

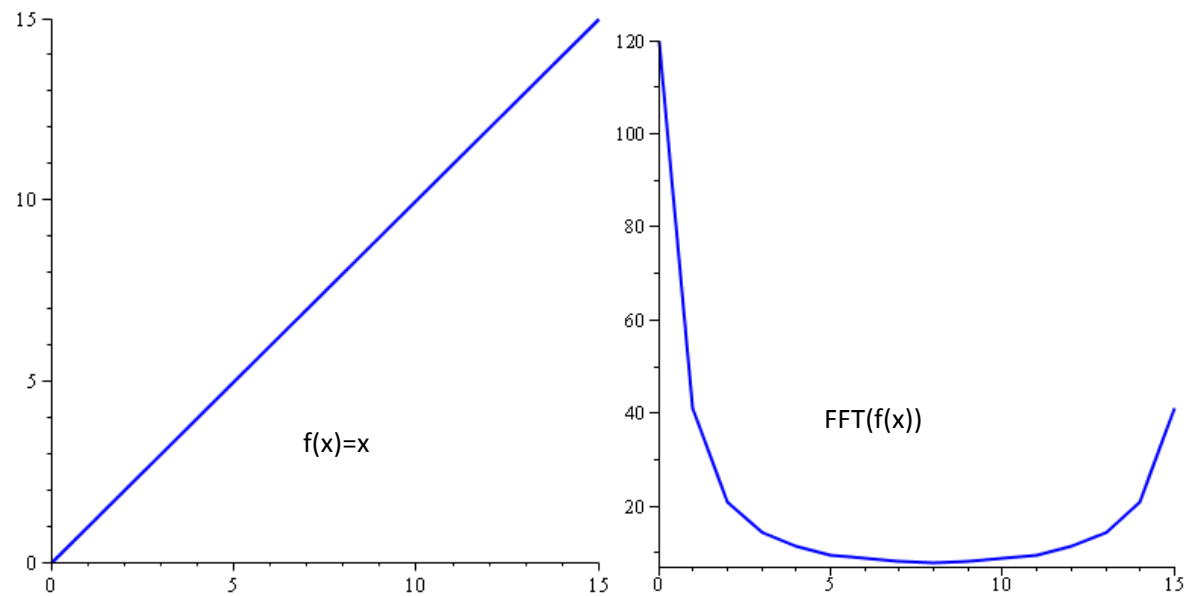
Question 1. Using loops and the maple function add(), we calculate X0, X1, X2 and then F extending the lesson method on 16 bits instead of 8. It can be implemented this way in Maple:

```

> restart:
> N:=16:
> f:=array(0..N-1):
> for i from 0 to N-1 do f[i]:=i;
> od:
> W:=evalf(exp(-2*Pi*I/N)):
> for a2 from 0 to 1 do
>   for a1 from 0 to 1 do
>     for a0 from 0 to 1 do
>       for b0 from 0 to 1 do
>         add(f[8*a3+4*a2+2*a1+a0]*W^(8*a3*b0),a3=0..1);
>         X0[8*a2+4*a1+2*a0+b0]:=%;
>       od: od: od: od:
>       for a1 from 0 to 1 do
>         for a0 from 0 to 1 do
>           for b0 from 0 to 1 do
>             for b1 from 0 to 1 do
>               add(X0[8*a2+4*a1+2*a0+b0]*W^(4*a2*(2*b1+b0)),a2=0..1);
>               X1[8*a1+4*a0+2*b0+b1]:=%;
>             od: od: od: od:
>             for a0 from 0 to 1 do
>               for b0 from 0 to 1 do
>                 for b1 from 0 to 1 do
>                   for b2 from 0 to 1 do
>                     add(X1[8*a1+4*a0+2*b0+b1]*W^(2*a1*(4*b2+2*b1+b0)),a1=0..1);
>                     X2[8*a0+4*b0+2*b1+b2]:=%;
>                   od: od: od: od:
>                   for b3 from 0 to 1 do
>                     for b2 from 0 to 1 do
>                       for b1 from 0 to 1 do
>                         for b0 from 0 to 1 do
>                           add(X2[8*a0+4*b0+2*b1+b2]*W^(a0*(8*b3+4*b2+2*b1+b0)),a0=0..1);
>                           F[8*b3+4*b2+2*b1+b0]:=%;
>                         od: od: od: od:

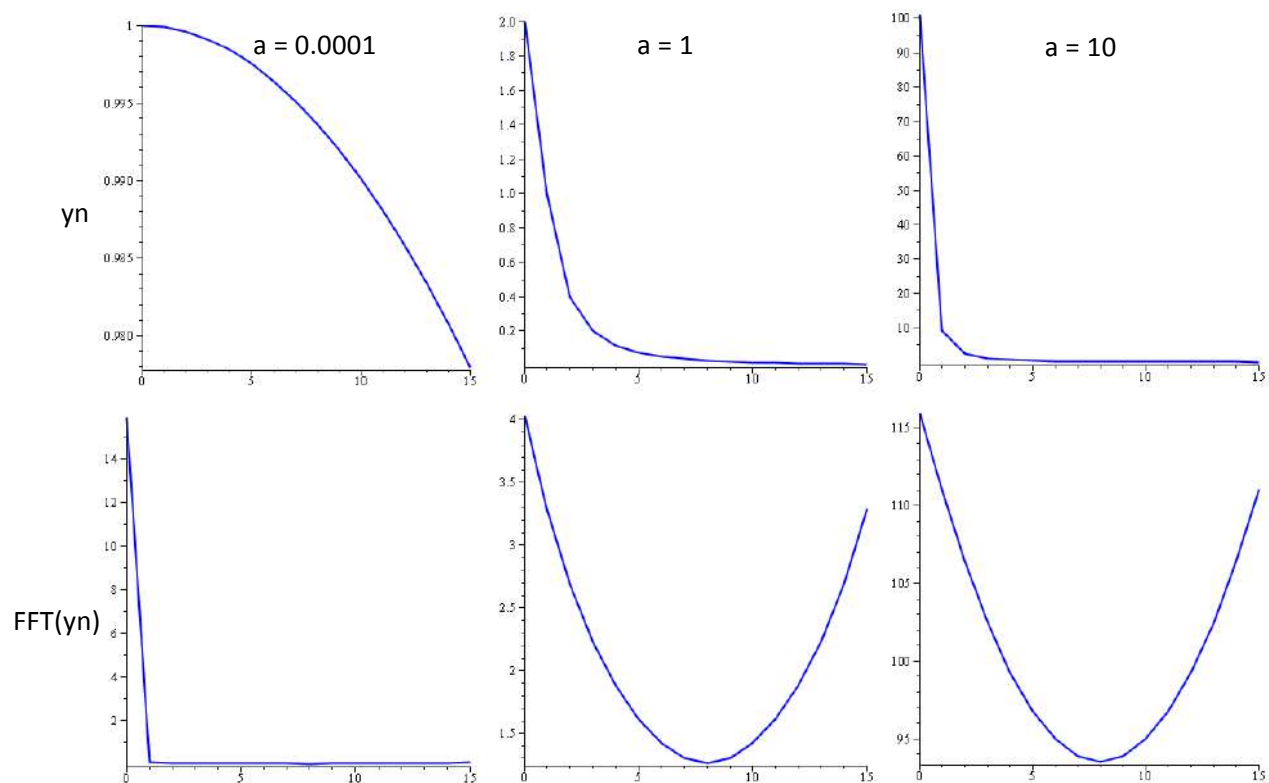
```

Plots of the example function f and its Fourier transform using the above FFT algorithm:



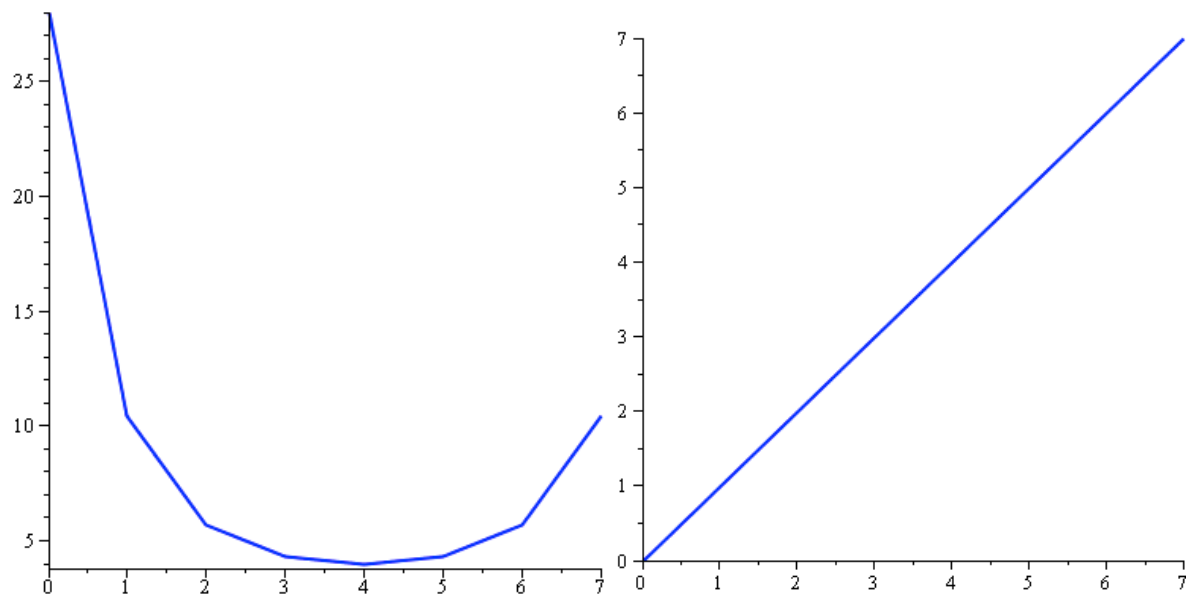
Question 2.

Applying the same algorithm to $\frac{1+a^2}{1+an^2}$ for different values of a :

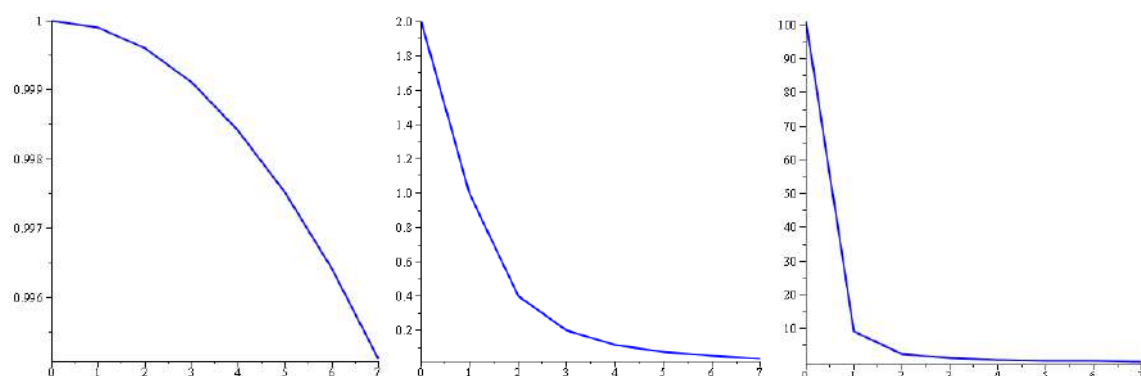


Question 3.

Reversing the above FFT algorithm can do the inverse FFT. We begin with F instead of f , and then we find the different X and finally find f . For $N=8$ there are the results with the trivial $f(x)=x$ function. With the 8 points FFT on the left and the recovered function on the right.

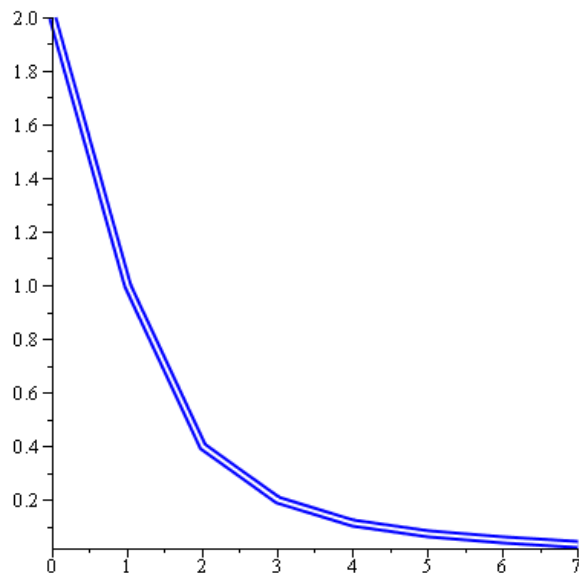
**Question 4.**

We use the inverse Fourier transform algorithm on 3 bits for the same values than in Question 2. So we find back the original function but on 8 points.



Question 5.

We can see on this plot that the inverse Fourier transform (white) is exactly the same than the base y_n function (blue).

**Question 6.**

As y_n is a parametric discrete function, we have to talk about specific values of a :

When $a = 1$, y_n is equivalent to the discrete function: $2 * \frac{1}{1+n^2}$ is it nearly the derivative of the inverse function so it has its shape. It's the same shape for all values of a , except if ' a ' is really small. More the value of a increase, more the curve will tend to 0 and near the (Ox) axis.

When a is small, f has the inversed shape.