A Modular Workflow for Model Building, Analysis, and Parameter Estimation in Systems Biology and Neuroscience

João P.G. Santos*[,1,2,3], Kadri Pajo*[,2], Daniel Trpevski[1], Andrey Stepaniuk[4], Olivia Eriksson[1], Anu G. Nair[2,5], Daniel Keller[4], Jeanette Hellgren Kotaleski[1,2] Andrei Kramer[1]

[1] Science for Life Laboratory, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, SE-10044 Stockholm, Sweden

[2] Department of Neuroscience, Karolinska Institute, SE-17165 Stockholm, Sweden

[3] Graduate Program in Areas of Basic and Applied Biology, Abel Salazar Institute of Biomedical Sciences, University of Porto, Rua Jorge de Viterbo Ferreira 228, 4050-313, Porto, Portugal

[4] Blue Brain Project, École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

[5] Department of Molecular Life Sciences, University of Zurich, Winterthurerstrasse 190, 8057 Zurich, Switzerland

* Contributed equally to this work

Email of the corresponding author: andreikr@kth.se, jeanette@kth.se

ORCID: 0000-0002-0550-0739 (JHK);0000-0002-0395-039X (JS); 0000-0003-2110-7889 (KP); 0000-0001-9068-6744 (DT); 0000-0002-3828-6978 (AK); 0000-0002-2543-4820 (AS); 0000-0003-3280-6255 (DK); 0000-0002-1952-9583 (AGN)

Information Sharing Statement: Both the source code and documentation of the Subcellular Workflow are available at https://github.com/jpgsantos/Subcellular_Workflow and licensed under GNU General Public License v3.0. The model is stored in the SBtab format (Lubitz et al. 2016). Model reduction, parameter estimation and sensitivity analysis tools are written in MATLAB® (RRID:SCR_001622) and require the SimBiology® toolbox. Conversion script to VFGEN (Weckesser 2008), MOD and SBML (RRID:SCR_007422) is written in R (RRID:SCR_001905). Conversion to SBML requires the use of libSBML (RRID:SCR_014134). Validations are run in COPASI (RRID:SCR_014260; Hoops et al. 2006), NEURON (RRID:SCR_005393; Hines and Carnevale 1997) and with the subcellular application (RRID:SCR_018790; available at https://subcellular.humanbrainproject.eu/model/simulations) that uses a solver provided by STEPS (RRID:SCR_008742; Hepburn et al. 2012). The medium spiny neuron model (Lindroos et al. 2018) used in NEURON simulations is available in ModelDB database (RRID:SCR_007271) with access code 237653.

Acknowledgements: We thank Pavlo Getta for engineering support. We also thank Geir Halnes for sharing his code for GSA with us. The simulations were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at Lunarc.

Abstract

Neuroscience incorporates knowledge from a range of scales from molecular dynamics to neural networks. Modeling is a valuable tool in understanding processes at a single scale or the interactions between two adjacent scales and researchers use a variety of different software tools in the model building and analysis process. Systems biology, for instance, is among the more standardized fields. However, conversion between different model formats and interoperability between various tools is still somewhat problematic. To offer our take on tackling these shortcomings and by keeping in mind the FAIR (findability, accessibility, interoperability, reusability) data principles, we have developed a workflow for building and analyzing biochemical pathway models using pre-existing tools that could be utilized for the storage and refinement of models in all phases of development. We have chosen the SBtab format which allows the storage of biochemical models and associated data in a single file and provides a human readable set of syntax rules. Next, we implement custom-made MATLAB® scripts to perform parameter estimation and sensitivity analysis used in model refinement. Additionally, we have developed a web-based application for biochemical models that allows simulations with either a network free solver or stochastic solvers and incorporating geometry. Finally, we illustrate convertibility and use of a biochemical model in a biophysically detailed single neuron model by running multiscale simulations in NEURON. By this we can simulate the same model in three principally different simulators, describing different aspects of the system, and with a smooth conversion between the different model formats.

Keywords: interoperability, multiscale modeling, SBtab, global sensitivity analysis, parameter estimation, systems biology

Declarations

Funding: The study was supported by the Swedish research council (VR-M-2017-02806), Swedish e-Science Research Centre (SeRC), EU/Horizon 2020 no. 785907 (HBP SGA2) and no. 945539 (HBP SGA3); EPFL Blue Brain Project Fund and the ETH Board Funding to the Blue Brain Project; scholarship PD/BD/114180/2016 from FCT Fundação para a Ciência e Tecnologia

Conflicts of interest/Competing interests: The authors declare that they have no conflict of interest

Ethics approval: not applicable

Consent to participate: not applicable

Consent for publication: not applicable

Availability of data and material: see Information Sharing Statement

Code availability: see Information Sharing Statement

Introduction

Computational systems biology is a data-driven field concerned with building models of biological systems, most commonly the intracellular signaling cascades. Methods from systems biology have been proven valuable in neuroscience, particularly when studying the composition of synapses that can reveal the molecular mechanisms of plasticity, learning and various other neuronal processes (Bhalla and Iyengar 1999; Hellgren Kotaleski and Blackwell 2010; Li et al. 2012). A wide variety of different software and toolboxes, each with their own strengths and weaknesses, are available within the field. This diversity, however, can obstruct model reuse as interoperability between the different software packages and the convertibility between various file types is only solved in part. Interoperability can either mean that the model built in one simulator can be run in another or that both simulators interoperate at run-time either at the same or different scales (Cannon et al. 2014). The former is addressed by standardizing model descriptions, and in systems biology by the standard machine-readable model formats as XML-based SBML (Systems Biology Markup Language; Hucka et al. 2003) and CellML (Hedley et al. 2001), and human readable format as SBtab (Lubitz et al. 2016). An analogous model description language for neurons and networks is the NeuroML (Neural Open Markup Language; Gleeson et al. 2010).

We start by providing examples of the available systems biology tools. We then proceed to describe our approach in developing a modular workflow to address some of these interoperability issues and present simulation results of an example use case in various simulators and frameworks. Our workflow starts with a human-readable representation of the model that is easily accessible to everyone and proceeds through various conversions into different simulation environments: MATLAB, COPASI, NEURON, and STEPS. Specifically, we will describe the conversion tools we created for this purpose.

Examples of Software and Toolboxes used in Systems Biology

No software package is perfectly suited for every task, some have programmable interfaces with scripting languages, like MATLAB's SimBiology toolbox, some focus on providing a fixed array of functions that can be run via graphical user interfaces, like COPASI, although it now offers a Python toolbox for scripting (Welsh et al. 2018). Most toolboxes and software packages offer a mixture of the two approaches: fine-grained programmable interface as well as fixed high-level operations. The extremes of this spectrum are a powerful but inflexible high-level software on one side and a complex, harder to learn but very flexible library or toolbox with an API on the other. Some examples of general modeling toolboxes in MATLAB are the SBPOP/SBToolbox2 (Schmidt and Jirstrand 2006), and the PottersWheel Toolbox (Maiwald and Timmer 2008). Specifically, for Bayesian parameter estimation, there are the MCMCSTAT toolbox (Haario et al. 2006) in MATLAB, as well as pyABC (Klinger et al. 2018) and pyPESTO (Schälte et al. 2020) in Python, and the standalone Markov Chain Monte Carlo (MCMC) software GNU MCsim (Bois 2009). For simulations in neuroscience, examples are NEURON (Hines and Carnevale 1997) and STEPS (Hepburn et al. 2012). Both are used for simulations of neurons and can include reaction-diffusion systems and electro-physiology.

These software packages do not all use the same model definition formats. Most have some compatibility with SBML, others use their own formats (e.g. NEURON uses MOD files). Often an SBML file exported into one of these packages cannot be imported into another package without errors. More generally, SBML has to be translated into code that can be used in model simulations, i.e. the right-hand side of an ordinary differential equation. There are also several tools that facilitate the conversion between formats, e.g. the SBFC (The Systems Biology Format converter; Rodriguez et al. 2016) as well as the more general VFGEN (A Vector Field File Generator; Weckesser 2008).

All toolboxes and software packages have great strengths and short-comings, and each programming language has different sets of (freely) available libraries which makes the development (or use) of numerical methods more (or less) feasible than in another language. One such example is the R package CDvine for parameter dependency modeling using copulas and vines, which has recently been used for modeling parameter spaces in-between MCMC iterations (Eriksson et al. 2019). This package is not easily replaced in many other languages. The Julia language, however, has a far richer set of differential equation solvers than R or MATLAB. Each researcher must therefore make decisions that result in the best compromise for them. If a researcher is familiar with a given set of programming/scripting languages it is probably not reasonable to expect them to be able to collaborate with other groups in languages they do not know. For this reason, it is our firm opinion that the conversion of models between different formats is a very important task and it is equally important to use formats that people can pick-up easily. To make format conversion flexible intermediate files are a great benefit which leads to a modular approach with possible validation between modules.

We also have to consider the FAIR data principles - the findability, accessibility, interoperability and reusability of data and associated infrastructure (Wilkinson et al. 2016). Here, we would like to address the interoperability principle by having developed a workflow for building biochemical pathway models using existing tools and custom-made, short, freely available scripts for the storage and refinement of models in all phases of development, ensuring interchangeability with other formats and toolkits at every step in the pipeline with many standardized intermediate files (Fig. 1). To illustrate all the tools in the workflow we have used it to retune a part of a model previously developed in our research group to make it more compatible with various simulators and to create a concrete use case for others to reuse and modify.

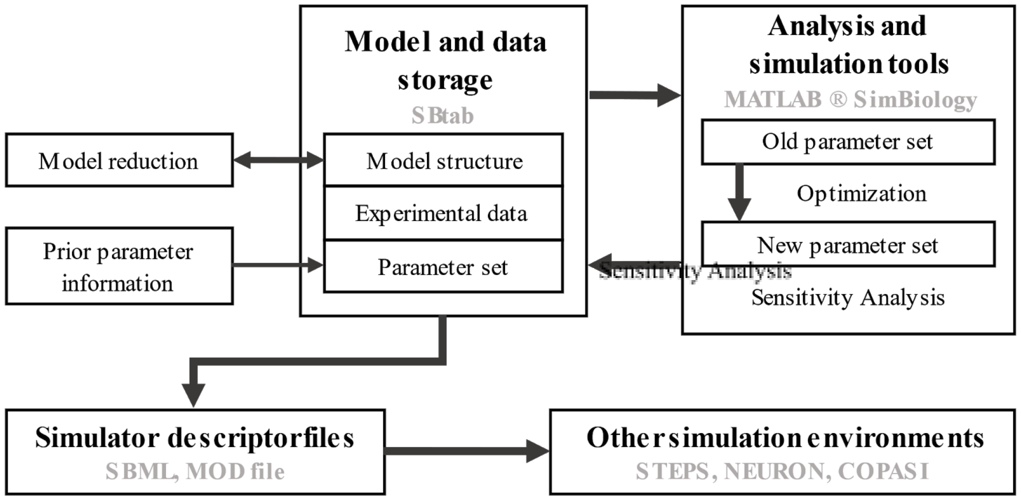

Fig. 1 Simplified scheme of the workflow. Thick arrows indicate steps for which we have developed automated tools. Text in bold refers to the generic components of the workflow and text in grey refers to existing software and data formats used in the current version of the suggested workflow

The workflow

While the standard model storage format in systems biology is SBML, it has a few drawbacks for the reasons mentioned above. Therefore, the workflow is centered around building an easy-to-use infrastructure with models and data expressed in a spreadsheet-based storage format called SBtab. We chose SBtab as the primary modeling source file because it is human-readable, it can contain both the model and the data, and because it is easy to write parsing scripts for it, such as a converter from SBtab to SBML using the libSBML[1] interface in R. This ease of convertibility is used in another focus of the workflow, convertibility between SBtab and other common formats and simulation software, since in systems biology and in any other computational sciences, the lack of compatibility between different tools and formats can often pose problems. A partially working conversion tool between SBtab and SBML had already been developed by the SBtab team. However, it can currently only read one table at a time and does not produce any functional SBML files with our model example. To combat these shortcomings, we wrote scripts to convert the SBtab into SBML, either using the R language or MATLAB, and validating it successfully in COPASI.

The bulk of our workflow is available as MATLAB code, particularly the parameter estimation tools and functions for global sensitivity analysis (SA). SA can be used to determine the importance of different parameters in regulating different outputs. *Local* sensitivity analysis is based on partial derivatives and investigates the behavior of the output when

[1] A library providing an application programming interface for SBML

parameters are perturbed in the close vicinity to a specific point in parameter space. Tools for local SA are already included in the MATLAB suite. Global SA (GSA), on the other hand, is based on statistical approaches and has a much broader range. GSA is more relevant for models that have a large uncertainty in their parameter estimates which is common for systems biology models where many of the parameters have not been precisely measured and the data are sparse.

The standard approach within biochemical modeling is to use deterministic simulations and ordinary differential equations (ODE) that follow the law of mass action as it is computationally efficient and provides good results for sufficiently large well-mixed biological systems. However, this approach has several restrictions in the case of neuronal biochemical cascades. First, such cascades are always subject to stochastic noise, which can be especially relevant in a compartment as small as a dendritic spine where the copy number of key molecules are small enough that the effect of randomness becomes significant (Bhalla 2004). For precise simulation of stochasticity in reaction networks several stochastic solvers are available, e.g. Gillespie's Stochastic Simulation Algorithm (SSA) (Gillespie 1976) and explicit and implicit tau-leaping algorithms (Gillespie 2001). Second, the number of possible states of a biochemical cascade often grows exponentially with the number of simulated molecule types, such that it becomes difficult to represent all these states in the model. In this case, for efficient simulation the reactions in the model could be represented and simulated in a network free form using rule-based modeling approaches (Chylek et al. 2015). To tackle these problems, we developed the subcellular application, a web-based software component for model development. It allows the extension and validation of deterministic chemical reaction network-based models by simulating them with stochastic solvers for reaction-diffusion systems and network free solvers.

Although the workflow is applicable to any biochemical pathway model our emphasis is on modeling biochemical signaling in neurons. Therefore, the last challenge we want to address is an important concept in the interoperability domain of computational neuroscience called multiscale modeling which concerns the integration of subcellular models into electrical models of single cells or microcircuits. This can be achieved either by run-time interoperability between two simulators of different systems or by expanding the capabilities of a single simulation platform as has been done with the NEURON software (McDougal et al. 2013). With this purpose in mind we have written a conversion function from SBtab to the MOD format which is used by NEURON. As such, the inputs and the outputs of a biochemical cascade can be linked to any of the biophysiological measures of the electrical neuron model.

Use case

As a use case to illustrate the workflow we have chosen a previously developed pathway model of the emergence of eligibility trace observed in reinforcement learning in striatal direct pathway medium spiny neurons (MSN) that carry the D1 receptor (Nair et al. 2016). In this model, a synapse that receives excitatory input which leads to an increase in calcium concentration is potentiated only when the signal is followed by a reinforcing dopamine input. Fig. 2a represents a simplified model scheme illustrating these two signaling cascades, one starts with calcium as the input and the other one with dopamine. In simulation experiments the inputs are represented as a calcium train and a dopamine transient (Fig. 3a). Calcium input refers to a burst of 10 spikes at 10 Hz reaching 5 μM. Dopamine input is represented by a single transient of 1.5 μM. The first cascade (species in blue) features the calcium-dependent activation of Ca^{2+}/calmodulin-dependent protein kinase II (CaMKII) and the subsequent phosphorylation of a generic CaMKII substrate which serves as a proxy for long term potentiation (LTP) and is the main output of the model. The second cascade (species in red) represents a G-protein dependent cascade following the dopamine input and resulting in the phosphorylation of the striatal dopamine- and cAMP-regulated phosphoprotein, 32 kDa (DARPP-32) that turns into an inhibitor of protein phosphatase 1 (PP1) which can dephosphorylate both CaMKII and its substrate. The phosphorylation of the substrate is maximal when two constraints are met. First, the time window between the calcium and dopamine inputs has to be short, corresponding to the input-interval constraint which is mediated by DARPP-32 via PP1 inhibition. Second, intracellular calcium elevation has to be followed by the dopamine input, corresponding to the input-order constraint that is mediated by another phosphoprotein, the cyclic AMP-regulated phosphoprotein, 21 kDa (ARPP-21), thanks to its ability to sequester calcium/calmodulin if dopamine arrives first (Fig. 3d).

In the originally published model, CaMKII is autophosphorylated in two compartments, both the cytosol and the post synaptic density (PSD), with a custom-written MATLAB rate function that was calculated based on the probability of two neighboring subunits being fully activated as described in Li et al. (2012). To make it possible to run the model in different software we replace the rate equation of autophosphorylation with a similar set of reactions in both compartments so that the model would only contain bimolecular reactions. The reactions represent a simplified version of the autophosphorylation reactions in Pepke et al. (2010), where in our case only the fully activated CaMKII can be phosphorylated. The same set of reactions is used in both compartments and the schematics is available in Fig. 2b along with the required six new parameters. We used our parameter estimation software to find and constrain parameters that preserved the behavior of the model. We used simulated data from the original model with different timings of the dopamine input relative to the calcium input (Fig. 2c) to obtain a comprehensive picture of its behavior which we want the updated model to reproduce.

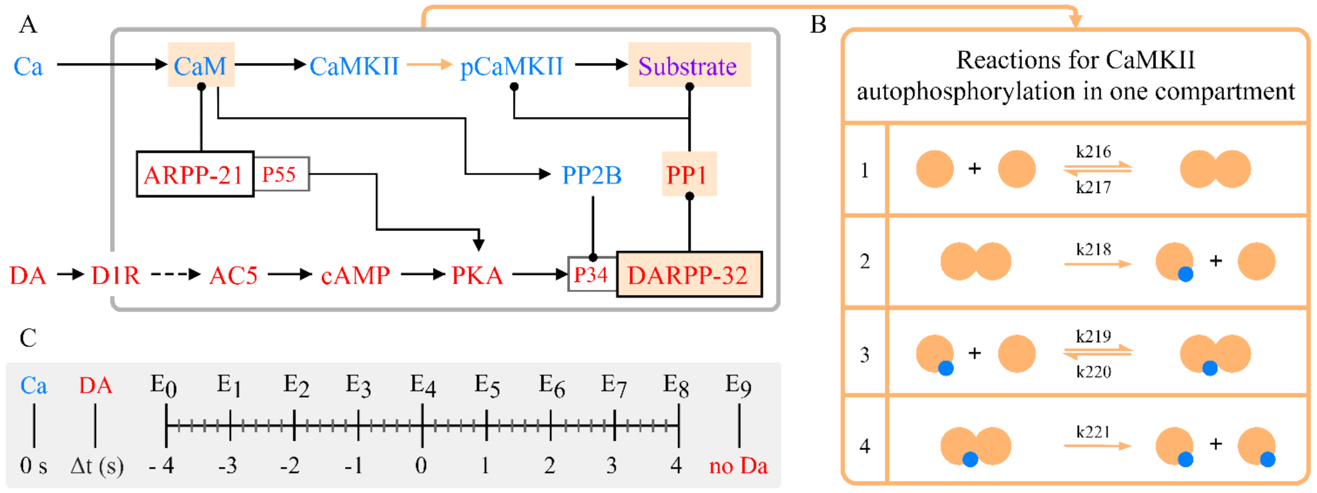

Fig 2 (A) Simplified schematics of the use case model with relevant second messengers with calcium and dopamine as inputs and phosphorylation of a generic CaMKII substrate (purple; top right) as the output. Species of the calcium cascade are blue, and species of the dopamine cascade are red. Lines ending in arrows represent activation and lines ending in circles represent inhibition. Time courses of the species with a beige background are later used in parameter estimation. Readjusted from Nair et al. (2016). (B) Schematics of the bimolecular reactions used for CaMKII autophosphorylation with yellow circles depicting fully activated CaMKII bound to calmodulin and calcium, and blue circles depicting phosphate groups. Six newly introduced parameters are shown on the reaction arrows with their ID's in the updated model. (C) Timing (in seconds) of the dopamine input (Δt={-4,-3,-2,-1,0,1,2,3,4} corresponding to E0-E8) relative to the calcium input (zero), and a single experiment without a dopamine input (E9)

SBtab

As described above we have chosen the SBtab format for model and data storage. This format allows the storage of biochemical models and associated data in a single file and provides a set of syntax rules and conventions to structure data in a tabulated form making it easy to modify and share. To ensure interoperability, SBtab provides an online tool to convert the models into the SBML format. SBtab is suitable for storing data that comes in spreadsheet or table formats, e.g. concentration time series or dose response curves, and it also supports various annotations. The SBtab file is intended to be updated manually during the process of model building. Additional instructions on how to make SBtab files work well within our toolchain can be found in the Subcellular Workflow GitHub repository. Some of the columns and sheets that we use should be considered as extensions to the format. However, SBtab is easy to parse so adjustments to parsers can be made quickly.

The SBtab file should include separate sheets for compartments, compounds, reactions, assignment expressions, parameters, inputs, outputs, and experiments (as well as data tables). The use case model has 99 compounds, 138 reactions and 227 parameters. An example of the SBtab reaction table can be found in Table 1. An important thing to note here is that while most software tools use seconds as their default time units, the NEURON simulator which we use to incorporate the biochemical cascades into an electrical neuron model uses milliseconds instead. We found no easy way to convert the parameter units in SBtab into the file format used by NEURON and hence, the time unit we use in the SBtab sheets representing the model (but not the data) is millisecond.

!ID	*!Name*	*!KineticLaw*	*!IsReversible*	*!Location*	*!ReactionFormula*
R0	ReactionFlux0	kf_R0*GaolfGTP	FALSE	Spine	GaolfGTP <=> GaolfGDP
R1	ReactionFlux1	kf_R1*D1R_Golf_DA	FALSE	Spine	D1R_Golf_DA <=> Gbgolf + D1R_DA + GaolfGTP
R2	ReactionFlux2	kf_R2*D1R_Golf*DA-kr_R2*D1R_Golf_DA	TRUE	Spine	D1R_Golf + DA <=> D1R_Golf_DA
R3	ReactionFlux3	kf_R3*D1R*DA-kr_R3*D1R_DA	TRUE	Spine	D1R + DA <=> D1R_DA
R4	ReactionFlux4	kf_R4*AC5*GaolfGTP-kr_R4*AC5_GaolfGTP	TRUE	Spine	AC5 + GaolfGTP <=> AC5_GaolfGTP
R5	ReactionFlux5	kf_R5*CaM*Ca-kr_R5*CaM_Ca2	TRUE	Spine	CaM + Ca <=> CaM_Ca2
R6	ReactionFlux6	kf_R6*PP2B*CaM-kr_R6*PP2B_CaM	TRUE	Spine	PP2B + CaM <=> PP2B_CaM

Table 1 An example of the tabulated representation of model reactions in the SBtab format. The kinetic law uses the parameter names from the parameter table and species names from the compound table

One of our goals with this study was to reproduce the original model behavior after replacing a single module inside the model to convert the model to bimolecular reactions only. The data we used therefore represents the simulated time series (20 s) of the concentrations of four selected species in response to different input combinations using the original model. Each individual data sheet (named E0-E9, Fig. 2c) in SBtab represents the outputs of one experiment. Another sheet called Experiments allows to define the input parameters differently for each experimental setup. By setting the initial concentrations of the unused species to zero the data could be mapped to a specific sub-module of the model (the remaining species). In this case the initial conditions are the same for all experiments. The Experiments table can also support various annotations relevant to each dataset. We used nine different timings (corresponding to E0-E8) between the calcium and the dopamine signal starting with a dopamine signal preceding calcium by four seconds and finishing with dopamine following calcium after four seconds as this corresponds to the time frame originally used in model development (Δt={-4,-3,-2-1,0,1,2,3,4}). Additionally, we used simulations with calcium as the only input (E9) (Fig. 2). The time series of the input species are in a separate sheet following each experiment sheet. An example of how experimental data is stored can be found in Table 2.

!TimePoint	*!Time*	*>Y0*	*SD_Y0*	*>Y1*	*SD_Y1*	*>Y2*	*SD_Y2*	*>Y3*	*SD_Y3*
E0T0	0	84.47891	1	2672.089	1	3200.526	1	36817.52	1
E0T1	0.01	84.47891	1	2672.089	1	3200.526	1	36817.52	1
E0T2	0.02	84.47891	1	2672.089	1	3200.526	1	36817.52	1
E0T3	0.03	84.47891	1	2672.089	1	3200.526	1	36817.52	1
…									

E0T2000	20	97.79736	1	2359.101	1	3192.558	1	37475.51	1

Table 2 An example of the tabulated representation of experimental time series data in the SBtab format. Columns titled Y0-Y3 refer to each of the output species followed by a standard deviation column

Model pre-processing tools

Model building entails frequent changes to the model structure by adding new species, reactions, and parameters. This can result in the emergence or disappearance of Wegscheider cyclicity conditions that refer to the relationships between reaction rate coefficients arising from conditions of thermodynamic equilibrium (Wegscheider 1901; Vlad and Ross 2004). Identified thermodynamic constraints show parameter dependencies that follow from physical laws and can reduce the number of independent parameters. These conditions are frequently difficult to determine by human inspection, especially for large systems. Similarly, identifying conserved moieties, like conserved total concentration of a protein, allows the reduction of the ODE model size, which leads to increased performance. In order to address these model pre-processing needs our toolkit includes scripts in MATLAB/GNU Octave that use the stoichiometric matrix of the reaction network as an input to determine the thermodynamic constraints as described in Vlad and Ross (2004), and conservation laws. These diagnostic tools output any identified constraints that, if needed, are to be implemented manually before the parameter estimation step. It should be noted that such constraints need to be re-examined after each addition of new reactions as the structure of the model might change and make previously true constraints invalid. This is true for all major changes to the model.

MATLAB tools

The bulk of our workflow is developed in MATLAB as it provides an easy-to-use biochemical modeling application with a graphical user interface called SimBiology along with a wide range of toolboxes for mathematical analysis. The workflow is divided into import, simulation, and analysis scripts. To ensure an easy and user-friendly usage, all operations are controlled by a single settings file where all specification options needing user input are represented as modifiable variables. An example settings file of the use case model along with instructive comments can be found in the GitHub repository.

Import from SBtab to MATLAB

The scripts we have written first convert the SBtab into a .mat format, this file is then used to generate representations of the base model in three different formats: the MATLAB format, the SimBiology format and SBML (level 2 encoding), while also creating scripts coding the input and each experiment inside the newly created ‘Formulas’ and ‘Data’ folders, respectively. All these new files and folders are stored inside the main model folder.

Parameter estimation

MATLAB offers a wide range of tools, such as various optimization algorithms, that can be utilized in model refinement. This allows e.g. the optimization for single point parameter estimation. For this step, we have also developed MATLAB code to automate the use of either of the listed methods with minimal user-defined input. Adjusted parameters can be manually updated in the SBtab or SimBiology model. Our scripts allow the use of Genetic Algorithm, Simulated Annealing, Pattern Search and Constrained Optimization for which MATLAB provides thorough documentation. The equation used to calculate the score for how well the model outputs fit the experimental data can be found below. In addition, we have incorporated a few other ways of calculating the score depending on the need (see documentation).

$$F\,(\theta; Y, \tau) = \sum_{k=1}^{l}\sum_{j=1}^{m}\frac{1}{n}\sum_{i=1}^{n}\left(\frac{Y_{ijk}-y_{ijk}(\theta)}{\tau_{ijk}}\right)^2$$

Here, Y represents the simulation results from the original (validated) model and y are the simulations of the updated model under parameterization θ. The allowed mismatch τ between the two simulation results is analogous to the standard deviation of a Gaussian noise model in data fitting. The resulting F is the objective function for Particle Swarm optimization. The error is summed over n, the number of points in a given experimental output, m, the number of experimental outputs in an experiment which is four in our use case (see Fig. 3b), and l, the number of experiments (E0-E9 in this case) (see Fig. 2c).

Parameter estimation is generally based on experimental data. Here, however, we would like to demonstrate the workflow with a simple use case and therefore the data we used represents the simulated time series of the concentrations of several species using the original version of the model in SimBiology. After modifying the model, we minimized the difference between the old behavior and the updated model’s response through optimization. The simulation results from the old model can be considered as analogous to experimental data in a normal parameter estimation setting. Here we merely aim to make an updated model agree with its earlier iteration, which itself was adjusted based on experimental data. When changing a module in a model it is crucial to protect the unchanged parts which is why we performed parameter estimation using the key species that intersect the calcium and dopamine cascades, namely PP1, calmodulin and DARPP-32 (Fig. 2 and 3b). We performed an optimization with the Particle Swarm algorithm to parameterize the replaced module and the use of the simulated data of these four species proved to be sufficient for the particular task as adding more species did not improve the fits.

Validation in Matlab

Parameter estimation resulted in a good fit for most of the species and the updated model was able to closely reproduce the results seen with the original model (Fig. 3c). The Subcellular Workflow includes the updated model in SBtab, SBML and MATLAB SimBiology along with a script that simulates the model in MATLAB and reproduces the traces of the updated model shown in Fig. 3c-d.

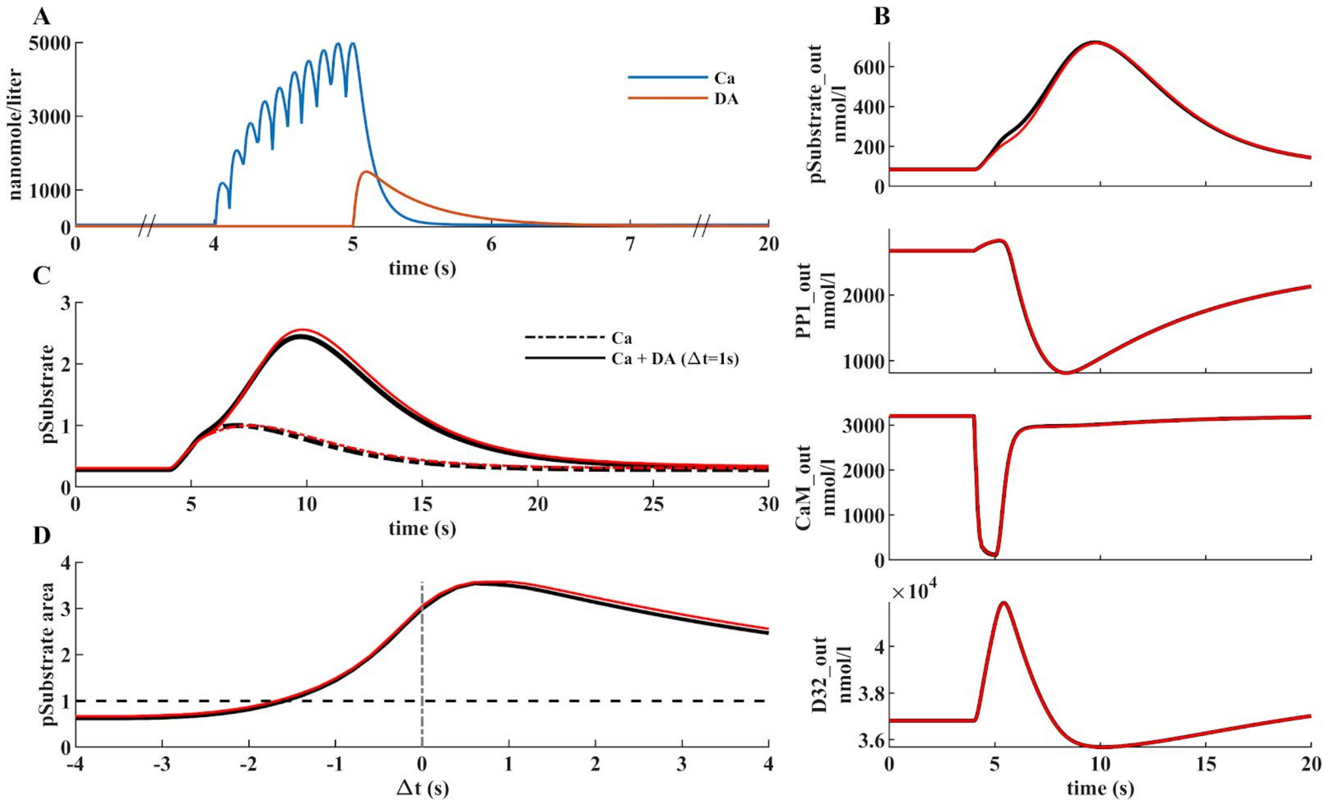

Fig 3 (A) Illustration of the model inputs. Calcium burst (blue) at 4 s used in all simulations and a dopamine transient (orange) applied at different timings in eight experiments and one without it. (B) Four species used in parameter estimation corresponding to the input combination in A. Black traces represent the data produced by simulating the original model, red traces represent fits with the best new parameter sets in the updated model. (C, D) Comparison of model performance with substrate phosphorylation as the main model readout. (C) Normalized time series of substrate phosphorylation, the main readout, with calcium as an only input or dopamine following it after 1 s. (D) Normalized area under the curve of substrate phosphorylation with different calcium and dopamine input intervals

Global sensitivity analysis

In many cases parameter estimation of biochemical pathway models does not result in one unique value for a parameter. Structural and practical unidentifiability (Raue et al. 2009) results in a large set of parameter values that all correspond to solutions with a good fit to the data, i.e. there is a large uncertainty in the parameter estimates (Eriksson et al. 2019). When this is the case *local* sensitivity analysis is not so informative, since this can be different depending on which point in parameter space it is performed at. A global sensitivity analysis (GSA), on the other hand, covers a larger range of the parameter space. Several methods for GSA exist (Zi 2011) but we have focused on a method by Sobol and Saltelli (Sobol 2001; Saltelli 2002; Saltelli 2004) as implemented by Halnes et al. (2009) which is based on the decomposition of variances (Saltelli 2004). Single parameters or subsets of parameters that have a large effect on the variance of the output get a high sensitivity score in this method. Intuitively, this method can be understood as varying all parameters but one (or a small subset) at the same time within a multivariate distribution to determine what effect this has on the output variance. If there is a large reduction in the variance, the parameter that was kept fixed is important for this output (Saltelli 2004).
Let the vector Θ denote the parameters of the model, and $y=f(\Theta)$ be a scalar output from the model. In the sensitivity analysis Θ are stochastic variables, sampled from a multivariate distribution, whose variation gives a corresponding uncertainty of the output, quantified by the variance $V(Y)$. In this setting the different Θ_i are assumed to be independent from each other. We consider two types of sensitivity indices, the *first order effects* S_i and the *total order effects* S_{Ti}. The first order effects describe how the uncertainty in the output depends on the parameter Θ_i, i.e. how much of the variance of the output can be explained by the parameter Θ_i alone. As an example, S_i=0.1 means that 10% of the output variance can be explained by Θ_i alone. The total order effects give an indication on the interactive effect the parameter Θ_i has with the rest of the parameters on the output. Parameters are said to interact when their effect on the output cannot be expressed as a sum of their single effects on the output.
The first order sensitivity index of the parameter Θ_i is defined as $S_i = \frac{V_{\Theta_i}(E_{\Theta_{-i}}(Y|\Theta_i))}{V(Y)}$, where Θ_{-i} corresponds to all elements of Θ except Θ_i, and $E(\ldots|\ldots)$ denotes conditional expected value. For first order sensitivity indices $\sum_i S_i \leq 1$ and for additive models this is an equality (all assuming that the distributions of the different Θ_i are independent from each other). The total order effects of the parameter Θ_i corresponds to $S_{Ti} = \frac{V(Y)-V(E(Y|\Theta_i))}{V(Y)}$, the sum of all S_{Ti} is always larger than or equal to one ($\sum_i S_{Ti} \geq 1$). If there is a large difference between S_i and S_{Ti} this is an indication that this parameter takes part in interactions. For a detailed description see e.g. chapter 5 of Saltelli (2004).

The optimization described earlier takes place on log transformed parameter values (log10(Θ)) and for the sensitivity analysis we perform the sampling on a lognormal distribution, meaning that log10(Θ)~ $N(\mu, \sigma)$. Below we use μ =log10(Θ*) and sigma=0.1, where Θ* correspond to the optimal values received from the optimization. We have also implemented some alternative distributions that can be used. We illustrate this method using only the six parameters corresponding to the model module that has been replaced (Fig. 2) and the results can be seen in Fig. 4. Only four of the parameters (k216-k219/k222-k225; Fig. 4) seem to be important for the output within the investigated parameter region. In experiments 1 and 2 these four parameters are shown to have equal importance, whereas in experiments 4-6 the parameter k219/k225 has the largest influence. Also, there seem to be some interactive effects between the parameters as shown by S_{Ti} analysis (Fig. 4b).

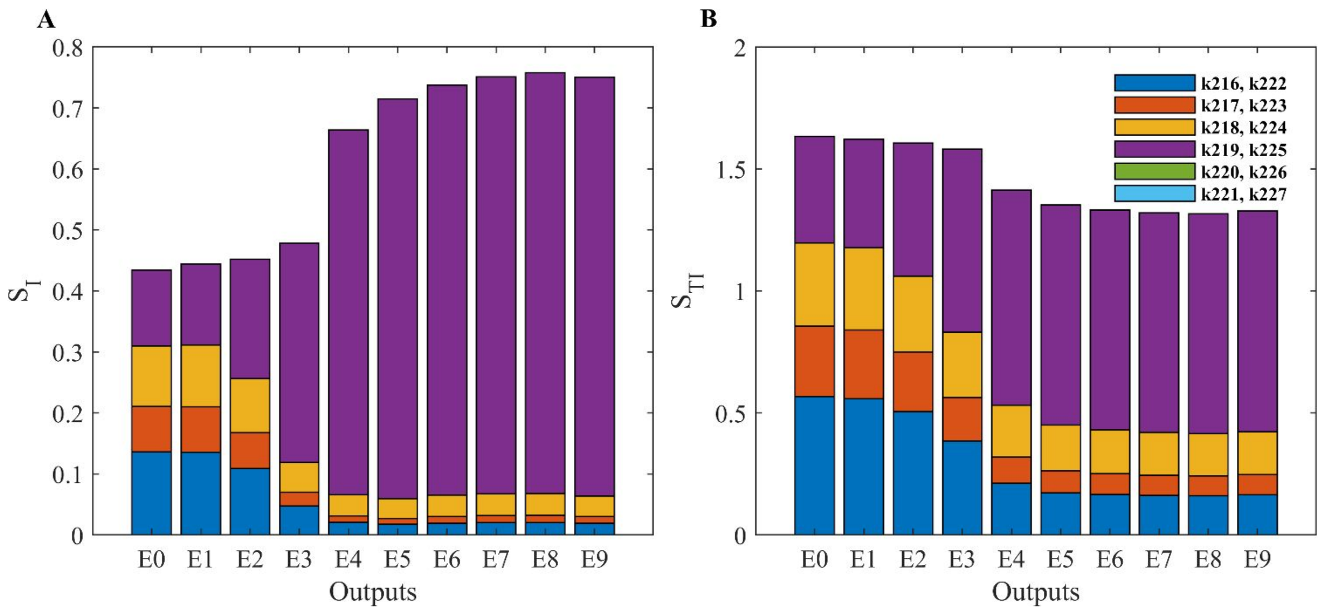

Fig 4 Stacked bar graphs of the sensitivity indexes. The first order, S_i, and total order, S_{Ti}, sensitivities indexes of the six new parameters (k216-k221 and k222-k227; see Fig. 2) for all ten experiments (E0-E9) are shown (panel A and B, respectively). The sensitivities indexes are defined in the main text and were calculated based on the scores used in the optimization for each experiment respectively. The parameters, Θ, were sampled independently from a multivariate lognormal distribution with log10(Θ)~ $N(\mu, \sigma)$, using μ =log10(Θ*) and sigma=0.1, where Θ* correspond to the optimal values received from the optimization. A sample size of N=10000 was used (corresponding to 80000 reshuffled samples used in the calculations (Saltelli 2004)). The analysis took 170 minutes on 50 compute cores (Intel Xeon E5-2650 v3). Smaller differences could be seen in the sensitivity scores when different random seeds were used

Compatibility and validation with other simulation environments

Conversion to SBML and simulations in COPASI

COPASI is one of the more commonly used modeling environments in systems biology and it can read SBML files (Hoops et al. 2006). Our first validation step is to use the SBML model retrieved from the SBtab model in COPASI. As the online conversion tool from SBtab to SBML turned out difficult to understand we wrote a new conversion function that can be found in the GitHub repository of the paper. It interprets the biological model and converts it into plain ODEs in VFGEN's custom format (.vf), the VFGEN file can then be used to create output in various languages[2] (Weckesser 2008). The conversion script (written in R) converts the SBtab saved as a series of .tsv files or one .ods file into a VFGEN vector field file and as by-products also the SBML and a MOD file (see chapter *Conversion to a MOD file and simulations in NEURON*). To create an SBML model libsbml has to be installed with R bindings. The SBML file can be imported directly into COPASI, although it might be necessary to remove the superfluous unit definitions manually.

Another way to convert the model into SBML is through a single MATLAB SimBiology function. The models, however, are created with long ID's that carry no biological information (the ID's are similar to hexadecimal hashes) for all model components, the units are not properly recognized, and some units may just be incorrectly defined in the output. We, therefore, created a script in R that asks for default units for the model and replaces the ones in the SBML file. It fixes most issues (units, ID's, and the time variable in assignments) allowing the model to be properly imported in[3] COPASI. To illustrate that the SBML-converted model imported into COPASI produces the same results as in SimBiology, we used a simplified calcium input corresponding to one double exponential spike analogous to the dopamine transient and simulated the model with deterministic solvers in both COPASI (LSODA solver) and MATLAB SimBiology under similar conditions. Both simulation environments produced almost overlapping results (Fig. 5b-c) validating the converted model in the SBML format.

[2] Including, but not limited to: Python (NumPy), C (GNU Scientific Library (GSL), CVODE), GNU Octave (LSODE), R (deSolve), and also core MATLAB (e.g. for ode15s)

[3] It should be noted that the units may not show up correctly in COPASI (depending on the version) even if they are correct in the SBML file itself.

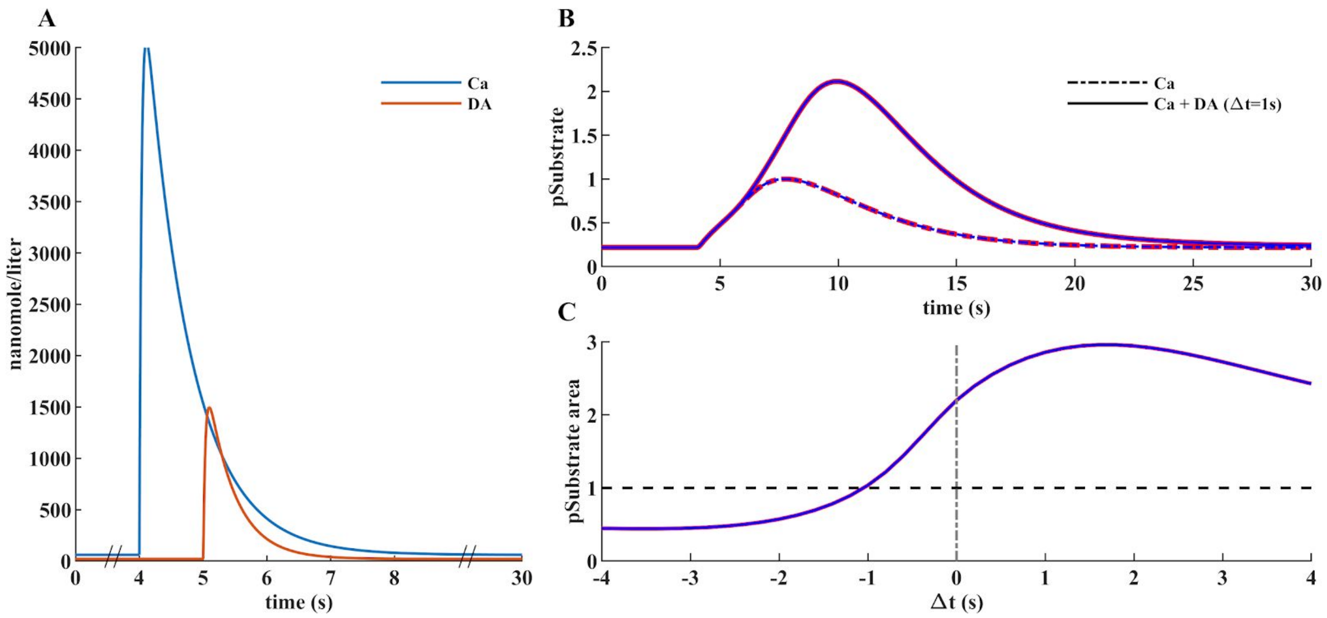

Fig. 5. Simulations in identical conditions in both MATLAB SimBiology and COPASI yielded almost identical results. (A) Inputs used in both simulators. The calcium input is kept constant at 4 s for all simulations and dopamine input time is varied from time 0 to 8 s at every one second. The difference from the previous simulations is in the calcium input which, for the sake of simplicity, is represented by a double exponential spike. (B) and (C) show substrate phosphorylation curves analogous to Fig. 3, the red line represents results obtained in MATLAB and blue line results from simulations in COPASI

Simulations in STEPS

The web-based subcellular application[4] allows importing, combining and simulating models expressed in the BioNetGen language (BNGL; Harris et al. 2016). It supports the import of SBML (level 2) models and their transformation to rule-based BNGL form using Atomizer (Tapia and Faeder 2013). The BioNetGen file format was extended to provide diffusion parameters, links to tetrahedral meshes describing the geometry of model compartments, as well as the additional parameters for solvers and stimulation protocols required for spatially-distributed models. The subcellular application is integrated with the network free solver NFsim (Sneddon et al. 2011) and it supports simulations of spatially distributed systems using STEPS (Hepburn et al. 2012). STEPS provides spatial stochastic and deterministic solvers for simulations of reactions and diffusion on tetrahedral meshes. Furthermore, the subcellular application provides a number of facilities for the visualization of models' geometries and the results of simulations.

To demonstrate the compatibility of the subcellular application with the workflow for model development described above, we imported the SBML version of the use case model to the subcellular application and simulated it with the

[4] The online application for subcellular simulations can be found in https://subcellular.humanbrainproject.eu/model/simulations

STEPS TetOpSplit solver. We have used a simple two-compartmental spine model with a tetrahedral compartment corresponding to the spine compartment of the use case model. There is also a PSD compartment on one of the faces. The results of the model simulations with a STEPS solver were qualitatively similar to the results obtained with the deterministic model simulated in MATLAB. Examples of simulated time courses for molecule concentrations as shown in Fig. 3 in comparison with corresponding MATLAB curves are shown in Fig. 6.

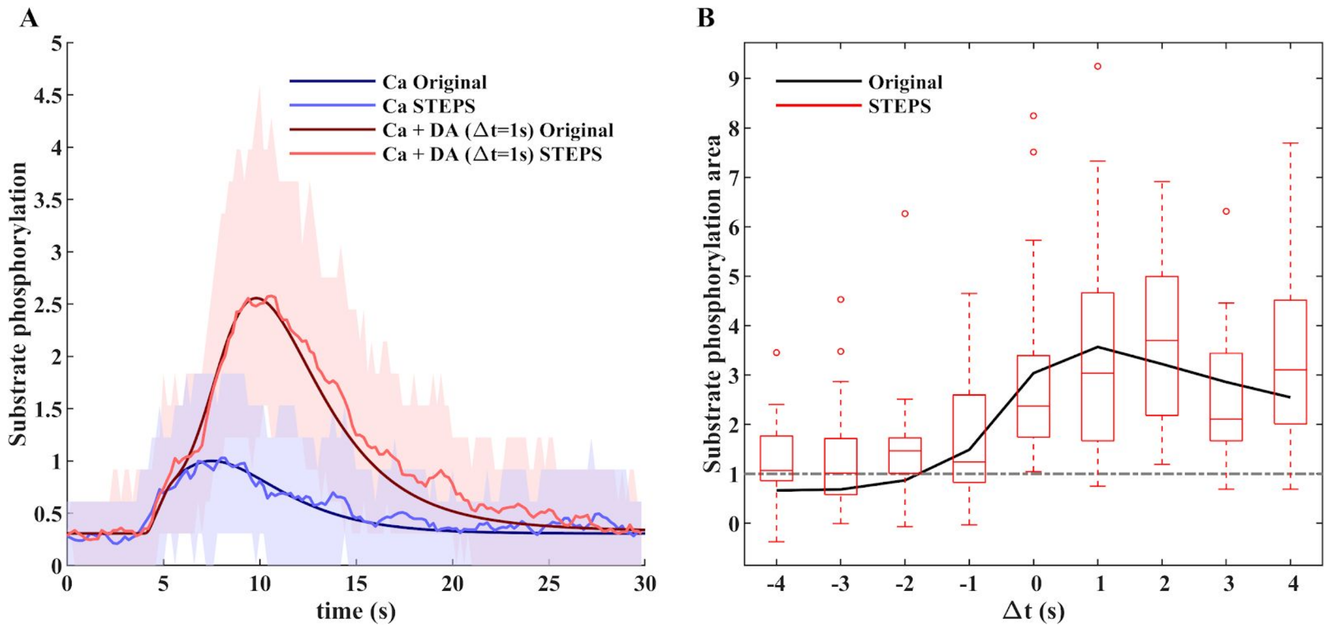

Fig 6 Validation of the model by stochastic STEPS simulation of substrate phosphorylation in a typical D1 MSN neuron spine. (A) Normalized time course of substrate phosphorylation in the original model (violet and orange lines correspond to calcium and dopamine, and calcium only stimuli, respectively) in comparison with averaged (n=50) stochastic STEPS simulations (blue - calcium and dopamine; red - calcium only) for a typical size of D1 MSN neuron synaptic spine ($V=0.02\mu m^3$). The same stimulation protocol as in Fig. 3 was used. Colored areas around averaged STEPS curves correspond to a range between 10% and 90% confidence intervals. (B) Normalized area under the curve of substrate phosphorylation with different calcium and dopamine input intervals simulated for the MATLAB version of the updated model (with MATLAB's ode15s solver, blue line) and averaged stochastic STEPS simulations (n=30) in the subcellular application version of the model. MATLAB statistical bar plots were added to the figure to characterize variability of synaptic plasticity between subsequent induction protocol applications to the same synaptic spine. Note that despite high variability of synaptic plasticity time courses averaged plasticity dynamics were in a good agreement with the ODE-based solution

Conversion to a MOD file and simulations in NEURON

As we suggested before, conversion between different modeling frameworks and formats facilitates collaboration. But conversion is made harder by the differences in the capabilities of different modeling packages. A model, such as the one above, could be useful for multiscale simulations investigating how network activity shapes synaptic plasticity. A large number of cellular level models are built and simulated in the NEURON environment which also supports simplified reaction-diffusion systems. It would thus be useful to be able to integrate a subcellular level model into a cellular level model specified using NEURON. Models in NEURON are built by adding features with MOD files that are written in the NMODL programming language which has very little documentation. While conversion from SBML to MOD is already possible via NeuroML (instructions can be found in Lindroos et al. 2018) it is not an automated or user-friendly approach and therefore we created a new conversion tool from SBtab to MOD which is the same one that creates an SBML file. As one of the options it is also possible to generate a MOD file with or without state variable substitution by observing conservation laws. An example on how to use the SBtab to VFGEN/MOD/SBML converter in R can be found in the GitHub repository. The MOD file is meant to be a starting point for the modeler as the limited subcellular model in SBtab form is not aware of its coupling to a larger model of the cell. The user must edit the resulting file manually to use it within a larger scope and assign a role to this model component.

We validate the biochemical cascade model and our conversion tools in NEURON (see Information Sharing Statement) by qualitatively reproducing the results obtained in MATLAB SimBiology. Our goal is to show that the cascade model can be integrated into a single neuron model and bridge spatial and temporal scales of system behavior by linking the output of the cascade to changes in the synaptic properties and ultimately to the electrical behavior of the neuron model. Therefore, the biochemical model in the MOD format was incorporated into a single biophysically detailed and compartmentalized D1 MSN model from Lindroos et al. (2018). To integrate the MOD file into single cell models a few user-specific modifications have to be made to interface this model with the larger electrochemical system. First, the input is modified so that the calcium burst is represented by calcium influx from the cell's calcium channels and adjusted so that the overall calcium level would be similar to the input used in MATLAB simulations. The dopamine transient is represented by an assignment expression (available in the Expression table of the SBtab) that creates its double exponential form (this can be used in other languages as well). When simulating the model with the same input timing combinations as in the MATLAB experiments, we were able to qualitatively reproduce the substrate phosphorylation curve that illustrates both the input-order and input-interval constraints seen in Fig. 3d. Minor differences can be explained by differences in the calcium input and the solver as MATLAB simulations were performed with ode15s while NEURON uses cnexp.

In the D1 MSNs this biochemical signaling cascade causes synaptic strengthening via several mechanisms, one of which is the phosphorylation of AMPA receptors. As mentioned above, the model instead includes a generic substrate whose level of phosphorylation is the output of the cascade. For the purpose of illustrating the workflow with a proof-of-concept example, we have here linked the fraction of phosphorylated substrate to the AMPA receptor conductance (the conductance is scaled by (1 + fraction of the phosphorylated substrate), i.e. when there is very little substrate phosphorylation, very little change in the AMPA conductance is elicited, and vice versa). Modifying the model to include the reactions for AMPA receptor phosphorylation will be made in a future study.

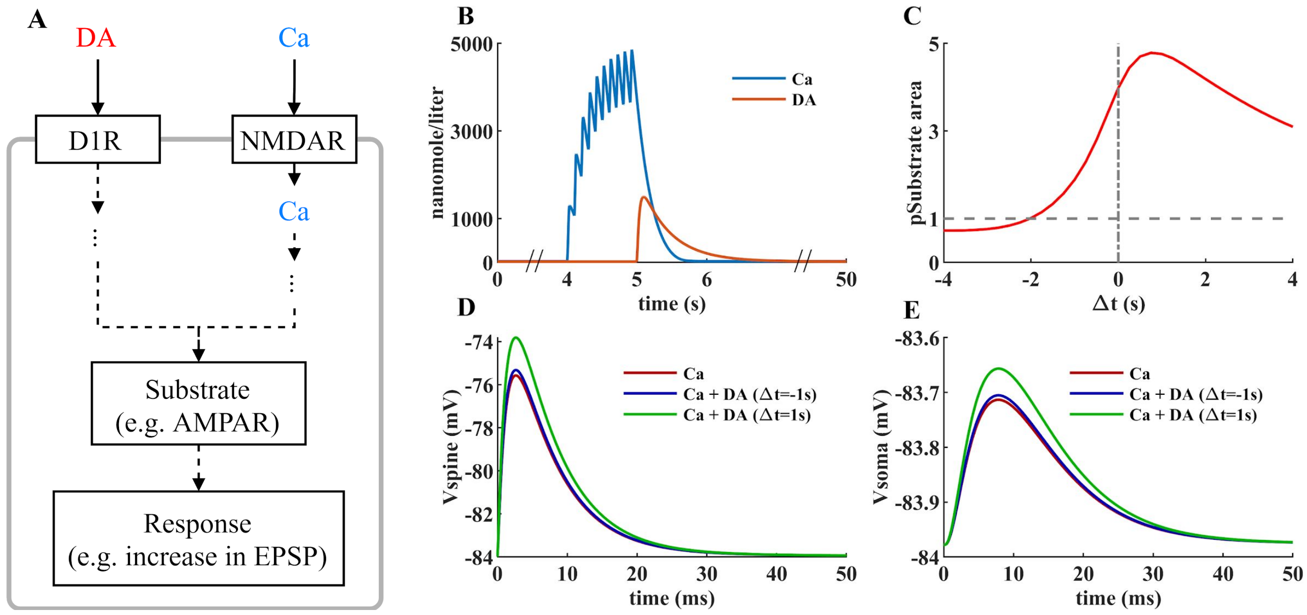

Fig 7 Inserting the biochemical signal transduction cascade into an electrical model in NEURON. (A) A schematic of the effects of the two inputs of the model, dopamine and calcium, on a generic substrate, which in this case is taken to represent the fraction of phosphorylated AMPA receptors with higher conductance levels. (B) Examples of the two inputs, calcium and dopamine. The large calcium signal is due to an NMDA spike evoked via clustered excitatory synapses on a dendritic branch. (C) The calcium input from NMDA receptors also qualitatively reproduces the timing dependence shown previously (Fig. 3d). (D, E) Predicted EPSP following a single synaptic input in the relevant spine and in the soma. The readout of the substrate phosphorylation level was done at 7s after the start of the calcium input. The relative timings of dopamine and calcium indicated with arrows in (C) are used, and the results are compared to the experimental setting without the dopamine input

Discussion

In order to address the growing need for interoperability in biochemical pathway modeling within the neuroscience field, we have developed a workflow that can be used to refine models in all phases of development, keeping in mind the fact that many of the users (including us) are natural scientists and not professional programmers. For the model and data storage we have chosen the SBtab format which can be easily read and modified by both modelers and experimentalists, and can be converted into other formats, e.g. SBML, MATLAB SimBiology or MOD. Our workflow is modularized into different steps allowing the use of each step depending on the need and ensuring interoperability with other tools, such as those described in a similar endeavor named FindSim (Viswan et al. 2018). There are distinct advantages to the workflow, by enforcing a common standard for information exchange, it inherently makes the models more generalizable and reduces the likelihood that simulation results are artifacts of a particular simulator, and nonetheless, it gives users the flexibility to leverage the strengths of each different simulation environment and provides distinct stages of processing that would not be possible in any single simulator. The presented workflow aims to use software components that are free (apart from MATLAB) and solve incremental sub-tasks within the workflow (with open standard intermediate files) to make the workflow easy to branch into scenarios we have not previously considered. It is also possible to circumvent MATLAB entirely, if desired (e.g. conversion from SBtab to a MOD file that is then used by NEURON).

When deciding which software packages to use we find that an important aspect that must be considered is the cost and licensing. For some researchers price may be a relevant concern, in other cases a researcher may have to undergo considerable overhead to make their institution/lab purchase a license and possibly operate a license server. Other than MATLAB, we made the choice to disregard commercial products, keeping in line with the field's trend towards open source platforms. We will also expand our tools to support the use of more complex geometries, with several compartments (which is also an SBML feature), and tetrahedral meshes that can be used with STEPS in the subcellular application. Such advanced geometries can in principle be defined within SBtab tables. When it comes to multiscale simulations, there is the possibility of using the Reaction-Diffusion module (RXD) in NEURON. Currently, however, it does not support the import of SBML as it lacks the concept of spatially extended models, but SBML support might be added to the future versions of RXD (McDougal et al. 2013).

Another interesting consideration is whether the user wants to define any model directly using rules (as in rule-based modeling). This would make the use of Atomizer unnecessary. The transformation from rules to classical reactions seems easier than the reverse, so even if a rule-based simulation is not necessary or too slow, a rule-based description may be shorter and more fundamental in terms of model translation.

When it comes to model analysis, we have here implemented a functionality for global sensitivity analysis. This is a thorough, but, computationally demanding approach and the model needs to be run in parallel on a supercomputer. There are also faster but more approximate screening methods that could have been used (Saltelli 2004). In the future we also intend to incorporate uncertainty quantification into the workflow (Eriksson et al. 2019).

In summary, we have here presented a workflow for biochemical pathway modeling and provided a concrete use case of reward dependent synaptic plasticity. Multiscale models are crucial when trying to understand the brain using modeling and simulations, e.g. how network activity shapes synaptic plasticity or how neuromodulation might affect cellular excitability on sub second timescales. Structured approaches for bridging from detailed cellular level neuron models, to more simplified or abstract cellular-, network-, and even brain region models are developing (Amsalem et al. 2020; Carlu et al. 2020; Schmutz et al. 2020). In the currently illustrated workflow, we add to these efforts by bridging from the subcellular scale to the cellular level scale. We updated parts of the use case model to accomplish a model with only bimolecular reactions that are easier to represent in standards such as SBML. The idea is that users can look at our model as a concrete test case, rerun the workflow (or parts thereof) and then replace the current example model with their own models. In this particular use case, we specifically focused on creating scripts to achieve interoperability between human readable model specification standards and machine- readable standards, and, we also wanted to facilitate how a subcellular signaling model could be implemented in different solvers with different strengths, in this case both SimBiology in MATLAB, as well as STEPS and NEURON. NEURON is currently the most commonly used simulation software for detailed cellular level neuron models, and STEPS can, as said, simulate signaling cascades in arbitrary dendritic morphologies both in a deterministic and stochastic manner. MATLAB on the other hand has a large number of functions, for example for parameter estimation and we included an implementation for global sensitivity analysis. Several other software is, however, used within the computational neuroscience community for cellular or subcellular model simulations (Ray and Bhalla 2008; Oliveira et al. 2010; Resasco et al. 2012; Akar et al. 2019). To successively make as many of those tools interoperable with standards for both model and data specification, various parameter estimation and model analysis methods, visualization software, etc., will further facilitate the creation of FAIR multiscale modeling pipelines in the future.

References:

Akar, N.A., Cumming, B., Karakasis,V., Küsters, A., Klijn, W., Peyser, A., Yates, S. Arbor – A morphologically-detailed neural network simulation library for contemporary high-performance computing architectures. *2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, Pavia, Italy, 2019, pp. 274-282.

Amsalem, O., Eyal, G., Rogozinski, N., Gevaert, M., Kumbhar, P., Schürmann, F., Segev, I. (2020). An efficient analytical reduction of detailed nonlinear neuron models. *Nature Communications,* 11(1), 288.

Bhalla, U.S., Iyengar, R. (1999). Emergent properties of networks of biological signaling pathways. *Science,* 283(5400), 381-387.

Bhalla, U.S. (2004). Models of cell signaling pathways. *Current Opinion in Genetics & Development,* 14, 375–381.

Bois, F.Y. (2009). GNU MCSim: Bayesian statistical inference for SBML-coded systems biology models. *Bioinformatics,* 25, 1453–1454.

Cannon, R. C., Gewaltig, M.-O., Gleeson, P., Bhalla, U. S., Cornelis, H., Hines, M. L., Howell, F.W., Muller, E., Stiles, J.R., Wils, S., De Schutter, E. (2007). Interoperability of neuroscience modeling software: current status and future directions. *Neuroinformatics,* 5(2), 127–138.

Carlu, M., Chehab, O., Dalla Porta, L., Depannemaecker, D., Héricé, C., Jedynak, M., Köksal Ersöz, E., Muratore, P., Souihel, S., Capone, C., Zerlaut, Y., Destexhe, A., di Volo, M. (2020). A mean-field approach to the dynamics of networks of complex neurons, from nonlinear Integrate- and Fire to Hodgkin-Huxley models. *Journal of Neurophysiology,* 123(3), 1042-1051.

Chylek, L.A., Harris, L.A., Faeder, J.R., Hlavacek, W.S. (2015). Modeling for (physical) biologists: an introduction to rule-based approach. *Physical Biology,* 12(4), 045007.

Eriksson, O., Jauhiainen, A., Maad Sasane, S., Kramer, A., Nair, A.G., Sartorius, C., Hellgren Kotaleski, J. (2019). Uncertainty quantification, propagation and characterization by Bayesian analysis combined with global sensitivity analysis applied to dynamical intracellular pathway models. *Bioinformatics,* 35(2), 284–292.

Gillespie, D.T. (1976). A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions. *Journal of Computational Physics,* 22(4), 403–434.

Gillespie, D.T (2001). Approximate accelerated stochastic simulation of chemically reacting systems. *Journal of Chemical Physics,* 115(4), 1716–1733.

Gleeson P., Steuber V., Silver R.A., Crook S. (2012) NeuroML. In: Le Novère N. (eds) *Computational Systems Neurobiology*. Springer, Dordrecht.

Haario, H., Laine, M., Mira, A., Saksman, E. (2006). DRAM: Efficient adaptive MCMC. *Statistics and Computing,* 16, 339-354.

Halnes, G., Ulfhielm, E., Eklöf Ljunggren, E., Hellgren Kotaleski, J., Rospars, J.P. (2009). Modelling and sensitivity analysis of the reactions involving receptor, G-protein and effector in vertebrate olfactory receptor neurons. *Journal of Computational Neuroscience,* 27(3), 471–491.

Harris, L.A, Hogg, J.S., Tapia, J.J., Sekar, J.A., Gupta, S., Korsunsky, I., Arora, A., Baruda, D., Sheehan, R.P., Faeder, J.R (2016). BioNetGen 2.2: advances in rule-based modeling. *Bioinformatics,* 1;32(21), 3366-3368.

Hedley, W. J., Nelson, M. R., Bullivant, D. P., Nielsen, P. F. (2001). A short introduction to CellML. *Philosophical Transactions of the Royal Society,* Series A 359, 1073-1089.

Hellgren Kotaleski, J., Blackwell, K.T. (2010). Modelling the molecular mechanisms of synaptic plasticity using systems biology approaches. *Nature Reviews Neuroscience,* 11, 239–251.

Hepburn, I., Chen, W., Wils, S., De Schutter, E. (2012). STEPS: Efficient simulation of stochastic reaction-diffusion models in realistic morphologies. *BMC Systems Biolog*y, 6:36.

Hines, M.L., Carnevale, N.T., (1997). The NEURON simulation environment. *Neural Computation,* 9, 1179–1209.

Hoops, S., Sahle, S., Gauges, R., Lee, C., Pahle, J., Simus, N., Singhal, M., Xu, L., Mendes, P., Kummer, U. (2006). COPASI: a COmplex PAthway SImulator. *Bioinformatics,* 22, 3067-74.

Hucka, M., Finney, A., Sauro, H. M., Bolouri, H., Doyle, J. C., Kitano, H., et al. (2003). The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics,* 19, 524–531.

Klinger, E., Rickert, D., Hasenauer, J. (2018). pyABC: distributed, likelihood-free inference. *Bioinformatics,* 34(20), 3591-3593.

Li, L., Stefan, M.I.,. Le Novere, N. (2012). Calcium input frequency, duration and amplitude differentially modulate the relative activation of calcineurin and CaMKII. *PLoS One*, 7:e43810.

Lindroos, R., Dorst, M.C., Du, K., Filipović, M., Keller, D., Ketzef, M., Kozlov, A.K., Kumar, A., Lindahl, M., Nair, A.G., Péréz-Fernandez, J., Grillner, S., Silberberg, G., Hellgren Kotaleski, J. (2018). Basal ganglia neuromodulation over multiple temporal and structural scales-simulations of direct pathway MSNs investigate the fast onset of dopaminergic effects and predict the role of Kv4.2. *Frontiers in Neural Circuits,* 12, 3.

Lubitz, T., Hahn, J., Bergmann, F.T., Noor, E.,. Klipp, E, Liebermeister, W. (2016). SBtab: A flexible table format for data exchange in systems biology. *Bioinformatics*, 15;32(16), 2559-61.

Maiwald, T., Timmer, J. (2008). Dynamical modeling and multi-experiment fitting with PottersWheel. *Bioinformatics,* 24(18), 2037-2043.

McDougal, R.A., Hines, M.L., Lytton, W.W. (2013). Reaction-diffusion in the NEURON simulator. *Frontiers in Neuroinformatics,* 7, 28.

Nair, A.G., Bhalla, U.S., Kotaleski J.H. (2016). Role of DARPP-32 and ARPP-21 in the emergence of temporal constraints on striatal Calcium and Dopamine integration. *PLoS Computational Biology,* 1;12(9):e1005080.

Oliveira, R.F., Terrin, A., Di Benedetto, G., Cannon, R.C., Koh, W., Kim M., Zaccolo, M., Blackwell K.T. (2010). The role of type 4 phosphodiesterases in generating microdomains of cAMP: large scale stochastic simulations. *PLoS One,* 5(7):e11725.

Pepke, S., Kinzer-Ursem, T., Mihalas, S., Kennedy, M.B. (2010). A dynamic model of interactions of Ca2+, calmodulin, and catalytic subunits of Ca2+/calmodulin-dependent protein kinase II. *PLoS Computational Biology*, 12;6(2):e1000675.

Raue, A., Kreutz, C., Maiwald, T., Bachman, J., Schilling, M., Klingmüller, U., Timer, J. (2009). Structural and practical identifiability analysis of partially observed dynamical models by exploiting the profile likelihood. *Bioinformatics*, 25(15), 1923-1929.

Ray, S., Bhalla, U.S. (2008). PyMOOSE: Interoperable scripting in Python and MOOSE. Frontiers in Neuroinformatics, 2, 6.

Resasco D.C., Gao, F., Morgan, F., Novak, I.L., Schaff, J.C., Slepchenko, B.M. (2012). Virtual Cell: computational tools for modeling in cell biology. *Wiley Interdisciplinary Reviews: Systems Biology and Medicine,* 4(2), 129-140.

Rodriguez, N., Pettit, J.-B., Dalle Pezze, P., Li, L., Henry, A., van Iersel, M.P., Jalowicki, G., Kutmon, M., Natarajan, K.N., Tolnay, D., Stefan, M.I., Evelo, C.T., Le Novere, N. (2016). The systems biology format converter. *BMC Bioinformatics,* 17, 154.

Saltelli, A., (2002). Making best use of model evaluations to compute sensitivity indices. *Computer Physics Communications*, 145, 280–297.

Saltelli, A., (2004). *Sensitivity analysis in practice: A guide to assessing scientific models*. Chichester, England: Wiley.

Schmidt, H, Jirstrand, M. (2006). Systems Biology Toolbox for MATLAB: a computational platform for research in systems biology. *Bioinformatics,* 22(4), 514–515.

Schmutz, V., Gerstner, W., Schwalger, T. (2020). Mesoscopic population equations for spiking neural networks with synaptic short-term plasticity. *The Journal of Mathematical Neuroscience,* 10(1), 5.

Schälte, Y., Fröhlich, F., Stapor, P., Wang, D., Vanhoefer, J., Weindl, D. et al. (2020). ICB-DCM/pyPESTO: pyPESTO 0.2.0 (Version v0.2.0). Zenodo. http://doi.org/10.5281/zenodo.3928322

Sneddon, M.W., Faeder, J.R., Emonet, T. (2011). Efficient modeling, simulation and coarse-graining of biological complexity with NFsim. *Nature Methods*, 8(2), 177-83.

Sobol, I. M. (2001). Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates. *Mathematics and Computers in Simulation,* 55(1-3), 271-280.

Tapia, J.J, Faeder, J.R. (2013). The Atomizer: extracting implicit molecular structure from reaction network models. In: *Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics (BCB'13)*, ACM, New York, NY, pp. 726–727.

Viswan, N.A., HarshaRani, G.V., Stefan, M.I., Bhalla, U.S. (2018). FindSim: A framework for integrating neuronal data and signaling models. *Frontiers in Neuroinformatics*, 12, 38.

Vlad, M.O., Ross, J. (1994). Thermodynamic approach to nonequilibrium chemical fluctuations. *Journal of Chemical Physics,* 100(10), 7295-7309.

Weckesser, W. (2008). VFGEN: a code generation tool. *Journal of Numerical Analysis, Industrial and Applied Mathematics*, 3, no.1-2, 151–165.

Wegscheider, R. (1901). Über simultane Gleichgewichte und die Beziehungen zwischen Thermodynamik und Reactionskinetik homogener Systeme. *Monatshefte für Chemie,* 22, 849–906.

Welsh, C.M., Fullard, N., Proctor, C.J., Martinez-Guimera, A., Isfort, R.J., Bascom, C.C., Tasseff, R., Przyborski, S.A., Shanley, D.P., (2018). PyCoTools: a Python toolbox for COPASI. *Bioinformatics*, 34(21), 3702-3710.

Zi, Z. (2011). Sensitivity analysis approaches applied to systems biology models. *IET Systems Biology,* 5(6), 336-6.