StochDynTools

A MATLAB® Toolbox to Compute Moment Dynamics for Stochastic Networks of Bio-Bhemical Reactions

João P. Hespanha

January 21, 2009

Abstract

This document provides a guide to use the `StochDynTools` MATLAB® toolbox [3]. The core of this library is a set of functions that compute the approximate moment dynamics for a network of chemical reactions. Examples are provided to illustrate the use of the library.

ATTENTION: This toolbox is still in a very early stage of development. It has been posted online mostly for the use of people that I collaborate with. However, anyone is welcome to try it. *Please let me know if you have problems or questions about it.* Your help in improving this package will be greatly appreciated!

Contents

Chapter 1

Quick start

The `StochDynTools` toolbox provides functions to compute the approximate moment dynamics for a network of chemical reactions. The approximate moment dynamics are based on moment closure techniques that approximate higher-order moments as static functions of lower-order moments [2, 4, 5]. This section shows by example the use of the `StochDynTools` toolbox.

Networks of chemical reactions are specified by a `.net` file. The syntax of a `.net` file is somewhat self-explanatory. For example, consider the Michaelis-Menten mechanism for enzyme kinetics

$$S + E \underset{D_C}{\overset{K_C}{\rightleftharpoons}} ES \xrightarrow{K_P} E + P$$

where E denotes the (free) enzyme, S the substrate, ES the enzyme-substrate complex, and P the product. This network of three elementary chemical reactions can be described by the following `.net` file:

```
species:
        E  stochastic 2;   % free  enzyme
        ES stochastic 1;   % enzyme-substrate  complex
        S  stochastic 9;   % free  substrate
        P  stochastic 1;   % product
parameters:
        K_C "k_C" = 1;     % rate  constant  for  S + E -> ES
        D_C "d_C" = 20;    % rate  constant  for  ES -> S + E
        K_P "k_P" =.05;    % rate  constant  for  ES -> P + E
reactions:
        % reversible    reaction
        rate = K_C*S*E;   {S,E,ES} > {S-1,E-1,ES+1};  % S + E -> ES
        rate = D_C*ES;    {S,E,ES} > {S+1,E+1,ES-1};  % ES -> S + E
        % product  creation   reaction
        rate = K_P*ES;    {P,E,ES} > {P+1,E+1,ES-1};  % ES -> P + E
```

In this network, the reversible reaction $S + E \rightleftharpoons ES$ typically occurs much more often than the production of P. This means that the total number of substrate molecules $St = S + ES$ and the total number of enzyme molecules $Et = E + ES$ have much slower dynamics that those of the free number of substrate molecules S and the free number of enzymes E. In fact, for the above network Et is constant and therefore its stochasticity can be ignored. Because of this, it is more convenient to keep track of the "slow" variables Et and St, instead of the "fast" variables ES and S. Note that the number of enzyme-substrate complex molecules can be obtained from $ES = Et - E$ and the number of free substrate molecules can be obtained from $S = St - ES = St - Et + E$. This approach was followed in the following `.net` file, in which E is the only "fast" variable.

```
% Filename: enzyme_with_Et_St .che
species:
       E  "e" stochastic   2;% free enzyme                          (fast variable )
       Et "e_t" constant   3;% total enzymes = free +compound   (constant )
       St "s_t" stochastic 10;% total substrate = free +compound (slow variable )
       P  "p"   stochastic 1;% product                             (slow variable )
parameters:
       K_C "k_C" = 1;        % rate constant for S + E -> ES
       D_C "d_C" = 20;       % rate constant for ES -> S + E
       K_P "k_P" =.05;       % rate constant for ES -> P + E
reactions:
       % fast reversible reaction
       rate = K_C*(St-Et+E)*E; {E} > {E-1};            % S + E -> ES
       rate = D_C*(Et-E);     {E} > {E+1};             % ES -> S + E
       % slow product creation reaction
       rate = K_P*(Et-E);     {St,P,E} > {St-1,P+1,E+1}; % ES -> P + E
```

The `readNet()` command of `StochDynTools` can be used to read the `enzyme_with_Et_St.net` file above:

```
net=readNet('enzyme_with_Et_St.net');
```

The following `StochDynTools` command then computes the first-order approximate moment dynamics, which is roughly the *deterministic chemical rate* equation[1]:

```
mdyn1=closureDynamics(net,1,'derMatch');
```

One could produce *LaTeX code* for these approximate moment dynamics using the following commands

```
dotMu=expand(mdyn1.approxDotMu.sym);
% dotMu=expand(subsParameters(net, dotMu));
latexmacros=mylatex()
latexMu=mylatex(mdyn1.Mu.sym,mdyn1.texrules)
latexDotMu=mylatex(dotMu,sym,mdyn1.texrules)
```

where, if one were to uncomment the second line, the parameters would be replaced by their numerical values. The LaTeX code in the variables `latexMu` and `latexDotMu` requires the macros in the variable `latexmacros`. Copying the content of `latexMu` and `latexDotMu` into a LaTeX equation environment one obtains

$$\frac{d}{dt}\begin{bmatrix}\mathrm{E}[e]\\ \mathrm{E}[s_t]\\ \mathrm{E}[p]\end{bmatrix} = \begin{bmatrix}-\mathrm{E}[e]d_C - k_P\,\mathrm{E}[e] + \mathrm{E}[e]k_C e_t + d_C e_t + k_P e_t - k_C\mathrm{E}[e]^2 - k_C\,\mathrm{E}[e]\,\mathrm{E}[s_t]\\ k_P\,\mathrm{E}[e] - k_P e_t\\ -k_P\,\mathrm{E}[e] + k_P e_t\end{bmatrix}$$

Once can compute the *equilibrium point* of the system above using the following MATLAB® commands

```
muEq=momentEquilibrium(net,mdyn1);
```

The following `StochDynTools` command then computes the *second-order approximate moment dynamics* using derivative matching (cf. Appendix 2.1) and produces an m-file `fun.m` that computes the derivatives of the uncentered moments of order up to 2:

```
mdyn2=closureDynamics(net,2,'derMatch','fun');
```

The following MATLAB® commands solve the approximate moment dynamics and plot the evolution of the *uncentered moments*:

[1] The main difference being that reactions of the form $A+A \to B$ gives rise to terms of the form $\pm kA(A-1)$, instead of $\pm kA^2$.

```
Tmax=300;                              % maximum simulation time
[t,mu]=ode23s(@(t,x)fun(x),[0,Tmax],mdyn2.Mu.x0);
plot(t,mu)
legend(fun(),'location','best','interpreter','none')
```

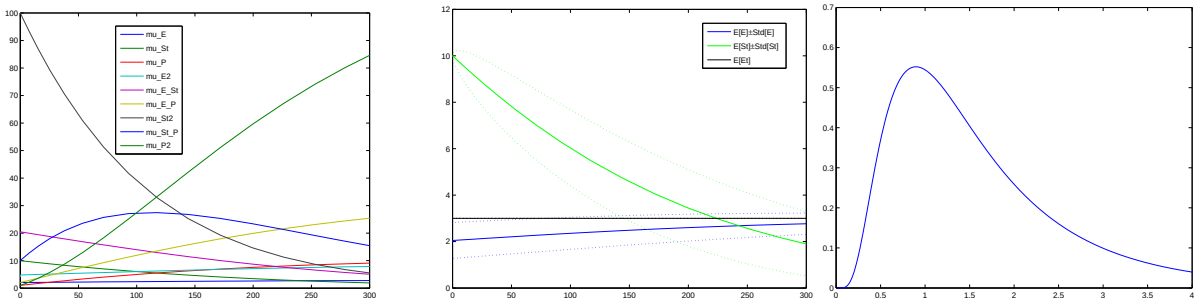
One could instead plot the evolution of the *centered moments* of `E`, `St`, and `Et` using the following `StochDynTools` command

```
h=plotCMoments(net,mdyn2,t,mu,'E','b-','St','g-','Et','k-');
legend(h(1:3:end),'E[E]\pm{}Std[E]','E[St]\pm{}Std[St]','E[Et]','location','best');
```

One could also plot the *distribution* of `St` using the following `StochDynTools` command

```
x=0:.01:4;
pdf=getDistribution(net,mdyn2,mu(end,:),'St',x,'lognormal');
plot(x,pdf)
```

The outputs of the previous plot commands are shown in Figure 1.1.



(a) uncentered moments (b) centered moments (c) distribution of St

Figure 1.1: Output of the plot commands in the example in Section 1.

The precise syntax of the `StochDynTools` commands above is explained in the subsequent sections.

Chapter 2

Moment Closure Methods

Before describing the `StochDynTools` toolbox, we review moment closure and the methods used in this software to achieve it.

Consider a set of chemical species $X_1, X_2, \ldots, X_n$ involved in a set of chemical reactions and let us denote by $x := (x_1, x_2, \ldots, x_n)$ a vector containing their molecule counts. Given a vector of integers $m := (m_1, m_2, \ldots, m_n)$, we use the notation $\mu^{(m)}$ to denote the following uncentered moment of x:

$$\mu^{(m)} := \mathrm{E}[x_1^{m_1} x_2^{m_2} \cdots x_n^{m_n}].$$

Such moment if said to be of order $\sum_i m_i$. With n species there are exactly n first order moments $\mathrm{E}[x_i]$, $\forall i \in \{1, 2, \ldots, n\}$, which are just the means; $n(n-1)/2$ second order moments $\mathrm{E}[x_i^2]$, $\forall i$ and $\mathrm{E}[x_i x_j]$, $\forall i \neq j$, which can be used to compute variances and covariance; $n(n-1)(n-2)/6$ third order moments; and so on.

It was show in [5], that if we construct a vector μ containing all the uncentered moments of x up to some order k, the evolution of μ is determined by a differential equation of the form

$$\dot{\mu} = A\mu + B\bar{\mu}, \quad \mu \in \mathbb{R}^K, \ \bar{\mu} \in \mathbb{R}^{\bar{K}} \tag{2.1}$$

where A and B are appropriately defined matrices and $\bar{\mu}$ is a vector containing moments of order larger[1] than k. The equation (2.1) is exact and we call it the *(exact) k-order moment dynamics* and the integer k is called the *order of truncation*. Note that the dimension K of (2.1) is always larger than k since there are many moments of each order. In fact, in general K is of order n^k.

When all chemical reactions have only one reactant, the term $B\bar{\mu}$ does not appear in (2.1) and we say that the exact moment dynamics are *closed*. However, when at least one chemical reaction has 2 or more reactants, then the term $B\bar{\mu}$ appears and we say that the moment dynamics are *open* since (2.1) depends on the moments in $\bar{\mu}$, which are not part of the state μ. When all chemical reactions are elementary (i.e., with at most 2 reactants), then all moments in $\bar{\mu}$ are exactly of order $k+1$.

Moment closure is a procedure by which one approximates the exact (but open) moment dynamics (2.1) by an approximate (but now closed) equation of the form

$$\dot{\nu} = A\nu + B\varphi(\nu), \quad \nu \in \mathbb{R}^K \tag{2.2}$$

where $\varphi(\nu)$ is a column vector that approximates the moments in $\bar{\mu}$. The function $\varphi(\nu)$ is called the moment closure function and (2.2) is called the *approximate kth order moment dynamics*. The goal of any moment closure method is to construct $\varphi(\nu)$ so that the solution ν to (2.2) is close to the solution μ to (2.1).

[1] When one does not include in μ the zero-order moment $\mu^{(0)} = 1$, this term will appear in $\bar{\mu}$.

Table 2.1: Which moment closure to use?

	distributions have low variability (i.e., low standard deviations when compared to the mean) or are fairly symmetric	distributions have large standard deviations when compared to the mean, but populations do not become zero with high probability	populations can become zero with high probability
accuracy	zero cumulant closure	derivative matching closure	no good solution (yet)
simple dynamics	quasi-deterministic closure or Van Kampen's linear noise approximation	derivative matching closure (but will not be very simple)	no good solution (yet)

Some moment closure methods approximate the exact moments dynamics (2.1) by a closed equation of larger order, such as in

$$\dot{\phi} = \psi(\phi), \quad \phi \in \mathbb{R}^N, \tag{2.3a}$$

$$\dot{\nu} = A\nu + B\varphi(\phi, \nu), \quad \nu \in \mathbb{R}^K, \tag{2.3b}$$

where one now approximates $\bar{\mu}$ by the function $\varphi(\phi, \nu)$ that is allowed to also depend on the state ϕ of an additional dynamic system. Often $\varphi(\phi, \nu)$ can be made linear in ν. In this case, once ϕ reaches a steady state, the ν dynamics became linear and time-invariant.

There are three main approaches to construct the moment closure function $\varphi(\cdot)$:

1. *Matching-based methods* directly attempt to match the solutions to (2.1) and (2.2) [or (2.3)].
2. *Distribution-based methods* construct $\varphi(\cdot)$ by making "reasonable" assumptions on the statistical distribution of the molecule counts vector x.
3. *Large volume methods* construct $\varphi(\cdot)$ by assuming that reactions take place on a large volume.

It is important to emphasize that this classification is about methods to *construct* moment closure. It turns out that sometimes different methods lead to the same moment closure function $\varphi(\cdot)$.

In the remainder of this section we discuss several methods to construct the moment closure function. We shall see that the choice of which method to use depends on the type of system (e.g., how population means compare with standard deviations for the system considered) and also on the primary goal in constructing the approximate moment dynamics (e.g., how important is accuracy versus simplicity of the equations). Table 2.1 summarizes some rules of thumb on the choice of which approximation to use.

2.1 Derivative Matching

Derivative matching is a matching-based method for moment closure described in [5]. It uses moment closure functions $\varphi(\cdot)$ in (2.2) whose entries are *separable*, i.e., of the form

$$\nu_1^{\gamma_1}\nu_2^{\gamma_2}\cdots\nu_n^{\gamma_n}.$$

The coefficients $\gamma_i \in \mathbb{R}$ are then computed to make the relative error

$$\frac{\left\| \frac{d^\ell \nu}{dt^\ell} - \frac{d^\ell \mu}{dt^\ell} \right\|}{\left\| \frac{d^\ell \mu}{dt^\ell} \right\|}$$

as small as possible for molecule counts larger than one. Somewhat surprisingly, this minimization leads to explicit formulas for the moment closure functions $\varphi(\cdot)$ that do not depend on the reaction parameters [5].

2.2 Zero Cumulants

Zero cumulants is a distribution-based method for moment closure that finds the kth order moment closure function $\varphi(\cdot)$ in (2.2) by assuming that all multi-variable cumulants of the population x with order larger than k are negligible. This makes the distribution of x "as close as possible" to a multi-variable Gaussian distribution, which has all cumulants or order higher than two equal to zero.

To construct zero cumulant closures, one uses the fact that the cumulant $\kappa^{(m)}$ can be expressed as

$$\kappa^{(m)} = \mu^{(m)} + \sum_{\sum \bar{m}_i < \sum m_i} \alpha_{\bar{m}} \mu^{(\bar{m})}, \tag{2.4}$$

where the summation is over moments $\mu^{(\bar{m})}$ of order strictly smaller than $\sum m_i$ and the $\alpha_{\bar{m}}$ are appropriately selected constants (c.f. Appendix A.3). This shows that the cumulant $\kappa^{(m)}$ depends only on the moment $\mu^{(m)}$ and lower-order moments $\mu^{(\bar{m})}$, so by setting $\kappa^{(m)} = 0$ one obtains an expression for $\mu^{(m)}$ as a function of lower-order moments.

The procedure to compute the *zero-cumulants* moment closure function $\varphi(\cdot)$ consists of setting to zero all cumulants corresponding to the moments that do not appear in μ and then solving the equations (2.4) for the moments in $\bar{\mu}$.

2.3 Low Dispersion

Low dispersion is a distribution-based method for moment closure that finds the moment closure function $\varphi(\cdot)$ in (2.2) by assuming that the distributions of the populations are tightly clustered around their means, with standard deviations much smaller than the means. Specifically, for the kth order moment closure one assumes that the normalized centered moments of order larger than k are much smaller than one. We recall that given a vector of integers $m := (m_1, m_2, \ldots, m_n)$, the corresponding *normalized centered moment* is defined by

$$\eta^{(m)} := \mathrm{E}\left[\left(\frac{x_1 - \mathrm{E}[x_1]}{\mathrm{E}[x_1]}\right)^{m_1}\left(\frac{x_2 - \mathrm{E}[x_2]}{\mathrm{E}[x_2]}\right)^{m_2}\cdots\left(\frac{x_n - \mathrm{E}[x_n]}{\mathrm{E}[x_n]}\right)^{m_n}\right].$$

Such moment if said to be of order $\sum_i m_i$. For fairly symmetric distributions the odd-order moments can be quite small and therefore this technique is especially useful for even-order moment closures for which the odd-order higher moments can be safely neglected.

To construct low dispersion closures, one uses the fact that an uncentered moment $\mu^{(m)}$ can be expressed in terms of the normalized centered moment as follows

$$\mu^{(m)} = \mathrm{E}[x_1]^{m_1}\,\mathrm{E}[x_2]^{m_2}\cdots\mathrm{E}[x_n]^{m_n}\left(1 + \eta^{(m)} + \sum_{2 \le \sum \bar{m}_i < \sum m_i} \beta_{\bar{m}}\,\eta^{(\bar{m})}\right),$$

where the summation is over moments $\eta^{(\bar{m})}$ of order two or larger and strictly smaller than $\sum m_i$, and the $\beta_{\bar{m}}$ are appropriately selected nonnegative constants (cf. Appendix A.4). When a particular normalized centered moment $\eta^{(m)}$ is much smaller than one, we have that

$$\mu^{(m)} \approx \mathrm{E}[x_1]^{m_1}\,\mathrm{E}[x_2]^{m_2}\cdots\mathrm{E}[x_n]^{m_n}\left(1 + \sum_{2 \le \sum \bar{m}_i < \sum m_i} \beta_{\bar{m}}\,\eta^{(\bar{m})}\right), \tag{2.5}$$

which allows one to express the uncentered moment $\mu^{(m)}$ solely in terms of normalized centered moments $\eta^{(\bar{m})}$ of order strictly smaller than $\sum m_i$. On the other hand, we can express all these normalized centered moments as linear combinations of the uncentered moments of order strictly smaller than $\sum m_i$ as follows

$$\eta^{(\bar{m})} = \frac{\mu^{(\bar{m})}}{\mathrm{E}[x_1]^{\bar{m}_1}\,\mathrm{E}[x_2]^{\bar{m}_2}\cdots\mathrm{E}[x_n]^{\bar{m}_n}} + \sum_{\sum \tilde{m}_i < \sum m_i} \gamma_{\tilde{m}}\,\frac{\mu^{(\tilde{m})}}{\mathrm{E}[x_1]^{\tilde{m}_1}\,\mathrm{E}[x_2]^{\tilde{m}_2}\cdots\mathrm{E}[x_n]^{\tilde{m}_n}}, \tag{2.6}$$

where the summation is over uncentered moments $\mu^{(\tilde{m})}$ of order strictly smaller than $\sum \bar{m}_i$ and the $\gamma_{\tilde{m}}$ are appropriately selected constants (cf. Appendix A.4).

The procedure to compute the *low dispersions* moment closure function $\varphi(\cdot)$ in (2.2) thus consists of using (2.5) and (2.6) to approximate any moment that does not appear in μ as a linear combination of the moments in μ. Note however that the coefficients of these linear combinations will depend on monomials of the form $\mathrm{E}[x_1]^{\hat{m}_1}\mathrm{E}[x_2]^{\hat{m}_2}\cdots\mathrm{E}[x_n]^{\hat{m}_n}$, with all the $\hat{m}_i \geq 0$ and therefore the moment closure function $\phi(\cdot)$ will be polynomial but nonlinear on μ.

Relationship with zero-cumulants closure For second order moment closure ($k = 2$) one sets to zero 3th-order normalized centered moments, which is equivalent to setting to zero the 3th-order cumulants. Therefore for 2nd-order closures, zero cumulant and low dispersion coincide.

2.4 Quasi-Deterministic

Quasi-deterministic is a distribution-based method for moment closure that finds the moment closure function $\varphi(\cdot)$ in (2.3) by assuming that the distributions of the populations are tightly clustered around the solution ϕ to the deterministic dynamics

$$\dot{\phi} = A_{\mathrm{det}}\phi + B_{\mathrm{det}}\psi(\phi), \qquad \phi := (\phi_1, \phi_2, \ldots, \phi_n) \in \mathbb{R}^n, \tag{2.7}$$

which are obtained by assuming that each $\phi_i := x_i$ is deterministic and therefore

$$\mathrm{E}[\phi_i\phi_j] = \mathrm{E}[\phi_i]\,\mathrm{E}[\phi_j] = \phi_i\phi_j.$$

Specifically, for the kth order moment closure one assumes that the quasi-deterministic normalized centered moments of order larger than k are much smaller than one. Given a vector of integers $m := (m_1, m_2, \ldots, m_n)$, the corresponding *quasi-deterministic normalized centered moment* is defined by

$$\hat{\eta}^{(m)} := \mathrm{E}\left[\left(\frac{x_1-\phi_1}{\phi_1}\right)^{m_1}\left(\frac{x_2-\phi_2}{\phi_2}\right)^{m_2}\cdots\left(\frac{x_n-\phi_n}{\phi_n}\right)^{m_n}\right].$$

Such moment if said to be of order $\sum_i m_i$.

To construct quasi-deterministic closures, one uses the fact that an uncentered moment $\mu^{(m)}$ can be expressed in terms of the quasi-deterministic normalized centered moment as follows

$$\mu^{(m)} = \phi_1^{m_1}\phi_2^{m_2}\cdots\phi_n^{m_n}\left(1 + \hat{\eta}^{(m)} + \sum_{1\leq\sum\bar{m}_i<\sum m_i}\beta_{\bar{m}}\,\hat{\eta}^{(\bar{m})}\right),$$

where the summation is over moments $\eta^{(\bar{m})}$ of order one or larger and strictly smaller than $\sum m_i$, and the $\beta_{\bar{m}}$ are appropriately selected nonnegative constants (cf. Appendix A.5). When a particular quasi-deterministic normalized centered moment $\hat{\eta}^{(m)}$ is much smaller than one, we have that

$$\mu^{(m)} \approx \phi_1^{m_1}\phi_2^{m_2}\cdots\phi_n^{m_n}\left(1 + \sum_{2\leq\sum\bar{m}_i<\sum m_i}\beta_{\bar{m}}\,\hat{\eta}^{(\bar{m})}\right), \tag{2.8}$$

which allows one to express the uncentered moment $\mu^{(m)}$ solely in terms of quasi-deterministic normalized centered moments $\hat{\eta}^{(\bar{m})}$ of order strictly smaller than $\sum m_i$. On the other hand, we can express all these quasi-deterministic normalized centered moments as linear combinations of the uncentered moments of order strictly smaller than $\sum m_i$ as follows

$$\hat{\eta}^{(\bar{m})} = \frac{\mu^{(\bar{m})}}{\phi_1^{m_1}\phi_2^{m_2}\cdots\phi_n^{m_n}} + \sum_{\sum\tilde{m}_i<\sum m_i}\gamma_{\tilde{m}}\,\frac{\mu^{(\tilde{m})}}{\phi_1^{\tilde{m}_1}\phi_2^{\tilde{m}_2}\cdots\phi_n^{\tilde{m}_n}}, \tag{2.9}$$

where the summation is over uncentered moments $\mu^{(\tilde{m})}$ of order strictly smaller than $\sum \bar{m}_i$ and the $\gamma_{\tilde{m}}$ are appropriately selected constants (cf. Appendix A.5).

The procedure to compute the *quasi-deterministic* moment closure function $\varphi(\cdot)$ in (2.3) thus consists of using (2.5) and (2.6) to approximate any moment that does not appear in μ as a linear combination of the moments in μ. The coefficients of these linear combinations will depend on monomials of the form $\phi_1^{\hat{m}_1}\phi_2^{\hat{m}_2}\cdots\phi_n^{\hat{m}_n}$, with all the $\hat{m}_i \geq 0$ and therefore the moment closure function will be linear on μ for a fixed ϕ. This means that the approximate dynamics in (2.3) are of the form

$$\dot{\phi} = A_{\text{det}}\phi + B_{\text{det}}\psi(\phi), \quad \phi \in \mathbb{R}^n, \qquad\qquad \dot{\nu} = A_\nu(\phi)\nu + c_\nu(\phi), \quad \nu \in \mathbb{R}^K, \tag{2.10}$$

and, when ϕ reaches a steady state value, the ν dynamics become linear.

Relationship with low dispersion closure In general the normalized centered moment are smaller than their quasi-deterministic version and therefore whenever quasi-deterministic moment closure provides a good approximation, one should expect low-dispersion moment closure to do at least as well. However, quasi-deterministic moment closure has the advantage that it results in moment dynamics that are "almost" linear and therefore generally easier to analyze.

2.5 Van Kampen's Linear Noise Approximation

Van Kampen's *Linear Noise Approximation* is developed in [6, Chapter X] and can be applied when the matrices A, B in (2.1) depend on some parameter V that can be assumed large, i.e., when we have

$$\dot{\mu} = A(V)\mu + B(V)\bar{\mu}, \quad \mu \in \mathbb{R}^K,$$

with V large. This form of moment closure results in a system of the form (2.3) and is exact in the limit as $V \to \infty$. Typically, V is the volume on which the chemical reactions take place.

To construct (2.3), one starts by choosing $\bar{\phi}$ to satisfy the *deterministic large-volume dynamics*

$$\dot{\bar{\phi}} = A_{\text{det}}\bar{\phi} + B_{\text{det}}\psi(\bar{\phi}), \qquad\qquad \bar{\phi} := (\bar{\phi}_1, \bar{\phi}_2, \ldots, \bar{\phi}_n) \in \mathbb{R}^n \tag{2.11}$$

which are obtained by assuming that each $\bar{\phi}_i := x_i/V$ is deterministic and therefore

$$\mathrm{E}[\bar{\phi}_i\bar{\phi}_j] = \mathrm{E}[\bar{\phi}_i]\,\mathrm{E}[\bar{\phi}_j] = \bar{\phi}_i\bar{\phi}_j$$

and also by making $V \to \infty$.

Regarding the vector $\bar{\phi}$ in (2.11) as a deterministic approximation to the stochastic vector x/V, motivates defining the following stochastic perturbation vector $\chi := (\chi_1, \chi_2, \ldots, \chi_n)$, with

$$\chi_i := \frac{x_i - V\bar{\phi}_i}{V^{\frac{1}{2}}} \quad \Leftrightarrow \quad x_i = V\bar{\phi}_i + V^{\frac{1}{2}}\chi_i, \tag{2.12}$$

where the normalization by $V^{\frac{1}{2}}$ will be needed to keep the moments of χ bounded as $V \to \infty$. Given a vector of integers $m := (m_1, m_2, \ldots, m_n)$, we use the notation $\xi^{(m)}$ to denote the following uncentered moment of χ:

$$\xi^{(m)} := \mathrm{E}[\chi_1^{m_1}\chi_2^{m_2}\cdots\chi_n^{m_n}].$$

The moments of x and χ are related by

$$\mu^{(m)} = \mathrm{E}[(V\bar{\phi}_1 + V^{\frac{1}{2}}\chi_1)^{m_1}\cdots(V\bar{\phi}_n + V^{\frac{1}{2}}\chi_n)^{m_n}] = V^{\sum_i m_i}\sum_{\sum\bar{m}_i \leq \sum m_i} \frac{\alpha_{\bar{m}}}{V^{\frac{\bar{m}}{2}}}\xi^{(\bar{m})} = V^{\sum_i m_i}\left(\bar{\phi}^{(m)} + \cdots + \frac{1}{V^{\frac{\sum_i m_i}{2}}}\xi^{(m)}\right), \tag{2.13}$$

where the summation is over moments $\xi^{(\bar{m})}$ of order up to $\sum m_i$ and the $\alpha_{\bar{m}}$ are appropriately selected constants.

Computing the (exact) moment dynamics for ξ, one obtains[2]

$$\dot{\xi} = A_\xi(V,\bar{\phi})\xi + B_\xi(V,\bar{\phi})\bar{\xi}, \quad \xi \in \mathbb{R}^K, \tag{2.14}$$

where ξ and $\bar{\xi}$ contain the moments of χ corresponding to the moments of x in μ and $\bar{\mu}$, respectively. For elementary reactions with reaction rates that depend on the volume as follows:

$$\emptyset \xrightarrow{\text{rate} = cV} * \qquad X \xrightarrow{\text{rate} = cX} * \qquad 2X \xrightarrow{\text{rate} = c\frac{X(X-1)}{V}} * \qquad X + Y \xrightarrow{\text{rate} = c\frac{XY}{V}} *$$

the open system (2.14) converges as $V \to \infty$ to a closed system of the form

$$\dot{\xi} = A_\xi(V,\bar{\phi})\xi + B_\xi(V,\bar{\phi})\bar{\xi} \xrightarrow{V\to\infty} A_\xi(\infty,\bar{\phi})\xi, \quad \xi \in \mathbb{R}^K. \tag{2.15}$$

Since the moments μ and ξ are related through (2.13), one can use (2.15) to obtain a closed equation for μ as in (2.3). Moreover, this equation will be linear in μ, leading to approximate dynamics similar to (2.10).

One could have done the derivation above in a more constructive way without pre-specifying the dynamics for $\bar{\phi}$ by (2.11). In this case, since we have no expression to replace for $\dot{\bar{\phi}}$, we would have obtained for the exact dynamics for ξ an expression of the form

$$\dot{\xi} = A_\xi(V,\bar{\phi})\xi + B_\xi(V,\bar{\phi})\bar{\xi} + C_\xi(V,\bar{\phi})\dot{\bar{\phi}}, \quad \xi \in \mathbb{R}^K.$$

If then tried to make $V \to \infty$, we would observe that to obtain a finite right-hand side we would need $\dot{\bar{\phi}}$ to satisfy precisely (2.11).

Relationship with quasi-deterministic closure The deterministic equations (2.7) and (2.11) differ by two facts:

1. the state in (2.11) was normalized through a division by the volume,
2. in (2.11) we took the limit as $V \to \infty$.

For elementary reactions with molecule counts much larger than one, taking the limit as $V \to \infty$ has almost no effect and we essentially have $\phi = V\bar{\phi}$. In this case,

$$\xi^{(m)} = \mathrm{E}\left[\left(\frac{x_1-\phi_1}{V^{\frac{1}{2}}}\right)^{m_1}\left(\frac{x_2-\phi_2}{V^{\frac{1}{2}}}\right)^{m_2}\cdots\left(\frac{x_n-\phi_n}{V^{\frac{1}{2}}}\right)^{m_n}\right] = \frac{\phi_1^{m_1}\phi_2^{m_2}\cdots\phi_n^{m_n}}{V^{\frac{\sum_i m_i}{2}}}\hat{\eta}^{(m)}.$$

So setting a quasi-deterministic moments to zero $\hat{\eta}^{(m)}$ is equivalent to setting to zero the corresponding uncentered moment $\xi^{(m)}$ of χ. This means that we can view the quasi-deterministic closure as taking the Van Kampen equations (2.11) and (2.15) and simply setting $\bar{\xi}$ in (2.15) to zero, without ignoring other terms that would also disappear as $V \to \infty$. Since we are keeping more terms of the exact equations, with quasi-deterministic closure one often obtains more accurate results then with Van Kampen's linear noise approximation.

[2] Assuming that ξ remain bounded as $V \to \infty$, for any moment $\mu^{(m)}$ in $\bar{\mu}$, we have that for sufficiently large V,

$$\mu^{(m)} \approx V^{\sum_i m_i} \sum_{\bar{m}\le k} \frac{\alpha_{\bar{m}}}{V^{\frac{m}{2}}}\xi^{(\bar{m})}$$

where we only kept in the summation the terms order up to k that are in μ. Since all the $\xi^{(\bar{m})}$, $\bar{m} \le k$ can be expressed as linear combinations of the moments in μ, we have obtained an expression for the moments in $\bar{\mu}$ as a linear combination of the moments in μ (generally with time varying coefficient that depend on $\bar{\phi}$). This approach only make $V \to \infty$ where this is absolutely needed and therefore could lead to better results than taking the limit in (2.15). However, in the current `StochDynTools` implementation we followed closely [6, Chapter X] and took the limit in (2.15).

2.6 Example

In this section we present the different moment closure dynamics for the network of chemical reactions considered in [6, p. 263]. This network of three elementary chemical reactions is described by the following `.net` file:

```
species:
       X stochastic;    % number of  X molecules
       Y stochastic;    % number of  Y molecules
parameters:
       V         = 20;  % volume
       phiA "\phi_A" = 5;  % concentration    of  A (fixed )
       al "\alpha"    = 10;
       be "\beta"    = 20;
       ga "\gamma"    = 30;
reactions:
       rate = al*phiA*V;    {X} > {X+1};         %   A -> X
       rate = ga*X*(X-1)/V; {X,Y} > {X-2,Y+1};  % 2 X -> Y
       rate = be*Y;         {Y} > {Y-1};         %   Y -> B
```

The exact 2nd-order moment dynamics for this system are given by

$$\frac{d}{dt}\begin{bmatrix}\mathrm{E}[X]\\ \mathrm{E}[Y]\\ \mathrm{E}[X^2]\\ \mathrm{E}[XY]\\ \mathrm{E}[Y^2]\end{bmatrix} = \begin{bmatrix} 2\frac{\gamma}{V} & 0 & -2\frac{\gamma}{V} & 0 & 0\\ -\frac{\gamma}{V} & -\beta & \frac{\gamma}{V} & 0 & 0\\ 2\alpha\phi_A V - 4\frac{\gamma}{V} & 0 & 8\frac{\gamma}{V} & 0 & 0\\ 2\frac{\gamma}{V} & \alpha\phi_A V & -3\frac{\gamma}{V} & -\beta+2\frac{\gamma}{V} & 0\\ -\frac{\gamma}{V} & \beta & \frac{\gamma}{V} & -2\frac{\gamma}{V} & -2\beta \end{bmatrix}\begin{bmatrix}\mathrm{E}[X]\\ \mathrm{E}[Y]\\ \mathrm{E}[X^2]\\ \mathrm{E}[XY]\\ \mathrm{E}[Y^2]\end{bmatrix} + \begin{bmatrix}\alpha\phi_A V & 0 & 0\\ 0 & 0 & 0\\ \alpha\phi_A V & -4\frac{\gamma}{V} & 0\\ 0 & \frac{\gamma}{V} & -2\frac{\gamma}{V}\\ 0 & 0 & 2\frac{\gamma}{V}\end{bmatrix}\begin{bmatrix}1\\ \mathrm{E}[X^3]\\ \mathrm{E}[X^2Y]\end{bmatrix}.$$

We now list the approximate 2nd-order moment dynamics for this system obtained using the different methods:

1. Derivative matching:

$$\begin{bmatrix}\mathrm{E}[X^3]\\ \mathrm{E}[X^2Y]\end{bmatrix} \approx \begin{bmatrix}\frac{\mathrm{E}[X^2]^3}{\mathrm{E}[X]^3}\\ \frac{\mathrm{E}[X^2]\mathrm{E}[XY]^2}{\mathrm{E}[X]^2\mathrm{E}[Y]}\end{bmatrix}$$

2. Zero cumulants and low dispersion:

$$\begin{bmatrix}\mathrm{E}[X^3]\\ \mathrm{E}[X^2Y]\end{bmatrix} \approx \begin{bmatrix}3\mathrm{E}[X^2]\mathrm{E}[X]-2\mathrm{E}[X]^3\\ \mathrm{E}[X^2]\mathrm{E}[Y]+2\mathrm{E}[X]\mathrm{E}[XY]-2\mathrm{E}[X]^2\mathrm{E}[Y]\end{bmatrix}$$

3. Quasi deterministic:

$$\frac{d}{dt}\begin{bmatrix}\phi_X\\ \phi_Y\end{bmatrix} = \begin{bmatrix}2\frac{\gamma\phi_X}{V}-2\frac{\gamma{\phi_X}^2}{V}+\alpha\phi_A V\\ -\frac{\gamma\phi_X}{V}-\beta\phi_Y+\frac{\gamma{\phi_X}^2}{V}\end{bmatrix}$$
$$\begin{bmatrix}\mathrm{E}[X^3]\\ \mathrm{E}[X^2Y]\end{bmatrix} \approx \begin{bmatrix}{\phi_X}^3-3{\phi_X}^2\mathrm{E}[X]+3\phi_X\mathrm{E}[X^2]\\ {\phi_X}^2\phi_Y+\phi_Y\mathrm{E}[X^2]-2\phi_X\phi_Y\mathrm{E}[X]-{\phi_X}^2\mathrm{E}[Y]+2\phi_X\mathrm{E}[XY]\end{bmatrix}$$

leading to

$$\frac{d}{dt}\begin{bmatrix}\mathrm{E}[X]\\ \mathrm{E}[Y]\\ \mathrm{E}[X^2]\\ \mathrm{E}[XY]\\ \mathrm{E}[Y^2]\end{bmatrix} \approx \begin{bmatrix} 2\frac{\gamma}{V} & 0 & -2\frac{\gamma}{V} & 0 & 0\\ -\frac{\gamma}{V} & -\beta & \frac{\gamma}{V} & 0 & 0\\ 2\alpha\phi_A V-4\frac{\gamma}{V}+12\frac{\gamma{\phi_X}^2}{V} & 0 & 8\frac{\gamma}{V}-12\frac{\gamma\phi_X}{V} & 0 & 0\\ 2\frac{\gamma}{V}-3\frac{\gamma{\phi_X}^2}{V}+4\frac{\gamma\phi_X\phi_Y}{V} & \alpha\phi_A V+2\frac{\gamma{\phi_X}^2}{V} & -3\frac{\gamma}{V}+3\frac{\gamma\phi_X}{V}-2\frac{\gamma\phi_Y}{V} & -\beta+2\frac{\gamma}{V}-4\frac{\gamma\phi_X}{V} & 0\\ -\frac{\gamma}{V}-4\frac{\gamma\phi_X\phi_Y}{V} & \beta-2\frac{\gamma{\phi_X}^2}{V} & \frac{\gamma}{V}+2\frac{\gamma\phi_Y}{V} & -2\frac{\gamma}{V}+4\frac{\gamma\phi_X}{V} & -2\beta\end{bmatrix}\begin{bmatrix}\mathrm{E}[X]\\ \mathrm{E}[Y]\\ \mathrm{E}[X^2]\\ \mathrm{E}[XY]\\ \mathrm{E}[Y^2]\end{bmatrix} + \begin{bmatrix}\alpha\phi_A V\\ 0\\ \alpha\phi_A V-4\frac{\gamma{\phi_X}^3}{V}\\ \frac{\gamma{\phi_X}^3}{V}-2\frac{\gamma{\phi_X}^2\phi_Y}{V}\\ 2\frac{\gamma{\phi_X}^2\phi_Y}{V}\end{bmatrix}$$

4. Van Kampen's linear noise approximation:

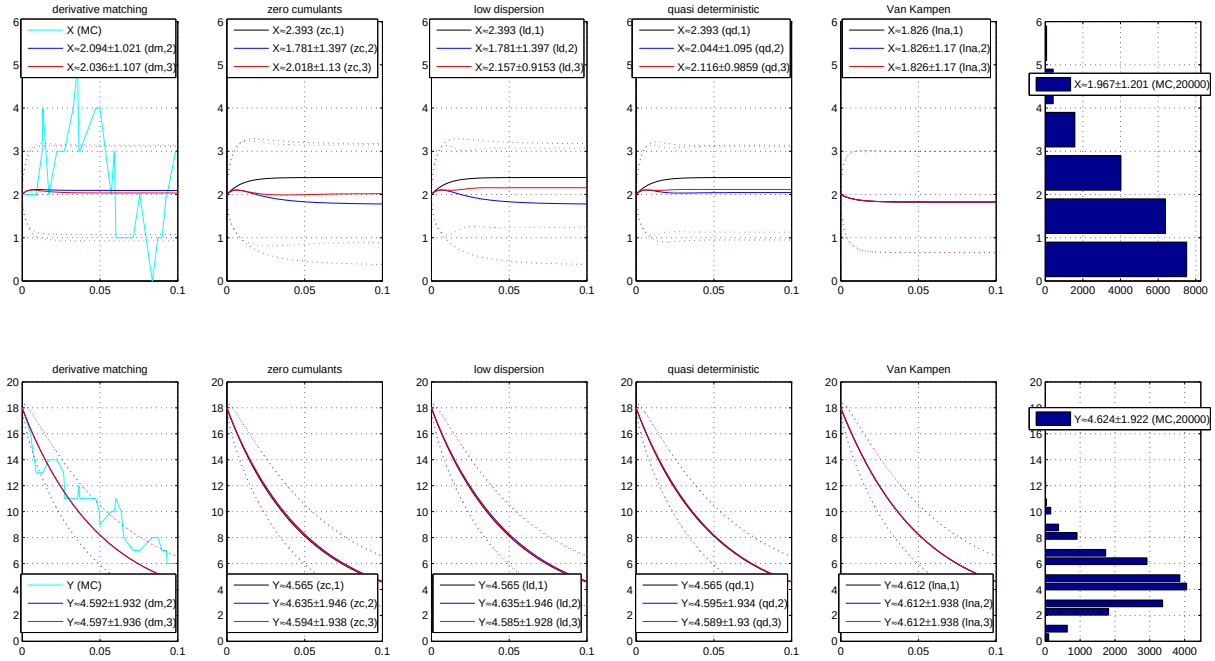
$$\frac{d}{dt}\begin{bmatrix}\phi_X\\ \phi_Y\end{bmatrix} = \begin{bmatrix}-2\gamma{\phi_X}^2+\alpha\phi_A\\ -\beta\phi_Y+\gamma{\phi_X}^2\end{bmatrix}$$

$$\frac{d}{dt}\begin{bmatrix}\xi_X\\ \xi_Y\\ \xi_{X^2}\\ \xi_{XY}\\ \xi_{Y^2}\end{bmatrix} \approx \begin{bmatrix}-4\gamma\phi_X & 0 & 0 & 0 & 0\\ 2\gamma\phi_X & -\beta & 0 & 0 & 0\\ 0 & 0 & -8\gamma\phi_X & 0 & 0\\ 0 & 0 & 2\gamma\phi_X & -\beta-4\gamma\phi_X & 0\\ 0 & 0 & 0 & 4\gamma\phi_X & -2\beta\end{bmatrix}\begin{bmatrix}\xi_X\\ \xi_Y\\ \xi_{X^2}\\ \xi_{XY}\\ \xi_{Y^2}\end{bmatrix} + \begin{bmatrix}0\\ 0\\ 4\gamma{\phi_X}^2+\alpha\phi_A\\ -2\gamma{\phi_X}^2\\ \gamma{\phi_X}^2+\beta\phi_Y\end{bmatrix}$$

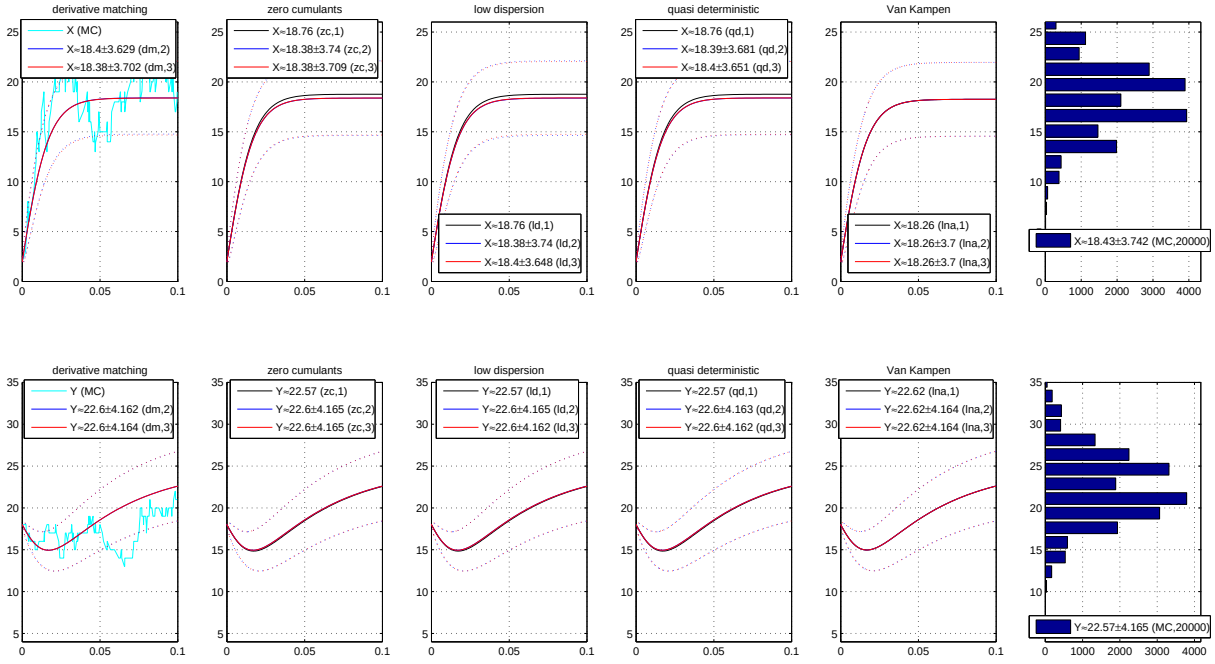
leading to

$$\frac{d}{dt}\begin{bmatrix}\mathrm{E}[X]\\ \mathrm{E}[Y]\\ \mathrm{E}[X^2]\\ \mathrm{E}[XY]\\ \mathrm{E}[Y^2]\end{bmatrix} \approx \begin{bmatrix}-4\gamma\phi_X & 0 & 0 & 0 & 0\\ 2\gamma\phi_X & -\beta & 0 & 0 & 0\\ 4\gamma V{\phi_X}^2+2\alpha\phi_A V & 0 & -8\gamma\phi_X & 0 & 0\\ -\gamma V{\phi_X}^2 & 2\gamma V{\phi_X}^2+\alpha\phi_A V & 2\gamma\phi_X & -\beta-4\gamma\phi_X & 0\\ 0 & -2\gamma V{\phi_X}^2 & 0 & 4\gamma\phi_X & -2\beta\end{bmatrix}\begin{bmatrix}\mathrm{E}[X]\\ \mathrm{E}[Y]\\ \mathrm{E}[X^2]\\ \mathrm{E}[XY]\\ \mathrm{E}[Y^2]\end{bmatrix} + \begin{bmatrix}2\gamma V{\phi_X}^2+\alpha\phi_A V\\ -\gamma V{\phi_X}^2\\ 4\gamma V{\phi_X}^2+\alpha\phi_A V\\ -2\gamma V{\phi_X}^2\\ \gamma V{\phi_X}^2+\beta V\phi_Y\end{bmatrix}$$

Figure 2.1 compares the accuracy of the different moment closure methods for a low volume ($V = 2$) and a high volume ($V = 20$). For the larger volume all moment closure techniques provide a very good match with Monte Carlo results, but for the smaller volume derivative matching produces the most accurate results even with only a second order truncation. These results are fairly typical.



(a) volume $V = 2$



(b) volume $V = 20$

Figure 2.1: Comparison of accuracy between different moment closure methods for the example described in Section 2.6 with two different volumes. The legends show (i) the values of the mean $\pm$ one standard deviation at the final time (ii) a two-character string indicating the moment closure method, and (iii) an integer indicating the order of truncation. The distributions, means and standard deviations in the right-most plots were obtained using 20,000 Monte Carlo simulations produced by [1]. The left-most plots include a typical Monte Carlo run.

Chapter 3

Specifying networks of chemical reactions

In `StochDynTools`, networks of chemical reactions are specified by "`.net` files" whose syntax is described in this section. These files are read using the `StochDynTools` function `readNet()`, which is also described here.

3.1 The `.net` file

A `.net` file can contain up to five sections: `species`, `parameters`, `substitutions`, `reactions`, and `derivatives`. The `substitutions` and `derivatives` sections are optional. Examples of `.net` files were provided in Section 1.

Species section The `species` section describes the chemical species involved in the network. It also specifies some assumptions that can be made in analyzing the system. Following a `species:` header, each line is of the form:

```
species_name "species_latex_name" species_type initial_value;
```

where

- `species_name` stands for the symbol that identifies a chemical species.

  The symbol should contain no spaces and be a valid name for a MATLAB® variable.

- `species_latex_name` stands for an optional LaTeX string that represents the species population to be used when expressions are converted to a LaTeX form.

  This string appears quoted by " and should not include any other quotes.

- `species_type` stands for the assumed type for the species.

  The species type specifies assumptions that can be made in analyzing the system. It can be one of the following keywords:

  - `constant` should be used when the population of the species remains constant;
  - `boolean` should be used when only 0 or 1 molecules can be present (typically genes that can either be active or inactive);
  - `deterministic` should be used when the expected number of molecules is much larger than its standard deviation and therefore stochastic effects can be neglected for this species;
  - `stochastic` should be used when no assumptions can be made about this species.

  Attention! In principle one could declare all populations as `stochastic`, but there are several advantages to providing more information about the populations:

1. Assuming that the assumptions are indeed true, the moment dynamics obtained will be more accurate.
2. The order of the set of ODEs will generally be lower.
3. For populations with low stochasticity (i.e., for which standard deviations are much smaller than the means), if one does not declare the population as `deterministic` one runs the risk of obtaining slightly negative variances due to the moment closure approximation errors.

- `initial_value` is an optional constant that specifies the initial number of molecules for numerical simulations.

  Attention! Starting with initial conditions for which some stochastic species have exactly zero molecules can be problematic when using moment closure based on derivative matching (leading to divisions by zero or errors in the ODE solver). To avoid this, one can initialize the system with a small but positive population (say .01 molecules).

Parameters section The `parameters` section declares all parameters that appear in the `reactions` and `derivatives` sections and provides default numerical values for the parameters . Following a `parameters:` header, each line is of the form:

```
parameter_name "parameter_latex_name" = default_value;
```

where

- `parameter_name` stands for the symbol that identifies the parameter.

  The symbol should contain no spaces and be a valid name for a MATLAB® variable.

- `parameter_latex_name` stands for an optional LaTeX string that represents the parameter to be used when expressions are converted to a LaTeX form.

  This string appears quoted by " and should not include any other quotes.

- `default_value` is a symbolic expression that stands for the default value of the parameter.

  All symbolic computations ignore this value and treat parameters as symbolic variables. Default values are only used when numerical values are needed.

Substitutions section The `substitutions` section provides replacement rules that should be used to simplify computations that involve the populations of the different species. Following a `rules:` header, each line is of the form:

```
{old_expression} > {new_expression};
```

where

- `old_expression` stands for a valid MATLAB® expression that should be replaced by `new_expression`. The rule is applied multiple times until a "fixed point" is achieved.

  For example, if a species `X` is known to have either 1 or 2 individuals, then one should use the rule

  ```
  {X^2} > {3*X-2}
  ```

  On the other hand, if the population of `X` is known to be in the set $\{0,1,2\}$, then one should use

  ```
  {X^3} > {3*X^2-2*X}
  ```

  Currently, these rules are used in the following situations

- To simplify the computation of moment dynamics (prior to any moment closure) in the functions `closureDynamics()` and `momentDynamics()`.

- To simplify the computation of expected values, standard deviations, and distributions in the functions `getCMoments()`, `plotCMoments`, and `getDistribution()`.

These rules are applied *before* processing any simplification rules that arise from species being declared as `boolean` or `deterministic`.

Reactions section The `reactions` section describes the chemical reactions involved in the network, including their stoichiometry and rates. Following a `reactions:` header, each line is of the form:

```
rate = rate_expr;  {list_species} > {post_reaction_counts};
```

where

- `rate_expr` stands for an expression describing the rate at which the reaction occurs.

  The expression should be a *polynomial*, possibly dependent on symbolic variables for which numerical values will be provided later. It should be a valid MATLAB® expression.

- `list_species` is a comma-separated list with the symbols of the chemical species whose stoichiometry changes in the reaction.

- `post_reaction_counts` is a comma-separated list of expressions that specifies how the molecule counts for each symbol changes when the reaction takes place.

Derivatives section The `derivatives` section describes equations for possible continuous evolutions for some/all of the species populations. Following a `derivatives:` header, each line is of the form:

```
d/dt species_name += derivative_expr;
```

where

- `species_name` stands for the symbol of the chemical species whose derivative is provided.

- `derivative_expr` stands for an expression describing the rate of change of the population of `species`. This change in the population is to be *added* to the discrete rules specified in the `reactions` section and to other rules for the same species that may be specified in the `derivatives` section.

  The expression should be a *polynomial*, possibly dependent on symbolic variables for which numerical values will be provided later. It should be a valid MATLAB® expression.

Comments Comments can be inserted anywhere in a `.net` file with the prefix `%`.

3.2 Reading a `.net` file

A `.net` file can be read using the following `StochDynTools` function:

readNet()

```
net=readNet(filename)
```

This `StochDynTools` function reads a `.net` file and stores its contents in the `net` structure.

Inputs and outputs:

- `filename` specifies the name of the `.net` file to read
- `net` is a structure that describes the network in a format that is recognized by the `StochDynTools` toolbox. This structure has the following main fields:
  - `net.species` describes the species involved (from the `species` section of the `.net` file);
  - `net.parameter` contains default numerical values for parameters that appear in the symbolic expressions in `net.reactions` and `net.raterule` (from the `parameters` section of the `.net` file);
  - `net.substitutionrule` describes the replacement rules that should be used to simplify computations that involve populations (from the `rules` section of the `.net` file).
  - `net.reaction` describes the chemical reactions (from the `reactions` section of the `.net` file).
  - `net.raterule` describes the continuous rates of change for the populations (from the `derivatives` section of the `.net` file).

Chapter 4

Computing moment dynamics

In `StochDynTools`, the exact and approximate moment dynamics for a network of chemical reactions are characterized by the `mdyn` structure described in this section.

4.1 Creating a `mdyn` structure

The approximate moment dynamics described in Section 2 can be computed using the `StochDynTools` function `closureDynamics()`.

closureDynamics()

```
mdyn=closureDynamics(net,maxdeg,method,funname,symParameters)
```

This `StochDynTools` function computes the exact (open) moment dynamics (2.1) and then uses the moment closure technique specified by the input parameter `method` to compute approximate (closed) moment dynamics of the form (2.2) or (2.3). As described in [5], the (exact) time-derivative of the uncentered moment

$$\mu^{(m)} := \mathrm{E}[\psi(x_1,x_2,\ldots,x_n)], \qquad \psi(x_1,x_2,\ldots,x_n) := x_1^{m_1}x_2^{m_2}\cdots x_n^{m_n}$$

can be obtained by computing

$$\frac{d\mu^{(m)}}{dt} = \mathrm{E}[(L\psi)(x_1,x_2,\ldots,x_k)] \tag{4.1}$$

where $(L\psi)(x_1,x_2,\ldots,x_k)$ is an expression obtained by applying to $\psi(\cdot)$ the generator of the Markov process that describes the populations. Regardless of the method used, the following is assumed in computing the expected value in the left-hand side of (4.1):

1. All replacement rules specified in the `rules` section of the `.net` file are applied to $(L\psi)(x_1,x_2,\ldots,x_k)$ *before* taking the expected value in the right-hand side of (4.1).

2. For every species x_1 that was declared `deterministic`, it is assumed that

$$\mathrm{E}\left[x_1 f(x_2,\ldots x_k)\right] = \mathrm{E}[x_1]\,\mathrm{E}\left[f(x_2,\ldots x_k)\right] \tag{4.2}$$

   for any function $f(.)$ of the remaining species.

3. For every species x_1 that was declared `boolean`, it is assumed that $x_1^n = x_1$, $\forall n \geq 1$. This introduces no error as long as the number of molecules of x_1 is indeed restricted to the set $\{0,1\}$.

Inputs:

- `net` is a structure that describes the network of chemical reactions. It is typically obtained from a `.net` file using `net=readNet(filename)`.
- `maxdeg` is an integer that specifies the largest degree for the uncentered moments in μ. If `maxdeg` is not an integer, then boolean variables are not taken into account for the degree of a moment. In this case, more moments are included in μ for the same value of `maxdeg`.
- `method` (optional) is a string that specifies the method that should be used for moment closure. This parameter is optional, and in its absence the first method below is used. The following methods are currently recognized:
  - `'dm'` or `'derivativematching'` for moment closure obtained by matching the derivatives of the exact and approximate moment dynamics [5]. See Section 2.1.
  - `'zc'` or `'zerocumulants'` for moment closure obtained by assuming that the high-order cumulants corresponding to all unknown moments are equal to zero. See Section 2.2.

    For second order closure and elementary reactions, this corresponds to the technique described in [2]. However, this function can be used for closures of any order and for reactions with more than two reactants (non-elementary).
  - `'ld'` or `'lowdispersion'` for moment closure obtained by assuming that the high-order normalized centered moments corresponding to all unknown moments are equal to zero. See Section 2.3.

    For second order closure and elementary reactions, this corresponds to the technique described in [2]. However, this function can be used for closures of any order and for reactions with more than two reactants (non-elementary).
  - `'qd'` or `'quasideterministic'` for moment closure obtained by assuming that the high-order quasi-deterministic normalized centered moments corresponding to all unknown moments are equal to zero.

    For this approximation, the closed dynamics are of the form (2.3), where ϕ is the solution to the deterministic dynamics (in molecule counts). See Section 2.4.
  - `{'lna','Volume'}` or `{'vankampen','Volume'}` or `{'linearnoiseapproximation','Volume'}` for Van Kampen's linear noise approximation [6, Chapter X]. The second element of the cell specifies the variable to be used as the volume. This variable should have be declared in the `parameters` section of the `.net` file. To achieve moment closure, one considers the limit as this variable converges to infinity. For this limit to result in a closed set of moment equations, the all reactions should be elementary and their rates should depend on the volume as follows:

    $$\emptyset \xrightarrow{\text{rate} = c\,\texttt{Volume}} * \qquad X \xrightarrow{\text{rate} = cX} * \qquad 2X \xrightarrow{\text{rate} = c\frac{X(X-1)}{\texttt{Volume}}} * \qquad X+Y \xrightarrow{\text{rate} = c\frac{XY}{\texttt{Volume}}} *$$

    For this approximation, the closed dynamics are of the form (2.3), where ϕ is the solution to the deterministic dynamics (in concentrations). See Section 2.5.
  - `'zv'` or `'zerovariance'` or `'zv'` for moment closure assuming a negligible variance for the populations, i.e., assuming that the equality (4.2) holds for all species.
- `funname` (optional) is a string containing the filename used to create an m-file that computes the left-hand side of the differential equations in (2.2) or (2.3). For approximate dynamics of the form (2.3), the μ components of the state appear above the ϕ components. When the function created by `closureDynamics()` is called with no arguments, it outputs the names of the state variables (as a string array).

  The function created can be used as input to `ode23s()` to simulate the moment dynamics (see Section 6 and the example in Section 1).
- `symParameters` (optional) is a vector of symbolic variables corresponding to parameters that appear in the rate expressions (e.g., rate constants).

This vector is used only in the creation of the function `funname` and it allows inclusion of additional inputs to this function that may be needed to specify numerical values for parameters that appear in the moment dynamics. These values will override any default values specified in `net.parameter`.

Outputs:

- `mdyn` is a structure characterizing the exact moment dynamics in (2.1) and the approximate moment dynamics in (2.2) or (2.3) (see Section 4.2).

Under the hood: The function `closureDynamics()` essentially combines the functionality of the three lower-level functions: `momentDynamics()`, `momentClosure()`, and `sym2mfile()`. Documentation for these functions is provided in the corresponding m-files.

4.2 The `mdyn` structure

The `mdyn` structure characterizes the exact and approximate moment dynamics, including

1. the entries of the μ vector in the exact and approximate moment dynamics (`Mu` field),
2. the exact (open) moment dynamics in (2.1) (`dotMu` field),
3. the approximate (closed) moment dynamics in (2.2) (`approxDotMu` field) or in (2.3) (`approxDotMu` and `dotPhi` fields), and
4. set of formatting rules to produce the moment dynamics in LaTeX format (`texrules` field).

A detailed explanation of the key fields of this structure follows. This structure may contain additional fields not described here, which are used internally by `StochDynTools`.

Exact moment dynamics The following fields describe the exact moment dynamics in (2.1):

- `Mu` is a structure describing the entries of the vector μ in (2.1) and the vector ν in (2.2) or (2.3b). The following table describes the key fields of this structure for a network of chemical reactions with species `X1,X2,X3,...`:

field	type	value of row corresponding to moment $E[X1^{m1}X2^{m2}X3^{m3}\cdots]$
`Mu.sym`	column vector of symbolic variables	`mu_X1m1_X2m2_X3m3...`
`Mu.mon`	column vector of symbolic expressions	`X1^m1*X2^m2*X3^m3...`
`Mu.ndx`	matrix of integers with one column per species (in the order they were declared in the `.net` file)	`m1,m2,m3,...`
`Mu.x0`	column vector of doubles	initial value from the `.net` file

- `barMu` is a structure describing the entries of the vector $\bar{\mu}$ in (2.1). The following table describes the key fields of this structure for a network of chemical reactions with species `X1,X2,X3,...`:

field	type	value of row corresponding to moment $E[X1^{m1}X2^{m2}X3^{m3}\cdots]$
`barMu.sym`	column vector of symbolic variables	`mu_X1m1_X2m2_X3m3...`
`barMu.mon`	column vector of symbolic expressions	`X1^m1*X2^m2*X3^m3...`
`barMu.ndx`	matrix of integers with one column per species (in the order they were declared in the `.net` file)	`m1,m2,m3,...`

- `dotMu` is a structure describing the exact moment dynamics (2.1). It contains the following fields:

field	type	value
`dotMu.A`	matrix of symbolic expressions	matrix A in (2.1)
`dotMu.B`	matrix of symbolic expressions	matrix B in (2.1)
`dotMu.sym`	column vector of symbolic expressions	whole right-hand side of (2.1)

The right-hand side of (2.1) can thus be obtained using any one of the following two symbolic expressions:

```
dotMu.A * Mu.sym + dotMu.B * barMu.sym
dotMu.sym
```

Approximate moment dynamics The following fields describe the approximate moment dynamics in (2.2) or (2.3b), depending on the moment closure technique used. It contains the following fields:

- `approxDotMu` is a structure describing the right-hand sides of (2.2) or (2.3b). It contains the following fields:

field	type	value
`approxDotMu.barMu`	column vector of symbolic expressions	approximate value of $\bar{\mu}$ in (2.1)
`approxDotMu.sym`	column vector of symbolic expressions	right-hand sides of (2.2) or (2.3b)
`approxDotMu.A`	matrix of symbolic expressions	Jacobian of `approxDotMu.sym` with respect to `Mu.sym`
`approxDotMu.c`	vector of symbolic expressions	`approxDotMu.sym-approxDotMu.A*Mu.sym`

The right-hand side of (2.2) or (2.3b) can thus be obtained using any one of the following three symbolic expressions:

```
dotMu.A * Mu.sym + dotMu.B * approxDotMu.barMu
approxDotMu.sym
approxDotMu.A * Mu.sym + approxDotMu.c
```

The last representation is especially useful for the `'qd'` and `'lna'` moment closure methods because in this case the matrices `approxDotMu.A` and `approxDotMu.c` do not depend on the entries of `Mu.sym` [cf. (2.10)].

Currently, the moment closure method `'lna'` does not return the field `approxDotMu.barMu`

The following fields are only needed to describe the approximate moment dynamics in (2.3).

- `Phi` is a structure describing the entries of the vector ϕ in (2.3a). The following table describes the key fields of this structure for a network of chemical reactions with species `X1,X2,X3,...`:

field	type	value of row corresponding to moment $E[X1^{m1}X2^{m2}X3^{m3}\cdots]$
`Phi.sym`	column vector of symbolic variables	`phi_X1m1_X2m2_X3m3...`
`Phi.mon`	column vector of symbolic expressions	`X1^m1*X2^m2*X3^m3...` (`'qd'` method) `(X1/V)^m1*(X2/V)^m2*(X3/V)^m3...` (`'lna'` method)
`Phi.x0`	column vector of doubles	initial value from the `.net` file

- `dotPhi` is a vector of symbolic expressions with the derivative of ϕ in (2.3a).

The following additional fields are only returned by the `'lna'` moment closure method:

- `Xi` is a structure describing the entries of the first-order "perturbation" vector ξ in (2.14). The following table describes the key fields of this structure for a network of chemical reactions with species `X1,X2,X3,...`:

field	type	value of row corresponding to $\mathrm{E}\left[\left(\frac{X1-V\phi_1}{V^{\frac{1}{2}}}\right)^{m1}\left(\frac{X2-V\phi_2}{V^{\frac{1}{2}}}\right)^{m2}\left(\frac{X3-V\phi_3}{V^{\frac{1}{2}}}\right)^{m3}\ldots\right]$
`Xi.sym`	column vector of symbolic variables	`xi_X1m1_X2m2_X3m3...`
`Xi.mon`	column vector of symbolic expressions	`((X1-V*phi_X1)/V^(1/2))^m1*...`
`Xi.mu2xi`	column vector of symbolic expressions	expression for $\mathrm{E}\left[\left(\frac{X1-V\phi_1}{V^{\frac{1}{2}}}\right)^{m1}\left(\frac{X2-V\phi_2}{V^{\frac{1}{2}}}\right)^{m2}\left(\frac{X3-V\phi_3}{V^{\frac{1}{2}}}\right)^{m3}\ldots\right]$ as a function of the entries of μ and ϕ
`Xi.xi2mu`	column vector of symbolic expressions	expression for the uncentered moment $\mathrm{E}\left[X1^{m1}X2^{m2}X3^{m3}\ldots\right]$ in μ as a function of the entries of ξ and ϕ
`Xi.xi2barMu`	column vector of symbolic expressions	expression for the uncentered moment $\mathrm{E}\left[X1^{m1}X2^{m2}X3^{m3}\ldots\right]$ in $\bar{\mu}$ as a function of the entries of ξ and ϕ

- `dotXi` is a symbolic vector with the exact derivative of ξ as in (2.14).
- `approxDotXi` is a structure describing the approximate derivative of ξ in the right-hand side of (2.15). It contains the following fields:

field	type	value
`approxDotXi.sym`	column vector of symbolic variables	right-hand side of (2.15)
`approxDotXi.A`	matrix of symbolic expressions	Jacobian of `approxDotXi.sym` with respect to `Xi.sym`
`approxDotXi.c`	vector of symbolic expressions	`approxDotXi.sym-approxDotXi.A*Xi.sym`

The right-hand side of (2.15) can thus be obtained using any one of the following two symbolic expressions:

```
approxDotXi.sym
approxDotXi.A * Xi.sym + approxDotXi.c
```

The last representation is especially useful because the matrices `approxDotXi.A` and `approxDotXi.c` do not depend on the entries of `Xi.sym` [cf. (2.10)].

LaTeX formatting The following field is used to produced LaTeX-formatted versions of any symbolic expression involving reaction parameters, moments, or exact and approximate moment dynamics. These rules should be used by the `StochDynTools` function `mylatex()` described in Section 7.

- `texrules` is a cell array of strings with one row per transformation rule and two columns:
  - the first column contains a regular expression, following the syntax of MATLAB®'s `regexp()`
  - the second column contains the string to replace the regular expression.

Chapter 5

Monte Carlo Simulations

One can run Monte Carlo simulations of a network of chemical reactions specified by a `.net` file within MATLAB® using `StochDynTools`or in C++ using StochKit.

Attention! For both simulation methods,

1. the `post_reaction_counts` in the `.net` file must correspond to increments/decrements; and
2. molecule counts are assume constant between reactions and therefore the `derivatives` section of the `.net` file is ignored.

5.1 Using `StochDynTools`

quadPropensities()

```
[Q,b,c,s,x0]=quadPropensities(net)
```

The function `quadPropensities()` describes a network of chemical reactions with quadratic propensity functions. The outputs of this function are used by `sampledSSA()` to run Monte Carlo simulations of the network of chemical reactions.

Inputs:

- `net` is a structure that describes the network of chemical reactions. It is typically obtained from a `.net` file using `net=readNet(filename)`.

Outputs:

- `Q` is a matrix with the quadratic terms for the propensity functions. Each reaction corresponds to a (# species) by (# species) square matrix. These matrices are stacked on top of each other so the size of `Q` is (# reactions $\times$ # species) by (# species).
- `b` is a matrix with the linear terms for the propensity functions. Each reaction corresponds to a vector of length (# species). These vectors are stacked on top of each other so size of `b` is (# reactions) by (# species).
- `c` is a vector with the constant terms for the propensity functions. The constants for each reaction are stacked on top of each other so size of `c` is (# reactions) by 1.
- `s` is a matrix with the stoichiometry for the reactions. Each reaction corresponds to a vector of length (# species). These vector are stacked side by side so size of `s` is (# species) by (# reactions).
- `x0` is a vector with the initial conditions in the `.net` file. The size of `x0` is (# species) by 1.

sampledSSA()

```
[X,Xmean,Xstd,XmeanCI,XstdCI] = sampledSSA(Q,b,c,s,x0,nMC,Ts,CI)
```

The function `sampledSSA` runs multiple Gillespie's stochastic Simulation Algorithms (SSA) for the network of chemical reactions with propsenties specified by the inputs `Q`, `b`, `c` and stoichiometry specified by `s`.

This function returns all the sample paths at a set of sample times, as well as estimates for the means and standard deviations at the sample times.

Inputs:

- `Q` is a matrix with the quadratic terms for the propensity functions. Each reaction corresponds to a (# species) by (# species) square matrix. These matrices are stacked on top of each other so the size of `Q` is (# reactions $\times$ # species) by (# species).
- `b` is a matrix with the linear terms for the propensity functions. Each reaction corresponds to a vector of length (# species). These vectors are stacked on top of each other so size of `b` is (# reactions) by (# species).
- `c` is a vector with the constant terms for the propensity functions. The constants for each reaction are stacked on top of each other so size of `c` is (# reactions) by 1.
- `s` is a matrix with the stoichiometry for the reactions. Each reaction corresponds to a vector of length (# species). These vector are stacked side by side so size of `s` is (# species) by (# reactions).
- `x0` is a vector with the initial conditions in the `.net` file. The size of `x0` is (# species) by 1.
- `nMC` is the number if Monte Carlo simulations to run.
- `Ts` is a vector with the desired sample times for the output.

  The sampled paths are computed exactly at all times, but their values are only returned at the time in the vector `Ts`.
- `CI` is an optional input with the desired percentage for the confidence intervals for the mean and standard deviation. If not specified, 95% is used.

Outputs:

- `X` is matrix with the molecule counts at the sample times. The size of `X` is (# species) by (# simulations) by (# of sample times).
- `Xmean` is a matrix with the mean molecule counts at the sample times. The size of `Xmean` is (# species) by (# of sample times).
- `Xstd` is a matrix with the standard deviation molecule counts at the sample times. The size of `Xstd` is (# species) by (# of sample times)
- `XmeanCI` is a matrix with the confidence interval for the `Xmean`. The size of `XmeanCI` is (# species) by (# of sample times) by 2.

  The computation of `XmeanCI` assumes that the central limit theorem is valid to determine the distribution of the mean
- `XstdCI` is a matrix with the confidence interval for the Xstd. The size of XstdCI is (# species) by (# of sample times) by 2.

  The computation of `XstdCI` assumes a normal distribution.

5.2 Using STOCHKIT

net2stochKit()

`net2stochKit(net,filename,x0,symParameters,valueParameters)`

The function `net2stochKit()` automatically creates a C++ file that can be used to create a STOCHKIT executable to run Monte Carlo simulations of a network of chemical reactions.

Attention! All the `post_reaction_counts` in the `.net` file must correspond to increments/decrements. If this is not the case the C++ code will not compile.

Inputs and Outputs:

- `net` is a structure that describes the network of chemical reactions. It is typically obtained from a `.net` file using `net=readNet(filename)`.
- `filename` is a string containing the filename of the C++ file to be created (without the `.cpp` extension).
- `x0` is a vector of initial populations for the Monte Carlo simulations with as many entries as the number of chemical species.
- `symParameters` (optional) is a vector of symbolic parameters that appear in the rate expressions (e.g., rate constants).
- `valueParameters` (optional) is a vector of numerical values for the symbolic parameters in symParameters.

The function `net2stochKit()` returns no output, but creates a C++ STOCHKIT "ProgramDefinition" file. To learn how to use this file, please consult the StochKit user guide [1]. An example Makefile and two "main" C++ files are provided as examples. However, these will only work if (i) STOCHKIT has been successfully installed and (ii) the variable `CSE_CPP_HOME` in the Makefile points to the STOCHKIT directory.

An error is returned if equations for continuous evolution are specified in the `derivatives` section because STOCKKIT cannot simulate reaction networks with continuous variations in the populations.

Chapter 6

Simulation and plots

The m-file produced by `closureDynamics()` can be passed to an ODE solver such as `ode23s()` to solve the approximate moment dynamics. The solution to the ODE can then be used to compute and plot centered moments using `getCMoments()` and `plotCMoments()`, respectively. It can also be used to obtain an approximate probability distribution using `getDistribution()`.

ode23s()

```
[t,mu]=ode23s(@(t,x)funname(x,parameter_list),[0,Tmax],x0);
```

This (standard) MATLAB® function can be used to solve the approximate moment dynamics.

Inputs:

- `funname` is a string containing the filename of the m-file created by `closureDynamics()` to compute the left-hand side of (2.2) or (2.3) (without the `.m` extension).
- `parameter_list` is a list of parameter values to be passed to `funname()`, as specified by the input parameter `symParameters` to the function `closureDynamics()`.
- `Tmax` is the time at which the simulation should terminate.
- `x0` is the initial condition for the moment dynamics.

  For moment dynamics of the form (2.2), `x0` must have the size of μ and to obtain the initial the conditions in the `.net` file one would choose `x0=mdyn.Mu.x0`, with `mdyn` returned by `closureDynamics()`.

  For moment dynamics of the form (2.3), `x0` must have the size of $\begin{bmatrix}\mu' & \phi'\end{bmatrix}'$ and to obtain the initial the conditions in the `.net` file one would choose `x0=[mdyn.Mu.x0;mdyn.Phi.x0]`, with `mdyn` returned by `closureDynamics()`.

  Attention! Starting with initial conditions for which some species have exactly zero molecules can be problematic when using moment closure based on derivative matching (because this may lead to divisions by zero or errors in the ODE solver). To avoid this, one can initialize the system with small but positive population (say .1 molecules).

The function `ode23s()` accepts several other parameters (see `help ode23s`). Other ODE solvers can also be used, however a stiff solver is generally preferable.

Outputs:

- `t` is a column vector with the time instants at which the solution was computed.
- `mu` is a time series of μ for moment dynamics of the form (2.2) or a time series of $\begin{bmatrix}\mu' & \phi'\end{bmatrix}'$ for moment dynamics of the form (2.3). This vector can be plotted using the following command

```
plot(t,mu)
legend(funname(),'location','best','interpreter','none')
```

  where `funname()` is the m-file created by `closureDynamics()`. Recall that when the function created by `closureDynamics()` is called without arguments it returns the names of the variables in μ, which can be used to label the plot. The option `'interpreter','none'` to the command `legend()` prevents MATLAB® from interpreting underscores as subscripts.

getCMoments()

```
[average,stddev]=getCMoments(net,mdyn,mu,symExpression)
```

Given a time series $\mu(t)$ of uncentered moments, typically computed using `ode23s()`, this `StochDynTools` function computes the means and standard deviations of a given vector of expressions involving the populations of the different species.

Inputs:

- `net` is a structure that describes the network of chemical reactions. It is typically obtained from a `.net` file using `net=readNet(filename)`.
- `mdyn` is a structure characterizing the exact moment dynamics in (2.1) and/or the approximate moment dynamics in (2.2) or (2.3), typically computed using `closureDynamics()`.
- `mu` is a matrix containing a time series of μ or $\begin{bmatrix}\mu' & \phi'\end{bmatrix}'$, typically computed using `ode23s()`. Each row of `mu` corresponds to a different time instant and each column to a different moment.
- `symExpression` is an array of symbolic expressions whose means and standard deviations will be computed. The function returns `NaN` if the computation of a mean and/or standard deviation cannot be computed exactly using the moments available.

  The replacement rules in `net.substitutionrule` will be applied to the symbolic expression (and to its square in order to compute the variance) before taking expectations. The boolean nature of the variables declared as such, will be taken into account.

  Attention! In the current implementation of this function, the deterministic nature of the variables declared as such will be ignore. This means, e.g., that `NaN` will be generated if one asks for the expected value of `X^2` and `mu` only contains the expected value of `X`, *even if* `X` *was declared deterministic*.

Outputs:

- `average` is a time series of the expected value of the vector `symExpression`, with one time instant per row and one element of the `symExpression` per column.
- `stddev` (optional) is a time series of the standard deviation of the vector `symExpression`, with one time instant per row and one element of the `symExpression` per column.

  Attention! For populations with low stochasticity (i.e., for which standard deviations are much smaller than the means), the moment closure approximation errors can lead to negative variances. Such populations should be declared as `deterministic` in the `.net` file.

plotCMoments()

```
[handles,average,stddev]=plotCMoments(net,mdyn,t,mu,symExpression_1,style_1,symExpression_2,style_2,...)
```

Given a time series $\mu(t)$ of uncentered moments, typically computed using `ode23s()`, this `StochDynTools` function computes and plots the means and standard deviations of a given vector of expressions involving the populations of the different species. Internally, this function uses `getCMoments()` to compute means and standard deviations.

Inputs:

- `net` is a structure that describes the network of chemical reactions. It is typically obtained from a `.net` file using `net=readNet(filename)`.
- `mdyn` is a structure characterizing the exact moment dynamics in (2.1) and/or the approximate moment dynamics in (2.2) or (2.3), typically computed using `closureDynamics()`.
- `mu` is a matrix containing a time series of μ or $\left[\mu' \quad \phi'\right]'$, typically computed using `ode23s()`. Each row of `mu` corresponds to a different time instant and each column to the different moment.
- `symExpression_1,symExpression_2,...` are symbolic expressions whose means and standard deviations will be plotted. The means of each symExpression_i will be plotted with the style defined by `style_i` and dotted lines will be used to depict the mean plus/minus one standard deviation.

  This function uses `getCMoments()` so the reader is referred to the documentation of that function for details on the computation of means and standard deviations.
- `style_1,style_2,...` are character strings defining the style of the line to be used, as in MATLAB®'s `plot()` command.

Outputs:

- `handles` is a vector with the handles of all the lines plotted. These handles are useful, for example to produce legends, as in

```
h=plotCMoments(net,monMu,t,mu,'E','b-','St','g-','Et','k-');
legend(h(1:3:end),'E[E]\pm{}Std[E]','E[St]\pm{}Std[St]','E[Et]','location','best');
```

  where by using only one out of each three handles, one skips legends for the lines corresponding to the standard deviations.
- `average` is a time series with the expected values of all expressions, with one time instant per row and one expression per column.
- `stddev` is a time series with the standard deviations of all expressions, with one time instant per row and one expression per column.

getDistribution()

```
pdf=getDistribution(net,mdyn,mu,symExpression,x,distribution,symBoolean)
```

Given a vector μ of uncentered moments, this `StochDynTools` function estimates the distribution of a given expression involving the populations of the different species. Internally, this function uses `getCMoments()` to compute means and standard deviations.

Inputs:

- `net` is a structure that describes the network of chemical reactions. It is typically obtained from a `.net` file using `net=readNet(filename)`.

  `mdyn` is a structure characterizing the exact moment dynamics in (2.1) and the approximate moment dynamics in (2.2) or (2.3)

- `mu` is a vector containing the value of μ or $\begin{bmatrix}\mu' & \phi'\end{bmatrix}'$ at a given time instant, typically computed using `ode23s()`.

- `symExpression` is a symbolic expression whose distribution will be estimated.

  This function uses `getCMoments()` to compute the mean and standard deviation from which the distribution is estimated. The reader is referred to the documentation of `getCMoments()` for details on the computation of means and standard deviations.

- `x` is a vector containing the points at which the distribution will be computed.

- `distribution` is a character string specifying which type of distribution should be assumed. This parameter can take the following values:

  - `'normal'` when a normal distribution should be assumed.
  - `'lognormal'` when a lognormal distribution should be assumed.
  - `'binomial'` when a binomial distribution should be assumed.
  - `'mix_normal'` when it should be assumed that the distribution is normal, when conditioned to any one of the two values for a given boolean-valued expression `symBoolean`. This will result in a convex combination (mixture) of two normal distributions.
  - `'mix_lognormal'` when it should be assumed that the distribution is lognormal, when conditioned to any one of the two values for a given boolean-valued expression `symBoolean`. This will result in a convex combination (mixture) of two lognormal distributions.

- `symBoolean` is a boolean-valued symbolic expression needed for any of the "mixture" distributions discussed above.

Output:

- `pdf` is a vector containing the values of the probability density function (pdf) at the points in `x`.

plotCompare()

`[exact,approx,MC]=plotCompare(net,yPlots,xPlots,Tmax,varargin)`

This function produces an array of plots that can be used to compare several moment closure techniques and Monte Carlo Simulations.

This function is currently undocumented but we provide an example of its use below. More examples can be found in the `/examples` *folder.*

```
figure(1);clf
[exact,approx,MC]=plotCompare(net,2,5,Tmax,...
    {{'derivative␣matching',X1minmax},...                    % subfigure (2,5,1)
           {'MCsample',1,'X1','c-'},{'dm',2,'X1','b-'},{'dm',3,'X1','r-'}},...
    {{'zero␣cumulants',X1minmax},...                         % subfigure (2,5,2)
           {'zc',1,'X1','k-'},{'zc',2,'X1','b-'},{'zc',3,'X1','r-'}},...
    {{'low␣dispersion',X1minmax},...                         % subfigure (2,5,3)
```

```
       {'ld',1,'X1','k-'},{'ld',2,'X1','b-'},{'ld',3,'X1','r-'}},...
{{'quasi deterministic',X1minmax},...                    % subfigure (2,5,4)
       {'qd',1,'X1','k-'},{'qd',2,'X1','b-'},{'qd',3,'X1','r-'}},...
{{'distribution (Monte Carlo)',X1minmax},...             % subfigure (2,5,5)
       'MCdistribution',nMC,'X1'},...
{{'derivative matching',X2minmax},..                     % subfigure (2,5,6).
       {'MCsample',1,'X2','c-'},{'dm',2,'X2','b-'},{'dm',3,'X2','r-'}},...
{{'zero cumulants',X2minmax},...                         % subfigure (2,5,7)
       {'zc',1,'X2','k-'},{'zc',2,'X2','b-'},{'zc',3,'X2','r-'}},...
{{'low dispersion',X2minmax},...                         % subfigure (2,5,8)
       {'ld',1,'X2','k-'},{'ld',2,'X2','b-'},{'ld',3,'X2','r-'}},...
{{'quasi deterministic',X2minmax},...                    % subfigure (2,5,9)
       {'qd',1,'X2','k-'},{'qd',2,'X2','b-'},{'qd',3,'X2','r-'}},...
{{'distribution (Monte Carlo)',X2minmax},...             % subfigure (2,5,10)
       'MCdistribution',nMC,'X2'});
```

In this example,

1. `net` is a structure that describes the network of chemical reactions. It is typically obtained from a `.net` file using `net=readNet(filename)`.

2. `2,5` specifies that an array of 2 by 5 subfigure should be draw. Each subfigure can have several plots. All subplots for the same figure are passed as a cell array.

3. `Tmax` specifies the that all simulations should be produced in the interval `[0,Tmax]` and that Monte Carlos distributions should be computed for the time `Tmax`.

4. `'derivative matching',X1minmax` specifies that the tile of the first subfigure should be `'derivative matching'` and `X1minmax` is a 2-vector that specifies the limits of the y-axis for this subfigure.

5. `'MCsample',1,'X1','c-'` specifies a plot within a subfigure, containing `1` Monte Carlo sample path of the species `X1`, using the line style `c-`.

6. `'dm',2,'X1','b-'` specifies a plot within a subfigure, displaying the moment closure approximate dynamics using derivative matching with moments up to order `2`. The plot with contain the mean of `X1` using the line style `b-`, and dotted lines specifying ± 1 standard deviation.

7. `'MCdistribution',nMC,'X2'` specifies a plot within a subfigure, displaying an histogram with the distribution of `X2` at time `Tmax` obtained from `nMC` Monte Carlo simulations.

Chapter 7

Utilities

subsParameters()

```
new_expression=subsParameters(net,old_expression)
```

The function `subsParameters()` takes a given expression and replaces the parameters from a network of chemical reactions by their default values.

Inputs and Outputs:

- `net` is a structure that describes the network of chemical reactions. It is typically obtained from a `.net` file using `net=readNet(filename)`.
- `old_expression` is a vector of symbolic expressions where the parameters should be eliminated.
- `new_expression` is a vector of symbolic expressions obtained from `old_expression` by replacing any parameters by their default values.

mylatex()

```
str=mylatex(sym,texrules)
```

The function `mylatex()` takes a symbolic expression and returns it into a fairly readable LaTeX syntax. This function allows the latex expression to be transformed by a set of replacement rules, which can be used to include the LaTeX names specified in the `.net` file.

Inputs and Outputs: When calling `mylatex()` without inputs, this function returns a set of macros needed to interpret the LaTeX output. Otherwise...

- `sym` is a symbolic expression to be transformed into LaTeX format.
- `texrules` is a cell array of strings with one row per transformation rule and two columns:
  - the first column contains a regular expression, following the syntax of MATLAB®'s `regexp()`
  - the second column contains the string to replace the regular expression.

  The transformation rules are applied sequentially and only once. Replacements are not applied to strings already replaced.

  `texrules` are typically obtained by the `StochDynTools` command `closureDynamcis()` or `momentDynamics()`.
- `str` is a string in the LaTeX output.

momentEquilibrium()

`[muEq,phiEq,xiEq]=momentEquilibrium(net,mdyn,option)`

This `StochDynTools` function computes the equilibrium points of approximate moment dynamics of the form (2.2) or (2.3).

Inputs:

- `net` is a structure that describes the network of chemical reactions. It is typically obtained from a `.net` file using `net=readNet(filename)`.
- `mdyn` is a structure describing the moment dynamics in (2.2) or (2.3). This structure is typically produced by `closureDynamics()`.
- `numerical` is an optional boolean variable indicating whether or not the parameters should be replaced by the numerical values in `net.parameter` prior to finding the equilibrium points.
- `option` is an optional string specifying options to be used in the computation of the equilibrium points. The following options are currently recognized:
  - `'none'` — no option specified.
  - `'lna'` — use Van Kampen's linear noise approximation (see [6, Chapter X]). Assumes that the `'lna'` moment closure method was used to produce `mdyn`.
  - `'smallCV'` — assume that the distribution of the populations is tightly concentrated around a mean value. Currently does not work for Van Kampen's linear noise approximation.

Outputs:

- `muEq` is a cell array of symbolic vectors with the equilibrium points of the vector μ in (2.2) or (2.3). Each entry of the cell array corresponds to one equilibrium point.

The following additional outputs are available with the options `'lna'` and `'smallCV'`

- `phiEq` is a cell array of symbolic vectors with the equilibrium points of the deterministic vector ϕ around which the populations are concentrated. Each entry of the cell array corresponds to one equilibrium point.
- `xiEq` is a cell array of symbolic vectors with the equilibrium points of the perturbations ξ around ϕ. Each entry of the cell array corresponds to one equilibrium point.

Appendix A

Appendix

A.1 Installation

The `.tar` file posted contains the following three folders:

- `stochdyntools/doc` — Packet documentation (including this file).
- `stochdyntools/lib` — The `StochDynTools` MATLAB® scripts described in this document.
- `stochdyntools/examples` — Examples on the use of this set of tools (including the examples in Section 1).

To start using `StochDynTools` one needs to

1. "Untar" the `.tar` file. In a PC this can be done using any of the usual "unzip" utilities and in Linux this can be done by executing the following command at the shell prompt:

```
tar xvf sotchdyntools.tar
```

2. Add the `stochdyntools/lib` folder to the MATLAB® path. This can be done by executing the following command at the MATLAB® prompt:

```
path('{BASE}stochdyntools/lib',path)
```

   where {`BASE`} should be replaced by the folder where the `.tar` file was "untared." Most likely, you will want to add the above command to your `startup.m` MATLAB® script.

If one wants to use the function `net2stochKit()` to create Monte Carlos simulations, StochKit must be installed [1]. However, this is not needed to compute moment dynamics.

A.2 Algebraic derivations

A.3 Cumulants

The multi-variable cumulants for the distribution of a random vector x are defined by the Taylor series of the cumulant-generating function:

$$g(\lambda) := \log\left(\mathrm{E}[e^{j\lambda\cdot x}]\right),$$

where $\lambda \in \mathbb{R}^n$ and $\cdot$ denotes inner product. In particular, the cumulant $\kappa^{(m)}$ associated with the vector of integers $m := (m_1, m_2, \ldots, m_n)$ is given by

$$\kappa^{(m)} := \frac{\partial^{m_1+m_2+\cdots+m_n} g(0)}{\partial \lambda_1^{m_1} \lambda_2^{m_2} \cdots \lambda_n^{m_n}},$$

which is one of the coefficients in the Taylor series of $g(\lambda)$ around $\lambda = 0$. The cumulant $\kappa^{(m)}$ can be computed by first expanding

$$\mathrm{E}[e^{j\lambda \cdot x}] = \sum_{\ell=0}^{\sum m_i} \frac{1}{\ell!} \mathrm{E}\left[(j \sum_i \lambda_i x_i)^\ell \right] + \mathrm{E}[R(\lambda, x)],$$

which results in

$$\mathrm{E}[e^{j\lambda \cdot x}] = \sum_{\sum \bar{m}_i \leq \sum m_i} c_{\bar{m}}(\lambda) \mu^{(\bar{m})} + \mathrm{E}\left[O\big((\lambda \cdot x)^{1+\sum m_i} \big) \right],$$

where the summation is over moments $\mu^{(\bar{m})}$ of order up to $\sum m_i$, the $c_{\bar{m}}(\lambda)$ are appropriately selected constants that depend on λ, and all terms in the remainder $R(\lambda, x)$ have powers in λ or order higher than $\sum m_i$. The cumulant $\kappa^{(m)}$ is then obtained from

$$\kappa^{(m)} = \sum_{\sum \bar{m}_i \leq \sum m_i} \mu^{(\bar{m})} \frac{\partial^{m_1+m_2+\cdots+m_n} c_{\bar{m}}(0)}{\partial \lambda_1^{m_1} \lambda_2^{m_2} \cdots \lambda_n^{m_n}}.$$

Note that since the derivative is computed at $\lambda = 0$, the terms in $R(\lambda, x)$ will note appear because they only have powers in λ or order higher than $\sum m_i$. In fact, it is straightforward to show that

$$\kappa^{(m)} = \mu^{(m)} + \sum_{\sum \bar{m}_i < \sum m_i} \mu^{(\bar{m})} \frac{\partial^{m_1+m_2+\cdots+m_n} c_{\bar{m}}(0)}{\partial \lambda_1^{m_1} \lambda_2^{m_2} \cdots \lambda_n^{m_n}},$$

which means that the cumulant $\kappa^{(m)}$ depends only on $\mu^{(m)}$ and lower-order moments, so by setting $\kappa^{(m)} = 0$ one obtains an expression for $\mu^{(m)}$ as a function of lower-order moments.

A.4 Uncentered Versus Normalized Centered moments

The uncentered moment $\mu^{(m)}$ can be expressed in terms of the normalized centered moment $\eta^{(\bar{m})}$ as follows

$$\begin{aligned}
\mu^{(m)} &= \mathrm{E}[x_1^{m_1} x_2^{m_2} \cdots x_n^{m_n}] \\
&= \mathrm{E}[x_1]^{m_1} \mathrm{E}[x_2]^{m_2} \cdots \mathrm{E}[x_n]^{m_n} \mathrm{E}\left[\left(1 + \frac{x_1 - \mathrm{E}[x_1]}{\mathrm{E}[x_1]}\right)^{m_1} \left(1 + \frac{x_2 - \mathrm{E}[x_2]}{\mathrm{E}[x_2]}\right)^{m_2} \cdots \left(1 + \frac{x_n - \mathrm{E}[x_n]}{\mathrm{E}[x_n]}\right)^{m_n} \right] \\
&= \mathrm{E}[x_1]^{m_1} \mathrm{E}[x_2]^{m_2} \cdots \mathrm{E}[x_n]^{m_n} \left(1 + \eta^{(m)} + \sum_{2 \leq \sum \bar{m}_i < \sum m_i} \beta_{\bar{m}} \eta^{(\bar{m})}\right),
\end{aligned}$$

where the summation is over normalized centered moments $\eta^{(\bar{m})}$ of order two[1] or larger and strictly smaller than $\sum m_i$; and the $\beta_{\bar{m}}$ are appropriately selected nonnegative constants.

Conversely, we can also express the any normalized centered moments $\eta^{(\bar{m})}$ in terms of uncentered moment $\mu^{(\tilde{m})}$ as follows

$$\eta^{(\bar{m})} = \mathrm{E}\left[\left(\frac{x_1 - \mathrm{E}[x_1]}{\mathrm{E}[x_1]}\right)^{\bar{m}_1} \left(\frac{x_2 - \mathrm{E}[x_2]}{\mathrm{E}[x_2]}\right)^{\bar{m}_2} \cdots \left(\frac{x_n - \mathrm{E}[x_n]}{\mathrm{E}[x_n]}\right)^{\bar{m}_n} \right]$$

[1] The normalized centered moments of order one are always zero.

$$
\begin{aligned}
&= \mathrm{E}\left[\left(\frac{x_1}{\mathrm{E}[x_1]} - 1\right)^{\bar{m}_1}\left(\frac{x_2}{\mathrm{E}[x_2]} - 1\right)^{\bar{m}_2}\cdots\left(\frac{x_n}{\mathrm{E}[x_n]} - 1\right)^{\bar{m}_n}\right] \\
&= \frac{\mu^{(\bar{m})}}{\mathrm{E}[x_1]^{\bar{m}_1}\,\mathrm{E}[x_2]^{\bar{m}_2}\cdots\mathrm{E}[x_n]^{\bar{m}_n}} + \sum_{\sum\tilde{m}_i<\sum m_i} \gamma_{\tilde{m}}\,\frac{\mu^{(\tilde{m})}}{\mathrm{E}[x_1]^{\tilde{m}_1}\,\mathrm{E}[x_2]^{\tilde{m}_2}\cdots\mathrm{E}[x_n]^{\tilde{m}_n}},
\end{aligned}
$$

where the summation is over uncentered moments $\mu^{(\tilde{m})}$ of order strictly smaller than $\sum\bar{m}_i$ and the $\gamma_{\tilde{m}}$ are appropriately selected constants.

The final formulas for low-dispersion moment closure can be obtained directly from expansions of the non-normalized centered moments

$$
\mathrm{E}\left[(x_1-\mathrm{E}[x_1])^{\bar{m}_1}(x_2-\mathrm{E}[x_2])^{\bar{m}_2}\cdots(x_n-\mathrm{E}[x_n])^{\bar{m}_n}\right] = \mu^{(\bar{m})} + \sum_{\sum\tilde{m}_i<\sum m_i} \hat{\gamma}_{\tilde{m}}(\mathrm{E}[x_1],\mathrm{E}[x_2],\ldots,\mathrm{E}[x_n])\mu^{(\tilde{m})},
$$

because setting to zero some normalized centered $\eta^{(m)}$ is equivalent to setting to zero the left-hand side of the equation above, from which we conclude that

$$
\mu^{(\bar{m})} \approx -\sum_{\sum\tilde{m}_i<\sum m_i} \hat{\gamma}_{\tilde{m}}(\mathrm{E}[x_1],\mathrm{E}[x_2],\ldots,\mathrm{E}[x_n])\mu^{(\tilde{m})}.
$$

A.5 Quasi-deterministic Normalized Centered moments

The uncentered moment $\mu^{(m)}$ can be expressed in terms of the quasi-deterministic normalized centered moment $\hat{\eta}^{(\bar{m})}$ as follows

$$
\begin{aligned}
\mu^{(m)} &= \mathrm{E}[x_1^{m_1}x_2^{m_2}\cdots x_n^{m_n}] \\
&= \phi_1^{m_1}\phi_2^{m_2}\cdots\phi_n^{m_n}\,\mathrm{E}\left[\left(1+\frac{x_1-\phi_1}{\phi_1}\right)^{m_1}\left(1+\frac{x_2-\phi_2}{\phi_2}\right)^{m_2}\cdots\left(1+\frac{x_n-\phi_n}{\phi_n}\right)^{m_n}\right] \\
&= \phi_1^{m_1}\phi_2^{m_2}\cdots\phi_n^{m_n}\left(1+\hat{\eta}^{(m)} + \sum_{2\le\sum\bar{m}_i<\sum m_i}\beta_{\bar{m}}\,\hat{\eta}^{(\bar{m})}\right),
\end{aligned}
$$

where the summation is over quasi-deterministic normalized centered moments $\hat{\eta}^{(\bar{m})}$ of order one[2] or larger and strictly smaller than $\sum m_i$; and the $\beta_{\bar{m}}$ are appropriately selected nonnegative constants.

Conversely, we can also express the any quasi-deterministic normalized centered moments $\hat{\eta}^{(\bar{m})}$ in terms of uncentered moment $\mu^{(\tilde{m})}$ as follows

$$
\begin{aligned}
\hat{\eta}^{(\bar{m})} &= \mathrm{E}\left[\left(\frac{x_1-\phi_1}{\phi_1}\right)^{\bar{m}_1}\left(\frac{x_2-\phi_2}{\phi_2}\right)^{\bar{m}_2}\cdots\left(\frac{x_n-\phi_n}{\phi_n}\right)^{\bar{m}_n}\right] \\
&= \mathrm{E}\left[\left(\frac{x_1}{\phi_1}-1\right)^{\bar{m}_1}\left(\frac{x_2}{\phi_2}-1\right)^{\bar{m}_2}\cdots\left(\frac{x_n}{\phi_n}-1\right)^{\bar{m}_n}\right] \\
&= \frac{\mu^{(\bar{m})}}{\phi_1^{m_1}\phi_2^{m_2}\cdots\phi_n^{\bar{m}_n}} + \sum_{\sum\tilde{m}_i<\sum m_i}\gamma_{\tilde{m}}\,\frac{\mu^{(\tilde{m})}}{\phi_1^{\tilde{m}_1}\phi_2^{\tilde{m}_2}\cdots\phi_n^{\tilde{m}_n}},
\end{aligned}
$$

where the summation is over uncentered moments $\mu^{(\tilde{m})}$ of order strictly smaller than $\sum\bar{m}_i$ and the $\gamma_{\tilde{m}}$ are appropriately selected constants.

The final formulas for quasi-deterministic moment closure can be obtained directly from expansions of the quasi-deterministic non-normalized centered moments

$$
\mathrm{E}\left[(x_1-\phi_1)^{\bar{m}_1}(x_2-\phi_2)^{\bar{m}_2}\cdots(x_n-\phi_n)^{\bar{m}_n}\right] = \mu^{(\bar{m})} + \sum_{\sum\tilde{m}_i<\sum m_i}\hat{\gamma}_{\tilde{m}}(\phi_1,\phi_2,\ldots,\phi_n)\mu^{(\tilde{m})},
$$

[2] Contrary to what happens for normalized centered moments, the quasi-deterministic normalized centered moments of order one are not necessarily zero.

because setting to zero some quasi-deterministic normalized centered $\hat{\eta}^{(m)}$ is equivalent to setting to zero the left-hand side of the equation above, from which we conclude that

$$\mu^{(\bar{m})} \approx - \sum_{\sum \tilde{m}_i < \sum m_i} \hat{\gamma}_{\tilde{m}}(\phi_1, \phi_2, \ldots, \phi_n) \mu^{(\tilde{m})}.$$

Bibliography

[1] Y. Cao, A. Hall, H. Li, and S. Lampoudi. *User's Guide for* StochKit. University of California, Santa Barbara. Available at http://www.engineering.ucsb.edu/~cse/StochKit.

[2] C. A. Gomez-Uribe and G. C. Verghese. Mass fluctuation kinetics: Capturing stochastic effects in systems of chemical reactions through coupled mean-variance computations. *J. of Chemical Physics*, 126(2):024109–024109–12, 2007.

[3] J. P. Hespanha. `StochDynTools` — a MATLAB toolbox to compute moment dynamics for stochastic networks of bio-chemical reactions. Available at http://www.ece.ucsb.edu/~hespanha/software, Dec. 2006.

[4] J. P. Hespanha and A. Singh. Stochastic models for chemically reacting systems using polynomial stochastic hybrid systems. *Int. J. on Robust Control*, Special Issue on Control at Small Scales: Issue 1, 15:669–689, Sept. 2005.

[5] A. Singh and J. P. Hespanha. Lognormal moment closures for biochemical reactions. In *Proc. of the 45th Conf. on Decision and Contr.*, Dec. 2006.

[6] N. G. Van Kampen. *Stochastic Processes in Physics and Chemistry*. Elsevier Science, 2001.

Index