Tutorial on IQM Tools Lite

General Model Specification, Simulation, etc.

Integrated Solutions for Quantitative Drug Development
From Mechanistic Models to Complex Trial Simulation

Henning Schmidt, IntiQuan GmbH

Tutorial Outline

- **General introduction to the IQM Tools Suite**

- **Model definition and simulation**
  - ODE based models
  - Biochemical reaction equation based models
  - Import and export of models (SBML, etc.)
  - Simple simulation and analysis of models
  - Commenting of models

- **Model analysis**
  - Steady-state analysis and stability
  - Moiety conservations and reduction
  - Parameter sensitivity analysis (steady-state & oscillating systems, MCA, local & global)
  - Localization of complex behaviors

Tutorial Outline

- **Model and dosing descriptions**
  - Setup models to define dosing inputs
  - Dosing description
  - Simulation of dosing scenarios

- **Definition of experiments and measurement data**
  - Excel and CSV measurement representation
  - Experiment descriptions and merging with models
  - Import and export measurements and experiments

Tutorial Info

In large parts you will have the opportunity to get hands-on-experience

```
>> installIQMtoolsInitial
```

Commands shown in these boxes should be entered on the MATLAB command line, during the tutorial

```
********** MODEL NAME
Simple model
********** MODEL STATES
d/dt(A) = -R
d/dt(B) = R
A(0) = 1
B(0) = 0
********** MODEL PARAMETERS
k1 = 0.5
********** MODEL REACTIONS
R = k1*A
```

Text shown in these boxes should be entered where appropriate (will become clear later)

henning.schmidt@intiquan.com

Tutorial Goal: „You should be able to“

- Define models using ODEs and/or biochemical reactions
- Simulate and analyze models
- Simulate models for defined dosing inputs
- Define and simulate experiments on models
- Define measurement data, import, export, plotting

Tutorial Outline

- **General introduction to the IQM Tools Suite**
- Model definition and simulation
- Model analysis
- Model and dosing descriptions
- Definition of experiments and measurement data

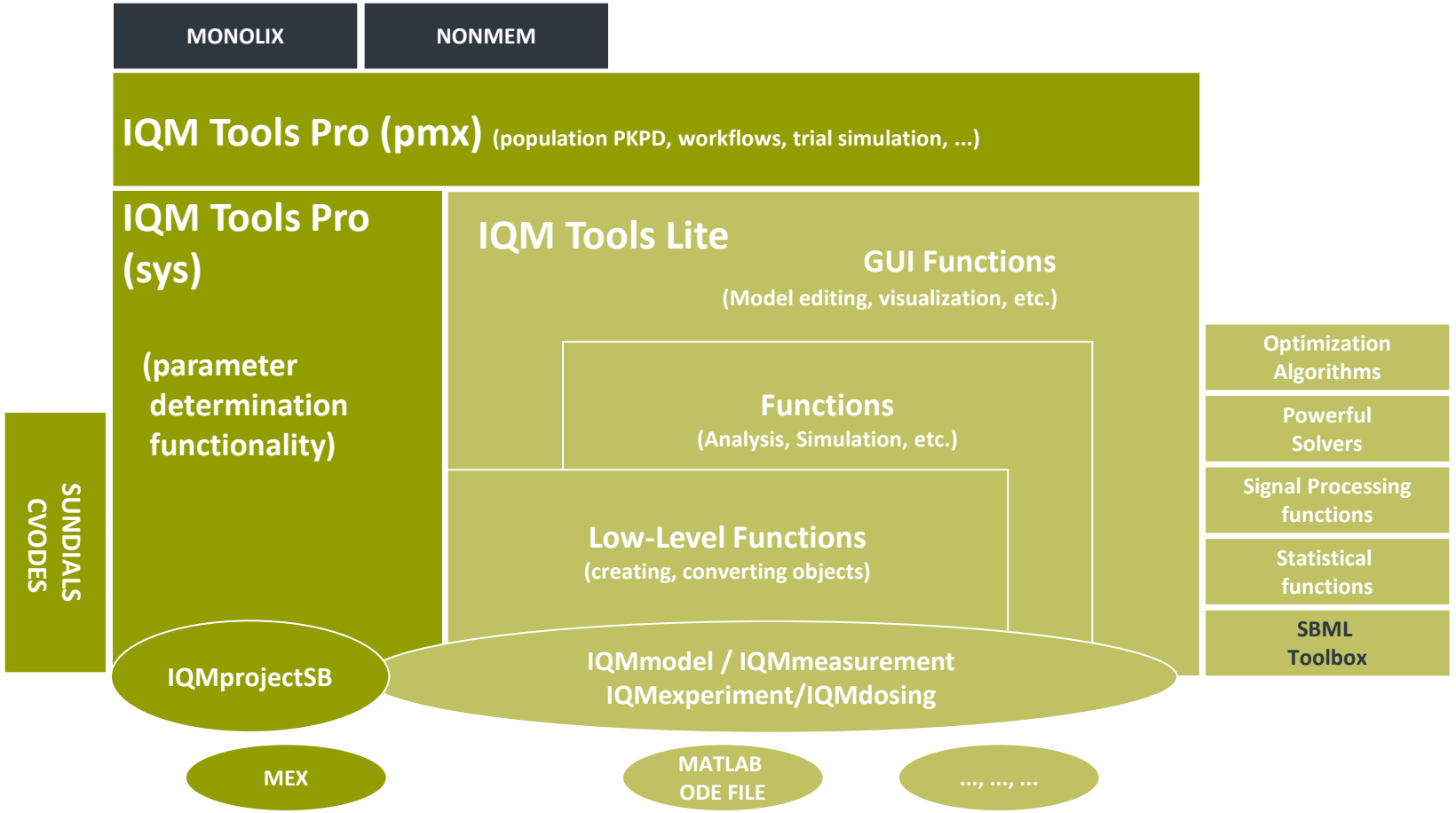
Introduction to the "IQM Tools Suite"

- The IQM Tools Suite is provided freely and as open source
- The IQM Tools Suite consists of two main packages
  - IQM Tools Lite
  - IQM Tools Pro
- The whole is based on MATLAB (www.mathworks.com)

IQM Tools Lite	Model, experiment, measurement & dosing representation, simulation, analysis functions, optimization, signal processing, statistical functions, etc.
IQM Tools Pro	Systems biology/pharmacology and PMX functionality, clinical data analysis, nonlinear mixed effect modeling, clinical trial simulations, high speed simulation through transparent C-code interface

henning.schmidt@intiquan.com

Modular Design of the IQM Tools Suite



henning.schmidt@intiquan.com

IntiQuan

Requirements

- **MATLAB R2013b (or later)**
  - Model reduction methods require the Symbolic Toolbox
  - Availability of the Parallel Toolbox useful for the Pharmacometric functions in IQM Tools Pro

- **Optional 3rd party software**
  - Monolix Version 4.2.3 (or later) (http://www.lixoft.org)
  - NONMEM Version 7.2 (or later) (http://www.iconplc.com)
  - SSm Global Optimization Toolbox (http://www.iim.csic.es/~gingproc/ssmGO.html)
  - SBML Toolbox (http://sbml.org/Software/SBMLToolbox)
    - Only needed for Unix/Linux/Mac
    - For Windows it is included in the distribution of the IQM Tools Lite

Where to get IQM Tool Suite from?

- Free download of IQM Tools Suite available from the IntiQuan webpage

- The IQM Tools Suite is distributed as a ZIP file with both IQM Tools Lite and IQM Tools Pro included

- Unzip this ZIP file on your computer at a location where you want to store the IQM Tools

How to Install the IQM Tool Suite?

- Start MATLAB
- Change into the “IQM Tools Suite” folder
- Read and update custom information in setup files:
  - IQMlite/SETUP_PATHS_TOOLS_IQMLITE.m
  - IQMpro/SETUP_PATHS_TOOLS_IQMPRO.m
- Execute the “installIQMtoolsInitial” script

You need to execute the “**installIQMtoolsInitial**” script once after obtaining a copy of IQM Tools. This will compile required libraries.

After this first installation, you can use the function “**installIQMtools**” to install IQM tools. This needs to be done each time you exit and start MATLAB again. This is on purpose for compliance and reproducibility reasons.

If you do not care about compliance, you might want to consider the use of a startup.m script (see http://www.mathworks.com/help/matlab/ref/startup.html).

Installation of optional 3rd party software

- Please follow the providers instructions when installing optional 3rd party software.

IQM Tools' Documentation

- **MATLAB style help**

```
>> help IQMlite
>> help IQMpro

>> help IQMsimulate
>> help IQMexportCSVdataset

>> doc IQMlite
>> doc IQMsimulate
```

- IQM Tools Tutorials with examples
  - **Part 1: IQM Tools Lite (General Model Specification, Simulation, etc.)**
  - Part 2: IQM Tools Pro (MEX / Systems Biology/Pharmacology Projects)
  - Part 3: IQM Tools Pro (Basic Pharmacometrics)
  - Part 4: IQM Tools Pro (General Dataset Specification and PMX Workflows)
  - Part 5: IQM Tools Pro (Advanced Clinical Trial Simulations)
  - Part 6: IQM Tools Pro (Linking Systems Pharmacology Models to Clinical Data)

- IQM Tools – Workshops
  - Given on demand and on some conferences during a year

henning.schmidt@intiquan.com

Tutorial Outline

- General introduction to the IQM Tools Suite
- **Model definition and simulation**
  - ODE based models
  - Biochemical reaction equation based models
  - Import and export of models (SBML, etc.)
  - Simple simulation and analysis of models
  - Commenting of models
- Model analysis
- Model and dosing descriptions
- Definition of experiments and measurement data

- ODE based models

IQMmodel

henning.schmidt@intiquan.com

A First Model

- Creating a first model

A —R—> B R = k1*A

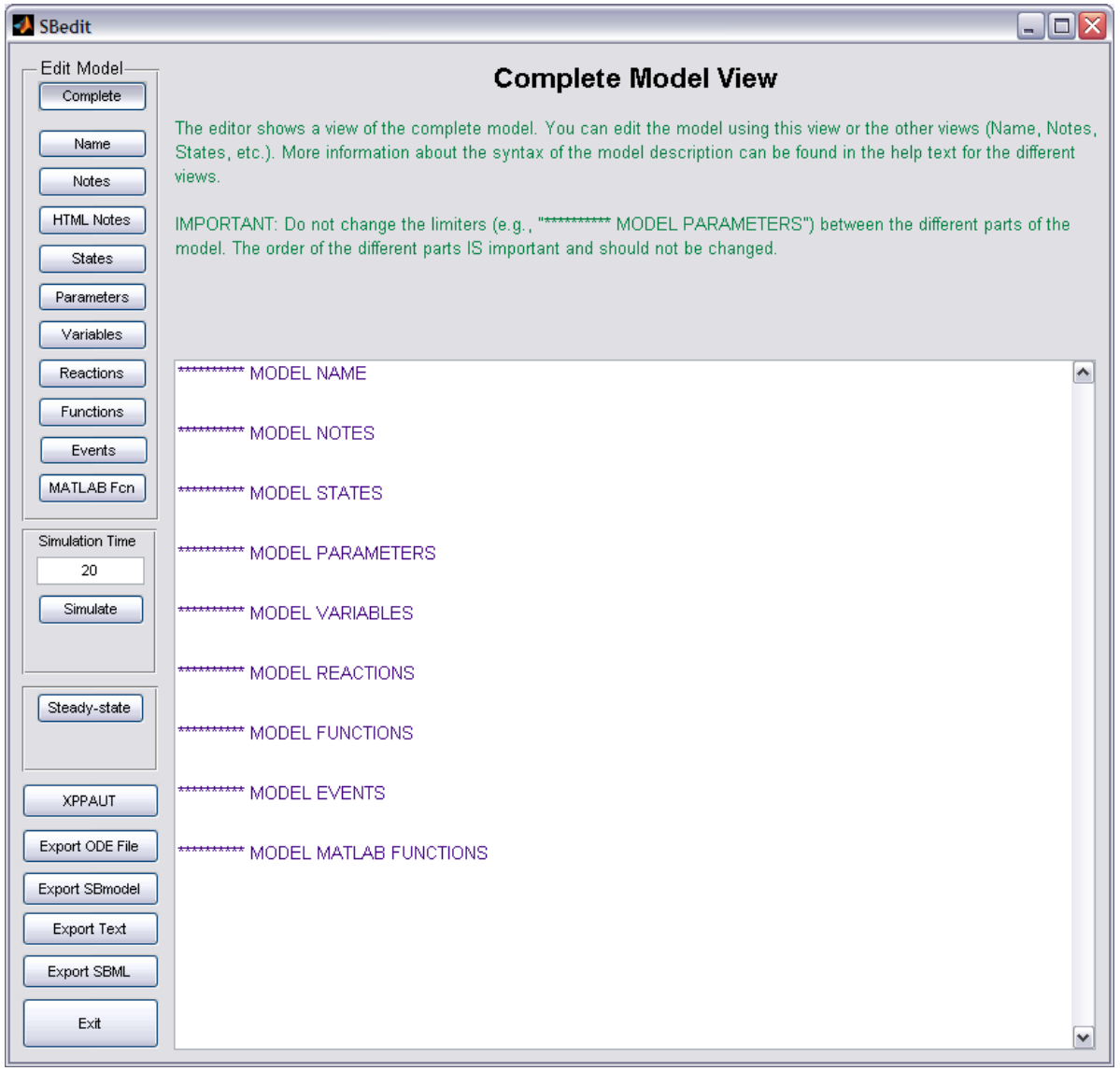
A(0) = 1, B(0) = 0 k1 = 0.5

```
>> model = IQMmodel()          % creating empty model
>> model = IQMedit(model)      % editing the model
```

```
>> model = IQMedit()           % starting the editor with an empty model
```

IQMedit

- Graphical User Interface allowing to edit models
- Click on the ”Edit Model” buttons
- **Each view provides a help text about the model syntax**
- The limiters are important – do not change them



henning.schmidt@intiquan.com

IntiQuan

Simple ODE Model (States, Parameters, Reactions)

- Enter the following information (keep all non-used limiters)

```
********** MODEL NAME
Simple model
********** MODEL STATES
d/dt(A) = -R
d/dt(B) = R
A(0) = 1
B(0) = 0
********** MODEL PARAMETERS
k1 = 0.5
********** MODEL REACTIONS
R = k1*A
```

A –R–> B

A(0) = 1
B(0) = 0

k1 = 0.5

R = k1*A

Click ”Simulate”

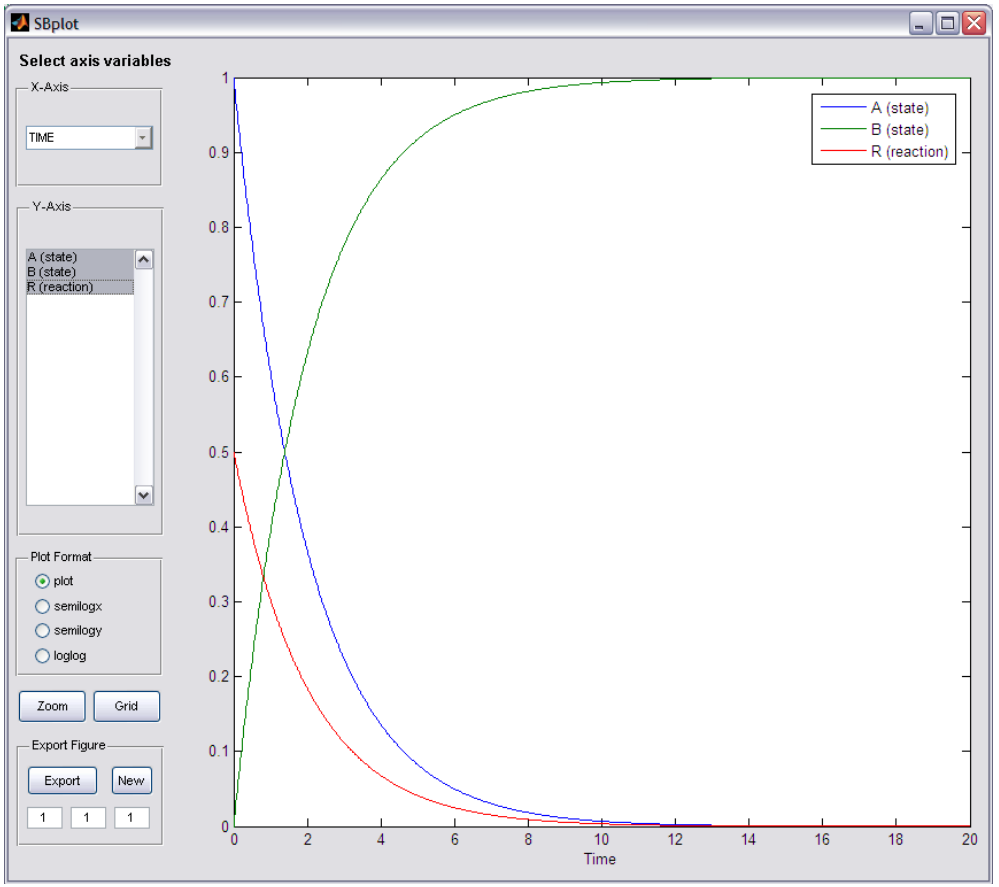
henning.schmidt@intiquan.com

Simulate Simple Model

- **IQMplot**

  Graphical User Interface
  for the display of time series
  type of data

- Play around with the features
  of the plotting window



henning.schmidt@intiquan.com

IntiQuan

Model Functions

- Model functions can be used to define often recurring calculations

```
********** MODEL STATES
d/dt(A) = -R
d/dt(B) = R
A(0) = 1
B(0) = 0
********** MODEL PARAMETERS
k1 = 0.5
********** MODEL REACTIONS
R = k1*f(A)
********** MODEL FUNCTIONS
f(x) = x^3
```

A –R–> B

A(0) = 1
B(0) = 0

k1 = 0.5

R = k1***f(A)**

f(x) = x^3

Click ”Simulate”

henning.schmidt@intiquan.com

IntiQuan

Model Events

- Model events can be used to define discrete state events

```
********** MODEL STATES
d/dt(A) = -R
d/dt(B) = R
A(0) = 1
B(0) = 0
********** MODEL PARAMETERS
k1 = 0.5
********** MODEL REACTIONS
R = k1*f(A)
********** MODEL FUNCTIONS
f(x) = x^3
```

```
********** MODEL EVENTS
event = lt(A,0.3), A, 1, B, 0
```

If A becomes less than 0.3 then reset A to 1 and B to 0

Click ”Simulate”

henning.schmidt@intiquan.com

Model Variables

- Simulation of these two models gives exactly the same results

```
********** MODEL NAME
Simple model
********** MODEL STATES
d/dt(A) = -R
d/dt(B) = R
A(0) = 1
B(0) = 0
********** MODEL PARAMETERS
k1 = 0.5
********** MODEL REACTIONS
R = k1*A
```

```
********** MODEL NAME
Simple model
********** MODEL STATES
d/dt(A) = -R
d/dt(B) = R
A(0) = 1
B(0) = 0
********** MODEL PARAMETERS
k1 = 0.5
********** MODEL VARIABLES
R = k1*A
```

So what is the difference?

Model Reactions vs. Model Variables

- Reaction rates should be defined under MODEL REACTIONS
- Variables for intermediate calculations (or for monitoring) should be defined under MODEL VARIABLES
- The only cases where it matters for the toolbox is when
  - Determining the stoichiometric matrix
  - Exporting the model to SBML

Command Line Simulation

- Clicking ”Exit” in the IQMedit GUI returns the model to the workspace

```
>> model = IQMedit()
IQMmodel
========
Name:  Simple Model
Number States:                2
Number Variables:             0
Number Parameters:            1
Number Reactions:             1
Number Functions:             1
Number Events:                1

>> IQMsimulate(model,50)      % simulation over 50 time units
```

Command Line Simulation, continued

```
>> output = IQMsimulate(model,50)
output =
              time: [50x1 double]
            states: {'A'  'B'}
       statevalues: [50x2 double]
         variables: {}
    variablevalues: []
         reactions: {'R'}
    reactionvalues: [50x1 double]


>> help IQMsimulate
```

- Array specification of TEXT IQMmodels

Array specification of TEXT IQMmodels

- Consider a system that can be described by the following set of differential equations:

$$d/dt\ x[n] = kon*x[1]*x[n-1] + koff*x[n+1] - (kon*x[1]+koff)*x[n],\ n=1\ldots N$$

  - Example:
    - Model for the length distribution of actin filaments, Edelstein-Keshet, Mathematical Biology, 1998

- If N is large it is very messy to type all these differential equations in by hand. Here, the array type model specification of IQMmodels helps in setting up such equations

Array specification of TEXT IQMmodels

Example

```
********** MODEL STATES
d/dt(x<n,0>) = 0
d/dt(x<n,1:N>)= kon<n>*x<1>*(x<n-1>) + koff*x<n+1> - (kon*x<1>+koff)*x<n> + R<n>
d/dt(x<n,N+1>) = 0

x<n,[1:2, 4:N]>(0) = n*N
x<n,3>(0) = 100
x<n,N+1>(0) = 0

********** MODEL PARAMETERS
N = 10
koff = 2
kon = 0.01

********** MODEL VARIABLES
kon<k,1:N> = kon*sqrt(k)

SUMEXAMPLE1 = 5 + arraysumIQM(n^2/(x<n,1:N> + n/N))+ 56
SUMEXAMPLE2 = arraysumIQM(x<n,1:N>)
SUMEXAMPLE3 = arraysumIQM(x<n,[1,3,5]>)
SUMEXAMPLE4 = arraysumIQM(x<n,[1:2:N-1]>)

********** MODEL REACTIONS
R<n,1:N> = koff*x<n+1> - kon<n>*x<n>*N*n
```

Simple to use format

ODEs, variables, and reactions can be defined by arrays

Negative indices possible

arraysumIQM is a powerful and general construct to determine sums etc. over desired array elements

During import of such a model the array notation is expanded

...

henning.schmidt@intiquan.com

Array specification of TEXT IQMmodels

Example

- The expansion of the array notation is best understood by running an example (change into the „Example Files“ folder):

```
>> edit array.txt                    % opens the model file as seen in the previous slide
>> model = IQMmodel('array.txt')     % import the model
>> IQMedit(model)                    % look at the model and compare to array.txt
```

- Now change „**N**“ in the array.txt file to **N=100** and save the file

```
>> edit array.txt                    % opens the model file as seen in the previous slide
>> model = IQMmodel('array.txt')     % import the model
>> IQMedit(model)                    % look at the model and compare to array.txt
```

- **You see the difference?**

Array specification of TEXT IQMmodels

Example

- Array-type notation and standard ODE specification can be combined

- More information about the array notation can be found in the **`array_notation_explained.txt`** model in the „**`Example Files`**“ folder

henning.schmidt@intiquan.com

- Biochemical reaction equation based models

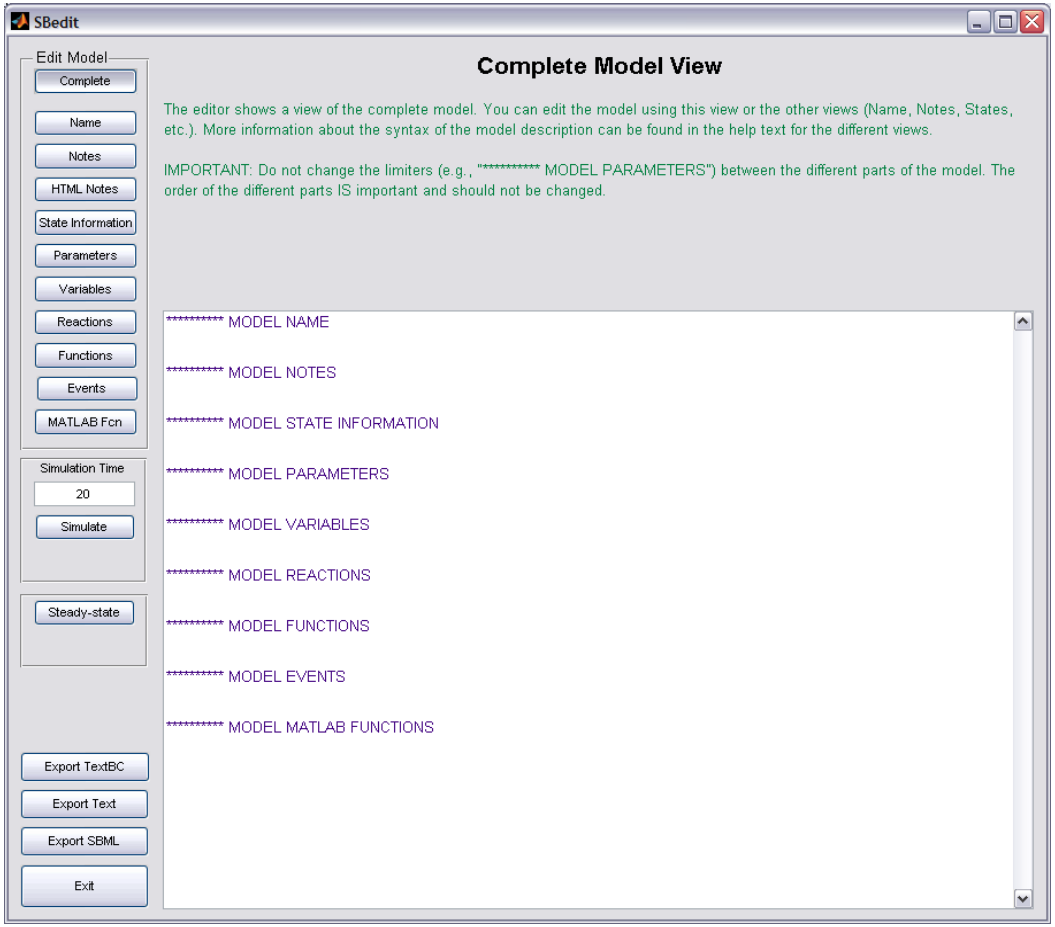
IQMmodel

henning.schmidt@intiquan.com

Modeling Using BioChemical Reaction Equations

```
>> model = IQMeditBC()
```

- **IQMeditBC**
  Graphical User Interface allowing to edit models based on reaction equations
- Click on the ”Edit Model” buttons
- Each view provides a help text about the model syntax
- The limiters are important – do not change them



henning.schmidt@intiquan.com

IntiQuan

Simple Model Again ...

Enter the following information (keep all non used limiters)

```
********** MODEL NAME
Simple model
********** MODEL STATE INFORMATION
A(0) = 1
B(0) = 0
********** MODEL PARAMETERS
k1 = 0.5
********** MODEL REACTIONS
A => B : R
    vf = k1*A
```

A –R–> B

A(0) = 1
B(0) = 0

k1 = 0.5

R = k1*A

Click ”Simulate”

henning.schmidt@intiquan.com

More Complex Model

```
********** MODEL STATE INFORMATION
B(0) = 1
********** MODEL PARAMETERS
k1 = 0.5
********** MODEL REACTIONS
A+B => 2*C : R1
       vf = k1*A*B
B <=> A+D : R2
       vf = 5.1*B
       vr = 3*A*D
2*A => A2 : R3
       vf = 2.7*A^2
```

- Click ”Simulate”
- Click ”Exit”

```
>> model = IQMedit(model)
```

=> Interchangeable formats

henning.schmidt@intiquan.com

- Import/Export of models
  - Textual description
  - SBML

Export a Model to the Textual Descriptions

```
>> IQMedit(model)   % the variable “model” contains a model from the previous work
```

- To export a model to the ODE / Biochemical textual description using IQMedit or IQMeditBC click **”Export Text”** / **”Export TextBC”**
  - Choose a file name – here: **textmodel.txt** / **textmodel.txtbc**
  - ODE textmodel files are required to have the extension **.txt**
  - Biochemical textmodel files are required to have the extension **.txtbc**
  - Click ”Exit”

- Export by command line commands

```
>> IQMcreateTEXTfile(model, 'textmodel')       % saves model as “textmodel.txt”

>> IQMcreateTEXTBCfile(model, 'textmodel')     % saves model as “textmodel.txtbc”
```

Import a Textual Description to an IQMmodel

- Open the TEXT(BC) model file with the MATLAB editor to see its syntax

```
>> edit textmodel.txt

>> edit textmodel.txtbc
```

- To import the TEXT(BC) model and convert it to an IQMmodel write

```
>> model = IQMmodel('textmodel.txt')    % the IQMmodel command loads a model in txt

>> model = IQMmodel('textmodel.txtbc')  % and in txtbc format. The extensions are important!
```

Import of SBML Models

- SBML Level Version 1,2,3 & 4 models can be imported
- **The SBML Toolbox needs to be present**
- Change into the "**Example Files**" folder

```
>> model = IQMmodel('CellCycle.xml')    % IQMmodel additionally also import SBML models
    IQMmodel
    ========
    Name: CellCycle
    Number States:              13
    Number Variables:           2
    Number Parameters:          41
    Number Reactions:           23
    Number Functions:           0

>> IQMedit(model)
```

Increase simulation time to 400 and click "Simulate"

henning.schmidt@intiquan.com

Import of Incompletely Defined SBML Models

- Per default IQM Tools requires an SBML model to be completely defined before it is correctly imported

- To allow the user to import incompletely defined models the following optional call to IQMmodel exists:

```
% File located in the “Example Files” folder
>> model = IQMmodel('SBMLfileIncomplete.xml',1);
>> IQMedit(model)
```

  - **`IQMmodel`** then does not require that an SBML model is fully defined in terms of parameter values, rate equations, kinetic parameters, etc.
  - **However, when trying to „Exit“ IQMedit or IQMeditBC model correctness is checked. To still exit on Windows click the red cross in the upper right corner of the window. On Unix/Linux/Mac also click the corresponding symbol ...**

Import of SBML Models

Name to ID Conversion

- 'Name' to 'id' conversion
  - E.g.: Not all models in Biomodels.net have informative IDs, and CellDesigner or Simbiology choose the ids of certain elements (e.g., species and reactions) automatically and the user only can choose names that need not be unique

```
********** 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)
```

  - In IQM Tools the ids are used as names for states, variables, etc. (since they are unique)
  - => an optional call to IQMmodel (see box below) allows to use the SBML names instead of the IDs, but making them unique by numerical extensions

```
>> model = IQMmodel('SBMLfileIncomplete.xml',1,1);
```

  - More information, see the help text:

```
>> help IQMmodel
```

henning.schmidt@intiquan.com

Export of SBML Models

- Export only to SBML Level 2 Version 1
- IQMmodels do not necessarily contain all needed information for export
  - Additional information is required

```
>> help IQMexportSBML          % for more information on the additional information
```

- Location of additional information (in IQMmodel internal data structure)

states.type	**algebraic.type**
states.compartment	**algebraic.compartment**
states.unittype	**algebraic.unittype**
parameters.type	**variables.type**
parameters.compartment	**variables.compartment**
parameters.unittype	**variables.unittype**

***.type:**	'isSpecie'	'isParameter'	'isCompartment'
***.compartment:**	compartment	-	outside compartment
***.unittype:**	'amount' or 'concentration'	-	-

henning.schmidt@intiquan.com

Export of SBML Models

Additional Information in the TEXT Description

- Additional information can be added in the text / textbc format

```
>> help IQMedit
>> help IQMeditBC
```

- After SBML import the additional information is already present

```
********** MODEL NAME
example

********** MODEL NOTES


********** MODEL STATES
d/dt(s1) = -re1 {isSpecie:cytosol:amount} % comment
d/dt(s2) = +re1 {isSpecie:cytosol:amount}
d/dt(s3) = -re2+re4 {isSpecie:cytosol:amount}
d/dt(s4) = +re2-re5 {isSpecie:cytosol:amount}
d/dt(s5) = (-re3)/nucleus {isSpecie:nucleus:concentration}
d/dt(s6) = +re3 {isSpecie:nucleus:amount}

s1(0) = 1

********** MODEL PARAMETERS
s7 = 1 {isParameter} % comment
s8 = 1 {isParameter}
cytosol = 1 {isCompartment:}
nucleus = 0.1 {isCompartment:cytosol}


********** MODEL VARIABLES


********** MODEL REACTIONS
re1 = 3 * s1 - 2 * s2 {reversible} % comment
re2 = s3
re3 = s4 * (s5 - s6) {reversible}
re4 = 1
re5 = s4
```

henning.schmidt@intiquan.com

Export of SBML Models

Automatic Determination of Additional Information

- Even if no additional information is given, the toolbox is under certain conditions able to determine the correct information.

novaktyson1.txt

```
d/dt(Cyclin) = R1-R2-R3
d/dt(YT) = R4-R5-R6-R7+R8+R3
d/dt(PYT) = R5-R8-R9-R10+R11
d/dt(PYTP) = R12-R11-R13-R14+R9
d/dt(MPF) = R6-R4-R12-R15+R13
d/dt(Cdc25P) = R16
d/dt(Wee1P) = R17
d/dt(IEP) = R18
d/dt(APCstar) = R19
```

=> Species in Amount units

novaktyson2.txt

```
d/dt(Cyclin) = (R1-R2-R3)/compartment
d/dt(YT) = (R4-R5-R6-R7+R8+R3)/compartment
d/dt(PYT) = (R5-R8-R9-R10+R11)/compartment
d/dt(PYTP) = (R12-R11-R13-R14+R9)/compartment
d/dt(MPF) = (R6-R4-R12-R15+R13)/compartment
d/dt(Cdc25P) = (R16)/compartment
d/dt(Wee1P) = (R17)/compartment
d/dt(IEP) = (R18)/compartment
d/dt(APCstar) = (R19)/compartment
```

=> Species in Concentration units

- Remember: SBML assumes reaction rates defined in amount/time

- Here it is important that the elements on the RHS are defined under the „**MODEL Reactions**“ header

henning.schmidt@intiquan.com

Export of SBML Models

Automatic Determination of Additional Information

- Example

```
>> model = IQMmodel('novaktyson1.txt')  % file located in ”example files” folder
>> IQMexportSBML(model)                 % choose sbmlmodel1 as name for the file

>> model = IQMmodel('novaktyson2.txt’)  % file located in ”example files” folder
>> IQMexportSBML(model)                 % choose sbmlmodel2 as name for the file
```

- Have a look at both files (novaktyson1.txt and ...2.txt)

- Have a look at the exported SBML files
  - => species, reactions, compartments and unittypes are correctly determined

- This does not work for all possible model definitions, and thus the manually added information will override the automatically generated one

henning.schmidt@intiquan.com

- Simple Simulation

Deterministic Simulation

- Serves also as example for using the Cell-Mode

- Change into the „**Example Files**“ folder

```
>> edit simpleSimulation
```

1. Read the documentation in the opened file
2. Execute the cells sequentially by pressing „**Ctrl+Enter**“

Stochastic Simulation

- Serves also as example for using the Cell-Mode

- Change into the „**`Example Files`**“ folder

```
>> edit stochasticSimulation
```

1. Read the documentation in the opened file
2. Execute the cells sequentially by pressing „**`Ctrl+Enter`**“

henning.schmidt@intiquan.com

- Commenting of models

WHY COMMENTS AND DOCUMENTATION?

- Models **without** comments and documentation **are even less useful** than software without documentation

Commenting a Model

- The more complex a model the better it is if there are comments included

- The IQMmodels allow 3 types of comments
  - Information in the „MODEL Notes“
  - Optional comment on each state, variable, parameter, etc.
  - Furthermore, .txt and .txtbc files allow to use whole lines for commenting if prefixed with the „%“ character.

- Example:

```
********** MODEL REACTIONS
% This is just a comment about
% the following reaction:
A+B => 2*C : R1  % comment
       vf = k1*A*B
```

henning.schmidt@intiquan.com

Commenting a Model

Example

```
********** MODEL NOTES
Unified phototransduction model from Hamer et al., Visual Neuroscience 22, 417-436
This model here uses only 6 phosphorylation states

********** MODEL STATE INFORMATION
% ODEs for the "back-end" model
d/dt(g) = alfamax/(1+power((c/Kc),m)) - (betadark + betasub*PDE_a)*g

% Initial Conditions
R(0) = 3.6e9                % (#) Rhodopsin unactivated (same value as parameter Rtot)
PDE(0) = 2.67e7             % (#) PDE (same value as parameter PDEtot)

********** MODEL PARAMETERS
% Total concentrations or numbers
Rtot = 3.6e9                % (#) total amount of Rhodopsin (same value as initial condition for R)

% R_n bound to RK  pre=>post
kRK3_ATP = 400                      % here it is already multiplied by ATP, see paper xyz

% Unbinding of R_n and RK
kRK4 = 20                           % value measured in paper WXY

********** MODEL REACTIONS
% Inactivation Pathways
% ====================
% R_n (activated Rhodopsin n-times phosphorylated) binding to RK
% RK is in number of molecules not in concentration. (Different to paper but
% taken into account by having scaled kRK1_n with RK in numbers.
R_0 + RK <=> R_0_RK_pre : v_A1a_0  % Comment about reaction
    vf = kRK1_0 * RK * R_0
    vr = kRK2 * R_0_RK_pre
```

henning.schmidt@intiquan.com

Commenting a Model

- Using MATLAB syntax highlighting comments are displayed in a different color
- Comments make the model more readable
- Modeling process, assumptions, references, etc. **well documented**

- Important:
  - Note that during import of a .txt or .txtbc model the comments shown in **blue** (lines) are removed from the model
  - However, when working directly on .txt and .txtbc models using the MATLAB editor the comments stay

Commenting a Model

Using Named Kinetic Rate Laws

- kin_allosteric_inihib_empirical_rev
- kin_allosteric_inihib_mwc_irr
- kin_catalytic_activation_irr
- kin_catalytic_activation_rev
- kin_comp_inihib_irr
- kin_comp_inihib_rev
- kin_constantflux
- kin_degradation
- kin_hill_1_modifier_rev
- kin_hill_2_modifiers_rev
- kin_hill_cooperativity_irr
- **kin_hill_rev**
- kin_hyperbolic_modifier_irr
- kin_hyperbolic_modifier_rev
- kin_iso_uni_uni_rev
- kin_mass_action_irr
- kin_mass_action_rev
- kin_michaelis_menten_irr
- kin_michaelis_menten_rev
- kin_mixed_activation_irr
- kin_mixed_activation_rev
- kin_mixed_inihib_irr
- kin_mixed_inihib_rev
- kin_noncomp_inihib_irr
- kin_noncomp_inihib_rev
- kin_ordered_bi_bi_rev
- kin_ordered_bi_uni_rev
- kin_ordered_uni_bi_rev
- kin_ping_pong_bi_bi_rev
- kin_specific_activation_irr
- kin_specific_activation_rev
- kin_substrate_activation_irr
- kin_substrate_inihib_irr
- kin_substrate_inihib_rev
- kin_uncomp_inihib_irr
- kin_uncomp_inihib_rev
- kin_uni_uni_rev

Instead of

```
********** MODEL REACTIONS
R = Vf*substrate/Shalve*(1-product/(substrate*Keq))*
    (substrate/Shalve+product/Phalve)^(h-1) /
    ( 1+(substrate/Shalve + product/Phalve)^h )
```

You can write

```
********** MODEL REACTIONS
R = kin_hill_rev(Vf,substrate,Shalve,product,Keq,Phalve,h)
```

```
>> help IQMlite
```

37 inbuild rate laws
More rate laws can easily be added

Readability of models is improved

henning.schmidt@intiquan.com

IntiQuan

Tutorial Outline

- General introduction to the IQM Tools Suite
- Model definition and simulation
- **Model analysis**
  - Steady-state analysis and stability
  - Moiety conservations and reduction
  - Parameter sensitivity analysis (steady-state & oscillating systems, MCA, local & global)
  - Localization of complex behaviors
- Model and dosing descriptions
- Definition of experiments and measurement data

henning.schmidt@intiquan.com

- Steady-State Analysis and Stability

Steady-state Determination

- Change into the „**Example Files**“ folder
- Determination of the steady-state

```
>> model = IQMmodel('CellCycle.txt')
>> IQMsteadystate(model)

Steady state could not be found.
Try different options and/or a different starting guess.

>> IQMinitialconditions(model)  % almost all zero
```

- Starting conditions are important
- One possibility to get starting conditions is to simulate a short time

```
>> output = IQMsimulate(model,20)
>> ss = IQMsteadystate(model,output.statevalues(end,:))
```

- Another possibility is the use of IQMedit to set new initial conditions

Jacobian and Stability

- Determination of the Jacobian

```
>> Jacobian = IQMjacobian(model,ss)
```

- Determination of stability by considering the Jacobian eigenvalues

```
>> eig(Jacobian)
          0.0413 + 0.1528i
          0.0413 - 0.1528i
          ...
```

- Two complex conjugated eigenvalues with positive real part
  => the considered steady-state is unstable and the system is oscillating around it

- Moiety Conservations and Reduction

Moiety Conservations

- Determination of moiety conservations

```
>> model = IQMmodel('CellCycle.txt')
>> IQMmoietyconservations(model)

Cdc25P = 1 - 1 Cdc25
Wee1 = 1 - 1 Wee1P
APC_ = 1 - 1 APC
IEP = 1 - 1 IE
```

- Moiety conservations (linear dependencies between ODEs) are not allowed to be present in a system, e.g., for bifurcation analysis

- Moiety conservations present => The model is singular

Simple Model Reduction

- Singular model -> Non singular model
- Replacement of linear dependent state variables by static variables

```
>> model = IQMmodel('CellCycle.txt')
>> modelred = IQMreducemodel(model)

>> IQMedit(modelred)
```

- The `modelred` model has 4 states less and 4 variables more
- The simulation results are identical

Local Parameter Sensitivity Analysis

- Steady-state sensitivities (states, reactions)
- Period and amplitude sensitivities for oscillating systems (states, reactions)
- Metabolic Control Analysis

- Two step approach (except for MCA)
  - 1) Generation of data for sensitivity analysis
  - 2) Determining of sensitivities and display of sensitivity data

```
>> help IQMsensdatastat
>> help IQMsensdataosc
>> help IQMmca
```

```
>> help IQMsensdataoscevents
```

henning.schmidt@intiquan.com

Local Parameter Sensitivity Analysis

Steady-State

```
>> model = IQMmodel('CellCycleRed.txt')
>> output = IQMsensdatastat(model)
    ...
          model: [1x1 IQMmodel]
         states: {9x1 cell}
         xssnom: [9x1 double]
        xsspert: {1x26 cell}
     reactions: {19x1 cell}
         rssnom: [19x1 double]
        rsspert: {1x26 cell}
    parameters: {26x1 cell}
     nomvalues: [1x26 double]
      pertSize: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
        absRel: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]

>> IQMsensstat(output)
```

henning.schmidt@intiquan.com

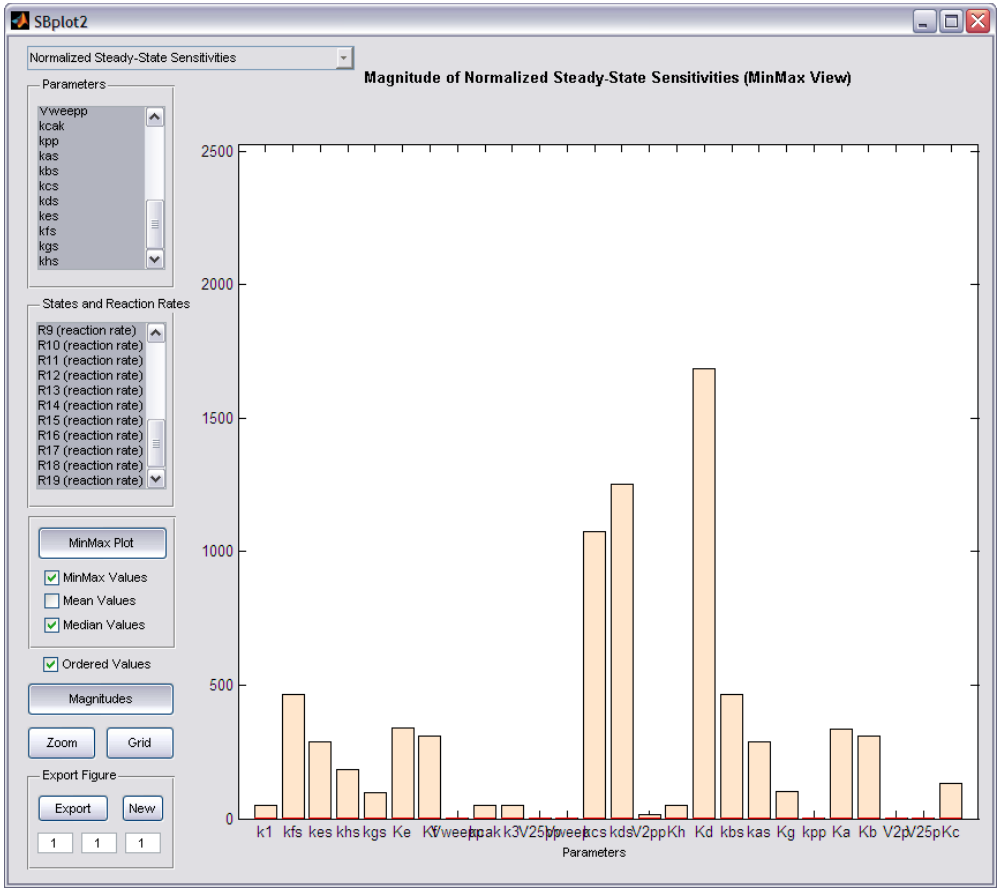
Local Parameter Sensitivity Analysis

Steady-State

- **IQMplot2**

  Graphical User Interface allowing to display block diagram type of data

- Play around with the GUI to get a feeling for its use



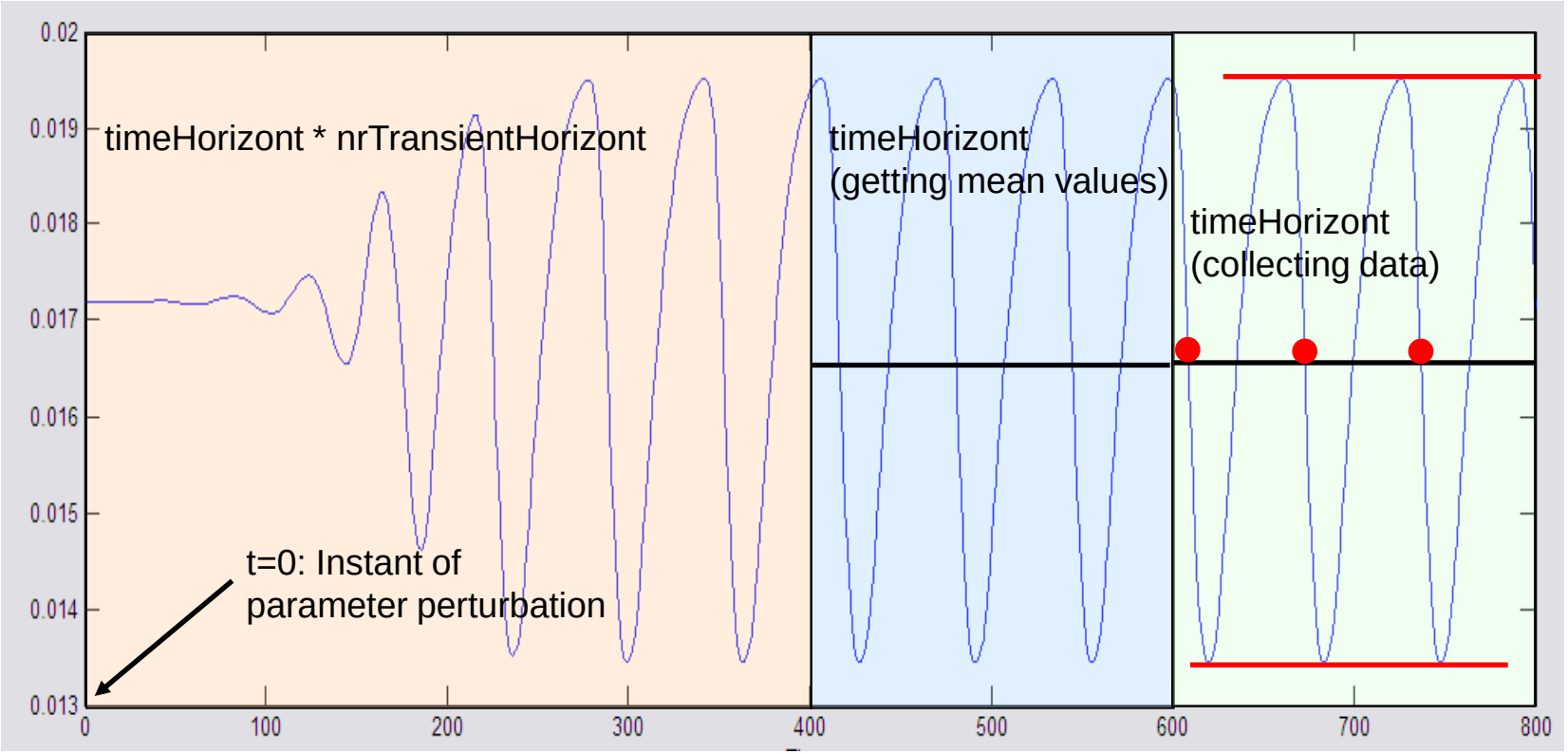
henning.schmidt@intiquan.com

IntiQuan

Local Parameter Sensitivity Analysis

Oscillating Systems (Period and Amplitude)

- How is it implemented?
- **`output = IQMsensdataosc(model,timeData)`**
  `timeData = [timeHorizont nrTransientHorizont]`



Local Parameter Sensitivity Analysis

Oscillating Systems (Period and Amplitude)

- Simulate the model to determine the time needed for the transients to die out and the time needed to have several oscillations within the horizont

```
>> model = IQMmodel('CellCycleRed.txt')
>> output = IQMsensdataosc(model,[200 2])
output =
         model: [1x1 IQMmodel]
          time: [1x1001 double]
         tenom: {1x9 cell}
        tepert: {1x26 cell}
        states: {9x1 cell}
          xnom: [1001x9 double]
         xpert: {1x26 cell}
    parameters: {26x1 cell}
     nomvalues: [1x26 double]
      pertSize: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
        absRel: [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

Local Parameter Sensitivity Analysis

Oscillating Systems (Period and Amplitude)

- Determining and displaying sensitivities

```
>> IQMsensperiod(output)    % display parameter sensitivities for the oscillation period
>> IQMsensamplitude(output) % display parameter sensitivities for the oscillation amplitude
```

- Accessing sensitivity data

```
>> sensdata = IQMsensperiod(output)
sensdata =
               S: [9x26 double]
         statesS: {9x1 cell}
     parametersS: {26x1 cell}
              Sn: [9x26 double]
        statesSn: {9x1 cell}
    parametersSn: {26x1 cell}
>> sensdata = IQMsensamplitude(output)
...
```

Metabolic Control Analysis

- MCA is a special case of sensitivity analysis
- Function: **IQMmca**
  - Flux Control Coefficients
  - Concentration Control Coefficients
  - Elasticity Coefficients

```
>> model = IQMmodel('CellCycleRed.txt') % load model

>> modelIR = IQMmakeirreversible(model) % make reversible reactions irreversible

>> ouput = IQMmca(modelIR)              % perform MCA
       states: {9x1 cell}
    reactions: {23x1 cell}
          FCC: [23x23 double]
          CCC: [9x23 double]
           EC: [23x9 double]
```

- The analyzed model needs to contain **irreversible reactions** only!

henning.schmidt@intiquan.com

Global Sensitivity Analysis

- 4 global sensitivity algorithems implemented
  - **IQMsensglobalfast** - Extended FAST
  - **IQMsensglobalprcc** - PRCC (Partial Rank Correlation Coefficient)
  - **IQMsensglobalsobol** - Sobols method
  - **IQMsensglobalwals** - WALS (weighted average of local sensitivities)

```
(output = ) IQMsensglobalfast(model,timevector)
(output = ) IQMsensglobalfast(model,timevector,paramNames)
(output = ) IQMsensglobalfast(model,timevector,paramNames,OPTIONS)
```

Syntax

```
OPTIONS.statenames:     cell-array with state names which to consider as model outputs
OPTIONS.variablenames:  cell-array with variable names which to consider as model outputs
OPTIONS.reactionnames:  cell-array with reaction names which to consider as model outputs
OPTIONS.Nsim:           Number of simulation to carry out (approximate value)
OPTIONS.range:          Order of magnitude of parameter perturbations
OPTIONS.firstorder:     =0: use total effect, =1: use first order approx.
OPTIONS.objectivefunction: 'relative' or ' absolute'
OPTIONS.integrator:     Structure with optional settings for the integrator
```

OPTIONS

henning.schmidt@intiquan.com

IntiQuan

Global Sensitivity Analysis

Example

- Running global sensitivity analysis on the Cell-cycle model

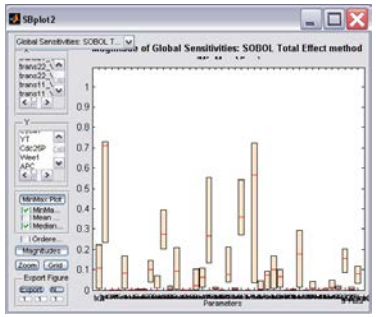
```
>> model = IQMmodel('CellCycle.txt');   % load the model
>> time = [0:2:400];                    % define time vector
>> parameters = IQMparameters(model);   % define the parameters to consider

>> OPTIONS = [];
>> OPTIONS.statenames = IQMstates(model);        % define all states as output variables
>> OPTIONS.Nsim = 1000;                          % around 1000 simulations
>> OPTIONS.range = 1;                            % order of magnitude of perturbation

>> IQMsensglobalprcc(model,time,parameters,OPTIONS)          % run PRCC method
>> IQMsensglobalsobol(model,time,parameters,OPTIONS)         % run Sobol's method
>> IQMsensglobalfast(model,time,parameters,OPTIONS)          % run FAST method
>> IQMsensglobalwals(model,time,parameters,OPTIONS)          % run WALS method

>> output = IQMsensglobalprcc(model,time,parameters,OPTIONS) % don't plot, just return data
```

Works with IQM Tools Lite only, but VERY slow
Presence of IQM Tools Pro accelerates it to be useful



henning.schmidt@intiquan.com

IntiQuan

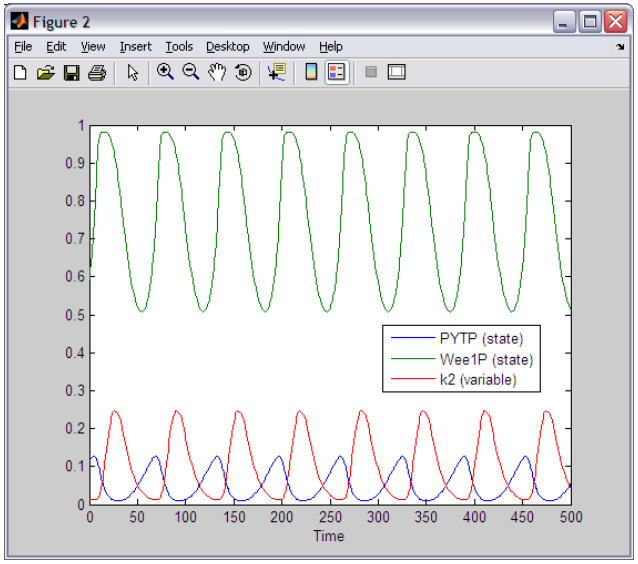
- Localization of Complex Behaviors

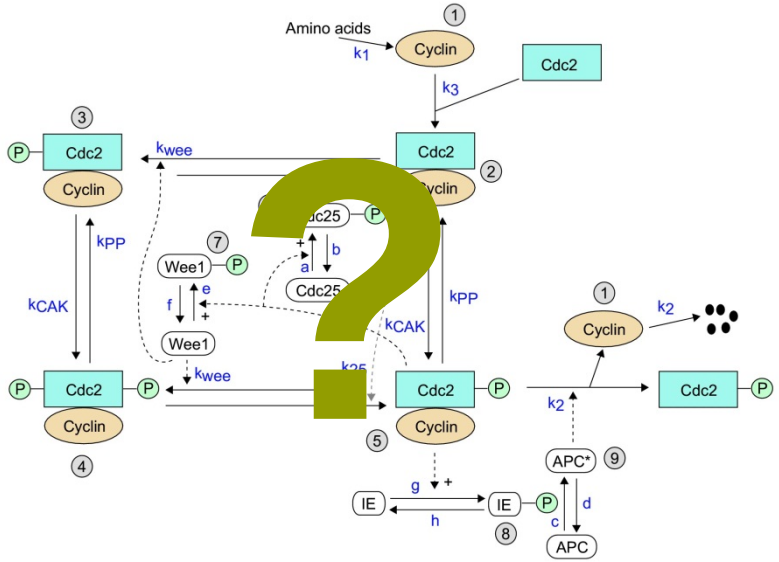
Schmidt, H., Jacobsen, E.W. (2004) Linear systems approach to analysis of complex dynamic behaviours in biochemical networks, IEE Systems Biology, 1, 149-158

henning.schmidt@intiquan.com

Localization of mechanisms leading to complex behavior

```
>> model = IQMmodel('CellCycleRed.txt')
>> IQMsimulate(model,500)
```

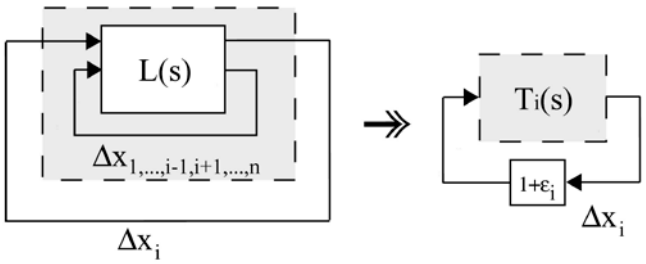




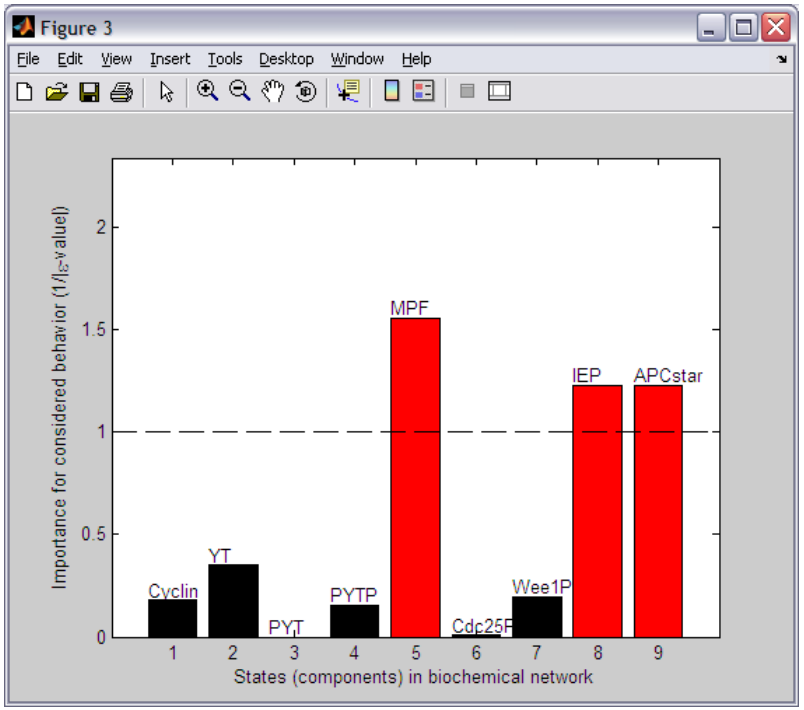
- Which mechanism is the source of the oscillations?

Localization of mechanisms leading to complex behavior

- Importance of individual components / feedback signals



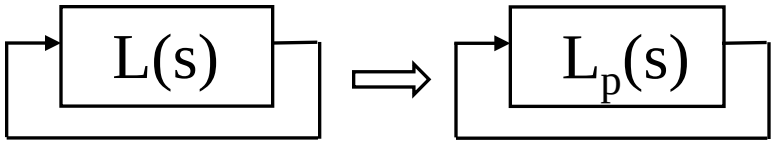

```
>> ss = IQMsteadystate(model)
>> IQMlocbehavcomp(model,ss)
```



IntiQuan

Localization of mechanisms leading to complex behavior

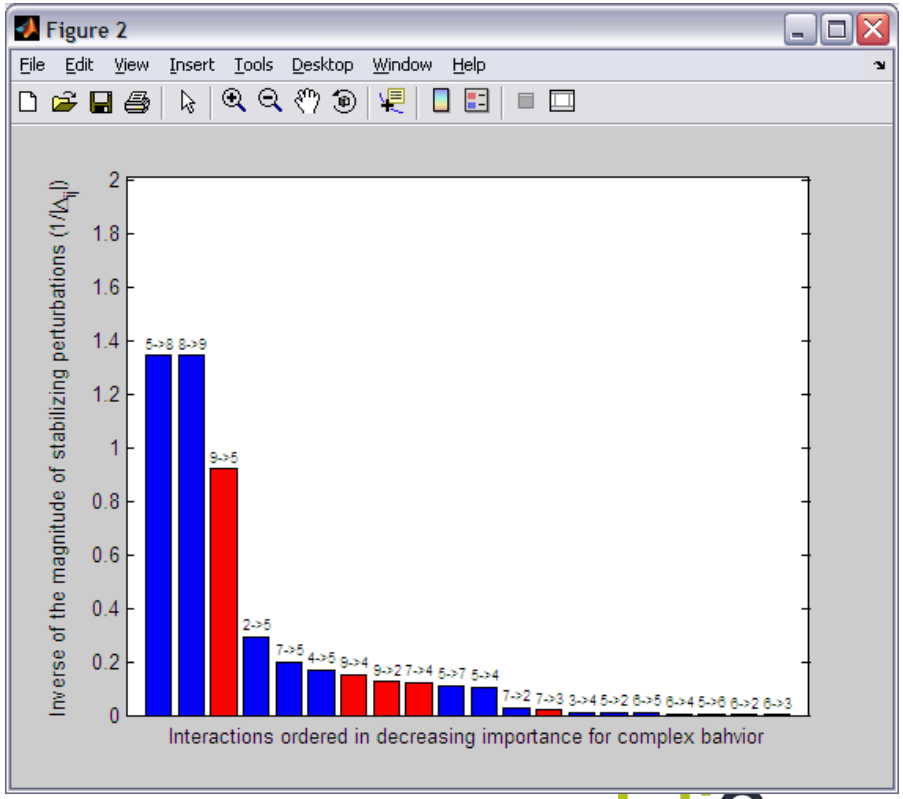
- Importance of pairwise component interactions



$$[L_p]_{ij} = [L]_{ij}\left(1 + \Delta_{ij}\right)$$

$$\Delta_{ij} = -\frac{1}{[RGA(I-L)]_{ij}}$$

```
>> ss = IQMsteadystate(model)
>> IQMlocbehavinteract(model,ss)
```

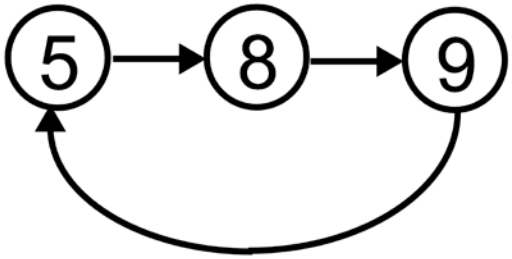


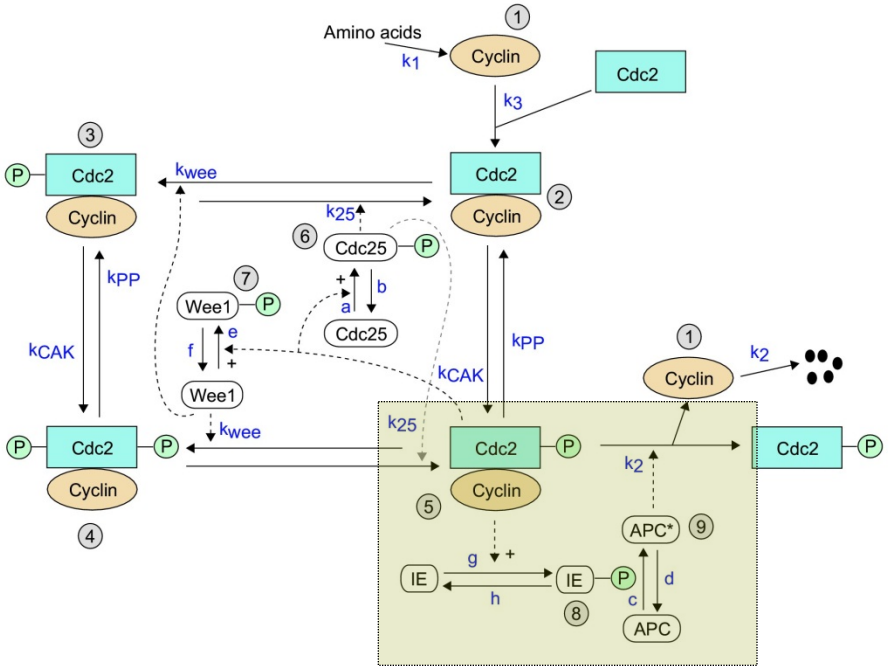
henning.schmidt@intiquan.com

IntiQuan

Identified mechanism behind oscillations

- Feedback mechanism involving
  - 5: YTP
  - 8: IEP
  - 9: APC*





henning.schmidt@intiquan.com

IntiQuan

Tutorial Outline

- General introduction to the IQM Tools Suite
- Model definition and simulation
- Model analysis
- **Model and dosing descriptions**
  - Setup models to defined dosing inputs
  - Dosing description
  - Simulation of dosing scenarios
- Definition of experiments and measurement data

henning.schmidt@intiquan.com

- Representation of models
  - Including INPUT and OUTPUT definitions in IQMmodel syntax
  - Define parameters to be estimated and parameters to function as “regression” parameters

IQMmodel Syntax for dosing inputs and link to pharmacometric parameter estimation tools (used in IQM Tools Pro)

- Same syntax as for **IQMmodel**s
  - ODEs or biochemical representation can be used
- Additional elements
  - INPUT*: Allows to define dosing inputs
  - OUTPUT*: Allows to define outputs

Example model

```
********** MODEL NAME
Linear two compartmental distribution PK model
********** MODEL STATES
d/dt(Ac) = -CL/Vc*Ac – Q/Vc*Ac + Q/Vp*Ap + INPUT1
d/dt(Ap) =           + Q/Vc*Ac - Q/Vp*Ap
********** MODEL PARAMETERS
CL = 20     % (L/hour)  Clearance
Vc = 32     % (L)              Central volume
Q  = 12     % (L/hour)  Intercompartmental clearance
Vp = 1450   % (L)       Peripheral volume
********** MODEL VARIABLES
Cc = Ac/Vc  % Calculation of plasma concentration
OUTPUT1 = Cc
```

henning.schmidt@intiquan.com

IQMmodel Syntax - INPUTs

- **INPUT Definitions in IQMmodels**
  - Please open the example file **inputExamples.m** in the folder **Example Files/MODEL_SYNTAX** and work it through
  - Example for a single input definition in an ODE based model (**model1.txt**)

```
********** MODEL STATES
d/dt(Ac)  = -VMAX*Cc/(KM+Cc) + F*INPUT1
```

  - Example for a single input definition in an model based on biochemical reaction equations (**model2.txtbc**)

```
********** MODEL REACTIONS
   => Ac : v_input
    vf = F*INPUT1

Ac =>  : v_clearance
    vf = VMAX*Cc/(KM+Cc)
```

  - Both models above are mathematically identical.
  - The **inputExamples.m** file contains also a more complex input example

IQMmodel Syntax - INPUTs

- A model can contain arbitrarily many input definitions
- INPUT* where "*": 1,2,3,4,5. Indices do not need to be sequential and do not need to start at 1, but they need to be numeric
- Input definitions are only allowed in differential equations and in reactions
- In the latter case the reaction name in the ODE is replaced by the reaction expression to have the input definition in the ODE
- A prefactor is allowed, e.g. 1.5*INPUT1 or (k1+k2)*INPUT2 or F*INPUT4 No postfactors are allowed. A "*" character has to stand between prefactor and INPUT* definition
- Input terms (INPUT* identifier and prefactor) are only allowed to be added (+) to the differential equation. But they can be the first term in the ODE, not requiring a leading "+" sign
- If a parameter with the name "INPUT*" is defined in the model, its value will not be changed during import. If the model does not contain an INPUT* parameter, it is added and set by default to "0". INPUT* is only allowed to be defined as a parameter. Not as a variable and not as a reaction
- Only parameters (and numerical values) are allowed to be used in the definition of INPUT* prefactors. But no states, variables and reactions
- INPUT* definitions are not allowed to be enclosed in any parentheses

IQMmodel Syntax - OUTPUTs

- **OUTPUT Definitions in IQMmodels**
  - Please open the example file **inputExamples.m** in the folder **Example Files/MODEL_SYNTAX** and work it through
  - Example for a single output definition in an IQMmodel (**model1.txt** and **model2.txtbc**)

```
********** MODEL VARIABLES
Cc      = Ac/Vc % (ng/ml) Plasma concentration
OUTPUT1 = Cc    % (ng/ml) Output variable
```

- The reason for the availability of these OUTPUT* definitions is that IQMtools uses these when interfacing IQM Tools with pharmacometric parameter estimation tools (NONMEM and MONOLIX)

- The right hand side of OUTPUT* definitions should NOT contain mathematical expressions, but only get assigned previously defined variables (in this example: Cc)

IQMmodel Syntax - OUTPUTs

- OUTPUT* where "*": 1,2,3,4,5, ... In contrast to INPUT* definitions, the OUTPUT indices NEED TO BE sequential, starting from 1. No number is allowed to be excluded
- Output definitions are only allowed to appear in the model variables section
- The right hand side of OUTPUT* definitions should NOT contain mathematical expressions, but only get assigned previously defined variables
- OUTPUT* variables are NOT allowed to depend on INPUT* components

IQMmodel Syntax – Parameter Info

- The last elements of the IQMmodel syntax are flags that indicate if a certain parameter should be estimated or obtained from a dataset (called: regression parameter)
- This is done by adding in the comment section of a parameter the reserved words **<estimate>** or **<regression>**
- If a parameter is neither estimated nor provided as regression parameter, its value will be kept on the value defined in the model itself

Example Files / Model_Syntax / model_estimation.txt

```
********** MODEL PARAMETERS
% PK parameters obtained from dataset
F_subcut  = 0.5      % <regression> (.)
CL        = 0.303    % <regression> (L/day)
Vc        = 2.83     % <regression> (L)
Q         = 0.724    % <regression> (L/day)
Vp        = 4.43     % <regression> (L)
VMX       = 0.5      % <regression> (mg/day)
KM        = 1.86     % <regression> (ug/ml)

% PD parameters to be estimated
BASELINE = 1         % (.)
kdeg      = 0.1      % <estimate> (1/day)
EMAX      = 1        % <estimate> (.)
EC50      = 1        % <estimate> (ug/ml)
```

- **Note: this applies only to pharmacometric type of parameter estimation**

henning.schmidt@intiquan.com

- Dosing scenarios
  - Definition of dosing scenarios
  - Simple simulation of dosing scenarios

IQMdosing Syntax

- The IQM Tools allow to define dosing schedules in a simple but efficient format
- The different dosing "inputs" then can be automatically applied to the inputs, defined in IQMmodels, using the INPUT* identifier
- In this section we will explain how to define dosing schemes and how to merge them with models to simulate the desired dosing scheme on the desired model

IQMdosing Syntax

- An example for a dosing scheme is the following

```
********** INPUT1
type:           INFUSION
time:           0       % (days) time for first application
deltaT:         14      % (days) time inbetween applications
nr_repetitions: 5       % number of applications
D:              10      % (unit) dose
Tinf:           2/24    % (days) 2 hours
```

- This dosing scheme defines that an INPUT1, defined in the model, should be implementing an infusion with a 14 day dosing interval, starting at time 0 with 5 repetitions, each dose 10 units and the infusion duration should be 2 hours

IQMdosing Syntax

- Dosing schedules can be defined in several different ways, also different doses can be given at different times. The following dose-application-types can be defined:
  - Bolus
  - Infusion (both defining infusion rate or infusion time)
  - 0th-order Absorption
  - 1st-order Absorption
- Optionally, each type can get a lag time assigned to it. Each dosing definition can define single doses or multiple doses

IQMdosing Syntax

- Dosing definitions are contained in dosing files with the extension ".dos"
  - ASCII text files which have a certain format and can contain an arbitrary number of input definitions
  - Limitation: Each input definition needs to have a different name. For example "INPUT1" is not allowed to be defined more than once

- Combining Models with Dosing Schedules
  - The underlying idea is that a model, containing inputs (INPUT1, INPUT2, etc.) can be merged with a dosing description that defines these inputs to implement the corresponding dosing definitions. The result of this merge is again a model that now can be simulated

  - Please open the example file **dosingExamples.m** in the folder **Example Files/Dosing_Syntax** and work it through

IQMdosing Syntax - Example

- Assume you have the following model
- And assume further that you want to simulate a first order absorption in the central compartment Ac
- Then this is your dosing description

```
********** MODEL STATES
d/dt(Ac)  = -VMAX*Cc/(KM+Cc) + F*INPUT1
********** MODEL PARAMETERS
VMAX = 1      % (ug/hour)
KM   = 1      % (ng/ml)
Vc   = 1      % (L)
F    = 0.6    % (.)
********** MODEL VARIABLES
Cc      = Ac/Vc % (ng/ml)
OUTPUT1 = Cc    % (ng/ml)
```

```
********** INPUT1
type:           ABSORPTION1
time:           0  % (hours)
D:              10 % (ug)
ka:             1  % (1/hour)
```

- To merge model with dosing definition we need to
  - 1) Import the model
  - 2) Import the dosing definition
  - 3) Merge model with dosing definition to obtain a model for simulation

henning.schmidt@intiquan.com

IQMdosing Syntax - Example

```
>> model = IQMmodel('model1.txt');                % Import of model
>> dosing = IQMdosing('dosing2.dos');             % Import of dosing description
>> modeldosing = IQMmergemoddos(model,dosing);    % Merge model and dosing description:
```

- Note the added elements and the naming of ka_input1, ...

```
********** MODEL STATES
d/dt(Ac) = -VMAX*Cc/(KM+Cc)+vAbsorption_input1  %(ug/hour)
d/dt(Comp_input1) = input1-vAbsorption_input1
********** MODEL PARAMETERS
VMAX = 1  %(ug/hour) Maximum rate of elimination
KM = 1    %(ng/ml)   At this concentration elimiation rate is half VMAX
Vc = 1    %(L)       Central volume
F = 0.59999999999999998  %(.)       Relative bioavailability
Dose_input1 = 10
Time_input1 = 0
DeltaT_input1 = 0.0001
ka_input1 = 1
********** MODEL VARIABLES
Cc = Ac/Vc  %(ng/ml) Plasma concentration
OUTPUT1 = Cc  %(ng/ml) Output variable

vAbsorption_input1 = ka_input1*Comp_input1
input1 = +F*Dose_input1/DeltaT_input1 *
         piecewiseIQM(1,andIQM(ge(time,Time_input1),lt(time,Time_input1+DeltaT_input1)),0)
```

Tutorial Outline

- General introduction to the IQM Tools Suite
- Model definition and simulation
- Model analysis
- Model and dosing descriptions
- **Definition of experiments and measurement data**
  - Excel and CSV measurement representation
  - Experiment descriptions and merging with models
  - Import and export measurements and experiments

- Representation of measurement data

IQMmeasurement

henning.schmidt@intiquan.com

IntiQuan

Measurements

- IQM Tools Lite can handle 2 representation formats
  - Excel
  - Comma separated values (CSV)
- One Excel file can contain measurements of several experiments
- One CSV file can contain measurements of a single experiment
- Contents
  - Name
  - Notes
  - Components (same names as model components)
  - Componentnotes (additional information, e.g. units)
  - Values

henning.schmidt@intiquan.com

Measurement Examples and Import

- Measurements in Excel

```
Change into the "Example Files" folder
Open the "MeasurementExample.xls" in Excel

>> data = IQMmeasurement('MeasurementExample.xls')

data = [1x1 IQMmeasurement]    [1x1 IQMmeasurement]

>> data = data{1}  % use only first data object in the cell-array
```

WINDOWS ONLY

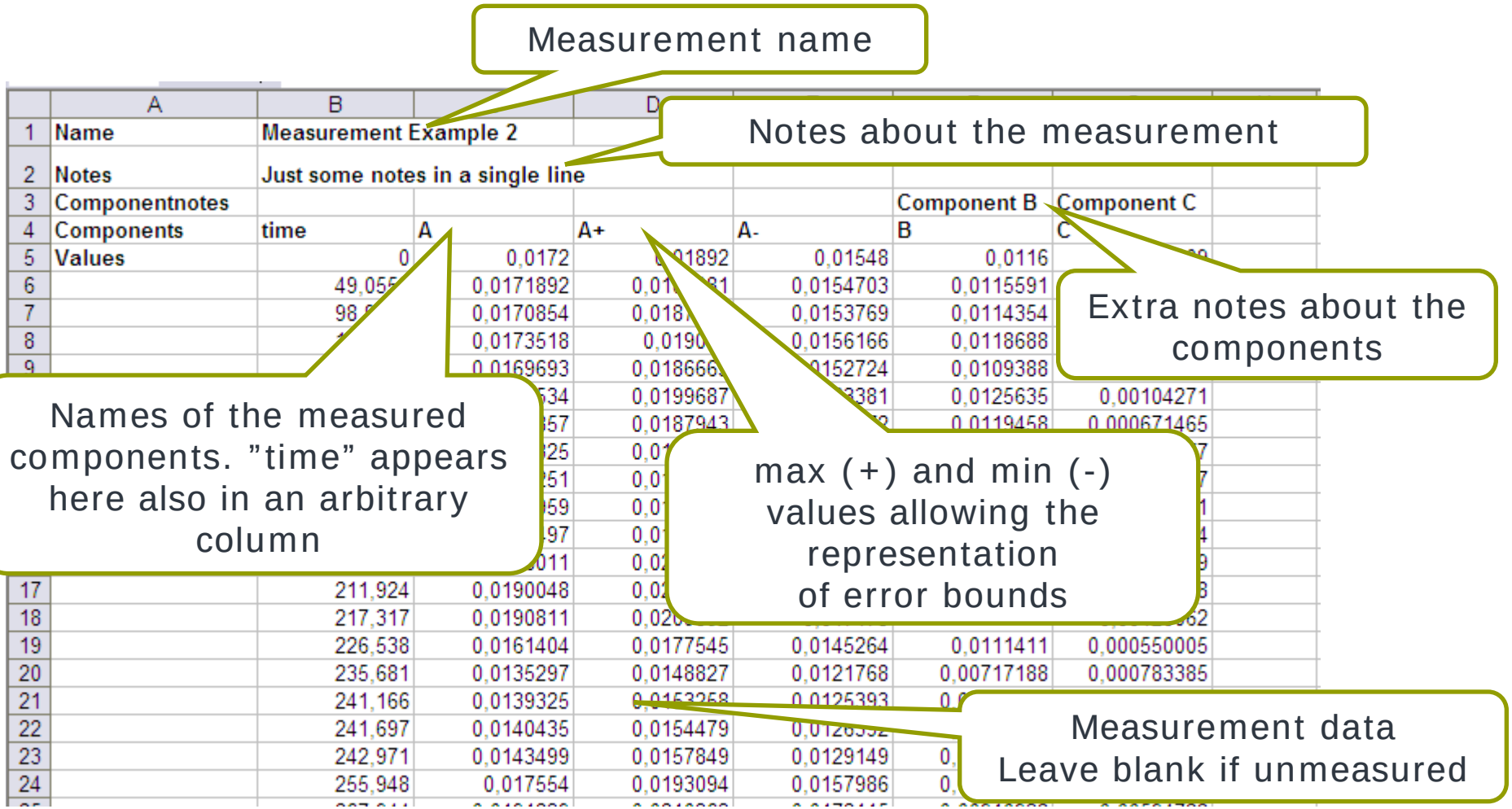
- Measurements as CSV (importing CSV is MUCH faster then XLS)

```
>> edit MeasurementExample.csv

>> data = IQMmeasurement('MeasurementExample.csv')

    IQMmeasurement
    ==============
    Name: MeasurementExampleCSV
    Measured components:      3
    Number time points:       42
    Error bound information present at least for one measurement.
    Measurements not present for all time points
```

IQMmeasurements – Excel format



	A	B		D			
1	Name	Measurement Example 2					
2	Notes	Just some notes in a single line					
3	Componentnotes					Component B	Component C
4	Components	time	A	A+	A-	B	C
5	Values	0	0,0172	01892	0,01548	0,0116	
6		49,055	0,0171892	0,01 81	0,0154703	0,0115591	
7		98,8	0,0170854	0,018	0,0153769	0,0114354	
8		1	0,0173518	0,0190	0,0156166	0,0118688	
9			0,0169693	0,018666	0152724	0,0109388	
			534	0,0199687	3381	0,0125635	0,00104271
			857	0,0187943	2	0,0119458	0,000671465
			925	0,01			7
			951	0,01			7
			959	0,01			1
			97	0,01			4
			011	0,02			9
17		211,924	0,0190048	0,02			8
18		217,317	0,0190811	0,020			062
19		226,538	0,0161404	0,0177545	0,0145264	0,0111411	0,000550005
20		235,681	0,0135297	0,0148827	0,0121768	0,00717188	0,000783385
21		241,166	0,0139325	0,0153268	0,0125393	0,0	
22		241,697	0,0140435	0,0154479	0,012		
23		242,971	0,0143499	0,0157849	0,0129149	0,	
24		255,948	0,017554	0,0193094	0,0157986	0,	

Worksheets in an Excel book can contain any data. BUT if A1 is set to ”Name” then the above format is expected.

henning.schmidt@intiquan.com

IntiQuan

IQMmeasurements – CSV format

```
[Name]
Measurement Example CSV

[Notes]
Notes about the measurement / experiment

[Components]
time,A,A+,A-,B,C

[Componentnotes]
A: Component A
B: Component B

[Values]
0, 0.0172, 0.01892, 0.01548, 0.0116, 0.0009
49.0552, 0.0171892, 0.0189081,  , 0.0115591, 0.000865671
98.9524, 0.0170854, 0.0187939, NaN , 0.0114354, 0.000858473
128.814, 0.0173518, 0.019087, 0.0156166, 0.0118688, 0.000825422
151.362, 0.0169693, 0.0186663, 0.0152724, 0.0109388, 0.00104578
160.548, 0.0181534, 0.0199687, 0.0163381, 0.0125635, 0.00104271
```

Good practice
Give a useful name and document the measurement data with telling notes

Measurement data
Leave empty or set to NaN if unmeasured

Important:
The measurements should be given in units that are to be used for the models components!

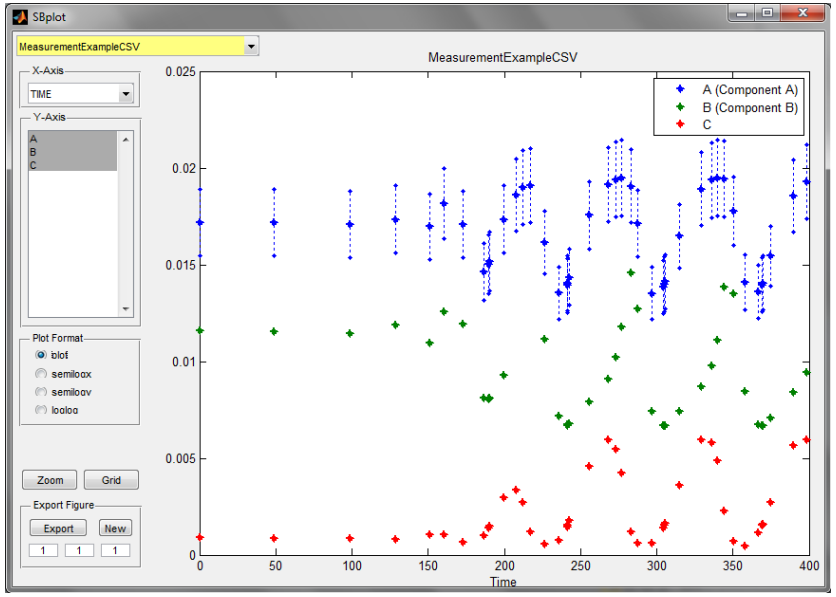
henning.schmidt@intiquan.com

IntiQuan

Handle Measurement Data

- Visualizing the data

```
>> IQMvisualizemeasurement(data)
```



- Extract information

```
>> [time, componentNames, values, minvalues, maxvalues] = IQMmeasurementdata(data)
```

henning.schmidt@intiquan.com

- Representation of in silico experiments

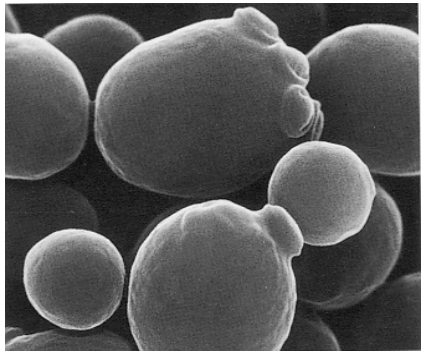
IQMexperiment

henning.schmidt@intiquan.com

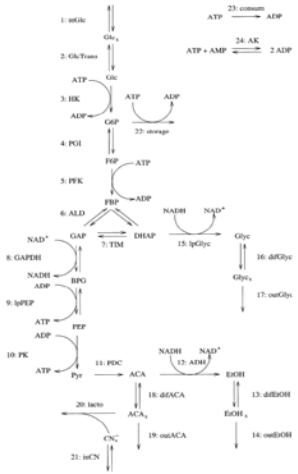
IntiQuan

Experiment Descriptions

- The System



The Model



- The Experiments
  - Knockout
  - Overexpression
  - Change of concentration 1
  - Change of concentration 2
  - etc.

One slightly different model for each experiment?

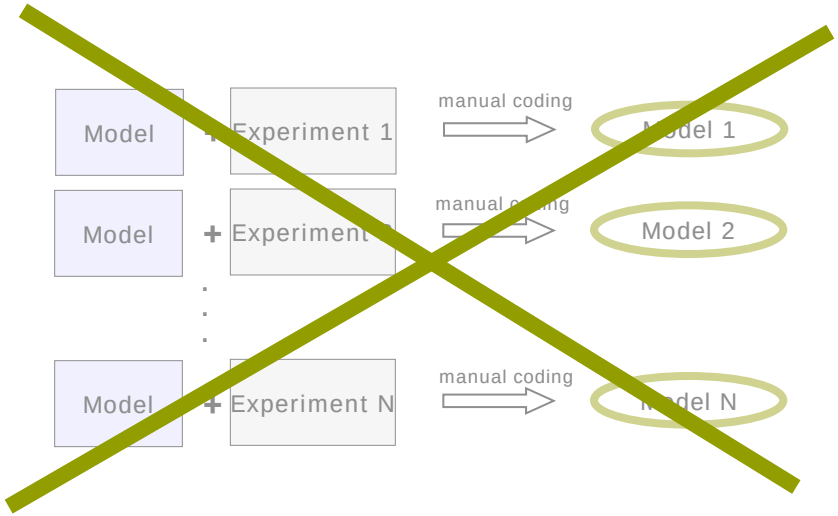
henning.schmidt@intiquan.com

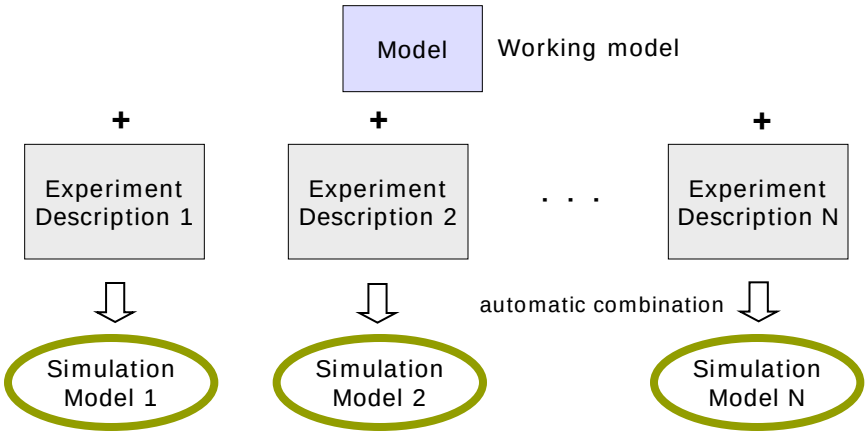
IntiQuan

Experiment Descriptions – Why?

- Typically many different biological experiments

- Coding each experimental setting in the model leads to
  - very complicated models or
  - a large number of models to keep track of





henning.schmidt@intiquan.com

IntiQuan

Experiment Descriptions

- Experiment descriptions allow to define experimental settings insilico
- Experiment descriptions in IQM Tools allow to
  - change initial conditions
  - change parameter values
  - change parameter values over time
  - change state variables at given time instants
- A simple example

```
>> edit model1.txt
>> model = IQMmodel('model1.txt')                  % load the model

>> edit experiment1.exp
>> experiment = IQMexperiment('experiment1.exp')   % load the experiment

>> modelexp = IQMmergemodexp(model,experiment)     % combine model with experiment

>> IQMedit(modelexp)                               % have a look at the new model
```

Click ”Simulate”

henning.schmidt@intiquan.com

Experiment Descriptions – How does it work?

```
********** MODEL NAME
Simple model
********** MODEL STATES
d/dt(A) = -R
d/dt(B) = R
A(0) = 1
B(0) = 0
********** MODEL PARAMETERS
k1 = 0.5
********** MODEL REACTIONS
R = k1*A
```

```
********** EXPERIMENT NAME
Simple Experiment for simple model
********** EXPERIMENT INITIAL PARAMETER AND STATE SETTINGS
k1 = 2
********** EXPERIMENT PARAMETER CHANGES

********** EXPERIMENT STATE CHANGES
time=10, A=1
```

```
********** MODEL NAME
Simple model
********** MODEL STATES
d/dt(A) = -R
d/dt(B) = R
A(0) = 1
B(0) = 0
********** MODEL PARAMETERS
k1 = 2
********** MODEL REACTIONS
R = k1*A
********** MODEL EVENTS
StateChange_1 = ge(time,10),A,1
```

The result is a new model where the experimental settings have been added

henning.schmidt@intiquan.com

Experiment Descriptions

- The general syntax is described in the **`experiment2.exp`** file

```
>> edit experiment2.exp
```

- An experiment object is realized as an object of class IQMexperiment

Internal Experiment Data Structure

```
>> experimentstructure = struct(experiment)

experimentstructure =

             name: 'Simple Experiment'
            notes: 'Simple Experiment for Simple Model‘
  paramicsettings: [1x1 struct]
 parameterchanges: [0x0 struct]
      stateevents: [1x1 struct]
```

- Import/Export of measurement data
  - CSV
  - Excel

Measurements Export

- Export a single measurement to an Excel file

```
>> IQMexportXLSmeasurement(data, 'filename')      % variable „data“ already defined
                                                  % from previous commands
```

- Exporting several measurements to an Excel file

```
>> IQMexportXLSmeasurements({data, data}, 'filename2')  % just export twice the same data
```

- Export to CSV file

```
>> IQMexportCSVmeasurement(data, 'filename')
```

- Import of data always by using the **IQMmeasurement** command

```
>> data = IQMmeasurement('filename.csv')          % imports the CSV file
>> datacellarray = IQMmeasurement('filename2.xls')   % imports the Excel data file
```

henning.schmidt@intiquan.com

- Import/Export of experiment descriptions

Experiment Descriptions

- Export of experiment descriptions

```
>> IQMcreateEXPfile(experiment, 'experimentfile') % “experiment” defined from before
```

- Import of experiment description files

```
>> exp = IQMexperiment('experimentfile.exp')
```

Tutorial Goal: „You should now be able to“

- Define models using ODEs and/or biochemical reactions
- Simulate and analyze models
- Simulate models for defined dosing inputs
- Define and simulate experiments on models
- Define measurement data, import, export, plotting

THE END

Thank you for your participation and interest!

The tutorial continues in Part 2 (IQM Tools Pro / MEX models, SB projects, parameter estimation)

henning.schmidt@intiquan.com