Process Analysis and Dynamic Simulation with EO-CAPE Tools

Argimiro R. Secchi
Chemical Engineering Program – COPPE/UFRJ
Rio de Janeiro, RJ - Brazil

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

(process design, process development, operator training, soft sensors, state estimation, model updating, RTO & DRTO, process monitoring & control, start-up & shut-down, enterprise wide optimization, planning & scheduling, parameter estimation, data reconciliation, sensitivity analysis, simulation, optimization)
 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)
All equipments or modules are simultaneously evaluated (Block decomposition can be used to explore sequential solution)

Open-source Modeling

Equipment contain only chemistry and physics of the model

ex: EMSO, Ascend, Jacobian, gPROMS, AspenDynamics, EcosimPro
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)
A process flowsheet model can be hierarchically decomposed:
Object-Oriented Modeling

Tray

abstract model

- mass balance
- energy balance
- thermodynamic equilibrium
- mol fraction normalization

concrete model (ideal tray)

+<br>

liquid flow model

vapor flow model

+<br>

efficiency model

concrete model (real tray)
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
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)
Streams
Inlet Material stream feeding the tank
Outlet Material stream leaving the tank

Parameters
k Valve constant
D Hydraulic diameter of the tank

Variables
A Tank cross section area
V Tank volume
h Tank level

Devices: source, tank, sink

Available model of the tank
>>> Model with circular cross section
>>> Model with square cross section
Inheritance

Model equations

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

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

liquid volume: \[ V = A \ h \]
Object-Oriented Modeling

Abstract model

- **mass balance:** \( F_{in} - F_{out} = \frac{dV}{dt} \)
- **valve equation:** \( F_{out} = k \sqrt{h} \)
- **liquid volume:** \( V = A \cdot h \)

**EMSO:**

```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
```
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

**Model** Tank_Square as Tank_Basic

**EQUATIONS**

"Cross section area" A = D^2;

end
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
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

```plaintext
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

```plaintext
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
```
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
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

<table>
<thead>
<tr>
<th>Categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTPI (Human toxicity potential by ingestion)</td>
</tr>
<tr>
<td>HTPE (Human toxicity potential by exposure)</td>
</tr>
<tr>
<td>ATP (Aquatic toxicity potential)</td>
</tr>
<tr>
<td>TTP (Terrestrial toxicity potential)</td>
</tr>
<tr>
<td>GWP (Global warming potential)</td>
</tr>
<tr>
<td>ODP (Ozone depletion potential)</td>
</tr>
<tr>
<td>PCOP (Photo chemical oxidation potential)</td>
</tr>
<tr>
<td>ARP (Acid rain potential)</td>
</tr>
<tr>
<td>EP (Eutrophication potential)</td>
</tr>
</tbody>
</table>
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 \( j \)

\( \omega_j \) weighting factor for each category \( j \)

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

...
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

<table>
<thead>
<tr>
<th>Impact category</th>
<th>% Contribution (top)</th>
<th>% Contribution (bottom)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human toxicity by ingestion</td>
<td>7.58</td>
<td>6.57</td>
</tr>
<tr>
<td>Human toxicity by exposure</td>
<td>0.58</td>
<td>5.23</td>
</tr>
<tr>
<td>Aquatic toxicity</td>
<td>7.58</td>
<td>6.57</td>
</tr>
<tr>
<td>Terrestrial toxicity</td>
<td>2.71</td>
<td>7.91</td>
</tr>
<tr>
<td>Global warming</td>
<td>0.34</td>
<td>0.1×10⁻³</td>
</tr>
<tr>
<td>Ozone depletion</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Photo chemical oxidation</td>
<td>81.21</td>
<td>73.72</td>
</tr>
<tr>
<td>Acid rain</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Eutrophication</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Results with pressure and level controllers

Disturbances:

- Pressure set-point at time = 2h
  - 5 atm \rightarrow 8 \text{ atm}
- Temperature and feed flow rate at time = 4h
  - 340 \text{ K} \rightarrow 360 \text{ K}
  - 500 \text{ kmol/h} \rightarrow 450 \text{ 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
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 consortilados.

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 abertos para 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 e distribui-lo gratuitamente para as universidades e parceiros (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 Simulation and Optimization.

EMSO é a sigla para Environment for Modeling Simulation and Optimization.

EMSO is the simulation software of the ALSOC project. Its development was started in 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 EMSO, veja o ChangeLog ou faça o seu download aqui!
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
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 an example and run it
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 –
Model equations

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

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

liquid volume:

\[
V = A \ h \\
V = \pi h^2 \left( \frac{D}{2} - \frac{h}{3} \right)
\]
Flowsheet

Source: $F^i = 20 \text{ m}^3/\text{h}$

- $D = 3 \text{ m}$
- $h(0) = 1 \text{ m}$

Sink:
- $D = 3 \text{ m}$
- $h(0) = 2.5 \text{ m}$

- $D = 3 \text{ m}$
- $h(0) = 2 \text{ m}$
Basic Elements in Modeling

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

- **Process Description**
  - Process objectives
  - Process flowsheet
  - Process operation
    - unit operations and control

- **Problem Definition**
  - Simulation objectives
  - Simulation applications
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.
• Bases to be used in the modeling

- mass conservation

\[ \frac{\partial \rho}{\partial t} = - (\nabla \cdot \rho \mathbf{v}) \]

- momentum conservation

\[ \frac{\partial (\rho \mathbf{v})}{\partial t} = - [\nabla \cdot \rho \mathbf{v} \mathbf{v}] - \nabla P - [\nabla \cdot \mathbf{\tau}] + \rho \mathbf{g} \]

\[ \text{advection} \quad \text{pressure forces} \quad \text{viscous forces} \quad \text{gravitational forces} \]

- energy conservation

\[ \frac{\partial}{\partial t} \left[ \rho \left( \dot{U} + \frac{1}{2} \mathbf{v}^2 \right) \right] = - \left( \nabla \cdot \rho \mathbf{v} \left( \dot{U} + \frac{1}{2} \mathbf{v}^2 \right) \right) - (\nabla \cdot q) + \rho (g \cdot \mathbf{v}) - (\nabla \cdot P \mathbf{v}) - (\nabla \cdot [\mathbf{\tau} \cdot \mathbf{v}]) \]

\[ \text{advection} \quad \text{conduction} \quad \text{gravit. forces work} \quad \text{pressure forces work} \quad \text{viscous forces work} \]
3. Simplifying Assumptions

- Establish the assumptions and simplifications
- Define the model limitations

- constant specific mass
- isothermal
- perfect mixture

\[- \frac{F_{\text{out}}}{k} = k \sqrt{h} \]
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
\[
\begin{align*}
\frac{dV}{dt} &= F \\
\frac{dX}{dt} &= \left(\mu - \frac{F}{V}\right)X \\
\frac{dT}{dt} &= \frac{F}{V}(T_e - T) - \frac{UA}{pVC_p}(T - T_c)
\end{align*}
\]

Parametric
\[
A(q)y(t) = \frac{B(q)}{F(q)}u(t) + \frac{C(q)}{D(q)}e(t)
\]

Neural Nets

Fuzzy Logic
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
mass balance: \[ F_{in} - F_{out} = \frac{dV}{dt} \]  \hspace{1cm} (1)

valve equation: \[ F_{out} = k \sqrt{h} \]  \hspace{1cm} (2)

liquid volume: \[ V = A \ h \]  \hspace{1cm} (3)

\[ A = \frac{\pi D^2}{4} \]  \hspace{1cm} (4)
5. Consistency Analysis

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

<table>
<thead>
<tr>
<th>variable</th>
<th>UOM</th>
<th>equations</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_{in}$, $F_{out}$</td>
<td>m$^3$ h$^{-1}$</td>
<td>(1): [m$^3$ h$^{-1}$] − [m$^3$ h$^{-1}$] = [m$^3$] / [h]</td>
</tr>
<tr>
<td>$V$</td>
<td>m$^3$</td>
<td>(2): [m$^3$ h$^{-1}$] = [m$^{2.5}$ h$^{-1}$] ([h])$^{0.5}$</td>
</tr>
<tr>
<td>$A$</td>
<td>m$^2$</td>
<td>(3): [m$^3$] = [m$^2$] [m]</td>
</tr>
<tr>
<td>$h$, $D$</td>
<td>m</td>
<td>(4): [m$^2$] = ([m])$^2$</td>
</tr>
<tr>
<td>$k$</td>
<td>m$^{2.5}$ h$^{-1}$</td>
<td></td>
</tr>
<tr>
<td>$t$</td>
<td>h</td>
<td></td>
</tr>
</tbody>
</table>
variables: $F_{in}$, $F_{out}$, $V$, $A$, $h$, $D$, $k$, $t$ $\Rightarrow$ 8
constants: $k$, $D$ $\Rightarrow$ 2
specifications: $t$ $\Rightarrow$ 1
driving forces: $F_{in}$ $\Rightarrow$ 1
unknown variables: $V$, $h$, $A$, $F_{out}$ $\Rightarrow$ 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: $h(0)$ $\Rightarrow$ 1
• Plan case studies
• Define:
  – Objectives of the study
  – Problems to be solved
  – Evaluation criteria

For the given example and initial condition \((h_0 \text{ or } V_0)\), 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

![Graph showing h_exp vs h_calc](image-url)
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
• 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.
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
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.
Differential index ($\nu$): 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: 
\[
\begin{align*}
    x_1' - x_2 &= 0 \\
    x_1 &= u(t) \\
\end{align*}
\]
\[
\begin{array}{c}
\text{differentiating twice in } t \\
\end{array}
\]
\[
\begin{align*}
    x_1' &= x_2 \\
    x_2' &= u''(t) \\
\end{align*}
\]

$\nu = 2$

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 $\nu > 1$. 
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.
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:

\[
\begin{align*}
 x'_1 - x_2 &= 0 \\
 x_1 &= u(t) \\
 x'_2 &= u''(t)
\end{align*}
\]

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 \( t_0 \).
Example: classical pendulum problem

Inconsistent initial condition:

\[
\begin{align*}
x' &= w \quad (1) \quad x(0) = 1 \quad x'(0) = 0 \\
y' &= z \quad (2) \quad y(0) = 0 \quad y'(0) = 0 \\
w' &= T \cdot x \quad (3) \quad w(0) = 0 \quad w'(0) = 1 \\
z' &= T \cdot y - g \quad (4) \quad z(0) = 0 \quad z'(0) = -g \\
x^2 + y^2 &= L^2 \quad (5) \quad T(0) = 1 \quad T'(0) = 0
\end{align*}
\]

\[
F(t, y, y') = 0 \quad F(0, y(0), y'(0)) = 0
\]

Hidden constraints:

Differentiating (5) and using (1) and (2):
\[
x \cdot w + y \cdot z = 0 \quad (6) \quad x(0) \cdot w(0) + y(0) \cdot z(0) = 0
\]

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

Differentiating (7) and using (2), (3), (4), (6):
\[
T' = -3 \frac{g \cdot z}{L^2} \quad (8) \quad T'(0) = -3 \frac{g \cdot z(0)}{L^2}
\]
Example: classical pendulum problem

\[ x' = w \quad (1) \]
\[ y' = z \quad (2) \]
\[ w' = T \cdot x \quad (3) \]
\[ z' = T \cdot y - g \quad (4) \]
\[ x^2 + y^2 = L^2 \quad (5) \]
\[ x \cdot w + y \cdot z = 0 \quad (6) \]
\[ w^2 + z^2 + T \cdot L^2 = g \cdot y \quad (7) \]
\[ T' = -3 \frac{g \cdot z}{L^2} \quad (8) \]

10 variables \((y, y')\)
8 equations
2 DDoF

Satisfies the inconsistent I.C.

But not any pair!
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”
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

\[ L = 0.9 \text{ m}, \ g = 9.8 \text{ m/s}^2 \ .. \ I.C.: x(0) = 0.9 \text{ m} \ \text{and} \ w(0) = 0 \]
Batch Distillation

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

\[
\frac{dH_0}{dt} = - \frac{V}{R + 1}
\]  \hspace{1cm} (1)

Component:

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

Mass balance in each tray component:

\[
\frac{dx_i^j}{dt} = \frac{V}{H_i} \left[ y_{i-1}^j - y_i^j + \frac{R}{R + 1} (x_{i+1}^j - x_i^j) \right] \hspace{1cm} i = 1, \ldots, np \hspace{1cm} j = 1, \ldots, nc - 1
\]  \hspace{1cm} (3)
Batch distillation modeling

Mass balance in the condenser

Component:

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

(4)

Molar fractions

\[
\sum_{j=1}^{nc} y_i^j = 1 \quad i = 0, \ldots, np + 1
\]  

(5)

\[
\sum_{j=1}^{nc} x_i^j = 1 \quad i = 0, \ldots, np + 1
\]  

(6)

Thermodynamic equilibrium

\[
y_i^j P = P_{ref} \exp \left( A_j \frac{B_j}{T_i + C_j} \right) x_i^j \quad i = 0, \ldots, np + 1
\]  

(7)

\[
j = 1, \ldots, nc
\]
### Consistency analysis

<table>
<thead>
<tr>
<th>variable</th>
<th>units of measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>$H_i$</td>
<td>kmol</td>
</tr>
<tr>
<td>$V$</td>
<td>kmol/s</td>
</tr>
<tr>
<td>$t$</td>
<td>s</td>
</tr>
<tr>
<td>$R$</td>
<td>–</td>
</tr>
<tr>
<td>$x_i$, $y_i$</td>
<td>kmol/kmol</td>
</tr>
<tr>
<td>$P$, $P_{\text{ref}}$</td>
<td>kPa</td>
</tr>
<tr>
<td>$T_i$</td>
<td>K</td>
</tr>
<tr>
<td>$A_j$</td>
<td>–</td>
</tr>
<tr>
<td>$B_j$</td>
<td>K</td>
</tr>
<tr>
<td>$C_j$</td>
<td>K</td>
</tr>
</tbody>
</table>
Consistency analysis

variables: \( H_i, V, t, R, x^i_j, y^i_j, P, P_{ref}, T_i, A_j, B_j, C_j \) \( \Rightarrow \) \( 5 + 2 (np+2)(nc+1) + 3 \ nc \)

constants: \( P_{ref}, A_j, B_j, C_j \) \( \Rightarrow \) \( 3 \ nc + 1 \)

specifications: \( t, V, P, H_i, x^{1\_np+1} \) \( \Rightarrow \) \( 5 + np \) \( (i = 1,...,np+1) \)

driving forces: 0

unknown variables: \( H_0, R, x^i_j, y^i_j, T_i \) \( \Rightarrow \) \( 3 + 2 (np+2) \ nc + np \)

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 \( nc = 2: H_0(0), R(0), x_0^{\_1}(0), T_i(0) \) \( (i = 2,...,np–2) \)
**VARIABLES**

- \( H_0 \) 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**

"Balâncio no Prato 0"

\[
\text{diff}(H0)'s' = -V/(R+1);
\text{diff}(x1(1))'s' = V/H0 * (x1(1) - y1(1)) + R * ((x1(2) - x1(1))/(R+1));
\]

"Balâncio demais pratos"

\[
\text{diff}(x1(2:Np))'s' = V/Hi * (y1(1:Np-1) - y1(2:Np)) + (R * (x1(3:Np+1) - x1(2:Np)))/(R+1));
\]

"Balâncio último prato"

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

"Equação de Controle Ótimo"

\[
x1(Np+1) = \text{purity};
\]

**INITIAL**

\[
H0 = 100;
x1(1) = 0.55;
T(2:9) = [89.8, 87.5, 85.4, 83.8, 82.6, 81.7, 81.1, 81.0];
R = 1;
\]
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 = 750;
  Hi = 1;
  V = 120;
OPTIONS
  TimeStep = 0.01;
  TimeEnd = 2.1;
  Integration = "original";
  # Integration = "index0";
  DAESolver(
    File = "dasslc" # "mebdf"
  );
end
AspenDynamics

- Reports system singularity:
Detects a high-index problem and gives the following error message:

```
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, for example, you cannot specify arbitrary initial values for the differential variable(s):
```

`gPROMS>`
5. System Analysis

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

\[ F_e, C_{Af}, C_{Bf}, T_f \]

\[ F_{we}, T_{we} \]

\[ V, T \]

\[ F_{ws}, T_w \]

\[ F_s, C_A, C_B, T \]

\[ A \xrightarrow{k} B \]
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 $6 \times 10^4$ 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.
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;
- \(\Delta\) (internal energy) \(\approx\) \(\Delta\) (enthalpy);
- negligible variation of potential and kinetic energies;
- constant volume in the jacket;
- thin metallic wall with negligible heat capacity.
Mass balance in the reactor

Overall:

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

\[
\frac{dV}{dt} = F_e - F_s
\]  \hspace{1cm} (1)

Component:

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

\[
V \frac{dC_A}{dt} = F_e (C_{A_f} - C_A) - (-r_A) V
\]  \hspace{1cm} (2)

\[
\tau = \frac{V}{F_e}
\]  \hspace{1cm} (3)
Energy balance in the reactor:

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

where \( \hat{H} = \hat{U} + P \hat{V} \)

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

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

\[
\rho VC_p \frac{dT}{dt} = F_e \rho C_p (T_f - T) + q_r - q
\]

(4)
where

\[ q = U A_t (T - T_w) \] (5)

\[ q_r = (-\Delta H_r) V (-r_A) \] (6)

\[ (-r_A) = k C_A \] (7)

\[ k = k_0 \exp(-E/RT) \] (8)

\[ A = \pi D^2/4 \] (9)

\[ V = A h \] (10)

\[ A_t = A + \pi D h \] (11)

\[ F_s = x \ C_v \ \sqrt{h} \] (12)

\[ x = f(h) \] Level control (13)

\[ T_w = f(T) \] Temperature control (14)
# Consistency analysis

<table>
<thead>
<tr>
<th>variable</th>
<th>units of measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>$F_e$, $F_s$</td>
<td>m³ s⁻¹</td>
</tr>
<tr>
<td>$V$</td>
<td>m³</td>
</tr>
<tr>
<td>$t$, $\tau$</td>
<td>s</td>
</tr>
<tr>
<td>$C_A$, $C_{Af}$</td>
<td>kmol m⁻³</td>
</tr>
<tr>
<td>$r_A$</td>
<td>kmol m⁻³ s⁻¹</td>
</tr>
<tr>
<td>$\rho$</td>
<td>kg m⁻³</td>
</tr>
<tr>
<td>$C_p$</td>
<td>kJ kg⁻¹ K⁻¹</td>
</tr>
<tr>
<td>$T$, $T_f$, $T_w$</td>
<td>K</td>
</tr>
<tr>
<td>$q_r$, $q$</td>
<td>kJ s⁻¹</td>
</tr>
<tr>
<td>$U$</td>
<td>kJ m² K⁻¹ s⁻¹</td>
</tr>
<tr>
<td>$A_p$, $A$</td>
<td>m²</td>
</tr>
<tr>
<td>$h$, $D$</td>
<td>m</td>
</tr>
<tr>
<td>$C_v$</td>
<td>m².⁵ h⁻¹</td>
</tr>
<tr>
<td>$\chi$</td>
<td>–</td>
</tr>
<tr>
<td>$\Delta H_r$, $E$</td>
<td>kJ kmol⁻¹</td>
</tr>
<tr>
<td>$R$</td>
<td>kJ kmol⁻¹ K⁻¹</td>
</tr>
<tr>
<td>$k$, $k_0$</td>
<td>s⁻¹</td>
</tr>
</tbody>
</table>
Consistency analysis

variables: $F_e, F_s, V, t, C_A, C_{Af}, r_A, \rho, Cp, T, T_f, T_w, q_r, q, U, A_p, A, h, D, Cv, x, \Delta Hr, E, R, k, k_0, \tau \rightarrow 27$

constants: $\rho, Cp, U, D, Cv, \Delta Hr, E, R, k_0 \rightarrow 9$

specifications: $t \rightarrow 1$

driving forces: $F_e, T_f, C_{Af} \rightarrow 3$

unknown variables: $F_s, V, C_A, r_A, T, T_w, q_r, q, A, A_p, h, x, k, \tau \rightarrow 14$

equations: 14


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

Needs 3 initial condition: $h(0), C_A(0), T(0) \rightarrow 3$
Running EMSO

Open MSO file
Consistency Analysis

Results
### Process with uncontrolled CSTR and multiple steady-states

#### FlowSheet CSTR_no_control

```plaintext
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/mA2/k';
CSTR1.ro = 1000 * 'kg/mA3';
CSTR1.Bo = 4 * 'kJ/kg/k';
CSTR1.Hr = -7000 * 'kJ/kmol';
CSTR1.Ea = 664 * 'kJ/kmol';
CSTR1.ka = 89 * '1/s';
CSTR1.KD = 1.2 * 'm';
CSTR1.CV = 2.7 * 'm3A.5/h';

EQUATIONS
"Manipulated Variables"
CSTR1.x = 1;
CSTR1.Tw = 300 * 'K';

"Feed Stream"
FEED.Ca = 300 * 'kmol/mA3';
FEED.F = 3.5 * 'm3A/h';

# Disturbance
if time < 50 * 'h' then
  "Feed Temperature" FEED.T = 300 * 'K';
else
  "Feed Temperature" FEED.T = 550 * 'K';
end

INITIAL
"Concentration" CSTR1.Ca = 50 * 'kmol/mA3';
"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
```
The CSTR example at the steady state satisfy:

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

\[
C_A = \frac{C_{Af}}{1 + \tau k_0 e^{-\frac{E}{RT}}}
\]

\[
\tau = \frac{V}{F_e}
\]
Rewriting the energy balance:

\[ Q_R(T) = Q_G(T) \]

\[ Q_R(T) = aT - b \]

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

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

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

stable:

\[ \frac{dQ_R}{dT} > \frac{dQ_G}{dT} \]

unstable:

\[ \frac{dQ_R}{dT} < \frac{dQ_G}{dT} \]
Multiplicty of Steady States

Path Following

\[ \frac{dx}{dt} = F(t, x) \quad \rightarrow \quad F(x) = 0 \]

Newton-Raphson:

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

\[ J_{ij}(x^{(k)}) = \frac{\partial F_i(x^{(k)})}{\partial x_j} \quad m \leq k \quad \text{and} \quad 0 < \alpha \leq 1 \]

Homotopic Continuation:

\[ H(x; p) = (1 - p) F(x) + p G(x) = 0 \quad , \quad 0 \leq p \leq 1 \]

\[ G(x) = J(x^{(0)}) (x - x^{(0)}) \quad \text{affine homotopy} \]

\[ G(x) = F(x) - F(x^{(0)}) \quad \text{Newton homotopy} \]

Multiples solutions can be obtained by continuously varying the parameter \( p \)
Parametric Continuation: \( F[x(s); p(s)] = 0 \)

where \( s \) is some parameterization, e.g., path arc length

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

\( DF = \begin{bmatrix} \frac{\partial F}{\partial x} & \frac{\partial F}{\partial p} \end{bmatrix} \) Frechet derivative

a point \((x_0, p_0)\) is:

- Regular if \( \frac{\partial F(x_0, p_0)}{\partial x} \) is non-singular

- Turning point if \( \frac{\partial F(x_0, p_0)}{\partial x} \) is singular and \( DF \) has rank = \( n \) reparameterization

- Bifurcation if \( \frac{\partial F(x_0, p_0)}{\partial x} \) is singular and \( DF \) has rank < \( n \)
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
Generate linearized model at given operating point.

Implicit DAE: \[ F(\ddot{x}', \dot{x}, t) = 0 \]

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 \( y(t) \):

\[ F(x', x, y, u, t) = 0 \]
Differentiating F:

\[
F_x' \, dx' + F_x \, dx + F_y \, dy + F_u \, du = 0
\]

and extracting:

\[
\begin{bmatrix}
\frac{dx'}{dy}
\end{bmatrix} = - \begin{bmatrix} F_x' & F_y \end{bmatrix}^{-1} \begin{bmatrix} F_x & F_u \end{bmatrix} \begin{bmatrix}
\frac{dx}{du}
\end{bmatrix}
\]

(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:

\[
x' = A \, x + B \, u
\]

\[
y = C \, x + D \, u
\]
Test example for a linear model: exact solution!

```
using "types";
FlowSheet linear :
PARAMETERS
nx as Integer(Default=2);
y as Integer(Default=2);
u 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)' = sum(A*x) + sum(B*u);
y = sum(C*x) + sum(D*u);
SPECIFY
u(1:nu) = sin((time/'s' + [1:nu])*'rad');
```
Verifying the results for the linear model:

```plaintext
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
```

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 0.500000 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 0.500000
B:
0.750000 0.000000 0.000000 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 0.750000 0.000000 0.000000 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 0.750000 0.000000 0.000000
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 0.750000
C:
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
0.850000 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
0.900000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000
0.950000 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 0.950000 0.950000 0.950000 0.950000 0.950000 0.950000
Simulation of 'linear' finished successfully in 0.234 seconds.
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.
Liapunov Stability: $\bar{x}(t)$ is stable (or Liapunov stable) if, given $\varepsilon > 0$, there exists a $\delta = \delta(\varepsilon) > 0$, such that, for any other solution, $y(t)$, of
\[
\frac{dx}{dt} = F(x) \quad \text{satisfying} \quad |\bar{x}(t_0) - y(t_0)| < \delta, \quad \text{then} \quad |\bar{x}(t) - y(t)| < \varepsilon \quad \text{for} \quad t > t_0.
\]
**Asymptotic Stability**: $\bar{x}(t)$ is asymptotically 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: $y(t) = x(t) - \bar{x}(t)$

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:
$$\frac{dy}{dt} = J[\bar{x}(t)] \cdot y = A(t) \cdot y$$
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 $> 0$. 
For a second-order linear system:

\[
\det(A - \lambda I) = \lambda^2 - tr(A) \cdot \lambda + \det(A) = 0
\]

\[
\lambda = \frac{tr(A) \pm \sqrt{tr(A)^2 - 4 \det(A)}}{2}
\]
Considering the CSTR example with constant volume:

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

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

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

\[ J(13.13, 659.46) = \begin{bmatrix} -1.6458 \times 10^{-3} & -3.4282 \times 10^{-4} \\ 2.7542 \times 10^{-3} & 4.8934 \times 10^{-4} \end{bmatrix} \quad \lambda = \begin{bmatrix} -1.0205 \times 10^{-3} \\ -1.3604 \times 10^{-4} \end{bmatrix} \]

2) Saddle Point, unstable

\[ J(132.87, 523.01) = \begin{bmatrix} -1.6260 \times 10^{-4} & -3.1753 \times 10^{-4} \\ 1.5852 \times 10^{-4} & 4.4509 \times 10^{-4} \end{bmatrix} \quad \lambda = \begin{bmatrix} -6.3659 \times 10^{-5} \\ 3.4614 \times 10^{-4} \end{bmatrix} \]

3) Stable Node

\[ J(299.86, 332.72) = \begin{bmatrix} -7.2050 \times 10^{-5} & -6.6220 \times 10^{-7} \\ 5.9285 \times 10^{-8} & -1.0944 \times 10^{-4} \end{bmatrix} \quad \lambda = \begin{bmatrix} -7.2051 \times 10^{-5} \\ -1.0944 \times 10^{-4} \end{bmatrix} \]
Complex Dynamic Behavior

CSTR example:

- Stable solutions
- Unstable solutions

Hopf point

$T_w = 200.37 \text{ K}$
A limit cycle is stable if all characteristics values of $\exp(Jp)$ (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 $\beta = |\lambda_{Hopf}|$. 

unstable limit cycle

file: CSTR_auto/cstr_bif.mso

Complex Dynamic Behavior
Interface EMSO-AUTO

- Parameters
- Equation system
- Jacobian matrix
- First steady-state solution

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
\[ \text{diff}(u1)'s' = -u1 + p1 \times (1 - u1) \times u3; \]
\[ \text{diff}(u2)'s' = -u2 + p1 \times p2 \times (1 - u1)' \times u3 - p3 \times u2; \]
\[ u3 = \exp(u2); \]
\[
\frac{dx_1(t)}{dt} = -x_1(t) + p \cdot [1 - x_1(t)] e^{x_2(t)}
\]

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

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

\[p = 0: \quad x^* = (0, 0) \quad \lambda(J) = (-1, -3)\]
<table>
<thead>
<tr>
<th>Parameter $p$</th>
<th>Eigenvalues</th>
<th>Phase plane</th>
</tr>
</thead>
</table>
| $p < 0.06361$           | Real negatives eigenvalues – **stable node**  
|                         | $p = 0.05$  \( \lambda = [-1.13, -2.06] \)                             | ![Phase plane 1](image1.png) |
| $p = 0.06361$           | Repeated real negatives eigenvalues – **stable node (star)**  
|                         | \( \lambda = [-1.4372, -1.4372] \)                                 | ![Phase plane 2](image2.png) |
| $0.06361 < p < 0.0889$  | Complex eigenvalues with negative real part - **stable focus**  
<p>|                         | $p = 0.085$  ( \lambda = -1.095 \pm 0.565 \ i )                      | <img src="image3.png" alt="Phase plane 3" /> |</p>
<table>
<thead>
<tr>
<th>$p = 0.0889$</th>
<th>$p &gt; 0.0889$</th>
<th>Turning point (fold): One stable solution (focus) and other unstable (node). (point 3 in figure below)</th>
</tr>
</thead>
<tbody>
<tr>
<td>unstable node gives rise to two points: unstable node and saddle point $p &gt; 0.0889$</td>
<td>$\lambda = -1.009 \pm 0.605 , i$ $\lambda = [0,, 3.432]$</td>
<td>$\lambda = -1.009 \pm 0.605 , i$ $\lambda = [0,, 3.432]$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$0.0889 &lt; p &lt; 0.0933$</th>
<th>$p = 0.09$</th>
<th>$\lambda = -0.982 \pm 0.614 , i$ $\lambda = [-0.213,, 3.332]$ $\lambda = [0.364,, 3.151]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>One stable solution (focus) and two unstable (saddle and node)</td>
<td>$\lambda = -0.982 \pm 0.614 , i$ $\lambda = [-0.213,, 3.332]$ $\lambda = [0.364,, 3.151]$</td>
<td>$\lambda = -0.982 \pm 0.614 , i$ $\lambda = [-0.213,, 3.332]$ $\lambda = [0.364,, 3.151]$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$0.0933 &lt; p &lt; 0.10574$</th>
<th>$p = 0.10$</th>
<th>$\lambda = -0.652 \pm 0.651 , i$ $\lambda = [-0.439,, 1.953]$ $\lambda = 1.431 \pm 1.851 , i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>at $p = 0.105738931$ the first point goes from stable focus to stable node: $\lambda = [-0.055,, -0.046]$</td>
<td>$\lambda = -0.652 \pm 0.651 , i$ $\lambda = [-0.439,, 1.953]$ $\lambda = 1.431 \pm 1.851 , i$</td>
<td>$\lambda = -0.652 \pm 0.651 , i$ $\lambda = [-0.439,, 1.953]$ $\lambda = 1.431 \pm 1.851 , i$</td>
</tr>
</tbody>
</table>
$p = 0.10574$
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 stable 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)

$\lambda = \pm 4.008 i$
For $p > 0.1309$, Complex eigenvalues with negative real part - stable focus

$p = 0.15$

$\lambda = -0.952 \pm 4.627 i$
Interface EMSO-AUTO

Hopf
1st turning point
2nd turning point

Trajectories:
- stable point
- saddle point
- unstable point

Re(\lambda)
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:** \( W_x = \frac{\partial x_i}{\partial p_j} \bigg|_{\bar{x}, \bar{p}} \quad W_y = \frac{\partial y_i}{\partial p_j} \bigg|_{\bar{x}, \bar{p}} \)
- **Global:** bifurcation diagram, surface response (case study)

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

Normalized form:
\[
W_y = \frac{\bar{p}_j}{\bar{y}_i} \frac{\partial y_i}{\partial p_j} \bigg|_{\bar{x}, \bar{p}}
\]
Sensitivity Analysis

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

\[ y = H(x, u; p) \]

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

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

where

\[ W_x(t) = \frac{\partial x}{\partial p} \quad \hat{W}_x(t) = \frac{dW_x}{dt} \]
**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

fl1OutletV.z(1);

fl1OutletV.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

<table>
<thead>
<tr>
<th></th>
<th>b0</th>
<th>b1</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>-0.1836734619</td>
<td>-0.472303207</td>
</tr>
<tr>
<td>Y</td>
<td>0.857142822</td>
<td>2.204081633</td>
</tr>
<tr>
<td>Status</td>
<td>OK!</td>
<td>OK!</td>
</tr>
</tbody>
</table>
- 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
Example: FlashDinamicoSemPID_PFD.mso
• Build the Model – cont.

```matlab
SET

NComp = PP.NumberOfComponents;
FL.diameter = 3 * 'm';
FL.orientation = 'vertical';

SPECIFY

Src.Composition(1) = 0.8;
Src.Composition(2) = 0.2;
Src.F = 600 * 'kmol/h';
Src.T = 338 * 'K';
Src.P = 507 * 'kPa';
Q.OutletQ = 0.0 * 'kw';
FL.OutletL.F = 370 * 'kmol/h';
FL.OutletV.F = 150 * 'kmol/h';

INITIAL

FL.Level = 0.4 * 'm';
FL.OutletL.T = 338 * 'K';
FL.OutletL.z(1) = 0.5;

OPTIONS

Dynamic = true;
Timestep = 0.1;
TimeEnd = 60;
TimeUnit = 'min';
Integration = 'original';
```

Input variables: specifications in EMSO
• Output variables: calculated by EMSO
• Interface configuration – EMSO-Matlab
- Build System in Simulink – without PID
• Configuring size of i/o ports
Interface EMSO-MATLAB
- Executing script in Matlab
• Visualizing Results

Input variables

Output variables
Interface EMSO-MATLAB

- Build System in Simulink – with PID
Configuring size of i/o ports
• Visualizing Results

Input variables

Output variables
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.
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
Graph for variable-equation relationship

\[
\begin{align*}
  f_1(x_1) &= 0 \\
  f_2(x_1, x_2) &= 0 \\
  f_3(x_1, x_2) &= 0 \\
  f_4(x_2, x_3, x_4) &= 0 \\
  f_5(x_4, x_5) &= 0 \\
  f_6(x_3, x_4, x_5) &= 0 \\
  f_7(x_5, x_6, x_7) &= 0
\end{align*}
\]

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
A Simple Example

\[ x_1' - x_2' = a(t) \]
\[ x_2 = b(t) \]

Solution:
\[ x_1(t) = x_1(0) + \int_0^t a(\tau) d\tau + b(t) \]
\[ x_2(t) = b(t) \]

✓ 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
Classic Algorithm

\[ x_1' - x_2' = a(t) \]
\[ x_2 = b(t) \]

- Who are the states?
- Which variables should be specified as initial conditions?
\[ x_1' - x_2' = a(t) \]
\[ x_2 = b(t) \]

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

<table>
<thead>
<tr>
<th>Set up of simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>All 2 variables will be monitored during this simulation!</td>
</tr>
<tr>
<td>The number of initial conditions (1) does not match the number of states (2)</td>
</tr>
<tr>
<td>Building mathematical problem description took 0 seconds.</td>
</tr>
</tbody>
</table>
$x_1' - x_2' = a(t)$

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

$x_2 = b(t)$

---

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.
\[ x'_1 - x'_2 = a(t) \]
\[ x_2 = b(t) \]

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
\textit{Debugging Techniques}  
– New Algorithm: debugging DAE system –

\begin{verbatim}
\text{DAESystems}(G = (V_e, V_v, E), M)
    1:    M \leftarrow \emptyset 
    2: \textbf{for} v_e \in V_e \textbf{do}
    3:        \textbf{if not} AugmentMatching2(G = (V_e, V_v, E), M, v_e, \text{false}) \textbf{then}
    4:            \text{mark all colored } v_k \in V_e 
    5:            \text{uncolour } V_e 
    6:        \textbf{if not} AugmentMatching2(G = (V_e, V_v, E), M, v_e, \text{true}) \textbf{then}
    7:            \text{return false}
    8:        \textbf{end if}
    9:        \text{diff all marked } v_k \in V_e 
10:    \textbf{else}
11:        \text{uncolour } V_e 
12:    \textbf{end if}
13: \textbf{end for}
14: \text{return true}
\end{verbatim}
AugmentMatching2(G = (Ve ∪ Vv, E), M, ve, alg)

1: colour ve
2: if exists \{ve, vv\} ∈ E and \{ve, vv\} ∉ M and vv is eligible then
3: \quad M ← M ∪ \{ve, vv\}
4: \quad return true
5: end if
6: for all \{ve, vv\} ∈ E do
7: \quad if exists \{ve2, vv\} ∈ M and ve2 not colored and vv is eligible then
8: \quad \quad if AugmentMatching2(G = (Ve ∪ Vv, E), M, ve2, alg) then
9: \quad \quad \quad M ← M ∪ \{ve, vv\}
10: \quad \quad \quad return true
11: \quad \quad end if
12: \quad end if
13: end for
14: return false
$x_1' - x_2' = a(t)$

$x_2 = b(t)$

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

\[ x_1' - x_2' = a(t) \]
\[ x_2 = b(t) \]
Debugging Techniques
– Applying the New Algorithm: high-index –

\[
\begin{align*}
x' &= w & (1) \\
y' &= z & (2) \\
w' &= T \cdot x & (3) \\
z' &= T \cdot y - g & (4) \\
x^2 + y^2 &= L^2 & (5)
\end{align*}
\]

only two states!
Dynamic model of a distillation column for the separation of isobutane from a mixture of 13 compounds

<table>
<thead>
<tr>
<th>N. Trays</th>
<th>N. Variables</th>
<th>Time* (s)</th>
<th>Time ( /N^2 ) (s ( \cdot 10^9 ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>2157</td>
<td>0.04</td>
<td>9.46</td>
</tr>
<tr>
<td>40</td>
<td>3877</td>
<td>0.14</td>
<td>9.58</td>
</tr>
<tr>
<td>80</td>
<td>7317</td>
<td>0.52</td>
<td>9.79</td>
</tr>
</tbody>
</table>

* Pentium M 1.7 GHz PC with 2 MB of cache memory, Ubuntu Linux 6.06
Other CAPE tools
Optimization (NLP)

```plaintext
Optimization FlashOpt1 as FlashSteadyTest

MAXIMIZE
  leven;

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

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

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

end

Optimization hs71

VAR
  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*x2*x3*x4 * (x1+x2+x3+x4) + 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)

```plaintext
Optimization minlp1

VARtAABLES
x0 as Integer(Defaut=0, Lower=0, Upper=1);
x1 as Real(Defaut=0, Lower=0, Upper=1.0e10);
x2 as Real(Defaut=0, Lower=0, Upper=1.0e10);
x3 as Integer(Defaut=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;
NLPSoLvelnLA = true;
NLPSoLvler(File = "minlp_emso", derivative_test = "second-order", print_level = 5);

end
```
## Parameter Estimation

**Estimation Biop_NE_Estt5 as Biop_NE_process5t**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>START</th>
<th>LOWER</th>
<th>UPPER</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_{ss}$</td>
<td>0.2009</td>
<td>0.004</td>
<td>0.004</td>
<td>kg/m$^3$;</td>
</tr>
<tr>
<td>$K_{sm}$</td>
<td>0.0446</td>
<td>0.005</td>
<td>0.005</td>
<td>kg/m$^3$;</td>
</tr>
<tr>
<td>$m_{im}$</td>
<td>0.7979</td>
<td>0.1</td>
<td>0.8</td>
<td>1/s;</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>2.0293</td>
<td>1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>$\gamma$</td>
<td>0.08502</td>
<td>0.05</td>
<td>5</td>
<td>1/s;</td>
</tr>
<tr>
<td>$K_1$</td>
<td>0.4059</td>
<td>0.1</td>
<td>3</td>
<td>kg/kg;</td>
</tr>
<tr>
<td>$K_2$</td>
<td>-0.00795</td>
<td>-1</td>
<td>3</td>
<td>1/s;</td>
</tr>
<tr>
<td>$Y_n$</td>
<td>10.62</td>
<td>0.1</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>$k_d$</td>
<td>0.007</td>
<td>0.0005</td>
<td>1</td>
<td>1/s;</td>
</tr>
</tbody>
</table>

**Experiments**

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

**Options**

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

**NLPSolver**

```
MaxIterations = 1000,  
File = "complex"  
#File = "ioprt_emso"
);
```

**Dynamic = true**

---

**Estimation PV_Est as PV_Flow**

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>START</th>
<th>LOWER</th>
<th>UPPER</th>
<th>UNIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1.5</td>
<td>-1</td>
<td>10</td>
<td>K</td>
</tr>
<tr>
<td>B</td>
<td>1000</td>
<td>800</td>
<td>3000</td>
<td>K</td>
</tr>
<tr>
<td>C</td>
<td>50</td>
<td>20</td>
<td>200</td>
<td>K</td>
</tr>
</tbody>
</table>

**Experiments**

```
# FILE  WEIGTH
"pv_est.dat"  1;
```

**Options**

```
NumJac = false;
```

**NLPSolver**

```
File = "ioprt_emso"  
#File = "complex"
);
```

```
Dynamic = false;
```
Data Reconciliation

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

Reconciliation HeatEx_Rec as HeatEx_Flow

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

FREE
x1; x2;

EXPERIMENTS
FILE "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
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!
Operator Training and Process Monitoring
Virtual Analyzer – EMSO-CEKF
Model parameters and variables selection
Parameters and variables configuration
Model Generation for MPC

- **Inferences**
- **Model updating for MPC**
- **Data treatment and data reconciliation**
- **Local optimization**
- **MPC**
- **Process + Regulatory Control**

Model Server (rigorous, empiric, hybrid, reduced)
Model Generation for MPC

Variables selection
Variables configuration

**Image Description:**
- The image shows a software interface for modeling and control (MPC) with a focus on variables configuration.
- The interface has a section labeled "Feed" with options for converting units, steady-state value source, and manual input.
- The "Steady-state value source" section includes options for selecting data from a result file, a tag, or manually.
- The "Manual" option includes fields for tagging and server information.
- The "Steady-State Value" field indicates "Not processed yet."
Standard Interfaces

CAFE-OPEN

Simulator

Executive

Libraries

Unit

Numerics

Thermo

External UO

External Thermo

External Numeric

U: Unit Interfaces
T: Thermo Interfaces
N: Numeric Interfaces
Example of CAPE-OPEN: DyOS (Dynamic Optimization Software) - Marquardt’s group (2000)
Another example of CAPE-OPEN: EMSO (Environment for Modeling, Simulation and Optimization) - Soares and Secchi (2004)

methanol plant
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.
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!


• 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.

DAE Solvers:


... thank you for your attention!

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

Process Modeling, Simulation and Control Lab
- Prof. Dr. Argimiro Resende Secchi
- Phone: +55-21-2562-8301
- 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**
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
 boilers.

- **FlowSheet Components**: This section likely introduces components used in the EMSO FlowSheet, which is a tool for modeling and simulating chemical processes.

- **Simulation options**: This part might discuss the options for setting up simulations within the EMSO environment.

- **Degree of Freedom**: This term is used in both control systems and chemical engineering to describe the number of independent variables that can change without affecting the system's overall dynamics.

- **Dynamic Degree of Freedom**: This is a specific type of degree of freedom that considers the system's response over time, often used in dynamic simulations.

- **Parameters of DEVICES**: This refers to the specific parameters that can be set for each device in the FlowSheet, allowing for detailed modeling and simulation.

The slide seems to be a part of an EMSO tutorial or guide, providing a visual and textual explanation of how to use the software for modeling and simulating chemical processes.
EMSO Tutorial
– Checking Units of Measurement –

Incompatible units:

- CSTR1.D = 3.2 * 'm';
- CSTR1.Cv = 2.7 * 'm^2.5/h';
- CSTR1.ko = 89 * '1/s';
- CSTR1.x = 1;
- CSTR1.Tw = 300 * 'm';

FEED.Ca = 300 * 'kmol/m^3';
FEED.F = 3.5 * 'm^2/h';
Flash multi-component

\[ F, z, P_f, T_f \]

\[ V, y \]

\[ L, x \]
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.
• negligible vapor holdup (no dynamics in vapor phase);
• thermodynamic equilibrium (ideal stage);
• no droplet drag in vapor stream;
• negligible heat loss to surroundings;
• $\Delta$(internal energy) $\approx \Delta$(liquid-phase enthalpy);
• perfect mixture in both phases.
Overall mass balance (molar base):

\[ \frac{dm}{dt} = F - V - L \]  \hspace{1cm} (1)

Component mass balance:

\[ \frac{d}{dt}(m x_i) = F z_i - V y_i - L x_i \]  \hspace{1cm} (2) \hspace{0.5cm} i = 1, 2, \ldots, C

Equilibrium:

\[ y_i = K_i x_i \]  \hspace{1cm} (3) \hspace{0.5cm} i = 1, 2, \ldots, C

\[ K_i = f(T, P, x, y) \]  \hspace{1cm} (4) \hspace{0.5cm} i = 1, 2, \ldots, C

Molar fraction:

\[ \sum_{i=1}^{C} x_i = 1 \]  \hspace{1cm} (5)
Energy balance:

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

Enthalpies:

\[ h = f(T, P, x) \]  \tag{7}

\[ H = f(T, P, y) \]  \tag{8}

\[ h_f = f(T_f, P_f, z) \]  \tag{9}

Controllers:

\[ L = f(m) \]  \tag{10}

\[ V = f(P) \]  \tag{11}
<table>
<thead>
<tr>
<th>variable</th>
<th>units of measurement</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m$</td>
<td>kmol</td>
</tr>
<tr>
<td>$F, L, V$</td>
<td>kmol s$^{-1}$</td>
</tr>
<tr>
<td>$t$</td>
<td>s</td>
</tr>
<tr>
<td>$x_i, y_i, z_i$</td>
<td>kmol kmol$^{-1}$</td>
</tr>
<tr>
<td>$K_i$</td>
<td>–</td>
</tr>
<tr>
<td>$T, T_f$</td>
<td>K</td>
</tr>
<tr>
<td>$P, P_f$</td>
<td>kPa</td>
</tr>
<tr>
<td>$q$</td>
<td>kJ s$^{-1}$</td>
</tr>
<tr>
<td>$h, H, h_f$</td>
<td>kJ kmol$^{-1}$</td>
</tr>
</tbody>
</table>
variables: \( m, F, L, V, t, x_i, y_i, z_i, K_i, T, T_f, P, P_f, q, h, H, h_f \) \( \Rightarrow \) 13+4C

countants: \( \Rightarrow \) 0

specifications: \( q, t \) \( \Rightarrow \) 2

driving forces: \( F, z_i, T_f, P_f \) \( \Rightarrow \) 3+C

unknown variables: \( m, L, V, x_i, y_i, K_i, T, P, h, H, h_f \) \( \Rightarrow \) 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: \( m(0), x_i(0), T(0) \) \( \Rightarrow \) 2+C

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

Note: file Sample\_flash\_pid.mso has level and pressure controllers.
EQUATIONS

"Overall Molar Balance"
50      diff(ML) = Inlet.F - OutletL.F - OutletV.F;

"Component Molar Balance"
54      diff(W) = Inlet.F*Inlet.z - OutletL.F*OutletL.z - OutletV.F*OutletV.z;

"Molar Holdup"
57      M = ML*OutletL.z;

"Equilibrium"
60      OutletV.z = K*OutletL.z;

"Equilibrium Constant"
63      PP.LiquidFugacityCoefficient(OutletL.T, OutletL.P, OutletL.z) =
64      PP.VapourFugacityCoefficient(OutletV.T, OutletV.P, OutletV.z) * K;

"Mol fraction normalization"
67      sum(OutletL.z) = sum(OutletV.z);

"Energy Balance"
69      diff(E) = Inlet.F*Inlet.h - OutletL.F*OutletL.h - OutletV.F*OutletV.h + InletQ.Q;

"Energy Holdup"
72      E = ML*OutletL.h;

"Thermal Equilibrium"
76      OutletV.T = OutletL.T;

"Mechanical Equilibrium"
79      OutletV.P = OutletL.P;

"Vaporization Fraction"
82      OutletV.F = Inlet.F * vfrac;

"Liquid Volume"
85      vl = PP.LiquidVolume(OutletL.T, OutletL.P, OutletL.z);

switch orientation case "vertical"
90      across = 0.5 * asin(1) * diameter^2;

"Liquid Level"
92      ML * vl = across * level;
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.Inlet.L.F = 400*sqrt(fl.Level/'m') * 'kmol/h';

SPECIFY
s1.Inlet.F = 496.3 * 'kmol/h';
s1.Inlet.T = 338 * 'K';
s1.Inlet.P = 507.1 * 'kPa';
s1.Inlet.z = [0.2379, 0.30820, 0.09958];
fl.Inlet.F = 68.5 * 'kmol/h';
Q.InletQ.F = 0 * 'kJ/h';

SET
fl.V = 50 * 'm3';
fl.diameter = 2 * 'm';
fl.orientation = "vertical";
fl.orientation = "horizontal";

INITIAL
fl.Inlet.T = 338 * 'K';
fl.Level = 0.4 * 'm';
fl.Inlet.Lz[1] = 0.1;
fl.Inlet.Lz[2] = 0.1;
fl.Inlet.Lz[3] = 0.1;
fl.Inlet.Lz[4] = 0.1;
fl.Inlet.Lz[5] = 0.1;
fl.Inlet.Lz[6] = 0.1;
Horizontal axis is always the independent variable (usually time)
Choose the file format

Right-click the mouse button and select “Export Image”
Choose the file format

**RLT:** MATLAB/SCiLAB  
**XML:** EXCEL/OpenOffice
Using EXCEL to analyze the results

Results separated by devices
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
When making a connection, only compatible ports become available to connect.

drag & drop ports to create a connection

click to create a device
EMSO Tutorial
– Building Block Diagrams: set case study –

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

<table>
<thead>
<tr>
<th>EOS Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ideal</td>
<td>Ideal Gas, valid only for vapour models</td>
</tr>
<tr>
<td>IdealLiquid</td>
<td>Ideal Liquid, valid only for liquid models</td>
</tr>
<tr>
<td>RK</td>
<td>Redlich-Kwong</td>
</tr>
<tr>
<td>SRK</td>
<td>Soave-Redlich-Kwong</td>
</tr>
<tr>
<td>PR</td>
<td>Peng-Robinson</td>
</tr>
<tr>
<td>APR</td>
<td>Asymmetric-PR</td>
</tr>
<tr>
<td>ASRK</td>
<td>Asymmetric-RK</td>
</tr>
<tr>
<td>UNIFAC</td>
<td>UNIFAC (Dortmund)</td>
</tr>
</tbody>
</table>

PC-SAFT
Note: LaTeX must be installed.
Dynamic Simulation of a Propane Refrigeration Cycle of a Natural Gas Processing Unit
Dynamic Simulation of a Depropanizer (165 trays, 2 comp.)
Dynamic Simulation of a Deisobutanizer (80 trays, 13 comp.)
Some Industrial Applications

Steady-State Simulation of a Power Plant with Pulverized Coal
Some Industrial Applications

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

Compounds: OD, TOC, Phenol, NH₃, NO₂, NO₃ and 6 groups of bacteria

Fonte: Ricardo Motta
Dynamic Simulation of a General Purpose Polystyrene Process

Some Industrial Applications
Method of Orthogonal Collocation with EMSO

- Boundaries
- Internal Points
- Alfa and Beta

EMSO Simulator

Plugin

Roots
A and B Matrices

- Jacobi roots
- A and B matrices

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

Plugin: ocfem_emso.dll
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:

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

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

Initial conditions:

$$y(0, x) = 0$$
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 and OCM_din.mso.
Comparing Results

OCM by EMSO $\alpha = 1 \quad \beta = 1$
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)
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:
    \[
    \text{CH}_3\text{COCH}_3 \rightarrow \text{CH}_2\text{CO} + \text{CH}_4
    \]
  - The second step is the reaction of ketene with acetic acid.
    \[
    \text{CH}_2\text{CO} + \text{CH}_3\text{COOH} \rightarrow (\text{CH}_3\text{CO})O
    \]

**Problem Definition**

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

![Diagram of a reactor](image)

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

\[ A \rightarrow B + C \]

- The reaction is of 1\textsuperscript{a} order in relation to acetone in the cracking reaction, with Arrhenius constant given by:

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

- \( k \) – seconds\(^{-1}\)
- \( T \) – Kelvin
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 m²;
  - total length of 2.28 m;
- Operating conditions
  - feed temperature 762°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 ($k_0$): $8.2 \times 10^{14}$ s⁻¹
  - activation energy ($E/R$): 34222 K
  - heat of reaction: -80.77 kJ/mol
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.