11. Trigonometrische Interpolation

11.1 Problemstellung

Man unterscheidet zwei Arten trigonometrischer Interpolationen:

Typ ① Reelle Form

Zu Stützstellen (t_k, f_k), $k = 0,1,\dots,n-1$ mit $t_k = k \cdot \frac{2\pi}{n}$, $f_k \in \mathbb{R}$ wird eine interpolierende trigonometrische Summe

$$Q(t) = \begin{cases} \frac{\alpha_0}{2} + \sum_{j=1}^{m} [\alpha_j \cos(jt) + \beta_j \sin(jt)] & : n = 2m+1 \\ \frac{\alpha_0}{2} + \sum_{j=1}^{m-1} [\alpha_j \cos(jt) + \beta_j \sin(jt)] + \frac{\alpha_m}{2} \cos(mt), & \text{für } n = 2m \end{cases} \tag{11.1}$$

mit $\alpha_j, \beta_j \in \mathbb{R}$ gesucht.

Typ ② Komplexe Form

Zu Stützstellen (t_k, f_k), $k = 0,1,\dots,n-1$ mit $t_k = k \cdot \frac{2\pi}{n}$, $f_k \in \mathbb{C}$ wird ein

interpolierendes trigonometisches Polynom

$$P(t) = \sum_{j=0}^{n-1} \gamma_j \, e^{ijt} \qquad (i = \sqrt{-1}) \tag{11.2}$$

mit $\gamma_j \in \mathbb{C}$ gesucht.

- Bei beiden Aufgaben geht es darum, eine periodische Funktion f mit (bekannter) Periode $T > 0$ zu interpolieren.

Durch die Transformation $\tilde{t} := \frac{2\pi}{T} t = \omega t$, $\boxed{\omega := \frac{2\pi}{T}}$ wird das Problem auf Periodenlänge $\tilde{T} = 2\pi$ normiert.

Wegen:

$$\begin{aligned} e^{ikt} &= \cos(kt) + i \sin(kt) \\ \cos(kt) &= \tfrac{1}{2}\left(e^{ikt} + e^{-ikt}\right) \\ \sin(kt) &= \tfrac{1}{2i}\left(e^{ikt} - e^{-ikt}\right) \end{aligned} \tag{11.3}$$

sind die beiden Problemstellungen nicht unabhängig voneinander:

Satz (11.4)

Für reelle f_k lässt sich zu jeder Interpolierenden vom Typ ① eine solche vom Typ ② konstruieren und umgekehrt.

Umrechnung:

$$\begin{cases} \alpha_0 = 2\gamma_0 \\ \alpha_j = \gamma_j + \gamma_{n-j}, \quad j = 1, \dots, m \\ \beta_j = i(\gamma_j - \gamma_{n-j}), \quad j = 1, \dots, m-1 \ (m) \end{cases} \tag{11.5}$$

$$\begin{cases} \gamma_0 = \frac{\alpha_0}{2} \\ \gamma_j = \frac{1}{2}(\alpha_j - i\beta_j), \quad j = 1, \dots, m-1, (m) \\ \gamma_{n-j} = \frac{1}{2}(\alpha_j + i\beta_j), \quad j = 1, \dots, m-1, (m) \\ \gamma_m = \frac{\alpha_m}{2} \end{cases} \tag{11.6}$$

Beweis: Werte $P(t)$ und $Q(t)$ auf dem Gitter $t_k = k \cdot \frac{2\pi}{n}$ aus:

$$P(t_k) = \sum_{j=0}^{n-1} \gamma_j e^{ijt_k}$$

$$Q(t_k) = \begin{cases} \alpha_0/2 + \sum [\alpha_j \cos(jt_k) + \beta_j \sin(jt_k)] \\ \dots + \alpha_m/2 \cos(mt_k) \end{cases}$$

Nun gilt

$$e^{-ijt_k} = e^{-ijk\frac{2\pi}{n}} = e^{i(n-j)t_k}$$

und damit

$$\cos(jt_k) = \frac{1}{2}\left(e^{ijt_k} + e^{i(n-j)t_k}\right)$$
$$\sin(jt_k) = \frac{1}{2i}\left(e^{ijt_k} - e^{i(n-j)t_k}\right)$$

Dies in $Q(t_k)$ eingesetzt und umgeformt liefert $P(t_k)$ mit γ_j gemäß (11.6).

Beachte: Aus beliebigen komplexen Daten $\gamma_0, \ldots, \gamma_{n-1}$ ergeben sich nur im Fall $\gamma_{n-j} = \overline{\gamma_j}$ auch reelle α_j, β_j ! ▲

Anmerkungen:

a) Im Allg. gilt $P(t) \neq Q(t)$.

b) Selbst für reelle f_k ist $P(t)$ im Allg. nicht reell.

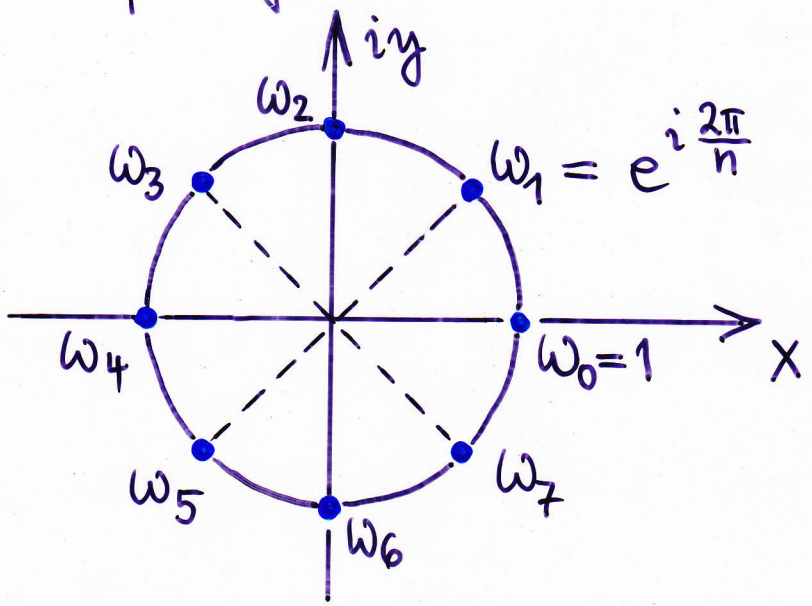
c) $Q(t)$ enthält die „Frequenzen" $0, \ldots, m$,
$P(t)$ enthält die „Frequenzen" $0, \ldots, n-1$.

Satz (11.7) (Existenz u. Eindeutigkeit)

Zu Stützstellen (t_k, f_k), $k = 0, 1, \ldots, n-1$, $t_k = k \cdot \frac{2\pi}{n}$, $f_k \in \mathbb{C}$ existiert ein eindeutig bestimmtes interpolierendes trigonom. Polynom $P(t) = \sum_{j=0}^{n-1} \gamma_j e^{ijt}$. Die Koeffizienten γ_j sind gegeben durch $\gamma_j = \frac{1}{n} \sum_{k=0}^{n-1} f_k e^{-ijt_k}$.

Beweis: Setze $\omega := e^{it}$.

$\Rightarrow$ $\omega_k := e^{it_k}$, $k = 0, 1, .., n-1$ sind die n-ten Einheitswurzeln. Insbesondere gilt $\omega_j \neq \omega_k$ für $j \neq k$.



$(n=8)$

Das transformierte Problem lautet:

$$\tilde{P}(\omega) = P(t) = \sum_{j=0}^{n-1} \gamma_j e^{ijt} = \sum_{j=0}^{n-1} \gamma_j \omega^j$$

Die γ_j sind so zu bestimmen, dass $\tilde{P}(\omega_k) = f_k$, $k = 0, 1, \ldots, n-1$.

$\longrightarrow$ Problem der Polynominterpolation!

Aus Satz (3.1) folgt die Existenz und Eindeutigkeit!

Lösungsdarstellung: Die γ_j seien wie im Satz (11.7) definiert. Wir prüfen die Interpolationsbedingungen:

$$P(t_\ell) = \sum_{j=0}^{n-1} \left(\frac{1}{n} \sum_{k=0}^{n-1} f_k e^{-ijt_k} \right) e^{ijt_\ell}$$

$$= \frac{1}{n} \sum_{k=0}^{n-1} f_k \left(\sum_{j=0}^{n-1} e^{ij(\ell-k)\frac{2\pi}{n}} \right)$$

Nun ist $\omega_{\ell-k} = e^{i(\ell-k)\frac{2\pi}{n}}$ eine n-te Einheitswurzel $\Longrightarrow$

$$0 = 1 - \omega_{\ell-k}^n = (1 - \omega_{\ell-k}) \left[\sum_{j=0}^{n-1} \omega_{\ell-k}^j \right]$$

Für $\ell \neq k \bmod n$ ist $\omega_{\ell-k} \neq 1$. Damit:

$$\sum_{j=0}^{n-1} e^{ij(\ell-k)\frac{2\pi}{n}} = \sum_{j=0}^{n-1} \omega_{\ell-k}^j = \begin{cases} 0, & \ell \neq k \bmod n \\ n, & \ell = k \bmod n \end{cases}$$

$\Longrightarrow P(t_\ell) = f_\ell$. ▲

Folgerung (11.8)

Auch das reelle trigonom. Interpolationsproblem (Typ ①) ist eindeutig lösbar und man erhält mit (11.7) und (11.5):

$$\alpha_j = \frac{2}{n} \sum_{k=0}^{n-1} f_k \cos\left(jk \frac{2\pi}{n}\right)$$
$$\beta_j = \frac{2}{n} \sum_{k=0}^{n-1} f_k \sin\left(jk \frac{2\pi}{n}\right) \qquad (11.9)$$

11.2 Fourier-Reihen

Wir betrachten den linearen Raum der stetigen, 2π-periodischen Funktionen

$$C_{2\pi} := \{ f \in C(\mathbb{R}) \mid \forall t \in \mathbb{R}: f(t+2\pi) = f(t) \} \qquad (11.10)$$

mit dem Skalarprodukt

$$\langle f, g \rangle := \frac{1}{\pi} \int_0^{2\pi} f(t)\, g(t)\, dt \qquad (11.11)$$

Satz (11.12)

Die Funktionen $\frac{1}{\sqrt{2}}$, $\cos t$, $\sin t$, ..., $\cos(mt)$, $\sin(mt)$ bilden ein ONS in $C_{2\pi}$; sie sind also

eine ONB von

$$\mathcal{D}_m := \operatorname{Spann}\left\{\frac{1}{\sqrt{2}}, \cos t, \ldots, \sin(mt)\right\}.$$

Die Bestapproximation von $f \in C_{2\pi}$ aus $\mathcal{D}_m$ bzgl. $\|.\|_2$ ist gegeben durch

$$(S_m f)(t) = \frac{a_0}{2} + \sum_{j=1}^{m} \left[a_j \cos(jt) + b_j \sin(jt)\right],$$

$$a_j = \frac{1}{\pi} \int_0^{2\pi} f(t) \cos(jt)\, dt, \quad b_j = \frac{1}{\pi} \int_0^{2\pi} f(t) \sin(jt)\, dt.$$

Beweis: Die Orthogonalität zeigt man direkt durch „Ausrechnen" (Übungsaufgabe!)
Die Darstellung der Bestapproximation erfolgt dann mittels Übungsaufgabe 19. ▲

Beachte: Die Fourier-Koeffizienten a_j, b_j sind unabhängig von m.

Für $m \to \infty$ erhält man aus (11.12) die Fourier-Reihe (Fourier-Entwicklung) einer Funktion $f \in C_{2\pi}$:

$$f(t) \sim \frac{a_0}{2} + \sum_{j=1}^{\infty} \left[a_j \cos(jt) + b_j \sin(jt)\right] \qquad (11.13)$$

mit a_j, b_j wie in (11.12).

Satz (11.14) (Kvgz. im quadratischen Mittel)

Für $f \in C_{2\pi}$ gilt: $\| f - S_m f \|_2 \to 0 \ (m \to \infty)$

Beweis: Königsberger: Analysis I, Abschn. 16.7

Aus (11.14) kann nicht auf die punktweise Kvgz. von $S_m f$ geschlossen werden! Auf du Bois-Reymond (1876) geht ein Beispiel einer Fkt. $f \in C_{2\pi}$ zurück, deren Fourier-Reihe divergiert.

Definition (11.15)

Eine Funktion $f: \mathbb{R} \to \mathbb{R}$ heißt stkw. glatt, falls

a) f in jedem kompakten Intervall $[a,b]$ bis auf endlich viele Ausnahmepunkte stetig diffb. ist,

b) in jedem Ausnahmepunkt τ die einseitigen Grenzwerte $f(\tau^-)$, $f(\tau^+)$, $f'(\tau^-)$, $f'(\tau^+)$ existieren und

c) dort $f(\tau) = \frac{1}{2}\left(f(\tau^-) + f(\tau^+)\right)$ gilt.

Satz (11.16) (Konvergenz)

a) Ist $f : \mathbb{R} \to \mathbb{R}$ 2π-periodisch und stkw. glatt, so kvgt. die Fourier-Reihe (11.13) in jedem Pkt. t gegen $f(t)$. Die Kvgz. ist in kompakten Intervallen ohne Ausnahmepkte. gleichmäßig.

b) Ist $f \in C^r(\mathbb{R}) \cap C_{2\pi}$ und ist $f^{(r)}$ stkw. glatt, so gilt für $j \to \infty$:

$$a_j = O\left(\frac{1}{j^{r+2}}\right), \quad b_j = O\left(\frac{1}{j^{r+2}}\right)$$

Berechnung der Fourier-Koeffizienten

Zur Berechnung von

$$\left.\begin{aligned} a_j &= \frac{1}{\pi}\int_0^{2\pi} f(t)\cos(jt)\,dt \\ b_j &= \frac{1}{\pi}\int_0^{2\pi} f(t)\sin(jt)\,dt \end{aligned}\right\} \quad j = 0,1,2,\ldots$$

werden die Integrale durch Trapezsummen approximiert:

$$t_k := k\frac{2\pi}{n}, \quad k = 0,1,\ldots,n, \quad h := \frac{2\pi}{n}, \quad f_k := f(t_k)$$

$\Longrightarrow$

$$a_j \approx \frac{1}{\pi}\frac{2\pi}{n}\left\{\frac{f_0}{2} + \sum_{k=1}^{n-1} f_k \cos(j t_k) + \frac{f_n}{2}\right\}$$

$$= \frac{2}{n}\sum_{k=0}^{n-1} f_k \cos\left(j \cdot k \frac{2\pi}{n}\right) =: \alpha_j$$

(→ (11.9))

$$b_j \approx \frac{1}{\pi}\frac{2\pi}{n}\left\{0 + \sum_{k=1}^{n-1} f_k \sin(j t_k) + 0\right\}$$

$$= \frac{2}{n}\sum_{k=0}^{n-1} f_k \sin\left(j \cdot k \frac{2\pi}{n}\right) =: \beta_j$$

Die durch trigonom. Interpolation berechneten Koeffizienten α_j, β_j sind damit Näherungen für die Fourier-Koeffizienten von $f(x)$.

Die Approximationen sind allerdings nur für Indizes $j \lesssim n/2$ brauchbar. Die α_j, β_j sind periodisch ($\alpha_j = \alpha_{j+n}$, $\beta_j = \beta_{j+n}$), während die a_j, b_j Nullfolgen bilden!

→ Abminderungsfaktoren

11.3 Algorithmus von Goertzel u. Reinsch

Der Algorithmus kombiniert die rekursive Berechnung der trigonometrischen Funktionen

$$c_k := \cos(kt), \quad s_k := \sin(kt) \tag{11.17}$$

über die Rekursion

$$p_k = 2\cos t \cdot p_{k-1} - p_{k-2}$$

(vgl. Übungsaufg. 3) in der stabilisierten Form nach C. Reinsch mit der Idee der adjungierten Summation (Algorithmus von Clenshaw; vgl. Übungsaufg. 11) zur Auswertung von

$$C_n := \sum_{k=0}^{n-1} f_k c_k, \quad S_n := \sum_{k=0}^{n-1} f_k s_k \tag{11.18}$$

Man erhält damit den folgenden Algorithmus:

Falls $\cos t > 0$: $\lambda := -4\sin^2\frac{t}{2}$, $\nu := 1$,

sonst : $\lambda := 4\cos^2\frac{t}{2}$, $\nu := -1$,

$$u_n := 0\,, \quad D_{n-1} := f_{n-1}$$

für $k = n-2, n-3, \ldots, 0$:

$$\begin{aligned} u_{k+1} &:= D_{k+1} + \nu \cdot u_{k+2} \\ D_k &:= f_k + \lambda \cdot u_{k+1} + \nu \cdot D_{k+1} \end{aligned}$$

$$c_n := D_0 - \frac{\lambda}{2} \cdot u_1\,, \quad s_n := u_1 \cdot \sin t \qquad (11.19)$$

Zur Berechnung sämtlicher Fourier-Koeffizienten α_j, β_j, $j = 0, 1, \ldots, n-1$ benötigt der obige Algorithmus $O(n^2)$ element. Operationen.

11.4 FFT

Die „schnelle" Fourier-Transformation (Fast Fourier Transform: FFT) geht auf eine Idee von Cooley u. Tukey (1965) zurück. Hiermit lassen sich sämtliche Fourier-Koeffiz. mit nur $O(n \cdot \log_2 n)$ berechnen:

n	$n \cdot \log_2 n$	n^2
10^3	$\sim 9.9 \cdot 10^3$	10^6
10^4	$\sim 1.3 \cdot 10^5$	10^8

Wir verwenden die komplexe Darstellung:

$$\gamma_j = \frac{1}{n} \sum_{k=0}^{n-1} f_k \, e^{-ijk\frac{2\pi}{n}}, \qquad j=0,\ldots,n-1, \tag{11.20}$$

und setzen voraus, daß n eine Zweierpotenz ist:

$$n = 2^r, \quad r \in \mathbb{N}. \tag{11.21}$$

Die Summe in (11.20) wird nun in zwei Summen halber Länge aufgeteilt. Dazu sei $m := n/2$. Wir sortieren nach geraden und ungeraden Indizes:

$$\begin{aligned}
\gamma_j &= \frac{1}{n}\left[\sum_0^{m-1} f_{2k}\, e^{-ij(2k)\frac{2\pi}{n}} + \sum_0^{m-1} f_{2k+1}\, e^{-ij(2k+1)\frac{2\pi}{n}}\right] \\
&= \left(\frac{1}{n}\sum_{k=0}^{m-1} f_{2k}\, e^{-ijk\frac{2\pi}{m}}\right) + \\
&\quad + e^{-ij\frac{\pi}{m}}\left(\frac{1}{n}\sum_{k=0}^{m-1} f_{2k+1}\, e^{-ijk\frac{2\pi}{m}}\right) \\
&=: g_j + e^{-ij\frac{\pi}{m}}\, u_j, \quad j=0,1,\ldots,n-1
\end{aligned}$$

Man braucht die g_j, u_j aber nur für $j=0,\ldots,m-1$ auszuwerten, da wegen der Periodizität:

$$g_{j+m} = g_j, \quad u_{j+m} = u_j, \quad j=0,1,\ldots,m-1$$

Reduktionsschritt:

Statt der y_j, $j = 0, 1, \ldots, n-1$, nach (11.20) berechnet man mit $m := n/2$:

für $j = 0, 1, \ldots, m-1$

$$
\begin{aligned}
g_j &:= \frac{1}{n} \sum_{k=0}^{m-1} f_{2k}\, e^{-ijk\frac{2\pi}{m}} \\
u_j &:= \frac{1}{n} \sum_{k=0}^{m-1} f_{2k+1}\, e^{-ijk\frac{2\pi}{m}} \\
y_j &:= g_j + e^{-ij\frac{\pi}{m}} \cdot u_j \\
y_{m+j} &:= g_j - e^{-ij\frac{\pi}{m}} \cdot u_j
\end{aligned}
\qquad (11.22)
$$

Der FFT-Algorithmus iteriert nun diesen Reduktionsschritt bis nur noch triviale Fourier-Summen für $m = 1$ auszuwerten sind. Die Fourier-Summen für $m = 2, 4, 8, \ldots, 2^r = n$ werden dann gemäß (11.22) aus diesen zusammengesetzt.

Im ersten Teilschritt werden die Ausgangsdaten (array $f_0 f_1, \ldots, f_{n-1}$) so umsortiert, dass die trivialen Fourier-Summen ausgewertet

werden können!

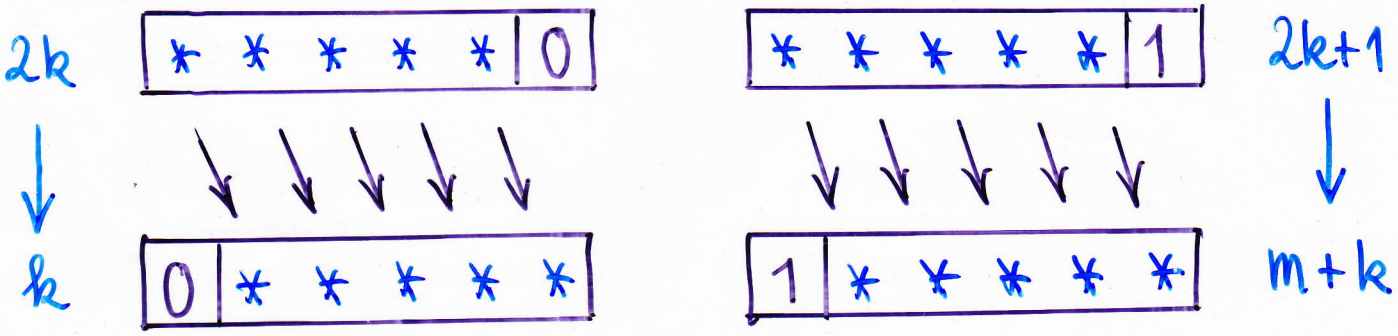
Beispiel (n=8)

Ausgangsdaten :	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
1. Schritt :	f_0	f_2	f_4	f_6	f_1	f_3	f_5	f_7
2. Schritt :	f_0	f_4	f_2	f_6	f_1	f_5	f_3	f_7
3. Schritt :	f_0	f_4	f_2	f_6	f_1	f_5	f_3	f_7

Wir betrachten den ersten Sortierschritt und sehen uns die Indizes an:

Ausgangsindex: $k = (k_r \dots k_1)_2 = \sum_{j=1}^{k} k_j 2^{j-1}$,

$k_j \in \{0,1\}$



Der nächste Sortierschritt wirkt analog auf den Rest der Ziffern $* \dots *$ – ohne die erste Ziffer zu verändern.

Für sämtliche r Sortierschritte ergibt sich

die Zuordnung

$$k = (k_r \dots k_1)_2 \rightarrow \bar{k} = (k_1 \dots k_r)_2 ,$$

d.h. die Ziffernfolge in der Dualdarstellung der Indizes dreht sich um.

Beispiel ($n = 8$):

k		$\bar{k}$	
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

Algorithmus (1. Teil: Sortieren) (11.23)

$d_0 := \frac{1}{n} f_0 ; \quad \bar{k} = 0 ;$

für $k = 1, 2, \dots, n-1$

$m := n/2;$

while $(m + \bar{k} \geq n)$: $m = m/2;$

$\bar{k} := \bar{k} + 3m - n;$

$d_{\bar{k}} := \frac{1}{n} f_k ;$

Erläuterung:

In der k-Schleife hat man die Ausgangssituation:

$$k_{alt} = (k_r \dots k_1)_2 \;\hat{=}\; k-1$$
$$\overline{k_{alt}} = (k_1 \dots k_r)_2 \;\hat{=}\; \overline{k-1}$$

Zunächst wird $\overline{k_{alt}}$ auf führende Einsen getestet:

$$m = (1,0 \dots 0) \;\hat{=}\; n/2$$
$$= (0,1,0..0) \;\hat{=}\; m/2$$
$$\vdots$$

$m + \overline{k_{alt}} \geq n \iff$ in $\overline{k_{alt}}$ ist $k_j = 1$, $j = 1,2,\dots$

Nach der while-Schleife hat man:

$$\overline{k_{alt}} = (1 \dots 1\ 0\ k_{j+2} \dots k_r)_2$$
$$m = (0..0\ 1\ 0 \dots 0)_2$$
$$k_{alt} = (k_r \dots k_{j+2}\ 0\ 1 \dots 1)_2$$
$$k_{neu} = (k_r \dots k_{j+2}\ 1\ 0 \dots 0)_2$$

$$\overline{k_{alt}} + 3m - n = (0 \dots 0\ 1\ k_{j+2} \dots k_r)_2$$
$$= \overline{k_{neu}}$$

Algorithmus (2. Teil: Reduktion)

(11.24)

für $l = 1, 2, \ldots, r$

$m = 2^{l-1}$; $m2 = 2 \cdot m$;

für $j = 0, 1, \ldots, m-1$

$c = \exp\left(-ij \frac{\pi}{m}\right)$

für $k = 0, (m2), n-m2$

$g := d_{k+j}$;

$u := c \cdot d_{k+j+m}$;

$d_{k+j} := g + u$;

$d_{k+j+m} := g - u$;