# Handleiding MATLAB/Simulink voor REG10

Ivo Grondman

## Inhoudsopgave



## 1 Inleiding

MATLAB/Simulink is een zeer uitgebreid softwarepakket van MathWorks waarmee talloze problemen uit verschillende disciplines kunnen worden aangepakt. Dit document is bedoeld als een (zeer) korte inleiding tot het gebruik van MATLAB/Simulink ten behoeve van het vak Regeltechniek (ELEREG10) en zal daarom slechts de functionaliteit behandelen die relevant is voor dit vak. Voor mensen die meer willen weten van het softwarepakket heeft MathWorks zelf een aantal bronnen beschikbaar gesteld:

- MATLAB Primer
  https://www.mathworks.com/help/pdf_doc/matlab/getstart.pdf
- Simulink Onramp
  https://nl.mathworks.com/matlabcentral/fileexchange/69056-simulink-onramp

De eerste paragrafen zullen beschrijven hoe MATLAB gebruikt kan worden voor het definiëren van overdrachtsfuncties, het plotten van responsies en het ontwerpen van een regelaar met behulp van grafische weergaven. De laatste paragraaf laat zien hoe e.e.a. ook in Simulink gedaan kan worden. De voorbeelden in dit document laten vaak alleen zien hoe een functie in zijn meest eenvoudige vorm gebruikt kan worden. Voor meer uitgebreide informatie over een functie kan de gebruiker altijd de documentatie van de functie raadplegen door gebruik van het `doc` commando.

**Voorbeeld 1.** Een gebruiker wil meer weten van het `feedback` commando:

```
>> doc feedback
```

Ook kan de documentatie doorzocht worden op bepaalde termen met behulp van het `docsearch` commando.

**Voorbeeld 2.** Een gebruiker wil in de documentatie zoeken naar functionaliteit die te maken heeft met Bode plots:

```
>> docsearch bode
```

Naast de voorbeelden in deze handleiding staan er ook in het boek wat gebruikt wordt bij het vak talloze voorbeelden waarin gebruik wordt gemaakt van MATLAB voor het berekenen van antwoorden. Tevens staat in Appendix C van het boek een wat langere lijst met relevante MATLAB commando's met daarbij een korte beschrijving van wat deze doen.

# 2 Overdrachtsfuncties

In de eerste weken van het vak is gebleken dat alle berekeningen in regeltechniek worden gedaan aan de hand van zogenaamde overdrachtsfuncties. Ook in MATLAB zal er dus begonnen moeten worden met het definiëren van een overdrachtsfunctie. In deze paragraaf wordt eerst beschreven hoe open-lus systemen gedefinieerd worden. Vervolgens wordt er getoond hoe het programma ingezet kan worden om gesloten-lus overdrachten te berekenen.

## 2.1 Open-loop

Tijdens het vak regeltechniek is er kennis gemaakt met verschillende vormen van een overdrachtsfunctie. Een van de gangbare vormen is die van een deling van polynomen:

$$H(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_0}{a_n s^n + a_{n-1} s^{n-1} + \cdots + a_0}$$

Als dit de vorm is waarin een overdracht gegeven is, dan is de functie `tf` het meest geschikt om te gebruiken, welke als argumenten de coëfficiënten van teller en noemer meekrijgt.

**Voorbeeld 3.** De overdrachtsfunctie

$$H(s) = \frac{5s^2 + 4s - 10}{s^3 + 4s^2 + 10s}$$

wordt in MATLAB gedefinieerd met:

```
>> H = tf([5 4 -10],[1 4 10 0])

H =

   5 s^2 + 4 s - 10
  ------------------
  s^3 + 4 s^2 + 10 s

Continuous-time transfer function.
```

Als een overdracht in gefactoriseerd is in zijn nulpunten en polen is het niet nodig om deze eerst uit te schrijven alvorens `tf` te gebruiken. Het commando `zpk` kan dan beter gebruikt worden.

**Voorbeeld 4.** De overdrachtsfunctie

$$H(s) = \frac{5(s+2)}{(s+3)(s+10)}$$

wordt in MATLAB gedefinieerd met:

```
>> H = zpk(-2,[-3 -10],5)

H =

    5 (s+2)
  ------------
  (s+3) (s+10)

Continuous-time zero/pole/gain model.
```

Als de voorgaande commando's niet goed te gebruiken zijn door de vorm waarin de overdrachtsfunctie is gegeven, is het ook mogelijk om overdrachtsfuncties in een wat vrijere vorm te definiëren, door eerst een variabele `s` aan te maken waarmee vervolgens een expressie kan worden opgebouwd.

**Voorbeeld 5.** De overdrachtsfunctie

$$H(s) = \frac{(2s+3)(7s+4)}{(s+1)(7s+9)(10s+50)}$$

is niet direct in `tf` te gebruiken omdat teller en noemer dan eerst uitgeschreven moeten worden om de coëfficiënten te achterhalen. Ook `zpk` biedt geen soelaas, omdat de losse nulpunten, polen en versterking niet handig zijn af te lezen. Daarom wordt er voor de volgende aanpak gekozen:

```
>> s = tf('s');
>> H = ((2*s+3)*(7*s+4))/((s+1)*(7*s+9)*(10*s+50))

H =

        14 s^2 + 29 s + 12
  -------------------------------
  70 s^3 + 510 s^2 + 890 s + 450

Continuous-time transfer function.
```

Tot slot is er nog de mogelijkheid om systemen te beschrijven in hun state-space notatie, waarvoor het commando `ss` gebruikt wordt.

**Voorbeeld 6.** Het systeem met state-space omschrijving

$$\dot{x} = \begin{bmatrix} -7 & -12 \\ 1 & 0 \end{bmatrix} x + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u$$
$$y = \begin{bmatrix} 1 & 2 \end{bmatrix} x - 4u$$

wordt in MATLAB gedefinieerd met:

```
>> A = [-7 -12; 1 0];
>> B = [1;0];
>> C = [1 2];
>> D = -4;
>> H = ss(A,B,C,D)

H =
```

```
  A =
        x1   x2
   x1   -7  -12
   x2    1    0

  B =
       u1
   x1   1
   x2   0

  C =
       x1  x2
   y1   1   2

  D =
       u1
   y1  -4

Continuous-time state-space model.
```

De commando's `tf`, `zpk` en `ss` zijn ook te gebruiken om systemen om te zetten van de ene notatie naar de andere.

**Voorbeeld 7.** De overdrachtsfunctie van het voorgaande voorbeeld wordt gevonden met:

```
>> A = [-7 -12; 1 0];
>> B = [1;0];
>> C = [1 2];
>> D = -4;
>> H = ss(A,B,C,D);
>> Htf = tf(H)

Htf =

  -4 s^2 - 27 s - 46
  ------------------
    s^2 + 7 s + 12

Continuous-time transfer function.
```

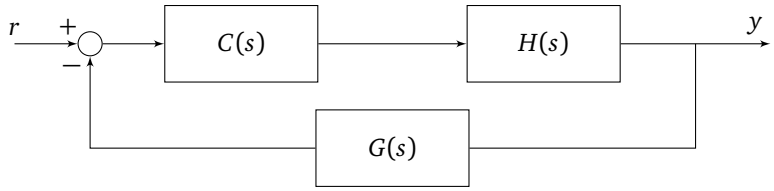
## 2.2 Closed-loop

De overdrachtsfunctie van een closed-loop systeem zoals in Figuur 1 kan in MATLAB worden berekend door middel van het commando `feedback`.

**Voorbeeld 8.** Als in Figuur 1 de overdracht $H(s)$ en de regelaar $C(s)$ gegeven zijn door

$$H(s) = \frac{1}{s(s+1)} \qquad\qquad C(s) = 91 \cdot \frac{s+2}{s+13}$$

en er geen sprake is van een overdracht in het feedback pad ($G(s) = 1$), dan wordt de overdrachtsfunctie $\frac{Y(s)}{R(s)}$ berekend met:




Figuur 1: Closed-loop systeem

```
>> C = zpk(-2,-13,91);
>> H = zpk([],[0 -1],1);
>> Hcl = feedback(C*H,1)

Hcl =

             91 (s+2)
  ---------------------------------
  (s+2.386) (s^2 + 11.61s + 76.29)

Continuous-time zero/pole/gain model.
```

Het resultaat is hier een "zero/pole/gain model", maar door `tf(Hcl)` aan te roepen zou ook een reguliere overdrachtsfunctie verkregen kunnen worden. Voor meer complexe configuraties zijn ook de commando's `parallel` en `series` beschikbaar.

# 3 Systeemeigenschappen

Als de gewenste overdrachtsfunctie eenmaal is bepaald en gedefinieerd, kunnen talloze eigenschappen van de overdracht bepaald worden door MATLAB. De volgende commando's zijn bijvoorbeeld beschikbaar:

- `damp`: dempingsfactor en natuurlijke frequenties
- `dcgain`: $H(0)$
- `margin`: versterkings- en fasemarge
- `pid` en `pidstd`: creëer PID regeleaar
- `pole`: polen berekenen
- `pzmap`: plot van nulpunten en polen in complexe vlak
- `zero`: nulpunten berekenen

**Voorbeeld 9.** Bereken de nulpunten en polen van

$$H(s) = \frac{10s^2 + 10s}{s^3 + 8s^2 + 22s + 20}$$

```
>> H = tf([10 10 0],[1 8 22 20]);
>> zero(H)
```

```
ans =

     0
    -1

>> pole(H)

ans =

  -3.0000 + 1.0000i
  -3.0000 - 1.0000i
  -2.0000 + 0.0000i
```

## 3.1 Responsies

Karaktereigenschappen als overshoot, settling time, etc. zijn het beste te halen uit de stapresponsie. Welke te verkrijgen is met het commando `step`.

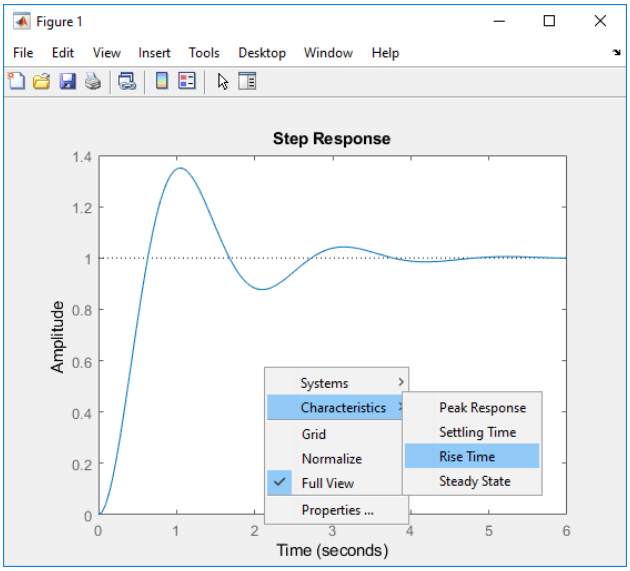
**Voorbeeld 10.** De stapresponsie van

$$H(s) = \frac{10}{s^2 + 2s + 10}$$

wordt verkregen door:

```
>> H = tf(10,[1 2 10]);
>> step(H);
```

Dit levert een figuurvenster op waarin de stapresponsie getoond wordt. Door rechts te klikken in deze plot kunnen verdere eigenschappen van de stapresponsie opgevraagd worden (zie Figuur 2).




Figuur 2: Stapresponsie van $H(s) = \frac{10}{s^2+2s+10}$.

Indien een impulsresponsie getoond moet worden kan het commando `impulse(H)` worden gebruikt.

# 4 Grafische weergaven

Tijdens het vak is er gebruik gemaakt van enkele plots, bijv. om stabiliteitsmarges en -intervallen te bepalen, zoals root locus, Bode plots en Nyquist plots.

## 4.1 Root locus

Door een overdracht $H(s)$ aan het commando `rlocus` mee te geven wordt er een figuurvenster getoond met daarin de poolbanen van het gesloten-lus systeem.

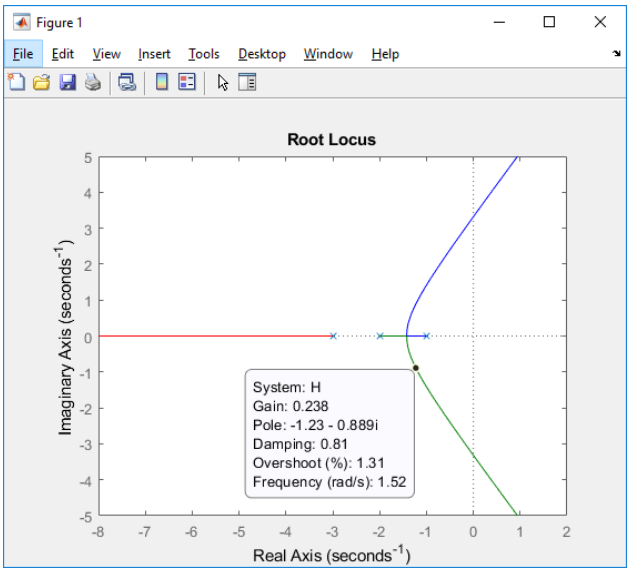
**Voorbeeld 11.** De poolbanen voor een proportioneel geregeld systeem met

$$H(s) = \frac{9}{(s+1)(s+2)(s+3)}$$

worden verkregen door

```
>> H = zpk([],[-1 -2 -3],9);
>> rlocus(H);
```

In deze plot kan op de poolbanen worden geklikt om te zien voor welke gain $K$ de polen in een bepaald punt komen te liggen. Ook worden in hetzelfde venster dan zaken als dempingsfactor en overshoot vermeld (zie Figuur 3).




Figuur 3: Root locus van $H(s) = \frac{9}{(s+1)(s+2)(s+3)}$.

## 4.2 Bode plot

Voor een Bode plot kan in het voorgaande voorbeeld het commando `bode(H)` worden aangeroepen. Het aardige van commando's als `rlocus`, `bode`, etc. is dat ze ook méér dan een argument aankunnen, wat de mogelijkheid biedt om systemen met verschillende regelaars (of zonder regelaar) met elkaar te vergelijken.

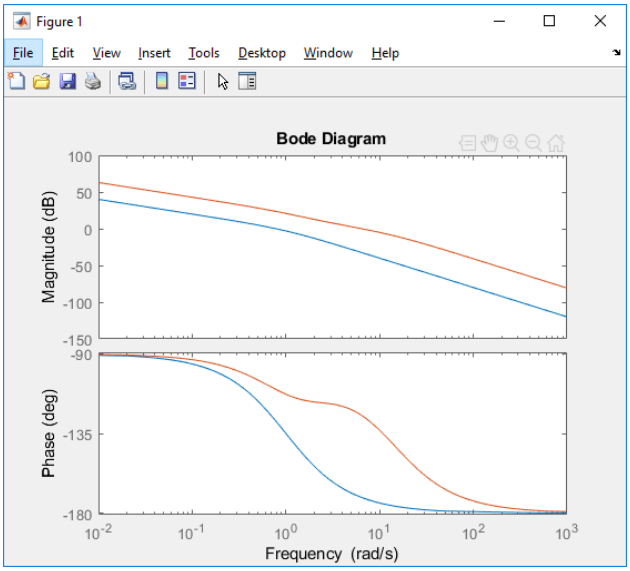
**Voorbeeld 12.** Er wordt nogmaals gekeken naar het systeem $H(s)$ en de lead compensator $C(s)$:

$$H(s) = \frac{1}{s(s+1)} \qquad\qquad C(s) = 91 \cdot \frac{s+2}{s+13}$$

Om te zien wat het effect van de lead compensator is kan het bode commando als volgt worden gebruikt:

```
>> H = zpk([],[0 -1],1);
>> C = zpk(-2,-13,91);
>> bode(H,C*H)
```

Dit levert een Bode plot op met daarin een blauwe lijn voor het systeem zónder regelaar en een rode voor het systeem mét regelaar (zie Figuur 4). Hierin is duidelijk het effect van phase lead te zien. Ook in deze figuur is het met een rechter muisklik mogelijk om eigenschappen van het systeem op te vragen.




Figuur 4: Bode plot van $H(s) = \frac{1}{s(s+1)}$ met en zonder lead compensator $C(s) = 91 \cdot \frac{s+2}{s+13}$.

## 4.3 Nyquist plot

Uiteraard biedt MATLAB ook nog de mogelijkheid om Nyquist plots te genereren. Standaard wordt er een plot gemaakt voor frequenties $\omega \in (-\infty, \infty)$, waardoor ook het Nyquist criterium snel kan worden gebruikt.

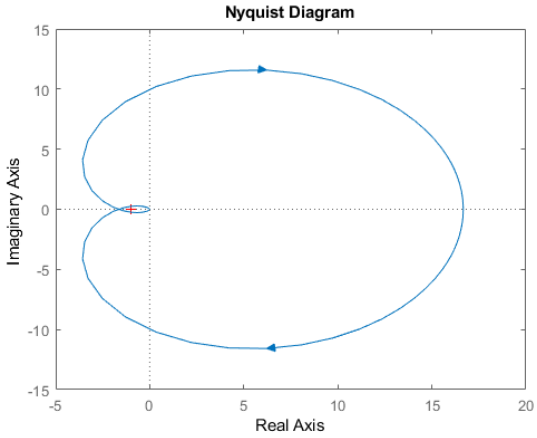
**Voorbeeld 13.** Met behulp van het Nyquist criterium kan worden bepaald of het systeem

$$H(s) = \frac{1}{(s+1)(s+2)(s+3)}$$

in een gesloten lus met proportionele regelaar $K = 100$ stabiel is.

```
>> H = zpk([],[-1 -2 -3],1);
>> K = 100;
>> nyquist(K*H);
```

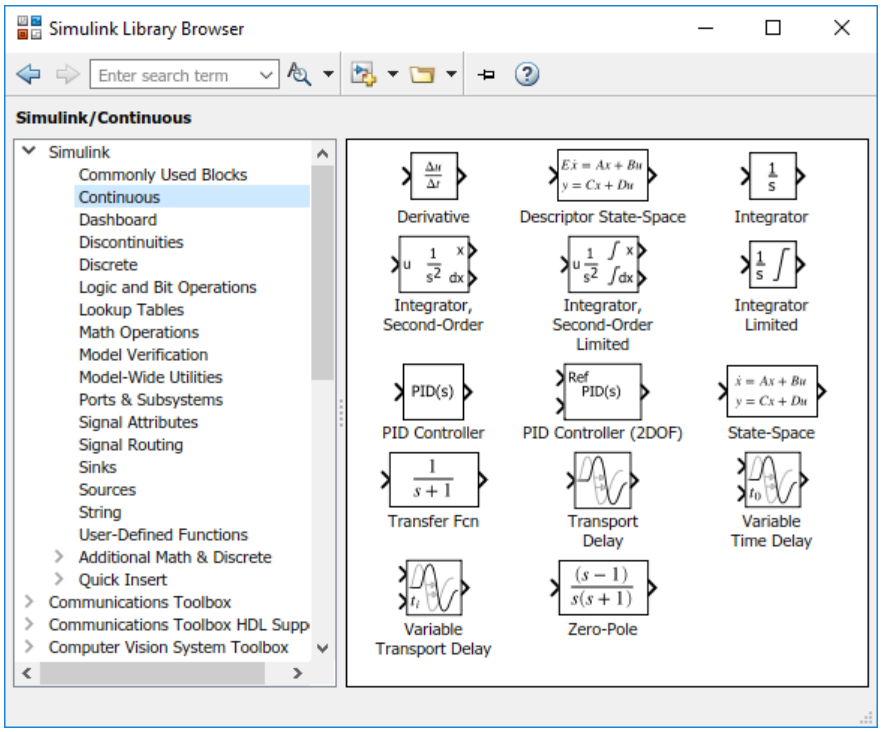
Om het Nyquist criterium toe te passen moet er worden geteld hoe vaak het punt $-1$ wordt omcirkeld en hoeveel instabiele polen er in het open-lus systeem $H(s)$ aanwezig zijn. In Figuur 5 wordt het punt $-1$ aangegeven met een rood plusteken. In totaal wordt het punt $-1$ twee maal met de klok mee omcirkeld, dus $N = 2$. Het open-lus systeem had *geen* instabiele polen, dus $P = 0$. Het Nyquist criterium levert nu op dat het aantal instabiele polen in de gesloten lus gelijk is aan $Z = N + P = 2 + 0 = 2$. Het gesloten-lus systeem is dus niet stabiel bij deze proportionele regeling.



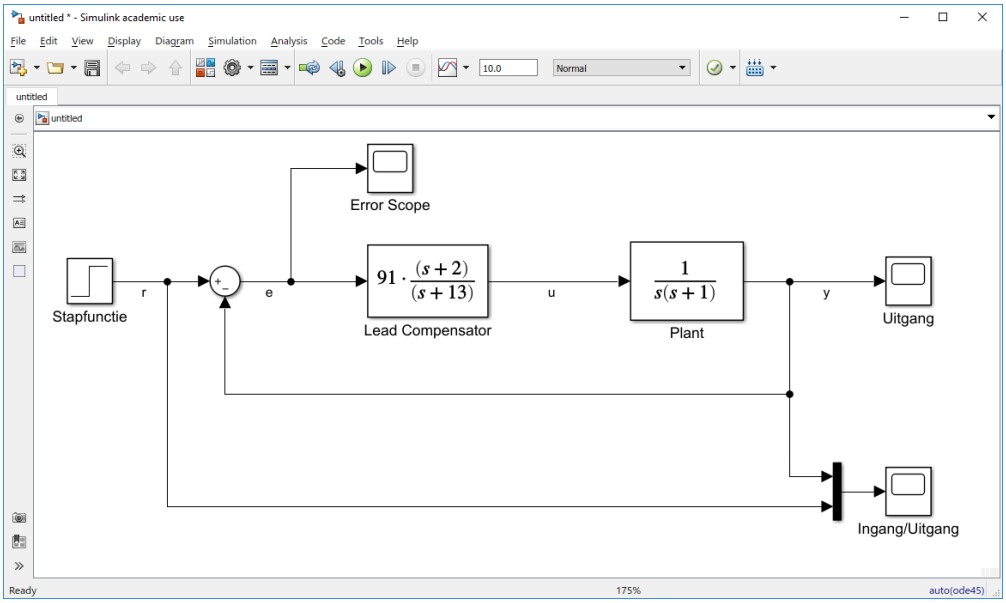

Figuur 5: Nyquist plot van $KH(s) = \frac{100}{(s+1)(s+2)(s+3)}$.

# 5 Simulink

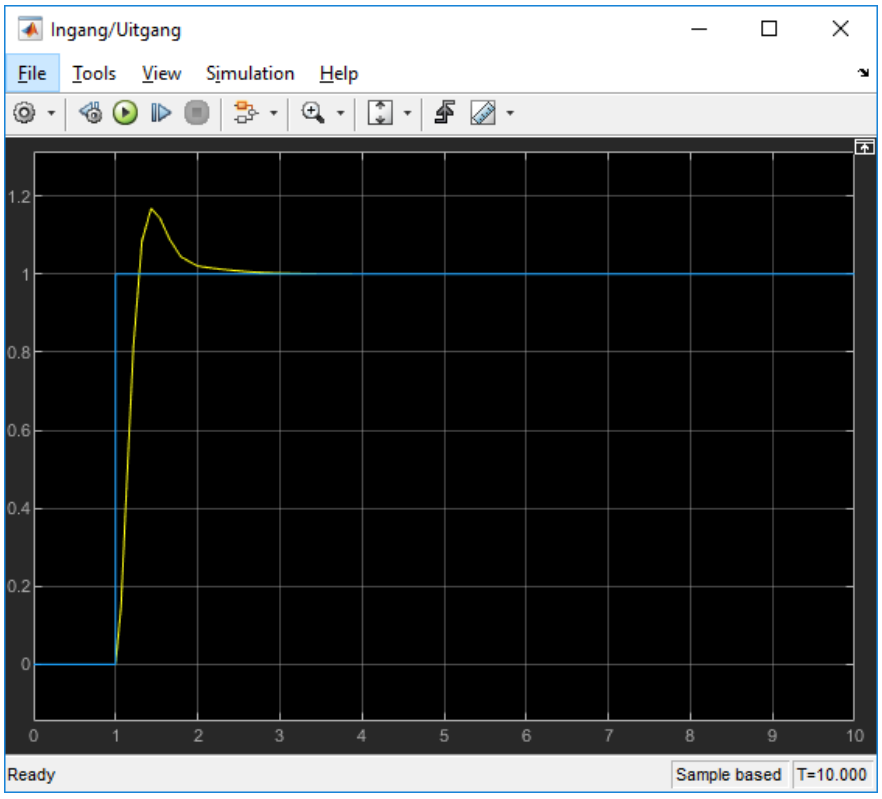
Als de schakelingen en schema's complexer worden (bijvoorbeeld door meerdere feedback loops) wordt het definiëren en simuleren van een systeem in MATLAB lastiger en is het gebruik van Simulink aan te raden, omdat het in Simulink mogelijk is om een diagram grafisch in elkaar te zetten. Een systeem als in Figuur 1 opbouwen en simuleren is in Simulink een kwestie van de juiste blokken kiezen uit de Library Browser (zie Figuur 6) en deze aan elkaar te verbinden. In de Library Browser zit aan de linkerkant een menu waarmee verschillende typen blokken kunnen worden geselecteerd. Zo zitten er in de `Sources` bibliotheek blokken die dienst kunnen doen als invoer: een stapfunctie (`Step`), een sinusoïde (`Sine Wave`), etc. Ook zijn er blokken beschikbaar waar de uitgang van een systeem naartoe kan worden gestuurd. Deze zijn te vinden onder het menu item `Sinks` en hieruit zal voornamelijk het `Scope` blok worden gebruikt.
In Figuur 7 is het eerdere voorbeeld 8 geïmplementeerd in Simulink. In dit geval wordt er een stapresponsie gesimuleerd. Er is hiervoor gebruik gemaakt van twee `Zero-Pole` blokken, een `Step` blok als ingang en een aangepast `Sum` blok om het verschil tussen referentie en uitgang te bepalen. Verder is de uitgang naar een `Scope` gestuurd. Ook is ervoor gekozen om de error $e = r - y$ weer te geven in een aparte `Scope`. Om te zien hoe goed de uitgang de referentie volgt kunnen beide signalen ook door middel van een `Mux` blok naar dezelfde `Scope` worden gestuurd om ze in één grafiek weer te geven.
Door na het opbouwen van het schema op de groene Play button te drukken (zie menubalk in Figuur 7) wordt het systeem voor 10 s gesimuleerd. Deze tijd is in te stellen in het tekstveld rechts naast de Play button. Zodra de simulatie is afgerond zijn de `Scope` blokken te openen door erop te dubbelklikken. In Figuur 8 zijn de grafieken van de Ingang/Uitgang `Scope` getoond.




Figuur 6: De Library Browser in Simulink.




Figuur 7: Een simpel feedbackshema in Simulink.




Figuur 8: De grafieken van ingang en uitgang in één Scope.