Simulation languages

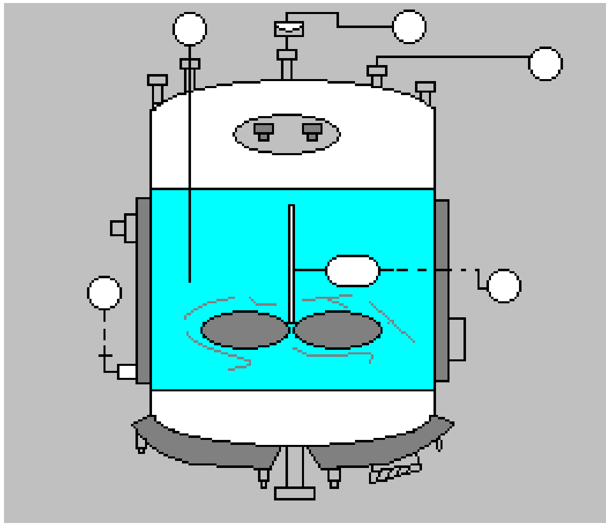
Prof. Cesar de Prada
Dpt. Systems Engineering and Automatic Control
EII, University of Valladolid, Spain
prada@autom.uva.es

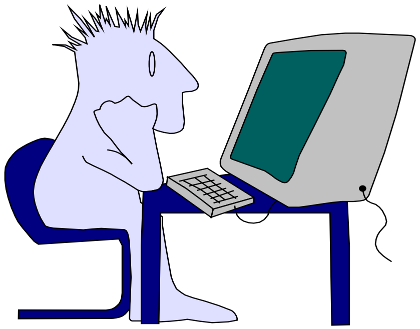
Outline

- ✓ Process Simulation
- ✓ Motivation
- ✓ Types of languages
  - Block oriented
  - Expression oriented
  - Equation oriented (Modelling languages)
  - Physics based
- ✓ How model equations are treated in Modelling languages (EcosimPro)

Digital Simulation

- ✓ Methods and tools oriented to “imitate” or predict the responses of a systems against certain changes or “stimulus” using a computer.





Uses of Simulation

- ✓ Study of a process, what if…? analysis
- ✓ Design (process, control,…)
- ✓ Testing a control system before actual implementation in the plant
- ✓ Personnel training
- ✓ Operation optimization
- ✓ Essays in a virtual plant …..

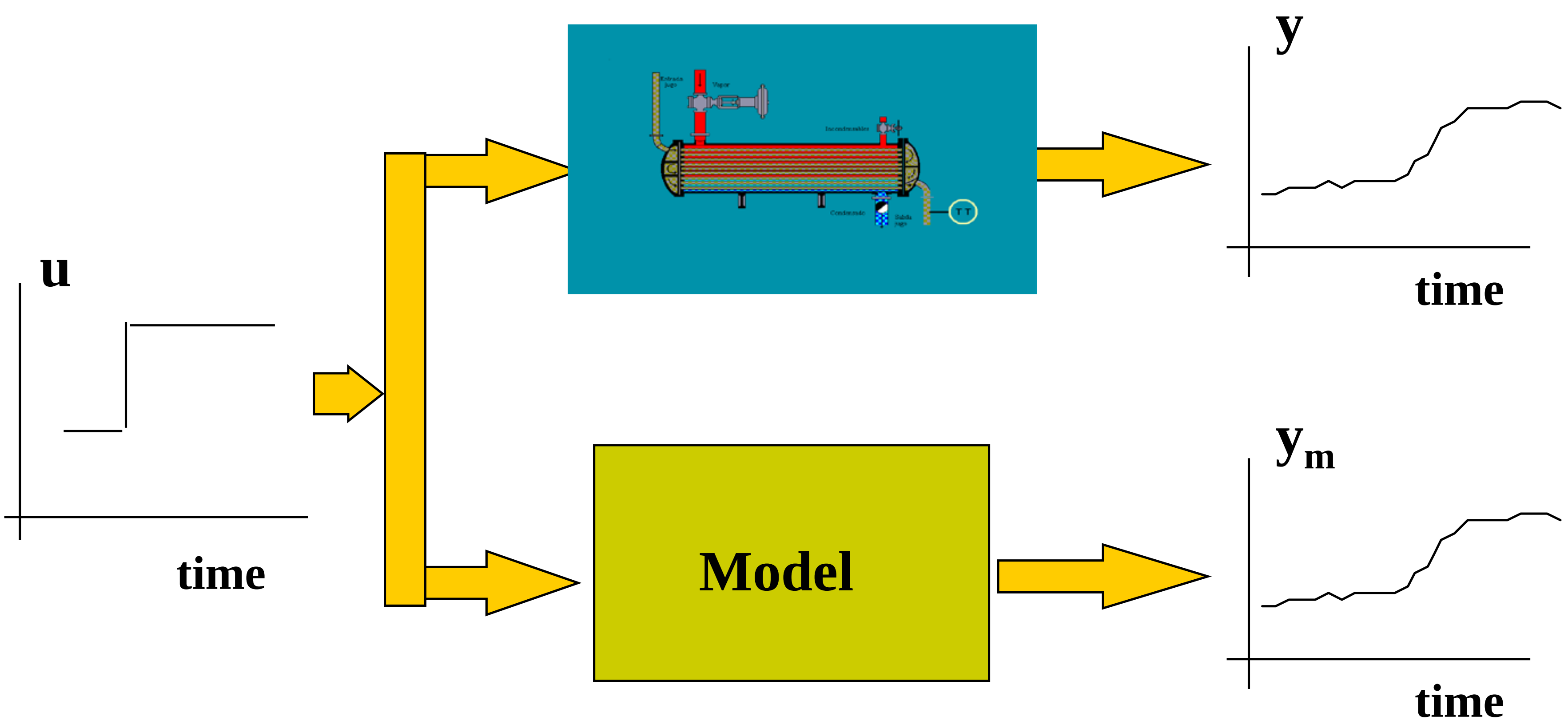
Advantages of the simulation

- Perform changes that, if implemented in the process, will be
  - Very costly,
  - Too slow / fast
  - dangerous, etc.
- Reproduces the experiment as many time as desired under the same conditions
- Saves time
- Provides safety
- Allows sensitivity studies
- Provides a model that can be used for many purposes
- Allows experimenting with systems that are not built yet

Models

- Simulation is based on mathematical models of the processes.
- Mathematical models are set of equations relating the variables of a process and being able to provide an adequate representation of its behaviour.
- They are always approximations of the real world
- Adequacy of a model depends on their intended use
- There are a wide variety of models according to the processes they represent and their aims.

Adequate representation

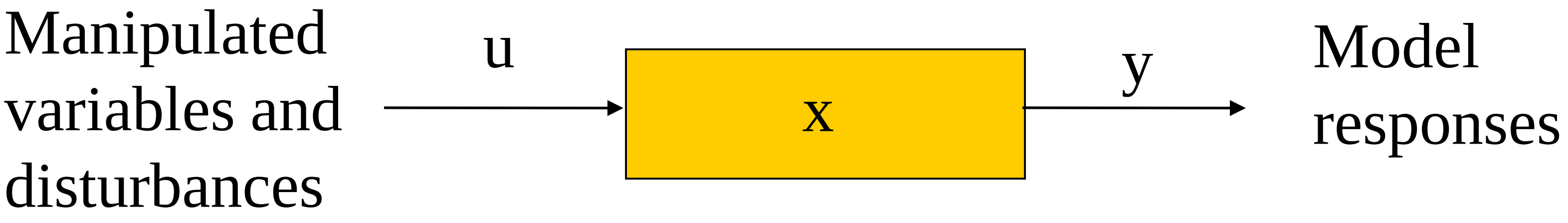


fidelity to the physical asset and facility of use in the intended application

State space models

$$\frac{d\,x(t)}{dt} = f\,(x(t), u(t), t)$$

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

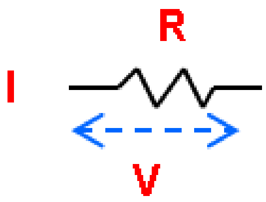


x States

Stages of a simulation project

- ✓ Study the process
- ✓ Set the simulation aims
  - – Specify the relevant variables
- ✓ Develop the model according to the simulation aims.
- ✓ Code the model in a simulation language
- ✓ Set the independent variables and choose the numerical solvers
- ✓ Exploit the results of the simulation

Concepts



Process -> Model

V - I R = 0 ó I = V/R ó V = IR...

Assignment of computational causality

$$V = I * R$$

Experiment

$$R = 10, I = 2$$

Numerical solution

$$V = 2*10 = 20$$

Simulation Languages

Computer program providing tools for:

- ✓ Describing the model and assigning computational causality
- ✓ Defining the experiments to be performed
- ✓ Solving numerically the set of equations
- ✓ Visualizing the results and communicating with the external world

Advantages

- Provide support in all phases of model development and exploitation
- Allows concentrating in the problem and the results, not spending time and efforts in programming
- Gives reliability to the numerical results
- Allows saving time
- Allows the non-expert in computing or numerical methods to solve complex models

First principles models

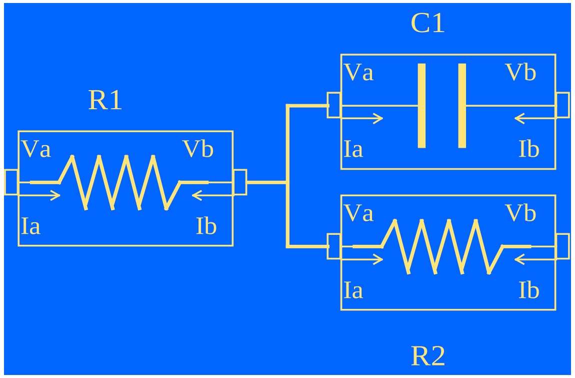
- Based on knowledge of the process and nature laws (Physics, chemistry,…)
- Sometimes are difficult to formulate from the scratch, requiring trained people, large development times, costs,..
- They need to be tested and validated

- This may limit their use in many fields (Design, decision making, training,….
- But,….which is the cost of non-using them?

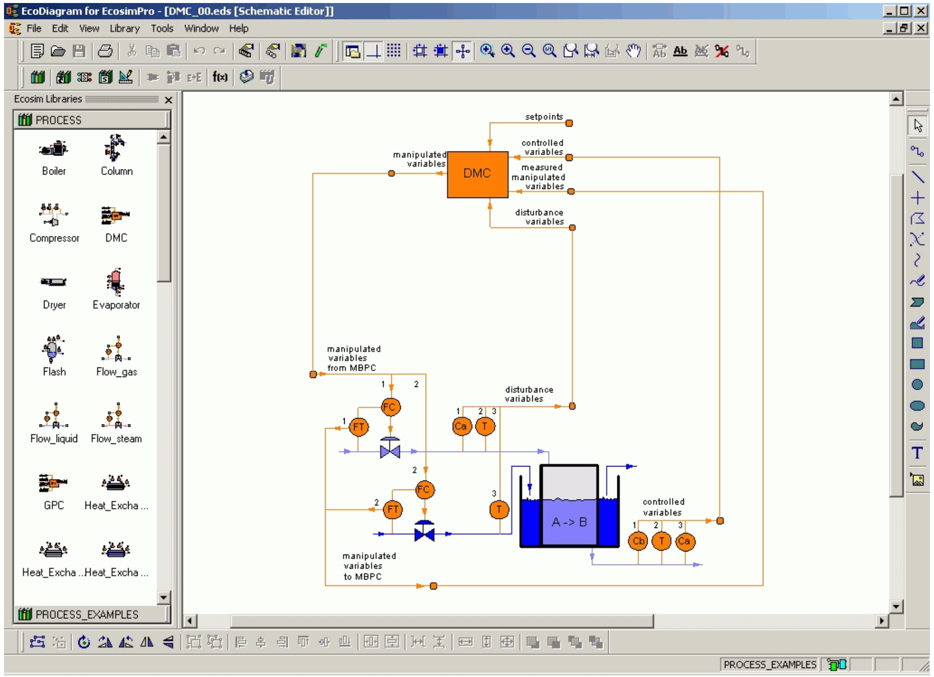
Solution: Libraries of models

- ✓ Models are built linking the tested modules or components of a model library
- ✓ Each component of the library contains the mathematical model of a process and can be configured by parameterization to fit the user needs
- ✓ Each component can be linked to others by an interface or port in order to built more complex models

Physical properties data bases and good user interfaces are also required



Model Libraries

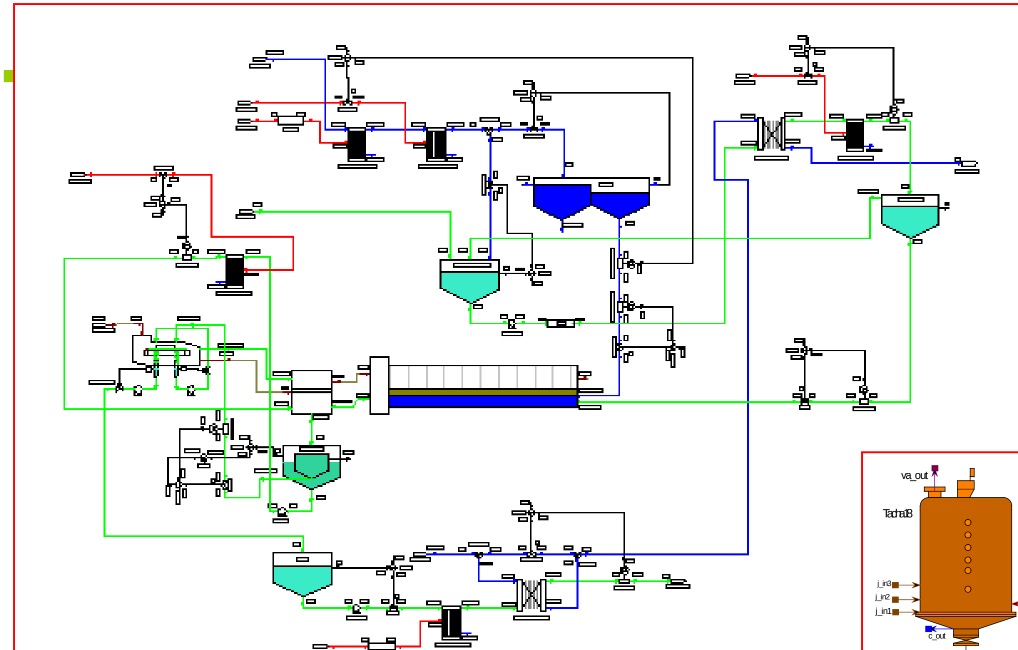


✓ Sets of components representing different processes, devices, etc.

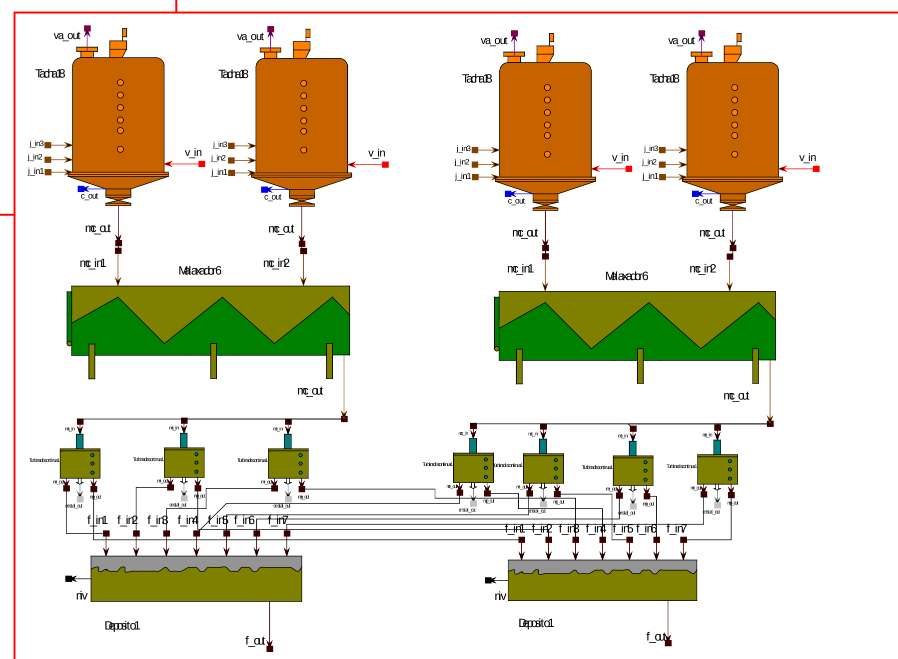
✓ Each one contains its mathematical model and connections to the external world

✓ Components can be parameterized to adapt them to the user requirements

Model Libraries



Select and connect components as in the real world



After parameterization, simulation code is generated

Model Libraries

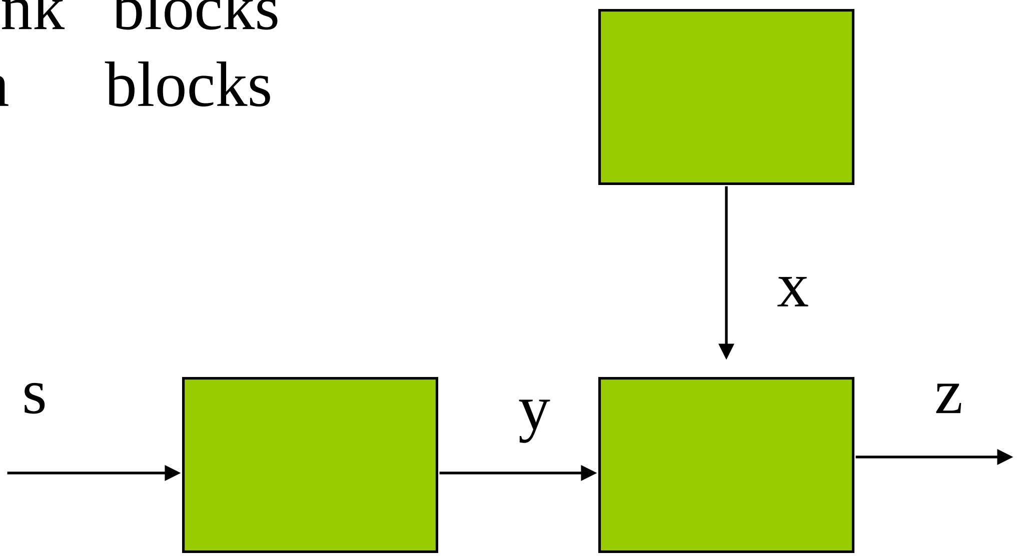
- Modular modelling:
  - Facilitates the re-use of models in different applications
  - Facilitates the use of simulation to those non-experts in simulation, but knowing the system to be simulated
- Modularity: Independent description of every module of the library
- Abstraction: Use the modules without knowing its internal details (model equations, etc.)
- Hierarchy: New modules can be built by linking the existing ones

Types of simulation languages according to the way they support modularity

- Block oriented languages
- Expression oriented languages CSSL'67
- Equation oriented (Modelling languages)
- Automated modelling (SIMPD)

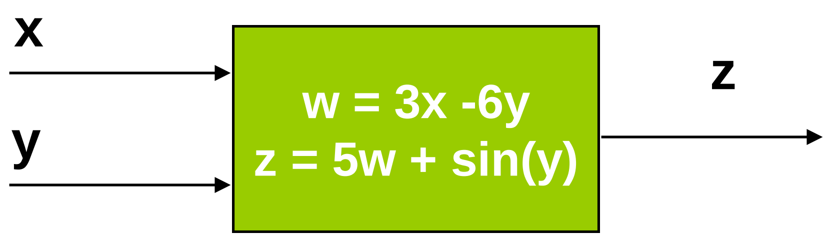
Block oriented languages

Simulink blocks
Tutsim blocks



Each block has fix input and output variables and contains equations or code to compute the value of the output variables as a function of the value of the input ones

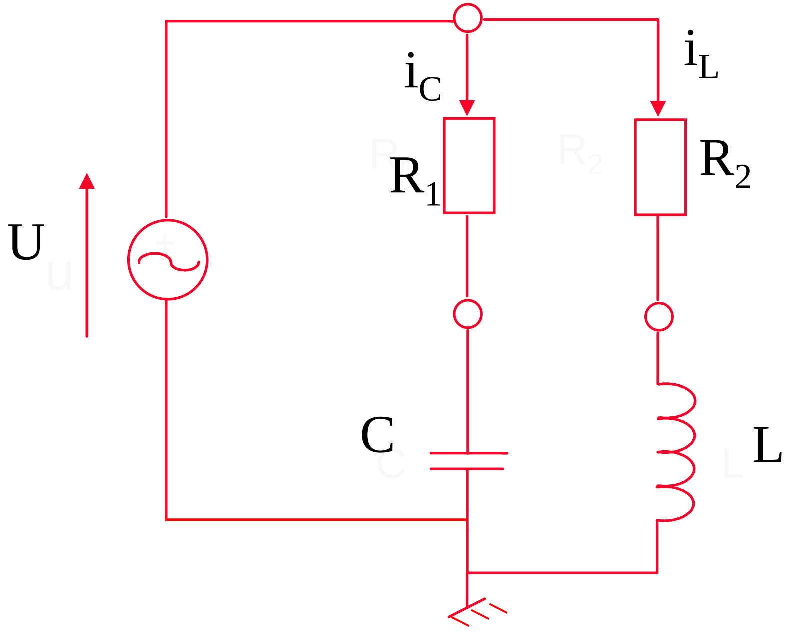
Blocks or macros



- ✓ Encapsulated code that is not manipulated by the simulation environment
- ✓ Fix computational causality, imposed by the inputs and outputs of the block
- ✓ Connections between blocks by linking input - output variables
- ✓ Block diagrams do not mimic the physical layout but the mathematical one

Simulink

Implementation of the model is done using predefined blocks that carry out specific operations and are linked together to perform the operations of the model equations



Physical system

$$i_C = (U - U_C) / R1$$

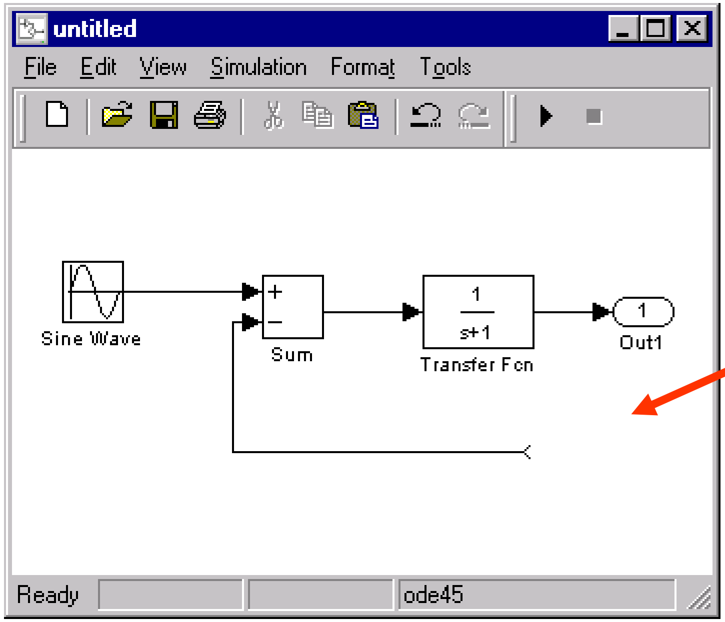
$$U_L = U - i_L \times R2$$

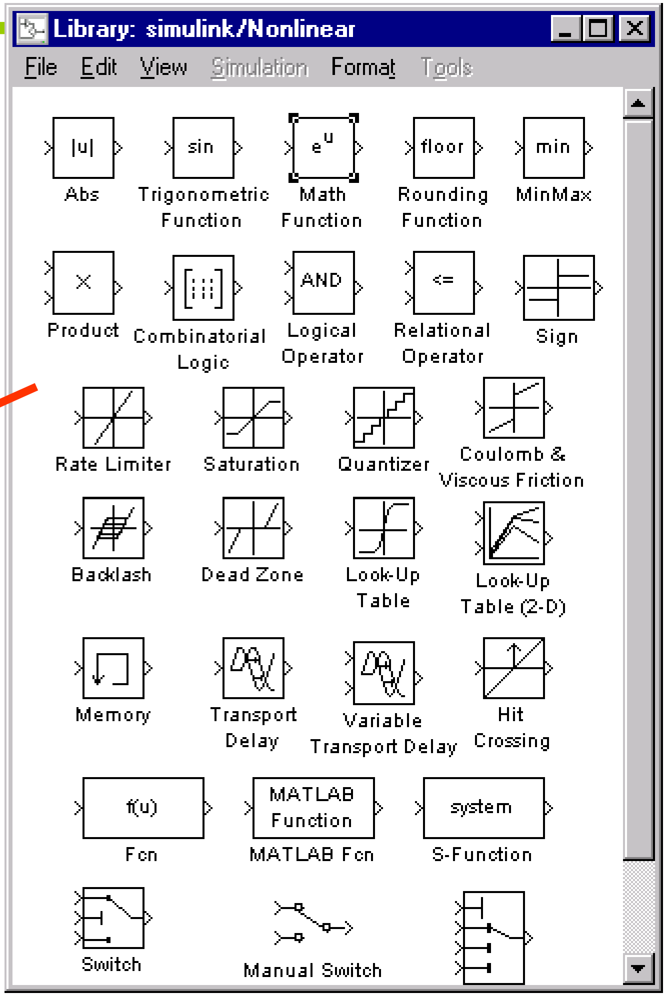
$$\frac{dU_C}{dt} = i_C \ / \ C$$

$$\frac{di_L}{dt} = U_L \ / \ L$$

Model equations

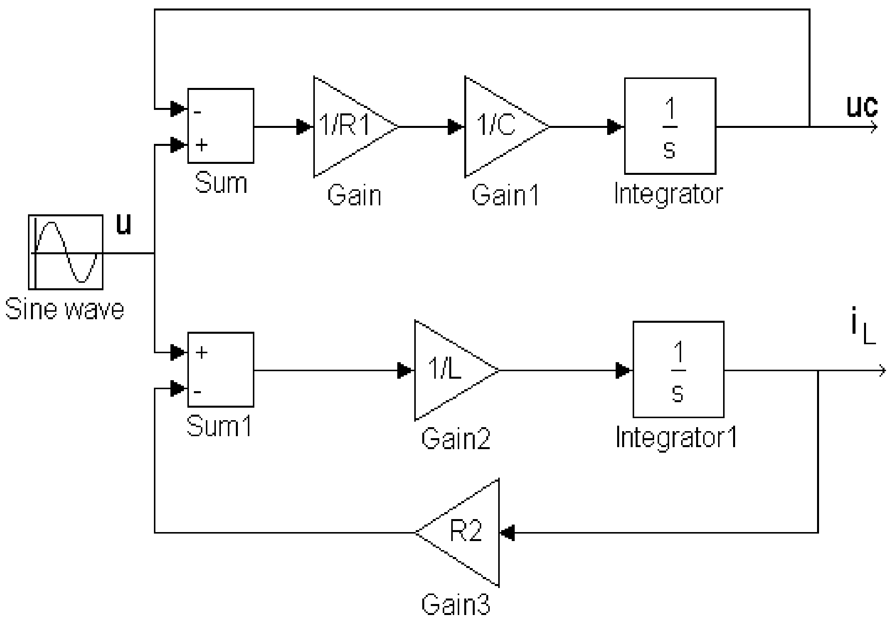
Simulink





The block diagram is built graphically from the blocks of the library

Block oriented languages: Simulink



$$i_C = (U - U_C) / R1$$

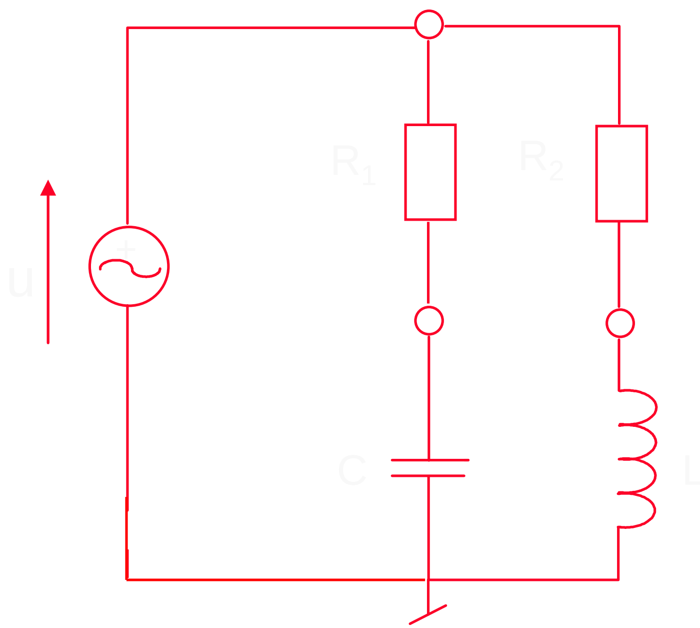
$$U_L = U - i_L \times R2$$

$$\frac{dU_C}{dt} = i_C \; / \; C$$

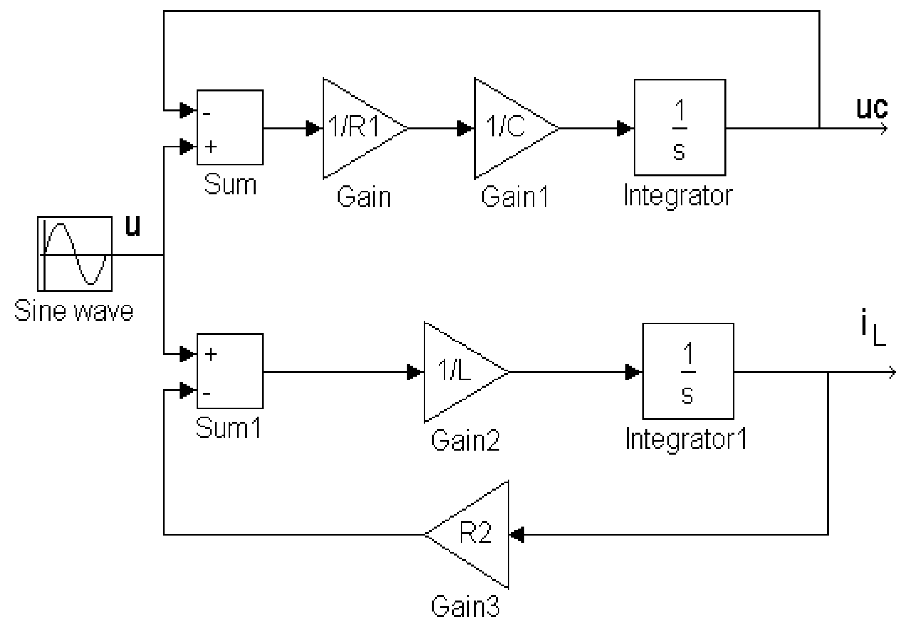
$$\frac{di_L}{dt} = U_L \; / \; L$$

Simulink

With block oriented languages, the user describes the mathematical model, not the physical system



Physical system



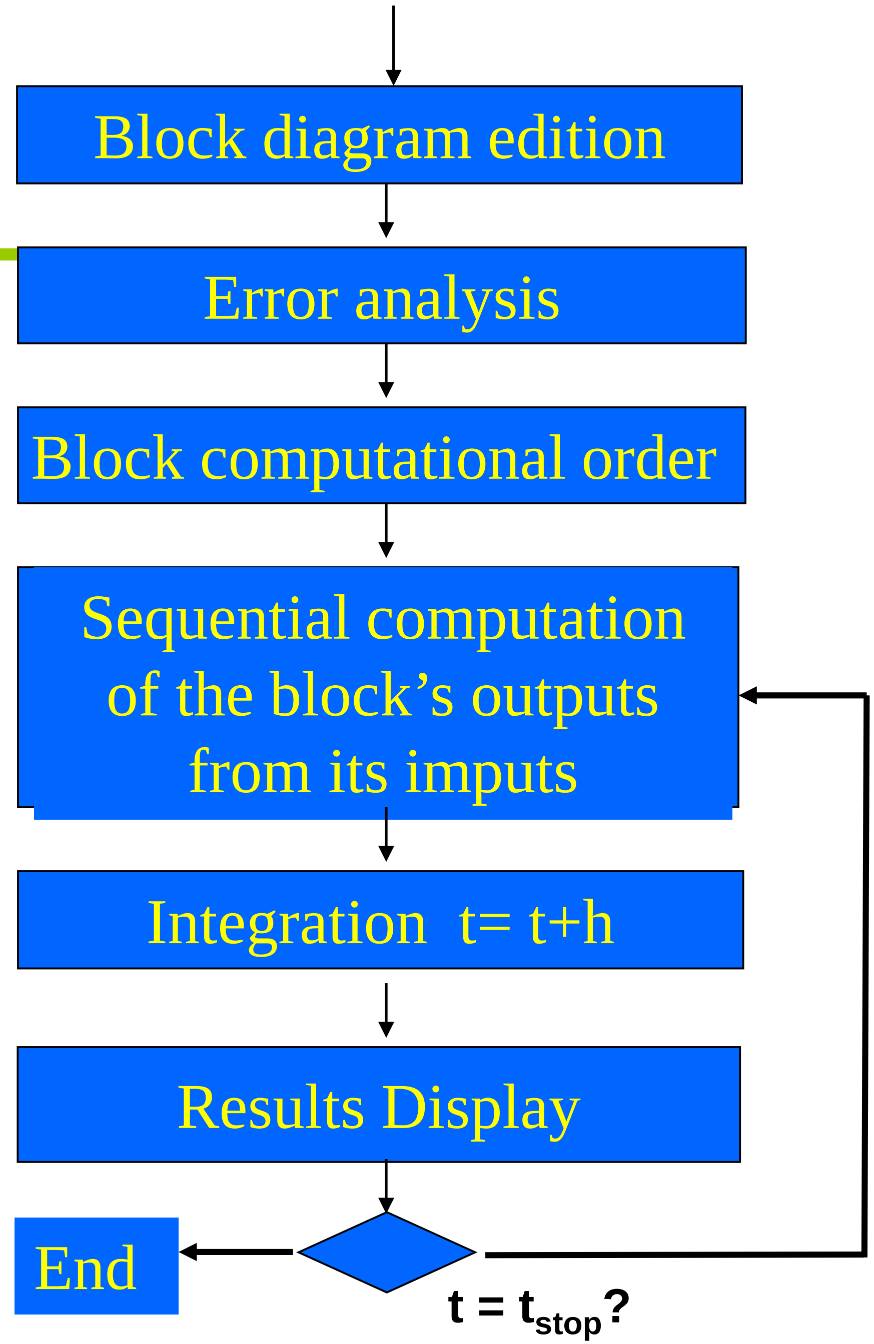
Block diagram

Structure

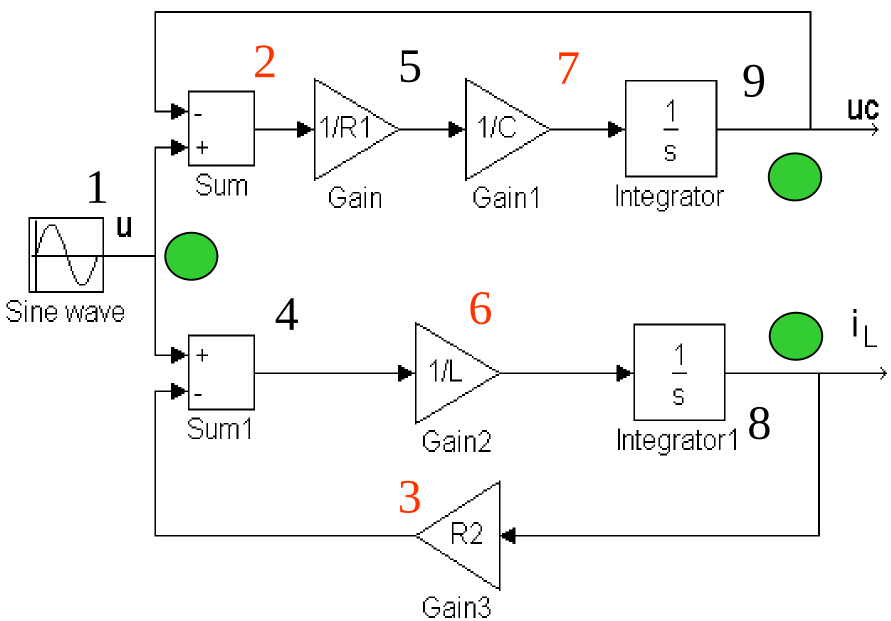
CSMP 1130
SADS
DSL/90

….

EASY-5
TUTSIM
Simulink



Block's computational order



Computational order **1, 9, 8, 2 ,3, 5, 4, 7, 6, 9, 8**

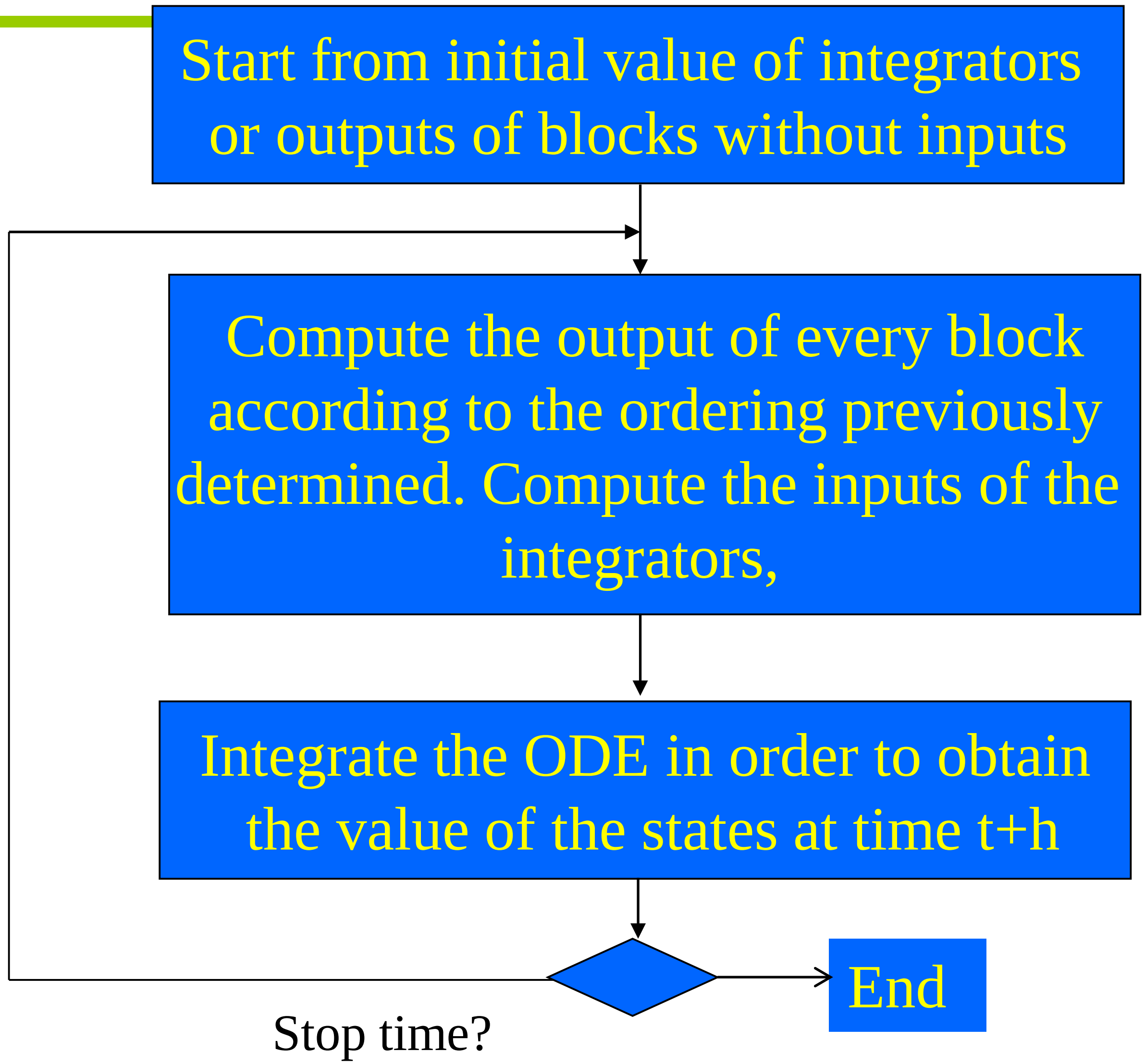
States or known values initially

1 Starting from the blocks with known initial values, check which blocks can be executed as all their inputs are known.

2 Write them down in a list and iterate with the new set of known blocks until all blocks are used up.

3 If any new block is added to the list in a full iteration over all blocks, an algebraic loop is detected.

Integration architecture

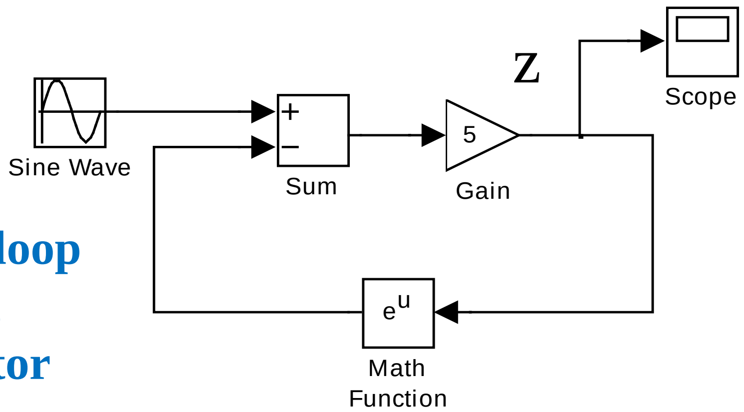


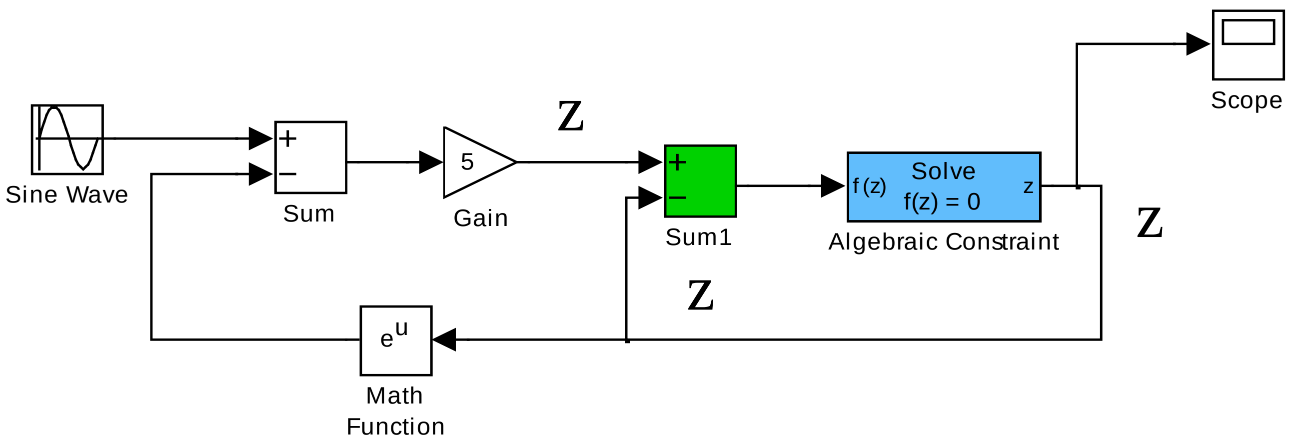
Algebraic loops

$$z = 5\,(\sin(t) - e^{z})$$

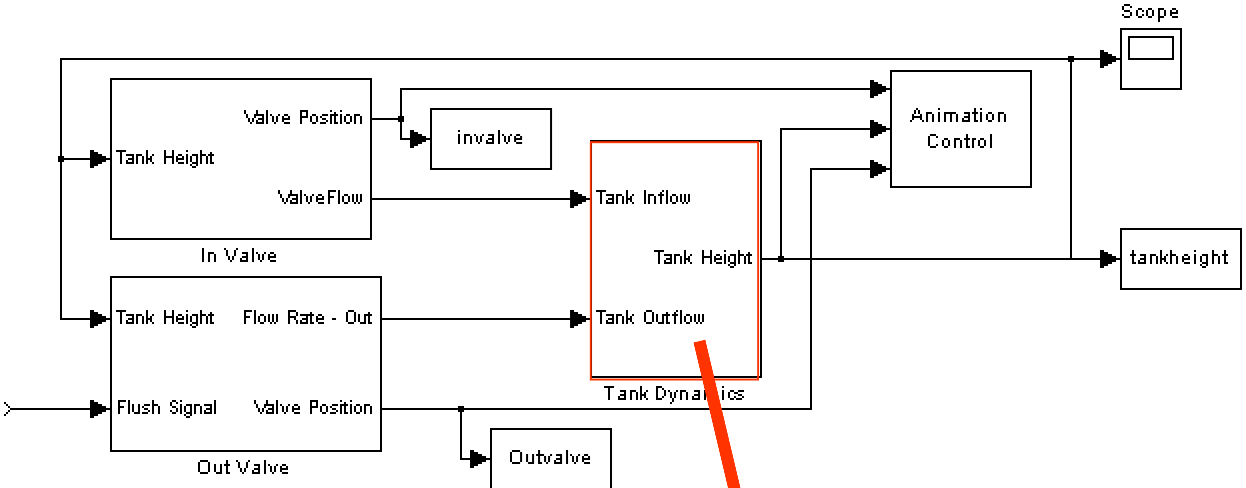
A special block need to be added. Iterations until convergence are required

Closed loop without integrator



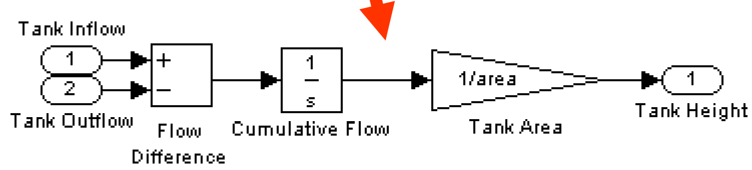


Hierarchical blocks



Modularity

Hierarchy



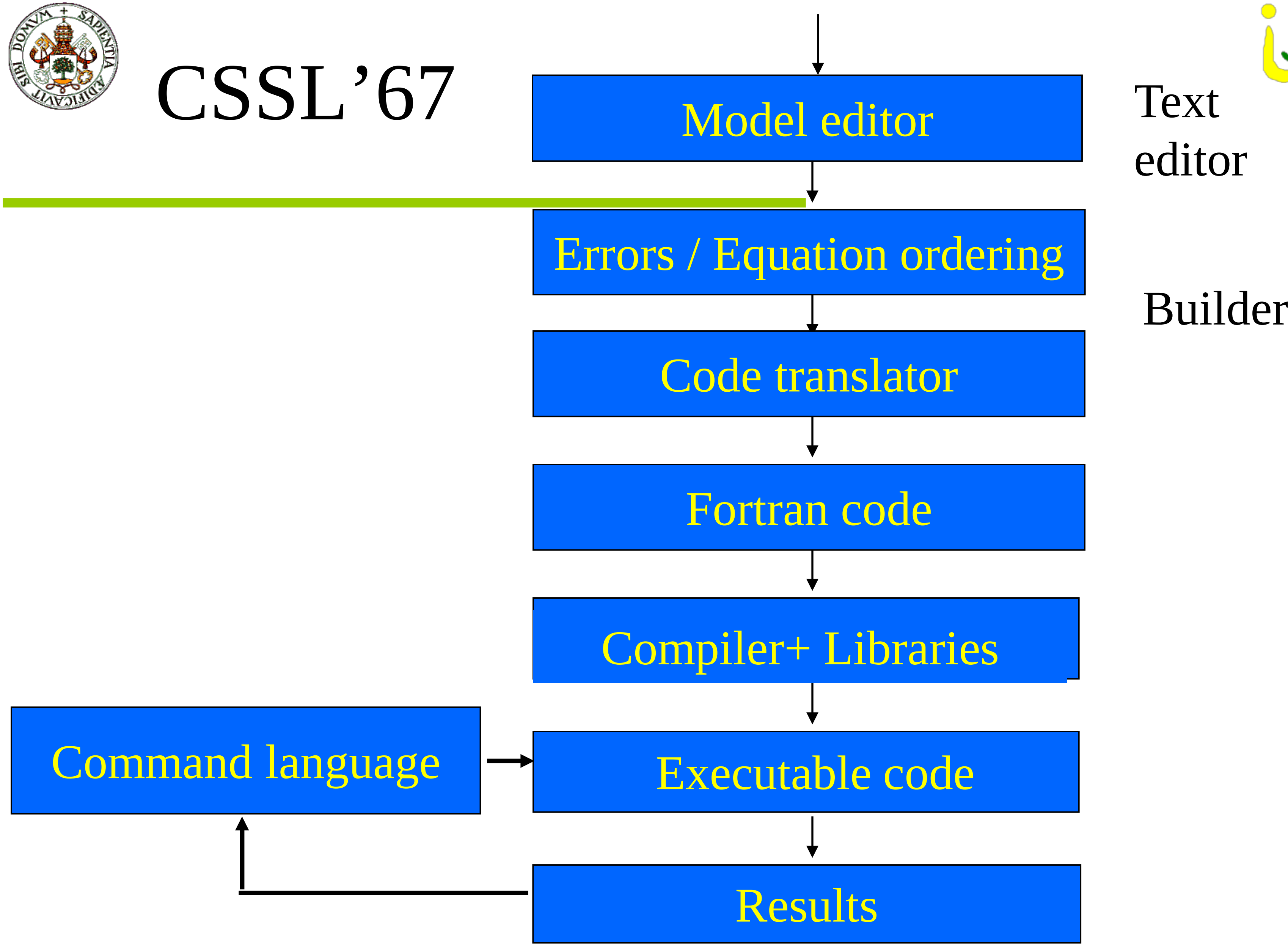
Block oriented languages

- There are easy to use and intuitive
- Modular and hierarchical architectures
- Model description does not match neither the physical process not the equations.
- There are difficult to build and debug in case of models with a large number of blocks
- Fix computational causality
- Slow: interpreters
- Algebraic loops must be explicitly solved with additional blocks
- Limited separation model-experiment

Expression oriented languages

- Standard CSSL’67 (Simulation 1967 Vol.9, pp.281-303)
- Direct declaration of the model equations
- Model description is given a temporal structure
- Separation model-experiment: command language
- Code generators, compiled simulation code: Speed
- Open to the outside world: Call...
- Reuse of code: Macros

CSSL’67


Model editor
Text editor
Errors / Equation ordering
Builder
Code translator
Fortran code
Compiler+ Libraries
Command language
Executable code
Results

CSSL’67

Program

Initial

End

Initial conditions. Code executed once at t= 0

Description model structure

Dynamic

Derivative

End

Continuous equations

Discrete

End

Discrete equations

Fix computational causality

End

Simulation code similar to the mathematical model

Terminal

End

Final computations. Code executed once at tstop

End

Computations

Program

Initial

End

Dynamic

Derivative

End

Discrete

End

End

Terminal

End

End

Initialization of variables, including states

Expressions evaluated and integrated every integration interval

Expressions evaluated at certain times (Synchronous or asynchronous modes) or when a event takes place.

Global variables

Transfers to initial region are possible to create loops.

Language

Equations similar to Fortran: exp, sin , IF THEN ELSE,...

Primitives: BOUND, REALP, DELAY,….

Function generators: SIN, PULSE,...

Tables 2D & 3D

Implicit equations: IMPLC

Integrators: INTEG, several methods: Stiff, DASSL,...

Event and discontinuities treatment: SCHEDULE, INTERVAL,..

External calls: Call...

Equation ordering

Automatic ordering of the equations following an algorithm similar to the one used with blocks

Procedural regions with fix sequential order

CONSTANT R = 4.		CONSTANT R = 4.
V = INTEG(F, 0.1)		S = 3.14 * R * R
F = S + exp(R)	→	F = S + exp(R)
S = 3.14 * R * R		V = INTEG(F, 0.1)

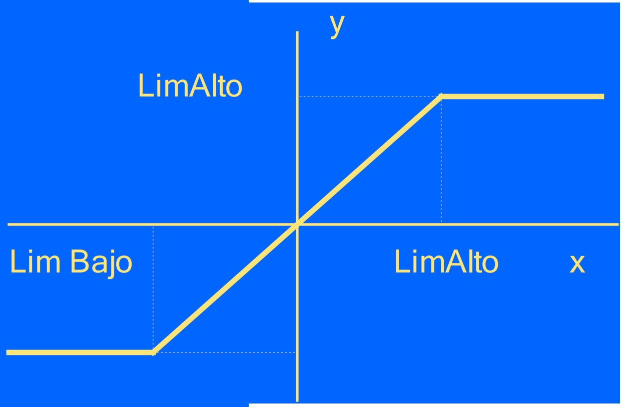
Fix computational causality

CSSL'67 ACSL

```
program prueba
initial
     constant  x0=0.1,  tmax=3.
     cinterval cint=0.35
     algorithm ialg=3
end
derivative
    constant tau=2.
    z = 5* x – 3*y
    x = integ(tau*y + sin(x) , x0)
    y = bound(-1.,1.,x)
    termt(t.gt.tmax)
end
end
```

$$\frac{dx}{dt} = \tau y(x) + \sin(x)$$

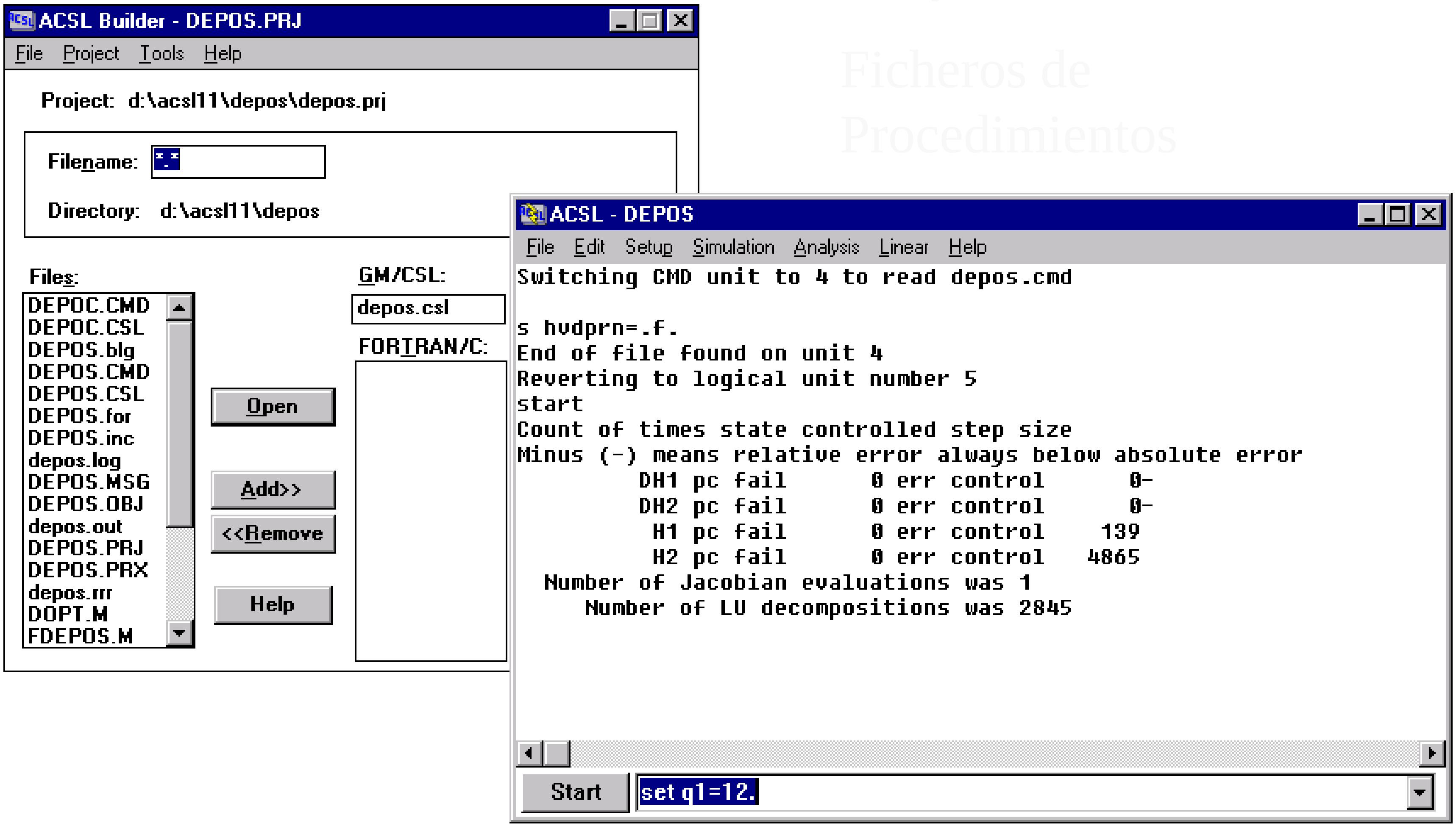
$$z = 5x - 3y$$



ACSL

Lenguaje de comandos: start, set, plot, analyse,…

Ficheros de Procedimientos



Modularity

- A modular approach provides support to the description of a complex system using pre-defined sub-systems
- Helps library maintenance
- Helps team working
- Helps improving the readability and use of the simulation code

Macros

Macros encapsulate simulation code to facilitate its repetitive use in different places of the model description

There are different from subroutines: The code of a macro is expanded and analysed with the other equations before compilation

```
………..

Valve(u,1)

………….

Valve(aper,6)
```

```
MACRO Valve (a,n)
        dp&n=(pe&n - ps&n)/den
        q&n = a*sqrt(dp&n)
MACRO END
```

Macros

Valve(u,1)

…………..

Valve(aper,6)

dp1=(pe1 - ps1)den

q1= u*sqrt(dp1)

………...

dp6=(pe6 - ps6)/den

q6= aper*sqrt(dp6)

```
MACRO Valve ( a,n)
        dp&n=(pe&n - ps&n)/den
        q&n = a*sqrt(dp&n)
MACRO END
```

Fix computational causality

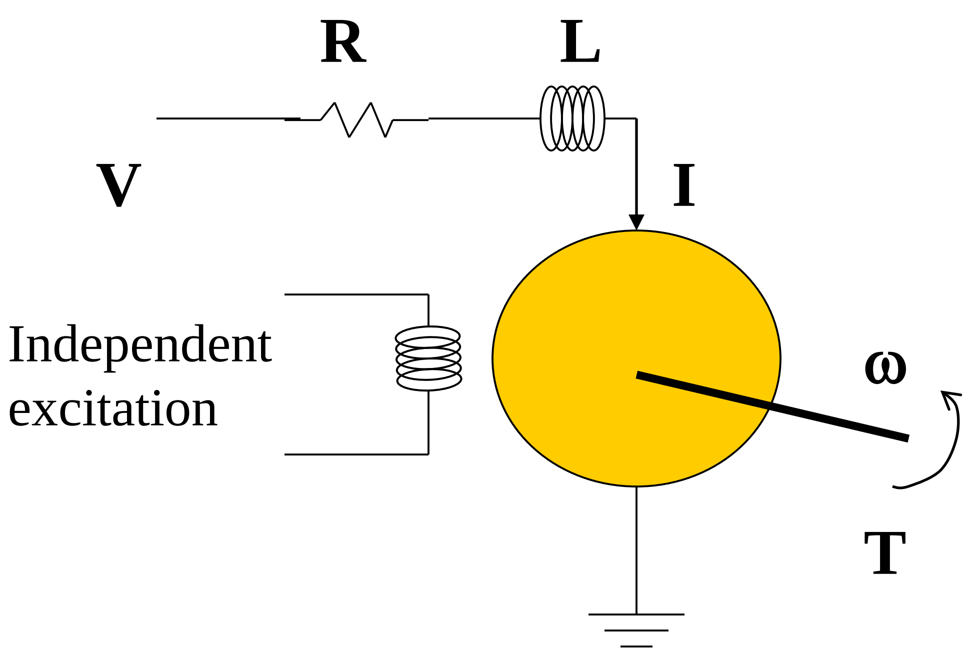
It is difficult to operate with parameters in long chain calls

Global variables

Modelling Languages

- Direct declaration of the model equations
- Model description is given a temporal structure
- Separation model-experiment
- Object oriented
- Code generators, compiled simulation code
- True modular modelling: They do not have fix computational causality

Example: DC Motor



T external torque

$k_e\omega$ e.m.f.

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

$$V = RI + L\frac{dI}{dt} + k_e\omega$$

If $L \cong 0$,

$$\boxed{\begin{aligned} J\frac{d\omega}{dt} &= kI - f\omega - T \\ V &= RI + k_e\omega \end{aligned}}$$

Structure of a model

```
COMPONENT motorDC

  DATA
     REAL J = 2          "momento de inercia"
     REAL K = 3          "constante de par"
     REAL f = 0.01       "friccion"
     REAL R = 0.1        "resistencia"
     REAL Ke = 0.5

  DECLS
     REAL T              “par”
     REAL  w             “velocidad”

  INIT
     w = 30         -- initial condition

  DISCRETE
    WHEN (w > 1500) THEN
         T = 20
    END WHEN

  CONTINUOUS
      J*w'= K*i - f*w – T
      v = R*i + Ke*w

END COMPONENT
```

Description of the model is similar to its mathematical formulation

Executed only once at time 0

Executed only when a logical condition is true

Executed continuously

Separation model- Experiment

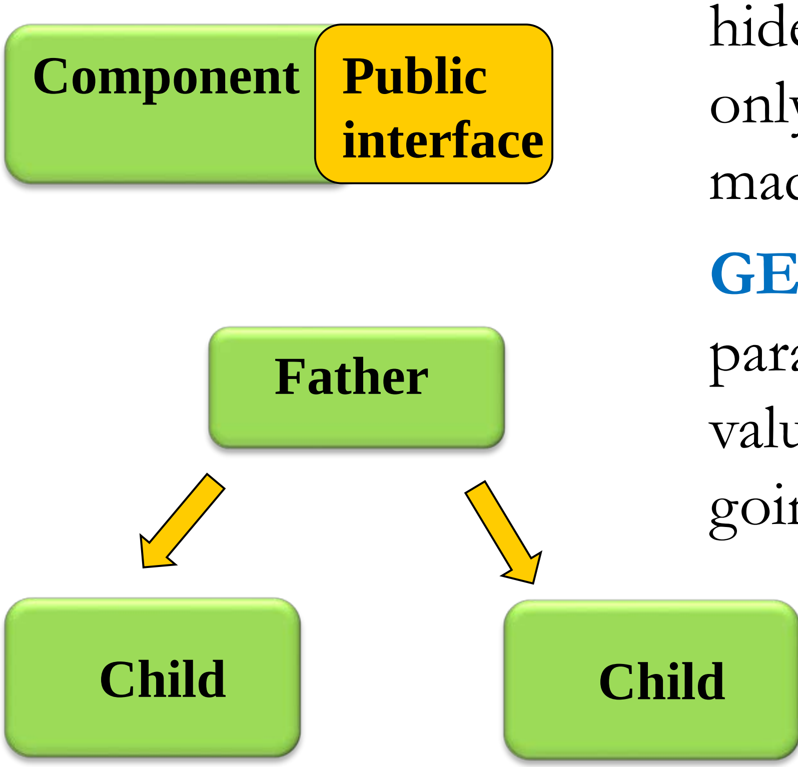
```
COMPONENT motorDC

  DATA
     REAL J = 2          "momento de inercia"
     REAL K = 3          "constante de par"
     REAL f = 0.01       "friccion"
     REAL R = 0.1        "resistencia"
     REAL Ke = 0.5

  DECLS
     REAL T              "par"
     REAL  w             "velocidad"

  INIT
     w = 30          -- initial condition

  DISCRETE
    WHEN (w > 1500) THEN
         T = 20
    END WHEN

  CONTINUOUS
      J*w'= K*i - f*w – T
      v = R*i + Ke*w

END COMPONENT
```

Component

```
EXPERIMENT exp1 ON motorDC.motor2
DECLS
INIT    -- set initial values for variables
   w = 0
BOUNDS    -- set expressions for
boundary      variables: v = f(t,...)
   v = 10
   T = 2
BODY
  REPORT_TABLE("reportAll", " * ")
  TIME = 0
  TSTOP = 5
  CINT = 0.1
  INTEG()
END EXPERIMENT
```

Experiment

Object oriented modelling



ENCAPSULATION: A component hides the complexity of the model as only a certain part of the model is made public

GENERICNESS: generic parameters/modes that are given values only when the component is going to be used

INHERITANCE: A component can inherit the behaviour and properties of other(s)

Connecting modules by ports

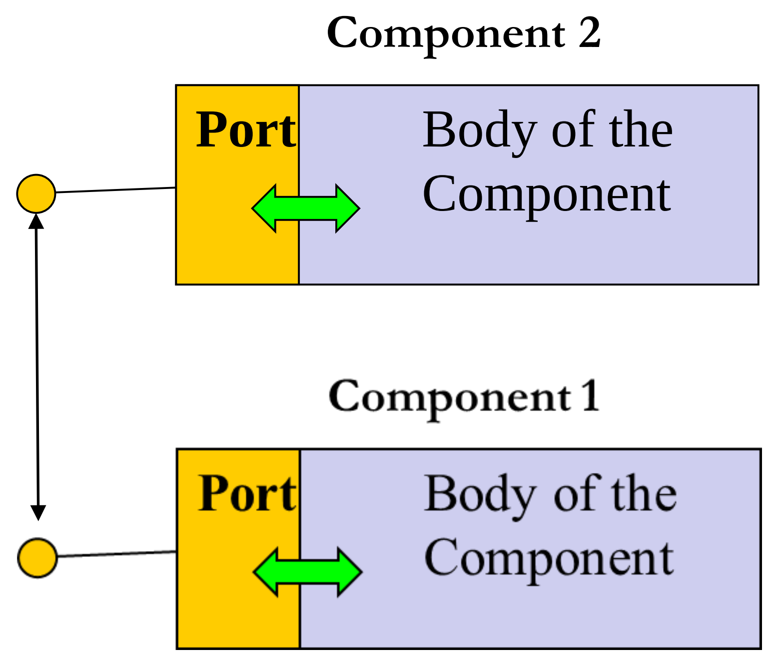
```
COMPONENT motorDC

PORTS
      IN  Elec      AL
      IN  Mech_rot eje
DATA
      REAL J = 2          "momento de inercia"
      REAL K = 3          "constante de par"
      REAL f = 0.01       "friccion"
      REAL R = 0.1        "resistencia"
      REAL Ke = 0.5

DECLS
      REAL T              “par”
      REAL w              “velocidad”

CONTINUOUS
       J*w'= K*AL.i - f*w – T
       AL.v = R*AL.i + Ke*w
       T = eje.T
       w = eje.omega

END COMPONENT
```

Electrical and mechanical ports have been defined…

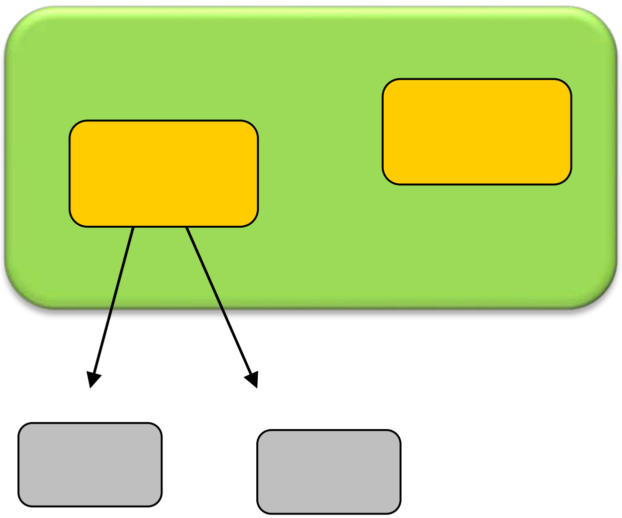
Model

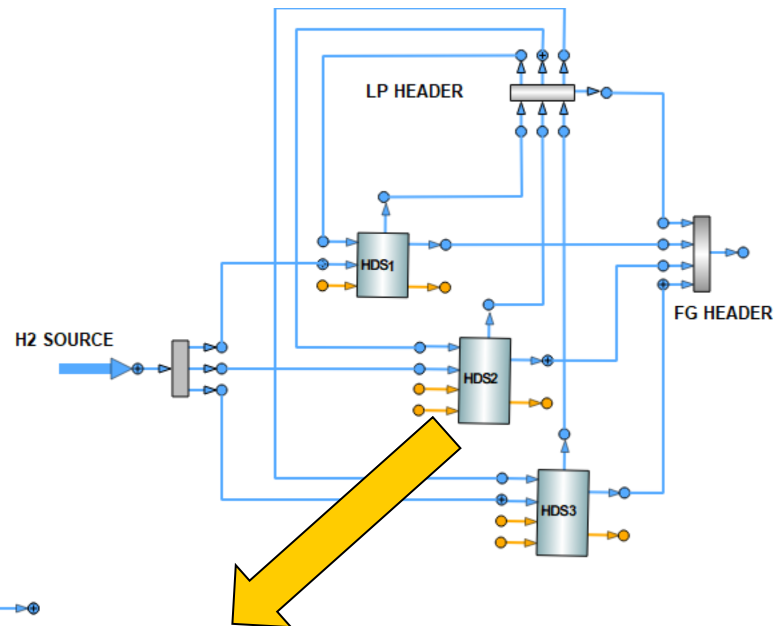
Component

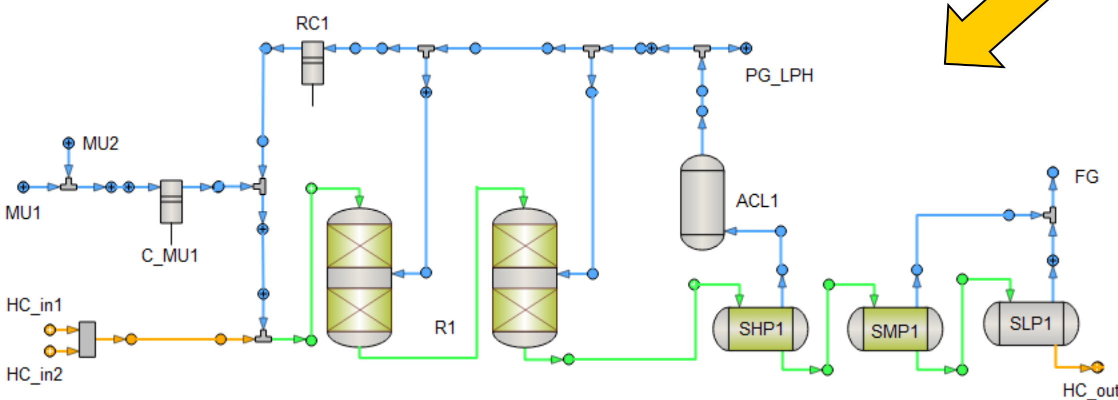
```
PORT Elec                "Electrical pin"
    EQUAL REAL v        "Potential (V)"
    SUM   REAL i         "Current (amp) "
END PORT
```



Hierarchical models







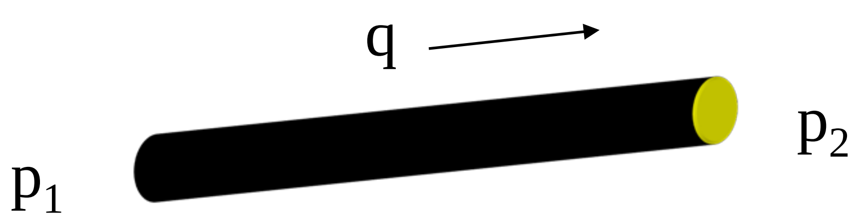
Modular modelling

Block oriented languages, do not allow true modular modelling, because they impose the computational causality at the model description stage

Modelling languages:

- They were developed to facilitate model reuse
- They do not have fix computational causality
- DYMOLA, GPROMS, MODELICA, OMOLA, ECOSIMPRO, ABACUS, JACOBIAN, ASPEN DYNAMICS…

Code to be executed depends on the aims and boundaries of the problem



If p_1 and p_2 are given:

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

If p_1 and q are given:

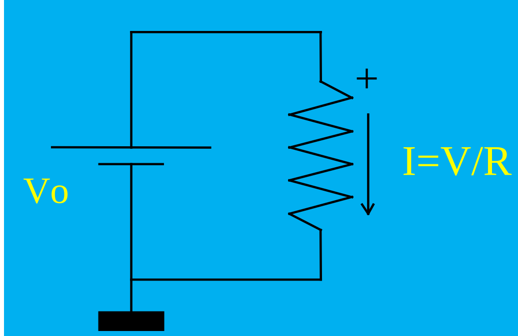
$$p_2 = p_1 - \frac{q^2}{k}$$

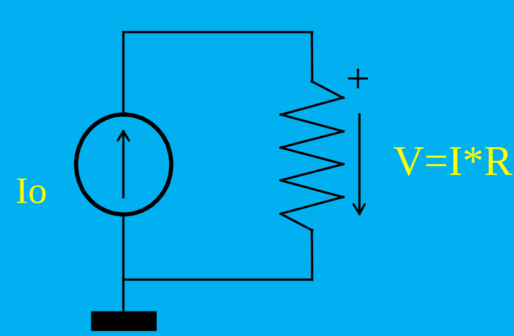
Aim: To have a description of the model of a component independent from its use in a specific case.

Computational Causality

Different to the equation ordering

Example: Two different implementations required for the resistor





Current is computed from the equation I = V/R

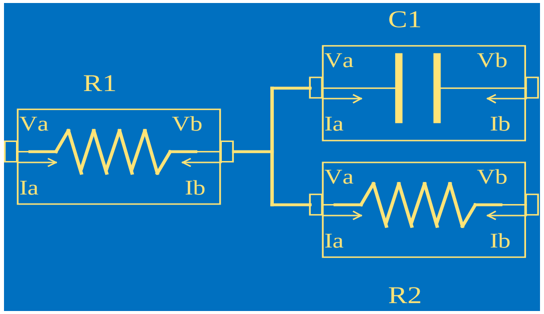
Voltage is computed from the equation V = IR

Computational causality assignment: Which equation should be used to compute every unknown variable? Modelling languages perform the assignment analysing the whole set of model equations as a function of the known boundaries.

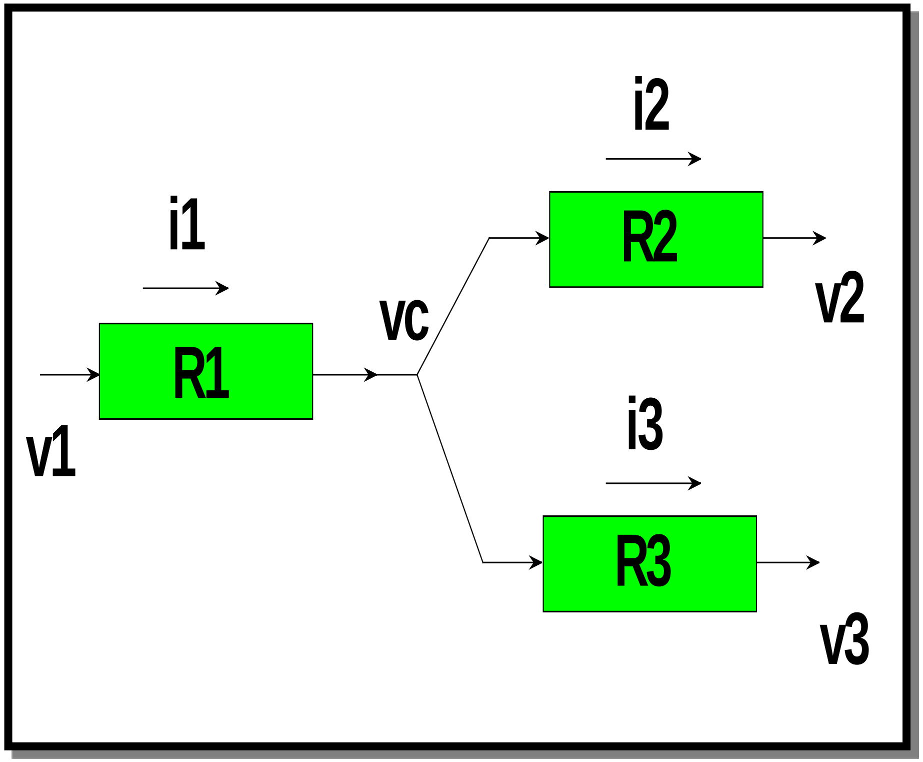
$$V = I\,R$$

Modelling Languages

- A model of a system is composed using a high level description, linking pre-defined modules representing sub-systems.
- Each module contains the mathematical description of a sub-system
- Each module is linked to others through an interface or port, in the same way as in the physical world.
- BUT, the mathematical model of the system is generated later on, manipulating the whole set of equations as a function of the chosen systems boundaries.



Modelling Languages




```
i1 = i2 + i3
v1 - vc = i1 * R1
vc - v2 = i2 * R2
vc - v3 = i3 * R3
```

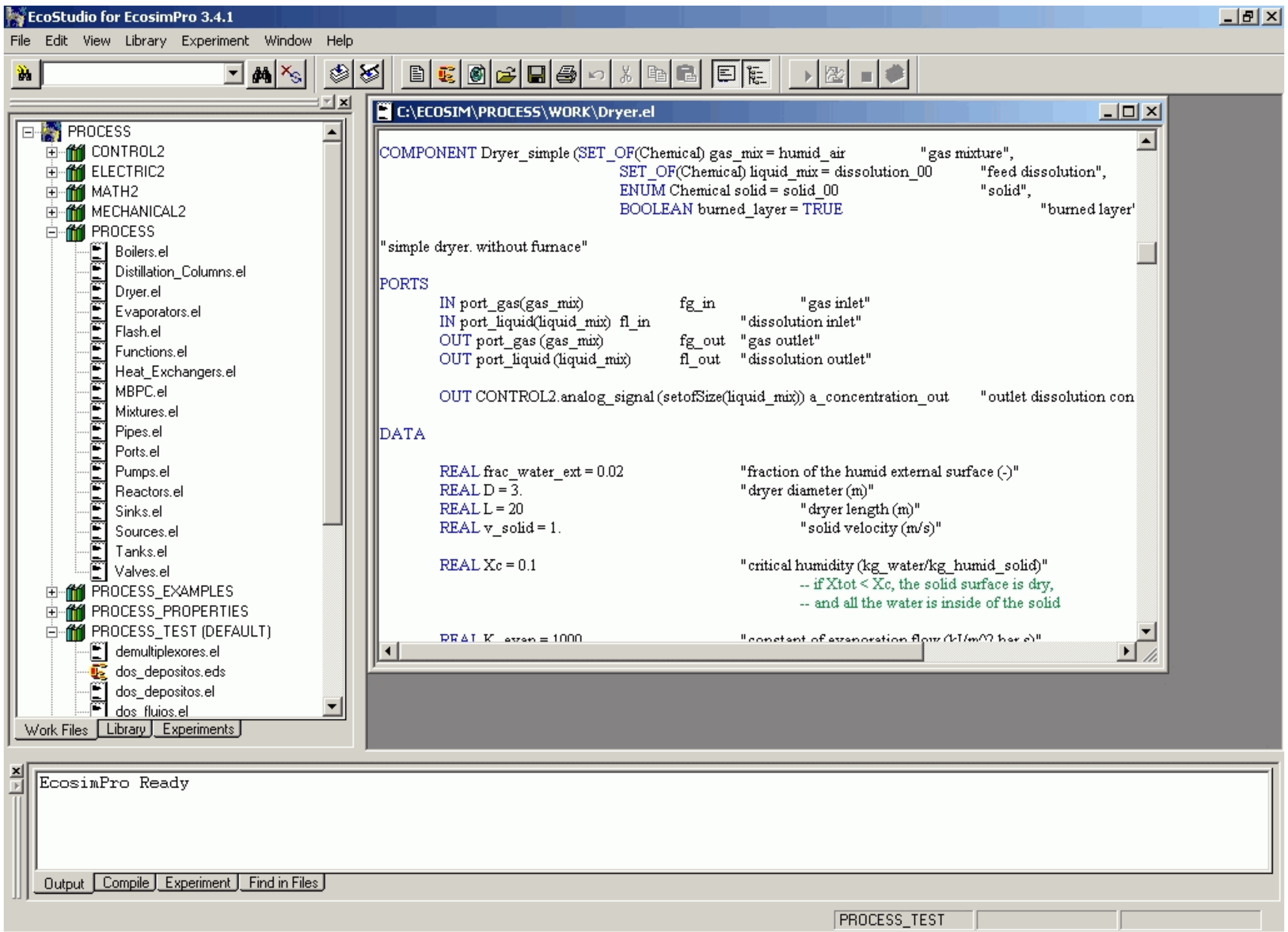
Analysis of the whole set of equations

Model and code generation

EcosimPro

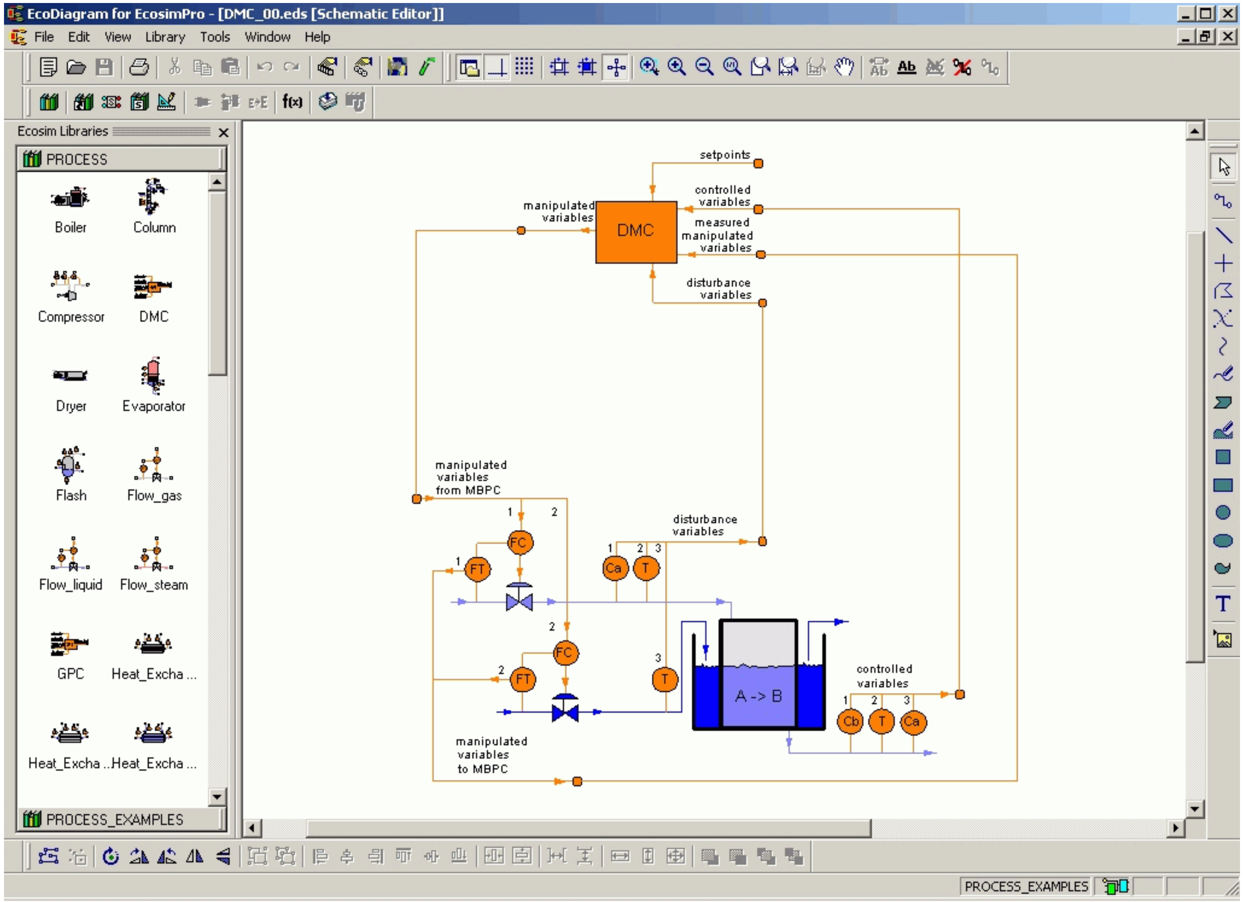
- First version 1992, Unix, ESA
- First version under Windows: 1999
- Object oriented tool
- Support continuous, discrete and discrete event processes
- Models are built by textual description of from graphical libraries.
- Provides a software development environment
- Open code, C++, ActiveX, OPC, FMI,…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

EcosimPro



EcoStudio for EcosimPro 3.4.1
File Edit View Library Experiment Window Help
C:\ECOSIM\PROCESS\WORK\Dryer.el
COMPONENT Dryer_simple (SET_OF(Chemical) gas_mix = humid_air "gas mixture",
SET_OF(Chemical) liquid_mix = dissolution_00 "feed dissolution",
ENUM Chemical solid = solid_00 "solid",
BOOLEAN burned_layer = TRUE "burned layer"
"simple dryer. without furnace"
PORTS
IN port_gas(gas_mix) fg_in "gas inlet"
IN port_liquid(liquid_mix) fl_in "dissolution inlet"
OUT port_gas (gas_mix) fg_out "gas outlet"
OUT port_liquid (liquid_mix) fl_out "dissolution outlet"
OUT CONTROL2.analog_signal (setofSize(liquid_mix)) a_concentration_out "outlet dissolution con
DATA
REAL frac_water_ext = 0.02 "fraction of the humid external surface (-)"
REAL D = 3. "dryer diameter (m)"
REAL L = 20 "dryer length (m)"
REAL v_solid = 1. "solid velocity (m/s)"
REAL Xc = 0.1 "critical humidity (kg_water/kg_humid_solid)"
-- if Xtot < Xc, the solid surface is dry,
-- and all the water is inside of the solid
PROCESS
CONTROL2
ELECTRIC2
MATH2
MECHANICAL2
PROCESS
Boilers.el
Distillation_Columns.el
Dryer.el
Evaporators.el
Flash.el
Functions.el
Heat_Exchangers.el
MBPC.el
Mixtures.el
Pipes.el
Ports.el
Pumps.el
Reactors.el
Sinks.el
Sources.el
Tanks.el
Valves.el
PROCESS_EXAMPLES
PROCESS_PROPERTIES
PROCESS_TEST (DEFAULT)
demultiplexores.el
dos_depositos.eds
dos_depositos.el
Work Files
Library
Experiments
EcosimPro Ready
Output
Compile
Experiment
Find in Files
PROCESS_TEST

Graphical environment



Modelling steps

Library of non causal models

Model built linking components

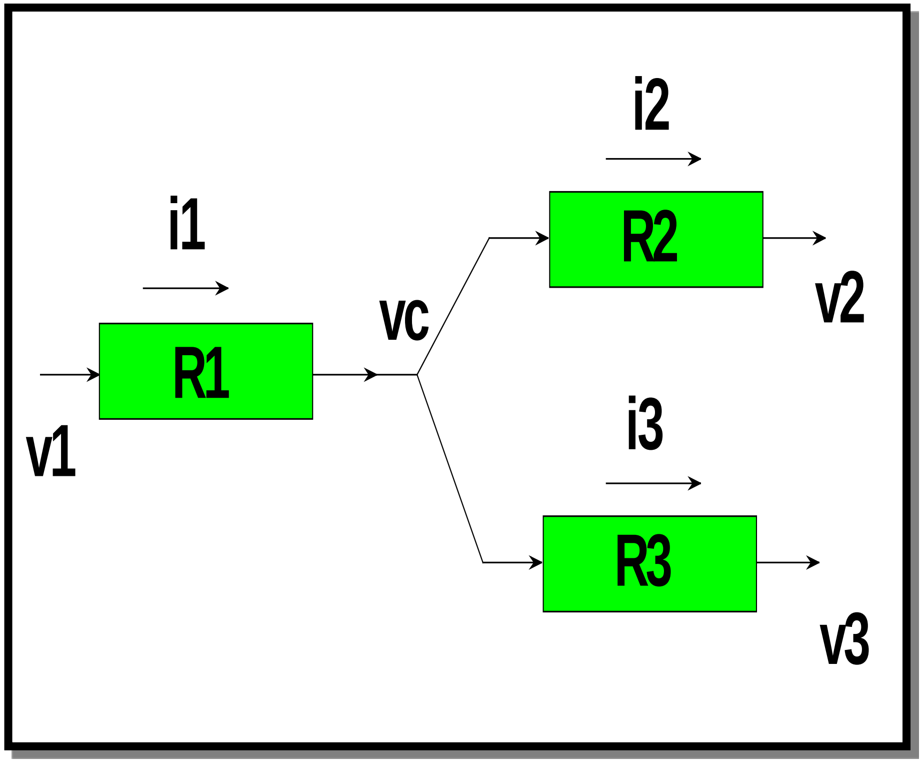
Analysis and computational causality of the whole set of equations (Partition)

Model with assigned computational causality

Experiment definition and Generation of C++ code

Simulation Code execution

Modelling Languages




```
i1 = i2 + i3
v1 - vc = i1 * R1
vc - v2 = i2 * R2
vc - v3 = i3 * R3
```

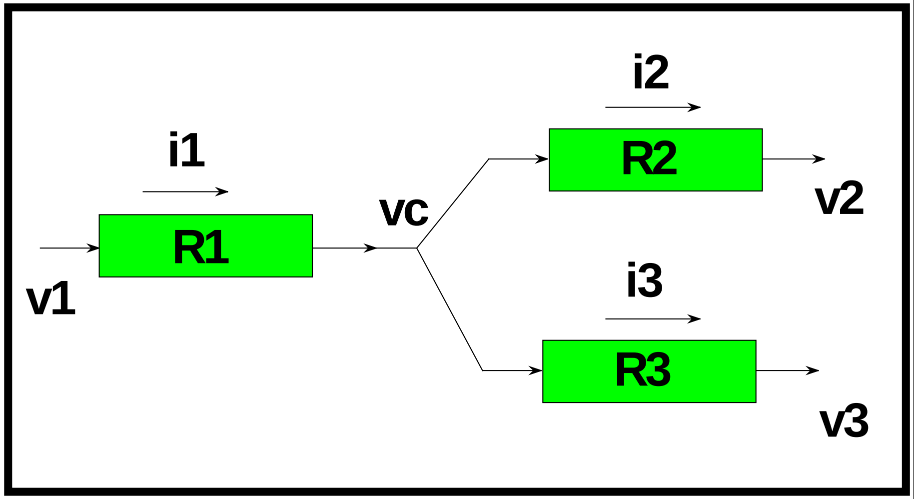
Analysis

Model analysis and assignment of computational causality (Partition generation)

1. Specify the boundary conditions
2. Is it feasible to solve the problem with the specified boundaries? (Detection of structural singularities, Maximum Transversal Algorithm)
    1. Inadequate Boundaries
    2. High index models
3. Specify the equation that will be used to compute every variable and stablish the order in which the equations will be used (BLT Algorithm)
    1. Whenever possible, work out every variable symbolically
    2. Identify the possible algebraic loops (Select tearing variables)
4. The partition is finished and ordered model equations are generated ready to be solved.

MaximumTransversal Algorithm

The number of required boundary conditions is determined as the difference between the number of variables and the number of equations. Once a set of boundary variables is proposed by the user, its validity is cheked with the Maximum Transversal algorithm



3 boundary conditions

7 unkowns:

v1, v2, v3, vc, i1, i2, i3

3 data: R1, R2, R3

4 equations

```
i1 = i2 + i3
v1 - vc = i1 * R1
vc - v2 = i2 * R2
vc - v3 = i3 * R3
```

MaximumTransversal Algorithm

Is the system with the selected boundary variables structurally correct?
A necessary condition for a model to be mathematically correct is the existence of a one to one correspondence between equations and variables.

Mathematically correct system	System with a structural singularity

Ecuaciones
Variables

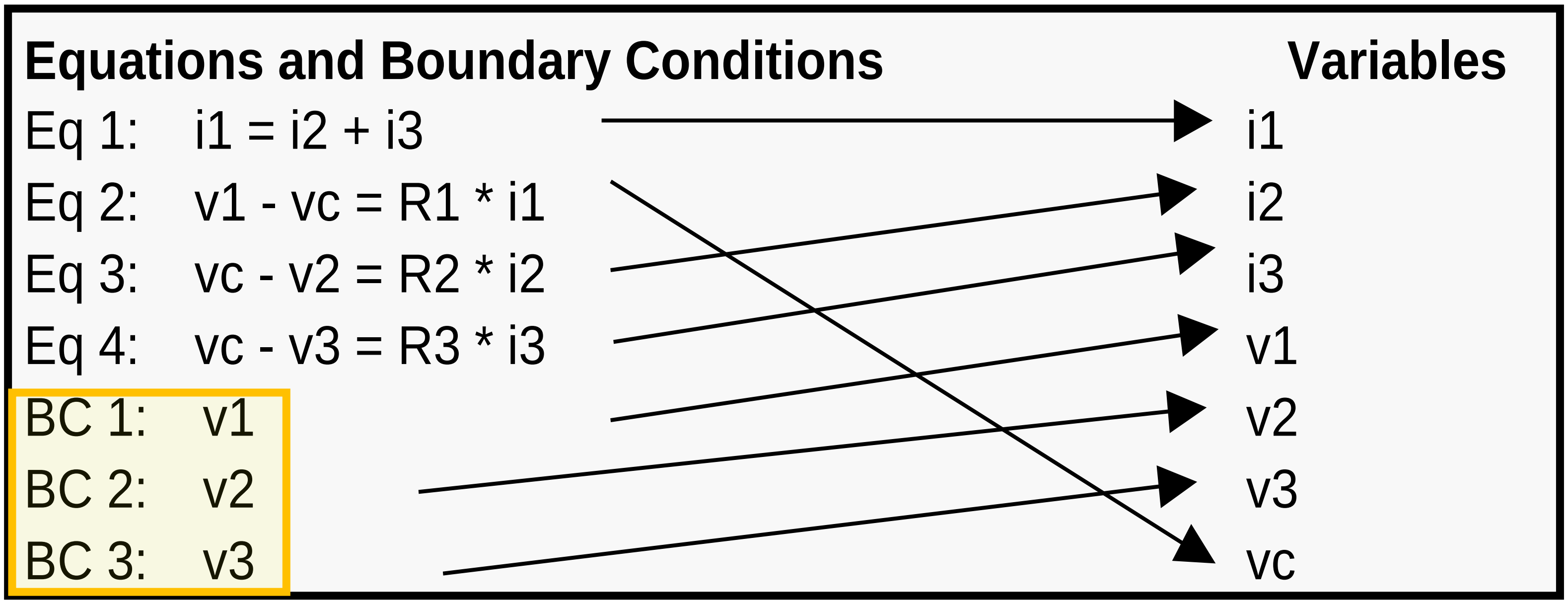
```
f1(x1)= 0   -----------> x1
f2(x1,x2,x3)= 0 -----> x2
f3(x1,x3)= 0 ---------> x3
```

Ecuaciones
Variables

```
f1(x1)= 0   -----------> x1
f2(x1,x2,x3)= 0 -----> x2
f3(x1)= 0   ---------> ??
```

MaximumTransversal Algorithm

Example: Valid boundary variables: v1,v2,v3



Maximum Transversal Algorithm

Example: Valid Boundary variables: vc,v2,v3

Equation		Variable
I1 = 12 + i3	→	i1
V1 – vc = R1*i1	→	v1
V2 – vc = R2*i2	→	i2
V3 – vc = R3*i3	→	i3
Vc	→	vc
V2	→	v2
V3	→	v3

But the simulation corresponds to a different problem

MaximumTransversal Algorithm

Example: Wrong boundary variables : i1,i2,i3

Equations and Boundary Conditions		**Variables**
Eq 1: $i1 = i2 + i3$	????	i1
Eq 2: $v1 - vc = R1 * i1$		i2
Eq 3: $vc - v2 = R2 * i2$		i3
Eq 4: $vc - v3 = R3 * i3$		v1
BC 1: i1		v2
BC 2: i2		v3
BC 3: i3		vc

MaximumTransversal Algorithm

In order to analyse the goodness of the selected boundary variables, a matrix is formed with two entries: the rows represent the equations and boundary variables, while the columns contain the model variables. This incidence matrix has a one in a element if the variable considered appears in the corresponding equation and has a zero otherwise.

$$
\begin{array}{l c}
 & \begin{array}{ccccccc} \mathrm{i1} & \mathrm{i2} & \mathrm{i3} & \mathrm{vc} & \mathrm{v1} & \mathrm{v2} & \mathrm{v3} \end{array} \\
\begin{array}{l}
\mathrm{i1} = \mathrm{i2} + \mathrm{i3} \\
\mathrm{v1} - \mathrm{vc} = \mathrm{i1R1} \\
\mathrm{vc} - \mathrm{v2} = \mathrm{i2}\ \mathrm{R2} \\
\mathrm{vc} - \mathrm{v3} = \mathrm{i3}\ \mathrm{R3} \\
\mathrm{v1} \\
\mathrm{v2} \\
\mathrm{v3}
\end{array}
&
\begin{bmatrix}
\times & \times & \times & & & & \\
\times & & & \times & \times & & \\
 & \times & & \times & & \times & \\
 & & \times & \times & & & \mathrm{x} \\
 & & & & \times & & \\
 & & & & & \times & \\
 & & & & & & \times
\end{bmatrix}
\end{array}
$$

$$
\begin{array}{l c}
 & \begin{array}{ccccccc} i1 & vc & i2 & i3 & v1 & v2 & v3 \end{array} \\
\begin{array}{l}
i1 = i2 + i3 \\
v1 - vc = i1\,R1 \\
vc - v2 = i2\ R2 \\
vc - v3 = i2\ R3 \\
v1 \\
v2 \\
v3
\end{array}
&
\begin{bmatrix}
\times & & \times & \times & & & \\
\times & \times & & & & \times & \\
 & \times & \times & & & \times & \\
 & \times & & \times & & & \times \\
 & & & & \times & & \\
 & & & & & \times & \\
 & & & & & & \times
\end{bmatrix}
\end{array}
$$

The Maximum transversal algorithm interchanges the matrix columns until all elements in the diagonal are ones. Then the problem is structurally solvable.

MaximumTransversal Algorithm

- ✓ Which model variables are included in the analysis performed by the Maximum Transversal algorithm?

$$\boxed{x' = dx / dt = f(x, t)}$$

- ✓ State variables x (which appear under the derivative sign) are considered known variables in the analysis because an initial value has to be assigned to them and, consequently, they are not included in the matrix of the Maximum Transversal.
- ✓ Derivatives of the state variables x’ are considered as unknown variables that must be evaluated for the integration of the system and, consequently, are in cluded in the matrix.

MaximumTransversal Algorithm

- The boundary conditions are selected freely by the user, but it is possible to suggest him a coherent set or check the user selection.
- When suggesting a set of boundary variables, a first choice refers to the variables assigned to unconnected ports, after checking that they satisfy the maximum transversal algorithm.
- If this choice fails, another set is selected iterating on the remaining variables.

Model analysis

- Have the model equations the adequate mathematical format to be solved?
- The Maximum Transversal algorithm fails when a high index problem is present.

	X_1'	X_2'
F1	x	
F2		x
g	?	?

$$\frac{dx_1}{dt} = f_1(x_1, x_2, u, t)$$

$$\frac{dx_2}{dt} = f_2(x_1, x_2, u, t)$$

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

u boundary variable

DAEs / ODEs Models

- A set of Ordinary Differential Equations, where the derivatives of the state variables appear explicitly, as functions of the states and of known functions of time is denoted as ODE.

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

- When the derivatives do not appear as explicit functions, the system of equations is called a set of DAEs, Differential Algebraic Equations. This includes implicit differential equations and coupled sets of differential and algebraic equations. In general:

$$F(\dot{x}, x, u) = 0$$

Index Problems

- Sometimes a set of DAEs can be converted to a set of ODEs managing the equations, nevertheless, if the matrix $\partial F/\partial \dot{x}$ is singular, the transformation is not possible unless some of the equations are differentiated with respect to time.

$$F(\dot{x}, x, u) = 0 \quad \Longrightarrow \quad \frac{dx}{dt} = f(x, u)$$

- The index of a DAE is the number of times needed to differentiate the DAEs to get a system of ODEs.
- A differential index of 1 is called low index, while it is called High index if it is 2 or larger.
- In systems with index 1 or larger, the maximum transversal algorithm fails in finding a feasible set of boundary conditions.

Index problems

- Index problems appear many times associated to the formulation of a DAE model where the state variables cannot be computed freely, but are constrained by some bond equations.

$$\frac{dx_1}{dt} = f_1(x_1, x_2, u, t)$$

$$\frac{dx_2}{dt} = f_2(x_1, x_2, u, t)$$

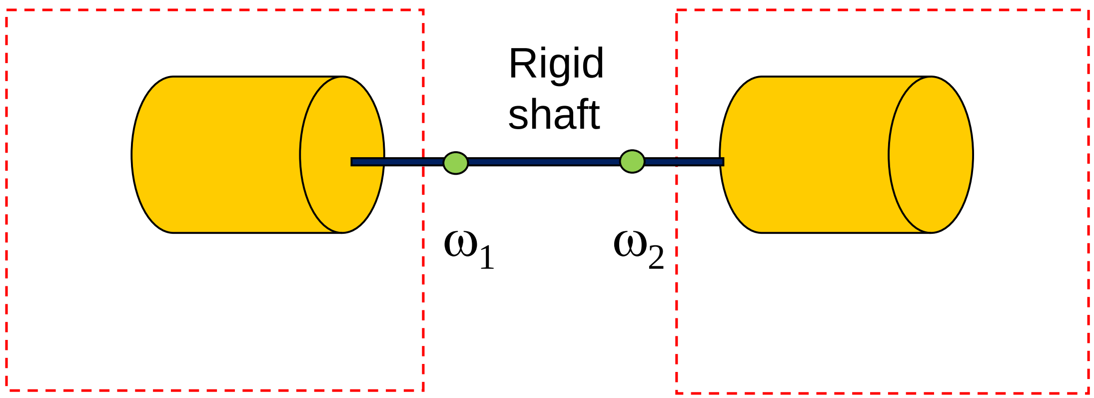
$$g(x_1, x_2, u) = 0$$

In the bond equations, all variables are known

- Some integration methods may not consider these bonds and, consequently, they fail if applied to a high index problem.
- In particular, we cannot assign initial values to the states freely, as they must satisfy the bond equations.

Index problems: Examples

High index problems can be generated when linking together two components of a library because of the bond equations added by the ports:



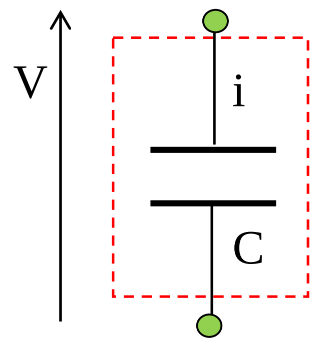
$$J_1 \frac{d\omega_1}{dt} = \ldots + T_1 + T_2 + \ldots \qquad J_2 \frac{d\omega_2}{dt} = \ldots + T_1 + T_2 + \ldots$$

$$\boxed{\omega_1 = \omega_2}$$

Equation generated by the mechanical port

Index problems: Examples

Index problems can be generated when linking together two components of a library because of the bond equations added by the ports:

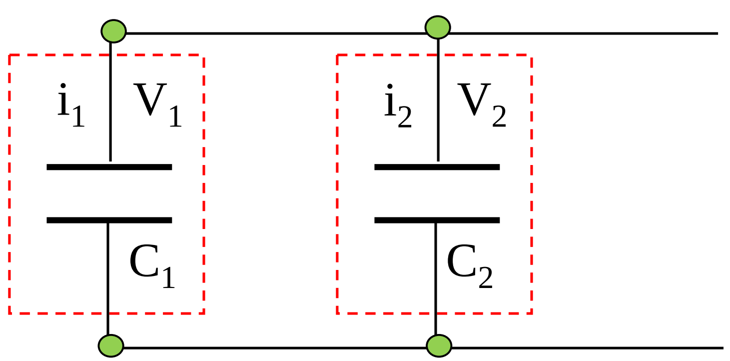


$$C\frac{dV}{dt} = i$$

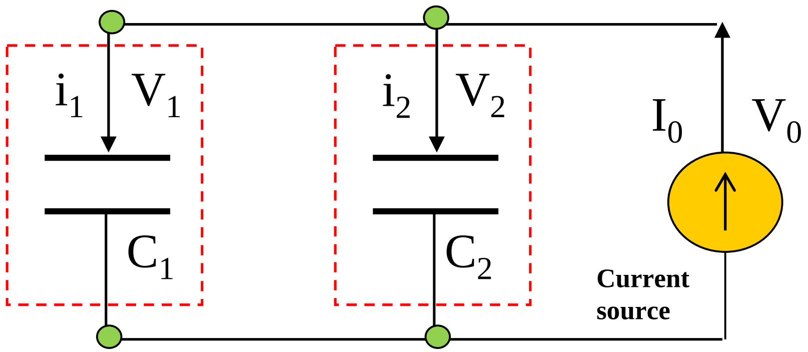
$$C_1\frac{dV_1}{dt} = i_1$$

$$C_2\frac{dV_2}{dt} = i_2$$

$$\boxed{V_1 = V_2}$$



Index problems: Examples



V_1, V_2 are state variables as they appear under the derivative sign. I_0 is chosen as boundary variable

Unknown variables: V_1', V_2', i_1, i_2, V_0
Known variables: V_1, V_2, I_0

$$C_1 V_1' = i_1$$
$$C_2 V_2' = i_2$$
$$V_1 = V_2 \quad ??$$
$$I_0 = i_1 + i_2$$
$$V_0 = V_2$$

Only 4 equations contain the 5 unknown variables. This structure implies that there is no solution to the assignment problem with the maximum transversal algorithm

5 variables and 5 equations

but the bond equation $V_1 = V_2$ creates a structural singularity

High Index problems: Examples

- Sometimes high index problems appear due to the formulation of the problem, that does not follows the physical causality but corresponds to other problems like, e.g. control
- Which is the force that must be applied to a particle in order to move it according to a certain pre-specified trajectory?

$$F = m \frac{d^2 x}{dt^2}$$

$$x(t) = e^{-t/10} \sin(t)$$

There is a boundary condition specified on a state variable

High Index problems

- The Maximum Transversal algorithm fails when a high index problem is present. Example:

3 Equations

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

Known variables

v & **x** state variables

m data

3 Unknown variables

F, v', x'

MAXIMUM TRANSVERSAL

$$F = m * v' \longrightarrow v'$$

$$x' = v \longrightarrow x'$$

$$x = \exp(-TIME/10) * \sin(TIME)\ ?? \quad F$$

Three variables that appear only in two equations. The last one is useless for estimating F

Pantelides algorithm

- The Pantelides algorithm is used to transform high index problems into an equivalent lower index one.
- The algorithm adds new equations to the model obtained by differentiation of the ones that create the structural singularity (the bond equations), facilitating the application of the maximum transversal algorithm.

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

$$g(x_1, x_2, u) = 0 \quad \Rightarrow \quad \frac{dg(x_1, x_2, u)}{dt} = 0$$

- As new equations are added, one should either incorporate more variables, or substitute the bond equations by its differentiate form to balance the number of equations and variables.
- The procedure is repeated until no structural singular set is found

Pantelides algorithm

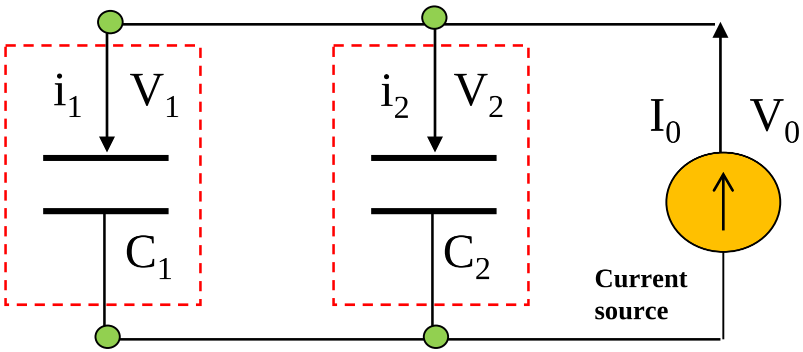
- One option to balance equations and variables is to substitute the bond equations by its differentiated form

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

$$\cancel{g(x_1, x_2, u) = 0} \quad \Rightarrow \quad \frac{d\,g(x_1, x_2, u)}{dt} = 0$$

- Another option is not replacing the bond equations, but adding some states as new variables. As the initial values of the state equations cannot be chosen arbitrarily, some state variables involved in the bonds are not computed by integration of the corresponding differential equation, but from the bond equations. This implies that these state variables can be considered as unknown and added to the list for the analysis of the maximum transversal algorithm.

Example



V_1 and V_2 are state variables

I_0 is chosen as boundary variable

Unknown variables: V_1', V_2', i_1, i_2, V_0

Known variables: V_1, V_2, I_0

5 variables and 5 equations

$$C_1 V_1' = i_1$$

$$C_2 V_2' = i_2$$

$$V_1' = V_2'$$

$$I_0 = i_1 + i_2$$

$$V_0 = V_2$$

Now there are 5 equations containing 5 unknown variables and the maximum transversal algorithm can be applied.

But coherent initialization is required or critical information can be lost about the initial values

V' = d \dt

Index one problem, as the bond equation $V_1 = V_2$ has been differentiated only once

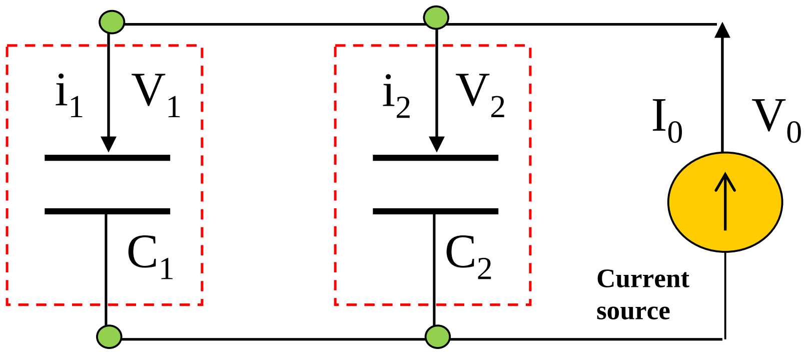
Example

	V_1'	V_2'	i_1	i_2	V_0
$C_1 V_1' = i_1$	**x**		**x**		
$C_2 V_2' = i_2$		x		x	
$V_1' = V_2'$	x	x			
$I_0 = i_1 + i_2$			x	x	
$V_0 = V_2$					x

V_1'	i_2	V_2'	i_1	V_0
x			**x**	
	x	x		
x		x		
	x		x	
				x

This implies that the problem is now structurally solvable. A different problem is finding the right assignment and order of calculus between variables and equations

Example



V_2 is a state variable, but it will be considered as a variable as it will not be computed from integration of V_2', but from $V_2 = V_1$

I_0 is chosen as boundary variable

Unknown variables: V_1', V_2', i_1, i_2, V_0, V_2

Known variables: V_1, I_0

6 variables and 6 equations

$$C_1 V_1' = i_1$$
$$C_2 V_2' = i_2$$
$$V_1 = V_2$$
$$V_1' = V_2'$$
$$I_0 = i_1 + i_2$$
$$V_0 = V_2$$

Now there are 6 equations containing 6 unknown variables and the maximum transversal algorithm can be applied

Index one problem, as the bond equation $V_1 = V_2$ has been differentiated only once

Pantelides algorithm

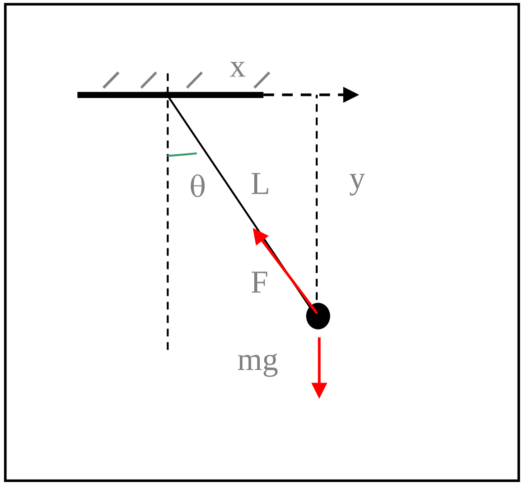
	V_1'	V_2'	i_1	i_2	V_0	V_2
$C_1V_1' = i_1$	x		x			
$C_2V_2' = i_2$		x		x		
$V_1 = V_2$						x
$I_0 = i_1 + i_2$			x	x		
$V_0 = V_2$					x	x
$V_1' = V_2'$	x	x				

$\Rightarrow$

V_1'	i_2	V_2	i_1	V_0	V_2'
x			**x**		
	x				x
		x			
	x		x		
		x		x	
x					x

This implies that the problem is now structurally solvable. A different problem is finding the right assignment and order of calcululus between variables and equations

Pendulum (Index 2 problem)



$$m\frac{d^2x}{dt^2} = -F\frac{x}{L} \qquad \frac{dx}{dt} = v_x$$

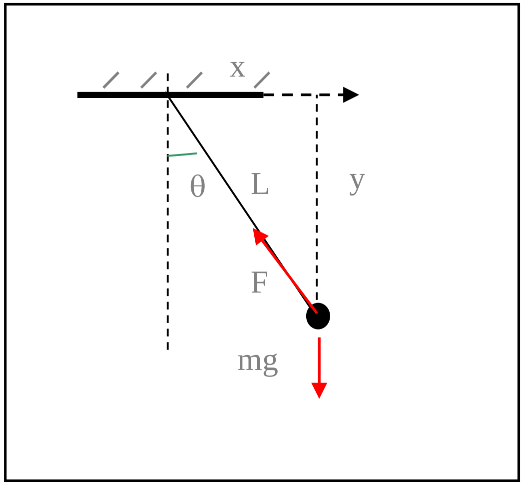
$$m\frac{d^2y}{dt^2} = -F\frac{y}{L} - mg \qquad \frac{dy}{dt} = v_y$$

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

There are 4 useful equations and 4 unknowns x’,y’, v_x’, v_y’ but as we cannot initialize arbitrarily the four states, the bond equation is differentiated twice to find equations that provide the value of two of them instead of using integration of the corresponding differential equations.

The structural singularity is created by inadequate modelling: using cylindrical coordinate, the problem can be described with a single variable θ without bonds

Pendulum (Index 2 problem)



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

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

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

$$\boxed{\begin{aligned} & x^2 + y^2 = L^2 \quad \Rightarrow 2xv_x + 2yv_y = 0 \quad \Rightarrow \\ & \Rightarrow 2x\frac{dv_x}{dt} + 2v_x^2 + 2y\frac{dv_y}{dt} + 2v_y^2 = 0 \end{aligned}}$$

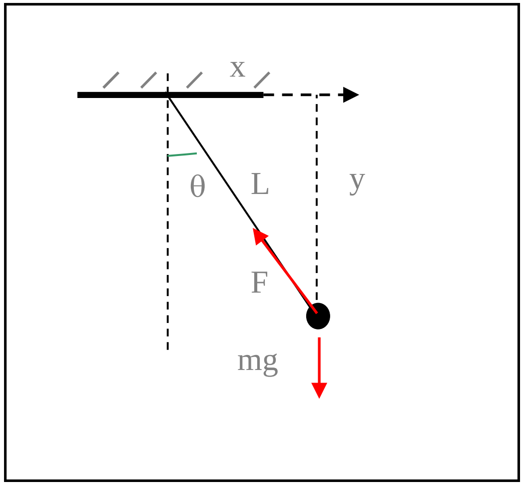
Obtained by differentiation

It is possible to find a subset without structural singularity and, then, compute the other variables from the bond and differentiated equations

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

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

Pendulum (another choice of state variables)



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

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

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

If we select instead y, v_y , as state variables, it is possible to have divisions by zero…

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

1 Solving first the subset of equations:

$$m\frac{dv_y}{dt} = -F\frac{y}{L} - mg \qquad \frac{dy}{dt} = v_y \qquad x = \sqrt{L^2 - y^2}$$

2 Then,computing the remaining variables from:

$$v_x = -\frac{yv_y}{x} \qquad \frac{dv_x}{dt} = \frac{-1}{x}\left[-y\frac{Fy}{mL} - g + v_x^2 + v_y^2\right]$$

Index problems. Boundary conditions

- When selecting the boundary variables, care must be taken to avoid generating undesired index problems.
- If a state variable is selected as a boundary, a bond is automatically created. But as the Pantelides algorithm requires computing derivatives of the bond equations, it needs a explicit form of the time dependency of the variable, which is not given at the time of partition definition. Because of this, state variables are not allowed as boundary variables.
- If one wants to impose a certain time evolution to a state variable, it must add the corresponding equation x = f(t) as part of the model, so that its explicit form is known at partition generation time.

High index example

$$F = m\frac{d^2x}{dt^2}$$

$$x(t) = e^{-t/10}\sin(t)$$

```
EQUATIONS
F = m * v'
x'= v
x = sin(TIME)
x' = cos(TIME)
v' = -sin(TIME))
```

```
F = m * v'         →   F
x'= v              →   v
x = sin(TIME)      →   x
x' = cos(TIME)     →   x'
v' = -sin(TIME))   →   v'
```

There is a bond on the state variable x

The bond equation is differentiated twice, generating two equations that allow computing x' and v' from them, instead of by integration, avoiding the problems associated to the need of consistent initial conditions

Index 2 problem

Ordering of equations
BLT Algorithm

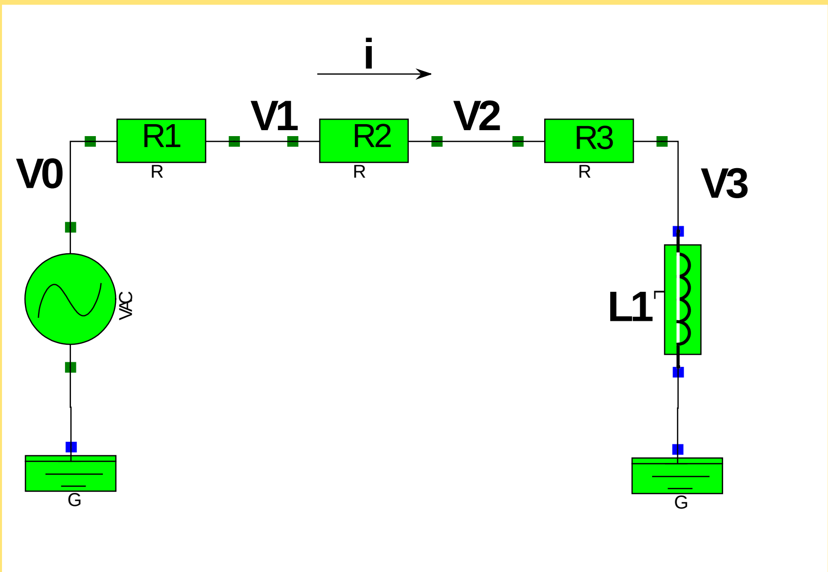
- ✓ Once we are sure there is no structural singularities in the model, the BLT (Block Lower Triangularization) algorithm can be used in order to find the right computational order of the system of equations. This algorithm operates with the incidence matrix, interchanging rows and colums until a lower triangular matrix is obtained.

$$
\begin{array}{c c}
 & \begin{array}{cccccccc} V_1 & V_2 & V_3 & V_4 & V_5 & V_6 & V_7 & V_8 \end{array} \\
\begin{array}{c} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \end{array} &
\begin{bmatrix}
x & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
x & x & 0 & 0 & 0 & 0 & 0 & 0 \\
x & x & x & 0 & 0 & 0 & 0 & 0 \\
x & x & x & x & 0 & 0 & 0 & 0 \\
x & x & x & x & x & 0 & 0 & 0 \\
x & x & x & x & x & x & 0 & 0 \\
x & x & x & x & x & x & x & 0 \\
x & x & x & x & x & x & x & x
\end{bmatrix}
\end{array}
$$

If this lower triangular matrix is found, then the system of equations is an explicit one, and V_1 can be computed form equation f_1 , V_2 from f_2 , V_3 from f_3 , …

Whenever possible, symbolic manipulation can be used to work out explicitely each variable from the corresponding equation

BLT Algorithm, example



V0 - sin(time) = 0

V0 - V1 - i * R1 = 0

V1 - V2 - i * R2 = 0

V2 - V3 - i * R3 = 0

V3 - L * i' = 0

	i'	V0	V1	V2	V3
V0 - sin(time)= 0	0	x	0	0	0
V0 - V1 - i * R1 = 0	0	x	x	0	0
V1 - V2 - i * R2 = 0	0	0	x	x	0
V2 - V3 - i * R3 = 0	0	0	0	x	x
V3 - L * i'	x	0	0	0	$\boxed{\text{X}}$

BLT →

V0	V1	V2	V3	i'
x	0	0	0	0
x	x	0	0	0
0	x	x	0	0
0	0	x	x	0
0	0	0	$\boxed{\text{x}}$	x

Symblic manipulation →

ORDERED EQUATIONS

V0 = sin(time)

V1 = V0 - i * R1

V2 = V1 - i * R2

V3 = V2 - i * R3

i' = V3 / L

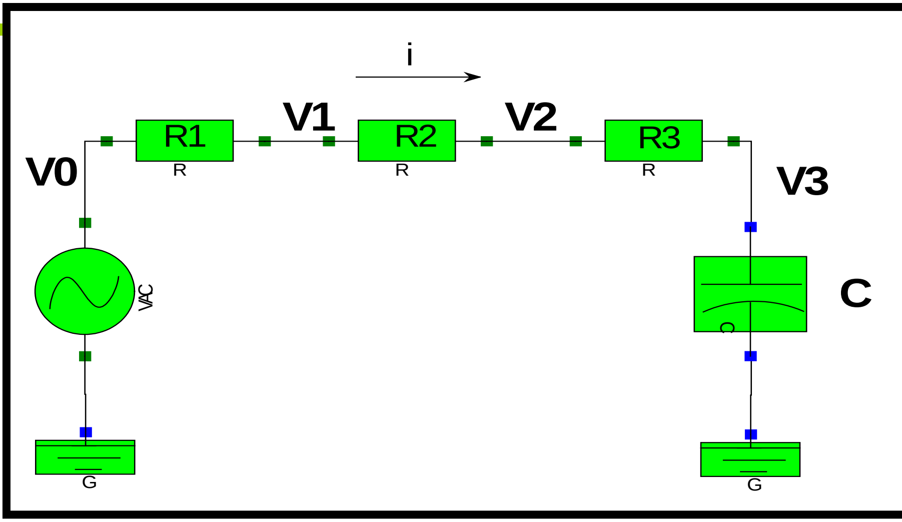
Ordering of equations
BLT Algorithm

- ✓ If a lower triangular matrix cannot be found, then, there are algebraic loops in the model.
- ✓ In this case, the BLT algorithm will find a block lower triangular matrix, with some square compact blocks A_{ii}

$$
\begin{bmatrix}
\mathbf{A}_{11} & \mathbf{0} & \mathbf{0} & \mathbf{0.} & . & . & \mathbf{0} \\
\mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{0} & \mathbf{0} & & & \mathbf{0} \\
\mathbf{A}_{31} & \mathbf{A}_{32} & \mathbf{A}_{33} & \mathbf{0} & & & . \\
\mathbf{A}_{41} & \mathbf{A}_{42} & \mathbf{A}_{43} & \mathbf{A}_{44} & \mathbf{0} & & . \\
. & & & & . & \mathbf{0} & . \\
. & & & & & . & \mathbf{0} \\
\mathbf{A}_{N1} & . & . & . & . & . & \mathbf{A}_{NN}
\end{bmatrix}
$$

Each block of size larger than 1, represents a subsystem of coupled equations that has to be solved jointly forming an algebraic loop.

BLT –Algebraic loops



V0 - sin(time) =0

V0 - V1 - i * R1 = 0

V1 - V2 - i * R2 = 0

V2 - V3 - i * R3 = 0

V3' - i / C = 0

	i	V0	V1	V2	V3'
V0 - sin(time)= 0	0	x	0	0	0
V0 - V1 - i * R1 = 0	x	x	x	0	0
V1 - V2 - i * R2 = 0	x	0	x	x	0
V2 - V3 - i * R3 = 0	x	0	0	x	x
V3' - i / C = 0	x	0	0	0	x

V0	V1	V2	i	V3'
x	0	0	0	0
x	x	0	x	0
0	x	x	x	0
0	0	x	x	0
0	0	0	x	x

ORDERED EQUATIONS

$V0 = \sin(time)$

$$\begin{bmatrix} -1 & 0 & -R1 \\ 1 & -1 & -R2 \\ 0 & 1 & -R3 \end{bmatrix} \begin{bmatrix} V1 \\ V2 \\ i \end{bmatrix} = \begin{bmatrix} -V0 \\ 0 \\ V3 \end{bmatrix}$$

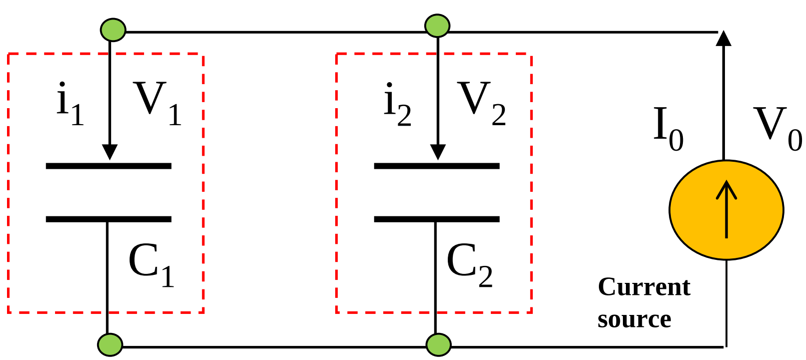
$V3' = i / C$

BLT example

	V_1'	i_2	V_2'	i_1	V_0
$C_1 V_1' = i_1$	**x**			**x**	
$C_2 V_2' = i_2$		x	x		
$V_1' = V_2'$	x		x		
$I_0 = i_1 + i_2$		x		x	
$V_0 = V_2$					x

	V_0	i_1	V_1'	i_2	V_2'
$V_0 = V_2$	**x**				
$C_1 V_1' = i_1$		x	x		
$I_0 = i_1 + i_2$		x		x	
$V_1' = V_2'$				x	x
$C_2 V_2' = i_2$				x	x

known:
V_1, V_2, I_0



Solved as a set of algebraic equations

Algebraic loops

- ✓ The BLT algorithm finds an ordered set of equations including possible algebraic loops (subsystems of coupled equations)
- ✓ In order to solve the algebraic loops:
  - If all equations of the block are linear, it is possible to work out explicitly the variables involved using a symbolic manipulator, or solve the loop with an efficient linear solver.
  - If the algebraic loop is non-linear, then the solution may require a non-linear solver, based on Newton-Raphson , besides the selection of the tearing variables.

BLT example

As system is linear, using symbolic manipulations:

	V_0	i_1	V_1‘	i_2	V_2’
$V_0 = V_2$	**x**				
$C_1 V_1' = i_1$		x	x		
$I_0 = i_1 + i_2$		x		x	
$V_1' = V_2'$				x	x
$C_2 V_2' = i_2$				x	x

$$V_0 = V_2$$

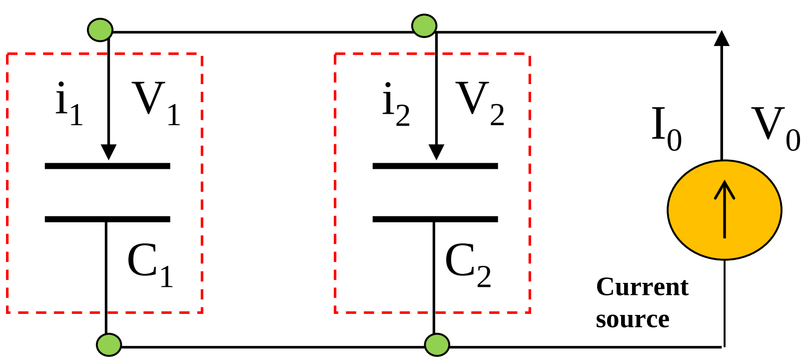
$$\left.\begin{array}{l} V_1' = V_2' \\ I_0 = i_1 + i_2 \end{array}\right\} C_1 V_2' + C_2 V_2' = I_0 \Rightarrow V_2'$$

$$V_1' = V_2'$$

$$i_2 = C_2 V_2'$$

$$i_1 = C_1 V_1'$$

known:
V_1, V_2, I_0



Solved as a set of algebraic equations

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.

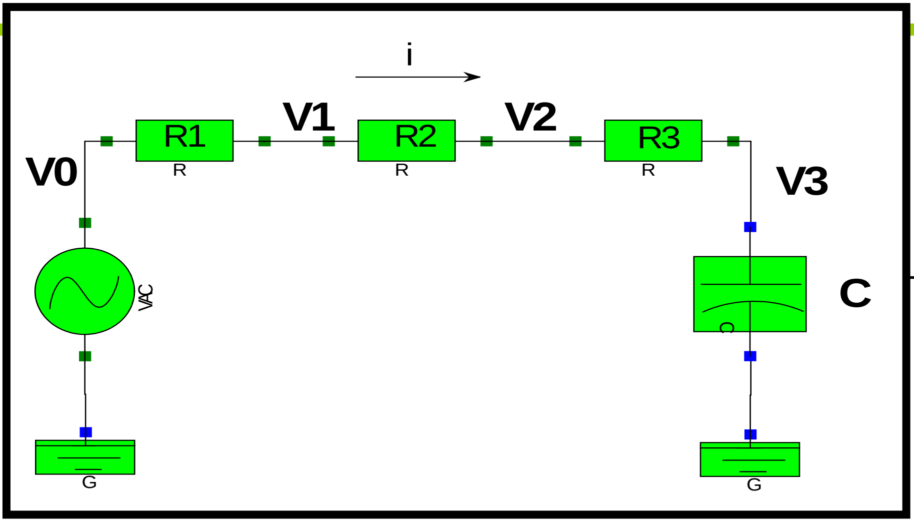
$F_1(x_1, x_2) = 0$
$F_2(x_1, x_2, x_3) = 0$
$F_3(x_1, x_2, x_3) = 0$

x_2 selected as tearing variable

$\Downarrow$

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

Loop Tearing



V0 - sin(time) =0

V0 - V1 - i**2 * R1 = 0

V1 - V2 - i**2 * R2 = 0

V2 - V3 - i**2 * R3 = 0

V3' - i / C = 0

	i	V0	V1	V2	V3'
V0 - sin(time)= 0	0	x	0	0	0
V0 - V1 - i * R1 = 0	x	x	x	0	0
V1 - V2 - i * R2 = 0	x	0	x	x	0
V2 - V3 - i * R3 = 0	x	0	0	x	x
V3' - i / C = 0	x	0	0	0	x

V0	V1	V2	i	V3'
x	0	0	0	0
x	x	0	x	0
0	x	x	x	0
0	0	x	x	0
0	0	0	x	x

ORDERED EQUATIONS

V0 = sin(time)

i *tearing variable*

V1 = V0 - i2 * R1**

V2 = V1 - i2 * R2**

F(i) = V2 - V3 - i2 * R3 = 0**

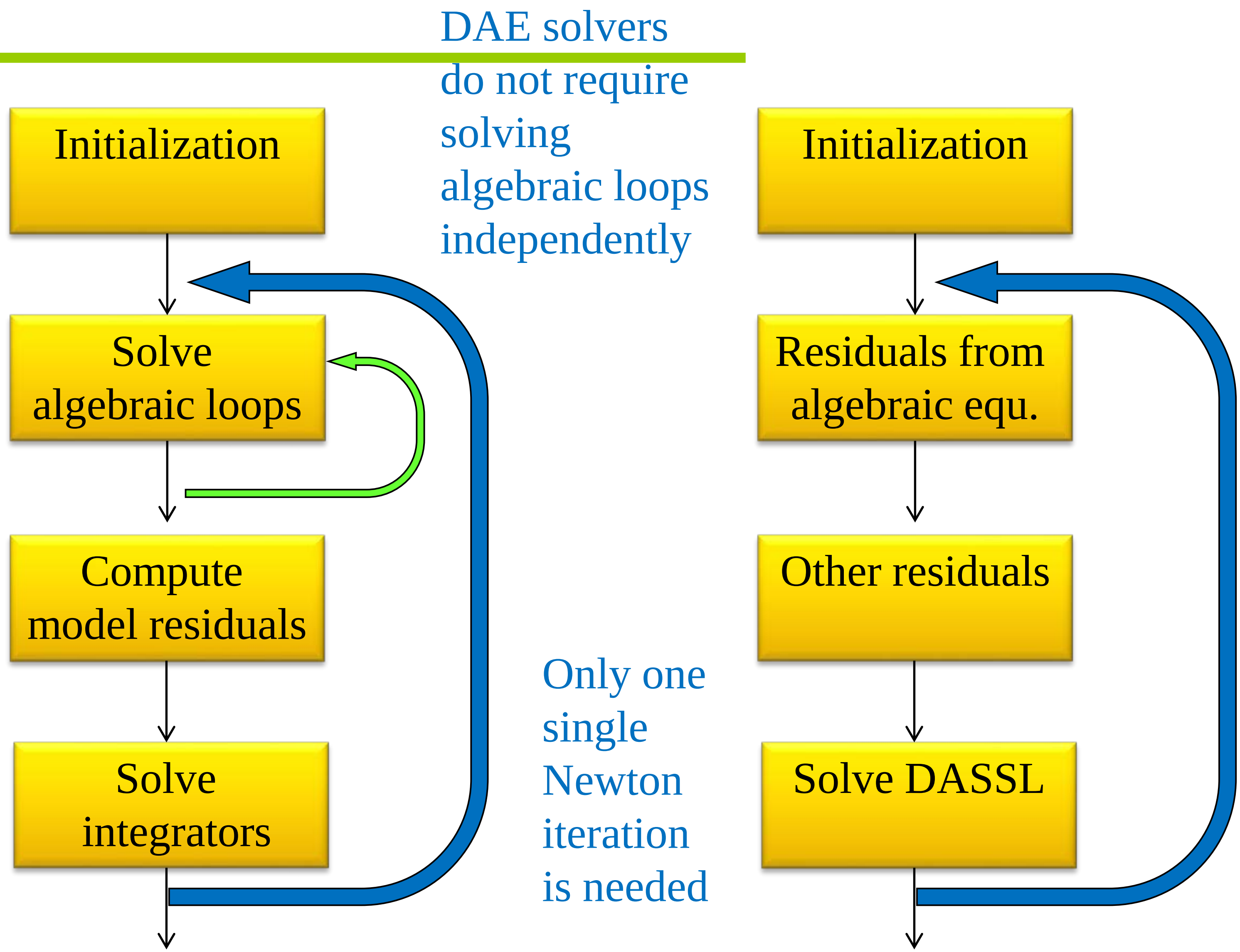
V3' = i / C

Residue equation

Loop Tearing

- Loop Tearing methods have some weakness:
- Tearing algorithms are based on heuristic rules
- There is no algorithm that provides the best choice among the different possible sets of tearing variables.
- As a consecuence, the user can select a better set of tearing variables if it is not satisfied with the selection made by the simulation environment

DAEs and algebraic loops



Overall steps

Model editor and error cheking (Compile)

↓

Selection of boundary variables and partition generation

↓

Specify experiment

↓

C++ Class

↓

Compiler + internal Libraries+ Calls to external software

↓

Run-time Executable code