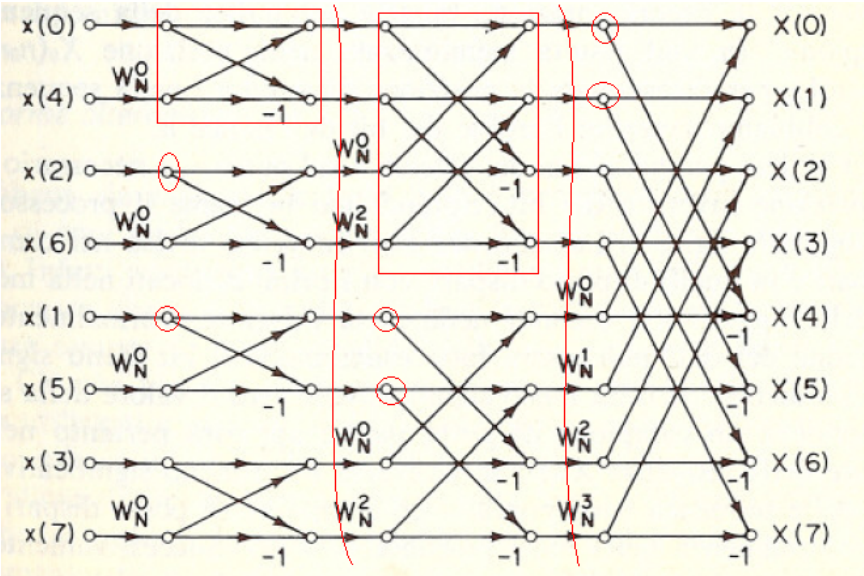
Trasformata di Fourier discreta

Descrizione dell'algoritmo

Inizialmente viene impostato un numero massimo di campioni analizzabili per consentire il calcolo della memoria necessaria. Viene allocata memoria per i vettori della sequenza reale e complessa e per i coefficienti W_i anch'essi reali e complessi. Viene creato il puntatore al file da analizzare e successivamente viene letta la dimensione del file stesso nel suo header. Successivamente viene verificato che il file non superi la quantità di campioni massima impostata all'inizio.
Se il test viene superato si passa all'adattamento alla più vicina potenza di 2. Disponendo ora della quantità precisa di campioni da analizzare, si può procedere all'inversione dei bit nelle posizioni del vettore di partenza. Successivamente vengono calcolati i due vettori contenenti i coefficienti W_i.

A questo punto viene effettuata l'inizializzazione dei vettori contenenti la sequenza in ingresso.
Per il vettore della parte immaginaria vengono impostati tutti i valori a zero.
Per il vettore reale invece viene posta una condizione che fa in modo di aggiungere degli zeri nel caso in cui il file non contenga abbastanza campioni da raggiungere la più vicina potenza di 2, nelle altre allocazioni invece vengono scritti i campioni presenti nel file secondo la disposizione invertita di bit calcolata precedentemente.

Viene creato un file .txt su cui scrivere la sequenza in uscita.
Inizia quindi il calcolo dopo aver dichiarato dei vettori di appoggio che hanno lo scopo di contenere inalterati i valori che la sequenza assume nello stadio precedente (una vera e propria memoria) in modo da renderli disponibili più volte durante la fase di calcolo sul posto che per definizione non conserva il valore precedente all'ultima scrittura.
I cicli di calcolo sono suddivisi in tre parti ciascuno con il suo contatore, il primo (quello più esterno) tiene conto dello stadio di calcolo, il secondo scorre i blocchi di calcolo in uno stadio, l'ultimo invece scorre gli elementi in ogni blocco di calcolo.



Nella figura a lato gli stadi sono separati dalle linee rosse verticali, alcuni blocchi sono indicati dai rettangoli, mentre alcuni elementi sono cerchiati.

Una volta effettuato il calcolo dei due vettori della serie finale, viene eseguito il modulo e il vettore risultante viene normalizzato e scritto sul file "ampiezze.txt".
Come ultima operazione vengono calcolati i valori delle ascisse corrispondenti ai centri dei bin di frequenza relativi alle ampiezze calcolate. I valori ottenuti vengono scritti sul file "ascisse.txt". Tale operazione viene eseguita allo scopo di una successiva rappresentazione grafica.

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define PI 3.14159265358979323
#define N 1024   //numero massimo di campioni

// coefficienti
double Wre[N/2];
double Wim[N/2];
// dichiarazione funzione di calcolo dei coefficienti
void CalcolaW(int n);

int main(int argc, char *argv[])
{
// locazioni per calcoli sul posto
  double Xre[N] = {0};
  double Xim[N] = {0};

  int numerobit;
  int i,j;
  int posizioni[N] = {0};
  int appoggio=0;
  int indice;
  int incremento;
  int inizioDati = 0;
  FILE* pf;

  short int valore;
  short int dimensione;
  int n;

  //pf=fopen("argv[1]","rb");
  pf=fopen("prova.wav","rb");
  if(pf == NULL)
  {
    printf("fallimento\n");
  }
  else
  {
  // lettura del file di partenza
    while(inizioDati<40)
     {
        fread(&valore,2,1,pf);
        inizioDati = inizioDati + 2;
     }

    int size;
    fread(&size,4,1,pf);
    //printf("size %d",size/2);
    size = size/2;

  if (size>N)
  printf("il file non può superare i 1000 campioni");
  else
  {
  int NC;
  float scarto;
  scarto = log2(size)-((int)log2(size));
  //printf("size %d",size);
  //printf("log2(size) %f",log2(size));
  if (scarto >= 0.5)
```

```
 NC = pow(2,(((int)log2(size))+1));
 else
 NC = pow(2,((int)log2(size)));

// inversione dei bit
numerobit=log2(NC);
//printf("numerobit = %d\n\n",numerobit);
for (i=0; i<NC; i++)
{
   appoggio =i;
   // scrittura sul vettore di inversione
   posizioni[i] = (appoggio&0x0001)|(posizioni[i]);
   for (j=0; j<(numerobit-1); j++)
   {
     appoggio = appoggio >> 1;
     posizioni[i] = ((posizioni[i])<<1);
     // scrittura di tutti i bit nel vettore di inversione
     posizioni[i] = (appoggio&0x0001)|(posizioni[i]);
   }

}
// calcolo dei coefficienti
CalcolaW(NC);

//printf("NC %d",NC);

// inizializzazione dei vettori sul posto
   i=0;
   if (size>=NC)
   {
 while(i<NC)
 {
    fread(&valore,2,1,pf);
      // scrittura dei vettori sul posto di partenza

   double valored = valore/32767.0;

 Xre[posizioni[i]] = valored;
   i=i+1;
   }
 }
 else
 {
    while(i<size)
 {
    fread(&valore,2,1,pf);
      // scrittura dei vettori sul posto di partenza

   double valored = valore/32767.0;

 Xre[posizioni[i]] = valored;
   i=i+1;
   }
    while(i<NC)
 {

     // scrittura dei vettori sul posto di partenza

   double valored = 0;

 Xre[posizioni[i]] = valored;
```

```
   i=i+1;
   }
 }
// apertura di un file .txt dove scrivere i dati
 FILE* fptxt;
 fptxt = fopen("ampiezza.txt","w");
// calcolo degli incrementi di Wi
int k=0;
int v=0;
int a,b;
int nBlocchi,elBlocco;
// costruzione vettori di appoggio da cui leggere i valori
double Xreapp[N] = {0};
double Ximapp[N] = {0};

for(i=0;i<NC;i++)
{
   Xreapp[i] = Xre[i];
   Ximapp[i] = Xim[i];
              }

for(i=0;i<numerobit;i++)
{
   nBlocchi = NC /pow(2,i+1);
   incremento = NC /pow(2,i+1);
   elBlocco= NC/nBlocchi;

    for (a=0; a<nBlocchi; a++)
    {

       for(b=0; b<(elBlocco/2); b++)
       {
        Xre[k]=Xreapp[k]+ (Wre[v]*Xreapp[k+(int)(pow(2,i))] - Wim[v]*Ximapp[k+(int)(pow(2,i))]);
        Xim[k]=Ximapp[k]+ (Wim[v]*Xreapp[k+(int)(pow(2,i))] + Wre[v]*Ximapp[k+(int)(pow(2,i))]);

        Xre[k+(int)(pow(2,i))]=Xreapp[k]- (Wre[v]*Xreapp[k+(int)(pow(2,i))] - Wim[v]*Ximapp[k+(int)(pow(2,i))]);
        Xim[k+(int)(pow(2,i))]=Ximapp[k]- (Wim[v]*Xreapp[k+(int)(pow(2,i))] + Wre[v]*Ximapp[k+(int)(pow(2,i))]);

        v = v + incremento;
        if (i != 0)
          k = k+1;
       }
       k = a*(elBlocco);
       k = k + (elBlocco);
       v=0;
    }
    k=0;
    // scrittura vettori d'appoggio successivi
    for(j=0;j<NC;j++)
    {
      Xreapp[j] = Xre[j];
      Ximapp[j] = Xim[j];
    }

   }

float A;
float max=0.0;
 // calcolo del modulo della serie finale
 for(i=0;i<NC;i++)
 {
```

```
   A = pow(((Xre[i]*Xre[i])+(Xim[i]*Xim[i])),0.5);
   if(A>max)
    max=A;
 }
 for(i=0;i<NC;i++)
{
   if (i<(NC/2))
   {
     A = pow(((Xre[i]*Xre[i])+(Xim[i]*Xim[i])),0.5);

   // normalizzazione
     fprintf(fptxt,"%2.3f\n",(A/max));
   }
   else
   {
      A = pow(((Xre[i]*Xre[i])+(Xim[i]*Xim[i])),0.5);

      }
 }
 FILE* fptxtasc;
 fptxtasc = fopen("ascisse.txt","w");

 float step=22050/NC;
 for(i=0; i<(NC/2); i++)
 {
 fprintf(fptxtasc,"%2.3f\n",step);
 step=step+(2*22050/NC);
 }
 // parentesi dell'else della condizione su size
 }
 // parentesi dell'else della condizione sul puntatore
 }
 system("PAUSE");
 return 0;
}
// definizione funzione CalcolaW()
void CalcolaW(int n)
{
    int i;
    for (i=0;i<(n/2);i++)
    {
       Wre[i]=cos(2*PI*i/n);
       Wim[i]=-sin(2*PI*i/n);

    }

}
```