
Systems Biology Toolbox for MATLAB

A computational platform for research in Systems Biology

(SBML Enabled Software)
A user's perspective

SBML Workshop 24/10/2005



Fraunhofer

CHALMERS
Research Centre
Industrial Mathematics

Henning Schmidt

Vision

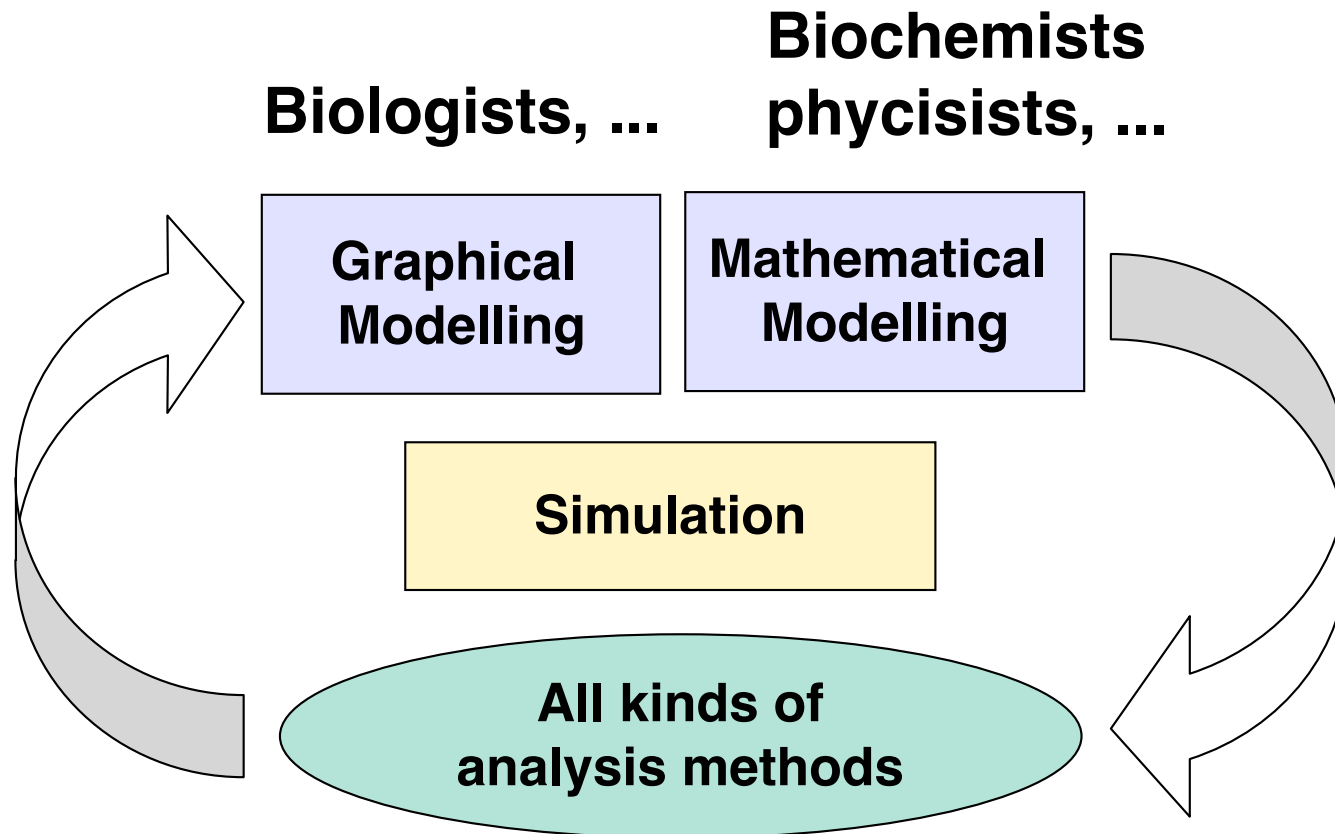
- The Systems Biology Toolbox for MATLAB offers systems biologists an **open and user extensible environment**, in which to explore ideas, prototype and share new algorithms, and build applications for the **analysis** and simulation of biological systems.



Henning Schmidt

User – "what user"?

"Target User"



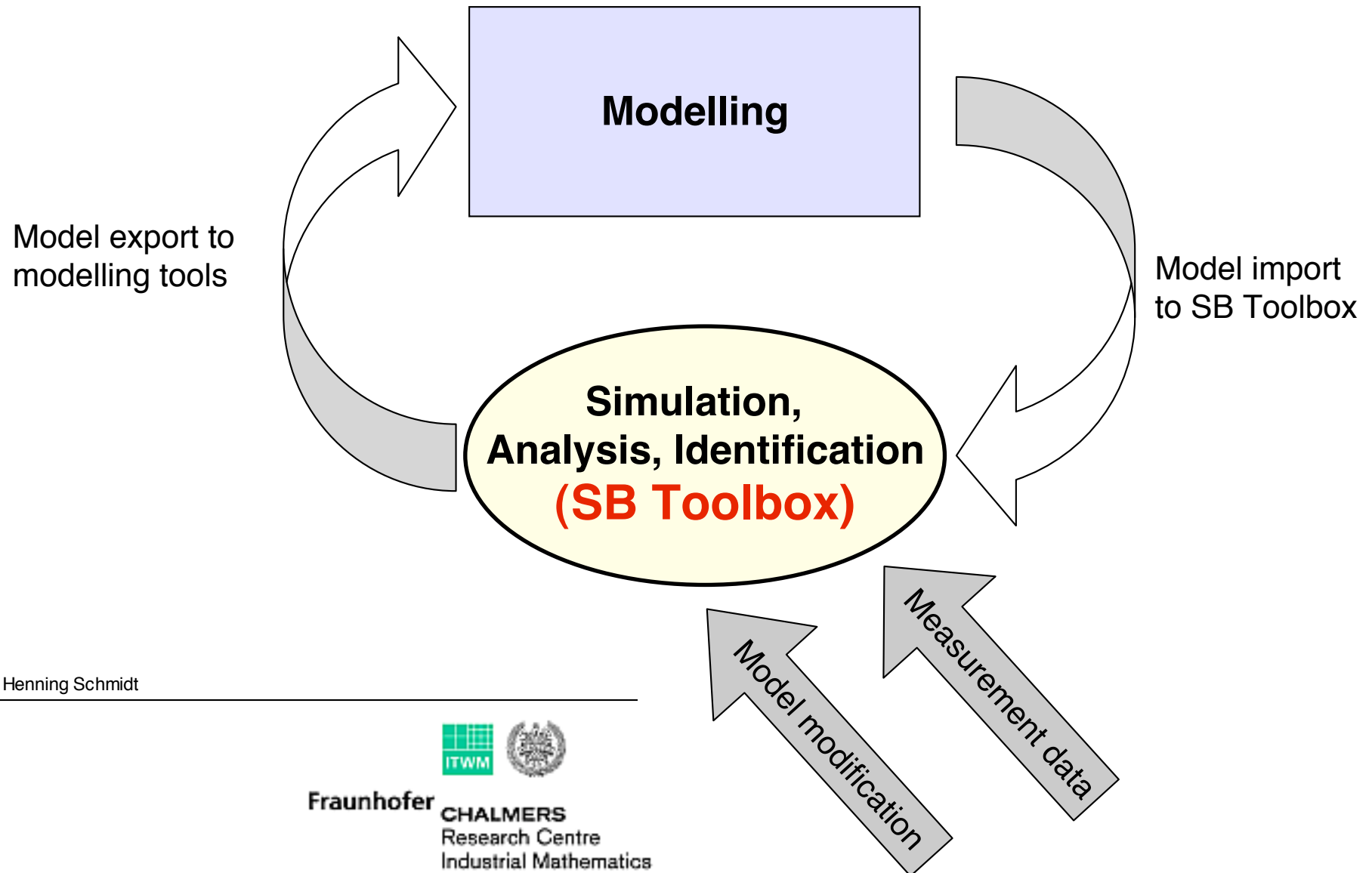
Henning Schmidt

**Engineers,
physicists, ...**

There is, of course, more ...

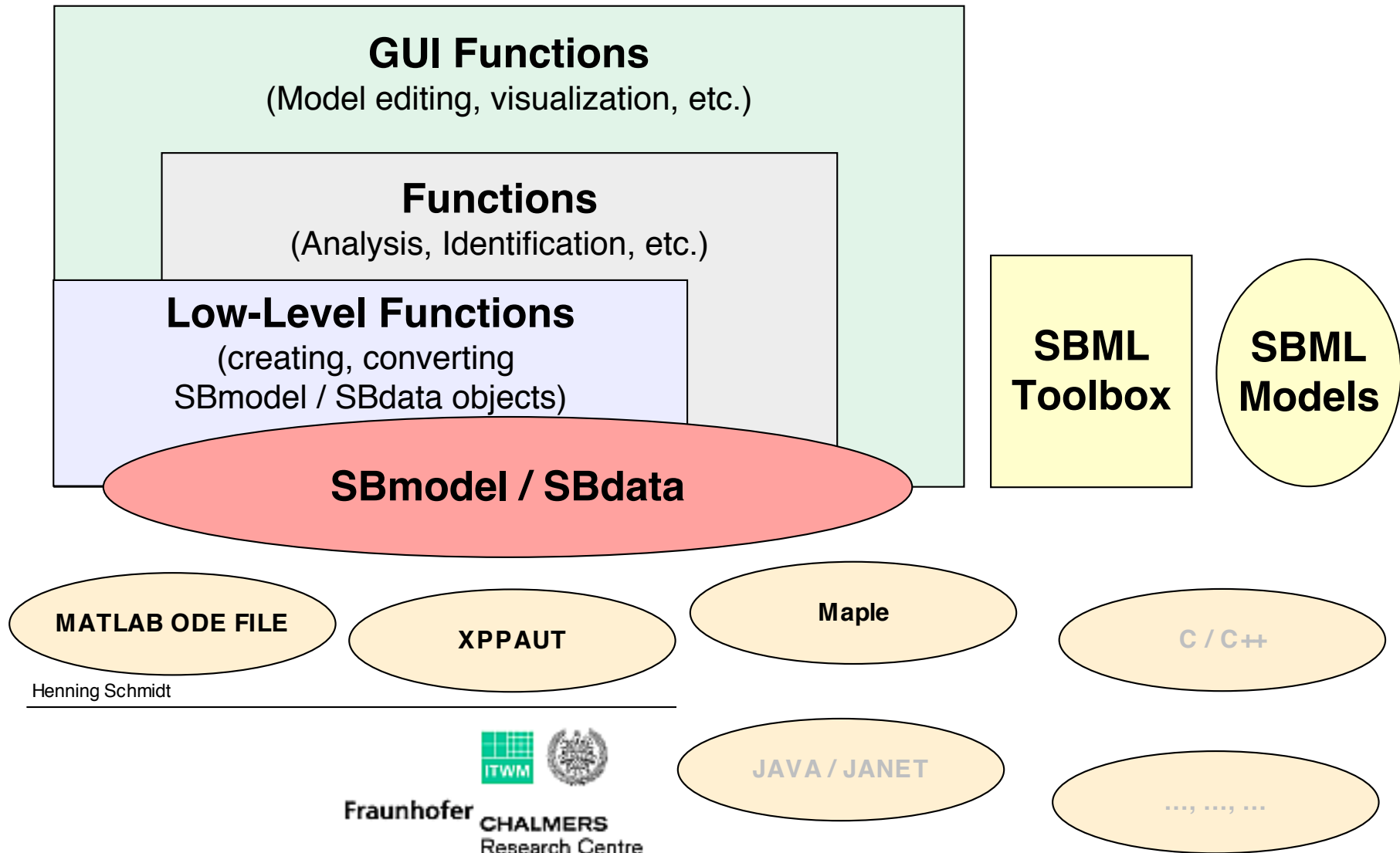
SBTs target user often an engineer

Focus of the SB Toolbox



Henning Schmidt

Design of the SB Toolbox



Henning Schmidt

SBmodel

- *An SBmodel is realized as an object of class SBmodel*
- *The internal data structure of an SBmodel object is given by*

name notes	states.name states.initial states.ODE states.notes states.sbmlnotes	parameters.name parameters.value parameters.notes parameters.sbmlnotes	variables.name variables.formula variables.notes variables.sbmlnotes
reactions.name reactions.formula reactions.notes	functions.name functions.arguments functions.formula functions.notes	functionsMATLAB	events.name events.trigger events.assignment events.notes

Henning Schmidt

SBML compatibility

- Import: L1V1 L1V2 L2V1
- Export: L2V1 (experimental)

- No algebraic rules
- No units
- No non-zero delay for events
- Only simple style of event trigger conditions ... e.g.: $lt/gt/le/ge(expr1,expr2)$

- That's not really limiting at the moment
- As soon as more and more models have these features they can be implemented
- ...

Henning Schmidt

SBdata

- *A data object is realized as an object of class SBdata*
- *The internal data structure of an SBdata object is given by*

name	notes	time
measurements.name		
measurements.stimulusflag		
measurements.data		
measurements.noiseoffset		
measurements.noisevariance		

Functions Overview

- Model Creation and Manipulation
- Measurement Data Creation and Manipulation
- Model Information
- Import/Export of SBmodel/SBdata
- Simulation
- Plotting
- Simple Analysis
- Identification / Parameter Estimation
- Model Reduction
- Bifurcation Analysis
- Parameter Sensitivity Analysis
- Specialized functions ...

Model Import

```
>> model = SBmodel('sunemads.txt')
```

```
SBmodel
```

```
=====
```

```
Name: sunemads  
Number States:      8  
Number Variables:   20  
Number Parameters:  26  
Number Reactions:   11  
Number Functions:   0
```

```
>> model = SBmodel('BIOMD0000000014.xml')
```

```
SBmodel
```

```
=====
```

```
Name: Levchenko2000_MAPK_Scaffold  
Number States:      86  
Number Variables:   0  
Number Parameters:  302  
Number Reactions:   300  
Number Functions:   0
```

Henning Schmidt



Fraunhofer
CHALMERS
Research Centre
Industrial Mathematics

Model Import

```
>> model = SBmodel('BIOMD0000000008.xml');
```

```
??? Error using ==> SBmodel.SBmodel
```

No size for compartment 'cytoplasm' given

```
>> model = SBmodel('BIOMD0000000007.xml');
```

```
??? Error using ==> SBmodel.SBmodel
```

The effect of event 'Start' is delayed. This can not be handled by this toolbox.

Trigger expression for event 'Start' probably contains unsupported operator.

Allowed is only le/t/ge/gt(expression1,expression2).

The variable affected by event 'Start' is not a state.

The effect of event 'Division' is delayed. This can not be handled by this toolbox.

Trigger expression for event 'Division' probably contains unsupported operator.

Allowed is only le/t/ge/gt(expression1,expression2).

The variable affected by event 'Division' is not a state.

Henning Schmidt



Fraunhofer
CHALMERS
Research Centre
Industrial Mathematics

model = SBmodel('BIOMD0000000034.xml');

- ***** MODEL REACTIONS
- reaction_0000001 = compartment_0000002 * (parameter_0000027 * parameter_0000021 + parameter_0000031)
- reaction_0000002 = compartment_0000001 * (parameter_0000028 * parameter_0000020 + parameter_0000032)
- reaction_0000003 = compartment_0000001 * delay(parameter_0000029 * parameter_0000022 + parameter_0000034, parameter_0000039)
- reaction_0000004 = compartment_0000001 * (parameter_0000030 * (power(species_0000009, 2) / (power(species_0000009, 2) + power(parameter_0000010, 2))) * (power(parameter_0000008, 2) / (power(species_0000007, 2) + power(parameter_0000008, 2))) + parameter_0000033)
- reaction_0000005 = compartment_0000001 * parameter_0000036 * species_0000008
- reaction_0000006 = compartment_0000001 * parameter_0000038 * species_0000009
- reaction_0000007 = compartment_0000001 * parameter_0000035 * species_0000007
- reaction_0000008 = compartment_0000002 * (parameter_0000040 * species_0000004 / (parameter_0000041 + species_0000004))
- reaction_0000009 = compartment_0000002 * (parameter_0000040 * species_0000005 / (parameter_0000041 + species_0000005))
- reaction_0000010 = compartment_0000001 * (parameter_0000044 * species_0000006 / (parameter_0000045 + species_0000006))
- reaction_0000011 = compartment_0000001 * (parameter_0000042 * species_0000001 / (parameter_0000043 + species_0000001))
- reaction_0000012 = compartment_0000001 * (parameter_0000042 * species_0000002 / (parameter_0000043 + species_0000002))
- reaction_0000013 = compartment_0000001 * (parameter_0000046 * species_0000003 / (parameter_0000047 + species_0000003))
- reaction_0000014 = compartment_0000001 * parameter_0000048 * species_0000008
- reaction_0000015 = compartment_0000001 * parameter_0000048 * species_0000009
- reaction_0000016 = compartment_0000001 * parameter_0000048 * species_0000007
- reaction_0000017 = compartment_0000002 * parameter_0000048 * species_0000004
- reaction_0000018 = compartment_0000002 * parameter_0000048 * species_0000005
- reaction_0000019 = compartment_0000001 * parameter_0000048 * species_0000001
- reaction_0000020 = compartment_0000001 * parameter_0000048 * species_0000002
- reaction_0000021 = compartment_0000001 * parameter_0000048 * species_0000002
- reaction_0000022 = compartment_0000001 * parameter_0000048 * species_0000003

Reasonable SBML "id"s
would be nice 😊

Almost all models I found
in the database have nice "id"s!!!

>> SBedit(model)

Model editing based on ODEs

The screenshot shows the SBedit software window with the title bar 'SBedit'. The main area is titled 'Model States View'. On the left, there is a vertical toolbar with buttons for 'Edit Model' (Complete, Name, Notes, HTML Notes, States, Parameters, Variables, Reactions, Functions, Events, MATLAB Fcn), 'Simulation' (20, Simulate, Update IC), 'Steady-state', 'Export Text', 'Export TextBC', 'Export SBML', and 'Exit'. The main text area contains the following ODEs and initial conditions:

```
d/dt(glcx) = lnGlc-GlcTrans
d/dt(glc) = 59*GlcTrans-PFK-Storage
d/dt(trioseP) = 2*PFK-GAPDH-glycerol
d/dt(BPG) = GAPDH-LowPart
d/dt(ACA) = LowPart-ADH-59*DifACA
d/dt(ACAx) = DifACA-OutACA
d/dt(NADH) = GAPDH-ADH-glycerol
d/dt(ATP) = -2*PFK+2*LowPart-ATPase-2*Storage

glcx(0) = 1.553070
glc(0) = 6.263074
trioseP(0) = 7.705000
BPG(0) = 0.000270
ACA(0) = 8.181530
ACAx(0) = 1.288360
NADH(0) = 0.330000
ATP(0) = 2.100000
```

Henning Schmidt



Fraunhofer CHALMERS
Research Centre
Industrial Mathematics

>> SBeditBC(model)

Model editing based
on reaction equations

Model Reactions View

In this view you can enter the models reactions. A new reaction definition starts always in a new line, but can go over several lines. The syntax is "reactionname = expression [reversible] % optional comment". Spaces and line breaks are ignored. The "reactionname" needs to consist of characters that are allowed for MATLAB variables. The expression can contain states, variables, and parameters. Reaction names are not allowed to have the same name as states, parameters, variables, or functions. The identifier "[reversible]" should be present when a reaction is reversible only. The comment is separated from the rest by "%". In the reaction comments the character "=" is NOT ALLOWED! You can specify the ODEs for species in terms of reactions or by directly using kinetic formulas in the ODEs. In the latter case reactions do not need to be specified. A combination of both approaches is possible but not very consistent, and should therefore be avoided.

```
<=> glcx : InGlc
      v_f = k0intra*glcx0/yvol
      v_r = k0intra*glcx/yvol

glcx => 59*glc : GlcTrans
      v_f = Vtrans*glcx/(glcx+Ktrans)/yvol

glc+2*ATP => 2*trioseP+2*ADP : PFK
      v_f = VPFK*glc*ATP/(1.0+(ATP/Ki)^q)

trioseP+NAD => BPG+NADH : GAPDH
      v_f = kGAPDH*trioseP*NAD

BPG+2*ADP => ACA+2*ATP : LowPart
      v_f = klow*BPG*ADP

ACA+NADH => NAD : ADH
      v_f = kADH*ACA*NADH

ATP => ADP : ATPase
      v_f = kATPase*ATP

trioseP+NADH => NAD : glycerol
      v_f = kglyc*trioseP*NADH

glc+2*ATP => 2*ADP : Storage
      v_f = kstor*glc*ATP

59*ACA <=> ACAx : DifACA
      v_f = kdif*ACA/yvol
      v_r = kdif*ACAx/yvol

ACAx => : OutACA
```

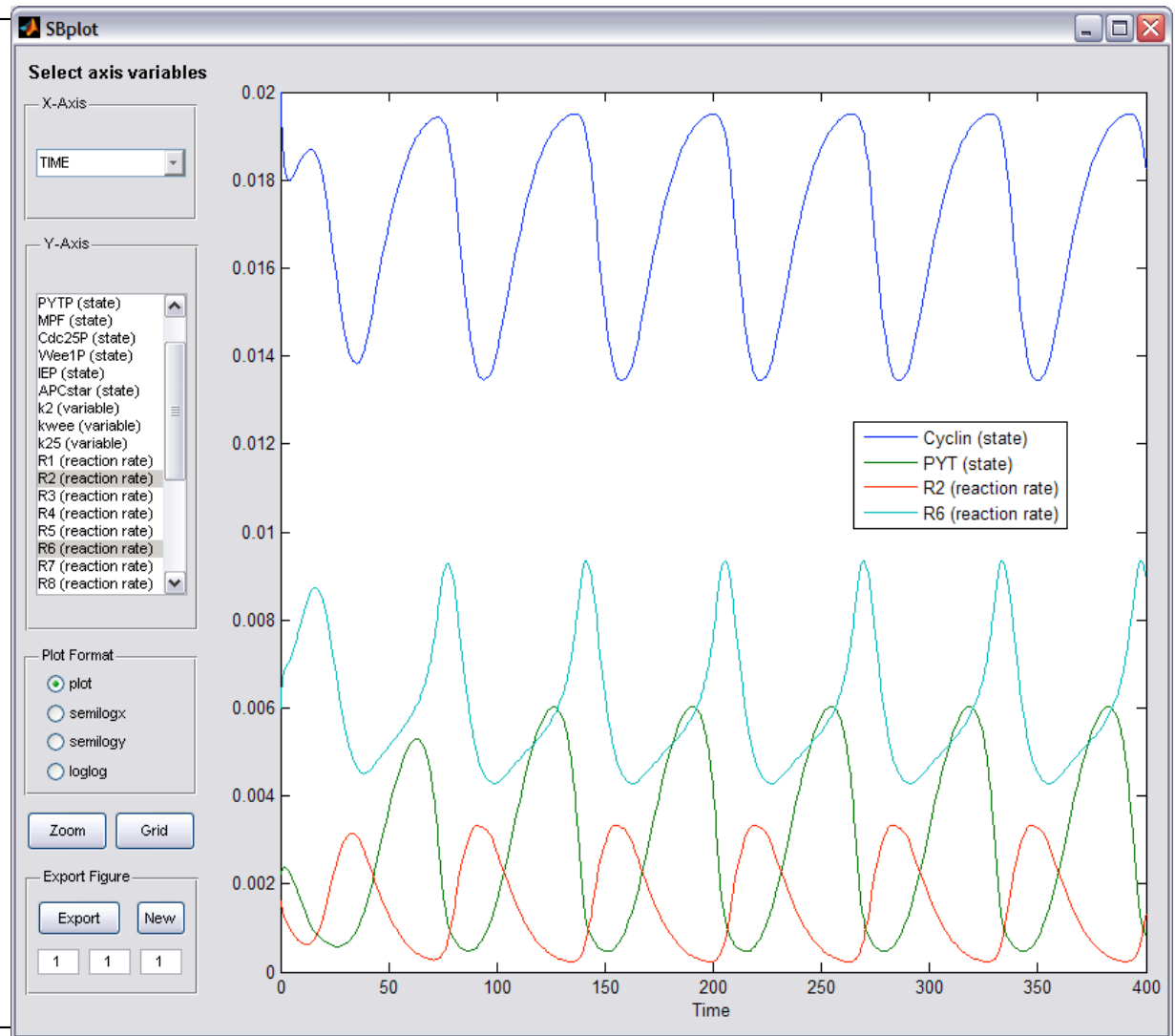
Henning Schmidt



Fraunhofer
CHALMERS
Research Centre
Industrial Mathematics

>> SBsimulate(model,400)

Display of simulation results



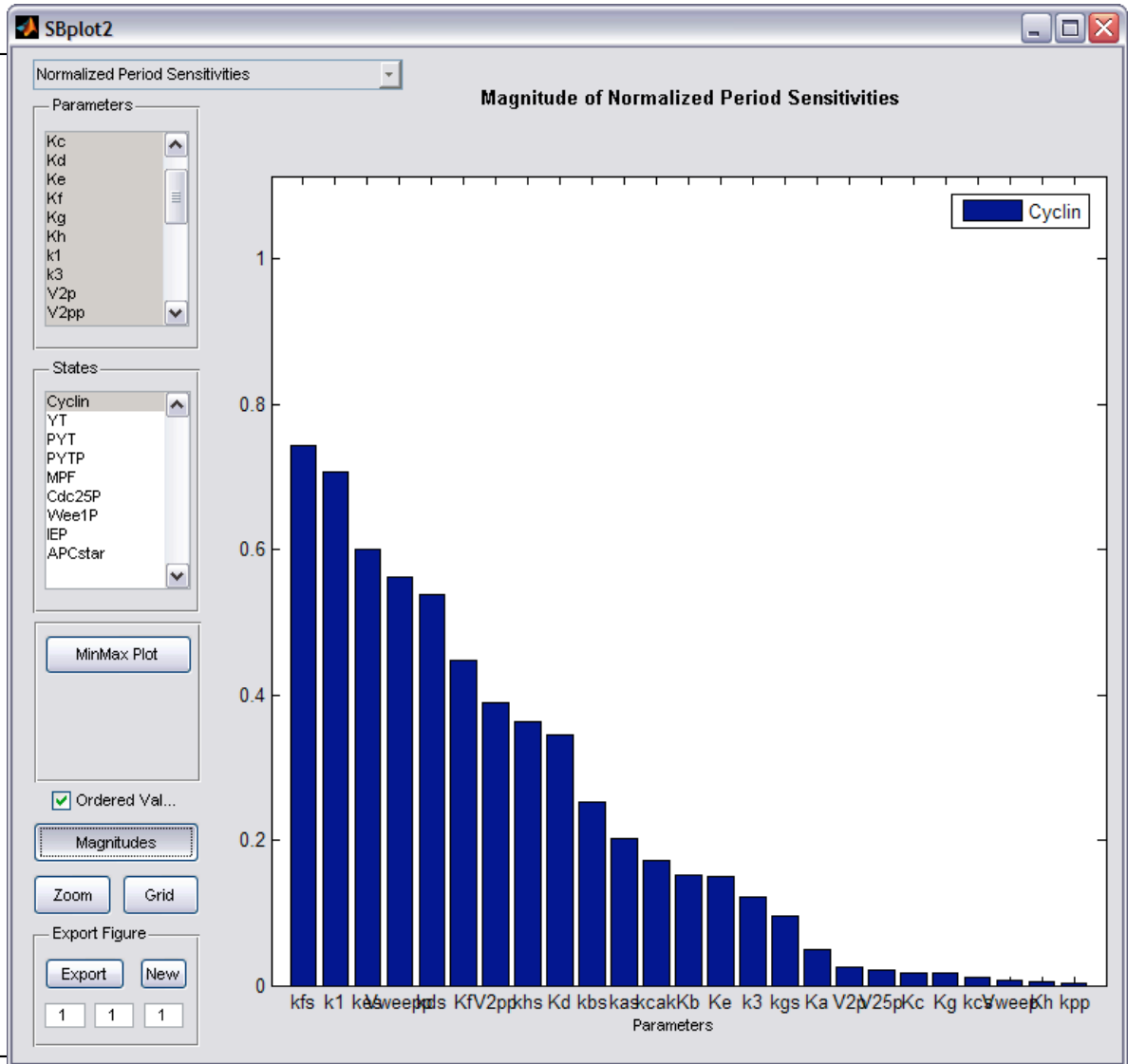
Henning Schmidt



Fraunhofer CHALMERS
Research Centre
Industrial Mathematics

```
>> output =
    SBsensdataosc(model,200)
>> SBsensperiod(output)
```

Parametric sensitivity analysis
(here: period)

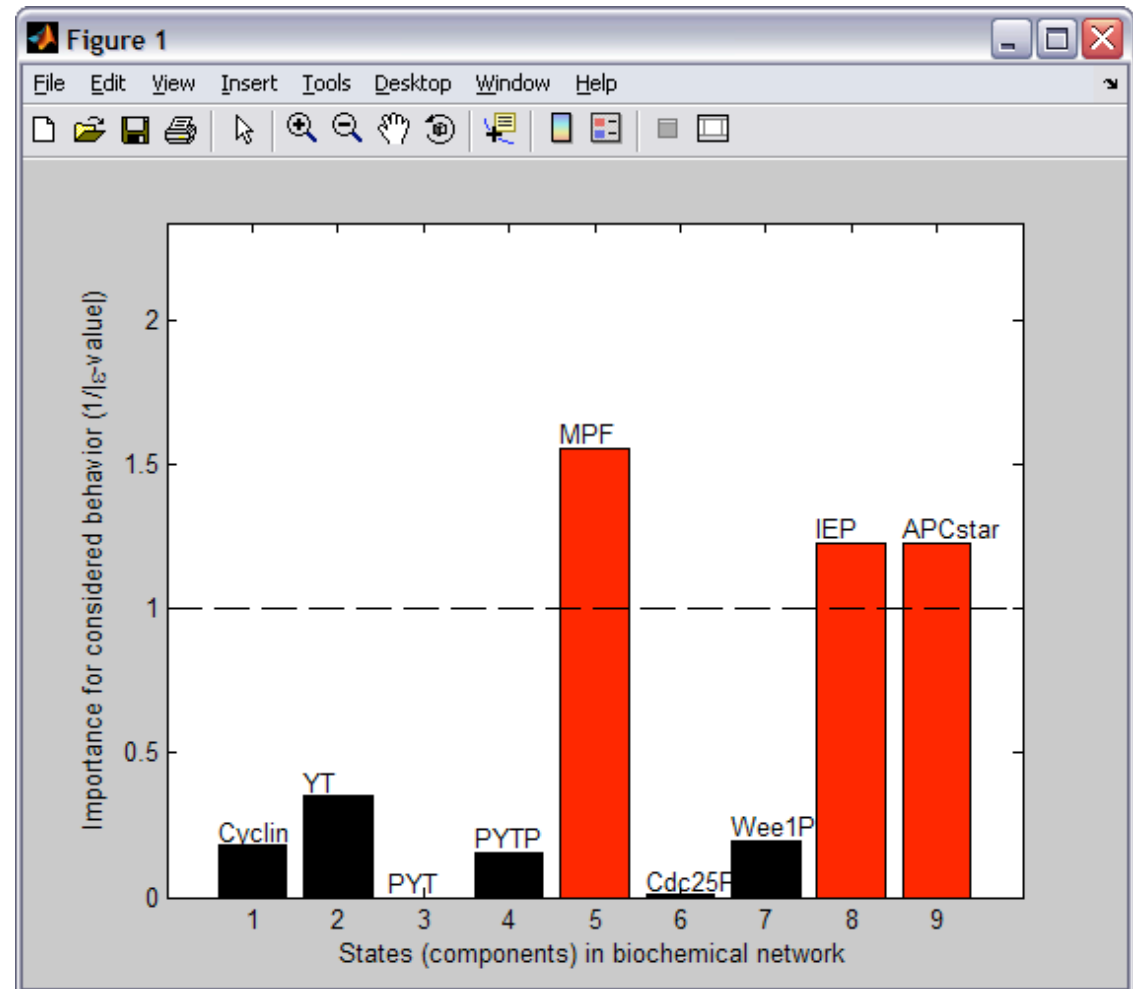


Henning Schmidt



```
>> ss = SBsteadystate(model)
>> SBlocbehavcomp(model,ss)
```

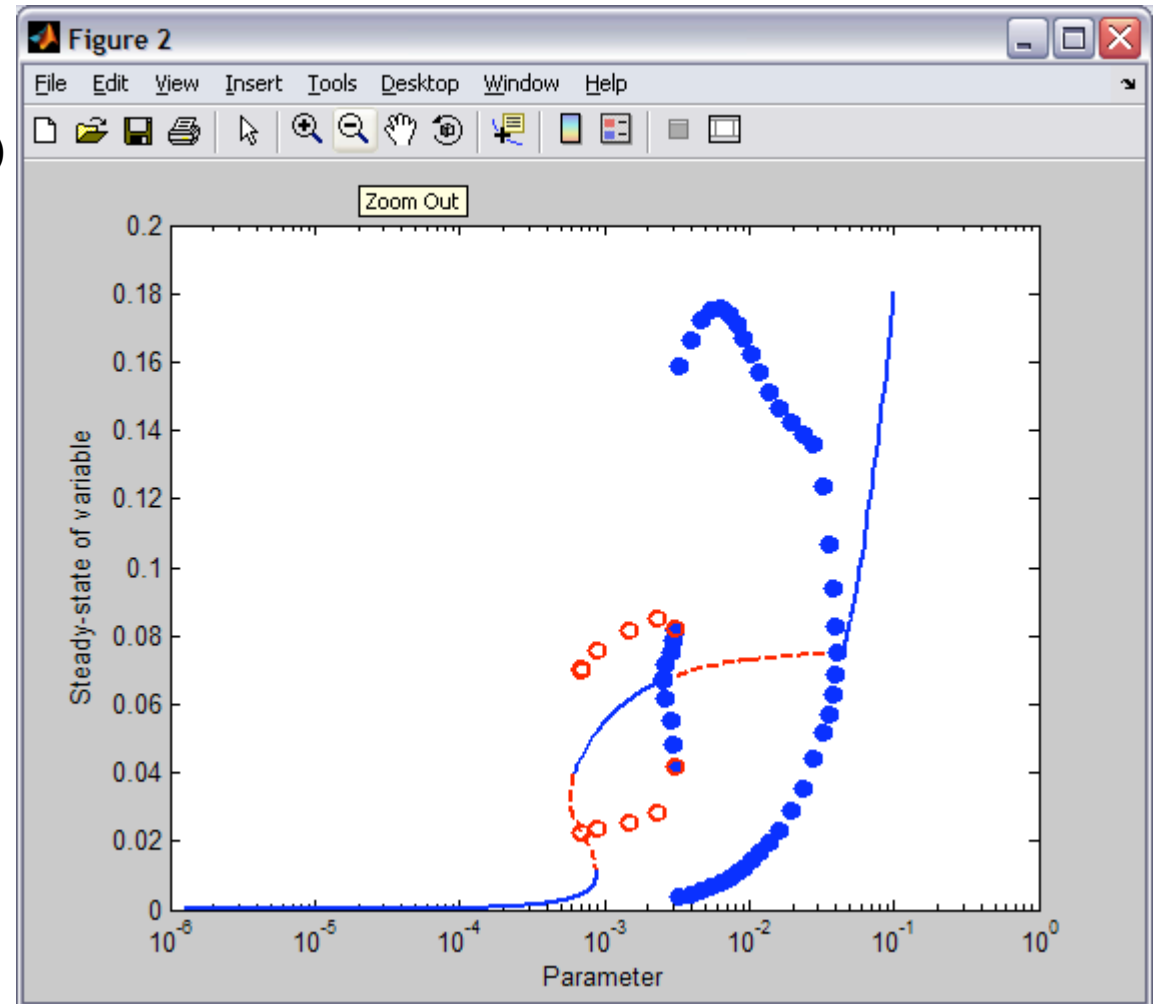
Localization of mechanism leading to oscillations



Henning Schmidt

```
>> SBxppaut(model)
>> SBplotxppaut('bif.dat', 'semilogx')
```

Bifurcation analysis using XPPAUT



Henning Schmidt

Toolbox Documentation

- Full MATLAB type of documentation
 - >> help SBTOOLBOX
 - >> help "functionname"
- Extensive documentation on the webpage ... including examples for each function and getting started examples

Henning Schmidt

Systems Biology Toolbox for MATLAB - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

SBT file:///J:/WRITING/CONFERENCES/ICSB2005/Tutorial/CD_Content/documentation

BANK OPTIMIZATION JOBS WOHNUNG FCC MATLAB TOOLBOX JDeco SBT SBT WORKING Google Scholar LEO

```
>> Jacobian = SBjacobian(sbm,SBinitialconditions(sbm))
```

[output] = SBmoietiesconservations(input)

Usage This function determines algebraic relationships, present in a given model. These algebraic relations can be due to moiety conservations and other conservation laws. The model can be specified as SBmodel or as ODE file.

```
input = [model]
input = [model, stateValues]
input = [model, stateValues, tol]
```

model: SBmodel or ODE file model description.
stateValues: Values of the state variables for which to perform the analysis.
tol: Tolerance for the determination of the number of algebraic relationships in the model. If the method fails than this might be due to a wrong rank computation and in this case the tolerance should be increased. If set, the same tolerance setting is used when determining the indices of the dependent variables.

When no output arguments are specified the function will simply display the found algebraic relations in the MATLAB window.

```
output = [depVarIndex, depVarConstant, depVarFactor, message]
```

depVarIndex: Index of dependent states
depVarConstant: Sum of states taking part in conservation
depVarFactor: Factor for states - non-zero values for the ones that take part in conservation
message: Any message that otherwise is printed to the MATLAB window

Examples An example SBML model is provided in the SBTOOLBOX/examples folder. Change into this folder and type:

```
>> sbm = SBmodel('testsingular');
>> SBmoietiesconservations(sbm)
```

[output] = SBstoichiometry(input)

Usage Determines the stoichiometric matrix for the given model. In order to be able to do that the differential equations for the components have to be

Done



Future Features and where to get the SB Toolbox

- Stochastic Simulation
- Parameter Estimation
- ...
- Scalability
 - right now for models up to 100 states and 200 reaction
 - `strcat('String1','String2')` vs. `char([double('String1') double('String2')])`
more then 100-200 times faster!!!
- The toolbox is freely available and can be found at

<http://www.sbtoolbox.org>

SBML Enabled Software - A user's perspective

- SBML has become a widely supported model description format
- Many software tools are available supporting SBML
- **BUT:**
**For a user new to SBML
it is very hard to find the
tool he needs**
- A classification of tools would be useful

Henning Schmidt



home • contacts • documents • downloads • FAQs •

The Systems Biology Markup Language (SBML) is a computer-readable format for representing **models of bio** metabolic networks, cell-signaling pathways, regulatory networks, and many others.

Internationally Supported and Widely Used

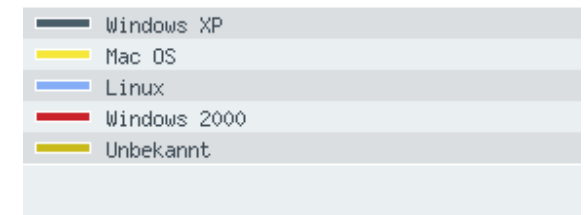
SBML has been evolving since mid-2000 through the efforts of an international group of software developers and **software systems**, including the following (where "*" indicates SBML support in development):

BALSA	COPASI	MesoRD	SBML ODE Solver
BASIS	Cytoscape	MetaboLogica	SBMLeditor
BIOCHAM	DBsolve	MetaFluxNet	SBMLR
BioCharon	Dizzy	MAT2	SBMLSim
bio2SBML	E-CELL	Modesto	SBMLToolbox
BioGrid	ecellJ	Molecuizer	SBW
BioModels	ESS	Monod	SClpath
BioNetGen	FluxAnalyzer	NetBuilder	Sigmoid*
BioPathway Explorer	Fluxor	PANTHER Pathway	SigPath
Bio Sketch Pad	Gepasi	PathArt	SigTran
BioSens	INSILICO discovery	PathScout	SIMBA
BioSPICE Dashboard	Jarnac	PathwayLab	Simpathica
BioSpreadsheet	JDesigner	Pathway Tools	SimWiz
BioTapestry	JigCell	PathwayBuilder	SmartCell
BioUML	JSIM	PaESy	SRS Pathway Editor
BSTLab	JWS Online?	PNK	StochSim
CADLIVE	Karyote*	Reactome	STOCKS
CellDesigner	KEGG2SBML	ProcessDB	TERANODE Suite
Cellerator	Kinsolver*	PROTON	Trellis
CellML2SBML	libSBML	pysbml	Virtual Cell
Cellware	MATLAB SBTtoolbox	PySCeS	WinSCAMP
CL-SBML	MathSBML	runSBML	XPPAUT



Once the user found a tool

- There are very useful tools
- But there are several things to be noted
 - Installation is sometimes too complicated (requires more knowledge than the target user has)
 - **Getting started examples and a good documentation are often missing, leading to deinstallation**
 - **Analysis tools are most often not extensible (by the target user), leading to the search for a different tool or own implementation**
 - **Many tools do not "complain"**



Possible Classification of SBML Tools

- **Targeted user group / task (modelling, simulation, analysis, etc.)**
- Open source / Non open source
 - For open source tools: Profile of the user that is able to extend the tool (relates, e.g., to the programming language)
- Ease of installation (dependent on target user)
- User targeting implementation
- Getting started examples available?
- Detailed documentation with examples available?

Useful Requirements for Software Tools

User targeting implementation of tools

- **Modeling tools:** Biologists, Biochemists, etc.
 - very simple installation process
 - => no extensibility by user necessary
 - (analysis and simulation tools can be present)
- **Analysis tools:** Engineers, Mathematicians, Physicists, ...
 - they expect an analysis tool to be flexible and easily extensible
 - Systems Biology is a very young scientific discipline
 - (modeling can be present)
 - reason for us to use MATLAB
- **Simulation tools:** Most involved user groups
 - depends ...