ლაბორატორია 5

დისკრეტული ფურიე გარდაქმნა (DFT)

1 შესავალი

დღევანდელი და მომდევნო ლაბორატორია დისკრეტული ფურიე გარდამნის (DFT) და სწრაფი ფურიე გარდაქმნის (FFT) მეთოდებს შეეხება. დღეს დისკრეტულ ფურიე გარდაქმნას და მის თანმხლებ დისკრეტიზაციის და ფანჯრის ეფექტებს შეისწავლით, შემდეგ ლაბორატორიაში კი FFT-ს ალგორითმზე ვკონცენტრირდებით.

აქამდე სიგნალების და წრფივი დროით ინვარიანტული (LTI) სისტემების ანალიზისთვის დისკრეტული-დროის ფურიე გარდაქმნას (DTFT) ვიყენებდით.

$$\text{(DTFT)} \qquad X\left(e^{j\omega}\right) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n} \tag{1}$$

$$\text{(IDTFT)} \qquad x[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} X\left(e^{j\omega}\right)e^{j\omega n}\, d\omega. \tag{2}$$

მიუხედავად იმისა, რომ DTFT ძალიან გამოსადეგი და მნიშვნელოვანი ანალიზური იარაღია, უმეტეს შემთხვევაში მისი კომპიუტერით გამოთვლა შეუძლებელია, რადგან DTFT (1) უსასრულო ჯამს, ხოლო IDTFT (2) ინტეგრალის გამოთვლას საჭიროებს.

დისკრეტული ფურიე გარდაქმნა (DFT) დისკრეტული დროის ფურიე გარდაქმნის (DTFT) დისკრეტული ვერსიაა. შესაბამისად, DFT რიცხვით გამოთვლებს უკეთ ერგება.

$$\text{(DFT)} \qquad X_N[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi kn/N} \tag{3}$$

$$\text{(IDFT)} \qquad x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X_N[k]e^{j2\pi kn/N}\,, \tag{4}$$

სადაც $X_N[k]$, $x[n]$-ის N წერტილიანი დისკრეტული ფურიე გარდაქმნაა. $X_N[k]$ დისკრეტული ცვლადის, k, ფუნქციაა, $k = 0,1,\dots,N-1$.

ამ ლაბორატორიაში DTFT-დან DFT-ის გამოყვანას და რამდენიმე DFT იმპლემენტაციას შევისწავლით. ყველაზე სწრაფ და მნიშვნელოვან იმპლემენტაციას, სწრაფ ფურიე გარდაქმნას (FFT) შემდეგ ლაბორატორიაში დავუბრუნდებით.

2 DFT-ს გამოყვანა DTFT-დან

2.1 დროითი სიგნალის შემოკლება

DTFT-ის ზუსტი გამოთვლა არ არის შესაძლებელი (1)-ში არსებული უსასრულო ჯამის გამო. DTFT-ის მიახლოებით გამოთვლისთვის შესაძლებელია უსასრულო ჯამის შემოკლება (ფანჯრის მეთოდის გამოყენებით) და სასრული ვერსიის გამოთვლა. N-წერტილიანი მართხკუთხა ფანჯრისთვის გვაქვს:

$$w[n] = \begin{cases} 1 & n = 0,1,\dots,N-1 \\ 0 & \text{სხვა} \end{cases} \tag{5}$$

შესაბამისად, განვმარტავთ შემოკლებულ სიგნალს (x truncated):

$$x_{tr}[n] = w[n]x[n]. \tag{6}$$

$x_{tr}[n]$ სიგნალის DTFT მოცემულია, როგორც:

$$X_{tr}\left(e^{j\omega}\right) = \sum_{n=-\infty}^{\infty} x_{tr}[n]e^{-j\omega n} = \sum_{n=0}^{N} x[n]e^{-j\omega n}. \tag{7}$$

იდეალურ შემთხვევაში $X\left(e^{j\omega}\right)$-ს გამოთვლა გვსურს, თუმცა როგორც ფილტრის დიზაინის შემთხვევაში, შემოკლებისთვის გამოყენებული ფანჯარა სასურველი სიხშირული თვისებების დისტორციას იწვევს. როგორც წესი $X\left(e^{j\omega}\right)$ და $X_{tr}\left(e^{j\omega}\right)$ სხვადასხვა ფუნქციაა. ორ DTFT-ს შორის კავშირის გასაგებად, შეგვიძლია სიხშირულ სივრცეში ფანჯრის და $X\left(e^{j\omega}\right)$-ს კონვოლუცია გამოვთვალოთ:

$$X_{tr}\left(e^{j\omega}\right) = \frac{1}{2\pi}\int_{-\pi}^{\pi} X\left(e^{j\sigma}\right)W\left(e^{j(\omega-\sigma)}\right) d\sigma, \tag{8}$$

სადაც $W(e^{j\omega})$, $w[n]$-ის დისკრეტული დროის ფურიე გარდაქმნაა.

- გამოთვალეთ $W(e^{j\omega})$ DTFT ფორმულის მიხეთვით. გამოსახეთ ის $sinc$ ფუნქციის გამოყენებით.

2.2 სიხშირული დისკრეტიზაცია

განტოლება (7) სასრული რაოდენობით წევრების ჯამს შეიცავს. გარდა ამისა, ჯამის გამოსათვლელად, სიხშირული ცვლადის, ω, დისკრეტიზაცია არის საჭირო. კომპიუტერში DTFT-ის გამოსათვლელად სიხშირული სივრცის დისკრეტიზაცია გვჭირდება. შეგვიძლია ნებისმიერი სიხშირული წერტილები ავარჩიოთ, თუმცა ω-ს დისკრეტიზაციას თანაბარდაშორებულ N წერტილზე განვახორციელებთ (შუალედზე $[0,2\pi]$.)

$$\omega = \frac{2\pi k}{N}, \quad k = 0,1, \dots, N-1. \tag{8}$$

ჩასვით სიხშირის მნიშვნელობები განტოლება (7)-ში და გამოიყვანეთ კავშირი DTFT და DFT მნიშვნელობებს შორის.

2.3 ფანჯრის გამოყენების ეფექტი DTFT-ზე

თქვენი მიზანია გამოიკვლიოთ ფანჯრის მეთოდის ეფექტი სიგნალის, $x[n] = \cos\left(\frac{\pi}{4}n\right)$, DFT-ის გამოთვლაზე $N = 20$ სიგრძის ფანჯრის გამოყენებისას. ამ აქტივობისთვის გადმოწერეთ DTFT.m.

ასევე, **2.1**-ში თქვენი გამოყვანილი $W(e^{j\omega})$ დაგჭირდებათ:

$$W\left(e^{j\omega}\right) = e^{-jW(N-1)/2}\frac{\sin\left(\frac{\omega N}{2}\right)}{\sin\left(\frac{\omega}{2}\right)}.$$

1. ერთსა და იმავე ფანჯარაში ააგეთ $W\left(e^{j\omega}\right)$-ს მაგნიტუდის და ფაზის გრაფიკები.
2. გამოიყვანეთ გამოსახულება $X\left(e^{j\omega}\right)$-სთვის (შეუმოკლებელი სიგნალის DTFT).
3. შეამოკლეთ სიგნალი, $x[n]$, $N = 20$ სიგრძის ფანჯრის გამოყენებით. შემდეგ გამოიყენეთ DTFT.m ფუნქცია (512 წერტილით) $X_{tr}\left(e^{j\omega}\right)$-ის გამოსათვლელად.
   [X,w]=DTFT(x,512)
4. ააგეთ $X_{tr}\left(e^{j\omega}\right)$-ს გრაფიკი.
5. განიხილეთ განსხვავებები $|X_{tr}\left(e^{j\omega}\right)|$ და $|X\left(e^{j\omega}\right)|$-ს შორის.
6. მართხკუთხა ფანჯრის ნაცვლად გამოიყენეთ სხვა ფანჯარა (მაგალითად ჰემინგის ფანჯარა) და განიხილეთ განსხვავებები.
   ჰემინგის (Hamming) ფანჯარა - MATLAB: hamming(N)

$$w[n] = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) & n = 0,1,\ldots,N-1 \\ 0 & \text{სხვა} \end{cases}$$

3 დისკრეტული ფურიე გარდაქმნა

3.1 DFT-ის გამოთვლა

თქვენი მიზანია შემნათ DFT-ის გამოთვლის თქვენი ფუნქციები, DTFT-დან DFT-ის მიღების უკეთ გასააზრებლად.

- დაწერეთ MATLAB-ის ფუნქცია, რომელიც (3)-ის გამოიყენებით DFT-ის გამოითვლის:

  X = DFTsum(x),

  სადაც არგუმენტი x N-სიგრძის ვექტორია $x[0], x[1], \ldots, x[N-1]$ და X შესაბამისი დისკრეტული ფურიე გარდაქმნაა. ფუნქციაში ჯამი ზუსტად ისე დანეგეთ, როგორც (3)-ში არის მოცემული, n და k ცვლადიანი “for loop”-ის გამოყენებით. MATLAB-ში წარმოსახვით ერთეულად შეგიძლიათ 1j გამოიყენოთ.

  გამოსცადეთ თქვენი DFTsum ფუნქცია შემდეგი მიმდევრობების DFT-ის გამოთვლით:

  1. $x[n] = \delta[n], \quad N = 10.$
  2. $x[n] = 1, \quad N = 10.$
  3. $x[n] = e^{j2\pi n/10}, \quad N = 10.$
  4. $x[n] = \cos\left(\frac{2\pi n}{10}\right), \quad N = 10.$

  ააგეთ თითოეული DFT-ს მაგნიტუდის გრაფიკი. ასევე, გამოიყვანეთ თითოეული სიგნალის DTF-ს ანალიზური ფორმა. (DTFT-ში არ აგერიოთ.)

დაწერეთ მეორე MATLAB ფუნქცია რომელიც შებრუნებულ დისკრეტულ ფურიე გარდაქმნას გამოითვლის (4)-ის მიხედვით:

x = IDFTsum(X),

სადაც X *N*-სიგრძის ვექტორია, რომელიც DFT მნიშვნელობებისგან შედგება. x შესაბამისი დროითი სიგნალია. გამოიყენეთ ფუნქცია IDFTsum წინა ამოცანაში გამოთვლილი თითოეული DFT-ის შესაბრუნებლად. ააგეთ შებრუნებული DFT-ების გრაფიკები. გადაამოწმეთ, რომ IDFTsum-ით მიღებული სიგნალები ემთხვევა ორიგინალ სიგნალებს. გამოიყენეთ abs(x) დამრგვალების ცდომილების გამო წარმოქმნილი წარმოსახვითი კომპონენტების მოსაშორებლად.

3.2 DFT-ის მატრიცებით წარმოდგენა

DFT-ის იმპლემენტაცია მატრიცა-ვექტორის ნამრავლითაც არის შესაძლებელი. ამისთვის განიხილეთ განტოლება

$$X = \boldsymbol{A}x \tag{9}$$

სადაც $\boldsymbol{A}$ $N \times N$ მატრიცაა, X და x კი $N \times 1$ ვექტორია. ეს ნამრავლი შემდეგი ჯამის ექვივალენტურია

$$X_k = \sum_{n=1}^{N} \boldsymbol{A_{kn}} x_n, \tag{10}$$

სადაც $\boldsymbol{A_{kn}}$ შესაბამისი მატრიცის ელემენტია k-ური რიგის n-ურ სვეტში. (10) და (3) განტოლებების შედარებით ვამჩნევთ, რომ DFT-ისთვის

$$\boldsymbol{A_{kn}} = e^{-j2\pi(k-1)(n-1)/N} \tag{11}$$

-1-ები ექსპონენტებში MATLAB-ში ინდექსირების 1-დან და არა 0-დან დაწყებითაა გამოწვეული (ჯამი განტოლება (10)-ში $n = 1$-და იწყება).

- დაწერეთ MATLAB-ის ფუნქცია A=DFTmatrix(N), რომელიც $N \times N$ DFT მატრიცას, $\boldsymbol{A}$, დააბრუნებს.
- მატრიცა $\boldsymbol{A}$-ს გამოყენებით გამოთვალეთ შემდეგი სიგნალების DFT, ააგეთ მათი მაგნიტუდის გრაფიკები და შეადარეთ მიღებული შედეგები **3.1**-ში მიღებულ შედეგებს.
    1. $x[n] = \delta[n]$, $N = 10$.
    2. $x[n] = 1$, $N = 10$.
    3. $x[n] = e^{j2\pi n/10}$, $N = 10$.
- რამდენი გამრავლებაა საჭირო მატრიცის მეთოდით *N*-წერტილიანი DFT-ის გამოთვლისთვის?

როგორც DFT, შებრუნებული დისკრეტული ფურიე გარდაქმნაც (IDFT) შესაძლოა მატრიცა-ვექტორის ნამრავლით წარმოვადგინოთ.

$$x = \boldsymbol{B}X \tag{12}$$

- ჩაწერეთ ანალიტიკური გამოსახულება შებრუნებული DFT მატრიცის, $\boldsymbol{B}$, ელემენტებისთვის განტოლება (11)-ის ფორმით.
- დაწერეთ მატლაბის ფუნქცია B=IDFTmatrix(N), რომელიც $N \times N$ შებრუნებულ DFT მატრიცას,$\boldsymbol{B}$, დააბრუნებს.
- თქვენი ფუნქციების გამოყენებით გამოთვალეთ მატრიცები $\boldsymbol{A}$ და $\boldsymbol{B}$, $N = 5$. შემდეგ გამოთვალეთ მატრიცების ნამრავლი $\boldsymbol{C} = \boldsymbol{AB}$. რა ფორმისაა $\boldsymbol{C}$ მატრიცა? რატომ აქვს ეს ფორმა?

3.3 გამოთვლითი დროების შედარება

მიუხედავად იმისა, რომ DFTsum-ში შესრულებული გამოთვლები მათემატიკურად მატრიცების ნამრავლის ექვივაენტურია, ამ ორი DFT-ს გამოთვლითი დრო MATLAB-ში მნიშვნელოვნად განსხვავდება. (მაშინაც კი, როდესაც გამოთვლითი კომპლექსურობა ორივე შემთხვევაში იგივე რიგისაა) MATLAB-ში “for loop“-ების გამოყენების საფასური საკმაოდ მაღალია. ეფექტიანი კოდის დაწერისთვის ციკლების ნაცვლად ამოცანის მატრიცა/ვექტორების ნამრავლის ფორმულირება გვეხმარება. ამ ფაქტის სადემონსტრაციოდ:

1. გამოთვალეთ სიგნალი $x[n] = \cos\left(\frac{2\pi n}{10}\right), N = 512.$
2. გამოთვალეთ მატრიცა $\boldsymbol{A}, N = 512.$
3. შეადარეთ X = DFTsum(x) გამოთვლის დრო მატრიცებით იმპლემენტაციის X=A*x დროს cputime ფუნქციის გამოყენებით.

   მოათავსეთ cputime თითოეული მეთოდის შესაბამისი კოდის წინ და შემდეგ.

   მაგ: t=cputime; კოდი; cputime-t

   რა დრო დასჭირდა თითოეული იმპლემენტაციის გამოთვლას? რომელი მეთოდია უფრო სწრაფი? რომელი მეთოდი საჭიროებს ნაკლებ მეხსიერებას?