

*Merging CompuCell3D and
SBW/SBML*

Julio M. Belmonte

Indiana University, Bloomington

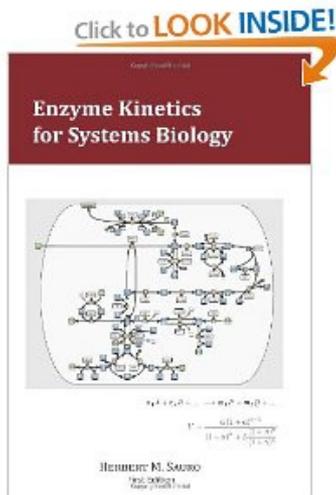
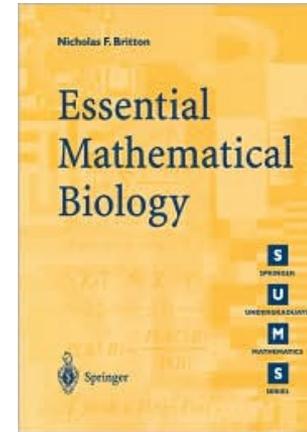
Outline

- Objectives
- Ways to add RK to CC3D
- SBML format
- Generating SBML using Jarnac
 - Simple Oscillator
- Integrating with CC3D
 - Adding Simple Oscillator to CC3D
 - Adding Cell Cycle model from sbml.org
 - John Tyson's Cell Cycle model
 - Collier *et al.* Delta-Notch patterning model

More on Reaction Kinetics Modeling

Essential Mathematical Biology

Nicholas Britton



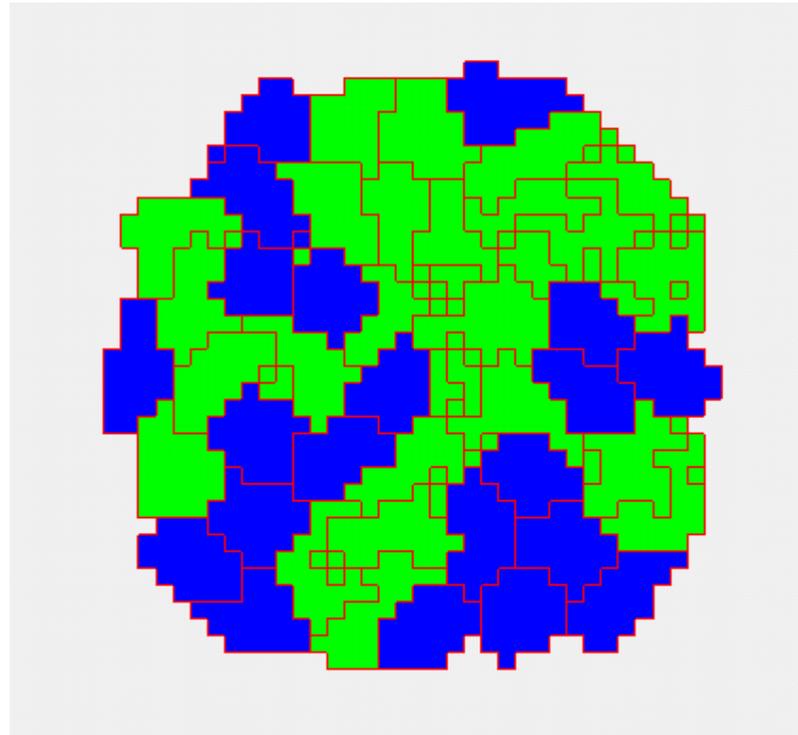
Enzyme Kinetics for Systems Biology

Herbert Sauro

www.sys-bio.org/sbwWiki/tutorials/bloomington2011

Cell-based modeling

- Cellular behaviors:
 - Location
 - Volume
 - Shape
 - Movement
 - Adhesion
 - Mitosis
 - Death
 - Differentiation
 - Polarization
 - Etc...

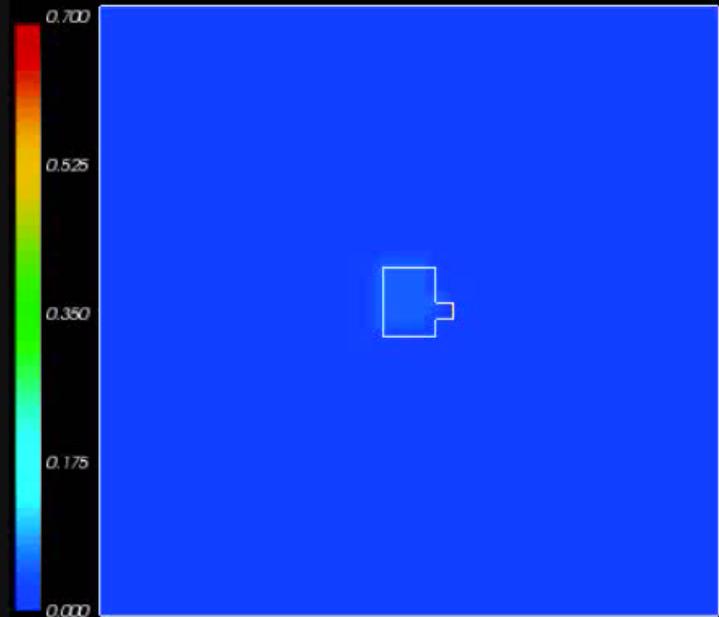
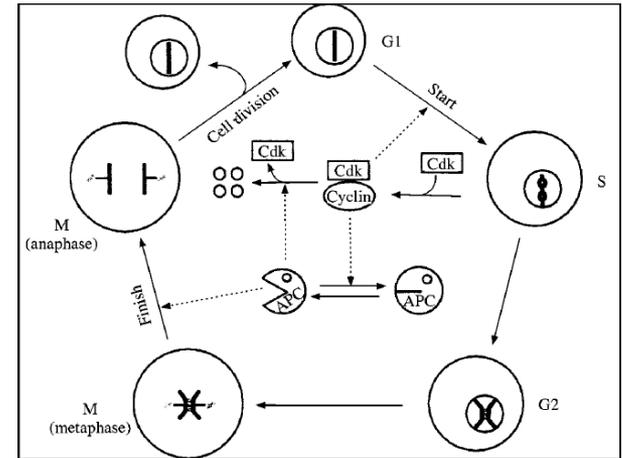


Subcellular modelling

- Biochemical Kinetics:
 - Cell-Cycle
 - Circadian rhythms
 - Cardiac rhythms
 - cAMP oscillations
 - Delta-Notch patterning
 - WNT pathway
 - FGF pathway
 - Etc...

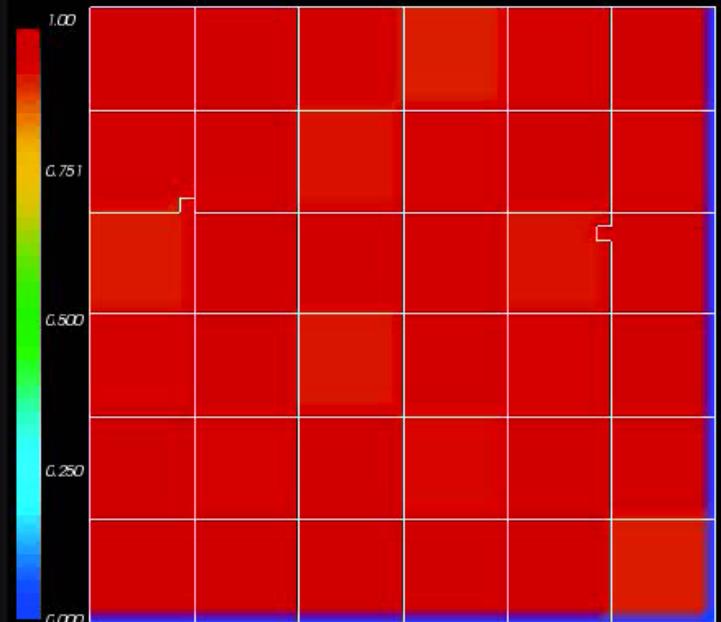
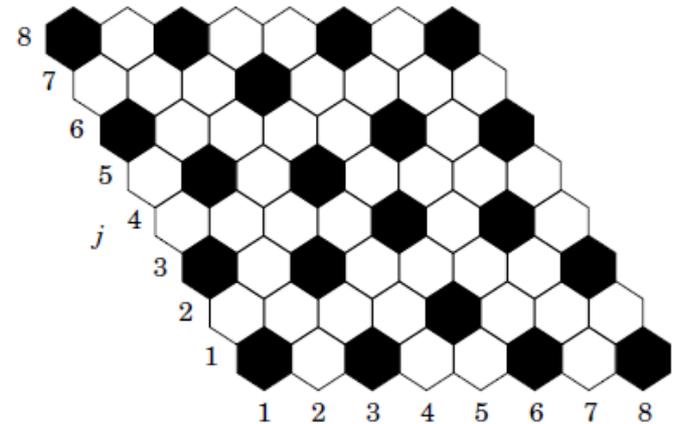
Subcellular modelling

- Biochemical Kinetics:
 - Cell-Cycle
 - Circadian rhythms
 - Cardiac rhythms
 - cAMP oscillations
 - Delta-Notch patterning
 - WNT pathway
 - FGF pathway
 - Etc...



Subcellular modelling

- Biochemical Kinetics:
 - Cell-Cycle
 - Circadian rhythms
 - Cardiac rhythms
 - cAMP oscillations
 - **Delta-Notch patterning**
 - WNT pathway
 - FGF pathway
 - Etc...



How to add this into CompuCell?

- 1) Just another Python class!
 - Too slow

How to add this into CompuCell?

- 1) Just another Python class!
 - Too slow
- 2) C++ file to be wrapped into Python
 - Too complicated

How to add this into CompuCell?

- 1) Just another Python class!
 - Too slow
- 2) C++ file to be wrapped into Python
 - Too complicated
- 3) Import SBML

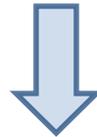
SBML – Systems Biology Markup Language

- Not a software!
- Machine-readable format for representing subcellular models
- Standard for storage and exchange of models
- Implementation agnostic

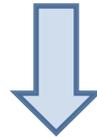
SBML

- How does it work?

Developer software (SBW/Jarnac)

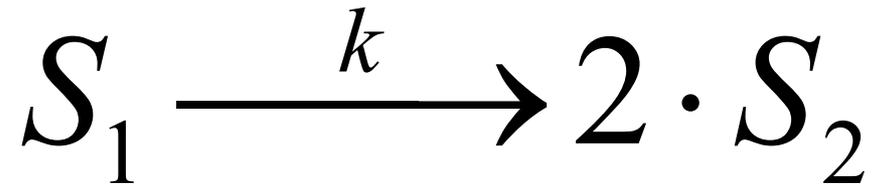


SBML



Simulation software (CompuCell3D)

SBML



- Initial conditions:

$$S_1 = 5 \text{ nM}$$

$$S_2 = 0 \text{ nM}$$

- Parameters:

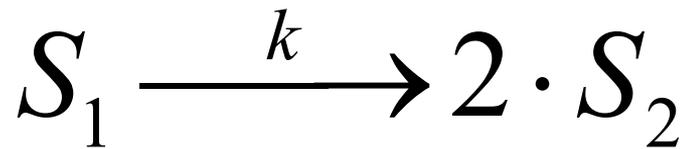
$$k = 0.1 \text{ min}^{-1}$$

SBML

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns = "http://www.sbml.org/sbml/level2" level = "2" version = "1">
  <model id = "cell">
    <listOfCompartments>
      <compartment id = "compartment" size = "1"/>
    </listOfCompartments>
    <listOfSpecies>
      <species id = "S1" boundaryCondition = "false" initialConcentration = "5.0" compartment = "compartment"/>
      <species id = "S2" boundaryCondition = "false" initialConcentration = "0.0" compartment = "compartment"/>
    </listOfSpecies>
    <listOfParameters>
      <parameter id = "k1" value = "0.1"/>
    </listOfParameters>
    <listOfReactions>
      <reaction id = "_J1" reversible = "false">
        <listOfReactants>
          <speciesReference species = "S1" stoichiometry = "1"/>
        </listOfReactants>
        <listOfProducts>
          <speciesReference species = "S2" stoichiometry = "2"/>
        </listOfProducts>
        <kineticLaw>
          <math xmlns = "http://www.w3.org/1998/Math/MathML">
            <apply>
              <times/>
              <ci>
                k1
              </ci>
              <ci>
                S1
              </ci>
            </apply>
          </math>
        </kineticLaw>
      </reaction>
    </listOfReactions>
  </model>
</sbml>
```

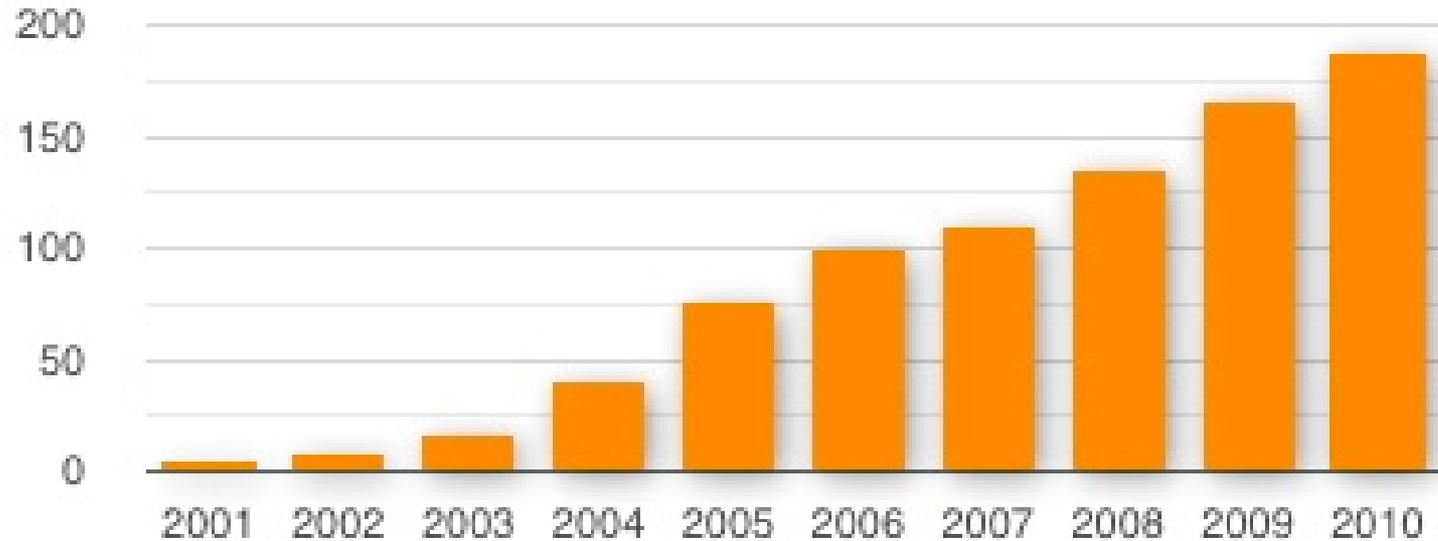
$$k = 0.1 \text{ min}^{-1}$$

$$\left. \begin{array}{l} S_1 = 5 \text{ nM} \\ S_2 = 0 \text{ nM} \end{array} \right\}$$



SBML

- Total number of known SBML-compatible software packages each year :



How to write SBML?

- [Bio-Spice](#)
 - Large collection of tools, integrated via a "Dashboard." Free download (BSD), various platforms.
- [Teranode](#)
 - Suite of tools for model management, design, and simulation. (Linux/Mac/Windows) Commercial (30-day trial available).
- [SBW](#)
 - Systems Biology Workbench.
- Check [http://sbml.org/SBML Software Guide](http://sbml.org/SBML_Software_Guide)

SBW/Jarnac

- SBW - Systems Biology Workbench:
 - Open-source software framework for systems biology
- Jarnac:
 - Software for writing and simulating reaction kinetics
 - Easy to use
 - Translate to SBML (C++, Matlab, Mathematica, etc..)
- Download at: <http://www.sys-bio.org/>

Integration with CC3D

- Reaction kinetic models can be easily added in CC3D when in SBML format.
- Once loaded, the model is converted into a set of ODEs and is solved by the BionetSolver library inside CC3D.
- The commands used to load and manipulate the models inside CC4D are summarized on the “Quick Reference Guide” for Python in CC3D.

Integration with CC3D

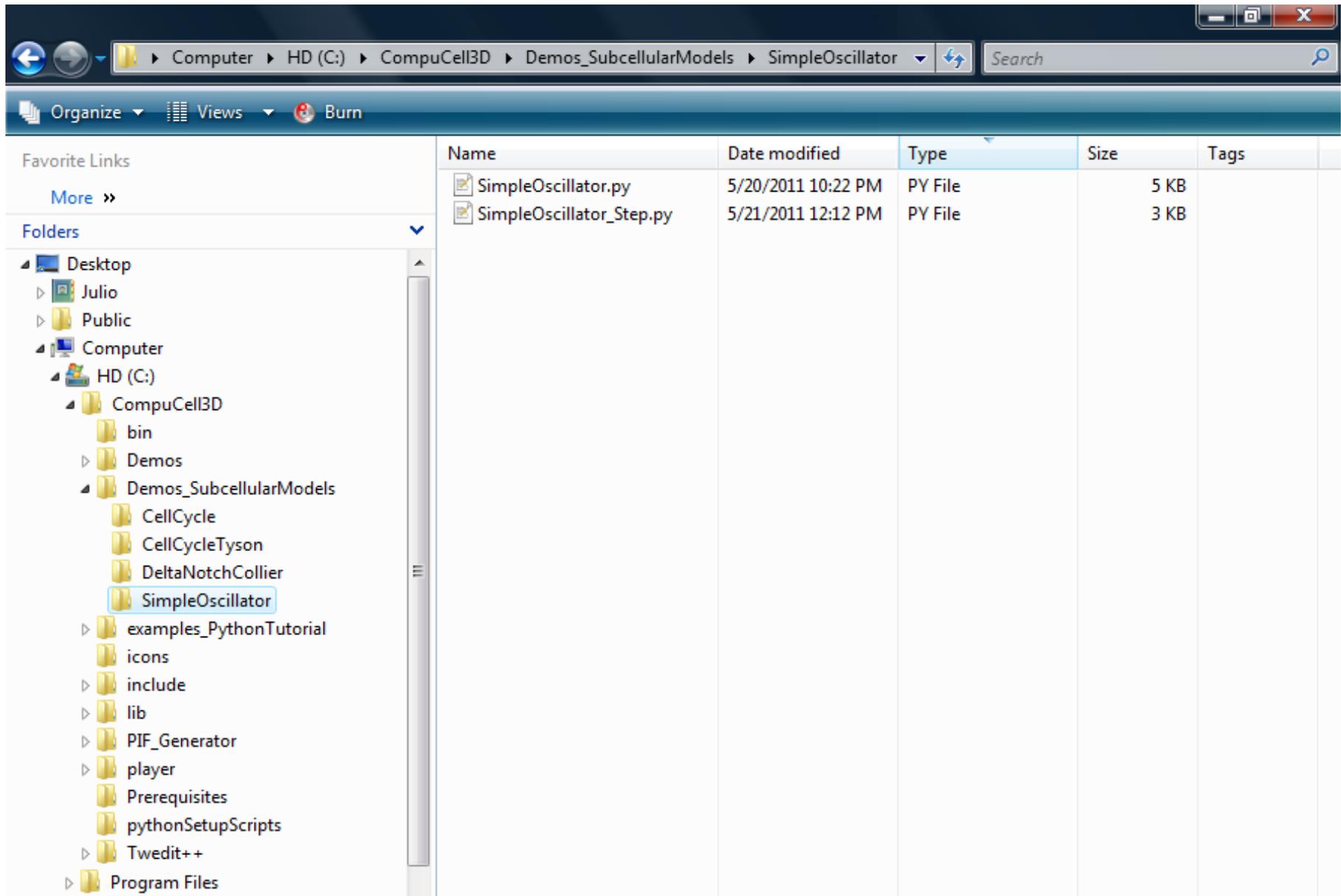
```
import bionetAPI # Import bionetAPI functions
class <someClass>(SteppableBasePy):
    def __init__(self, _simulator, _frequency=1):
        SteppableBasePy.__init__(self, _simulator, _frequency)
        bionetAPI.initializeBionetworkManager(self.simulator) # Initialize bionet inside class

    def start(self):
        # Load a specific subcellular SBML submodel
        ModelName = <sbmlModelName> # Name of the model
        ModelPath = <sbmlModelPath> # Path where the model is stored
        ModelKey = <modelKey> # Nickname of the model
        IntegrationStep = <timeStep> # Time step of integration
        bionetAPI.loadSBMLModel( ModelName, ModelPath, ModelKey, IntegrationStep )

        # Add SBML submodel to a group of cells or a single cell
        bionetAPI.addSBMLModelToTemplateLibrary(<sbmlModelName>, {<cellType> or <cellId>})
        ...
        # Modify the parameter value or molecular concentration of a cell (or group of cells)
        bionetAPI.setBionetworkValue(<molecule/parameter>, <value>, {<cellType> or <cellId>})
        ...
        # Initialize model
        bionetAPI.initializeBionetworks()

    def step(self, mcs):
        # Iterate the model (run it for the time step specified on the load command)
        bionetAPI.timestepBionetworks()
        ...
        # Get the parameter value or molecular concentration from a cell (or group of cells)
        <var>=bionetAPI.getBionetworkValue({<parameter> or <molecule>}, {<cellType> or <cellId>})
        ...
        # Modify the parameter value or molecular concentration of a cell (or group of cells)
        bionetAPI.setBionetworkValue(<molecule/parameter>, <value>, {<cellType> or <cellId>})
```

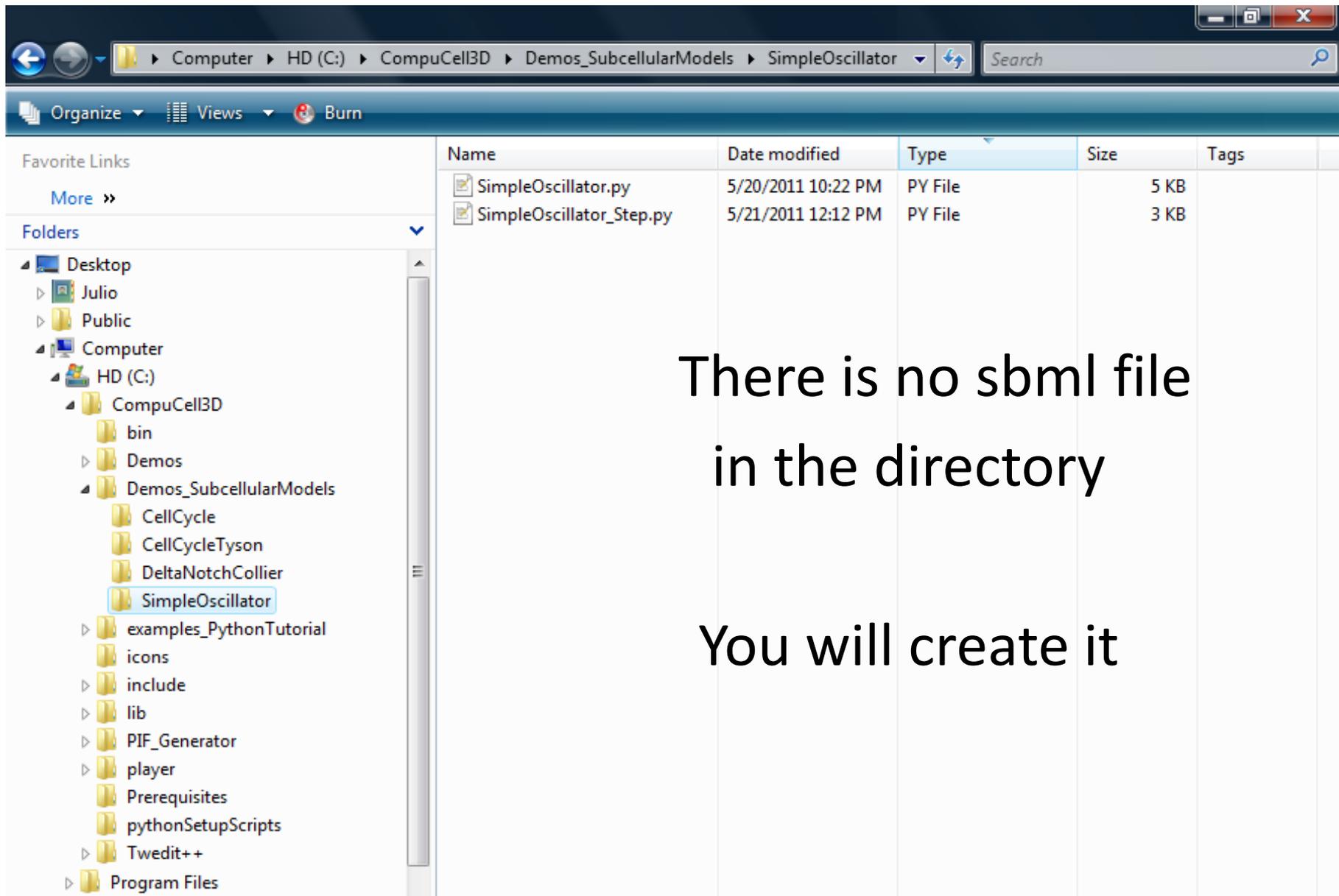
First Example – Simple Oscillator



The screenshot shows a Windows Explorer window with the address bar displaying the path: Computer > HD (C:) > CompuCell3D > Demos_SubcellularModels > SimpleOscillator. The left sidebar shows the folder tree, with 'SimpleOscillator' selected under 'Demos_SubcellularModels'. The main pane displays a table of files:

Name	Date modified	Type	Size	Tags
SimpleOscillator.py	5/20/2011 10:22 PM	PY File	5 KB	
SimpleOscillator_Step.py	5/21/2011 12:12 PM	PY File	3 KB	

First Example – Simple Oscillator



The screenshot shows a Windows Explorer window with the address bar displaying the path: Computer > HD (C:) > CompuCell3D > Demos_SubcellularModels > SimpleOscillator. The left sidebar shows the folder tree with 'SimpleOscillator' selected under 'Demos_SubcellularModels'. The main pane shows a table of files:

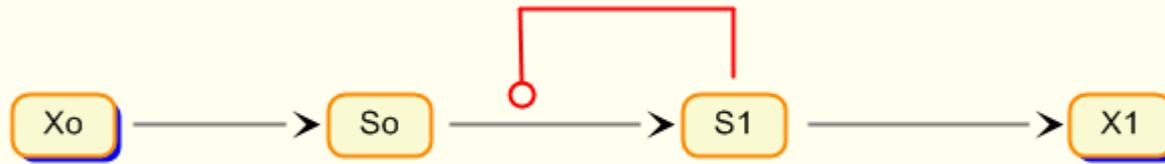
Name	Date modified	Type	Size	Tags
SimpleOscillator.py	5/20/2011 10:22 PM	PY File	5 KB	
SimpleOscillator_Step.py	5/21/2011 12:12 PM	PY File	3 KB	

There is no sbml file in the directory

You will create it

First Example – Simple Oscillator

- Relaxation oscillator:

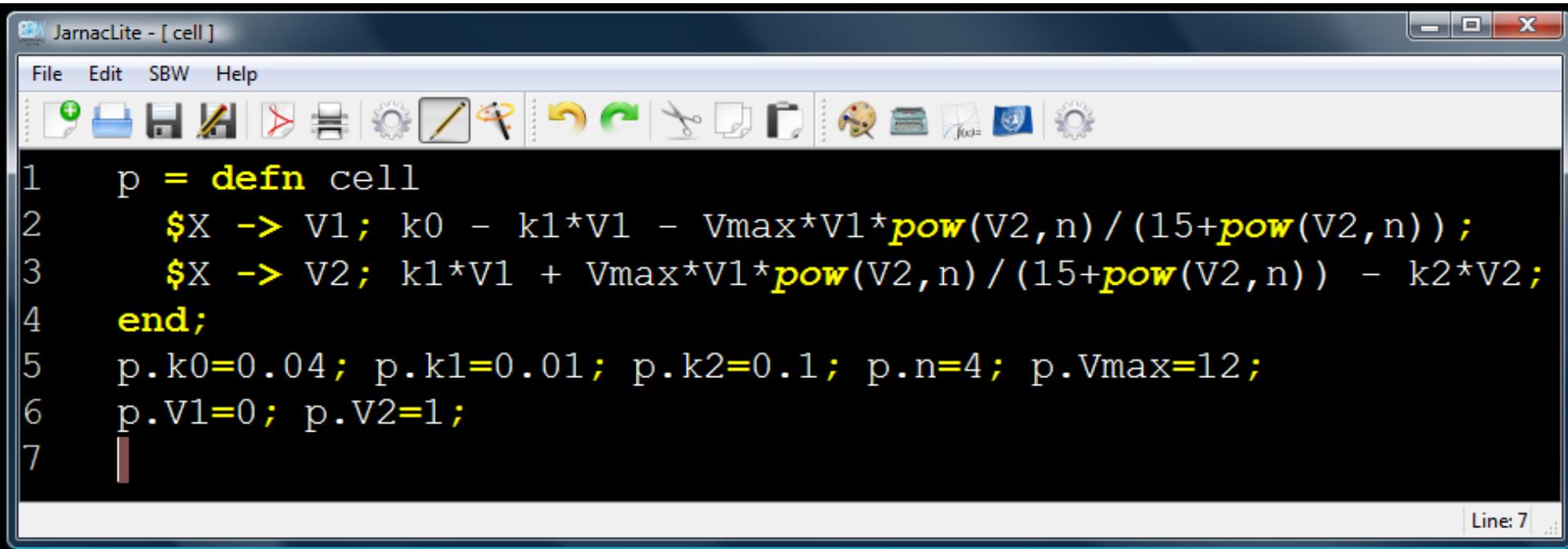


$$\frac{dV_1}{dt} = k_0 - k_1 \cdot V_1 - V_{\max} \cdot V_1 \frac{V_2^n}{15 + V_2^n}$$

$$\frac{dV_2}{dt} = k_1 \cdot V_1 + V_{\max} \cdot V_1 \frac{V_2^n}{15 + V_2^n} - k_2 \cdot V_2$$

First Example – Simple Oscillator

- Load your old exercise in Jarnac Lite.
- If you don't have it, it can be written as:

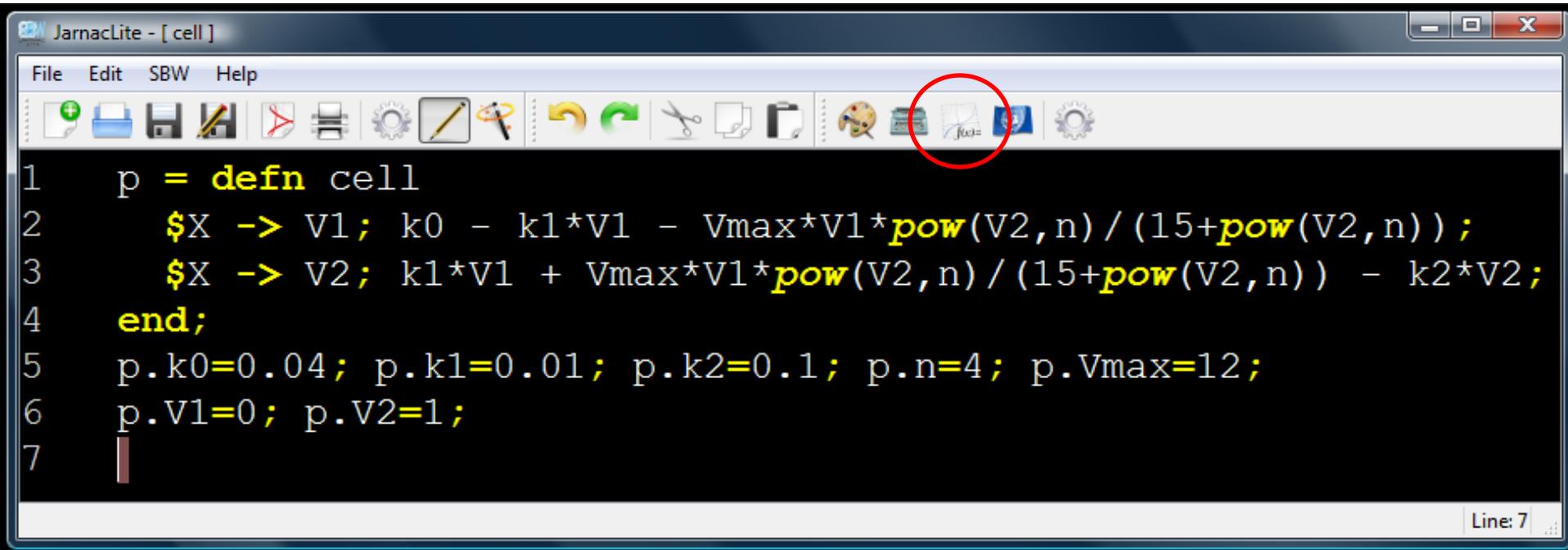


```
JarnacLite - [ cell ]
File Edit SBW Help
1  p = defn cell
2    $X -> V1; k0 - k1*V1 - Vmax*V1*pow(V2,n) / (15*pow(V2,n));
3    $X -> V2; k1*V1 + Vmax*V1*pow(V2,n) / (15*pow(V2,n)) - k2*V2;
4  end;
5  p.k0=0.04; p.k1=0.01; p.k2=0.1; p.n=4; p.Vmax=12;
6  p.V1=0; p.V2=1;
7
```

- Note that V_2^n should be written as pow(V2,n)

First Example – Simple Oscillator

- Click on the Simulation Tool icon:

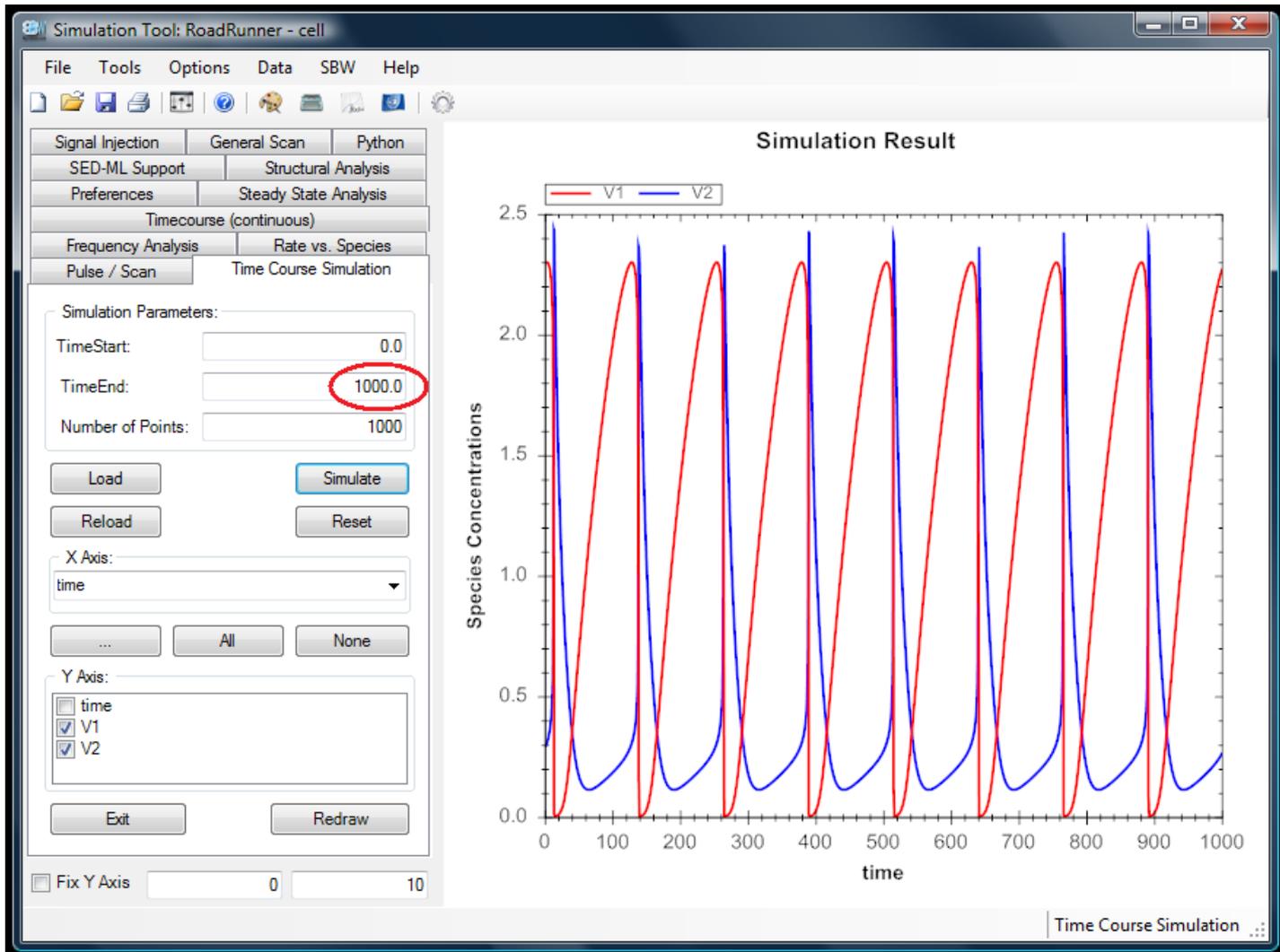


```
1  p = defn cell
2    $X -> V1; k0 - k1*V1 - Vmax*V1*pow(V2,n) / (15+pow(V2,n));
3    $X -> V2; k1*V1 + Vmax*V1*pow(V2,n) / (15+pow(V2,n)) - k2*V2;
4  end;
5  p.k0=0.04; p.k1=0.01; p.k2=0.1; p.n=4; p.Vmax=12;
6  p.V1=0; p.V2=1;
7
```

Line: 7

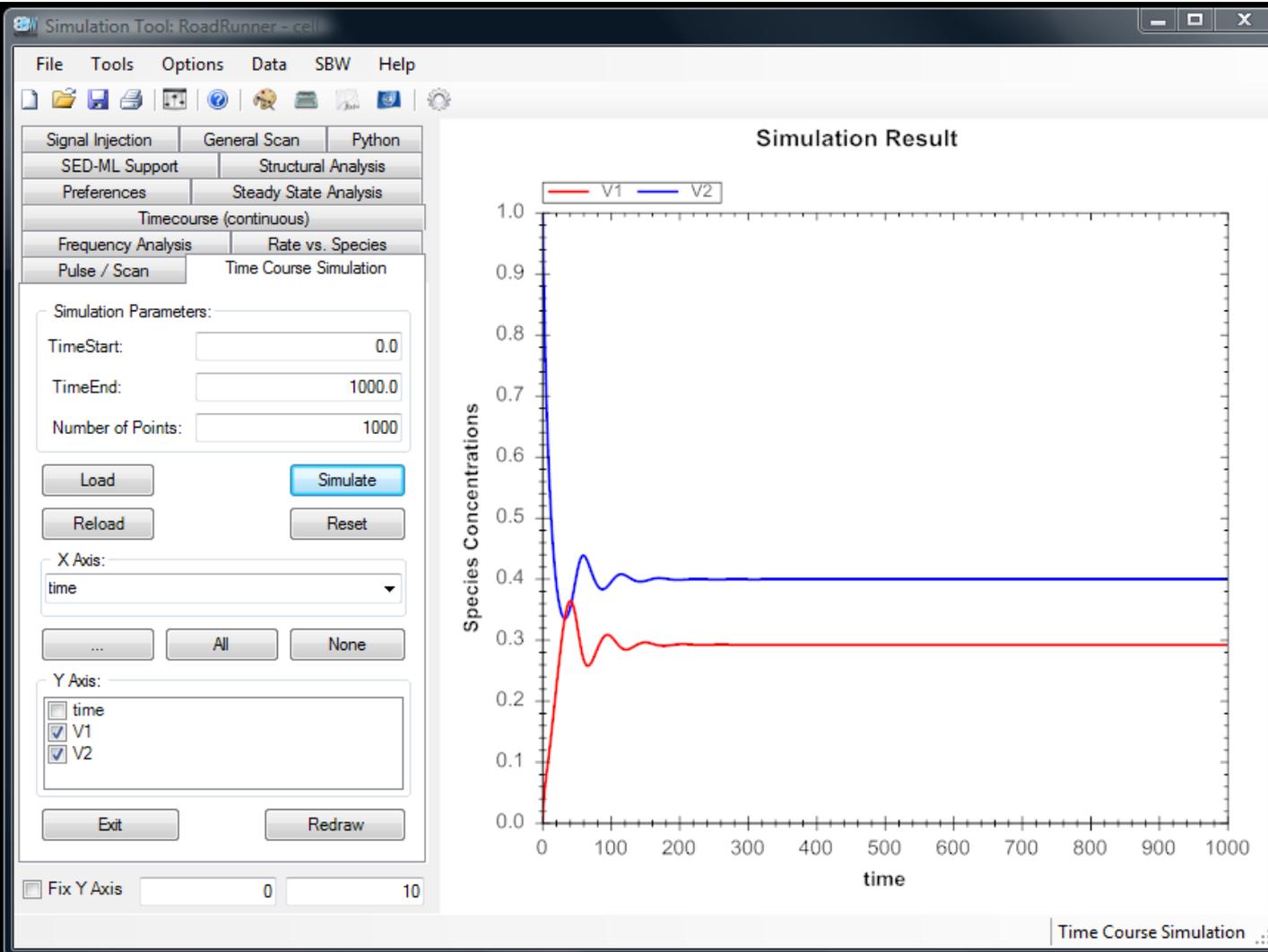
First Example – Simple Oscillator

- The parameter values give oscillations (set TimeEnd to 1000):



Simple Oscillator

- But the oscillations cease if n is changed to 2:



Sliders

To change one of the parameters below, click on the parameter. A slider and a textbox will appear, moving the slider, or entering a value, will instantaneously change the value in the Simulator.

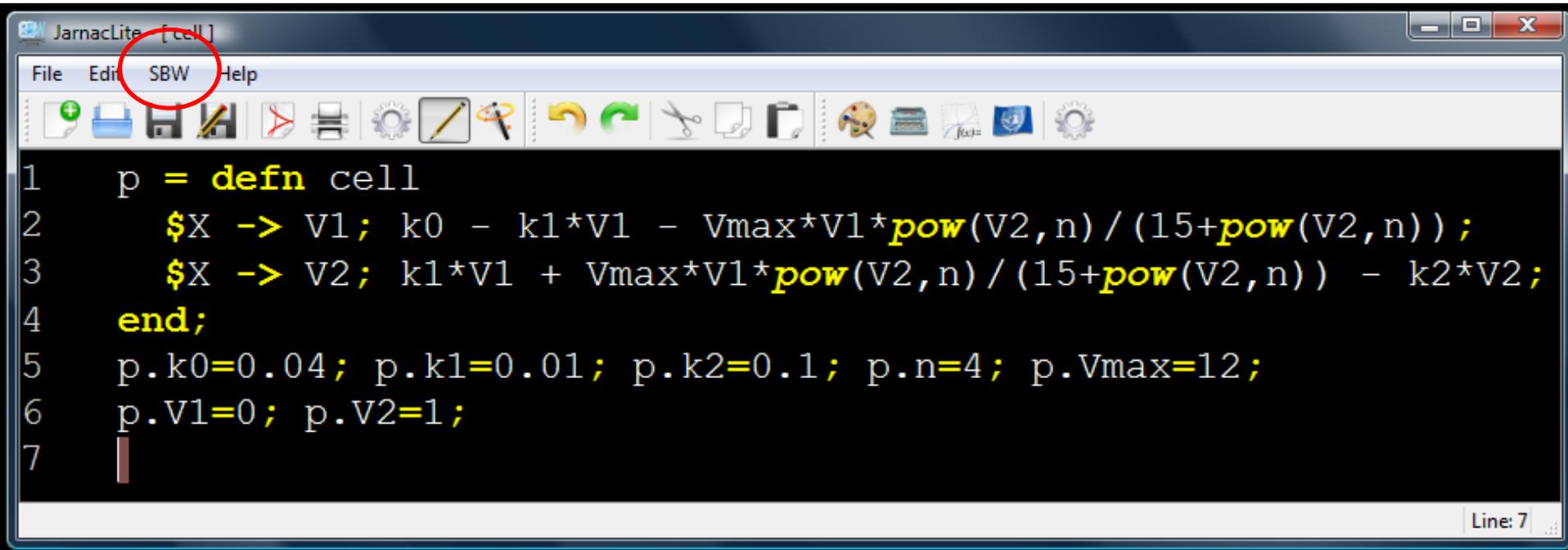
Available Parameters:

- Initial Conditions
- Boundary Species
- Parameters
 - k0: 0.04
 - k1: 0.01
 - Vmax: 12
 - n: 2
 - k2: 0.1

Reset Save Close

First Example – Simple Oscillator

- Go back to Jarnac Lite, click on SBW:



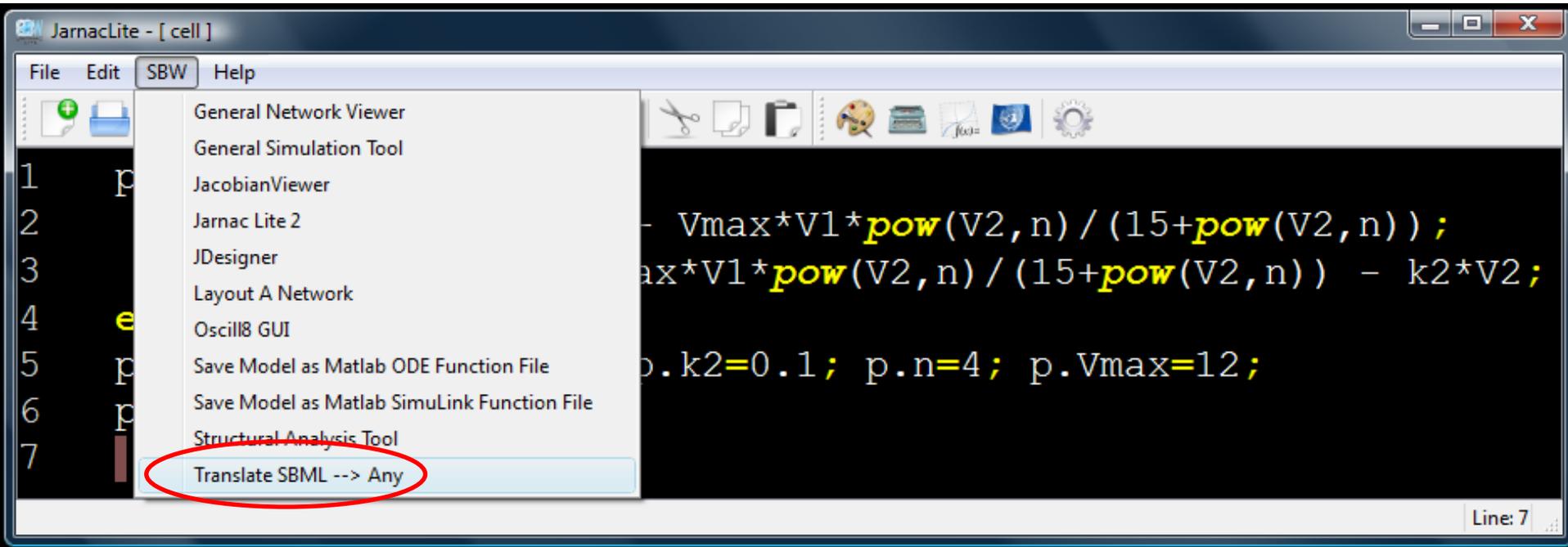
The screenshot shows the Jarnac Lite software interface. The title bar reads "JarnacLite [cell]". The menu bar includes "File", "Edit", "SBW", and "Help". The "SBW" menu item is circled in red. Below the menu bar is a toolbar with various icons. The main editing area contains the following code:

```
1  p = defn cell
2    $X -> V1; k0 - k1*V1 - Vmax*V1*pow(V2,n) / (15+pow(V2,n));
3    $X -> V2; k1*V1 + Vmax*V1*pow(V2,n) / (15+pow(V2,n)) - k2*V2;
4  end;
5  p.k0=0.04; p.k1=0.01; p.k2=0.1; p.n=4; p.Vmax=12;
6  p.V1=0; p.V2=1;
7
```

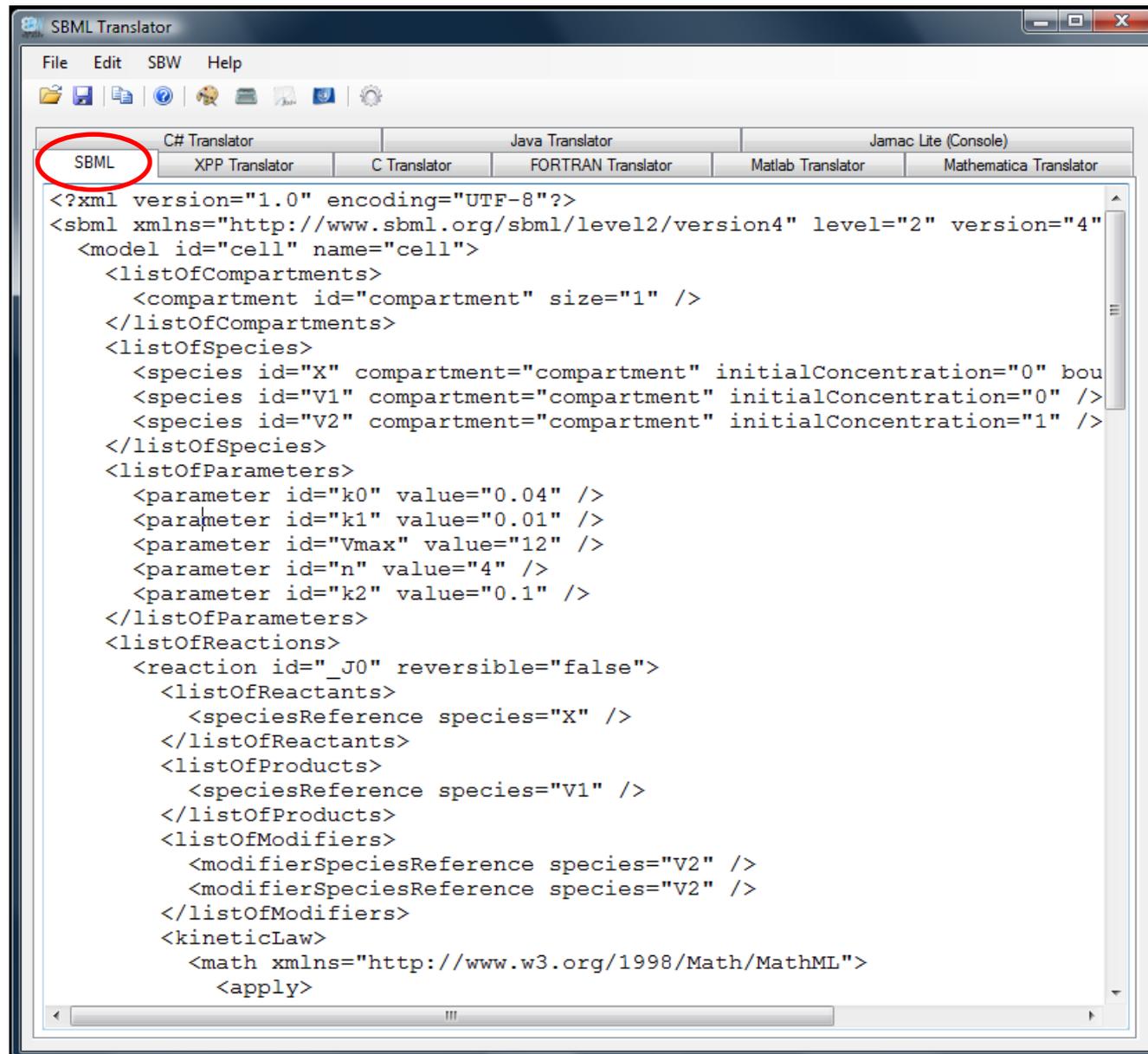
The status bar at the bottom right indicates "Line: 7".

First Example – Simple Oscillator

- Then click on Translate SBML --> Any:



First Example – Simple Oscillator

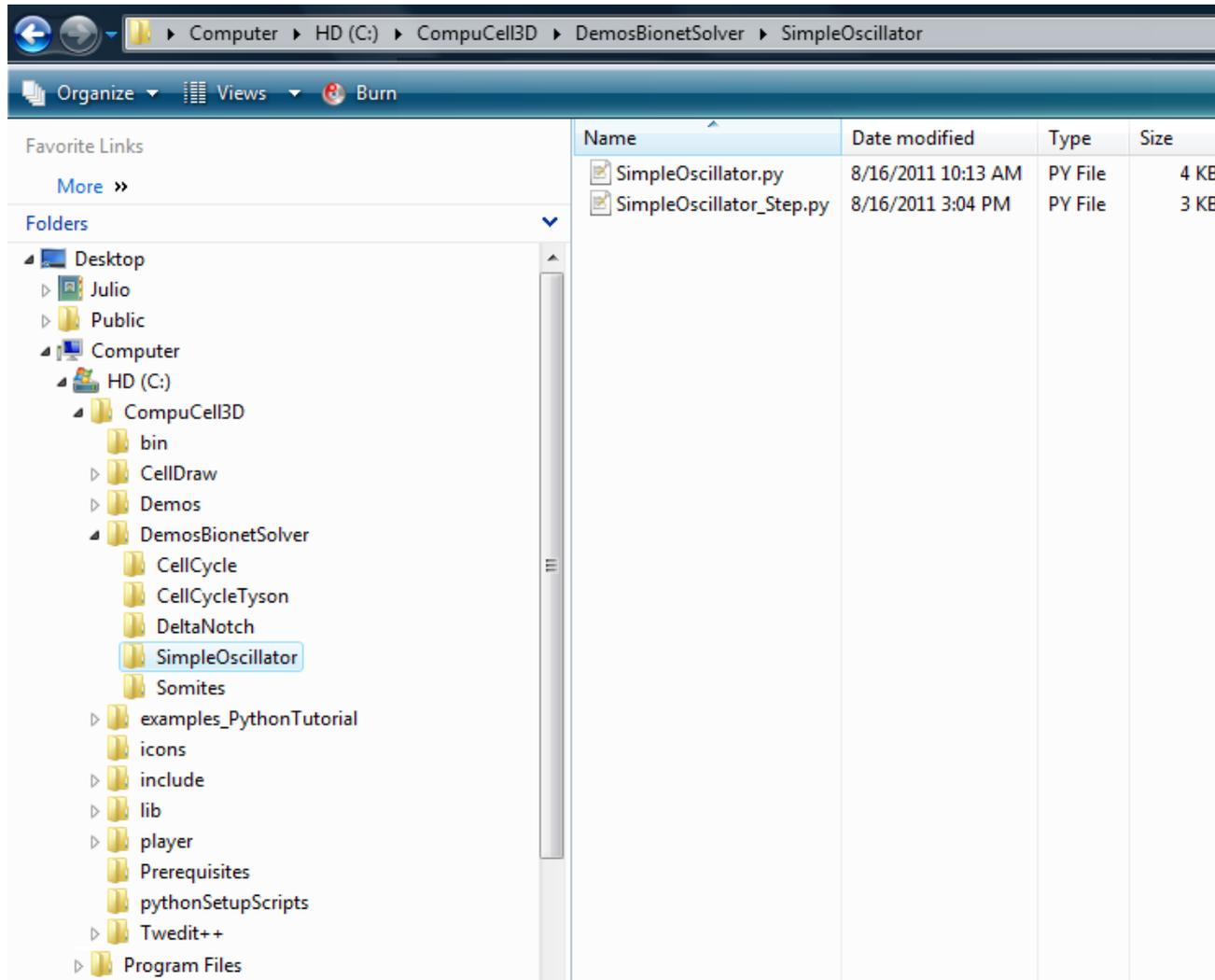


The screenshot shows the SBML Translator application window. The title bar reads "SBML Translator". The menu bar includes "File", "Edit", "SBW", and "Help". The toolbar contains icons for file operations and application settings. The main window is divided into several tabs: "SBML" (highlighted with a red circle), "C# Translator", "Java Translator", "Jamac Lite (Console)", "XPP Translator", "C Translator", "FORTRAN Translator", "Matlab Translator", and "Mathematica Translator". The SBML tab is active and displays the following XML code:

```
<?xml version="1.0" encoding="UTF-8"?>
<sbml xmlns="http://www.sbml.org/sbml/level2/version4" level="2" version="4"
  <model id="cell" name="cell">
  <listOfCompartments>
    <compartment id="compartment" size="1" />
  </listOfCompartments>
  <listOfSpecies>
    <species id="X" compartment="compartment" initialConcentration="0" bou
    <species id="V1" compartment="compartment" initialConcentration="0" />
    <species id="V2" compartment="compartment" initialConcentration="1" />
  </listOfSpecies>
  <listOfParameters>
    <parameter id="k0" value="0.04" />
    <parameter id="k1" value="0.01" />
    <parameter id="Vmax" value="12" />
    <parameter id="n" value="4" />
    <parameter id="k2" value="0.1" />
  </listOfParameters>
  <listOfReactions>
    <reaction id="_J0" reversible="false">
      <listOfReactants>
        <speciesReference species="X" />
      </listOfReactants>
      <listOfProducts>
        <speciesReference species="V1" />
      </listOfProducts>
      <listOfModifiers>
        <modifierSpeciesReference species="V2" />
        <modifierSpeciesReference species="V2" />
      </listOfModifiers>
      <kineticLaw>
        <math xmlns="http://www.w3.org/1998/Math/MathML">
          <apply>
```

First Example – Simple Oscillator

- Save it as “*SimpleOscillator.sbml*” inside:
CompuCell3D\DemosBionetSolver\SimpleOscillator



First Example – Simple Oscillator

- The *SimpleOscillator.py* file contains the parameters of the cellular model.
- It is important to note that in this model there are 2 cell types: “TypeA” and “TypeB”

```
34 ...
35 ... cellType=cc3d.ElementCC3D("Plugin",{"Name":"CellType"})
36 ... cellType.ElementCC3D("CellType",{"TypeName":"Medium","TypeId":"0"})
37 ... cellType.ElementCC3D("CellType",{"TypeName":"TypeA","TypeId":"1"})
38 ... cellType.ElementCC3D("CellType",{"TypeName":"TypeB","TypeId":"2"})
39
40 ... #CONTACT
41 ... contact=cc3d.ElementCC3D("Plugin",{"Name":"Contact"})
42 ... contact.ElementCC3D("Energy",{"Type1":"Medium","Type2":"Medium"},0)
43 ... contact.ElementCC3D("Energy",{"Type1":"Medium","Type2":"TypeA"},10)
44 ... contact.ElementCC3D("Energy",{"Type1":"Medium","Type2":"TypeB"},10)
45 ... contact.ElementCC3D("Energy",{"Type1":"TypeA","Type2":"TypeA"},10)
46 ... contact.ElementCC3D("Energy",{"Type1":"TypeA","Type2":"TypeB"},10)
47 ... contact.ElementCC3D("Energy",{"Type1":"TypeB","Type2":"TypeB"},10)
48 ... #-neighbor order
49 ... contact.ElementCC3D("NeighborOrder",{},nOrder)
50 ...
```

First Example – Simple Oscillator

- The *SimpleOscillator.py* file calls 3 steppables:
 - InitCond: where the initial conditions are set
 - Oscillator: where the SBML model will be loaded
 - ExtraFields: used to visualize one of the model's variable

```
81 from SimpleOscillator_Step import InitCond
82 initCond=InitCond(_simulator=sim,_frequency=1,_LamV=LamV,_tV=tV)
83 steppableRegistry.registerSteppable(initCond)
84
85 from SimpleOscillator_Step import Oscillator
86 oscillator=Oscillator(_simulator=sim,_frequency=1)
87 steppableRegistry.registerSteppable(oscillator)
88
89 #Create extra player fields here or add attributes
90 dim=sim.getPotts().getCellFieldG().getDim()
91 Field=simthread.createScalarFieldCellLevelPy("Osc")
92
93 from SimpleOscillator_Step import ExtraFields
94 extraFields=ExtraFields(_simulator=sim,_frequency=5)
95 extraFields.setScalarFields(Field)
96 steppableRegistry.registerSteppable(extraFields)
97
98 CompuCellSetup.mainLoop(sim,simthread,steppableRegistry)
99 ##sys.exit()
```

First Example – Simple Oscillator

- Let's open the *SimpleOscillator_Step.py* file and look at the beginning of the Oscillator steppable:

```
21 import bionetAPI
22 class Oscillator(SteppableBasePy):
23     def __init__(self, _simulator, _frequency):
24         SteppableBasePy.__init__(self, _simulator, _frequency)
25         bionetAPI.initializeBionetworkManager(self.simulator)
26
27     def start(self):
28         #Loading model
29         ModelName = "SimpleOscillator"
30         ModelKey = "SO"
31         ModelPath = os.getcwd() + "/DemosBionetSolver/SimpleOscillator/SimpleOscillator.sbml"
32         IntegrationStep = 1.0
33         bionetAPI.loadSBMLModel(ModelName, ModelPath, ModelKey, IntegrationStep)
34
35         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeA")
36         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeB")
37
38         #Initial conditions
39         bionetAPI.initializeBionetworks()
40         # bionetAPI.setBionetworkValue("SO_n", 2, "TypeB")
```

First Example – Simple Oscillator

- The first thing to be done is to import the BionetSolver library by this command:

```
21 import bionetAPI ←
22 class Oscillator(SteppableBasePy):
23     def __init__(self, _simulator, _frequency):
24         SteppableBasePy.__init__(self, _simulator, _frequency)
25         bionetAPI.initializeBionetworkManager(self.simulator)
26
27     def start(self):
28         #Loading model
29         ModelName = "SimpleOscillator"
30         ModelKey = "SO"
31         ModelPath = os.getcwd() + "/DemosBionetSolver/SimpleOscillator/SimpleOscillator.sbml"
32         IntegrationStep = 1.0
33         bionetAPI.loadSBMLModel(ModelName, ModelPath, ModelKey, IntegrationStep)
34
35         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeA")
36         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeB")
37
38         #Initial conditions
39         bionetAPI.initializeBionetworks()
40         # bionetAPI.setBionetworkValue("SO_n", 2, "TypeB")
```

First Example – Simple Oscillator

- Next, inside the steppable, we initialize the solver by using this command:

```
21 import bionetAPI
22 class Oscillator(SteppableBasePy):
23     def __init__(self, _simulator, _frequency):
24         SteppableBasePy.__init__(self, _simulator, _frequency)
25         bionetAPI.initializeBionetworkManager(self.simulator)
26
27     def start(self):
28         #Loading model
29         ModelName = "SimpleOscillator"
30         ModelKey = "SO"
31         ModelPath = os.getcwd() + "/DemosBionetSolver/SimpleOscillator/SimpleOscillator.sbml"
32         IntegrationStep = 1.0
33         bionetAPI.loadSBMLModel(ModelName, ModelPath, ModelKey, IntegrationStep)
34
35         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeA")
36         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeB")
37
38         #Initial conditions
39         bionetAPI.initializeBionetworks()
40         # bionetAPI.setBionetworkValue("SO_n", 2, "TypeB")
```

First Example – Simple Oscillator

- Once the BionetSolver is loaded and initialized, it is time to load the model:

```
21 import bionetAPI
22 class Oscillator(SteppableBasePy):
23     def __init__(self, _simulator, _frequency):
24         SteppableBasePy.__init__(self, _simulator, _frequency)
25         bionetAPI.initializeBionetworkManager(self.simulator)
26
27     def start(self):
28         #Loading model
29         ModelName = "SimpleOscillator"
30         ModelKey = "SO"
31         ModelPath = os.getcwd() + "/DemosBionetSolver/SimpleOscillator/SimpleOscillator.sbml"
32         IntegrationStep = 1.0
33         bionetAPI.loadSBMLModel(ModelName, ModelPath, ModelKey, IntegrationStep) ←
34
35         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeA")
36         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeB")
37
38         #Initial conditions
39         bionetAPI.initializeBionetworks()
40         # bionetAPI.setBionetworkValue("SO_n", 2, "TypeB")
```

First Example – Simple Oscillator

- This takes 4 parameters:

- First is the model name: _____

```
21 import bionetAPI
22 class Oscillator(SteppableBasePy):
23     def __init__(self, _simulator, _frequency):
24         SteppableBasePy.__init__(self, _simulator, _frequency)
25         bionetAPI.initializeBionetworkManager(self.simulator)
26
27     def start(self):
28         #Loading model
29         ModelName = "SimpleOscillator" ←
30         ModelKey = "SO"
31         ModelPath = os.getcwd() + "/DemosBionetSolver/SimpleOscillator/SimpleOscillator.sbml"
32         IntegrationStep = 1.0
33         bionetAPI.loadSBMLModel(ModelName, ModelPath, ModelKey, IntegrationStep)
34
35         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeA")
36         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeB")
37
38         #Initial conditions
39         bionetAPI.initializeBionetworks()
40         # bionetAPI.setBionetworkValue("SO_n", 2, "TypeB")
```

- For convenience the name will be SimpleOscillator, but it can be anything.

First Example – Simple Oscillator

- This takes 4 parameters:
 - Second is the model pathway:

```
21 import bionetAPI
22 class Oscillator(SteppableBasePy):
23     def __init__(self, _simulator, _frequency):
24         SteppableBasePy.__init__(self, _simulator, _frequency)
25         bionetAPI.initializeBionetworkManager(self.simulator)
26
27     def start(self):
28         #Loading model
29         ModelName = "SimpleOscillator"
30         ModelKey = "SO"
31         ModelPath = os.getcwd() + "/DemosBionetSolver/SimpleOscillator/SimpleOscillator.sbml"
32         IntegrationStep = 1.0
33         bionetAPI.loadSBMLModel(ModelName, ModelPath, ModelKey, IntegrationStep)
34
35         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeA")
36         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeB")
37
38         #Initial conditions
39         bionetAPI.initializeBionetworks()
40         # bionetAPI.setBionetworkValue("SO_n", 2, "TypeB")
```

- `os.getcwd()` gives the CompuCell3D root directory.
- Here it is crucial that the correct path and model name are given.

First Example – Simple Oscillator

- This takes 4 parameters:
 - Third is the model nickname:

```
21 import bionetAPI
22 class Oscillator(SteppableBasePy):
23     def __init__(self, _simulator, _frequency):
24         SteppableBasePy.__init__(self, _simulator, _frequency)
25         bionetAPI.initializeBionetworkManager(self.simulator)
26
27     def start(self):
28         #Loading model
29         ModelName = "SimpleOscillator"
30         ModelKey = "SO" ←
31         ModelPath = os.getcwd() + "/DemosBionetSolver/SimpleOscillator/SimpleOscillator.sbml"
32         IntegrationStep = 1.0
33         bionetAPI.loadSBMLModel(ModelName, ModelPath, ModelKey, IntegrationStep)
34
35         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeA")
36         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeB")
37
38         #Initial conditions
39         bionetAPI.initializeBionetworks()
40         # bionetAPI.setBionetworkValue("SO_n", 2, "TypeB")
```

- This is used as an abbreviation of the model name when referring to parameters of this model.

First Example – Simple Oscillator

- This takes 4 parameters:
 - And the last one is the size of the integration step:

```
21 import bionetAPI
22 class Oscillator(SteppableBasePy):
23     def __init__(self, _simulator, _frequency):
24         SteppableBasePy.__init__(self, _simulator, _frequency)
25         bionetAPI.initializeBionetworkManager(self.simulator)
26
27     def start(self):
28         #Loading model
29         ModelName = "SimpleOscillator"
30         ModelKey = "SO"
31         ModelPath = os.getcwd() + "/DemosBionetSolver/SimpleOscillator/SimpleOscillator.sbml"
32         IntegrationStep = 1.0 ←
33         bionetAPI.loadSBMLModel(ModelName, ModelPath, ModelKey, IntegrationStep)
34
35         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeA")
36         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeB")
37
38         #Initial conditions
39         bionetAPI.initializeBionetworks()
40         # bionetAPI.setBionetworkValue("SO_n", 2, "TypeB")
```

- This specifies the correspondence between MCS and the unit of time of the model.

First Example – Simple Oscillator

- Now we have to add this model to the cells in our simulation:

```
21 import bionetAPI
22 class Oscillator(SteppableBasePy):
23     def __init__(self, _simulator, _frequency):
24         SteppableBasePy.__init__(self, _simulator, _frequency)
25         bionetAPI.initializeBionetworkManager(self.simulator)
26
27     def start(self):
28         #Loading model
29         ModelName = "SimpleOscillator"
30         ModelKey = "SO"
31         ModelPath = os.getcwd() + "/DemosBionetSolver/SimpleOscillator/SimpleOscillator.sbml"
32         IntegrationStep = 1.0
33         bionetAPI.loadSBMLModel(ModelName, ModelPath, ModelKey, IntegrationStep)
34
35         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeA")
36         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeB") ←
37
38         #Initial conditions
39         bionetAPI.initializeBionetworks()
40         # bionetAPI.setBionetworkValue("SO_n", 2, "TypeB")
```

- The first line add the SimpleOscillator model to all cells of type “TypeA”, and the second to all cells of type “TypeB”.

First Example – Simple Oscillator

- Finally, we initialize the SBML model in each cell by using the following command:

```
21 import bionetAPI
22 class Oscillator(SteppableBasePy):
23     def __init__(self, _simulator, _frequency):
24         SteppableBasePy.__init__(self, _simulator, _frequency)
25         bionetAPI.initializeBionetworkManager(self.simulator)
26
27     def start(self):
28         #Loading model
29         ModelName = "SimpleOscillator"
30         ModelKey = "SO"
31         ModelPath = os.getcwd() + "/DemosBionetSolver/SimpleOscillator/SimpleOscillator.sbml"
32         IntegrationStep = 1.0
33         bionetAPI.loadSBMLModel(ModelName, ModelPath, ModelKey, IntegrationStep)
34
35         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeA")
36         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeB")
37
38         #Initial conditions
39         bionetAPI.initializeBionetworks() ←
40         # bionetAPI.setBionetworkValue("SO_n", 2, "TypeB")
```

First Example – Simple Oscillator

- On the step function we create a dictionary where the two variables of the SimpleOscillator will be stored:

```
41
42 def step(self,mcs):
43     for cell in self.cellList:
44         if (cell):
45             #Storing V1 and V2 levels of each cell
46             cellDict=CompuCell.getPyAttrib(cell) ←
47             cellDict["V1"]=bionetAPI.getBionetworkValue("SO_V1",cell.id)
48             cellDict["V2"]=bionetAPI.getBionetworkValue("SO_V2",cell.id)
49             #Iterating the ODEs
50             bionetAPI.timestepBionetworks()
51
```

First Example – Simple Oscillator

- To extract the current value of a variable or parameter from the SBML model inside the cell, we use the command:

```
41
42 def step(self,mcs):
43     for cell in self.cellList:
44         if (cell):
45             #Storing V1 and V2 levels of each cell
46             cellDict=CompuCell.getPyAttrib(cell)
47             cellDict["V1"]=bionetAPI.getBionetworkValue("SO_V1",cell.id)
48             cellDict["V2"]=bionetAPI.getBionetworkValue("SO_V2",cell.id)
49             #Iterating the ODEs
50             bionetAPI.timestepBionetworks()
51
```

- Where the first parameter indicate the model (by its nickname) followed by an underscore “_” and the name of the variable.
- The second parameter indicate the cell from which this information will be extracted.

First Example – Simple Oscillator

- The last command runs the SBML model inside each cell for one time step of integration:

```
41
42 def step(self,mcs):
43     for cell in self.cellList:
44         if (cell):
45             #Storing V1 and V2 levels of each cell
46             cellDict=CompuCell.getPyAttrib(cell)
47             cellDict["V1"]=bionetAPI.getBionetworkValue("SO_V1",cell.id)
48             cellDict["V2"]=bionetAPI.getBionetworkValue("SO_V2",cell.id)
49             #Iterating the ODEs
50             bionetAPI.timestepBionetworks() ←
51
```

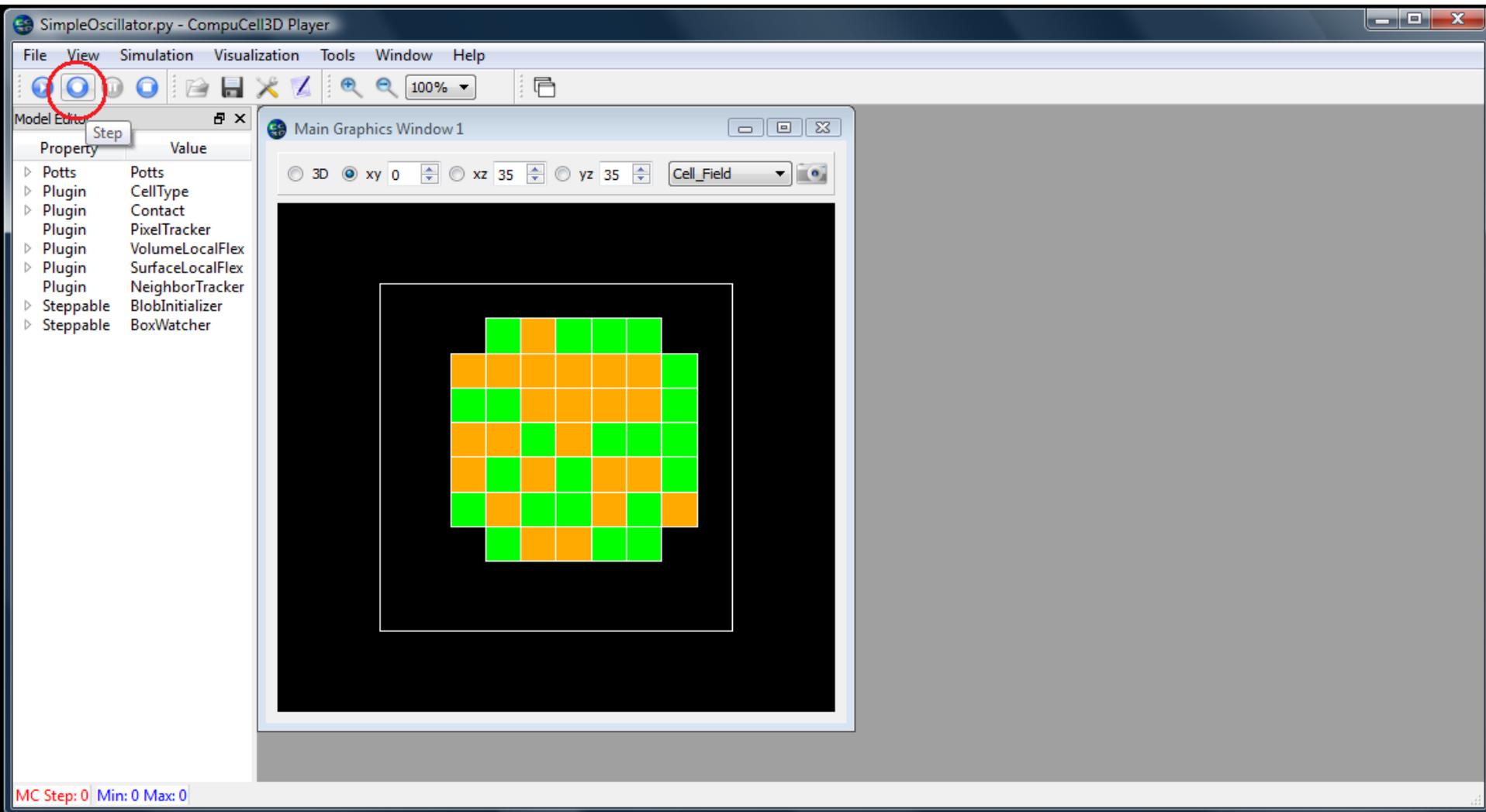
- If this command is not called, the ODE model will not run and all variables will stay at their initial values.

First Example – Simple Oscillator

- Next, open the file *SimpleOscillator.py* on the CC3D player.

First Example – Simple Oscillator

- Hit the “step” button, as indicated below, to run the simulation just one MCS.



First Example – Simple Oscillator

- Click the indicated button to open a new graphics window and select *V1* in place of *Cell_field* in it:

SimpleOscillator.py - CompuCell3D Player

File View Simulation Visualization Tools Window Help

Model Editor

Property	Value
▶ Potts	Potts
▶ Plugin	CellType
▶ Plugin	Contact
▶ Plugin	PixelTracker
▶ Plugin	VolumeLocalFlex
▶ Plugin	SurfaceLocalFlex
▶ Plugin	NeighborTracker
▶ Steppable	BlobInitializer
▶ Steppable	BoxWatcher

Main Graphics Window 1

3D xy 0 xz 35 yz 35 Cell_Field

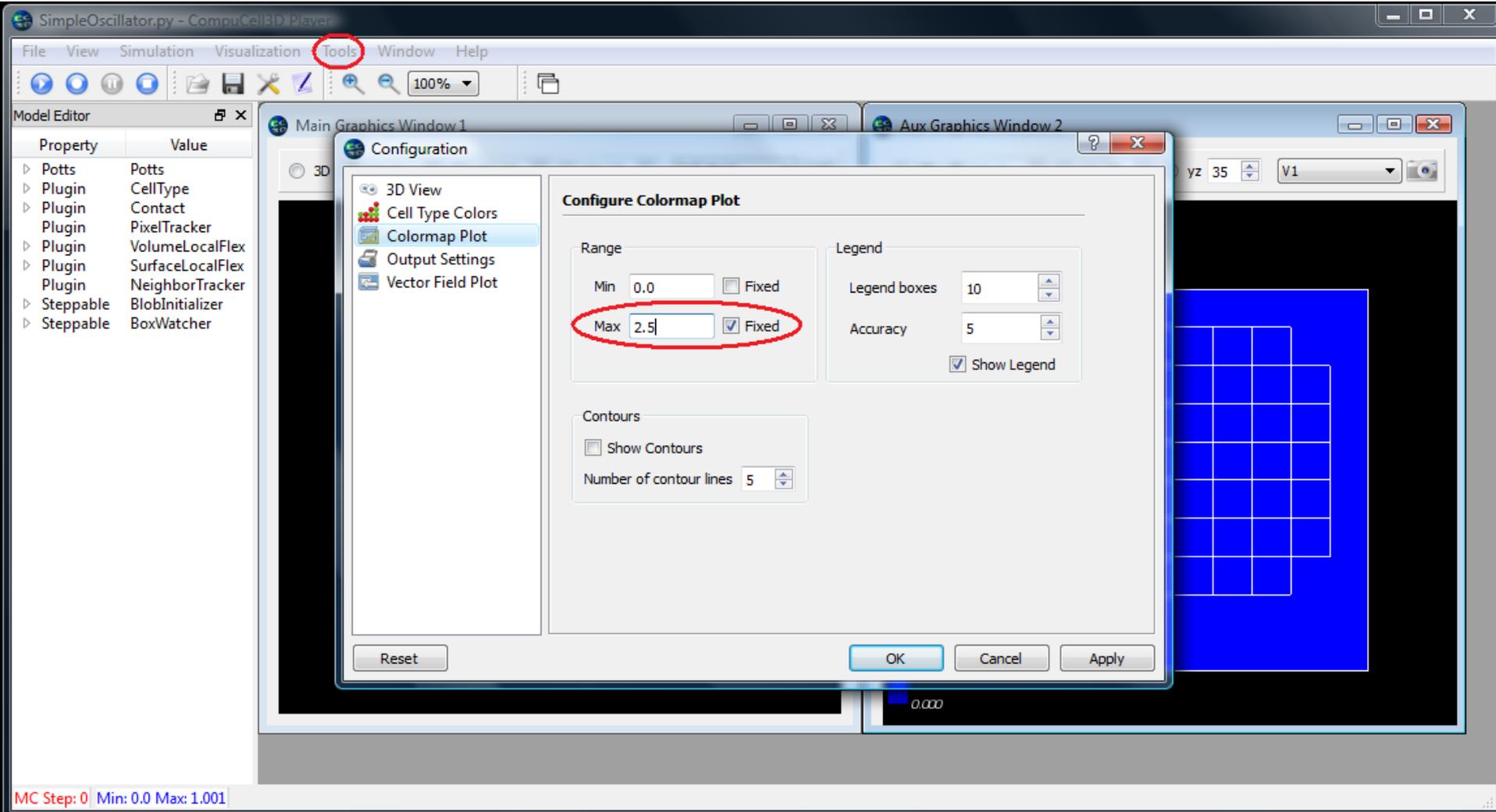
Aux Graphics Window 2

3D xy 0 xz 35 yz 35 V1

MC Step: 0 Min: 0.0 Max: 1.001

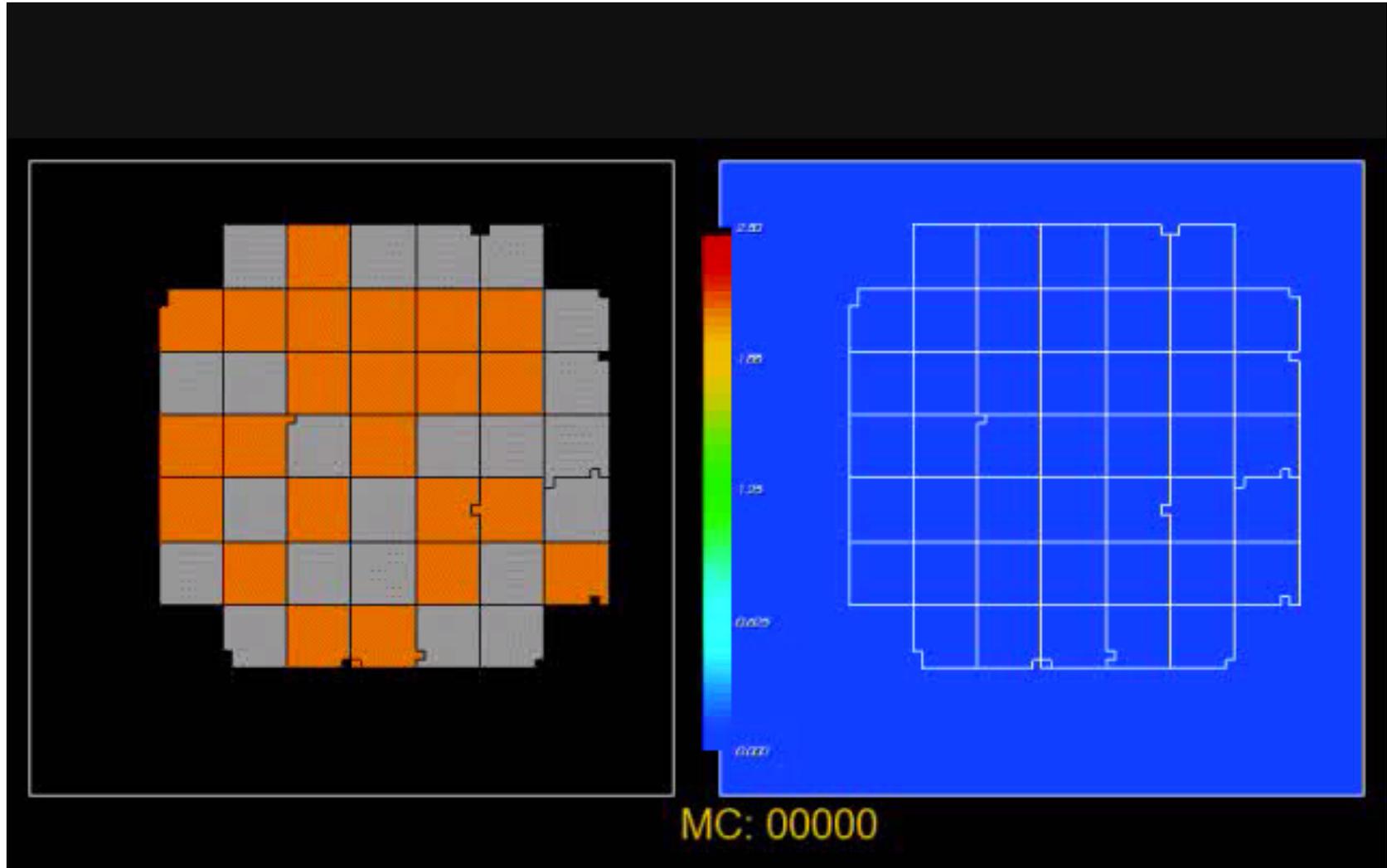
First Example – Simple Oscillator

- Next go to Tools -> Configuration, and under “Colormap Plot”, fix the maximum range at 2.5:



First Example – Simple Oscillator

- When you run the simulation, the second graphics window should display an oscillating V1 concentration similar to this:



Second Example – Simple Oscillator 2

- In the last example both cell types, “TypeA” and “TypeB”, had exactly the same oscillator with the same parameters and initial conditions.
- In this second example we are going to assign the same SBML model to both cell types, but change the parameter “n” for one of them.

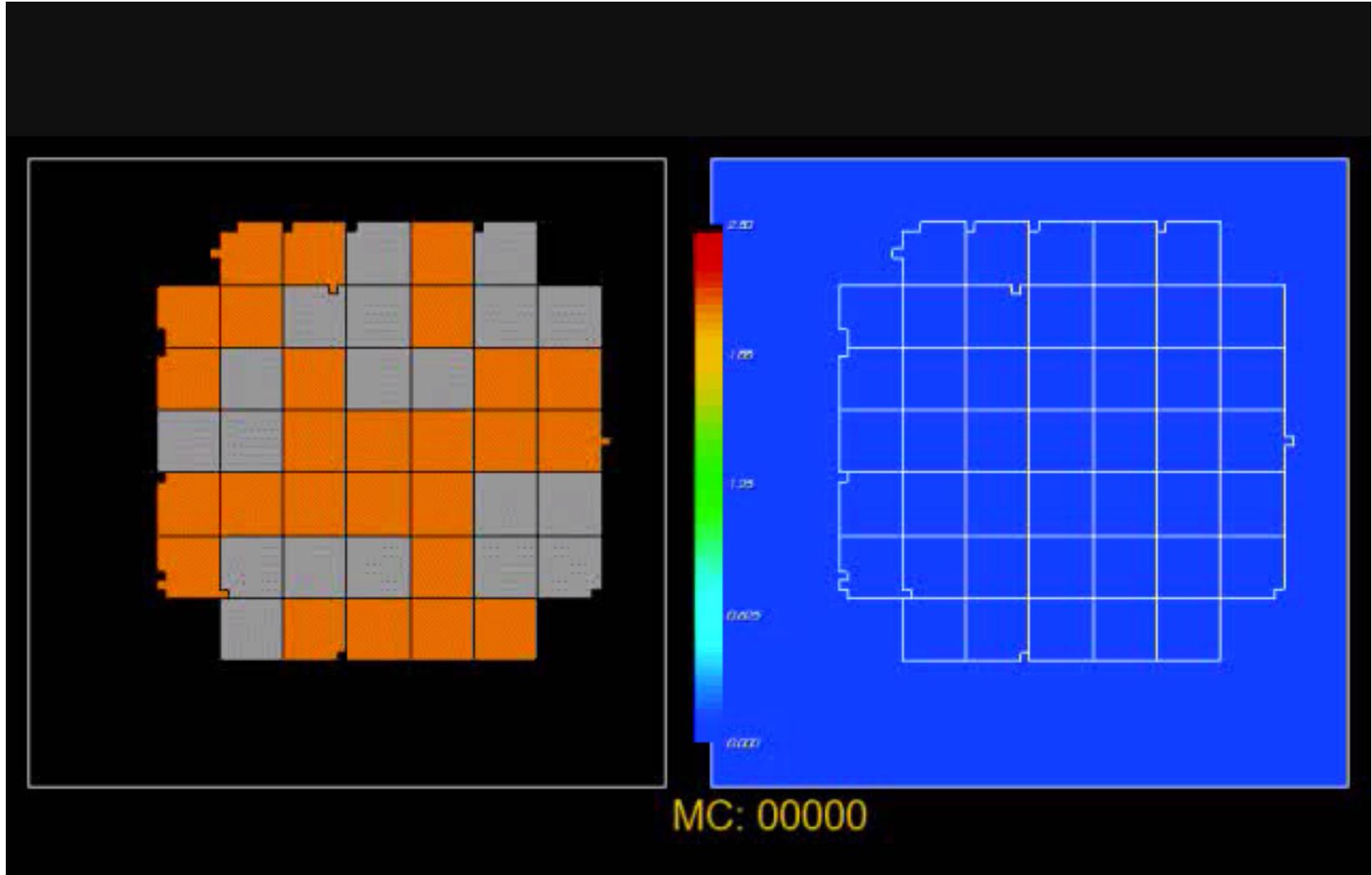
Second Example – Simple Oscillator 2

- To do this, uncomment the following line from the steppable file *SimpleOscillator_Step.py*:

```
21 import bionetAPI
22 class Oscillator(SteppableBasePy):
23     def __init__(self, _simulator, _frequency):
24         SteppableBasePy.__init__(self, _simulator, _frequency)
25         bionetAPI.initializeBionetworkManager(self.simulator)
26
27     def start(self):
28         #Loading model
29         ModelName = "SimpleOscillator"
30         ModelKey = "SO"
31         ModelPath = os.getcwd() + "/DemosBionetSolver/SimpleOscillator/SimpleOscillator.sbml"
32         IntegrationStep = 1.0
33         bionetAPI.loadSBMLModel(ModelName, ModelPath, ModelKey, IntegrationStep)
34
35         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeA")
36         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeB")
37
38         #Initial conditions
39         #bionetAPI.initializeBionetworks()
40         bionetAPI.setBionetworkValue("SO_n", 2, "TypeB") ←
41
```

Second Example – Simple Oscillator 2

- As a result, cells of type “TypeB” will cease to oscillate:



Third Example – Cell Cycle from web

- In our third example, instead of building our own SBML model, we are going to use an existing one.
- The website www.sbml.org contains a repository of published models in SBML format.
- If you wish to submit your own SBML to the repository, follow the instructions at:
www.ebi.ac.uk/biomodels-main/submit

Third Example – Cell Cycle from web

- To access the SBML model repository click on the link “BioModels Database” and then on “Curated models”:



The Systems Biology

News Documents Downloads Forums Facilities Community Events

Welcome to the portal for the **Systems Biology Markup Language (SBML)**, a free and open interchange format for computer models of biological processes. SBML is useful for models of metabolism, cell signaling, and more. It has been in development by an international community since the year 2000.



For the curious

What *is* SBML? Read our [introduction](#), then perhaps browse the [mailing lists](#) to glimpse what's happening with SBML today.



For modelers

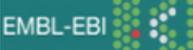
Looking for software that supports SBML? Our [software guide](#) lists over **210 systems**. Are you instead looking for models? Visit [BioModels Database](#) where you can find hundreds!



For software developers

Interested in supporting SBML in your software? Read our [basic introduction](#) and then the [SBML specifications](#) to understand how to use SBML. After that, you may want to look at [libSBML](#).

No matter how you use SBML, we invite you to sign up for news updates either through our [RSS feed](#), our [Twitter feed](#), or one of the [mailing lists](#), and get involved with [community efforts](#) to help keep improving SBML. You can also call attention to your project's support of SBML by displaying the [SBML logo](#).



Enter Text Here Find

Databases Tools Research Training Industry About Us Help

BioModels Home Models Submit Support About BioModels Contact us

BioModels Database - A Database of Annotated Published Models

BioModels Database is a repository of peer-reviewed, published, computational models. These mathematical models are published mathematical models. In addition, models in the database can be used to generate sub-models, can be

All unmodified models in the database are available freely for use and distribution, to all users. This resource is dev

Browse models

[Curated models \(326\)](#)

- [Browse models using GO](#)
- [Non-curated models \(373\)](#)

Simulate in JWS Online

Submit a model

Third Example – Cell Cycle from web

- From the model list select the third one by clicking on the link under the column “BioModels ID”

[BioModels Home](#) [Models](#) [Submit](#) [Support](#) [About BioModels](#) [Contact us](#)

Browse - Curated models

☐ The following fields are used to describe a model:

- *BioModels ID* → A unique string of characters associated with the model, which will never be re-used even if the model is deleted from the BioModels Database.
- *Name* → The name of the model, as written in the model itself by its creator(s).
- *Publication ID* → The unique identifier of the reference publication describing the model, specified either as a [PubMed](#) identifier (linked to the EBI Medline database), or as a [DOI](#) (linked to the original must have one publication identifier, and the same identifier can be shared amongst several models if they have been described in the same publication).
- *Last Modified* → The date when the model was last modified.

To view a model, simply click on the correspondent BioModels ID provided within the leftmost column of the row corresponding to the model.

← 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 →

BioModels ID ▾	Name	Publication ID
BIOMD0000000001	Edelstein1996_EPSP_AChEvent	8983160
BIOMD0000000002	Edelstein1996_EPSP_AChSpecies	8983160
BIOMD0000000003	Goldbeter1991_MinMitOscil	1833774
BIOMD0000000004	Goldbeter1991_MinMitOscil_ExplInact	1833774
BIOMD0000000005	Tyson1991_CellCycle_6var	1831270
BIOMD0000000006	Tyson1991_CellCycle_2var	1831270
BIOMD0000000007	Novak1997_CellCycle	9256450
BIOMD0000000008	Gardner1998_CellCycle_Goldbeter	9826676
BIOMD0000000009	Huang1996_MAPK_ultrasens	8816754
BIOMD0000000010	Kholodenko2000_MAPK_feedback	10712587

Third Example – Cell Cycle from web

- To download the model click on “Download SBML” and select “SBML L2 V4 (curated)”

BioModels Home Models Submit Support About BioModels Contact us

BIOMD0000000003 - Goldbeter1991_MinMitOscil

[Download SBML](#) | [Other formats \(auto-generated\)](#) | [Actions](#) | [Submit Model Comment/Bug](#)

[SBML L2 V1 \(auto-generated\)](#) | [Overview](#) | [Math](#) | [Physical entities](#) | [Parameters](#) | [Curation](#)

[SBML L2 V2 \(auto-generated\)](#)

[SBML L2 V3 \(auto-generated\)](#)

[SBML L2 V4 \(curated\)](#) **Reference Publication**

Publication ID: [1833774](#)

Proc Natl Acad Sci U S A 1991 Oct;88(20):9107-11.
A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 kinase.
Goldbeter A.
Faculté des Sciences, Université Libre de Bruxelles, Belgium. [\[more\]](#)

Model

Original Model: [BIOMD0000000003.xml.origin](#)

Submitter: [Nicolas Le Novère](#)

Submission ID: MODEL6614271263

Submission Date: 13 Sep 2005 12:24:56 UTC

Last Modification Date: 17 Mar 2010 00:25:38 UTC

Creation Date: 06 Feb 2005 23:39:40 UTC

Encoders: [Bruce Shapiro](#)
[Vijayalakshmi Chelliah](#)

set #1	bqbiol:occursIn	Taxonomy Amphibia
set #2	bqbiol:isVersionOf	KEGG Pathway hsa04110 Gene Ontology mitotic cell cycle
	bqbiol:isHomologTo	Reactome REACT_152

Notes

This a model from the article:

A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 kinase.

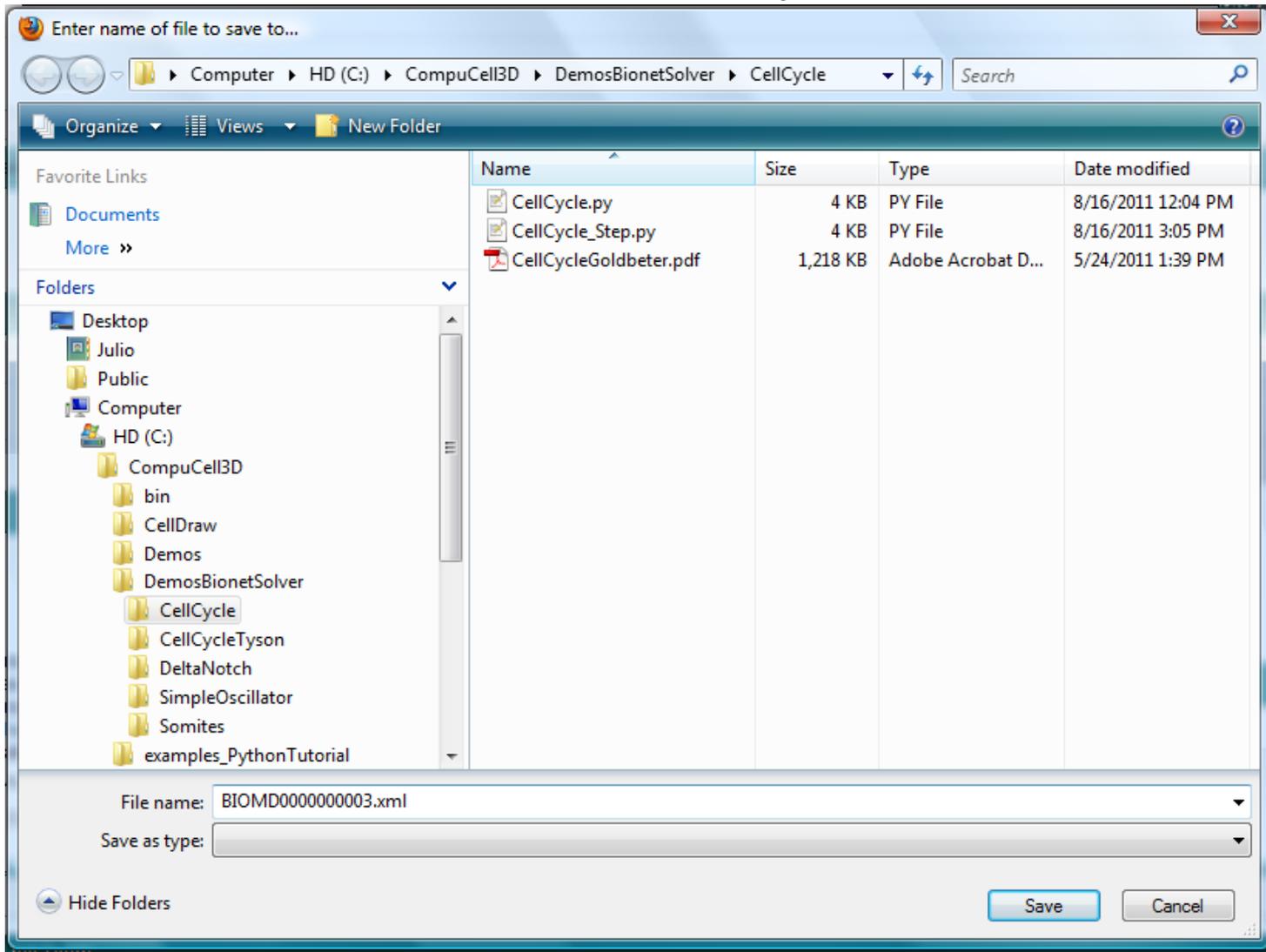
Goldbeter A *Proc. Natl. Acad. Sci. U.S.A.* 1991;88(20):9107-11 [1833774](#).

Abstract:

A minimal model for the mitotic oscillator is presented. The model, built on recent experimental advances, is based on the cascade of post-translational modification tha

Third Example – Cell Cycle from web

- Save the file *BIOMD0000000003.xml* inside the directory *CompuCell3D\DemosBionetSolver\CellCycle*



Third Example – Cell Cycle from web

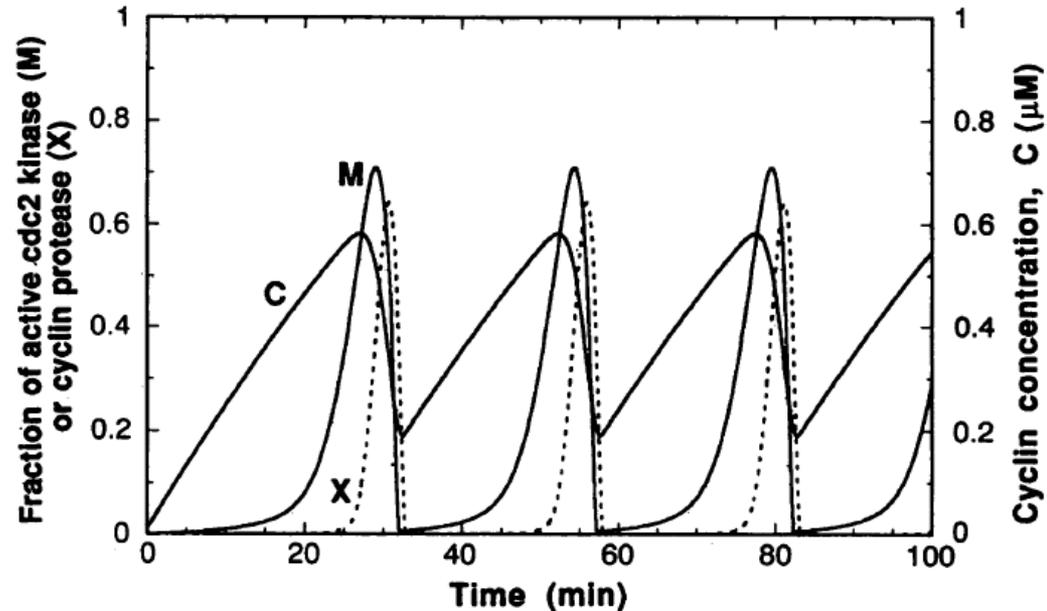
- This model is composed of 3 ODEs that forms an oscillating system:

$$\frac{dC}{dt} = v_i - v_d X \frac{C}{K_d + C} - k_d C,$$

$$\frac{dM}{dt} = V_1 \frac{(1 - M)}{K_1 + (1 - M)} - V_2 \frac{M}{K_2 + M},$$

$$\frac{dX}{dt} = V_3 \frac{(1 - X)}{K_3 + (1 - X)} - V_4 \frac{X}{K_4 + X}$$

$$V_1 = \frac{C}{K_c + C} V_{M1}, \quad V_3 = M V_{M3}.$$



- C : cyclin concentration
- M : fraction of active cdc2 kinase
- X : fraction of active cyclin protease

Third Example – Cell Cycle from web

- The model is implemented into CC3D in the same ways as in the first example, but this time we just have 1 cell type:

```
30 import bionetAPI
31 class Cycle(SteppableBasePy):
32     def __init__(self, _simulator, _frequency, _tV):
33         SteppableBasePy.__init__(self, _simulator, _frequency)
34         bionetAPI.initializeBionetworkManager(self.simulator)
35         self.tV=_tV
36
37     def start(self):
38         #Loading model
39         ModelName = "CellCycle"
40         ModelKey = "CC"
41         ModelPath = os.getcwd() + "/DemosBionetSolver/CellCycle/BIOMD0000000003.xml"
42         IntegrationStep = 0.05
43         bionetAPI.loadSBMLModel(ModelName, ModelPath, ModelKey, IntegrationStep)
44
45         #Initial conditions
46         bionetAPI.addSBMLModelToTemplateLibrary(ModelName, "TypeA")
47         bionetAPI.initializeBionetworks()
48         for cell in self.cellList:
49             cellDict=CompuCell.getPyAttrib(cell)
50             cellDict["M"]=bionetAPI.getBionetworkValue("CC_M",cell.id)
51             cellDict["C"]=bionetAPI.getBionetworkValue("CC_C",cell.id)
```

Third Example – Cell Cycle from web

- According to the original paper, mitosis happens after the fraction of active cdc2 kinase (M) reaches its maximum at around 0.7.
- To implement this we store the value of M at each MCS and the previous MCS:

```
52
53 def step(self,mcs):
54     for cell in self.cellList:
55         #Storing CycB and Cdh1 levels of each cell
56         cellDict=CompuCell.getPyAttrib(cell)
57         cellDict["M0"]=cellDict["M"]
58         cellDict["M"]=bionetAPI.getBionetworkValue("CC_M",cell.id)
59         cellDict["C"]=bionetAPI.getBionetworkValue("CC_C",cell.id)
60         #Iterating the ODEs
61         bionetAPI.timestepBionetworks()
```

Third Example – Cell Cycle from web

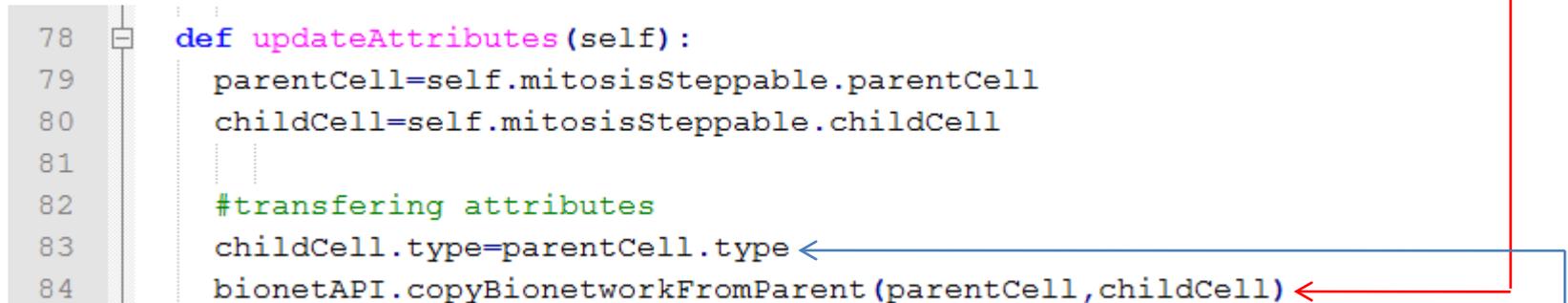
- Then, inside the Mitosis steppable, we check if the cell's internal fraction of M crosses the 0.7 threshold:

```
64 class MitosisSteppable(MitosisSteppableBase):
65     def __init__(self, _simulator, _frequency, _tV):
66         MitosisSteppableBase.__init__(self, _simulator, _frequency)
67         self.tV=_tV;
68
69     def step(self, mcs):
70         cells2divide=[]
71         for cell in self.cellList:
72             cellDict=CompuCell.getPyAttrib(cell)
73             if (cellDict["M"]>=0.7 and cellDict["M0"]<0.7):
74                 cells2divide.append(cell)
75         for cell in cells2divide:
76             self.divideCellRandomOrientation(cell)
```

Third Example – Cell Cycle from web

- Inside the updateAttributes function, we must copy the SBML model network from the parent to the child:

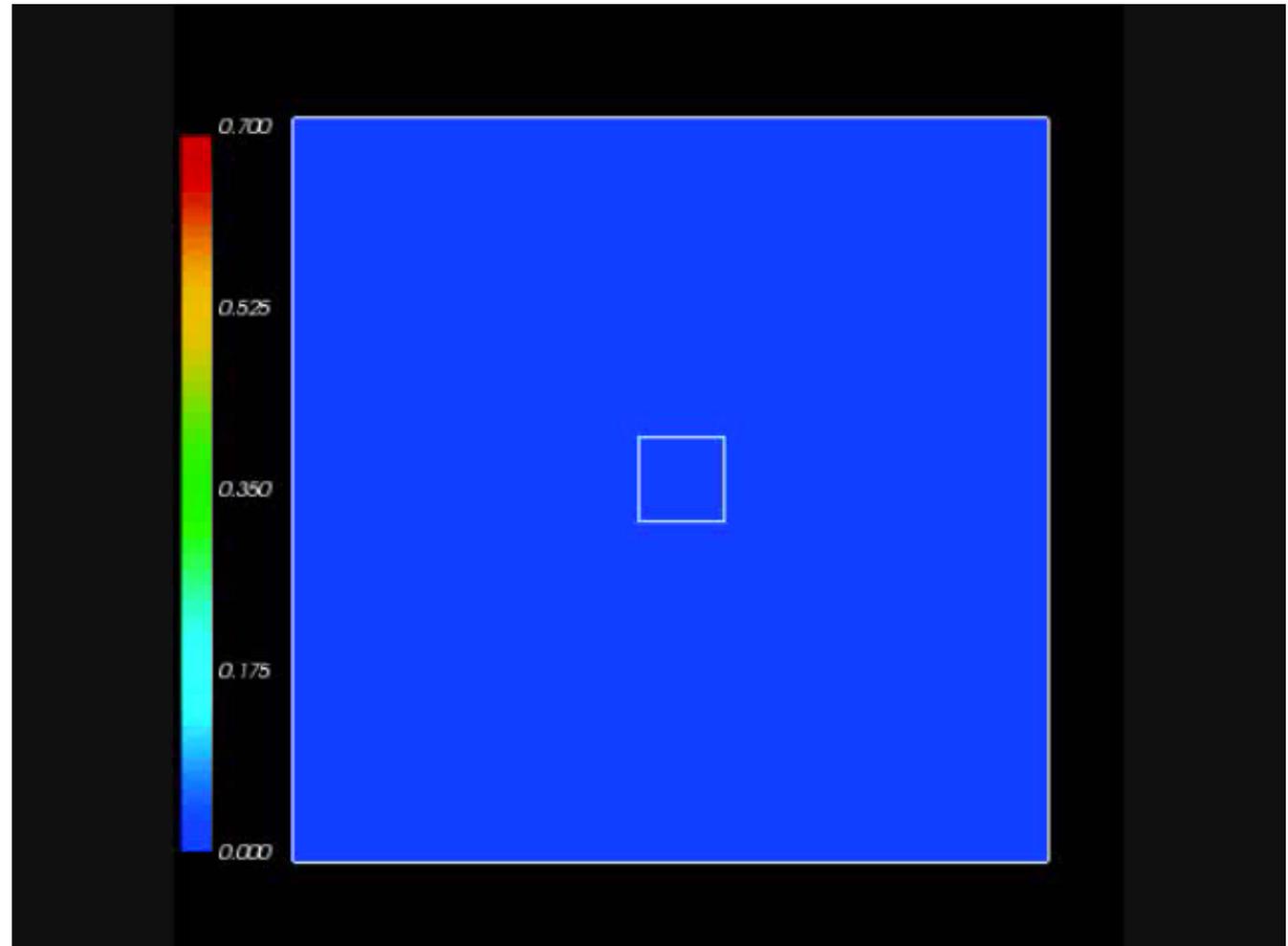
```
78 def updateAttributes(self):  
79     parentCell=self.mitosisSteppable.parentCell  
80     childCell=self.mitosisSteppable.childCell  
81     .....  
82     #transferring attributes  
83     childCell.type=parentCell.type  
84     bionetAPI.copyBionetworkFromParent (parentCell,childCell)
```



- But before doing this, we must assign a cell type to the child cell:

Third Example – Cell Cycle from web

- Next, open the model in CC3D, set the maximum concentration of the “Colormap Plot” to 0.75, and run the simulation:



Fourth Example – Tyson's Cell Cycle

- In the last example all cells divided in synchrony.
- The reason for this is the absence of any flow of information from the cell level to the subcellular level that would alter the state of the cell cycle oscillations.
- A more realistic model, where the cells do not maintain their cell cycle's phase, is the one proposed by Tyson and Novak.

Fourth Example – Tyson's Cell Cycle

- This model has 5 variables, from which only the first 2 forms the core of the cell cycle oscillations:

$$\frac{d[\text{CycB}]}{dt} = k_1 - (k'_2 + k''_2 [\text{Cdh1}]) [\text{CycB}],$$

$$\frac{d[\text{Cdh1}]}{dt} = \frac{(k'_3 + k''_3 A)(1 - [\text{Cdh1}])}{J_3 + 1 - [\text{Cdh1}]} - \frac{k_4 m [\text{CycB}] [\text{Cdh1}]}{J_4 + [\text{Cdh1}]},$$

$$\frac{d[\text{Cdc20}_T]}{dt} = k'_5 + k''_5 \frac{([\text{CycB}] m / J_5)^n}{1 + ([\text{CycB}] m / J_5)^n} - k_6 [\text{Cdc20}_T],$$

$$\frac{d[\text{Cdc20}_A]}{dt} = \frac{k_7 [\text{IEP}]([\text{Cdc20}_T] - [\text{Cdc20}_A])}{J_7 + [\text{Cdc20}_T] - [\text{Cdc20}_A]} - \frac{k_8 [\text{Mad}] \cdot [\text{Cdc20}_A]}{J_8 + [\text{Cdc20}_A]} - k_6 [\text{Cdc20}_A],$$

$$\frac{d[\text{IEP}]}{dt} = k_9 m [\text{CycB}] (1 - [\text{IEP}]) - k_{10} [\text{IEP}].$$

Fourth Example – Tyson’s Cell Cycle

- The crucial difference from the previous model lies in the presence of the parameter “m”, which is the normalized total mass of the cell:

$$\frac{d[\text{CycB}]}{dt} = k_1 - (k'_2 + k''_2 [\text{Cdh1}])[\text{CycB}],$$

$$\frac{d[\text{Cdh1}]}{dt} = \frac{(k'_3 + k''_3 A)(1 - [\text{Cdh1}])}{J_3 + 1 - [\text{Cdh1}]} - \frac{k_4 m [\text{CycB}][\text{Cdh1}]}{J_4 + [\text{Cdh1}]},$$

- This parameter varies between ~ 0.5 (right after mitosis) and ~ 1 (at normal size) and corresponds in CC3D to the ratio of volume to target volume:

$$V_\sigma / V_{\text{target}}$$

Fourth Example – Tyson’s Cell Cycle

- In the steppable file, *CellCycle_Tyson_Step.py*, this is implemented as:

```
55
56 def step(self,mcs):
57     for cell in self.cellList:
58         if (cell):
59             #Setting volume parameter for each cell
60             bionetAPI.setBionetworkValue("CC m",cell.volume/self.tV,cell.id)
61             #Storing CycB and Cdh1 levels of each cell
62             cellDict=CompuCell.getPyAttrib(cell)
63             cellDict["CycB0"]=cellDict["CycB"]
64             cellDict["CycB"]=bionetAPI.getBionetworkValue("CC_CycB",cell.id)
65             cellDict["Cdh1"]=bionetAPI.getBionetworkValue("CC_Cdh1",cell.id)
66             #Iterating the ODEs
67             bionetAPI.timestepBionetworks()
```

- Where “self.tV” is a variable which contains the original target volume of the cells and is passed into the steppable class from the main Python file.

Fourth Example – Tyson's Cell Cycle

- This passage of variables is done in the following way:
 - At the beginning of the *CellCycle_Tyson.py* file I declare and define global variables, from which *tV* – target volume – is one:

```
7
8  global Lx; global Ly; global cd; global T; global nOrder
9  global LamV; global tV
10
11  #PARAMETERS:
12  cd=7 ..... #typical cell diameter
13  Lx=5*cd ..... #Lattice size -- x
14  Ly=5*cd ..... #Lattice size -- y
15  T=10 ..... #Temperature
16  nOrder=4 ..... #Distance of interaction
17  #
18  #VOLUME/SURFACE PARAMETERS:
19  LamV=10 ..... #Lambda Volume
20  tV=cd*cd ..... #Target Volume
21
22
23  def configureSimulation(sim) :
24      import CompuCellSetup
25      from XMLUtils import ElementCC3D
26      cc3d=ElementCC3D("CompuCell3D")
27      potts=cc3d.ElementCC3D("Potts")
```

Fourth Example – Tyson’s Cell Cycle

- This passage of variables is done in the following way:
 - Later, when the steppable is called, the variable is passed as an argument:

```
82
83 from CellCycle_Tyson_Step import Cycle
84 cycle=Cycle(_simulator=sim, _frequency=1, _tV=tV)
85 steppableRegistry.registerSteppable(cycle)
86
```

- And in the *CellCycle_Tyson_Step.py* file, the argument is stored in the following way:

```
30 import bionetAPI
31 class Cycle(SteppableBasePy):
32     def __init__(self, _simulator, _frequency, _tV):
33         SteppableBasePy.__init__(self, _simulator, _frequency)
34         bionetAPI.initializeBionetworkManager(self.simulator)
35         self.tV=_tV
36
```

- The storage of the parameter as `self.tV` is necessary for it to be accessible to all functions inside the class.

Fourth Example – Tyson's Cell Cycle

- We went through all this trouble because the right way to implement cell growth in CC3D is by gradual increases in its target volume after mitosis:

```
23 def step(self,mcs):
24     for cell in self.cellList:
25         if (cell):
26             if (cell.volume<self.tV):
27                 cell.targetVolume+=0.1
```

- If we keep the target volume (V_t) constant after mitosis, instead of resetting it to the actual cell volume (V), the difference between V and V_t would be too great, leading to unrealistic cell dynamics.

```
85 def updateAttributes(self):
86     parentCell=self.mitosisSteppable.parentCell
87     childCell=self.mitosisSteppable.childCell
88
89     #transferring attributes
90     childCell.type=parentCell.type
91     bionetAPI.copyBionetworkFromParent (parentCell,childCell)
92     #volume
93     parentCell.targetVolume=parentCell.volume
94     childCell.targetVolume=childCell.volume
95     childCell.lambdaVolume=parentCell.lambdaVolume
```

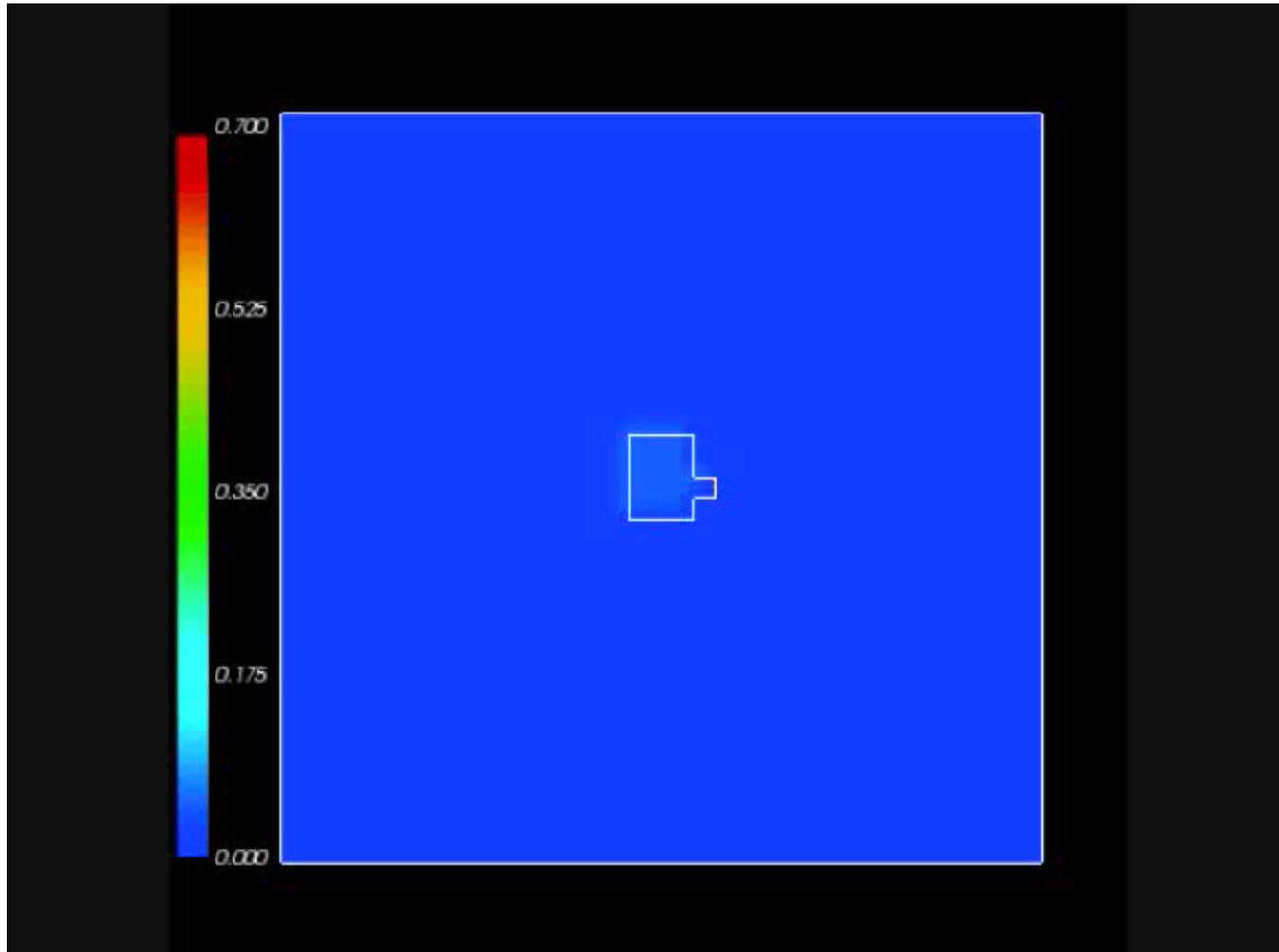
Fourth Example – Tyson's Cell Cycle

- Back to Tyson's cell cycle model.
- Here, the criteria for cell division is a low level of cycling B variable.
 - Once cycling B drops below a concentration of 0.1 grams of protein per gram of total cell mass, the cell undergoes mitosis

```
75 def step(self,mcs):
76     cells2divide=[]
77     for cell in self.cellList:
78         if cell:
79             cellDict=CompuCell.getPyAttrib(cell)
80             if (cellDict["CycB"]<=0.1 and cellDict["CycB0"]>0.1):
81                 cells2divide.append(cell)
82     for cell in cells2divide:
83         self.divideCellRandomOrientation(cell)
84
```

Fourth Example – Tyson's Cell Cycle

- When we run this model we can see that due to the fluctuations in cell volume, the divisions get out of sync:

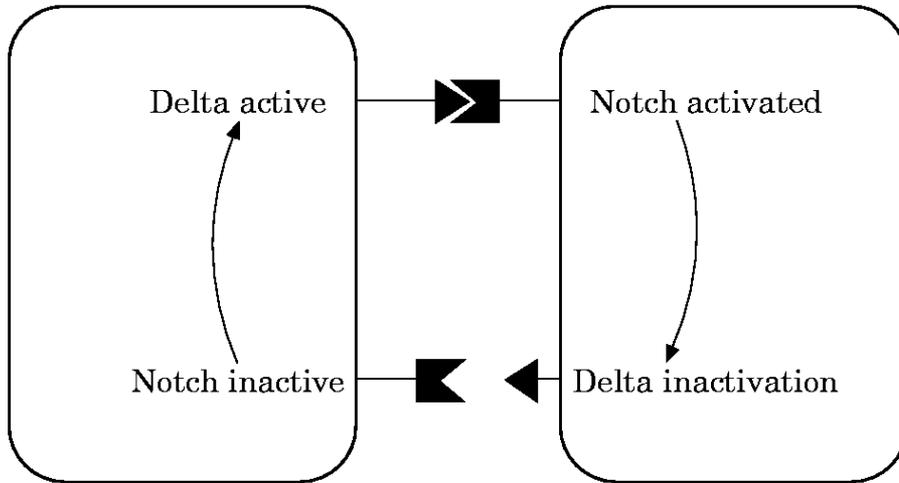


Fifth Example – Delta-Notch Patterning

- The fourth example (Tyson's cell cycle model) illustrated how changes at the single cell level can affect the subcellular level (and in turn affect the cell behavior by initiating mitosis).
- This last example will show how conditions external to the cell (the neighboring cells' Delta) can affect the cell internal state (its Notch levels).

Fifth Example – Delta-Notch Patterning

- We will use the model published by Collier *et al.* in 1996:



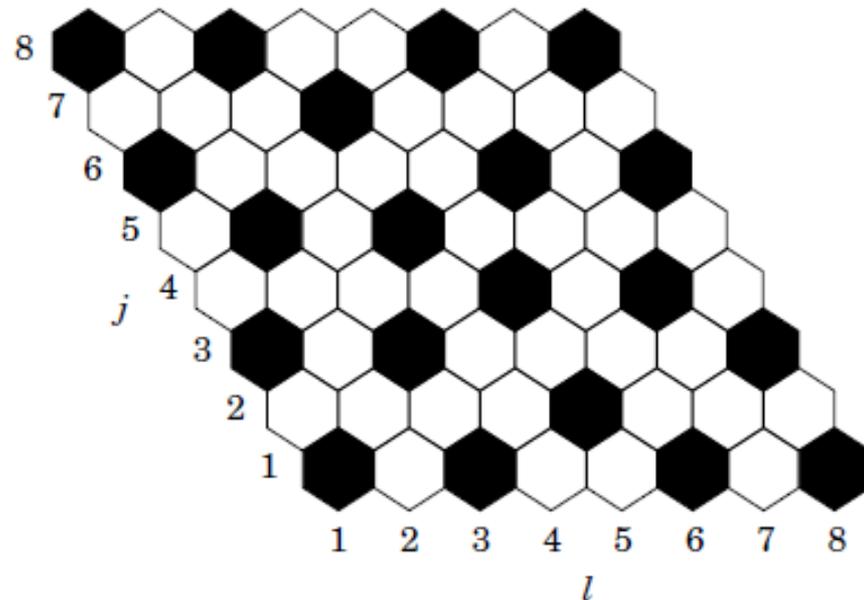
$$\frac{dD}{dt} = v \cdot \left(\frac{1}{1 + b \cdot N^h} - N \right)$$

$$\frac{dN}{dt} = \frac{\bar{D}^k}{a + \bar{D}^k} - N$$

- N : Notch
- D : Delta
- \bar{D} : average Delta from neighbors

Fifth Example – Delta-Notch Patterning

- In this model, when a cell receives high levels of Delta from neighbors its Notch level becomes downregulated.
- This leads to the high/low Notch patterning shown by their simulations on an hexagonal lattice:



Fifth Example – Delta-Notch Patterning

- In CC3D we first loop over all cells' neighbors and store their Delta:

```
37 def step(self,mcs):
38     for cell in self.cellList:
39         if (cell):
40             D=0.0; nn=0 ←
41             cellNeighborList=self.getCellNeighbors(cell)
42             for neighbor in cellNeighborList:
43                 if (neighbor.neighborAddress):
44                     nn+=1 ←
45                     D+=bionetAPI.getBionetworkValue("DN_D",neighbor.neighborAddress.id) ←
46             if (nn>0):
47                 D=D/nn ←
48                 bionetAPI.setBionetworkValue("DN_Davg",D,cell.id) ←
49                 cellDict=CompuCell.getPyAttrib(cell)
50                 cellDict["D"]=D
51                 cellDict["N"]=bionetAPI.getBionetworkValue("DN_N",cell.id)
52             bionetAPI.timestepBionetworks()
```

- Then we average it and use it as the new \bar{D} parameter of that cell:

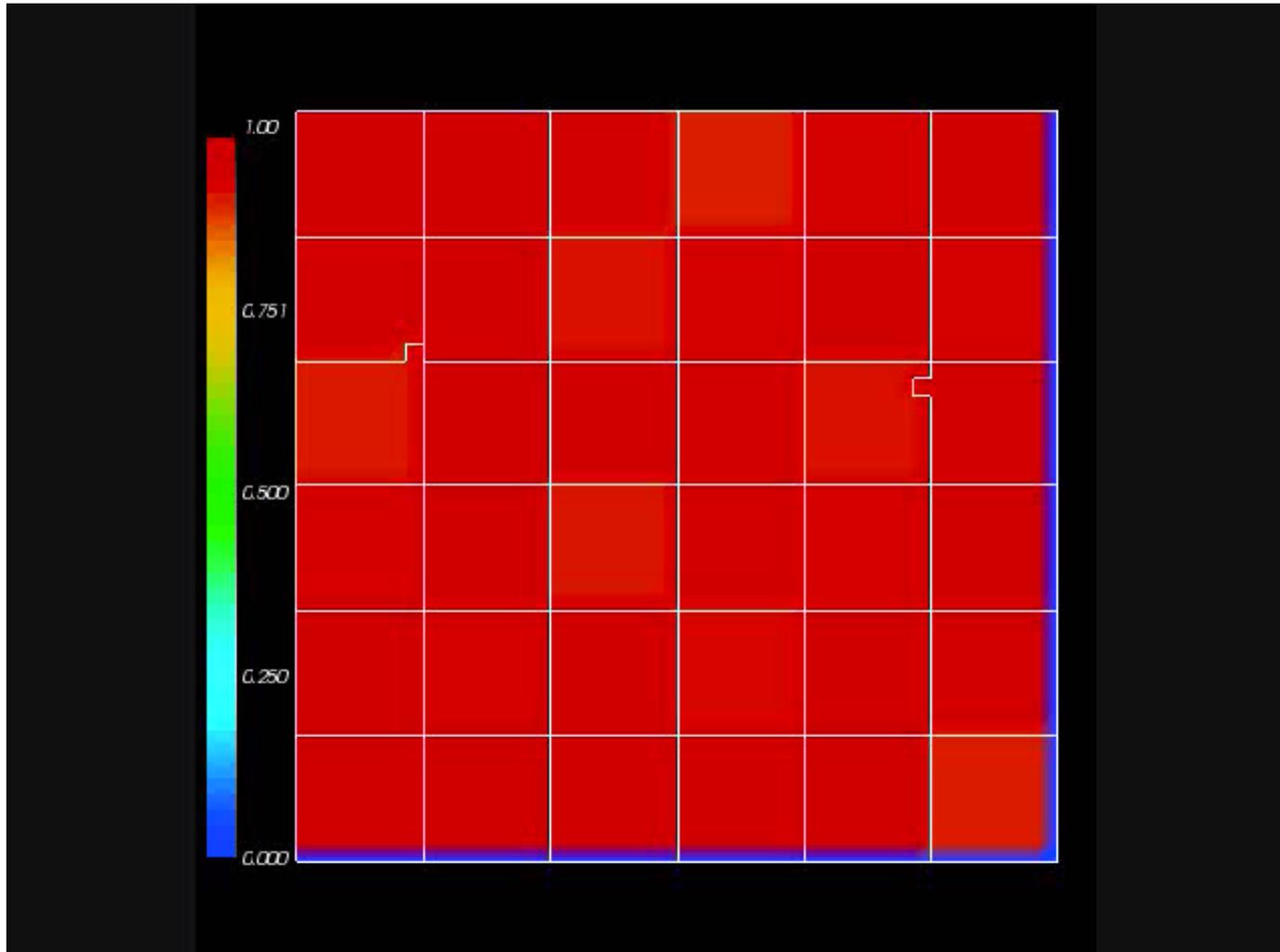
Fifth Example – Delta-Notch Patterning

- As an initial condition all cells start with random values of Delta and Notch around 0.9.
- To implement this we use the Python random function as shown below:

```
25     #Initial conditions
26     import random
27     for cell in self.cellList:
28         if (cell):
29             D = random.uniform(0.9,1.0)
30             N = random.uniform(0.9,1.0)
31             bionetAPI.setBionetworkValue("DN_D",D,cell.id)
32             bionetAPI.setBionetworkValue("DN_N",N,cell.id)
33             cellDict=CompuCell.getPyAttrib(cell)
34             cellDict["D"]=D
35             cellDict["N"]=N
```

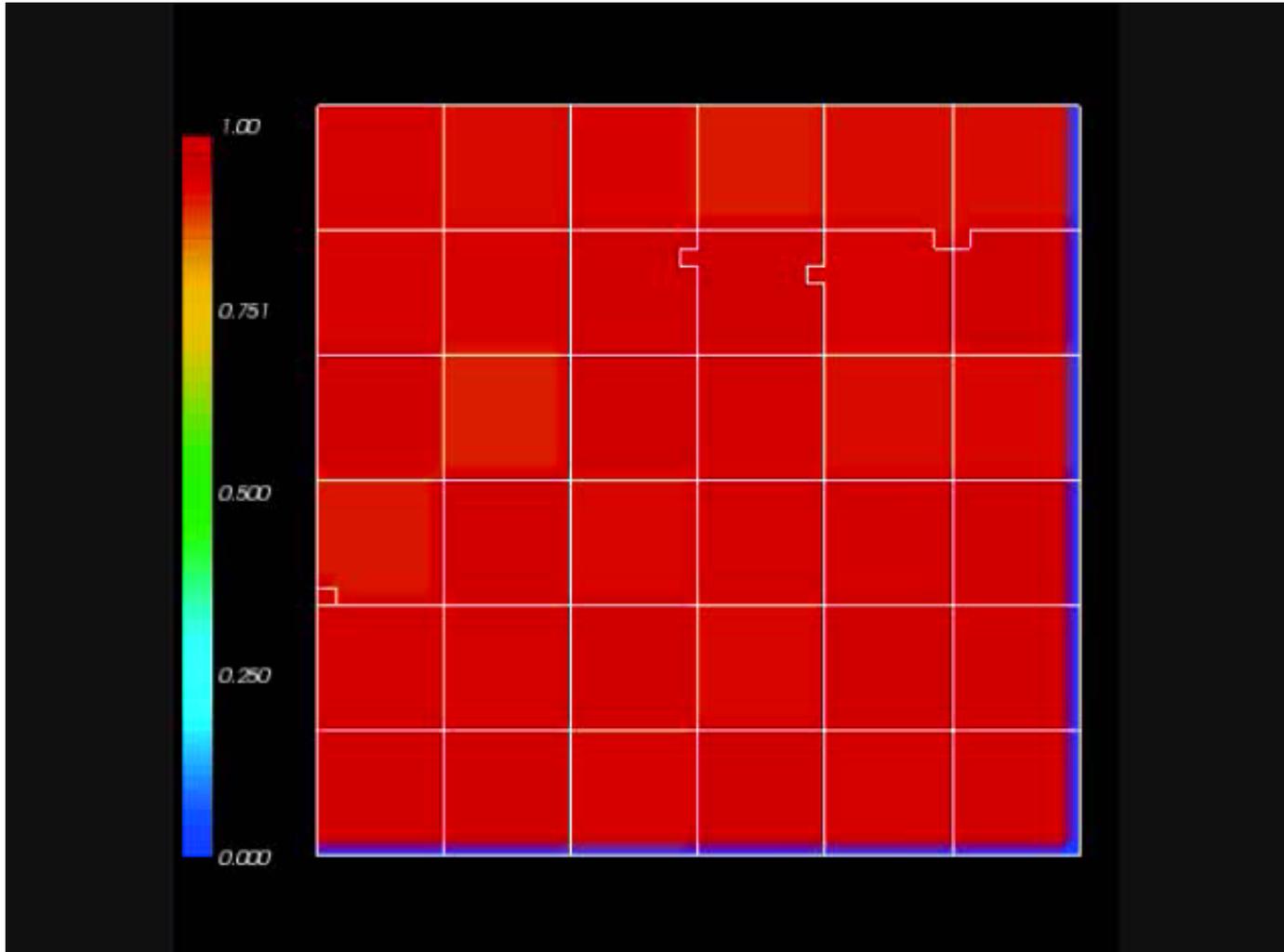
Fifth Example – Delta-Notch Patterning

- When we run this model we can see that first the Notch values go down before the pattern emerges:



Fifth Example – Delta-Notch Patterning

- If we increase the level of membrane fluctuations the pattern will be disrupted :



Sixth Example – 2 ODE models

- Below is a simulation with Tyson's Cell Cycle and Collier's Delta Notch models:

