Modeling Language HYSDEL

Martin Herceg

Slovak University of Technology in Bratislava

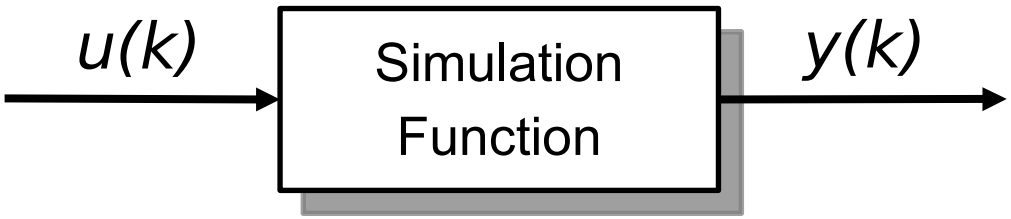
Automatic Control Laboratory, ETH Zürich

Outline

- HYSDEL introduction
- New version HYSDEL 3.0
- Illustrative example
- Conclusion

Hybrid modeling

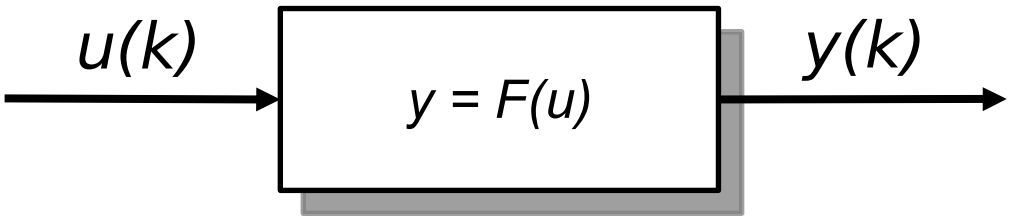
- Hybrid models
  - describe evolution of systems including real and logical variables in discrete time
- Model is typically written as a simulation function which contains
  - logical conditions, IF-THEN clauses, repeating expressions etc.



Very hard analysis and control synthesis

What is HYSDEL?

- HYSDEL = HYbrid System DEscription Language
- Framework for modeling of **hybrid systems**
  - uses simple language statements to model complex relations
  - outputs a model which is suitable for further use



Easy analysis and control synthesis

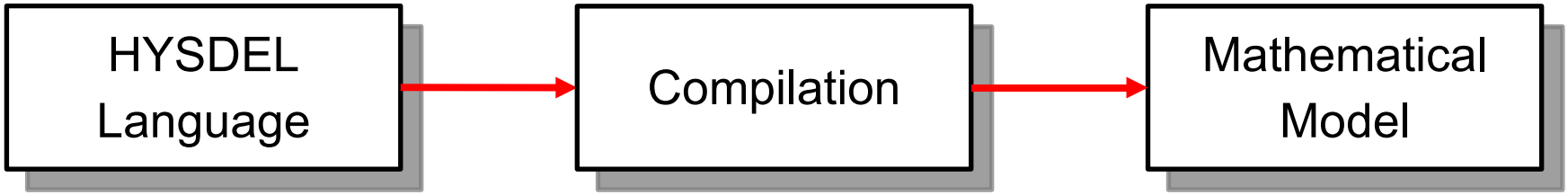
What is the model?

$$
\begin{aligned}
x(k+1) &= Ax(k)+B_u u(k)+B_{aux} w(k)+B_{aff} \\
y(k) &= Cx(k)+D_u u(k)+D_{aux} w(k)+D_{aff}
\end{aligned}
$$

$$
E_x x(k)+E_u u(k)+E_{aux} w(k) \leq E_{aff}
$$

- captures relations between real and logical variables
- incorporates constraints
- suitable for control synthesis

How HYSDEL operates

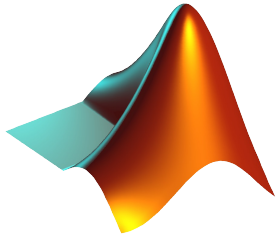


1. User creates a HYSDEL file

HYSDEL

2. Compiler translates the code into mathematical form

3. Output model can be processed by MATLAB



A bit of history

- HYSDEL is available since 2000
- Very successful in industry
  - cement mill control
  - kiln control
  - pump schedule optimization
  - ...

A bit of history

- HYSDEL is available since 2000
- Very successful in industry
  - cement mill control
  - kiln control
  - pump schedule optimization
  - ...
- Contains a lot of shortcomings
- New version is coming with several enhancements

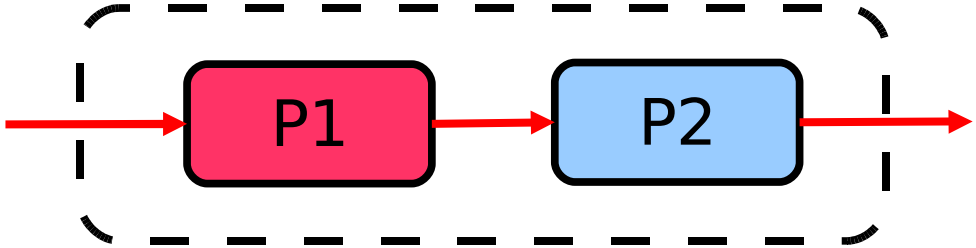
Outline

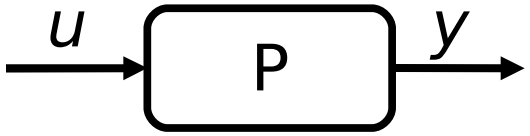
- HYSDEL introduction
- New version HYSDEL 3.0
- Illustrative example
- Conclusion

Features of HYSDEL 3.0

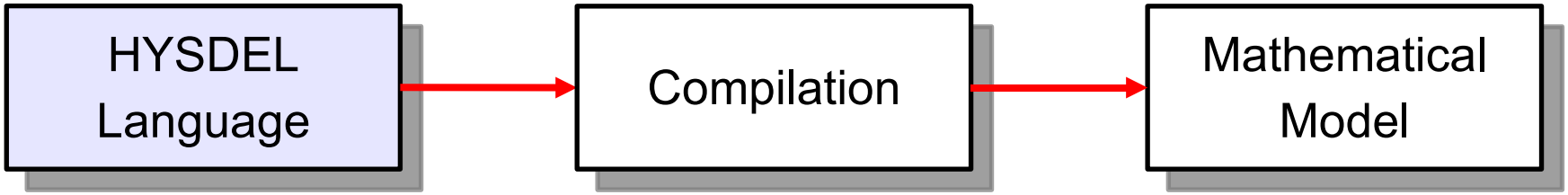
- Extended language
- Merging feature
- Graphical modeling
- Model optimization

HYSDEL





Language extensions



- Language is similar to MATLAB
  - Variables can be defined as vectors/matrices
  - Particular elements can be accessed via indexing
  - FOR loops allowed
- Allows to declare submodels

Example of extended syntax

- Vectors, matrices

```
PARAMETER { REAL A = [1, 2; 3, 4]; }
STATE {
  REAL x(nx*N, 2) [lb, ub];
}
```

- Indexing

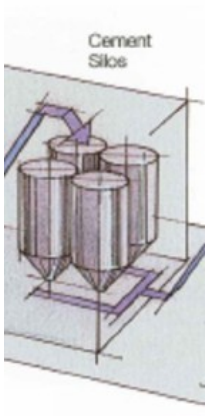
```
PARAMETER { REAL N(2); }
CONTINUOUS {
  x = x(N(1:2), 1:3) + u(2*N);
}
```

- FOR loops

```
FOR (i = 1:N) {
  x(i) = 2*x(N-i+1);
}
```

Submodel declaration

- Motivation:
  - reduce the effort of creating and maintaining complex models
- Approach:
  - allow model hierarchy directly on the language level

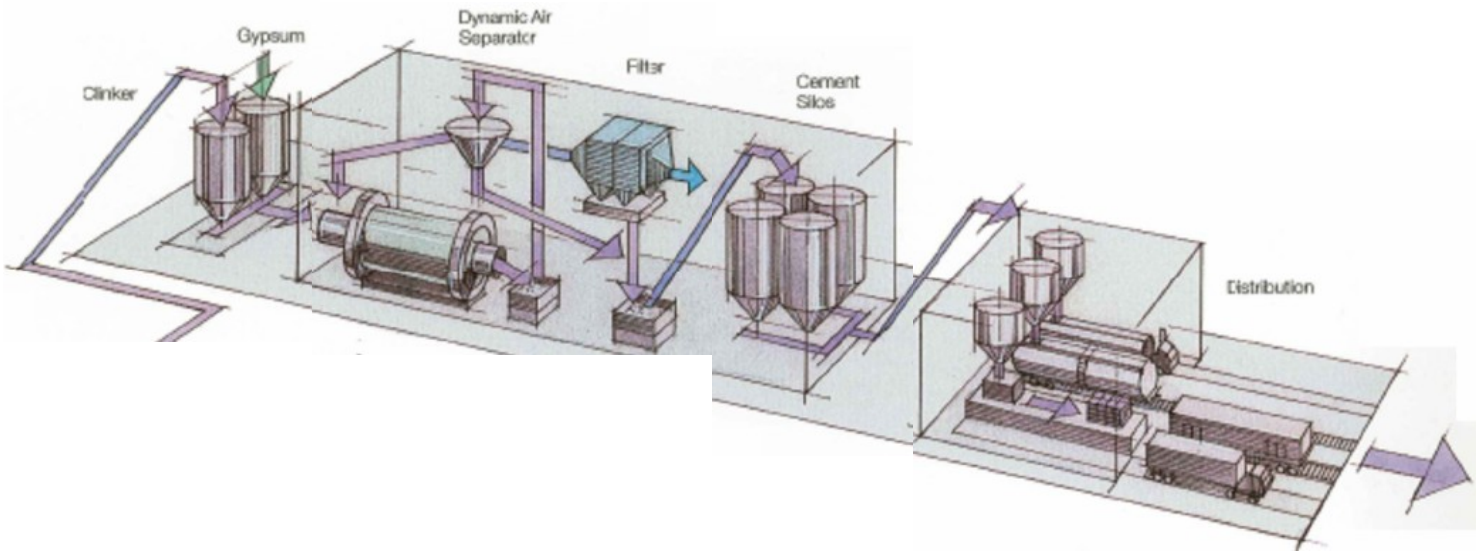



```
MODULE {
 silo  S1, S2, S3, S4;
}
```

➡ **silos.hys**

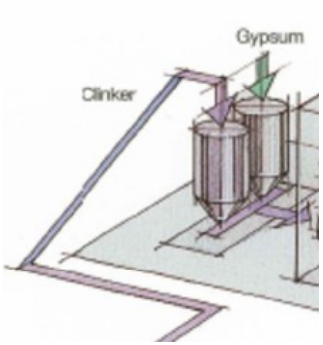
Merging feature

- Illustrative example – cement plant
- Usually different parts of such a complex system are modeled by different people



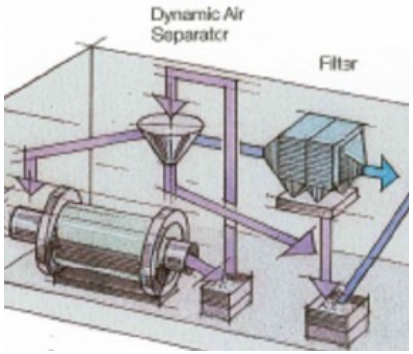
Step 1

- Split plant into parts and create individual modules



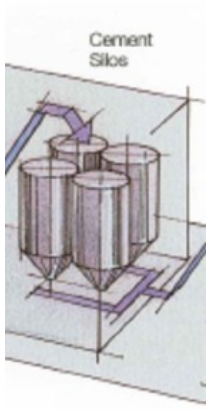
Feeder

feeder.hys



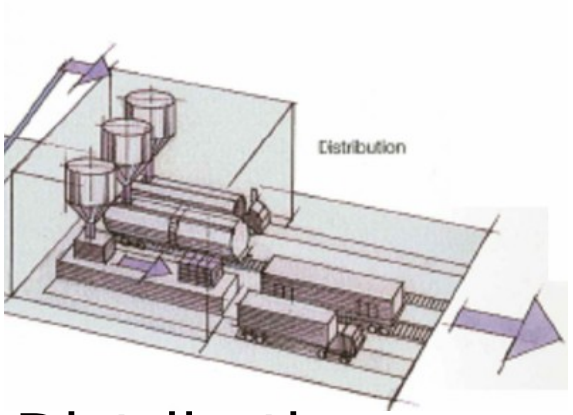
Separator

separator.hys



Silos

silos.hys

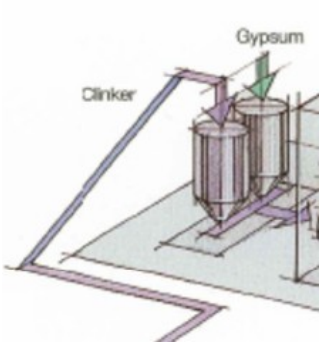


Distribution

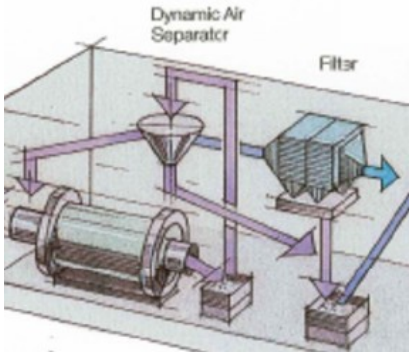
distribution.hys

Step 2

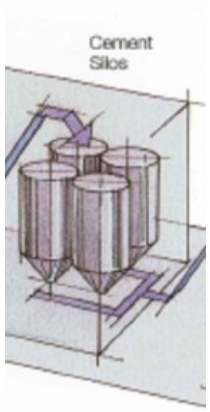
- Define interconnections between modules



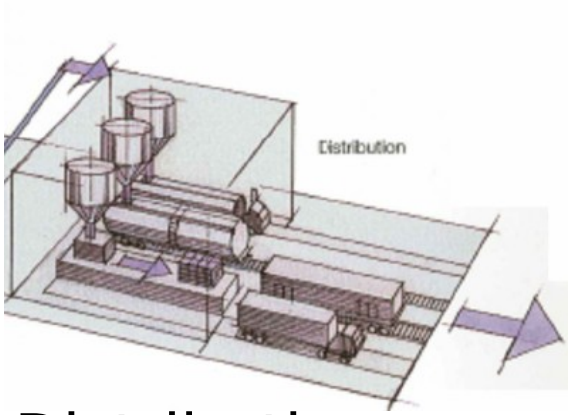
Feeder



Separator



Silos

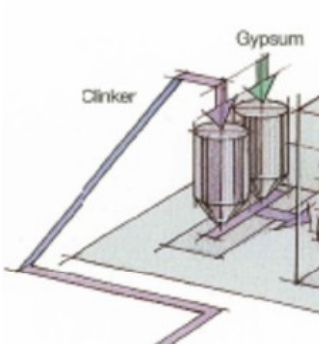


Distribution

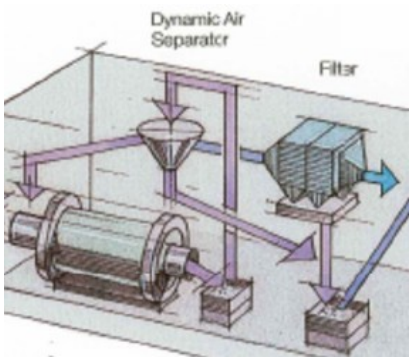
```
feeder.output     = separator.input
```

Step 2

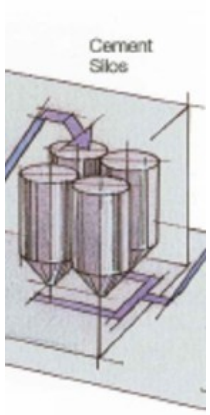
- Define interconnections between modules



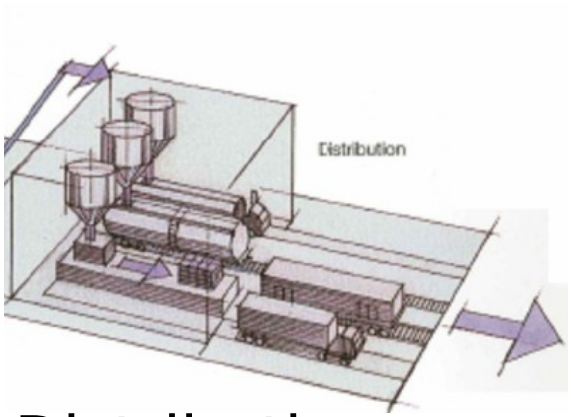
Feeder



Separator



Silos

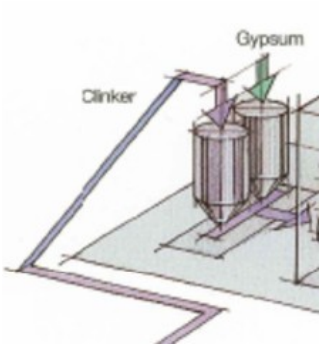


Distribution

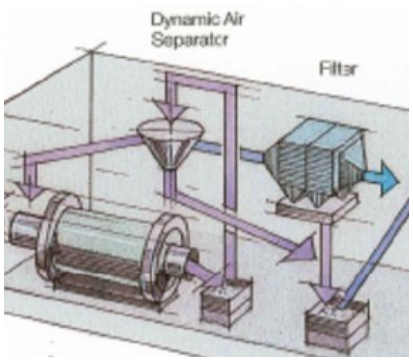
```
feeder.output    = separator.input
separator.output = silos.input
```

Step 2

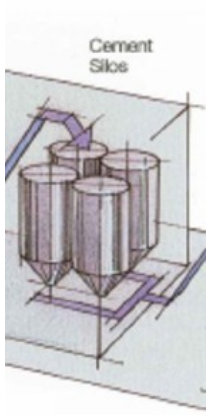
- Define interconnections between modules



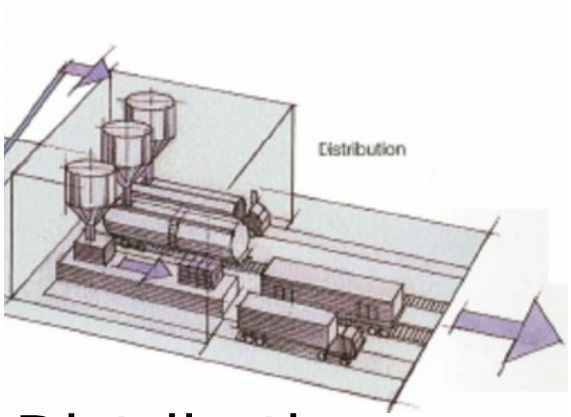
Feeder



Separator



Silos



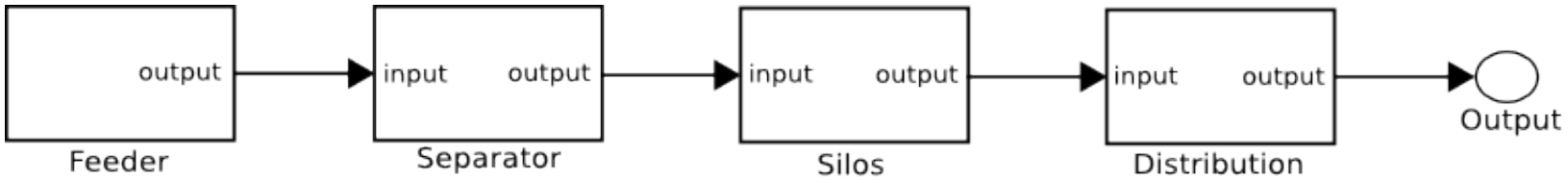
Distribution

```
feeder.output    = separator.input
separator.output = silos.input
silos.output     = distribution.input
```

Graphical level

- Why doing merging of submodels manually?
- Use Simulink to draw connections between submodels!

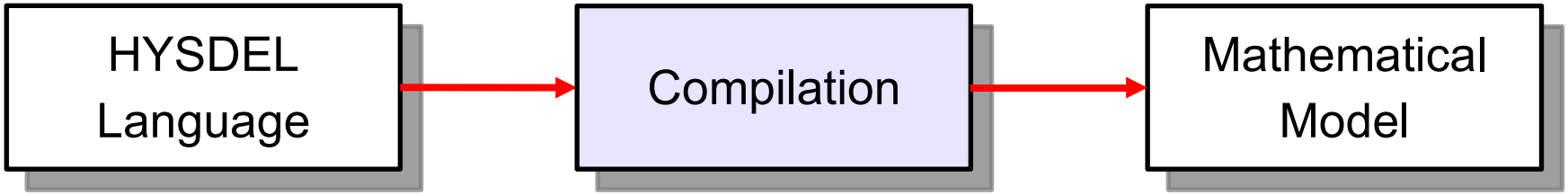
1. Create Simulink scheme



2. Let HYSDEL do the rest

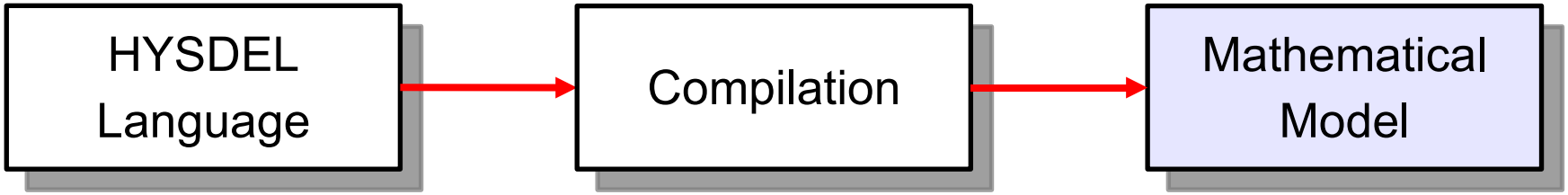
production.hys

Translation process



- Compiler is based on the YALMIP package
  - easy to maintain
  - platform independent
  - provides means to improve the quality of the model

Exploiting the model



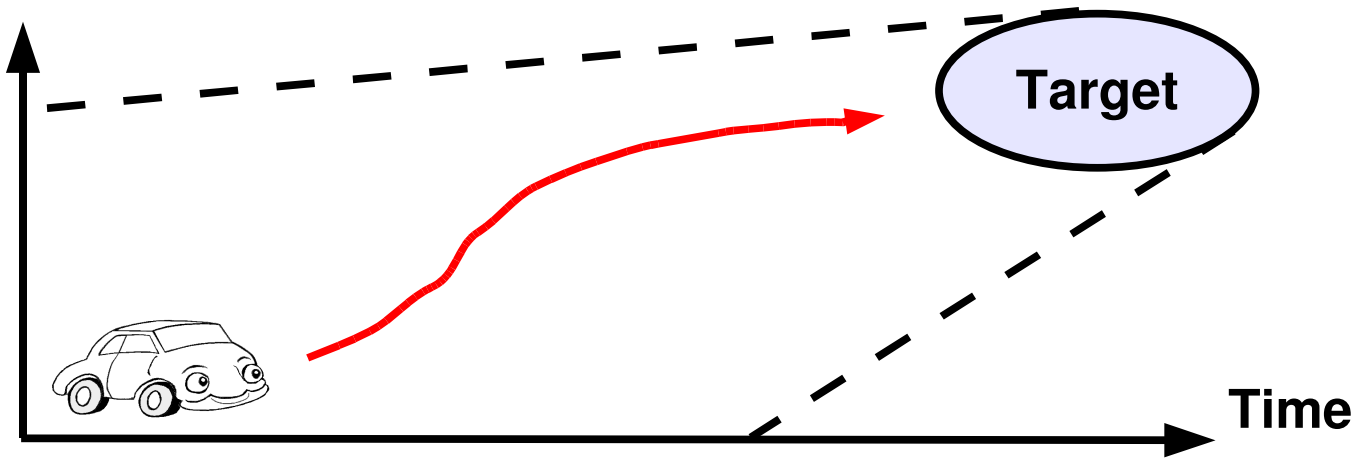
- Interoperable with MPT toolbox
  - model analysis
  - simulation in Matlab & Simulink
  - control design

Outline

- HYSDEL introduction
- New version HYSDEL 3.0
- Illustrative example
- Conclusion

Control of a hybrid car

- Task:
  - create model “turbo_car.hys”
  - design a predictive controller
- Plant characteristics
  - 3 real states
  - 1 real and 1 logical input
  - constraints on states/inputs



Operating modes

- Hybrid nature comes from input switch “TURBO”
  - Normal mode

$$\boldsymbol{z}(k)=\boldsymbol{u}(k)$$

  - Turbo mode
    - input signal is doubled
    - can last only 10 sampling times

$$\boldsymbol{z}(k)=2\boldsymbol{u}(k)$$

- Model description

$$\boldsymbol{x}(k+1)=\boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{B}\boldsymbol{z}(k)$$

HYSDEL code

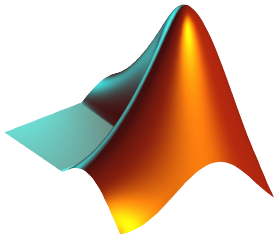
```
SYSTEM turbo_car {
  INTERFACE {
    STATE {   REAL position [-50, 50];
              REAL velocity [-10, 10];
              REAL turbocount [0, -10];  }
    INPUT {   REAL acc [-1, 1];
              BOOL turbo;    }
    OUTPUT {   REAL y;    }
  }
  IMPLEMENTATION {
    AUX {   REAL z;   }
    DA {
          z = {IF turbo THEN 2*acc ELSE acc};
    }
    CONTINUOUS {
      position = position + velocity + z;
      velocity = velocity + 0.5*z;
      turbocount = turbocount - (REAL turbo);
    }
    OUTPUT {   y = position; }
  }
}
```

declaration of variables and parameters

implementation of variable relations

Predictive control design

- Obtain model "*F*" using MPT toolbox



```
>> F = mpt_sys('turbo_car');
```

- Define optimal control problem "*P*"

$$\min_{u_k} \sum_{k=1}^{N} |Q(x_k - r)| + |Ru_k|$$

$$\text{subject to} \begin{cases} x_{k+1} = F(x_k, u_k) \\ x_k \in X \\ u_k \in U \end{cases}$$

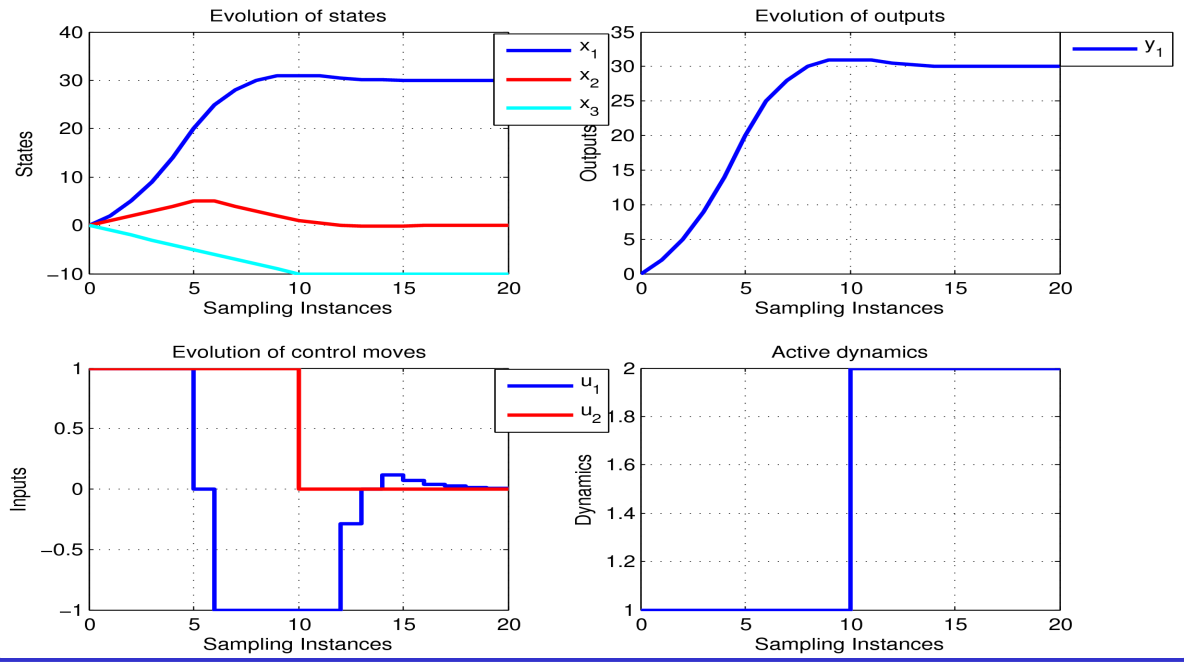
```
>> P.N = 5;
>> P.Q = [1, 1];
>> P.R = 1;
>> P.norm = 1;
>> P.xref = 30;
```

Results

- Calculate predictive controller

```
>> controller = mpt_control(F, P);
```

- Closed loop simulation



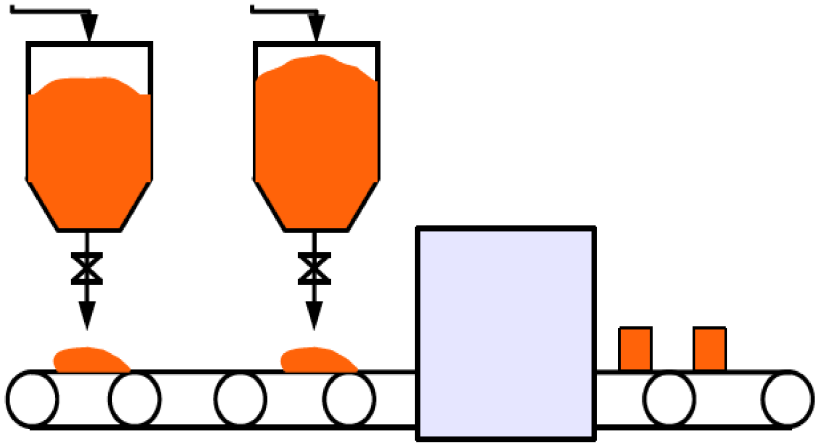
Conclusion

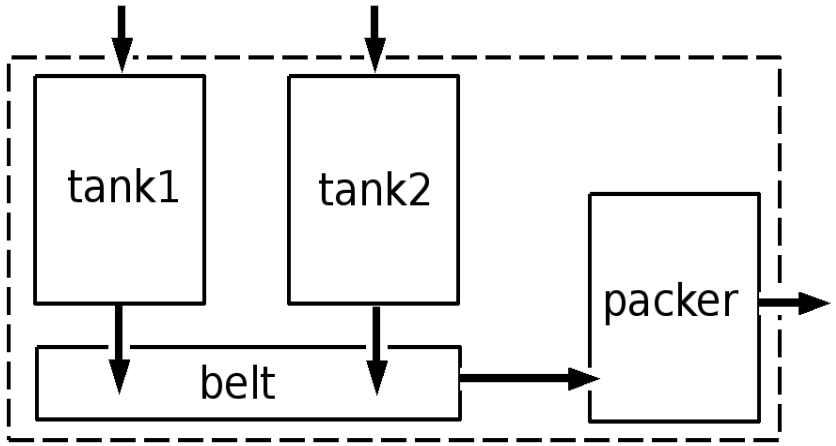
- HYSDEL generate hybrid models suitable for analysis and control design
- HYSDEL 3.0 offers
  - extended syntax for easier modeling (vectors, matrices, FOR-loops)
  - model merging for modeling of complex systems
  - generation of better quality models for a more efficient control synthesis
- HYSDEL 3.0 will be publicly available soon

Additional slides

Production system

- Task:
  - create model using HYSDEL
  - simulate the outputs
- Plant characteristics:
  - 3 dynamical systems
  - 1 static system
  - ON/OFF switches
  - constraints on variables





Processing using HYSDEL

- Create individual files
  - tank.hys
  - belt.hys
  - packer.hys
- Merge them into one master and simulate
  - production.hys