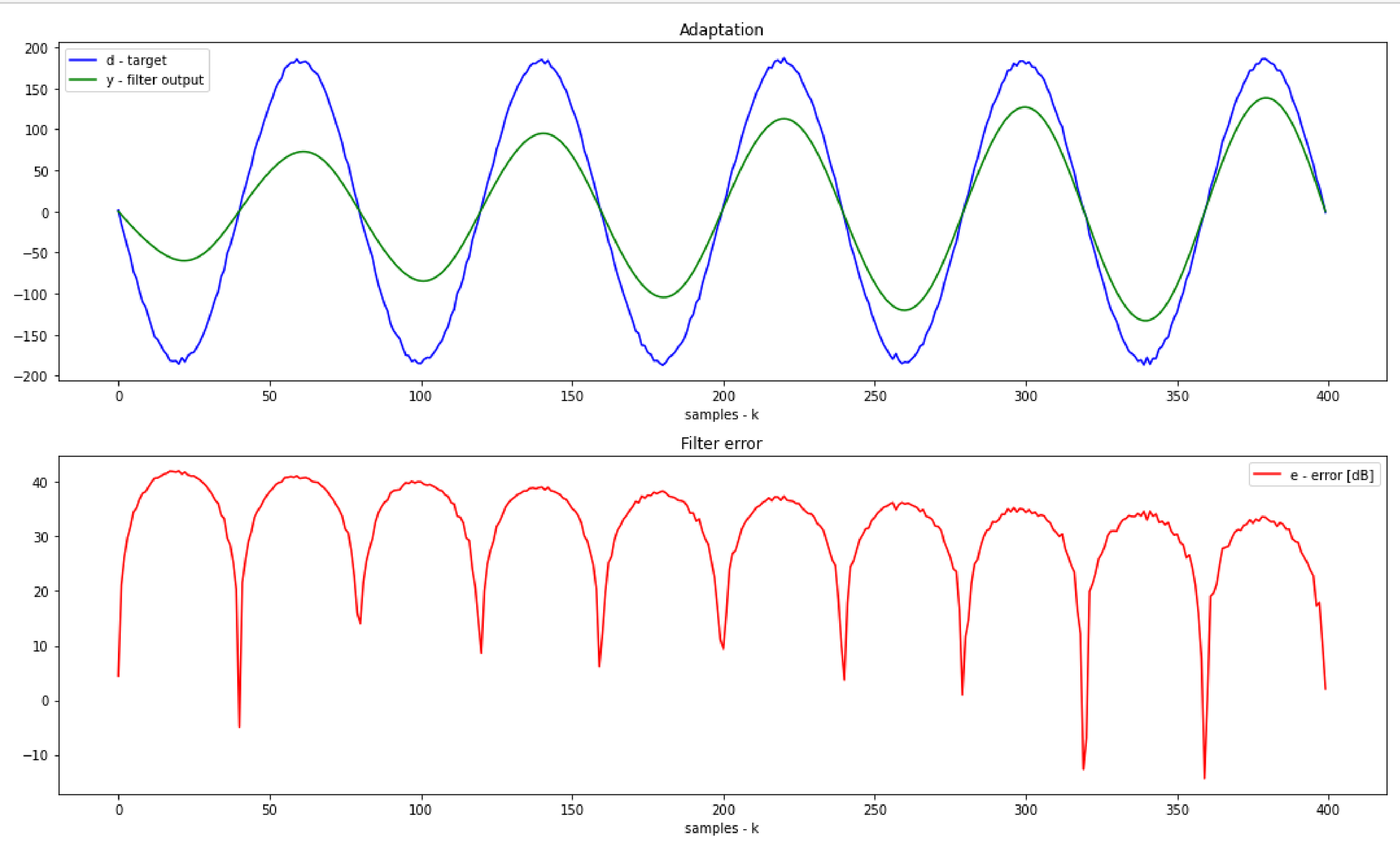

```
maximum absolute error= 48.258770455066255
average of the last 50 points of the error= 0.6365891861429095
```

In [13]:
```
f = pa.filters.FilterLMS(n=4, mu=0.00001, w="random")
y, e, w = f.run(d, x)
error=10*np.log10(e**2)
# show results
plt.figure(figsize=(15,9))
plt.subplot(211);plt.title("Adaptation");plt.xlabel("samples - k")
plt.plot(d,"b", label="d - target");#desired output
plt.plot(y,"g", label="y - filter output");plt.legend()#filter output
plt.subplot(212);plt.title("Filter error");plt.xlabel("samples - k");#error
plt.plot(error,"r", label="e - error [dB]");plt.legend()
plt.tight_layout()
plt.show()
print('maximum absolute error=', max(abs(error)))
print('average of the last 50 points of the error=', np.mean(error[N-50:N]))
```



```
maximum absolute error= 56.5055757884249224
average of the last 50 points of the error= -0.28976920759272729
```

```
In [14]: f = pa.filters.FilterLMS(n=4, mu=0.000001, w="random")
y, e, w = f.run(d, x)
error=10*np.log10(e**2)
# show results
plt.figure(figsize=(15,9))
plt.subplot(211);plt.title("Adaptation");plt.xlabel("samples - k")
plt.plot(d,"b", label="d - target")#desired output
plt.plot(y,"g", label="y - filter output");plt.legend()#filter output
plt.subplot(212);plt.title("Filter error");plt.xlabel("samples - k")#error
plt.plot(error,"r", label="e - error [dB]");plt.legend()
plt.tight_layout()
plt.show()
print('maximum absolute error=', max(abs(error)))
print('average of the last 50 points of the error=', np.mean(error[N-50:N]))
```




```
maximum absolute error= 42.02560953840995
average of the last 50 points of the error= 25.3628209129305
```

References

[1]Monson H. Hayes: Statistical Digital Signal Processing and Modeling, Wiley, 1996, ISBN 0-471-59431-8 [2]S. Haykin: Adaptative Filter Theory, Hamilton, 2014, ISBN 978-0-132-67145-3 [3]https://matousc89.github.io/padasip/_modules/padasip/filters/lms.html#FilterLMS.run