Introduction to EcosimPro

www.ecosimpro.com

Simulation languages

- Advantages:
  - Provide support in all phases of model development and exploitation
  - Allow the user focusing the attention in the problem and not in the programming
  - Allow saving time
  - Provide confidence in the results obtained
  - Open the field to non-experts in modelling or computing and to the use of models in other fields

Key steps and concepts

- ✓ Process represented by a mathematical model $V - R*I = 0$
- ✓ Specify the aims of the simulation (which variables are known, boundary conditions, and which ones must be computed): Example: I is known, voltage drop V wish to be computed
- ✓ Formulate the mathematical model according to the aims (Assign computational causality, create a partition)
  $V = I*R$
- ✓ Specify an experiment (Give values to the parameters and boundary conditions) $R = 10, I = 2$
- ✓ Solve the equations and display the results $V = 10*2 = 20$

Modelling languages

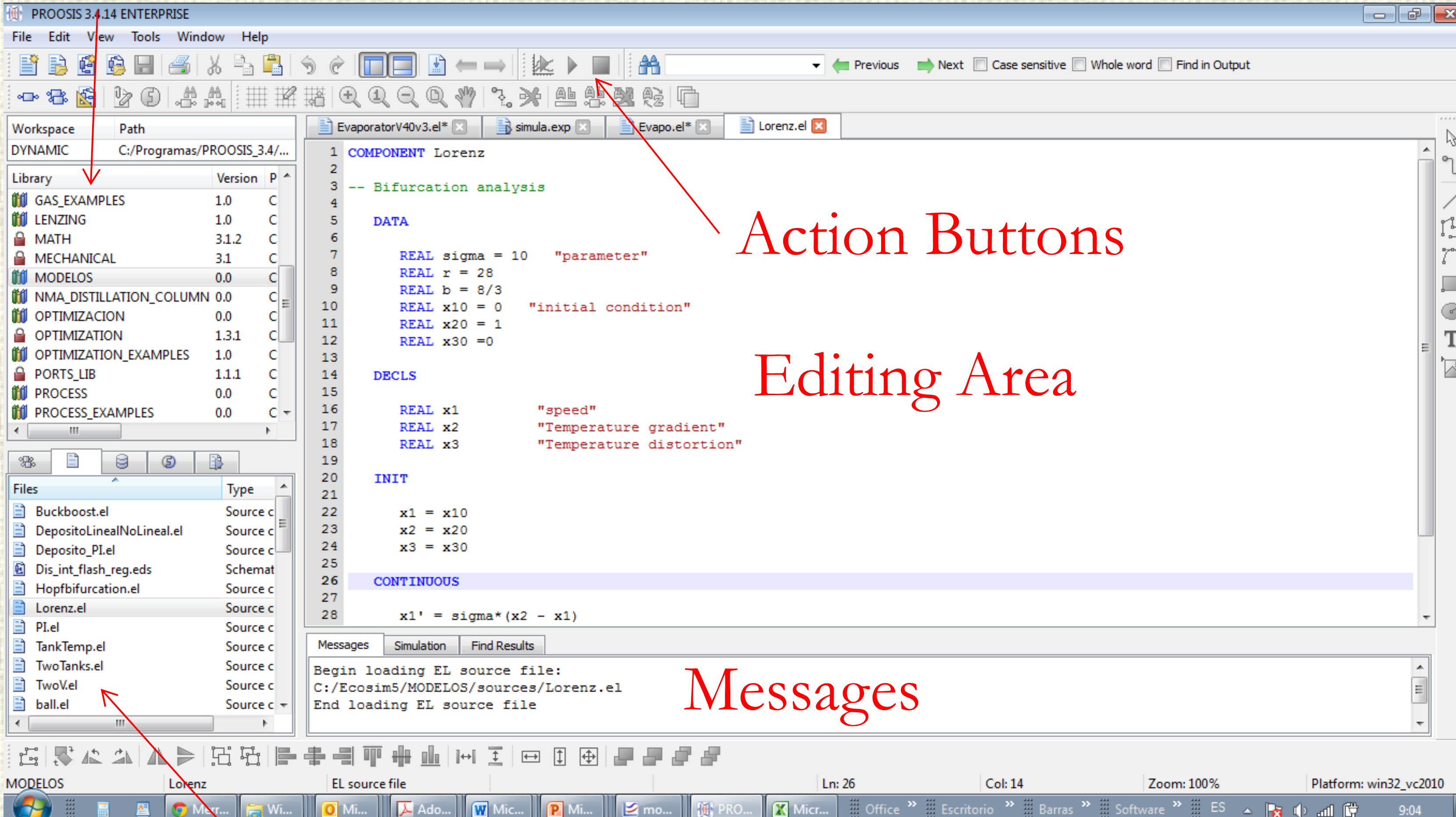
- Software tools that facilitate:
  - The description of a process model and the assignment of computational causality
  - The description of the experiments to be performed
  - Solving the equations
  - Displaying results
  - Provide other functionalities (optimization, parameter estimation, validation,…)

EcosimPro

- First version 1992, Unix, ESA
- First version under Windows: 1999
- Object oriented tool
- Support continuous, discrete and discrete event processes
- Models are built by textual description of from graphical libraries.
- Provides a software development environment
- Open code, C++, ActiveX, OPC,…
- Version 5 , 2013, multiplatform QT
- Proosis

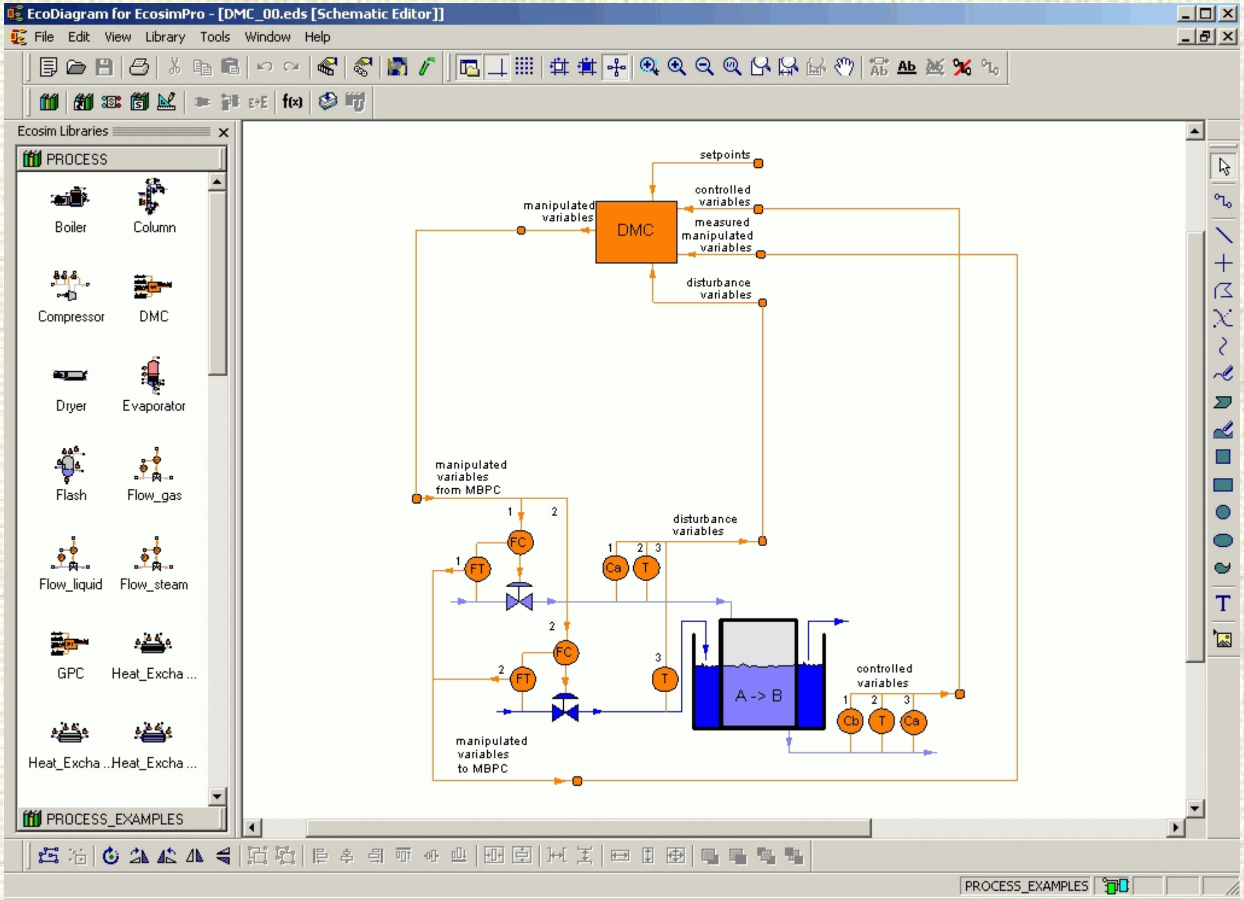
EcosimPro environment

Libraries /Workspaces



Models

Graphical environment



Basic elements

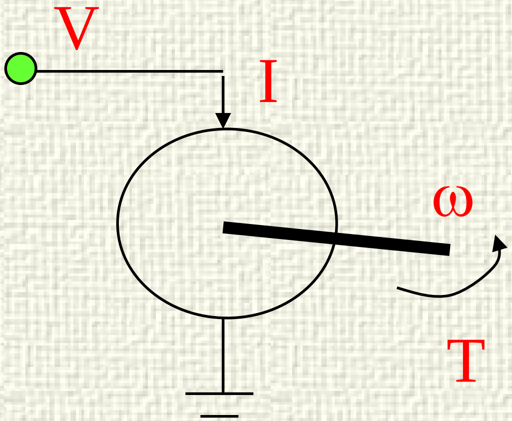
- ✓ COMPONENT: Represents a model. Includes data, variables, equations, events, topology,…
- ✓ PORT Defines the link of a component with the outside world. It plays the role of electrical connections, pipes, etc. that appear in the real world connecting elements.
- ✓ EXPERIMENT: Defines how to perform a simulation, giving values to data, boundary conditions, etc.
- ✓ LIBRARY: Set of files with ports, components, functions, etc. that belong to a certain field (e.g. CONTROL, ELECTRICAL, THERMAL, etc.) and can be used to define other components.

EcosimPro Environment

- ✓ Creating a Workspace / library
- ✓ Models described in Components
- ✓ Components can be linked by ports
- ✓ Editing a component. Example: a D.C. motor

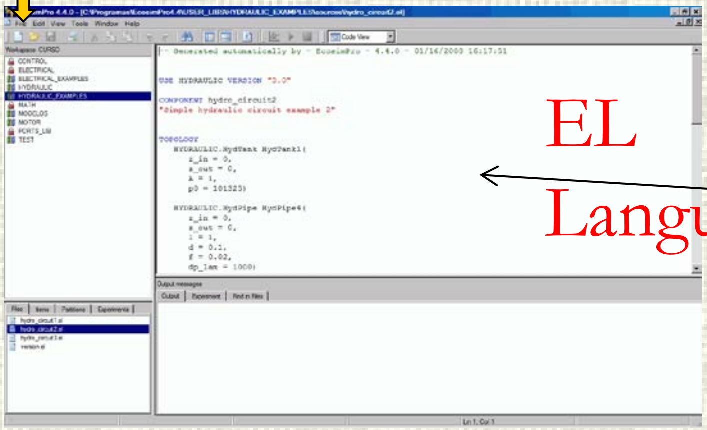
$$J\frac{d\omega}{dt} = ki - f\omega - T$$

$$V = Ri + k_e\omega$$



Creating a component in a Library

New



EL Language

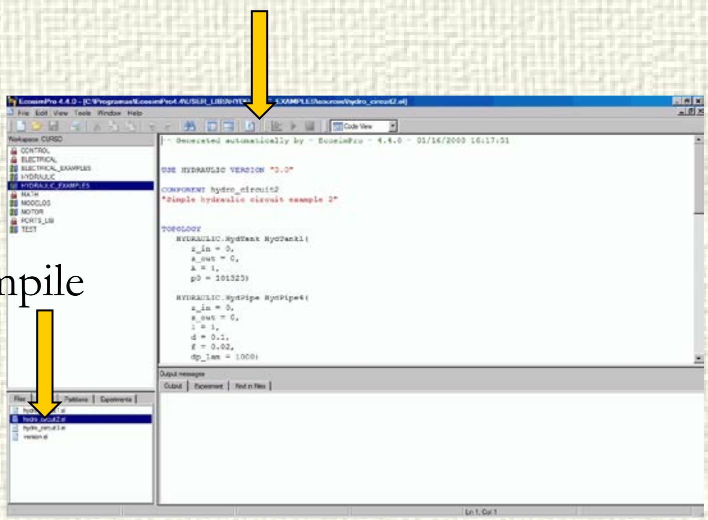
Declarative equations. They will be manipulated symbolically according to the aims and boundary conditions of the simulation

```
COMPONENT motorDC
DATA
     REAL J = 2          "Momentum of inertia"
     REAL K = 3          "torque constant"
     REAL f = 0.01       "friction coefficient"
     REAL R = 0.1        "electrical resistance"
     REAL Ke = 0.5
DECLS
    REAL T         "Torque"
    REAL  w              "speed"
   REAL  V        "voltage"
   REAL  i        "current"
```

Compiling

Analysing the correctness of the model from the point of view of the EL language

Compile



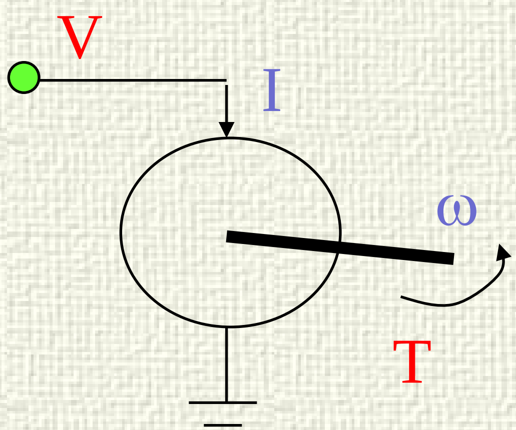
```
COMPONENT motorDC
DATA
      REAL J = 2          “Momentum of
inertia"
      REAL K = 3          “torque constant"
      REAL f = 0.01       "friction
coefficient
      REAL R = 0.1        “electrical
resistance"
      REAL Ke = 0.5
DECLS
     REAL T                  “Torque”
     REAL  w                “speed”
    REAL  v                  “voltage”
    REAL  i                   “current”
CONTINUOUS
        J * w‘ = K * i – f * w - T
      v = R * i + Ke * w
END COMPONENT
```

Partitions

- A partition is a math model associated to a process ready to define experiments on it.
- When there are more variables than equations the user should define the boundary conditions and, sometimes, solve problems related with high index and algebraic loops

$$J\frac{d\omega}{dt} = k_1 i - f\omega - T$$

$$V - Ri + k_2\omega = 0$$

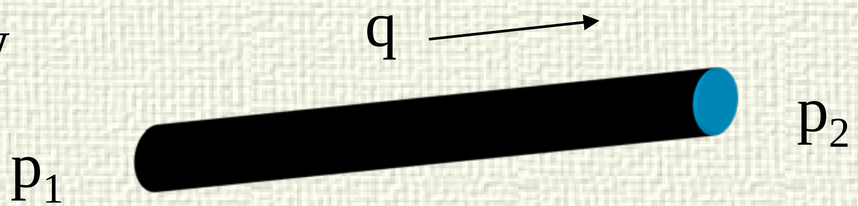


Boundary conditions, e.g.: Applied voltage V and external torque T

Why partitions?

The mathematical formulation of the equations depends on the context

Same physical element and law



If p_1 and p_2 are given:

$$q = k\sqrt{p_1 - p_2}$$

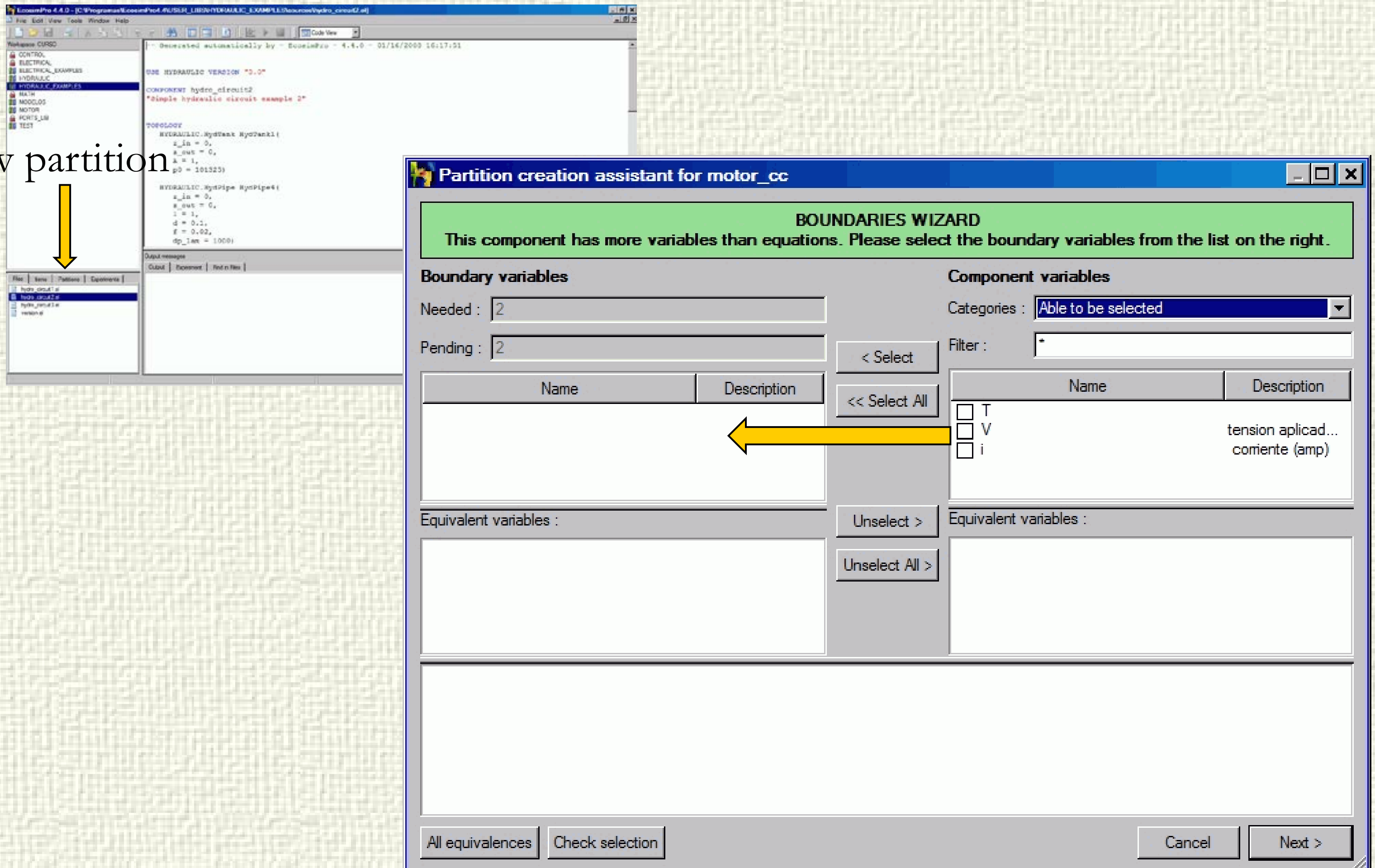
If p_1 and q are given:

$$p_2 = p_1 - \frac{q^2}{k}$$

Aim: Making the model of a process independent of its use in a particular situation

Creating a partition

New partition



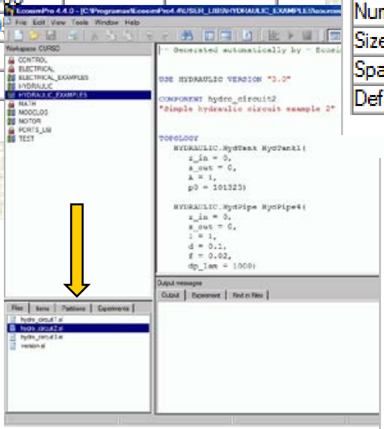
Viewing a partition

VARIABLES:

NUM	NAME	UNITS	EQUIV-TO	TYPE	MATH-TYPE	INITIAL	LRANGE	RRANGE
1	J			REAL	DATA_VAR	0.001		
2	L	H		REAL	DATA_VAR	0.01		
3	R	ohmios		REAL	DATA_VAR	0.2		
4	T			REAL	BOUNDARY			
5	V	volts		REAL	BOUNDARY			
6	f			REAL	DATA_VAR	0.004		
7	i	amp		REAL	EXPLICIT			
8	k1			REAL	DATA_VAR	0.006		
9	k2			REAL	DATA_VAR	0.0		
10	omega	rad/min		REAL	DYNAMIC			
11	omega'			REAL	DERIVATIVE			

GENERAL STATISTICS

INFO	#
Number of equations:	2
Number of boxes (coupled subsystems of equations):	0
Number of linear boxes:	0
Number of nonlinear boxes:	0
Number of EXPLICIT variables:	1
Number of DERIVATIVE variables:	1
Number of ALGEBRAIC variables:	0
EXPLICIT + DERIVATIVE + ALGEBRAIC variables:	2
Number of BOUNDARY variables:	2
Size of Jacobian matrix (DYNAMIC+ALGEBRAIC):	1x1
Sparsity factor in Jacobian matrix (% of zeros):	0
Default integration method:	DASSL



TYPE OF VARIABLES

TYPE	VARIABLE	DATA	CONSTANT
REAL	5	6	0
INTEGER	0	0	0
STRING	0	0	0
TABLE	0	0	0

GLOBAL FLAGS:

FLAG	VALUE
Remove derivatives	FALSE

BOUNDARIES:

NAME	UNITS	DESCRIPTION	INITIAL
T			
V	volts	tension aplicada al inducido (volts)	

JACOBIAN INDEPENDENT VARIABLES:

POS	VARIABLE	CATEGORY	UNITS	DESCRIPTION	INITIAL	CLOSURE EQUATION
1	omega	DYNAMIC	rad/min	velocidad angular (rad/min)		omega' = 0

Note 3: In equations 'E' means explicit,'I' implicit,'L' linear,,'F' function

SORTED EQUATIONS:

###eqts
[2] i = (V - k2 * omega) / R {E@i@@}
[1] omega' = (k1 * i - f * omega - T) / J {E@omega'@@}

End of document: TEST.motor_cc.+t+v
Terminology for Equations/Variable matrix:
- X: Variable used in equation
- E: Explict variable
- A: Algebraic variable
- L: Variable solved linearly
- O: Calculated as output of a function or SEQUENTIAL block

NOTE: Some internal equations are not presented (typically with variables ended in ".")

EQUATIONS/VARIABLES MATRIX:

e/v	v7	v11
e2	E	
e1	X	E

Types of variables of a partition

Explicit

Boundaries

$$i = (V + k_2\omega) / R$$

$$\omega' = \frac{d\omega}{dt} = (k_1 i - f\omega - T) / J$$

Dynamic

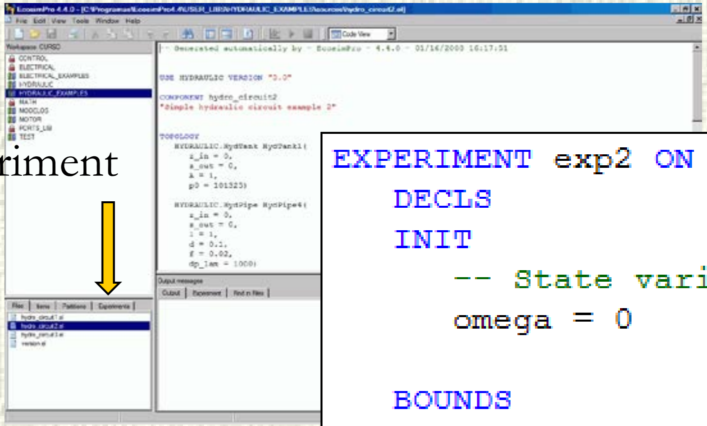
Derivative

VARIABLES:

NUM	NAME	UNITS	EQUIV-TO	TYPE	MATH-TYPE	INITIAL	LRANGE	RRANGE
1	J			REAL	DATA_VAR	0.001		
2	L	H		REAL	DATA_VAR	0.01		
3	R	ohmios		REAL	DATA_VAR	0.2		
4	T			REAL	BOUNDARY			
5	V	volts		REAL	BOUNDARY			
6	f			REAL	DATA_VAR	0.004		
7	i	amp		REAL	EXPLICIT			
8	k1			REAL	DATA_VAR	0.006		
9	k2			REAL	DATA_VAR	0.055		
10	omega	rad/min		REAL	DYNAMIC			
11	omega'			REAL	DERIVATIVE			

Creating an experiment

New experiment

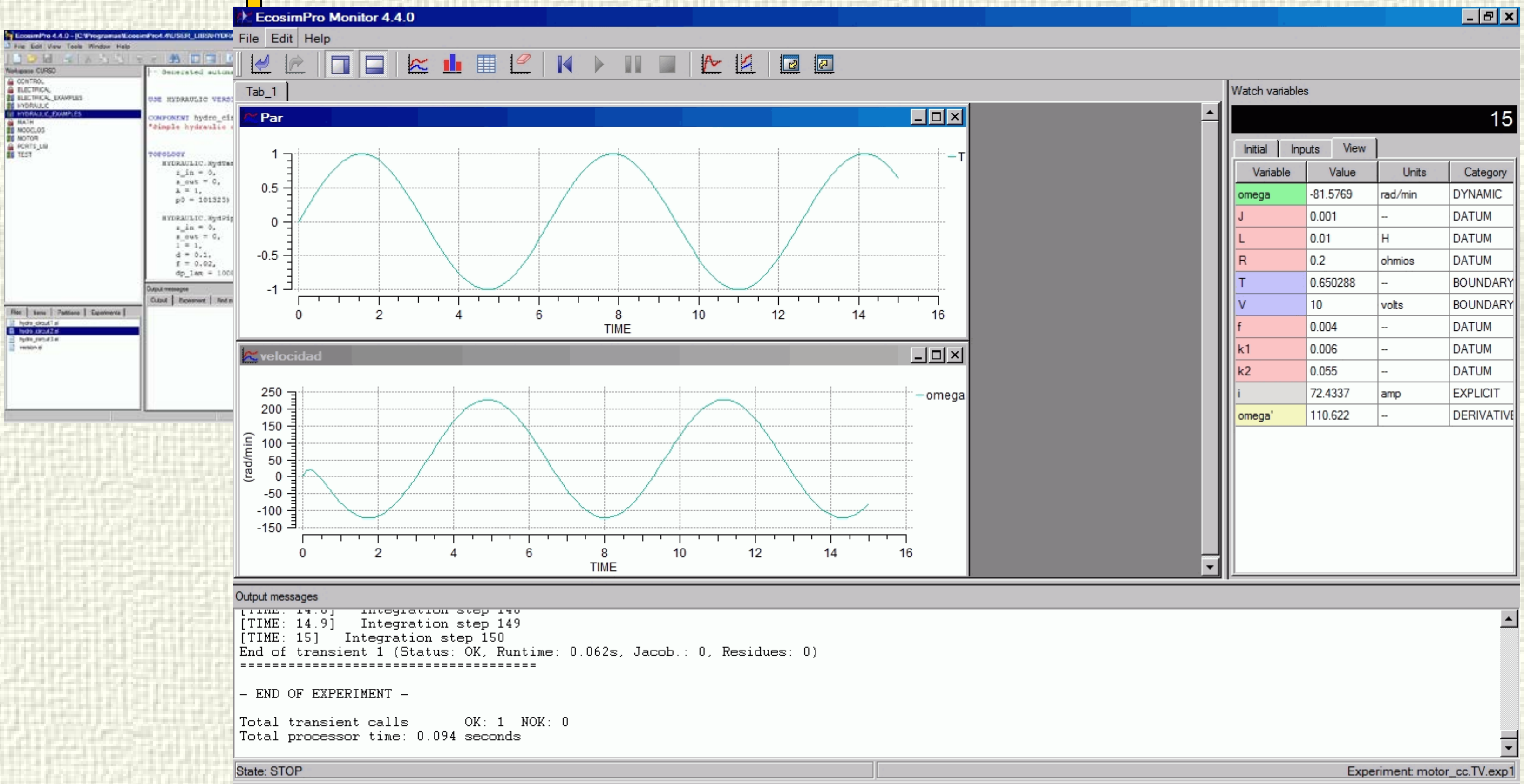


```
EXPERIMENT exp2 ON motor_cc.TV
   DECLS
   INIT
      -- State variables
      omega = 0

   BOUNDS
      -- Set expressions for boundary variables: v = f(t;...)
      T = 0
      V = 0

   BODY
      REPORT_TABLE("reportAll", "*")

      TIME = 0

      TSTOP = 15
      CINT = 0.1
      INTEG()

END EXPERIMENT
```

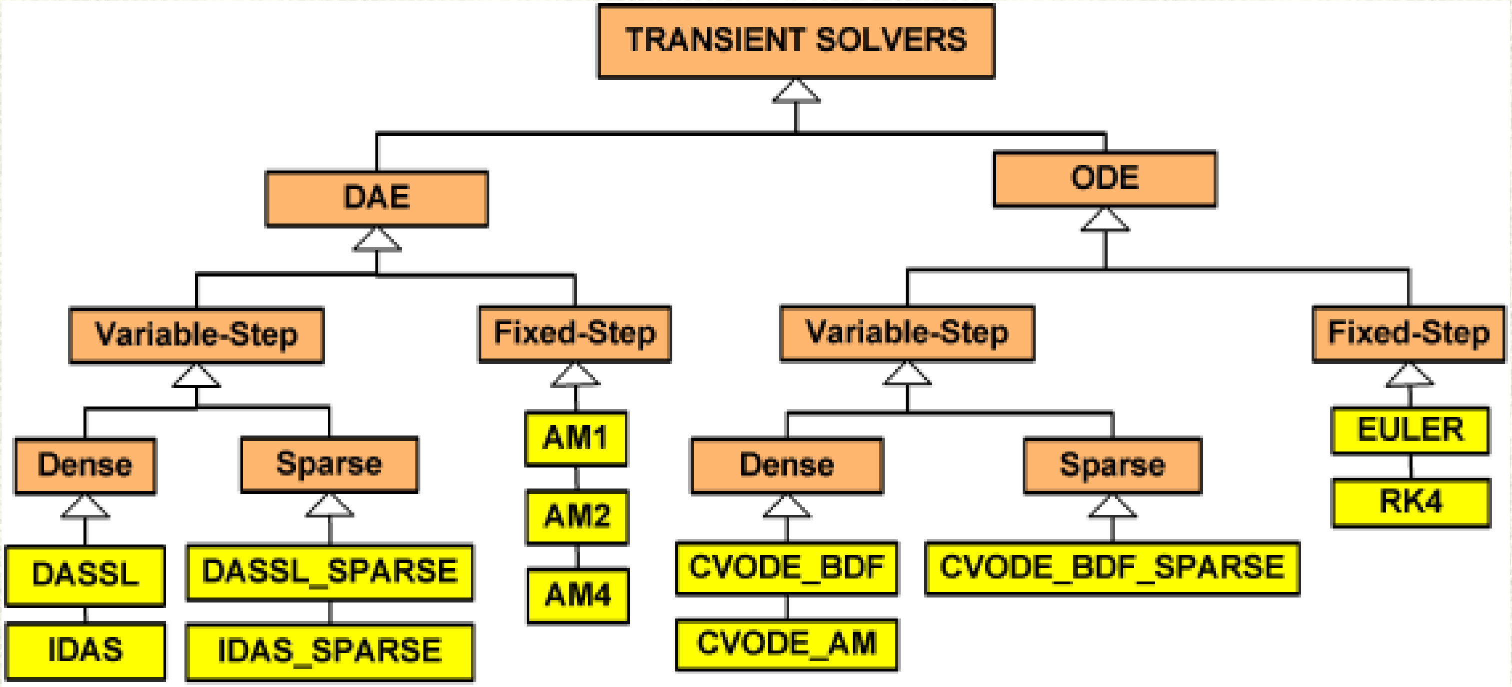
Executing an experiment

Simulate in monitor

Graphical environment: Monitor



Integration methods



DAE systems

- Many problems are formulated in terms of coupled differential and algebraic equations (DAE)

$$\frac{dx}{dt} = f(x, y, u)$$

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

Or with implicit equations where it is not possible to solve dx/dt in terms of the remaining variables

$$F\left(\frac{dx}{dt}, x, u, t\right) = 0$$

Integration: DASSL, IDAS

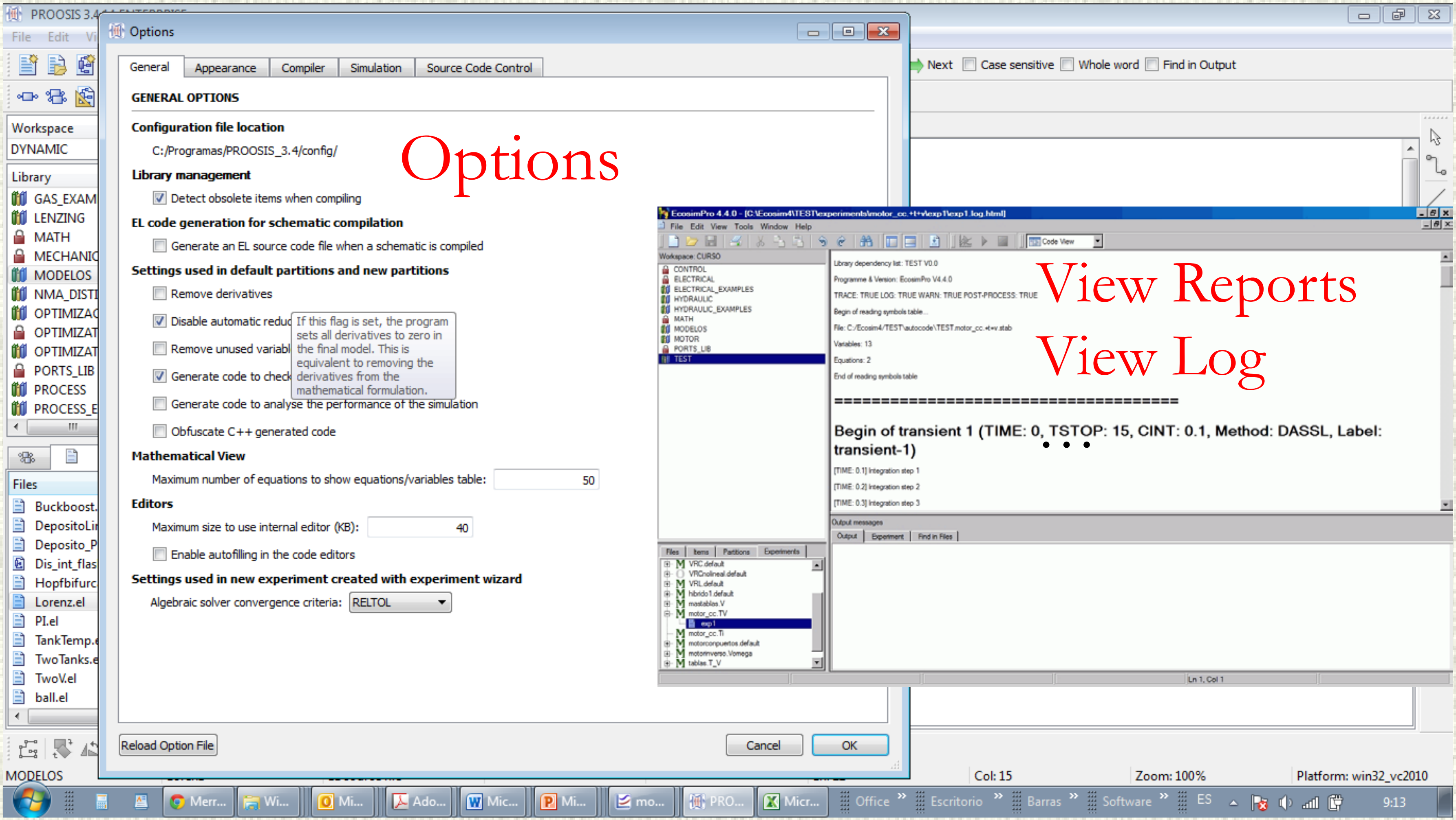
$$F(\frac{dx}{dt}, x, t) = 0$$

Implicit DAE equations can be solved approximating the derivatives by BDF formulas of variable order and solving the resulting non-linear implicit equation in x(t+h) with the Newton-Raphson method. The procedure is initialized by means of extrapolation.

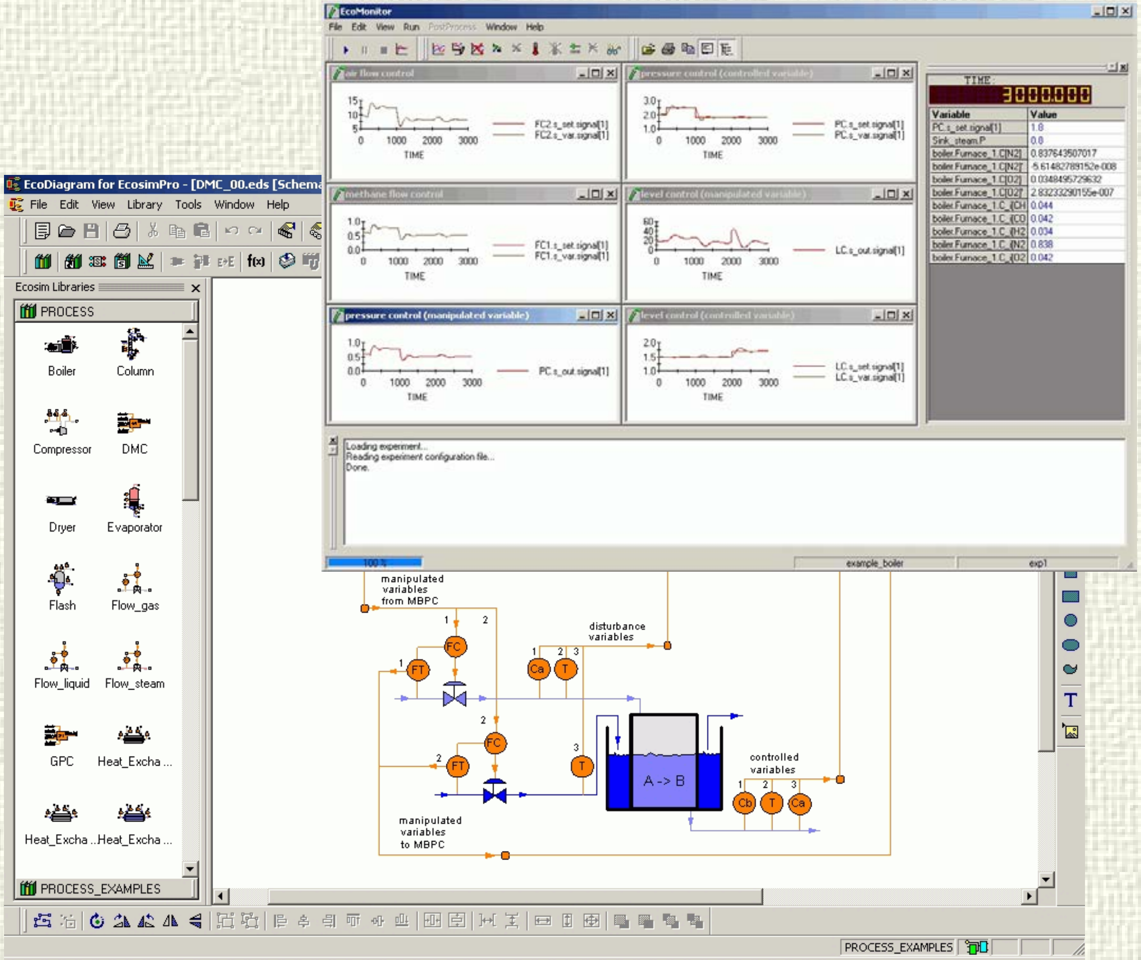
$$F(\frac{x(t+h) - old(x(t))}{h}, x(t+h), t+h) = 0$$

Variable order approximation of dx/dt (BDF 1 to 5) and variable step-size h in order to bound the integration error.

EcosimPro


Options
View Reports
View Log
Options
General
Appearance
Compiler
Simulation
Source Code Control
GENERAL OPTIONS
Configuration file location
C:/Programas/PROOSIS_3.4/config/
Library management
Detect obsolete items when compiling
EL code generation for schematic compilation
Generate an EL source code file when a schematic is compiled
Settings used in default partitions and new partitions
Remove derivatives
Disable automatic reduc
If this flag is set, the program sets all derivatives to zero in the final model. This is equivalent to removing the derivatives from the mathematical formulation.
Remove unused variabl
Generate code to check
Generate code to analyse the performance of the simulation
Obfuscate C++ generated code
Mathematical View
Maximum number of equations to show equations/variables table:
50
Editors
Maximum size to use internal editor (KB):
40
Enable autofilling in the code editors
Settings used in new experiment created with experiment wizard
Algebraic solver convergence criteria:
RELTOL
Reload Option File
Cancel
OK
Begin of transient 1 (TIME: 0, TSTOP: 15, CINT: 0.1, Method: DASSL, Label: transient-1)
Next
Case sensitive
Whole word
Find in Output

Steps



- ✓Write the model and check correctness (compile)
- ✓Define Partition
- ✓Define experiment
- ✓Generate source code (C++)
- ✓Compile and link
- ✓Execute the experiment in a graphical environment

EL Introduction

```
COMPONENT Cntrl_on_off IS_A Controller
DATA
   REAL   e_off = -1.   "Error for switching to OFF state"
   REAL   e_on = 1.     "Error for switching to ON state"
   REAL   u_off  = 0.   "Value of controller output when OFF"
   REAL   u_on   = 1.   "Value of controller output when ON"

DECLS
             ENUM state_type = {OFF, ON}
             ENUM state_type state      "Current state"

DISCRETE
      WHEN (e > e_on) THEN
            state = ON
      END WHEN

      WHEN (e < e_off) THEN
           state = OFF
      END WHEN
 CONTINUOUS
      u = ZONE (state == ON) u_on
                 OTHERS u_off
END COMPONENT
```

Parent Component

Data

Local declarations

Discrete events

Continuous equations

Component

```
Component_def::= ABSTRACT? COMPONENT ID
          (IS_A ID (,ID)* )?
          ('(' parameter_s ')')?
          ( PORTS port_decl_s )?
          ( DATA var_decl_s ) ?
          ( DECLS comp_decl_s )?
          ( TOPOLOGY topology_stm_s )?
          ( INIT seq_stm_s )?
          ( DISCRETE discrete_stm_s )?
          ( CONTINUOUS labelled_stm_s )?
       END COMPONENT
```

Data Types

- Basic: REAL, INTEGER, BOOLEAN, STRING

```
REAL x, y
STRING str = "hello world"
BOOLEAN isConnected = FALSE
```

- Enumerative types:

```
ENUM chemicals = {N2, H2O, CO2, N2, O2, H2SO4 }
SET_OF(chemicals) air = {N2, O2, H2O, CO2}
SET_OF(chemicals) water = {H2O}
```

Arrays:

```
REAL v[3]
REAL w[3,6,2]
ENUM chemicals mix[2]= { H20, O2 }
STRING colors[3]= {"red","white","blue"}
```

Data Types

Constants: The user can declare a variable as constant, nobody can modify it afterwards.

```
CONST REAL PI= 3.141592
```

Different scopes in EL:

```
LIBRARY DEFAULT_LIB
REAL i= 9                -- Global variable
COMPONENT test
DECLS
  REAL v[4],y, i         -- Local scope
INIT
  i= DEFAULT_LIB.i + 4
  y= SUM(i IN 1,4; v[i]) -- expr. scope
```

Data Types: Tables

```
EXPERIMENT Tinterpol ON tablas.T_V
  DECLS
          TABLE_1D tabT= { {0.,  1,    2,    3,    4,  5,   6,  7,   8, 9},    -- time values
                           { 0.3, 0.6, 0.7, 0.75, 1, 1.1, 1, 1.2, 1, 0.8 } }    -- output
  INIT
    -- State variables
      omega = 0
      i = 0
  BOUNDS
    -- Set expressions for boundary variables: v = f(t;...)
    -- timeTableInterp use TIME as the input parameter in the table
    -- and avoid discontinuity problems between two intervals
    --  Constant after the last value

    T = timeTableInterp(TIME, tabT)
    V = 250
  BODY
    ……..
```

Tables

```
COMPONENT mastablas
     DATA
          ….
          TABLE_1D tabT= { {0., 1,  2,  3,   4, 5,   6, 7,  8, 9},     -- time values
                           {0.3,0.6,0.7,0.75,1, 1.1, 1, 1.2,1, 0.8 } }     -- output
     DECLS
          …….
          REAL Tfile
          INTEGER last = 0  -- variable auxiliar para mejorar la velocidad
          TABLE_1D tabF
     INIT
          readTableCols1D(expandFilePath("@TEST@/docs/mytable.txt"), 2, 3, tabF)
     CONTINUOUS
          ……
          Tspline = splineInterp1D(tabT, TIME)
          Tinterplast = linearInterpHist1D(tabT, TIME, last) -- no queda cte tras ultim
          Tinterp = linearInterp1D(tabT, TIME)    -- no queda cte tras ultimo valor
          T = timeTableInterp(TIME, tabT)    -- si queda cte tras el ultimo valor
          Tfile = timeTableInterp(TIME, tabF)
END COMPONENT
```

Expressions

Arithmetic: a * 2 + (c - u) / (x**2)

SUM

x= SUM(i IN 1,3; inertia[i])

is equivalent to x= inertia[1]+inertia[2]+inertia[3]

Relational: 2 > (x - y)

Logical: (x > 9.8 AND n != 7 OR m == 6)

TIME contains the current integration time

TSTOP contains the current final integration time

x= sin(TIME)

WHEN(TIME >= (TSTOP / 2))

Types of statements supported

- ✓ EcosimPro provides three different paradigms:

- Sequential statements like IF, WHILE, FOR, etc. The order of the statements is fundamental. Supported in Fortran, Java, C++
- Continuous statements like Differential-Algebraic equations. The order is indifferent. Used to express the dynamic behaviour of the dynamic model.
- Discrete statements like WHEN. The order is indifferent. Used to express the discrete behaviour of the dynamic model.

Sequential statements

They are executed in the order the user write them. Can be used in any sequential part:

Assignments: `x= 8`

Function calls: `x= add(2,2)`

IF-THEN-ELSE:

```
IF ( x > 8.3 ) THEN
  y= sqrt(x)
ELSE
  y= x
ENDIF
```

```
WHILE speed < maxSpeed
  speed += 0.1
END WHILE
```

```
FOR (i IN 0,4)
  v[i]= 0
END FOR
```

EXPAND / EXPAND_BLOCK

EXPAND: Insertion of multiple equations in one go

```
EXPAND( i IN 1,2) out_entropy[i]= in_entropy[i]
```

equivalent to: (don't confuse with FOR statement!)

```
out_entropy[1]= in_entropy[1]
out_entropy[2]= in_entropy[2]
```

(Note: Each equation in totally independent)

```
EXPAND_BLOCK   (i IN 1, n)
    mg[i] = P[i]*PM_g[i]*Vf_g/cte_R/(Tg[i]+273.15)
    P[i]= mg[i]*cte_R*(Tg[i]+273.15)/(PM_g[i]*Vf_g)
END EXPAND_BLOCK
```

Functions

The user can define its own functions in EL and then call them from any component or port.

```
FUNCTION REAL square(REAL x)
BODY
   RETURN x * x
END FUNCTION
...
x= square(y)
```

SUM

it generates a summation of elements in a given range. For example

v = SUM (j iN 2,5; x[i] * alpha[2*i])

generates the following equation:

v = x[2]*alpha[4] + x[3]*alpha[6] + x[4]*alpha[8]

INIT / DISCR

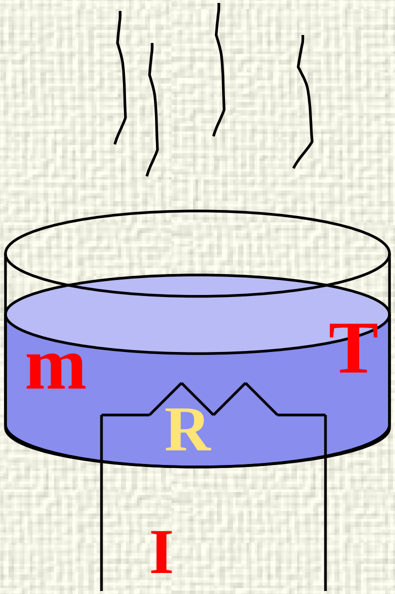
```
COMPONENT reactorAB
DATA
        REAL L = 3.03 "altura del reactor (m)"
        REAL D = 3.03 "Diametro (m)"
        REAL T0 = 65 "Valor inicial de T (ºC)“
        …..
DECLS
        REAL T "Temperatura (ºC)"
        DISCR REAL A "Superficie de transmisión de calor del encamisado (m2)"
        DISCR REAL V "Volumen del reactor (m3)"
        ……
INIT
        V = PI*D*D*L /4   -- calculo del volumen del reactor
        A = PI*D*L        -- calculo de la superficie
        T = T0
        Tr = 51.5
        …..
```

Events and Discontinuities

- In many processes, sharp changes take place at certain time instants, which modify the continuity of f(x,u) or its derivative.
- Such events change the model, so that f(x,u) is transformed at this time instant from $f_1(x,u)$ to $f_2(x,u)$
- Variable structure models, hybrid models,….
- Under this circumstances, direct application of the previous integration methods can lead to wrong results.

Events and Discontinuities: Examples

- Heating and boiling at constant pressure:

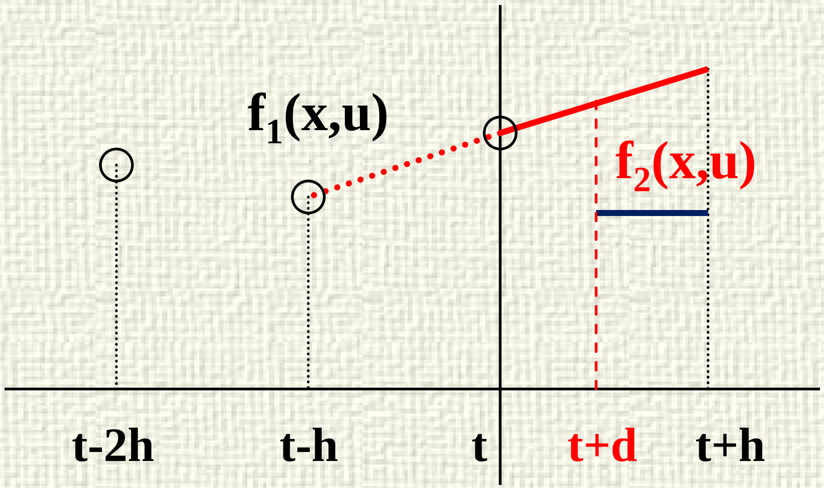


$$\frac{dT}{dt} = \begin{cases} I^2R/(mc_e) & \text{if } T < T_e \\ 0 & \text{if } T \geq T_e \end{cases}$$

$$\frac{dm}{dt} = \begin{cases} 0 & \text{if } T < T_e \\ -I^2R/\lambda & \text{if } T \geq T_e \end{cases}$$

T_e Boiling temperature

Events and Discontinuities



$$x(t+h) = x(t) + \int_{t}^{t+h} f(x(\tau),u)\,d\tau$$

$$\frac{dx}{dt} = f_1(x,u) \qquad \text{time} < t+d$$

$$\frac{dx}{dt} = f_2(x,u) \qquad \text{time} \geq t+d$$

Discontinuities in ECOSIMPRO

Language declarations that control explicitly the location of discontinuities, the model changes and the new initial conditions

- Discrete events

  WHEN (condition)

  equations

  END WHEN

- Changes in the continuous model structure

  x = ZONE (condition 1) equation 1

  ZONE (condition 2) equation 2

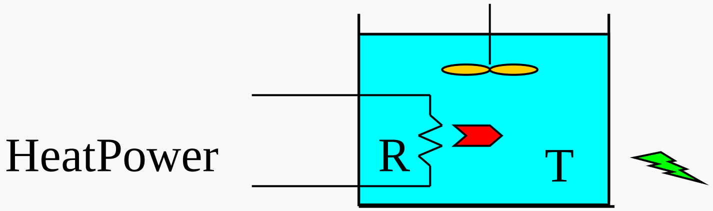
  OTHERS equation 3

  END

- AFTER - Delayed Assignation

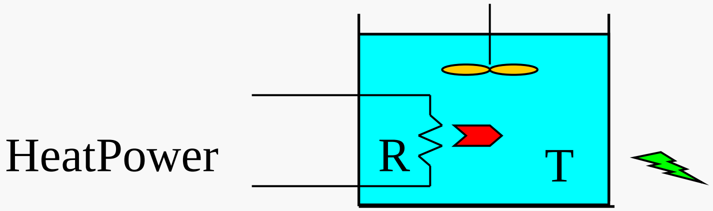
WHEN

```
COMPONENT WhenExample

   DATA
      REAL Tmin = 20
      REAL Tmax = 50.
   DECLS
      REAL HeaterPower
      REAL T = 10.
   DISCRETE
      WHEN (T < Tmin) THEN
         HeaterPower = 50.
      END WHEN
      WHEN (T > Tmax) THEN
         HeaterPower = 0.
      END WHEN
   CONTINUOUS
      T' = 0.1 * (HeaterPower - 10)

END COMPONENT
```



AFTER

```
COMPONENT WhenExample

     DATA
         REAL Tmin = 20
         REAL Tmax = 50
     DECLS
         REAL HeaterPower
         REAL T = 10.
     DISCRETE
         WHEN (T < Tmin) THEN
             HeaterPower = 50. AFTER 5
         END WHEN
         WHEN (T > Tmax) THEN
             HeaterPower = 0.  AFTER 2
         END WHEN

      CONTINUOUS
          T' = 0.1 * (HeaterPower - 10)
END COMPONENT
```



ZONE

```
--Limitation of  a variable

COMPONENT Limits_0

    DECLS
        REAL x
        REAL xmax
        REAL xmin
        REAL y

    CONTINUOUS
        xmax = 0.5 + 0.2 * sin(TIME)
        xmin = -0.5 - 0.2 * sin(2 * TIME)
        x = sin(3*TIME)

        y  = ZONE (x > xmax ) xmax
             ZONE (x < xmin ) xmin
             OTHERS    x
END COMPONENT
```

Construction Parameters
IF INSERT

```
COMPONENT tinsert (INTEGER sw = 1)
  DECLS
      REAL x
      REAL y
  CONTINUOUS
   IF  ( sw == 1 )  INSERT
       3*x - 6*y = 9
       4*x - 4*y = 9
   ELSE
       5*x + 7.6*y = 9.5
       4.34*x - 64*y = 86.4
   END IF
 END COMPONENT
```

Loop Tearing

- ✓ Direct solution of an algebraic loop using Newton-Raphson method leads to an algorithm with a size of the Jacobian as large as the number of variables involved in the loop.
- ✓ The use of Equation Tearing techniques allows sustantial reductions of the size of the Jacobian

Some tearing variables are selected, so that, if given an initial value, it is possible to compute explicitly the remaining variables of the loop. As the initial value may be wrong, there will be as many equations of the loop as tearing variables that will not compute equal to zero (residual equations). The Newton- Raphson algorithm will iterate modifying the tearing variables until the residual equations are satisfied, but with a reduced Jacobian size.

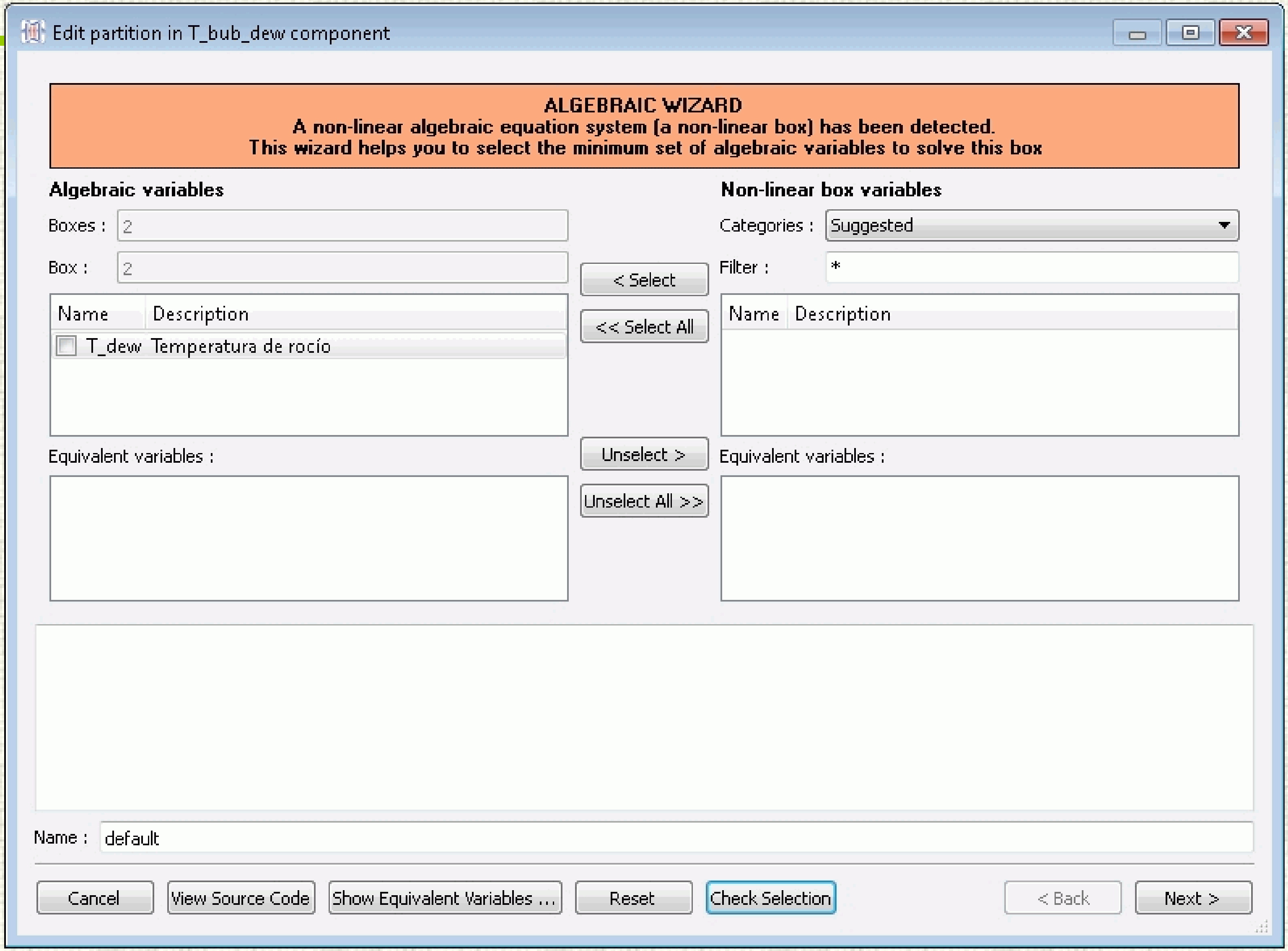
$F_1(x_1, x_2) = 0$
$F_2(x_1, x_2, x_3) = 0$
$F_3(x_1, x_2, x_3) = 0$

x_2 selected as tearing variable

⇩

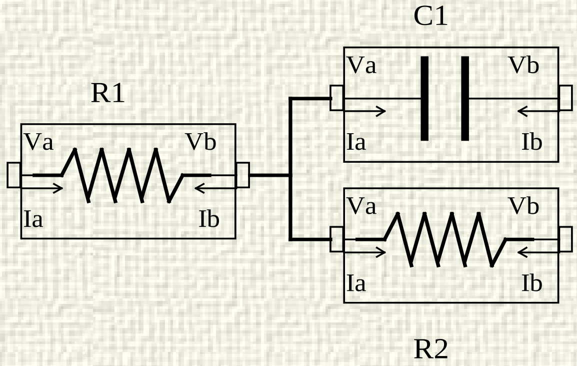
$x_1 = f_1(x_2)$
$x_3 = f_2(x_1, x_2)$
$F_3(x_1, x_2, x_3) = \text{residual}$

Algebraic Loops



Building models

- A model can be composed linking predefined and tested modules
- Each module contains the mathematical model of a particular sub-system
- Each module is connected to the others through an interface or port



- BUT the model equations are generated later on for the whole system taking into account the boundary conditions and associated constraints. High level description.

Model libraries

Facilitate the re-use of models

There are based in the following principles:

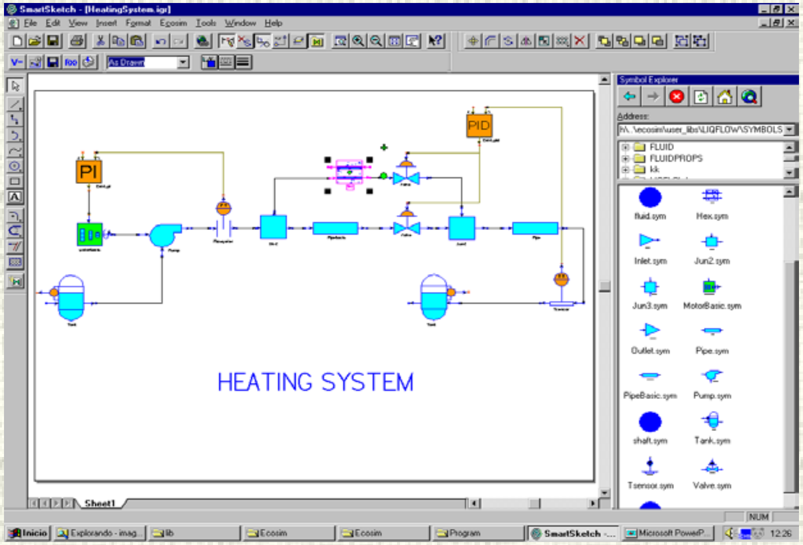
✓Modularity: Independent description of each module

✓Abstraction: Every module can be used through its interface with no need to know details of its internal structure

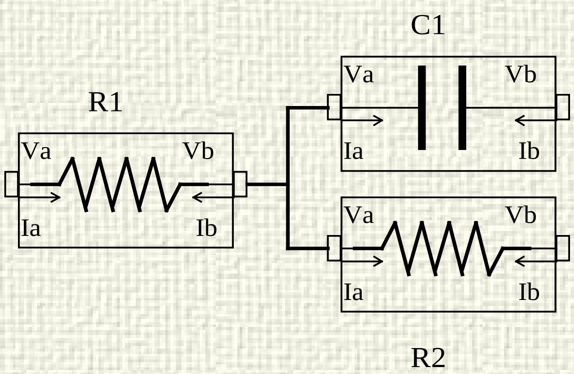
✓Hierarchy

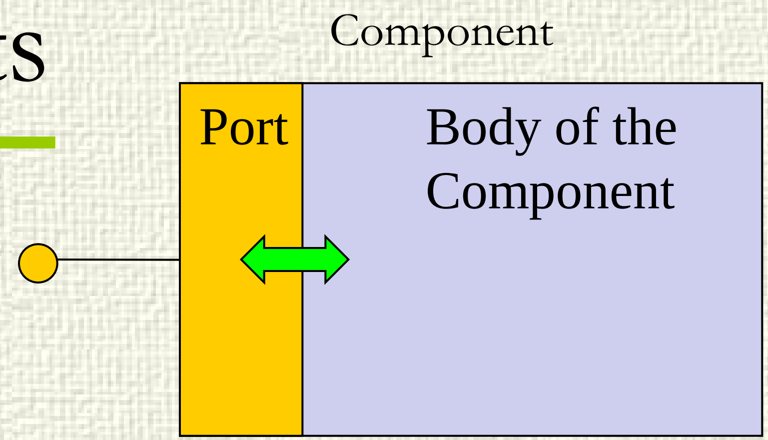
✓Genericity

EcosimPro:
Object oriented
Modelling



Ports





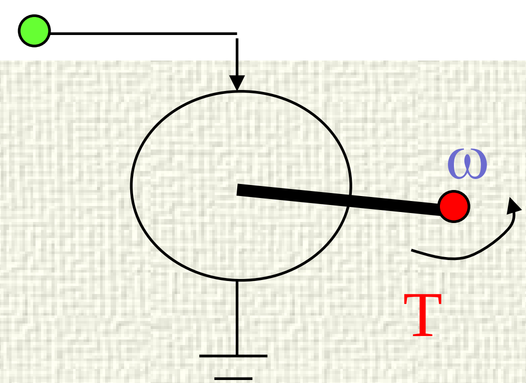
Electrical Port Name

```
PORT Elec
        SUM REAL     c   "current (Amperes)"
        EQUAL REAL v  "voltage (Volts)"
END PORT
```

Current and voltage variables

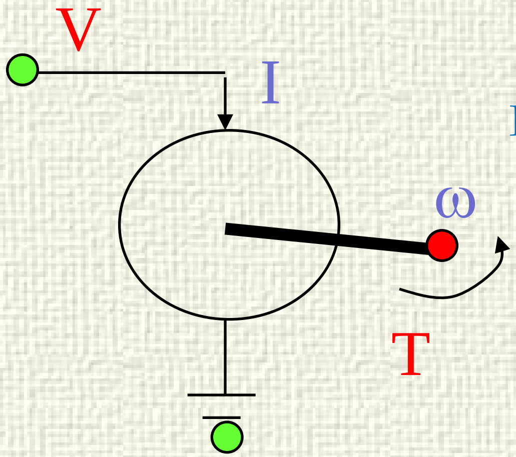
Ports

```
PORT mech_rot  "1D rotational flange“

        SUM REAL      T        UNITS u_Nm  "Torque "
        EQUAL REAL omega  UNITS u_rad_s "Absolute angular velocity"
        REAL               n        UNITS u_rpm "Angular velocity"

    CONTINUOUS
        omega = n * (2*MATH.PI/60)
END PORT
```



DC Motor with Ports

```
USE MATH
USE PORTS_LIB

COMPONENT motorconpuertos

PORTS
      IN elec           feed
      IN elec           ground
      OUT mech_ rot     eje

DATA
  …..
DECLS
 …..
```




```
…
CONTINUOUS

 J*w'= K*i - f*w - T
 V = R*i + Ke*w
 feed.i = ground.i
 feed.i = i
 V = feed.v - ground.v
 eje.T = T
 eje.omega = w

END COMPONENT
```

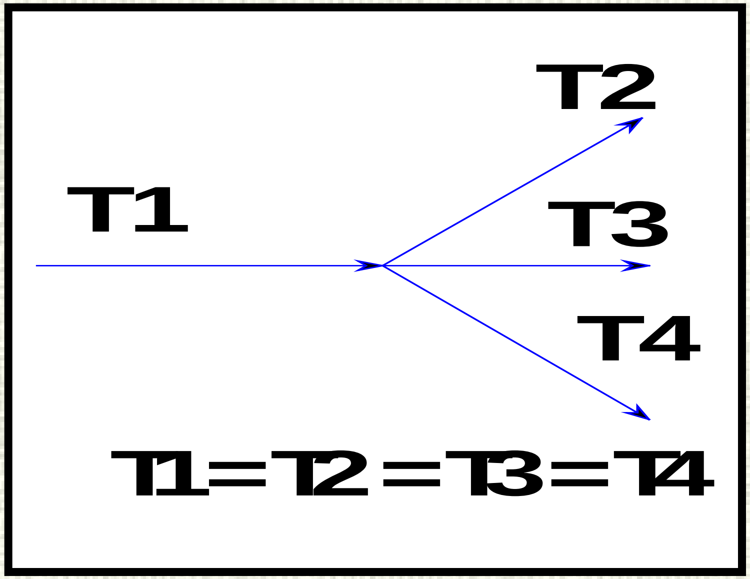
Ports

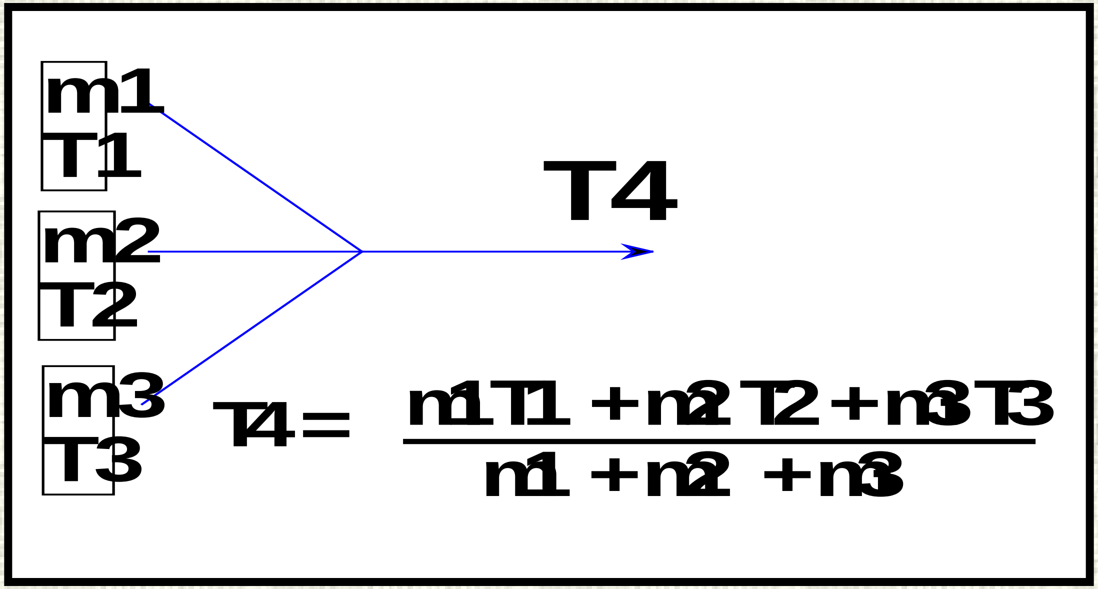
```
PORT Gas
   SUM   REAL          W    RANGE 0, Inf  “Mass Flow (Kg/s)"
   EQUAL REAL          P    RANGE 0, Inf  "Pressure  (Pa)"
   EQUAL OUT   REAL    H = 700000         "Enthalpy  (J/Kg)"
   EQUAL OUT   REAL    FAR                "Fuel Air Ratio"
   SUM         REAL    WF                 "Fuel Flow (Kg/s)"
   SUM   IN    REAL    WH                 "Energy Flow (W)"
               REAL    T  = 500.          "Temperature (K)"

   CONTINUOUS
        T  = T_H_FAR(H, FAR)
        WH = W * H
        WF = (FAR / (1 + FAR)) * W
END PORT
```

Additional equations are generated automatically according to the connections of the port

EQUAL OUT / SUM IN

- Transport Variables:
  - Temperature and Concentrations are very special variables, they travel with the fluid.
  - In case of flow splitting, the temperatures of the leaving flows are equal to the inlet temperature
  - In case of flow merging, the temperature of the leaving flow is the mass flow weighted average of the inlet temperatures:

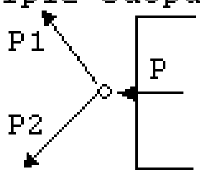




Adding the auxiliary modifiers IN or OUT to SUM or EQUAL. It means that a variable will have the SUM or EQUAL behaviour only if the port has the same direction as the auxiliary modifier. If not, the connecting equation is not generated. Example:

```
PORT fluid                "fluid port"
        SUM REAL w                  "mass flow"
        EQUAL REAL p                "pressure"
        SUM IN REAL E               "energy flow"
        EQUAL OUT REAL T            "temperature"
  CONTINUOUS
     E = w * T
END PORT
```

Multiple output port



Connecting Eqts:

P.w = P1.w + P2.w

P.p = P1.p = P2.p

P.T = P1.T = P2.T

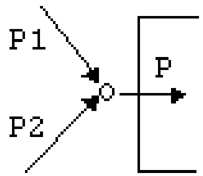
CONTINUOUS Eqts:

P1.E = P1.w * P1.T

P2.E = P2.w * P2.T

P.E = P.w * P.T

Multiple input port



Connecting Eqts:

P.w = P1.w + P2.w

P.p = P1.p = P2.p

P.E = P1.E + P2.E

CONTINUOUS Eqts:

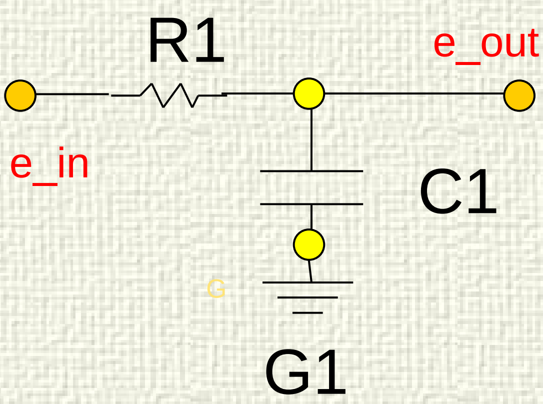
P1.E = P1.w * P1.T

P2.E = P2.w * P2.T

P.E = P.w * P.T

Modelling Languages

Component
LowPassFilter




```
COMPONENT LowPassFilter
    PORTS
        IN    Elec    e_in
        OUT Elec    e_out
    DATA
        REAL Zin=1000   -- Inlet Impedance
        REAL fc=100       -- Cut Frequency

    TOPOLOGY
        Resistor     R1   (R=Zin)
        Capacitor    C1   (C= 1 / (Zin * 2 * PI * fc))
        Ground    G1

        CONNECT e_in  TO  R1  TO  C1  TO  G1
        CONNECT R1  TO  e_out

END COMPONENT
```

Electric Port

```
PORT  Elec
    SUM      REAL    I   -- corriente
    EQUAL  REAL    V   -- tension
END PORT
```

Modelling Languages

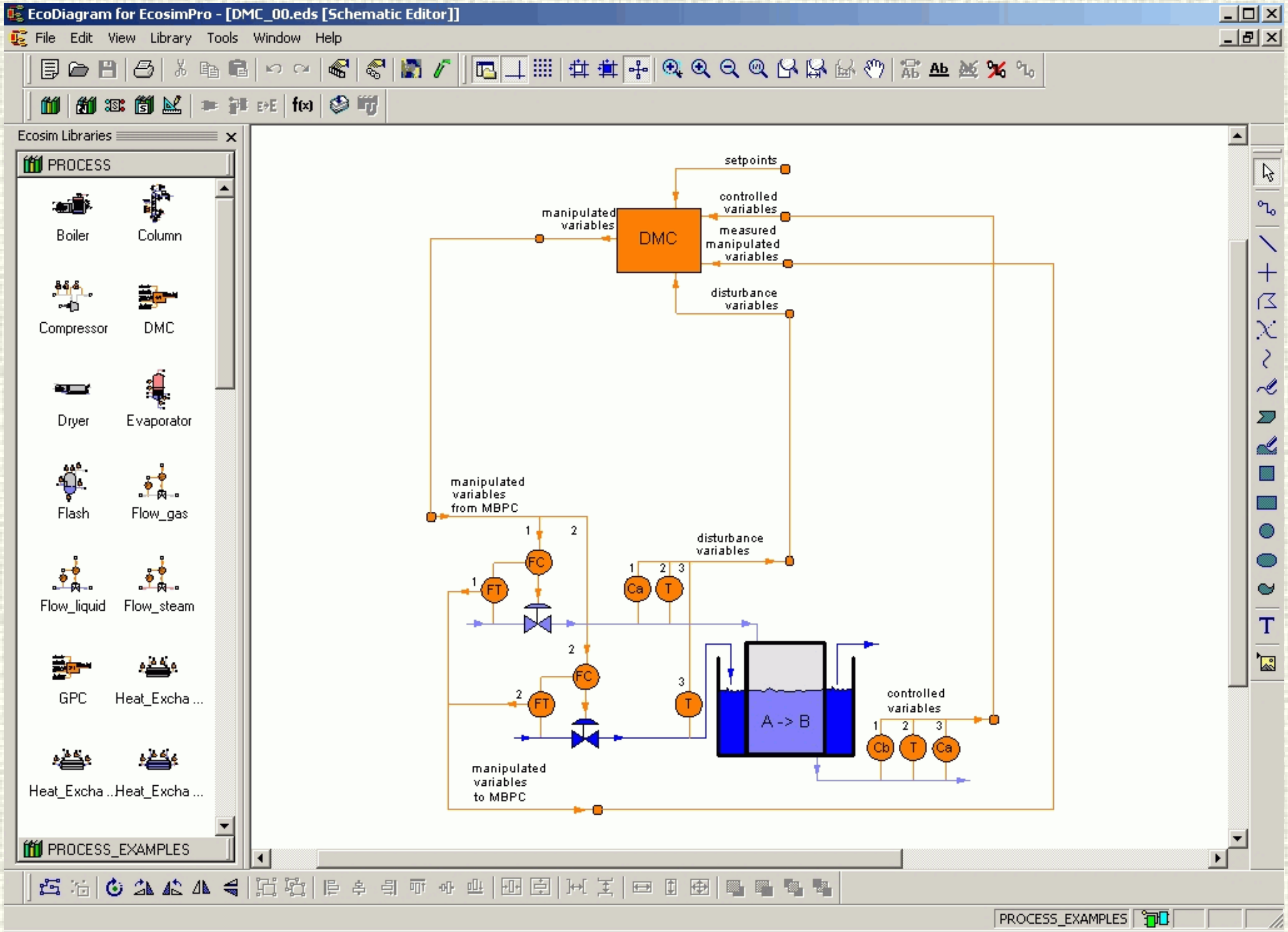
- Component LowPassFilter

```
✓ COMPONENT LowPassFilter
✓     PORTS
✓         IN  Elec e_in
✓         OUT Elec e_out
✓     DATA
✓         REAL Zin=1000   --Inlet Impedance
✓         REAL fc=100     --Cut Frequency
✓     TOPOLOGY
✓         R R1(R=Zin)
✓         C C1(C= 1 / (Zin * 2 * PI * fc))
✓         G G1
✓         CONNECT e_in TO R1 TO C1 TO G1
✓         CONNECT R1 TO e_out
✓ END COMPONENT
```

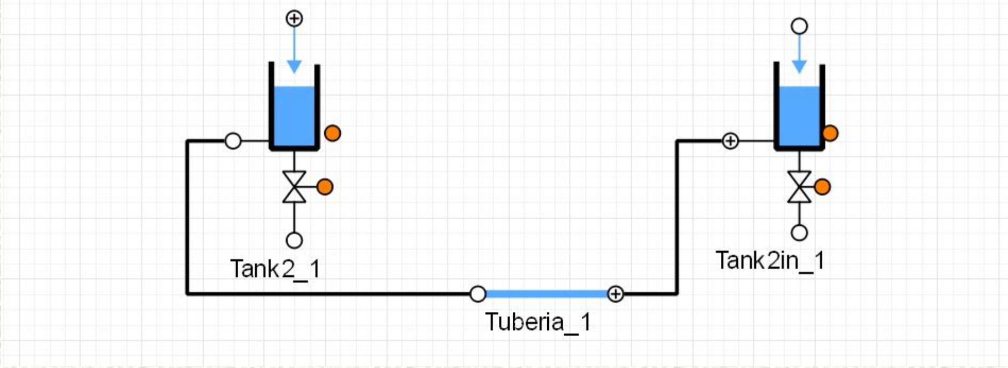
- Experiment

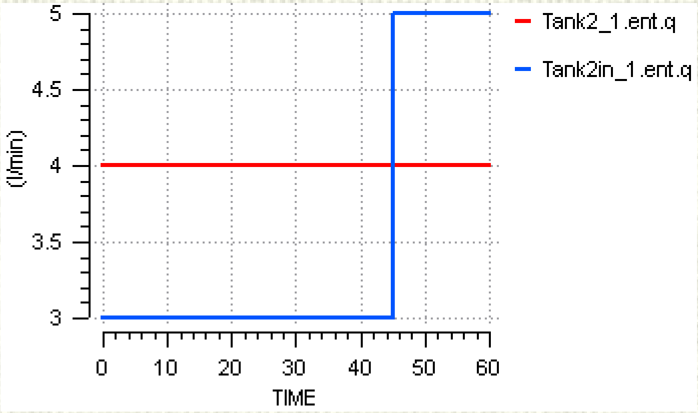
```
BOUNDS
   e_in.v = sin(2*PI*100*(1+5*TIME/0.1)*TIME)
   e_out.i = 0
BODY
   TSTOP = 0.1
   CINT = 0.0002
```

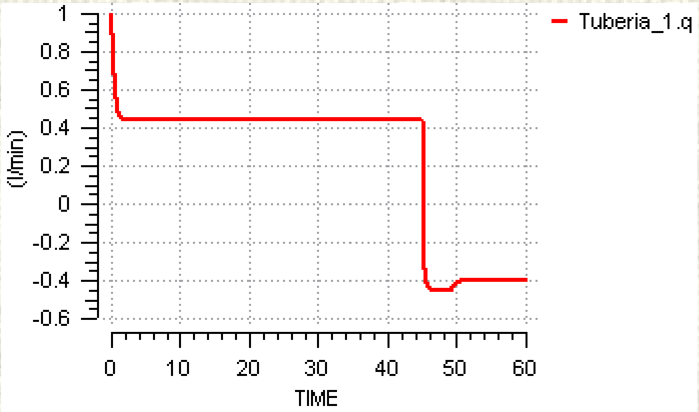
Working with graphical libraries

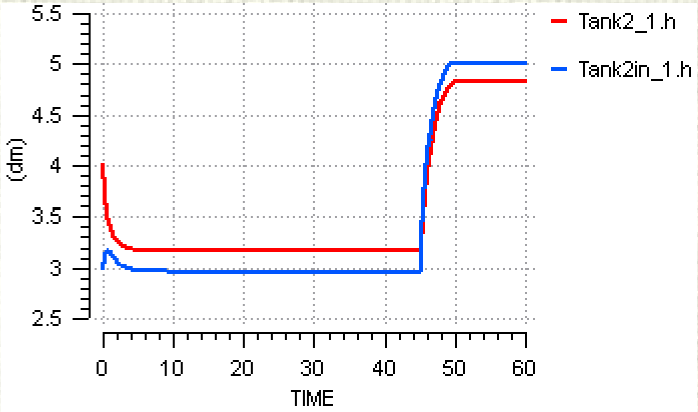


Bidirectional flow

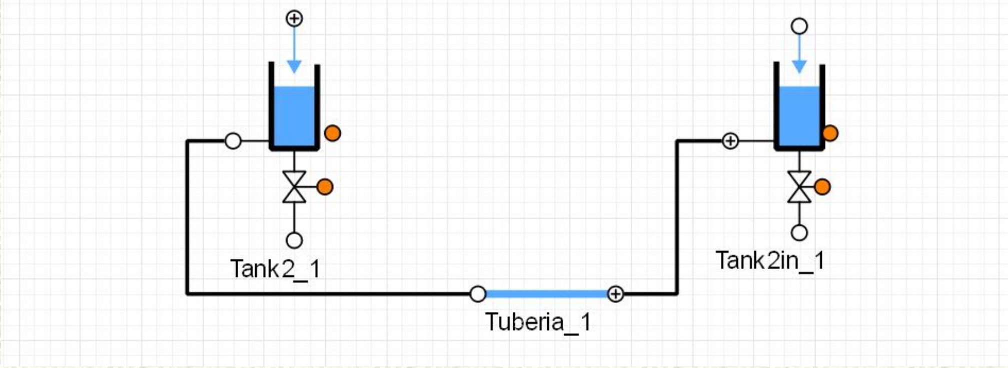


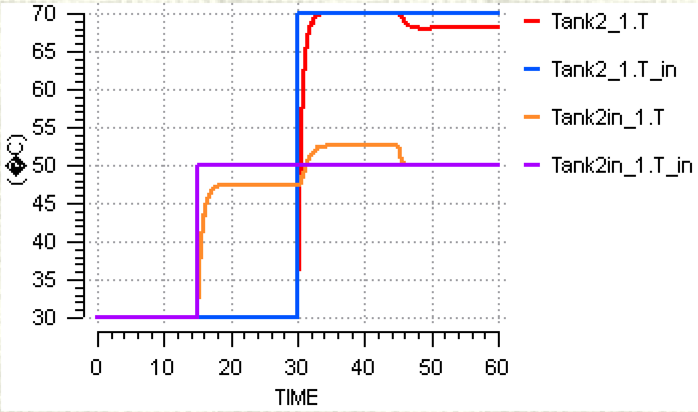


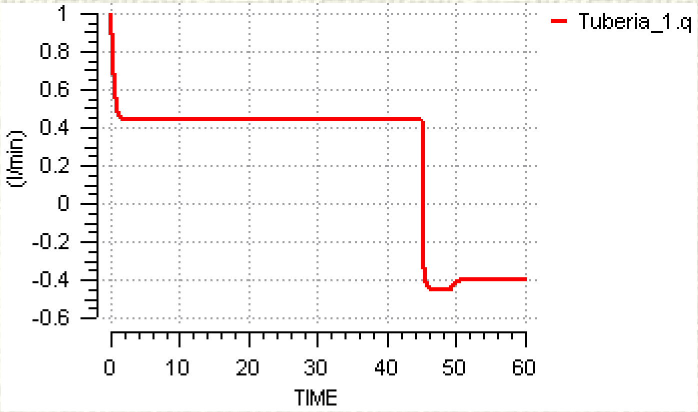


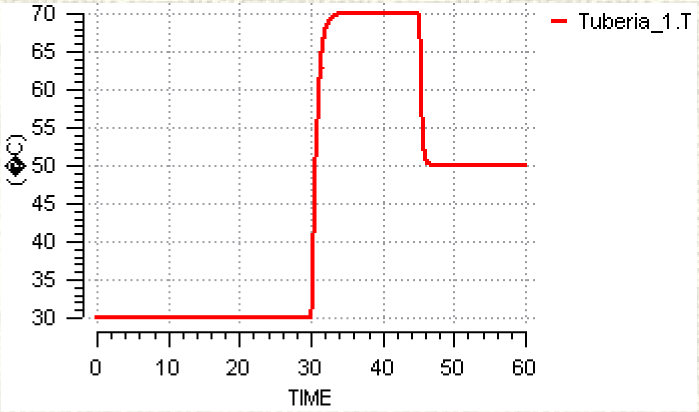


Bidirectional flow









Links among state variables

Sometimes the process model are formulated with algebraic equations that constraint the state variables

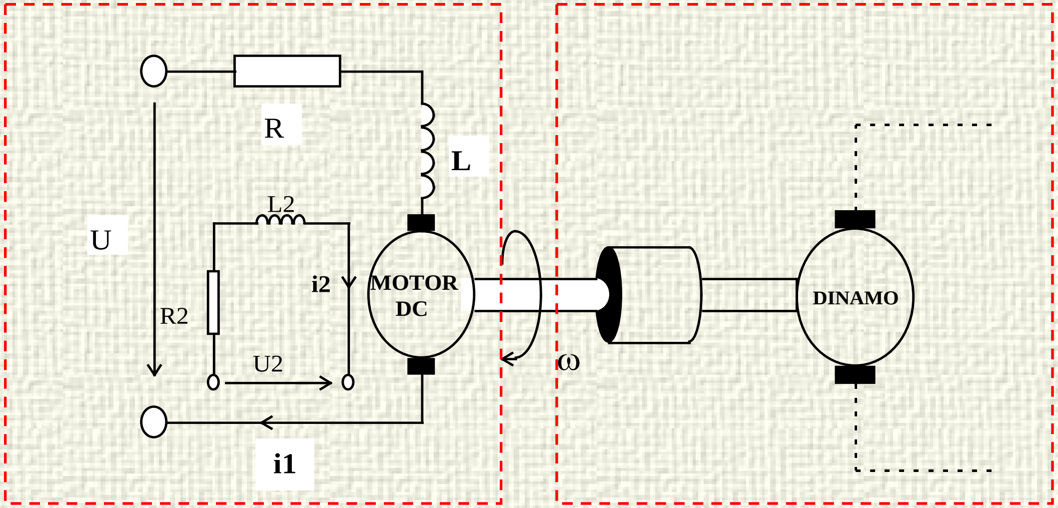
$$\frac{d x_1}{d t} - f_1(x_1, x_2, u) = 0 \qquad \frac{d x_2}{d t} - f_2(x_1, x_2, u) = 0$$

$$g(x_1, x_2) = 0$$

These constraints does not appear in the ODE format and are not considered in the integration methods

High index problems

High index problems can appear as the result of joining together components of a model library due to the bounding equations of the ports.

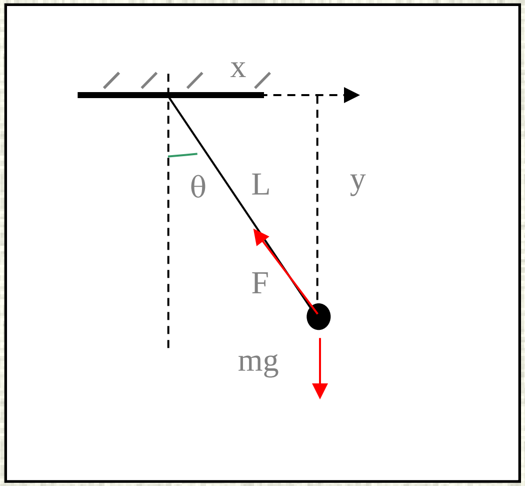


$$J_1 \frac{d\omega_1}{dt} = \ldots + T_1 + T_2 + \ldots$$

$$J_2 \frac{d\omega_2}{dt} = \ldots + T_1 + T_2 + \ldots$$

$$\omega_1 = \omega_2$$

Example: Pendulum



$$m\frac{dv_x}{dt} = -F\frac{x}{L} \qquad \frac{dx}{dt} = v_x$$

$$m\frac{dv_y}{dt} = -F\frac{y}{L} - mg \qquad \frac{dy}{dt} = v_y$$

$$x^2 + y^2 = L^2$$

The model could be described also in polar coordinates with only two state variables

$$m\frac{d\omega}{dt} = -mg\sin(\theta)$$

$$\frac{d\theta}{dt} = \omega$$

They also may appear due to modelling approaches that include non minimum number of state variables

Index of a DAE

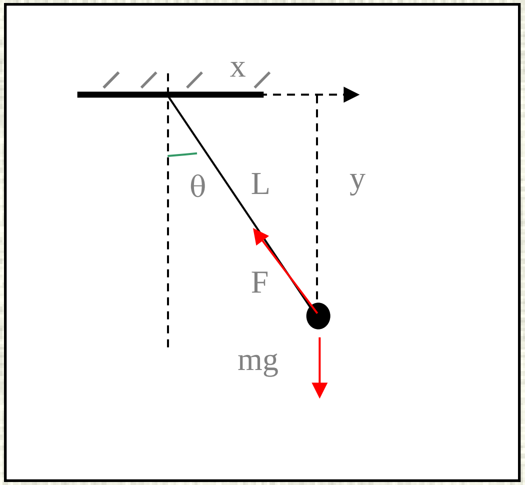
It is possible to reduce a system with links among its state variables to an equivalent ODE one using the Pantelides algorithm, which differentiates n times the state constraint equations.

$$\frac{dx_1}{dt} - f_1(x_1, x_2, u) = 0 \qquad \frac{dx_2}{dt} - f_2(x_1, x_2, u) = 0$$

$$g(x_1, x_2) = 0$$

Index of a DAE system: Number of times that the state constraint equations must be differentiated in order to convert the DAE system into an equivalent ODE one.

Example: Pendulum (index 2)



$$m\frac{dv_x}{dt} = -F\frac{x}{L} \qquad \frac{dx}{dt} = v_x$$

$$m\frac{dv_y}{dt} = -F\frac{y}{L} - mg \qquad \frac{dy}{dt} = v_y$$

$$x^2 + y^2 = L^2$$

$$x^2 + y^2 = L \;\Rightarrow 2xv_x + 2yv_y = 0 \;\Rightarrow 2x\frac{dv_x}{dt} + 2v_x^2 + 2y\frac{dv_y}{dt} + 2v_y^2 = 0$$

1 Solving the sub-set of equations:

$$m\frac{dv_x}{dt} = -F\frac{x}{L} \qquad \frac{dx}{dt} = v_x$$

2 Solving the remaining variables with:

$$y = \sqrt{L^2 - x^2} \qquad v_y = -\frac{xv_x}{y} \qquad \frac{dv_y}{dt} = \frac{-1}{y}\left[-x\frac{Fx}{mL} + v_x^2 + v_y^2\right]$$

SAPIENTIA ADIFICAVIT SIBI DOMVM

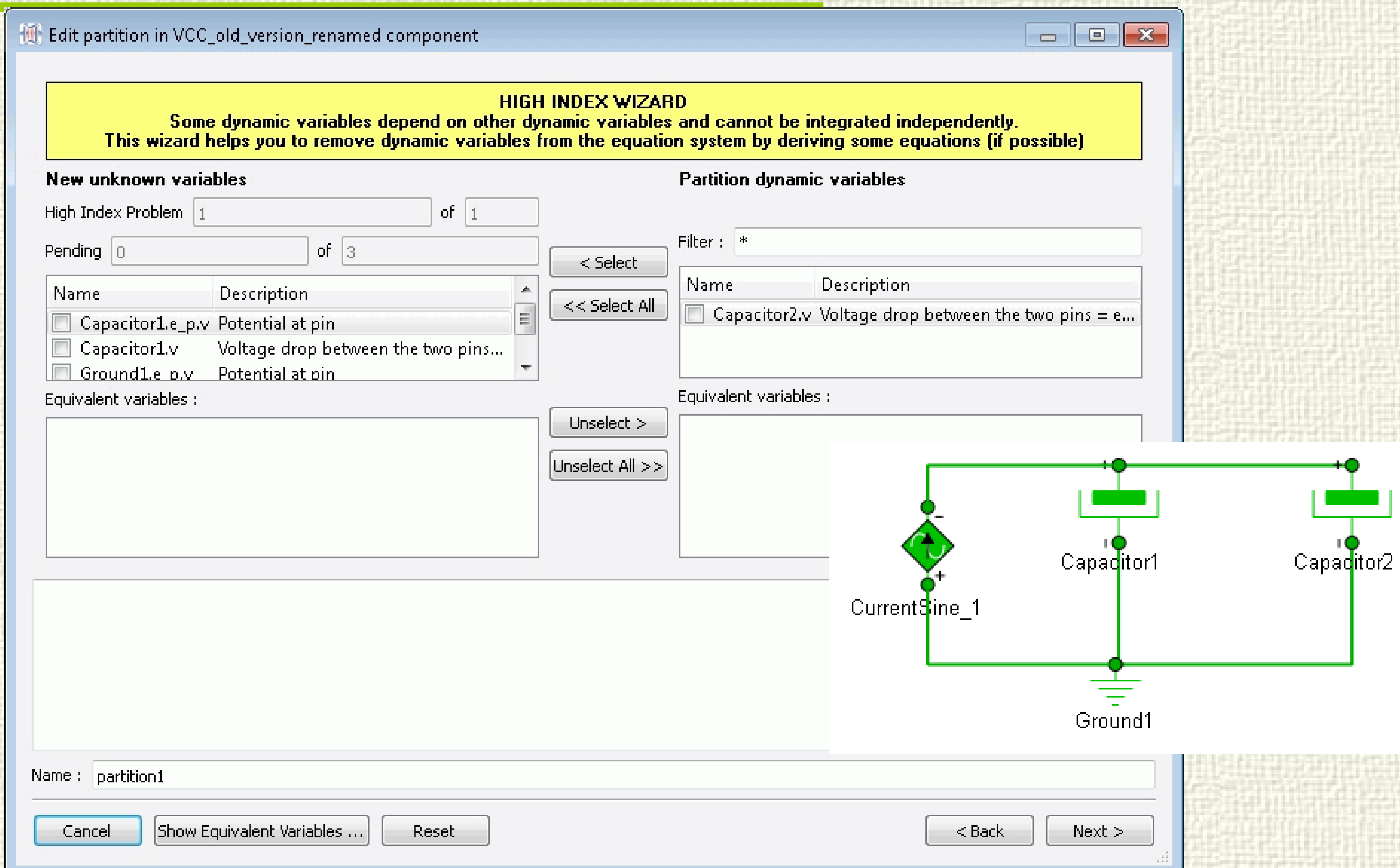
isa

High Index

```
COMPONENT Fuerza

  DATA
          REAL m = 2
  DECLS
          REAL F
          REAL v
           REAL x

  CONTINUOUS
          F = m * v'
          x'= v
          x = exp(-TIME/10) * sin(TIME)

END COMPONENT
```

State variables need explicit time expressions to be used as boundaries and create index problems

High index


Edit partition in VCC_old_version_renamed component
HIGH INDEX WIZARD
Some dynamic variables depend on other dynamic variables and cannot be integrated independently.
This wizard helps you to remove dynamic variables from the equation system by deriving some equations (if possible)
New unknown variables
High Index Problem 1 of 1
Pending 0 of 3
Name
Description
Capacitor1.e_p.v Potential at pin
Capacitor1.v Voltage drop between the two pins...
Ground1.e_p.v Potential at pin
Equivalent variables :
< Select
<< Select All
Unselect >
Unselect All >>
Partition dynamic variables
Filter : *
Name
Description
Capacitor2.v Voltage drop between the two pins = e...
Equivalent variables :
Name : partition1
Cancel
Show Equivalent Variables ...
Reset
< Back
Next >
CurrentSine_1
Capacitor1
Capacitor2
Ground1