

# The SBML ODE Solver Library

## capabilities, limitations and outlook

<http://www.tbi.univie.ac.at/~raim/odeSolver/>

**James Lu**

Johann Radon Institute for Computational and Applied Mathematics  
Inverse Problems Group  
RICAM Linz, Austria

**Rainer Machné**

Theoretical Biochemistry Group  
University of Vienna, Austria

11<sup>th</sup> SBML Forum Meeting, Tokyo 2006

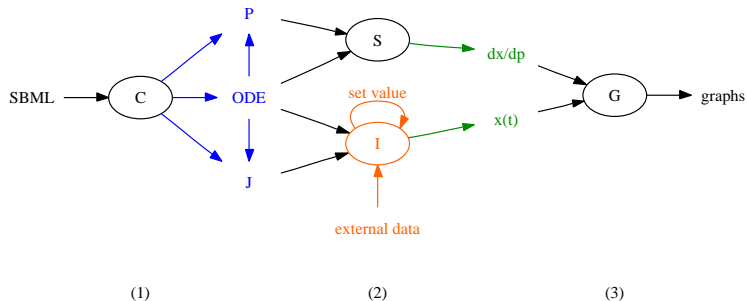
# Outline

- ▶ SOSlib Overview
- ▶ Capabilities:  $reactions \Rightarrow ODE \Rightarrow x(t)$ 
  - ▶ Detailed Interfaces to **Structure** and **Dynamics**
  - ▶ **Forward and Adjoint Sensitivity Analysis**
- ▶ Limitations: *DAEs*, events, delays
- ▶ Outlook:  $reactions \Leftarrow ODE \Leftarrow x(t)$
- ▶ People

# Outline

- ▶ SOSlib Overview
- ▶ Capabilities:  $reactions \Rightarrow ODE \Rightarrow x(t)$ 
  - ▶ Detailed Interfaces to **Structure** and **Dynamics**
  - ▶ **Forward and Adjoint Sensitivity Analysis**
- ▶ Limitations: *DAEs*, events, delays
- ▶ Outlook:  $reactions \Leftarrow ODE \Leftarrow x(t)$
- ▶ People

# The SBML ODE Solver Library: ISO-C90, LGPL



```
sbml      = readSBML(`glycolysis.xml`);  
settings  = CvodeSettings_createWithTime(100, 50);  
results   = SBML_odeSolver(sbml, settings);  
timecourse = SBMLResults_getTimeCourse(results, `flux_G6PDH`);  
value     = TimeCourse_getValue(timecourse, 3);
```

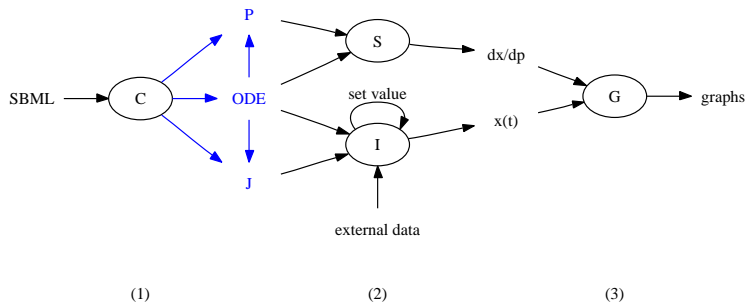
*A native API for symbolic and fast numerical analysis of reaction networks.*

Machné R, Finney A, Müller S, Lu J, Widder S, Flamm C. Bioinformatics 2006, 22(11):1406-7

**Webservice:**

[http://rna.tbi.univie.ac.at/cgi-bin/SBML\\_odeSolver.cgi](http://rna.tbi.univie.ac.at/cgi-bin/SBML_odeSolver.cgi)

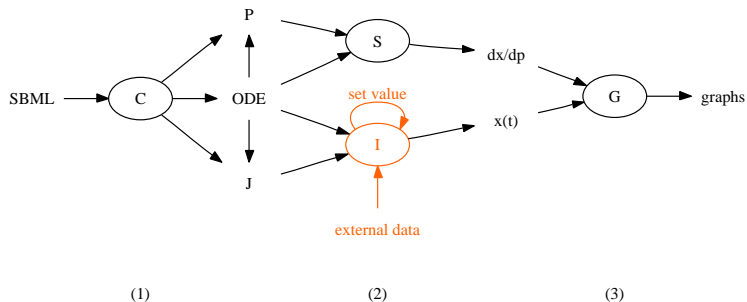
# (1) Analytic Capabilities - via libSBML



```
odeModel = ODEModel_createFromFile('glycolysis.xml');  
if ( ODEModel_constructJacobian(odeModel) )  
    formula = ODEModel_getJacobianIJEntry(odeModel, i, j);
```

- ▶ **AST** evaluation, differentiation, simplification
- ▶ ODE model structure:  $dx/dt = f(x, t, p)$
- ▶ Jacobian and parametric matrices:  $J = df/dx$  and  $P = df/dp$

## (2) Numerical Capabilities - via SUNDIALS/CVODES

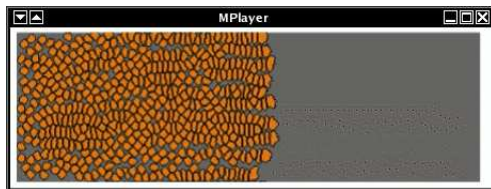


```
ii = IntegratorInstance_create(odemodel, settings);  
while ( !IntegratorInstance_timeCourseCompleted(ii) ) {  
    IntegratorInstance_integrateOneStep(ii);  
    if ( IntegratorInstance_getTime(ii) > 30 )  
        IntegratorInstance_setVariableValue(variable, 300);  
}
```

- ▶ External data sets
- ▶ Multi-scale modeling
- ▶ Hybrid solvers

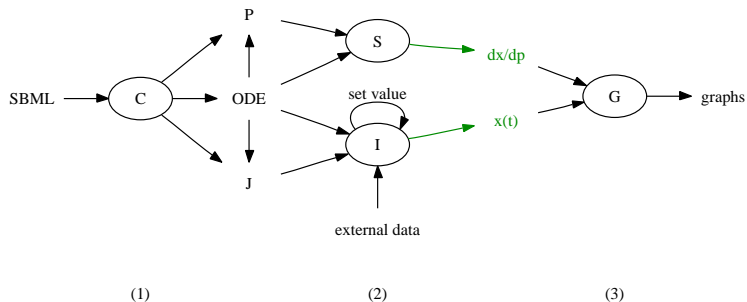
# Multi-scale modeling

A collaboration with Camille Stephan-Otto Attolini *CelloS*:



- ▶ Cellular Potts model + One SBML model
- ▶ Multiple IntegratorInstances, communicating via `setVariableValue`

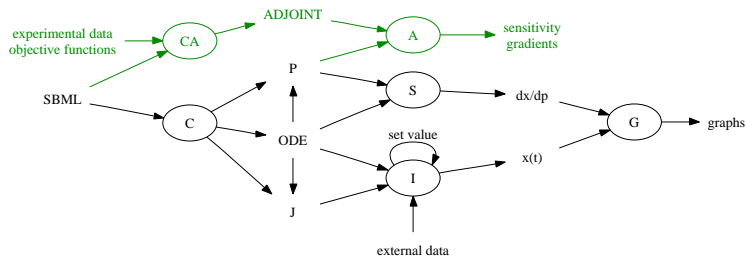
## (2) Numerical Capabilities - via SUNDIALS/CVODES



```
CvodeSettings_setSensitivity(settings, 1);  
...  
value = TimeCourse_getSensitivity(timecourse, i, j);
```

- ▶ non-stiff and stiff ODEs via Adams-Moulton or BDF methods:  $x(t)$
- ▶ Forward sensitivities:  $s_p(t) = dx(t)/dp$  and  $s_x(t) = dx(t)/dx_0$
- ▶ Adjoint sensitivities - gradients towards target data

## (2) Numerical Capabilities - via SUNDIALS/CVODES



```
CvodeSettings_setSensitivity(settings, 1);  
...  
value = TimeCourse_getSensitivity(timecourse, i, j);
```

- ▶ non-stiff and stiff ODEs via Adams-Moulton or BDF methods:  $x(t)$
- ▶ Forward sensitivities:  $s_p(t) = dx(t)/dp$  and  $s_x(t) = dx(t)/dx_0$
- ▶ Adjoint sensitivities - gradients towards target data

# Forward Sensitivity Analysis

- ▶ Differentiating solution time-series in  $t \in [0, T]$  wrt:

$$\text{param : } s_p(0) = \mathbf{0}; \quad \dot{s}_p(t) = \frac{df(x, t, p)}{dx} s_p(t) + \frac{df(x, t, p)}{dp}$$

$$\text{IC : } s_x(0) = \mathbf{I}; \quad \dot{s}_x(t) = \frac{df(x, t, p)}{dx} s_x(t)$$

- ▶ SUNDIALS/CVODES methods:  
simultaneous, staggered, ...
- ▶ Biological applications
  - ▶ Fisher Information Matrix (FIM)
  - ▶ Finite-time Lyapunov exponent (FTLE)

# Forward Sensitivity Analysis

- ▶ Differentiating solution time-series in  $t \in [0, T]$  wrt:

$$\begin{aligned} \text{param : } s_p(0) = \mathbf{0}; \quad \dot{s}_p(t) &= \frac{df(x, t, p)}{dx} s_p(t) + \frac{df(x, t, p)}{dp} \\ \text{IC : } s_x(0) = \mathbf{I}; \quad \dot{s}_x(t) &= \frac{df(x, t, p)}{dx} s_x(t) \end{aligned}$$

- ▶ SUNDIALS/CVODES methods:  
simultaneous, staggered, ...
- ▶ Biological applications
  - ▶ **Fisher Information Matrix** (FIM)
    - ▶ Confidence intervals for parameter estimation :  
**Eryk Wolski**, Andreas Kremling, Francis Doyle, *etc*
  - ▶ **Finite-time Lyapunov exponent** (FTLE)
    - ▶ Analysis of transient signalling governing cell behavior:  
B. B. Aldridge, G. Haller, P. K. Sorger, D. A. Lauffenburger,  
*IEE Sys. Biol.* (2006).

# Adjoint Sensitivity Analysis

- ▶ Forward sensitivity:  $\mathcal{O}(\dim(p))$  ODE solves
- ▶ Can do (much) better when principal quantity of interest is a *functional*, i.e. mapping solutions to numbers.

Computational cost  $\mathcal{O}(1)$ !

- ▶ Given  $v(t) \in L_2[0, T]$ , consider functional

$$G(x) = \int_0^T v(t) \cdot x(t) dt$$

- ▶ Introduce *adjoint equation* for  $\psi(t)$  over  $t \in [T, 0]$ :

$$\psi(T) = \mathbf{0}; \dot{\psi}(t) = - \left[ \frac{df(x, t, p)}{dx} \right]^T \psi(t) + v(t).$$

- ▶ Biological applications of adjoint methods:

- ▶ *Parameter estimation*, e.g. for plane cell polarity in *Drosophila* wing: Keith Amonlirdviman, Claire Tomlin
- ▶ *Phase response curves* (PRC), e.g. in neuron models: Willy Govaerts, Bart Sautois
- ▶ *Inverse bifurcation analysis*, e.g. in cell cycle and circadian rhythm: in collaboration with Heinz Engl, Peter Schuster, etc

# Adjoint Sensitivity Analysis

- ▶ Forward sensitivity:  $\mathcal{O}(\dim(p))$  ODE solves
- ▶ Can do (much) better when principal quantity of interest is a *functional*, i.e. mapping solutions to numbers.

Computational cost  $\mathcal{O}(1)$ !

- ▶ Given  $v(t) \in L_2[0, T]$ , consider functional

$$G(x) = \int_0^T v(t) \cdot x(t) dt$$

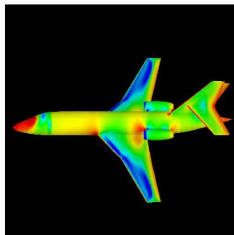
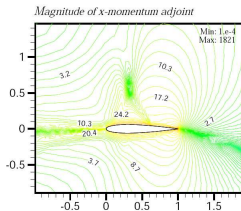
- ▶ Introduce *adjoint equation* for  $\psi(t)$  over  $t \in [T, 0]$ :

$$\psi(T) = \mathbf{0}; \dot{\psi}(t) = - \left[ \frac{df(x, t, p)}{dx} \right]^T \psi(t) + v(t).$$

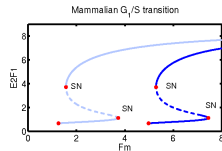
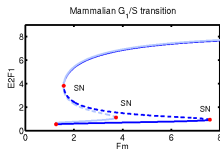
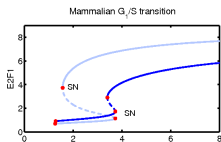
- ▶ Biological applications of adjoint methods:
  - ▶ **Parameter estimation**, e.g. for plane cell polarity in *Drosophila* wing: Keith Amonlirdviman, Claire Tomlin
  - ▶ **Phase response curves** (PRC), e.g. in neuron models: Willy Govaerts, Bart Sautois
  - ▶ **Inverse bifurcation analysis**, e.g. in cell cycle and circadian rhythm: in collaboration with Heinz Engl, Peter Schuster, etc

# Adjoint Solution: Example Applications

- ▶ Aerodynamic design of aircrafts: Antony Jameson (Stanford), Jaime Peraire (MIT)



- ▶ Design of gene switches



# Functional Equivalence Check

- ▶ Functional sensitivity computed in forward manner:

$$\frac{dG}{d\{x,p\}} = \int_0^T v(t) \cdot s_{\{p,x\}}(t) dt$$

- ▶ Functional sensitivity computed in adjoint manner, wrt:

- ▶ parameters:  $\frac{dG}{dp} = - \int_0^T \frac{df}{dp} \cdot \psi(t) dt$

- ▶ initial conditions:  $\frac{dG}{dx} = -\psi(0)$

- ▶ Calculation of sensitivities:

# Functional Equivalence Check

- ▶ Functional sensitivity computed in forward manner:

$$\frac{dG}{d\{x,p\}} = \int_0^T v(t) \cdot s_{\{p,x\}}(t) dt$$

- ▶ Functional sensitivity computed in adjoint manner, wrt:

- ▶ parameters:  $\frac{dG}{dp} = - \int_0^T \frac{df}{dp} \cdot \psi(t) dt$

- ▶ initial conditions:  $\frac{dG}{dx} = -\psi(0)$

- ▶ Calculation of sensitivities:

# Functional Equivalence Check

- ▶ Functional sensitivity computed in forward manner:

$$\frac{dG}{d\{x,p\}} = \int_0^T v(t) \cdot s_{\{p,x\}}(t) dt$$

- ▶ Functional sensitivity computed in adjoint manner, wrt:

- ▶ parameters:  $\frac{dG}{dp} = - \int_0^T \frac{df}{dp} \cdot \psi(t) dt$

- ▶ initial conditions:  $\frac{dG}{dx} = -\psi(0)$

- ▶ Calculation of sensitivities:

# Functional Equivalence Check

- ▶ Functional sensitivity computed in forward manner:

$$\frac{dG}{d\{x,p\}} = \int_0^T v(t) \cdot s_{\{p,x\}}(t) dt$$

- ▶ Functional sensitivity computed in adjoint manner, wrt:

- ▶ parameters:  $\frac{dG}{dp} = - \int_0^T \frac{df}{dp} \cdot \psi(t) dt$

- ▶ initial conditions:  $\frac{dG}{dx} = -\psi(0)$

- ▶ Calculation of sensitivities:

```
IntegratorInstance_setLinearObjectiveFunction(ii, "G.txt");
CvodeSettings_setDoAdj(settings);
...
while( !IntegratorInstance_timeCourseCompleted(ii) )
    if ( !IntegratorInstance_integrateOneStep(ii) ) break;
flag = IntegratorInstance_CVODEQuad(ii);

CvodeSettings_setAdjPhase(settings);
...
while( !IntegratorInstance_timeCourseCompleted(ii) )
    if ( !IntegratorInstance_integrateOneStep(ii) ) break;
flag = IntegratorInstance_CVODEQuad(ii);
```

# Functional Equivalence Check

- ▶ Functional sensitivity computed in forward manner:

$$\frac{dG}{d\{x,p\}} = \int_0^T v(t) \cdot s_{\{p,x\}}(t) dt$$

- ▶ Functional sensitivity computed in adjoint manner, wrt:

- ▶ parameters:  $\frac{dG}{dp} = - \int_0^T \frac{df}{dp} \cdot \psi(t) dt$

- ▶ initial conditions:  $\frac{dG}{dx} = -\psi(0)$

- ▶ Calculation of sensitivities:

**Table:** Comparison using tolerances  $ATOL = 10^{-5}$ ,  $RTOL = 10^{-5}$

	$dG/dMAPK$	$dG/dMAPK\_P$	$dG/dK1$	$dG/dKi$
Forw. sens.	8394.728491	556.1006739	1593.340344	-2867.770424
Adj. sens.	8394.744358	555.9706561	1593.196428	-2867.321621

# Functional Equivalence Check

- ▶ Functional sensitivity computed in forward manner:

$$\frac{dG}{d\{x,p\}} = \int_0^T v(t) \cdot s_{\{p,x\}}(t) dt$$

- ▶ Functional sensitivity computed in adjoint manner, wrt:

- ▶ parameters:  $\frac{dG}{dp} = - \int_0^T \frac{df}{dp} \cdot \psi(t) dt$

- ▶ initial conditions:  $\frac{dG}{dx} = -\psi(0)$

- ▶ Calculation of sensitivities:

**Table:** Comparison using tolerances  $ATOL = 10^{-8}$ ,  $RTOL = 10^{-8}$

	$dG/dMAPK$	$dG/dMAPK\_P$	$dG/dK1$	$dG/dKi$
Forw. sens.	<b>8394.74758</b>	555.974022	1593.187535	-2867.35402
Adj. sens.	<b>8394.747585</b>	555.9737532	1593.187067	-2867.352683

# Functional Equivalence Check

- ▶ Functional sensitivity computed in forward manner:

$$\frac{dG}{d\{x,p\}} = \int_0^T v(t) \cdot s_{\{p,x\}}(t) dt$$

- ▶ Functional sensitivity computed in adjoint manner, wrt:

- ▶ parameters:  $\frac{dG}{dp} = - \int_0^T \frac{df}{dp} \cdot \psi(t) dt$

- ▶ initial conditions:  $\frac{dG}{dx} = -\psi(0)$

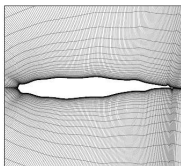
- ▶ Calculation of sensitivities:

Table: Comparison using tolerances  $ATOL = 10^{-11}$ ,  $RTOL = 10^{-11}$

	$dG/dMAPK$	$dG/dMAPK\_P$	$dG/dK1$	$dG/dKi$
Forw. sens.	8394.747589	555.9737615	1593.187067	-2867.352763
Adj. sens.	8394.747589	555.9737615	1593.187067	-2867.352763

# Parameter Identification as (Ill-posed) Inverse Problems

- ▶ A requirement for a mathematical problem to be well-posed (Hadamard)
  - ▶ the solution depends continuously on the data
- ▶ For **ill-posed** problems, **regularization** (stabilization) needed
- ▶ Example of unstable solution: rough airfoil boundary



- ▶ Regularization methods:
  - ▶ Tikhonov regularization:  $\dots + \mu \|p - p^*\|_{\mathcal{P}}$
  - ▶ *Iteratively regularized* Gauss-Newton, Levenberg-Marquadt,  $\dots$
- ▶ Relate data noise to stabilization term: discrepancy principle,  $\dots$

# Limitations - in order of ToDo list

## Current framework:

- ▶ Online compilation via TCC (Andrew Finney's code)
- ▶ Variable compartments: ODEs in substance/time
- ▶ Exact event handling (Sundials *root finder*)
- ▶ SWIG-based interfaces
- ▶ Differential Algebraic Equations (Sundials *IDA*)
- ▶ Delay Differential Equations (e.g. *XPP*)

## Hook in alternative solvers and libraries:

- ▶ mixed BDF/Adams-Moulton: *LSODA*
- ▶ Parameter estimation: *libSRES*, *SBML-PET*
- ▶ Stochastic Solvers: *::basis::*

Do you have preferences? Want to join?

Contact us at [sbmlsolver-discuss@lists.sourceforge.net](mailto:sbmlsolver-discuss@lists.sourceforge.net)

# Limitations - in order of ToDo list

Current framework:

- ▶ Online compilation via TCC (Andrew Finney's code)
- ▶ Variable compartments: ODEs in substance/time
- ▶ Exact event handling (Sundials *root finder*)
- ▶ SWIG-based interfaces
- ▶ Differential Algebraic Equations (Sundials *IDA*)
- ▶ Delay Differential Equations (e.g. *XPP*)

Hook in alternative solvers and libraries:

- ▶ mixed BDF/Adams-Moulton: *LSODA*
- ▶ Parameter estimation: *libSRES*, *SBML-PET*
- ▶ Stochastic Solvers: *::basis::*

Do you have preferences? Want to join?

Contact us at [sbmlsolver-discuss@lists.sourceforge.net](mailto:sbmlsolver-discuss@lists.sourceforge.net)

# Limitations - in order of ToDo list

## Current framework:

- ▶ Online compilation via TCC (Andrew Finney's code)
- ▶ Variable compartments: ODEs in substance/time
- ▶ Exact event handling (Sundials *root finder*)
- ▶ SWIG-based interfaces
- ▶ Differential Algebraic Equations (Sundials *IDA*)
- ▶ Delay Differential Equations (e.g. *XPP*)

## Hook in alternative solvers and libraries:

- ▶ mixed BDF/Adams-Moulton: *LSODA*
- ▶ Parameter estimation: *libSRES*, *SBML-PET*
- ▶ Stochastic Solvers: *::basis::*

Do you have preferences? Want to join?

Contact us at [sbmlsolver-discuss@lists.sourceforge.net](mailto:sbmlsolver-discuss@lists.sourceforge.net)

# Outlook

## New Modules:

- ▶ Inverse methods: *reactions*  $\Leftrightarrow$  ODE  $\Leftrightarrow$   $x(t)$ 
    - ▶ Parameter identification, inverse bifurcation
    - ▶ Separate package, GPL or other license
  - ▶ Timescale decomposition
    - ▶ Fast reaction analysis
  - ▶ Stoichiometry matrix
    - ▶ *reactions*  $\Leftrightarrow$  ODE  $i \frac{1}{2}$
    - ▶ Graph drawing module !
- > 400 downloads (w/o CVS), **we'd appreciate more feedback**
- ▶ What do users use?
  - ▶ What do users need?

Do you have preferences? Want to join?

Contact us at [sbmlsolver-discuss@lists.sourceforge.net](mailto:sbmlsolver-discuss@lists.sourceforge.net)

# Outlook

## New Modules:

- ▶ Inverse methods:  $reactions \Leftrightarrow ODE \Leftrightarrow x(t)$ 
  - ▶ Parameter identification, inverse bifurcation
  - ▶ Separate package, GPL or other license
- ▶ Timescale decomposition
  - ▶ Fast reaction analysis
- ▶ Stoichiometry matrix
  - ▶  $reactions \Leftrightarrow ODE \dot{x} = \frac{1}{2}$
  - ▶ Graph drawing module !

> 400 downloads (w/o CVS), **we'd appreciate more feedback**

- ▶ What do users use?
- ▶ What do users need?

**Do you have preferences? Want to join?**

Contact us at [sbmlsolver-discuss@lists.sourceforge.net](mailto:sbmlsolver-discuss@lists.sourceforge.net)

# Outlook

## New Modules:

- ▶ Inverse methods:  $reactions \Leftrightarrow ODE \Leftrightarrow x(t)$ 
  - ▶ Parameter identification, inverse bifurcation
  - ▶ Separate package, GPL or other license
- ▶ Timescale decomposition
  - ▶ Fast reaction analysis
- ▶ Stoichiometry matrix
  - ▶  $reactions \Leftrightarrow ODE \dot{c} \frac{1}{2}$
  - ▶ Graph drawing module !

> 400 downloads (w/o CVS), **we'd appreciate more feedback**

- ▶ What do users use?
- ▶ What do users need?

Do you have preferences? Want to join?

Contact us at [sbmlsolver-discuss@lists.sourceforge.net](mailto:sbmlsolver-discuss@lists.sourceforge.net)

# SOSlib - an open source project

- ▶ TBI Vienna, Peter Schuster's group
  - ▶ Christoph Flamm, Stefanie Widder, Rainer Machné
- ▶ RICAM Linz, Heinz Engl's group
  - ▶ Stefan Müller, James Lu, Philipp Kügler
- ▶ SBML core team
  - ▶ Andrew Finney
  
- ▶ Akira Funahashi (CellDesigner team)
- ▶ Nicolas Le Novère (BioModels DB team)
- ▶ Eryk Wolski (SBML\_odeSolverR)



Wiener Wissenschafts-, Forschungs- und Technologiefonds

**Project Nr. MA05**

# SOSlib - an open source project

- ▶ TBI Vienna, Peter Schuster's group
  - ▶ Christoph Flamm, Stefanie Widder, Rainer Machné
- ▶ RICAM Linz, Heinz Engl's group
  - ▶ Stefan Müller, James Lu, Philipp Kügler
- ▶ SBML core team
  - ▶ Andrew Finney
  
- ▶ Akira Funahashi (CellDesigner team)
- ▶ Nicolas Le Novère (BioModels DB team)
- ▶ Eryk Wolski (SBML\_odeSolve**R**)



Wiener Wissenschafts-, Forschungs- und Technologiefonds

**Project Nr. MA05**