Frequenzanalyse mit libfftw

Carl Wenninger
cw@carl-wenninger.de

Vortrag auf dem Linux Infotag der luga
am 25. März 2006

Skript und Ressourcen:
http://www.carl-wenninger.de/vortrag/lit06

DFT, FFT und libfftw

- Jean Baptiste Fourier (1768-1830)

- DFT: Discrete Fourier Transform.

- FFT: Fast Fourier Transform (ab etwa 1960; verschiedene Varianten).
  Komplexität $O(N \log(N))$ statt $O(N^2)$.
  Bereits Gauß (1777-1855) war sehr geschickt im Rechnen mit komplexen Einheitswurzeln.

- libfftw: C-Funtionsbibliothek zur Berechnung von DFTs.
  http://fftw.org: "Fastest Fourier Transform in the West".

Aufbau dieses Spaziergangs

- Propädeutik der Fourier-Analyse oder “DFT für Fußgänger“.
- Frequenzanalyse eines Ausschnitts der d-Moll Toccata.
- Ein einfaches Sound-Filter gegen 50Hz Brummen.
- Die Zeitgleichung oder das Dilemma mit meiner Sonnenuhr.
- Die Formel für den Augsburger Sonnenaufgang.

Eine letzte Warnung

- Vorsicht: Ein Mathematiker erklärt Mathematik fast ohne Formeln!
- Vorsicht: Ein Software-Entwickler wird zum Hacker!
- Ziel: Ein hoffentlich unterhaltsamer Spaziergang!

Audio CDs

- Ausgangspunkt: Akustisches Signal, 2 Kanal Stereo.
- Digitalisierung: 44100 Samples pro Sekunde und Kanal.
- Jedes Sample wird als 16-bit Ganzzahl binär gespeichert.
- Abwechselnd: linker Kanal, rechter Kanal, ...

Typische WAV-Datei

- WAV-Datei = Header + Samples.
- Header typisch 48 byte (Vorsicht: Hacker).
- Samplingrate, Anzahl Kanäle, Bits pro Sample, ...
- Linker Kanal, rechter Kanal, ...

Hack: wavdump.c

- Konvertiert 16-Bit-Wavs nach ASCII.
- Optionen: -m für Mono, -l für linker Kanal, -r für rechter Kanal.
- gcc -Wall -O2 -s -o wavdump wavdump.c

Frequenzanalyse mit bloßem Auge

- aplay freizeichen.wav
- wavdump -m < freizeichen.wav > freizeichen.dump
- less freizeichen.dump
- gnuplot
  plot [0:100] “freizeichen.dump“ with impulses

Parameter einer Sinus-Wellen

- Amplitude = Lautstärke.
- Frequenz = Tonhöhe (vgl. tonleiter.pdf).
- Phase (nicht hörbar).

Freizeichen als Standortnachteil?

- Auszählen: 9 Maxima von Sample Nr. 17 bis Sample Nr. 945.
- 9 Schwingungen in $\frac{928}{44100}$ Sekunden entsprechen etwa 428 Hz.
- Das Freizeichen liegt also außerhalb der 12-Tonleiter etwa in der Mitte zwischen Gis $=$ 415 Hz und A $=$ 440 Hz.
- Laut Wikipedia wird das Wiener Freizeichen (440 Hz) gerne als Stimmgabel verwendet!

Ein Kosinus-Generator

- Gegeben: Frequenz f, Amplitude A und Phasenwinkel φ.
- Tonsignal: Zeit $t \mapsto A \cdot \cos(2\pi f t - \varphi)$.
- Gesucht: Passende 16-bit Ganzzahl-Samples für dieses Tonsignal.
- In der Schule nennt man das eine Wertetabelle dieser Funktion.

Eine Stimmgabel

- Wir wählen $f = 440$ Hz.
- "Aussteuern": $A = 30000 < 2^{15}$.
- Unsere Zeit läuft von $t_0 = 0$ bis $t_1 = 5$ Sekunden.
- Mit $\varphi = 90°$ also $\varphi = \frac{\pi}{2}$ wird $\cos(x - \varphi) = \sin(x)$.
- Unser Tonsignal hat folglich Nullstellen bei t_0 und t_1.
- Damit wird störendes Knacksen vermieden.

Hack: stimmgabel.c

- gcc -Wall -O2 -s -o stimmgabel stimmgabel.c -lm
- stimmgabel.c liefert binäre 16 bit samples auf stdout.
- Mono, Samplingrate = 44100.
- Rohe Daten = WAV-File ohne Header.
- Also: stimmgabel | aplay -f cd -c 1 -t raw -

Diskrete Fourier Transformation

- Samplebasierte Beschreibung von Tonsignalen:
  Wertetabelle in Form einer WAV-Datei.

- Frequenzbasierte Beschreibung von Tonsignalen:
  Frequenzen + deren Amplituden + Phasen.

- DFT: Umrechnen von Samples auf Frequenzen.
  Einfachster Fall: gnuplot + Taschenrechner.

- Inverse DFT: Umrechnen von Frequenzen auf Samples.
  Einfachster Fall: stimmgabel.c

Diskrete Fourier Transformation aus Sicht eines Musikers

- Samplebasiert: WAV Datei.
- Frequenzbasiert: Noten.
- Merke: Der Mann am Klavier ist ein inverser Fourier Transformator.
  Er setzt Noten in hörbare Musik um!

Beispiel: Anfang der D-Moll Toccata von J.S.Bach

- Noten: toccata.png.
- aplay toccata.wav.
- Gelingt Ihnen die Zuordnung?

Kurzer Ausschnitt aus der D-Moll Toccata

- aplay ausschnitt.wav
- wavdump -l < ausschnitt.wav > ausschnitt.dump
- plot [0:2000] “ausschnitt.dump“ with impulses
- Welche Frequenzen kommen hier vor?

Die Mathematik behält den Überblick

- Vorgriff: Frequenzanalyse mit powerspec (siehe unten).
- powerspec 44100 < ausschnitt.dump > ausschnitt.power
- plot “ausschnitt.power“ with impulses
- Das ist jetzt ein Frequenz-Spektrum.
  X-Achse: Frequenz in Hz.
  Y-Achse: Leistung relativ zur Gesamtleistung.
  (Gesamtleistung: ohne Berücksichtigung des DC-Anteils)

Welche Töne kommen vor?

- plot [0:500] “ausschnitt.power“ with impulses
- D bei 37 Hz. Länge der Pfeife $\approx \frac{\lambda}{2} = 4,6$m.
- 1. Oberton bei 74 Hz ist eine Oktave höher.
- Stimmung der Orgel: 446 Hz.
- Scharfer Peak bei 409 Hz. Außerhalb der Tonleiter. Also ein dauerhafter Störton (Windpfeifen?).
- “Heißenbergsche“ Unschärferelation: $\Delta t \cdot \Delta f \approx 1$
  Je kürzer ein Ton andauert, desto unschärfer ist seine Frequenz.

Amplitude, Phase, Leistung

- $a\cos(x) + b\sin(x) = A\cos(x - \varphi)$ mit
- Amplitude $A^2 = a^2 + b^2$ (Satz des Pythagoras)
- Phase $\varphi = \text{atan2}(a, b)$
- Anders ausgedrückt:
  Der Punkt (a, b) besitzt die Polarkoordinaten (A, φ).
- Physik: A^2 ist ein Maß für die Schall-Leistung der betrachteten Frequenz.
- Die DFT liefert (a, b). Uns interessiert oft (A, φ) oder gar nur A^2.

Leistungsspektrum: powerspec.c

- Kurze Beschreibung des API von libfftw3.
- gcc -Wall -O2 -s -o powerspec powerspec.c -lfftw3 -lm
- Benötigt libfftw3.so (und libm.so) zur Laufzeit (SuSE: fftw3.rpm).
- Benötigt fttw3.h zur Übersetzungszeit (SuSE: fftw3-devel.rpm).

Prinzip eines frequenzbasierten Audiofilters

$$\text{Samples} \xrightarrow{\text{DFT}} \text{Frequenzen} \xrightarrow{\text{manipuliere}} \text{Frequenzen} \xrightarrow{\text{invDFT}} \text{Samples}$$

Beispiel: dämpfe unerwünschtes 50Hz Brummen

- Mikrofonaufnahme mit 50Hz Brummen meist aufgrund mangelhafter Abschirmung des zu langen Kabels.
- Manipulation = dämpfe Frequenzen nahe bei 50 Hz.
- Zunächst Frequenzanalyse. Brummen = starker Peak bei 50Hz + Obertöne.
- Beobachtung: Zu starkes Dämpfen wirkt kontraproduktiv.
  Dämpfe in etwa auf umgebendes Rauschnivau zurück.

Hack: daempfe.c

- mp3blaster brummt.mp3 (Mono, 16 bit, 44100)
- gcc -Wall -O2 -s -o daempfe daempfe.c -lfftw3 -lm
- lame –decode brummt.mp3 - | daempfe daempfe.conf | aplay -f cd -c 1 -
- daempfe.conf: Zeilen der Form “Startfreq: dB1 dB2 ...“
  49: 3 6 3 dämpft 50Hz um 6dB, 49Hz und 51Hz um 3dB
- 6dB = Faktor 4 in Leistung = Faktor 2 in Amplitude.
  3dB = Faktor 2 in Leistung = Faktor $\sqrt{2}$ in Amplitude.

Wozu eine Formel für den Sonnenaufgang?

- Motivation: verbesserte Steueruhr für elektrischen Rolladen + ein wenig Langeweile an einem regnerischen Wochenede.
- Die sog. Zeitgleichung wird hier experimentell erfahrbar!
- Zeitgleichung = mittlere Ortszeit - wahre Ortszeit.
- Zeitgleichung = Gangfehler perfekter Sonnenuhren.
- Analemma vgl. http://antwrp.gsfc.nasa.gov/apod/ap030320.html
- Größte Zeitgleichung im Herbst: gut 15 Minuten.

Ursache 1 der Zeitgleichung

- Erdbahn um die Sonne ist kein Kreis.
- Anfang Januar ist die Winkelgeschwindigkeit besonders hoch (Perihel).
- 2. Keplersches Gesetz: Der Fahrstrahl von der Sonne zur Erde überstreicht in gleichen Zeiten gleiche Flächen.
- Grafik: http://monet.unibas.ch/intro-physik/Kapitel_5/sld022.htm

Ursache 2 der Zeitgleichung

- Der Abstand zwischen zwei Längenkreisen ist am Äqator am größten.
- Die Sonne kommt also auf ihrem Weg um die Erde im Sommer und Winter schneller voran als im Frühjahr und im Herbst.
- Ich hatte selbst lange Zeit Probleme mit diesem Argument. Aber es ist wirklich wahr!

xephem

- Sternkarte + Ephemeride + noch viel mehr!
- Tolles Programm!
- Erstaunliche Präzission!
- http://www.xephem.com

xephem: Demo zu Ursache 2 der Zeitgleichung

- View $\rightarrow$ Earth
- Control $\rightarrow$ Objects
- Sun: Display + Label + Center
- Main: Step = Sidereal Day, N Steps = 100000, Update
- Jetzt sehen Sie, wie die Sonne im Laufe eines Jahres von West nach Ost(!) über die Erde wandert!

Was ist ein Sterntag?

- Die Erde dreht sich einmal in 24 Stunden um sich selbst. Richtig?
- Nein, dazu braucht sie genau 23:56:04 Stunden oder einen Sterntag!
- Die restlichen 3:56 Minuten braucht sie, um die Sonne wieder einzuholen, die sich zwischenzeitlich $\frac{360}{365} \approx 1$ Grad nach Osten bewegt hat!
- Im Zeitraffer mit Step = 23:56:04 erscheint die Erde relativ zum Sternenhimmel (Frühlingspunkt) unbewegt!
- Ausmessen: Für 15 Längengrade braucht die Sonne im Juni etwa 14 Tage, im September jedoch circa 17 Tage!

Ansatz: Formel für Sonnenaufgang

- Elementare Theorie berücksichtigt Zeitgleichung nicht.
- Praktischer Ansatz: Wertetabelle aus Beobachtung bzw. xephem.
- plot “rise.txt“
  Sieht ziemlich sinusförmig aus!
- Handelt es sich gar um einen phasenverschobenen Sinus?
- DFT kann diese Frage klären!

DFT: Sonnenaufgang in Augsburg

- $d =$ laufende Nummer des Tages im Kalenderjahr.
- $\delta = \frac{2\pi}{365} \cdot (d-1)$ "Winkel des Tages".
- Sonnenaufgang in Minuten MEZ (nicht MESZ):
  $368 + 111\cos(\delta + 0,120) + 10\cos(2\delta - 1,104) + 4\cos(3\delta + 0,471)$
- Auch Partialsummen sind brauchbar: vgl. sonnenaufgang.gnuplot
- Sonnenuntergang:
  $1104 + 115\cos(\delta - 2,885) + 9\cos(2\delta - 1,295) + 4\cos(3\delta - 2,475)$

Wo steht die Sonne um 12 Uhr mittags?

- Azimuth der Sonne in Augsburg jeweils Mittags um zwölf.
- M. a. W.: Steht die Sonne um 12 Uhr exakt im Süden?
- Nein. Wenn dann erst um 12:16 Uhr! Denn Augsburg liegt 4 Grad westlich vom 15. östlichen Längengrad. Pro Längengrad 4 Minuten.
- Nochmals Nein: Analemma, vgl. oben!

DFT der Sonne jeweils um 12:00 Uhr

- Daten aus xephem im Grad:Minute:Sekunden Format.
- Konvertieren in Bogensekunden.
- Dann Fourier-Analyse:
- plot [7:50] “sonne12.power“ with impulses.
- Auffallende Peaks beim 12 bis 13 fachen der Grundfrequenz.
- Der Mond wackelt an der Erde. Und das $\frac{365}{29,53} = 12,36$ Mal pro Jahr. Amplitude: Etwa 6 Bogensekunden. Voila.

Vielen Dank für Ihre Aufmerksamkeit!

Noch Fragen?