EC2_Non_Recursive_FFT

April 2, 2019

```
In [1]: import math
        from cmath import exp, pi
        import numpy as np
```

```
In [2]: def fft_fn( v ):
            n, h  = len(v), len(v) >> 1
            previous = np.zeros((n,),dtype=np.complex128)
            previous = v[:]
            latest = np.zeros((n,),dtype=np.complex128)
            sublen, stride = 1, n

            while sublen <n:
                stride>>=1
                for i in range( stride ):
                    for k in range( 0,n,2*stride):
                        factor = exp(-2j*pi * k / n)
                        latest[i+(k>>1)]   = previous[i+k] + factor * previous[i+k+stride]
                        latest[i+(k>>1)+h] = previous[i+k] - factor * previous[i+k+stride]
                previous, latest = latest, previous
                sublen <<= 1

            return previous
```

```
In [3]: X = np.random.rand(1024,)
```

0.1 Now Lets test our non-recursive FFT

```
In [4]: fft_fn(X)
```

```
Out[4]: array([523.03315668+0.j        ,   4.69304583-0.74800533j,
                 5.73716514+5.57015994j, ...,   0.71648219-1.89630633j,
                 5.73716514-5.57015994j,   4.69304583+0.74800533j])
```

0.2 Now lets see the Numpy's FFT result for comparision

```
In [5]: np.fft.fft(X)
```

```
Out[5]: array([523.03315668+0.j        ,   4.69304583-0.74800533j,
                 5.73716514+5.57015994j, ...,   0.71648219-1.89630633j,
                 5.73716514-5.57015994j,   4.69304583+0.74800533j])
```

0.3 Now lets see if our Implimentation of non-recursive FFT is same as the Numpy's FFT

```
In [6]: np.allclose(fft_fn(X), np.fft.fft(X))
```

```
Out[6]: True
```

0.4 Hurray!! yes, they both are same :)