# *Mathematica Platforms for Modeling in Systems Biology:*

# Recent Developments in *MathSBML* and *xCellerator*


Bruce E. Shapiro*, James Lu**, Michael Hucka*, Eric D. Mjolsness†

*Biological Network Modeling Center, California Institute of Technology, Pasadena, California, USA; **Johann Radon Institute for Computational and Applied Mathematics, Linz, Austria; †Institute of Genomics and Bioinformatics and Department of Computer Science, University of California, Irvine, California, USA.


## The Extended Cellerator Modeling Suite

We report on xCellerator , the extended *Cellerator* library, a collection of *Mathematica* packages for signal transduction modeling. The suite contains the following packages (a package is a *Mathematica* software library):

- *xlr8r.m* - Conversion of reaction networks to ODES.
- *MathSBML.m* - Reading, writing, simulation of SBML models.
- *SBMLInvEigAnalyzer.m* - Inverse eigenvalue analysis.
- *Cellzilla.m* - Networks on arbitrary grids of cells.
- *mPower.m* - Computational geometry libraries.
- *Tissue3D.m* - Finite element simulations of cells and tissues.
- *SSA.m* - Gillespie's stochastic simulation algorithim.
- *xlr8r2SBML.m/SBML2xlr8r.m* - xlr8r/SBML conversion.

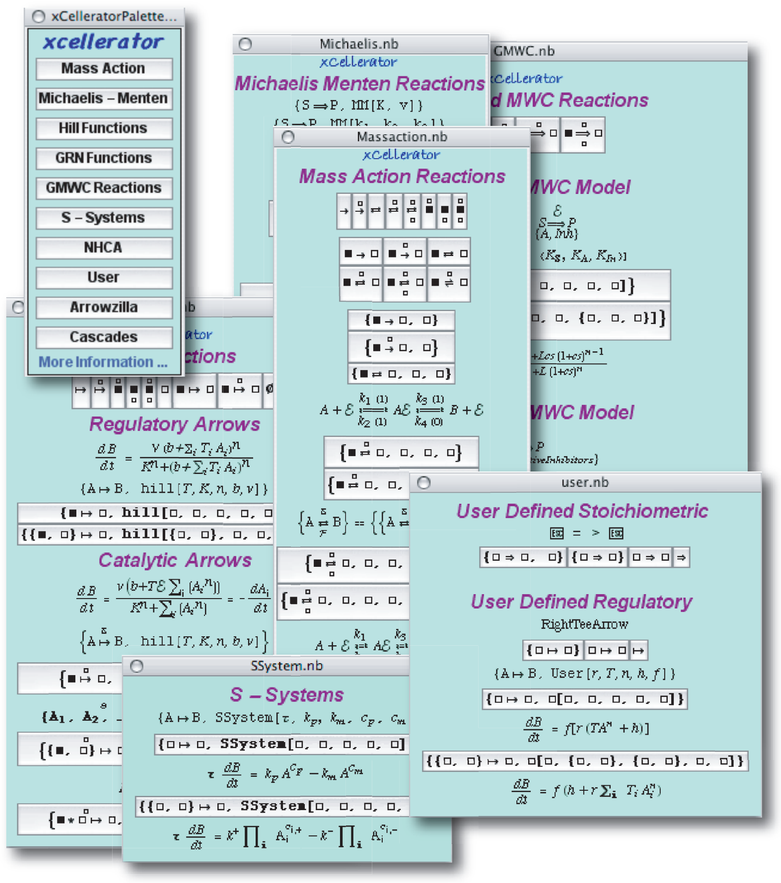
All of the above software is fully open source and freely downloadable (LGPL license).

*xCellerator* implements an extended library of *Cellerator* reactions. *xCellerator* is compatible with *Mathematica* versions 4 through 5.2. (Compatibilty with *Mathematica* version 6.0 is under development).

Models can be built in several ways:

- Textually, via the *Mathematica* notebook interface;
- By using the *xCellerator* palettes;
- As textual script files;
- Via the *MathSBML* model builder;
- By reading an SBML file or Mathematica notebook;
- With the Sigmoid Model Explorer (http://sme.org).

A variety of different arrows represent different types of reactions; in addition, the user can can write his/her own kinetic laws. To aide remembering the types of reactions, a collection of clickable palettes are available.




*Cellerator* reaction networks can be automatically converted into systems of differential equation and then numerically solved to produce predictive time-courses of all model variables. SBML models can either be translated directly to *Cellerator* networks or input by *MathSBML*, which can then convert the model into the corresponding differential and algebraic equations and event data structures so that a full oredictive simulation can be run independent of *xlr8r*.

## A Worked Example: The Brusselator

As an example we will consider a model of coupled Brusselators. Originally proposed by Prignone & Lefever [1968, Journal of Chemical Physics, 48(4): 1695-1700] to study symmetry breaking instabilities in dissipative systems., the Brusselator has been widely utilized in pattern formation models.

The basic kinetic scheme is based on the reation network

$$A \rightarrow X$$
$$B+X \rightarrow Y+D$$
$$2X+Y \rightarrow 3X$$
$$X \rightarrow E$$

The equations for X and Y, after transforming to dimensionless variables u and v, become

$$u' = 1 - (b+1)u + au^2v$$
$$u' = bu - au^2v$$

**http://xCellerator.info**
**http://sbml.org**

## The Brusselator in SBML

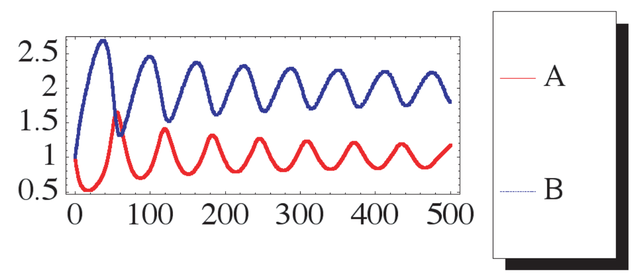
We can model the essential features of the Brusselator with the following reduced model:

$$\{ \varnothing \rightleftarrows A,\ 2A+B \rightarrow 3A,\ A \rightarrow B \}$$

Commands that generate the model, run a simulation, and export SBML.

```
(* define model, initial conditions, and rate constants *)
m = {  {∅ ⇄ A, k1, k2},  {2 A + B → 3 A, k3}, {A → B,  k4}};
ic =  {A → 1,  B → 1};
rates = { k1 → 0.1,  k4 → 0.2,  k2 → 0.1,  k3 → 0.1};

(* print diferential equations *)
Print[TableForm[interpret[m][[1]]]];

(* run a simulation and plot the results *)
n = run[m /. rates,  initialConditions → ic,  timeSpan → 500,
   plot → True,  plotVariables → {A,  B},  plotOptions];

(* export an SBML file *)
sbml = convertToSBML[m,  ic,  rates,  "SBMLID",  "NAME",  "Cell"];
Export["brusselator.xml",  sbml ,  "Text"];
```

```
A'[t] == k1 - k2 A[t] - k4 A[t] + k3 A[t]^2 B[t]
B'[t] == k4 A[t] - k3 A[t]^2 B[t]
```



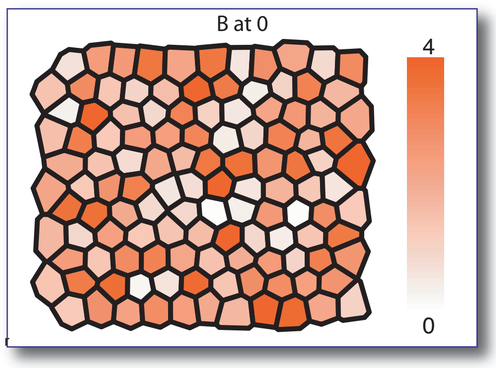

Definition of a pair of coupled Brusselators.

```
m = { {∅ ⇄ A1,  k1,  k2},  {2 A1 + B1 → 3 A1,  k3},  {A1 → B1,  k4},
      {∅ ⇄ A2,  k1,  k2},  {2 A2 + B2 → 3 A2,  k3},  {A2 → B2,  k4},
      {A1 ⇄ A2,  D1,  D1}};
```

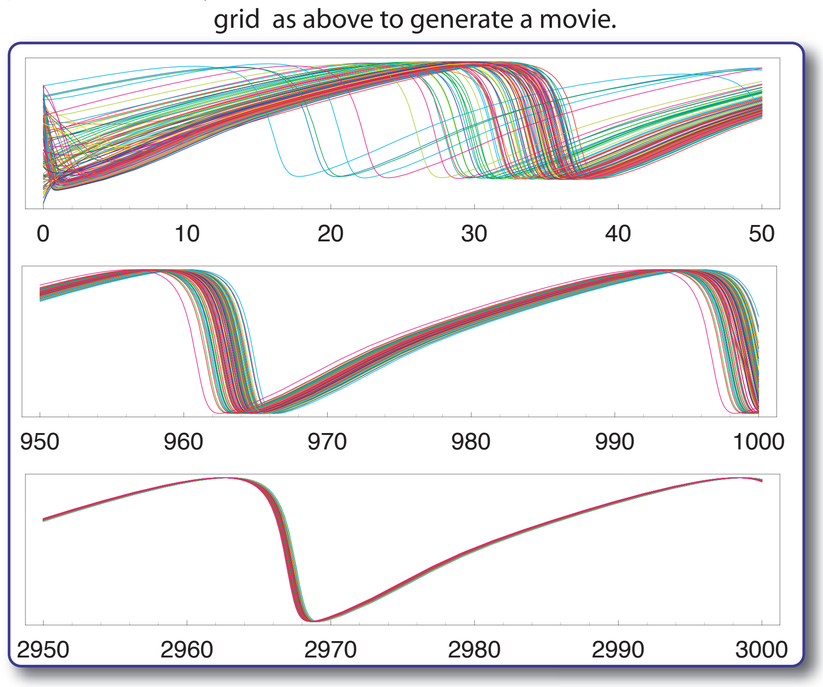
A system of 100 coupled Brusselators in a rectangular grid.

```
(* Define the Grid *)
xy = rectGrid[10,  10,  1];

(* Define the reaction network *)
n = Length[xy]; internal[i_] := {{∅ ⇄ A[i],  k1,  k2},
  {2 A[i] + B[i] → 3 A[i],  k3},  {A[i] → B[i],  k4}};
external[i_,  j_] := { {A[i] → A[j],  DA},  {B[i] → B[j],  DB}};

(* Generate the entire sytem of reactions *)
STN = createNetwork[xy,  internalNetwork → internal,
   interactionNetwork → external];

(* Generate the differential equations *)
istn = interpret[STN];

(* Define random initial conditions *)
r := Random[Real,  {0,  4}];
ic = Table[{A[i][0] == r,  B[i][0] == r},  {i,  1,  n}] // Flatten;

(* Define the rate constants *)
rates = {k1 → .2,  k2 → .2,  k3 → .2,  k4 → .6,  DA → 0.002,  DB → 0.002};

(* Run a simulation *)
sim = run[istn /. rates,  {0,  3000},  initialConditions → ic,
   MaxSteps → 100000];
```

Visualization of grid with random initial conditions.




Results of simulation for three different time spans showing the approach towards synchronization. The results can also be visualized on a grid as above to generate a movie.

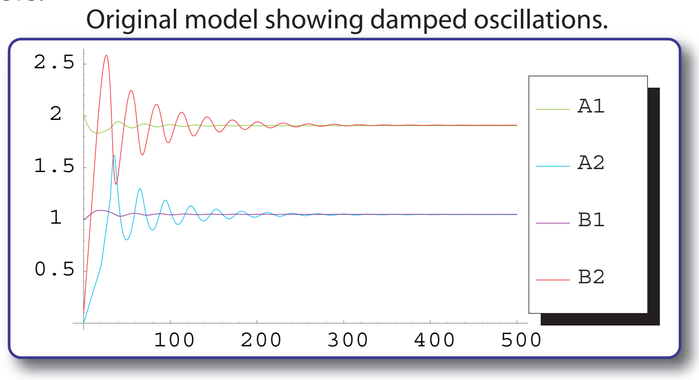



## Inverse Eigenvalue Analysis

Inverse eigenvalue analysis allows one to look for the source of specific qualitative dynamical behaviors. This involves prescribing the values of eigenvalues and partially prescribing the components of the critical eigenvector. Computationally, lift-and-project and quasi-Newton iterations are performed. To solve the inverse problem, sparsity-promoting regularization based on lp or log functionals are used.

We have implemented the function *FindOscillations*, in which a *MathSBML* model is passed in together with the desired imaginary component of the critical eigenvector. Additional conditions on the oscillatory pattern can be set by using the *CritEigVecConstraint* option. After performing the inverse analysis, a *MathSBML* model with the idenfitied parameters is then returned, which subsequently can either be simulated within *MathSBML* or analyzed by the bifurcation software, *MATCONT* (matcont.ugent.be). We illustrate this fora pair of coupled Brusselators.

Original model showing damped oscillations.



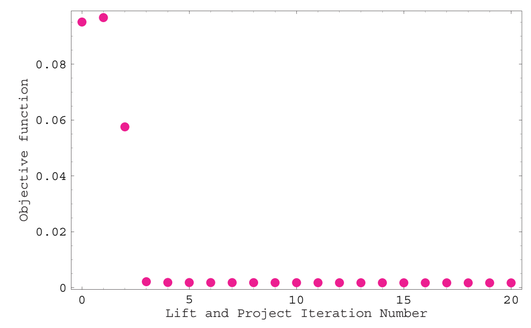

Call to *FindOscillations*.

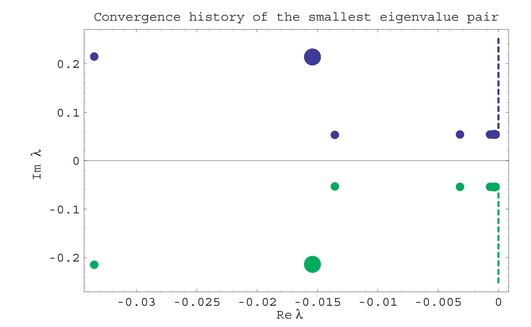
```
{mIden, IdenParam, ReturnedEigVal, ReturnedFuncVal,
   ReturnedSol, status} =
  FindOscillations[
   "ModifiedModels/two-free-coupled.xml", 0.05,
   NMax → 20, DisplayLevel → 2,
   FixedParam → {twoFreeCoupled`C1, twoFreeCoupled`C2},
   RegularizationStructure → {mu → 1`*^-4},
   SolnMethod → {Strategy → "LP"}, tEnd → 200];
```

Partial output of *FindOscillations*.

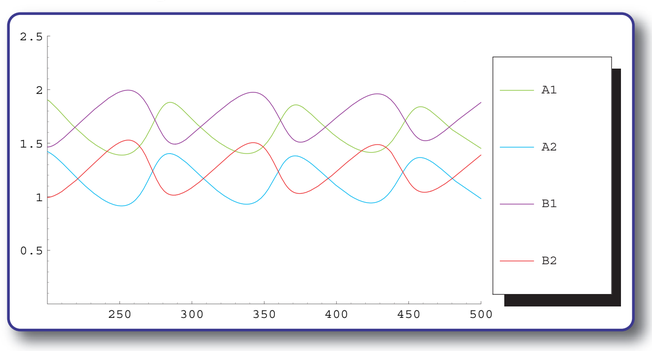
```
LP # 19 Jac.Diff.Frob.Norm: 3.44×10^-6  Eig.Err.Norm: 3.33×10^-5  Penalty:
   1.66×10^-3  Min.Eig:{-3.91×10^-4 + ( 5.41×10^-2) i, -3.91×10^-4 - ( 5.41×10^-2) i}
  ->{-0.000048 +  0.05 i, -0.000048 -  0.05 i}
Using last iterate...
Optim solution found in list position = 21
Eigenvalue solution: {-0.000354995 + 0.0540615 i, -0.000354995 - 0.0540615 i}
Param.Select.Criterion: 0.05
Identified Parameters: {twoFreeCoupled`k1 → 0.00676708,
   twoFreeCoupled`k2 → 0.000100026, twoFreeCoupled`k4 → 0.287528}
Occurrence of identified parameters in ODE system:
```

```
twoFreeCoupled`A1'[t] == -twoFreeCoupled`D1 twoFreeCoupled`A1[t] + twoFree
twoFreeCoupled`A2'[t] == twoFreeCoupled`k5 + twoFreeCoupled`D1 twoFreeCoup
twoFreeCoupled`B1'[t] == -twoFreeCoupled`k3 twoFreeCoupled`A1[t]^2 twoFree
twoFreeCoupled`B2'[t] == twoFreeCoupled`k8 twoFreeCoupled`A2[t] - twoFreeC
```




Oscillating solutions with identified parameters.




Optimization is performed using the *Optimization Toolbox* developed by M. Asghar Bhatti, who has kindly consented to allow distribution of his sofware with the MathSBML Inverse Eigenvalue Analysis distribution [Bhatti, *Practical Optimization Methods with Mathematica Applications*, Springer, 2001].

bnmc
BECKMAN INSTITUTE
BIOLOGICAL NETWORK MODELING CENTER

## Acknowledgements


Portions of this work were supported by grants from the Vienna Science and Technology Fund (Project MA 05); the Beckman Institute at the California Institute of Technology; and the US National Science Foundation (Award 0330786).