PASI 2011

Process Modeling and Optimization for Energy and Sustainability

Process Analysis and Dynamic Simulation with EO-CAPE Tools

Argimiro R. Secchi

Chemical Engineering Program – COPPE/UFRJ

Rio de Janeiro, RJ - Brazil

Solutions for Process Control and Optimization

COPPE | UFRJ

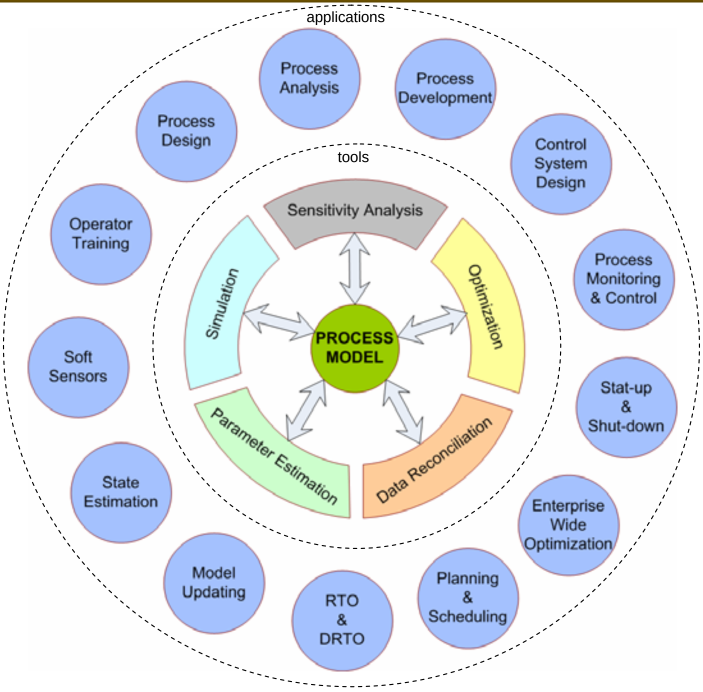
Angra dos Reis

July 23th, 2011

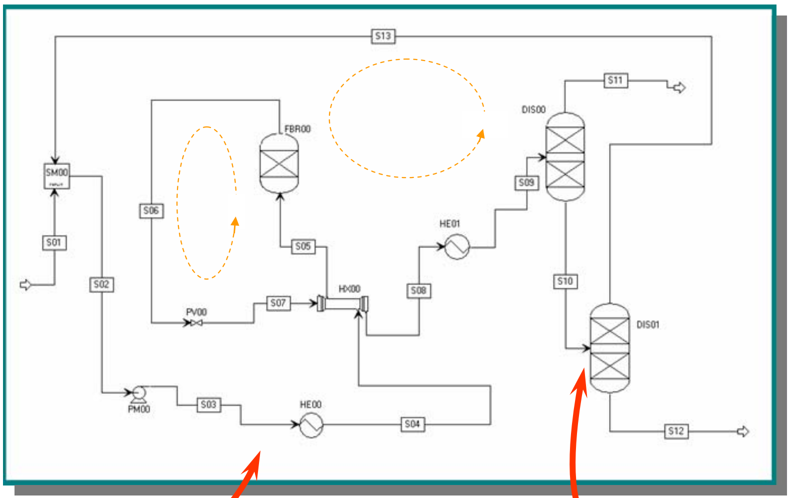
Outline

1. Introduction
2. Object-Oriented Modeling
3. Modeling workshop (introducing EMSO)
4. Dynamic degree of freedom
5. System analysis
6. Debugging techniques

Applications of Process Modeling



Sequential Modular Simulators



Equipments or modules are sequentially evaluated:

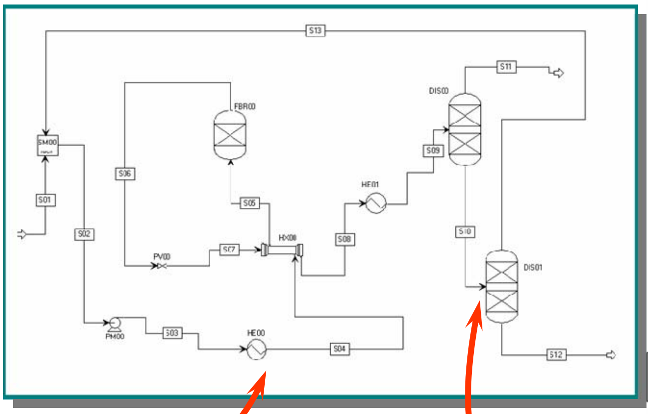
Output result of a block is the input for the next block, with iterative calculation for solving recycle streams.

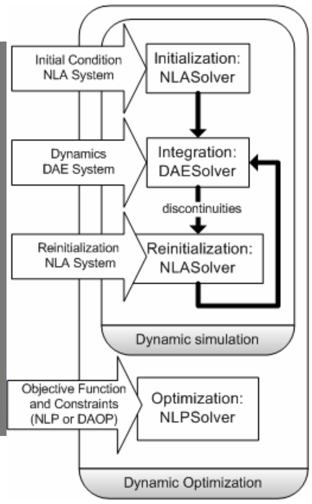
ex: AspenPlus, Hysys, PRO/II, Chemcad, Petrox

Black-box Modeling

Code developed for solving specific equipment (chemistry and physics of the model are mixed with the mathematics)

Equation-Oriented Simulators





Open-source Modeling

Equipments contain only chemistry and physics of the model

ex: EMSO, Ascend, Jacobian, gPROMS, AspenDynamics, EcosimPro

All equipments or modules are simultaneously evaluated
(Block decomposition can be used to explore sequential solution)

CAPE Tools

- A movement from Sequential Modular to Equation-Oriented (EO) tools is clear
- Key advantages of EO:
  - Models can be inspected, refined, or reused
  - Computationally more efficient and easier to diagnose ill-posed problems
  - Same model as the source for several tasks: simulation, optimization, design, parameter estimation, data reconciliation, etc. → integrated environment
- Some disadvantages:
  - More difficult to establish good initial guesses
  - More demand on computer resources

Modeling Tools

The available tools for process modeling may be classified into:

- Block-Oriented

  focus on the flowsheet topology using standardized unit models and streams to link these unit models

- Equation-Oriented

  rely purely on mathematical rather than phenomena-based descriptions, making difficult to customize and reuse existing models

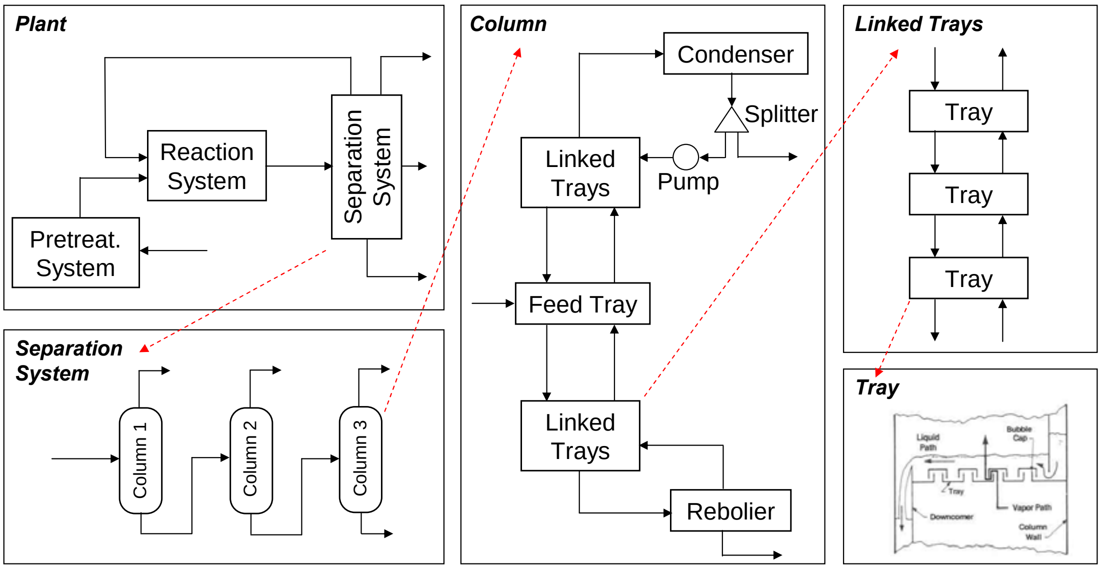
- Object-Oriented

  Models are recursively decomposed into a hierarchy of sub-models and inheritance concepts are used to refine previously defined models into new models

(Bogusch and Marquardt, 1997)

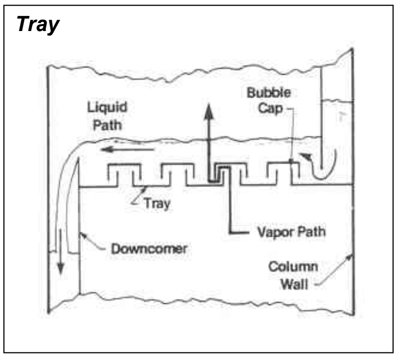
2. Object-Oriented Modeling

A process flowsheet model can be hierarchically decomposed:



8

Object-Oriented Modeling



=

abstract model:
- mass balance
- energy balance
- thermodynamic equilibrium
- mol fraction normalization

+

- liquid flow model
- vapor flow model

concrete model (ideal tray)

+

- efficiency model

concrete model (real tray)

Object-Oriented Modeling

Model types

Abstract models: are models that embody coherent and cohesive, but incomplete concepts, and in turn, make these characteristics available to their specializations via inheritance. While we would never create instances (devices) of abstract models, we most certainly would make their individual characteristics available to more specialized models via inheritance.

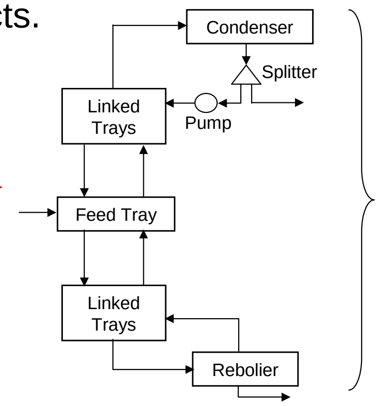
Concrete models: are complete models, usually derived from abstract models, ready to be instantiated, i.e., we can create devices (e.g., equipments) of concrete models.

Object-Oriented Modeling

OOM main concepts

Inheritance: the process whereby one object acquires (gets, receives) characteristics from one or more other objects.

Aggregation: the process of creating a new object from two or more other objects, or an object that is composed of two or more other objects.



Column model = Condenser + Splitter + Pump + Linked Trays + Feed Tray + Reboiler

Object-Oriented Modeling

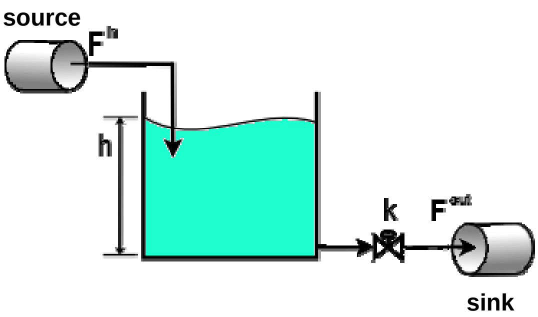
Examples of general-purpose object-oriented modeling languages:

- ABACUSS II (Barton, 1999)
- ASCEND (Piela, 1989)
- Dymola (Elmqvist, 1978)
- EcosimPro (EA Int. & ESA, 1999)
- EMSO (Soares and Secchi, 2003)
- gPROMS/Speedup (Barton and Pantelides, 1994)
- Modelica (Modelica Association, 1996)
- ModKit (Bogusch et al., 2001)
- MPROSIM (Rao et al., 2004)
- Omola (Andersson, 1994)
- ProMoT (Tränkle et al., 1997)

12

Object-Oriented Modeling

A simpler example



Level Tank

<u>Available model of the tank</u>

>>> Model with circular cross section

>>> Model with square cross section

<u>Streams</u>

Inlet	Material stream feeding the tank
Outlet	Material stream leaving the tank

<u>Parameters</u>

k	Valve constant
D	Hydraulic diameter of the tank

<u>Variables</u>

A	Tank cross section area
V	Tank volume
h	Tank level

<u>Devices:</u> source, tank, sink

13

Hydraulic diameter = 4 A / perimeter

Object-Oriented Modeling

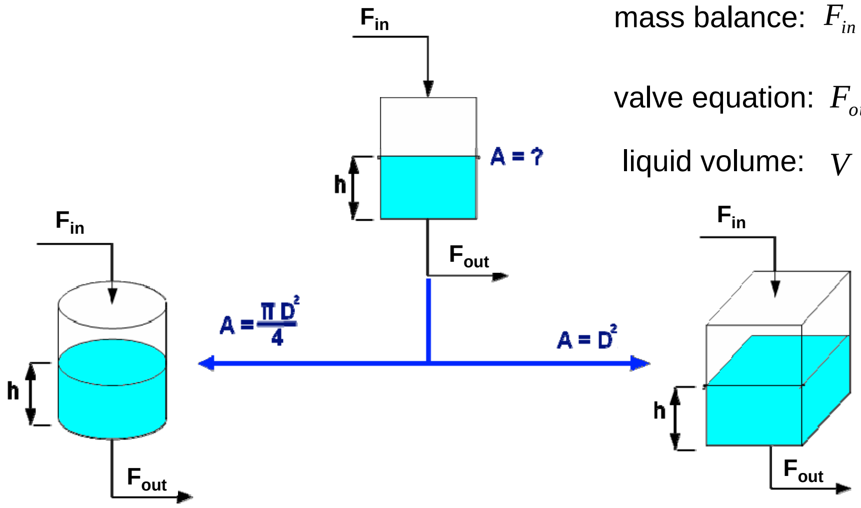
Inheritance

Model equations

mass balance: $F_{in} - F_{out} = \frac{dV}{dt}$

valve equation: $F_{out} = k\sqrt{h}$

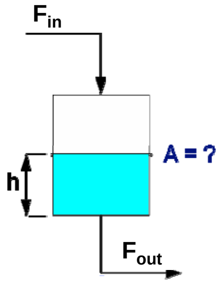
liquid volume: $V = A\,h$



14

Object-Oriented Modeling

Abstract model



mass balance: $F_{in} - F_{out} = \frac{dV}{dt}$

valve equation: $F_{out} = k\sqrt{h}$

liquid volume: $V = A\ h$

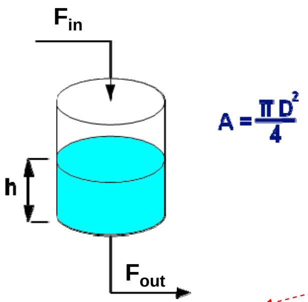
EMSO:

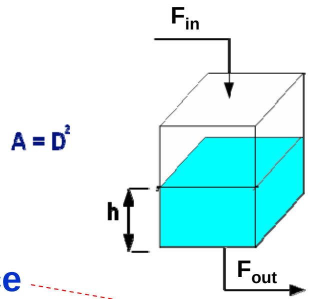
```
using "types";
Model Tank_Basic
PARAMETERS
    k as Real (Brief="Valve constant", Unit='m^2.5/h', Default = 12);
    D as length (Brief="Tank hydraulic diameter", Default = 4);
VARIABLES
in   Fin as flow_vol (Brief="Feed flow rate");
out Fout as flow_vol (Brief ="Output flow rate");
     A as area (Brief="Cross section area");
     V as volume (Brief="Liquid volume");
     h as length (Brief="Tank level");
EQUATIONS
    "Mass balance"  Fin - Fout = diff(V);
    "Valve equation" Fout = k * sqrt(h);
    "Liquid volume"  V = A * h;
end
```

15

Object-Oriented Modeling

Concrete models





Inheritance

```
Model Tank_Circular as Tank_Basic
PARAMETERS
    Pi as Real  (Default = 3.1416);
EQUATIONS
   "Cross section area"   A = (Pi * D^2) / 4;
end
```

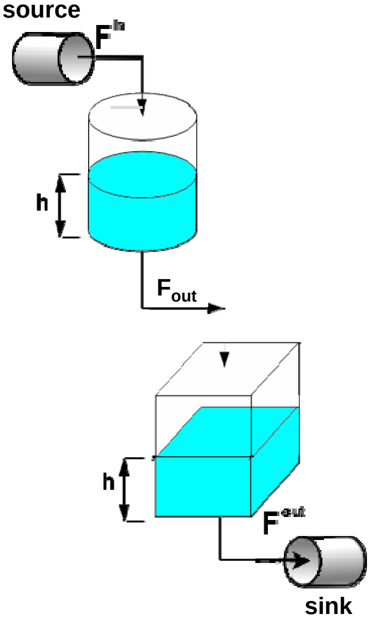
EMSO

```
Model Tank_Square as Tank_Basic
EQUATIONS
   "Cross section area"  A = D^2;
end
```

16

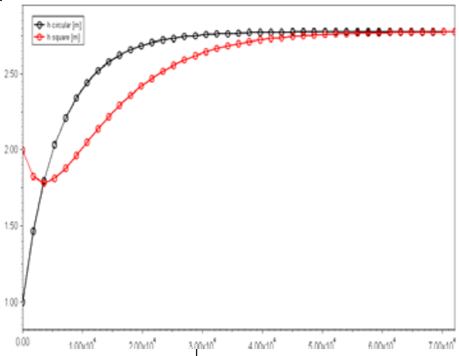
Object-Oriented Modeling

Flowsheet



EMSO:

```
using "tank_oom";
FlowSheet Tanks
DEVICES
   source as Feed;
   T_c as Tank_Circular;
   T_sq as Tank_Square;
   sink as Sink;
CONNECTIONS
    source.F to T_c.Fin;
    T_c.Fout to T_sq.Fin;
    T_sq.Fout to sink.F;
SET
    T_c.D = 3 * 'm';
    T_sq.D = 3 * 'm';
SPECIFY
    source.F = 20 * 'm^3/h';
INITIAL
    T_c.h = 1 * 'm';
    T_sq.h = 2 * 'm';
OPTIONS
    TimeStart = 0;
    TimeEnd = 20;
    TimeStep = 0.5;
    TimeUnit = 'h';
end
```



17

Object-Oriented Modeling

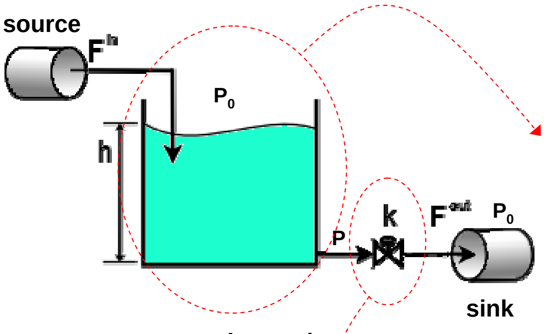
Model switching

```
Model Tank_Section as Tank_Basic
PARAMETERS
    Pi as Real  (Default = 3.1416);
    Section as Switcher (Valid = ["Circular", "Square"],
                         Default = "Circular");
EQUATIONS
  switch Section
    case "Circular":
        "Cross section area"   A = (Pi * D^2)/4;
    case "Square":
        "Cross section area"   A = D^2;
  end
end
```

```
using "tank_oom";
FlowSheet Tanks2
DEVICES
   source as Feed;
   T_c as Tank_Section;
   T_sq as Tank_Section;
   sink as Sink;
CONNECTIONS
    source.F to T_c.Fin;
    T_c.Fout to T_sq.Fin;
    T_sq.Fout to sink.F;
SET
    T_c.D = 3 * 'm';
    T_sq.D = 3 * 'm';
    T_c.Section = "Circular";
    T_sq.Section = "Square";
SPECIFY
    source.F = 20 * 'm^3/h';
INITIAL
    T_c.h = 1 * 'm';
    T_sq.h = 2 * 'm';
OPTIONS
    TimeStart = 0;
    TimeEnd = 20;
    TimeStep = 0.5;
    TimeUnit = 'h';
end
```

Object-Oriented Modeling

Aggregation



Level Tank

Tank model

mass balance: $F_{in} - F_{out} = \frac{dV}{dt}$

liquid volume: $V = A\,h$

outlet pressure: $P = P_0 + \rho\, g\, h$

Valve model

mass balance: $F_{in} = F_{out}$

valve equation: $F_{out} = k\sqrt{\frac{\Delta P}{\rho\, g}}$

pressure drop: $\Delta P = P_{in} - P_{out}$

Object-Oriented Modeling

Tank model with valve

```
using "types";
Model Tank_Basic
PARAMETERS
    D as length (Brief="Tank hydraulic diameter", Default = 4);
    rg as Real (Brief="rho * g", Unit ='kg/(m*s)^2', Default = 1e4);
VARIABLES
in   Sin as stream (Brief="Inlet stream");
out Sout as stream (Brief ="Outlet stream");
     A as area (Brief="Cross section area");
     V as volume (Brief="Liquid volume");
     h as length (Brief="Tank level");
     valve as Valve (Brief="Valve model");
CONNECTIONS
    Sout to valve.Sin;
EQUATIONS
    "Mass balance"  Sin.F – Sout.F = diff(V);
    "Liquid volume"  V = A * h;
    "Outlet pressure"  Sout.P = Sin.P + rg * h;
end
```

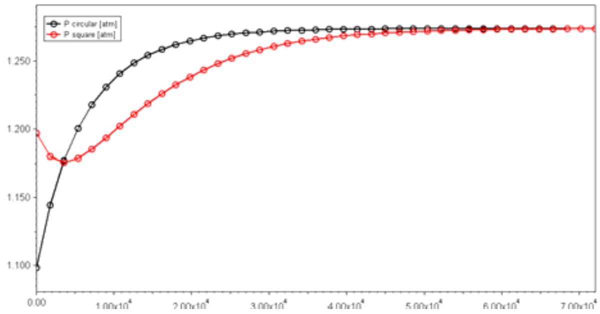
Valve model

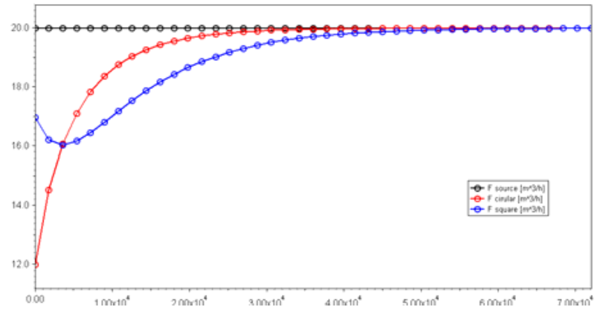
```
using "types";
Model Valve
PARAMETERS
    k as Real (Brief="Valve constant",
            Unit='m^2.5/h', Default = 12);
    rg as Real (Brief="rho * g",
            Unit ='kg/(m*s)^2', Default = 1e4);
VARIABLES
in   Sin as stream (Brief="Inlet stream");
out Sout as stream (Brief ="Outlet stream");
     DP as press_delta (Brief="Pressure drop");
EQUATIONS
    "Mass balance"  Sin.F = Sout.F;
    "Valve equation" Sout.F = k * sqrt(DP/rg);
    "Pressure drop"  DP = Sin.P – Sout.P;
end
```

Object-Oriented Modeling

Flowsheet

```
using "tank_valve_oom";
FlowSheet Tanks
DEVICES
   source as Feed;
   T_c as Tank_Circular;
   T_sq as Tank_Square;
   sink as Sink;
CONNECTIONS
    source.Sout to T_c.Sin;
    T_c.valve.Sout to T_sq.Sin;
    T_sq.valve.Sout to sink.Sin;
SET
    T_c.D = 3 * 'm';     T_sq.D = 3 * 'm';
SPECIFY
    source.Sout.F = 20 * 'm^3/h';
    source.Sout.P = 1 * 'atm';
    T_c.Sout.P = 1 * 'atm';
    sink.Sin.P = 1 * 'atm';
INITIAL
    T_c.h = 1 * 'm';     T_sq.h = 2 * 'm';
OPTIONS
    TimeStart = 0;
    TimeEnd = 20;
    TimeStep = 0.5;
    TimeUnit = 'h';
end
```





Object-Oriented Modeling

How to apply for Energy and Sustainability?

Creating specialized models by reusing the available library of models via inheritance and incorporating necessaries characteristics for the new application.

Ex: Using the environmental impact factors of the different materials

Categories
HTPI (Human toxicity potential by ingestion)
HTPE (Human toxicity potential by exposure)
ATP (Aquatic toxicity potential)
TTP (Terrestrial toxicity potential)
GWP (Global warming potential)
ODP (Ozone depletion potential)
PCOP (Photo chemical oxidation potential)
ARP (Acid rain potential)
EP (Eutrophication potential)

22

Object-Oriented Modeling

$$\Psi_{k,j} = F_k \gamma_{k,j}$$ environment impact of component *k in category j*

$$\Psi_j = \sum_k \Psi_{k,j}$$ environment impact of *category j*

$$\Psi = \sum_j \omega_j \Psi_j$$ total potential environmental impact

$$\%\Psi_j = 100 \frac{\omega_j \Psi_j}{\Psi}$$ percentage contribution of category *j*

F_k flow rate of component *k*

$\gamma_{k,j}$ *impact score of component k in category j*

ω_j *weighting factor for each category j*

(Heijungs et al., 1992; Eliceche et al., 2007)

23

Creating a new stream with environment impact characterization

Existing models

```
Model stream
PARAMETERS
outer NComp as Integer (Brief="Number of chemical components", Lower = 1);
VARIABLES
    F as flow_mol (Brief="Stream Molar Flow Rate");
    T as temperature (Brief ="Stream Temperature");
    P as pressure (Brief="Stream Pressure");
    h as enth_mol (Brief="Stream Enthalpy");
    v as fraction (Brief="Vaporization fraction");
    z(NComp) as fraction (Brief="Stream Molar Fraction");
end
```

```
Model simple_sink
VARIABLES
in   Inlet as stream (Brief="Inlet Stream");
end
```

Object-Oriented Modeling

New stream model

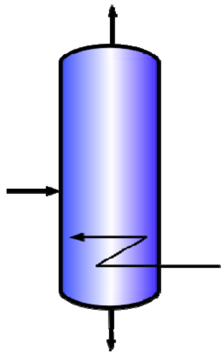
```
Model sink_impact as simple_sink
PARAMETERS
outer PP as Plugin (Brief="External Physical Properties", Type="PP");
outer NComp as Integer (Brief="Number of chemical components", Lower = 1);
        Nfactor as Integer (Brief="Number of categories", Lower=1, Default=9);
        w(Nfactor) as fraction (Brief="Weighting factor");
VARIABLES
     psiX(NComp,Nfactor) as frequency (Brief="Component environment impact");
     psiC(Nfactor) as frequency (Brief="Category environment impact");
     psiCp(Nfactor) as percent (Brief="Category percentage contribution");
     psi as frequency (Brief="Total environment impact");
     Fw(NComp) as flow_mass (Brief="Component Mass Flow Rate");
EQUATIONS
    Fw = Inlet.F * Inlet.z * PP.MolecularWeight();
    for i in [1:NComp]
        psiX(i,:) = Fw(i) * PP.ImpactFactor(i,Nfactor);
    end
    psiC = sum(psiX);
    psi = sum(psiC * w);
    psiCp = 100*w*psiC/psi;
end
```

Object-Oriented Modeling

A simple example

file ex_impact.mso

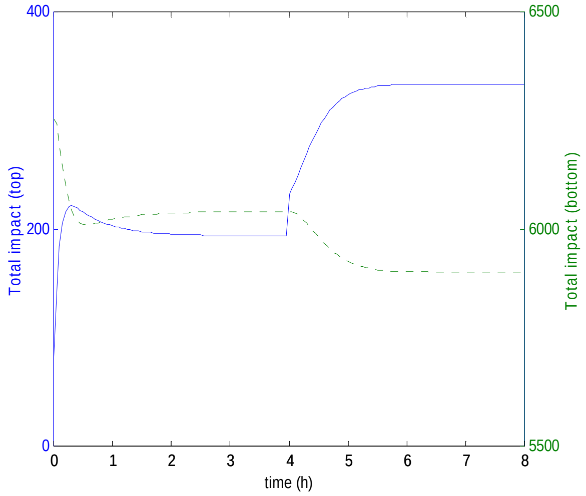
```
using "streams_impact";
using "stage_separators/flash";
FlowSheet flash_impact
PARAMETERS
   PP as Plugin (Brief="Physical Properties", Type="PP",
              Components = ["hydrogen", "methane", "benzene", "toluene", "biphenyl", "water"],
              LiquidModel = "PR",    VapourModel = "PR");
   NComp as Integer;
VARIABLES
   Q  as energy_source (Brief="Heat supplied");
SET
   NComp = PP.NumberOfComponents;
DEVICES
   fl as flash;
   s1 as source;
   top as sink_impact;
   bot as sink_impact;
CONNECTIONS
   s1.Outlet to fl.Inlet;
   Q.OutletQ to fl.InletQ;
   fl.OutletV to top.Inlet;
   fl.OutletL to bot.Inlet;
 ...
```



26

Object-Oriented Modeling

Response of the potential environment impact for a +20% step change in the feed flow rate at time = 4h.

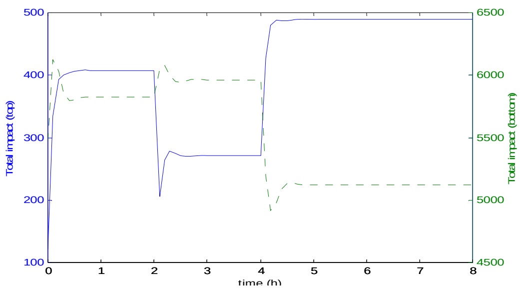


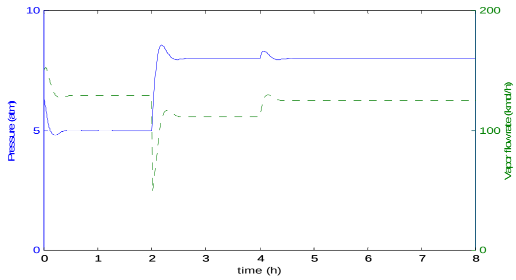
Category contribution for the potential environment impact at the final time

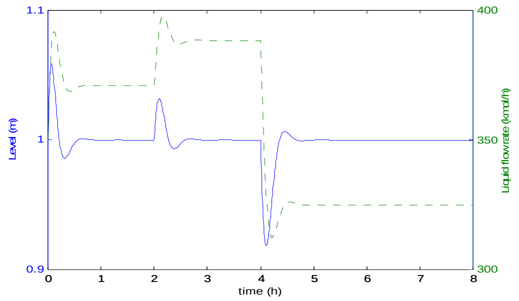
Impact category	% Contribution (top)	% Contribution (bottom)
Human toxicity by ingestion	7.58	6.57
Human toxicity by exposure	0.58	5.23
Aquatic toxicity	7.58	6.57
Terrestrial toxicity	2.71	7.91
Global warming	0.34	0.1×10^{-3}
Ozone depletion	0.00	0.00
Photo chemical oxidation	81.21	73.72
Acid rain	0.00	0.00
Eutrophication	0.00	0.00

Object-Oriented Modeling

Results with pressure and level controllers







Disturbances:

Pressure set-point at time = 2h

5 atm → 8 atm

Temperature and feed flow rate at time = 4h

340 K → 360 K

500 kmol/h → 450 kmol/h

3. Modeling Workshop

Introducing EMSO

- EMSO stands for "**Environment for Modeling, Simulation, and Optimization**"
- Development started in 2001 (by Rafael P. Soares), written in C++ language
- Available in Windows and Linux
- Models are written in an object-oriented modeling language
- Equation-oriented simulator and optimizer
- Computationally efficient for dynamic and steady-state simulations
- Continuous improvements through ALSOC project:

http://www.enq.ufrgs.br/alsoc

ALSOC PROJECT - EMSO - Trac - Mozilla Firefox
http://www.enq.ufrgs.br/trac/alsoc
HOME | Project Index | Stats | Contributions | Users | Support | Contact | Help/Guide | About Trac | Login | Settings | Register
Wiki | Timeline | Roadmap | Browse Source | View Tickets | Search | FAQ | Downloads
Start Page | Index by Title | Index by Date | Last Change
Welcome to the ALSOC Project homepage
ALSOC is the acronym used to identify the project of a free environment for simulation, optimization, and process control.
ALSOC é a sigla utilizada para identificar o projeto de um Ambiente Livre para Simulação, Otimização e Controle de Processos.
The ALSOC Project is an effort to bring together university-industry through the standardization and distribution without cost of specifications and software tools among universities and partner companies.
O Projeto ALSOC é um esforço de aproximação universidade-indústria através da padronização e distribuição sem custo de especificações e ferramentas de software entre universidades e empresas consorciadas.
Look here the list of institutions that participate and sponsor the project.
Veja aqui a lista de instituições que participam e patrocinam o projeto.
Project Goals
The main goals of the ALSOC Project are:
As principais finalidades do Projeto ALSOC são:
• to develop, maintain, and distribute specifications of a modeling language and a library of models for the synthesis, simulation, optimization and control of general processes (check the ALSOC OPEN LICENSE);
desenvolver, manter e distribuir especificações de uma linguagem de modelagem e uma biblioteca de modelos abertas para síntese, simulação, otimização e controle de processos em geral (veja a licença aberta ALSOC);
• to develop and maintain state-of-the-art software and to distribute it at no cost to the universities and partner companies (check the ALSOC LICENSE);
desenvolver e manter software no estado-da-arte distribuído sem custo entre os consorciados e entidades educacionais (veja a licença ALSOC);
• to certify third party solution and models as conforming to the developed standards.
certificar a conformidade de soluções externas com os padrões desenvolvidos e adicionar ao Projeto contribuições externas.
EMSO Process Simulator
EMSO is the acronym for Environment for Modeling Simuation and Optimization.
EMSO é a sigla para Environment for Modeling Simuation and Optimization.
EMSO is the simulation software of the ALSOC project. Its development was started at 2001 by Rafael de Pelegrini Soares, today the EMSO process simulator is developed and maintained by ALSOC.
O EMSO é o software de simulação do projeto ALSOC. Sua construção foi iniciada em 2001 por Rafael de Pelegrini Soares, hoje o simulador EMSO é desenvolvido e mantido pelo projeto ALSOC.
Learn more about EMSO, check the ChangeLog, or download it here!
Saiba mais sobre o EMSO, veja o ChangeLog ou faça o seu download aqui!
News!
• mar 07 2008: EMSO version 0.9.55 released! Download it here
• dez 25 2007: EMSO version 0.9.54 released! Download it here
• aug 31 2007: EMSO version 0.9.53 released! Download it here
• aug 27 2007: Curso do simulador EMSO. Clique AQUI para fazer o download do material do curso.
• jun 15 2007: A nice Quick Reference is now available.
• feb 14 2007: Displaying a formula Click here.
• dec 18 2006: ALSOC meeting: sponsors and developers discussing about the future of the project and recent advances. Read more...
• nov 1 2006: Get involved!: Contribute your models to the EMSO Model Library. Check the Contribution Page!
DEQUI - Departamento de ENGENHARIA QUÍMICA
GIMSCOP
Concluído

EMSO Key Features

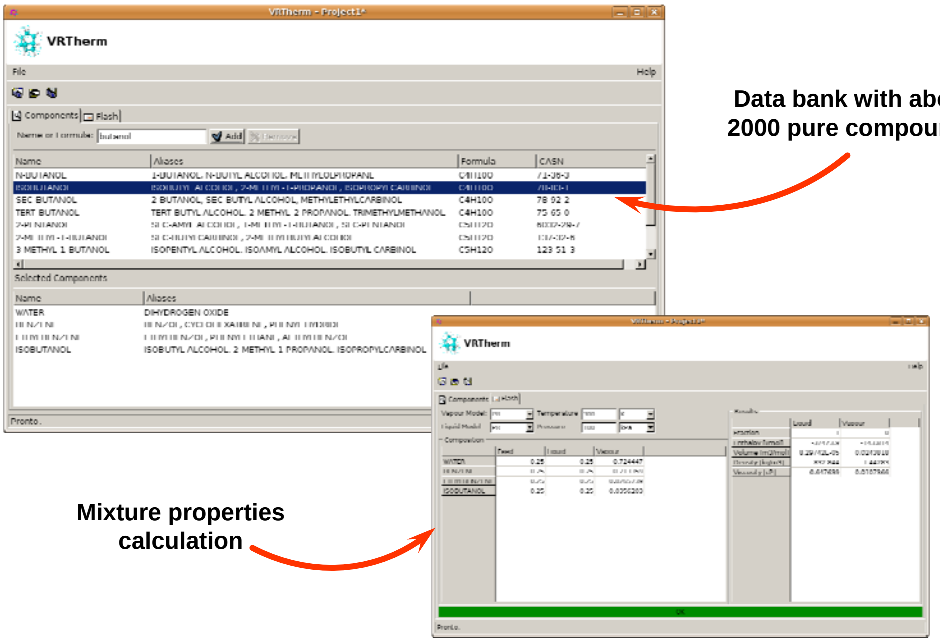
- ✓ Open source library of models
- ✓ Object-oriented modeling
- ✓ Built-in automatic and symbolic differentiation
- ✓ Automatic checking and conversion of units of measurement
- ✓ Solve high-index problem
- ✓ Perform consistency analysis (DoF, DDoF, initial condition)
- ✓ Integrated Graphical User Interface (GUI)
- ✓ Building blocks to create flowsheets
- ✓ Discrete (state and time) event handling
- ✓ Multitask for concurrent and real-time simulations
- ✓ Modular architecture and support to sparse algebra
- ✓ Multiplatform: win32 and posix
- ✓ Interface with user code written in C/C++ or Fortran
- ✓ Automatic documentation of models using hypertexts and LaTeX

What can I do with EMSO?

- ☑ Steady-state simulations
- ☑ Dynamic simulations
- ☑ Steady-state optimizations (NLP, MINLP)
- ☑ Steady-state parameter estimations
- ☑ Dynamic parameter estimations
- ☑ Steady-state data reconciliations
- ☑ Process follow-up and inferences with OPC communication
- ☑ Build bifurcation diagrams (interface with AUTO for DAEs)
- ☑ Dynamic simulations with SIMULINK (interface with MATLAB)
- ☑ Add new solvers (DAE, NLA, NLP)
- ☑ Add external routines using the Plugins resource

32

Thermodynamic and Physical Properties – Plugin



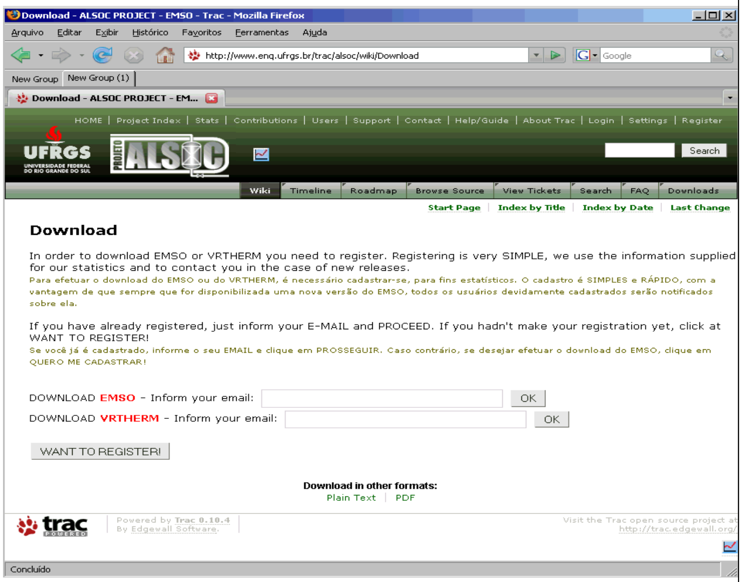
Data bank with about 2000 pure compounds

Mixture properties calculation

33

How can I install EMSO?

- Download EMSO and VRTherm packages from http://www.enq.ufrgs.br/alsoc
- Run the setup programs
- Run EMSO
- Add the physical properties package using the Config Plugins option in the menu
- Select an example and run it

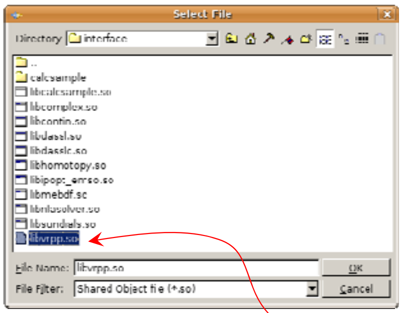


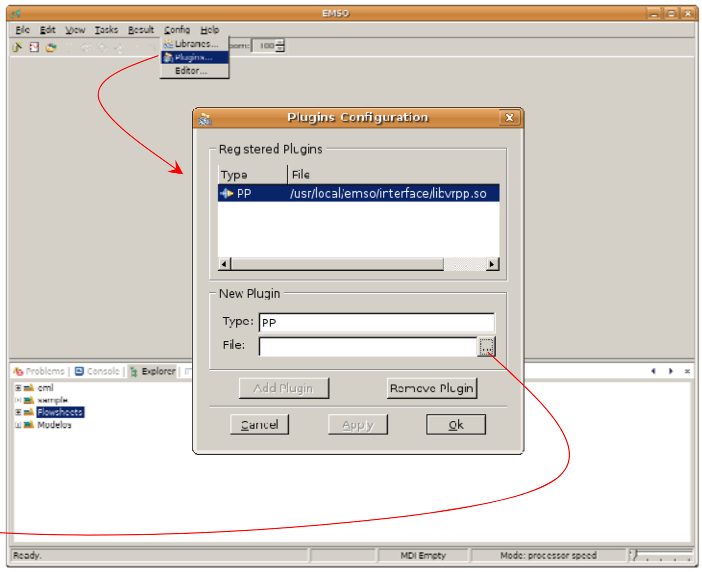
34

Configuring Plugin
– VRTherm package: vrpp –

To use a plug-in the user needs to register it through the menu

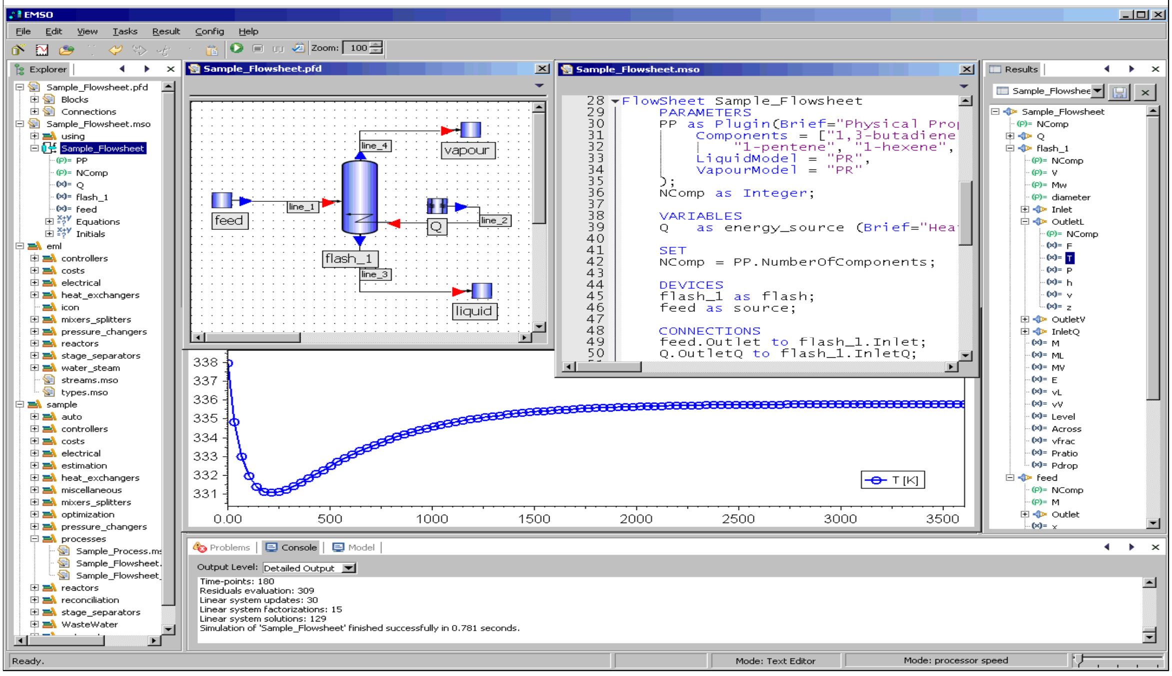
Config ⟶ *Plugins*





Windows plug-in is a DLL file, and Linux plug-in is a SO file

Integrated GUI
– Running an example –

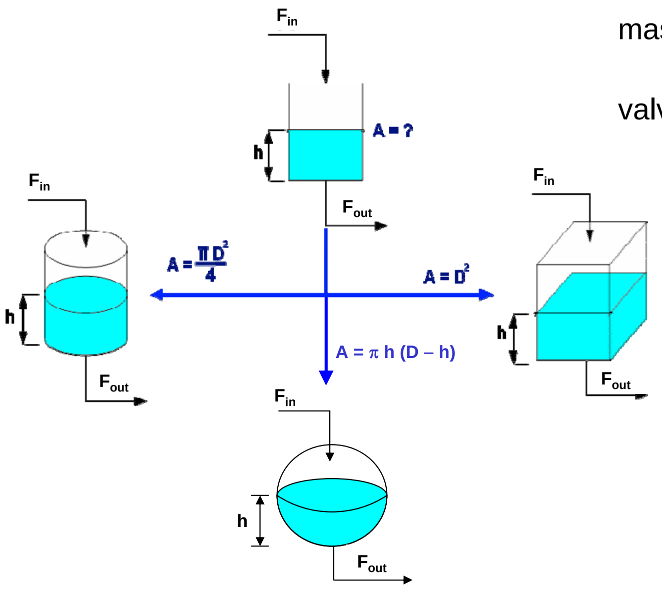


Modeling workshop

Model equations

mass balance: $F_{in} - F_{out} = \dfrac{dV}{dt}$

valve equation: $F_{out} = k\sqrt{h}$



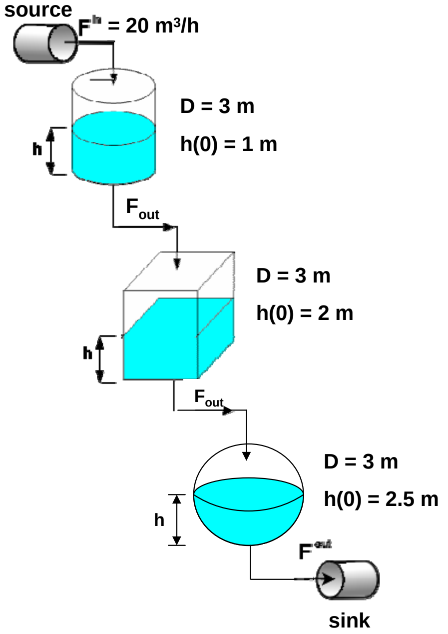
liquid volume:

$$V = A\,h$$

$$V = \pi h^2 \left(\frac{D}{2} - \frac{h}{3} \right)$$

Modeling workshop

Flowsheet



38

Remarks about Modeling

Basic Elements in Modeling

	1. Process description and problem definition
Model definition	2. Fundamental laws: theory and application
	3. Simplifying assumptions
Model building	4. Mathematical model
	5. Consistency analysis
Model validation	6. Desired solution
	7. Computation
	8. Solution and validation

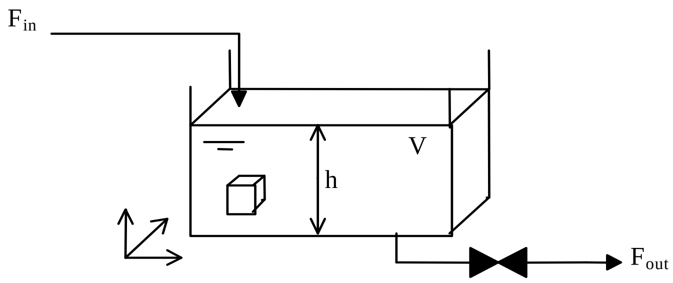
39

1. Process Description and Problem Definition

- Process Description
  - Process objectives
  - Process flowsheet
  - Process operation
    - unit operations and control
- Problem Definition
  - Simulation objectives
  - Simulation applications

Process Description

Example: level tank



A liquid flows in and out of a tank due to gravitational forces. We wish to analyze the volume, height and flowrate variations in the tank (system response) as function of feed disturbances.

41

- Bases to be used in the modeling

  - mass conservation

$$\frac{\partial \rho}{\partial t} = -(\nabla . \rho v)$$

  - momentum conservation

$$\frac{\partial(\rho v)}{\partial t} = \underbrace{-[\nabla . \rho v v]}_{\text{advection}} \quad \underbrace{-\nabla P}_{\text{pressure forces}} \quad \underbrace{-[\nabla . \tau]}_{\text{viscous forces}} \quad \underbrace{+\rho g}_{\text{gravitational forces}}$$

  - energy conservation

$$\frac{\partial}{\partial t}\left[\rho\left(\hat{U} + \frac{1}{2}v^2\right)\right] = \underbrace{-\left(\nabla . \rho v\left(\hat{U} + \frac{1}{2}v^2\right)\right)}_{\text{advection}} \quad \underbrace{-(\nabla . q)}_{\text{conduction}} \quad \underbrace{+\rho\left(g . v\right)}_{\text{gravit. forces work}} \quad \underbrace{-(\nabla . P v)}_{\text{pressure forces work}} \quad \underbrace{-(\nabla . [\tau . v])}_{\text{viscous forces work}}$$

42

3. Simplifying Assumptions

- Establish the assumptions and simplifications
- Define the model limitations

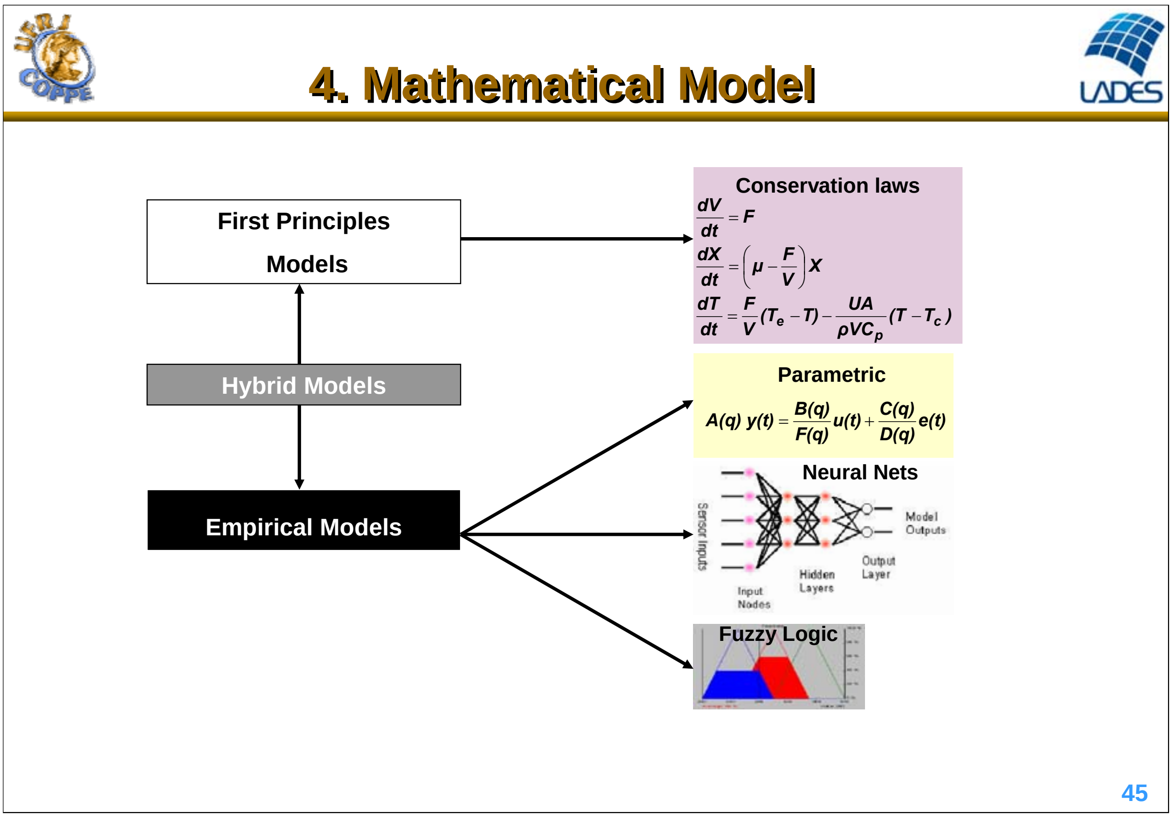
- constant specific mass
- isothermal
- perfect mixture
- $F_{out} = k\sqrt{h}$

43

4. Mathematical Model

- Data mining for simulation
  - Collect data and information of the studied system
  - Identify the engineering unit of measurements
  - Specify operating procedures
  - Specify the operating regions of the variables

- Memory of Calculation
  - Mathematical model
  - Define unit of measurements of variables and parameters
  - Define and specify free variables
  - Define and determine values of parameters
  - Define and establish initial conditions


4. Mathematical Model
First Principles Models
Hybrid Models
Empirical Models
Conservation laws
$\frac{dV}{dt} = F$
$\frac{dX}{dt} = \left(\mu - \frac{F}{V}\right)X$
$\frac{dT}{dt} = \frac{F}{V}(T_e - T) - \frac{UA}{\rho V C_p}(T - T_c)$
Parametric
$A(q)\,y(t) = \frac{B(q)}{F(q)}u(t) + \frac{C(q)}{D(q)}e(t)$
Neural Nets
Sensor Inputs
Input Nodes
Hidden Layers
Output Layer
Model Outputs
Fuzzy Logic
45

Mathematical Model

In the simulator

- Build process equipment models
  - Identify and create abstract and concrete models
  - Declare variables and parameters
  - Write model equations
  - Compose the equipment model via inheritance and aggregation

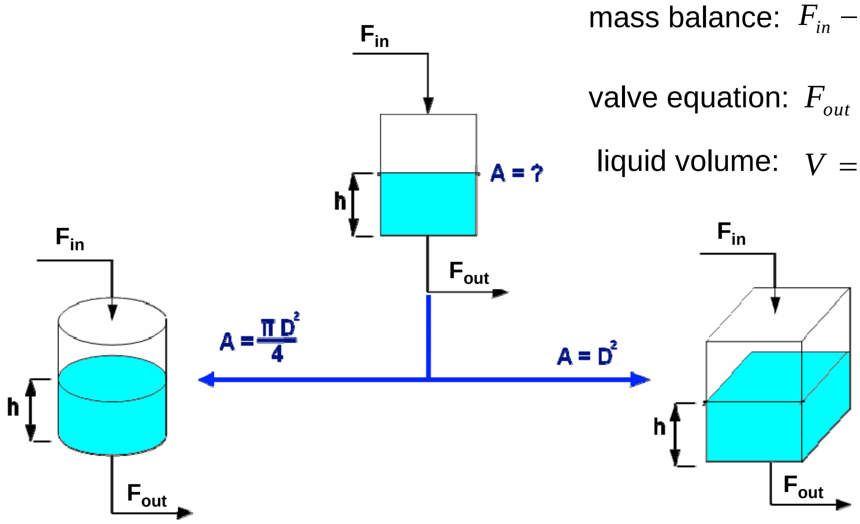
- Build process flowsheet
  - Declare flowsheet devices
  - Define process connections
  - Set process parameters values
  - Specify process free variables
  - Establish initial conditions
  - Establish simulation options

Mathematical Model

mass balance: $F_{in} - F_{out} = \frac{dV}{dt}$ (1)

valve equation: $F_{out} = k\sqrt{h}$ (2)

liquid volume: $V = A\,h$ (3)



(4)

5. Consistency Analysis

- Model consistency analysis for unit of measurements (UOM)
- Degree of freedom analysis
- Dynamic degree of freedom analysis

variable	UOM	equations
F_{in}, F_{out}	$m^3\ h^{-1}$	(1): $[m^3\ h^{-1}] - [m^3\ h^{-1}] = [m^3] / [h]$
V	m^3	(2): $[m^3\ h^{-1}] = [m^{2.5}\ h^{-1}]\ ([h])^{0.5}$
A	m^2	(3): $[m^3] = [m^2]\ [m]$
h, D	m	(4): $[m^2] = ([m])^2$
k	$m^{2.5}\ h^{-1}$	
t	h	

Consistency Analysis

variables: F_{in}, F_{out}, V, A, h, D, k, t ➔ 8
constants: k, D ➔ 2
specifications: t ➔ 1
driving forces: F_{in} ➔ 1
unknown variables: V, h, A, F_{out} ➔ 4
equations: 4

Degree of Freedom = variables – constants – specification – driving forces – equations = unknown variables – equations = 8 – 2 – 1 – 1 – 4 = 0

Dynamic Degree of freedom (index < 2) = differential equations = 1

Needs 1 initial condition: h(0) ➔ 1

49

6. Desired Solution

- Plan case studies
- Define:
  - Objectives of the study
  - Problems to be solved
  - Evaluation criteria

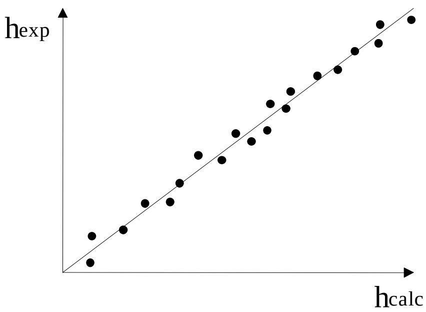
For the given example and initial condition (h_0 or V_0), we wish to analyze h(F_{in}), V(F_{in}) and F_{out}(F_{in}).

7. Computation

- Define the desired accuracy
- Specify the simulation time and reporting interval
- Verify the necessity of specialized solvers (high-index problems)

8. Solution and Validation

- Analyze simulation results
- Analyze state variables dynamics
- Test model fitting with plant data
  - Compare simulation x plant



52

Solution and Validation

- Check output sensitivity to input disturbances
- Carry out parametric sensitivity analysis
- Analyze output data with statistical techniques
- Verify results coherence
- Document obtained results

Comments

• Start with a simple model and gradually increase complexity when necessary;

• The model should have sufficient details to capture the essence of the studied system;

• It is not necessary to reproduce each element of the system;

• Models with excessive details are expensive, difficult to implement and to solve;

• Interact with people that operate the equipment;

• Deeply understand the process behavior.

54

4. Dynamic Degree of Freedom

✓ **In Equation-Oriented (EO) simulators a model has:**

- A set of model parameters (reaction order, valve constant, etc.)
- A set of variables (temperatures, pressures, flow rates, etc.)
- A set of equations (algebraic and differential) relating the variables

✓ **Problems in model building:**

- Number of equations and variables do not match
- Equations of the model are inconsistent (linear dependence, etc.)
- The number of initial conditions and DDoF do not match

Dynamic Degree of Freedom
– Consistency Analysis –

Degree of Freedom (DoF)

- = 0 (for simulation)
- > 0 (for optimization)

Dynamic Degree of Freedom (DDoF)

→ = number of given initial conditions

Check → Units of measurement
→ Structural non-singularity
→ Consistent initial conditions

Dynamic Degree of Freedom
– General Concept –

Given a system of DAE: $F(t, y, y') = 0$

The Dynamic Degree of Freedom (DDoF) is the number of variables in $y(t_0)$ that can be assigned arbitrarily to compute a set of consistent initial conditions $\{y(t_0), y'(t_0)\}$ of the DAE system. Is the true number of states of the system (or the system order of the DAE). Is the number of initial conditions that must be given.

For low-index DAE system (index 0 and 1) the DDoF is equal to the number of differential equations.

For high-index DAE system (index > 1) the DDoF is equal to the number of differential variables minus the number of hidden constraints.

Dynamic Degree of Freedom
– DAE System Characterization –

Differential index (ν): Is the minimum number of times the DAE system $F(t,x,x',u) = 0$ needs to be differentiated with respect to t to be transformed in an explicit ODE system in terms of x'.

Example:

$$x_1' - x_2 = 0 \qquad\qquad x_1' = x_2$$
$$x_1 = u(t) \xrightarrow[\text{twice in } t]{\text{differentiating}} x_2' = u''(t)$$

$\nu = 2$

If the resolution of a DAE system presents difficulties for initializing and/or presents error propagation in the numerical integration, then this system has an ***index problem***, this problem may occur in DAE systems with $\nu > 1$.

58

Dynamic Degree of Freedom
– Consistent Initial Conditions –

Substituting the first differentiation: $x_1' = u'(t)$

in the first equation, results in: $x_2 = u'(t)$

In the initial time:
$$x_1(0) = u(0)$$
$$x_2(0) = u'(0)$$
$$x_1'(0) = u'(0)$$
$$x_2'(0) = u''(0)$$

Therefore, there is no dynamic degree of freedom, i.e., the system did not accept any arbitrary initial conditions.

Dynamic Degree of Freedom

– Consistent Initial Conditions –

The most difficult step for solving DAE systems is the determination of consistent initial conditions.

During the differentiation process to reduce the index of a DAE system to zero, hidden constraints may arise. The original system augmented by the set of the hidden equations is called ***extended system***.

Extended system of the example:

$$x_1' - x_2 = 0 \qquad x_2 = u'(t)$$

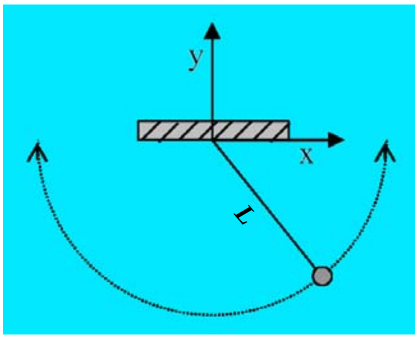
$$x_1 = u(t) \qquad x_2' = u''(t)$$

Consistent initial conditions: The vectors $x(t_0)$ and $x'(t_0)$ form a consistent initial condition of the DAE system $F(t,x,x',u) = 0$ at t_0 if they satisfy the extended system at t_0.

60

Dynamic Degree of Freedom
– High-Index DAE System –

Example: classical pendulum problem



Inconsistent initial condition:

Equation		Initial values	
$x' = w$	(1)	$x(0) = 1$	$x'(0) = 0$
$y' = z$	(2)	$y(0) = 0$	$y'(0) = 0$
$w' = T.x$	(3)	$w(0) = 0$	$w'(0) = 1$
$z' = T.y - g$	(4)	$z(0) = 0$	$z'(0) = -g$
$x^2 + y^2 = L^2$	(5)	$T(0) = 1$	$T'(0) = 0$

OK!

$$F(t, y, y') = 0 \qquad F(0, y(0), y'(0)) = 0$$

NOT OK!

Hidden constraints:

Differentiating (5) and using (1) and (2): $x \cdot w + y \cdot z = 0$ (6) $\longrightarrow$ $x(0) \cdot w(0) + y(0) \cdot z(0) = 0$

Differentiating (6) and using (1)–(5): $w^2 + z^2 + T \cdot L^2 = g \cdot y$ (7) $\longrightarrow$ $w(0)^2 + z(0)^2 + T(0) \cdot L^2 \neq g \cdot y(0)$

Differentiating (7) and using (2), (3), (4), (6): $T' = -3\,g \cdot z / L^2$ (8) $\longrightarrow$ $T'(0) = -3\,g \cdot z(0) / L^2$

61

Dynamic Degree of Freedom
– High-Index DAE System –

Example: classical pendulum problem

$x' = w$ (1)

$y' = z$ (2)

$w' = T.x$ (3)

$z' = T.y - g$ (4)

$x^2 + y^2 = L^2$ (5)

$x \cdot w + y \cdot z = 0$ (6)

$w^2 + z^2 + T \cdot L^2 = g \cdot y$ (7)

$T' = -3g \cdot z / L^2$ (8)

↓

10 variables (y, y´)

↓

8 equations

↓

2 DDoF

But not any pair!

Index 3

$x' = w$ (1)

$y' = z$ (2)

$w' = T.x$ (3)

$z' = T.y - g$ (4)

$x^2 + y^2 = L^2$ (5)

Index 2

$x' = w$ (1)

$y' = z$ (2)

$w' = T.x$ (3)

$z' = T.y - g$ (4)

$x \cdot w + y \cdot z = 0$ (6)

Index 1

$x' = w$ (1)

$y' = z$ (2)

$w' = T.x$ (3)

$z' = T.y - g$ (4)

$w^2 + z^2 + T \cdot L^2 = g \cdot y$ (7)

Index 0

$x' = w$ (1)

$y' = z$ (2)

$w' = T.x$ (3)

$z' = T.y - g$ (4)

$T' = -3g \cdot z / L^2$ (8)

Satisfies the inconsistent I.C.

62

Dynamic Degree of Freedom
– High-Index DAE: solution –

Three general approaches:

1) Manually modify the model to obtain a lower index equivalent model

2) Integration by specifically designed high-index solvers (e.g., PSIDE, MEBDF, DASSLC)

EMSO: Integration = “original”

3) Apply automatic index reduction algorithms

EMSO: Integration = “index0”

or EMSO: Integration = “index1”

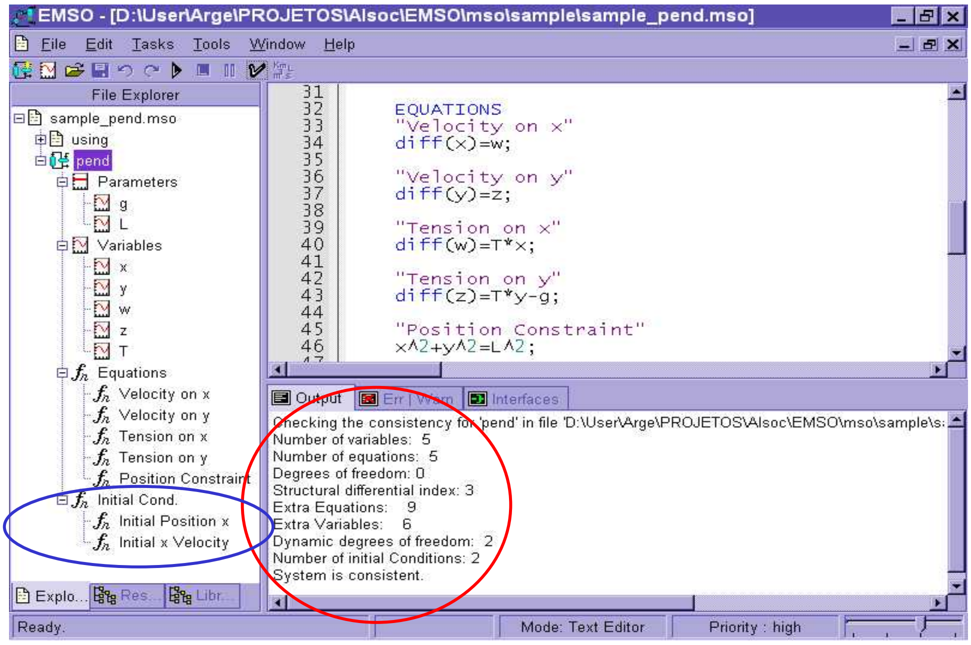
63

Dynamic Degree of Freedom
– High-Index DAE: modeling –

```
using "types.mso";

FlowSheet pend
    PARAMETERS
    g       as acceleration (Brief="Gravity acceleration");
    L       as length (Brief="Pendulum cable length");

    VARIABLES
    x    as length_delta (Brief="Position x");
    y    as length_delta (Brief="Position y");
    w    as velocity (Brief="Velocity for x");
    z    as velocity (Brief="Velocity for y");
    T    as Real  (Brief="Tension on cable",
                   Default=10,Unit="1/s^2");

    EQUATIONS
    "Velocity on x"
    diff(x)=w;

    "Velocity on y"
    diff(y)=z;

    "Tension on x"
    diff(w)=T*x;

    "Tension on y"
    diff(z)=T*y-g;

    "Position Constraint"
    x^2+y^2=L^2;

    SET
    g      = 9.8 * 'm/s^2';
    L      = 0.9 * 'm';

    INITIAL
    "Initial Position x"
    x = 0.9 * 'm';

    "Initial x Velocity"
    w = 0 * 'm/s';

    OPTIONS
    TimeStep = 0.1;
    TimeEnd = 36;
    Integration = "index0";
#   Integration = "original";

    NLASolver(
        RelativeAccuracy = 1e-8,
        AbsoluteAccuracy = 1e-9
    );
    DAESolver(
        File = "dassl",
        #File = "mebdf",
        RelativeAccuracy = 1e-6,
        AbsoluteAccuracy = 1e-8
    );
    SparseAlgebra = true;
end
```

Dynamic Degree of Freedom

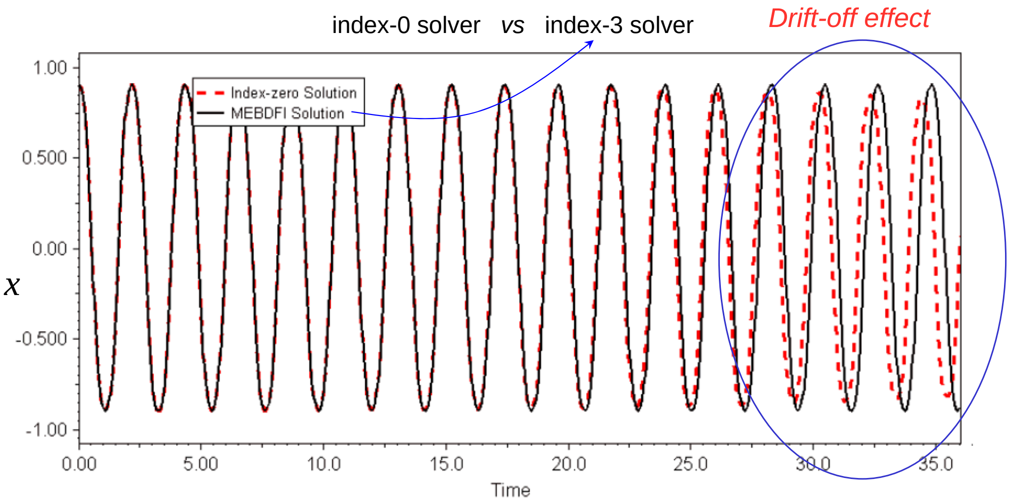
– High-Index DAE: consistency analysis –



65

Error propagation

index-0 solver *vs* index-3 solver

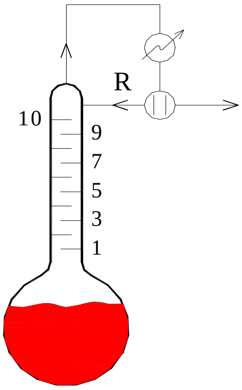


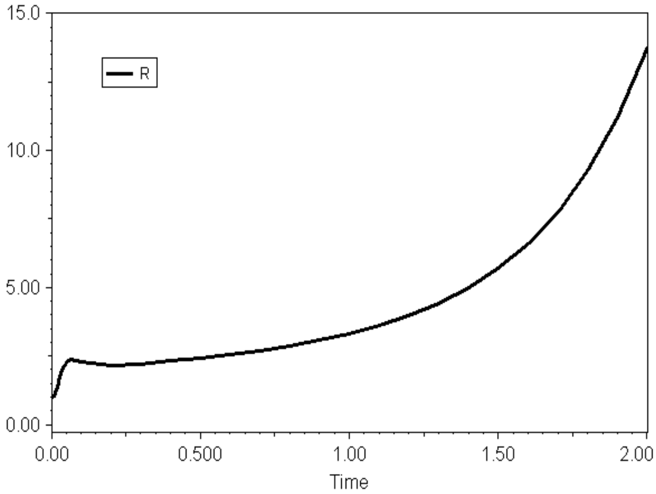
L = 0.9 m , g = 9.8 m/s^2 $\therefore$ I.C.: $x(0)$ = 0.9 m and $w(0) = 0$

Dynamic Degree of Freedom
– Workshop –

Batch Distillation

Batch distillation column with optimal composition control (index 3) (Logsdson and Biegler, 1993)





67

Model assumptions

- negligible vapor holdup (no dynamics in vapor phase);
- thermodynamic equilibrium (ideal stage);
- no droplet drag in vapor stream;
- ideal gas and liquid;
- constant liquid holdup in each tray;
- perfect mixture in both phases;
- constant pressure;
- optimal control of distillate composition;
- vapor pressure described by Antoine equation.

Batch distillation modeling

Mass balance in the reboiler

Overall:

$$\frac{dH_0}{dt} = -\frac{V}{R+1} \qquad (1)$$

Component:

$$\frac{dx_0^j}{dt} = \frac{V}{H_0}\left[x_0^j - y_0^j + \frac{R}{R+1}\left(x_1^j - x_0^j\right)\right] \qquad j = 1,\ldots,nc-1 \qquad (2)$$

Mass balance in each tray

component:

$$\frac{dx_i^j}{dt} = \frac{V}{H_i}\left[y_{i-1}^j - y_i^j + \frac{R}{R+1}\left(x_{i+1}^j - x_i^j\right)\right] \qquad \begin{array}{c} i = 1,\ldots,np \\ j = 1,\ldots,nc-1 \end{array} \qquad (3)$$

69

Batch distillation modeling

Mass balance in the condenser

Component:

$$\frac{dx_{np+1}^{j}}{dt}=\frac{V}{H_{np+1}}\left(y_{np}^{j}-x_{np+1}^{j}\right) \qquad j=1,\ldots,nc-1 \qquad (4)$$

Molar fractions

$$\sum_{j=1}^{nc} y_i^j = 1 \qquad i=0,\ldots,np+1 \qquad (5)$$

$$\sum_{j=1}^{nc} x_i^j = 1 \qquad i=0,\ldots,np+1 \qquad (6)$$

Thermodynamic equilibrium

$$y_i^j\,P = P_{ref}\,\exp\left(A_j-\frac{B_j}{T_i+C_j}\right)x_i^j \qquad i=0,\ldots,np+1;\ j=1,\ldots,nc \qquad (7)$$

Consistency analysis

variable	units of measurement
H_i	kmol
V	kmol/s
t	s
R	–
x_i^j, y_i^j	kmol/kmol
P, P_{ref}	kPa
T_i	K
A_j	–
B_j	K
C_j	K

Dynamic Degree of Freedom
– Workshop –

Consistency analysis

variables: H_i, V, t, R, x_i^j, y_i^j, P, P_{ref}, T_i, A_j, B_j, C_j ➔ $5 + 2\,(np+2)(nc+1) + 3\,nc$
constants: P_{ref}, A_j, B_j, C_j ➔ $3\,nc + 1$
specifications: t,V, P, H_i, x^1_{np+1} ➔ $5 + np$ $(i = 1,...,np+1)$
driving forces: 0
unknown variables: H_0, R, x_i^j, y_i^j, T_i ➔ $3 + 2\,(np+2)\,nc + np$
equations: $3 + 2\,(np+2)\,nc + np$

Degree of Freedom = variables – constants – specifications – driving forces – equations = unknown variables – equations = 0

Dynamic Degree of Freedom (index = 3) = $np\,(nc-1) + 2\,(nc-2)$

Needs $np\,(nc-1) + 2\,(nc-2)$ initial conditions.

Para $nc = 2$: $H_0(0)$, $R(0)$, $x_0^1(0)$, $T_i(0)$ $(i = 2,...,np-2)$

72

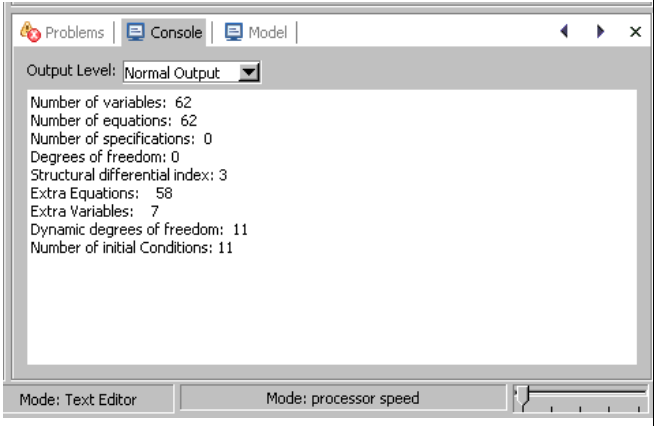
EMSO

```
VARIABLES
    H0    as   Real(Default=1);
    x1(Np+1)  as Real(Default=0.55, Lower = 0, Upper = 1);
    x2(Np+1)  as Real(Default=0.45, Lower = 0, Upper = 1);
    y1(Np+1)  as Real(Default=0.3, Lower = 0, Upper = 1);
    y2(Np+1)  as Real(Default=0.7, Lower = 0, Upper = 1);
    T(Np+1)   as Real(Default=85, Lower = 0, Upper = 400);
    R as Real(Default=1);

EQUATIONS
    "Balanço no Prato 0"
    diff(H0)*'s' = -V/(R+1);
    diff(x1(1))*'s' = V/H0 * (x1(1)-y1(1) + R * ((x1(2) - x1(1))/(R+1)));

    "Balanço demais pratos"
    diff(x1(2:Np))*'s' = V/Hi * (y1(1:Np-1) - y1(2:Np) + (R * (x1(3:Np+1) - x1(2:Np))/(R+1)));

    "Balanço último prato"
    diff(x1(Np+1))*'s' = V * (y1(Np) - x1(Np+1))/Hi;

    x1  = 1 - x2;
    y1*P = x1 * exp(A(1) - (B(1)/(T + 273.15 + C(1))));
    y1  = 1 - y2;
    y2*P = x2 * exp(A(2) - (B(2)/(T + 273.15 + C(2))));

    "Equação de Controle Ótimo"
    x1(Np+1) = purity;

INITIAL
    H0 = 100;
    x1(1) = 0.55;
    T(2:9) = [89.8, 87.5, 85.4, 83.8, 82.6, 81.7, 81.1, 81.0];
    R = 1;
```

Dynamic Degree of Freedom
– Workshop –

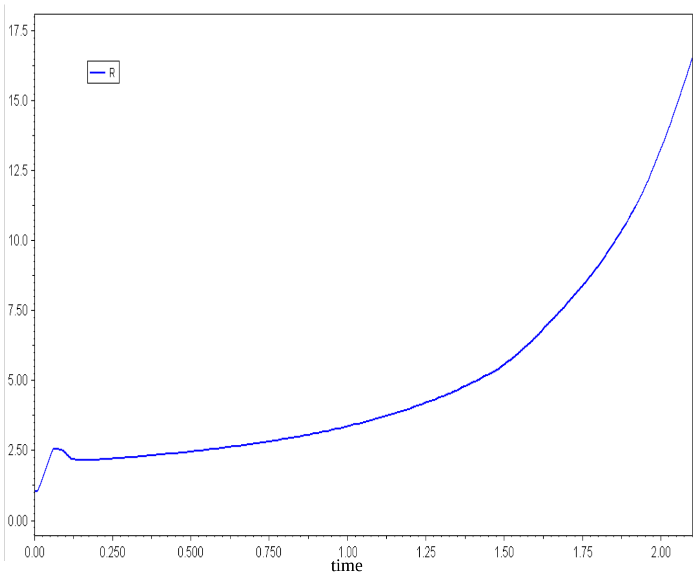
EMSO

```
FlowSheet batch  as  BatchColumn
    SET
        Nc = 2;
        Np = 11;

        A = [15.7527, 16.0137];
        B = [2766.63, 3096.52];
        C = [ -50.5, -53.67];

        purity = 0.998;
        P = 760;
        Hi = 1;
        V = 120;

    OPTIONS
        TimeStep = 0.01;
        TimeEnd = 2.1;
        Integration = "original";
#       Integration = "index0";
        DAESolver(
            File = "dasslc" # "mebdf"
        );
end
```



Dynamic Degree of Freedom
– Workshop –

EMSO

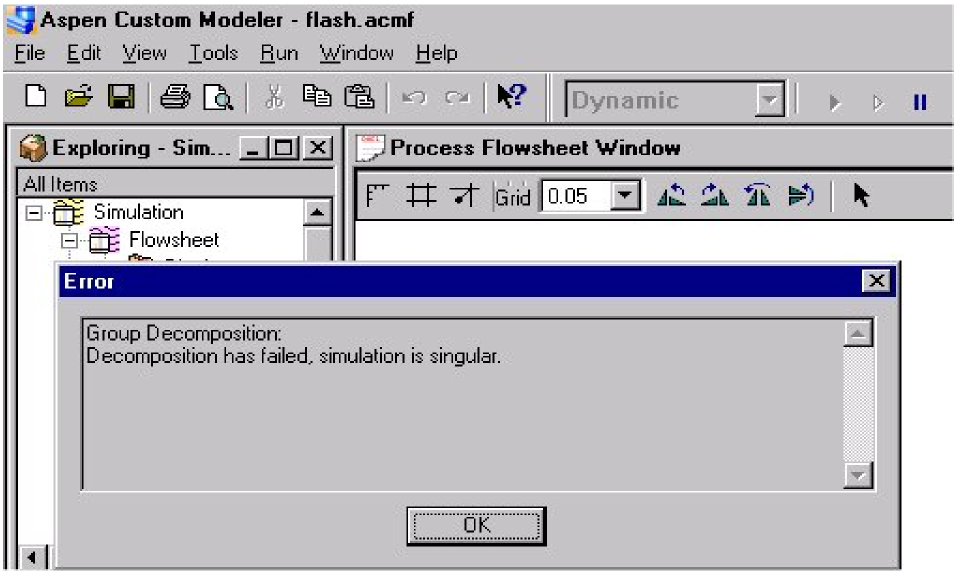


75

Dynamic Degree of Freedom
– Workshop –

AspenDynamics

- **Reports system singularity:**

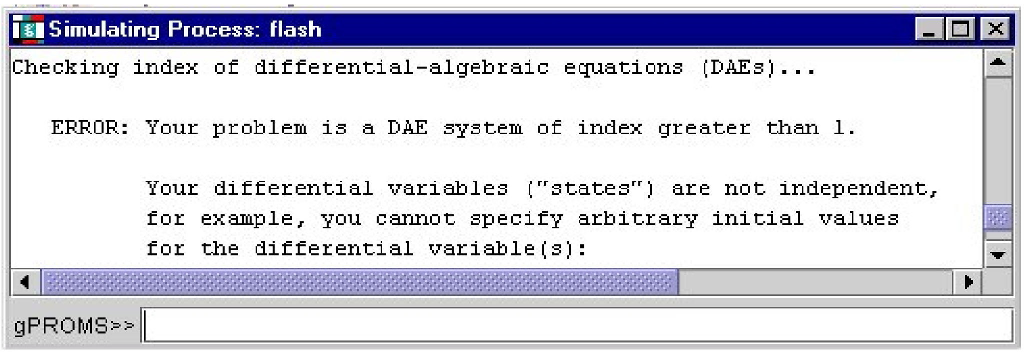


76

Dynamic Degree of Freedom
– Workshop –

gPROMS

- **Detects a high-index problem and gives the following error message:**

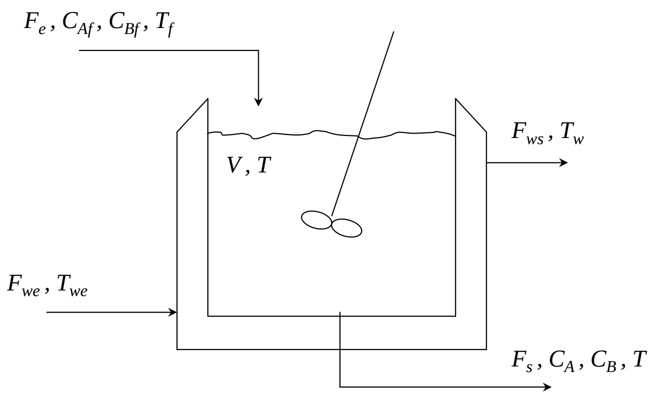


5. System Analysis

- Multiplicity of steady states
- Linearization
- System stability
- Complex dynamic behaviors (limit cycles, strange attractors)
- Parametric sensitivity and input sensitivity

Multiplicity of Steady States

Non-isothermal CSTR



$$A \xrightarrow{k} B$$

79

Multiplicity of Steady States

Process description

In a non-isothermal continuous stirred tank reactor, with diameter of 3.2 m and level control, pure reactant is fed at 300 K and 3.5 m^3/h with concentration of 300 $kmol/m^3$. A first order reaction occur in the reactor, with frequency factor of 89 s^{-1} and activation energy of 6 x 10^4 kJ/kmol, releasing 7000 kJ/kmol of reaction heat. The reactor has a jacket to control the reactor temperature, with constant overall heat transfer coefficient of 300 kJ/(h.m^2.K). Assume constant density of 1000 kg/m^3 and constant specific heat of 4 kJ/(kg.K) in the reaction medium. The fully-open output linear valve has a constant of 2.7 $m^{2.5}/h$.

Multiplicity of Steady States

Model assumptions

- perfect mixture in the reactor and jacket;
- negligible shaft work;
- $(-r_A) = k\, C_A$;
- constant density;
- constant overall heat transfer coefficient;
- constant specific heat;
- incompressible fluids;
- negligible heat loss to surroundings;
- Δ(internal energy) $\approx$ Δ(enthalpy);
- negligible variation of potential and kinetic energies;
- constant volume in the jacket;
- thin metallic wall with negligible heat capacity.

Multiplicity of Steady States

CSTR modeling

Mass balance in the reactor

Overall:

$$\frac{d(\rho V)}{dt} = \rho_f F_e - \rho F_s = \rho \frac{dV}{dt}$$

$$\frac{dV}{dt} = F_e - F_s \qquad (1)$$

Component:

$$\frac{d(VC_A)}{dt} = V\frac{dC_A}{dt} + C_A\frac{dV}{dt} = F_e C_{Af} - F_s C_A - V(-r_A)$$

$$V\frac{dC_A}{dt} = F_e(C_{Af} - C_A) - (-r_A)V \qquad (2)$$

$$\tau = \frac{V}{F_e} \qquad (3)$$

82

Multiplicity of Steady States

CSTR modeling

Energy balance in the reactor:

$$\frac{d}{dt}\left[\rho V\left(\hat{U}+\hat{K}+\hat{\phi}\right)\right]=F_e\rho\left(\hat{U}_f+P_f\hat{V}_f+\frac{v_f^2}{2}+gz_f\right)-F_s\rho\left(\hat{U}+P\hat{V}+\frac{v_s^2}{2}+gz_s\right)+q_r-q-w_s$$

where $\hat{H}=\hat{U}+P\hat{V}$

$$\frac{d(\rho V\hat{H})}{dt}=\rho V\frac{d\hat{H}}{dt}+\rho\hat{H}\frac{dV}{dt}=F_e\rho\hat{H}_f-F_s\rho\hat{H}+q_r-q$$

$$\rho V\frac{d\hat{H}}{dt}=F_e\rho(\hat{H}_f-\hat{H})+q_r-q$$

$$\rho VCp\frac{dT}{dt}=F_e\rho Cp(T_f-T)+q_r-q \qquad (4)$$

Multiplicity of Steady States

CSTR modeling

where

$$q = U A_t (T - T_w) \quad (5)$$

$$q_r = (-\Delta H_r) V (-r_A) \quad (6)$$

$$(-r_A) = k C_A \quad (7)$$

$$k = k_0 \exp(-E/RT) \quad (8)$$

$$A = \pi D^2/4 \quad (9)$$

$$V = A h \quad (10)$$

$$A_t = A + \pi D h \quad (11)$$

$$F_s = x\, Cv \sqrt{h} \quad (12)$$

$x = f(h)$ Level control (13)

$T_w = f(T)$ Temperature control (14)

Consistency analysis

variable	units of measurement
F_e, F_s	$m^3\ s^{-1}$
V	m^3
t, τ	s
C_A, C_{Af}	$kmol\ m^{-3}$
r_A	$kmol\ m^{-3}\ s^{-1}$
ρ	$kg\ m^{-3}$
Cp	$kJ\ kg^{-1}\ K^{-1}$
T, T_f, T_w	K
q_r, q	$kJ\ s^{-1}$
U	$kJ\ m^{-2}\ K^{-1}\ s^{-1}$
A_t, A	m^2
h, D	m
Cv	$m^{2.5}\ h^{-1}$
x	–
ΔH_r, E	$kJ\ kmol^{-1}$
R	$kJ\ kmol^{-1}\ K^{-1}$
k, k_0	s^{-1}

Multiplicity of Steady States

Consistency analysis

variables: $F_e, F_s, V, t, C_A, C_{Af}, r_A, \rho, Cp, T, T_f, T_w, q_r, q, U, A_t, A, h, D, Cv, x, \Delta Hr, E, R, k, k_0, \tau$ ➔ 27
constants: $\rho, Cp, U, D, Cv, \Delta Hr, E, R, k_0$ ➔ 9
specifications: t ➔ 1
driving forces: F_e, T_f, C_{Af} ➔ 3
unknown variables: $F_s, V, C_A, r_A, T, T_w, q_r, q, A, A_t, h, x, k, \tau$ ➔ 14
equations: 14

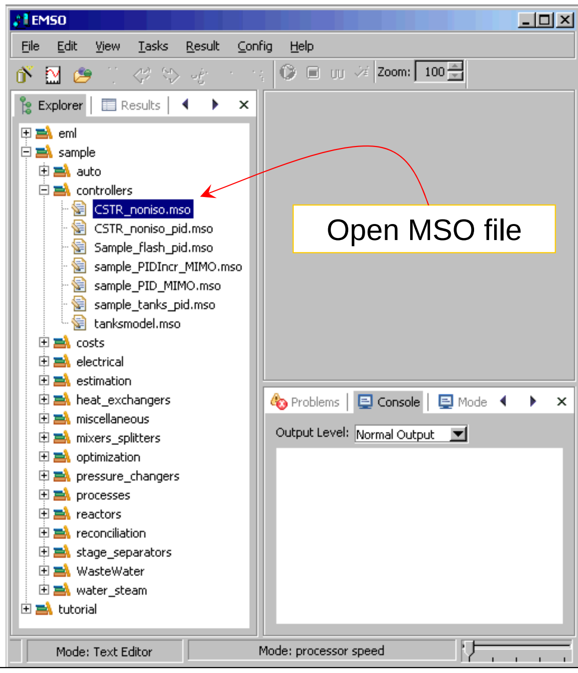
Degree of Freedom = variables – constants – specifications – driving forces – equations = unknown variables – equations = 27 – 9 – 1 – 3 – 14 = 0

Dynamic Degree of Freedom (index < 2) = differential equations = 3 ➔

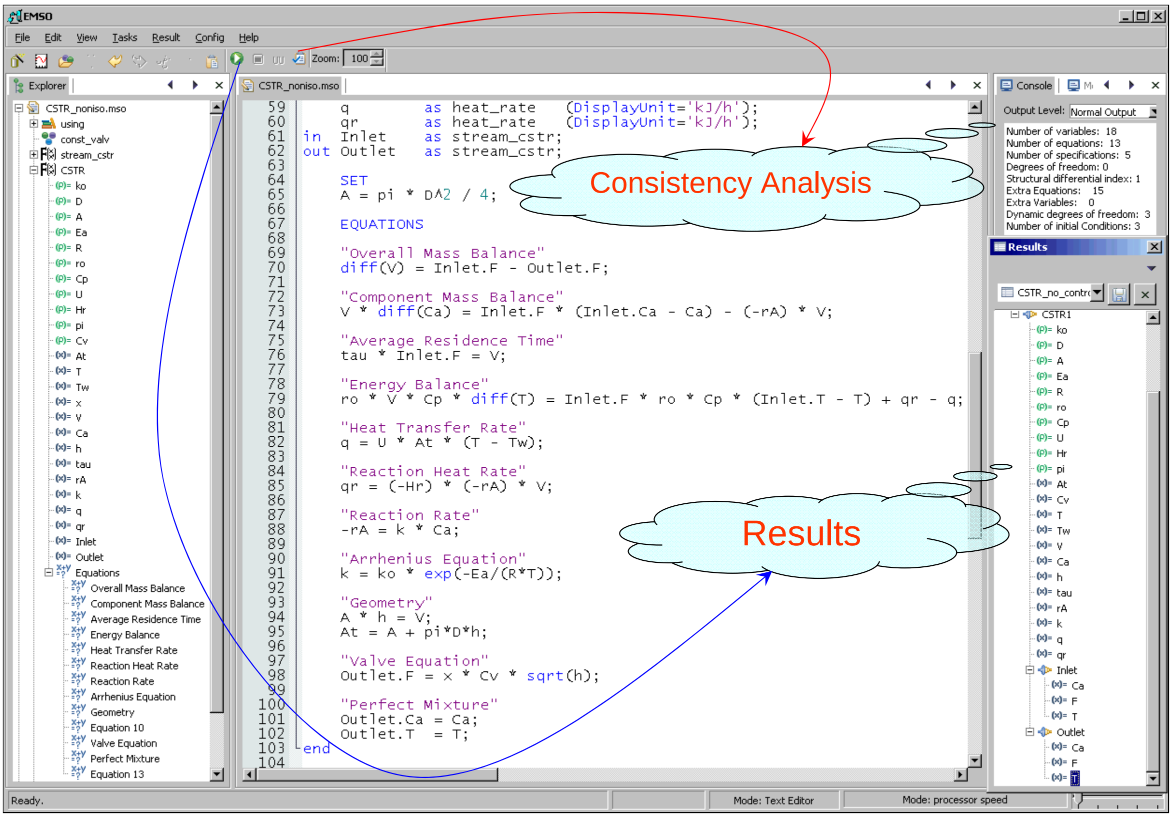
Needs 3 initial condition: $h(0)$, $C_A(0)$, $T(0)$ ➔ 3

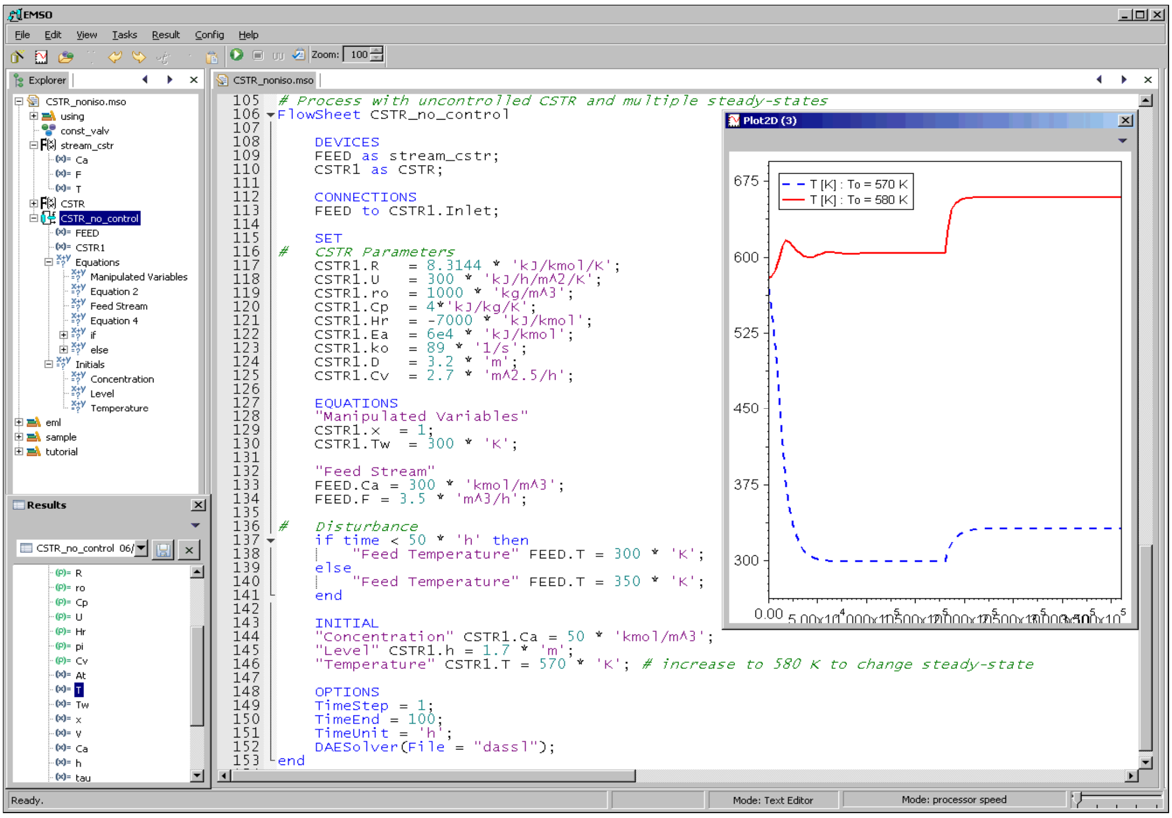
Multiplicity of Steady States

- Running EMSO



87


EMSO
File Edit View Tasks Result Config Help
Zoom: 100
Explorer
CSTR_noniso.mso
using
const_valv
stream_cstr
CSTR
Equations
Overall Mass Balance
Component Mass Balance
Average Residence Time
Energy Balance
Heat Transfer Rate
Reaction Heat Rate
Reaction Rate
Arrhenius Equation
Geometry
Equation 10
Valve Equation
Perfect Mixture
Equation 13
q as heat_rate (DisplayUnit='kJ/h');
qr as heat_rate (DisplayUnit='kJ/h');
in Inlet as stream_cstr;
out Outlet as stream_cstr;
SET
A = pi * D^2 / 4;
EQUATIONS
"Overall Mass Balance"
diff(V) = Inlet.F - Outlet.F;
"Component Mass Balance"
V * diff(Ca) = Inlet.F * (Inlet.Ca - Ca) - (-rA) * V;
"Average Residence Time"
tau * Inlet.F = V;
"Energy Balance"
ro * V * Cp * diff(T) = Inlet.F * ro * Cp * (Inlet.T - T) + qr - q;
"Heat Transfer Rate"
q = U * At * (T - Tw);
"Reaction Heat Rate"
qr = (-Hr) * (-rA) * V;
"Reaction Rate"
-rA = k * Ca;
"Arrhenius Equation"
k = ko * exp(-Ea/(R*T));
"Geometry"
A * h = V;
At = A + pi*D*h;
"Valve Equation"
Outlet.F = x * Cv * sqrt(h);
"Perfect Mixture"
Outlet.Ca = Ca;
Outlet.T = T;
end
Consistency Analysis
Results
Console
Output Level: Normal Output
Number of variables: 18
Number of equations: 13
Number of specifications: 5
Degrees of freedom: 0
Structural differential index: 1
Extra Equations: 15
Extra Variables: 0
Dynamic degrees of freedom: 3
Number of initial Conditions: 3
Results
CSTR1
Inlet
Outlet
Ready.
Mode: Text Editor
Mode: processor speed


EMSO
File Edit View Tasks Result Config Help
Zoom: 100
Explorer
CSTR_noniso.mso
using
const_valv
stream_cstr
Ca
F
T
CSTR
CSTR_no_control
FEED
CSTR1
Equations
Manipulated Variables
Equation 2
Feed Stream
Equation 4
if
else
Initials
Concentration
Level
Temperature
eml
sample
tutorial
Results
CSTR_no_control 06/
R
ro
Cp
U
Hr
pi
Cv
At
T
Tw
x
v
Ca
h
tau
Process with uncontrolled CSTR and multiple steady-states
FlowSheet CSTR_no_control
DEVICES
FEED as stream_cstr;
CSTR1 as CSTR;
CONNECTIONS
FEED to CSTR1.Inlet;
SET
# CSTR Parameters
CSTR1.R = 8.3144 * 'kJ/kmol/K';
CSTR1.U = 300 * 'kJ/h/m^2/K';
CSTR1.ro = 1000 * 'kg/m^3';
CSTR1.Cp = 4*'kJ/kg/K';
CSTR1.Hr = -7000 * 'kJ/kmol';
CSTR1.Ea = 6e4 * 'kJ/kmol';
CSTR1.ko = 89 * '1/s';
CSTR1.D = 3.2 * 'm';
CSTR1.Cv = 2.7 * 'm^2.5/h';
EQUATIONS
"Manipulated Variables"
CSTR1.x = 1;
CSTR1.Tw = 300 * 'K';
"Feed Stream"
FEED.Ca = 300 * 'kmol/m^3';
FEED.F = 3.5 * 'm^3/h';
# Disturbance
if time < 50 * 'h' then
"Feed Temperature" FEED.T = 300 * 'K';
else
"Feed Temperature" FEED.T = 350 * 'K';
end
INITIAL
"Concentration" CSTR1.Ca = 50 * 'kmol/m^3';
"Level" CSTR1.h = 1.7 * 'm';
"Temperature" CSTR1.T = 570 * 'K'; # increase to 580 K to change steady-state
OPTIONS
TimeStep = 1;
TimeEnd = 100;
TimeUnit = 'h';
DAESolver(File = "dassl");
end
Plot2D (3)
T [K] : To = 570 K
T [K] : To = 580 K
675
600
525
450
375
300
0.00
Ready.
Mode: Text Editor
Mode: processor speed

Multiplicity of Steady States

The CSTR example at the steady state satisfy:

$$\frac{1}{\tau}(T - T_f) + \frac{U A_t}{\rho V C_P}(T - T_w) = \frac{(-\Delta H_r)\, k_0\, e^{-\frac{E}{RT}}\, C_{Af}}{\rho C_P \left(1 + \tau\, k_0\, e^{-\frac{E}{RT}}\right)}$$

$$C_A = \frac{C_{Af}}{\left(1 + \tau\, k_0\, e^{-\frac{E}{RT}}\right)}$$

$$\tau = \frac{V}{F_e}$$

Multiplicity of Steady States

Rewriting the energy balance:

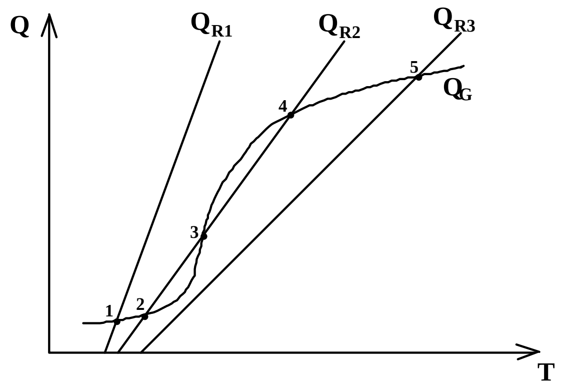
$$Q_R(T) = Q_G(T)$$

$$Q_R(T) = a\,T - b$$

$$Q_G(T) = \frac{(-\Delta H_r)\,k_0\,e^{-\frac{E}{RT}}C_{Af}}{\rho C_P\left(1+\tau\,k_0\,e^{-\frac{E}{RT}}\right)}$$

$$a = \frac{1}{\tau} + \frac{U A_t}{\rho V C_P}$$

$$b = \frac{T_f}{\tau} + \frac{U A_t T_w}{\rho V C_P}$$



stable:

$$\frac{dQ_R}{dT} > \frac{dQ_G}{dT}$$

unstable:

$$\frac{dQ_R}{dT} < \frac{dQ_G}{dT}$$

Multiplicity of Steady States

Path Following

$$\frac{dx}{dt} = F(t,x) \longrightarrow F(x) = 0$$

Newton-Raphson: $x^{(k+1)} = x^{(k)} - \alpha\left[J(x^{(m)})\right]^{-1} F(x^{(k)}) \quad ,k = 0,1,2,\ldots$

$$J_{ij}(x^{(k)}) = \frac{\partial F_i(x^{(k)})}{\partial x_j}$$

$m \le k$ and $0 < \alpha \le 1$

Homotopic Continuation: $H(x;p) = (1-p)F(x) + p\,G(x) = 0 \quad ,0 \le p \le 1$

$G(x) = J(x^{(0)})(x - x^{(0)})$ affine homotopy

$G(x) = F(x) - F(x^{(0)})$ Newton homotopy

→ Multiples solutions can be obtained by continuously varying the parameter *p*

Multiplicity of Steady States

Path Following

Parametric Continuation: $F[x(s); p(s)] = 0$

where *s* is some parameterization, e.g., path arc length

$$\frac{\partial F}{\partial x}\dot{x}(s) + \frac{\partial F}{\partial p}\dot{p}(s) = 0 \quad , \dot{x} = \frac{dx}{ds} \quad \text{e} \quad \dot{p} = \frac{dp}{ds}$$

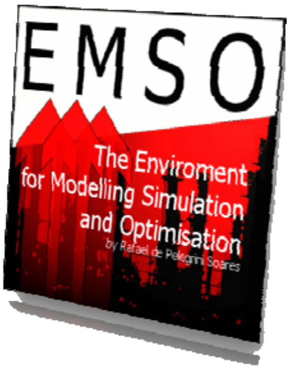
$$DF = \left[\frac{\partial F}{\partial x} \;\; \frac{\partial F}{\partial p} \right]$$ Frechet derivative

a point (x_o, p_o) is:

- Regular if $\frac{\partial F(x_o, p_o)}{\partial x}$ is non-singular
- Turning point if $\frac{\partial F(x_o, p_o)}{\partial x}$ is singular and DF has rank = n ⟶ reparameterization
- Bifurcation if $\frac{\partial F(x_o, p_o)}{\partial x}$ is singular and DF has rank < n

Multiplicity of Steady States

Example: a) execute flowsheet in file CSTR_noniso.mso with initial condition of 578 K and compare with result changing the initial condition to 579 K; b) find the three steady states using file CSTR_sea.mso by changing the initial guess for T and C_A (use the section GUESS).



Solutions: 1) C_A = 13,13 kmol/m^3 and T = 659,46 K
2) C_A = 132,87 kmol/m^3 and T = 523,01 K
3) C_A = 299,86 kmol/m^3 and T = 332,72 K

Linearization

Generate linearized model at given operating point.

Implicit DAE: $F(\tilde{x}', \tilde{x}, t) = 0$

Considering the specification as input, $u(t)$, (SPECIFY section in EMSO):

$$F(\hat{x}', \hat{x}, u, t) = 0$$

And identifying the algebraic variables as $y(t)$:

$$F(x', x, y, u, t) = 0$$

Linearization

Differentiating F: $F_{x'}\, dx' + F_x\, dx + F_y\, dy + F_u\, du = 0$

and extracting: $\begin{bmatrix} dx' \\ dy \end{bmatrix} = -\begin{bmatrix} F_{x'} & F_y \end{bmatrix}^{-1} \begin{bmatrix} F_x & F_u \end{bmatrix} \begin{bmatrix} dx \\ du \end{bmatrix}$ (index < 2)

The partition: $-\begin{bmatrix} F_{x'} & F_y \end{bmatrix}^{-1} \begin{bmatrix} F_x & F_u \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$

Define the linearized system:

$$x' = Ax + Bu$$

$$y = Cx + Du$$

96

Linearization

Test example for a linear model: exact solution!

sample_linear.mso

```
using "types";

FlowSheet linear
    PARAMETERS
    nx      as Integer(Default=2);
    ny      as Integer(Default=2);
    nu      as Integer(Default=2);

    A(nx,nx) as Real;
    B(nx,nu) as Real;
    C(ny,nx) as Real;
    D(ny,nu) as Real;

    VARIABLES
    x(nx)   as Real (Brief="State Variables");
    y(ny)   as Real (Brief="Output Variables");
    u(nu)   as Real (Brief="Control Variables");

    EQUATIONS

    diff(x)*'s' = sumt(A*x) + sumt(B*u);
    y           = sumt(C*x) + sumt(D*u);

    SPECIFY
    u(1:nu) = sin((time/'s' + [1:nu])*'rad');
```

97

Linearization

Verifying the results for the linear model:

sample_linear.mso

```
	SET

	nx = 7;
	ny = 3;
	nu = 5;

	A = 0;
	B = 0;
	C = 0;
	D = 0;

	A(1,1) = 0.5;
	A(2,2) = 0.5;
	A(3,3) = 0.5;

	B(1,1) = 0.75;
	B(2,2) = 0.75;
	B(3,3) = 0.75;

	C(2,1) = 0.85;
	D = 0.95;

	INITIAL
	x = 0;

	OPTIONS
	TimeStep = 0.1;
	TimeEnd = 2;
	TimeUnit = 's';
	SparseAlgebra = true;
	Linearize = true;
end
```

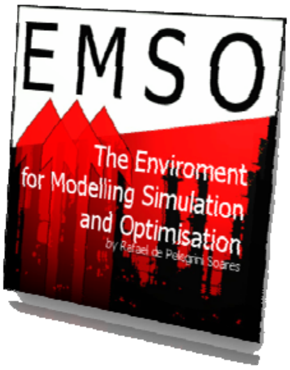
Console

Output Level: Normal Output

```
Advancing time from 1.4 to 1.5
Advancing time from 1.5 to 1.6
Advancing time from 1.6 to 1.7
Advancing time from 1.7 to 1.8
Advancing time from 1.8 to 1.9
Advancing time from 1.9 to 2
Linearized system:
A:
0.500000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.500000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.500000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000

B:
0.750000 0.000000 0.000000 0.000000 0.000000
0.000000 0.750000 0.000000 0.000000 0.000000
0.000000 0.000000 0.750000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000

C:
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.850000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000

D:
0.950000 0.950000 0.950000 0.950000 0.950000
0.950000 0.950000 0.950000 0.950000 0.950000
0.950000 0.950000 0.950000 0.950000 0.950000

Simulation of 'linear' finished successfully in 0.234 seconds.
```

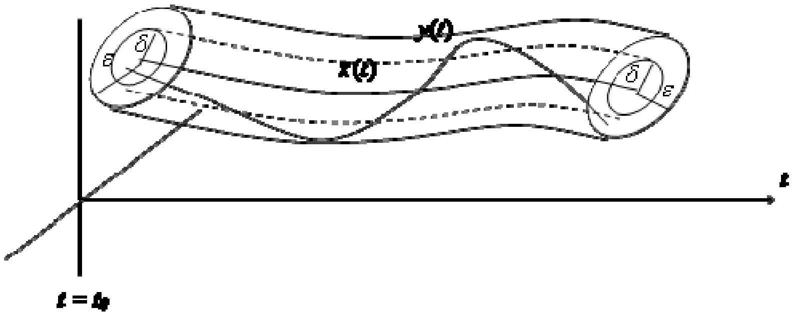
Non-isothermal CSTR: linearization

Example: execute the flowsheet in file CSTR_linearize.mso with the option Linearize = true and evaluate the characteristic values of the Jacobian matrix (matrix A). Repeat the example with the value of Cp 10 times smaller, i.e., 0.4 kJ / (kg K). Compare the ratio between the greater and the smaller characteristic values in module.



Stability Analysis

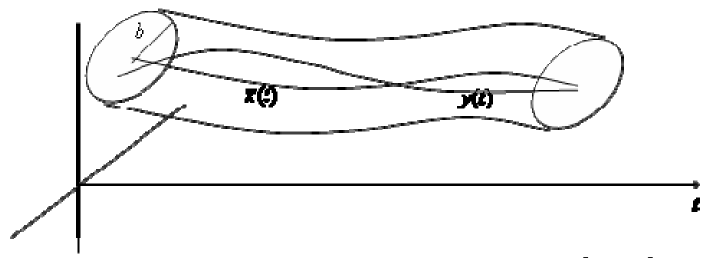
Liapunov Stability: $\bar{x}(t)$ is stable (or Liapunov stable) if, given ε > 0, there exists a δ = δ(ε) > 0, such that, for any other solution, $y(t)$, of $\frac{dx}{dt} = F(x)$ satisfying $|\bar{x}(t_0) - y(t_0)| < \delta$, then $|\bar{x}(t) - y(t)| < \varepsilon$ for $t > t_0$.



100

Stability Analysis

Asymptotic Stability: $\bar{x}(t)$ is asymptotic stable if Liapunov stable and there exists a constant $b > 0$ such that, if $|\bar{x}(t_0) - y(t_0)| < b$ then $\lim_{t \to \infty} |\bar{x}(t) - y(t)| = 0$



Defining deviation variables: $y(t) = x(t) - \bar{x}(t) \longrightarrow \frac{dx}{dt} = \frac{d\bar{x}}{dt} + \frac{dy}{dt} = F(\bar{x}(t) + y)$

Expanding in Taylor series: $\frac{dx}{dt} = F(x) = F(\bar{x}(t)) + \frac{\partial F[\bar{x}(t)]}{\partial x} \cdot y + O\left(\|y\|^2\right)$

Linearization: $\frac{dy}{dt} = J[\bar{x}(t)] \cdot y = A(t) \cdot y$

101

Stability Analysis

For an equilibrium point $\bar{x}(t) = x^*$, the stability is characterized by the characteristics values of the Jacobian matrix $J(x^*) = A$:

⇒ x^* is a hyperbolic point if none characteristics values of $J(x^*)$ has zero real part.

⇒ x^* is a center if the characteristics values are pure imaginary. Fixed point non-hyperbolic.

⇒ x^* is a saddle point, unstable, if some characteristics values have real part > 0 and the remaining have real part < 0.

⇒ x^* is stable or attractor or sink point if all characteristics values have real part < 0.

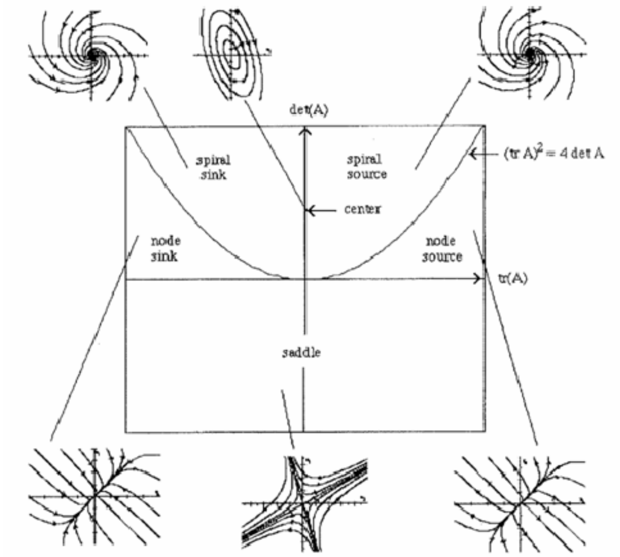
⇒ x^* is unstable or repulsive or source point if at least one characteristic value have real part > 0.

102

Stability Analysis

For a second-order linear system:

$$\det(A-\lambda I)=\lambda^2 - tr(A)\cdot\lambda+\det(A)=0 \longrightarrow \lambda=\frac{tr(A)\pm\sqrt{tr(A)^2-4\det(A)}}{2}$$



Stability Analysis

Considering the CSTR example with constant volume:

$$\frac{dC_A}{dt} = \frac{F_e}{V}(C_{Af} - C_A) - k_0\, e^{-\frac{E}{RT}} C_A \quad , \quad C_A(0) = C_{A0}$$

$$\frac{dT}{dt} = \frac{F_e}{V}(T_f - T) + \frac{(-\Delta H_r)\, k_0\, e^{-\frac{E}{RT}} C_A}{\rho\, C_p} - \frac{UA_t (T - T_w)}{V\, \rho\, C_p} \quad , T(0) = T_0$$

$$J(C_A, T) = \begin{bmatrix} -\frac{F_e}{V} - k_0\, e^{-\frac{E}{RT}} & -\frac{E}{RT^2} k_0\, e^{-\frac{E}{RT}} C_A \\ \frac{(-\Delta H_r)\, k_0\, e^{-\frac{E}{RT}}}{\rho\, C_p} & -\frac{F_e}{V} + \frac{E}{RT^2} \frac{(-\Delta H_r)\, k_0\, e^{-\frac{E}{RT}} C_A}{\rho\, C_p} - \frac{UA_t}{V\, \rho\, C_p} \end{bmatrix}$$

Stability Analysis

1) Stable node

$$J(13.13,\ 659.46)=\begin{bmatrix}-1.6458\times10^{-3} & -3.4282\times10^{-4}\\ 2.7542\times10^{-3} & 4.8934\times10^{-4}\end{bmatrix}\qquad \lambda=\begin{bmatrix}-1.0205\times10^{-3}\\ -1.3604\times10^{-4}\end{bmatrix}$$

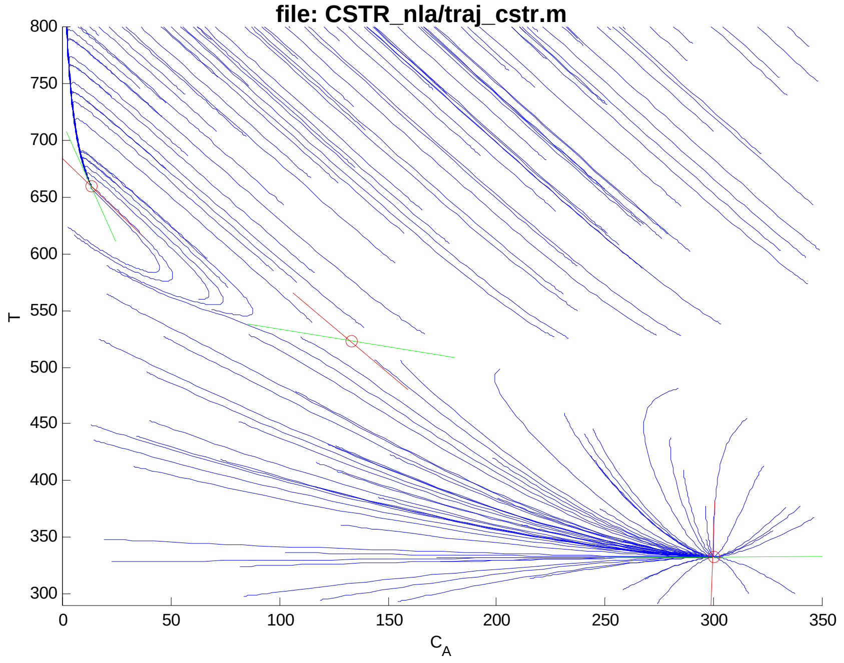
2) Saddle Point, unstable

$$J(132.87,\ 523.01)=\begin{bmatrix}-1.6260\times10^{-4} & -3.1753\times10^{-4}\\ 1.5852\times10^{-4} & 4.4509\times10^{-4}\end{bmatrix}\qquad \lambda=\begin{bmatrix}-6.3659\times10^{-5}\\ 3.4614\times10^{-4}\end{bmatrix}$$

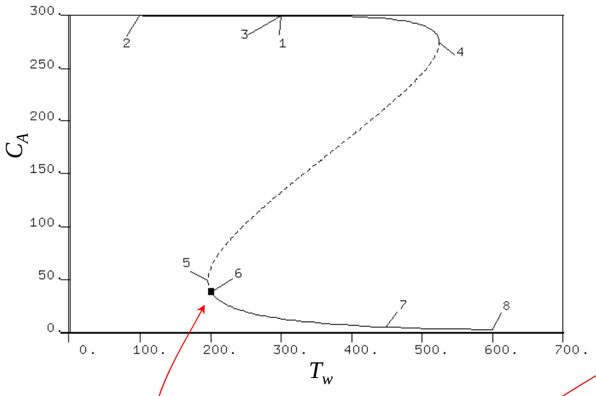
3) Stable Node

$$J(299.86,\ 332.72)=\begin{bmatrix}-7.2050\times10^{-5} & -6.6220\times10^{-7}\\ 5.9285\times10^{-8} & -1.0944\times10^{-4}\end{bmatrix}\qquad \lambda=\begin{bmatrix}-7.2051\times10^{-5}\\ -1.0944\times10^{-4}\end{bmatrix}$$

Stability Analysis



Complex Dynamic Behavior



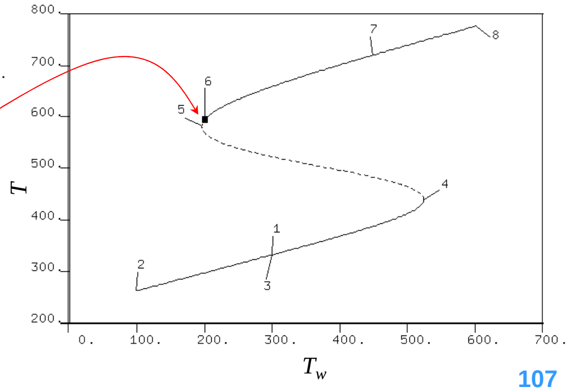
CSTR example:

stable solutions

unstable solutions

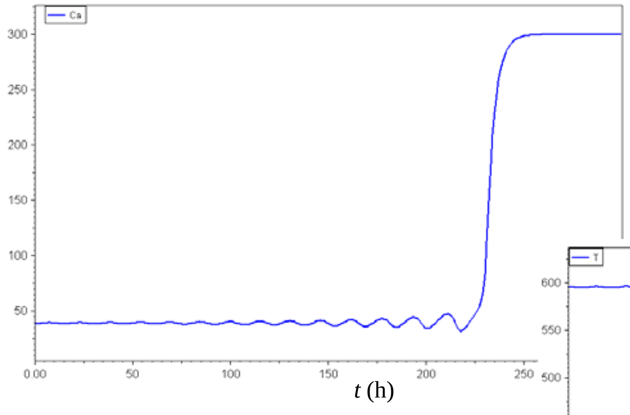
Hopf point

T_w = 200,37 K



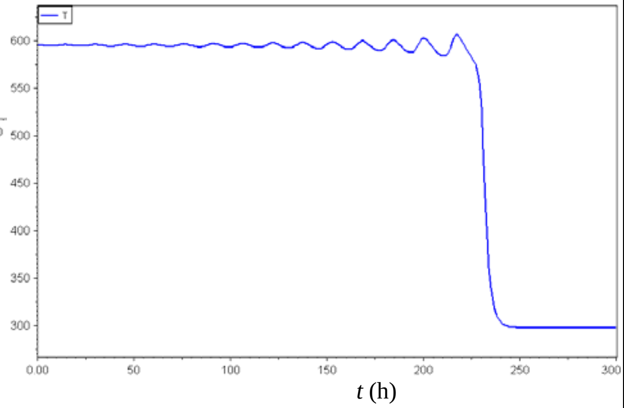
107

Complex Dynamic Behavior



unstable limit cycle

file: CSTR_auto/cstr_bif.mso



A limit cycle is stable if all characteristics values of $\exp(J\,p)$ (Floquet multipliers) are inside the unitary cycle, where J is the Jacobian matrix in the cycle, $p = 2\,\pi\,/\,\beta$ is the oscillation period and $\beta = |\lambda_{Hopf}|$.

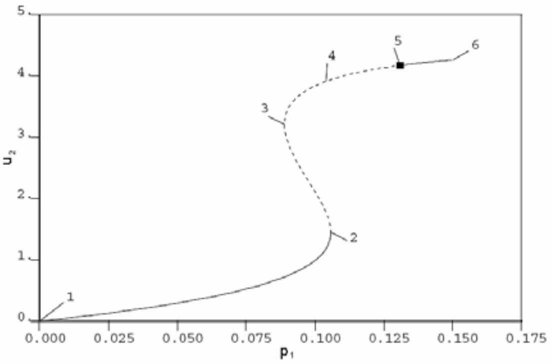
Interface EMSO-AUTO

parameters

EMSO → Equation system / Jacobian matrix / First steady-state solution → **AUTO**

(AUTO → EMSO: parameters)

```
using "types";

FlowSheet ab_dae

    PARAMETERS
    p1 as Real;
    p2 as Real;
    p3 as Real;

    VARIABLES
    u1 as Real;
    u2 as Real;
    u3 as Real;

    SET

    p1 = 0;
    p2 = 14;
    p3 = 2;

    EQUATIONS

    diff(u1)*'s' = -u1 + p1 * (1 - u1) * u3;
    diff(u2)*'s' = -u2 + p1 * p2 * (1 - u1) * u3 - p3 * u2;
    u3 = exp(u2);
```



Interface EMSO-AUTO

$$\frac{dx_1(t)}{dt} = -x_1(t) + p \cdot \left[1 - x_1(t)\right] e^{x_2(t)}$$

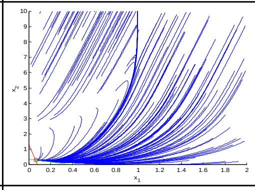
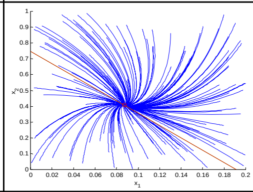
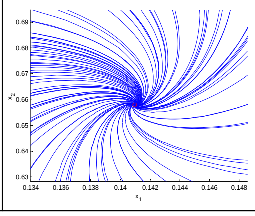
$$\frac{dx_2(t)}{dt} = -3x_2(t) + 14\,p \cdot \left[1 - x_1(t)\right] e^{x_2(t)}$$

$$J(x) = \begin{bmatrix} -1 - p \cdot e^{x_2} & p \cdot (1 - x_1) \cdot e^{x_2} \\ -14\,p \cdot e^{x_2} & -3 + 14\,p \cdot (1 - x_1) \cdot e^{x_2} \end{bmatrix}$$

$p = 0$: $x^* = (0, 0)$ $\lambda(J) = (-1, -3)$

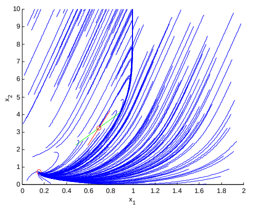
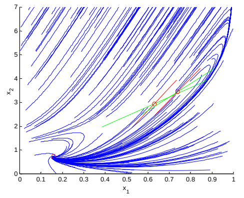
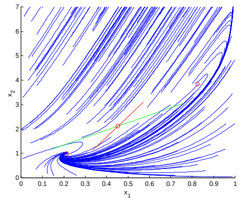
110

Interface EMSO-AUTO

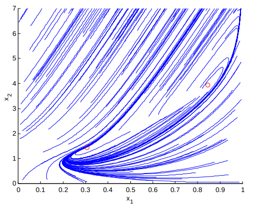
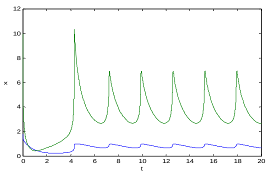
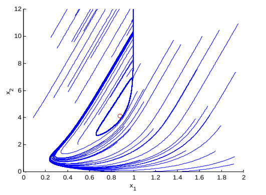
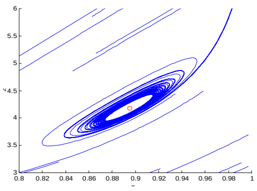
Parameter *p*	**Eigenvalues**	**Phase plane**
$p < 0.06361$	Real negatives eigenvalues – **stable node** $p = 0.05$ $\lambda = [-1.13, -2.06]$	
$p = 0.06361$	Repeated real negatives eigenvalues – **stable node (star)** $\lambda = [-1.4372, -1.4372]$	
$0.06361 < p < 0.0889$	Complex eigenvalues with negative real part - **stable focus** $p = 0.085$ $\lambda = -1.095 \pm 0.565\, i$	

111

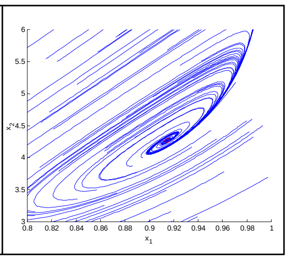
Interface EMSO-AUTO

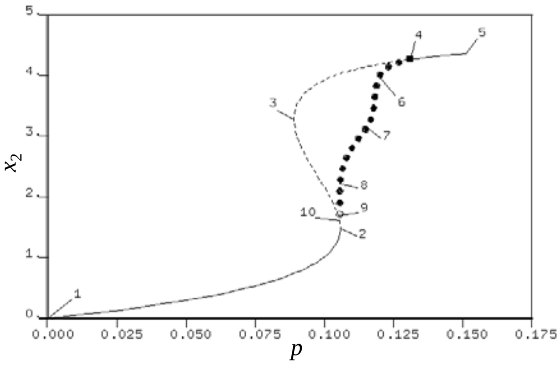
p = 0,0889 unstable node gives rise to two points: unstable node and saddle point $p > 0.0889$	**Turning point (fold)**: One stable solution (**focus**) and other unstable (**node**). (point 3 in figure below) $\lambda = -1.009 \pm 0.605\, i$ $\lambda = [0,\ 3.432]$	
$0.0889 < p < 0.0933$	One stable solution (**focus**) and two unstable (**saddle** and **node**) $p = 0.09$ $\lambda = -0.982 \pm 0.614\, i$ $\lambda = [-0.213,\ 3.332]$ $\lambda = [0.364,\ 3.151]$	
$0.0933 < p < 0.10574$ at $p = 0.105738931$ the first point goes from stable focus to stable node: $\lambda = [-0.055,\ -0.046]$	One stable solution (**focus**) and two unstable (**saddle** and **focus**) $p = 0.10$ $\lambda = -0.652 \pm 0.651\, i$ $\lambda = [-0.439,\ 1.953]$ $\lambda = 1.431 \pm 1.851\, i$	

Interface EMSO-AUTO

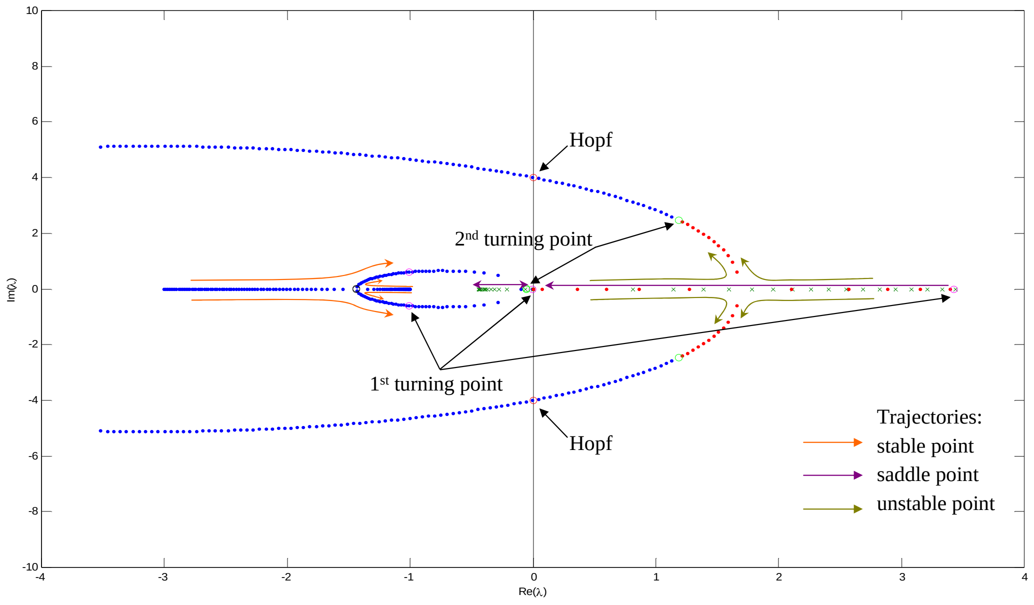
$p = 0.10574$ the stable node gives rise to two points: stable node and saddle for $p < 0.10574$	**Turning point (fold)**: One stable solution (**node**), other unstable (**focus**), and one **sable limit cycle**. (point 2 in figure below) $\lambda = [-0.097,\ 0]$ $\lambda = 1.186 \pm 2.478\, i$	
$0.10574 < p < 0.1309$ 	One unstable solution (**focus**) and one **stable limit cycle** $p = 0.12$ $\lambda = 0.528 \pm 3.487\, i$	
$p = 0.1309$	**Hopf bifurcation**: pure imaginary eigenvalues. (point 4 in figure below) $\lambda = \pm 4.008\, i$	

Interface EMSO-AUTO

$p > 0.1309$	Complex eigenvalues with negative real part - **stable focus** $p = 0.15$ $\lambda = -0.952 \pm 4.627\, i$	



Interface EMSO-AUTO



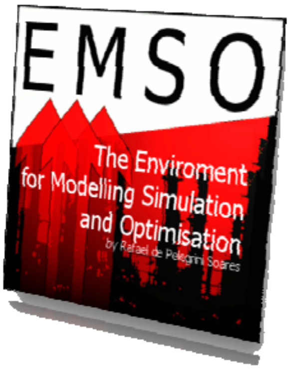
Interface EMSO-AUTO

Example: copy files auto_emso.exe and r-emso.bat (Windows) or @r-emso (linux) in “bin” folder of EMSO to the folder CSTR_auto and execute the command below in a *prompt* of commands (*shell*):

Windows: r-emso cstr_bif

Linux: ./@r-emso cstr_bif

The results are stored in file fort.7. In Linux the graphic tool PLAUT can be used to plot the results using the command @p.



Sensitivity Analysis

Objective: determine the effect of variation of parameters (p) or input variables (u) on the output variables.

Steady-state simulation: $F(x,u;p)=0$

$y = H(x,u;p)$

Sensitivity analysis

- local: $W_x = \left.\frac{\partial x_i}{\partial p_j}\right|_{\bar{x},\bar{p}}$ $\quad W_y = \left.\frac{\partial y_i}{\partial p_j}\right|_{\bar{x},\bar{p}}$
- global: bifurcation diagram, surface response (case study)

$$W_x = -\left(\frac{\partial F}{\partial x}\right)^{-1}\frac{\partial F}{\partial p} \qquad W_y = \frac{\partial H}{\partial x}W_x + \frac{\partial H}{\partial p}$$

Normalized form: $\bar{W}_y = \left.\frac{\bar{p}_j}{\bar{y}_i}\frac{\partial y_i}{\partial p_j}\right|_{\bar{x},\bar{p}}$

117

Sensitivity Analysis

Dynamic simulation: $F(t, x, \dot{x}, u; p) = 0$, $x(t_0; p) = x_0(p)$

$y = H(x, u; p)$

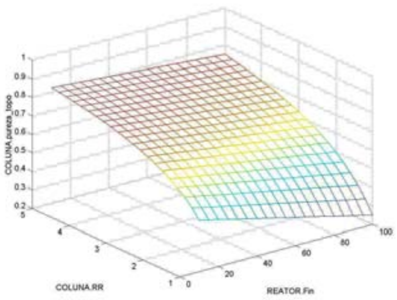
$$\frac{\partial F}{\partial \dot{x}} \dot{W}_x(t) + \frac{\partial F}{\partial x} W_x(t) + \frac{\partial F}{\partial p} = 0 \ , \ W_x(t_0) = \frac{\partial x_0}{\partial p}$$

$$W_y = \frac{\partial H}{\partial x} W_x(t) + \frac{\partial H}{\partial p}$$

where $W_x(t) = \frac{\partial x}{\partial p}$ $\dot{W}_x(t) = \frac{dW_x}{dt}$

Sensitivity Analysis

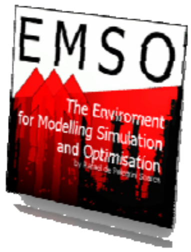
```
CaseStudy caseFlash2 as flashSteady_Test

    VARY

    s1.Composition(1) = [ 0.2379, 0.3 ,  0.35];
    s1.F = [496.3, 496.31]*'kmol/h';

    RESPONSE

    fl.OutletV.z(1);
    fl.OutletV.F 'kmol/h';

end
```




```
CaseStudy CS_Batch as BatchProcess

    VARY

    Q = [505e3: 100: 506e3] * 'W';

    RESPONSE

    T;

end
```

Sensitivity Analysis

```
Sensitivity Sense_Flash as flashSteady_Test
    VARY
    s1.F 'kmol/h';
    RESPONSE
    fl.OutletV.F 'kmol/h';
end
```




```
Sensitivity Sense4 as Test_CS1
    VARY
    b0;
    b1;
    RESPONSE
    X;
    Y;
end
```

	b0	b1
X	-0.1836734619	-0.472303207
Y	0.857142822	2.204081633
Status	OK!	OK!

120

Interface EMSO-MATLAB

- Integrate a model written in EMSO
  - Receiving input data from Matlab
  - Sending output data to Matlab (discrete mode)
  - Sending time derivatives to Matlab (continuous mode)

(Similar procedure exists with SCILAB)

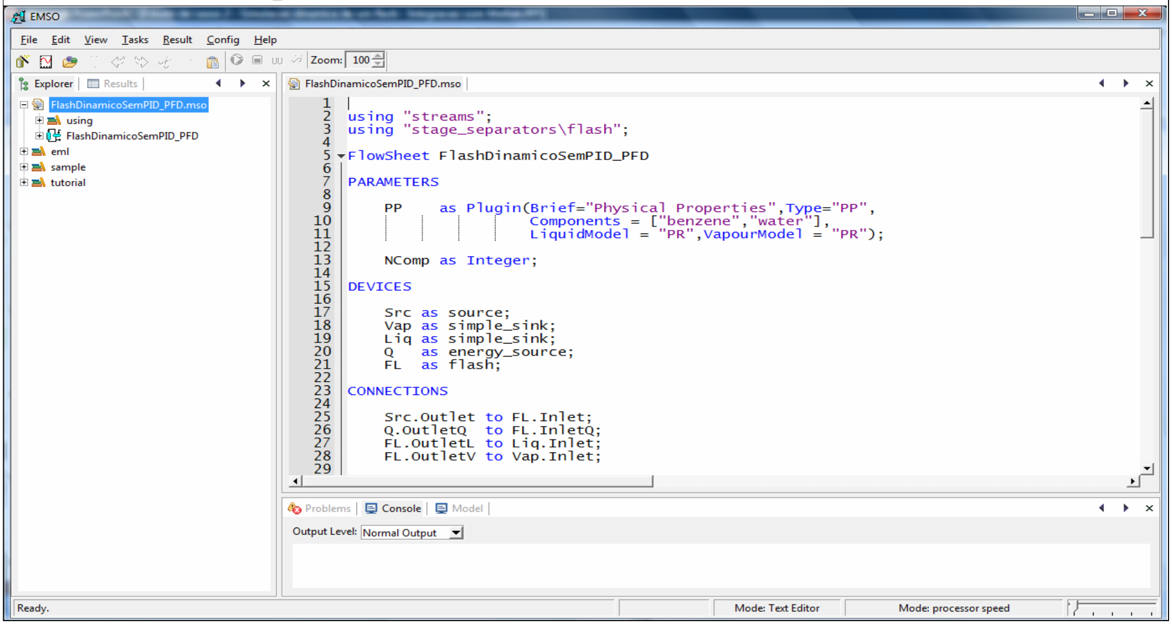
Integration Procedure

- Build a process model in EMSO
- Define input variables to be read from Matlab
  - must be specified variables in EMSO
- Define output variables to be send to Matlab
- Configure the Interface EMSO-Matlab
- Build the system model in simulink
  - Using S-function (discrete or continuous)
- Write additional calculation in Matlab
- Run the simulation from Matlab

122

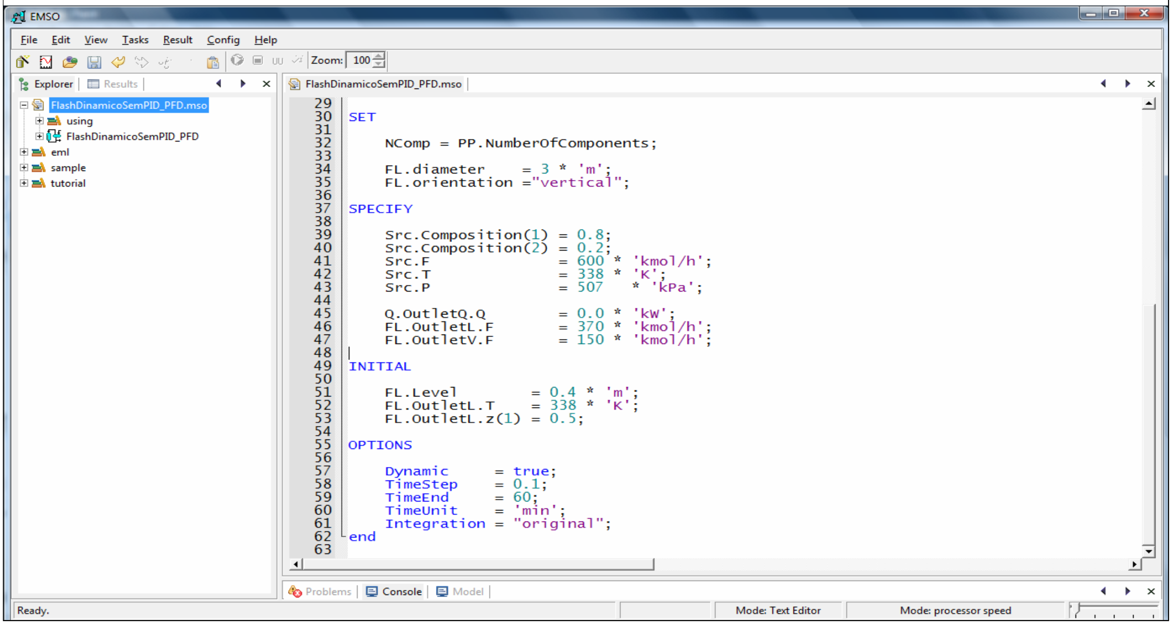
Interface EMSO-MATLAB

- Example: FlashDinamicoSemPID_PFD.mso



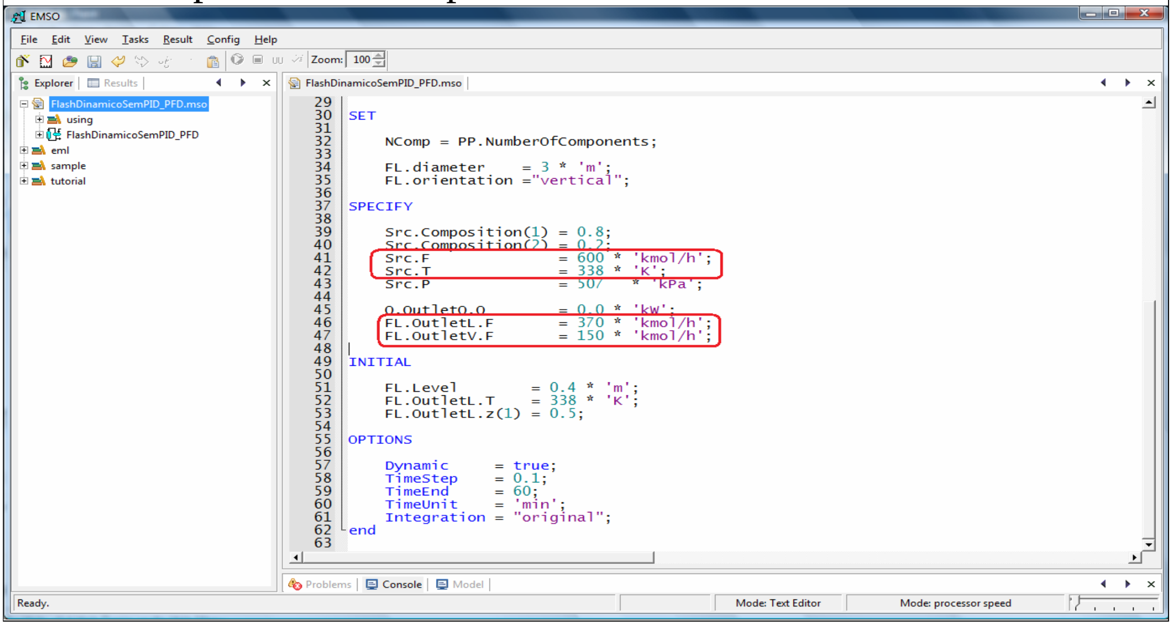
Interface EMSO-MATLAB

- Build the Model – cont.



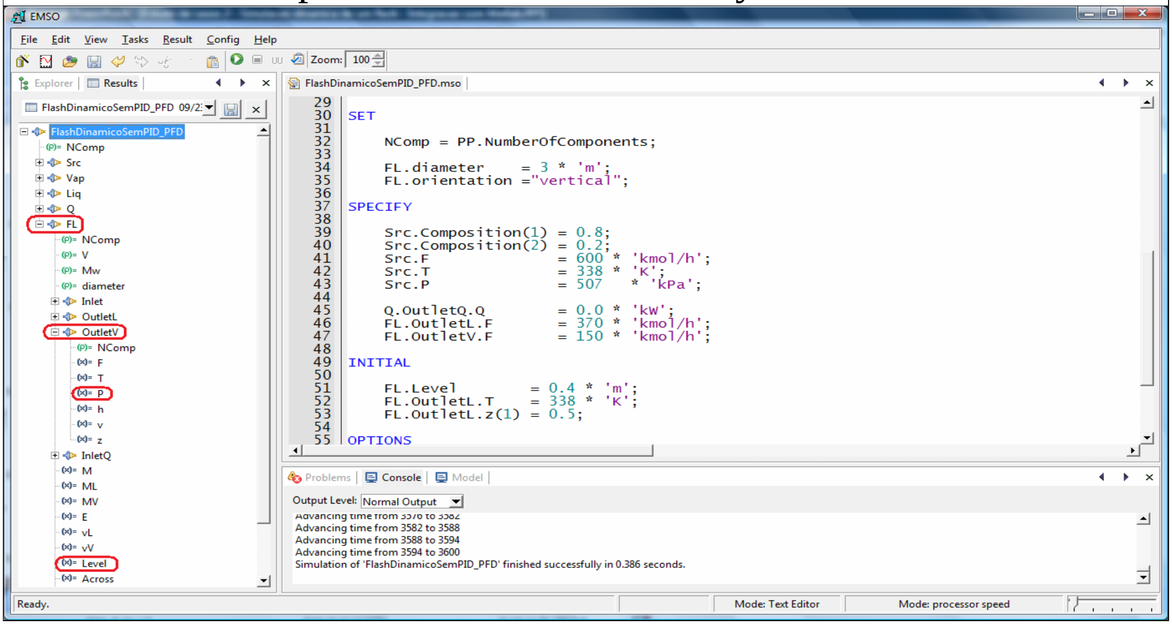
Interface EMSO-MATLAB

- Input variables: specifications in EMSO



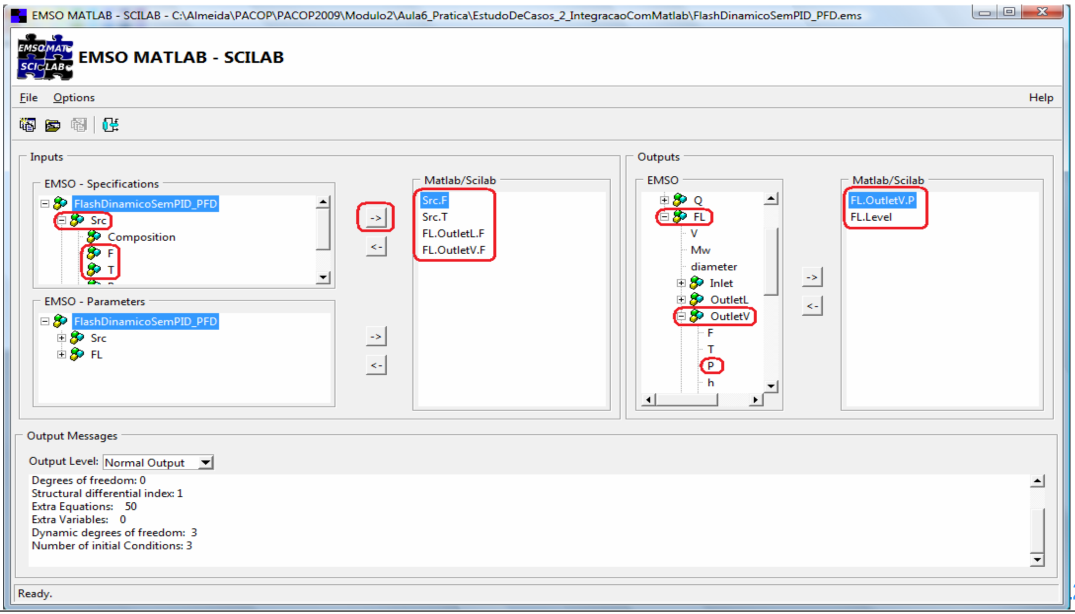
Interface EMSO-MATLAB

- Output variables: calculated by EMSO



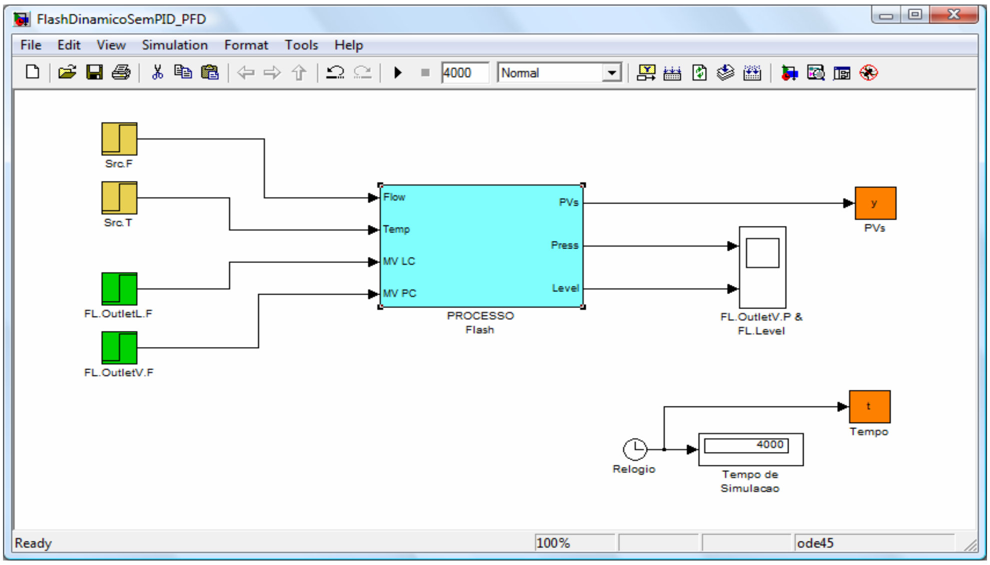
Interface EMSO-MATLAB

- Interface configuration – EMSO-Matlab



27

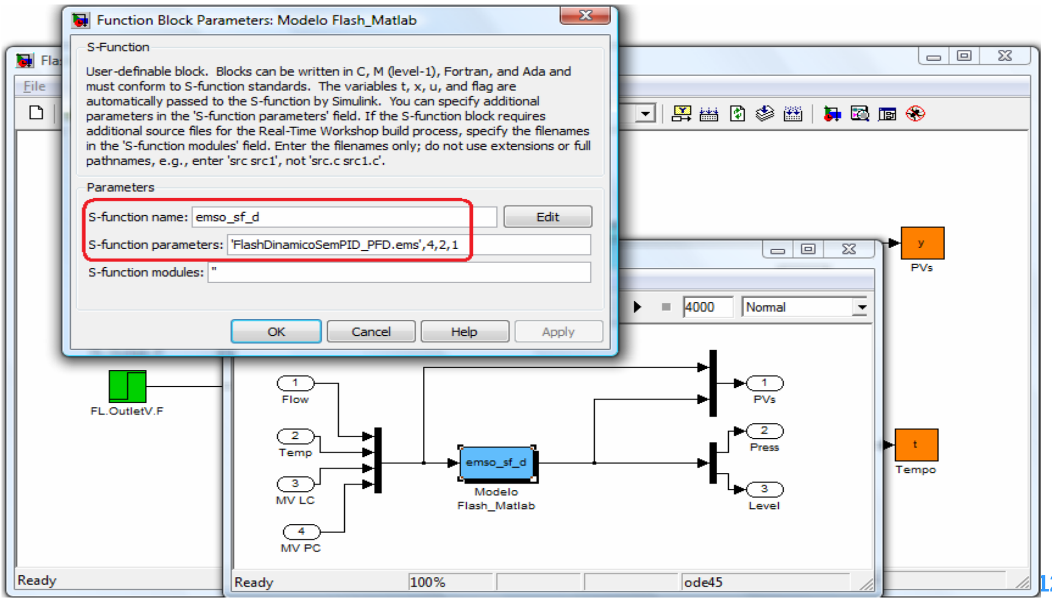
- Build System in Simulink – without PID



128

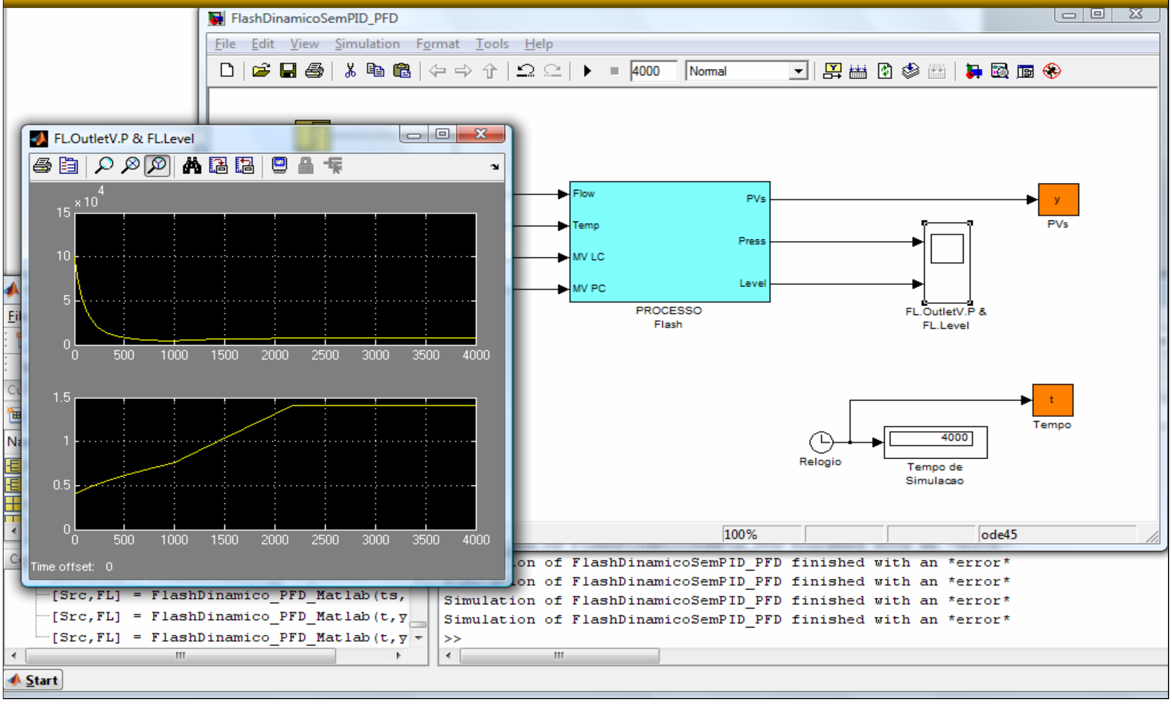
Interface EMSO-MATLAB

- Configuring size of i/o ports



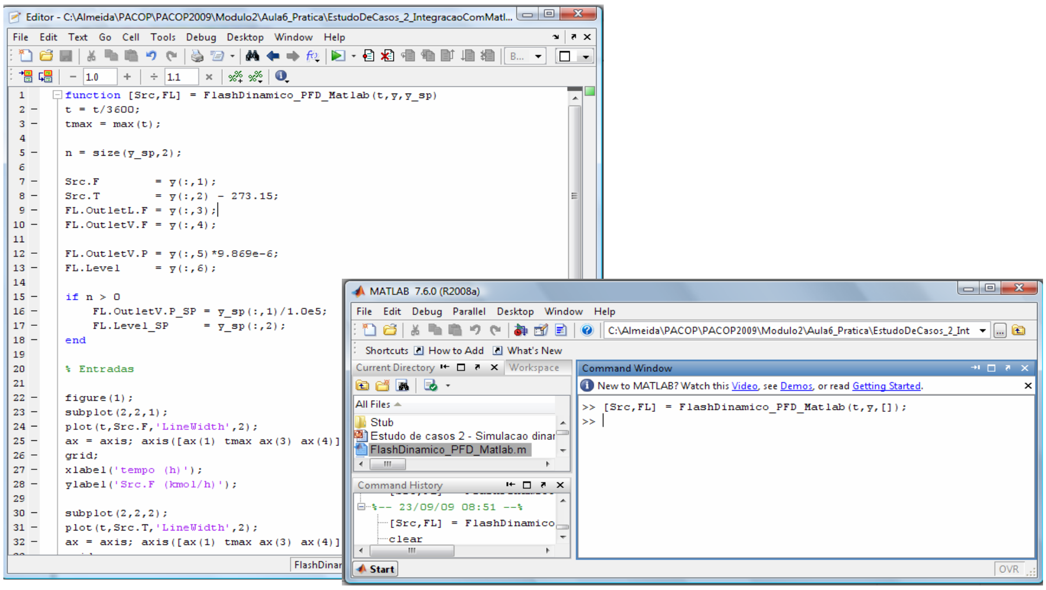
129

Interface EMSO-MATLAB



Interface EMSO-MATLAB

- Executing script in Matlab

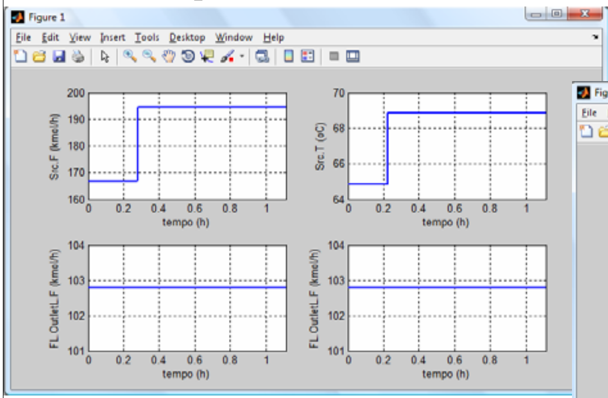


131

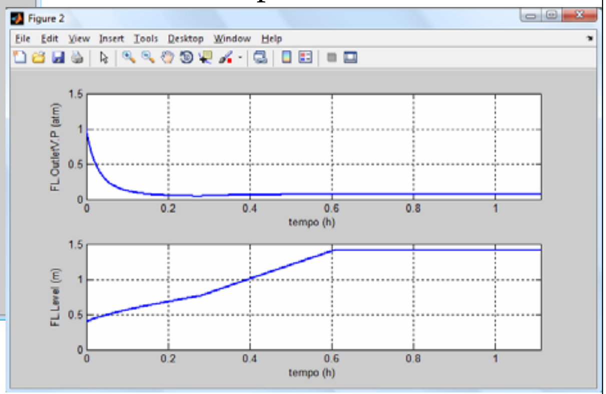
Interface EMSO-MATLAB

- Visualizing Results

Input variables



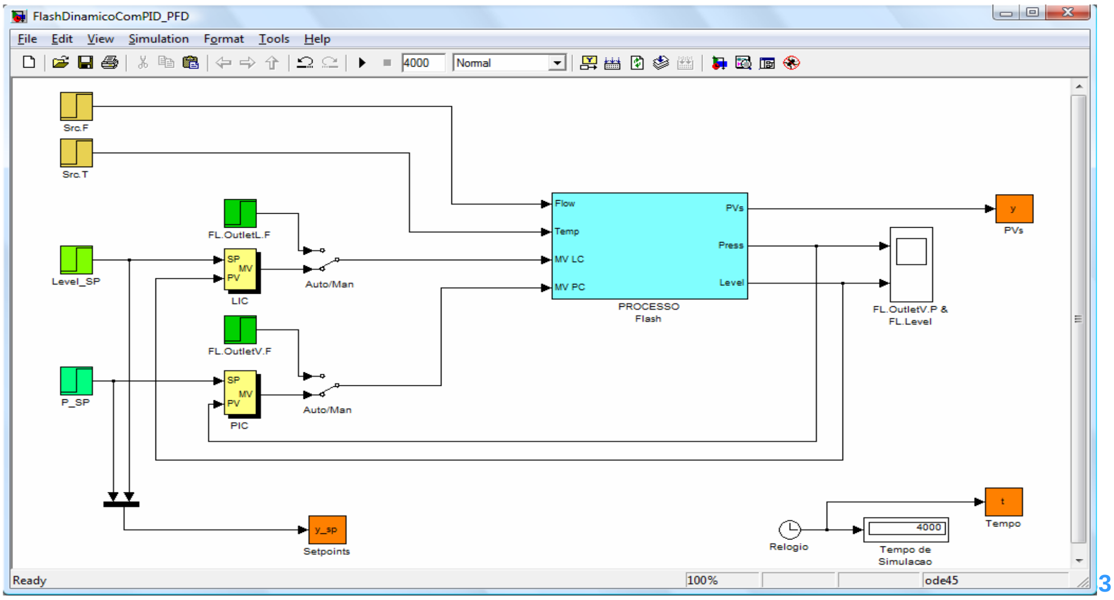
Output variables



132

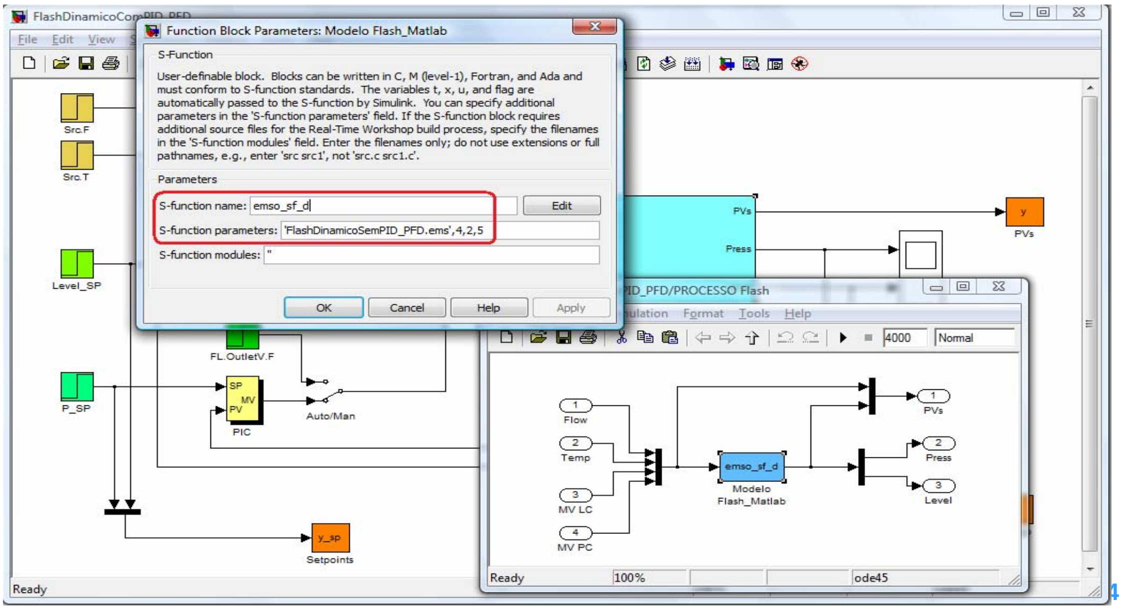
Interface EMSO-MATLAB

- Build System in Simulink – with PID

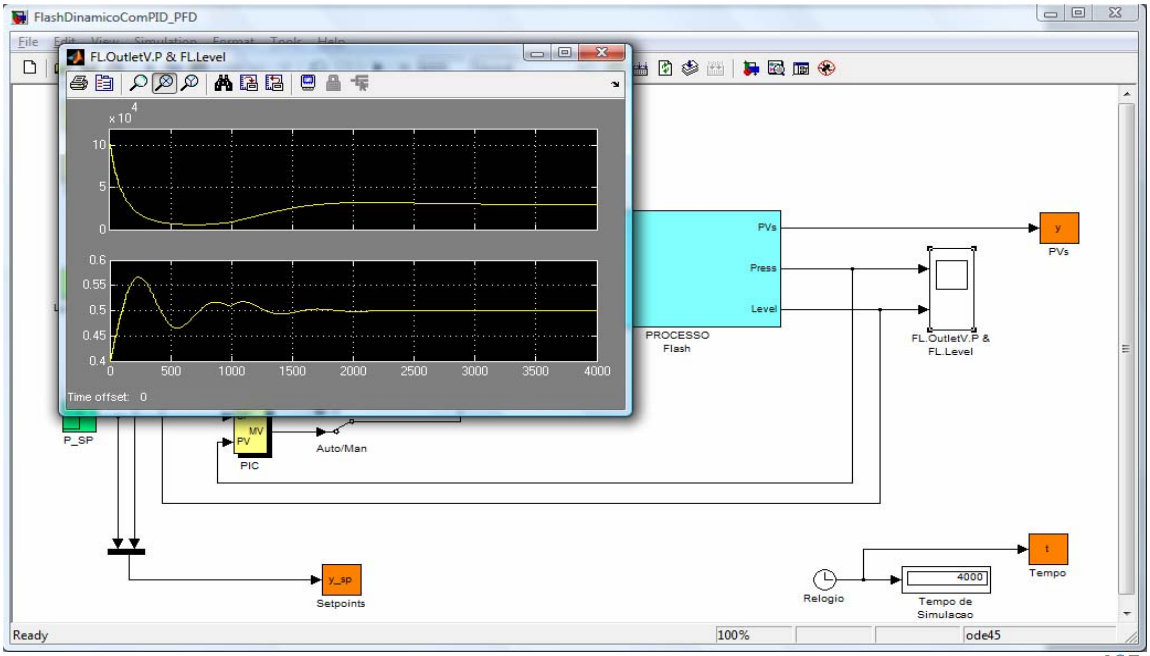


Interface EMSO-MATLAB

- Configuring size of i/o ports



Interface EMSO-MATLAB

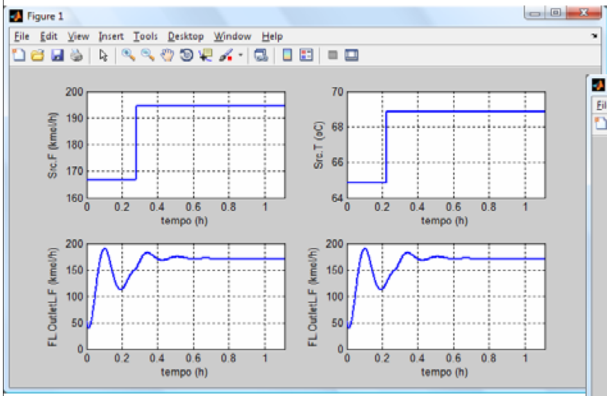


135

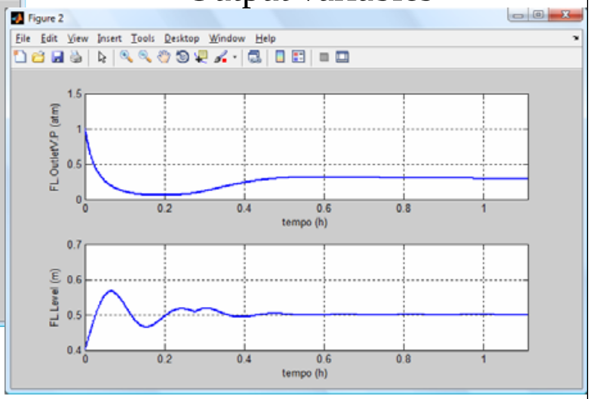
Interface EMSO-MATLAB

- Visualizing Results

Input variables



Output variables



5. Debugging Techniques

- Questions to be answered to assist the user of a CAPE tool - debugging:
  - For an under-constrained model which variables can be fixed or specified?
  - For an over-constrained model which equations should be removed?
  - For dynamic simulations, which variables can be supplied as initial conditions?
  - How to report the inconsistencies making it easy to fix?

- In other words, debugging methods need to go beyond degrees of freedom and the currently available index analysis methods

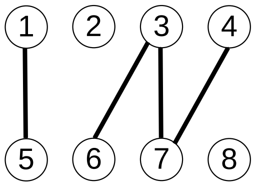
Debugging Techniques
– Current Status –

- Static models - Nonlinear Algebraic (NLA) systems:
  - Several structural analysis methods available on the literature
  - Most EO tools implement a degrees of freedom (DoF) and structural solvability analysis but user assistance is very limited when ill-posed models are found
- Dynamic models - Differential Algebraic Equation (DAE) systems:
  - Currently available methods are limited to index and dynamic degrees of freedom (DDoF) analysis
  - The well-known EO commercial tools have a high-index check which can fail even for some simple low-index problems

Debugging Techniques
– Bipartite Graphs –

✓ Bipartite graphs can be used to solve combinatorial problems:

- Tasks to machines
- Classes to rooms
- Equations to variables



- Bipartite graph $G(V = V_e \cup V_v , E)$ have two independent sets of vertices
- Vertices in the same partition must not be adjacent
- We can have *alternating* and *augmenting paths*

Matching {{1,5}, {3,7}} w/ alternating path

Matching {{1,5}, {3,6}, {4,7}} w/ augmenting path

139

Graph for variable-equation relationship

$f_1(x_1) = 0$

$f_2(x_1, x_2) = 0$

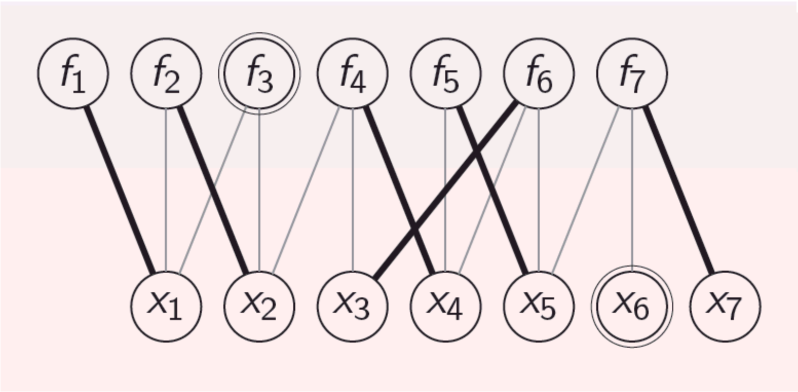
$f_3(x_1, x_2) = 0$

$f_4(x_2, x_3, x_4) = 0$

$f_5(x_4, x_5) = 0$

$f_6(x_3, x_4, x_5) = 0$

$f_7(x_5, x_6, x_7) = 0$

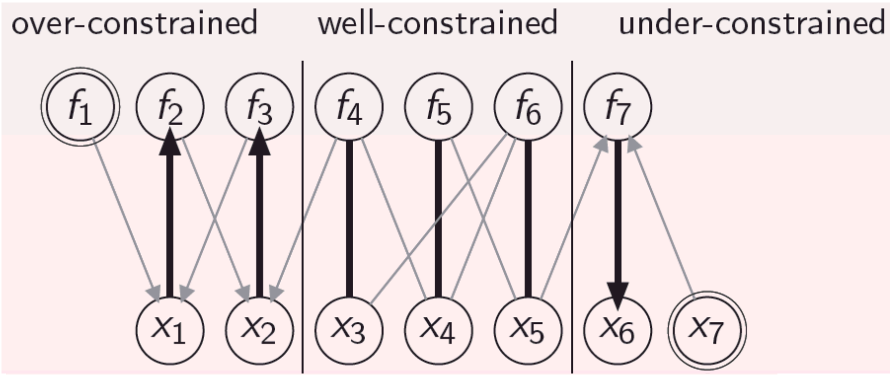


variables values
or equations forms
are irrelevant

Maximum Matching
Multiple Solutions

Debugging Nonlinear Problems

- Discover if there are over or under-constrained partitions
- Start from unconnected vertices and walk in alternating paths



Dulmage and Mendelsohn (DM) decomposition

Debugging Techniques
– Differential-Algebraic Equations –

A Simple Example

$$x_1' - x_2' = a(t)$$
$$x_2 = b(t)$$

- ✓ Only two differential variables
- ✓ Index-1 system
- ✓ Requires only one initial condition
- ✓ Initial condition must be x_1
- ✓ x_1 is the only state of the model

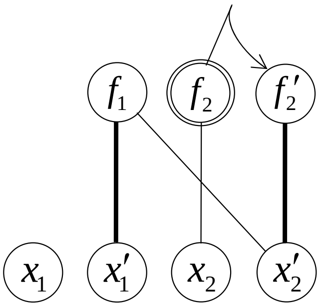
Solution:

$$x_1(t) = x_1(0) + \int_0^t a(\tau)d\tau + b(t)$$
$$x_2(t) = b(t)$$

142

Classic Algorithm

$$x_1' - x_2' = a(t)$$
$$x_2 = b(t)$$



- Who are the states?
- Which variables should be specified as initial conditions?

Debugging Techniques

– gPROMS output –

$$x_1' - x_2' = a(t)$$
$$x_2 = b(t)$$

- If only one initial condition is given (which is correct):

```
Set up of simulation
All 2 variables will be monitored during this simulation!
The number of initial conditions (1) does not match the
    number of states (2)
Building mathematical problem description took 0 seconds.
```

144

Debugging Techniques
– gPROMS output –

$$x_1' - x_2' = a(t)$$
$$x_2 = b(t)$$

- If two initial condition are given (which is wrong):

```
 Performing initialization calculation at time: 0
  Variables
       Known              : 0
       Unknown            : 2
          Differential    : 2
          Algebraic       : 0
   Model equations        : 2
   Initial conditions     : 2
Checking index of differential-algebraic equations (DAEs)...
   ERROR: Your problem is a DAE system of index greater than 1.
          Your differential variables ("states") are not independent
```

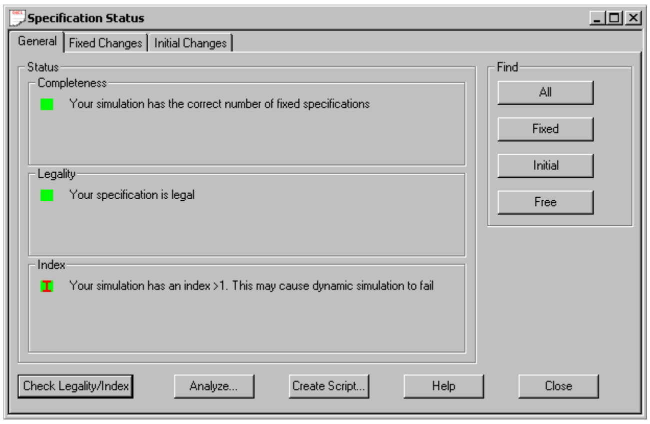
Debugging Techniques

– AspenDynamics output –

$$x_1' - x_2' = a(t)$$

$$x_2 = b(t)$$

```
Determining specification state...
specification state determined.
Preparing simulation for solution
Starting new snapshot file.
Simulation ready for solution
Simulation has 4 variables, 2 equations and 3 non-zeros
Number of equations = 2, number of states = 2
```



146

Debugging Techniques

– New Algorithm: debugging DAE system –

$DAESystems(G = (V_e, V_v, E), M)$

$M \leftarrow \emptyset$
for $v_e \in V_e$ **do**
  if not $AugmentMatching2(G = (V_e, V_v, E), M, v_e,$ **false**$)$ **then**
    mark **all** *colored* $v_k \in V_e$
    uncolour V_e
    if not $AugmentMatching2(G = (V_e, V_v, E), M, v_e,$ **true**$)$ **then**
      return false
    end if
    diff all *marked* $v_k \in V_e$
  else
    uncolour V_e
  end if
end for
return true

147

Debugging Techniques

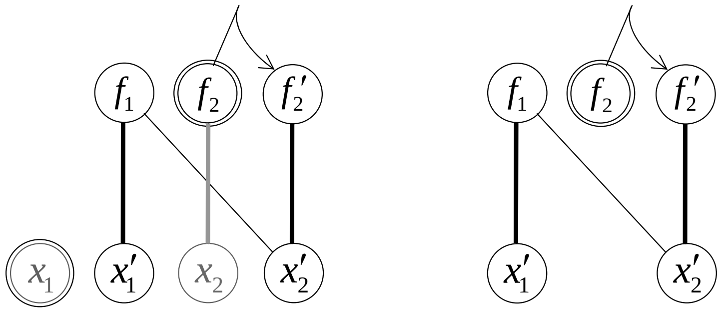
– New Algorithm: debugging DAE system –

$AugmentMatching2(G = (V_e \cup V_v, E), M, v_e, alg)$

colour v_e
if exists $\{v_e, v_v\} \in E$ **and** $\{v_e, v_v\} \notin M$ **and** v_v *is elegible* **then**
  $M \leftarrow M \cup \{v_e, v_v\}$
  return true
end if
for all $\{v_e, v_v\} \in E$ **do**
  if exists $\{v_{e2}, v_v\} \in M$ **and** v_{e2} **not** *colored* **and** v_v *is elegible* **then**
    if $AugmentMatching2(G = (V_e \cup V_v, E), M, v_{e2}, alg)$ **then**
      $M \leftarrow M \cup \{v_e, v_v\}$
      return true
    end if
  end if
end for
return false

Debugging Techniques
– Applying the New Algorithm –

$$x_1' - x_2' = a(t)$$
$$x_2 = b(t)$$



Classic Algorithm

- ✓ All equations and all *x´* are connected when it finishes
- ✓ Free variable nodes are the real states
- ✓ DM decomposition can be applied to the final matching
- ✓ Singularities are detected (classic algorithm runs indefinitely)

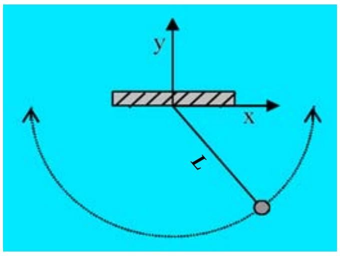
Debugging Techniques
– EMSO output –

$$x_1' - x_2' = a(t)$$
$$x_2 = b(t)$$

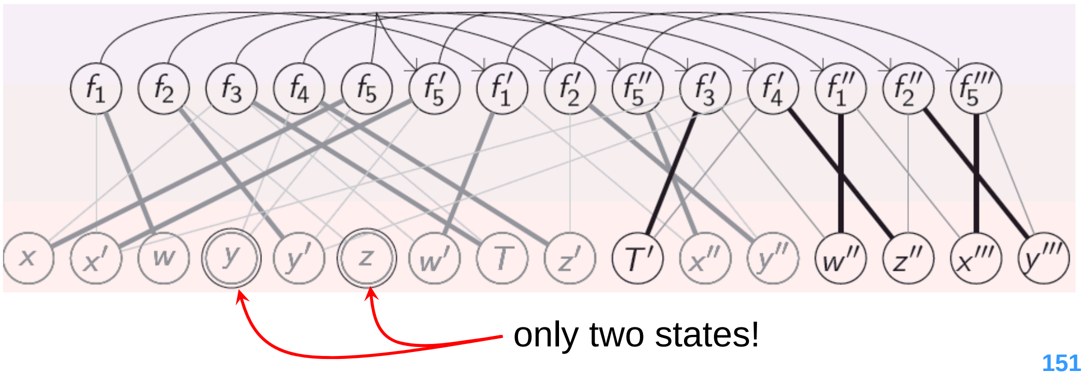
- If only one initial condition is given (which is correct):

```
Number of variables:  2
Number of equations:  2
Number of specifications:  0
Degrees of freedom: 0
Structural differential index: 1
Extra Equations:    1
Extra Variables:    0
Dynamic degrees of freedom (states):  1
Number of initial Conditions: 1
```

Debugging Techniques
– Applying the New Algorithm: high-index –

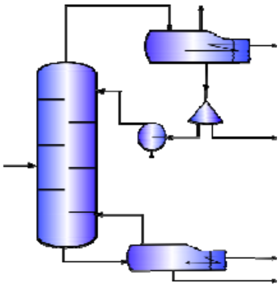


$$x' = w \quad (1)$$
$$y' = z \quad (2)$$
$$w' = T.x \quad (3)$$
$$z' = T.y - g \quad (4)$$
$$x^2 + y^2 = L^2 \quad (5)$$



Debugging Techniques

– Applying the New Algorithm: performance –



❖ Dynamic model of a distillation column for the separation of isobutane from a mixture of 13 compounds

N. Trays	N. Variables	Time* (s)	Time $/N^2$ $(s \cdot 10^9)$
20	2157	0.04	9.46
40	3877	0.14	9.58
80	7317	0.52	9.79

* Pentium M 1.7 GHz PC with 2 MB of cache memory, Ubuntu Linux 6.06

Other CAPE tools

Optimization (NLP)

flash_opt.mso

```
Optimization FlashOpt1 as FlashSteadyTest
    MAXIMIZE
    leves;

    FREE
    fl.OutletL.T;
    #fl.OutletL.P;

    EQUATIONS
    fl.OutletL.T < 320 * 'K';
    fl.OutletL.T > 300 * 'K';

    OPTIONS
    Dynamic = false;
    NLPSolveNLA = false;
    NLPSolver(#File = "complex",
              #File = "optpp_emso",
              File = "ipopt_emso",
              RelativeAccuracy = 1e-6);
end
```

sample_optimization.mso

```
Optimization hs71
    VARIABLES
    x1 as Real(Default=2, Lower=1, Upper=5);
    x2 as Real(Default=5, Lower=1, Upper=5);
    x3 as Real(Default=5, Lower=1, Upper=5);
    x4 as Real(Default=1, Lower=1, Upper=5);

    MINIMIZE
    x1*x4*(x1+x2+x3) + x3;

    EQUATIONS
    x1*x2*x3*x4 > 25;

    x1*x1 + x2*x2 + x3*x3 + x4*x4 = 40;

    OPTIONS
    NLPSolver(#File = "ipopt_emso"
              #File = "complex"
              File = "optpp_emso"
              );
    Dynamic = false;
end
```

Optimization (MINLP)

sample_minlp.mso

```
Optimization minlp1
	VARIABLES
	x0 as Integer(Default=0, Lower=0, Upper=1);
	x1 as Real(Default=0, Lower=0, Upper=1.0e10);
	x2 as Real(Default=0, Lower=0, Upper=1.0e10);
	x3 as Integer(Default=0, Lower=0, Upper=5);


	MINIMIZE
	-1*(x0 + x1 + x2);

	EQUATIONS

	(x1 - 0.5)^2 + (x2 - 0.5)^2 <= 0.25;
	x0 - x1 <= 0;
	x0 + x2 + x3 <= 2;

	OPTIONS
	Dynamic = false;
	NLPSolveNLA = true;

	NLPSolver(File = "minlp_emso", derivative_test = "second-order",
	print_level = 5);

end
```

Parameter Estimation

Bio.mso

```
Estimation Biop_NE_Estt5 as Biop_NE_process5t
    ESTIMATE
    # PARAMETER   START      LOWER     UPPER     UNIT
    Kss           0.2009     0.004     7         'kg/m^3';
    Ksn           0.0446     0.005     1         'kg/m^3';
    mim           0.7979     0.1       0.8       '1/s';
    alfa          2.0293     1         5          ;
    gama          0.08502    0.05      5         '1/s';
    K1            0.4059     0.1       3         'kg/kg';
    K2            -0.00795   -1        3         '1/s';
    Yn            10.62      0.1       18         ;
    kd            0.007      0.0005    1         '1/s';

    EXPERIMENTS
    # DATA FILE     WEIGTH
    "Bio.dat"       1;

    OPTIONS
    Statistics(
        Fits=true,
        Parameters=false,
        Predictions=false
    );

    NLPSolver(
        MaxIterations = 1000,
        File = "complex"
        #File = "ipopt_emso"
    );
    Dynamic = true;
end
```

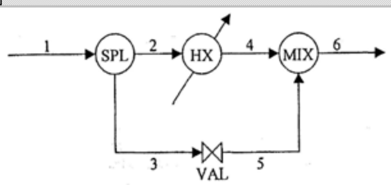
pv_est.mso

```
Estimation PV_Est as PV_Flow

    ESTIMATE
    # PAR    START    LOWER    UPPER    UNIT
    A        1.5      -3       10;
    B        1000     800      3000     'K';
    C        50       20       200      'K';

    EXPERIMENTS
    # FILE                      WEIGTH
    "pv_est.dat"        1;

    OPTIONS
    NumJac = false;

    NLPSolver(
        File = "ipopt_emso"
        #File = "complex"
    );
    Dynamic = false;
end
```

Data Reconciliation

heatEx.mso

```
FlowSheet HeatEx_Flow
    VARIABLES
    x1 as Real (Default=50.00, Lower=0.00, Upper=150);
    x2 as Real (Default=50.00, Lower=0.00, Upper=150);
    x3 as Real (Default=50.00, Lower=0.00, Upper=150);
    x4 as Real (Default=50.00, Lower=0.00, Upper=150);
    x5 as Real (Default=50.00, Lower=0.00, Upper=150);
    x6 as Real (Default=50.00, Lower=0.00, Upper=150);

    SPECIFY
    x1 = 100.91;
    x2 = 64.45;
    #x3 = 34.65;

    EQUATIONS
    x1 - x2 - x3 = 0;
    x2 - x4 = 0;
    x3 - x5 = 0;
    x4 + x5 - x6 = 0;

    OPTIONS
    Dynamic = false;
end
```



heatEx.mso

```
Reconciliation HeatEx_Rec as HeatEx_Flow

    RECONCILE
    x1; x2; x3; x4; x5; x6;
    #x1; x2; x5; x6;
    #x1; x2;
    #x1; x6;

    FREE
    x1; x2;

    EXPERIMENTS
    # FILE              WEIGTH
    "heatEx.dat"        1;
    #"heatEx_1.dat"         1;
    #"heatEx_2.dat"         1;
    #"heatEx_3.dat"         1;
    #"heatEx_GE.dat"        1;

    OPTIONS
    Filter = "mean";
    Significance = 0.95;

    GrossErrorTests(
        Global = true,
        Nodal = true,
        Measurements = true
    );

    NLPSolver(
        MaxIterations=1000,
        File = "complex"
        #File = "ipopt_emso"
    );
    Dynamic = false;
end
```

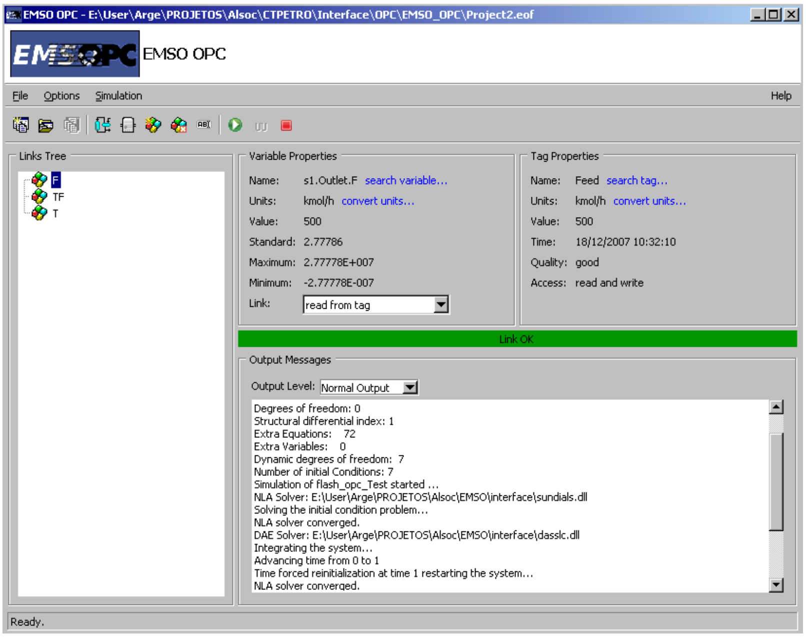
Interface EMSO-OPC

Support to the following possible applications:

- Virtual analyzer (inferences with models)
- Process monitoring
- Testing control systems
- Operator training
- State estimators
- Model updating
- Any application that needs integrating models with plant data in real time!

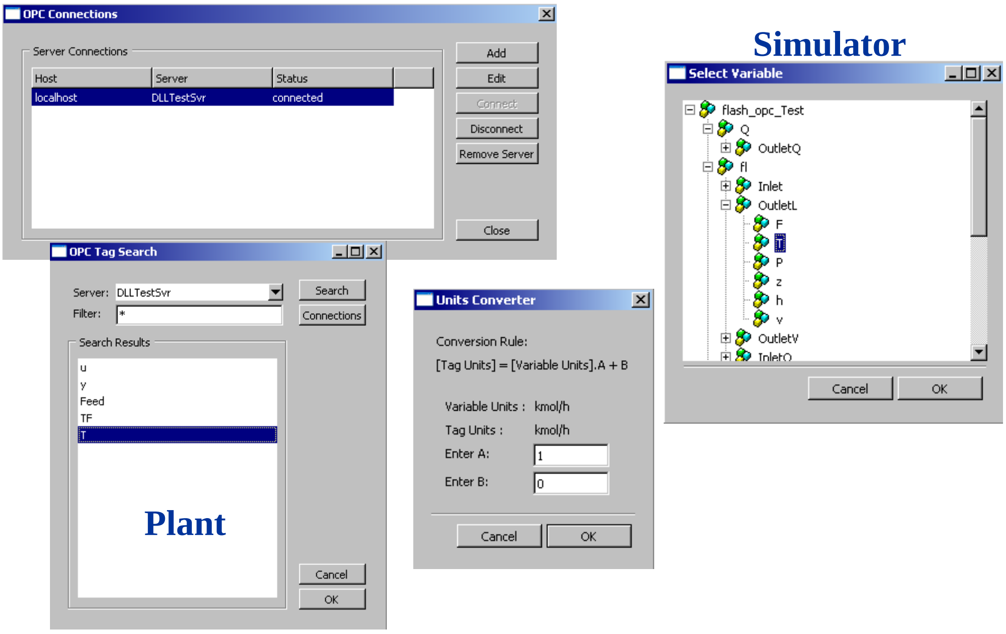
158

Interface EMSO-OPC

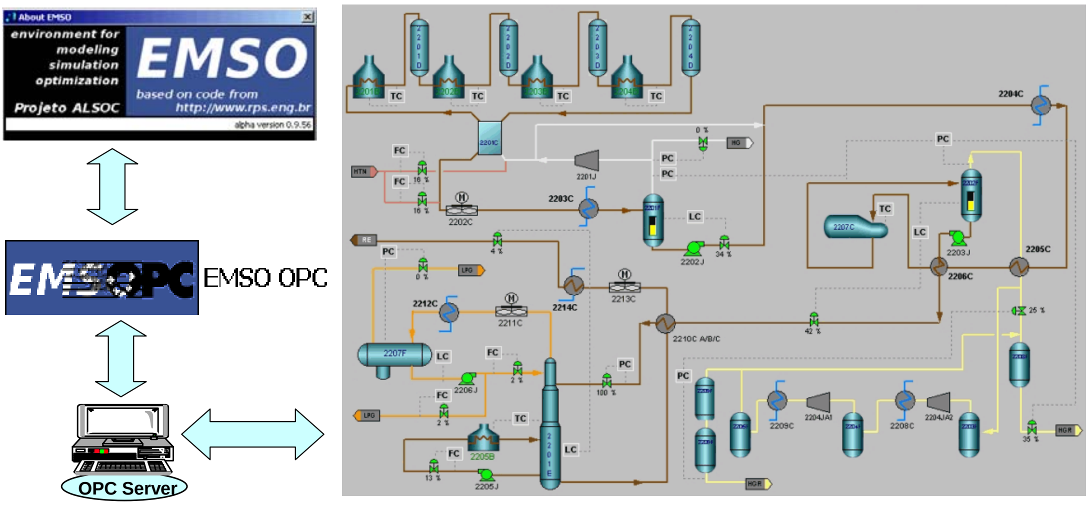


159

Interface EMSO-OPC

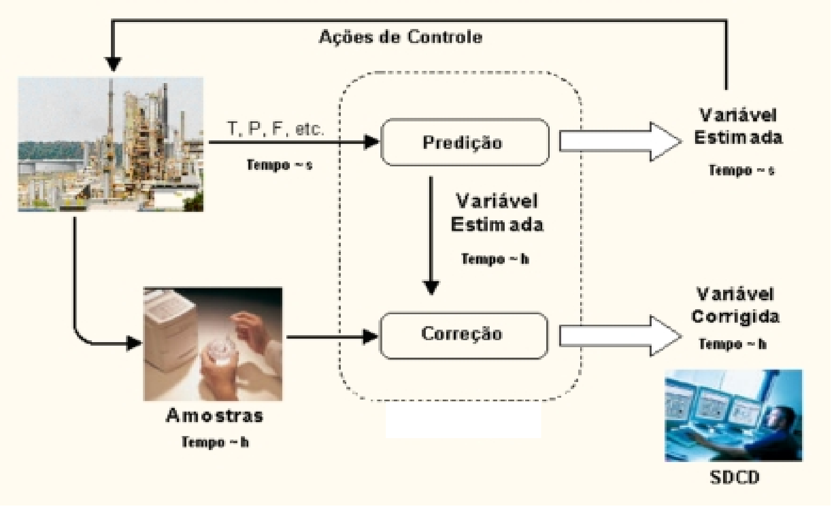


Operator Training and Process Monitoring



161

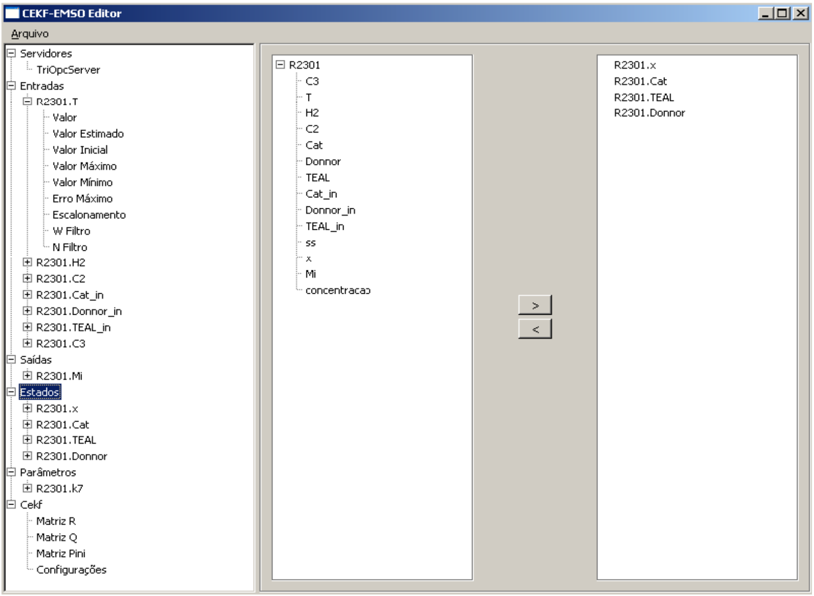
Virtual Analyzer – EMSO-CEKF



162

EMSO-CEKF

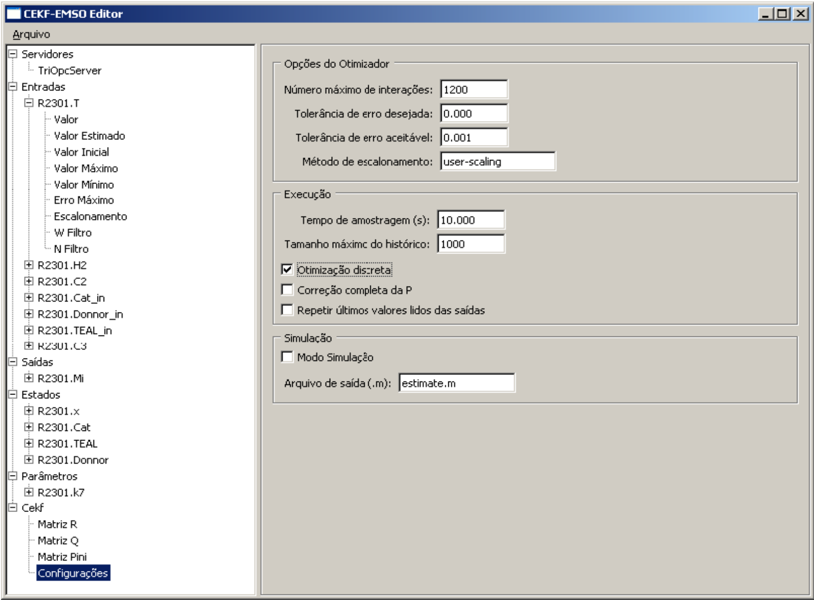
Model parameters and variables selection



163

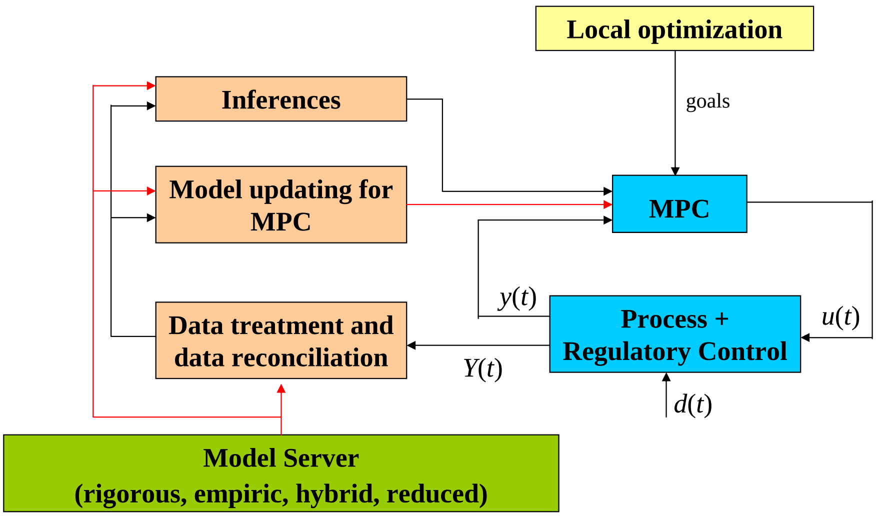
EMSO-CEKF

Parameters and variables configuration



164

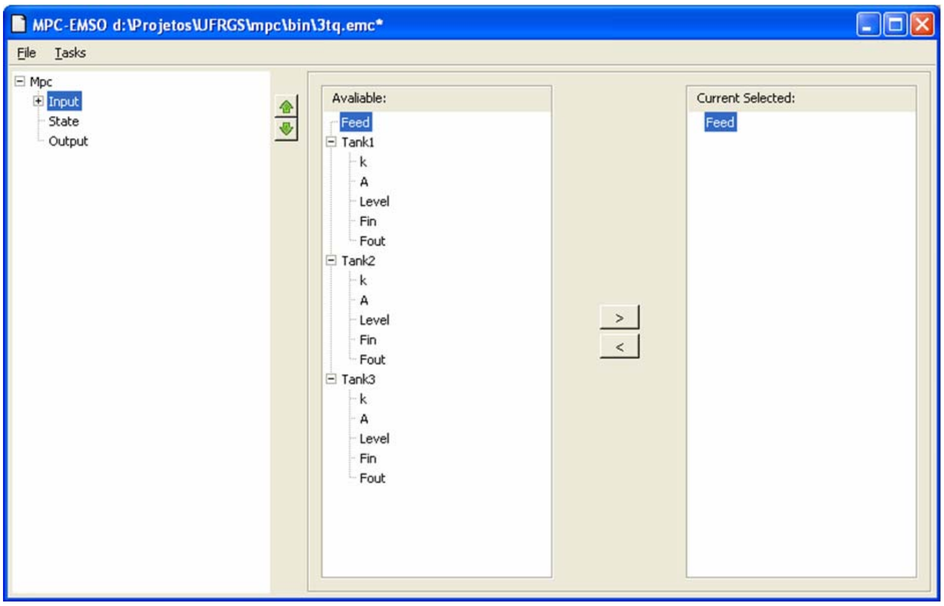
Model Generation for MPC



165

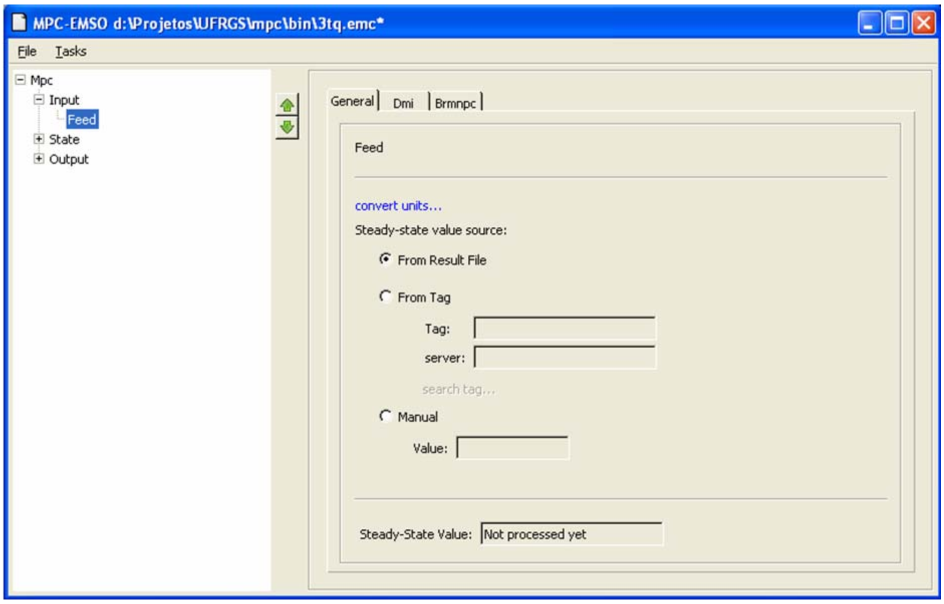
Model Generation for MPC

Variables selection



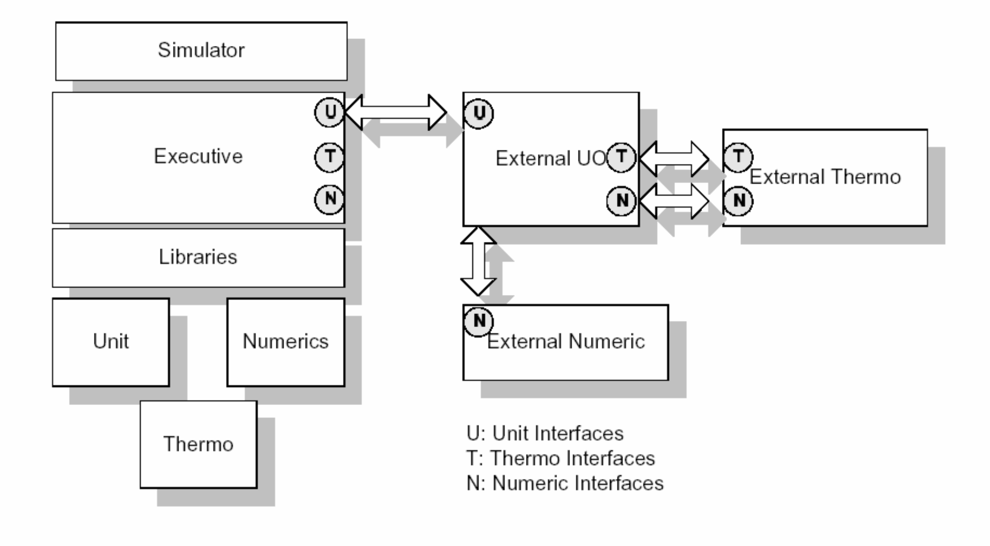
Model Generation for MPC

Variables configuration



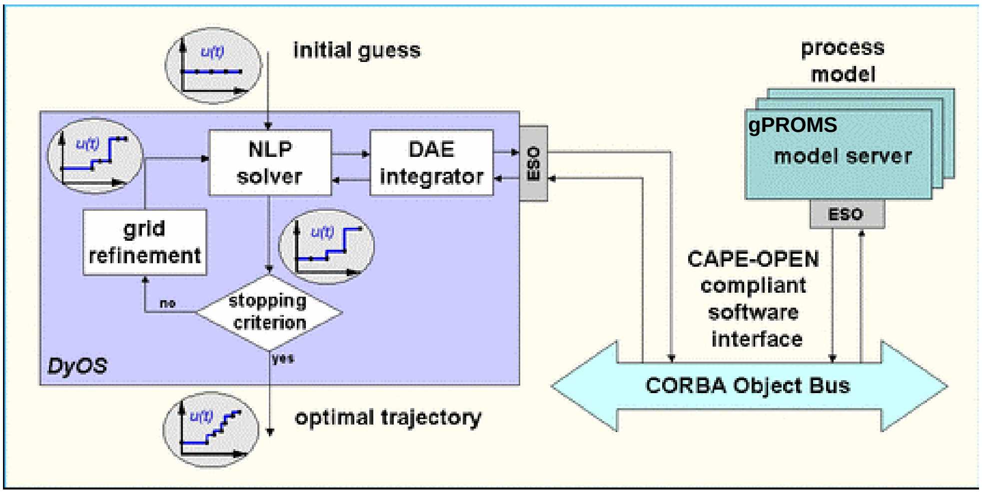
Standard Interfaces

CAPE-OPEN



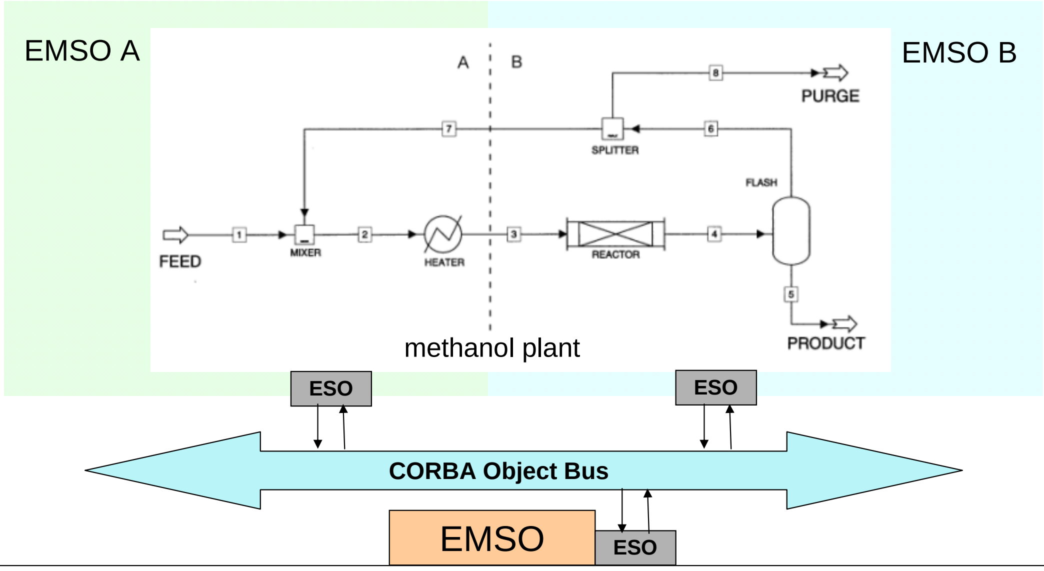
168

Example of CAPE-OPEN: DyOS (Dynamic Optimization Software) - Marquardt's group (2000)



CAPE OPEN

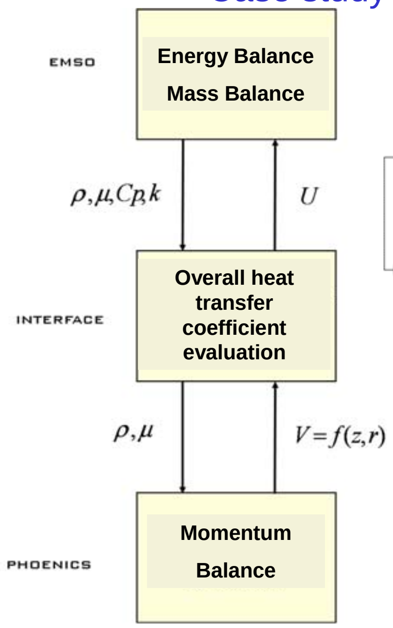
Another example of CAPE-OPEN: EMSO (Environment for Modeling, Simulation and Optimization) - Soares and Secchi (2004)

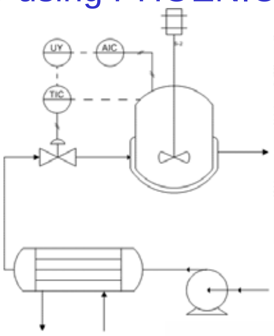


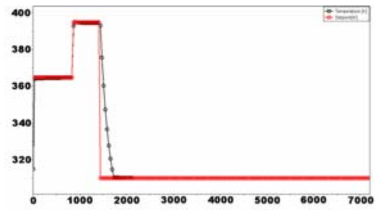
170

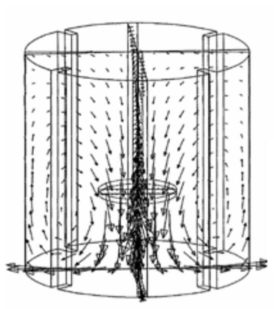
Interface EMSO-CFD

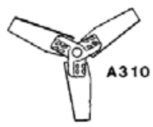
Case study using PHOENICS and FLUENT











Final Remarks

- The concepts of inheritance and aggregation of the object-oriented modeling paradigm make possible to refine, reuse, and extend available models to more specialized applications, reducing considerably the modeling stage of a project
- A complete consistency analysis of process models described by differential-algebraic equation systems is a very important mechanism to aid the development of new models, specially for large-scale systems
- The integration of a process simulator with model-based tools, such as AUTO and Simulink/Scicos, allows us to carry out more complex analysis of rigorous models and complete flowsheet simulations

172

Final Remarks

Available CAPE tools ...

- Process simulators and optimizers
- System identification packages
- System analysis
- Standard communication interfaces
- Numerical solvers (NLA, NLP, MINLP, DAE, ...)
- User-friendly graphical interfaces

... that need high-tech people to use and improve them!

173

References

• Andersson, M. 1994. Object-oriented modeling and simulation of hybrid systems. Ph.D. diss., Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

• Barton, P.I. 1999. ABACUSS II. Retrieved from http://yoric.mit.edu/abacuss2/abacuss2.html

• Barton, P.I. and Pantelides, C.C., 1994, The modeling of combined discrete=continuous processes, AIChE J., 40, 966–979.

• Bogusch, R., Lohmann, B., Marquardt, W. 2001. Computer-aided process modeling with MODKIT. Comp. & Chem. Engng., 25, 963–995.

• Bogusch, R. and Marquardt, W. 1997. A formal representation of process model equations. Comp. & Chem. Engng. 21 (10) 1105-111.

• EA International and ESA. 1999. EcosimPro ver. 3.0: Getting started, users manual, modeling language (EL), modeling and simulation guide, and mathematical algorithms. Madrid, Spain: EA International.

• Eliceche, A.M., Corvalán, S.M. and Martinez, P.E. 2007. Environmental Life Cycle Impact as a Tool for Process Optimization of a Utility Plant. *Comp. & Chem. Engng*., 31, 648–656.

• Elmqvist, H. 1978. A structured model language for large continuous systems. Ph.D. diss., Lund Institute of Technology, Lund, Sweden.

• Elmqvist, H., D. Bruck, and M. Otter. 1999. Dymola: Dynamic modeling laboratory: User’s manual.Version 4.0. Lund, Sweden: Dynamic AB.

• Halim, I. and Srinivasan, R. 2011. A Knowledge-Based Simulation-Optimization Framework and System for Sustainable Process Operations. *Comp. & Chem. Engng*., 35, 92–105.

• Heijungs, R., Guinée, J., Huppes, G., Lankreijer, R.M., Ansems, A.A.M., Eggels, P.G., van Duin, R and. de Goede, H.P. 1992. Environmental Life Cycle Assessment of Products-Guide and Backgrounds. Centre of Environmental Science (CML). Leiden.

References

• Jia, X.P., Han, F.Y. and Tan, X.S. 2004. Integrated Environmental Performance Assessment of Chemical Processes. *Comp. & Chem. Engng.*, 29, 243–247.

•Modelica Association. 1996. Modelica: A unified object-oriented language for physical systems modeling: Tutorial, rationale and language specification. Retrieved from http://www.modelica.org.

• Piela, P.C. 1989. ASCEND:Anobject-oriented environment for modeling and analysis. Ph.D. diss., Engineering Design Research Center, Carnegie Mellon University, Pittsburgh, PA.

• Rao, R.M., Rengaswamy, R., Suresh, A.K. and Balaraman, K.S. 2004. Industrial Experience with Object-Oriented Modelling FCC Case Study. Chem. Engng. Res. & Des., 82 (A4) 527–552.

• Rodrigues, R., Soares, R.P. and Secchi, A.R. 2010. Teaching Chemical Reaction Engineering Using EMSO Simulator. *Computer Applications in Engineering Education*, 18 (4) 607-618.

• Salau, N.P.G., Neumann, G.A, Trierweiler, J.O. and Secchi, A.R. 2008. Dynamic Behavior and Control in an Industrial Fluidized-Bed Polymerization Reactor. *Ind. Eng. Chem. Res.*, 47, 6058–6069.

• Soares, R.P. and Secchi, A.R. 2003. EMSO: A New Environment for Modeling, Simulation and Optimization. *ESCAPE* 13, Lappeenranta, Finlândia, 947 – 952.

• Soares, R.P. and Secchi, A.R. 2005. Direct Initialisation and Solution of High-Index DAE Systems, *ESCAPE* 15, Barcelona, Spain, 157–162.

• Soares, R.P. and Secchi, A.R. 2007. Debugging Static and Dynamic Rigorous Models for Equation-oriented CAPE Tools, DYCOPS 2007, Cancún, Mexico, 2, 291–296.

•Tränkle, E, Gerstlauer, A., Zeitz, M., and Gilles, E. D. 1997.The Object-Oriented Model Definition Language MDL of the Knowledge-Based Process Modeling Tool PROMOT. In A. Sydow (Ed.), 15th IMACS World Congress, Berlin, Germany, 4, 91-96.

• Valle, E.C., Soares, R.P., Finkler, T.F. and Secchi, A.R. 2008. A New Tool Providing an Integrated Framework for Process Optimization, EngOpt 2008 - International Conference on Engineering Optimization, Rio de Janeiro, Brazil.

References

DAE Solvers:

DASSL: Petzold, L.R. (1989), http://www.enq.ufrgs.br/enqlib/numeric/numeric.html

DASSLC: Secchi, A.R. (1992), http://www.enq.ufrgs.br/enqlib/numeric/numeric.html

MEBDFI: Abdulla, T.J. and J.R. Cash (1999), http://www.netlib.org/ode/mebdfi.f

PSIDE: Lioen, W.M., J.J.B. de Swart, and W.A. van der Veen (1997), http://www.cwi.nl/cwi/projects/PSIDE/

SUNDIALS: Serban, R. et al. (2004), http://www.llnl.gov/CASC/sundials/description/description.html

... thank you for your attention!

http://www.enq.ufrgs.br/alsoc

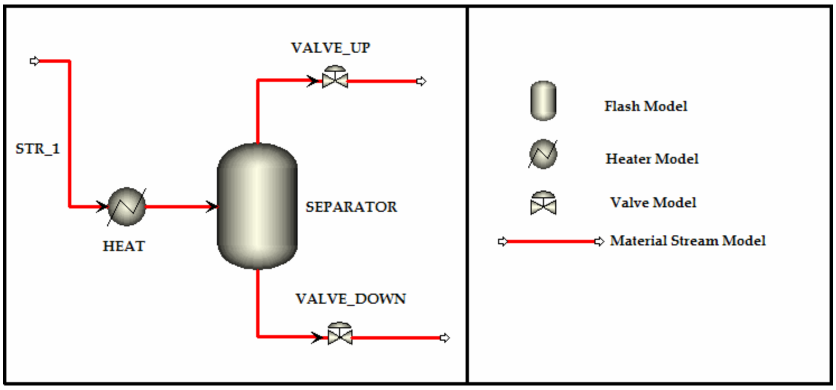
Solutions for Process Control and Optimization
COPPE | UFRJ

Process Modeling, Simulation and Control Lab

- Prof. Dr. Argimiro Resende Secchi
- Phone: +55-21-2562-8301
- E-mail: arge@peq.coppe.ufrj.br
- http://www.peq.coppe.ufrj.br/Areas/Modelagem_e_simulacao.html

Extra slides

EMSO has 3 main entities in the modeling structure



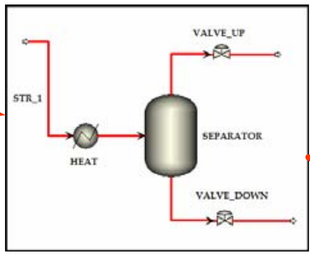
FlowSheet – process model, is composed by a set of DEVICES

DEVICES – components of a FlowSheet, an unit operation or an equipment

Model – mathematical description of a DEVICE

– Modeling Structure –

Model



FlowSheet

Model: equation-based

```
using "streams";

Model heater

 PARAMETERS
  outer PP as Plugin (Brief="Physical Properties", Type="PP");
  outer NComp as Integer (Brief="Number of Components");
  NInlet as Integer (Brief="Number of Inlet Streams");
  Kvalues as Switcher (Brief="Option for Display Phase Equilibrium K values",Val

 VARIABLES
  QDuty as power (Brief="Actual Duty");
  Vfrac as fraction (Brief="Vapor fraction Outlet Stream");
  Lfrac as fraction (Brief="Liquid fraction Outlet Stream");
  Kvalue(NComp) as Real (Brief = "Phase Equilibrium K values");
in Inlet(NInlet) as stream (Brief="Inlet Streams", PosX=0, PosY=0.4833);
out Outlet as streamPH (Brief="Outlet Stream", PosX=1, PosY=0.4783);

 EQUATIONS

 "Flow"
  Outlet.F = sum(Inlet.F);

 for j in [1 : NComp]

 "Composition"
  Outlet.F*Outlet.z(j) = sum(Inlet.F*Inlet.z(j));

 end

 "Vapor fraction Outlet Stream"
  Vfrac = Outlet.v;

 "Liquid fraction Outlet Stream"
  Lfrac = 1-Vfrac;
```

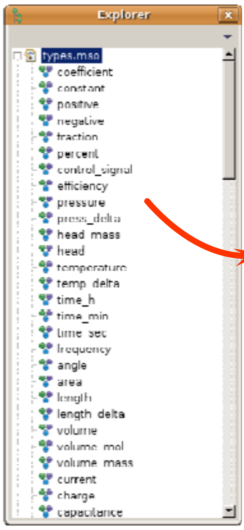
FlowSheet: component-based

```
using "heater";
using "flash";
using "valve";
using "stream";

FlowSheet Process

 DEVICES
  HEAT       as heater  (Brief="Heater Componente");
  SEPARATOR  as flash   (Brief="Flash Componente");
  VALVE_UP   as valve   (Brief="Valve upstream Componente");
  VALVE_DOWN as valve   (Brief="Valve downstream Componente");
  STR_1      as streamPH (Brief="Feed stream Componente");

 CONNECTIONS

  STR_1              to HEAT.Inlet;
  HEAT.Outlet        to SEPARATOR.Inlet;
  SEPARATOR.OutletV  to VALVE_UP.Inlet;
  SEPARATOR.OutletL  to VALVE_DOWN.Inlet;
```

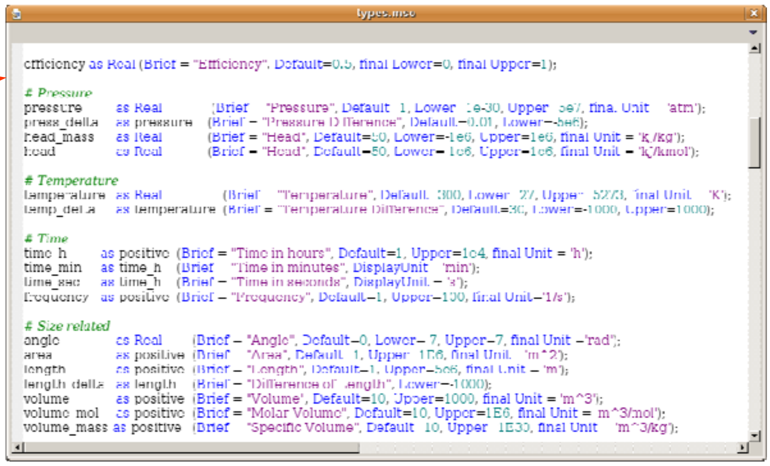
180

EMSO Tutorial

– Object-Oriented Variable Types –



Parameters and variables are declared within their valid domains and units using types created based on the built-in types: Real, Integer, Switcher, Plugin



EMSO Tutorial
– Model Components –

Including sub-models and types

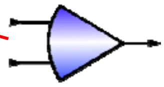
Automatic model documentation

Basic sections to create a math. model

Input and output connections

mixer.mso

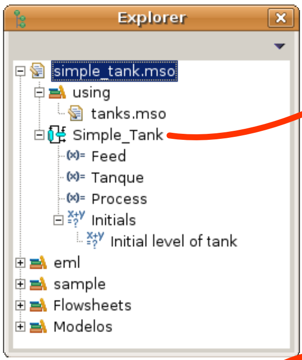
```
using "streams";

Model mixer
	ATTRIBUTES
	Pallete 	= true;
	Icon 		= "icon/mixer";
	Brief 		= "Model of a mixer";
	Info 		=
"== Assumptions ==
* static
* adiabatic

== Specify ==
* the inlet streams";

	PARAMETERS
outer NComp as Integer (Brief = "Number of chemical components", Lower = 1);
	Ninlet as Integer (Brief = "Number of Inlet Streams", Lower = 1, Default = 2);

	VARIABLES
in	Inlet(Ninlet) as stream (Brief = "Inlet streams", PosX=0, PosY=0.5, Symbol="_{in}");
out	Outlet 		as streamPH (Brief = "Outlet stream", PosX=1, PosY=0.5, Symbol="_{out}");

	EQUATIONS

	"Flow"
	Outlet.F = sum(Inlet.F);

	for i in [1:NComp]

		"Composition"
		Outlet.F*Outlet.z(i) = sum(Inlet.F*Inlet.z(i));
	end

	"Energy Balance"
	Outlet.F*Outlet.h = sum(Inlet.F*Inlet.h);

	"Pressure"
	Outlet.P = min(Inlet.P);
end
```



Symbol of variable in LaTeX command for documentation

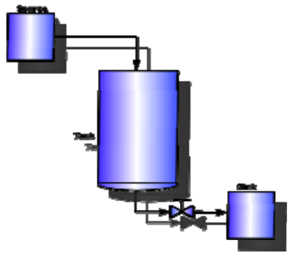
Port location to draw a flowsheet connection

EMSO Tutorial
– FlowSheet Components –



simple_tank.mso

```
using "tanks";

FlowSheet Simple_Tank

DEVICES

   Feed     as source          (Brief = "Feed stream");
   Tanque   as tank_circular   (Brief = "Tank");
   Process  as sink            (Brief = "Sink stream");

CONNECTIONS

   Feed.Outlet     to Tanque.Inlet;
   Tanque.Outlet   to Process.Inlet;

SET
   Tanque.k    8 * 'm^2.5/h';
   Tanque.Dh   2 * 'm';

SPECIFY
"Feed flow"
   Feed.Outlet.Fvol  = 10 *'m^3/h';

"Feed temperature de entrada"
   Feed.Outlet.T  = 300 *'K';

"Feed pressure"
   Feed.Outlet.P  = 1 *'atm';

INITIAL
"Initial level of tank"
   Tanque.h = 1 * 'm';

OPTIONS
   TimeStart = 0 ;
   TimeStep    0.1 ;
   TimeEnd     2 ;
   TimeUnit  = 'h' ;
end
```



Degree of Freedom

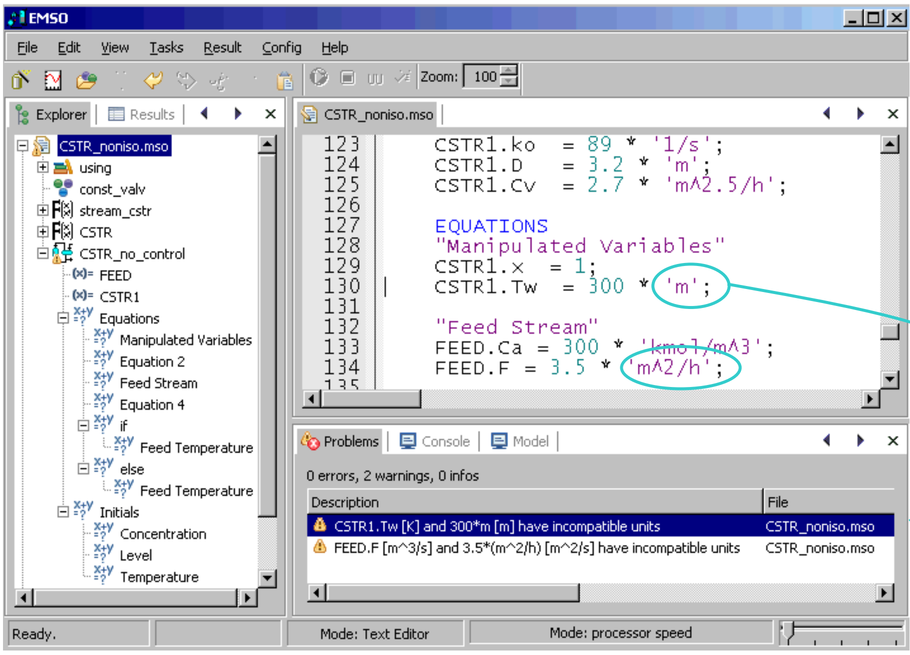
Dynamic Degree of Freedom

Parameters of DEVICES

Simulation options

183

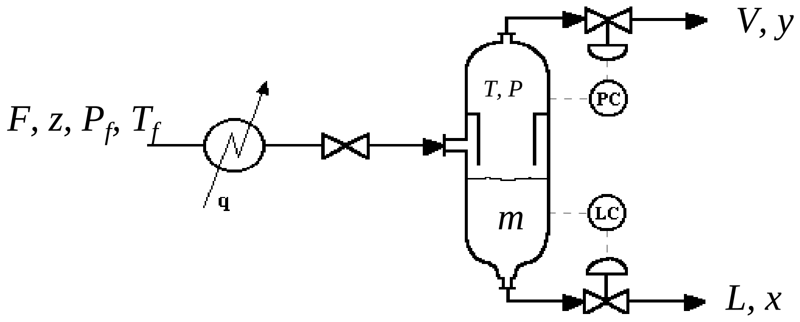
EMSO Tutorial
– Checking Units of Measurement –



incompatible units

184

Flash multi-component



A liquid-phase mixture of C hydrocarbons, at given temperature and pressure, is heated and continuously fed into a vessel drum at lower pressure, occurring partial vaporization. The liquid and vapor phases are continuously removed from the vessel through level and pressure control valves, respectively. Determine the time evolution of liquid and vapor stream composition and the vessel temperature and pressure, due to variations in the feed stream, keeping the heating rate constant.

– FLASH: model assumptions –

- negligible vapor holdup (no dynamics in vapor phase);
- thermodynamic equilibrium (ideal stage);
- no droplet drag in vapor stream;
- negligible heat loss to surroundings;
- Δ(internal energy) $\approx$ Δ(liquid-phase enthalpy);
- perfect mixture in both phases.

Overall mass balance (molar base):

$$\frac{dm}{dt} = F - V - L \qquad (1)$$

Component mass balance:

$$\frac{d}{dt}(m\, x_i) = F\, z_i - V\, y_i - L\, x_i \qquad (2)\ i = 1, 2, \ldots, C$$

Equilibrium:

$$y_i = K_i x_i \qquad (3)\ i = 1, 2, \ldots, C$$

$$K_i = f(T, P, x, y) \qquad (4)\ i = 1, 2, \ldots, C$$

Molar fraction:

$$\sum_{i=1}^{C} x_i = 1 \qquad (5)$$

Energy balance:

$$\frac{d}{dt}(m\,h) = F\,h_f + q - V\,H - L\,h \qquad (6)$$

Enthalpies:

$$h = f(T, P, x) \qquad (7)$$

$$H = f(T, P, y) \qquad (8)$$

$$h_f = f(T_f, P_f, z) \qquad (9)$$

Controllers:

$$L = f(m) \qquad (10)$$

$$V = f(P) \qquad (11)$$

variable	units of measurement
m	kmol
F, L, V	kmol s^{-1}
t	s
x_i, y_i, z_i	kmol $kmol^{-1}$
K_i	–
T, T_f	K
P, P_f	kPa
q	kJ s^{-1}
h, H, h_f	kJ $kmol^{-1}$

variables: m, F, L, V, t, x_i, y_i, z_i, K_i, T, T_f, P, P_f, q, h, H, h_f ➔ 13+4C
constants: ➔ 0
specifications: q, t ➔ 2
driving forces: F, z_i, T_f, P_f ➔ 3+C
unknown variables: m, L, V, x_i, y_i, K_i, T, P, h, H, h_f ➔ 8+3C
equations: 8+3C

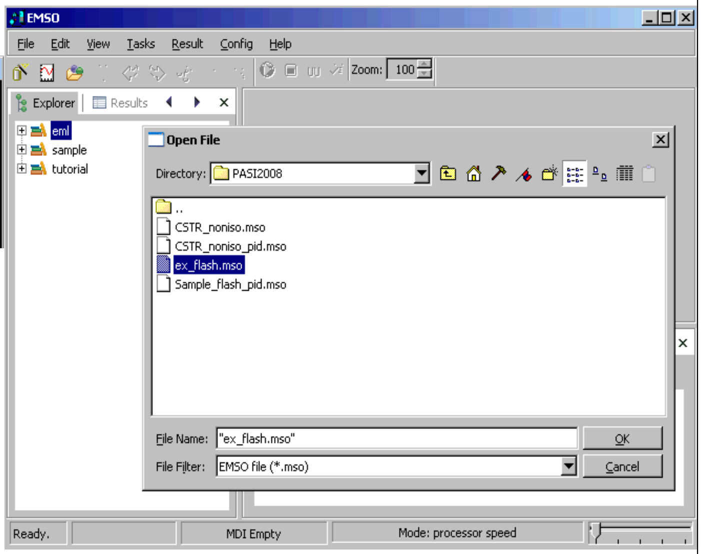
Degree of Freedom = variables – constants – specifications – driving forces – equations = unknown variables – equations = (13+4C) – 0 – 2 – (3+C) – (8+3C) = 0

Initial condition: $m(0)$, $x_i(0)$, $T(0)$ ➔ 2+C

Dynamic Degree of Freedom (index < 2) = differential equations – initial conditions = (2+C) – (2+C) = 0

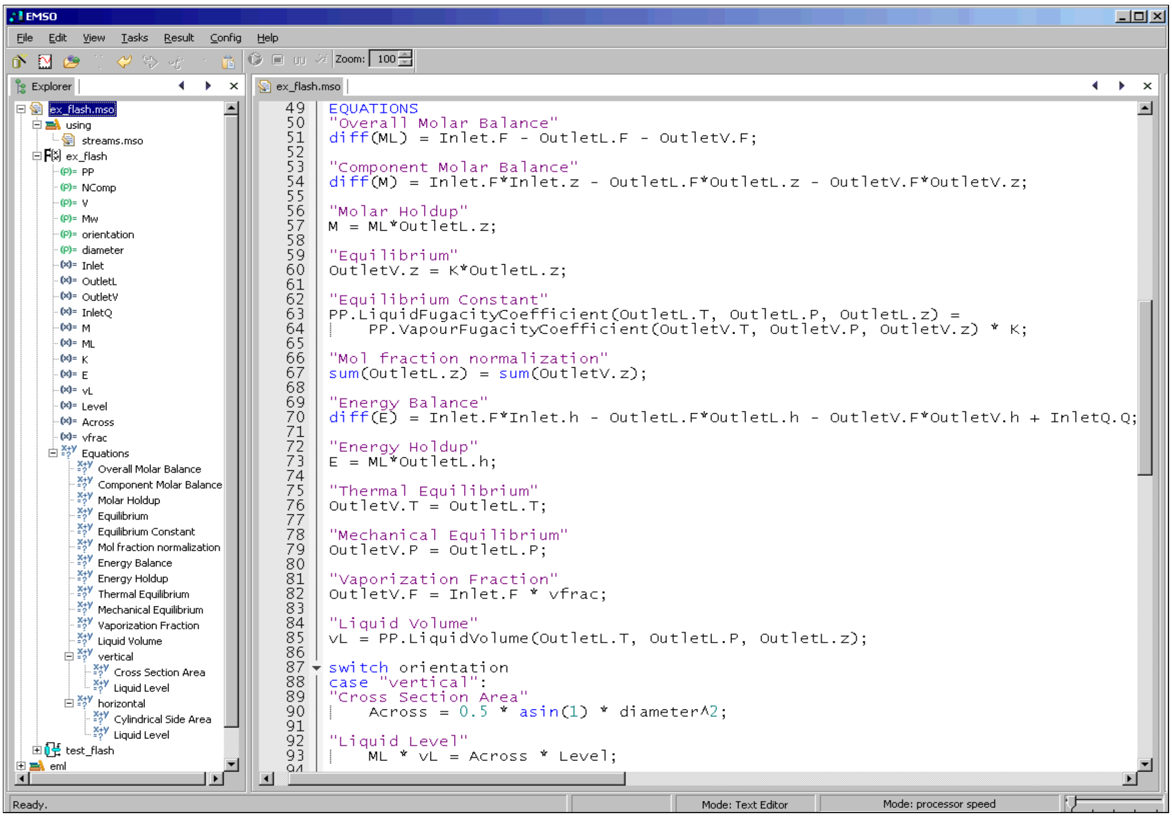
EMSO Tutorial
– FLASH: EMSO version –

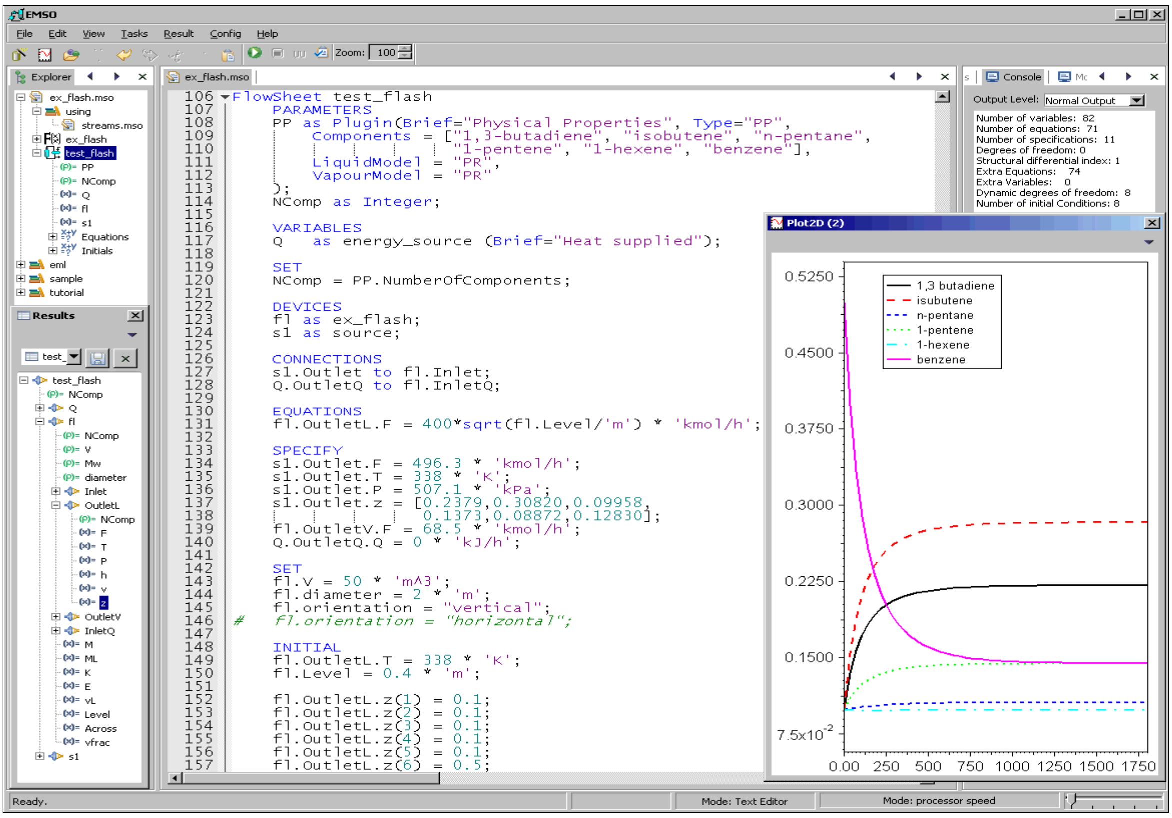
➢Running EMSO



Note: file `Sample_flash_pid.mso` has level and pressure controllers.

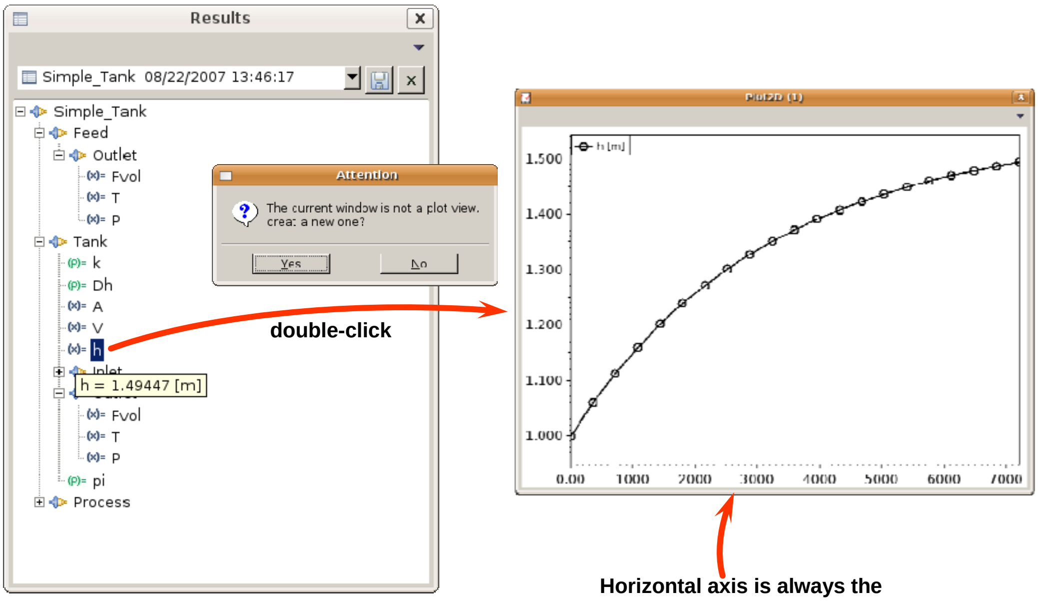
192


EMSO
File Edit View Tasks Result Config Help
Zoom: 100
Explorer
ex_flash.mso
using
streams.mso
ex_flash
PP
NComp
V
Mw
orientation
diameter
Inlet
OutletL
OutletV
InletQ
M
ML
K
E
vL
Level
Across
vfrac
Equations
Overall Molar Balance
Component Molar Balance
Molar Holdup
Equilibrium
Equilibrium Constant
Mol fraction normalization
Energy Balance
Energy Holdup
Thermal Equilibrium
Mechanical Equilibrium
Vaporization Fraction
Liquid Volume
vertical
Cross Section Area
Liquid Level
horizontal
Cylindrical Side Area
Liquid Level
test_flash
eml
ex_flash.mso
EQUATIONS
"Overall Molar Balance"
diff(ML) = Inlet.F - OutletL.F - OutletV.F;
"Component Molar Balance"
diff(M) = Inlet.F*Inlet.z - OutletL.F*OutletL.z - OutletV.F*OutletV.z;
"Molar Holdup"
M = ML*OutletL.z;
"Equilibrium"
OutletV.z = K*OutletL.z;
"Equilibrium Constant"
PP.LiquidFugacityCoefficient(OutletL.T, OutletL.P, OutletL.z) =
PP.VapourFugacityCoefficient(OutletV.T, OutletV.P, OutletV.z) * K;
"Mol fraction normalization"
sum(OutletL.z) = sum(OutletV.z);
"Energy Balance"
diff(E) = Inlet.F*Inlet.h - OutletL.F*OutletL.h - OutletV.F*OutletV.h + InletQ.Q;
"Energy Holdup"
E = ML*OutletL.h;
"Thermal Equilibrium"
OutletV.T = OutletL.T;
"Mechanical Equilibrium"
OutletV.P = OutletL.P;
"Vaporization Fraction"
OutletV.F = Inlet.F * vfrac;
"Liquid Volume"
vL = PP.LiquidVolume(OutletL.T, OutletL.P, OutletL.z);
switch orientation
case "vertical":
"Cross Section Area"
Across = 0.5 * asin(1) * diameter^2;
"Liquid Level"
ML * vL = Across * Level;
Ready.
Mode: Text Editor
Mode: processor speed


EMSO
File Edit View Tasks Result Config Help
Zoom: 100
Explorer
ex_flash.mso
using
streams.mso
ex_flash
test_flash
PP
NComp
Q
fl
s1
Equations
Initials
eml
sample
tutorial
Results
test_flash
ex_flash.mso
FlowSheet test_flash
PARAMETERS
PP as Plugin(Brief="Physical Properties", Type="PP",
Components = ["1,3-butadiene", "isobutene", "n-pentane",
"1-pentene", "1-hexene", "benzene"],
LiquidModel = "PR",
VapourModel = "PR"
);
NComp as Integer;
VARIABLES
Q as energy_source (Brief="Heat supplied");
SET
NComp = PP.NumberOfComponents;
DEVICES
fl as ex_flash;
s1 as source;
CONNECTIONS
s1.Outlet to fl.Inlet;
Q.OutletQ to fl.InletQ;
EQUATIONS
fl.OutletL.F = 400*sqrt(fl.Level/'m') * 'kmol/h';
SPECIFY
s1.Outlet.F = 496.3 * 'kmol/h';
s1.Outlet.T = 338 * 'K';
s1.Outlet.P = 507.1 * 'kPa';
s1.Outlet.z = [0.2379,0.30820,0.09958,
0.1373,0.08872,0.12830];
fl.OutletV.F = 68.5 * 'kmol/h';
Q.OutletQ.Q = 0 * 'kJ/h';
SET
fl.V = 50 * 'm^3';
fl.diameter = 2 * 'm';
fl.orientation = "vertical";
# fl.orientation = "horizontal";
INITIAL
fl.OutletL.T = 338 * 'K';
fl.Level = 0.4 * 'm';
fl.OutletL.z(1) = 0.1;
fl.OutletL.z(2) = 0.1;
fl.OutletL.z(3) = 0.1;
fl.OutletL.z(4) = 0.1;
fl.OutletL.z(5) = 0.1;
fl.OutletL.z(6) = 0.5;
Console
Output Level: Normal Output
Number of variables: 82
Number of equations: 71
Number of specifications: 11
Degrees of freedom: 0
Structural differential index: 1
Extra Equations: 74
Extra Variables: 0
Dynamic degrees of freedom: 8
Number of initial Conditions: 8
Plot2D (2)
1,3 butadiene
isubutene
n-pentane
1-pentene
1-hexene
benzene
0.5250
0.4500
0.3750
0.3000
0.2250
0.1500
7.5x10^-2
0.00 250 500 750 1000 1250 1500 1750
Ready.
Mode: Text Editor
Mode: processor speed

EMSO Tutorial

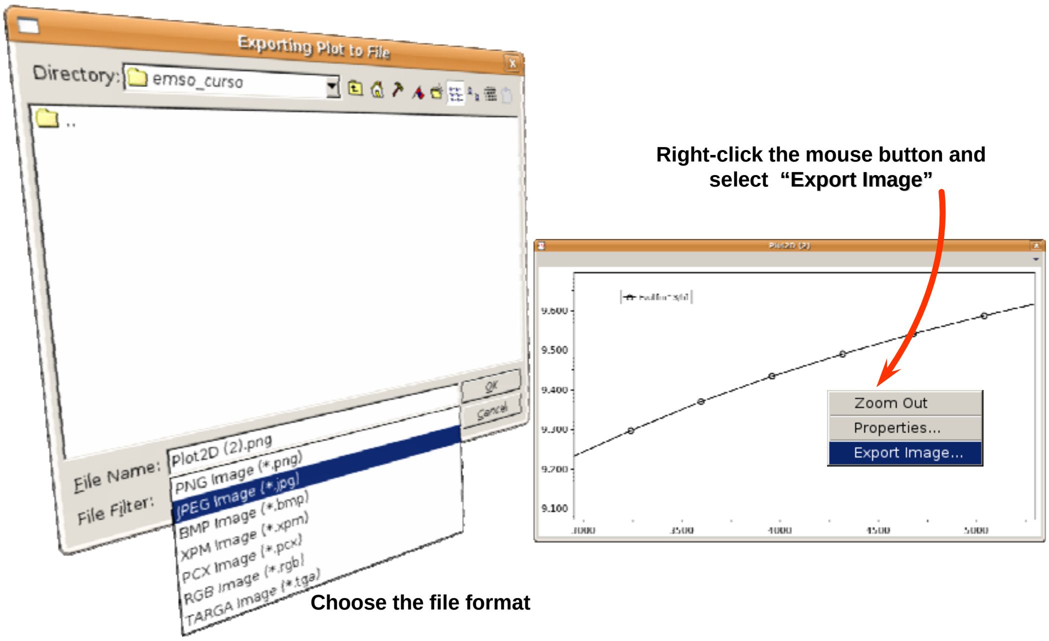
– Simulation Results: graphics –



Horizontal axis is always the independent variable (usually time)

EMSO Tutorial

– Simulation Results: exporting graphics –



Right-click the mouse button and select "Export Image"

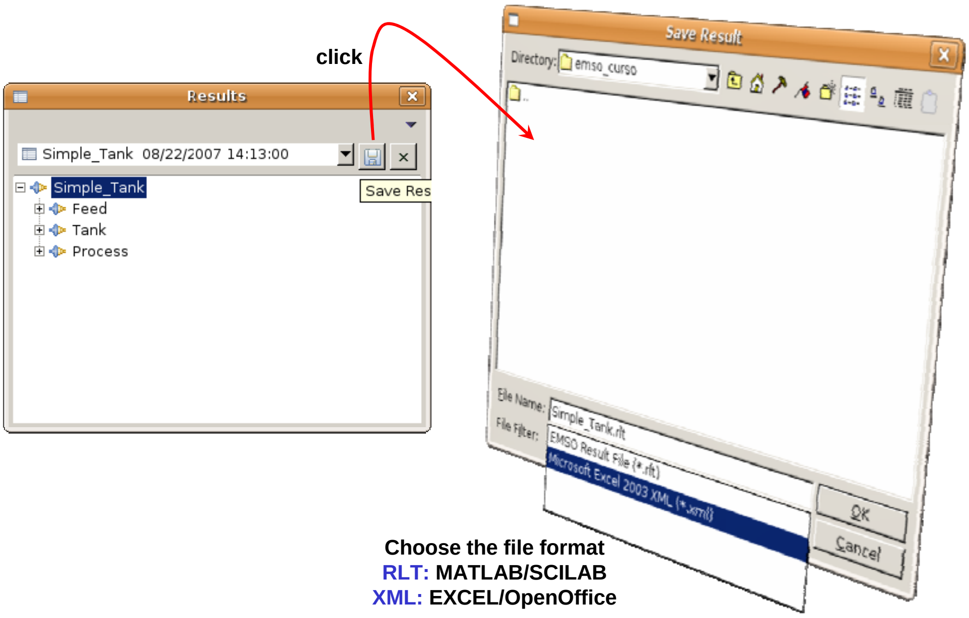
Choose the file format

196

EMSO Tutorial

– Simulation Results: exporting data –

click



Choose the file format

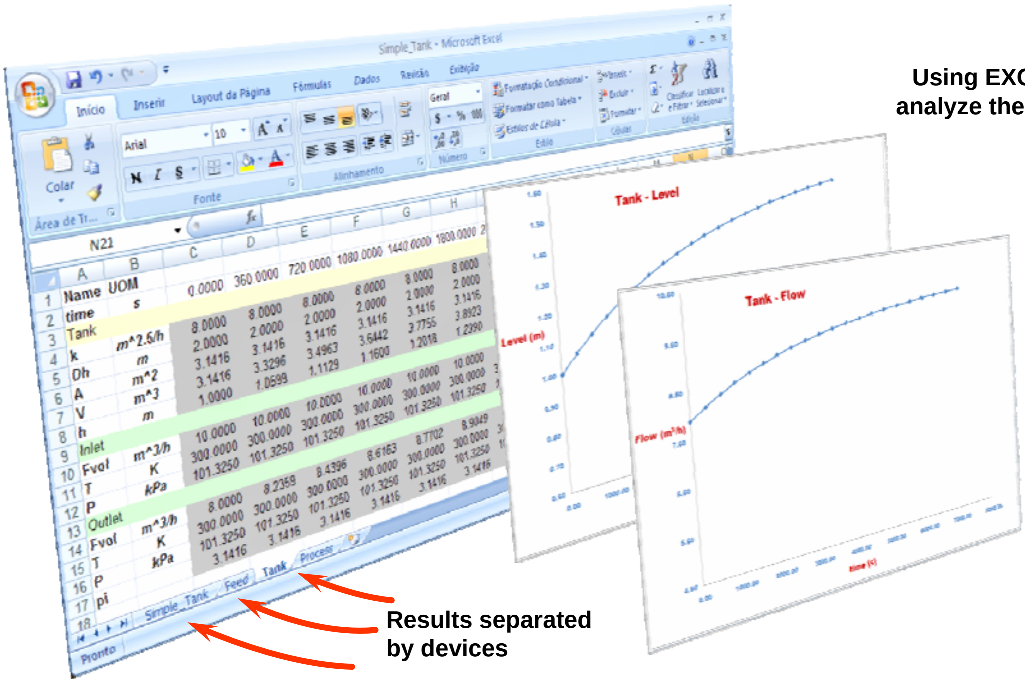
RLT: MATLAB/SCILAB

XML: EXCEL/OpenOffice

EMSO Tutorial

– Simulation Results: in spreadsheets –

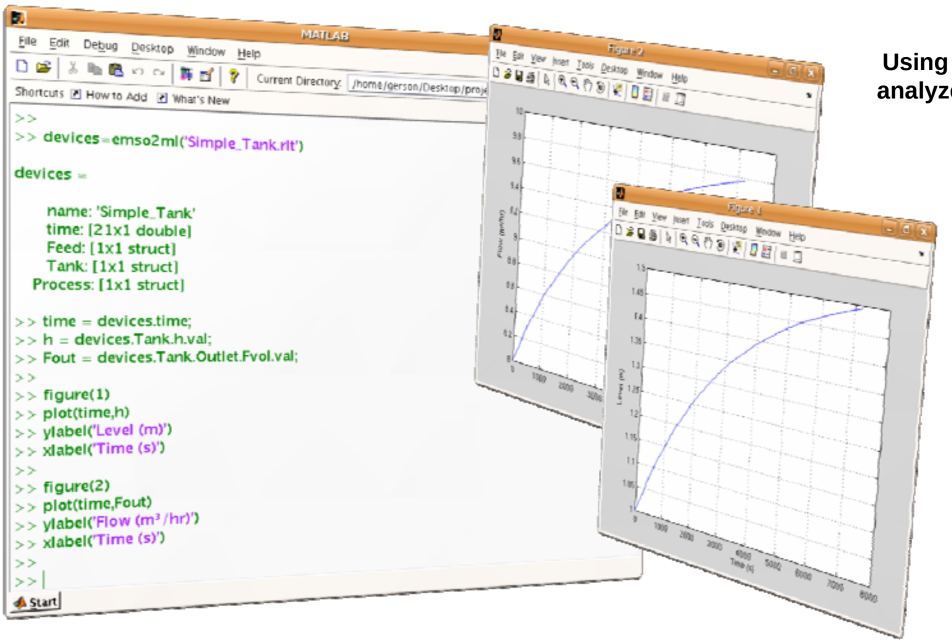
Using EXCEL to analyze the results



Results separated by devices

EMSO Tutorial

– Simulation Results: in MATLAB/SCILAB –

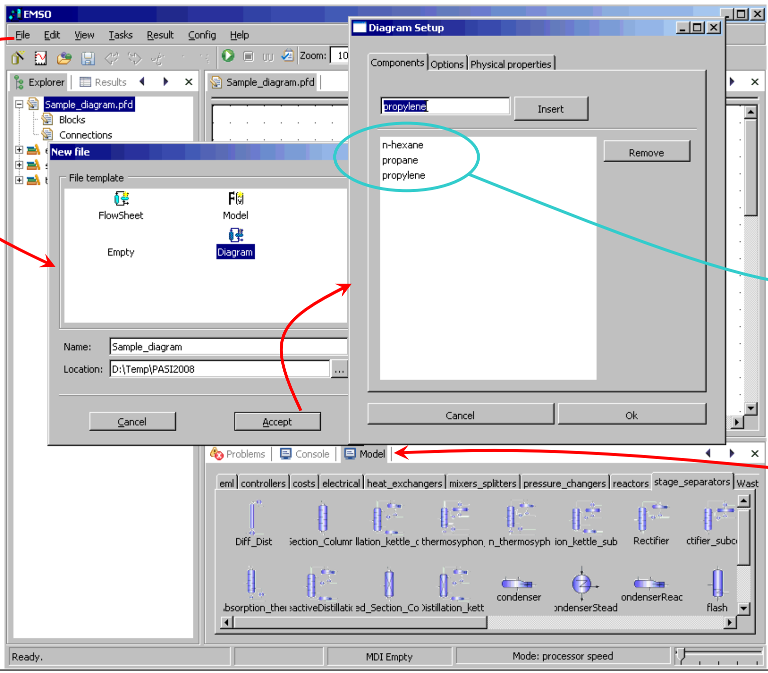


Using MATLAB to analyze the results

199

EMSO Tutorial

– Building Block Diagrams: create file –



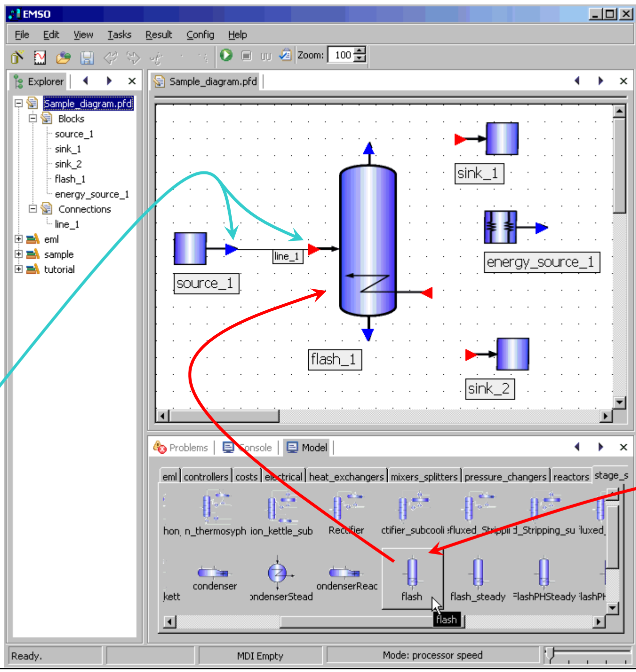
Selected components from physical properties package

Devices found in the model library

200

EMSO Tutorial

– Building Block Diagrams: select devices –



When making a connection, only compatible ports become available to connect

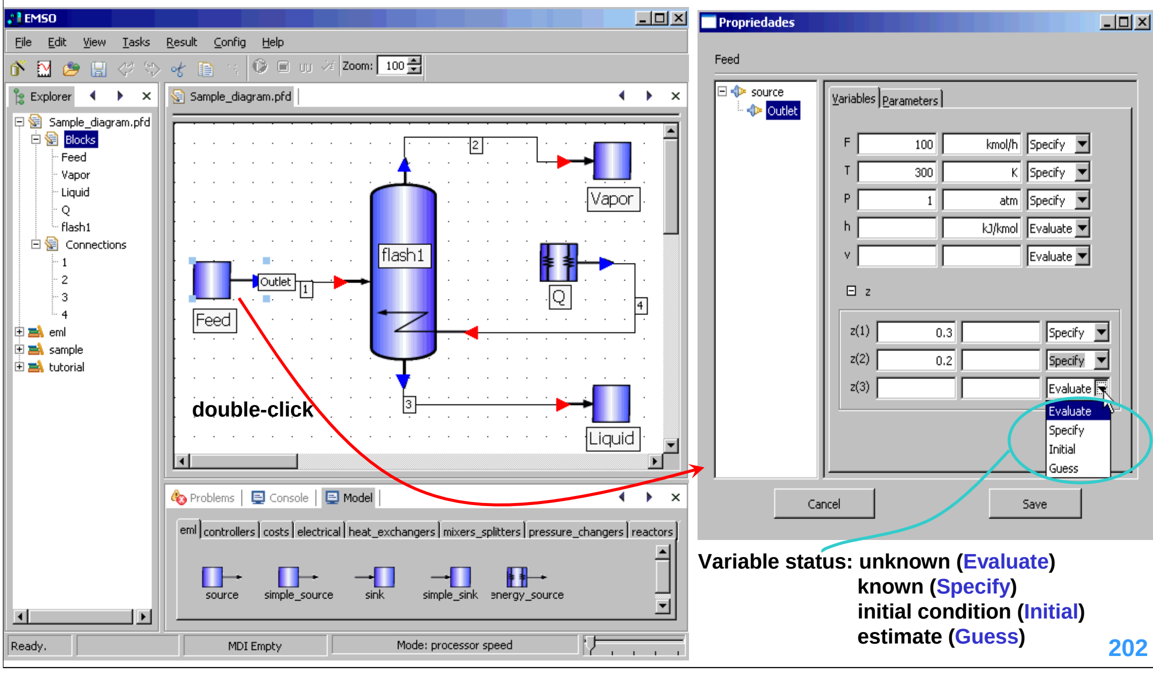
drag & drop ports to create a connection

click to create a device

201

EMSO Tutorial

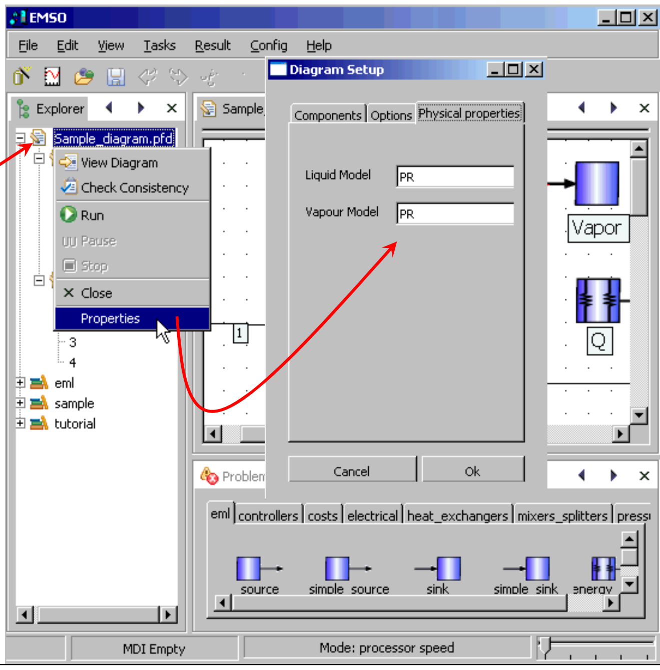
– Building Block Diagrams: set case study –



Variable status: unknown (Evaluate)
known (Specify)
initial condition (Initial)
estimate (Guess)

EMSO Tutorial
– Building Block Diagrams: thermodynamic –

right-click



Available models

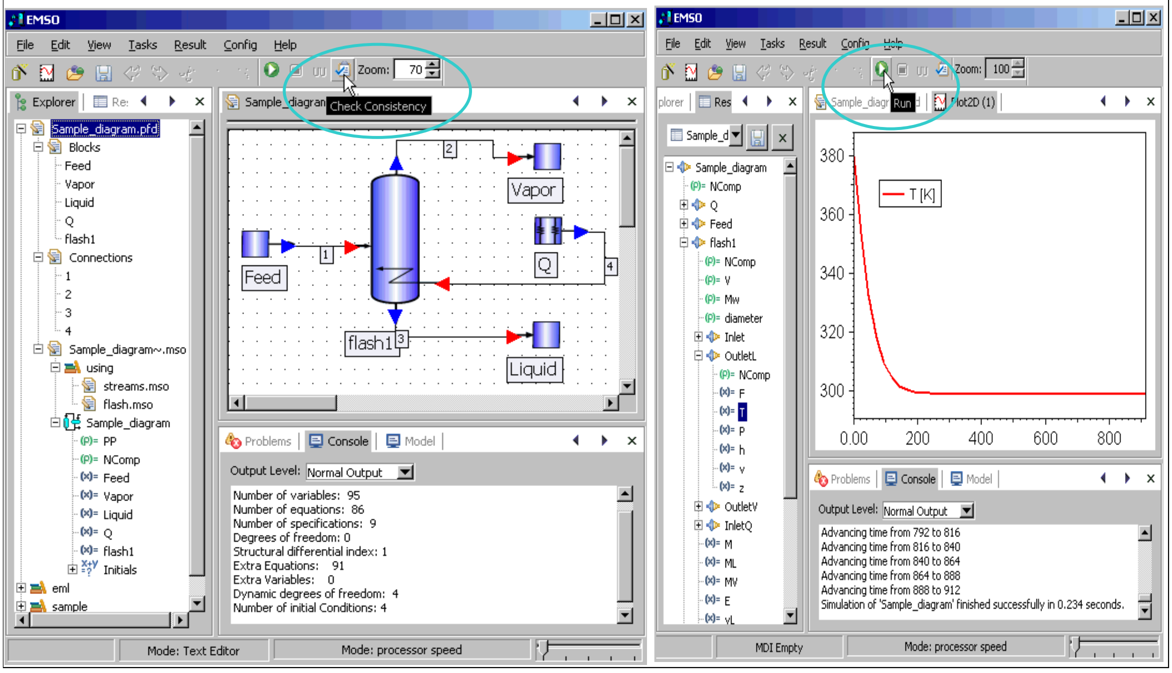
EOS Name	Description
Ideal	Ideal Gas, valid only for vapour models
IdealLiquid	Ideal Liquid, valid only for liquid models
RK	Redlich-Kwong
SRK	Soave-Redlich-Kwong
PR	Peng-Robinson
APR	Assymetric-PR
ASRK	Assymetric-RK
UNIFAC	UNIFAC (Dortmund)

PC-SAFT

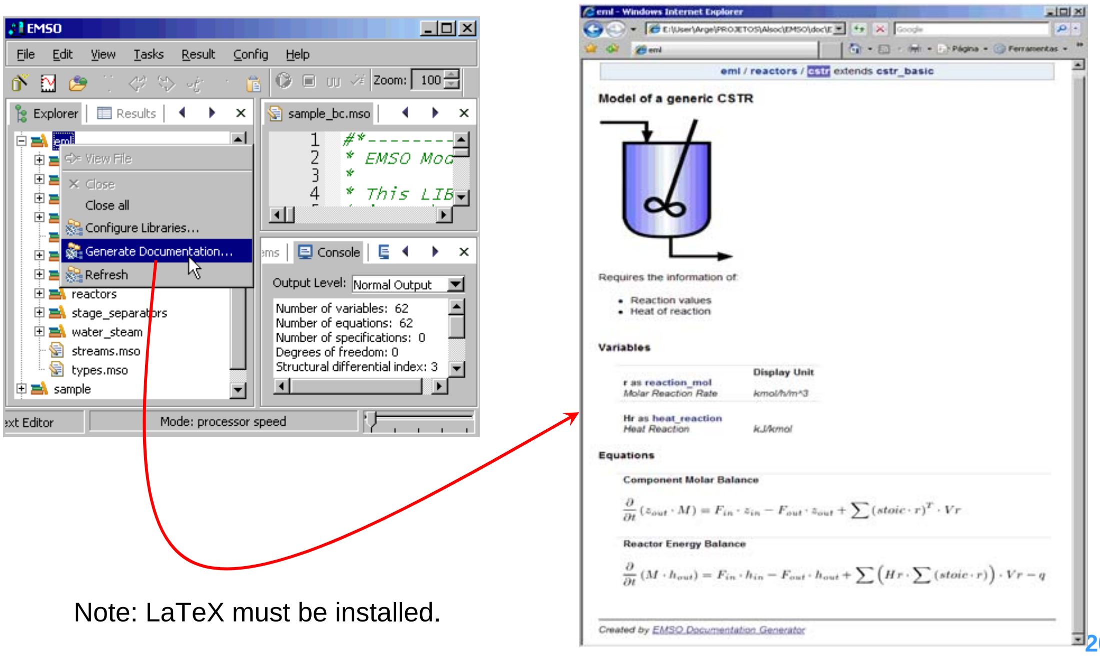
203

EMSO Tutorial

– Building Block Diagrams: simulating –



EMSO Tutorial
– Automatic Documentation –

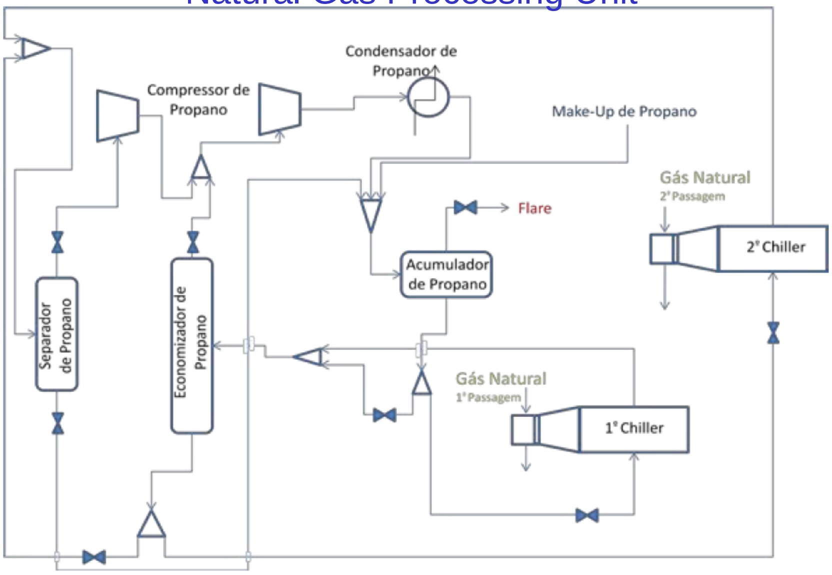


Note: LaTeX must be installed.

205

Some Industrial Applications

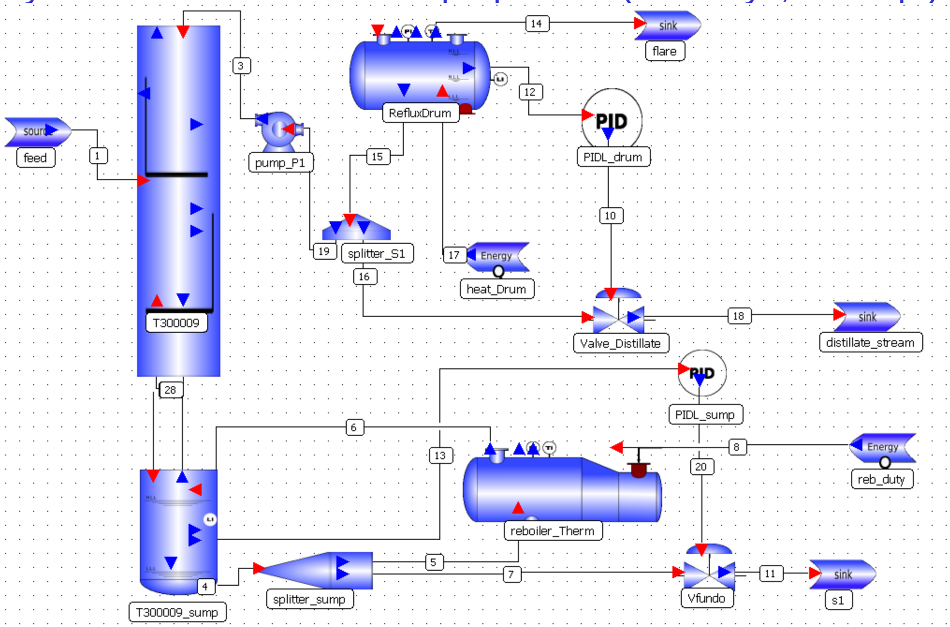
Dynamic Simulation of a Propane Refrigeration Cycle of a Natural Gas Processing Unit



206

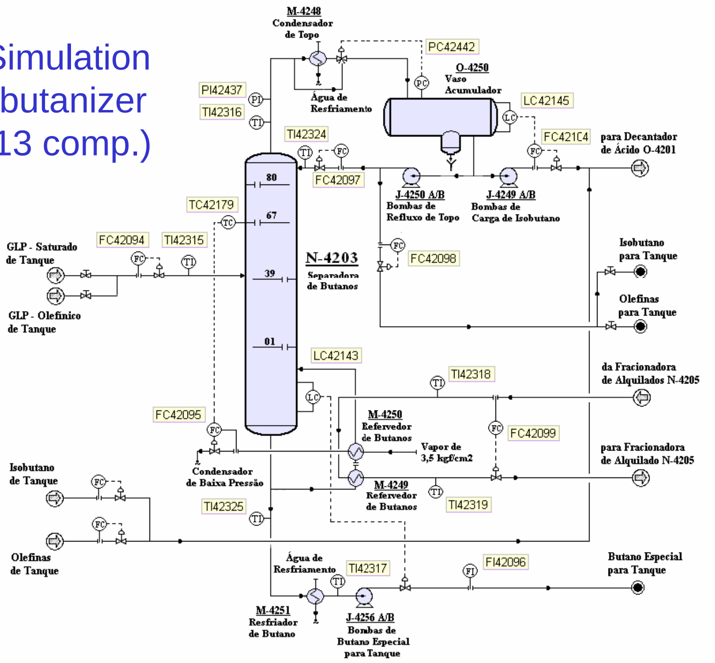
Some Industrial Applications

Dynamic Simulation of a Depropanizer (165 trays, 2 comp.)



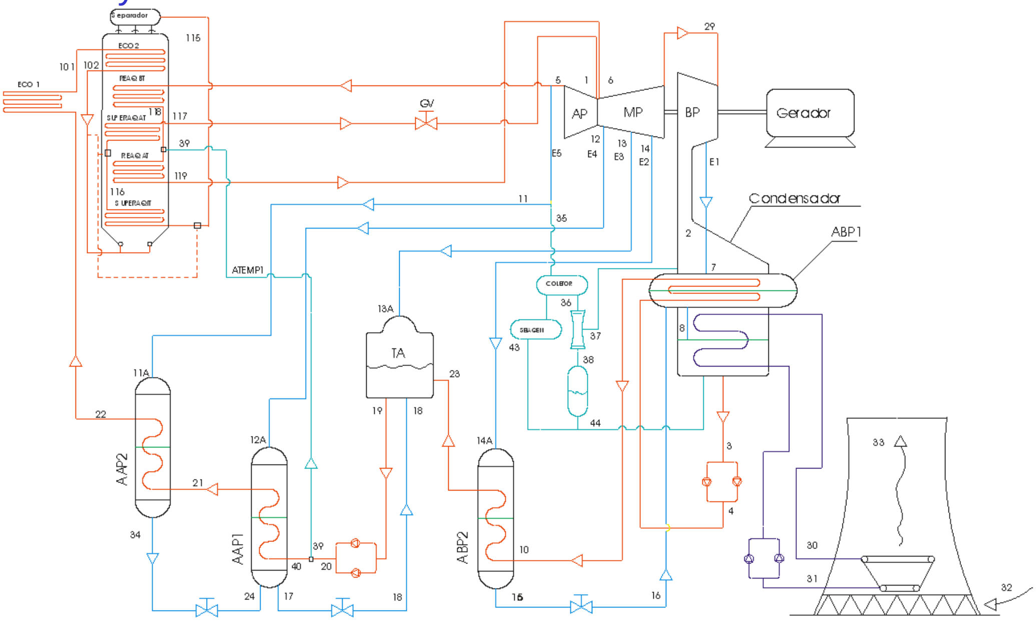
Some Industrial Applications

Dynamic Simulation of a Deisobutanizer (80 trays, 13 comp.)



Some Industrial Applications

Steady-State Simulation of a Power Plant with Pulverized Coal

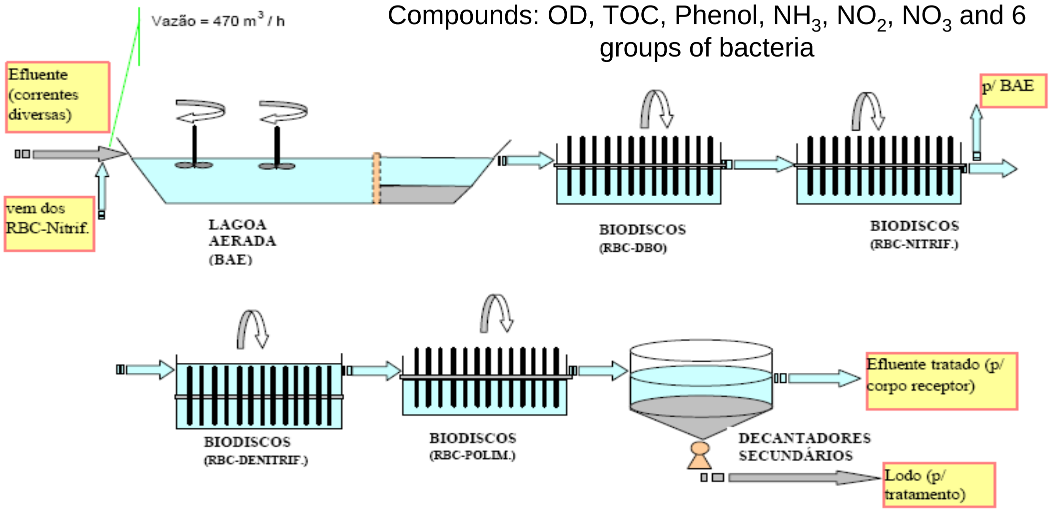


209

Some Industrial Applications

Dynamic Simulation of a Industrial Waste Water Treatment Unit (Müller et al., 2009)

Compounds: OD, TOC, Phenol, NH_3, NO_2, NO_3 and 6 groups of bacteria

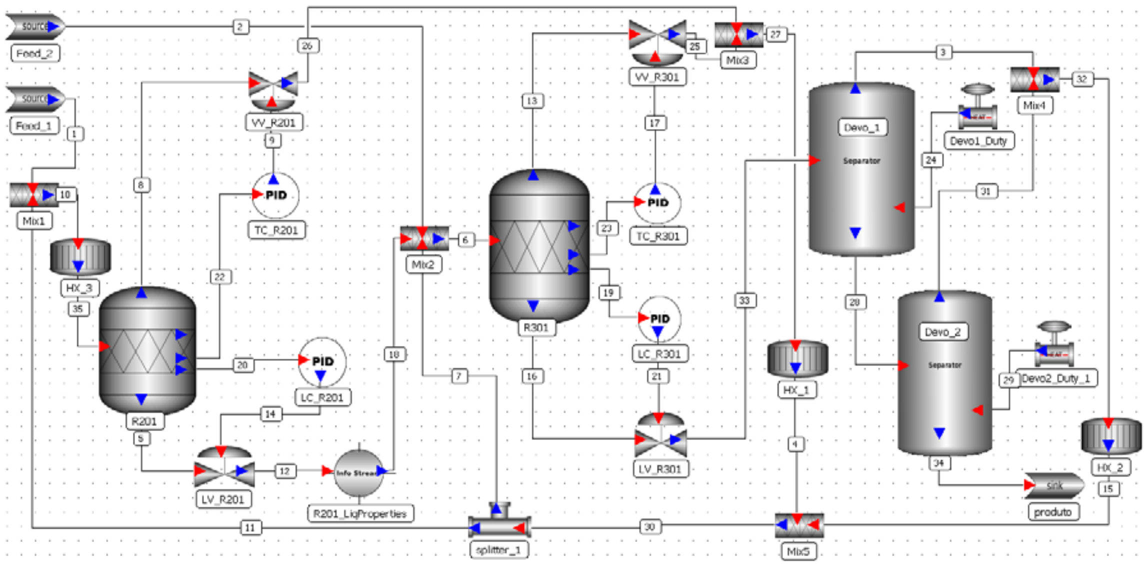


Fonte: Ricardo Motta

210

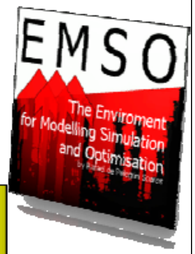
Some Industrial Applications

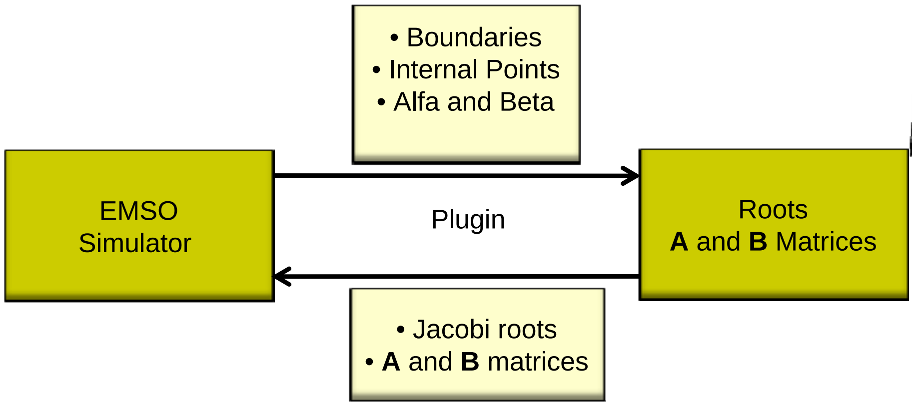
Dynamic Simulation of a General Purpose Polystyrene Process



211

Method of Orthogonal Collocation with EMSO





DD as Plugin (Type="OCFEM", Boundary="BOTH", InternalPoints=5 alfa=1, beta=1)

Plugin: ocfem_emso.dll

212

Fixed-bed Reactor with Axial Dispersion
(reaction of order *m*)

$$\frac{\partial y}{\partial \tau} + \frac{\partial y}{\partial x} = \frac{1}{Pe}\frac{\partial^2 y}{\partial x^2} - Da\, y^m$$

Boundary conditions:

$$-\frac{1}{Pe}\left.\frac{\partial y}{\partial x}\right|_{x=0} = 1 - y(\tau,0) \qquad \text{or} \qquad y(\tau,0) = 1$$

$$\left.\frac{\partial y}{\partial x}\right|_{x=1} = 0$$

Initial conditions:

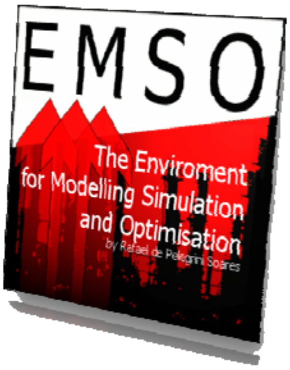
$$y(0,x) = 0$$

213

PDE

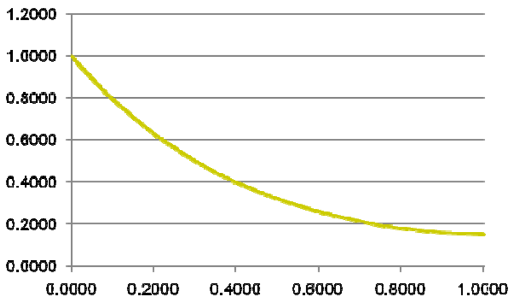
Method of Lines: D.F. and Orthogonal Collocation

Example: add Plugin ocfem_emso.dll and execute flowsheets of files FDM_ss.mso, OCM_ss.mso and OCFEM_ss.mso, and compare results of discretizations. Repeat for the dynamic simulation in files FDM_din.mso e OCM_din.mso.



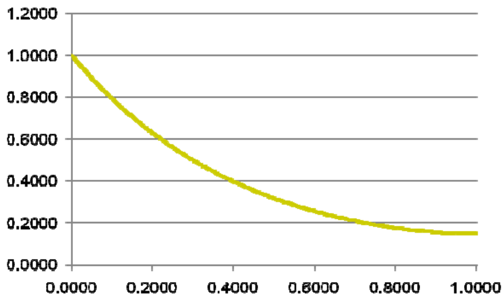
214

Comparing Results



OCM by EMSO $\alpha = 1$ $\beta = 1$
Number of internal points: 5

y(x=1) = 0.151475 (error of 0.038%)



Method of Finite Differences
Number of internal points: 6000

y(x=1) = 0.15155 (error of 0.087%)

y(x=1) = 0.151418 (exact)

215

Case Study

- **Production of acetic anhydride in adiabatic PFR**
  - Acetic anhydride is often produced by reacting acetic acid with ketene, obtained by heating acetone at 700-770°C.
  - A important step is the vapor phase cracking of acetone to ketene and methane:

$$CH_3COCH_3 \rightarrow CH_2CO + CH_4$$

  - The second step is the reaction of ketene with acetic acid.

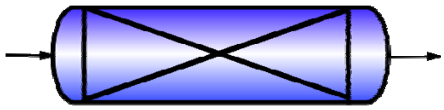
$$CH_2CO + CH_3COOH \rightarrow (CH_3CO)O$$

Ref: G. V. Jeffreys, *A Problem in Chemical Engineering Design: The Manufacture of Acetic Anhydride*, 2nd ed. (London: Institution of Chemical Engineers, 1964)

Case Study

Problem Definition

– The first production step is carried out in a vapor phase reaction of acetone in an adiabatic PFR.



where A = acetone; B = ketene and C = methane

$$A \rightarrow B + C$$

– The reaction is of 1ª order in relation to acetone in the cracking reaction, with Arrhenius constant given by:

- k – seconds^{-1}
- T – Kelvin

$$k = \exp\left(34.34 - \frac{34222}{T}\right)$$

217

Case Study

Process Description

- Reactor geometry
  - adiabatic continuous tubular reactor;
  - bank of 1000 tubes of 1 in sch. 40 with cross section of 0.557 m^2;
  - total length of 2.28 m;
- Operating conditions
  - feed temperature 762^{o}C (1035 K);
  - operating pressure: 1.6 atm
  - feed flow rate of 8000 kg/h (137.9 kmol/h);
- Composition
  - acetone, ketene and methane
  - feed of pure acetone
- Kinetics
  - first order reaction,
  - pre-exponential factor (k_0): 8.2 x 10^{14} s^{-1}
  - activation energy (E/R): 34222 K
  - heat of reaction: -80.77 kJ/mol

Case study

– Production of acetic anhydride –

Example: run FlowSheet in file PFR_Adiabatico.mso and plot steady-state temperature and composition profiles. Show also the evolution of the temperature profile. Discuss the type and quality of discretization.