A short tutorial for Dynsys
A program for dynamical systems based on dynamic graphs

Stephane Despreaux, Aude Maignan
stephane.despreaux@imag.fr, aude.maignan@imag.fr

http://www-ljk.imag.fr/membres/Stephane.Despreaux/Ingenierie/Dynsys/doc/doc.html

DynSys is a program dedicated to the modeling and the simulation of dynamical systems based on dynamic graphs. Each agent and link of the graph obeys a dynamical system. The state of each agent is determined thanks to the state of its neighbors. Agents and links between agents can be added to or removed from the graph. And consequently the dynamic of agents can evolve. The aim of this paper is to discover main command functions of Dynsys with the help of a simple example.

1 The presentation of the example

Let us consider a simple example. The state of nodes and edges are one-dimensional variables, X and Y respectively. The dynamic function of the nodes is $\dot{X}_i = \frac{6}{\sharp\sigma(i)}$ where $\sharp\sigma(i)$ denotes the number of neighbors of the node i. Let Y_{ij} be the age of the edge (i,j). Its dynamic is $\dot{Y}_{ij} = 1$.

At a given time t^*, a node i and one of its neighbors j create a new node k (see figure 1) when the constraint $X_i(t^*) + X_j(t^*) \geq 12$ is verified.

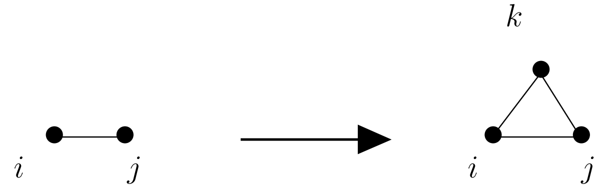


Figure 1: Local transformation

The value of $X_k(t^*)$ is defined and the values of $X_i(t^*)$ and $X_j(t^*)$ are modified thanks to reset functions: $X_k(t^*) = \frac{X_i(t^*)+X_j(t^*)}{2}$, $X_i(t^*) = \frac{X_i(t^*)}{2}$ and $X_j(t^*) = \frac{X_j(t^*)}{2}$. The edge (i,j) tenses and the nodes i and j merge when the constraint $Y_{ij}(t^*) \geq 5$ is verified. The reset function of the node i is then $X_i(t^*) = \frac{X_i(t^*)+X_j(t^*)}{2}$.

2 The modeling with DynSys

DynSys software has been developed in C++ and Java.

When lauching DynSys (with the command function rundynsys.sh) the interface looks as figure 2.

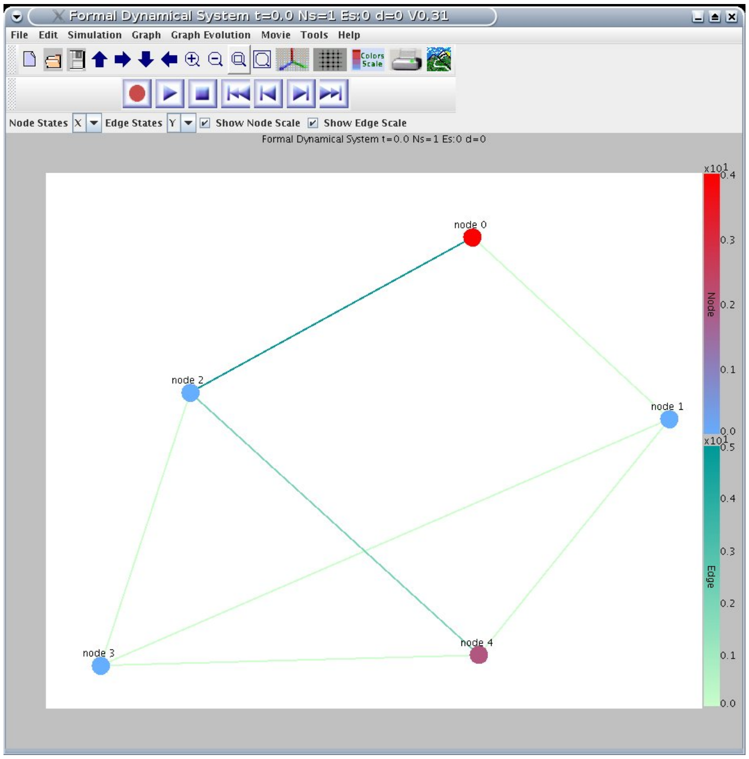


Figure 2: Dynsys interface

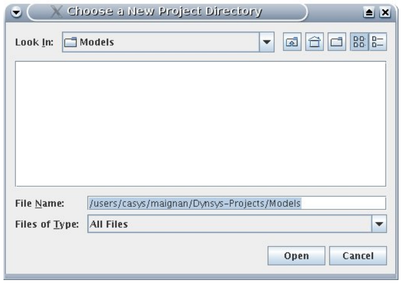
A new project can be opened with sub-menu file/New Project (see figure 3)

To edit the modeling window click edit/Dynamical Systems and fill in its fields as follow (figure 4).

The name of the variables which describe node states and edge states must begin with the letter “m”. If a node variable describes the position of the node, the name of this variable must begin by “mPos”. In that case the position of nodes is taken into account when the graph will be drawn on the main window.

In the example, mX denotes the name of the node state and mY denotes the name of the edge state.

Write “mX” on the *Name of node states* field and write “mY” on the *Name of edge states* field. Click on the buttons *add* (figure 5).

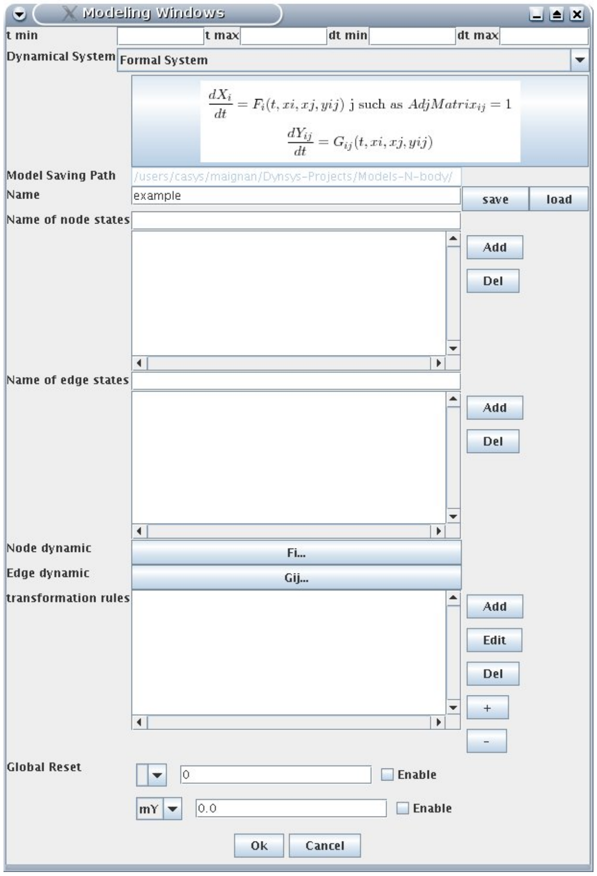


Look In: Models

File Name: /users/casys/maignan/Dynsys-Projects/Models

Files of Type: All Files

Figure 3: Open a new project



Dynamical System: Formal System

$$\frac{dX_i}{dt} = F_i(t, xi, xj, yij) \text{ j such as } AdjMatrix_{ij} = 1$$

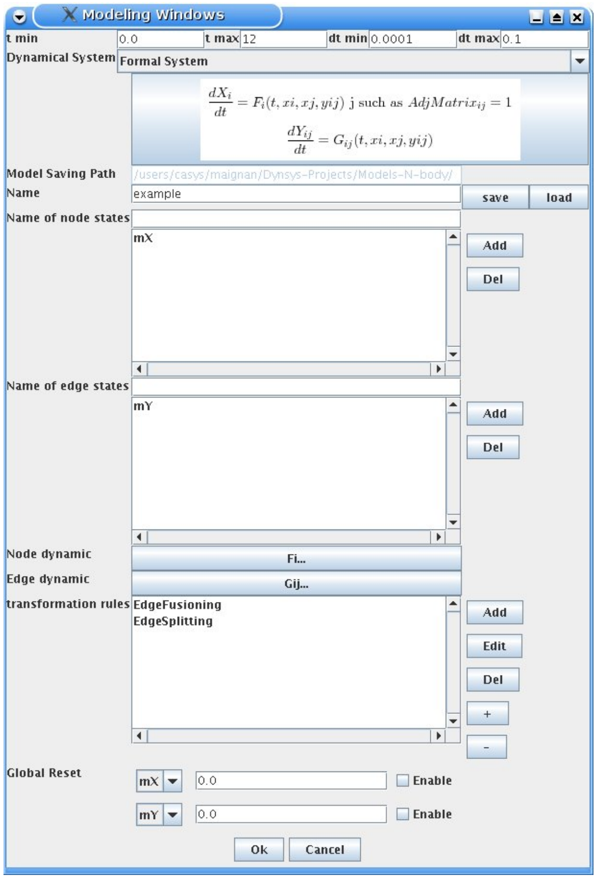
$$\frac{dY_{ij}}{dt} = G_{ij}(t, xi, xj, yij)$$

Model Saving Path: /users/casys/maignan/Dynsys-Projects/Models-N-body/

Name: example

Global Reset: 0 □ Enable

mY 0.0 □ Enable



t min 0.0 t max 12 dt min 0.0001 dt max 0.1

Dynamical System: Formal System

$$\frac{dX_i}{dt} = F_i(t, xi, xj, yij) \text{ j such as } AdjMatrix_{ij} = 1$$

$$\frac{dY_{ij}}{dt} = G_{ij}(t, xi, xj, yij)$$

Model Saving Path: /users/casys/maignan/Dynsys-Projects/Models-N-body/

Name: example

Name of node states: mX

Name of edge states: mY

transformation rules: EdgeFusioning, EdgeSplitting

Global Reset: mX 0.0 □ Enable

mY 0.0 □ Enable

Figure 4: empty modeling window

Figure 5: filled modeling window

2.1 Dynamics

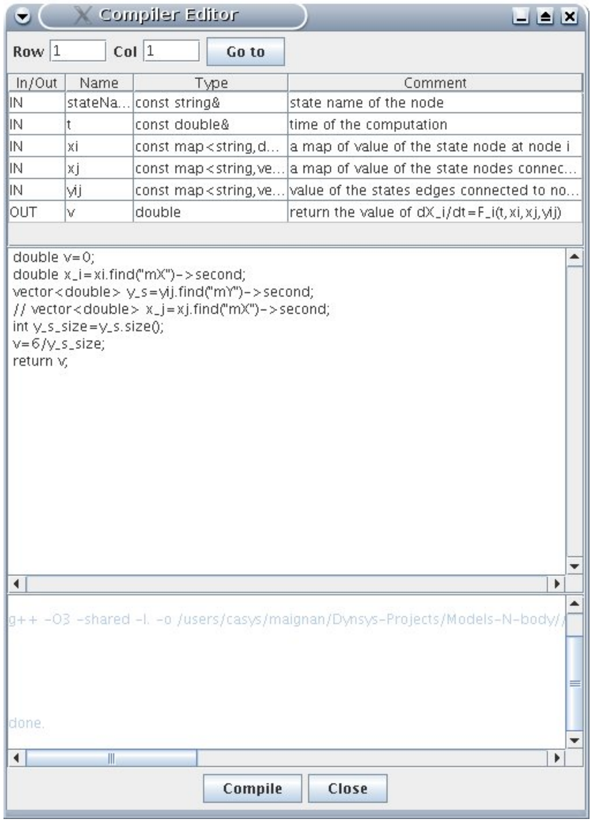
In the example, the dynamic function of nodes is $\dot{X}_i = \frac{6}{\sharp\sigma(i)}$.

Click on the *Fi...* button to specify the *Node dynamic*. The compiler editor appear. Available variables and their specification are listed in the panel. Fill in the text editor as shown below (figure 6) and click on the compile button. Informations on the compilation will be written on the bottom of the window editor.

The states of the neighbors of node i and the states of the edges linked to the node i are available and can also be used in the dynamic.

In the example, the dynamic function of edges is $\dot{Y}_{ij} = 1$.

Click on the *Gij...* button to specify the *Edge dynamic*. A compiler editor appear. Fill in the window as shown below (figure 7) and click on the compile button.

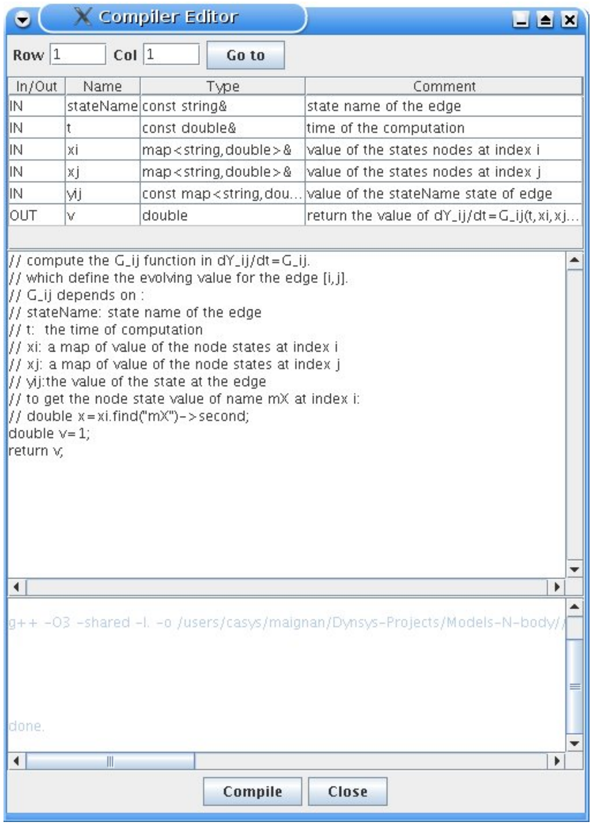


In/Out	Name	Type	Comment
IN	stateNa...	const string&	state name of the node
IN	t	const double&	time of the computation
IN	xi	const map<string,d...	a map of value of the state node at node i
IN	xj	const map<string,ve...	a map of value of the state nodes connec...
IN	yij	const map<string,ve...	value of the states edges connected to no...
OUT	v	double	return the value of dX_i/dt=F_i(t,xi,xj,yij)

```
double v=0;
double x_i=xi.find("mX")->second;
vector<double> y_s=yij.find("mY")->second;
// vector<double> x_j=xj.find("mX")->second;
int y_s_size=y_s.size();
v=6/y_s_size;
return v;
```

```
g++ -O3 -shared -I. -o /users/casys/maignan/Dynsys-Projects/Models-N-body/

done.
```

Figure 6: Node dynamic



In/Out	Name	Type	Comment
IN	stateName	const string&	state name of the edge
IN	t	const double&	time of the computation
IN	xi	map<string,double>&	value of the states nodes at index i
IN	xj	map<string,double>&	value of the states nodes at index j
IN	yij	const map<string,dou...	value of the stateName state of edge
OUT	v	double	return the value of dY_ij/dt=G_ij(t,xi,xj...

```
// compute the G_ij function in dY_ij/dt=G_ij.
// which define the evolving value for the edge [i,j].
// G_ij depends on :
// stateName: state name of the edge
// t:  the time of computation
// xi: a map of value of the node states at index i
// xj: a map of value of the node states at index j
// yij:the value of the state at the edge
// to get the node state value of name mX at index i:
// double x=xi.find("mX")->second;
double v=1;
return v;
```

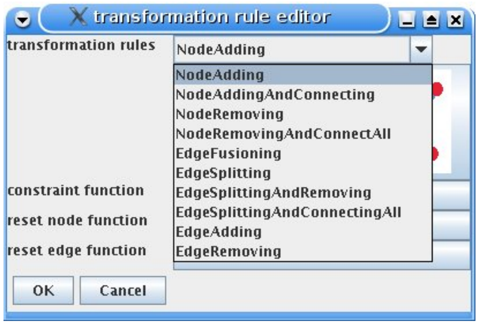
```
g++ -O3 -shared -I. -o /users/casys/maignan/Dynsys-Projects/Models-N-body/

done.
```

Figure 7: Edge dynamic

The states of the nodes i and j are available and can also be used in the edge dynamic.

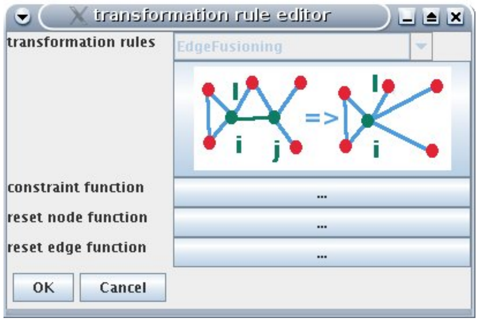
2.2 Rule transformations

Predefined transformation rules are proposed in the menu. Click on the Add button of the *transformation rules* field, figure 4, and click to the arrow to edit the menu (Figure 8).



transformation rules: NodeAdding

Figure 8: Transformation rule menu



transformation rules: EdgeFusioning

Figure 9: Rule: *Edge Fusionning*

In the current version of DynSys, there are ten predefined transformations: *Node Adding, Node Adding And Connecting, Node Removing, Node Removing And Connecting All, Edge Fusioning, Edge Splitting, Edge Splitting and Removing, Edge Splitting and Reconnecting All, Edge Removing* and *Edge Adding.*

An intuitive definition of these transformation rules is given thanks to a draw. Users can choose several transformation rules and create a list (figure 5). For each transformation rule, the constraint equations and (if necessary) the reset functions for nodes and edges must be specified by C++ instructions in the constraint window. If, simultaneously, two rules can be applied the first rule in the order of the list will be applied (possibly several times). If the constraint of the second rule is ever verified on the non final new graph, the second rule is applied. So rules must be chosen in the order of priority.

In the example we suppose that the death of nodes has the highest order of priority.

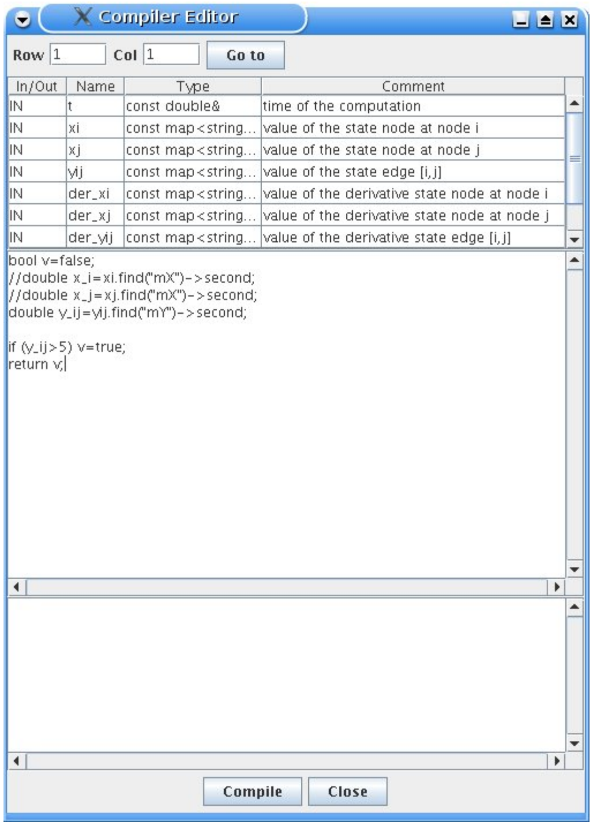
To add the first transformation rule, click on the button *Add* of the *transformation rules* field (figure 4) and choose the rule *Edge fusionning* (figure 9).

The constraint $Y_{ij} \geq 5$ must be written on a C++ compiler editor. Click on the button *constraint function* and fill in the constraint editor as in figure 10.

After the transformation rule X_i get the new value $\frac{X_i+X_j}{2}$.

Click on the button *reset node function* in order to define the new values of the variables mX_i and fill in the compiler editor as in figure 11.

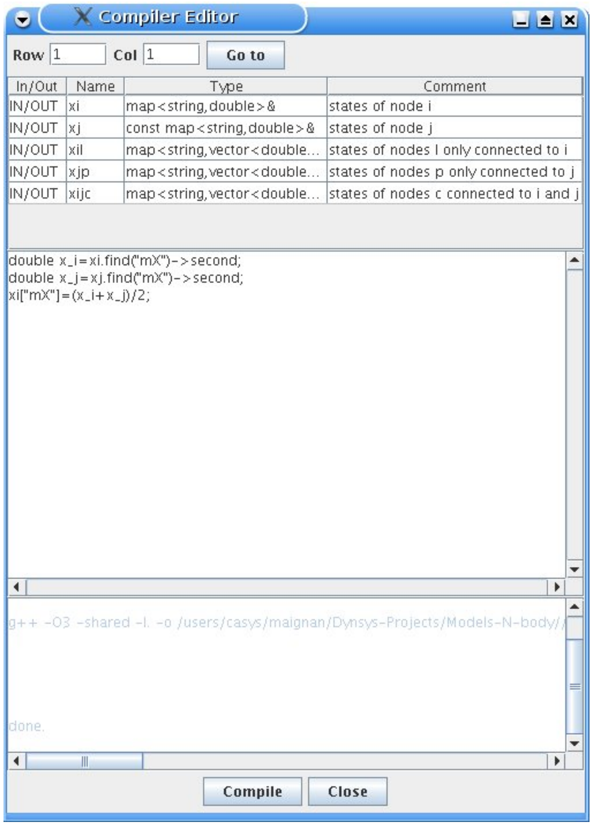
The reset function of the nodes i is then: $X_i(t^*) = \frac{X_i(t^*)+X_j(t^*)}{2}$. No edge has to be redefined. The window of *reset edge function* (figure 9) remains empty.



In/Out	Name	Type	Comment
IN	t	const double&	time of the computation
IN	xi	const map<string...	value of the state node at node i
IN	xj	const map<string...	value of the state node at node j
IN	yij	const map<string...	value of the state edge [i,j]
IN	der_xi	const map<string...	value of the derivative state node at node i
IN	der_xj	const map<string...	value of the derivative state node at node j
IN	der_yij	const map<string...	value of the derivative state edge [i,j]

```
bool v=false;
//double x_i=xi.find("mX")->second;
//double x_j=xj.find("mX")->second;
double y_ij=yij.find("mY")->second;

if (y_ij>5) v=true;
return v;
```

Figure 10: Constraint function



In/Out	Name	Type	Comment
IN/OUT	xi	map<string,double>&	states of node i
IN/OUT	xj	const map<string,double>&	states of node j
IN/OUT	xil	map<string,vector<double...	states of nodes l only connected to i
IN/OUT	xjp	map<string,vector<double...	states of nodes p only connected to j
IN/OUT	xijc	map<string,vector<double...	states of nodes c connected to i and j

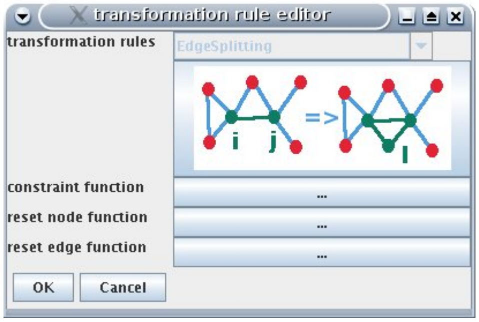
```
double x_i=xi.find("mX")->second;
double x_j=xj.find("mX")->second;
xi["mX"]=(x_i+x_j)/2;
```

```
g++ -O3 -shared -I. -o /users/casys/maignan/Dynsys-Projects/Models-N-body//



done.
```

Figure 11: Reset function for nodes

The *Edge Splitting and Reconnecting All* transformation rule is also add in our example. Click on the button *Add* of the *transformation rules* field (figure 4) and choose the rule *Edge Splitting and Reconnecting All* (figure 12).

The constraint $X_i(t^*) + X_j(t^*) \geq 12$ must be written on the C++ compiler editor. Click on the button *constraint function* and fill in the constraint editor as in figure 13.



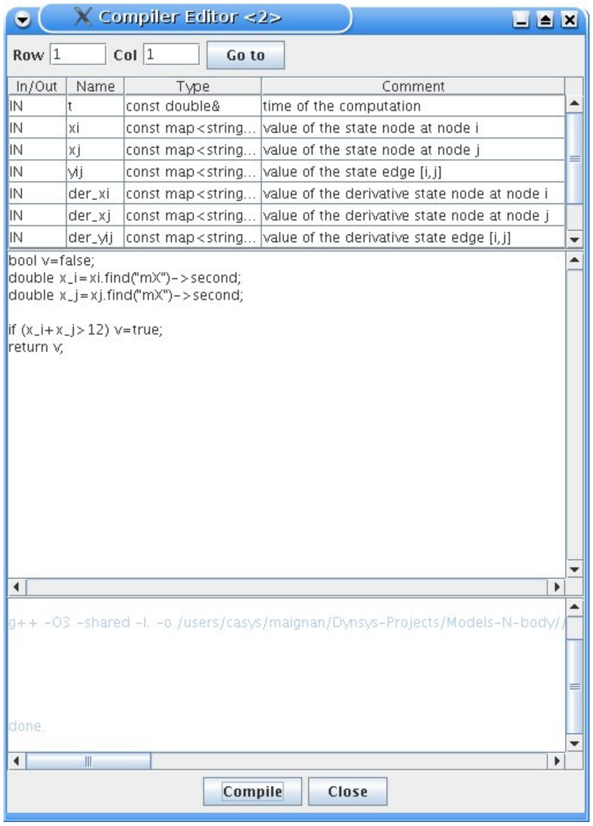
transformation rules: EdgeSplitting

constraint function

reset node function

reset edge function

Figure 12: Rule: *Edge Splitting and Reconnecting All*



Row 1 Col 1

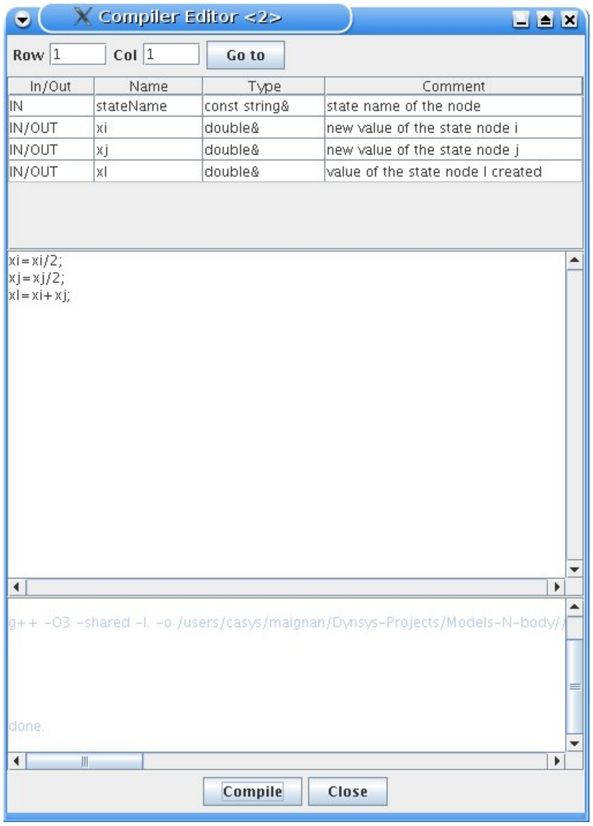
In/Out	Name	Type	Comment
IN	t	const double&	time of the computation
IN	xi	const map<string...	value of the state node at node i
IN	xj	const map<string...	value of the state node at node j
IN	yij	const map<string...	value of the state edge [i,j]
IN	der_xi	const map<string...	value of the derivative state node at node i
IN	der_xj	const map<string...	value of the derivative state node at node j
IN	der_yij	const map<string...	value of the derivative state edge [i,j]

```
bool v=false;
double x_i=xi.find("mX")->second;
double x_j=xj.find("mX")->second;

if (x_i+x_j>12) v=true;
return v;
```

```
g++ -O3 -shared -I. -o /users/casys/maignan/Dynsys-Projects/Models-N-body//



done.
```

Figure 13: Constraint function

Click on the button *reset node function* (figure 12) in order to define the new value of the variables mX_i, mX_j and mX_k (figure 14). Click on the button *reset edge function* (figure 12) in order to define the new value of the variables mY_{ij}, mY_{ik} and mY_{jk} (figure 15).

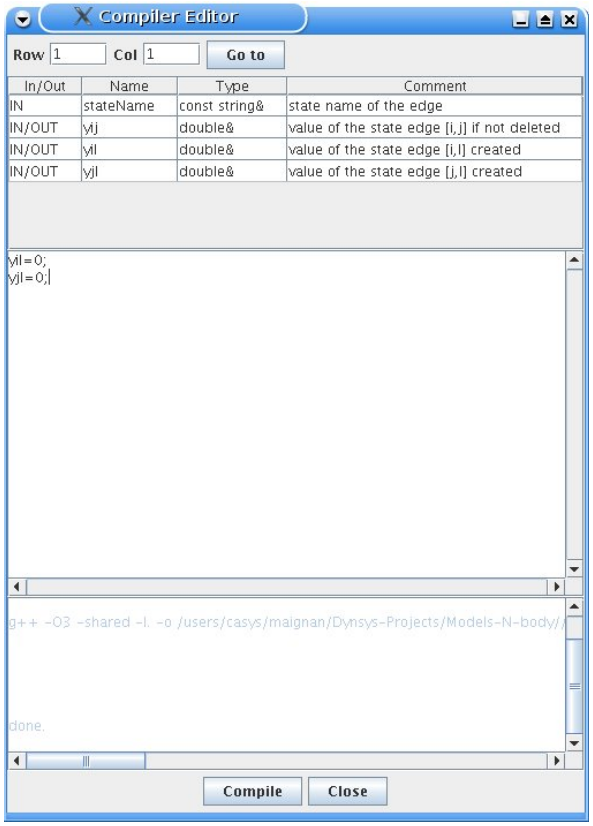


In/Out	Name	Type	Comment
IN	stateName	const string&	state name of the node
IN/OUT	xi	double&	new value of the state node i
IN/OUT	xj	double&	new value of the state node j
IN/OUT	xl	double&	value of the state node l created

```
xi=xi/2;
xj=xj/2;
xl=xi+xj;
```

```
g++ -O3 -shared -I. -o /users/casys/maignan/Dynsys-Projects/Models-N-body//



done.
```

Figure 14: Reset function for nodes

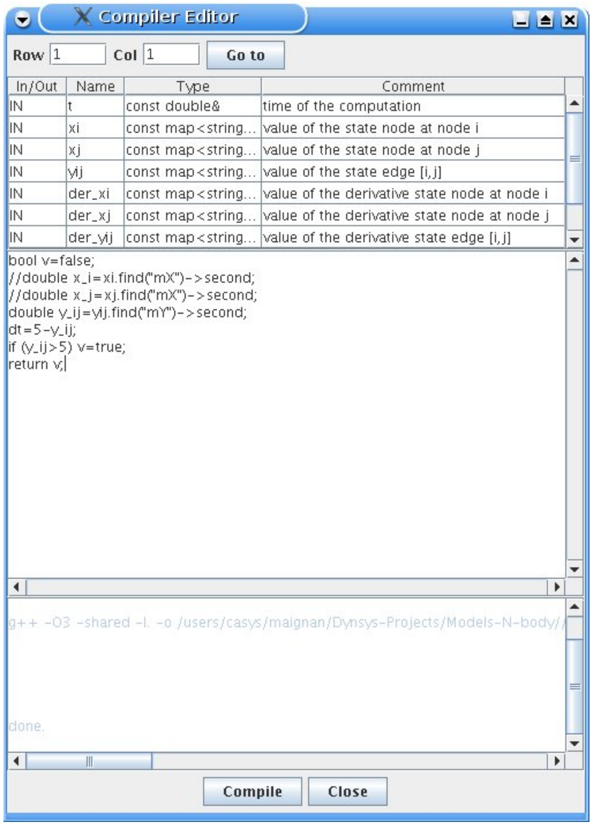


In/Out	Name	Type	Comment
IN	stateName	const string&	state name of the edge
IN/OUT	yij	double&	value of the state edge [i,j] if not deleted
IN/OUT	yil	double&	value of the state edge [i,l] created
IN/OUT	yjl	double&	value of the state edge [j,l] created

```
yil=0;
yjl=0;
```

```
g++ -O3 -shared -I. -o /users/casys/maignan/Dynsys-Projects/Models-N-body//



done.
```

Figure 15: Reset function for edges

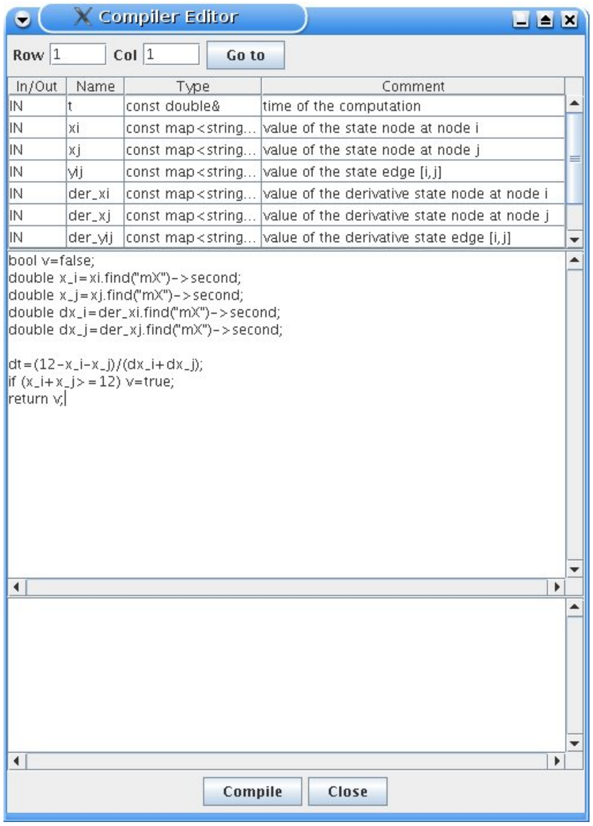


In/Out	Name	Type	Comment
IN	t	const double&	time of the computation
IN	xi	const map<string...	value of the state node at node i
IN	xj	const map<string...	value of the state node at node j
IN	yij	const map<string...	value of the state edge [i,j]
IN	der_xi	const map<string...	value of the derivative state node at node i
IN	der_xj	const map<string...	value of the derivative state node at node j
IN	der_yij	const map<string...	value of the derivative state edge [i,j]

```
bool v=false;
//double x_i=xi.find("mX")->second;
//double x_j=xj.find("mX")->second;
double y_ij=yij.find("mY")->second;
dt=5-y_ij;
if (y_ij>5) v=true;
return v;
```

```
g++ -O3 -shared -I. -o /users/casys/maignan/Dynsys-Projects/Models-N-body//


done.
```

Figure 16: Constraint function for fusionning rule which include dynamic time step



In/Out	Name	Type	Comment
IN	t	const double&	time of the computation
IN	xi	const map<string...	value of the state node at node i
IN	xj	const map<string...	value of the state node at node j
IN	yij	const map<string...	value of the state edge [i,j]
IN	der_xi	const map<string...	value of the derivative state node at node i
IN	der_xj	const map<string...	value of the derivative state node at node j
IN	der_yij	const map<string...	value of the derivative state edge [i,j]

```
bool v=false;
double x_i=xi.find("mX")->second;
double x_j=xj.find("mX")->second;
double dx_i=der_xi.find("mX")->second;
double dx_j=der_xj.find("mX")->second;

dt=(12-x_i-x_j)/(dx_i+dx_j);
if (x_i+x_j>=12) v=true;
return v;
```

Figure 17: Constraint function for splitting rule which include dynamic time step

2.3 Time step

The time steps are precised thanks to the first fields of the modeling window (figure 4. $[tmin, tmax]$ is the time interval of simulation. The minimal and maximal time steps are specified, respectively, on the fields *dt min* and *dt max*. Dynsys uses the time step *dt max* by default. Users can also integrate a dynamic time step on the constraint functions.

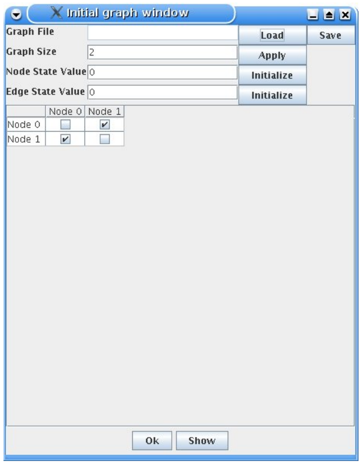
Add the line

$$dt = 5 - y\_ij;$$

in the compiler editor of the *constraint function* (figure 10). The value of the time step become $time\_step\_1 = max(min(dt, dt\ max), dt\ min)$ (figure 16)

Add the lines

$$\begin{gathered} double\ dx\_i = der\_xi.find("mX") - > second; \\ double\ dx\_j = der\_xj.find("mX") - > second; \\ dt = (12 - x_i - x_j)/(dx_i + dx_j); \end{gathered}$$



Graph File

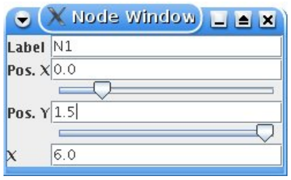
Graph Size 2

Node State Value 0

Edge State Value 0

	Node 0	Node 1
Node 0	☐	☑
Node 1	☑	☐

Figure 18: Initial graph



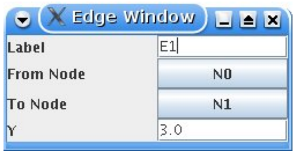
Label N1

Pos. X 0.0

Pos. Y 1.5

X 6.0

Figure 19: State of node 1



Label E1

From Node N0

To Node N1

Y 3.0

Figure 20: State of edge (1,2)

in the compiler editor of the *constraint function* (figure 13). The value of the time step become $time\_step\_2 = min(max(min(dt, dt\ max), dt\ min), time\_step\_1)$ (figure 17).

In this simple example, the time of the next graph transformation can be computed exactly. And thanks to the time step computation, a new graph is created at each step.

2.4 Initial graph

The initial graph is implemented thanks to an adjacency matrix. Choose *initial graph* in the *edit* menu. Fill in the *Graph size* field and click on *Apply* (figure 18). A matrix appears in the *initial graph window* (figure 18). Click on the put (i, j) to create the edge (i, j). Click on the button *show* and *OK*.

Dynsys produces the graph in the main window. Click on a node to see its coordinates and its state (figure 19). With this window, you can modify the position of the node on the draw and the node state as you like. Click on an edge and modify its state (figure 20).

3 Simulation

Finally, users have to click on the simulation button to begin the simulation. During the computation, the integration method is a first order numerical integration. Click on the right arrow to see step by step the evolution of the graph.

At each step, DynSys specifies the time variable t, the number of nodes N_s, the number of edges E_s, the maximal degree of the graph d and the number of transitions which have been applied T_r.

Four steps of the example are drawn bellow thanks to Dynsys.

At each step of the computation, a click on a node or an edge posts its state.

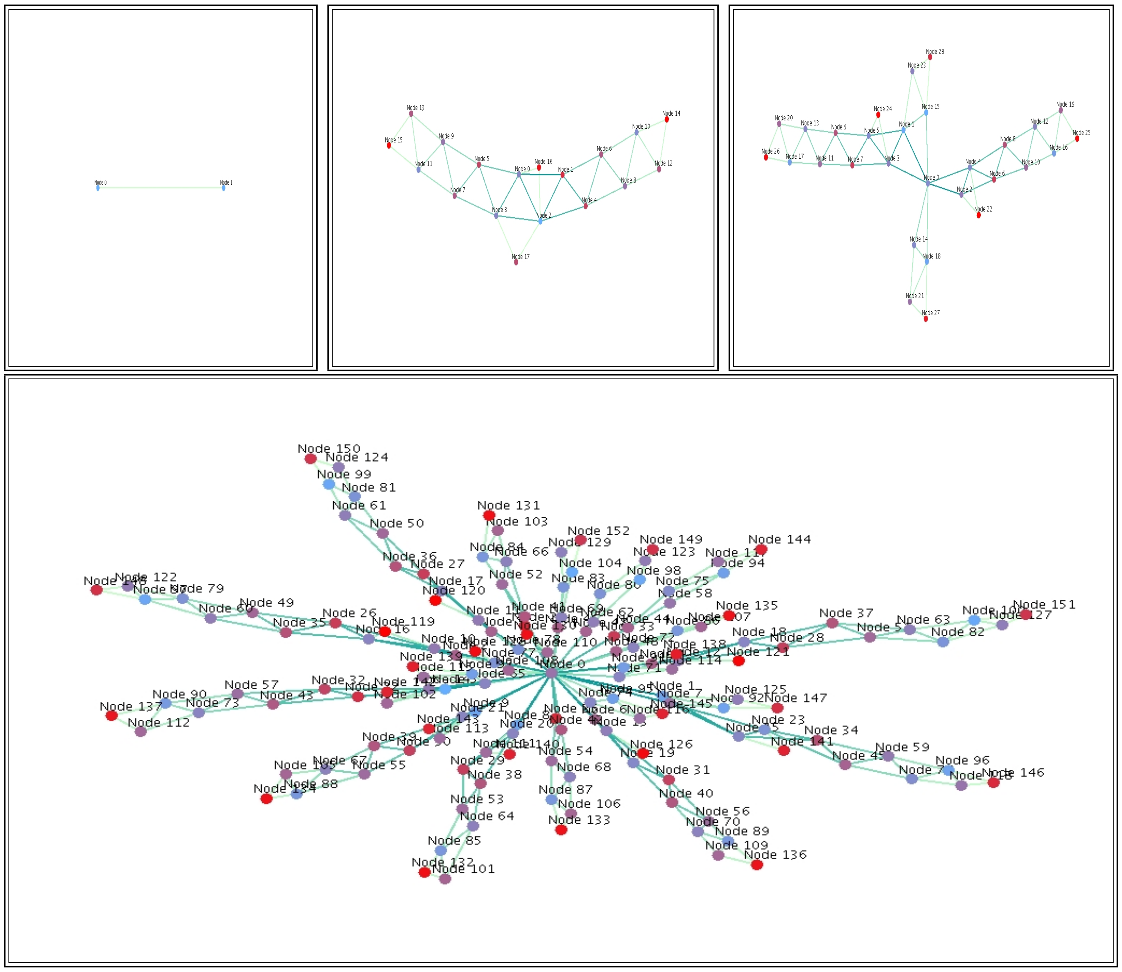


Figure 21: Figure a: t=0, Ns=2, Es=1, d=1, tr=0; Figure b: t=4.625, Ns=18, Es=33, d=6, tr=13; Figure c: t=6.2093, Ns=29, Es=53, d=7, tr =28; Figure d: t=11.0242,Ns=153, Es=289, d=38, tr=188