Signal Processing - Report

Lab 2 – Group 2

OLIVIER JAIS-NIELSEN

25 novembre 2008

Signal Processing - Report

Lab 2 – Group 2

Z-Transform

1. Implementation of the z-transform

```
function Z = ztrans(x, z)
  Z = polyval(x(end:-1:1), 1./z);
```

The built-in function polyval does almost what we need : the coefficient just have to be reversed. The "polynomial" has to be applied to 1/z

2. Implementation of DFT using ztrans

```
dft =
@(x)ztrans(x,exp(2*pi*i.*[0:numel(x)-
1]/numel(x)))
```

We use the fact that the k-ieth value of the DFT of x is equal to the z-transform of x with $z = e^{2i\pi(k-1)/N}$.

To compare the speed of this function and of the built-in fft, I measured the time taken by each of these implementations to compute the DFT of a unit sample signal, a unit step signal and a unit ramp signal for various length (from 1 to 20). For a better precision, each duration is actually the average duration of 1000 calculations.

```
N=20;
I=1000;
```

We work on signals of length going from 1 to 20. Each calculation si made 1000 times

```
timeFFT = zeros(3,N);
timeDFT = zeros(3,N);
```

We store the durations in arrays.

```
tic
```

We start the timer

```
for n = 1:N
```

For signal of length n :

```
  unitSample = [1];
  unitStep = ones(n);
  unitRamp = [0:1:n-1];
```

We define our three signals

```
  t0=toc;
  for k = 1:I
    dft(unitSample);
  end
```

We measure the duration of computing 1000 times the dft of the first signal using our implementation.

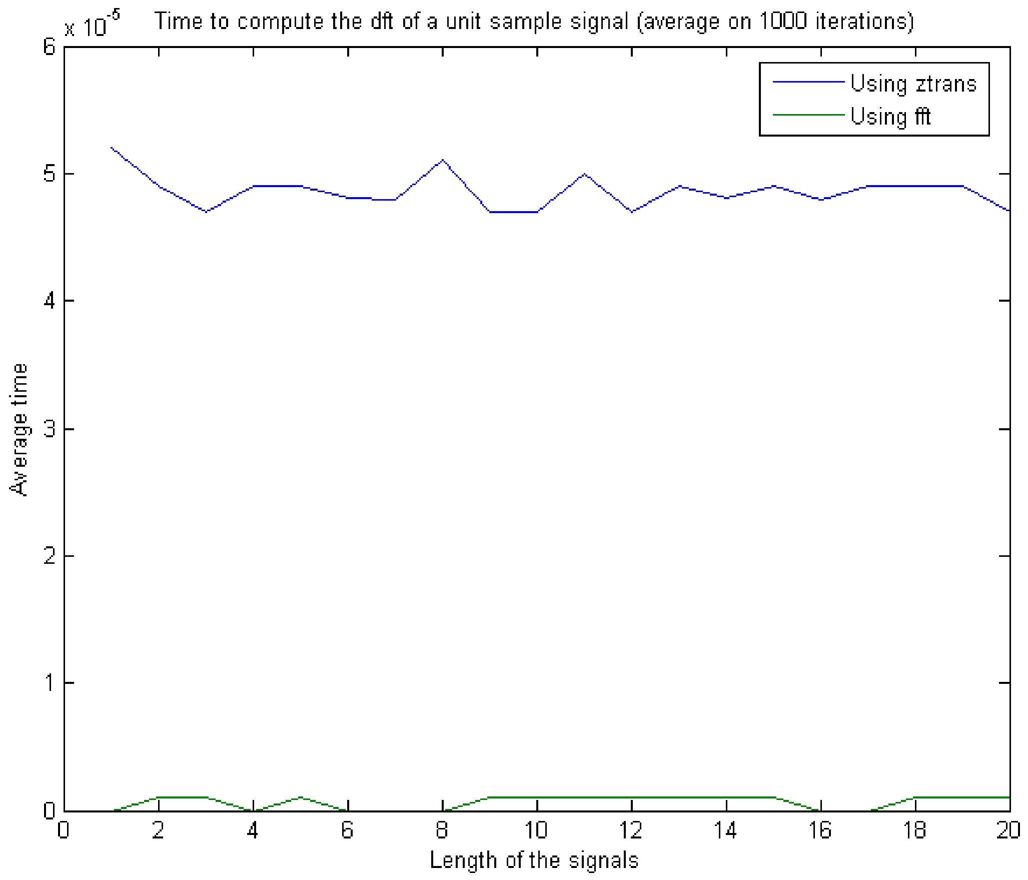
```
  t1=toc;
  timeDFT(1,n)=(t1-t0)/I;
```

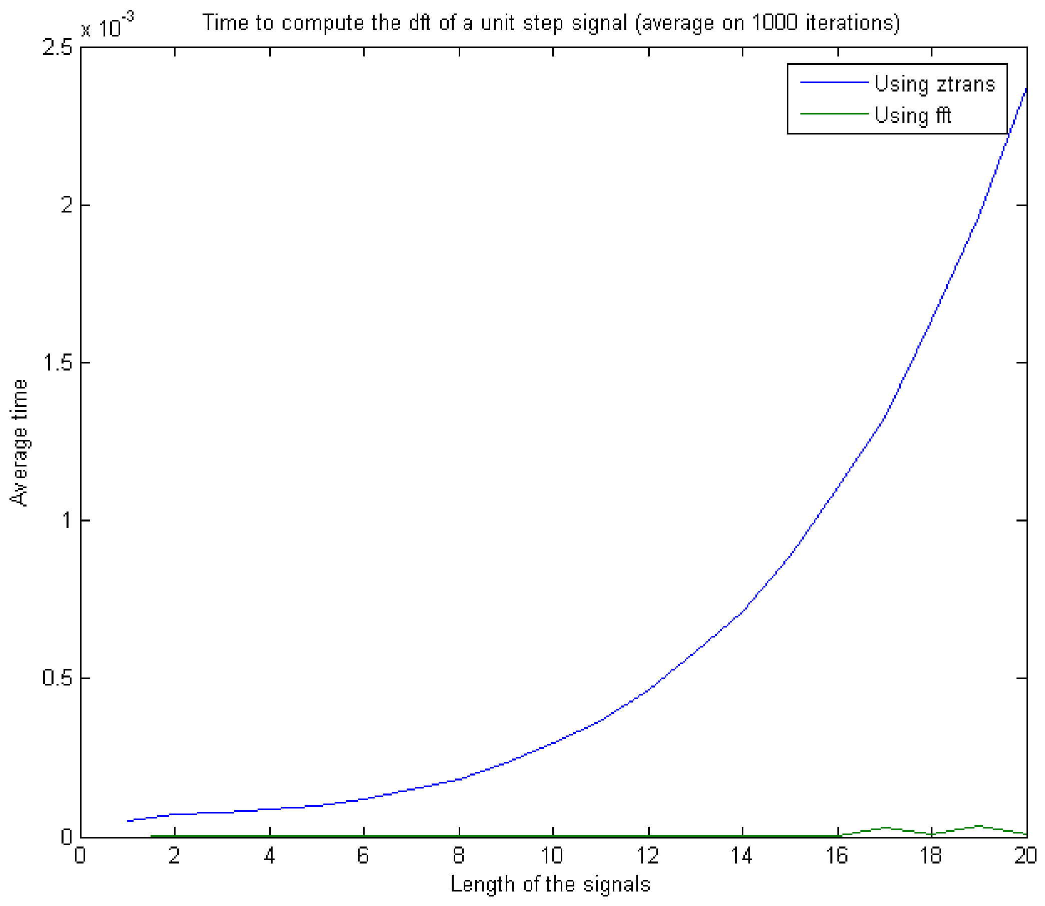
We store the result devided by 1000 in the array.

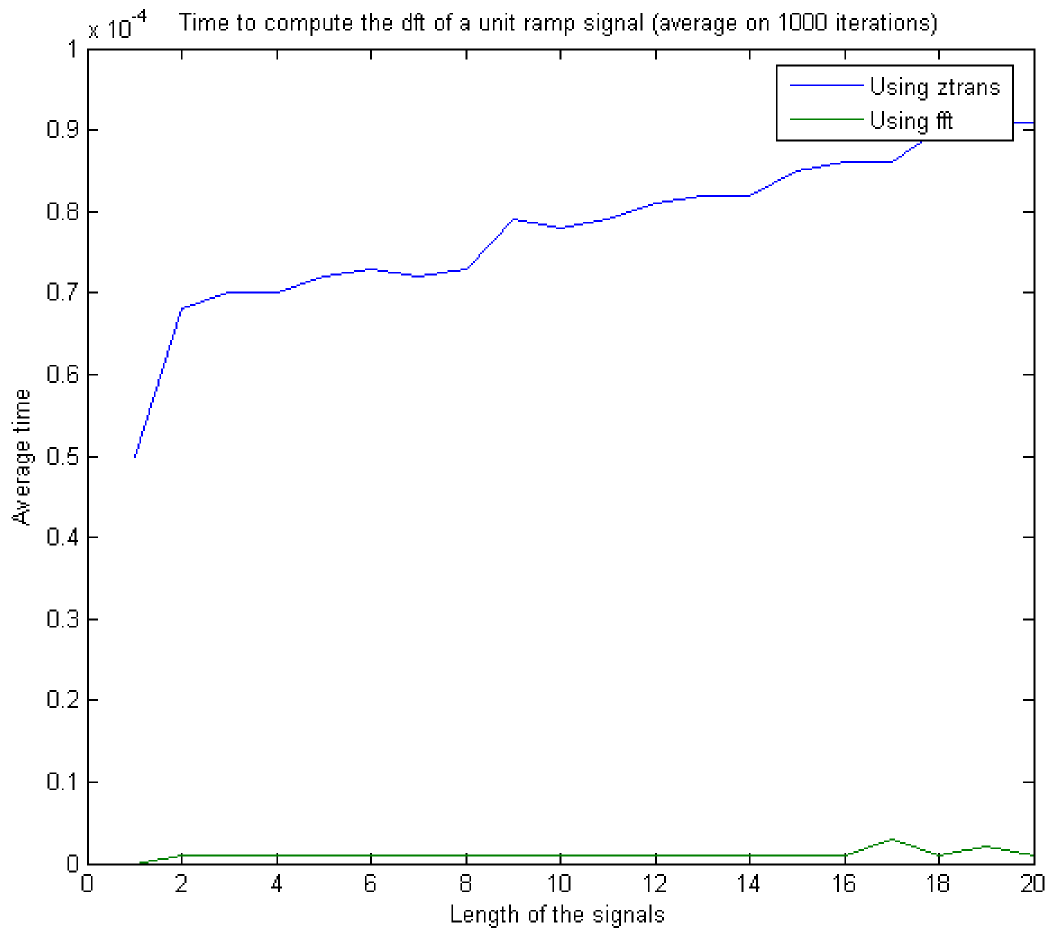
```
    …
end
```

We do so for each signal and each implementation of the dft

Here are the results :




Time to compute the dft of a unit step signal (average on 1000 iterations)
x 10^-3
Average time
Length of the signals
Using ztrans
Using fft
0
0.5
1
1.5
2
2.5
0
2
4
6
8
10
12
14
16
18
20



We can conclude that the fft implementation is way faster than the other one. Actually, it is so fast that for such short signals, the time elapsed during the computation using fft doesn't seem relevant : it isn't strictly increasing.

3. Visualizing a z-variable function

```
function []=drawZFunc(pointf)
```

Pointf is a handle to the function we want to represent.

```
  tx = -10:10;
  ty = -10:10;
  m=numel(tx);
  n=numel(ty);
```

We define arbitrary a range for the real part (tx) and the imaginary part (ty) of the z-variable. Here, this is a square but the function works with any rectangle.

```
  [X, Y] = meshgrid(tx,ty);
  Z = X+i*Y;
```

We create the matrix $Z = (k + il)_{k,l}$. It stores the values of z we will use.

```
  for k = 1:n
    for l =1:m
      tz(k,l)= feval(pointf,
Z(k,l));
    end
  end
```

For each value of z, we apply the function to z and store the result in the matrix tz.

```
  mesh(tx,ty,tz);
```

We show the three-dimensional plot.

4. Visualizing the amplitude of the z-transform

```
function []=drawZTrans(x)
  handle = @(z) abs(ztrans(x,z));
  drawZFunc (handle);
```

All we have to do is apply the last function to the composition of the modulus and $z \to \mathrm{ztrans}(x, z)$ for the chosen signal x.