DrControl - An Interactive Course Material for Teaching Control Engineering

Mohsen Torabzadeh-Tari
Linköping University

Content

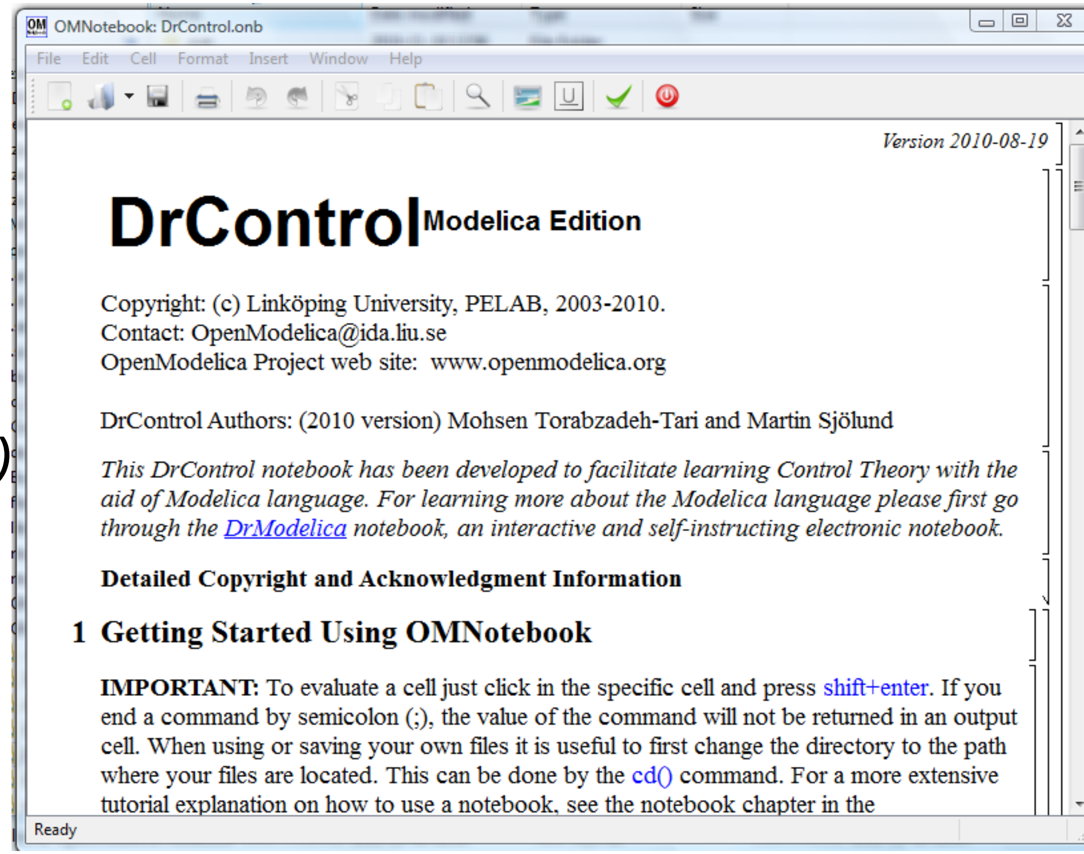
- OMNotebook
- Other Interactive Notebooks
- DrControl
  - Content List
  - Teaching Cycle
- Examples
- Demonstration
- Future work
- Conclusions

OMNotebook – A Literate Programming Notebook

- Primarily for Teaching
- Interactive electronic book
- Platform independent

Commands:
- *Shift-return (evaluates a cell)*

Cell types: text cells & executable code cells



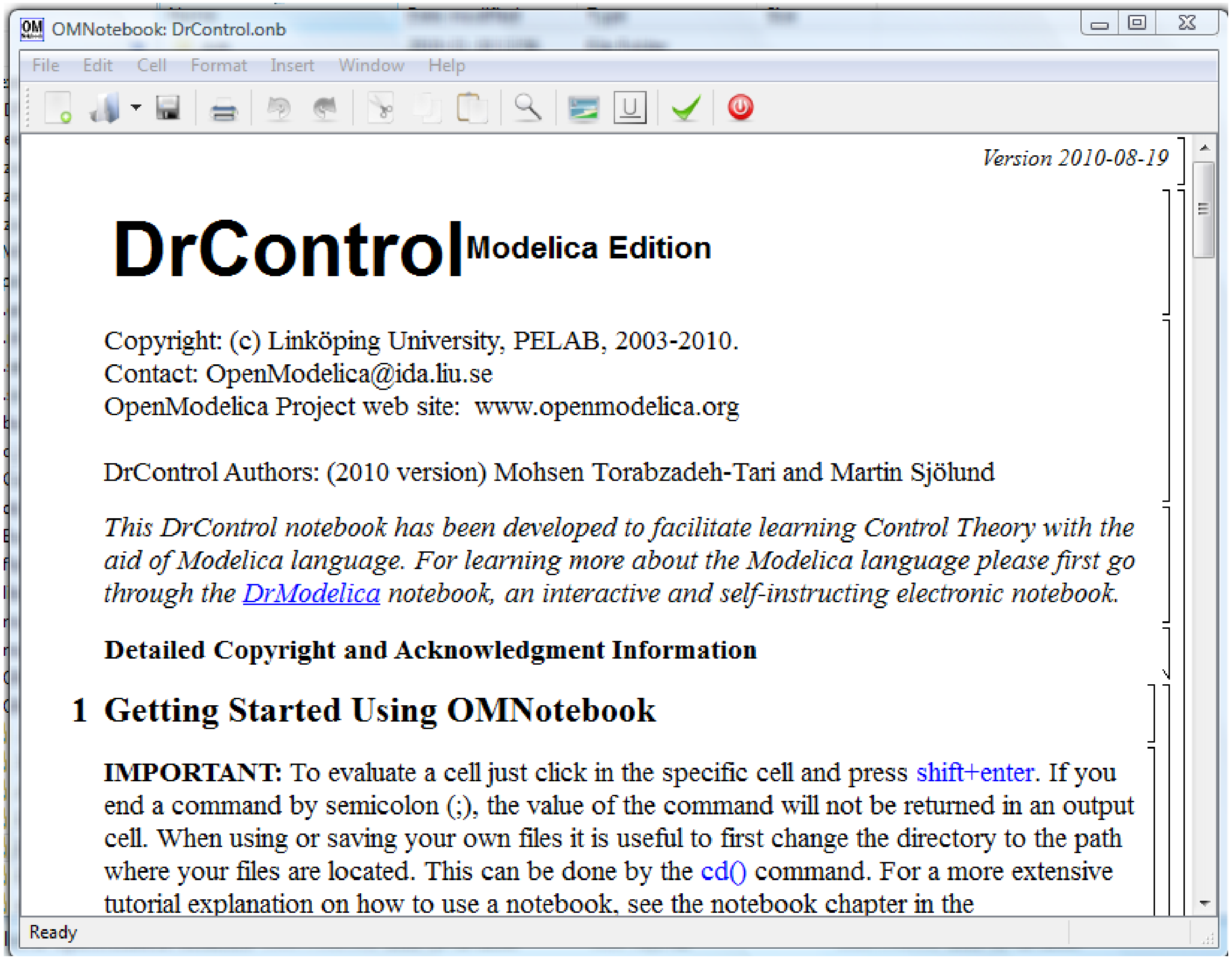
OMNotebook – A Literate Programming Notebook Cont.

- Alternative or complement to traditional methods
- More engagements from students
- Contain interactive technical computations, text and figures
- Suitable for teaching, experimentation, simulation, scripting, model documentation, storage

Other Interactive Notebooks

- DrScheme
- DrJava
- Sagenb
- DrModelica

DrControl – Front Page

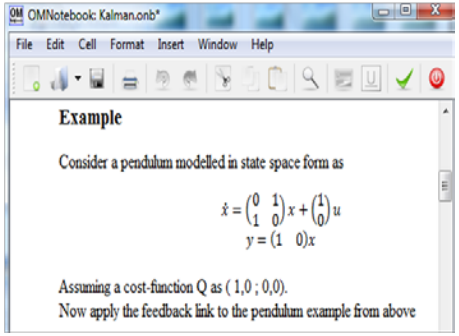


DrControl – Content List

- Feedback loop
- Mathematical modeling
- Transfer function
- State-space form
- Observer – Kalman filter
- Linear quadratic optimization
- Linearization

DrControl – Teaching Cycle

Examples


OMNotebook: Kalman.onb*

File Edit Cell Format Insert Window Help

Example

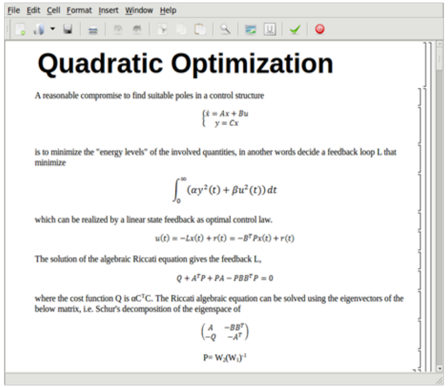
Consider a pendulum modelled in state space form as

$$\dot{x} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} x + \begin{pmatrix} 1 \\ 0 \end{pmatrix} u$$
$$y = \begin{pmatrix} 1 & 0 \end{pmatrix} x$$

Assuming a cost-function Q as (1,0 ; 0,0).

Now apply the feedback link to the pendulum example from above

Theory


File Edit Cell Format Insert Window Help

Quadratic Optimization

A reasonable compromise to find suitable poles in a control structure

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases}$$

is to minimize the "energy levels" of the involved quantities, in another words decide a feedback loop L that minimize

$$\int_0^\infty (\alpha y^2(t) + \beta u^2(t))\,dt$$

which can be realized by a linear state feedback as optimal control law.

$$u(t) = -Lx(t) + r(t) = -B^T Px(t) + r(t)$$

The solution of the algebraic Riccati equation gives the feedback L,

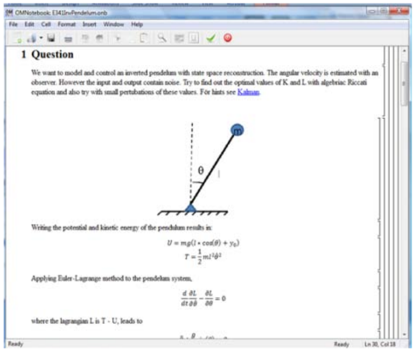
$$Q + A^T P + PA - PBB^T P = 0$$

where the cost function Q is $\alpha C^T C$. The Riccati algebraic equation can be solved using the eigenvectors of the below matrix, i.e. Schur's decomposition of the eigenspace of

$$\begin{pmatrix} A & -BB^T \\ -Q & -A^T \end{pmatrix}$$

$$P = W_2(W_1)^{-1}$$

Questions


OMNotebook: E341InvPendelum.onb

File Edit Cell Format Insert Window Help

1 Question

We want to model and control an inverted pendelum with state space reconstruction. The angular velocity is estimated with an observer. However the input and output contain noise. Try to find out the optimal values of K and L with algebriac Riccati equation and also try with small pertubations of these values. For hints see Kalman.

Writing the potential and kinetic energy of the pendulum results in:

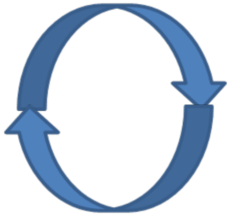
$$U = mg(l \cdot \cos(\theta) + y_0)$$
$$T = \frac{1}{2} ml^2\dot{\theta}^2$$

Applying Euler-Lagrange method to the pendelum system,

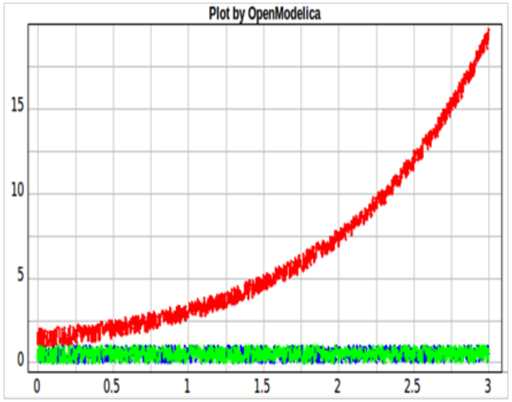
$$\frac{d}{dt}\frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = 0$$

where the lagrangian L is T - U, leads to

Ready Ready Ln 30, Col 18



Experimentation



Models

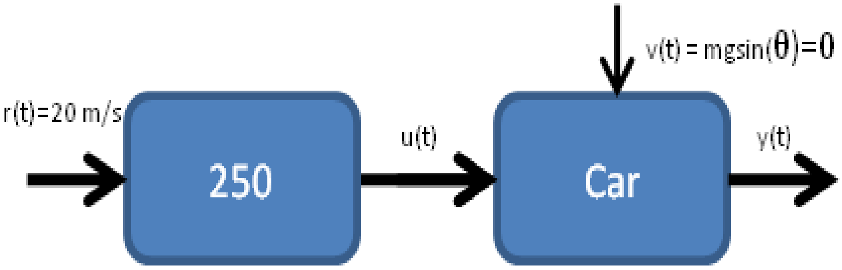
```
model stateSpaceNoise
  StateSpaceWithNoise stateSpace;
   Modelica.Blocks.Sources.Exponentials ref(
     outMax=4,riseTime=1,riseTimeConst=1,
     fallTimeConst=0.2,offset=0,startTime=-1);
initial equation
  stateSpace.x[1]=1;
  stateSpace.x[2]=0;
equation
  connect(ref.y, stateSpace.u[1]);
end stateSpaceNoise;
```

Simple Car Model with Open Loop Control

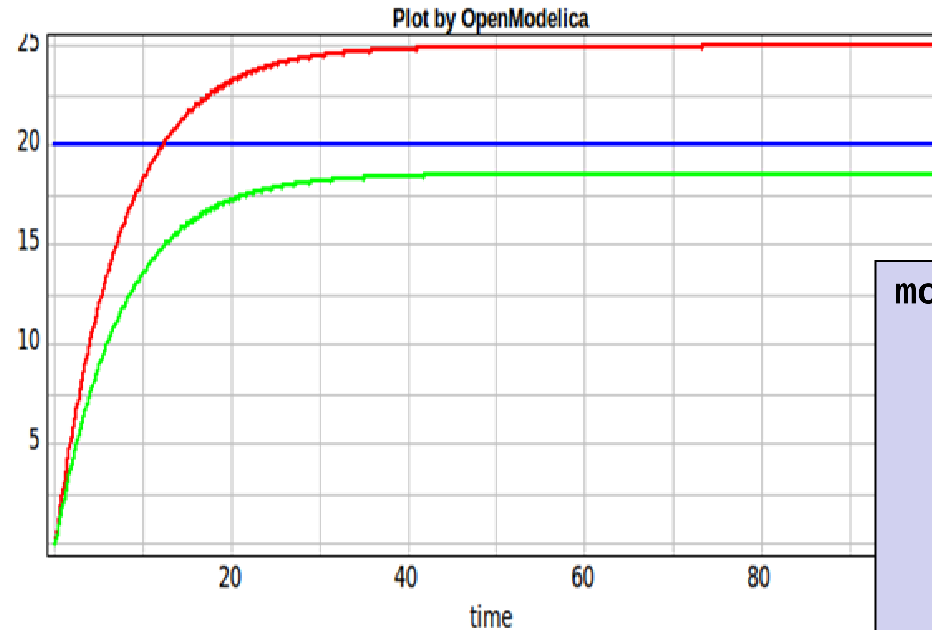
Assume that we want to control the velocity of a car with an open loop control

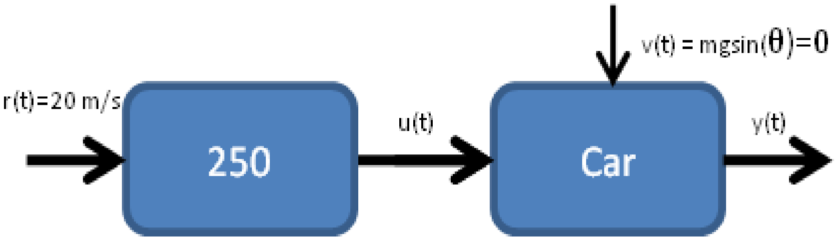
$$m\dot{y} = u - \alpha y - mgsin(\theta)$$

- Mass m
- Velocity y
- Aerodynamic coefficient α
- Road slope θ



Simple Car Model with Open Loop Control





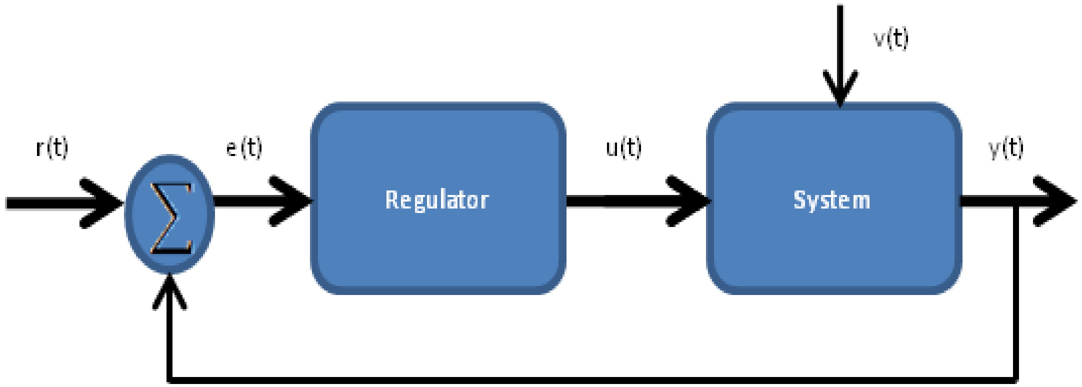

```
model NoFeedback
  import SI = Modelica.SIunits;
  SI.Velocity y          "No noise";
  SI.Velocity yNoise     "With noise";
  parameter SI.Mass m = 1500;
  parameter Real alpha = 200;
  parameter SI.Angle theta = 5*3.14/180;
  parameter SI.Acceleration g = 9.82;
  SI.Force u;
  SI.Velocity r = 20 "Reference signal";
equation
  m*der(y)=u - alpha*y;
  m*der(yNoise)= u - alpha*yNoise –
     m*g*sin(theta);
  u = 250A*r;
end NoFeedback;
```


Simple Car Model with Closed Loop Control

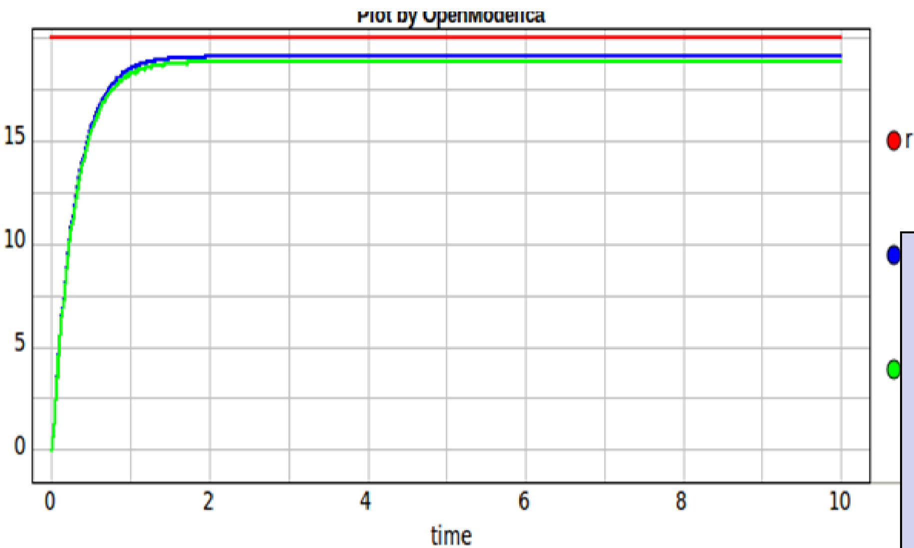
A slope angle<>0 can be regarded as noise in this model.

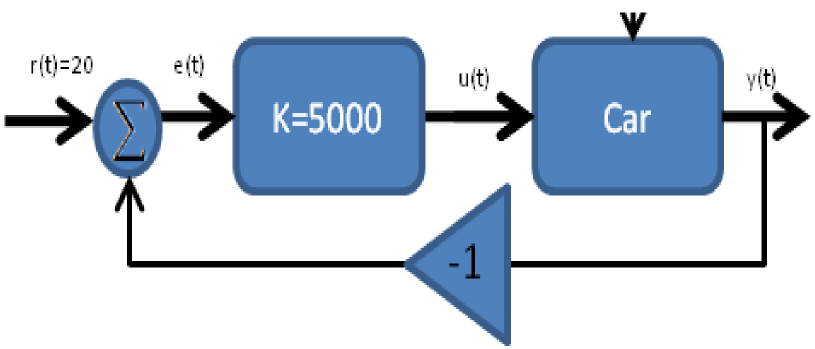
Apply a feedback loop for eliminating this effect and the overshoot through a proportional regulator

$$u = K * (r - y)$$



Simple Car Model with Closed Loop Control



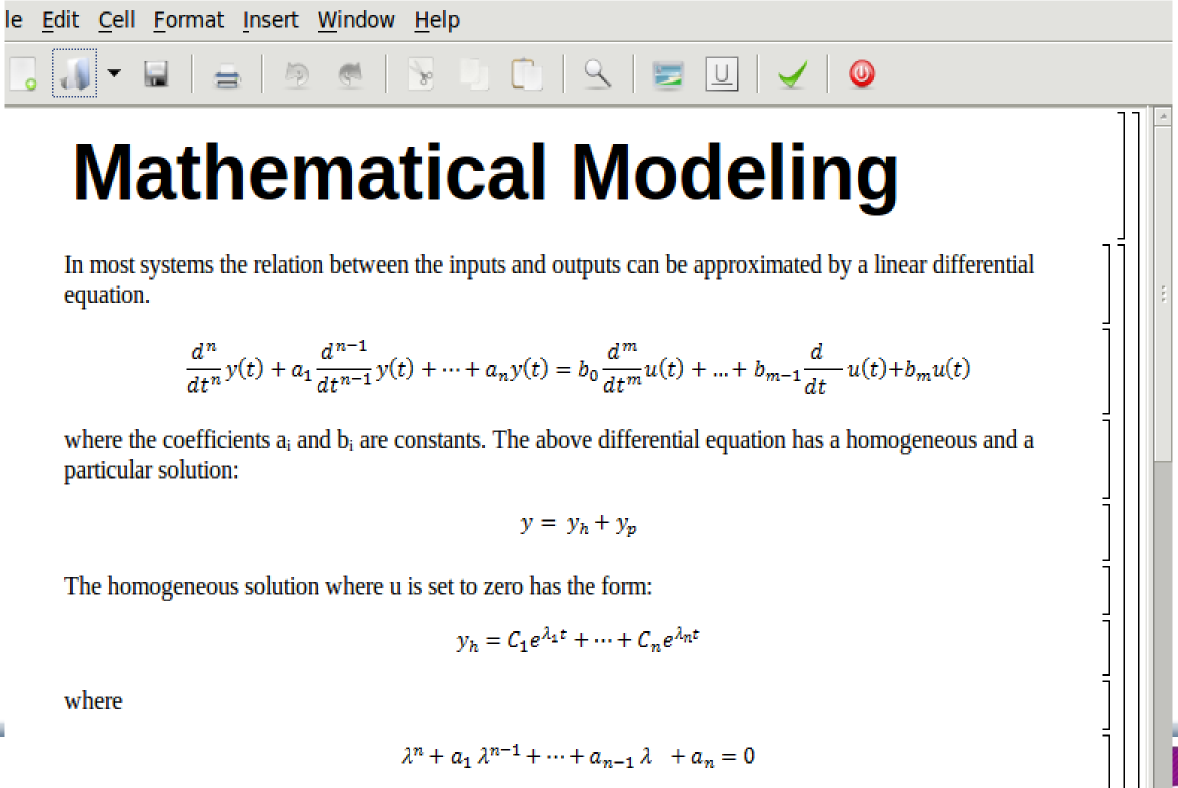



```
model WithFeedback
  import SI = Modelica.SIunits;
  SI.Velocity y      "Output, No noise";
  SI.Velocity yNoise "Output With noise";
  parameter SI.Mass m = 1500;
  parameter Real alpha = 250;
  parameter SI.Angle theta = 5*3.14/180;
  parameter SI.Acceleration g = 9.82;
  SI.Force u;
  SI.Force uNoise;
  SI.Velocity r = 20  "Reference signal";
equation
  m*der(y) = u - alpha*y;
  m*der(yNoise) = uNoise - alpha*yNois –
    m*g*sin(theta);
  u = 5000*(r - y);
  uNoise = 5000*(r - yNoise);
end WithFeedback;
```

Mathematical Modeling – Stability

In most systems the relation between the inputs and outputs can be described by a linear differential equation.

$$\frac{d^n y}{dt^n} + a_1 \frac{d^{n-1} y}{dt^{n-1}} + \dots + a_n y = b_0 \frac{d^m u}{dt^m} + \cdots + b_{m-1} \frac{du}{dt} + b_m u$$



Mathematical Modeling

In most systems the relation between the inputs and outputs can be approximated by a linear differential equation.

$$\frac{d^n}{dt^n} y(t) + a_1 \frac{d^{n-1}}{dt^{n-1}} y(t) + \cdots + a_n y(t) = b_0 \frac{d^m}{dt^m} u(t) + \dots + b_{m-1} \frac{d}{dt} u(t) + b_m u(t)$$

where the coefficients a_i and b_i are constants. The above differential equation has a homogeneous and a particular solution:

$$y = y_h + y_p$$

The homogeneous solution where u is set to zero has the form:

$$y_h = C_1 e^{\lambda_1 t} + \cdots + C_n e^{\lambda_n t}$$

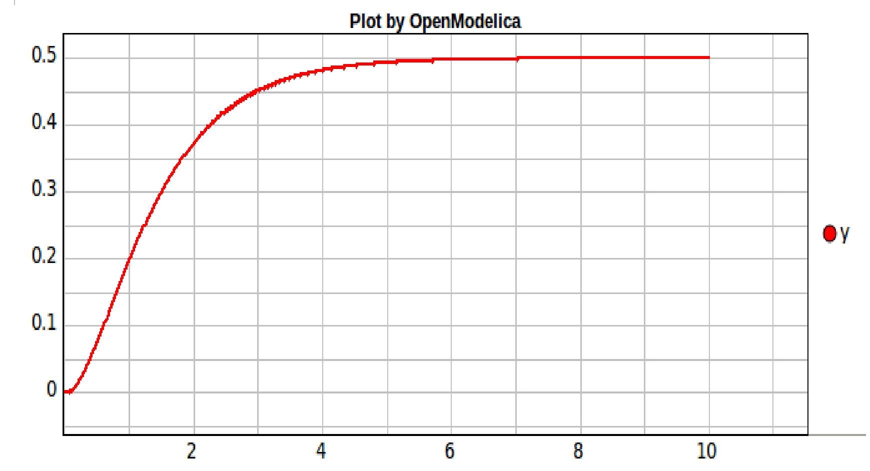
where

$$\lambda^n + a_1 \lambda^{n-1} + \cdots + a_{n-1} \lambda + a_n = 0$$

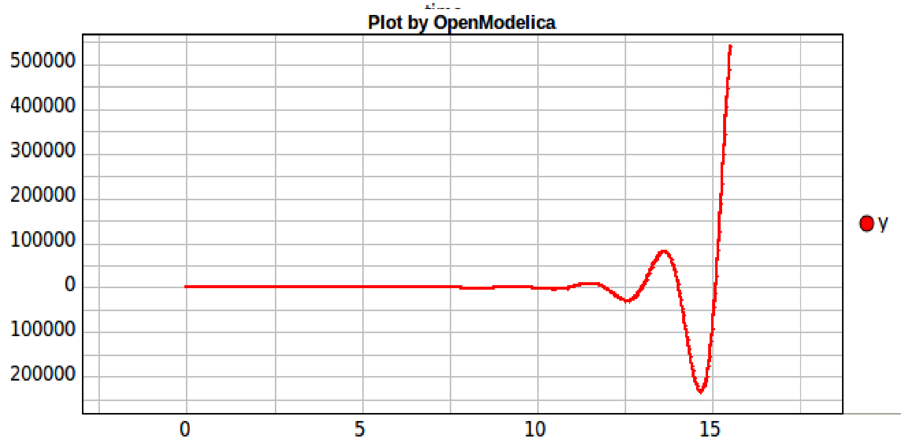
Stability Analysis of A Second Order System

$$\ddot{y} + a_1\dot{y} + a_2 y = 1$$

```
model NegRoots
  Real y;
  Real der_y;
  parameter Real a1 = 3;
  parameter Real a2 = 2;
equation
  der_y = der(y);
  der(der_y) + a1*der_y + a2*y = 1;
end NegRoots;
```




```
model PosImgRoots
  Real y;
  Real der_y;
  parameter Real a1 = -2;
  parameter Real a2 = 10;
equation
  der_y = der(y);
  der(der_y) + a1*der_y + a2*y = 1;
end PosImgRoots;
```

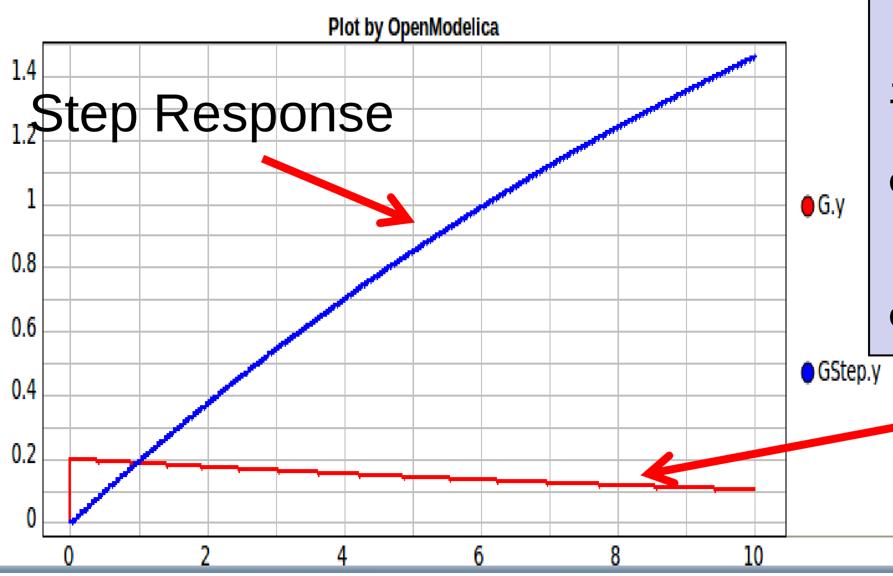


Transfer Function – Pulse and Step Responses

$$Y(s) = G(s)U(s)$$

$$G(s) = \frac{\frac{1}{A}}{s + \frac{1}{T}}$$

```
model Tank
  import Modelica.Blocks.Continuous.*;
  TransferFunction G(b = {1/A}, a =
    {1,1/T});
  TransferFunction GStep(b = {1/A},a =
    {1,1/T});
  parameter Real T = 15 "Time constant";
  parameter Real A = 5;
  Real uStep = if (time > 0 or time<0)
     then 1 else 0 "step function";
initial equation
  G.y = 1/A;
equation
  G.u= if time > 0 then 0 else 1e6;
  GStep.u = uStep;
end Tank;
```



Pulse Response

Differential Equations – Initial Conditions

$$\ddot{y} + a_1\dot{y} + a_2 y = bu$$

Second order

to

First order

Auxilary Varaibles $\begin{cases} x_1 = y \\ x_2 = \dot{y} \end{cases}$

$$\begin{pmatrix} \dot{x}_1 \\ \dot{x}_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -a_2 & -a_1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ b \end{pmatrix} u$$

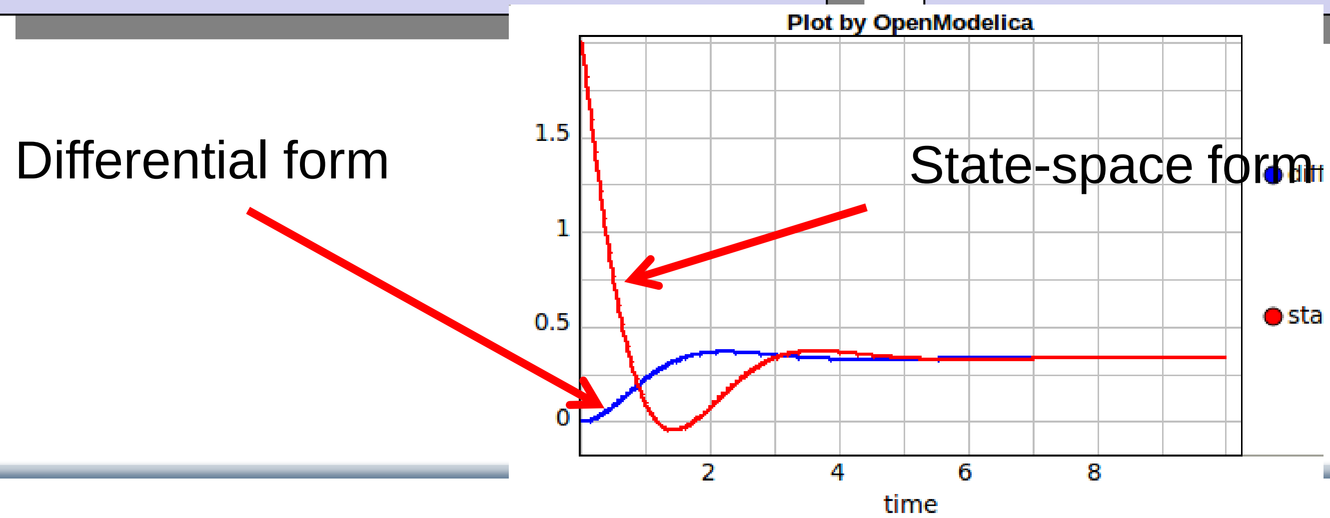
Differential Equations – Initial Conditions, Cont.

$$\ddot{y} + a_1\dot{y} + a_2 y = bu$$

$$\begin{pmatrix}\dot{x}_1\\ \dot{x}_2\end{pmatrix} = \begin{pmatrix}0 & 1\\ -a_2 & -a_1\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix} + \begin{pmatrix}0\\ b\end{pmatrix}u$$

```
model DiffEqHD
  Real u = 1;
  Real y;
  Real uprim = der(u);
  Real z = der(y);
equation
  der(z)+2*z+3*y =
     2*der(uprim)+uprim+u;
end DiffEqHD;
```

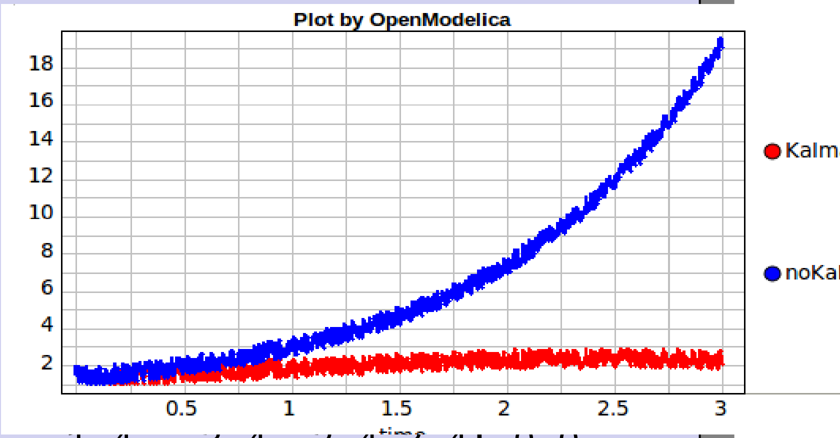
```
model StateSpaceHD
  Modelica.Blocks.Continuous.StateSpace
    stateSpace(A=[-2,1; -3,0],B=[-3;5]
                ,C=[1,0],D=[2]);
  Modelica.Blocks.Sources.Step
    step(height=1.0);
equation
  connect(step.y, stateSpace.u[1]);
end StateSpaceHD;
```



Observer

No access to the internal states of a system and can only measure the outputs of the system and have to reconstruct the internal state of the system based on these measurements, e.g. observer .

```
model KalmanFeedback
  parameter Real A[:,size(A, 1)] =  {{0
  parameter Real B[size(A, 1),:] =  {{0
  parameter Real C[:,size(A, 1)] =  {{1
  parameter Real[2,1] K = [2.4;3.4];
  parameter Real[1,2] L = [2.4,3.4];
  parameter Real[:,:] ABL = A-B*L;
  parameter Real[:,:] BL = B*L;
  parameter Real[:,:] Z =  zeros(size(A
  parameter Real[:,:] AKC = A-K*C;
  parameter Real[:,:] Anew = [0,1,0,0 ; -1.4, -3.4, 2.4,3.4; 0,0,-
                            2.4,1;0,0,  2.4,0];
  parameter Real[:,:] Bnew = [0;1;0;0];
  parameter Real[:,:] Fnew = [1;0;0;0];
  StateSpaceNoise Kalman(StateSpace.A=Anew, StateSpace.B=Bnew,
    StateSpace.C=[1,0,0,0],  StateSpace.F = Fnew);
  StateSpaceNoise noKalman;
end KalmanFeedback;
```



Linearization

Many nonlinear problems can be handled more easily by linearization around an equilibrium point. We can investigate the behavior of the nonlinear system by analyzing the linearized approximation.

```
setCommandLineOptions({"+d=lineaization"})
```

```
buildModel(TwoTankModel)

system("TwoTankModel.exe -l 0.0 –v >log.out")

readFile("log.out")
```

```
model TwoTankModel
  Real h1(start = 2);
  Real h2(start = 1);
  Real F1;
  parameter Real A1 = 2,A2 = 0.5;
  parameter Real R1 = 2,R2 = 1;
  input Real F;
  output Real F2;
equation
  der(h1) = (F/A1) - (F1/A1);
  der(h2) = (F1/A2) - (F2/A2);
  F1 = R1 * sqrt(h1-h2);
  F2 = R2 * sqrt(h2);
end TwoTankModel;
```

Linearization, cont.

The file `log.out` contains now the linearized model:

```
model Linear_TwoTankModel
  parameter Integer n = 2; // states
  parameter Integer k = 1;
  parameter Integer l = 1;
  parameter Real x0[2] = {2,1};
  parameter Real u0[1] = {0};
  parameter Real A[2,2] = [-0.5,0.5;2,-3];
  parameter Real B[2,1] = [0.5;0];
  parameter Real C[1,2] = [0,0.5];
  parameter Real D[1,1] = [0];
  Real x[2](start = x0);
  input Real u[1](start = u0);
  output Real y[1];
  Real x_Ph1 = x[1];
  Real x_Ph2 = x[2];
  Real u_PF = u[1];
  Real y_PF2 = y[1];
equation
  der(x) = A * x + B * u;
  y = C * x + D * u;
end Linear_TwoTankModel;
```

Conclusions

- One of few open source effort
- Programming and Modeling
- OMNotebook applied to Control

Future Work

- 3D visualization
- Other engineering fields, DrMechanics
- Frequency Analysis
- Integration to OMEdit

