In[1]:= **(∗DISCRETE FOURIER TRANSFORM∗)**

In[2]:= **? Fourier**

Fourier[*list*] finds the discrete Fourier transform of a list of complex numbers. ≫

In[3]:= **(∗The discrete Fourier transform v_s of a list u_r of length n is by default defined to be**
$\frac{1}{\sqrt{n}}\sum_{r=1}^{n} u_r e^{2\pi i(r-1)(s-1)/n}$ **∗)**

In[4]:= **? InverseFourier**

InverseFourier[*list*] finds the discrete inverse Fourier transform of a list of complex numbers. ≫

In[5]:= **(∗The inverse Fourier transform u_r of a list v_s of length n is defined to be**
$\frac{1}{\sqrt{n}}\sum_{s=1}^{n} v_s e^{-2\pi i(r-1)(s-1)/n}$ **∗)**

In[6]:= **list = {1, 2, 3};**

In[7]:= **Fourier[list](∗can be only numerical∗)**

Out[7]= {3.4641 + 0. ⅈ, −0.866025 − 0.5 ⅈ, −0.866025 + 0.5 ⅈ}

In[8]:= **Chop[%]**

Out[8]= {3.4641, −0.866025 − 0.5 ⅈ, −0.866025 + 0.5 ⅈ}

In[9]:= **myFour[list_] := Block[{n = Length[list]}, 1 / Sqrt[n] ∗ Table[Sum[list[[r]] Exp[2 Pi I (r − 1) ∗ (s − 1) / n], {r, n}], {s, n}]]**

In[10]:= **myFour[list]**

Out[10]= $\left\{2\sqrt{3},\ \frac{1+3e^{-\frac{2i\pi}{3}}+2e^{\frac{2i\pi}{3}}}{\sqrt{3}},\ \frac{1+2e^{-\frac{2i\pi}{3}}+3e^{\frac{2i\pi}{3}}}{\sqrt{3}}\right\}$

In[11]:= ***N*[%](∗same thing∗)**

Out[11]= {3.4641, −0.866025 − 0.5 ⅈ, −0.866025 + 0.5 ⅈ}

In[12]:= **list = {*a*, *b*, *c*};**

In[13]:= **myFour[list]**

Out[13]= $\left\{\frac{a+b+c}{\sqrt{3}},\ \frac{a+c\,e^{-\frac{2i\pi}{3}}+b\,e^{\frac{2i\pi}{3}}}{\sqrt{3}},\ \frac{a+b\,e^{-\frac{2i\pi}{3}}+c\,e^{\frac{2i\pi}{3}}}{\sqrt{3}}\right\}$

In[14]:= **Clear[prod]; prod[list1_, list2_] := Conjugate[list1].list2(∗this is the "scalar product"∗)**

In[15]:= **list1 = {*I*, −*I*, Exp[*I* Pi / 4]}; list2 = list;**

In[16]:= **prod[list1, list]**

Out[16]= $-i\,a + i\,b + c\,e^{-\frac{i\pi}{4}}$

In[17]:= **Clear[*e*]; *e*[k_] := Table[Exp[2 Pi *I* (*m* − 1) ∗ (*k* − 1) / *n*] , {*m*, *n*}](∗n will be dimension∗)**

In[18]:= ***n* = 4; *e*[2]**

Out[18]= {1, ⅈ, −1, −ⅈ}

In[19]:= **prod[*e*[3], *e*[2]] (∗orthogonality∗)**

Out[19]= 0

In[20]:= **prod[*e*[3], *e*[3]]**

Out[20]= 4

In[21]:= **A = Table[*e*[*i*], {*i*, *n*}];**

In[22]:= **A // MatrixForm**

Out[22]//MatrixForm=

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}$$

In[23]:= **Clear[myFlist]; myFlist[list_] := Block[{*n* = Length[list]}, Block[{*A* = Table[*e*[*i*], {*i*, *n*}]}, 1 / Sqrt[*n*] ∗ *A*.list]]**

In[24]:= **myFlist[list](∗same thing∗)**

Out[24]= $\left\{\frac{a+b+c}{\sqrt{3}}, \frac{a+c\,e^{-\frac{2i\pi}{3}}+b\,e^{\frac{2i\pi}{3}}}{\sqrt{3}}, \frac{a+b\,e^{-\frac{2i\pi}{3}}+c\,e^{\frac{2i\pi}{3}}}{\sqrt{3}}\right\}$

In[25]:= **myFinverse := Block[{*n* = Length[#]}, Block[{*A* = Table[*e*[*i*], {*i*, *n*}]}, Sqrt[*n*] ∗ Inverse[*A*].#]] &**

In[26]:= **myFinverse[myFlist[list]];**

In[27]:= **FullSimplify[%] (∗the original list∗)**

Out[27]= {a, b, c}

In[28]:= **(∗now let us apply to smoothing of data − will use a convolution theorem in discrete version∗)**

In[29]:= **Clear[*a*, *s*, *d*, kist, kern, fkern]**

In[30]:= ***d*[list_] := Length[list]**

In[31]:= ***s*[list_] := Plus @@ list (∗sum of all elements of list∗)**

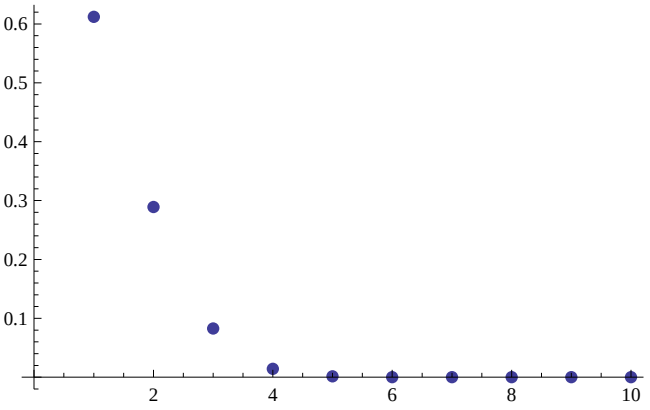
In[32]:= **kist[a_, list_] := Table[*N*[Exp[−(*i* / *a*)^2]], {*i*, *d*[list]}] (∗a corresp to width of kernel;a large−−>ker broad;**
prelim to construct Gaussian kernel;
not normalized:need sum of all elements to be 1∗)

In[33]:= **kern[a_, list_] := kist[*a*, list] / *s*[kist[*a*, list]]**
(∗Gaussian kernel because divide by value − the sum of all elements is 1 − see e.g. below∗)

In[34]:= **list = Table[*i*, {*i*, 10}];**

In[35]:=
```
ListPlot[kern[2, list], PlotStyle → PointSize[.02], AxesOrigin → {0, 0}]
```

Out[35]=



In[36]:=
```
s[kern[2, list]]
```

Out[36]=
```
1.
```

In[37]:=
```
fkern[a_, list_] := Fourier[kern[a, list]] // Chop(*FourierTransform of the Kernel*)
```

In[38]:=
```
g[a_, list_] := InverseFourier[Fourier[list] * fkern[a, list]] (*InvFT of the product~=convultion of list with kernel*)
```

In[39]:=
```
ans[a_, list_] := g[a, list] * Sqrt[d[list]] // Chop
    (*mult by Sqrt[n] bc Fourier and fkern both have 1/Sqrt[n] and InvFour accounts for one of them...*)
```

In[40]:=

In[41]:=
```
(*Example below*)
```

In[42]:=
```
list = {1, 2, 3};
```

In[43]:=
```
ans[1., list] (*almost the same as original list − kern is narrow*)
```

Out[43]=
```
{1.09514, 1.95291, 2.95195}
```

In[44]:=
```
ans[10., list] (*for a broad kernel every element is close to average, 2*)
```

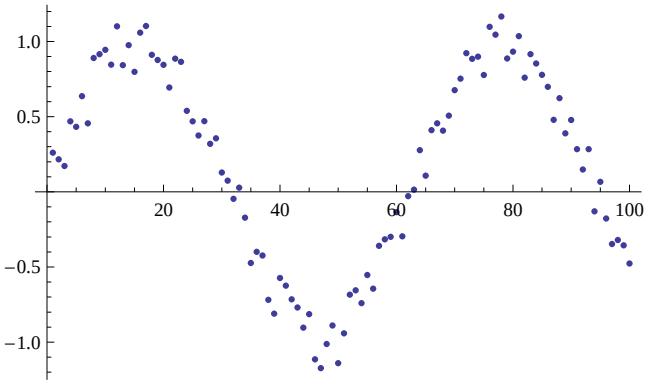
Out[44]=
```
{1.98979, 1.98364, 2.02657}
```

In[45]:=
```
(*this was a bit artificial example;now take something more "real"*)
```

In[46]:=
```
res = Table[Sin[0.1 x] + 0.4 * (Random[] − 1 / 2), {x, 100}];(*Random[] gives # btwn 0 and 1;
creating a smooth funct plus some noise*)
```

In[47]:= **ListPlot[res]**

Out[47]=



In[48]:= **red = RGBColor[1, 0, 0]; blue = RGBColor[0, 0, 1];**

In[49]:= **ploSin := Plot[Sin[*x* / 10] + .1, {*x*, 0, 100}, PlotStyle → blue, DisplayFunction→ Identity]**

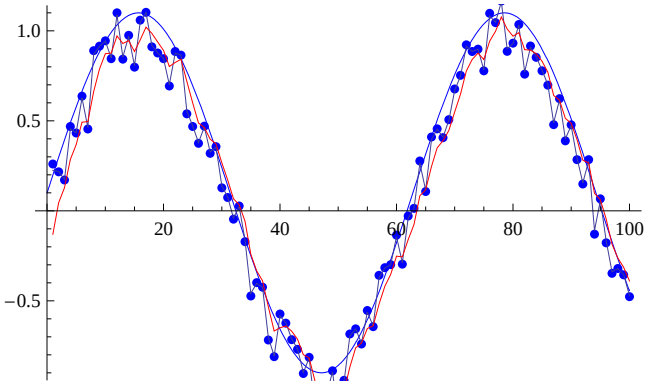
In[50]:= **ploRes := ListPlot[res, PlotStyle → {blue, PointSize[.015]}, DisplayFunction→ Identity]**

In[51]:= **plo1 := ListPlot[ans[1, res], PlotJoined → True, DisplayFunction→ Identity]**

In[52]:= **plo3 := ListPlot[ans[3, res], PlotJoined → True, DisplayFunction→ Identity, PlotStyle → red]**

In[53]:= **Show[ploSin, ploRes, plo1, plo3, DisplayFunction→ $DisplayFunction]**

Out[53]=

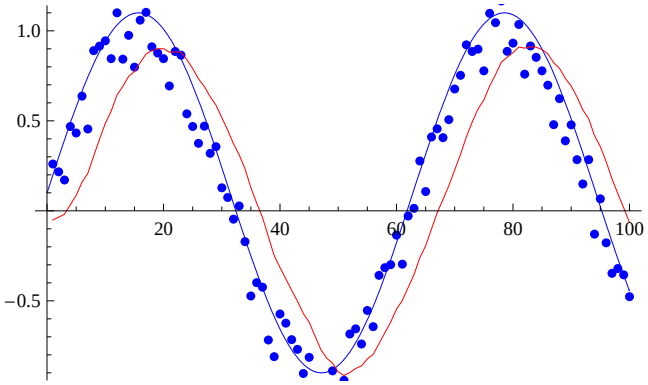


In[54]:= **(∗note that black line – plo1 – practically does not alter original data∗)**

In[55]:= **plo10 := ListPlot[ans[10, res], PlotJoined → True, DisplayFunction→ Identity, PlotStyle → red]**

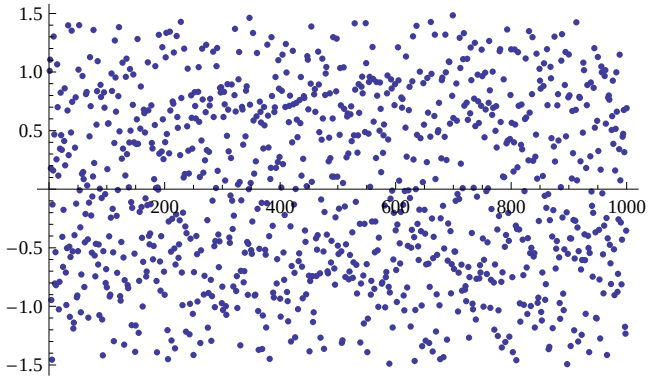
In[56]:= **Show[ploSin, ploRes, plo10, DisplayFunction→ $DisplayFunction]**

Out[56]=



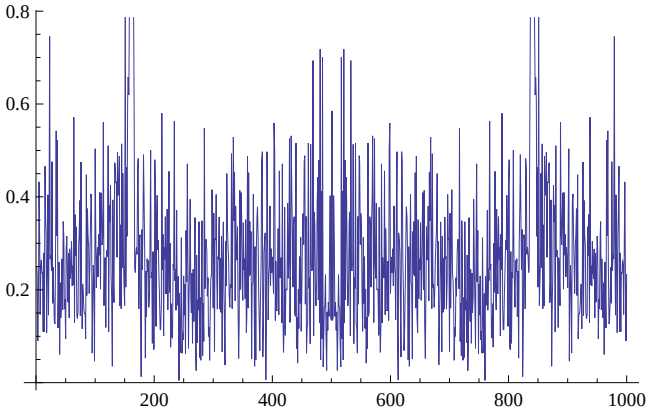
In[57]:=
```
(*extracting fourier-components*)
res = Table[Sin[ x] + (Random[] - 1 / 2), {x, 1000}];
ListPlot[res]
(*practically no useful information*)
```

Out[58]=



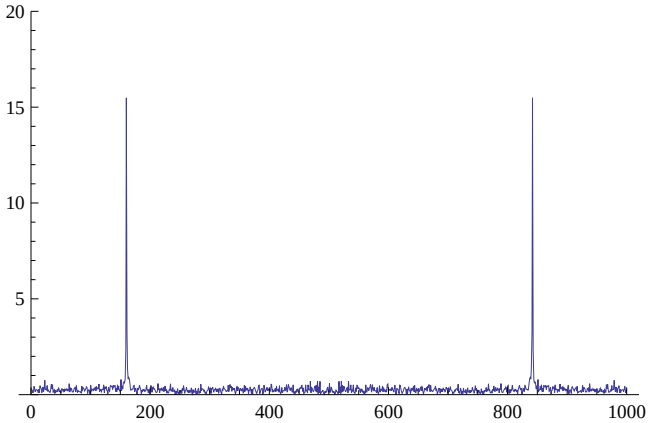
In[59]:=
```
ListPlot[Abs[Fourier[res]], PlotJoined → True](*too much noise on small scale*)
```

Out[59]=



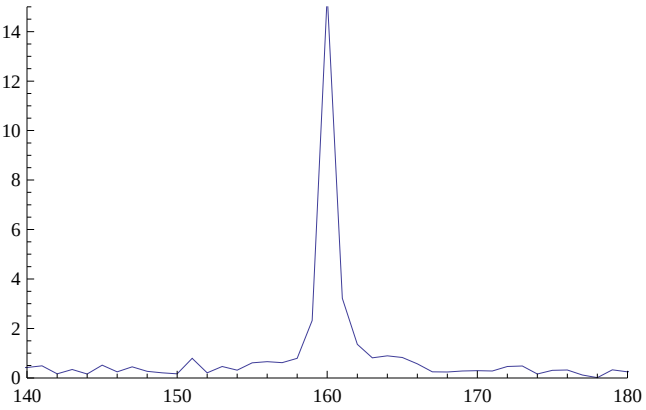
In[60]:=
```
ListPlot[Abs[Fourier[res]], PlotJoined → True, PlotRange → {0, 20}]
```

Out[60]=



In[61]:=
```
ListPlot[Abs[Fourier[res]], PlotJoined → True, PlotRange → {{140, 180}, {0, 15}}]
```

Out[61]=



In[62]:=
```
(*the peak is at 1000/(2Pi)~=160*)
```

In[63]:=

In[64]:=

In[65]:=

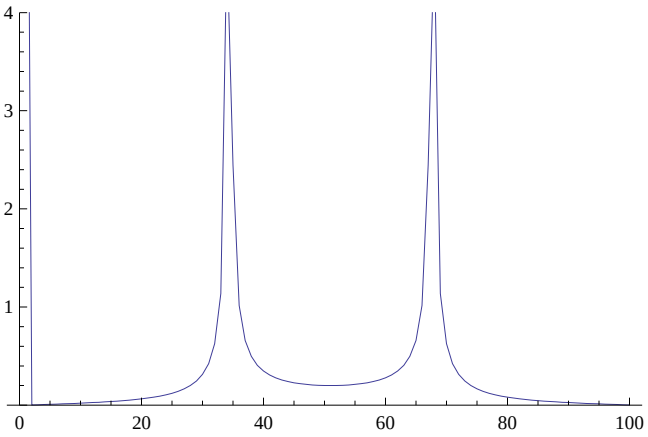
In[66]:=

In[67]:=
```
(*consider now a periodic non-sinusoidal function*)
```

In[68]:=
```
res = Table[Mod[i, 3], {i, 100}];
```

In[69]:=
```
ListPlot[Abs[Fourier[res]], PlotJoined → True, PlotRange → {0, 4}]
```

Out[69]=



In[70]:=
```
(*the peak is at k=n/period ~= 33, since 2Pi are included in
    definition. There is a mirror peak at
    n-n/period~=67*)
```

In[71]:=