Title

Model reduction for stochastic CaMKII reaction kinetics in synapses by graph-constrained correlation dynamics

Journal

Physical Biology, 12(4)

ISSN

1478-3967

Authors

Johnson, Todd
Bartol, Tom
Sejnowski, Terrence
et al.

Publication Date

2015

DOI

10.1088/1478-3975/12/4/045005

Peer reviewed

Model Reduction for Stochastic CaMKII Reaction Kinetics in Synapses by Graph-Constrained Correlation Dynamics

Todd Johnson[1], Tom Bartol[2], Terrence Sejnowski[2], and Eric Mjolsness[3]

April 7, 2015

Abstract

A stochastic reaction network model of Ca^{2+} dynamics in synapses [1] is expressed and simulated using rule-based reaction modeling notation in Dynamical Grammars and in MCell. The model tracks the response of calmodulin and CaMKII to calcium influx in synapses. Data from numerically intensive simulations is used to train a reduced model that, out of sample, correctly predicts the evolution of interaction parameters characterizing the instantaneous probability distribution over molecular states in the much larger fine-scale models. The novel model reduction method, "Graph-Constrained Correlation Dynamics" (GCCD), requires a graph of plausible state variables and interactions as input. It parametrically optimizes a set of constant coefficients appearing in differential equations governing the time-varying interaction parameters that determine all correlations between variables in the reduced model at any time slice.

1 Introduction

Given a stochastic reaction network, even one specified by high-level "parameterized reactions" or "rule-based" notation [2-6], there is a corresponding Chemical Master Equation (CME) for the evolution of probability distributions over all possible molecular states of the system. These states are ultimately described in terms of discrete-valued random variables. Unfortunately as the number of such random variables grows, the number of molecular states appearing directly in the CME grows exponentially. On the other hand even for a dynamical system that is nonlinear in its observable variables, the CME is a (very large) system of linear differential equations for time-evolving probabilities. The exponential explosion of state space size with number of random variables can often be bypassed in sampling-style simulation (such as the Gillespie Stochastic Simulation Algorithm (SSA) [7] and its many variants), and also to a lesser extent for reaction rate inference, provided that enough trajectories are sampled to evaluate a required expected value. But the sampling approach requires a lot of computing power to sample enough trajectories, and also poses substantial obstacles for analysis.

The problem of state space growth is compounded in the case of rule-based stochastic models [2,5,6,4] since in that case even the number of molecular species suffers an exponential growth in terms of natural problem size parameters such as the number of binding sites in a molecular complex.

[1] Department of Computer Science, University of California Irvine CA 92697. E-mail: johnson.todd@gmail.com
[2] Salk Institute, La Jolla CA. E-mail: {bartol, sejnowski}@salk.edu
[3] Department of Computer Science, University of California Irvine CA 92697. E-mail: emj@uci.edu

Then the state space described in the master equation grows doubly exponentially in such problem size parameters, and it can be hard to really understand the resulting stochastic dynamical system. And yet, molecular complexes with combinatorially many states (such as transcription complexes, signal transduction complexes, and allosteric enzyme complexes) are ubiquitous in biology, so the problem cannot simply be avoided. For example, the signal transduction cascade in response to calcium influx through NMDA receptors in synapses, which plays a key role in synaptic plasticity in spatial learning and memory in mice, has such combinatorially generated number of local states of molecular complexes involving calcium, calmodulin, CaMKII and phosphorylation sites, as will be described in section 3.1 below ([1,8], and references therein). Each of the many possible states of each complex is itself a "molecular species" with associated reaction channels.

To address these problems of model size and state space size, *reduced models* may have substantial advantages. In general, model reduction replaces a large model with a smaller and more tractable model that approximates the large model in some relevant aspect. The ideas of model "size", "tractability", "approximation", and "relevant aspect" can be defined in various ways. We will suggest one framework for these definitions in section 2.1 below. In sections 2.2 and 2.3 below we use this framework to introduce a new model reduction method for stochastic biochemical networks which in our target application are also rule-based, though they need not be.

Our method can be viewed as a form of "moment closure" method as will be explained in section 2.4, which also contains further comparisons to related work. Compared to other methods of moment closure we seek a much more aggressive reduction in the model size, as counted by the number of degrees of freedom (chemical or biological variables with dynamically changing values) required *even in a sampling approach*, such as SSA, to the original unreduced biochemical model. This claim is substantiated in sections 2.4 and 3.4 below. Such a strategy may be appropriate to the eventual goal of finding usable "phenomenological" but mechanistically well-founded approximate models of fine-scale subsystems to place within yet larger super-system models, for example placing calcium signal pathway reduced models within neuronal-level synaptic plasticity simulations, although we have not yet attempted such an application of this method. Unlike most but not all other moment closure approaches, we retain (approximations to) correlations of arbitrarily high order rather than truncating them to zero. Our approach applies naturally to the case of rule-based models. And perhaps most importantly from the biological point of view, it is based on a problem-specific graph of possible interactions between key system variables. Such a graph is a natural place to impose human biological expertise on the approximate model reduction method.

2 Theory

2.1 Model reduction criteria

Figure 1 illustrates our general setting. The basic idea is that the results of following the red arrows around from earlier to later observations, by way of a fine-scale predictive dynamical model as one would do in an ordinary simulation, should approximately agree with the results of following the green arrows around through a coarse-scale model instead. Since the coarse-scale model is smaller, following the green arrows could be cheaper computationally and also more amenable to human understanding of the dynamics. To define this possibility technically, figure 1 also includes mappings M and $\hat{P}$ directly between the fine-scale and coarse-scale model state spaces.

All vectors and maps defined below are assumed to be defined in the sense of probability distributions, so that for example the fine-scale system state vector $S(t)$ is a distribution over all possible individual microscopic states s. In the deterministic limit that distribution could be a delta function that picks out one winning microstate. Likewise maps $M, \Delta_f[\Delta t]$ etc. take distribution vectors to distribution vectors. The forward time-evolution maps $\Delta_f[\Delta t]$ and $\Delta_c[\Delta t]$ are linear on probability

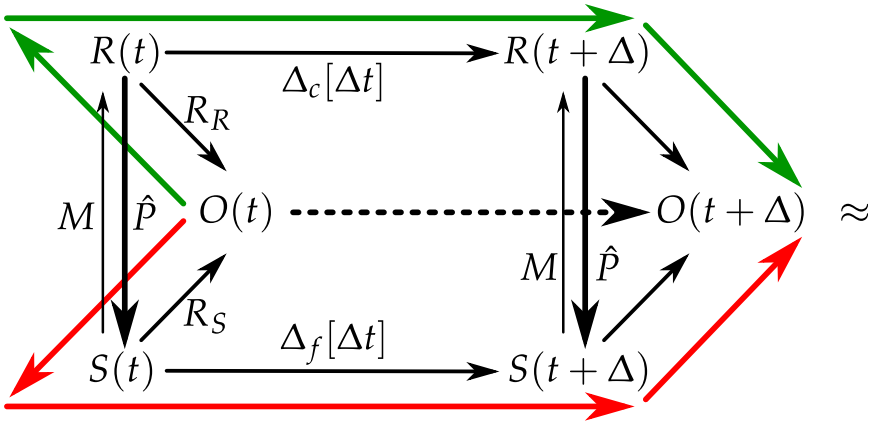

Figure 1: Commutative diagram for model reduction. Fine-scale system S with state $S(t)$ at time t evolves according to fine-scale dynamics $\Delta_f[\Delta t]$ (lower horizontal arrow) to some new state $S(t+\Delta t)$ at time $t+\Delta t$ after (finite or infinitesimal) time interval Δt. Likewise, reduced or coarse-scale system R with state $R(t)$ at time t evolves under reduced coarse-scale dynamics $\Delta_c[\Delta t]$ (upper horizontal arrow). There is a model reduction or restriction map M (vertical arrows), and a prolongation map $\hat{P}$ right-inverse to M. Comparison for approximation ($\approx$) may be made in some relevant third space O, which could specialize (as we assume below) to be the same as S or R. The black arrow quadrilateral comprising two short paths from $S(t)$ to $R(t+\Delta t)$ corresponds to approximation equations (1) and (2). Approximation equations (3) and (4) correspond in their left hand sides to the green highlighted path and in their right hand sides to the red highlighted path. In principle multiple copies of this diagram can be composed, either horizontally or vertically.

distributions, and thus preserve mixtures, but in this paper we do not assume linearity of M or the other maps introduced below.

Fine-scale and (reduced) course-scale dynamics are illustrated in figure 1. A model-reduction or "restriction" map M should ideally *commute*, at least approximately, with time-evolution maps $\Delta_f[\Delta t]$ and $\Delta_c[\Delta t]$; thus for example

$$\Delta_c[\Delta t] \circ M \cdot S(t) \approx M \circ \Delta_f[\Delta t] \cdot S(t), \tag{1}$$

for all $S(t)$ or more strongly for almost all possible S, which in operator space could be stated as

$$\Delta_c[\Delta t] \circ M \approx M \circ \Delta_f[\Delta t]. \tag{2}$$

We will omit operator-space variants of the approximation statements below but they should be clear if the same vector appears at the rightmost end of each side of the approximation. Here and in this section the sense of approximation $\approx$ has yet to be defined but it requires aggregating some measure of error over microstates s, r, or o. For deterministic systems, sum-of-squares error over dynamical variables is plausible; for stochastic systems, an asymmetric Kullback-Leibler (K-L) divergence or relative entropy between two distributions over system states is plausible. The K-L divergence is useful when approximating probability distributions because (a) it measures the extra information contained in one distribution beyond what is in a second distribution, and (b) it takes its minimal value, zero, when the two distributions are equal almost everywhere.

Equations (1) and (2) are not entirely satisfactory since they provide no control over the space in which approximation comparisons are to be made. Alternatively to equation (1), and adopting

terminology used in multigrid/multiscale algorithms [9,10], one could introduce a "prolongation" map $\hat{P}$ that is exactly or approximately right-inverse to M (so that $M \circ \hat{P} = I$ or $M \circ \hat{P} \approx I$), and make the commutativity comparison in the fine-scale system space, S, rather than course-scale, R. But a more general scheme that encompasses both alternatives is to compare time-evolved states or probability distributions on states in a third space of significant "observables", $O(t)$, as shown, using restriction maps $R_S : S \to O$ (and its right powerset inverse or prolongation map P_S for which $R_S \circ P_S = I$) and $R_R : R \to O$ (and its right powerset inverse or prolongation map P_R for which $R_R \circ P_R = I$ if space R is not smaller than O so that R_R can be surjective) that predict the targeted observables based on the current state of each system. Then we seek

$$R_R \circ \Delta_c[\Delta t] \circ P_R \cdot O(t) \approx R_S \circ \Delta_f[\Delta t] \circ P_S \cdot O(t), \tag{3}$$

as illustrated by the red and green three-arrow paths in figure 1. This, or the corresponding operator statement in the O space:

$$R_R \circ \Delta_c[\Delta t] \circ P_R \approx R_S \circ \Delta_f[\Delta t] \circ P_S \tag{4}$$

is our most general statement of the commutation condition.

If we initialize $O(t) = R_S \cdot S(t)$, and assume for consistency the triangular commutation relation $P_R \circ R_S = M$, and define the projection operator $\Pi_S = P_S \circ R_S$, then equation (3) becomes

$$R_R \circ \Delta_c[\Delta t] \circ M \cdot S(t) \approx R_S \circ \Delta_f[\Delta t] \circ \Pi_S \cdot S(t). \tag{5}$$

Two special cases are salient for our computational experiments. In the special case $O = S$, which we will use, then $R_S = I$, $R_R = \hat{P}$, $P_S = I$ and $P_R = M$, (note R_R and P_R exchange roles so that $R_R \circ P_R \neq I$ but instead $P_R \circ R_R = M \circ \hat{P} = I$), and we deduce $\Pi_S = I$ and the foregoing condition becomes

$$\hat{P} \circ \Delta_c[\Delta t] \circ M \cdot S(t) \approx \Delta_f[\Delta t] \cdot S(t). \tag{6}$$

And in the special case $O = R$, which we will also use, $R_R = I$, $R_S = M$, $P_R = I$, and $P_S = \hat{P}$, so equation (3) reverts to

$$\Delta_c[\Delta t] \cdot R(t) \approx M \circ \Delta_f[\Delta t] \circ \hat{P} \cdot R(t), \tag{7}$$

which is closely related to (1).

In all cases some measure of difference or distance is required to define approximation $\approx$; such a measure may operate directly on microstates s, r, o, or on probability distribution state vectors S, R, O over these microstates as we assume below. Particular definitions of $\approx$ will be made in section 3.3 (for $O = R$) and Appendix section 6.1 (for ($O = S$). The foregoing considerations apply to any dynamical system including stochastic, deterministic, and mixed stochastic/deterministic ones.

2.2 Fine- and Coarse-Scale Dynamics

To apply the foregoing framework we need to define fine scale dynamics, coarse scale dynamics, observables, mappings between them, and a sense of approximation ("$\approx$" in figure 1). As in equation 7 above, we will report on the results of taking $O = R$. The approximation metrics will be defined in section 3.3. We now define fine and coarse scale models.

For a Master Equation derived from a large fine-scale reaction network (whether rule-based or not) we seek reduced coarse-scale models in the form of a Boltzmann distribution over states at each time point, with successive time points linked by an artificial and approximating dynamics on the "coupling constant" or interaction parameters (now time-varying rather than constant) appearing in

the Boltzmann energy function formula. In machine learning terms, our prolongation map $\hat{P}$ is given at each instant in time by a probability distribution on fine-scale variables specified by a Markov Random Field (MRF) [11]. The MRF comprises a set of "clique potentials" or "interaction potentials". Each interaction potential is a function, usually a monomial or polynomial, of just a few random variables. Each such potential function is multiplied by a scalar interaction potential strength, which we will call an "interaction parameter". The sum of all the potentials (including their interaction parameter multiplicative factors) yields the energy function in the Boltzmann distribution. Unlike the potentials, the energy function depends on all the random variables. The way we apply this standard apparatus is as follows: The MRF random variables are interpreted as the fine-scale variables, and the MRF interaction parameters are taken to be the coarse-scale model dynamical variables. Only these interaction parameters can vary with time, and in our model they will vary continuously in time. Otherwise, the structure of each interaction potential is fixed and independent of time. Thus, the energy function and the MRF model depend on time only through the interaction parameters which are the coarse-scale dynamical variables.

Without the constraint between successive time points enforced by coarse-scale dynamics on the interaction parameters, the classic Boltzmann Machine Learning Algorithm (BMLA) [12] can be used separately at each time point to optimize these unknown interaction parameters to fit samples drawn from many simulations of the full model. This learning algorithm optimizes the K-L divergence or relative entropy between sampled and modeled distributions, thereby defining a sense for the approximation relationship in section 2.1, but only for one instant in time. We slightly generalize the BMLA learning algorithm so that it allows for weight-sharing (components of the μ interaction parameter vector that are constrained eg. to be equal) and for polynomial interaction potentials of degree higher than two.

Coupling many such interaction-parameter inference problems together by insisting that the inferred interaction parameters μ all evolve according to imposed Ordinary Differential Equation (ODE) dynamics, with a further set of learnable model-specifying meta-parameters θ defined in section 2.3 below, results in the (GCCD) method presented here and in [13].

A large space of stochastic dynamical systems $(S, \Delta_f[\Delta t])$ which can specialize to deterministic ones is specified by the Master Equation governing (possibly singular) probability distributions $p(s,t)$:

$$\frac{dp(t)}{dt} = W \cdot p(t), \tag{8}$$

where W is some linear operator acting on the state space of all distributions p over S and obeying conservation of probability, $\mathbf{1} \cdot W = 0$. Even though the Master Equation is linear, its effect on moments such as $\langle s_i \rangle_p(t)$ may be highly nonlinear.

For stochastic chemical kinetics the Master Equation specializes to the Chemical Master Equation (CME) which can be written:

$$\frac{d}{dt}p([n_i],t) = \sum_r k_r \Big[\big(\prod_j (n_j - S_j^{(r)})_{m_j^{(r)}}\big) p([n_i - S_i^{(r)}]),t) - \big(\prod_j (n_j)_{m_j^{(r)}}\big) p([n_i],t) \Big] \tag{9}$$

where $[n_i]$ is the vector of numbers n_i of molecules of each type i; also in each reaction r, the stoichiometry of the reaction is defined by the following integers: $m_j^{(r)}$ copies of molecule j are destroyed and $n_j^{(r)}$ are created resulting in a net change of $S_j^{(r)} = n_j^{(r)} - m_j^{(r)}$ in the number of molecules of type j; also the notation $(n)_m$ means the falling factorial $n!/(n-m)!$, and k_r is the reaction rate for reaction number r. This fully defines the fine-scale stochastic system for a mass-action chemical reaction network. Using the same notation, the chemical reaction network itself may be expressed as:

$$\forall_r \quad \sum_j m_j^{(r)} A_j \xrightarrow{k_r} \sum_j n_i^{(r)} A_i, \tag{10}$$

where A_i represents reacting chemical species number i and the sums are to be interpreted chemically rather than mathematically.

The Plenum implementation of Dynamical Grammars [2,4] and the MCell Monte Carlo simulation software [3] can be used to express rule-based models and thereby to concisely define combinatorially many elementary reactants and reactions for molecular complexes such as CamKII (which will be introduced in section 3.1 below) and to efficiently simulate them. In addition, spatial diffusion processes can be added. Plenum uses computer algebra to express high-level biological models including those in which the number of compartments varies dynamically and hybrid stochastic/ODE systems. MCell has strong stochastic spatial capabilities and has been used extensively for synapse and neuronal simulations.

For coarse-scale approximate models, if we assume that the state space of the reduced model can be described as the product of state spaces for a fixed set of variables $r = \{\mu_\alpha\}$, then we may consider coarse-scale dynamical systems that through prolongation $\hat{P}$ induce an instantaneous fine-scale probability distribution $\tilde{p}(s,t)$ defined by some Boltzmann distribution

$$\tilde{p}(s|t,\mu) = \exp\Big[-\sum_\alpha \mu_\alpha(t) V_\alpha(s)\Big] / Z(\mu(t)), \tag{11}$$

where $Z(\mu)$ normalizes $\tilde{p}$. This formula separates the time-evolution (which can only affect interaction parameters μ_α) from the correlation-controlling structure of interactions V_α. If there are as many values of α as elements in the full state space of s, then any distribution can be described, but generally we choose a far sparser set of interaction terms. In general equation (11) has nonzero moments of all orders, though only a few moments $-\partial \log Z[\mu]/\partial\mu_\alpha = \langle V_\alpha(s)\rangle$ can be controlled independently by varying the μ_α interaction parameters. This control would be exercised eg. when one derives equation (11) by maximizing entropy subject to equality constraints on these moments $\langle V_\alpha(s)\rangle$ and on total probability. All other moments (which effectively have $\mu_{\alpha'} = 0$) would fall where they may, following the principle of constrained maximum entropy obeyed by the Boltzmann distribution.

The essential information about the coarse-scale dynamics is contained in equation (11) above and equation (12) in section 2.3 below. In this setting, prolongation $\hat{P}$ from coarse to fine is obtained by sampling from the Boltzmann distribution $\tilde{p}(r|t,\mu)$. The model reduction map M will be defined by statistical inference, from a sample of S to μ. Unlike the time-evolution maps $\Delta_f[\Delta t]$ and $\Delta_c[\Delta t]$, neither M nor $\hat{P}$ must necessarily be linear on distributions, and in the special case of optimization algorithms such as Maximum Likelihood (ML) inference, Maximum A Posteriori (MAP) inference, and BMLA, M would be nonlinear due to its optimization of an objective function that is not quadratic in both p and $\tilde{p}$. In sections 2.3 and 3 we will specialize and then apply the theory to stochastic biochemical networks.

2.3 Basis Functions, Smoothing, Test Cases

To define the coarse-scale stochastic model in terms of the time-evolving Boltzmann distribution of equation (11), we hypothesize that even though any particular sample of the stochastic nonlinear system will in general undergo discontinous evolution, the probability distribution governing the ensemble of such samples are likely to evolve continuously in time (as does the Master Equation itself) even when projected down to distributions described by the statistical interaction parameters μ. We therefore further hypothesize continuous and deterministic ODE dynamics for the $\mu(t)$ interaction parameters:

$$\frac{d}{dt}\mu_\alpha(t) = f_\alpha(\mu(t)) = \sum_A \theta_{\alpha A} f_A(\mu(t)), \tag{12}$$

which is linear in new trainable model meta-parameters $\theta_{\alpha A}$ (referred to below as "model parameters", and which must be distinguished from the interaction parameters μ) that are constant in time,

unlike the time-varying interaction parameters μ. For basis functions $f_{\alpha A}(\mu)$ we use bases that arose in elementary solvable examples (two- and four-state binding models mentioned at the end of this subsection and described in [13]):

$$f_A(\mu) \in \bigcup_{\alpha,\beta}\{1, e^{\mu_\alpha}, e^{-\mu_\alpha}, \mu_\alpha^k, \frac{1}{(\mu_\alpha+1)}, e^{(\mu_\alpha-c)^2/2}, \mu_\alpha\mu_\beta, e^{2(\mu_\alpha-\mu_\beta)^2}, e^{2(\mu_\alpha+\mu_\beta)^2}\}, \tag{13}$$

for $k \in \{1,\ldots,5\}$ and $c \in \{-3,\ldots,+3\}$. Machine learning model selection algorithms (such as the "lasso" algorithm of section 3.3 below) can be used to encourage sparsity in the use of bases i.e. in the matrix θ, which in turn favors good generalization performance. Many other forms for trainable ODE dynamics could be used in this "system identification" subproblem, such as (nonlinear in θ) neural networks.

Like the Markovian equation (8), the deterministic equation (12) is differential in time so that the evolution of the coarse scale dynamical interaction parameters μ depends only on their state at the current time and not directly on their state at earlier times. However as remarked in [14], many consistent non-Markovian stochastic processes can be obtain from Markovian ones by integrating out extra degrees of freedom not otherwise needed. Similar phenomena obtain for differential equations. In GCCD it may be possible to do this by adding extra "hidden" interaction parameters and/or extra random variables to the GCCD graph.

As a postprocessing step, the BMLA-learned trajectories of $\mu_\alpha(t)$ interaction parameters could be smoothed in time t by convolution with a Gaussian in t and then differentiated with respect to time to get $d\mu/dt$; what we actually do is the mathematically equivalent operation of convolving with the analytic derivative of a Gaussian.

The solvable examples used to derive the basis functions in equation (13) were: (1) a two-state binding site model in which ligand binds to and unbinds from a site, and (2) a four-state, two-site cooperative binding model, both obeying detailed balance. The K-L divergence minimization algorithm derived in the Appendix solved both of these problems correctly to high accuracy. Unfortunately, with these basis functions, the GCCD K-L divergence minimization algorithm exhibited numerical instability and failure to converge on the realistic CaMKII problem outlined below. It is possible that this problem could be solved by variations such as more extensive stacking (defined in the Appendix), which would allow the use of more training data, or a different form for the ODEs such as different basis functions and/or ODEs nonlinear in θ. In particular the ODE right hand sides could take the mathematical form of trainable nonlinear neural networks. Multilayer neural networks with a final linear layer would generalize equation (12) to include trainable basis functions. In section 3 below we report on the results of a different strategy, which is to optimize approximation in the $O = R$ or μ space (as in equation (1) or (7)) rather than the $O = S$ space (as in equation (6)).

2.4 Previous work

If we multiply the appropriate Master Equation (equation (9) above) by monomials in key observables such as numbers of molecules of selected species in a chemical reaction network, and then sum over all states, we find ordinary differential equations for the time-evolution of various moments of the distribution over states. Unfortunately these equations do not *close*: the time derivative of lower-degree moments depends on the present value of higher-degree moments recursively, generating a countable infinity of coupled differential equations.

The goal of "moment closure" methods [15-27] is to obtain explicit though approximate dynamics for some finite subset of the first-order moments $C_i = \langle s_i \rangle_{p(t)}$, the second-order moments $C_{ij} = \langle s_i s_j \rangle_{p(t)}$, and higher moments $C^k_{i_1\ldots i_k} = \langle s_{i_1} \ldots s_{i_k} \rangle_{p(t)}$ of a collection of random variables s_i under some dynamics of their joint probability distribution $p(\vec{s}, t)$. Many approximate moment closure schemes have been developed starting from $k = 1$ mean field theory (MFT) (systematicaly replacing

$\langle s_i s_j \rangle_{p(t)}$ with a function $\langle s_i \rangle_{p(t)} \langle s_j \rangle_{p(t)}$ of the first-order moments as would be correct for independent distributions) from which one recovers ordinary deterministic chemical kinetics, and escalating to second-order ($k = 2$) moment closures that consistently keep track only of means and variances (as would be correct for a Gaussian joint distribution) in chemical reaction networks [15,16], or in population biology reaction-diffusion models [17] explicitly, or by means of the Fokker-Planck equation [18] or stochastic differential equations such as the Chemical Langevin Equation [19] which may sometimes be further reduced [20].

A slightly higher order scheme is the Kirkwood Superposition Approximation (KSA) that retains $\langle s_i s_j \rangle_{p(t)}$ ($k = 2$) but approximates triple correlations ($k = 3$) by a function of second-order correlations, is derivable from [21,22] a constrained maximum-entropy sense of approximation $\approx$, and has been used in multicellular stochastic modeling [23]. Fully dynamic higher order cutoffs to the moment hierarchy for $k > 2$ include setting higher cumulants to zero [24], dropping higher order terms from the Kramers-Moyal expansion [25] using moment closure functions that would be correct for log-normal rather than Gaussian distributions [26], and "equation-free" moment closure [27] by sparingly invoking fine-scale simulations.

Each of these moment closure methods has the character, when compared to a sampling algorithm, of first exponentially expanding the space in which dynamics are formulated using the Master Equation, and then trying to cut the exponentially large space back down to size. Typical results start from a small reaction network with $n < 10$ molecular species (and thus chemical/biological degrees of freedom if there is a single well-stirred compartment), and produce a more efficient algorithm for determining low-order moment trajectories for a possibly reduced model of between about $n/2$ and n molecular species, yielding model size reductions on the order of a factor of 1 to 2. The initial combinatorial explosion is not fully mitigated. This is an unpromising route if the goal is to find a large model reduction beyond what one already had at the fine scale (though not an impossible one, due to the need to run sampling simulations many times). We are proposing a different strategy for moment closure which more naturally results in model reduction with fewer chemical/biological degrees of freedom. From the moment closure point of view, what we are proposing is an arbitrary-order method particularly suited to approximating the Chemical Master Equation and possibly related master equations, by a time-dependent variant of a Boltzmann distribution.

Additional model reductions for stochastic chemical kinetics, other than moment closure methods, include the classic strategy of using separation of time scales to eliminate fast degrees of freedom as carried out in eg. the Quasi-Steady State Approximation [28], in adiabatic course-graining [29], and with power law scaling of time with respect to an overall problem size parameter, differentially for different subsets of molecular species [30]. Another strategy for molecular species with small expected population sizes is the Finite State Projection method which proposes an adaptive truncation of the state space followed by analytic solution or bounding of the (exponentially big, were it not truncated) master equation [31]. Other reaction network model reduction methods are restricted to deterministic models [32,33] including a reduction from 42 to 29 molecular species [34]. The most comparable method in terms of problem size may be [35] which like GCCD applies to rule-based reaction networks. We will quantitatively compare degrees of model reduction of these methods to GCCD in section 3.4.

As in the case of moment closure methods, all of these methods have advantages and interesting ideas but none that we know of are as yet in the same class of radical model size reduction as GCCD, in terms of fraction of molecular species retained after reduction, in a stochastic biochemical network model.

3 Computational Experiments

Molecular complexes in general, and signal transduction complexes in particular often have state space explosions that pose problems for simulation and modeling. One such macromolecular complex is Ca^{2+}/calmodulin-dependent protein kinase II (CaMKII) in synapses, which is activated when it binds to multiple calmodulin (CaM) molecules that in turn bind to multiple calcium ions. These processes trigger a cascade of reactions in a signal transduction pathway following Ca^{2+} influx in response to activation of ligand-gated n-methyl D-aspartate receptors (NMDARs), supporting memory formation and learning.

We applied GCCD to this system as modeled by [1] and as simulated by the Plenum implementation of Dynamical Grammars [2,4] and also in a much larger spatial simulation using MCell [3].

3.1 CaMKII System

The synaptic signal transduction pathway that starts with calcium ion (Ca^{2+}) influx through voltage-dependent calcium channels (VDCCs) and N-methyl D-aspartate receptors (NMDARs) leads ultimately to functional consequences including long-term potentiation (LTP), long-term depression (LTD), and spike-timing dependent plasticity (STDP) underlying learning and memory formation in the hippocampus. The pathway as studied and modeled in [1] is structurally a bit involved. It begins with an externally imposed influx of calcium ion Ca^{2+}. Calcium ions bind to the calmodulin protein (CaM), which has four calcium-binding sites, two at the N-terminal end and two at the C-terminal end. CaM in any state of calcium-loading can then bind to unphosphorylated CaMKII monomer (which has one phosphorylation site relevant for activation). However, the binding and unbinding rates for calcium ion to CaM depends on the state of the other binding sites of the CaM protein, and also on whether or not that CaM is bound to CaMKII. Likewise the binding and unbinding rates for CaM to unphosphorylated CaMKII monomer depend on the state of CaM. Two CaMKII monomers, each loaded with CaM in any calcium-binding state, but at most one of which is phosphorylated, may then dimerize (again with state-dependent rates). Dimers may phosphorylate one of their constituent subunits and promptly dissociate; autophosphorylated monomer CaMKII is taken to be the pathway output. Our goal is to produce a reduced stochastic model simplifying the structure of this fine-scale model as formulated in the MCell and Plenum models (model files in Supplementary Information) that aim to implement stochastic versions of the reaction network of [1].

Many subsequent stages of the biological pathway are thereby omitted, notably the formation of a CaMKII holoenzyme comprising a ring of six dimers in dodecameric complex. This holoenzyme structure implies an even greater combinatorial explosion of states that poses a future modeling challenge, that may best be met by aggressive model reduction methods such as the GCCD method proposed here. Further downstream components of the pathway beyond CaM and the CaMKII holoenzyme are outlined in [8]. However we leave such explorations, which could aim to extract novel biological consequences from combinatorially large stochastic reaction network models of the NMDA receptor pathway by applying the GCCD model reduction technique, to future research.

3.2 Boltzmann machine preprocessing step

Figure 2 shows part of the assumed Boltzmann distribution model interaction graph, or Markov Random Field, of binary-valued random variables (circles) and monomial interaction potentials of degree 1, 2, and 3 (hexagons). The first row of variables represent the binding of calcium to calmodulin (CaM). With subscripts these ± 1-valued random variables are labelled "$\mathrm{CaM}_{\mathrm{c/n},a,i}$". They are indexed by the C-terminus vs the N-terminus of the calmodulin protein (c or n), the numerical index

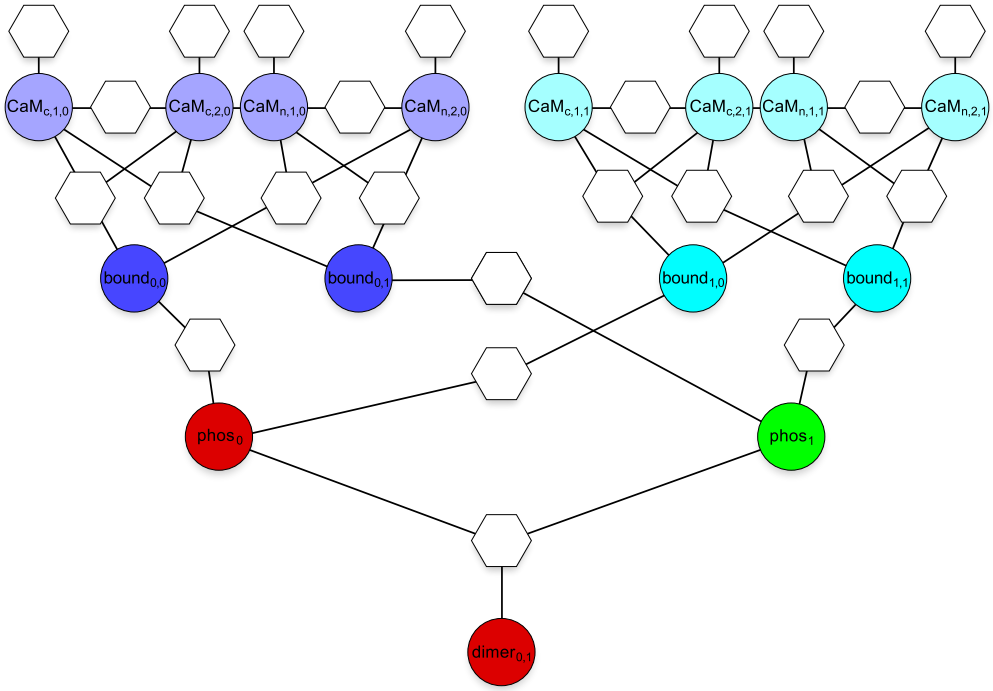

Figure 2: Markov Random Field or Boltzmann distribution model of interaction degree 3 assumed for the CaMKII model. Diagram shows the graph-structure of the interaction terms $\mu_\alpha V_{\alpha\gamma}$ (hexagons) of degrees 1, 2, and 3 assumed between all the binary-valued random variables (colored labelled circles) described in the text. The left and right sides of the graph correspond to two CaMKII subunits that may dimerize (bottom variable), each of which may be phosphorylated (third row of variables) and/or bind a calmodulin (second row of variables) which in turn may bind calcium at its N-terminus or C-terminus and at site 1 or 2 at each end (first row). Omitted from the figure are extra factors that ensure each binding site is occupied by either zero or one molecules and not more. This figure comprises a template for the labelled graph of all random variables and their interactions in this application of "graph-constrained correlation dynamics".

of the binding site on that end ($a = 1$ or 2), and the numerical index i ($i = 0$ or 1 illustrated; $i \in \{0, 1, 2\}$ used in Plenum simulations below) of a calmodulin molecule which may bind to a CaMKII subunit (indexed by $j = 0$ or 1 illustrated; $j \in \{0, \ldots 8\}$ used in Plenum simulations below). The second row of variables "bound$_{i,j}$" records the state of binding of CaM to CaMKII subunits. The third row of variables "phos$_j$", also written (in the notation of the MCell model) as "Kkp$_j$", records the binary phosphorylation state of each CaMKII subunit, and the fourth row of variables "dimer$_{jj'}$" records whether or not two such subunits dimerize. Not shown are additional cardinality-fixing or winner-take-all interactions enforcing the constraints that (if it occurs) binding is an exclusive relationship between CaM molecules and CaMKII subunits, and likewise for dimerization between two CaMKII subunits.

Weight-sharing is an important strategy in machine learning, widely used to reduce the number of trainable parameters and thereby increase generalization power for a given amount of data. Our weight-sharing scheme shares interaction parameters μ_α within categories of monomial interactions that seem likely, according to testable human intuition, to have similar interaction strengths if trained separately on far more data. We now introduce some notation for the weight-sharing. Let $\mathcal{I} = \{0, \ldots \#\text{CaM} - 1\}$ and $\mathcal{J} = \{0, \ldots \#\text{CaMKIIsubunits} - 1\}$. We have the following categories of ± 1-

valued random variables s_I:

$$\begin{aligned} \mathrm{CaM}_{\mathrm{c/n},a,i} \quad & (\forall \mathrm{c/n} \in \{\mathrm{c},\mathrm{n}\})(\forall a \in \{1,2\})(\forall i \in \mathcal{I}) \\ \mathrm{bound}_{i,j} \quad & (\forall i \in \mathcal{I})(\forall j \in \mathcal{J}) \\ \mathrm{Kkp}_j \quad & (\forall j \in \mathcal{J}) \\ \mathrm{dimer}_{jj'} \quad & (\forall j < j' | j, j' \in \mathcal{J}) \end{aligned}$$

We have used the following eight categories of monomial interactions s_I, $s_I s_J$, or $s_I s_J s_K$ (all taking values in $\{\pm 1\}$):

$$\begin{aligned} \mathrm{CaM}_{\mathrm{c},a,i} \quad & (\forall a \in \{1,2\})(\forall i \in \mathcal{I}) \\ \mathrm{CaM}_{\mathrm{n},a,i} \quad & (\forall a \in \{1,2\})(\forall i \in \mathcal{I}) \\ \mathrm{CaM}_{\mathrm{c},1,i}\mathrm{CaM}_{\mathrm{c},2,i} \quad & (\forall i \in \mathcal{I}) \\ \mathrm{CaM}_{\mathrm{n},1,i}\mathrm{CaM}_{\mathrm{n},2,i} \quad & (\forall i \in \mathcal{I}) \\ \mathrm{bound}_{i,j}\mathrm{CaM}_{\mathrm{c},1,i}\mathrm{CaM}_{\mathrm{c},2,i} \quad & (\forall i \in \mathcal{I})(\forall j \in \mathcal{J}) \\ \mathrm{bound}_{i,j}\mathrm{CaM}_{\mathrm{n},1,i}\mathrm{CaM}_{\mathrm{n},2,i} \quad & (\forall i \in \mathcal{I})(\forall j \in \mathcal{J}) \\ \mathrm{Kkp}_j\mathrm{bound}_{i,j} \quad & (\forall i \in \mathcal{I})(\forall j \in \mathcal{J}) \\ \mathrm{Kkp}_j\mathrm{Kkp}_{j'}\mathrm{dimer}_{jj'} \quad & (\forall j < j' | j, j' \in \mathcal{J}) \end{aligned}$$

We number these weight-sharing categories $\alpha \in \{1, \ldots 8\}$. Different categories α have different numbers n_α of monomial interactions depending on which bound indices a, i, j, j' they run over. We take the potential function V_α for each category to be the category-average of the constituent monomials $V_{\alpha,\gamma} = s_I$, $s_I s_J$, or $s_I s_J s_K$ given above, so that $V_\alpha = (1/n_\alpha) \sum_{\gamma=1}^{n_\alpha} V_{\alpha,\gamma}$.

The resulting Boltzmann distribution can be sampled by standard Monte Carlo methods such as Metropolis-Hastings or Gibbs sampling. We use the "Dependency Diagrams" software [13] (availability described in Supplementary Information) to do this. A crucial wrinkle on our use of such sampling algorithms is that we have two different biological conditions under which they get used: external calcium influx can be "on" or "off". We train four different sets of coarse-scale dynamical model parameters θ as described in section 3.3 below, for four phases (early and late phases for calcium influx on and for calcium influx off), and for pulsatile or periodic calcium influx we cycle through the four trained (or partly trained) models, while preserving the interaction parameters μ through each instantaneous phase-switching event. Once trained, these four models predict dynamical responses to any other temporal pattern of calcium influx switching including the use of periodic switching with frequencies not in the training set.

A logical alternative to this procedure would be to use just two phases for calcium on and off, and to add the binary calcium-influx variable into the GCCD graph of figure 2 with suitable connections to allow the switch variable to join in modulating some or all of the potentials V_α. However, we were not able to get this somewhat cleaner approach to work numerically. Just switching between two phases (Ca^{2+} influx on vs. off) rather than four phases without modifying the GCCD graph also produced less robust behavior.

For the MCell simulations below the waveform for calcium influx was a pulse train or rectangular wave, with the "on" duration being 16 msec. For the Plenum simulations we used the "alpha model" theoretical calcium influx profile [37].

In figures 3 and 4, the inferred trajectories of μ time-varying interaction parameters are shown for the non-spatial Plenum [2] model (figure 3) and the spatial MCell [3] model (figure 4). The corresponding moments $\langle V_\alpha \rangle \in [-1, 1]$ are shown in figures 6 and 7 of the Supplementary Information, where their relation to concentrations is discussed. From figures 3 and 4 it is evident that the inferred μ_α interaction parameter trajectories are remarkably continuous in time, empirically justifying the assumption of ODE dynamics in equation (12).

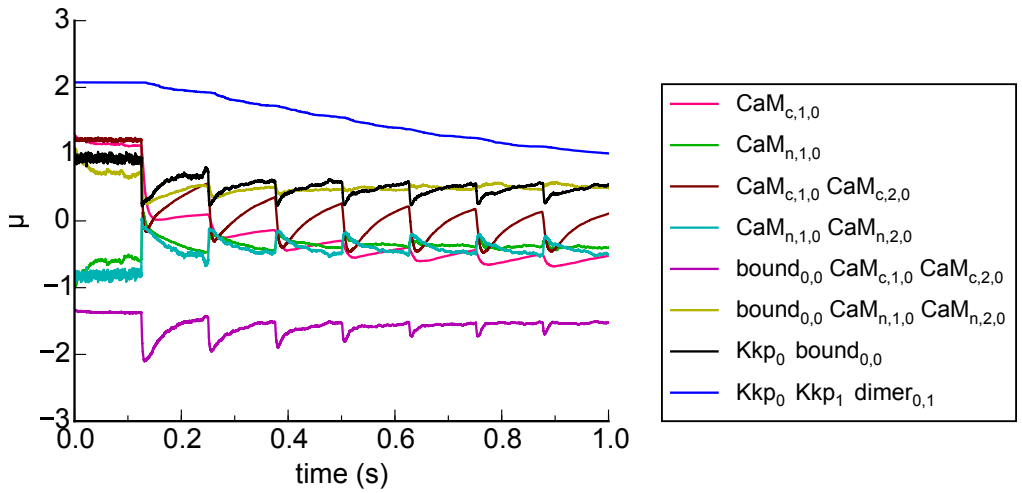

Figure 3: Time series of eight BMLA-inferred $\mu_\alpha(t)$ interaction parameter statistics, from the simulation data generated by the Plenum model in section 7.4 of the Supplementary Information text. Seven successive periods of an 8 Hz pulse train (alpha model waveform [37]) of calcium influx are plotted. In each pulse period the μ dynamics switches through four phases: fast calcium-on (1.6 msec), slow calcium-on (14.4 msec), fast calcium-off (8 msec), slow calcium-off (the rest of the period), with different sets of trained weight θ for each phase, while maintaining continuity of μ values through the switching events. The legend shows the color of plot line associated to each of eight potentials V_α indicated by the given triples of state variables. Time course traces are labelled by random-variable monomials following the indexing of figure 2, although the actual graph is larger due to increased range of indices a, i, j, j' as defined in the text. Weight-sharing allows many different potential-function monomials $V_{\alpha,\gamma}$ differing only by the value of their indices γ (mapped to various combinations of a, i, j, j' as defined in the text), to share weights μ_α. The corresponding moments $\langle V_\alpha \rangle(t)$ are plotted in figure 6, Supplementary Information text.

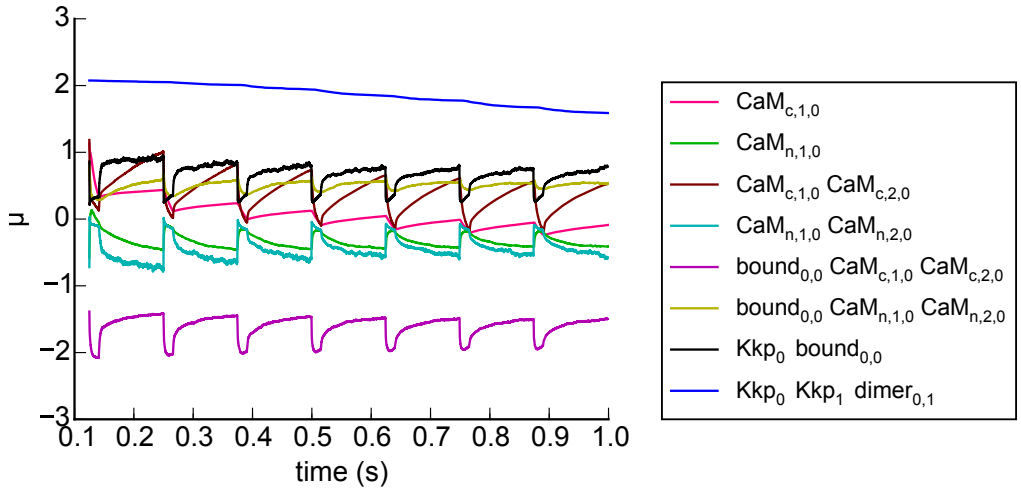

Figure 4: Time series of eight BMLA-inferred $\mu_\alpha(t)$ interaction parameter statistics, from the simulation data generated by the MCell model in section 7.4 of the Supplementary Information text, with 8 Hz rectangular wave pulse train of Ca^{2+} influx. Other plotting details are as in figure 3. Results differ significantly from those of figure 3 because the MCell model is extended over space, includes diffusion of reactants, and models a larger volume with more molecules of each species. The corresponding moments $\langle V_\alpha \rangle(t)$ are plotted in figure 7, Supplementary Information text.

3.3 GCCD

The resulting time series (such as figures 3 and 4) are convolved with the analytic temporal derivative of a Gaussian filter in order to smoothly estimate the rates of change $d\mu_\alpha/dt$ of the interaction parameters $\mu_\alpha(t)$. Such temporal smoothing is useful since taking derivatives of a noisy time series tends to enhance the noise, and in the absence of smoothing that occurs for our time series [13]. The resulting smoothed time derivatives are fit to equation (12) using either (1) online minimization of the K-L divergence as outlined in Appendix I for which $O = S$, or (2) lasso-regularized linear regression [36] for which $O = R$, and which performs model selection on the bases of equation (13) as discussed there. Here we report numerical results for the second method, which also defines a meaning for the $\approx$ symbol in section 2.1 as the lasso-regularized (L_1-regularized) sum of squared differences for the time derivatives of the $\mu_\alpha(t)$ statistical interaction parameters (summed over α, discretized t, and over any set of initial conditions and/or other input conditions c such as calcium influx frequency). Thus, we optimize the model parameters $\theta_{\alpha A}$ by minimizing the score

$$S([\theta_{\alpha A}]) = \sum_{\alpha, t_{discr}, c} \left|\left| \frac{d\mu_\alpha(t)}{dt}\Big|_{fit}[\theta_{\alpha A}] - \frac{d\mu_\alpha(t)}{dt}\Big|_{BMLA} \right|\right|^2 + \lambda \sum_{\alpha A} |\theta_{\alpha A}|. \tag{14}$$

(defining a sense of approximation $\approx$ as needed in section 2.1) which by equation (12) is equivalent to

$$S([\theta_{\alpha A}]) = \sum_{\alpha, t_{discr}, c} \left|\left| \frac{d\mu_\alpha(t)}{dt}\Big|_{BMLA} - \sum_A \theta_{\alpha A} f_A(\mu) \right|\right|^2 + \lambda \sum_{\alpha A} |\theta_{\alpha A}|, \tag{15}$$

a form that is explicitly lasso-regularized least squares optimization of the trainable interaction parameters θ. The single scalar hyperparameter λ was set using leave-one-out cross validation, with each calcium influx spike held out in turn. 150 out of 253 model bases parameters were always rejected by the lasso algorithm, and the remaining 103 bases were used sparsely depending on the μ_α

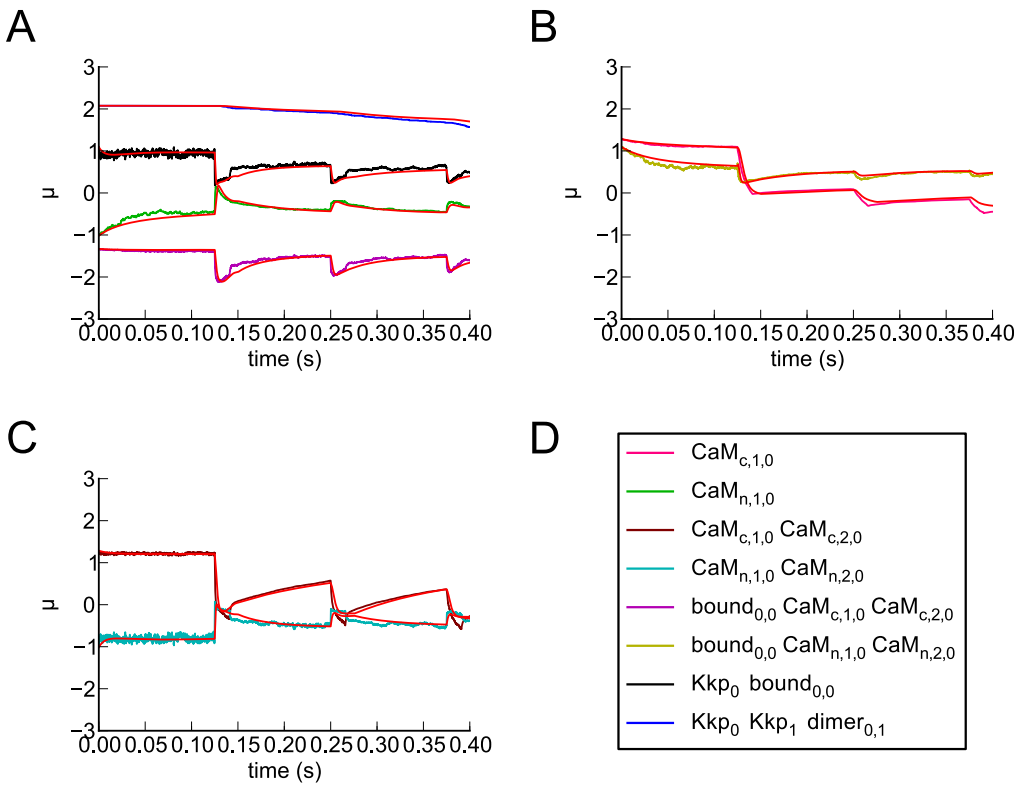

Figure 5: Dynamics of μ for coarse-scale GCCD model tested out-of-sample (red lines), plotted along with BMLA inferred values of μ (other colors) from Plenum stochastic simulations of Ca^{2+}/CaM/CaMKII network, using the alpha model [37] for calcium influx. Panels A, B and C visually separate pairs of (model, data) curves that would otherwise overlap, for ease of visual comparison. A high degree of overlap between (model, data) curve pairs is evident. Quantitative comparison of each (model,data) pair is given in Table 1. Simulation is out-of-sample in that the calcium influx bursts occur with frequency 8 Hz, though GCCD was trained at frequencies 2, 4, and 10 Hz.

derivative being fit, which of the four phases was being fit, and which BMLA experiment data set was being used.

The resulting constrained time-evolution of ODE-constrained interaction parameters $\mu_\alpha(t)$ evaluated out-of-sample (i.e. using different data than was used for training the model parameters) was almost indistinguishable from the unconstrained values of optimal interaction parameters obtained by BMLA, as a function of time over several calcium influx cycles, as shown in figure 5.

Numerical errors in figure 5 are shown in Table 1, computed as root mean squared error (RMSE) and also as normalized root mean squared error, in which the normalization is done by finding the average of the absolute value of each target BMLA time course, and dividing both the target and corresponding prediction time course by this average absolute value before computing the RMSE as usual.

The results show good out-of-sample quantitative agreement for all 8 time series.

3.4 Discussion of Results

In quantitative terms, the degree of model reduction obtained by GCCD in the CaMKII example is large. Eight dynamical variables (the interaction parameters μ) suffice to predict key outputs such

Normalized RMS errors in figure 5

Interaction parameter	RMS Error	Normalized RMS Error
CaM(c,1,0)	rms1 = 0.062	cvrms1 = 0.132
CaM(n,1,0)	rms2 = 0.070	cvrms2 = 0.174
CaM(c,1,0) CaM(c,2,0)	rms3 = 0.096	cvrms3 = 0.170
CaM(n,1,0) CaM(n,2,0)	rms4 = 0.075	cvrms4 = 0.141
bound(0,0) CaM(c,1,0) CaM(c,2,0)	rms5 = 0.068	cvrms5 = 0.044
bound(0,0) CaM(n,1,0) CaM(n,2,0)	rms6 = 0.061	cvrms6 = 0.118
Kkp0 bound(0,0)	rms7 = 0.090	cvrms7 = 0.133
Kkp0 Kkp1 dimer(0,1)	rms8 = 0.044	cvrms8 = 0.023

as the global degree of CaMKII phosphorylation, each of which can be computed from expectations of random variables s in the Boltzmann distribution of equation (11). But the fine-scale set of dynamical variables is much larger. In the MCell simulations we may conservatively count it as the integer-valued populations of the following classes of molecular species: free Ca^{2+}, free CaM ($3 \times 3 + 1 = 10$ species), monomeric CaMKII subunit which can bind CaM in any of its states and can also be phosphorylated ($3 \times 3 \times 2 = 18$ species), and dimerize if at most one subunit is phosphorylated ($9 \times 9 + 9 \times 10/2 = 126$ species; phosphorylated dimers dissociate before they can doubly phosphorylate) for a total of 155 species each of which has a dynamical random variable, and therefore a total reduction from 155 to just 8 dynamical variables, which is very large.

In one sense this reduction in the number of dynamical variables strongly understates the situation, because in the actual MCell simulation (though not in the Plenum simulations), every individual chemical species has its own independent three-dimensional position which is also a dynamical random variable. Given that a typical molecular population excluding Ca^{2+} is 145 CaM + 385 CaMK = 530, and including each 24-unit Ca^{2+} pulse is still greater, and that the number of position degrees of freedom is three times greater (1590 or more), the reduction to just 8 dynamical variables μ_1 through μ_8 as listed in figures 3-5 is even more remarkable.

Comparable figures for the (deliberately non-spatial and well-mixed) Plenum simulations are again 155 molecular species reduced down to 8. An intermediate step is the formulation of the graph in figure 2 which has just eight interaction parameters (associated with the hexagonal interaction nodes) due to weight sharing, but it has many more molecular binding variables (circular nodes in figure 2). For the Plenum model we can count these variables as follows: Due to the smaller simulated volume than for the MCell model, the index i for CaM and the index j for CaMKII run from 0 to 2 and 0 to 8 respectively. If we replicate the circular nodes in the graph of figure 2 accordingly, there are 4×3, 3×9, 9, and $9 \times 10/2$ variables respectively in rows 1-4 of the full version of the graph illustrated in figure 2, for a total of 93 fine-scale binary-valued variables in a highly structured pattern.

As stated in section 2.4, most current results for stochastic model reduction start from a small reaction network with handful of chemical species, and produce a more efficient (sometimes much more efficient) modeling algorithm for a possibly reduced model with model size reductions on the order of a factor of 1 to 2. The authors of [35], a stochastic and rule-based model reduction method as is GCCD, achieve a larger model reduction in an EGF signal transduction model from 2768 to 609 molecular species, for a 4.5$\times$ reduction factor or a 0.81-power relation of reduced to full model ($609 \simeq 2768^{0.809}$). We have demonstrated for GCCD at least 155 to 8 for a 19.5$\times$ reduction factor, or a .41-power relation ($8 \simeq 155^{0.412}$). This factor of almost 20 breaks decisively out of the pattern of model size reductions on the order of a factor of just 1 to 2 in number of chemical or biological degrees of freedom. Exploring tradeoffs between accuracy of approximation and amount of model size reduction (whether measured in direct ratios, or powers, of number of degrees of freedom before

and after reduction) for increasingly large models would therefore seem to be an attractive topic for future work.

To what features of GCCD or the present problem do we owe these very substantial reductions in model size? Future work could disentangle the following seemingly relevant factors: (1) GCCD can be applied to rule-based models, which are highly structured in that (as shown in the Plenum model code in the Supplementary Information text) a small number of rules can combinatorially code for a much larger molecular species-level reaction network. Thus an underlying simplicity is available. (2) The use of *weight-sharing* may make it possible to exploit such underlying simplicity in the problem domain. (3) The machine learning components of GCCD (BMLA and the new procedures for determining GCCD model parameters θ) generalize directly from specific simulation data rather than algebraically and in full generality from the mathematical form of the fine-scale model. So currently available computer power is used effectively. (4) The Boltzmann distribution is a natural form for prolongation from coarse to fine models since by constrained entropy maximization it doesn't add any more information than is given by constraints on whatever moments were chosen for use in the GCCD graph structure. (5) The graph structure of GCCD is a good mechanism for importing biological knowledge and expertise into the model reduction process.

4 Conclusion

We propose a nonlinear model reduction method particularly suited to approximating the Chemical Master Equation for stochastic chemical reaction networks (including highly structured ones resulting from "parameterized" or "rule-based" reactions), by a time-dependent variant of a Boltzmann distribution. The resulting Graph-Constrained Correlation Dynamics (GCCD) method can be an accurate nonlinear model reduction method for stochastic molecular reaction networks involving a combinatorial explosion of states, such as the CaMKII signal transduction complex that is essential to synaptic function, particularly in neuronal learning processes such as long-term potentiation. The GCCD method could be further developed in many directions, including application to model-reduction for the Master Equation semantics of more challenging molecular complexes such as the CaMKII dodecamer, use of the resulting reduced models to extract novel biologically relevant predictions, and generalizing the method to yet more general reaction-like biological modeling formalisms capable of expressing multiscale models in developmental biology [4].

Acknowledgements:

We wish to acknowledge many useful discussions with Mary Kennedy, Shirley Pepke, Tamara Kinzer-Ursem, and David H. Sharp. This work was supported by NIH grant RO1 GM086883; also supported in part by the United States Air Force under Contract No. FA8750-14-C-0011 under the DARPA PPAML program, by NIH grant R01 HD073179 to Ken Cho and EM, by the Leverhulme Trust, and by the hospitality of the Sainsbury Laboratory Cambridge University.

5 References

[1] Pepke S, Kinzer-Ursem T, Mihalas S, and Kennedy MB. A Dynamic Model of Interactions of Ca^{2+}, Calmodulin, and Catalytic Subunits of Ca^{2+}/Calmodulin-Dependent Protein Kinase II. *PLoS Comp Bio*, Feb. 12, 2010.

[2] Mjolsness E and Yosiphon G. Stochastic Process Semantics for Dynamical Grammars. *Annals of Mathematics and Artificial Intelligence*, 47(3-4) August 2006.

[3] Kerr RA, Bartol TM, Kaminsky B, Dittrich M, Chang JCJ, Baden SB, Sejnowski TJ, and Stiles JR. Fast Monte Carlo Simulation Methods for Biological Reaction-Diffusion Systems in Solution and on Surfaces. *SIAM Journal on Scientific Computing* 30(6):3126, 2008.

[4] Mjolsness E, Time-ordered product expansions for computational stochastic systems biology. *Physical Biology* 10 035009, 2013.

[5] Hlavacek WS, Faeder JR, Blinov ML, Posner RG, Hucka M, Fontana W. Rules for modeling signal-transduction systems. *Science's STKE* Jul 18;2006(344):re6, 2006.

[6] Danos V, Laneve C. Formal molecular biology. *Theoretical Computer Science* 325(1), 69 - 110 (2004).

[7] Gillespie DT, "Exact stochastic simulation of coupled chemical reactions". J. Phys. Chem. 81(25), pp. 2340-2361, 1977.

[8] Kennedy MB, Beale HC, Carlisle HJ, and Washburn LR, "Integration of biochemical signalling in spine". Nature Reviews Neuroscience, 6, pp. 423-434, 2005.

[9] Yavneh I, "Why Multigrid Methods Are So Efficient". Computing in Science and Engineering, 8 , pp. 12-22, 2006.

[10] Brandt A, "Multi-level adaptive solutions to boundary-value problems". Mathematics of Computation 31(138), pp. 333-390, 1977. For spatial problems "prolongation" is usually called "interpolation"'.

[11] Smyth P, "Belief networks, hidden Markov models, and Markov random fields: A unifying view ". Pattern Recognition Letters 18 pp. 1261-1268 1997.

[12] Ackley DH, Hinton GE, and Sejnowski TJ. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9(1):147-169, 1985.

[13] Johnson T. Dependency Diagrams and Graph-Constrained Correlation Dynamics: New Systems for Probabilistic Graphical Modeling. PhD thesis, UC Irvine Computer Science Department, May 2012. Available at URL http://www.ics.uci.edu/ johnsong/thesis/ .

[14] Gillespie DT, "The multivariate Langevin and Fokker-Planck equations". Am. J. Physics 64(10), pp. 1246-1257, 1996.

[15] Lee C H, Kim K-H, Kim P, "A moment closure method for stochastic reaction networks". J. Chem. Phys 130, p. 1341-7, 2009.

[16] Schnoerr D, Sanguinetti G, Grima R, "Validity conditions for moment closure approximations in stochastic chemical kinetics". J. Chem. Phys 141, p. 084193, 2014.

[17] Gandhi A, Levin S, and Orszag S, "Moment expansions in spatial ecological models and moment closure through Gaussian approximations". Bull. Math. Bio. 62, pp. 595-632, 2000.

[18] Risken H, *The Fokker-Planck Equation*. Second Edition, Springer 1989.

[19] Gillespie DT. The chemical Langevin equation. Journal of Chemical Physics. 2000; 113:297

[20] Sotiropoulos V, Contou-Carrere M-N, Daoutidis P, Kaznessis YN, "Model reduction of multiscale chemical Langevin equations: A numerical case study". IEEE/ACM Trans. computational biology and bioinformatics 6:3, pp. 470-82, 2009.

[21] Singer A, Maximum entropy formulation of the Kirkwood superposition approximation. *J. Chem. Physics*, v 121 no 8, 22 Aug 2004.

[22] Raghib M, Hill NA, Dieckmann U, "A multiscale maximum entropy moment closure for locally regulated space-time point process models of population dynamics". J. Math. Biol. 62 p. 605-653, 2011.

[23] Markham DC, Baker RE and Maini PK, Modelling Collective Cell Behavior. *Discrete and Continuous Dynamical Systems*, Volume 34, Number 12, December 2014 pp. 5123-5133.

[24] Hespanha J, "Moment closure for biochemical networks". 3rd International Symposium on Communications, Control and Signal Processing (ISCCSP), 2008.

[25] van Kampen, N. Stochastic processes in physics and chemistry. North Holland: 1992.

[26] Singh A and Hespanha J P, "Lognormal Moment Closures for Biochemical Reactions". Proc. 45th IEEE Conference on Decision and Control, San Diego, December 13-15, pp 2363-2068, 2006.

[27] Alexander FJ, Johnson G, Eyink GL, and Kevrekidis IG, Equation-Free Implementation of Statistical Moment Closures, *Phys. Rev. E.* 77,26701, 2008.

[28] Rao C and Arkin AP, "Stochastic chemical kinetics and the quasi-steady-state assumption: Application to the Gillespie algorithm". J. Chem. Phys. 118:11, pp. 4999-5010, 2003.

[29] Sinitsyn N A, Hengartner N, and Nemenman I, "Adiabatic coarse-graining and simulations of stochastic biochemical networks". Proc. Nat. Acad. Sci. USA 106: 26 pp. 10546-10551 2009.

[30] Kang HW and Kurtz T G, "Separation of time-scales and model reduction for stochastic reaction networks". Annals of Applied Probability 23:2, pp 529-583, 2013.

[31] Munsky B and Khammash M, "The finite state projection algorithm for the solution of the chemical master equation". J. Chem. Phys. 124:4 p. 044104, 2006.

[32] Lebiedz D, Skanda D, Fein M, "Automatic Complexity Analysis and Model Reduction of Nonlinear Biochemical Systems". *Computational Methods in Systems Biology*, Springer Lecture Notes in Computer Science 5307, pp 123-140, 2008.

[33] Hangos K M, Gabor A, Szerderkenyi G, "Model reduction in bio-chemical reaction networks with Michaelis-Menten kinetics". Proc. 2013 European Control Conference (ECC), Zurich Switzerland, 2013.

[34] Rao S, van der Schaft A, van Eunen K, Bakker B M, and Jayawardhana B, "A model reduction method for biochemical reaction networks". BMC Systems Biology 8:52, 2014.

[35] Feret J, Henzinger T, Koeppl H, Petrov T, "Lumpability abstractions of rule-based systems". Theoretical Computer Science 431, pp 137-164, 2012.

[36] Friedman J, Hastie T, and Tibshirani R. Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software*, 33(1):1-22, 2010. ISSN 15487660.

[37] Destexhe A, Mainen ZF, and Sejnowski TJ. "Synthesis of models for excitable membranes, synaptic transmission and neuromodulation using a common kinetic formalism". Journal of Computational Neuroscience, 1(3):195Ð230, 1994.

6 Appendices

6.1 Online learning derivations

We begin with the definition of KL divergence,

$$\mathcal{D}_{\mathcal{KL}}(\tilde{p}\|p) = -\int \tilde{p}\log(p/\tilde{p}) \tag{16}$$

(or equivalently, $\int \tilde{p}\log(\tilde{p}/p)$). We could equally well begin with the divergence in the other direction, as $-\int p\log(\tilde{p}/p)$; the analogous derivation in that case is similar to what follows (and is performed in Appendix A of [13]) but in our experience the resulting training algorithm produced slightly less reliable results.

Our approach will be to compute the derivative $\partial\mathcal{D}_{\mathcal{KL}}/\partial\mu_\alpha$, then take the time-derivative of this term, and minimize that. Minimization of $\partial\mathcal{D}_{\mathcal{KL}}/\partial\mu_\alpha$ corresponds to matching the distribution $\tilde{p}$ to p at an initial time point, and we will need to take for granted the ability to do this well once, as an initialization. Then, if we have done a good job of that, keeping the change in this term 0 – setting the derivative of this term equal to zero and solving for the parameters of a GCCD model – will track the optimal solution as the two distributions change in time.

Though we have so far defined X to be a discrete space, we use the integral notation throughout this derivation, as integration specializes to summation on a discrete domain, but the converse is not true.

The first few steps here are just pulling apart terms, starting from the definitions of the Markov Random Field (MRF), as follows:

$$\mathcal{D}_{\mathcal{KL}}(\mu(t)) = -\int \tilde{p}(x;\mu(t))\log\left(\frac{p(x;t)}{\tilde{p}(x;\mu(t))}\right)\mathrm{d}x$$

so

$$\mathcal{D}_{\mathcal{KL}}(\mu(t)) = -\int \tilde{p}(x;\mu(t))\left(\sum_{\beta=1}^{\text{cliques}}\mu_\beta(t)V_\beta(x) + \log\left(Z(\mu(t))\right) + \log\left(p(x;t)\right)\right)\mathrm{d}x.$$

Now evaluate the derivative $\partial\left[\mathcal{D}_{\mathcal{KL}}(\mu(t))\right]/\partial\mu_\alpha$. Beginning with the product rule, we have

$$\begin{aligned}\frac{\partial\mathcal{D}_{\mathcal{KL}}(\mu(t))}{\partial\mu_\alpha} = &-\int\left(\frac{\partial}{\partial\mu_\alpha}\tilde{p}(x;\mu(t))\right) &(17)\\ &\quad\times\left(\sum_{\beta=1}^{\text{cliques}}\mu_\beta(t)V_\beta(x) + \log\left(Z(\mu(t))\right) + \log\left(p(x;t)\right)\right)\mathrm{d}x\\ &-\int\tilde{p}(x;\mu(t)) &(18)\\ &\quad\times\frac{\partial}{\partial\mu_\alpha}\left(\sum_{\beta=1}^{\text{cliques}}\mu_\beta(t)V_\beta(x) + \log\left(Z(\mu(t))\right) + \log\left(p(x;t)\right)\right)\mathrm{d}x. &(19)\end{aligned}$$

Breaking this expression down, we begin by evaluating $\frac{\partial}{\partial\mu_\alpha}\tilde{p}(x;\mu(t))$.

$$\frac{\partial}{\partial\mu_\alpha}\tilde{p}(x;\mu(t)) = \frac{\partial}{\partial\mu_\alpha}\frac{\prod_\beta e^{-\mu_\beta V_\beta(x)}}{Z(\mu(t))} = \tilde{p}(x)\left(\langle V_\alpha\rangle - V_\alpha(x)\right). \tag{20}$$

Unless otherwise noted, all expectations are with respect to the distribution $\tilde{p}$.

Plugging this result back into Equation (19), and using $\partial \log\left(p(x;t)\right)/\partial\mu_\alpha = 0$, along with the evaluation of $\partial \log\left(Z(\mu(t))\right)/\partial\mu_\alpha$ as $-\langle V_\alpha\rangle$, we have

$$\begin{aligned}\frac{\partial \mathcal{D}_{\mathcal{KL}}(\mu(t))}{\partial \mu_\alpha} = &-\int \tilde{p}(x;\mu(t))\left(\langle V_\alpha\rangle - V_\alpha(x)\right) \\ &\times\left(\sum_{\beta=1}^{\text{cliques}} \mu_\beta(t)V_\beta(x) + \log\left(Z(\mu(t))\right) + \log\left(p(x;t)\right)\right)\mathrm{d}x \\ &-\int \tilde{p}(x;\mu(t)) \\ &\times\frac{\partial}{\partial\mu_\alpha}\left(\sum_{\beta=1}^{\text{cliques}} \mu_\beta(t)\left(V_\beta(x) - \langle V_\alpha\rangle\right)\right)\mathrm{d}x. \end{aligned} \tag{21}$$

If we separate the second integral, we see that the two halves cancel:

$$\begin{aligned}\int \tilde{p}(x;\mu(t))\left(V_\alpha(x) - \langle V_\alpha\rangle\right)\mathrm{d}x &= \int \tilde{p}(x;\mu(t))V_\alpha(x)\mathrm{d}x - \int \tilde{p}(x;\mu(t))\langle V_\alpha\rangle\mathrm{d}x \\ &= \int \tilde{p}(x;\mu(t))V_\alpha(x)\mathrm{d}x - \langle V_\alpha\rangle\int \tilde{p}(x;\mu(t))\mathrm{d}x \\ &= \langle V_\alpha\rangle - \langle V_\alpha\rangle = 0.\end{aligned}$$

We can now absorb the leading minus sign, leaving

$$\begin{aligned}\frac{\partial \mathcal{D}_{\mathcal{KL}}(\mu(t))}{\partial \mu_\alpha} = &\int \tilde{p}(x;\mu(t))\left(V_\alpha(x) - \langle V_\alpha\rangle\right) \\ &\times\left(\sum_{\beta=1}^{\text{cliques}} \mu_\beta(t)V_\beta(x) + \log\left(Z(\mu(t))\right) + \log\left(p(x;t)\right)\right)\mathrm{d}x.\end{aligned} \tag{22}$$

Now we wish to find the derivative of equation (22) with respect to time. In order to accomplish this, we first name pieces of equation (22). Let $A = \tilde{p}(x;\mu(t))\left(V_\alpha(x) - \langle V_\alpha\rangle\right)$, $B = \sum_{\beta=1}^{\text{cliques}} \mu_\beta(t)V_\beta(x)$, $C = \log Z(\mu(t))$, $D = \log p(x;t)$. Then we can write the desired derivative as

$$\frac{d}{dt}\frac{\partial \mathcal{D}_{\mathcal{KL}}(\mu(t))}{\partial \mu_\alpha} = \int\left[A\frac{\partial}{\partial t}B + A\frac{\partial}{\partial t}C + A\frac{\partial}{\partial t}D + (B+C+D)\frac{\partial}{\partial t}A\right]\mathrm{d}x. \tag{23}$$

Concentrating on the first two of these terms, $\int[A\ \partial/\partial t\ B + A\ \partial/\partial t\ C]\mathrm{d}x$, we see that $\partial B/\partial t = \sum_\beta(\partial\mu_\beta/\partial t)V_\beta$ and, using the chain rule, $\partial C/\partial t = \sum_\beta(\partial\mu_\beta/\partial t)(\partial \log Z/\partial\mu_\beta)$. Thus,

$$\begin{aligned}\int[A\frac{\partial}{\partial t}B + A\frac{\partial}{\partial t}C]\mathrm{d}x &= \int A\left[\sum_\beta \frac{\partial\mu_\beta}{\partial t}\left(V_\beta - \langle V_\beta\rangle\right)\right]\mathrm{d}x \\ &= \int \tilde{p}(x;\mu(t))\sum_\beta \frac{\partial\mu_\beta}{\partial t}\left(V_\beta - \langle V_\beta\rangle\right)\left(V_\alpha - \langle V_\alpha\rangle\right)\mathrm{d}x.\end{aligned}$$

Shifting the integral inside the sum, we recognize the expression for the covariance between functions V_α and V_β, where $\langle(x - \langle x\rangle)(y - \langle y\rangle)\rangle = \mathrm{Cov}(x,y) = \langle xy\rangle - \langle x\rangle\langle y\rangle$, so

$$\begin{aligned}\int[A\frac{\partial}{\partial t}B + A\frac{\partial}{\partial t}C]\mathrm{d}x &= \sum_{\beta=1}^{\text{cliques}} \frac{\partial\mu_\beta(t)}{\partial t}\mathrm{Cov}(V_\alpha, V_\beta) \\ &= \sum_{\beta=1}^{\text{cliques}} \frac{\partial\mu_\beta(t)}{\partial t}\left(\langle V_\beta V_\alpha\rangle - \langle V_\alpha\rangle\langle V_\beta\rangle\right).\end{aligned} \tag{24}$$

Turning our attention now to the fourth term of equation (23) :

$$\int \left(\frac{\partial}{\partial t} \left(\tilde{p}(x;\mu(t)) \left(V_\alpha(x) - \langle V_\alpha \rangle \right) \right) \right) \times \left(\left(\sum_{\beta=1}^{\text{cliques}} \mu_\beta(t) V_\beta(x) \right) + \log\left(Z(\mu(t)) \right) + \log\left(p(x;t) \right) \right) \mathrm{d}x,$$

applying product and chain rules to equation (20), the time derivative of A, where

$$\partial / \partial t \; A = \partial \left[\tilde{p}(x;\mu(t)) \left(V_\alpha(x) - \langle V_\alpha \rangle \right) \right] / \partial t \; ,$$

is

$$\begin{aligned} \frac{\partial}{\partial t} A &= \sum_\beta \left[\frac{\partial \mu_\beta}{\partial t} \tilde{p}(x) \left(\langle V_\beta \rangle - V_\beta(x) \right) \left(V_\alpha(x) - \langle V_\alpha \rangle \right) \right] - \tilde{p}(x;\mu(t)) \frac{\partial}{\partial t} \langle V_\alpha \rangle \\ &= -\tilde{p}(x;\mu(t)) \sum_\beta \frac{\partial \mu_\beta}{\partial t} \left(V_\beta(x) - \langle V_\beta \rangle \right) \left(V_\alpha(x) - \langle V_\alpha \rangle \right) \\ &\quad + \tilde{p}(x;\mu(t)) \sum_\beta \frac{\partial \mu_\beta}{\partial t} \left(\langle V_\alpha V_\beta \rangle - \langle V_\alpha \rangle \langle V_\beta \rangle \right) \end{aligned} \tag{25}$$

Note that the covariance between V_α and V_β has appeared again, and that it is being subtracted from terms which have the same form as the terms inside a covariance. That is, integrating over the first part of equation (25) would again produce the covariance between V_α and V_β, times the derivative of μ.

Terms such as $V_\beta(x) - \langle V_\beta \rangle$ recur regularly throughout this derivation. Therefore, we define a new notation. Let $\Delta x \equiv x - \langle x \rangle$. Then we can rewrite the covariance in equation (24) as

$$\sum_{\beta=1}^{\text{cliques}} \frac{\partial \mu_\beta(t)}{\partial t} \left(\langle V_\beta V_\alpha \rangle - \langle V_\alpha \rangle \langle V_\beta \rangle \right) = \sum_{\beta=1}^{\text{cliques}} \frac{\partial \mu_\beta(t)}{\partial t} \langle \Delta V_\alpha \Delta V_\beta \rangle.$$

Additionally, equation (25) fits the same pattern, with

$$x = \left(V_\beta(x) - \langle V_\beta \rangle \right) \left(V_\alpha(x) - \langle V_\alpha \rangle \right) = (\Delta V_\alpha \Delta V_\beta).$$

So, we may rewrite this as

$$\frac{\partial}{\partial t} A = -\tilde{p}(x;\mu(t)) \sum_\beta \frac{\partial \mu_\beta}{\partial t} \Delta(\Delta V_\alpha \Delta V_\beta).$$

Plugging this in and copying the definitions for B, C, and D, we have

$$\begin{aligned} \int (B + C + D) \frac{\partial}{\partial t} A \mathrm{d}x = &- \int \tilde{p}(x;\mu(t)) \sum_\beta \frac{\partial \mu_\beta}{\partial t} \Delta(\Delta V_\alpha \Delta V_\beta) \left(\sum_{\gamma=1}^{\text{cliques}} \mu_\gamma(t) V_\gamma(x) \right) \mathrm{d}x \\ &- \int \tilde{p}(x;\mu(t)) \sum_\beta \frac{\partial \mu_\beta}{\partial t} \Delta(\Delta V_\alpha \Delta V_\beta) \log Z(\mu(t)) \mathrm{d}x \\ &- \int \tilde{p}(x;\mu(t)) \sum_\beta \frac{\partial \mu_\beta}{\partial t} \Delta(\Delta V_\alpha \Delta V_\beta) \log p(x;t) \mathrm{d}x. \end{aligned}$$

In the second of these terms, the factor $\log Z(\mu(t))$ is not a function of x. As a result, when the integral is evaluated, the resulting expected values all cancel. So this term is zero, leaving

$$\begin{aligned}\int (B+C+D)\frac{\partial}{\partial t}A\mathrm{d}x = &-\int \tilde{p}(x;\mu(t))\sum_{\beta}\frac{\partial \mu_\beta}{\partial t}\Delta(\Delta V_\alpha \Delta V_\beta)\left(\sum_{\gamma=1}^{\text{cliques}}\mu_\gamma(t)V_\gamma(x)\right)\mathrm{d}x \\ &-\int \tilde{p}(x;\mu(t))\sum_{\beta}\frac{\partial \mu_\beta}{\partial t}\Delta(\Delta V_\alpha \Delta V_\beta)\log p(x;t)\mathrm{d}x. \end{aligned} \tag{26}$$

Once again, we break apart equation (26) and consider the parts individually. Beginning with the first integral, we regroup the terms of the multiplication so that there is just one double-sum over cliques, then move terms which don't depend on x outside of the integral, and integrate. This leaves another expected value.

$$\begin{aligned}&-\int \tilde{p}(x;\mu(t))\sum_{\beta}\frac{\partial \mu_\beta}{\partial t}\Delta(\Delta V_\alpha \Delta V_\beta)\left(\sum_{\gamma=1}^{\text{cliques}}\mu_\gamma(t)V_\gamma(x)\right)\mathrm{d}x \\ &\qquad = -\sum_{\beta=1}^{\text{cliques}}\sum_{\gamma=1}^{\text{cliques}}\mu_\gamma(t)\frac{\partial \mu_\beta(t)}{\partial t}\langle V_\gamma \Delta\left(\Delta V_\alpha \Delta V_\beta\right)\rangle. \end{aligned} \tag{27}$$

This expression belies some of the symmetry of this term. Note that, if $X = V_\gamma$ and $Y = \left(\Delta V_\alpha \Delta V_\beta\right)$, the inner most term is $\langle X\Delta Y\rangle$. From the definitions of Δ and expectation, this is equivalent to $\langle XY\rangle - \langle X\rangle\langle Y\rangle$, which once again is $\mathrm{Cov}(X,Y)$. Of course, covariance relationships are symmetric, so this is also $\mathrm{Cov}(Y,X)$, which from the preceding argument is $\langle Y\Delta X\rangle$. Thus,

$$\langle X\Delta Y\rangle = \mathrm{Cov}(X,Y) = \langle Y\Delta X\rangle. \tag{28}$$

Applying this transformation to equation (27), we have finally

$$-\sum_{\beta=1}^{\text{cliques}}\sum_{\gamma=1}^{\text{cliques}}\mu_\gamma(t)\frac{\partial \mu_\beta(t)}{\partial t}\langle V_\gamma \Delta\left(\Delta V_\alpha \Delta V_\beta\right)\rangle = -\sum_{\beta=1}^{\text{cliques}}\sum_{\gamma=1}^{\text{cliques}}\mu_\gamma(t)\frac{\partial \mu_\beta(t)}{\partial t}\langle \Delta V_\alpha \Delta V_\beta \Delta V_\gamma\rangle. \tag{29}$$

Turning to the second half of equation (26), the steps are similar, but the V_γ terms are replaced with $\log p(x)$ terms:

$$-\int \tilde{p}(x;\mu(t))\sum_{\beta}\frac{\partial \mu_\beta}{\partial t}\Delta(\Delta V_\alpha \Delta V_\beta)\log p(x;t)\mathrm{d}x = -\sum_{\beta=1}^{\text{cliques}}\frac{\partial \mu_\beta(t)}{\partial t}\langle \Delta V_\alpha \Delta V_\beta \Delta \log p(x;t)\rangle \tag{30}$$

The final piece of equation (23) is

$$\begin{aligned}\int A\frac{\partial}{\partial t}D\mathrm{d}x &= \int \tilde{p}(x;\mu(t))\left(V_\alpha(x) - \langle V_\alpha\rangle\right)\frac{\partial}{\partial t}\log p(x;t)\mathrm{d}x \\ &= \langle \Delta V_\alpha \frac{\partial}{\partial t}\log p(x;t)\rangle. \end{aligned} \tag{31}$$

Now that we have analyzed each of the parts of equation (23), we can set it equal to zero and move the terms with a sum over cliques across the equals sign. Then we have as a solution

$$\begin{aligned}\langle \Delta V_\alpha \frac{\partial}{\partial t}\log\left(p(x;t)\right)\rangle = \sum_{\beta=1}^{\text{cliques}}\frac{\partial \mu_\beta(t)}{\partial t}\Bigg(&\langle \Delta V_\alpha \Delta V_\beta \Delta \log\left(p(x;t)\right)\rangle - \langle \Delta V_\alpha \Delta V_\beta\rangle \\ &+ \sum_{\gamma=1}^{\text{cliques}}\mu_\gamma(t)\langle \Delta V_\alpha \Delta V_\beta \Delta V_\gamma\rangle\Bigg). \end{aligned} \tag{32}$$

Following the Master Equation for the derivative of p, and setting $\partial\mu_\beta(t)/\partial t \equiv f_\beta(\mu(t))$ this expression becomes

$$\langle \Delta V_\alpha \frac{(W \cdot p(;t))(x)}{p(x;t)} \rangle = \sum_{\beta=1}^{\text{cliques}} f_\beta(\mu(t)) \left(\langle \Delta V_\alpha \Delta V_\beta \Delta \log p(x;t) \rangle - \langle \Delta V_\alpha \Delta V_\beta \rangle + \sum_{\gamma=1}^{\text{cliques}} \mu_\gamma(t) \langle \Delta V_\alpha \Delta V_\beta \Delta V_\gamma \rangle \right).$$

Then, using equation (12) to define a linear form for f_β, we have

$$\langle \Delta V_\alpha \frac{(W \cdot p(;t))(x)}{p(x;t)} \rangle = \sum_{\beta=1}^{\text{cliques}} \sum_{A=1}^{\text{bases}} \theta_{\beta A} f_A(\mu(t)) \left(\langle \Delta V_\alpha \Delta V_\beta \Delta \log p(x;t) \rangle - \langle \Delta V_\alpha \Delta V_\beta \rangle + \sum_{\gamma=1}^{\text{cliques}} \mu_\gamma(t) \langle \Delta V_\alpha \Delta V_\beta \Delta V_\gamma \rangle \right).$$

The p expressions in numerator and denominator may be evaluated using the BMLA-trained approximation of $p(x,t)$ at time t. Then, these expressions are finally in a form which can be evaluated during Monte Carlo simulations of $\tilde{p}$, resulting in an online learning algorithm in the spirit of BMLA itself, though more complicated. To this end we now define a vector $\mathbf{B}$ with α entries

$$\mathbf{B}_\alpha = \langle \Delta V_\alpha \frac{(W \cdot p(;t))(x)}{p(x;t)} \rangle, \tag{33}$$

and a structured α-by-(β, A) matrix $\mathbf{A}$ with entries

$$\mathbf{A}_{\alpha,(\beta,A)} = f_A(\mu(t)) \left(\langle \Delta V_\alpha \Delta V_\beta \Delta \log p(x;t) \rangle - \langle \Delta V_\alpha \Delta V_\beta \rangle + \sum_{\gamma=1}^{\text{cliques}} \mu_\gamma(t) \langle \Delta V_\alpha \Delta V_\beta \Delta V_\gamma \rangle \right). \tag{34}$$

Then $\mathbf{B} = \mathbf{A} \cdot \theta$, where the dot product is taken over the compound index (β, A). Finally, by calculating values for $\mathbf{A}$ and $\mathbf{B}$ we can solve for optimal θ.

It will generally be true that this system of equations is under-determined, as the dimensions of the matrix $\mathbf{A}$ are $n \times (n \times m)$, where n is the number of potentials in the MRF and m is the total number of bases, and vector $\mathbf{B}$ has length n. Therefore, it is useful to build larger $\mathbf{A}$ and $\mathbf{B}$ matrices by *stacking* together in a block fashion several copies of equations (33) and (34) together which have been computed using different distributions p and $\tilde{p}$, coming from different initial conditions or other input conditions as suggested in the operator approximations of equations (2) and (4), and characterized by different values of μ. As long as these copies are linearly independent this procedure can produce a fully constrained system of equations.

7 Supplementary Information

Title: Model Reduction for Stochastic CaMKII Reaction Kinetics in Synapses by Graph-Constrained Correlation Dynamics

Authors: Todd Johnson, Tom Bartol, Terrence Sejnowski, and Eric Mjolsness

Journal: Physical Biology, 2015

7.1 Further Plots

Further experimental results are shown in figures 6-8 below. Figures 6 and 7 show the average of all the moments controlled by each interaction parameter in the corresponding figures 3 and 4. Figure 8 shows the in-sample fitting of GCCD dynamics to a long pulse train.

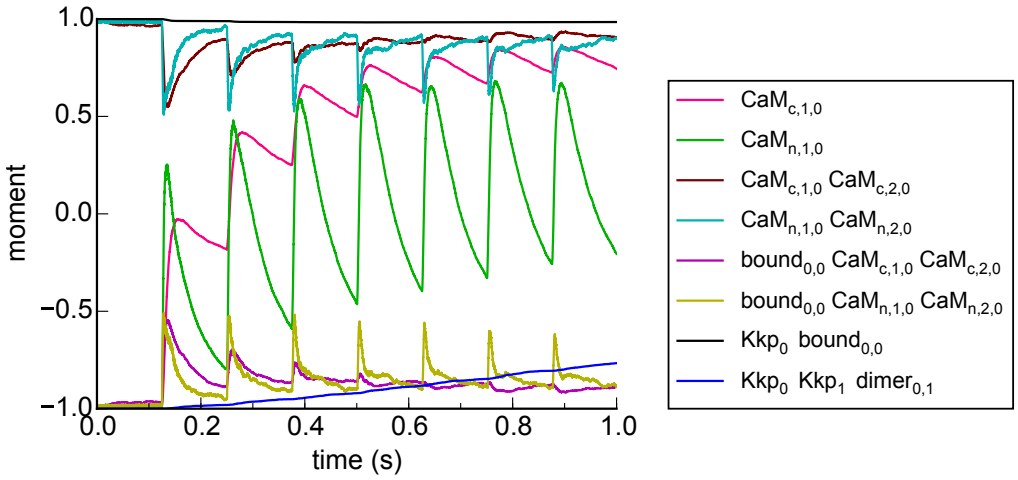

Figure 6: BMLA-inferred moments $\langle V_\alpha \rangle(t)$ for the Plenum model corresponding to the interaction parameters $\mu_\alpha(t)$ of figure 3. Each moment is an average over all interaction monomials of the corresponding form as labelled in figure 2, i.e. an average over a weight-sharing category α of interactions. Since all binary variables take values ± 1, the resulting monomials $V_{\alpha,\gamma}$ and their weight-sharing category averages V_α (without scalar factor μ_α) have ensemble-average values i.e. moments $\langle V_\alpha \rangle$ in the interval $[-1,+1]$.

7.1.1 Relation to concentrations

We now discuss lower and upper bounds for concentrations in the trained coarse-scale model. The evolving moments $\langle V_\alpha \rangle(t)$ plotted in figures 6 and 7 all take values in the interval $[-1,1]$ because they are averages (both within weight-sharing category α and under the Boltzmann distribution) of monomials $V_\alpha(s)$ that take values in $\{-1,+1\}$, which in turn is because each constituent $V_{\alpha,\gamma}(s)$ is a product of random variables s_I that take values in $\{-1,+1\}$. (Here the generic index I stands for any of the multi-indices used in the main text, e.g. $(\mathrm{c/n}, a, i)$ in $\mathrm{CaM}_{\mathrm{c/n},a,i}$.) On the other hand these fine-scale variables s_I represent Boolean distinctions (unbound/bound, unphosphorylated/phosphorylated, and undimerized/dimerized) that are more conventionally assigned indicator variables w_I taking values in $\{0,1\}$. An advantage of the $\{0,1\}$ encoding is that single-variable moments become probabilities: $\langle w_I \rangle_{Pr} = Pr(w_I = 1)$. Fortunately there is a trivial affine map from s to w: $w_I = (s_I + 1)/2$ and inversely $s_i = 2w_i - 1$. This mapping will likewise take every $\langle V_{\alpha,\gamma}(s) \rangle \in [-1,1]$ (by substitution of the inverse mapping) to a linear combination of $\langle V_{\alpha,\gamma}(w) \rangle \in [0,1]$ and lower-degree moments,

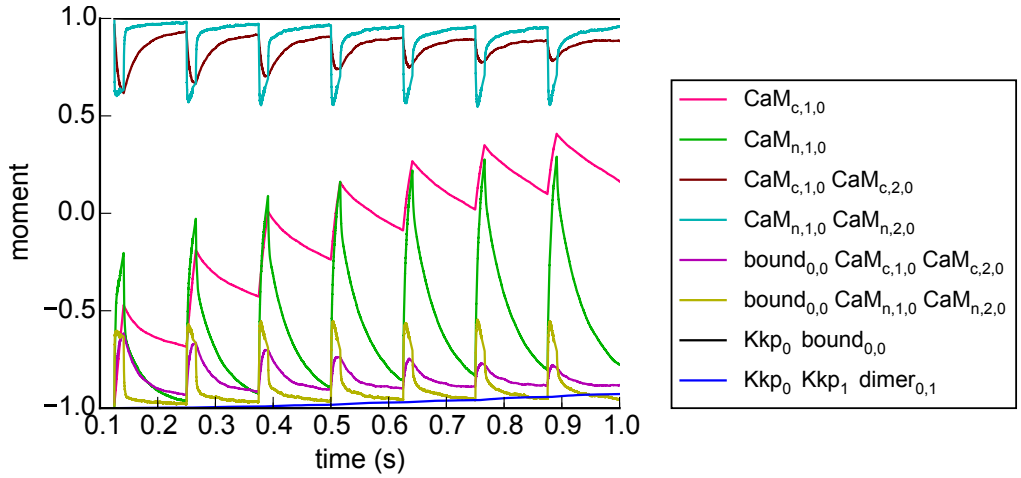

Figure 7: BMLA-inferred moments $\langle V_\alpha \rangle(t)$ for the MCell model corresponding to the interaction parameters $\mu_\alpha(t)$ of figure 4. Other details as in figure 6.

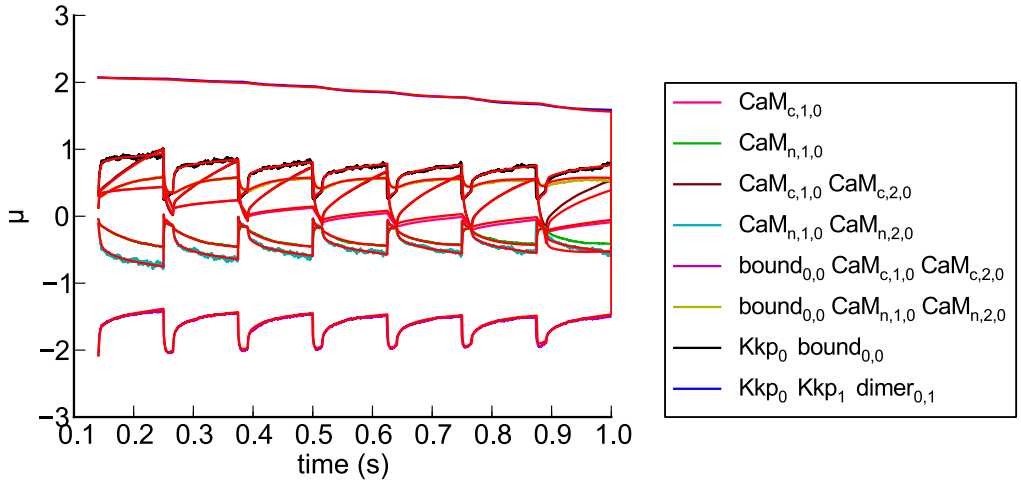

Figure 8: In-sample trained evolution of interaction parameters μ for the MCell model over a long pulse train (7 cycles at 8 Hz) using a pulse train (rectangular wave) for calcium influx. Compared to out-of-sample plots in Figure 5, here the exact waveforms are fitted more accurately; but predictive power is only demonstrated out-of-sample.

and therefore will also take every $\langle V_\alpha(s) \rangle \in [-1, 1]$ to a linear combination of $\langle V_\alpha(w) \rangle \in [0, 1]$ and lower-degree moments. Degree-one single-variable moments $\langle \mathrm{CaM}_{c,a,i} \rangle$ and $\langle \mathrm{CaM}_{n,a,i} \rangle$ simply get remapped to site occupancy probabilities (averaged over weight-sharing category) taking values in $[0, 1]$ rather than $[-1, 1]$, which for those two moments would just change the labeling of the vertical axis in figures 6 and 7. Other single-variable moments could be computed from the Boltzmann distribution equation (11) and plotted in a similar manner.

Converting such moments into concentrations requires multiplying the known global concentration of a particular molecule that has binding sites (or other modification sites), such as CaM or CaMKII subunit monomer, by the probability of that molecule being in the binding state (and/or other micro-states) of interest. The latter probability can be calculated using moments as above, or more directly using the Boltzmann distribution, or estimated using Monte Carlo sampling algorithms, all using the known values of interaction parameters $\mu(t)$ shown in figures 3-5. By equation

(11) for the Boltzmann distribution they are all normalized probabilities that must be in the interval [0,1]. Therefore the required concentration will always be in the interval between zero and the upper bound given by the known global concentration for that species if its binding sites and phosphorylation states are all ignored, i.e. summed over. In the CaMKII model, the global concentration (summed over binding sites and phosphorylation states) is fixed for CaM and for CaMKII subunit, though CaMKII subunits are dynamically partitioned into monomer and dimer super-species each of which has many such micro-states. By conservation of CaMKII subunits, the population of each CaMKII dimer state is more tightly upper bounded by the maximum possible number of dimers, which is half the maximum number of CaMKII monomers in the volume studied, rounded down to the nearest integer.

7.2 Alpha model parameters

The alpha model waveform is [37]

$$\alpha(t|\tau_1, \tau_2) = c \exp(-t/\tau_1)(1 - \exp(-t/\tau_2) \tag{35}$$

and we have used $c = 5747.53$ micro molar, $\tau_1 = 5.9$ msec, $\tau_2 = 1244.55$ msec. With these parameters, this function attains a maximum value of 10 micromolar at $t = 5.89$ msec.

7.3 Source codes

The software used to compute the results in this paper is of several different kinds, all of which we make available for pure research. (1) MCell [3] is well established and available at :

`http://www.mcell.org/download.html`

It is used for fine-scale simulations as described in the main text. It implements a general-purpose declarative biological modeling language, and thus requires a model file which is largely presented below. (2) Plenum [2] is the UC Irvine PhD thesis code of Guy Yosiphon and is available at :

`http://computableplant.ics.uci.edu/theses/guy/downloads/DGPublications.html`

It is used for fine-scale simulations as described above. It implements a general-purpose declarative biological modeling language, and thus requires a model file which is largely presented below. (3) Dependency Diagrams [13] is part of the UC Irvine PhD code of the first author and is available at:

`http://computableplant.ics.uci.edu/sw/dd/`

It is used to implement a standard Monte Carlo (heat-bath) algorithm for sampling the Boltzmann distribution over binary-valued random variables, including cardinality constraints on binding site variables, in the GCCD method according to equation (11). (4) Special-purpose GCCD code. This code implements: (a) the restriction map by modified BMLA algorithm, using the junction tree algorithm (standard in machine learning) to evaluate exactly the gradient of KL divergence followed in BMLA, rather than using sampling to evaluate the gradient stochastically; (b) the time-evolution of the interaction parameters μ_α according to equation (12), and for the generation of data files corresponding to figures 3-5 and 6-8. It assumes that values for the trainable parameters θ are presented as input; (c) the GCCD code also implements the training of model parameters θ by lasso-regularized linear regression as described in [13], the UC Irvine PhD thesis of the first author. GCCD code is available at:

`http://computableplant.ics.uci.edu/sw/gccd/`

But all this GCCD-specific code is also available as a separate Supplementary Information file with the present paper. Codes (2)-(4) above are written using, and require, the commercially available "Mathematica" computer algebra system. Code (4) also requires the Python and R computer languages, both available non-commercially.

7.4 Plenum model

The Plenum [2] (Dynamical Grammars) model file for CaMKII model [1] has parameterized reactions or "rules" specified as follows:

```
(* this keeps track of time in the simulation, so we can have timed spikes *)
c==clock[time,state] -> c, solving[time'==1/timeMultiplier],

(* add Ca if it's time to do so *)
{c==clock[time,s], Ca[ii]} -> {clock[time,1], Ca[spikeTrainSize]},
with[If[ii < spikeTrainSize, spikeOn[time,s]/timeMultiplier, 0]],

(* remove Ca if it's time to do so *)
{c==clock[time,s], Ca[ii]} -> {clock[time,0], Ca[baseSize]},
with[If[ii > 0,spikeOff[time,s]/timeMultiplier, 0]],

(* Ca binding/unbinding CaM *)
{Ca[ii], CaM[n,c]} -> {Ca[ii], CaM[n+1,c]},
    with[ii * If[n < amax, kloadn[n,c], 0]/timeMultiplier],
{Ca[ii], CaM[n,c]} -> {Ca[ii], CaM[n-1,c]},
    with[If[n > 0, kunloadn[n,c], 0]/timeMultiplier],
{Ca[ii], CaM[n,c]} -> {Ca[ii], CaM[n,c+1]},
    with[ii * If[c < amax, kloadc[n,c], 0]/timeMultiplier],
{Ca[ii], CaM[n,c]} -> {Ca[ii], CaM[n,c-1]},
    with[If[c > 0, kunloadc[n,c], 0]/timeMultiplier],

(* Ca binding/unbinding CaM bound to CaMKII *)
{Ca[ii], Kk[a0,b0,0]} -> {Ca[ii], Kk[a0+1,b0,0]},
    with[ii*If[a0<amax, kload2n[a0,b0,0], 0]/timeMultiplier],
{Ca[ii], Kk[a0,b0,0]} -> {Ca[ii], Kk[a0,b0+1,0]},
    with[ii*If[b0<amax, kload2c[a0,b0,0],0]/timeMultiplier],
{Ca[ii], Kk[a0,b0,0]} -> {Ca[ii], Kk[a0-1,b0,0]},
    with[If[a0>0,kunload2n[a0,b0,0],0]/timeMultiplier],
{Ca[ii], Kk[a0,b0,0]} -> {Ca[ii], Kk[a0,b0-1,0]},
    with[If[b0>0,kunload2c[a0,b0,0],0]/timeMultiplier],

(* CaM binding/unbinding free CaMKII *)
{CaM[n,c], CaMKII[num]} -> {Kk[n,c,0], CaMKII[num-1]},
    with[num*kon2[n,c,p0]/timeMultiplier],
{Kk[a0,b0,0], CaMKII[num]} -> {CaM[a0,b0], CaMKII[num+1]},
    with[koff2[a0,b0,0]If[a0>=0&&b0>=0,1,0]/timeMultiplier],

(* Dimerization *)
{Kk[a0,b0,p0], Kk[a1,b1,p1]} -> {Dimer[a0,b0,p0,a1,b1,p1]},
    with[If[p0<1||p1<1 && a0<=a1,
        kdimerize[a0,b0,p0,a1,b1,p1]/timeMultiplier,0]],
{Dimer[a0,b0,p0,a1,b1,p1]} -> {Kk[a0,b0,p0],Kk[a1,b1,p1]},
    with[kundimerize[a0,b0,p0,a1,b1,p1]/timeMultiplier],

(* phosphorylation *)
```

```
Dimer[a0,b0,p0,a1,b1,p1] -> {Kk[a0,b0,1],Kk[a1,b1,p1]},
    with[If[a0>=0&&b0>=0&&p0<1,kautotop[a0,b0,a1,b1,p1],0]/timeMultiplie],
Dimer[a0,b0,p0,a1,b1,p1] -> {Kk[a0,b0,p0],Kk[a1,b1,1]},
    with[If[a1>=0&&b1>=0&&p1<1,kautobot[a0,b0,p0,a1,b1],0]/timeMultiplier]
```

This model file requires also definitions of the functions

```
spikeOn, spikeOff, kloadn, kunloadn, kloadc, kunloadc,
kload2n, kload2c, kunload2n, kunload2c, kon2, koff2,
kdimerize, kundimerize, kautotop, kautobot
```

which are determined by the published model [1] and defined in the full model notebook and support files, available with the source code for this paper. The more arguments appear in such rate functions, the greater the degree of combinatorial notational compaction that is afforded by parameterized reactions, or rules with rates, over ordinary reaction notation.

7.5 MCell model

MCell model file for CaMKII model [1] has parameterized reactions or "rules" specified as follows:

```
DEFINE_MOLECULES
{

  // Calcium
  Ca {DIFFUSION_CONSTANT = DCa}

  // mCaMKII
  K {DIFFUSION_CONSTANT = DK CUSTOM_SPACE_STEP = lr_mCaMKII}

  // CaM, 9 states
  // Fig. 2, Pepke et al., 2011 PLoSCompBio
  CaM0N0C {DIFFUSION_CONSTANT = DCaM CUSTOM_SPACE_STEP = lr_CaM}
  CaM1N0C {DIFFUSION_CONSTANT = DCaM CUSTOM_SPACE_STEP = lr_CaM}
  CaM2N0C {DIFFUSION_CONSTANT = DCaM CUSTOM_SPACE_STEP = lr_CaM}
  CaM0N1C {DIFFUSION_CONSTANT = DCaM CUSTOM_SPACE_STEP = lr_CaM}
  CaM1N1C {DIFFUSION_CONSTANT = DCaM CUSTOM_SPACE_STEP = lr_CaM}
  CaM2N1C {DIFFUSION_CONSTANT = DCaM CUSTOM_SPACE_STEP = lr_CaM}
  CaM0N2C {DIFFUSION_CONSTANT = DCaM CUSTOM_SPACE_STEP = lr_CaM}
  CaM1N2C {DIFFUSION_CONSTANT = DCaM CUSTOM_SPACE_STEP = lr_CaM}
  CaM2N2C {DIFFUSION_CONSTANT = DCaM CUSTOM_SPACE_STEP = lr_CaM}

  // KCaM, 9 states
  // Fig. 2, Pepke et al., 2011 PLoSCompBio
  KCaM0N0C {DIFFUSION_CONSTANT = DKCaM CUSTOM_SPACE_STEP = lr_KCaM}
  KCaM1N0C {DIFFUSION_CONSTANT = DKCaM CUSTOM_SPACE_STEP = lr_KCaM}
  KCaM2N0C {DIFFUSION_CONSTANT = DKCaM CUSTOM_SPACE_STEP = lr_KCaM}
  KCaM0N1C {DIFFUSION_CONSTANT = DKCaM CUSTOM_SPACE_STEP = lr_KCaM}
  KCaM1N1C {DIFFUSION_CONSTANT = DKCaM CUSTOM_SPACE_STEP = lr_KCaM}
  KCaM2N1C {DIFFUSION_CONSTANT = DKCaM CUSTOM_SPACE_STEP = lr_KCaM}
  KCaM0N2C {DIFFUSION_CONSTANT = DKCaM CUSTOM_SPACE_STEP = lr_KCaM}
  KCaM1N2C {DIFFUSION_CONSTANT = DKCaM CUSTOM_SPACE_STEP = lr_KCaM}
  KCaM2N2C {DIFFUSION_CONSTANT = DKCaM CUSTOM_SPACE_STEP = lr_KCaM}

  // pKCaM, 9 states
  // Fig. 6, Pepke et al., 2011 PLoSCompBio
  pKCaM0N0C {DIFFUSION_CONSTANT = DKCaM CUSTOM_SPACE_STEP = lr_KCaM}
  pKCaM1N0C {DIFFUSION_CONSTANT = DKCaM CUSTOM_SPACE_STEP = lr_KCaM}
  pKCaM2N0C {DIFFUSION_CONSTANT = DKCaM CUSTOM_SPACE_STEP = lr_KCaM}
  pKCaM0N1C {DIFFUSION_CONSTANT = DKCaM CUSTOM_SPACE_STEP = lr_KCaM}
  pKCaM1N1C {DIFFUSION_CONSTANT = DKCaM CUSTOM_SPACE_STEP = lr_KCaM}
  pKCaM2N1C {DIFFUSION_CONSTANT = DKCaM CUSTOM_SPACE_STEP = lr_KCaM}
  pKCaM0N2C {DIFFUSION_CONSTANT = DKCaM CUSTOM_SPACE_STEP = lr_KCaM}
  pKCaM1N2C {DIFFUSION_CONSTANT = DKCaM CUSTOM_SPACE_STEP = lr_KCaM}
  pKCaM2N2C {DIFFUSION_CONSTANT = DKCaM CUSTOM_SPACE_STEP = lr_KCaM}
```

```
// KCaM-KCaM dimers, 45 states
// Fig. 6, Pepke et al., 2011 PLoSCompBio
KCaM0N0C_KCaM0N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM0N0C_KCaM1N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM0N0C_KCaM2N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM0N0C_KCaM0N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM0N0C_KCaM1N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM0N0C_KCaM2N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM0N0C_KCaM0N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM0N0C_KCaM1N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM0N0C_KCaM2N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM1N0C_KCaM1N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM1N0C_KCaM2N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM1N0C_KCaM0N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM1N0C_KCaM1N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM1N0C_KCaM2N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM1N0C_KCaM0N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM1N0C_KCaM1N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM1N0C_KCaM2N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM2N0C_KCaM2N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM2N0C_KCaM0N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM2N0C_KCaM1N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM2N0C_KCaM2N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM2N0C_KCaM0N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM2N0C_KCaM1N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM2N0C_KCaM2N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM0N1C_KCaM0N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM0N1C_KCaM1N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM0N1C_KCaM2N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM0N1C_KCaM0N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM0N1C_KCaM1N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM0N1C_KCaM2N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM1N1C_KCaM1N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM1N1C_KCaM2N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM1N1C_KCaM0N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM1N1C_KCaM1N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM1N1C_KCaM2N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM2N1C_KCaM2N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM2N1C_KCaM0N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM2N1C_KCaM1N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM2N1C_KCaM2N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM0N2C_KCaM0N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM0N2C_KCaM1N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM0N2C_KCaM2N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM1N2C_KCaM1N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM1N2C_KCaM2N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
KCaM2N2C_KCaM2N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}

// pKCaM-KCaM dimers, 81 states
```

```
// Fig. 6, Pepke et al., 2011 PLoSCompBio
pKCaM0N0C_KCaM0N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM0N0C_KCaM1N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM0N0C_KCaM2N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM0N0C_KCaM0N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM0N0C_KCaM1N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM0N0C_KCaM2N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM0N0C_KCaM0N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM0N0C_KCaM1N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM0N0C_KCaM2N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM1N0C_KCaM0N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM1N0C_KCaM1N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM1N0C_KCaM2N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM1N0C_KCaM0N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM1N0C_KCaM1N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM1N0C_KCaM2N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM1N0C_KCaM0N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM1N0C_KCaM1N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM1N0C_KCaM2N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM2N0C_KCaM0N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM2N0C_KCaM1N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM2N0C_KCaM2N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM2N0C_KCaM0N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM2N0C_KCaM1N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM2N0C_KCaM2N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM2N0C_KCaM0N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM2N0C_KCaM1N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM2N0C_KCaM2N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM0N1C_KCaM0N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM0N1C_KCaM1N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM0N1C_KCaM2N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM0N1C_KCaM0N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM0N1C_KCaM1N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM0N1C_KCaM2N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM0N1C_KCaM0N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM0N1C_KCaM1N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM0N1C_KCaM2N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM1N1C_KCaM0N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM1N1C_KCaM1N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM1N1C_KCaM2N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM1N1C_KCaM0N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM1N1C_KCaM1N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM1N1C_KCaM2N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM1N1C_KCaM0N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM1N1C_KCaM1N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM1N1C_KCaM2N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM2N1C_KCaM0N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM2N1C_KCaM1N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
pKCaM2N1C_KCaM2N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
```

```
  pKCaM2N1C_KCaM0N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM2N1C_KCaM1N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM2N1C_KCaM2N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM2N1C_KCaM0N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM2N1C_KCaM1N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM2N1C_KCaM2N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM0N2C_KCaM0N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM0N2C_KCaM1N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM0N2C_KCaM2N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM0N2C_KCaM0N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM0N2C_KCaM1N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM0N2C_KCaM2N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM0N2C_KCaM0N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM0N2C_KCaM1N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM0N2C_KCaM2N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM1N2C_KCaM0N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM1N2C_KCaM1N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM1N2C_KCaM2N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM1N2C_KCaM0N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM1N2C_KCaM1N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM1N2C_KCaM2N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM1N2C_KCaM0N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM1N2C_KCaM1N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM1N2C_KCaM2N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM2N2C_KCaM0N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM2N2C_KCaM1N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM2N2C_KCaM2N0C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM2N2C_KCaM0N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM2N2C_KCaM1N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM2N2C_KCaM2N1C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM2N2C_KCaM0N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM2N2C_KCaM1N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
  pKCaM2N2C_KCaM2N2C {DIFFUSION_CONSTANT = Ddimer CUSTOM_SPACE_STEP = lr_dimer}
}

DEFINE_REACTIONS
{

  // Calcium interaction with CaM, 12 rxns
  // Fig. 2, Pepke et al., 2011 PLoSCompBio
  Ca + CaM0N0C <-> CaM1N0C [>Kon1N, <Koff1N]
  Ca + CaM1N0C <-> CaM2N0C [>Kon2N, <Koff2N]
  Ca + CaM0N0C <-> CaM0N1C [>Kon1C, <Koff1C]
  Ca + CaM1N0C <-> CaM1N1C [>Kon1C, <Koff1C]
  Ca + CaM2N0C <-> CaM2N1C [>Kon1C, <Koff1C]
  Ca + CaM0N1C <-> CaM1N1C [>Kon1N, <Koff1N]
  Ca + CaM1N1C <-> CaM2N1C [>Kon2N, <Koff2N]
  Ca + CaM0N1C <-> CaM0N2C [>Kon2C, <Koff2C]
  Ca + CaM1N1C <-> CaM1N2C [>Kon2C, <Koff2C]
```

```
Ca + CaM2N1C <-> CaM2N2C [>Kon2C, <Koff2C]
Ca + CaM0N2C <-> CaM1N2C [>Kon1N, <Koff1N]
Ca + CaM1N2C <-> CaM2N2C [>Kon2N, <Koff2N]

// CaM interaction with mCaMKII (K), 9 rxns
// Fig. 2, Pepke et al., 2011 PLoSCompBio
K + CaM0N0C <-> KCaM0N0C [>KonCaM0N0C, <KoffCaM0N0C]
K + CaM1N0C <-> KCaM1N0C [>KonCaM1N0C, <KoffCaM1N0C]
K + CaM2N0C <-> KCaM2N0C [>KonCaM2N0C, <KoffCaM2N0C]
K + CaM0N1C <-> KCaM0N1C [>KonCaM0N1C, <KoffCaM0N1C]
K + CaM1N1C <-> KCaM1N1C [>KonCaM1N1C, <KoffCaM1N1C]
K + CaM2N1C <-> KCaM2N1C [>KonCaM2N1C, <KoffCaM2N1C]
K + CaM0N2C <-> KCaM0N2C [>KonCaM0N2C, <KoffCaM0N2C]
K + CaM1N2C <-> KCaM1N2C [>KonCaM1N2C, <KoffCaM1N2C]
K + CaM2N2C <-> KCaM2N2C [>KonCaM2N2C, <KoffCaM2N2C]

// Calcium interaction with KCaM, 12 rxns
// Fig. 2, Pepke et al., 2011 PLoSCompBio
Ca + KCaM0N0C <-> KCaM1N0C [>KonK1N, <KoffK1N]
Ca + KCaM1N0C <-> KCaM2N0C [>KonK2N, <KoffK2N]
Ca + KCaM0N0C <-> KCaM0N1C [>KonK1C, <KoffK1C]
Ca + KCaM1N0C <-> KCaM1N1C [>KonK1C, <KoffK1C]
Ca + KCaM2N0C <-> KCaM2N1C [>KonK1C, <KoffK1C]
Ca + KCaM0N1C <-> KCaM1N1C [>KonK1N, <KoffK1N]
Ca + KCaM1N1C <-> KCaM2N1C [>KonK2N, <KoffK2N]
Ca + KCaM0N1C <-> KCaM0N2C [>KonK2C, <KoffK2C]
Ca + KCaM1N1C <-> KCaM1N2C [>KonK2C, <KoffK2C]
Ca + KCaM2N1C <-> KCaM2N2C [>KonK2C, <KoffK2C]
Ca + KCaM0N2C <-> KCaM1N2C [>KonK1N, <KoffK1N]
Ca + KCaM1N2C <-> KCaM2N2C [>KonK2N, <KoffK2N]

// KCaM autophosphorylation:
// Scheme, Fig. 6, Pepke et al., 2011 PLoSCompBio
// Rates, Table S1, Pepke et al., 2011 PLoSCompBio

// KCaM autophosphorylation of KCaM, 45 combinations, 126 rxns:
KCaM0N0C + KCaM0N0C <-> KCaM0N0C_KCaM0N0C [>KonCaMKII, <KoffCaMKII]
KCaM0N0C_KCaM0N0C -> pKCaM0N0C + KCaM0N0C [2*KpCaM0N0C]
KCaM0N0C + KCaM1N0C <-> KCaM0N0C_KCaM1N0C [>KonCaMKII, <KoffCaMKII]
KCaM0N0C_KCaM1N0C -> pKCaM0N0C + KCaM1N0C [KpCaM0N0C]
KCaM0N0C_KCaM1N0C -> KCaM0N0C + pKCaM1N0C [KpCaM1N0C]
KCaM0N0C + KCaM2N0C <-> KCaM0N0C_KCaM2N0C [>KonCaMKII, <KoffCaMKII]
KCaM0N0C_KCaM2N0C -> pKCaM0N0C + KCaM2N0C [KpCaM0N0C]
KCaM0N0C_KCaM2N0C -> KCaM0N0C + pKCaM2N0C [KpCaM2N0C]
KCaM0N0C + KCaM0N1C <-> KCaM0N0C_KCaM0N1C [>KonCaMKII, <KoffCaMKII]
KCaM0N0C_KCaM0N1C -> pKCaM0N0C + KCaM0N1C [KpCaM0N0C]
KCaM0N0C_KCaM0N1C -> KCaM0N0C + pKCaM0N1C [KpCaM0N1C]
KCaM0N0C + KCaM1N1C <-> KCaM0N0C_KCaM1N1C [>KonCaMKII, <KoffCaMKII]
KCaM0N0C_KCaM1N1C -> pKCaM0N0C + KCaM1N1C [KpCaM0N0C]
```

```
KCaM0N0C_KCaM1N1C -> KCaM0N0C + pKCaM1N1C [KpCaM1N1C]
KCaM0N0C + KCaM2N1C <-> KCaM0N0C_KCaM2N1C [>KonCaMKII, <KoffCaMKII]
KCaM0N0C_KCaM2N1C -> pKCaM0N0C + KCaM2N1C [KpCaM0N0C]
KCaM0N0C_KCaM2N1C -> KCaM0N0C + pKCaM2N1C [KpCaM2N1C]
KCaM0N0C + KCaM0N2C <-> KCaM0N0C_KCaM0N2C [>KonCaMKII, <KoffCaMKII]
KCaM0N0C_KCaM0N2C -> pKCaM0N0C + KCaM0N2C [KpCaM0N0C]
KCaM0N0C_KCaM0N2C -> KCaM0N0C + pKCaM0N2C [KpCaM0N2C]
KCaM0N0C + KCaM1N2C <-> KCaM0N0C_KCaM1N2C [>KonCaMKII, <KoffCaMKII]
KCaM0N0C_KCaM1N2C -> pKCaM0N0C + KCaM1N2C [KpCaM0N0C]
KCaM0N0C_KCaM1N2C -> KCaM0N0C + pKCaM1N2C [KpCaM1N2C]
KCaM0N0C + KCaM2N2C <-> KCaM0N0C_KCaM2N2C [>KonCaMKII, <KoffCaMKII]
KCaM0N0C_KCaM2N2C -> pKCaM0N0C + KCaM2N2C [KpCaM0N0C]
KCaM0N0C_KCaM2N2C -> KCaM0N0C + pKCaM2N2C [KpCaM2N2C]
KCaM1N0C + KCaM1N0C <-> KCaM1N0C_KCaM1N0C [>KonCaMKII, <KoffCaMKII]
KCaM1N0C_KCaM1N0C -> pKCaM1N0C + KCaM1N0C [2*KpCaM1N0C]
KCaM1N0C + KCaM2N0C <-> KCaM1N0C_KCaM2N0C [>KonCaMKII, <KoffCaMKII]
KCaM1N0C_KCaM2N0C -> pKCaM1N0C + KCaM2N0C [KpCaM1N0C]
KCaM1N0C_KCaM2N0C -> KCaM1N0C + pKCaM2N0C [KpCaM2N0C]
KCaM1N0C + KCaM0N1C <-> KCaM1N0C_KCaM0N1C [>KonCaMKII, <KoffCaMKII]
KCaM1N0C_KCaM0N1C -> pKCaM1N0C + KCaM0N1C [KpCaM1N0C]
KCaM1N0C_KCaM0N1C -> KCaM1N0C + pKCaM0N1C [KpCaM0N1C]
KCaM1N0C + KCaM1N1C <-> KCaM1N0C_KCaM1N1C [>KonCaMKII, <KoffCaMKII]
KCaM1N0C_KCaM1N1C -> pKCaM1N0C + KCaM1N1C [KpCaM1N0C]
KCaM1N0C_KCaM1N1C -> KCaM1N0C + pKCaM1N1C [KpCaM1N1C]
KCaM1N0C + KCaM2N1C <-> KCaM1N0C_KCaM2N1C [>KonCaMKII, <KoffCaMKII]
KCaM1N0C_KCaM2N1C -> pKCaM1N0C + KCaM2N1C [KpCaM1N0C]
KCaM1N0C_KCaM2N1C -> KCaM1N0C + pKCaM2N1C [KpCaM2N1C]
KCaM1N0C + KCaM0N2C <-> KCaM1N0C_KCaM0N2C [>KonCaMKII, <KoffCaMKII]
KCaM1N0C_KCaM0N2C -> pKCaM1N0C + KCaM0N2C [KpCaM1N0C]
KCaM1N0C_KCaM0N2C -> KCaM1N0C + pKCaM0N2C [KpCaM0N2C]
KCaM1N0C + KCaM1N2C <-> KCaM1N0C_KCaM1N2C [>KonCaMKII, <KoffCaMKII]
KCaM1N0C_KCaM1N2C -> pKCaM1N0C + KCaM1N2C [KpCaM1N0C]
KCaM1N0C_KCaM1N2C -> KCaM1N0C + pKCaM1N2C [KpCaM1N2C]
KCaM1N0C + KCaM2N2C <-> KCaM1N0C_KCaM2N2C [>KonCaMKII, <KoffCaMKII]
KCaM1N0C_KCaM2N2C -> pKCaM1N0C + KCaM2N2C [KpCaM1N0C]
KCaM1N0C_KCaM2N2C -> KCaM1N0C + pKCaM2N2C [KpCaM2N2C]
KCaM2N0C + KCaM2N0C <-> KCaM2N0C_KCaM2N0C [>KonCaMKII, <KoffCaMKII]
KCaM2N0C_KCaM2N0C -> pKCaM2N0C + KCaM2N0C [2*KpCaM2N0C]
KCaM2N0C + KCaM0N1C <-> KCaM2N0C_KCaM0N1C [>KonCaMKII, <KoffCaMKII]
KCaM2N0C_KCaM0N1C -> pKCaM2N0C + KCaM0N1C [KpCaM2N0C]
KCaM2N0C_KCaM0N1C -> KCaM2N0C + pKCaM0N1C [KpCaM0N1C]
KCaM2N0C + KCaM1N1C <-> KCaM2N0C_KCaM1N1C [>KonCaMKII, <KoffCaMKII]
KCaM2N0C_KCaM1N1C -> pKCaM2N0C + KCaM1N1C [KpCaM2N0C]
KCaM2N0C_KCaM1N1C -> KCaM2N0C + pKCaM1N1C [KpCaM1N1C]
KCaM2N0C + KCaM2N1C <-> KCaM2N0C_KCaM2N1C [>KonCaMKII, <KoffCaMKII]
KCaM2N0C_KCaM2N1C -> pKCaM2N0C + KCaM2N1C [KpCaM2N0C]
KCaM2N0C_KCaM2N1C -> KCaM2N0C + pKCaM2N1C [KpCaM2N1C]
KCaM2N0C + KCaM0N2C <-> KCaM2N0C_KCaM0N2C [>KonCaMKII, <KoffCaMKII]
KCaM2N0C_KCaM0N2C -> pKCaM2N0C + KCaM0N2C [KpCaM2N0C]
```

```
KCaM2N0C_KCaM0N2C -> KCaM2N0C + pKCaM0N2C [KpCaM0N2C]
KCaM2N0C + KCaM1N2C <-> KCaM2N0C_KCaM1N2C [>KonCaMKII, <KoffCaMKII]
KCaM2N0C_KCaM1N2C -> pKCaM2N0C + KCaM1N2C [KpCaM2N0C]
KCaM2N0C_KCaM1N2C -> KCaM2N0C + pKCaM1N2C [KpCaM1N2C]
KCaM2N0C + KCaM2N2C <-> KCaM2N0C_KCaM2N2C [>KonCaMKII, <KoffCaMKII]
KCaM2N0C_KCaM2N2C -> pKCaM2N0C + KCaM2N2C [KpCaM2N0C]
KCaM2N0C_KCaM2N2C -> KCaM2N0C + pKCaM2N2C [KpCaM2N2C]
KCaM0N1C + KCaM0N1C <-> KCaM0N1C_KCaM0N1C [>KonCaMKII, <KoffCaMKII]
KCaM0N1C_KCaM0N1C -> pKCaM0N1C + KCaM0N1C [2*KpCaM0N1C]
KCaM0N1C + KCaM1N1C <-> KCaM0N1C_KCaM1N1C [>KonCaMKII, <KoffCaMKII]
KCaM0N1C_KCaM1N1C -> pKCaM0N1C + KCaM1N1C [KpCaM0N1C]
KCaM0N1C_KCaM1N1C -> KCaM0N1C + pKCaM1N1C [KpCaM1N1C]
KCaM0N1C + KCaM2N1C <-> KCaM0N1C_KCaM2N1C [>KonCaMKII, <KoffCaMKII]
KCaM0N1C_KCaM2N1C -> pKCaM0N1C + KCaM2N1C [KpCaM0N1C]
KCaM0N1C_KCaM2N1C -> KCaM0N1C + pKCaM2N1C [KpCaM2N1C]
KCaM0N1C + KCaM0N2C <-> KCaM0N1C_KCaM0N2C [>KonCaMKII, <KoffCaMKII]
KCaM0N1C_KCaM0N2C -> pKCaM0N1C + KCaM0N2C [KpCaM0N1C]
KCaM0N1C_KCaM0N2C -> KCaM0N1C + pKCaM0N2C [KpCaM0N2C]
KCaM0N1C + KCaM1N2C <-> KCaM0N1C_KCaM1N2C [>KonCaMKII, <KoffCaMKII]
KCaM0N1C_KCaM1N2C -> pKCaM0N1C + KCaM1N2C [KpCaM0N1C]
KCaM0N1C_KCaM1N2C -> KCaM0N1C + pKCaM1N2C [KpCaM1N2C]
KCaM0N1C + KCaM2N2C <-> KCaM0N1C_KCaM2N2C [>KonCaMKII, <KoffCaMKII]
KCaM0N1C_KCaM2N2C -> pKCaM0N1C + KCaM2N2C [KpCaM0N1C]
KCaM0N1C_KCaM2N2C -> KCaM0N1C + pKCaM2N2C [KpCaM2N2C]
KCaM1N1C + KCaM1N1C <-> KCaM1N1C_KCaM1N1C [>KonCaMKII, <KoffCaMKII]
KCaM1N1C_KCaM1N1C -> pKCaM1N1C + KCaM1N1C [2*KpCaM1N1C]
KCaM1N1C + KCaM2N1C <-> KCaM1N1C_KCaM2N1C [>KonCaMKII, <KoffCaMKII]
KCaM1N1C_KCaM2N1C -> pKCaM1N1C + KCaM2N1C [KpCaM1N1C]
KCaM1N1C_KCaM2N1C -> KCaM1N1C + pKCaM2N1C [KpCaM2N1C]
KCaM1N1C + KCaM0N2C <-> KCaM1N1C_KCaM0N2C [>KonCaMKII, <KoffCaMKII]
KCaM1N1C_KCaM0N2C -> pKCaM1N1C + KCaM0N2C [KpCaM1N1C]
KCaM1N1C_KCaM0N2C -> KCaM1N1C + pKCaM0N2C [KpCaM0N2C]
KCaM1N1C + KCaM1N2C <-> KCaM1N1C_KCaM1N2C [>KonCaMKII, <KoffCaMKII]
KCaM1N1C_KCaM1N2C -> pKCaM1N1C + KCaM1N2C [KpCaM1N1C]
KCaM1N1C_KCaM1N2C -> KCaM1N1C + pKCaM1N2C [KpCaM1N2C]
KCaM1N1C + KCaM2N2C <-> KCaM1N1C_KCaM2N2C [>KonCaMKII, <KoffCaMKII]
KCaM1N1C_KCaM2N2C -> pKCaM1N1C + KCaM2N2C [KpCaM1N1C]
KCaM1N1C_KCaM2N2C -> KCaM1N1C + pKCaM2N2C [KpCaM2N2C]
KCaM2N1C + KCaM2N1C <-> KCaM2N1C_KCaM2N1C [>KonCaMKII, <KoffCaMKII]
KCaM2N1C_KCaM2N1C -> pKCaM2N1C + KCaM2N1C [2*KpCaM2N1C]
KCaM2N1C + KCaM0N2C <-> KCaM2N1C_KCaM0N2C [>KonCaMKII, <KoffCaMKII]
KCaM2N1C_KCaM0N2C -> pKCaM2N1C + KCaM0N2C [KpCaM2N1C]
KCaM2N1C_KCaM0N2C -> KCaM2N1C + pKCaM0N2C [KpCaM0N2C]
KCaM2N1C + KCaM1N2C <-> KCaM2N1C_KCaM1N2C [>KonCaMKII, <KoffCaMKII]
KCaM2N1C_KCaM1N2C -> pKCaM2N1C + KCaM1N2C [KpCaM2N1C]
KCaM2N1C_KCaM1N2C -> KCaM2N1C + pKCaM1N2C [KpCaM1N2C]
KCaM2N1C + KCaM2N2C <-> KCaM2N1C_KCaM2N2C [>KonCaMKII, <KoffCaMKII]
KCaM2N1C_KCaM2N2C -> pKCaM2N1C + KCaM2N2C [KpCaM2N1C]
KCaM2N1C_KCaM2N2C -> KCaM2N1C + pKCaM2N2C [KpCaM2N2C]
```

```
KCaM0N2C + KCaM0N2C <-> KCaM0N2C_KCaM0N2C [>KonCaMKII, <KoffCaMKII]
KCaM0N2C_KCaM0N2C -> pKCaM0N2C + KCaM0N2C [2*KpCaM0N2C]
KCaM0N2C + KCaM1N2C <-> KCaM0N2C_KCaM1N2C [>KonCaMKII, <KoffCaMKII]
KCaM0N2C_KCaM1N2C -> pKCaM0N2C + KCaM1N2C [KpCaM0N2C]
KCaM0N2C_KCaM1N2C -> KCaM0N2C + pKCaM1N2C [KpCaM1N2C]
KCaM0N2C + KCaM2N2C <-> KCaM0N2C_KCaM2N2C [>KonCaMKII, <KoffCaMKII]
KCaM0N2C_KCaM2N2C -> pKCaM0N2C + KCaM2N2C [KpCaM0N2C]
KCaM0N2C_KCaM2N2C -> KCaM0N2C + pKCaM2N2C [KpCaM2N2C]
KCaM1N2C + KCaM1N2C <-> KCaM1N2C_KCaM1N2C [>KonCaMKII, <KoffCaMKII]
KCaM1N2C_KCaM1N2C -> pKCaM1N2C + KCaM1N2C [2*KpCaM1N2C]
KCaM1N2C + KCaM2N2C <-> KCaM1N2C_KCaM2N2C [>KonCaMKII, <KoffCaMKII]
KCaM1N2C_KCaM2N2C -> pKCaM1N2C + KCaM2N2C [KpCaM1N2C]
KCaM1N2C_KCaM2N2C -> KCaM1N2C + pKCaM2N2C [KpCaM2N2C]
KCaM2N2C + KCaM2N2C <-> KCaM2N2C_KCaM2N2C [>KonCaMKII, <KoffCaMKII]
KCaM2N2C_KCaM2N2C -> pKCaM2N2C + KCaM2N2C [2*KpCaM2N2C]

// pKCaM autophosphorylation of KCaM, 81 combinations, 162 rxns:
pKCaM0N0C + KCaM0N0C <-> pKCaM0N0C_KCaM0N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N0C_KCaM0N0C -> pKCaM0N0C + pKCaM0N0C [KpCaM0N0C]
pKCaM0N0C + KCaM1N0C <-> pKCaM0N0C_KCaM1N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N0C_KCaM1N0C -> pKCaM0N0C + pKCaM1N0C [KpCaM1N0C]
pKCaM0N0C + KCaM2N0C <-> pKCaM0N0C_KCaM2N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N0C_KCaM2N0C -> pKCaM0N0C + pKCaM2N0C [KpCaM2N0C]
pKCaM0N0C + KCaM0N1C <-> pKCaM0N0C_KCaM0N1C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N0C_KCaM0N1C -> pKCaM0N0C + pKCaM0N1C [KpCaM0N1C]
pKCaM0N0C + KCaM1N1C <-> pKCaM0N0C_KCaM1N1C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N0C_KCaM1N1C -> pKCaM0N0C + pKCaM1N1C [KpCaM1N1C]
pKCaM0N0C + KCaM2N1C <-> pKCaM0N0C_KCaM2N1C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N0C_KCaM2N1C -> pKCaM0N0C + pKCaM2N1C [KpCaM2N1C]
pKCaM0N0C + KCaM0N2C <-> pKCaM0N0C_KCaM0N2C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N0C_KCaM0N2C -> pKCaM0N0C + pKCaM0N2C [KpCaM0N2C]
pKCaM0N0C + KCaM1N2C <-> pKCaM0N0C_KCaM1N2C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N0C_KCaM1N2C -> pKCaM0N0C + pKCaM1N2C [KpCaM1N2C]
pKCaM0N0C + KCaM2N2C <-> pKCaM0N0C_KCaM2N2C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N0C_KCaM2N2C -> pKCaM0N0C + pKCaM2N2C [KpCaM2N2C]
pKCaM1N0C + KCaM0N0C <-> pKCaM1N0C_KCaM0N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM1N0C_KCaM0N0C -> pKCaM1N0C + pKCaM0N0C [KpCaM0N0C]
pKCaM1N0C + KCaM1N0C <-> pKCaM1N0C_KCaM1N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM1N0C_KCaM1N0C -> pKCaM1N0C + pKCaM1N0C [KpCaM1N0C]
pKCaM1N0C + KCaM2N0C <-> pKCaM1N0C_KCaM2N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM1N0C_KCaM2N0C -> pKCaM1N0C + pKCaM2N0C [KpCaM2N0C]
pKCaM1N0C + KCaM0N1C <-> pKCaM1N0C_KCaM0N1C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM1N0C_KCaM0N1C -> pKCaM1N0C + pKCaM0N1C [KpCaM0N1C]
pKCaM1N0C + KCaM1N1C <-> pKCaM1N0C_KCaM1N1C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM1N0C_KCaM1N1C -> pKCaM1N0C + pKCaM1N1C [KpCaM1N1C]
pKCaM1N0C + KCaM2N1C <-> pKCaM1N0C_KCaM2N1C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM1N0C_KCaM2N1C -> pKCaM1N0C + pKCaM2N1C [KpCaM2N1C]
pKCaM1N0C + KCaM0N2C <-> pKCaM1N0C_KCaM0N2C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM1N0C_KCaM0N2C -> pKCaM1N0C + pKCaM0N2C [KpCaM0N2C]
```

```
pKCaM1N0C + KCaM1N2C <-> pKCaM1N0C_KCaM1N2C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM1N0C_KCaM1N2C -> pKCaM1N0C + pKCaM1N2C [KpCaM1N2C]
pKCaM1N0C + KCaM2N2C <-> pKCaM1N0C_KCaM2N2C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM1N0C_KCaM2N2C -> pKCaM1N0C + pKCaM2N2C [KpCaM2N2C]
pKCaM2N0C + KCaM0N0C <-> pKCaM2N0C_KCaM0N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM2N0C_KCaM0N0C -> pKCaM2N0C + pKCaM0N0C [KpCaM0N0C]
pKCaM2N0C + KCaM1N0C <-> pKCaM2N0C_KCaM1N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM2N0C_KCaM1N0C -> pKCaM2N0C + pKCaM1N0C [KpCaM1N0C]
pKCaM2N0C + KCaM2N0C <-> pKCaM2N0C_KCaM2N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM2N0C_KCaM2N0C -> pKCaM2N0C + pKCaM2N0C [KpCaM2N0C]
pKCaM2N0C + KCaM0N1C <-> pKCaM2N0C_KCaM0N1C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM2N0C_KCaM0N1C -> pKCaM2N0C + pKCaM0N1C [KpCaM0N1C]
pKCaM2N0C + KCaM1N1C <-> pKCaM2N0C_KCaM1N1C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM2N0C_KCaM1N1C -> pKCaM2N0C + pKCaM1N1C [KpCaM1N1C]
pKCaM2N0C + KCaM2N1C <-> pKCaM2N0C_KCaM2N1C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM2N0C_KCaM2N1C -> pKCaM2N0C + pKCaM2N1C [KpCaM2N1C]
pKCaM2N0C + KCaM0N2C <-> pKCaM2N0C_KCaM0N2C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM2N0C_KCaM0N2C -> pKCaM2N0C + pKCaM0N2C [KpCaM0N2C]
pKCaM2N0C + KCaM1N2C <-> pKCaM2N0C_KCaM1N2C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM2N0C_KCaM1N2C -> pKCaM2N0C + pKCaM1N2C [KpCaM1N2C]
pKCaM2N0C + KCaM2N2C <-> pKCaM2N0C_KCaM2N2C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM2N0C_KCaM2N2C -> pKCaM2N0C + pKCaM2N2C [KpCaM2N2C]
pKCaM0N1C + KCaM0N0C <-> pKCaM0N1C_KCaM0N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N1C_KCaM0N0C -> pKCaM0N1C + pKCaM0N0C [KpCaM0N0C]
pKCaM0N1C + KCaM1N0C <-> pKCaM0N1C_KCaM1N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N1C_KCaM1N0C -> pKCaM0N1C + pKCaM1N0C [KpCaM1N0C]
pKCaM0N1C + KCaM2N0C <-> pKCaM0N1C_KCaM2N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N1C_KCaM2N0C -> pKCaM0N1C + pKCaM2N0C [KpCaM2N0C]
pKCaM0N1C + KCaM0N1C <-> pKCaM0N1C_KCaM0N1C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N1C_KCaM0N1C -> pKCaM0N1C + pKCaM0N1C [KpCaM0N1C]
pKCaM0N1C + KCaM1N1C <-> pKCaM0N1C_KCaM1N1C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N1C_KCaM1N1C -> pKCaM0N1C + pKCaM1N1C [KpCaM1N1C]
pKCaM0N1C + KCaM2N1C <-> pKCaM0N1C_KCaM2N1C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N1C_KCaM2N1C -> pKCaM0N1C + pKCaM2N1C [KpCaM2N1C]
pKCaM0N1C + KCaM0N2C <-> pKCaM0N1C_KCaM0N2C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N1C_KCaM0N2C -> pKCaM0N1C + pKCaM0N2C [KpCaM0N2C]
pKCaM0N1C + KCaM1N2C <-> pKCaM0N1C_KCaM1N2C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N1C_KCaM1N2C -> pKCaM0N1C + pKCaM1N2C [KpCaM1N2C]
pKCaM0N1C + KCaM2N2C <-> pKCaM0N1C_KCaM2N2C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N1C_KCaM2N2C -> pKCaM0N1C + pKCaM2N2C [KpCaM2N2C]
pKCaM1N1C + KCaM0N0C <-> pKCaM1N1C_KCaM0N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM1N1C_KCaM0N0C -> pKCaM1N1C + pKCaM0N0C [KpCaM0N0C]
pKCaM1N1C + KCaM1N0C <-> pKCaM1N1C_KCaM1N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM1N1C_KCaM1N0C -> pKCaM1N1C + pKCaM1N0C [KpCaM1N0C]
pKCaM1N1C + KCaM2N0C <-> pKCaM1N1C_KCaM2N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM1N1C_KCaM2N0C -> pKCaM1N1C + pKCaM2N0C [KpCaM2N0C]
pKCaM1N1C + KCaM0N1C <-> pKCaM1N1C_KCaM0N1C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM1N1C_KCaM0N1C -> pKCaM1N1C + pKCaM0N1C [KpCaM0N1C]
pKCaM1N1C + KCaM1N1C <-> pKCaM1N1C_KCaM1N1C [>KonCaMKIIp, <KoffCaMKIIp]
```

```
pKCaM1N1C_KCaM1N1C -> pKCaM1N1C + pKCaM1N1C [KpCaM1N1C]
pKCaM1N1C + KCaM2N1C <-> pKCaM1N1C_KCaM2N1C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM1N1C_KCaM2N1C -> pKCaM1N1C + pKCaM2N1C [KpCaM2N1C]
pKCaM1N1C + KCaM0N2C <-> pKCaM1N1C_KCaM0N2C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM1N1C_KCaM0N2C -> pKCaM1N1C + pKCaM0N2C [KpCaM0N2C]
pKCaM1N1C + KCaM1N2C <-> pKCaM1N1C_KCaM1N2C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM1N1C_KCaM1N2C -> pKCaM1N1C + pKCaM1N2C [KpCaM1N2C]
pKCaM1N1C + KCaM2N2C <-> pKCaM1N1C_KCaM2N2C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM1N1C_KCaM2N2C -> pKCaM1N1C + pKCaM2N2C [KpCaM2N2C]
pKCaM2N1C + KCaM0N0C <-> pKCaM2N1C_KCaM0N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM2N1C_KCaM0N0C -> pKCaM2N1C + pKCaM0N0C [KpCaM0N0C]
pKCaM2N1C + KCaM1N0C <-> pKCaM2N1C_KCaM1N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM2N1C_KCaM1N0C -> pKCaM2N1C + pKCaM1N0C [KpCaM1N0C]
pKCaM2N1C + KCaM2N0C <-> pKCaM2N1C_KCaM2N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM2N1C_KCaM2N0C -> pKCaM2N1C + pKCaM2N0C [KpCaM2N0C]
pKCaM2N1C + KCaM0N1C <-> pKCaM2N1C_KCaM0N1C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM2N1C_KCaM0N1C -> pKCaM2N1C + pKCaM0N1C [KpCaM0N1C]
pKCaM2N1C + KCaM1N1C <-> pKCaM2N1C_KCaM1N1C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM2N1C_KCaM1N1C -> pKCaM2N1C + pKCaM1N1C [KpCaM1N1C]
pKCaM2N1C + KCaM2N1C <-> pKCaM2N1C_KCaM2N1C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM2N1C_KCaM2N1C -> pKCaM2N1C + pKCaM2N1C [KpCaM2N1C]
pKCaM2N1C + KCaM0N2C <-> pKCaM2N1C_KCaM0N2C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM2N1C_KCaM0N2C -> pKCaM2N1C + pKCaM0N2C [KpCaM0N2C]
pKCaM2N1C + KCaM1N2C <-> pKCaM2N1C_KCaM1N2C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM2N1C_KCaM1N2C -> pKCaM2N1C + pKCaM1N2C [KpCaM1N2C]
pKCaM2N1C + KCaM2N2C <-> pKCaM2N1C_KCaM2N2C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM2N1C_KCaM2N2C -> pKCaM2N1C + pKCaM2N2C [KpCaM2N2C]
pKCaM0N2C + KCaM0N0C <-> pKCaM0N2C_KCaM0N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N2C_KCaM0N0C -> pKCaM0N2C + pKCaM0N0C [KpCaM0N0C]
pKCaM0N2C + KCaM1N0C <-> pKCaM0N2C_KCaM1N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N2C_KCaM1N0C -> pKCaM0N2C + pKCaM1N0C [KpCaM1N0C]
pKCaM0N2C + KCaM2N0C <-> pKCaM0N2C_KCaM2N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N2C_KCaM2N0C -> pKCaM0N2C + pKCaM2N0C [KpCaM2N0C]
pKCaM0N2C + KCaM0N1C <-> pKCaM0N2C_KCaM0N1C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N2C_KCaM0N1C -> pKCaM0N2C + pKCaM0N1C [KpCaM0N1C]
pKCaM0N2C + KCaM1N1C <-> pKCaM0N2C_KCaM1N1C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N2C_KCaM1N1C -> pKCaM0N2C + pKCaM1N1C [KpCaM1N1C]
pKCaM0N2C + KCaM2N1C <-> pKCaM0N2C_KCaM2N1C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N2C_KCaM2N1C -> pKCaM0N2C + pKCaM2N1C [KpCaM2N1C]
pKCaM0N2C + KCaM0N2C <-> pKCaM0N2C_KCaM0N2C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N2C_KCaM0N2C -> pKCaM0N2C + pKCaM0N2C [KpCaM0N2C]
pKCaM0N2C + KCaM1N2C <-> pKCaM0N2C_KCaM1N2C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N2C_KCaM1N2C -> pKCaM0N2C + pKCaM1N2C [KpCaM1N2C]
pKCaM0N2C + KCaM2N2C <-> pKCaM0N2C_KCaM2N2C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM0N2C_KCaM2N2C -> pKCaM0N2C + pKCaM2N2C [KpCaM2N2C]
pKCaM1N2C + KCaM0N0C <-> pKCaM1N2C_KCaM0N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM1N2C_KCaM0N0C -> pKCaM1N2C + pKCaM0N0C [KpCaM0N0C]
pKCaM1N2C + KCaM1N0C <-> pKCaM1N2C_KCaM1N0C [>KonCaMKIIp, <KoffCaMKIIp]
pKCaM1N2C_KCaM1N0C -> pKCaM1N2C + pKCaM1N0C [KpCaM1N0C]
```

```
  pKCaM1N2C + KCaM2N0C <-> pKCaM1N2C_KCaM2N0C [>KonCaMKIIp, <KoffCaMKIIp]
  pKCaM1N2C_KCaM2N0C -> pKCaM1N2C + pKCaM2N0C [KpCaM2N0C]
  pKCaM1N2C + KCaM0N1C <-> pKCaM1N2C_KCaM0N1C [>KonCaMKIIp, <KoffCaMKIIp]
  pKCaM1N2C_KCaM0N1C -> pKCaM1N2C + pKCaM0N1C [KpCaM0N1C]
  pKCaM1N2C + KCaM1N1C <-> pKCaM1N2C_KCaM1N1C [>KonCaMKIIp, <KoffCaMKIIp]
  pKCaM1N2C_KCaM1N1C -> pKCaM1N2C + pKCaM1N1C [KpCaM1N1C]
  pKCaM1N2C + KCaM2N1C <-> pKCaM1N2C_KCaM2N1C [>KonCaMKIIp, <KoffCaMKIIp]
  pKCaM1N2C_KCaM2N1C -> pKCaM1N2C + pKCaM2N1C [KpCaM2N1C]
  pKCaM1N2C + KCaM0N2C <-> pKCaM1N2C_KCaM0N2C [>KonCaMKIIp, <KoffCaMKIIp]
  pKCaM1N2C_KCaM0N2C -> pKCaM1N2C + pKCaM0N2C [KpCaM0N2C]
  pKCaM1N2C + KCaM1N2C <-> pKCaM1N2C_KCaM1N2C [>KonCaMKIIp, <KoffCaMKIIp]
  pKCaM1N2C_KCaM1N2C -> pKCaM1N2C + pKCaM1N2C [KpCaM1N2C]
  pKCaM1N2C + KCaM2N2C <-> pKCaM1N2C_KCaM2N2C [>KonCaMKIIp, <KoffCaMKIIp]
  pKCaM1N2C_KCaM2N2C -> pKCaM1N2C + pKCaM2N2C [KpCaM2N2C]
  pKCaM2N2C + KCaM0N0C <-> pKCaM2N2C_KCaM0N0C [>KonCaMKIIp, <KoffCaMKIIp]
  pKCaM2N2C_KCaM0N0C -> pKCaM2N2C + pKCaM0N0C [KpCaM0N0C]
  pKCaM2N2C + KCaM1N0C <-> pKCaM2N2C_KCaM1N0C [>KonCaMKIIp, <KoffCaMKIIp]
  pKCaM2N2C_KCaM1N0C -> pKCaM2N2C + pKCaM1N0C [KpCaM1N0C]
  pKCaM2N2C + KCaM2N0C <-> pKCaM2N2C_KCaM2N0C [>KonCaMKIIp, <KoffCaMKIIp]
  pKCaM2N2C_KCaM2N0C -> pKCaM2N2C + pKCaM2N0C [KpCaM2N0C]
  pKCaM2N2C + KCaM0N1C <-> pKCaM2N2C_KCaM0N1C [>KonCaMKIIp, <KoffCaMKIIp]
  pKCaM2N2C_KCaM0N1C -> pKCaM2N2C + pKCaM0N1C [KpCaM0N1C]
  pKCaM2N2C + KCaM1N1C <-> pKCaM2N2C_KCaM1N1C [>KonCaMKIIp, <KoffCaMKIIp]
  pKCaM2N2C_KCaM1N1C -> pKCaM2N2C + pKCaM1N1C [KpCaM1N1C]
  pKCaM2N2C + KCaM2N1C <-> pKCaM2N2C_KCaM2N1C [>KonCaMKIIp, <KoffCaMKIIp]
  pKCaM2N2C_KCaM2N1C -> pKCaM2N2C + pKCaM2N1C [KpCaM2N1C]
  pKCaM2N2C + KCaM0N2C <-> pKCaM2N2C_KCaM0N2C [>KonCaMKIIp, <KoffCaMKIIp]
  pKCaM2N2C_KCaM0N2C -> pKCaM2N2C + pKCaM0N2C [KpCaM0N2C]
  pKCaM2N2C + KCaM1N2C <-> pKCaM2N2C_KCaM1N2C [>KonCaMKIIp, <KoffCaMKIIp]
  pKCaM2N2C_KCaM1N2C -> pKCaM2N2C + pKCaM1N2C [KpCaM1N2C]
  pKCaM2N2C + KCaM2N2C <-> pKCaM2N2C_KCaM2N2C [>KonCaMKIIp, <KoffCaMKIIp]
  pKCaM2N2C_KCaM2N2C -> pKCaM2N2C + pKCaM2N2C [KpCaM2N2C]
}
```