Conversión de modelos PowerDEVS al lenguaje Modelica

Luciano Andrade

Universidad Nacional de Rosario

andrade.luciano@gmail.com

16 de diciembre de 2015

Introducción

Motivación

- El sistema físico no se encuentra construido.

Motivación

- El sistema físico no se encuentra construido.
- El experimento puede ser peligroso. Se realizan simulaciones para determinar si el experimento real “explotara”.

Motivación

- El sistema físico no se encuentra construido.
- El experimento puede ser peligroso. Se realizan simulaciones para determinar si el experimento real “explotara”.
- El costo del experimento es demasiado alto o las herramientas necesarias no se encuentran disponibles o son muy costosas.

Motivación

- El sistema físico no se encuentra construido.
- El experimento puede ser peligroso. Se realizan simulaciones para determinar si el experimento real “explotara”.
- El costo del experimento es demasiado alto o las herramientas necesarias no se encuentran disponibles o son muy costosas.
- Los tiempos del sistema no son compatibles con los tiempos del experimentador, ya sea porque es demasiado rápido o porque es demasiado lento.

Motivación

- El sistema físico no se encuentra construido.
- El experimento puede ser peligroso. Se realizan simulaciones para determinar si el experimento real "explotara".
- El costo del experimento es demasiado alto o las herramientas necesarias no se encuentran disponibles o son muy costosas.
- Los tiempos del sistema no son compatibles con los tiempos del experimentador, ya sea porque es demasiado rápido o porque es demasiado lento.
- Variables de control, de estado y/o del sistema pueden no ser accesibles. Las simulaciones también nos permite manipular el modelo en formas que no podríamos manipular el sistema real.

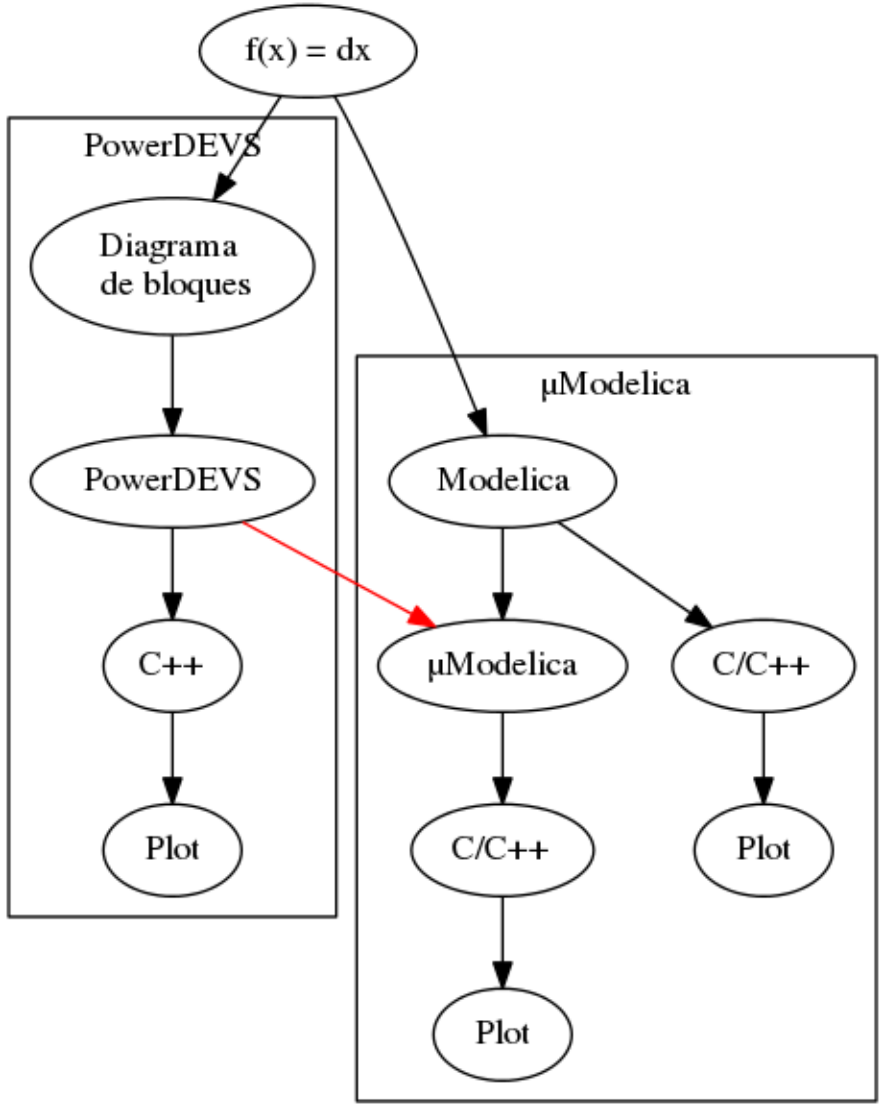
Motivación

- El sistema físico no se encuentra construido.
- El experimento puede ser peligroso. Se realizan simulaciones para determinar si el experimento real "explotara".
- El costo del experimento es demasiado alto o las herramientas necesarias no se encuentran disponibles o son muy costosas.
- Los tiempos del sistema no son compatibles con los tiempos del experimentador, ya sea porque es demasiado rápido o porque es demasiado lento.
- Variables de control, de estado y/o del sistema pueden no ser accesibles. Las simulaciones también nos permite manipular el modelo en formas que no podríamos manipular el sistema real.
- Eliminación de perturbaciones. Lo que nos permite aislar efectos particulares, y puede conducir a mejores apreciaciones sobre el comportamiento general del sistema.

Motivación

- El sistema físico no se encuentra construido.
- El experimento puede ser peligroso. Se realizan simulaciones para determinar si el experimento real "explotara".
- El costo del experimento es demasiado alto o las herramientas necesarias no se encuentran disponibles o son muy costosas.
- Los tiempos del sistema no son compatibles con los tiempos del experimentador, ya sea porque es demasiado rápido o porque es demasiado lento.
- Variables de control, de estado y/o del sistema pueden no ser accesibles. Las simulaciones también nos permite manipular el modelo en formas que no podríamos manipular el sistema real.
- Eliminación de perturbaciones. Lo que nos permite aislar efectos particulares, y puede conducir a mejores apreciaciones sobre el comportamiento general del sistema.
- Eliminación de efectos de segundo orden (como no linealidades de componentes del sistema).

Esquema de conversiones



Trabajos relacionados

- En “Simulating Modelica models with a Stand–Alone Quantized State Systems Solver”, 2012 se describe una extensión del Compilador OpenModelica el cual traslada modelos regulares Modelica a un subconjunto más simple μ-Modelica, el cual puede ser interpretado directamente por el QSS-Solver.

Trabajos relacionados

- En "Simulating Modelica models with a Stand–Alone Quantized State Systems Solver", 2012 se describe una extensión del Compilador OpenModelica el cual traslada modelos regulares Modelica a un subconjunto más simple μ-Modelica, el cual puede ser interpretado directamente por el QSS-Solver.
- ModelicaDEVS es una librería Modelica que permite describir simulaciones DEVS, ofrece una re-implementación de PowerDEVS dentro del marco de Modelica.

Trabajos relacionados

- En "Simulating Modelica models with a Stand–Alone Quantized State Systems Solver", 2012 se describe una extensión del Compilador OpenModelica el cual traslada modelos regulares Modelica a un subconjunto más simple μ-Modelica, el cual puede ser interpretado directamente por el QSS-Solver.
- ModelicaDEVS es una librería Modelica que permite describir simulaciones DEVS, ofrece una re-implementación de PowerDEVS dentro del marco de Modelica.
- DESlib es una librería para la descripción de modelos Parallel DEVS y Modelado orientado a proceso en Modelica. La librería contiene cuatro paquetes que pueden ser utilizados para modelar sistemas de eventos discretos: RandomLib, DEVSLib, SIMANLib y ARENALib.

Trabajos relacionados

- En "Simulating Modelica models with a Stand–Alone Quantized State Systems Solver", 2012 se describe una extensión del Compilador OpenModelica el cual traslada modelos regulares Modelica a un subconjunto más simple μ-Modelica, el cual puede ser interpretado directamente por el QSS-Solver.
- ModelicaDEVS es una librería Modelica que permite describir simulaciones DEVS, ofrece una re-implementación de PowerDEVS dentro del marco de Modelica.
- DESlib es una librería para la descripción de modelos Parallel DEVS y Modelado orientado a proceso en Modelica. La librería contiene cuatro paquetes que pueden ser utilizados para modelar sistemas de eventos discretos: RandomLib, DEVSLib, SIMANLib y ARENALib.
- M/CD++ es una herramienta para convertir simulaciones de un subconjunto de Modelica, a simulaciones DEVS.

Conceptos Previos

Sistemas Continuos y Discretos

$$\dot{x}(t) = f(x(t), u(t))$$

Con condiciones iniciales :

$$x(t = t_0) = x_0$$

Si f es continua puede ser aproximada mediante series de Taylor:

$$x_i(t^* + h) = x_i(t^*) + \frac{dx_i(t^*)}{dt} \cdot h + \frac{d^2x_i(t^*)}{dt^2} \cdot \frac{h^2}{2!} + \cdots$$

$$x_i(t^* + h) = x_i(t^*) + f_i(t^*) \cdot h + \frac{d^2x_i(t^*)}{dt^2} \cdot \frac{h^2}{2!} + \cdots$$

Lotka Volterra

$$\frac{dx}{dt} = x(\alpha - \beta y)$$
$$\frac{dy}{dt} = -y(\gamma - \delta x)$$

donde:

- y es el número de algún predador (por ejemplo, un lobo)
- x es el número de sus presas (por ejemplo, conejos)
- $\frac{dy}{dt}$ y $\frac{dx}{dt}$ representa el crecimiento de las dos poblaciones en el tiempo
- t representa el tiempo
- α, β, γ y δ son parámetros que representan las interacciones de las dos especies.

Modelica

```
1  class LotkaVolterra
2    Real x(start = 0.5);
3    Real y(start = 0.5);
4    parameter Real a = 0.1;
5    parameter Real b = 0.1;
6    parameter Real c = 0.1;
7    parameter Real d = 0.1;
8  equation
9    0 =  x * (a - b * y) - der(x);
10   0 = der(y) + y * (d - c * x);
11 end LotkaVolterra;
```

Listing 1: LotkaVolterra.mo

Métodos de Integración QSS

El método de QSS de primer orden (llamado QSS1) aproxima la ecuación por:

$$\dot{x}(t) = f(q(t), v(t))$$

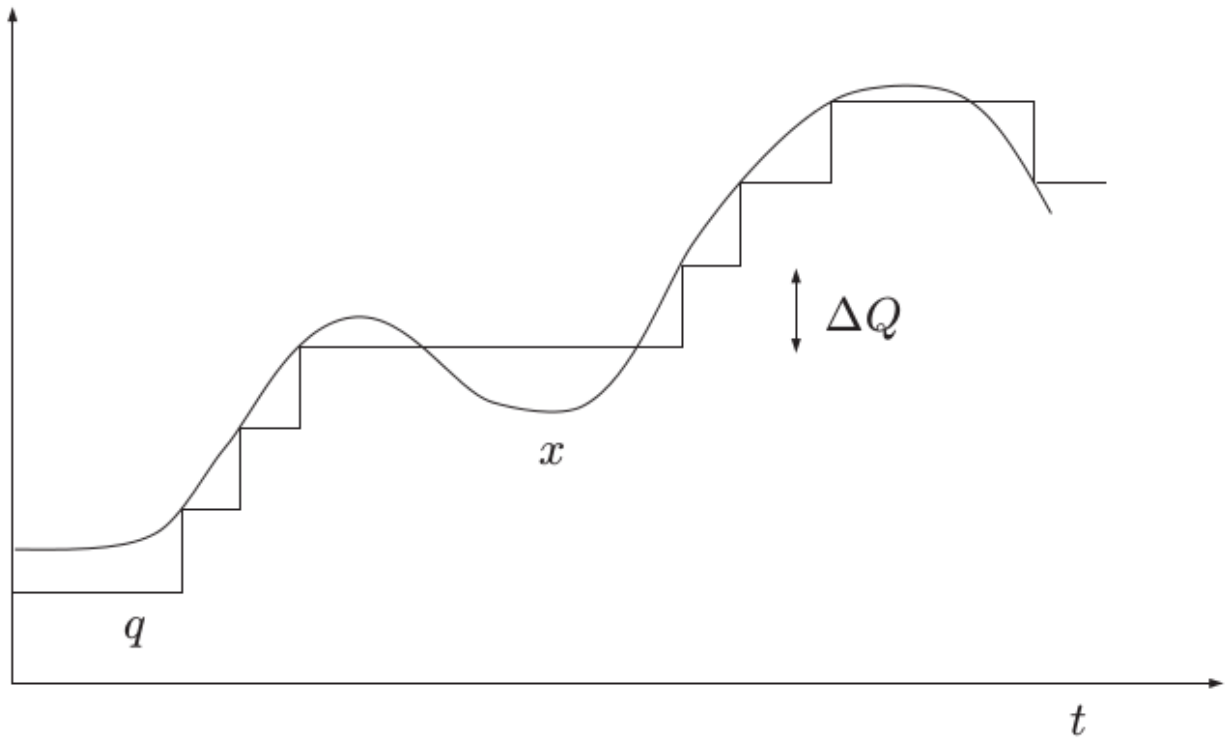
donde q es el vector de estados cuantificados y sus componentes están relacionadas una a una con las del vector de estados x siguiendo una función de cuantificación con histéresis:

$$q_j(t) = \begin{cases} x_j(t) & \text{si } \mid x_j(t) - q_j(t^-) \mid \geq \Delta Q_j \\ q_j(t^-) & \text{en caso contrario} \end{cases}$$

donde $q_j(t^-)$ es el límite por izquierda de q_j en t.

Métodos de Integración QSS

Variables Relacionadas con una Función de Cuantificación con Histéresis de orden cero.



μ-Modelica (1)

- El modelo es plano, es decir no permite clases.

μ-Modelica (1)

- El modelo es plano, es decir no permite clases.
- Todas las variables pertenecen al tipo predefinido Real y solo hay tres categorías de variables: estado continuo, estado discreto y variables algebraicas.

μ-Modelica (1)

- El modelo es plano, es decir no permite clases.
- Todas las variables pertenecen al tipo predefinido Real y solo hay tres categorías de variables: estado continuo, estado discreto y variables algebraicas.
- Los parámetros también son de tipo Real.

μ-Modelica (1)

- El modelo es plano, es decir no permite clases.
- Todas las variables pertenecen al tipo predefinido Real y solo hay tres categorías de variables: estado continuo, estado discreto y variables algebraicas.
- Los parámetros también son de tipo Real.
- Solo Arreglos unidimensionales están permitidos.

μ-Modelica (1)

- El modelo es plano, es decir no permite clases.
- Todas las variables pertenecen al tipo predefinido Real y solo hay tres categorías de variables: estado continuo, estado discreto y variables algebraicas.
- Los parámetros también son de tipo Real.
- Solo Arreglos unidimensionales están permitidos.
- Discontinuidades son expresadas solo con las clausulas *when* y *elsewhen* dentro de la sección *algorithm*. Las condiciones dentro de las dos clausulas solo pueden ser relaciones ($<$, $\leqslant$, $>$ $\geqslant$) y, dentro de la clausula, solo asignaciones de variables discretas y *reinit* de estados continuos son permitidos.

μ-Modelica (2)

- Indices en los arreglos dentro de cláusulas `for` están restringidos a la forma $\alpha \cdot i + \beta$, donde α y β son expresiones enteras e `i` es el índice de la iteración.

μ-Modelica (2)

- Indices en los arreglos dentro de cláusulas `for` están restringidos a la forma $\alpha \cdot i + \beta$, donde α y β son expresiones enteras e `i` es el índice de la iteración.
- La sección de ecuaciones está compuesta de :
  - Definición de variables de estados :
    $der(x) = f(x(t), d, a(t), t)$; ODE en forma explícita
  - Definición algebraica : $a_2 = g(a_1)$;

  con la restricción de que cada variable algebraica solo puede depender de variables estado y de variables algebraicas previamente definidas.

Stand-Alone QSS-Solver

```
model lotka_volterra
  Real x(start = 0.5);
  Real y(start = 0.5);
  parameter Real a = 0.1;
  parameter Real b = 0.1;
  parameter Real c = 0.1;
  parameter Real d = 0.1;
initial algorithm
  x := 0.5;
  y := 0.5;
equation
  der(x) =  x * (a - b * y);
  der(y) =  - y * (d - c * x);
        annotation(
        experiment(
                MMO_Description="Lotka Volterra model",
                MMO_Solver=QSS3,
                MMO_Output={x[:]},
                StartTime= 0.0,
                StopTime= 300.0,
                Tolerance={ 1e-3},
                AbsTolerance={ 1e-6}
        ));
end lotka_volterra;
```

Formalismo DEVS / Modelos Atómicos

Formalmente un modelo atómico está conformado por la 7-upla:

$$(X, Y, S, \delta_{int}, \delta_{ext}, \lambda, t_a) \text{ donde :} \quad (1)$$

- X es el conjunto de valores de entrada que acepta el modelo atómico.

Formalismo DEVS / Modelos Atómicos

Formalmente un modelo atómico está conformado por la 7-upla:

$$(X, Y, S, \delta_{int}, \delta_{ext}, \lambda, t_a) \text{ donde :} \tag{1}$$

- X es el conjunto de valores de entrada que acepta el modelo atómico.
- Y es el conjunto de valores de los eventos de salida que puede emitir el modelo atómico.

Formalismo DEVS / Modelos Atómicos

Formalmente un modelo atómico está conformado por la 7-upla:

$$(X, Y, S, \delta_{int}, \delta_{ext}, \lambda, t_a) \text{ donde :} \tag{1}$$

- X es el conjunto de valores de entrada que acepta el modelo atómico.
- Y es el conjunto de valores de los eventos de salida que puede emitir el modelo atómico.
- S es el conjunto de estados internos del modelo.

Formalismo DEVS / Modelos Atómicos

Formalmente un modelo atómico está conformado por la 7-upla:

$$(X, Y, S, \delta_{int}, \delta_{ext}, \lambda, t_a) \text{ donde :} \tag{1}$$

- X es el conjunto de valores de entrada que acepta el modelo atómico.
- Y es el conjunto de valores de los eventos de salida que puede emitir el modelo atómico.
- S es el conjunto de estados internos del modelo.
- *ta* es una función $S \rightarrow \mathbb{R}_0^+$ de *Avance de Tiempo*.

Formalismo DEVS / Modelos Atómicos

Formalmente un modelo atómico está conformado por la 7-upla:

$$(X, Y, S, \delta_{int}, \delta_{ext}, \lambda, t_a) \text{ donde :} \tag{1}$$

- X es el conjunto de valores de entrada que acepta el modelo atómico.
- Y es el conjunto de valores de los eventos de salida que puede emitir el modelo atómico.
- S es el conjunto de estados internos del modelo.
- ta es una función $S \rightarrow \mathbb{R}_0^+$ de *Avance de Tiempo*.
- δ_{int} es una función $S \rightarrow S$ de *Transición Interna*.

Formalismo DEVS / Modelos Atómicos

Formalmente un modelo atómico está conformado por la 7-upla:

$$(X, Y, S, \delta_{int}, \delta_{ext}, \lambda, t_a) \text{ donde :} \tag{1}$$

- X es el conjunto de valores de entrada que acepta el modelo atómico.
- Y es el conjunto de valores de los eventos de salida que puede emitir el modelo atómico.
- S es el conjunto de estados internos del modelo.
- ta es una función $S \to \mathbb{R}_0^+$ de *Avance de Tiempo*.
- δ_{int} es una función $S \to S$ de *Transición Interna*.
- δ_{ext} es una función $(S \times \mathbb{R}_0^+ \times X) \to S$ de *Transición Externa*.

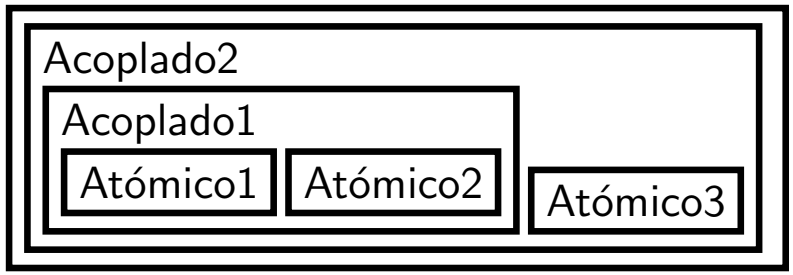
Formalismo DEVS / Modelos Atómicos

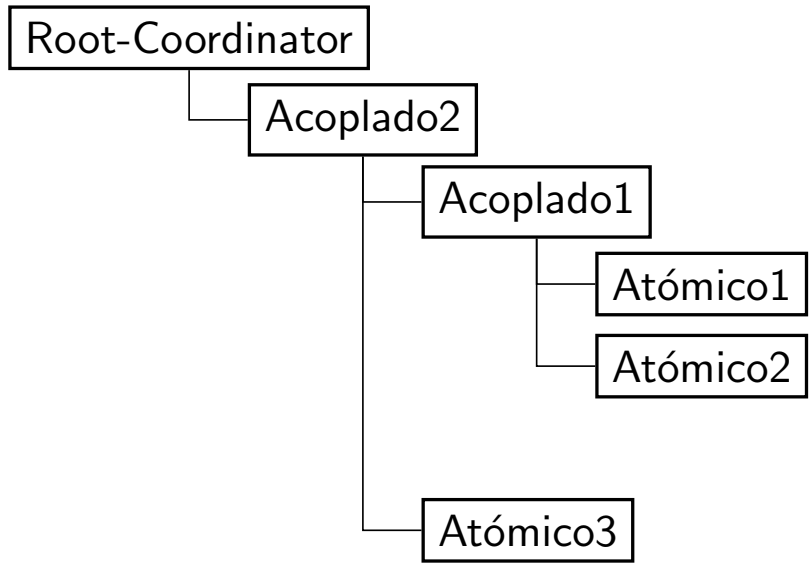
Formalmente un modelo atómico está conformado por la 7-upla:

$$(X, Y, S, \delta_{int}, \delta_{ext}, \lambda, t_a) \text{ donde :} \tag{1}$$

- X es el conjunto de valores de entrada que acepta el modelo atómico.
- Y es el conjunto de valores de los eventos de salida que puede emitir el modelo atómico.
- S es el conjunto de estados internos del modelo.
- ta es una función $S \rightarrow \mathbb{R}_0^+$ de *Avance de Tiempo*.
- δ_{int} es una función $S \rightarrow S$ de *Transición Interna*.
- δ_{ext} es una función $(S \times \mathbb{R}_0^+ \times X) \rightarrow S$ de *Transición Externa*.
- λ es una función $S \rightarrow Y$ de *Salida*.

Modelo Acoplados





Modelos DEVS Parametrizados

$$M(p) = \{X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta, p\}$$

donde $p \in P$ es un parámetro que pertenece a un conjunto de parámetros arbitrario tal que δ_{int} , δ_{ext} , λ y ta dependen también de p. Notar que dos modelos DEVS $M(p_1)$, $M(p_2)$ con $p_1 \neq p_2$ pueden exhibir distintos comportamientos aunque compartan los mismos conjuntos de entrada, salida y de estados (X, Y , y S, respectivamente).

Modelos Vectoriales

$$M(p) = \{X, Y, S, \lambda_{int}, \lambda_{ext}, \lambda, ta, p\}$$

definimos un modelo Vectorial DEVS como la estructura:

$$V_D = \{N, X_V, Y_V, P, \{M_i\}\},$$

Modelos Vectoriales

- $N \in \mathbb{N}$ es la dimensión del modelo vectorial.

Modelos Vectoriales

- $N \in \mathbb{N}$ es la dimensión del modelo vectorial.
- $X_V = X \times Index \bigcup \{-1\}$ es el conjunto de eventos de entradas vectorial donde X es el conjunto de eventos de entrada del modelo escalar e $Index = 1, \ldots, N$ es el conjunto de índices que indican cuál de los modelos DEVS atómicos recibirá el evento.

Modelos Vectoriales

- $N \in \mathbb{N}$ es la dimensión del modelo vectorial.
- $X_V = X \times Index \bigcup \{-1\}$ es el conjunto de eventos de entradas vectorial donde X es el conjunto de eventos de entrada del modelo escalar e $Index = 1, \ldots, N$ es el conjunto de índices que indican cuál de los modelos DEVS atómicos recibirá el evento.
- $Y_V = Y \times Index$ es el conjunto de eventos de salida vectorial donde Y es el conjunto de eventos de salida del modelo escalar e $Index = 1, \ldots, N$ es el conjunto de índices que indica que modelo escalar de los N , emitió el evento.

Modelos Vectoriales

- $N \in \mathbb{N}$ es la dimensión del modelo vectorial.
- $X_V = X \times Index \bigcup \{-1\}$ es el conjunto de eventos de entradas vectorial donde X es el conjunto de eventos de entrada del modelo escalar e $Index = 1, \ldots, N$ es el conjunto de índices que indican cuál de los modelos DEVS atómicos recibirá el evento.
- $Y_V = Y \times Index$ es el conjunto de eventos de salida vectorial donde Y es el conjunto de eventos de salida del modelo escalar e $Index = 1, \ldots, N$ es el conjunto de índices que indica que modelo escalar de los N , emitió el evento.
- P es un conjunto de parámetros arbitrario.

Modelos Vectoriales

- $N \in \mathbb{N}$ es la dimensión del modelo vectorial.
- $X_V = X \times Index \bigcup \{-1\}$ es el conjunto de eventos de entradas vectorial donde X es el conjunto de eventos de entrada del modelo escalar e $Index = 1, \ldots, N$ es el conjunto de índices que indican cuál de los modelos DEVS atómicos recibirá el evento.
- $Y_V = Y \times Index$ es el conjunto de eventos de salida vectorial donde Y es el conjunto de eventos de salida del modelo escalar e $Index = 1, \ldots, N$ es el conjunto de índices que indica que modelo escalar de los N , emitió el evento.
- P es un conjunto de parámetros arbitrario.
- Para cada índice $i \in Index$, $p(i) \in P$ es un parámetro y $M_i = M(p(i))$ es el modelo DEVS Parametrizado escalar.

Modelos Vectorial

- Escalar a Vector (Scalar to Vector): Este bloque simplemente agrega al indice i al evento escalar que recibe, transformándolo en un evento vectorial. Este modelo también posee un comportamiento especial para enviar el mismo evento en todas las componentes vectorial al mismo tiempo, cuando $i = -1$, cada evento de entrada es trasmitido para todas las componentes del vector salida.

Modelos Vectorial

- Escalar a Vector (Scalar to Vector): Este bloque simplemente agrega al indice i al evento escalar que recibe, transformándolo en un evento vectorial. Este modelo también posee un comportamiento especial para enviar el mismo evento en todas las componentes vectorial al mismo tiempo, cuando $i = -1$, cada evento de entrada es trasmitido para todas las componentes del vector salida.
- Vector a escalar (Vector to Scalar): Este bloque tiene un parámetro i que contiene el índice del vector de eventos a retransmitir, cuando recibe un evento con indice $j = i$, remueve el indice y retransmite el evento escalar.

Modelos Vectorial

- Escalar a Vector (Scalar to Vector): Este bloque simplemente agrega al indice i al evento escalar que recibe, transformándolo en un evento vectorial. Este modelo también posee un comportamiento especial para enviar el mismo evento en todas las componentes vectorial al mismo tiempo, cuando $i = -1$, cada evento de entrada es trasmitido para todas las componentes del vector salida.
- Vector a escalar (Vector to Scalar): Este bloque tiene un parámetro i que contiene el índice del vector de eventos a retransmitir, cuando recibe un evento con indice $j = i$, remueve el indice y retransmite el evento escalar.
- Index Shift: El modelo más simple es el Index Shift. Cuando se recibe un evento con el valor (x, i), emite un evento de salida $(x, i + sh)$, donde sh es un parámetro entero.

PowerDEVS

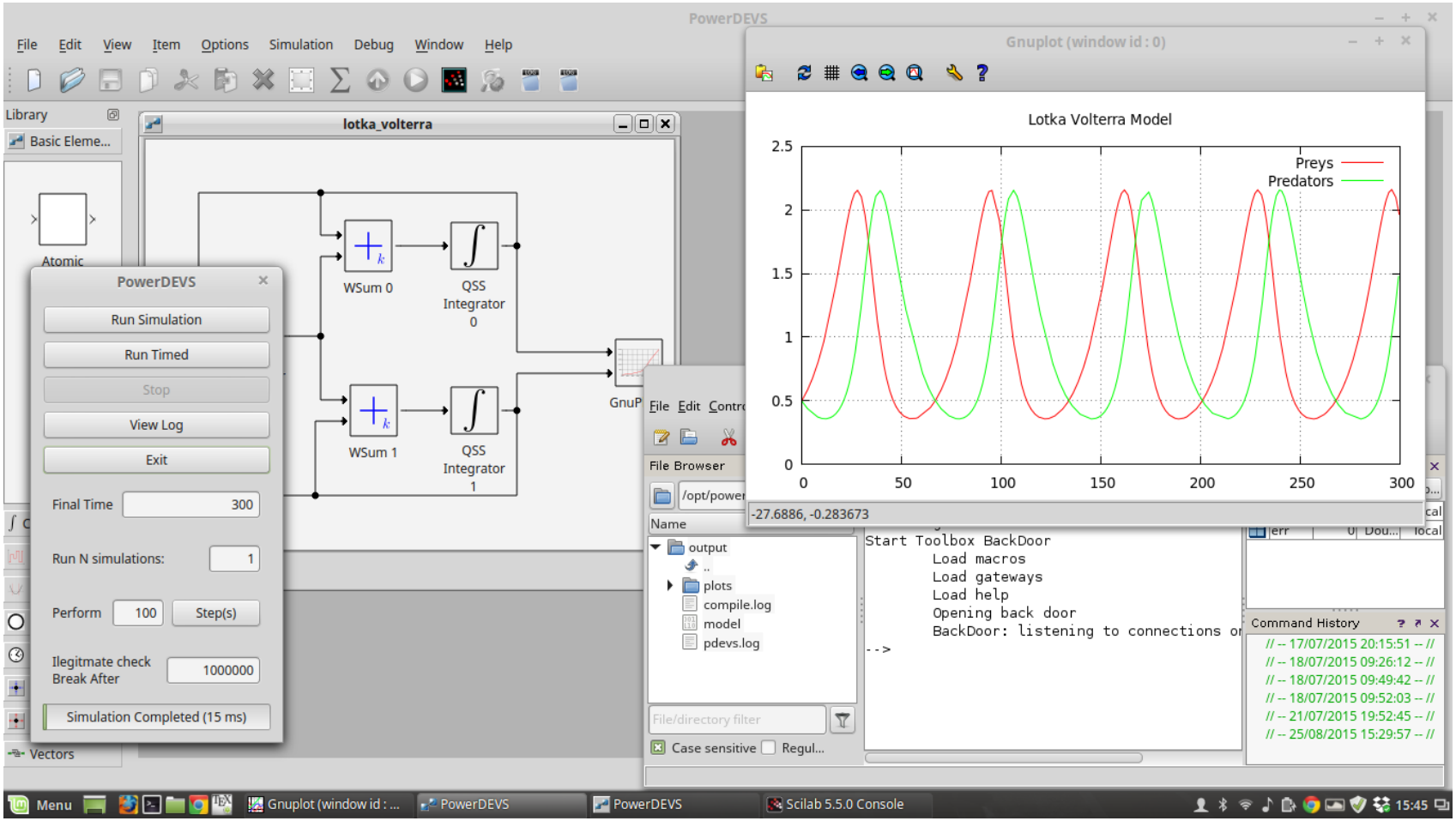


Figura : Interfaz gráfica de PowerDEVS

Conversión de modelos DEVS

Estructura del archivo lotka_volterra.pds, modelos atómicos (continua) 1

```
1  Root-Coordinator
2   {
3    Simulator
4     {
5      Path = qss/qss_multiplier.h
6      Parameters = "Purely static","1e-6","1e-3"
7     }
8    Simulator
9     {
10     Path = qss/qss_integrator.h
11     Parameters = "QSS3","1e-6","1e-3","0.5"
12    }
```

Estructura del archivo lotka_volterra.pds, modelos atómicos (continua) 2

```
13      Simulator
14       {
15        Path = qss/qss_integrator.h
16        Parameters = "QSS3","1e-6","1e-3","0.5"
17       }
18      Simulator
19       {
20        Path = qss/qss_wsum.h
21        Parameters =
    ↪    "0.1","-0.1","0","0","0","0","0","0",2.000000e+00
22       }
```

Estructura del archivo lotka_volterra.pds, modelos atómicos (continua) 3

```
23      Simulator
24       {
25        Path = qss/qss_wsum.h
26        Parameters =
    ↪   "0.1","-0.1","0","0","0","0","0","0",2.000000e+00
27       }
28      Simulator
29       {
30        Path = sinks\gnuplot.h
31        Parameters = 2.000000e+00,"set xrange [0:%tf] @
    ↪   set grid @ set title 'Lotka Volterra
    ↪   Model'","with lines title 'Preys'","with lines
    ↪   title 'Predators'","","",""
32       }
```

conexiones del archivo lotka_volterra.pds. 4

```
33      EIC
34       {
35       }
36      EOC
37       {
38       }
39      IC
40       {
41         (3,0);(1,0)
42         (4,0);(2,0)
43         (0,0);(4,0)
44         (0,0);(3,1)
45         (1,0);(5,0)
46         (1,0);(0,0)
47         (1,0);(3,0)
48         (2,0);(5,1)
49         (2,0);(0,1)
50         (2,0);(4,1)
51       }
52    }
```

Modelos Continuos

```
class WSum
        parameter Real p[9]={0,0,0,0,0,0,0,0};
        parameter Integer n= integer(p[9]);
        parameter Real w[n] = p[1:n];
         Real u[n];
        Real y[1];
equation
        y[1]=u*w;
end WSum;
```

Modelos Discontinuos - qss_switch.mo - 1

```
1  model qss_switch
2    parameter Real p[1] = {0};
3    parameter Real level = p[1];
4    Real u[3];
5    Real y[1];
6    discrete Real d;
7  equation
8    y[1] = u[1] * d + u[3] * (1 - d);
```

Modelos Discontinuos - qss_switch.mo - 2

```
9   initial algorithm
10    if u[2] > level then
11      d := 1;
12    elseif u[2] < level then
13      d := 0;
14    end if;
15  algorithm
16    when u[2] > level then
17      d := 1;
18    elsewhen u[2] < level then
19      d := 0;
20    end when;
21  end qss_switch;
```

Protocolo

- El código debe ser Modelica (μ-modelica) válido y estar ubicado en el mismo directorio (y nombre del archivo) del código C que el modelo atómico PowerDEVS, con el mismo nombre que el archivo .h, pero con extensión .mo, es decir un modelo con `vector\qss_sum_vec.h` utilizará un modelo `vector/qss_sum_vec.h`

Protocolo

- El código debe ser Modelica (μ-modelica) válido y estar ubicado en el mismo directorio (y nombre del archivo) del código C que el modelo atómico PowerDEVS, con el mismo nombre que el archivo .h, pero con extensión .mo, es decir un modelo con `vector\qss_sum_vec.h` utilizará un modelo `vector/qss_sum_vec.h`
- Los parámetros del modelo DEVS deben ser pasado en el parámetro p, el cual es un arreglo de reales.

Protocolo

- El código debe ser Modelica (μ-modelica) válido y estar ubicado en el mismo directorio (y nombre del archivo) del código C que el modelo atómico PowerDEVS, con el mismo nombre que el archivo .h, pero con extensión .mo, es decir un modelo con `vector\qss_sum_vec.h` utilizará un modelo `vector/qss_sum_vec.h`
- Los parámetros del modelo DEVS deben ser pasado en el parámetro p, el cual es un arreglo de reales.
- Los valores de entrada son asociados a la variable u

Protocolo

- El código debe ser Modelica (μ-modelica) válido y estar ubicado en el mismo directorio (y nombre del archivo) del código C que el modelo atómico PowerDEVS, con el mismo nombre que el archivo .h, pero con extensión .mo, es decir un modelo con `vector\qss_sum_vec.h` utilizará un modelo `vector/qss_sum_vec.h`
- Los parámetros del modelo DEVS deben ser pasado en el parámetro p, el cual es un arreglo de reales.
- Los valores de entrada son asociados a la variable u
- Los valores de salida son asociados a la variable y

QSSIntegrator

```
class QSSIntegrator
  parameter Real p[4]={0,0,0,0,0,0,0,0};
  parameter Real x0 = p[4];
  Real u[1];
  Real y[1](start = {x0});
equation
  der(y[1]) = u[1];
end QSSIntegrator;
```

Extracto del modelo Lotka Volterra, modelo atómico de un integrator.

```
...
Simulator
 {
  Path = qss/qss_integrator.h
  Parameters = "QSS3","1e-6","1e-3","0.5"
 }
 ...
```

Transformación parcial de un modelo atómico de un integrator en el modelo de ejemplo Lotka Volterra.

```
class QSSIntegrator
  parameter Real QSSIntegrator_1_p[4]={0,1e-6, 1e-3, 0.5};
  parameter Real QSSIntegrator_1_x0 = p[4];
  Real QSSIntegrator_1_u[1];
  Real QSSIntegrator_1_y[1](start = {QSSIntegrator_1_x0});
equation
  der(QSSIntegrator_1_y[1]) = QSSIntegrator_1_u[1];
end QSSIntegrator;
```

Modelo Lotka Volterra convertido de PowerDEVS a μ-Modelica - 1

```
1  model lotka_volterra
2    Real qss_multiplier_0_u[2];
3    Real qss_multiplier_0_y[1];
4    parameter Real QSSIntegrator_1_p[4] =
   ↪  {0,1e-06,0.001,0.5};
5    parameter Real QSSIntegrator_1_x0 = 0.5;
6    Real QSSIntegrator_1_u[1];
7    Real QSSIntegrator_1_y[1](start =
   ↪  {QSSIntegrator_1_x0});
8    parameter Real QSSIntegrator_2_p[4] =
   ↪  {0,1e-06,0.001,0.5};
9    parameter Real QSSIntegrator_2_x0 = 0.5;
```

Modelo Lotka Volterra convertido de PowerDEVS a μ-Modelica - 2

```
10    Real QSSIntegrator_2_u[1];
11    Real QSSIntegrator_2_y[1](start =
    ↪  {QSSIntegrator_2_x0});
12    parameter Real WSum_3_p[9] =
    ↪  {0.1,(-0.1),0,0,0,0,0,0,2};
13    parameter Integer WSum_3_n = integer(2);
14    parameter Real WSum_3_w[WSum_3_n] =
    ↪  WSum_3_p[1:WSum_3_n];
15    Real WSum_3_u[WSum_3_n];
16    Real WSum_3_y[1];
```

Modelo Lotka Volterra convertido de PowerDEVS a μ-Modelica - 3

```
17    parameter Real WSum_4_p[9] =
    ↪  {0.1,(-0.1),0,0,0,0,0,0,2};
18    parameter Integer WSum_4_n = integer(2);
19    parameter Real WSum_4_w[WSum_4_n] =
    ↪  WSum_4_p[1:WSum_4_n];
20    Real WSum_4_u[WSum_4_n];
21    Real WSum_4_y[1];
```

Modelo Lotka Volterra convertido de PowerDEVS a μ-Modelica - 4

```
22    equation
23      qss_multiplier_0_y[1] =
    ↪   qss_multiplier_0_u[1]*qss_multiplier_0_u[2];
24      der(QSSIntegrator_1_y[1]) = QSSIntegrator_1_u[1];
25      der(QSSIntegrator_2_y[1]) = QSSIntegrator_2_u[1];
26      WSum_3_y[1] = WSum_3_u*WSum_3_w;
27      WSum_4_y[1] = WSum_4_u*WSum_4_w;
28      qss_multiplier_0_u[1] = QSSIntegrator_1_y[1];
29      qss_multiplier_0_u[2] = QSSIntegrator_2_y[1];
30      QSSIntegrator_1_u[1] = WSum_3_y[1];
31      WSum_3_u[2] = qss_multiplier_0_y[1];
32      WSum_3_u[1] = QSSIntegrator_1_y[1];
33      QSSIntegrator_2_u[1] = WSum_4_y[1];
34      WSum_4_u[1] = qss_multiplier_0_y[1];
35      WSum_4_u[2] = QSSIntegrator_2_y[1];
36  end lotka_volterra;
```

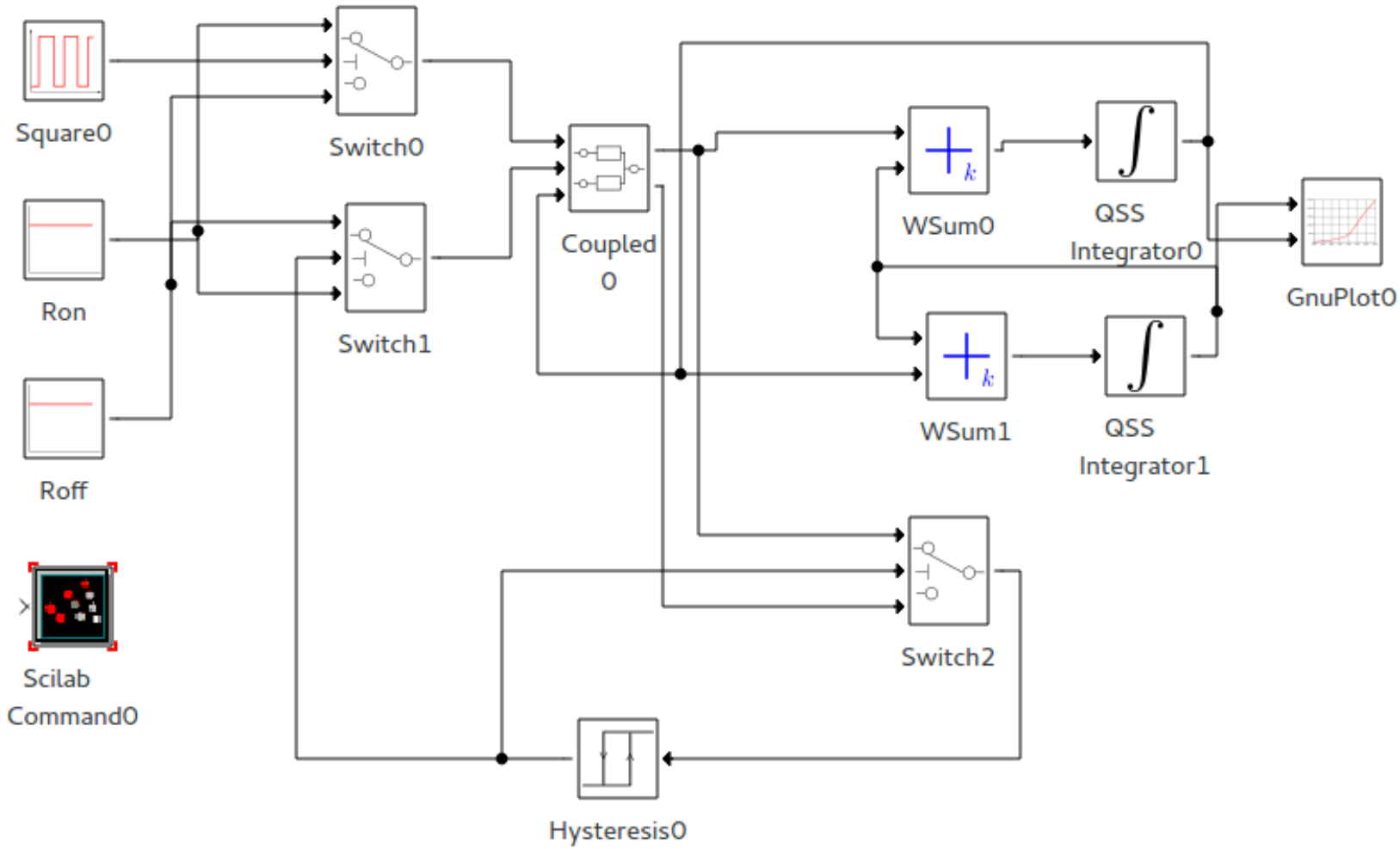
Modelos Acoplados Jerárquico

- por cada modelo acoplado si solo tiene modelos atómicos, es remplazado por los modelos atómicos internos, los cuales se encuentran conectados sin modificaciones excepto por las conexiones externas, las cuales son reasignadas de forma de mantener las conexiones.

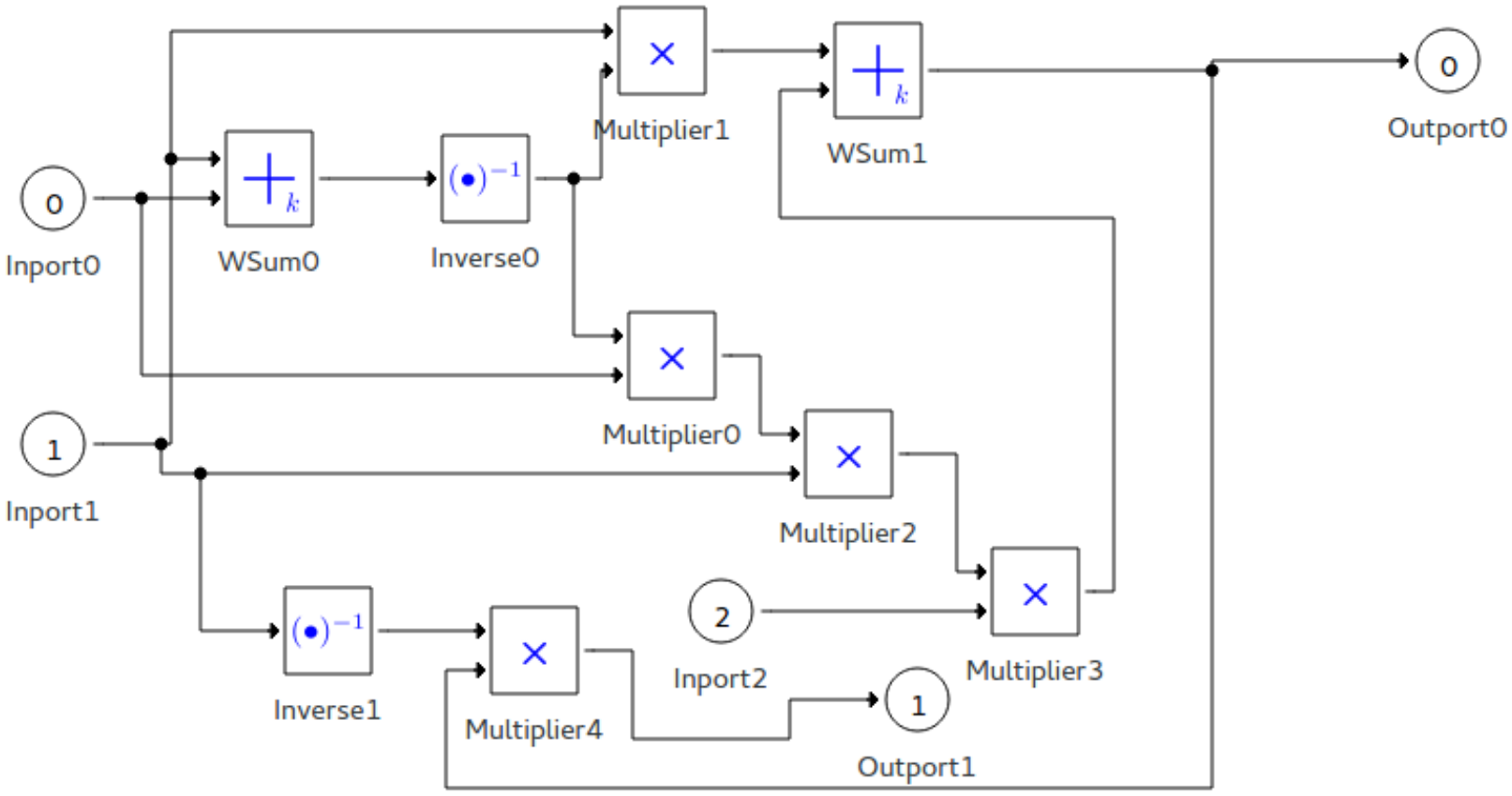
Modelos Acoplados Jerárquico

- por cada modelo acoplado si solo tiene modelos atómicos, es remplazado por los modelos atómicos internos, los cuales se encuentran conectados sin modificaciones excepto por las conexiones externas, las cuales son reasignadas de forma de mantener las conexiones.
- si el modelo acoplado contiene otros modelos acoplados entonces aplanamos ese modelo recursivamente.

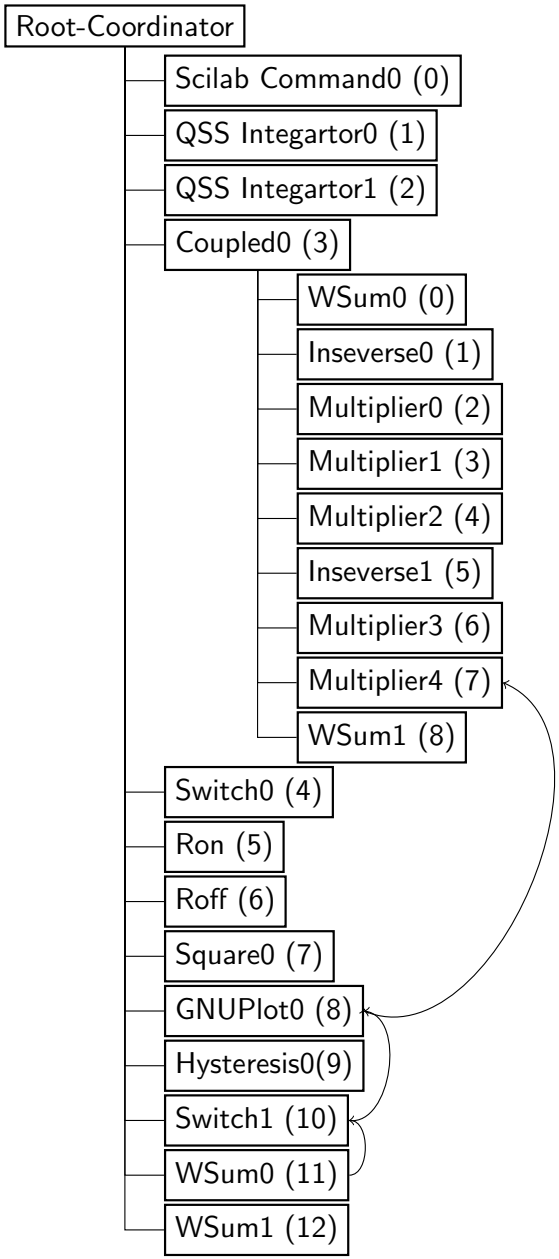
Ejemplo de modelo acoplado - Convertidor de Voltaje

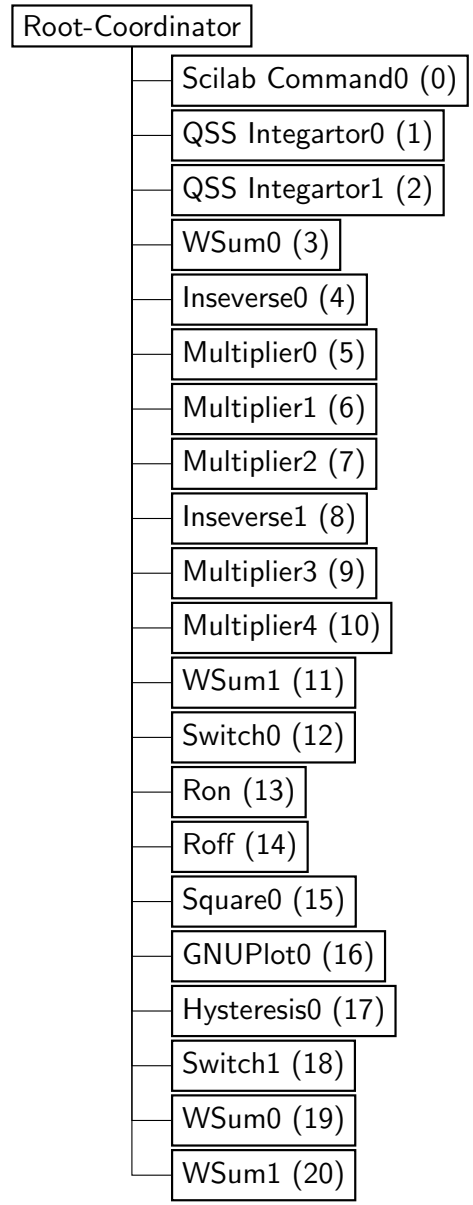


Ejemplo de modelo acoplado - Convertidor de Voltaje - Coupled0



Convertidor de Voltaje





Conexiones del Modelo Convertidor de Voltaje

```
IC
  {
    (13,0);(1,0)
    (12,0);(2,0)
    (8,0);(4,1)
    (4,0);(3,0)
    (5,0);(3,1)
    (3,1);(11,2)
    (11,0);(10,0)
    (2,0);(9,0)
    (2,0);(12,0)
    (2,0);(13,1)
    (3,0);(11,0)
    (3,0);(13,0)
    (10,0);(11,1)
    (10,0);(5,1)
    (1,0);(3,2)
    (1,0);(12,1)
    (1,0);(9,1)
    (6,0);(5,2)
    (6,0);(4,0)
    (7,0);(5,0)
    (7,0);(4,2)
  }
```

Conexiones del Modelo Convertidor de Voltaje - Modelo Acoplado

```
EIC
  {
   (0,2);(4,1)
   (0,0);(2,1)
   (0,0);(0,1)
   (0,1);(3,1)
   (0,1);(5,0)
   (0,1);(7,0)
   (0,1);(0,0)
  }
 EOC
  {
   (6,0);(0,1)
   (8,0);(0,0)
  }
```

```
IC
 {
  (0,0);(1,0)
  (7,0);(8,0)
  (2,0);(3,0)
  (3,0);(4,0)
  (5,0);(6,0)
  (4,0);(8,1)
  (1,0);(2,0)
  (1,0);(7,1)
  (8,0);(6,1)
 }
```

Remapeo de Conexiones Internas

- Conexiones que involucran modelos atómicos ubicados después del modelo acoplado y no están relacionadas con el modelo acoplado, estas conexiones deben ser modificadas dado que insertaremos los modelos atómicos del modelo acoplado que estamos aplanando, y los modelos ubicados después del modelo acoplado serán desplazados.

Conexiones Internas

```
IC
  {
    (13,0);(1,0)
    (12,0);(2,0)
    (8,0);(4,1)
    (11,0);(10,0)
    (2,0);(9,0)
    (2,0);(12,0)
    (2,0);(13,1)
    (10,0);(11,1)
    (10,0);(5,1)
    (1,0);(12,1)
    (1,0);(9,1)
    (6,0);(5,2)
    (6,0);(4,0)
    (7,0);(5,0)
    (7,0);(4,2)
  }
```

```
IC
  {
    (21,0);(1,0)
    (20,0);(2,0)
    (16,0);(12,1)
    (19,0);(18,0)
    (2,0);(17,0)
    (2,0);(20,0)
    (2,0);(21,1)
    (18,0);(19,1)
    (18,0);(13,1)
    (1,0);(20,1)
    (1,0);(17,1)
    (14,0);(13,2)
    (14,0);(12,0)
    (15,0);(13,0)
    (15,0);(12,2)
  }
```

Conexiones Internas del Modelo Acoplado

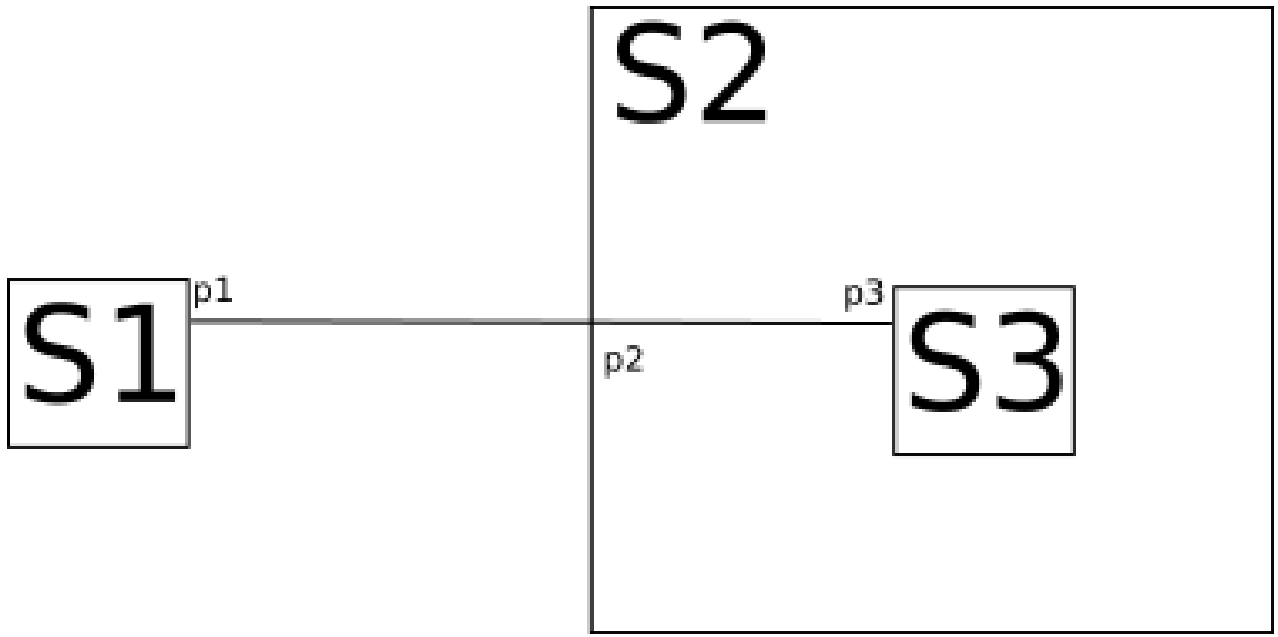
- Agregamos las conexiones internas del modelo acoplado que eliminaremos al modelo acoplado `Root-Coordinator`, estas conexiones deben ser modificadas ya que los modelos atómicos insertados son insertados en la posición del modelo acoplado eliminado.

Conexiones Internas del Modelo Acoplado

```
IC
 {
  (0,0);(1,0)
  (7,0);(8,0)
  (2,0);(3,0)
  (3,0);(4,0)
  (5,0);(6,0)
  (4,0);(8,1)
  (1,0);(2,0)
  (1,0);(7,1)
  (8,0);(6,1)
 }
```

```
IC
 {
  (3,0);(4,0)
  (10,0);(11,0)
  (5,0);(6,0)
  (6,0);(7,0)
  (8,0);(9,0)
  (7,0);(11,1)
  (4,0);(5,0)
  (4,0);(10,1)
  (11,0);(9,1)
 }
```

Conexiones Externas del Modelo Acoplado



Conexiones Externas del Modelo Acoplado

Si consideramos que $p2$ es un puerto de entrada las conexiones serán:

- IC : `(S1,p1);(S2,p2)`
- EIC : `(0,p2);(S3,p3)`

entonces la conexiones en el modelo aplanado es $(S1, p1);(S3, p3)$ y $S1$ debe ser desplazado según la cantidad de modelos que contiene $S2$ y $S3$ deberá ser desplazado según la posición de $S2$.
Si consideramos que $p2$ es un puerto de salida las conexiones serán:

- IC : `(S2,p2);(S1,p1)`
- EOC : `(S3,p3);(0,p2)`

entonces la conexiones en el modelo aplanado es $(S3, p3);(S1, p1)$ e igual que en el caso anterior, $S1$ debe ser desplazado según la cantidad de modelos que contiene $S2$ y $S3$ deberá ser desplazado según la posición de $S2$.

Aplanado de Conexiones Externas de Salida

Acoplado

```
EOC
 {
  (6,0);(0,1)
  (8,0);(0,0)
 }
```

Modelo

```
IC
 {
  (3,1);(11,2)
  (3,0);(11,0)
  (3,0);(13,0)
 }
```

Resultado

```
IC
 {
  (9,0);(19,2)
  (11,0);(19,0)
  (11,0);(21,0)
 }
```

Aplanado de Conexiones Externas de Entrada

Modelo

```
IC
 {
  (4,0);(3,0)
  (5,0);(3,1)
  (1,0);(3,2)
 }
```

Acoplado

```
EIC
 {
  (0,0);(2,1)
  (0,0);(0,1)
  (0,1);(3,1)
  (0,1);(5,0)
  (0,1);(7,0)
  (0,1);(0,0)
  (0,2);(4,1)
 }
```

Resultado

```
IC
 {
  (12,0);(5,1)
  (12,0);(3,1)
  (13,0);(6,1)
  (13,0);(8,0)
  (13,0);(10,0)
  (13,0);(3,0)
  (1,0);(7,1)
 }
```

Modelos Vectoriales

- `annotation(PD2MO = {Scalar, Scalar});` entrada y salida son escalares, este es el caso por omisión y no es necesario declararlo.

Modelos Vectoriales

- `annotation(PD2MO = {Scalar, Scalar});` entrada y salida son escalares, este es el caso por omisión y no es necesario declararlo.
- `annotation(PD2MO = {Scalar, Vector});` entrada escalar y salida vectorial

Modelos Vectoriales

- `annotation(PD2MO = {Scalar, Scalar});` entrada y salida son escalares, este es el caso por omisión y no es necesario declararlo.
- `annotation(PD2MO = {Scalar, Vector});` entrada escalar y salida vectorial
- `annotation(PD2MO = {Vector, Scalar});` entrada escalar y salida vectorial

Modelos Vectoriales

- `annotation(PD2MO = {Scalar, Scalar});` entrada y salida son escalares, este es el caso por omisión y no es necesario declararlo.
- `annotation(PD2MO = {Scalar, Vector});` entrada escalar y salida vectorial
- `annotation(PD2MO = {Vector, Scalar});` entrada escalar y salida vectorial
- `annotation(PD2MO = {Vector, Vector});` entrada y salida vectoriales.

Transformaciones Extra

- Causalización de variables

Transformaciones Extra

- Causalización de variables
- Transformaciones de Modelica a μ-Modelica

Resultados

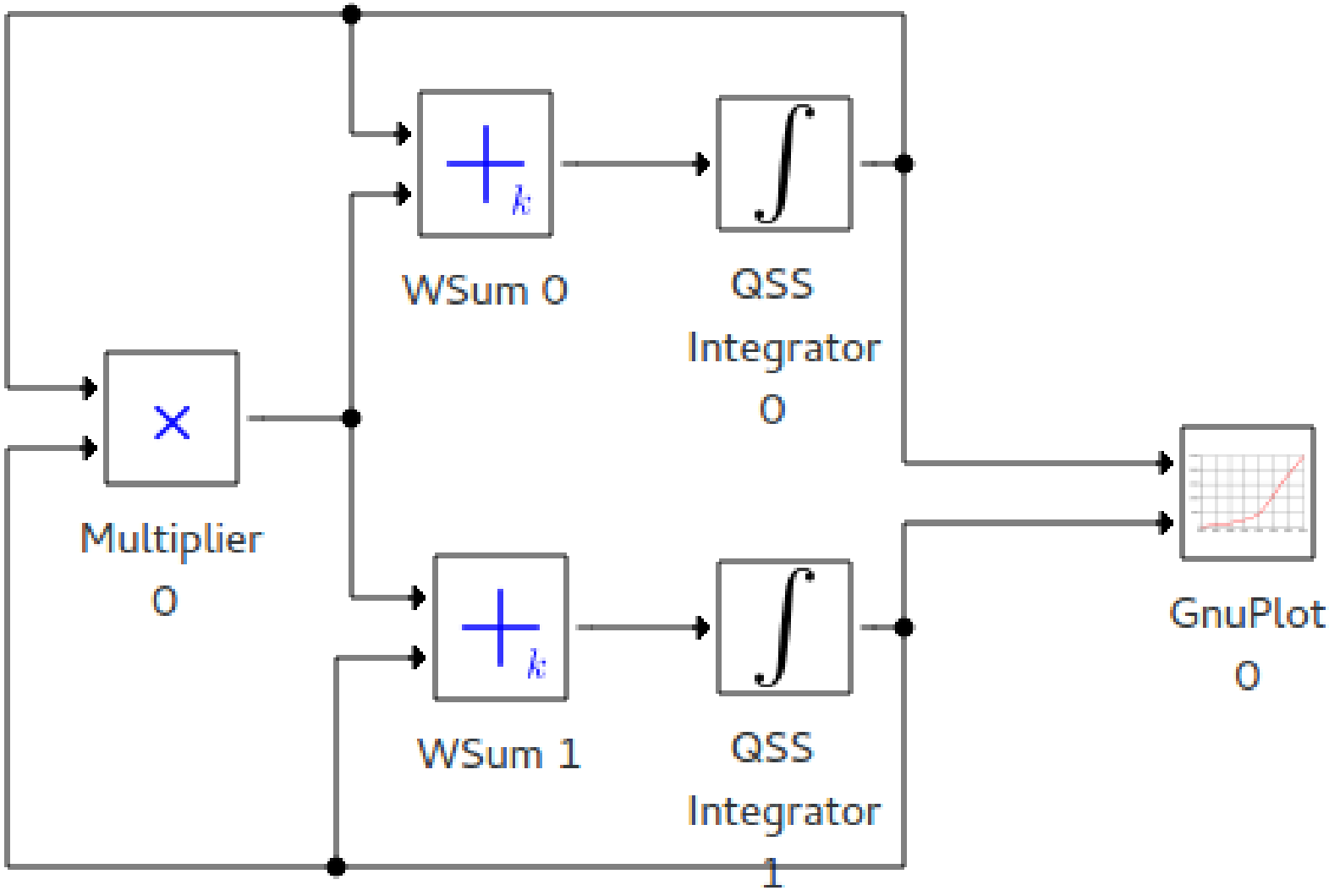
Resultados - Lotka-Volterra

$$\frac{dx}{dt} = x(\alpha - \beta y)$$
$$\frac{dy}{dt} = y(\gamma - \delta x)$$

donde:

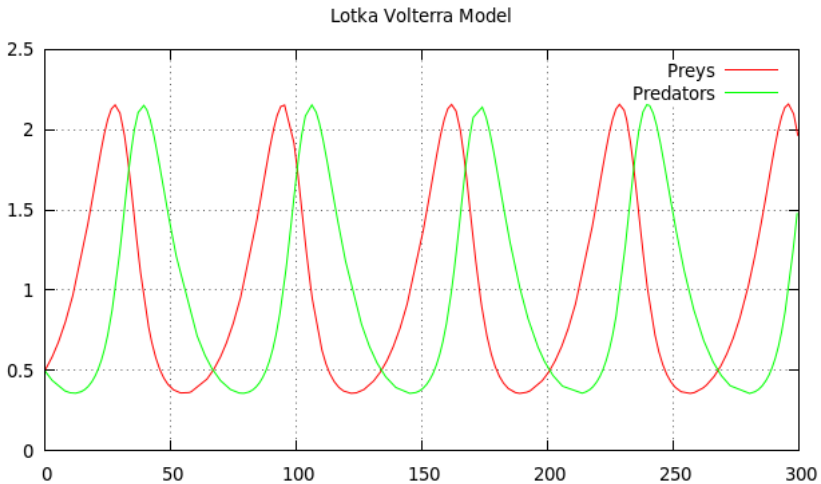
- y es el número de algún predador (por ejemplo, un lobo);
- x es el número de sus presas (por ejemplo, conejos);
- t representa el tiempo; y
- α, β, γ, δ son parámetros que representan las interacciones de las dos especies.

Resultados - Lotka-Volterra PowerDEVS

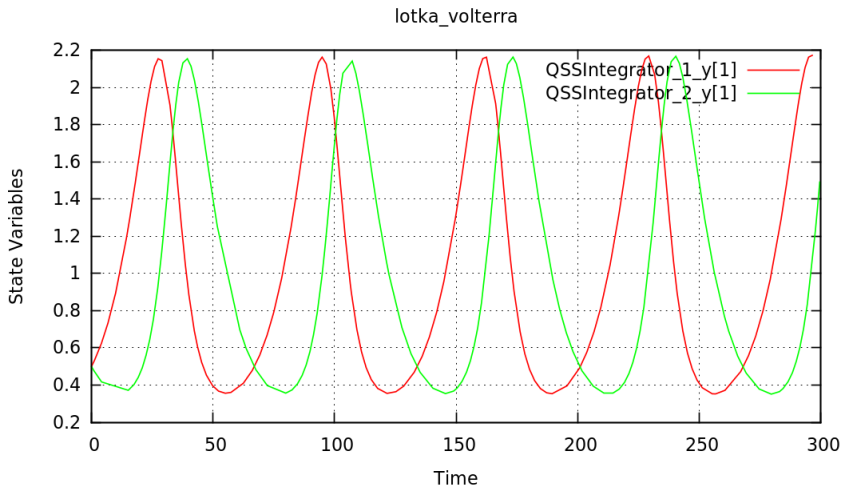


Resultados - Lotka-Volterra

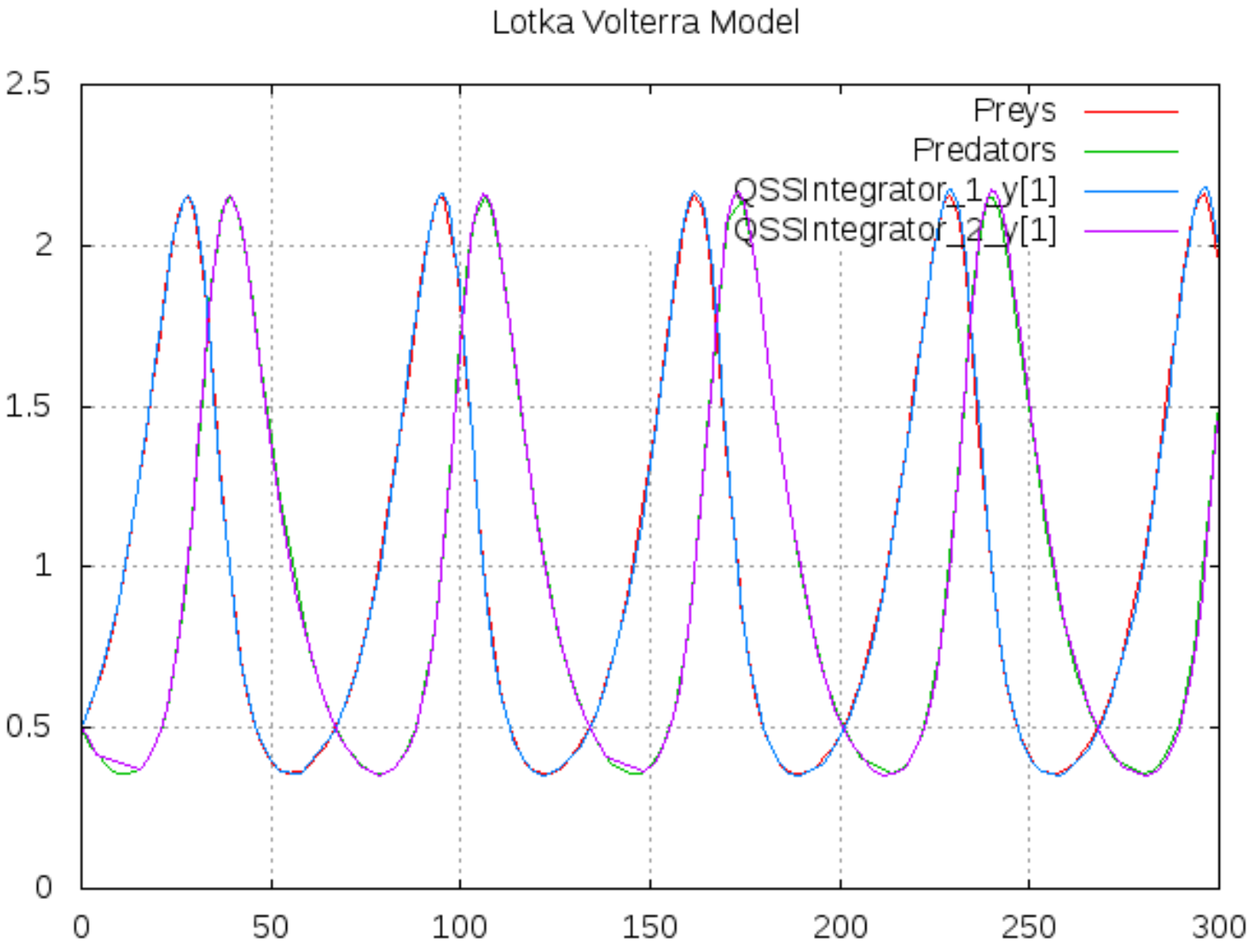
PowerDEVS



QSS-Solver



Resultados - Lotka-Volterra



Resultados - Líneas de Transmisión

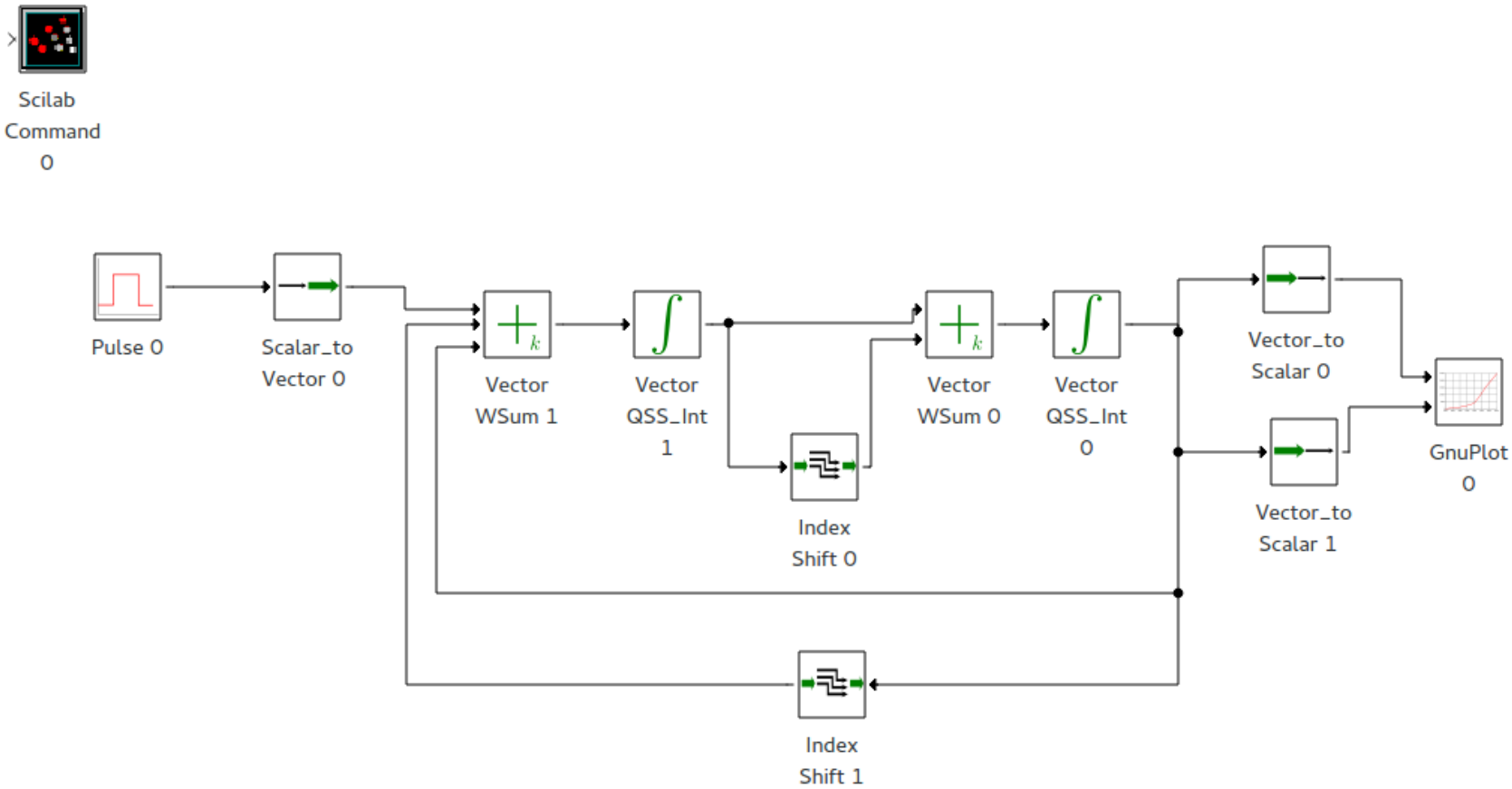
$$\frac{dv_j}{dt} = \frac{i_j - i_{j+1}}{C}$$
$$\frac{di_j}{dt} = \frac{v_{j-1} - v_j}{L}$$

para $j = 2 \ldots N$

Consideramos un pulso de entrada:

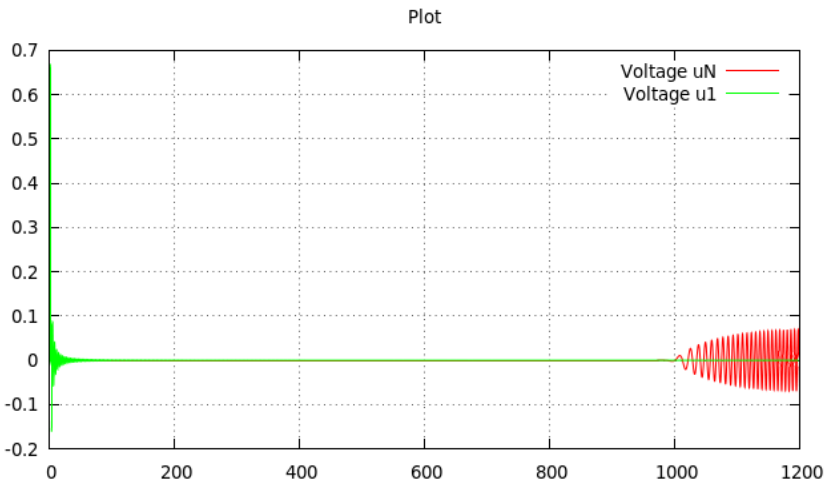
$$v_0(t) = \begin{cases} 1 \text{ si } t < 1 \\ 0 \text{ en caso contrario} \end{cases}$$

Resultados - Modelo Líneas de Transmisión en PowerDEVS

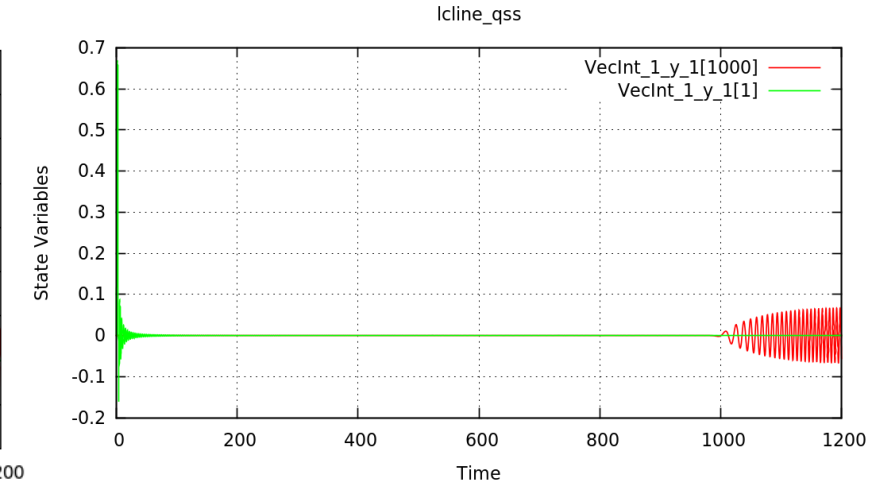


Resultados - Líneas de Transmisión

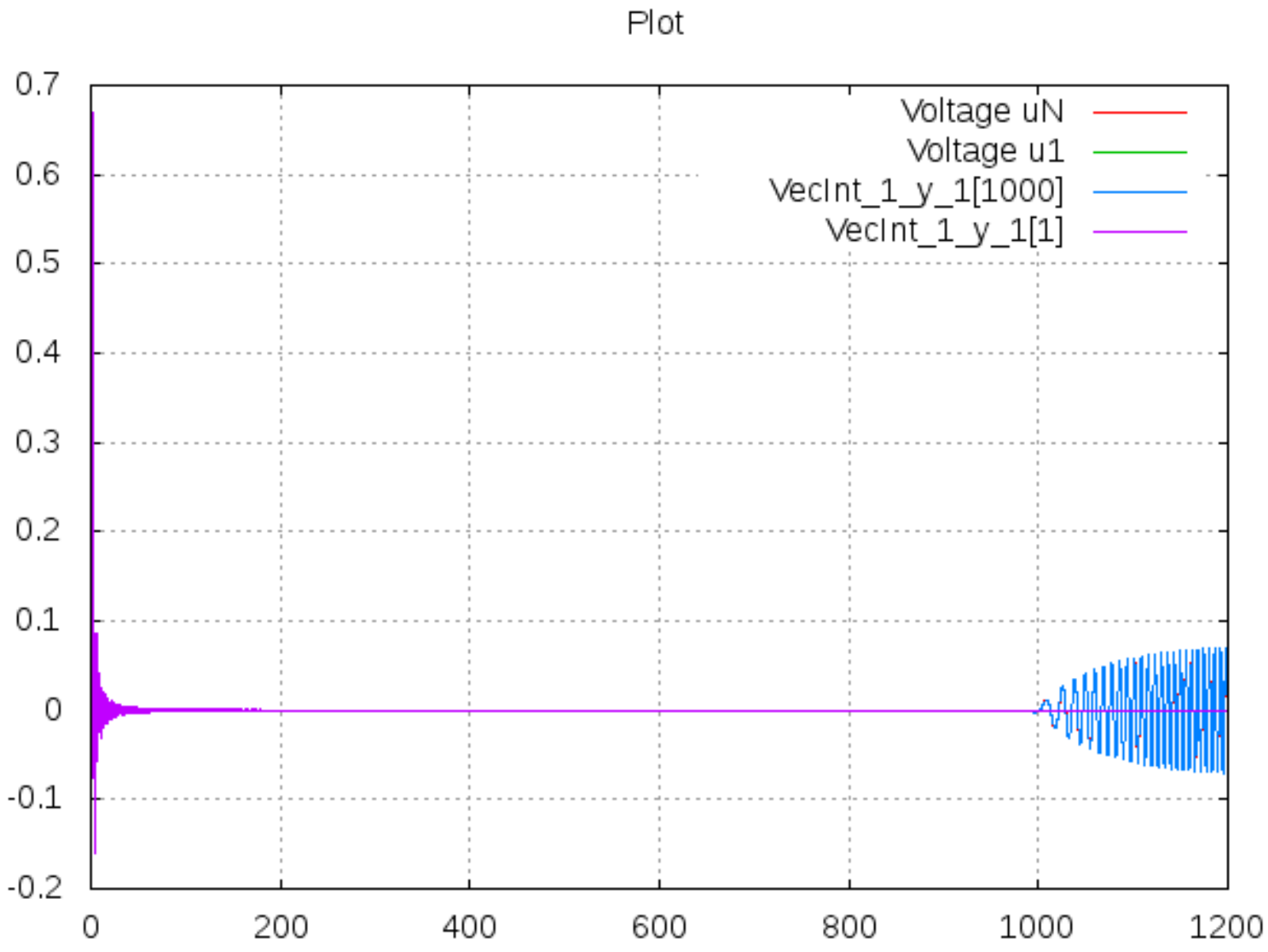
PowerDEVS



QSS-Solver



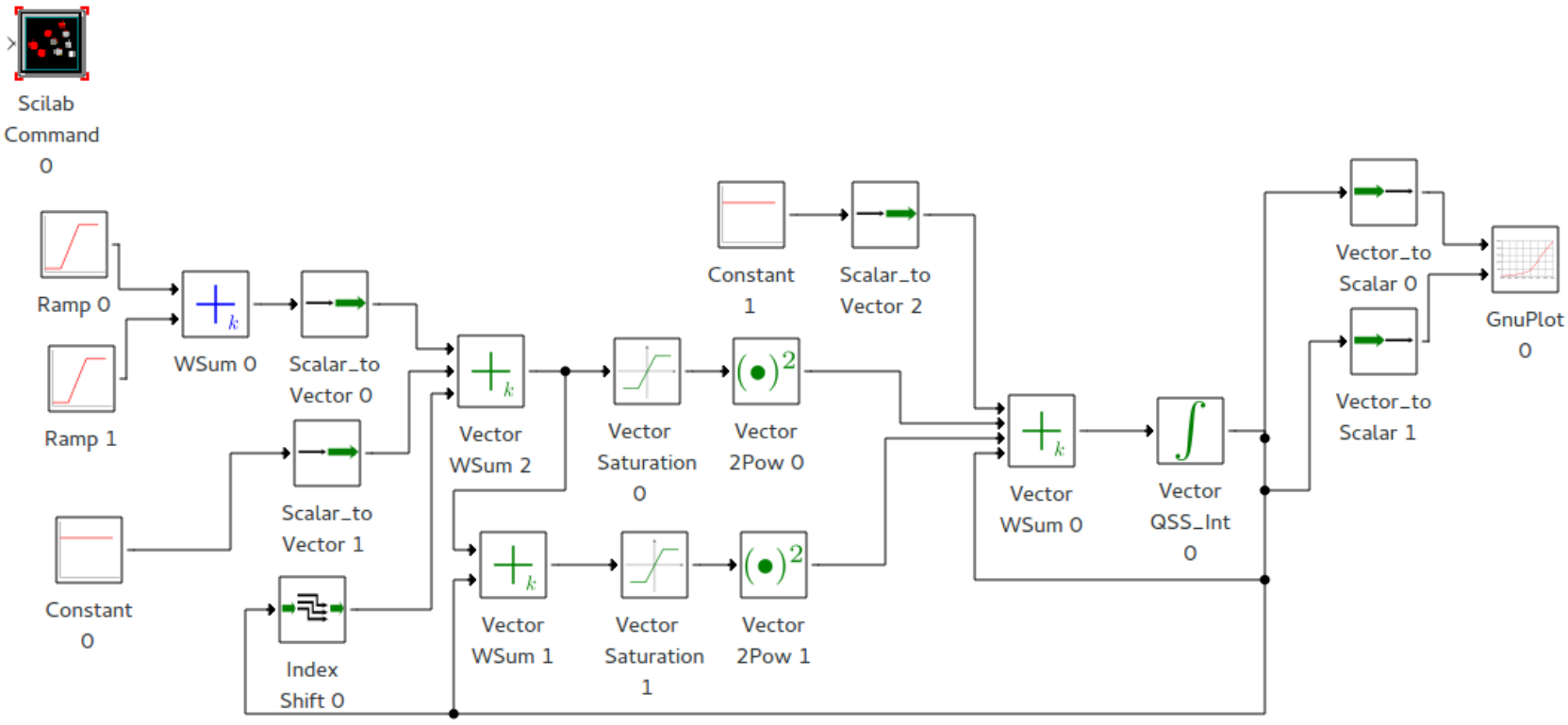
Resultados - Líneas de Transmisión



Resultados - Inversores Lógicos

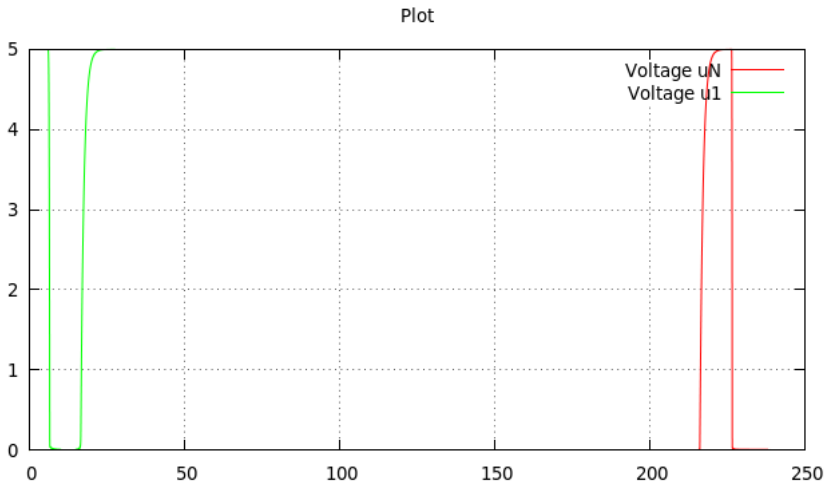
$$\frac{d\omega_1}{dt} = U_{op} - \omega_1(t) - \Upsilon g(u_{in}(t), \omega_1(t))$$
$$\frac{d\omega_j}{dt} = U_{op} - \omega_j(t) - \Upsilon g(\omega_{j-1}(t), \omega_j(t)) \text{ donde } j = 2, 3, .., m$$

Resultados - Inversores Lógicos

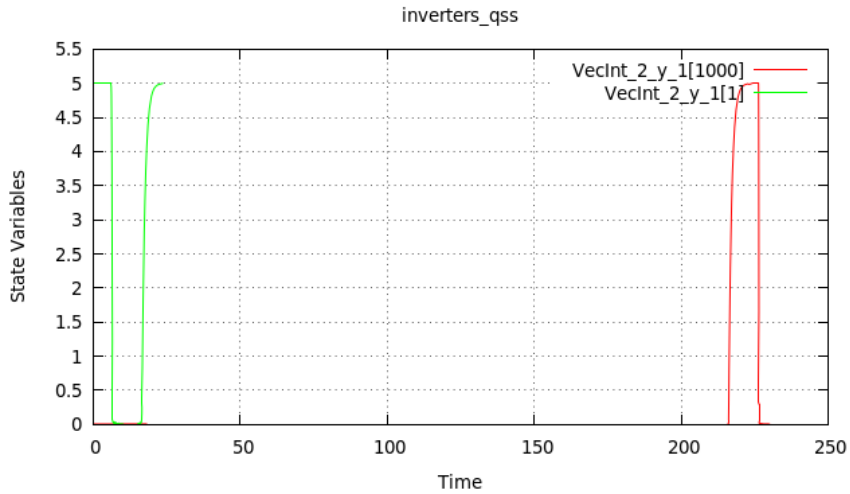


Resultados - Inversores Lógicos

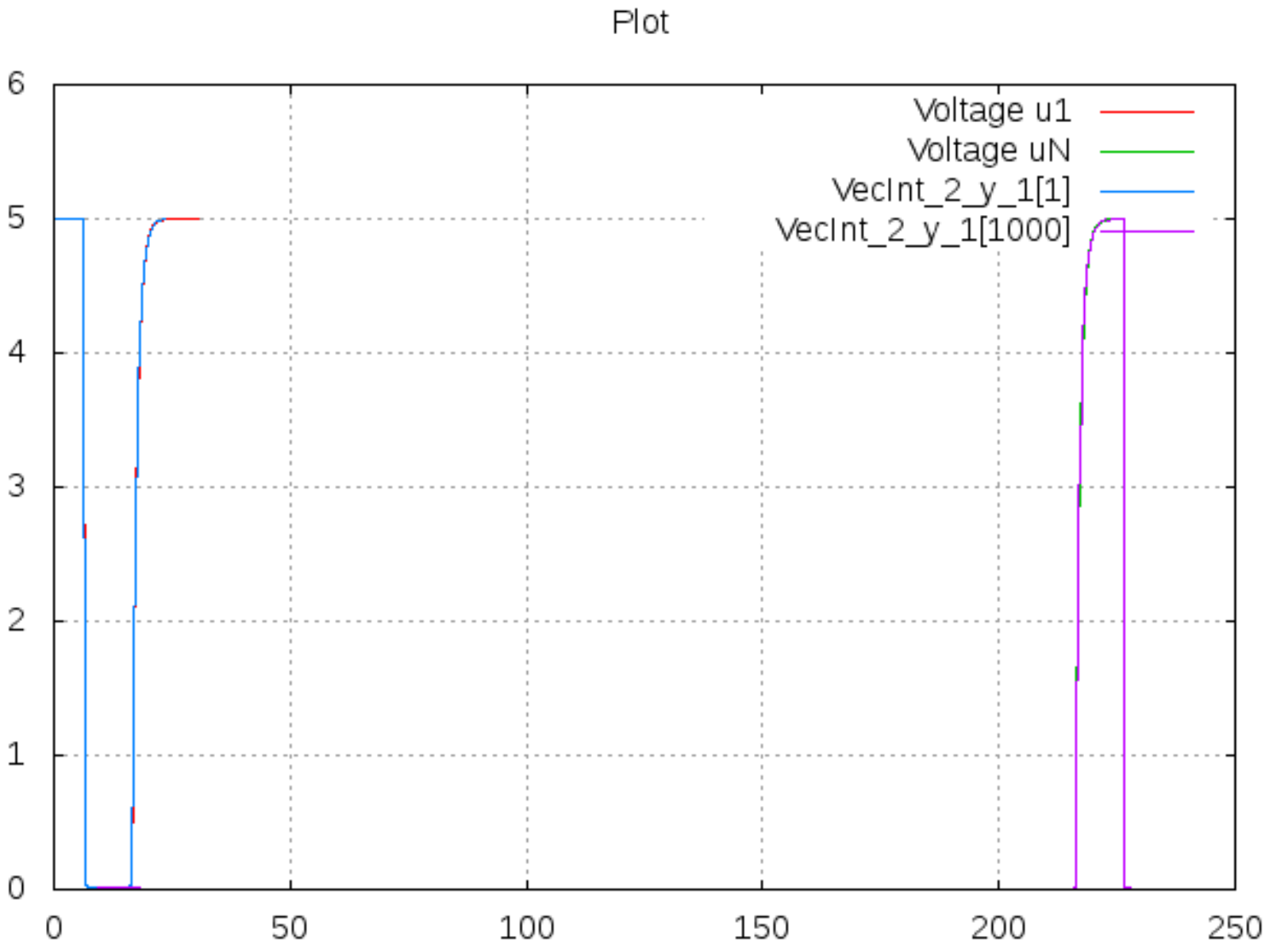
PowerDEVS



QSS-Solver



Resultados - Inversores Lógicos

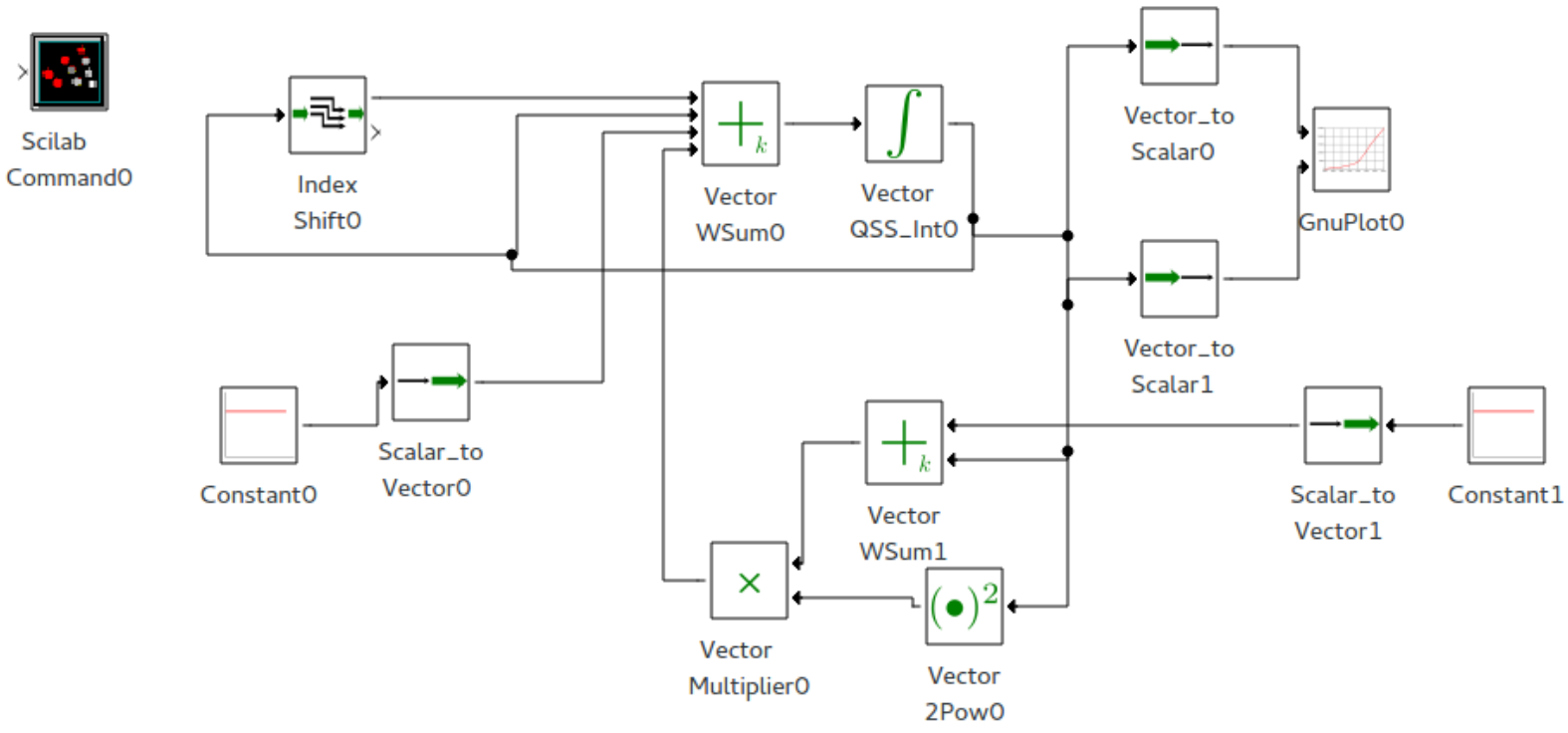


Resultados - Advection-Diffusion-Reaction (ADR)

$$\frac{du(x,t)}{dt} + a\frac{du(x,t)}{dx} = d\frac{d^2u(x,t)}{d^2x} + r(u(x,t)^2 - u(x,t)^3)$$

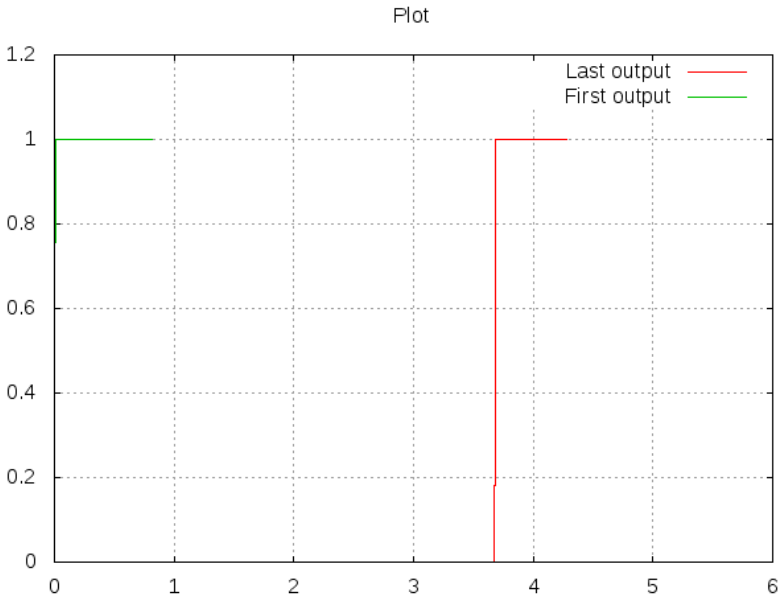
corresponde al modelo ADR, donde a,d y r son parámetros expresando coeficientes de advección, difusión y reacción, el cual es aproximado mediante el método de las lineas [10] en el modelo de la figura **??**

Resultados - Modelo ADR PowerDEVS

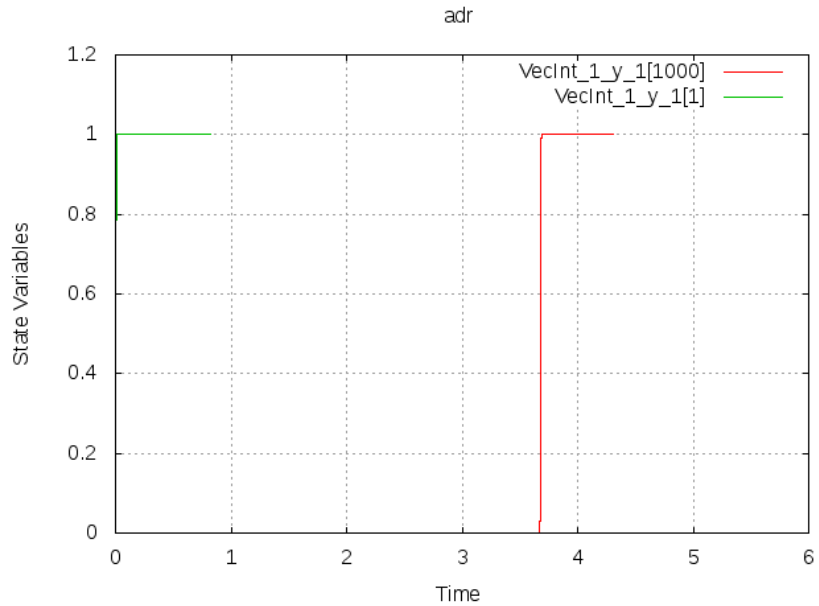


Resultados - Modelo ADR

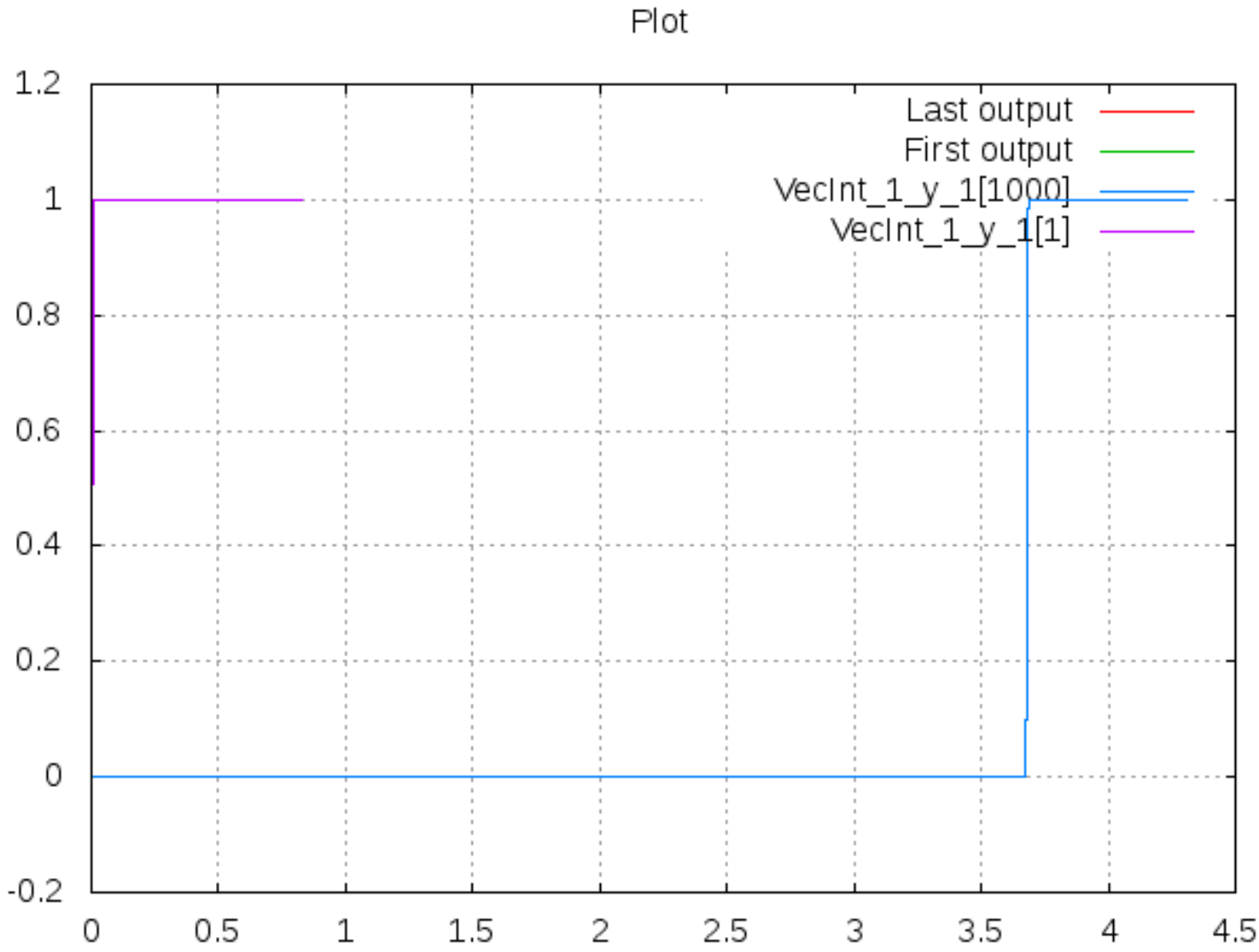
PowerDEVS



QSS-Solver



Resultados - Modelo ADR



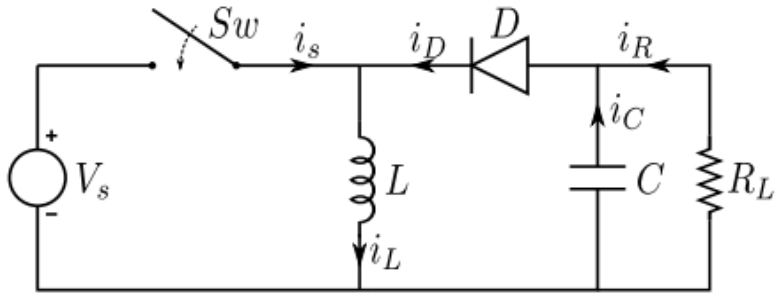
Resultados - Convertidor de Voltaje

$$\frac{di_L}{dt} = \frac{-u_C - R_D i_D}{L}$$
$$\frac{du_C}{dt} = i_D \frac{i_D}{C} - \frac{u_C}{R_L C}$$

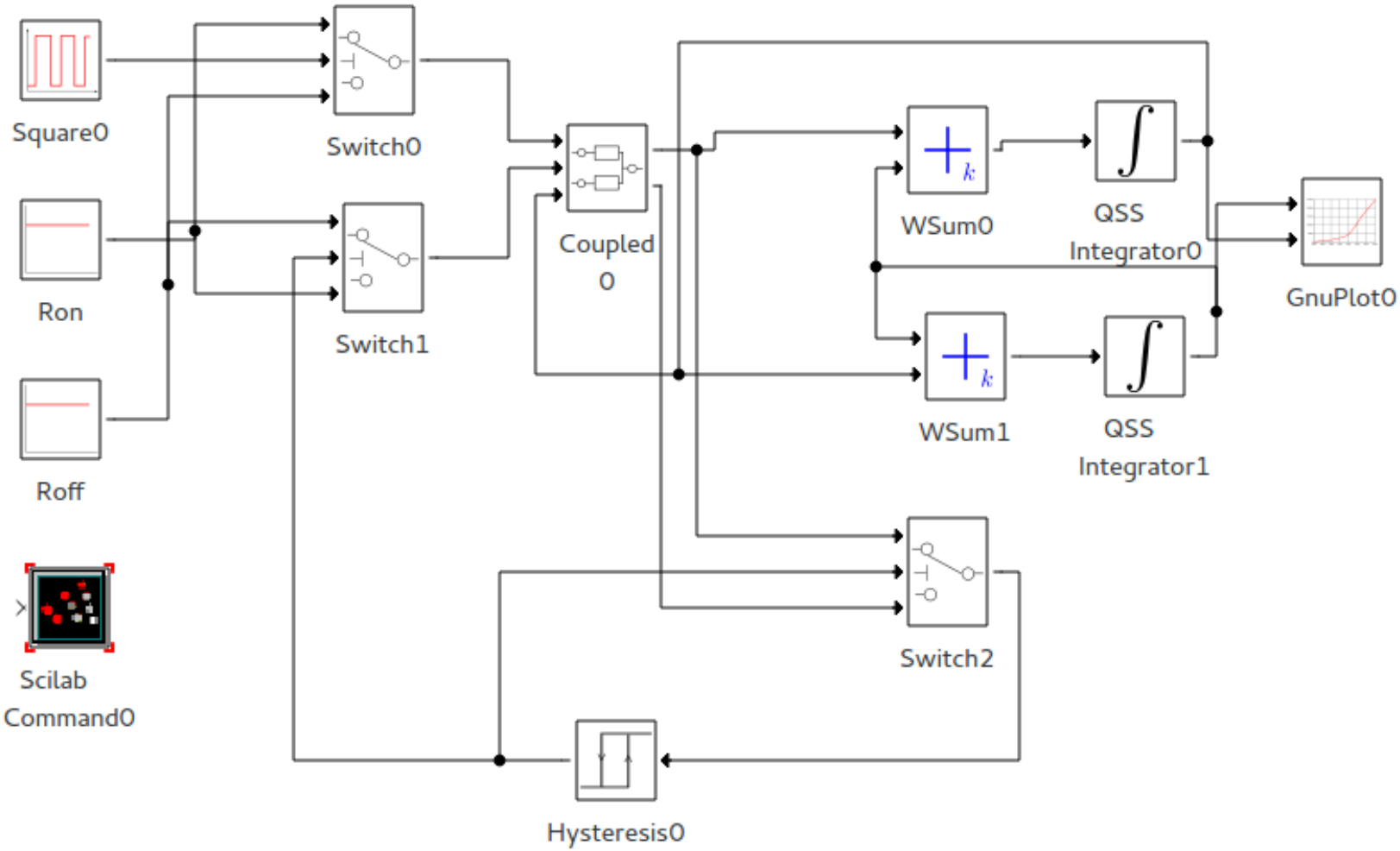
donde

$$i_D = \frac{R_s i_L - u_C - U}{R_D + R_s}$$

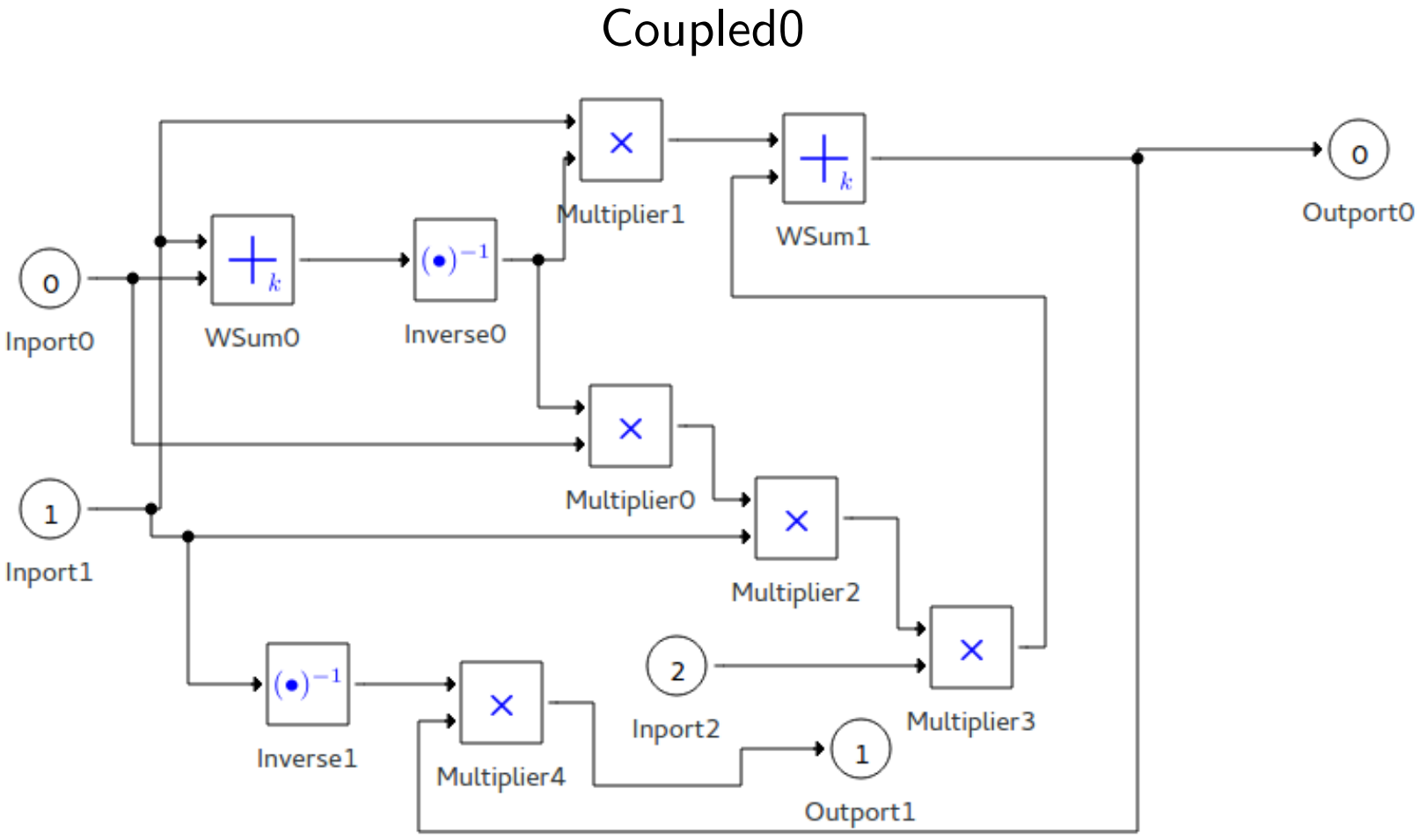
Esquema eléctrico



Resultados - Convertidor de Voltaje

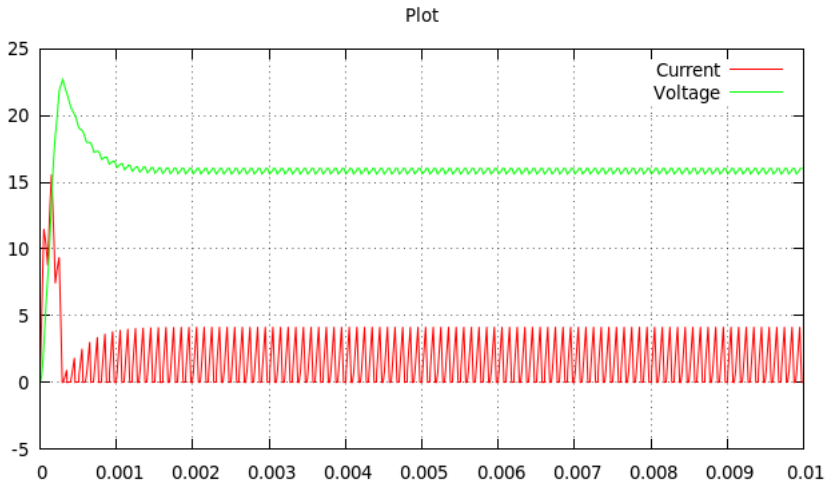


Resultados - Convertidor de Voltaje

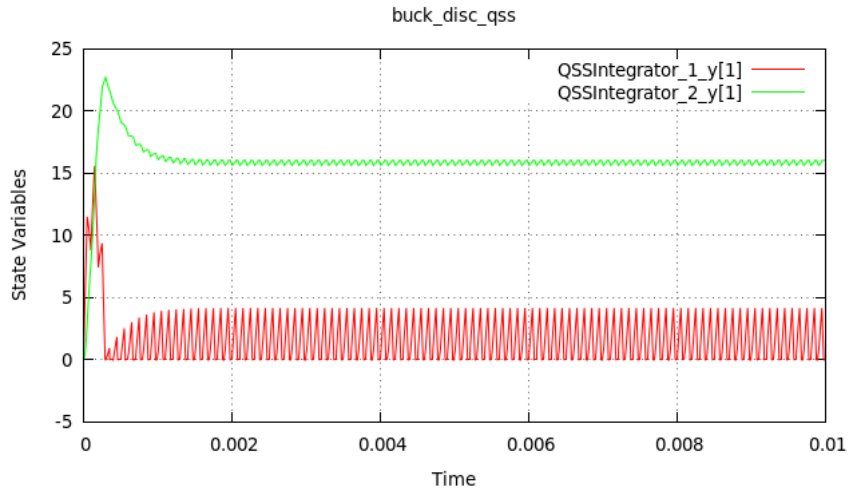


Resultados - Convertidor de Voltaje

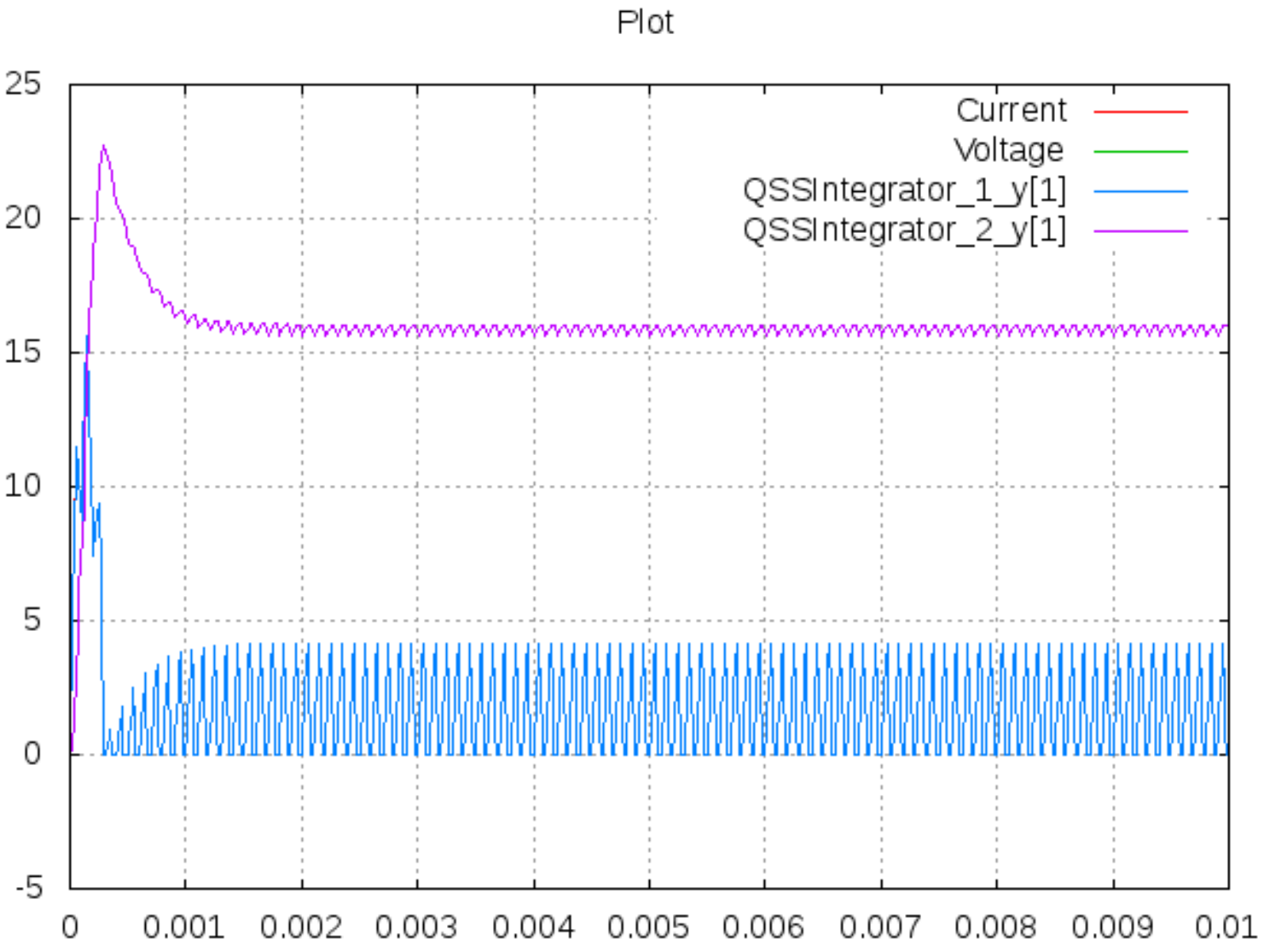
PowerDEVS



QSS-Solver



Resultados - Convertidor de Voltaje



Resultados - Resumen

Modelos	**P.DEVS(ms)**	**QSS-S. (ms)**	**Mejora (%)**
Lotka Voltera	11	2	81
Lineas de Transmisión	76402	34982	54
Inversores(LIQSS2)	25046	7694	69
ADR(LIQSS2)	6089	568	90
Convertidor de Voltaje	268	10	96

¿Preguntas?

The End