Nonlinear Loop Replacement Applied to Realtime Capable Jet Pump Model

MODPROD Workshop

Linköping, 06.-07.02.2024

Oliver Lenord[1],
Andreas Heuermann[2],
Alexander Fischer[1]
Bernhard Bachmann[2],

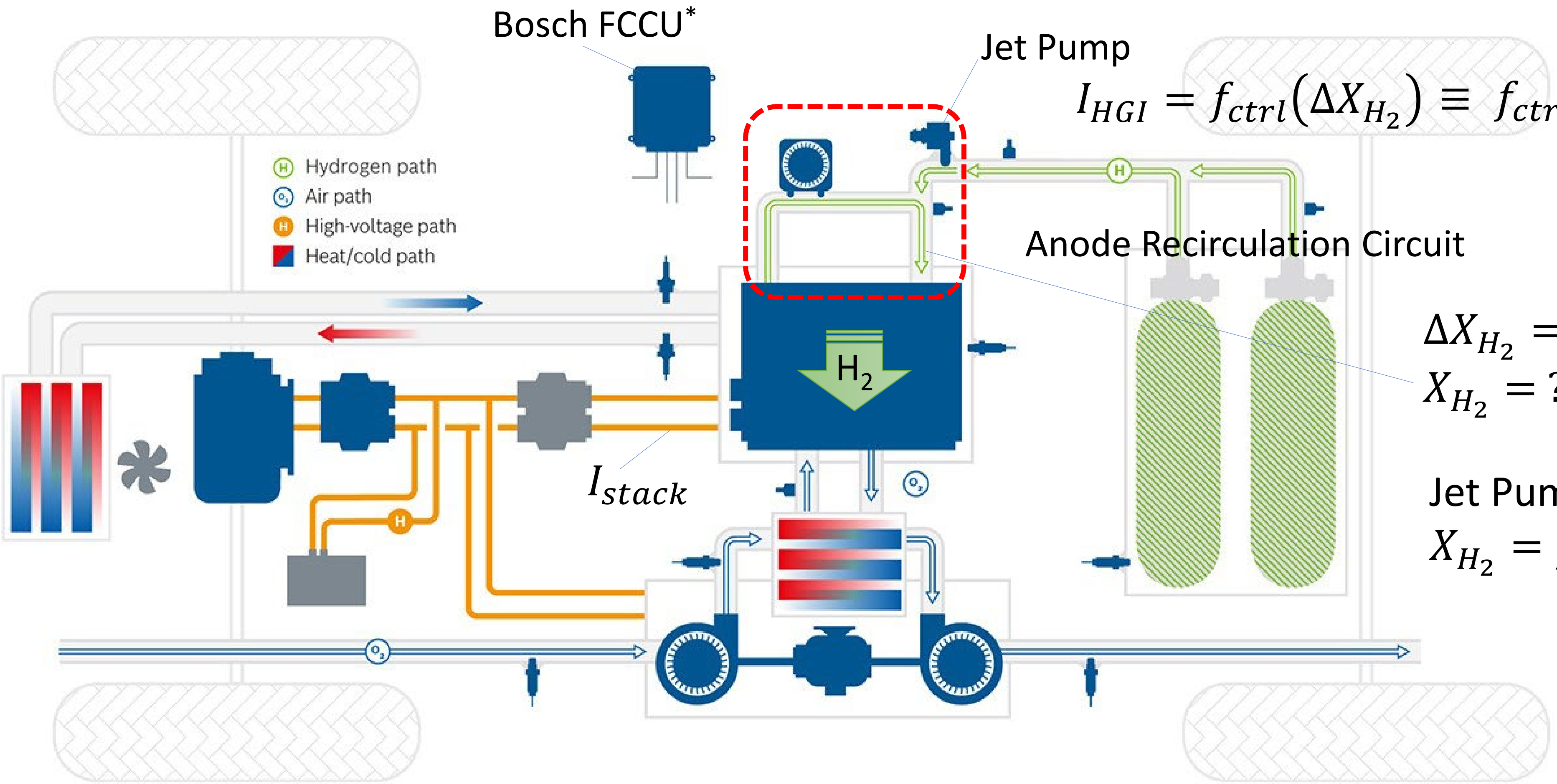
1) Robert Bosch GmbH, Germany
2) Hochschule Bielefeld
University of Applied Sciences and Arts, Germany

Funded by the European Union
NextGenerationEU

Supported by:
Federal Ministry for Economic Affairs and Climate Action
on the basis of a decision by the German Bundestag

Content

Application background



$$I_{HGI} = f_{ctrl}(\Delta X_{H_2}) \equiv f_{ctrl}(p_i, T_i, I_{stack})$$

$$\Delta X_{H_2} = X_{H_2,\text{set}} - X_{H_2}$$

$$X_{H_2} = ?$$

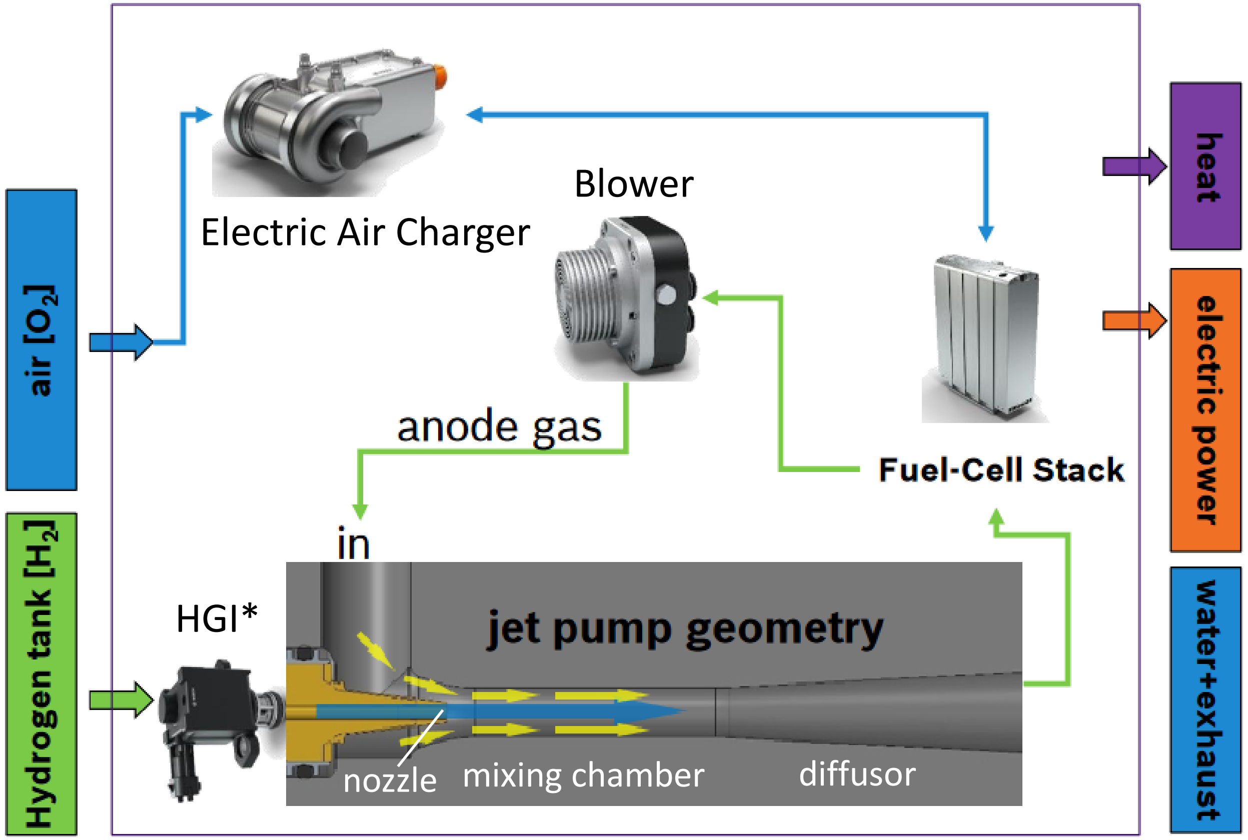
Jet Pump Model:

$$X_{H_2} = f_{JP}(p_{1,2,3}, T_{1,2,3}, X_{i_{reci}})$$

Fuel cell electric drive system from Bosch Mobility Solutions

*Fuel Cell Control Unit

Application background



Jet pump in the anode recirculation path of a fuel cell electric drive system

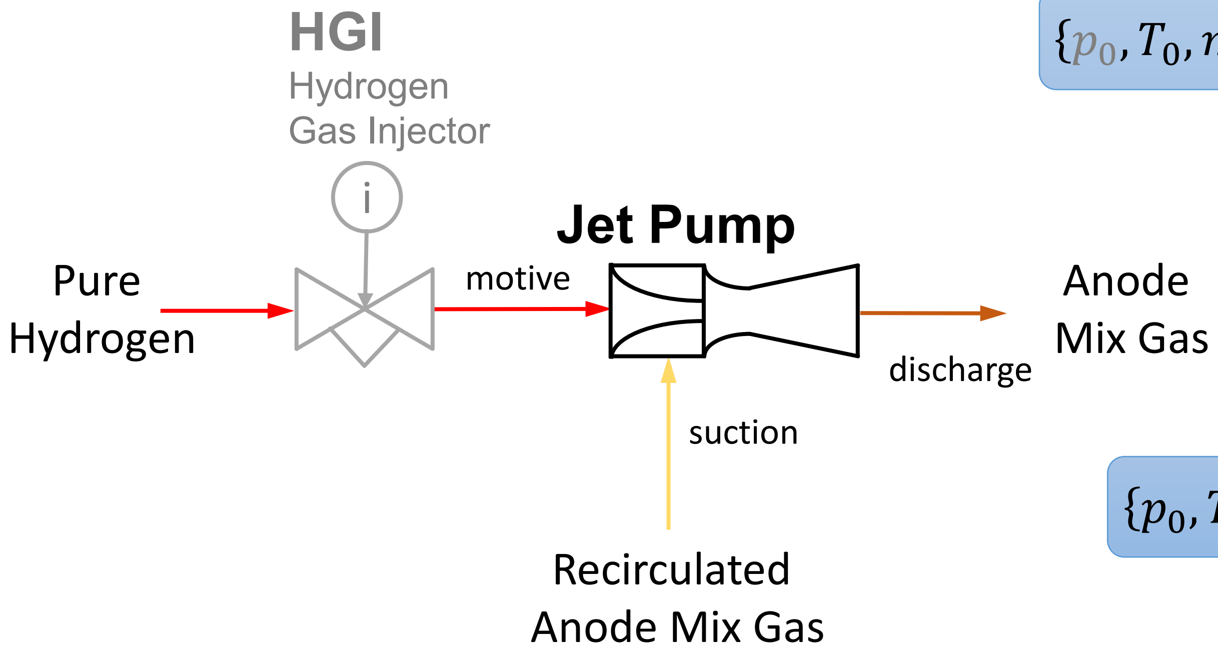
*Hydrogen Gas Injector

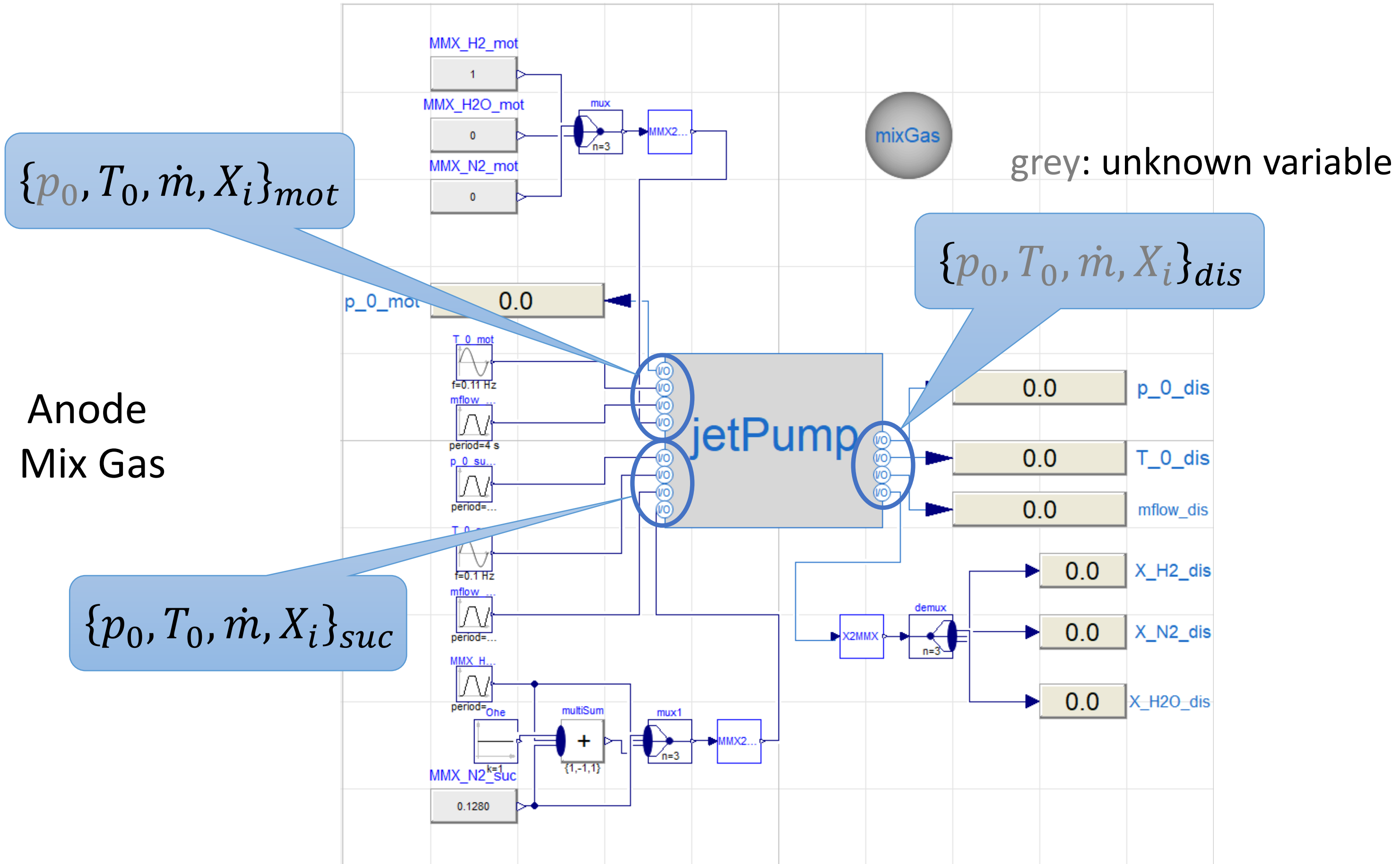
Realtime requirements on the model

- Realtime capable model of the jet pump:
  - Guaranteed worst case execution time.
  - Robust at a fixed sample rate.
  - Low computational cost and low memory footprint.
  - Mathematical operators available in embedded software service library.
  - Reliable results over the entire operation range.
- Jet pump model requirements:
  - No iterative solvers → No NLS → No non-linear algebraic loops.
  - No high dimensional look-up tables.
  - No overfitting, no outliers.

*Non-linear systems of equations

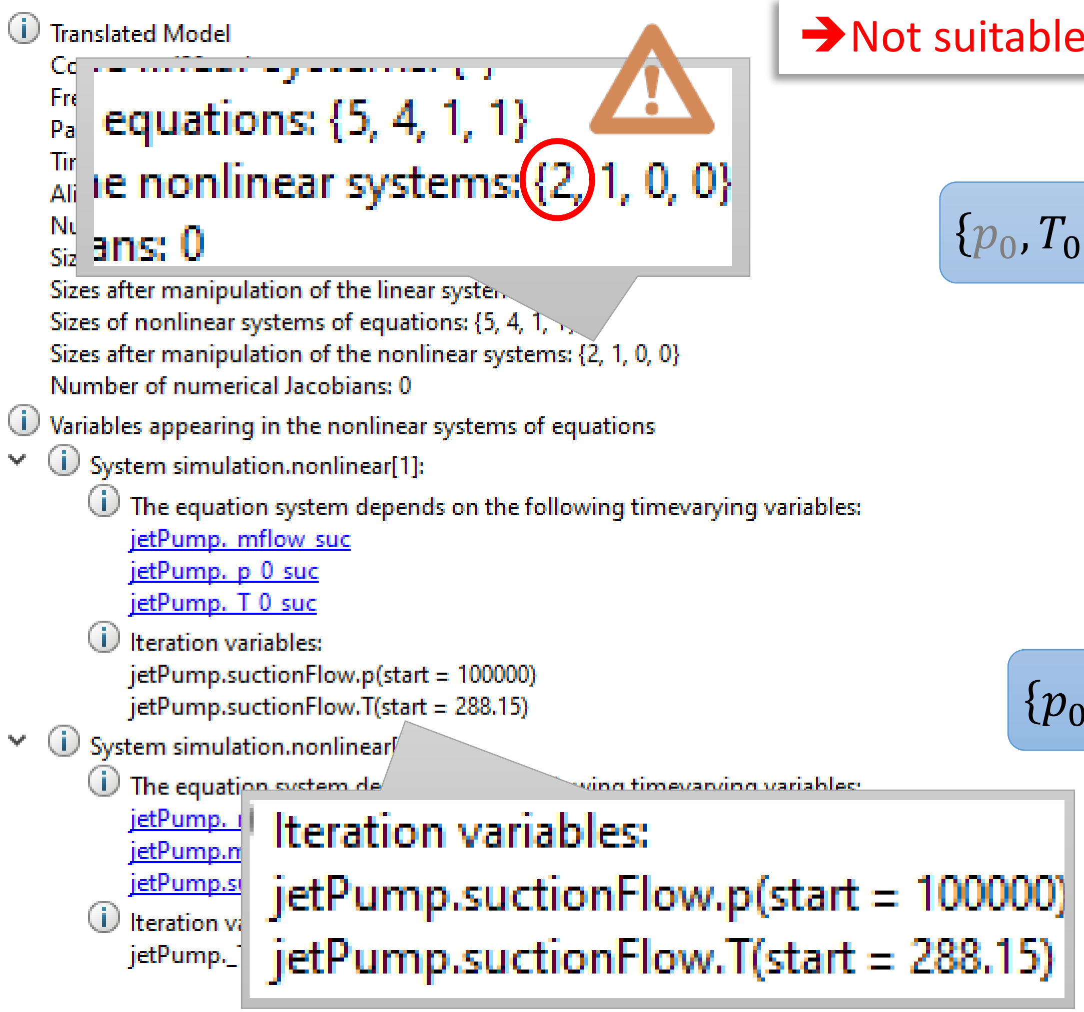
State-of-the-art jet pump model



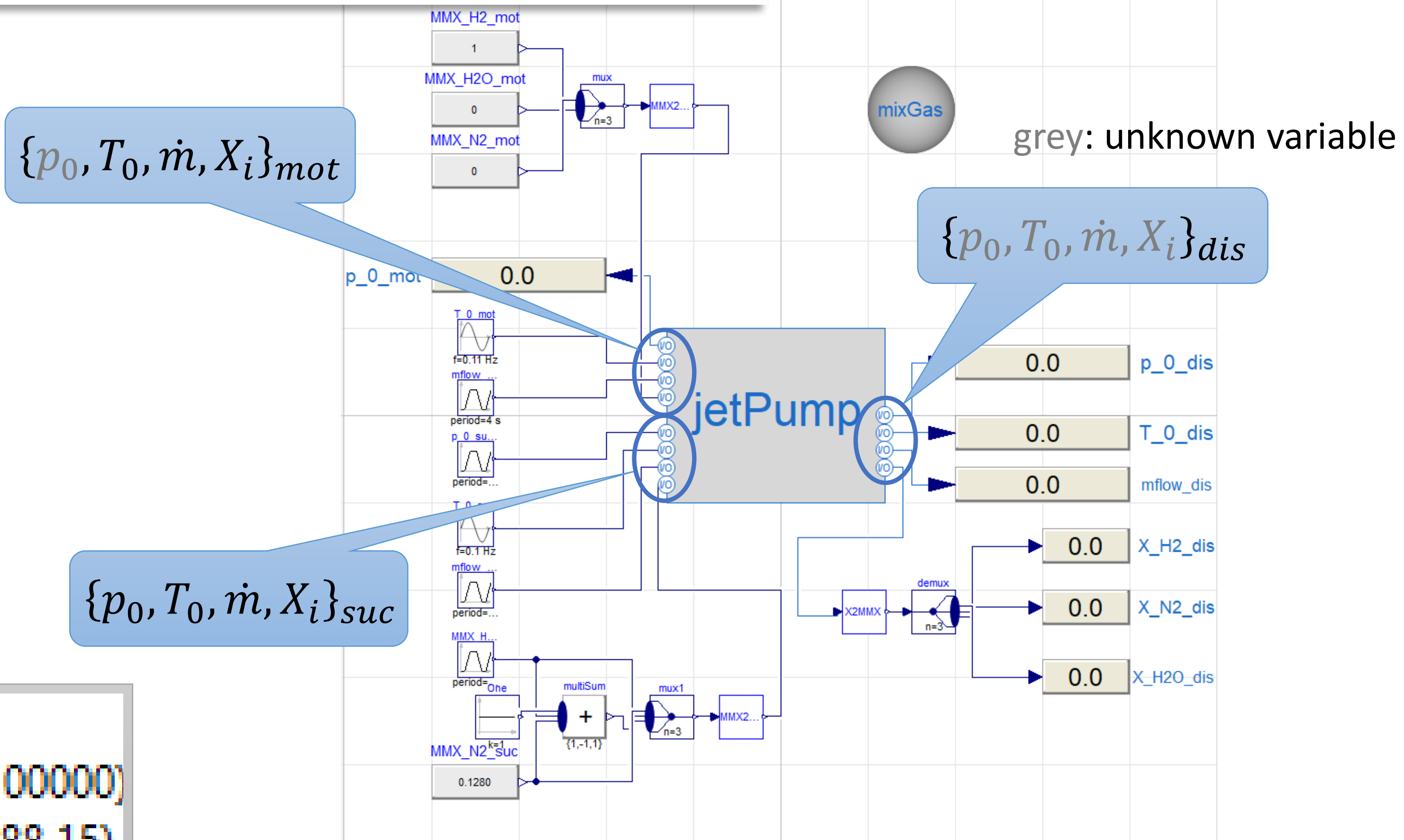


Causality applicable for model validation against CFD results

State-of-the-art jet pump model


Translated Model
Sizes after manipulation of the linear systems
Sizes of nonlinear systems of equations: {5, 4, 1, 1}
Sizes after manipulation of the nonlinear systems: {2, 1, 0, 0}
Number of numerical Jacobians: 0
Variables appearing in the nonlinear systems of equations
System simulation.nonlinear[1]:
The equation system depends on the following timevarying variables:
jetPump._mflow_suc
jetPump._p_0_suc
jetPump._T_0_suc
Iteration variables:
jetPump.suctionFlow.p(start = 100000)
jetPump.suctionFlow.T(start = 288.15)
System simulation.nonlinear[
The equation system de... timevarying variables:
Iteration v...

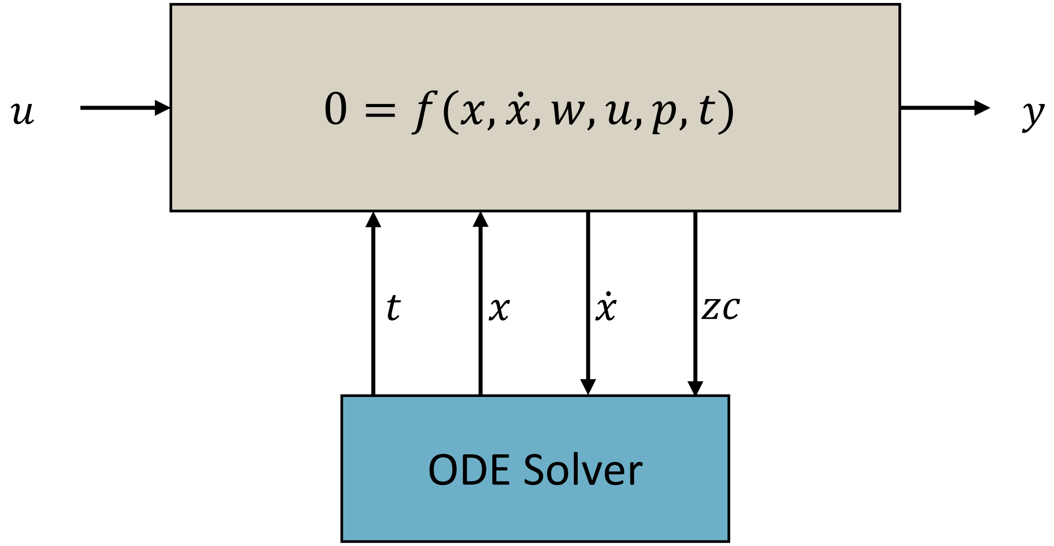
→ Not suitable for embedded control function!



grey: unknown variable

Causality applicable for model validation against CFD results

Replacing Non-Linear Systems



t	Time
p	Parameters
$u(t)$	Inputs
$x(t)$	Continuous states
$\dot{x}(t)$	State derivative
$w(t)$	Local variables
$y(t)$	Outputs
$zc(t)$	Event indicators

Replacing Non-Linear Systems

PHyMOS

$$z(t) := \begin{pmatrix} \dot{x}(t) \\ w(t) \end{pmatrix}$$ system unknowns

$$\begin{aligned} f_1(z_3, z_4) &= 0 \\ f_2(z_2) &= 0 \\ f_3(z_2, z_3, z_5) &= 0 \\ f_4(z_1, z_2) &= 0 \\ f_5(z_1, z_3, z_5) &= 0 \end{aligned}$$

$$\begin{aligned} f_2(z_2) &= 0 \\ f_4(z_1, z_2) &= 0 \\ f_{NLS}(z_1, z_2, z_3, z_5) &= 0 \\ f_1(z_3, z_4) &= 0 \end{aligned}$$

Machine learning → surrogate f_S

$$\begin{aligned} f_2(z_2) &= 0 \\ f_4(z_1, z_2) &= 0 \\ f_S(z_1, z_2) &= z_3, z_5 \\ f_1(z_3, z_4) &= 0 \end{aligned}$$

	z_1	z_2	z_3	z_4	z_5
f_1	0	0	1	1	0
f_2	0	1	0	0	0
f_3	0	1	1	0	1
f_4	1	1	0	0	0
f_5	1	0	1	0	1

before sorting and matching

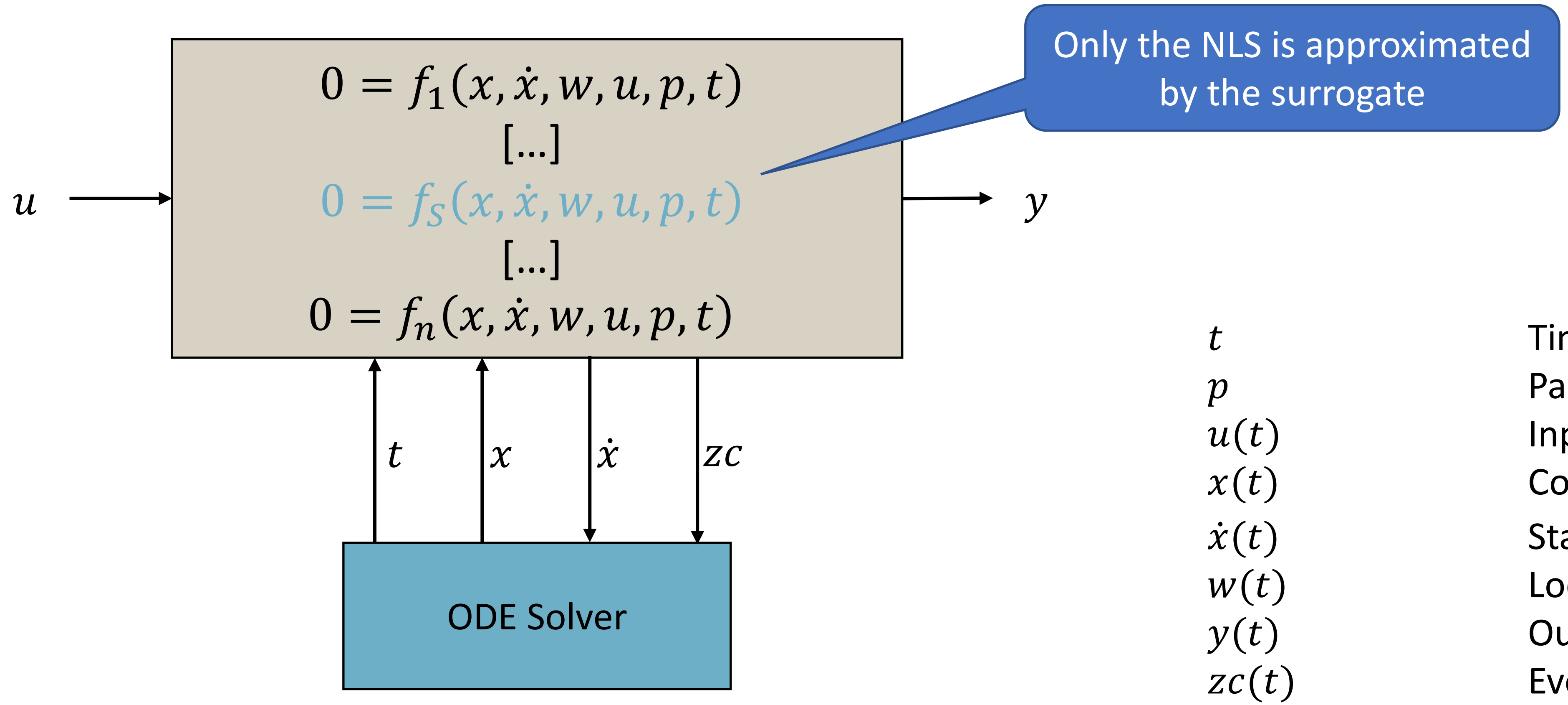
	z_2	z_1	z_3	z_5	z_4
f_2	1	0	0	0	0
f_4	1	1	0	0	0
f_3	1	0	1	1	0
f_5	0	1	1	1	0
f_1	0	0	1	0	1

implicit

	z_2	z_1	z_3, z_5	z_4
f_2	1	0	0	0
f_4	1	1	0	0
f_S	1	1	1	0
f_1	0	0	1	1

explicit

Replacing Non-Linear Systems



t	Time
p	Parameters
$u(t)$	Inputs
$x(t)$	Continuous states
$\dot{x}(t)$	State derivative
$w(t)$	Local variables
$y(t)$	Outputs
$zc(t)$	Event indicators

The “Loop Replacement” method

PHyMOS

Pre-requisite

- Create simulation scenario covering the required operation range.

Workflow

1. Identify and extract NLS ☑ — OM
2. Generate training data ☑ — julia*
3. Train surrogate model ☑ — julia*
4. Replace NLS with surrogate ☐ — manual

Verification

- Compare results of simulation scenario.

* https://github.com/AnHeuermann/NonLinearSystemNeuralNetworkFMU.jl

Application to the Jet Pump Model

1. Identify and extract NLS

NLS occurs in the SuctionFlow model

Iteration variables:
jetPump.suctionFlow.p(start = 100000)
jetPump.suctionFlow.T(start = 288.15)

$$p = f(Ma)$$
$$T = f(Ma)$$
$$Ma = f(p, T)$$

```
partial model EQS_SuctionFlow "Suction flow equation 85 to 90"

  input Modelica.Units.SI.AbsolutePressure p_0 "total pressure at inlet";
  input Modelica.Units.SI.Temperature T_0 "total temperature at inlet";

  input Modelica.Units.SI.AbsolutePressure p "static pressure at outlet";
  output Modelica.Units.SI.Temperature T = 338 "static temperature at outlet";
  output Modelica.Units.SI.MassFlowRate mflow = 0 "mass flow rate";
  Modelica.Units.SI.MachNumber Ma "Mach number at outlet";
  Modelica.Units.SI.VelocityOfSound c "velocity of sound at outlet";
  Modelica.Units.SI.Velocity v "fluid velocity at outlet";
  output Modelica.Units.SI.ImpulseFlowRate I = 0 "impulse at outlet";

  parameter Modelica.Units.SI.Area A=7.5775215E-05 "area of outlet";
  parameter Real k=1.4 "specific isentropic exponent";
  parameter Modelica.Units.SI.SpecificHeatCapacity R_s=716.41 "specific gas constant";

equation
  p_0/p = (1+(k-1)/2 * Ma^2)^(k/(k-1)) "eq. 85";
  T_0/T = (1+(k-1)/2 * Ma^2) "eq. 86";
  v = R_s * mflow/A * T/p "eq. 87";
  c = sqrt(k*R_s*T) "eq. 88";
  Ma = v/c "eq. 89";
  I = v * mflow "eq. 91";

  a
end EQS_SuctionFlow;
```

1)

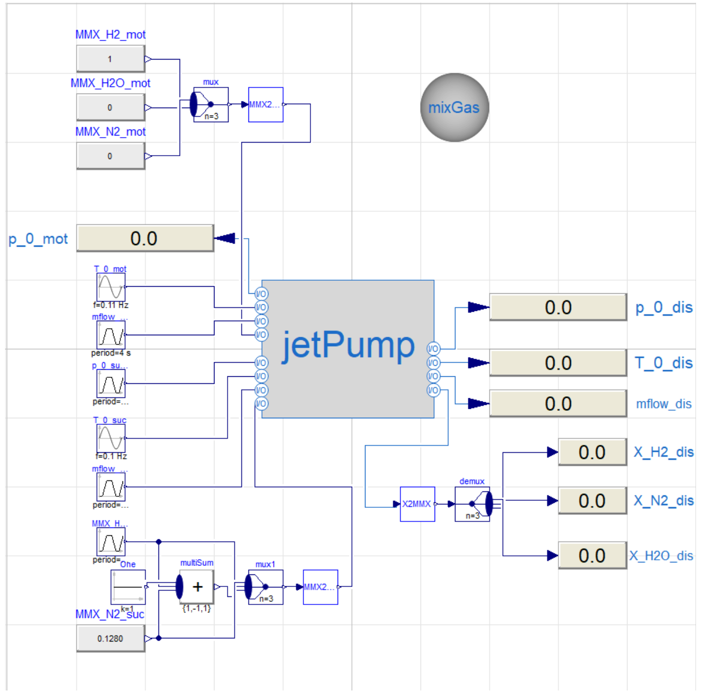
Functional Mock-Up Interface

- Derive value ranges of the loop inputs from scenario

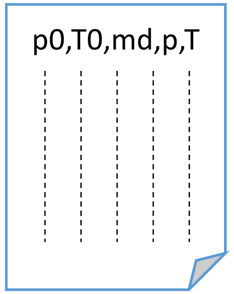
- Evaluate FMU to solve:

$$p = f(Ma)$$
$$T = f(Ma)$$
$$Ma = f(p, T)$$

- Sample the input ranges
- Record as training data set

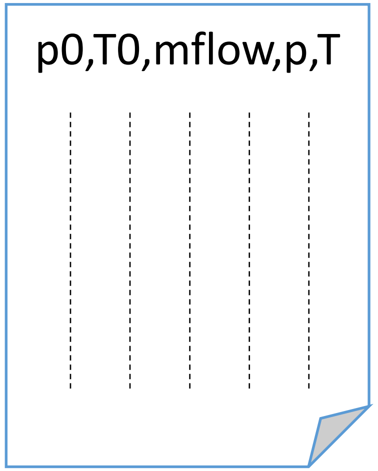


2)



Application to the Jet Pump Model
3. Train surrogate model

- Define accepted <u>operators</u>:
  - +, -, *, /, sqrt, …
- Run <u>symbolic regression</u> on subset of training data
- Determine RMSE on validation data
- Output the found <u>symbolic expressions</u>



3)

Surrogate.mo

```
p = ((p_0 - (sqrt(T_0)))
    * (0.5483443 + (1.7069604 * sqrt(sqrt(mflow))))
  - (1 * (-0.004339092 + mflow) * (5.246307e6 + T_0))
   "eq. 85 surrogate";

T = T_0 * (1.0472907 - (mflow * (15.949435
  - (0.20818402 * sqrt(mflow * (p_0 - (2.307301 * T_0)))))))
    "eq. 86 surrogate";
```

Application to the Jet Pump Model
4. Replace NLS with surrogate

- Duplicate SuctionFlow class
  - Replace equations with symbolic expressions
- Integration into validation scenario
  - Make suctionFlow component replaceable within validation scenario
  - Redeclare suctionFlow component as SuctionFlow_surr within validation scenario

Surrogate.mo

```
p = ((p_0 - (sqrt(T_0)))
     * (0.5483443 + (1.7069604 * sqrt(sqrt(mflow))))
  - (1 * (-0.004339092 + mflow) * (5.246307e6 + T_0))
   "eq. 85 surrogate";

T = T_0 * (1.0472907 - (mflow * (15.949435
  - (0.20818402 * sqrt(mflow * (p_0 - (2.307301 * T_0)))))))
     "eq. 86 surrogate";
```

4)

SuctionFlow_surr.mo

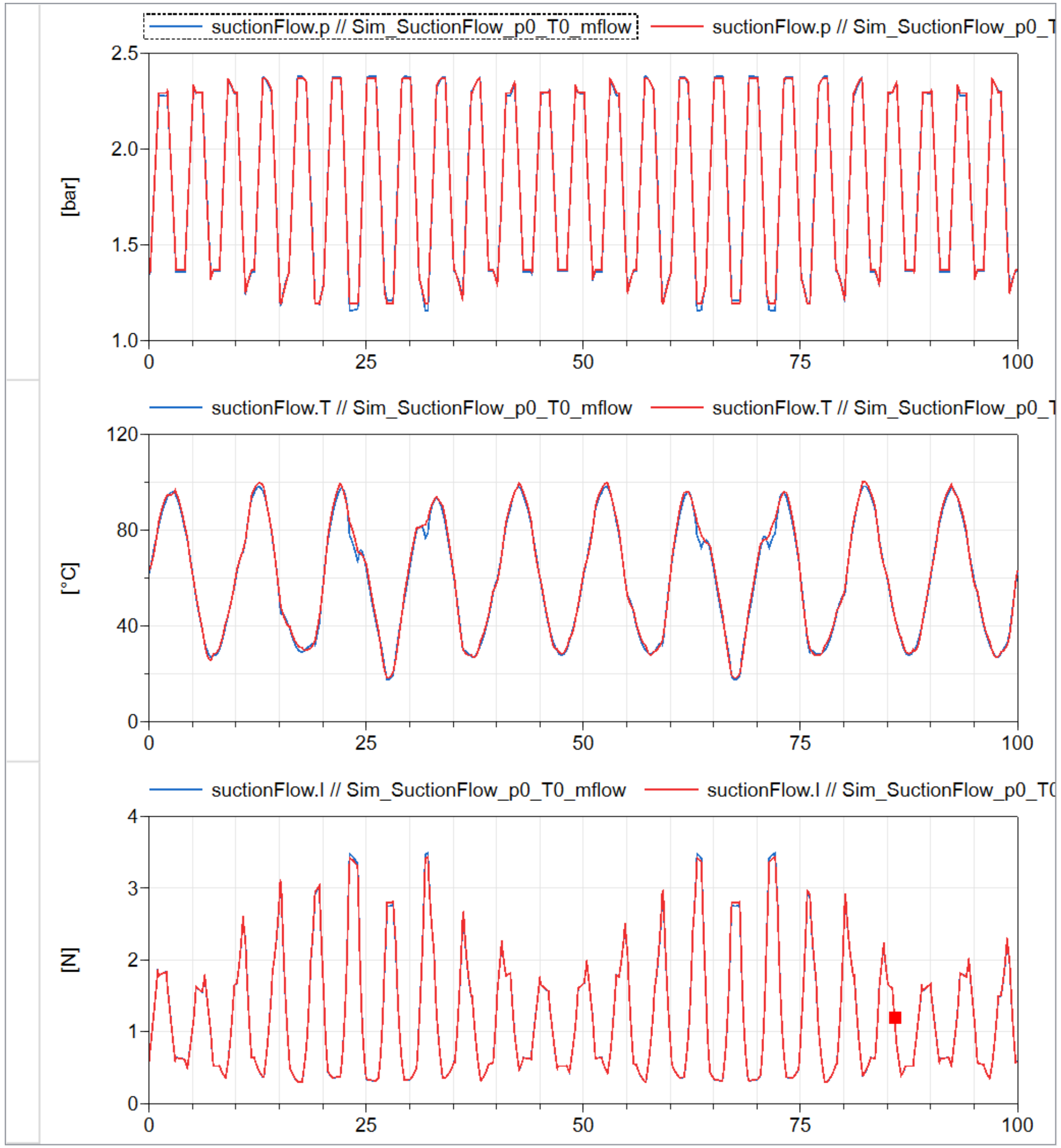
```
partial model EQS_SuctionFlow_surr
  "Suction flow equation 85 to 90 replaced with surrogate eqs"
    ⋮
equation
  p = ((p_0 - (sqrt(T_0))) * (0.5483443 + (1.7069604 * sqrt(sqrt(mflow))))
    - (1 * (-0.004339092 + mflow) * (5.246307e6 + T_0)) "eq. 85 surrogate";
  T = T_0 * (1.0472907 - (mflow * (15.949435
    - (0.20818402 * sqrt(mflow * (p_0 - (2.307301 * T_0)))))))"eq. 86 surrog
  v = R_s * mflow/A * T/p "eq. 87";
  c = sqrt(k*R_s*T) "eq. 88";
  Ma = v/c "eq. 89";
  I = v * mflow "eq. 91";

  a
end EQS_SuctionFlow_surr;
```

Application to the Jet Pump Model

Verification against original model

suctionFlow vs. suctionFlow_surr

- Max abs. error:
  - p = 0.04bar (3.3%)
  - T = 5K (6.3%)
  - I = 0.06N (1.8%)
- Mean abs. error
  - p = 0.008bar (0.66%)
  - T = 1.1K (1.4%)
  - I = 0.007N (0.22%)

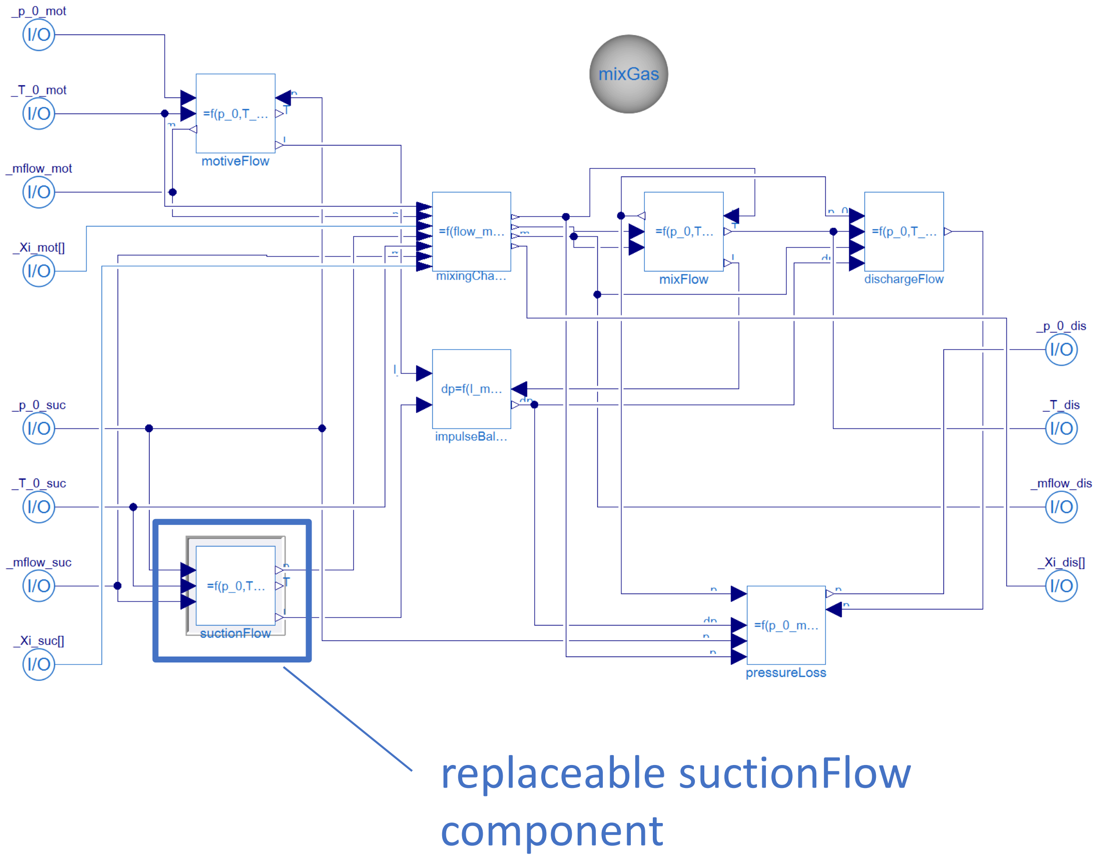


[1] with respect to size of value range

[2] Euler solver, PC

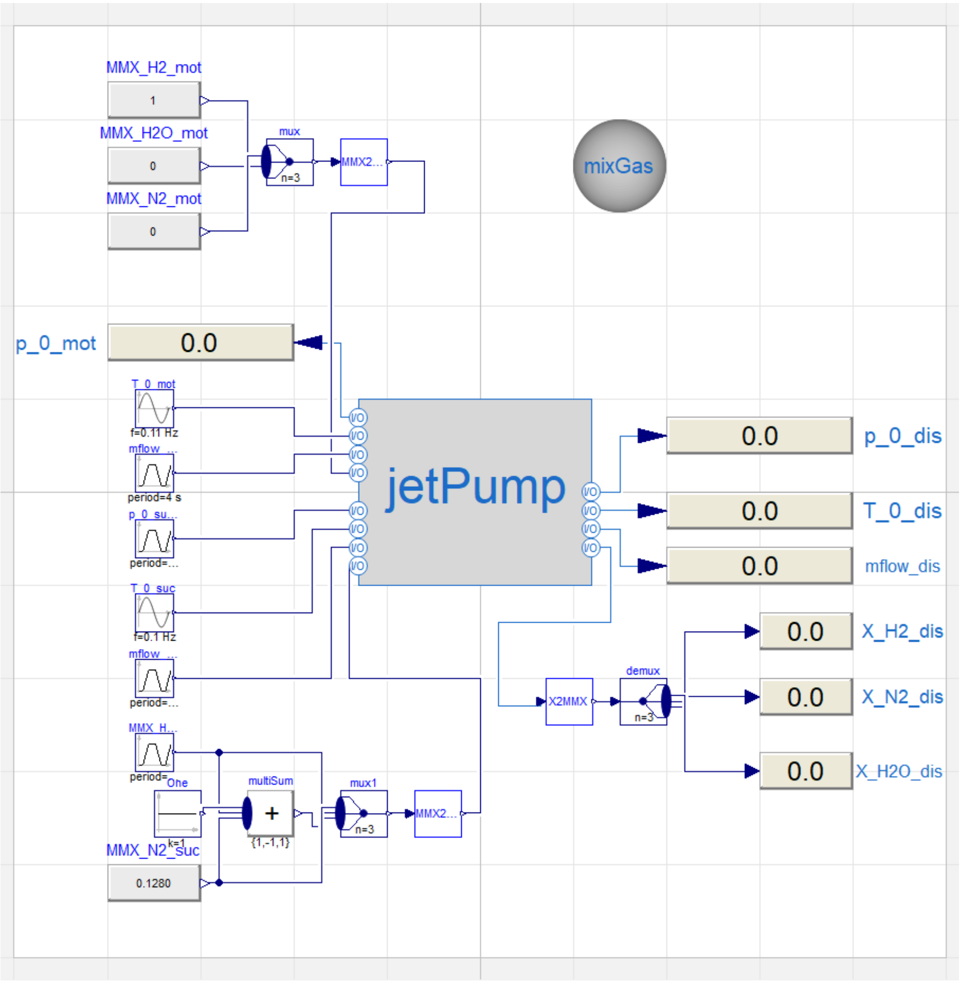
Application to the Jet Pump Model

- Integration of the surrogate into Jet Pump Model:
  - Make suctionFlow component replaceable in JetPump class.



Application to the Jet Pump Model

- Integration of the surrogate into Jet Pump Model:
  - Make suctionFlow component replaceable in JetPump class.
  - Redeclare suctionFlow component as SuctionFlow_surr within jetPump.



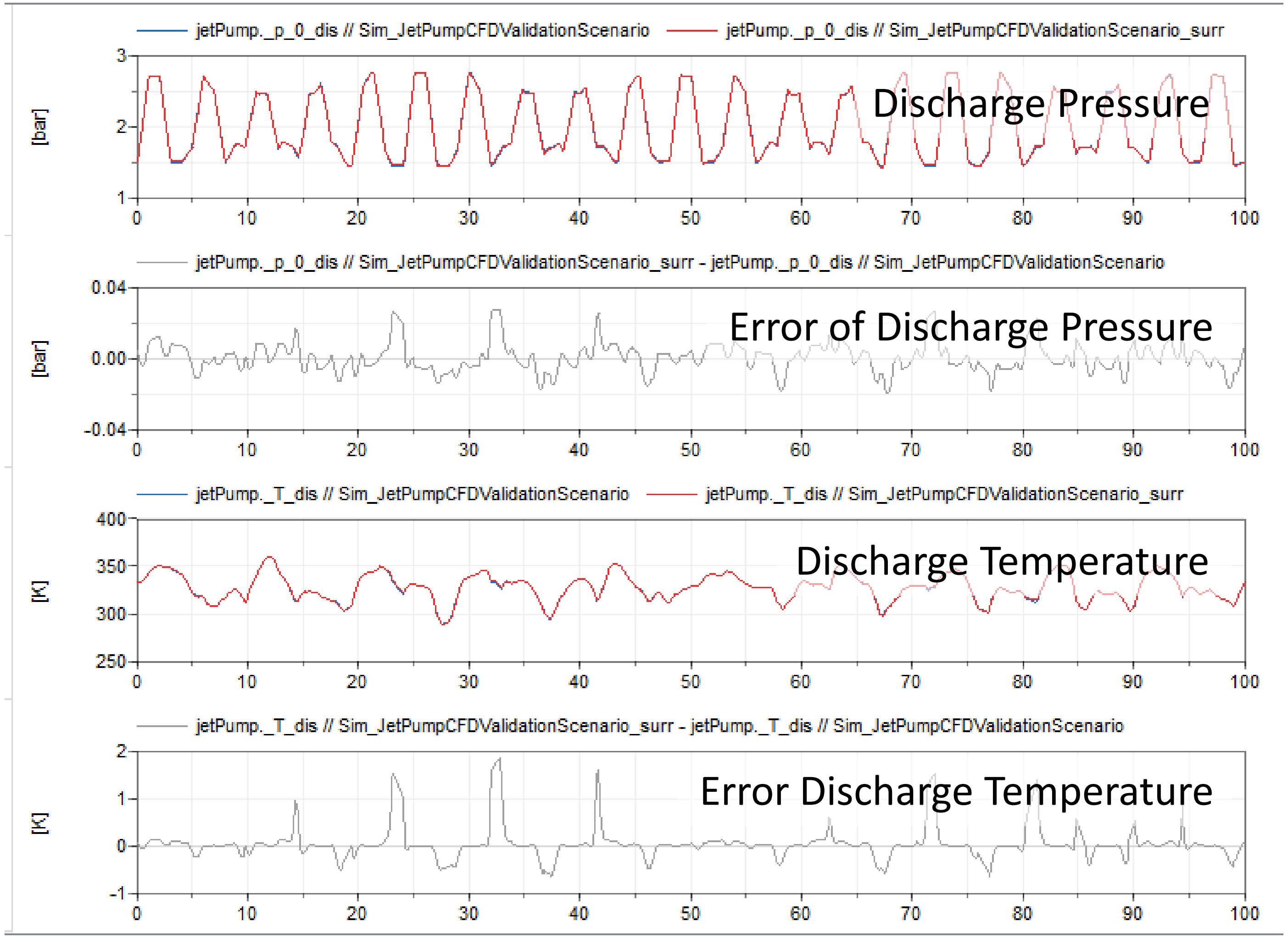
Application to the Jet Pump Model

Validation against original model

jetPump vs. jetPump_surr

- Max error abs. (rel.):
  - p_0_dis = 0.028bar (2.1%)[1)]
  - T_0_dis = 1.8K (2.6%)[1)]
- Mean error abs. (rel.)
  - p_0_dis = 0.006bar (0.46%)[1)]
  - T_0_dis = 0.97K (1.4%)[1)]

	size NLS	CPU time[2)]	speed-up
jetPump	{2,1]	12ms	-
jetPump_surr	{0,1}	6ms	2x



1) with respect to size of value range

2) Euler solver, PC

Realtime requirements on the model

PHyMOS

- Realtime capable model of the jet pump.
  - Guaranteed worst case execution time. ✓ Executable with Euler
  - Robust at a fixed sample rate. ✓ Results not dependent on step size
  - Low computational cost and low memory footprint. ✓ 2x speed-up, only few constants added
  - Mathematical operators available in embedded software service library. ✓ Basic math only
  - Reliable results over the entire operation range. ✓ Training data from dense sampling
- Jet pump model requirements
  - No iterative solvers → No NLS → No non-linear algebraic loops. ✓ replaced with surrogate
  - No high dimensional look-up tables. ✓ Not needed
  - No overfitting, no outliers. ✓ No higher order polynomials, successful validation

*Non-linear systems of equations

Conclusions

- Loop Replacement with Symbolic Surrogate:
  - Applicable to jet pump model (good performance).
  - Satisfaction of realtime requirements.
  - Applicable to embedded software.
  - Tunable Trade-off between complexity and accuracy.
  - Straight forward integration of symbolic surrogate into model.
  - No special treatment in the further processing (embedded code generation).
  - Training data generated from exact solution at low computation cost.

Outlook

- Validation of the jet pump behavior in system context.
- Evaluate performance on embedded target.
- Application to higher dimensional problems (inverse jet pump {8}).
- Evaluate other symbolic regression packages:
  - E.g., preserve physical dimensions.
  - Try out other base operations.
- Export surrogate as Base Modelica:
  - Automated integration of symbolic surrogate.
  - Convert to other languages, e.g. SYQ.

Thanks for your attention.

Oliver Lenord[1],
Andreas Heuermann[2],
Alexander Fischer[1]
Bernhard Bachmann[2],

1) Robert Bosch GmbH, Germany
2) Hochschule Bielefeld
University of Applied Sciences and Arts, Germany

Funded by the European Union
NextGenerationEU

Supported by:
Federal Ministry for Economic Affairs and Climate Action
on the basis of a decision by the German Bundestag