Projekt: (Schnelle) Fourier-Transformation

Florian Reinecke, Tobias Steinbrecher

5. Januar 2023

Inhaltsverzeichnis

1 Einleitung

Die *schnelle Fourier-Transformation (*FFT*)* ist ein Algorithmus zur Durchführung einer *diskreten Fourier-Transformation (DFT)*. Das Interesse besteht hierbei zum Einen im algorithmischen Hintergrund der *FFT*, welche auf einem *Divide & Conquer* Verfahren basiert. Zum Anderen in der Relevanz der Anwendung der *DFT*, wie z.B. die Zerlegung in das Frequenzspektrum eines Signals oder digitale Bildverarbeitung. Ein weiteres Problem ist das Multiplizieren von Polynomen, welches durch eine Reduktion auf die *DFT* schnell mithilfe der *FFT* gelöst werden kann. Der Algorithmus zur Multiplikation von Polynomen kann beispielsweise von der Laufzeit $\Theta(n^2)$, mithilfe der *FFT* auf eine Laufzeit von $\Theta(n \log n)$ optimiert werden.
Insgesamt lässt sich das *DFT*-Problem als

$$y_k = \sum_{j=0}^{n-1} a_j \cdot e^{\frac{i2\pi kj}{n}} \qquad k = 0, 1, ..., n-1 \tag{1}$$

bzw. in der Schreibweise einer Matrix-Vektor-Multiplikation:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & e^{1\cdot\frac{2\pi i}{n}} & e^{2\cdot\frac{2\pi i}{n}} & \dots & e^{(n-1)\cdot\frac{2\pi i}{n}} \\ 1 & e^{2\cdot\frac{2\pi i}{n}} & e^{4\cdot\frac{2\pi i}{n}} & \dots & e^{2(n-1)\cdot\frac{2\pi i}{n}} \\ 1 & e^{3\cdot\frac{2\pi i}{n}} & e^{6\cdot\frac{2\pi i}{n}} & \dots & e^{3(n-1)\cdot\frac{2\pi i}{n}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{(n-1)\cdot\frac{2\pi i}{n}} & e^{(n-1)\cdot 2\cdot\frac{2\pi i}{n}} & \dots & e^{(n-1)(n-1)\cdot\frac{2\pi i}{n}} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{n-1} \end{pmatrix} \tag{2}$$

formulieren. Somit ist die *FFT* lediglich ein schnelles Verfahren um einen Vektor mit einer speziellen Matrix (*DFT*-Matrix) zu multiplizieren.

2 Ziele

Zunächst ist das Ziel ein Verständnis und Intuition in Bezug auf die *Fourier-Analyse* zu entwickeln. Dabei soll primär die *DFT* vollständig verstanden werden und praktisch mit Hilfe des *FFT*-Algorithmus angewendet werden. Beispielsweise könnte ein Programm entwickelt werden, mit welchem man eine Audiodatei (`.wav` oder `.mp3`) in die verschiedenen Frequenzanteile zerlegen kann. Dafür gibt es selbstverständlich schon viele ausgereifte Anwendung, dies ist allerdings eine gute Methode, um das Projekt greifbar vorzustellen.
Besonders schön wäre es natürlich eine interaktive Sofortanalyse zu implementieren, dies ist allerdings etwas komplizierter, weswegen zunächst der Erkenntnisgewinn im Vordergrund steht.

3 Mathematischer Hintergrund

Für das richtige Verständnis der *FFT* und *DFT* muss man sich mit den komplexen n-ten Einheitswurzeln beschäftigen. Die n-ten Einheitswurzeln sind eine Menge $\Omega \subset \mathbb{C}$ an komplexen Zahlen, für die gilt:

$$\omega^n = 1; \forall \omega \in \Omega \tag{3}$$

Es gibt n verschiedene n-te Einheitswurzeln. Die k-te der n-ten Einheitswurzeln ist die k-te Potenzen der n-ten Einheitswurzel (im Folgenden ω_n^k). Die Einheitswurzeln lassen sich mithilfe der Formel für die Darstellung von komplexen Zahlen berechnen:

$$\omega_n^k = e^{2\pi i \frac{k}{n}} = cos(2\pi\frac{k}{n}) + isin(2\pi\frac{k}{n}) \tag{4}$$

Die Darstellung von komplexen Zahlen erfolgt auf dem Einheitskreis.

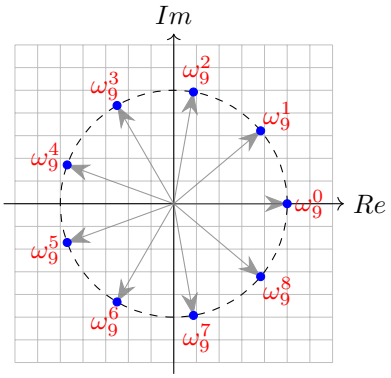

Abbildung 1: Einheitskreis mit den 9 verschiedenen 9-ten Einheitswurzeln

Eine entscheidende Eigenschaft der Einheitswurzeln ist das Quadrieren. Quadriert man die k-te n-te Einheitswurzel ω_n^k, so erhält man die k-te der $\frac{n}{2}$-ten Einheitswurzeln.

Lemma 3.1 *Es gilt* $(\omega_n^k)^2 = \omega_{\frac{n}{2}}^k$

Diese Eigenschaft ist entscheidend für den *Divide & Conquer* Ansatz.
Zudem folgt eine weitere wichtige Eigenschaft, welche für den Algorithmus (Abschnitt 4) relevant ist:

Lemma 3.2 *Es gilt* $\omega_n^{k+\frac{n}{2}} = -\omega_n^k$

Hinzu kommt eine weitere Eigenschaft, welche mit der rotierenden Wirkung von Potenzen der Einheitswurzeln zu begründen ist:

Lemma 3.3 *Es gilt* $\omega_n^{k+n} = \omega_n^k$

Die Einheitswurzeln ähneln ganzen Zahlen mit modularer Arithmetik (z.B. $\mathbb{Z}_5$).
Es gilt beispielsweise auch $3 + 5 = 3 \mod 5$.

4 Algorithmus

Die Idee hinter dem *FFT*-Algorithmus ist ein *Divide & Conquer* Ansatz. Das Problem der *DFT* wird somit in kleinere Subprobleme eingeteilt und mithilfe der Lösungen jener, gelöst. Interessant ist es dabei das Problem vom Multiplizieren von Polynomen zu betrachten. Die Polynome befinden sich dabei zunächst in einer Koeffizienten-Repräsentation (Liste an allen Koeffizienten des Polynoms) und lassen sich mit der *FFT* in eine Punkte-Repräsentation (Liste an Stellen/Werte-Paare) umwandeln. Dies optimiert den Prozess vom Multiplizieren, da die Multiplikation von Polynomen in ihrer Koeffizienten-Repräsentation in einer Laufzeit von $\Theta(n^2)$ geschieht (anwenden des Distributivgesetzes). In der Punkte-Repräsentation erfolgt das Multiplizieren in $\Theta(n)$, da lediglich die y-Werte der einzelnen Punkte der beiden Polynome multipliziert werden müssen (insofern sich die Punkte an gleichen Stellen befinden). Hieraus folgt die Punkte-Repräsentation für das Ergebnispolynom.

Sei A ein Polynom $(n-1)$-ten Grades in Koeffizienten-Rep.: $(a_0, a_1, ..., a_{n-1})$. Um die Koeffizienten-Rep. in Punkte-Rep. umzuwandeln, berechne man für die Werte $(\omega_n^0, \omega_n^1, ..., \omega_n^{n-1})$, die Funktionswerte. Somit werden n-Punkte generiert. Der Grund für das Verwenden von n-ten Einheitswurzeln als Stellen für das Berechnen der Punkte-Rep., lässt sich später mit Lemma 3.1 begründen.
Das Berechnen der Funktionswerte eines Polynoms an den Stellen $(\omega_n^0, \omega_n^1, ..., \omega_n^{n-1})$ mit dem Koeffizienten Vektor $\vec{a}$ und dem Ergebnisvektor $\vec{y}$ ist äquivalent zu Gleichung 1 (Multiplizieren eines Vektors mit der *DFT*-Matrix). Aus diesem Grund ist folgende Erläuterung des *FFT*-Algorithmus anhand der Multiplikation von Polynomen, auch gültig für eine Anwendung der *FFT* auf *DFT*-Probleme. Es folgt der Algorithmus mit der Koeffizienten-Liste a und dem Grad $n-1$ als Eingabe (n ist die Eingabe).

Algorithm 1 schnelle Fourier-Transformation

```
1:  procedure FFT(a, n)
2:      if n == 1 then
3:          return a
4:      end if
5:      y ← []
6:      ω_n ← e^{i 2π/n}
7:      ω ← 1
8:      a_e ← (a_0, a_2, ..., a_{n-2})
9:      a_o ← (a_1, a_3, ..., a_{n-1})
10:     y_e ← FFT(a_e, n/2)
11:     y_o ← FFT(a_o, n/2)
12:     for k ← 0 to n/2 − 1 do
13:         y[k] ← y_e[k] + ω y_o[k]
14:         y[k + n/2] ← y_e[k] − ω y_o[k]
15:         ω ← ω ω_n
16:     end for
17:     return y
18: end procedure
```

Zeilen 2 und 3 beschreiben den Basis-Fall für ein Polynom 0-ten Grades. Der Funktionswert entspricht hierbei dem ersten und einzigen Koeffizienten a_0 und kann somit zurückgegeben werden. Die grundlegende Idee hinter *FFT* ist das Unterteilen des Polynoms $A(x)$, in zwei Polynome $A_e(x)$ und $A_o(x)$. Dabei werden die geraden Koeffizienten $(a_0, a_2, ..., a_{n-2})$ A_e und die ungeraden Koeffizienten $(a_1, a_3, ..., a_{n-1})$ A_o zugeordnet (Zeilen 8 und 9). Es folgt:

$$A(x) = A_e(x) + A_o(x) = (a_0 + a_2x^2 + ... + a_{n-2}x^{n-2}) + (a_1x + a_3x^3 + ... + a_{n-1}x^{n-1}) \tag{5}$$

Eine wichtige Idee ist nun, x aus A_o auszuklammern, sodass auch A_o nur noch gerade Potenzen enthält. Es folgt:

$$A(x) = A_e(x) + xA_o(x) = (a_0 + a_2x^2 + ... + a_{n-2}x^{n-2}) + x(a_1 + a_3x^2 + ... + a_{n-1}x^{n-2}) \tag{6}$$

Da nun in $A_e(x)$ und $A_o(x)$ lediglich gerade Potenzen von x vorhanden sind, lässt sich dieses substituieren ($z = x^2$).
Es folgt:

$$A_e(z) = (a_0 + a_2z + ... + a_{n-2}z^{\frac{n}{2}-1}) \tag{7}$$

$$A_o(z) = (a_1 + a_3z + ... + a_{n-1}z^{\frac{n}{2}-1}) \tag{8}$$

und

$$A(x) = A_e(x^2) + xA_o(x^2) \tag{9}$$

Der Grad der Polynome $A_e(z)$ und $A_o(z)$ beträgt nur noch $\frac{n}{2} - 1$. Das Problem wurde auf kleinere Subprobleme reduziert.
Die kleineren Subprobleme sind jedoch noch nicht äquivalent zum ersten Aufruf, da der Grad und die Anzahl an zu berechnenden Punkten nicht zusammenpassen ($\frac{n}{2} - 1 \neq n$).
Aus diesem Grund wurden die Einheitswurzeln verwendet. Die Eigenschaft Lemma 3.1 der Einheitswurzeln wird relevant. Dadurch, dass

$$(\omega_n^k)^2 = \omega_{\frac{n}{2}}^k \tag{10}$$

folgt nach dem übergeben von x^2 (Quadrieren der zu berechnenden Stelle), dass nur noch $\frac{n}{2}$-te Einheitswurzeln bestehen und somit als Parameter für den erneuten *FFT* Aufruf $\frac{n}{2}$ übergeben werden kann (Zeilen 9 und 10). Durch das Quadrieren halbiert sich die Anzahl an zu berechnenden Stellen. Die Subprobleme sind nun äquivalent zum Ursprungsproblem: Berechne für ein Polynom $n-1$-Grades, den Funktionswert an n Stellen.
Es folgen Halbierungen, bis der Basis-Fall ($n = 1$) erreicht wird.
Folgend muss die Kombination der beiden Subergebnisse y_e und y_o erfolgen. Was zunächst unmöglich scheint, da nur $\frac{n}{2}$ Stellen berechnet wurden und n Punkte gebraucht werden. Hierbei kommt die nächste Eigenschaft der Einheitswurzeln zum Tragen. Die Ergebnisse der Subaufrufe haben jeweils die Größe von lediglich $\frac{n}{2}$ berechneten Punkten.
In einer Iteration über die beiden Listen (Zeile 12-16), werden die Ergebnisse gemäß Gleichung 9 kombiniert. Dabei werden in jeder Iteration zwei Ergebnisse konstruiert. Es lässt sich verifizieren, dass die Werte für y_k sachgemäß berechnet werden. Dafür muss gezeigt werden, dass die beiden Berechnungen (Zeile 13 und 14) gemäß Gleichung 9 korrekt sind:

$$A(\omega_n^k) = A_e((\omega_n^k)^2) + \omega_n^k A_o((\omega_n^k)^2) \tag{11}$$

Das $y[k]$ bzw. y_k ist dabei gleichbedeutend mit $A(\omega_n^k)$. Dabei ist eine interessante Eigenschaft der Einheitswurzeln dafür verantwortlich, dass mit einem Ergebnis mit $\frac{n}{2}$-ten Einheitswurzeln jeweils zwei Ergebnisse mit n-ten Einheitswurzeln erzeugt werden können. Zunächst die Berechnung für y_k mit $k = 0, 1, ..., \frac{n}{2} - 1$ nach Gleichung 9 in Zeile 13:

$$\begin{aligned} y_k &= A_e((\omega_n^k)^2) + \omega_n^k A_o((\omega_n^k)^2) \qquad \text{| mit Gleichung 9} \\ y_k &= A(\omega_n^k) \end{aligned} \tag{12}$$

Somit konnte verifiziert werden, dass die ersten y_k für $k = 0, 1, ..., \frac{n}{2} - 1$ erfolgreich aus den Ergebnissen der Subprobleme, verifiziert werden konnten. Die zweite Berechnung (Zeile 14) erfordert die Eigenschaft aus Lemma 3.2 und der Nachweis der Korrektheit ist etwas komplexer:

$$\begin{aligned} y_{k+\frac{n}{2}} &= A_e((\omega_n^k)^2) - \omega_n^k A_o((\omega_n^k)^2) \qquad \text{| mit } \omega_n^{k+\frac{n}{2}} = -\omega_n^k \text{ (Lemma 3.2)} \\ y_{k+\frac{n}{2}} &= A_e((\omega_n^k)^2) + \omega_n^{k+\frac{n}{2}} A_o((\omega_n^k)^2) \qquad \text{| mit Potenzgesetzen} \\ y_{k+\frac{n}{2}} &= A_e(\omega_n^{2k}) + \omega_n^{k+\frac{n}{2}} A_o(\omega_n^{2k}) \qquad \text{| mit } \omega_n^{c+n} = \omega_n^c \text{ (Lemma 3.3)} \\ y_{k+\frac{n}{2}} &= A_e(\omega_n^{2k+n}) + \omega_n^{k+\frac{n}{2}} A_o(\omega_n^{2k+n}) \qquad \text{| mit Potenzgesetzen} \\ y_{k+\frac{n}{2}} &= A_e((\omega_n^{k+\frac{n}{2}})^2) + \omega_n^{k+\frac{n}{2}} A_o((\omega_n^{k+\frac{n}{2}})^2) \qquad \text{| mit Gleichung 9} \\ y_{k+\frac{n}{2}} &= A(\omega_n^{k+\frac{n}{2}}) \end{aligned} \tag{13}$$

Durch gezieltes Anwenden von Potenzgesetzen und einigen Eigenschaften der Einheitswurzeln, können auch die zweite Hälfte der Ergebnisse y_k für $k = \frac{n}{2}, ..., n-1$ erfolgreich aus den Ergebnissen der Subprobleme berechnet werden. Es scheint unmöglich, jedoch können aus den $\frac{n}{2}$ berechneten Punkten mit den Eigenschaften der Einheitswurzeln, n Punkte wiederhergestellt werden. Dies ist auch mit der Symmetrie von Funktionen begründbar. Der algebraische Beweis ist möglicherweise jedoch etwas eleganter.
Zuletzt in jeder Iteration (Zeile 15) wird lediglich die Einheitswurzel um eine Potenz erhöht (somit das nächste Ergebnis berechnet) und schlussendlich die berechneten Ergebnisse (Punkte) zurückgegeben (Zeile 17).

5 Quellen

1. *Introduction to Algorithms* (4th Edition) Cormen et al. (p. 886-895 *Polynomials and the FFT*)
2. Wikipedia: Fast Fourier Transform
3. Wikipedia: Diskrete Fourier Transformation
4. Algorithms for Competetive Programming: FFT
5. Wikipedia: Einheitswurzel