Group Members

Sreeja K Narne	57
Swati Verma	66
Rekapalli Nirajakshi	48

Computation of DFT using Radix-2 DIT-FFT algorithm

Aim:

To find DIT of a given sequence using DIT-FFT algorithm using MATLAB.

Abstract:

In the context of fast Fourier transform algorithms, a butterfly is a portion of the computation that combines the results of smaller discrete Fourier transforms (DFTs) into a larger DFT, or vice versa. Though it is not the efficient algorithm, it lays foundation for time-efficient DFT calculations. In Radix-2 decimation-in-frequency (DIF) FFT algorithm, original sequence s(n) is decomposed into two sub-sequences as first half and second half of a sequence. There is no need of reordering (shuffling) the original sequence as in Radix-2 decimation-in-time (DIT) FFT algorithm.

DIT-FFT algorithm

This method splits the DFTs into odd and even-complexed components.

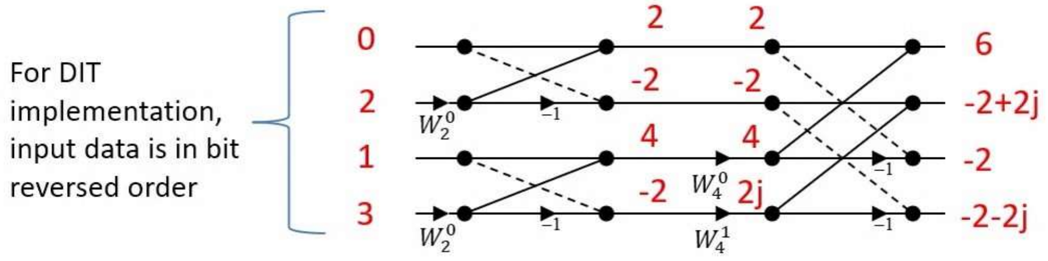
$$X[k] \triangleq \sum_{n=0}^{N-1} x[n]\, e^{-j2\pi \frac{nk}{N}}$$

$$= \sum_{n \text{ even}} x[n]\, e^{-j2\pi \frac{nk}{N}} + \sum_{n \text{ odd}} x[n]\, e^{-j2\pi \frac{nk}{N}}$$

$$= \sum_{m=0}^{\frac{N}{2}-1} x[2m]\, e^{-j2\pi \frac{(2m)k}{N}} + \sum_{m=0}^{\frac{N}{2}-1} x[2m+1]\, e^{-j2\pi \frac{(2m+1)k}{N}}$$

$$= \sum_{m=0}^{\frac{N}{2}-1} x[2m]\, e^{-j2\pi \frac{mk}{N/2}} + e^{-j2\pi \frac{k}{N}} \sum_{m=0}^{\frac{N}{2}-1} x[2m+1]\, e^{-j2\pi \frac{mk}{N/2}}$$

This is the last pass of the DIT-FFT algorithm.



$$W_2^0 = 1 \qquad W_4^0 = 1$$
$$W_4^1 = -j$$

Signal flow graph of DIT-FFT algorithm.

Procedure (by DIT-FFT method):

1 Step: First, we determine whether the length of the given array is even or not. If not, then we make it even by adding zeros behind it. To do this we first compute whether the size of the array is nearest to any power of 2. We then proceed to add zeros to the array accordingly.

2 Step: Then we compute the array size and number of conversion stages. The no. of conversion stages is given by log of N to the base 2 where N is the size of the array. Followed by the bit reversal of the input sequence.

3 Step: We initialise the start points of the butterfly structure. The butterfly structure should have a pair of (1,2), (1,3) and (1,5). We introduce a variable n which would determine the number of butterfly structures that are occurring in the algorithm.

4 Step: We introduce the twiddle factor. We multiply and add the upper points of the butterfly structure, subsequently we multiply and subtract from the lower points of the butterfly structure. Then we increment the points as well as n for the next structure.

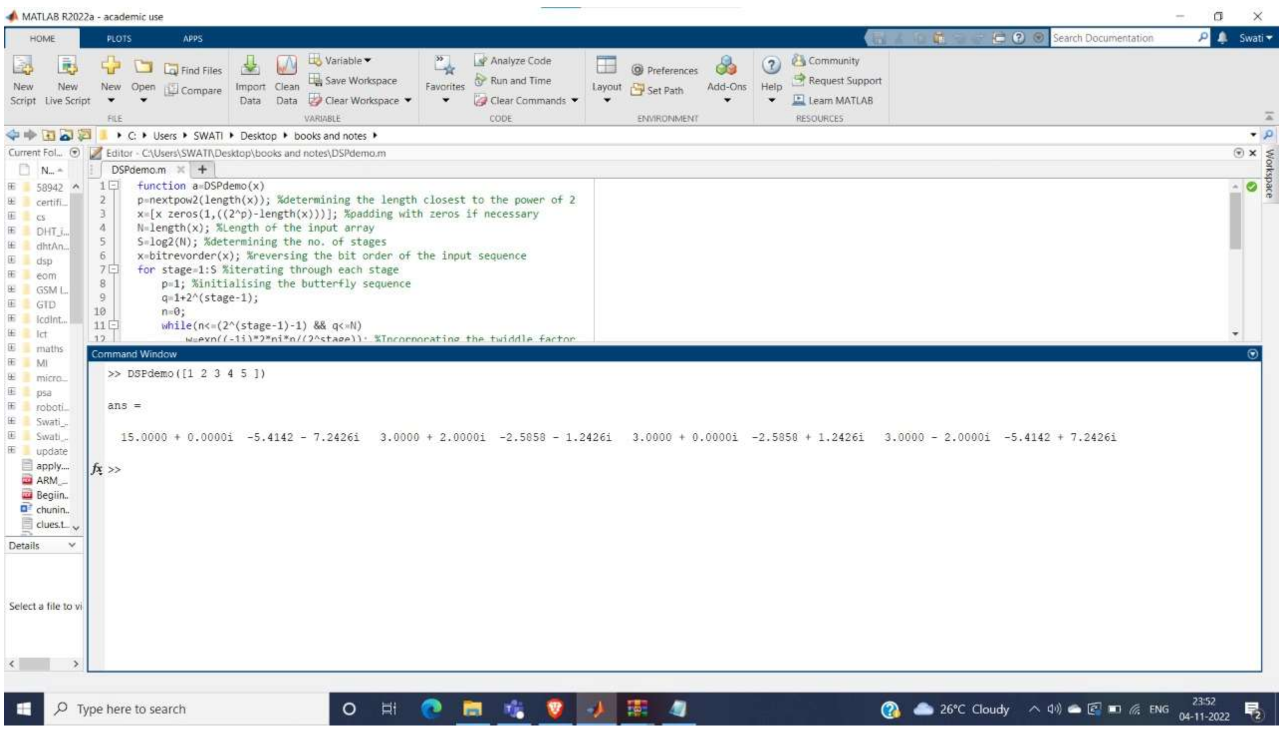
5 Step: We introduce a condition that would discontinue the butterfly structure and move on to the next stage.

Code for computation of DFT by DIT-FFT method.

```
function a=DSPdemo(x)
p=nextpow2(length(x)); %determining the length closest to
the power of 2
x=[x zeros(1,((2^p)-length(x)))]; %padding with zeros if
necessary
N=length(x); %Length of the input array
S=log2(N); %determining the no. of stages
x=bitrevorder(x); %reversing the bit order of the input
sequence
for stage=1:S %iterating through each stage
    p=1; %initialising the butterfly sequence
    q=1+2^(stage-1);
    n=0;
while(n<=(2^(stage-1)-1) && q<=N)
        w=exp((-1i)*2*pi*n/(2^stage)); %Incorporating the
twiddle factor
        y=x(p)+w*x(q); %Equation for 1st part of butterfly
operation
        z=x(p)-w*x(q); %Equation for 2nd part of butterfly
operation
        x(p)=y; %Equating
        x(q)=z;
        p=p+1; % Incrementing the sequences for next
butterfly structure
        q=q+1;
        n=n+1;
if(rem(q,2^stage)==1) % repetition of the butterfly
```

```
                p=p+2^(stage-1);% structure for other input
sequences
                q=q+2^(stage-1);
                n=0;
end
end
end
a=x;
end
```

Output



```
function a=DSPdemo(x)
p=nextpow2(length(x)); %determining the length closest to the power of 2
x=[x zeros(1,((2^p)-length(x)))]; %padding with zeros if necessary
N=length(x); %Length of the input array
S=log2(N); %determining the no. of stages
x=bitrevorder(x); %reversing the bit order of the input sequence
for stage=1:S %iterating through each stage
    p=1; %initialising the butterfly sequence
    q=1+2^(stage-1);
    n=0;
    while(n<=(2^(stage-1)-1) && q<=N)
        w=exp((-1i)*2*pi*n/(2^stage)); %Incorporating the twiddle factor
```

```
>> DSPdemo([1 2 3 4 5 ])

ans =

  15.0000 + 0.0000i  -5.4142 - 7.2426i   3.0000 + 2.0000i  -2.5858 - 1.2426i   3.0000 + 0.0000i  -2.5858 + 1.2426i   3.0000 - 2.0000i  -5.4142 + 7.2426i

>>
```

Differences between DIT-FFT and DIF-FFT method:

In DIF-FFT the output bit is in reversed order (complete opposite of the DIT-FFT method where the input bit is in reversed order). The DIF-FFT method splits the DFT inputs into two half samples in contradiction to DIT-FFT method which splits the DFTs of inputs into odd and even components. The DIT-FFT method reduces the no. of multiplications from N^2 to N*logbase2(N), DIF reduces it to (N/2)*logbase2(N) from N^2.

$$X[k] \triangleq \sum_{n=0}^{N-1} x[n]\, e^{-j2\pi\frac{nk}{N}}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x[n]\, e^{-j2\pi\frac{nk}{N}} + \sum_{n=\frac{N}{2}}^{N-1} x[n]\, e^{-j2\pi\frac{nk}{N}}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x[n]\, e^{-j2\pi\frac{nk}{N}} + \sum_{n=0}^{\frac{N}{2}-1} x[n+\tfrac{N}{2}]\, e^{-j2\pi\frac{(n+N/2)k}{N}}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x[n]\, e^{-j2\pi\frac{nk}{N}} + e^{-j2\pi\frac{(N/2)k}{N}} \sum_{n=0}^{\frac{N}{2}-1} x[n+\tfrac{N}{2}]\, e^{-j2\pi\frac{nk}{N}}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x[n]\, e^{-j2\pi\frac{nk}{N}} + e^{-j\pi k} \sum_{n=0}^{\frac{N}{2}-1} x[n+\tfrac{N}{2}]\, e^{-j2\pi\frac{nk}{N}}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} x[n]\, e^{-j2\pi\frac{nk}{N}} + (-1)^k \sum_{n=0}^{\frac{N}{2}-1} x[n+\tfrac{N}{2}]\, e^{-j2\pi\frac{nk}{N}}$$

$$= \sum_{n=0}^{\frac{N}{2}-1} \left(x[n] + (-1)^k x[n+\tfrac{N}{2}]\right) e^{-j2\pi\frac{nk}{N}}$$

This is how the butterflies are defined on the first pass of DFT for DIF method in contradiction to the last pass of DFT in DIT method.

Irrespective of the differences explained above both the methods would yield the same output.