

Studienarbeit

zur Erlangung des akademischen Grades
Diplom-Ingenieur

Simulationsprogramm zur Veranschaulichung der Signaltheorie Teil2: Darstellung im Frequenzbereich

angefertigt von
cand.-ing. Matthias Lenord

bei
Prof. Dr.-Ing. Heinz Luck

Fachgebiet
Nachrichtentechnik

an der
Universität-Gesamthochschule-Duisburg

Duisburg, August 1993

Vorwort

Ich habe die Vorlesungen Nachrichtentechnik 1-3 bei Prof. Luck besucht. Ich selber gehörte zu den Studenten, die den Stoff zwar sehr interessant, aber besonders in den Anfängen der Vorlesung ziemlich unanschaulich fanden. Dieser Punkt wurde auch genannt, als Prof. Luck am Ende des Wintersemesters 91/92 mit uns Studenten ins Gespräch über die Vorlesung kam. In dieser Vorlesungsstunde entstand dann die Idee, ein Programmpaket zu erstellen, das besonders den Stoff aus dem Beginn der Nachrichtentechnikvorlesung veranschaulichen soll.

Auch bei der Fragebogenaktion unter den Studenten wurde der Wunsch deutlich, anschauliche Arbeitshilfen einzusetzen.

Ich hatte deshalb großes Interesse, einen Teil dieses Programmpakets als Studienarbeit zu übernehmen. Zum einen konnte ich für mich den Vorlesungsstoff noch einmal vertiefen, zum anderen hoffe ich für meine Mitstudenten ein Programm entwickelt zu haben, das wenigstens ein wenig zur Veranschaulichung der Signaldarstellung im Zeit- und Frequenzbereich helfen kann.

Matthias Lenord

Inhaltsverzeichnis

Vorwort	1
Inhaltsverzeichnis.....	2
1. Einführung.....	1
1.1. Aufgabenstellung.....	1
1.2. Das Programm	1
1.3. Die Dokumentation	2
2. Theoretische Grundlagen	3
2.1. Die Fouriertransformation.....	3
2.2. LZI-Systeme	4
2.3. Definition der verwendeten Funktionen.....	4
2.4. Definition der Deltadistribution	7
2.5. Definition der verwendeten Operationen	9
2.6. Die Abtastung.....	11
2.6.1. Die Abtastung im Zeitbereich.....	11
2.6.2. Abtastung im Frequenzbereich.....	13
2.6.3. Zusammenfassung	13
2.7. Die diskrete Fouriertransformation DFT.....	14
2.7.1. Die schnelle Fouriertransformation FFT	16
2.7.2. Theoreme der DFT	17
2.7.3. Vergleich der DFT mit der kontinuierlichen FT	18
3. Benutzeranleitung	20
3.1. Vorbemerkungen	20
3.1.1. Was macht das Programm?	20
3.1.2. Wie die Funktionen verarbeitet werden.....	20
3.2. Installation und Starten des Programms.....	20
3.2.1. Hard- und Softwarevoraussetzungen.....	20
3.2.2. Installation	21
3.2.3. Start des Programms.....	21
3.2.4. Die Datei "STD.SGL"	21
3.3. Größe des Funktionsvektors eingeben	22
3.4. Der Bildschirmaufbau	22
3.4.1. Die Fenster.....	23
3.4.1.1. Funktionen im Fenster	23
3.4.1.2. Aktives Fenster festlegen.....	23
3.4.1.3. Fenster vergrößern	23
3.4.2. Die Statuszeile und Hilfstextzeile	24
3.4.2.1. Die Statusanzeige.....	24
3.4.2.2. Die Anzeige von Fehlern und Warnungen	25
3.4.2.3. Hilfstexte.....	25
3.4.3. Die Menuleiste.....	25
3.4.4. Abfragefenster	25
3.5. Die Fouriertransformation.....	26
3.6. Das DATEI-Menü	26
3.6.1. LADEN.....	26
3.6.2. ALLES SPEICHERN	27
3.6.3. AKTUELLES FENSTER SPEICHERN	28
3.6.4. FUNKTION LÖSCHEN	28

3.6.5. INFORMATION.....	28
3.6.6. NEUSTART.....	30
3.6.7. ENDE.....	31
3.7. Das KONTI.FKT-Menu.....	31
3.7.1. Die Funktion $\sin(z)$	31
3.7.2. Die Funktion $\cos(z)$	31
3.7.3. Die Funktion $\text{rect}(z)$	31
3.7.4. Die Funktion $\text{delta}(z)$	32
3.7.5. Die Funktion Deltafolge.....	32
3.7.6. Die Funktion $\exp(-z^2/b)$	32
3.7.7. Die Funktion $\text{triangel}(z)$	32
3.7.8. Die Funktion $\text{si}(z)$	32
3.7.9. Die konstante Funktion.....	33
3.7.10. Die Sprungfunktion.....	33
3.7.11. Die Funktion $\text{sign}(z)$	33
3.7.12. Die Sägezahnfolge.....	33
3.7.13. Die Rechteckfolge.....	34
3.7.14. Die Rauschfunktion.....	34
3.8. Das DSKR.FKT-Menu.....	34
3.8.1. Die Funktion $\sin(k)$	34
3.8.2. Die Funktion $\cos(k)$	34
3.8.3. Die Funktion $\text{rect}(k)$	34
3.8.4. Die Funktion $\text{delta}(k)$	35
3.8.5. Die diskrete Sprungfunktion.....	35
3.8.6. Die Funktion a-k.....	35
3.9. Das BEARBEITEN-Menu.....	35
3.9.1. Abtasten.....	36
3.9.2. Exchange.....	36
3.9.3. Kopieren.....	37
3.9.4. Die Operation $x(z) + y(z)$	37
3.9.5. Die Operation $x(z) - y(z)$	37
3.9.6. Die Operation $x(z) \cdot y(z)$	37
3.9.7. Die Operation $x(z) \div y(z)$	37
3.9.8. Die Operation $f(b \cdot z)$	38
3.9.9. Die Operation $f(z - c)$	38
3.9.10. Die Operation $f(-z)$	38
3.9.11. Die Operation $b \cdot f(z)$	38
3.9.12. Die Operation $f(z) \cdot \exp(js)$	38
3.9.13. Die Operation $x(z) (*) y(z)$	38
3.9.14. Die Operation $x(-z) (*) y(z)$	38
3.10. Das FENSTER-Menu.....	39
3.11. Das EINSTELL.-Menu.....	39
3.11.1. Neuzeichnen.....	39
3.11.2. Polarkoord.....	39
3.11.3. Kartes. Koord.....	40
3.11.4. Skalierung (manuell).....	40
3.11.5. Skalierung (rechn.).....	40
3.11.6. Definitionsbereich.....	40
3.11.7. Abtastperiode.....	41
3.12. Das MODUS-Menu.....	41

3.12.1. Aufgabe starten.....	41
3.12.2. Aufgabe beenden.....	42
3.12.3. Autoskalierung (an/aus).....	42
3.12.4. Autodefinition (an/aus).....	42
3.12.5. Ausgabegeschwindigkeit.....	42
3.12.6. Diracerkennung.....	43
3.12.7. Gleiche Skal. RE und IM.....	43
3.12.8. Defber. beim Bearbeiten.....	43
3.12.9. Winkeldarstellung.....	44
3.13. Warum sieht nicht alles so aus wie erwartet?.....	44
3.13.1. Funktionen werden periodisch fortgesetzt.....	44
3.13.2. Die Abtastperioden im Zeit und Frequenzbereich.....	45
3.13.3. Probleme beim Umrechnen in einen anderen Definitionsbereich.....	46
3.14. Fehlermeldungen.....	46
3.15. Die Dateiformate.....	49
3.15.1. Das Format für eine Funktion.....	50
3.15.2. Das Format für eine Arbeitsumgebung.....	51
3.15.3. Ein einfaches Format für komplexe Werte.....	53
3.15.4. Ein einfaches Format für einzelne Funktionswerte.....	54
3.15.5. Das Format für eine Aufgabenstellung.....	54
4. Programmbeschreibung.....	56
4.1. Einleitung.....	56
4.1.1. Die Entwicklungsumgebung.....	56
4.1.2. Der wesentlichen Probleme bei der Programmierung.....	57
4.2. Programmaufbau.....	58
4.3. Das Headerfile NT.h.....	59
4.3.1. Die Struktur FUNKTION.....	61
4.4. Das Sourcefile MAIN.C.....	64
4.4.1. Menüpunkt hinzufügen.....	65
4.5. Das Sourcefile GRAFIK.C.....	65
4.5.1. Die Benutzereingabe.....	66
4.5.2. Die Funktionsausgabe.....	67
4.6. Das Sourcefile MAUS.C.....	68
4.7. Das Sourcefile M_FUNK.C.....	68
4.8. Das Sourcefile NEUBAUER.C.....	69
4.9. Das Sourcefile MATHE1.C.....	69
4.9.1. Die Anpassungsfunktion dadjust().....	70
4.9.2. Die diskrete Fouriertransformation dft().....	72
4.10. Das Sourcefile MATHE2.C.....	75
4.11. Das Sourcefile INFO.C.....	76
4.12. Das Sourcefile FILE.C.....	77
5. Zusammenfassung.....	78
5.1. Verbesserungsmöglichkeiten.....	78
5.2. Schlußwort.....	79
6. Verwendete Formelzeichen.....	80
7. Literaturverzeichnis.....	82

1. Einführung

1.1. Aufgabenstellung

Das Thema dieser Studienarbeit ist die Erstellung eines Programms zur Veranschaulichung der Signaldarstellung im Zeit- und Frequenzbereich. Es soll an das bereits erstellte Programm¹ zur Darstellung im Zeitbereich anschließen.

Es soll eine grafische Oberfläche entwickelt werden, mit deren Hilfe der Anwender auf einfache Weise Signalfunktionen und Operationen der Nachrichtentechnik darstellen kann. Es sollen analoge Signale, diskrete Signale und nicht-deterministische Signale implementiert werden. Diese Signale sollen dann im Zeit- und Frequenzbereich modifiziert werden können.

Angewendet werden kann das Programm von Studenten, die ergänzend zu Vorlesung und Übung den Stoff anschaulich vertiefen möchten. Es ist aber auch denkbar, Funktionssample aus der Praxis zu importieren und zu verarbeiten.

1.2. Das Programm

Das Programm ist mit der Programmiersprache C entwickelt worden. Das Kernstück ist die bereits von Andre Neubauer implementierte Fast Fouriertransformation.

Die grafische Oberfläche, die mit Hilfe von Fenstern und Pulldownmenus arbeitet, kann mit der Tastatur oder Maus bedient werden. Dabei wurde Wert darauf gelegt, dem Benutzer möglichst viele Parameter der Funktionsdarstellung zugänglich zu machen, um deren Auswirkung auf die Signaldarstellung kennenzulernen.

In vier Fenstern können zwei Funktionen $x(t)$, $y(t)$, $X(\omega)$, $Y(\omega)$ im Zeit- und Frequenzbereich bearbeitet werden. Dabei steht auch eine Zoomfunktion zur Vergrößerung der Fenster zur Verfügung. Alle Funktionen können entweder nach Betrag und Phase oder Real- und Imaginärteil dargestellt werden.

Es besteht die Möglichkeit alle Funktionen und Parameter in Dateien zu speichern und zu laden. Außerdem können auch fremde Funktionsvektoren importiert werden.

Es können Aufgabenstellungen abgerufen werden, die den Benutzer zum Nachdenken über spezielle Anwendungen in der Nachrichtentechnik anregen sollen. Die Aufgabenstellungen können selbstständig ergänzt und verändert werden.

¹Diplomarbeit von Frank Morgenstern [4]

1.3. Die Dokumentation

Dieses Programm soll Teil eines Programmpaketes sein, das weitere Themen der Nachrichtentechnik beinhaltet. Von daher spielt die Dokumentation eine wichtige Rolle, um nachfolgenden Programmierern den Einblick in das Programm zu erleichtern.

Zur Ausarbeitung gehören deshalb separat noch das Programmlisting und das Referenzhandbuch.

Ich habe mich bemüht das **Programmlisting** einigermaßen ausführlich zu kommentieren (daher auch die Länge des Sourcecodes). Leider ist es nicht immer möglich Vorgänge, die man selbst für einleuchtend hält, in prägnanten Sätzen zu formulieren. Zentrale Programmteile werden deshalb in der Ausarbeitung noch einmal erläutert.

Das **Referenzhandbuch** ist eine alphabetische Liste aller verwendeten Unterprogramme. Zu jedem Unterprogramm wird eine kurze Beschreibung gegeben. Außerdem werden die Übergabeparameter und globalen Variablen aufgezählt.

Diese **Ausarbeitung** gliedert sich in drei Teile:

1. Theoretische Grundlagen:

Dieser Teil enthält alle mathematischen Definitionen und die theoretischen Grundlagen zum Verständnis des Programms. Einige Formulierungen sind dabei mathematisch nicht ganz exakt aber, so hoffe ich, recht anschaulich dargestellt.

2. Benutzeranleitung:

Dies ist die Erklärung wie das Programm benutzt wird. Es wird der Aufbau der grafischen Oberfläche und alle Funktionen erläutert. Dabei wird nicht so viel Wert auf komplizierte Hintergründe gelegt, sondern es werden nur die Problemstellungen erwähnt, die direkten Einfluß auf die Anwendung des Programms haben.

3. Programmbeschreibung:

Hier wird der Aufbau des Programms beschrieben. Einige ausgewählte Programmteile werden jeweils ausführlich beschrieben.

2. Theoretische Grundlagen

2.1. Die Fouriertransformation

Die Fouriertransformation ist eine Integraltransformation, die in der Nachrichtentechnik angewendet wird, da sie anschaulich das Frequenzspektrum einer Zeitfunktion darstellt. Wendet man die Fouriertransformation auf periodische Funktionen an, so erhält man den Spezialfall der Fourierreihenentwicklung.

Die Fouriertransformation ist wie folgt definiert:

$$S(\omega) = \int_{-\infty}^{\infty} s(t) \cdot e^{-j\omega t} dt \quad (1)$$

$$s(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} S(\omega) \cdot e^{j\omega t} d\omega \quad (2)$$

Eigenschaften der Fouriertransformation:

1. Superpositionssatz:

$$a \cdot s_1(t) + b \cdot s_2(t) \xrightarrow{F} a \cdot S_1(\omega) + b \cdot S_2(\omega) \quad (3)$$

2. Ähnlichkeitssatz:

$$s(bt) \xrightarrow{F} \frac{1}{|b|} \cdot S\left(\frac{\omega}{b}\right) \quad (4)$$

3. Verschiebungssatz:

$$s(t - t_0) \xrightarrow{F} S(\omega) \cdot e^{-j\omega t_0} \quad (5)$$

4. Symmetrie:

$$S(t) \xrightarrow{F} 2\pi \cdot s(-\omega) \quad (6)$$

5. Faltung:

$$s_1(t) * s_2(t) \xrightarrow{F} S_1(\omega) \cdot S_2(\omega) \quad (7)$$

6. Multiplikation:

$$s_1(t) \cdot s_2(t) \xrightarrow{F} \frac{1}{2\pi} S_1(\omega) * S_2(\omega) \quad (8)$$

2.2. LZI-Systeme

Ein lineares, zeitinvariantes System (LZI-System) läßt sich mit folgendem Schema anschaulich darstellen:

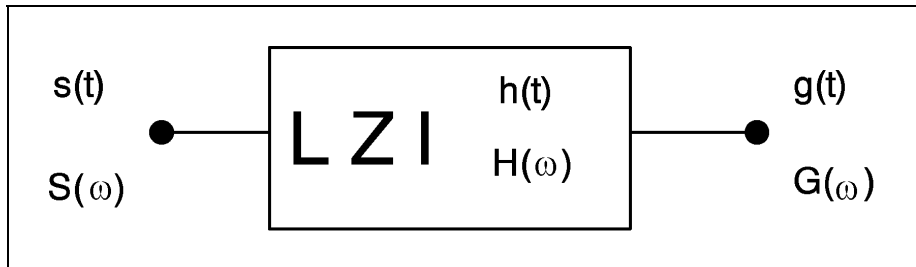


Abb. 1: Schematische Darstellung eines LZI-Systems

Ein lineares, zeitinvariantes System zeichnet sich durch zwei Eigenschaften aus:

- Linearität:

$$\begin{array}{l} \text{aus } s_1(t) \longrightarrow g_1(t) \\ s_2(t) \longrightarrow g_2(t) \end{array} \text{ folgt } a \cdot s_1(t) + b \cdot s_2(t) \longrightarrow a \cdot g_1(t) + b \cdot g_2(t) \quad (9)$$

- Zeitinvarianz:

$$\text{aus } s(t) \longrightarrow g(t) \text{ folgt } s(t - \tau) \longrightarrow g(t - \tau) \quad (10)$$

Ein LZI-System ist vollständig durch seine Stoßantwort $h(t)$ bzw. seine Übertragungsfunktion $H(\omega)$ beschrieben. Für Ein- und Ausgangssignal gilt folgender Zusammenhang:

$$g(t) = s(t) * h(t) = \int_{-\infty}^{\infty} s(\tau) \cdot h(t - \tau) d\tau \quad (11)$$

$$G(\omega) = S(\omega) \cdot H(\omega)$$

Das Programm kann eingesetzt werden, um sich Vorgänge bei LZI-Systemen im Zeit- und Frequenzbereich zu veranschaulichen.

2.3. Definition der verwendeten Funktionen

Im folgenden werden Standardfunktionen definiert. Diese Funktionen sind Grundfunktionen, die im Programm abgerufen werden können. Durch Verknüpfung und Bearbeitung dieser Funktionen können kompliziertere Funktionen zusammengesetzt werden.

Alle Funktionen sind rein reell. Zu jeder Funktion wird die Fouriertransformierte angegeben. Diese wird in Real- und Imaginärteil aufgespalten: $S(\omega) = R(\omega) + jX(\omega)$.

- Die Rechteckfunktion:

$$\text{rect}(t) = \begin{cases} 0 & \text{für } t < -0.5 \\ 1 & \text{für } -0.5 \leq t \leq 0.5 \\ 0 & \text{für } t > 0.5 \end{cases} \quad (12)$$

$$\text{rect}\left(\frac{t}{T}\right) \xrightarrow{F} \begin{cases} R(\omega) = T \cdot \text{si}\left(\omega \frac{T}{2}\right) \\ X(\omega) = 0 \end{cases} \quad (13)$$

- Das Gaußsignal:

$$s(t) = e^{-t^2} \quad (14)$$

$$e^{-\alpha t^2} \xrightarrow{F} \begin{cases} R(\omega) = \sqrt{\frac{\pi}{\alpha}} \cdot e^{-\frac{\omega^2}{4\alpha}} \\ X(\omega) = 0 \end{cases} \quad (15)$$

- Die Sprungfunktion:

$$\varepsilon(t) = \begin{cases} 0 & \text{für } t < 0 \\ 1 & \text{für } t \geq 0 \end{cases} \quad (16)$$

$$\varepsilon(t) \xrightarrow{F} \begin{cases} R(\omega) = \pi \cdot \delta(\omega) \\ X(\omega) = -\frac{1}{\omega} \end{cases} \quad (17)$$

- Die Dreiecksfunktion²:

$$\Lambda(t) = \begin{cases} 1 - |t| & \text{für } |t| \leq 1 \\ 0 & \text{für } |t| > 1 \end{cases} \quad (18)$$

$$\Lambda\left(\frac{t}{T}\right) \xrightarrow{F} \begin{cases} R(\omega) = T^2 \cdot \text{si}^2\left(\omega \frac{T}{2}\right) \\ X(\omega) = 0 \end{cases} \quad (19)$$

²Definition nach Lücke, Signalübertragung, 5. Auflage, S. 2 [1]

- Die Signumfunktion:

$$\operatorname{sgn}(t) = \begin{cases} -1 & \text{für } t < 0 \\ 0 & \text{für } t = 0 \\ 1 & \text{für } t > 0 \end{cases} \quad (20)$$

$$\operatorname{sgn}(t) \xrightarrow{F} \begin{cases} R(\omega) = 0 \\ X(\omega) = -\frac{2}{\omega} \end{cases} \quad (21)$$

- Die si-Funktion:

$$\operatorname{si}(t) = \frac{\sin(t)}{t} \quad (22)$$

$$\operatorname{si}(\omega_0 t) \xrightarrow{F} \begin{cases} R(\omega) = \frac{2\pi}{\omega_0} \operatorname{rect}\left(\frac{\omega}{2\omega_0}\right) \\ X(\omega) = 0 \end{cases} \quad (23)$$

- Die Rechteckfolge:

$$s_r(t) = \sum_{\nu=-\infty}^{\infty} \operatorname{rect}(t - \nu T) \quad (24)$$

T: Periodendauer

$$\sum_{\nu=-\infty}^{\infty} \operatorname{rect}\left(\frac{t - \nu \cdot T}{T_0}\right) \xrightarrow{F} \begin{cases} R_d\left(k \frac{2\pi}{T}\right) = \frac{2\pi \cdot T_0}{T} \operatorname{si}\left(k\pi \frac{T_0}{T}\right) \\ X_d\left(k \frac{2\pi}{T}\right) = 0 \end{cases} \quad (25)$$

T₀: Breite des Rechtecks

Eine periodische Funktion hat eine diskrete Fouriertransformierte. Dieser Zusammenhang wird bei der Rechteckfolge deutlich.

- Die Sägezahnfunktion

$$s_z(t) = \begin{cases} 0 & \text{für } t \leq -\frac{1}{2} \\ 2 \cdot t & \text{für } -\frac{1}{2} < t < \frac{1}{2} \\ 0 & \text{für } t \geq \frac{1}{2} \end{cases} \quad (26)$$

$$s_z\left(\frac{t}{T}\right) \xrightarrow{F} \begin{cases} R(\omega) = 0 \\ X(\omega) = \frac{2}{\omega} \left[\cos\left(\omega \frac{T}{2}\right) - \text{si}\left(\omega \frac{T}{2}\right) \right] \end{cases} \quad (27)$$

- Die Sägezahnfolge:

$$s_f(t) = \sum_{v=-\infty}^{\infty} s_z(t - vT) \quad (28)$$

$$s_f(t) \xrightarrow{F} \begin{cases} R_d\left(k \frac{2\pi}{T}\right) = 0 \\ X_d\left(k \frac{2\pi}{T}\right) = \begin{cases} 0 & \text{für } k = 0 \\ \frac{2 \cdot (-1)^k}{k} & \text{für } k \neq 0 \end{cases} \end{cases} \quad (29)$$

2.4. Definition der Deltadistribution

Der Deltaimpuls wird in einem getrennten Abschnitt behandelt, da er zwar oft wie eine "normale" Funktion gehandhabt wird, aber doch einige besondere Eigenschaften aufweist. Der Definition des Deltaimpulses liegt die Distributionentheorie zugrunde.

- Die Deltadistribution:

Die mathematische Definition erfolgt mit Hilfe des Skalarproduktes:

$$\begin{aligned} \langle \delta, f \rangle &:= f(0) \\ \langle \delta_{t_0}, f \rangle &:= f(t_0) \end{aligned} \quad (30)$$

Dabei wird das Skalarprodukt wie folgt definiert:

$$f(t_0) = \int_{-\infty}^{\infty} f(t) \cdot \delta(t - t_0) dt = \langle \delta_{t_0}, f \rangle \quad (31)$$

Die Deltafunktion ist eine verallgemeinerte Funktion und ordnet jeder Testfunktion $f(t)$ ihren Funktionswert an der Stelle t_0 zu (Ausblendeigenschaft).

Anschaulich kann man den Deltaimpuls folgendermaßen beschreiben:

$$\delta(t) = \lim_{T_0 \rightarrow 0} \left(\frac{1}{T_0} \text{rect}\left(\frac{t}{T_0}\right) \right) \quad (32)$$

Die Deltafunktion ist eine unendlich hohe und unendliche schmale Rechteckfunktion mit dem Flächeninhalt 1.

- Eigenschaften von $\delta(t)$

1. Die $\delta(t)$ -Funktion ist unendlich schmal.

$$\delta(t) = 0 \quad \text{für} \quad t \neq 0 \quad (33)$$

2. Die Fläche, die die Funktion $\delta(t)$ einschließt, hat den Flächeninhalt 1.

$$\int_{-\infty}^{\infty} \delta(t) dt = 1 \quad (34)$$

3. Die Diracfunktion ist linear und homogen.

$$a \cdot \delta(t) + b \cdot \delta(t) = (a + b) \cdot \delta(t) \quad (35)$$

4. Es gilt das Kommutativgesetz.

$$s(t) * \delta(t) = \delta(t) * s(t) \quad (36)$$

5. Der Diracimpuls wirkt bei der Faltung wie ein Verschiebeoperator.

$$s(t - t_0) = s(t) * \delta(t - t_0) \quad (37)$$

6. Die Deltadistribution ist gerade. Es gilt allgemein:

$$\delta(b \cdot t) = \frac{1}{|b|} \delta(t) \quad (38)$$

7. Die Ableitung der Sprungfunktion ergibt den Deltaimpuls.

$$\frac{d}{dt} \varepsilon(t) = \delta(t) \quad (39)$$

8. Die Fouriertransformierte ist eine konstante Funktion:

$$a \cdot \delta(t) \xrightarrow{F} a \quad (40)$$

9. Ausblendeigenschaft:

$$s(t) \cdot \delta(t - t_0) = s(t_0) \cdot \delta(t - t_0) \quad (41)$$

- Die Deltaimpulsfolge:

$$\text{II}(t) = \sum_{\nu=-\infty}^{\infty} \delta(t - \nu T) \quad (42)$$

Die Deltaimpulsfolge spielt bei der Abtastung von Funktionen eine wichtige Rolle. Bei einer Multiplikation einer kontinuierlichen Funktion mit der Deltaimpulsfolge werden jeweils Funktionswerte im Abstand T ausgeblendet.

$$\sum_{n=-\infty}^{\infty} \delta(t - nT) \xrightarrow{F} \frac{2\pi}{T} \sum_{n=-\infty}^{\infty} \delta(\omega - n \frac{2\pi}{T}) \quad (43)$$

2.5. Definition der verwendeten Operationen

Im folgenden Abschnitt werden einige Operationen definiert, die eine wichtige Rolle beim Umgang mit der Fouriertransformation spielen. In der Literatur werden einige Operationen teilweise abweichend definiert. Die folgenden Definitionen sind der Vorlesung NT1 von Prof. Luck³ entnommen.

- Die Faltung:

$$x(t) * y(t) = \int_{-\infty}^{\infty} x(\tau) \cdot y(t - \tau) d\tau \quad (44)$$

Das Faltungsprodukt ist eine für LZI-Systeme allgemein geltende Transformationsgleichung. Die Antwort $g(t)$ eines LZI-Systems mit der Stoßantwort $h(t)$ auf ein Eingangssignal $s(t)$ lautet: $g(t) = s(t) * h(t)$.

Die Faltung ist kommutativ:

$$s(t) * g(t) = g(t) * s(t) \quad (45)$$

Für die Differentiation gilt:

$$(s(t) * g(t))' = s(t)' * g(t) = g(t)' * s(t) \quad (46)$$

³Vorlesungsskript NT1 Prof. Luck [3]

- Die Kreuzkorrelation:

Die Energie eines Signals ist definiert durch die Integration des quadrierten Signals:

$$E_s = \int_{-\infty}^{\infty} |s(t)|^2 dt \quad (47)$$

Ist die Energie endlich, so spricht man von einem Energiesignal:

$$E_s = \int_{-\infty}^{\infty} |s(t)|^2 dt < \infty \quad (48)$$

Um ein Maß für die Ähnlichkeit zweier Energiesignale zu erhalten, wird der normierte Korrelationskoeffizient verwendet:

$$\rho_{sg}^E = \frac{\int_{-\infty}^{\infty} s(t)g^*(t)dt}{\sqrt{E_s E_g}} \quad (49)$$

Bei größter Ähnlichkeit wird der Korrelationskoeffizient 1 und bei größter Unähnlichkeit -1. Er erhält den Wert 0, wenn die beiden Funktionen orthogonal sind.

$$\begin{aligned} \rho_{sg}^E &= 1 \quad \text{für} \quad s(t) = g(t) \\ \rho_{sg}^E &= 0 \quad \text{für} \quad \int_{-\infty}^{\infty} s(t)g^*(t)dt = 0 \\ \rho_{sg}^E &= -1 \quad \text{für} \quad s(t) = -g(t) \end{aligned} \quad (50)$$

Bei der Definition des Korrelationskoeffizienten wird eine feste zeitliche Lage der beiden Funktionen angenommen. Werden die Signale gegeneinander auf der Zeitachse verschoben, so wird die normierte Kreuzkorrelation für deterministische Energiesignale definiert:

$$\rho_{sg}(\tau) = s(t) \otimes g(t) = s(-t) * g^*(t) = \int_{-\infty}^{\infty} s(\tau) \cdot g^*(t + \tau) d\tau \quad (51)$$

Die Kreuzkorrelation ist nicht kommutativ:

$$s(t) \otimes g(t) \neq g(t) \otimes s(t) \quad (52)$$

- Die Autokorrelation:

Wird die Ähnlichkeit einer Funktion zu sich selbst betrachtet, so wird die Autokorrelation gebildet:

$$\rho_{ss}(\tau) = \mathbf{s}(t) \otimes \mathbf{s}(t) = \mathbf{s}(-t) * \mathbf{s}^*(t) = \int_{-\infty}^{\infty} \mathbf{s}(\tau) \cdot \mathbf{s}^*(t + \tau) d\tau \quad (53)$$

$$\rho_{ss}(0) = \int_{-\infty}^{\infty} \mathbf{s}(\tau) \cdot \mathbf{s}^*(t + 0) d\tau = E_s \quad (54)$$

- Das Energiedichtespektrum:

Wird die Autokorrelationsfunktion in den Frequenzbereich übertragen, so erhält man das Energiedichtespektrum. Diese Funktion stellt den Energieinhalt der Funktion über der Frequenz aufgetragen dar:

$$\begin{aligned} \rho_{ss}(t) &\xrightarrow{F} \mathfrak{R}_{ss}(\omega) \\ \mathfrak{R}_{ss}(\omega) &= \mathbf{S}(-\omega) \cdot \mathbf{S}^*(-\omega) = |\mathbf{S}(-\omega)|^2 \end{aligned} \quad (55)$$

Das Energiedichtespektrum reeller Energiesignale ist immer gerade, reell und nicht negativwertig.

2.6. Die Abtastung

2.6.1. Die Abtastung im Zeitbereich

Die diskrete Funktion $s_a(t)$ ergibt sich aus der ursprünglichen Funktion $s(t)$ durch Abtastung.

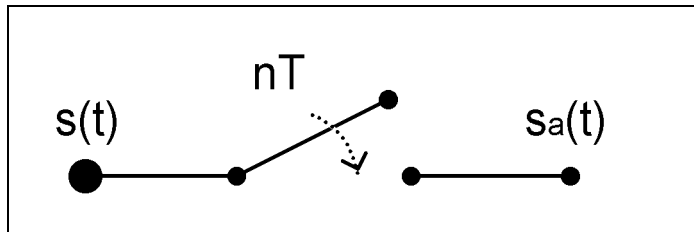


Abb. 2: Der ideale Abtaster

Es gilt:⁴

$$s_a(t) = s(t) \cdot \sum_{n=-\infty}^{\infty} \delta(t - nT) \quad (56)$$
$$s_a(t) = \sum_{n=-\infty}^{\infty} s(nT) \delta(t - nT)$$

Wird jetzt der gesamte Ausdruck unter Verwendung von Glg.(43) fouriertransformiert, so folgt:

$$S_a(\omega) = \frac{1}{2\pi} S(\omega) * \frac{2\pi}{T} \sum_{n=-\infty}^{\infty} \delta(\omega - n \frac{2\pi}{T}) = \frac{1}{T} \sum_{n=-\infty}^{\infty} S(\omega - n \frac{2\pi}{T}) \quad (57)$$

⁴Vorlesungsskript NT2 Prof. Luck [3]

Ein abgetastetes Signal im Zeitbereich führt zu einer periodischen Fortsetzung der Fouriertransformierten des Signals im Frequenzbereich. Die Abbildungen⁵ 3-5 sollen dies veranschaulichen:

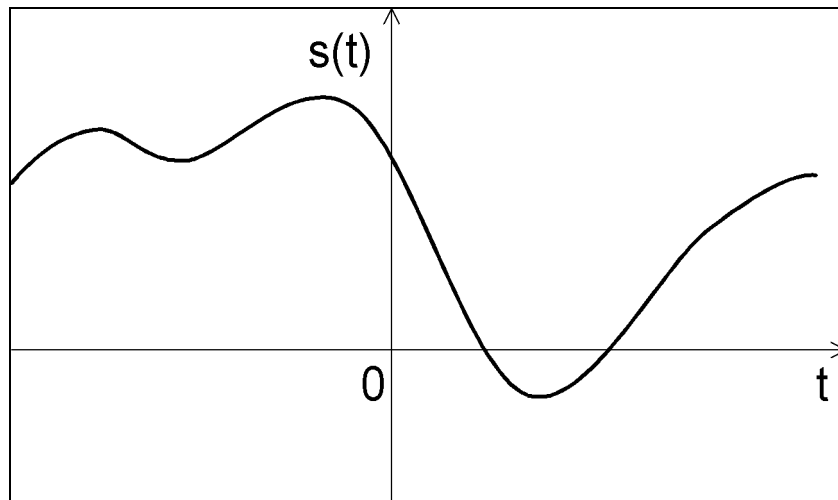


Abb. 3: Das Zeitsignal $s(t)$

Wird die Funktion $s(t)$ jetzt abgetastet, ergibt sich die Funktion $s_a(t)$:

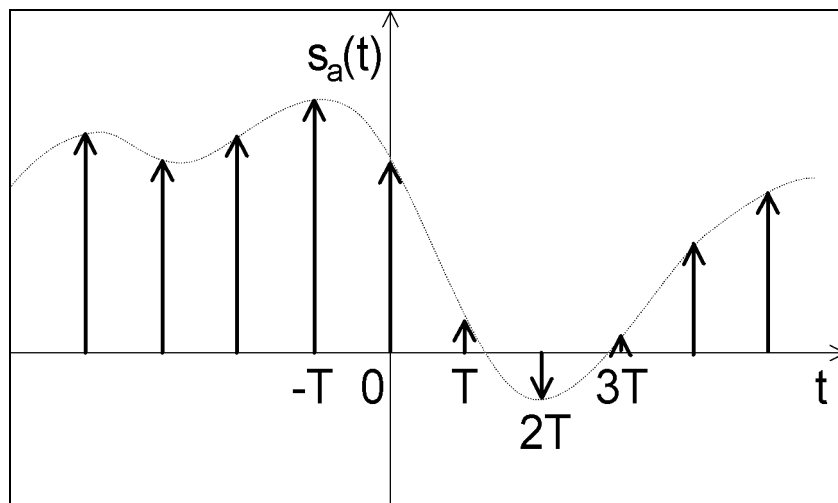


Abb. 4: Das mit der Periode T abgetastete Zeitsignal $s_a(t)$

⁵Lüke, Signalübertragung, 5. Auflage, S.51 [1]

Führt man jetzt eine Fouriertransformation aus, so entsteht das periodische Spektrum der Funktion $S_a(\omega)$:

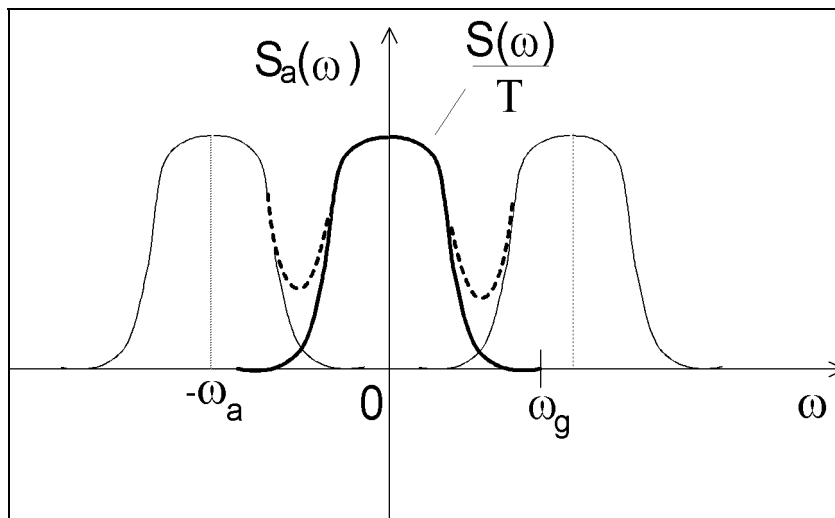


Abb. 5: Das periodische Frequenzsignal

$S_a(\omega)$ entsteht aus der periodischen Fortsetzung der Funktion

$$\frac{S(\omega)}{T} \quad (58)$$

mit der Periodendauer:

$$\omega_a = \frac{2\pi}{T}. \quad (59)$$

An dieser Stelle sei schon einmal auf den Faktor $\frac{1}{T}$ vor der Funktion $S(\omega)$ hingewiesen.

Er spielt bei der Berechnung der kontinuierlichen Fouriertransformation aus der DFT eine wichtige Rolle.

In Abb. 5 wird deutlich, daß sich die periodischen Funktionen überlappen können. Um dies zu vermeiden, muß die die Bedingung:

$$\omega_g \leq \frac{\omega_a}{2} \quad \text{oder} \quad T \leq \frac{\pi}{\omega_g} \quad \text{erfüllt sein.}$$

Die oben gemachte Rechnung gilt demnach nur für bandbegrenzte Signale. Für nicht bandbegrenzte Signale z.B. der $\text{rect}(t)$ -Funktion treten immer Überlappungen auf.

2.6.2. Abtastung im Frequenzbereich

Für die Abtastung im Frequenzbereich läßt sich eine ähnliche Rechnung durchführen wie für die Abtastung im Zeitbereich:

$$S_a(\omega) = S(\omega) \cdot \sum_{n=-\infty}^{\infty} \delta(\omega - n\Omega) \quad (60)$$

mit Ω : Abtastperiode im Frequenzbereich

Wird die inverse Fouriertransformation angewendet, so folgt:

$$s_a(t) = s(t) * \frac{1}{\Omega} \sum_{n=-\infty}^{\infty} \delta(t - n \frac{2\pi}{\Omega}) \quad (61)$$

$$s_a(t) = \frac{1}{\Omega} \sum_{n=-\infty}^{\infty} s(t - n \frac{2\pi}{\Omega}) \quad (62)$$

Ein Signal mit einem diskreten Frequenzspektrum hat ein periodisches Zeitsignal.

2.6.3. Zusammenfassung

Geht man also von einem zeitbegrenzten Signal im Intervall $[t_1, t_2[$ aus, so folgt für die Breite B des Signals:

$$B = t_2 - t_1 \quad (63)$$

Wird das Signal jetzt in N Abtastwerte aufgeteilt, so ergibt sich die Abtastperiode T zu:

$$T = \frac{B}{N} \quad (64)$$

Der rechte Rand t_2 gehört nicht zum Intervall dazu, so daß der letzte Abtastwert bei $t_2 - T$ liegt. Ansonsten müßte durch $N-1$ dividiert werden. Für die Abtastperiode T gilt die Formel (59):

$$T = \frac{2\pi}{\omega_a}$$

Wenn eine Periode ω_a dieses Frequenzspektrums jetzt wieder auf N diskrete Werte aufgeteilt wird, so ergibt sich für die Abtastperiode im Frequenzbereich Ω :

$$\Omega = \frac{\omega_a}{N} = \frac{2\pi}{N \cdot T} \quad (65)$$

Wichtig ist auch hier, daß die rechte Grenze des Intervalls für eine Periode Teil der nächsten Periode ist. Sie wird nicht abgetastet.

Es folgt also für den Zusammenhang der Abtastperioden im Zeit- und Frequenzbereich:

$$\frac{\Omega \cdot T \cdot N}{2\pi} = 1$$

Es gilt also:

$$\Omega \cdot T \cdot N = 2\pi$$

(66)

Dieser Zusammenhang ist sehr zentral bei dem Umgang mit der diskreten Fouriertransformation.

2.7. Die diskrete Fouriertransformation DFT

Da man es bei der Signalverarbeitung mit dem Rechner mit diskreten Signalen zu tun hat, wird die diskrete Fouriertransformation angewendet.

Die DFT ist wie folgt definiert:

$$S_d(k) = \sum_{n=0}^{N-1} s_d(n) \cdot e^{-j \frac{2\pi \cdot n \cdot k}{N}} \quad (67)$$

$$s_d(n) = \frac{1}{N} \sum_{k=0}^{N-1} S_d(k) \cdot e^{j \frac{2\pi \cdot k \cdot n}{N}} \quad (68)$$

Die DFT geht von einem Zeitsignal mit N diskreten Werten aus. Dieses Signal wird periodisch fortgesetzt, so daß sich wieder ein periodisches, diskretes Spektrum im Frequenzbereich ergibt. Für den Zusammenhang zwischen den Abtastperioden im Zeit- und Frequenzbereich folgt oben abgeleiteter Zusammenhang aus Gleichung (66):

$$\Omega \cdot T \cdot N = 2\pi$$

Das folgende Bild 6 soll den Zusammenhang verdeutlichen:

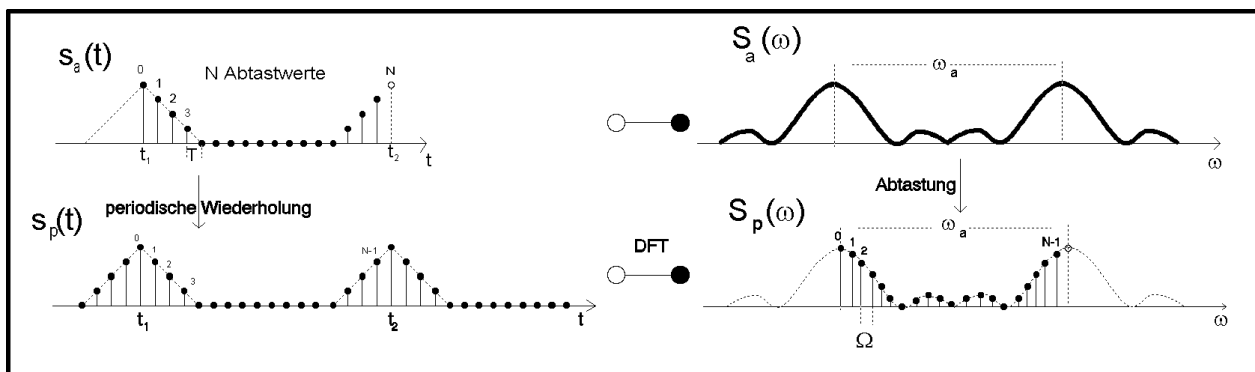


Abb. 6: Zusammenhänge zwischen dem zeitbegrenzten, zeitdiskreten Signal $s_a(t)$ und seinem Spektrum $S_a(\omega)$ sowie dem periodischen, zeitdiskreten Signal $s_p(t)$ und der DFT $S_p(\omega)$

Man geht von einer Funktion $s(t)$ im Intervall $[t_1, t_2[$ aus. Die Funktion $s_a(t)$ ist aus dieser Funktion durch Abtasten mit N Abtastwerten entstanden. Die DFT geht jetzt von der periodischen Fortsetzung dieser Funktion aus $s_p(t)$. Nach der Transformation liegt das periodische, diskrete Spektrum $S_p(\omega)$ vor.

Man könnte auch direkt die Fouriertransformation der Funktion $s_a(t)$ bilden und so die kontinuierliche, periodische Funktion $S_a(\omega)$ erhalten. Tastet man diese Funktion dann mit der Abtastperiode Ω ab, so erhält man das gleiche Ergebnis wie die DFT.

Die DFT geht von einer endlichen Anzahl N an Abtastwerten aus. Daraus folgt, daß Funktionen zeitbegrenzt bzw. frequenzbandbegrenzt sind. Das entspricht einer Multiplikation mit der rect -Funktion. Das folgende Bild⁶ 7 soll zeigen, wie sich dieser Effekt auf die Transformierte auswirkt:

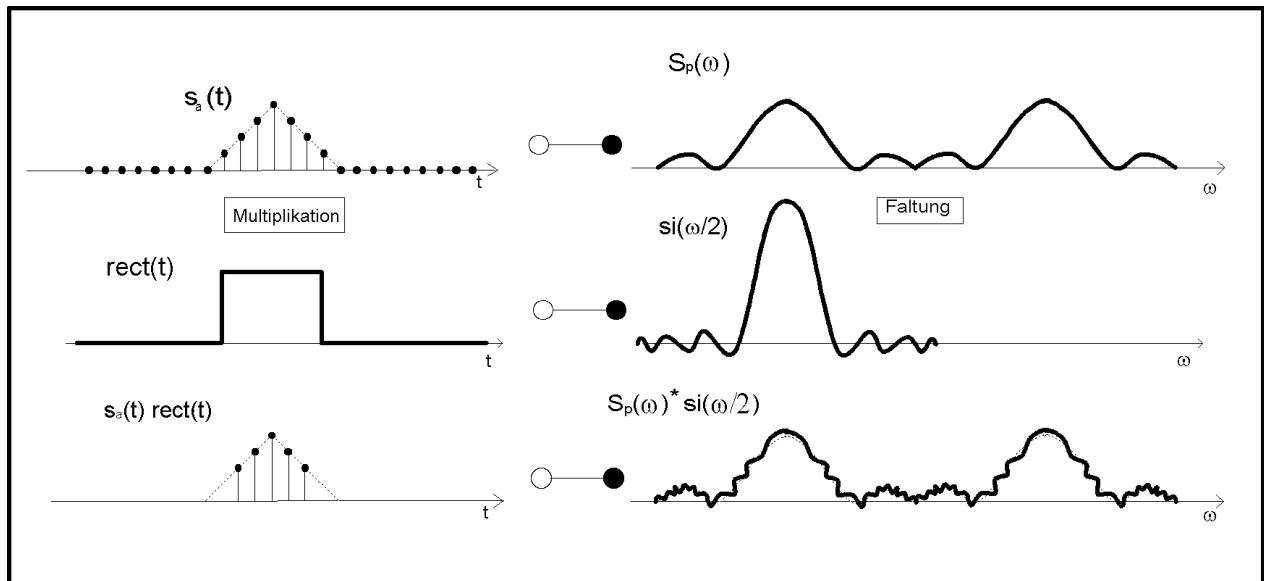


Abb. 7: Multiplikation einer Funktion mit einer Rechteckfunktion

Der Multiplikation mit einer rect -Funktion entspricht im Frequenzbereich eine Faltung mit der si -Funktion. Durch die Bandbegrenzung im Zeitbereich entsteht im Frequenzbereich eine Welligkeit. Diese Welligkeit ist umso geringer je breiter die rect -Funktion ist. Die si -Funktion nähert sich nämlich bei immer breiter werdendem Rechteck immer mehr einem Diracstoß an. Ein unendlich breiter rect entspräche einem Diracimpuls, bei dem dann auch keine Welligkeit mehr entstehen würde.

Zusammenfassend möchte ich drei Effekte der DFT festhalten, die grundlegend bei der Interpretation der Ergebnisse sind:

1. Die DFT geht von abgetasteten, diskreten Funktionen aus. Daraus folgt, daß Überlagerungseffekte entstehen können. Diese können durch eine kleinere Abtastperiode T bzw. F vermieden werden.
2. Die DFT geht von einer endlichen Anzahl Abtastwerte aus. Daraus folgt, daß im transformierten Bereich Welligkeiten entstehen können. Diese können durch einen breiten Definitionsbereich der Funktion verringert werden.
3. Für die Abtastperioden T und Ω im Zeit- und Frequenzbereich gilt folgender Zusammenhang aus Gleichung (66):

$$\Omega \cdot T \cdot N = 2\pi$$

Diese drei Punkte zeigen die Grenzen der DFT. Die ersten beiden Punkte schließen sich gegenseitig praktisch aus. Weil ein größerer Bereich B der Funktion eine größere Abtastperiode T zur Folge hat (Gleichung: (64)):

⁶Brigham, FFT, 4. Auflage, S.115 [2]

$$T = \frac{B}{N}$$

Hier zeigt sich auch, daß alle drei Punkte optimiert werden können, indem eine große Anzahl N an Abtastwerten gewählt. Dem sind aber durch den zu kleinen Computerspeicher und der Rechengeschwindigkeit Grenzen gesetzt.

2.7.1. Die schnelle Fouriertransformation FFT

Die Berechnung der DFT mit dem Digitalrechner erfordert die Durchführung von sehr vielen Multiplikationen. Eine Multiplikation zweier komplexen Zahlen im Fließkommaformat erfordert eine verhältnismäßig lange Rechenzeit, so daß sich die Rechenzeit für die gesamte DFT proportional zur Anzahl der Multiplikationen verhält. Aus diesem Grund findet auf Digitalrechnern die schnelle Fouriertransformation FFT ihre Anwendung.

Die FFT soll an dieser Stelle nicht erklärt werden, sondern nur ein kurzer Einblick in die Idee gegeben werden. Die DFT läßt sich auch als Matrixtransformation schreiben⁷:

$$\begin{aligned}\vec{S}_d &= \vec{T} \cdot \vec{s}_d \\ \vec{s}_d &= \vec{T}^{-1} \cdot \vec{S}_d\end{aligned}\tag{69}$$

Die Transformationsmatrix \vec{T} ist eine NxN-Matrix. Diese Matrix wird faktorisiert, d.h. in n Teilmatrizen zerlegt. Diese Matrizen werden so gewählt, daß die Anzahl der komplexen Multiplikation jeder Matrix minimiert wird. Die Anzahl der Teilmatrizen ist mit der Anzahl an Abtastenwerten wie folgt verknüpft:

$$N = 2^n\tag{70}$$

Daraus folgt, daß die Anzahl der diskreten Werte N entsprechend dieser Beziehung gewählt werden muß.

Für die Anzahl der Multiplikationen der FFT und DFT gilt folgender Zusammenhang:

$$\begin{aligned}M_{FFT} &= \frac{N \cdot n}{2} \\ M_{DFT} &= N^2 \\ \frac{M_{DFT}}{M_{FFT}} &= \frac{2 \cdot N}{n}\end{aligned}\tag{71}$$

Bei N=1024 Abtastwerten ist die FFT demnach im den Faktor 204 schneller. Die direkte Berechnungsmethode ergab auf einem 486'er eine Rechenzeit von ca. 30 Sekunden. Die FFT rechnete dagegen in der Größenordnung von einer zehntel Sekunde.

⁷Brigham, FFT, 4.Auflage, S.182 [2]

Das folgende Bild⁸ soll diesen Zusammenhang noch einmal grafisch veranschaulichen:

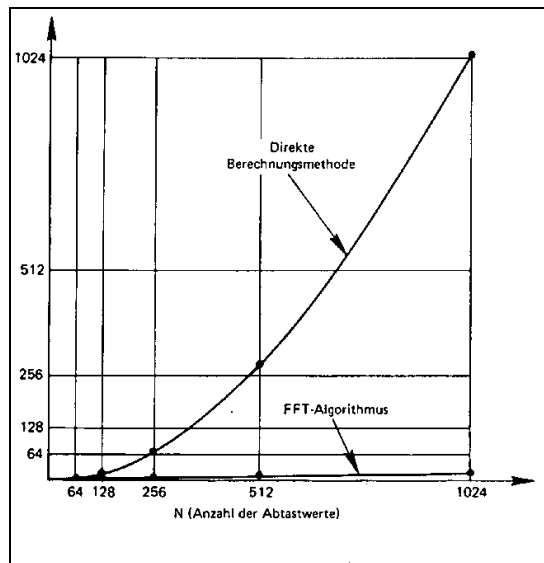


Abb. 8: Anzahl der Multiplikationen bei der FFT und der DFT über der Anzahl an Abtastwerten aufgetragen

2.7.2. Theoreme der DFT

Im diesem Programm finden einige Theoreme der DFT ihre Anwendung, die im Folgenden aufgelistet werden⁹:

- Zeitumkehr:

$$s_d(-n) \xrightarrow{F} S_d(-k) \quad (72)$$

- komplexe s(n):

$$s_d^*(n) \xrightarrow{F} S_d^*(-k) \quad (73)$$

- Symmetrie:

$$S_d(n) \xrightarrow{F} N \cdot s_d(-k) \quad (74)$$

- periodische Faltung:

$$\sum_{m=0}^{N-1} s_d(m) \cdot g_d(n-m) \xrightarrow{F} S_d(k) \cdot G_d(k) \quad (75)$$

- Multiplikation:

$$s_d(n) \cdot g_d(n) \xrightarrow{F} \frac{1}{N} \sum_{m=0}^{N-1} S_d(m) \cdot G_d(k-m) \quad (76)$$

⁸Brigham, FFT, 4.Auflage, S.185 [2]

⁹Lüke, Signalübertragung, 5. Auflage, S.71 [1]

- periodische Verschiebung:

$$s_d(n-m) \xrightarrow{F} S_d(k) \cdot e^{-j \frac{2\pi \cdot m \cdot k}{N}} \quad (77)$$

2.7.3. Vergleich der DFT mit der kontinuierlichen FT

Im folgenden soll der Schritt von der kontinuierlichen FT zur einer diskreten Form gemacht werden. Diese soll dann mit der Definition der DFT verglichen werden.

Gehen wir von der kontinuierlichen FT aus (1):

$$S(\omega) = \int_{-\infty}^{\infty} s(t) \cdot e^{-j\omega t} dt$$

Wird das Integral berechnet, indem die Fläche unter der Kurve $s(t)e^{-j\omega t}$ durch Rechtecke der Breite T approximiert wird, so folgt:

$$S(\omega) = \lim_{T \rightarrow 0} \sum_{n=-\infty}^{\infty} s(nT) \cdot e^{-j\omega \cdot nT} \cdot T \quad (78)$$

Zieht man jetzt die Rechteckbreite T vor die Summe, so ergibt sich:

$$S(\omega) = \lim_{T \rightarrow 0} \left(T \cdot \sum_{n=-\infty}^{\infty} s(nT) \cdot e^{-j\omega \cdot nT} \right) \quad (79)$$

Wir gehen jetzt nicht mehr von unendlich schmalen Rechtecken aus, sondern von Rechtecken der endlichen Breite T. Diese entspricht bei der diskreten Signalverarbeitung der Abtastperiode T. Die Frequenz ω soll jetzt ebenfalls nur noch in diskreten Schritten Ω angegeben werden $\omega = k\Omega$:

$$S(k \cdot \Omega) = T \cdot \sum_{n=-\infty}^{\infty} s(nT) \cdot e^{-j \cdot k \cdot \Omega \cdot nT} \quad (80)$$

Wird jetzt noch der Zusammenhang aus Gleichung (66) $\Omega \cdot T = \frac{2\pi}{N}$ verwendet und der Schritt zu einer endlichen Anzahl Abtastwerte N gemacht, so ergibt sich:

$$S(k \cdot \Omega) = T \cdot \sum_{n=0}^{N-1} s(nT) \cdot e^{-j \cdot \frac{2\pi \cdot k \cdot n}{N}} \quad (81)$$

Zum Vergleich nochmal die Definition der DFT aus (67):

$$S(k \cdot \Omega) = \sum_{n=0}^{N-1} s(nT) \cdot e^{-j \cdot \frac{2\pi \cdot k \cdot n}{N}}$$

Die Gleichung der DFT unterscheidet sich von dem Ergebnis der Herleitung über die kontinuierliche FT durch den Faktor T. Dieser Faktor trat auch in Gleichung (58) bei der periodischen Fortsetzung des transformierten Signals auf.

Im Programm ist die DFT so implementiert, daß sie dem Benutzer wie eine "normale", kontinuierliche FT erscheint. Deshalb muß das Ergebnis der DFT mit dem Faktor T multipliziert werden:

$$\mathbf{S}_{d,\text{kontinuierlicheFT}} = T \cdot \mathbf{S}_{d,\text{DFT}} \quad (82)$$

Eine ähnliche Herleitung für die inverse kontinuierliche FT ergibt:

$$s(nT) = \frac{1}{T \cdot N} \sum_{k=0}^{N-1} S(k \cdot \Omega) \cdot e^{j \frac{2\pi \cdot k \cdot n}{N}} \quad (83)$$

Der Vergleich mit der inversen DFT liefert ein analoges Ergebnis:

$$\mathbf{S}_{d,\text{kontinuierlicheFT}} = \frac{1}{T} \cdot \mathbf{S}_{d,\text{DFT}} \quad (84)$$

Ein Sonderrolle spielt die **Deltafunktion**. Theoretisch wäre es gar nicht möglich eine unendlich schmale Funktion diskret darzustellen. Jeder Funktionswert hat von dem nächsten den Abstand der Abtastperiode T. Wird ein Deltaimpuls diskret dargestellt, so hat er praktisch eine Breite von T. Es gilt also für den diskreten Deltaimpuls $\delta_d(t)$:

$$\delta_d(t) = \delta\left(\frac{t}{T}\right) \quad (85)$$

Wird jetzt die Eigenschaft aus Gleichung (38) verwendet, so gilt:

$$\delta\left(\frac{t}{T}\right) = T \cdot \delta(t)$$

Erscheint die Deltafunktion also im Ergebnis der DFT, so muß hier nicht mehr der Faktor T berücksichtigt werden. Da es schwierig ist in einem diskreten Funktionsvektor eine Deltafunktion zu identifizieren, liegt in diesem Punkt ein Problem bei der Anwendung der DFT in diesem Programm.

Der Faktor T hat auch Bedeutung bei der Berechnung der **diskreten Faltung**. Die Faltung wird in dem Programm über eine Multiplikation der Fouriertransformierten berechnet. Im Frequenzbereich ergibt sich deshalb ein Faktor T^2 . Dieser Faktor wird bei der Rücktransformation einmal gekürzt, so daß das Ergebnis der gesamten Faltung noch mit dem Faktor T multipliziert werden muß.

3. Benutzeranleitung

3.1. Vorbemerkungen

3.1.1. Was macht das Programm?

Das Programm dient zur Veranschaulichung von Signalen im Zeit- und Frequenzbereich. Es können Funktionen grafisch dargestellt werden. Diese Funktionen können auf einfache Weise verknüpft und bearbeitet werden. Dabei kann das Ergebnis sowohl im Frequenz- als auch im Zeitbereich angeschaut werden. Durch Experimentieren und Spielen mit den Funktionen werden Zusammenhänge zwischen Operationen im Zeit- und Frequenzbereich deutlich.

Die Darstellung hat natürlich ihre Grenzen. Sehen Sie dazu im Kapitel "3.13. Warum sieht nicht alles so aus wie erwartet?".

3.1.2. Wie die Funktionen verarbeitet werden

Die Funktionen werden in sog. Funktionsvektoren abgespeichert, d.h. es wird eine bestimmte Anzahl diskreter Funktionswerte in Feldern abgelegt.

Der Bereich der Funktion, der in einem Funktionsvektor gespeichert wird, wird durch den **Definitionsbereich** festgelegt. Dieser Bereich kann vom Benutzer manuell oder vom Programm automatisch festgelegt werden. Sehen Sie dazu in den Kapiteln "3.12.4. Autodefinition (an/aus)", "3.11.6. Definitionsbereich" und "3.11.7. Abtastperiode".

Die Funktionen werden immer nach Real- und Imaginärteil abgespeichert, auch wenn die Darstellung mit Betrag und Phase (polare Koordinaten) eingestellt worden ist. Daraus folgt auch, daß die Funktionen in polaren Koordinaten etwas eingeschränkter bearbeitet werden können.

Bei der Darstellung der Funktionen kann der Ausschnitt der Funktion festgelegt werden, der angezeigt werden soll: die sog. **Skalierung**. Auch hier kann der Benutzer die Skalierung eingeben oder das Programm automatisch eine Skalierung festlegen. Sehen Sie dazu in den Kapiteln "3.12.3. Autoskalierung (an/aus)", "3.11.4. Skalierung (manuell)" und "3.11.5. Skalierung (rechn.)".

Wichtig ist, daß Definitionsbereich und Skalierung nicht gleich sein müssen. Man kann sich z.B. eine Funktion außerhalb ihres Definitionsbereiches anzeigen lassen. Das Resultat ist dann bei nicht periodischen Funktionen eine Gerade $f(t) = 0$.

3.2. Installation und Starten des Programms

3.2.1. Hard- und Softwarevoraussetzungen

Das Programm ist zur Verwendung unter DOS programmiert worden, da dieses Betriebssystem das verbreitetste ist. Es ist unter der Version 5.0 entwickelt und getestet worden. Es müßte aber ohne weiteres auch unter niedrigeren DOS-Versionen laufen.

Bei der Verarbeitung von großen Funktionsvektoren kann es zu Speicherplatzproblemen kommen. Deshalb ist ein möglichst großer und freier Speicher erforderlich.

Zur Ausführung ist mindestens ein 80286-Prozessor erforderlich. Ein mathematischer Coprozessor ist wünschenswert, da bei Berechnungen sehr viele Fließkommaoperationen stattfinden.

Weiterhin arbeitet das Programm im 640x480 VGA Grafikmodus mit 16 Farben. Ein anderer Grafikmodus wird nicht unterstützt, da eine flexible Darstellung von Text und Grafik ein zu großer Programmieraufwand gewesen wären. Die Grafikkarte des Rechners muß diesen Modus deshalb unterstützen.

Eine Maus wird zwar unterstützt, ist aber nicht erforderlich.

3.2.2. Installation

Zur Installation braucht einfach nur das File "SIGNAL.EXE" in ein gewünschtes Verzeichnis kopiert werden. Das Programm ist schon ausführbar. Es bietet sich noch an, Dateien mit Aufgabenstellungen "*.TUT" und Dateien mit gespeicherten Funktionen zu kopieren "*.SGL". Diese können dann als Muster im Programm geladen werden. Sehen Sie auch unter "3.6.1. LADEN", "3.6.2. ALLES SPEICHERN" und "3.12.1. Aufgabe starten".

3.2.3. Start des Programms

Zum Starten wird das Programm "SIGNAL.EXE" aufgerufen. Es kann noch der optionale Parameter **/n** angegeben werden. Er veranlaßt das Programm eine vom Benutzer definierte Datei mit Standardparametern zu laden. Dazu muß die Datei "STD.SGL" im aktuellen Verzeichnis vorhanden sein.

Außerdem kann der Parameter **/b** angegeben werden, um eine kontrastreichere Darstellung zu erhalten. Dieser Parameter ist dazu gedacht, die Ausgabe mit einem Overheadprojektor zu machen.

Vor dem Start sollte in das Verzeichnis gewechselt werden, in dem sich die Datenfiles befinden ("*.TUT" und "*.SGL"), da alle Dateien im aktuellen Verzeichnis gesucht werden..

Tritt beim Starten ein Fehler auf, ohne daß ein Titelbild erscheint, liegt dies sehr wahrscheinlich an der Grafikkarte, die den oben genannten Modus nicht verarbeiten kann. Abhilfe kann hier nur eine neuere Grafikkarte schaffen.

3.2.4. Die Datei "STD.SGL"

Diese Datei kann (muß aber nicht) sich in dem aktuellen Verzeichnis befinden. Dieses File wird nach dem Start automatisch gesucht und geladen, wenn der Parameter **/n** angegeben wird. Der Anwender kann in diesem File eine Arbeitsumgebung abspeichern, die er nach dem Start vorfinden will. Sehen Sie auch im Kapitel "3.6.2. ALLES SPEICHERN".

Tritt nach dem Titelbild ein Dateifehler auf, so liegt das daran, daß die Datei "STD.SGL" zwar vorhanden ist, aber in einem falschen Format vorliegt. In diesem Fall setzt das Programm eigene Standardwerte für die Umgebung. Der Fehler hat sonst keine Auswirkungen und kann ignoriert werden.

3.3. Größe des Funktionsvektors eingeben

Nach dem Start erscheint das Titelbild. Wenn kein Parameter /**n** eingegeben wurde, wird hier die "Anzahl der diskret verarbeiteten Werte" abgefragt. Dies ist die Anzahl der Funktionswerte, die pro Funktion ($x(t)$, $y(t)$, $X(\omega)$, $Y(\omega)$) verarbeitet werden. Diese Zahl muß ein Wert der Folge 2^n sein (512, 1024, 2048, 4096, 8192). Eine Fehleingabe führt zu einer Fehlermeldung und muß korrigiert werden.

Dieser Parameter hat für das gesamte Programm Gültigkeit und kann außer beim Neustart nicht mehr verändert werden. Eine größere Zahl führt zu einer genaueren Funktionsdarstellung, aber auch zu längeren Rechenzeiten. Hier sollte man etwas experimentieren, welche Auflösung noch zu annehmbaren Rechenzeiten führt.

Nach der Eingabe dieses Parameters versucht das Programm für die Funktionen Speicherplatz zu reservieren. Führt dies zur Fehlermeldung "Nicht genug Speicherplatz", muß ein kleinerer Wert eingegeben werden.

Wurde beim Start der Parameter /**n** angegeben, so wird die Anzahl der Abtastwerte der Datei "STD.SGL" entnommen.

Anmerkung: Eine hohe Auflösung im Zeitbereich führt nicht automatisch zu einer hohen Auflösung im Frequenzbereich. Sehen Sie hierzu auch im Kapitel "3.13.2. Die Abtastperioden im Zeit und Frequenzbereich".

3.4. Der Bildschirmaufbau

Abbildung 9 zeigt den Bildschirmaufbau des Programms.

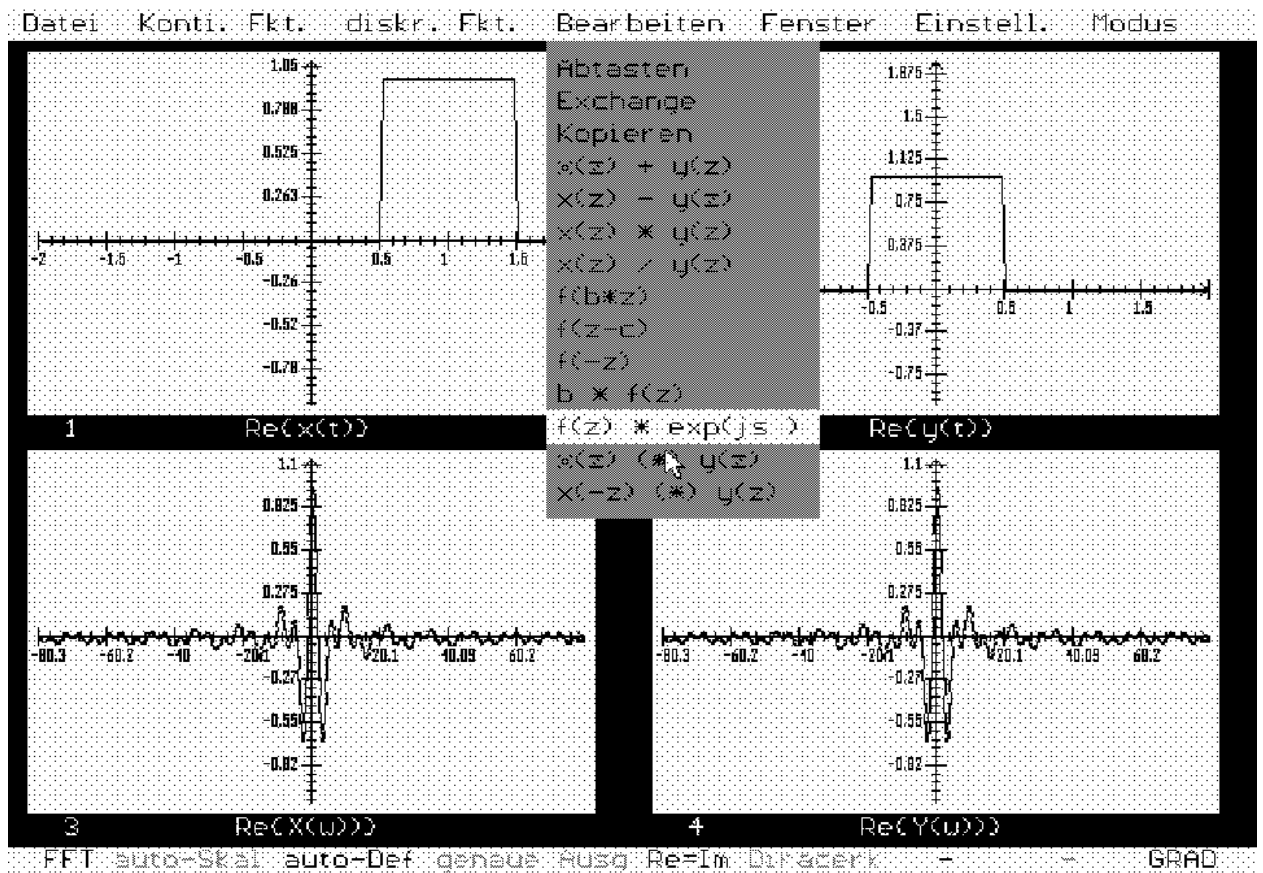


Abb. 9: Grafische Oberfläche des Programms mit geöffnetem Pull-downmenu

3.4.1. Die Fenster

Nach dem Start stellt das Programm vier Fenster dar. Jedes Fenster hat eine Nummer, die in der linken Ecke im Untertitel erscheint. Mit diesen Nummern können die Fenster angesprochen werden.

3.4.1.1. Funktionen im Fenster

Jedes Fenster stellt eine bestimmte Funktion dar. Es kann entweder der Realteil, der Imaginärteil oder Real- und Imaginärteil einer Funktion im Fenster dargestellt werden. Welche Funktion gerade dargestellt wird, sieht man an dem Untertitel jedes Fensters. Jedes Fenster kann jede beliebige Funktion darstellen. Welche Funktion dargestellt werden soll, wird mit dem Menu "FENSTER" eingestellt. So kann eine individuelle Fensterbelegung eingestellt werden. Will man z.B. nur Signaloperationen im Zeitbereich ausführen, kann man auf eine Darstellung der Funktionen $X(\omega)$ und $Y(\omega)$ verzichten.

Wird eine Funktion gerade nicht in einem Fenster dargestellt, ist sie trotzdem nicht verloren. Die Funktionsvektoren bleiben erhalten und können wieder angezeigt werden.

Die verschiedenen Funktionsarten werden in bestimmten Farben dargestellt:

Realteil bzw. Betrag einer kontinuierlichen Funktion :	BLAU
Imaginärteil bzw. Phase einer kontinuierlichen Funktion:	ROT
Realteil eines Diracimpulses:	HELLBLAU
Imaginärteil eines Diracimpulses:	HELLROT
Realteil einer diskreten Funktion:	GELB
Imaginärteil einer diskreten Funktion:	LILA

Diracimpulse werden als Pfeile und diskrete Funktionswerte als Stecknadelköpfe dargestellt.

3.4.1.2. Aktives Fenster festlegen

Das Fenster mit rotem Schatten ist das **aktive** Fenster. Ein Fenster wird aktiv durch Drücken einer der Tasten "F1", "F2", "F3", "F4" oder durch einfaches Anklicken mit der Maus.

Auf das aktive Fenster beziehen sich die in den Menüpunkten angewählten Operationen. Wird das Fenster 2 aktiviert und danach der Menüpunkt "SIN(z)" angewählt erscheint im Fenster 2 eine Sinusfunktion.

Manche Operationen z. B. SIN(z) sind nur bei Fenstern möglich, die entweder Real- oder Imaginärteil enthalten und nicht beides. Dies wird durch eine Meldung signalisiert.

Anmerkung: Das vergrößerte Fenster ist nicht unbedingt das aktive Fenster.

3.4.1.3. Fenster vergrößern

Durch Drücken einer der Zahlentasten "1", "2", "3", "4" kann das Fenster mit der entsprechenden Nummer vergrößert werden, so daß es den gesamten Bildschirm füllt. In dieser Darstellung werden mehr Werte des Funktionsvektors dargestellt, so daß Details besser sichtbar sind.

In dem Vergrößerungsmodus kann genauso mit den Menüs gearbeitet werden, wie in dem Normalmodus mit vier Fenstern.

Durch Drücken von **SPACE** wird wieder in den Normalmodus zurückgeschaltet.

In den Vergrößerungsmodus gelangt man auch durch Anklicken eines aktiven Fensters. Zurückschalten kann man durch nochmaliges Anklicken des großen Fensters.

3.4.2. Die Statuszeile und Hilfstextzeile

In der letzten Zeile werden für den Benutzer wichtige Informationen angezeigt.

3.4.2.1. Die Statusanzeige

Wenn kein Menüpunkt angewählt wurde und auch kein Fehler auftritt, wird in der letzten Zeile angezeigt, in welchem Modus das Programm gerade arbeitet.

Der Modus kann in dem Menu "MODUS" verändert werden.

- "**IFFT**" oder "**FFT**" zeigen an, in welche Richtung die Fouriertransformation zuletzt ausgeführt wurde. Mit dem Menüpunkt "3.11.1. Neuzeichnen" kann die Richtung der Fouriertransformation direkt beeinflusst werden. "IFFT" heißt inverse Fouriertransformation und "FFT" normale Fouriertransformation.
- "**auto-Skal**" oder "**manu-Skal**" zeigen an, ob das Programm die Funktionen beim Bearbeiten automatisch skalieren soll oder ob der Benutzer die Skalierung manuell für jede Funktion eingeben soll.
- "**auto-Def**" oder "**manu-Def**" zeigen an, ob das Programm den Definitionsbereich von Funktionen bestimmen soll oder ob der Defaultbereich genommen werden soll.
- "**genaue Ausg**" oder "**schn. Ausg**" zeigen an, ob für die Funktionen ein schneller aber etwas ungenauer oder eine genauer, langsamer Ausgabealgorithmus genommen werden soll.
- "**Re=Im**" oder "**Re#Im**" zeigen an, ob Real- und Imaginärteil automatisch gleich ("Re=Im") skaliert werden sollen oder ob jeder individuell ("Re#Im") berechnet werden soll.
- "**Diracerk**" zeigt an, ob bei der Fouriertransformation Diracimpulse automatisch entdeckt werden sollen oder nicht.
- "**AUFG**" zeigt an, daß gerade eine Aufgabe bearbeitet wird.
- "**Fen-Def**" zeigt an, daß der Definitionsbereich bei Operationen aus dem Menu "BEARBEITEN" vom Benutzer beeinflusst werden kann.
- "**GRAD**" oder "**RAD**" zeigen an, ob die Phase der Funktionen in Grad oder Radiant dargestellt werden sollen.

Detaillierte Erklärungen finden Sie in dem Kapitel "3.12. Das MODUS-Menü".

3.4.2.2. Die Anzeige von Fehlern und Warnungen

Im Programm wird zwischen Fehlern und Warnungen unterschieden.

- Fehler haben eine falsche oder gar keine Ausführung der angewählten Aufgabe zur Folge. Meistens beruhen sie auf einer Fehlbedienung des Programms z.B. eine falsche Eingabe oder eine Operation, die nicht ausgeführt werden darf. Bei einem Fehler erscheint in der Statuszeile eine blinkende Fehlermeldung, die durch Drücken von **CR** oder eines Mausknopfs bestätigt werden muß.
- Warnungen sind Hinweise vom Programm auf Rechenungenauigkeiten oder problematische Operationen. Eine Warnung soll nur darauf hinweisen, daß das Ergebnis einer Operation möglicherweise Abweichungen von den erwarteten Resultaten hat. Warnungen können also ignoriert werden. Bei einer Warnung erscheint eine kurze Meldung in der Statuszeile, die nach einer Sekunde wieder verschwindet.

3.4.2.3. Hilfstexte

Beim Aktivieren des Menus erscheint in der letzten Zeile jeweils ein kurzer Hilfstext zum dunkel unterlegten Menüpunkt.

3.4.3. Die Menuleiste

Die Menuleiste ist die erste Zeile auf dem Bildschirm. Mit ihrer Hilfe können Menüpunkte angewählt und ausgeführt werden.

Durch Betätigen von **ESC** wird das Menu aktiviert, d.h. durch Cursorbewegungen kann durch die Menüpunkte gefahren werden. Durch nochmaliges Drücken von **ESC** wird das Menu verlassen. Um einen Menüpunkt auszuführen drücken Sie **CR**.

Die Menuleiste kann auch direkt mit der Maus angeklickt werden, um ein Pulldownmenu zu aktivieren. Danach kann man einen Menüpunkt anklicken. Ist dieser Menüpunkt noch nicht dunkel unterlegt, wird er dunkel unterlegt und der Hilfstext erscheint. Wird dagegen ein unterlegter Menüpunkt angeklickt, führt das Programm die Operation aus. Verlassen wird das Menu, indem man irgendeine andere Stelle des Bildschirms anklickt.

Die genaue Beschreibung der Menüpunkte befindet sich im Kapitel über die MENUS.

3.4.4. Abfragefenster

Nach dem Anwählen von bestimmten Menüpunktes erscheinen Abfragefenster. Dabei gibt es zwei Arten von Fenstern.

- Logische Fenster wollen nur eine Frage beantwortet haben. In einer Zeile erscheint die Frage z.B. "Automatische Skalierung ?". Darunter erscheinen drei Boxen mit Antwortmöglichkeiten z.B. "manuell", "auto", "Abbrechen". Man kann diese Boxen entweder anklicken oder den Anfangsbuchstaben des Textes in der Box drücken. "Abbrechen" wird dabei mit **ESC** bewirkt. Dabei wird der Menüpunkt ohne Funktion verlassen. Das Betätigen von **CR** nimmt den Inhalt der ganz linken Box als Antwort.
- Eingabefenster verlangen nach konkreten Zahlen oder Texteingaben, z.B. wird im Menüpunkt LADEN der Dateiname abgefragt. Hierfür stehen folgende Editierungstasten zur Verfügung:

ESC	Fenster verlassen und Menüpunkt abbrechen
↓	Cursor in das Eingabefeld unter das aktuelle Feld bewegen
↑	Cursor in das Eingabefeld über das aktuelle Feld bewegen
←	Cursor in dem Feld nach links bewegen
→	Cursor in dem Feld nach rechts bewegen
Entf	Zeichen unter dem Cursor löschen
Del	Zeichen links vom Cursor löschen
CR	Eingabe beenden und Menüpunkt ausführen
Einfg	Umschalten zwischen Einfügemodus und Überschreibmodus

Im Überschreibmodus ist der Cursor ein Strich unter den Zeichen und im Einfügemodus besteht er aus zwei rechtwinkligen Linien.

Der Cursor kann auch in ein Eingabefeld bewegt werden, indem mit der Maus auf das entsprechende Feld geklickt wird. Zum Beenden der Eingabe oder Abbrechen können die beiden Felder "FERTIG" und "ABBRECHEN" unten im Fenster angeklickt werden.

3.5. Die Fouriertransformation

Das Kernstück des Programms ist die Fouriertransformation. Jedesmal wenn eine Funktion bearbeitet oder erstellt wurde, wird die Fouriertransformierte errechnet. Wurde z.B. die Funktion $Y(\omega)$ bearbeitet, so wird direkt die inverse Fouriertransformierte $y(t)$ ausgerechnet. In welche Richtung die zuletzt ausgeführte Fouriertransformation stattgefunden hat, geht aus der Statuszeile hervor: "FFT" bzw. "IFFT".

Da intern diskrete Funktionsvektoren verarbeitet werden, wird auch die diskrete Fouriertransformation verwendet. Die Probleme, die dadurch entstehen können, werden im Kapitel "3.13. Warum sieht nicht alles so aus wie erwartet?" angesprochen.

3.6. Das DATEI-Menu

3.6.1. LADEN

Dieser Menüpunkt dient dazu, Funktionen oder auch die ganze Arbeitsumgebung nachzuladen. Es können vier verschiedene Dateiformate geladen werden, die automatisch erkannt werden.

Nach dem Aufruf dieser Funktion fragt das Programm den Filenamen der Datei ab, die geladen werden soll. Es wird nur der Name der Datei eingegeben ohne Extension und ohne Pfad. Die Datei wird dann im aktuellen Verzeichnis gesucht. Wird die Datei nicht gefunden, erscheint die Fehlermeldung "Datei kann nicht geöffnet werden".

Wenn eine Datei gefunden wurde, stellt das Programm selbstständig fest, welches Dateiformat vorliegt und lädt die Daten nach.

1. Liegt das Datenformat für eine Arbeitsumgebung vor, wird die komplette Arbeitsumgebung mit allen Funktionen und Parametern nachgeladen. Dazu zählen auch die Modi, die in der Statuszeile angezeigt werden. Alle vorherigen Einstellungen und Funktionen gehen verloren. Dieses Datenformat ist sehr nützlich, wenn man das Programm verlassen will und später an derselben Stelle weiterarbeiten will. Zum Abspeichern einer Arbeitsumgebung sehen Sie im Kapitel "3.6.2. ALLES SPEICHERN". Wichtig ist, daß die Anzahl der Abtastwerte der neuen Umgebung nicht größer als die beim Start eingegebene Anzahl ist.
Eine Arbeitsumgebung kann nur geladen werden, wenn die Anzahl der Abtastwerte der gespeicherten Funktionen kleiner oder gleich der Anzahl Abtastwerte ist, die beim Programmstart festgelegt worden ist.
2. Liegt das Datenformat für eine einzelne Funktion vor, wird eine Funktion in das aktuelle Fenster geladen. Für dieses Datenformat ist es also wichtig das richtige Fenster angewählt zu haben. Der Inhalt der alten Funktion geht verloren. Sehen Sie auch im Kapitel "3.6.3. AKTUELLES FENSTER SPEICHERN".
Eine Funktion kann nur geladen werden, wenn die Anzahl der Abtastwerte der gespeicherten Funktion genau gleich der aktuellen Anzahl Abtastwerte ist.
3. Es kann auch ein Format geladen werden, mit dem komplexe Funktionswerte geladen werden können. Dieses Datenformat ist sehr einfach und übergibt keine Funktionsparameter (z.B. Definitionsbereich, Skalierung, usw.). Es ist deshalb geeignet einfache komplexe Funktionssample zu verarbeiten. Die Funktionsparameter werden auf die Standardwerte gesetzt.
4. Ein anderes Format lädt einfache reelle Funktionswerte nach. In diesem Format kann auch die Abtastperiode übergeben werden. Auch die letzten beiden Formate werden in das aktuelle Fenster geladen.
Bei diesem und dem vorherigen Format Nr. 3, darf die Anzahl der Abtastwerte von der aktuellen Anzahl abweichen. In diesem Fall gibt das Programm einen Fehler aus, lädt aber die Funktionswerte ordnungsgemäß und setzt alle Parameter.

Im Kapitel "3.15. Die Dateiformate" werden die Dateiformate detailliert beschrieben und auch erklärt, wie mit einem Editor Veränderungen in den Dateien vorgenommen werden können.

Findet das Programm in dem Dateiformat Fehler, erscheint eine Fehlermeldung mit der fehlerhaften Dateizeile. Dies ist mit höchster Wahrscheinlichkeit auf eigene Änderungen in der Datei zurückzuführen. Schauen Sie deshalb mit einem Editor in dieser Datei nach und korrigieren den Fehler.

3.6.2. ALLES SPEICHERN

Dieser Menüpunkt speichert alle Funktionen und Einstellungen der aktuellen Arbeitsumgebung ab. Auch die Modi, die in der Statuszeile angezeigt werden, werden gesichert. Man kann also jederzeit wieder an dieser Stelle weiterarbeiten. Auch hier wird ein Filename abgefragt und die Datei im aktuellen Verzeichnis gespeichert. Wird als Filename "STD" eingegeben, wird eine Standardumgebung gespeichert, die beim Programmstart ggf. geladen wird. Auf diese Weise kann das Programm für individuelle Bedürfnisse eingestellt werden.

Die Dateien erhalten die Extension ".SGL".

3.6.3. AKTUELLES FENSTER SPEICHERN

Hier kann die Funktion im aktuellen Fenster abgespeichert werden. Es werden alle Funktionsparameter gesichert (Skalierung, Definitionsbereich, Periodendauer, usw.). Es werden immer Real- und Imaginärteil abgespeichert, auch wenn sich im aktuellen Fenster nur ein Teil der Funktion befindet.

Auch diese Datei erhält die Extension ".SGL".

3.6.4. FUNKTION LÖSCHEN

Mit diesem Menüpunkt kann die aktuelle Funktion gelöscht werden. Es wird vorher abgefragt, ob nur der Teil im aktuellen Fenster (z.B. nur der Imaginärteil) oder die ganze Funktion gelöscht werden soll. Dieser Menüpunkt sollte vor jedem neuen Erzeugen von Funktionen aufgerufen werden.

Wird z.B. nur der Realteil einer Funktion dargestellt, so kann durch Rundungsfehler oder Rechenungenauigkeit ein Rest im Imaginärteil entstehen, der nicht bemerkt wird. Dies sollte bei unerwarteten Ergebnissen kontrolliert werden.

Diese Funktion kann auch mit der Taste **F11** aufgerufen werden.

3.6.5. INFORMATION

Dieser Menüpunkt zeigt Informationen über die aktuelle Funktion an. Es werden alle Funktionsparameter (Skalierung, ...) und die Funktionsvektoren angezeigt. Dieser Punkt war ursprünglich dazu gedacht, Fehler bei der Entwicklung zu entdecken. Es ist möglich jeden Funktionswert einzeln zu kontrollieren und so Operationen genau zu überprüfen. Für den Benutzer ist diese Punkt vielleicht nicht ganz so interessant, man gewinnt aber sehr gut einen Einblick in die interne Darstellung der Funktionen.

All diese Daten werden auch gesichert, wenn der Menüpunkt "AKTUELLES FENSTER SPEICHERN" aufgerufen wird.

Der Informationsbildschirm zeigt Informationen zum Real- und Imaginärteil der Funktion an.

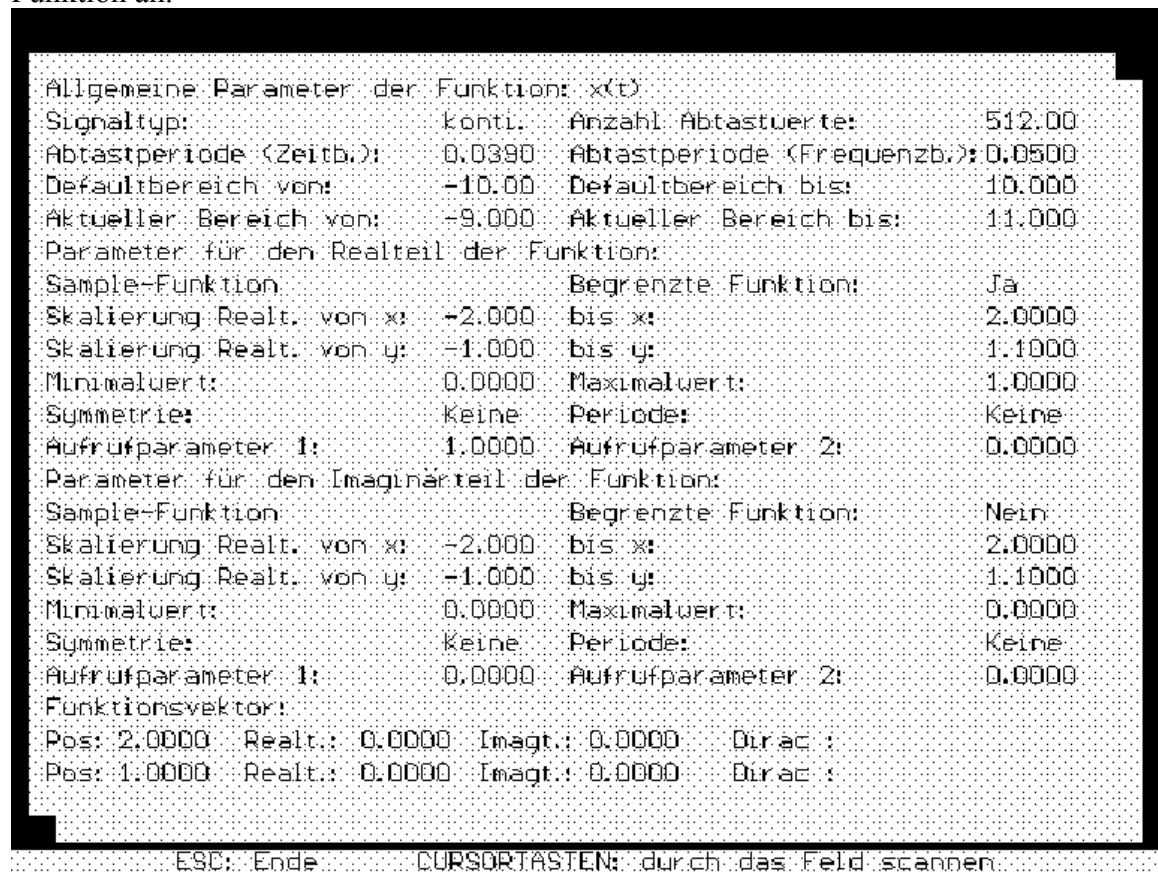


Abb. 10: Die Darstellung des Menüpunktes INFORMATION

Im ersten Abschnitt werden die allgemeinen Parameter der Funktion ausgegeben, d.h. die Daten, die für Real- und Imaginärteil gleich sind:

- Funktionsname: z.B. x(t) wird in der ersten Zeile ausgegeben.
- Signaltyp: Es gibt drei Signaltypen, die hier angegeben werden: "konti." für kontinuierliche Signale, "diskr." für diskrete Signal und "kein", wenn kein Signal vorhanden ist.
- Anzahl Abtastwerte N: Das ist der Parameter, der beim Programmstart eingegeben wurde.
- Abtastperiode (Zeitb.) T: Das ist der Abstand zwischen zwei Funktionswerten im Zeitbereich.
- Abtastperiode (Frequenzb.) Ω : Das ist der Abstand zwischen zwei Funktionswerten, der sich für die Fouriertransformierte ergibt.

Die letzten drei Parameter sind mathematisch voneinander abhängig. Es gilt:

$$\underline{\Omega \cdot T \cdot N = 2\pi} \quad (86)$$

Aus diesem Zusammenhang wird deutlich, daß eine größere Abtastgenauigkeit im Zeitbereich zu einer ungenaueren Darstellung im Frequenzbereich führt.

- Defaultbereich: Das ist der Definitionsbereich, der verwendet wird, wenn die automatische Definitionsbereichseinstellung abgeschaltet worden ist. Wie der Defaultbereich eingestellt wird, steht im Kapitel "3.12. Das MODUS-Menü".

- Aktueller Bereich: Das ist der Definitionsbereich, der für die aktuelle Funktion eingestellt worden ist. In diesem Intervall ist der Funktionsvektor abgespeichert. Aus dem Definitionsbereich errechnet sich auch die Abtastperiode:

$$T \cdot N = X_2 - X_1 \quad (87)$$

Danach werden die Parameter angezeigt, die nur für Real- oder Imaginärteil gültig sind.

- Funktionsausdruck: Dies ist die analytische, mathematische Beschreibung der Funktion z.B. $\text{rect}(t/b)$. Wird die Funktion bearbeitet, ist sie nicht mehr analytisch darstellbar und es wird an dieser Stelle "Sample-Funktion" ausgegeben, d.h. sie liegt nur noch als Funktionsvektor vor.
- Begrenzte Funktion: Dieser Parameter gibt an, ob die Funktion auf der x-Achse begrenzt ist. Die $\sin(t)$ -Funktion ist z.B. unbegrenzt und die $\text{rect}(t)$ -Funktion begrenzt. Sobald eine Funktion zur Samplefunktion wird, kann es passieren, daß dieser Parameter nicht mehr stimmt. Dies hat mit der internen Verarbeitung zu tun.
- Skalierung von x: Dies ist der Bereich auf der x-Achse, der von der Funktion dargestellt wird.
- Skalierung von y: Dies ist der Bereich auf der y-Achse, der von der Funktion dargestellt wird.
- Minimalwert: Dies ist der kleinste y-Wert, der im Funktionsvektor abgespeichert ist. Dieser Parameter ist wichtig, wenn die Funktion automatisch skaliert werden soll.
- Maximalwert: Dies ist der größte y-Wert.
- Symmetrie: Hier wird angegeben, ob die Funktion "gerade", "ungerade" oder nicht symmetrisch ist. Dieser Parameter spielt in der internen Verarbeitung keine Rolle. Wenn er also falsch sein sollte, hat er keinen Einfluß auf die korrekte Funktion des Programms.
- Periode: Dies ist die Periodendauer von periodischen Funktionen. Solange hier ein Wert ungleich Null steht, kann eine Funktion auch außerhalb ihres Definitionsbereichs bearbeitet und dargestellt werden.
- Aufrufparameter: Dies sind die Parameter, die beim Aufruf der Funktion abgefragt worden sind z.B. Breite der rect -Funktion.

Danach werden die diskreten Werte des Funktionsvektors ausgegeben.

- Pos: ist die Position in dem Vektor. Sie läuft von eins bis zur eingestellten Anzahl diskret verarbeiteter Werte.
- Realt.: Ist der Realteil der Funktion an der Stelle Pos.
- Imagt: Ist der Imaginärteil der Funktion an der Stelle Pos.
- Dirac: Hier wird angegeben, ob an der Stelle ein Diracimpuls im Real- oder Imaginärteil vorhanden ist.

Mit den Tasten " \uparrow ", " \downarrow ", "**Bild** \downarrow " und "**Bild** \uparrow " kann durch den Funktionsvektor "gefahren" werden. Mit den Tasten "**Pos1**" und "**Ende**" kann an den Anfang oder an das Ende des Funktionsvektors gesprungen werden. Verlassen wird der Informationsbildschirm mit "**ESC**".

Das Informationsfenster kann auch mit der Taste **F12** aufgerufen werden.

3.6.6. NEUSTART

Mit dieser Funktion kann man das Programm neu starten. Es erscheint wieder das Titelbild und die Eingabe der Anzahl diskreter Werte. Alle Parameter und Funktionen gehen verloren und werden neu initialisiert.

Bevor dieser Punkt ausgeführt wird, findet noch eine Sicherheitsabfrage statt.

3.6.7. ENDE

Diese Funktion verläßt das Programm, nachdem eine Sicherheitsabfrage stattgefunden hat. Wenn die aktuellen Einstellungen nicht gespeichert werden, gehen sie verloren.

3.7. Das KONTI.FKT-Menu

In diesem Pulldownmenu können kontinuierliche Funktionen abgerufen werden. Alle Funktionen werden dann intern berechnet und im Funktionsvektor der Funktion im aktuellen Fenster abgelegt. Befinden sich im Fenster gerade Real- und Imaginärteil einer Funktion kann, diese Operation nicht ausgeführt werden. Es muß eindeutig sein, ob Real- oder Imaginärteil bzw. Betrag oder Phase einer Funktion angesprochen werden sollen.

Der Funktionsparameter z z.B. bei $\sin(z)$ muß für Zeitfunktionen durch die Zeit t und für Frequenzfunktionen durch die Frequenz ω ersetzt werden.

Die Angaben, die im weiteren Text zum Definitionsbereich gemacht werden, sind für die Einstellung "Definitionsbereich automatisch" gültig. Bei manueller Bereichseinstellung wird immer der Defaultbereich genommen.

3.7.1. Die Funktion $\sin(z)$

Es wird eine Sinusfunktion erzeugt:

$$f(t) = \sin(\omega_0 t - \varphi) \quad (88)$$

Der Phasenwinkel φ und die Kreisfrequenz ω_0 werden abgefragt. Der Definitionsbereich wird so gewählt, daß genau n Perioden in das Intervall passen.

3.7.2. Die Funktion $\cos(z)$

Es wird eine Cosinusfunktion erzeugt:

$$f(t) = \cos(\omega_0 t - \varphi) \quad (89)$$

Der Phasenwinkel φ und die Kreisfrequenz ω_0 werden abgefragt. Der Definitionsbereich wird so gewählt, daß genau n Perioden in das Intervall passen.

3.7.3. Die Funktion rect(z)

Es wird eine Rechteckfunktion erzeugt:

$$f(t) = \text{rect}(t/b) \quad (90)$$

Die Breite **b** wird abgefragt. Der Definitionsbereich wird so gewählt, daß das Intervall zwanzigmal so groß wie die Breite der Rechteckfunktion ist.

3.7.4. Die Funktion delta(z)

Es wird die Deltafunktion erzeugt:

$$f(t) = a \cdot \delta(t - t_0) \quad (91)$$

Das Gewicht **a** und die Verschiebung **t₀** werden abgefragt. Der Definitionsbereich wird von **t₀-10** bis **t₀+10** eingestellt.

3.7.5. Die Funktion Deltafolge

Es wird eine Deltaimpulsfolge erzeugt:

$$f(t) = \sum_{\nu=0}^{31} a \cdot \delta(t - \nu T) \quad (92)$$

Das Gewicht **a** und der Impulsabstand **T** werden abgefragt. Der Definitionsbereich wird auch bei manueller Bereichseinstellung so gewählt, daß 32 Deltaimpulse in das Intervall passen. Eine kurze Warnmeldung weist darauf hin, daß der Defaultbereich nicht eingehalten werden kann.

3.7.6. Die Funktion exp(-z²/b)

Es wird die Gauß'sche Glockenkurve erzeugt:

$$f(t) = e^{-\frac{t^2}{b}} \quad (93)$$

Die Breite **b** wird abgefragt. Der Definitionsbereich wird so gewählt, daß er zwanzigmal so groß wie die Breite der Gaußkurve ist.

3.7.7. Die Funktion triangel(z)

Es wird eine Dreiecksfunktion erzeugt:

$$f(t) = \Lambda\left(\frac{2 \cdot t}{b}\right) \quad (94)$$

Die Breite **b** wird abgefragt. Der Definitionsbereich wird so gewählt, daß er zwanzigmal so groß wie die Breite des Dreiecks ist.

3.7.8. Die Funktion si(z)

Es wird die si-Funktion erzeugt:

$$f(t) = \text{si}\left(\frac{2\pi \cdot t}{b}\right) \quad (95)$$

Die Breite **b** wird abgefragt. Der Definitionsbereich wird so gewählt, daß er vierzigmal so groß wie die Breite der Funktion ist.

3.7.9. Die konstante Funktion

Es wird eine konstante Funktion erzeugt:

$$f(t) = c \quad (96)$$

Die Konstante **c** wird abgefragt. Als Definitionsbereich wird der Defaultbereich genommen.

3.7.10. Die Sprungfunktion

Es wird die Sprungfunktion erzeugt:

$$f(t) = a \cdot \varepsilon(t-t_0) \quad (97)$$

Die Sprunghöhe **a** und die Verschiebung **t₀** werden abgefragt. Der Definitionsbereich wird auf das Intervall von **t₀-50** bis **t₀+50** eingestellt.

3.7.11. Die Funktion sign(z)

Es wird die Signumfunktion erzeugt:

$$f(t) = a \cdot \text{sign}(t-t_0) \quad (98)$$

Das Gewicht **a** und die Verschiebung **t₀** werden abgefragt. Der Definitionsbereich wird auf das Intervall von **t₀-50** bis **t₀+50** eingestellt.

3.7.12. Die Sägezahnfolge

Es wird ein periodischer Sägezahn erzeugt:

$$s(t) = \frac{2b}{T} \cdot t \quad \text{für} \quad -\frac{T}{2} \leq t \leq \frac{T}{2} \quad (99)$$

$$f(t) = \sum_v s(t - vT) \quad (100)$$

Die Periodendauer T und das Gewicht b der Sägezahnfunktion werden abgefragt. Der Definitionsbereich wird so gewählt, daß genau n Perioden in das Intervall passen. Die Sägezahnfolge ist in erster Linie implementiert worden, um Funktionen zur Verfügung zu haben, die für die Darstellung der Phase sinnvoll sind.

3.7.13. Die Rechteckfolge

Es wird eine periodische Rechteckfunktion erzeugt:

$$f(t) = \pi \cdot \sum_v \text{rect}\left(\frac{t - vT}{b}\right) \quad (101)$$

Die Periodendauer T und die Rechteckbreite b werden abgefragt. Die Höhe des Rechtecks wird auf π gesetzt. Auch diese Funktion ist in erster Linie für die Darstellung der Phase geeignet. Der Definitionsbereich wird für die Periodendauer passend gewählt.

3.7.14. Die Rauschfunktion

Es wird ein weißes Rauschen erzeugt. Der Mittelwert u und die Streuung σ werden abgefragt. Als Definitionsbereich wird der Defaultbereich gewählt.

3.8. Das DSKR.FKT-Menu

In diesem Pulldownmenu können diskrete Signale abgerufen werden. Dieses Menu ist nicht so umfangreich wie das Menu für kontinuierliche Signale. Aus einem kontinuierlichen Signal kann aber jederzeit durch Abtastung ein diskretes Signal erzeugt werden.

Der Definitionsbereich wird bei allen diskreten Signalen auf das Intervall von $-N/2$ bis $N/2$ eingestellt, so daß die Abtastperiode $T=1$ beträgt.

3.8.1. Die Funktion sin(k)

Es wird eine diskrete Sinusfunktion erzeugt:

$$f_d(k) = \sin(\omega \cdot k \cdot T - \varphi) \quad (102)$$

Die Kreisfrequenz ω und der Phasenwinkel φ werden abgefragt.

3.8.2. Die Funktion cos(k)

Es wird eine diskrete Kosinusfunktion erzeugt:

$$f_d(k) = \cos(\omega \cdot kT - \varphi) \quad (103)$$

Die Kreisfrequenz ω und der Phasenwinkel φ werden abgefragt.

3.8.3. Die Funktion rect(k)

Es wird eine diskrete Rechteckfunktion erzeugt:

$$f_d(k) = a \cdot \text{rect}\left(\frac{kT}{b}\right) \quad (104)$$

Die Breite **b** und die Höhe **a** werden abgefragt.

3.8.4. Die Funktion delta(k)

Es wird ein diskreter Funktionswert erzeugt:

$$\begin{aligned} f_d(k) &= a \quad \text{für } k = k_0 \\ f_d(k) &= 0 \quad \text{für } k \neq k_0 \end{aligned} \quad (105)$$

Der Funktionswert **a** und die Verschiebung **k₀** werden abgefragt.

3.8.5. Die diskrete Sprungfunktion

Es wird eine diskrete Sprungfunktion erzeugt:

$$f_d(k) = a \cdot \varepsilon((k - k_0) \cdot T) \quad (106)$$

Die Basis **a** wird abgefragt.

3.8.6. Die Funktion a^{-k}

Es wird eine diskrete Exponentialfunktion erzeugt:

$$\begin{aligned} f_d(k) &= a^{-kT} \quad \text{für } k \geq 0 \\ f_d(k) &= 0 \quad \text{für } k < 0 \end{aligned} \quad (107)$$

Die Basis **a** wird abgefragt.

3.9. Das BEARBEITEN-Menu

Mit diesem Pulldownmenu können Operationen aufgerufen werden, um Funktionen zu bearbeiten. Dabei gibt es Operationen die sich nur auf eine einzelne Funktion beziehen und Operationen, die zwei Funktionen miteinander verknüpfen.

- Bezieht sich eine Operation nur auf eine Funktion z.B. f(t-c), so muß vor dem Aufruf das Fenster, das die Funktion enthält, aktiviert werden.

- Werden zwei Funktionen verknüpft, so können entweder die beiden Zeitfunktionen oder die beiden Frequenzfunktionen bearbeitet werden. Wird z.B. eine Frequenzfunktion markiert, so werden die Funktionen $X(\omega)$ und $Y(\omega)$ miteinander verknüpft.

Das Ergebnis einer solchen Operation wird immer in der Funktion $x(t)$ bzw. $X(\omega)$ abgelegt. Die Funktion $y(t)$ bzw. $Y(\omega)$ bleibt erhalten.

Bevor eine Operation, die sich auf zwei Funktionen bezieht, ausgeführt werden kann, muß das Programm die Definitionsbereiche anpassen. Die Funktionen werden so umgerechnet, daß sie im gleichen Intervall liegen. Insofern wird auch die Funktion y von der Operation beeinflusst.

Die Wahl des Definitionsbereichs kann beeinflusst werden, wenn der Menüpunkt "DEFBER. BEIM BEARBEITEN" aufgerufen wird. In der Statuszeile erscheint dann der Modus "Fen-Def". Jetzt werden beide Funktionen auf den Definitionsbereich der aktuellen Funktion angepaßt.

Will man z.B. die Funktionen $x(t)$ und $y(t)$ dividieren und aktiviert das Fenster der Funktion $y(t)$, so werden beide Funktionen auf das Intervall von $y(t)$ angepaßt.

Ebenso können manche Operationen wahlweise auf Real- oder Imaginärteil oder die ganze Funktion angewendet werden. Dazu erscheint ein Abfragefenster "Nur Teil im Fenster bearbeiten?".

- Hier kann das Feld "Fenster" angewählt werden, wenn nur der Teil im aktuellen Fenster bearbeitet werden soll. Befindet sich im aktuellen Fenster gerade der Realteil einer Funktion und wird die Multiplikation aufgerufen, so werden der Realteil von x und der Realteil von y multipliziert.
- Wählt man dagegen das Feld "ganze Funktion" an, so werden Real- und Imaginärteil bearbeitet.

Findet diese Abfrage nicht statt, so bezieht sich die Operation immer auf die ganze Funktion. In der Betrags- und Phasendarstellung können meistens nur ganze Funktionen bearbeitet werden.

3.9.1. Abtasten

Diese Operation stellt einen idealen Abtaster dar. Die aktuelle Funktion wird mit der Abtastperiode T abgetastet:

$$s_a(t) = s(t) \cdot \sum_{k=-\infty}^{\infty} \delta(t - k \cdot T) \quad (108)$$

Die abgetastete Funktion wird dann über der Variablen k dargestellt.

Die Abtastperiode T wird abgefragt. Wird die Abtastperiode kleiner als die interne Abtastperiode für die Funktionsvektoren gewählt, erscheint eine Fehlermeldung. In diesem Fall muß die Abtastperiode größer gewählt werden. Im Zweifelsfall kann man mit dem Menüpunkt INFORMATION nachschauen, wie groß die Abtastperiode der

aktuellen Funktion ist. Der Definitionsbereich wird immer auf das Intervall $\left[-\frac{N}{2}, \frac{N}{2} \right]$

(N : Anzahl Abtastwerte) eingestellt.

3.9.2. Exchange

Mit dieser Operation werden Real- und Imaginärteil der Funktionen x und y im Zeit- und Frequenzbereich vertauscht. Diese Operation wird gebraucht um die Reihenfolge von nicht assoziativen Operationen zu vertauschen. Will man z.B. $x(t)$ von $y(t)$ abziehen, so muß EXCHANGE aufgerufen werden und danach der Menüpunkt "X - Y".

3.9.3. Kopieren

Diese Operation kopiert Real- und Imaginärteil der Funktion im aktuellen Fenster in die andere Funktion. Dies ist z.B. nützlich um das Autokorrelationsprodukt einer Funktion zu berechnen, indem man die Funktion kopiert und anschließend das Korrelationsprodukt bildet.

3.9.4. Die Operation $x(z) + y(z)$

Es wird die Summe der beiden Funktionen x und y errechnet:

$$x(t) = x(t) + y(t) \quad (109)$$

Es ist möglich Diracimpulse zu einer kontinuierlichen Funktion zu addieren. Dabei werden die beiden Funktionen nicht mathematisch addiert, sondern einfach gemeinsam in einem Funktionsvektor abgespeichert.

3.9.5. Die Operation $x(z) - y(z)$

Es wird die Differenz der Funktionen x und y errechnet:

$$x(t) = x(t) - y(t) \quad (110)$$

Es ist möglich Diracimpulse von einer kontinuierlichen Funktion zu subtrahieren. Dabei werden die beiden Funktionen nicht mathematisch subtrahiert, sondern einfach gemeinsam in einem Funktionsvektor abgespeichert.

3.9.6. Die Operation $x(z) \cdot y(z)$

Die Funktionen x und y werden multipliziert:

$$x(t) = x(t) \cdot y(t) \quad (111)$$

Werden Diracimpulse mit einer kontinuierlichen Funktion multipliziert, entspricht dies einer Abtastung der kontinuierlichen Funktion an der Position der Diracimpulse. Es ist nicht möglich zwei Diracfunktionen miteinander zu multiplizieren

3.9.7. Die Operation $x(z) \div y(z)$

Es wird der Quotient von x und y gebildet:

$$x(t) = \frac{x(t)}{y(t)} \quad (112)$$

Dabei ist zu beachten, daß die Funktion y Nullstellen aufweisen kann. In diesem Fall wird die Null durch eine sehr kleine Zahl ersetzt, um einen "Division by Zero"-Laufzeitfehler zu vermeiden. Das Ergebnis ist demnach eine sehr große Zahl aber nicht unendlich.

In diesem Programm ist es nicht erlaubt durch eine Diracfunktion zu dividieren.

3.9.8. Die Operation $f(b \cdot z)$

Die aktuelle Funktion wird in x-Richtung gedehnt ($b < 1$) bzw. gestaucht ($b > 1$). Für negative b ergibt sich eine Spiegelung an der y-Achse.

$$x(t) = x(b \cdot t) \quad (113)$$

3.9.9. Die Operation $f(z - c)$

Die aktuelle Funktion wird für $t_0 > 0$ nach rechts und für $t_0 < 0$ nach links verschoben.

$$x(t) = x(t - t_0) \quad (114)$$

3.9.10. Die Operation $f(-z)$

Diese Operation ist ein Sonderfall der Dehnung in x-Richtung. Die aktuelle Funktion wird an der x-Achse gespiegelt.

3.9.11. Die Operation $b \cdot f(z)$

Die aktuelle Funktion wird in y-Richtung gedehnt ($b > 0$) bzw. gestaucht ($b < 0$).

$$x(t) = b \cdot x(t) \quad (115)$$

3.9.12. Die Operation $f(z) \cdot \exp(js)$

Die Phase der aktuellen Funktion wird um den Phasenwinkel φ gedreht.

$$x(t) = x(t) \cdot e^{j\varphi} \quad (116)$$

3.9.13. Die Operation $x(z) (*) y(z)$

Die Funktionen x und y werden miteinander gefaltet.

$$x(t) = x(t) * y(t) = \int_{-\infty}^{\infty} x(\tau) \cdot y(t - \tau) d\tau \quad (117)$$

Die Faltung ist natürlich auch für Diracimpulse definiert.

3.9.14. Die Operation $x(-z) (*) y(z)$

Es wird das Korrelationsprodukt der Funktionen x und y gebildet.

$$x(t) = x(-t) * y^*(t) = \int_{-\infty}^{\infty} x(\tau) \cdot y^*(t + \tau) d\tau \quad (118)$$

Um die Autokorrelation einer Funktion zu berechnen, kopiert man sie einmal und berechnet dann das Korrelationsprodukt.

3.10. Das FENSTER-Menu

Dieses Pulldownmenu dient dazu den einzelnen Fenstern Funktionen zuzuweisen, die dargestellt werden sollen. Es wird jeweils dem aktuellen Fenster die angewählte Funktion zugewiesen. Dabei ist es egal, ob alle Fenster die gleiche Funktion darstellen oder ob manche Funktionen gar nicht dargestellt werden. Wichtig ist, daß Funktionsvektoren unabhängig von ihrer Darstellung nie verloren gehen. Man kann einzelne Funktionen also für kurze Zeit ausblenden.

3.11. Das EINSTELL.-Menu

3.11.1. Neuzeichnen

Es werden alle Funktionen neu fouriertransformiert und alle Fenster neu ausgegeben. Die Richtung der Fouriertransformation wird abgefragt.

Mit dieser Funktion kann man z.B. eine gerade berechnete Fouriertransformierte probeweise zurücktransformieren. Dies ist normalerweise nicht sinnvoll, da bei der Fouriertransformation Rechenungenauigkeiten auftreten, die vermieden werden könnten.

NEUZEICHNEN kann man auch mit der Taste **F9**.

3.11.2. Polarkoord.

Das Programm wechselt in die Darstellung mit polaren Koordinaten, d.h. in die Darstellung nach Betrag und Phase.

Die Funktionsvektoren beinhalten die Funktionen weiterhin als Real- und Imaginärteil. Es ändert sich lediglich die Ausgabe. Das hat den großen Vorteil, daß für den Wechsel nicht alle Vektoren umgerechnet werden müssen. Dadurch werden Ungenauigkeiten vermieden, die sonst bei häufigem Hin- und Herschalten zwischen den beiden Darstellungsarten entstehen würden.

Der Nachteil ist, daß in der Darstellung mit Polarkoordinaten beschränkere Bearbeitungsmöglichkeiten bestehen. Die meisten Operationen können z.B. nicht auf Betrag oder Phase getrennt angewendet werden. Es kann nur die ganze Funktion bearbeitet werden. Die Darstellung mit kartesischen Koordinaten ist deshalb vorzuziehen.

Beim Umschalten gibt das Programm manchmal die Warnung "Nicht alle Daten können übernommen werden." aus. Dies weist darauf hin, daß nicht alle Informationen von der kartesischen Funktion auf die polare Funktion übertragen werden können. War z.B. der Realteil periodisch und der Imaginärteil nicht, so ist nicht ohne weiteres feststellbar, welche Periodendauer der Betrag hat. Diese Warnung soll also auf mögliche Darstellungsänderungen hinweisen.

Der Wechsel zu Polarkoordinaten kann auch mit der Taste **F5** bewirkt werden.

3.11.3. Kartes. Koord.

Das Programm wechselt in die Darstellung mit kartesischen Koordinaten. Sehen Sie dazu auch im Kapitel "3.11.2. Polarkoord."

Diese Funktion kann auch mit der Taste **F6** aufgerufen werden.

3.11.4. Skalierung (manuell)

Mit dieser Operation kann eine Funktion neu skaliert werden. Es erscheint ein Fenster, in dem die aktuelle Skalierung verändert werden kann. Es können beliebige Werte eingegeben werden. Wird die linke Grenze X_1 der Skalierung größer gewählt als die rechte Grenze X_2 erscheint eine Fehlermeldung.

Befinden sich im aktuellen Fenster Real- und Imaginärteil, so werden die Skalierungen von Real- und Imaginärteil neu eingestellt.

Ist im MODUS-Menü die Einstellung "Re=Im" gewählt worden, so skaliert das Programm Real- und Imaginärteil immer gleich.

Diese Funktion kann auch mit der Taste **F7** aufgerufen werden.

3.11.5. Skalierung (rechn.)

Mit diesem Menüpunkt veranlaßt man das Programm, die Funktion im aktuellen Fenster automatisch neu zu skalieren. Die Skalierung wird dann so gewählt, daß die Maximal- und Minimalwerte der Funktionen genau in das Fenster passen und daß der Definitionsbereich insgesamt dargestellt wird. Die automatische Skalierung hat nicht immer ihren Sinn. Besonders dann nicht, wenn die darzustellende Funktion klein gegenüber dem Definitionsbereich ist. In diesem Fall muß dann die manuelle Skalierung verwendet werden.

Diese Funktion kann auch mit der Taste **F8** aufgerufen werden.

3.11.6. Definitionsbereich

Mit diesem Menüpunkt kann der Definitionsbereich eingestellt werden. Es erscheint ein Fenster, indem die aktuellen Werte neu eingegeben werden können.

Der Definitionsbereich hat ganz entscheidenden Einfluß auf die Fouriertransformation und die Abtastperiode der Funktion. Deshalb sollte mit dieser Funktion vorsichtig umgegangen werden.

Man muß sich auch Gedanken über die Grenzen einer Änderung des Definitionsbereichs machen. Wird eine Funktion z.B. im Intervall $[-10, 10]$ dargestellt, so ist ein Definitionsbereich, der außerhalb dieses Intervalls liegt wenig sinnvoll. Eine Ausnahme besteht, wenn die Funktion periodisch ist. In diesem Fall kann der Definitionsbereich umgerechnet werden.

Ebenso ist es nicht sinnvoll für oben genanntes Beispiel einen Definitionsintervall von $[-100, 100]$ einzustellen. In diesem Fall wäre die Darstellung der Funktion so ungenau, daß man sie nicht mehr erkennen könnte. Eine Warnmeldung weist auf diesem Umstand hin.

Eine Ausnahme bilden Funktionen, die analytisch darstellbar sind. Diese Funktionen werden für andere Definitionsbereiche neu errechnet und verlieren deshalb nicht an Darstellungsgenauigkeit.

3.11.7. Abtastperiode

Die Abtastperiode ist mit dem Definitionsbereich über Gleichung (87) verknüpft. Es gilt:

$$T = \frac{x_2 - x_1}{N}$$

Wenn also die Abtastperiode verändert wird, wird indirekt der Definitionsbereich verändert. Das Definitionsintervall wird symmetrisch vergrößert bzw. verkleinert.

Die Abtastperiode bestimmt, wie genau die Darstellung der Fouriertransformierten wird. Ihre Eingabe ist anschaulicher als die Angabe des Definitionsbereichs.

3.12. Das MODUS-Menu

Mit diesem Menu werden die Modi beeinflusst, die in der Statuszeile angezeigt werden. Ein Modus wird wie mit einem Schalter entweder aktiviert oder ausgeschaltet.

3.12.1. Aufgabe starten

Das Programm bietet die Möglichkeit, kleine Aufgabenstellungen zu laden. Diese sollen Probleme der Nachrichtentechnik veranschaulichen und dazu motivieren sich selbst Lösungen zu erarbeiten.

Nach dem Aufrufen dieses Punktes wird nach dem Namen der Aufgabe gefragt. Dies ist der Name einer *.TUT-Datei, die Informationen zur Aufgabe enthält. Implementiert sind bereits folgende Dateien:

- 1.TUT Bearbeitung einer rect-Funktion
- 2.TUT Der ideale Modulator
- 3.TUT Das Energiedichtespektrum
- 4.TUT Matched Filter
- 5.TUT LZI-System

Sobald ein gültiger Aufgabenname eingegeben wurde, wird die Aufgabe geladen. Dabei wird auch eine Arbeitsumgebung geladen. Bei Ladefehlern sind also ähnliche Ursachen wie unter dem Punkt "3.6.1. LADEN" erläutert anzunehmen.

Nach dem Laden erscheint eine neue Arbeitsumgebung und ein Fenster mit der Aufgabenstellung. Nach dem Durchlesen kann durch einen Tastendruck die Bearbeitung der Aufgabe beginnen. In der Statuszeile erscheint jetzt die Meldung "Aufg".

Die Aufgabenstellungen sind so formuliert, daß eine geforderte Funktion erarbeitet werden soll. Es dürfen grundsätzlich alle Operationen und Funktionsaufrufe benutzt werden.

Nachdem eine Funktion erstellt worden ist, kann der Punkt AUFGABE BEENDEN aufgerufen werden.

3.12.2. Aufgabe beenden

Wie oben bereits bemerkt wird mit diesem Punkt die Bearbeitung einer Aufgabe beendet. Vor dem Aufruf muß das Fenster der Lösungsfunktion aktiviert werden. Das Programm stellt jetzt drei Fenster dar. Ein Fenster enthält die Funktion, die erarbeitet worden ist, ein Fenster die Funktion, die gefordert wurde und ein Fenster die Differenz der beiden Funktionen. Auf diese Weise kann anschaulich kontrolliert werden, ob die Lösungsfunktion richtig erstellt worden ist.

Nachdem eine Taste betätigt worden ist, kehrt das Programm in seinen normalen Modus zurück und die Meldung "Aufg" in der Statuszeile verschwindet.

3.12.3. Autoskalierung (an/aus)

Mit diesem Punkt kann die automatische Skalierung an- bzw ausgeschaltet werden. Dies wird in der Statuszeile mit der Meldung "auto-Skal" bzw "manu-Skal" signalisiert. Nachdem eine neue Funktion erstellt worden ist oder eine Operation ausgeführt worden ist, führt das Programm standardmäßig eine automatische Skalierung der Ergebnisfunktion durch. Dies ist besonders beim Abrufen von neuen Funktionen sinnvoll, da eine optimale Skalierung eingestellt wird. Auf der anderen Seite ist es aber schwierig Funktionen zu vergleichen und Änderungen an der Funktion festzustellen, wenn immer neu skaliert wird. Für diesen Fall kann die automatische Skalierung abgeschaltet werden. Das Programm behält dann immer die eingestellte Skalierung bei.

3.12.4. Autodefinition (an/aus)

Mit diesem Punkt kann die automatische Einstellung des Definitionsintervalls an- bzw ausgeschaltet werden. In der Statuszeile erscheint dann "auto-Def" bzw "manu-Def". Standardmäßig wird der Definitionsbereich einer Funktion automatisch vom Programm eingestellt. Da die Wahl des Definitionsbereichs großen Einfluß auf das Ergebnis der Fouriertransformation hat, ist es meistens sinnvoll die automatische Einstellung zu belassen. Mehr zu diesem Thema finden Sie im Kapitel "3.13. Warum sieht nicht alles so aus wie erwartet?".

Wird die manuelle Einstellung gewünscht, stellt das Programm den Definitionsbereich immer auf den Defaultbereich ein. Dieser ist standardmäßig auf das Intervall $[-10, 10]$ eingestellt.

Nachdem ein Modus ausgewählt worden ist, fragt das Programm den Defaultbereich ab. Wird ein neuer Bereich eingegeben, so bezieht er sich auf alle Funktionen, die ab jetzt erstellt werden. Ist eine Funktion bereits erstellt worden, so hat der Defaultbereich keinen Einfluß.

Bei automatischer Definitionsbereichseinstellung dient der Defaultbereich dem Programm als Orientierung, in welcher Größenordnung das Intervall eingestellt werden soll.

3.12.5. Ausgabegeschwindigkeit

Mit diesem Punkt kann eingestellt werden, mit welchem Algorithmus die Funktionen ausgegeben werden sollen. Dies wird in der Statuszeile mit "genaue Ausg" bzw "schn. Ausg" signalisiert.

In diesem Programm sind zwei Ausgabealgorithmen implementiert:

- Die schnelle Ausgabe gibt nur so viele Funktionswerte aus, wie im Fenster nebeneinander passen. In das kleine Fenster passen z.B. 200 Werte. Also werden von beispielsweise 1024 abgespeicherten Werten nur ca. 20% ausgegeben. Dabei kann es passieren, daß besonders markante Werte (Maximum oder Minimum) überlesen werden und die Ausgabe ungenau wird.
- Die genaue Ausgabe behebt diesen Fehler. Es werden alle Werte, die im Funktionsvektor gespeichert sind, dargestellt. Deshalb ist die Kurve des Graphen auch etwas dicker. Aus diesem Verfahren ergibt sich aber auch, daß die Ausgabe mitunter sehr lange dauern kann.

Hier kann der Benutzer je nach Anwendung eine Darstellungsart ausprobieren.

3.12.6. Diracerkennung

In diesem Programm wird die Fast Fouriertransformation verwendet. Tritt bei der Transformation z.B. der Sinusfunktion ein Dirac im Ergebnis auf, so muß dieser erkannt und als Pfeil dargestellt werden. Dies ist für komplexere Funktionen gar nicht so einfach, da entweder "normale" Funktionswerte (z.B. Pole) als Dirac erkannt werden oder Diracimpulse im Gewirr der Funktionswerte "übersehen" werden.

Die automatische Diracerkennung ist grundsätzlich sinnvoll und funktioniert normalerweise auch richtig. Sie ist besonders deshalb wichtig, weil Diracimpulse, wenn sie nicht erkannt werden, zu groß dargestellt werden. Besteht die Fouriertransformierte dagegen nur noch aus Diracstößen oder treten andere fehlerhafte Phänomene auf, sollte die automatische Erkennung ausgeschaltet werden und mit dem Punkt NEUZEICHNEN eine erneute Transformation ausgeführt werden.

3.12.7. Gleiche Skal. RE und IM

Oft ist es sinnvoll eine gleiche Skalierung für den Real- und Imaginärteil einer Funktion zu wählen, wenn sie in verschiedenen Fenstern dargestellt werden. Die beiden Teilfunktionen können dann besser verglichen werden und die Gesamtfunktion wird überschaubarer.

Mit diesem Menüpunkt kann der Modus für gleiche Skalierung von Real- und Imaginärteil aktiviert werden. In der Statuszeile wechselt die Meldung von "Re#Im" nach "Re=Im". Das Programm nimmt für die Darstellung jetzt immer den maximalen Skalierungsbereich der beiden Funktionen.

Die gleiche Skalierung von Real- und Imaginärteil kann auch Nachteile haben. Hat der Imaginärteil z.B. eine recht große Skalierung und wird für den Realteil eine Funktion mit kleinen Funktionswerten gewählt, so kann die Darstellung unter Umständen sehr unanschaulich sein.

3.12.8. Defber. beim Bearbeiten

Beim Verknüpfen von zwei Funktionen wird der Definitionsbereich der Ergebnisfunktion normalerweise automatisch eingestellt. Das Programm wählt meistens das maximale Intervall der beiden Funktionen. Dies ist in den meisten Fällen auch sinnvoll. Will man diese automatische Veränderung des Definitionsbereich verhindern, so kann die Funktion "DEFBER. BEIM BEARBEITEN" aktiviert werden. In der Statuszeile erscheint die Meldung "Fen-Def".

In diesem Modus wird immer der Definitionsbereich der Funktion übernommen, die vor dem Aufrufen eines Bearbeitungspunktes aktiviert worden ist. Will man z.B. $X(\omega)$ und $Y(\omega)$ addieren und den Definitionsbereich von $X(\omega)$ übernehmen, so muß vor dem Aufruf der Addition das Fenster der Funktion $X(\omega)$ aktiviert werden.

3.12.9. Winkeldarstellung

Mit diesem Menüpunkt kann eingestellt werden, ob die Phase der Funktionen in Grad oder Radiant dargestellt werden soll. Bei der Darstellung in Grad wird intern weiterhin im Bogenmaß gerechnet.

Die Darstellung in Grad liegt im Intervall $[-180^\circ, 180^\circ]$. Die Darstellung in Radiant liegt im Intervall $[-\pi, \pi]$. In der Statuszeile erscheint entweder die Meldung "GRAD" oder "RAD".

3.13. Warum sieht nicht alles so aus wie erwartet?

In diesem Kapitel soll kurz erklärt werden, welche Probleme beim Arbeiten mit dem Programm auftreten können und welche Ursachen dafür verantwortlich sind. Dabei spielen besonders die Eigenschaften der Diskreten Fouriertransformation (DFT) eine große Rolle. An dieser Stelle sei dazu nur bemerkt, daß die DFT von einer diskreten und periodischen Zeitfunktion ausgeht. Die Fouriertransformierte einer periodischen Funktion ist diskret und die Transformierte einer diskreten Funktion ist periodisch. Aus diesem Grund ergibt sich für das Spektrum im Frequenzbereich wieder eine diskrete und periodische Funktion.

Die Aufgabe des Programms liegt nun darin dieses Phänomen so gut wie möglich zu verbergen, so daß die DFT für den Benutzer wie eine "normale" Fouriertransformation aussieht. Dies ist nicht immer möglich. Im folgenden werden deshalb einige grundlegende Ursachen erklärt, die für scheinbar falsche Ergebnisse verantwortlich sind.

3.13.1. Funktionen werden periodisch fortgesetzt

Eine Funktion, die in einem Funktionsvektor abgespeichert worden ist, wird von der DFT periodisch fortgesetzt. Das soll das folgende Bild veranschaulichen:

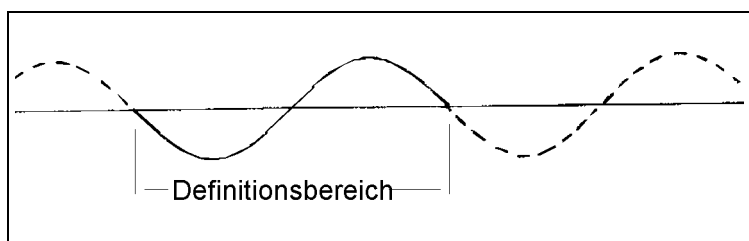


Abb. 11: Darstellung der Sinusfunktion, wenn der Definitionsbereich einer Periodendauer entspricht.

Abb. 11 zeigt eine Sinusfunktion, die genau mit einer Periode in das Definitionsintervall paßt. Wird diese Funktion jetzt periodisch fortgesetzt, so entsteht die "richtige" Sinusfunktion. Wird diese Funktion transformiert, entstehen zwei Diracstöße im Frequenzbereich. Abb. 12 dagegen zeigt eine Sinusfunktion, die nicht genau mit einer festen Anzahl von Perioden in den Definitionsbereich paßt:

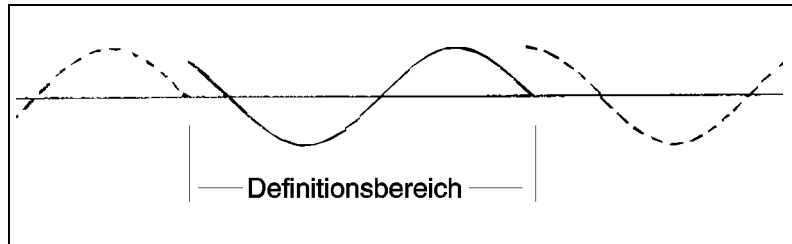


Abb. 12: Darstellung der Sinusfunktion, wenn der Definitionsbereich ungünstig gewählt worden ist.

Wird diese Funktion periodisch fortgesetzt, entsteht eine der Sinusfunktion ziemlich unähnliche Funktion. Die Fouriertransformierte wird deshalb nicht das erwartete Ergebnis mit zwei Diracstößen liefern.

Normalerweise wählt das Programm für Funktionen einen geeigneten Definitionsbereich, so daß immer ganze Perioden in ein Definitionsintervall passen. Dies ist aber nicht immer möglich. Der Definitionsbereich von Real- und Imaginärteil einer Funktion muß immer gleich sein. Von daher erklärt sich, daß das Definitionsintervall z.B. des Imaginärteils nicht ohne weiteres geändert werden kann, wenn im Realteil schon eine Funktion existiert.

Bei Änderungen des Definitionsintervalls sollte man also immer die periodische Fortsetzung der Funktion im Kopf haben. Außerdem sollte man sich über die Grenzen der Fouriertransformation bewußt sein.

3.13.2. Die Abtastperioden im Zeit und Frequenzbereich

Die Abtastperioden, d.h. die Abstände zwischen zwei diskreten Funktionswerten im Funktionsvektor, im Zeit- und Frequenzbereich stehen im direkten Zusammenhang:

$$\Omega \cdot T \cdot N = 2\pi$$

T: Abtastperiode im Zeitbereich

Ω : Abtastperiode im Frequenzbereich

N: Anzahl der Abtastwerte

Aus diesem Zusammenhang folgt, daß eine genaue Abtastung im Zeitbereich zu einer ungenauen Abtastung im Frequenzbereich führt. Wählt man z.B. N=512 Abtastwerte und für die Abtastperiode im Zeitbereich T=0.001, so folgt für die Abtastperiode im Frequenzbereich ein Wert von $\Omega \approx 12$. Die Darstellung im Frequenzbereich ist demnach höchst unanschaulich. Abhilfe kann hier eine größere Anzahl der Abtastwerte bringen. Dafür muß das Programm neu gestartet werden und ein neuer Wert eingegeben werden. Es kann aber auch direkt die Abtastperiode T vergrößert werden. Dazu muß der Menüpunkt "ABTASTPERIODE" aufgerufen werden.

Das Programm stellt für nicht-periodische Funktionen automatisch einen relativ großen Definitionsbereich ein, z.B. für die Rechteckfunktion einen Bereich, der zwanzigmal so groß wie die Breite des Rechtecks ist. Dadurch wird erreicht, daß die Abtastperiode für die Rechteckfunktion genau so groß ist, daß eine anschauliche Darstellung der Fouriertransformierten möglich ist. Nachteil eines großen Definitionsbereichs ist natürlich die ungenaue Darstellung des Rechtecks im Zeitbereich. Hier ist immer wieder gefordert gute Kompromisse zu machen.

3.13.3. Probleme beim Umrechnen in einen anderen Definitionsbereich

Ein anderes Problem besteht darin, wenn Funktionen in ein anderes Definitionsintervall umgerechnet werden müssen. Dies ist z.B. dann der Fall, wenn eine Addition ausgeführt werden soll. Es können immer nur zwei Funktionen addiert werden, die auch im gleichen Intervall liegen. Ein anderer Fall wäre natürlich, wenn der Definitionsbereich oder die Abtastperiode manuell geändert werden würden.

Wenn eine Funktion nicht mehr analytisch darstellbar ist, gehen meistens Funktionswerte verloren. Wird z.B. eine Funktion, die im Intervall $[-20, -10]$ definiert ist, zu einer Funktion im Intervall $[30, 40]$ addiert, so werden beide Funktionen auf das Intervall $[-20, 40]$ angepaßt. Die Abtastperiode wird viel größer. Es gehen also einige Funktionswerte verloren und die Darstellung wird ungenauer.

Eine ungenaue Abtastperiode wird an einer Rechteckfunktion besonders gut deutlich:

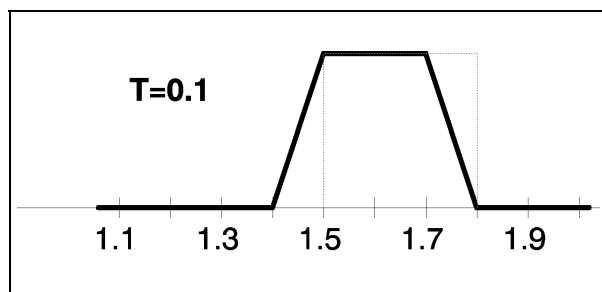


Abb. 13: Darstellung einer Rechteckfunktion bei zu großer Abtastperiode T

Die gestrichelte Linie ist die Rechteckfunktion, wie sie vom Benutzer definiert worden ist. Die dicke Linie zeigt, wie die Rechteckfunktion bei der Darstellung verzerrt werden kann. Das Beispiel zeigt also, daß eine Funktion zwar richtig im Funktionsvektor abgespeichert sein kann, aber die Darstellung als Graph ungenau wird

3.14. Fehlermeldungen

Im folgenden werden alle Fehlermeldungen aufgelistet, die im Programm verwendet worden sind:

1. Bearbeitung in doppelt belegtem Fenster nicht möglich.

Manche Operationen können nicht ausgeführt werden, wenn im aktuellen Fenster Real- und Imaginärteil dargestellt werden. In diesem Fall muß die Funktion im Fenster eindeutig sein.

2. Nicht genug Speicherplatz vorhanden !!!!!

Wenn die Anzahl der Funktionswerte in einem Vektor so groß gewählt worden ist, daß der Speicher nicht ausreicht, erscheint diese Meldung. Diese Meldung kann auch bei der Ausgabe von Fenstern und Pulldownmenüs erscheinen, da der überschriebene Bildschirmbereich im Speicher abgelegt wird. So kann es passieren, daß für diese Operation nicht genug Speicherplatz vorhanden ist.

3. Zahl muß ein Wert der 2er-Exponenten-Reihe sein.

Die Anzahl der Funktionswerte muß eine Zahl der Folge 2^n (512, 1024, 2048, 4096) sein.

4. Skalierung für das Koordinatensystem falsch!

Wurden die Grenzen des Skalierungsintervalls falsch gewählt, erscheint diese Meldung.

5. Linker Wert muß kleiner als der rechte sein.

Diese Meldung zeigt an, daß z.B. die linke Grenze des Definitionsintervalls größer als die rechte Grenze gewählt worden ist.

6. Fehler im Dateinamen.

Der Dateiname wurde im falschen Format angegeben. Er muß ohne Pfad und Extension eingegeben werden.

7. Operation bei diskreten Funktionen nicht möglich.

Manche Operationen sind mit diskreten Funktionen nicht möglich. Es kann z.B. keine kontinuierliche Funktion im Realteil eingegeben werden, wenn sich im Imaginärteil eine diskrete Funktion befindet.

8. Es wurde eine falsche '0' eingegeben!

Manche Parameter z.B. die Breite eines Rechtecks dürfen nur Werte ungleich Null annehmen.

9. Werte kleiner als '0' sind nicht erlaubt.

Es wurde ein Parameter eingegeben, der nur für positive Zahlen definiert ist. Die Abtastperiode im Menüpunkt ABTASTEN kann z.B. nicht negativ sein.

10. Division mit Diracimpulsen nicht möglich.

Enthält beim Aufrufen der Division eine der beiden Funktionen einen Diracimpuls, erscheint diese Fehlermeldung. Die Division durch einen Dirac ist nicht definiert.

11. Multiplikation mit zwei Diracimpulsen nicht möglich.

Die Multiplikation einer kontinuierlichen Funktion mit einem Diracimpuls ist möglich. Wird aber versucht zwei Diracimpulse miteinander zu multiplizieren, erscheint die Fehlermeldung. In der Praxis wird diese Operation zwar oft mit der Begründung der Ausblendeigenschaft der Diracfunktion ausgeführt. Sie ist mathematisch aber nicht definiert.

12. Es kann nicht nur der Betrag gelöscht werden.

Diese Meldung soll darauf hinweisen, daß es nicht möglich ist, nur die Betragsfunktion zu löschen. Wird der Betrag gelöscht, so ist die ganze Funktion gelöscht.

13. Abstand muß größer als 0 sein.

Der Abstand zwischen zwei Diracimpulsen in der Diracfolge muß positiv sein.

14. Abtastperiode ist kleiner als die interne Auflösung.

Will man eine Funktion im Menüpunkt "ABTASTEN" genauer abtasten als sie überhaupt im Funktionsvektor vorliegt, erscheint diese Meldung. Die Abtastperiode muß größer gewählt werden.

15. Datei kann nicht geöffnet werden.

Die Datei existiert entweder gar nicht oder ist fehlerhaft und kann deshalb nicht geöffnet werden.

16. Fehler beim Schließen der Datei.

Beim Schließen einer Datei trat ein Fehler auf.

17. Fehler im Datenformat der Datei in Zeile:

Alle Dateien, die geladen werden sollen, müssen ein spezielles Datenformat haben. Treten Abweichungen auf, erscheint diese Meldung. Mit einem Editor kann in der Dateizeile der Fehler korrigiert werden.

18. Es wurde eine falsche Zahl eingegeben!

Diese Meldung weist allgemein darauf hin, daß ein falscher Parameter eingegeben worden ist.

19. Zu viele oder zu wenig Abtastwerte.

Will man eine neue Arbeitsumgebung laden, so darf die neue Arbeitsumgebung höchstens so viele Abtastwerte haben, wie beim Programmstart eingegeben wurde. Für größere Funktionen wurde nicht genug Speicherplatz reserviert. Abhilfe kann hier ein Neustart und die Eingabe einer größeren Anzahl an Abtastwerten schaffen. Der Fehler kann auch beim Laden einer einzelnen Funktion auftreten. Hier muß die Anzahl der Abtastwerte genau mit der aktuellen übereinstimmen.

20. Unerwartetes Dateiende erreicht. Zeile:

Diese Fehlermeldung weist auf einen Fehler im Datenformat hin. Das Dateiende ist erreicht, obwohl das Programm noch Daten erwartet.

21. Funktionen sind nicht vom gleichen Typ.

Es können keine diskreten und kontinuierlichen Funktionen miteinander verknüpft werden.

22. Fehler bei Fouriertransformation.

Bei der Fouriertransformation ist ein Fehler aufgetreten.

23. Bearbeitung nur für die ganze Funktion möglich.

Manche Operationen können nur auf die ganze Funktion angewendet werden und nicht auf Real- oder Imaginärteil bzw. Betrag und Phase getrennt.

24. Es läuft schon eine Aufgabe.

Man kann erst eine neue Aufgabe starten, wenn die alte beendet worden ist.

25. Keine Aufgabe angewählt

Man kann nur eine Aufgabe beenden, wenn auch eine gestartet worden ist.

Im folgenden werden alle Warnmeldungen aufgelistet, die im Programm verwendet worden sind:

1. Achtung, es wird der Betrag gebildet!

Bei der Darstellung in polaren Koordinaten, kann die Betragsfunktion nur positive Werte darstellen. Will man ihr eine Funktion mit negativen und positiven Werten (z.B. $\sin(z)$) zuweisen, so bildet das Programm automatisch den Betrag der Funktion.

2. Nicht alle Daten können übernommen werden.

Diese Warnung weist auf den Umstand hin, daß beim Wechsel von der Darstellung mit kartesischen Koordinaten in die Darstellung mit polaren Koordinaten u.U. Informationen verloren gehen. Die Funktionen sind z.B. nicht mehr analytisch darstellbar.

3. Der Defaultbereich wird nicht eingehalten.

Das Programm orientiert sich bei der automatischen Einstellung des Definitionsbereich am Defaultbereich. Ergeben sich dabei große Abweichungen, erscheint diese Warnung.

3.15. Die Dateiformate

Im folgenden werden die Dateiformate beschrieben, die von diesem Programm verarbeitet werden können. Dateien, die Funktionsdaten enthalten, haben die Extension ".SGL". Die Aufgabendatei hat die Endung ".TUT".

In den Dateiformaten werden Schlüsselwörter verwendet, die vom Programm interpretiert werden:

U:	Kennzeichen für ein Datenformat Arbeitsumgebung
V:	Kennzeichen für ein Datenformat einer einzelnen Funktion
E:	Kennzeichen für ein Datenformat für einfache, komplexe Funktionswerte
S:	Kennzeichen für ein Datenformat für einfache Funktionswerte
NEIN:	Schlüsselwort für einen abgeschalteten Modus
JA:	Schlüsselwort für einen aktivierten Modus
KONTI:	Signaltyp kontinuierlich
DISKR:	Signaltyp diskret
KEIN:	Signaltyp keine Funktion vorhanden
GERADE:	Gerade Symmetrie
UNGERADE:	Ungerade Symmetrie
KEINE:	nicht symmetrisch
RE:	Dirac im Realteil
IM:	Dirac im Imaginärteil

Alle Zeichen vom Zeilenanfang bis zum nächsten Doppelpunkt werden als Kommentar aufgefaßt. Das Programm schreibt zu jedem Parameter automatisch einen kurzen Kommentar, so daß der Anwender relativ leicht Änderungen an einer Datei vornehmen kann. Eine Zeile, die keinen Doppelpunkt enthält, wird komplett als Kommentar aufgefaßt.

Alle Datenformate können beliebig geändert werden. Wichtig ist nur, daß das Programm die Schlüsselwörter richtig erkennt.

3.15.1. Das Format für eine Funktion

Das folgende Bild zeigt einen Ausschnitt aus einer Datei für eine einzelne Funktion:

```
Kennzeichen : V
Betrag: NEIN
Aufl: 512
Def.-ber. von: -45.000000
Def.-ber. bis: 55.000000
Signaltyp: KONTI
Nummer Fkt. (Real) : -1
Nummer 2. Fkt. (Real) : -1
Parameter 1 (Real) : 10.000000
Parameter 2 (Real) : 0.000000
Maximalwert (Real) : 1.000000
Minimalwert (Real) : 0.000000
Skalierung x1 (Real) : -5.000000
Skalierung x2 (Real) : 15.000000
Skalierung y1 (Real) : -1.000000
Skalierung y2 (Real) : 1.100000
Symmetrie (Real) : KEINE
Symmetrie 2 (Real) : KEINE
Periode (Real) : 0.000000
Periode 2 (Real) : 0.000000
Begrenzt (Real) : JA
Nummer Fkt. (Imag) : -1
Nummer 2. Fkt. (Imag) : -1
Parameter 1 (Imag) : 0.000000
Parameter 2 (Imag) : 0.000000
Maximalwert (Imag) : 0.000000
Minimalwert (Imag) : 0.000000
Skalierung x1 (Imag) : -5.000000
Skalierung x2 (Imag) : 15.000000
Skalierung y1 (Imag) : -1.000000
Skalierung y2 (Imag) : 1.100000
Symmetrie (Imag) : KEINE
Symmetrie 2 (Imag) : KEINE
Periode (Imag) : 20.000000
Periode 2 (Imag) : 0.000000
Begrenzt (Imag) : NEIN
```

```
0: 0.000000, 0.000000, ,
1: 0.000000, 0.000000, ,
2: 0.000000, 0.000000, ,
3: 0.000000, 0.000000, ,
4: 0.000000, 0.000000, ,
5: 0.000000, 0.000000, ,
6: 0.000000, 0.000000, ,
7: 0.000000, 0.000000, ,
8: 0.000000, 0.000000, ,
9: 0.000000, 0.000000, ,
10: 0.000000, 0.000000, ,
```

Abb. 14: Das Dateiformat für eine Funktion

Das Format für eine Funktion fängt mit dem entsprechenden Kennzeichen für das Datenformat an. Darauf folgen die Anzahl der Funktionswerte, die abgespeichert worden sind. Der nächste Eintrag gibt an, ob die Funktion in polaren oder kartesischen Koordinaten gespeichert worden ist. Diese Information ist wichtig, da viele Parameter für beide Darstellungsformen angegeben werden. Das Programm braucht diese Information, um die beiden Parameter richtig zuzuordnen zu können. Die Periode wird z.B. zweimal abgespeichert, da der Betrag einer Funktion nicht automatisch die gleiche Periodendauer hat wie der Realteil.

Danach folgen die Parameter der Funktion. Diese werden im Kapitel "3.6.5. INFORMATION" ausführlich beschrieben.

Schließlich folgt zeilenweise der Funktionsvektor. Jede Zeile ist im Kommentar durchnummeriert. Danach folgen Real- und Imaginärteil durch Komma getrennt. Im Anschluß kann noch ein Eintrag folgen, der Auskunft gibt, ob sich an dieser Stelle ein Diracimpuls im Real- oder Imaginärteil befindet ("RE" oder "IM").

3.15.2. Das Format für eine Arbeitsumgebung

Das folgende Bild zeigt einen Ausschnitt aus einer Datei für eine Arbeitsumgebung:

```
Kennzeichen : U
Auf: 512
Default.-ber. von: -10.000000
Default.-ber. bis: 10.000000
Betrag: NEIN
Autoskale: JA
Fourier: JA
Ausgabe schnell: NEIN
Diracerk.: JA
Autodef.: JA
Def-bereich b. Bearbeiten: NEIN
GRAD als Winkeldarst.: JA
Fenster 1: 0
Fenster 2: 3
Fenster 3: 6
Fenster 4: 7
```

Funktion : $x(t)$
 Def.-ber. von: -9.000000
 Def.-ber. bis: 11.000000
 Signaltyp: KONTI
 Nummer Fkt. (Real) : -1
 Nummer 2. Fkt. (Real) : -1
 Parameter 1 (Real) : 1.000000
 Parameter 2 (Real) : 0.000000
 Maximalwert (Real) : 1.000000
 Minimalwert (Real) : 0.000000
 Skalierung x1 (Real) : -2.000000
 Skalierung x2 (Real) : 2.000000
 Skalierung y1 (Real) : -1.000000
 Skalierung y2 (Real) : 1.100000
 Symmetrie (Real) : KEINE
 Symmetrie 2 (Real) : KEINE
 Periode (Real) : 0.000000
 Periode 2 (Real) : 0.000000
 Begrenzt (Real) : JA
 Nummer Fkt. (Imag) : -1
 Nummer 2. Fkt. (Imag) : -1
 Parameter 1 (Imag) : 0.000000
 Parameter 2 (Imag) : 0.000000
 Maximalwert (Imag) : 0.000000
 Minimalwert (Imag) : 0.000000
 Skalierung x1 (Imag) : -2.000000
 Skalierung x2 (Imag) : 2.000000
 Skalierung y1 (Imag) : -1.000000
 Skalierung y2 (Imag) : 1.100000
 Symmetrie (Imag) : KEINE
 Symmetrie 2 (Imag) : KEINE
 Periode (Imag) : 0.000000
 Periode 2 (Imag) : 0.000000
 Begrenzt (Imag) : NEIN
 0: 0.012069, -0.037151, ,
 1: -0.000001, -0.038605, ,
 2: -0.011509, -0.035418, RE,
 3: -0.020575, -0.028320, ,
 4: -0.025846, -0.018777, ,
 5: -0.026761, -0.008695, , IM
 6: -0.023669, -0.000000, ,
 7: -0.017729, 0.005761, ,
 8: -0.010659, 0.007745, ,
 9: -0.004348, 0.005984, RE, IM
 10: -0.000000, 0.000000, ,

```
11: 0.000000, -0.000000, ,  
12: -0.003228, -0.009936, ,  
13: -0.009464, -0.013025, ,  
14: -0.017297, -0.012567, ,  
15: -0.024885, -0.008086, ,  
16: -0.030342, 0.000001, ,  
17: -0.032156, 0.010449, ,
```

Abb. 15: Das Dateiformat für eine Arbeitsumgebung

Die Datei beginnt mit ihrem Erkennungszeichen "U". Hieran erkennt das Programm, daß eine Arbeitsumgebung geladen werden soll.

Danach folgen die globalen Variablen. Sie beinhalten die Anzahl der Funktionswerte(!), die eingestellten Arbeitsmodi und die Belegung der Fenster.

Darauf werden der Reihe nach die Funktionen $x(t)$, $y(t)$, $X(\omega)$ und $Y(\omega)$ abgespeichert. Am Anfang jeder Funktion werden die Parameter, die schon im Abschnitt "3.6.5. INFORMATION" beschrieben worden sind, gesichert. Schließlich ist der Funktionsvektor zeilenweise in der Datei abgelegt. Sehen Sie dazu auch in der Beschreibung des Datenformats für einzelne Funktionen.

3.15.3. Ein einfaches Format für komplexe Werte

Das folgende Bild zeigt einen Ausschnitt aus einer Datei für komplexe Werte:

```
Kennzeichen : E  
-0.000000, 1.000000  
0.024541, 0.999699  
0.049068, 0.998795  
0.073564, 0.997290  
0.098017, 0.995185  
0.122411, 0.992480  
0.146730, 0.989177  
0.170962, 0.985278  
0.195090, 0.980785  
0.219101, 0.975702  
0.242980, 0.970031  
0.266713, 0.963776  
0.290285, 0.956940  
0.313682, 0.949528  
0.336890, 0.941544  
0.359895, 0.932993  
0.382684, 0.923879
```

Abb. 16: Das Dateiformat für einen komplexen Funktionsvektor

Dieses Format ist bewußt sehr einfach gehalten. Es besteht nur aus dem Kennzeichen und den zeilenweise gespeicherten, komplexen Funktionswerten. Vor den Funktionswerten wird kein Kommentar erwartet. Es sollten soviele Funktionswerte vorhanden sind, wie der Funktionsvektor aufnehmen kann. Sind weniger Werte vorhanden erscheint eine Fehlermeldung. Diese kann ignoriert werden, da die bisherigen Funktionswerte ordnungsgemäß geladen worden sind. Die Parameter der Funktion werden auf die Standardwerte gesetzt.

3.15.4. Ein einfaches Format für einzelne Funktionswerte

Das folgende Bild zeigt einen Ausschnitt aus einer Datei für einzelne Funktionswerte:

```
Kennzeichen : S
Abtastperiode: 0.05
-0.000000
0.024541
0.049068
0.073564
0.098017
0.122411
0.146730
0.170962
0.195090
0.219101
0.242980
0.266713
0.290285
0.313682
0.336890
0.359895
0.382684
0.405241
0.427555
```

Abb. 17: Das Dateiformat für einen einfachen Funktionsvektor

Dieses Datenformat ist dem vorherigen sehr ähnlich. Es unterscheidet sich dadurch, daß jetzt nur einzelne Funktionswerte angegeben werden. Es werden auch alle Parameter standardmäßig gesetzt. Nur der Definitionsbereich wird der Abtastperiode entsprechend berechnet.

3.15.5. Das Format für eine Aufgabenstellung

Eine Aufgabendatei enthält eine Aufgabenstellung. Der Name der Datei wird abgefragt, wenn der Punkt AUFGABE STARTEN aufgerufen worden. Das Programm erwartet eine Datei mit folgendem Format:

Umgebung für die Aufgabenstellung: aufgabe1.sgl Darstellungsart : RE Lösungsfunktion: ergebn1.sgl :Bearbeiten Sie den rect in $x(t)$, so daß der rect in $y(t)$ entsteht.

Abb. 18: Das Dateiformat für eine Aufgabenstellung

Wie bei den vorherigen Formaten werden alle Zeichen vom Zeilenanfang bis zum Doppelpunkt als Kommentar aufgefaßt. Deshalb ist der Doppelpunkt am Zeilenanfang der letzten Zeile sehr wichtig.

Die erste Zeile enthält den Namen der Datei, die die Arbeitsumgebung zum Starten der Aufgabe enthält. Wenn Sie also eine Aufgabe entwerfen wollen, stellen Sie zuerst eine Arbeitsumgebung her, wie sie beim Aufgabenstart erscheinen soll. Die Arbeitsumgebung muß auch die Funktionen enthalten, die dann bearbeitet werden sollen.

Hinter "Darstellungsart" steht ein Schlüsselwort, das Auskunft gibt, ob als Lösung nur ein Realteil (RE), ein Imaginärteil (IM) oder beides (REIM) erwartet wird. Das Programm braucht die Information um zu wissen, aus welchen Funktionsteilen die Testdifferenz gebildet werden soll.

Die folgende Zeile enthält den Namen der Lösungsfunktion. Diese Funktion wird beim Beenden der Aufgabe geladen und mit dem erarbeiteten Ergebnis verglichen.

Die letzte Zeile enthält schließlich den eigentlichen Aufgabentext. Der Text darf beliebig lang sein. Wenn er nicht in das Fenster für die Aufgabenstellung paßt, wird der Rest abgeschnitten.

4. Programmbeschreibung

4.1. Einleitung

In diesem Abschnitt soll der Programmaufbau und einige Grundüberlegungen erläutert werden. Dabei wird nur kurz auf besonders zentrale Funktionen eingegangen.

Als zusätzliche Dokumentation steht ein Programmlisting und ein Referenzhandbuch, das eine Kurzbeschreibung aller Unterprogramme enthält, zur Verfügung.

Das Programm SIGNAL.EXE hat einen Umfang von ca. 230 KByte. Alle Treiber wurden direkt zum Programmcode gelinked, so daß keine zusätzlichen Dateien zum Ausführen des Programms notwendig sind.

Der Sourcecode ist ca. 16500 Zeilen (≈250 Seiten) lang. Er ist in neun Dateien aufgeteilt. Dadurch werden die einzelnen Teile übersichtlicher und es ist eine klare Hierarchie des Programms möglich.

4.1.1. Die Entwicklungsumgebung

Das Programm wurde mit dem Borland C++ Compiler V3.0 unter DOS 5.0 entwickelt. Dabei wurde allerdings nicht auf die objektorientierte Programmierung OOP von C++ zurückgegriffen. Soweit wie möglich wurde versucht den ANSI-C-Standard einzuhalten. Dies war bei speziellen DOS-Operationen wie Grafikausgabe und speziellen Ein-Ausgabe-Operationen nicht immer möglich. Dazu wurden einige TURBO-C-Befehle verwendet.

Compiliert wurde der Sourcecode für ein sog. Large-Memory-Model, d.h. sowohl für das Datensegment als auch für das Codesegment werden Far-Pointer verwendet. Das Code und Datensegment sind also beide größer als 64K und nur durch den unter DOS verfügbaren Speicher (<1024K) beschränkt.

Weiterhin wurde die Mathebibliothek für die 80x87 Mathecoprozessor emulation verwendet. Das Programm verwendet also eine Mathecoprozessor nur, sofern er vorhanden ist, andernfalls wird er emuliert.

Der Code wurde für 80286 und modernere Prozessoren erzeugt.

Beim Linkvorgang müssen folgende Objectcodedateien angegeben werden:

- | | |
|------------------|--------------------------------------|
| 1. MAIN.OBJ | |
| 2. GRAFIK.OBJ | |
| 3. MATHE1.OBJ | |
| 4. MATHE2.OBJ | |
| 5. M_FUNK.OBJ | |
| 6. FILE.OBJ | |
| 7. INFO.OBJ | |
| 8. MAUS.OBJ | |
| 9. NEUBAUER.OBJ | |
| 10. EGAVGA.OBJ | Grafiktreiber |
| 11. LITT.OBJ | Vektorzeichensatz |
| 12. GRAPHICS.LIB | Grafikbibliothek |
| 13. EMU.LIB | Bibliothek zur Emulation eines 80x87 |
| 14. MATHL.LIB | Mathebibliothek (Large-Memory-Model) |
| 15. CL.LIB | Codebibliothek (Large-Memory-Model) |
| 16. COL.OBJ | Objectcode (Large-Memory-Model) |

Die Dateien 12-16 werden bei Verwendung einer Entwicklungsumgebung mit Projektdatei automatisch dazugelinkt und brauchen dem Linker nicht speziell bekannt gegeben zu werden.

4.1.2. Der wesentlichen Probleme bei der Programmierung

Dies Programm soll an das von Frank Morgenstern erstellte Programm "SIGNL-ZB" anschließen. Dabei war ursprünglich geplant, Teile des bereits bestehenden Programms in das neu zu erstellenden zu übernehmen. Dies war leider nicht möglich, da sich dies Programm in wesentlichen Punkten von dem Vorgängerprogramm unterscheidet:

- Es werden komplexe Zahlen verarbeitet, so daß ein komplexer Funktionsvektor eingerichtet werden muß.
- Die Darstellung der Funktionen mußte wesentlich flexibler gestaltet werden, weil acht Funktionen (jeweils Real- und Imaginärteil von zwei Zeit- und zwei Frequenzfunktionen) in Fenstern ausgegeben werden müssen.
- Das Funktionsfeld ist nicht mehr fest auf 1024 Werte begrenzt, sondern ist dynamisch angelegt.
- Es können Diracimpulse und kontinuierliche Funktionswerte in einem Feld verarbeitet werden.

Die Grundprobleme, die bei der Programmierung gelöst werden mußten, waren:

- Die **Implementierung der DFT** in der Form, daß sie dem Benutzer wie eine "normale" kontinuierliche Fouriertransformation erscheint. Es mußten also spezielle Eigenschaften der DFT verborgen werden. Diese Aufgabe läßt sich allerdings nur begrenzt lösen.
- Die Verarbeitung von **Diracimpulsen** ist problematisch. Einige Operationen mit Diracimpulsen sind mathematisch nicht eindeutig definiert, z.B. ist die Multiplikation von zwei Diracimpulsen zwar mit der Ausblendeigenschaft zu begründen, mathematisch mit der Distributionentheorie aber nicht zu erklären. Ebenso ist nicht klar, wie Betrag und Phase einer Funktion aussehen, die im Real- und Imaginärteil einen Dirac hat. Ein anderes Problem tritt bei der DFT auf. Wenn Diracimpulse im Spektrum der DFT auftreten, müssen sie erkannt werden und als Pfeil dargestellt werden. Dies ist bei komplizierten Spektren nicht immer möglich. Diese Aufgabe übernimmt die Funktion **detectdirac()**.
- Die Funktionsfelder brauchen sehr viel **Speicherplatz**, so daß beim Debuggen des Programms häufig nicht genug Speicherplatz zur Verfügung stand. Deshalb gibt es in der Datei NT.h die Option "#define TEST NEIN". Wird hier TEST als JA definiert, erzeugt der Compiler einen Code, der nur drei der vier komplexen Funktionsfelder initialisiert. Auf diese Weise werden Speicherplatzprobleme vermieden. Wird das Programm ohne Compiler und Debugger ausgeführt, treten keine Probleme auf.
- Problematisch war auch die Rechengenauigkeit. Bei der Anwendung der DFT haben sich kleine Rundungsfehler gravierend ausgewirkt. Z.B. ist eine Funktion bei wiederholter Faltung im Feld nach rechts gewandert. Die Ursache dafür war, daß man nicht einfach eine Float-Zahl in eine Int-Zahl konvertieren darf. Bei der Umwandlung muß gleichzeitig gerundet werden. Zu diesem Zweck wurde die Funktion **toint()** geschrieben.

4.2. Programmaufbau

Der Gesamte Sourcecode besteht aus 10 Dateien, die im Folgenden aufgezählt werden:

1. NT.h
2. MAIN.C
3. GRAFIK.C
4. MATHE1.C
5. MATHE2.C
6. M_FUNK.C
7. FILE.C
8. INFO.C
9. MAUS.C
10. NEUBAUER.C

Durch die Aufteilung in mehrere Sourcedateien werden die einzelnen Teile übersichtlicher und lassen sich schneller compilieren. Außerdem entsteht eine klare Programmstruktur.

Abbildung 19 soll diese Struktur veranschaulichen. Die Pfeile zeigen in die Richtung der Datei auf die zugegriffen wird. Der Pfeil zwischen MAIN und INFO bedeutet z.B., daß in MAIN ein Funktion aus INFO aufgerufen wird.

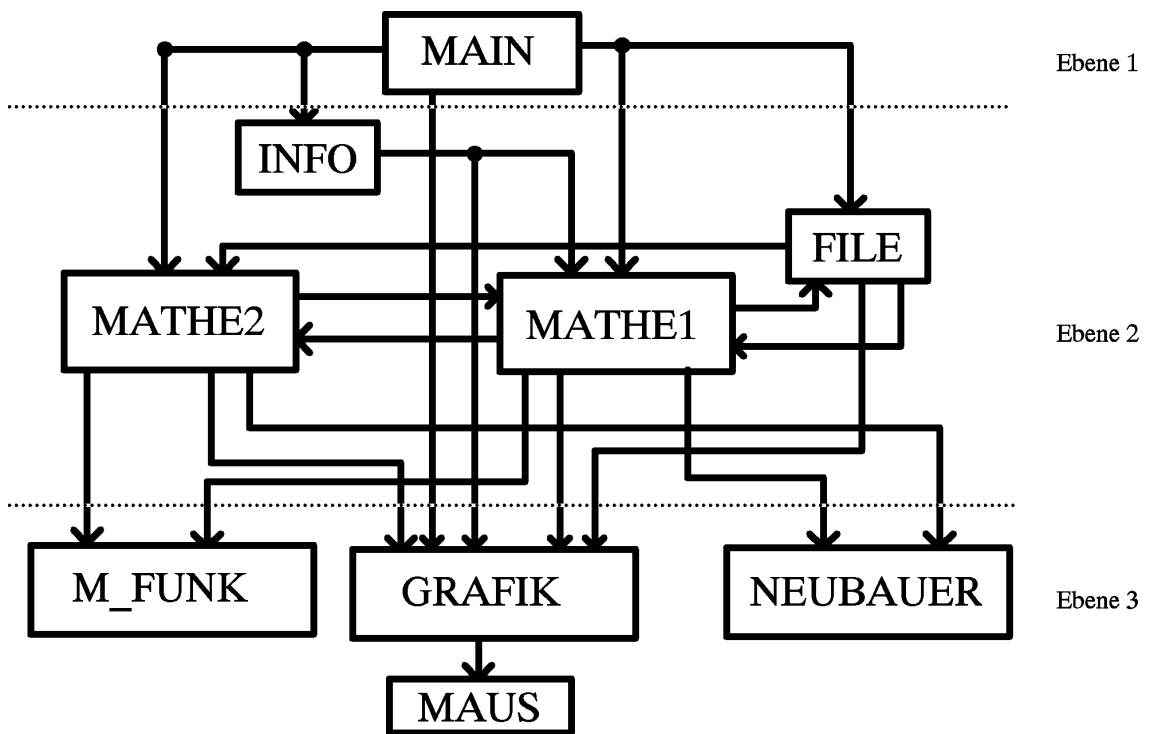


Abb. 19: Die Hierarchische Programmstruktur

Die gestrichelten Linien in Abb. 19 sind Trennlinien zwischen zwei Hierarchieebenen. Es finden keine Aufrufe von einer niedrigeren in eine höhere Ebene statt. Es dürfen nur Funktionen in Files, die sich in der gleichen oder niedrigeren Ebene befinden, aufgerufen werden.

Auf **externe, globale Variablen** greifen nur Funktionen aus Dateien der Ebene 1 und Ebene 2 zu. Z.B. wird auf die externe Variable **auf1**, die die Anzahl der Abtastwerte enthält, von allen Funktionen der Ebenen 1 und 2 zugegriffen.

4.3. Das Headerfile NT.h

Die Datei NT.h wird von allen anderen Sourcedateien außer der Datei MAUS.C mit `#include` eingebunden. Sie ist das Bindeglied zwischen den verschiedenen Programmteilen. In ihr sind die Prototypen aller Funktionen, die extern aufgerufen werden, definiert. Sie sind nach dem Sourcefile, in dem sich der Funktionskörper befindet, sortiert.

In NT.h werden alle notwendigen Includefiles eingebunden:

- alloc.h
- conio.h
- ctype.h
- dos.h
- graphics.h
- io.h
- math.h
- stdio.h
- stdlib.h
- string.h
- stddef.h
- time.h

Außerdem werden neue Typen definiert:

typedef float FZAHL

Dieser Typ ist eingeführt worden um eine Umstellung von einfachen float-Variablen auf double-Variablen zu erleichtern. Dazu kann der Typ FZAHL neu definiert werden.

typedef FZAHL CPLX[2]

Dies ist der Typ für komplexe Zahlen. Er ist nicht über eine Struktur definiert worden, da ein Zugriff über Indizes universeller ist. Mit den fest definierten Konstanten RE=0 und IM=1 kann auf Real- und Imaginärteil zugegriffen werden.

In NT.h werden außerdem eine ganze Reihe von Konstanten definiert. Es empfiehlt sich hier nicht ohne weiteres Änderungen vorzunehmen, da die meisten Konstanten in vielen Funktionen benutzt werden, so daß die Auswirkungen oft nicht genau absehbar sind. Die Konstante RE muß z.B. kleiner als IM sein, da beide als Parameter für for-Schleifen benutzt werden: for (i = RE; i <= IM; i ++).

Ansonsten enthält diese Datei noch die Aufzählung aller Farben. FARBE1 - FARBE4 sind die Schattenfarben der Fenster. Ihnen wurde direkt ein Eintrag in der Farbpalette zugewiesen, so daß das Ändern der Schattenfarbe über das Ändern des Eintrags in der Farbpalette vollzogen werden kann, ohne den Bildschirm neu auszugeben.

4.3.1. Die Struktur FUNKTION

Das Kernstück der Mathefunktionen bildet die Struktur FUNKTION. In ihr werden die vier Funktionen $x(t)$, $y(t)$, $X(\omega)$ und $Y(\omega)$ abgespeichert:

```
typedef struct {
    int call[2];
    int call2[2];
    CPLX par1;
    CPLX par2;
    FZ AHL xb1;
    FZ AHL xb2;
    CPLX yb1;
    CPLX yb2;
    CPLX xs1;
    CPLX xs2;
    CPLX ys1;
    CPLX ys2;
    int sym[2];
    int sym2[2];
    int dskr;
    FZ AHL ped[2];
    FZ AHL ped2[2];
    int begr[2];
    CPLX far *fkt;
    short far *drk;
} FUNKTION;
```

Im folgenden werden die Strukturelemente erklärt:

CPLX far *fkt:

Dies ist der Zeiger auf das komplexe Funktionsfeld, in dem alle Werte des Real- und Imaginärteils der Funktion diskret abgespeichert sind. Die Länge des Funktionsfeldes wird durch die Anzahl der diskreten Abtastwerte in der Variablen **auf1** bestimmt. Die Funktionswerte werden immer als Real- und Imaginärteil abgelegt, auch wenn die Darstellung nach Betrag und Phase erfolgt.

Es wird ein Far-Pointer verwendet, weil ein Funktionsfeld länger als 64K werden kann.

short far *drk:

Dieser Zeiger zeigt auf ein Feld, in dem vermerkt ist, ob an dieser Stelle ein Diracimpuls vorliegt. Es wird Bit 0 gesetzt, wenn ein Dirac im Realteil vorliegt, und Bit 1, wenn ein Dirac im Imaginärteil vorhanden ist. Daraus ergeben sich folgende Kombinationen:

- 0: kein Dirac vorhanden
- 1: Dirac im Realteil vorhanden
- 2: Dirac im Imaginärteil vorhanden
- 3: Dirac im Imaginär- und Realteil

Das Gewicht des Diracimpulses steht an der entsprechenden Stelle im komplexen Funktionsfeld **fkt**.

FZAHL xb1, xb2

Dies ist der Definitionsbereich, in dem das Zahlenfeld liegt. Soll eine Funktion z.B. im Bereich von -5 bis 5 abgespeichert werden, so wird $xb1=-5$ und $xb2=5$ gesetzt. Der Wert der im Feld den Index 0 hat, entspricht dann dem Funktionswert bei -5 und der Wert der im maximalen Index hat (z.B. 1023 bei $N=1024$) entspricht dem Funktionswert an der Stelle 5-T. Wobei die Abtastperiode

$$T = \frac{xb2 - xb1}{N} \text{ ist.}$$

Der Bereich ist für Real- und Imaginärteil immer gleich.

CPLX yb1, yb2

Dies ist der Wertebereich der Funktion. **yb1** enthält den minimalen Wert und **yb2** den maximalen Wert. **yb1** und **yb2** werden für Real- und Imaginärteil getrennt abgespeichert.

Wird die Funktion mit Betrag und Phase dargestellt, so enthält z.B. **yb1[RE]** den minimalen Betrag und **yb1[IM]** die minimale Phase.

CPLX xs1, xs2, ys1, ys2

Dies ist der Darstellungsbereich der Funktion. Die Funktion wird in einem Koordinatensystem, das auf der x-Achse von **xs1** bis **xs2** und auf der y-Achse von **ys1** bis **ys2** geht dargestellt. Werden Betrag und Phase dargestellt, so enthält der Feldeintrag für den Realteil die Koordinaten für den Betrag und der Feldeintrag für den Imaginärteil die Koordinaten für die Phase.

int dskr

Dies ist der Signaltyp:

- KONTI: wenn die Funktion kontinuierlich ist
- DISKR: wenn die Funktion diskret ist
- KEIN: wenn gar keine Funktion abgespeichert worden ist

Dieser Eintrag gilt für Real- und Imaginärteil.

int begr[2]

Dieser Eintrag wird für Real- und Imaginärteil getrennt behandelt. Er kann folgende Werte annehmen:

- JA: Das Signal ist auf der x-Achse begrenzt und liegt komplett im Definitionsbereich oder anders formuliert, es liegen nur Funktionswerte $\neq 0$ außerhalb des Bereiches **xb1** und **xb2**.

NEIN: Das Signal ist periodisch oder paßt nicht in den Definitionsbereich, so daß Teile abgeschnitten sind.

Dieser Parameter ist nur für die grafische Darstellung wichtig. Er hat keinen Einfluß auf interne Berechnungen mit der Funktion.

CPLX par1, par2

Beim Aufrufen der Funktion kann der Benutzer bis zu zwei Parameter zur Funktion eingeben z.B. die Breite des Rechtecks. Diese Parameter werden in den Variablen **par1** und **par2** abgelegt. Dadurch ist es möglich, die Funktion für genaue Berechnungen noch einmal nachzustellen.

Die folgenden Strukturelemente sind jeweils zweimal vorhanden: z.B. **sym** und **sym2**. Dies ist notwendig, weil diese Parameter für die Darstellung in polaren und kartesischen Koordinaten verschieden sind. Ist z.B. der Realteil einer Funktion gerade, so kann der Betrag trotzdem unsymmetrisch sein. Beim Wechsel der Darstellungsart werden die Inhalte der beiden Elemente einfach vertauscht.

int sym bzw. int sym2

Diese Variable gibt Auskunft über die Symmetrie der Funktion.

GERADE: gerade Funktion

UNGERADE: ungerade Funktion

NEIN: keine Symmetrie

Ursprünglich war geplant von der Symmetrie Rückschlüsse auf die Fourier-transformierte zu ziehen. Dies wurde aber nicht angewendet, so daß dieser Parameter außer einer informativen Aufgabe keine Funktion hat.

FZahl ped[2] bzw. FZahl ped2[2]

Ist eine Funktion periodisch, so enthält diese Variable die Periodendauer der Funktion. So ist es möglich eine periodische Funktion auch außerhalb ihres Definitionsbereichs darzustellen. Die Periodendauer muß kleiner als die Breite des Definitionsbereiches sein.

Ist diese Variable 0, so ist die Funktion nicht periodisch.

int call[2] bzw. int call2[2]

Hier wird die Nummer des Unterprogramms abgespeichert, mit der die Funktion erzeugt wurde. Die Unterprogramme zum Erzeugen einer Funktion sind in einer Tabelle abgelegt, so daß sie über diese Nummer aufgerufen werden können. Mit dieser Information ist es also möglich Funktionen bei Rechnungen ggf. neu zu erzeugen.

Nach Verknüpfungen und Bearbeitungen von Funktionen wird call auf KEIN gesetzt. Die Funktion ist jetzt nicht mehr analytisch und liegt nur noch als Funktionsvektor vor.

4.4. Das Sourcefile MAIN.C

MAIN.C enthält die Funktion main() mit der Hauptprogrammschleife. Hier wird der Ablauf des Programms koordiniert und die ganze Benutzereingabe gesteuert. Keine Funktion aus einer anderen Sourcedatei greift auf MAIN zu, so daß gewährleistet ist, daß der Ablauf des Programms nur von MAIN beeinflusst werden kann.

Abb. 20 zeigt den Ablauf der Hauptprogrammschleife:

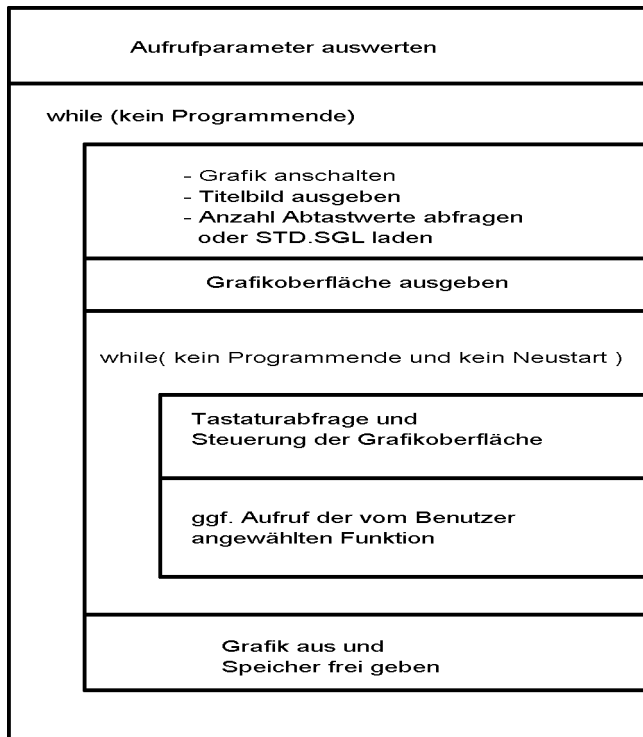


Abb. 20: Die Hauptprogrammschleife

In dieser Datei sind keine besonderen Spitzfindigkeiten vorhanden. Es werden Maus und Tastatur abgefragt. Je nach Benutzereingabe werden Pulldownmenüs ausgegeben oder Fenster dargestellt. Wird vom Benutzer eine Funktion aufgerufen, verzweigt MAIN in eine untergeordnete Sourcedatei.

4.4.1. Menüpunkt hinzufügen

Wie ein neuer Menüpunkt hinzugefügt wird, soll an einem Beispiel erklärt werden:

Es soll der Menüpunkt "MUSIK" in dem Pulldownmenu "Bearbeiten" eingefügt werden. Beim Aufrufen soll die Funktion **int mach_musik (void)** aufgerufen werden:

- Es muß in dem Feld des Pulldownmenus **m_bearb** der Eintrag "MUSIK" ergänzt werden. Wichtig ist, daß "\$" immer am Ende des Feldes steht.
- In dem Feld **helptext** muß ein Hilfstext eingegeben werden, der dann zur Erklärung in der Statuszeile erscheint.
- In der Tabelle **funkpoint** muß die Adresse der Funktion **&mach_musik** eingetragen werden.

Beim Eintragen in die verschiedenen Felder muß selbstverständlich auf die Reihenfolge der Einträge geachtet werden.

Will man eine neue Tastenfunktion hinzufügen, so muß der Tastencode in die Tabelle **keytab** und der Funktionspointer in die Tabelle **keypoint** eingetragen werden.

4.5. Das Sourcefile GRAFIK.C

In dieser Datei befinden sich alle Unterprogramme, die die Grafikausgabe steuern. Funktionen aus anderen Sourcefiles steuern die Ausgabe nur über GRAFIK.C. Auch die Abfrage der Maus erfolgt über diese Datei.

Es wird nur der VGA-Grafikmodus 640x480 mit 16 Farben unterstützt. Der Grafiktreiber "EGAVGA.BGI" wird als Objectfile beim Linken direkt eingebunden. Außerdem wird zur Textausgabe im Grafikmodus der Zeichensatz "LITT.CHR" verwendet. Auch er wird beim Linken eingebunden.

Einige Routinen, die Eingabefenster ausgeben oder Pulldownmenüs zeichnen, speichern den Bildschirmbereich, den sie überschreiben. Dazu werden die Funktionen **remove()** **remember()** aufgerufen. Es kann deshalb zu Speicherplatzproblemen kommen. Es sollte deshalb nicht mit zu großen Pulldownmenüs oder Eingabefenstern gearbeitet werden.

Im Folgenden wird auf zwei zentrale Themen eingegangen:

4.5.1. Die Benutzereingabe

GRAFIK.C stellt zwei Funktionen zur Abfrage von Benutzereingaben zur Verfügung:

1. Die Funktion getinfo

Der Prototyp der Funktion lautet:

```
int getinfo ( struct request abfrag[] )
```

Diese Funktion ist geeignet um mehrere Texteingaben abzufragen. Ein Beispiel ist die Abfrage der Skalierung. Zur Parameterübergabe wird die Struktur request verwendet:

```
struct request { int art;  
                 char *texta;  
                 char *twert;  
                 float fwert;  
                 };
```

int art

In diesem Element wird eingetragen, welche Art Parameter abgefragt werden soll.

FLT: ein Zahleneingabe
TXT: eine Texteingabe

Wird FLT angegeben können keine alphanummerischen Eingaben gemacht werden. TXT erlaubt Zahlen und Buchstaben zur Eingabe.

char *texta

Dies ist der Text, der vor dem Eingabefeld ausgegeben wird, z.B. "Bitte Name eingeben:".

char *twert

Dieser Textstring kann einen Eingabevorschlag enthalten, der ggf. vom Benutzer editiert werden kann. Soll hier eine Zahl erscheinen muß sie mit der Funktion **zahlstr** in einen String convertiert werden. Die Länge dieses Strings bestimmt die Größe des Eingabefeldes. Enthält dieser String acht Leerzeichen, so können auch nur acht Zeichen eingegeben werden.

Mit diesem String wird die Benutzereingabe zurückgegeben.

float fwert

Wenn in der Variablen **art** FLT eingetragen worden ist, wird mit diesem Element zusätzlich die Benutzereingabe im Floatformat übergeben.

Der Funktion **getinfo** wird die oben erklärte Struktur als Feld übergeben, so daß in einem Fenster mehrere Eingaben gemacht werden können. So ein Feld könnte folgendermaßen aussehen:

```
struct request skala[] = {
    { FLT,"von x-Achse:", "-10  " },
    { FLT,"bis x-Achse:", "10  " },
    { FLT,"von y-Achse:", "-5  " },
    { FLT,"bis y-Achse:", "5  " },
    { KEIN }
};
```

Wichtig ist, daß das Feld als Endekennzeichen den Wert KEIN enthält, sonst versucht die Funktion eine unbestimmte Anzahl Eingaben abzufragen.

2. Die Funktion ask_box

Der Prototyp der Funktion lautet:

```
int ask_box ( char *text, char *te1, char *te2 )
```

Die Funktion zeichnet ein Fenster und gibt in dem Fenster den Text **text** aus, z.B. "Wollen Sie Tee mit Zucker?". In drei kleinen Kästchen erscheinen die Auswahlmöglichkeiten, die mit der Maus angeklickt werden können. Im linken Kästchen wird der Text te1 (z.B: "Milch"), im mittleren Text te2 (z.B. Zucker) und im rechten immer "ABBRECHEN" ausgegeben. Wenn der Benutzer eine Auswahl getroffen hat, kehrt die Funktion mit folgenden Parametern zurück:

JA:	wenn das linke Kästchen ausgewählt wurde
NEIN:	wenn das mittlere Kästchen ausgewählt wurde
KEIN:	wenn "ABBRECHEN" ausgewählt worden ist

4.5.2. Die Funktionsausgabe

Zum Ausgeben von Funktionsfeldern stellt GRAFIK.C eine Reihe von Funktionen zur Verfügung, die nacheinander aufgerufen werden müssen.

setfen initialisiert die Parameter zur Funktionsausgabe. Mit dieser Funktion wird die Nummer des Fensters übergeben, in das geplottet werden soll. Daraus wird dann ein Offset erzeugt, so daß die eigentliche Plottfunktion mit relativen Koordinaten arbeiten kann. Zurückgegeben wird die Anzahl Werte, die auf der x-Achse nebeneinander ausgegeben werden können. Dies ist also die Anzahl Pixel auf der x-Achse. Diese Übergabe ist notwendig, da große Fenster wesentlich genauer ausgeben können als kleine Fenster.

koordinaten zeichnet in das zuvor initialisierte Fenster ein Koordinatensystem.

plotout zeichnet einen Funktionswert in das Fenster ein. Der Funktion wird die Position übergeben, an der der Wert ausgegeben werden soll. Dieser Wert wird in Pixeln gezählt und kann Integerwerte von 0 bis zum von **setfen** zurückgegebenen Wert annehmen.

Außerdem wird der y-Wert als Floatzahl übergeben. Dieser Wert muß auf 1000 bezogen sein, damit **plotout** ihn richtig ins Fenster einpassen kann. Zur Berechnung dient folgende Formel:

$$y_{rel} = \frac{y_{funkt} - y_{k1}}{y_{k2} - y_{k1}} * 1000 \quad (119)$$

y_{k1} und y_{k2} ist der y-Bereich des Koordinatensystems

y_{funkt} ist der Funktionswert der geplottet werden soll

y_{rel} ist der Wert in auf 1000 bezogenen Koordinaten

Der Funktion muß außerdem noch übergeben werden, was für eine Art Funktionswert geplottet werden soll.

KONTI: Der aktuelle Funktionswert wird mit dem vorherigen durch eine Linie verbunden.

DIRAC: Es wird ein Pfeil gezeichnet.

DISKR: Es wird ein Stecknadelkopf gezeichnet.

resetfen kann ohne Parameter aufgerufen werden und setzt alle internen Einstellungen zurück.

4.6. Das Sourcefile MAUS.C

Hier befinden sich fünf Funktionen zur Maussteuerung. Sie verwenden den Interrupt 33, der die Schnittstelle zum Maustreiber bildet.

Da die Grafikfunktionen von C direkt auf den Videospeicher zugreifen, bemerkt der Maustreiber nicht, wenn sich der Untergrund des Mauspeils geändert hat. Deshalb muß vor jeder Grafikoperation der Mauszeiger ausgeschaltet werden.

4.7. Das Sourcefile M_FUNK.C

In dieser Datei befinden sich kurze Mathe- und Konvertierungsfunktionen. Dazu gehören Funktionen wie $\text{rect}(t)$, $\text{si}(t)$, $\Lambda(t)$, $\text{sign}(t)$ usw. Es werden verschiedene Konvertierungsfunktionen zur Verfügung gestellt, z.B. **zahlstr()** zum Umwandeln einer Zahl in einen String.

Außerdem befinden sich hier Funktionen, zur Bearbeitung von komplexen Zahlen, z.B. **real()**, **imag()**, **phase()** usw.

4.8. Das Sourcefile NEUBAUER.C

Funktionen in diesem File waren bereits von Andre Neubauer implementiert worden und brauchten nur noch eingebunden werden. Kernstück ist die Fast Fouriertransformation FFT. Sie wurde nach dem Algorithmus von Cooley-Tukey programmiert.

Außerdem wird ein Rauschgenerator, der annähernd Weißes Rauschen erzeugt, zur Verfügung gestellt. Hier liegt der Box-Muller-Algorithmus zugrunde.

4.9. Das Sourcefile MATHE1.C

Dies ist das längste Sourcefile. Es bildet zusammen mit MATHE2.C den Hauptteil der mathematischen Operationen.

Die Funktionen aus MATHE1.C lassen sich in sechs Themengruppen einteilen:

Funktionen zum Zugreifen auf die komplexen Felder:

getfeld()	putfeld()
maxwert()	clrdirac()
clrfunk()	getnext()
getwert()	

Initialisierungsfunktionen:

init_skala()	init_par
init_feld()	init_mathe()
reset_mathe()	

Funktionen zum Überprüfen und Berechnen von Parametern:

testnull()	linksnull()
rechtsnull()	checkkoord()
checkzwei()	checkpar()
chkfunkdirac()	testdirac()
get_funk()	changepar()

Funktionen, die die Funktionsfelder darstellen und Parameter dafür setzen:

skale_new()	skale_newy()
getkoord()	funkplot()
fenplot()	

Funktionen, die Benutzerabfragen durchführen:

getdef()	getabtast()
getauto()	getskala()
changedef()	changebet()
ganzeftk()	

Funktionen zum Bearbeiten und Verknüpfen von Funktionsfeldern:

dadjust()	adjustall()
anpassen()	alltrans()
dft()	splitdirac()
splitkonti()	merge()
c_trans()	detectdirac()
exchange()	kopieren()
addition()	differenz()
multiplikation()	division()
dehnen()	expand()
movefunkt()	mirrorx()
magnify()	drehung()
convolution()	convol()
correlation()	

Zwei Funktionen sollen in besonderer Weise dargestellt werden: die Anpassungsroutine **dadjust()** und die diskrete Fouriertransformation **dft()**.

4.9.1. Die Anpassungsfunktion **dadjust()**

Diese Funktion rechnet den ihr übergebenen Funktionsvektor in einen neuen Bereich um. Dies ist z.B. notwendig, wenn zwei Funktionen $x(t)$ und $y(t)$ addiert werden sollen. Ist die Funktion $x(t)$ im Bereich -5 bis 5 und die Funktion $y(t)$ im Bereich -6 bis 4 definiert, so können die beiden Funktionsfelder nicht direkt addiert werden. Zuerst müssen beide auf den gleichen Bereich z.B. -6 bis 5 angepaßt werden. Liegt die Funktion noch analytisch vor, so wird das Feld für den neuen Bereich einfach neu berechnet. Ist diese Information nicht mehr bekannt, so tritt die Funktion **dadjust()** in Aktion.

Ihr Prototyp lautet:

```
void dadjust (FUNKTION *f_poi, FZAHL xk1, FZAHL xk2, int re_im)
```

Der Funktionsvektor der Funktionsstruktur **f_poi** wird auf den neuen Bereich **xk1**, **xk2** umgerechnet. **Re_im** gibt an, ob Real- oder Imaginärteil umgerechnet werden sollen. Diese Funktion kann nicht auf Betrag oder Phase angewendet werden.

Abb. 21 zeigt das Struktogramm der Funktion:

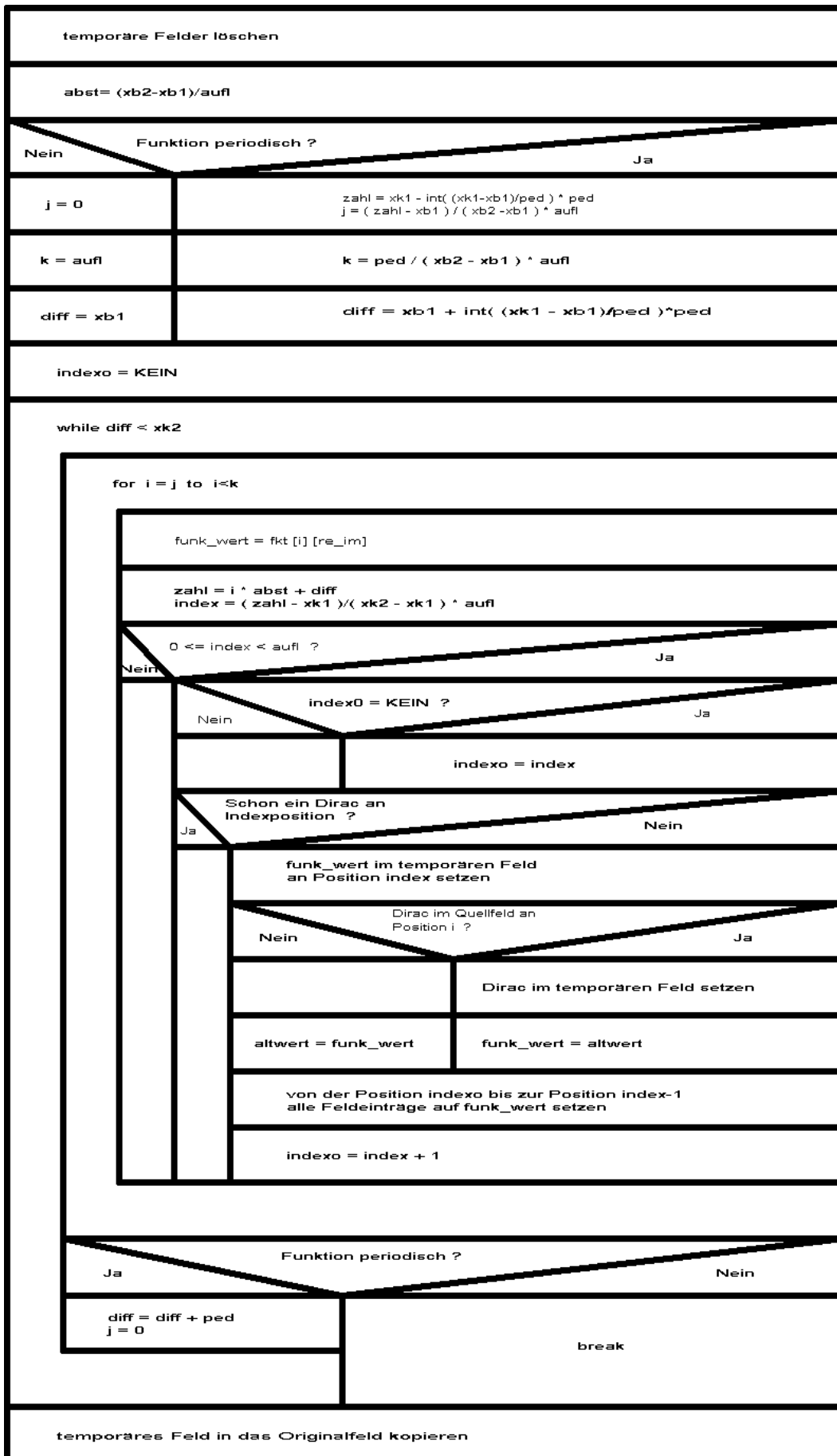


Abb. 21: Struktogramm der Funktion dadjust()

Abb. 22 soll die in der Funktion errechneten Parameter veranschaulichen. **diff** ist der nächste kleinere Beginn einer neuen Periode vor dem neuen Bereichsbeginn **xk1**. **zahl** ist der Abstand zwischen **diff** und **xk1**.

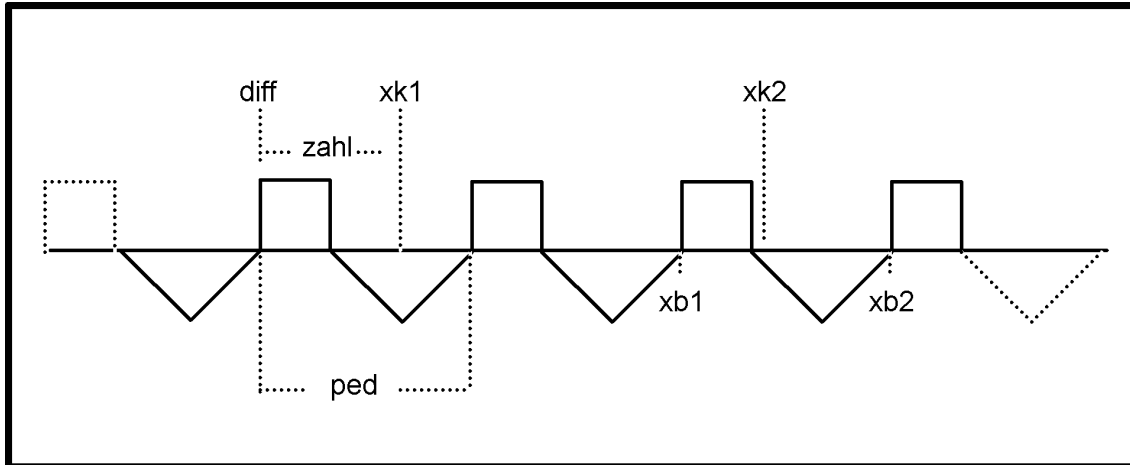


Abb. 22: Interne Parameter der Funktion dadjust()

dadjust() geht das Quellfeld Eintrag für Eintrag durch und prüft, ob der entsprechende Funktionswert im neuen Definitionsintervall liegt. Wenn ja, wird der Funktionswert in den neuen Bereich übertragen. Ist die Abtastperiode des neuen Bereichs kleiner, so ist es notwendig mehrer Einträge zu füllen. Gefüllt werden dann die Einträge vom Index **indexo** bis zum Index **index-1**.

Ist eine Funktion periodisch, so muß das alte Feld immer wieder durchlaufen werden bis alle Perioden innerhalb des neuen Bereichs übertragen worden sind.

4.9.2. Die diskrete Fouriertransformation dft()

Die diskrete Fouriertransformation **dft()** wird im Programm nach jeder Operation aufgerufen, die eine Funktionsstruktur verändert hat. Auf diese Weise ist die Darstellung immer auf dem aktuellsten Stand.

Die eigentliche Transformation wird von der Funktion **fft()** in der Datei NEUBAUER.C ausgeführt. Aufgabe von **dft()** ist es, alle Parameter der Transformierten zu setzen und die Funktion so umzurechnen, daß sie dargestellt werden kann.

Abb. 23 zeigt das Struktogramm der Funktion.

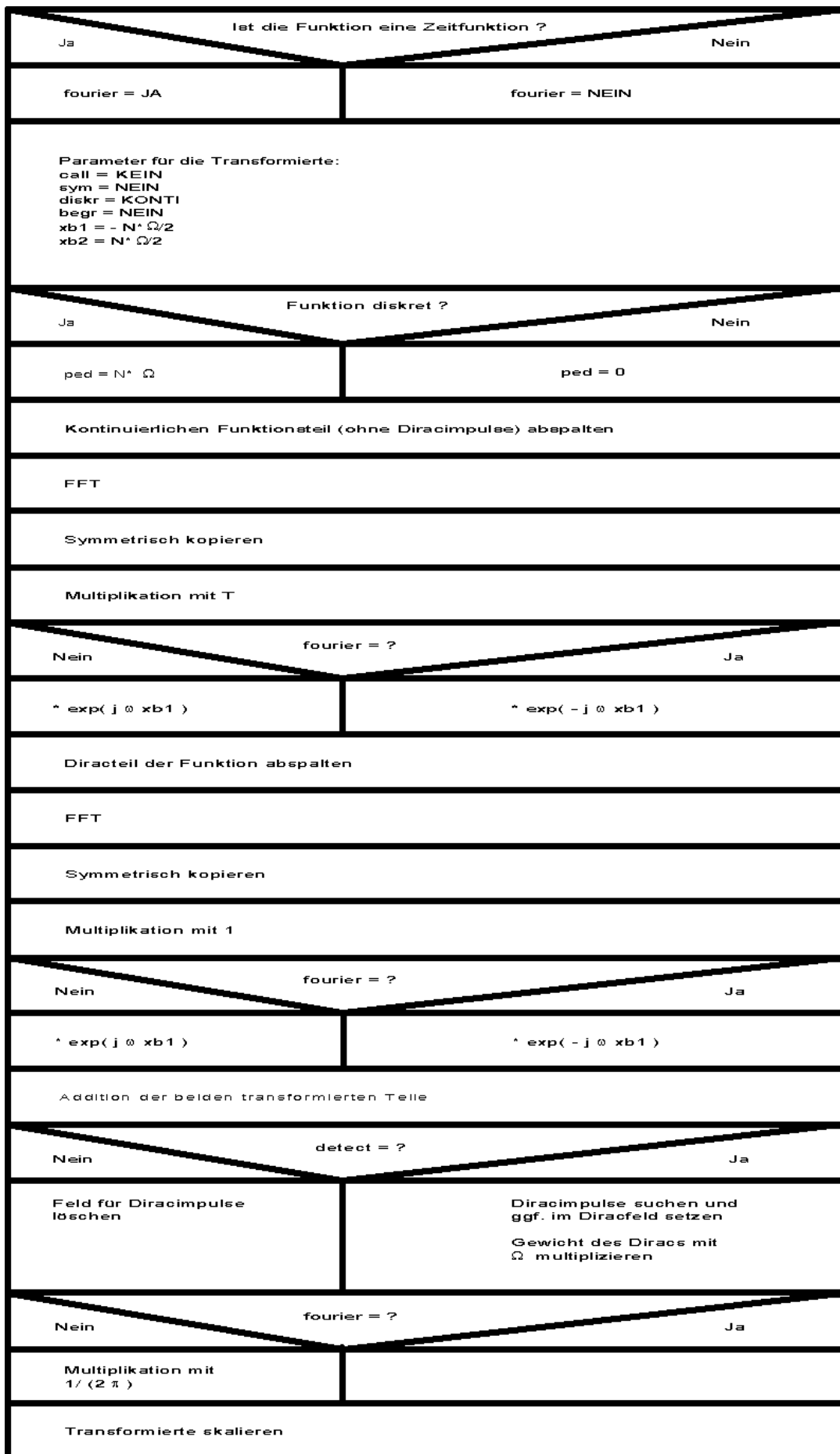


Abb. 23: Das Struktogramm der Funktion dft()

Es wird zunächst festgestellt, in welche Richtung die Fouriertransformation angewendet werden soll. Die entsprechende externe Variable **fourier** wird demnach gesetzt. Danach werden die Parameter für die Transformierte gesetzt:

- Die Funktion ist nicht analytisch: **call** = KEIN.
- Die Funktion ist nicht symmetrisch: **sym** = NEIN.
- Der Definitionsbereich wird symmetrisch zur y-Achse gesetzt. Dabei wird die Gleichung für die Abtastperiode im Zeit- und Frequenzbereich (Glg. 66) verwendet:

$$\Omega \cdot T \cdot N = 2\pi$$
- Die Funktion ist nicht begrenzt: **begr** = NEIN.
- Wenn die Ursprungsfunktion diskret ist, wird für die Transformierte die Periodendauer auf die Breite des Definitionsbereiches gesetzt. Ansonsten wird die Periode auf Null gesetzt.

Da in einer Funktionsstruktur sowohl Diracimpulse als auch der kontinuierliche Funktionsteil verarbeitet werden, muß jeder Teil getrennt transformiert werden und anschließend addiert werden:

$$\mathbf{s}(t) = \mathbf{s}_k(t) + \mathbf{s}_d(t) \xrightarrow{F} \mathbf{S}(\omega) = \mathbf{S}_k(\omega) + \mathbf{S}_d(\omega)$$

\mathbf{S}_d : Diracimpulse
 \mathbf{S}_k : kontinuierlicher Teil

Zuerst wird der kontinuierliche Teil abgespalten und transformiert.

Da die FFT ein Feld liefert, das mit Null beginnt, wird das Feld jetzt so kopiert, daß das Intervall symmetrisch zur y-Achse liegt.

Auf Grund des Zusammenhangs zwischen DFT und kontinuierlicher FT (Glg. 82, S. 21):

$$\mathbf{S}_{d,kontinuierlicheFT} = T \cdot \mathbf{S}_{d,DFT}$$

wird das Feld mit **T** multipliziert.

Außerdem geht die FFT davon aus, daß das Ursprungsfeld auch bei Null beginnt. Es muß also noch um **xb1** verschoben werden. Dazu macht man sich den Satz über die Verschiebung (Glg. 5, S. 3) zunutze:

$$\mathbf{s}(t - t_0) \xrightarrow{F} \mathbf{S}(\omega) \cdot \mathbf{e}^{-j\omega \cdot t_0}$$

Das transformierte Feld wird also mit $\mathbf{e}^{-j\omega \cdot \mathbf{xb1}}$ multipliziert.

Die gleiche Prozedur wird mit dem Diracanteil der Funktion wiederholt. Dabei ist die Multiplikation mit dem Faktor **T** aber nicht mehr nötig.

Beide Anteile werden jetzt addiert.

Problematisch ist die Entdeckung der Diracimpulse im transformierten Feld. Diese ist notwendig, weil das Gewicht eines Diracimpulses noch mit dem Faktor Ω multipliziert werden muß und weil im Diracfeld ein Flag gesetzt werden muß, damit der Diracimpuls als Pfeil dargestellt wird. Diese Aufgabe ist noch nicht zufriedenstellend gelöst worden, da in manchen Fällen Diracimpulse falsch erkannt werden.

Um die inverse FT zu berechnen wird von dem Zusammenhang über die Symmetrie der Fouriertransformation (Glg. 6, S. 3) gebrauch gemacht:

$$s(t) \xrightarrow{F} S(\omega) \Rightarrow S(t) \xrightarrow{F} 2\pi \cdot s(-\omega)$$

Das Ergebnisfeld wird deshalb ggf. mit $\frac{1}{2\pi}$ multipliziert.

Schließlich werden die Parameter für die Skalierung gesetzt.

4.10. Das Sourcefile MATHE2.C

MATHE2.C ist eine Ergänzung zu MATHE1.C. Hier werden alle Funktionen definiert:

Funktionsnummer	Erstellungsfunktion	Initialisierungsfunktion	Kommentar
0	get_clr	ini_clrall	leere Funktion
1	get_sin	ini_sin	Sinus
2	get_cos	ini_cos	Cosinus
3	get_rect	ini_rect	Rechteck
4	get_delta	ini_delta	Deltaimpuls
5	get_dfolge	ini_dfolge	Impulsfolge
6	get_gauss	ini_gauss	gauss. Glockenkurve
7	get_tri	ini_tri	Dreiecksfunktion
8	get_si	ini_si	si-Funktion
9	get_konst	ini_konst	Konstante
10	get_sprung	ini_sprung	Sprungfunktion
11	get_sign	ini_sign	Signumfunktion
12	get_saege	ini_saege	Sägezahnfunktion
13	get_rectfl	ini_rectfl	Rechteckfolge
14	get_noise	ini_noise	Rauschen
15	get_sind	ini_sind	Sinus (diskret)
16	get_cosd	ini_cosd	Cosinus (diskret)
17	get_rectd	ini_rectd	Rechteck (diskret)
18	get_deltad	ini_deltad	Einheitssprung (diskret)
19	get_sprungd	ini_sprungd	Sprungfunktion (diskret)
20	get_expfolg	ini_expfolg	Exponentialfolge

Abb. 24: Tabelle aller im Programm implementierten Funktionen

Abb. 24 führt alle Funktionen auf, die in dem Programm implementiert worden sind. Die erste Spalte enthält die Funktionsnummern, die schon im Kapitel "4.3.1. Die Struktur FUNKTION" erwähnt worden sind. Diese Nummern werden in das Strukturelement **call** eingetragen, damit ein Funktionsfeld neu berechnet werden kann.

In der zweiten Spalte befinden sich die Erstellungsfunktionen. Diese Funktionen werden auch aufgerufen, wenn eine Funktion neu erstellt wird. Der Prototyp lautet:

```
int get_XXX (FUNKTION *f_poi, int re_im)
```

Es muß der Pointer auf die Funktionsstruktur, für die die Funktion erstellt werden soll, übergeben werden. Außerdem muß noch mitgeteilt werden, ob Real- oder Imaginärteil berechnet werden sollen.

Für den Aufruf dieser Funktion müssen der Definitionsbereich (**f_poi->xb1**, **f_poi->xb2**) und die Parameter (**f_poi->par1** und **f_poi->par2**) gesetzt sein. Es werden dann folgende Daten in die übergeben Funktionsstruktur zurückgegeben:

- Die Symmetrie wird gesetzt (**f_poi->sym**).
- Es wird bestimmt, ob das Signal begrenzt ist (**f_poi->begr**).
- Die Periode des Signals wird gesetzt (**f_poi->ped**).
- Der minimale und maximale Funktionswert wird bestimmt (**f_poi->yb1** und **f_poi->yb1**).
- Schließlich wird der komplexe Funktionsvektor (**f_poi->fkt**) und des Feld für die Diracimpulse (**f_poi->drk**) berechnet.

In der zweiten Spalte stehen die entsprechenden Initialisierungsroutinen. Diese Funktionen werden aufgerufen, wenn der entsprechende Menüpunkt in den Pulldownmenüs ausgewählt worden ist. Hier werden die Parameter gesetzt, die für den Aufruf von **get_XXX()** benötigt werden:

- Nach einer Benutzereingabe werden die Parameter **f_poi->par1** und **f_poi->par2** gesetzt.
- Die Funktionsvektoren werden ggf. gelöscht.
- Die Nummer der get-Funktion wird gesetzt (**f_poi->call**).
- Die Art des Signals wird bestimmt (**f_poi->dskr**).
- Der Definitionsbereich wird berechnet (**f_poi->xb1** und **f_poi->xb2**).
- Die Skalierung wird gesetzt (**f_poi->xs1**, **f_poi->xs2**, **f_poi->ys1**, **f_poi->ys2**).
- Die get-Funktion wird aufgerufen.
- Die Fouriertransformierte wird errechnet **dft(f_poi)**.

Jeweils alle Erzeugungs und Initialisierungsfunktionen sind nach diesem Schema aufgebaut.

4.11. Das Sourcefile INFO.C

Diese Datei ist ursprünglich geschrieben worden, um beim Debuggen des Programms auf schnelle Art und Weise die Funktionsparameter und Felder überprüfen zu können. Dabei hat sich herausgestellt, daß diese Informationen auch für den Benutzer interessant sind, und diese Funktion wurde im Programm belassen.

Kernstück ist das Unterprogramm **showinfo()**, das aus einer Aneinanderreihung von **outtext()**-Befehlen besteht, die nacheinander alle wichtigen Parameter ausgeben.

4.12. Das Sourcefile FILE.C

Hier befinden sich alle Funktionen zur Datenein- und ausgabe: Funktionen werden geladen und gespeichert und Aufgaben werden gestartet und beendet.

Ebenso wie in INFO.C stecken in dieser Datei keine komplizierten Algorithmen. Die Funktionen sind nur so lang geworden, da sehr viele verschieden Parameter geladen und gespeichert werden. Dabei findet eine ausführliche Prüfung der Parameter statt, um falsche Ein- Ausgaben zu vermeiden.

Kernstücke sind die Funktionen:

- **savefunk()** zum Speichern der Funktion im aktuellen Fenster,
- **saveumg()** zum Speichern aller Funktionen und Parameter,
- **loadumg()** zum Laden aller Funktionen und Parameter,
- **loadfunk()** zum Laden der Funktion in das aktuelle Fenster,
- **loadfremd()** zum Laden eines einfachen komplexen Datenformats,
- **loadsingel()** zum Laden eines einfachen reellen Datenformats,
- **strtaufg()** startet eine Aufgabe und
- **stopaufg()** beendet eine Aufgabe.

5. Zusammenfassung

5.1. Verbesserungsmöglichkeiten

Dieses Programm ist im Rahmen einer Studienarbeit entstanden. Der Zeitrahmen von drei Monaten ist relativ knapp bemessen, so daß nicht alle Ideen im vollen Umfang umgesetzt werden konnten. Folgende Verbesserungen wären denkbar:

- Die analytische Darstellung der Funktionen könnte erweitert werden. Dazu könnte gehören, daß Funktionen direkt eingegeben werden können z.B. " $\sin(2x)\cdot\text{rect}(t/3)$ " und daß das Programm die Bearbeitungen und Verknüpfung analytisch verfolgt. Wenn der Benutzer also eine rect-Funktion und eine cos-Funktion addiert, merkt der Computer sich diesen Vorgang als eine interne Formel.
Das Programm könnte so sehr viel genauer rechnen und würde den Schritt von einer formelmäßigen Operation zur Darstellung verdeutlichen. So würde z.B. direkt klar, daß die Multiplikation des Argument einer Funktion $f(b\cdot t)$ mit einer Konstanten einer Dehnung auf der x-Achse oder gar einer Spiegelung an der y-Achse entspricht.
- Verbessert werden kann sicherlich auch die Benutzereingabe. Es ist z.B. nicht möglich Directories zu laden und eine direkte Benutzerauswahl mit Maus anzubieten. Ebenso können die Funktionen nicht mit der Maus bearbeitet werden, wie dies im Programm von Frank Morgenstern realisiert worden ist.
- Es können auch noch weitere Operationen ergänzt werden. So könnte z.B. der Hilberttransformator implementiert werden.
- Außerdem könnte man die Fouriertransformation um die Laplacetransformation erweitern. Ebenso wäre es denkbar für diskrete Signale die Z-Transformation zu implementieren.
- Interessant wäre auch eine direkte Eingabe von Signalen über einen A/D-Wandler zu ermöglichen. Dadurch wäre eine praxisnahe Anwendung des Programms in der Nachrichtentechnik möglich.

Begrenzt sind Erweiterungen und Verbesserungen des Programms allerdings durch den Speicher unter DOS. Das Programm ist schon ca. 230K lang, so daß Erweiterungen auf Kosten des Speichers gehen, der für die Sicherung der Funktionsvektoren benutzt werden kann.

5.2. Schlußwort

Ich selber habe bei der Anwendung -nicht nur bei der Programmierung- des Programms einiges über die Grundlagen der Signaldarstellung gelernt. Ich kann mir gut vorstellen, daß es für Studenten der Elektrotechnik hilfreich und anschaulich ist, mit diesem Programm herumzuspielen.

6. Verwendete Formelzeichen

a	reelle Konstante
b	reelle Konstante
B	Breite des Definitionsbereiches
E	Signalenergie
f	Parameter im Frequenzbereich (Frequenz)
$f(t)$	Zeitfunktion
$f_d(k)$	diskrete Funktion
F	Abtastperiode im Frequenzbereich
$g(t)$	Zeitfunktion
$g(\omega)$	Funktion im Frequenzbereich
$h(t)$	Stoßantwort
$H(\omega)$	Übertragungsfunktion
k	natürliche Zahl (Parameter für diskrete Funktionen)
n	natürliche Zahl (Parameter für diskrete Funktionen)
m	natürliche Zahl (konstanter Parameter für diskrete Funktionen)
M	Anzahl Multiplikationen
N	Anzahl diskreter Werte
$s(t)$	Zeitfunktion
$s_a(t)$	abgetastete Funktion
$s_d(t)$	diskretes Zeitsignal
$s_p(t)$	periodisches, diskretes Zeitsignal
$s_r(t)$	Rechteckfolge
$s_z(t)$	Sägezahnfunktion
$s_f(t)$	Sägezahnfolge
$S(\omega)$	Funktion im Frequenzbereich
$S_a(\omega)$	Fouriertransformierte einer abgetasteten Zeitfunktion
$S_d(\omega)$	diskretes Frequenzsignal
$S_p(\omega)$	periodisches, diskretes Frequenzsignal
t	Parameter im Zeitbereich
t_0	konstanter Parameter im Zeitbereich
t_1	konstanter Parameter im Zeitbereich
t_2	konstanter Parameter im Zeitbereich
T	Abtastperiode im Zeitbereich, Dehnungs- und Stauchungsfaktor
T_0	Dehnungs- und Stauchungsfaktor,
$x(t)$	Zeitfunktion
X_1	linke Grenze des Definitionsintervalls
X_2	rechte Grenze des Definitionsintervalls
$X(\omega)$	Funktion im Frequenzbereich
$y(t)$	Zeitfunktion
$Y(\omega)$	Funktion im Frequenzbereich

α	reelle Konstante
δ	Deltadistribution
δ_{t_0}	Deltadistribution um t_0 nach rechts verschoben
$\varepsilon(t)$	Sprungfunktion
φ	Phasenwinkel
$\Lambda(t)$	Dreiecksfunktion
ρ_{sg}^E	Korrelationskoeffizient
$\rho_{sg}(\tau)$	Kreuzkorrelation
π	Kreiskonstante Pi
τ	Integrationsvariable
ω	Parameter im Frequenzbereich (Kreisfrequenz)
ω_0	konstanter Parameter im Frequenzbereich
ω_a	Periodendauer im Frequenzbereich
ω_g	Nyquist-Grenzfrequenz
Ω	Abtastperiode im Frequenzbereich (Kreisfrequenz)
$\amalg(t)$	Deltaimpulsfolge
$\mathfrak{R}_{ss}(\omega)$	Energiedichtespektrum
\xrightarrow{F}	Fouriertransformation
$\xrightarrow{F^{-1}}$	inverse Fouriertransformation

7. Literaturverzeichnis

- [1] Lüke, Signalübertragung, 5. Auflage
1992 Springer-Verlag

- [2] Oran Brigham, FFT, 4. Auflage
1989 Oldenbourg Verlag

- [3] Heinz Luck, Vorlesungsskript NT1 - NT3
1991 - 1993 Universität Duisburg

- [4] Frank Morgenstern, Simmulationsprogramm zur
Veranschaulichung der Signaltheorie, Teil1: Darstellung im
Zeitbereich, 1993 Diplomarbeit im Fachgebiet
Nachrichtentechnik an der Uni-GH-Duisburg