

*Multi-Scale Modeling with
CompuCell3D and SBW/SBML*

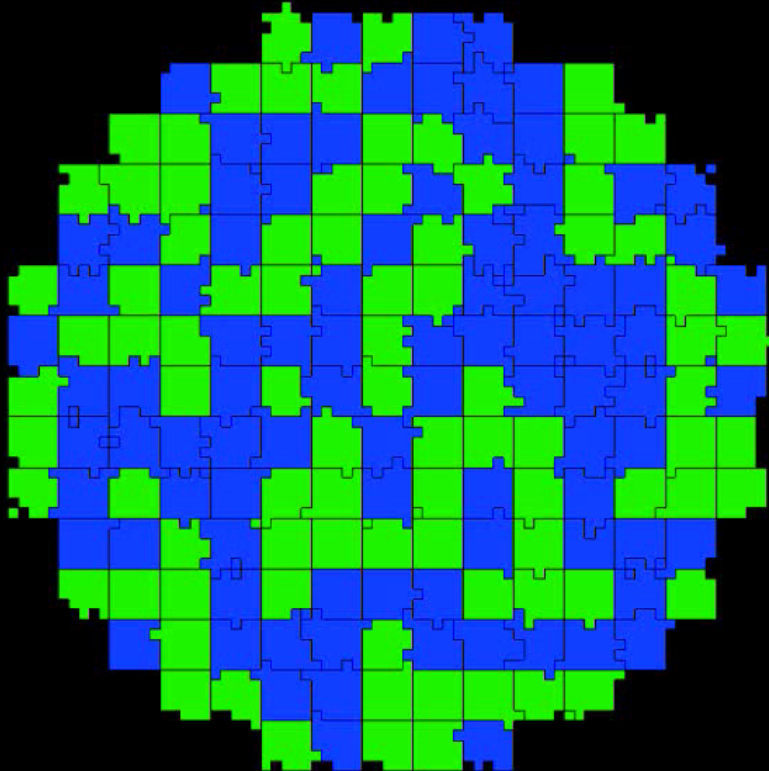
Julio M. Belmonte

Indiana University, Bloomington

Outline

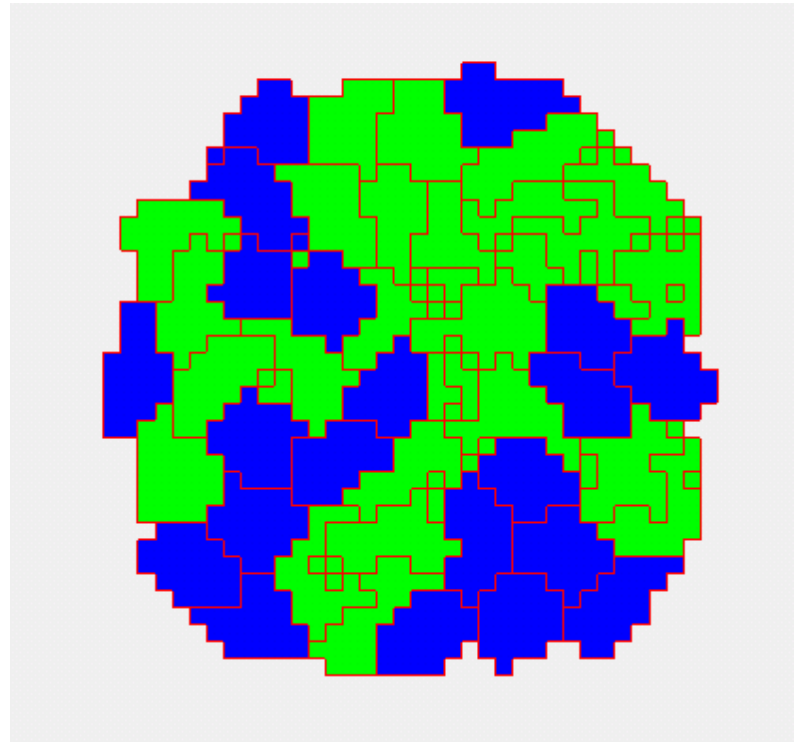
- Multi-scale extensions of the CPM
- Ways to add RK to CC3D
- SBML format
- Integrating with CC3D
 - Delta-Notch example
 - Adding Cell Cycle model from sbml.org
- Matching scales
 - Spatial scale
 - Time scale - Diffusion

Typical Cellular Potts Model Simulation

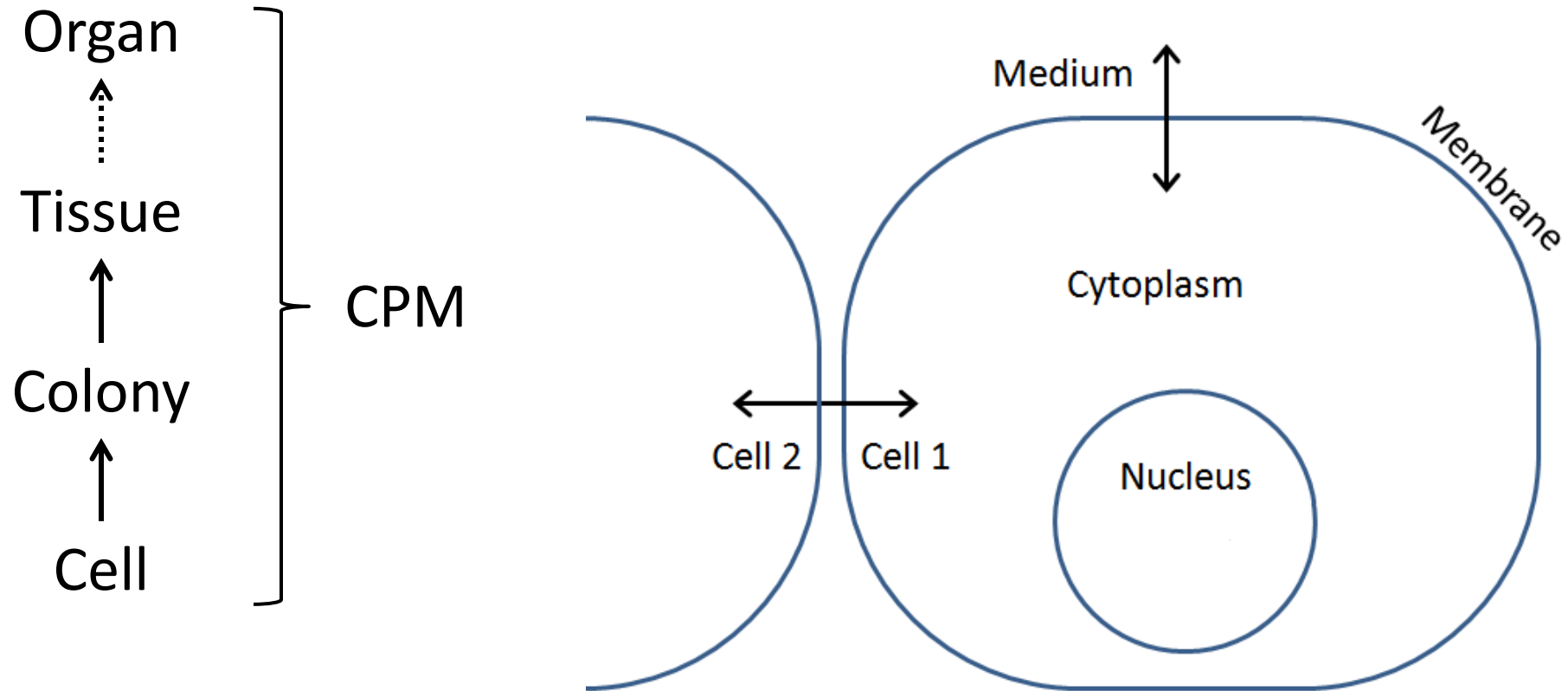


CPM modeling Scope

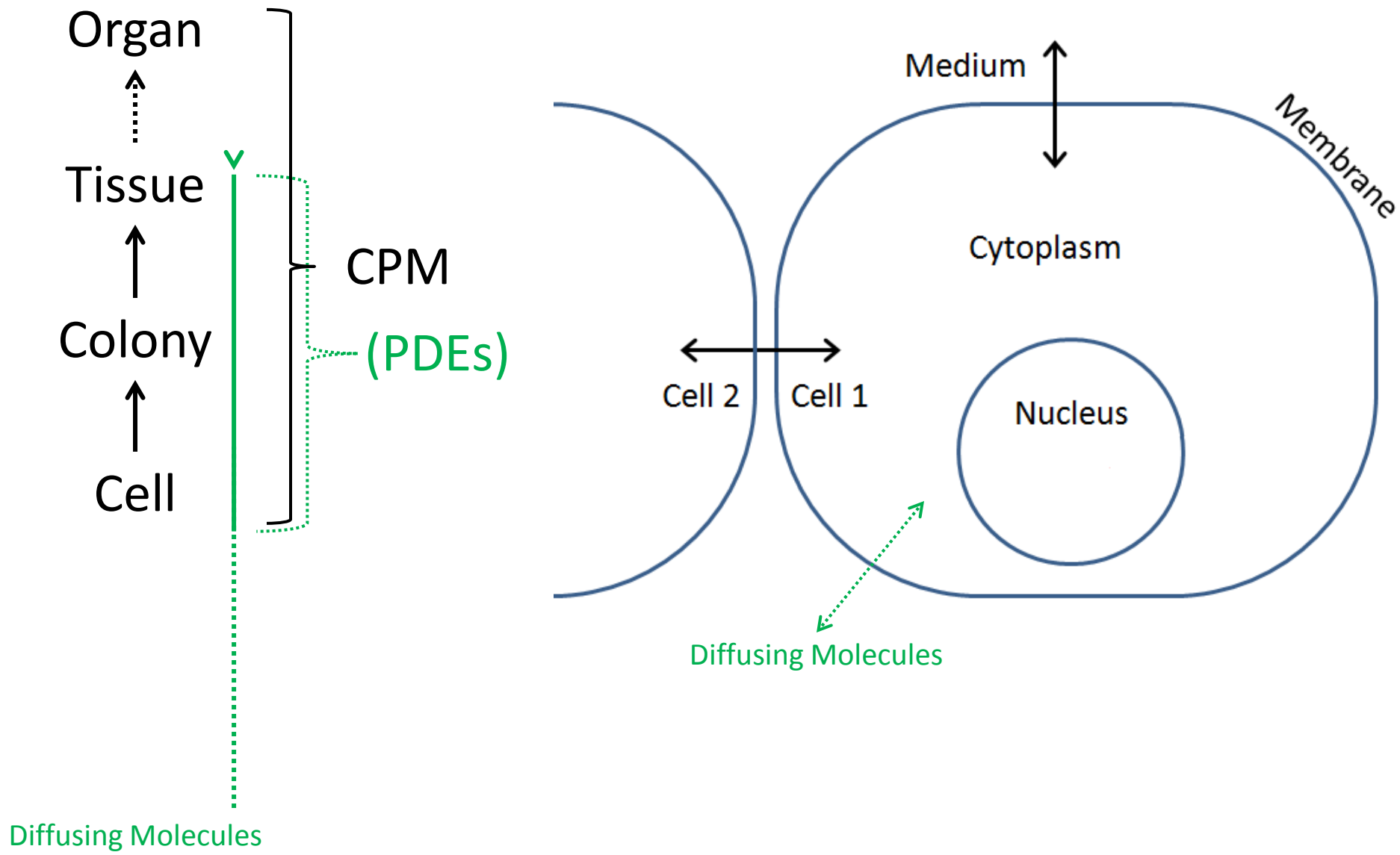
- Cellular behaviors:
 - Location
 - Volume
 - Shape
 - Movement
 - Adhesion
 - Mitosis
 - Death
 - Differentiation
 - Polarization
 - Etc...



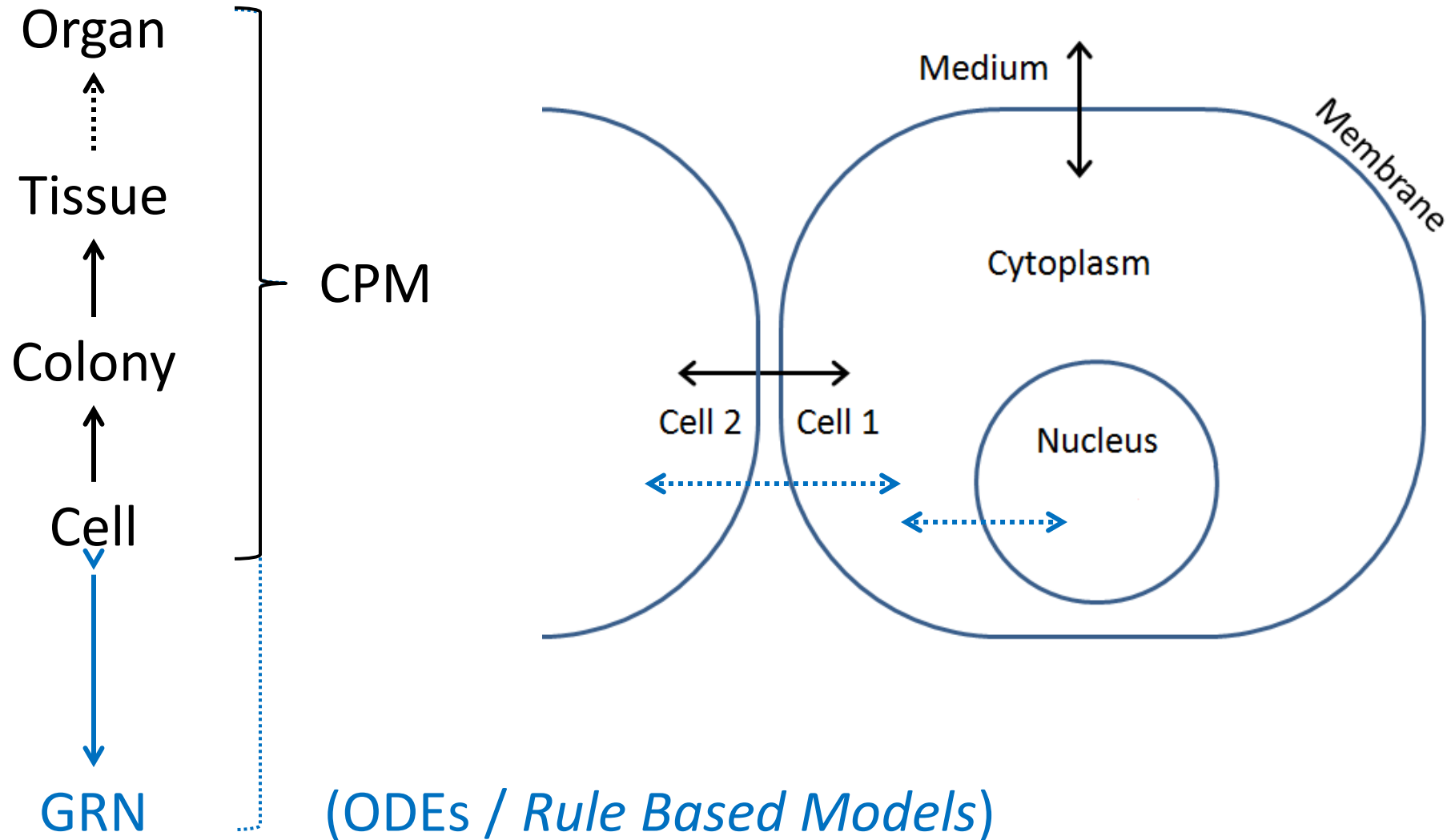
Modelling Scope – CPM



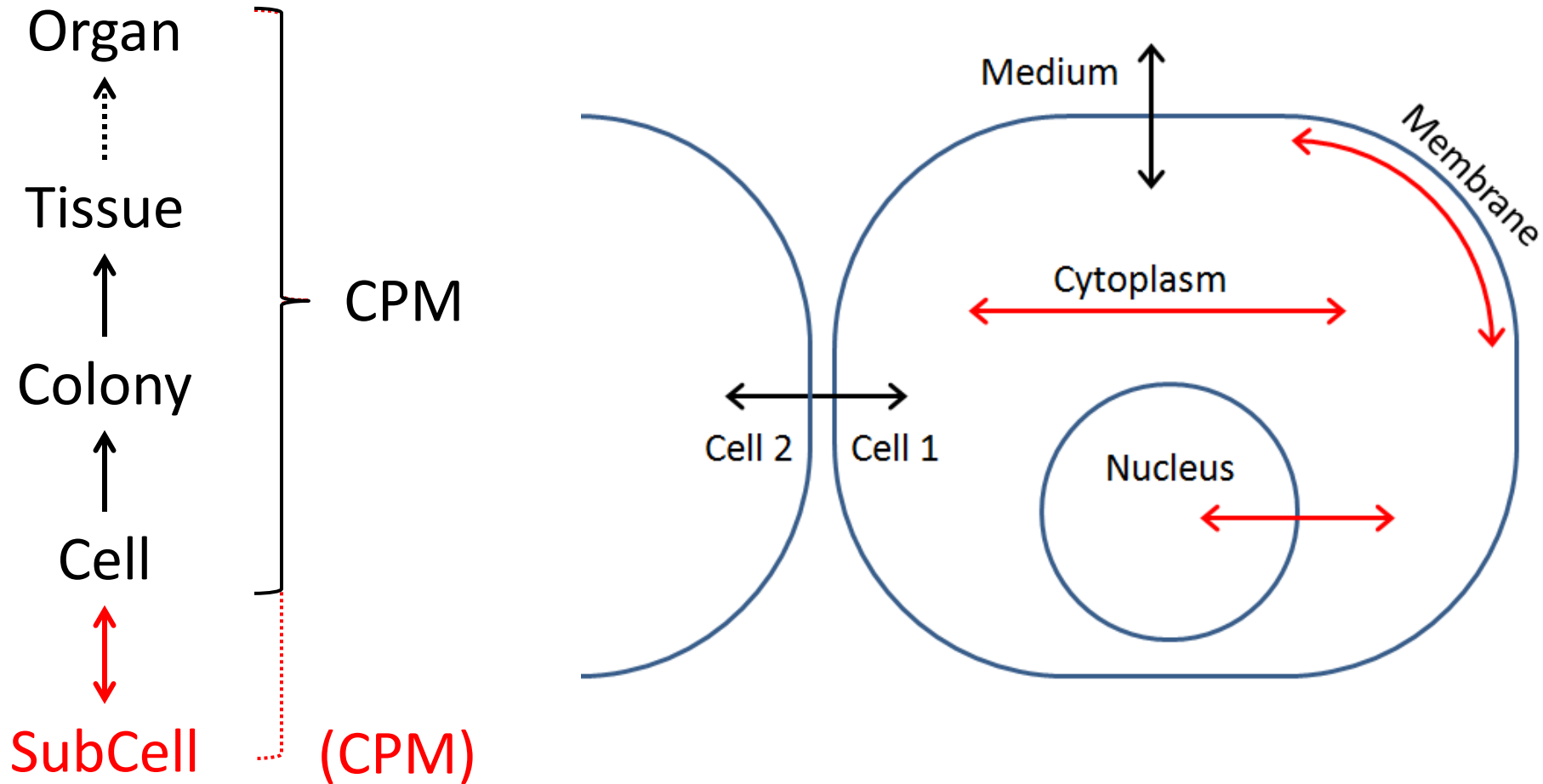
Modelling Scope – CPM + PDEs



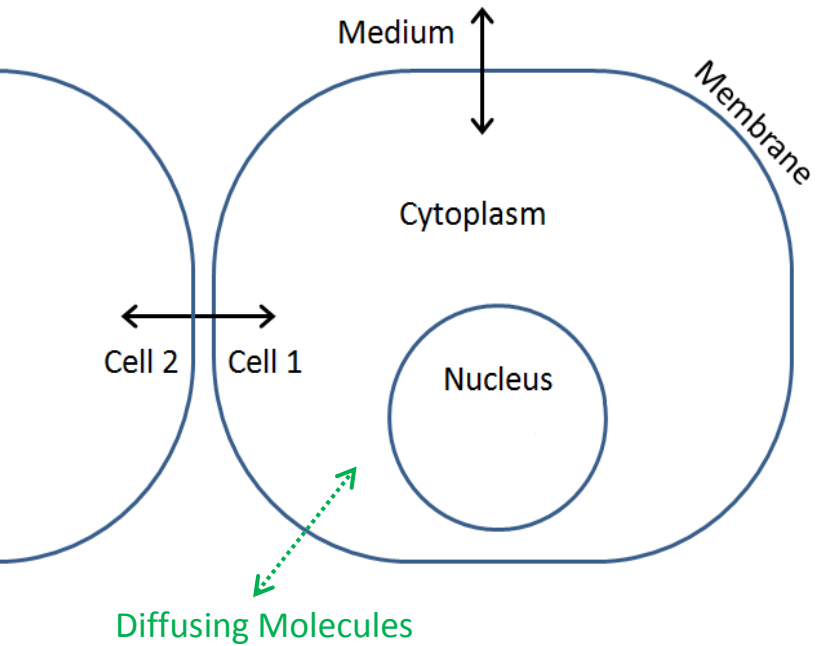
Modelling Scope – CPM + RK



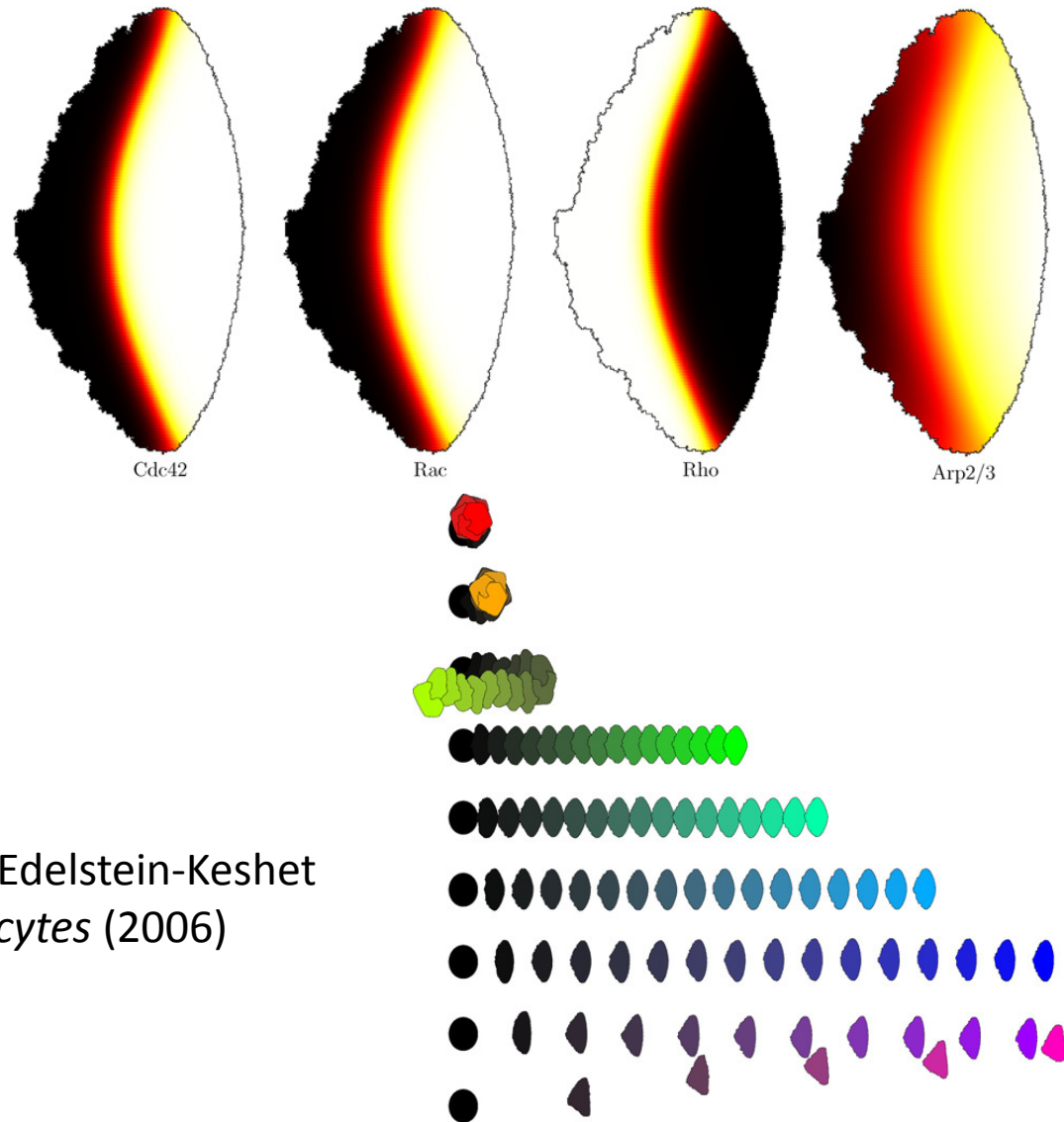
Modelling Scope – *sub*CPM



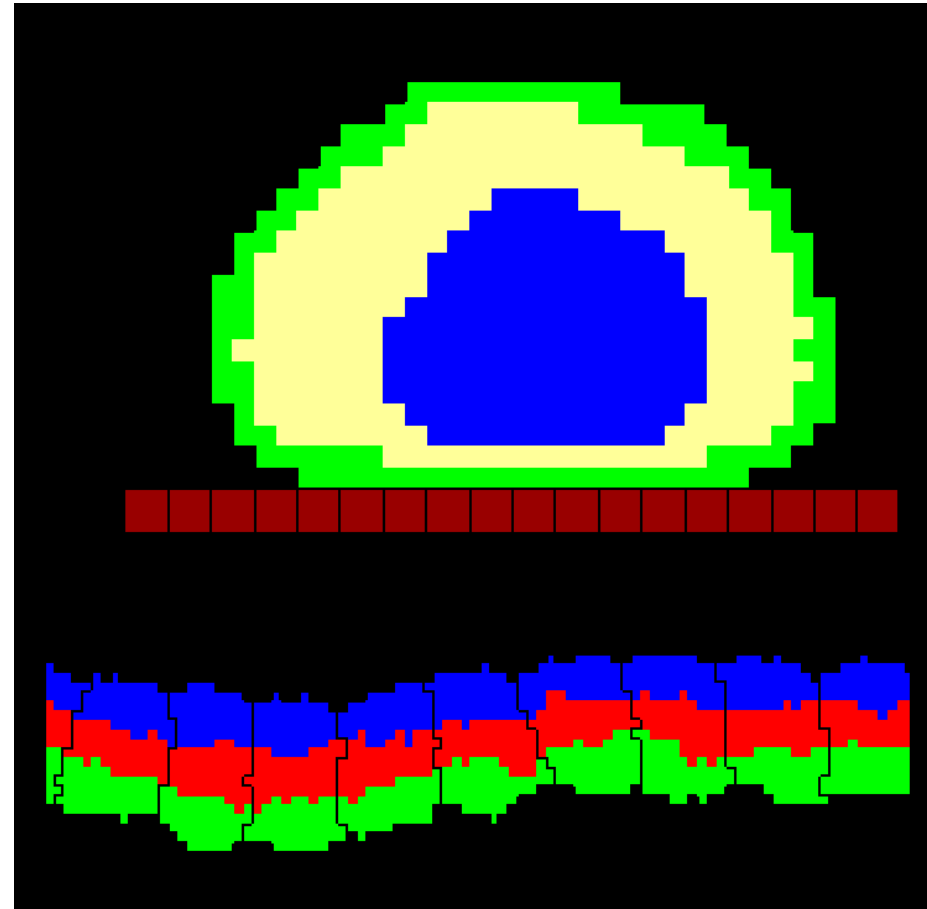
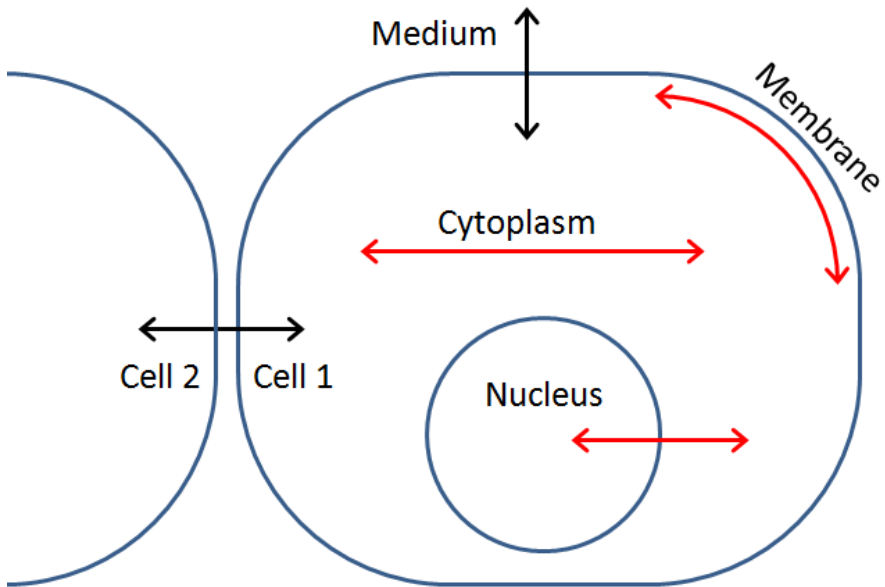
CPM + PDEs



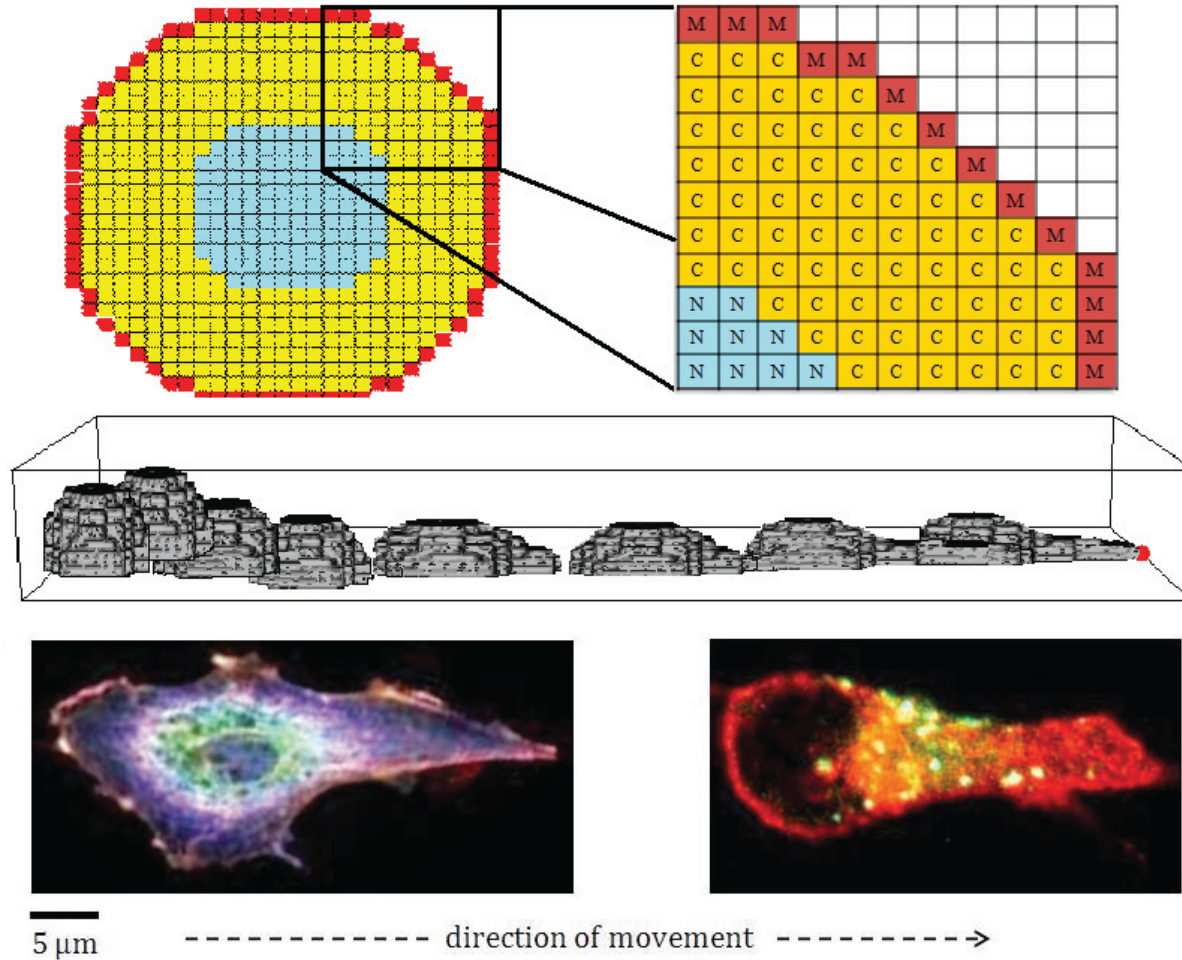
Marée, Jilkineb, Dawesb, Grieneisen, Edelstein-Keshet
Polarization and Movement of Keratocytes (2006)



Cell Compartments

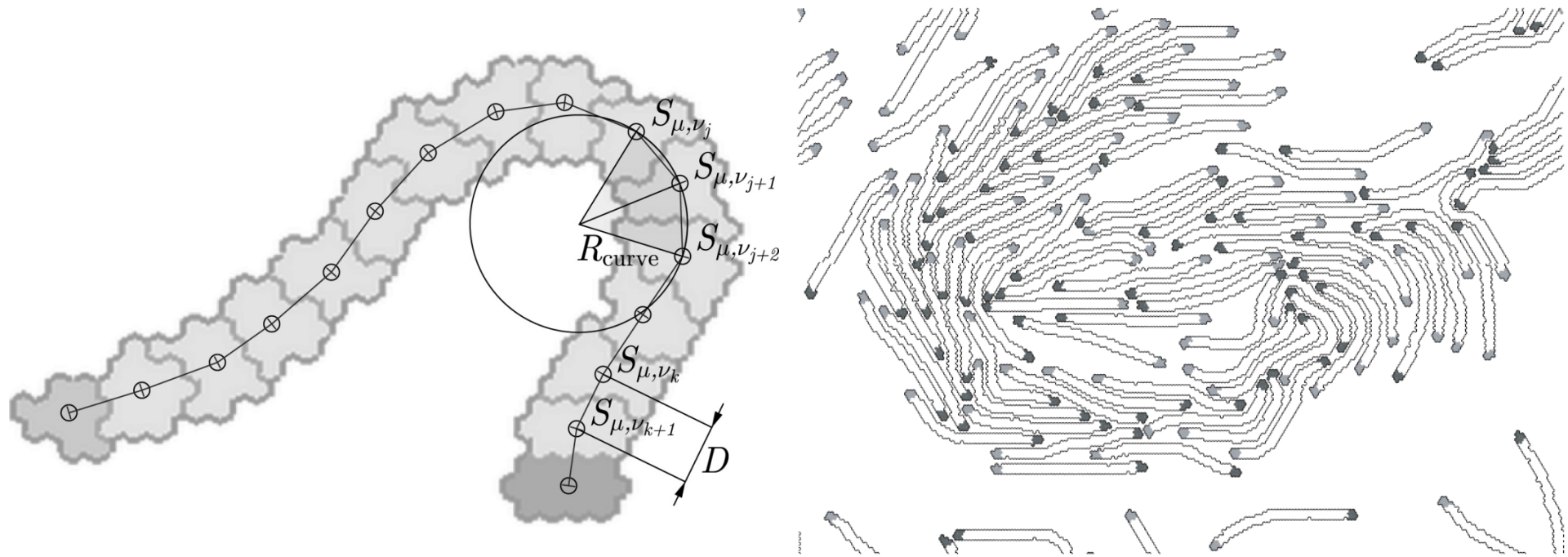


Cell Compartments



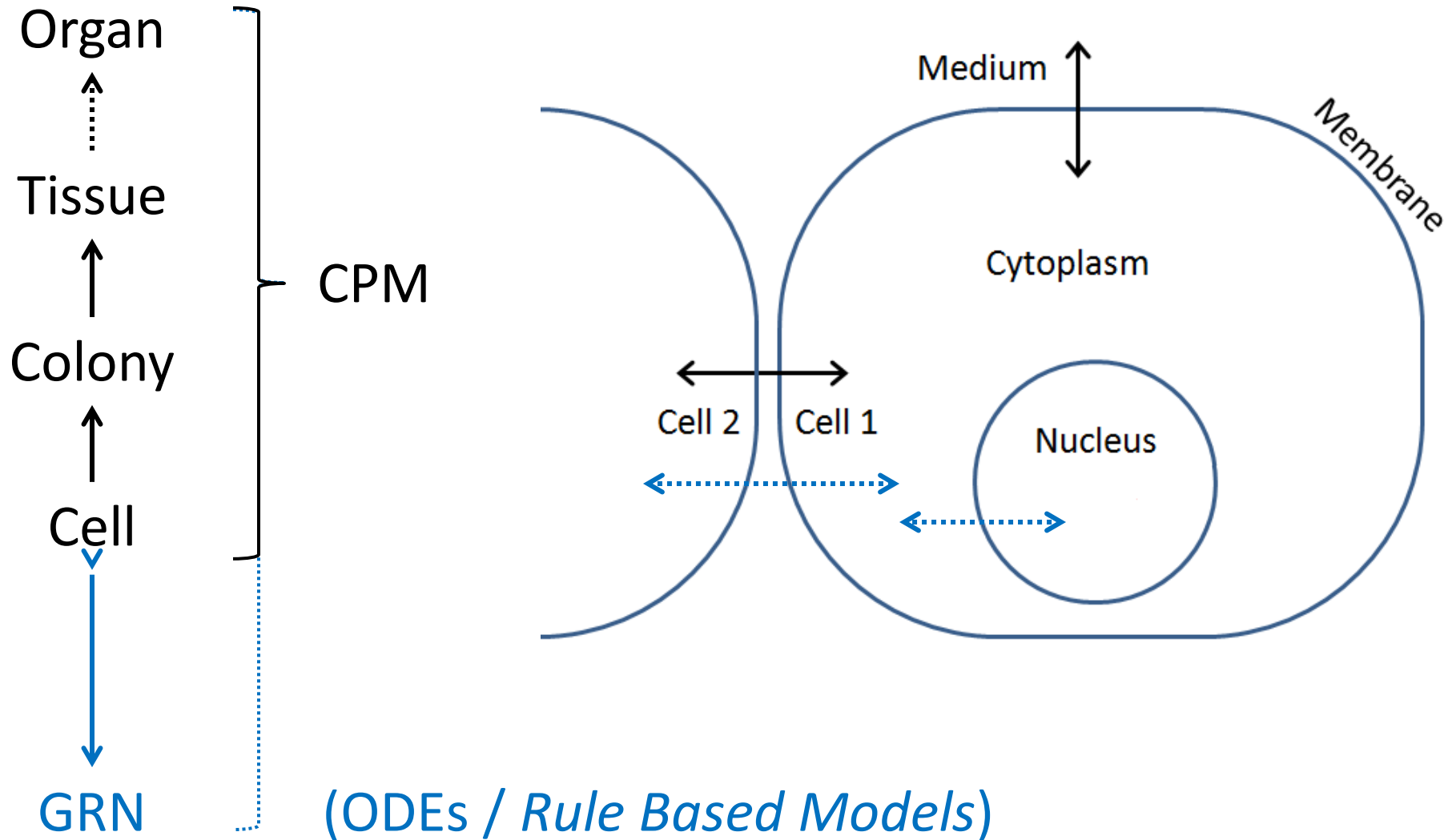
Scianna, Preziosi *Multiscale Developments of the CPM* (2012)

Cell Compartments



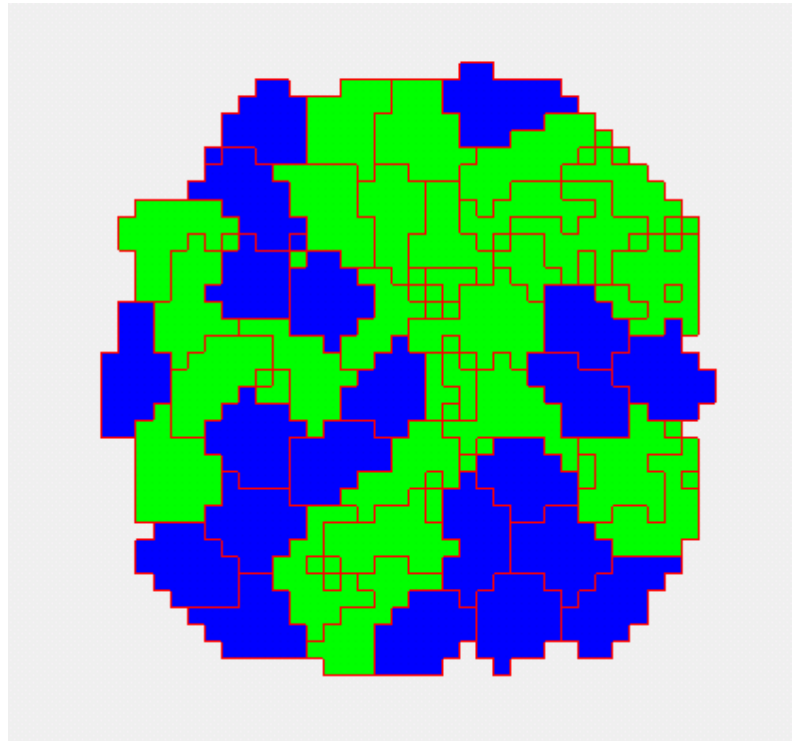
Starruss, Peruani, Bär, Deutsch *Bacterial swarming driven by rod shape* (2007)

Modelling Scope – CPM + RK



Cell-based modeling

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

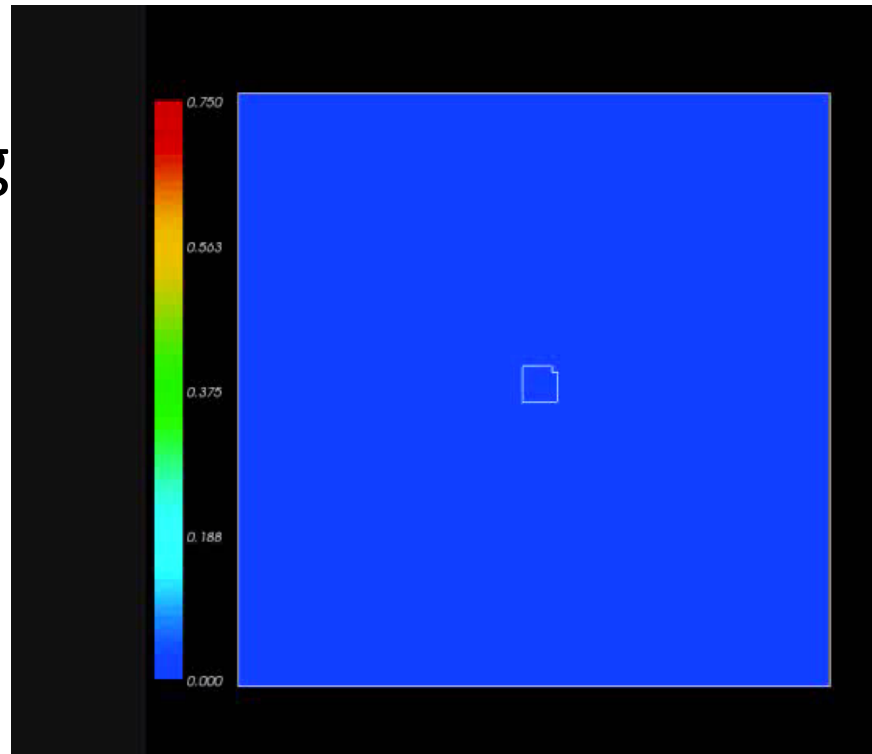
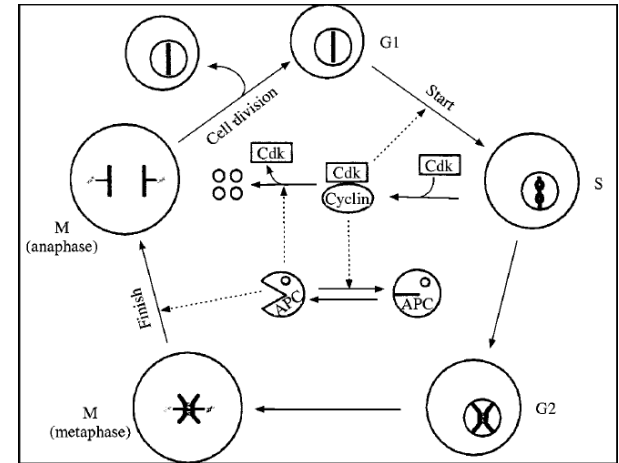


Subcellular modelling

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

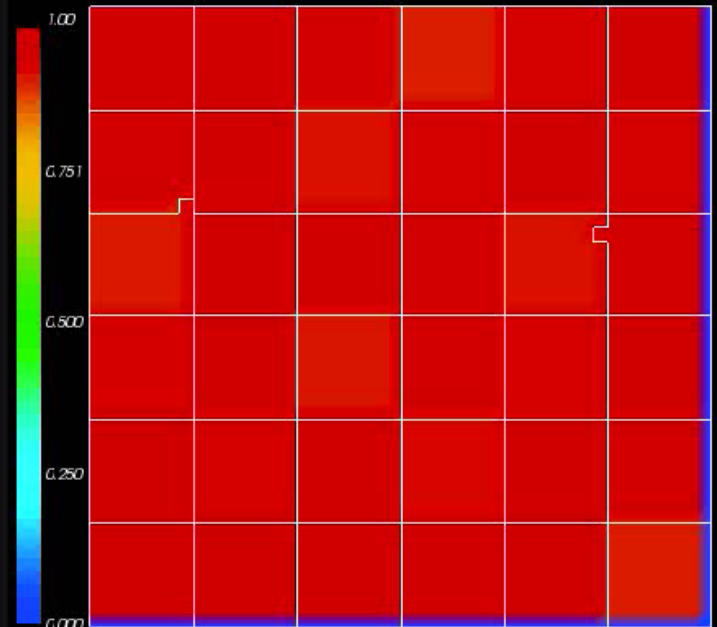
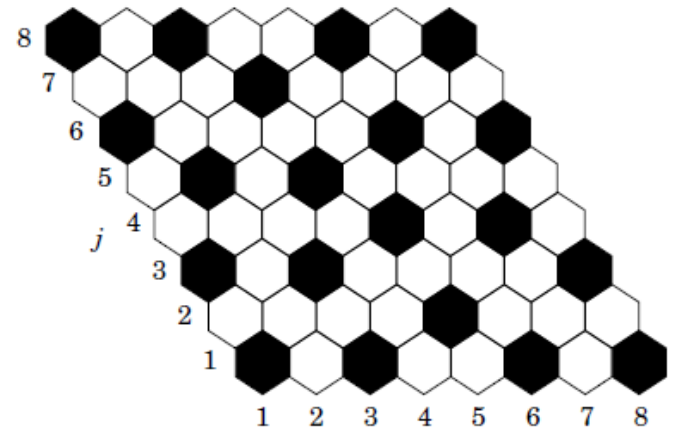
Subcellular modelling

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

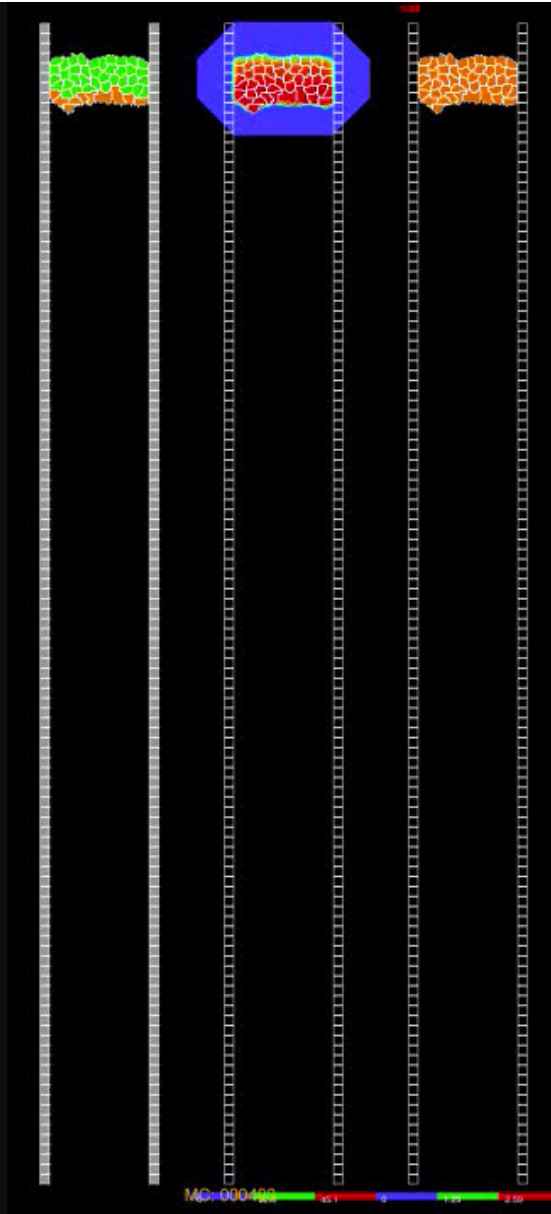


Subcellular modelling

- 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?

Developer software (SBW/Jarnac)

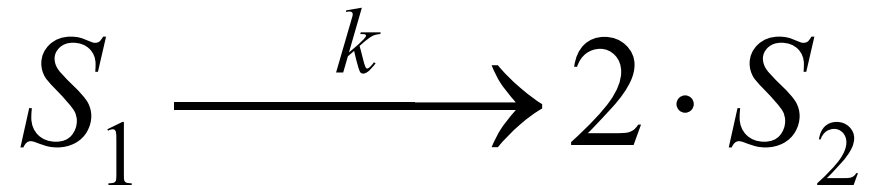


SBML



Simulation software (CompuCell3D)

SBML



- Initial conditions:

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

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

- Parameters:

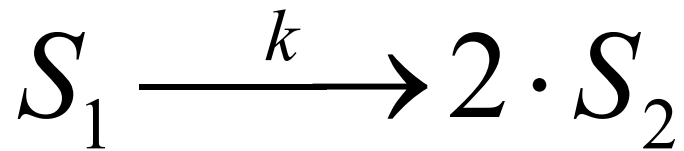
$$k = 0.1 \text{ min}^{-1}$$

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>
```

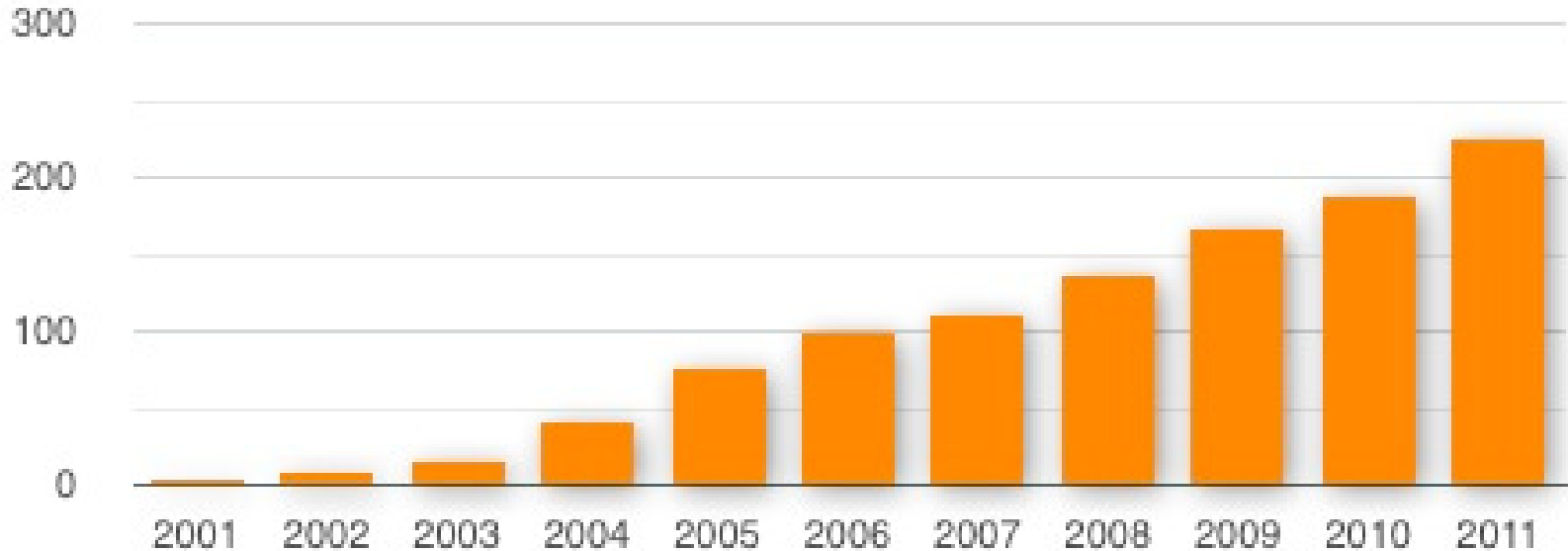
$$k = 0.1 \text{ min}^{-1}$$

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



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)

Integration with CC3D – cell

```
class <someClass>(SteppableBasePy):
    def __init__(self, _simulator, _frequency):
        SteppableBasePy.__init__(self, _simulator, _frequency)

    def start(self):
        # Define a SBML model
        Path = <ModelPath>      # Path where the model is stored
        Name = <ModelName>      # Name of the model
        Step = <timeStep>      # Time step of integration

        # Add SBML model to a cell
        for cell in self.cellList:
            self.addSBMLToCell(_modelName=Path, _modelName=Name, _cell=cell, _stepSize=Step)

    def step(self, mcs):
        # Iterate the model (run it for the time step specified on the load command)
        self.timeStepSBML()

        # Get the parameter value or molecular concentration from a cell
        for cell in self.cellList:
            <var> = self.getSBMLValue(_modelName=<Name>, _valueName=<param/molecule>, _cell=cell)

            # Set a new parameter value or molecular concentration value for a cell
            self.setSBMLValue(_modelName=<Name>, _valueName=<param/molecule>, _value=<val>, _cell=cell)

            # Change integration step of a cell
            self.setStepSizeForCell(_modelName=<Name>, _cell=cell, _stepSize=<newStep>)

        # Copy a SBML model from one cell to another
        self.copySBMLs(_fromCell=cell1, _toCell=cell2, _sbmlNames=[<ModelName1>, <ModelName2>, ...])

        # Delete a SBML model from a cell
        self.deleteSBMLFromCell(_modelName=<Name>, _cell=cell)
```

Integration with CC3D – cell id

```
class <someClass>(SteppableBasePy):
    def __init__(self, _simulator, _frequency):
        SteppableBasePy.__init__(self, _simulator, _frequency)

    def start(self):
        # Define a SBML model
        Path = <ModelPath>    # Path where the model is stored
        Name = <ModelName>    # Name of the model
        Step = <timeStep>     # Time step of integration

        # Add SBML model to a cell
        for cell in self.cellList:
            self.addSBMLToCellIds(_modelFile=Path, _modelName=Name, _ids=[cell.id], _stepSize=Step)

    def step(self, mcs):
        # Iterate the model (run it for the time step specified on the load command)
        self.timeStepSBML()

        # Get the parameter value or molecular concentration from a cell
        for cell in self.cellList:
            <var> = self.getSBMLValue(_modelName=<Name>, _valueName=<param/molecule>, _cell=cell)

            # Set a new parameter value or molecular concentration value for a cell
            self.setSBMLValue(_modelName=<Name>, _valueName=<param/molecule>, _value=<val>, _cell=cell)

            # Change integration step of a cell (overwrites the initial step value)
            self.setStepSizeForCellIds(_modelName=<Name>, _ids=[cell.id], _stepSize=<newStep>)

        # Copy a SBML model from one cell to another
        self.copySBMLs(_fromCell=cell1, _toCell=cell2, _sbmlNames=[<ModelName1>, <ModelName2>, ...])

        # Delete a SBML model from a cell
        self.deleteSBMLFromCellIds(_modelName=<Name>, _ids=[id1, id2, ...])
```

Integration with CC3D – cell types

```
class <someClass>(SteppableBasePy):
    def __init__(self, _simulator, _frequency):
        SteppableBasePy.__init__(self, _simulator, _frequency)

    def start(self):
        # Define a SBML model
        Path = <ModelPath>      # Path where the model is stored
        Name = <ModelName>      # Name of the model
        Step = <timeStep>      # Time step of integration

        # Add SBML model to a cell type (all cells of that cell type will have it)
        self.addSBMLToCellTypes(_modelFile=Path, _modelName=Name, _types=[self.TYPE1...], _stepSize=Step)

    def step(self, mcs):
        # Iterate the model (run it for the time step specified on the load command)
        self.timeStepSBML()

        # Get the parameter value or molecular concentration from a cell
        for cell in self.cellList:
            <var> = self.getSBMLValue(_modelName=<Name>, _valueName=<param/molecule>, _cell=cell)

            # Set a new parameter value or molecular concentration value for a cell
            self.setSBMLValue(_modelName=<Name>, _valueName=<param/molecule>, _value=<val>, _cell=cell)

        # Change integration step of a cell type (overwrites the initial step value)
        self.setStepSizeForCellTypes(_modelName=<Name>, _types=[self.TYPE1,...], _stepSize=<newStep>)

        # Copy a SBML model from one cell to another
        self.copySBMLs(_fromCell=cell1, _toCell=cell2, _sbmlNames=[<ModelName1>, <ModelName2>, ...])

        # Delete a SBML model from a cell type (all cells of that type will loose the model)
        self.deleteSBMLFromCellTypes(_modelName=<ModelName>, _types=[self.TYPE1,...])
```

Integration with CC3D – outside cells

```
class <someClass>(SteppableBasePy):
    def __init__(self, _simulator, _frequency):
        SteppableBasePy.__init__(self, _simulator, _frequency)

    def start(self):
        # Define a SBML model
        Path = <ModelPath>    # Path where the model is stored
        Name = <ModelName>    # Name of the model
        Step = <timeStep>    # Time step of integration

        # Add a free floating SBML
        self.addFreeFloatingSBML(_modelFile=Path, _modelName=Name, _stepSize=Step)

    def step(self, mcs):
        # Iterate the model (run it for the time step specified on the load command)
        self.timeStepSBML()

        # Get the parameter value or molecular concentration from a free floating SBML
        <var> = self.getSBMLValue(_modelName=<Name>, _valueName=<param/molecule>)

        # Set a new parameter value or molecular concentration value for a free floating SBML
        self.setSBMLValue(_modelName=<Name>, _valueName=<param/molecule>, _value=<val>)

        # Change integration step of a free floating SBML (overwrites the initial step value)
        self.setStepSizeForFreeFloatingSBML(_modelName=<Name>, _stepSize=<newStep>)

        # Delete a SBML model from a free floating SBML
        self.deleteFreeFloatingSBML(_modelName=<ModelName>)
```

Integration with CC3D

- Other commands:

```
self.timestepSBML() # time step all models
```

```
self.timestepCellSBML() # time step all models associated with cells
```

```
self.timestepFreeFloatingSBML() # time step all free floating models
```

```
# check if a model is loaded into a cell
```

```
self.getSBMLSimulator(_modelName='', _cell=)
```

```
# check if a free floating model is defined
```

```
self.getSBMLSimulator(_modelName='')
```

```
# returns dictionary with all parameters and concentration values
```

```
state = self.getSBMLState(_modelName='', _cell=)
```

```
state = self.getSBMLState(_modelName='myModel', _cell=cell)
```

```
state >> {'S1':1.0, 'S2':0.0, 'k':0.1}
```

```
# sets new values for all parameters and concentrations defined in the dictionary
```

```
self.setSBMLState(_modelName='', _cell=, _state={})
```

```
newState = {'S1':2.0, 'S2':0.5}
```

```
self.setSBMLState(_modelName='myModel', _cell=cell, _state=newState)
```


Integration with CC3D

The screenshot shows the CC3D Python interface with a context menu open over the 'SBML Solver' option. The menu lists various actions related to SBML integration, such as deleting SBML from different contexts and setting SBML values for different simulation models.

CC3D Python File Edit Search View Language Configuration Help

CC3D C++ CC3DML CC3D Project **CC3D Python**

CC3D Project

CC3D Simulation

Adhesion Flex
Bionet Solver
Cell Attributes
Cell Constraints
Cell Manipulation
Chemical Field Manipulation
Chemotaxis
Elasticity
Extra Fields
Focal Point Plasticity
Inertia Tensor
Mitosis
Python Utilities
SBML Solver
Scientific Plots
Secretion
Simulation
Visit

environ
etcwd
environ["PYTHON_MODULE_PATH"]
environ["SWIG_LIB_INSTALL_DIR"]
z
Flip, nOrderContact, time
n
S, tV, tS
PARAMETERS:

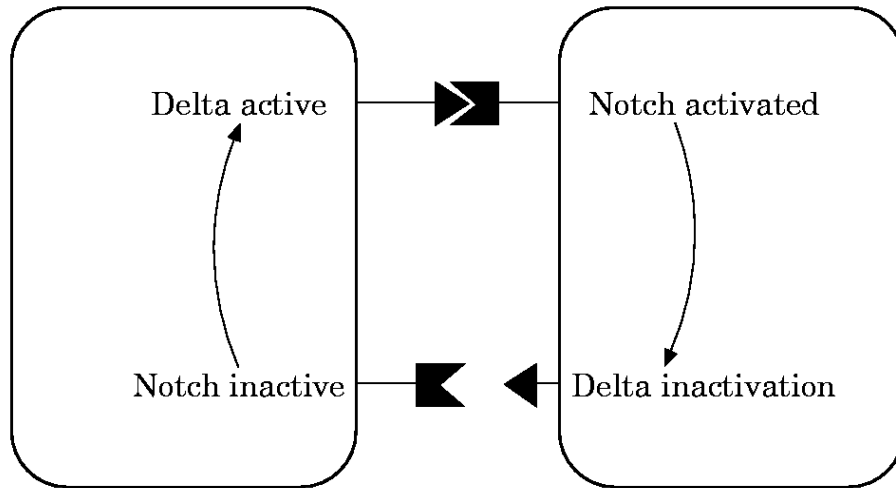
- Delete Free Floating SBML
- Delete SBML from cell ids
- Delete SBML from cell types
- Delete SBML from individual cell
- Get SBML Simulator for Free Floating Model(advanced)
- Get SBML Simulator for individual cell(advanced)
- Get SBML State for Free Floating model
- Get SBML State for individual cell
- Get SBML Value for Free Floating model
- Get SBML Value for SBML in a specific cell
- Set SBML State for Free Floating model
- Set SBML State for individual cell
- Set SBML Value for Free Floating model
- Set SBML Value for SBML in a specific cell

```
22 I=20  
23 nOrderFlip=2  
24 nOrderContact=4  
25 time=1500  
26 margin=2*cd  
27 #  
28 #SPACE PARAMETE  
29 Lx=15*cd+2*marg  
30 Ly=15*cd+2*marg  
31 Lz=1  
32  
33 def configureSi  
34 import Compu  
35 from XMLUtil  
36 cc3d=Element  
37  
38 md=cc3d.Elem  
39 md.ElementCC  
40 # md.Element  
41 md.ElementCC  
42  
43 potts=cc3d.E  
44 potts.Elemen  
45 potts.Elemen  
46 potts.ElementCC3D("Temperature", {}, T)
```

1. Model Initial Data
2. Add free floating model e.g. PBPK
2. Add model to cell ids
2. Add model to cell types
2. Add model to individual cell
3. Timestep SBML models
5. Copy All SBMLs
5. Copy select SBMLs

Example – Delta-Notch Patterning

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



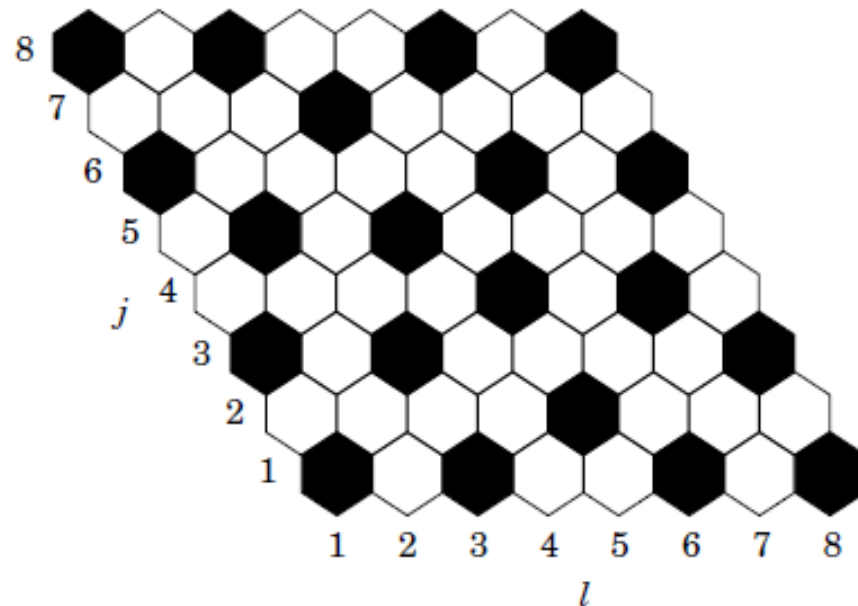
$$\frac{dD}{dt} = \nu \cdot \left(\frac{1}{1 + b \cdot N^h} - D \right)$$

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

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

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:



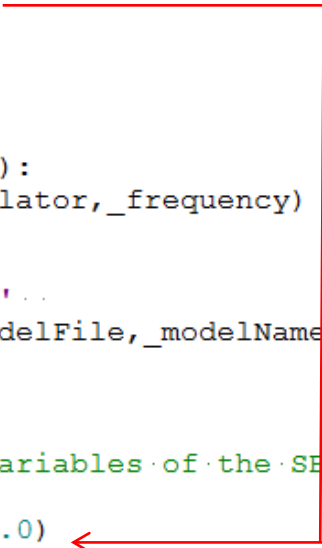
Example – Delta-Notch Patterning

- CC3D comes with this model as one of its examples:

CompuCell3D\Demos\BoolChapterDemos_ComputationalMethodsInCellBiology\DeltaNotch

- As an initial condition all cells start with random values of Delta and Notch between 0.9 and 1.0:

```
8
9 class DeltaNotchClass(SteppableBasePy):
10 def __init__(self, simulator, frequency):
11     SteppableBasePy.__init__(self, simulator, frequency)
12
13 def start(self):
14     modelFile='Simulation/DN_Collier.sbml'
15     self.addSBMLToCellTypes(_modelFile=modelFile, _modelName='DN', _types=[self.TYPES], _stepSize=0.2)
16
17     #Initial conditions
18     import random
19     state={} #dictionary to store state variables of the SBML model
20     for cell in self.cellList:
21         state['D'] = random.uniform(0.9,1.0)
22         state['N'] = random.uniform(0.9,1.0)
23         self.setSBMLState(_modelName='DN', _cell=cell, _state=state)
24
25         cellDict=self.getDictionaryAttribute(cell)
26         cellDict['D']=state['D']
27         cellDict['N']=state['N']
28
29 def stop(self):
```



Example – Delta-Notch Patterning

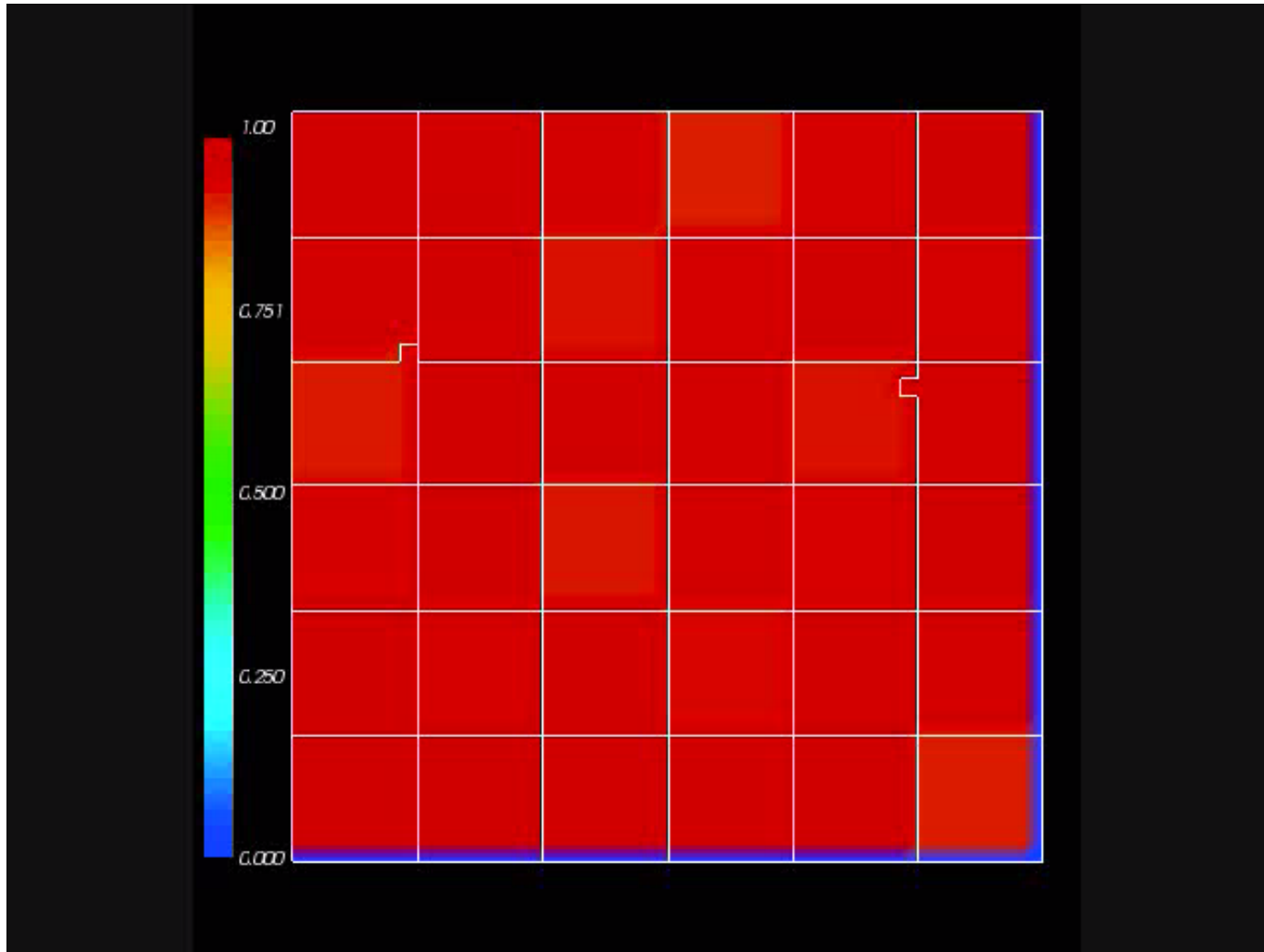
- At every MCS we loop over all cells' neighbors and store their Delta:

```
28 |
29 | def step(self, mcs):
30 |     for cell in self.cellList:
31 |         .....
32 |         D=0.0; nn=0 .....
33 |         for neighbor, commonSurfaceArea in self.getCellNeighborDataList(cell):
34 |             if neighbor:
35 |                 nn+=1
36 |                 state=self.getSBMLState(_modelName='DN', _cell=neighbor)
37 |                 D+=state['D'] .....
38 |         .....
39 |         if (nn>0):
40 |             D=D/nn
41 |             state={} .....
42 |             state['Davg']=D .....
43 |             self.setSBMLState(_modelName='DN', _cell=cell, _state=state)
44 |         .....
45 |         state=self.getSBMLState(_modelName='DN', _cell=cell)
46 |         cellDict=self.getDictionaryAttribute(cell)
47 |         cellDict['D']=D
48 |         cellDict['N']=state['N'] .....
49 |         self.timestepSBML() .....
50 |
51 |
```

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

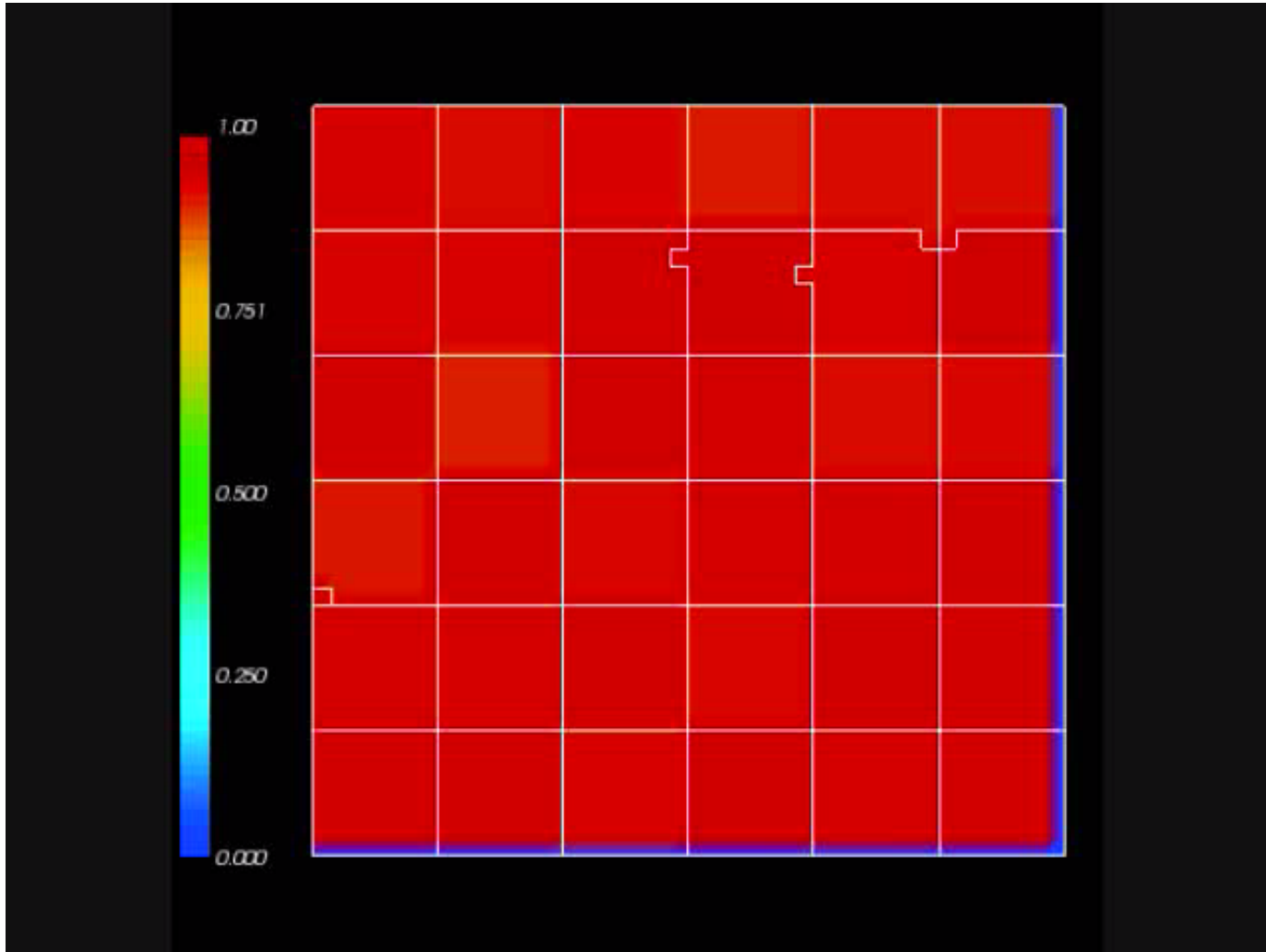
Example – Delta-Notch Patterning

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



Example – Delta-Notch Patterning

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



Exercise 1


- Open the Delta-Notch example and run it
- Make sure you understand the code
- Check that the pattern is lost when the motility of the cells is high

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*”:



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

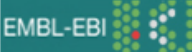
Looking for software that supports SBML? Our [software guide](#) lists over **210 systems**. Are you instead looking for models? Visit [BioModels Database](#) where you can find hundreds!



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

Second 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

Second 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
SBML L2 V2 (auto-generated)						Parameters
SBML L2 V3 (auto-generated)						Curation
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	set #1 bqbiol:occursIn Taxonomy Amphibia
Submitter:	Nicolas Le Novère	set #2 bqbiol:isVersionOf KEGG Pathway hsa04110 Gene Ontology mitotic cell cycle
Submission ID:	MODEL6614271263	bqbiol:isHomologTo Reactome REACT_152
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	

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

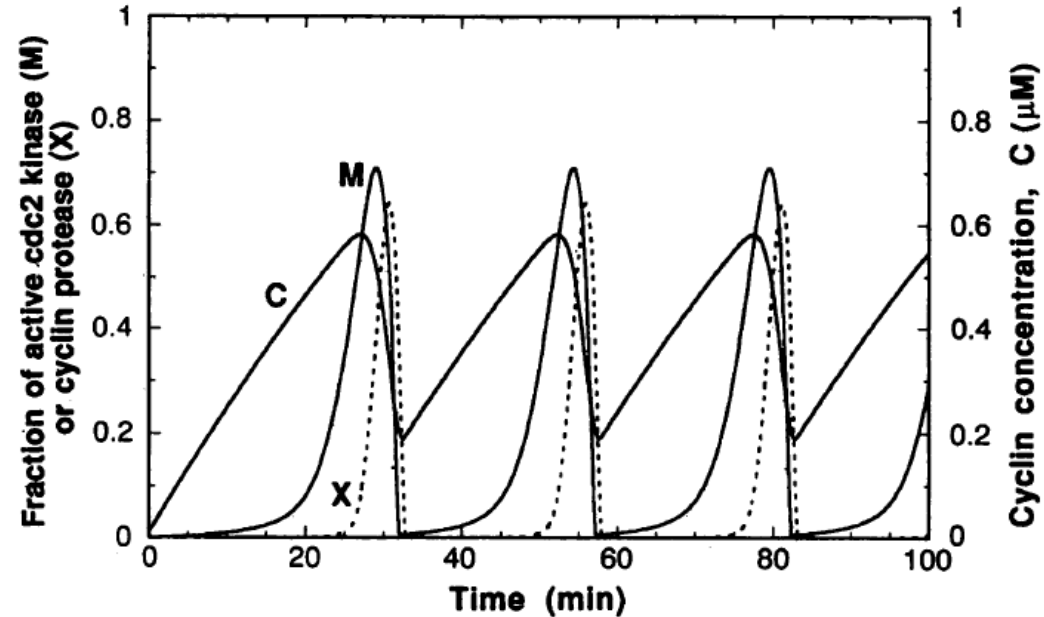
- This model is composed of 3 ODEs that forms an oscillating system:

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

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

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

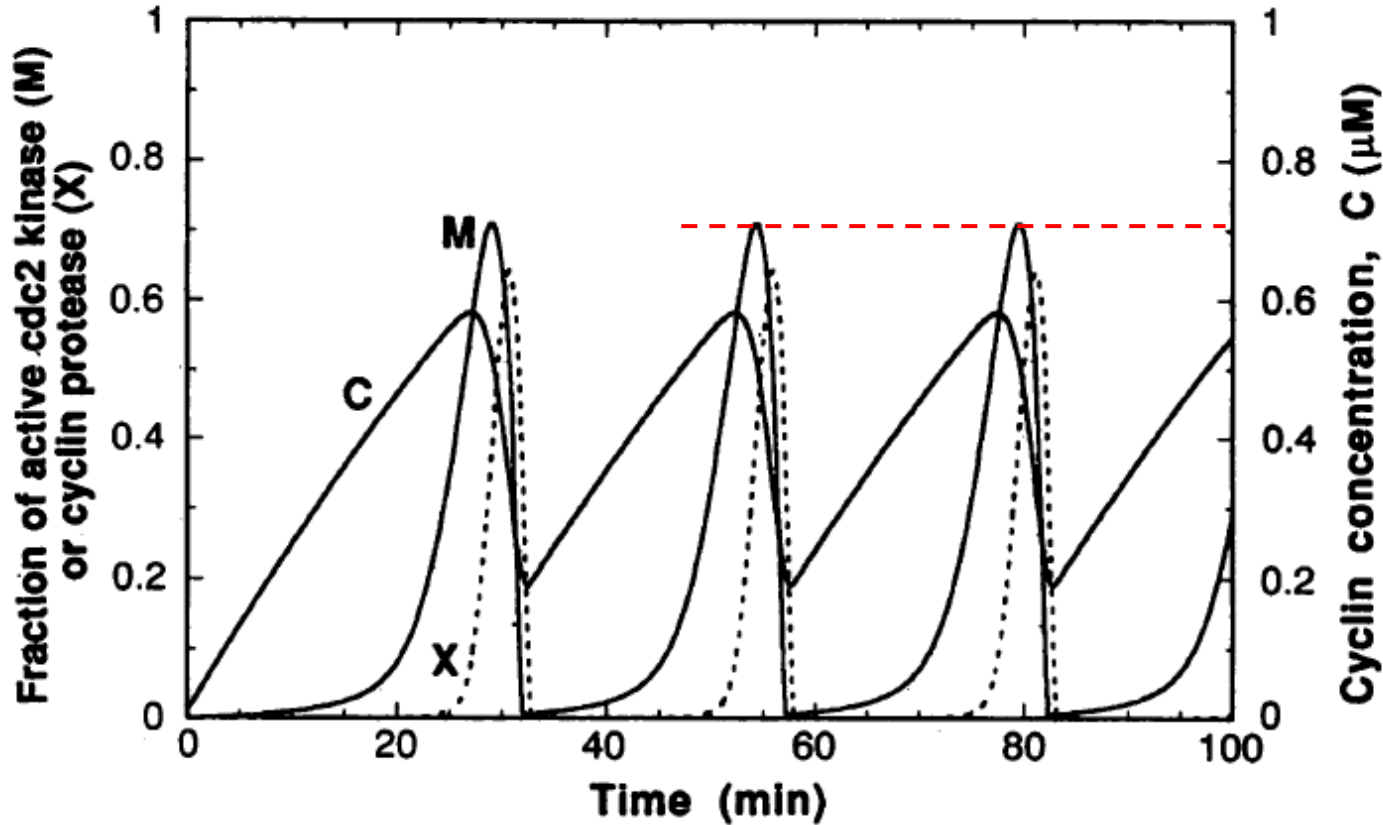
$$V_1 = \frac{C}{K_c + C} V_{M1}, \quad V_3 = M V_{M3}.$$



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

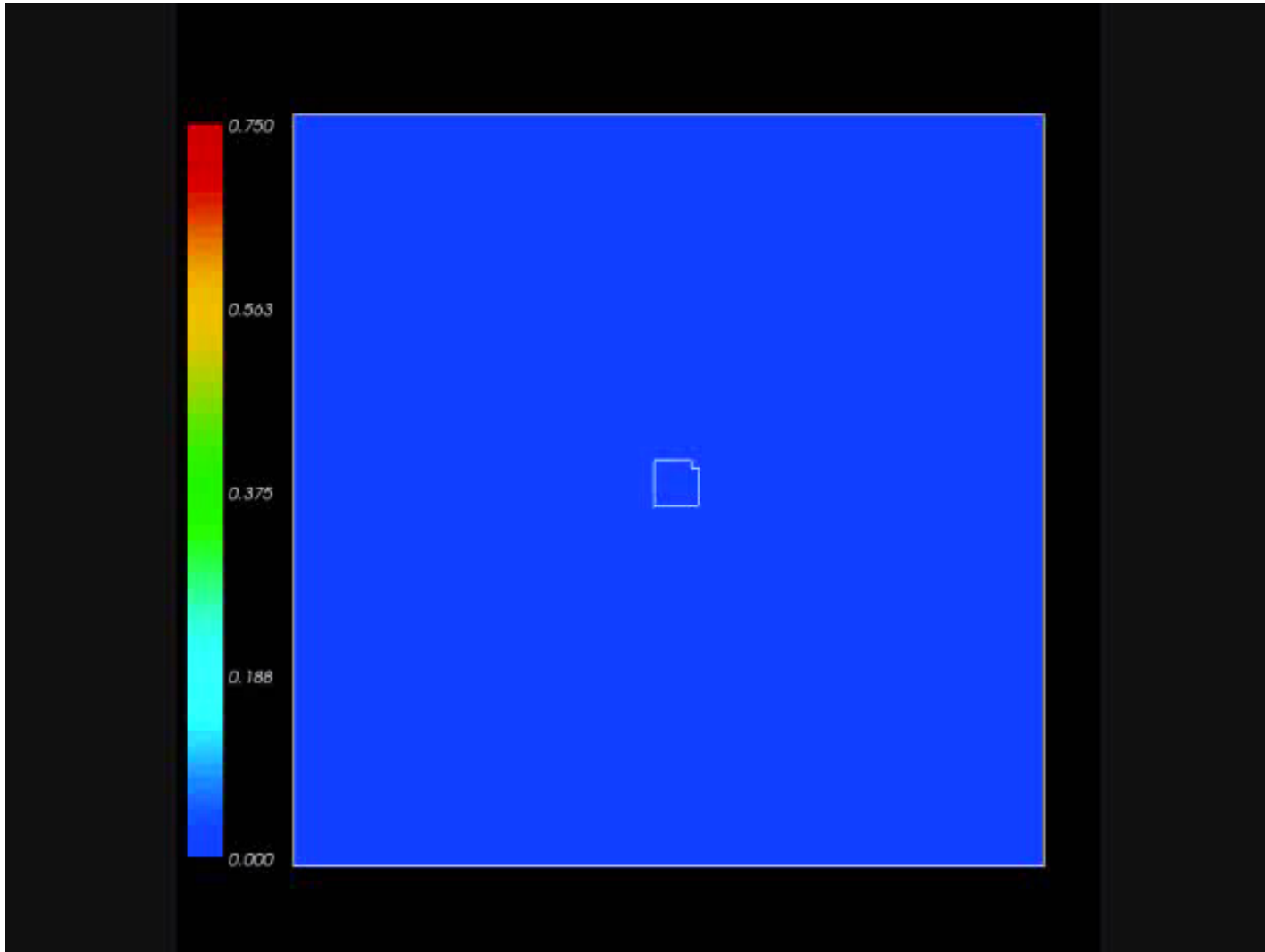
Second Example – Cell Cycle from web

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



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:

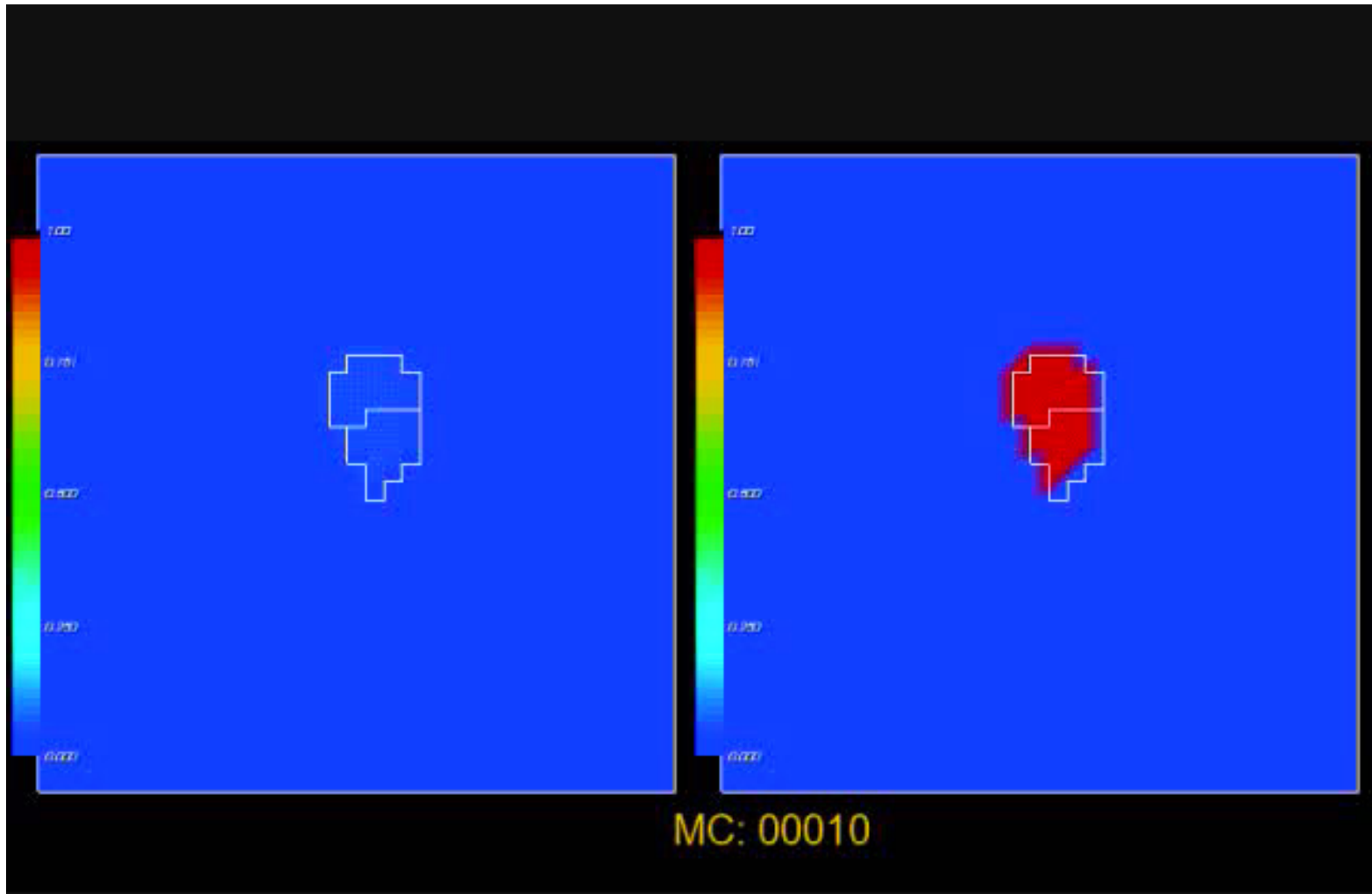


Exercise 2

- Download the SBML model from the web
- Run it on SBW
- Create a CC3D simulation with mitosis and just one cell as the initial condition
- Add the SBML model to the cell, make the integration step size to be 0.1
- Set the growth rate to zero and change the mitosis condition to be $[M] > 0.7$. Change the frequency of the mitosis steppable to 10 MCS.

2 SBML models

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

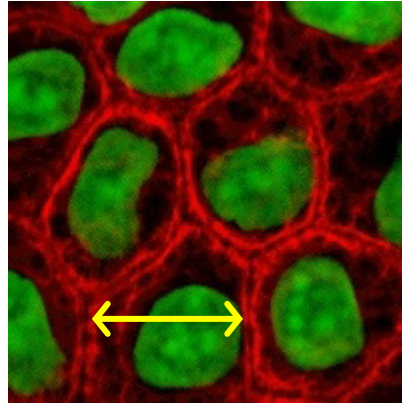
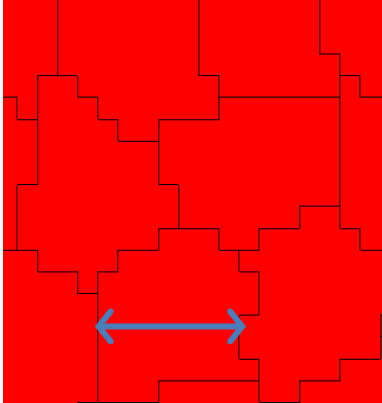


Exercise 3

- Using the codes from the Delta-Notch example and the cell cycle exercise make a simulation with the two models

Matching Scales

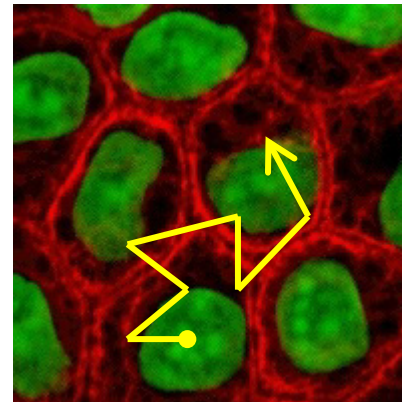
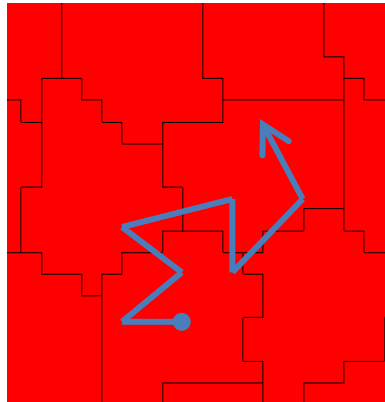
- Spatial scale:
 - Pixel-to-micrometer correspondence set by target volume of cells



7 pixel \Leftrightarrow 10 μm

1 pixel = 1.43 μm

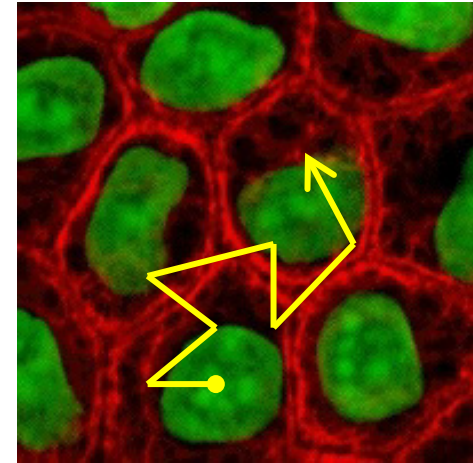
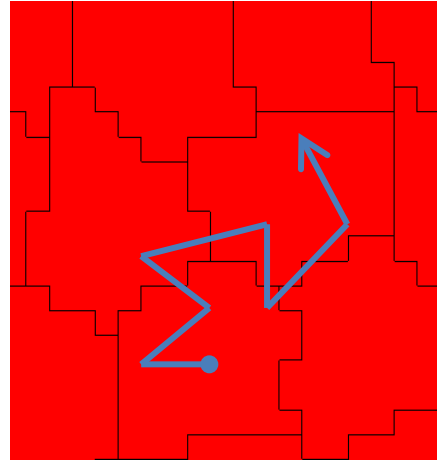
- Time scale:
 - MCS-to-second/minutes/hours set by diffusion constant of cells



Matching Scales

- Diffusion:

$$D = \frac{\langle \Delta x^2 \rangle}{2 \cdot d \cdot t}$$

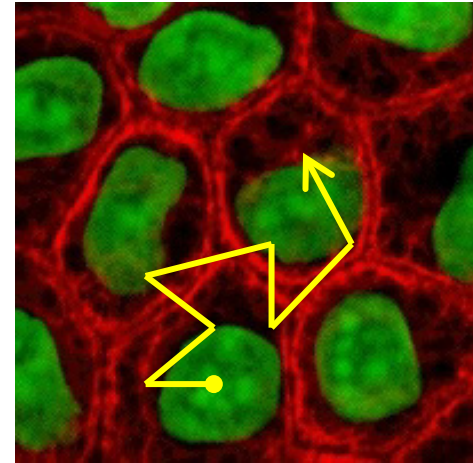
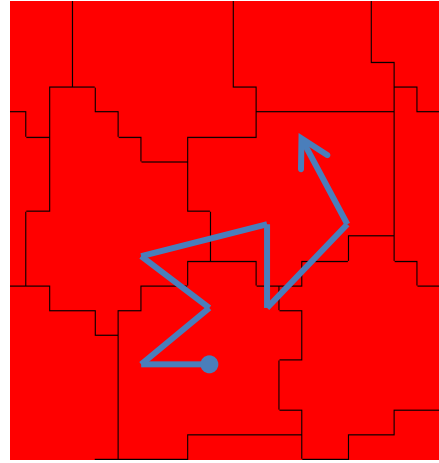


Matching Scales

- Diffusion:

$$D = \frac{\langle \Delta x^2 \rangle}{2 \cdot d \cdot t}$$

$$|\Delta x| \propto \sqrt{t}$$



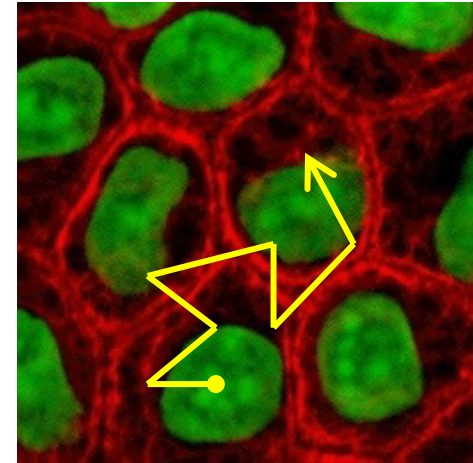
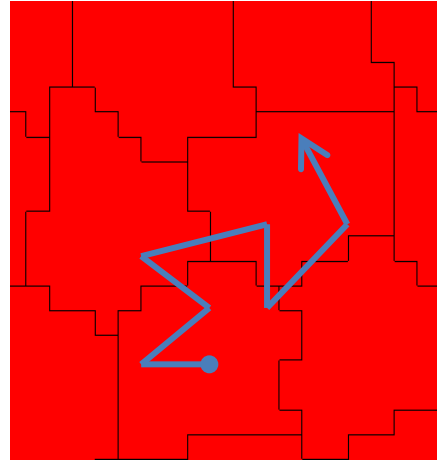
Matching Scales

- Diffusion:

$$D = \frac{\langle \Delta x^2 \rangle}{2 \cdot d \cdot t}$$

$$|\Delta x| \propto \sqrt{t}$$

Only true if cells behave like a random walk



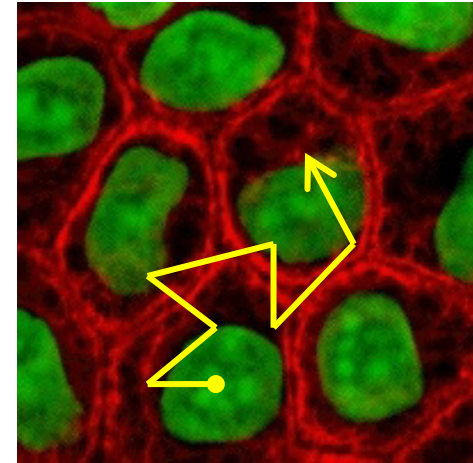
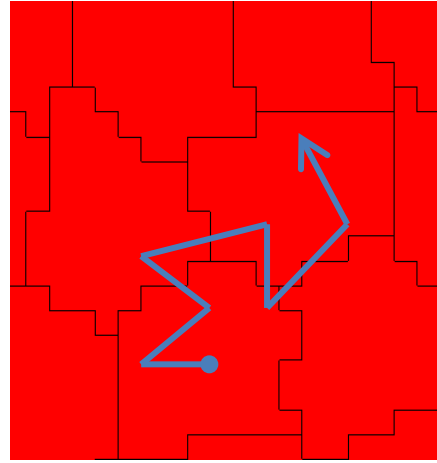
Matching Scales

- Diffusion:

$$D = \frac{\langle \Delta x^2 \rangle}{2 \cdot d \cdot t}$$

$$|\Delta x| \propto \sqrt{t}$$

Only true if cells behave like a random walk



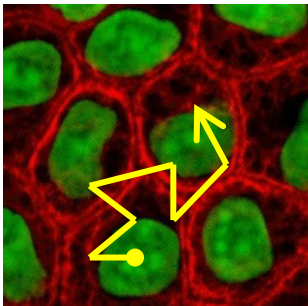
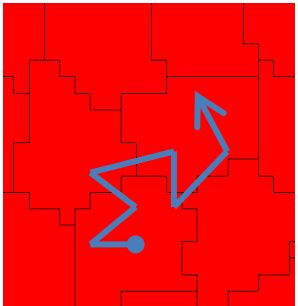
- Diffusion as a relation between displacement and time

$$\langle \Delta x^2 \rangle = f(\langle \Delta t \rangle)$$

Matching Scales

- Diffusion:

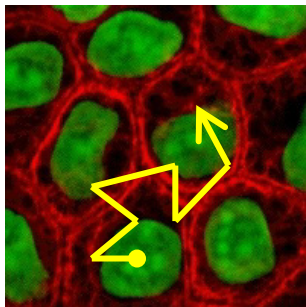
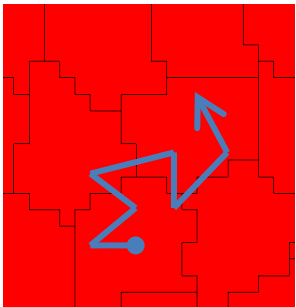
$$\langle \Delta x^2 \rangle = D \cdot \langle \Delta t \rangle^\alpha$$



Matching Scales

- Diffusion:

$$\langle \Delta x^2 \rangle = D \cdot \langle \Delta t \rangle^\alpha \begin{cases} \alpha < 1 & \textit{subdiffusion} \\ \alpha = 1 & \textit{random walk} & |\Delta x| \propto \sqrt{\Delta t} \\ 1 < \alpha < 2 & \textit{superdiffusion} \\ \alpha = 2 & \textit{ballistic} & \Delta x = D \cdot \Delta t \\ \alpha > 2 & \textit{acceleration} \end{cases}$$



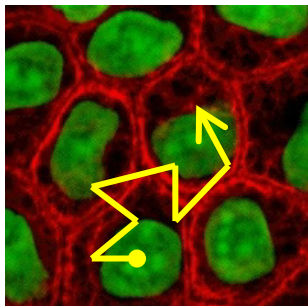
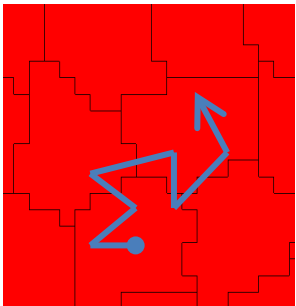
Matching Scales

- How to calculate:

$$\langle \Delta x^2 \rangle = D \cdot \langle \Delta t \rangle^\alpha$$

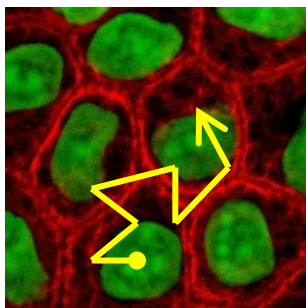
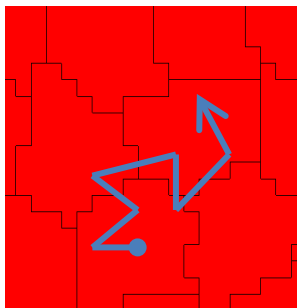
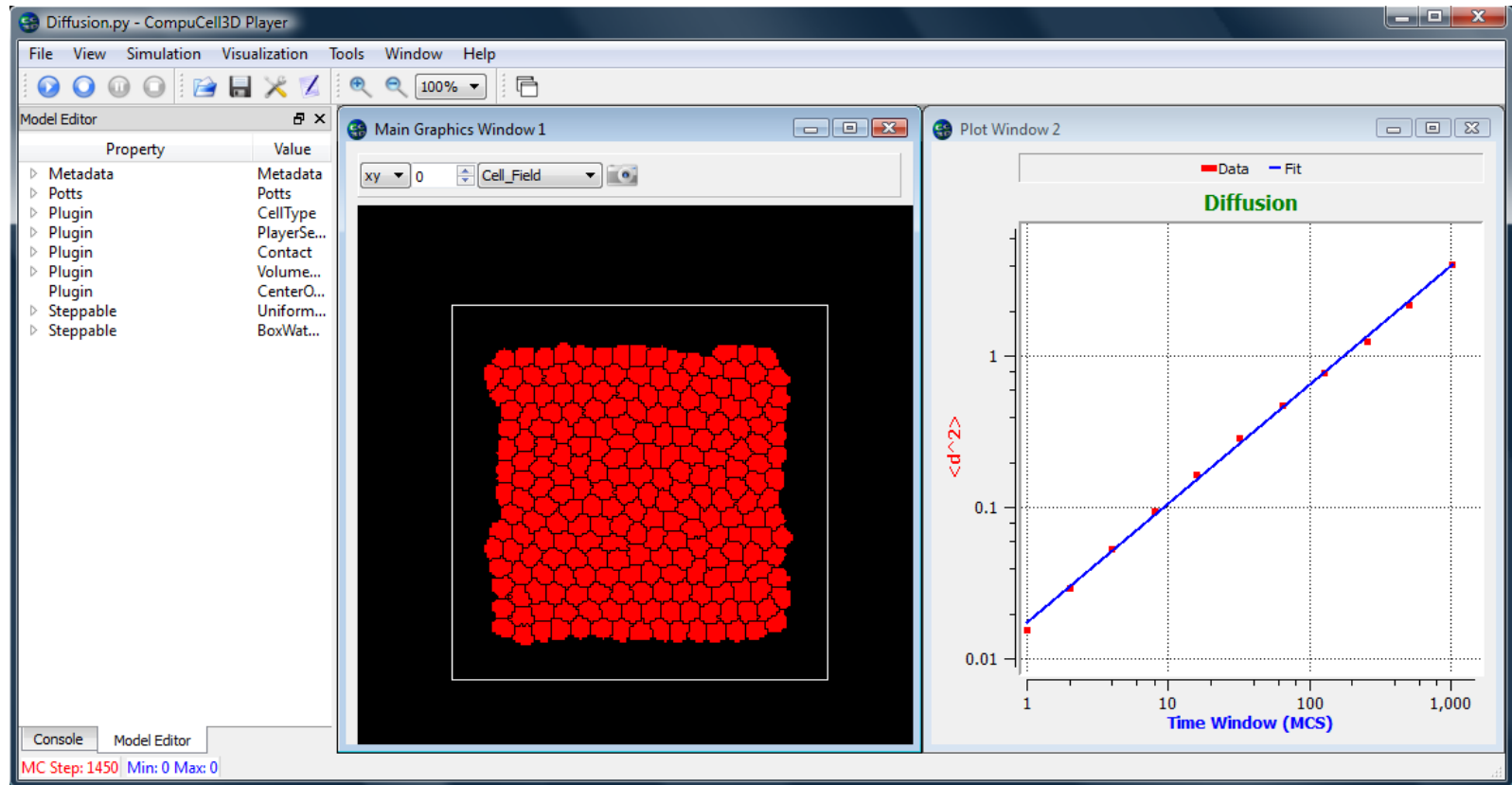
- Keep a record of all the positions of your cells (x) at each time (t).
- Change the time window (Δt) and average the mean square displacement (Δx^2) of all cells from all time points.
- Plot $\langle \Delta x^2 \rangle$ vs. $\langle \Delta t \rangle$ on a log-log scale and use a linear fit.

$$\log \langle \Delta x^2 \rangle = \log \left(D \cdot \langle \Delta t \rangle^\alpha \right) = \log D + \alpha \cdot \log \langle \Delta t \rangle$$



Matching Scales

- Diffusion calculator (CC3D-PASI webpage):



Exercise 4

- Download and run the diffusion calculator for CC3D
- Check if you understand the code
- Which type of diffusion does the simulated cells have?
- Is that what you expect?
- Change the parameters to check how does the diffusion coefficient change.

Exercises

- Exercise 1:
 - Open the Delta-Notch example, understand it and run it
 - Check that the pattern is lost when the motility of the cells is high
- Exercise 2:
 - Download the SBML model from the web and run it on SBW
 - Build a simulation with mitosis and just one cell as the initial condition
 - Add the SBML model, make the integration step size to be 0.1
 - Set the growth rate to zero and change the mitosis condition to $[M]>0.7$. Change the frequency of the mitosis steppable to 10 MCS.
- Exercise 3:
 - Using the codes from the Delta-Notch example and the cell cycle exercise make a simulation with the two models
- Exercise 4:
 - Use the diffusion calculator and check how do the cells move