EcosimPro Frequently Asked Questions (FAQs, UserManual §3.Simulation Guide, adapted)

Sommaire

1 General

Can EcosimPro be used purely as a solver of algebraic and differential equations?

It certainly can.
Users who do not wish to make complex simulation models can use EcosimPro purely as an equation solver.
All they have to do is encapsulate the equations in an isolated component, first declaring the parameters, data, and variables, then creating a partition, and then resolving the system.

What is the main advantage of EcosimPro compared to popular block-oriented simulation environments like Simulink?

The main advantage of EcosimPro is that

- It is not necessary to know the mathematical causality of the problem when equations are written into the component.
- So, the order in which equations are written is therefore not important for EcosimPro.

In clear: EcosimPro needs only equations, and not the how to solve the problem. One says that equations are treated as symbolic or non-causal equations. This is incredibly powerful for solving complex systems.

- This means that components are **universal**; they can consequently be put to any use.
- Software applications such as Simulink are less flexible: the order of the equations greatly restricts the performance of experiments that are only slightly different, while for EcosimPro it is totally flexible by essence.

Who are the potential users of EcosimPro?

EcosimPro is a software tool that can be widely used in many different fields of science and engineering:

- Simulation of continuous systems in sectors such as aeronautics and space, energy generation, chemical process, mechanical, electrical, fluids, etc.
- EcosimPro facilitates sophisticated studies of transients and steady states.
- Creation of component libraries in the different disciplines of physics and engineering.
- Resolution of differential-algebraic equation systems.
- In education it can be used to learn how to model and simulate physical systems.

What are the main advantages of object-oriented physical modelling?

Object-oriented system modelling has important advantages over other system modelling methods:

- Reuse of the modelling code through simple or multiple inheritance. This means that new models can be created by expanding on models that already exist, incorporating behaviour that is common to several models into a parent component that can be independently tested.
- Encapsulation of information. Modelling with objects, the modeller encapsulates in components the data and behaviour of physical models. This makes it easier to understand complex models and to follow their execution.
- Aggregation. Instances from other components can be aggregated into a single component. This makes it possible to successively create more sophisticated components based on basic components.

2 Continuous and Discrete Modelling

Can EcosimPro be used purely for discrete simulation?

Yes, EcosimPro has a very powerful event handling mechanism to cope with discrete simulation.

Can you write in time events with EcosimPro?

Yes, using the **AFTER** statement. This statement delays events execution; for example, if we write: `x = y + 2 AFTER 2.3` It means that the value of x will be equal to y+2 when the time will be 2.3 seconds after the actual time. Or, in other words, an event is prepared for the time TIME+2.3 in which integration will halt and the statement will be executed. Then it will continue as normal. This technique is widely used in the modelling of digital circuits to simulate delays in electronic components.

Can one variable be made to follow another with a delay?

Yes, you can do this by using the function **delay()**. This function creates a historic archive for the variable in question and it will return the value with a certain delay. For example: `x = delay (y, 2.3)` This statement makes the value of x equal to the value of y after a delay of 2.3 seconds.

What is a virtual equation?

A virtual equation is an equation which is labelled with an identifier and which can be substituted by a different equation in a child component. For example, given this parent component:

```
COMPONENT parent
                DECLS
                REAL x, y
    CONTINUOUS
        y - x + 8.9 = 0.9
        <vEqt > x' = y / sin ( T I M E )
END COMPONENT
```

It is possible to modify the second equation in a child component:

```
COMPONENT child IS_A parent
    CONTINUOUS
                <:vEqt > x' = y / cos ( T I M E )
END COMPONENT
```

The equation labelled "vEqt" is what is known as a virtual equation. Using the substitute operator ":" within the label, the child component replaces it with a new one.

3 Discontinuities

Where should I include the discontinuities in my model?

The most accurate behaviour of models can only be guaranteed if discontinuities are modelled with the **ZONE** statement. Modelling them in any other way could give rise to convergence problems if we are dealing with continuous variables (i.e. variables that are coming from symbolic non-causal equations and put in the CONTINUOUS part of the code).

A typical error is to write discontinuities into external function calls in FORTRAN or C. The system will not complain, but the model will probably not converge.

Another typical place where discontinuities are erroneously modelled is in experiments using IF statements.

Handling discontinuities is one of the **most sophisticated** parts of EcosimPro. A great deal of intelligence is required to go back in time and find the exact point where discontinuities occur. Users are therefore advised to take utmost care when they come to modelling discontinuities. They should always be modelled using the above-mentioned statement and within the specification of components.

What is the main difference between using WHEN and ZONE?

The main difference is that **WHEN** is used for modelling discrete simulation events and **ZONE** is used for modelling conditional symbolic non-causal equations (those put in the CONTINUOUS part of the code). The keyword **WHEN** is associated with a code that is executed only once and that is when the condition is fulfilled. To re-execute the code, the condition must become first unfulfilled and then fulfilled again. On the other hand, **ZONE** ensures that one of the conditional equations is valid at all times. For example, given the following discrete event:

```
WHEN (x > 5.0) THEN
   PRINT ("Ok")
END WHEN
```

The **PRINT** statement will be executed when the value of x becomes greater than 5, but if x then becomes less than 5, nothing will happen. The statement will be executed once again when x becomes greater than 5 from a lower value. However, given the continuous statement:

```
y = ZONE (x > 5) 8
     OTHERS 6
```

EcosimPro will continuously check the condition "x > 5". When x is greater than 5, the first branch becomes valid (y=8), but when this condition is no longer fulfilled, and x is lower than 5, the second branch becomes valid (y=6).

Can the discontinuity of a function be modelled using ZONE or WHEN indiscriminately?

If you want to model continuous behaviour, you should use the **ZONE** statement. By forcing the operator **WHEN** we could use it to model a discontinuity but, as we will see, the application is completely different. For example, we can model the following discontinuous function:

```
y = 8 if x > 5
y = 6 if x <= 5
```

Using ZONE this would be:

```
COMPONENT zoneTest1
      DECLS
        REAL x, y
      CONTINUOUS
          y = ZONE (x > 5) 8
            OTHERS 6
END COMPONENT
```

It can be modelled using two WHEN statements, as follows:

```
COMPONENT whenTest1
      DECLS
        REAL x, y
      DISCRETE
          WHEN (x > 5) THEN
              y = 8
          END WHEN
          WHEN (x <= 5) THEN
              y = 6
          END WHEN
END COMPONENT
```

It can even be modelled yet another way, using just one WHEN statement:

```
COMPONENT whenTest2
      DECLS
        REAL x, y
      DISCRETE
          WHEN ((x > 5) OR (x <= 5)) THEN
              IF (x > 5) THEN
                  y = 8
              ELSE
                  y = 6
              END IF
          END WHEN
END COMPONENT
```

There is a big difference between using ZONE and WHEN: the ZONE statement provides a continuous equation to calculate variable "y", whereas the WHEN statement changes the value of "y" in the precise time when the condition is fulfilled but it doesn't provide an equation. This means that "y" will be a boundary variable unless it is declared as discrete. As we can see, the most appropriate way to model a discontinuity depends strongly on the behaviour that we expect.

Is the terminology the same as that used by classical object-oriented languages such as C++, Java...?

Let's see how the object-oriented terminology changes:

In classical object-oriented languages	In EL (EcosimPro language)
Class	Component
Object	Object
Public inheritance	Public inheritance (any inheritance is Public)
Protected and private inheritance	It doesn't exist
Virtual method	Virtual equation (there are no methods in EcosimPro)
Aggregation	Aggregation

4 Components

What is the main structure of a component?

There following code sketch the most general structure. Blocks are optional. The most powerful is CONTINUOUS.

```
COMPONENT whenTest2
        ...
        PORTS
        ...
        DATA
        ...
        DECLS
        ...
        OBJECTS
        ...
        TOPOLOGY
        ....
        INIT
        ...
        DISCRETE
        ...
        CONTINUOUS
        ...
        ...
        ...
        ...
        ...
END COMPONENT
```

- PORTS: Connection ports. These are the **interface** (if any) with the universe outside the component. *Ports must be of a type already declared.*
- DATA: Component data. These are **known data** items of the component.
- DECLS: The component's **local variables**. They are used to model the component's physical behaviour.
- OBJECTS: The instances of classes internal or external used in this component (if any).
- TOPOLOGY: Aggregation of other existing components and their connections between them (if any).
- INIT: Initialization of the component, that is mainly the **initial values** for the time-derivatives used in equations. These are written as classical sequential statements.
- DISCRETE: Discrete behaviour of the component. Here are written events conditions as discrete statements and written as classical sequential statements (the order is frozen).
- CONTINUOUS: Continuous behaviour of the component. Here are written the so called continuous statements: that is **any kinds of equations** explicit, implicit, algebraic, differential wrt time, conditional, with time derivatives, etc... ,.**written in any order** (non-sequential). Those equations are treated as **symbolic** or **non-causal equations** making the tool handling any kind of model and complex systems, incredibly powerful for solving them.

5 Ports

Why is there a need of IN or OUT for a port used in a component?

Ports variable are algebraic for handling any possible type of flow.
IN or OUT are qualifier mode: IN indicates that the port will be used for input and OUT for output.
By defining new ports you can model any kind of exchange of values between components, and ports have the intelligence necessary to handle multiple connections.
Each port of a component must have a qualifier mode, either IN or OUT in order

- to serve as a reference for **the sign** of the flow-variables.

  A positive flow-variable at an IN port implies an inward flow.

  and a positive flow-variable at an OUT port indicates an outward flow.
- to set criterion for applying **connection equations generated** automatically by EcosimPro.

Qualifier mode IN or OUT are also used to check for violations of connection restrictions specified in the port definition.

What means the SUM EQUAL and IN OUT for a port definition declaration?

EcosimPro add automatically some equations needed to handle correctly the connections between components.
Connecting Equations are necessary in two cases:

- When a port of a components is connected to more than one components at the same port type.
- When a port is connected to another port on the same mode. IN-IN or OUT-OUT.

In both these cases the connected ports cannot be considered equivalent, so the EcosimPro must add the connection equations to relate the variables to the ports involved.

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

Here, SUM and EQUAL indicate how to construct the connection equations for each port variable, while IN or OUT restrict the SUM or EQUAL behaviour only if the port used in a component has the indicated direction.
There are three basic behaviours:

- EQUAL: a variable with this behaviour maintains the same value for all the ports of a connection, regardless whether it's a multiple connection or whether it connects ports with the same mode. In other words, it forces equivalence of the variables.
- SUM: this is associated with flow type variables and indicates that on one connection the sum of the incoming flows is equal to that of the outgoing flow.
- No indication: no specific connection equation is generated for the variable.

```
PORT control SINGLE IN
```

The SINGLE modifier expressly forces all connections for a port type to be single. If no mode is specified, the restriction applies to all ports of this type, regardless of mode. If either of the modifiers IN or OUT is added, the restriction only applies to ports with that mode.
