Gyors eljárások a diszkrét Fourier-transzformáció számítására. 3. rész

DR. KOCSIS FERENC
Budapesti Műszaki Egyetem,
Híradástechnikai Elektronika Intézet

ÖSSZEFOGLALÁS

Jelen dolgozatunk egy három részes sorozat utolsó tagja. E harmadik rész az első két részbeli eredmények alapján hatékony eljárásokat származtat a DFT számítására. Ezt követi a DFT algoritmusok összehasonlító értékelése. A Függelék gyors, kis pontszámú DFT algoritmusokat tartalmaz.

1. A DFT meghatározása a gyors konvolúciós eljárások felhasználásával

A dolgozat alább következő harmadik része a megelőző részekben közölt részeredmények felhasználásával (fokozatos részekre osztás, (1 – D)→(n – D), átalakítás, a DFT visszavezetése periodikus konvolúció számítására, $O(N)$ szorzásigényű konvolúciós eljárások) hatékony DFT számítási algoritmusokat mutat be. Az irodalomjegyzéket az 1. rész tartalmazza, a hivatkozások számozása annak megfelelő.

A periodikus konvolúció és a DFT kapcsolatának a vizsgálatánál kiderült, hogy a pontszámtól függően a DFT számítása visszavezethető periodikus konvolúció kiértékelésére. Láttuk azonban, hogy a periodikus konvolúció a szorzások számát tekintve elvileg O(N) (lineáris) bonyolultságú. Ez egyúttal azt is jelenti, hogy léteznek olyan pontszámok, amelyre a DFT í(N)=O(N) lépésszámmal meghatározható. A gyakorlati felhasználás szempontjából felmerülő két lényeges probléma: nagy N értékekre az algoritmus végrehajtása nehézségekbe ütközik, ill. sokszor van szükség a fentiektől eltérő alakban felírható pontszámú transzformációra. Ha N prímszám, akkor a (z^N-1) polinom például az adott feltételek mellett csak két tényezőre bontható $(z^N-1=(z-1)\sum_{j=0}^{N-1} z^j)$, s a szorzat második tagjának hossza miatt $N>13$ esetben a 2. rész (3-17)-beli **C** mátrix meghatározása igen nehéz. Mindkét nehézség megkerülhető a fokozatos részekre osztással, ill. az $1-D\rightarrow n-$D átalakítással. Ha N prím, akkor zérus értékű mintákkal kiegészítve az eredeti sorozatot, a pontszám összetett számmá változtatható. Továbbá minden N összetett szám előállítható törzstényezős alakban, másrészt a számunkra fontos pontszámok ($N\leqq 5000$) esetén a sorozatra bontás viszonylag kevés számú és kis értékű prímszámok felhasználásával elvégezhető. Kis N értékekre viszont a korábbi eredmények felhasználásával (periodikus konvolúcióvá történő átalakítás, ill. a konvolúció meghatározása a polinomokra érvényes kínai maradéktétellel) könnyen előállíthatók az optimális DFT eljárások (esetleg heurisztikus egyszerűsítéseket is felhasználva), majd ezek összekombinálásával adódnak a kívánt pontszámú transzformációk.

Az úgynevezett kis pontszámú (N=2, 3, 4, 5, 7, 8, 9 és 16) optimális DFT eljárásokat először Winograd dolgozta ki [28] és az 1. FÜGGELÉK tartalmazza. A szükséges szorzások száma az 1. táblázatból olvasható ki.

1. táblázat

N	Egyszerűsítés nélkül				±1, ±j értékkel való szorzások nélkül			
	M_c	A_c	$f(N)$	A_r	M_c	A_c	$f(N)$	A_r
2	2	2	4	4	0	2	0	4
3	3	6	6	12	2	6	4	12
4	4	8	8	16	0	8	0	16
5	6	17	12	34	5	17	10	34
7	9	36	18	72	8	36	16	72
8	8	26	16	52	2	26	4	52
9	11	45	22	90	9	45	18	90
16	18	74	36	148	10	74	20	148

M_c: a komplex szorzások száma, A_c: a komplex összeadások száma, A_r: a valós összeadások száma.

Noha az összeadások számának optimálására nem ismert szisztematikus eljárás, az összeadások száma ügyes csoportosítással csökkenthető. Az 1. FÜGGELÉK szerinti algoritmusok már ezen tény figyelembevételével készültek, így a gyakorlati megvalósítás során az előírt műveleti sorrend betartása lényeges. Az algoritmusokról meg kell jegyezni, hogy sok ±1 és ±j értékkel való szorzást (triviális szorzások) is tartalmaznak. A periodikus konvolúciót a kínai maradéktétel alapján számító módszer ugyanis sem-

Baérkezett: 1985. II. 6. (↔)

DR. KOCSIS FERENC

1975-ben szerzett villamosmérnöki diplomát a BME Villamosmérnöki Karán, majd a Távközlési Kutató Intézetben kezdett dolgozni. Egyetemi doktori értekezését 1978-ban védte meg. 1983 szeptembere óta a BME HEI-ben dolgozik tudományos ösztöndíjasként, ahol a digitális jelfeldolgozás és jelszintézis algoritmikus kérdéseivel foglalkozik. Szakmai érdeklődési köre: rendszertechnika, digitális jelfeldolgozás, számítástechnika, algoritmusok elmélete.

milyen feltevést nem tesz az $\{x(i)\}$, ill. a $\{h(i)\}$ sorozatokra vonatkozóan. *DFT* esetén azonban a $\{h(i)\}$ előre ismert speciális sorozat: $\left\{e^{-j\left(\frac{2\pi}{N}\right)kl}\right\}$, s az exponenciális függvény tulajdonságai következtében sok szorzás triviális szorzása egyszerűsödik. Ezenkívül a kis pontszámú algoritmusok kihasználják a komplex konjugáltak szimmetriatulajdonságait, aminek eredményeképp a nem triviális szorzások is vagy tisztán valósak, vagy tisztán képzetesek (sohasem általános komplex szorzások). Az egyes algoritmustipusok műveletigényének meghatározásakor a 2 és a 4 szerinti faktorizációnál követett eljáráshoz hasonlóan figyelembe vesszük a lehetséges egyszerűsítéseket is. A kis pontszámú *DFT* eljárások Winograd-algoritmussal nagyobb pontszámú (hosszabb) transzformációba való összekombinálásához azonban egyelőre szükség van a triviális szorzások nem triviálisként való kezelésére. Ennek megfelelően az 1. FÜGGELÉK is tartalmazza ezen szorzásokat. A triviális szorzások megtartása egyúttal az első részben említett redukált transzformációnak az együtthatómátrix elhagyott első sorával és első oszlopával való kiegészítését is jelenti. Az algoritmusok szerkezetét vizsgálva látható, hogy a számítások lényegében 3 fő lépésben végezhetők el:

- a bemeneti adatokból (az $\{x(i)\}$ sorozat elemeiből) összeadások és kivonások felhasználásával segédmennyiségek képzése (előrendezés),
- a segédmennyiségekből és az előre ismert állandókból (az exponenciális *DFT* együtthatókból) álló szorzatok előállítása,
- végül a kapott szorzatokból összeadások és kivonások útján segédmennyiségek, ill. a keresett transzformáltak meghatározása (utórendezés).

Az eljárás azonos a periodikus konvolúció szisztematikus (Toom—Cook algoritmuson alapuló) meghatározására alkalmazott módszerrel. A számítási lépéseket formalizálva az egyes lépésekhez mátrixműveletek rendelhetők. A bemeneti adatokból összeadással és kivonással kapott segédmennyiségek származtathatók az $\{x(i)\}$ sorozat értékeiből álló vektor egy ± 1 és 0 elemeket tartalmazó mátrixszal való szorzásával. Az együtthatókkal történő szorzás megvalósítható diagonálmátrixszal való szorzással is (az állandók a diagonál mentén helyezkednek el). Végül a transzformáltakat a szorzatokból ismét egy ± 1 és 0 elemeket tartalmazó mátrixszal való szorzással lehet képezni. Azaz a kis pontszámú *DFT* eljárások együtthatómátrixa három mátrix szorzataként is előállítható:

(1-1) $\mathbf{X}=\mathbf{W}\mathbf{x}=\mathbf{SCT}\mathrm{x}$, vagyis $\mathbf{W}=\mathbf{SCT}$.

T a bemeneti adatokon végrehajtott összeadások és kivonások mátrixa (előrendező), **C** a szorzások mátrixa, míg **S** a kimeneti adatokon végrehajtott összeadások és kivonások mátrixa (utórendező).

A felbontások létezésének formális bizonyítása általános esetre meglehetősen nehéz, azonban kis N értékekre magukból az algoritmusokból származtathatók. **C** elemei csak tisztán valósak vagy tisztán képzetesek, továbbá a képzetes és a valós elemek is mindig egy csoportban, a diagonál ellentétes oldalán helyezkednek el. A kis pontszámú algoritmusoknál **T** M_cxN, **C** M_cxM_c és **S** NxM_c méretű (M_c a 2. táblázat szerint szükséges komplex szorzások száma). Egyes nagy N értékekre a felbontás létezését annak előállításával mutatjuk meg.

2. táblázat

N	Tényezők	*WFTA*		Good-algoritmus		*FFT*	
		$f(N)$	$A_r(N)$	$f(N)$	$A_r(N)$	$f(N)$	$A_r(N)$
30	5,3,2	68	384	68	384		
32	2					102	422
48	16,3	88	636				
60	5,4,3	136	888	136	896		
63	9,7	194	1408	512	1386		
64	2^6					294	960
120	8,5,3	276	2076	332	2076		
126	9,7,2	388	3312	512	2920		
128	2^7					774	2566
168	8,7,3	432	3068	580	3492		
240	16,5,3	628	5016				
252	9,7,4	776	7128	1024	6344		
256	2^8					1926	6022
315	9,7,5	1184	10426	1784	8812		
420	5,4,3,7	1288	11352	1712	9936		
504	9,8,7	1572	14540	2300	13948		
512	2^9					4614	13830
840	8,7,5,3	2580	24804	4244	23172		
1008	16,9,7	3532	34668	5120			
1024	2^{10}					10758	31238
1260	9,7,5,4	4736	46664	7136	40288		
2048	2^{11}					24582	69638
2520	9,8,7,5	9408	99628	15532	86876		
5040	16,9,7,5	21368	233928				

A kis pontszámú ($N=2, 3, 4, 5, 7, 8$ és 9), triviális szorzásokat nem tartalmazó optimális *DFT* eljárásokat a 2. FÜGGELÉK tartalmazza [13]. Az algoritmusok kialakításánál elsődleges cél volt a műveletszám minimálása, az esetleges mátrixfaktorizáció szempontjainak a mellőzésével. A 2. FÜGGELÉK szerinti alaktól eltérő fórmák a nagyobb transzformációk ún. Good-algoritmussal való előállításánál előnyösek.

1.2. A Good-féle eljárás

Ha az N pontszám egymáshoz relatív prím számok szorzatára bontható, akkor az $1-D$ *DFT* több dimenziós transzformációba alakítható át. A 2. rész (1-20) szerint az eredeti transzformáció együtthatómátrixa az egyes dimenzióbeli együtthatómátrixok Kronecker-szorzataként állítható elő. Ennek algoritmikus megfelelője:

1. N/N_n db N_n pontszámú *DFT* számítása
⋮
j. N/N_{n+1-j} db N_{n+1-j} pontszámú *DFT* számítása
⋮
n. N/N_1 db N_1 pontszámú *DFT* számítása.

Ha $f(N_j)$ az N_j pontszámú *DFT* meghatározásának költsége (pl. a szükséges valós szorzások száma), akkor az első rész szerint az N pontszámú *DFT* számításának költségfüggvénye $f(N)=\sum_{j=1}^{n} N/N_j \cdot f(N_j)$. A kis pontszámú *DFT* algoritmusok viszont $cN=O(N)$

rendűek, így $f(N)=O(nN)$. Ha $n<\log N$, akkor valóban $O(N \log N)$ rendűnél hatékonyabb eljárást kapunk. Például $N=1001=7\cdot11\cdot13$ pontszámnál $n=3$, vagyis $Of(N)=(3N)$. A kisebb pontszámú transzformációk összekombinálásának fenti módszerét Good javasolta először [12] 1958-ban, de addig nem volt versenyképes a 2 és 4 szerinti faktorizációt alkalmazó *FFT* módszerekkel, amíg a hatékony kis pontszámú algoritmusokat fel nem fedezték. A Good-eljárás általában az összeadások és nem a szorzások számát minimálja. Az optimális *DFT* blokkokat alkalmazó Good-algoritmus az irodalomban prímtényezős algoritmus (*PFA*) néven ismeretes. Műveletigénye:

$$M_c(n, N)=\sum_{j=1}^{n} M_c(N_j)\frac{N}{N_j} \qquad N=\prod_{j=1}^{n} N_j$$

$$A_c(n, N)=\sum_{j=1}^{n} A_c(N_j)\cdot(N/N_j) \quad (N_j, N_k)=1 \tag{1-2}$$

komplex szorzás és összeadás. Mivel az optimális *DFT* blokkokban csak tisztán képzetes, vagy valós szorzások szerepelnek, a valós szorzások és összeadások száma:

$$f(N)=2\cdot M_c(n, N) \quad \text{és} \quad A(N)=2\cdot A_c(n, N). \tag{1-3}$$

azaz az eredeti transzformáció együtthatómátrixa is felbontható három mátrix szorzatára. Az 1. FÜGGELÉK szerinti kis pontszámú *DFT* blokkokra támaszkodó, az (1-6) alatti felbontást alkalmazó *DFT* eljárásokat szokás Winograd-féle Fourier-transzformációs algoritmusoknak (*WFTA*) is nevezni [28]. A szükséges szorzásokat a középső mátrix írja le, amely maga is diagonálmátrix. Elemei előre számítható állandók. Mérete megadja a szükséges szorzások számát:

$$f(N)=2\cdot\left[\sum_{j=1}^{n} M_c(N_j)-K\right], \tag{1-7}$$

ahol a K korrekciós tényezővel lehet figyelembe venni az egyes *DFT* modulokban levő triviális szorzások számát. Értéke: $K=K_1\cdot K_2\ldots K_n$, ahol K_j a j-edik blokkban a triviális szorzások száma. A kis pontszámú *DFT* eljárások összekombinálásának fenti módját szokás egymásba ágyazásnak nevezni. A komplex összeadások számát megadó kifejezés bizonyítás nélkül [25]:

$$A_c(n, N)=\sum_{j=1}^{n}\left(\prod_{l=1}^{n-j} N_l\right)\cdot[I(N_j)+O(N_j)], \tag{1-8}$$

ahol $\prod_{l=a}^{b}(\cdot)=1$, ha $a>b$. $I(N_j)$ jelöli a $\mathbf{T}_j$ mátrixból adódó, $O(N_j)$ pedig az $\mathbf{S}_j$ mátrixból adódó összeadások és kivonások számát.

A tényezőkre bontás sorrendje sem a *Good*-, sem a *WFT*-algoritmusnál nem befolyásolja a szorzások számát, csupán az utóbbi esetben az összeadások számát. Az összeadásszám minimálásának egy lehetséges módja: az összes lehetséges $n!$ sorrendre meghatározni az összeadások számát, és a minimális számúra vezethető sorrendet választani.

Az (1-7) kifejezés szerint a *WFTA* szorzásigénye N pontszámnál $O(N)$, mivel az egyes optimális *DFT* blokkok szorzásigénye is $O(N_j)$ rendű. A korábbi feltételek mellett ismét meghatározva az elérhető jelfeldolgozási frekvenciát ($N=1008$, $t_{szorzás}=200$ ns, $f(N)=3524$) $f_{maxjel}=707{,}07$ kHz adódik, ami másfélszeres növekedést jelent a *PFA* algoritmussal elérhető frekvenciákhoz képest.

Végül a *PFA* eljárással elérhető maximális kötött idejű jelfeldolgozási frekvencia értéke:

$$f_{maxjel}=470{,}5 \text{ kHz} \tag{1-4}$$

($N=1008=7\cdot9\cdot16\approx10^3$, $f(N)=5356$, $t_{szorzás}=200$ ns), ami 2,7-szeres növekedést jelent a 2 szerinti faktorizációt alkalmazó *FFT* eljáráshoz képest.

1.3. A Winograd-algoritmus (*WFTA*)

Winograd a kis pontszámú *DFT* algoritmusokon kívül új módszert is talált a kis pontszámú blokkok összerakására [28]. Az egyes dimenzióbeli együtthatómátrixokra érvényes (1-1) összefüggés szerinti felbontással a kiindulási, N pontszámú, $1-D$ *DFT* együtthatómátrixát meghatározva:

$$\mathbf{W}_N=\mathbf{W}_1\otimes\mathbf{W}_2\otimes\ldots\otimes\mathbf{W}_n= \\ =(\mathbf{S}_1\mathbf{C}_1\mathbf{T}_1)\otimes(\mathbf{S}_2\mathbf{C}_2\mathbf{T}_2)\otimes\ldots\otimes(\mathbf{S}_n\mathbf{C}_n\mathbf{T}_n) \tag{1-5}$$

A Kronecker-szorzatokra érvényes $(\mathbf{AB})\otimes(\mathbf{CD})=(\mathbf{A}\otimes\mathbf{C})(\mathbf{B}\otimes\mathbf{D})$ azonosság ismételt felhasználásával:

$$\mathbf{W}_N=(\mathbf{S}_1\otimes\mathbf{S}_2\otimes\ldots\otimes\mathbf{S}_n)(\mathbf{C}_1\otimes\mathbf{C}_2\otimes\ldots\otimes\mathbf{C}_n)(\mathbf{T}_1\otimes\mathbf{T}_2\otimes\ldots\otimes\mathbf{T}_n), \tag{1-6}$$

2. A DFT számítására szolgáló algoritmusok összehasonlítása és értékelése

A *DFT* első részben említett közvetlen kiértékelésének $O(N^2)$ szorzásigényétől eljutottunk a *WFT* algoritmus $O(N)$ műveletigényéig. A származtatott összefüggések alapján néhány pontszámra konkrétan meg is határoztuk a szükséges valós szorzások és összeadások számát. Az eredményeket a 2. táblázat foglalja össze. A kapott adatok alapján látható, hogy a *PFA* típusú algoritmusokhoz a vizsgált tartományban 0—65%-kal több szorzásra van szükség, mint a *WFT* eljárásokhoz. Az összehasonlítás a 2 szerinti faktorizációt alkalmazó *FFT* algoritmusoknál még kedvezőbb képet mutat: az *FFT* szorzásigénye 1,5—3-szorosa a *WFT* eljárásénak. Az összeadások száma azonban némiképp nőtt (0—19%-kal a *PFA* algoritmushoz képest. Az egyes algoritmusoknak a gyakorlati felhasználás szempontjából történő összehasonlításánál azonban a szorzások számán kívül esetleg más tényezőket is figyelembe kell venni. Mint arról már az előzőekben szó volt, az $1-D\rightarrow n-D$ átalakítás, valamint a *DFT* periodikus konvolúcióvá történő átalakítás során az együtthatómátrix, a kiindulási adatok és a transzformáltak átrendezésére is szükség lehet, azaz az aritmetikai műveleteken kívül adatátviteli műveleteket is kell végezni. Noha a műveletszám meghatározásánál ezen problémák vizsgálatára nem térünk ki, a korábbi elemzések szerint [15], [16] számuk olyan nagy lehet, hogy a végrehaj-

táshoz szükséges idő összemérhető lehet a szorzáshoz szükséges idővel. Ugyanakkor az *FFT* és a *PFA* algoritmusok minden nehézség nélkül megvalósíthatók úgy, hogy nincs szükség átrendezésre. A *WFT* és a *PFA* eljárások kombinálásával, a szorzások számának kismértékű növekedése árán a *WFT* algoritmus is kialakítható úgynevezett helyben történő transzformáció végzésére [6].

A másik problémát az összeadások nagy száma jelenti. Noha a *DFT* algoritmusok bonyolultságát a szükséges valós szorzások számával mértük, a megvalósítás során az összeadások száma is jelentős lehet.

A gyors eljárások részét képező fokozatos részekre osztás konkrét megvalósításától, a kapott kisebb pontszámú transzformációk számításának a módjától és az eredmények összerakásától függően nagyszámú *DFT* algoritmus képzelhető el. Összehasonlításukra először a szorzások száma alapján próbálkoztunk, de kiderült, hogy ez nem minden esetben célravezető. Fokozza a nehézséget, hogy a teljes műveletigény (összeadások, kivonások, szorzások, adatmozgatások) és az elérhető pontosság (a véges szóhosszúságból eredő hatások) mindig a konkrét megvalósítás függvényei: általános célú számítógépen kívánjuk-e a számításokat végezni, milyen utasításkészlet áll rendelkezésre, mekkora a szóhosszúság, mi az egyes műveletek végrehajtási idők egymáshoz való viszonya, mekkora a rendelkezésre álló operatív tár stb. Az adott alkalmazásban optimális számítási algoritmus kiválasztásának egy lehetséges módja lehet az egyes tényezőket figyelembe vevő, több változós, a változók által kifeszített téren értelmezett költségfüggvény definiálása, majd ennek valamilyen matematikai módszerrel történő optimálása. A függvény értékkészletének pontjai konkrét *DFT* algoritmusoknak felelnek meg. Az előzőekben felsorolt tényezőket figyelembe vevő lineáris költségfüggvény lehet a következő:

(2-1)

$$C_{DFT}(N)=K_1f(N)+K_2A(N)+K_3D(N)+K_4S(N),$$

ahol $D(N)$ az adatátvitelek száma, $S(N)$ a szükséges tárterület mérete, K_1—K_4 alkalmas együtthatók (esetleg állandók). Természetesen sokkal általánosabb költségfüggvények is elképzelhetők. Az együtthatók mindig a rendelkezésre álló erőforrásoktól függőek, azokra jellemzőek.

Az *FFT* algoritmusok a 2—2048 tartományban 11 pontszámra értelmezettek. A *PFA* algoritmus a 2. FÜGGELÉK szerinti (2, 3, 4, 5, 7, 8 és 9) pontszámú *DFT* modulokkal 47 különböző pontszámú transzformációt definiál a 2—2520 tartományban, míg a *WFTA* 1. FÜGGELÉKBEN található *DFT* blokkjai a 2—5040 tartományban 59 pontszámú *DFT* számítását teszik lehetővé. A pontok száma sűríthető újabb kis pontszámú *DFT* modulok (pl. 11, 13, 17 stb.) meghatározásával. Ha olyan pontszámú *DFT* meghatározására van szükség, amelyre egyik algoritmus sem értelmezett, jól használható eljárás valamely, következő ismert pontszámig a zérusokkal való kiegészítés. Másik lehetőség a pontszám egymáshoz relatív prím számokra szorzatára való bontása, majd az adódó részeredmények összekombinálása az $f(N)=\sum_{j=1}^{n} f(N_j)\frac{N}{N_j}+(n-1)N$ összefüggés alapján a forgatási tényezők figyelembevételével.

Az elérhető kötött idejű jelfeldolgozási frekvencia (csak a szorzásokat figyelembe véve) a 850 Hz körüli értékről 707 kHz-re nőtt. Az algoritmusok gyakorlati kivitelezése során az elvileg elérhető $2N-K$ szorzásnál (K az N osztóinak a száma) nagyobb számú szorzásra van szükség. Optimális esetet feltéve, azon nagyobb pontszámokra, amelyekre K értéke a $2N$ mellett már elhanyagolható, és a *DFT* átalakítható periodikus konvolúcióvá, f_{maxjel} értéke:

(2-2)
$$f_{maxjel}\cong\frac{N}{2\cdot t_{szorzás}\cdot 2N}=\frac{1}{4\cdot t_{szorzás}}$$

200 ns szorzásidőnél $f_{maxjel}\sim 1{,}25$ MHz. Következésképp elvileg még a *WFTA* 707 kHz-es eredménye is algoritmikusan továbbfejleszthető: újabb módszerek kereshetők a kis pontszámú *DFT* modulok összeillesztésére, nagyobb pontszámokra is közvetlenül a polinomokra vonatkozó kínai maradéktétel alapján származtatni a transzformációt stb.

Az eddigiekben hallgatólagosan feltettük, hogy a számítások során a szorzások mindig szekvenciális módon hajtódnak végre. Azonnal adódik, a kötött idejű jelfeldolgozási frekvencia növelésének másik kézenfekvő módja: a számításokat egymástól függetlenül végezhető részekre bontani, és ezeket egymással párhuzamosan végrehajtani. Az elérhető sebességnövekedés ára a szükséges szorzók (processzorok) számának növekedése.

A bemutatott számítási algoritmusok nemcsak egydimenziós, hanem egyéb jelfeldolgozási feladatokban (pl. képfeldolgozás) felmerülő több dimenziós transzformációk számítására is alkalmasak. Az algoritmusokkal egyszerű változócserével a

(3-3)
$$x(i)=\frac{1}{N}\sum_{k=0}^{N-1}X(k)e^{j(2\pi/N)ki}\qquad 0\le N\le 1$$

inverz *DFT* is meghatározható (*IDFT*).

A periodikus és a lineáris konvolúció gyors meghatározására ismertetett, a Toom—Cook tétel alkalmazásán alapuló algoritmusok a véges súlyfüggvényű (*FIR*) szűrők tervezőinek a lehetőségeit is növelik.

Az eredmények szerint csupán algoritmikus módszerekkel (lényegében a műveletvégrehajtási sorrend ügyes szervezésével) a kötött idejű jelfeldolgozási frekvenciatartomány közel két nagyságrenddel megnőtt azonos sebességű áramköri elemeket alkalmazva.

1. FÜGGELÉK

Gyors eljárások a kis pontszámú diszkrét Fourier-transzformáltak számítására a Winograd-algoritmushoz.

Algoritmus 1. $N=2$, $N_A=2$, $M_N=0$, $A=2$

$m_0=1\cdot(x(0)+x(1))$ $m_1=1\cdot(x(0)-x(1))$ $X(0)=m_0$

$X(1)=m_1$

Algoritmus 2. $N=3$, $M_A=3$, $M_N=2$, $A=6$, $u=2\pi/3$

$t_1=x(1)+x(2)$ $m_0=x(0)+t_1$ $m_1=t_1(\cos u-1)$
$m_2=j(x(1)-x(2))\cdot\sin u$ $s_1=m_0+m_1$ $X(0)=m_0$
$X(1)=s_1+m_2$ $X(2)=s_1-m_2$

Algoritmus 3. $N=4$, $M_A=4$, $M_N=0$, $A=8$
$t_1=x(0)+x(2)$ $t_2=x(1)+x(3)$ $m_0=1\cdot(t_1+t_2)$
$m_1=1\cdot(t_1-t_2)$ $m_2=j(x(0)-x(2))$ $m_3=j(x(1)-x(3))$
$X(0)=m_0$ $X(1)=m_2+m_3$ $X(2)=m_1$ $X(3)=m_2-m_3$

Algoritmus 4. $N=5$, $M_A=6$, $M_N=5$, $A=17$, $u=2\pi/5$
$t_1=x(1)+x(4)$ $t_2=x(2)+x(4)$ $t_3=x(1)-x(4)$
$t_4=x(2)+x(4)$ $t_5=t_1+t_2$ $m_0=1\cdot(x(0)+t_5)$
$m_1=0{,}5(\cos u+\cos 2u-2)t_5$
$m_2=0{,}5(t_1-t_2)$ $(\cos n-\cos 2u)$
$m_3=j(t_3+t_4)\sin u.$
$m_4=jt_4(\sin u+\sin 2u)$ $m_5=jt_3(\sin 2u-\sin u)$
$s_1=m_0+m_1$ $s_2=s_1+m_2$ $s_3=m_3-m_4$ $s_4=s_1-m_2$
$s_5=m_3+m_5$
$X(0)=m_0$ $X(1)=s_2+s_3$ $X(2)=s_1+s_5$ $X(3)=s_4-s_5$
$X(4)=s_2-s_3$

Algoritmus 5. $N=7$, $M_A=9$, $M_N=8$, $A=36$, $u=2\pi/7$
$t_1=x(1)+x(6)$ $t_2=x(2)+x(5)$ $t_3=(3)x+x(4)$
$t_4=t_1+t_2+t_3$ $t_5=x(1)-x(6)$ $t_6=x(2)-x(5)$
$t_7=x(4)-x(3)$ $m_0=1\cdot(x(0)+t_4)$
$m_1=t_4(\cos u+\cos 2u+\cos 3u-3)/3$
$m_2=(t_1-t_3)(2\cos u-\cos 2u-\cos 3u)$
$m_3=(t_3-t_2)(\cos u-2\cos 2u+\cos 3u)/3$
$m_4=(t_2-t_1)(\cos u+\cos 2u-2\cos 3u)/3$
$m_5=j(t_5+t_6+t_1)(\sin u+\sin 2u-\sin 3u)/3$
$m_6=j(t_5-t_7)(\sin u-2\sin 2u+\sin 3u)/3$
$m_7=j(t_7-t_6)(\sin u-2\sin u-\sin 3u)/3$
$m_8=j(t_6-t_5)(\sin u+\sin 2u+2\sin 3u)/3$
$s_1=m_0+m_1$ $s_2=s_1+m_2+m_3$ $s_3=s_1-m_2-m_4$
$s_4=s_1-m_3+m_4$ $s_5=m_5+m_6+m_7$ $s_6=m_5-m_6-m_8$
$s_7=m_5-m_7+m_8$ $X(0)=m_0$ $X(1)=s_2+s_5$
$X(2)=s_3+s_6$ $X(3)=s_4-s_7$ $X(4)=s_4+s_7$
$X(5)=s_3-s_6$ $X(6)=s_2-s_5$

Algoritmus 6. $N=8$, $M_A=8$, $M_N=2$, $A=26$, $u=2\pi/8$
$t_1=x(0)+x(4)$ $t_2=x(2)+x(6)$ $t_3=x(1)+x(5)$
$t_4=x(1)-x(5)$ $t_5=x(3)+x(7)$ $t_6=x(3)-x(7)$
$t_7=t_1+t_2$
$t_8=t_3+t_8$ $m_0=1\cdot(t_7+t_8)$ $m_1=1\cdot(t_7-t_8)$
$m_2=1\cdot(t_1-t_2)$
$m_3=1\cdot(x(0)-x(4))$ $m_4=(t_4-t_6)\cos u$ $m_5=j(t_3-t_5)$
$m_6=j(x(2)-x(6))$ $m_7=j(t_4+t_6)\sin u$ $s_1=m_3+m_4$
$s_2=m_3-m_4$ $s_3=m_6+m_7$ $s_4=m_6-m_7$ $X(0)=m_0$
$X(1)=s_1+s_3$ $X(2)=m_2+m_5$ $X(3)=s_2-s_4$
$X(4)=m_1$ $X(5)=s_2+s_4$ $X(6)=m_2-m_5$ $X(7)=s_1-s_3$

Algoritmus 7. $N=9$, $M_A=11$, $M_n=9$, $A=45$, $u=2\pi/9$
$t_1=x(1)+x(8)$ $t_2=x(1)-x(8)$ $t_3=x(7)+x(2)$
$t_4=x(7)-x(2)$ $t_5=x(3)+x(6)$ $t_6=x(3)-x(6)$
$t_7=x(4)+x(5)$ $t_8=x(4)-x(5)$ $t_9=t_1+t_3$ $t_{10}=t_9+t_7$
$t_{11}=t_{10}+t_5$ $t_{12}=t_{11}+x(0)$ $t_{13}=t_2+t_4$ $t_{14}=t_{13}+t_8$
$t_{15}=t_1-t_3$ $t_{16}=t_3-t_7$ $t_{17}=t_7-t_1$ $t_{15}=t_2-t_4$
$t_{19}=t_4-t_8$
$t_{20}=t_8-t_2$ $m_0=1\cdot t_{12}$ $m_2=-0{,}5t_{10}$ $m_2=\mathrm{j}\sin 3u$
$m_3=t_5(\cos 3u-1)$ $m_4=\mathrm{j}t_6\sin 3u$
$m_5=t_{15}(2\cos u-\cos 2u-\cos 4u)/3$
$m_6=t_{16}(\cos a+\cos 2u-2\cos 4u)/3$
$m_7=t_{17}(\cos u-2\cos 2u+\cos 4u)/3$
$m_8=\mathrm{j}t_{18}(2\sin u+\sin 2u-\sin 4u)/3$
$m_9=\mathrm{j}t_{19}(\sin u-\sin 2u-2\sin 4u)/3$
$m_{10}=\mathrm{j}t_{20}(\sin n+2\sin 2u+\sin 4u)/3$
$s_1=m_1+m_2$ $s_2=t_{20}+m_1$ $s_3=m_0+s_2$ $s_4=s_3+m_2$
$s_5=s_3-m_2$ $s_6=m_0+m_3$ $s_7=s_6+s_1$
$s_8=s_7+m_S$ $s_9=s_8+m_6$ $s_{10}=s_7-m_6$ $s_{11}=s_{10}+m_7$
$s_{12}=s_7-m_5$ $s_{13}=s_{12}-m_7$ $s_{14}=m_4+m_8$ $s_{15}=s_{14}+m_9$
$s_{16}=m_4-m_9$ $s_{17}=s_{16}+m_{10}$ $s_{18}=m_4-m_8$
$s_{19}=s_{18}-m_{10}$
$s_{20}=s_9+s_{15}$ $s_{21}=s_9-s_{15}$ $s_{22}=s_{11}+s_{17}$ $s_{23}=s_{11}-s_{17}$
$s_{24}=s_{13}+s_{19}$ $s_{25}=s_{15}-s_{19}$ $X(0)=m_0$ $X(1)=s_{20}$
$X(2)=s_{23}$ $X(3)=s_4$ $X(4)=s_{24}$ $X(5)=s_{25}$
$X(6)=s_3$ $X(7)=s_{22}$ $X(8)=s_{21}$

Algoritmus 8. $N=16$, $M_A=18$, $M_N=10$, $A=74$
$u=2\pi/16$
$t_1=x(0)+x(8)$ $t_2=x(4)+x(1)$ $t_3=x(2)+x(10)$
$t_4=x(2)-x(10)$ $t_5=x(6)+x(14)$ $t_6=x(6)-x(14)$
$t_7=x(1)+x(9)$ $t_8=x(1)-x(9)$ $t_9=x(3)+x(11)$
$t_{10}=x(3)-x(11)$ $t_{14}=x(5)+x(13)$ $t_{12}=x(5)-x(13)$
$t_{13}=x(7)+x(15)$ $t_{14}=x(7)-x(15)$ $t_{15}=t_1+t_2$
$t_{16}=t_3+t_5$
$t_{17}=t_{15}+t_{16}$ $t_{18}=t_7+t_{11}$ $t_{19}=t_7-t_{11}$ $t_{20}=t_3+t_{13}$
$t_{21}=t_9-t_{13}$ $t_{22}=t_{18}+t_{20}$ $t_{23}=t_8+t_{14}$ $t_{24}=t_8-t_{14}$
$t_{25}=t_{10}+t_{12}$ $t_{26}=t_{12}-t_{10}$ $m_0=1\cdot(t_{17}+t_{22})$
$m_1=1\cdot(t_{17}-t_{22})$ $m_2=1\cdot(t_{15}-t_{16})$ $m_3=1\cdot(t_1-t_2)$
$m_4=1\cdot(x(0)-x(8))$ $m_5=(t_{19}-t_{21})\cos 2u$
$m_6=(t_4-t_6)\cos 2u$ $m_7=(t_{24}+t_{26})\cos 3u$
$m_8=(\cos u+\cos 3u)t_{24}$ $m_9=t_{26}(\cos 3u-\cos u)$
$m_{10}=\mathrm{j}(t_{18}-t_{20})$ $m_{11}=\mathrm{j}(t_3-t_5)$
$m_{12}=\mathrm{j}(x(4)-x(12))$ $m_{13}=\mathrm{j}(t_{19}+t_{21})\sin 2u$
$m_{14}=\mathrm{j}(t_4+t_6)\sin 2u$ $m_{15}=\mathrm{j}(t_{23}+t_{25})\sin 3u$
$m_{16}=\mathrm{j}t_{23}(\sin u-\sin 3u)$ $m_{17}=\mathrm{j}t_{26}(\sin u+\sin 3u)$
$s_1=m_3+m_5$ $s_2=m_3-m_5$ $s_3=m_{11}+m_{13}$
$s_4=m_{13}-m_{11}$

$s_5=m_1+m_6$ $s_6=m_4-m_6$ $s_7=m_S-m_7$
$s_8=m_9-m_7$ $s_9=s_5+s_7$ $s_{10}=s_5-s_7$ $s_{11}=s_6+s_8$
$s_{12}=s_6-s_8$ $s_{13}=m_{12}+m_{14}$ $s_{14}=m_{12}-m_{14}$

$s_{15}=m_{15}+m_{16}$ $s_{16}=m_{15}-m_{17}$ $s_{17}=s_{13}+s_{15}$
$s_{18}=s_{13}-s_{15}$ $s_{19}=s_{14}+s_{16}$ $s_{20}=s_{14}-s_{16}$ $X(0)=m_0$
$X(1)=s_9+s_{17}$ $X(2)=s_1+s_3$ $X(3)=s_{12}-s_{20}$
$X(4)=m_2+m_{10}$ $X(5)=s_{11}+s_{19}$ $X(6)=s_2+s_4$
$X(7)=s_{10}-s_{18}$ $X(8)=m_1$ $X(9)=s_{10}+s_{18}$
$X(10)=s_2-s_4$ $X(11)=s_{11}-s_{19}$ $X(12)=m_2-m_{10}$
$X(13)=s_{12}+s_{20}$ $X(14)=s_1-s_3$ $X(15)=s_9-s_{17}$

2. FÜGGELÉK

Gyors kis pontszámú *DFT* eljárások a Good-algoritmushoz.

Algoritmus 1. $N=2$, $M_A=2$, $M_N=0$, $A=2$
$X(0)=x(0)+x(1)$ $X(1)=x(0)-x(1)$

Algoritmus 2. $N=3$, $M_A=2$, $M_N=1$, $A=6$
$a_1=x(1)+x(2)$ $a_2=x(1)-x(2)$ $a_3=x(0)+a_1$ $m_0=0{,}5a_1$
$m_1=0{,}86603a_2$ $c_1=x(0)-m_0$ $X(0)=a_3$ $X(1)=c_1-\mathrm{j}m_1$
$X(2)=c_1+\mathrm{j}m_1$

Algoritmus 3. $N=4$, $M_A=4$, $M_N=0$, $A=8$
$a_1=x(0)+x(2)$ $a_2=x(1)+x(3)$ $a_3=x(0)\times -x(2)$
$a_4=x(1)-x(4)$ $X(0)=a_1+a_2$ $X(1)=a_2+\mathrm{j}a_3$
$X(2)=a_1-a_2$ $X(3)=a_2-\mathrm{j}a_3$

Algoritmus 4. $N=5$, $M_A=5$, $M_N=4$, $A=17$
$a_1=x(1)+x(4)$ $a_2=x(1)-x(4)$ $a_3=x(2)+x(3)$
$a_4=x(2)-x(3)$ $a_5=a_2+a_4$ $a_4=a_1-a_3$ $a_7=a_1+a_3$
$a_8=x(0)+a_7$ $m_0=0{,}95106a_5$ $m_1=1{,}53884a_2$
$m_2=0{,}36327a_6$ $m_4=0{,}25a_7$ $c_1=x(0)-m_4$
$c_2=c_1+m_3$
$c_3=c_1-m_3$ $c_4=m_0-m_2$ $c_5=m_1-m_0$ $X(0)=a_8$
$X(1)=c_2-\mathrm{j}c_4$ $X(2)=c_3-\mathrm{j}c_5$ $X(3)=c_3+\mathrm{j}c_5$
$X(4)=c_2+\mathrm{j}c_4$

Algoritmus 5. $N=7$, $M_A=8$, $M_N=8$, $A=36$
$a_1=x(1)+x(6)$ $a_2=x(1)-x(6)$ $a_3=x(2)+x(5)$
$a_4=x(2)-x(5)$ $a_5=x(3)+x(4)$ $a_5=x(3)-x(4)$
$a_7=a_1+a_3+a_5$ $a_8=a_1-a_5$ $a_9=a_5-a_3$ $a_{10}=a_3-a_1$
$a_{11}=a_2+a_4-a_5$ $a_{12}=a_2+a_6$ $a_{13}=-a_4-a_6$
$a_{14}=a_4-a_2$ $a_{15}=x(0)+a_7$ $m_0=0{,}16667a_7$
$m_1=0{,}79016a_8$ $m_2=0{,}05585a_9$ $m_3=0{,}7343a_{10}$
$m_4=0{,}44096a_{11}$ $m_5=0{,}34087a_{12}$ $m_6=0{,}53397a_{13}$
$m_7=0{,}87484a_{14}$ $c_1=x(0)-m_0$ $c_2=c_1+m_1+m_2$
$c_3=c_1-m_1-m_3$ $c_4=c_1-m_2+m_3$ $c_5=m_4+m_5-m_6$
$c_6=m_4-m_5-m_7$ $c_7=-m_4-m_6-m_7$ $X(0)=a_{15}$
$X(1)=c_2-\mathrm{j}c$ $X(2)=c_3-\mathrm{j}c_6$ $X(3)=c_4-\mathrm{j}c_7$
$X(4)=c_4+\mathrm{j}c_7$ $X(5)=c_3+\mathrm{j}c_6$ $X(6)=c_2+\mathrm{j}c_5$

Algoritmus 6. $N=8$, $M_A=8$, $M_N=2$, $A=26$
$a_1=x(0)+x(4)$ $a_2=x(2)+x(6)$ $a_3=x(1)+x(5)$
$a_4=x(1)-x(5)$ $a_5=x(3)+x(7)$ $a_6=x(3)-x(7)$
$a_7=a_1+a_2$ $a_5=a_3+a_5$ $m_0=0{,}7071(a_4-a_6)$
$m_1=0{,}7071(a_4+a_6)$ $b_1=a_7+a_8$ $b_2=a_7-a_8$
$b_3=a_1-a_2$
$b_4=x(0)-x(4)$ $b_5=a_3-a_5$ $b_6=x(2)-x(6)$
$c_1=b_4+m_0$
$c_2=b_1-m_0$ $c_3=\mathrm{j}(b_6+m_1)$ $c_4=\mathrm{j}(b_6-m_1)$ $X(0)=b_1$
$X(1)=c_1+c_3$ $X(2)=b_3+\mathrm{j}b_5$ $X(3)=c_2-c_4$ $X(4)=b_3$
$X(5)=c_2+c_4$ $X(6)=b_3-\mathrm{j}b_5$ $X(7)=c_1-c_3$

Algoritmus 7. $N=9$, $M_A=10$, $M_N=8$, $A=49$
$a_1=x(1)+x(8)$ $a_2=x(1)-x(8)$ $a_3=x(2)+x(7)$
$a_4=x(2)-x(7)$ $a_5=x(4)+x(5)$ $a_6=x(4)-x(5)$
$a_7=x(3)+x(6)$ $a_8=x(3)-x(6)$ $a_9=a_5-a_1$
$a_{10}=a_1-a_3$
$a_{11}=a_5-a_3$ $a_{12}=a_2-a_6$ $a_{13}=a_2+a_4$ $a_{14}=a_4-a_6$
$a_{15}=a_1+a_5+a_5$ $a_{16}=a_2-a_4+a_6$ $a_{17}=x(0)+a_7+a_{15}$
$m_0=0{,}1974a_9$ $m_1=0{,}56858a_{10}$ $m_2=0{,}37111a_{11}$
$m_3=0{,}54253a_{12}$ $m_4=0{,}10026a_{13}$ $m_5=0{,}56858a_{14}$
$m_6=0{,}5a_7$ $m_7=0{,}86603a_8$ $m_8=0{,}5a_{15}$
$m_9=0{,}86603a_{16}$

$c_1=x(0)-m_6$ $c_2=m_1-m_2$ $c_3=m_0+m_2$ $c_4=m_0+m_1$
$c_5=c_1+c_2-c_3$ $c_6=c_1+c_3+c_4$ $c_7=c_1-c_2-c_4$
$c_8=m_3-m_5$ $c_9=m_4-m_5$ $c_{10}=m_3-m_4$
$c_{11}=c_8+c_9+m_7$
$c_{12}=c_8+c_{10}-m_7$ $c_{13}=-c_9+c_{10}+m_7$
$c_{14}=x(0)+a_7-m_8$
$X(0)=c_8-\mathrm{j}c_{11}$ $X(2)=c_6-\mathrm{j}c_{12}$ $X(3)=c_{14}-\mathrm{j}m_9$
$X(4)=c_7-\mathrm{j}c_{13}$ $X(5)=c_7+\mathrm{j}c_{13}$ $X(6)=c_{14}+\mathrm{j}m_9$
$X(7)=c_6+\mathrm{j}c_{12}$ $X(8)=c_5+\mathrm{j}c_{11}$