Komponenten und Programmierung

Scilab und Scicos

SS 2008

Stand 24.06.2008

1 Scilab – ein Kurzeinstieg

1.1 Open-Source- und Open-Domain-Clone von Matlab

Scilab ist ein frei verfügbares, sehr mächtiges Programmsystem, das in Intention und Syntax sehr der (mittlerweile) kommerziellen Software Matlab ähnelt. Im Gegensatz zu Matlab, das nur in seinen Anfangsjahren eine Open-Source und Open-Domain-Software war, ist Scilab auch in der Industrie ohne Einschränkungen und ohne Kosten frei einsetzbar. Seine Quellen sind offen, so daß man sie durch eigene Programmmodule leicht ergänzen und modifizieren kann. Dies ist vor allem für mittelständische Firmen höchst interessant, die sich die hohen (auch laufenden!) Kosten für Matlab nicht leisten können und wollen. Zu diesem Zweck hat das Institut National de Rechere en Informatique et Automatique (I.N.R.I.A) entwickelt. Es wird laufend erweitert und aktualisiert. Die neueste Version kann man sich aus dem Internet unter www.**scilab**.org/download/ herunterladen.
Scilab beinhaltet auch einen graphischen Editor ähnlich Simulink, mit dem man dynamische System aus Blöcken zusammensetzen und untersuchen kann.

Neben englischsprachiger Literatur gibt es auch sehr gute deutschsprachige Unterlagen. Besonders erwähnt werden soll die didaktisch hervorragende Einführung „Arbeiten mit Scilab und Scicos“ ([**2**]) von Jean-Marie Zogg (HTW Chur), das als pdf-Datei im Internet unter www.fh-htwchur.ch/uploads/media/Arbeiten_mit_Scilab_und_Scicos_v1.pdf erhältlich ist.

Eine weitere sehr gute Einführung ist die deutsche Übersetzung des Unterrichtstextes „Eine Einführung in Scilab“ von Bruno Pinçon ([**1**]). Er wurde für die Studenten der École Supérieure d’Informatique et Application Lorraine geschrieben und an der RWTH Aachen von Agnes Mainka und Helmut Jarausch ins Deutsch übersetzt. Version 0.9999 (für Scilab 4.0 mit neuer Grafik) ist unter http://kiwi.math.uni-mannheim.de/~mfenn/Matlabuebung/PinconD.pdf abrufbar

Beide Schriften werden Sie als Nachschlagewerk und zum Erlernen des Umgangs mit Scilab verwenden. Im Unterricht behandeln wir vor allem Themen, die in dieser Form nicht in den beiden oben genannten Unterlagen zu finden sind.

1.2 Erstes Kennenlernen von Scilab

1.2.1 Wichtige Fenster beim Arbeiten mit Scilab

1.2.1.1 Arbeits- und Kommandofenster

Nach dem Aufruf von Scilab erscheint das Hauptfenster, von dem aus alle weiteren Fenster erreichbar sind und mit dem Befehle eingegeben und Berechnungen mit Matrizen und komplexen Zahlen durchgeführt werden können, ähnlich einem überdimensionierten Taschenrecher.

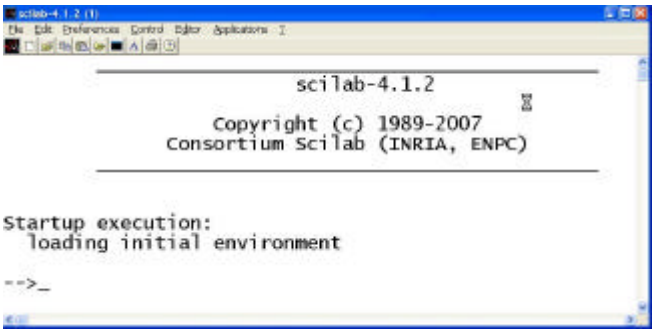


Abb. 1

1.2.1.2 Editor zum Erstellen von Programmen

Die im Arbeitsfenster eingebbaren Befehle können auch in eine Textdatei geschrieben werden (sog. Script) , die dann in Scilab geladen und ausgeführt werden kann. So kann man in Scilab programmieren. Diese Programmdateien haben die Endung .sce.
Der Editor kann über die Menüzeile durch Anklicken von „Editieren“ (Abb. 2) oder durch Eingabe von scipad() im Kommandofenster aufgerufen werden.

Abb. 2 Aufruf des scipad-Editors

Im sich öffnenden Editorfenster kann nun nach den Regeln von Sclilab programmiert, das Programm in Scilab geladen und gestartet oder mit dem Debugger schrittweise abgearbeitet werden.

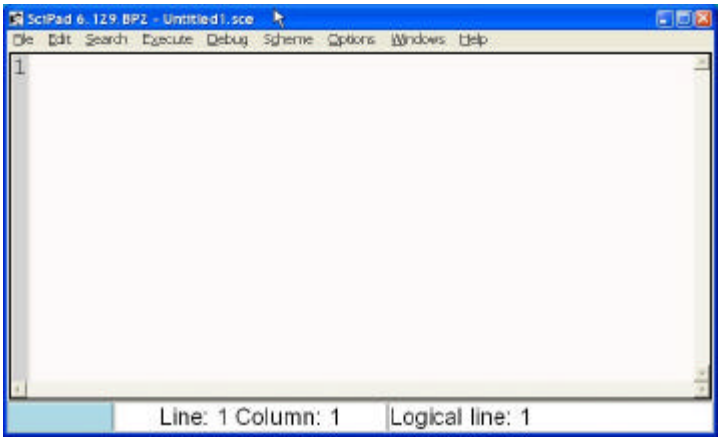


Abb. 3 Fenster des Editor scipad

Der Editor numeriert die Zeilen automatisch. In der Statuszeile unten am Bildrand (Abb. 3) sieht man die Angabe der aktuellen Zeile. Dabei ist ein Unterschied zwischen der Zeilennummer des Editors („Line“) und der Zeilennummer, wie sie der Compiler numeriert („Logical Line“). Letztere werden wir beim schrittweise Abarbeiten des Programms mit dem Debugger wieder benötigen. Während der Editor nämlich die Zeilen durchnumeriert, betrachtet der Compiler Befehle, die über mehrere Zeilen mit Hilfe der drei Punkte „...“ aufgeteilt werden, als eine einzige Programmzeile.

1.2.1.3 Hilfe zu Scilab

Im Kommandofenster kann man Hilfe zu beliebigen Befehlen aufrufen mit „help <Befehl>“. Dadurch öffnet sich das Hilfefenster mit einem alphabetischen Index, in dem man die gewünschte Funktion wählen kann.

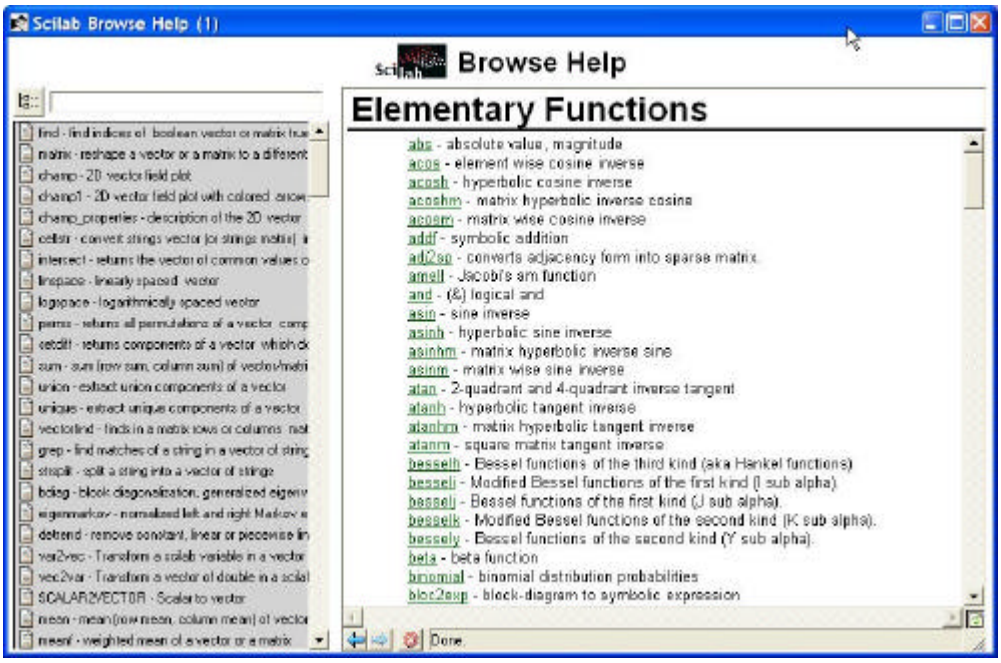


Abb. 4 Hilfefunktion

Für bestimmte Befehle werden sogar Demo’s angeboten, z.B. für den Befehl plot:

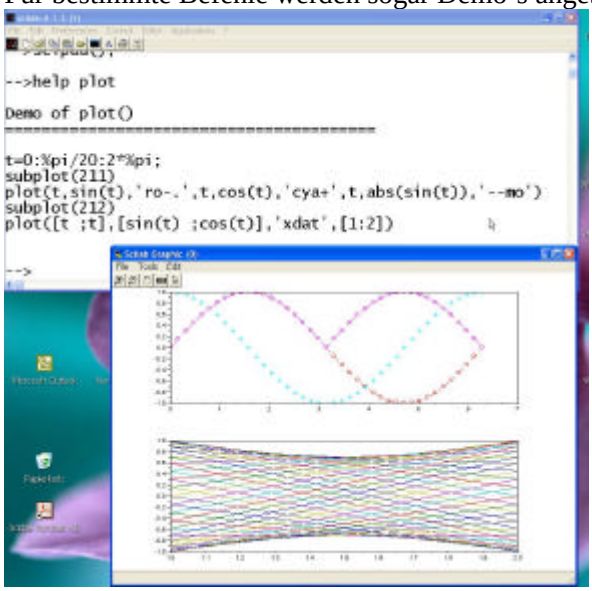


Abb. 5 Demo für den Befehl plot

1.2.2 Untersuchungen an einem existierenden Programms

Im folgenden werden Sie ein fertiges Programm in den scipad-Editor laden, in Scilab ablaufen lassen und anschließend mit dem Debugger Schritt für Schritt analysieren. Dabei werden Sie

- den Umgang mit Vektoren und Matrizen üben,
- kennen lernen, wie man mit Funktionen umgeht,
- sehen, wie man eine Funktion als Graph auf dem Bildschirm darstellt,
- mit dem Debugger ein Programm schrittweise abarbeiten und den Inhalt von Variablen beobachten.

1.2.2.1 Laden und Starten eines existierenden Programms

Rufen Sie aus dem Kommandofenster den Scipad-Editor auf und wählen Sie im Menü des Editors unter „file/open“ die Datei „bessel_gce_gcf_test.sce“ (Abb. 1).

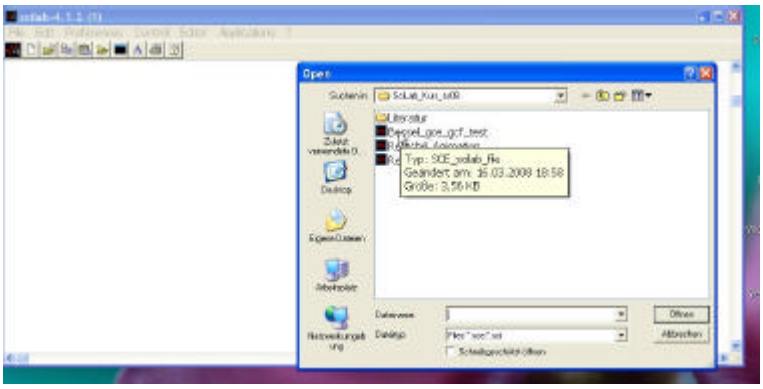


Abb. 6 Aufruf des zu untersuchenden Programms

Der Scipad-Editor öffnet die Datei, die Sie nun gleich in Scilab laden und ausführen werden (Abb. 7).

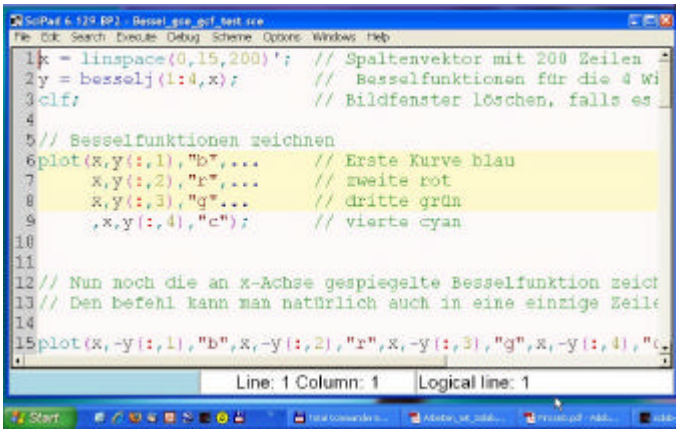


Abb. 7 Testprogramm für das Zeichnen einer Schar von Besselfunktionen

In dieses Programm wurden mehrere Halt-Befehle eingebaut, damit Sie die Wirkung einzelner Programmteile zunächst einzeln erleben können, bevor Sie das Programm mit dem Debugger analysieren.

Sobald das Programm an einem Haltepunkt stehen bleibt, müssen Sie ins Kommandofenster wechseln und die Return-Taste drücken.

Zum Laden und Starten des Programms wird im Menü „Execute/Load into Scilab“ gewählt.

Es öffnet sich nun zusätzlich zu, Kommando- und Zum Editorfenster ein Graphikfenster. In dem die Besselfunktionen mit verschiedenen Farben gezeichnet werden.

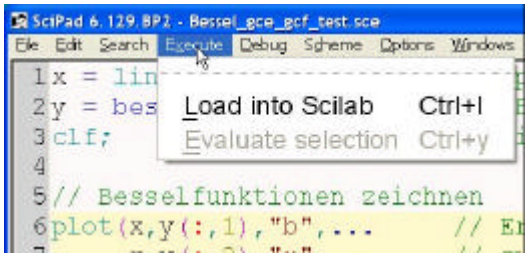


Abb. 8

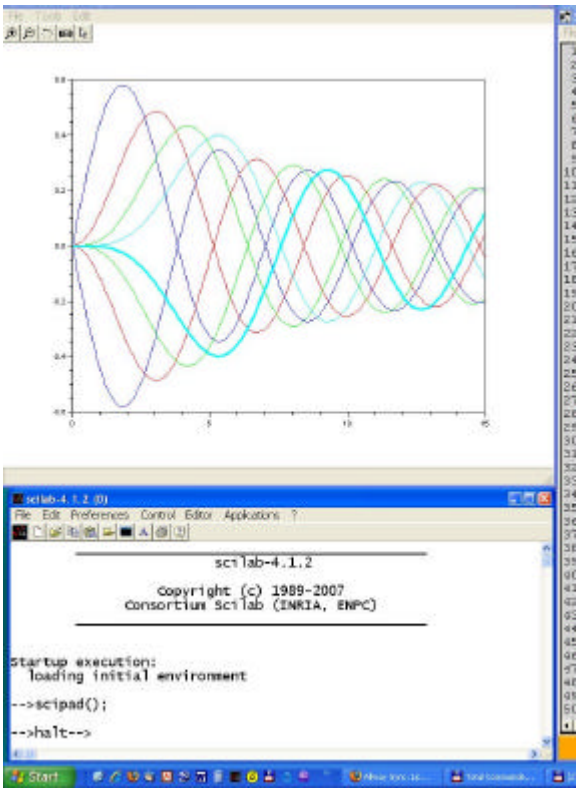


Abb. 9 Programm ist nach dem Zeichnen der Funktionenschar beim ersten Programmhalt angelangt.

obere Ecke geschrieben sowie der ganze Graph mit einem Titel versehen wird (Abb. 11).

.

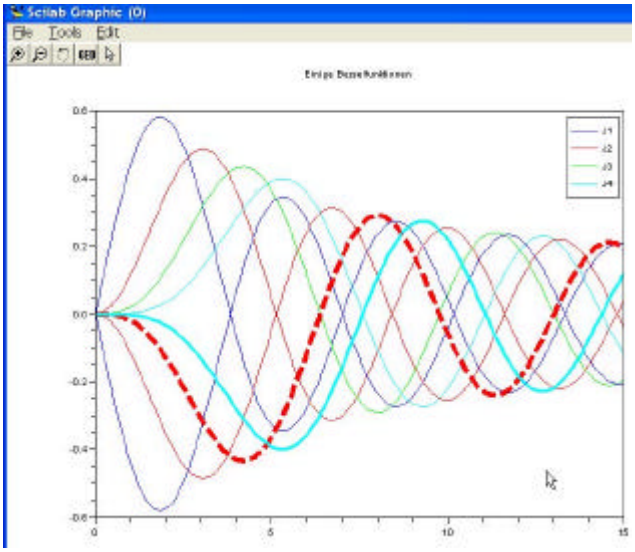


Abb. 11 Die Graphik erhält eine Legende und einen Titel

Im nächsten Schritt nach Return wird diejenige Kurve, die als zweitletzte gezeichnet wurde, mit anderer Farbe, dick und gestrichelt gezeichnet (Abb. 10).

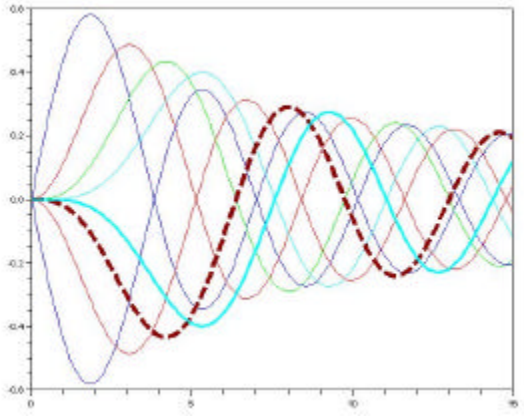


Abb. 10 Vorletzte Kurve wird geändert

Bei der selben Kurve wird nun zweimal die Farbe geändert, zunächst nach blau (Abb. 12) und anschließennd nach rot, bevor in zwei weiteren Schritten eine Legende erzeugt und in die rechte

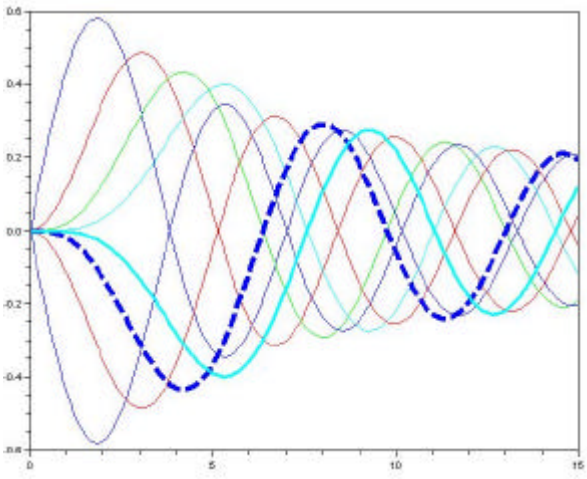


Abb. 12 Farbänderung derselben Kurve nach Blau

Noch einige weitere Änderungen werden gemacht, die im einzelnen bei der folgenden schrittweisen Abarbeitung mit dem Debugger im Programm durch ausführliche Kommentare erläutert werden.

1.2.2.2 Analyse des Programms und Arbeiten mit dem Debugger

Der Debugger wird in der Menüzeile des Scipad-Editors aufgerufen. Dazu muß zunächst das Programm im Editor vorbereitet werden mit „Configure execution“ (Abb. 13).

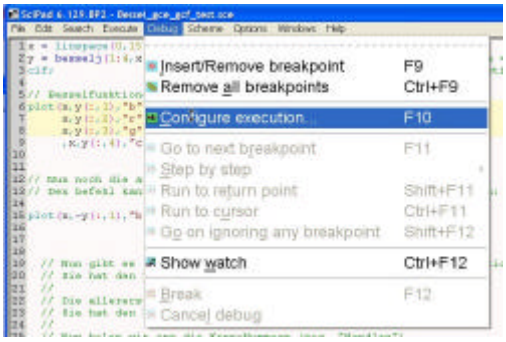


Abb. 13 Aufruf des Debuggers, Vorbereitung des Programms zum Debuggen

Im nachfolgende Fenster wird der Debugger endgültig gestartet (Abb. 14).

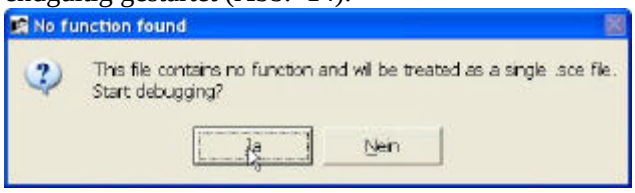


Abb. 14 Start des Debuggers

Der Debugger springt auf den ersten Befehl, gekennzeichnet durch die rote Markierung. Mit der Taste F8 kann nun das Programm Befehl für Befehl abgearbeitet werden (Abb. 15).

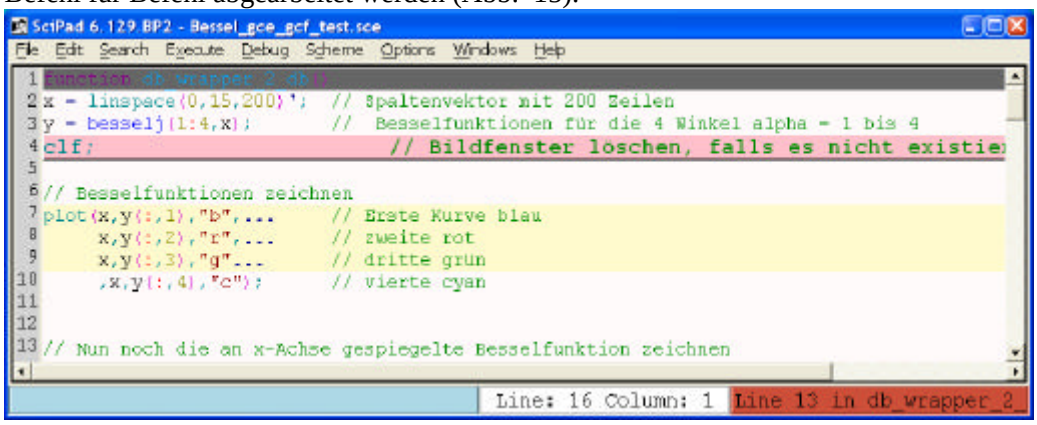


Abb. 15 Bei Betätigung von F8 wird nun ein neues Fenster für die Grafik aufgemacht.

Der Befehl clf erzeugt ein neues Fenster für die Grafik. Falls bereist ein Grafikfenster existiert, wird der Inhalt gelöscht.
Interessant wird die Statusanzeige am unteren Rand des Editors (Abb. 15) , wenn der Plotbefehl ausgeführt wird. Während die Anzeige der Editorzeile (links) dabei auf 16 springt, geht die Anzeige des Debuggers nur auf 13, da er die mit den drei Punkten „...“ zusammengehängten Zeilen als eine einzige interpretiert.

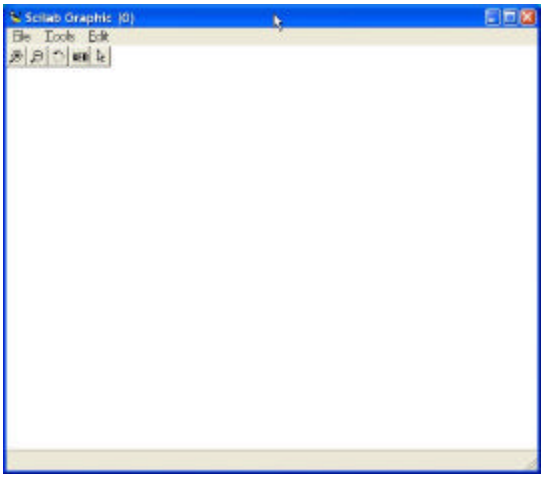


Abb. 16 ein Grafikfenster wird erzeugt

Lesen Sie im folgenden erst die Befehle und versuchen Sie diese zu verstehen, bevor Sie mit F8 die Durchführung veranlassen. Vergleichen Sie dann Ihre Erwartungen mit dem tatsächlichen Ergebnis. Studieren Sie insbesondere dabei die ausführlichen Kommentare im Programm.

Die Manipulation der Zeichnungsobjekte wird ausführlich im nächsten Kapitel besprochen. Hier nur soviel:
Die Grafik ist hierarchisch aufgebaut. In der obersten Ebene ist das (oder die) Bildfenster („figure“). Jedes Fenster kann als nachfolgende ebene mehrere Koordinatensysteme („axes“) enthalten. An unterster Ebene stehen als Einheiten („entities“) die Kurvenzüge und Zeichenketten, die teilweise zu Verbünden („compounds“) zusammengefaßt werden. Handles zu den einzelnen, gerade aktuell aktiven Elementen können mit den Funktionen gcf() („get current figure“), gca() („get current axes“) und gce() („get current entity“) angefordert werden, und erlauben damit Zugriff auf die Eigenschaften der gezeichneten Elemente.

Sobald Sie das Programm einmal schrittweise abgearbeitet haben, ändern Sie im Editor in der ersten Zeile die Größe des Vektors x von 200 Elementen auf 20, damit beim anschließenden Betrachten des unabhängigen Variablenvektors x und der Ergebnismatrix x im „Watch-Fenster“ kein Problem auftritt.

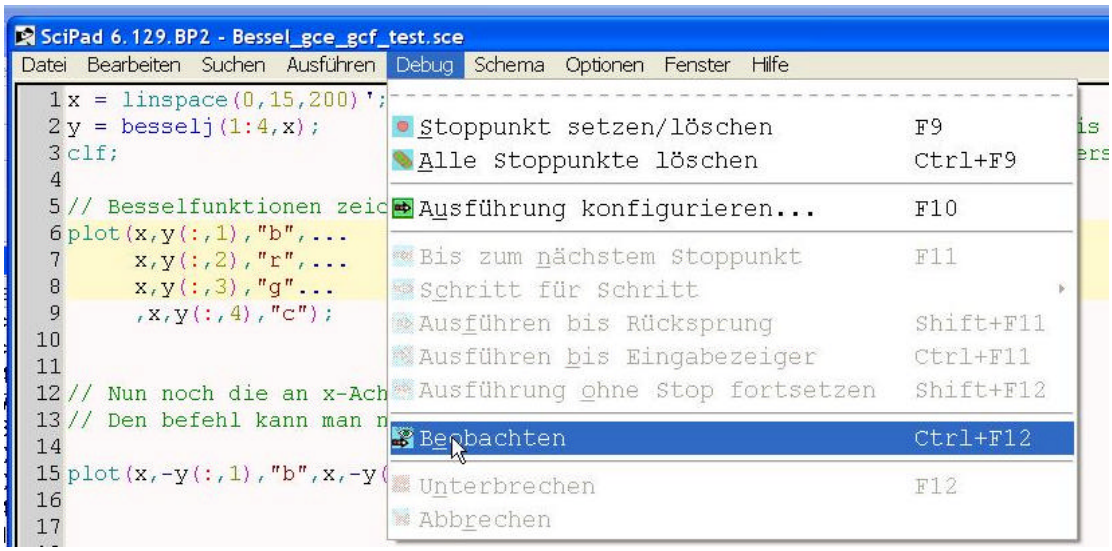


Abb. 17 Aufruf des Watch-Fensters zur Überwachung von Variablen

Zur Beobachtung des Spaltenvektors x mit 20 Zeilen und der 20 x 4-Ergebnismatrix y wird im Editormenü unter dem Menüpunkt „Debug“ der Unterpunkt „Beobachten“ aufgerufen (Abb. 17) und im dadurch sich öffnenden Watchfenster „Hinzufügen“ gewählt. Danach wird zunächst die Variable x eingegeben (Abb. 19), das Auswahlfeld markiert und mit OK die Eingabe bestätigt. In gleicher Weise wird auch die Matrixvariable y eingefügt. Im Watchfenster erscheint beim ersten Mal für den Inhalt der Variablen ein Fragezeichen, weil noch kein Wert zugeordnet war (Abb. 20).

Abb. 18 Hinzufügen zu beobachtender Variabler

Abb. 19 Eingabe der Variablen

Beim nachfolgenden schrittweise Abarbeiten des Programms ist nach der ersten Programmzeile der Spaltenvektor x belegt. Daß es sich um einen Spaltenvektor handelt, sieht man an den Strichpunkten nach jedem Wert(Abb. 22).

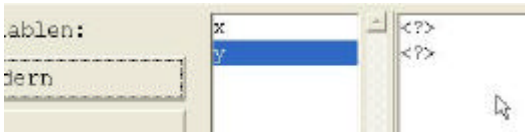


Abb. 20 Beide Variable sind noch nicht belegt

Nach Berechnung der Besselfunktionen für die 4 Werte von 1 bis 4 steht auch die Ergebnismatrix y im Watchfenster. Die Zeilenvektoren aus 4 Elementen stehen nur durch Zwischenräume getrennt im Fenster. Die nächste Zeile der Matrix beginnt jeweils mit dem Strichpunkt (Abb. 21).

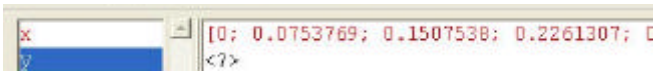


Abb. 22 Spaltenvektor x, Elemente durch Strichpunkt getrennt

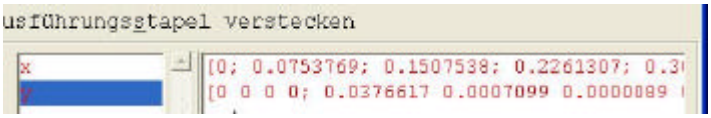


Abb. 21 4x20-Ergebnismatrix y

1.3 Zusammenfassung wichtiger Lernziele

Aus dem gesamten Kapitel 2 sollen Sie folgendes dauerhaft mitnehmen:

1. Einblick in die Lehrbücher „Arbeiten mit Scilab und Scicos“ von Jean-Marie Zogg und „Eine Einführung in Scilab“ von Bruno Pinçon.
2. Einblick in das Arbeiten mit den wichtigsten Fenster von Scilab, dem Kommandofenster, dem Editorfenster, dem Debugfenster und den Grafikfenstern.
3. Umgang mit dem Debugger zum Austesten von Programmen und Überwachen von Variablen
4. Einblick in den Umgang und die Darstellung von Matrizen und Vektoren, insbesondere der Darstellung von Zeilen- und Spaltenvektoren, der Erzeugung eine s Vektors mit einer vorgegebenen Anzahl äquidistanter Elemente, Entnahme eines Zeilen- oder Spaltenvektors aus einer Matrix
5. Einblick in die Darstellung von Funktionen als 2D-Graph
6. Einblick in die Struktur der Grafiken in Scilab und Modifikation seiner gezeichneten Elemente

Im folgenden Kapitel werden Sie

- die Struktur der Graphik in Scilab und die 3D-Darstellung von Kurven genauer kennen lernen
- mit komplexen Variablen rechnen lernen
- die Entstehung einer 3D-Kurve im Raum als Film darstellen.

Dazu werden wir noch einmal die Struktur der Graphik mit den Bessel-Funktionen näher untersuchen und uns anschließend die Aufgabe stellen, die Funktion

$u = \cos(2 \cdot \omega \cdot t) = \frac{1}{2} \cdot \left(e^{+j\omega t} + e^{-j\omega t}\right)$ in einem dreidimensionalen Koordinatensystem mit der Zeit t als x-Koordinate, dem Realteil $\mathrm{Re}\{u\}$der Funktion $u(t)$ als y-Achse und dem Imaginärteil $\mathrm{Im}\{u\}$ als z-Achse animiert auf dem Bildschirm entstehen zu lassen. Bei der Animation sollen als Zwischenergebnisse Abb. 23, Abb. 24, Abb. 25 und als Endzustand Abb. 26 sichtbar sein.

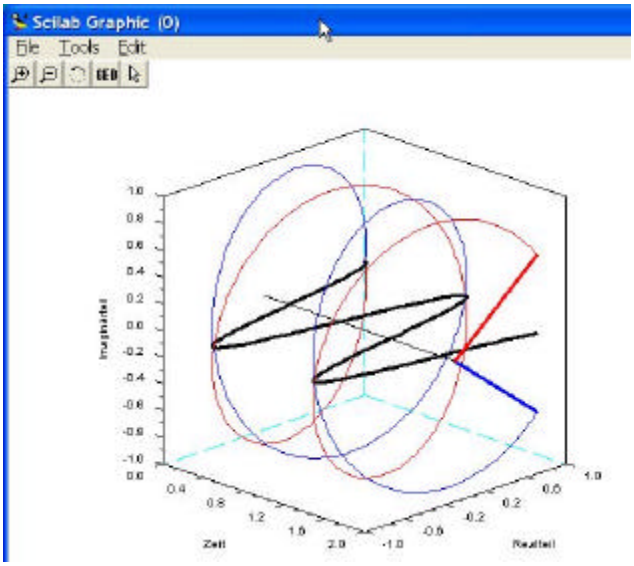


Abb. 23 u(t) in 3D-Darstellung

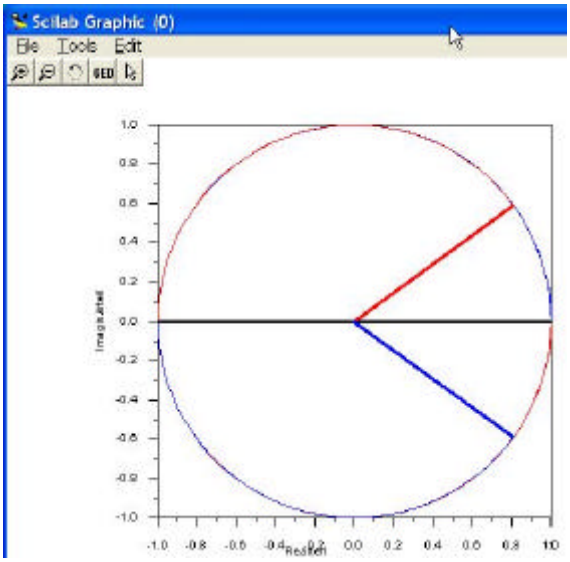


Abb. 24 u(t)-Ansicht von vorne direkt auf die Zeitachse

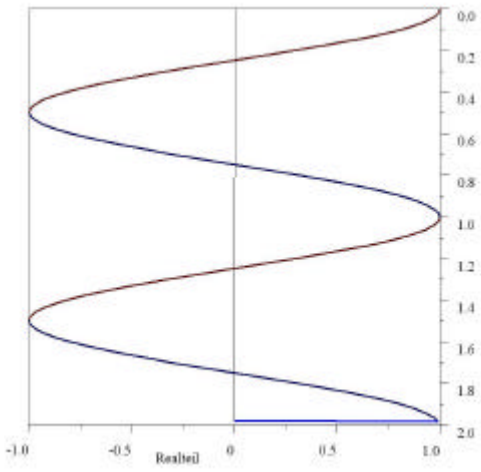


Abb. 26 u(t)-Ansicht von oben auf die imaginäre Achse und die Spiralen $e^{+j? t}$, $e^{-j? t}$ sowie den Kosinus

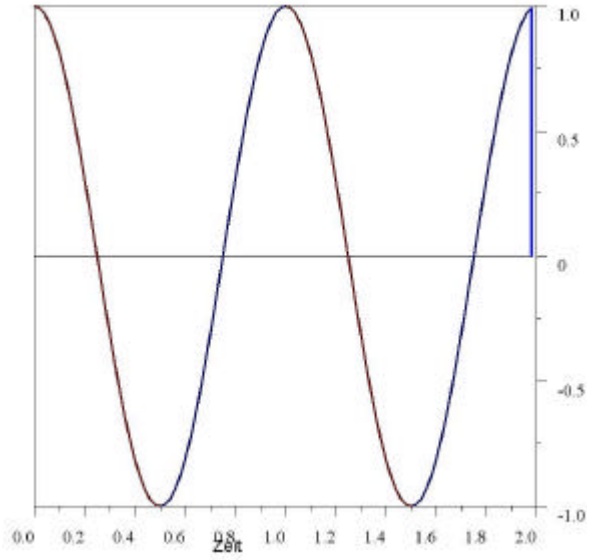


Abb. 25 u(t)-Ansicht von vorne auf die reelle Achse

2 Grafik in Scilab

Die Graphik in Scicos ist hierarchisch aufgebaut. An oberster Stelle ist das Bildfenster („figure“), für das man ein Handle mit der Funktion gcf() anfordern kann. Ein Handle ist so etwas wie die Hausnummer eines mehrstöckigen Hauses, über die man dann alle Ebenen und Zimmer des Hauses erreichen und verändern kann.

An nächstniedrigerer Rangordnung sind die Koordinatensysteme, die man in ein Bild zeichnen kann. Zu den Koordinatensystemen zählen 2D-, 3D-Systeme ebenso wie eine Legende, in der die Kurven gekennzeichnet sind.
Das Handle des gerade aktiven Koordinatensystems erhält man mit der Funktion gca(). Die Abkürzung gca steht für „get current axes“. Sie entsprechen im Bild des mehrstöckigen Hauses den Geschoßen, auf denen sich jeweils mehrere abgeschlossene Wohnungen („compounds“ also Verbünden von Zimmern) befinden können. Ebenso aber ist es möglich, daß es auf einem Stockwerk (= Koordinatensystem „axes“) nur Zimmer ohne Wohnungstüre gibt.

Die „Wohnungen“ (compounds) und „Zimmer“ stellen die unterste Hierarchieebene dar. Es sind Verbünde einzelner Kurven oder Zeichenketten zur Beschriftung, aber auch (falls es keine „Wohnungstüre“ gibt) diese Zeichenelemente selber. Ob es sich bei dieser Einheit („entity“) um eine „Wohnung“ (compound) handelt oder nur ein „einzelnes Zimmer“ (z.B. Polyline, String...), hängt vom vorausgegangenen Plotbefehl ab. Die Zeichnung der 4 Besselkurven mit einem einzigen Plotbefehl führt jedenfalls auf einen Verbund („Wohnung“) mit 4 Polylinien („Zimmer“).

Das aktuelle Handle der letzten gezeichneten Einheit wird mit gce() angefordert. Die Abkürzung gce bedeutet „get current entity“.

2.1 Grafikstruktur des Testprogramms Bessel_gce_gcf_test.sce

Bis zum ersten Halt bei Zeile 63 des Programms hat das Diagramm die Gestalt in Abb. 27 erreicht. Weder Legende noch Gitternetz oder Titel sind vorhanden. Ein Mausklick auf „GED“ öffnet den Editor für die Zeichenelemente, mit dem man nachträglich die Eigenschaften der Zeichnung wie Farben, Strichart und Strichdicke usw. verändern kann. Aus dem Editor kann man auch die Struktur der Zeichnung ablesen (Abb. 30).

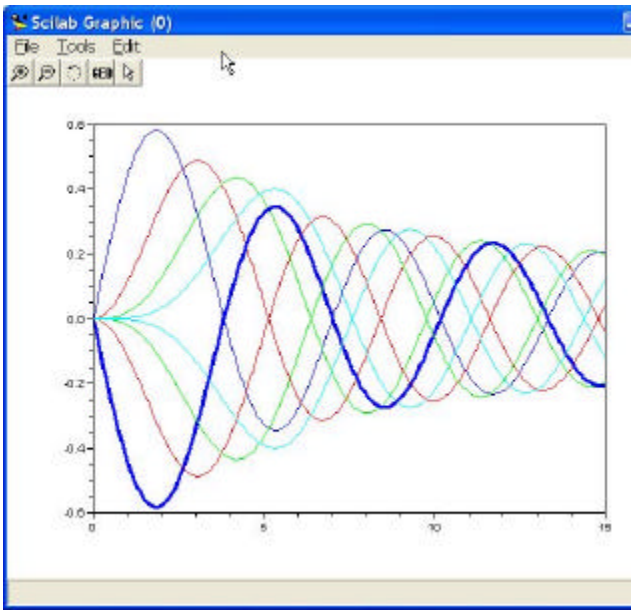


Abb. 27 Grafik bis zum Halt in Zeile 63

Unter der obersten Ebene Figure(1) ist zunächst nur als einziges der Children ein System Axes(1). Es enthält als Kinder die zwei Zeichenverbünde Compound(1) (gespiegelte, später gezeichnete Besselkurven) und Compound(1) (die zu Beginn gezeichnete Besselkurven).

Kinder der beiden Compounds sind jeweils 4 Polylinien, die insgesamt 8 Kurven des Diagramms.

Läßt man das Programm bis zum Schluß laufen, dann erscheint ein weiteres Koordinatensystem, das nun den Index 1 hat, während das alte mit den Besselkurven auf den zweiten Platz gerutscht ist.

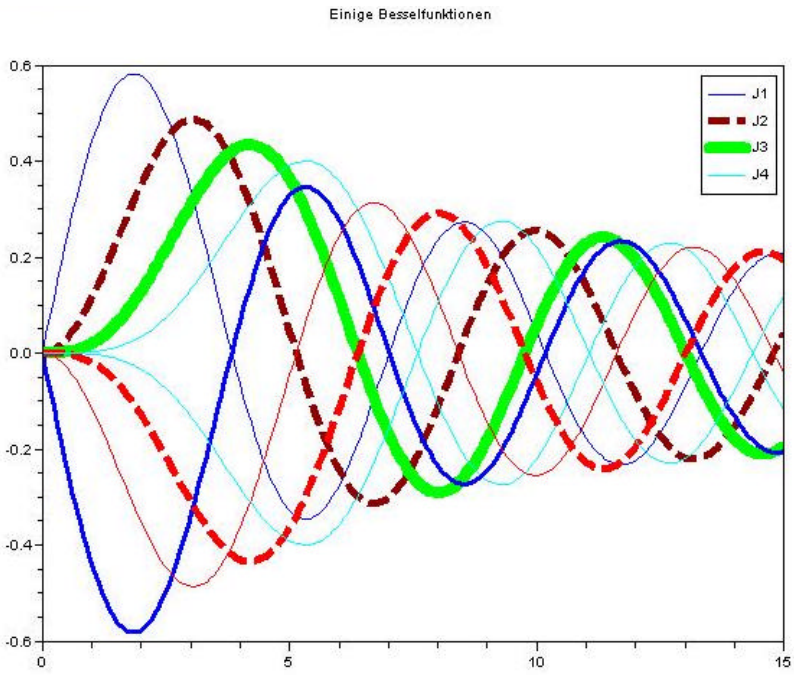


Abb. 29 Vollständiger Graph mit allen Elementen

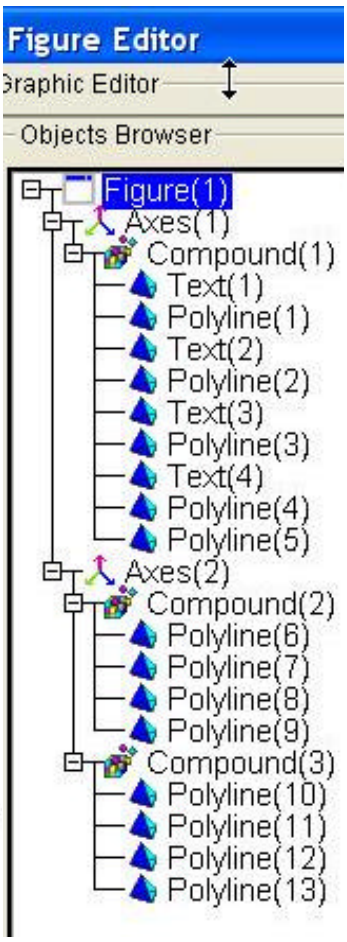


Abb.28 Struktur der Graphik mit allen Elementen

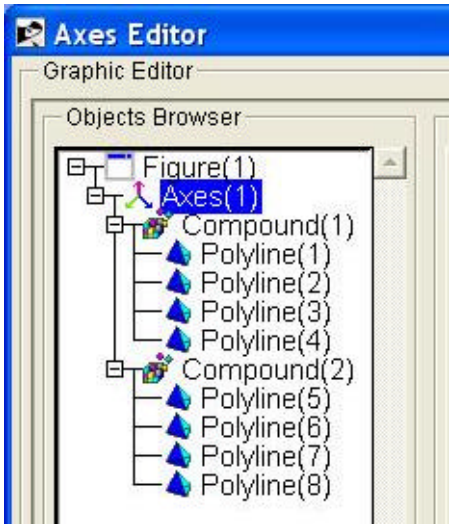


Abb. 30 Stuktur der Grafik ohne Legende (Programm bis Zeile 63)

Aufgabe:
Versuchen Sie, nach Ende des Programms einige Parameter der Grafik mit dem Grafik-Editor zu ändern

2.2 3D-Kurvenzüge, animierte Darstellung und Modifikation der Darstellung

Neben dem Umgang mit der dreidimensionalen Plotausgabe von Kurvenzügen lernen Sie im folgenden Kapitel

- Darstellung und Umgang mit komplexen Zahlen in Scilab
- Flexible Definition von Wertetabellen
- Zuweisung von Farben an einzelne Kurvenzüge mit Hilfe der Scilab-Funktion list
- Definition und Zeichnen einzelner Linien („Zeiger“) in vorgegebener Farbe
- Darstellung der Entstehung von Kurvenzügen auf dem Bildschirm als bewegtes Bild
- Rotation des Koordinatensiystems als bewegtes Bild
- Nachträgliche Änderung der Beschriftung des Bildes
- Nachträgliche Änderung der Achsteilung und Achsbeschriftung mit Hilfe der Scilab-Funktion tlist

2.2.1 Darstellung von Kurvenzügen im Raum

Scilab stellt zur Darstellung von Raumkurven den Plotbefehl param3d1 zur Verfügung. Wir werden uns in folgenden Schritten mit dem Befehl vertraut machen:

1. Verwendung des Befehls und Bedeutung der Parameter

2. Erzeugung des Verlaufs der beiden Funktionen $e^{+j\omega t}$ als blauer und $e^{-j\omega t}$ als roten Kurvenzug sowie der Funktion $\cos\omega t = \frac{1}{2}\cdot\left(e^{+j\omega t} + e^{-j\omega t}\right) = \mathrm{Re}\left\{e^{j\omega t}\right\}$ als schwarze Linie

3. Animierte Darstellung der rotierenden Zeiger

Dabei werden wir auch einen tieferen Einblick in den Aufbau und die Modifikation von Graphiken gewinnen und weitere Scilab-Befehle, etwa zur Behandlung von Listen kennen lernen.

2.2.1.1 Der Befehl param3d1

Das Zeichnen von 3D-Kurven wird aufgerufen mit
param3d1(x,y,z, w_Dreh, w_Neig, Achsbeschriftung, [3D-Darstellungsart, Umrahmung])

Für einen räumlichen Kurvenzug sind die drei Spaltenvektoren x, y und z mit je n Elementen nötig. Sie stellen die Wertetabelle dar. Will man mehr als einen Kurvenzug mit einem einzigen Befehl darstellen, so kann man die drei Vektoren zu Matrizen mit je k Spalten erweitern. Damit zeichnet der Befehl k Kurven.

Für die Darstellung ist darüber hinaus noch ein Standort für die Betrachtung des 3D-Objekts zu wählen. Den Beobachtungspunkt charakterisieren der Drehwinkel w_Dreh und der Neigungswinkel w_Neig w_Dreh wird gegenüber der x- und w_Neig gegenüber der z-Achse gemessen (Abb. 31).

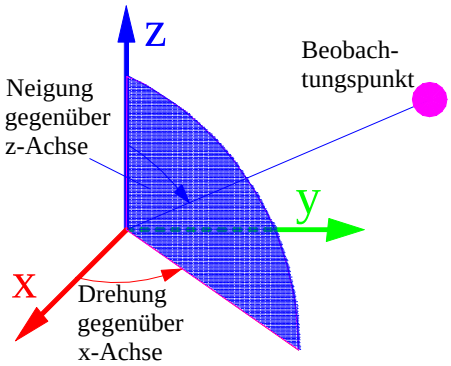


Abb. 31 Definition des Beobachtungspunktes

Als weiterer Parameter kann man noch die Achsbeschriftungen der drei Achsen als String angeben. Die drei Bezeichnungen werden in einen einzigen String geschrieben und mit dem Symbol @ voneinander getrennt, z.B. 'x@y@z'. Will man die Achsenbeschriftung später mit einem eigenen Befehl durchführen, so bleibt die Position zwischen den betreffenden Kommata im Befehl param3d1 einfach leer.

Die Bedeutung des Vektors aus den beiden Komponenten „3D-Darstellung" und „Umrahmung" (je eine Zahl unter 10) ist am besten in der Online-Hilfe nachzulesen. Für die Zeichnung der Spirale von $e^{j\omega t}$ lautet ein typischer Befehl:
param3d1(x,y,z,-45,70,'Zeit@Realteil@Imaginärteil',[6,4])

2.2.1.2 Erzeugung der Wertetabellen der Zeitfunktionen

Die Funktion $\cos\omega t = \frac{1}{2}\cdot\left(e^{+j\omega t} + e^{-j\omega t}\right) = \mathrm{Re}\left\{e^{j\omega t}\right\}$ soll als schwarzer Kurvenzug zusammen mit ihren beiden Unterfunktionen $e^{+j\omega t}$ als blauer und $e^{-j\omega t}$ als roten Kurvenzug dargestellt werden vom Zeitpunkt t = 0 bis t = t_end = 2.0 sec dargestellt werden. Der Abstand delta_t der zu berechnenden Stützpunkte betrage zunächst 0.2sec. Wie der Endpunkt der Berechnungszeit soll auch dieser Abstand einstellbar sein.

Aufgabe:
*Erzeugen Sie einen Zeilenvektor **t** von 0 bis t_end, wobei die Elemente des Vektors delta_t Abstand voneinander haben. ([2], S.42). Tragen Sie die nötigen Befehle im Scipad-Editor ein und lassen Sie das Programm laufen.*
Geben Sie anschließend im Kommandofenster den Buchstaben t ein und drücken Sie <Return>.

Mit Hilfe dieses Vektors soll nun ein Vektor **x** mit derselben Anzahl an Komponenten erzeugt werden, der die komplexe Zahl $e^{+j\omega t}$ für alle Stützwerte enthält. Die Schwingfrequenz des Kosinus soll 1Hz betragen. Als Symbol für die Kreisfrequenz wählen wir w, da es dem griechischen omega sehr ähnlich sieht. Die Kreiszahl Pi wird in Scilab durch %pi dargestellt.
Die Exponentialfunktion in Scilab lautet exp(Argument).

Mit den Scilab-Funktionen real(Argument) und imag(Argument) lassen sich der Real- und der Imaginärteil aus der komplexen Zahl „Argument" entnehmen.

Aufgabe:
*Erzeugen Sie den Vektor x der Stützwerte der Funktion $e^{+j\omega t}$ und daraus die Vektoren des Realteils **re** und Imaginärteils **im**. Überprüfen Sie wieder die Ergebnisse **x**, **re** und **im** durch Eingabe dieser Variablen im Kommandofenster.*

Nun sind die drei Vektoren t, re und im erzeugt, die als Punkte im 3D-Koordinatensystem eingetragen werden können. Aus **re** und **im** lassen sich auch der Kurvenzug der Funktion $e^{-j\omega t}$ und $\cos\omega t$ gewinnen. Erstere unterscheidet sich nur durch das Vorzeichen des Vektors im und der Kosinus ist der Realteil **re** allein mit einem Imaginärteil, der für alle Vektorelemente Null ist.

2.2.1.3 Erzeugung der Wertetabellen für die Darstellung im xyz-Koordinatensystem

Aus den Wertetabellen der drei Zeitfunktionen müssen nun die Wertetabellen für $e^{+j\omega t}$, $e^{-j\omega t}$ und $\cos \omega t$ erzeugt werden. Zusätzlich soll im 3D-Koordinatensystem bei y = x = 0 eine Zeitachse gezeichnet werden, da die Achsen des Koordinatensystems aus Gründen der Übersichtlichkeit außerhalb der Graphen dargestellt werden. Es sind also folgende Wertetabellen aufzustellen:

Zeitachse:	**[t, 0, 0]**
$e^{+j\omega t}$	**[t, re, im]**
$e^{-j\omega t}$	**[t, re, -im]**
$\cos \omega t$	**[t, re, 0]**

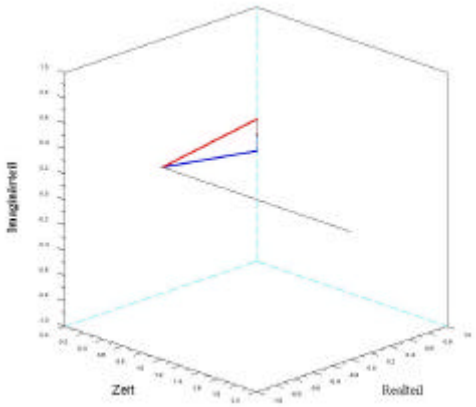


Abb. 32

Die Funktion param3d1 zeichnet dabei etwa bei $e^{+j\omega t}$ Polygonstücke vom Punkt (t_i, re_i, im_i) zum nächsten Punkt $(t_{i+1}, re_{i+1}, im_{i+1})$.
Da bei der animierten Darstellung zum aktuellen Zeitpunkt t_i auch noch die beiden Zeiger von der Zeitachse zum Punkt $e^{+j\omega t_i}$ und zum Punkt $e^{-j\omega t_i}$ gezeichnet werden sollen, sind noch die beiden Vektoren $([t_i, t_i],[0, re_i],[0, im_i])$ und $([t_i, t_i],[0, re_i],[0, -im_i])$ nötig. Die ersten Komponenten beinhalten jeweils den Anfangspunkt (x,y,z) auf der t-Achse, die zweiten den Endpunkt auf der jeweiligen Spirale $e^{+j\omega t_i}$ bzw. $e^{-j\omega t_i}$ (Abb. 32).

Um nicht Gefahr zu laufen, bei der Programmierung die Originaldatensätze versehentlich zu verändern, werden für die Darstellung die Daten in andere Vektoren kopiert. Die Gefahr einer Veränderung besteht insbesondere bei den Berechnungen für die bewegte Darstellung. Folgende Arbeitsvektoren werden definiert:

t	→	**x0**	für gesamte Zeitachse
re	→	**y0**	Realteilvektor von $e^{+j\omega t_i}$
im	→	**z0**	Imaginärteilvektor $e^{+j\omega t_i}$

t	→	**t_**	für bis t_end wachsende Zeit
re	→	**y0_**	für bis t_end sich ändernden Realteil von $e^{+j\omega t_i}$ und $e^{-j\omega t_i}$
		y2_	für bis t_end sich ändernden Realteil von $\cos \omega t$
im	→	**z0_**	für bis t_end sich ändernden Imaginärteil von $e^{+j\omega t_i}$
		z1_	für bis t_end sich ändernden Imaginärteil von $e^{-j\omega t_i}$
		z2_	für bis t_end unveränderten Imaginärteil = 0 von $\cos \omega t$

2.2.1.4 Farbzuordnung für die einzelnen Kurven – die Funktion list

In Scilab lassen sich Farben in vielfältigster Weise selber zusammenstellen, indem man die Anteile blau gelb und rot separat für sich durch einen Wert zwischen 0 und 255 definiert. Dies ist allerdings nur bei der Darstellung von Flächen sinnvoll. Für die Darstellung von Kurven reicht es, einzelne Farben über den Farbindex anzusprechen. Für die meisten Anwendungen reicht folgende Zuordnung (**Fehler! Verweisquelle konnte nicht gefunden werden.**] ,S. 74):

Farbnr.		Farbnr.		Farbnr.	
1	schwarz	5	rot	23	violett
2	blau	6	lila	26	kastanienbraun
3	hellgrün	13	dunkelgrün	29	rosa
4	hellblau	16	türkis	32	gelborange

Die Zuordnung einer Farbe zu einem Kurvenzug erfolgt dadurch, daß man **jedem** Wert des z-Vektors mit Hilfe der Funktion list die gewünschte Farbnummer zuordnet. Dazu erzeugt man einen Vektor mit derselben Dimension, wie der z-Vektor und füllt ihn mit der gewünschten Farbnummer.
Eine Möglichkeit, einen Vektor mit der gleichen Dimension zu erzeugen, ist, die Dimension von **z** mit der Funktion size zu ermitteln und einen Farbvektor mit dieser Dimension zu erzeugen, ihn mit Nullen zu füllen und auf jedes Element die Farbnummer zu addieren:

```
[zeilenzahl, spaltenzahl] = size(im);          // liefert einen Vektor mit den 2 Komponenten
                                               //„Zeilenzahl von im“ und „Spaltenzahl von im“
FarbeBlau = zeros(zeilenzahl,Spaltenzahl) + 2; // Hier wird ein Nullvektor mit dieser
                                               // Dimension erzeugt und auf jede Komponente
                                               // die Zahl 2 aufaddiert
```

Mit diesem Vektor läßt sich die blaue Spirale $e^{+j\omega t}$ durch param3d1 erzeugen:

```
param3d1(t, re, list(im, FarbeBlau),…          // Koordinatenvektoren der Spirale mit Farbe
                                               // blau
        Winkel_Dreh, Winkel_Neig,…             // 3D-Beobachtungspunkt
          ,…                                   // zunächst keine Achsbeschriftung wie z.B.
                                               // „Realteil“
        [6,5]);                                // 3D-Darstellungsart und Achsbegrenzungen
```

2.2.1.5 Animierte Darstellung der 3D-Funktionsgraphen – direkter Zugriff auf die Datenvektoren der Kurvenzüge

Die Kurvenzüge sind vom Typ polyline (siehe help polyline) und besitzen ein Datenfeld, das als Komponente .data erreichbar ist. Es besteht aus 3 Zeilenvektoren. Sie beinhalten die Komponentenvektoren x, y und z des Kurvenzuges.

Je nach Organisation der Grafik gehört die Polylinie unmittelbar zu den Kindern der Achsen oder zur nächsten „Generation", den Kindern eines Verbundes. Die Polylinien des Beispiels „Besselfunktionen" in 2.1 sind Kinder eines von den Achsen abstammenden Verbunds (compound), während bei der vorliegenden Aufgabe die Polylinien direkte Abkommen der Achsen sind. Ist hp das Handle einer Polylinie (z.B. hp = ha.children(1) für die letzte gezeichnete Polylinie des Koordinatensystems mit dem Handle ha), so kann man allen Koordinaten der Matrix mit den 3 Zeilen und den n Spalten (n = Anzahl der Stützpunkte) in einem Schritt belegen ([2], Seite 52) durch

hp.data(:,:) = (t_, y0_, z0_) ;

Mit dieser Zugriffsart kann man die Spiralen, ausgehend vom Startpunkt (0,1,1), wachsen lassen bis zum Endpunkt bei (t_end, $\mathrm{Re}\{e^{j\omega t\_end}\}$, $\mathrm{Im}\{e^{j\omega t\_end}\}$) Dazu belegt man zunächst die gesamte Matrix mit 3 Zeilen und n Spalten mit dem ersten Polygonwert: $\begin{matrix} 0 & \dots & 0 \\ 1 & \dots & 1 \\ 1 & \dots & 1 \end{matrix}$.

In einer Schleife werden dann nacheinander die 2. bis n-te, dann die dritte bis n-te, die vierte bis n-te, die i-te bis n-te Spalte usw. mit dem aktuellen Vektor ($\left(t\_i, \mathrm{Re}\{e^{j\omega \cdot i \cdot deltat\_t}\}, \mathrm{Im}\{e^{j\omega \cdot i \cdot deltat\_t}\}\right)$ belegt.

Damit wächst die blaue Spirale $e^{j\omega t}$ von t = 0 bis t = t_end.

Störend ist nun noch das Flackern beim Bildaufbau. Dies läßt sich unterbinden, wenn das Bild im Hintergrund vorab in einem Puffer aufgebaut und dann mit einem einzigen Schritt angezeigt wird. Dazu stellt Scilab den Befehl pixmap und show_pixmap zur Verfügung. pixmap wir mit Hilfe des Handles des Bildes (figure, erreichbar durch hf = gcf();) aktiviert:

hf.pixmap = „on";

Jedesmal wenn ein neues Stück der Polygone neu belegt ist, holt die Anweisung

show_pixmap;

den im Hintergrund vorbereiteten 3D-Graph an der Oberfläche.

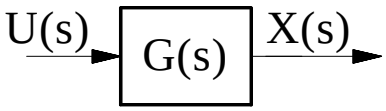
Aufgabe:
Studieren Sie gründlich das Programm Rettich4_Animation5.sce und arbeiten Sie es mit dem Debugger schrittweise ab.

3 Lineare Systeme in der Regelungstechnik

3.1 Polynome und Nullstellen von Polynomen

3.1.1 Produkt- und Summendarstellung von Polynomen

Der Übertragungsfunktion $G(s) = \frac{X(s)}{U(s)}$ eines linearen Systems mit dem Eingangssignal U(s) und dem Ausgangssignal X(s) kommt eine zentrale Bedeutung bei der Beschreibung des dynamischen Verhaltens zu.



Sie läßt sich darstellen als Quotient eines Zählerpolynoms $Z(s)$ und eines Nennerpolynoms $N(s)$.
Ein souveräner Umgang mit Polynomen ist also äußerst wichtig bei der Untersuchung dynamischer Systeme.
Polynome, z.B. das Nennerpolynom, lassen sich in Produktform durch ihre Wurzeln ebenso darstellen, wie als Summe von Potenzen der Variablen, die mit Koeffizienten gewichtet sind:

$$N(s) = K \cdot \prod_{1}^{n} (s - s_i) \qquad \text{bzw.} \qquad N(s) = \sum_{0}^{n} a_i \cdot s^i$$

Beispiel: Ein Polynom mit den Nullstellen 1 und 2 (man sagt auch „Wurzeln“ dazu) hat mit K = 1 die Form $N(s) = (s - 1)\cdot(s - 2) = s^2 - 3s + 2$
Wählt man für $K = 0.5$, dann hat das Polynom in Summenform einen konstanten Summanden mit dem Wert 1: $N(s) = 2\cdot(s - 1)\cdot(s - 2) = 0.5s^2 - 1.5s + 1$

.

Auch in Scilab kann man Polynome sowohl durch ihre Wurzeln in Produktform als auch durch ihre Koeffizienten in Summenform definieren:

N = poly(a,’s’,flag)

Mit flag wird bestimmt, ob a als Koeffizienten oder als Wurzeln behandelt wird. Flag ist ein String und kann die beiden Werte “coeff“ oder “roots“ annehmen. Auch die Abkürzungen “c“ und “r“ werden akzeptiert. Läßt man das flag weg, so wird die Darstellung als Produkt, also “roots“, angenommen.

Der String ,’s’ ist das Symbol für die Variable. Würde man ’x’ wählen, so würde man ein Polynom in x erhalten.

a enthält den Wurzel- bzw. Koeffizientenvektor des Polynoms. Wählt man die Summendarstellung mit Koeffizienten, dann besitzt a eine Komponente mehr, als in der Produktdarstellung mit Wurzeln. Bei der Darstellung mit Hilfe der Wurzeln wird also der konstante Multiplikator zu K = 1 angenommen.

Ziel der folgenden Aufgabe ist es, Sie Schritt für Schritt in den Umgang mit Polynomen einzuführen. Vor allem soll ein Weg vorbereitet werden (siehe 3.1.3), damit wir Polynome in der uns gewohnten Art, z.B. $n = s^2 - 2s + 3$, eingeben können , statt mit der etwas spröden Funktion poly.

Aufgabe:

- *Wählen Sie für a den Vektor [1, 2] und weisen Sie der Variablen N1 das Polynom in Koeffizientendarstellung und anschließend der Variablen N2 in Wurzeldarstellung zu. Lassen Sie bei der Eingabe den abschließende Strichpunkt weg, damit das Ergebnis angezeigt wird.*
  Die Ergebnisse N1 und N2 werden wir später wieder verwenden.

- *Erzeugen Sie das Ergebnis der Produktdarstellung auch mit Hilfe der Koeeffizientendarstellung und weisen Sie dieses Ergebnis der Variablen N3 zu.*

- *Erzeugen Sie ein Polynom N4 mit den Koeffizienten [1, -1.5, 0.5].*

- *Wählen Sie als einzige Nullstelle a=0 und stellen Sie das Polynom in Produktdarstellung dar. Dieses Ergebnis nennen Sie s. Wir werden es später wieder brauchen, wenn wir Polynome so darstellen wollen, wie wir sie üblicherweise schreiben, z.B.* $s^2 - 3s + 2$

3.1.2 Wurzeln eines Polynoms

Die Ermittlung der Wurzeln eines Polynoms ist die komplementäre Operation zur Produktdarstellung des Polynoms aus seinen Wurzeln. Wir werden später den Verlauf der Wurzeln der charakteristischen Gleichung eines dynamischen Systems graphisch darstellen (sog. Wurzelortskurve WOK), mit der wir das dynamische Verhalten und die Stabilität beurteilen.

Die Berechnung der Wurzeln erfolgt in Scilab mit der Funktion

w = roots(Polynom)

Aufgabe :
In der Aufgabe unter 3.1.1 haben Sie die Polynome N1 bis N4 berechnet. Ermitteln Sie nun umgekehrt die Nullstellen dieser Polynome.
Was fällt Ihnen insbesondere auf, wenn Sie die Wurzeln von N3 und N4 betrachten und mit den Wurzeln von N2 vergleichen. Was folgt daraus bei Scilab für die Flexibilität der Darstellung von Polynomen in Produkt- und Koeffizientenform?

3.1.3 Operationen mit Polynomen – Darstellung mit der Variablen s

Mit Polynomen lassen sich dieselben Operationen durchführen, wie mit Skalaren, Vektoren und Matrizen. Man kann sie addieren, subtrahieren, multiplizieren und dividieren. Letzteres ist gerade für die Bildung der Übertragungsfunktion G(s) von Bedeutung.
Dies sollen Sie nun in der folgenden Aufgabe verifizieren und einüben.

Aufgabe:
Bilden Sie folgende Verknüpfungen mit den Polynomen N1 bis N3:
G1 = N1 + N2
G2 = N1 – 2 N2^2*
*G3 = N1 * N2*
G4 = N1/N2
G5 = N2/N1

In 3.1.1 haben wir ein **Polynom s** erzeugt, dessen Nullstelle bei Null lag und dessen ***Variable*** den ***Namen*** s besitzt. Auch mit diesem Polynom s können wir die obigen Operationen durchführen.

Aufgabe:
Führen Sie mit dem Polynom s folgende Operationen durch:
*z = s^2 –3*s + 2*
*n = 1 + 2*s*
G = z/n

Wie Sie sehen, können wir auf diese Weise die Übertragungsfunktion in der uns vertrauten Art aufstellen und damit weiterrechnen, z.B. ein Bodediagramm erstellen oder das dynamische System in einen Regelkreis einbauen

Aufgabe:
Stellen Sie die Übertragungsfunktion G_PID_T1 eines PID-T_1-Reglers mit den Koeffizienten $K_R = 2$, $T_n = 6\,\text{sec}$, $T_v = 2\,\text{sec}$, *Verzögerungszeit* $T_1 = 0.5\,\text{sec}$ *auf*

3.1.4 Polynome, Übertragungsfunktionen und Scilab-Listen list

Die nachfolgenden Informationen sind dazu gedacht, Ihnen einen kleinen Einblick in die Interna der Programmierung von Scilab zu geben. Um ein Polynom zu speichern sind folgende Informationen nötig:
- Angabe, ob es sich um komplexe oder reelle Koeffizienten handelt
- Koeffizienten des Zählers
- Koeffizienten des Nenners

Auf diese Elemente der Liste kann man zugreifen. Wenn G die Übertragungsfunktion aus der Aufgabe in 3.1.3 ist, dann erhält man folgende Elemente der Liste:

G(1) = 'r', da es sich um eine reelle Funktion handelt
G(2) = 2 – 3s + s^2, das Zählerpolynom
G(3) = 1 + 2s, das Nennerpolynom

Es gibt noch ein 4. Listenelement, das bei der vorliegenden Übertragungsfunktion leer ist.

Aufgabe:
Verifizieren Sie die Elemente der Liste

3.2 Beschreibung linearer dynamischer Systeme

In 3.1.3 wurde die Übertragungsfunktion eines PIDT_1-Systems aufgestellt. Im folgenden werden wir eine Verzögerungsstrecke 3. Ordnung aus der Hintereinanderschaltung dreier PT1-Systeme mit den 3dB-Grenzfrequenzen $f_g = 1.0\,\text{Hz}$ in ihren verschiedenen Darstellungsarten untersuchen.
Dabei werden wir
- die Übertragungsfunktion G_{PT3} als Multiplikation der drei gleichen Einzelübertragungsfunktionen $G_{PT1}(s)$,

- die Zustandsraumdarstellung aus der Verkettung der drei Einzelsysteme (siehe „Modellbildung und Simulation“) sowie
- die Zustandsraumdarstellung in Regelungs-Normalform

darstellen und aus diesen drei Darstellungen jeweils die (von vornherein bekannten) Wurzeln der charakteristischen Gleichung mit Scilab berechnen.

Im Anschluß daran lernen wir die Umwandlung der Darstellung durch eine Übertragungsfunktion in die Zustandsraumdarstellung und umgekehrt durch Scilab kennen, bevor wir das Bodediagramm dieser Strecke sowie eines $PIDT_1$-Reglers darstellen.

Abschließend bauen wir Regler und Strecke zu einem Regelkreis zusammen und untersuchen den Verlauf der Wurzeln des geschlossenen Regelkreises, wenn wir nur einen P-Regler verwenden und die Verstärkung von 0 bis ∞ gehen lassen.

Diese Darstellung soll auch durch die 3D-Darstellung der Übertragungspole für einige Reglerverstärkungen K_R mit der Flächendarstellung in Scilab veranschaulicht werden.

3.2.1 Die Scilab-Funktion syslin

Lineare dynamische Systeme lassen sich im Frequenzbereich durch die Übertragungsfunktion $G(s)$ und im Zeitbereich durch ein System von Differentialgleichungen 1. Ordnung darstellen. Beide Darstellungen sind völlig äquivalent. Um solche Systeme in den verschiedenen Darstellungen einheitlich behandeln zu können, stellt Scilab die Funktion *syslin* zur Verfügung. Den Gebrauch dieser Funktion und den Einsatz von Umwandlungsfunktionen aus einem in den anderen Bereich lernen wir anhand einer Regelstrecke 3. Ordnung, wie wir sie aus „Modellbildung und Simulation“ kennen.

Als Strecke soll uns die Hintereinanderschaltung dreier PT_1-Glieder dienen (Abb. 33), die vom Standpunkt des Schwingungstechnikers aus Tiefpässe sind. Bei der Grenzfrequenz f_g ist die Amplitude einer Schwingung auf $1/\sqrt{2}$ abgesunken und die Phasenverschiebung beträgt 45° ([5], [6]). Diese Grenzfrequenz hängt mit der Zeitkonstante T eines Tiefpasses folgendermaßen zusammen: $T = \frac{1}{2\pi \cdot f_g}$. Der Einfachheit sei der proportionale Übertragungsbeiwert $K_p = 1$.

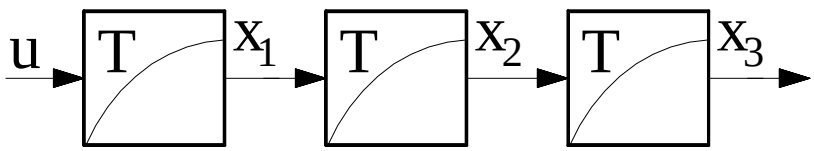


Abb. 33 PT_3-System mit Übertragungsbeiwert 1

3.2.1.1 Übertragungsfunktion des PT_3-Systems

Die Funktion syslin sammelt die Daten eines linearen Systems in einer Variablen vom Typ tlist, und überprüft dabei die Eingaben auf Schlüssigkeit. Der Funktionsaufruf sieht im Prinzip folgendermaßen aus:

sy = syslin('c',Zählerpolnom, Nennerpolynom)

Liegt die Übertragungsfunktion bereits als Quotient $G(s)$ vor, so wird das Nennerpolynom einfach weggelassen und als Zählerpolynom die bereits berechnete Übertragungsfunktion $G(s)$ genommen. Das Zeichen 'c' bedeutet „kontinuierliches System" .

Das Ergebnis ist eine durch die Funktion tlist erzeugte Liste, die folgende Elemente enthält: ['r','num','den','dt'], Zählerpolynom, Nennerpolynom, 'c'

Die Funktion tlist haben wir bereits verwendet, um nachträglich eine Achsteilung mit Beschriftung zu ändern (siehe *Rettich4_Animation5.sce)*

Aufgabe:

- *Stellen Sie die Übertragungsfunktion dreier PT_1-Systems mit den obigen Werten in Scilab zunächst durch eine gebrochen rationale Funktion dar, wie Sie es in 3.1.3 gelernt haben. Verwenden Sie als Grenzfrequenzen der Tiefpässe $f_{g1} = 1.0\text{Hz}$, $f_{g2} = 5.0\text{Hz}$ und $f_{g3} = 25\text{Hz}$. Die zugehörigen Zeitkonstanten ergeben sich daraus durch $T = 1/(2\pi f_g)$.*
  Nennen Sie die entstandenen Übertragungsfunktion G_PT1_1, G_PT1_2 und G_PT1_3.
- *Bilden Sie mit Scilab daraus die Übertragungsfunktion G_PT3 des PT_3-Systems.*
- *Weisen Sie der Variablen sy_Freq das durch die Übertragungsfunktion definierte lineare System zu.*

3.2.1.2 Zustandsraumdarstellung der hintereinandergeschalteten PT_1-Systeme

Die Differentialgleichungen der 3 PT_1-Systeme haben die Form

$$\dot{x}_1 = -\frac{1}{T_1}\cdot x_1 + \frac{1}{T_1}\cdot u$$

$$\dot{x}_2 = -\frac{1}{T_2}\cdot x_2 + \frac{1}{T_2}\cdot x_1$$

$$\dot{x}_3 = -\frac{1}{T_3}\cdot x_3 + \frac{1}{T_3}\cdot x_2$$

In Matrixschreibweise ergibt sich daraus

$$\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\\ \dot{x}_3\end{pmatrix} = \begin{pmatrix}-\frac{1}{T_1} & 0 & 0\\ +\frac{1}{T_2} & -\frac{1}{T_2} & 0\\ 0 & +\frac{1}{T_3} & -\frac{1}{T_3}\end{pmatrix}\cdot\begin{pmatrix}x_1\\ x_2\\ x_3\end{pmatrix} + \begin{pmatrix}\frac{1}{T_1}\\ 0\\ 0\end{pmatrix}\cdot u$$

Die spätere Regelgröße y ergibt sich mit der Meßmatrix C durch

$$y = (0\ 0\ 1)\cdot\begin{pmatrix}x_1\\ x_2\\ x_3\end{pmatrix} + 0\cdot u$$

oder allgemein (siehe „Modellbildung und Simulation“)

$$\dot{\vec{x}} = A\cdot\vec{x} + B\cdot\vec{u}$$
$$\vec{y} = C\cdot\vec{x} + D\cdot\vec{u}$$

mit Zustandsvektor $\vec{x}$, Systemmatrix A, Eingangsvektor $\vec{u}$, Meßvektor $\vec{y}$, Meßmatrix C und Durchgriffmatrix D. Letztere ist in den allermeisten Fällen 0, da der Eingangsvektor in der Praxis nie unmittelbar unverzögert auf den Ausgang wirkt.

In Scilab läßt sich das dynamisches System, definiert durch das obige Differentialgleichungssystem, ebenfalls darstellen durch die Funktion syslin. Der Aufruf unterscheidet sich geringfügig von der Darstellung mit Hilfe der Übertragungsfunktion:

sy = syslin('c',A,B,C,D) oder
sy = syslin('c',A,B,C)

Im letzteren Fall wird D zu Null angenommen.

Das Flag 'c' signalisiert, daß es sich um die Darstellung eines kontinuierlichen Systems handelt. Ein 'd' steht für ein digitales System, das hier nicht behandelt wird.
A,B und C sind die oben angegebenen Matrizen.
sy ist eine Variable vom Typ tlist, die wir als Funktion bereits im Programm *Rettich4_Animation5.sce* kennengelernt haben zur nachträglichen Änderung der Achsteilung.

Aufgabe:
Stellen Sie das PT_3-System mit Hilfe der Funktion syslin dar und weisen Sie das Ergebnis der Variablen sy_zeit zu.

3.2.2 Darstellung des Frequenzverhaltens eines dynamischen Systems

Das Frequenzverhalten eines linearen dynamischen Systems läßt sich durch Auswertung der Übertragungsfunktion $G(s) = \frac{\hat{y}}{\hat{u}} \cdot e^{j(\varphi_y - \varphi_u)}$ für technisch-physikalische Frequenzen f mit $s = j\omega = j \cdot 2\pi f$, also längs der imaginären Achse auswerten. Die Länge $|G(j2\pi f)|$ des Zeigers, der sich bei einer bestimmten Frequenz f ergibt, stellt das Verhältnis $a^* = \hat{y}/\hat{u}$ der Amplituden von Eingangssignal $u(t)$ und Ausgangssignal $y(t)$ dar. Der Winkel b, den dieser Zeiger bei der Frequenz f mit der reellen Achse einschließt, ist die Phasenverschiebung $b = \varphi_y - \varphi_u$ zwischen Ein- und Ausgangssignal.

Bei der Darstellung kann man beide Größen Amplitudenverhältnis a und Phasenverschiebung b in einem einzigen Diagramm darstellen, der Ortskurve des Frequenzgangs, bekannt auch als Nyquistdiagramm. Es hat den Vorteil, daß man schnell einen qualitativen Überblick über das dynamische Verhalten, insbesondere über die Stabilität des Systems bei geschlossenem Regelkreis erhält. Allerdings kann man es nur schwer für quantitative Auswertungen beim Entwurf eines geeigneten Reglers verwenden. Geigneter dafür ist die Darstellung, bei der über der logarithmisch geteilten Frequenzachse die Größe $a = 20 \cdot \lg(a^*)$ in Dezibel dB und in einem zweiten Diagramm über derselben Frequenzteilung die Phasenverschiebung b aufgetragen wird. Diese Darstellung ist als Bodediagramm bekannt.

3.2.2.1 Die Ortskurve des Frequenzgangs (Nyquist-Diagramm)

Aus der Ortskurve des Frequenzgangs läßt sich die Stabilität eines Regelkreises allein aus dem Frequenzverhalten des offenen Regelkreises beurteilen. Dazu wird der geschlossene Regelkreis aufgetrennt (Abb. **34**) und die Übertragungsfunktion $G(s) = G_R(s) \cdot G_S(s)$ des offenen Regelkreises aufgestellt.

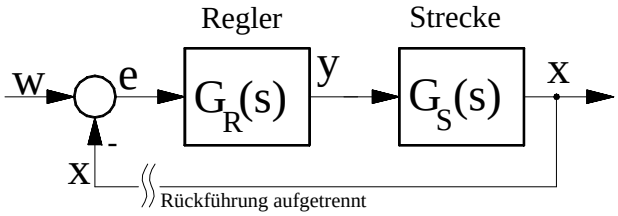


Abb. 34 Auftrennung der Rückführung zur Ermittlung der Ortskurve des Frequenzgangs und des Bodediagramms

Schließt man den Regelkreis, so ergibt sich die Übertragungsfunktion des geschlossenen Regelkreises zu

$$G_{RK}(s) = \frac{G_R(s) \cdot G_S(s)}{1 + G_R(s) \cdot G_S(s)}$$

Wenn für irgendeine technisch-physikalische Frequenz f_0 die Übertragungsfunktion $G(s)$ des offenen Regelkreises den Wert –1 annimmt, dann wird der Nenner zu Null und das System schwingt ungedämpft bei dieser Frequenz f_0. Denn bei dieser Frequenz schwingt x genau mit einer Phasenverschiebung von –180° mit gleicher Amplitude, wie das Eingangssignal e. Durch die zusätzliche Einspeisung mit einem Minuszeichen in die Summationsstelle gelang x wieder phasenrichtig mit gleicher Amplitude an den Eingang e, so daß zur Aufrechterhaltung der Schwingung w nicht benötigt wird und die Schwingung x sich selbst aufrecht erhält.

Dieser Gedankengang führt letztlich auf das (vereinfachte) Stabilitätskriterium von Nyquist, das zwei Jahre vor der Veröffentlichung durch Harry Nyquist bereits in Deutschland 1930 von Felix Strecker bei Siemens ähnlich formuliert wurde (s. Vorlesung Automatisierungstechnik, Modellbildung und Simulation).

Will man also beispielsweise untersuchen, wie groß man die Verstärkung K_R eines P-Reglers wählen darf, bis der geschlossene Regelkreis mit einer PT_3-Strecke ungedämpft schwingt, dann kann man dies mit Hilfe des Nyquist-Diagramms machen, indem man die Verstärkung so lange vergrößert, bis die Ortskurve den Punkt –1 in der komplexen $G(j \cdot 2\pi f)$-Ebene schneidet.
Dies werden wir im folgenden Beispiel untersuchen.

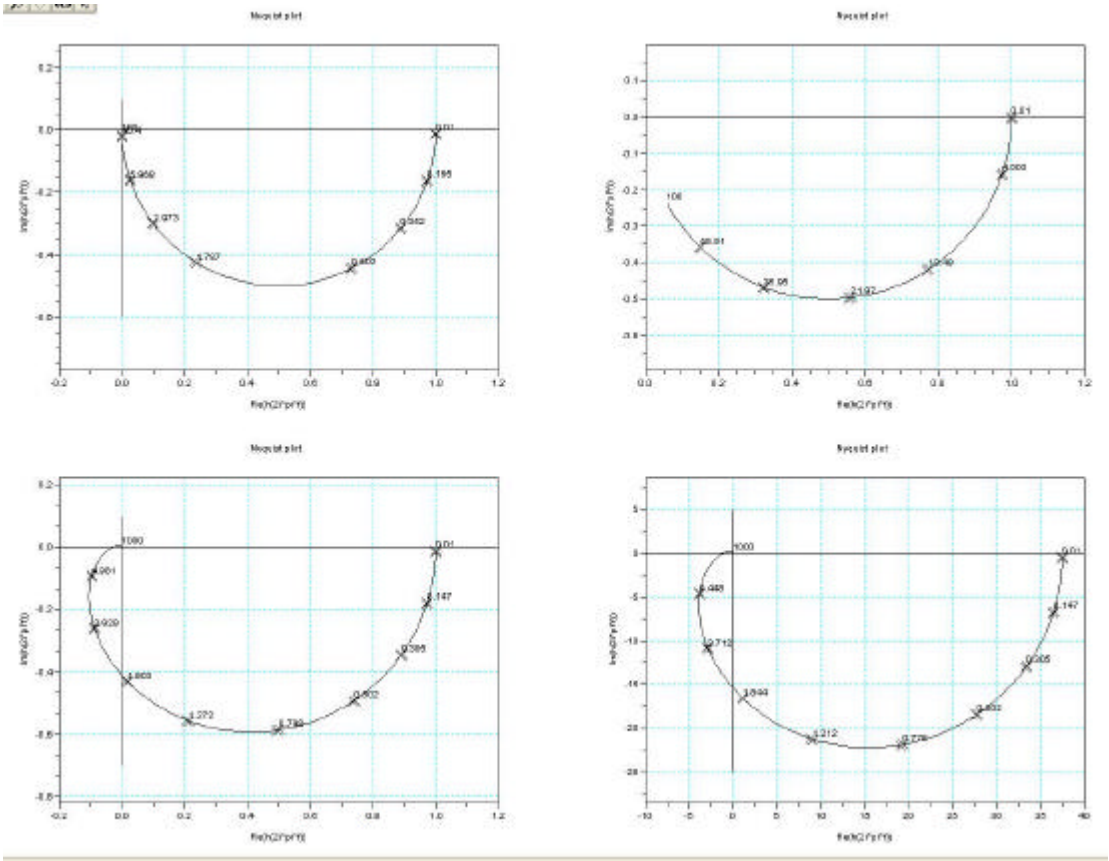


Abb. 35 Ortskurven des Frequenzgangs der PT_1-Glieder mit f_g = 1.0Hz und 25Hz, PT_3-Strecke aus PT_1-Gliedern mit Grenzfrequenz 1.0Hz, 5.0Hz und 25Hz sowie des offenen Regelkreises mit K_R = 37.5

Als Strecke wird das PT_3 – System aus der Aufgabe in Kap. 3.2.1.1 gewählt. und in Reihe dazu ein P-Regler mit der Verstärkung K_R geschaltet. Dargestellt werden in einer ersten Bilderserie (Abb. 35) die Ortskurven des Frequenzgangs der

- zwei PT_1-Glieder mit 1Hz und mit 25Hz Grenzfrequenz (erste Zeile links und rechts),
- der PT_3-Systems mit Streckenverstärkung 1
- der PT_3-Strecke und in Reihe geschalteter P-Regler mit K_R = 37.5

Um herauszufinden, bei welcher Frequenz der geschlossene Regelkreis später ungedämpft schwingen wird, erfolgt noch eine Darstellung in der Nähe der Frequenzen, bei der die Ortskurve in der Nähe des Punktes –1 verläuft.

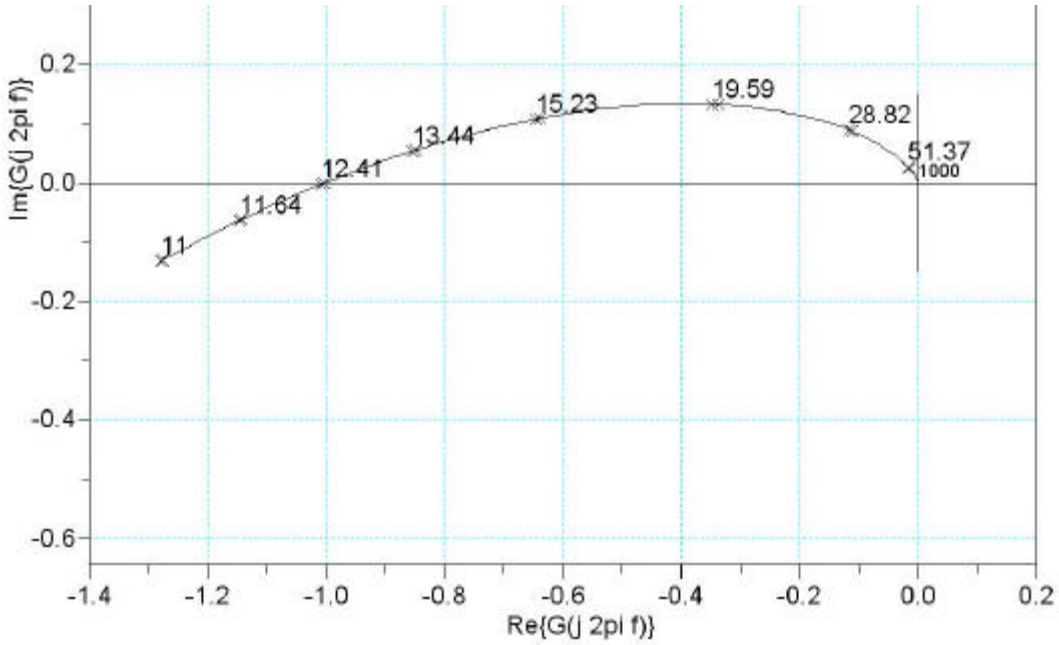


Abb. 36 Schnittpunkt der Ortskurve mit dem Punkt -1

Scilab stellt mit dem Befehl *nyquist(sy, fmin, fmax)* die Übertragungsfunktion $G(s)$ eines linearen Systems sy längs der imaginären Achse, also für $s = j \cdot 2\pi f$ dar.

Aufgabe:
Laden Sie das Programm Aufg_32.sce und gehen Sie mit dem Debugger alle Schritte durch bis hin zum Bodediagramm. Achten Sie insbesondere darauf,
- *wie mehrere Diagramme in einem einzigen Bild erzeugt werden*
- *wie die Achsen auf gleiche Dimension gebracht werden*

3.2.2.2 Das Bode-Diagramm

Die Ortskurve des Frequenzgangs gibt zwar einen guten Überblick über die Dynamik eines Systems, sobald aber quantitative Aussagen zu treffen sind, ist es besser, die Amplituden- und die Phaseninformation von $G(j \cdot 2\pi f)$ getrennt darzustellen. Zudem liegen äquidistante Frequenzwerte mit zunehmender Frequenz immer enger zusammen. Daher ist im Bodediagramm die Einteilung der Abszisse logarithmisch, so daß die Frequenzdekaden gleiche Größe haben. Darüber hinaus wird auch der Amplitudengang a(f) logarithmisch dargestellt durch $a(f) = 20 \cdot \lg|G(j \cdot 2\pi f)|$ mit der Pseudodimension dB (Dezibel). Damit lassen sich wichtige Werte für den Reglerentwurf mit genügender Genauigkeit entnehmen (siehe Automatisierungstechnik: Modellbildung und Simulation)

Das folgende Bodediagramm in Abb. 37 zeigt Amplituden- und Phasengang des PT_3-Systems mit den Eckfrequenzen 1Hz, 5Hz und 25 Hz

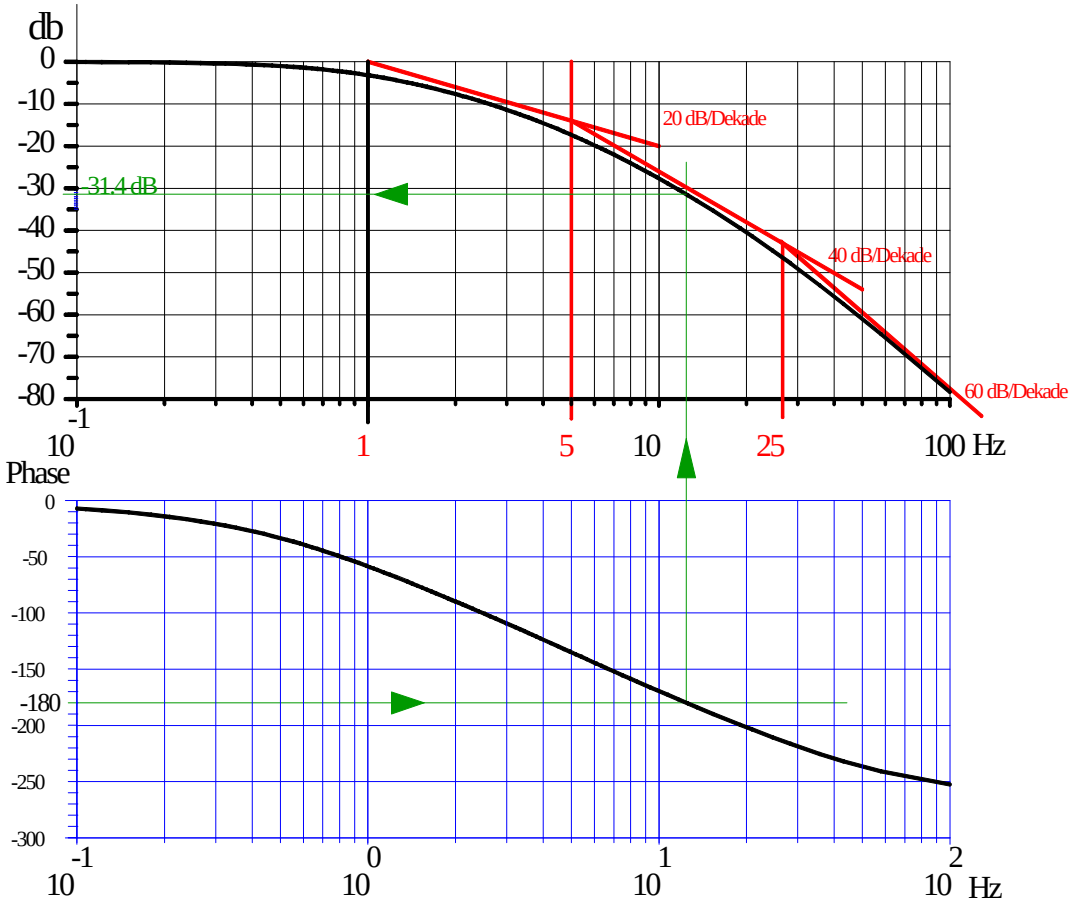


Abb. 37 Amplituden- und Phasengang des PT_3-Systems, Ermittlung der Amplitudenreserve

Aus dem Diagramm läßt sich viel genauer und ohne „Probieren“ ablesen, bei welcher Verstärkung K_R ein in Reihe geschalteter P-Regler haben muß, um genau die Stabilitätsgrenze zu erreichen. Wie unter 3.2.2.1 erläutert, muß die Übertragungsfunktion des offenen Regelkreises bei dieser Frequenz gerade den Wert –1 einnehmen. Anders ausgedrückt muß die Phase –180° betragen und der Amplitudengang muß 0dB sein.
Betrachtet man den Phasengang der PT_3-Srecke allein, so muß man nur nachsehen, bei welcher Frequenz eine Phasenverschiebung von –180° erreicht wird. Sieht man im Amplitudengang nach, welche „Verstärkung“ a das PT_3-System bei dieser Frequenz hat (im obigen Beispiel –31.4dB), dann muß die Verstärkung des in Reihe geschalteten P-Reglers –a (also + 31.4 dB $\hat{=}$ K_R = 37.15) betragen.

In Scilab kann man das Bode-Diagramm mit dem Befehl bode(sy,fmin,fmax) darstellen

Aufgabe:

Stellen Sie die Bode-Diagramme der drei PT_1-Systeme und das Bode-Diagramm der PT_3-Strecke in einem einzigen Bild mit 4 Unterbildern analog zu den Ortskurven in Abb. 36 dar.

[1] Pinçon, Bruno — Eine Einführung in Scilab, Version 0.9999, Institut Elie Cartan Nancy E.S.I.A.L. Université Henri Poincaré , Übersetzung: Agnes Mainka, Helmut Jarausch, IGPM, RWTH Aachen, http://kiwi.math.uni-mannheim.de/~mfenn/Matlabuebung/PinconD.pdf

[2] Zogg, Jean-Marie — Arbeiten mit Scilab und Scicos, Hochschule für Technik und Wirtschaft HTW Chur, http://www.fh-htwchur.ch/uploads/media/Arbeiten_mit_Scilab_und_Scicos_v1.pdf

[3] Scilab Group INRIA — Signal Processing with Scilab, http://www.scilab.org/doc/signal.pdf

[4] Scilab Group INRIA — Introduction to Scilab, http://www.scilab.org/doc/intro/intro.pdf

[5] Höcht, Johannes — Zeitverhalten und Stabilität linearer dynamischer Systeme, Hochschule München, FK 03, 2008

[6] Höcht, Johannes — Steuerungs- und Regelungstechnik – Zusammenfassung wichtiger Grundlagen, Hochschule München, FK 03, 2005