

151-0563-00 **Regelungstechnik 1** (HS 2007)**Musterlösung****Übung 3)** m-files und Simulink, Bode- und Nyquist plots

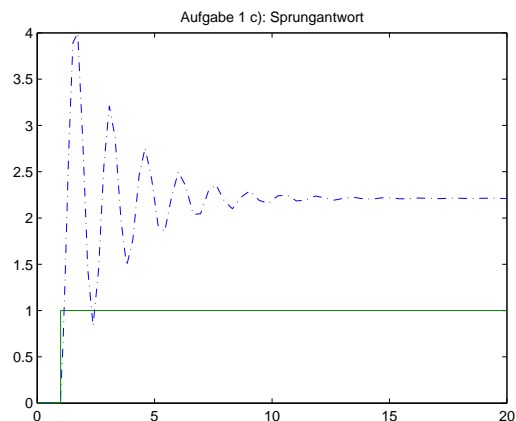
Stefan Ammann (stammann@student.ethz.ch), 12. Dezember 2007

Aufgabe 1 (Verbindung von Simulink und dem Workspace)

- a) Die komplette Musterlösung des m-files ist als 'U3muloe' gespeichert.
- b) i) Eine mögliche Lösung ist in der Datei 'U3MatlabA1bMuloe' abgebildet.
ii) Mit dem folgenden Matlab®-Code simuliert man das Modell und plottet dann die Resultate:

```
sim('U3matlabA1b',20);  
figure('Name','Aufgabe 1 c): Sprungantwort')  
plot(t,y,'-.',t,u)
```

Der Plot sieht so aus:

**Aufgabe 2 (Bode- und Nyquistdiagramme)**

- a) Mit dem Code

```
s=tf('s');  
omega_0=1  
delta=[0.05;0.1;0.2;0.33;0.5;0.7;5]  
  
f1=figure('Name','Aufgabe 2 a): Bodediagramm')  
f2=figure('Name','Aufgabe 2 a): Nyquistdiagramm')  
  
for(i=1:7)
```

```
TP2=(omega_0^2/(s^2+2*delta(i)*omega_0*s+omega_0^2));
```

```
figure(f1)
```

```
hold on;
```

```
bode(TP2);
```

```
figure(f2)
```

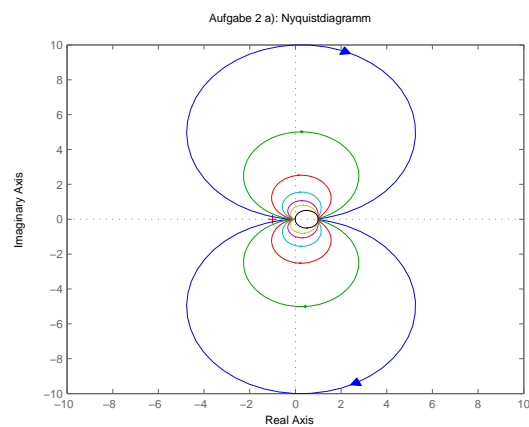
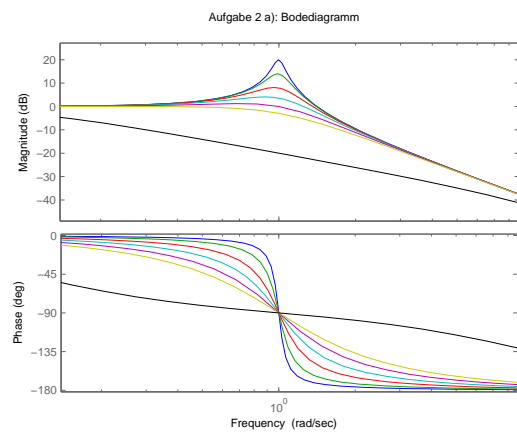
```
hold on;
```

```
nyquist(TP2)
```

```
axis([-10 10 -10 10]);
```

```
end;
```

Erstellt man die beiden Diagramme, die dann folgendermassen aussehen:



b) Der folgende Code löst die Teilaufgabe b):

```
delta=0.1;
```

```
TP2=omega_0^2/(s^2+2*delta*omega_0*s+omega_0^2)
```

```
k=1;
```

```
T=0.5;
```

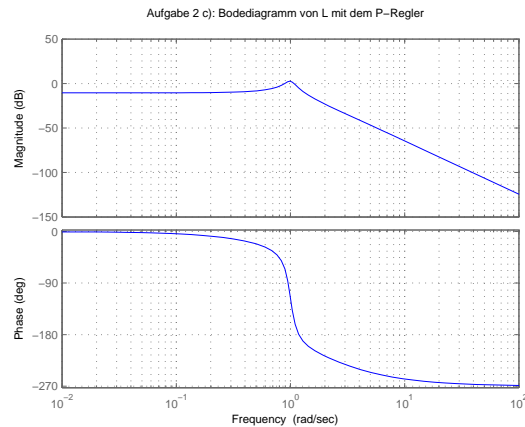
```
TP=k/(1+T*s);
```

```
plant=series(TP2,TP);
```

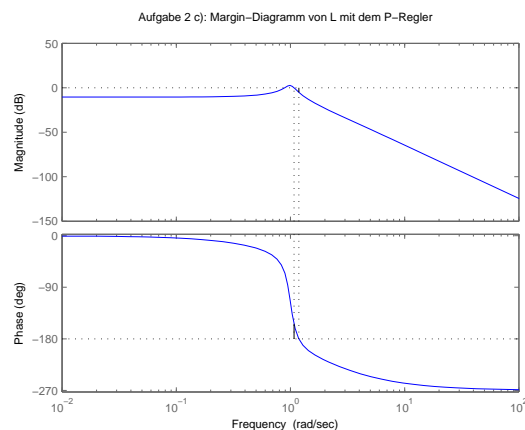
c) i) Das mit

```
k_p=0.3;
C=k_p
L_p=series(plant,C);
figure('Name','Aufgabe 2 c): Bodediagramm von L mit dem P-Regler')
bode(L_p);
grid on;
```

erstelle Bodediagramm von L_p sieht dann so aus:



ii) Um ω_c und φ abzulesen ist der Befehl `margin` etwas besser geeignet, er liefert das gleiche Bild, jedoch mit eingezeichneten Hilfslinien: Die exakten Werte für ω_c und φ ,

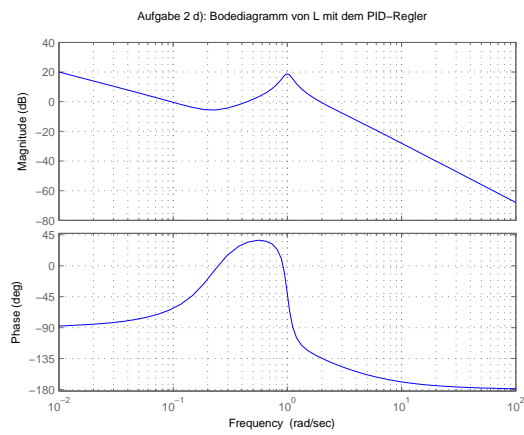


die in den Workspace geschrieben werden lauten:

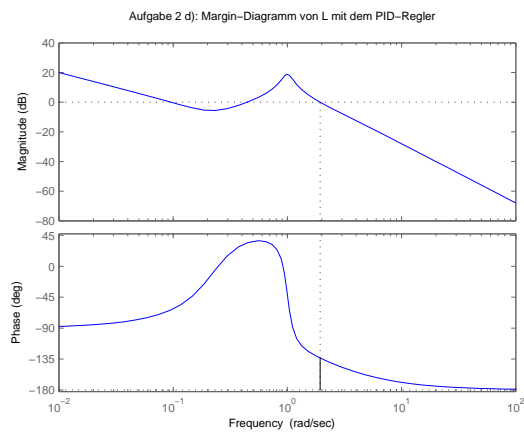
$\omega_{cp} = \omega_c = 1.0742$

$\varphi_{pm} = \varphi = 26.1405$

d) Das Bodediagramm von L_{pid} liefert:



Der mit `margin` erstellte Plot von L_{pid} für das mit dem PID-Regler geregelte System sieht dann so aus:

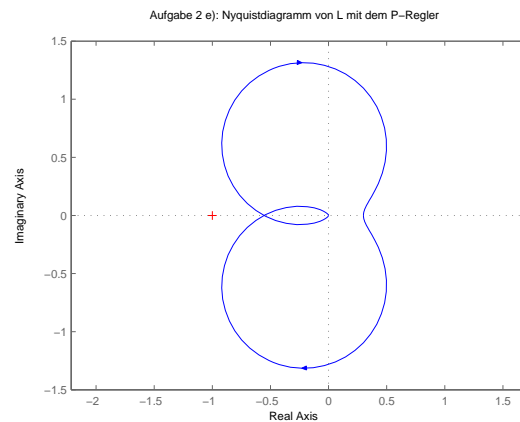


Die exakten Werte für ω_c und φ , die in den Workspace geschrieben werden lauten:

$$W_{cp_pid} = \omega_c = 1.9338$$

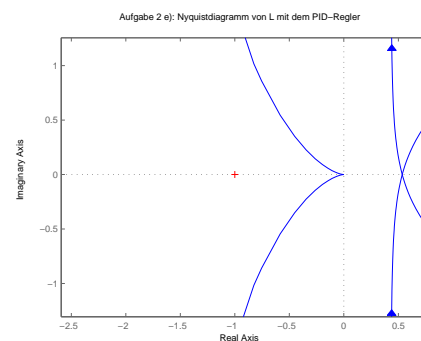
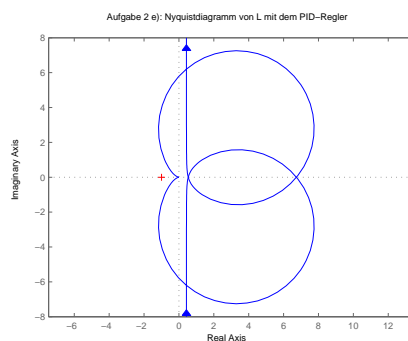
$$Pm_pid = \varphi = 46.5349$$

- e) Das für L_p erstellte Nyquistdiagramm sieht folgendermassen aus:



Man sieht, dass der Punkt -1 nicht umlaufen wird. Das Resultat von `pole(L_p)` zeigt, dass der Loop Gain L nur Pole mit negativem Realteil hat. Deshalb erfüllt dieser Loop Gain die Forderung des Nyquist Theorems.

Analog für L_{pid} des PID-Reglers:



Der Zoom auf den kritischen Punkt bei -1 (rechtes Bild) zeigt den Verlauf etwas genauer. Es gibt 0.5 Umläufe. Nötig wären laut `pole(L_pid)` wiederum genau diese 0.5 positiven Umläufe. Auch dieser Loop Gain erfüllt also die Bedingung.

- f) Mit dem folgenden Code werden die beiden Sprungantworten geplottet:

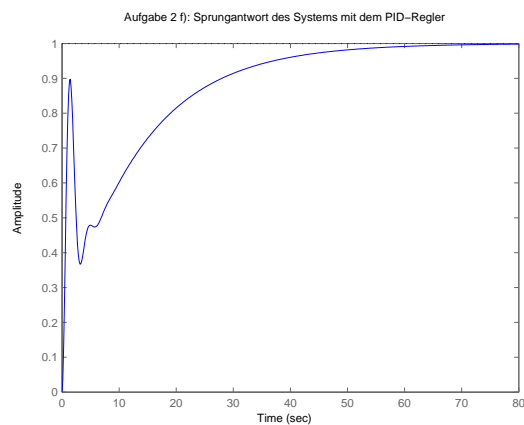
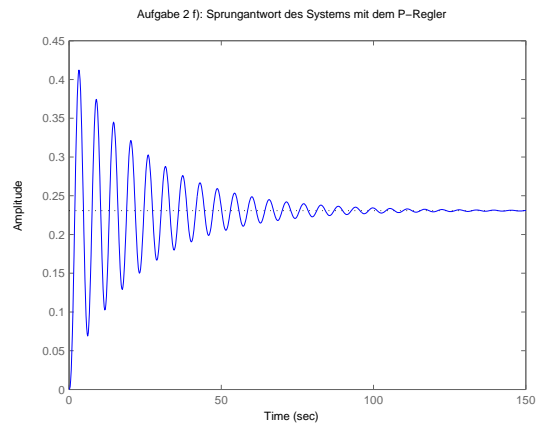
```
T_p=feedback(L_p,1);
```

```
figure('Name','Aufgabe 2 f): Sprungantwort des Systems mit dem P-Regler')
step(T_p);
```

```
T_pid=feedback(L_pid,1);
```

```
figure('Name','Aufgabe 2 f): Sprungantwort des Systems mit dem PID-Regler')
step(T_pid);
```

Die Sprungantworten sehen folgendermassen aus:



Es fällt dabei auf, dass die nur mit dem P-Regler geregelte Strecke einen (grossen) statischen Nachlauffehler hat, was auf den fehlenden Integrator zurückzuführen ist. Ausserdem schwingt das PID-geregelte System viel weniger stark und weniger lang, was man aufgrund der besseren Phasen- und Verstärkungsreserve erwarten konnte.