Conversión de modelos PowerDEVS al lenguaje Modelica

Luciano Andrade

Universidad Nacional de Rosario

andrade.luciano@gmail.com

19 de diciembre de 2015

Resumen

El modelado y simulación

son utilizados en el análisis de sistemas ayudándonos a ganar un mejor entendimiento de su funcionamiento.

PowerDEVS

PowerDEVS es un entorno integrado para el modelado y simulación basado en el formalismo DEVS

QSS-Solver

es una implementación independiente de los métodos de integración QSS para simulaciones de sistemas continuos e híbridos.

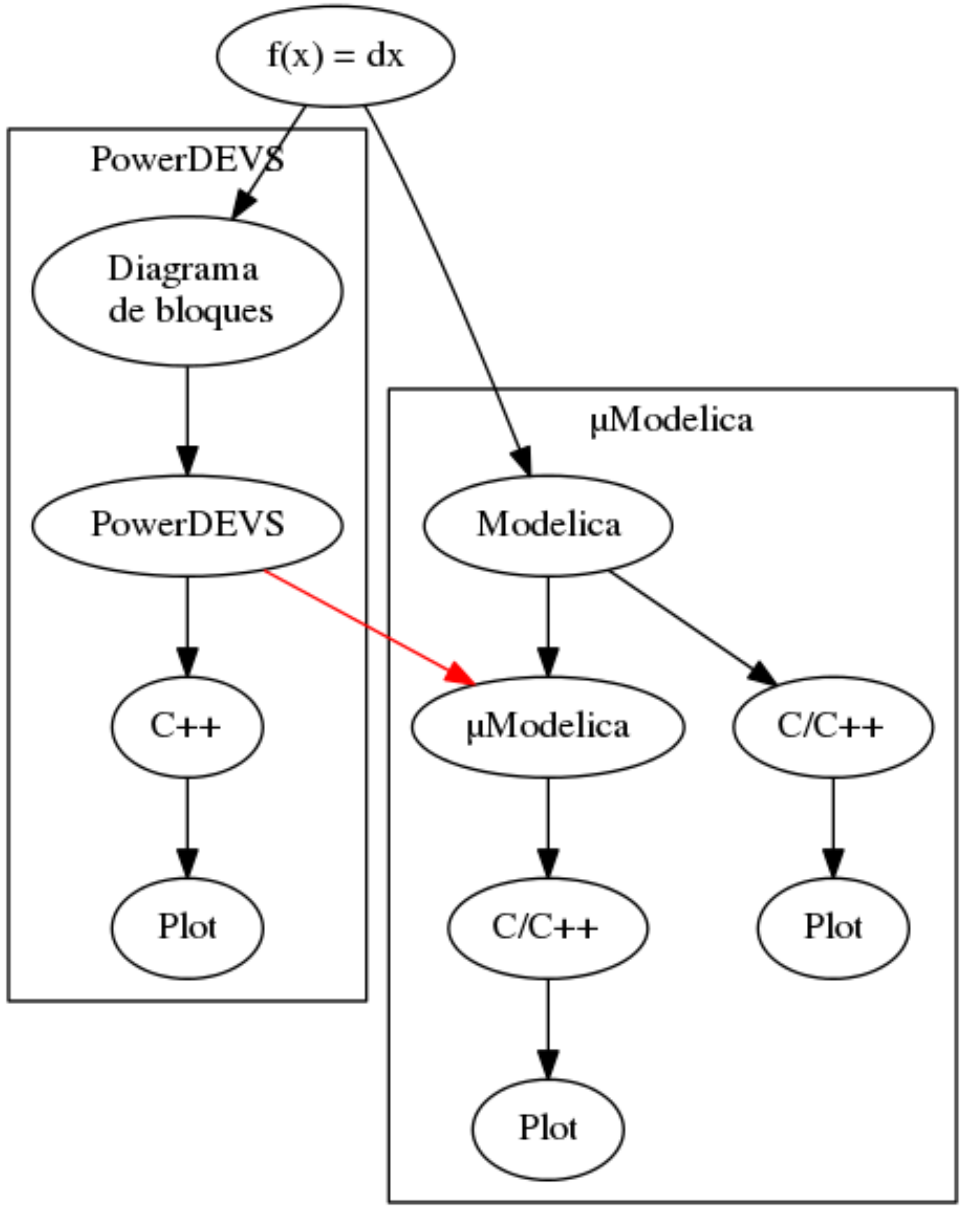
En este trabajo

se presenta una aplicación capaz de convertir modelos, en PowerDEVS en modelos Modelica, más específicamente μ-Modelica, permitiendo ejecutar este en "QSS Solver", lo cual esperamos nos permitá ganar al menos un orden de magnitud en los tiempos incurridos en la simulación.

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



El Modelado y Simulación

de un sistema es el proceso por el cual se desarrolla un modelo, el cual es luego ejecutado, de forma de obtener datos sobre el comportamiento del sistema. El modelo debe conservar las principales características del sistema, pero al mismo tiempo ser significativamente más simple, de forma que al momento de simularlo sea más eficaz utilizar la simulación que el sistema en sí.

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:

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

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

Un ejemplo - 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

Modelica

es un lenguaje orientado a objetos desarrollado para describir de manera sencilla modelos de sistemas dinámicos eventualmente muy complejos.

Modelica - Modelo Lotka Volterra

```
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;
```

Métodos de Integración QSS

El método QSS

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

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

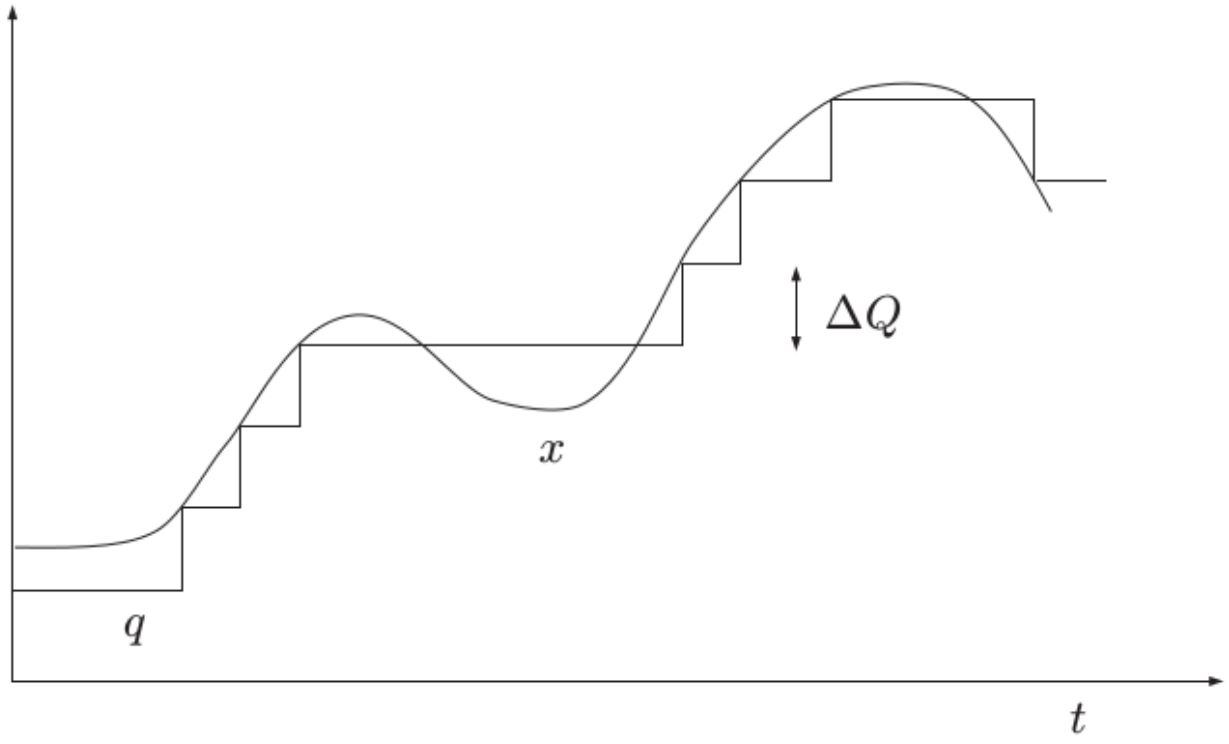
donde q es el vector de estados cuantificados y sus componentes están relacionadas una a una con las del vector de estados x siguiendo una función de cuantificación con histéresis:

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

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

La especificación de

μ-Modelica esta basado en la espeficación de Modelica y contiene solo las palabra claves y estructuras necesarias para definir modelos hibridos basados en ODE.

μ-Modelica

- 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

- 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.
- 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.

μ-Modelica

```
1  model lotka_volterra
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  initial algorithm
9    x := 0.5;
10   y := 0.5;
11 equation
12   der(x) =  x * (a - b * y);
13   der(y) =  - y * (d - c * x);
14 end lotka_volterra;
```

Stand-Alone QSS-Solver

El Stand-Alone QSS-Solver

es un entorno de modelado y simulación para sistemas hibridos y continuos.

Los modelos

son descriptos en μ-Modelica

Formalismo DEVS

DEVS

DEVS es un formalismo para modelar y analizar sistemas de eventos discretos

Un modelo DEVS

puede ser visto como un autómata que procesa una serie de eventos de entrada y genera una serie de eventos de salida y está descripto por dos clases de componentes, modelos atómicos y modelos acoplados.

Formalismo DEVS - Modelos Atómicos

Un modelo Atómico

representa la unidad "indivisible" de especificación, en el sentido que es la pieza fundamental y más básica de un modelo DEVS.

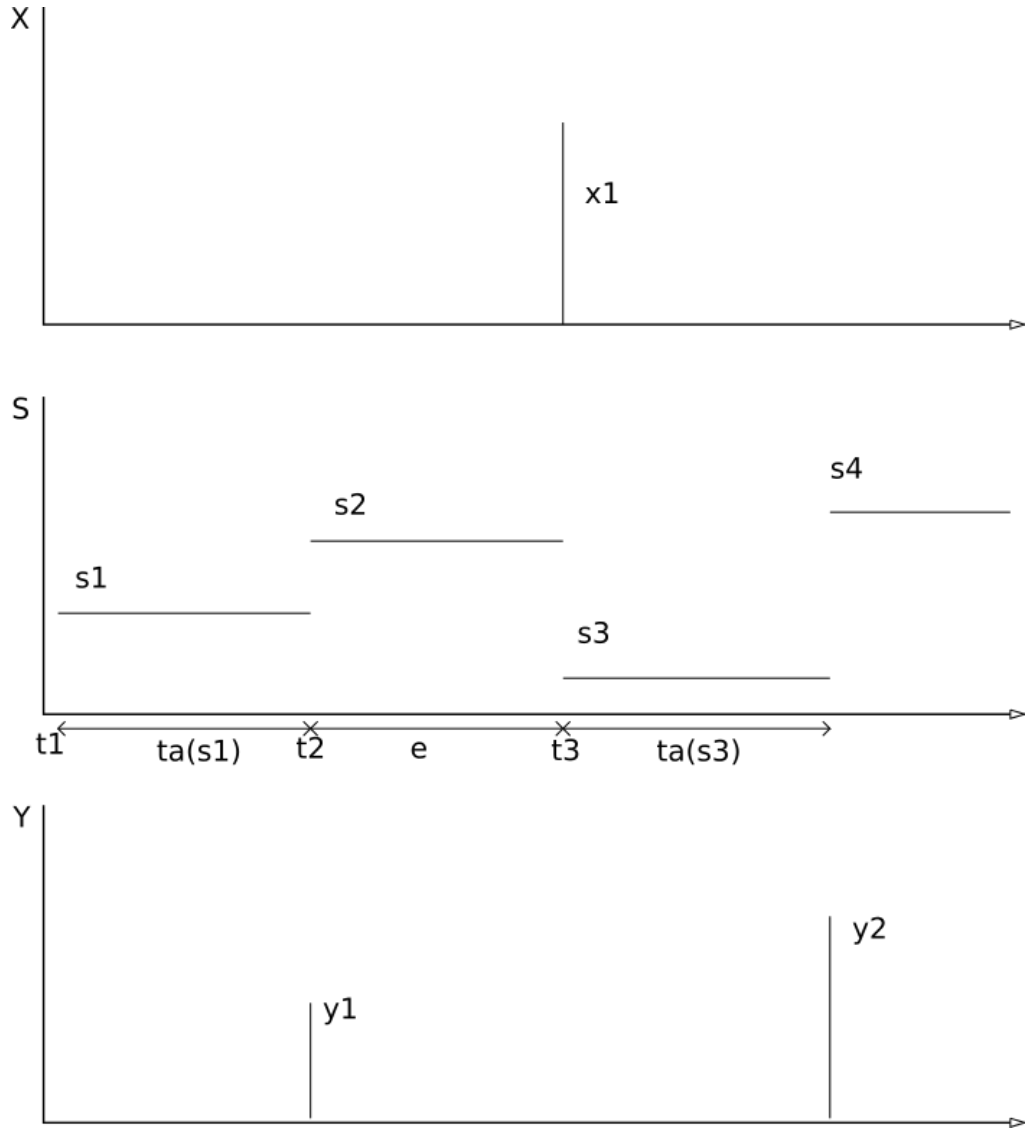
Formalismo DEVS - Modelos Atómicos

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

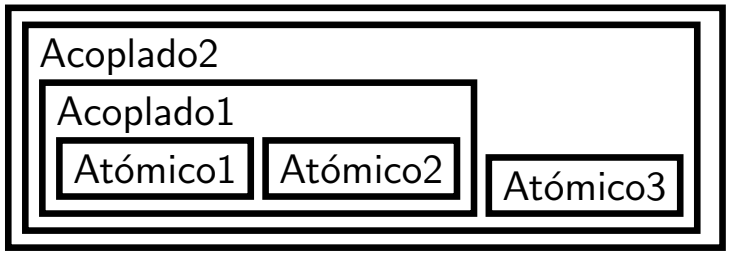
$$(X, Y, S, \delta_{int}, \delta_{ext}, \lambda, t_a) \text{ donde :}$$

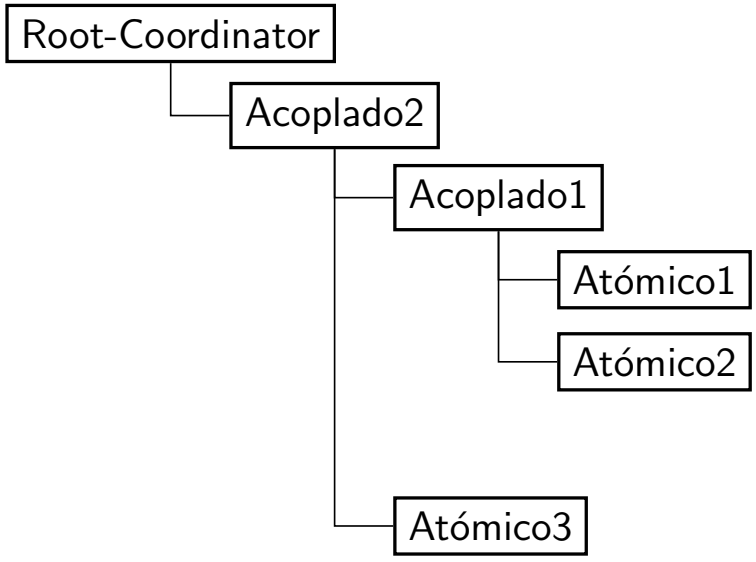
- X es el conjunto de valores de entrada que acepta el modelo atómico.
- Y es el conjunto de valores de los eventos de salida que puede emitir el modelo atómico.
- S es el conjunto de estados internos del modelo.
- ta es una función $S \rightarrow \mathbb{R}_0^+$ de *Avance de Tiempo*.
- δ_{int} es una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

Formalismo DEVS - Modelos Atómicos



Modelo Acoplados



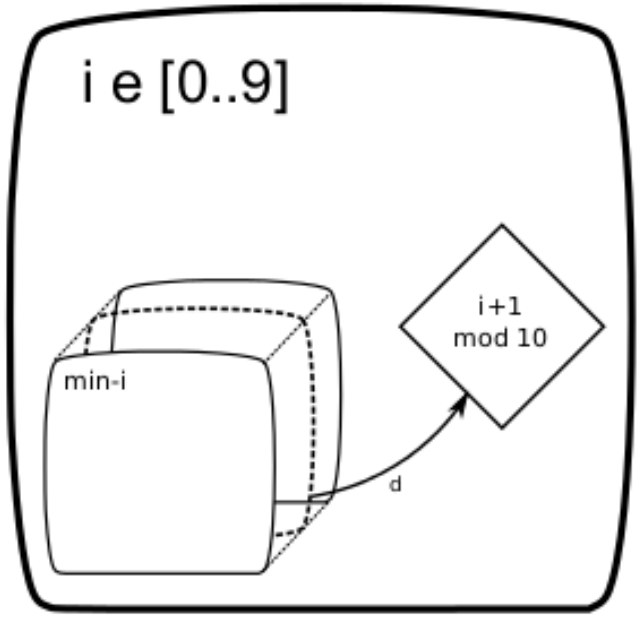


Modelos Vectoriales

Modelos Vectoriales

Nos permiten especificar un vector de Modelos y facilita el modelado de grandes sistemas.

StateChart de un modelo Vectorial

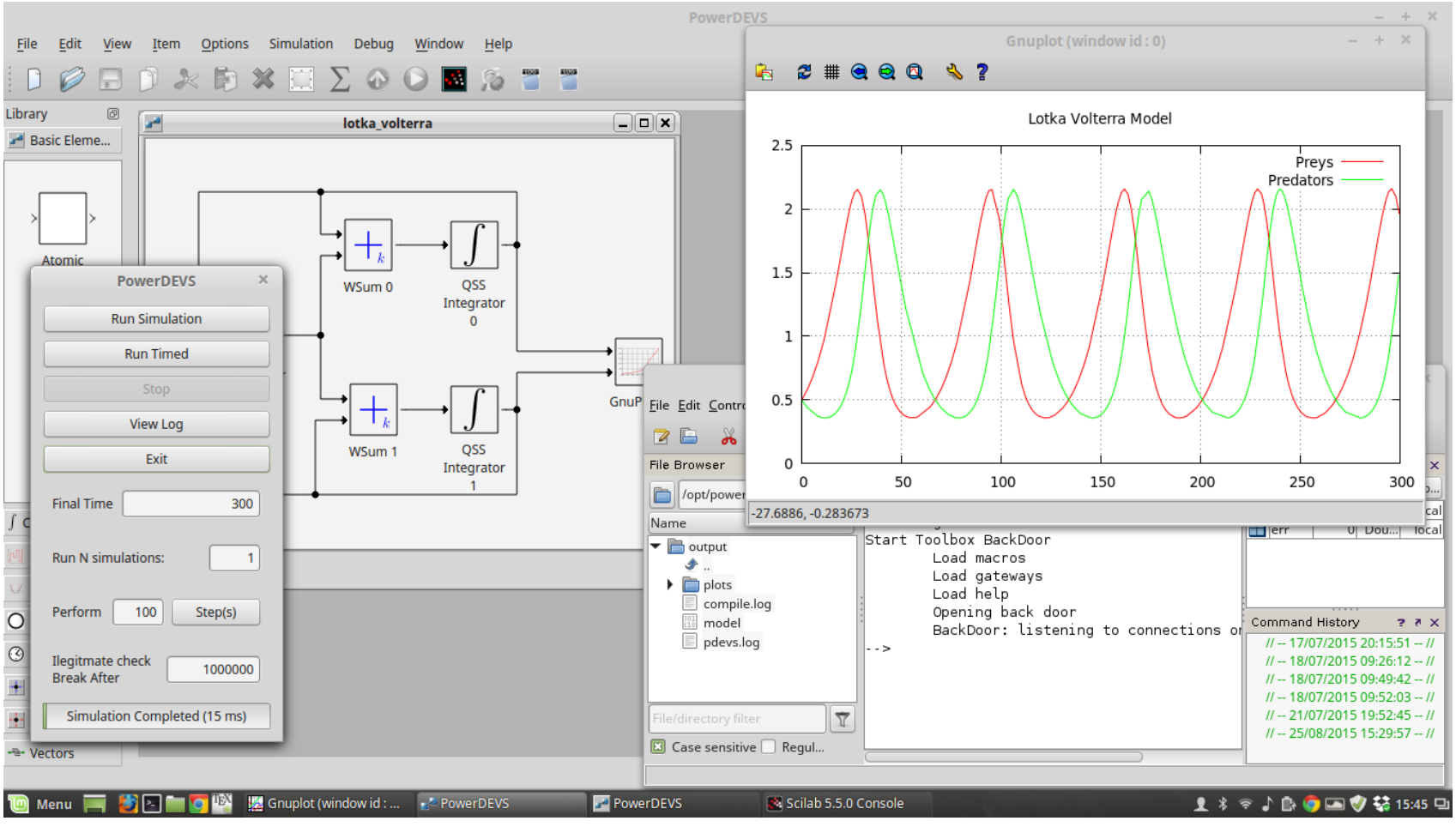


PowerDEVS

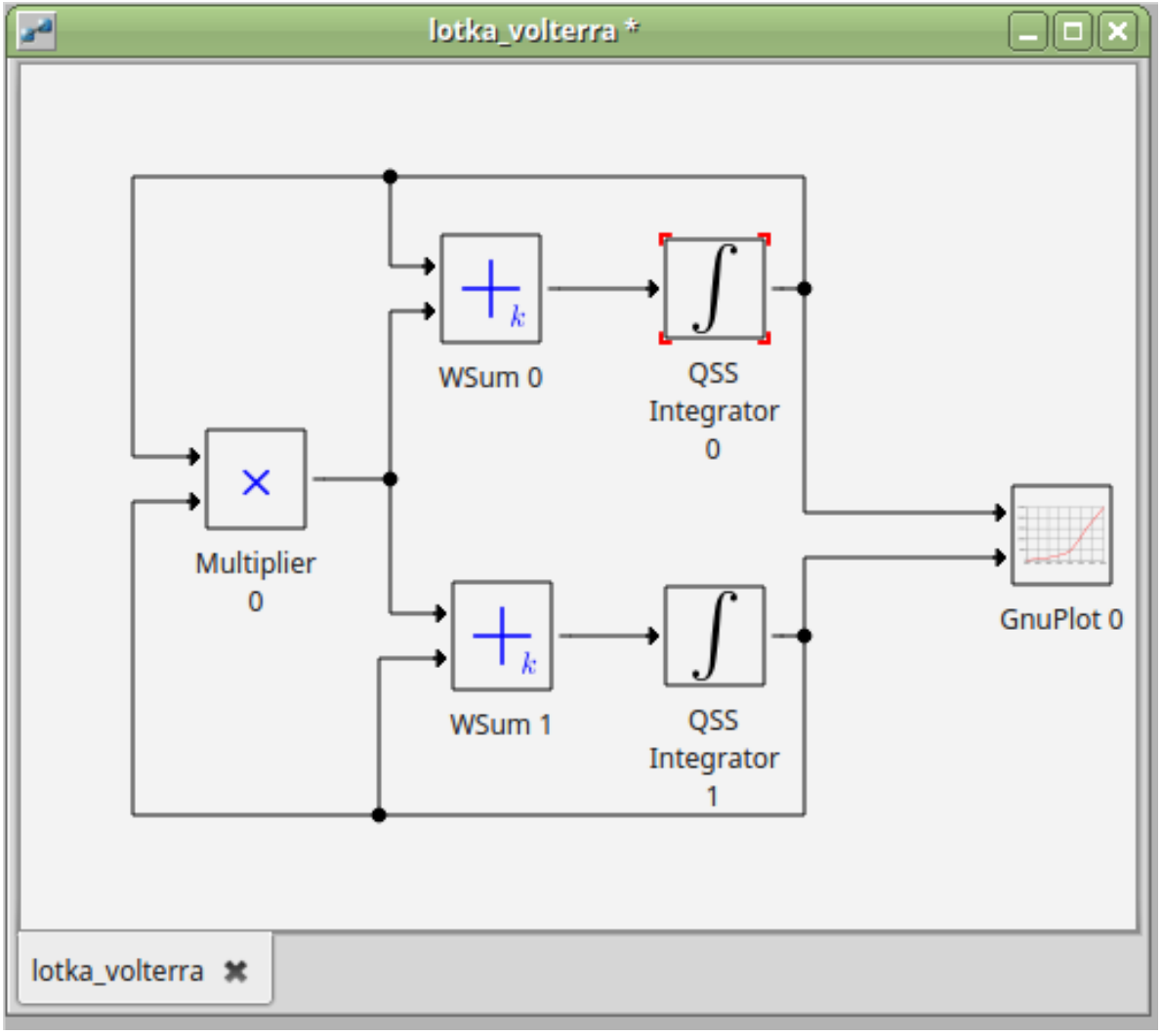
PowerDEVS

es un programa, concebido para ser utilizado por expertos programadores DEVS, así como usuarios finales que solo quieren conectar bloques y simular.

PowerDEVS


PowerDEVS
File Edit View Item Options Simulation Debug Window Help
Library
Basic Eleme...
Atomic
lotka_volterra
WSum 0
QSS Integrator 0
WSum 1
QSS Integrator 1
PowerDEVS
Run Simulation
Run Timed
Stop
View Log
Exit
Final Time 300
Run N simulations: 1
Perform 100 Step(s)
Ilegitmate check Break After 1000000
Simulation Completed (15 ms)
Vectors
Gnuplot (window id : 0)
Lotka Volterra Model
Preys
Predators
2.5
2
1.5
1
0.5
0
0 50 100 150 200 250 300
-27.6886, -0.283673
File Edit Contr
File Browser
/opt/power
Name
output
..
plots
compile.log
model
pdevs.log
File/directory filter
Case sensitive Regul...
Start Toolbox BackDoor
Load macros
Load gateways
Load help
Opening back door
BackDoor: listening to connections or
-->
Command History
// -- 17/07/2015 20:15:51 -- //
// -- 18/07/2015 09:26:12 -- //
// -- 18/07/2015 09:49:42 -- //
// -- 18/07/2015 09:52:03 -- //
// -- 21/07/2015 19:52:45 -- //
// -- 25/08/2015 15:29:57 -- //
Menu
Gnuplot (window id : ...
PowerDEVS
PowerDEVS
Scilab 5.5.0 Console
15:45

PowerDEVS - Modelo Lotka Volterra



Conversión de modelos DEVS

PowerDEVS

trabaja principalmente con dos archivos, .PDM y .PDS. Los archivos PDM son utilizados, principalmente, por el editor de modelos, contiene información estructural, de posición de los modelos, sus parámetros, nombre, tipo y valor, descripción de los modelos, la cantidad de puertos de cada modelo y las conexiones entre los modelos, los detalles de cómo esas conexiones son visualizadas por líneas y los detalles del recorrido de esas líneas. Todos estos elementos son utilizados en primera medida por el editor de modelos, pero también son utilizados para generar el archivo PDS el cual genera el código de la simulación. El archivo PDS contiene información estructural del modelo necesaria para realizar la simulación.

Archivos PDS

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

...
```

QSSIntegrator

qss/qss_integrator.mo

```
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;
```

Archivos PDS

```
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;
```

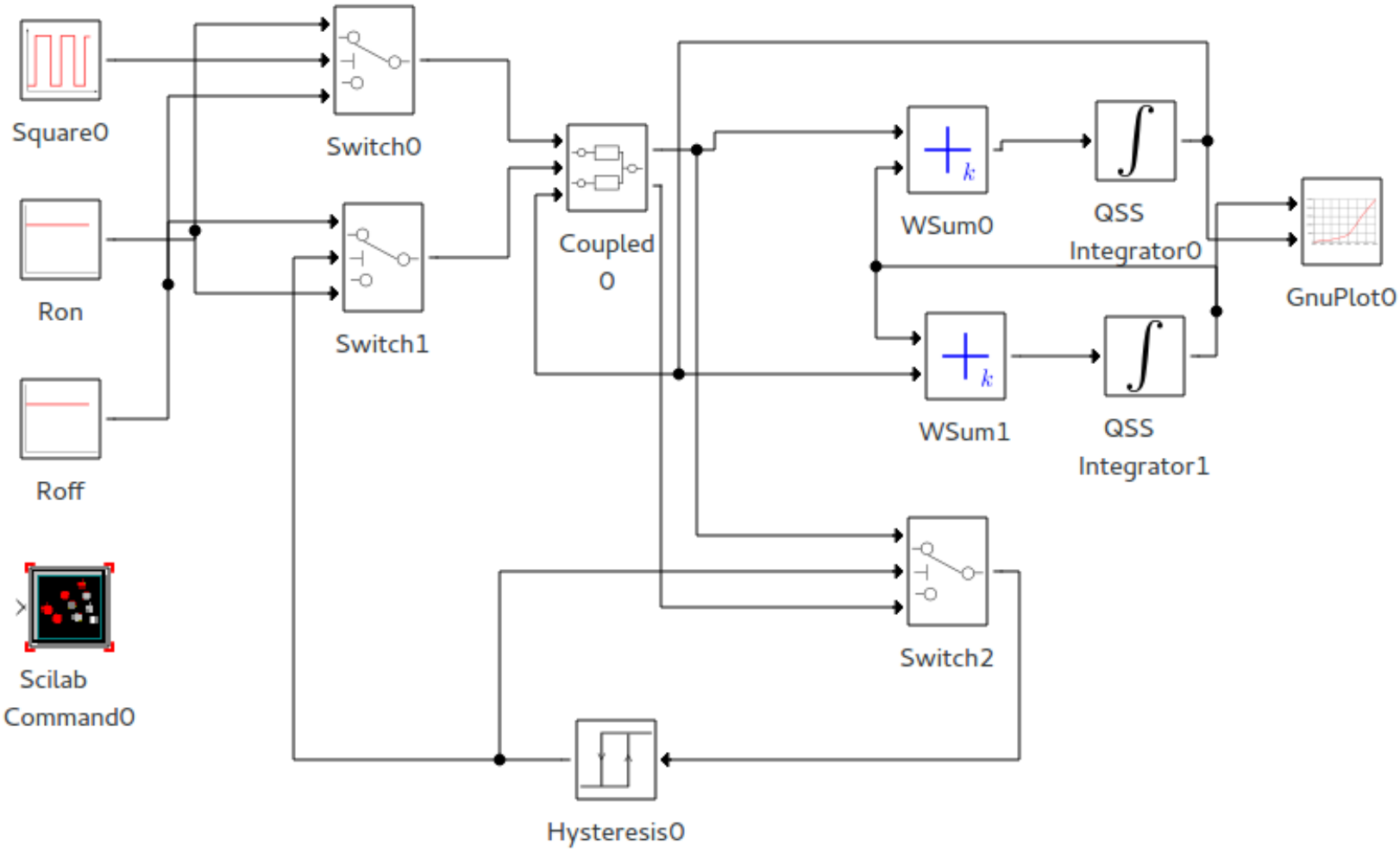
Protocolo

- 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

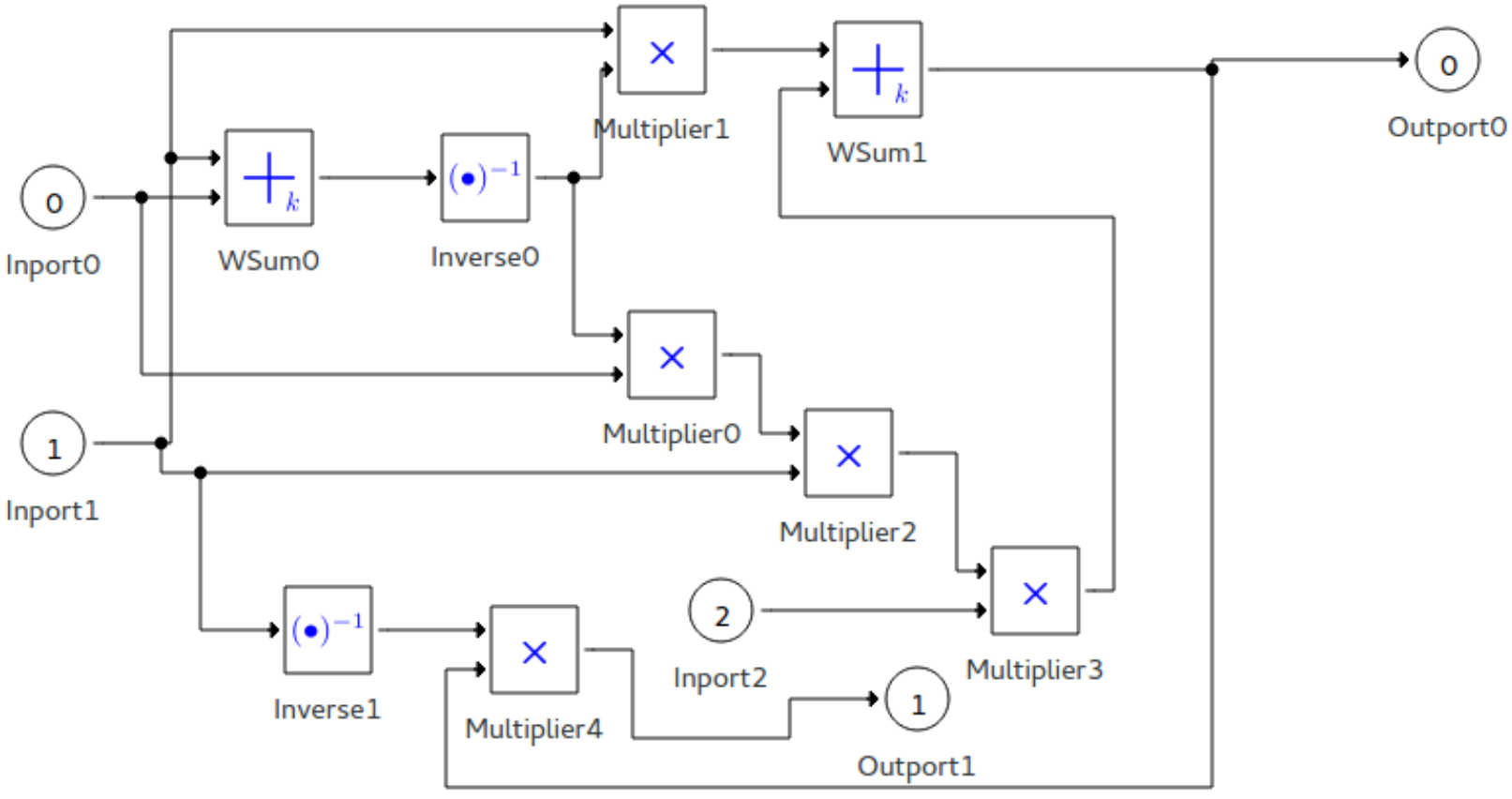
Modelos Acoplados Planos

```
model lotka_volterra
...
  Real QSSIntegrator_1_u[1];
  Real QSSIntegrator_1_y[1](start = QSSIntegrator_1_x0);
  Real QSSIntegrator_2_u[1];
  Real QSSIntegrator_2_y[1](start = QSSIntegrator_2_x0);
...
equation
...
  qss_multiplier_0_u[1] = QSSIntegrator_1_y[1];
  qss_multiplier_0_u[2] = QSSIntegrator_2_y[1];
  WSum_3_u[1] = QSSIntegrator_1_y[1];
  WSum_4_u[2] = QSSIntegrator_2_y[1];
  der(QSSIntegrator_2_y[1]) = QSSIntegrator_2_u[1];
  der(QSSIntegrator_1_y[1]) = QSSIntegrator_1_u[1];
...
end lotka_volterra;
```

Ejemplo de modelo acoplado - Convertidor de Voltaje

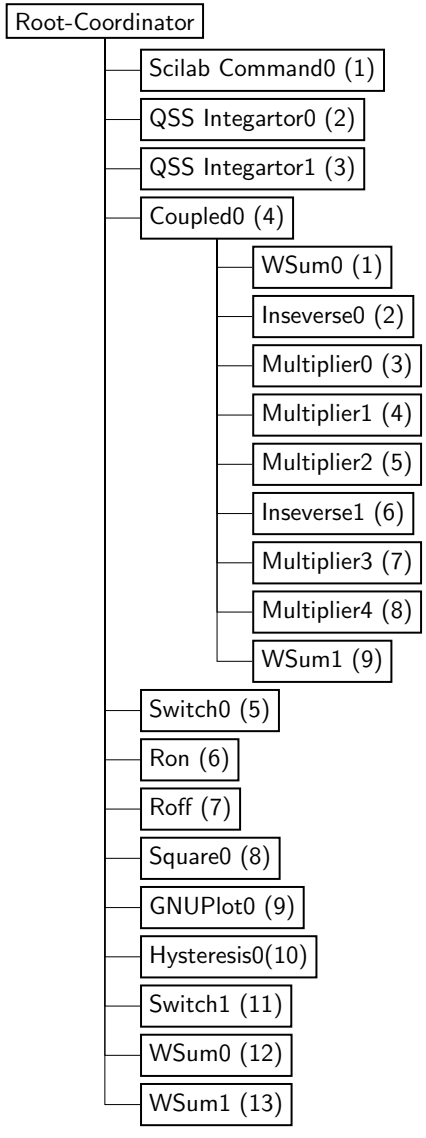


Ejemplo de modelo acoplado - Convertidor de Voltaje - Coupled0

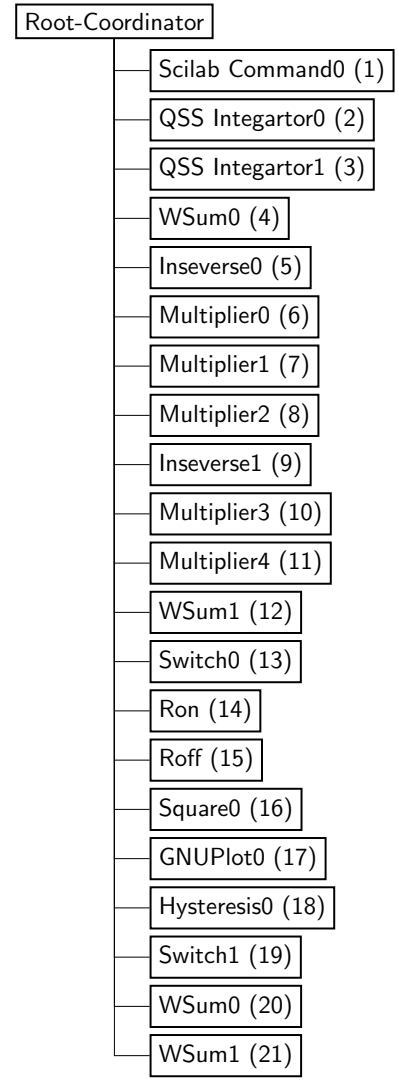


Convertidor de Voltaje

Jerárquico

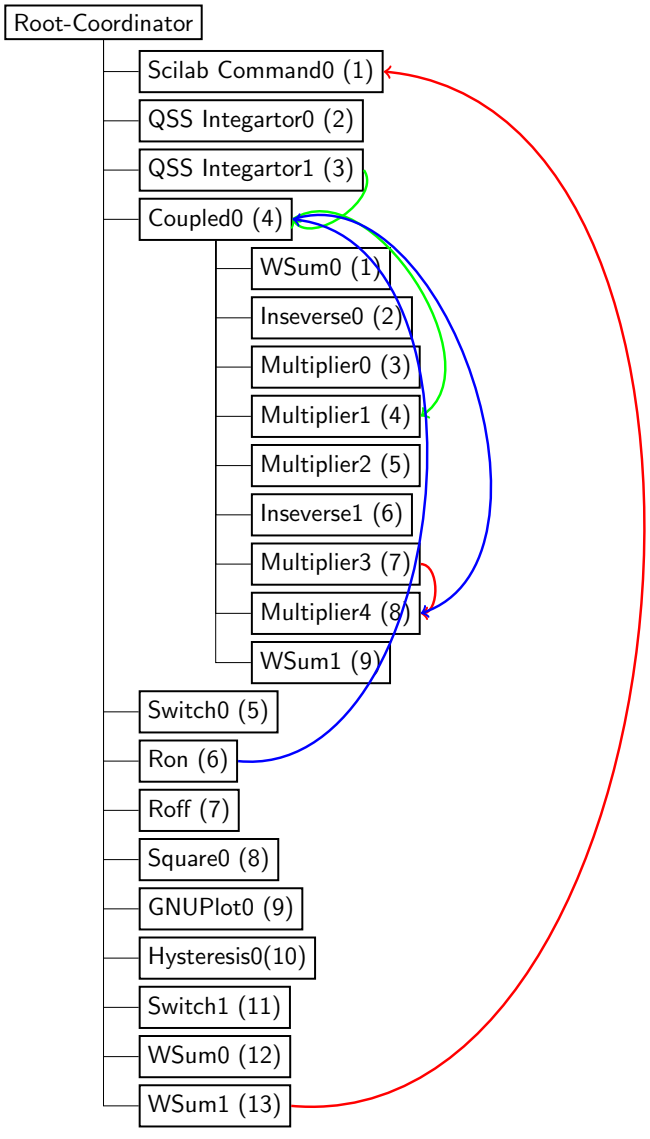


Plano

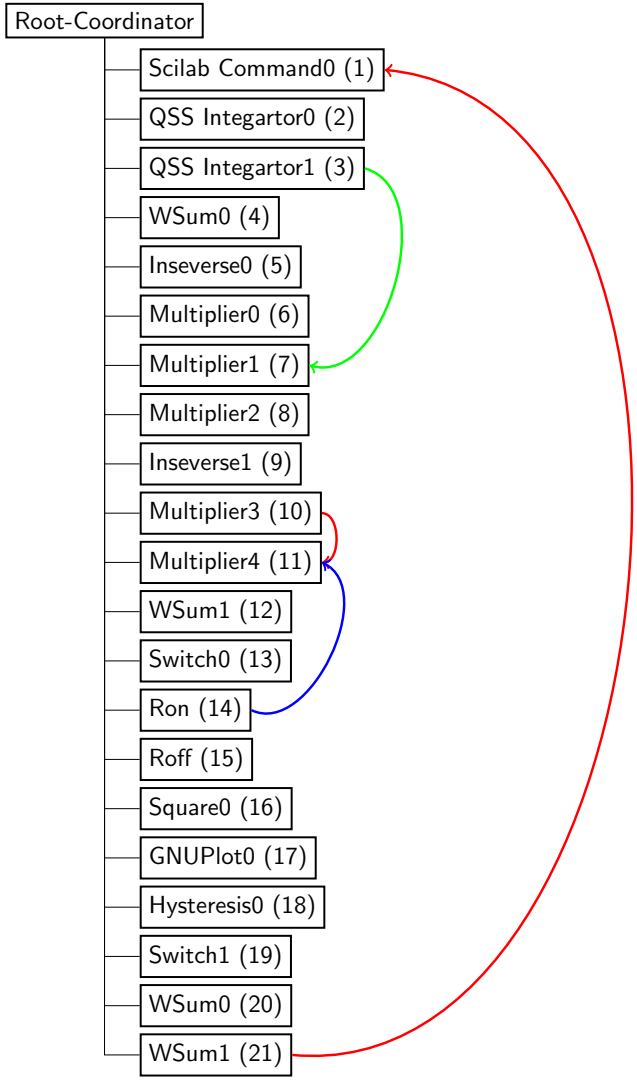


Convertidor de Voltaje

Jerárquico



Plano



Modelos Vectoriales

```
for i in 1:1000 loop
  IndexShift_2_u[1][i] = VecInt_1_y[1][i];
end for;
for i in 1:1000 loop
  Vec2Scalar_5_u[1][i] = VecInt_1_y[1][i];
end for;
for i in 1:1000 loop
  vector_pow2_6_u[1][i] = VecInt_1_y[1][i];
end for;
for i in 1:1000 loop
  VectorSum_8_u[2][i] = VecInt_1_y[1][i];
end for;
```

Modelos Vectoriales

```
for i in 1:1000 loop
  IndexShift_2_u_1[i] = VecInt_1_y_1[i];
end for;
Scalar2Vector_3_u[1] = Constant_4_y[1];
for i in 1:1000 loop
  Vec2Scalar_5_u_1[i] = VecInt_1_y_1[i];
end for;
for i in 1:1000 loop
  vector_pow2_6_u_1[i] = VecInt_1_y_1[i];
end for;
for i in 1:1000 loop
  VectorSum_8_u_2[i] = VecInt_1_y_1[i];
end for;
```

Transformaciones Extras

- Producto Interno
- Arreglos Bidimensionales
- Condicional

Ejemplo Arreglos Bidimensionales

Antes

```
Real IndexShift_2_u[IndexShift_2_N,1];
```

Despues

```
Real IndexShift_2_u_1[IndexShift_2_N];
```

Ejemplo de producto Interno

Antes

```
    VectorSum_3_y_1[VectorSum_3_i] =
↪   VectorSum_3_u[VectorSum_3_i, 1:VectorSum_3_nin] *
↪   VectorSum_3_w;
```

Despues

```
    VectorSum_3_y[1,VectorSum_3_i] =
↪   VectorSum_3_u[1,VectorSum_3_i]*VectorSum_3_w[1]+
↪   VectorSum_3_u[2,VectorSum_3_i]*VectorSum_3_w[2]+
↪   VectorSum_3_u[3,VectorSum_3_i]*VectorSum_3_w[3]+
↪   VectorSum_3_u[4,VectorSum_3_i]*VectorSum_3_w[4];
```

Ejemplo Condicional - Antes

```
  if IndexShift_2_Shift > 0 then
    for IndexShift_2_i in
↪   1:IndexShift_2_N-IndexShift_2_Shift loop
      IndexShift_2_y_1[IndexShift_2_i+IndexShift_2_Shift]
↪   = IndexShift_2_u_1[IndexShift_2_i];
    end for;
    for IndexShift_2_i in 1:IndexShift_2_Shift loop
      IndexShift_2_y_1[IndexShift_2_i] = 0;
    end for;
  else
    for IndexShift_2_i in
↪   1:IndexShift_2_N-IndexShift_2_Shift loop
      IndexShift_2_y_1[IndexShift_2_i] =
IndexShift_2_u_1[IndexShift_2_i - +IndexShift_2_Shift];
    end for;
    for IndexShift_2_i in IndexShift_2_N +
↪   IndexShift_2_Shift : IndexShift_2_N loop
      IndexShift_2_y_1[IndexShift_2_i] = 0;
    end for;
  end if;
```

Ejemplo Condicional - Despues

```
  for IndexShift_2_i in
↪   1:IndexShift_2_N-IndexShift_2_Shift loop
    IndexShift_2_y_1[IndexShift_2_i+IndexShift_2_Shift] =
↪   IndexShift_2_u_1[IndexShift_2_i];
  end for;
  for IndexShift_2_i in 1:IndexShift_2_Shift loop
    IndexShift_2_y_1[IndexShift_2_i] = 0;
  end for;
```

Resultados - Lotka-Volterra

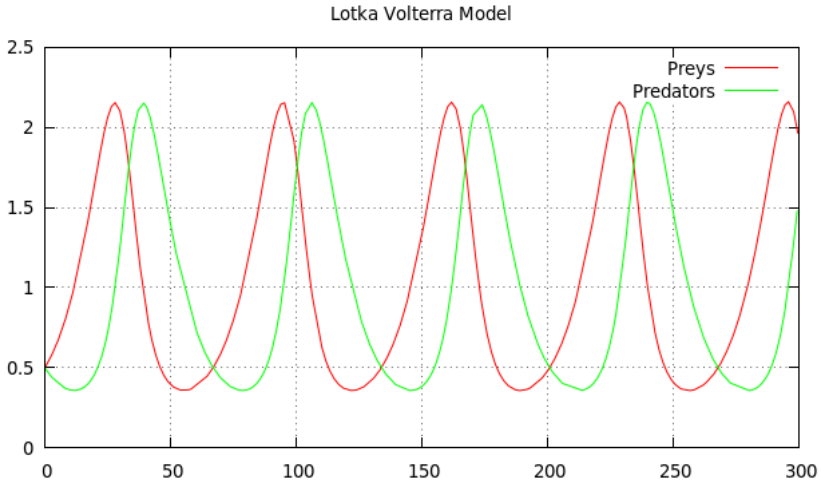
$$\frac{dx}{dt} = x(\alpha - \beta y)$$
$$\frac{dy}{dt} = y(\gamma - \delta x)$$

donde:

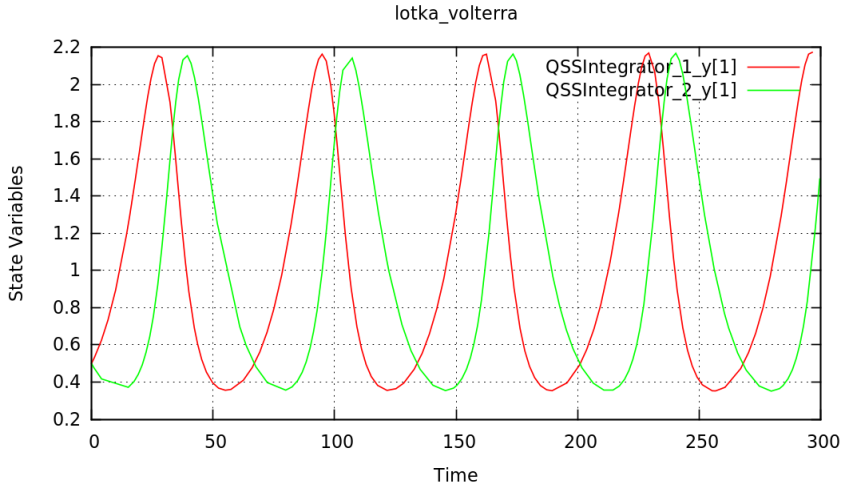
- y es el número de algún predador (por ejemplo, un lobo);
- x es el número de sus presas (por ejemplo, conejos);
- t representa el tiempo; y
- α, β, γ, δ son parámetros que representan las interacciones de las dos especies.

Resultados - Lotka-Volterra

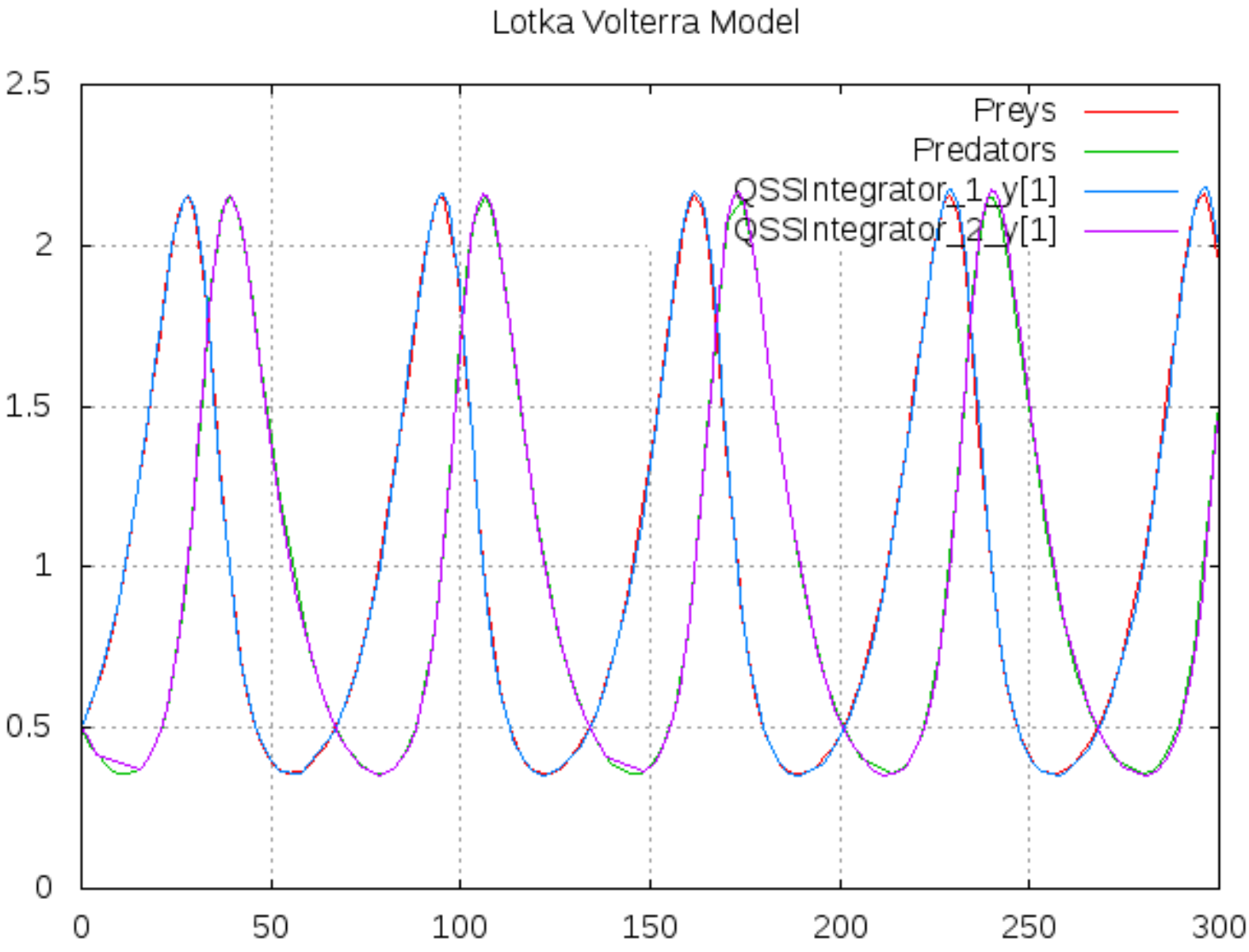
PowerDEVS



QSS-Solver



Resultados - Lotka-Volterra



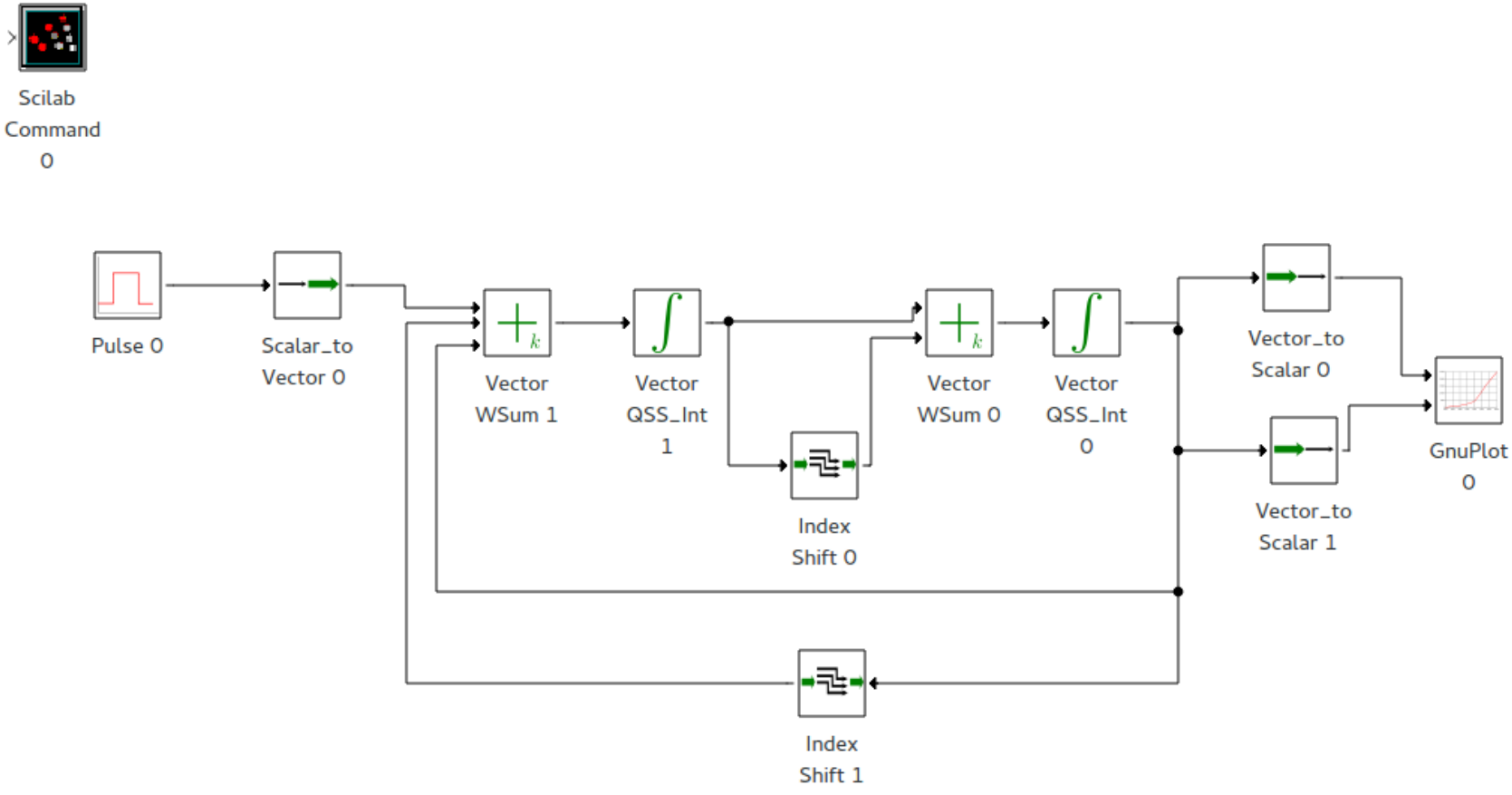
Resultados - Líneas de Transmisión

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

para $j = 2 \ldots N$
Consideramos un pulso de entrada:

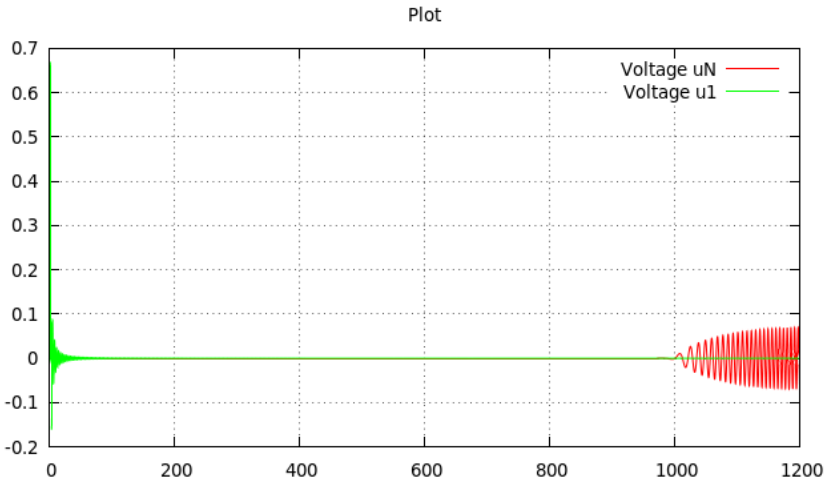
$$v_0(t) = \begin{cases} 1 \text{ si } t < 1 \\ 0 \text{ en caso contrario} \end{cases}$$

Resultados - Modelo Líneas de Transmisión en PowerDEVS

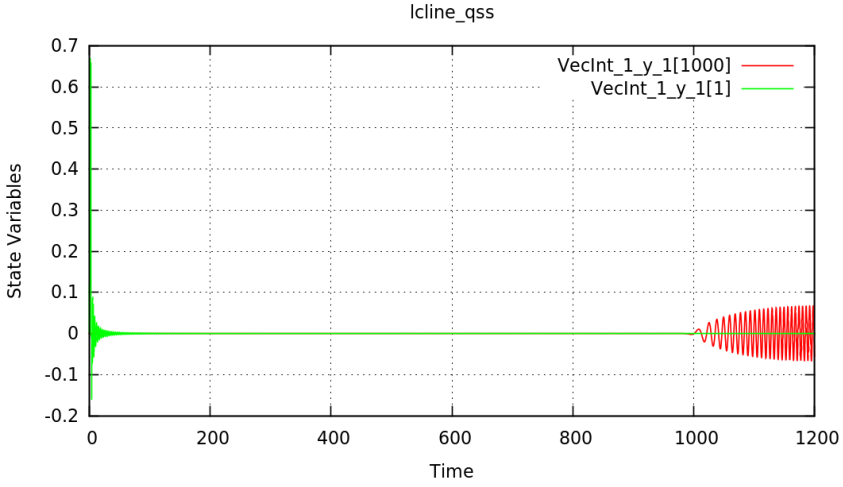


Resultados - Líneas de Transmisión

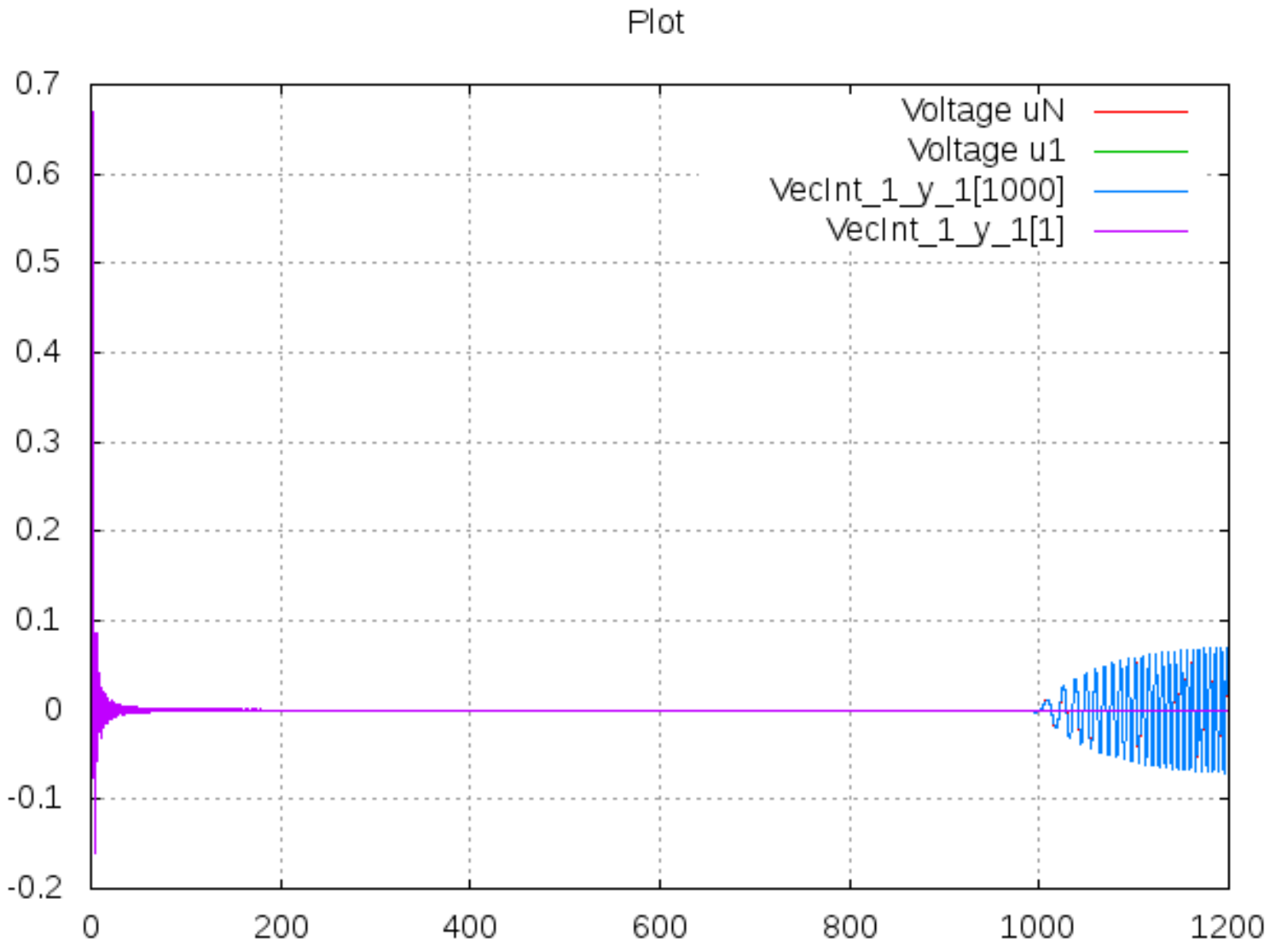
PowerDEVS



QSS-Solver



Resultados - Líneas de Transmisión



Resultados - 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

Conclusiones

- Realizamos una traducción para permitir ejecutar modelos PowerDEVS dentro de la herramienta de simulación QSS-Solver.
- Para lo cual se generaron de forma manual los modelos Modelica equivalentes a los bloques PowerDEVS utilizados en los ejemplos.
- Se desarrolló un algoritmo de traducción para la estructura de archivos PDS de PowerDEVS sin modelos acoplados.
- Se desarrolló un algoritmo para el aplanado de modelos PowerDEVS.
- Se implementaron transformaciones automáticas sobre el código Modelica para utilizar QSS-Solver.
- Se logro una mejora del tiempo de simulación las cuales alcanzan cerca del 90 % en modelos grandes.
- Mostramos además como se mantienen los valores obtenidos de la simulación.

Trabajos Futuros

- Para poder convertir una mayor variedad de modelos de PowerDEVS, es necesario escribir más modelos en Modelica equivalentes a los modelos atómicos.
- Algunos modelo atómicos necesitan expandir el QSS-Solver para poder ser ejecutados, por ejemplo los modelos de tiempo real.
- La librería modelicacc , ha sido actualizada con un nuevo “parser”.
- La expansión de variables vectoriales, donde los valores provienen de expresiones del entorno Scilab es siempre tratado como un valor escalar y no como uno vectorial.
- Es deseable que podamos llamar a esta conversión desde el mismo PowerDEVS y ejecutar el modelo resultante en el QSS-Solver.

¿Preguntas?

The End