v1.2

Matlab-Übungen zu Deterministische Signale und Systeme Musterlösung zur 5. Übung

1. Aufgabe

In dieser Aufgabe geht es um grundlegende Techniken, die bei der Laplacetransformation Anwendung finden. Um die Aufgabe lösen zu können, sollten Sie sich über den Matlab-Befehl `residue` informieren (wie Sie die Hilfe nutzen, wissen Sie inzwischen vermutlich). *Hinweis: Die Ergebnisausgabe von Matlab und Octave unterscheidet sich hier geringfügig, beachten Sie daher bitte die Funktionsweise des verwendeten Programms.*

1.1. Gegeben sei die Funktion

$$f(p) = \frac{-6p^5 - 10p^4 - p^3 + 12p^2 + 6p - 2}{2p^3 + 4p^2 + 3p}.$$

Berechnen Sie die Polstellen von $f(p)$ und wenden Sie eine Partialbruchzerlegung an. Überprüfen Sie Ihr Ergebnis mit der `residue`-Funktion aus Matlab.

1.2. Plotten Sie die Polstellen der Funktion

$$g(p) = \frac{5p^2 + 10p + 8}{p^3 + 3p^2 + 4p + 2}.$$

Überprüfen Sie das Ergebnis durch Rechnung auf dem Papier.

Hinweis: Sie können den Plot-Befehl anweisen, nur einzelne Punkte zu setzen, indem Sie ihm eine entsprechende Formatierungsanweisung als drittes Argument übergeben, etwa `'o'`.

1.3. Schreiben Sie eine Funktion `poldiag`, der Sie ein Zähler- und ein Nennerpolynom übergeben und die die Polstellen für diese Funktion zeichnet. Achten Sie dabei auf eine korrekte Achsenbeschriftung. Der dargestellte Bereich soll dabei immer dem 1,2-fachen des höchsten vorkommenden Real- oder Imaginärteils entsprechen.

Lösung

1.1. Zuerst durch Polynomdivision den Zählergrad verringern

$$\begin{array}{r} \left(-6p^5-10p^4\ -p^3+12p^2+6p-2\right) : \left(2p^3+4p^2+3p\right) = -3p^2+p+2+\dfrac{p^2-2}{2p^3+4p^2+3p} \\ \underline{6p^5+12p^4+9p^3} \\ 2p^4+8p^3+12p^2 \\ \underline{-2p^4-4p^3\ -3p^2} \\ 4p^3\ +9p^2+6p \\ \underline{-4p^3\ -8p^2-6p} \\ p^2\qquad -2 \end{array}$$

Der Rest

$$\frac{p^2-2}{2p^3+4p^2+3p}$$

wird nun mittels Partialbruchzerlegung zerlegt. Zuerst bestimmt man die Nullstellen des Nennerpolynoms zu

$$p_1 = 0; \quad p_{2,3} = -1 \pm j\frac{1}{\sqrt{2}}$$

Nun kann man aus der Gleichung

$$\frac{p^2-2}{2p\left(p+1-\frac{j}{\sqrt{2}}\right)\left(p+1+\frac{j}{\sqrt{2}}\right)} = \frac{A}{2p} + \frac{B}{p+1-\frac{j}{\sqrt{2}}} + \frac{C}{p+1+\frac{j}{\sqrt{2}}}$$

A, B und C bestimmen.

Nach dem Multiplizieren mit $2p$

$$\left.\frac{2p(p^2-2)}{2p(p^2+2p+\frac{3}{2})}\right|_{p=0} = A$$

ergibt sich A durch Einsetzen von 0 zu

$$A = -\frac{4}{3}.$$

Auf dem gleichen Weg bestimmt man nun B zu

$$B = \frac{7}{12} - j\frac{\sqrt{2}}{12} \approx 0{,}5833 - 0{,}1179j.$$

Damit ist

$$C = \frac{7}{12} + j\frac{\sqrt{2}}{12} \approx 0{,}5833 + 0{,}1179j.$$

Die vollständige Zerlegung hat also die Form

$$f(p) = -3p^2 + p + 2 - \frac{\frac{4}{3}}{2p} + \frac{\frac{7}{12} - j\frac{\sqrt{2}}{12}}{p+1-\frac{j}{\sqrt{2}}} + \frac{\frac{7}{12} + j\frac{\sqrt{2}}{12}}{p+1+\frac{j}{\sqrt{2}}}$$

Auf die Eingabe

```
[R, P, K] = residue([-6, -10, -1, 12, 6, -2], [2, 4, 3, 0])
```

liefert Matlab die Werte

```
R =
   0.5833 - 0.1179i
   0.5833 + 0.1179i
  -0.6667
P =
  -1.0000 + 0.7071i
  -1.0000 - 0.7071i
        0
K =
    -3     1     2
```

wobei `R` die Zähler und `P` die Polstellen des zerlegten Restpolynoms enthält und in `K` die Koeffizienten des abgespalteten Polynoms enthalten sind. Ein Vergleich der Koeffizienten zeigt, dass die Lösungen identisch sind.

1.2. Mit

```
[R, P, K] = residue([5, 10, 8], [1, 3, 4, 2]);
plot(real(P), imag(P), 'o');
xlabel('\sigma'); ylabel('j\omega');
xlim([-2, 2]); ylim([-2, 2]); grid;
```

plottet man die Polstellen der Funktion $g(p)$. Ein Nachrechnen auf dem Papier ergibt, dass die Polstellen bei $p_1 = -1$ und bei $p_{2,3} = -1 \pm j$ liegen.

1.3. Die geforderte Funktion führt genau die Schritte aus, die im vorangegangenen Beispiel erforderlich waren. Folglich löst die Funktion

```
function poldiag(Z, N)
% POLDIAG   plots the poles of the function given by the
%   numerator polynomial Z and the denominator polynom N.

[R, P, K] = residue(Z, N);
figure;
plot(real(P), imag(P), 'o');
grid;
xlabel('\sigma'); ylabel('j\omega');

m = 1.2* max(abs([ real(P); imag(P)]));

xlim([-m, m]); ylim([-m, m]);
```

die Aufgabe.

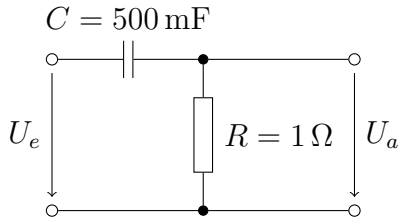

Abbildung 1: Schaltung

2. Aufgabe

Mit den Befehlen `mesh` und `surf` lassen sich Funktionen dreidimensional darstellen. Dabei erwartet Matlab, dass den Befehlen drei Matrizen übergeben werden. Die erste übergebene Matrix enthält x-Werte und die zweite Matrix y-Werte so, dass sich aus an der gleichen Stelle in den Matrizen liegende Werte Wertepaare ergeben, zu denen ein zugeordneter z-Wert an der entsprechenden Stelle in einer dritten Matrix zu finden ist.

Will man etwa die Funktion $z(x, y) = (x + y)^2$ von -1 bis 1 darstellen, wobei in beide Richtungen jeweils 5 Werte gegeben sein sollen, so benötigt man die Matrizen

$$X = \begin{pmatrix} -1 & -0{,}5 & 0 & 0{,}5 & 1 \\ -1 & -0{,}5 & 0 & 0{,}5 & 1 \\ -1 & -0{,}5 & 0 & 0{,}5 & 1 \\ -1 & -0{,}5 & 0 & 0{,}5 & 1 \\ -1 & -0{,}5 & 0 & 0{,}5 & 1 \end{pmatrix} \quad \text{und} \quad Y = \begin{pmatrix} -1 & -1 & -1 & -1 & -1 \\ -0{,}5 & -0{,}5 & -0{,}5 & -0{,}5 & -0{,}5 \\ 0 & 0 & 0 & 0 & 0 \\ 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 & 0{,}5 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

und kann dann mit

```
>> Z = (X+Y).^2;
```

die Funktion $z(x, y)$ in der Matrix `Z` abbilden. Die Matrizen `X` und `Y` lassen sich sehr einfach mit der Funktion `meshgrid` erzeugen, der Sie je einen Vektor mit x- und y-Werten übergeben. Die beiden Matrizen aus dem Beispiel lassen sich etwa mit

```
>> [X, Y] = meshgrid(-1 : .5 : 1, -1 : .5 : 1)
```

erzeugen.

Gegeben sei die Schaltung aus Abbildung 1 sowie die Laplace-Übertragungsfunktion

$$G(p) = \frac{1}{p + 2 + \frac{1}{p}}.$$

Der Kondensator ist zum Zeitpunkt $t = 0$ ungeladen.

2.1. Bestimmen Sie die Laplace-Übertragungsfunktion

$$F(p) = \frac{U_a(p)}{U_e(p)}$$

der Schaltung aus Abbildung 1.

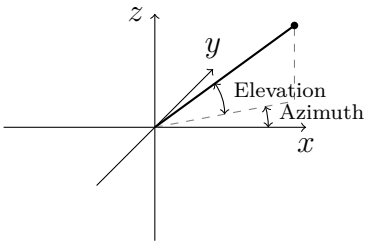

Abbildung 2: Azimuth und Elevation.

2.2. Stelle Sie die beiden Übertragungsfunktionen grafisch in einem Bereich von $\sigma \in [-5, 5]$ und $\omega \in [-5, 5]$ dar.

2.3. Mit dem Befehl `view` können Sie die »virtuelle Kamera« des 3D-Plots bewegen. Dem Befehl wird dabei ein Vektor mit zwei Elementen übergeben. Der erste Wert bestimmt den Azimuth, also den Horizontalwinkel der Kamera, der zweite Wert die Elevation, also den Vertikalwinkel (siehe auch Abbildung 2).

Mit den Befehlen `xlim`, `ylim` und `zlim` können sie wie beim 2D-Plot den gewünschten Achsenausschnitt wählen.

Stellen Sie mit diesen Hilfsmitteln die Fourier-Übertragungsfunktion des Systems aus Abbildung 1 ausgehend von der Laplace-Übertragungsfunktion dar.

2.4. Verwenden Sie die Technik aus der vorherigen Teilaufgabe, um herauszufinden, um was für ein System es sich bei $G(p)$ handelt (Hoch-, Tief-, Bandpass oder Bandsperre).

2.5. Fügen Sie $G(p)$ eine Nullstelle bei $p = 0$ hinzu. Um was für ein System handelt es sich nun? Zeichnen Sie eine Schaltung, die dieses System realisiert.

Lösung

2.1. Da der Kondensator bei $t = 0$ ungeladen ist, lässt sich die Übertragungsfunktion einfach aus der Schaltung ablesen:

$$F(p) = \frac{p}{p + \frac{1}{RC}}$$

2.2. Mit den zu Beginn der Aufgabe erwähnten Techniken lassen sich die Vorbereitungen für den 3D-Plot treffen (`Aufgabe2.m`). Zuerst wird das »Gitter« für die σ- und $j\omega$-Achse erstellt:

```
[S, W] = meshgrid(-5 : .1 : 5, -5 : .1 : 5);
```

Um die Übertragungsfunktionen leichter eingeben zu können, kann man eine Variable (genauer eine Matrix) anlegen, die die Funktion $p = \sigma + j\omega$ realisiert:

```
P = S + j*W;
```

Nun können die beiden Funktionen mit `mesh()` oder `surf()` dargestellt werden:

```
figure(1);
surf(S, W, abs(F));
xlabel('\sigma');
ylabel('j\omega');

figure(2);
surf(S, W, abs(G));
xlabel('\sigma');
ylabel('j\omega');
```

Da es sich bei $F(p)$ und $G(p)$ um komplexe Funktionen handelt, wurde hier für die Anzeige der Betrag gewählt.

2.3. Die Laplaceübertragungsfunktion entspricht für $\sigma = 0$ der Fourier-Übertragungsfunktion. Um sie anzuzeigen, kann z. B. die entsprechende Zeile der Matrix `F` entlang der ω-Achse extrahiert und mit **`plot`** dargestellt werden. Allerdings ist nicht unbedingt ein Wert genau bei $\sigma = 0$ vorhanden. Daher ist es hier einfacher, im 3D-Plot den Bereich auf einen schmalen Streifen um die $j\omega$-Achse einzugrenzen und die Darstellung mit **`view()`** so zu wählen, daß ein 2D-Plot entsteht:[1]

```
figure(1);
xlim([-.01, .01]);
zlim([0, 2]);
view([90, 0]);
```

Es zeigt sich, dass es sich bei $F(p)$ um einen Hochpass handelt.

2.4. Auf die gleiche Weise findet man mit

```
figure(2);
xlim([-.01 .01]);
zlim([-2 2]);
view([90 0]);
```

heraus, dass es sich bei $G(p)$ um einen Bandpass handelt.

2.5. Eine Nullstelle bei $p = 0$ erzeugt man, indem man $G(p)$ mit p multipliziert:

```
H = P .* G;

figure(3);
surf(S, W, abs(H));
xlabel('\sigma');
ylabel('j\omega');
xlim([-.01, .01]);
zlim([0, 1]);
view([90, 0]);
```

[1] Das funktioniert nur, wenn für den Plot die Parallelprojektion gewählt ist – das ist auch die Standardeinstellung. Neben der Parallelprojektion beherrscht Matlab auch noch die Fluchtpunktdarstellung, welche Sie im Menü des Plotfensters auswählen können.

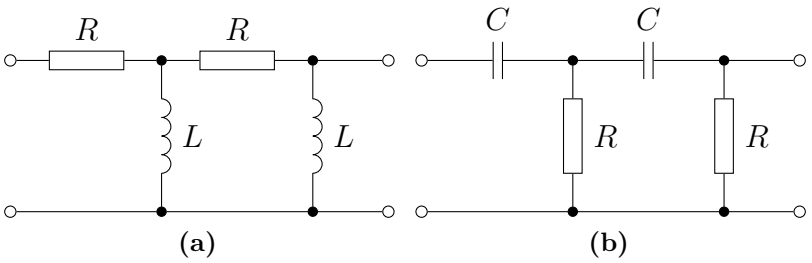

Abbildung 3: Hochpasskonfigurationen zur Übertragungsfunktion $H(p)$.

Es zeigt sich, dass nun wieder um einen Hochpass handelt. Die Übertragungsfunktion ist nun

$$H(p) = pG(p) = \frac{p}{p + 2 + \frac{1}{p}} = \frac{p^2}{p^2 + 2p + 1}.$$

Zerlegt man das Nennerpolynom nun gemäß

$$H(p) = \frac{p}{(p+1)} \frac{p}{(p+1)}$$

kann man erkennen, dass man das System durch die Hintereinanderschaltung zweier RL-Hochpässe realisieren kann. Allerdings ist die Darstellung

$$H(p) = \frac{1}{\left(1 + \frac{1}{p}\right)} \frac{1}{\left(1 + \frac{1}{p}\right)}$$

äquivalent, was der Hintereinanderschaltung zweier RC-Hochpässe entspricht. Die Übertragungsfunktion ist also nicht eindeutig einer Schaltung zuzuordnen. Die beiden genannten Schaltungen sind in Abbildung 3 gezeichnet.

3. Aufgabe

Diese Aufgabe ist leider nur in Matlab und nicht in Octave lösbar, da Octave die GUI-Funktionalität von Matlab fehlt.

Für diese Aufgabe gibt es ein Matlab-GUI[2] und damit ein vorgegebenes Framework, dessen Funktion Sie vervollständigen sollen. Die Dateien finden Sie zum Herunterladen auf der Web-Seite zur Vorlesung.

Zum GUI gehören die Datei `faltung.fig`, die die grafische Darstellung enthält und die Datei `faltung.m`, die den Quelltext zum GUI enthält. Zum Starten des GUI führen Sie bitte `faltung.m` mit Matlab aus.

Die Datei `faltung.m` versucht dabei, auf eine Funktion `falt.m` im gleichen Verzeichnis zurückzugreifen, die allerdings fehlt (und die Sie schreiben sollen, Sie haben ja in einer vergangenen Übung schon einmal die Faltung implementiert, wissen also, wie das geht).

Die Funktion `falt` soll dabei die Signatur

[2] Graphical User Interface

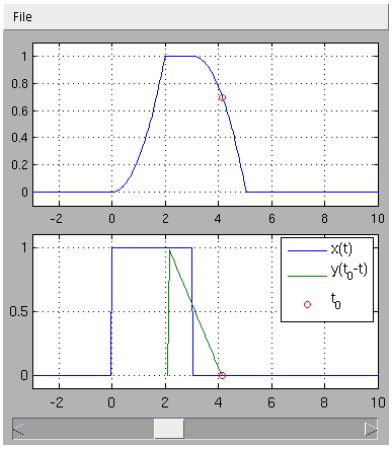

Abbildung 4: GUI zur Faltung.

```
function out = falt(t0, t, siga, tau, sigb)
```

haben. Die übergebenen Parameter sind

`t0` der aktuelle Zeitpunkt,

`t` die »globale« Zeitachse und gleichzeitig die Zeitachse von Signal `siga`,

`siga` das eine Signal,

`tau` die Zeitachse des Signals `sigb` und

`sigb` das zweite Signal.

Schreiben Sie die Funktion so, dass immer für die übergebenen Werte der Wert der Faltung zum Zeitpunkt `t0` berechnet wird. Beachten Sie dabei, dass die Zeitachsen trotz scheinbar »runder« Werte kleine Abweichungen von diesen Werten aufweisen können, ein direkter Vergleich wie `find(t==0.1)` muß also nicht unbedingt funktionieren.

Wenn Sie die Funktion `faltung` ohne Parameter in Matlab aufrufen, so sollte sich – falls Sie die Funktion `falt` implementiert haben – ein Fenster wie in Abbildung 4 öffnen. Das untere Diagramm zeigt dabei die zu faltenden Funktionen (wobei die Funktion $y(t)$ schon an der Ordinate gespiegelt dargestellt wird). Mit dem Schieberegler unter dem Graphen können Sie diese Funktion gegen die Funktion $x(t)$ verschieben. Ein roter Kreis markiert dabei die Stelle t_0. Im oberen Diagramm wird das Ergebnis der Faltung der beiden Funktionen zur Kontrolle Ihres Programms gezeichnet und der rote Punkt fährt, sollten Sie `falt` richtig implementiert haben, diese Kurve entsprechend der Stellung des Schiebereglers nach.

Im Folgenden finden Sie noch vier Zusatzaufgaben, die Sie lösen können, falls Sie sich in die GUI-Programmierung in Matlab weiter einarbeiten wollen (sie werden aber nicht in der Übung vorgerechnet und es wird auch keine Musterlösung erstellt, wir können aber Ihre Lösung über die Web-Seite zur Vorlesung anderen Studenten zur Verfügung stellen, falls Sie Interesse daran haben, schicken Sie einfach eine kurze E-Mail an `d.papsdorf@nt.tu-darmstadt.de`).

3.1. Erweitern Sie das GUI so, dass das Ergebnis der Faltung nur bis zum aktuellen Zeitpunkt t_0 sichtbar ist.

3.2. Ändern Sie das GUI so, dass zwei andere Funktionen gefaltet werden.

3.3. Erweitern Sie das GUI so, dass die Zeitachsen und Funktionen in einem weiteren M-File eingegeben werden können und man so beliebige Funktionen miteinander Falten kann.

3.4. Erweitern Sie das GUI so, dass man den Namen der Datei mit den Funktionen und Zeitachsen in einem Eingabefeld in dem GUI eingeben (oder über einen Auswahldialog auswählen) kann.

Lösung

Die Funktion

```
function out = falt(t0, t, siga, tau, sigb)
% FALT   returns the solution of the convolution integral of
%   siga and sigb at time t0 with respect to the global time
%   axis t. tau is the time axis for sigb.

a = max(t(1), t0-tau(end));
b = min(t(end), t0-tau(1));

out = zeros(1, length(t));
if (a <= b)
    n = floor((b-a) / (t(2)-t(1)));

    limc1 = find(t >= a, 1);

    limb2 = find(t0 - tau <= b, 1, 'last');

    sigc = zeros(1, length(t));
    sigc(limc1 + (0 : n-1)) = sigb(limb2 - (0 : n-1));

    out(find(t >= t0, 1)) = trapz(t, siga .* sigc);
end

end
```

bestimmt zuerst die Grenzen des Überlapps zwischen der Funktion `siga` und der gewendeten und veschobenen Funktion `sigb`. Sofern ein Überlapp vorhanden ist, wird der invertierte und verschobene Ausschnitt der Funktion `sigb` (`sigc`) mit `siga` multipliziert und aufintegriert.

Bei Benutzung der GUI fällt auf, dass beim halten des Sliders das Fenster langsam oder gar nicht mehr reagiert. Dies liegt daran, dass bei jeder Aktion des Steuerelemnts aufwändige Berechnungen in Form der Faltung initiiert werden. Dieses Problem läßt sich beheben, indem während der Berechnung ein erneuter Aufruf daktiviert wird. An dieser Stelle soll jedoch nicht zu weit auf die GUI Programmierung eingegangen werden.

4. Aufgabe

Durch das Einbinden zusätzlicher Programmoptionen, sogenannter Toolboxen, ist Matlab in der Lage spezielle mathematische Operationen aus natur- und ingenieurwissenschaftlichen Disziplinen zu berechen. So kann Matlab beispielsweise Probleme wie die Laplace-Transformation auch analytisch lösen, was allerdings voraussetzt, dass auf Ihrem Arbeitsplatz die Symbolic Toolbox installiert ist. Diese Aufgabe richtet sich also nur an Studenten, an deren Arbeitsplatz diese Option installiert ist.

Diese Aufgabe ist leider nur in Matlab und nicht in Octave lösbar, da für Octave keine Erweiterung zum symbolischen Rechnen existiert. Sie können diese Aufgabe aber in einem der zahlreichen Computer-Algebra-Systeme wie Maple, Derive, Mathematica oder als freie Alternative z. B. (WX)Maxima oder Axiom lösen, unser Lösungsvorschlag wird sich allerdings auf Matlab beschränken.

4.1. Führen Sie zunächst eine Partialbruchzerlegung des Ausdrucks

$$X(p) = \frac{9p^2 + 28p + 37}{p^3 + 8p^2 + 37p + 50}$$

aus Aufgabe 2 der 7. Übung zu Deterministische Signale und Systeme durch. Benutzen Sie den Befehl `residue`. Entspricht das Ergebnis der Lösung aus der Übungsaufgabe?

4.2. Führen Sie nun die inverse Laplace-Transformation des Signals durch. Schauen Sie sich hierzu zunächst die Hilfe zum Befehl `syms` an und benutzen Sie diesen um die Lösung analytisch zu berechen. Vergleichen Sie die inverse Transformierte der ursprünglichen mit der der partialbruchzerlegten Funktion.

4.3. Erzeugen Sie einen Vektor t, der die Werte von 0 bis 3 in Schritten von 0,1 enthält. Mit welchem Befehl kann man den »analytischen Variablen« nun Zahlenwerte zuweisen? Berechnen Sie die Funktionswerte der inversen Transformierten in dem genannten Bereich und stellen Sie sie graphisch dar – vergessen Sie die Achsenbeschriftung nicht.

Lösung

4.1. Die Fouriertransformierte liegt in der Form $F = b/a$ vor. Die Koeffizienten der Nenner- und Zählerpolynome werden in die Vektoren a und b geschrieben. Der Matlabhilfe kann man die Bedeutung von r, p, und k entnehmen. Das Ergebnis stellt sich etwas anders dar, da bei dieser Zerlegung jede Polstelle einen »eigenen Summanden« erhält. Die Ergebnisse können aber ineinander überführt werden.

```
b = [0, 9, 28, 37];
a = [1, 8, 37, 50];
[r, p, k] = residue(b, a)
```

4.2. Mit dem Befehl `syms` können analytische Variablen und Funktionen festgelegt werden. Operationen wie bspw. `ilaplace` liefern dann das analytische Ergebnis der inversen Transformation. Zum Vergleich wurden einmal die ursprüngliche und einmal die partialbruchzerlegte Funktion transformiert. Beides liefert ein und das selbe Ergebnis.

```
syms p; F1 = 1/(p+2)+(4+j*2.25)/(p-(-3+j*4))+(4-j*2.25)/(p-(-3-j*4));
f1 = ilaplace(F1)

syms p; F2 = (9*p^2+28*p+37)/(p^3+8*p^2+37*p+50);
f2 = ilaplace(F2)
```

4.3. Möchte man nun numerische Ergbnisse berechnen um z. B. das Erbnis graphisch darstellen zu können, muss man die Variablen durch Ensprechende Werte ersetzen. Dazu bedient man sich des `subs`-Befehls. Näheres kann der Hilfe entnommen werden.

```
t = 0 : .1 : 3;
f_plot = subs(f1, t);
plot(t, f_plot);
xlabel('t');
ylabel('Inverse Laplace-Transformierte');
```