v1.2

Matlab-Übungen zu Deterministische Signale und Systeme Musterlösung zur 6. Übung

1. Aufgabe

1.1. Stellen Sie die Ausgangsgleichung eines Systems auf, dessen Ausgangswert die halbe Summe des aktuellen und des letzten Eingangswertes ist.

1.2. Zeichnen Sie ein Blockschaltbild für das im letzten Aufgabenteil bestimmte System.

1.3. Implementieren Sie eine Matlab-Funktion `mittelwert2`, die 2 oder 3 Werte übergeben bekommt und die die Antwort des oben bestimmten Systems auf eine Eingangsfolge berechnet. Dabei können Sie annehmen, daß die Eingangsfolge nur für $n \geq 0$ existiert und für alle Werte $n = (0, 1, \ldots, N-1)$ gegeben ist. Implementieren Sie das Verhalten mithilfe einer for-Schleife.

Wenn der Funktion zwei Argumente übergeben werden, so handelt es sich dabei um den Vektor mit den Zeitwerten n und den Vektor mit den Signalwerten x. Sind 3 Werte gegeben, so wird zusätzlich zuerst ein Vektor mit den Anfangswerten (die Sie sicher brauchen werden) übergeben.

Als Anfangswerte sind Werte zu verstehen, die Sie benötigen, um schon für den ersten Zeitschritt eine Ausgabe zu erzeugen, obwohl noch nicht genug Werte vorhanden sind, um den Mittelwert zu bilden. Sie können das beispielsweise implementieren, indem Sie ein Schieberegister für die Eingangswerte bilden und dort am Anfang schon Werte geladen haben.

1.4. Gegeben seien ein Vektor mit Zeitwerten

$$n = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{pmatrix}$$

und die Eingangsfolgen

$$x_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$
$$x_2 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

und

$$x_3 = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

M. Sc. M. Trinh Hoang (S108/207); K. Cui
Prof. Dr.-Ing. A. Klein
Prof. Dr.-Ing. M. Pesavento

Wintersemester 2018/19

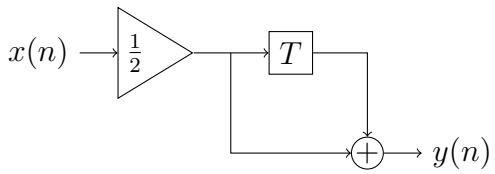

Abbildung 1: Blockschaltbild zum Mittelwertfilter.

Bestimmen Sie mit Hilfe der soeben geschriebenen Funktion die Antworten y_1 bis y_3 des Systems auf die Eingansfolgen x_1 bis x_3. Bei x_2 sollten Sie zusätzlich einen Durchlauf mit einem Anfangswert von 1 machen.

1.5. Schreiben Sie eine weitere Funktion `mittelwert3`, die ein System implementiert, das analog zu dem bisher in dieser Aufgabe betrachteten System funktioniert, aber einen weiteren vergangenen Eingangswert in die Mittelwertbildung mit einbezieht.

1.6. Können Sie aus einer der Ausgangsfolgen schließen, um was für ein Filter es sich bei `mittelwert2` und `mittelwert3` handelt (Hoch- oder Tiefpass)? Wenn ja, aus welcher?

1.7. Um wieviel Schritte »verzögern« die beiden Filter den »Mittelpunkt« des gegebenen Eingangsimpulses x_3?

1.8. Finden Sie ein Filter, das das gegenteilige Verhalten zu dem gebebenen im Frequenzbereich zeigt und dabei auch 2 Eingangswerte verarbeitet (Also ein Hoch- statt ein Tiefpass bzw. umgekehrt, je nachdem, was für ein Filter bisher betrachtet wurde. Es ist dabei nicht wichtig, daß das Filter genau invers zu dem anderen ist, es soll nur die grundlegende Charakteristik zeigen.). Implementieren Sie auch dieses Filter und testen Sie es mit den gegebenen Eingangsfolgen.

Lösung

1.1. Die Gleichung

$$y(n) = \frac{1}{2}(x(n) + x(n-1))$$

realisiert das beschriebene System.

1.2. Das entsprechende Blockschaltbild ist in Abbildung 1 zu sehen.

1.3. Durch den Einsatz von `varargin` in der Funktiondefinition

```
function y = mittelwert2(varargin)
```

kann eine variable Anzahl an Parametern verarbeitet werden.

Nach Überprüfung der Anzahl der Parameter mit

```
if length(varargin) == 2
    a = 0;
    t = varargin{1};
```

```
    x = varargin{2};
elseif length(varargin) == 3
    a = varargin{1};
    t = varargin{2};
    x = varargin{3};
else
    error('Invalid number of input arguments!');
end
```

und entsprechendem Zuweisen von `a`, was die Anfangswerte speichert, kann nun die Ausgangsfolge berechnet werden. Hierfür bietet sich z. B. eine `for`-Schleife an. Um die Anfangswerte zu berücksichtigen, wird ein Hilfsvektor für die Eingangswerte mit den vorangestellten Anfangswerten gebildet, über den dann iteriert wird:

```
xw = [a, x];
```

Die Anzahl der Anfangswerte wird berechnet, obwohl sie bekannt ist und auch direkt verwendet werden könnte:

```
la = length(a);
```

So ist es aber einfacher, den Code (in einer späteren Aufgabe) wiederzuverwenden.

Die `for`-Schleife für die Ausgabe sieht dann schließlich so aus:

```
for k = 1 : length(t)
    y(k) = 1/2 * ( xw(k+la) + xw(k+la-1) );
end
```

1.4. Mit dem Code

```
t =  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9];
x1 = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0];
x2 = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1];
x3 = [0, 0, 0, 1, 1, 1, 1, 0, 0, 0];

x = x1;
figure;
hold on;
stem(t, x, '+');
stem(t, mittelwert2(t, x), 'r')
hold off;
xlabel('n'); legend('x(n)', 'y(n)');
title('Systemantwort auf Delta-Impuls');

x = x2;
figure;
hold on;
stem(t, x, '+');
stem(t, mittelwert2(t, x), 'r');
hold off;
```

```
xlabel('n'); legend('x(n)', 'y(n)');
title('Systemantwort␣auf␣Einsenfolge');

x = x2;
figure;
hold on;
stem(t, x, '+');
stem(t, mittelwert2(1, t, x), 'r');
hold off;
xlabel('n'); legend('x(n)', 'y(n)');
title('Systemantwort␣auf␣Einsenfolge␣mit␣Anfangswert␣1');

x = x3;
figure;
hold on;
stem(t, x, '+');
stem(t, mittelwert2(t, x), 'r');
hold off;
xlabel('n'); legend('x(n)', 'y(n)');
title('Systemantwort␣auf␣Rechteckimpuls');
```

wird die Antwort des Systems auf die drei Eingangsfolgen berechnet.

1.5. Da die Funktionsweise von `mittelwert3` der von `mittelwert2` entspricht, habe ich auf ein Niederschreiben der Funktion hier verzichtet. Im Wesentlichen ändert sich nur die Anzahl der gegebenen Anfangswerte (`a = [0 0]`), falls diese nicht übergeben werden und die Mittelwertbildung selbst, die nun durch die Zeile

```
y(k) = 1/3 * ( xw(k+la) + xw(k+la-1) + xw(k+la-2) );
```

realisiert ist.

Auch der Test mit den gegebenen Eingangsfolgen verläuft analog.

1.6. In der Antwortfolge auf den Rechteckimpuls kann man beispielsweise sehen, dass die Sprungstellen in eine Rampe geglättet werden, je nach Anzahl der Filterkoeffizienten ist diese Rampe mehr oder weniger lang. Es handelt sich also um ein Tiefpassfilter. Zudem ist die Berechnungsvorschrift für y ein Mitelwert, d.h. eine Summe. Die Funktion hat einen integrierenden Charakter, was ebenfalls ein Indiz für einen Tiefpass ist.

1.7. Das Filter mit zwei Koeffizienten verzögert um einen halben Takt, das mit drei Koeffizienten um einen ganzen.

1.8. Ein entsprechendes Filter kann realisiert werden indem man ein differenzierendes Verhalten erzeugt, z.B. mit der Gleichung

$$y(n) = \frac{1}{2}(-x(n) + x(n-1)).$$

Dieses Filter hat Hochpass-Charakter. Gegenüber `mittelwert2.m` ändert sich nur die Berechnung der Ausgangsfolge in der `for`-Schleife. Die Überprüfung der Funktion funktioniert dann analog zu Aufgabe 1.4.

2. Aufgabe

2.1. Schreiben Sie eine Funktion `ztrf`, die die z-Transformierte eines gegebenen Systems berechnet. Gehen Sie davon aus, dass nur kausale Systeme vorkommen und dass der Ausgang nur von den Eingangswerten abhängt. Die Funktion bekommt einen Vektor mit Zeitindizes `n`, einen Vektor `x` mit dazu passenden Funktionswerten, einen Vektor `re_z` mit den zu berechnenden Stellen auf der reellen Achse des z-Bereichs und einen Vektor `im_z` mit den zu berechnenden Stellen auf der imaginären z-Achse übergeben.

Ein weiteres optionales Argument soll dafür benutzt werden, die berechnete Funktion ab einem gewissen Betrag abzuschneiden, falls sie gegeben ist. benutzen Sie hierfür das Schlüsselwort `varargin`. Dieses Vorgehen ist nötig, da eventuelle Polstellen bei einem Plot zu einer schlecht interpretierbare Skalierung der z-Achse führen.

Die Funktionsdefinition sieht also so aus: `function X = ztrf(n,x,re_z,im_z,varargin)`.

`X` ist also eine Matrix, die zu jeder Kombination von `re_z` und `im_z` den entsprechenden Wert der z-Transformierten enthält.

Die Funktion soll die Transformierte nicht nur berechnen, sondern auch mittels eines `surf`-Plots plotten. Dabei soll die »Grundstellung« des Plots eine Ansicht von oben sein, so dass Sie die Höhe der Funktion nur über die Farbe ablesen können. Blenden Sie als Ablesehilfe einen Farbbalken (»colorbar«) ein (dafür gibt es einen entsprechenden Button im Plot-Fenster). Außerdem soll die Skalierung der x- und y-Achse gleich sein, was Sie durch den Aufruf von `axis equal` nach dem Plot erreichen.

2.2. Gehen Sie von dem Mittelwertfilter mit zwei Koeffizienten aus der ersten Aufgabe aus. Transformieren Sie dieses Filter (von Hand) in den z-Bereich und bestimmen Sie Pole und Nullstellen.

Lassen Sie sich von Ihrer Funktion `ztrf` die Transformierte des Filters im Bereich $\Re(z) \in [-2{,}1]$ und $\Im(z) \in [-1{,}1]$ darstellen (wählen Sie dabei für die Achsen selbst eine geeignete Auflösung).

Überprüfen Sie nun das Ergebnis mit den von Ihnen berechneten Polen und Nullstellen.

2.3. Benutzen Sie die Matlab-Funktion `zplane`, um die von Ihnen bestimmten Pole und Nullstellen zu überprüfen.

Die Funktion `zplane` bekommt dabei zwei Parameter `b` und `a` übergeben, wobei `b` einen Vektor mit den Koeffizienten des Zählerpolynoms und `a` einen Vektor mit den Koeffizienten des Nennerpolynoms enthält. Dabei sind diese Vektoren nach folgendem Schema zu bilden:

$$H(z) = \frac{b(1) + b(2)z^{-1} + b(3)z^{-2} + \cdots}{a(1) + a(2)z^{-1} + a(3)z^{-2} + \cdots}$$

$b(2)$ meint hier das zweite Element des Vektors `b` (für `a` entsprechend).

2.4. Berechnen Sie analog zu den vorangegangenen Teilaufgaben die z-Transformierte des Systems aus Abbildung 2 und überprüfen Sie auch hier die Lage der Pole und Nullstellen mit der Funktion `zplane`.

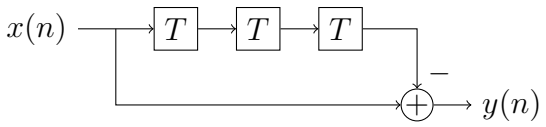

Abbildung 2: System im diskreten Zeitbereich.

2.5. Mit der Funktion `freqz` können Sie den Frequenzgang eine Systems nach Betrag und Phase anzeigen lassen. Um was für ein Filter es sich bei dem Mittelwertfilter handelt, haben Sie schon zuvor geklärt. Finden Sie heraus, bei welcher »normierten Frequenz« (also im Bereich zwischen 0 und 1) sich der Amplitudenwert $-3\,\mathrm{dB}$ ergibt.

Um was für ein Filter handelt es sich bei einem System mit

$$H_2(z) = -\frac{1}{2} + \frac{1}{2}z^{-1}?$$

Lösung

2.1. Die erste Zeile enthält wie immer die Funktionsdefinition, wobei hier mit der Angabe des Parameters `varargin` die Möglichkeit gegeben ist, beliebig viele[1] zusätzliche Parameter zu verarbeiten.

```
function X = ztrf(n, x, re_z, im_z, varargin)
```

Die variable `maxval` wird auf einen Dummy-Wert gesetzt

```
maxval = Inf;
```

und anschließend wird überprüft, ob der zusätzliche Parameter gesetzt wurde. Ist er gesetzt, wird `maxval` überschrieben:

```
if length(varargin) == 1
    maxval = varargin{1};
elseif length(varargin) > 1
    error('too␣many␣input␣arguments')
end
```

Sind mehr als ein Parameter übergeben worden, wird eine Fehlermeldung ausgegeben.

In der mit

```
X = zeros(length(re_z), length(im_z));
```

initialisierten Matrix werden die Funktionswerte der z-Transformierten gespeichert.

Die Transformierte selbst berechnet man für jeden Punkt im Frequenzbereich nach der Gleichung

$$X(z) = \sum_{n=0}^{N-1} x(n)z^{-n}$$

[1] Es wird wohl eine Obergrenze geben, aber leider konnten dazu keine Angaben in der Matlab-Hilfe gefunden werden.

wobei die Summe auch in einer for-Schleife gebildet wird:

```
for u = 1 : length(re_z)
    for v = 1 : length(im_z)
        for k = 1 : length(n)
            X(u,v) = X(u,v) + x(k).*(re_z(u)+j*im_z(v)).^(-n(k));
        end
    end
end
```

Mit weniger Schleifen und deutlich effizienter läßt sich die Berechnung unter Verwendung der Funktion `bsxfun` durchführen (mehr Informationen zur Funktion in der MATLAB Hilfe):

```
for k = 1:length(n)
    X = X + x(k) .* bsxfun(@plus, re_z(:), 1i*im_z(:)') .^ (-n(k));
end
```

Nun wird die Funktion noch gemäß dem zusätzlichen Parameter beschnitten, falls ein solcher gegeben war:

```
if maxval ~= Inf
    [a, b] = find(abs(X) > maxval);
    X(a, b) = sign(X(a, b)) .* maxval;
end
```

Schließlich plottet

```
[RE_Z, IM_Z] = meshgrid(re_z, im_z);
surf(RE_Z, IM_Z, abs(X).');
xlabel('Re(z)');
ylabel('j␣Im(z)');
axis equal;
view([0 90]);
colorbar;
```

den Betrag der Transformierten.

2.2. Aus der Filterfunktion

$$y(n) = \frac{1}{2}(x(n) + x(n-1))$$

ergibt sich die Transformierte aus

$$Y(z) = \frac{1}{2}(X(z) + z^{-1}X(z))$$

zu

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{2} + \frac{1}{2}z^{-1} = \frac{z+1}{2z}.$$

Diese Funktion hat eine Nullstelle bei $z = -1$ und einen Pol bei $z = 0$. Um die Transformierte in Matlab zu berechnen, legt man mit

```
t = 0 : 1;
re_z = -2 : .01 : 1;
im_z = -1 : .01 : 1;
```

zuerst Zeit- und Frequenzachsen und einen Vektor mit den Koeffizienten der Impulsantwort

```
h = [.5, .5];
```

an.

Das Ergebnis der Transformation zeigt man mit

```
ztrf(t, h, re_z, im_z, 100);
```

an. In dem Plot kann man deutlich die Pol- und die Nullstelle erkennen.

2.3. Die Funktion `zplane` bekommt Zähler- und Nennerpolynom übergeben, was bei diesem Beispiel recht einfach ist. Das Zählerpolynom aus Gleichung 2.2 ergibt den Vektor

```
b = [.5, .5]; % Zaehlerpolynom
```

und das Nennerpolynom den Vektor

```
a = [1]; % Nennerpolynom
```

Mit

```
zplane(b, a);
```

bekommt man von Matlab die Pol- und Nullstellen angezeigt.

2.4. Die Zeichnung zeigt ein System mit der Gleichung

$$y(n) = x(n) - x(n-3),$$

dessen Transformierte

$$H(z) = 1 - z^{-3} = \frac{z^3 - 1}{z^3}$$

ist. Damit ergibt sich ein dreifacher Pol bei $z = 0$ und die Nullstellen liegen auf einem Einheitskreis in der z-Ebene bei 0°, 120° und 240°.

Um das mit Matlab zu überprüfen, wählen wir einen geeigneten Frequenzbereich und bilden die Vektoren mit Zeitachse und Filterkoeffizienten

```
t = 0 : 3;
h = [1, 0, 0, -1];
re_z = -2 : .1 : 2;
im_z = -2 : .1 : 2;
```

und lassen uns die Transformierte anzeigen. Diesmal schneiden wir die Transformierte bei dem Wert 10 ab, um die Nullstellen besser sehen zu können:

```
ztrf(t, h, re_z, im_z, 10);
```

Auch dieses Ergebnis überprüfen wir mit Matlab:

```
b = [1, 0, 0, -1]; % Zaehlerpolynom
a = [1]; % Nennerpolynom

figure;
zplane(b, a);
```

2.5. Aus den mit

```
b = [.5, .5]; % Zaehlerpolynom
a = [1]; % Nennerpolynom

figure;
freqz(b, a);
```

geplotteten Kurven liest man ab, dass sich $-3\,\mathrm{dB}$ bei der normierten Frequenz $\Omega = 0{,}5$ ergeben.

Mit

```
b = [-.5, .5]; % Zaehlerpolynom
a = [1]; % Nennerpolynom

figure;
freqz(b, a);
```

findet man heraus, dass es sich bei dem System $H_2(z)$ um einen Hochpass handelt.

3. Aufgabe

3.1. Schreiben Sie eine Funktion `stabiz`, die die Stabilität und Kausalität von Systemen im z-Bereich überprüfen soll.

Die Funktion soll hier erst auf Kausalität prüfen und nur im Falle eines kausalen Systems auch die Stabilität überprüfen.

Stellen Sie hierfür zuerst die Bedingungen für Stabilität und Kausalität unter den genannten Voraussetzungen auf.

3.2. Implementieren Sie nun die Funktion `stabiz`, die ein Zähler- und ein Nennerpolynom übergeben bekommt und eine Bildschirmausgabe mit dem Ergebnis der Überprüfung auf Kausalität und Stabilität generiert.

3.3. Überprüfen Sie die folgenden Systeme auf Stabilität und Kausalität. Wenden Sie die

Kriterien zur Kontrolle auch von Hand an.

$$H_1(z) = \frac{1 + 2z^{-1}}{7 - 4z^{-1} + 3z^{-5}}$$
$$H_2(z) = \frac{1 + z^{-1} + z^{-2} + z^{-3}}{1 - 2z^{-1}}$$
$$H_3(z) = \frac{1 + 2z^{-1}}{-4z^{-1} + z^{-6}}$$

3.4. Erweitern Sie die Funktion `stabiz` so, dass im Falle eines nicht-kausalen Systems dieses durch eine entsprechende Verzögerung in ein kausales System überführt und dann auf Stabilität untersucht wird.

Lösung

3.1. Die Bedingung für die Stabilität eines kausalen Systems im z-Bereich ist, dass alle Pole im Einheitskreis liegen müssen. Um das in einem Matlab-Programm zu überprüfen, muss für alle N Pole $z_p^{(n)}$ des Systems die Bedingung

$$\left|z_p^{(n)}\right| < 1, \quad n \in \{1, 2, \ldots, N\}$$

überprüft werden.

Betrachtet man die Differenzengleichung eines Systems, ergibt sich die allgemeine Form

$$a_0 y(n) + a_1 y(n-1) + \cdots = \underbrace{c_K x(n+K) + \cdots + c_1 x(n+1)}_{\text{»nichtkausaler Anteil«}} + b_0 x(n) + b_1 x(n-1) + \ldots$$

wobei K nichtkausale Werte existieren. Bildet man die z-Transformation dieses Systems, ergibt sich

$$Y(z)(a_0 + a_1 z^{-1} + \ldots) = X(z)(c_k z^K + \cdots + c_1 z + b_0 + b_1 z^{-1} + \ldots).$$

Daraus folgt die Übertragungsfunktion

$$H(z) = \frac{Y(z)}{X(z)} = \frac{c_k z^K + \cdots + c_1 z + b_0 + b_1 z^{-1} + \ldots}{a_0 + a_1 z^{-1} + \ldots}$$

Um die Übertragungsfunktion auf die Form

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots}{a_0 + a_1 z^{-1} + a_2 z^{-2} + \ldots}$$

zu bringen, muss man Zähler- und Nennerpolynom mit z^{-K} multiplizieren. Daraus ergibt sich die Form

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots}{0 + 0z^{-1} + \cdots + a_0 z^{-K} + a_1 z^{-(K+1)} + \ldots}.$$

Hier sieht man, dass immer, wenn ein nichtkausales System vorliegt, $a_0 = 0$ ist. Das kann man in Matlab überprüfen, da a_0 der erste Wert des Vektors `a` ist.

3.2. Die Vorüberlegungen sind nun leicht in ein Matlab-Programm umzusetzen. Nach dem Kopf der Funktion

```
function stabiz(b, a)
```

folgt die Überprüfung auf Kausalität:

```
if a(1) == 0
    disp(['The␣given␣system␣is␣not␣causal!' ...
          '␣Stability␣check␣will␣be␣omitted!'])
    return;
end
```

Um die Pole der Funktion zu bestimmen, wird das Nennerpolynom zuerst auf die richtige Form erweitert:

```
g = 0;
if length(b) > length(a)
    g = length(b) - length(a);
end

ag = [a, zeros(1, g)];
p = roots(ag);
```

Nun wird jede Polstelle mit dem Stabilitätskriterium untersucht und falls eine »durchfällt« am Ende eine entsprechende Ausgabe erzeugt:

```
stab = true;
for n = 1 : length(p)
    if abs(p(n)) >= 1
        stab = false;
    end
end

if stab == false
    disp('The␣given␣system␣is␣not␣stable!')
else
    disp('The␣given␣System␣is␣causal␣and␣stable!')
end
```

3.3. Nach Eingabe von

```
disp('H1');
b1 = [ 1,  2 ];
a1 = [ 7, -4, 0, 0, 0, 3 ];

stabiz(b1, a1);

disp('H2');
b2 = [ 1, 1, 1, 1 ];
a2 = [ 1, -2 ];
```

```
stabiz(b2, a2);

disp('H3');
b3 = [ 1,  2 ];
a3 = [ 0, -4, 0, 0, 0, 3 ];

stabiz(b3, a3);
```

erweist sich H_2 als instabil und H_3 als nicht kausal, was an der Ausgabe

```
H1
The given System is causal and stable!
H2
The given system is not stable!
H3
The given system is not causal! Stability check will be omitted!
```

ablesen kann. Dass das System H_3 nicht kausal ist, sieht man auch direkt, da $a_0 = 0$ ist. Etwas schwieriger gestaltet sich die Untersuchung von H_2. Um die Pole zu bestimmen, muß der Bruch erst mit z^3 erweitert werden. Daraus ergibt sich

$$H_2(z) = \frac{z^3 + z^2 + z + 1}{z^3 - 2z^2} = \frac{z^3 + z^2 + z + 1}{z^2(z-2)},$$

man sieht also, dass das System einen doppelten Pol bei $z = 0$ und einen Pol bei $z = 2$ hat, letzterer fällt beim Stabilitätskriterium also durch.

3.4. Jedes nicht-kausale System lässt sich zu einem kausalen machen, indem man Verzögerungsglieder »einbaut«, die dafür sorgen, dass der Eingang so lange verzögert wird, bis die entsprechenden Werte zur Verfügung stehen. Benötigt ein System also den Eingabewert, der einen Schritt »in der Zukunft« liegt, verzögert man den Eingang um einen Zeitschritt und kann nun mit diesem Wert arbeiten.

Eine Verzögerung des Eingangs um einen Zeitschritt entspricht im Frequenzbereich einer Multiplikation des Zählerpolynoms mit z^{-1} bzw. einer Multiplikation des Nennerpolynoms mit $\frac{1}{z^{-1}}$. Die neun Funktion `stabiz2` bekommt die gleichen Werte wie die Funktion `stabiz` übergeben:

```
function stabiz2(b, a)
```

Wenn das übergebene System nicht kausal ist, werden nun im Vektor `a` die führenden Nullen entfernt:

```
if a(1) == 0
    disp('The␣given␣system␣is␣not␣causal!')

    ccount = find(a, 1) - 1;
    disp(['The␣input␣will␣be␣delayed␣by␣' num2str(ccount) '␣steps␣' ...
    'and␣the␣stability␣check␣will␣be␣performed␣for␣this␣new␣' ...
```

```
    'system!'])
    a = a((ccount+1):end);
end
```

Der Rest der Funktion arbeitet analog zu `stabiz`:

```
% finding the roots of the denominator polynomial
g = 0;
if length(b) > length(a)
    g = length(b) - length(a);
end

ag = [a, zeros(1, g)];
p = roots(ag);

% checking for stability
stab = true;
for n = 1 : length(p)
    if abs(p(n)) >= 1
        stab = false;
    end
end

if stab == false
    disp('The given system is not stable!')
else
    disp('The given system is stable!')
end
```

Überprüft man die Funktion $H_3(z)$ mit dieser neuen Funktion, so bekommt man durch die Ausgabe

```
The given system is not causal!
The input will be delayed by 1 steps and the stability
check will be performed for this new system!
The given system is stable!
```

gezeigt, dass das System mit verzögertem Eingang stabil ist. Überprüft man

$$H_4 = \frac{1 + z^{-1} + z^{-2} + z^{-3}}{z^{-1} - 2z^{-2}}$$

erkennt man durch

```
The given system is not causal!
The input will be delayed by 1 steps and the stability
check will be performed for this new system!
The given system is not stable!
```

dass dieses System mit verzögertem Eingang nicht stabil ist.