Gyors Fourier-transzformáció és alkalmazásai grafikus processzorokon

Nagy Ádám
nagyadam888@gmail.com

Eötvös Loránd Tudományegyetem
Informatikai Kar
Programtervező Informatikus MSc

2013. június 20.

Tartalom

- A feladat rövid megfogalmazása.
- A megoldás formája.
- Ismert FFT algoritmusok.
- Alkalmazások és kihasználható specialitásaik.
- Néhány implementálási megfontolás GPU-ra.
- A FFT generátor.

Bővebben: Nagy Ádám – Gyors Fourier-transzformáció és alkalmazásai grafikus processzorokon (diplomamunka)

A feladat – Fourier-transzformáció

Különböző alkalmazások során ki szeretnénk számolni az $(x = (x[0], x[2], \ldots, x[N-1]) \in \mathbb{K}^N$, sorozat diszkrét Fourier-transzformáltját és/vagy annak inverzét:

$$\widehat{x}[k] = \sum_{n=0}^{N-1} x[n]\omega^{-nk} \quad \text{vagy} \quad x[n] = \sum_{k=0}^{N-1} \widehat{x}[k]\omega^{nk},$$

ahol $\omega = e^{2\pi i N^{-1}}$ N-edik primitív egységgyök és N **„nagy kettőhatvány”** $(N = 2^m,\ m = 20, 21, \ldots)$.

Az x-et tárolhatjuk szimpla vagy dupla lebegőpontos szám(pár)ként is (sőt ...).

A megoldás formája

A transzformációt elvégző eljárást egy OpenCL kódot generáló programmal készítem el. Ennek előnyei:

- A probléma rugalmas kezelése.
- Nem csak egy adott architektúrára tudunk viszonylag hatékony kódot generálni.
- Könnyű lesz a mérések alapján történő optimalizálás.
- A program megoldása során felhasznált struktúrák egy része a generátorba kerülhet.

Hátrány: Nem tudunk mindent a kézben tartani.
Habár ez általában a GPU programozásnál nem is lehetséges (architekturális szintű eszközök hiányossága).

FFT – Radix-2

A legegyszerűbb változata a Cooley és Tukey munkájában ismertetett FFT algoritmusoknak kettőhatvány hosszra a Radix-2, melynek az első rekurziós lépését felírhatjuk a periodikusságból kapható $\omega^{-k+2^{m-1}} = e^{\pi i}\omega^{-k} = -\omega^k$ összefüggést felhasználva

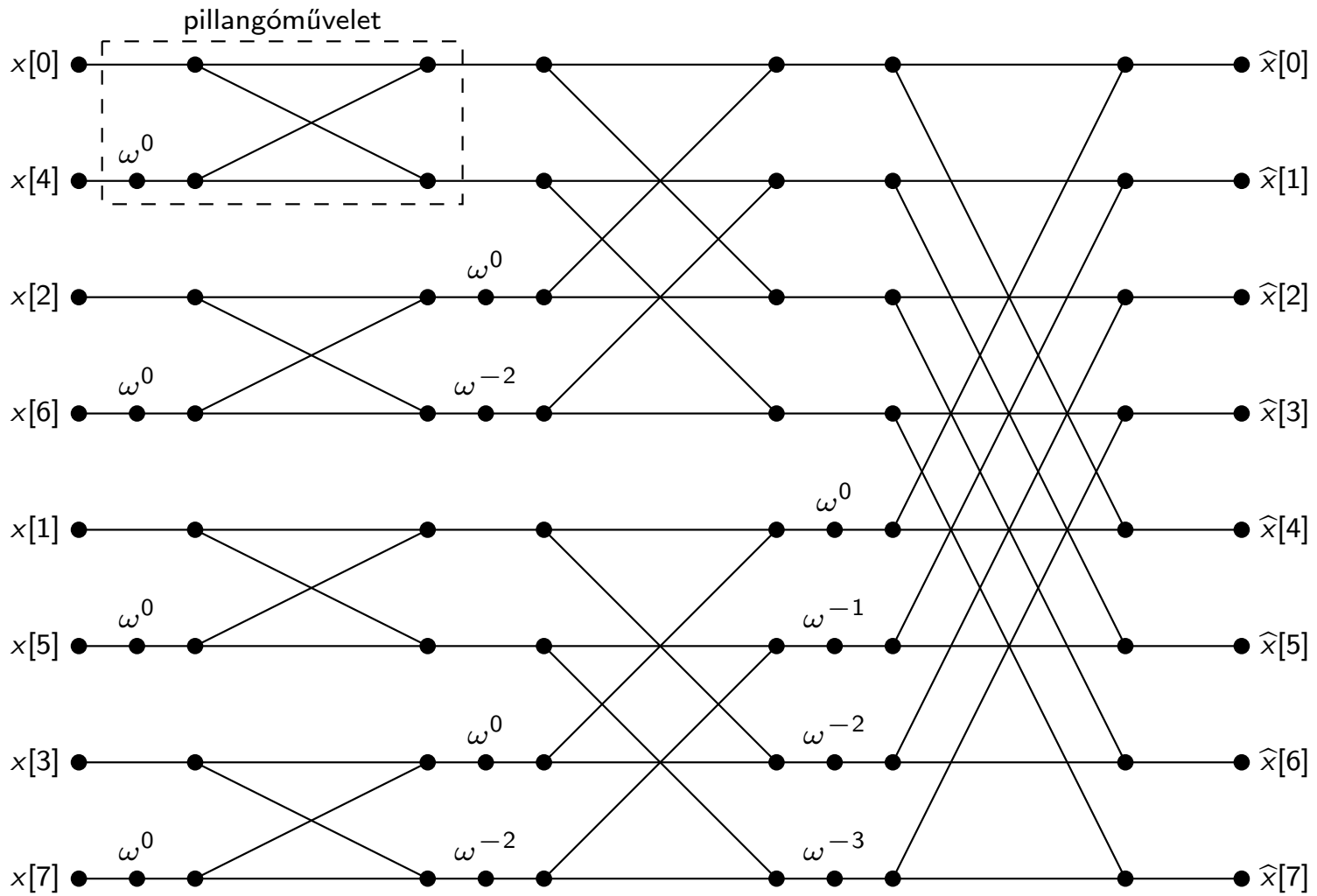
$$\widehat{x}[k] = \widehat{y_0}[k] + \omega^{-k}\widehat{y_1}[k] \quad \text{és} \quad \widehat{x}[k+2^{m-1}] = \widehat{y_0}[k] - \omega^{-k}\widehat{y_1}[k]$$

alakban, ahol

$$\widehat{y_0}[k] = \sum_{n=0}^{2^{m-1}-1} x[2n]\omega^{-kn} \quad \text{és} \quad \widehat{y_1}[k] = \sum_{n=0}^{2^{m-1}-1} x[2n+1]\omega^{-kn}$$

minden $k = 0, 1, \ldots, 2^{m-1} - 1$ esetén.

FFT – Radix-2 helyben $N = 8$-ra



FFT – Radix-4

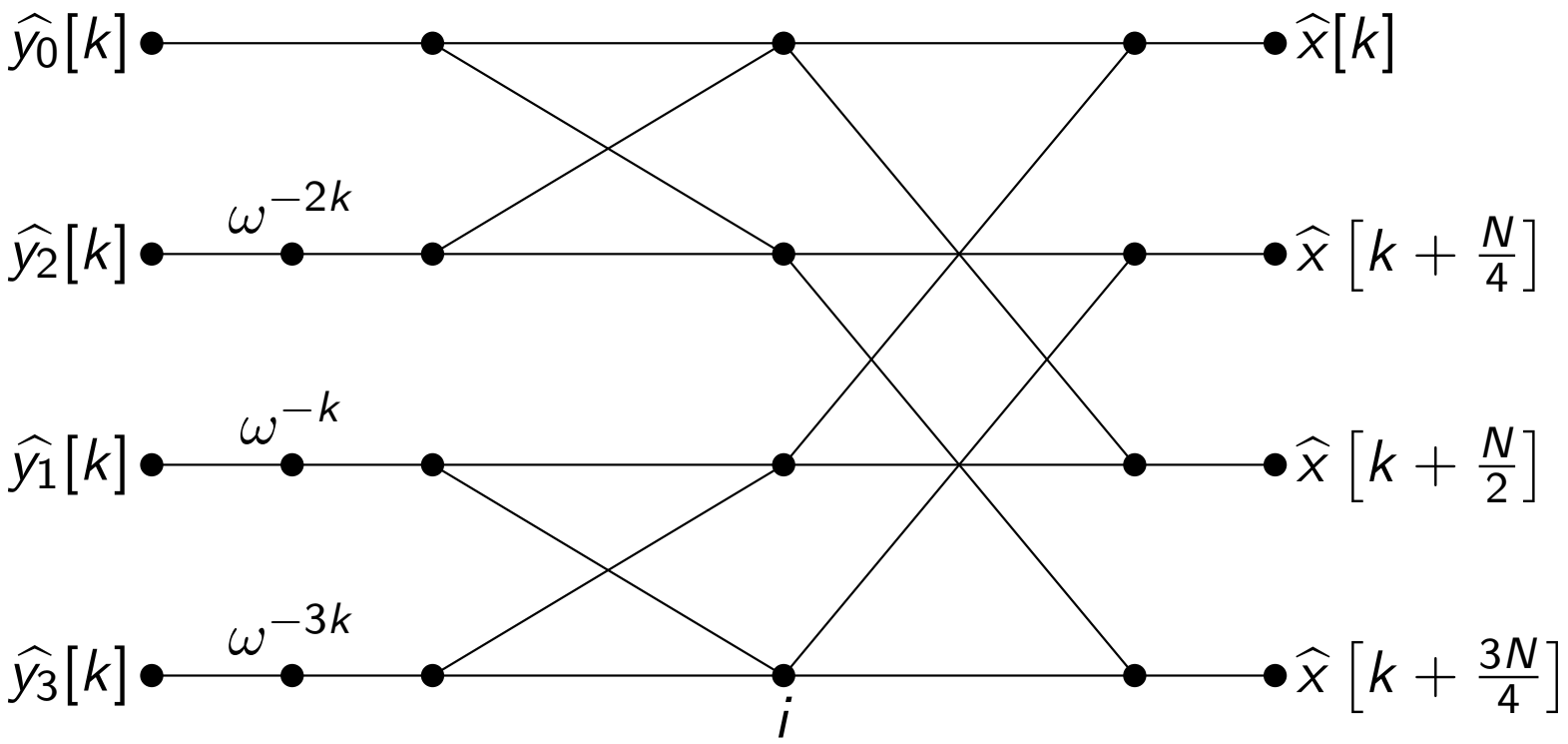
Most tegyük fel, hogy $N = 2^m$ és m páros. Ekkor felhasználva, hogy $\omega^{-(k+2^{m-2})} = -i\omega^{-k}$, $\omega^{-k(k+2^{m-1})} = -\omega^{-k}$ és $\omega^{-(k+3\cdot 2^{m-2})} = i\omega^{-k}$;

$$\begin{aligned}
\widehat{x}[k + 0 \cdot 2^{m-2}] &= \widehat{y_0}[k] + \omega^{-k}\widehat{y_1}[k] + \omega^{-2k}\widehat{y_2}[k] + \omega^{-3k}\widehat{y_3}[k] \\
\widehat{x}[k + 1 \cdot 2^{m-2}] &= \widehat{y_0}[k] - i\omega^{-k}\widehat{y_1}[k] - \omega^{-2k}\widehat{y_2}[k] + i\omega^{-3k}\widehat{y_3}[k] \\
\widehat{x}[k + 2 \cdot 2^{m-2}] &= \widehat{y_0}[k] - \omega^{-k}\widehat{y_1}[k] + \omega^{-2k}\widehat{y_2}[k] - \omega^{-3k}\widehat{y_3}[k] \\
\widehat{x}[k + 3 \cdot 2^{m-2}] &= \widehat{y_0}[k] + i\omega^{-k}\widehat{y_1}[k] - \omega^{-2k}\widehat{y_2}[k] - i\omega^{-3k}\widehat{y_3}[k]
\end{aligned},$$

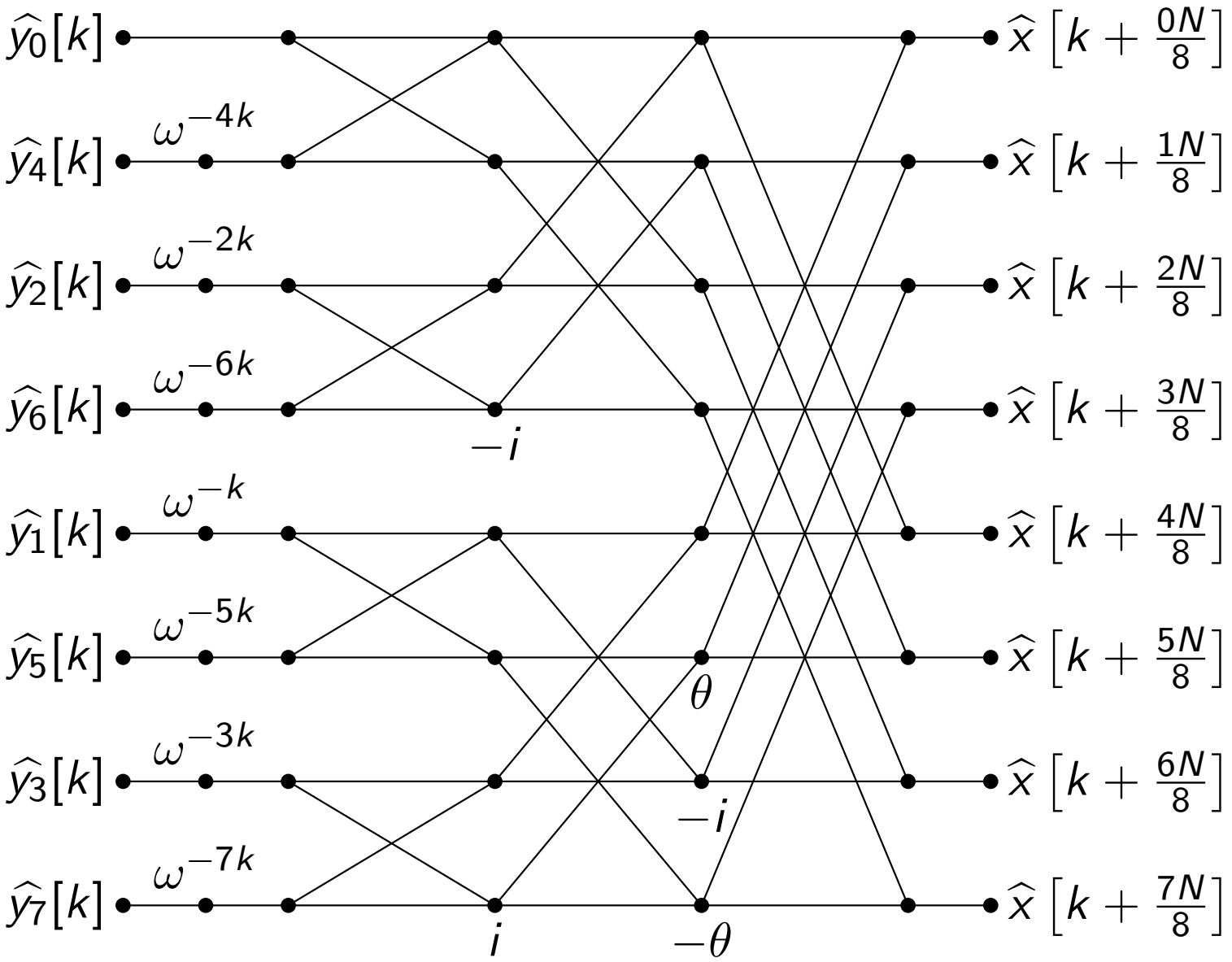
ahol

$$\widehat{y_j}[k] = \sum_{n=0}^{2^{m-2}-1} x[4n + j]\omega^{-4n} \quad (j = 0, 1, 2, 3; 0 \leq k < 2^{m-2}).$$

FFT – Radix-4 pillangóműveletе



FFT – Radix-8 pillangóműve



FFT – Valós műveleti igények R2, R4, R8

A valós műveleti igények (a negyedik egységgyökökkel való szorzás 2-2) a következőek szerint alakulnak.

- Radix-2:

$$5N \log_2 N - 10N + 16$$

- Radix-4:

$$\frac{17N}{4} \log_2 N - \frac{43N}{6} + \frac{32}{3}$$

- Radix-8:

$$\frac{49N}{12} \log_2 N - \frac{25N}{4} + 8$$

FFT – Split-Radix

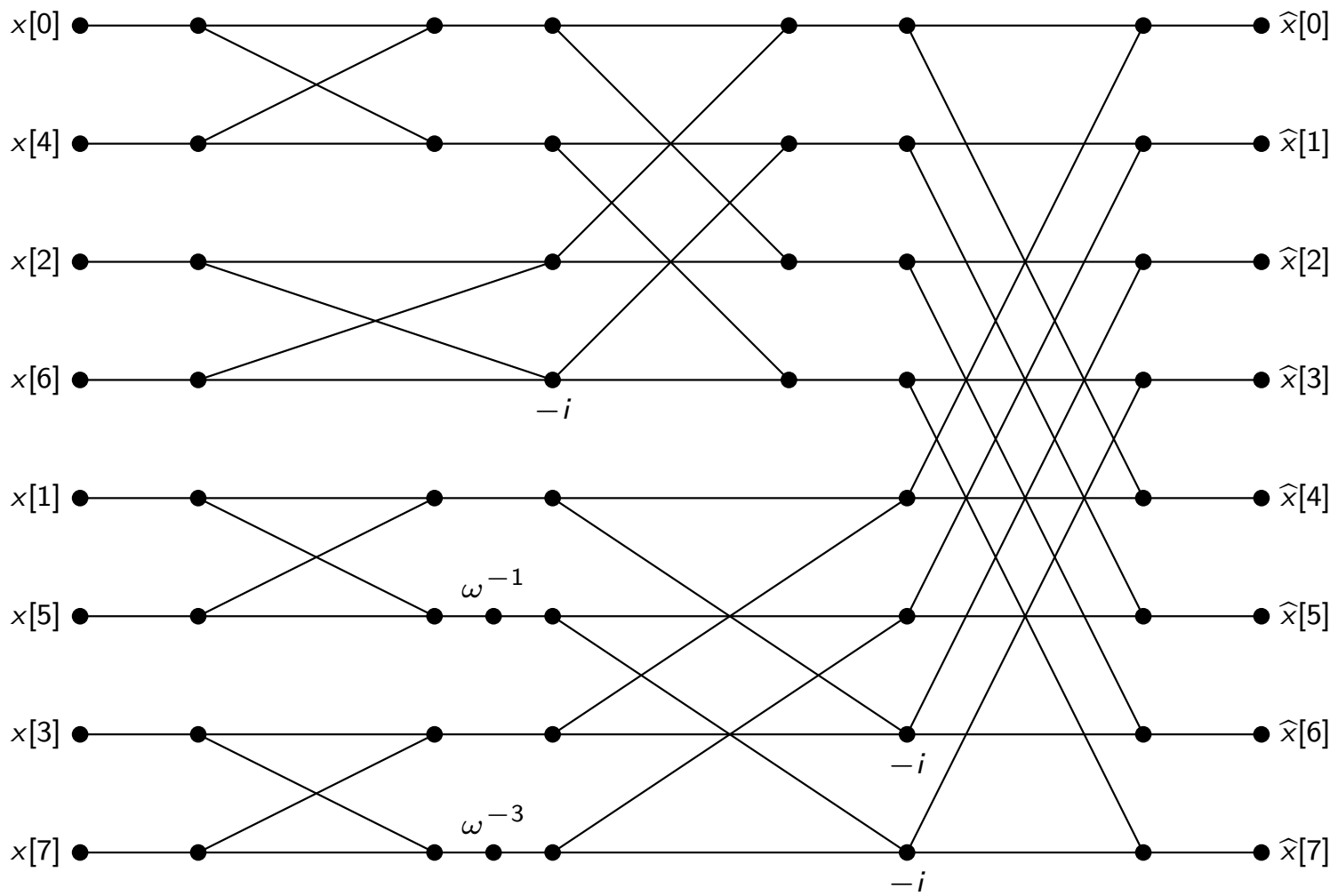
A műveletek száma tovább javítható Yavne ötlete alapján, ha a Radix-2 esetén csak a páratlan tagokat bontjuk fel a Radix-4-nek megfelelően.

$$\begin{aligned}
\widehat{x}[k+0\cdot 2^{m-2}] &= \widehat{y}[k]+(\omega^{-k}\widehat{\widehat{y_0}}[k]+\omega^{-3k}\widehat{\widehat{y_1}}[k])\\
\widehat{x}[k+1\cdot 2^{m-2}] &= \widehat{y}[k+2^{m-2}]-i(\omega^{-k}\widehat{\widehat{y_0}}[k]-\omega^{-3k}\widehat{\widehat{y_1}}[k])\\
\widehat{x}[k+2\cdot 2^{m-2}] &= \widehat{y}[k]-(\omega^{-k}\widehat{\widehat{y_0}}[k]+\omega^{-3k}\widehat{\widehat{y_1}}[k])\\
\widehat{x}[k+3\cdot 2^{m-2}] &= \widehat{y}[k+2^{m-2}]+i(\omega^{-k}\widehat{\widehat{y_0}}[k]-\omega^{-3k}\widehat{\widehat{y_1}}[k])
\end{aligned},$$

$$\widehat{y}[k] = \sum_{n=0}^{2^{m-1}-1} x[2n]\omega^{-2kn}$$

$$\widehat{\widehat{y_0}}[k] = \sum_{n=0}^{2^{m-2}-1} x[4n+1]\omega^{-4kn} \qquad \widehat{\widehat{y_1}}[k] = \sum_{n=0}^{2^{m-2}-1} x[4n+3]\omega^{-4kn}.$$

FFT – Split-Radix $N = 8$-ra



FFT – Split-Radix műveleti igénye és annak javítása

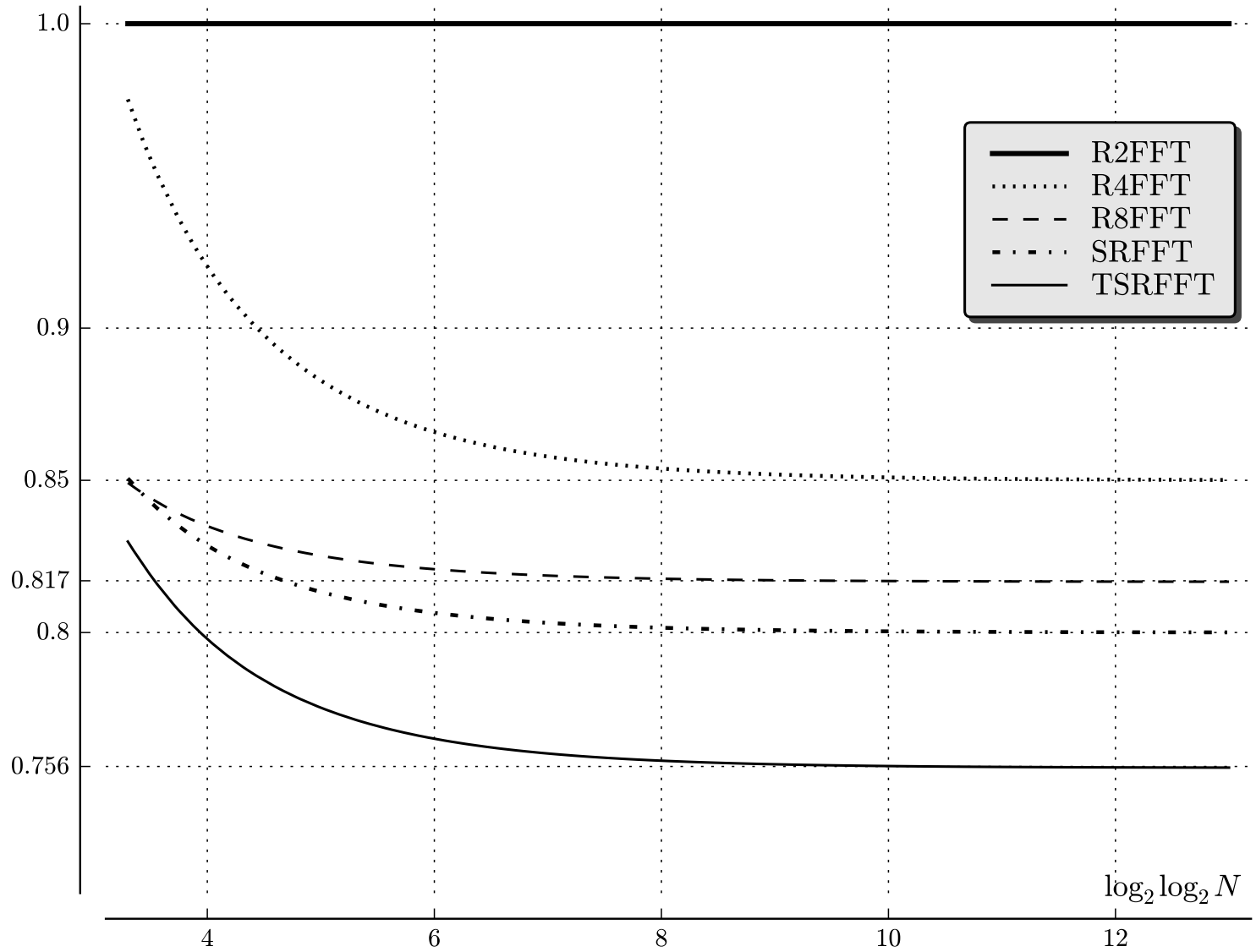
A Split-Radix műveleti igénye tovább javítható. Ezt a változatot Bernstein a cikkében Tangent FFT-nak nevezi.
A Split-Radix valós műveleti igénye

$$4N \log_2 N - 6N + 8,$$

amíg a javításáé

$$\frac{34N}{9} \log_2 N - \frac{124N}{27} - 2 \log_2 N - \left(\frac{2}{9} \log_2 N + \frac{16}{27} \right) (-1)^{\log_2 N} + 8.$$

FFT – Műveleti igények szemléletesen



FFT – Egységgyökök

A komplex egységgyökökkel való szorzást el lehet végezni 4 valós összeadással és 2 szorzással vagy 3 szorzással és 3 összeadással, habár ez utóbbi adatfüggőséget okoz.

Az egységgyököket tartalmazó táblák méretei:

	Összes	**(4/2) redukált**	**(3/3) redukált**
R2FFT	N	$N/4$	N
R4FFT	$3N/2$	$N/4$	N
R8FFT	$7N/4$	$N/4$	N
SRFFT	N	$N/4$	N
CSRFFT	$N/2$	$N/4$	$N/2$
TSRFFT	$3N/2$	$3N/2$	$3N/2$

FFT – Dimenziókra bontás

Legyen x egy $N = N_1 N_2$. Ekkor minden k-ra

$$\widehat{x}[k] = \sum_{n=0}^{N-1} x[n]\omega^{-kn} = \sum_{n_1=0}^{N_1-1}\sum_{n_2=0}^{N_2-1} x[n_2 N_1 + n_1]\omega^{(n_2 N_1+n_1)(k_1 N_2+k_2)} =$$

$$= \sum_{n_1=0}^{N_1-1} \omega^{-k_1 n_1 N_2}\left(\omega^{-k_2 n_1}\sum_{n_2=0}^{N_2-1} x[n_2 N_1 + n_1]\omega^{-k_2 n_2 N_1}\right)$$

összefüggés, ahol $0 \leq k_1, n_1 < N_1$ és $0 \leq k_2, n_2 < N_2$.
Szemléletesen egy N hosszú vektort egy N_1 sorral és N_2 oszloppal rendelkező mátrixként ábrázoltuk. Számolás:

1. Oszlopokra külön-külön N_2 darab N_1 méretű FFT.
2. Szorozzuk a mátrix elemeit a megfelelő egységgyökökkel.
3. Sorokra külön-külön N_1 darab N_2 méretű FFT.

FFT – Tisztán valós adatsorokra I.

„The shortest path between two truths in the real domain passes through the complex domain."
Jacques Salomon Hadamard

Ez az eset visszavezethető komplex FFT-re, ha szükséges.

- **Kettőt egyért**: Legyen x és y két N szerint periodikus valós adatsor és

$$z[n] = x[n] + iy[n], \quad (\overline{z[n]} = x[n] - iy[n]),$$

így

$$\widehat{x}[k] = \frac{\widehat{z}[k] + \overline{\widehat{z}[N-k]}}{2} \quad \text{és} \quad \widehat{y}[k] = i\frac{\widehat{z}[k] - \overline{\widehat{z}[N-k]}}{2}.$$

Az inverz transzformáció teljesen hasonlóan készíthető el, azaz, ha $\widehat{x}$ és $\widehat{y}$ két N szerint periodikus valós sorozat DFT-ja, akkor a $\widehat{z} = \widehat{x} + i\widehat{y}$ sor z iDFT-jából

$$x[n] = \Re(z[n]) \quad \text{és} \quad y[n] = \Im(z[n]) \quad (0 \leq n < N)$$

.

FFT – Tisztán valós adatsorokra II.

- **Egyet félért**: Legyen x egy $2N$ szerint periodikus valós sorozat és

$$z[n] = x[2n] + ix[2n+1] \quad (0 \leq n < N),$$

így

$$\widehat{x}[k] = \frac{1}{2}\left((1 + i\omega^{-k})\widehat{z}[k] + (1 - i\omega^{-k})\overline{\widehat{z}[N-k]}\right)$$

és $\widehat{x}[N+k] = \overline{\widehat{x}[k]}$ minden $0 \leq k < N$ esetében.

Az iFFT egy $\widehat{x}$ valós $2N$ szerint periodikus sorozat DFT-jából

$$\widehat{z}[k] = \Re(\widehat{x}[k]) + \Im(\widehat{x}[k]) = \widehat{x}[k]$$

választással

$$x[2n] = \Re(z[n]) \quad \text{és} \quad x[2n+1] = \Im(z[n]) \quad (0 \leq n < N).$$

Alkalmazás I. – iFFT sávszűrt adatra

Sok alkalmazás esetén a frekvenciatérbeli adatok csak egy része nem nulla.

Nulla elemek okozta optimalizáció lehetősége

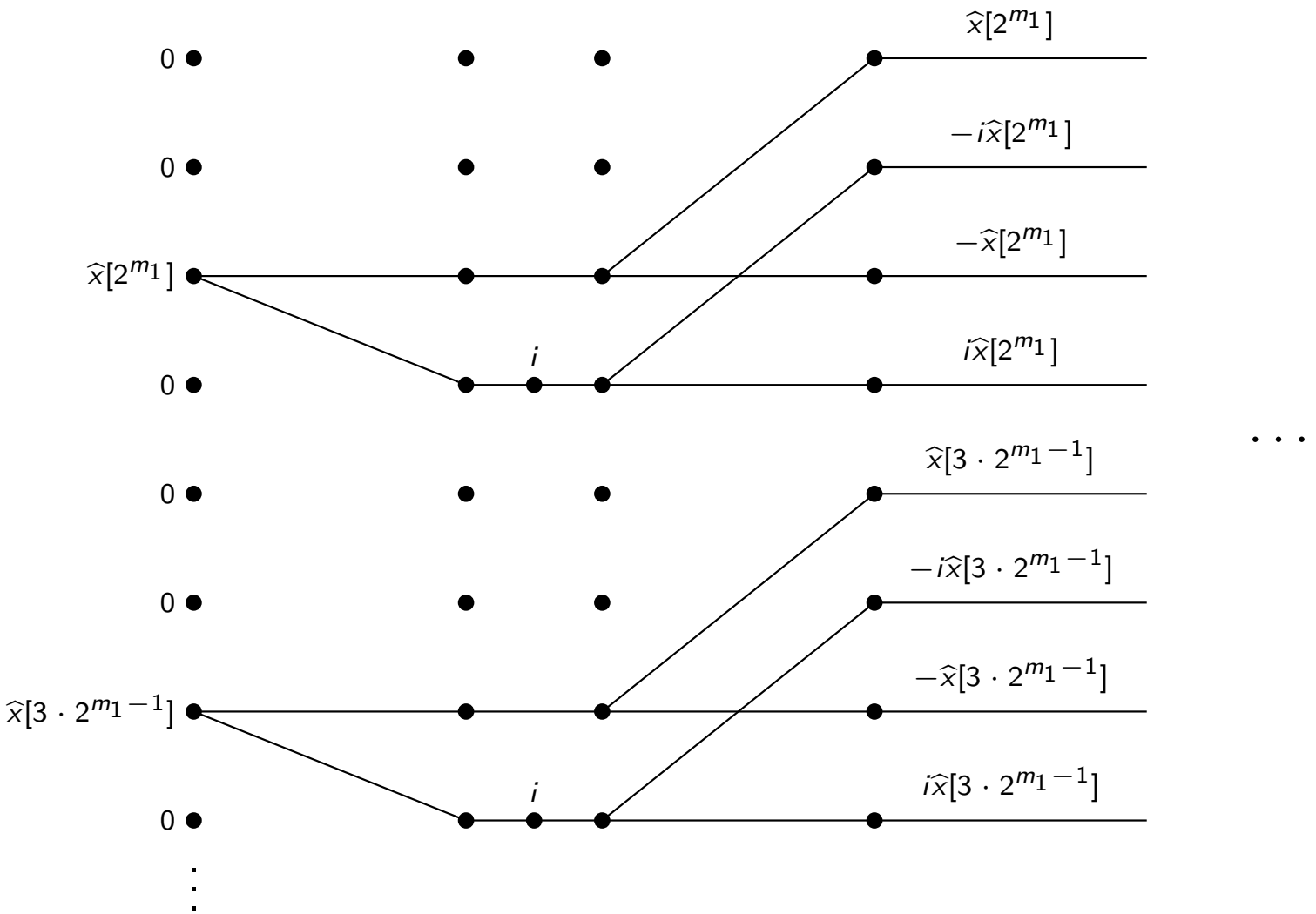
Tegyük fel, hogy a bemenő adatok csak egy $\{a, \ldots, b\}$ $(a < b)$ indextartományban nem nullák. Ha valamely μ természetes számra

$$\mu 2^{m_2} \leq a \quad \text{és} \quad b < (\mu + 1)2^{m_2} - 1 \quad (0 \leq \mu \leq 2^{m_1}),$$

akkor a rendezés után minden m_1 hosszú blokkban pontosan a $\mu + 1$-edik elem lesz nullától különböző.

Eredmény: csak a μ-edik szint után kell a transzformációt elvégezni.
Érdemes egy transzlációval a nemnulla elemeket a sor elejére tenni.

Alkalmazás I. – Nemnulla elemek „terjedése”



Alkalmazás II. – Nagy egészek szorzása

Schönhage–Strassen gyors-szorzó algoritmus. Röviden:

- Egy egész szám felírható egy B alapú számrendszerben, amely tekinthető egy polinom B helyen való kiértékelésének.

$$a = a_0 + a_1 B + a_2 B^2 + \cdots + a_{N-1} B^{N-1}$$

- A polinomok szorzása konvolúció, amely $\mathcal{O}(N^2)$ futásidő helyett 2 db FFT-vel egy elemenkénti szorzással és egy iFFT-vel $\mathcal{O}(N \log N)$ művelettel elvégezhető lenne.
- Figyelni kell, hogy a bemeneti sorok elemei (a szám jegyei) olyanok legyenek, hogy az a számolás során ne csorduljon túl.

Pontosság

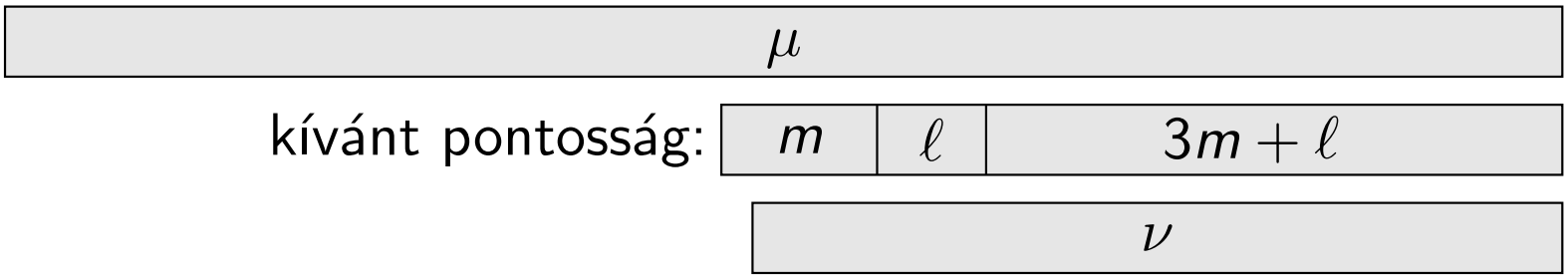
μ pontossággal számolva, $\mu \geq 4m + 2\ell$ és ℓ bites számjegyekkel a

$$\mu \geq 4m + 2\ell$$

feltételnek teljesülnie kell.

Alkalmazás II. – Optimalizálási lehetőségek

1. *Első szint elhagyása*, mivel a bemeneti vektorok felső része 0.
2. ℓ *pontosság használata az első két szinten.* Erre lehetőség van mivel szorzás csak ± 1-el és $\pm i$-vel történik.
3. *Adaptív pontosság használata*. Az eredmény kiszámolásánál, ha már nincs szükség a teljes pontosságra, akkor ne használjuk azt.
   Egy kedvező eset ν pontosságra való áttérésre:



Generálásnál figyelembe vett legfontosabb irányelvek

- A rendszertelen memóriaelérések minimalizálása (getter-scatter alkalmazható legyen).
- A feltételes elágazások, ciklusok és szinkronizáció minimalizálása.
- A konstans memória paraméterezhető használata.
- A felhasznált programmemória mérete paraméterezhető legyen.

A generátor működése

Eredmény: OpenCL kód, C függvények formájában.

1. Paraméterek megadása, beolvasása.
2. Paraméterek ellenőrzése.
3. Terv készítése.
4. Egységgyökök előállítása/beolvasása.
5. Kernelek generálása.
6. Platform és fejállományok generálása.
7. Esetleges iFFT generálása az FFT mellé.

Tekintsük át a paramétereket és velük a pontos működést.

Paraméterek – Az eredményre

(E1) *Adatok típusa:* `DATA=(C|I|R)`

(E2) *Adattárolás alaptípusa:* `FFTDATA=(float|double)`

(E3) *Adathossz:* `LOGN=([1-9][0-9]*)`
A minimális érték 8, a maximális jelen munkában 26.
(Ekkor legfeljebb 1 GB-os sorozataink lehetnek.)

(E4) *Elő- és utórendezés:* `BITREV=(Y|N)`

(E5) *Transzformáció:* `DIRECTION=(A|F|B)`

(E6) *Algoritmus azonosító:* `ID=(|[0-9a-zA-Z_]+)`

(E7) *Nemnulla elemek:*
`NONEZ=(|0*X0*|[0-9]+,[0,9]+|[1-9][0-9]*;[1-9][0-9]*)`

- `0*X0*`
- `[0-9]+,[0,9]+`
- `[1-9][0-9]*;[1-9][0-9]*`

(E8) *Kimenet skálázása:* `SCALING=(Y|N)`

(E9) *Bemeneti pontosság:* `INPRECISION=(0|[1-9][0-9]*)`

(E10) *Kimeneti pontosság:* `OUTPRECISION=(0|[1-9][0-9]*)`

Paraméterek – Az algoritmusra

(G1) *Algoritmus kiválasztása:* `ALGORITHM=(2|4|8|S|C|T)`
Megjegyzés: Mixed-Radix használata, ha LOGN nem megfelelő.

(G2) *Egységgyökök:* `TWIDLE=(YES|ONFLY|[0-9a-zA-Z._/]+)`

(G3) *Diemenziók:* `DIM=(0|[1-9][0-9]*(,[0-9]+)*)`

(G4) *Befejezési szint:* `STOPLEVEL=(0|[1-9][0-9]*)`

Paraméterek – A környezetre

(K1) *Konstans-memória használata:* `USECONSTANTS=(Y|N)`

(K2) *Kernel maximális magassága:* `KERNELHEIGHT=(6|[1-9][0-9]*)`
Az a FFT szerinti szint, amely egy SIMD kötegben számításra kerül egyszerre.

(K3) *Kernel szélessége:* `KERNELWIDTH=(16|[1-9][0-9]*)`
A itt megadott `KERNELWIDTH` szám azt jelöli, hogy egyszerre $2^{\texttt{KERNELWIDTH}}$ vektorelemre végezzük el a számításokat egyetlen kernelben. A privát memória mérette ettől a paramétertől függ.

(K4) *Kernel maximális mérete:* `KERNELSIZE=(0|[1-9][0-9]*)`
A kernel programhoz tartozó egységgyökök tárolására használt konstans memória maximális mérete..

(K5) *Kernelek maximális száma:* `KERNELNUM=(0|[1-9][0-9]*)`
Párhuzamosan futtatható kernelek száma.

Fejlesztési tervek

- További algoritmusok implementálása: SRFFT, más sima hossz.
- Grafikus felület, elemző.
- Dinamikus transzformáció.
- Pontosság növelése.
- Nagyobb bemenő adatok.

Kérdések.

Köszönöm a figyelmet!