Programmation de la FFT

- **En Maple**

```
> restart;interface(echo=3);
                                 1
> with(LinearAlgebra):
> read "AGREG/PROGRAMMES_MAPLE/myfft.mw":
> myfft:=proc(x)
#NB : charger LinearAlgebra
> local N,xI,xII,hxi,hxp,hx,PM,d,M,j;
> N:=Dimension(x);
> if N= 1 then
>    hx:=x;
> else
>    M:=N/2;
>    xI:=Vector([seq(x[i],i=1..M)]);
>    xII:=Vector([seq(x[M+i],i=1..M)]);
>    d:=Vector([seq(exp(-I*Pi*j/M),j=0..M-1)]);
>    PM:=DiagonalMatrix(d);
>    hxp:=myfft(xI+xII);
>    hxi:=myfft(PM.(xI-xII));
>    hx:=Vector(N);
>    for j from 1 to M do
>       hx[2*j-1]:=hxp[j];
>       hx[2*j]:=hxi[j];
>    od;
#   hx:=x;
> end if;
> return(hx);
> end proc;
> x:=Vector([0,2,3,4]);
                                  [0]
                                  [ ]
                                  [2]
                           x := [ ]
                                  [3]
                                  [ ]
                                  [4]
> hx:=myfft(x);
                                [   9    ]
                                [        ]
                                [-3 + 2 I]
                         hx := [        ]
                                [   -3   ]
                                [        ]
```

```
                                [-3 - 2 I]
> with (DiscreteTransforms);
          [FourierTransform, InverseFourierTransform]
> FourierTransform(x)*sqrt(Dimension(x));
                            [9. + 0. I ]
                            [          ]
                            [-3. + 2. I]
                            [          ]
                            [-3. + 0. I]
                            [          ]
                            [-3. - 2. I]
```

• **En Scilab**

NB : toutes les lignes commencant par "disp" ci-dessous ont ete mise en commentaire car elles sont facultatives. Mais elles ont servi à debugger le programme (notamment à voir les tailles des vecteurs et matrices)

```
function hx=myFFT(x)
//transformee de Fourier rapide pour N=2^m
//x est un vecteur colonne, et hx est un vecteur colonne de meme taille
N=length(x);
  if N==1 then
     hx=x;
  else
     hx=zeros(x);//NB : x est un vecteur colonne
     M=N/2;
     d=zeros(M,1);
     for j=1:M
         d(j)=exp(-(j-1)*%i*%pi/M);
     end
     PM=diag(d);
     //disp('PM=')
     //disp(PM)
     xI=x(1:M);
     xII=x(M+1:2*M);
     hxp=myFFT(xI+xII);
     //disp('xI=')
     //disp(xI)
     //disp('xII=')
     //disp(xII)
     hxi=myFFT(PM*(xI-xII));
     for k=0:M-1
         hx(2*k+1)=hxp(k+1);
         hx(2*k+2)=hxi(k+1);
     end
  end

endfunction
```

Résultat d'une exécution

```
-1->x=[1 2 3 4]'
 x  =

    1.
    2.
    3.
    4.

-1->hx=FFourierT(x)

 PM=

    1.    0
    0     6.123E-17 - i

 PM=

    1.

 xI=

    4.

 xII=

    6.

 xI=

    1.
    2.

 xII=

    3.
    4.

 PM=

    1.

 xI=

  - 2.

 xII=

  - 1.225E-16 + 2.i
 hx  =
```

```
    10.
  - 2. + 2.i
  - 2.
  - 2. - 2.i

-1->hx-fft(x,-1)
 ans  =

    0
    0
    0
    2.220E-16

-1->
```

Un programme comme celui-ci permet de voir les différentes possibilités en taille mémoire des programmes.

Pour `dft`, on peut aller jusqu'à 2^{10} inclus : ensuite, pour 2^{11}, problème de taille mémoire. On met la ligne `dft` en commentaire pour continuer à augmenter N.

Pour `myFFT`, 2^{11} inclus.

Pour `fft`, jusqu'à 2^{20} inclus. Pour $N = 2^{20}$, le temps d'exécution reste petit (moins d'une seconde). Pour $N = 2^{10}$, `myFFT` est exécuté en 0.13 s et `dft` en 0.228 s.

On remarque que `fft` marche également avec des nombres quelconques. Pour un nombre premier grand, l'exécution est plus longue que pour un nombre proche de la forme 2^k.

```
//temps d'execution et taille maximale de diverses FFT

//taille maximale
exec('myFFT.sci');
N=2^18 //N=puissance de 2
x=zeros(N,1);
for i=0:N-1
     x(i+1)= i;
end
'myFFT'
tic()
//hx=myFFT(x);
toc()
'fft'
tic()
fft(x);
toc()
'dft'
tic()
//dft(x,-1);
toc()
```

```
exec('Crible.sci') ;//Crible(N)=tableau des nombres premiers <=N
N=max(Crible(2^15))
x=zeros(N,1);
for i=0:N-1
     x(i+1)= i;
end
'fft premier'
tic()
fft(x);
toc()
```