Merging CompuCell3D and SBW/SBML

Julio M. Belmonte

Indiana University, Bloomington

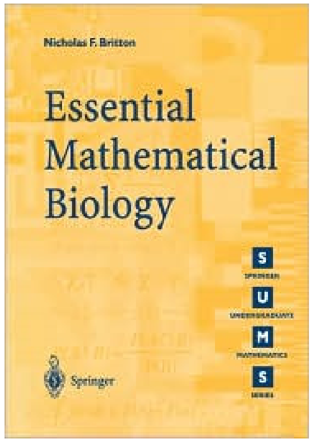
Outline

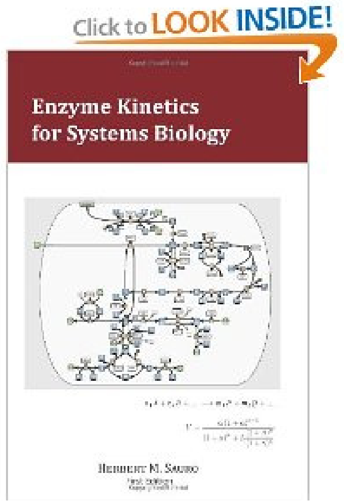
- Objectives
- Ways to add RK to CC3D
- SBML format
- Generating SBML using Jarnac
    - Simple Oscillator
- Integrating with CC3D
    - Bionet example – simple oscillator
    - Adding Cell Cycle model from *sbml.org*
    - John Tyson's Cell Cycle model
    - Collier *et al.* Delta-Notch patterning model

More on Reaction Kinetics Modeling

Essential Mathematical Biology

Nicholas Britton





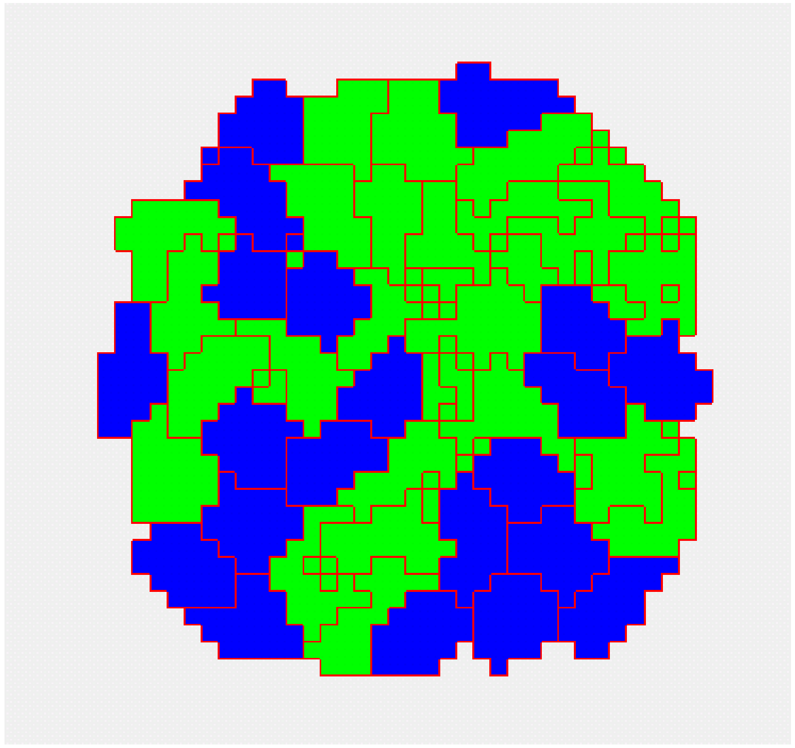
Enzyme Kinetics for Systems Biology

Herbert Sauro

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

Cell-based modeling

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

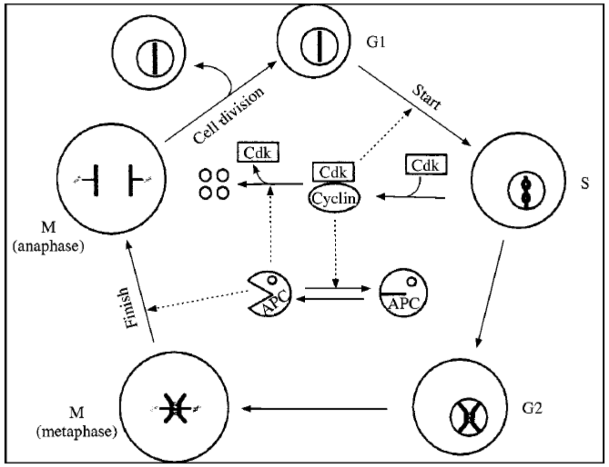


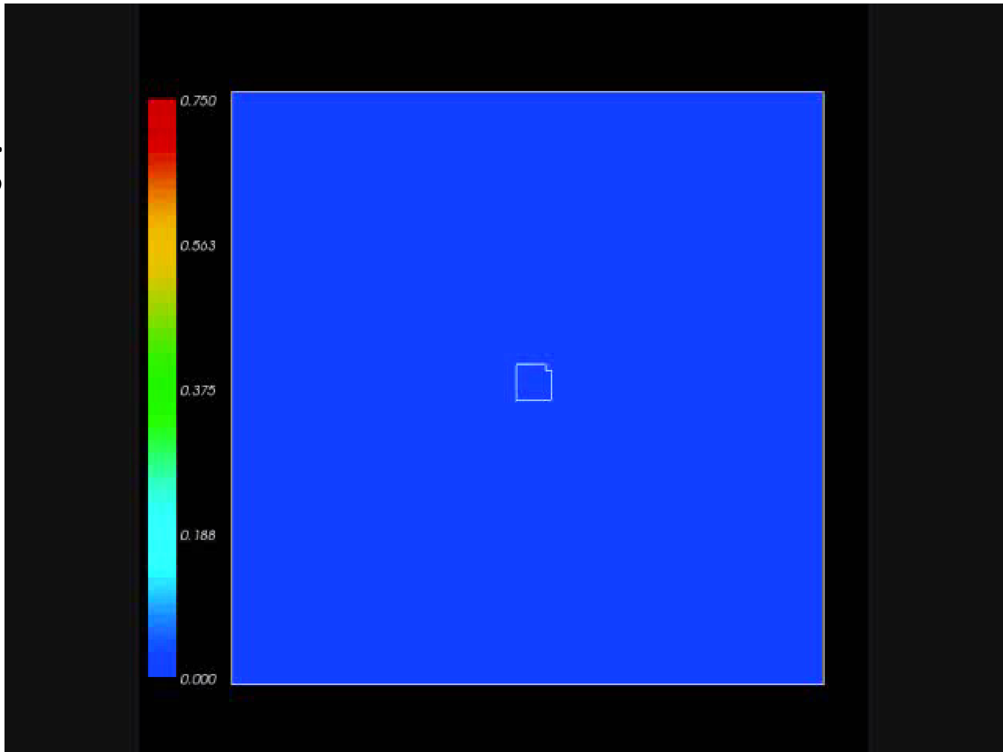
Subcellular modelling

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

Subcellular modelling

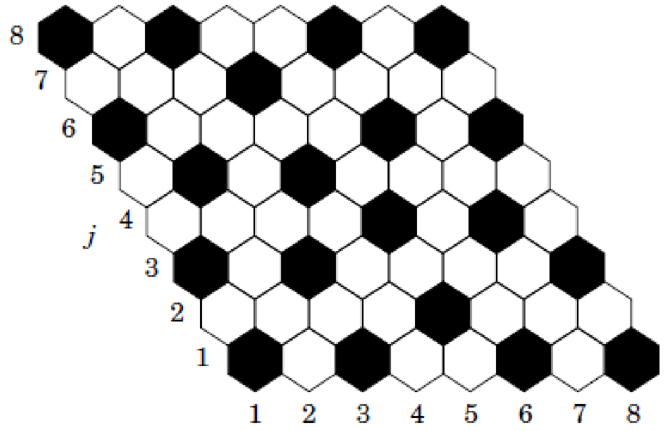
- Biochemical Kinetics:
  - Cell-Cycle
  - Circadian rhythms
  - Cardiac rhythms
  - cAMP oscillations
  - Delta-Notch patterning
  - WNT pathway
  - FGF pathway
  - Etc…

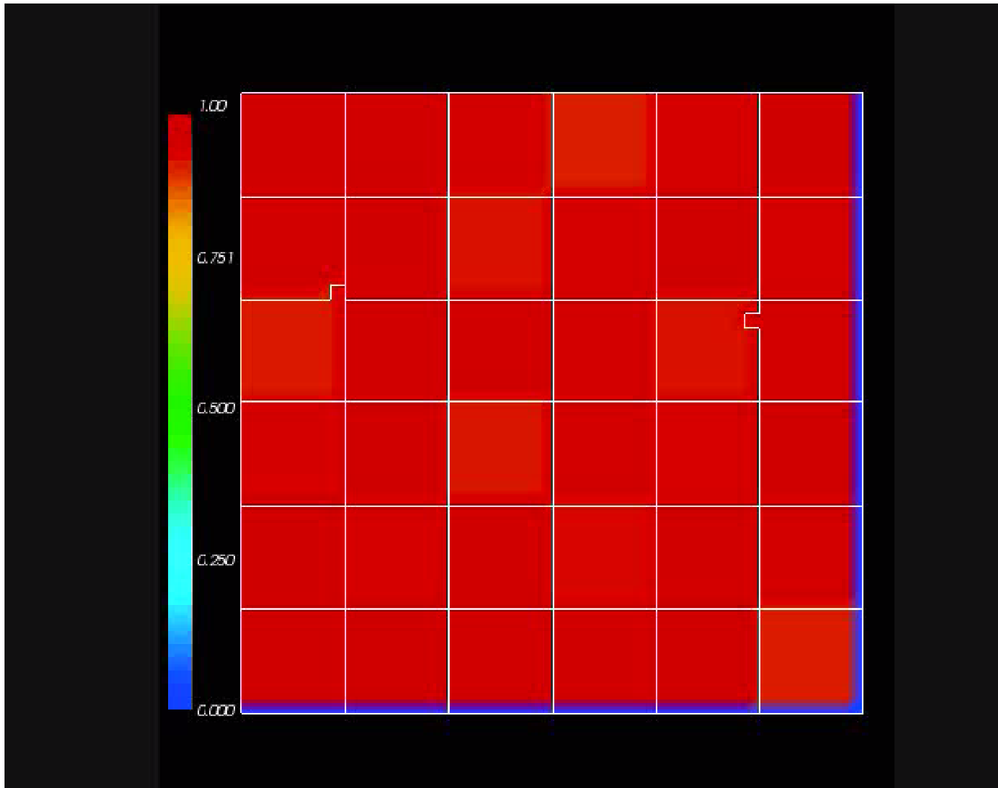




Subcellular modelling

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





Multiscale model - Somitogenesis

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?

$$\begin{tikzcd}
\text{Developer software (SBW/Jarnac)} \arrow[d] \\
\text{SBML} \arrow[d] \\
\text{Simulation software (CompuCell3D)}
\end{tikzcd}$$

SBML

$$S_1 \xrightarrow{k} 2 \cdot S_2$$

- 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>
```

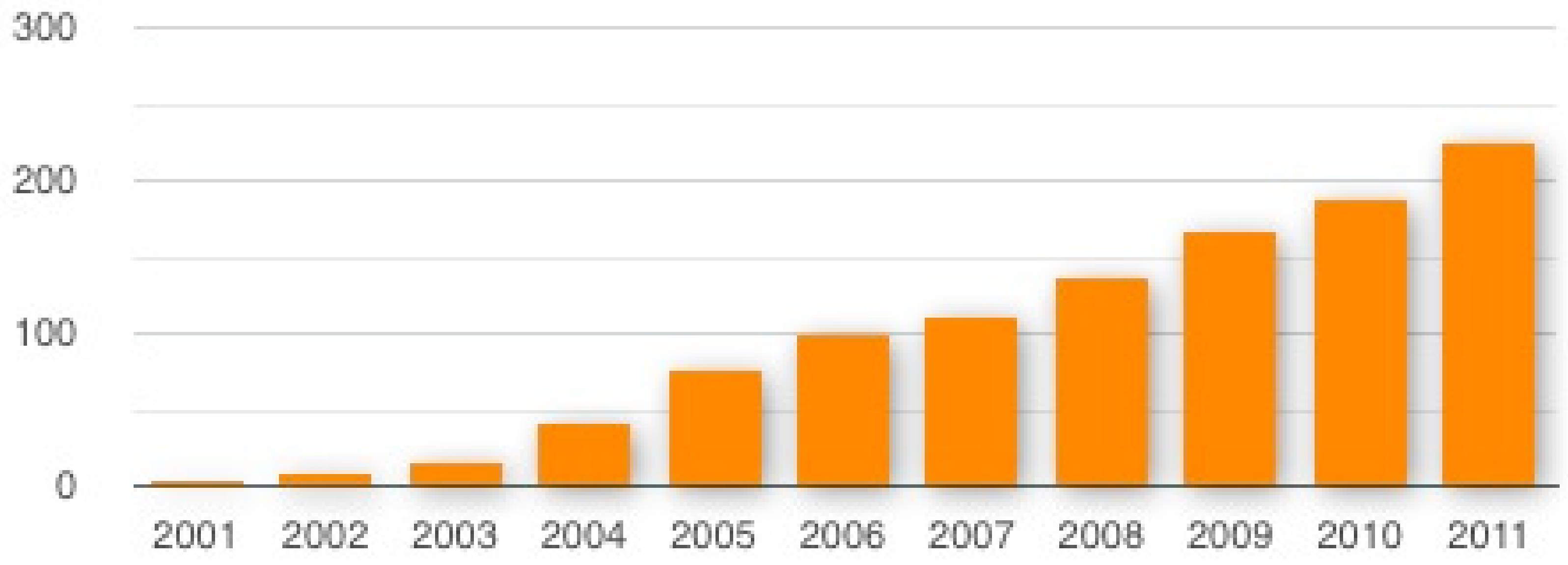
$$S_1 = 5 \;\text{nM}$$
$$S_2 = 0 \;\text{nM}$$

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

$$S_1 \xrightarrow{k} 2 \cdot S_2$$

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

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 CC3D 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>})
```

Integration with CC3D

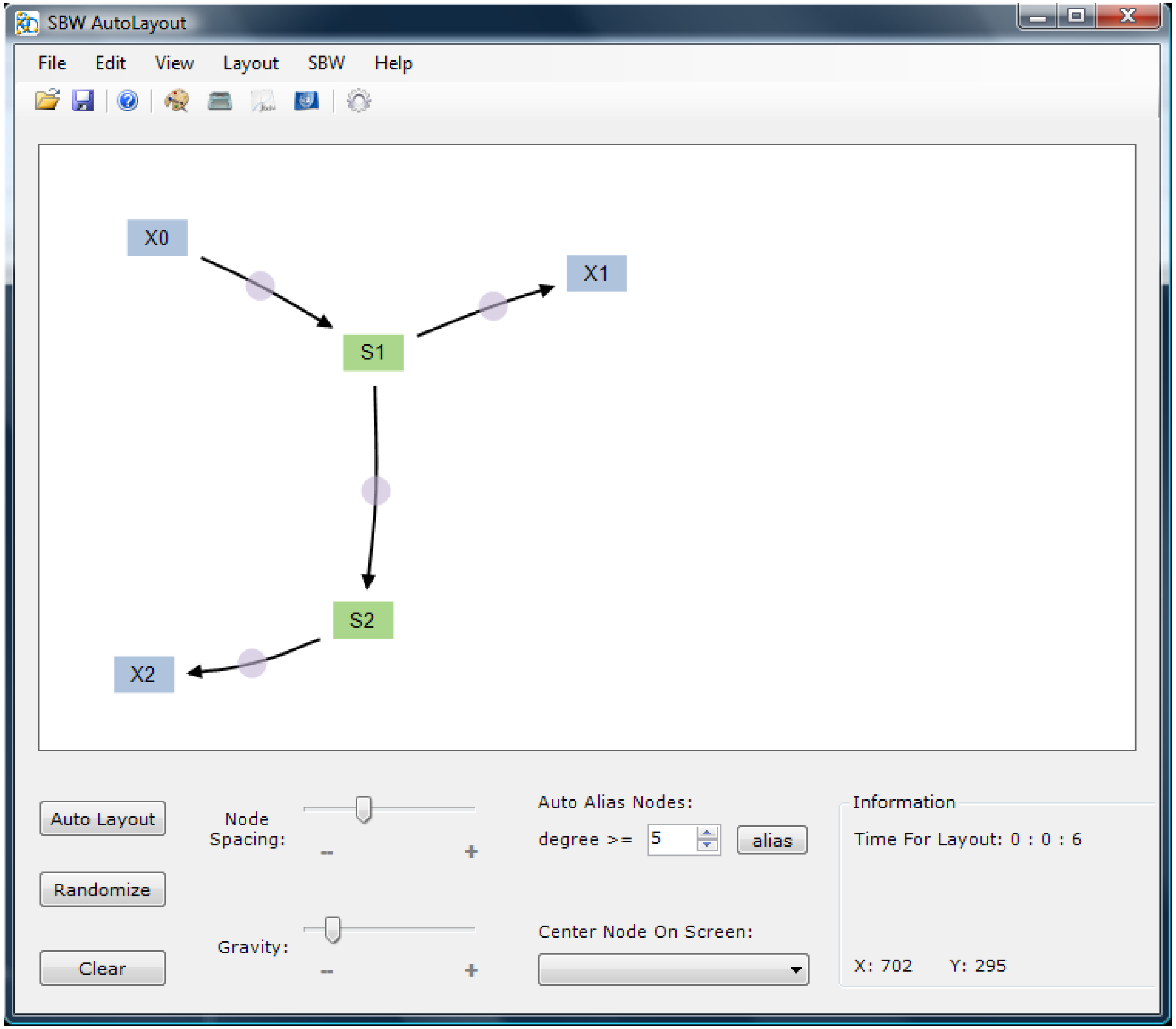
```
1    import bionetAPI # Import bionetAPI functions
     class <someClass>(SteppableBasePy):
       def __init__(self,_simulator,_frequency=1):
         SteppableBasePy.__init__(self,_simulator,_frequency)
2 →      bionetAPI.initializeBionetworkManager(self.simulator) # Initialize bionet inside class

       def start(self):
         # Load a specific subcellular SBML submodel
         ModelName = <sbmlModelName>      # Name of the model
         ModelPath = <sbmlModelPath>      # Path where the model is stored
3 →      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
4 →      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
5 →      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

- MODEL:
  - 2 cell types:
    - Condensing
    - NonCondensing
  - Condensing cells have stable volume
  - NonCondensing cells’ volume oscillate
  - Volume oscillation is driven by a subcellular model:
    - Oscli.sbml

First Example

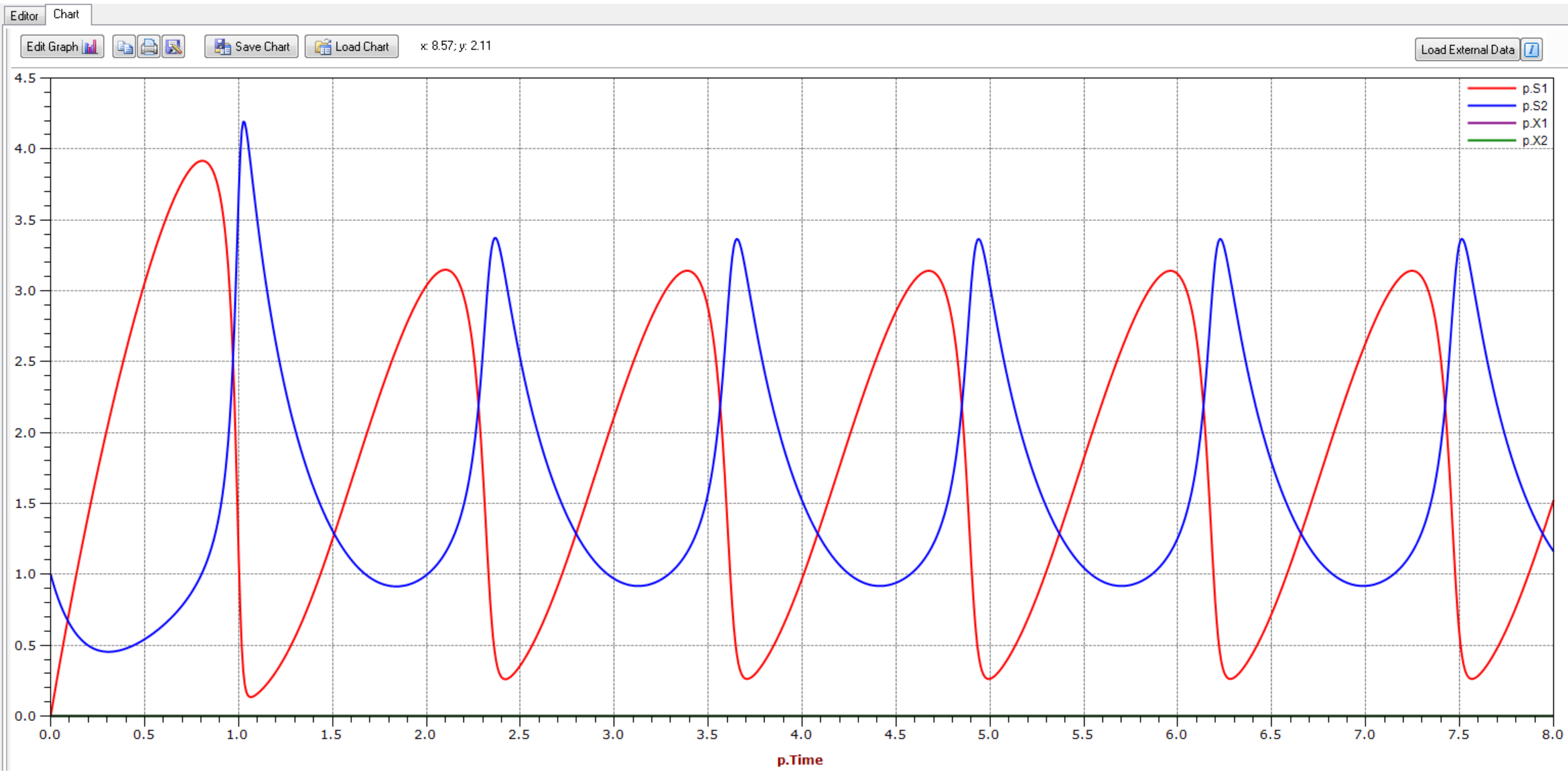
- *Oscli.sbml*:



Auto Alias Nodes:
degree >= 5
Center Node On Screen:
Time For Layout: 0 : 0 : 6
X: 702 Y: 295

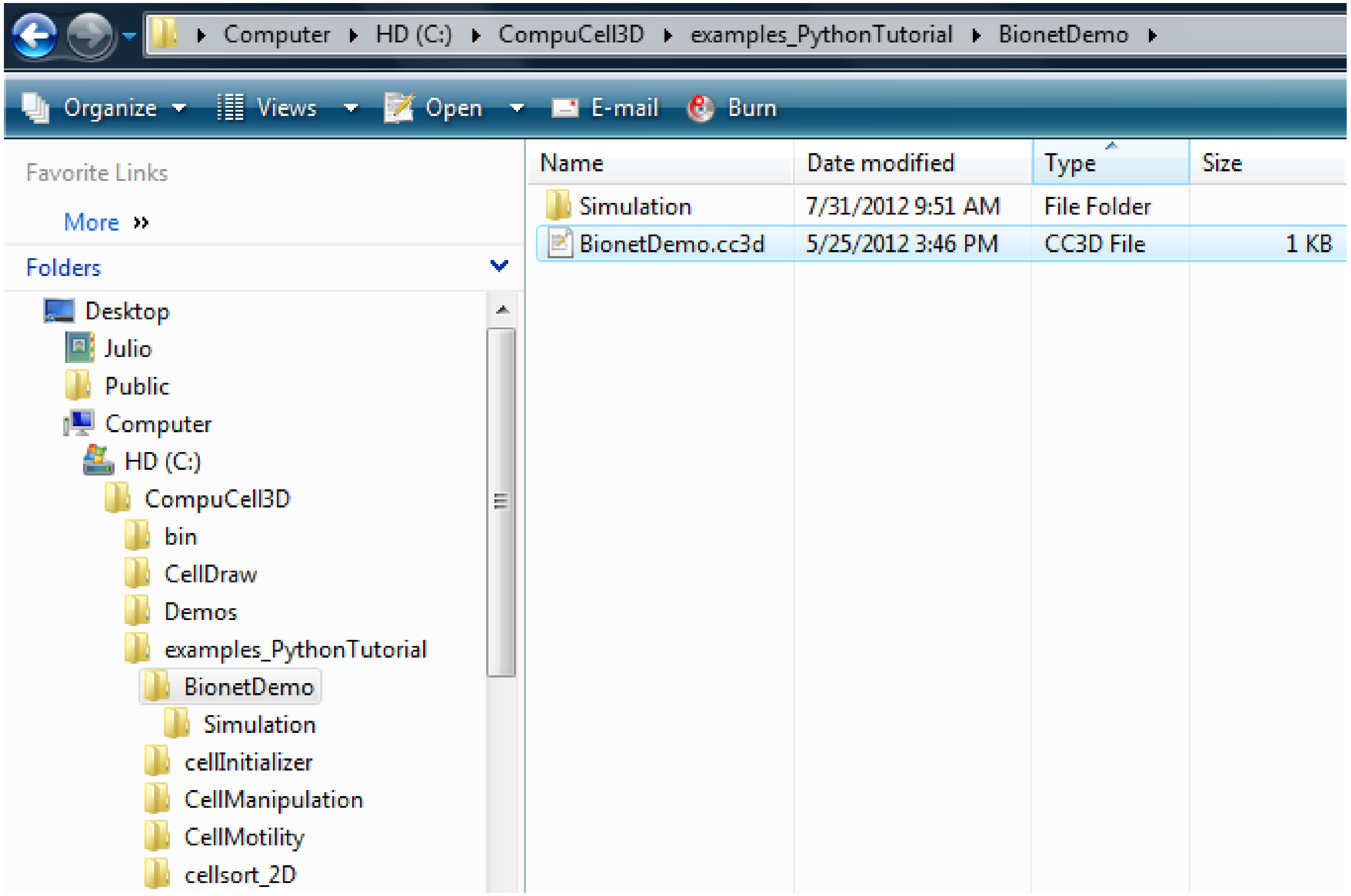
First Example

- Oscli.sbml:



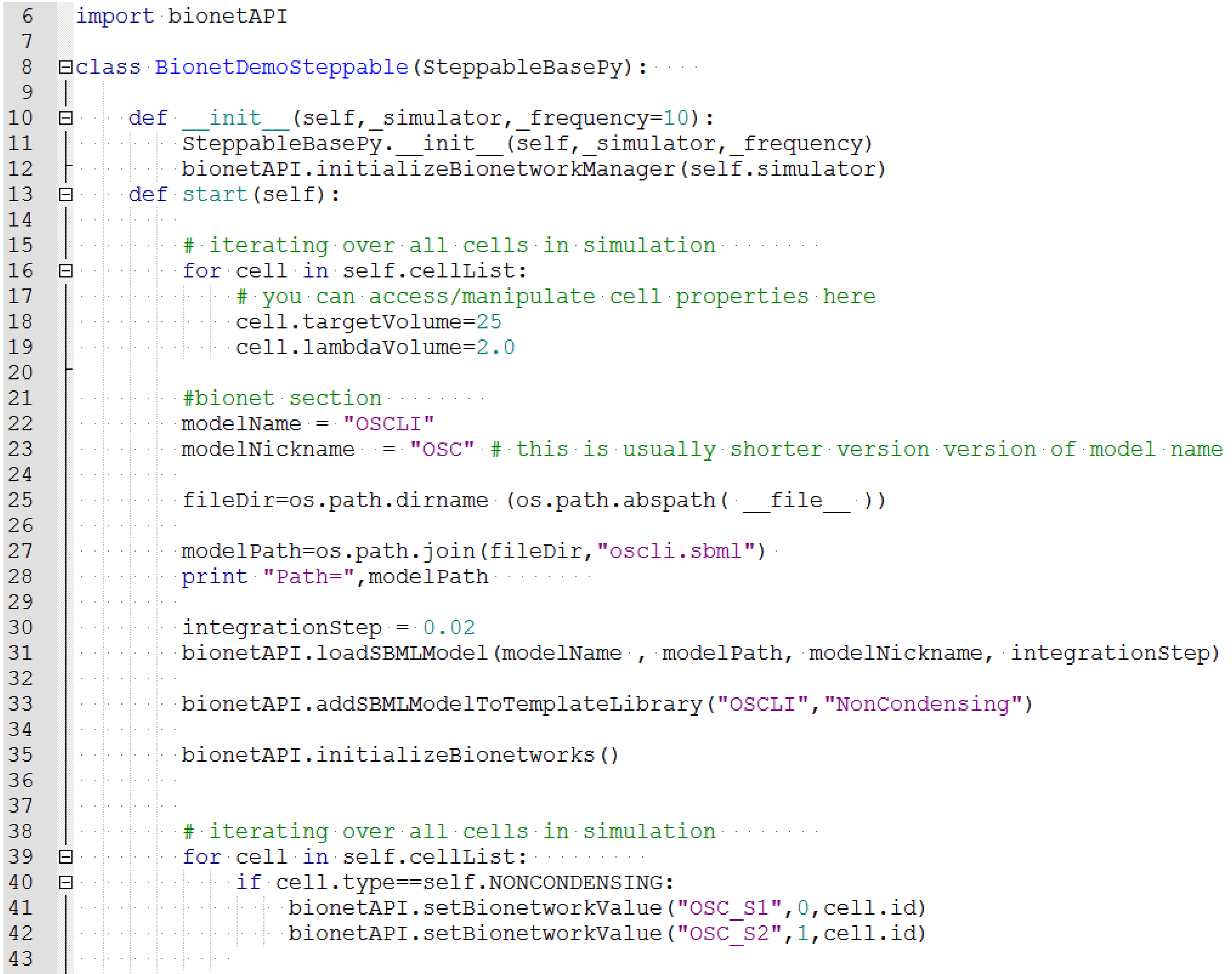
First Example

- On Twedit++, open the Project File BionetDemo:



Name	Date modified	Type	Size
Simulation	7/31/2012 9:51 AM	File Folder	
BionetDemo.cc3d	5/25/2012 3:46 PM	CC3D File	1 KB

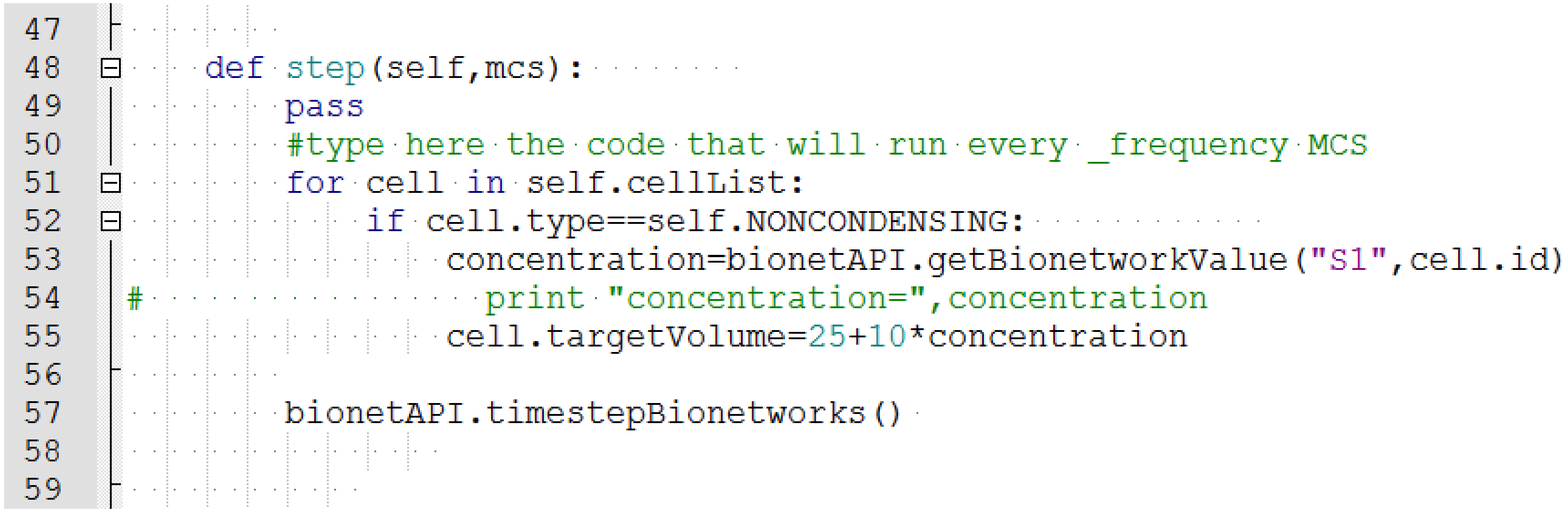
First Example



```python
import bionetAPI

class BionetDemoSteppable(SteppableBasePy):

    def __init__(self,_simulator,_frequency=10):
        SteppableBasePy.__init__(self,_simulator,_frequency)
        bionetAPI.initializeBionetworkManager(self.simulator)
    def start(self):

        # iterating over all cells in simulation
        for cell in self.cellList:
            # you can access/manipulate cell properties here
            cell.targetVolume=25
            cell.lambdaVolume=2.0

        #bionet section
        modelName = "OSCLI"
        modelNickname  = "OSC" # this is usually shorter version version of model name

        fileDir=os.path.dirname (os.path.abspath( __file__ ))

        modelPath=os.path.join(fileDir,"oscli.sbml")
        print "Path=",modelPath

        integrationStep = 0.02
        bionetAPI.loadSBMLModel(modelName , modelPath, modelNickname, integrationStep)

        bionetAPI.addSBMLModelToTemplateLibrary("OSCLI","NonCondensing")

        bionetAPI.initializeBionetworks()


        # iterating over all cells in simulation
        for cell in self.cellList:
            if cell.type==self.NONCONDENSING:
                bionetAPI.setBionetworkValue("OSC_S1",0,cell.id)
                bionetAPI.setBionetworkValue("OSC_S2",1,cell.id)

```

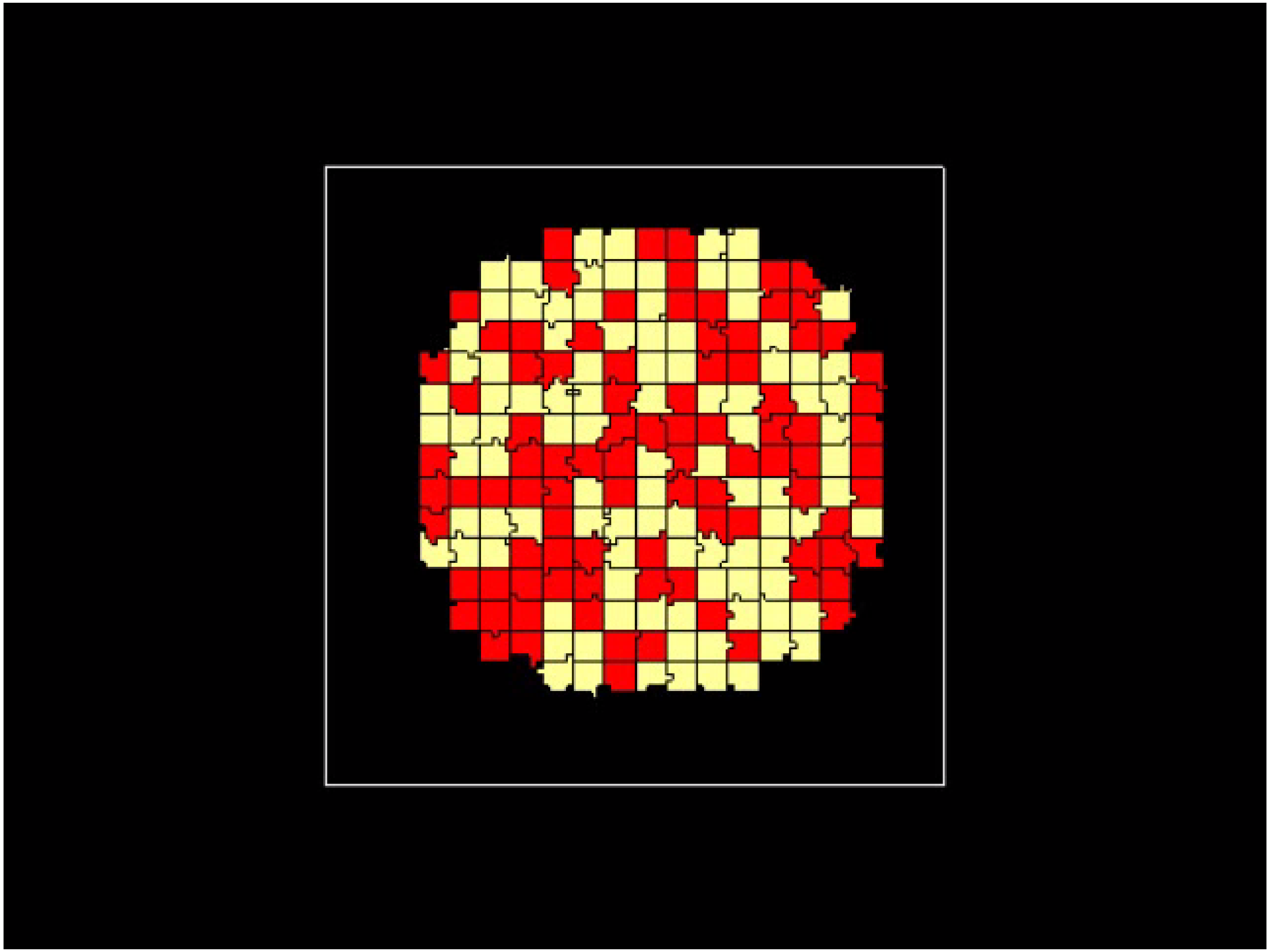
First Example

- This is how we read concentration values:



```
    def step(self,mcs):
        pass
        #type here the code that will run every _frequency MCS
        for cell in self.cellList:
            if cell.type==self.NONCONDENSING:
                concentration=bionetAPI.getBionetworkValue("S1",cell.id)
#                 print "concentration=",concentration
                cell.targetVolume=25+10*concentration

        bionetAPI.timestepBionetworks()

```

First Example



Exercises

- 1
  - Change volume oscillation amplitude
  - Make Condensing cells oscillate
  - Make Condensing cells oscillate at opposite phase

Exercises

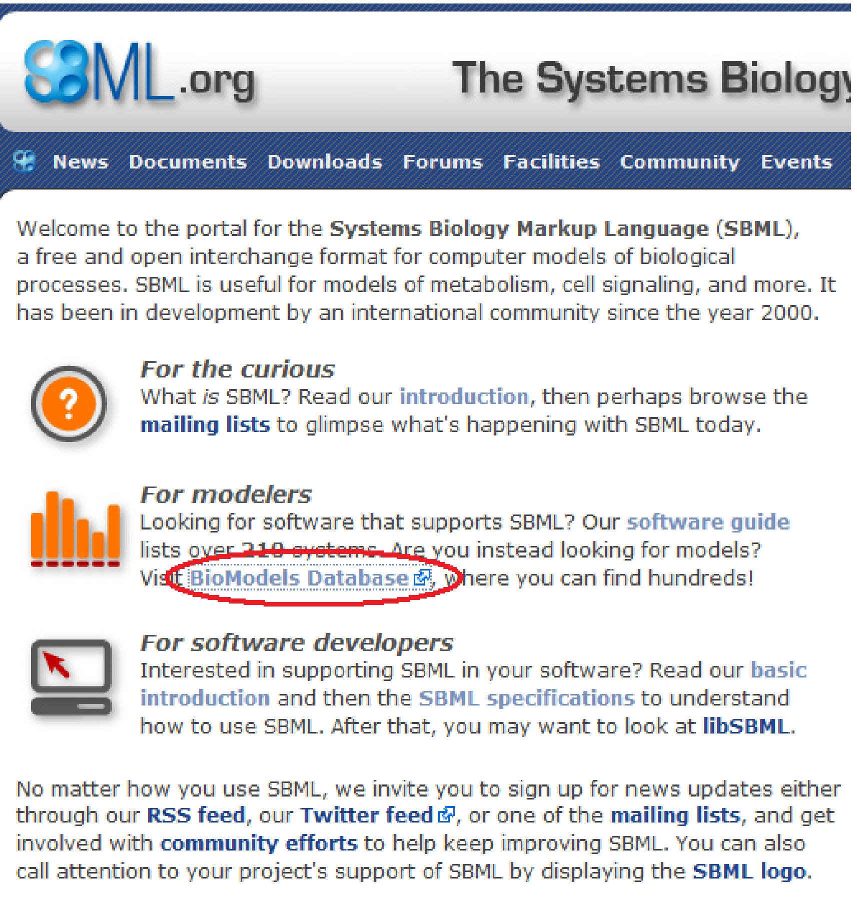
- 1
  - Change volume oscillation amplitude
  - Make Condensing cells oscillate
  - Make Condensing cells oscillate at opposite phase
- 2
  - Replace SBML model with the one from Tuesday: Boris Kholodenko, Eur J Biochem. 2000 Mar;267(6):1583-8

Second Example – Cell Cycle from web

- In our second example we will use a published model for the cell cycle.
- 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

Second Example – Cell Cycle Model

- On www.sbml.org, click on the link "*BioModels Database*" and then on "*Curated models*":


SBML.org The Systems Biology

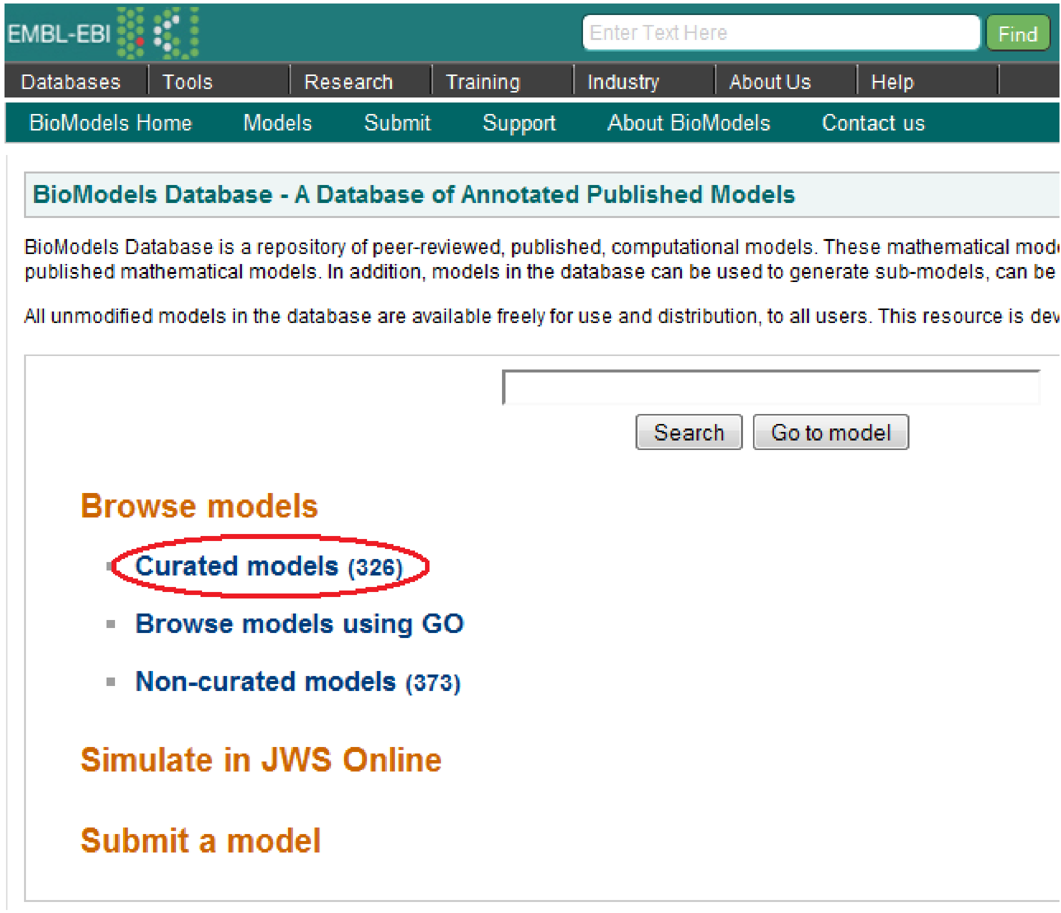
Welcome to the portal for the **Systems Biology Markup Language (SBML)**, a free and open interchange format for computer models of biological processes. SBML is useful for models of metabolism, cell signaling, and more. It has been in development by an international community since the year 2000.

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**.


EMBL-EBI
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 mod
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

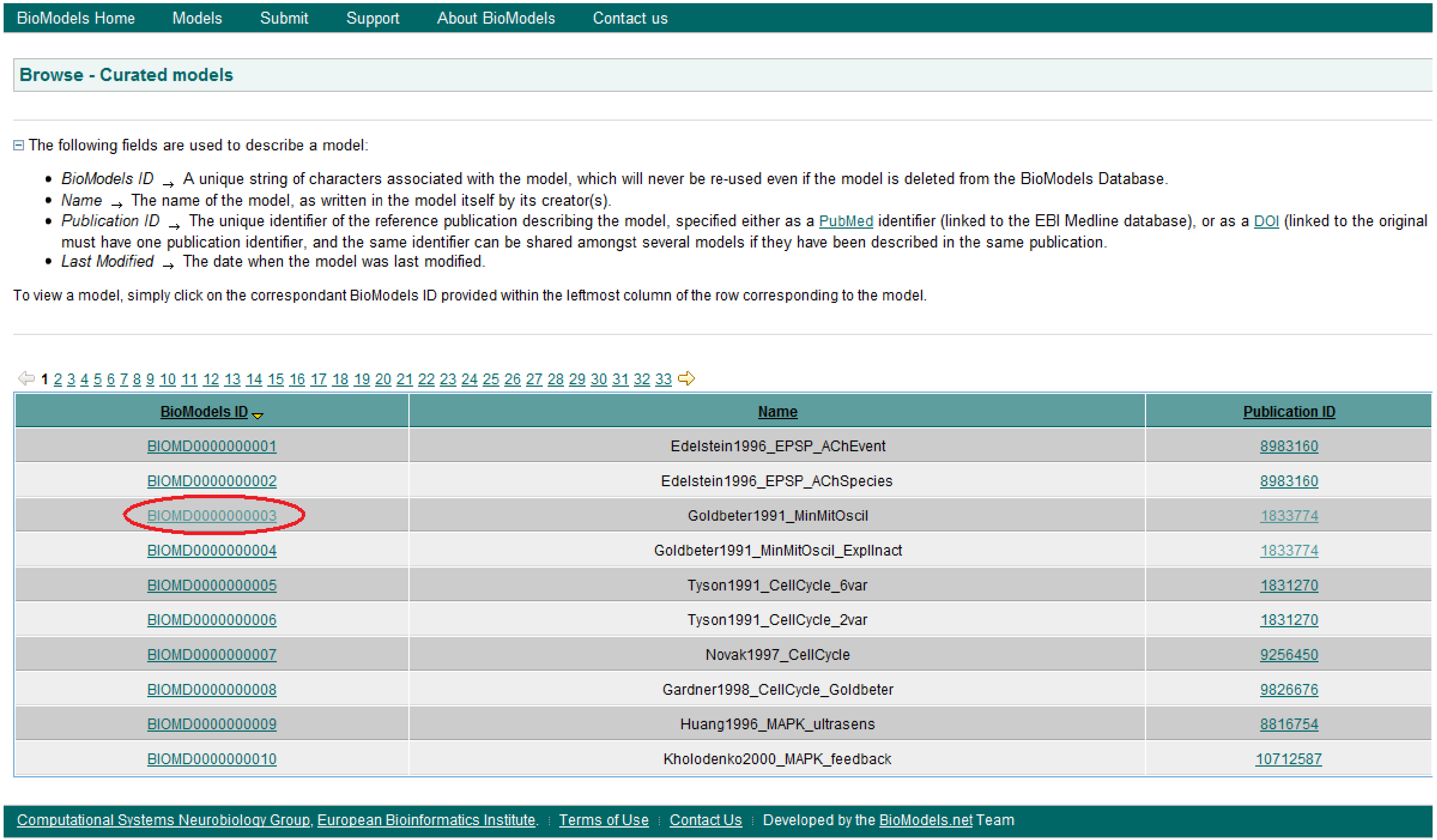
Browse models
- **Curated models** (326)
- **Browse models using GO**
- **Non-curated models** (373)

Simulate in JWS Online

Submit a model

Second Example – Cell Cycle from web

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



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 correspondant BioModels ID provided within the leftmost column of the row corresponding to the model.

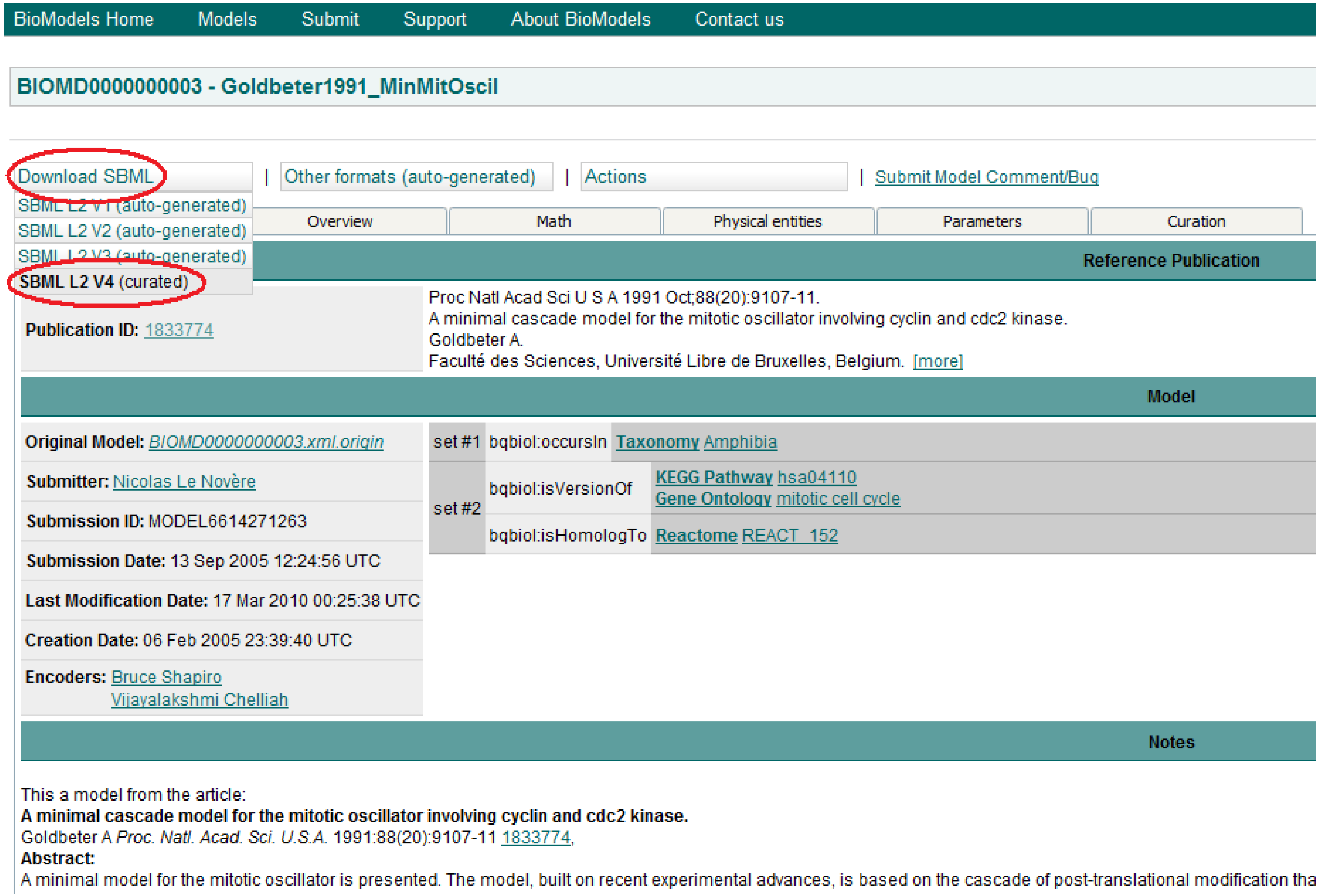
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33

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

Computational Systems Neurobiology Group, European Bioinformatics Institute. | Terms of Use | Contact Us | Developed by the BioModels.net Team

Second Example – Cell Cycle from web

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



BIOMD0000000003 - Goldbeter1991_MinMitOscil

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

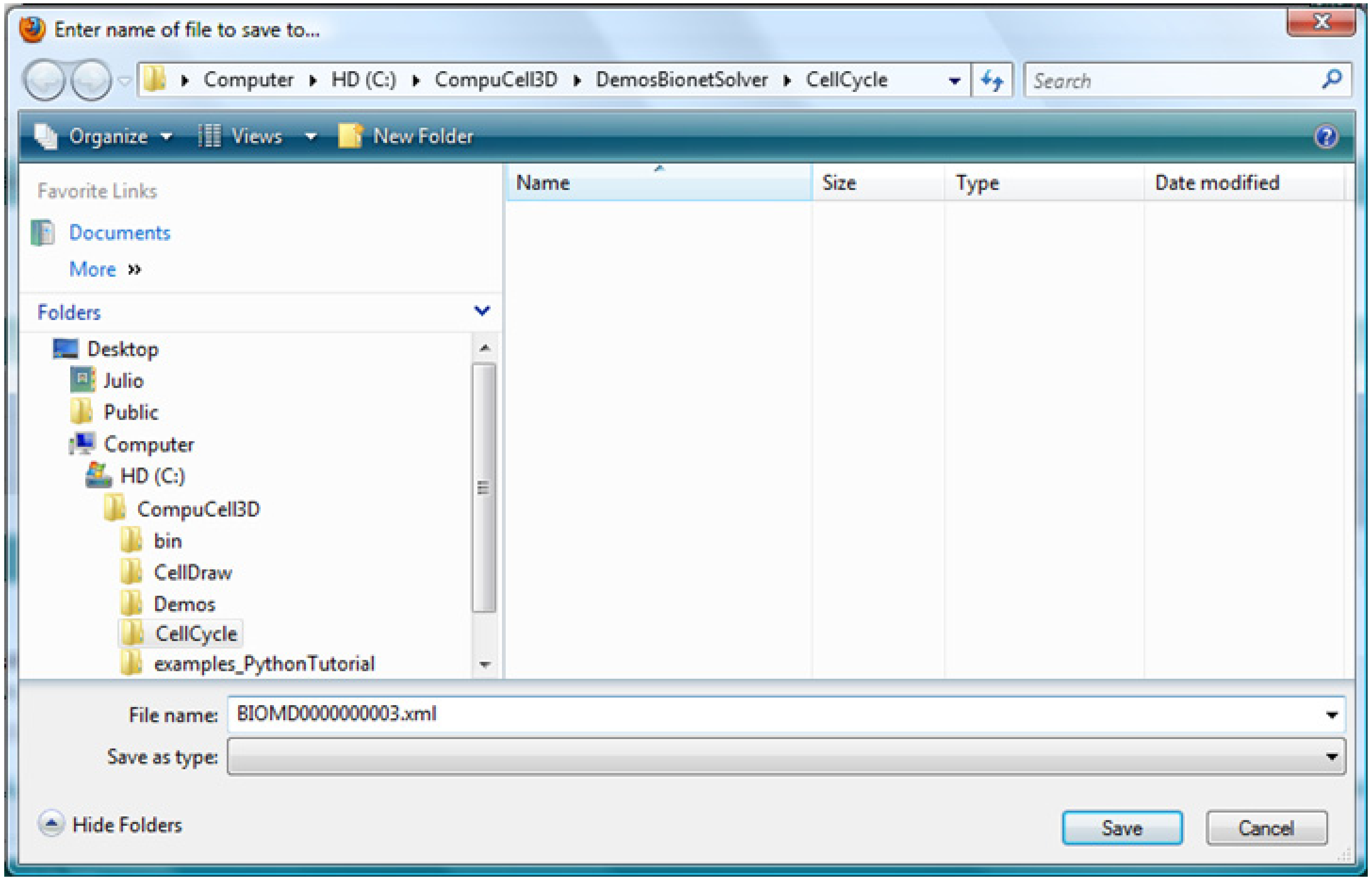
set #1	bqbiol:occursIn	**Taxonomy** Amphibia
set #2	bqbiol:isVersionOf	**KEGG Pathway** hsa04110 **Gene Ontology** mitotic cell cycle
	bqbiol:isHomologTo	**Reactome** REACT_152

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

Second Example – Cell Cycle from web

- Save the file *BIOMD0000000003.xml* anywhere in your computer, later we will transfer it to the appropriate directory.



File name: BIOMD0000000003.xml

Save as type:

Second Example – Cell Cycle from web

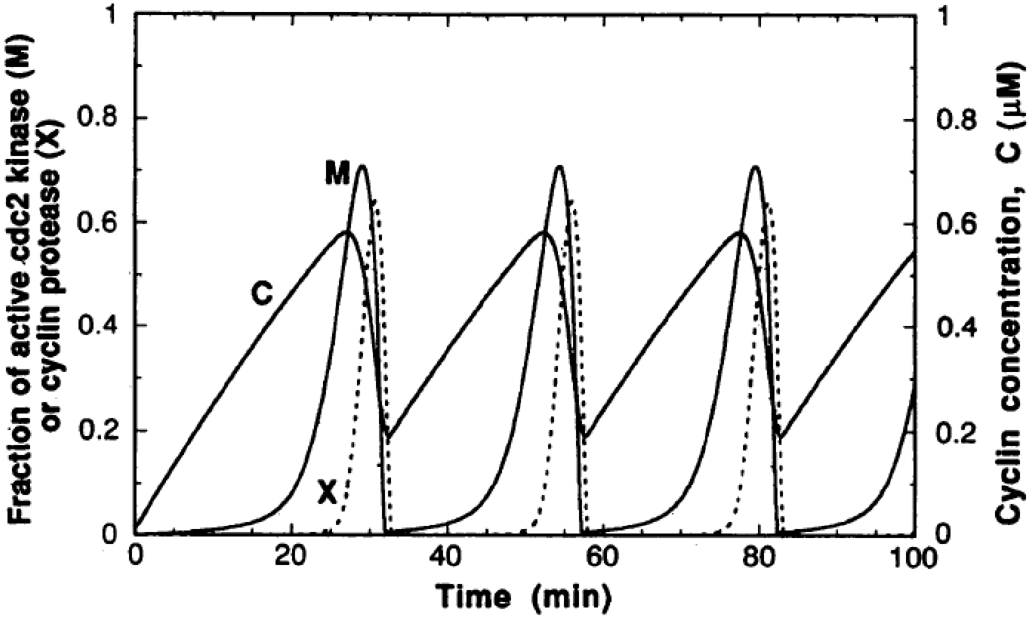
- This model is composed of 3 ODEs that forms an oscillating system:

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

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

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

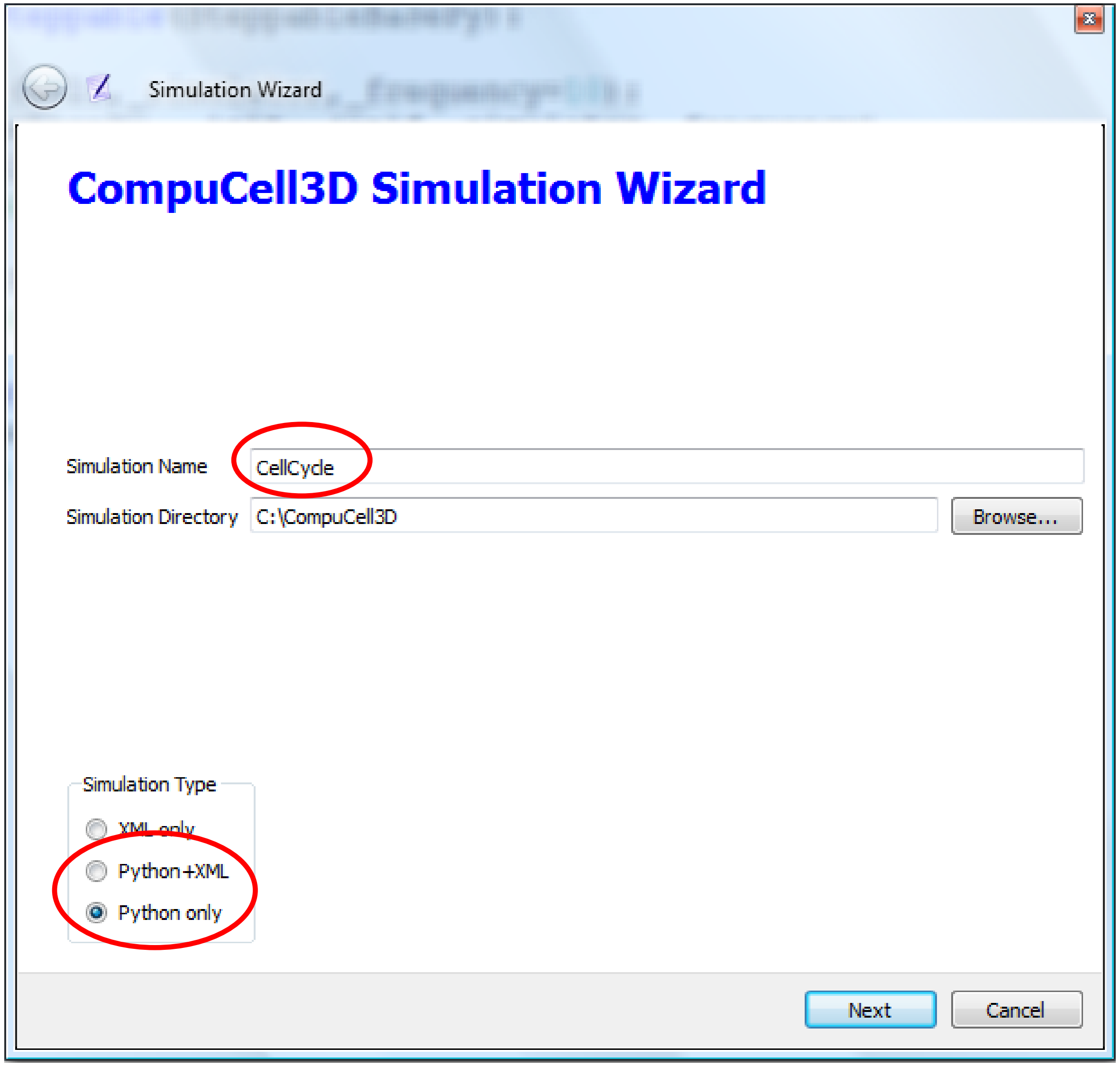
$$V_1 = \frac{C}{K_c + C} V_{M1}, \; V_3 = M V_{M3}.$$



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

Second Example – Cell Cycle from web

- Using Tweddit++ Wizard, create a new simulation:



Simulation Name: CellCycle

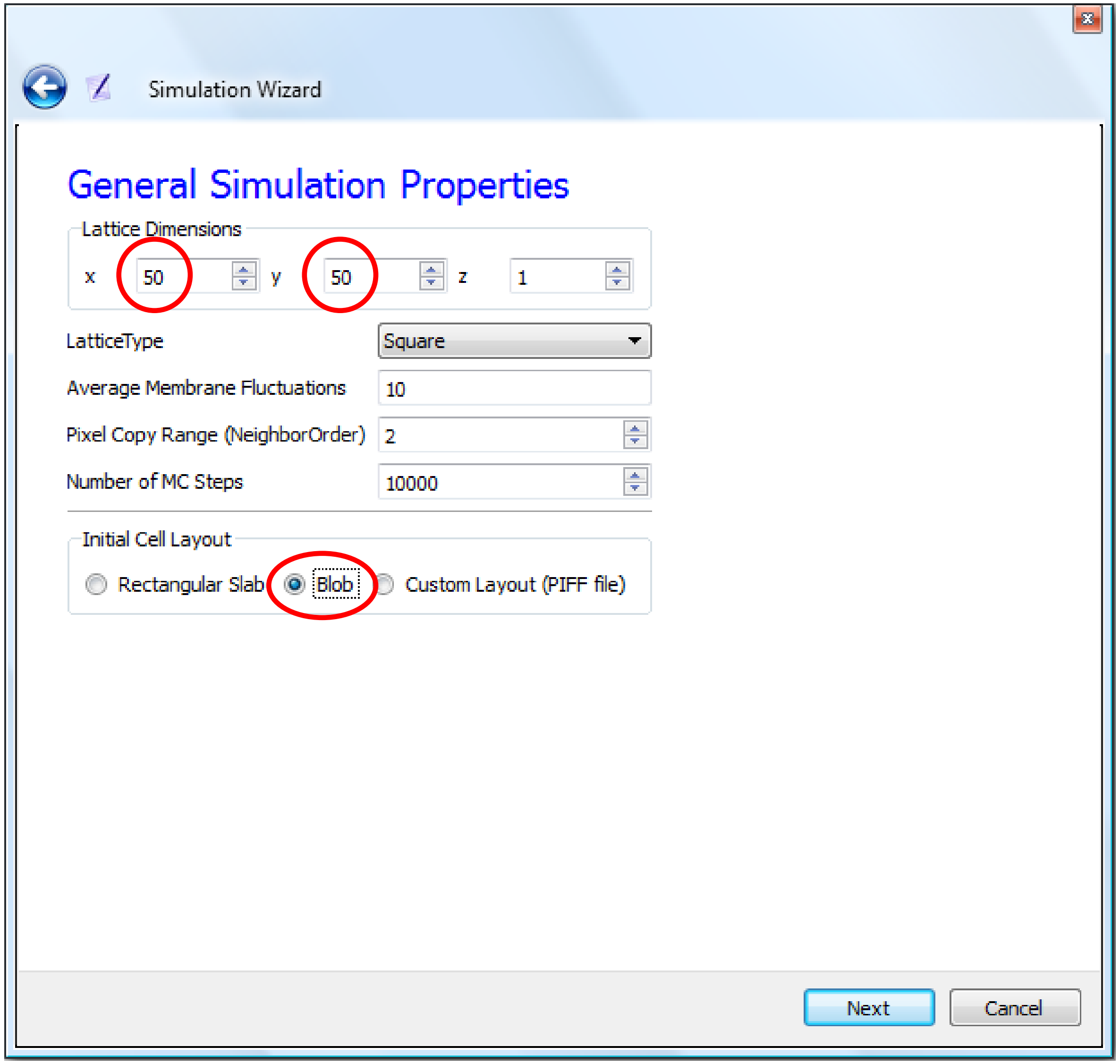
Simulation Directory: C:\CompuCell3D

Simulation Type

- XML only
- Python+XML
- [✓ Python only]

Second Example – Cell Cycle from web

- Make the cell lattice 50x50 and the initial layout a blob:



Simulation Wizard

General Simulation Properties

Lattice Dimensions

x 50 y 50 z 1

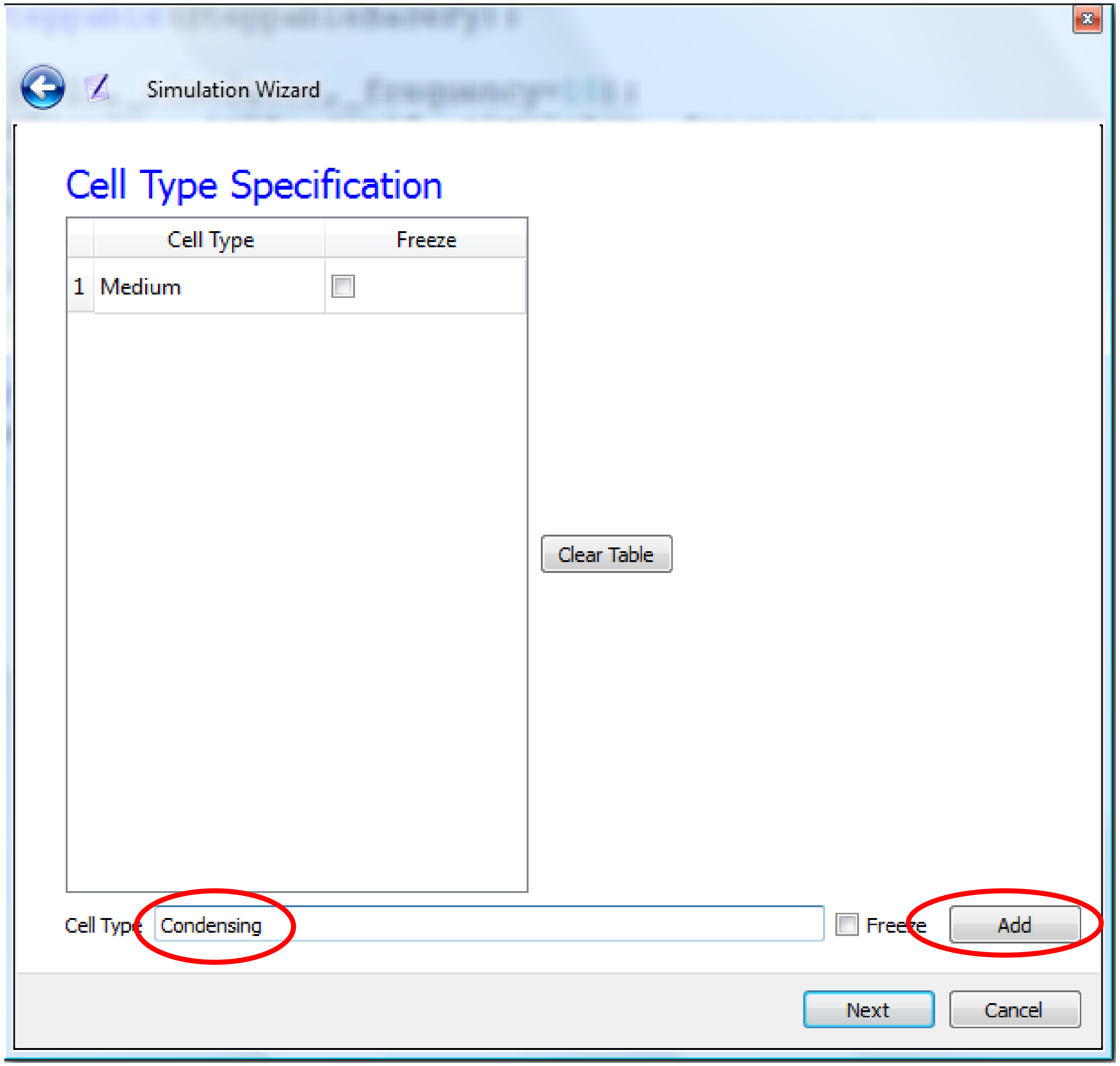
LatticeType	Square
Average Membrane Fluctuations	10
Pixel Copy Range (NeighborOrder)	2
Number of MC Steps	10000

Initial Cell Layout

Rectangular Slab [✓ Blob] Custom Layout (PIFF file)

Second Example – Cell Cycle from web

- Add cell type “Condensing”:



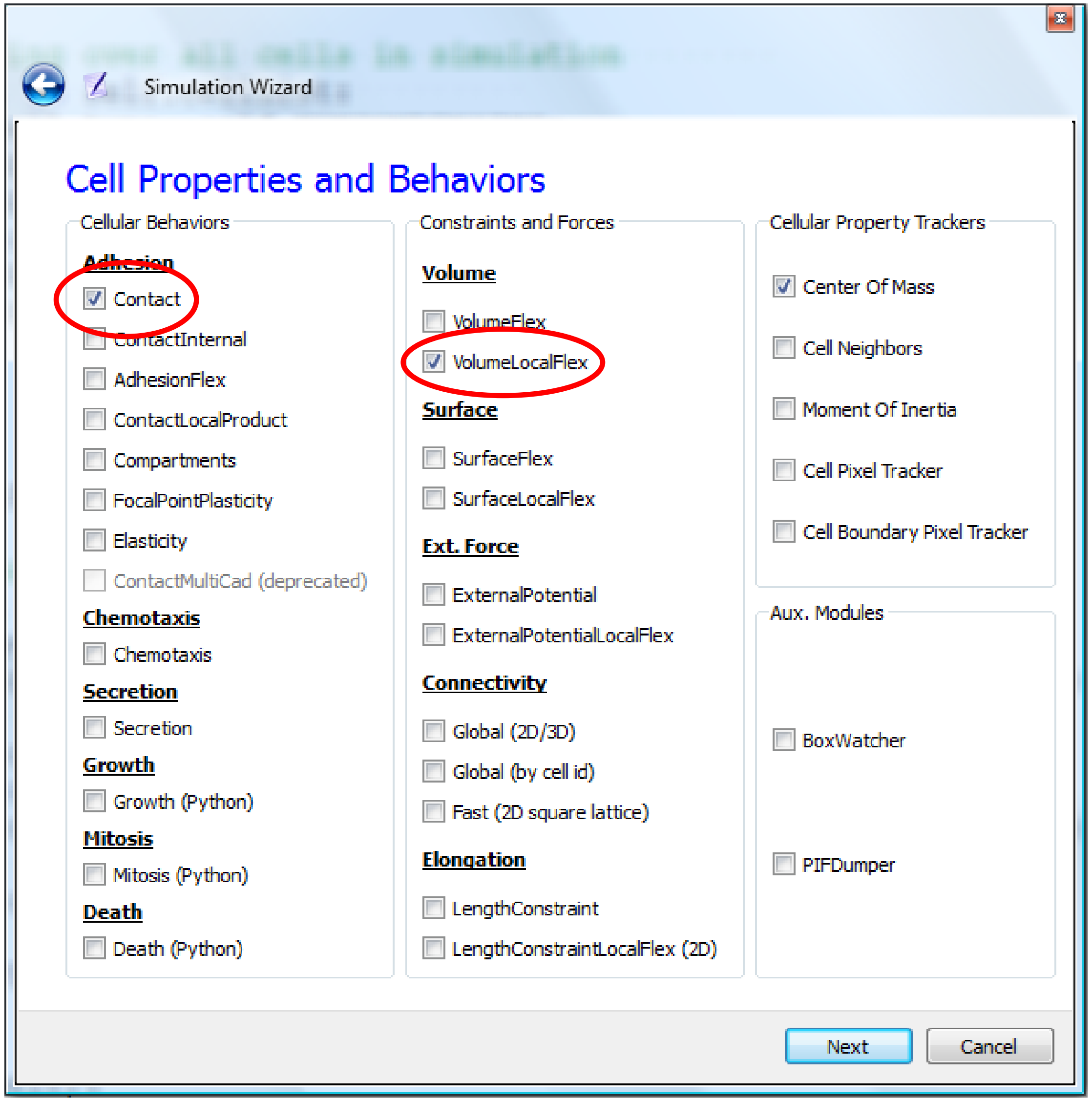
Cell Type Specification

	Cell Type	Freeze
1	Medium	☐

Cell Type: Condensing ☐ Freeze

Second Example – Cell Cycle from web

- Select Contact and VolumeLocalFlex



Simulation Wizard

Cell Properties and Behaviors

Cellular Behaviors

Adhesion

- [✓] Contact
- [ ] ContactInternal
- [ ] AdhesionFlex
- [ ] ContactLocalProduct
- [ ] Compartments
- [ ] FocalPointPlasticity
- [ ] Elasticity
- [ ] ContactMultiCad (deprecated)

Chemotaxis

- [ ] Chemotaxis

Secretion

- [ ] Secretion

Growth

- [ ] Growth (Python)

Mitosis

- [ ] Mitosis (Python)

Death

- [ ] Death (Python)

Constraints and Forces

Volume

- [ ] VolumeFlex
- [✓] VolumeLocalFlex

Surface

- [ ] SurfaceFlex
- [ ] SurfaceLocalFlex

Ext. Force

- [ ] ExternalPotential
- [ ] ExternalPotentialLocalFlex

Connectivity

- [ ] Global (2D/3D)
- [ ] Global (by cell id)
- [ ] Fast (2D square lattice)

Elongation

- [ ] LengthConstraint
- [ ] LengthConstraintLocalFlex (2D)

Cellular Property Trackers

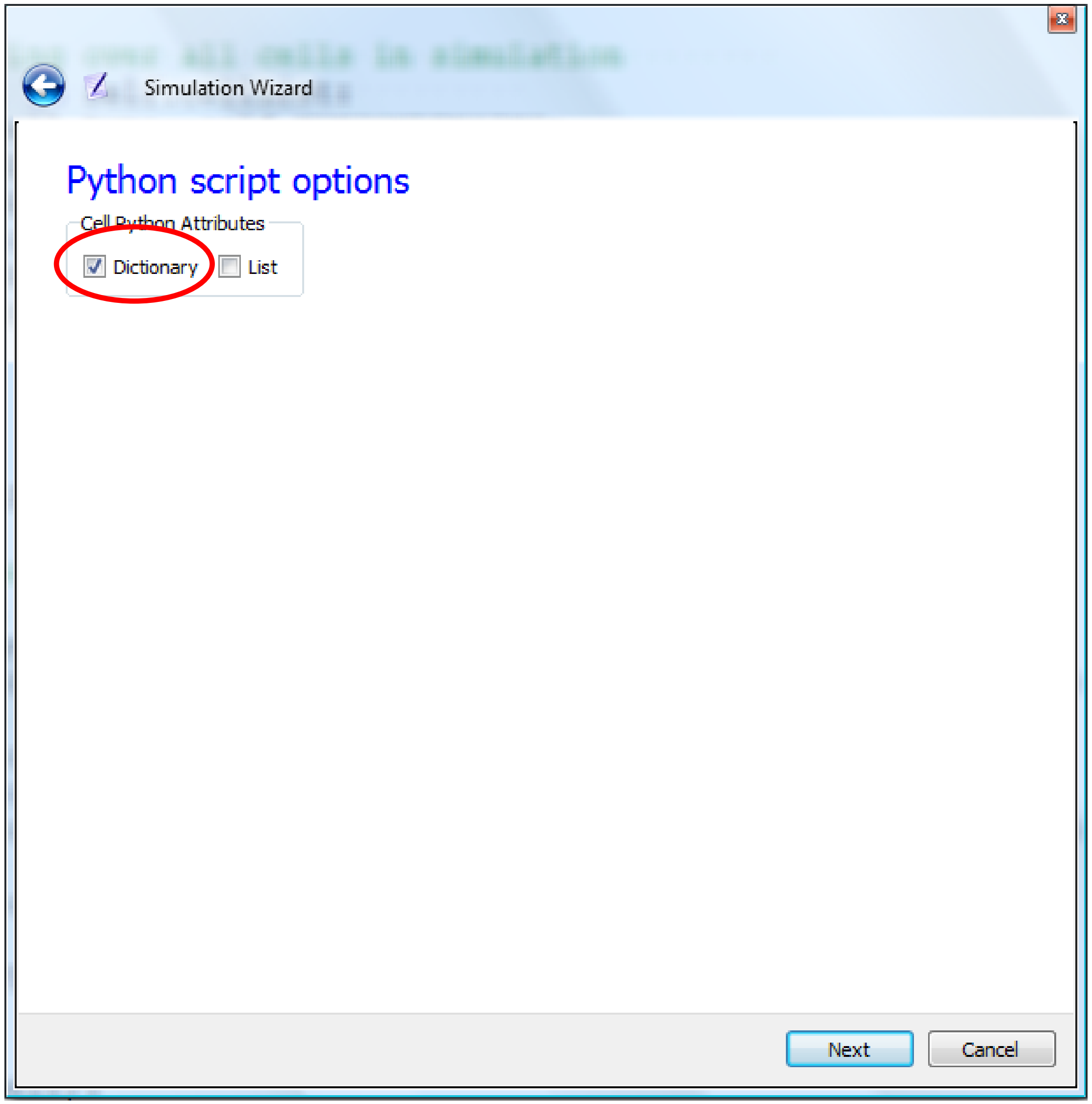
- [✓] Center Of Mass
- [ ] Cell Neighbors
- [ ] Moment Of Inertia
- [ ] Cell Pixel Tracker
- [ ] Cell Boundary Pixel Tracker

Aux. Modules

- [ ] BoxWatcher
- [ ] PIFDumper

Second Example – Cell Cycle from web

- Add a dictionary to the cells



Simulation Wizard

Python script options

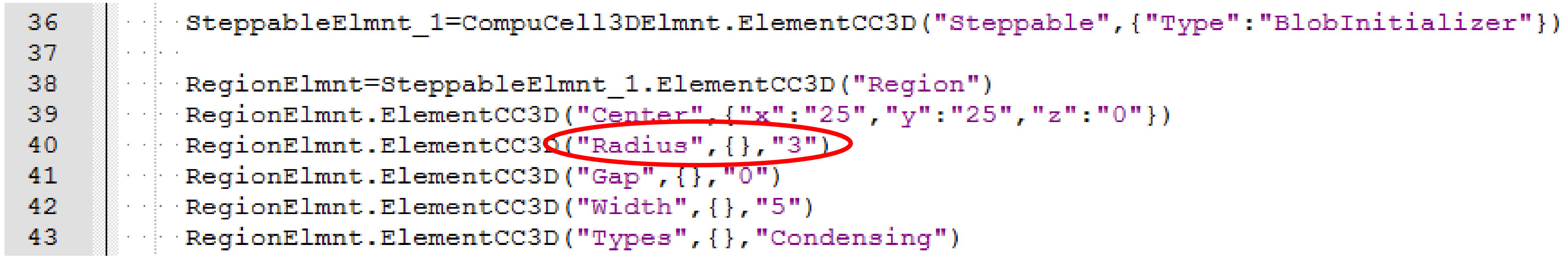
Cell Python Attributes

[✓ Dictionary] List

Second Example – Cell Cycle from web

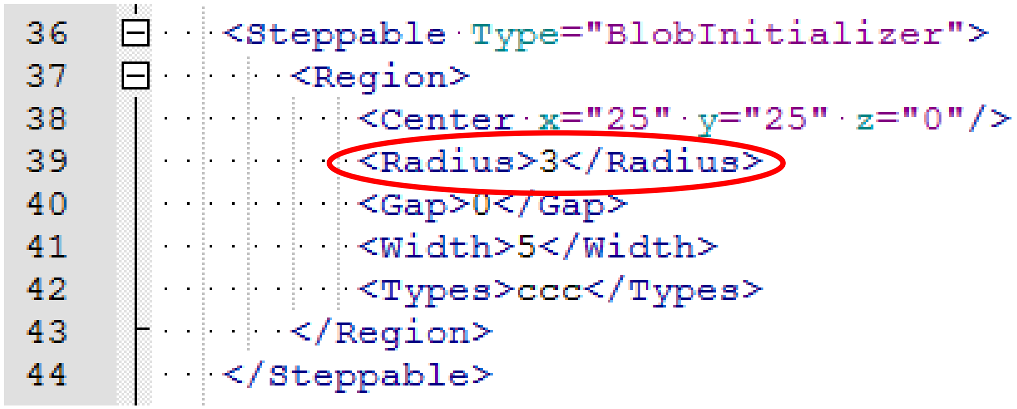
- For the simulation to run faster, change the BlobInitializer so that the simulation only has 1 cell:

- If you are using “Python only” option:



```
    SteppableElmnt_1=CompuCell3DElmnt.ElementCC3D("Steppable",{"Type":"BlobInitializer"})

    RegionElmnt=SteppableElmnt_1.ElementCC3D("Region")
    RegionElmnt.ElementCC3D("Center",{"x":"25","y":"25","z":"0"})
    RegionElmnt.ElementCC3D("Radius",{},"3")
    RegionElmnt.ElementCC3D("Gap",{},"0")
    RegionElmnt.ElementCC3D("Width",{},"5")
    RegionElmnt.ElementCC3D("Types",{},"Condensing")
```

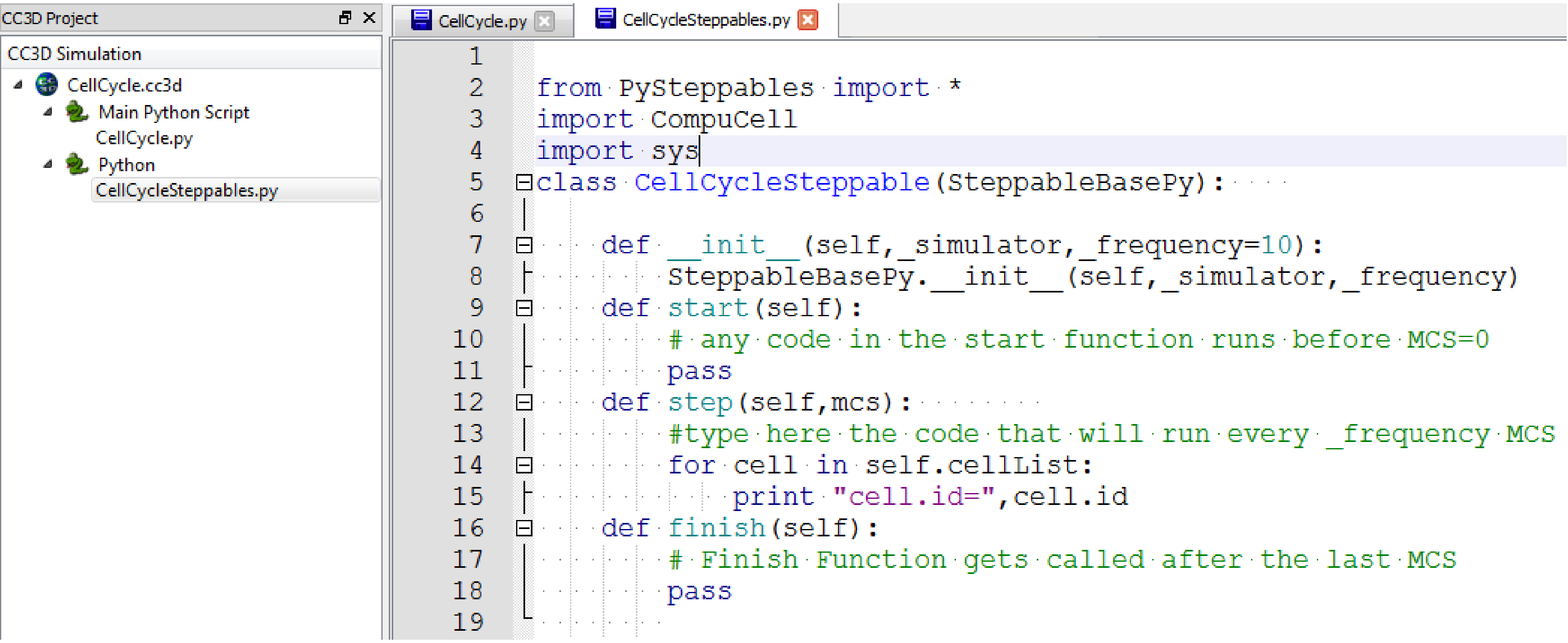
- If you are using XML:



```
    <Steppable Type="BlobInitializer">
        <Region>
            <Center x="25" y="25" z="0"/>
            <Radius>3</Radius>
            <Gap>0</Gap>
            <Width>5</Width>
            <Types>ccc</Types>
        </Region>
    </Steppable>
```

Second Example – Cell Cycle from web

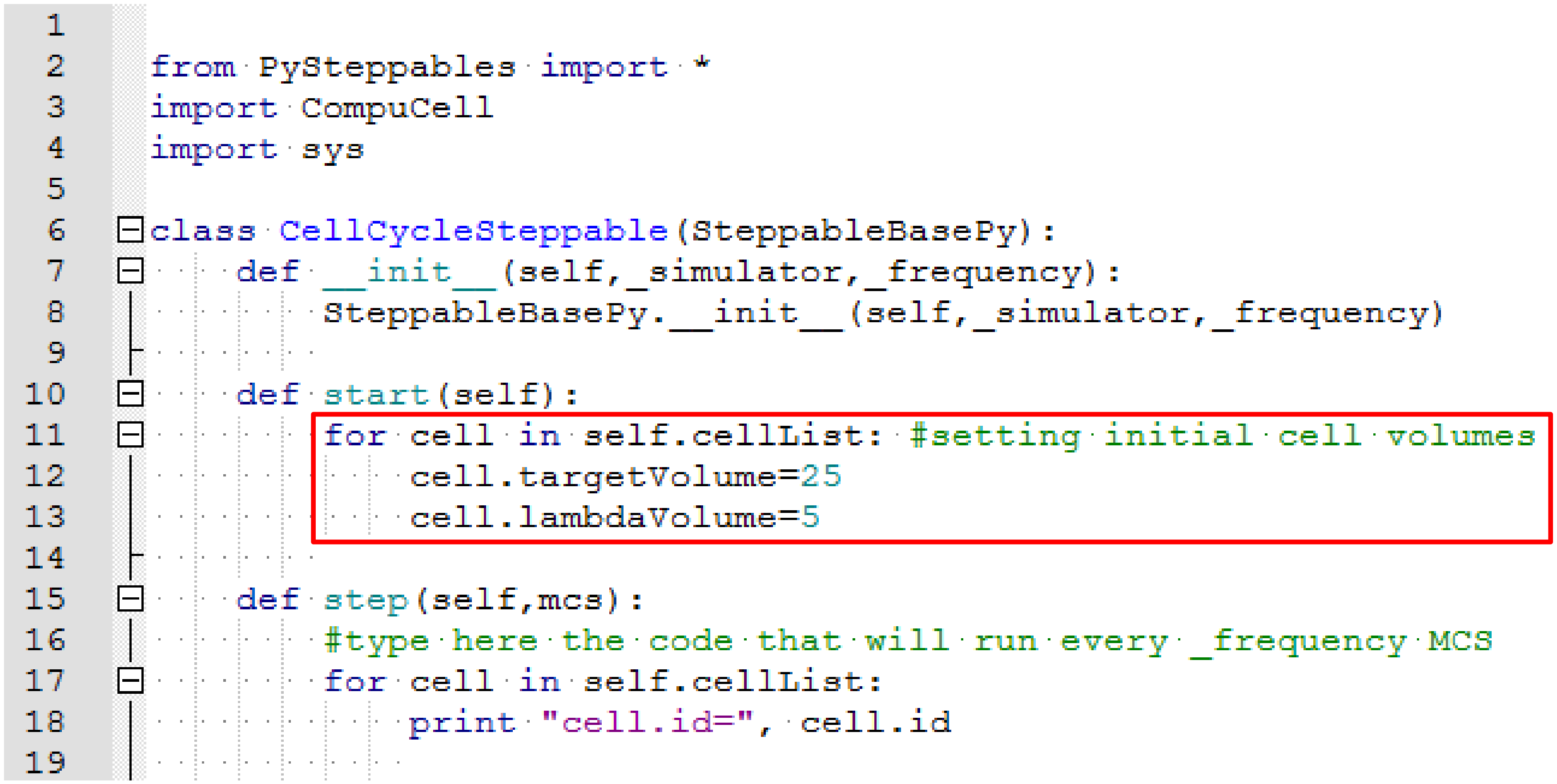
- Your Steppable file (*CellCycleSteppables.py*) will look like that:



```

from PySteppables import *
import CompuCell
import sys
class CellCycleSteppable(SteppableBasePy):

    def __init__(self,_simulator,_frequency=10):
        SteppableBasePy.__init__(self,_simulator,_frequency)
    def start(self):
        # any code in the start function runs before MCS=0
        pass
    def step(self,mcs):
        #type here the code that will run every _frequency MCS
        for cell in self.cellList:
            print "cell.id=",cell.id
    def finish(self):
        # Finish Function gets called after the last MCS
        pass

```

Second Example – Cell Cycle from web

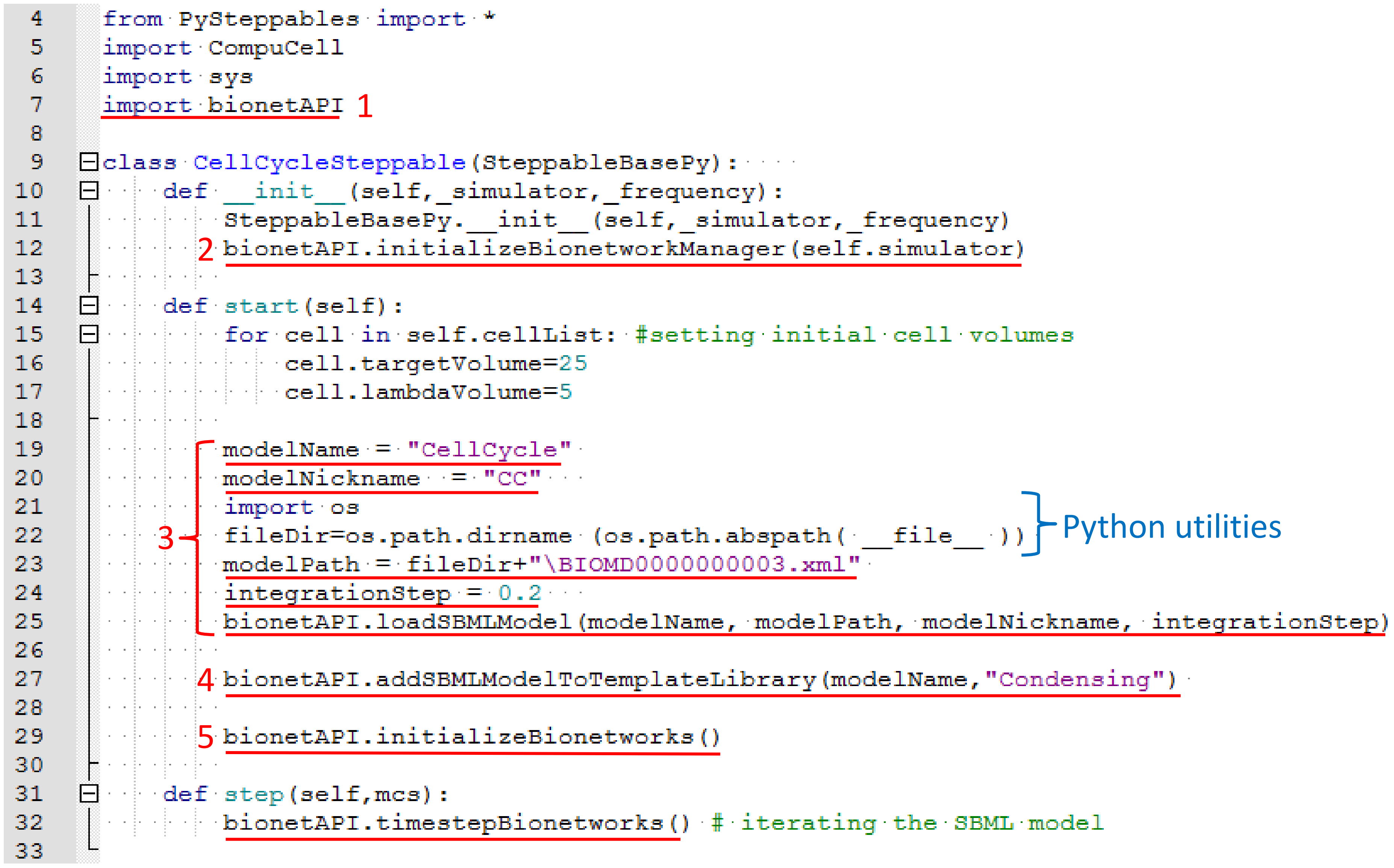
- Initialize the cell volume constraints:



```

from PySteppables import *
import CompuCell
import sys

class CellCycleSteppable(SteppableBasePy):
    def __init__(self,_simulator,_frequency):
        SteppableBasePy.__init__(self,_simulator,_frequency)

    def start(self):
        for cell in self.cellList: #setting initial cell volumes
            cell.targetVolume=25
            cell.lambdaVolume=5

    def step(self,mcs):
        #type here the code that will run every _frequency MCS
        for cell in self.cellList:
            print "cell.id=", cell.id

```

Second Example – Cell Cycle from web

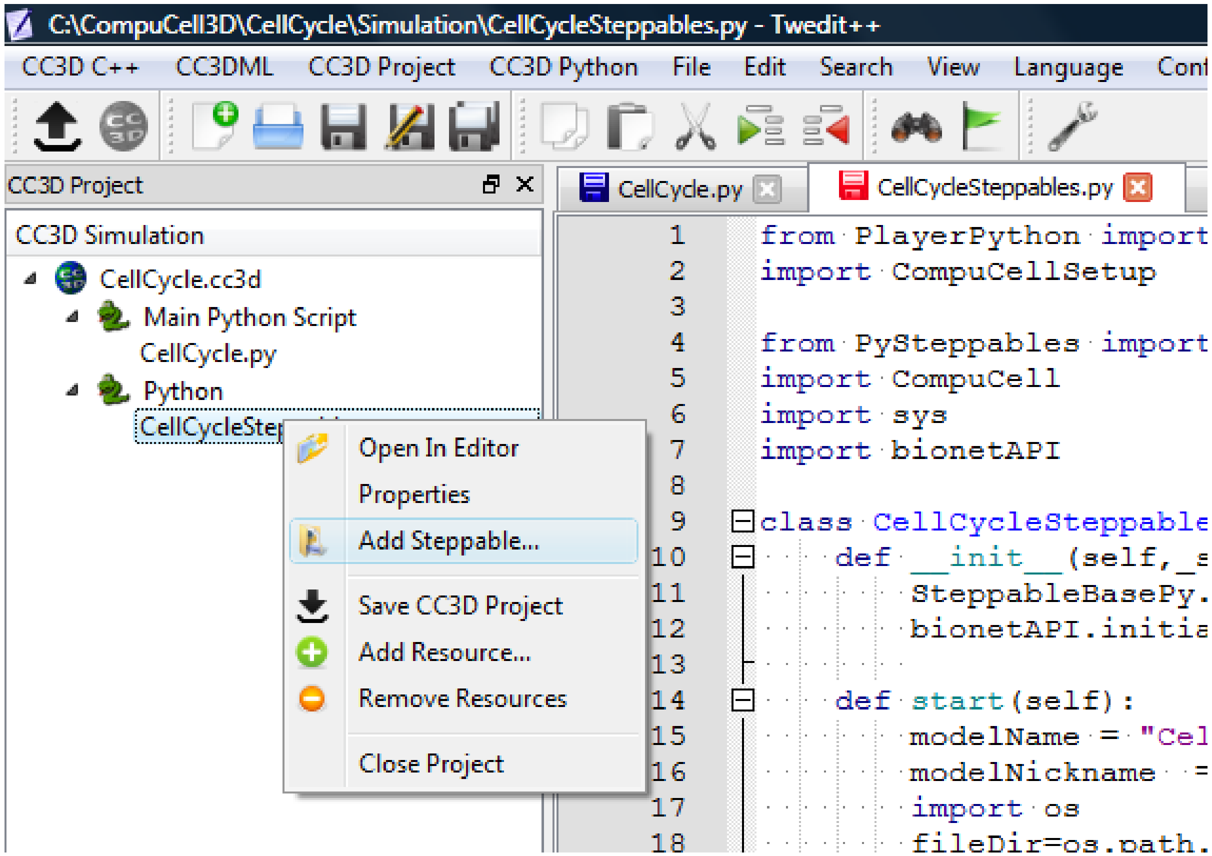
- Now fill in the BionetSolver commands:



```
from PySteppables import *
import CompuCell
import sys
import bionetAPI

class CellCycleSteppable(SteppableBasePy):
    def __init__(self,_simulator,_frequency):
        SteppableBasePy.__init__(self,_simulator,_frequency)
        bionetAPI.initializeBionetworkManager(self.simulator)

    def start(self):
        for cell in self.cellList: #setting initial cell volumes
            cell.targetVolume=25
            cell.lambdaVolume=5

        modelName = "CellCycle"
        modelNickname  = "CC"
        import os
        fileDir=os.path.dirname (os.path.abspath( __file__ ))
        modelPath = fileDir+"\BIOMD0000000003.xml"
        integrationStep = 0.2
        bionetAPI.loadSBMLModel(modelName, modelPath, modelNickname, integrationStep)

        bionetAPI.addSBMLModelToTemplateLibrary(modelName,"Condensing")

        bionetAPI.initializeBionetworks()

    def step(self,mcs):
        bionetAPI.timestepBionetworks() # iterating the SBML model
```

Second Example – Cell Cycle from web

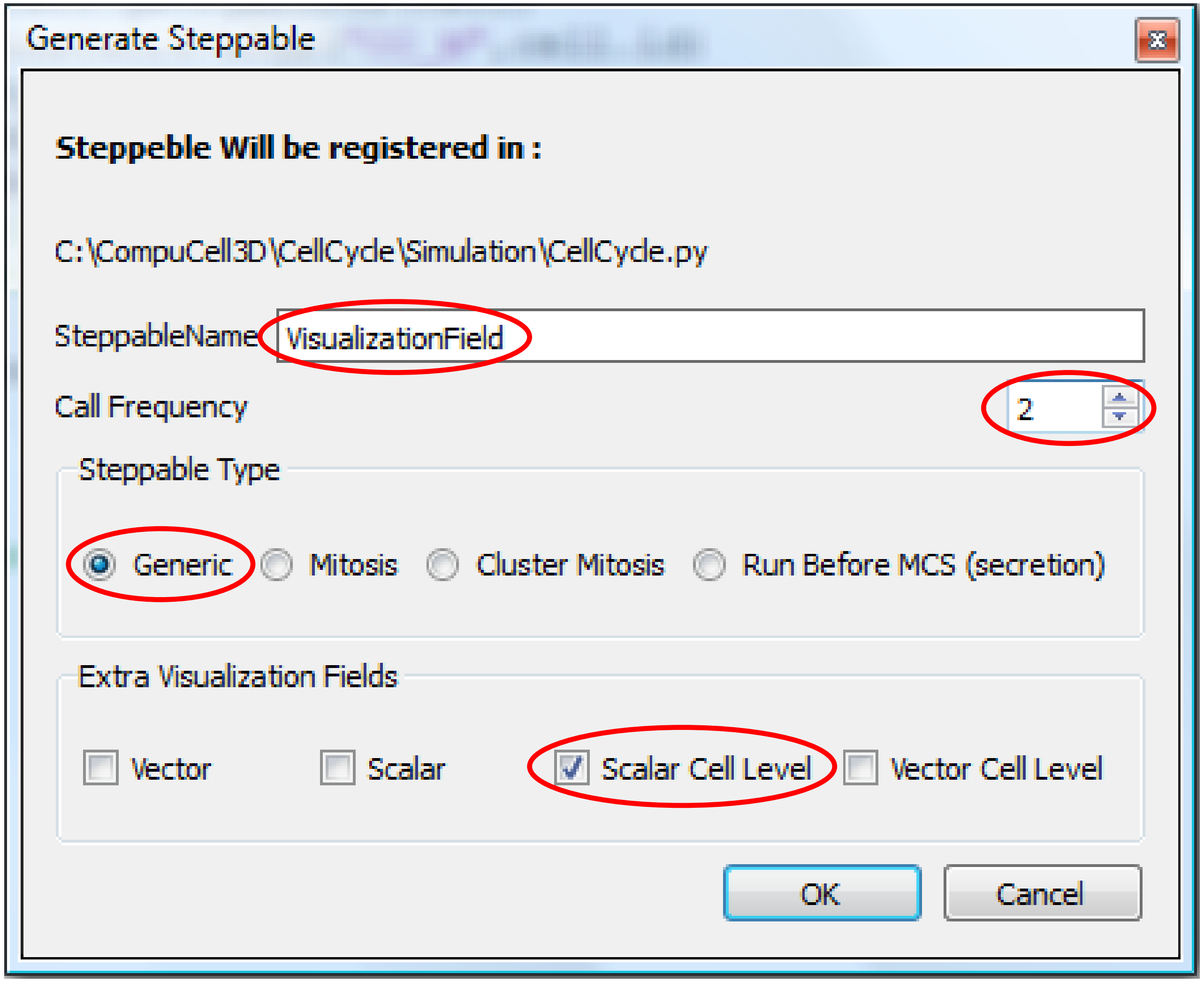
- Right now the simulation should work, the cells are running the SBML model, but we can't see anything.
- Let's create a visualization field for the fraction of Cdc2 kinase:



```
from PlayerPython import
import CompuCellSetup

from PySteppables import
import CompuCell
import sys
import bionetAPI

class CellCycleSteppable
    def __init__(self,_s
        SteppableBasePy.
        bionetAPI.initia

    def start(self):
        modelName = "Cel
        modelNickname  =
        import os
        fileDir=os.path.
```

Second Example – Cell Cycle from web

- Create a visualization field for the fraction of Cdc2 kinase.



Generate Steppable

Steppeble Will be registered in :

C:\CompuCell3D\CellCycle\Simulation\CellCycle.py

StepppableName: VisualizationField

Call Frequency: 2

Steppable Type

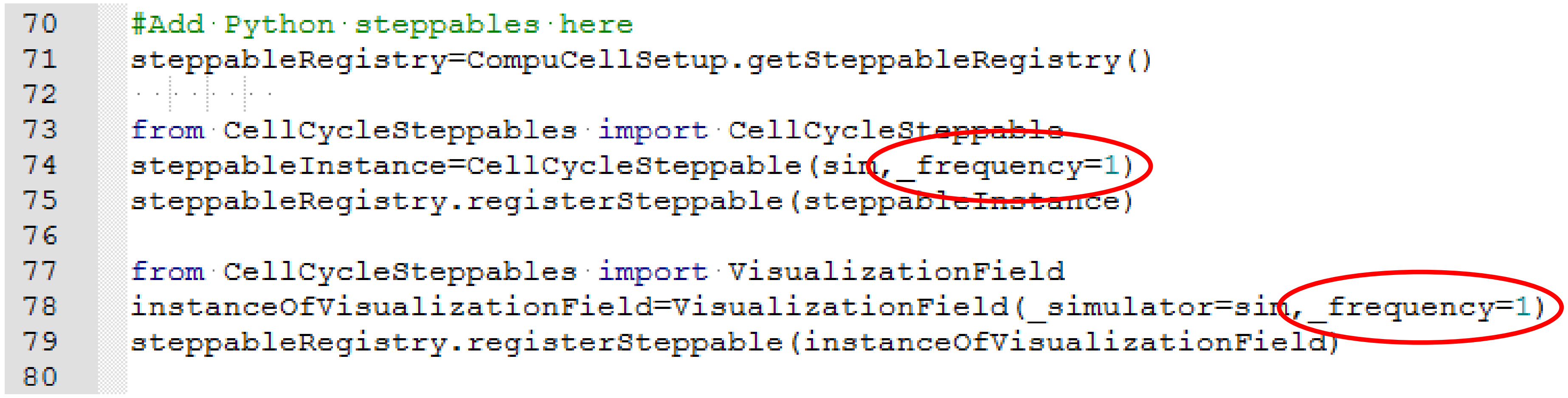
[✓ Generic] Mitosis Cluster Mitosis Run Before MCS (secretion)

Extra Visualization Fields

Vector Scalar [✓ Scalar Cell Level] Vector Cell Level

Second Example – Cell Cycle from web

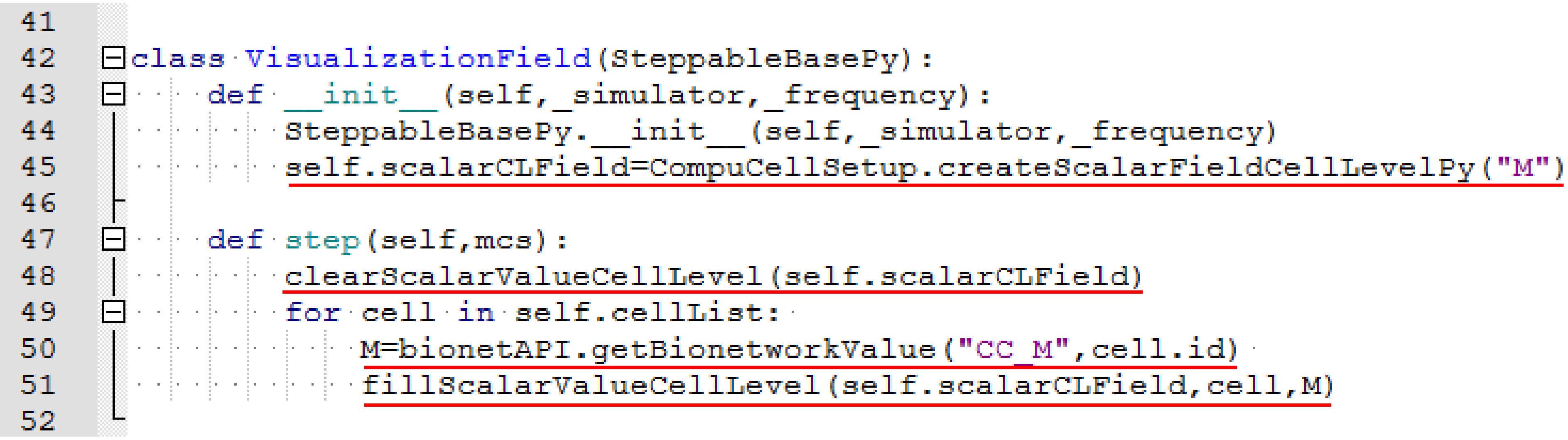
- On the main Python file (*CellCycle.py*) make all frequencies equal to 1:



```
#Add Python steppables here
steppableRegistry=CompuCellSetup.getSteppableRegistry()
    
from CellCycleSteppables import CellCycleSteppable
steppableInstance=CellCycleSteppable(sim,_frequency=1)
steppableRegistry.registerSteppable(steppableInstance)

from CellCycleSteppables import VisualizationField
instanceOfVisualizationField=VisualizationField(_simulator=sim,_frequency=1)
steppableRegistry.registerSteppable(instanceOfVisualizationField)
```

Second Example – Cell Cycle from web

- On the Steppable file (*CellCycleSteppables.py*) modify/add the following lines:

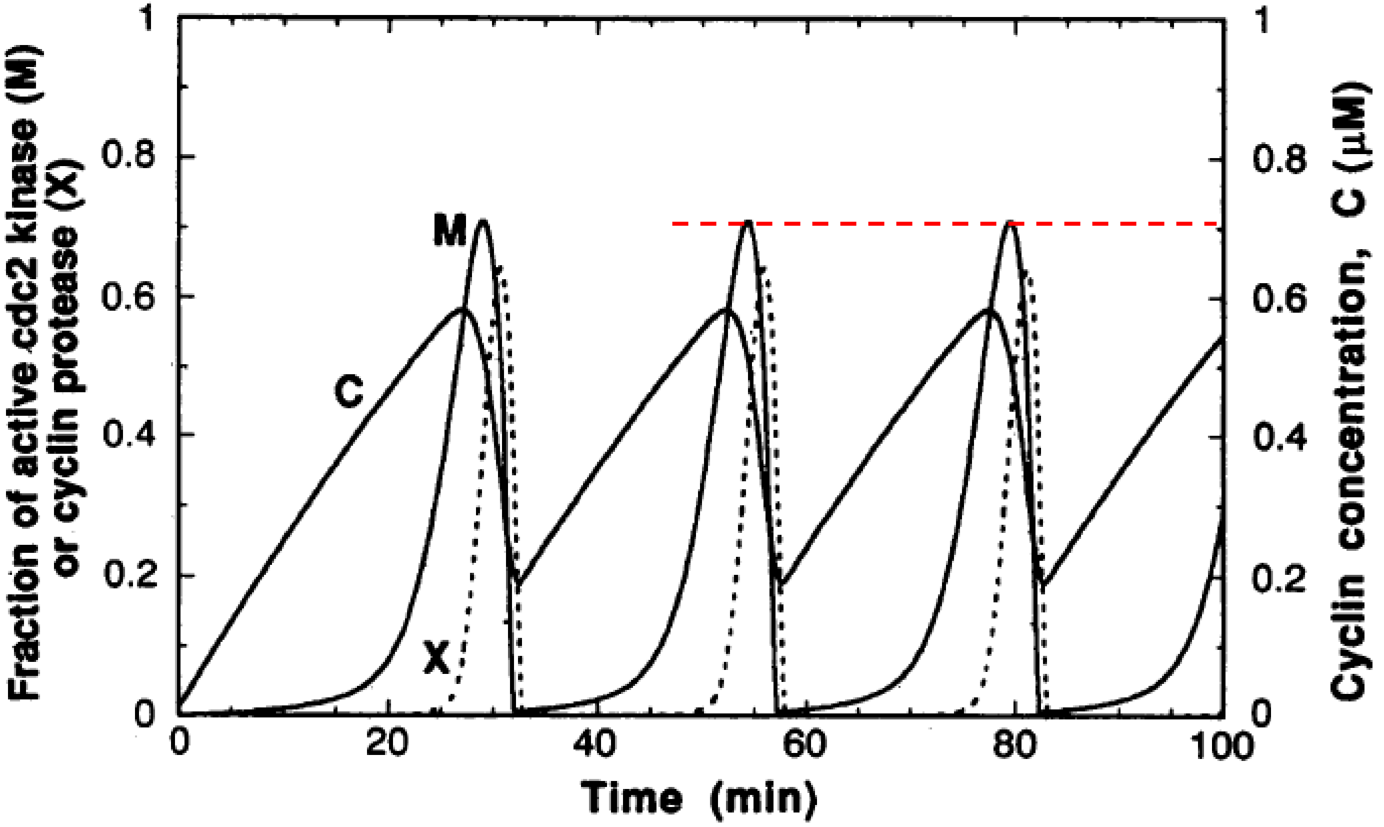


```
class VisualizationField(SteppableBasePy):
    def __init__(self,_simulator,_frequency):
        SteppableBasePy.__init__(self,_simulator,_frequency)
        self.scalarCLField=CompuCellSetup.createScalarFieldCellLevelPy("M")

    def step(self,mcs):
        clearScalarValueCellLevel(self.scalarCLField)
        for cell in self.cellList:
            M=bionetAPI.getBionetworkValue("CC_M",cell.id)
            fillScalarValueCellLevel(self.scalarCLField,cell,M)
```

- The first underlined command creates a field called “M”
- The second clears the field every MCS
- The third stores the current value of the M variable
- The last fills the current cell with the stored value

Second Example – Cell Cycle from web

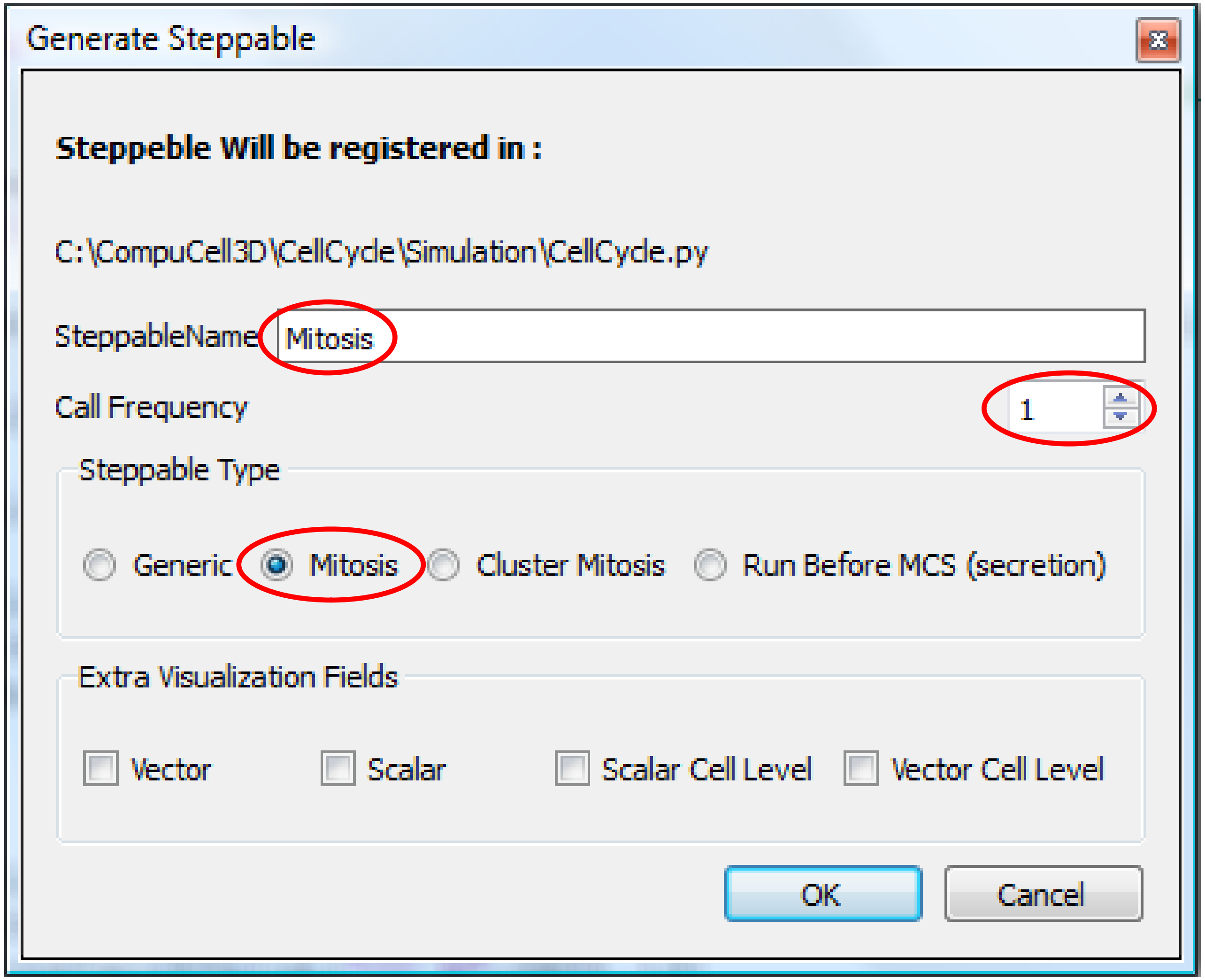
- Mitosis occur when fraction of active Cdc2 kinase (M) reaches 0.7.



- To model this we need to track the concentration of M in each cell and check when it passes the 0.7 mark.

Second Example – Cell Cycle from web

- Add a Steppable to divide cells based on the fraction of Cdc2 kinase:



Generate Steppable

Steppeble Will be registered in :

C:\CompuCell3D\CellCycle\Simulation\CellCycle.py

SteppableName: Mitosis

Call Frequency: 1

Steppable Type

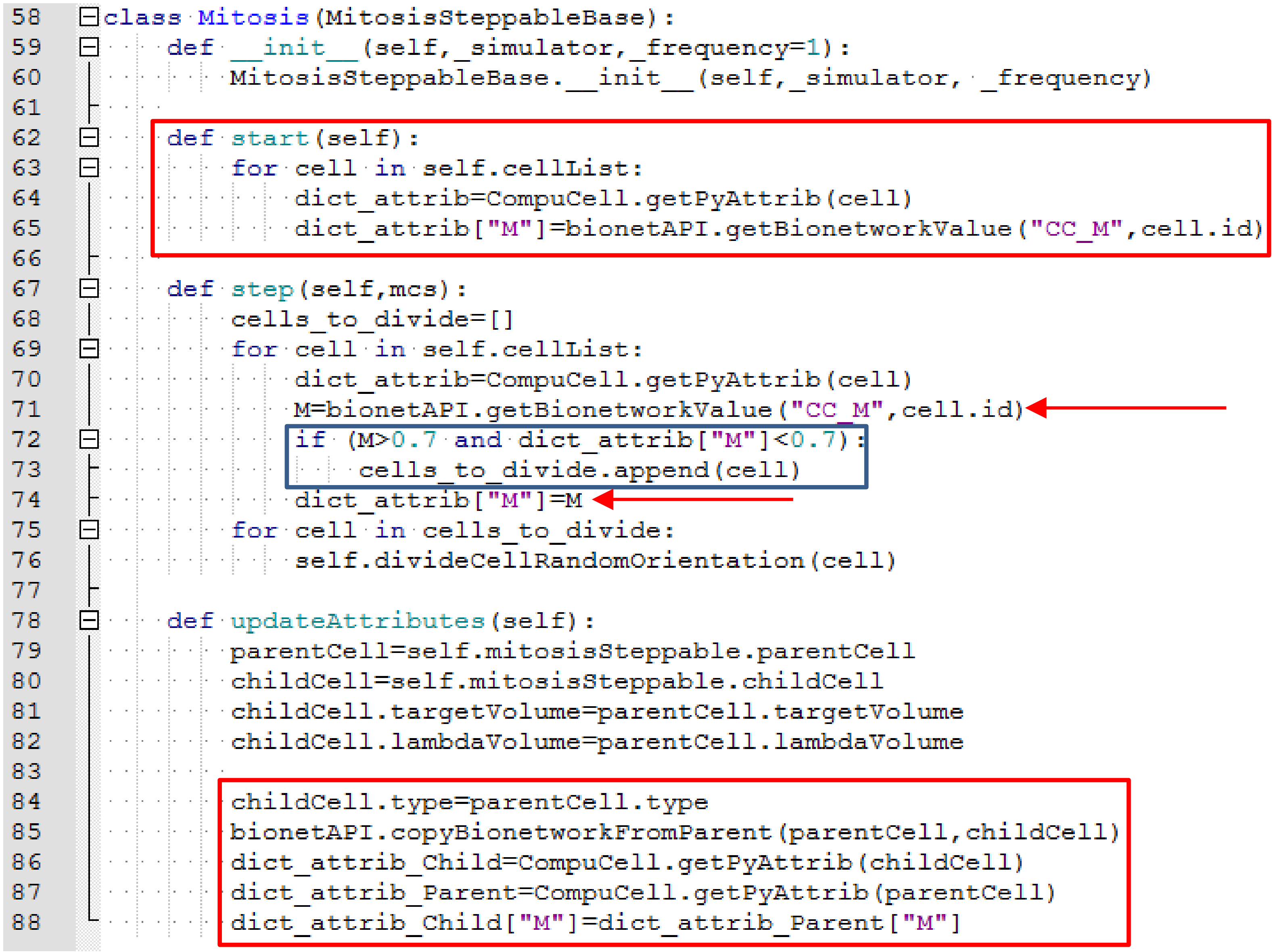
Generic [✓ Mitosis] Cluster Mitosis Run Before MCS (secretion)

Extra Visualization Fields

☐ Vector ☐ Scalar ☐ Scalar Cell Level ☐ Vector Cell Level

Second Example – Cell Cycle from web

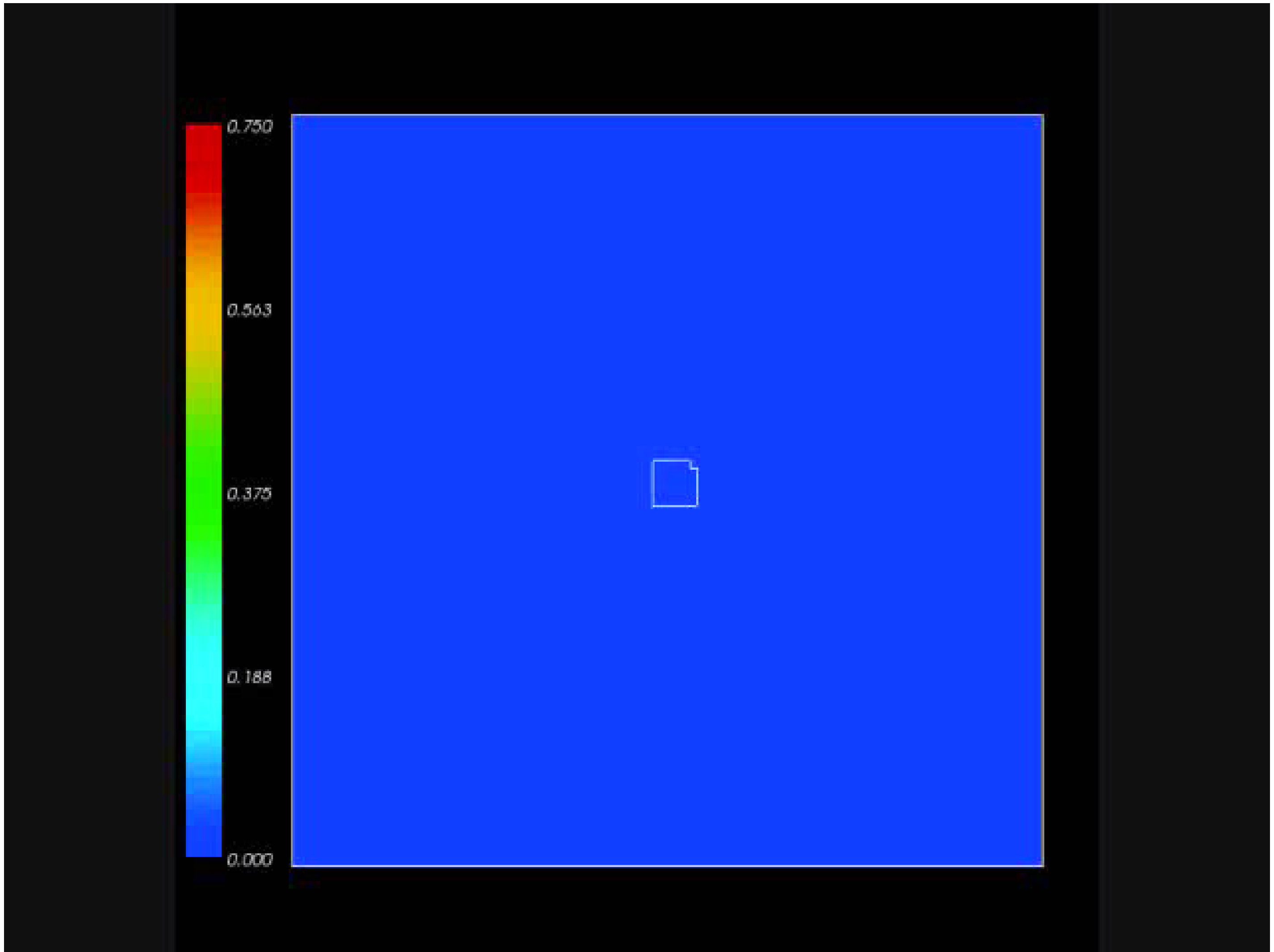
- Go to the Steppable and add the following lines:



```
class Mitosis(MitosisSteppableBase):
    def __init__(self,_simulator,_frequency=1):
        MitosisSteppableBase.__init__(self,_simulator, _frequency)

    def start(self):
        for cell in self.cellList:
            dict_attrib=CompuCell.getPyAttrib(cell)
            dict_attrib["M"]=bionetAPI.getBionetworkValue("CC_M",cell.id)

    def step(self,mcs):
        cells_to_divide=[]
        for cell in self.cellList:
            dict_attrib=CompuCell.getPyAttrib(cell)
            M=bionetAPI.getBionetworkValue("CC_M",cell.id)
            if (M>0.7 and dict_attrib["M"]<0.7):
                cells_to_divide.append(cell)
            dict_attrib["M"]=M
        for cell in cells_to_divide:
            self.divideCellRandomOrientation(cell)

    def updateAttributes(self):
        parentCell=self.mitosisSteppable.parentCell
        childCell=self.mitosisSteppable.childCell
        childCell.targetVolume=parentCell.targetVolume
        childCell.lambdaVolume=parentCell.lambdaVolume

        childCell.type=parentCell.type
        bionetAPI.copyBionetworkFromParent(parentCell,childCell)
        dict_attrib_Child=CompuCell.getPyAttrib(childCell)
        dict_attrib_Parent=CompuCell.getPyAttrib(parentCell)
        dict_attrib_Child["M"]=dict_attrib_Parent["M"]
```

Second Example – Cell Cycle from web

- Open the model in CC3D, set the maximum concentration of the “M” field to 0.75, and run the simulation:



Exercises

- 1
  - Change the Volume plugin so that cells slowly grow back to the original target Volume after mitosis
- 2
  - Add a second cell type (NonCondensing) with half of the cycle time

Third Example – Tyson’s Cell Cycle

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

Third Example – Tyson's Cell Cycle

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

$$\boxed{\begin{aligned}
\frac{\mathrm{d}[\mathrm{CycB}]}{\mathrm{d}t} &= k_1 - (k_2' + k_2''\,[\mathrm{Cdh1}])\,[\mathrm{CycB}], \\[2ex]
\frac{\mathrm{d}[\mathrm{Cdh1}]}{\mathrm{d}t} &= \frac{(k_3' + k_3''A)(1 - [\mathrm{Cdh1}])}{J_3 + 1 - [\mathrm{Cdh1}]} - \frac{k_4 m [\mathrm{CycB}]\,[\mathrm{Cdh1}]}{J_4 + [\mathrm{Cdh1}]},
\end{aligned}}$$

$$\frac{\mathrm{d}[\mathrm{Cdc20}_T]}{\mathrm{d}t} = k_5' + k_5''\,\frac{([\mathrm{CycB}]\,m/J_5)^n}{1 + ([\mathrm{CycB}]\,m/J_5)^n} - k_6\,[\mathrm{Cdc20}_T],$$

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

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

Third 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:

$$\frac{d[\mathrm{CycB}]}{dt} = k_1 - (k_2' + k_2''\,[\mathrm{Cdh1}])\,[\mathrm{CycB}],$$

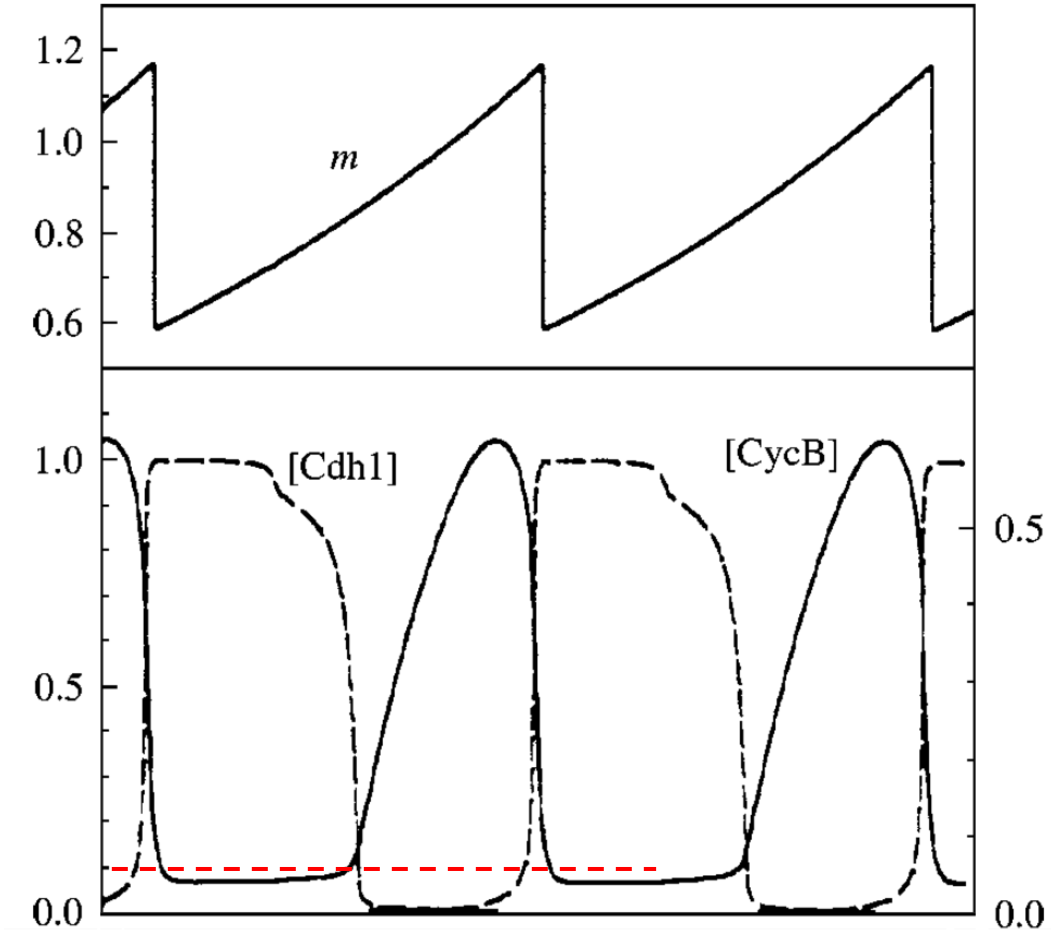
$$\frac{d[\mathrm{Cdh1}]}{dt} = \frac{(k_3' + k_3'' A)(1 - [\mathrm{Cdh1}])}{J_3 + 1 - [\mathrm{Cdh1}]} - \frac{k_4 \boxed{m}\,[\mathrm{CycB}]\,[\mathrm{Cdh1}]}{J_4 + [\mathrm{Cdh1}]},$$

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

$$V_\sigma / V_0$$

Third Example – Tyson's Cell Cycle

- This time mitosis occur when the level of CycB drops below 0.1

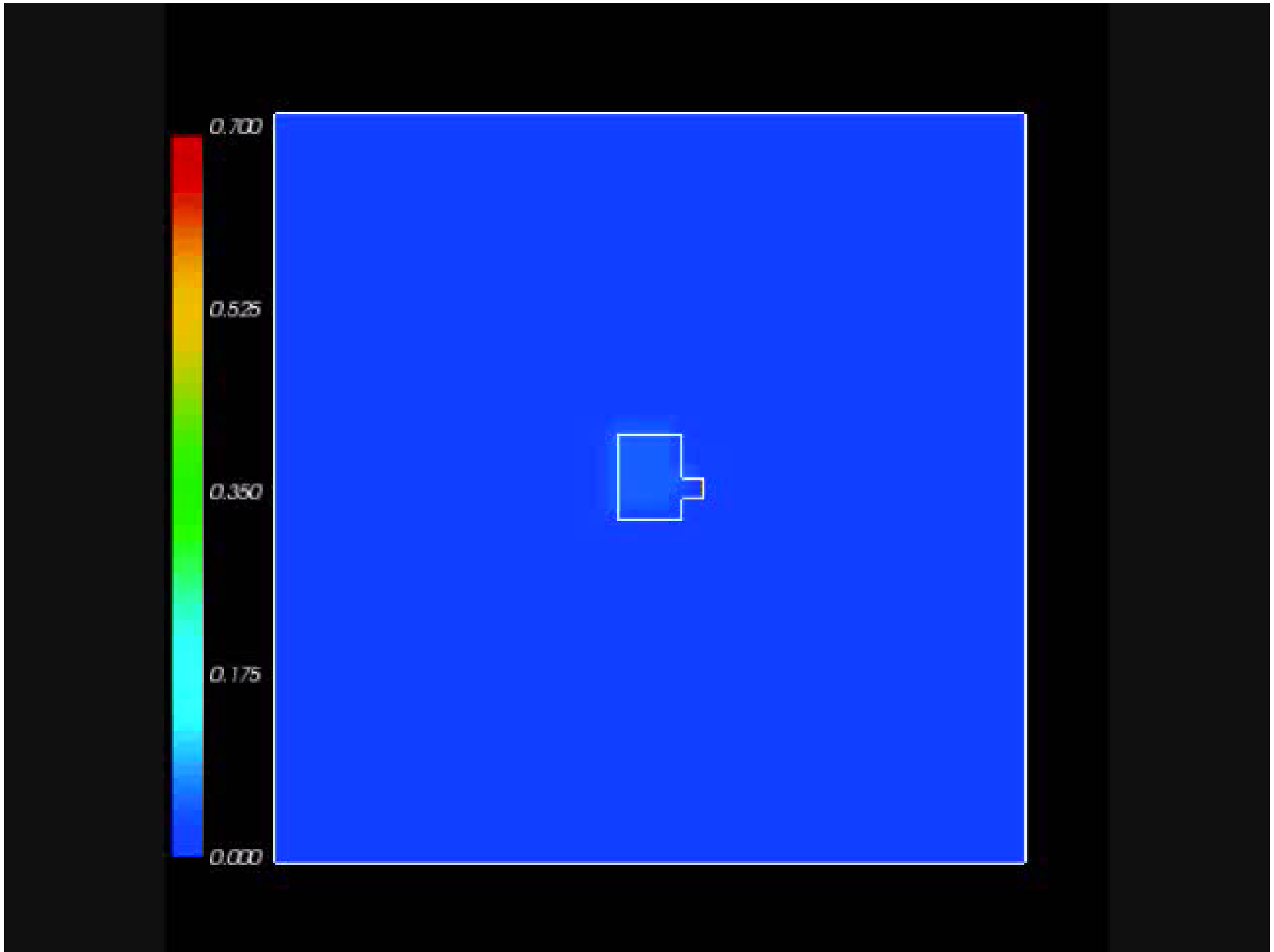


Exercises

- 1
  - Download the paper by Tyson and Novak
  - Code the first two ODEs in Jarnac and generate the SBML
- 2
  - Using the code from the second example as a template, create a model that uses Tyson’s cell cycle

Third 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:



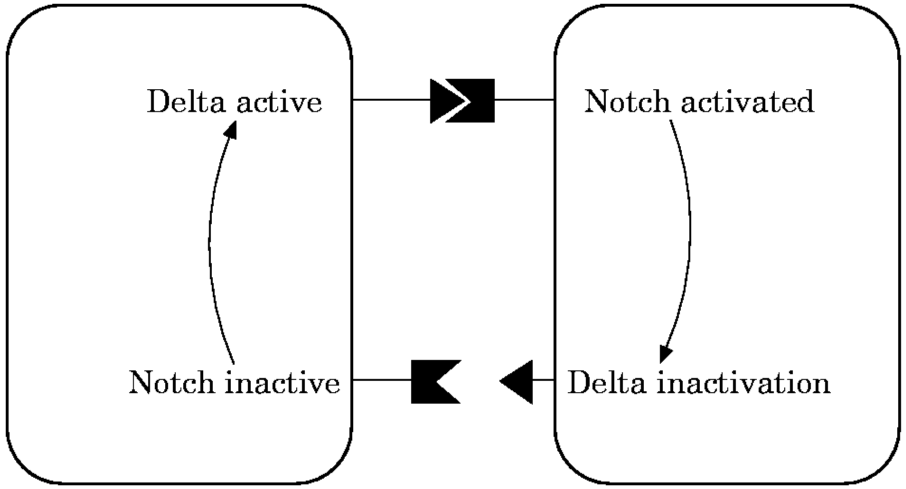
Fourth Example – Delta-Notch Patterning

- The third 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).

Fourth Example – Delta-Notch Patterning

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



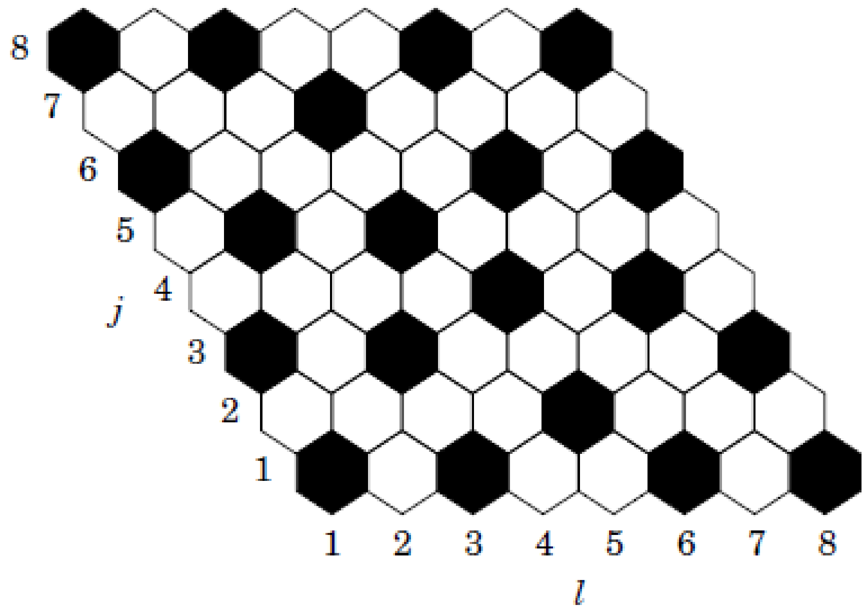
$$\frac{dD}{dt} = v \cdot \left(\frac{1}{1 + b \cdot N^h} - D \right)$$

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

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

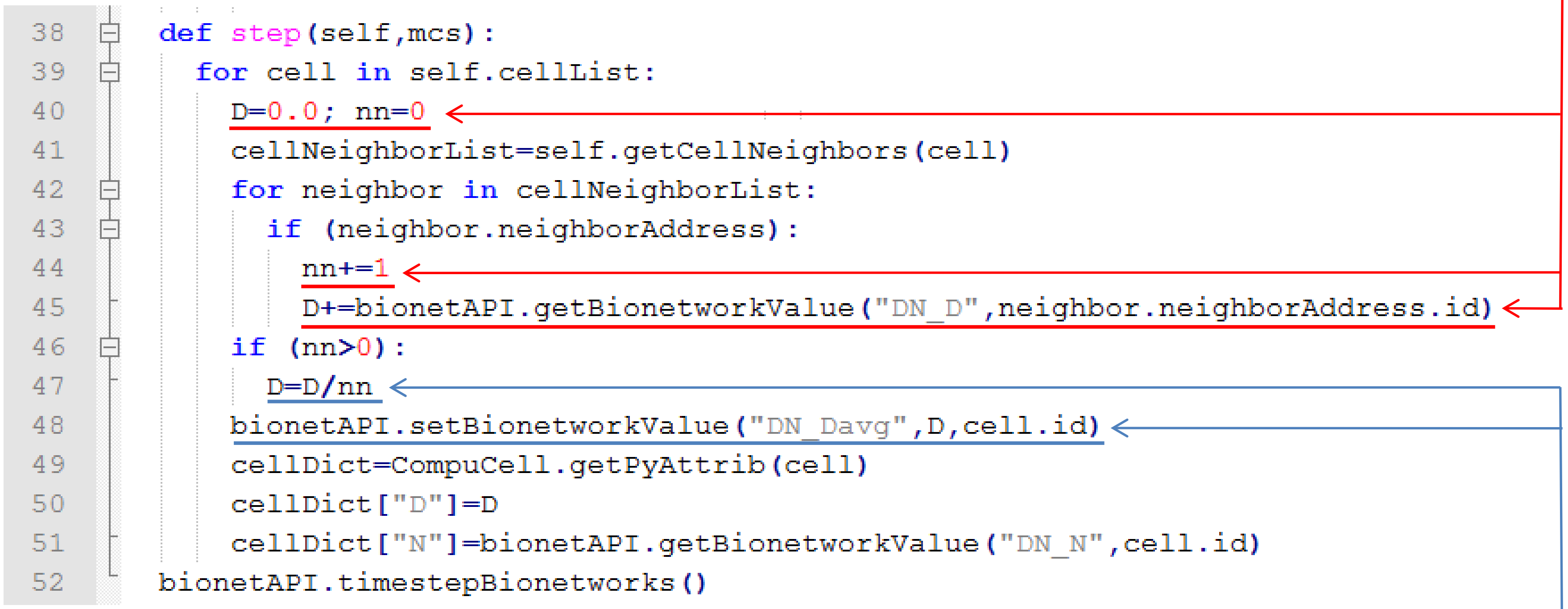
Fourth Example – Delta-Notch Patterning

- 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:



Fourth Example – Delta-Notch Patterning

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



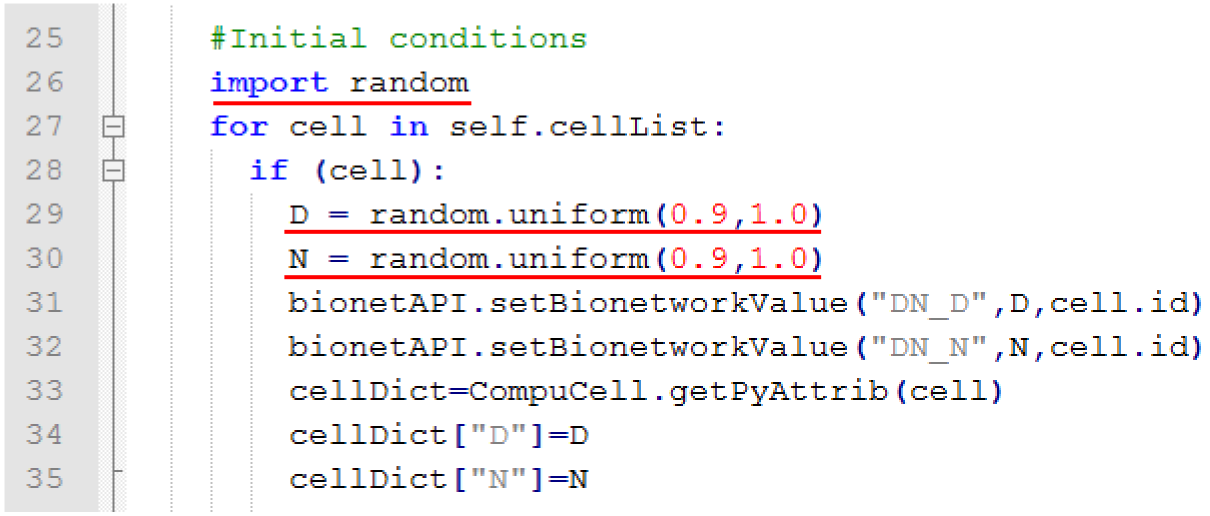
```
def step(self,mcs):
    for cell in self.cellList:
        D=0.0; nn=0
        cellNeighborList=self.getCellNeighbors(cell)
        for neighbor in cellNeighborList:
            if (neighbor.neighborAddress):
                nn+=1
                D+=bionetAPI.getBionetworkValue("DN_D",neighbor.neighborAddress.id)
        if (nn>0):
            D=D/nn
        bionetAPI.setBionetworkValue("DN_Davg",D,cell.id)
        cellDict=CompuCell.getPyAttrib(cell)
        cellDict["D"]=D
        cellDict["N"]=bionetAPI.getBionetworkValue("DN_N",cell.id)
bionetAPI.timestepBionetworks()
```

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

CompuCell3D\Demos\BoolChapterDemos_ComputationalMethodsInCellBiology\DeltaNotch

Fourth 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:



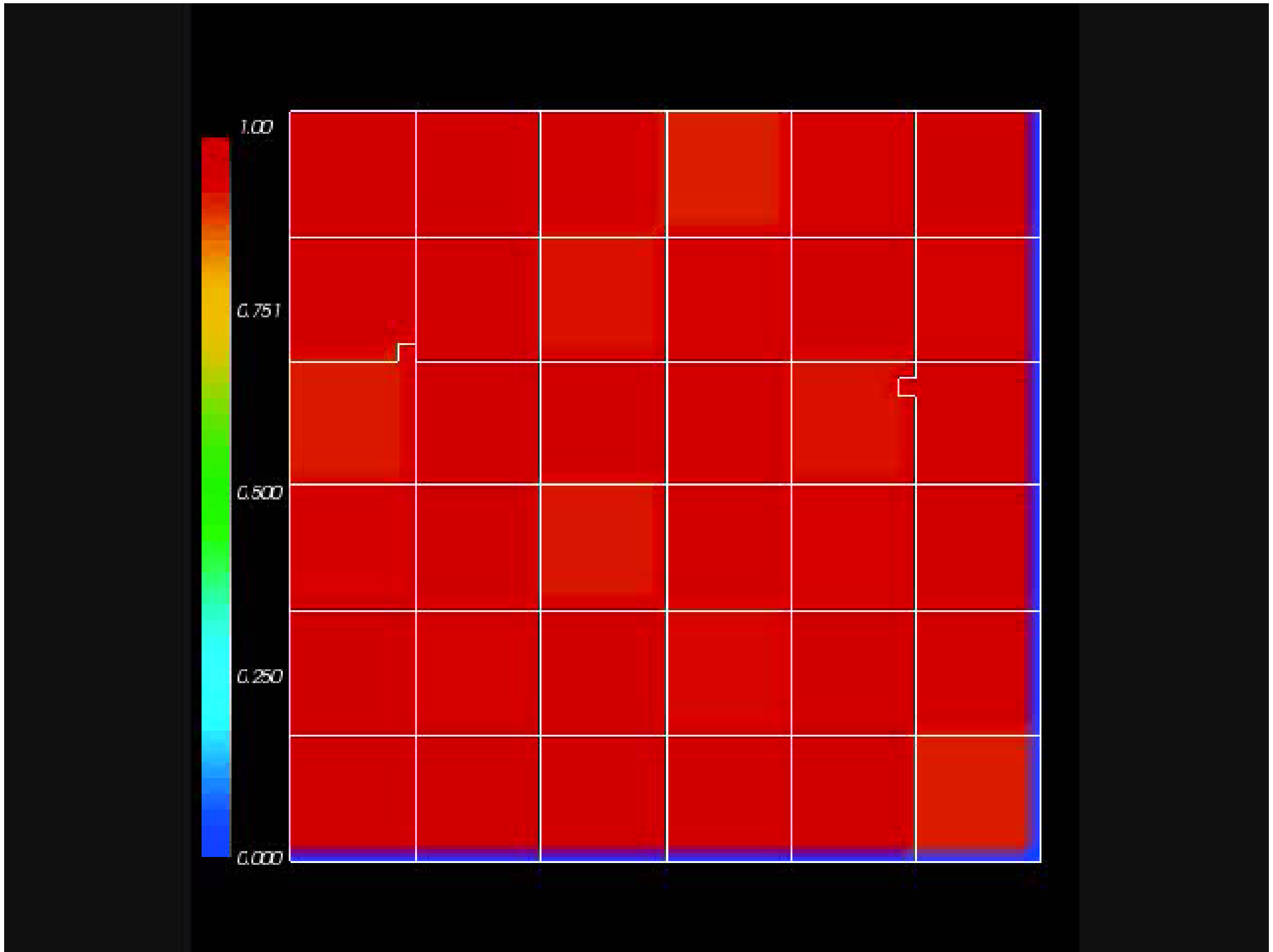
```
#Initial conditions
import random
for cell in self.cellList:
  if (cell):
    D = random.uniform(0.9,1.0)
    N = random.uniform(0.9,1.0)
    bionetAPI.setBionetworkValue("DN_D",D,cell.id)
    bionetAPI.setBionetworkValue("DN_N",N,cell.id)
    cellDict=CompuCell.getPyAttrib(cell)
    cellDict["D"]=D
    cellDict["N"]=N
```

- File:

CompuCell3D\Demos\BoolChapterDemos_ComputationalMethodsInCellBiology\DeltaNotch

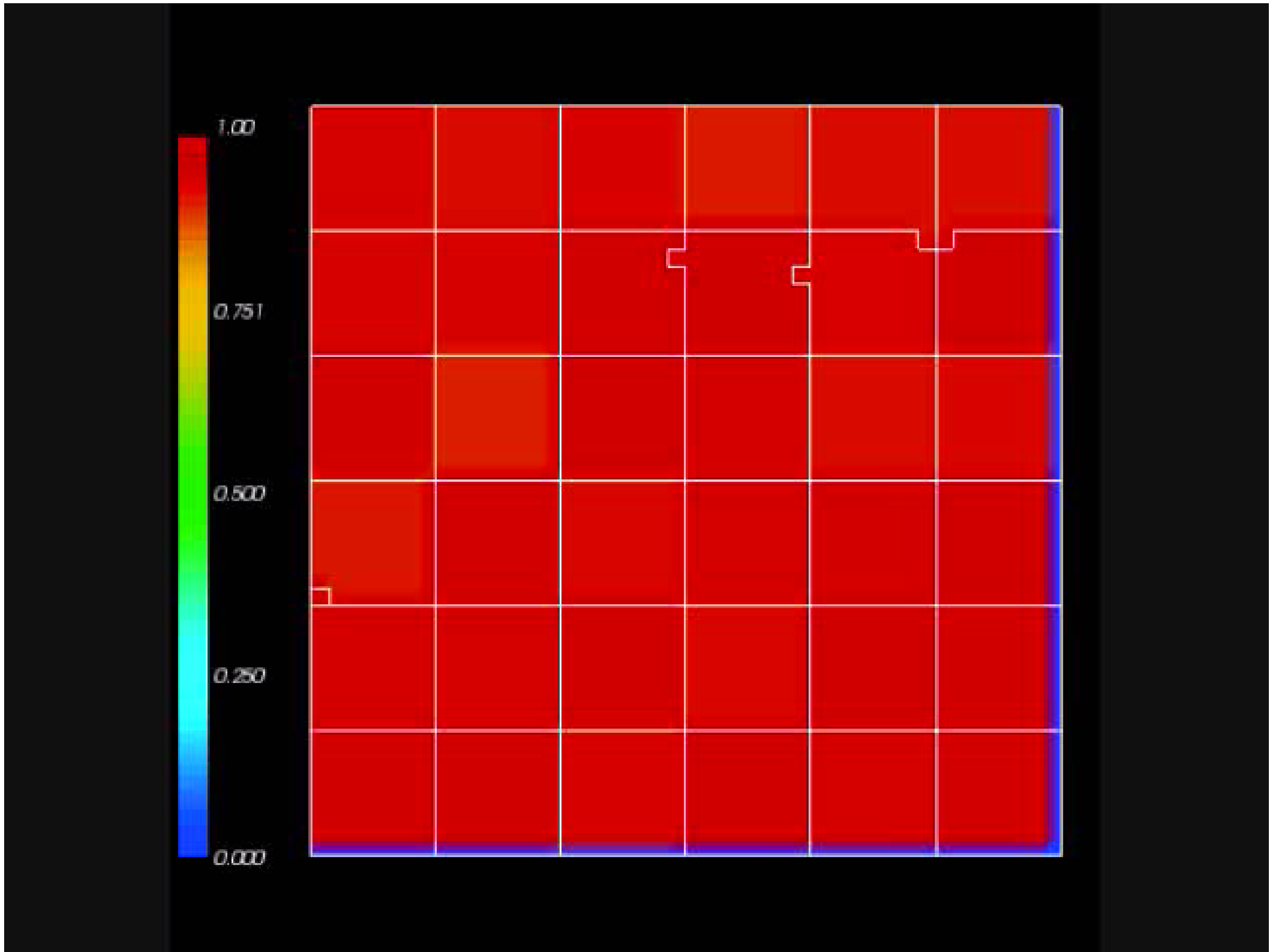
Fourth Example – Delta-Notch Patterning

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



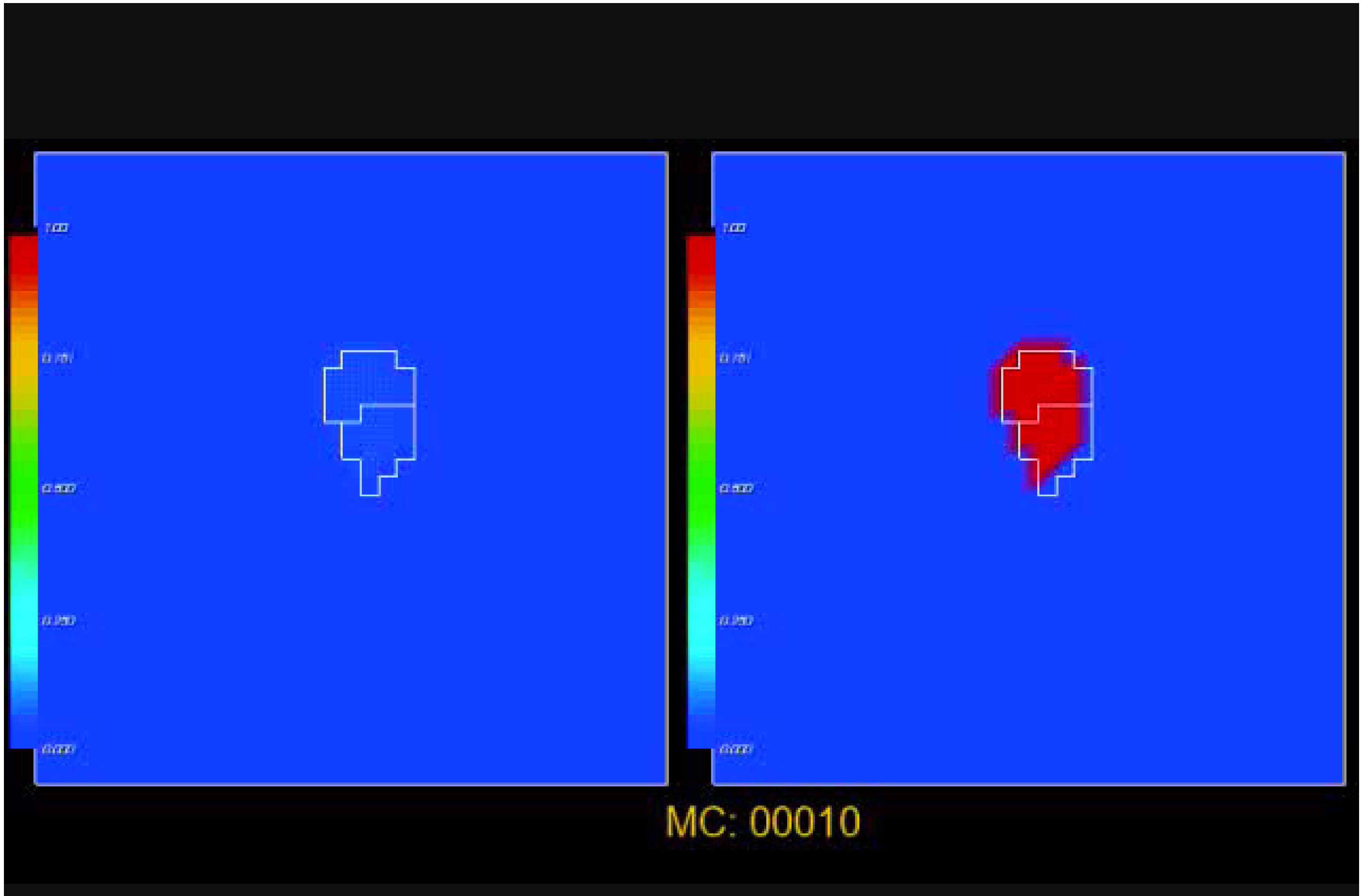
Fourth Example – Delta-Notch Patterning

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



Exercise – 2 SBML models

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


1.00
0.750
0.500
0.250
0.000
1.00
0.750
0.500
0.250
0.000
MC: 00010