Smoothing using convolution

In[63]:=
```
SetDirectory[NotebookDirectory[]];
```

Define a simple signal:

In[64]:=
```
f[x_] := Cos[2 π 26 x / 1024] + Cos[2 π 34 x / 1024];
```

We will be smoothing noisy signals with two filters: a Gaussian filter FG of length
and with a moving average filter of length lhfiltA=21:

In[65]:=
```
lhfiltG = 35;
```

In[66]:=
```
lhfiltA = 21;
```

Our signals will have length of lh=900 samples; thus, the nearest power of 2 larger
plus the length of the signal lhfiltG+lh-1=35+900-1<1024.

In[67]:=
```
lh = 900;
```

In[68]:=
```
LH = 1024;
```

In[69]:=
```
diff = LH - lh
```

Out[69]=
```
124
```

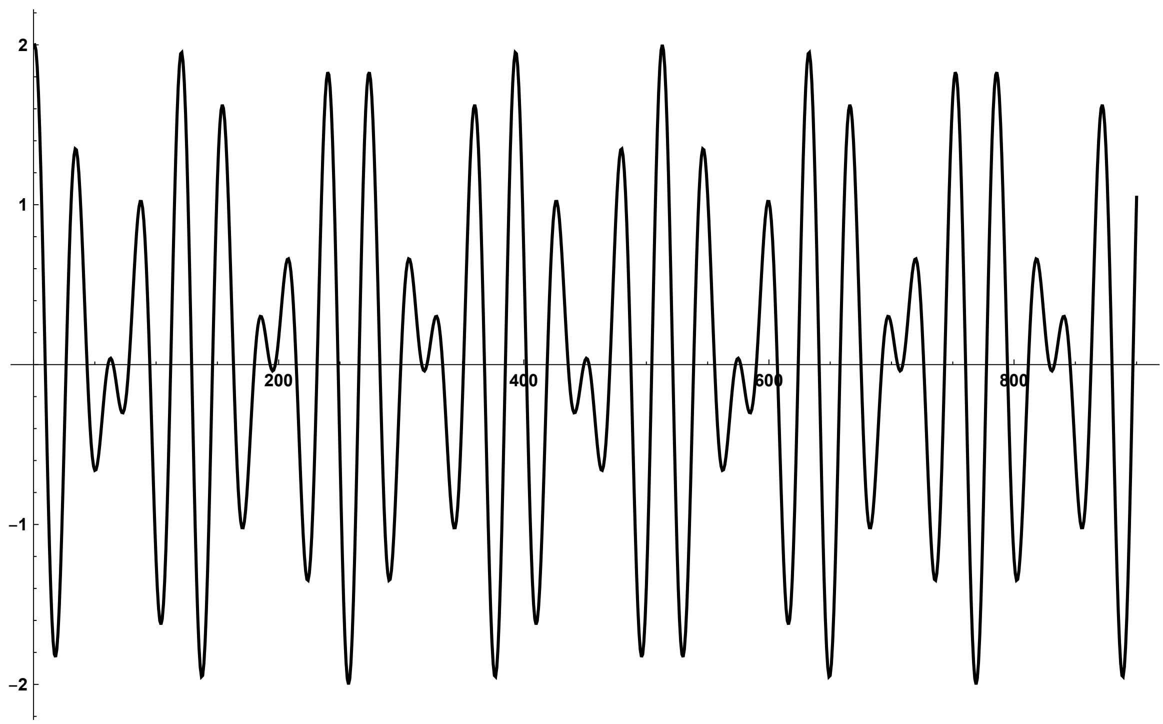
Let us produce a list (in Mathematica a Table) of 900 samples of our signal at all i

In[70]:=
```
Sig = Table[f[x], {x, 0, lh - 1}];
```

Let us plot such a signal, joining the sampling points for easy vieweing:

In[71]:=
```
P1 = ListPlot[Sig, ImageSize → 600, PlotStyle → {Black}, Joined → True]
```

Out[71]=



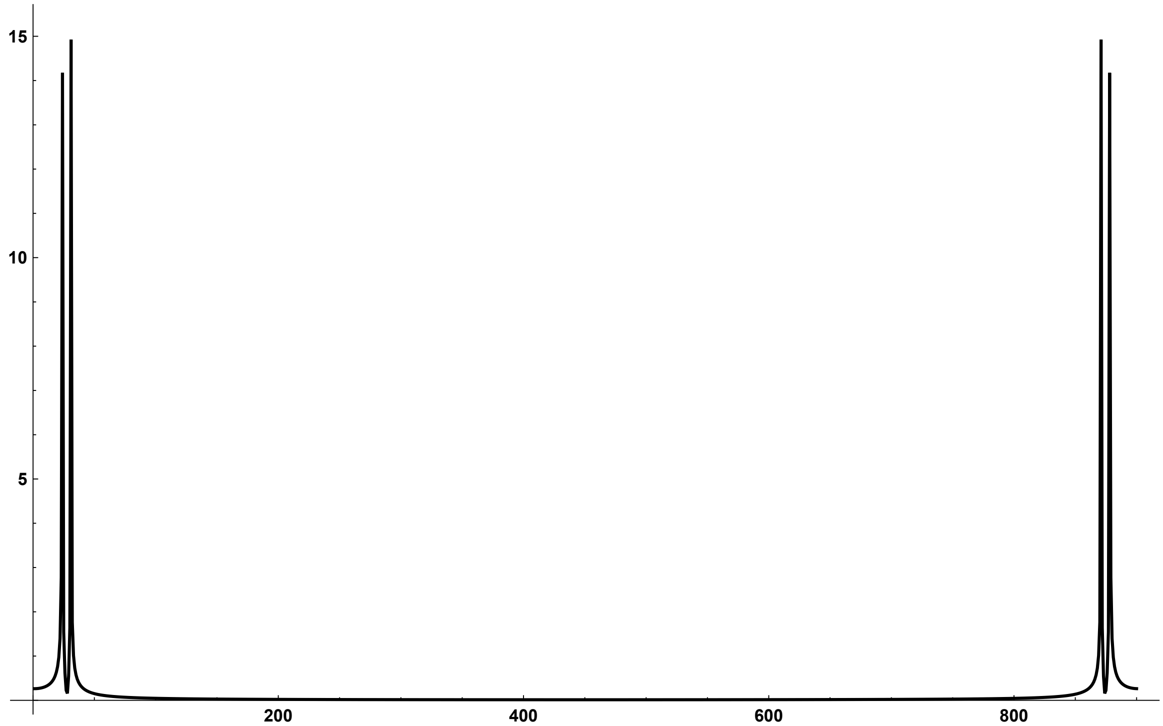
Let us take the FFT of such a signal; in Mathematica FFT is called Fourier:

In[72]:=
```
FT = Fourier[Sig];
```

FFT produces the DFT[Sig] which is a sequence of complex values, so let us plot the the DFT[Sig]:

In[73]:=
```
ListPlot[Abs[FT], ImageSize → 600,
 PlotRange → All, Joined → True, PlotStyle → {Black}]
```

Out[73]=

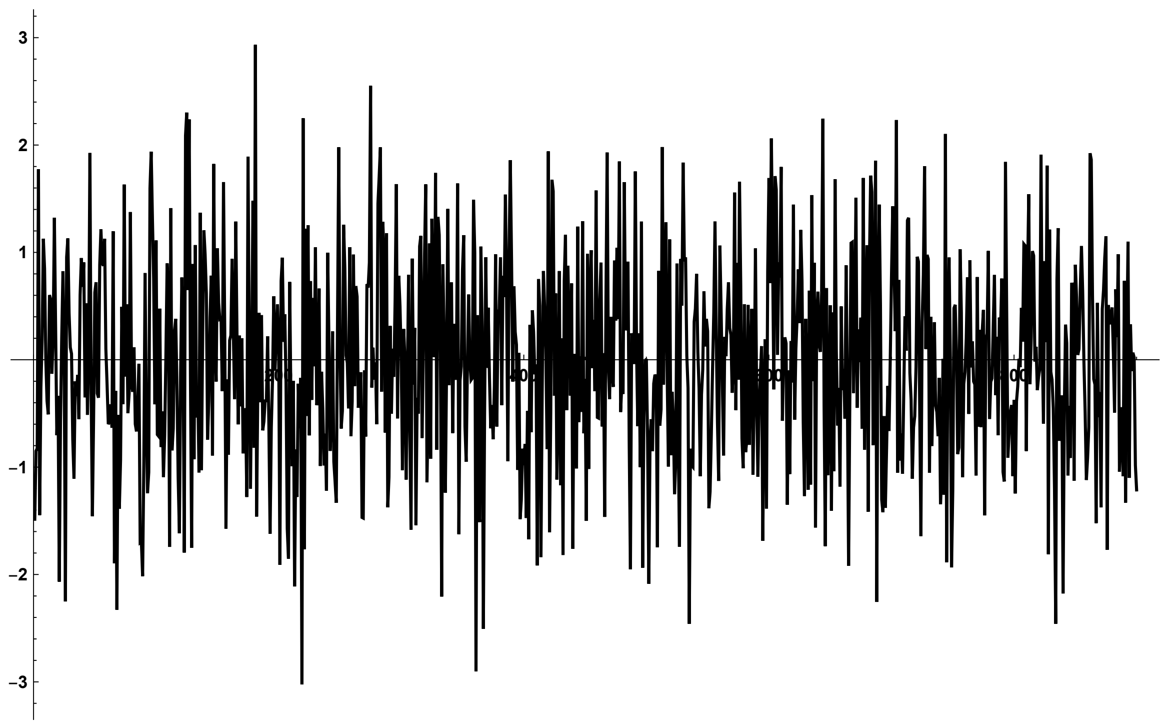


As explained in the slides, two cosines produce 4 peaks. We now generate some Gaussi
variance 1, of length also lh and plot it.

In[74]:=
```
noise = Table[RandomVariate[NormalDistribution[0, 1]], {x, 1, lh}];
```

In[75]:= `P2 = ListPlot[noise, ImageSize → 600, PlotStyle → {Black}, Joined → True]`

Out[75]=

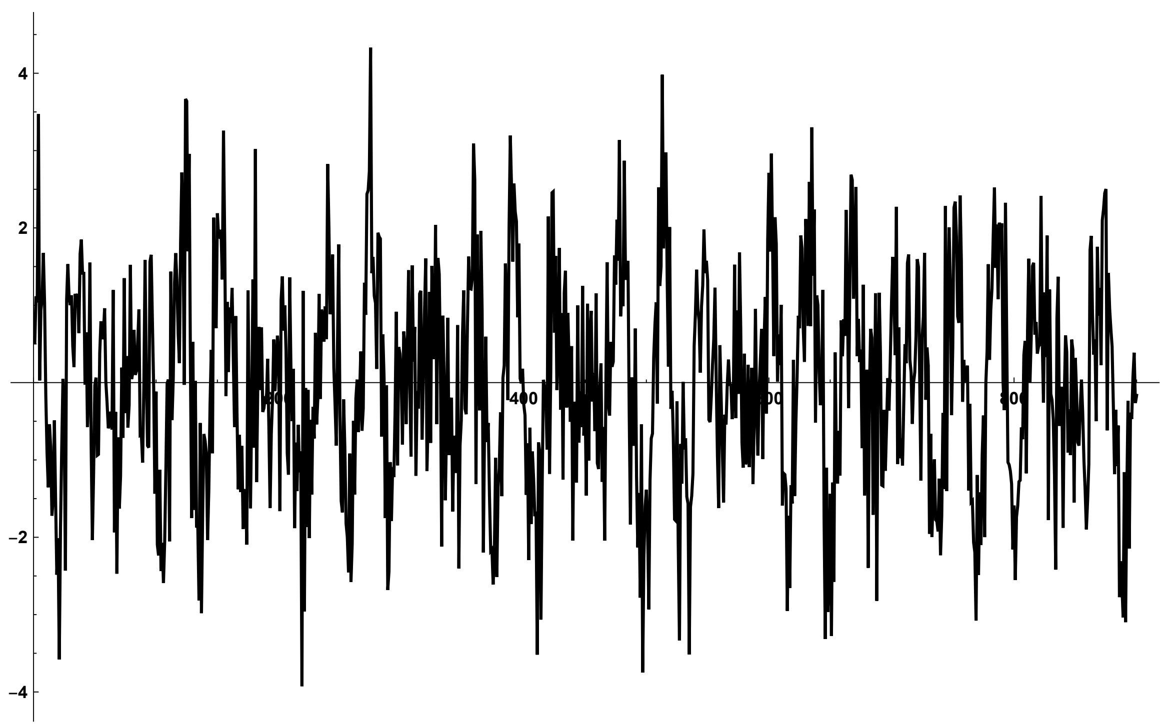


We add signal Sig and the noise to produce a very noisy signal 'noisySignal' and the

In[76]:= `noisySig = Sig + noise;`

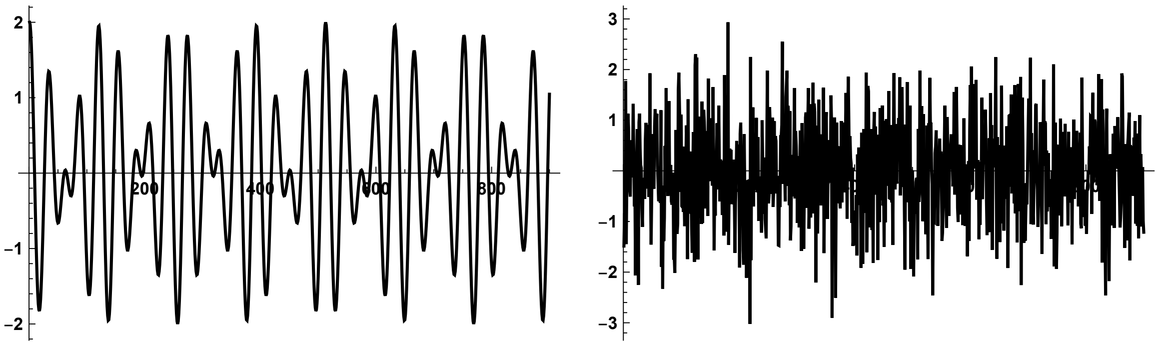
In[77]:= `P3 = ListPlot[noisySig, ImageSize → 600, PlotStyle → {Black}, Joined → True]`

Out[77]=



In[78]:=
```
GG = GraphicsGrid[{{P1, P2}}, ImageSize → 600]
```

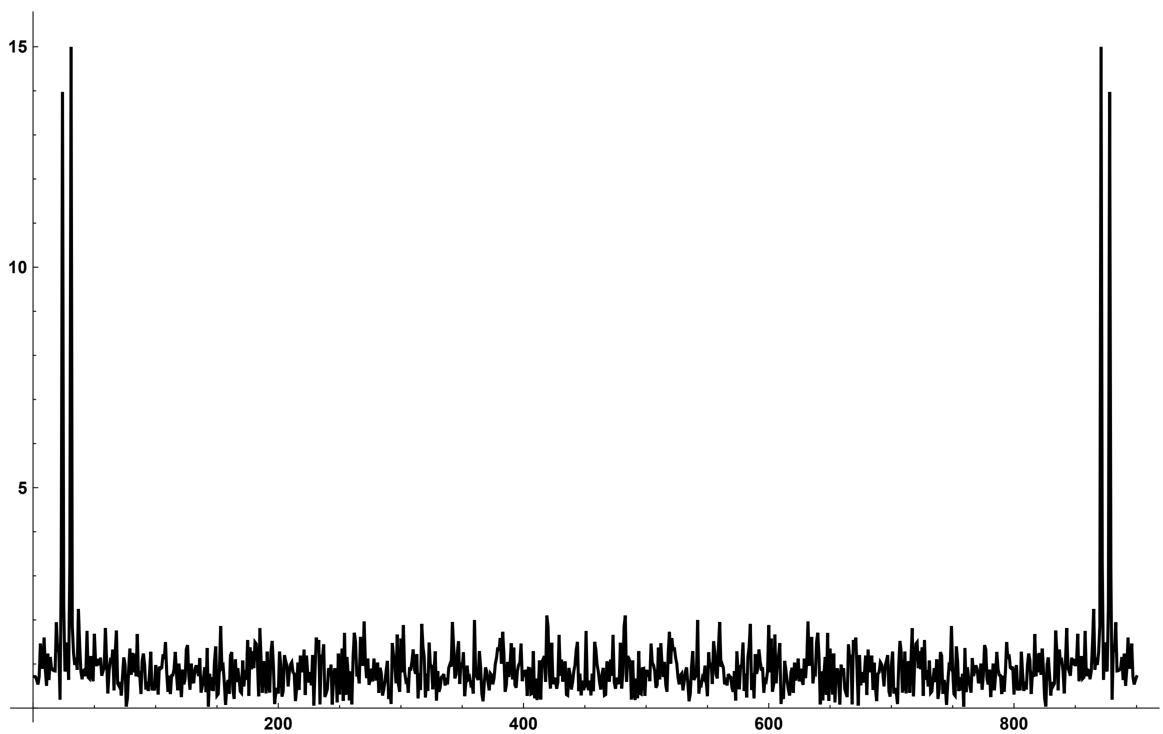
Out[78]=



We also plot the absolute value of the DFT of the 'noisySignal'. Even though noisySi
a sum of thwo cosines, the DFT still clearly shows just 4 peaks (as symmetric pairs)

In[79]:=
```
ListPlot[Abs[Fourier[noisySig]], ImageSize → 600,
 PlotStyle → {Black}, Joined → True, PlotRange → All]
```

Out[79]=



We make a smoothing Gaussian filter using a zero mean Gaussian with variance
((lhfiltG-1)/6)^2 so that 3v^1/2 = (lhfiltG-1)/2:

In[80]:= $v = \left(\frac{\text{lhfiltG} - 1}{6}\right)^2$

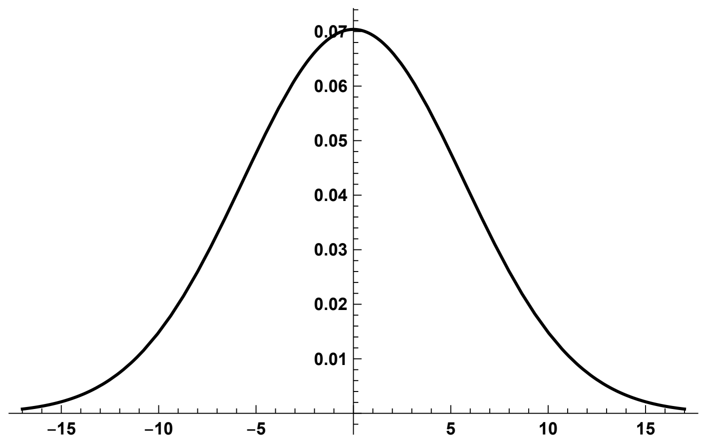
Out[80]= $\frac{289}{9}$

In[81]:=

In[82]:= $\texttt{hlhfilt} = \frac{\texttt{lhfiltG} - 1}{2}$;

In[83]:= $\texttt{Plot}\left[\frac{1}{\sqrt{2\,\pi\,v}}\,e^{-\frac{t^2}{2\,v}},\ \{t, -\texttt{hlhfilt}, \texttt{hlhfilt}\},\ \texttt{PlotStyle} \to \texttt{Black}\right]$

Out[83]=



In[84]:=

We evaluate such a Gaussian at all integers in the interval [-hlhfilt, hlhfilt]:

In[85]:= $\texttt{FG0} = \texttt{N}\left[\texttt{Table}\left[\frac{1}{\sqrt{2\,\pi\,v}}\,e^{-\frac{t^2}{2\,v}},\ \{t, -\texttt{hlhfilt}, \texttt{hlhfilt}\}\right]\right]$;

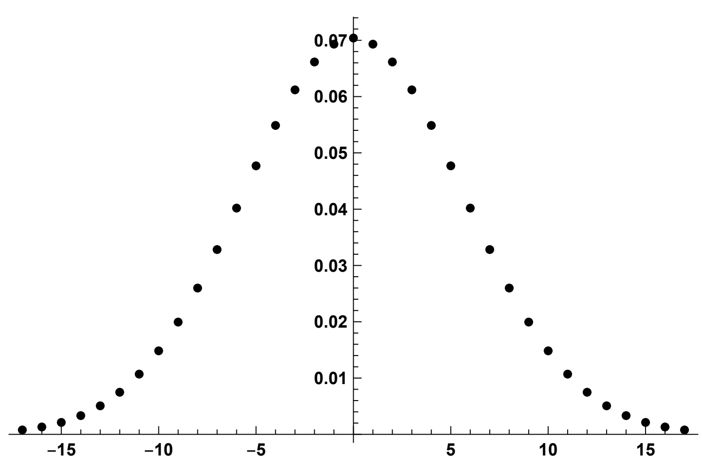
Note that weights obtained in this way sumup almost to 1, so we do not need to norma

In[86]:= Total[FG0]

Out[86]= 0.998014

In[87]:= $\texttt{ListPlot}\left[\texttt{Table}\left[\left\{t, \frac{1}{\sqrt{2\,\pi\,v}}\,e^{-\frac{t^2}{2\,v}}\right\},\ \{t, -\texttt{hlhfilt}, \texttt{hlhfilt}\}\right],\right.$
$\left.\texttt{PlotRange} \to \texttt{All},\ \texttt{PlotStyle} \to \texttt{Black}\right]$

Out[87]=



In[88]:=
```
FG = FG0/Total[FG0]; (* normalization,
so that all the weights sum up to 1 exactly (not really needed) *)
```

In[89]:=
```
Total[FG]
```

Out[89]= 1.

To be able to compute the convolution of the noisySig and our Gaussian smoother FG w
as lh+lhfiltG-1 and then find the nearest power of 2 which is LH in our case. So we

In[90]:=
```
lh + lhfiltG - 1
```

Out[90]= 934

In[91]:=
```
padd = LH - (lh + lhfiltG - 1)
```

Out[91]= 90

So we padd noisySig with LH-lh many zeros:

In[92]:=
```
noisySigE = Join[noisySig, ConstantArray[0, LH - lh]];
```

In[93]:=
```
Length[noisySigE]
```

Out[93]= 1024

We now find the DFT of the noisySig:

In[94]:=
```
FNSE = Fourier[noisySigE];
```

Similarly, we padd the weights vector with LH-lhfiltG many zeros and compute its DCT

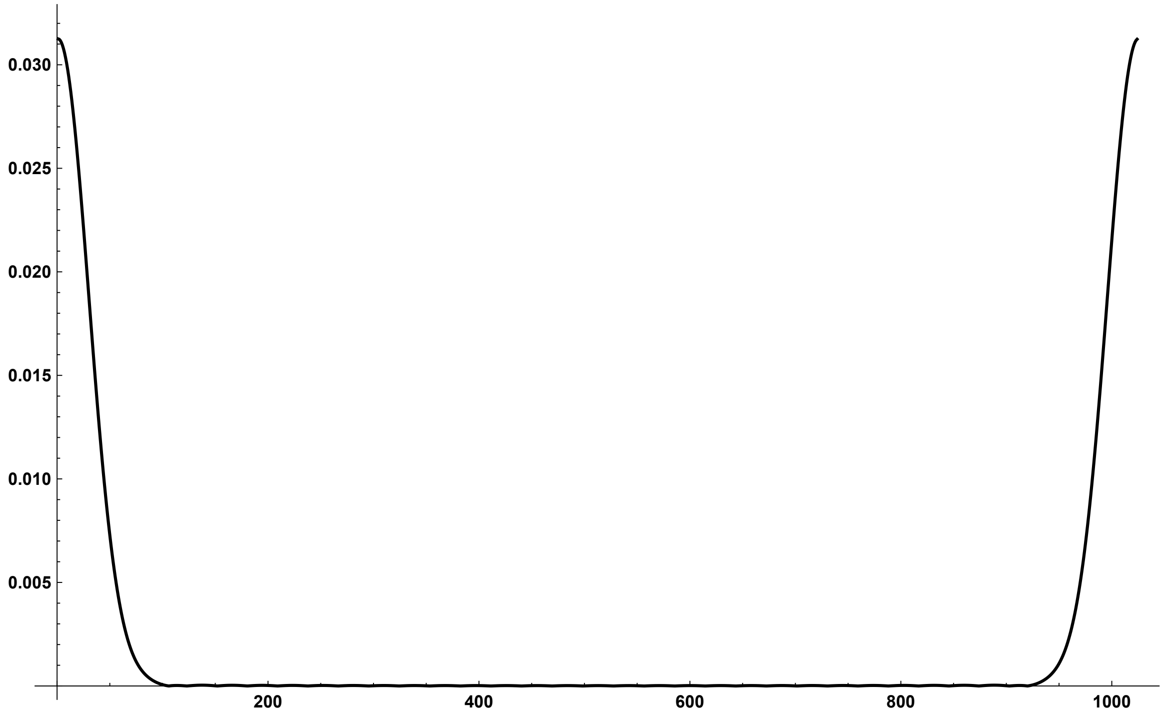
In[95]:=
```
FGE = Join[FG, ConstantArray[0, LH - lhfiltG]];
```

In[96]:=
```
FFG = Fourier[FGE];
```

Let us look what the DCT of our Gaussian filter looks like:

In[97]:=
```
ListPlot[Abs[FFG], ImageSize → 600,
 PlotStyle → {Black}, Joined → True, PlotRange → All]
```

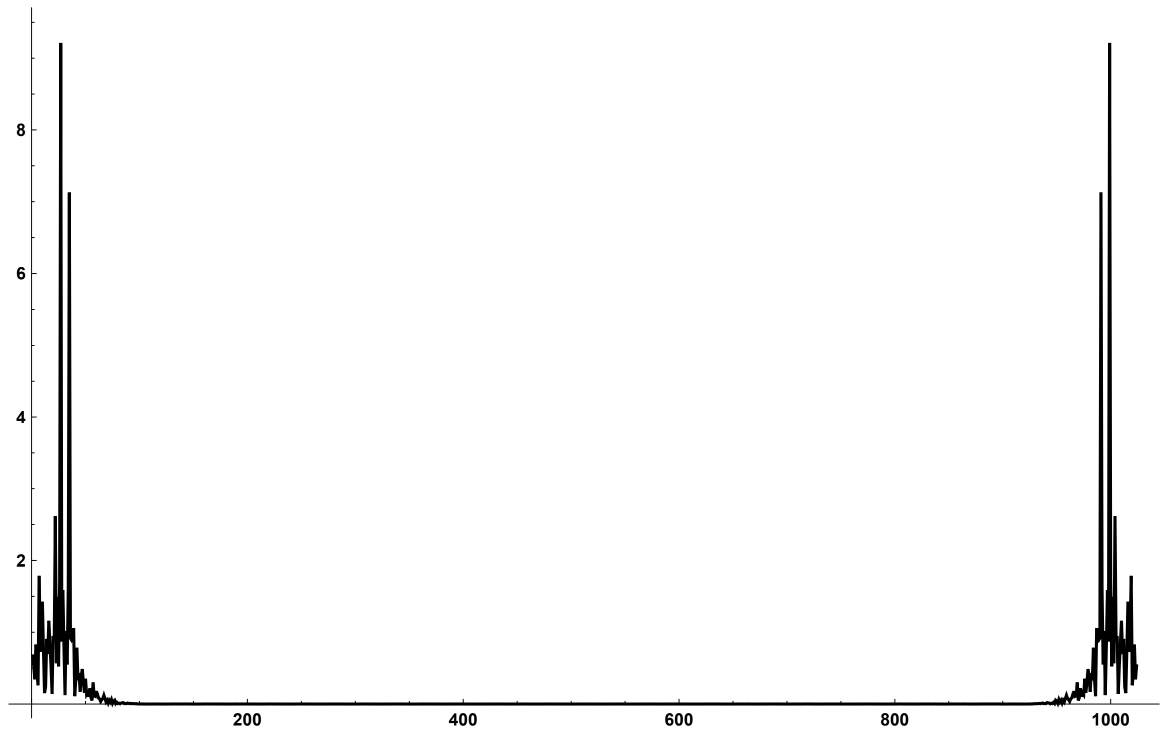
Out[97]=



Clearly, it will remove high frequency noise after we multiply the DCT of the noisy our Gaussian filter. We no find the product of the DCT of the noisy signal and the I Mathematica computes the DCT by function 'Fourier' which includes a normalisation wi the direct and inverse DCT, so, as explained in the slides we have to multiply the p by Sqrt[Length[FFG]] to get a proper scaling:

In[98]:=
```
ListPlot[Abs[FNSE * FFG * √Length[FFG]], ImageSize → 600,
 PlotStyle → {Black}, Joined → True, PlotRange → All]
```

Out[98]=



As it can be seen on the plot, most of the high frequency noise is gone. Further bel
of the smoothed signal (blue) with the DCT of the noisy signal (red)

As it can be seen on the plot, most of the high frequency noise is gone. We now find the Inverse DCT of the product which will produce the convolution of the noisySig and the smoothing weights FG:

In[99]:=
```
smoothedSigG1 = Chop[InverseFourier[FNSE * FFG] √Length[FFG]];
```

'Chop' function removes small imaginary components which are due to roundoff error.
We now remove the part of the convolution which is due to padding by dropping 'padd'

In[100]:=
```
smoothedSigG2 = smoothedSigG1[[1 ;; -padd - 1]];
```

To obtain a cleaned signal of the same size as the original signal for an easy compa
many points from both end:

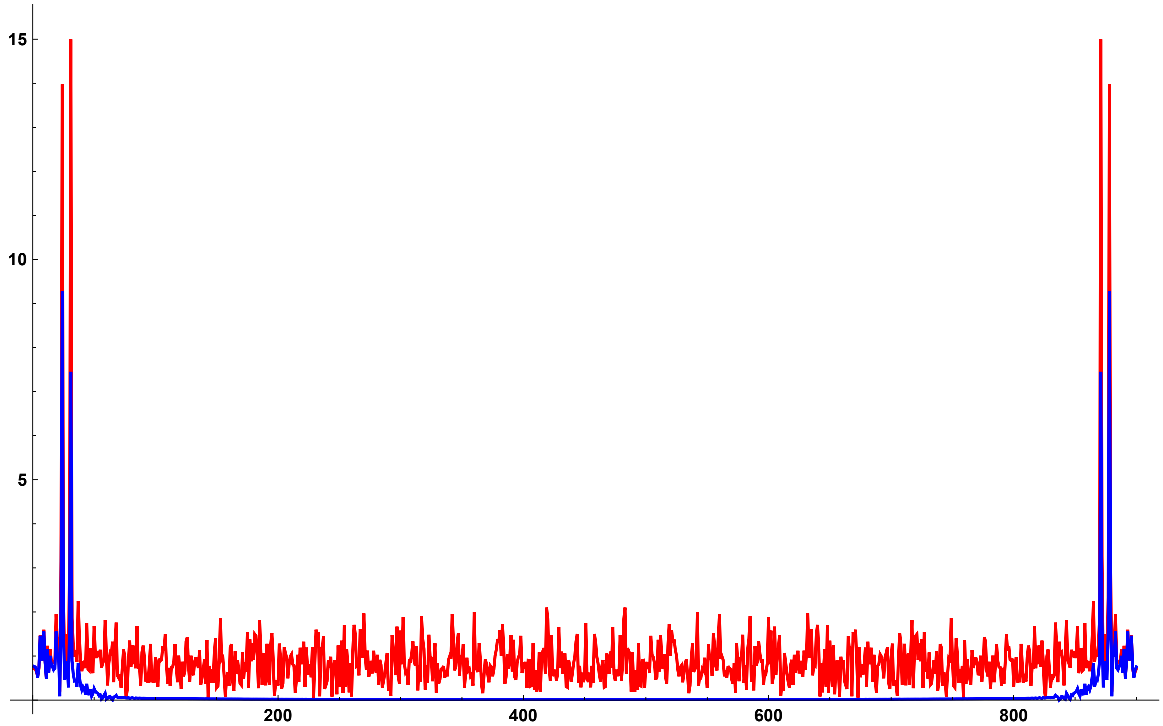
In[101]:=
```
offset = (lhfiltG - 1) / 2
```

Out[101]= 17

In[102]:=
```
smoothedSigG = smoothedSigG2[[offset + 1 ;; -offset - 1]];
```

In[103]:=
```
ListPlot[{Abs[Fourier[noisySig]], Abs[Fourier[smoothedSigG]]},
  ImageSize → 600, PlotRange → All, Joined → True, PlotStyle → {Red, Blue}]
```

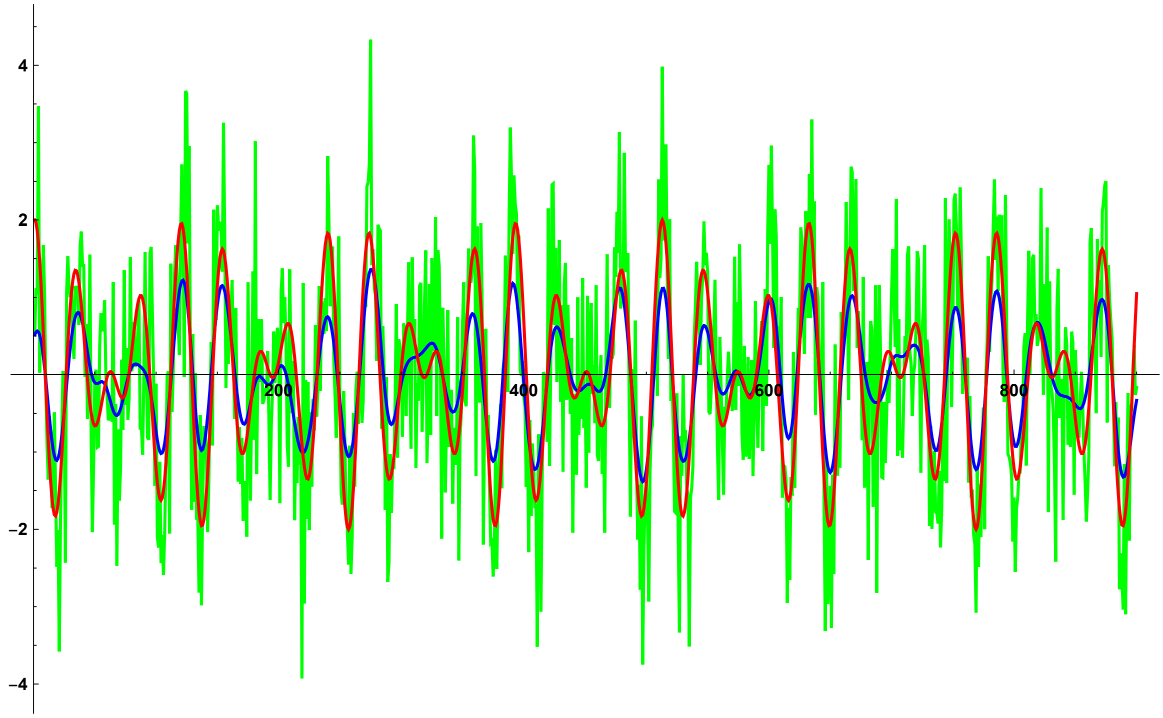
Out[103]=



We now compare the smoothed noisy signal with the clean initial signal and with the ploting the noisy signal noisySig in green, the original "pure" signal in red and tl obtained by our gaussian smoothing in blue:

In[104]:=
```
ListPlot[{ noisySig, smoothedSigG, Sig},
 ImageSize → 600, Joined → True, PlotStyle → {Green, Blue, Red}]
```

Out[104]=

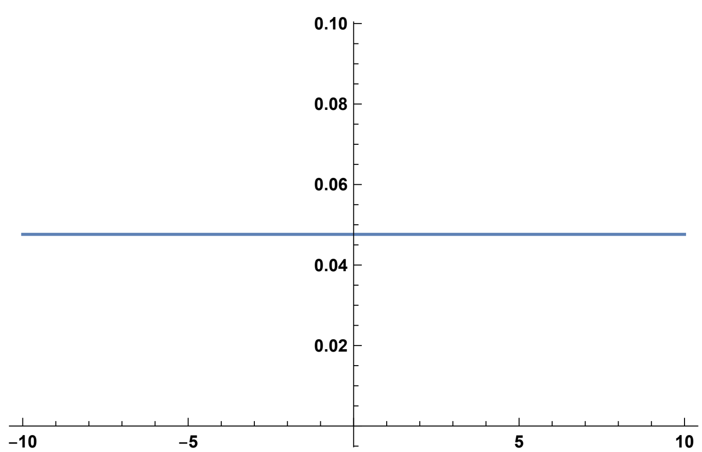


We repeat the same procedure now with a Moving Average smoothing weights. So let us using a Moving Average:

In[105]:=
```
hlhfiltA = (lhfiltA - 1) / 2;
```

In[106]:=
```
Plot[1/lhfiltA, {t, -hlhfiltA, hlhfiltA}]
```

Out[106]=



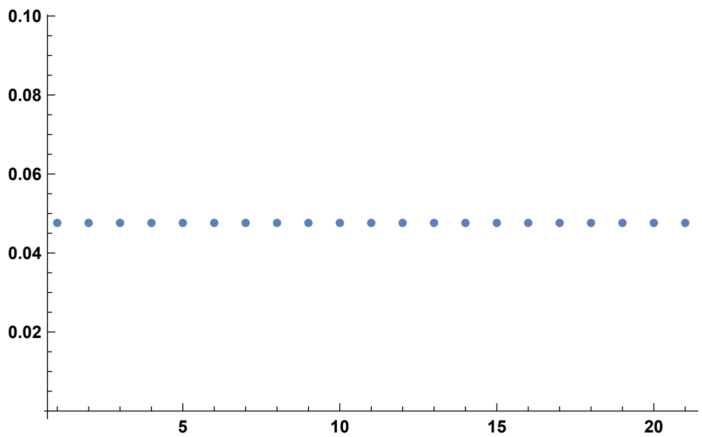
In[107]:=

In[108]:=
```
FA = N[Table[1/lhfiltA, {t, -hlhfiltA, hlhfiltA}]];
```

All the weights are the same:

In[109]:=
```
ListPlot[FA, PlotRange → All]
```

Out[109]=



In[110]:=
```
paddA = LH - (lh + lhfiltA - 1)
```

Out[110]= 104

In[111]:=
```
lh + lhfiltA - 1
```

Out[111]= 920

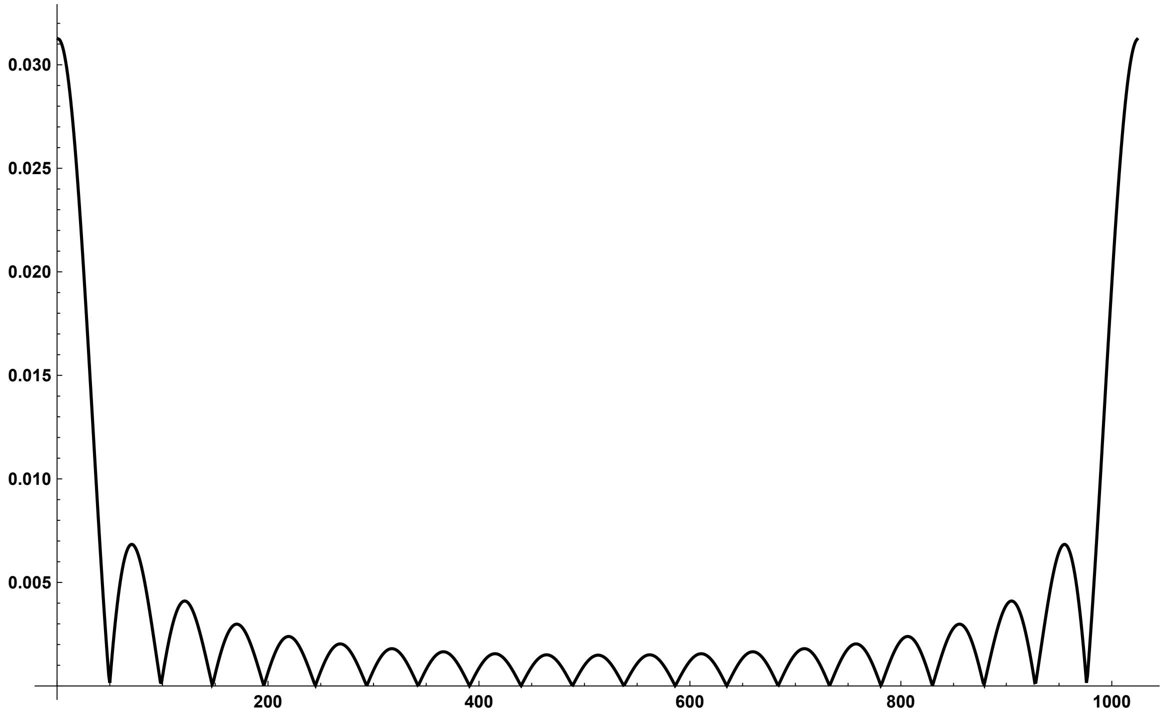
In[112]:=
```
FAE = Join[FA, ConstantArray[0, LH - lhfiltA]];
```

Let us see what the DCT of such weights looks like:

In[113]:=
```
FFA = Fourier[FAE];
```

In[114]:=
```
ListPlot[Abs[FFA], ImageSize → 600,
 PlotStyle → {Black}, Joined → True, PlotRange → All]
```

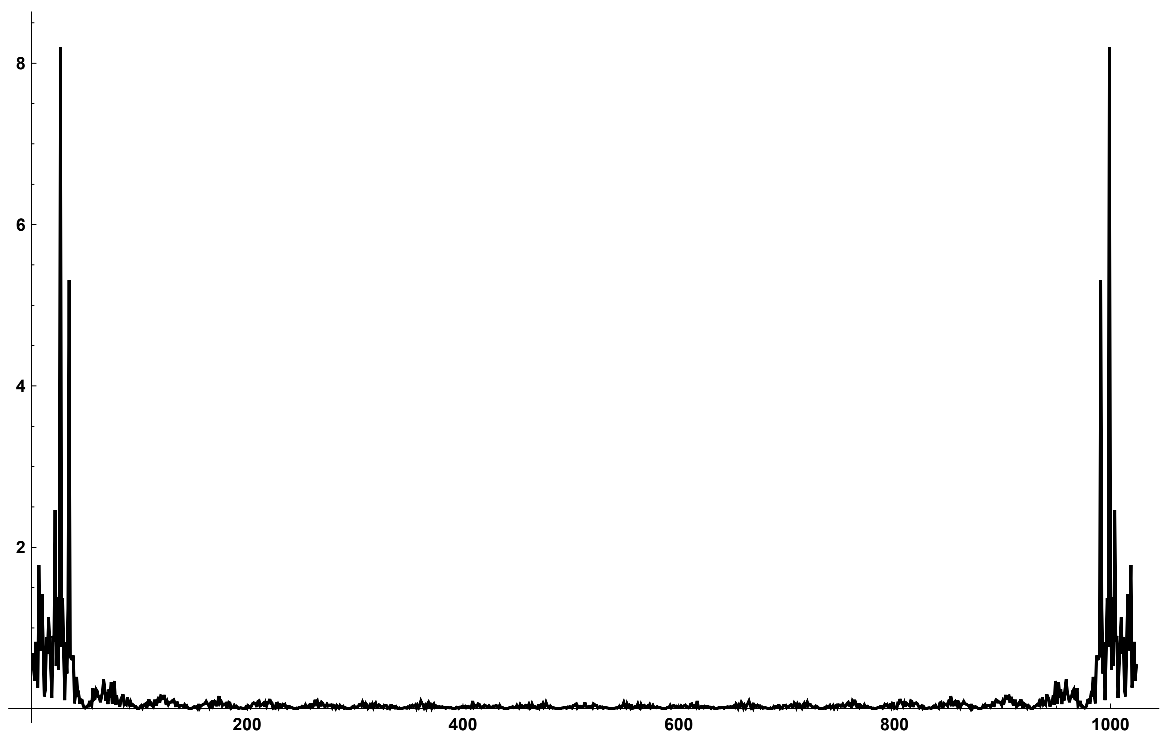
Out[114]=



Obviously, such a smoother will suppress high frequency noise, but not nearly as cle

In[115]:=
```
ListPlot[Abs[FNSE * FFA * √Length[FFA] ], ImageSize → 600,
 PlotStyle → {Black}, Joined → True, PlotRange → All]
```

Out[115]=



We can see that there is some high frequency noise left. We now find the Inverse DCT
of the Moving Average smoothed noisy signal:

In[116]:= smoothedSigA1 = Chop[InverseFourier[FNSE * FFA] $\sqrt{\text{Length[FFA]}}$];

In[117]:= Length[smoothedSigA1]

Out[117]= 1024

In[118]:= smoothedSigA2 = smoothedSigA1[[1 ;; -paddA - 1]];

In[119]:= Length[smoothedSigA2] - lhfiltA + 1

Out[119]= 900

In[120]:= offsetA = (lhfiltA - 1) / 2

Out[120]= 10

In[121]:= smoothedSigA = smoothedSigA2[[offsetA + 1 ;; -offsetA - 1]];

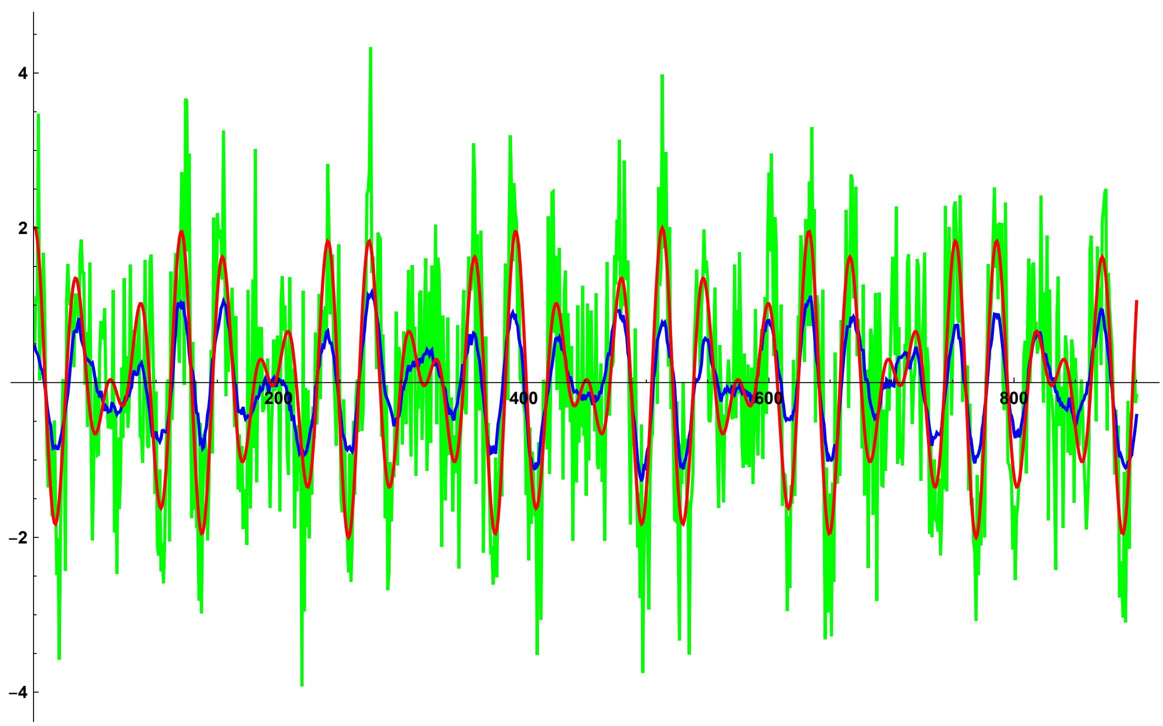
In[122]:= Length[smoothedSigA]

Out[122]= 900

Compare the smoothed noisy signal with the clean initial signal and with the noisy s

In[123]:= ListPlot[{ noisySig, smoothedSigA, Sig},
 ImageSize → 600, Joined → True, PlotStyle → {Green, Blue, Red}]

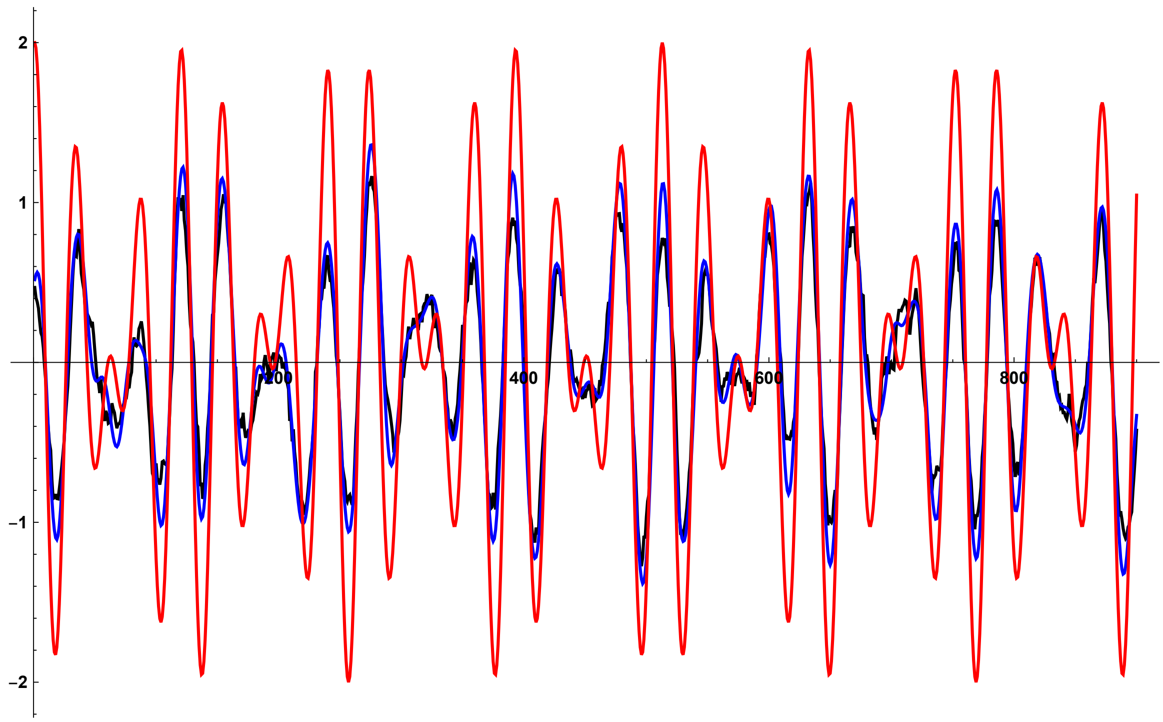
Out[123]=



Compare the smoothed noisy signal using the Gaussian smoothing weights (blue) with t
(black) with the clean initial signal (red):

In[124]:=
```
ListPlot[{ smoothedSigA, smoothedSigG , Sig},
 ImageSize → 600, Joined → True, PlotStyle → {Black, Blue, Red}]
```

Out[124]=



Clearly, Gaussian smoothing weights worked better than the Moving Average smoothing