

FFT und Anwendungen

Projekt zur Vorlesung „Einführung in die Numerik“

David Cohen, Christian Stohrer

Frühlingssemester 2011

Zusammenfassung

In diesem Beschrieb zum Numerik Projekt werden einerseits die Bedingungen für das Bestehen des Projekts aufgeführt. Andererseits werden die Aufgaben detailliert erklärt. Die benötigte Theorie wird nicht hier näher erläutert, sondern wurde bei der Projekt-Einführung erklärt. Die Slides zu dieser Einführung sind auf dem Internet verfügbar.

Bedingungen für das Bestehen des Projekts

- Bearbeitung und Lösung der gestellten Aufgaben. Die mit einem Sternchen (*) markierten Aufgaben sind fakultativ. Die Bearbeitung wird jedoch sehr empfohlen, da dadurch das Verständnis gefördert wird und die weiteren Aufgaben dadurch einfacher werden. Die Programmieraufgaben sind mit MATLAB zu lösen. Auf der Webseite finden Sie Skelette zu einigen Aufgaben.
- Zusammenstellung der Ergebnisse in einem kurzen Bericht (ca. 3 Seiten). Der Bericht soll vorzugsweise mit L^AT_EX geschrieben werden. Eine Vorlage wird auf der Webseite bereitgestellt. Der Bericht soll die Lösungen und Ergebnisse zu den gestellten Aufgaben enthalten. Als Anhang sollen die Codes zu den Programmen aufgeführt werden.
- Besprechung des Projekts und des Berichts. Bei dieser Besprechung sollen die Programme vorgeführt und erklärt werden. Dabei soll auch ein Teil Ihrer Resultate mit ihren Programmen nachgerechnet werden. Schreiben Sie daher `main`-Files, welche die Resultate ihres Berichts erzeugen.

Termine

26. April 2011	Herausgabe des Projekts
Mai 2011	Bearbeitung des Projekts (Fragestunden nach Absprache)
19. Juni 2011	späteste Abgabe des Berichts
25. - 29. Juli 2011	Besprechung der Berichte (Ist nach Absprache auch früher möglich.)

Fragen

Bei Fragen wenden Sie sich an Christian Stohrer (`christian.stohrer@unibas.ch`).

Aufgaben

Die Aufgaben sind viergegliedert. Der erste Teil soll zuerst bearbeitet werden, da die folgenden Teile darauf aufbauen. Der zweite, dritte und vierte Teil sind voneinander unabhängig.

Teil 1: Fast Fourier Transform

Von der Webseite können Sie die Programme `dft` und `idft` herunterladen. Diese Programme berechnen die diskrete Fouriertransformation und die inverse diskrete Fourier Transformation, allerdings nicht mit dem FFT-Algorithmus.

Aufgabe 1.1 Berechnen Sie von Hand die diskrete Fouriertransformierte des Vektors $y = [2, 1, 0, 3]$. Geben Sie nicht nur das Zwischenergebnis, sondern auch den Rechenweg an.

Aufgabe 1.2 * Wiederholen Sie die Aufgabe 1. Verwenden Sie jedoch dieses Mal die FFT.

Aufgabe 1.3 Implementieren Sie die schnelle Fourier Transformation (FFT). Schreiben Sie dazu eine MATLAB-Funktion `z = fft_2(y)`. Wie in der Einführung beschränken wir uns auf den Fall, dass y ein (möglicherweise komplexer) Vektor der Länge 2^n ist. Programmieren Sie ebenfalls die Funktion `y = ifft_2(z)` zur Berechnung der schnellen inversen Fourier Transformation des Vektors z . Testen Sie ihre Programme mit Hilfe der Aufgabe 1.1 und den Funktionen `dft` und `idft`.

Aufgabe 1.4 * Spielen Sie mit Ihren Algorithmen um ein Gefühl für das Frequenzspektrum zu bekommen. Betrachten Sie z.B. die Spektren der folgenden Funktionen:

$$y = \sin(2x), \quad y = \sin(10x) \quad y = \cos(4x), \quad y = 2 + \sin(2x) + \sin(10x) + \cos(4x), \quad \dots$$

Setzen Sie dazu $x_k = 2k\pi/N$ für $k = 0, 1, \dots, N-1$, wobei $N = 2^n$, $n \in \mathbb{N}$. Berechnen Sie $y_k = f(x_k)$, wobei f die zu untersuchende Funktion ist und wenden sie die Fouriertransformation darauf auf. Zum Betrachten der Spektren können sie die Funktion `plotfreq` verwenden.

Aufgabe 1.5 Vergleichen Sie die Laufzeit Ihres Programms `fft_2` mit derjenigen von `dft` für Vektoren verschiedener Länge. Für die Zeitmessungen können Sie die MATLAB-Funktionen `tic` und `toc` verwenden. Für ein besseres Resultat verwenden Sie Zufallsvektoren (Befehl `rand`) und machen pro Länge jeweils mehrere Messungen. Stellen Sie ihre Ergebnisse graphisch dar. Passen die Ergebnisse zur Theorie?

Aufgabe 1.6 Entwickeln Sie einen Algorithmus, der die diskrete Fourier Transformierte eines Vektors der Länge $N = 4^n$ folgendermassen berechnet. Der Algorithmus teilt den Vektor $y = (y_k)_{k=0}^{N-1}$ in vier Subvektoren y^0 , y^1 , y^2 und y^3 auf. Dabei gilt

$$y_k^j = y_{4k+j}, \quad \text{für } k = 0, \dots, N/4 - 1 \text{ und } j = 0, \dots, 3.$$

Die Fourier Transformierte von y kann aus den Fourier Transformationen der Subvektoren zusammengesetzt werden. Der Algorithmus gleicht sehr dem klassischen FFT und kann auch auf dieselbe Art hergeleitet werden. Implementieren und testen Sie diesen Algorithmus.

Teil 2: Trigonometrische Interpolation

Für diesen Teil übernehmen wir die Notationen aus Aufgabe 1.4. Gesucht ist das reelle trigonometrische Interpolationspolynom $p_N(x)$ zu den N Stützpunkten (x_k, y_k) mit $x_k = 2k\pi/N$ für $k = 0, 1, \dots, N-1$.

Aufgabe 2.1 Schreiben Sie zur trigonometrischen Interpolation eine MATLAB-Funktion `yy = triginterpol(y,xx)`, welche $p_N(x)$ zu den Stützwerten y an den Stellen xx auswertet. Dabei ist y der Vektor der Stützwerte und xx sind die Stellen, an welchen das trigonometrische Interpolationspolynom ausgewertet werden soll. Zur Implementierung verwenden Sie Ihre Funktion `fft_2` für die Fouriertransformation, berechnen Sie dann die Koeffizienten A_ℓ und B_ℓ des Interpolationspolynoms mit der Formel von der Einführung des Projekts und werten schliesslich das Polynom an den Stellen xx aus.

Aufgabe 2.2 Berechnen Sie die trigonometrische Interpolation für die folgenden vier 2π -periodischen Funktionen

$$\begin{aligned} f_1(x) &= e^{\sin x + \cos x}, & f_2(x) &= \left| 1 - \frac{x}{\pi} \right|, \\ f_3(x) &= \begin{cases} 0 & \text{falls } x \leq \pi \\ 1 & \text{falls } x > \pi \end{cases}, & f_4(x) &= x. \end{aligned}$$

Verwenden Sie jeweils eine verschiedene Anzahl von Stützwerten. Genauer: Wählen Sie $N = 2^n$ für $n = 2, \dots, 6$. (Für jede der vier Funktionen, machen Sie also fünf Interpolationen.) Stellen Sie ihre Resultate möglichst sinnvoll dar und kommentieren Sie ihre Ergebnisse. Stimmen die Resultate mit Ihrer Erwartung überein?

Teil 3: Tonerkennung und Soundbearbeitung

Für diesen Teil benötigen Sie Ihre MATLAB-Programme `fft_2` und `ifft_2`. Machen Sie sich mit der Funktionsweise der Befehle `wavread` und `wavwrite` zum einlesen und speichern von wav-Dateien vertraut.

Aufgabe 3.1 Auf der Webseite finden Sie Klangbeispiele der Töne c bis h (eingestrichene Tonlage, reine Stimmung), sowie ein Klangbeispiel eines Akkords. Finden Sie heraus, aus welchen Tönen der Akkord zusammengesetzt ist. Lesen Sie dazu die verschiedenen Dateien in MATLAB ein (`wavread`), danach berechnen und betrachten Sie die Frequenz-Spektren (`fft_2` und `plotfreq`). Sie können Ihr Ergebnis überprüfen, indem Sie den Akkord erneut aus den einzelnen Grundtönen „zusammen mischen“. Beschreiben Sie im Bericht ihr Vorgehen und ihre Erkenntnis.

Aufgabe 3.2 * Versuchen Sie mit Hilfe des Frequenzspektrums aus dem Akkord die einzelnen Töne zu extrahieren.

Aufgabe 3.3 Bei der Tonaufnahme von Händels „Halleluja“ begann ein Mobiltelefon nervtötend zu piepsen. Diese Aufnahme können Sie von der Website herunterladen. Versuchen Sie diese Aufnahme unter Verwendung der Programm `fft_2` und `ifft_2` zu verbessern, indem Sie das Telefonklingeln abschwächen oder ausblenden. Beschreiben Sie Ihre Methode und den damit erzielten Erfolg. Als Hilfe finden Sie auf der Website Bilder des Spektrums des Telefons und von einer ungestörten Aufnahme des „Halleluja“. Das Resultat muss nicht perfekt sein.

Teil 4: Bildbearbeitung

Hier soll das Frequenzspektrum eines Bildes berechnet und bearbeitet werden. Der Einfachheit halber verwenden wir nur Graubilder mit 256×256 Pixel. Damit Sie nicht lange nach einem passenden Bild suchen müssen können Sie die MATLAB-Funktion `imprep` verwenden. Diese liest ein farbiges Bild ein, rechnet dieses in Graustufen um und schneidet es auf die entsprechende Größe zu. Das Resultat ist eine 256×256 -Matrix, in welcher die Grauwerte der einzelnen Pixel gespeichert sind. Die Funktion `imfinish` verwandelt eine solche Matrix wieder in das Bildformat von MATLAB. Ein solches Bild können Sie mittels dem Befehl `image` anzeigen und mittels `imwrite` abspeichern.

Aufgabe 4.1 Programmieren Sie zwei MATLAB-Funktion `fft2d` und `ifft2d`, welche die Fouriertransformation, resp. die inverse Fouriertransformation einer 256×256 -Matrix berechnen. Verwenden Sie dazu Ihre Programme `fft_2` und `ifft_2`.

Aufgabe 4.2 * Berechnen und betrachten Sie die Fouriertransformationen der Testbilder, welche Sie von der Website herunterladen können. Wo befinden sich die hoch- und wo die niederfrequentigen Anteile?

Aufgabe 4.3 Implementieren Sie einen Hochpassfilter. Dieser Filter lässt nur hochfrequente Anteile passieren und unterdrückt die niedrigen Frequenzen. Wenden Sie diesen Filter auf ein von Ihnen gewähltes Bild an. Als Ergebnis sollten die Kanten hervorgehoben werden.

Aufgabe 4.4 Implementieren Sie einen Tiefpassfilter. Dieser Filter lässt nur niederfrequente Anteile passieren und unterdrückt hohen Frequenzen. Wenden Sie diesen Filter auf ein von Ihnen gewähltes Bild an. Der Tiefpassfilter wirkt wie ein Weichzeichner.