CEPAC Chemical Engineering PanAmerican Collaboration

PASI 2008

Pan American Advanced Studies Institute Program on Process Systems Engineering

UFRGS UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

UFRJ COPPE

Chemical Engineering Departments

GIMSCOP

Group of Integration, Modeling, Simulation, Control, and Optimization of Processes

Tutorial on

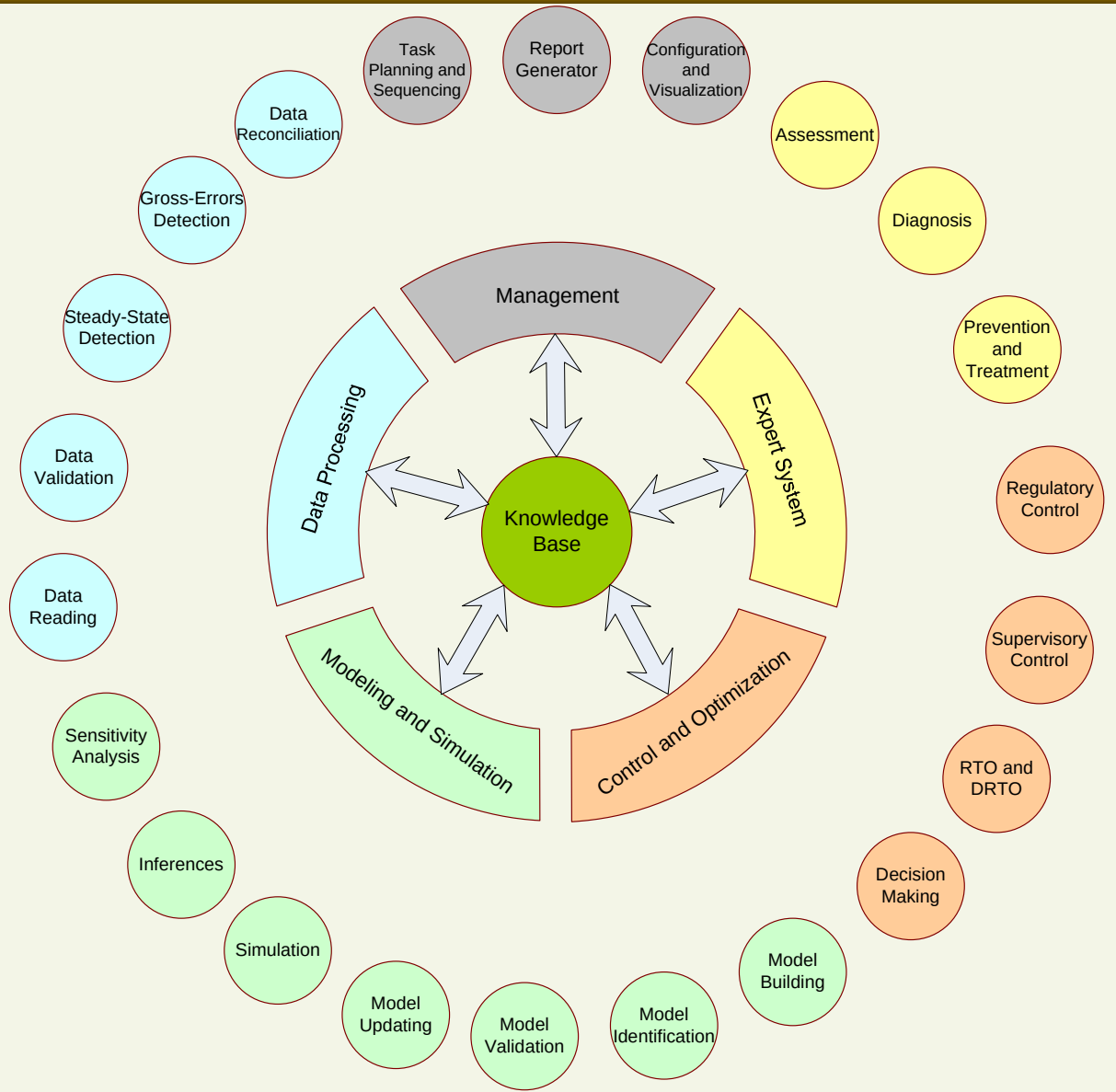
Equation-Oriented Dynamic Simulation using EMSO

Argimiro R. Secchi

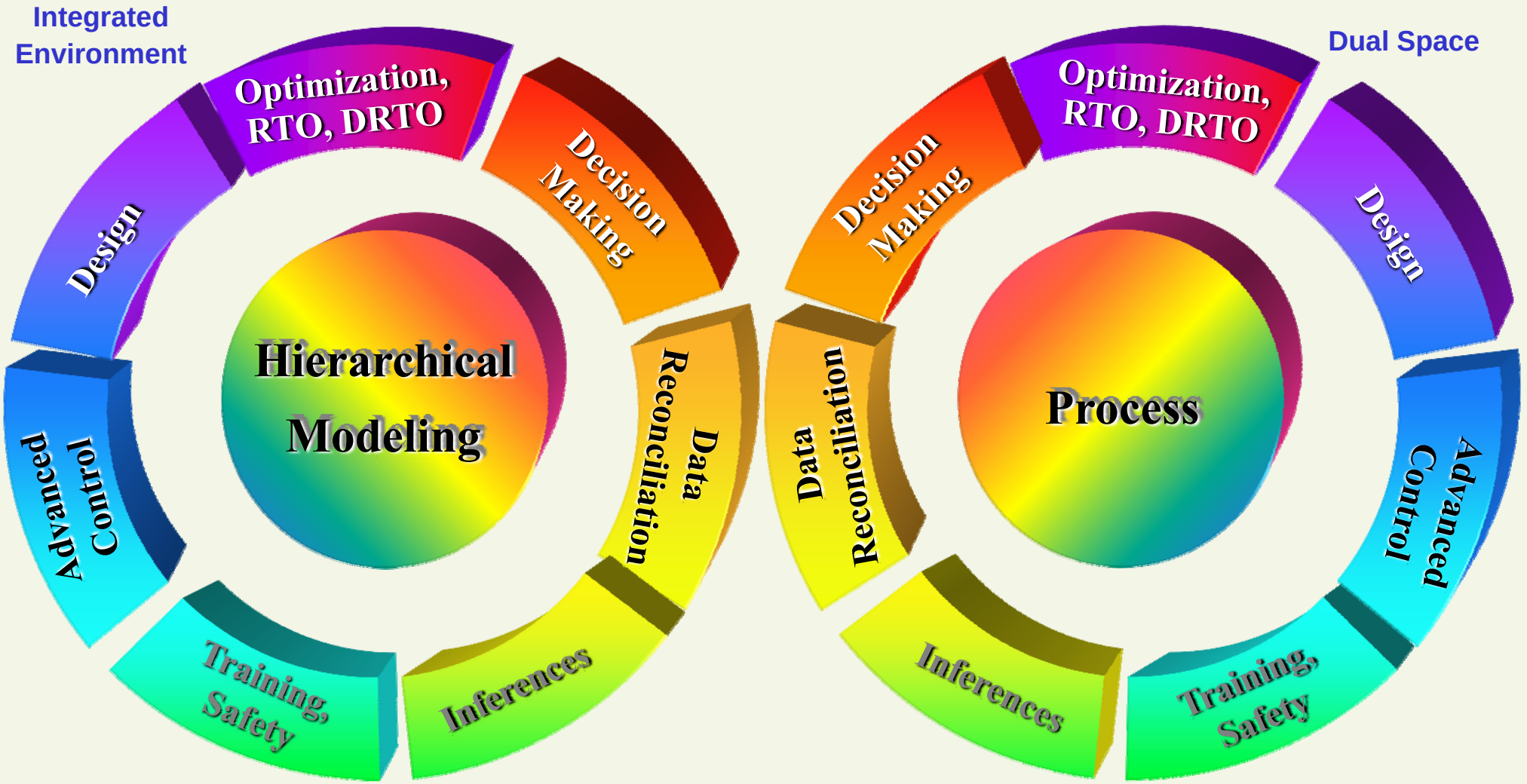
Rafael de Pelegrini Soares

Mar del Plata
August 12th, 2008

The Dream of a Process Engineer
– Fully-Integrated System and Tools –

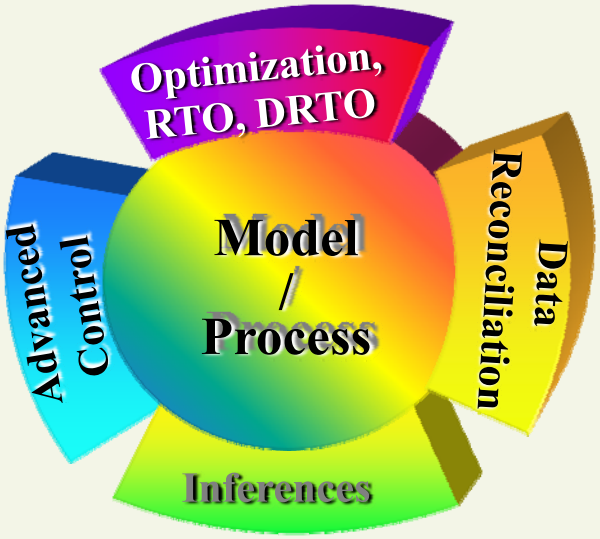


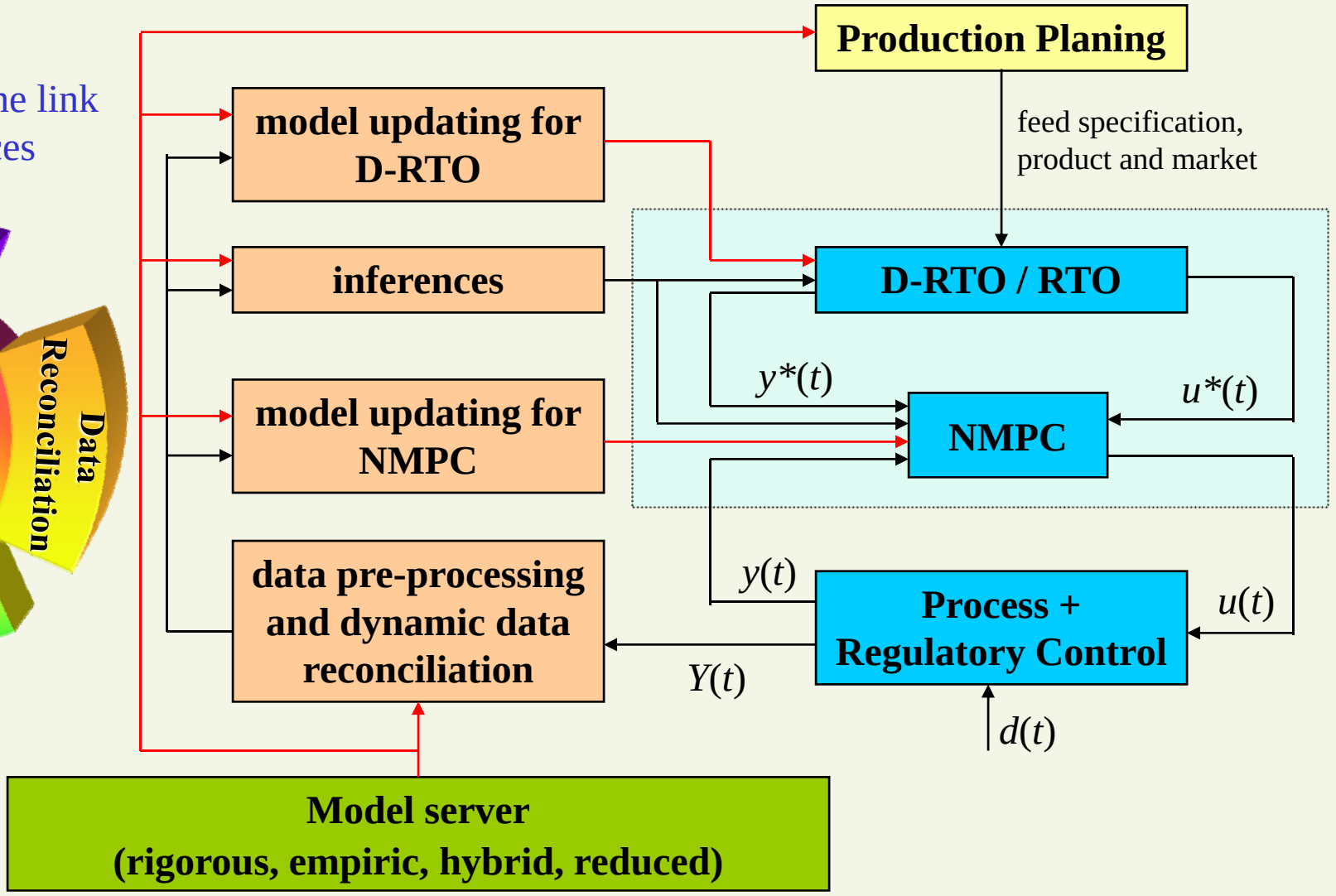
Unified Modeling Environment
– A Necessary Condition –



An Example of System Integration

In this context, the Knowledge Base is the link between the two spaces





CAPE Tools

- A movement from Sequential Modular to Equation-Oriented (EO) tools is clear
- Key advantages of EO:
  - Models can be inspected
  - Models can be refined or reused
  - Same model as the source for several tasks: simulation, optimization, parameter estimation, data reconciliation, etc. → integrated environment
- Some disadvantages:
  - Lack of assistance in model development
  - It is very difficult to fix ill-posed models

Outline

Equation-Oriented Dynamic Simulation using EMSO:

1. What is ESMO?
2. Building dynamic models
3. EMSO tutorial
4. Dynamic Degree of Freedom
5. Debugging techniques

1. What is EMSO?

- EMSO stands for "**Environment for Modeling Simulation and Optimization**"
- Development started in 2001, 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

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 **A**mbiente **L**ivre para **S**imulação, **O**timização e **C**ontrole 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

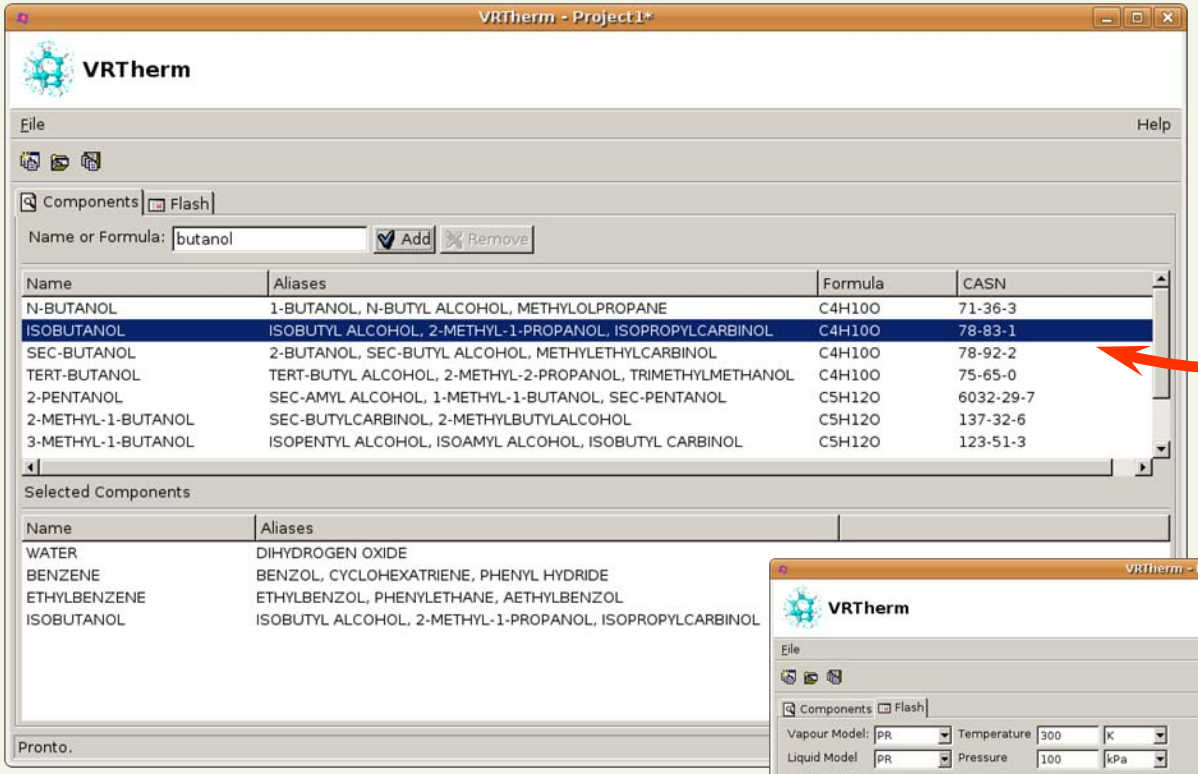
EMSO Key Features

- ✓ 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 event handling
- ✓ Multitask for concurrent and real-time simulations
- ✓ Very 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

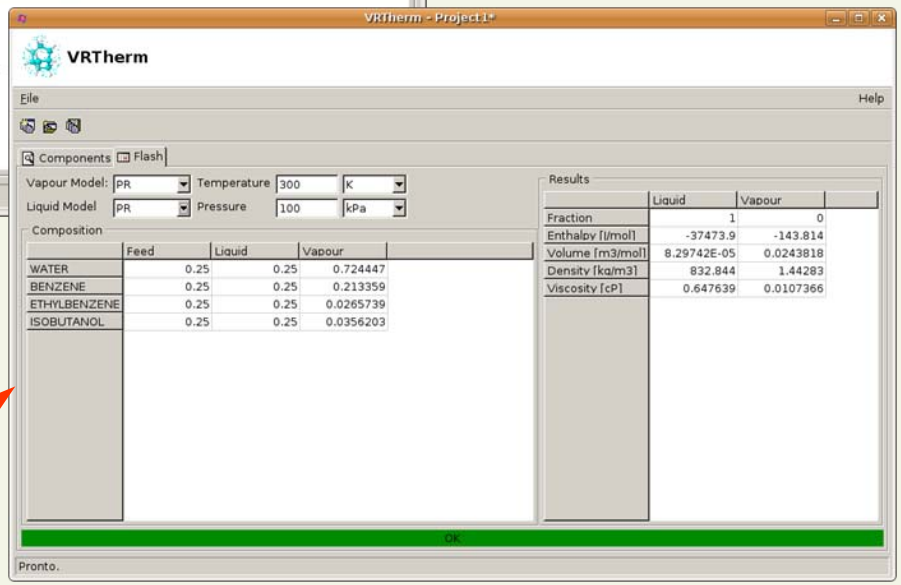
What can I do with EMSO?

- ☑ Steady-state simulations
- ☑ Dynamic simulations
- ☑ Steady-state optimizations
- ☑ 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

Thermodynamic and Physical Properties – Plugin



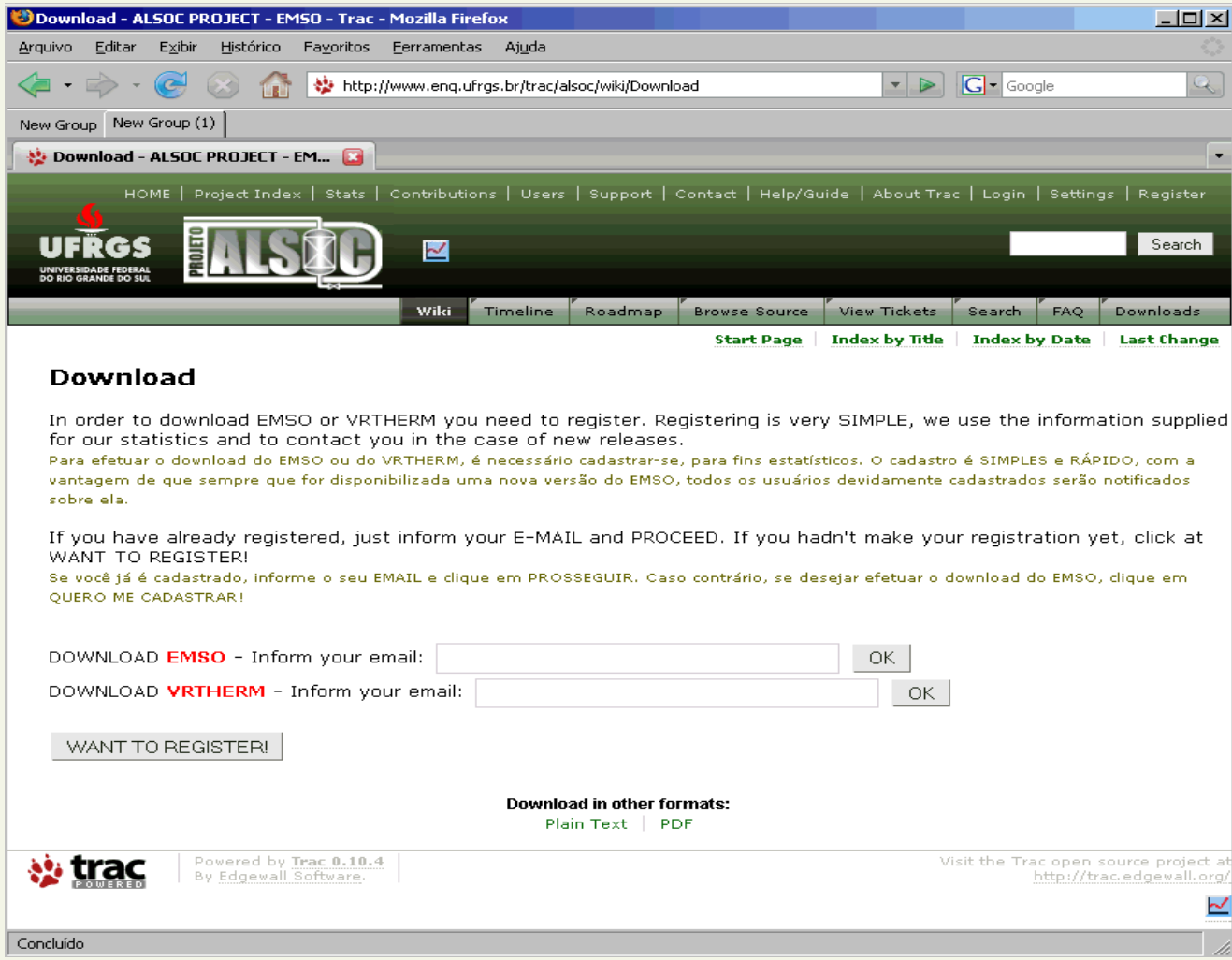
Data bank with about 2000 pure compounds



Mixture properties calculation

How can I install EMSO?

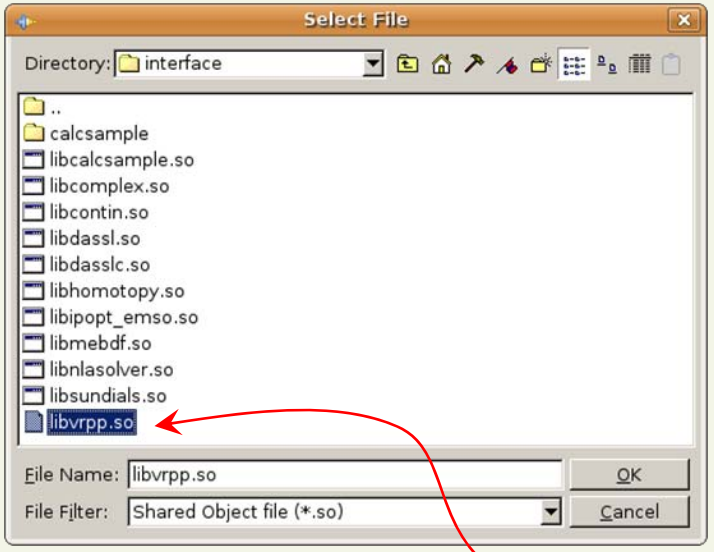
- Download EMSO and VRTherm packages from http://www.enq.ufrgs.br/alsoc
- Run the setup programs
- Run EMSO
- Add the physical properties package using the Config Plugins option in the menu
- Select and example and run it

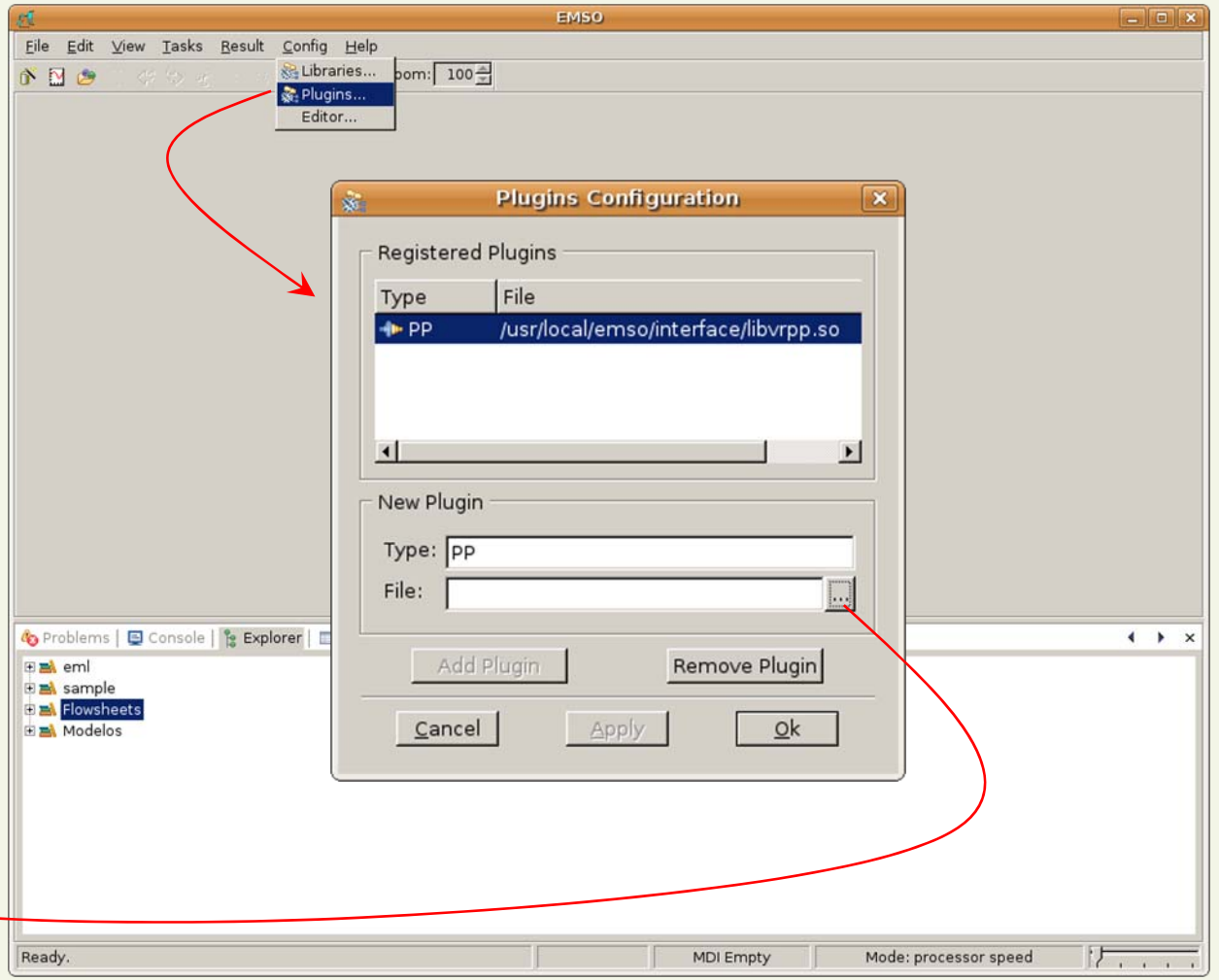


Configuring Plugin
– VRTherm package: vrpp –

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

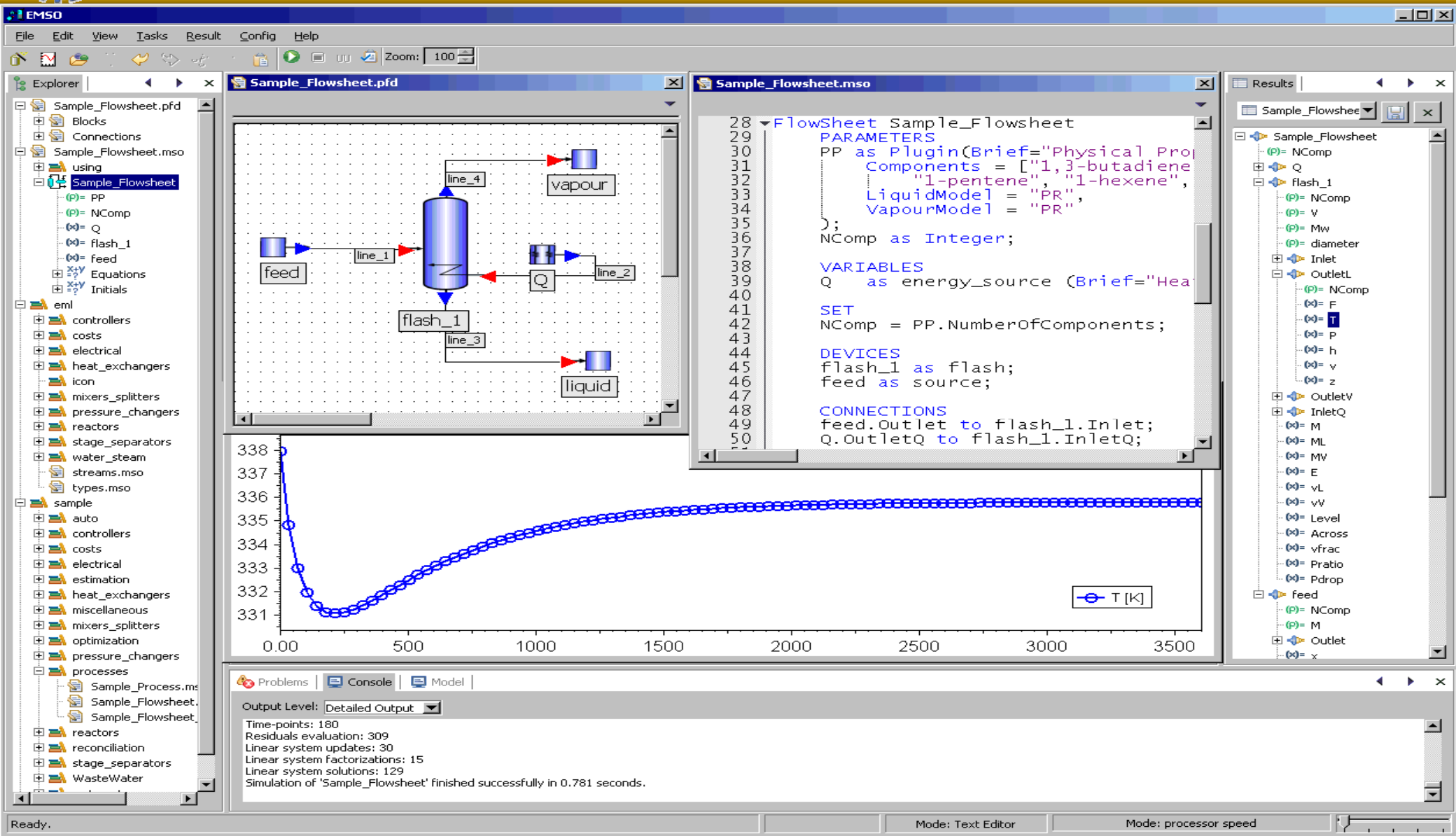
Config ⟶ *Plugins*





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

Integrated GUI
– Running an example –



2. Building Dynamic Models
– Where dynamic simulation is necessary –

- **Batch and semi-batch processes**

(Analysis, Control, Dynamic optimization, Optimal design, Parameter estimation, Start-up operations)

- **Dynamic real-time optimization (D-RTO)**

(NMPC, Plant-wide optimization, Product transitions, Model updating, Virtual analyzers)

- **Advanced process control**

(Control structure design, Model reduction, Controllability and operability, Model-based control, Controller tuning, Nonlinear dynamics)

- **Startups, shutdowns and transitions**

(Start-up strategies, Safety studies, Plant shutdown, Process transitions, Troubleshooting)

- **Process intensification**

(Complex systems, Oscillatory motion, Reaction/separation processes, Auto-refrigerated reactors)

- **Teaching and training**

(Classroom teaching, Operators training)

Building Dynamic Models
– Equation-Oriented models –

✓ **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:**

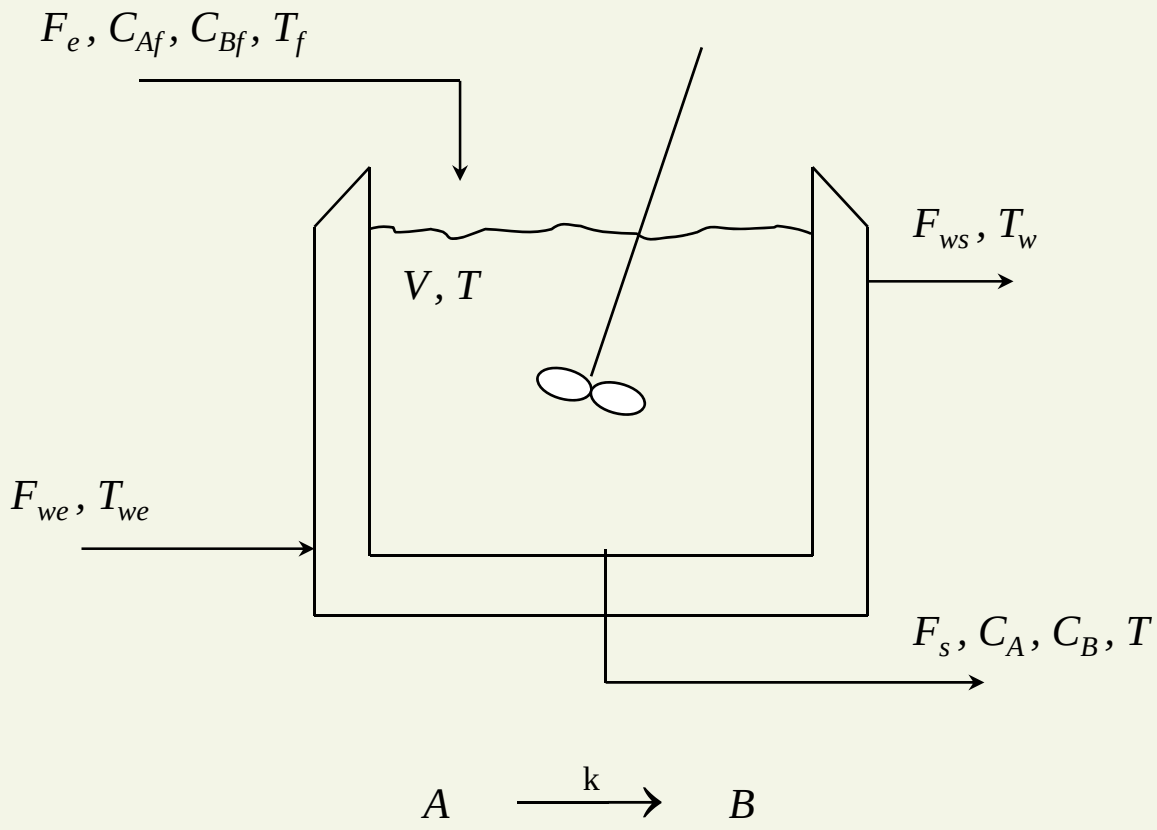
- Number of equations and variables does not match
- Equations of the model are inconsistent (linear dependence, etc.)
- The number of initial conditions does not match (dynamic simulation)

Building Dynamic Models
– Difficulties in Dynamic Simulation –

- **Reliable models**
- **Truly standard interfaces and open source models**
- **High-Index DAE systems**
- **Large-Scale systems**
- **Model consistency:**
  - Degree of Freedom (DoF)
  - Dynamic Degree of Freedom (DDoF)
  - Units of measurement
  - Structural non-singularity
  - Consistent initial condition

Building Dynamic Models
– A simple example –

Non-isothermal CSTR



Building Dynamic Models
– CSTR: 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 6×10^4 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 $m^{2.5}/h$.

Building Dynamic Models
– CSTR: 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) $\approx \Delta$(enthalpy);
- negligible variation of potential and kinetic energies;
- constant volume in the jacket;
- thin metallic wall with negligible heat capacity.

Building Dynamic Models
– 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:

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

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

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

Building Dynamic Models
– 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 $\hat{H}=\hat{U}+P\hat{V}$

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

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

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

Building Dynamic Models
– CSTR: modeling –

where

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

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

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

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

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

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

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

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

$$x = f(h) \qquad \text{Level control} \quad (13)$$

$$T_w = f(T) \qquad \text{Temperature control} \quad (14)$$

Building Dynamic Models
– CSTR: 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}

Building Dynamic Models
– CSTR: 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

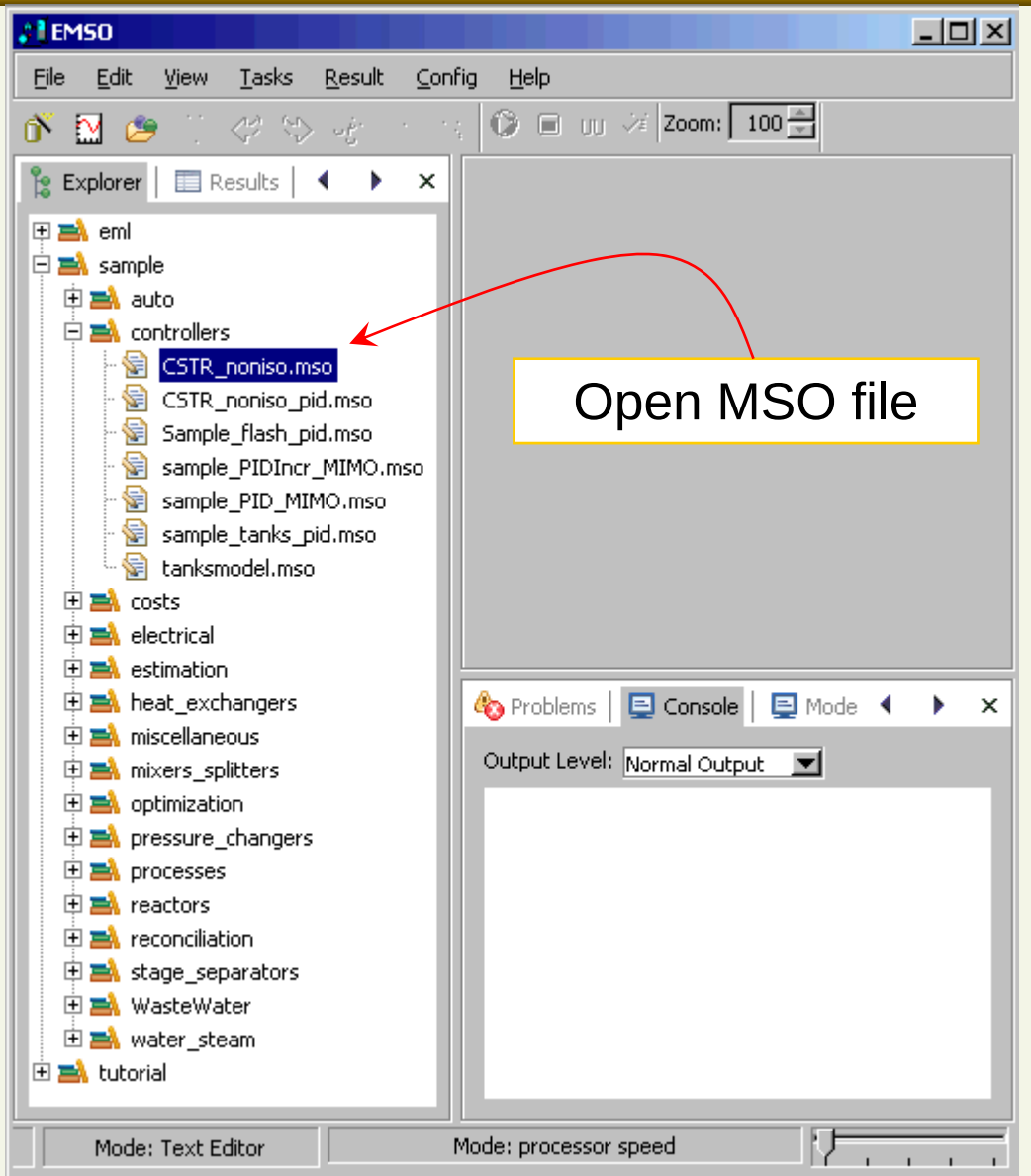
Degree of Freedom = variables – constants – specifications – driving forces – equations = unknown variables – equations = 27 – 9 – 1 – 3 – 14 = 0

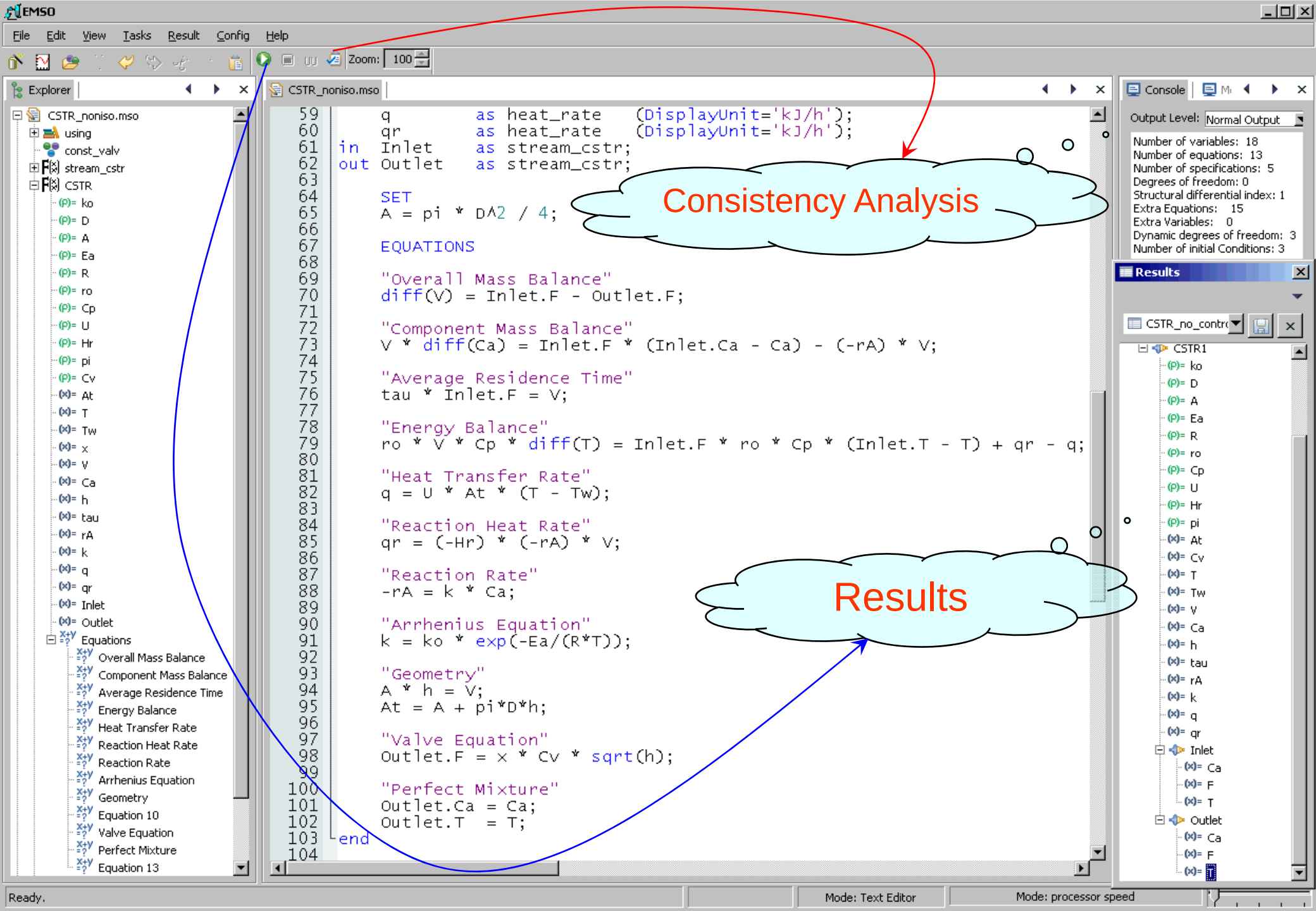
Initial condition: $h(0), C_A(0), T(0)$ ➔ 3

Dynamic Degree of Freedom (index < 2) = differential equations – initial conditions = 3 – 3 = 0

Building Dynamic Models
– CSTR: EMSO version –

- Running EMSO




EMSO
File Edit View Tasks Result Config Help
Zoom: 100
Explorer
CSTR_noniso.mso
using
const_valv
stream_cstr
CSTR
ko
D
A
Ea
R
ro
Cp
U
Hr
pi
Cv
At
T
Tw
x
V
Ca
h
tau
rA
k
q
qr
Inlet
Outlet
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
CSTR_noniso.mso
59 q as heat_rate (DisplayUnit='kJ/h');
60 qr as heat_rate (DisplayUnit='kJ/h');
61 in Inlet as stream_cstr;
62 out Outlet as stream_cstr;
63
64 SET
65 A = pi * D^2 / 4;
66
67 EQUATIONS
68
69 "Overall Mass Balance"
70 diff(V) = Inlet.F - Outlet.F;
71
72 "Component Mass Balance"
73 V * diff(Ca) = Inlet.F * (Inlet.Ca - Ca) - (-rA) * V;
74
75 "Average Residence Time"
76 tau * Inlet.F = V;
77
78 "Energy Balance"
79 ro * V * Cp * diff(T) = Inlet.F * ro * Cp * (Inlet.T - T) + qr - q;
80
81 "Heat Transfer Rate"
82 q = U * At * (T - Tw);
83
84 "Reaction Heat Rate"
85 qr = (-Hr) * (-rA) * V;
86
87 "Reaction Rate"
88 -rA = k * Ca;
89
90 "Arrhenius Equation"
91 k = ko * exp(-Ea/(R*T));
92
93 "Geometry"
94 A * h = V;
95 At = A + pi*D*h;
96
97 "Valve Equation"
98 Outlet.F = x * Cv * sqrt(h);
99
100 "Perfect Mixture"
101 Outlet.Ca = Ca;
102 Outlet.T = T;
103 end
104
Consistency Analysis
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
CSTR_no_contr
CSTR1
ko
D
A
Ea
R
ro
Cp
U
Hr
pi
At
Cv
T
Tw
V
Ca
h
tau
rA
k
q
qr
Inlet
Ca
F
T
Outlet
Ca
F
T
Results
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

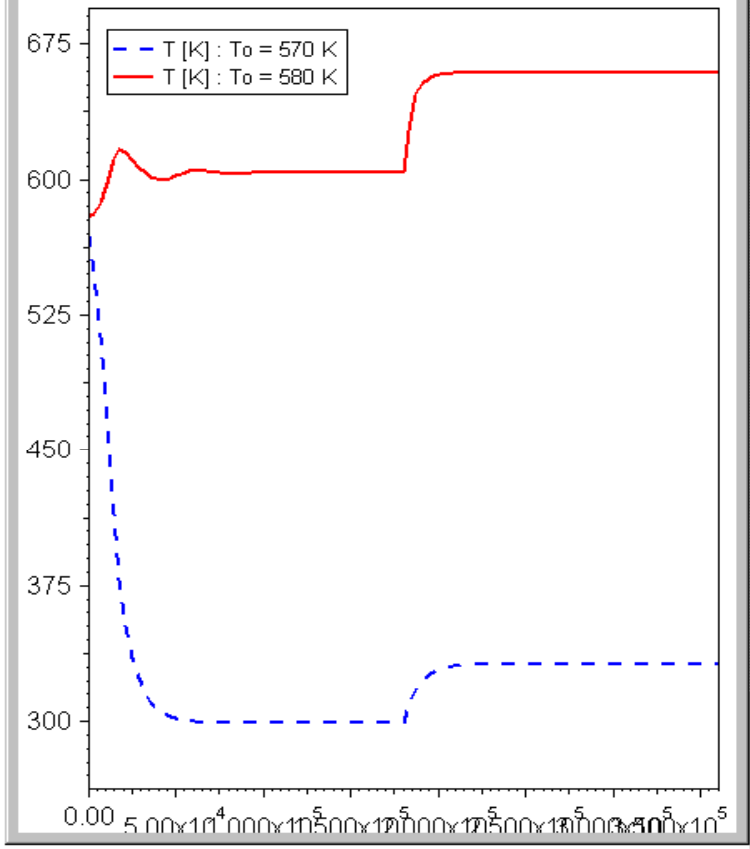
CSTR_no_control 06/

- R
- ro
- Cp
- U
- Hr
- pi
- Cv
- At
- T
- Tw
- x
- v
- Ca
- h
- tau

CSTR_noniso.mso

```
105  # Process with uncontrolled CSTR and multiple steady-states
106  FlowSheet CSTR_no_control
107
108      DEVICES
109      FEED as stream_cstr;
110      CSTR1 as CSTR;
111
112      CONNECTIONS
113      FEED to CSTR1.Inlet;
114
115      SET
116  #   CSTR Parameters
117      CSTR1.R    = 8.3144 * 'kJ/kmol/K';
118      CSTR1.U    = 300 * 'kJ/h/m^2/K';
119      CSTR1.ro   = 1000 * 'kg/m^3';
120      CSTR1.Cp   = 4*'kJ/kg/K';
121      CSTR1.Hr   = -7000 * 'kJ/kmol';
122      CSTR1.Ea   = 6e4 * 'kJ/kmol';
123      CSTR1.ko   = 89 * '1/s';
124      CSTR1.D    = 3.2 * 'm';
125      CSTR1.Cv   = 2.7 * 'm^2.5/h';
126
127      EQUATIONS
128      "Manipulated Variables"
129      CSTR1.x  = 1;
130      CSTR1.Tw  = 300 * 'K';
131
132      "Feed Stream"
133      FEED.Ca = 300 * 'kmol/m^3';
134      FEED.F = 3.5 * 'm^3/h';
135
136  #   Disturbance
137      if time < 50 * 'h' then
138          "Feed Temperature" FEED.T = 300 * 'K';
139      else
140          "Feed Temperature" FEED.T = 350 * 'K';
141      end
142
143      INITIAL
144      "Concentration" CSTR1.Ca = 50 * 'kmol/m^3';
145      "Level" CSTR1.h = 1.7 * 'm';
146      "Temperature" CSTR1.T = 570 * 'K'; # increase to 580 K to change steady-state
147
148      OPTIONS
149      TimeStep = 1;
150      TimeEnd = 100;
151      TimeUnit = 'h';
152      DAESolver(File = "dassl");
153  end
```

Plot2D (3)

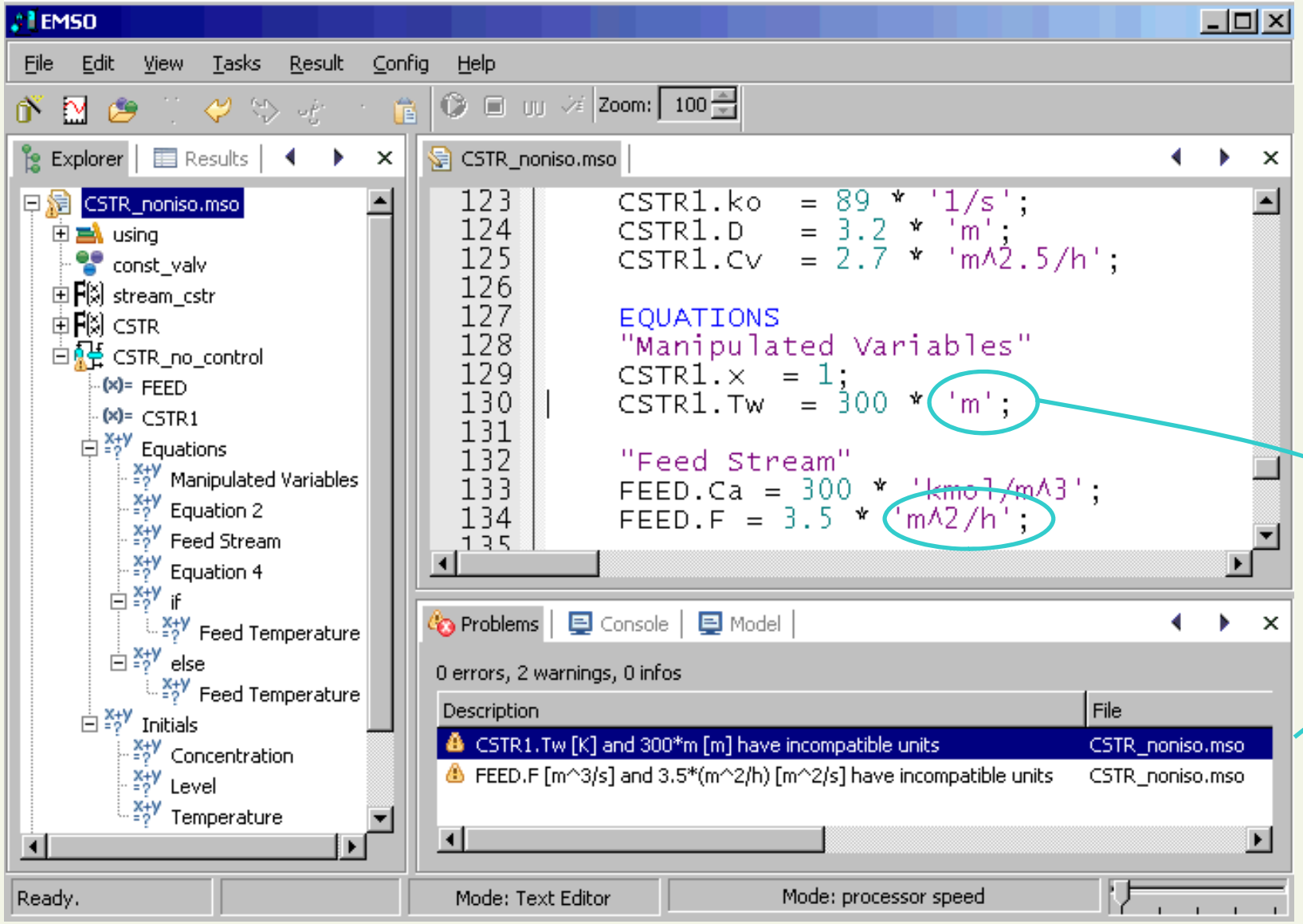


Ready.

Mode: Text Editor

Mode: processor speed

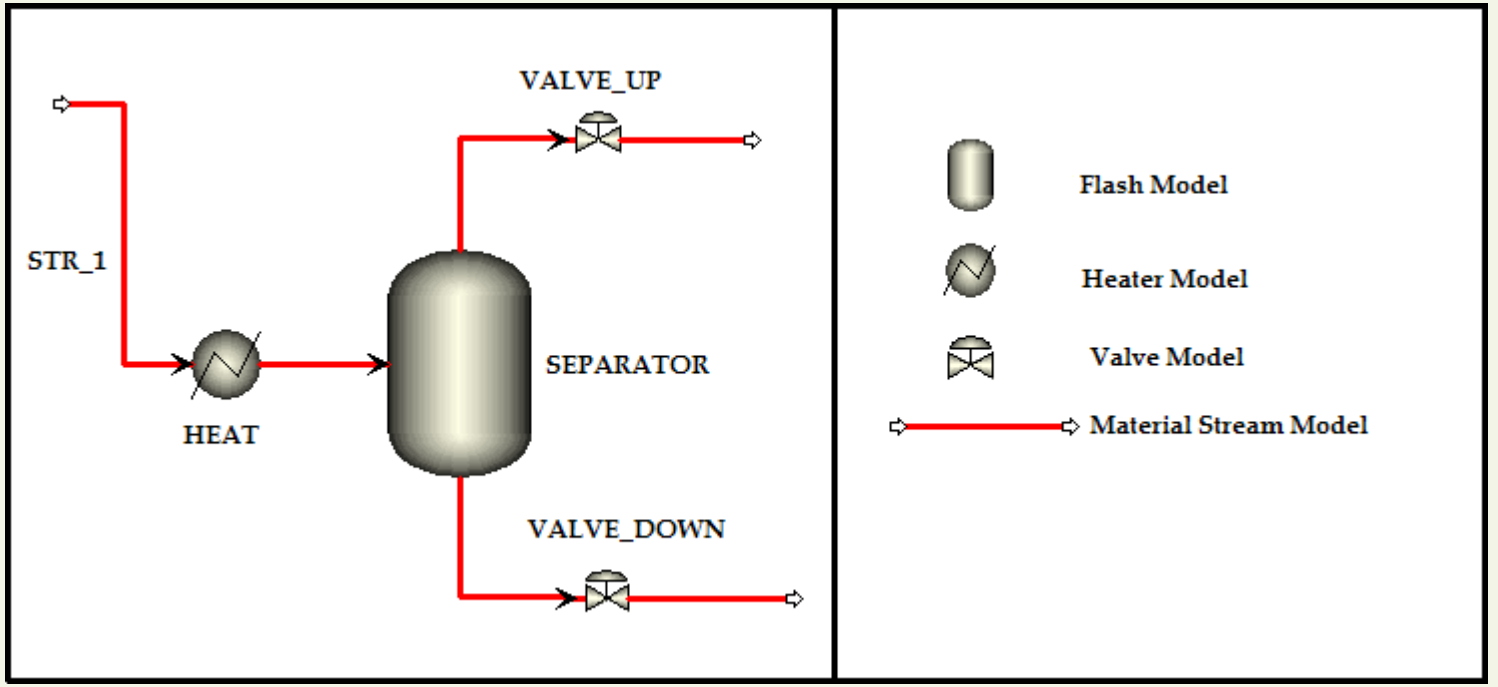
Building Dynamic Models
– Checking Units of Measurement –



incompatible units

3. EMSO Tutorial
– Modeling Structure –

EMSO has 3 main entities in the modeling structure

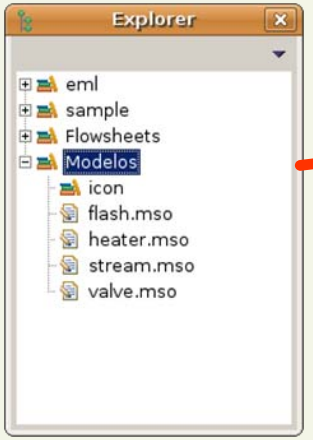


FlowSheet – process model, is composed by a set of DEVICES

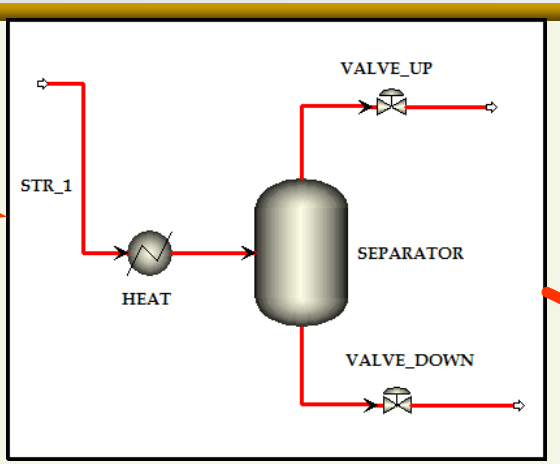
DEVICES – components of a FlowSheet, an unit operation or an equipment

Model – mathematical description of a DEVICE

EMSO Tutorial
– Modeling Structure –



Model



FlowSheet

Model: equation-based

heater.mso

```
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=0.4782);

 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;
```

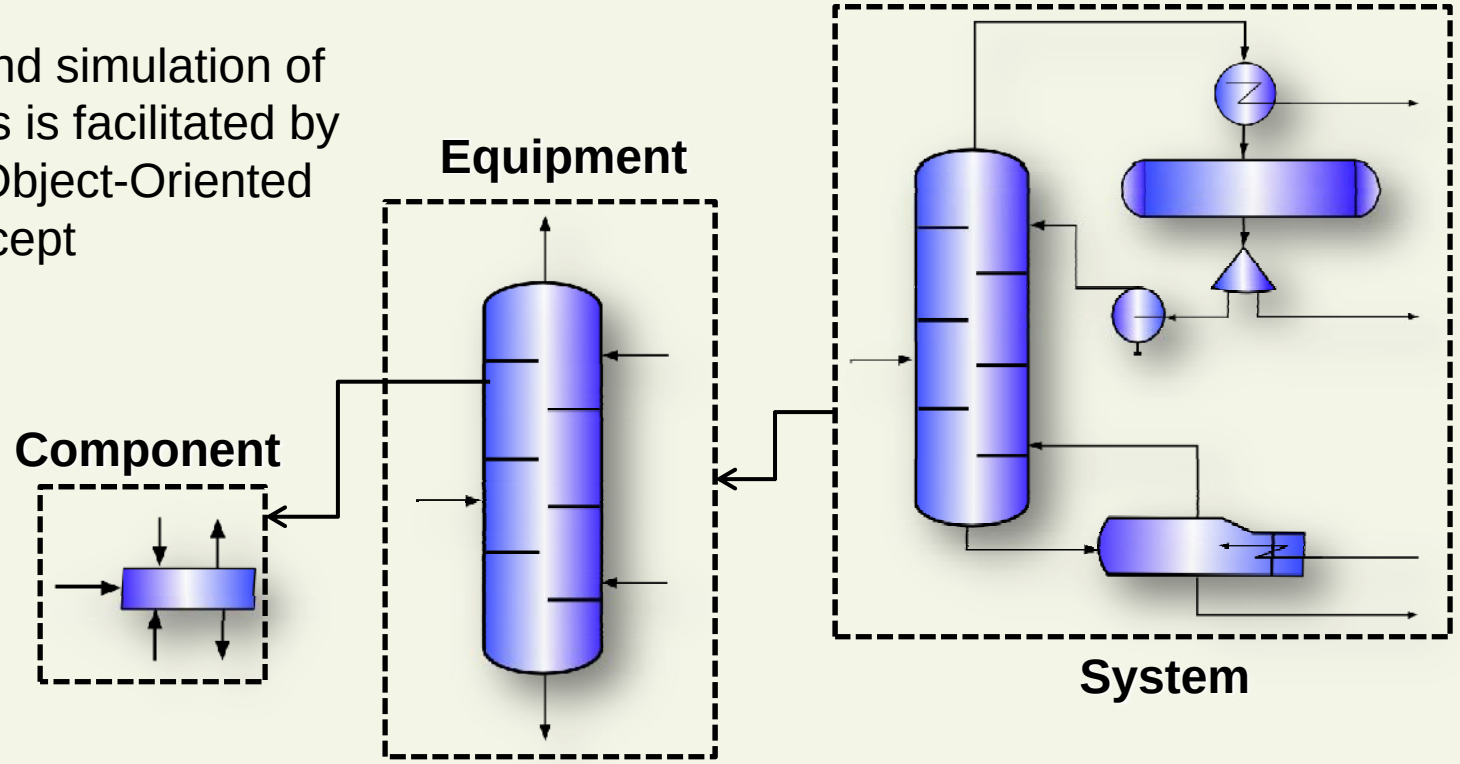
FlowSheet: component-based

FlowSheet.mso

```
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;
```

EMSO Tutorial
– Object-Oriented Modeling Language –

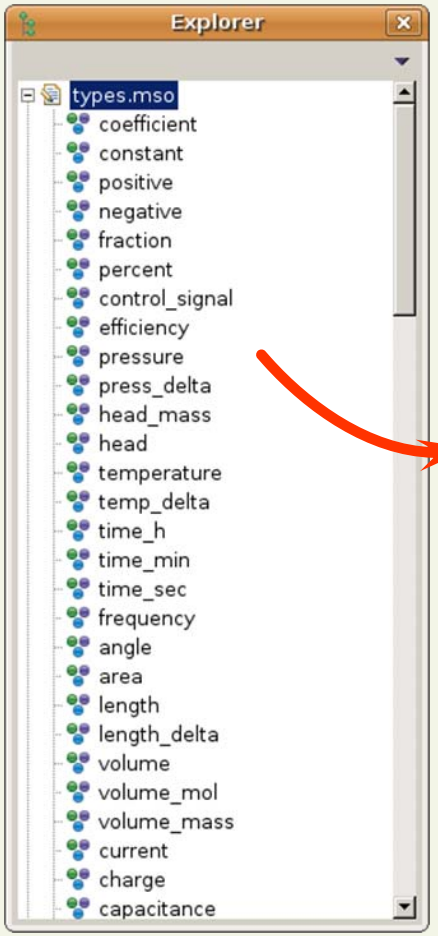
The modeling and simulation of complex systems is facilitated by the use of the Object-Oriented concept



The system can be decomposed in several components, each one described separately using its constitutive equations

The components of the system exchange information through the connecting ports

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**

types.mso

```
efficiency as Real (Brief = "Efficiency", Default=0.5, final Lower=0, final Upper=1);

# Pressure
pressure      as Real        (Brief = "Pressure", Default=1, Lower=1e-30, Upper=5e7, final Unit = 'atm');
press_delta   as pressure    (Brief = "Pressure Difference", Default=0.01, Lower=-5e6);
head_mass     as Real        (Brief = "Head", Default=50, Lower=-1e6, Upper=1e6, final Unit = 'kJ/kg');
head          as Real        (Brief = "Head", Default=50, Lower=-1e6, Upper=1e6, final Unit = 'kJ/kmol');

# Temperature
temperature   as Real          (Brief = "Temperature", Default=300, Lower=27, Upper=5273, final Unit = 'K');
temp_delta    as temperature   (Brief = "Temperature Difference", Default=30, Lower=-1000, Upper=1000);

# Time
time_h      as positive  (Brief = "Time in hours", Default=1, Upper=1e4, final Unit = 'h');
time_min    as time_h    (Brief = "Time in minutes", DisplayUnit ='min');
time_sec    as time_h    (Brief = "Time in seconds", DisplayUnit = 's');
frequency   as positive  (Brief = "Frequency", Default=1, Upper=100, final Unit='1/s');

# Size related
angle         as Real     (Brief = "Angle", Default=0, Lower=-7, Upper=7, final Unit ='rad');
area          as positive (Brief = "Area", Default=1, Upper=1E6, final Unit ='m^2');
length        as positive (Brief = "Length", Default=1, Upper=5e6, final Unit = 'm');
length_delta  as length   (Brief = "Difference of Length", Lower=-1000);
volume        as positive (Brief = "Volume", Default=10, Upper=1000, final Unit = 'm^3');
volume_mol    as positive (Brief = "Molar Volume", Default=10, Upper=1E6, final Unit = 'm^3/mol');
volume_mass as positive   (Brief = "Specific Volume", Default=10, Upper=1E30, final Unit = 'm^3/kg');
```

EMSO Tutorial
– Model Components –

Including sub-models and types

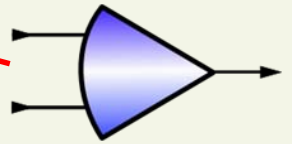
Automatic model documentation

Basic sections to create a math. model

Input and output connections

Symbol of variable in LaTeX command for documentation

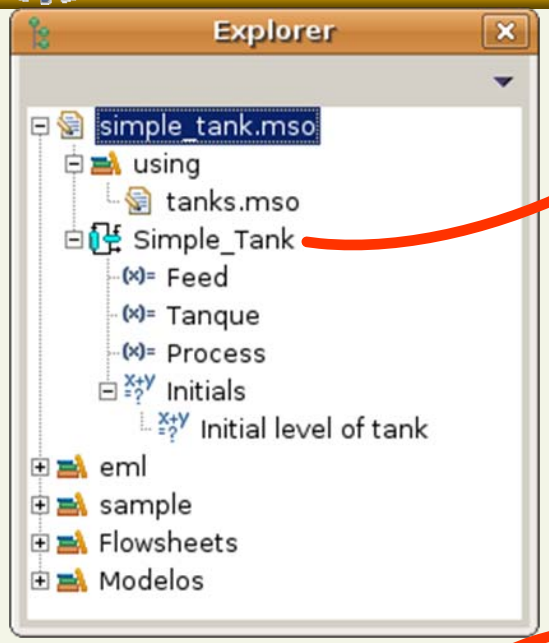
Port location to draw a flowsheet connection

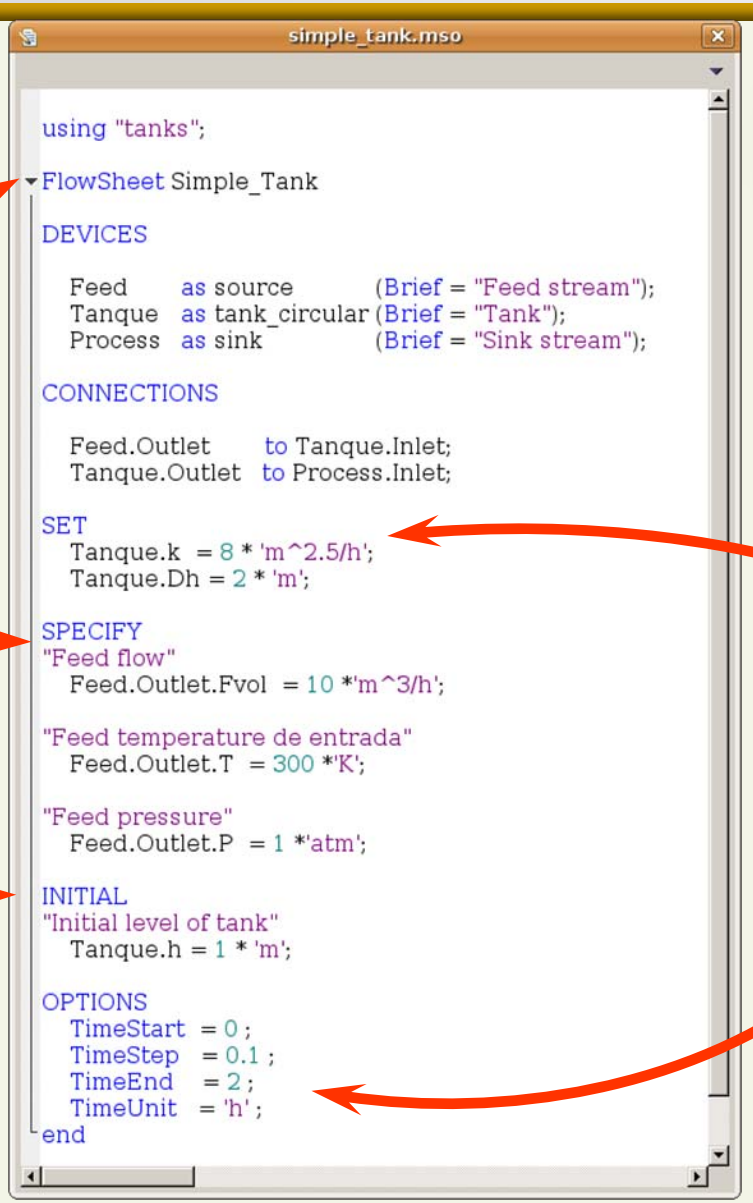


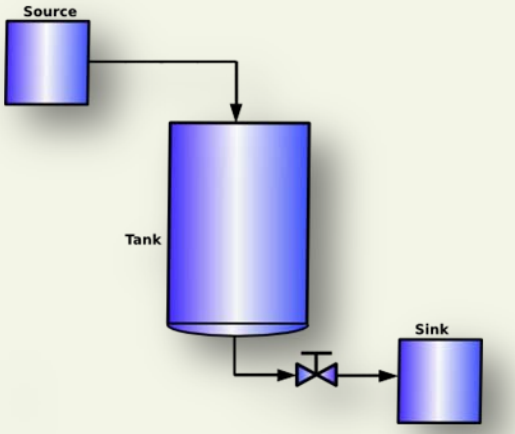
mixer.mso

```
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
```

EMSO Tutorial
– FlowSheet Components –







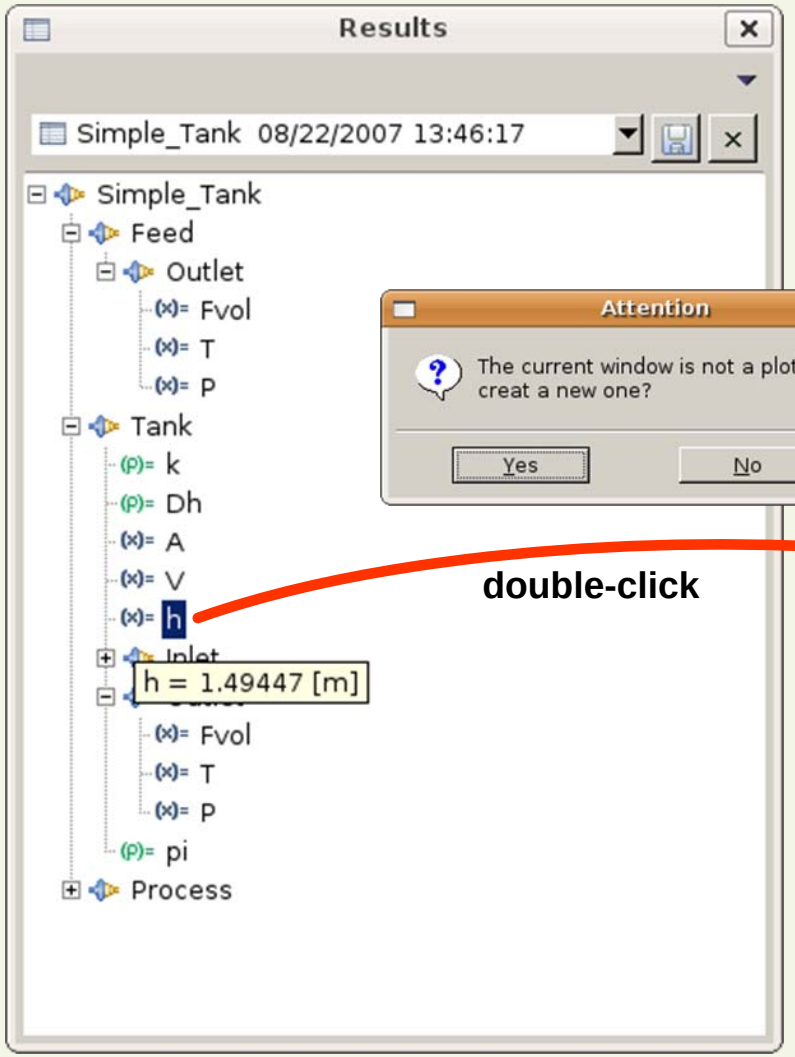
Parameters of DEVICES

Degree of Freedom

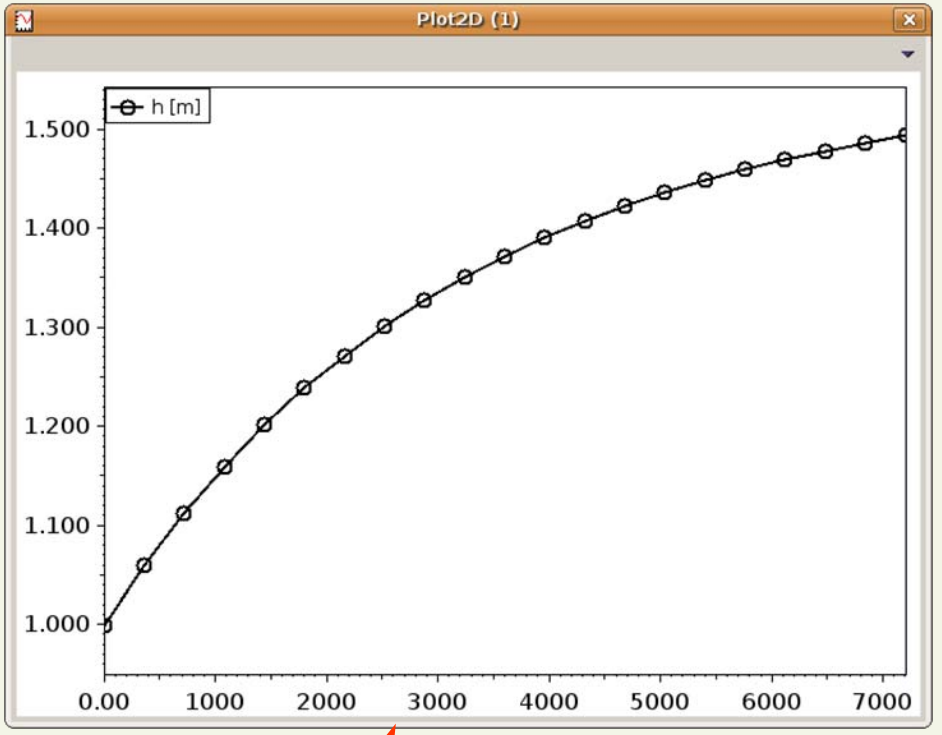
Simulation options

Dynamic Degree of Freedom

EMSO Tutorial
– Simulation Results: graphics –



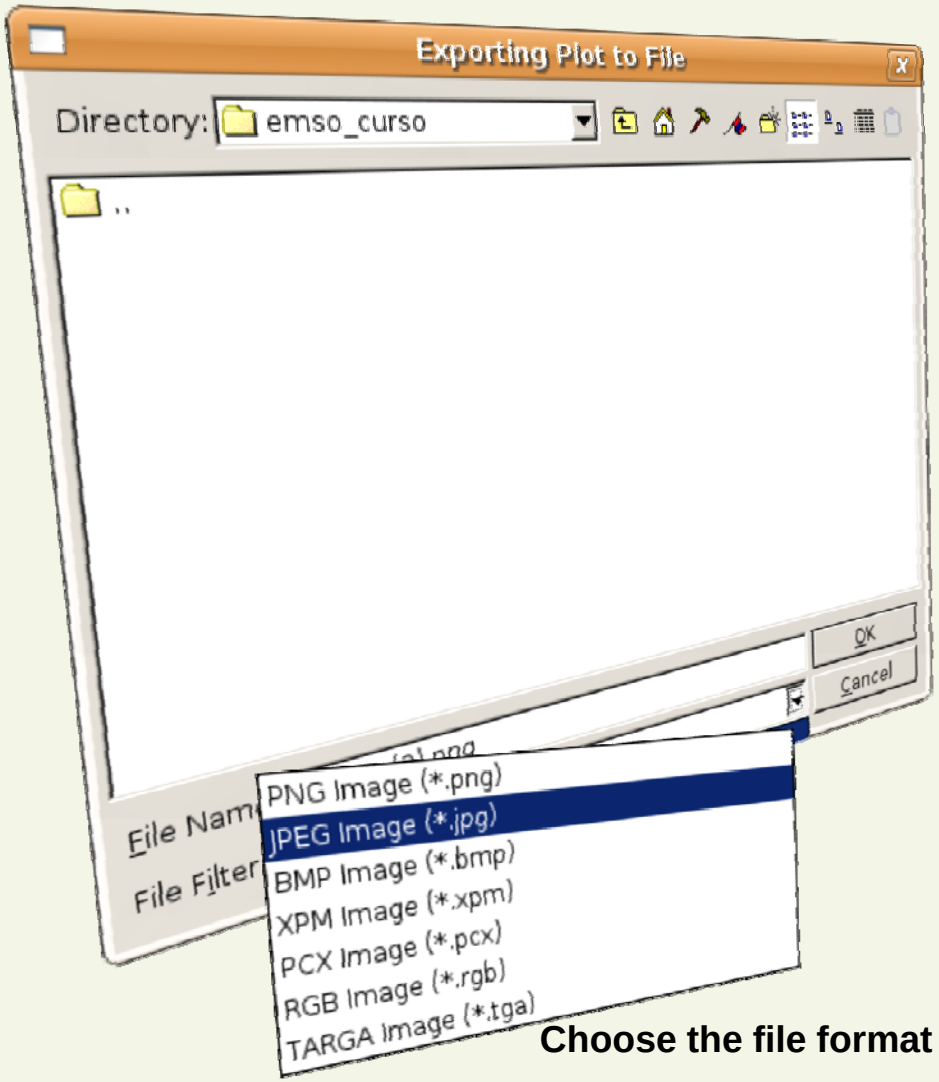
double-click



Horizontal axis is always the independent variable (usually time)

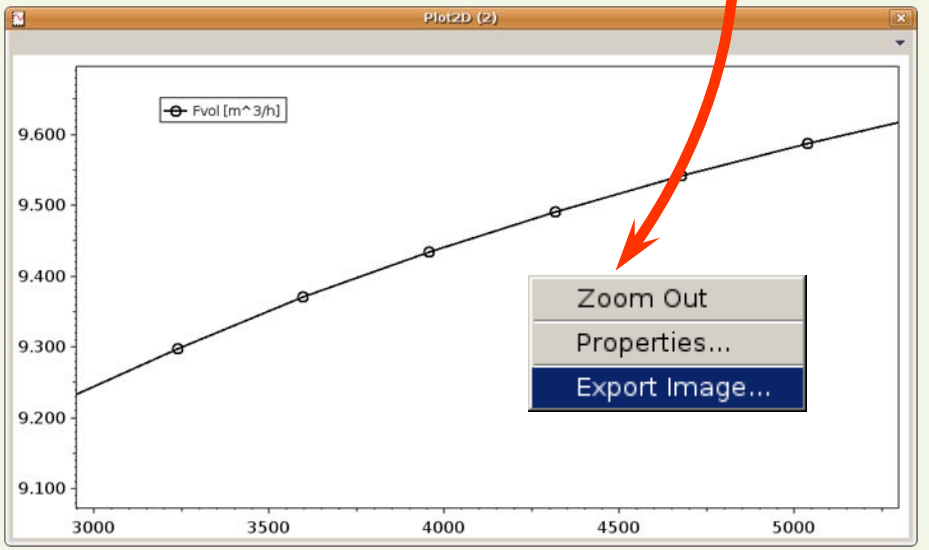
EMSO Tutorial

– Simulation Results: exporting graphics –

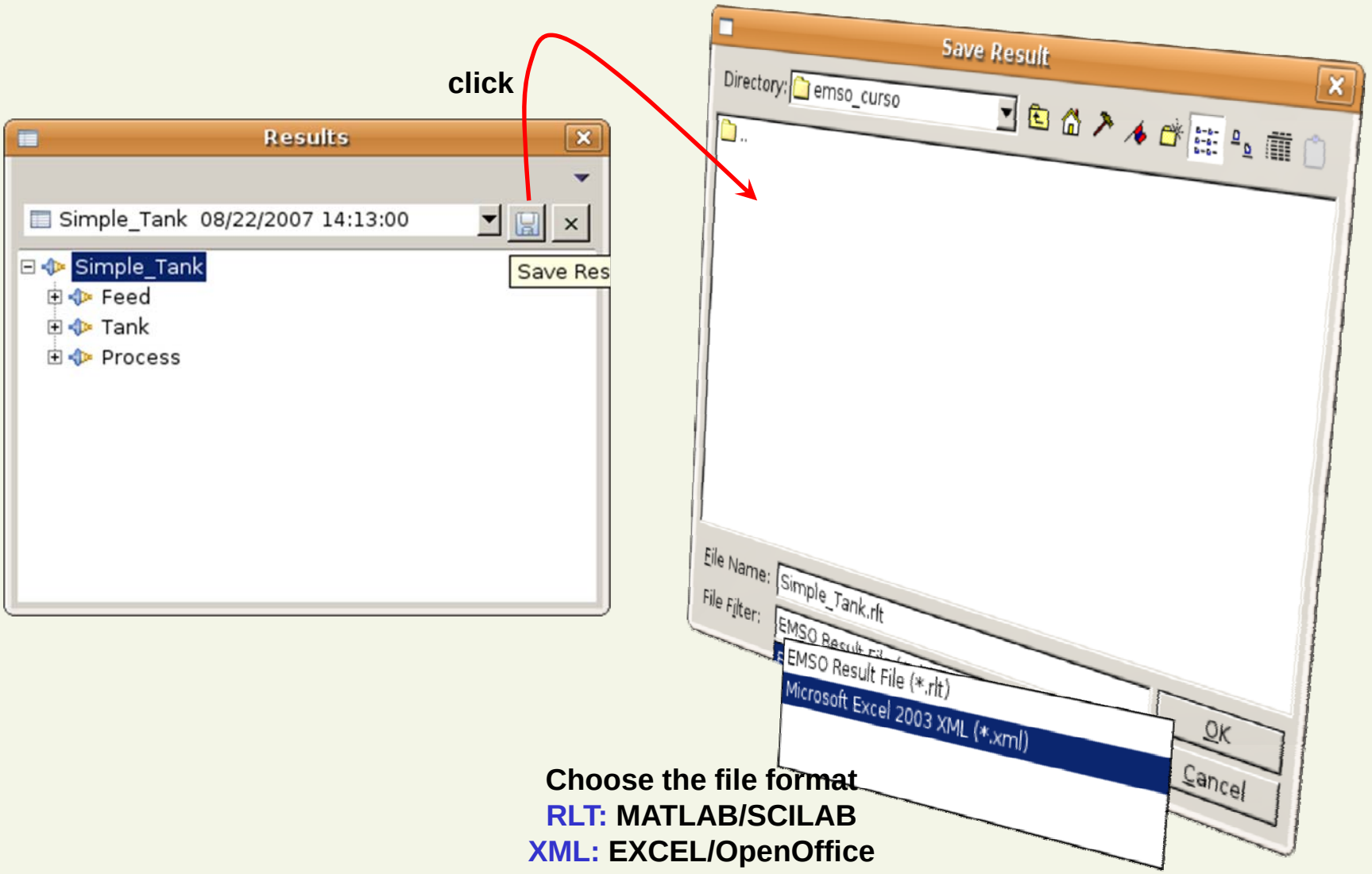


Choose the file format

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



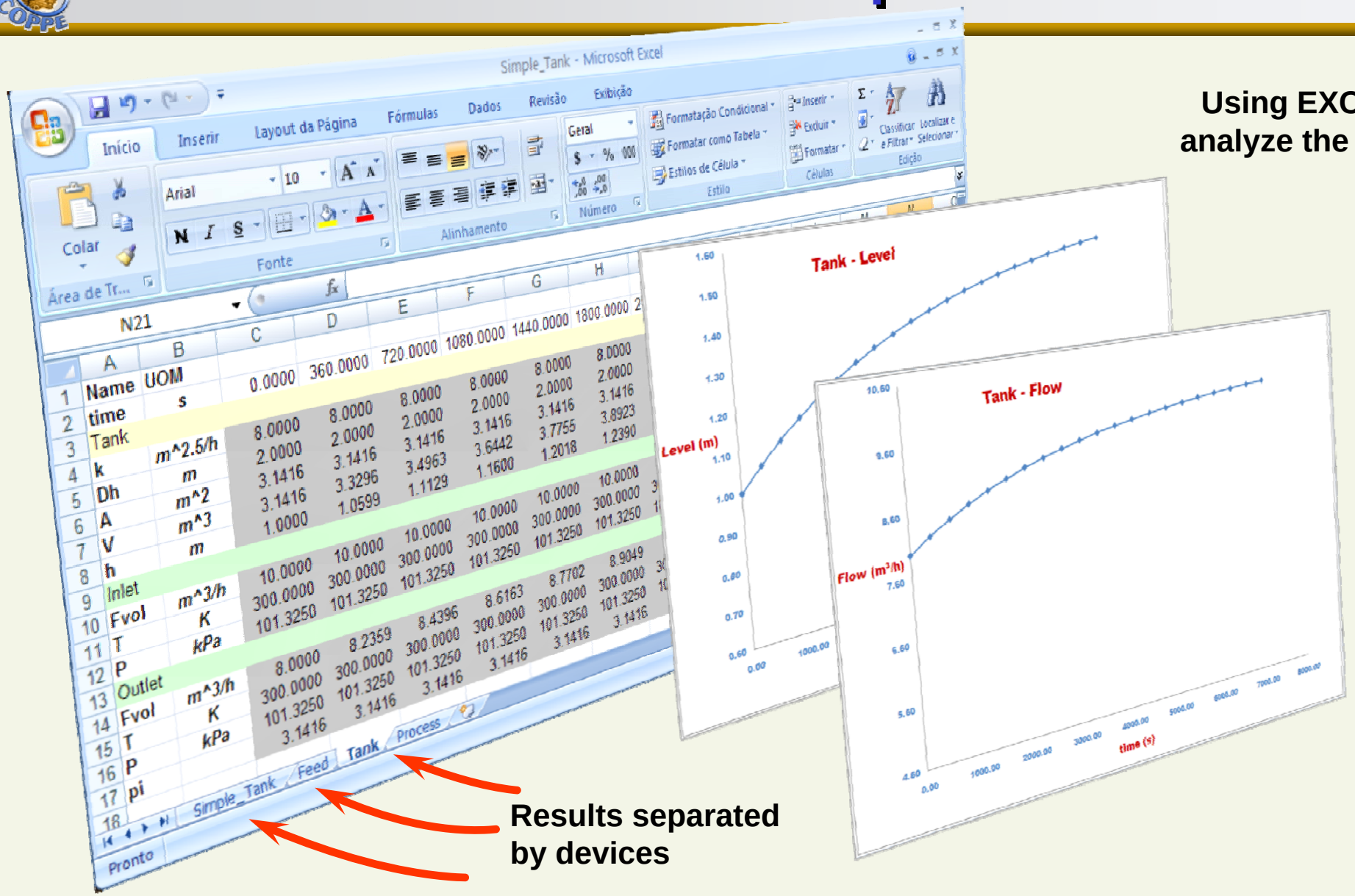
EMSO Tutorial
– Simulation Results: exporting data –



Choose the file format
RLT: MATLAB/SCILAB
XML: EXCEL/OpenOffice

EMSO Tutorial

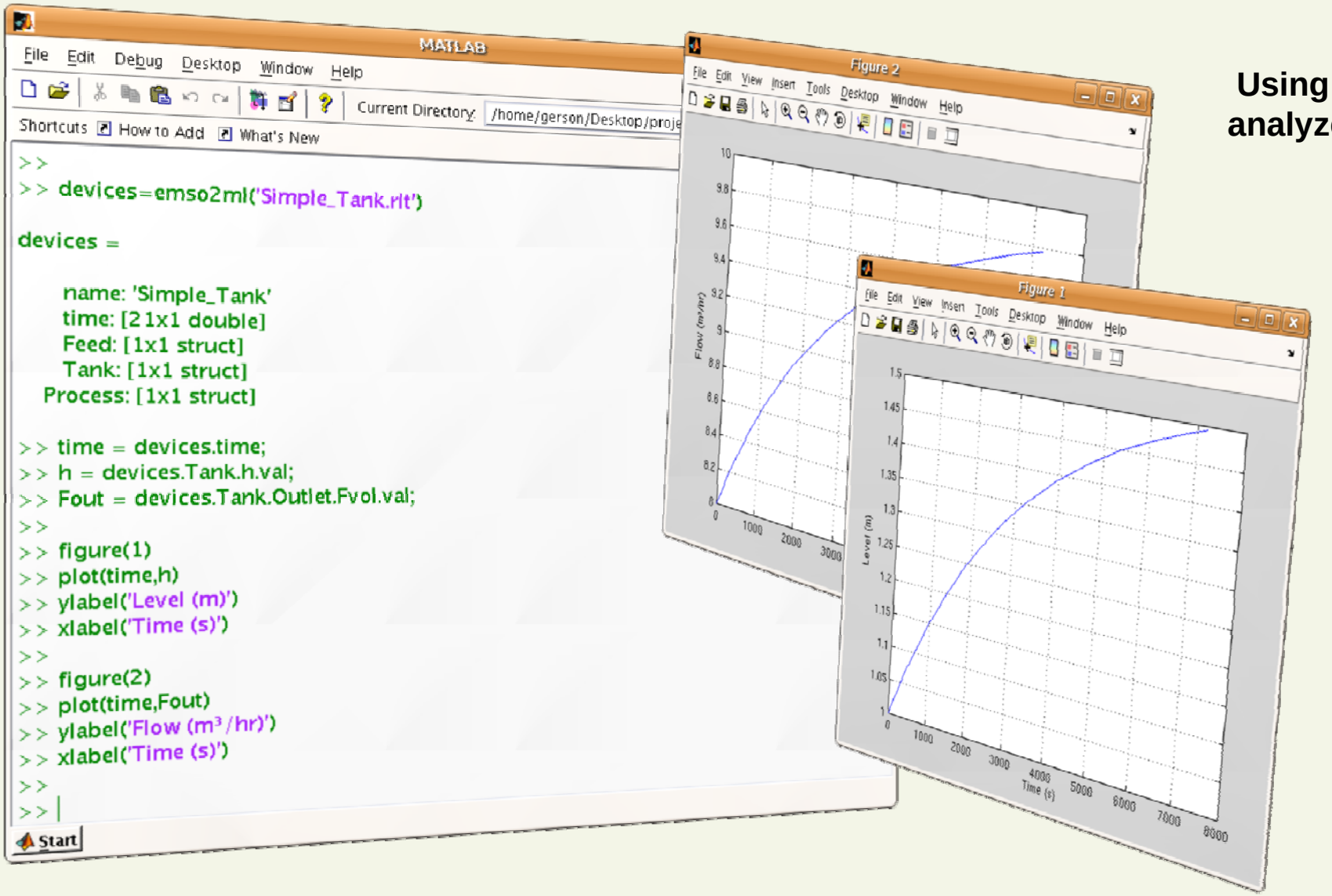
– Simulation Results: in spreadsheets –



Using EXCEL to analyze the results

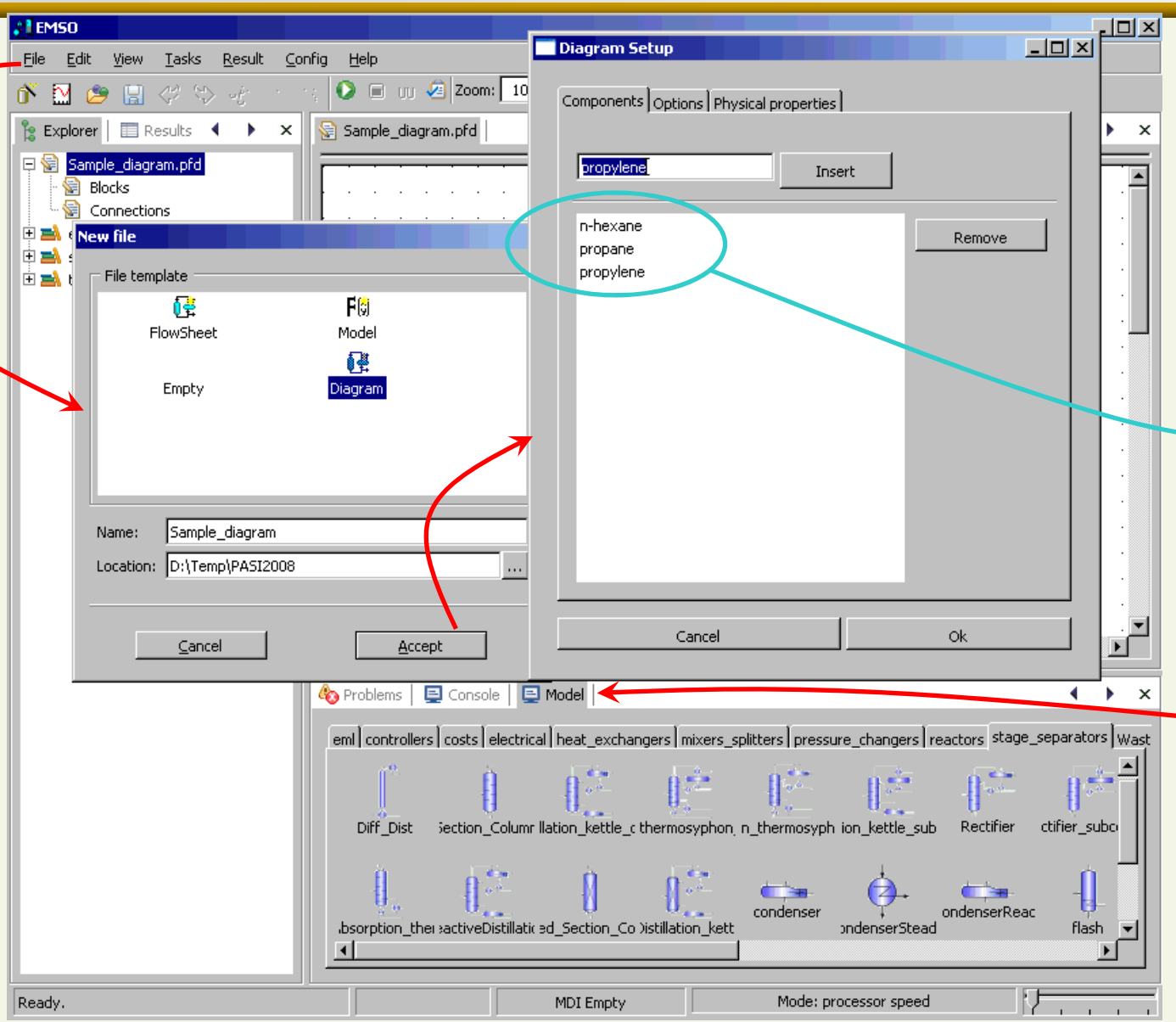
Results separated by devices

EMSO Tutorial
– Simulation Results: in MATLAB/SCILAB –



Using MATLAB to analyze the results

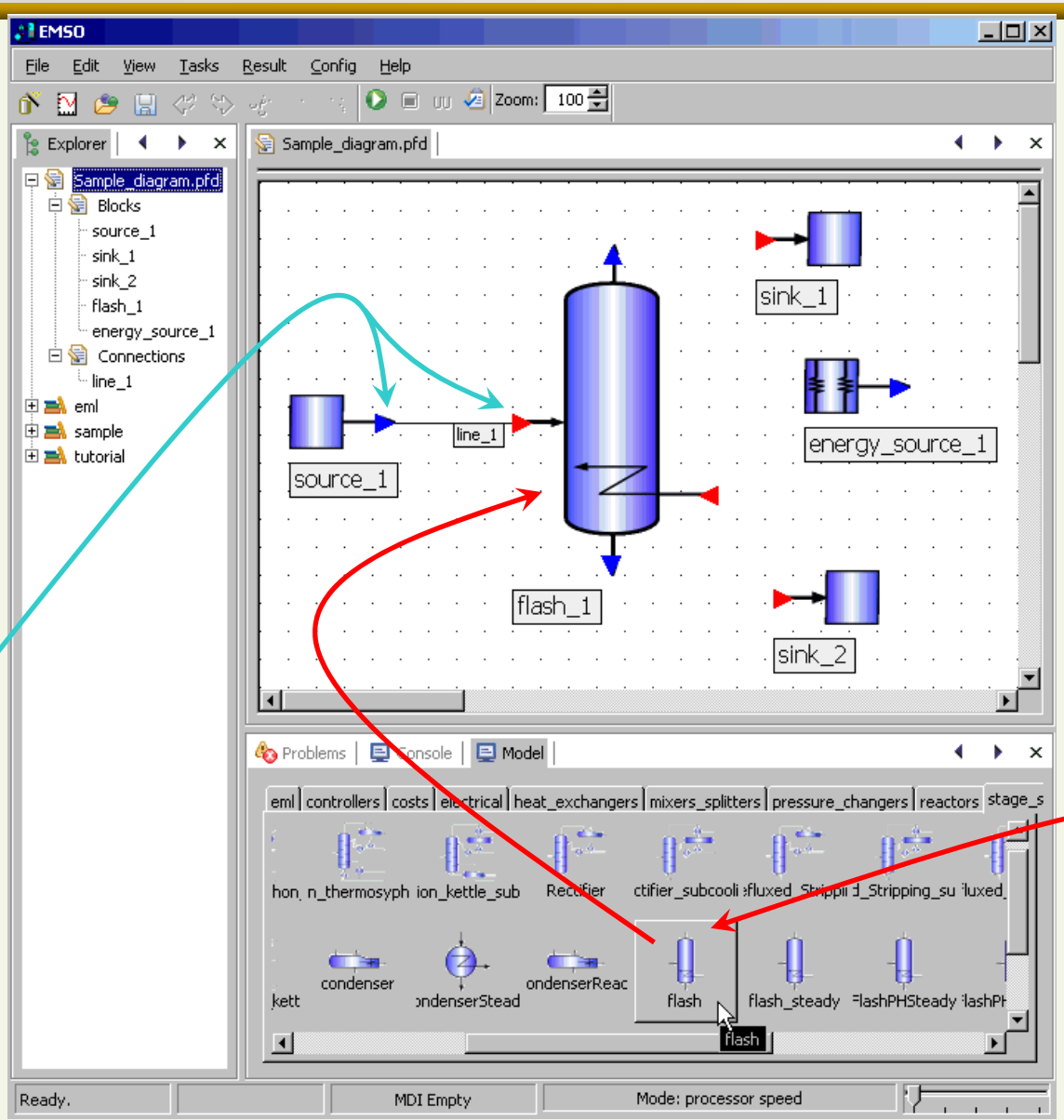
EMSO Tutorial
– Building Block Diagrams: create file –



Selected components from physical properties package

Devices found in the model library

EMSO Tutorial
– Building Block Diagrams: select devices –

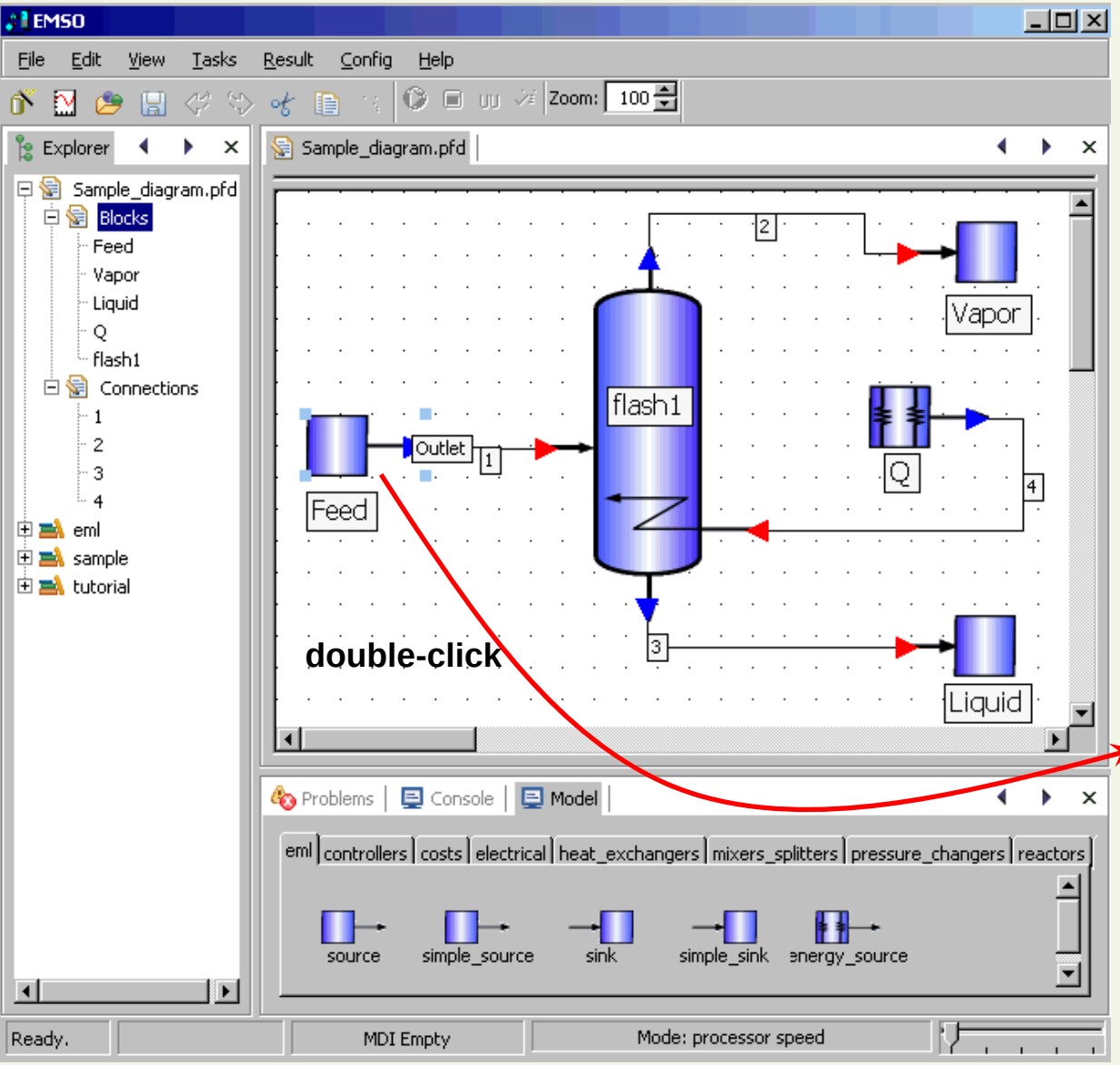


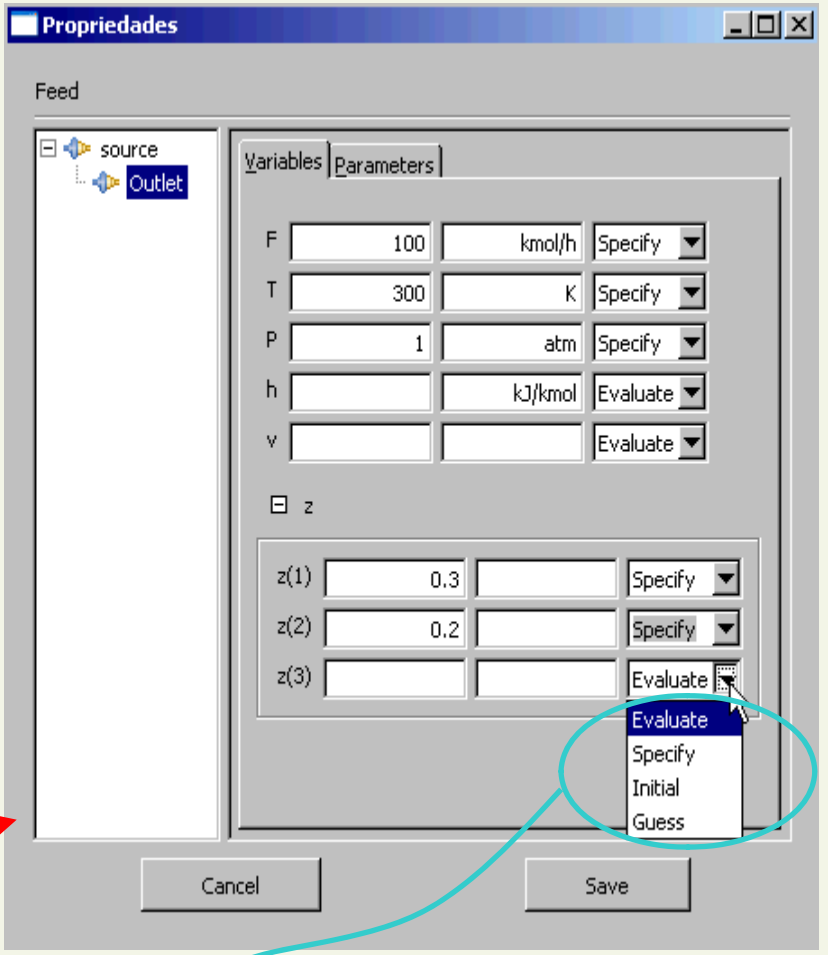
drag & drop ports to create a connection

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

click to create a device

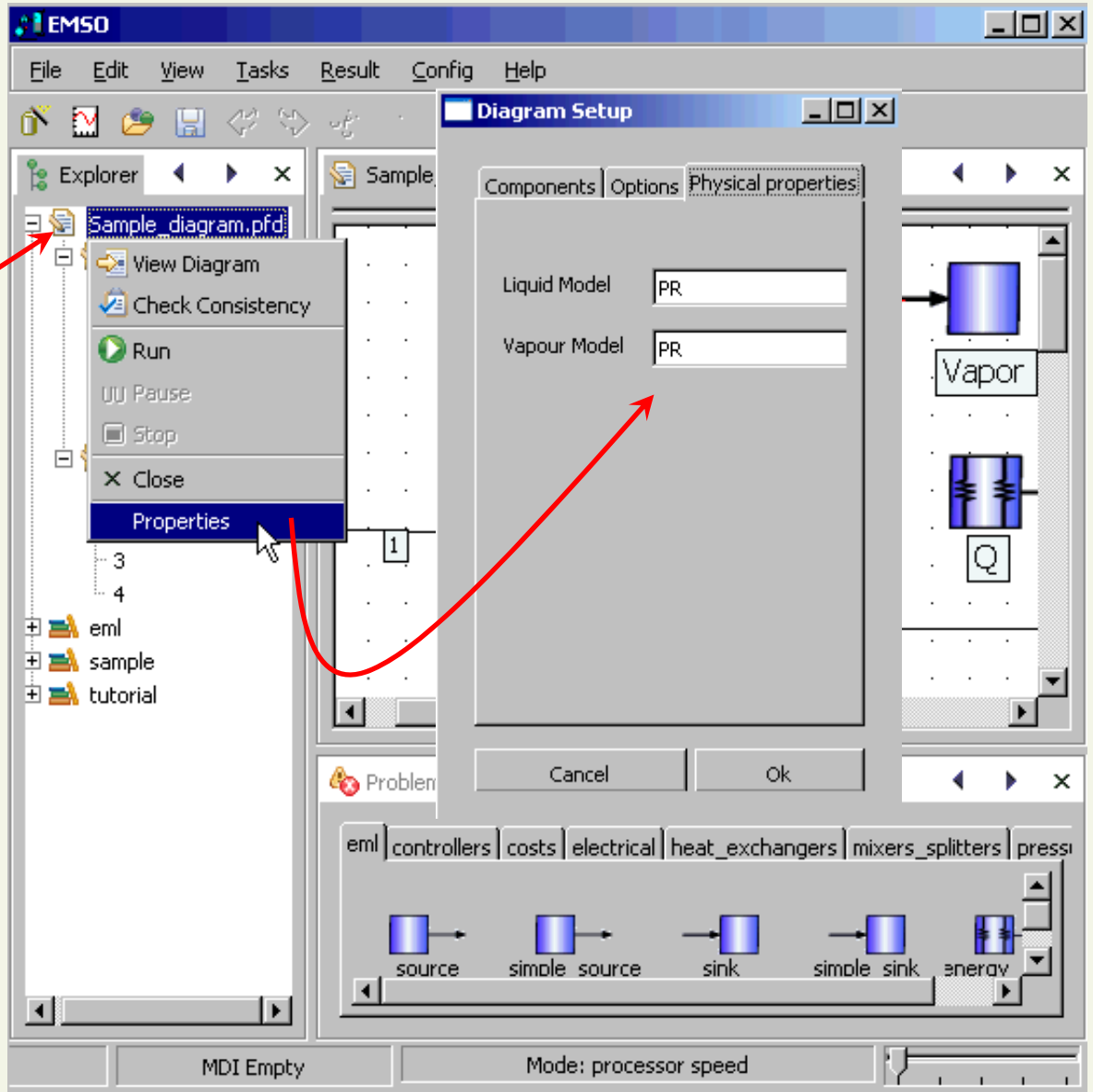
EMSO Tutorial
– Building Block Diagrams: set case study –





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

EMSO Tutorial
– Building Block Diagrams: thermodynamic –



right-click

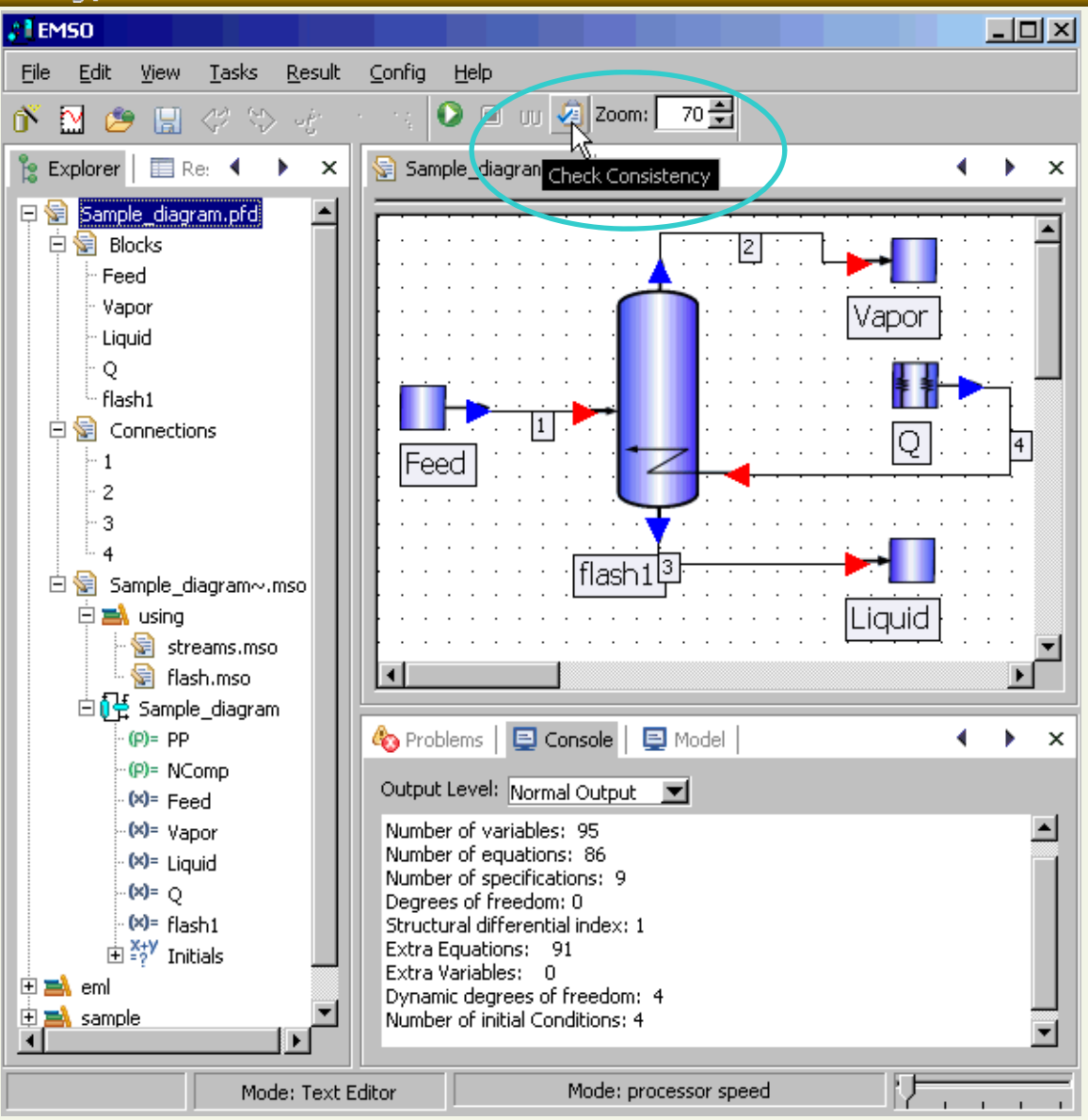
Available models

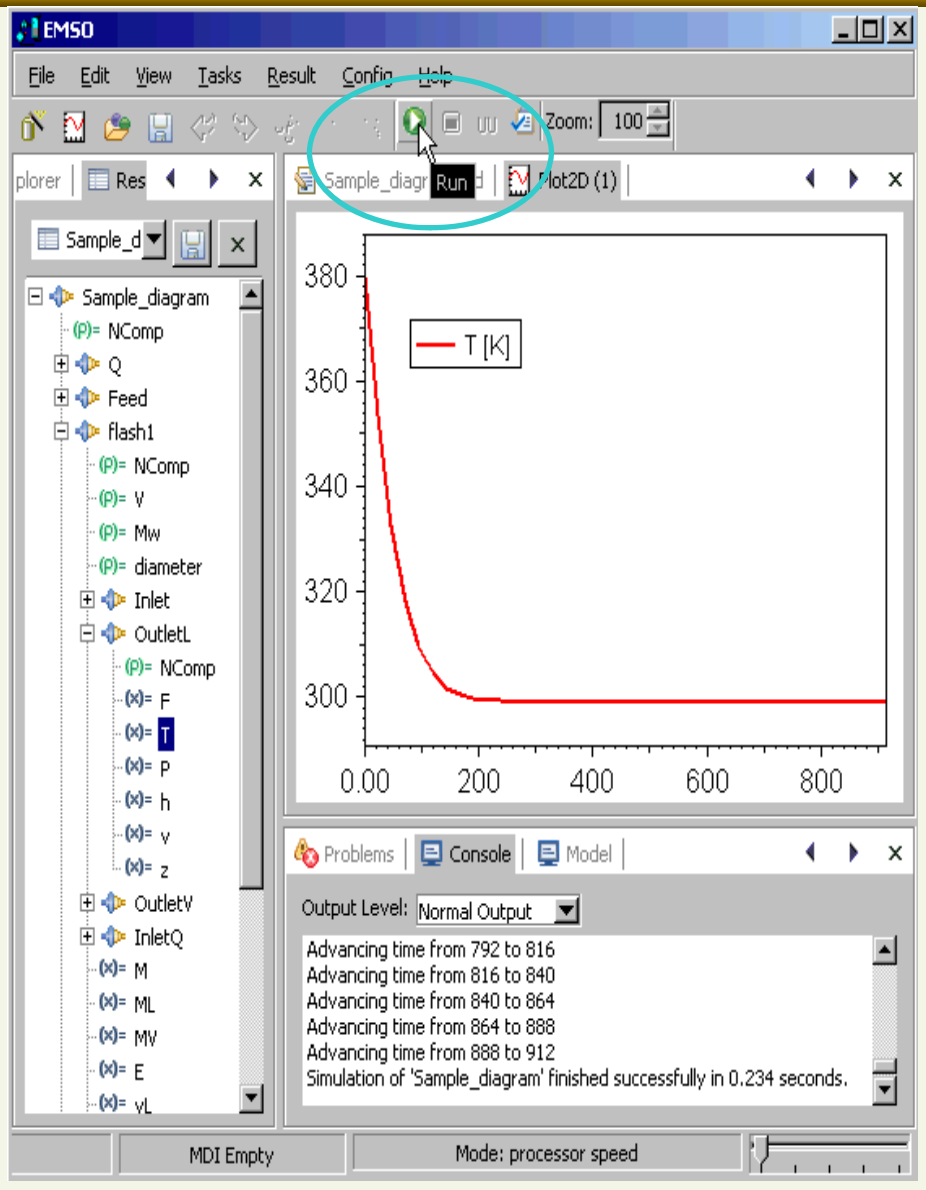
EOS Name	Description
Ideal	Ideal Gas, valid only for vapour models
IdealLiquid	Ideal Liquid, valid only for liquid models
RK	Redlich-Kwong
SRK	Soave-Redlich-Kwong
PR	Peng-Robinson
APR	Assymetric-PR
ASRK	Assymetric-RK
UNIFAC	UNIFAC (Dortmund)

In development:
PC-SAFT

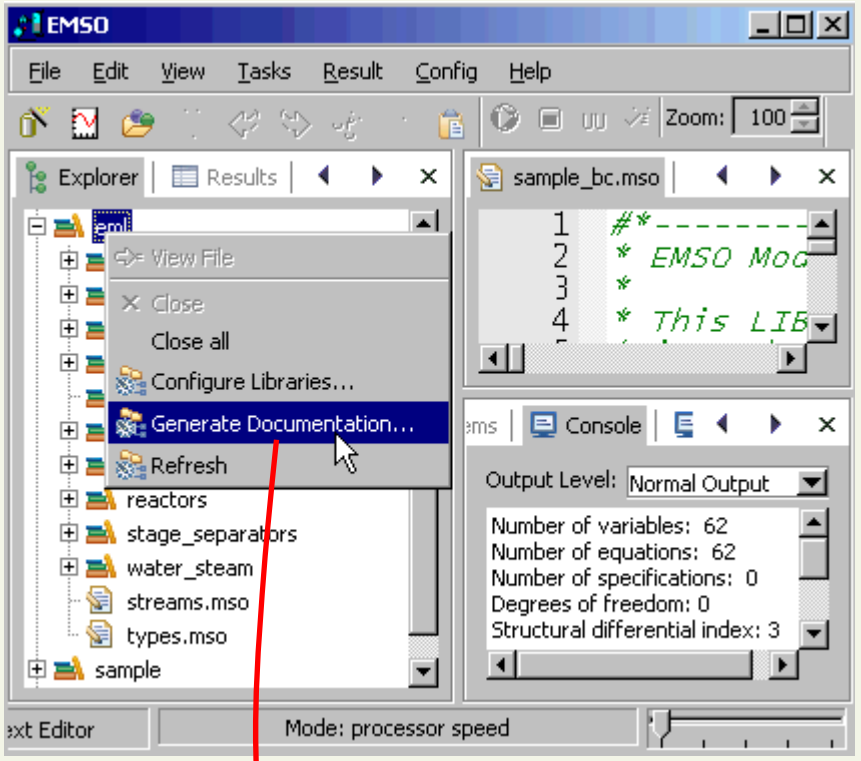
EMSO Tutorial

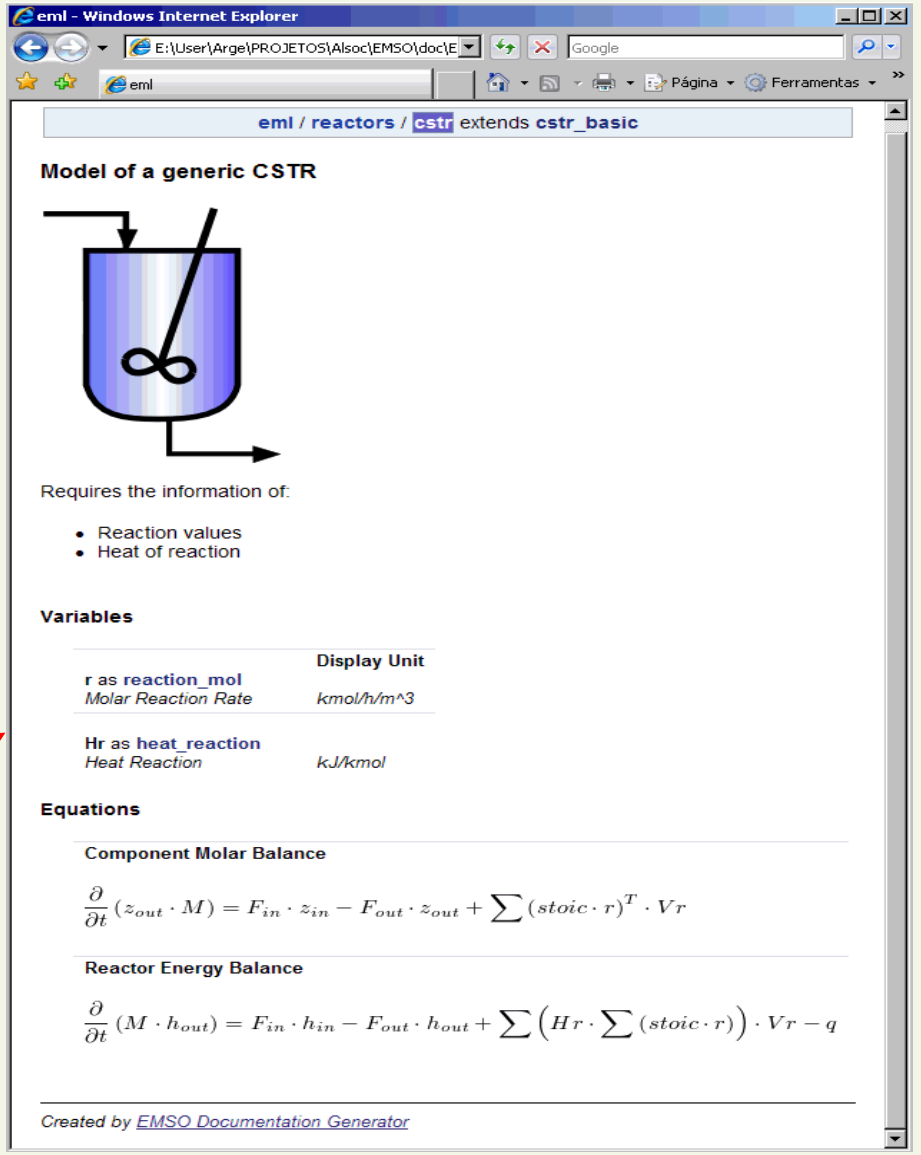
– Building Block Diagrams: simulating –





EMSO Tutorial
– Automatic Documentation –





Note: LaTeX must be installed.

4. 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: $F(t, y, y') = 0$

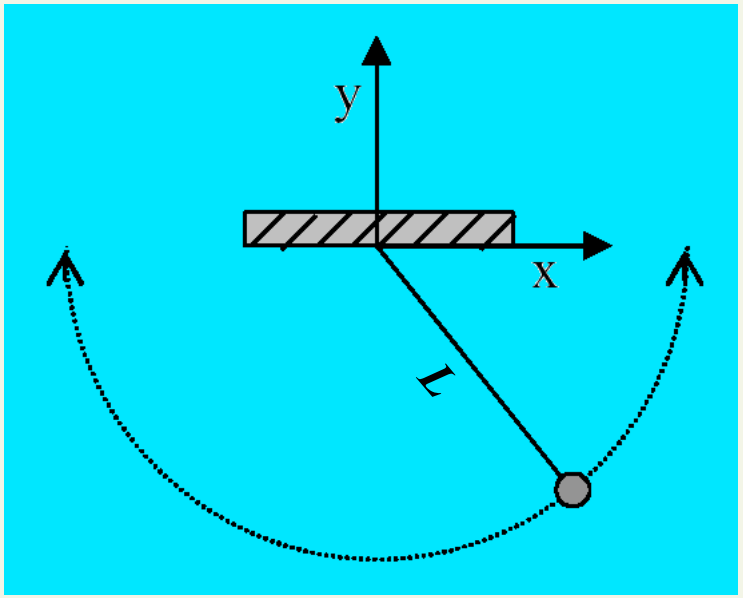
The Dynamic Degree of Freedom (DDoF) is the number of variables in $y(t_0)$ that can be assigned arbitrarily to compute a set of consistent initial conditions $\{y(t_0), y'(t_0)\}$ of the DAE system. Is the true number of states of the system (or the system order of the DAE). Is the number of initial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

Example: classical pendulum problem



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

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

Inconsistent initial condition:

$$x(0) = 1 \quad x'(0) = 0$$
$$y(0) = 0 \quad y'(0) = 0$$
$$w(0) = 0 \quad w'(0) = 1$$
$$z(0) = 0 \quad z'(0) = -g$$
$$T(0) = 1 \quad T'(0) = 0$$

OK!

$$F(0, y(0), y'(0)) = 0$$

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) $\longrightarrow$ $T'(0) = -3\, g \cdot z(0) / L^2$

Dynamic Degree of Freedom
– High-Index DAE System –

Example: classical pendulum problem

$$x' = w \quad (1)$$
$$y' = z \quad (2)$$
$$w' = T.x \quad (3)$$
$$z' = T.y - g \quad (4)$$
$$x^2 + y^2 = L^2 \quad (5)$$
$$x \cdot w + y \cdot z = 0 \quad (6)$$
$$w^2 + z^2 + T \cdot L^2 = g \cdot y \quad (7)$$
$$T' = -3\,g \cdot z / L^2 \quad (8)$$

10 variables (y, y´)

8 equations

2 DDoF

But not any pair!

Index 3

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

Index 2

$$x' = w \quad (1)$$
$$y' = z \quad (2)$$
$$w' = T.x \quad (3)$$
$$z' = T.y - g \quad (4)$$
$$x \cdot w + y \cdot z = 0 \quad (6)$$

Index 1

$$x' = w \quad (1)$$
$$y' = z \quad (2)$$
$$w' = T.x \quad (3)$$
$$z' = T.y - g \quad (4)$$
$$w^2 + z^2 + T \cdot L^2 = g \cdot y \quad (7)$$

Index 0

$$x' = w \quad (1)$$
$$y' = z \quad (2)$$
$$w' = T.x \quad (3)$$
$$z' = T.y - g \quad (4)$$
$$T' = -3\,g \cdot z / L^2 \quad (8)$$

Satisfies the inconsistent I.C.

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, MEBDFI, DASSLC)

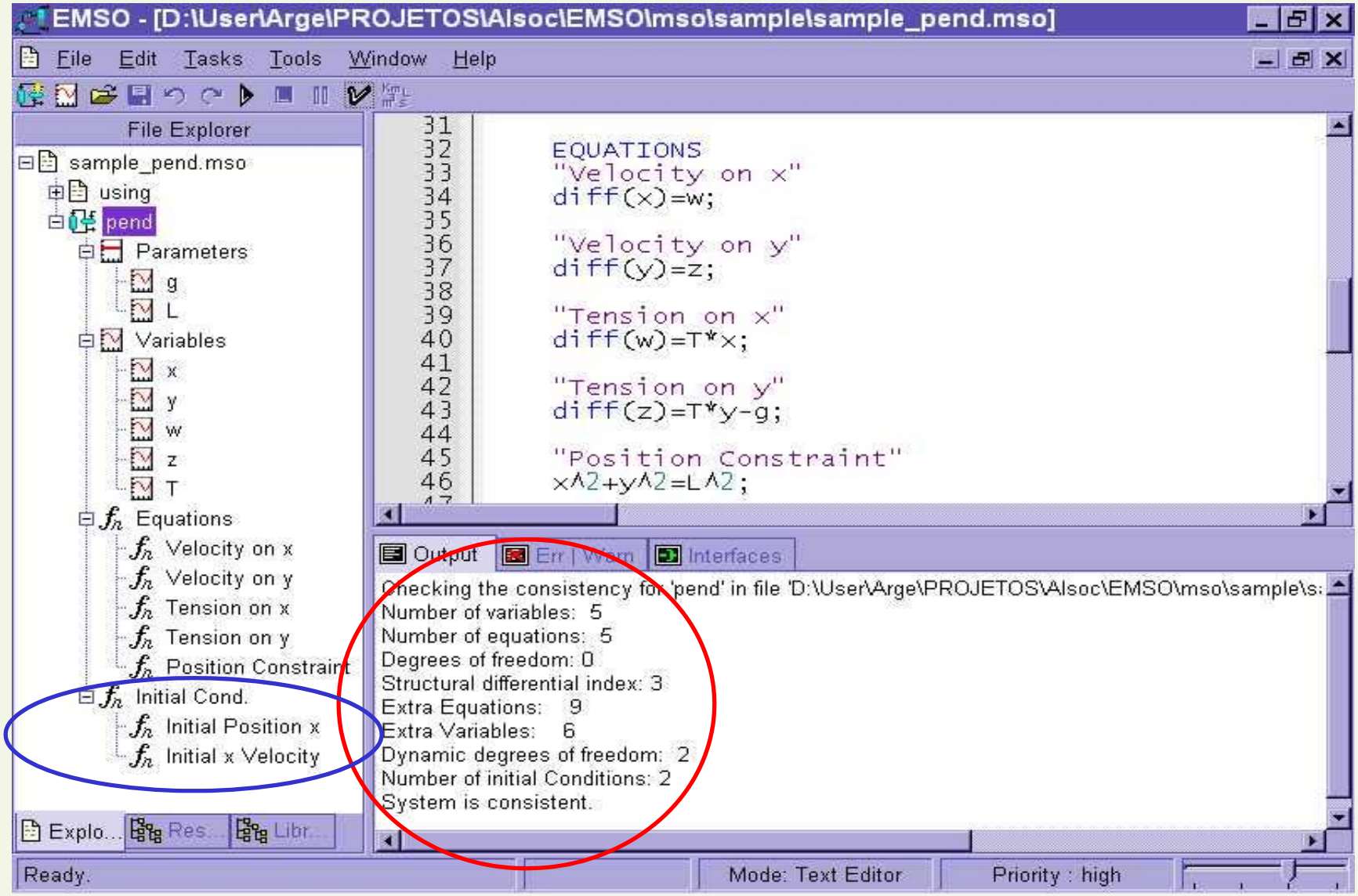
3) Apply automatic index reduction algorithms

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 –

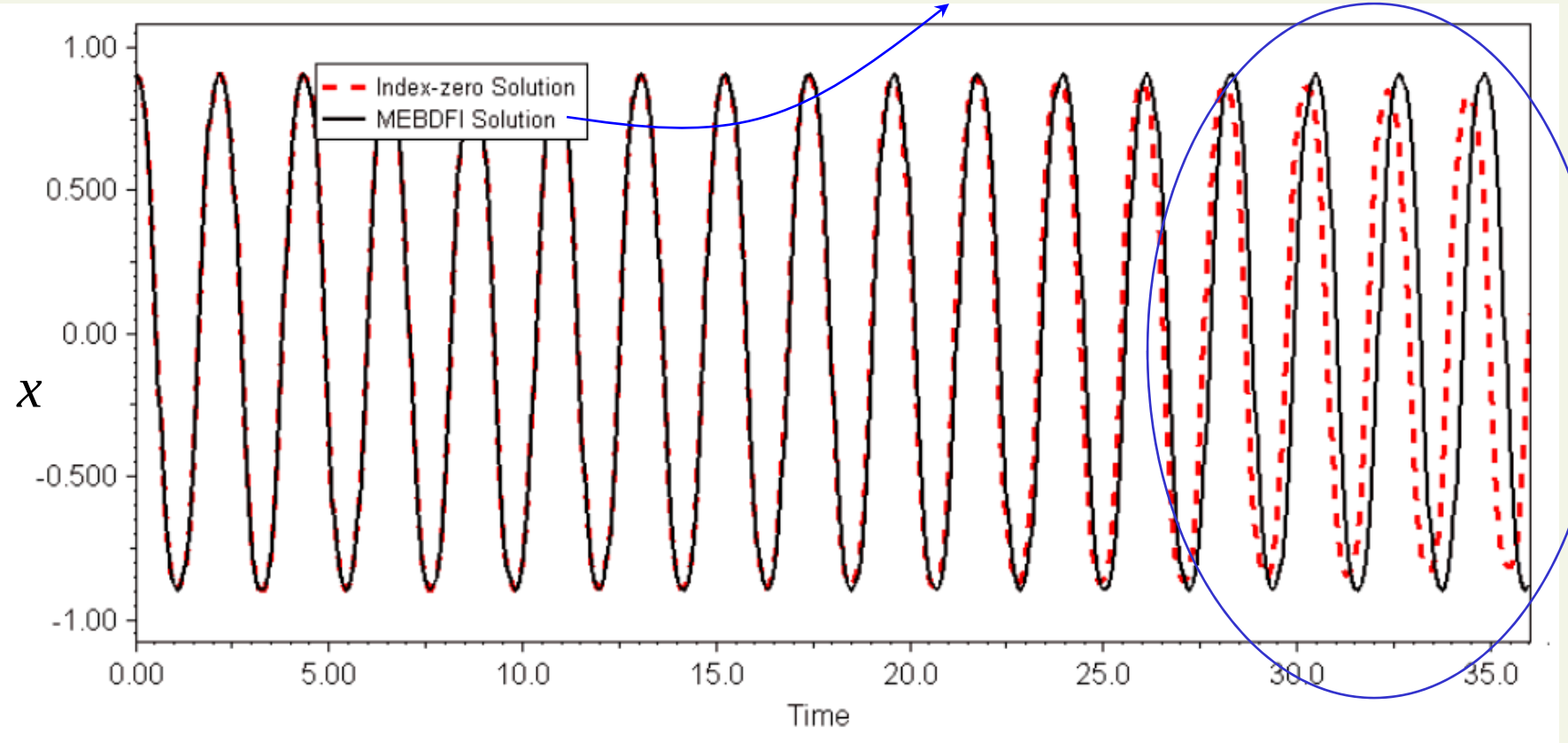


Dynamic Degree of Freedom
– High-Index DAE: simulation –

Error propagation

index-0 solver *vs* index-3 solver

Drift-off effect



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

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

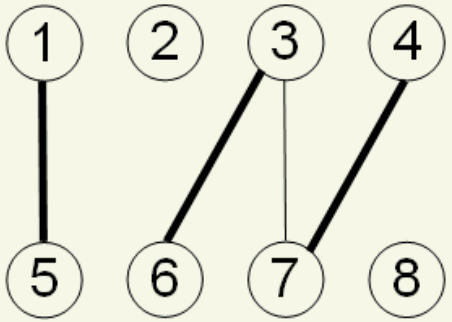
Debugging Techniques
– Current Status –

- 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}, {3,6}, {4,7}} w/ augmenting path

Debugging Techniques
– Bipartite Graphs: variable-equations –

Graph for variable-equation relationship

$f_1(x_1) = 0$

$f_2(x_1, x_2) = 0$

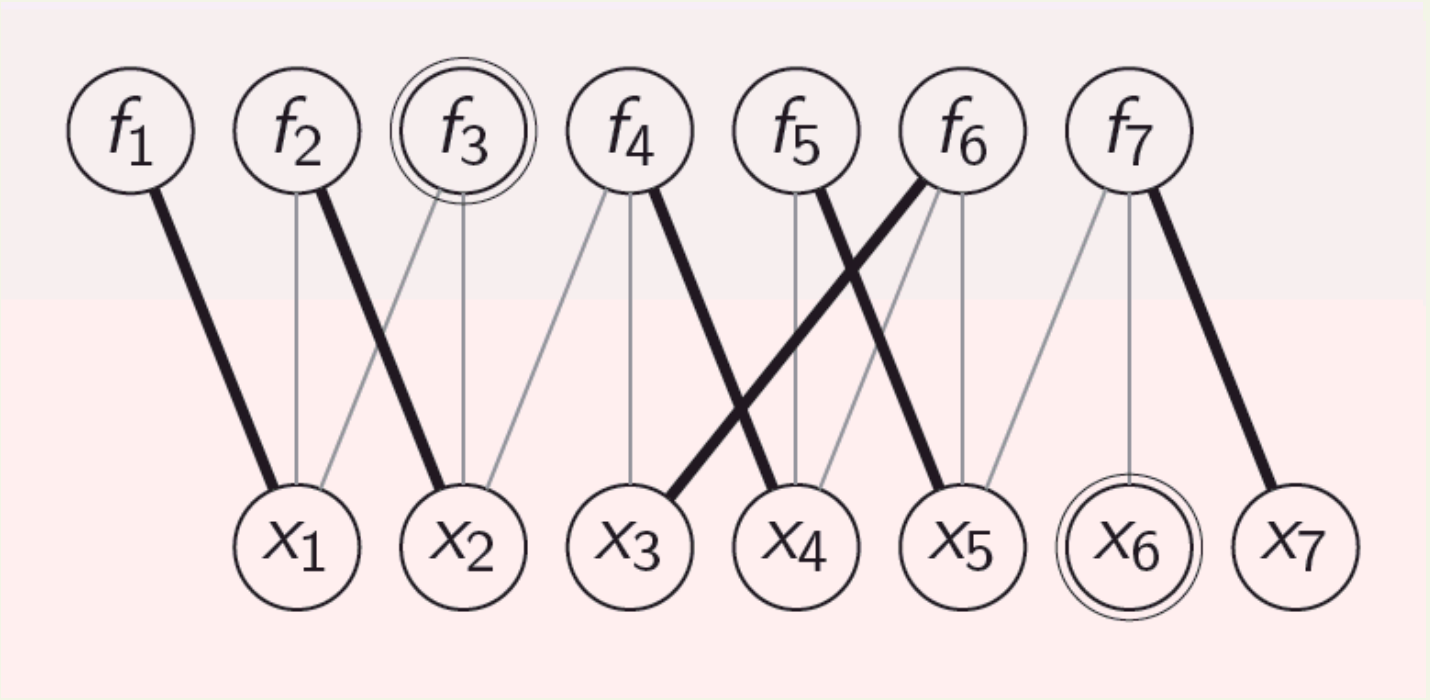
$f_3(x_1, x_2) = 0$

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

$f_5(x_4, x_5) = 0$

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

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



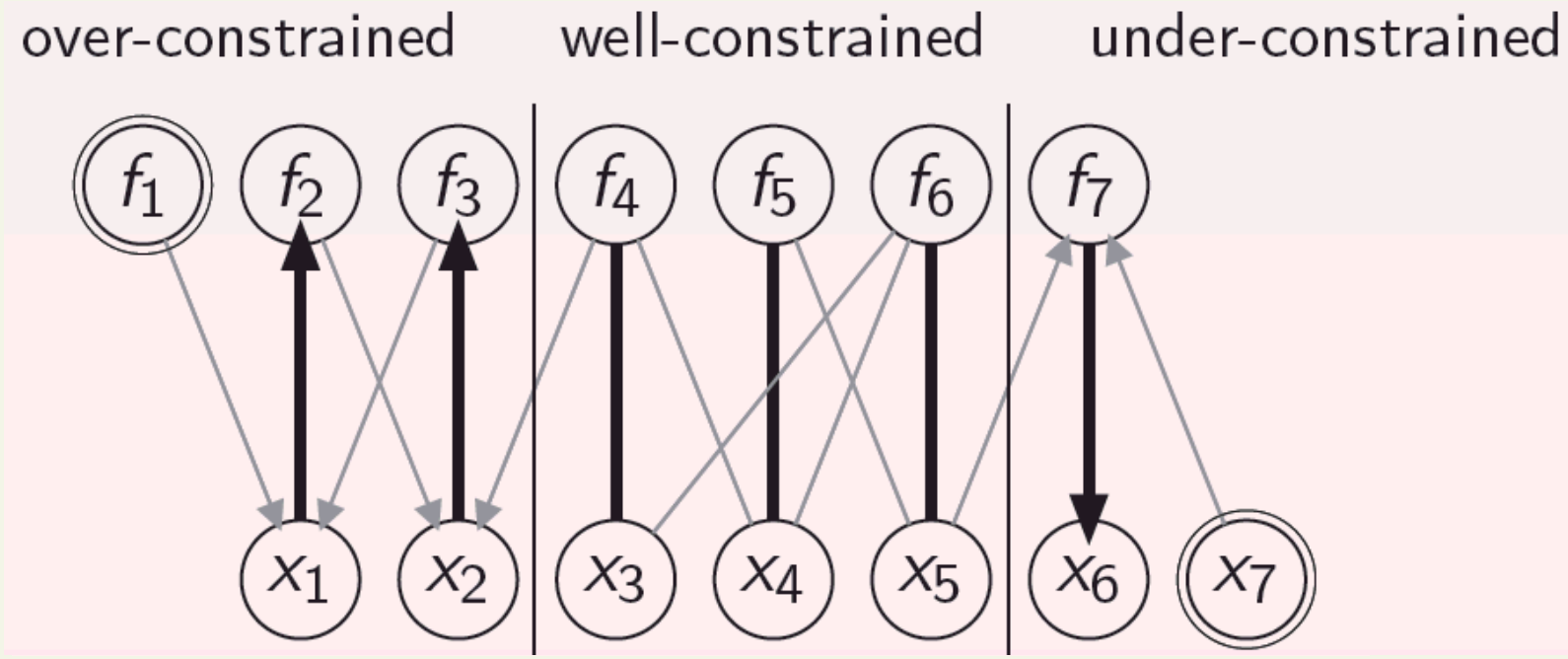
variables values
or equations forms
are irrelevant

Maximum Matching
Multiple Solutions

Debugging Techniques
– Nonlinear Algebraic Equations –

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

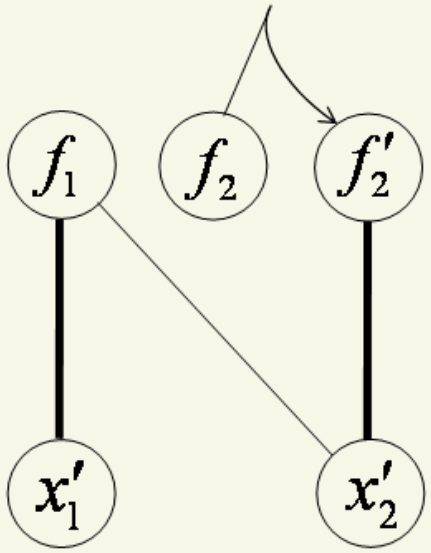
Solution:

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

Debugging Techniques
– Bipartite Graphs: DAE system –

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.
```

Debugging Techniques
– gPROMS output –

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

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

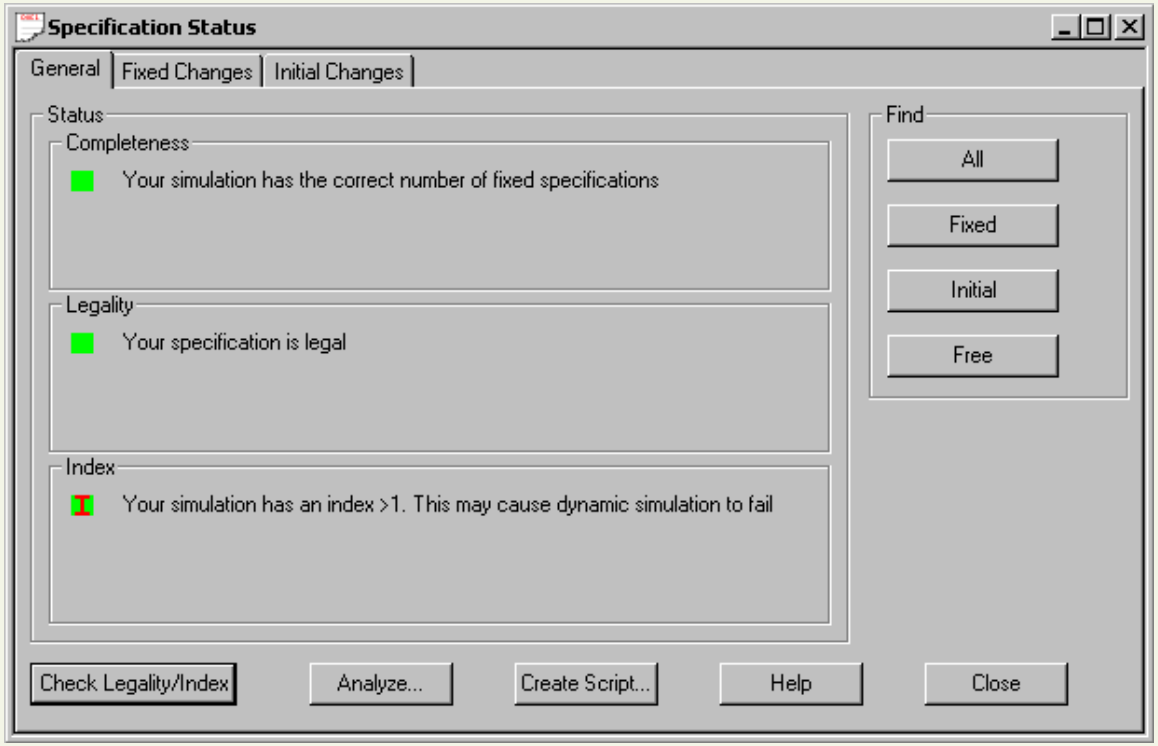
```
 Performing initialization calculation at time: 0
  Variables
       Known              : 0
       Unknown            : 2
          Differential    : 2
          Algebraic       : 0
    Model equations       : 2
    Initial conditions    : 2
Checking index of differential-algebraic equations (DAEs)...
    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
```



Debugging Techniques
– New Algorithm: debugging DAE system –

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

1: $M \leftarrow \emptyset$
2: **for** $v_e \in V_e$ **do**
3: **if not** $AugmentMatching2(G = (V_e, V_v, E), M, v_e,$ **false**$)$ **then**
4: *mark* **all** *colored* $v_k \in V_e$
5: *uncolour* V_e
6: **if not** $AugmentMatching2(G = (V_e, V_v, E), M, v_e,$ **true**$)$ **then**
7: **return false**
8: **end if**
9: **diff all** *marked* $v_k \in V_e$
10: **else**
11: *uncolour* V_e
12: **end if**
13: **end for**
14: **return true**

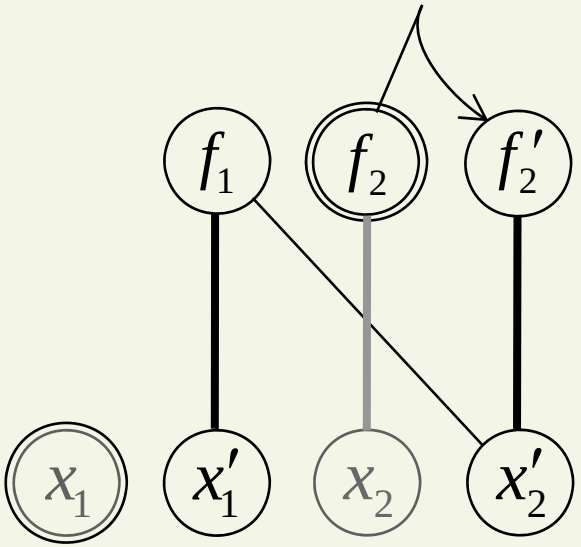
Debugging Techniques
– New Algorithm: debugging DAE system –

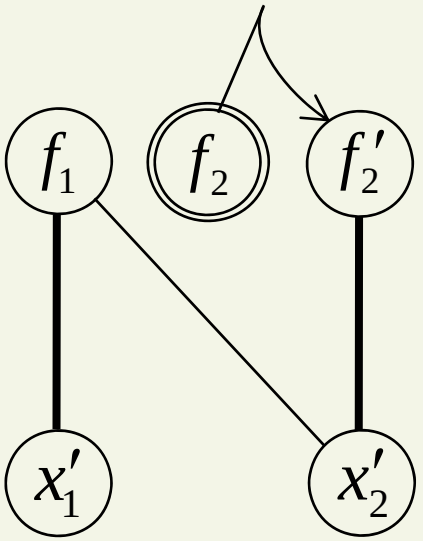
$AugmentMatching2(G = (V_e \cup V_v, E), M, v_e, alg)$

1: *colour* v_e
2: **if exists** $\{v_e, v_v\} \in E$ **and** $\{v_e, v_v\} \notin M$ **and** v_v *is elegible* **then**
3: $M \leftarrow M \cup \{v_e, v_v\}$
4: **return true**
5: **end if**
6: **for all** $\{v_e, v_v\} \in E$ **do**
7: **if exists** $\{v_{e2}, v_v\} \in M$ **and** v_{e2} **not** *colored* **and** v_v *is elegible* **then**
8: **if** $AugmentMatching2(G = (V_e \cup V_v, E), M, v_{e2}, alg)$ **then**
9: $M \leftarrow M \cup \{v_e, v_v\}$
10: **return true**
11: **end if**
12: **end if**
13: **end for**
14: **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)

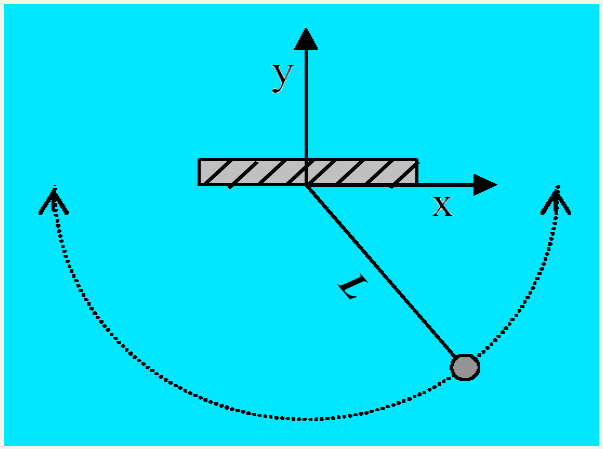
Debugging Techniques
– EMSO output –

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

- 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 –



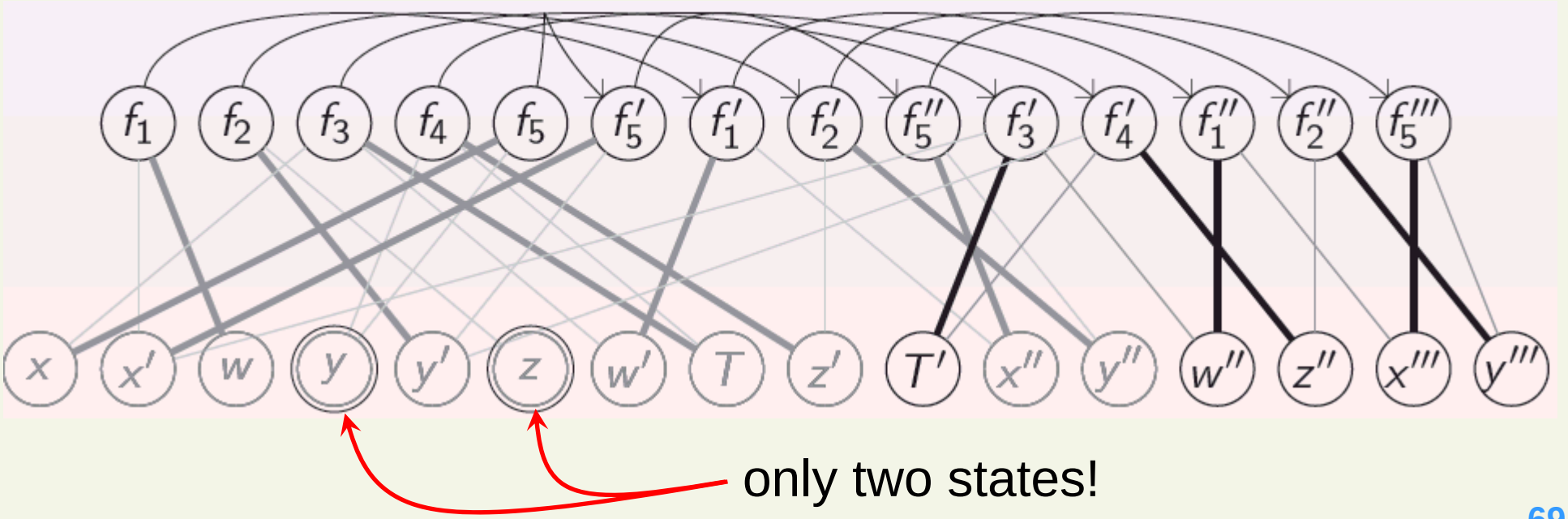
$$x' = w \quad (1)$$

$$y' = z \quad (2)$$

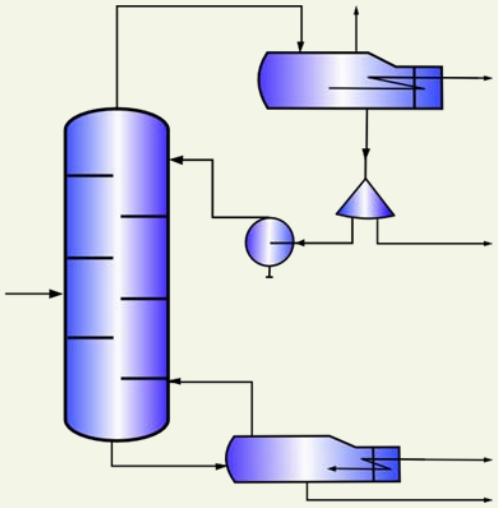
$$w' = T.x \quad (3)$$

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

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



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

What is coming next?

- Tools
  - Model updating tool and development of virtual analyzer based on Constrained Extended Kalman Filter (CEKF)
  - Model generation tool for predictive controllers

- Features
  - Creation of discretization functions for integral-partial differential equations
  - Implementation of MINLP solver interfaces

- Technologies
  - Hessian evaluation by reverse-mode automatic differentiation
  - New resources for incremental building of flowsheets in the G.U.I.

Challenges

Robust strategies for on-line updating of dynamic models

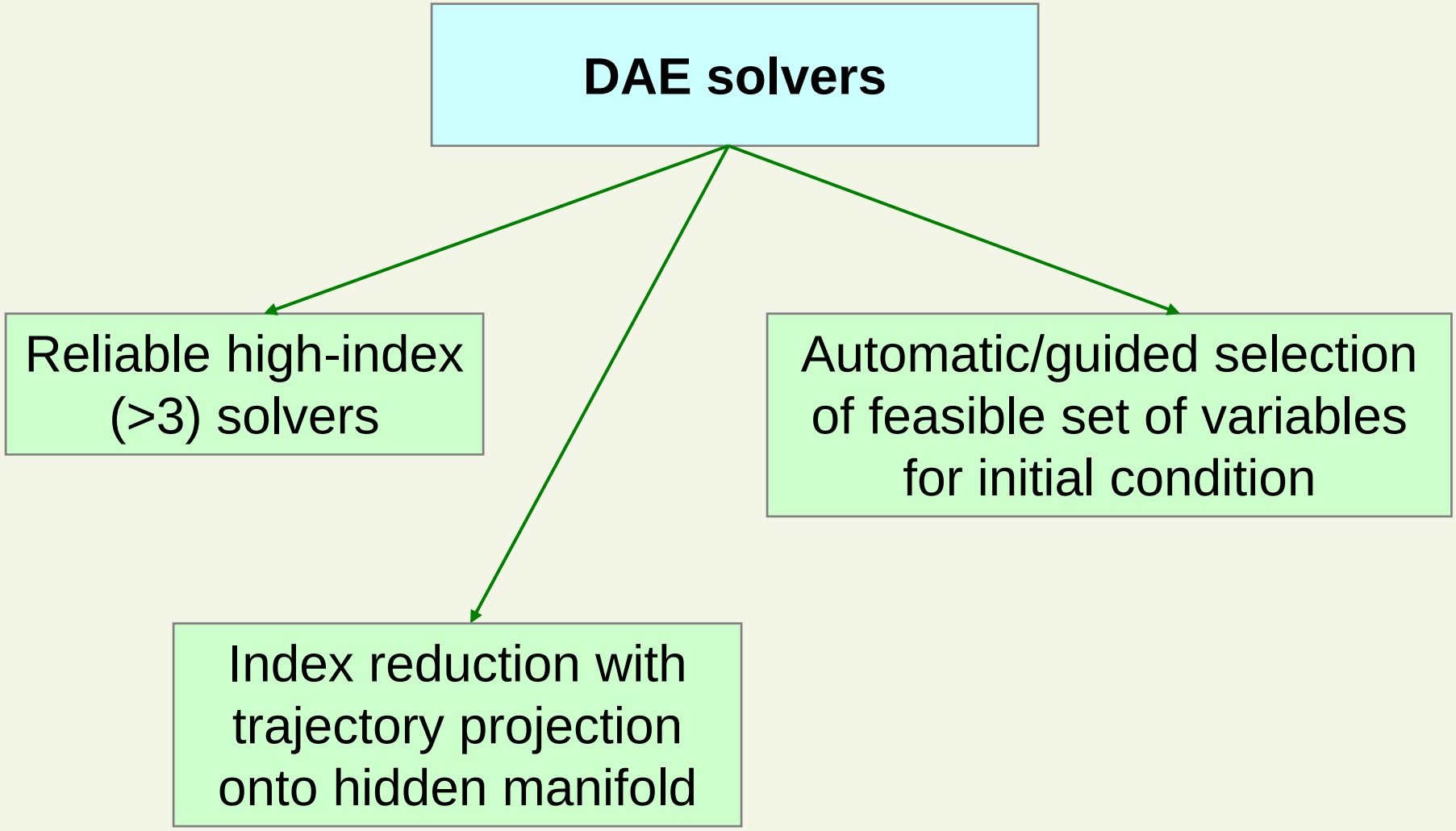
Dynamic data reconciliation and gross error detection

Parameters selection and estimation

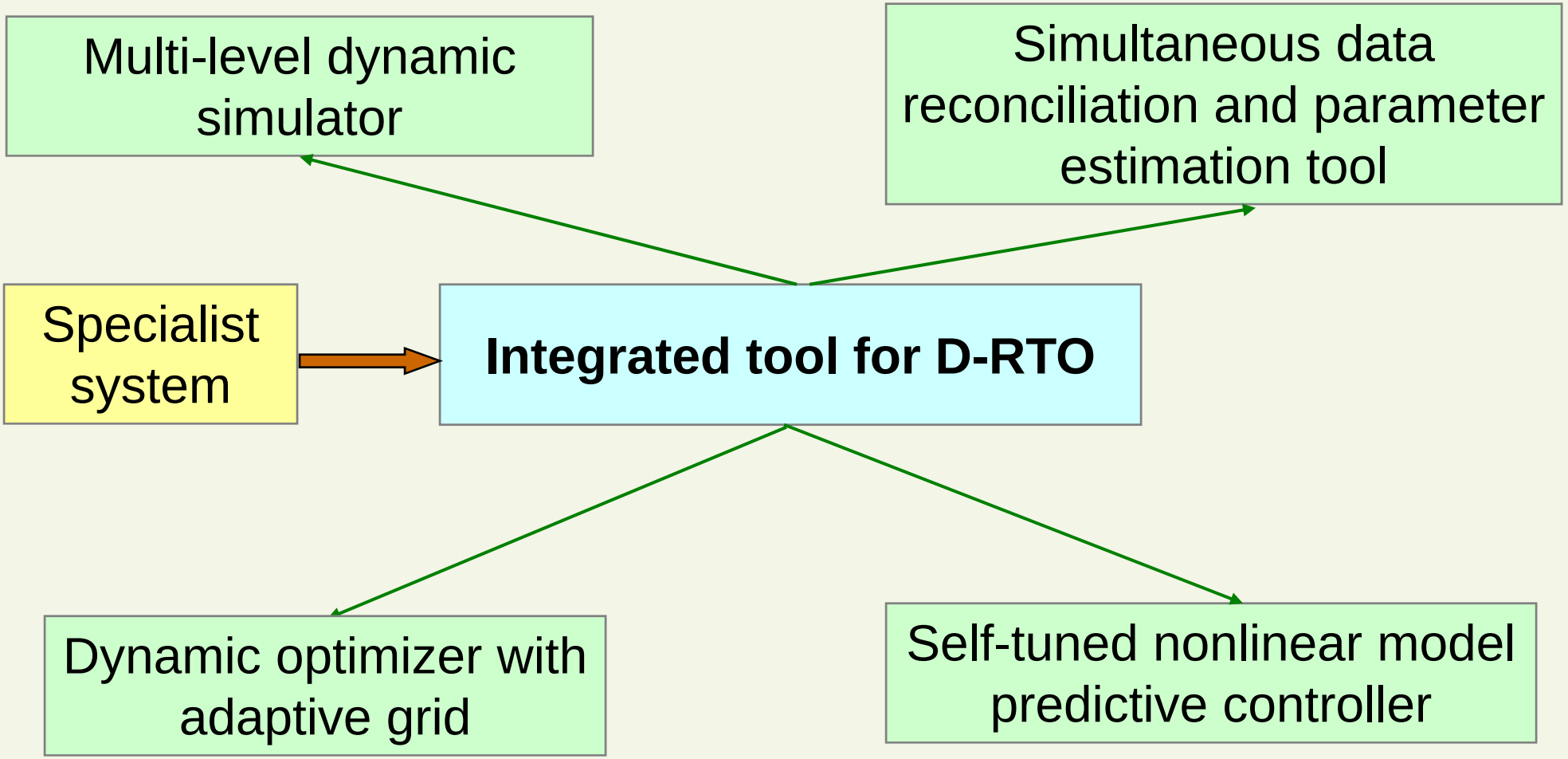
Related topics:

- Hybrid and rigorous modeling
- Order reduction of nonlinear models
- Fault diagnosis
- NMPC tuning strategies

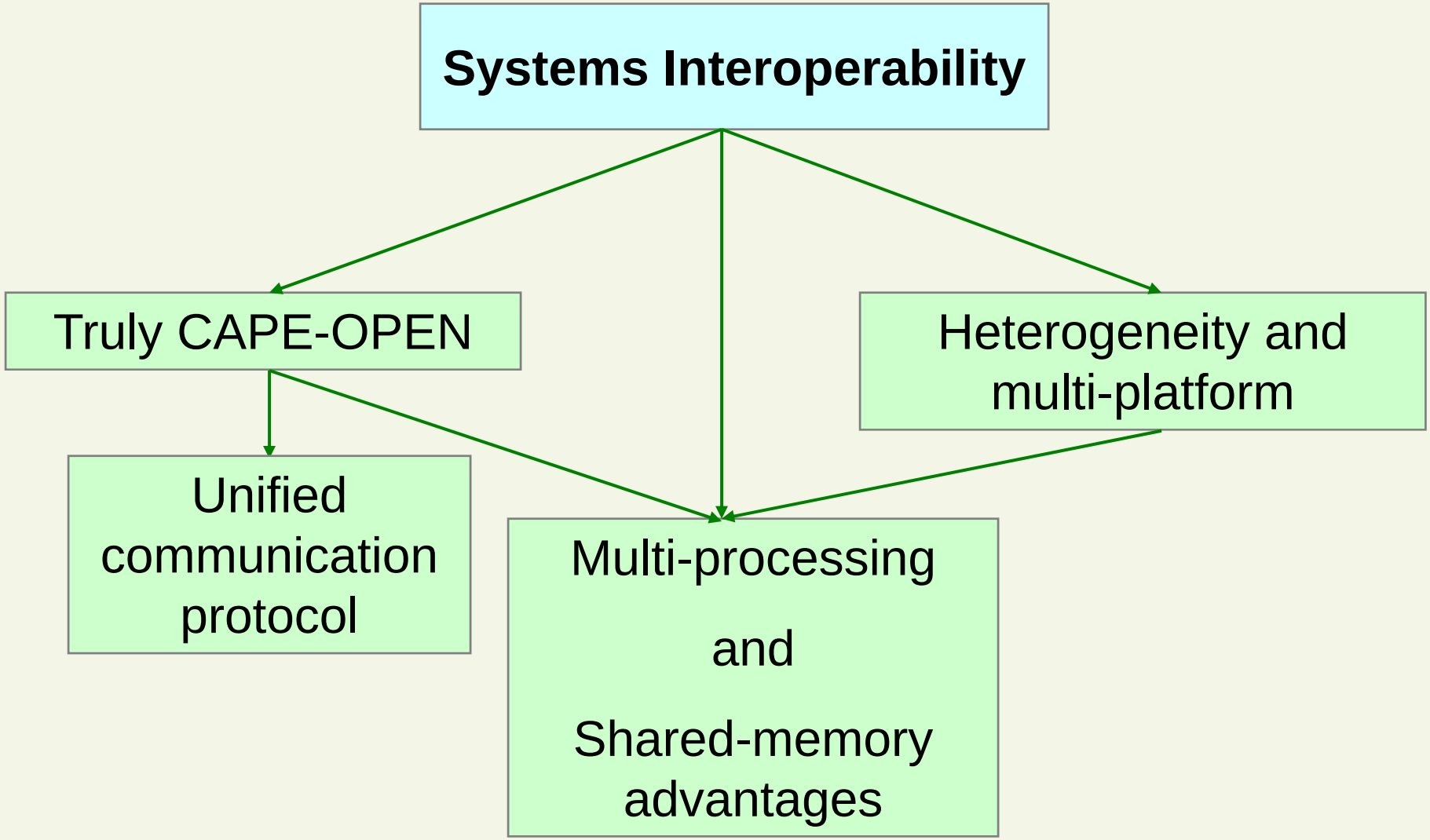
Challenges


DAE solvers
Reliable high-index (>3) solvers
Automatic/guided selection of feasible set of variables for initial condition
Index reduction with trajectory projection onto hidden manifold

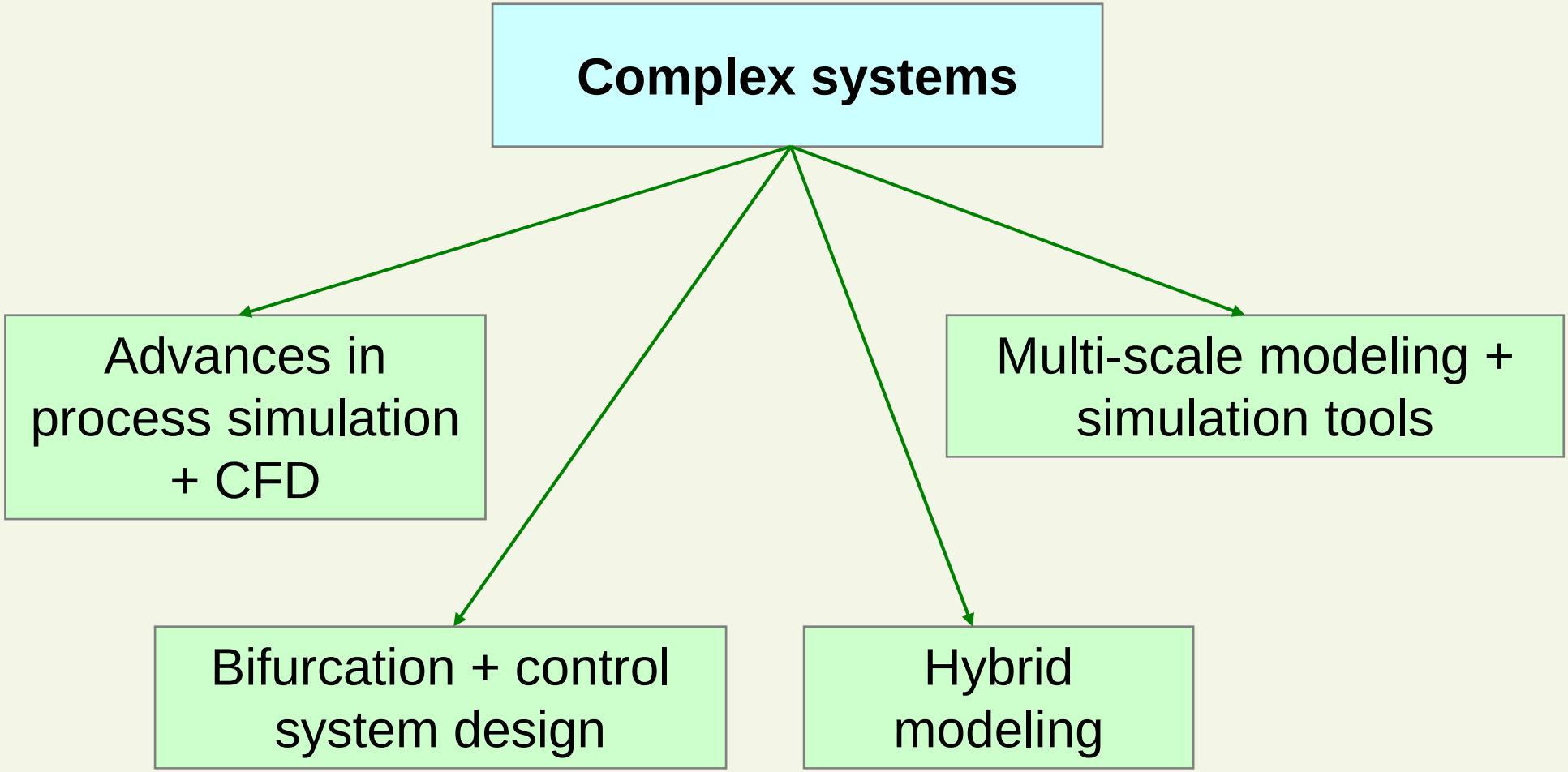
Challenges


Multi-level dynamic simulator
Simultaneous data reconciliation and parameter estimation tool
Specialist system
Integrated tool for D-RTO
Dynamic optimizer with adaptive grid
Self-tuned nonlinear model predictive controller

Challenges


Systems Interoperability
Truly CAPE-OPEN
Heterogeneity and multi-platform
Unified communication protocol
Multi-processing and Shared-memory advantages

Challenges


Complex systems
Advances in process simulation + CFD
Multi-scale modeling + simulation tools
Bifurcation + control system design
Hybrid modeling

References

• Biegler, L.T., A.M. Cervantes and A. Wächter. Advances in Simultaneous Strategies for Dynamic Process Optimization. *Chemical Engineering Science*, 57, 575–593 (2002).

• Charpentier, J.C. and T.F. McKenna. Managing Complex Systems: Some Trends for the Future of Chemical and Process Engineering. *Chemical Engineering Science*, 59, 1617–1640 (2004).

• Costa Jr., E.F., R.C. Vieira, A.R. Secchi and E.C. Biscaia Jr. Dynamic Simulation of High-Index Models of Batch Distillation Processes. *Journal of Latin American Applied Research*, 32 (2) 155–160 (2003).

• Marquardt, W. and M. Mönnigmann. Constructive Nonlinear Dynamics in Process Systems Engineering. *Computers and Chemical Engineering*, 29, 1265–1275 (2005).

• Martinson, W.S. and P.I. Barton. Distributed Models in Plantwide Dynamic Simulators. *AIChE Journal*, 47 (6) 1372–1386 (2001).

• Rodrigues, R., R.P. SOARES and A.R Secchi. Teaching Chemical Reaction Engineering Using EMSO Simulator. *Computer Applications in Engineering Education*, Wiley (2008).

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

• Soares, R.P. and A.R. Secchi. Modifications, Simplifications, and Efficiency Tests for the CAPE-OPEN Numerical Open Interfaces. *Computers and Chemical Engineering*, 28, 1611–1621 (2004).

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

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

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

References

DAE Solvers:

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

DASSLC: Secchi, A.R. and F.A. Pereira (1997), 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: R. Serban et al. (2004), http://www.llnl.gov/CASC/sundials/description/description.html

Research Group

GIMSCOP - 2008

Argimiro Resende Secchi, D.Sc.
Evaristo Chalbaud Biscaia Jr, D.Sc.
Jorge Otávio Trierweiler, D.Sc.
Nilo Sérgio Medeiros Cardozo, D.Sc.
Marcelo Farenzena, D.Sc.
Rafael de Pelegrini Soares, D.Sc.
Adriano Giraldi Fisch, M.Sc.
Débora Jung Luvizetto, M.Sc.
Edson Cordeiro do Valle, M.Sc
Eduardo Moreira de Lemos, M.Sc.
Euclides Almeida Neto, M.Sc.
Gabriela Sporleder Straatmann, M.Sc.
Gérson Balbueno Bicca, M.Sc.
Gustavo Alberto Neumann, M.Sc.
Luciane da Silveira Ferreira, M.Sc.
Marcelo Escobar, M.Sc.
Nina Paula Gonçalves Salau, M.Sc.
Paula Betio Staudt, M.Sc.
Ricardo Guilherme Duraiski, M.Sc.
Tiago Fiorenzano Finkler, M.Sc.
Anderson de Campos Paim, Eng.
Andrea Cabral Farias, Eng.
Antonio José V. Nascimento, Eng.
Bruna Racoski, Eng.
Cristine Alessandra Kayser, Eng.
Fabio Cesar Diehl, Eng.
Gustavo Rodrigues Sandri, Eng.
Jovani Luiz Fávero, Eng.
Luciano Forgiarini, Eng.
Marcos Lovato Alencastro, Eng.
Rafael Busato Sartor, Eng.
Rodolfo Rodrigues, Eng.
Thais Machado Farias, Eng.
Bruno Cardozo Mohler, I.C.
Caio Felippe Curitiba Marcellos, I.C.
Ivana Martins, I.C.
Josias José Junges, I.C.
Luiza Gueller Zardin, I.C.
Maria Aparecida Paula Lima, I.C.
Sara Scomazzon Masiero, I.C.
Igor Rodacovski, Tec. Inf.
Irma Maria Bueno, Sec.

... thank you for your attention!

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

Process Modeling, Simulation and Control Lab

- Prof. Dr. Argimiro Resende Secchi
- Phone: +55-21-2562-8349
- E-mail: arge@peq.coppe.ufrj.br
- http://www.peq.coppe.ufrj.br/Areas/Modelagem_e_simulacao.html

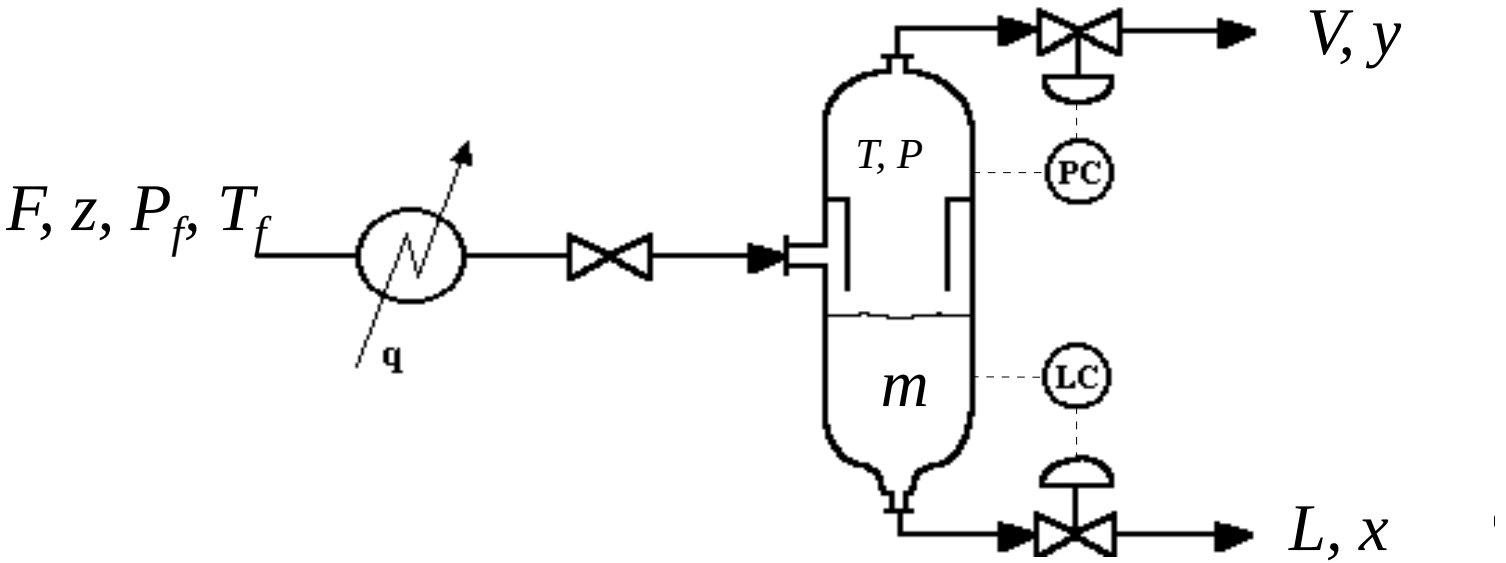
Process Simulation and Optimization Lab

- Prof. Dr. Rafael de Pelegrini Soares
- Phone: +55-51-3308-4166
- E-mail: rafael@enq.ufrgs.br
- http://www.enq.ufrgs.br/labs/lasim.html

Extra slides

Building Dynamic Models
– Another simple example –

Flash multi-component



Building Dynamic Models
– FLASH: process description –

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.

Building Dynamic Models
– 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) $\approx \Delta$(liquid-phase enthalpy);
- perfect mixture in both phases.

Building Dynamic Models
– FLASH: modeling –

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:

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

Exercícios de Modelagem

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:

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

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

Building Dynamic Models
– FLASH: consistency analysis –

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 $kmol^{-1}$

Building Dynamic Models
– FLASH: consistency analysis –

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

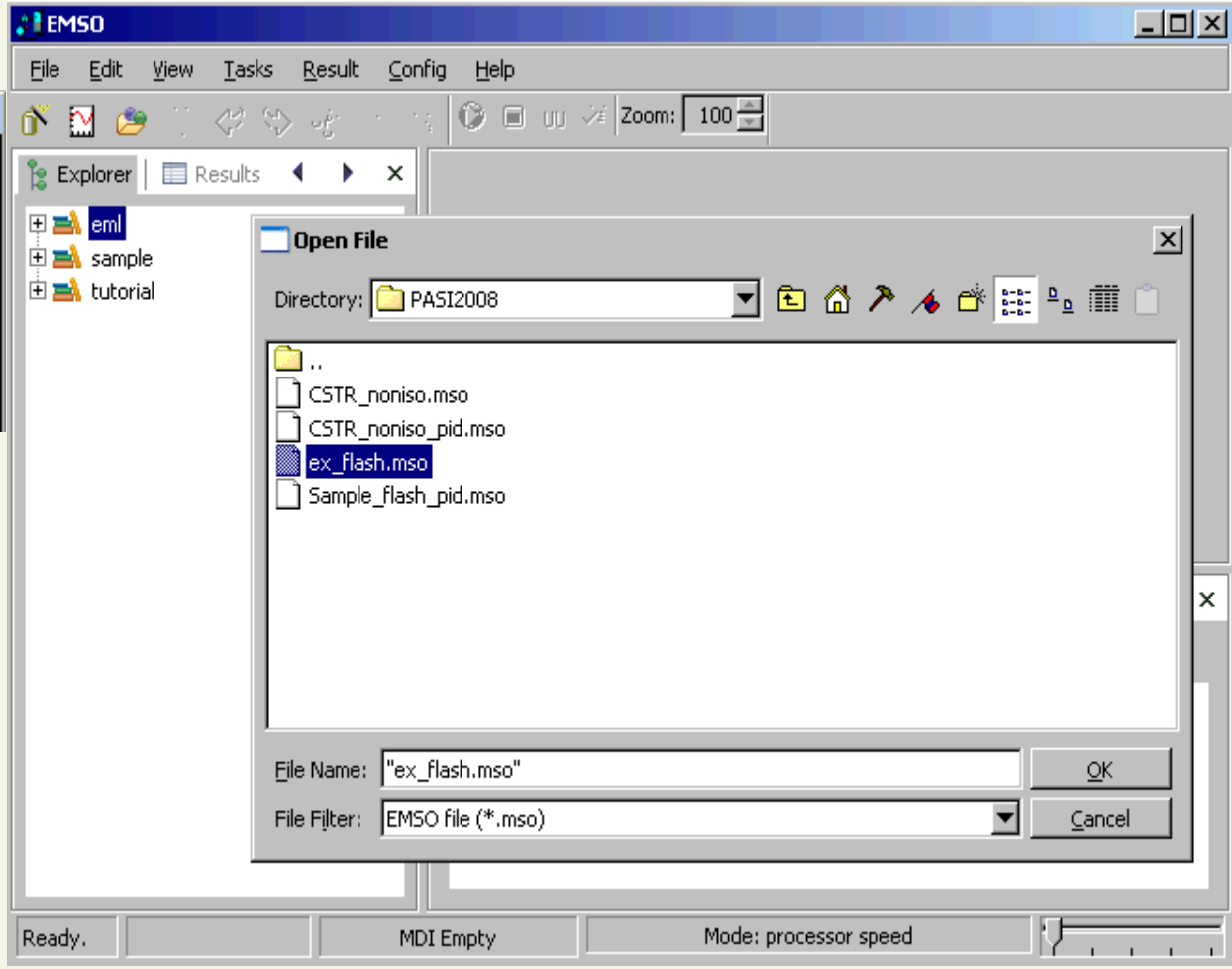
Degree of Freedom = variables – constants – specifications – driving forces – equations = unknown variables – equations = (13+4C) – 0 – 2 – (3+C) – (8+3C) = 0

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

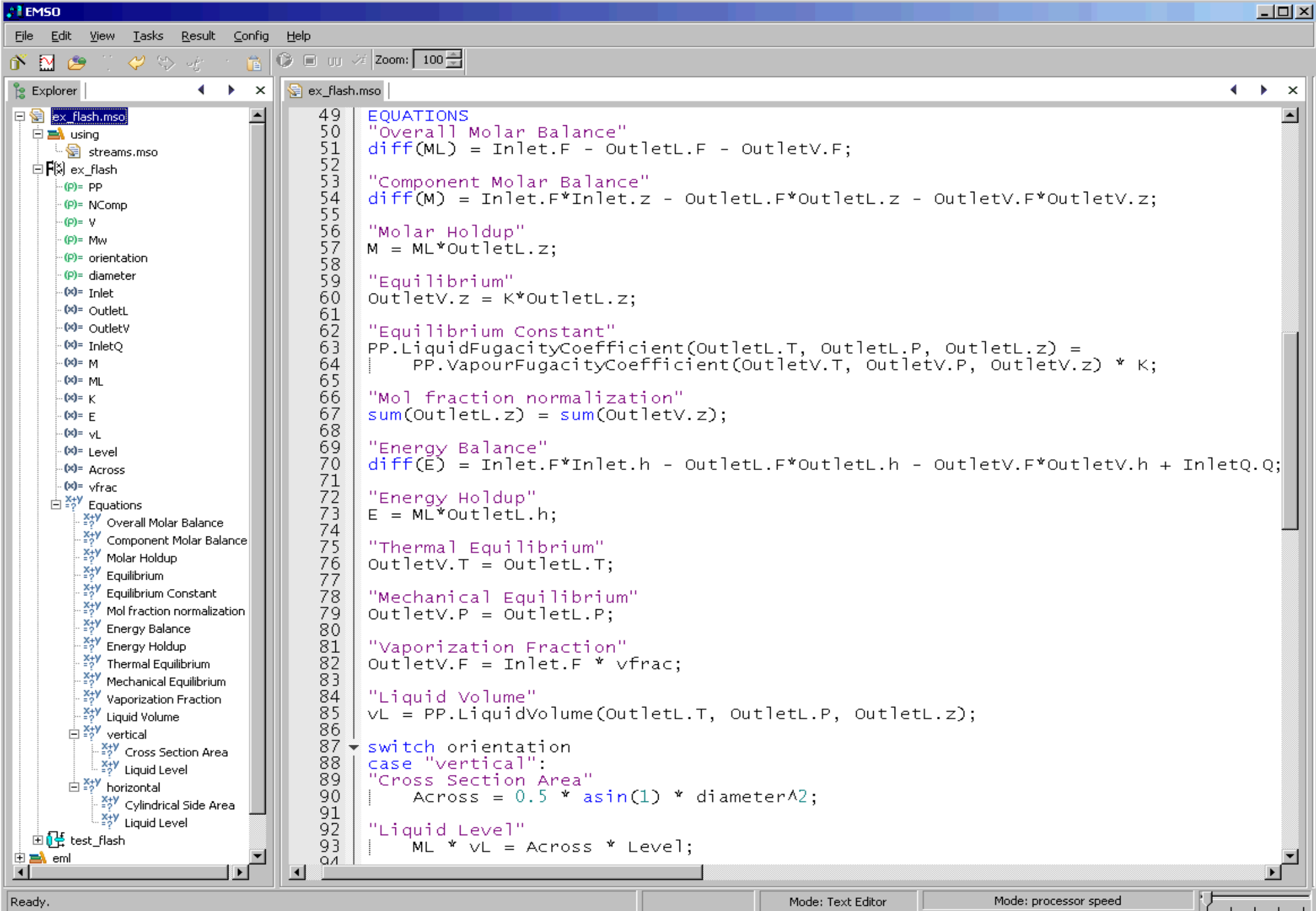
Dynamic Degree of Freedom (index < 2) = differential equations – initial conditions = (2+C) – (2+C) = 0

Building Dynamic Models
– FLASH: EMSO version –

- Running EMSO



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


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
49 EQUATIONS
50 "Overall Molar Balance"
51 diff(ML) = Inlet.F - OutletL.F - OutletV.F;
52
53 "Component Molar Balance"
54 diff(M) = Inlet.F*Inlet.z - OutletL.F*OutletL.z - OutletV.F*OutletV.z;
55
56 "Molar Holdup"
57 M = ML*OutletL.z;
58
59 "Equilibrium"
60 OutletV.z = K*OutletL.z;
61
62 "Equilibrium Constant"
63 PP.LiquidFugacityCoefficient(OutletL.T, OutletL.P, OutletL.z) =
64 PP.VapourFugacityCoefficient(OutletV.T, OutletV.P, OutletV.z) * K;
65
66 "Mol fraction normalization"
67 sum(OutletL.z) = sum(OutletV.z);
68
69 "Energy Balance"
70 diff(E) = Inlet.F*Inlet.h - OutletL.F*OutletL.h - OutletV.F*OutletV.h + InletQ.Q;
71
72 "Energy Holdup"
73 E = ML*OutletL.h;
74
75 "Thermal Equilibrium"
76 OutletV.T = OutletL.T;
77
78 "Mechanical Equilibrium"
79 OutletV.P = OutletL.P;
80
81 "Vaporization Fraction"
82 OutletV.F = Inlet.F * vfrac;
83
84 "Liquid Volume"
85 vL = PP.LiquidVolume(OutletL.T, OutletL.P, OutletL.z);
86
87 switch orientation
88 case "vertical":
89 "Cross Section Area"
90 Across = 0.5 * asin(1) * diameter^2;
91
92 "Liquid Level"
93 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_

- test_flash
  - NComp
  - Q
  - fl
    - NComp
    - V
    - Mw
    - diameter
    - Inlet
    - OutletL
      - NComp
      - F
      - T
      - P
      - h
      - v
      - z
    - OutletV
    - InletQ
    - M
    - ML
    - K
    - E
    - vL
    - Level
    - Across
    - vfrac
  - s1

ex_flash.mso

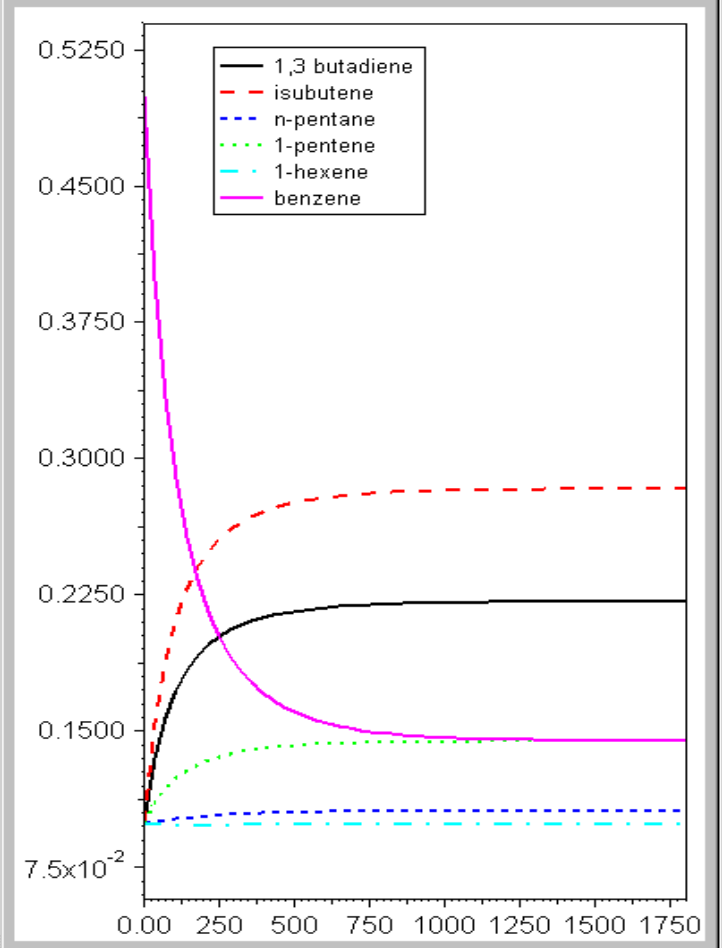
```
106 FlowSheet test_flash
107     PARAMETERS
108     PP as Plugin(Brief="Physical Properties", Type="PP",
109         Components = ["1,3-butadiene", "isobutene", "n-pentane",
110                       "1-pentene", "1-hexene", "benzene"],
111         LiquidModel = "PR",
112         VapourModel = "PR"
113     );
114     NComp as Integer;
115
116     VARIABLES
117     Q    as energy_source (Brief="Heat supplied");
118
119     SET
120     NComp = PP.NumberOfComponents;
121
122     DEVICES
123     fl as ex_flash;
124     s1 as source;
125
126     CONNECTIONS
127     s1.Outlet to fl.Inlet;
128     Q.OutletQ to fl.InletQ;
129
130     EQUATIONS
131     fl.OutletL.F = 400*sqrt(fl.Level/'m') * 'kmol/h';
132
133     SPECIFY
134     s1.Outlet.F = 496.3 * 'kmol/h';
135     s1.Outlet.T = 338 * 'K';
136     s1.Outlet.P = 507.1 * 'kPa';
137     s1.Outlet.z = [0.2379,0.30820,0.09958,
138                    0.1373,0.08872,0.12830];
139     fl.OutletV.F = 68.5 * 'kmol/h';
140     Q.OutletQ.Q = 0 * 'kJ/h';
141
142     SET
143     fl.V = 50 * 'm^3';
144     fl.diameter = 2 * 'm';
145     fl.orientation = "vertical";
146 #   fl.orientation = "horizontal";
147
148     INITIAL
149     fl.OutletL.T = 338 * 'K';
150     fl.Level = 0.4 * 'm';
151
152     fl.OutletL.z(1) = 0.1;
153     fl.OutletL.z(2) = 0.1;
154     fl.OutletL.z(3) = 0.1;
155     fl.OutletL.z(4) = 0.1;
156     fl.OutletL.z(5) = 0.1;
157     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)



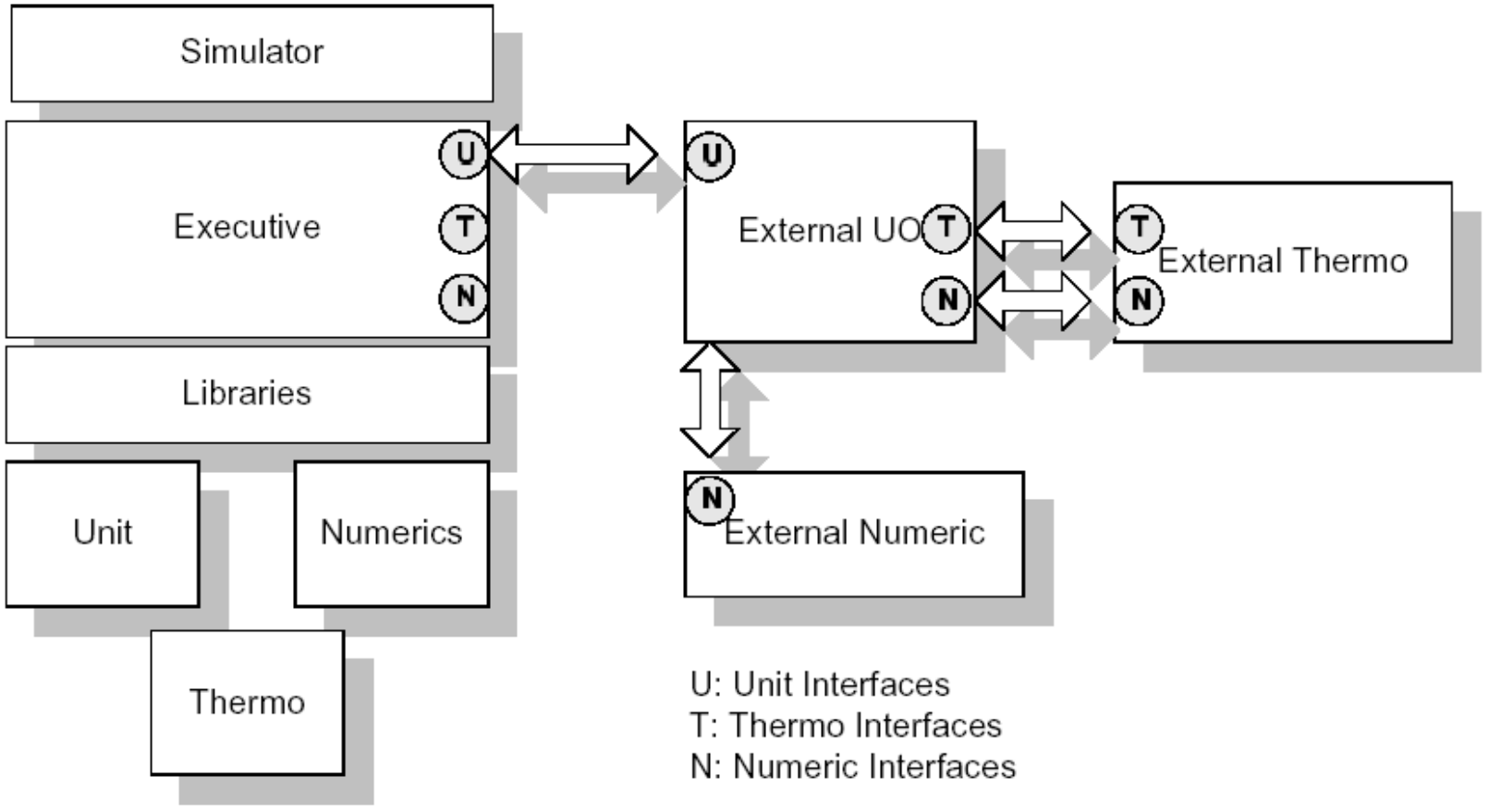
Ready.

Mode: Text Editor

Mode: processor speed

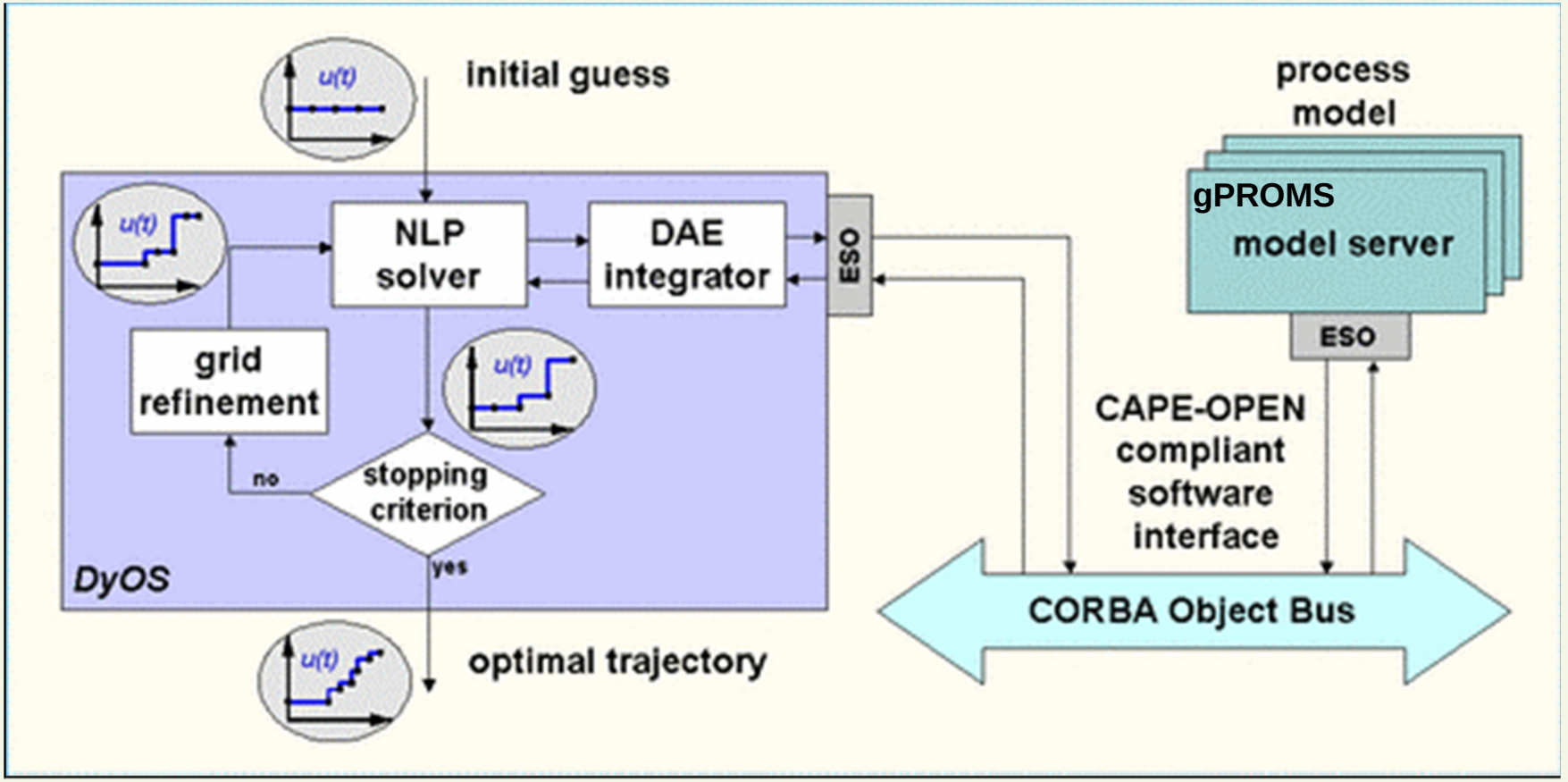
Standard Interfaces

CAPE-OPEN



CAPE OPEN

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

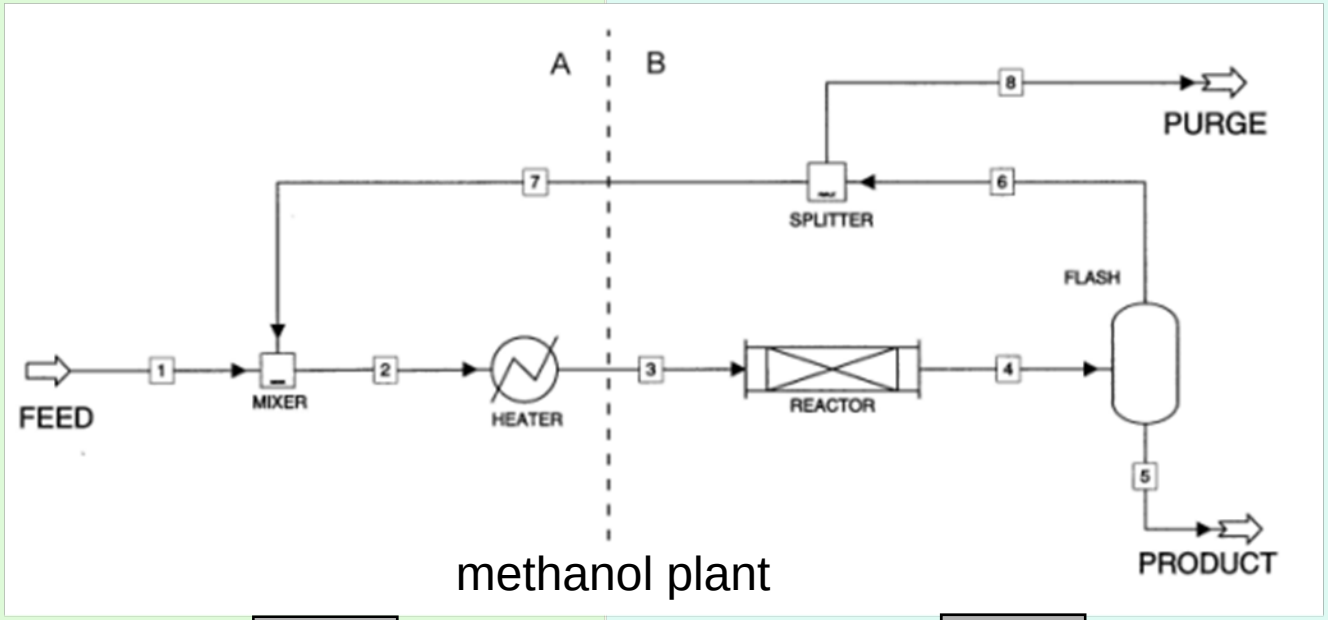


CAPE OPEN

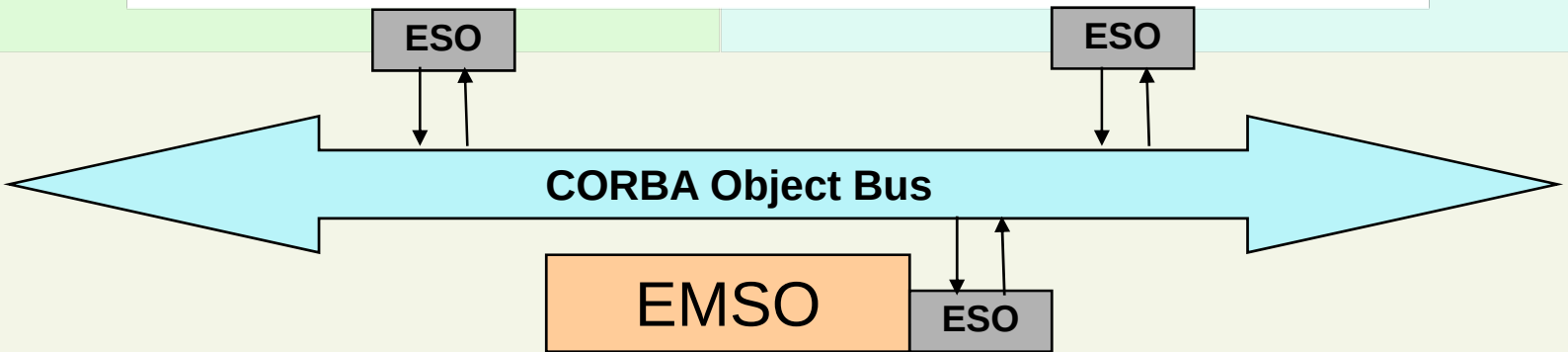
Another example of CAPE-OPEN: EMSO (Environment for Modeling, Simulation and Optimization) - Soares and Secchi (2004)

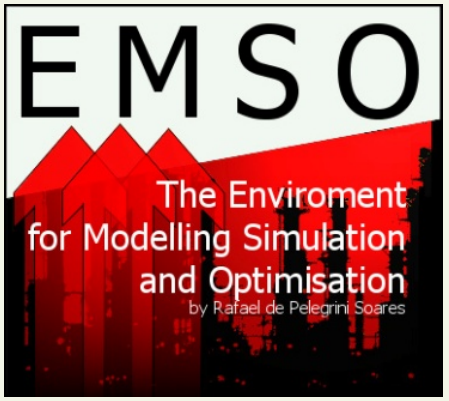
EMSO A

EMSO B



methanol plant





Other available tools and features

Optimization

flash_opt.mso

```
75 Optimization FlashOpt1 as FlashSteadyTest
76     MAXIMIZE
77     leves;
78
79     FREE
80     fl.OutletL.T;
81     #fl.OutletL.P;
82
83     EQUATIONS
84     fl.OutletL.T < 320 * 'K';
85     fl.OutletL.T > 300 * 'K';
86
87     OPTIONS
88     Dynamic = false;
89     NLPSolveNLA = false;
90     NLPSolver(#File = "complex",
91               #File = "optpp_emso",
92               File = "ipopt_emso",
93               RelativeAccuracy = 1e-6);
94 end
```

sample_optimization.mso

```
1  Optimization hs71
2      VARIABLES
3      x1 as Real(Default=2, Lower=1, Upper=5);
4      x2 as Real(Default=5, Lower=1, Upper=5);
5      x3 as Real(Default=5, Lower=1, Upper=5);
6      x4 as Real(Default=1, Lower=1, Upper=5);
7
8      MINIMIZE
9      x1*x4*(x1+x2+x3) + x3;
10
11     EQUATIONS
12     x1*x2*x3*x4 > 25;
13
14     x1*x1 + x2*x2 + x3*x3 + x4*x4 = 40;
15
16     OPTIONS
17     NLPSolver(#File = "ipopt_emso"
18               #File = "complex"
19               File = "optpp_emso"
20               );
21     Dynamic = false;
22 end
```

Parameter Estimation

Bio.mso

```
100 Estimation Biop_NE_Estt5 as Biop_NE_process5t
101     ESTIMATE
102     # PARAMETER   START     LOWER      UPPER      UNIT
103     Kss           0.2009      0.004    7          'kg/m^3';
104     Ksn           0.0446      0.005    1          'kg/m^3';
105     mim           0.7979      0.1      0.8        '1/s';
106     alfa          2.0293      1        5                ;
107     gama          0.08502     0.05     5          '1/s';
108     K1            0.4059      0.1      3          'kg/kg';
109     K2            -0.00795    -1       3          '1/s';
110     Yn            10.62       0.1      18               ;
111     kd            0.007       0.0005   1          '1/s';
112
113     EXPERIMENTS
114     # DATA FILE     WEIGTH
115     "Bio.dat"       1;
116
117     OPTIONS
118     Statistics(
119         Fits=true,
120         Parameters=false,
121         Predictions=false
122     );
123
124     NLPSolver(
125         MaxIterations = 1000,
126         File = "complex"
127         #File = "ipopt_emso"
128     );
129     Dynamic = true;
130 end
```

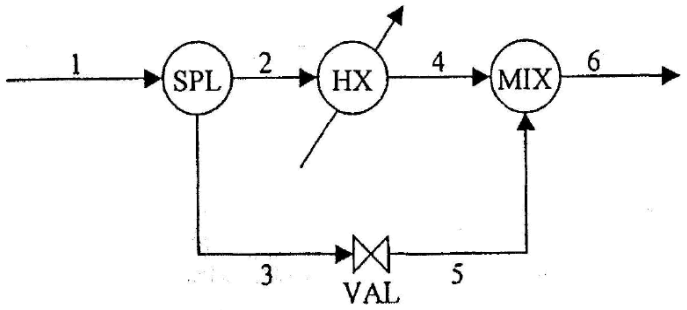
pv_est.mso

```
23 Estimation PV_Est as PV_Flow
24
25     ESTIMATE
26     # PAR    START   LOWER   UPPER   UNIT
27     A        1.5     -3      10;
28     B        1000    800     3000    'K';
29     C        50      20      200     'K';
30
31     EXPERIMENTS
32     # FILE                    WEIGTH
33     "pv_est.dat"        1;
34
35     OPTIONS
36     NumJac = false;
37
38     NLPSolver(
39         File = "ipopt_emso"
40         #File = "complex"
41     );
42     Dynamic = false;
43 end
```

Data Reconciliation

heatEx.mso

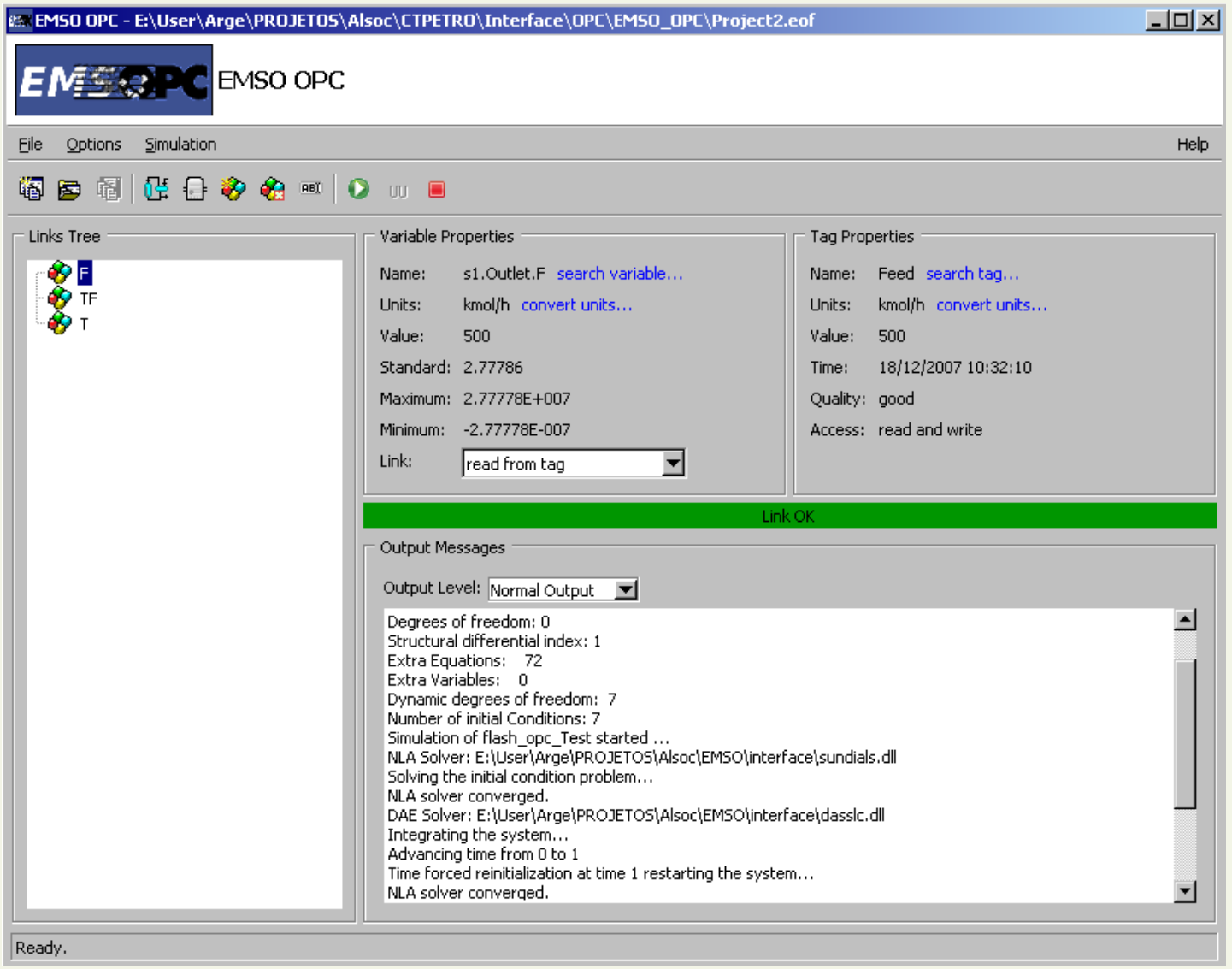
```
3  FlowSheet HeatEx_Flow
4      VARIABLES
5      x1 as Real (Default=50.00, Lower=0.00, Upper=150);
6      x2 as Real (Default=50.00, Lower=0.00, Upper=150);
7      x3 as Real (Default=50.00, Lower=0.00, Upper=150);
8      x4 as Real (Default=50.00, Lower=0.00, Upper=150);
9      x5 as Real (Default=50.00, Lower=0.00, Upper=150);
10     x6 as Real (Default=50.00, Lower=0.00, Upper=150);
11
12     SPECIFY
13     x1 = 100.91;
14     x2 = 64.45;
15     #x3 = 34.65;
16
17     EQUATIONS
18     x1 - x2 - x3 = 0;
19     x2 - x4 = 0;
20     x3 - x5 = 0;
21     x4 + x5 - x6 = 0;
22
23     OPTIONS
24     Dynamic = false;
25 end
```



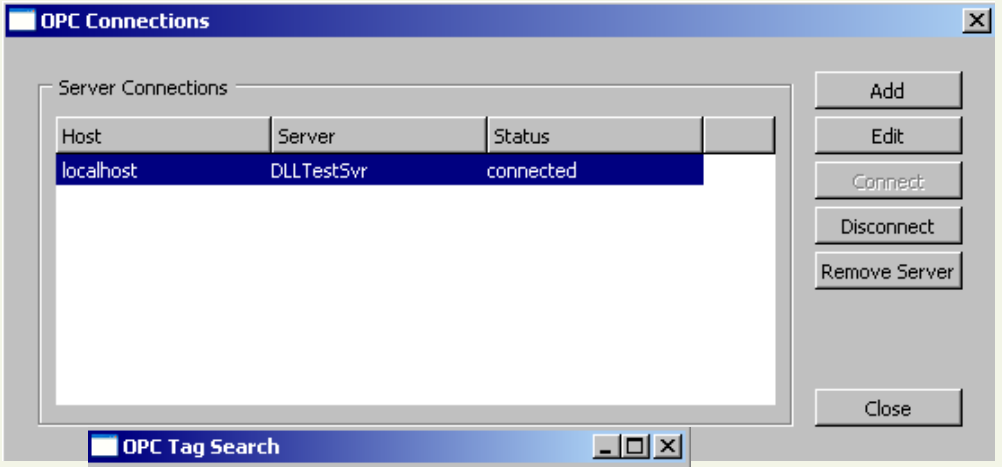
heatEx.mso

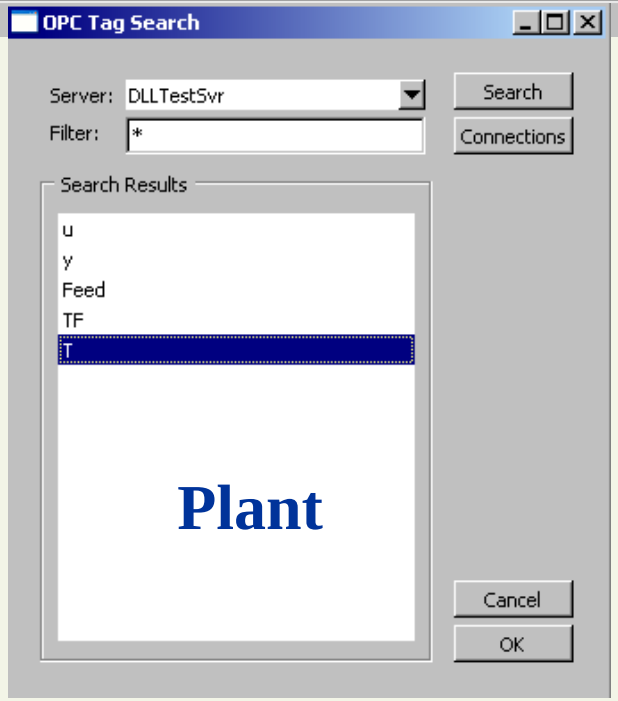
```
27 Reconciliation HeatEx_Rec as HeatEx_Flow
28
29     RECONCILE
30     x1; x2; x3; x4; x5; x6;
31     #x1; x2; x5; x6;
32     #x1; x2;
33     #x1; x6;
34
35     FREE
36     x1; x2;
37
38     EXPERIMENTS
39     # FILE                WEIGTH
40     "heatEx.dat"          1;
41     #"heatEx_1.dat"         1;
42     #"heatEx_2.dat"         1;
43     #"heatEx_3.dat"         1;
44     #"heatEx_GE.dat"        1;
45
46     OPTIONS
47     Filter = "mean";
48     Significance = 0.95;
49
50     GrossErrorTests(
51         Global = true,
52         Nodal = true,
53         Measurements = true
54     );
55
56     NLPSolver(
57         MaxIterations=1000,
58         File = "complex"
59         #File = "ipopt_emso"
60     );
61     Dynamic = false;
62 end
```

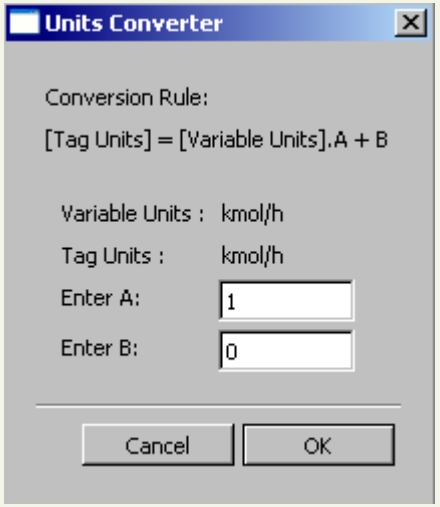
Interface EMSO-OPC



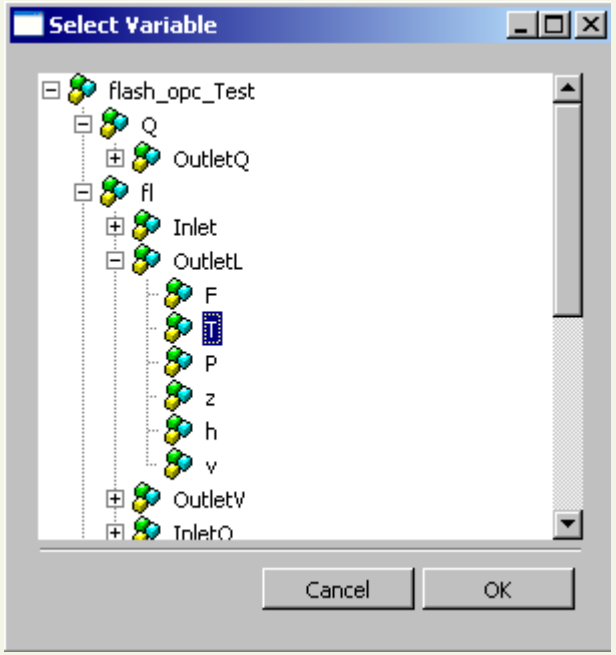
Interface EMSO-OPC



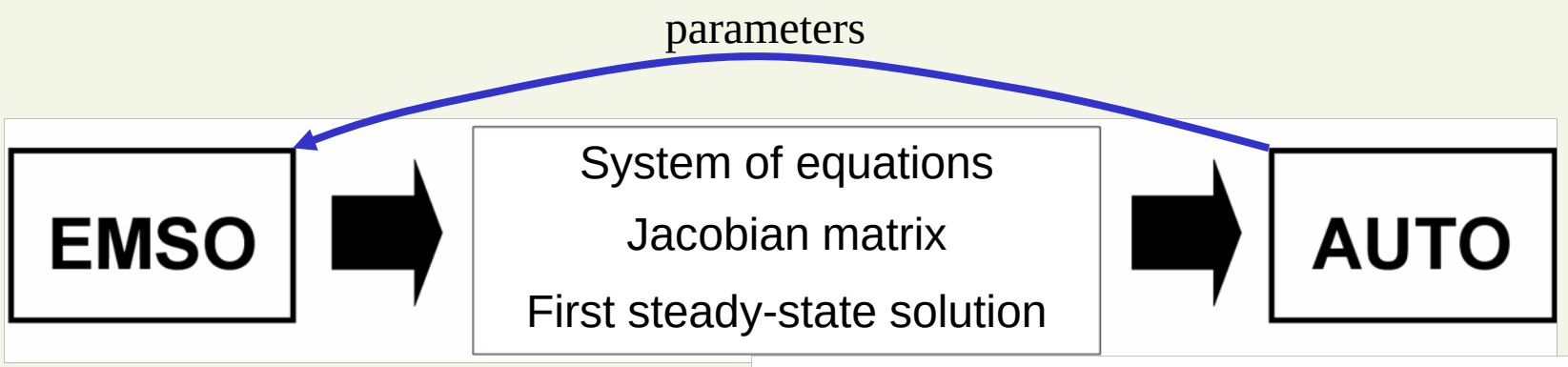




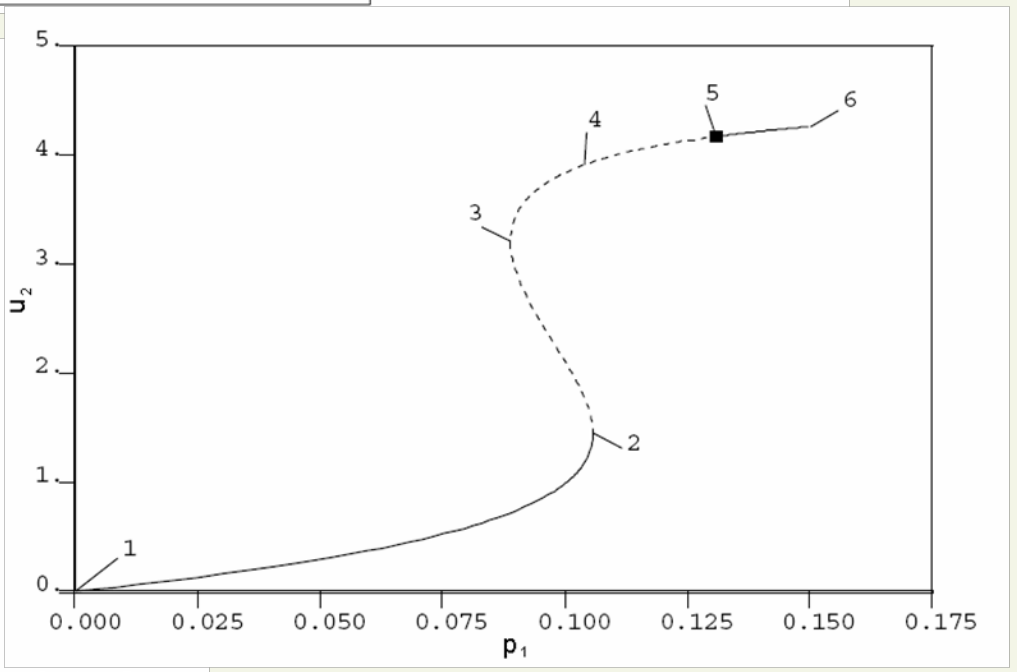
Simulator



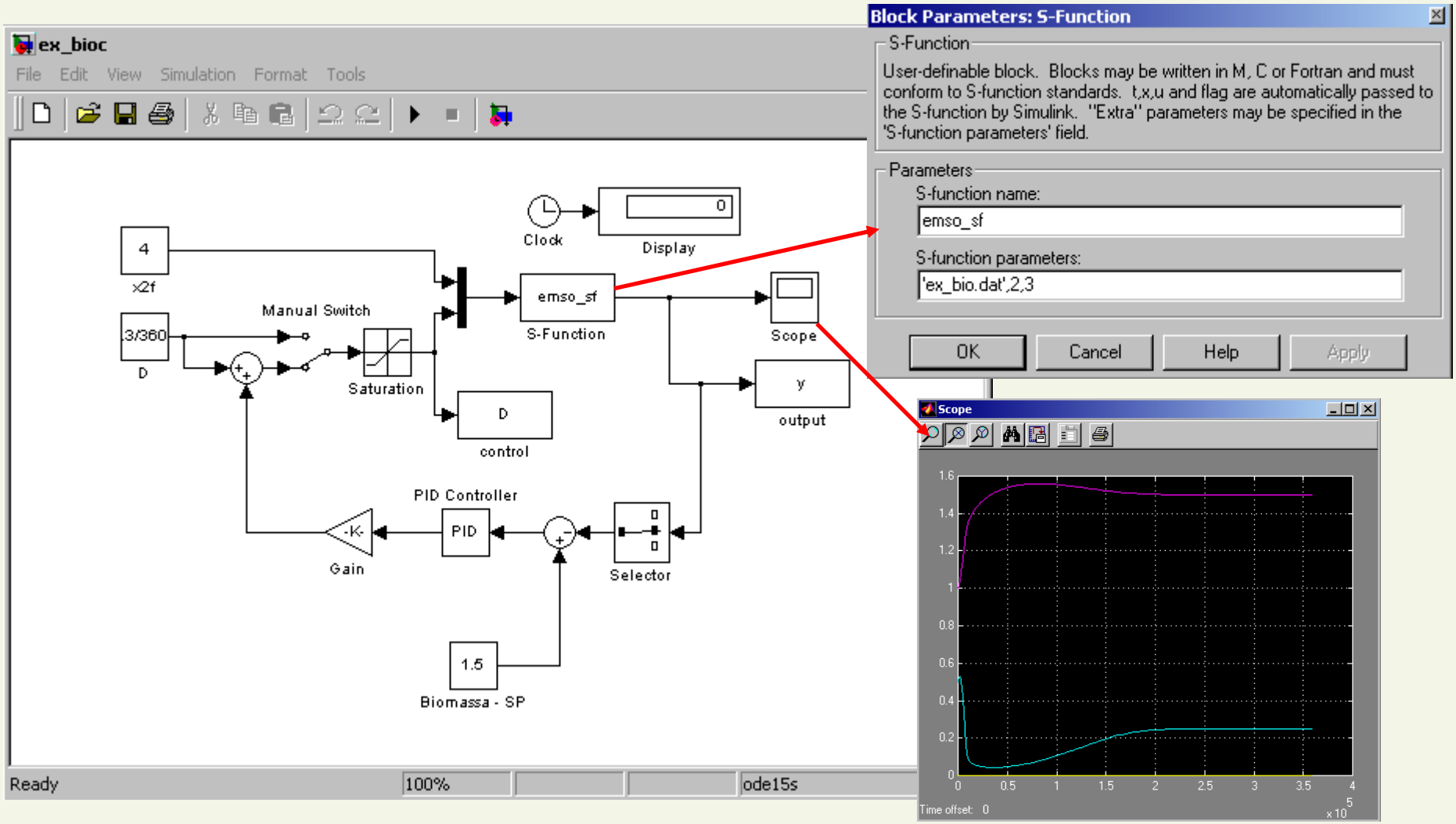
Interface EMSO-AUTO




```
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-MATLAB



Interface EMSO-MATLAB

EMSO

File Edit View Tasks Result Config Help

Zoom: 100

Explorer

- ex_bio.mso
  - using
  - bio
- eml
- sample
- tutorial
- Modelos
- Flowsheets
- ModeloHips

ex_bio.mso

```
21
22  using "types";
23
24  FlowSheet bio
25      PARAMETERS
26      miMax   as frequency (Brief="Monod parameter", DisplayUnit='1/h');
27      Km      as conc_mass (Brief="Monod parameter", Default=0.12);
28      K1      as inv_conc_mass (Brief="Monod parameter", Default=0.4545);
29      Y       as coefficient (Brief="Yield biomass/substrate", Default=0.4);
30      x2f     as conc_mass (Brief="Substrate feed concentration", Default=4);
31      D       as frequency (Brief="Dilution rate F/V", DisplayUnit='1/h');
32
33      VARIABLES
34      biomass   as conc_mass (Brief="Biomass concentration");
35      substrate as conc_mass (Brief="Substrate concentration");
36      mi        as frequency (Brief="Specific growth rate", Lower=-1e-3, DisplayUnit='1/h');
37
38      EQUATIONS
39      "Specific growth rate"
40      mi = miMax*substrate/(Km+substrate+(K1*substrate^2));
41
42      "Biomass production"
43      diff(biomass)/biomass = mi-D;
44
45      "Substrate consumption"
46      diff(substrate) = D*(x2f-substrate)-(biomass*mi/Y);
47
48      SET
49      miMax = 0.53  *  '1/h';
50      x2f = 4  * 'kg/m^3';
51      D   = 0.3 * '1/h';
52
53      INITIAL
54      biomass = 1   *  'kg/m^3';
55      substrate = 0.5   *  'kg/m^3';
56
57      OPTIONS
58      Dynamic = true;
59      TimeStep = 0.5;
60      TimeEnd = 15;
61      TimeUnit = 'h';
62      SparseAlgebra = true;
63  end
```

Problems | Console | Model

Output Level: Normal Output

Ready. | Mode: Text Editor | Mode: processor speed

UltraEdit-32 - [E:\User\Arge\...

File Edit Search Project View
Format Column Macro Advanced Window
Help

ex_bio.dat

```
1  ex_bio.mso
2  bio
3  2
4  x2f
5  D
6  3
7  mi
8  biomass
9  substrate
10 0
11 0
12
```

For Help, pre | Ln 1, Col. 1, CW | UNI:

Interface EMSO-CFD

EMSO

Energy Balance

Mass Balance

ρ, μ, Cp, k

U

INTERFACE

Overall heat transfer coefficient evaluation

ρ, μ

$V = f(z, r)$

PHOENICS

Momentum Balance