Use the DST to Compute the DFT for Odd Inputs

input如何修改

1. 將下圖的N改成要輸入的資料的長度，下圖為N=13時的例子

```
#include <cstdlib>
#include <iostream>
#include <cmath>

#define PI 3.14159265358979323846
#define N 13
```

2. 將輸入資料填入x_n[N]陣列中，如下圖

```
int main(int argc, char *argv[])
{
    float x_n[N] = {0, 1, 2, 3 , 4 , 5 , 6 , -6 , -5 , -4 , -3 , -2 , -1};
    float ans_n[N];
    int Q = N / 2;
    /////////// reset array ///////////
    for(int i=0; i<N ;i++){
        ans_n[i] = 0;
    }
    /////////// reset array ///////////
```

3. 執行程式，輸出的ans即為經過DST轉換的結果，其結果在輸入為odd時會和DFT相同

```
ans[0]= 0
ans[1]= -27.1608
ans[2]= 13.9868
ans[3]= -9.80211
ans[4]= 7.89809
ans[5]= -6.95175
ans[6]= 6.54774
ans[7]= -6.54774
ans[8]= 6.95175
ans[9]= -7.89809
ans[10]= 9.80211
ans[11]= -13.9868
ans[12]= 27.1608
請按任意鍵繼續 . . .
```

舉例與驗證

這邊使用python中的numpy模組的fft來與c++結果核對，python的程式碼如下圖：

```
import numpy as np

N = 7
f1 = np.ones(N , dtype = float)
f1 = [0, 1.1, 2.2, 3.3 , -3.3 , -2.2 , -1.1]
print('input: ')
print(f1)
dft = np.fft.fft(f1)
print('output: ')
print(dft)
```

1. N = 7，下圖為程式結果

```
x_n[0]= 0
x_n[1]= 1.1
x_n[2]= 2.2
x_n[3]= 3.3
x_n[4]= -3.3
x_n[5]= -2.2
x_n[6]= -1.1

ans[0]= 0
ans[1]= -8.87334
ans[2]= 4.92434
ans[3]= -3.94901
ans[4]= 3.94901
ans[5]= -4.92434
ans[6]= 8.87334
```

下圖為使用python DFT得出的結果

```
input:
[0, 1.1, 2.2, 3.3, -3.3, -2.2, -1.1]
output:
[0.+0.j         0.-8.87334475j 0.+4.92433483j 0.-3.94900992j
 0.+3.94900992j 0.-4.92433483j 0.+8.87334475j]
```

2. N = 6，下圖為程式結果

```
x_n[0]= 0
x_n[1]= 1.1
x_n[2]= 2.2
x_n[3]= 0
x_n[4]= -2.2
x_n[5]= -1.1

ans[0]= 0
ans[1]= -5.71577
ans[2]= 1.90526
ans[3]= 8.0824e-016
ans[4]= -1.90526
ans[5]= 5.71577
請按任意鍵繼續 . . .
```

下圖為使用python DFT得出的結果

```
input:
[0, 1.1, 2.2, 0, -2.2, -1.1]
output:
[0.+0.00000000e+00j 0.-5.71576766e+00j 0.+1.90525589e+00j
 0.+4.44089210e-16j 0.-1.90525589e+00j 0.+5.71576766e+00j]
```

3. N = 11，下圖為程式結果

```
x_n[0]= 0
x_n[1]= 1.1
x_n[2]= 2.2
x_n[3]= 3.3
x_n[4]= 1.414
x_n[5]= 6.9
x_n[6]= -6.9
x_n[7]= -1.414
x_n[8]= -3.3
x_n[9]= -2.2
x_n[10]= -1.1

ans[0]= 0
ans[1]= -17.7498
ans[2]= 6.793
ans[3]= -6.89269
ans[4]= 10.8805
ans[5]= -14.316
ans[6]= 14.316
ans[7]= -10.8805
ans[8]= 6.89269
ans[9]= -6.793
ans[10]= 17.7498
請按任意鍵繼續 . . .
```

下圖為使用python DFT得出的結果

```
input:
[0, 1.1, 2.2, 3.3, 1.414, 6.9, -6.9, -1.414, -3.3, -2.2, -1.1]
output:
[0. +0.j         0.-17.74978118j 0. +6.79300468j 0. -6.89268911j
 0.+10.88051775j 0.-14.31603583j 0.+14.31603583j 0.-10.88051775j
 0. +6.89268911j 0. -6.79300468j 0.+17.74978118j]
```

4. N = 12，下圖為程式結果

```
x_n[0]= 0
x_n[1]= 1.1
x_n[2]= 2.2
x_n[3]= 3.3
x_n[4]= 1.414
x_n[5]= 6.9
x_n[6]= 0
x_n[7]= -6.9
x_n[8]= -1.414
x_n[9]= -3.3
x_n[10]= -2.2
x_n[11]= -1.1

ans[0]= 0
ans[1]= -20.8596
ans[2]= 8.6845
ans[3]= -9.4
ans[4]= 11.4073
ans[5]= -8.34037
ans[6]= -3.31947e-014
ans[7]= 8.34037
ans[8]= -11.4073
ans[9]= 9.4
ans[10]= -8.6845
ans[11]= 20.8596
請按任意鍵繼續 . . .
```

下圖為使用python DFT得出的結果

```
input:
[0, 1.1, 2.2, 3.3, 1.414, 6.9, 0, -6.9, -1.414, -3.3, -2.2, -1.1]
output:
[0.+0.00000000e+00j 0.-2.08596316e+01j 0.+8.68450275e+00j
 0.-9.40000000e+00j 0.+1.14072866e+01j 0.-8.34036838e+00j
 0.-8.88178420e-16j 0.+8.34036838e+00j 0.-1.14072866e+01j
 0.+9.40000000e+00j 0.-8.68450275e+00j 0.+2.08596316e+01j]
```

注意事項

- 輸入要滿足x[n] = -x[N-n]，否則結果會與DFT的結果不相符
- c程式輸出的結果的數值會等於python結果的虛部
- N是奇數時，必定有x[0] = 0，因為x[n] = -x[N-n]
- N是偶數時，必定有x[0] = 0和x[N/2] = 0，因為x[n] = -x[N-n]