54. Weinheimer UKW-Tagung, Bensheim

Digitale Signalverarbeitung für Einsteiger

Dipl.-Ing. Erich H. Franke, DK6II

erich.franke@afusoft.com

Die Aufbereitung analoger Signale durch digitale Systeme ist heutzutage in der überwiegenden Zahl an Geräten und Systemen „State of the Art“. Kein Mobiltelefon wäre denkbar ohne diese Technik und GPS schon gar nicht. Computertomographen, moderne Röntgengeräte oder das simple Telefon? Undenkbar! MP3-und DVD-Player und Fernsehgeräte sind voll von digitaler Signalverarbeitung und auch im Amateurfunk beginnt sich die digitale Übertragungstechnik mehr und mehr zu etablieren. Nun ja! In der Bild- und Schriftübertragung war der Trend zur prozessorgestützten Digitaltechnik schon viele Jahre erkennbar. Doch damals war der Übergang von der analogen in die digitale Welt durch eine bestimmte Baugruppe dominiert, Modem genannt. Heute tritt an dessen Stelle meist die Soundkarte eines Computers unterstützt durch ein Stück Software. Doch wie diese Technik funktioniert ist für viele Amateure zunehmend schwer durchschaubar. Dieser Beitrag will helfen, ein wenig Licht in das digital-analoge Dunkel zu bringen.

Königsbach-Stein, im Juli 2009

Motivation

Warum werden analoge Schaltungen immer häufiger digital realisiert? Der Grund ist gleichermaßen kommerziell wie technisch motiviert. Digitale Schaltungen sind einfacher zu integrieren und in der Großserie viel billiger zu fertigen als ihre analogen Gegenstücke!

Jeder von uns, der schon einmal HF-Schaltungen gebaut und in Betrieb genommen hat kennt vielleicht das Problem: „Alles schwingt, nur der Oszillator nicht!“ Der Hintergrund dieser scherzhaften Techniker-Regel liegt in der Tatsache, dass Entwickler analoger Schaltungen häufig mit parasitären Effekten leben müssen: Zwischen Leiterbahnen gibt es nun einmal Kapazitäten und diese bilden unerwünschte Verkopplungen. Die eingekoppelten – mit hin unerwünschten – Störsignale überlagern sich mit dem Nutzsignal.

Da aber analoge Verstärkerschaltungen mit stetigen Signalen arbeiten, machen sie keine Unterschiede zwischen erwünschten und unerwünschten Signalkomponenten. Wird also ein Signalgemisch aus Nutz- und Störgrößen auf einen linearen Verstärker gegeben, so wird im besten – theoretischen - Fall die Störgröße mit verstärkt. In der – ungünstigeren - Praxis bilden sich, streng gemäß dem Gesetz von Murphy, an Nichtlinearitäten der Schaltung zusätzliche Mischprodukte. Haben wir also eine Verkopplung zwischen Ein- und Ausgang eines analogen Verstärkers und „passt“ die Phasenbedingung beginnt das ganze Gebilde zu schwingen.

Das „HF-Händchen“ des Entwicklers analoger Schaltungen basiert also auf dem Wissen, derartige Verkopplungen zu vermindern, passend und phasenrichtig gegenzukoppeln und zu filtern, um Schmutzeffekte zu vermeiden. Damit erreicht man, dass beispielsweise das Ausgangssignal eines Senders nicht mehr unerwünschte Anteile enthält als es die aktuelle Vorschriftenlage erlaubt.

Ärgerliche Effekte wie die Temperaturabhängigkeit von Bauelementen, Variation der Parameter über die Betriebsspannung, (thermisches) Rauschen und Alterung von Bauelementen erschweren dem Analogentwickler das Leben zusätzlich.

Wie gut haben es da doch die Digitaltechniker: Um eine logische Null von einer Eins zu unterscheiden muss der Signalpegel eine bestimmte Schwelle über- oder unterschreiten. Tut er das nicht, bleibt das Ausgangssignal in seiner ursprünglichen Lage und damit faktisch ungestört.

In 3,3 V Logiksystemen liegt diese Schwelle bei ungefähr der halben Betriebsspannung. Vom typischen low-Pegel von 0,4 V aus gesehen müssten dem Nutzsignal also schon massive Störsignale überlagert werden, also Spannungsspitzen von mehr als einem Volt, damit eine Verfälschung des Zustands auftritt.

Damit lassen sich Schaltungen und integrierte Schaltkreise viel einfacher störungssicher realisieren, ohne dass man in der Serienfertigung Schaltungen individuell und mithin teuer nachgleichen müsste.

Von Designfehlern einmal abgesehen ist Software in der Fertigung weitaus einfacher zu handhaben. Sie kennt keine Alterung, keine Temperatur- und Spannungsabhängigkeit und muss daher während der Lebensdauer des Gerätes nicht abgeglichen werden. Man spielt sie in der Fertigung auf das Gerät und ist fertig. So lautet zumindest die Theorie.

Wir werden gleich ein paar Fallstricke kennen lernen, über die man auch in der heilen digitalen Welt stolpern kann.

Die Grenze zwischen den Welten

Wie bringt man analoge Signale in die digitale Welt. Die Antwort ist einfach: Mit dem *Analog-Digital-Wandler* (ADC). Der ADC setzt Spannungswerte an seinem Eingang in Bitmuster um. Diese repräsentieren im einfachsten

Fall Zahlenwerte die der Amplitude der Eingangsspannung entsprechen. Umgekehrt formt der *Digital-Analog-Wandler* (DAC) aus digitalen Bitmustern wieder analoge Pegel.

Die Soundkarte im PC beispielsweise enthält von jeder Sorte gleich zwei Stück, nämlich für jeden der Stereo-Kanäle einen ADC und einen DAC.

Jeder dieser Wandler besitzt zwei wichtige Kenngrößen, die *Abtastrate* (SR) und die *Auflösung*.

In den allermeisten Fällen wandelt der ADC das analoge Eingangssignal nämlich in regelmäßigen zeitlichen Intervallen um. Die *Abtastrate* bestimmt wie viele Wandlungen pro Sekunde vorgenommen werden.

Die *Auflösung* wiederum gibt an wie viele unterschiedliche Zahlenwerte das digitale Wandlungsergebnis enthält. Der einfachste denkbare ADC wäre ein Komparator. Sein Ausgang kann die Werte Null oder Eins annehmen, je nach dem, ob der Eingangspegel eine bestimmte Schwelle über- oder unterschreitet. Ein Komparator wäre demnach also ein „Ein-Bit"-ADC mit $2^1 = 2$ Stufen.

Typische Werte kostengünstiger handelsüblicher Wandler sind 8, 10 oder 12 Bit, bei denen das Ergebnis 256, 1024 oder 4096 unterschiedliche Stufen annehmen kann. Derartige ADC sind bereits häufig in Ein-Chip-Mikroprozessoren integriert. Je nach Abtastrate werden inzwischen sogar bereits ADC mit 32-Bit beworben [ADC32].

Beim CD-Spieler und in PC-Soundkarten kommen heute 16 Bit-Wandler mit 65536 Stufen zum Einsatz.

High-Tech Soundkarten bringen es mit vertretbarem Preis bereits auf 24 Bit, womit 2^{24} und damit 16 Millionen Stufen unterschieden werden können. Bei 32 Bit wären das gar über 4 Milliarden unterschiedlicher Werte.

Gehen wir einmal davon aus, der ADC würde Eingangspegel *linear* und ohne Betrachtung eines Vorzeichens in Zahlenwerte umwandeln.

Dann würde bei kurzgeschlossenem Eingang, d.h. bei einer Eingangsspannung von Null Volt am Ausgang der Zahlenwert Null erscheinen. Der größte Zahlenwert von 255, 1023 oder 65535, allgemein also $2^n - 1$ träte dann bei der höchsten spezifizierten Eingangsspannung auf, welche wir als U_0 bezeichnen wollen.

Dann kann man sagen, dass bei einer gegebenen Anzahl *n* an Bits in einem Abtastwert der kleinste unterscheidbare Spannungswert am Eingang wie folgt bestimmt werden kann: $U_{\min} = U_0 \cdot \frac{1}{2^n}$ Dieser Wert wird als *Dynamik* bezeichnet und bestimmt das Verhältnis zwischen kleinster Stufe und dem gesamten Wandlungsbereich.

Die Dynamik wird wie üblich in dB angegeben: $\frac{D}{db} = 20 \cdot \log\left(\frac{U_0}{U_{\min}}\right) = 20 \cdot \log\left(2^n\right) = n \cdot 20 \cdot \log(2)$. Wie man sieht, erhält man damit die Faustformel die besagt, dass uns jedes Bit an Auflösung des ADC ungefähr 6 dB an Dynamik bringt[1].

Bei 16 Bit Auflösung ergibt sich beispielsweise eine Dynamik von 96 dB. Dies ist die theoretische Grenze beim CD-Spieler.

Mit acht Bit erhält man eine größtmögliche Dynamik von 48 dB, was für Anwendungen beim Telefon gerade einmal ausreicht. So einfach ist das.

Die erwähnte *Abtastrate* hingegen bestimmt die höchst mögliche Frequenz, welche der ADC verarbeiten kann. Es gibt dazu eine Bedingung in der Nachrichtentechnik, die als Abtasttheorem nach Shannon bekannt[2] ist. Dieses besagt, dass im Normalfall[3] die Abtastrate eines ADC mindestens doppelt so groß gewählt werden muss wie die *höchste auftretende Frequenzkomponente* im Eingangssignal.

Die mathematische Herleitung würde den Rahmen dieser Einführung bei Weitem sprengen, jedoch kann man sich vorstellen, dass bei einem *unendlich* lange dauernden, sinusförmigen Signal irgendwann einmal alle Amplitudenwerte durchlaufen, wenn man in regelmäßigen Zeitabständen Proben entnimmt, wobei aber die Zeit zwischen den Probe-Entnahmen nicht genau mit der Signalfrequenz übereinstimmen darf. Würde man aber seltener abtasten, so verliert man Information[4]!

Hier ein erstes Beispiel: Über einen ideal bandbegrenzten Telefonie-Kanal werden Frequenzanteile zwischen 300Hz und 3,4 kHz transportiert. Die höchste Frequenzkomponente läge damit unter 4 kHz und wir dürfen dieses Signal unter Beachtung des Shannon-Kriteriums mit 8000 Samples pro Sekunde abtasten und umwandeln.

Bei einer angenommenen Dynamik von 48 dB kämen wir demnach mit einer AD-Wandlung mit einer Auflösung von acht Bit aus. Diese acht Bit umfassenden Zahlenwerte treten bei einer Abtastrate von 8 kSamples/s 8000 Mal pro Sekunde auf. Damit ließe sich solch ein Signal mit einer Bitrate von 8*8000 Bit/s also mit 64 kBit/s übertragen.

---

[1] 20*log(2) ≈ 20*0,3010 ≈ 6

[2] Eigentlich als Nyquist-Shannon-Abtasttheorem bekannt. Claude Elwood Shannon, ein amerikanischer Mathematiker formulierte es 1948

[3] Lassen Sie uns die so genannte „Unterabtastung" (Subsampling) für den Moment ignorieren.

[4] Bitte verraten Sie mich nicht bei Prof. Dr. Wolff aus Karlsruhe, der mir vor vielen Jahren die mathematische Seite der Signaltheorie hatte beibringen wollen und für meine anschaulichen aber unpräzisen Vergleiche vermutlich kein Verständnis hätte…

Klingt das vertraut? Ein einzelner ISDN-Kanal besitzt die Übertragungsgeschwindigkeit von 64 kBit/s[5].

Nun zum umgekehrten Weg. Der *Digital-Analog-Wandler* (DAC) wandelt digitale Zahlenwerte in analoge Ausgangssignale, beispielsweise eine Spannung um. In den allermeisten Fällen findet man am ADC einen Eingang, der mit V_{REF} bezeichnet ist. Diese Referenzspannung definiert den höchsten Ausgangsspannungswert der beim wertgrößten Digitalwort mehr oder weniger skaliert am Ausgang zum Vorschein kommt.

Legt man Digitalworte am Eingang an, welche einen kleineren Zahlenwert repräsentieren wird das Ausgangssignal entsprechend reduziert. Für lineare DAC mit einer Auflösung von *n* Bit gilt: $U_A(b) = U_{REF} \cdot 2^{b-n}$. Die Dynamik lässt sich genau so wie für den ADC angeben.

Dynamische Überlegungen

Was wir im vorangegangenen Abschnitt gesagt haben bedarf ein paar kleinen aber wichtigen Ergänzungen.

Die Einhaltung des bereits erwähnten Abtasttheorems ist schwieriger als man gemeinhin denkt, denn es gilt für alle im Eingangssignal auftretenden Frequenzkomponenten. Doch können wir nur bei völlig reinen sinusförmigen Signalen die Frequenzkomponenten wirklich abschätzen.

In der Praxis aber dominieren vom reinen Sinus abweichende Kurvenformen: Ist das Sinussignal beispielsweise in der Amplitude beschnitten, so enthält das Signal neben der Grundschwingung zusätzliche Oberwellen in unterschiedlichen Amplitudenanteilen.

Ein Rechtecksignal mit einer Frequenz f_0 = 1,5 kHz kann zum Beispiel nicht einfach mit 8 kSamples/s digitalisiert werden. Die zweite Oberwelle ($3*f_0$ = 4,5 kHz) würde nämlich bereits das Shannon-Kriterium verletzen.

Doch wenn das passiert, so würden sich Mischprodukte mit der Abtastrate bilden und so Frequenzanteile vortäuschen, welche im Ursprungssignal gar nicht vorhanden sind. Im Ergebnis verfälschen diese „*Artefakte*“[6] dann das Ergebnis der Wandlung komplett.

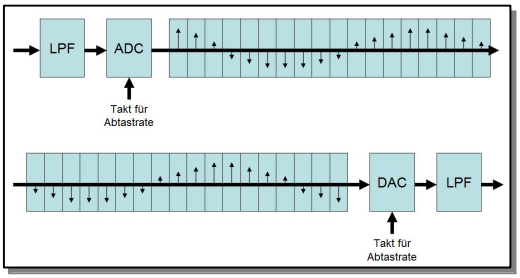
Deshalb wird im Regelfall vor dem ADC bzw. dem Abtast-Halteglied ein analoges Tiefpassfilter angebracht, welches die Frequenzanteile jenseits der Shannon-Grenze bedämpft und damit das „*Aliasing*“[7] verhindert.

Umgekehrt muss auch am Ausgang des DAC ein Tiefpass-Filter vorgesehen werden. Die Ausgangsspannung des DAC ändert sich nämlich in Stufen immer dann, wenn sich das Binärwort am Eingang ändert. Wegen der endlichen Auflösung mit der jeweiligen Zweierpotenz wäre das Ausgangssignal im dynamischen Fall für die meisten Anforderungen zu „rauh“. Eine nachfolgende Integrationsstufe – nichts anderes ist gewöhnlich ein Tiefpass - glättet das Ausgangssignal, indem es die Oberwellen entfernt und damit aus der Treppenfunktion wieder ein kontinuierliches Signal „rekonstruiert“.

Trotzdem ist eine Restwelligkeit am Ausgang nie ganz zu vermeiden. Ein digitaler Sinusgenerator wird diesbezüglich immer schlechtere Eigenschaften besitzen als zum Beispiel ein rein analog arbeitender Wien-Brücken-Oszillator.

Wie nutzen wir die Soundkarte im PC

Eins gleich vorneweg: Für wirklich gute digitale Empfänger eignet sich die Sound-Karte eines üblichen PC nicht wirklich. Viele Entwickler, die sich ernsthaft mit digitaler Empfangstechnik beschäftigen haben hier schon Lehrgeld bezahlt. Neben der Auflösung und der Abtastrate muss ein AD-Wandler nämlich auch gute Linearität und Intermodulationseigenschaften besitzen. Außerdem muss das analoge „Frontend“ bis hin zur Stromversorgung gerade bei höheren Auflösungen über gute Fremdsignaleigenschaften verfügen. Was nützen uns 96 dB Dynamik und eine Auflösung von 1,5µV wenn sich unser Eingangssignal im Störnebel im Millivolt-Bereich bewegt.



Doch für den Amateur, der Komplexität gegen leichte Implementierung aufwiegt und sich zum Beispiel mit einfachen Modulations- und Demodulationsverfahren beschäftigen möchte, bietet die Soundkarte des PC einen raschen und einfachen Einstieg.

Die Soundkarte des PC kombiniert zwei ADC und zwei DAC mit Taktgenerator und einem Pool von Speichern in den die Abtastwerte eingeschrieben werden und aus denen auch digitale Signale wieder ausgelesen werden können.

Leider ist der Ausgang der Sound-Karte nicht gleichspannungsgekoppelt. Doch folgt auch bei diesem DAC der Ausgang dem anliegenden Binärwort. Legen wir also beispielsweise bei einem vorzeichenlosen 16-Bit-DAC mit maximaler Ausgangsspannung von einem Volt den binären Wert 0x7FFF ($32767_d = 2^{15}-1$) an den Digitaleingang an, so würde am Ausgang ungefähr die halbe Spannung erscheinen.

---

[5] Ich habe bei der Dynamik aus Gründen der Anschaulichkeit ein Wenig vereinfacht: ISDN nutzt in Wirklichkeit eine nicht-lineare Zuordnung zwischen Amplitude und Binärwert und erreicht damit eine von der Aussteuerung abhängige Dynamik.

[6] von lat. „ars“ Kunst, d.h. ein künstlich entstandener Anteil

[7] von lat. „alias“ „mit anderem Namen…“

Der Maximalwert von einem Volt würde bei der Bitkombination 0xFFFF ($65535_d = 2^{16}-1$) geliefert. Die kleinste Schrittweite wäre demnach $^{1V}/_{65536} \approx 15\mu V$.

Möchte man nun aber ein zeitvariantes Signal am Ausgang haben, so muss man hintereinander verschieden große Binärwerte anliefern. Deshalb besitzt die Sound-Karte einen Speicher. In diesen Speicher schreibt man per Programm eine Folge von Zahlenwerten die man am Ausgang sehen möchte. Ein Taktgenerator sorgt dann dafür, dass die Werte mit festem zeitlichem Abstand hintereinander dem DAC angeboten werden. Dies führt dann am Ausgang des DAC zu einer über der Zeit veränderlichen Spannung. Sie sehen wie einfach das im Prinzip ist.

Die CPU muss lediglich rechtzeitig für Nachschub sorgen und „rasch genug“ Binärworte nachschieben, bevor der gesamte Speicherinhalt im ADC verschwunden ist. In der Praxis macht man das entweder über ein Fifo oder – wie bei den Soundkarten üblich – mit zwei „Wechselpuffern“, also Speichern von denen immer einer von der CPU beschrieben wird, während der andere gleichzeitig ausgelesen wird. Ist er leer, so tauscht man die Speicher einfach aus. Nun hat die CPU wieder so lange Zeit Nachschub zu liefern wie die Soundkarte zum Abspielen des Speicherinhalts benötigt.

Reservieren wir also 2 kByte bei einer 16-Bit-Soundkarte als Speicher und verlangen dabei eine Abtastrate von 8000 Samples pro Sekunde, so bestünde der Bedarf von ungefähr 2 Byte/Sample * 8kSamples/Sekunde, also für eine konstante Datenrate von 16 kByte/s.

Stehen im Puffer aber nur 2 kByte zur Verfügung, so reicht das lediglich für (2kByte / 16000 Byte/s), also für $^1/_8$ Sekunde. Die CPU muss also spätestens nach 125 ms nachlegen um einen kontinuierlichen Datenstrom am Ausgang zu gewährleisten.

Beim ADC funktioniert das ähnlich. Ein Taktgenerator stößt zu regelmäßigen (zeitlich äquidistanten) Zeitpunkten eine A/D-Wandlung an und legt das digitale Ergebnis in einem Puffer ab. Ist dieser voll, so wird auf den alternativen Puffer umgeschaltet und die CPU darüber informiert, dass sie sich den ersten Pufferinhalt abholen kann.

So sauber und leicht verständlich das Verfahren auch ist: In Echtzeit-Szenarien hat es eine Schwachstelle. Es erzeugt nämlich Verzögerungen. Behalten wir unser Beispiel mit den 2kByte und 8kSampls/s bei, so wird die CPU prinzipbedingt erst nach 125 ms über die Abtastwerte informiert, nämlich dann, wenn der Puffer voll ist.

Wollte man diese Abtastwerte nun einfach wieder ausgeben, beispielsweise um ein simples digitales Telefon zu bauen, so würde die Ausgabe des Speicherinhaltes bei obiger Implementierung wiederum 125 ms benötigen, selbst wenn die eigentliche Übertragung verzögerungsfrei erfolgen würde.

Wir erhalten also ein unvermeidliches Delay von mindestens 250 ms zwischen Ein- und Ausgang, welches lediglich durch die Zwischenspeicherung der Abtastwerte entsteht.

Musikfreunde, die Sound-Samples abspielen wird das nicht weiter auffallen. Das Musikstück beginnt eben ein wenig später. Auch bei reiner Halbduplex-Übertragung stört die Verzögerung kaum.

Doch wenn wir eine Anwendung mit Vollduplex im Auge haben wie beim Telefon, so hört sich diese prinzipiell nicht zu verhindernde Verzögerung unangenehm an.

An diesem Problem hatten Voice-Over-IP-Anwendungen früher erheblich gekrankt.

Noch ein Wort zu den Abtastraten: Historisch bedingt haben sich Abtastraten wie 44100, 22050 und 11025 Samples/s etabliert. Ersterer Wert stammt von der Standard-CD. Woher diese seltsamen Zahlenwerte kommen wird rasch klar, wenn man sich die Primfaktoren anschaut, aus denen sie gebildet werden.

Die Zahl 44100 kann man nämlich aus dem Produkt $2^2 * 3^2 * 5^2 * 7^2$ bilden. Wollen wir die Frequenz 44100 Hz also auf niederere Werte herunterteilen, so stehen uns eine ganze Reihe von Möglichkeiten offen passende Teilerfaktoren zu bilden. Wir werden diesem Prinzip unter dem Begriff der „Dezimation“ später wieder begegnen.

Für Anwendungen in der Datenübertragung über schmalbandige Kanäle wählt man gerne die niederste verfügbare Abtastrate, denn letztendlich sollen die Abtastwerte durch eine Schaltung oder einen Digitalrechner ja weiter verarbeitet werden.

Bei 44100 Samples/s kommt am ADC alle 22,6µs ein neuer Abtastwert zum Vorschein. Ist das mit einem Signalprozessor oder einem modernen PC eigentlich kein Problem, so stoßen kleinere Mikroprozessoren oder Single-Chip-Implementierungen ggf. schon an ihre Leistungsgrenzen.

Unser erster digitaler Sinusgenerator

Dies alles bringt uns zu unserem ersten Beispiel. Lassen Sie uns versuchen, einen Generator für sinusartige Ausgangssignale zu entwickeln.

Wir können davon ausgehen, dass die Soundkarte uns eine beliebige Kurvenform erzeugt, sofern wir nur die passenden Abtastwerte im Speicher ablegen. Allerdings muss die resultierende Kurvenform dem Abtasttheorem entsprechen, darum kommen wir nicht herum!

Wollten wir also einen Fernschreibsender bauen, so müssten wir dazu Töne in Form sinusförmiger Signale erzeugen.

Für das Beispiel wollen wir eine vorzeichenhafte 16-Bit Ausgabe mit 11025 Samples pro Sekunde verwenden. Das sind Parameter, wie dies jede normale Soundkarte beherrscht.

„Vorzeichenhaft“ bedeutet in diesem Zusammenhang, dass die Binärwerte zwischen -32767 über Null bis +32767 reichen können. Diese Darstellung passt besser zu unserer Anschauung eines Signals, dessen Amplitudenwerte zwischen $-U_0$ und $+U_0$ liegen können.

Wir wollen erst einmal einen Ton von 1200 Hz für die Dauer von 5 Sekunden erzeugen. Fünf Sekunden bei 11025 Abtastwerten pro Sekunde ergeben ausmultipliziert 55125 Abtastwerte.

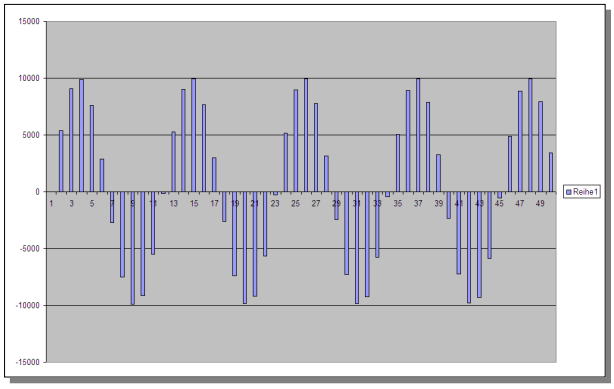
Bei 16 Bit/Sample, also 2 Byte pro Abtastwert müssen also 110250 Byte erzeugt und gespeichert werden.

Wie kann man nun eine Folge von Werten erzeugen, die hintereinander abgespielt die Punkte auf einer Sinuskurve bilden?

Als kontinuierliche Zeitfunktion würde die Vorschrift wie folgt lauten $u(t) = a \cdot \sin(\omega t)$

Dabei ist $\omega = 2\pi f$ mit *f* als Frequenz in Hz, bzw. $^1/_s$. Die Zeit *t* bildet die Variable.

Wir kennen die Sinusfunktion als „zyklische“ Funktion: Startet das Argument bei Null liefert die Funktion ebenfalls 0 als Ergebnis. Nimmt das Argument den Wert $^\pi/_2$ an, so erreicht das Funktionsergebnis den Wert +1, um bis π wieder auf Null abzufallen. Das Spiel geht weiter bis $^3/_2\,\pi$ wobei das Funktionsergebnis den Wert -1 erreicht, bis es schließlich beim Argument von 2π wieder zu Null wird. Damit beginnt das Spiel von Neuem.



Startet *t* nun von Null an, so beginnt der Zyklus bei 0. Wir erreichen den Endwert des Zyklus 2π, wenn $f * t = 1$ ist. Bei der Frequenz f=1000Hz (oder 1000 $^1/_s$) haben wir bereits bei 10^{-3} Sekunden das Ende des Zyklus erreicht.

Nach einer tausendstel Sekunde haben wir also eine komplette Periode hinter uns. In einer Sekunde werden also 1000 Schwingungen ausgeführt.

In der digitalen Verarbeitung haben wir es aber mit diskreten Werten zu tun. Wie funktioniert das dann?

Aus unserem Beispiel mit 11025 Abtastungen pro Sekunden geht hervor, dass eine Sekunde Signaldauer aus genau 11025 Werten besteht.

Bei einer gedachten Signalfrequenz von 1000 Hz haben wir per Definition 1000 Schwingungen pro Sekunde, für die wiederum 11025 Abtastwerte zur Verfügung stehen. Auf eine Schwingung entfallen demnach im Mittel ein wenig mehr als 11 Abtastwerte.

```
nAbtastrate         = 11025;
fFrequenz = 1000.0;
for (n=0; n<nAbtastrate; n++)
   {
   y[n] =
     (fAmplitude * sin((2*pi*fFrequenz*n) / nAbtastrate));
   }
```

Ersetzen wir die laufende Zeit durch den Zähler der Abtastwerte *n* dividiert durch die Abtastrate *N*, so erhalten wir: $y(n) = a \cdot \sin(\omega \cdot \frac{n}{N}) = a \cdot \sin(2\pi \frac{f}{N} \cdot n) \cdot$

Für n gleich Null beginnt der Zyklus. Er endet, wenn der Zähler n die Anzahl der Abtastwerte pro Sekunde erreicht hat. Wir sehen, dass bereits ein neuer Zyklus beginnt, wenn n>11 ist.

Nun zur Amplitude: Sowohl in der Beziehung für die kontinuierliche Spannung u(t) als auch der für die diskreten Abtastwerte y(n) haben wir das Ergebnis der Sinusfunktion mit einem Faktor a multipliziert.

Die Sinusfunktion liefert schließlich nur Werte im Bereich {-1 ... +1}. Um daraus beispielsweise Abtastwerte mit einer Auflösung von 16 Bit zu bekommen müssen wir mit einem passenden Faktor skalieren. Ein möglicher Wert wäre dabei 32767. Das Funktionsergebnis würde dann Werte im Bereich {-32767 ... +32767} liefern und aus der Soundkarte kämen das ±1V an Signalpegel.

```
nAbtastrate         = 11025;
fFrequenz = 1000.0;
fAmplitude          = 10000.0;
for (n=0; n<nAbtastrate; n++)
   {
   double fA;
   // Alle 100 Abtastwerte die Amplitude umschalten
   if ((n%100)==0)
      bToggle = !bToggle;
   // Amplitude beeinflussen
   fA = bToggle ? fAmplitude : (fAmplitude/4.0) ;
   y[n] = ( fA * sin( (2*pi*fFrequenz*n) / nAbtastrate) );
   }
```

Ist uns das zu viel, so müssen wir lediglich den Faktor verkleinern. Damit verringert sich proportional die Aussteuerung. Obiges Beispiel benutzt ein *a* mit dem Wert 10000. Damit liefert die Soundkarte nur noch den Bruchteil ($^{10000}/_{32767}$*1V) also ungefähr ±300mV.

Wir haben also soeben ganz nebenbei einen Pegelregler erfunden. Dieser wird uns beispielsweise auch für eine Amplitudenmodulation nützlich sein.

Doch Achtung! Die erzeugten Amplitudenwerte sind diskret. Das heißt, am Ausgang des DAC tritt eine „stufige“ Treppenfunktion auf. Wir brauchen einen geeigneten analogen Tiefpass um sie passend zu glätten.

Die Sound-Karte bringt diesen bereits mit. Wollen wir aber einen Sender mit einem beliebigen DAC bauen, so müssen wir das Ausgangssignal von den unerwünschten Oberwellen befreien. Sonst bekommen wir Ärger.

Sinusgenerator mit Modulation

Einfache binäre Modulationsarten wie ASK, FSK (A1A, F1A) oder auch 2PSK, sowie deren „analoge“ Schwestern AM und FM (A3E, F3E) sind mit unserem Ansatz ebenfalls bequem herzustellen.

Letztendlich muss nur der Koeffizient für die Amplitude *a* oder die Frequenz *f* zeitvariant manipuliert werden: $y(n,m) = a(m) \cdot \sin(2\pi \frac{f}{N} \cdot n)$ erzeugt eine Wertefolge, welche in der Amplitude moduliert ist.

Analog dazu liefert die Funktion $y(n,m) = a \cdot \sin(2\pi \frac{f(m)}{N} \cdot n)$ die passende frequenzmodulierte Wertefolge.

Mit der Funktion $y(n,m) = a \cdot \sin(2\pi \frac{f}{N} \cdot n + \varphi(m))$ erhielte man eine in der Phase modulierte Folge[8].

Obiger Codeausschnitt zeigt am Beispiel der ASK, wie einfach diese Manipulation sein kann: Nach jeweils 100 Samples wird die Amplitude zwischen 100% und 25% hin und hergeschaltet.

Fallstricke

Unsere soeben beschriebenen Verfahren besitzen jedoch ein paar kleine aber wichtige Einschränkungen über die wir jetzt reden müssen:

Als Erstes ist sicherlich die Zeit ein Problem. Genauer gesagt, ist es oft die Laufzeit der Sinus-Funktion und der Fließkommarechnung und damit die Performance der gesamten Anwendung.

Also wird man in vielen Fällen versuchen, mit im Voraus berechneten so genannten „Lookup“-Tabellen und mit ganzzahligen Werten zu arbeiten, vor allem, wenn man kleinere Mikroprozessoren einsetzt.

Kritisch ist auch die Wahl der Abtastfrequenz. Aus Gründen der Rechenleistung sind wir bestrebt, diese so niedrig wie möglich zu wählen.

Je niederer die Abtastrate jedoch gegenüber der zu erzeugenden Frequenz ist, desto weniger Stützstellen wird die erzeugte Kurvenform besitzen und desto „stufiger“ wird sie am Ausgang des D/A-Wandlers erscheinen.

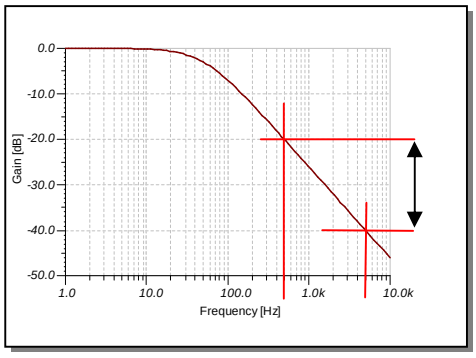
Jedoch wissen wir, dass jede „Ecke“ in der Kurvenform letztendlich die Reinheit des Signals beeinträchtigt. Also gilt es einen Kompromiss zu finden. Stufige, amplitudenbegrenzte Signale enthalten Oberwellen.

Daher werden wir am Ausgang des DAC praktisch immer ein *analoges* Tiefpassfilter benötigen, um eben diese „Ecken und Kanten“ zu glätten und das Sinussignal zu rekonstruieren (Rekonstruktionsfilter).

Zum anderen gilt noch heute, was bereits früher für analoge Fernschreibkonverter galt. Die Umschaltung der Frequenz muss phasenrichtig, d.h. ohne Amplituden- und Phasensprünge erfolgen. Das lässt sich programmtechnisch aber ganz gut in den Griff bekommen.

Digitale Filter

Bisher haben wir immer von der Erzeugung analoger Signale mit Hilfe des Digitalrechners gesprochen und bereits den Analog-Digital-Wandler als Bestandteil der Soundkarte kennengelernt. Nun wollen wir uns die andere Seite anschauen und sehen, was wir mit den digitalisierten Werten anfangen können.



Stellvertretend wollen wir uns zwei wichtige Techniken näher betrachten: Die digitale Filterung und die Umwandlung der Signale aus dem „Zeitbereich“ in den „Frequenzbereich“ um das Spektrum eines Signals zu berechnen.

Im Grundsatz gelten für digitale Filter die gleichen Überlegungen wie für ihre analogen Brüder.

Wichtige Kenngrößen sind die Grenzfrequenz ω_g sowie die „Ordnung“ des Filters, welche das Verhalten jenseits der Grenzfrequenz anzeigt.

In der üblichen doppelt-logarithmischen Darstellung erscheint die Filterkurve eines RC-Tiefpasses erster Ordnung beispielsweise wie nebenstehend dargestellt.

Betrachtet man dessen Übertragungsfunktion, so sinkt die Verstärkung an der Grenzfrequenz gegenüber dem Durchlassbereich auf -3dB bzw. und ab dort um 6dB pro Oktave bzw. 20dB pro Dekade ab.

Braucht man eine höhere Steilheit, so muss man ein Filter höherer Ordnung realisieren. Hiermit lassen sich im Sperrbereich Dämpfungen von *n * 20 dB pro Dekade* erreichen, wobei *n* für die Ordnung des Filters steht. Dann kommen aktive analoge Filter ins Spiel. Aktive analoge Filter sind in [TIETZE] und in anderen Quellen ausführlich beschrieben.

---

[8] Phasenmodulation erzeugt man in der Praxis nicht so, sondern durch einen Ansatz mittels komplexer Zahlen, welcher viel einfacher zu implementieren ist.

Doch wir bekommen nichts geschenkt! Je steiler Filter im Sperrbereich sind, desto mehr muss man sich mit Überschwingern und dem „Klingeln“ von Filtern herumschlagen. Das sind Eigenschaften, die alle Filter besitzen, gleichgültig, wie sie auch immer implementiert sein mögen.

Bessel-Tiefpässe besitzen geringe, Tschebyscheff-Tiefpässe große Überschwinger, dafür besitzen sie im Sperrbereich eine höhere Steilheit. Die Filtertypen unterscheiden sich in Ihren „Koeffizienten“.

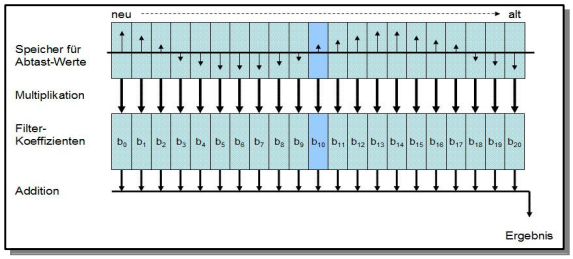
Gegen das Klingeln, die Gruppenlaufzeitverzerrungen und ähnliches helfen auch digitale Filter nichts. Es ist ein weit verbreiteter Irrtum zu glauben, man könne digitale Filter steiler bauen als analoge.

Der wahre Vorteil liegt vielmehr in der leichteren Implementierung.

Jeder, der sich schon einmal mit analogen Filtern auseinandergesetzt hat kennt die Tücken. Da ist zum einen die Bauelemente-Auswahl. Braucht man wirklich exakte Frequenzen, dann ist häufig die Beschaffung der passenden Werte ein Problem.

Obiges Filter besitzt eine Grenzfrequenz ω_g von 50Hz. Mit $\omega_G = \frac{1}{R \cdot C}$ und dem willkürlich gewählten C=0,1µF brauchen wir aber einen Widerstand von 31830Ω. Und dies ist lediglich ein Filter erster Ordnung. Bei höheren Ordnungen werden wir meist auch exotische Werte für die Kondensatoren bekommen, was die Bauelementebeschaffung erschwert.

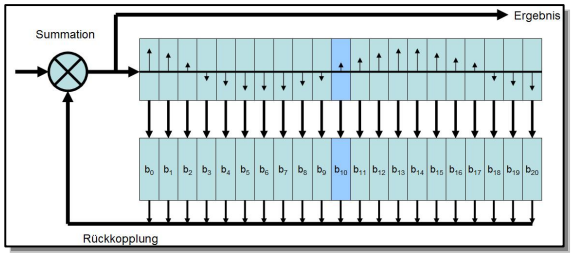
Denkt man dann noch an Bauelementetoleranzen – vor allem die der Kondensatoren, sowie Alterung und Temperaturgang wird es endgültig unhandlich.



Filter bis zur vierten Ordnung lassen sich mit kaskadierten Operationsverstärkern trotz aller Widrigkeiten noch einigermaßen sicher aufbauen. Braucht man in einer Anwendung aber *wirklich* hohe Ordnungen um große Steilheiten zu erreichen machen uns analoge aktive Filter mehr und mehr Probleme. Hier helfen uns die digitalen Implementierungen. Wie in der analogen Technik gibt es Filter ohne und mit Rückkopplungen.

Aus Erfahrung weiß man, dass sich mit Filter mit Rückkopplung größere Steilheiten erreichen lassen, jedoch mit dem Pferdefuß, dass diese Filter instabil werden können und damit zur Selbsterregung neigen.

Die gutmütigeren Varianten ohne Rückkopplung besitzen eine endliche Impulsantwort. Das bedeutet, dass die Auswirkungen eines Impulses am Eingang nach einer endlichen Zeit von selbst wieder ausklingen. Im englischen bezeichnet man dies als „*finite impulse response*“ oder kurz FIR.



Im Gegensatz dazu besitzen IIR-Filter eine durch die Rückkopplung potentiell unendlich lange andauernde Impulsantwort („*infinite impulse response*“). Diese bieten uns höhere Steilheiten, können jedoch instabil sein.

Beide Filtertypen lassen sich programmtechnisch verblüffend einfach realisieren. Nehmen wir einmal an, dass die Abtastwerte vom ADC bzw. der Sound-Karte in einen Speicher hineinlaufen.

Beim FIR-Filter ist das einfach ein eindimensionales Array.

Die Werte kommen von links und schieben sich nach rechts durch den Speicher hindurch, so dass ganz links der neueste und ganz rechts der älteste Wert stehen. In unserem (willkürlich gewählten) Beispiel seien gerade 21 solcher Werte gespeichert worden.

Nun wird jeder der Werte mit dem zu seinem Platz gehörenden Koeffizienten multipliziert: Der neueste Wert auf dem Platz ganz links mit dem Wert b_0, der nächste mit b_1 bis hin zum ältesten Wert ganz rechts mit b_{21} .

```
double FirFilter(double *pfFilter,
            const double *pfKoeffiz,  int nLaenge)
   {
   int i;
   double fErgebnis = 0.0;

   for(i=0; i<nLaenge; i++)
      fErgebnis += (*pfFilter++) * (*pfKoeffiz++);
```

Die Ergebnisse aller Multiplikationen werden aufsummiert und ergeben den ersten gefilterten Wert. Wie Sie sehen wird beim FIR-Filter nichts aus dem Signalpfad rückgeführt. Alle Werte wandern vom Eingang lediglich zum Ausgang.

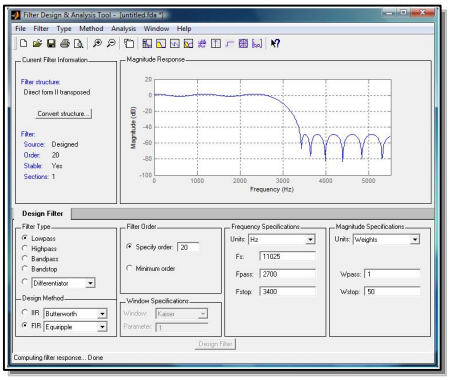
Nun füllt man den nächsten Abtastwert ganz links nach. Alle gespeicherten Werte rutschen nach rechts und der zuvor älteste Wert fällt nach rechts aus dem Speicher heraus.

Nun wird wieder multipliziert und addiert und man erhält den nächsten gefilterten Wert. Das war auch bereits die ganze Rechenvorschrift.

Ähnlich einfach programmiert sich das IIR-Filter, obgleich der Anwender hier sorgfältiger beim Filterdesign vorgehen muss, damit die Stabilität erhalten bleibt.

Jetzt bleibt eigentlich nur noch die Frage zu klären, woher wir die Filter-Koeffizienten bekommen.

Die mathematische Herleitung ist zwar nicht wirklich schwer, jedoch manchmal ziemlich trickreich und sprengt daher den Rahmen dieser Einführung. Sowohl in den bereits zitierten Standardwerken als auch im Internet sind für den Interessierten mehr oder minder brauchbare Darstellungen enthalten.



Für den reinen Anwender gibt es entweder gedruckte Tabellenwerke oder aber zum Teil sehr bequeme Filterentwurfsprogramme, die uns die Koeffizienten direkt liefern.

Das „*Filter Design and Analysis Tool*“ aus der Mathlab-Suite ist beispielsweise ein bei Ingenieuren und Studenten bekanntes Werkzeug. Selbst exotische Anforderungen lassen sich damit modellieren.

Nebenstehendes Design beschreibt zum Beispiel einen (stabilen) Butterworth-Tiefpass für den Sprachbereich als FIR-Filter zwanzigster Ordnung mit einer Dämpfung im Sperrbereich von mehr als 50 dB, welches in analoger Technik nicht besonders sinnvoll zu realisieren wäre, jedoch als digitale Implementierung problemlos zu erreichen ist.

```
const double a[21] =
 {
  0.02415237206547,  0.05685787886241,
  0.03415401461216, -0.0332744964021,
 -0.02784127660223,  0.05204706058397,
  0.02861031785369, -0.09825259360331,
 -0.03019649273871,  0.315616943063,

  0.5307343338446,

  0.315616943063,   -0.03019649273871,
 -0.09825259360331,  0.02861031785369,
  0.05204706058397, -0.02784127660223,
 -0.0332744964021,   0.03415401461216,
  0.05685787886241,  0.02415237206547
 };
```

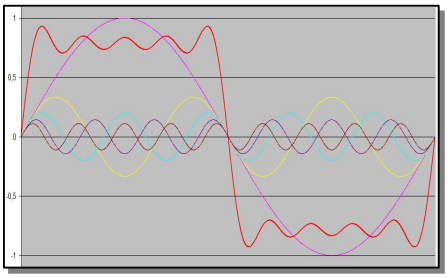
Freundlicherweise liefern uns Programme wie Mathlab häufig bereits C-Header-Dateien, in denen die Koeffizienten als Array von Konstanten abgelegt sind. Schaut man sich die Koeffizienten genauer an so sieht man, dass das Array symmetrisch ist.

In obigem Fall ist $a_0 = a_{20}$, $a_1 = a_{19}$ und so weiter. Koeffizient a_{10} bildet dabei die „Mitte“.

Änderungen der Filtercharakteristik, beispielsweise der Wechsel des Filtertyps oder der Grenzfrequenzen ergibt einfach einen neuen Satz an Koeffizienten.

Selbst wenn die dazu gehörende Theorie manchmal etwas trickreich ist: Die Anwendung digitaler Filter ist viel einfacher als es gemeinhin angenommen wird.

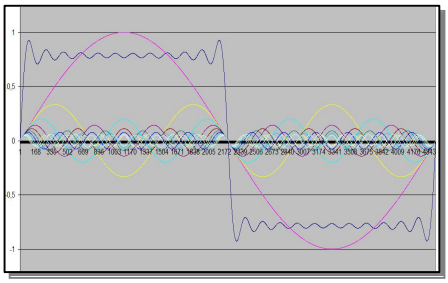
Spektralanalyse / Spektralsynthese



In der digitalen Signalverarbeitung gibt es noch weitere Beispiele zu Sachverhalten, die beliebig kompliziert oder auch einfach und anschaulich dargestellt werden können.

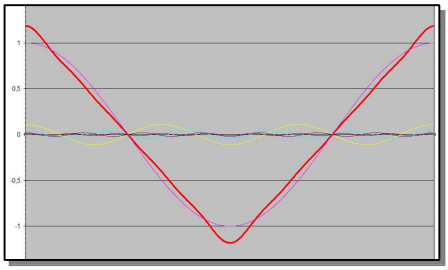
In diesem Fall sprechen wir von der Fourier-Analyse, welche erstmals zu Beginn des 19. Jahrhunderts von dem französischen Physiker und Mathematiker Jean Baptiste Joseph Fourier[9] beschrieben und angewandt worden ist.

In Kürze besagt diese, dass fast alle periodischen Kurvenformen durch Summation von Sinus- und Cosinus-Funktionen synthetisiert werden können.



Nebenstehend habe ich zwei Beispiele aufgeführt. Eine periodische Rechteck-Funktion kann durch folgende Reihenentwicklung nachgebildet werden: $f(t) = \frac{4}{\pi}\left(\sin\omega t + \frac{1}{3}\sin 3\omega t + \frac{1}{5}\sin 5\omega t + \frac{1}{7}\sin 7\omega t + \ldots\right)$.

Mit nur vier Summationsgliedern $\left(\frac{4}{\pi}, \frac{4}{3\pi}, \frac{4}{5\pi} \text{ und } \frac{4}{7\pi}\right)$ in unserem Beispiel besitzt die Rechteck-Kurve eine recht große Welligkeit.



Fügt man aber weitere Summanden mit höheren Frequenzanteilen hinzu, zum Beispiel bis n=23, so wird die Kurve immer „glatter“. Würde man *unendlich* viele Frequenzanteile aufsummieren, so erhielte man schließlich das Abbild einer echten Rechteck-Funktion[10].

Mit folgender Reihe erhält man eine periodische Dreiecksfunktion:

$$f(t) = \frac{4}{\pi}\left(\cos\omega t + \frac{1}{3^2}\cos 3\omega t + \frac{1}{5^2}\cos 5\omega t + \frac{1}{7^2}\cos 7\omega t + \ldots\right).$$

Sie ist jedem Musiker von gezupften Saiteninstrumenten (Gitarre, Cembalo) wohlbekannt.

---

[9] Jean Baptiste Joseph Fourier, 21.3.1768 bis 16.5.1830; Mathematiker und Physiker; beschrieb u. a. bereits 1822 den Treibhauseffekt

[10] Den so genannten „Gibbs“-Effekt, welcher an Kanten zu Überschwingern, dem „*ringing*“ führt, habe ich für diese Überlegung weggelassen.

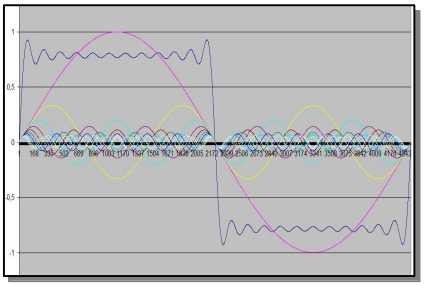
Selbst *unsymmetrische* periodische Funktionen wie Sägezahnkurven und sogar (periodische) Pulsfolgen lassen sich durch Aufsummieren der mit den Koeffizienten gewichteten Frequenzanteile nachbilden.

In diesem Fall muss man gegebenenfalls die Phasenlagen der einzelnen Frequenzanteile gegeneinander verschieben, was in der Praxis jedoch problemlos möglich ist.

Allgemein formuliert man also $f(t) = \sum_{n=1}^{\infty} a_n \cos(n\omega t + \varphi_n)$, was zwar vielleicht etwas gewöhnungsbedürftig aussieht, den Sachverhalt jedoch ganz gut beschreibt.

Anwendungen für die Synthese gibt es genug: Die in den sechziger Jahren in die Mode gekommene „elektronische Musik“ sowie MIDI Wavetable Synthesizer benutzen die Signalsynthese zur Nachbildung von Musikinstrumenten. Und wie jeder Funkamateur weiß, entstehen in sinusförmigen Signale mithin unerwünschte Oberwellen wenn man sie ‚clippt‘[11], also rechteckförmig macht!

Dreht man den Gedankengang um, so lassen sich beliebige periodische Signale in die gewichteten Summen sinusförmiger Signale zerlegen.

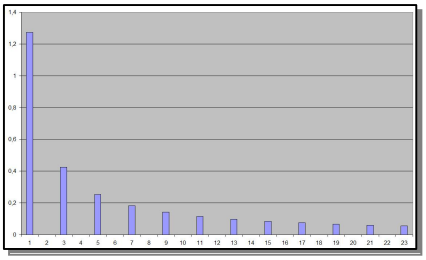


Aus einem Frequenzgemisch wird damit ein „Nebeneinander“ unterschiedlicher Sinussignale, die sich als Spektrum darstellen lassen.

Man spricht bildhaft von der Transformation der Signale aus dem „Zeitbereich“ in den „Frequenzbereich“.

Im Zeitbereich werden die Signale wie auf einem Oszilloskop als Amplitudenwerte in zeitlicher Abfolge dargestellt.

Das dazu gehörende Spektrum zeigt dann die Signale als Amplitude über die jeweilige Frequenz.



In nebenstehendem Beispiel wurde ein Rechteck-Signal aus der Grundwelle und zweiundzwanzig Oberwellen approximiert. Man kann nun die relativen Amplituden der Grund- und Oberwellen als Linienspektrum darstellen.

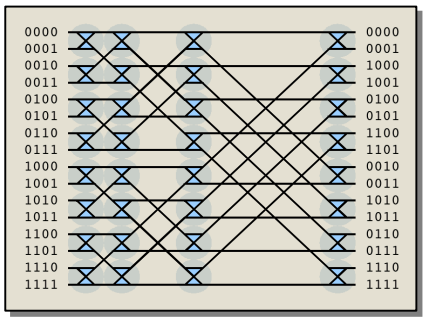
Die „Höhe“ der jeweiligen Spektrallinien im Frequenzbereich entspricht dabei dem jeweiligen Fourier-Koeffizienten.

Wenn wir also ein Verfahren finden um aus einer Folge von Abtastwerten diese Fourier-Koeffizienten zu bestimmen, so wären wir sogar in der Lage ein beliebiges Spektrum zu berechnen.

Die Antwort zu diesem Problem liegt in der so genannten *Diskreten Fourier-Transformation* (DFT) und ihrer rechentechnisch einfacheren Variante der „schnellen Fourier-Transformation“ *(fast fourier transformation*, FFT).

Der Begriff „diskret“ kommt daher, dass die DFT bzw. die FFT mit *zeitdiskreten* Werten – also mit unseren zu festen Zeitpunkten abgetasteten Werten – arbeitet und nicht mit (mathematischen) kontinuierlichen Funktionen. Die FFT reduziert den Aufwand für die Berechnungen dadurch, dass stufenweise Zwischenergebnisse wieder verwendet werden.

Allgemein gesprochen ist die Aufgabe der FFT, eine Abfolge von Abtastwerten in diskrete Speicherzellen, den so genannten „bins“ (Eimer) umzufüllen. Jedem „bin“ ist eine Spektrallinie zugeordnet.



Quelle: Wikipedia, DL5MDA

Der „Butterfly“-Graph illustriert das prinzipielle Vorgehen, wobei die programmtechnische Umsetzung wieder erstaunlich simpel zu bewerkstelligen ist, sofern der Rechner schnelle Fließkomma-Multiplikationen durchführen kann.

Es gibt eine Reihe abgewandelter Algorithmen, welche die zur Transformation notwendigen Fließkomma-Multiplikationen verringern, damit das Verfahren schneller auch auf kleinen Mikroprozessoren gerechnet werden kann. Normale PC können FFT im Audio-Bereich in Echtzeit problemlos durchzuführen.

Die FFT geht von einer bestimmten Anzahl an Abtastwerten aus und errechnet daraus die Spektrallinien.

Wie beim digitalen Filter werden dazu aufeinanderfolgende Abtastwerte gespeichert. Im Regelfall entspricht diese Anzahl gespeicherter Abtastwerte – man nennt diese Anzahl auch die „Länge der FFT“ - einer Zweierpotenz.

Der Abstand der Frequenzlinien nach erfolgter Transformation ergibt sich aus der Abtastrate, mit der die Abtastwerte erfasst wurden, dividiert durch die „Länge“ der FFT.

---

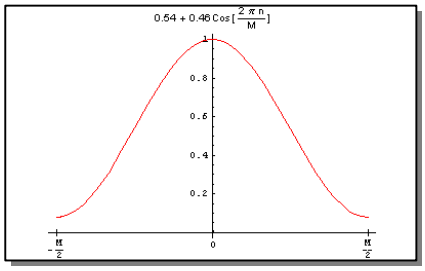
[11] Von engl. To clip: Abschneiden. D.h. Signale die in der Amplitude beschnitten sind

Eine FFT mit einer Länge von 1024 Punkten in Verbindung mit einer Abtastrate von 11025 Samples pro Sekunde ergibt wiederum 1024 Frequenzlinien mit einem Abstand von ungefähr 10,8 Hz. Selbstverständlich muss bei der Abtastung das inzwischen wohl bekannte Shannon-Nyquist-Kriterium eingehalten werden.

Benötigt man nicht solch eine feine Aufteilung so empfiehlt es sich, die Abtastrate zu reduzieren. Alternativ kann man aber auch die Abtastwerte „dezimieren[12]“.

Man lässt dabei in regelmäßigen Abständen einfach Abtastwerte weg. Da dies jedoch ein „stufigeres“ Ergebnis zur Folge hätte, lässt man die dezimierten Werte einfach ein digitales Tiefpassfilter geeigneter Grenzfrequenz, das „Dezimationsfilter“ durchlaufen und stellt so die Nyquist-Bedingung wieder her, um das bereits weiter oben beschriebene „Aliasing“ zu verhindern.

Den letzten Begriff, den wir im Zusammenhang mit der FFT erwähnen müssen ist die „Fensterfunktion“.



Hamming Fenster, Quelle: Wikipedia, STUDI111

Die FFT berechnet das Spektrum lediglich aus einem Ausschnitt des Gesamtsignals, im Beispiel oben aus 1024 Abtastwerten. Bei 11025 Abtastwerten pro Sekunde sind das gerade einmal knapp 93 Millisekunden.

Das Spektrum bildet sich – zumindest mathematisch – jedoch aus einem *unendlich* ausgedehnten Signal.

Durch die Begrenzung der FFT auf ihre Länge verlieren wir also Information. Das Ergebnis würde nur dann stimmen, wenn eine komplette Signalperiode in die Länge der FFT fallen würde, was nur selten der Fall ist.

Das heißt, dass ohne weitere Maßnahmen das Signal durch ein „Rechteck“-Fenster beschnitten wäre, welches in unserem Beispiel 1024 Abtastwerte lang ist. Anteile des Signals vor dem Beginn dieses „Sehschlitzes“ entfallen komplett.

Beginnt der Abtastbereich nun zufällig an einer Stelle in der das abgetastete Signal ein Maximum hat sieht es für die FFT sieht es so aus, als ob am Beginn der Funktion ein steiler Sprung steht. Sprünge bedingen jedoch Oberwellen, welche das Spektrum verfälschen.

Die „Fensterung“ mit einer bewerteten Funktion soll diesen Effekt abmildern. Doch auch sie verfälscht das Ergebnis ein wenig. Korrekt wäre es, das gesamte Signal in die FFT einzubeziehen was jedoch nicht praktikabel ist.

Die Wahl der geeigneten Fensterfunktion hängt stark vom gewünschten Ergebnis ab. Wir kennen das klassische Hamming-Fenster, das Kaiser-Fenster, welches auf Bessel-Reihen aufsetzt und weitere mehr oder weniger einfache Bewertungsfunktionen.

Wir sehen, dass auch in der digitalen Signalverarbeitung ein gewisses „Händchen“, also eigenes Know-How notwendig ist, wie eigentlich überall in der Technik.

Abkürzungen und Begriffe

ADC	Analog Digital Converter; Analog Digital Wandler
CW	Continous Wave. Aussendung eines konstanten Trägersignals. Oft synonym für Morsetelegraphie benutzt.
DAC	Digital Analog Converter.
DFT	Discrete Fourier Transform; Fourier-Transformation von nicht-kontinuierlichen bzw. (zeit-) diskreten Signalen
DVD	Digital Video Disc
EloKa	Elektronische Kampfführung
FFT	Fast Fourier Transform; Schnelle Fourier-Transformation
HF	High Frequency. Hochfrequenz
MIDI	Musical Instrument Digital Interface
SR	Sampling Rate; Abtastrate (eines ADC); Gibt die Anzahl der Konvertierungen pro Sekunde an.
SSB	Single Side Band (Modulation). Einseitenbandmodulation. Wird häufig zur Sprachübertragung auf Kurzwelle benutzt.
VFO	Variable Frequency Oscillator. An diesem stellt man am Empfänger die Betriebsfrequenz ein.
VoIP	„Voice over IP“; Technik zur Übertragung von Sprache über das Internet

Quellen

[TIETZE]	U. Tietze, Ch. Schenk, „Halbleiter-Schaltungstechnik“; Springer-Verlag
[JAYANT]	Jayant, "N.S. Digital Coding of Waveforms”, Prentice-Hall
[MTLB]	Mathlab; The MathWorks Inc.
[FFT]	http://de.wikipedia.org/wiki/Schnelle_Fourier-Transformation
[ADC32]	Sabre[32]: http://www.esstech.com; Signifikanz nicht geprüft!

[12] Angeblich wurden im alten Rom ganze Truppenteile damit bestraft, dass man jeden zehnten aus den Reihen der Soldaten tötete. Ob das nun wirklich stimmt vermag ich nicht zu sagen. Doch der Name „Dezimation“ lässt auf die Zahl 10 schließen.