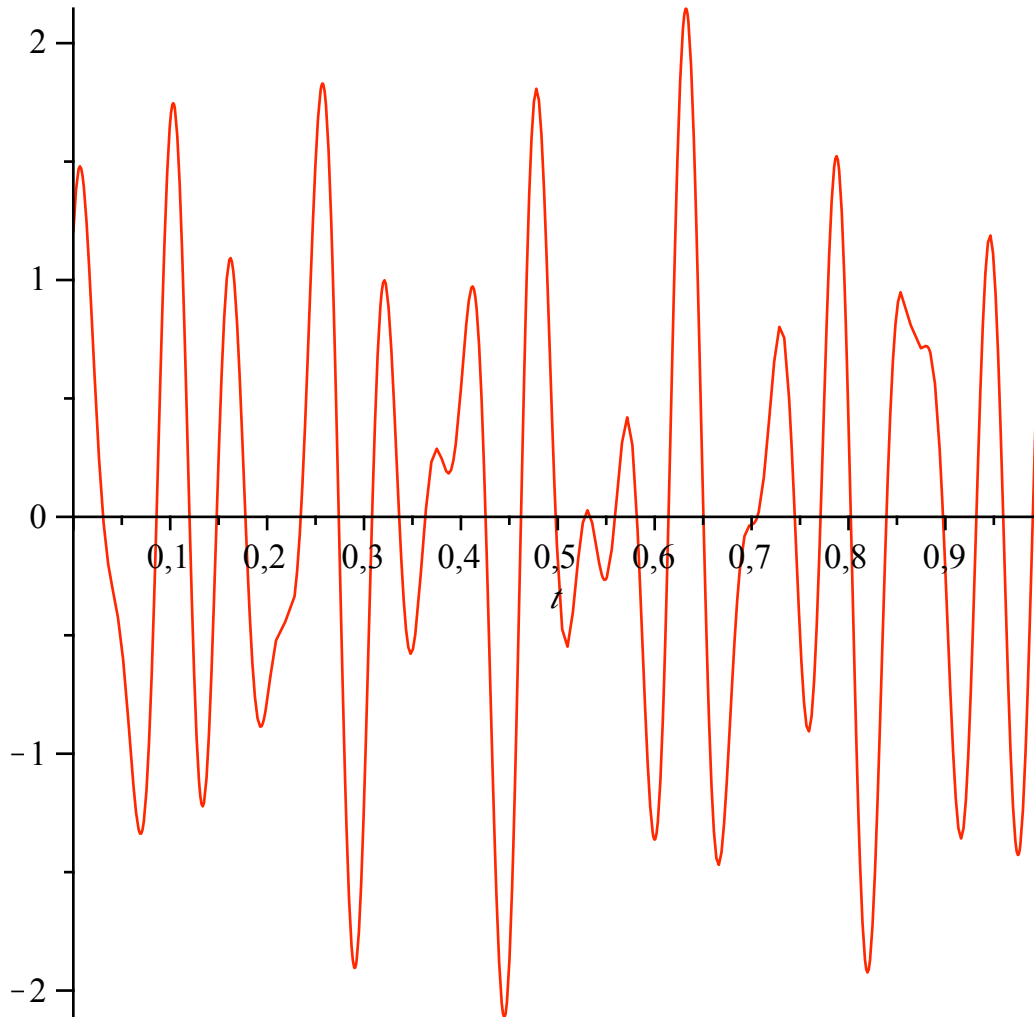


▼ Exemple 1 (somme de sinusoides)

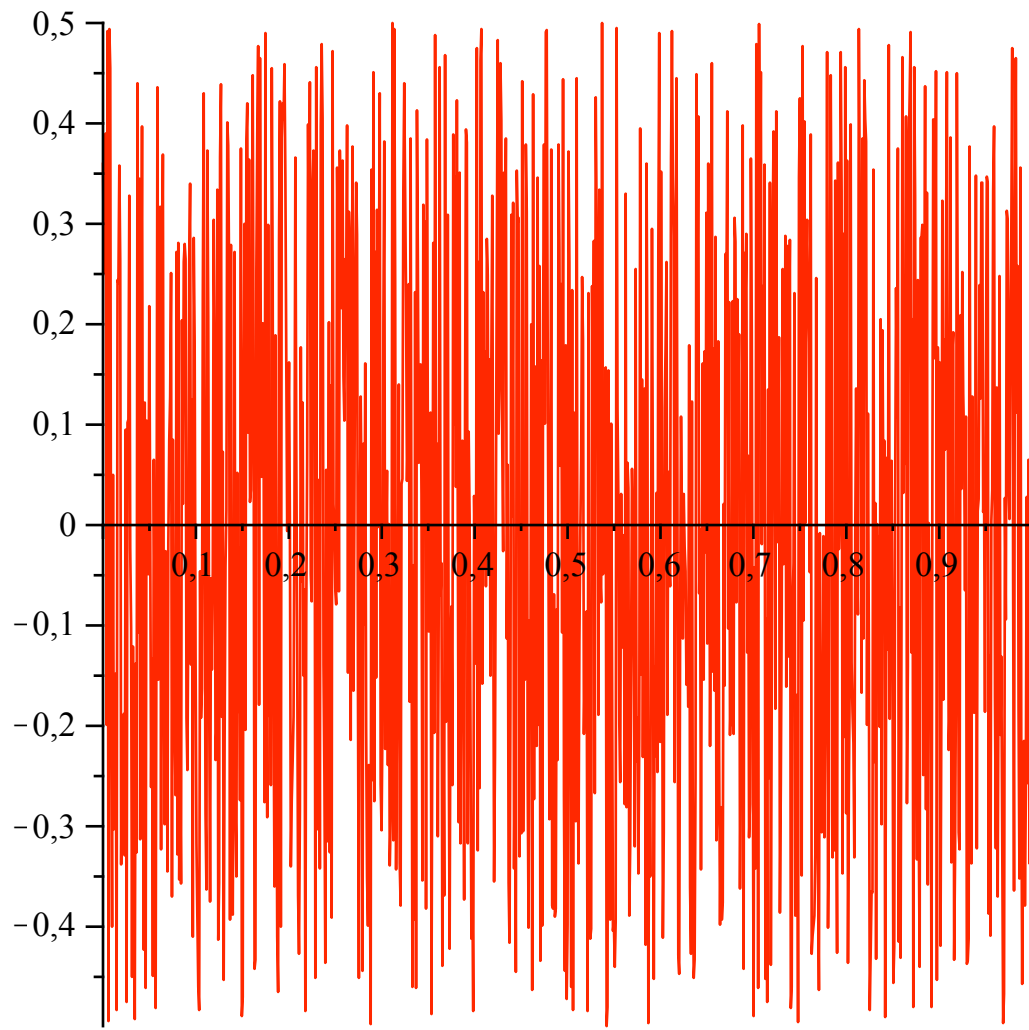
```
> restart;  
> X:=sin(13*2*Pi*t)+0.5*cos(8*2*Pi*t)+0.7*cos(19*2*Pi*t);  
      X:=sin(26 π t) + 0.5 cos(16 π t) + 0.7 cos(38 π t)  
> plot(X,t=0..1.023/1024);
```

(1.1)

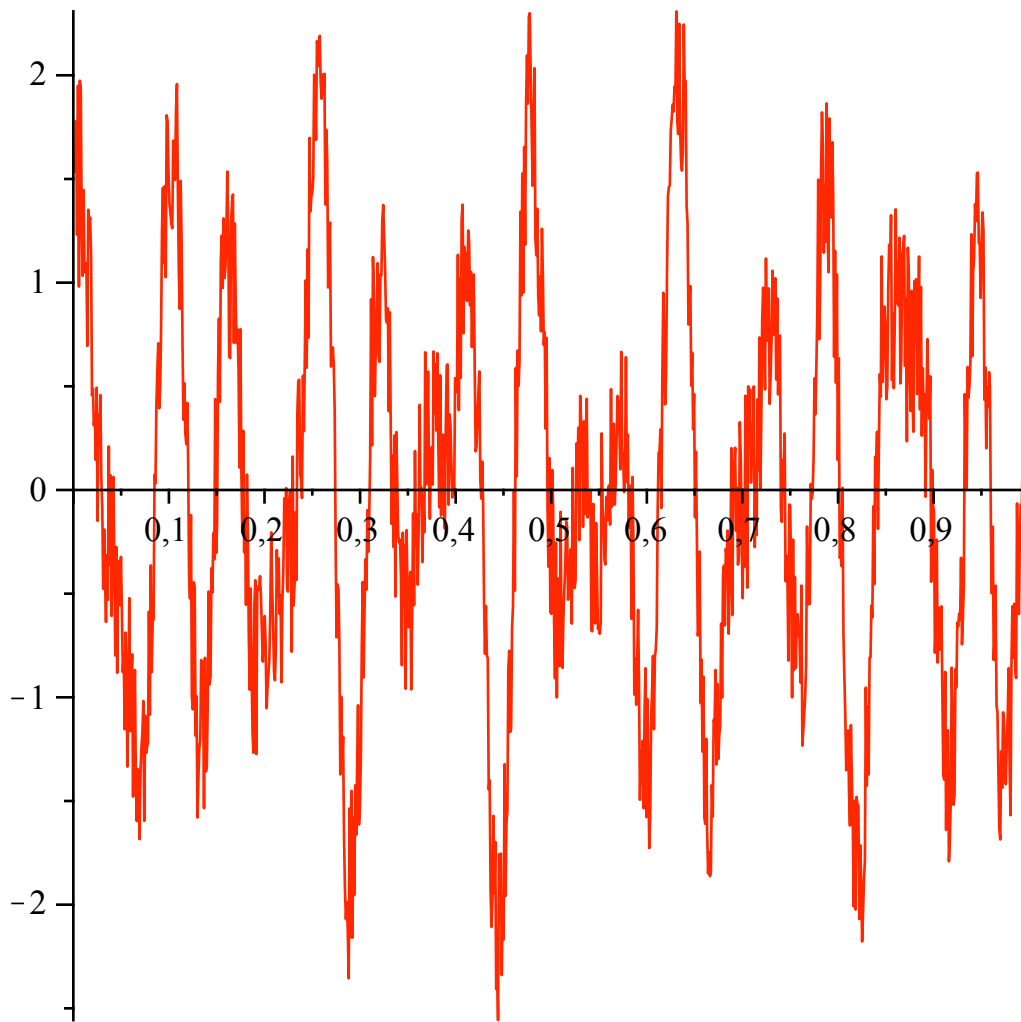


```
>  
> t:=[evalf(($ 0..1.023)/1024)]:  
> X:=[seq([t[i],evalf(sin(13*2*Pi*t[i])+0.5*cos(8*2*Pi*t[i])  
+0.7*cos(19*2*Pi*t[i]))],i=1..1024)]:  
> X[1];  
[0., 1.2]  
> plot(X);
```

(1.2)



```
> plot(z);
```



```
> ZZ:= [seq(Z[i][2], i=1..1024)]:
> with(DiscreteTransforms);
      [FourierTransform, InverseFourierTransform]
```

(1.4)

```
> ZZ:=convert(ZZ, Vector):
> for i from 1 to 10 do print(ZZ[i]) od;
      1.561000000
      1.533328530
      1.588508473
      1.779233318
      1.230331391
      1.949764686
      0.9806259650
      1.974134224
      1.929628584
      1.258560707
```

(1.5)

```
> F:=FourierTransform(ZZ);
```

$$F := \begin{bmatrix} 1..1024 \text{ Vector}_{\text{column}} \\ \text{Data Type: anything} \\ \text{Storage: rectangular} \\ \text{Order: Fortran_order} \end{bmatrix} \quad (1.6)$$

```
> for i from 1 to 20 do print(F[i]) od;
-0.279249995909375158 + 0. I
-0.00627542862629427122 - 0.262298879438103482 I
-0.342342517460109352 + 0.266139789292381190 I
-0.0310861118121861652 + 0.0876641229229681851 I
-0.0925413357818133526 + 0.467914373478336365 I
0.00588703977980710448 - 0.147148057705133172 I
-0.0455327905451952953 - 0.0267176507323262091 I
0.124955349939043739 - 0.0603759911925317583 I
7.99102398856521212 + 0.00868661624969278845 I
-0.0957392435713840251 + 0.00245956362614596636 I
0.0286444893855760281 - 0.0159867798434999846 I
0.0455288062304227970 + 0.560081529885917861 I
0.195941661983977777 - 0.0398927850609489126 I
-0.157624519654755629 - 16.1790776245031793 I
-0.222962959913888870 - 0.0580942632378217971 I
0.301558060769613556 - 0.117082487496512427 I
0.112359863998866174 + 0.174206205416435583 I
0.0463086433380387186 - 0.0641531350063713757 I
-0.182460143802803093 + 0.190710685257308637 I
11.3700653706289039 - 0.253898411568083071 I
```

(1.7)

```
> abs(F[9]);
7.991028710
```

(1.8)

```
> abs(F[20]);
11.37289985
```

(1.9)

```
> abs(F[14]);
16.17984543
```

(1.10)

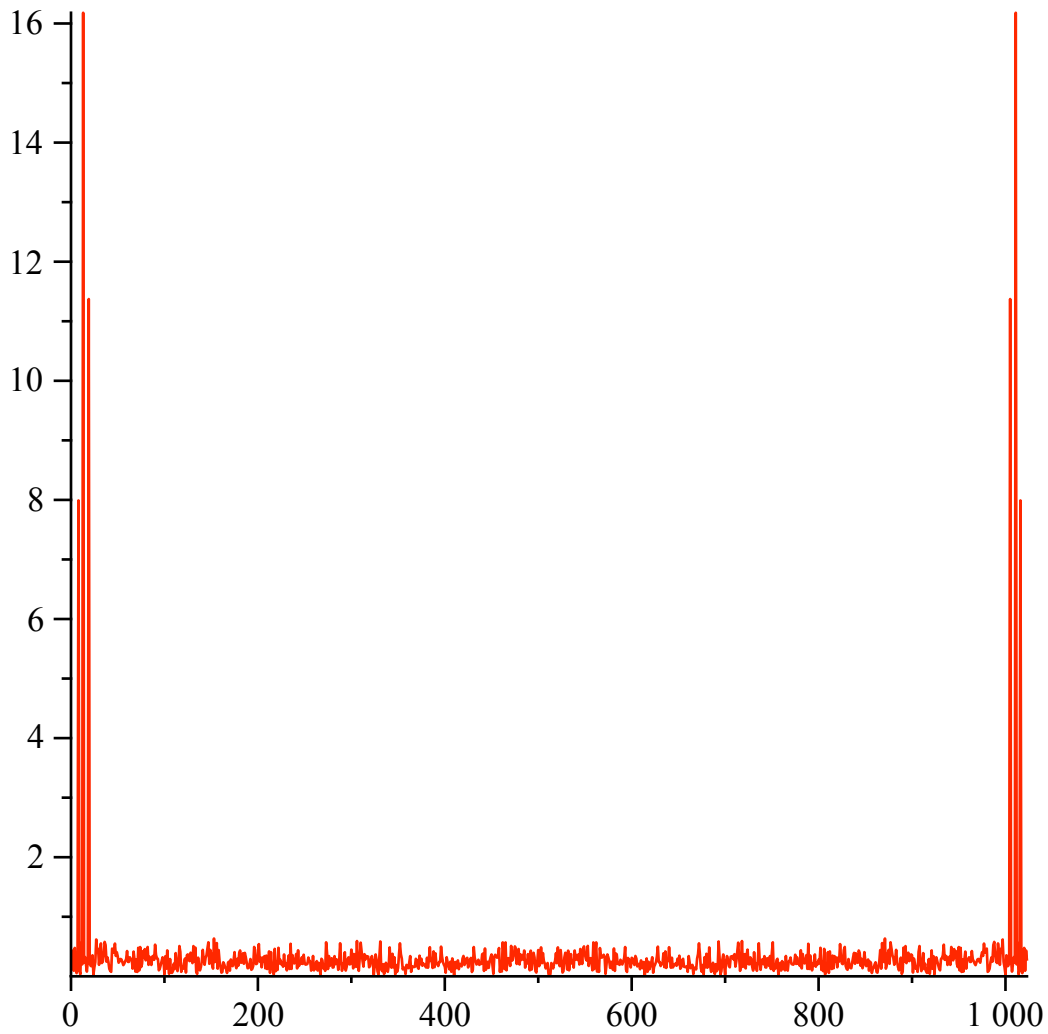
```
>
> abs(F[1017]);
7.991028710
```

(1.11)

```
> abs(F[1012]);
16.17984543
```

(1.12)

```
> FF:= [seq([i-1, abs(F[i])], i=1..1024)];
> plot(FF);
```



```
> FS:=Vector(1024);
```

```
FS:= [ 1..1024 Vectorcolumn
      Data Type: anything
      Storage: rectangular
      Order: Fortran_order ]
```

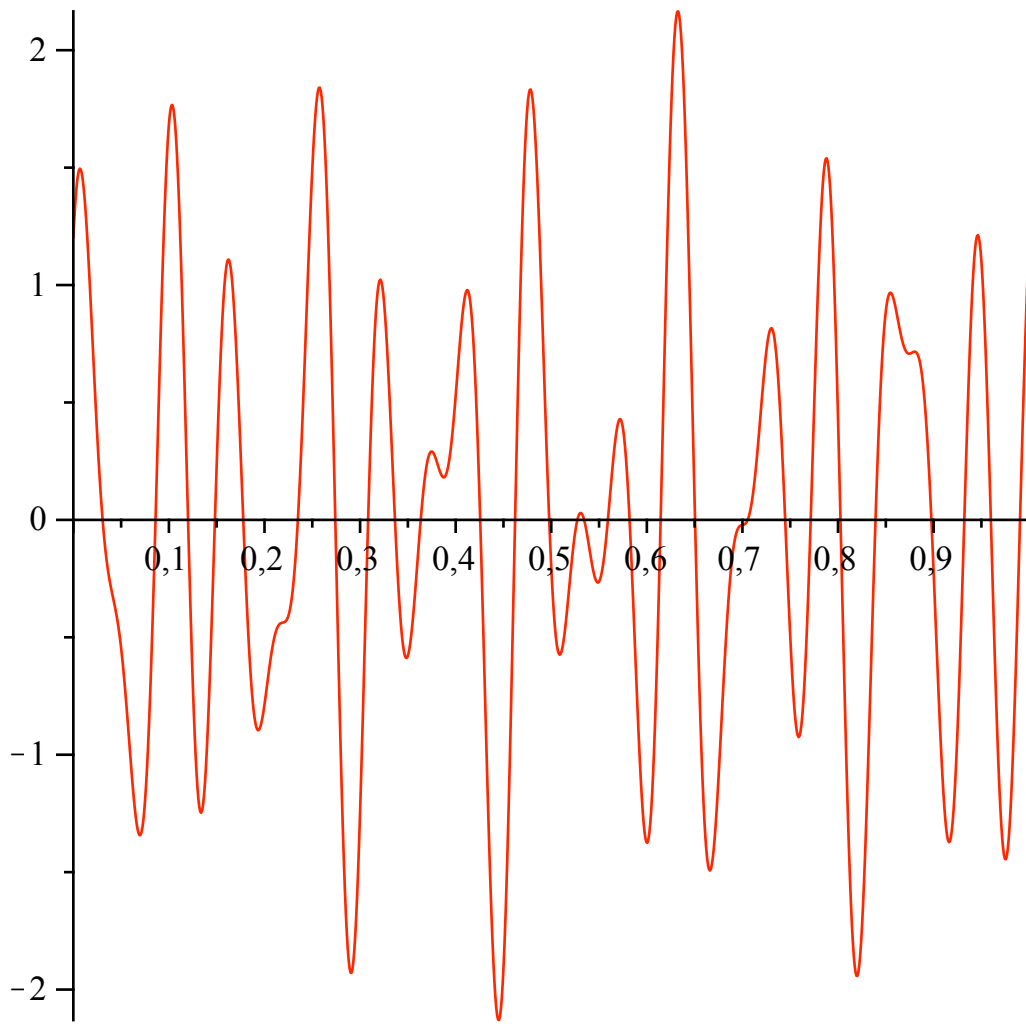
(1.13)

```
>
> for i from 1 to 1024 do if abs(F[i])>1 then FS[i]:=F[i] else
FS[i]=0 fi od;
> FS[1];
```

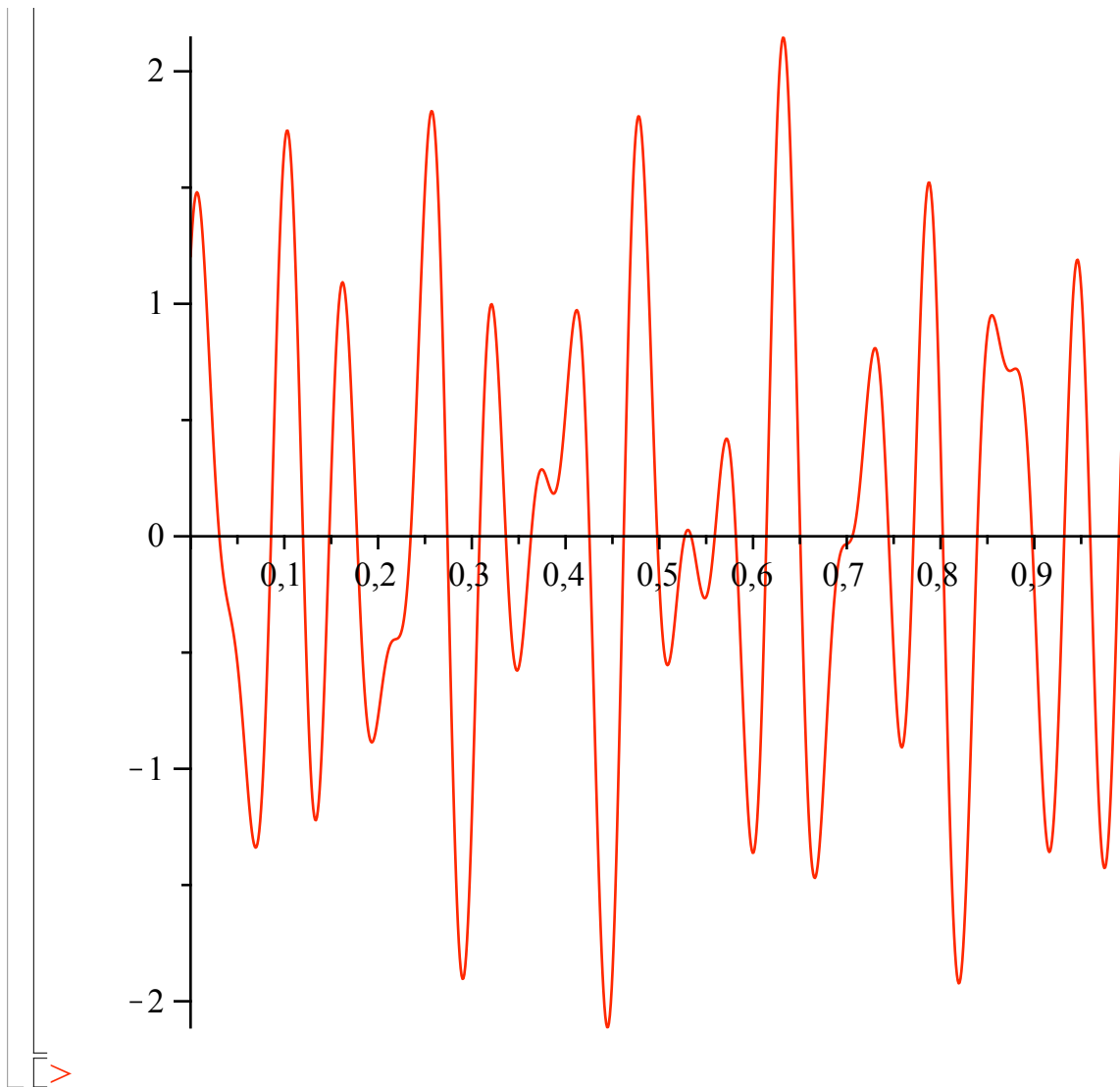
0

(1.14)

```
> S:=InverseFourierTransform(FS):
> SS:=[seq([t[i],Re(S[i])],i=1..1024)]:
> plot(SS);
```



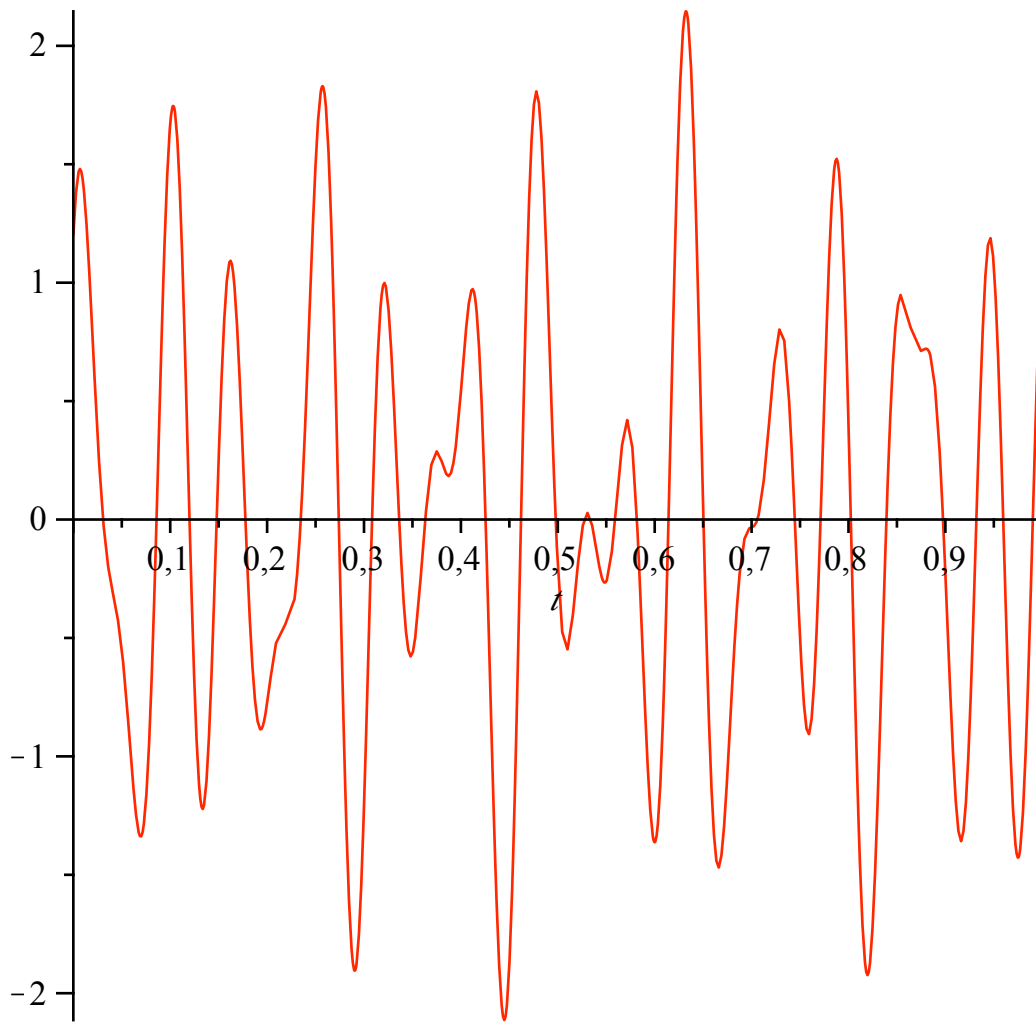
```
> plot(x);
```



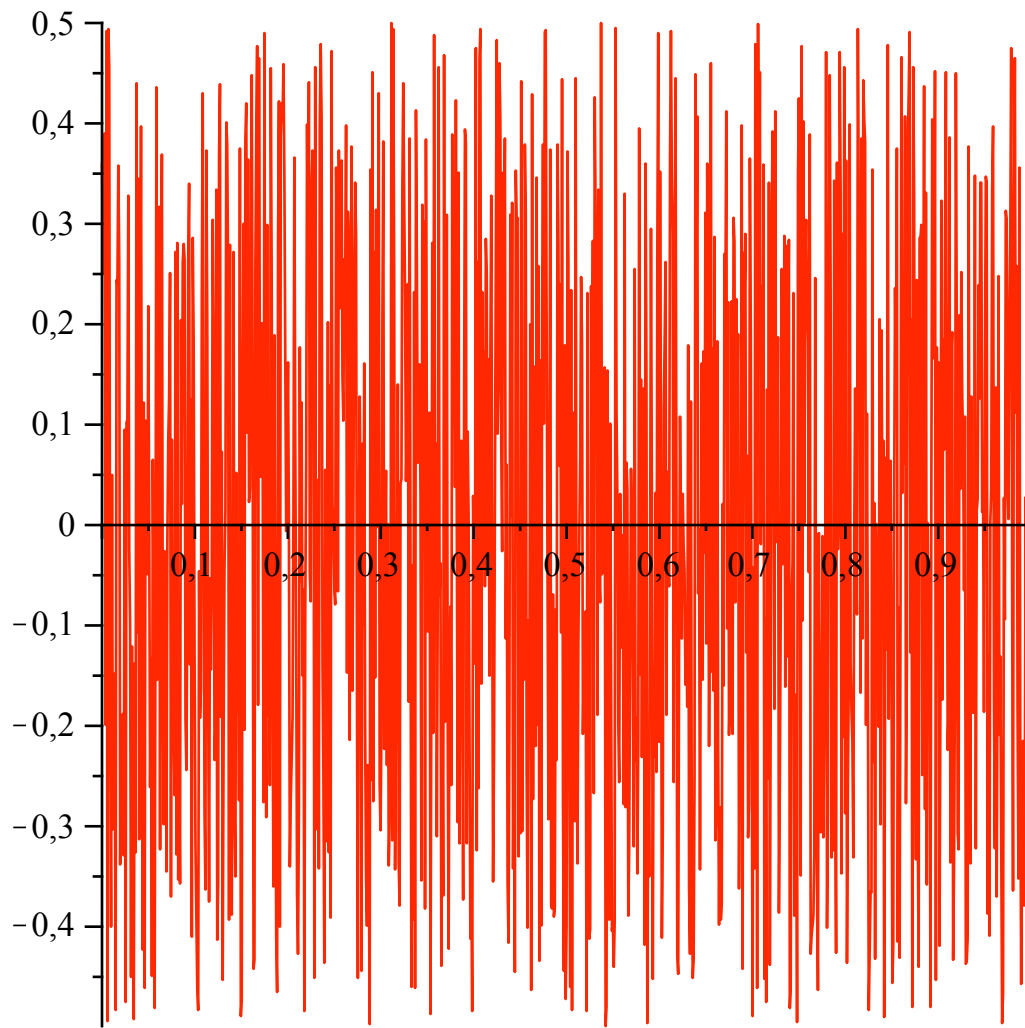
▼ Exemple 1 (autre méthode)

```
> restart;
> X:=sin(13*2*Pi*t)+0.5*cos(8*2*Pi*t)+0.7*cos(19*2*Pi*t);
      X:= sin(26 π t) + 0.5 cos(16 π t) + 0.7 cos(38 π t)
> plot(X,t=0..1023/1024);
```

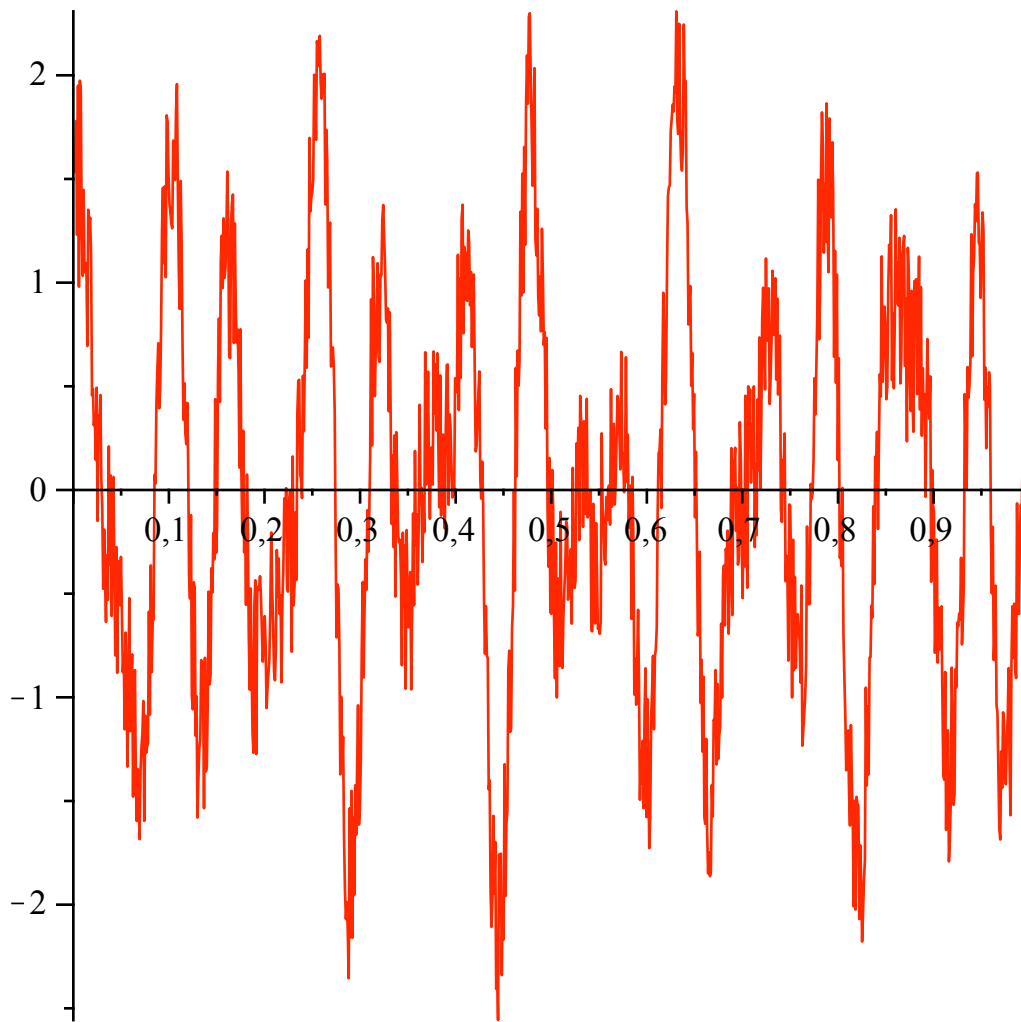
(2.1)



```
> t:=[evalf(($ 0..1.023)/1024)]:  
> X:=[seq([t[i],evalf(sin(13*2*Pi*t[i])+0.5*cos(8*2*Pi*t[i])  
+.7*cos(19*2*Pi*t[i]))],i=1..1024)]:  
> plot(X);
```

```
> plot(z);
```



```
> ZZ:= [seq(Z[i][2], i=1..1024)]:
> with(DiscreteTransforms);
      [FourierTransform, InverseFourierTransform]
```

(2.3)

```
> ZZ:=convert(ZZ, Vector):
> for i from 1 to 10 do print(ZZ[i]) od;
      1.561000000
      1.533328530
      1.588508473
      1.779233318
      1.230331391
      1.949764686
      0.9806259650
      1.974134224
      1.929628584
      1.258560707
```

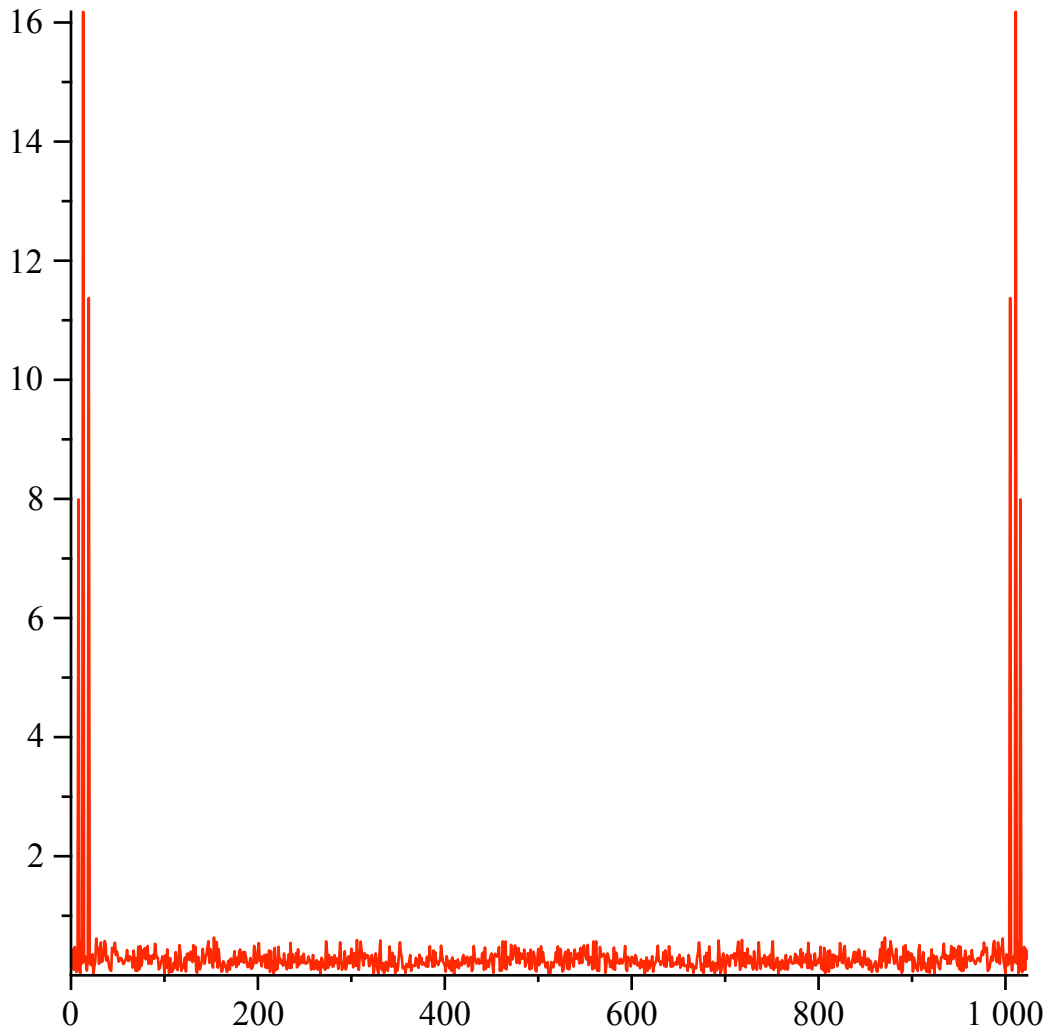
(2.4)

```
> F:=FourierTransform(ZZ);
```

$$F := \left[\begin{array}{l} 1..1024 \text{ Vector}_{\text{column}} \\ \text{Data Type: anything} \\ \text{Storage: rectangular} \\ \text{Order: Fortran_order} \end{array} \right]$$

(2.5)

```
> FF:= [seq([i-1,abs(F[i])],i=1..1024)]:
> plot(FF);
```



```
> FS:=Vector(1024);
```

$$FS := \left[\begin{array}{l} 1..1024 \text{ Vector}_{\text{column}} \\ \text{Data Type: anything} \\ \text{Storage: rectangular} \\ \text{Order: Fortran_order} \end{array} \right]$$

(2.6)

```
> for i from 1 to 30 do FS[i]:=F[i] od:
> for i from 31 to 994 do FS[i]:=0 od:
> for i from 995 to 1024 do FS[i]:=F[i] od:
```

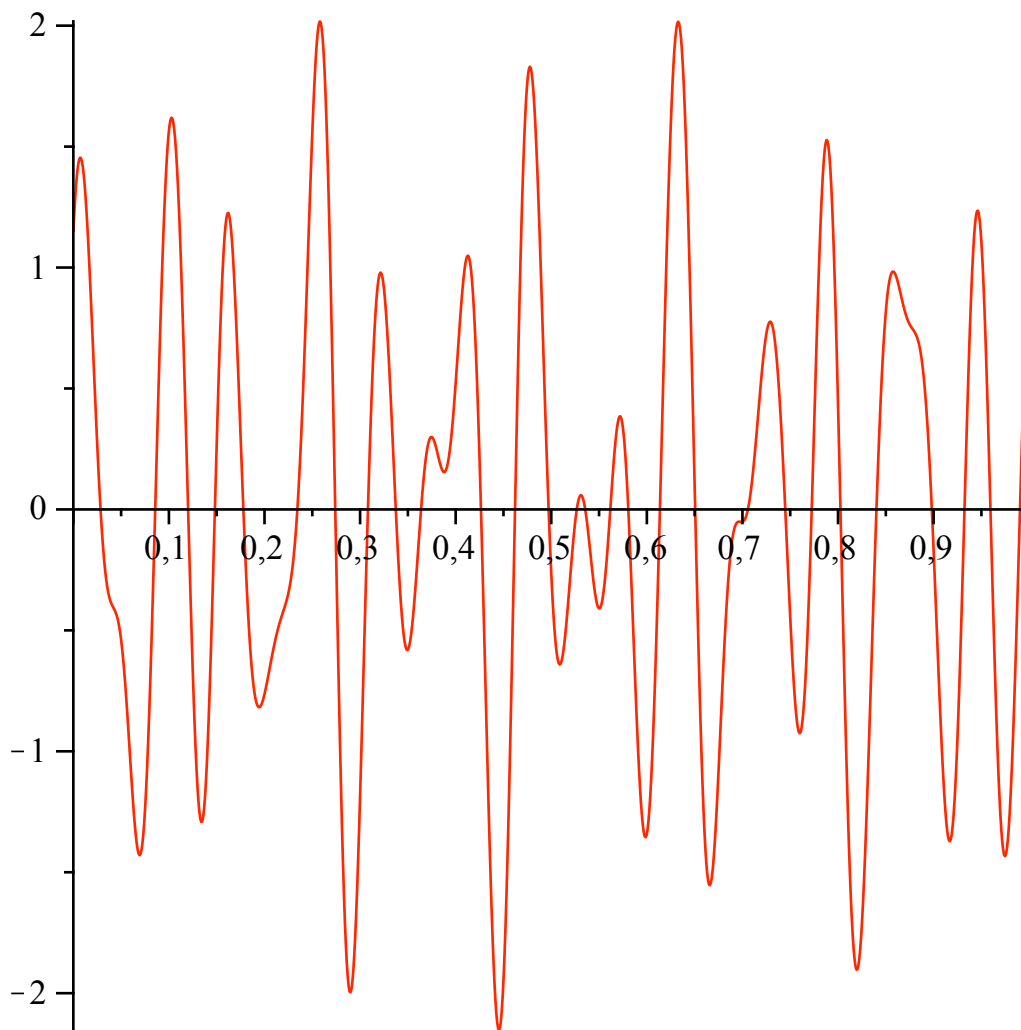
```
>
> FS[1];
```

(2.7)

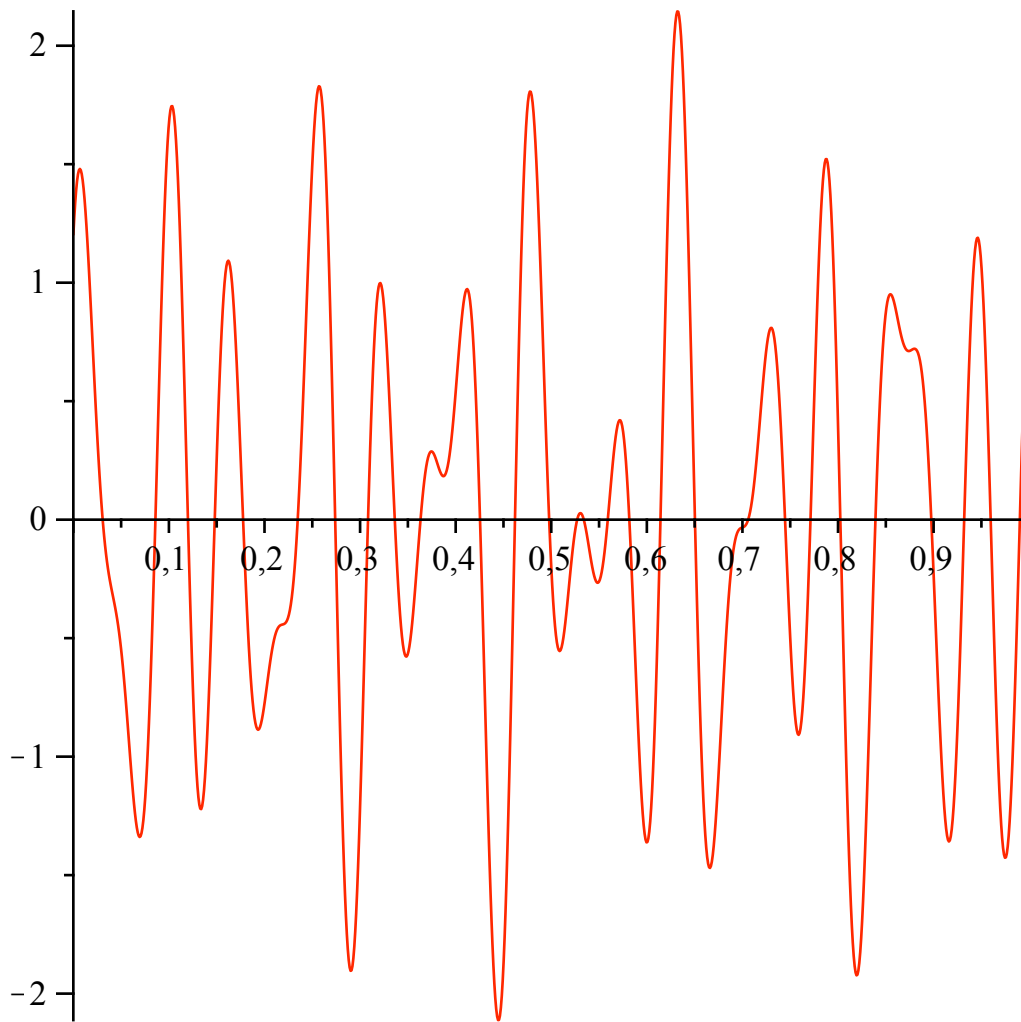
-0.279249995909375158 + 0. I

(2.7)

```
> S:=InverseFourierTransform(FS):  
> SS:=seq([t[i],Re(S[i])],i=1..1024):  
> plot(SS);
```



```
> plot(X);
```

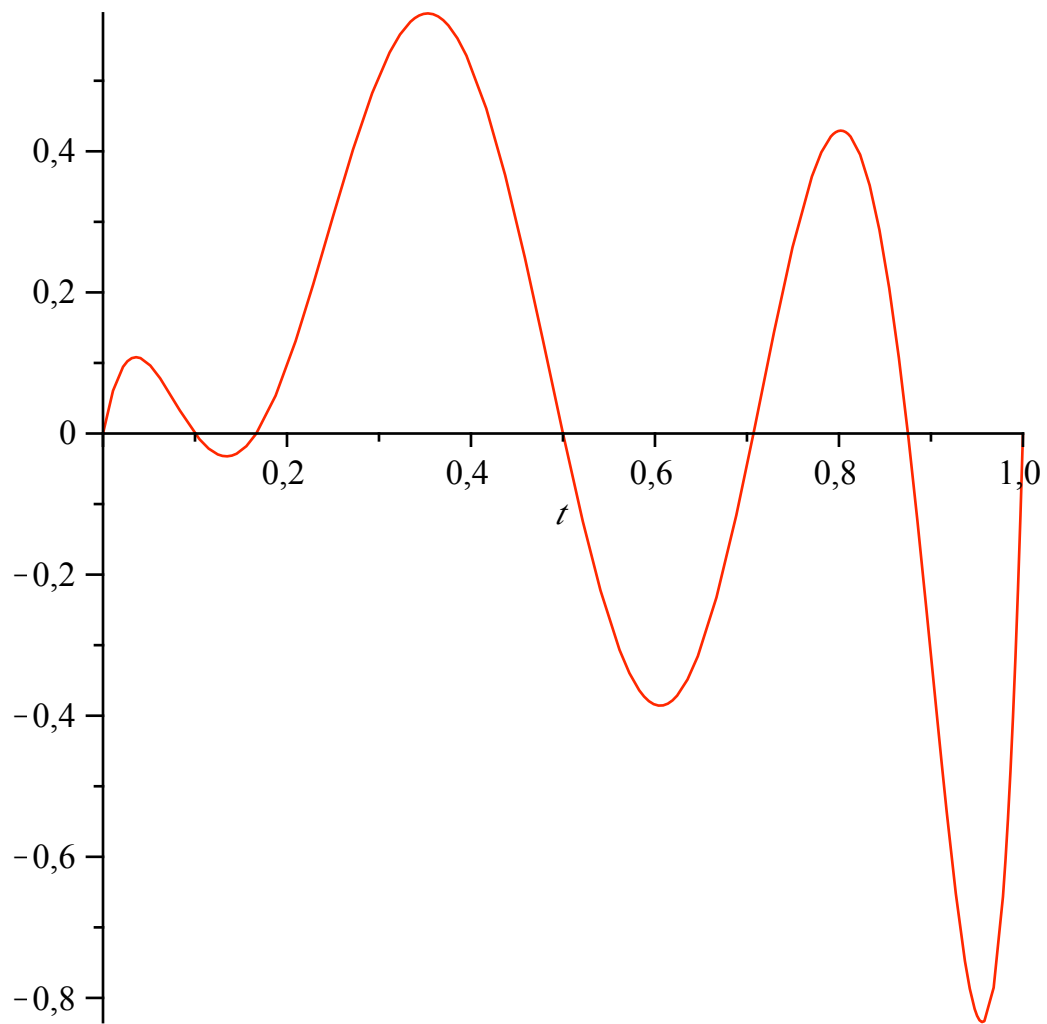


>

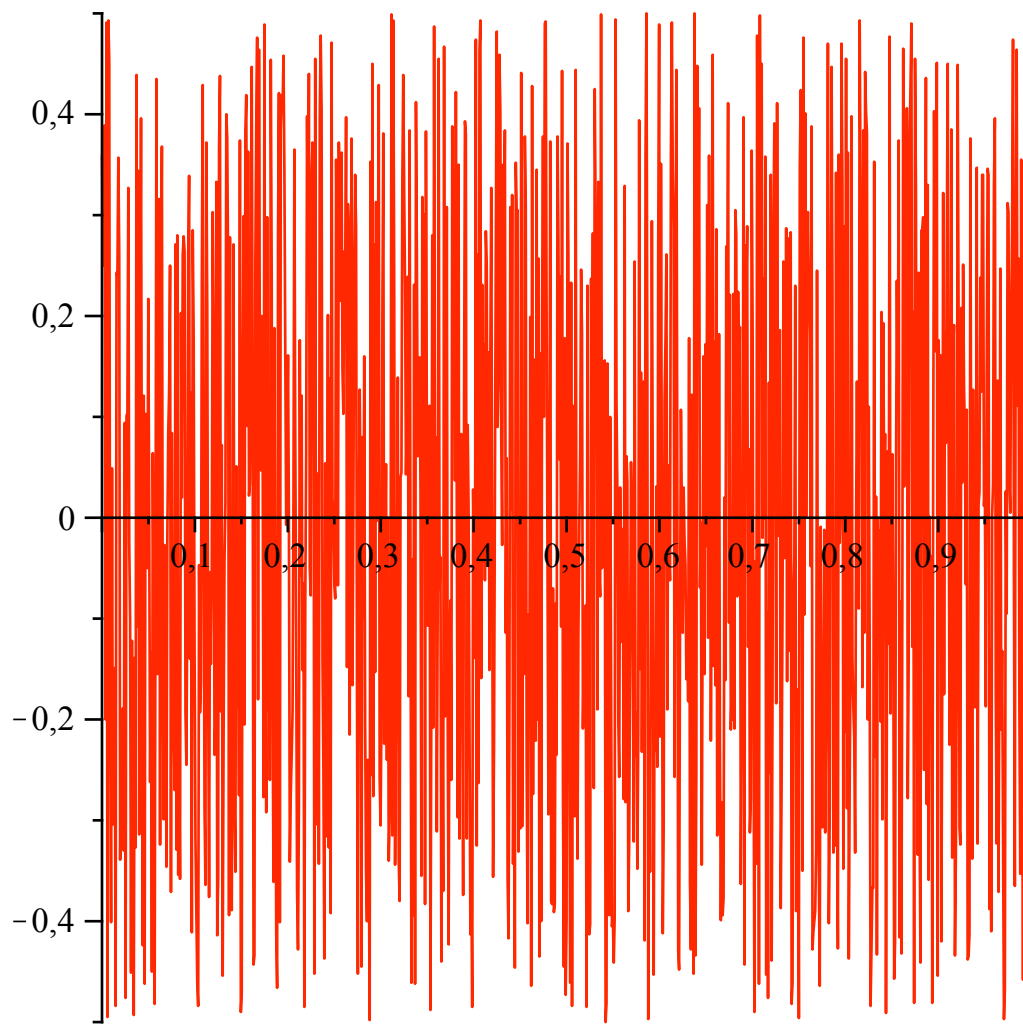
▼ Exemple 2 (polynôme)

```
> restart;
> X:=expand(1920*(t-1/6)*(t^2-1/2)*(t-1/2)*(t-7/8)*(t-1/10)*
(t-1)*t);
      X:=1920 t8 - 5072 t7 + 3768 t6 + 692 t5 - 2082 t4 + 908 t3 - 141 t2 + 7 t
> plot(X,t=0..1);
```

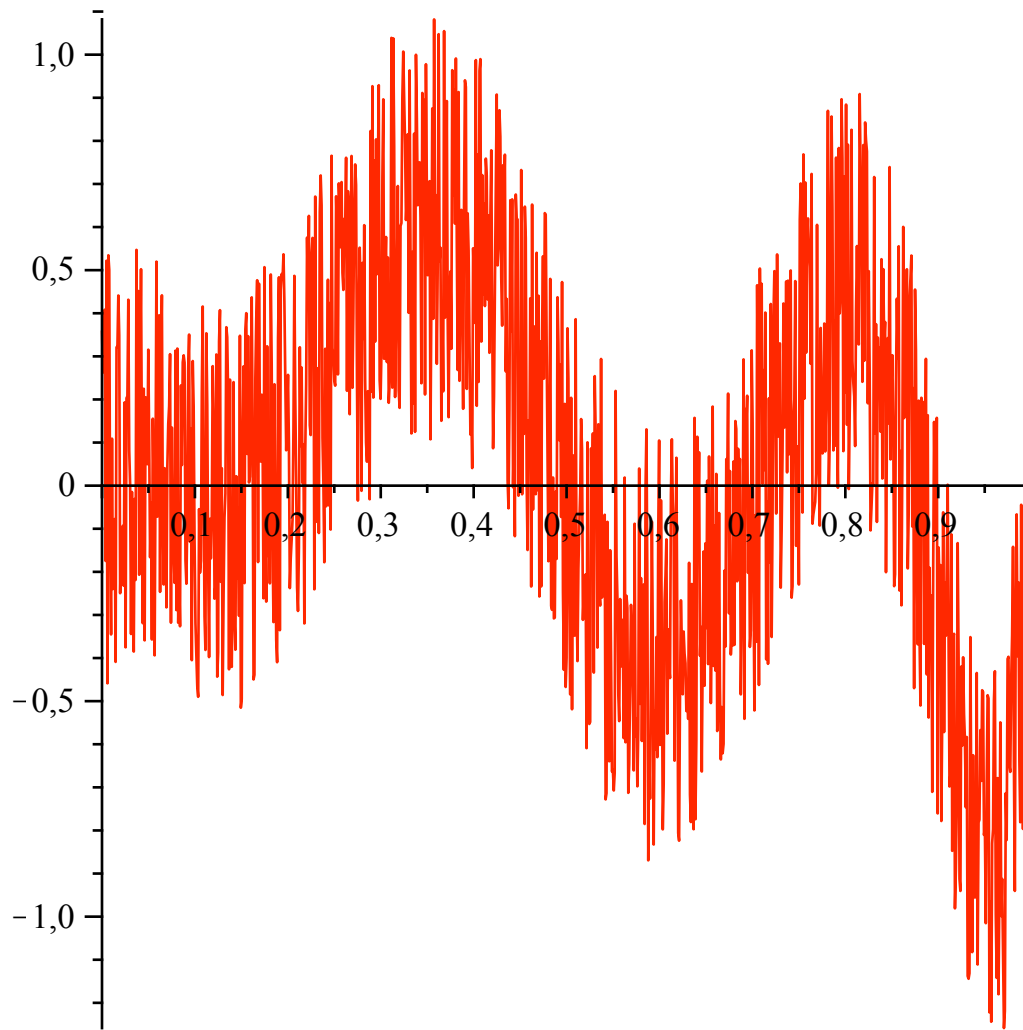
(3.1)



```
> t:=[evalf(($ 0..1023)/1024)]:  
> X:=[seq([t[i],evalf(1920*(t[i]-1/6)*(t[i]^2-1/2)*(t[i]-1/2)*  
  (t[i]-7/8)*(t[i]-1/10)*(t[i]-1)*t[i])],i=1..1024)]:  
> plot(X);
```

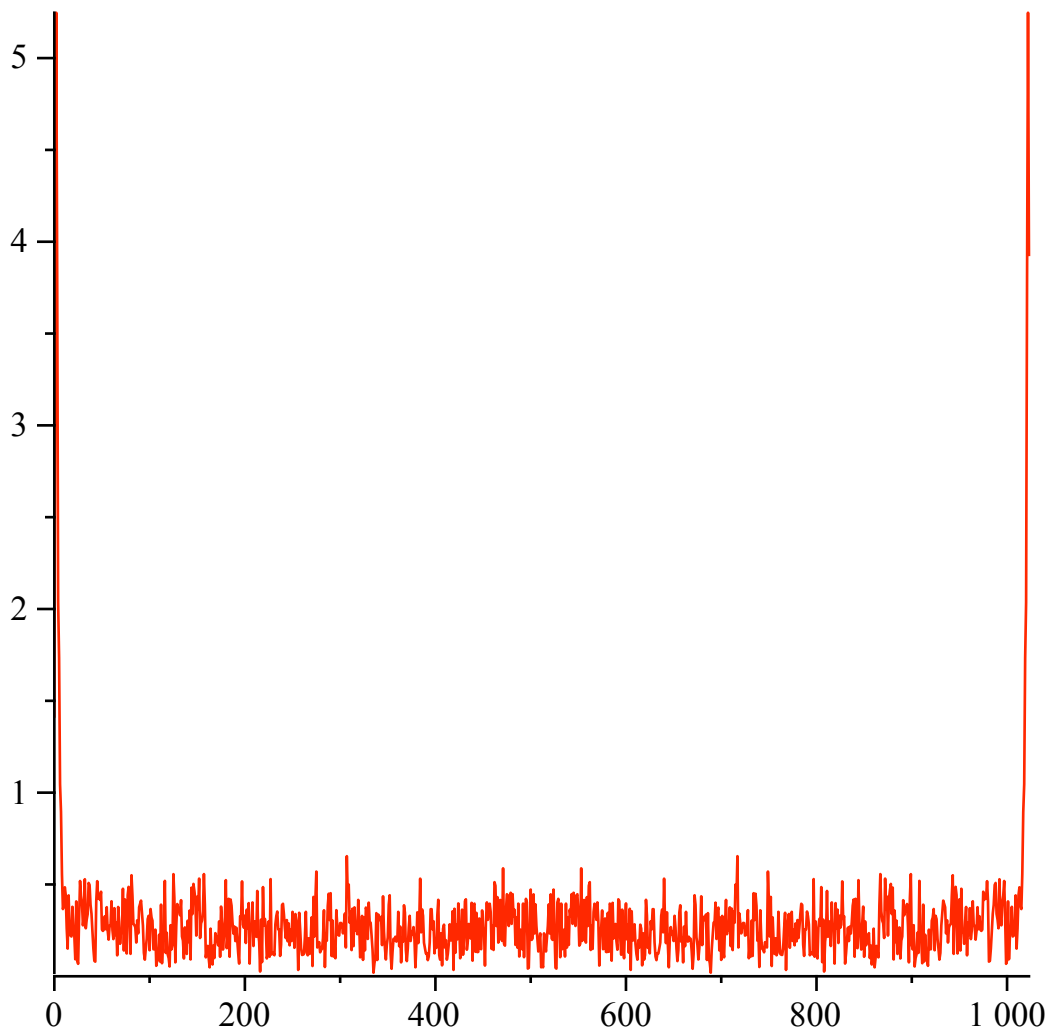
```
> plot(z);
```



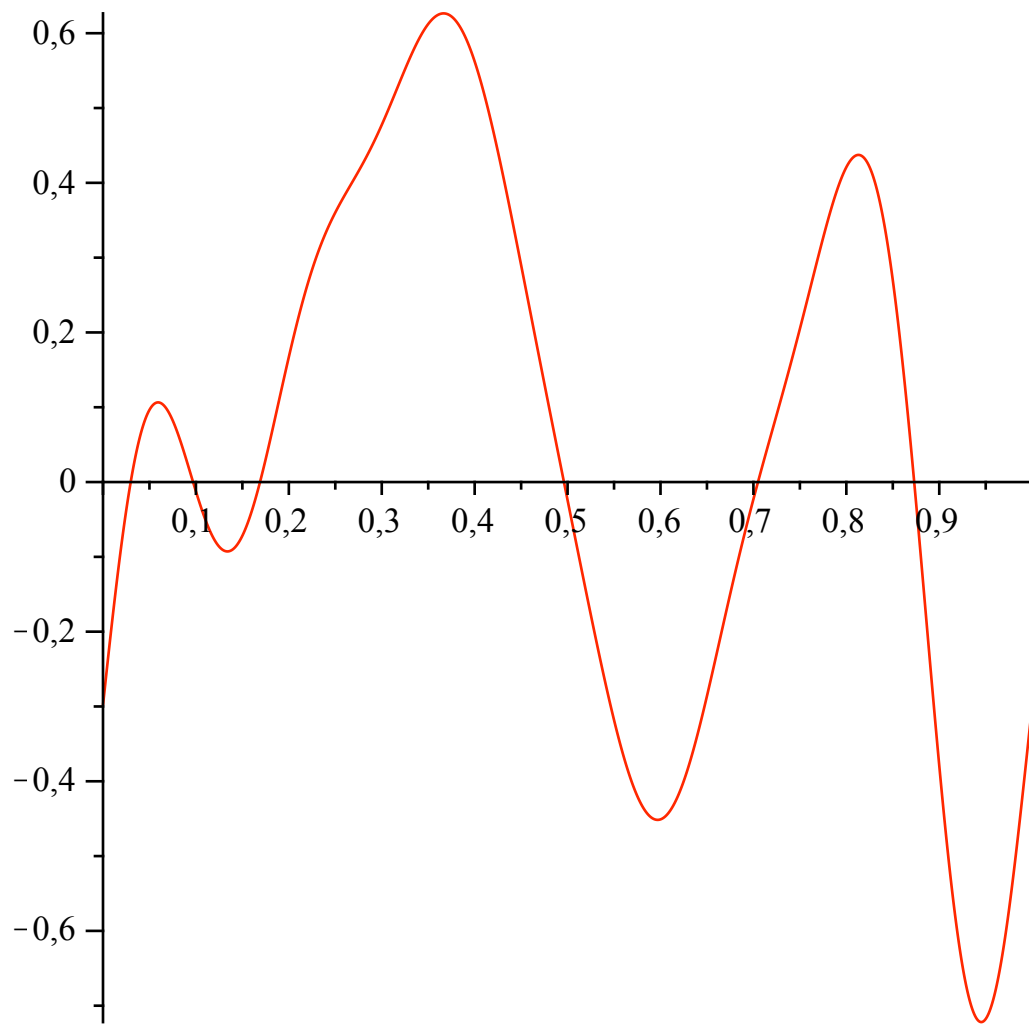
```
> ZZ:= [seq(Z[i][2], i=1..1024)]:
> with(DiscreteTransforms);
      [FourierTransform, InverseFourierTransform] (3.3)
```

```
> ZZ:=convert(ZZ, Vector):
> F:=FourierTransform(ZZ);
      F:= [ 1 .. 1024 Vectorcolumn
           Data Type: anything
           Storage: rectangular
           Order: Fortran_order ] (3.4)
```

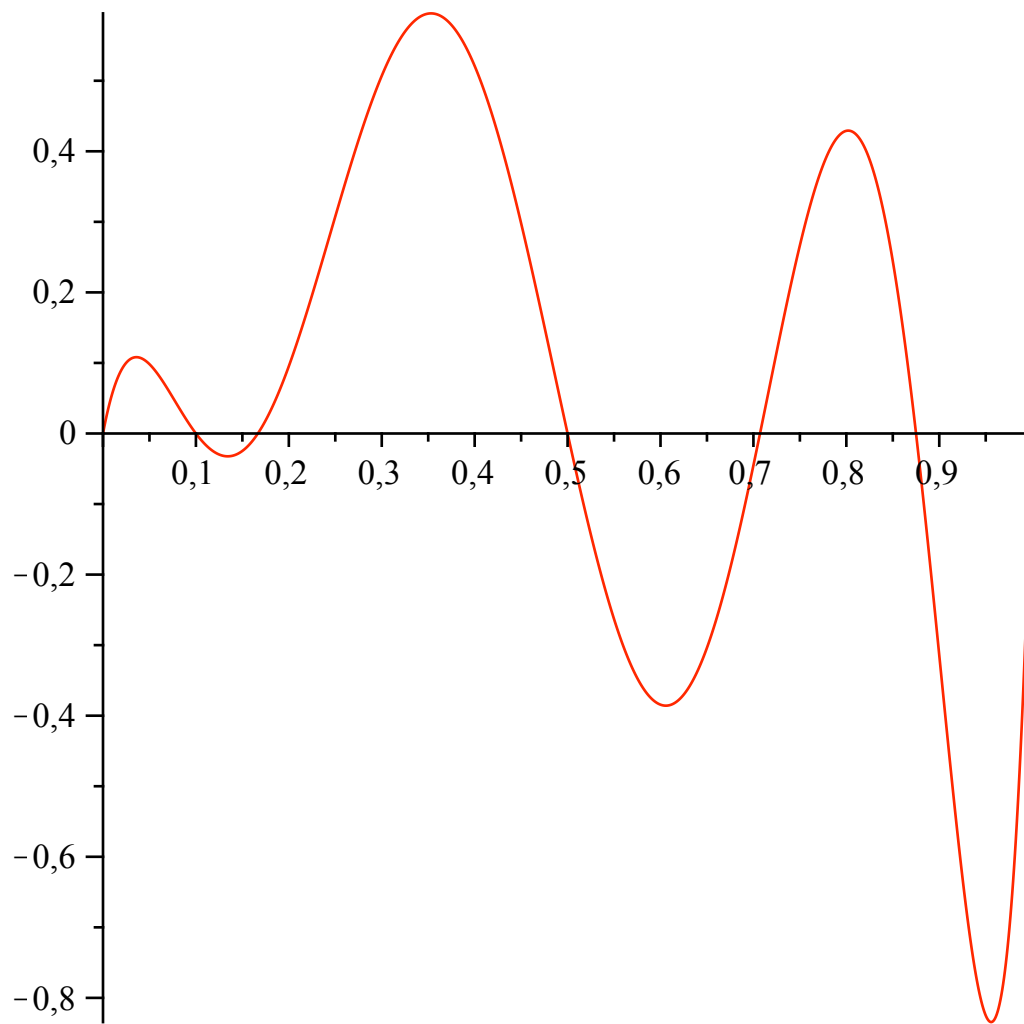
```
> FF:= [seq([i-1, abs(F[i])], i=1..1024)]:
> plot(FF);
```



```
> FS:=Vector(1024):  
> for i from 1 to 1024 do if abs(F[i])>1 then FS[i]:=F[i] else  
FS[i]=0 fi od;  
> S:=InverseFourierTransform(FS):  
> SS:=[seq([t[i],Re(S[i])],i=1..1024)]:  
> plot(SS);
```



```
> plot(x);
```

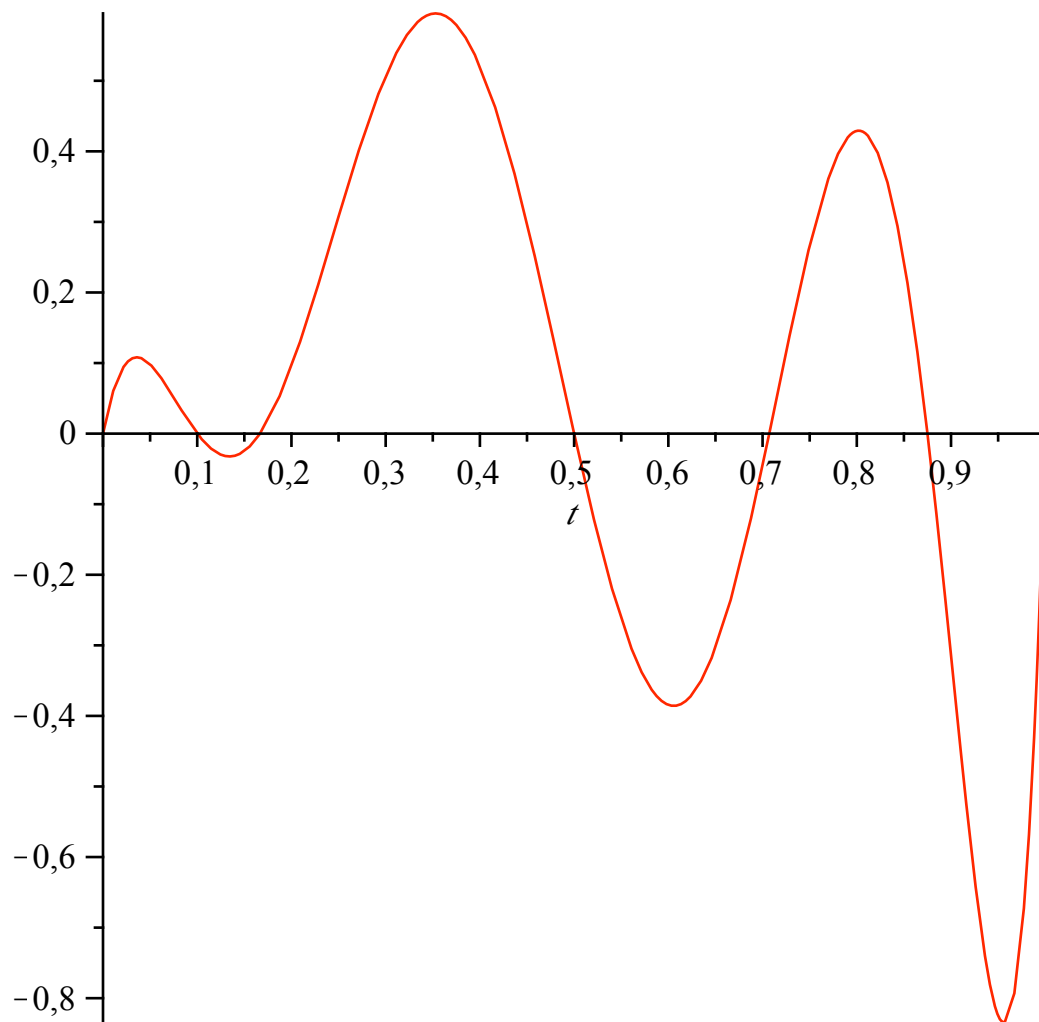


>

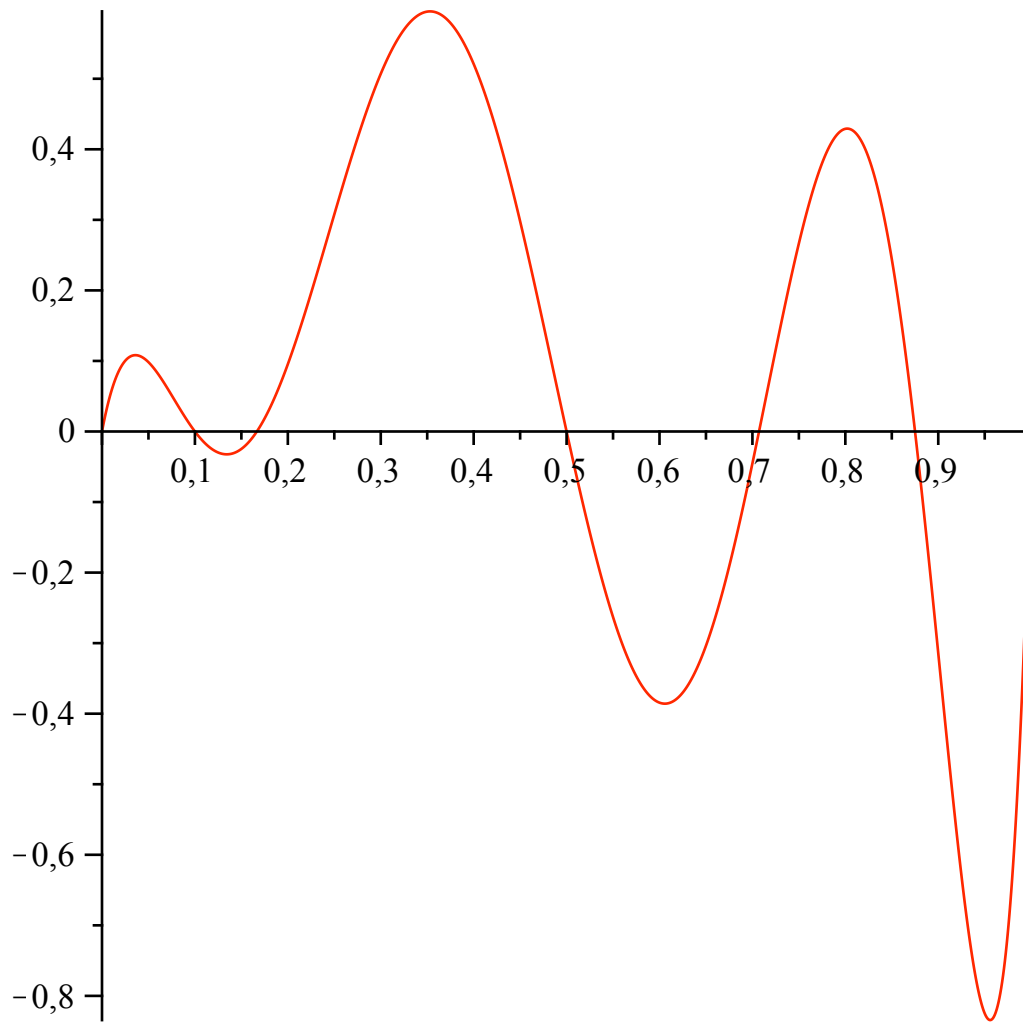
▼ Exemple 2 (autre méthode)

```
> restart;
> X:=expand(1920*(t-1/6)*(t^2-1/2)*(t-1/2)*(t-7/8)*(t-1/10)*
(t-1)*t);
      X:=1920 t8 - 5072 t7 + 3768 t6 + 692 t5 - 2082 t4 + 908 t3 - 141 t2 + 7 t
> plot(X,t=0..1023/1024);
```

(4.1)



```
> t:=[evalf(($ 0..1023)/1024)]:  
> X:=[seq([t[i],evalf(1920*(t[i]-1/6)*(t[i]^2-1/2)*(t[i]-1/2)*  
  (t[i]-7/8)*(t[i]-1/10)*(t[i]-1)*t[i])]),i=1..1024)]:  
> plot(X);
```

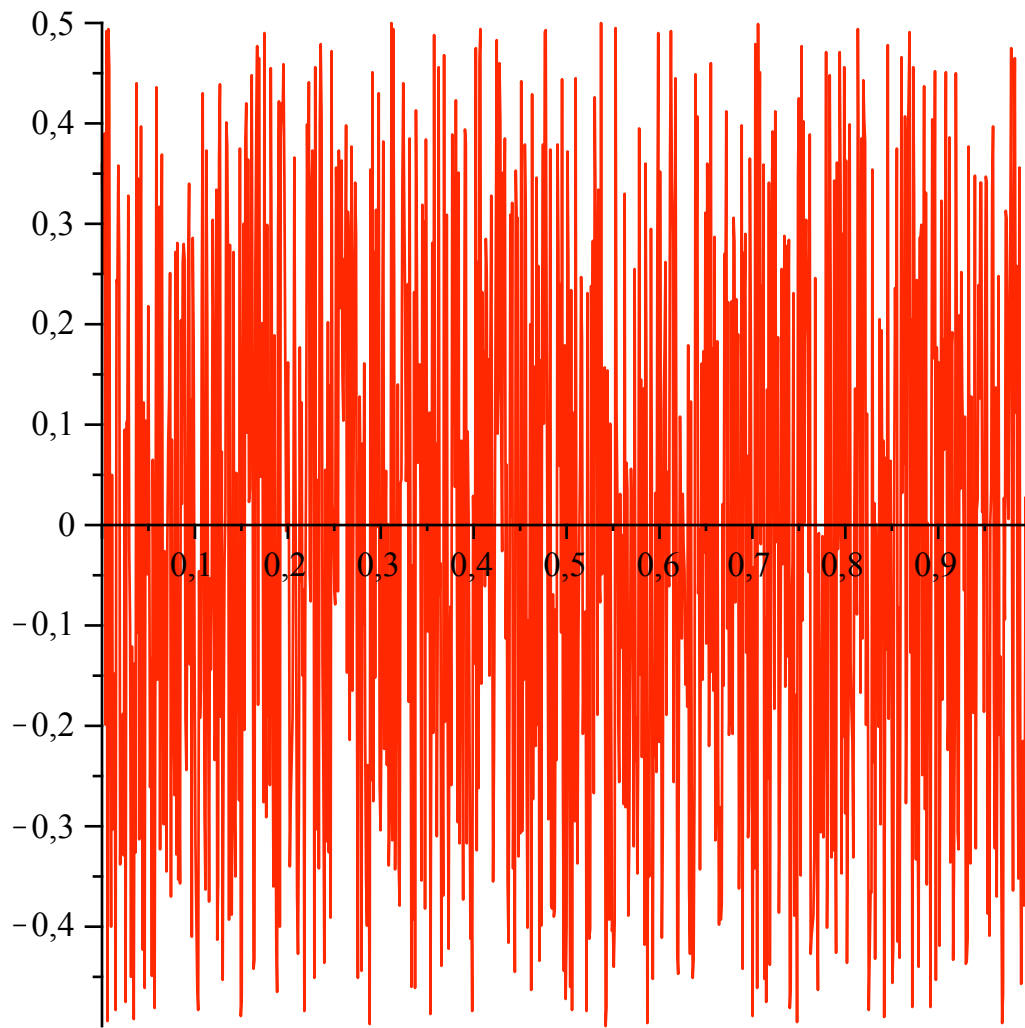


```

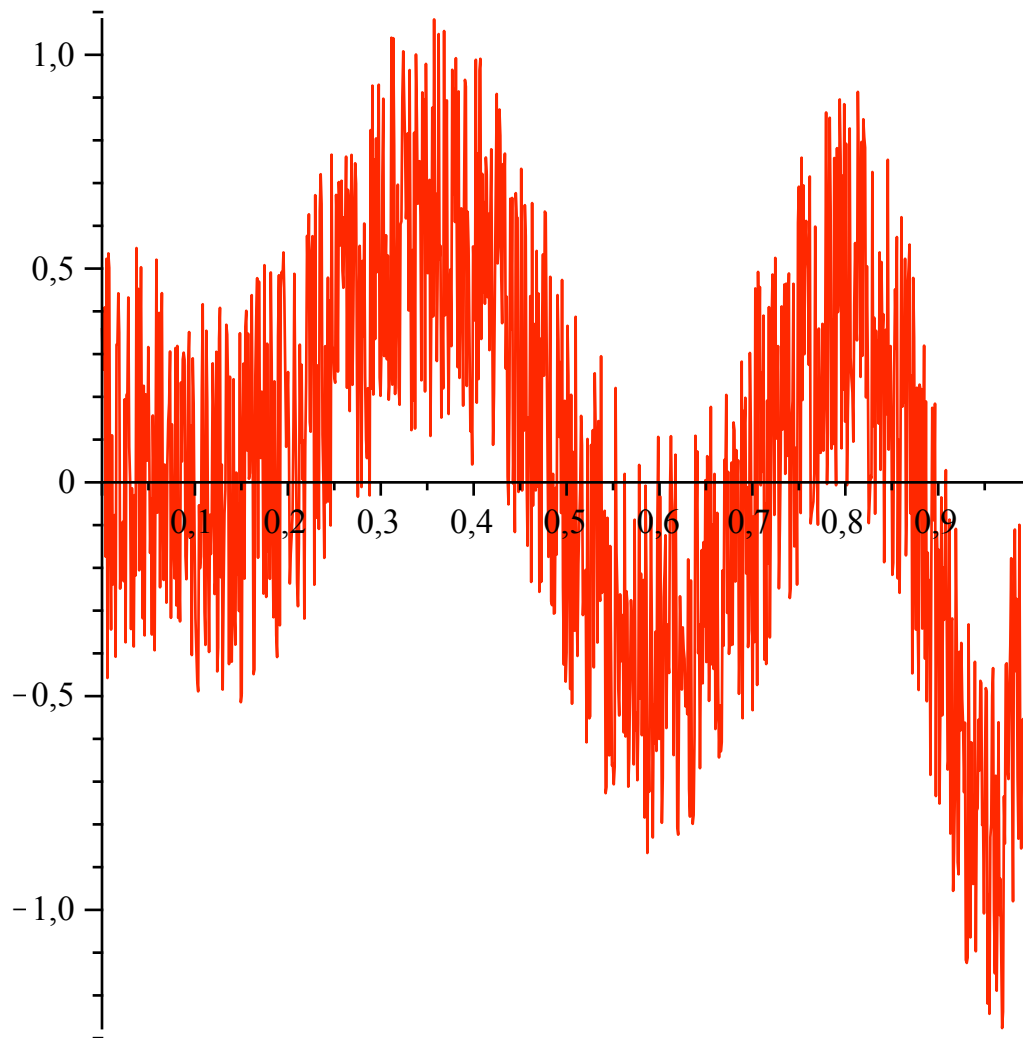
> h:=evalf(rand(-499..500)/1000);
h:=0.001000000000 proc( )
    proc( ) option builtin=RandNumberInterface, end proc(6, 1000, 10) - 499
end proc
> Y:=[seq([t[i],h()],i=1..1024)]:
> Z:=[seq([t[i],X[i][2]+Y[i][2]],i=1..1024)]:
> plot(Y);

```

(4.2)

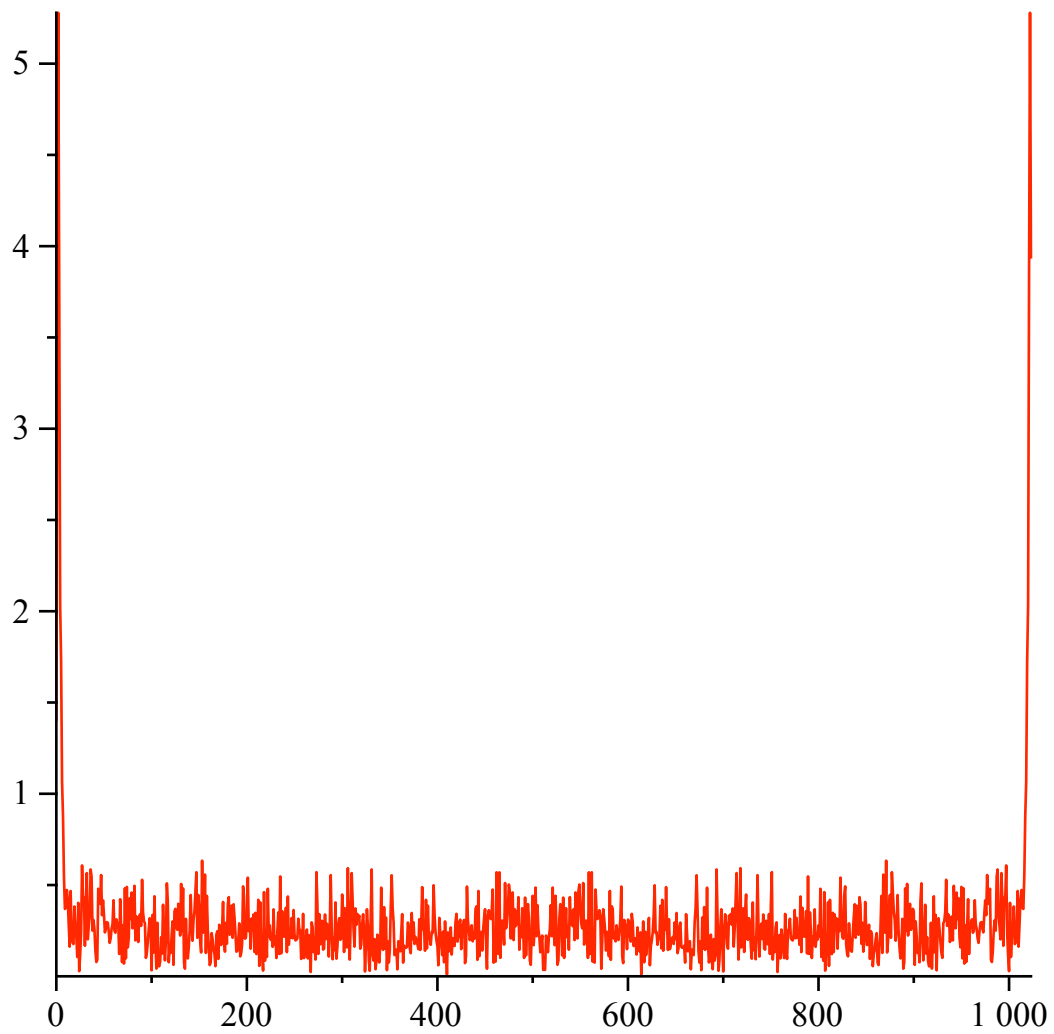


```
> plot(z);
```

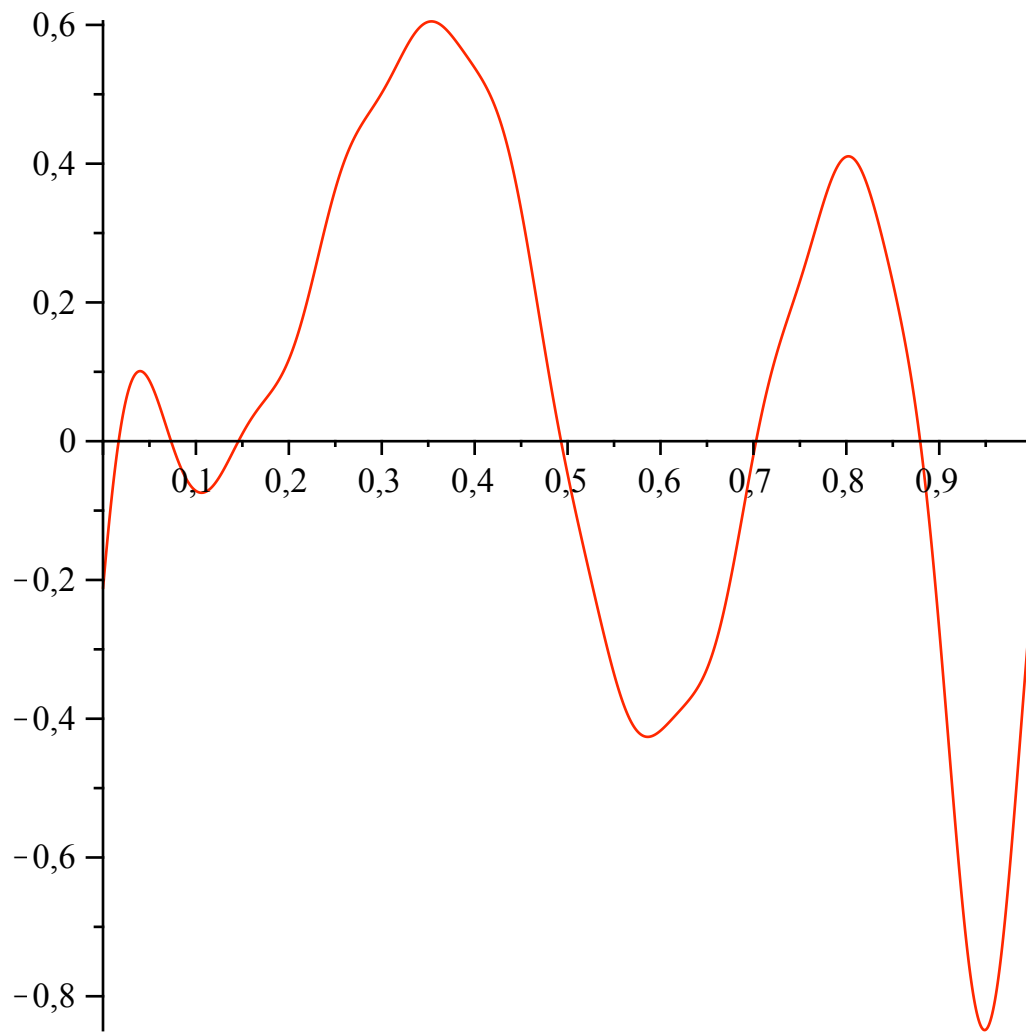


```
> ZZ:= [seq(Z[i][2], i=1..1024)]:  
> with(DiscreteTransforms);  
      [FourierTransform, InverseFourierTransform]  
> ZZ:=convert(ZZ, Vector):  
> F:=FourierTransform(ZZ):  
> FF:= [seq([i-1, abs(F[i])], i=1..1024)]:  
> plot(FF);
```

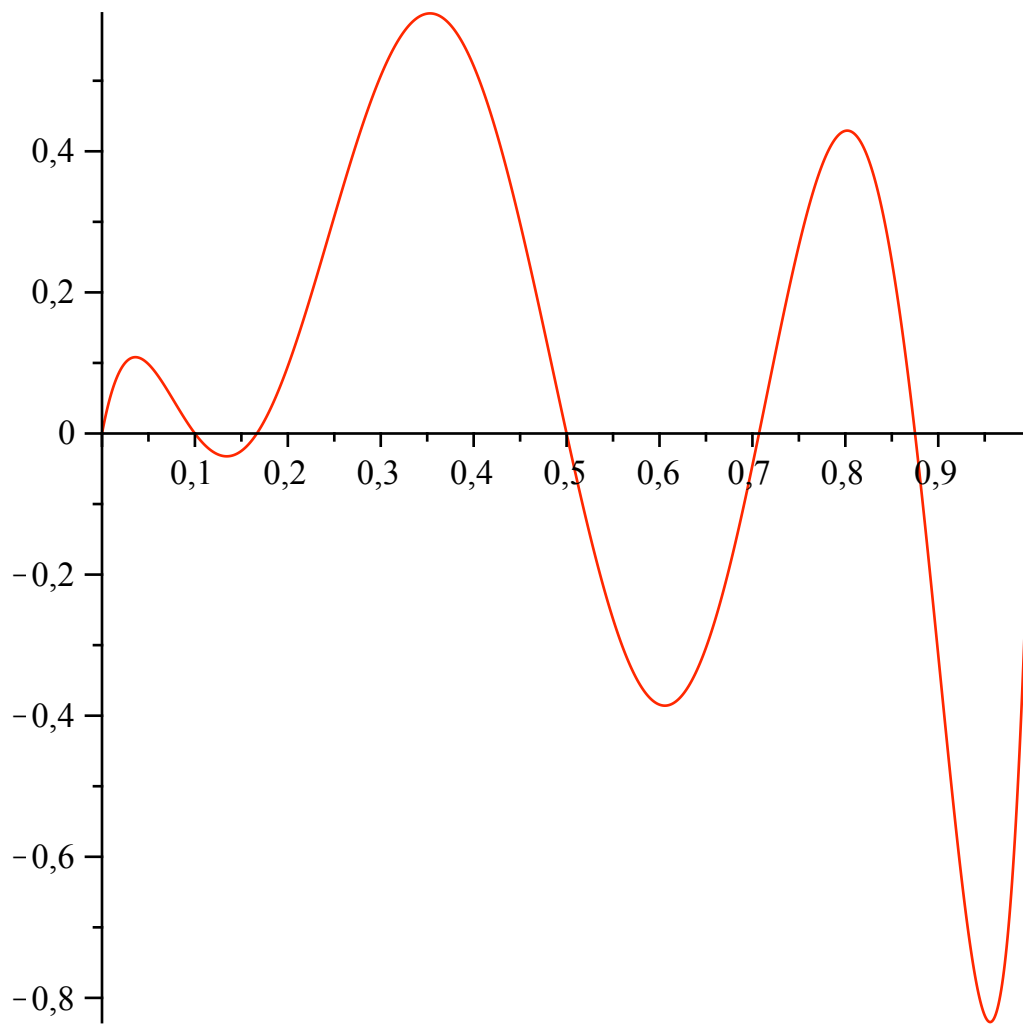
(4.3)



```
> FS:=Vector(1024):  
> for i from 1 to 10 do FS[i]:=F[i] od:  
> for i from 11 to 1014 do FS[i]:=0 od:  
> for i from 1014 to 1024 do FS[i]:=F[i] od:  
> S:=InverseFourierTransform(FS):  
> SS:=[seq([t[i],Re(S[i])],i=1..1024)]:  
> plot(SS);
```



```
> plot(x);
```

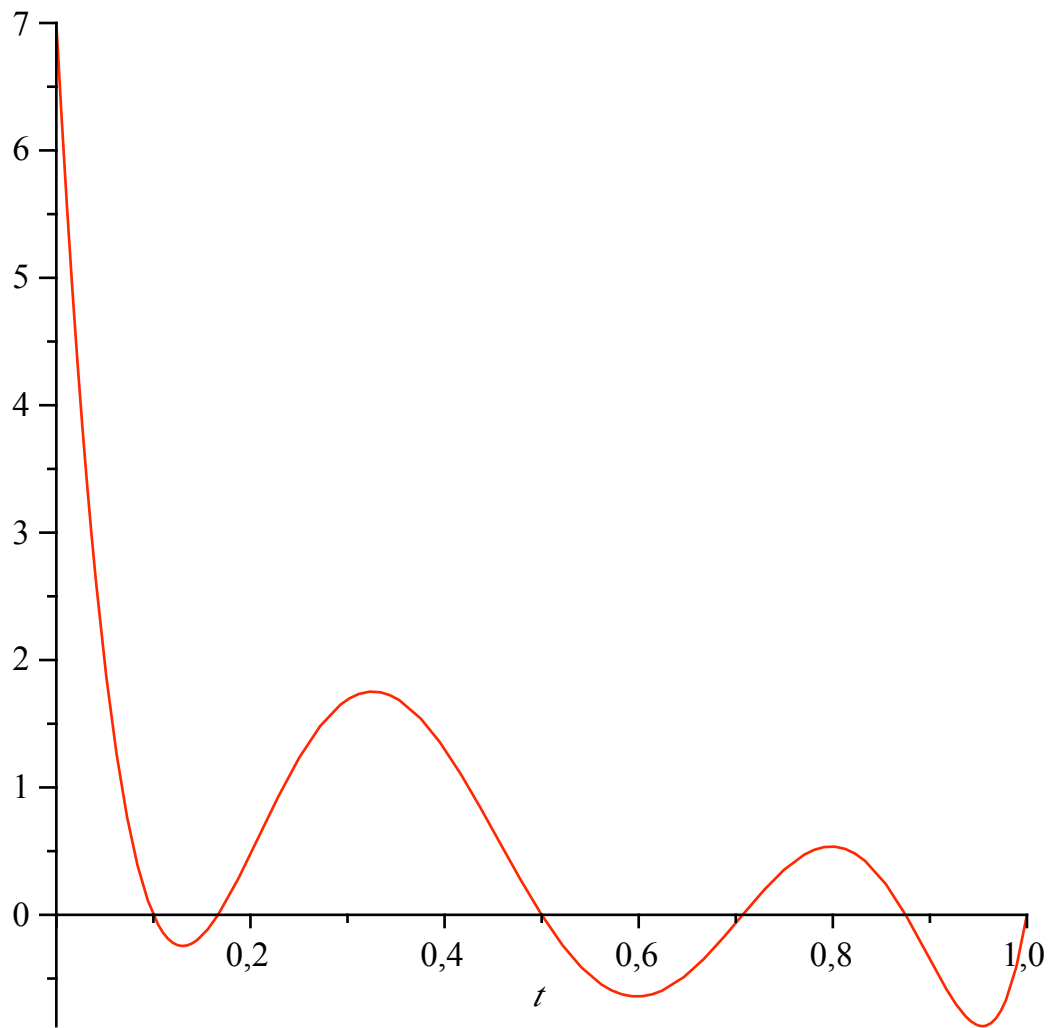


>

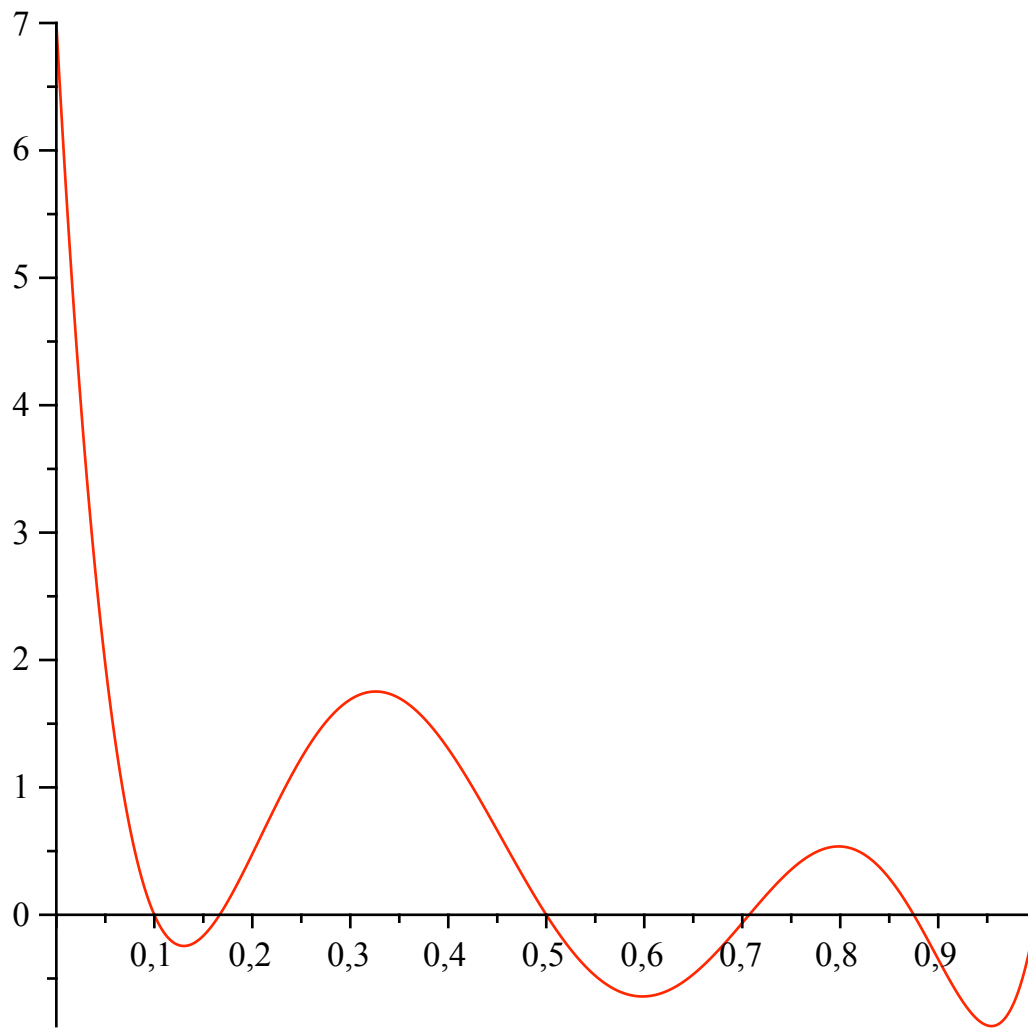
▼ Exemple 3 (polynôme, avec "effet de bord")

```
> restart;
> X:=expand(1920*(t-1/6)*(t^2-1/2)*(t-1/2)*(t-7/8)*(t-1/10)*
(t-1));
      X:=1920 t7 - 5072 t6 + 3768 t5 + 692 t4 - 2082 t3 + 908 t2 - 141 t + 7
> plot(X,t=0..1);
```

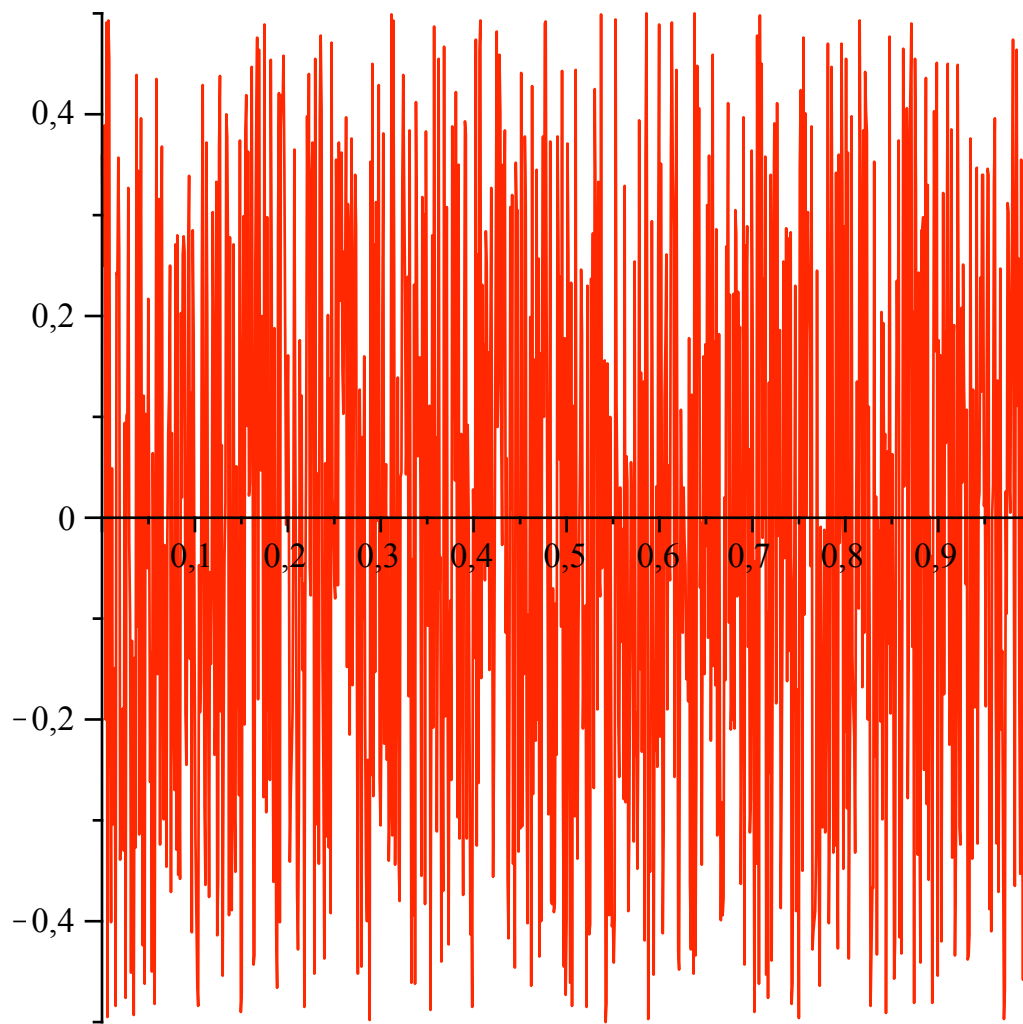
(5.1)



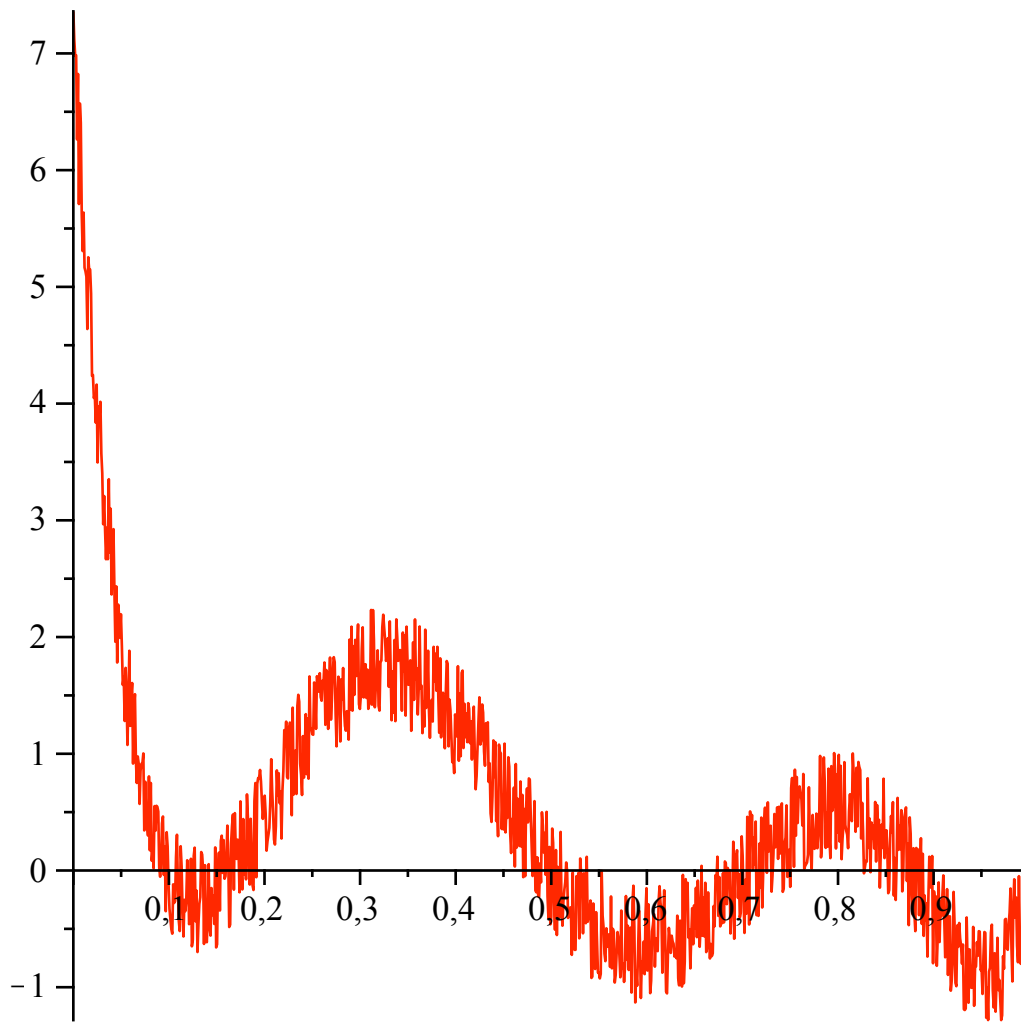
```
> t:=[evalf(($ 0..1023)/1024)]:  
> X:=[seq([t[i],evalf(1920*(t[i]-1/6)*(t[i]^2-1/2)*(t[i]-1/2)*  
  (t[i]-7/8)*(t[i]-1/10)*(t[i]-1))],i=1..1024)]:  
> plot(X);
```



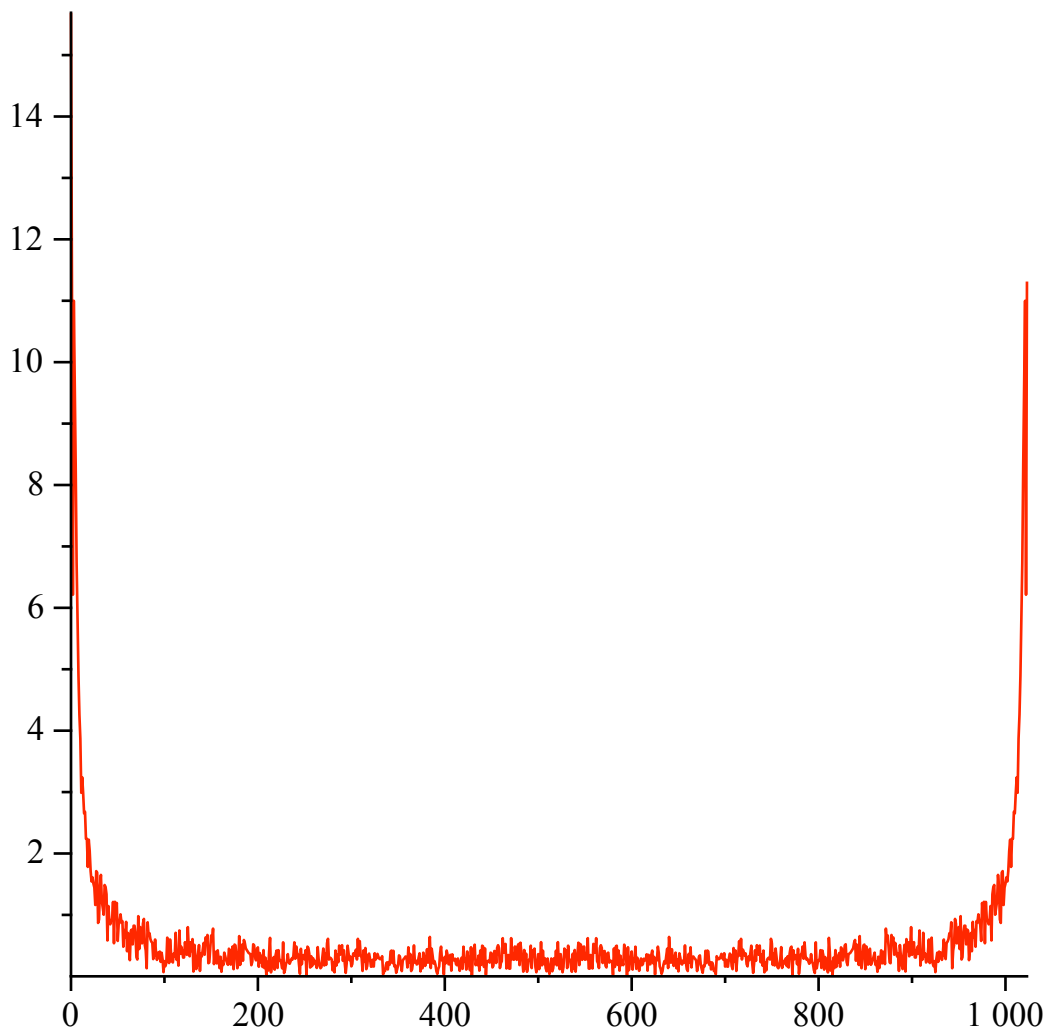
```
> h:=evalf(rand(-500..500)/1000):  
> Y:=[seq([t[i],h()],i=1..1024)]:  
> Z:=[seq([t[i],X[i][2]+Y[i][2]],i=1..1024)]:  
> plot(Y);
```



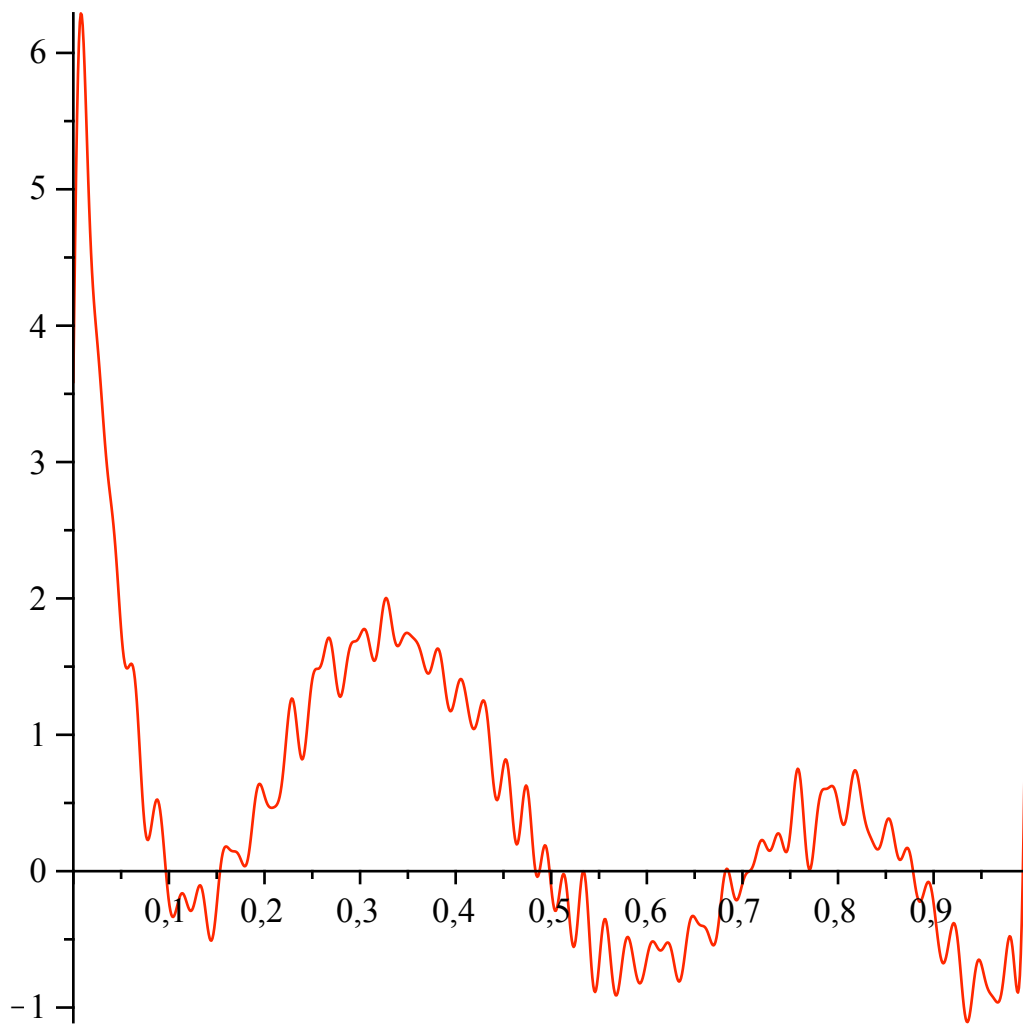
```
> plot(z);
```



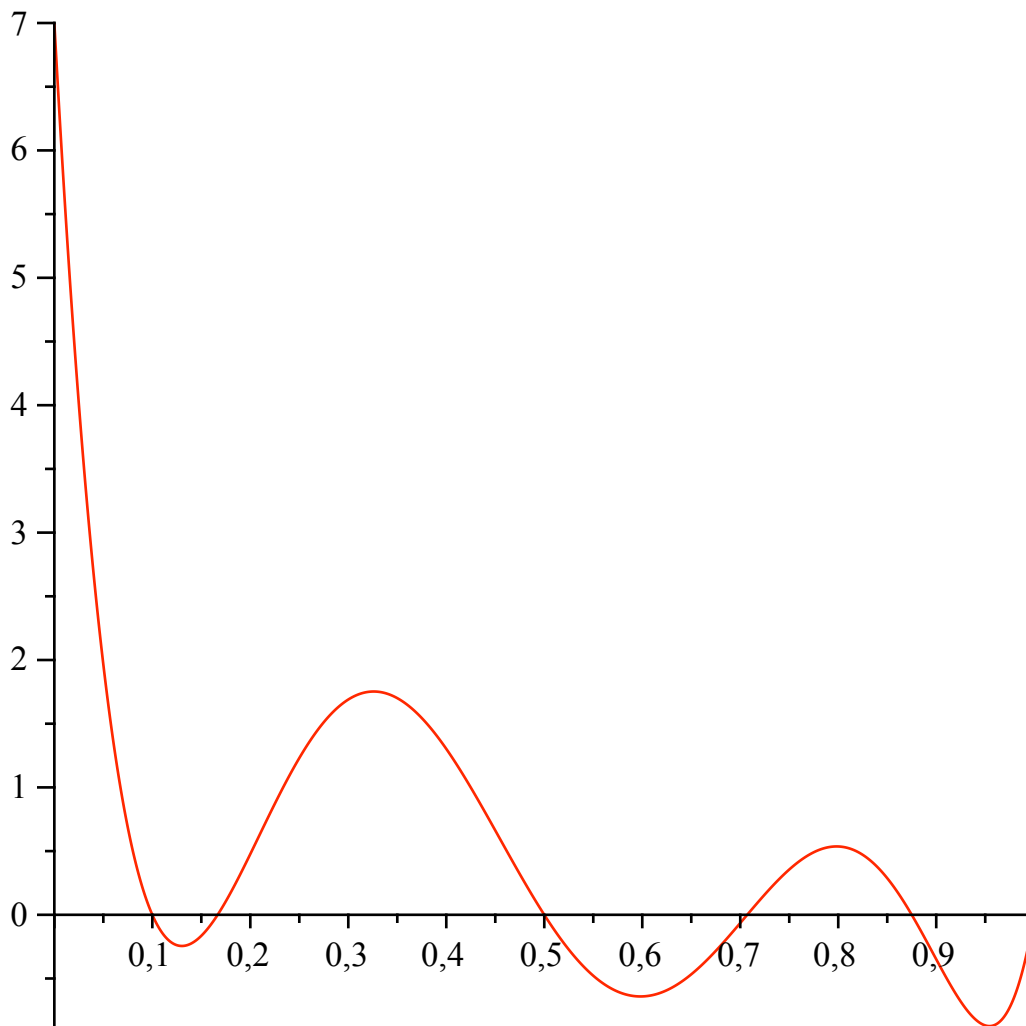
```
> ZZ:= [seq(Z[i][2], i=1..1024)]:  
> with(DiscreteTransforms):  
> ZZ:=convert(ZZ, Vector):  
> F:=FourierTransform(ZZ):  
> FF:= [seq([i-1, abs(F[i])], i=1..1024)]:  
> plot(FF);
```



```
> FS:=Vector(1024):  
> for i from 1 to 1024 do if abs(F[i])>1 then FS[i]:=F[i] else  
FS[i]=0 fi od;  
> S:=InverseFourierTransform(FS):  
> SS:= [seq([t[i],Re(S[i])],i=1..1024)]:  
> plot(SS);
```



```
> plot(x);
```



>

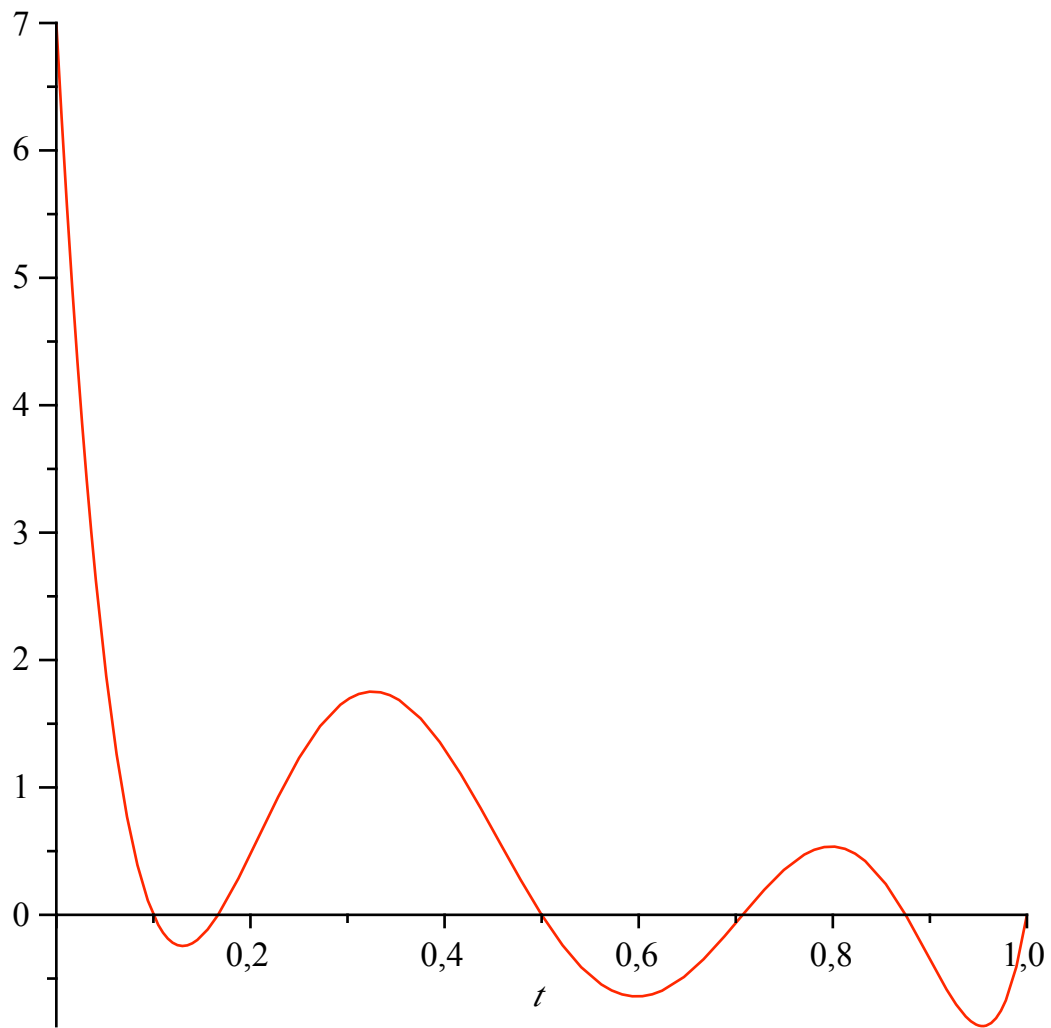
▼ Exemple 3 (autre méthode)

```

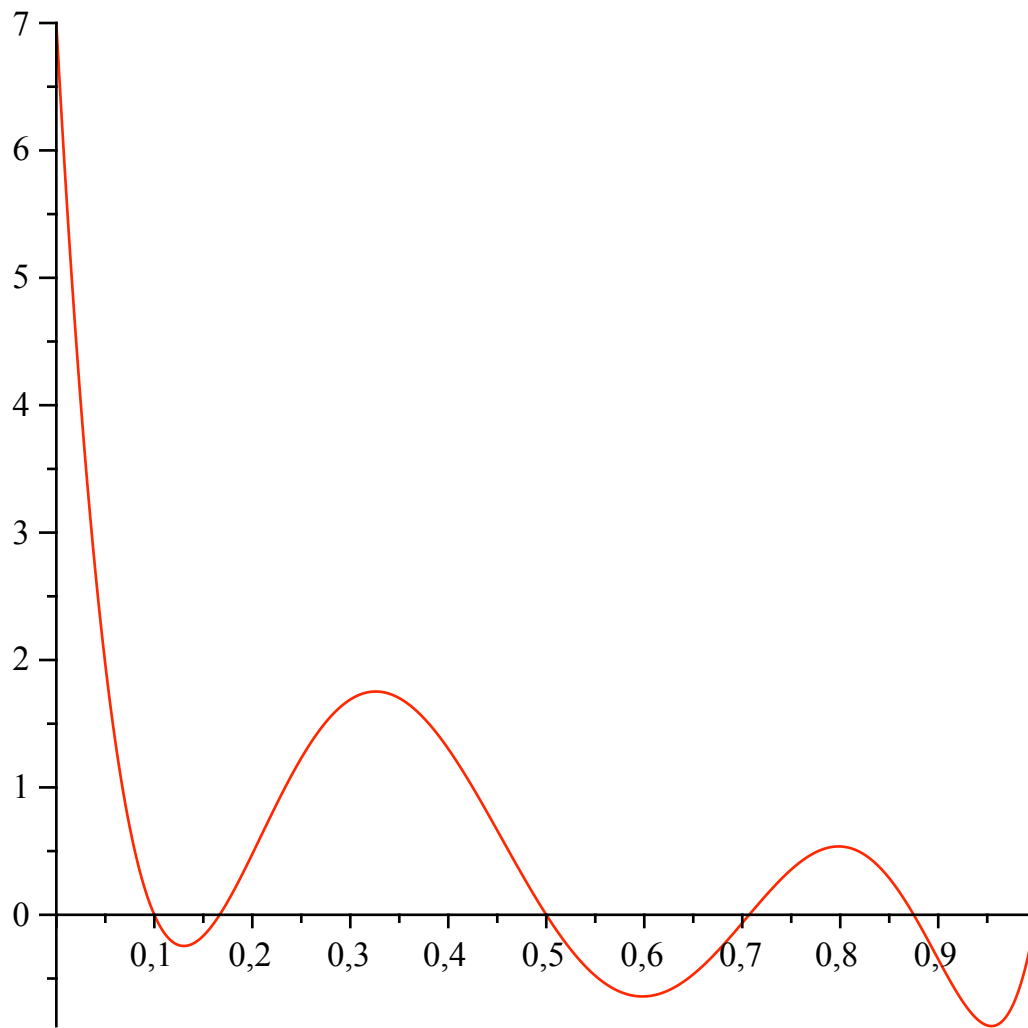
> restart;
> X:=expand(1920*(t-1/6)*(t^2-1/2)*(t-1/2)*(t-7/8)*(t-1/10)*
(t-1));
      X:= 1920 t7 - 5072 t6 + 3768 t5 + 692 t4 - 2082 t3 + 908 t2 - 141 t + 7
> plot(X,t=0..1);

```

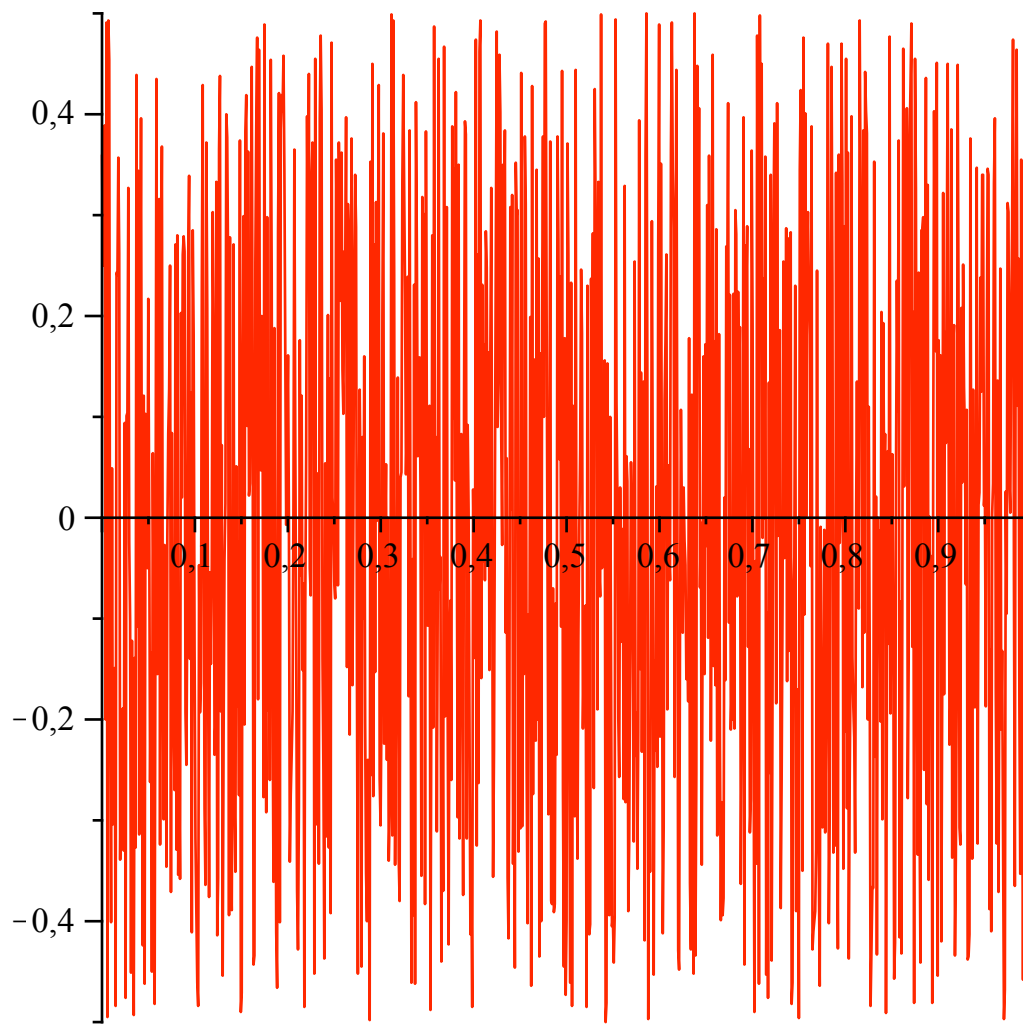
(6.1)



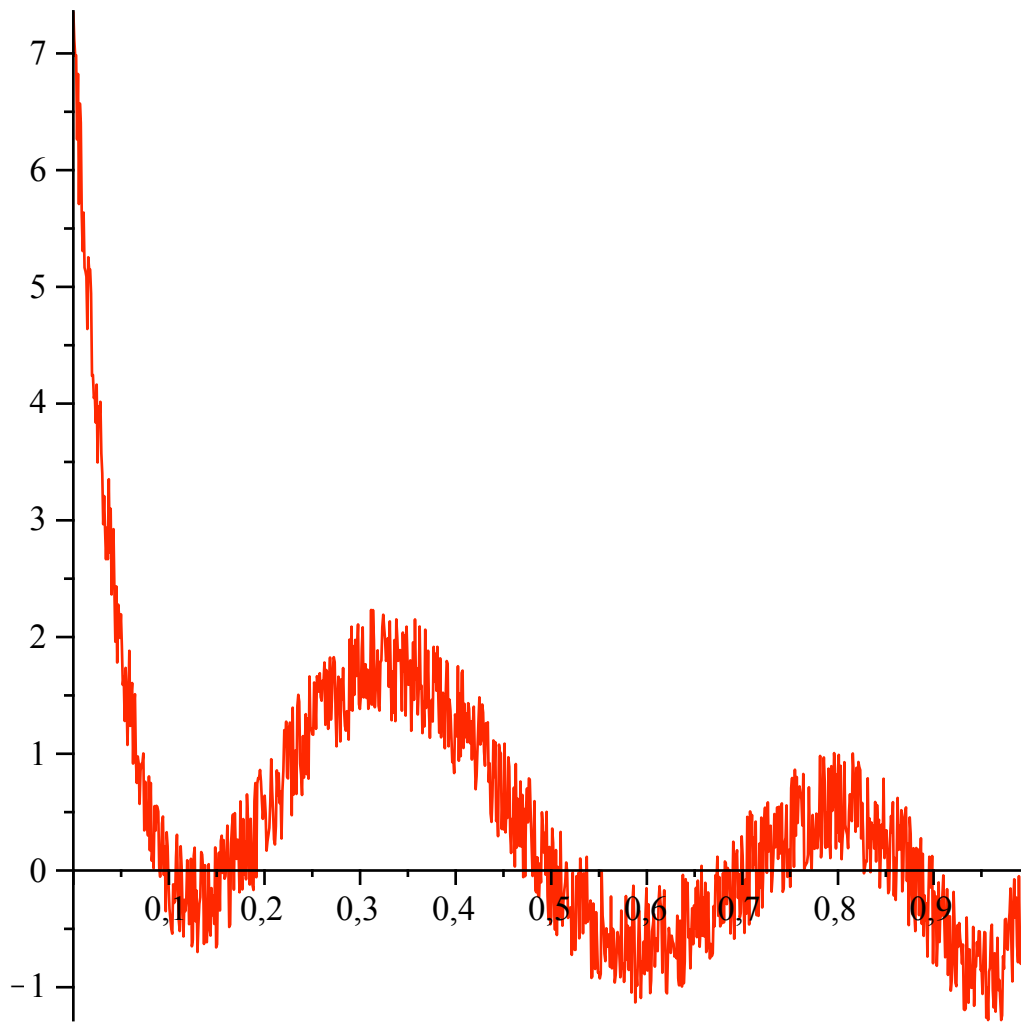
```
> t:=[evalf(($ 0..1023)/1024)]:  
> X:[seq([t[i],evalf(1920*(t[i]-1/6)*(t[i]^2-1/2)*(t[i]-1/2)*  
  (t[i]-7/8)*(t[i]-1/10)*(t[i]-1))],i=1..1024)]:  
> plot(X);
```



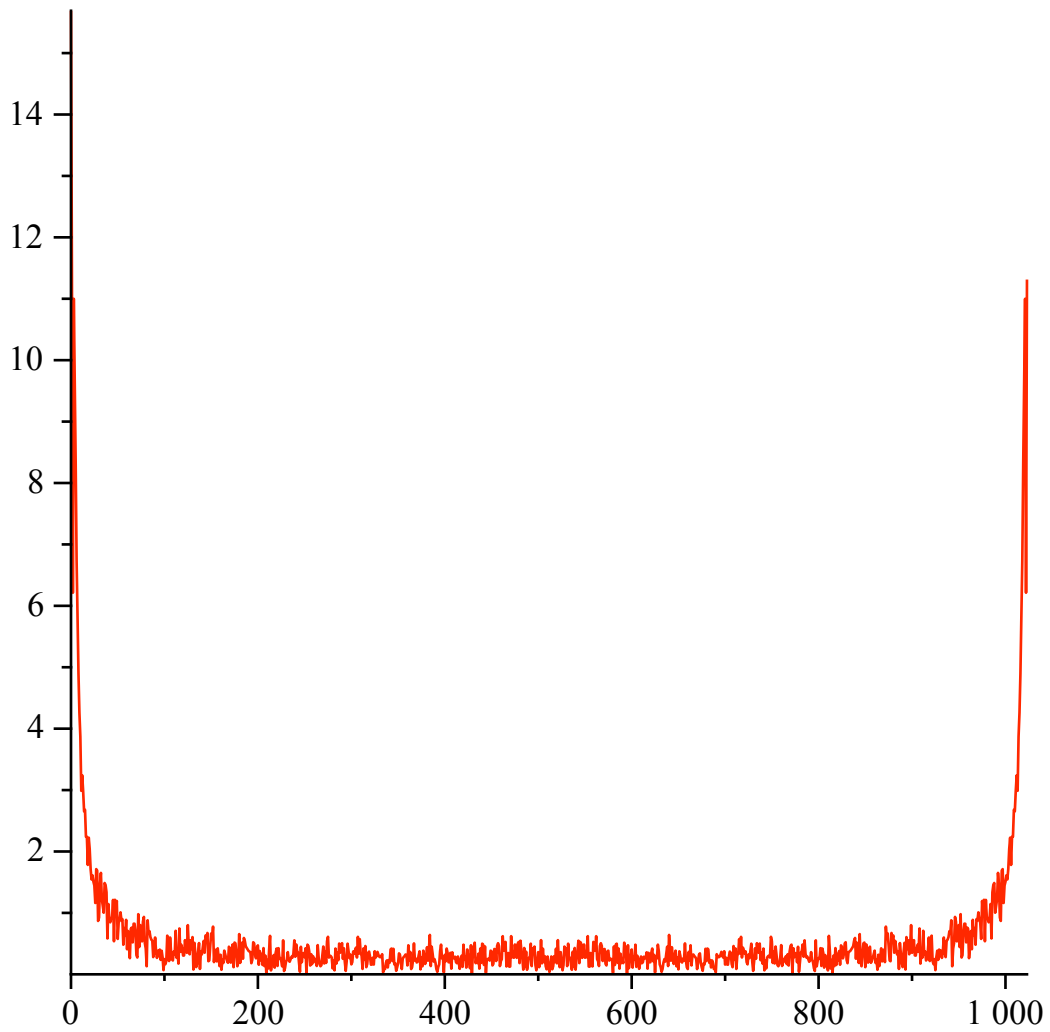
```
> h:=evalf(rand(-500..500)/1000):  
> Y:=[seq([t[i],h()],i=1..1024)]:  
> Z:=[seq([t[i],X[i][2]+Y[i][2]],i=1..1024)]:  
> plot(Y);
```



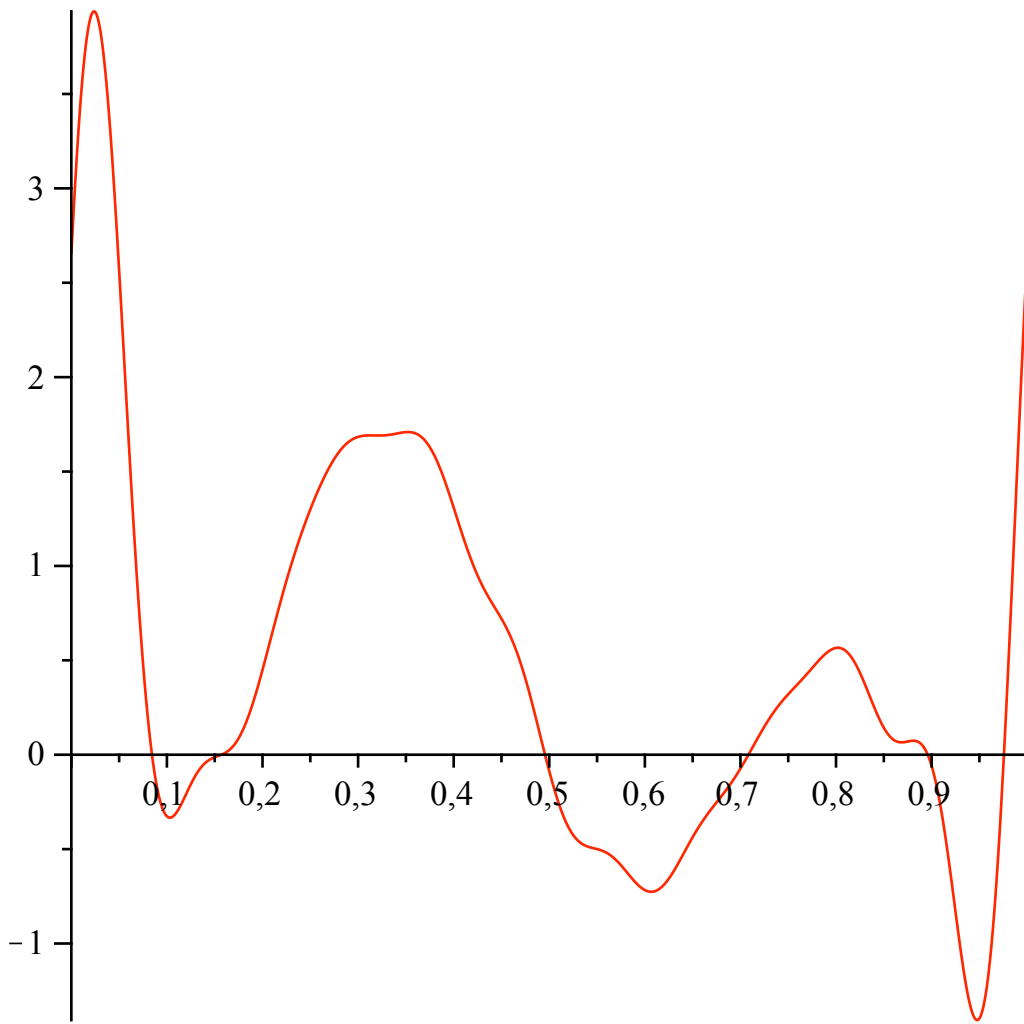
```
> plot(z);
```



```
> ZZ:= [seq(Z[i][2], i=1..1024)]:  
> with(DiscreteTransforms):  
> ZZ:=convert(ZZ, Vector):  
> F:=FourierTransform(ZZ):  
> FF:= [seq([i-1, abs(F[i])], i=1..1024)]:  
> plot(FF);
```



```
> FS:=Vector(1024):  
> for i from 1 to 10 do FS[i]:=F[i] od:  
> for i from 11 to 1014 do FS[i]:=0 od:  
> for i from 1014 to 1024 do FS[i]:=F[i] od:  
> S:=InverseFourierTransform(FS):  
> SS:=[seq([t[i],Re(S[i])],i=1..1024)]:  
> plot(SS);
```



```
> plot(x);
```

