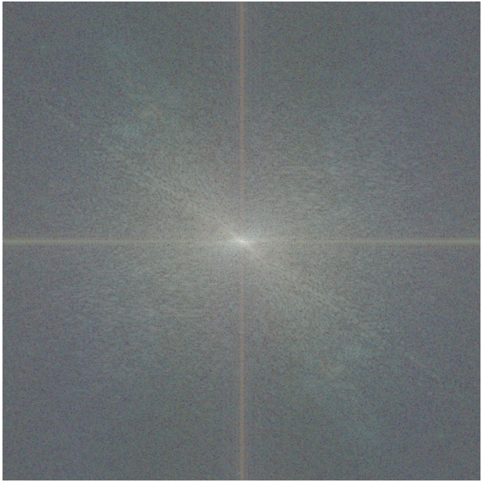
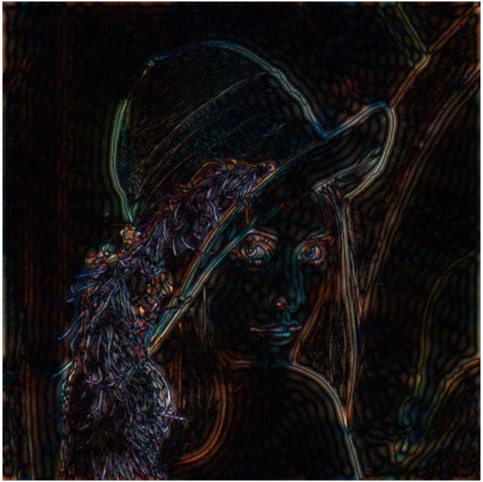
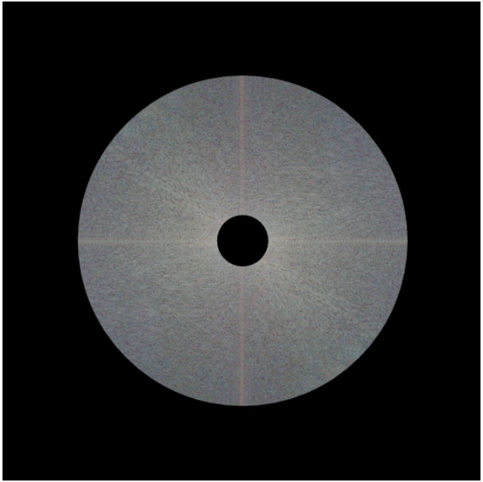
Fast Fourier Transform

Markus Pawellek

03.Februar 2016







Gliederung

Fouriertransformation periodischer Funktionen

Um welche Funktionen geht es?

Fouriertransformation periodischer Funktionen

Um welche Funktionen geht es?

(i) komplexwertig:

$$\boxed{f : \mathbb{R} \longrightarrow \mathbb{C}}$$

Fouriertransformation periodischer Funktionen

Um welche Funktionen geht es?

(i) komplexwertig:

$$\boxed{f : \mathbb{R} \longrightarrow \mathbb{C}}$$

(ii) periodisch mit Periode $T > 0$:

$$\boxed{\forall x \in \mathbb{R} : \quad f(x) = f(x+T)}$$

Fouriertransformation periodischer Funktionen

Um welche Funktionen geht es?

(i) komplexwertig:

$$\boxed{f : \mathbb{R} \longrightarrow \mathbb{C}}$$

(ii) periodisch mit Periode $T > 0$:

$$\boxed{\forall x \in \mathbb{R} : \quad f(x) = f(x+T)}$$

(iii) stückweise stetig differenzierbar

Fouriertransformation

$$\forall k \in \mathbb{Z}: \quad \mathcal{F}f(k) := \frac{1}{T}\int_0^T f(x)\exp\left(-\frac{2\pi i}{T}kx\right)\,\mathrm{d}x$$

$$\forall x \in \mathbb{R}: \quad f(x) := \sum_{k\in\mathbb{Z}} \mathcal{F}f(k)\exp\left(\frac{2\pi i}{T}kx\right)$$

Diskrete Fouriertransformation

Orthonormalbasis des $\mathbb{C}^n$ bezüglich Standardskalarprodukt

$$D := \left\{ \omega_k : \mathrm{N}_n \longrightarrow \mathbb{C} \;\middle|\; k \in \mathrm{N}_n,\ \forall x \in \mathrm{N}_n :\ \omega_k(x) = \exp\left(\frac{2\pi i}{n} kx\right) \right\}$$

Parsevalsche Gleichung

$$\forall x \in \mathrm{N}_n :\ g(x) = \sum_{k=0}^{n-1} \langle \omega_k, g \rangle\, \omega_k(x)$$

Diskrete Fouriertransformation

$$\forall k \in \mathrm{N}_n : \ \ \hat{g}(k) := \frac{1}{n} \sum_{x=0}^{n-1} g(x) \exp\left(-\frac{2\pi i}{n} kx \right)$$

$$\forall x \in \mathrm{N}_n : \ \ g(x) := \sum_{k=0}^{n-1} \hat{g}(k) \exp\left(\frac{2\pi i}{n} xk \right)$$

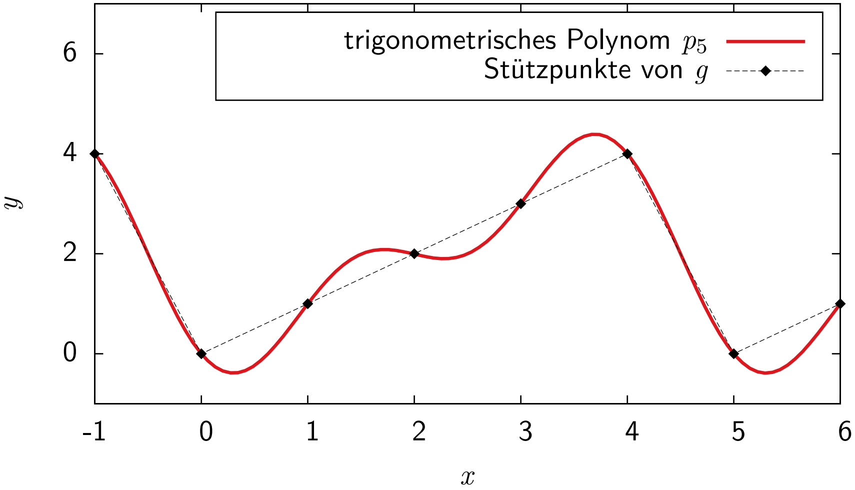
Ein Beispiel

$$g : \{0, 1, 2, 3, 4\} \longrightarrow \mathbb{C}, \qquad g(x) = x$$

x	$\sim \hat{g}(x)$
0	10
1	$-0.5 + 0.688191\, i$
2	$-0.5 + 0.162460\, i$
3	$-0.5 - 0.162460\, i$
4	$-0.5 - 0.688191\, i$

Tabelle: Fourierkoeffizienten der Beispielfunktion g

Ein Beispiel



Diskrete Fouriertransformation

$$\forall k \in \mathrm{N}_n : \;\; \hat{g}(k) := \frac{1}{n} \sum_{x=0}^{n-1} g(x) \exp\left(-\frac{2\pi i}{n} kx\right)$$

$$\forall x \in \mathrm{N}_n : \;\; g(x) := \sum_{k=0}^{n-1} \hat{g}(k) \exp\left(\frac{2\pi i}{n} xk\right)$$

Idee der Fast Fourier Transform

Rekursionsformeln der DFT

$$\begin{aligned} g_0 : \mathrm{N}_m &\longrightarrow \mathbb{C}, \qquad & g_0(x) &:= g(2x) \\ g_1 : \mathrm{N}_m &\longrightarrow \mathbb{C}, \qquad & g_1(x) &:= g(2x+1) \end{aligned}$$

$$\begin{aligned} \mathcal{F}_n g(k) &= \frac{1}{2}\left(\mathcal{F}_m g_0(k) + e^{-\frac{\pi i}{m}k}\ \mathcal{F}_m g_1(k)\right) \\ \mathcal{F}_n g(k+m) &= \frac{1}{2}\left(\mathcal{F}_m g_0(k) - e^{-\frac{\pi i}{m}k}\ \mathcal{F}_m g_1(k)\right) \end{aligned}$$

Ein rekursiver Algorithmus

Listing: rekursiver FFT-Algorithmus Teil 1

program Recursive-FFT
declare
  procedure fft(out, in : array$[0..2^q - 1 - 1]$ of complex ; q : integer);
  t : integer ;
  ω, b, c : array$[0..2^p - 1]$ of complex
initially
  {look-up table for twiddle factors}
  $\langle\ ;\ t\ :\ 0 \le t < 2^p\ ::\ \omega[t] = \exp(-\frac{2\pi it}{n})\ \rangle$;
  $\langle\ ;\ t\ :\ 0 \le t < 2^p\ ::\ b[t] = g(t)\ \rangle$
assign
  fft(c, b, p)
end

Ein rekursiver Algorithmus

Listing: rekursiver FFT-Algorithmus Teil 2

{*extra procedure for recursive algorithm*}

procedure fft(out, in : array$[0..2^q - 1 - 1]$ of complex ; q : integer)

begin

  if $q = 0$ **then**

    $out[0] = in[0]$

  else begin

    fft($out[0..2^{q-1} - 1]$, $in[0..2..2^q - 2]$, $q - 1$);

    fft($out[2^{q-1}..2^q - 1]$, $in[1..2..2^q - 1]$, $q - 1$);

    $\langle \parallel t \ : \ 0 \leq t < 2^{q-1} \ ::$

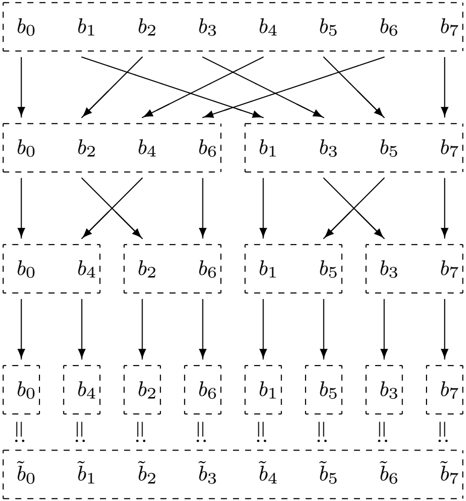
$$\begin{bmatrix} b[t] \\ b[t + 2^{q-1}] \end{bmatrix} := \frac{1}{2} \begin{bmatrix} 1 & \omega[2^{p-q}t] \\ 1 & -\omega[2^{p-q}t] \end{bmatrix} \begin{bmatrix} b[t] \\ b[t + 2^{q-1}] \end{bmatrix}$$

    $\rangle$

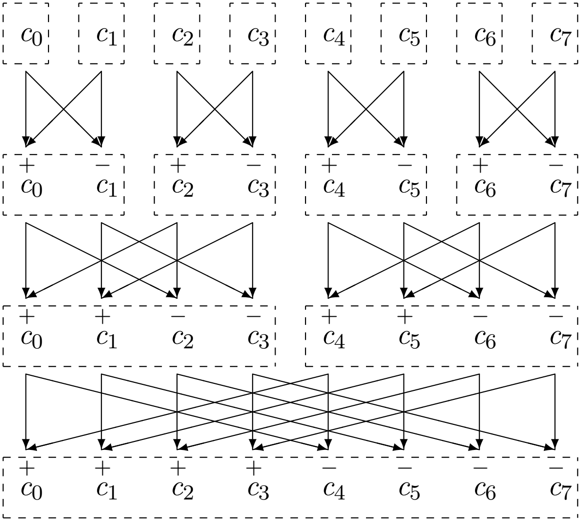
  end

end

Bit-Reversal



Butterfly Graph



Nicht-Rekursiver Algorithmus

Listing: nicht-rekursiver FFT-Algorithmus

program Serial-FFT
declare
  $m, t \; : \;$ integer ;
  $\omega, b \; : \;$ array$[0..2^p - 1]$ of complex
initially
  {look-up table for twiddle factors}
  $\langle \; ; \; t \; : \; 0 \le t < 2^p \; :: \; \omega[t] = \exp(-\frac{2\pi i t}{n}) \; \rangle$;
  {bit-reverse ordering}
  $\langle \; ; \; t \; : \; 0 \le t < 2^p \; :: \; b[t] = g(\sigma_p(t)) \; \rangle$
assign
  {butterfly calculation}
  $\langle \; ; \; m \; : \; 0 \le m < p \; ::$
    $\langle \; \| \; t \; : \; 0 \le t < 2^p - 1$ **and** $t \wedge 2^m = 0 \; ::$

$$\begin{bmatrix} b[t] \\ b[t+2^m] \end{bmatrix} := \frac{1}{2} \begin{bmatrix} 1 & \omega[2^{p-m-1} t \bmod 2^{p-1}] \\ 1 & -\omega[2^{p-m-1} t \bmod 2^{p-1}] \end{bmatrix} \begin{bmatrix} b[t] \\ b[t+2^m] \end{bmatrix}$$

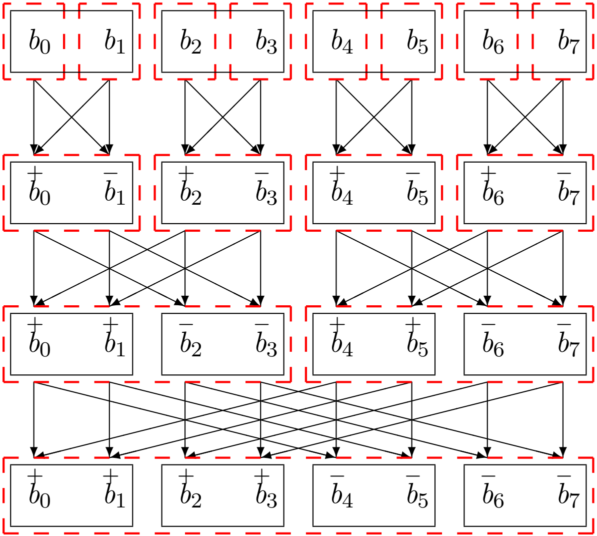
    $\rangle$
  $\rangle$
end

Datenverteilung

$\tilde{b}_0$ $\tilde{b}_1$	$\tilde{b}_2$ $\tilde{b}_3$	$\tilde{b}_4$ $\tilde{b}_5$	$\tilde{b}_6$ $\tilde{b}_7$
ρ_0	ρ_1	ρ_2	ρ_3

Abbildung: Verteilung des Eingabearrays $\tilde{b}$ auf den Prozessen ρ_i für $p = 3, q = 2$

Kommunikation



Ein paralleler Algorithmus

Listing: paralleler FFT-Algorithmus Teil 1

$0..2^q - 1 \parallel \rho$ **program** Parallel-FFT

declare

$\quad m, t$: integer ;

$\quad \omega$: array$[0..2^p - 1]$ of complex ;

$\quad b, x$: array$[0..2^{p-q} - 1]$ of complex

initially

$\quad$ *{look-up table for twiddle factors}*

$\quad \langle\ ;\ t\ :\ 0 \le t < 2^p\ ::\ \omega[t] = \exp(-\frac{2\pi i t}{n})\ \rangle$;

$\quad$ *{bit-reverse ordering}*

$\quad \langle\ ;\ t\ :\ 0 \le t < 2^{p-q}\ ::\ b[t] = g(\sigma_p(2^{p-q}\rho + t))\ \rangle$

assign

$\quad$ *{butterfly calculation with no communication}*

$\quad \langle\ ;\ m\ :\ 0 \le m < p - q\ ::$

$\quad\quad \langle\ ;\ t\ :\ 0 \le t < 2^{p-q}$ **and** $t \wedge 2^m = 0\ ::$

$$\begin{bmatrix} b[t] \\ b[t+2^m] \end{bmatrix} := \frac{1}{2}\begin{bmatrix} 1 & \omega[\xi(m,\rho,t)] \\ 1 & -\omega[\xi(m,\rho,t)] \end{bmatrix}\begin{bmatrix} b[t] \\ b[t+2^m] \end{bmatrix}$$

$\quad\quad \rangle$

$\quad \rangle$;

Ein paralleler Algorithmus

Listing: paralleler FFT-Algorithmus Teil 2

$\{$*butterfly calculation with communication*$\}$

$\langle\ ;\ m\ :\ 0 \leq m < q\ ::$

$\quad$ **send** $\{b[t] \mid 0 \leq t < 2^{p-q}\}$ **to** $\rho \bar{\nabla} 2^m$;

$\quad$ **receive** $\{x[t] \mid 0 \leq t < 2^{p-q}\}$ **from** $\rho \bar{\nabla} 2^m$;

$\quad$ **if** $\rho \wedge 2^m = 0$ **then**

$\quad\quad \langle\ ;\ t\ :\ 0 \leq t < 2^{p-q}$ **and** $t \wedge 2^m = 0\ ::$

$\quad\quad\quad b[t] := \frac{1}{2}\,(b[t] + \omega[\xi(m,\rho,t)]x[t])\ \rangle$

$\quad$ **else**

$\quad\quad \langle\ ;\ t\ :\ 0 \leq t < 2^{p-q}$ **and** $t \wedge 2^m = 0\ ::$

$\quad\quad\quad b[t] := \frac{1}{2}\,(x[t] - \omega[\xi(m,\rho,t)]b[t])\ \rangle$

$\quad$ **end**

$\rangle$

end

Leistungsanalyse

Gesamtzeit des parallelen FFT-Algorithmus

$$T(n,P) = 2\frac{n}{P}\left(\tau_{\mathrm{A}} \log_2 n + \beta \log_2 P\right) + 2\tau_{\mathrm{S}} \log_2 P$$

Speedup des parallelen FFT-Algorithmus

$$S(n,P) := \frac{T(n,1)}{T(n,P)} = \frac{P}{1 + \frac{\log_2 P}{\log_2 n}\left(\frac{\beta}{\tau_{\mathrm{A}}} + \frac{P}{n}\frac{\tau_{\mathrm{S}}}{\tau_{\mathrm{A}}}\right)}$$

Literatur und Quellen

[1] http://www.ejectamenta.com/Imaging-Experiments/fourierimagefiltering.html

[2] https://en.wikipedia.org/wiki/Fast_Fourier_transform

[3] https://en.wikipedia.org/wiki/Cooley-Tukey_FFT_algorithm

[4] http://cnx.org/contents/ulXtQbN7@15/Implementing-FFTs-in-Practice

[5] Eric F. Van de Velde, *Concurrent Scientific Computing*, Springer-Verlag, 1994

[6] Rami Shakarchi, Elias M. Stein, *Fourier Analysis - An Introduction*, Band I der *Princeton Lectures in Analysis*, Princeton University Press, 2003

[7] Jürgen Elstrodt, *Maß- und Integrationstheorie*, Springer, 2009

[8] Martin Hermann, *Numerische Mathematik*, Oldenbourg Verlag München, 2011