Modeling Molecular Regulatory Networks with JigCell and PET

Clifford A. Shaffer, Jason W. Zwolak, Ranjit Randhawa, and John J. Tyson

Abstract We demonstrate how to model macromolecular regulatory networks with JigCell and the Parameter Estimation Toolkit (PET). These software tools are designed specifically to support the process typically used by systems biologists to model complex regulatory circuits. A detailed example illustrates how a model of the cell cycle in frog eggs is created and then refined through comparison of simulation output with experimental data. We show how parameter estimation tools automatically generate rate constants that fit a model to experimental data.

Key words: Systems biology, parameter estimation, model validation.

Clifford A. Shaffer
Department of Computer Science, Virginia Tech, Blacksburg, VA 24061, e-mail: `shaffer@vt.edu`

Jason W. Zwolak
Department of Biological Sciences, Virginia Tech, Blacksburg, VA 24061, e-mail: `jzwolak@vt.edu`

Ranjit Randhawa
Department of Computer Science, Virginia Tech, Blacksburg, VA 24061, e-mail: `rrandhawa@vt.edu`

John J. Tyson,
Department of Biological Sciences, Virginia Tech, Blacksburg, VA 24061, e-mail: `tyson@vt.edu`

1 Introduction

Mathematical models of gene-protein regulatory networks play key roles in archiving and advancing our understanding of the molecular basis of cell physiology. Models provide rigorous connections between the physiological properties of a cell and the molecular wiring diagrams of its control systems. A simple example is the set of reactions controlling the activity of MPF (mitosis promoting factor) in *Xenopus* oocytes ***(1)***, which we refer to herein as the frog egg model. In the diagram of this network (**Fig. 1**), vertices represent substrates and products (collectively referred to as species), solid directed edges represent biochemical reactions, and dashed directed edges represent regulatory signals.

Collectively, these biochemical reactions cause the concentrations of the chemical species (S_i) to change in time according to a set of differential equations (one for each species)

$$\frac{dS_i}{dt} = \sum_{j=1}^{R} b_{ij} v_j, i = 1, \dots, N,$$

where R is the number of reactions, N is the number of species, v_j is the velocity of the j^{th} reaction in the network, and b_{ij} is the stoichiometric coefficient of species i in reaction j ($b_{ij} < 0$ for substrates, $b_{ij} > 0$ for products, $b_{ij} = 0$ if species i takes no part in reaction j). **Fig. 1** shows differential equations derived from the reactions in the network diagram. The set of rate equations and associated parameter values is a mathematical representation of the temporal behavior of the regulatory network.

Since the purpose of these models is to codify a systems-level understanding of the control of some aspect of cell physiology, it is necessary to validate a proposed model against observed behavior of the reference system. In most cases it is essential to model the behavior of not only the wild-type form of the organism, but also of many mutant forms (where each mutant form typically represents one or two variations in the genetic specification of the control system). For example, if we are modeling the cell cycle of an organism, then we would wish to know features such as the cell size at division, the time required for various phases of the cell cycle (G1, S, G2, M), and the viability or point of failure for each mutation. Measurements of the amounts for various control species within the cell over time would also be valuable information. In the case of a thoroughly studied organism such as *Saccharomyces cerevisiae* (budding yeast), a model can be compared against many dozens of mutants defective in the regulatory network.

A realistic model of the budding yeast cell cycle consists of over 30 differential equations and 100 rate constants and is tested against the phenotypes of over 150 mutants ***(2)***. A model of this complexity represents the upper limit of what a dedicated modeler can produce “by hand” with nothing but a good numerical integrator like LSODE ***(3)***. Beyond this size, we begin to lose our ability even to meaningfully display the wiring diagram that represents the model, let alone comprehend the information it contains, or determine suitable rate constants in the corresponding high-dimensional space. To adequately describe fundamental physiological processes (such as the control of cell division) in mammalian cells will require models of 100-1000 equations. To handle this next generation of dynamical models will require sophisticated software to automate the modeling process: network specification, equation generation, simulation and data management, and parameter estimation.

There are a number of distinct approaches to simulation. Deterministic models usually represent the system of chemical reactions with ordinary differential equations ***(4, 5, 6)***. In some cases, partial differential equations are used to account for spatial effects ***(7)***. Stochastic modeling is in its infancy, and most often is done by some variation of Gillespie’s algorithm ***(8, 9, 10)***. For the remainder of our discussion, we will consider only deterministic simulation by ordinary differential equations (ODEs).

Creating a model that mimics the observed behavior of a living organism is a difficult task. This process involves a combination of biological insight, persistence, and support by good modeling tools. In the following sections, we will describe the model development process that we employ and the software tools that we have developed to construct and test models. We then provide a detailed example of how the tools can be used to create a simple model of the frog egg cell cycle and to estimate the associated rate constants.

2 The Modeling Process

Successful modeling of macromolecular regulatory networks can be aided by software tools based on a well-defined modeling process. Such tools should support the line of thought followed by modelers as they approach a problem. Mid-sized models of macromolecular regulatory networks track reactions among tens of species and are tested against hundreds of experimental observations. Thus, modelers need tools that help to organize the relevant information and automate as many steps of the process as possible. **Fig. 2** shows our conception of the modeling process. The modeler starts with an idea about an organism and a regulatory system to model. Next, the modeler gathers information (from the literature and from their own experiments) related to the regulatory system of the organism. During the literature search the modeler builds a hypothesis from information already published, continuously checking the hypothesis against the existing literature. Once the modeler has a testable hypothesis about the regulatory system, the hypothesis can be codified into four types of technical information:

- Experimental Data: The information that will be used to validate the model. This information might come as time series data of the concentrations of certain regulatory chemical species, as other observables such as the average size of cells at division, or as qualitative properties such as the viability or inviability of a mutant.
- Simulation Runs: Specifications for the simulations that will be made to model the experimental data. For example, each simulation might relate to a specific mutation of the organism. The specification will define the distinct conditions necessary to simulate that mutation, such as differences in rate constants from the wild-type values.
- Reaction network: The chemical equations that describe the regulatory processes.
- Rate constants: The parameters that govern the reaction rates.

Typically, the experimental data and simulation run descriptions are part of the problem definition and are not subject to frequent modifications. Nor are they considered to be "right" or "wrong" in the same way as the reaction network and rate constant values typically will be. The network and rate constants together define the mathematical model that will be simulated, compared to experimental observations, and judged "acceptable" or "unacceptable."

One simulation run of an ODE model takes only a fraction of a second on a typical desktop computer in 2007. As described above, a complete model actually involves a large collection of simulations, to be compared against a collection of experimental results. This entire set of runs might take a second or so for a smaller model such as our frog egg example on a desktop computer for one choice of rate constants, and about a minute or two for a larger model needed to describe the budding yeast cell cycle.

Once an initial specification of these four types of information has been made, the next phase of the process begins. This is a simulation-compare-update loop, whereby simulation results are compared to the experimental data. In some way, either a human or a computer will make a judgement as to the quality of the relationship between the two. At that point, since the model is typically judged unsatisfactory, the modeler will make adjustments and repeat the cycle. We prefer to view this as a double loop, in that changes to rate constant values are made much more frequently than changes to the reaction network. That is, the modeler will typically "twiddle" the rate constants so long as progress is being made in matching simulation output to experimental data. When changes to the rate constants appear no longer to improve the match, then the modeler will attempt to improve the model by changing the reaction network, which in turn will trigger another round of changes to the rate constants. The process is continued until the model is judged satisfactory or totally hopeless.

Modelers often try to assign values to rate constants by a time-consuming process of "parameter twiddling" and visual comparison of simulation results to experimental data. A better approach is automated parameter estimation (once the modeler is confident that the basic structure of the reaction network is sound enough). To fit a model to experimental data by automated optimization algorithms requires thousands to millions of repetitions of the full calculations.

The process of comparing real-world observation (experimental data) with the mathematical model (time-series output from a simulation) is called model validation. Model validation is closely related to automated (or manual) parameter estimation, because both require that some measure of the quality of the model can be made. In the case of automated parameter estimation, we need a way to take the experimental data and the output from a simulation run, and create a single number as a measure of the quality of the fit.

This can be extremely difficult. First, the simulation data (usually in the form of time series plots) might not be similar to the form of the experimental data (often qualitative information such as whether a cell is viable or not). In general, some complex computation must be done to relate the two. The function that does this computation is called a *transform* and is discussed further in **Section 3.3.1**. Second, while it might be a simple judgement to measure the goodness of fit between one simulation and one experiment, it is often difficult to judge the goodness of fit of an entire ensemble of runs, where improvements in matching some experiments might come at the cost of worse fits for others. The function that balances these fits is called the objective function and is discussed in **Section 3.3.2**.

3 Software Tools

Before the current generation of modeling tools for systems biology was developed, many stages in the modeling cycle described in **Section 2** were done by hand. This presents two problems. First, it takes a great deal of time and effort to convert the original intuitive concept of a model into a suitable set of reaction equations and simulations. Second, there are many opportunities for errors, especially at the (essentially mechanical) step of converting a reaction mechanism into differential equations.

A wiring diagram, like **Fig. 1**, nicely represents the topology of a reaction network (reactants, products, enzymes). But it is not a good representation for specifying the kinetics of the network (the reaction rate laws, v_j). A large reaction network can become so complex that even its topological features are obscured by a large number of intersecting lines. Obscurity is increased by the fact that there is no standard format for drawing such graphs. Without precise notational conventions, it is impossible to convert a wiring diagram unambiguously into a model, either by hand or by machine.

Another approach for deriving a model is to explicitly write out the chemical reactions. This loses some of the intuitive appeal of the diagrammatic approach, but allows for a more compact definition of a reaction network. Normally, the modeler has already made a hand or CAD-drawn version of the network in graphical form, showing the interactions in a qualitative sense but without the quantitative information of the rate equations or the rate constant values.

Models often include concepts not captured by the differential equations alone. Conservation relations are defined by linear combinations of species concentrations that remain constant throughout a simulation: $T_i \equiv \sum_{i=1}^{N} a_i S_i(t)$, where a_i is a constant and T_i is constant. Such constraints arise from linear dependencies in the stoichiometry matrix: $\sum_{i=1}^{N} a_i b_{ij} = 0$. Events are special actions that trigger in the model under given conditions. For example, cellular division could be represented by a halving of cell mass, and might occur when a given function involving some number of chemical species reaches a threshold during a simulation.

The key to successfully creating and managing such complex models is to use software tools that organize the information in a coherent way and catch inconsistencies and errors early in the process. In this section we will describe the JigCell Model Builder ***(11, 12)***, which is used to define the reaction equations and rate constants of the model. We then present the JigCell Run Manager ***(13)***, which is used to define a series of simulation runs that will generate output to validate the model. Finally, we describe the Parameter Estimation Tool (PET) ***(14)***, which supports exploration of the parameter space and automated parameter estimation with the goal of selecting rate constant values that best fit the simulation output to the experimental data.

Underlying any such software tool is a representation scheme for describing a model, that is, a language for expressing the model in a complete and formal sense. The Systems Biology Markup Language (SBML) ***(15, 16)*** has now become the standard reference language for reaction network modeling. SBML describes all necessary features pertaining to the reaction network, conservation relations, events, and rate constants. SBML does not describe all data necessary for modeling, including information describing the simulation runs and experimental data from **Fig. 2**, which must be stored in separate files. SBML also is not a suitable language for human comprehension. Thus, software tools are needed to provide an interface between the user and SBML.

3.1 The JigCell Model Builder

The JigCell Model Builder (referred to herein as the "Model Builder") is used to define the components that make up an SBML model. The Model Builder uses a spreadsheet interface, allowing a large amount of data to be displayed in an organized manner.

The Model Builder provides functionalities for both first-time users and expert modelers. The Model Builder supports the definition of events and user-defined units. An event, such as cell division, can be defined by specifying a condition that must be met to trigger the event, and the changes that result due to the event. A major goal of the Model Builder is to minimize the time and errors associated with translating a regulatory network to a set of equations. As the user enters reaction equations, rate laws, and functions into their cells in the main spreadsheet, several other spreadsheets are updated to track the various entities that make up a model. After the user has finished defining a model using the Model Builder, this model can be used with other SBML-compliant software to simulate the response of the model to given conditions.

The Model Builder's interface is broken into 10 spreadsheets, all accessible by clicking on the appropriate tab. **Fig. 3** shows the "Reactions" spreadsheet. There is a spreadsheet for each of the eight SBML components in a model (reactions, functions, rules, compartments, species, parameters, units and events). There is one spreadsheet for conservation relations and one spreadsheet for the equations (including both ODEs and rule equations).

The “Reactions” spreadsheet is the primary tool used to create the reaction network of a model. The other spreadsheets are either partially or completely filled by the Model Builder from the “Reactions” spreadsheet. A reaction represents any chemical transformation, transport, or binding process that can change the amount of one or more species. Each row defines a single chemical reaction. **Fig. 3** shows the “Reactions” spreadsheet loaded with the frog egg model. The three main columns in this spreadsheet are: “Reaction,” “Type,” and “Equation”.

1. The “Reaction” column defines the species (reactants and products) and their stoichiometries. A list of substrates separated by ‘+’ signs is entered first. An arrow ($\rightarrow$) is then entered, and is followed by a list of products, also separated by ‘+’ signs. Substrate and product names can contain any combination of letters, numbers, underscores, and apostrophes. There is no limit to the number of species that can be entered as substrates or products. The stoichiometry of a reaction is defined by placing a number and an ‘*’ character in front of the species (e.g., $3 * M_a$).
2. By picking a rate law from a drop down list in the “Type” column, the user can specify the kinetics of the reaction being defined. The Model Builder provides three built-in rate laws (Mass Action, Michaelis Menten, Local) and also allows users to define their own rate laws in the “Functions” spreadsheet. For all rate laws other than Local, the Model Builder will enter the associated rate law in the “Equation” field. The Local type allows the user to define the reaction rate of a single reaction without creating a new rate law. If the user selects Local, the equation field will remain empty until the user defines the equation for the

reaction rate. Local rate laws may contain algebraic expressions with parameters and species.

3. The “Equation” column specifies the equation for the rate of the reaction. If the reaction type is not Local, the “Equation” column displays the unsubstituted equation of the selected rate law until the user edits the rate law equation by clicking on the cells in this column. Clicking on one of these cells displays the “Parameters/Modifiers” Editor (**Fig. 4**), where the user assigns ‘interpretations’ to the rate constants and modifiers. The ‘interpretations’ can be numeric constants, expressions, species or species related expressions. The Model Builder partially fills the “Parameters/Modifiers” Editor when built-in rate laws are used (e.g., S1 becomes Ci in **Fig. 4** automatically because the user defined the reaction Ci $\rightarrow$ Ca). The Model Builder will substitute the user’s interpretations (entered via the “Parameters/Modifiers” Editor) into the Equation field of the “Reactions” spreadsheet so that the user can see the final rate law used to govern the reaction. Expressions are evaluated to numerical values when the model is simulated.

The “Functions” spreadsheet (**Fig. 5**) is used to create and edit function definitions. A function definition is a named mathematical function that may be used throughout the rest of a model. For example, user-defined rate laws are created as function definitions. Checking the box in the “Rate Law” column causes the newly created rate law to be included in the drop-down list of rate laws in the “Type” column of the “Reactions” spreadsheet. Functions are defined with place holders for arguments of the form *A#*, where # is some number. The function My_rate_law in (**Fig. 5**) contains five arguments A1 – A5. These arguments can be assigned in the “Parameters/Modifiers” Editor (**Fig. 4**) when the function is selected as the rate law for a reaction. Otherwise, to use this function it may be called like this: $My\_rate\_law(vwp, Wi, vwpp, Wa, Ma)$. Any of the function arguments can be a parameter, species, or algebraic expression.

The “Rules” spreadsheet (**Fig. 6**) serves two purposes. First, it displays algebraic rules, which are the conservation relations in the model. The program deduces these relations from the stoichiometric matrix of the model and displays each conservation relation in the form $(a_1S_1 + a_2S_2 + ...) - T_i = 0$, where T_i is the conserved quantity and a_1, a_2, $\ldots$ are constants calculated from the stoichiometry matrix. The user cannot edit an algebraic rule on this spreadsheet but may specify how the Model Builder uses the rule on the “Conservation Relation” spreadsheet. The second purpose of the “Rules” spreadsheet is to create and edit assignment rules. Assignment rules are used to express equations that set the value of variables. The “Variable” field in the assignment rule can be a species, parameter or compartment. In the case of species the “Equation” field sets the quantity to be determined (either concentration or substance amount), in the case of compartments the “Equation” field sets the compartment’s size, and in the case of parameters the “Equation” field sets the parameter’s value. The value calculated by the assignment rule’s “Equation” field overrides the value assigned in the “Compartments,” “Species,” or “Parameters” spreadsheet.

The next three tabs are used to define compartments, species, and parameters. A compartment represents a bounded space in which species can be located. Spatial relationships between different compartments can be specified. Modelers are not required to enter compartment information when defining a model, as a single compartment called ‘cell’ is created by default. The “Species” spreadsheet (**Fig. 7**) provides a list of all species that are part of a chemical reaction or defined in a Rule. The list of species is generated automatically by the Model Builder, though a user can add, delete, and modify species. There are several editable attributes associated with each species. The “Parameter” spreadsheet (**Fig. 8**) is used by the Model Builder to manage all parameters and their values associated with a model. A parameter is used to declare a value for use in mathematical formulae. The Model

Builder recognizes as a parameter any name on the “Reactions” spreadsheet that is not defined as a species.

The “Events” spreadsheet (**Fig. 9**) allows the user to define actions associated with a model. For example, when modeling the cell cycle, some trigger for cell division must be defined and the consequences of that division must be specified. The “Name” column provides an (optional) identifier for an event. The “Trigger” column defines the conditions under which the event takes place. The format of this entry allows the user to specify an equality relationship. Whenever the relationship entered in the “Trigger” column is satisfied, the actions specified in the “Assignments” column will occur. The “Event Assignment Editor” lets the user define the changes that will occur when an event is executed.

The “Units” Spreadsheet lists all unit types used in the model, along with their definitions. A unit definition provides a name for a unit that can then be used when expressing quantities in a model. The Model Builder has a number of basic units and 5 built-in unit definitions (area, length, time, substance and volume). Complex unit definitions such as $meter/second^2$ can be created.

The “Conservation Relations” spreadsheet (**Fig. 10**) is used to view a list of all conservation relationships that exist between species in the model. The list of conservation equations is generated automatically, using Reder’s method ***(17)***.

The “Equations” spreadsheet (**Fig. 11**) allows the modeler to see a list of the different types of equations that define the model. The user does not edit equations here, as they are created automatically from data entered on other spreadsheets. The “Equation” column displays differential equations, assignment rule equations, conservation relation equations or the condition set on the species when no equation exists.

3.2 The JigCell Run Manager

The JigCell Run Manager (referred to herein as the “Run Manager”) lets users define specifications for an ensemble of simulation runs. Hierarchies of simulations can be built up, whereby a given simulation inherits parameter changes from a “basal” run definition. This hierarchical organization of simulations is useful because models are often validated against a collection of experimental protocols, each one of which requires only slightly different simulation conditions. For example, the budding yeast cell cycle model must capture the differences among many dozens of mutations of the wild type organism. If the ‘basal’ run represents the wild-type organism, then the hierarchy can define unambiguously and compactly the deviations from wild-type that are necessary to specify each mutant type.

Users input the description of ensembles using 5 spreadsheets: Runs, Basal Parameters, Basal Initial Conditions, Simulator Settings and Plotter Settings. The “Runs” spreadsheet (**Fig. 12**) specifies how to simulate a certain experiment. The name column can (optionally) be used to identify the experiment being simulated. The parents column lists all runs from which the row inherits changes. The changes column lists additional changes to parameters, initial conditions, simulator settings and plotter settings that are needed for this run. The changes are specified using the “Changes” editor (**Fig. 13**), which opens when clicking on the changes cell for a particular run. The changes for a particular run override the changes inherited from any parents, and these changes propagate to its children. Color is used to reflect where the changes are made: Blue is used to indicate changes made in the current run (locally) and green to indicate changes inherited from a parent run (or some previous ancestor). This information is also indicated in the “Parents” column of **Fig. 13**, which indicates either the name of the ancestor that caused that parameter’s setting to change, or states “local” if the change was explicitly made by the user for this run. **Fig. 12** shows a “Runs” spreadsheet for simulating some experiments done on frog egg extracts to characterize the activation of MPF.

Each row corresponds to a separate experiment. The run named “Interphase” (on row 1) describes changes to the initial model in order to simulate an extract starting in interphase. This run is then set as a parent to the run named “Kumagai and Dunphy 1995 Fig 3C Interphase” on row 6. The run on row 6 inherits all its parent’s changes and represents an experiment to measure the phosphorylation of MPF by Wee1 during interphase. The “Changes” column displays changes made by the current run but not changes inherited from the parents.

The Run Manager provides a “Plot” button on the “Runs” spreadsheet that generates an immediate simulation for a specified row and then plots the results.

The “Simulator Settings” spreadsheet (**Fig. 14**) specifies the simulator to be used and appropriate values for the simulator’s configuration parameters, such as total time of integration, tolerances, output interval, etc. In this case the simulator chosen is XPP ***(18)***. Other simulators are also provided, such as StochKit ***(19)*** (for stochastic simulation) and Oscill8 ***(20)***.

The “Plotter Settings” spreadsheet (**Fig. 15**) enables the user to specify the variables to be plotted from a simulation run’s output. The “Plotter Settings” spreadsheet also contains options to customize the plot by selecting colors, mark styles, whether to connect points, etc.

3.3 PET: Parameter Estimation Toolkit

The Parameter Estimation Toolkit (PET) is designed to help users explore parameter space and fit simulation output (e.g., time course simulations) to experimental data. Typical use of PET follows the modeling process discussed in **Section 2**:

1. The user imports an SBML file created by the Model Builder or some other SBML editor.
2. A basal parameter set is created directly from the SBML file or imported from the Run Manager's basal file.
3. Simulation runs are defined in PET or imported from a run file created by the Run Manager.
4. At this point the user may simulate the model, even though experimental data have not yet been defined.
5. Experimental data are defined and transforms set up for the simulation runs.
6. Experimental data and model output are compared by the user (Human Analysis) or by the parameter estimator (Automated Analysis). Parameters are adjusted to seek a better fit of the model to the data.

PET supports cut and paste of experimental data into and from applications, such as Microsoft Excel, copying of plots into presentations or other documents, and generation of PDF files containing plots. PET supports undo and redo of most operations (including all delete operations), semantic checks of user input, and color coding (e.g., of parameters changed by the user in the "Edit Basals" spreadsheet).

The following subsections detail some general features of PET. Specific examples of these features are provided in **Section 4**.

3.3.1 Experimental Data and Transforms

Users enter experimental data and define what transforms to use on the model output in the "Edit Data" screen (**Fig. 16**). Transforms convert the time series data generated by a simulation into a form comparable to the experimental data. For example, experimental data might measure the time it takes for a specific event to happen (timelag) or how much of a species must be added to a system to change a steady state (threshold), or the viability of a mutant. In these cases, the computer simulation must produce a number comparable to the experimental datum (i.e., measuring the same observable). Automated parameter estimation routines then take the difference between the experimental observation and the transformed output of the model, and attempt to minimize this difference by adjusting parameter values. A transform might be quite sophisticated. For example, it might need to analyze the time series output for some measurement (such as cell size) to deduce that an oscillation is taking place, and its period. Transforms are implemented as FORTRAN functions.

The name of every simulation run defined in the “Edit Simulations” screen (**Fig. 17**) appears in the “Edit Data” screen (**Fig. 16**). In the “Edit Data” screen the user can select the name of a simulation run and define experimental data and a transform. Note that some run specifications might not define either experimental data or a transform. These specifications might be inherited by other runs (e.g., the “M-phase” and “Interphase” runs in the example in **Section 4**), or the modeler might wish to store these specifications for another purpose.

3.3.2 Parameter Exploration and Estimation

A user can explore parameter space by setting parameter values (**Fig. 18**), clicking the “Simulate” button, and view the results (**Fig. 19**). This will generate time course plots of selected species (**Fig. 19**). Changes in basal parameters and initial conditions can be made in the “Edit Basals” screen. The user might wish to keep track of multiple basal sets, which are all displayed in the “Edit Basals” screen. When the user clicks the “Simulate” button a simulation is run for each basal set checked in the “Edit Basals” screen, paired with each simulation checked in the “Edit Simulations” screen. For the example, in **Fig. 17** and **Fig. 18**, sixteen simulations are performed: the eight simulations checked in **Fig. 17** are run for each of the two basal parameter sets checked in **Fig. 18**. Every simulation performed generates a plot and the appropriate experimental data from the "Edit Data" screen is plotted with the model simulation points. This allows the user to quickly compare model simulations with experimental data.

By manually changing parameters, running simulations, and viewing plots, a user might discover parameter values that bring the simulations into acceptable agreement with the experimental data. But this manual process is time consuming. PET also provides automated parameter estimation, which searches for parameter values that best fit a model to experiments. Automated parameter estimation can be configured through the "Estimator Settings" screen (**Fig. 20**) and then run with the "Estimate" button.

Two algorithms are currently available in PET for automated parameter estimation: ODRPACK95 ***(21, 22)*** and VTDirect ***(23)***. Both minimize an objective function defined as the weighted sum of squares of the differences between the model and experimental data:

$$E(\beta) = \sum_{i=1}^{n} w_{\varepsilon_i} \varepsilon_i^2 + w_{\delta_i} \delta_i^2, \tag{1}$$

$$f_i(x_i + \delta_i; \beta) = y_i + \varepsilon_i, \quad \text{for} \quad i = 1 \ldots n, \tag{2}$$

where β is the parameter vector (referred to as a parameter set in this chapter), each f_i is a function of the model (e.g., a time course simulation) and could be different for each i, x_i is the i^{th} independent experimental datum (e.g., time), y_i is the i^{th} dependent experimental datum (e.g., species concentration), δ and ε are the respective errors attributed to the independent and dependent experimental data, and w_δ and w_ε are the weights for δ and ε supplied by the user (PET automatically calculates default values for these). The algorithms search for a δ and β to minimize Equation 1 (note that ε can be calculated from Equation 2 once δ and β are chosen). Zwolak et. al. ***(21)*** and Boggs et. al. ***(24, 22)*** explain this objective function in more detail. ODRPACK95 is a local optimization algorithm based on Levenberg-Marquardt. VTDirect is a global optimization algorithm based on the "DIViding RECTangles" algorithm of Jones ***(23)***.

When estimating parameters automatically, the user can select which experiments are to be fit by checking them in the "Edit Simulations" screen (**Fig. 17**). For a particular "estimation," the user might allow only certain parameters to be varied by PET. The fixed parameters might be part of a conserved quantity, have a known value, or are not well constrained by the current data. Such parameters are selected as "fixed" by checking the box in the "Fixed" column of the "Estimator Settings" screen (**Fig. 20**).

Ranges on each parameter can also be defined (and *must* be defined for global optimization with VTDirect). **Fig. 20** shows the "Estimator Settings" screen in PET where the ranges can be edited. When the parameter range extends over multiple orders of magnitude, then the user may wish to use a logarithmic scale by checking the box in the "Log" column. This feature is only available for global estimation and affects the way VTDirect searches parameter space. For example, for a linear scale with a range of 0.01 to 1000 for some parameter p_1, VTDirect might select values of approximately 200, 400, 600, and 800. If a logarithmic scale is selected, the equivalent points selected by VTDirect would be 0.1, 1, 10, and 100. In the linear case, small values of p_1 are never explored, which might not be desirable.

Weights can be assigned to the experimental data to reflect relative confidence in the data in the "Estimator Settings" screen (**Fig. 20**). These are the weights appearing in Equation 1. PET assigns default values for the weights of

$$w_{\delta_i} = \frac{1}{x_i^2+1}, \quad w_{\varepsilon_i} = \frac{1}{y_i^2+1}.$$

These weights can reflect error bounds on the data determined by repeats of the experiment, if available. Larger values for the weight can be assigned for data with small error bounds. Similarly, smaller values for the weight can be assigned for data with large error bounds.

4 A Modeling Example

We now provide a detailed example of how our tools are used to build a model, based on the modeling process described in **Section 2**. The model used here was derived from Marlovits et. al. ***(1)*** and Zwolak et. al. ***(25, 26)*** and can be seen in **Fig. 1**. It models the regulation of entry into mitosis in frog egg extracts by MPF, Cdc25, and Wee1. Experimental data from Kumagai and Dunphy ***(27, 28)*** and Tang et. al. ***(29)*** are fit using local and global optimization. We discuss an alternative model motivated by the parameter set returned from the global optimizer. Implementing the alternative model would continue the modeling process beyond this example.

4.1 Entering the Molecular Network

We begin by entering the molecular network from **Fig. 1** into the Model Builder. Each reaction appears as a line in the reaction spreadsheet (**Fig. 3**). Michaelis-Menten kinetics are used for the forward and reverse reactions of Cdc25 and Wee1. A user defined rate law (My_rate_law) is used to define MPF phosphorylation and dephosphorylation by the active forms of Cdc25 (C_a) and Wee1 (W_a) as well as a small residual activity of the inactive forms of Cdc25 (C_i) and Wee1 (W_i). Two species, L and L2, are added to the model for comparison to measurements of labeled MPF. L is used to measure the rate at which Cdc25 removes the phosphate group from MPF (Kumagai and Dunphy ***(28)*** Figure 3C). L2 is used to measure the rate at which Wee1 adds the phosphate group to MPF (Kumagai and Dunphy ***(28)*** Figure 4B). The map of names used in the model to the biological names can be seen in **Fig. 1**.

The Marlovits ***(1)*** parameter set ($\beta_{Marlovits}$ in Tab. 1) is entered into the Model Builder via the "Parameters" spreadsheet, and exported to a basal file for later use with the Run Manager and PET. Initial conditions for the species are defined in the "Species" spreadsheet for interphase (Tab. 2). Interphase is defined as a state of low MPF and Cdc25 activity and high Wee1 activity.

In some experiments, a buffer is added to an extract, thereby diluting the endogenous concentrations of proteins in the extract. The dilution factor is set to 1 for the experiments from Kumagai and Dunphy Figures 3C and 4B ***(28)***. For the other experiments we use a dilution factor ("Dilution") relative to the Kumagai and Dunphy ***(28)*** experiments. The dilution of species in the model is handled in the Run Manager, as discussed in **Section 4.2**. For Wee1 and Cdc25 we would like the total concentration to be scaled to 1, even after they have been diluted, and this can be specified in the "Rules" spreadsheet of the Model Builder. In the "Species" spreadsheet we create two new species and assign them values in the "Rules" spreadsheet with the rules $\text{CaScaled} = \text{Ca}/\text{Dilution}$ and $\text{WaScaled} = \text{Wa}/\text{Dilution}$.

4.2 Defining Simulation Runs

In this section, we define simulation runs in the Run Manager. Each experiment has a line in the Run Manager and all the values set for the runs can be seen in **Fig. 12**. The Run Manager reads in the SBML file containing our model and the file containing basal parameter values and initial conditions. One way to specify these files is through the "File" menu.

Experiments from Kumagai and Dunphy Figures 3C and 4B ***(28)***, Kumagai and Dunphy Figure 10A ***(27)***, and Tang et. al. Figure 2 ***(29)*** specify what state the extract was in when the experiment began, either interphase or M-phase. Initial conditions for the model are created to mimic these extract states, and the values of these initial conditions can be seen in Tab. 2.

For the initial conditions for M-phase and interphase we create two runs in the Run Manager called "M-phase" and "Interphase" respectively (**Fig. 12**). All runs starting in M-phase will inherit from the M-phase basal run. Similarly, all runs starting in interphase will inherit from the Interphase run.

Experiments in Kumagai and Dunphy Figure 10A ***(27)*** and Tang et. al. Figure 2 ***(29)*** add a buffer that dilutes the extracts by a factor of 0.83 and 0.67, respectively. We handle this by creating a simulation run for each case, called "Dilution = 0.83" and "Dilution = 0.67". These runs set the parameter Dilution to the correct value. Then we create a simulation run "Dilute" that applies the parameter Dilution to all species that are diluted (e.g., $C_T = C_T \cdot \text{Dilution}$, $W_T = W_T \cdot \text{Dilution}$, etc.). The initial conditions for the species are diluted by their assignments (Tab. 2). None of these runs are intended to be simulated. They exist just to be inherited by runs that use diluted species.

4.3 Entering the Experimental Data

With this model we will attempt to reproduce the experimental data from Kumagai and Dunphy ***(27, 28)*** and Tang et. al. ***(29)***. The data from these papers (images of gels, the points on plots, etc.) are quantified in Tab. 3. These data are entered into PET via the "Edit Data" screen (**Fig. 16**). A basal set is defined in the "Edit Basals" screen from the basal file containing the Marlovits parameters. For each experiment the "time series" transform is selected in the "Edit Data" screen, the measured species is selected, and the experimental data is entered so that the optimization code will be able to compare simulation output to the experimental data. Now a set of simulations can be run and we can see how well the Marlovits parameters fit the experimental data (**Fig. 21**).

4.4 Performing Local Parameter Estimation

We choose the Marlovits parameters as an initial guess to be used by the local optimization algorithm ODRPACK95 and set some reasonable lower bounds on the parameters (Tab. 4). Only the simulation runs that we wish to fit to data are checked in the "Simulations" screen of PET, and only the parameters we wish to be estimated are checked in the "Estimator Settings" screen. We use the default settings for ODRPACK95, which, in practice, are usually adequate. As the initial guess we select the "Marlovits (1998)" basal set. The optimizer returns the parameters β_{local} in Tab. 1, and we can compare the results to the Marlovits set by running simulations on the basal set and on the fitted parameter values. (Running the simulation would actually show a window similar to **Fig. 19**, but here we show the plots more compactly in **Fig. 21**). We see from **Fig. 21** that the parameter estimator does return parameters that fit the data better. We can also see that the parameter values are close to the starting value of Marlovits (Tab. 1).

4.5 Global Parameter Estimation

In some cases the user may not have a good starting point for the parameters, or the user might wish to explore parameter space in search of other good parameter sets. PET supports these cases by providing a global parameter estimation algorithm, VT-

Direct. VTDirect requires upper and lower bounds on the parameter values. In our example, we assume that we know little about the true values of the parameters. We give bounds that span several orders of magnitude, and we use a logarithmic scale to distribute the search evenly across these orders of magnitude. Since we use a logarithmic scale, we must set non-zero lower bounds. We set most lower bounds to 10^{-6}, which allows these parameters to get sufficiently close to zero to have a negligible quantitative effect on the model. The bounds are recorded in Tab. 4. VTDirect is run with the settings from Tab. 5, and the resulting parameter set is passed to ODRPACK95 for refinement. We reset the parameter bounds for the ODRPACK95 run to those of Tab. 4. ODRPACK95 does not use the logarithmic scale setting and therefore can have lower bounds of 0 for this run. The global refined parameter set is called β_{global} in Tab. 1.

4.6 Next Steps

Visually, the parameters generated by the global and local optimization runs both fit the experimental data (**Fig. 21**). The parameter sets (β_{local} and β_{global} in Tab. 1) are similar, except for the values of v_c, v_c''', and K_{mc}. For $K_{mc} = 20$ and $C_T = 1$, the Michaelis-Menten rate law for reaction $C_i \rightarrow C_a$ in **Fig. 1** should be replaced by a mass action rate law, $(v_c/K_{mc} \cdot M_a \cdot C_i)$. This change to the model is addressed in Zwolak et. al. ***(26)***, and we will not go through the analysis here.

Next, we can create another variation of the model by adding experimental data for timelags and thresholds, as discussed in Zwolak et. al. ***(25)***. Automated parameter estimation can be run to find parameter values that fit these new experiments, as well as the experiments discussed in this section. The model can continue to be refined and expanded in this way to test further hypotheses and achieve new goals.

The files for the modeling example and its variations are distributed with JigCell and PET and can be found at http://mpf.biol.vt.edu/MMRN_chapter/.

5 Summary

We have demonstrated how a modeler would enter all of the necessary information needed to define, simulate, and validate a model of a molecular regulatory network. Advanced support tools like the JigCell Model Builder make it easy to check the syntactic consistency and completeness of the model. This makes it possible to construct larger models than can be done "by hand" and thus opens the possibility of constructing more complex models than previously possible. The JigCell Run Manager provides a way to organize and manage the information needed to define the ensemble of simulation runs for validating the model against a specific set of experiments. PET provides a tool to help the user compare simulation output to experimental data. PET also provides automated tools for finding "best fitting" values of the rate constants in a model. Our example walks the reader through a complete cycle of entering the model, testing it for initial validity, and using parameter estimation to improve the model.

While tools such as JigCell and PET allow modelers to build and test larger models than were possible before, there is still a long way to go before it will be possible to build models that capture the complex regulatory systems within mammalian cells. Current models are defined as a single monolithic block of reaction equations, an approach that is reaching its limits. In future, modelers will be able to express their models as a collection of interacting components, thus allowing them to build large models from smaller pieces. Improvements are also needed in simulators (including the ability to perform efficient stochastic simulations), in parameter estimation, and in computer performance.

References

1. G. Marlovits, C.J. Tyson, B. Novak, and J.J. Tyson (1998) Modeling M-phase control in *xenopus* oocyte extracts: the surveillance mechanism for unreplicated DNA. *Biophys. Chem.* **72**, 169–184.
2. K.C. Chen, L. Calzone, A. Csikasz-Nagy, F.R. Cross, B. Novak, and J.J. Tyson (2004) Integrative analysis of cell cycle control in budding yeast. *Mol. Biol. Cell* **15**, 3841–3862.
3. A.C. Hindmarsh (1983) ODEPACK: A systematized collection of ODE solvers, in *Scientific Computing*, ed. by R.S. Stepleman, North Holland Publishing Company, 55–64.
4. P. Mendes (1997) Biochemistry by numbers: Simulation of biochemical pathways with Gepasi 3. *Trends in Biochem. Sci.* **22**, 361–363.
5. N.A. Allen, L. Calzone, K.C. Chen, A. Ciliberto, N. Ramakrishnan, C.A. Shaffer, J.C. Sible, J.J. Tyson, M.T. Vass, L.T. Watson, and J.W. Zwolak (2003) Modeling regulatory networks at Virginia Tech. *OMICS* **7**, 285–299.
6. H. Sauro, M. Hucka, A. Finney, C. Wellock, H. Bolouri, J. Doyle, and H. Kitano (2003) Next generation simulation tools: The Systems Biology Workbench and BioSPICE integration. *OMICS* **7**, 355–372.
7. J. Schaff, B. Slepchenko, Y. Choi, J. Wagner, D. Resasco, and L. Loew (2001) Analysis of non-linear dynamics on arbitrary geometries with the Virtual Cell. *Chaos* **11**, 115–131.
8. Y. Cao, H. Li, and L. Petzold (2004) Effi cient formulation of the stochastic simulation algorithm for chemically reacting systems. *J. Chem. Phys.* **121**, 4059–67.
9. M. Gibson, and J. Bruck (2000) Effi cient exact stochastic simulation of chemical systems with many species and many channels. *J. Phys. Chem. A* **104**, 1876–1889.
10. D. Gillespie (2001) Approximate accelerated stochastic simulation of chemically reacting systems. *J. Chem. Phys.* **115**, 1716–1733.
11. M. Vass, C. Shaffer, N. Ramakrishnan, L. Watson, and J. Tyson (2006) The JigCell Model Builder: a spreadsheet interface for creating biochemical reaction network models. *IEEE/ACM Trans. Computat. Biol. and Bioinform.* **3**, 155–164.
12. N. Allen, R. Randhawa, M. Vass, J.W. Zwolak, J.J. Tyson, L.T. Watson, and C. Shaffer (2007) JigCell, `http://jigcell.biol.vt.edu/`.
13. M. Vass, N. Allen, C. Shaffer, N. Ramakrishnan, L. Watson, and J. Tyson (2004) The JigCell Model Builder and Run Manager. *Bioinformatics* **20**, 3680–3681.

14. J.W. Zwolak, T. Panning, and R. Singhania (2007) PET: Parameter Estimation Toolkit, `http://mpf.biol.vt.edu/pet`.

15. M. Hucka, A. Finney, H. Sauro, and 40 additional authors (2003) The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* **19**, 524–531.

16. M. Hucka, A. Finney, B.J. Bornstein, S.M. Keating, B.E. Shapiro, J. Matthews, B.L. Kovitz, M.J. Schilstra, A. Funahashi, J.C. Doyle, and H. Kitano (2004) Evolving a lingua franca and associated software infrastructure for computational systems biology: The systems biology markup language (SBML) project. *Syst. Biol.* **1**, 41–53.

17. H. Sauro, and B. Ingalls (2004) Conservation analysis in biochemical networks: computational issues for software writers. *Biophys. Chem.* **109**, 1–15.

18. B. Ermentrout (2002) *Simulating, Analyzing, and Animating Dynamical Systems: A Guide to XPPAUT for Researchers and Students*, SIAM.

19. StochKit (2005) Project website, `www.cs.ucsb.edu/~cse/StochKit`.

20. E. Conrad (2007) Oscill8, `http://oscill8.sourceforge.net/`.

21. J. Zwolak, P. Boggs, and L. Watson (to appear) Odrpack95: A weighted orthogonal distance regression code with bound constraints. *ACM Trans. Math. Softw.* .

22. P.T. Boggs, J.R. Donaldson, R.H. Byrd, and R.B. Schnabel (1989) Algorithm 676: Odrpack: software for weighted orthogonal distance regression. *ACM Trans. Math. Soft.* **15**, 348–364.

23. D. Jones, C. Perttunen, and B. Stuckman (1993) Lipschitzian optimization without the Lipschitz constant. *J. Optim. Theory. Appl.* **79**, 157–181.

24. P.T. Boggs, R.H. Byrd, and R.B. Schnabel (1987) A stable and effi cient algorithm for nonlinear orthogonal distance regression. *SIAM J. Sci. Stat. Comput.* **8**, 1052–1078.

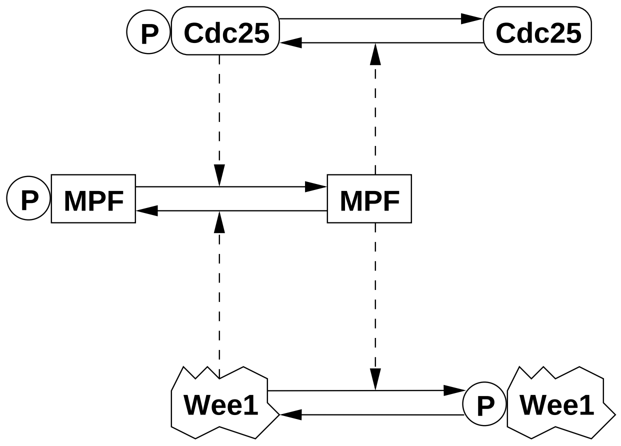
25. J.W. Zwolak, J.J. Tyson, and L.T. Watson (2005) Parameter estimation for a mathematical model of the cell cycle in frog eggs. *J. Comp. Biol.* **12**, 48–63.

26. J.W. Zwolak, J.J. Tyson, and L.T. Watson (2005) Globally optimized parameters for a model of mitotic control in frog egg extracts. *IEE Syst. Biol.* **152**, 81–92.

27. A. Kumagai, and W.G. Dunphy (1992) Regulation of the cdc25 protein during the cell cycle in xenopus extracts. *Cell* **70**, 139–151.

28. A. Kumagai, and W.G. Dunphy (1995) Control of the cdc2/cyclin B complex in *Xenopus* egg extracts arrested at a G2/M checkpoint with DNA synthesis inhibitors. *Mol. Biol. Cell* **6**, 199–213.

29. Z. Tang, T.R. Coleman, and W.G. Dunphy (1993) Two distinct mechanisms for negative regulation of the wee1 protein kinase. *EMBO J.* **12**, 3427–3436.



Species	Description	Phosphorylated
M_a	Active MPF	no
M_i	Inactive MPF	yes
C_a	Active Cdc25	yes
C_i	Inactive Cdc25	no
W_a	Active Wee1	no
W_i	Inactive Wee1	yes

$$\frac{dM_a}{dt} = (v'_c \cdot C_i + v''_c \cdot C_a) \cdot M_i - (v'_w \cdot W_i + v''_w \cdot W_a) \cdot M_a$$
$$\frac{dC_a}{dt} = \frac{v_c \cdot M_a \cdot C_i}{K_{mc} + C_i} - \frac{v'''_c \cdot v_c \cdot C_a}{K_{mcr} + C_a}$$
$$\frac{dW_a}{dt} = -\frac{v_w \cdot M_a \cdot W_a}{K_{mw} + W_a} + \frac{v'''_w \cdot v_w \cdot W_i}{K_{mwr} + W_i}$$

Fig. 1: Network diagram, mapping of species names, and the corresponding set of ordinary differential equations for a model of the mitotic regulatory system in frog eggs. The regulation of MPF (Mitosis Promoting Factor) by Wee1 (kinase) and Cdc25 (phosphatase) controls when the cell enters mitosis. Notice the two positive feedback loops whereby MPF actives Cdc25 (MPF's activator) and inactivates Wee1 (MPF's inactivator). The active forms (M_a, C_a, and W_a) have associated differential equations. The total amounts of MPF (M_T), Wee1 (W_T) and Cdc25 (C_T) are conserved (i.e., remain constant throughout the process). $M_i + M_a = M_T$, $W_i + W_a = W_T$, and $C_i + C_a = C_T$. Therefore, the inactive forms (M_i, C_i, and W_i) do not have differential equations because they can be calculated from these conservation relationships.

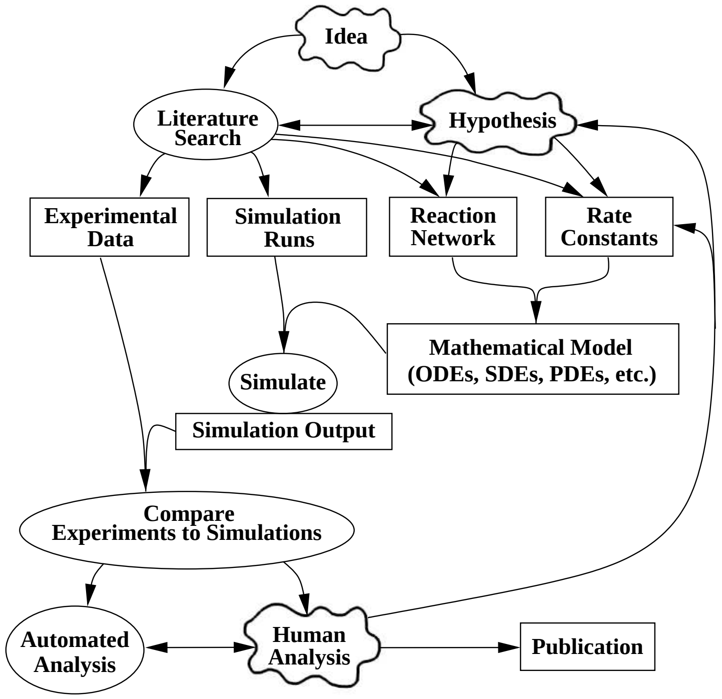
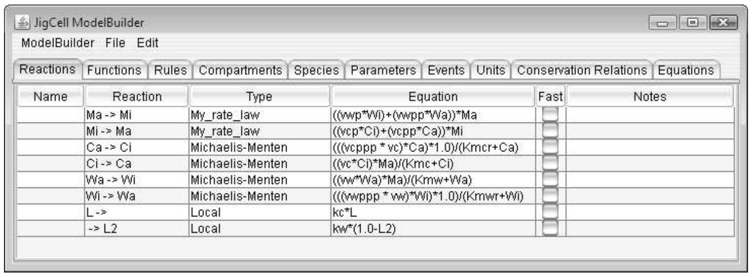
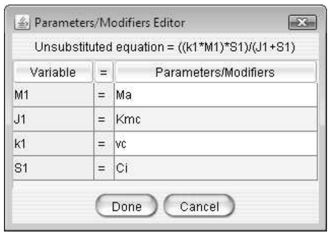


Fig. 2: The modeling process. Once the modeler has generated a testable hypothesis about the organism, he or she must assemble the four necessary collections of information (experimental data, simulation runs, reaction network, and rate constants). This defines both the mathematical model and the behavior that the model must reproduce. The modeler then will repeatedly simulate and update the model, perhaps with the aid of automated analysis tools, until an acceptable result is obtained.


JigCell ModelBuilder
ModelBuilder File Edit
Reactions Functions Rules Compartments Species Parameters Events Units Conservation Relations Equations

Name	Reaction	Type	Equation	Fast	Notes
	Ma -> Mi	My_rate_law	((vwp*Wi)+(vwpp*Wa))*Ma	☐	
	Mi -> Ma	My_rate_law	((vcp*Ci)+(vcpp*Ca))*Mi	☐	
	Ca -> Ci	Michaelis-Menten	(((vcppp * vc)*Ca)*1.0)/(Kmcr+Ca)	☐	
	Ci -> Ca	Michaelis-Menten	((vc*Ci)*Ma)/(Kmc+Ci)	☐	
	Wa -> Wi	Michaelis-Menten	((vw*Wa)*Ma)/(Kmw+Wa)	☐	
	Wi -> Wa	Michaelis-Menten	(((vwppp * vw)*Wi)*1.0)/(Kmwr+Wi)	☐	
	L ->	Local	kc*L	☐	
	-> L2	Local	kw*(1.0-L2)	☐	

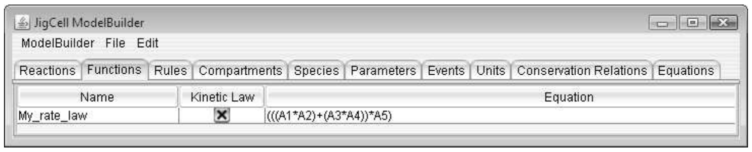
Fig. 3: The “Reactions” spreadsheet.



Unsubstituted equation = ((k1*M1)*S1)/(J1+S1)

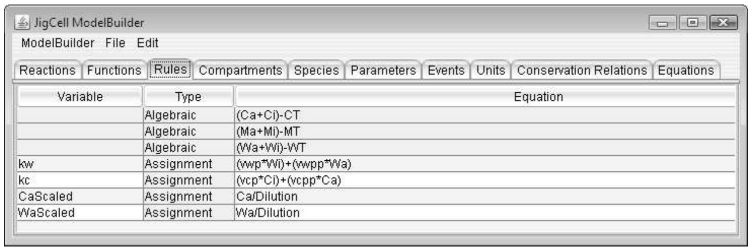
Variable	=	Parameters/Modifiers
M1	=	Ma
J1	=	Kmc
k1	=	vc
S1	=	Ci

Fig. 4: The “Parameters/Modifiers” Editor.


JigCell ModelBuilder
ModelBuilder File Edit
Reactions Functions Rules Compartments Species Parameters Events Units Conservation Relations Equations

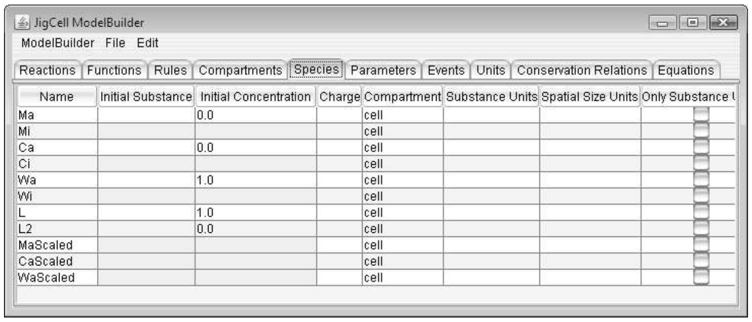
Name	Kinetic Law	Equation
My_rate_law	☒	(((A1*A2)+(A3*A4))*A5)

Fig. 5: The “Functions” spreadsheet.



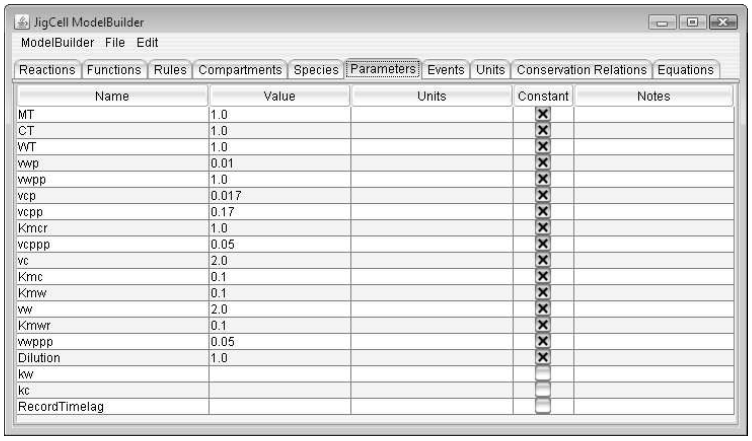
Variable	Type	Equation
	Algebraic	(Ca+Ci)-CT
	Algebraic	(Ma+Mi)-MT
	Algebraic	(Wa+Wi)-WT
kw	Assignment	(vwp*Wi)+(vwpp*Wa)
kc	Assignment	(vcp*Ci)+(vcpp*Ca)
CaScaled	Assignment	Ca/Dilution
WaScaled	Assignment	Wa/Dilution

Fig. 6: The Model Builder "Rules" spreadsheet. The algebraic rules are automatically created by the Model Builder from the conservation relations. The lines for kw and kc define the rates for the reactions of L2 and L, respectively. CaScaled and WaScaled scale the concentrations of Ca and Wa to 1 after they have been diluted by Dilution. See **Section 4.1** for more about dilution.



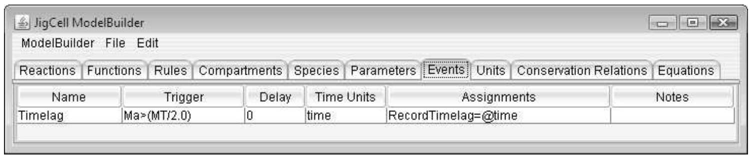
Name	Initial Substance	Initial Concentration	Charge	Compartment	Substance Units	Spatial Size Units	Only Substance
Ma		0.0		cell			☐
Mi				cell			☐
Ca		0.0		cell			☐
Ci				cell			☐
Wa		1.0		cell			☐
Wi				cell			☐
L		1.0		cell			☐
L2		0.0		cell			☐
MaScaled				cell			☐
CaScaled				cell			☐
WaScaled				cell			☐

Fig. 7: The Model Builder “Species” spreadsheet.



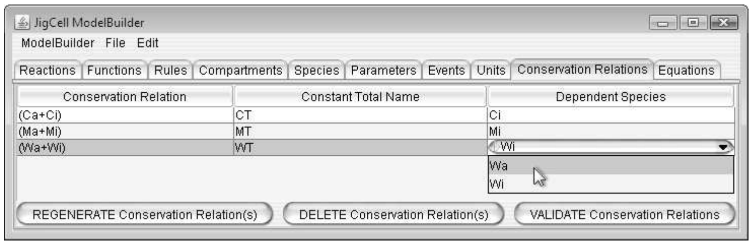
Name	Value	Units	Constant	Notes
MT	1.0		☒	
CT	1.0		☒	
WT	1.0		☒	
vwp	0.01		☒	
vwpp	1.0		☒	
vcp	0.017		☒	
vcpp	0.17		☒	
Kmcr	1.0		☒	
vcppp	0.05		☒	
vc	2.0		☒	
Kmc	0.1		☒	
Kmw	0.1		☒	
vw	2.0		☒	
Kmwr	0.1		☒	
vwppp	0.05		☒	
Dilution	1.0		☒	
kw			☐	
kc			☐	
RecordTimelag			☐	

Fig. 8: The “Parameter” spreadsheet.


JigCell ModelBuilder
ModelBuilder File Edit
Reactions Functions Rules Compartments Species Parameters Events Units Conservation Relations Equations

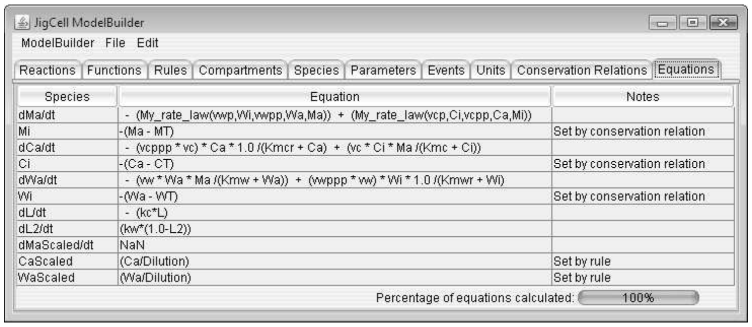
Name	Trigger	Delay	Time Units	Assignments	Notes
Timelag	Ma>(MT/2.0)	0	time	RecordTimelag=@time	

Fig. 9: The “Events” spreadsheet. The symbol “@time” represents time in the system of differential equations. This event sets “RecordTimelag” to the value of time when the “Trigger” becomes true and is used to get the time for active MPF (Ma) to reach half the total MPF concentration (MT). This is provided as an example of how events are defined, but it is not used in the later modeling example.



Conservation Relation	Constant Total Name	Dependent Species
(Ca+Ci)	CT	Ci
(Ma+Mi)	MT	Mi
(Wa+Wi)	WT	Wi

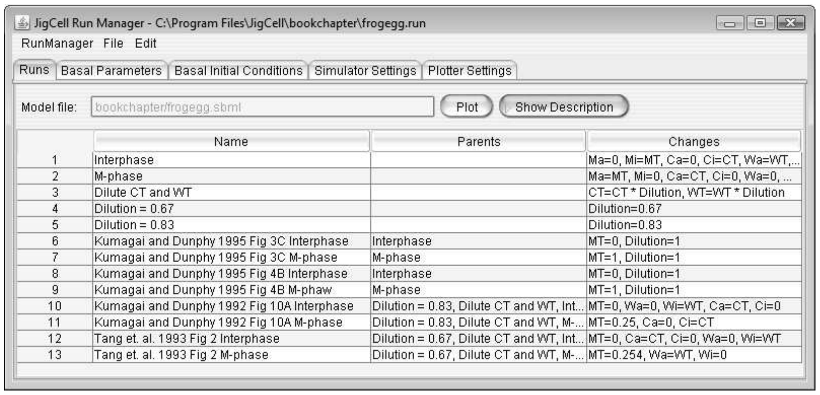
Fig. 10: The “Conservation Relations” spreadsheet.



Species	Equation	Notes
dMa/dt	- (My_rate_law(vwp,Wi,vwpp,Wa,Ma)) + (My_rate_law(vcp,Ci,vcpp,Ca,Mi))	
Mi	-(Ma - MT)	Set by conservation relation
dCa/dt	- (vcppp * vc) * Ca * 1.0 /(Kmcr + Ca) + (vc * Ci * Ma /(Kmc + Ci))	
Ci	-(Ca - CT)	Set by conservation relation
dWa/dt	- (vw * Wa * Ma /(Kmw + Wa)) + (vwppp * vw) * Wi * 1.0 /(Kmwr + Wi)	
Wi	-(Wa - WT)	Set by conservation relation
dL/dt	- (kc*L)	
dL2/dt	(kw*(1.0-L2))	
dMaScaled/dt	NaN	
CaScaled	(Ca/Dilution)	Set by rule
WaScaled	(Wa/Dilution)	Set by rule

Percentage of equations calculated: 100%

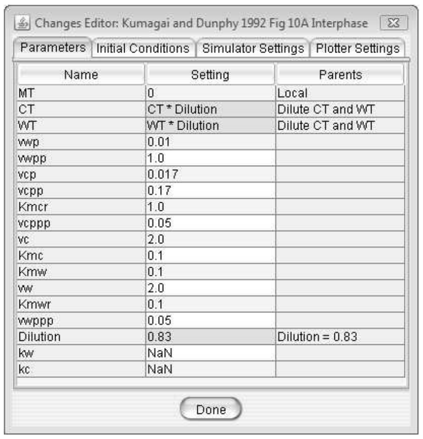
Fig. 11: The “Equations” spreadsheet.



Model file: bookchapter/frogegg.sbml

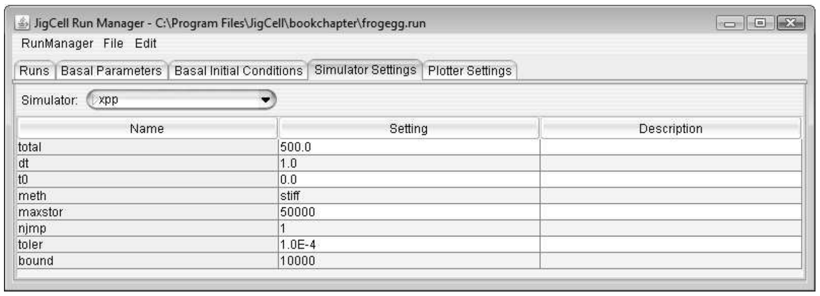
	Name	Parents	Changes
1	Interphase		Ma=0, Mi=MT, Ca=0, Ci=CT, Wa=WT,...
2	M-phase		Ma=MT, Mi=0, Ca=CT, Ci=0, Wa=0, ...
3	Dilute CT and WT		CT=CT * Dilution, WT=WT * Dilution
4	Dilution = 0.67		Dilution=0.67
5	Dilution = 0.83		Dilution=0.83
6	Kumagai and Dunphy 1995 Fig 3C Interphase	Interphase	MT=0, Dilution=1
7	Kumagai and Dunphy 1995 Fig 3C M-phase	M-phase	MT=1, Dilution=1
8	Kumagai and Dunphy 1995 Fig 4B Interphase	Interphase	MT=0, Dilution=1
9	Kumagai and Dunphy 1995 Fig 4B M-phaw	M-phase	MT=1, Dilution=1
10	Kumagai and Dunphy 1992 Fig 10A Interphase	Dilution = 0.83, Dilute CT and WT, Int...	MT=0, Wa=0, Wi=WT, Ca=CT, Ci=0
11	Kumagai and Dunphy 1992 Fig 10A M-phase	Dilution = 0.83, Dilute CT and WT, M-...	MT=0.25, Ca=0, Ci=CT
12	Tang et. al. 1993 Fig 2 Interphase	Dilution = 0.67, Dilute CT and WT, Int...	MT=0, Ca=CT, Ci=0, Wa=0, Wi=WT
13	Tang et. al. 1993 Fig 2 M-phase	Dilution = 0.67, Dilute CT and WT, M-...	MT=0.254, Wa=WT, Wi=0

Fig. 12: The “Runs” spreadsheet.



Name	Setting	Parents
MT	0	Local
CT	CT * Dilution	Dilute CT and WT
WT	WT * Dilution	Dilute CT and WT
vwp	0.01	
vwpp	1.0	
vcp	0.017	
vcpp	0.17	
Kmcr	1.0	
vcppp	0.05	
vc	2.0	
Kmc	0.1	
Kmw	0.1	
vw	2.0	
Kmwr	0.1	
vwppp	0.05	
Dilution	0.83	Dilution = 0.83
kw	NaN	
kc	NaN	

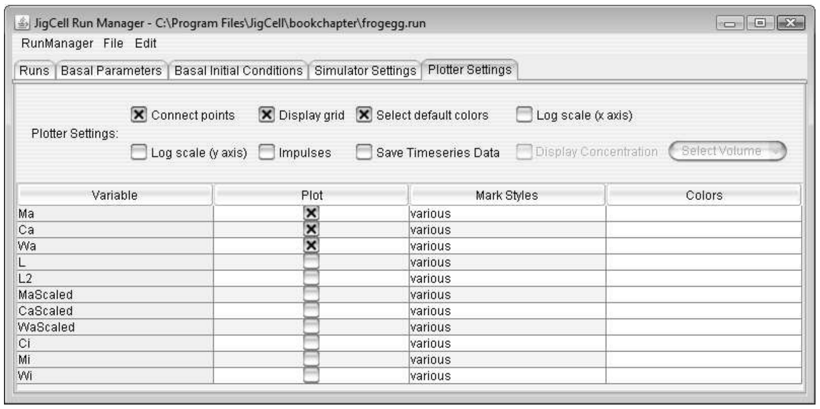
Fig. 13: The “Changes” Editor for a particular run. In the “Setting” column of MT the cell would be colored blue to represent a local change. In the “Setting” column for CT, WT, and Dilution, the cell would be colored green to represent changes inherited from a parent run.



Simulator: xpp

Name	Setting	Description
total	500.0	
dt	1.0	
t0	0.0	
meth	stiff	
maxstor	50000	
njmp	1	
toler	1.0E-4	
bound	10000	

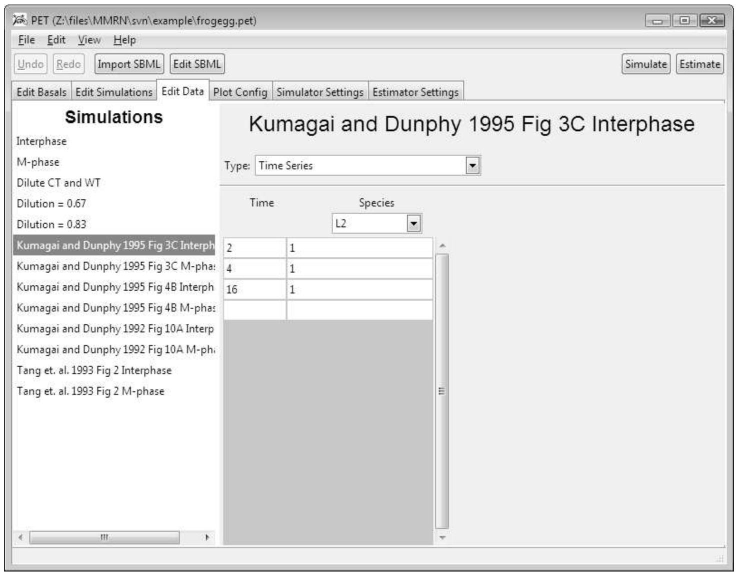
Fig. 14: The “Simulator Settings” spreadsheet.



Plotter Settings: [✓ Connect points] [✓ Display grid] [✓ Select default colors] Log scale (x axis) Log scale (y axis) Impulses Save Timeseries Data Display Concentration

Variable	Plot	Mark Styles	Colors
Ma	✓	various	
Ca	✓	various	
Wa	✓	various	
L		various	
L2		various	
MaScaled		various	
CaScaled		various	
WaScaled		various	
Ci		various	
Mi		various	
Wi		various	

Fig. 15: The “Plotter Settings” spreadsheet.



Simulations

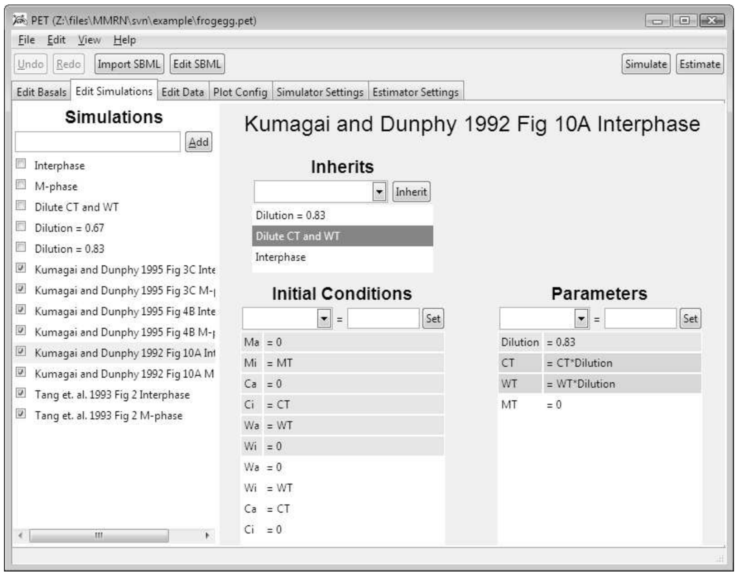
Interphase
M-phase
Dilute CT and WT
Dilution = 0.67
Dilution = 0.83
Kumagai and Dunphy 1995 Fig 3C Interph
Kumagai and Dunphy 1995 Fig 3C M-pha:
Kumagai and Dunphy 1995 Fig 4B Interph
Kumagai and Dunphy 1995 Fig 4B M-phas
Kumagai and Dunphy 1992 Fig 10A Interp
Kumagai and Dunphy 1992 Fig 10A M-ph:
Tang et. al. 1993 Fig 2 Interphase
Tang et. al. 1993 Fig 2 M-phase

Kumagai and Dunphy 1995 Fig 3C Interphase

Type: Time Series

Time	Species L2
2	1
4	1
16	1

Fig. 16: The “Edit Data” screen shows experimental data and the set up for transforms. This figure shows a list of numbers for the time series concentration of L2. The “Time Series” transform is selected for the type of experimental data.


PET (Z:\files\MMRN\svn\example\frogegg.pet)
File Edit View Help
Undo Redo Import SBML Edit SBML Simulate Estimate
Edit Basals Edit Simulations Edit Data Plot Config Simulator Settings Estimator Settings
Add
Inherit
Set

Simulations

- ☐ Interphase
- ☐ M-phase
- ☐ Dilute CT and WT
- ☐ Dilution = 0.67
- ☐ Dilution = 0.83
- ☑ Kumagai and Dunphy 1995 Fig 3C Inte
- ☑ Kumagai and Dunphy 1995 Fig 3C M-
- ☑ Kumagai and Dunphy 1995 Fig 4B Inte
- ☑ Kumagai and Dunphy 1995 Fig 4B M-
- ☑ Kumagai and Dunphy 1992 Fig 10A Int
- ☑ Kumagai and Dunphy 1992 Fig 10A M
- ☑ Tang et. al. 1993 Fig 2 Interphase
- ☑ Tang et. al. 1993 Fig 2 M-phase

Kumagai and Dunphy 1992 Fig 10A Interphase

Inherits

Dilution = 0.83
Dilute CT and WT
Interphase

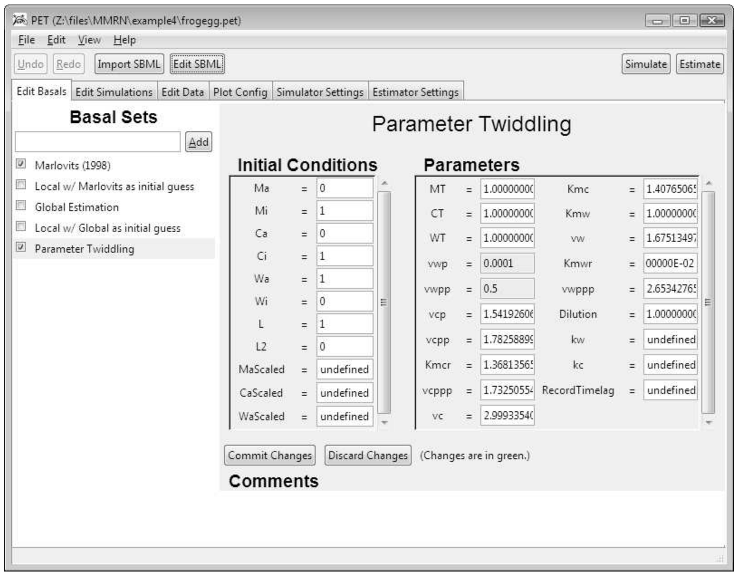
Initial Conditions

Ma	= 0
Mi	= MT
Ca	= 0
Ci	= CT
Wa	= WT
Wi	= 0
Wa	= 0
Wi	= WT
Ca	= CT
Ci	= 0

Parameters

Dilution	= 0.83
CT	= CT*Dilution
WT	= WT*Dilution
MT	= 0

Fig. 17: The "Edit Simulations" screen showing parameter and initial condition values. PET highlights inherited changes in gray. When a parent is selected in the "Inherits" list, the changes inherited from that parent are highlighted in a pastel purple (also shown in gray in this figure).


PET (Z:\files\MMRN\example4\frogegg.pet)

File Edit View Help
Undo Redo Import SBML Edit SBML Simulate Estimate
Edit Basals | Edit Simulations | Edit Data | Plot Config | Simulator Settings | Estimator Settings

Basal Sets

- [✓] Marlovits (1998)
- [ ] Local w/ Marlovits as initial guess
- [ ] Global Estimation
- [ ] Local w/ Global as initial guess
- [✓] Parameter Twiddling

Parameter Twiddling

Initial Conditions

Ma	=	0
Mi	=	1
Ca	=	0
Ci	=	1
Wa	=	1
Wi	=	0
L	=	1
L2	=	0
MaScaled	=	undefined
CaScaled	=	undefined
WaScaled	=	undefined

Parameters

MT	=	1.0000000	Kmc	=	1.4076506
CT	=	1.0000000	Kmw	=	1.0000000
WT	=	1.0000000	vw	=	1.6751349
vwp	=	0.0001	Kmwr	=	00000E-02
vwpp	=	0.5	vwppp	=	2.6534276
vcp	=	1.5419260	Dilution	=	1.0000000
vcpp	=	1.7825889	kw	=	undefined
Kmcr	=	1.3681356	kc	=	undefined
vcppp	=	1.7325055	RecordTimelag	=	undefined
vc	=	2.9993354			

(Changes are in green.)

Comments

Fig. 18: The “Edit Basals” screen lets users define basal sets of initial conditions and parameters. Changes made to parameters and initial conditions are highlighted in green (parameters vwp and vwpp in this figure). The “Commit Changes” button saves changes and removes the highlight colors. Alternatively, the “Discard Changes” button will restore all changed values to the last commit or the original basal set, whichever is more recent.

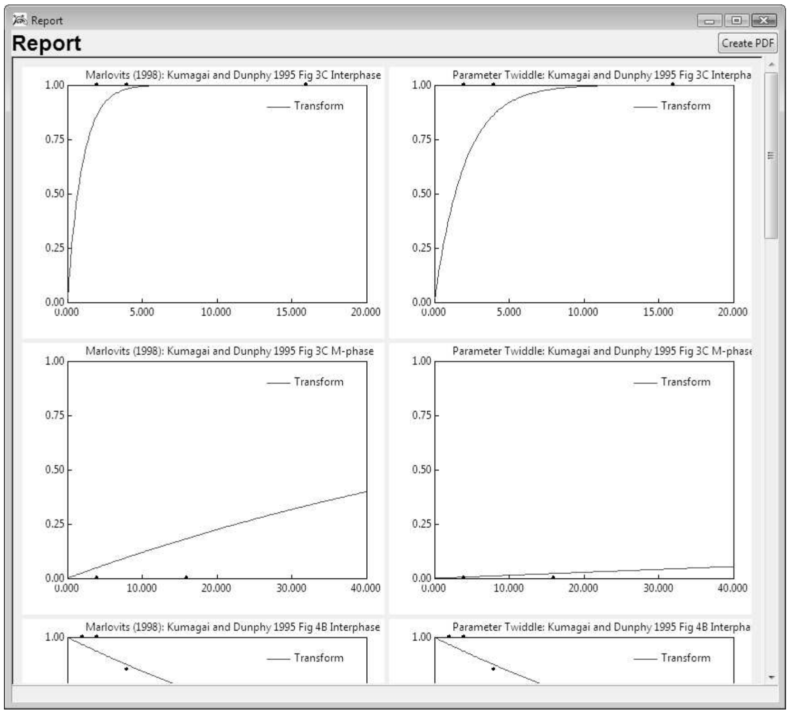
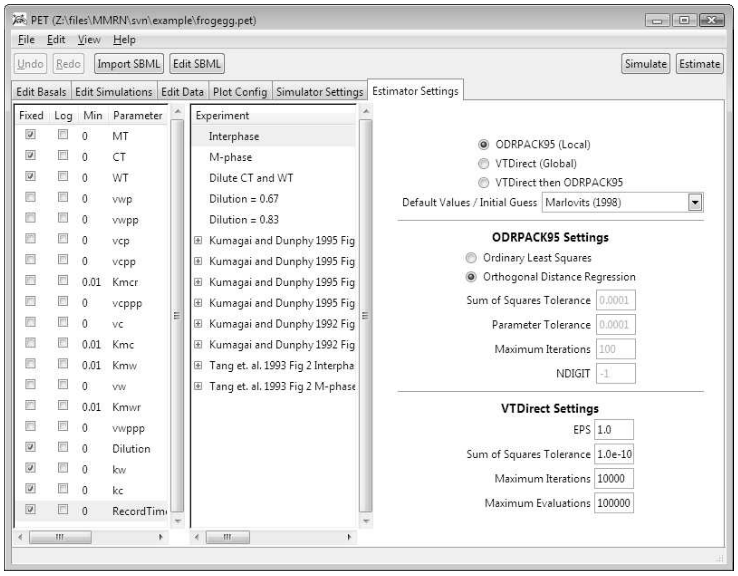

Report
Report
Create PDF
Marlovits (1998): Kumagai and Dunphy 1995 Fig 3C Interphase
Parameter Twiddle: Kumagai and Dunphy 1995 Fig 3C Interpha
Marlovits (1998): Kumagai and Dunphy 1995 Fig 3C M-phase
Parameter Twiddle: Kumagai and Dunphy 1995 Fig 3C M-phase
Marlovits (1998): Kumagai and Dunphy 1995 Fig 4B Interphase
Parameter Twiddle: Kumagai and Dunphy 1995 Fig 4B Interpha
Transform

Fig. 19: The PET report window shows the plots using the basal set named "Marlovits (1998)" (left column) side-by-side with plots using the basal set shown in **Fig. 18** (right column). Each simulation run takes a row in the grid of plots. The simulation run "Kumagai and Dunphy 1995 Figure 3C Interphase" is on the first row, "Kumagai and Dunphy 1995 Figure 3C M-phase" is on the second row, and so forth. As many simulation runs and basal sets will be simulated as the user checks in the "Edit Simulations" (**Fig. 17**) and "Edit Basals" (**Fig. 18**) screens in PET. This feature of PET allows the user to quickly compare multiple basal sets to experiments and assess which basal set best fits experimental data.


PET (Z:\files\MMRN\svn\example\frogegg.pet)
File Edit View Help
Undo Redo Import SBML Edit SBML
Simulate Estimate
Edit Basals Edit Simulations Edit Data Plot Config Simulator Settings Estimator Settings

Fixed	Log	Min	Parameter
✓	☐	0	MT
✓	☐	0	CT
✓	☐	0	WT
☐	☐	0	vwp
☐	☐	0	vwpp
☐	☐	0	vcp
☐	☐	0	vcpp
☐	☐	0.01	Kmcr
☐	☐	0	vcppp
☐	☐	0	vc
☐	☐	0.01	Kmc
☐	☐	0.01	Kmw
☐	☐	0	vw
☐	☐	0.01	Kmwr
☐	☐	0	vwppp
✓	☐	0	Dilution
✓	☐	0	kw
✓	☐	0	kc
✓	☐	0	RecordTim

Experiment
Interphase
M-phase
Dilute CT and WT
Dilution = 0.67
Dilution = 0.83
⊞ Kumagai and Dunphy 1995 Fig
⊞ Kumagai and Dunphy 1995 Fig
⊞ Kumagai and Dunphy 1995 Fig
⊞ Kumagai and Dunphy 1995 Fig
⊞ Kumagai and Dunphy 1992 Fig
⊞ Kumagai and Dunphy 1992 Fig
⊞ Tang et. al. 1993 Fig 2 Interpha
⊞ Tang et. al. 1993 Fig 2 M-phase

[✓ ODRPACK95 (Local)] VTDirect (Global) VTDirect then ODRPACK95

Default Values / Initial Guess: Marlovits (1998)

ODRPACK95 Settings

Ordinary Least Squares [✓ Orthogonal Distance Regression]

Sum of Squares Tolerance	0.0001
Parameter Tolerance	0.0001
Maximum Iterations	100
NDIGIT	-1

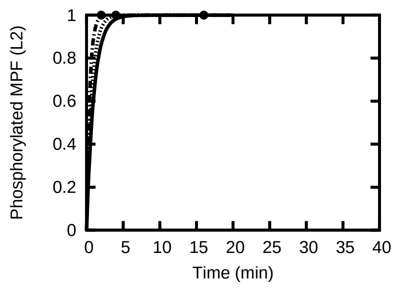
VTDirect Settings

EPS	1.0
Sum of Squares Tolerance	1.0e-10
Maximum Iterations	10000
Maximum Evaluations	100000

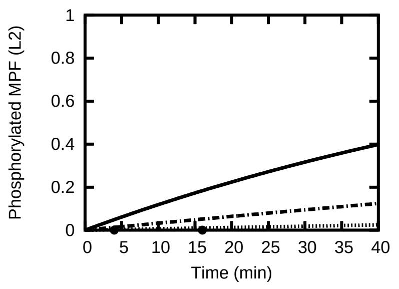
Fig. 20: The “Estimator Settings” screen shows the parameters to be estimated and ranges on those parameters (left), experimental data weights (center), and algorithm settings (right).

Table 1: Parameter sets $\beta_{Marlovits}$ from Marlovits et. al. ***(1)***, β_{local} from the local parameter estimator, and β_{global} from the global parameter estimator. The weighted sum of squares (the value of the objective function E) for each estimated set is shown in the last row.

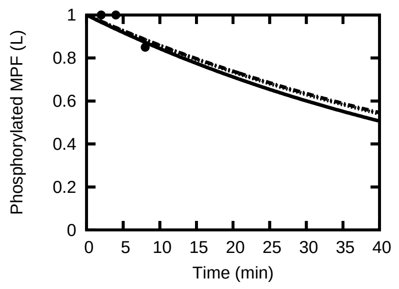
Parameter	$\beta_{Marlovits}$	β_{local}	β_{global}
v_w	2	1.7	3.0
v'_w	0.01	2.4e-4	3.5e-6
v''_w	1	1.4	2.4
v'''_w	0.05	0.027	0.014
v_c	2	3.0	120
v'_c	0.017	0.015	0.015
v''_c	0.17	0.18	0.18
v'''_c	0.05	0.017	0.0027
K_{mw}	0.1	0.01	0.099
K_{mwr}	1	0.01	0.01
K_{mc}	0.1	0.14	20
K_{mcr}	1	0.14	3.4
E		0.018	0.059



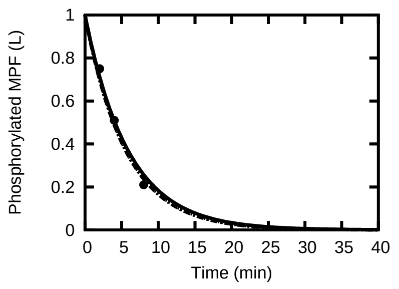
(a) Kumagai 95 Figure 3C Interphase



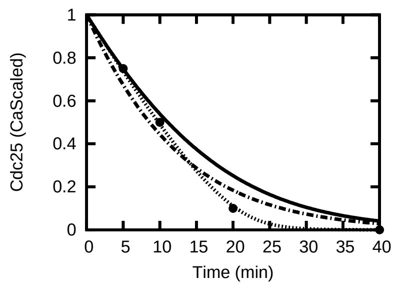
(b) Kumagai 95 Figure 3C M-phase



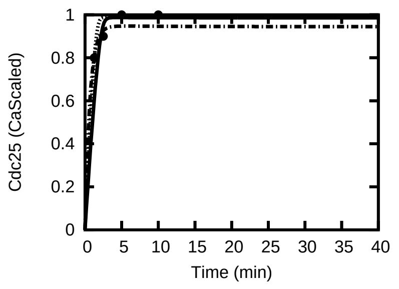
(c) Kumagai 95 Figure 4B Interphase



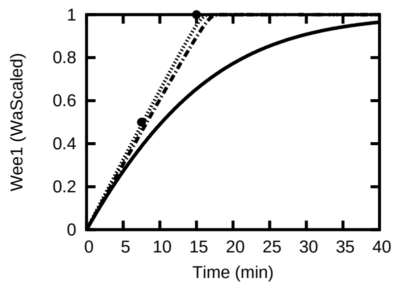
(d) Kumagai 95 Figure 4B M-phase



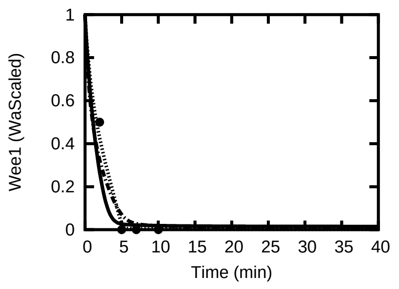
(e) Kumagai 92 Figure 10A Interphase



(f) Kumagai 92 Figure 10A M-phase



(g) Tang 93 Figure 2 Interphase



(h) Tang 93 Figure 2 M-phase

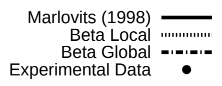


Fig. 21: The parameter set “Marlovits (1998)” ($\beta_{Marlovits}$), “Beta Local” (β_{Local}), and “Beta Global” (β_{Global}) are plotted along with the experimental data for comparison.

Table 2: Initial conditions of the species to model extracts starting in M-phase or Interphase. For example, in Interphase the initial value of inactive MPF (Mi) is set to the total amount of MPF (MT) while the initial value of active MPF (Ma) is set to 0.

Species	M-phase	Interphase
Ma	MT	0
Mi	0	MT
Ca	CT	0
Ci	0	CT
Wa	0	WT
Wi	WT	0

Table 3: Experimental data quantified and compiled for the frog egg extract model.

Experiment	Species	Time	Concentration
Kumagai and Dunphy Figure 3C ***(28)*** Interphase	L2	2	1
		4	1
		16	1
Kumagai and Dunphy Figure 3C ***(28)*** M-phase	L2	4	0
		16	0
Kumagai and Dunphy Figure 4B ***(28)*** Interphase	L	2	1
		4	1
		8	0.85
Kumagai and Dunphy Figure 4B ***(28)*** M-phase	L	2	0.75
		4	0.51
		8	0.21
Kumagai and Dunphy Figure 10A ***(27)*** Interphase	Ca	5	0.75
		10	0.5
		20	0.1
		40	0
Kumagai and Dunphy Figure 10A ***(27)*** M-phase	Ca	1.25	0.8
		2.5	0.9
		5	1
		10	1
Tang et. al. Figure 2 ***(29)*** Interphase	Wa	7.5	0.5
		15	1
Tang et. al. Figure 2 ***(29)*** M-phase	Wa	2	0.5
		5	0
		7	0
		10	0

Table 4: Lower and upper bounds for the parameters. VTDirect will only explore parameter space within these bounds. We use different lower bounds for VTDirect and ODRPACK95, as explained in the text.

Parameter	Lower (VTDirect)	Lower (ODRPACK95)	Upper
v_w	1e-6	0	1e4
v'_w	1e-6	0	1e4
v''_w	1e-6	0	1e4
v'''_w	1e-3	0	100
v_c	1e-6	0	1e4
v'_c	1e-6	0	1e4
v''_c	1e-6	0	1e4
v'''_c	1e-3	0	100
K_{mw}	0.01	0.01	100
K_{mwr}	0.01	0.01	100
K_{mc}	0.01	0.01	100
K_{mcr}	0.01	0.01	100

Table 5: Settings used by VTDirect for the example.

Setting	Value
EPS	1.0
Sum of Squares Tolerance	1.0e-10
Maximum Iterations	1.0e4
Maximum Evaluations	1.0e5