Prof. Dr.-Ing. P. Pelz
Fachgebiet
Fluidsystemtechnik
Fachbereich 16 · Maschinenbau

Magdalenenstr.4
64289 Darmstadt
Telefon (0 61 51) 16 - 6554
Telefax (0 61 51) 16 - 2453
E-Mail: thomas.bedarff@fst.tu-darmstadt.de

TECHNISCHE UNIVERSITÄT DARMSTADT

Einführung in Modelica

Modelica ist ein objektorientierter Sprachstandard zur Simulation von technischen Systemen. Es existieren verschiedene kommerzielle Distributionen von Modellica (z.B. Dymola, SimulationX, MathModelica). In dieser Übung soll die Open Source Software OpenModelica verwendet werden. OpenModelica kann für die eigene Verwendung kostenfrei von der Seite http://www.ida.liu.se/labs/pelab/modelica/OpenModelica.html herunter geladen werden.

Diese Übung gibt eine grundlegende Einführung in die Modelica Sprache. Das Verständnis der behandelten Beispiele ist Grundlage zur Bearbeitung der anschließenden Übungen.

Vorbereitung: Laden Sie von der Homepage des Fachegebietes (http://www.fst.tu-darmstadt.de/lehre/studienmaterial_1/sommersemester/gtf_1/studienmaterial_gtf.de.jsp) das Übungsmaterial für diese Übung herunter.
Öffnen Sie das pdf-Dokument Aufgabenstellung_Uebung1.pdf.

1. Aufgabe: Kennenlernen des OMNotebook

Das OMNotebook ist eine Benutzeroberfläche von OpenModelica. Bevor Sie das OMNotebook öffnen, erstellen Sie - wenn noch nicht vorhanden - auf Ihrem persönlichen Netzlaufwerk einen Ordner openmodelica. Es muss in diesem Ordner gearbeitet werden. Öffnen Sie nun das OMNotebook aus dem Startmenü CAx. Führen Sie die Modelle *Hello World*, *Differential Algebraic Equation System* und *VanDerPol* aus. Bestätigen Sie innerhalb der Modelle jede blaue Eingabemaske mit SHIFT + ENTER. Machen Sie sich mit dem Aufbau und der Syntax einfacher Modelle in Modelica vertraut.

2. Aufgabe: Differentialgleichungssystem

Öffnen Sie den FreeModelicaEditor (FME). Erstellen Sie ein Modell mit dem Namen „*DglSystem*“ zur Lösung des folgenden Differentialgleichungssystems:

$$\ddot{x} = -ax^2 y + \sin^2(x)$$
$$y = \dot{x}^2$$

Bei x und y handelt es sich um Variablen. Wählen Sie für x den Startwert 1. Die Größe a ist als Parameter mit dem Wert 1 zu setzen. Speichern Sie das Modell in Ihrem Arbeitsverzeichnis mit einer frei gewählten Bezeichnung *.mo ab.

Die Simulation des von Ihnen erstellten Modells, erfolgt wiederum im OMNotebook. Öffnen Sie das vorgefertigte OMNotebook File *simo_aufgabe2.onb* und folgen Sie den dort beschriebenen Anweisungen.

Prof. Dr.-Ing. P. Pelz
Fachgebiet
Fluidsystemtechnik
Fachbereich 16 · Maschinenbau

Magdalenenstr.4
64289 Darmstadt
Telefon (0 61 51) 16 - 6554
Telefax (0 61 51) 16 - 2453
E-Mail: thomas.bedarff@fst.tu-darmstadt.de

TECHNISCHE UNIVERSITÄT DARMSTADT

3. Aufgabe: Komponentenprüfstand

Ein Komponentenprüfstand wird in der Praxis verwendet, um Hysteresekurven und Übertragungsverhalten von Kraftelementen (hydraulisch gedämpfter Federbeine, Luftfedern, Motorlager, …) zu untersuchen. Der prinzipielle Aufbau sieht wie folgt aus:

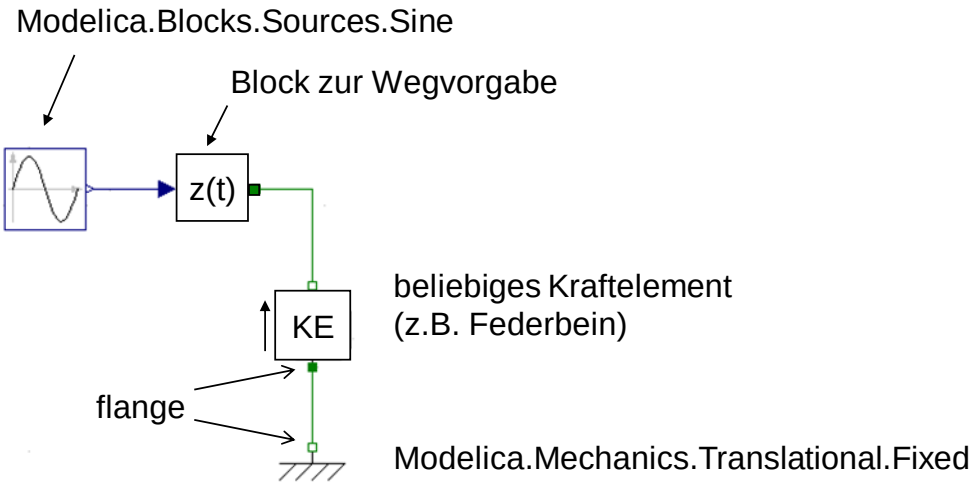

Abbildung 1: objektorientierter Aufbau des Komponentenprüfstands

Wie in Abbildung 1 ersichtlich, erzeugt der Sinusblock ein zeitabhängiges Signal. Dieses wird im Block Wegvorgabe in einen Weg umgewandelt und in das Kraftelement eingeleitet. Das Kraftelement wird auf der Unterseite über ein Festlager eingespannt. Jeder Block verfügt über sog. Konnektoren. Konnektoren stellen die Verbindung zwischen den Blöcken her. Innerhalb dieses Modells werden mechanische Konnektoren sowie ein Signalkonnektor verwendet.

Beim Modell des Komponentenprüfstandes wird das Element Federbein (Kraftelement) mit dem Element Fixed verbunden. Dafür wird der Konnektor des Federbeins (Federbein.flange_a) mit dem Konnektor des Fixed-Elementes (Fixed.flange_b) über den Befehl ‚connect' verbunden:

```
connect(Federbein1.flange_a,Fixed1.flange_b);
```

Auf dieselbe Art und Weise lässt sich das Federbein mit der Wegvorgabe und die Wegvorgabe mit dem Sinusblock verbinden
Folgende Tabelle fasst die verwendeten Konnektoren zusammen:

Prof. Dr.-Ing. P. Pelz
Fachgebiet
Fluidsystemtechnik
Fachbereich 16 · Maschinenbau

Magdalenenstr.4
64289 Darmstadt
Telefon (0 61 51) 16 - 6554
Telefax (0 61 51) 16 - 2453
E-Mail: thomas.bedarff@fst.tu-darmstadt.de

TECHNISCHE UNIVERSITÄT DARMSTADT

Mechanische Konnektoren	flange_a	Komponenten:	flange_a.s flange_a.f	Symbol:
	flange_b	Komponenten:	flange_b.s flange_b.f	Symbol:
Signalkonnektor	RealInput		y	Symbol:

Tabelle 1: Konnektoren des Komponentenprüfstands

Jeder mechanische Konnektor überträgt die Größen Weg s und Kraft f. Zum Beispiel wird die Konnektorklasse flange_a angesprochen über flange_a.s und flange_a.f. Der Signalkonnektor über eine Signalgröße und besitzt nur die Komponente y.

In der nächsten Abbildung ist der OpenModelica Quellcode für den Komponentenprüfstand dargestellt. Die Untermodelle *Wegvorgabe*, *Federbein* und *Systemaufbau* sind in das Package *Testrig* integriert. Ein package ist ein Container für Klassen. Diese Vorgehensweise bietet sich immer an, wenn mehrere Untermodelle vorhanden sind. Nach der Packagedefinition werden die verschiedenen Untermodelle definiert. Das Gesamtmodell wird schließlich im Model ‚Systemaufbau' aus den selbstdefinierten Modellen ‚Federbein' und ‚Wegvorgabe' sowie den vordefinierten Modellen ‚Fixed' und ‚Sine' zusammengebaut. Dafür wird von den benötigten Klassen eine Instanz gebildet (z.B. die Instanz Wegvorgabe1 von der Klasse Wegvorgabe) und diese mittels des ‚connect'-Befehls verbunden.

```
package Testrig

  model Wegvorgabe
    Modelica.Blocks.Interfaces.RealInput u;                          // Laden des Modelica Konnektors RealInput
    Modelica.Mechanics.Translational.Interfaces.Flange_a flange_a;   // Laden des Modelica Konnektors Flange_a
  equation
    flange_a.s = u;
  end Wegvorgabe;

  model Federbein
    parameter Real c(unit="N/m")=30000;
    parameter Real d(unit="Ns/m")=3000;
    Real F(unit="N");
    Real z(unit="m");
    Modelica.Mechanics.Translational.Interfaces.Flange_a flange_a;  // Laden des Modelica Konnektors Flange_a
    Modelica.Mechanics.Translational.Interfaces.Flange_b flange_b;  // Laden des Modelica Konnektors Flange_b
  equation
    z=flange_a.s-flange_b.s;
    F=z*c+d*der(z);
    flange_a.f=F;
    flange_b.f=-F;
  end Federbein;

  model Systemaufbau          // In diesem Modell wird das gesamte System zusammengesetzt
    Wegvorgabe Wegvorgabe1;   // Einfügen des Modells Wegvorgabe. Dieses trägt innerhalb des Modells Systemaufbau die Bezeichnung Wegvorgabe1
    ...                       // Hier das Federbein einfügen, analog wie bei Block Wegvorgabe
    Modelica.Mechanics.Translational.Fixed Fixed1;               // Einfügen des Modelica Blocks Fixed.
    Modelica.Blocks.Sources.Sine Sine1(amplitude=0.01, freqHz=1); // Einfügen des Modelica Blocks Sine
  equation
    connect(Sine1.y,Wegvorgabe1.u);       // Hier wird der Block Sine1 mit dem Block Wegvorgabe1 verbunden
    connect(Wegvorgabe1.flange_a, ... );  // Hier den Block Wegvorgabe mit Federbein verbinden
    connect( ... ,Fixed1.flange_b);       // Hier den Block Federbein mit Fixed verbinden
  end Systemaufbau;

end Testrig;
```

Abbildung 2: OpenModelica Quellcode für den Komponentenprüfstand

Prof. Dr.-Ing. P. Pelz
Fachgebiet
Fluidsystemtechnik
Fachbereich 16 · Maschinenbau

Magdalenenstr.4
64289 Darmstadt
Telefon (0 61 51) 16 - 6554
Telefax (0 61 51) 16 - 2453
E-Mail: thomas.bedarff@fst.tu-darmstadt.de

TECHNISCHE UNIVERSITÄT DARMSTADT

Aufgabenstellung:

a) Laden Sie das vorgefertigte Modell Testrig.mo herunter und starten es im FreeModelicaEditor. Ergänzen Sie im Untermodell Systemaufbau die fehlenden Zeilen. **Orientieren Sie sich dabei an den eingefügten Kommentaren im Quellcode.**
b) Zur Simulation verwenden Sie das vorgefertigte OMNotebook File *simo_aufgabe3.onb*. Laden Sie darin das File Testrig.mo. Simulieren Sie Testrig.Systemaufbau und schauen Sie sich die Ergebnisse an.

4. Aufgabe: Tanksystem

Ziel dieser Aufgabe ist die objektorientierte Abbildung eines Tanksystems mit Zu- und Ablauf sowie Regelung des Flüssigkeitsstands. Die Lernziele dieser Übung liegen in der Erstellung eigener Konnektorklassen für Fluidsysteme, Sensoren und Aktoren sowie dem Aufbau einer Regelung in Modelica.

Eine Pumpe fördert einen konstanten Volumenstrom in einen Tank. Ein Sensor erfasst den Höhenstand der Flüssigkeit im Tank und leitet diese Information an einen Regler weiter. Der Regler habe die Aufgabe den Höhenstand der Flüssigkeit auf einen Sollwert zu regeln. Dazu wird das Auslassventil am Tank entsprechend geöffnet bzw. geschlossen.
Zur Umsetzung in Modelica ist eine Einteilung des Systems in Untersysteme erforderlich. Es wird folgende Einteilung vorgenommen:

- Fluidquelle
- Tank
- Regler

Im zweiten Schritt werden die erforderlichen Konnektoren zusammengestellt. Es werden drei Klassen von Konnektoren verwendet. Diese erhalten die Bezeichnungen:

Sensor	Bezeichnung:	Sensor	Symbole: 
Aktor		Actor	
Volumenstrom		LiquidFlow	

Tabelle 2: Konnektoren des Tanksystems

Das objektorientierte Modell in Modelica erhält somit folgenden Prinzipaufbau:

Prof. Dr.-Ing. P. Pelz
Fachgebiet
Fluidsystemtechnik
Fachbereich 16 · Maschinenbau

Magdalenenstr.4
64289 Darmstadt
Telefon (0 61 51) 16 - 6554
Telefax (0 61 51) 16 - 2453
E-Mail: thomas.bedarff@fst.tu-darmstadt.de

TECHNISCHE UNIVERSITÄT DARMSTADT

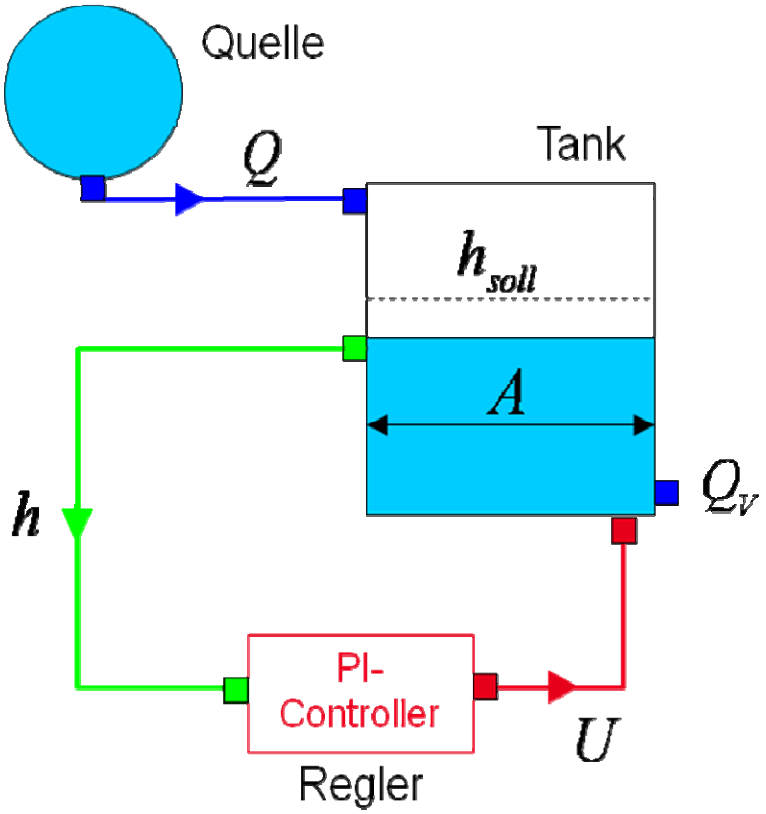

Abbildung 3: objektorientierter Aufbau des Tanksystems

Die Umsetzung im Modelica Quellcode wird im Folgenden erklärt. Um den Aufbau der Untermodelle übersichtlich zu gestalten, bietet es sich an, diese in ein Package zu schreiben. Das Package erhält den Namen *TankSystem*. Zu Beginn werden die in den Modellen verwendeten Konnektorklassen definiert:

```
package TankSystem

  connector Sensor
    Real value;
  end Sensor;

  connector Actor
    Real value;
  end Actor;

  connector LiquidFlow
    Real value;
  end LiquidFlow;

....
```

Nach der Definition der Konnektoren werden die Modelle Fluidquelle, Tank und PI Regler definiert.

Beschreibung Fluidquelle:

Prof. Dr.-Ing. P. Pelz
Fachgebiet
Fluidsystemtechnik
Fachbereich 16 · Maschinenbau

Magdalenenstr.4
64289 Darmstadt
Telefon (0 61 51) 16 - 6554
Telefax (0 61 51) 16 - 2453
E-Mail: thomas.bedarff@fst.tu-darmstadt.de

TECHNISCHE UNIVERSITÄT DARMSTADT

```
model LiquidSource
  parameter Real flowrate(unit="m3/s")=0.02;
  TankSystem.LiquidFlow q_out;

equation
  q_out.value=if time < 150 then flowrate else 3*flowrate;

end LiquidSource;
```

Die Fluidquelle stellt bis zum Zeitpunkt t=150s einen konstanten Volumenstrom bereit. Für t>150s steigt die Fördermenge auf einen dreifachen Wert an.

Beschreibung Tank:

```
model Tank

  parameter Real area(unit="m2")=1;
  parameter Real flowGain(unit="m2/s")=0.05;
  Real h;
  TankSystem.Sensor Sensor1;
  TankSystem.Actor Actor1;
  TankSystem.LiquidFlow q_in;
  TankSystem.LiquidFlow q_out;

equation
  der(h)=(q_in.value-q_out.value)/area;
  q_out.value=-flowGain*Actor1.value;
  Sensor1.value=h;

end Tank;
```

Im Tank werden zunächst die Parameter Querschnittsfläche und ein Skalierungsfaktor für das Auslassventil definiert. Es folgt die Definition der Variable h für die Flüssigkeitshöhe im Tank. Es werden die Konnektoren für den Sensor1, Actor1, q_in und q_out eingeführt. Nach dem Schlüsselwort *equation* folgen die Gleichungen zur Beschreibung des Tanks. Die Flüssigkeitshöhe h lässt sich aus der Differentialgleichung:

$$\frac{dh}{dt} = \frac{q_{in} - q_{out}}{A}$$

berechnen. Der austretende Volumenstrom wird bestimmt von der Ventilposition am Auslassventil. Dieses wird über die vom Reglerausgang kommende Größe Actor1.value entsprechend verstellt. Zuletzt wird dem Sensorkonnektor der im Tank vorherrschende Flüssigkeitsstand h zugewiesen.

Beschreibung PI Regler:

Prof. Dr.-Ing. P. Pelz
Fachgebiet
Fluidsystemtechnik
Fachbereich 16 · Maschinenbau

Magdalenenstr.4
64289 Darmstadt
Telefon (0 61 51) 16 - 6554
Telefax (0 61 51) 16 - 2453
E-Mail: thomas.bedarff@fst.tu-darmstadt.de

TECHNISCHE UNIVERSITÄT DARMSTADT

```
model PI_Controller

  parameter Real flowGain(unit="m2/s")=0.05;
  parameter Real K=2 "Gain";
  parameter Real T(unit="s")=10 "Time Constant";
  parameter Real ref=0.25 "Reference level";
  Real error "Difference to Reference level";
  Real outCtr "Control signal without limiter";
  Real x "State variable for controller";
  TankSystem.Actor cOut;
  TankSystem.Sensor cIn;
equation
  der(x)=error/T;
  outCtr=K*(error+x);
  error=ref-cIn.value;
  cOut.value=outCtr;

end PI_Controller;
```

Der PI Regler hat die Aufgabe die Regelabweichung zwischen der Flüssigkeitshöhe *h* und einer Referenzhöhe, *ref* genannt, auszuregeln. Mathematisch wird der PI Regler beschrieben über

$$outCtr = K\left(error(t) + \frac{1}{T}\int_0^t error(t)dt \right)$$

Für die Umsetzung in Modelica muss der Integralausdruck noch verarbeitet werden, da es keinen Befehl zum Integrieren gibt. Vielmehr muss der Integralausdruck in die Lösung einer Differentialgleichung umgeschrieben werden. Setzt man z.B.

$$\frac{dx}{dt} = \frac{error(t)}{T}$$

so ist x der gesuchte Ausdruck für das Integral:

$$x = \int_0^t \frac{dx}{dt}dt = \int_0^t \frac{error(t)}{T}dt$$

Die komplette Gleichung für den Regler kann nun geschrieben werden als Gleichungssystem:

$$outCtr = K\left(error(t) + x\right)$$
$$\frac{dx}{dt} = \frac{error(t)}{T}$$

Umgesetzt in Modelica:

```
der(x)=error/T
outCtr=K(error+x)
```

Prof. Dr.-Ing. P. Pelz
Fachgebiet
Fluidsystemtechnik
Fachbereich 16 · Maschinenbau

Magdalenenstr.4
64289 Darmstadt
Telefon (0 61 51) 16 - 6554
Telefax (0 61 51) 16 - 2453
E-Mail: thomas.bedarff@fst.tu-darmstadt.de

TECHNISCHE UNIVERSITÄT DARMSTADT

Der Integralausdruck ist somit durch die Lösung einer Differentialgleichung (x) ersetzt worden, diese Lösung wird automatisch numerisch berechnet.
Die Ausgangsgröße des PI Reglers setzt sich aus einem der Regelabweichung proportionalen und einem integralen Faktor zusammen. K ist ein Reglerverstärkungsfaktor und T die Zeitkonstante für den I-Anteil des Reglers.
Nach Deklaration der erforderlichen Parameter, Variablen und Konnektoren im ersten Teil erfolgt nach equation die Eingabe der entsprechenden Gleichungen für den Regler. Der Sensor leitet die vorliegende Flüssigkeitshöhe im Tank an den Regler weiter. Die berechnete Reglergröße wird dem Aktorkonnektor übergeben.

Im letzten Schritt erfolgt der Zusammenbau der Einzelmodelle:

```
  model Systemaufbau

  TankSystem.PI_Controller PI_Controller1;
  TankSystem.LiquidSource LiquidSource1;
  TankSystem.Tank Tank1;

  equation
    connect(LiquidSource1.q_out, Tank1.q_in);
    connect(Tank1.Sensor1, PI_Controller1.cIn);
    connect(PI_Controller1.cOut, Tank1.Actor1);

  end Systemaufbau;

end TankSystem;
```

Im ersten Abschnitt werden die Modelle *Tank*, *Fluidquelle* und *PI Regler* deklariert. Nach dem Schlüsselwort *equation* werden die Modelle entsprechend Abbildung 3 zusammengefügt.

Aufgabenstellung:

a) Erstellen Sie ein Modell für das oben beschriebene Tanksystem. Öffnen Sie dazu den FreeModelicaEditor und speichern das erstellte Modell unter *Tanksystem.mo* in Ihrem Netzlaufwerk ab.
b) Öffnen Sie das OMNotebook. Laden Sie im OMNotebook das erstellte Modell *Tanksystem.mo*. Führen Sie einen Syntaxcheck durch und simulieren das Modell. Schauen Sie sich die Simulationsergebnisse an.