Laboratorul 8

Calculul şi utilizarea Transformatei Fourier Discrete

8.1 Tema

Utilizarea transformatei Fourier discrete (TFD) pentru calculul spectrului unui semnal cu suport finit. Implementarea algoritmului FFT (transformata Fourier rapidă), în variantele bazate pe decimare în timp, respectiv frecvenţă.

8.2 Suport teoretic

Fie $x[n]$ un semnal cu suport $0 : N-1$. Transformata sa Fourier (TF) este

$$X(\omega) = \sum_{n=0}^{N-1} x[n] \mathrm{e}^{-j\omega n}. \tag{8.1}$$

Desigur, această expresie poate fi utilizată pentru evaluarea spectrului în orice punct $\omega \in [0, 2\pi]$. Totuşi, un mod de calcul mai eficient, prin care se obţine spectrul doar în N puncte echidistante, se bazează pe transformata Fourier discretă.

Definiţia 8.1 Transformata Fourier discretă (TFD) *a semnalului $x[n]$ cu suport $0 : N-1$ este secvenţa*

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn} = \sum_{n=0}^{N-1} x[n] w_N^{kn},\ k = 0 : N-1, \tag{8.2}$$

unde

$$w_N = e^{-j\frac{2\pi}{N}}. \tag{8.3}$$

Vom nota în continuare $X[k] = TFD_N(x[n])$ (ignorând eventual indicele N atunci când lungimea suportului rezultă din context).

Observaţia 8.1 Comparând (8.1) şi (8.2), observăm relaţia dintre TF şi TFD. Se observă că punând $\omega = 2\pi k/N$, $k \in 0 : N-1$, în TF (8.1), obţinem exact TFD (8.2). Aşadar, pentru semnale cu suport finit, TFD reprezintă o eşantionare a TF, în N frecvenţe echidistante aflate în intervalul $[0, 2\pi]$. Pentru ilustrare, prezentăm în figura 8.1 un semnal cu suport $0 : 7$ (deci $N = 8$), transformata Fourier a acestuia (care este o funcţie continuă cu suport $[0, 2\pi]$) şi transformata Fourier discretă a semnalului, care eşantionează TF în frecvenţele $0 : \pi/4 : 7\pi/4$.

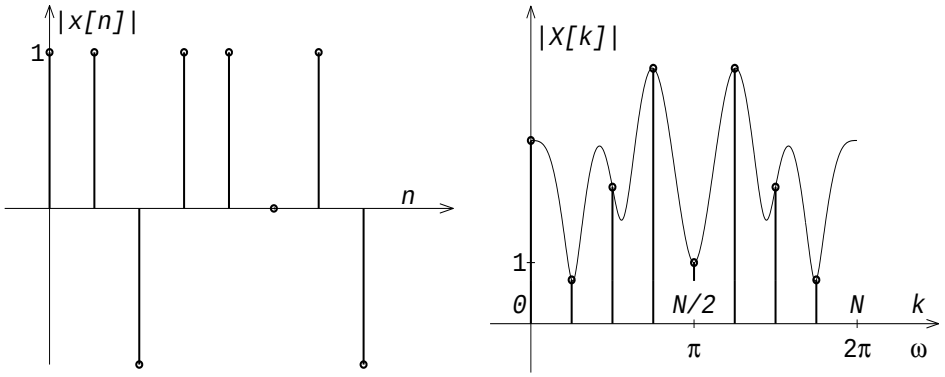

Figura 8.1: Stânga: un semnal cu suport $0:7$. Dreapta: spectrul semnalului (amplitudine) şi eşantioanele obţinute cu TFD.

Propoziţia 8.1 *Dacă $x[n]$ este un semnal cu suport $0:N-1$ şi $X[k]=TFD(x[n])$, atunci are loc egalitatea*

$$x[n]=\frac{1}{N}\sum_{k=0}^{N-1}X[k]e^{j\frac{2\pi}{N}kn}=\frac{1}{N}\sum_{k=0}^{N-1}X[k]w_N^{-kn}. \tag{8.4}$$

care defineşte transformata Fourier discretă inversă (TFDI).

Transformata Fourier rapidă

Transformata Fourier rapidă (FFT – Fast Fourier Transform) este numele generic pentru o clasă de algoritmi rapizi de calcul al transformatei Fourier discrete (TFD), pentru semnale cu suport finit. Numele FFT provine din complexitatea de $O(N\log_2 N)$ operaţii aritmetice necesare calculului TFD al unui semnal cu N eşantioane. Spre deosebire, algoritmul banal care implementează (8.2) ca atare (printr-o înmulţire matrice-vector) necesită $O(N^2)$ operaţii.

Algoritmii de tip FFT sunt bazaţi pe ideea de decimare, adică de reducere a calculului $\mathrm{TFD}_N(\cdot)$ la calculul unor TFD de lungime mai mică. Pentru simplitatea prezentării, vom considera doar cazul practic în care N este o putere a lui 2. În acest caz, decimarea este de fapt înjumătăţire, iar calculul $\mathrm{TFD}_N(\cdot)$ se reduce la calculul a două transformări $\mathrm{TFD}_{N/2}(\cdot)$. Ideea esenţială este de a aplica recursiv procedeul de decimare, pentru calculul fiecărei TFD de lungime mai mică. Aşadar, algoritmii FFT sunt de tip *divide et impera*.

Algoritmul FFT cu *decimare în timp* se bazează pe următoarea separare a relaţiei de definiţie (8.2):

$$\begin{aligned}
X[k] &= \sum_{n=0}^{N-1}x[n]w_N^{kn}\\
&= \sum_{n\ \text{par}}^{N-1}x[n]w_N^{kn}+\sum_{n\ \text{impar}}^{N-1}x[n]w_N^{kn}\\
&= \sum_{m=0}^{N/2-1}x[2m]w_N^{2km}+\sum_{m=0}^{N/2-1}x[2m+1]w_N^{k(2m+1)}\\
&= \sum_{m=0}^{N/2-1}x[2m]w_{N/2}^{km}+w_N^k\sum_{m=0}^{N/2-1}x[2m+1]w_{N/2}^{km}.
\end{aligned} \tag{8.5}$$

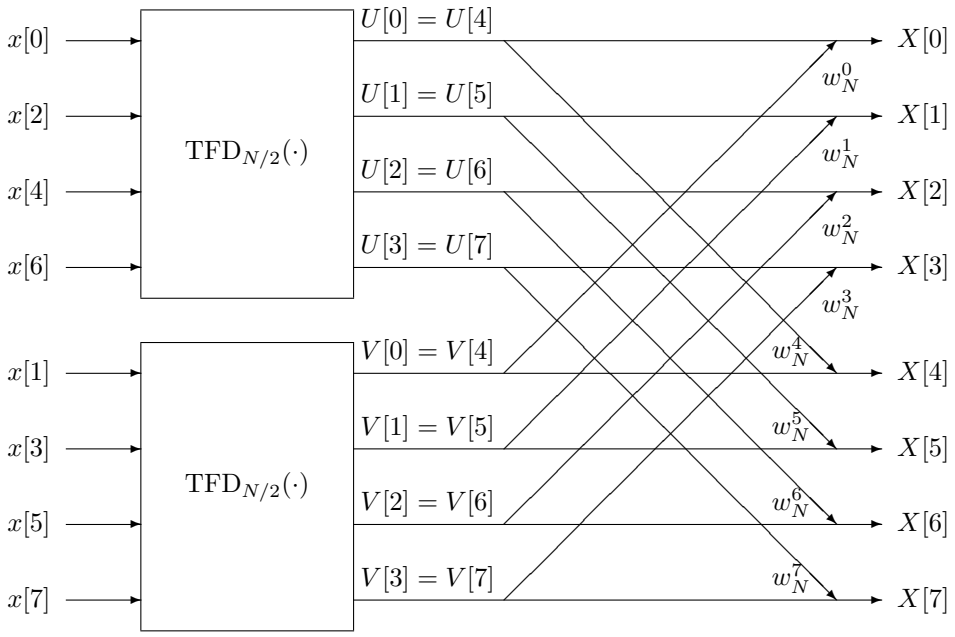

Figura 8.2: Calculul unei TFD de lungime $N = 8$ utilizând două TFD de lungime $N/2 = 4$ (decimare în timp).

Aşadar, în prima linie a şirului de egalităţi de mai sus avem definiţia (8.2) a TFD. În a doua linie, separăm suma din prima linie în două sume, una corespunzând indicilor pari, cealaltă celor impari. În a treia linie exprimăm formal separarea prin substituţiile $n = 2m$, $m = 0 : N/2 - 1$, în prima sumă, şi $n = 2m + 1$, $m = 0 : N/2 - 1$, în a doua. Trecerea la ultima linie se bazează pe relaţia evidentă

$$w_N^{2km} = \mathrm{e}^{-j\frac{2\pi}{N}2km} = \mathrm{e}^{-j\frac{2\pi}{N/2}km} = w_{N/2}^{km}.$$

Din (8.5) rezultă că $\mathrm{TFD}_N(x[n])$ se calculează ca sumă a două TFD a unor semnale cu suport $0 : N/2 - 1$, formate din eşantioanele pare, respectiv impare, ale semnalului $x[n]$. Notând $x_p[m] = x[2m]$ şi $x_i[m] = x[2m + 1]$ aceste semnale şi

$$U[k] = \mathrm{TFD}_{N/2}(x_p[m]), \quad V[k] = \mathrm{TFD}_{N/2}(x_i[m])$$

transformatele Fourier discrete ale lor (de lungime $N/2$), putem scrie

$$X[k] = U[k] + w_N^k V[k], \ k = 0 : N - 1. \tag{8.6}$$

Atenţie, în această relaţie avem $k = 0 : N - 1$, deci apar *două* perioade ale transformatelor Fourier discrete $U[k]$, $V[k]$. Schema de calcul corespunzătoare relaţiei (8.6) este ilustrată în figura 8.2; convenim ca unirea a două săgeţi să reprezinte o operaţie de sumare; o constantă, e.g. w_N^k, asociată unei săgeţi înseamnă înmulţirea semnalului reprezentat de săgeată cu constanta respectivă.

Aplicând recursiv modul de calcul din relaţia (8.5) şi figura 8.2, pentru calculul fiecărei TFD de lungime mai mică, se obţine următoarea funcţie recursivă, care calculează $\mathrm{TFD}_N(x[n])$.

funcţie $X = \mathrm{FFT}(x, N)$

1. **dacă** $N = 1$ **atunci**
    1. $X = x$
2. **altfel**

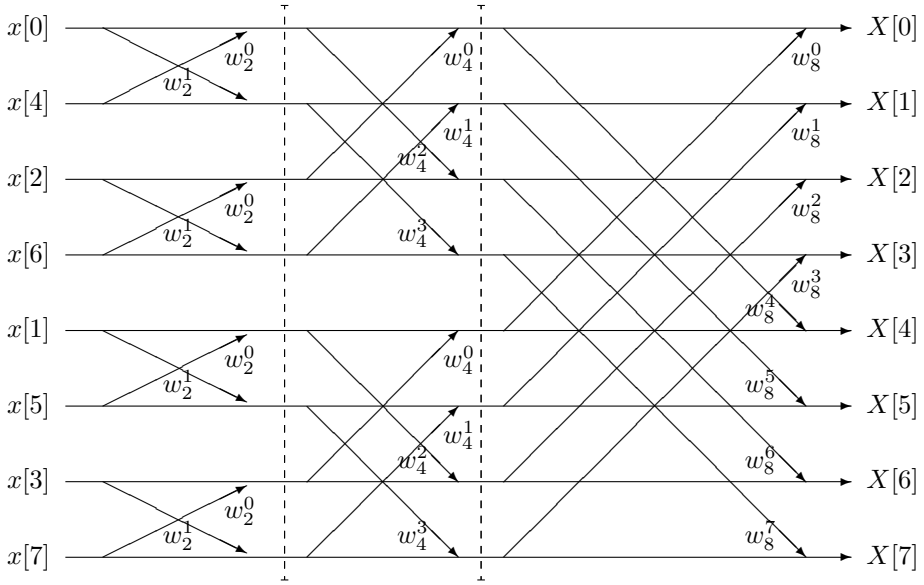

Figura 8.3: Schemă completă de calcul al TFD de lungime $N = 8$ (decimare în timp).

1. Pune $x_p[m] = x[2m]$, $x_i[m] = x[2m+1]$, pentru $m = 0 : N/2 - 1$
2. Calculează $U = \text{FFT}(x_p, N/2)$, $V = \text{FFT}(x_i, N/2)$ (apel recursiv)
3. $X[k] = U[k] + w_N^k V[k]$, pentru $k = 0 : N/2 - 1$
4. $X[k] = U[k - N/2] + w_N^k V[k - N/2]$, pentru $k = N/2 : N - 1$

În implementările practice ale algoritmului FFT nu se folosesc apelurile recursive, ci variante iterative echivalente, care conduc la programe mai eficiente. Pentru a ilustra varianta iterativă, am detaliat schema operaţiilor pentru calculul $\text{TFD}_8(\cdot)$ în figura 8.3. Schema, numită diagramă *fluture*, datorită formei sale, este obţinută din cea din figura 8.2 prin detalierea operaţiilor necesare tuturor TFD de lungime mai mică. Liniile punctate verticale separă etapele de calcul (iteraţiile sau nivelele de recursie, după cum interpretăm algoritmul); în fiecare etapă se combină TFD de lungime mai mică, pentru a obţine un număr de două ori mai mic de TFD de lungime dublă. Observăm cu uşurinţă că sunt $\log_2 N$ etape şi că în fiecare dintre ele se efectuează $2N$ operaţii, ceea ce conduce la totalul deja calculat de $2N \log_2 N$.

Eşantioanele semnalului de intrare sunt ordonate după numerele obţinute prin inversarea biţilor indicilor. De exemplu, cu referinţă la figura 8.3, elementul cu indicele $1 = 001_2$ este în poziţia $4 = 100_2$; ordinea completă este (toate numerele sunt în baza 2): 000, 100, 010, 110, 001, 101, 011, 111. Spunem că un semnal astfel ordonat se află în *ordine bit-inversă a indicilor*.

Înainte de a scrie algoritmul iterativ anunţat mai devreme, vom face mici modificări în diagrama din figura 8.3. Rezultatul este diagrama de calcul din figura 8.4, obţinută prin simple manipulări ale puterilor numerelor de tip (8.3). Observăm întâi că, cu referire la (8.6), avem $w_N^k = -w_N^{k-N/2}$. Apoi, pentru K putere a lui 2, putem scrie $w_K^m = w_N^{mN/K}$. În concluzie, constatăm că se pot utiliza doar constantele w_N^k, $k = 0 : N/2-1$. În fine, calculele se pot desfăşura pe loc în vectorul care conţine semnalul iniţial $x[n]$. Algoritmul corespunzător diagramei din figura 8.4 este următorul:

funcţie $x = \text{FFT\_decimare\_timp}(x, N)$

1. Ordonează x în ordine bit-inversă a indicilor
2. **pentru** $i = 1 : \log_2 N$
    1. $K = 2^i$

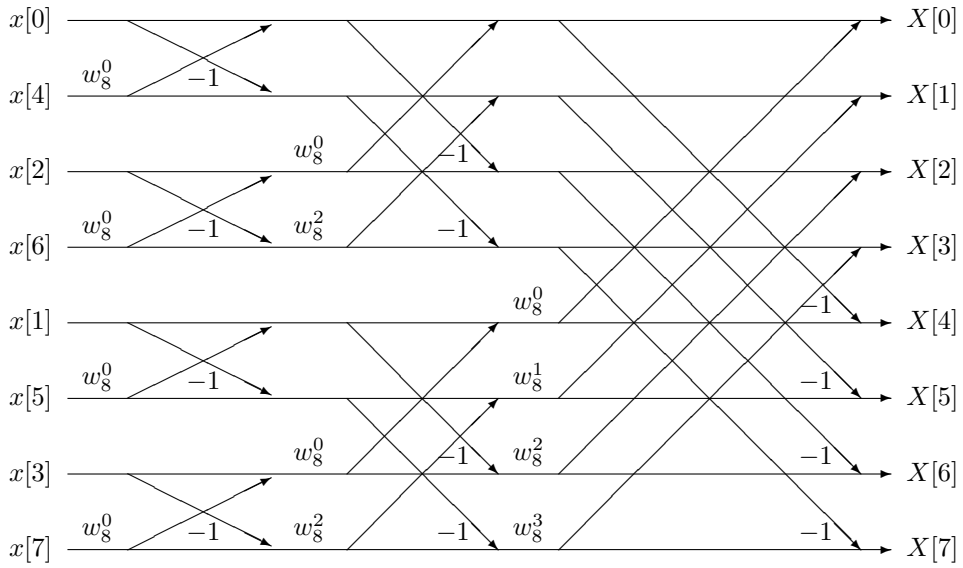

Figura 8.4: Schemă îmbunătăţită de calcul al TFD de lungime $N = 8$.

2. **pentru** $k = 0 : K : N - 1$
   1. **pentru** $m = 0 : K/2 - 1$
      1. $y = x[k + m]$
      2. $z = w_N^{mN/K} x[k + K/2 + m]$
      3. $x[k + m] = y + z$
      4. $x[k + K/2 + m] = y - z$

Acest algoritm necesită $3N/2 \log_2 N$ operaţii complexe, din care doar $N/2$ înmulţiri.

8.3 Ghid Matlab

Transformata Fourier discretă a unui semnal `x` (memorat într-un vector) se calculează cu

```
>> X = fft(x)
```

Dacă suportul mesajului are lungime putere a lui 2, atunci algoritmul folosit este FFT; altfel, se foloseşte un algoritm mai lent. Apelul

```
>> X = fft(x, N)
```

calculează TFD a semnalului `x` prelungit cu zerouri până la lungime `N`. O astfel de TFD calculează o eşantionare mai fină a spectrului semnalului.

Transformata Fourier discretă inversă se calculează cu

```
>> x = ifft(X)
```

Evident, `ifft(fft(x))` produce semnalul original (cu inevitabilele erori numerice).

Deoarece funcţia `fft` calculează DFT conform definiţiei (8.2), vectorul `X` obţinut reprezintă o eşantionare a spectrului definit pe $[0, 2\pi)$. Funcţia `fftshift` realizează deplasarea vectorului X astfel încât spectrul eşantionat să corespundă intervalului $[\pi, \pi)$. Apelul

```
>> Y = fftshift(X)
```

mută în primele $N/2$ poziţii din `Y` a doua jumătate a vectorului `X`.

8.4 Sarcini de lucru

Tema 8.1 (Utilizarea TFD pentru calculul spectrului unui semnal)

Luaţi un semnal $x[n]$ de lungime $N = 64$, de exemplu o sumă de sinusoide.

a. Desenaţi spectrul semnalului utilizând funcţia `freqz`.

b. Calculaţi TFD a semnalului, utilizând funcţia `fft`, şi reprezentaţi grafic amplitudinea TFD. Observaţi legătura dintre spectrul obţinut acum şi cel de la punctul **a**. Desenaţi un grafic în stilul celui din dreapta figurii 8.1.

c. Transformaţi TFD a semnalului cu ajutorul funcţiei `fftshift` şi redesenaţi spectrul. Care este legătura cu spectrul de la punctul **b** ?

d. Prelungiţi semnalul $x[n]$ cu zerouri până la lungimile $2N$ şi $4N$. Repetaţi operaţiile de la punctele **b** şi **c**. Observaţi că obţineţi o eşantionare mai fină a spectrului semnalului.

e. Calculaţi şi desenaţi TFD ale semnalelor $x_0[n] = 1$ şi $x_1[n] = (-1)^n$, ambele cu suport $0 : N - 1$. Încercaţi să anticipaţi rezultatele.

Tema 8.2 (Algoritmul FFT cu decimare în timp)

a. Funcţia `sort_bitinv` (aflată în acelaşi director cu lucrarea de laborator) primeşte un vector format din N numere şi întoarce vectorul ordonat în ordine bit-inversă a indicilor (se presupune că indicii încep de la zero şi că N este o putere a lui 2). Observaţi că operaţia se poate face pe loc în vector, prin interschimbări de elemente; dacă n este un indice şi $\tilde{n} = \mathit{bitinv}(n)$ indicele obţinut prin inversarea biţilor în reprezentarea binară a lui n, atunci avem $n = \mathit{bitinv}(\tilde{n})$. Apelaţi funcţia pentru un vector de dimensiune 8 sau 16 (de exemplu vectorul 0:7) şi constataţi corecta ei funcţionare.

b. Scrieţi o funcţie Matlab urmând algoritmul FFT_decimare_timp de la sfârşitul secţiunii teoretice a acestei lucrări.

c. Testaţi funcţia scrisă folosind-o în locul funcţiei `fft` în programele de la tema anterioară.

d. Completaţi funcţia pentru a funcţiona şi în cazul în care semnalul nu are lungimea putere a lui 2. În acest scop, prelungiţi semnalul cu zerouri până la prima putere a lui 2 mai mare ca lungimea semnalului.

Tema 8.3 (Supliment: algoritmul FFT cu decimare în frecvenţă)

Studiaţi din curs algoritmul FFT cu decimare în frecvenţă şi scrieţi un program Matlab pentru implementarea lui. Comparaţi rezultatele obţinute de cele două programe scrise în cadrul acestei lucrări de laborator.

Completaţi programele astfel încât să calculeze transformata Fourier discretă pentru orice lungime a vectorului de date.