Numerik I+II
Wintersemester 2009/10

Martin Arnold
Martin–Luther–Universität Halle–Wittenberg
Naturwissenschaftliche Fakultät III
Institut für Mathematik

21. Dezember 2009

Bemerkung 7.32 (Schnelle Fourier–Transformation)
engl.: Fast Fourier Transform(ation) (FFT)

Problem: Standard–Algorithmus zur Auswertung von $\mathcal{F}_N$ oder $\mathcal{F}_N^{-1}$ würde $\mathcal{O}(N^2)$ Rechenoperationen erfordern (Matrix–Vektor–Multiplikation).

Cooley–Tuckey (1965): Sei $N = 2M$ gerade und $\omega = \mathrm{e}^{i\frac{2\pi}{N}}$ oder $\omega = \mathrm{e}^{-i\frac{2\pi}{N}}$.
Dann gilt für

$$\alpha_j = \sum_{k=0}^{N-1} f_k \omega^{kj}\,, \quad (\, j = 0, 1, \ldots, N-1\,)$$

$$\alpha_{2l} = \sum_{k=0}^{M-1} g_k \xi^{kl}\,, \qquad \alpha_{2l+1} = \sum_{k=0}^{M-1} h_k \xi^{kl}\,, \quad (\, l = 0, 1, \ldots, M-1\,)$$

mit $M := N/2$, $\xi := \omega^2$ und

$$g_k := f_k + f_{k+M}\,, \qquad h_k := (f_k - f_{k+M})\,\omega^k\,.$$

Mit $2M$ Additionen und $2M$ Multiplikationen ($\omega^k \to \omega^{k+1} = \omega \cdot \omega^k$, $h_k = (\ldots) \cdot \omega^k$) wird die Berechnung von N Summen der Länge N zurückgeführt auf $2M = N$ Summen der Länge $M = N/2$.

Beweis:

$$\alpha_{2l} = \sum_{k=0}^{N-1} f_k \omega^{k\cdot 2l} = \sum_{k=0}^{\frac{N}{2}-1} \left(f_k \omega^{2kl} + f_{k+\frac{N}{2}} \omega^{2(k+\frac{N}{2})l} \right) = \sum_{k=0}^{M-1} (f_k + f_{k+M})\,\omega^{2kl}$$

$$\alpha_{2l+1} = \sum_{k=0}^{N-1} f_k \omega^{(2l+1)k} = \sum_{k=0}^{\frac{N}{2}-1} \left(f_k \omega^{2kl+k} + f_{k+\frac{N}{2}} \omega^{(2l+1)(k+\frac{N}{2})} \right) = \sum_{k=0}^{M-1} (f_k - f_{k+\frac{N}{2}})\,\omega^k \cdot \omega^{2kl}$$

Rekursive Anwendung besonders einfach für $N = 2^p$ $\Rightarrow$ Aufwand zur Berechnung von $\alpha_0, \alpha_1, \ldots, \alpha_{N-1}$ (Analyse oder Synthese) beträgt $2N \log_2 N$ Multiplikationen.

Speicherschema: Wünschenswert ist die Überspeicherung von (f_k) durch (α_j) ohne aufwändigen Tausch von Komponenten, jedoch Beibehaltung der einfachen Struktur der Terme für α_{2l} und α_{2l+1}.

Problem: Einfaches Überspeichern tauscht Komponenten

k	dual	$N=8 \to M=4$	$N=4 \to M=2$	dual
0	000	0	0	000
1	001	2	4	100
2	010	4	2	010
3	011	6	6	110
4	100	1	1	001
5	101	3	5	101
6	110	5	3	011
7	111	7	7	111

Alternative: „bit reversal“, Permutation $\sigma : \{0,1,\ldots,N-1\} \to \{0,1,\ldots,N-1\}$,

$$\sigma\Big(\sum_{j=0}^{N-1} a_j 2^j\Big) := \sum_{j=0}^{N-1} a_{N-j} 2^j \quad \text{mit} \quad a_0, a_1, \ldots, a_{N-1} \in \{0,1\}\,.$$

Einfache Implementierung durch Bit–Manipulationen.

Algorithmus 7.33 (Schnelle Fourier–Transformation)

Sei $N = 2^p$ und $\omega = \mathrm{e}^{i\frac{2\pi}{N}}$ oder $\omega = \mathrm{e}^{-i\frac{2\pi}{N}}$.

Eingabe: $f_0, f_1, \ldots, f_{N-1} \in \mathbb{C}$

Ausgabe: $\alpha_0, \alpha_1, \ldots, \alpha_{N-1} \in \mathbb{C}$ mit $\alpha_j := \sum_{k=0}^{N-1} f_k \omega^{kj}$.

```
N_red := N ;  z := ω
while N_red > 1 do
    M_red := N_red/2
    for j = 0 : (N/N_red) − 1
        l := jN_red
        for k = 0 : M_red − 1
            a := f_{l+k} + f_{l+k+M_red}
            f_{l+k+M_red} := (f_{l+k} − f_{l+k+M_red}) z^k
            f_{l+k} := a
    N_red := M_red ;  z := z^2
for k = 0 : N − 1
    α_{σ(k)} := f_k
```