UPTEC X 06 023
MAY 2006

ISSN 1401-2138

ANDREAS HELLANDER

Coupling the macroscales and mesoscales in the simulation of cellular reaction networks

Master’s degree project

UPPSALA UNIVERSITET

Molecular Biotechnology Programme

Uppsala University School of Engineering

UPTEC X 06 023	**Date of issue 2006-05**

Author

Andreas Hellander

Title (English)

Coupling the macroscales and mesoscales in the simulation of cellular reaction networks

Abstract

A hybrid solver for coupled macroscales and mesoscales has been implemented and its performance has been evaluated on a number of model systems. The approach relies on the splitting of the set of variables into one subset of approximately normally distributed species and one subset that needs a stochastic treatment.

Keywords

Hybrid method, master equation, macroscale, mesoscale, SSA, quasi-Monte Carlo integration

Supervisors

Per Lötstedt
Dept of information technology, Uppsala University

Scientific reviewer

Lina von Sydow
Dept of information technology, Uppsala University

Language	Security
English	
ISSN 1401-2138	Classification
Supplementary bibliographical information	Pages **43**

Biology Education Centre Biomedical Center Husargatan 3 Uppsala
Box 592 S-75124 Uppsala Tel +46 (0)18 4710000 Fax +46 (0)18 555217

Coupling the macroscales and mesoscales in the simulation of cellular reaction networks

Andreas Hellander

Sammanfattning

Kemiska reaktioner kan modelleras matematiskt genom att formulera ekvationer som beskriver hur medelvärdena av antalet molekyler för de olika molekylslagen förändras i tiden. Det fungerar bra när antalet molekyler av varje sort är stort och mängderna inte ändras för snabbt. Om reaktionerna sker inuti en cell kan de villkoren inte alltid uppfyllas. Ofta är kopietalet av vissa molekylslag lågt, vilket innebär att det gör en stor skillnad om enstaka molekyler omvandlas eller försvinner ur systemet. En modell som endast ser till medelvärden kommer därför inte att ge en bra bild av förloppet. En stokastisk modell baserad på den s.k. masterekvationen ger en bättre beskrivning, men kan vara i princip omöjlig att lösa numeriskt på grund av att beräkningsarbetet växer exponentiellt med antalet molekyler. I det här examensarbetet har en metod som behandlar vissa molekylslag med den enklare medelvärdesmodellen och andra med en stokastisk simuleringsalgoritm implementerats och utvärderats. Den här metoden är ett exempel på en s.k. hybridmetod och ger en mer korrekt beskrivning av reaktionerna än den helt deterministiska modellen, men är mindre krävande numeriskt än den helt stokastiska. Att utvecka snabbare algoritmer är en viktig del av beräkningsbiologin, eftersom man vill kunna studera större, mer komplicerade system, utan att för den skull vara tvungen att vänta alltför länge på resulaten.

Examensarbete 20p i Molekylär bioteknikprogrammet
Uppsala universitet maj 2006

Abstract

In this thesis, a hybrid solver for coupled macroscales and mesoscales has been implemented and its performance has been evaluated on some different model systems. The approach rely on the splitting of the set of variables into a subset of approximately normally distributed species, and a subset that needs a stochastic treatment. This gives a system of integro-differential equations for the expected values of the deterministic variables, which can be solved for given the probability distribution function of the other subset. The discrete stochastic variables have been simulated with Gillespie's exact stochastic simulation algorithm (SSA) and the ODE system has been solved with the second order backward differentiation formula (BDF-2) with variable coefficients. It has been shown that the hybrid solver can reduce the time needed to compute the solution if the number of reacting species is sufficiently large, while it is still able to capture important features of the fully stochastic models. Alternatively, the hybrid algorithm can be viewed upon as an improvement of the macroscopic model by adding stochasticity to some components. While being more computationally demanding than the pure ODE formulation, the hybrid solver gives more realistic results to a low additional cost.

Contents

1 Introduction

In recent years, large-scale investigations of e.g. genome sequences have been made possible, and the large amount of information generated has made the field of bioinformatics flourish. The main focus has been on the development of computational tools for data-mining and the prediction of different kind of properties based on huge data sets. This has led to many new insights, and the field is still developing rapidly. However, with our newfound knowledge, it has been realized that information concerning e.g. genome sequences or protein structures is not alone sufficient to understand the function of biological systems [15].

The processes in a living cell are controlled by biochemical reaction networks interacting in a complicated manner. For example, the circadian rhythm, which is responsible for the adaption of an organism to periodic variations in the environment, is controlled by such chemical systems giving rise to oscillations of certain molecular species. Obviously, the understanding of the underlying networks is essential in order to understand the phenomenon they give rise to on a system level. Characteristic of the control networks is the fact that they are generally difficult to understand based only on intuition [11], and to meet the need to analyze their behavior, the field of computational systems biology has emerged.

The development of techniques to quickly obtain interaction data on the proteome level has provided the research community with a wealth of information. So far it has shown that some proteins have many interaction partners, while other have just one or a few, leading to a general picture of cellular control systems as complex, branched networks composed of hubs and nodes [6, 19]. The dynamics of the systems is of great interest for the understanding of fundamental processes such as development and differentiation.

In the process of drug development, it is of importance to identify key regulatory elements and obtain a thorough understanding of what role different components play in the systems, both to be able to select good drug targets and to get an understanding of the wider consequences the manipulation of the levels of certain molecular species will have on the system . It is often hard and time consuming to approach these questions with traditional biochemical methods, and thus mathematical models could greatly simplify the analysis, at least by being able to suggest appropriate experiments in the lab.

In a well stirred system with macroscopic concentrations the coupled chemical reactions are often modeled by the reaction rate equations, a system of (generally) non-linear, coupled ordinary differential equations. In many cases, the macroscopic model provides a good description of the time evolution of the system. In the cell however, the underlying assumptions are often violated. At least some species are present in low copy numbers. For example, mRNA often exists in one or a few copies, while transcription factors may range from ten to hun-

dreds of molecules. Yet other components could be present in large numbers and approach macroscopic values. This imposes several problems for the modeler. First, the low copy number species are not well described with a deterministic model since they are subject to random fluctuations which can not be neglected and in many cases have a great impact on the behavior of the system [27, 30, 33]. Thus, a realistic model must take the inherent randomness into account and need therefore be of stochastic nature.

Second, the different scales in both the copy number and in time and reaction rates, give rise to computational difficulties. One way to model coupled chemical reactions stochastically is by the use of the (exact) Stochastic Simulation Algorithm (SSA) proposed by Gillespie [8]. This algorithm yield a correct realization of the process, but the time required to generate information about the probability distribution of the species in the system is often dictated by the reactions involving the molecules of largest copy numbers or fastest reaction rates, which may well be the components where the stochastic description is the least important. The convergence rate is also slow for this method and therefore it can be difficult to obtain detailed information of the distributions when the number of reacting molecules are large.

The underlying stochastic process is often assumed to be memory lacking, i.e. *Markovian*, and the time evolution of the process is described by a difference-differential equation, the (chemical) master equation. Unfortunately, there is no good way of solving this equation analytically for non-trivial problems. The problem suffers from the curse of dimensionality as the computational work grows exponentially with the number of dimensions (number of reacting species). Consequently, this often limits the complexity of the models to three or maybe four dimensions.

Some different ways to determine the solution, either by approximations of the master equation [31] or dimension reduction by making assumptions about the behavior of different components [12, 35] have been investigated. In the first case, the master equation is approximated by the Fokker-Planck equation, a partial differential equation derived from a truncated Taylor expansion of the master equation. The Fokker-Planck equation is easier to solve than the master equation, but it is still limited by an unfavorable exponential growth in computational time with increasing number of species. The other approach rely on some previous knowledge of the system in order to reduce the dimension of the problem. While this can result in a considerable reduction of the complexity, a profound knowledge of the biological system is required to make good assumptions.

Since there are efficient numerical methods to solve ordinary differential equations, it would be a great simplification if some components could be modeled by deterministic equations, and other components treated with a stochastic model. Such *hybrid* models are promising alternatives to fully stochastic models, and some attempts have been made to implement this kind of solvers [12, 29].

Lötstedt and Ferm, 2005 [21] suggested a separation of the components into

a subset of variables that can be treated as normally distributed with small variance and a subset of variables that need a stochastic treatment. They derive equations for the expected values of the first subset which can be solved for given the probability distribution of the stochastic variables. The intent of this thesis was to implement and evaluate this approach by comparing the results to those of a fully stochastic description of the model systems.

The remainder of this thesis is organized as follows. First, the underlying theory of the modeling of coupled chemical reactions and the system decomposition is discussed and the main computational demands are introduced. Second, the theory of the numerical methods used in the implementation is reviewed and the structure of the implementation is presented. The hybrid solver is then applied to a couple of model systems and the results are evaluated and compared to the results from simulations with SSA. Finally, some conclusions are made regarding the efficiency of this implementation and some extensions and improvements are suggested.

2 Background

2.1 Modeling chemical reactions

2.1.1 Macroscopic and mesoscopic views

Throughout this thesis it will be necessary to make a distinction between the *macroscopic* and the *mesoscopic* view. Macroscopically, we assume that the chemical species are present in large copy numbers and formulate a description for the expected values of the species. Obviously, each chemical reaction means a discrete jump in the number of molecules, but as the numbers are large this does not have a great impact on the mean values. In a mesoscopic model some molecular species are often present in small copy numbers, and thus these discrete jumps can change the state of the system profoundly. Therefore we have to consider each molecule individually in these models, and a stochastic description is nesscesary. *Meso* means 'between' and here refers to the fact that the mesoscopic description lies between macroscopic and microscopic (molecular dynamics, quantum mechanics) models.

2.1.2 The Macroscopic view

The purpose of this section is to introduce some basic notation and quickly review how chemical reactions are described and how the deterministic equations are formulated. Consider a general chemical reaction where reactants $y_i,\ i = 1 \dots n$ interact to form products $x_j,\ j = 1 \dots m$:

$$a_1 y_1 + a_2 y_2 + \dots a_n y_n \xrightarrow{k} b_1 x_1 + b_2 x_2 + \dots b_m x_m.$$

The constants a_i and b_j are stoichiometric coefficients and dictate how many equivalents of respective molecular species that are consumed and formed in the reaction. The constant k is a *rate constant*. This name is somewhat misleading, since it is not really a constant but depend for example on variables such as temperature. From the above expression it is possible to formulate a *rate law* for the reaction. This can be done in different ways depending on what kind of molecules we model (e.g. reactions involving enzymes can have different expressions than reactions without). We will see examples of some different rate laws later in this thesis. In a sense, all reactions are *reversible*, meaning that the reaction can proceed both to the right and to the left. For our purpose however, it will be convenient to divide these reactions in separate forward and backward reactions with different rate constants.

At the macroscale we can generally describe the time evolution of the concentrations of the reacting species accurately with a set of deterministic equations. If we let $\mathbf{y}$ be the state vector consisting of the n reacting species $y_i,\ i = 1 \dots n$, we can write the familiar *convection-diffusion* equation:

$$\frac{\partial[\mathbf{y}]}{\partial t} + (\mathbf{v} \cdot \nabla)\mathbf{y} - \nabla \cdot (D\nabla\mathbf{y}) = R_i(\mathbf{y}, t).$$

This equation describes the change in concentration of the components at a given point in space and time, when the changes are due to both diffusion and convection as well as chemical reactions. For many problems in e.g. chemical engineering, this equation can provide a good approximation of the time evolution of the different reacting molecules, and good solvers for these kind of problems are avaliable.

In a well stirred system we can assume spatial homogeneity and are left with the reaction term. This leaves us with the system of ODEs known as the reaction rate equations

$$\frac{\partial[y_i]}{\partial t} = -\sum_{r=1}^{R} n_{ri}\omega_r(\mathbf{y}, t), \quad i = 1, \dots, n. \tag{1}$$

Here, n_{ri} are stochiometric coefficients that tell if species y_i is formed, consumed (and how many) or unaffected by the reaction r (compare to the coefficients a_i and b_j above). $\omega_r(\mathbf{y}, t)$ is the reaction propensity related to the reaction r, and is generally a function of the whole state vector. There is no recipe of how to formulate the rate laws, but they will obey the law of mass action. To better understand how the reaction rate equations are constructed, we consider another example. This also serves to introduce one of the test system used later on. This system will be referred to as "coupled flows", and is an idealized description of the flow of two metabolites under enzymatic control. It has been used as model system in earlier work, e.g. [32, 36]. We consider two metabolites, A,B, that react to form a third species in which we take no interest (the metabolites can be

e.g. different amino acids, consumed in protein synthesis on the ribosome). The formation of A and B is controlled by two different enzyme systems, E_1 and E_2, and their action is represented as a single enzymatic step. Further, the enzymes are controlled by feedback inhibition, meaning that as the metabolite pool grows, the enzyme available for synthesis is inhibited, and thus the influx of metabolites decreases. The set of reactions describing this scenario is

$$\begin{aligned}
&\emptyset \xrightarrow{\frac{k_a[E_1]}{1+\frac{[A]}{K_i}}} A \quad \emptyset \xrightarrow{\frac{k_b[E_1]}{1+\frac{[B]}{K_i}}} B,\\
&A \xrightarrow{\mu} \emptyset \quad B \xrightarrow{\mu} \emptyset,\\
&A + B \xrightarrow{k_2} \emptyset,\\
&\emptyset \xrightarrow{\frac{k_{ea}}{1+\frac{[A]}{K_r}}} E_1 \quad \emptyset \xrightarrow{\frac{k_{eb}}{1+\frac{[B]}{K_r}}} E_2,\\
&E_1 \xrightarrow{\mu} \emptyset \quad E_2 \xrightarrow{\mu} \emptyset,
\end{aligned}$$

where expressions in brackets denote concentrations of the respective component. For the system above the reaction rate equations are given by:

$$\begin{cases}
\frac{d[A]}{dt} &= \frac{k_a[E_1]}{1+\frac{[A]}{K_i}} - k_2[A][B] - \mu[A],\\
\frac{d[B]}{dt} &= \frac{k_a[E_2]}{1+\frac{[B]}{K_i}} - k_2[A][B] - \mu[B],\\
\frac{d[E_1]}{dt} &= \frac{k_{ea}}{1+\frac{[A]}{K_r}} - \mu[E_1],\\
\frac{d[E_2]}{dt} &= \frac{k_{eb}}{1+\frac{[B]}{K_r}} - \mu[E_2],
\end{cases}$$

In this case, the second order reaction in which the metabolites are consumed is described with a second order rate law, but in the general case, all reaction propensities could be any function $\omega(\mathbf{y}, t)$.

2.1.3 The mesoscopic view: Markov processes and the master equation

This section follows [10, 34] where thorough reviews of the general properties of Markov processes are given. Consider a molecule that can undergo the following one-step reaction:

$$X \longrightarrow Y$$

Assume that we know the state of the molecule at some time t. Now consider the probability that the molecule is still in state X at time $t + \delta t$. This probability is the same as the probability that the molecule was in state X at time t minus the probability that it has made the transition to state Y in the time interval given that it was in state X at time t. I.e.

$$P(X, t+\delta t) = P(X,t) - P_{XY}(t, t+\delta t),$$

by the law of total probability

$$P_{XY}(t, t+\delta t) = P_{XY}(t, t+\delta t|X, t)P(X,t)$$

We can now write

$$P(X, t+\delta t) - P(X,t) = -P_{XY}(t, t+\delta t|X, t)P(X,t).$$

This gives a differential equation for $P(X,t)$ if we divide by δt and take the limit as $\delta t \to 0$

$$\frac{dP(X,t)}{dt} = -\lim_{\delta t \to 0} \frac{P_{XY}(t, t+\delta t|X,t)}{\delta t} P(X,t). \tag{2}$$

Equation (2) is a master equation for the simple one step reaction. Up until now, no assumptions have been made about the stochastic process. The key to the behavior lies in the transition probability per time unit

$$\lim_{\delta t \to 0} \frac{P_{XY}(t, t+\delta t|X,t)}{\delta t}$$

which generally can be any time dependent function. From here on we will make some assumptions regarding the nature of the process. Let $Z(t)$ be a stochastic process for which $Z(t_0) = z_0$. To the n first random variables $Z(t_n)$ we can formulate a joint probability density

$$P_n^{(1)}(z_n, t_n|z_{n-1}, t_{n-1}; \ldots; z_0, t_0) = Prob\{Z(t_i) \in [z_i, z_i + dz_i), i = 1, \ldots, n \text{ given} \quad Z(t_0) = z_0\}$$

Let us consider a subclass of such stochastic process for which

$$P_1^{(j)}(z_j, t_j|z_{j-1}, t_{j-1}; \ldots; z_0, t_0) = P(z_j, t_j|z_{j-1}, t_{j-1}).$$

That is to say that the next state of the process only depends on the current state, and no information of values of the stochastic variables at previous times is needed. Processes with this property are said to be Markovian [10]. The memory lacking property is typical of Markov processes, and make them easier to handle than more general processes. For a stochastic process to be Markovian, the probability density needs to obey the Chapman-Kolmogorov equation

$$P(z_3, t_3|z_1, t_1) = \int_{-\infty}^{\infty} P(z_3, t_3|z_2, t_2)P(z_2, t_2|z_1, t_1)dz_2 \tag{3}$$

which can be viewed upon as a consistency condition on the density function [34].

If the process has these properties, the transition probability in (2) is constant, that is

$$\frac{dP(X,t)}{dt} = -kP(X,t)$$

and this is a great simplification. For this one-step process we can easily obtain the analytic solution to the master equation, but this is usually not the case. For a general Markov process, the master equation describes the probability flow to and from a state at a given time and can be written

$$\frac{\partial p(\mathbf{x},t)}{\partial t} = \sum_{r=1}^{R}(\omega_r(\hat{\mathbf{x}},t)p(\hat{\mathbf{x}},t) - \omega_r(\mathbf{x},t)p(\mathbf{x},t)) \tag{4}$$

where $\hat{\mathbf{x}}$ denote values of $\mathbf{x}$ after a shift associated with the reaction r. Even though it is theoretically possible to solve this equation numerically in this form, it is infeasible for more than a few molecular species.

2.2 Dimension reduction

This section follows Lötstedt and Ferm, 2005 [21], where formal proofs are given to the results presented here. Consider a set of chemical species $X_i, i = 1\ldots, n$, taking part in the reactions $R_j, j = 1\ldots, r$ with reaction propensities $\omega_j(\mathbf{x}, t)$. We denote the number of molecules of species X_i by x_i. If we now make the assumption that a subset $Y_i, i = 1\ldots, m$, of the reacting molecules are present in relatively large copy numbers and vary slowly, that is, they are assumed to be approximately normally distributed with a small variance, we obtain a partitioned set of variables $\mathbf{x} \to (\mathbf{x}', \mathbf{y})$ where the dimension of $\mathbf{x}'$ is $n_X = n - m$ (from here on the set of stochastically treated variables in the partitioned set will be denoted $\mathbf{x}$, dropping the prime for convenience). Furthermore, we make the assumption that the two sets of stochastic variables are independent. Then the probability distribution function can be written

$$p(\mathbf{x}, \mathbf{y}, t) = p_X(\mathbf{x}, t)p_Y(\mathbf{y}, t)$$

If the variables Y_i are independent, the density function $p_Y(\mathbf{y}, t)$ has the form

$$p_Y(\mathbf{y}, t) = \gamma_m \exp(-\sum_{j=1}^{m} \frac{(y_j - \bar{y}_j)^2}{2\sigma^2}),$$

where γ_m is a normalizing constant and $\bar{y_j}(t)$ is the mean value of the chemical species Y_j and σ is the standard deviation. The expected values $\bar{y}_j(t)$ satisfy

$$\frac{d\bar{y}_j}{dt} = -\sum_{r=1}^{R} n_{rj} \sum_{\mathbb{Z}_+^m} w_r(\mathbf{x} + \mathbf{m}_r, \bar{\mathbf{y}}(t), t)p_0(\mathbf{x} + \mathbf{m}_r, t) \quad j = 1\ldots, m \tag{5}$$

where $\mathbf{m}_r$ is the shift in $\mathbf{x}$ associated with reaction r and $p_0(\mathbf{x}+\mathbf{m}_r, t)$ is the marginal probability distribution function (PDF). In [21] it was shown with numerical experiments that the shift $\mathbf{m}_r$ can be neglected for practical purposes, and this has been done in the implementation of the hybrid solver. The marginal PDF satisfies the master equation (4), and it is given by

$$p_0(\mathbf{x}, t) = \int p(\mathbf{x}, \mathbf{y}, t)\, d\mathbf{y} = \gamma_m p_X(\mathbf{x}, t) \int \exp(-\sum_{j=1}^{m} \frac{(y_j - \bar{y}_j)^2}{2\sigma^2})\, d\mathbf{y}.$$

The normalizing constant γ_m satisfies

$$\gamma_m \int \exp(-\sum_{j=1}^{m} \frac{(y_j - \bar{y}_j)^2}{2\sigma^2})\, d\mathbf{y} = 1,$$

therefore

$$p_0(\mathbf{x}, t) = p_X(\mathbf{x}, t).$$

When the assumptions made in the derivation of these equations hold, the original problem of solving the master equation in n dimensions is reduced to solving a master equation in $n - m = n_X$ dimensions for $p_0(\mathbf{x}, t)$ and m ordinary differential equations (5) for the expected values of the remaining species Y_j. Obviously, this approach does not provide any information of the distributions for species Y_j, other than the time dependent mean. However, the underlying assumption was that they have a normal distribution with small variance, σ^2.

The equation for the remaining n_X stochastic variables will still need to be solved. In [5] the marginal PDF is successfully solved for using the Fokker-Planck approximation and the applicability of the method is evaluated by the solution of the Vilar oscillator [35]. However, as already discussed the Fokker-Planck equation is still numerically difficult in many dimensions, and consequently n_X must be kept small.

Another way of obtaining an approximation of $p_0(\mathbf{x}, t)$ is by exact simulation of the Markov process using the SSA algorithm [8], which is the approach taken in this thesis.

2.3 Coupling the macro and mesoscales

From the previous section we conclude that a solution of the system of equations (5) will require both the approximation of the marginal PDF and the solution of the system of ordinary differential equations. This in turn requires the evaluation of a sum of dimension n_X. As will be discussed in more detail later, the sum needs to be evaluated both as the right hand side of the system of equations, and at least once for each element of the Jacobian matrix required to solve the non-linear system in each time step. The requirement for the hybrid approach

to be successful is that this can be done considerably faster than the stochastic simulation of the full system.

There are numerous methods for numerical integration, but one of the most generally applicable is the use of Monte Carlo techniques [2]. In the implementation of the hybrid solver the summation is performed with an algorithm using quasirandom numbers.

2.4 Exact Stochastic Simulation Algorithm (SSA)

The SSA algorithm has long been a popular method to simulate chemical reactions. Even though some effort has been made in recent years to improve the performance of the algorithm [7, 14], it is still widely used due to its conceptual simplicity and the ease of the implementation. In the original paper [8], two mathematically equivalent formulations were presented, the *first reaction method* and the *direct method*. In the first reaction method, all possible events are assigned a reaction time sampled from a exponential distribution, and the reaction with the shortest reaction time is executed. This requires that the same amount of pseudorandom numbers as the number of possible reactions is generated in each time step. Therefore this method is quite expensive and hardly ever used in actual implementations.

The direct method on the other hand, needs only two random numbers and is thus more efficient. The algorithm can be outlined as follows. First initialize, that is compute the reaction propensities for each reaction from the initial values. Then generate a pair of random numbers (τ, μ) from the joint probability distribution function for the reaction time and the reactions

$$P(\tau, \mu) = \omega_\mu(\mathbf{x}, t) e^{-\sum_{\mu=1}^{r} \omega_\mu(\mathbf{x},t)\tau}. \tag{6}$$

This is usually done with the inversion method. First, τ can be chosen by sampling a pseudorandom number z_1 from the uniform distribution and taking

$$\tau = (\sum_{\mu=1}^{r} \omega_\mu(\mathbf{x}, t))^{-1} \ln(1/z_1)$$

We can then form a random integer μ that tells which reaction to execute at time τ by sampling another uniform number z_2 and taking μ to be the number for which

$$\sum_{v=1}^{\mu-1} \omega_v(\mathbf{x}, t) < z_2 \sum_{\mu=1}^{r} \omega_\mu(\mathbf{x}, t) \leq \sum_{v=1}^{\mu} \omega_v(\mathbf{x}, t)$$

Now advance the time by τ and execute the reaction according to μ. Then recompute the reaction propensities ω_μ and repeat until the desired time is reached.

This procedure of course, only give one possible outcome of the stochastic process. However, we can obtain an approximation of the probability density by using the information from many independent trajectories starting with the same initial values.

We can note some things characteristic of the Gillespie algorithm. If we let the number of reactions be R, the work is proportional to cR, where c is some constant that can be large for some systems. Many reactions and large reaction propensities leads to small values of τ, and this in turn means that many steps have to be taken in order to arrive at the final time. Furthermore, when it is used to form the probability distribution, several independent trajectories need to be sampled. As for other Monte Carlo methods, the error of the distribution decreases as $M^{-1/2}$, where M is the number of realizations.

3 Numerical methods

3.1 Computation of the marginal PDF

The first step of the time stepping scheme will be the simulation of the stochastic variables in order to obtain an approximation of the marginal PDF. If several independent trajectories are simulated up to the same time t, the value of the PDF in a certain point in the state space can be approximated by the ratio between the number of trajectories with just that value and the total number of trajectories. The values of the trajectories are stored as rows in a $M \times (n_X + 1)$ matrix $\mathbf{T_M}$

$$\mathbf{T_M}^{(n)} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1n_X} & t_1 \\ x_{21} & x_{22} & \dots & x_{2n_X} & t_2 \\ \vdots & \vdots & & \vdots & \vdots \\ x_{M1} & x_{M2} & \dots & x_{Mn_X} & t_M \end{pmatrix}^{(n)}_{M\times(n_X+1)}$$

and updated each time step by SSA. M is the total number of trajectories and is typically large. The last column holds the individual times of each trajectory. Since the time steps in SSA are chosen randomly, the value t_i of trajectory i will not exactly match the desired time at t^{n+1}, thus it is important to keep track of the individual times of the trajectories. Also, the states are only updated in SSA if the present time $t_i < t^{n+1}$. If the state of a trajectory i at time t is denoted $\hat{\mathbf{x}}_i(t) = \{x_{i1}\ x_{i2}\ \dots x_{in_X}\}$, the value of the distribution at point $\mathbf{x}$ is computed as

$$p_0(\mathbf{x}, t) = \frac{1}{M}\sum_{i=1}^{M} w_i(\mathbf{x}, t),$$

$$w_i(\mathbf{x}, t) = \begin{cases} 1 & \text{if } \hat{\mathbf{x}}_i(t) = \mathbf{x} \\ 0 & \text{otherwise} \end{cases}$$

The mean values of the individual components can of course be computed in the same spirit. The matrix $\mathbf{T_M}$ is stored and maintained in the main routine written in MATLAB, and submitted as a parameter to the C-routines that perform SSA and evaluates the distribution.

Even though the state space needs to be truncated, it can still be very large. One realizes that computing and storing the value of p_0 in every discrete point will be very expensive. However, this is not necessary, since the reason why we need the PDF is to evaluate the sum in the right hand side of (5). Since this will be done with a Monte Carlo method, we know the points where the values are needed in advance. Thus, we only need to evaluate p_0 in those points, and this will reduce the work and storage requirements tremendously if the dimension is large.

This means that to evaluate the PDF we need to find the entries in the matrix that correspond to the quadrature points $\mathbf{x}$. By applying a sorting algorithm to the matrix, followed by a binary search algorithm this can be done efficiently. In the implementation, a sorting routine provided by MATLAB (`sortrows`) that implements a stable quicksort algorithm has been used. It sorts the rows of the trajectory matrix in ascending order with respect to the columns (first the matrix is sorted according to the leftmost column, then it proceeds from left to right). The binary search has been implemented in C, and simply finds the row in the matrix that correspond to the quadrature point. To avoid unnesscesary work, trajectories with the same value (identical rows) are first removed from the original matrix before the searching algorithm is applied. This is done with the MATLAB routine `unique`.

It is necessary to discuss the error of the PDF. When approximating of the sum of (5), a number of quasirandom sequences are used. This will be described in more detail in section 3.2. For each quadrature point of these sequences, $p_0(\mathbf{x}, t)$ is evaluated. We let a quasirandom sequence for the summation be denoted $S_j, j = 1, \dots n$. Further, denote the elements of S_j by $X_i(j), i = 1, \dots, N$, where N is the length of S_j (i.e. the number of evaluation points). If the trajectory matrix T_M is divided into m submatrices, $T_k, k = 1, \dots, m$ of length $\hat{M}$, $M = m\hat{M}$, the variance of $p_0(\mathbf{x}_i(j), t)$, $\sigma^2_{x_i(j)}$, can be computed as

$$\sigma^2_{x_i(j)} = \frac{1}{m-1} \sum_{k=1}^{m} (P^{(k)}_{x_i(j)} - \bar{P}_{x_i(j)})^2,$$

$$P_{x_i(j)} = \frac{1}{m} \sum_{k=1}^{m} P^{(k)}_{x_i(j)}.$$

This means that the error of the pooled estimate is bounded by (by Student's t-distribution)

$$|P^*_{x_i(j)} - \bar{P}_{x_i(j)}| \leq \frac{1.96\sigma_{x_i(j)}}{\sqrt{m}}$$

within a 95% confidence interval. In the implemetation of the hybrid solver, the error in p is measured as

$$\epsilon_P = \frac{1}{n}\sum_{j=1}^{n} \|\sigma_{x_i(j)}\|_\infty$$

i.e. as the mean value based on the n different QMC sequences of the maximum absolute standard deviation over all evaluated quadrature points N. This measure decreases as $\frac{1}{\sqrt{M}}$ with increasing number of trajectories. It has been seen that $\sigma_{x_i(j)}$ varies little with j. Within the present framework it is not possible to adjust the number of trajectories to meet some predefined error tolerance. However, for the test systems considered later, M in the range $10^5 - 10^6$ seem to be suitable values with respect to the error.

3.2 Monte Carlo integration

From the formulation of the coupled ordinary differential equations for the expected values it is evident that the evaluation of the right hand side will be very expensive. Considering also the fact that solving the system of equations in every time step involves evaluating the Jacobian in which every element demands at least one additional computation of the integral in the case of a forward difference approximation of the derivatives, makes this step the main bottleneck in the solution of the system. This will be true at least for moderate dimensions.

A viable option when facing multidimensional integrals is the use of Monte Carlo methods. The simple Monte Carlo estimator of an integral is:

$$\int_{[0,1]^d} f(\mathbf{x})d\mathbf{x} \approx \frac{1}{N}\sum_{i=1}^{N} f(\mathbf{x}_i) \tag{7}$$

where x_i are chosen from the uniform distribution. By independent replication the root mean square error (RMSE) of the monte Carlo quadrature can easily be computed [2]. By the central limit theorem the mean of the errors of these estimates are normally distributed, and the RMSE is an unbiased estimate of the standard deviation. Evidently, there are different options when reporting the error of the estimator (7), but the RMSE is a common indicator of the error

$$\begin{aligned} \epsilon^N_{RMSE} &= \sqrt{\tfrac{1}{M-1}\textstyle\sum_{j=1}^{M}(I_j^N - \hat{I}_N)^2} \\ \hat{I}_N &= \tfrac{1}{M}\textstyle\sum_{j=1}^{M} I_j^N \end{aligned} \tag{8}$$

where M is the number of independent replications of estimates I_j^N according to (7). The total number of points used is MN, and in the case of pseudorandom numbers the error could as well be estimated by the partitioning of one single sequence of length MN. However, we will be using quasirandom sequences, and

for practical reasons it is more convenient to use M different sequences of length N in the implementation of the hybrid solver. It is a well known fact that the convergence rate of (7) is of order $N^{-1/2}$. This can be shown using the central limit theorem, and a partial proof is given in [2].

In our case, we are dealing with a sum rather than an integral, but we apply the techniques of Monte Carlo integration to evaluate it. The remainder of this section will discuss some of the theory regarding Monte Carlo quadrature in terms of integrals, but we need to keep in mind that in the hybrid solver it is actually a summation as in (5) that we carry out. The corresponding integral is on the form

$$\int_{[0,1]^d} f(\mathbf{x})p(\mathbf{x},t)d\mathbf{x}.$$

To evaluate it with Monte Carlo (MC) integration we have two obvious options. We can use the estimator (7) in a plain (raw) MC integration with the integrand being the product $f(\mathbf{x})p(\mathbf{x},t)$, or we can notice that the problem is equivalent with the computation of the expected value of $f(\mathbf{x})$ when $\mathbf{x}$ is distributed according to $p(\mathbf{x},t)$. In either case, we can use (7) to approximate the value of the integral, and 8 to estimate the error. The difference lies in that while we in the first case use points sampled from the uniform distribution, the second approach require the sampling of points from some unknown probability distribution, $p_0(\mathbf{x},t)$. In the scheme we shall develop, this distribution will however be simulated by the Gillespie algorithm, and is therefore available.

3.2.1 The rejection method

One way of sampling from the distribution $p(\mathbf{x},t)$ is to use the *acceptance-rejection* method. This method is described in e.g. [2]. Consider a function $f(x)$ with the property $f(x) \leq p(x) \quad , \forall x$. Assume also that we have a way of normalizing the function $f(x)$ so that $\hat{f}(x) = f(x)/\int f(x)dx$. Then the following procedure yields random samples form the distribution $p(x)$: Sample x from the density $\hat{f}(x)$ and a trial variable y from the uniform distribution. Accept x if $0 \leq y \leq p(x)/\hat{f}(x)$, otherwise reject and try another x. Repeat until the desired amount of sample points is obtained. The closer $f(x)$ is to $p(x)$, the more efficient the sampling procedure will be, but a method to generate samples from $\hat{f}(x)$ is needed, and therefore in practice $f(x)$ is often chosen to be a constant. The efficiency of the sampling method depends on the ratio of accepted to rejected trial points. Therefore, it is crucial to obtain as tight an upper bound on $p(x)$ as possible. In practice, this can be difficult, especially when the distribution is not evaluated in every possible point in the state space.

The major drawback with regular Monte Carlo integration as in (7) is the slow rate of convergence $\mathcal{O}(N^{-1/2})$. There are a number of different variance reduction

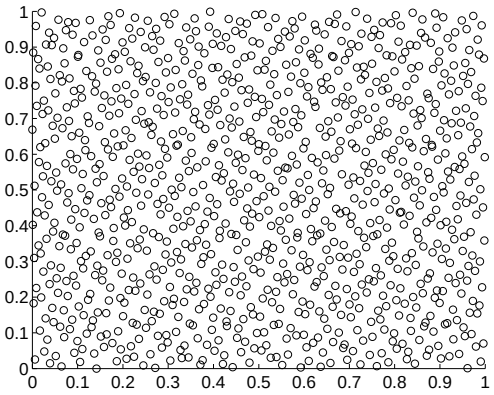

Fig. 1: Two dimensional scrambled Faure quasirandom sequence. The sequence was generated with algorithm 823 in [13] scrambling 10 digits.

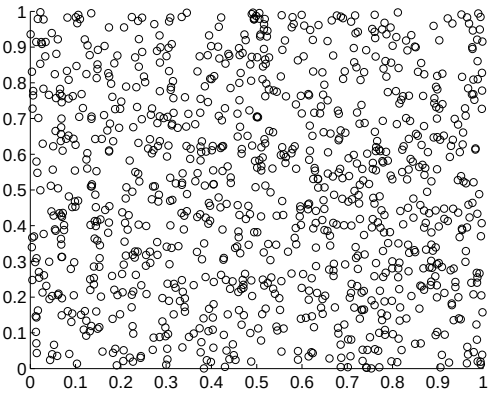

Fig. 2: Pseudorandom sequence. Note the areas with high and low density of points.

techniques such as importance sampling, stratification and control variates, but they only affect the constant, not the convergence rate in N.

3.2.2 Quasi-Monte Carlo sequences

Pseudorandom numbers have a tendency to aggregate, so that the sequence displays clusters and gaps [24]. One could expect the error to be smaller if the sample points were more evenly spread in the integration domain. Quasi-Monte Carlo (QMC) point sets are deterministic sequences with the property that they in some sense minimize the discrepancy, that is, they are more evenly distributed than their pseudorandom counterparts. QMC methods have been extensively used e.g. in the field of computational finance and have been shown to outperform Monte Carlo methods for many problems [16]. The difference between pseudorandom and quasirandom point sets is visualized in Fig. 1 and Fig. 2, where the clustering of the pseudorandom point set is evident.

Many different constructs of low discrepancy sequences have been proposed, and the implementation made in this thesis have used the sequence of Faure [3]. For details concerning the generation of the sequences and implementational details see e.g. [13, 1]. It can be shown that for a QMC quadrature rule the Kokshma-Hlawka inequality provides an upper bound of the error. For a quasirandom sequence $X_1, X_2, \ldots, X_N$ it states

$$|\hat{I}_N - I| \leq D_n^* V_{HK}(f) \tag{9}$$

where D_n^* is the star discrepancy of the sequence (a measure of how equidistributed the points are) and V_{HK} is the total variation of the integrand in the sense of Hardy and Krause [2].

The practical outcome of this is that for the best sequences, the theoretical convergence rate can be $\mathcal{O}(\frac{(\log N)^d)}{N})$ where d is the dimension of the integral, which is a considerable improvement over the simple Monte Carlo estimator at least for small d and large N. Even if this bound on the error is valid if V_{HK} is bounded, it is impractical. In practice, both the star discrepancy and V_{HK} is computationally infeasible, in fact, it can be as difficult to compute them as the computation of the integral, and furthermore it often results in overestimations of the error [2, 25]. To overcome the difficulty of error estimation, techniques to generate scrambled nets have been developed [13, 26, 4]. A scrambled quasirandom sequence is obtained if the points in the underlying sequence is shifted in such way that the low discrepancy property is conserved. The type of scrambling used in the hybrid solver (and the generation of data in Figs. 1 to 3) is a scrambling due to Owen, 1998 [26]. As for pseudorandom sequences, the scrambled sequences are independent samples from the uniform distribution. The use of such randomized sequences can be seen as a hybrid of QMC and regular Monte Carlo integration, having the superior convergence properties QMC methods while allowing for simple error estimation according to (8).

In Fig. 3 we see a comparison of the computed relative RMSE obtained by evaluating the integral

$$\int_{[0,1]^3} (x^2 + y^2 + z^2) e^{-\frac{(x^2+y^2+z^2)}{2}} dV$$

using the randomized Faure sequence and pseudorandom numbers. No variance reduction technique has been applied, and the points are sampled form the uniform distribution in a plain Monte Carlo method. The number in parentheses in the legend is the convergence rate. 20 sequences were used to compute the RMSE, i.e. $M = 20$ in (8). The QMC method yields much lower error than the pseudorandom method, and the convergence rate of the former is superior.

3.2.3 Smoothing techniques

The rejection method has been seen to have an undesirable effect on the convergence rate of QMC integration [2, 18]. This can be understood if we rewrite the integral

$$\int_0^1 f(x)p(x)dx = \int_0^1 \int_0^1 f(x)\chi(y < p(x))dydx$$

where $\chi(y < p(x))$ is the characteristic function

$$\chi(y < p(x)) = \begin{cases} 1 & \text{if } 0 < y < p(x) \\ 0 & \text{if } y > p(x) \end{cases}$$

This function if discontinuous, and does not have a bounded variation in the sense of Hardy and Krause. Therefore, the Kokshma-Hlawka inequality can

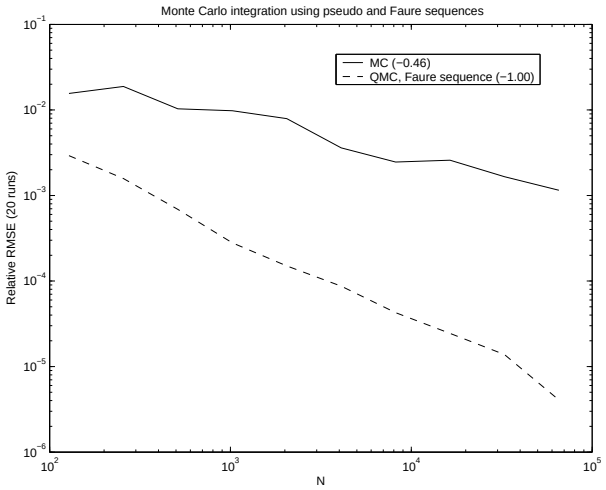

Fig. 3: The relative RMSE using pseudorandom numbers and the Faure sequence. Typically, the error of the QMC method is much smaller, and has a convergence rate of $\mathcal{O}(N^{-1})$ compared to $\mathcal{O}(N^{-1/2})$ for the plain Monte Carlo integration.

not be used to obtain a bound on the error and convergence rates are often lower than $\mathcal{O}(N^{-1})$ [18]. In our case, one additional complication is that the probability density $p(\mathbf{x}, t)$ is defined for discrete $\mathbf{x}$, and is therefore not smooth. This is expected to have consequences on the performance of the quadrature.

One way to overcome the problems with discontinuous characteristic functions is to replace them with continuous or differentiable weight functions. This approach is called smoothed rejection sampling. In this work, some different smoothing techniques have been considered. In [18] a differentiable B-splines smoothed characteristic function is suggested.

$$q(\mathbf{x}, y) = \begin{cases} 1 & \text{if } 0 \leq y < a - 3h/2, \\ \frac{(a-y+h)-(\frac{(a-y-h/2)^2}{2h})}{2h} & \text{if } a - 3h/2 \leq y < a - h/2, \\ \frac{a-y+h}{2h} & \text{if } a - h/2 \leq y < a + h/2, \\ \frac{(a-y+3h/2)^2}{4h^h} & \text{if } a + h/2 \leq y < a + 3h/2, \\ 0 & \text{if } a + 3h/2 \leq y \leq 1, \end{cases}$$

$$a = \gamma^{-1} p(\mathbf{x}), \quad \gamma \leq \sup_{\mathbf{x} \in \Omega^d} p(\mathbf{x}).$$

Even though this sampling procedure is shown to improve the convergence rate, the RMSE is often lager than the corresponding error obtained with pure rejection sampling for the test integrals evaluated in [18]. Here $2h$ is referred to as the smoothing width and must be chosen manually.

In [23] a linearly smoothed rejection sampling is proposed, and is shown to improve the convergence for some classical test integrals. In this case the weight function is

$$q(\mathbf{x}, y) = \begin{cases} 1 & \text{if } y < p(\mathbf{x}) - h \\ 0 & \text{if } y > p(\mathbf{x}) + h \\ \text{linear} & \text{if } p(\mathbf{x}) - h < y < p(\mathbf{x}) + h \end{cases}$$

Another alternative is to use weighted uniform sampling. In this method, the decision process is eliminated altogether, and instead every point is assigned a weight equal to the acceptance probability

$$q_i = \gamma^{-1} p(\mathbf{x}_i, t).$$

The the value of the sum is approximated as

$$\hat{I}_N = \frac{1}{\sum_{i=1}^{N} q_i} \sum_{i=1}^{N} q_i f(\mathbf{x}_i).$$

This method is biased, but the bias is normally small compared to the RMSE [23].

3.3 Time discretization

The time discretization has been made with the second order backward differentiation formula (BDF-2) with variable coefficients [22]. This is an implicit scheme, which is desirable since we would like to be able to take as large time steps as possible without instability in order to gain more advantage compared to the full simulation algorithm, SSA. The implementation has been made with the possibility of time adaptivity, but as we will see later, sometimes a fixed time step is preferable due to the large cost of recomputing the right hand side of equations (5).

The solver was implemented as a predictor-corrector pair where the corrector is iterated to convergence with the Newton method. The Jacobian is computed once initially in each time step, and recomputed only if the convergence criterion has not been met within some fixed number of iterations. Due to the large cost of evaluating the sum the elements of the Jacobian are approximated with first order forward differences. In such way, only one evaluation is needed for each element plus one additional evaluation which is the same for every entry. The linear system is solved with a direct method, but since the main part of the routine is written in Matlab, using different iterative solvers would be possible without changing much in the code. The explicit predictor is [22]

$$\hat{\alpha}_0^n \hat{\mathbf{y}}^{n+1} = \Delta t^n \bar{\omega}(\hat{\mathbf{y}}^n, t) - \hat{\alpha}_1^n \hat{\mathbf{y}}^n - \hat{\alpha}_2^n \hat{\mathbf{y}}^{n-1}, \tag{10}$$

$$\hat{\alpha}_0^n = 1/(1+\theta^n), \quad \hat{\alpha}_1^n = \theta^n - 1, \quad \hat{\alpha}_2^n = -(\theta^n)^2/(1+\theta^n), \quad \theta^n = \Delta t^n/\Delta t^{n-1},$$

$$\bar{\omega}_k(\hat{\mathbf{y}}^n, t^n) = -\sum_{r=1}^{R} n_{rk} \sum_{\mathbb{Z}_+^m} w_r(\mathbf{x}, \hat{\mathbf{y}}^n, t^n) p_0(\mathbf{x}, t^n)\, d\mathbf{x},$$

and the implicit corrector is given by

$$\alpha_0^n \mathbf{y}^{n+1} = \Delta t^n \bar{\omega}(\mathbf{y}^{n+1}, t^{n+1}) - \alpha_1^n \mathbf{y}^n - \alpha_2^n \mathbf{y}^{n-1}, \tag{11}$$

$$\alpha_0^n = 1/(1+\theta^n), \quad \alpha_1^n = \theta^n - 1, \quad \alpha_2^n = -(\theta^n)^2/(1+\theta^n)$$

The local error in the solution $\mathbf{y}^{n+1}$ can be computed as

$$\epsilon^{n+1} = \frac{C_i(\theta^{n+1})(\mathbf{y}^{n+1} - \hat{\mathbf{y}}^{n+1})}{(C_p - C_i)}, \tag{12}$$

$$C_i(\theta^{n+1}) = -(1+\theta^{n+1})^2/(6\theta^{n+1}(1+2\theta^{n+1})), \quad C_p(\theta^{n+1}) = (1+1/\theta^{n+1})/6.$$

The implicit step (11) will then be solved using Newton iteration with the initial guess $\mathbf{y}^{n+1,0} = \hat{\mathbf{y}}^{n+1}$

$$\text{Solve} \quad (\alpha_0^n I - \Delta t^n \mathbf{J}_\omega)\delta^{\mathbf{k}} = -f(\mathbf{y}^{n+1,k}, \mathbf{y}^n, \mathbf{y}^{n-1}), \tag{13}$$

$$\mathbf{J}_\omega = \frac{\partial \bar{\omega}(\mathbf{y}^{n+1}, t^{n+1})}{\partial \mathbf{y}^{n+1}},$$

$$f(\mathbf{y}^{n+1,k}, \mathbf{y}^n, \mathbf{y}^{n-1}) = \alpha_0^n \mathbf{y}^{n+1,k} - \Delta t^n \bar{\omega}(\mathbf{y}^{n+1,k}, t^{n+1}) + \alpha_1^n \mathbf{y}^n + \alpha_2^n \mathbf{y}^{n-1}.$$

Update the solution according to

$$\mathbf{y}^{n+1,k+1} = \mathbf{y}^{n+1,k} + \delta^{\mathbf{k}}.$$

This leaves us with the following time stepping scheme, which also serves to illustrate the main algorithm of the hybrid solver.

First, a number of quasirandom sequences are computed and stored. These sequences contain the quadrature points used in the scheme below, and need only be generated once. Then the solution vectors and the trajectory matrix are initialized. Usually, the stochastic variables are set to be distributed according to a normal distribution centered at some user supplied point. The user also supplies an error tolerance for the time integration and a vector containg the times where an output of the solution are desired.

Then, the following procedure is repeated until the stopping time has been reached:

1. Simulate the stochastic variables with SSA up to time t^{n+1} using the values $\mathbf{y}^n$. This allows the computation of the distribution $p_0(\mathbf{x}, t^{n+1})$ at the selected quadrature points.

2. Compute predicted values for the deterministic subset according to eq. (10), using the distribution generated in the previous step (1).

3. Take the predicted value of $\mathbf{y}^{n+1}$ from step (2) as initial guess for the corrector in eq. (11) and perform Newton iterations as in eq. (13) until convergence.

4. Compute the local error as in eq. (12) and decide to accept or reject step based on the criterion below.

5. Repeat from step (1) until the stopping time has been reached.

In the case of adaptivity in the time step selection, some care need to be taken in the choice of Δt^n since it is expensive to recompute a step. The time step has been chosen in the following way. First compute

$$\zeta = \sqrt{\frac{e}{\|\epsilon\|_2}}$$

where e is the tolerance supplied by the user. Accept the step without any change in Δt if $\zeta \geq 0.98$. Increase $\Delta t^{n+1} = 1.1\Delta t$ if $\zeta \geq 1.15$, otherwise reject the step and try $\Delta t^n = 0.9\zeta\Delta t^n$.

3.4 Complexity of the algorithm

We will here consider the complexity of the steps involved in the simulation of the system. First, the work of the Gillespie algorithm is $\mathcal{O}(RM)$ if the number of reactions are R and the number of trajectories M. The constant can be quite large for some systems. In order to evaluate $p_0(\mathbf{x}, t)$ we need to sort the trajectory matrix twice. Let n_X, n_Y be the number of stochastic and deterministic variables respectively. The work in the quicksort algorithm is on average $\mathcal{O}(M \log_2(M))$, and to find N points of dimension n_X using binary search is $\mathcal{O}(n_X N \log_2(M))$. The evaluation of the integral need work of $\mathcal{O}(N)$ with a constant that depend on the cost of evaluating the function $\bar{\omega}(\mathbf{y}, t)$. Finally the evaluation of the Jacobian is $\mathcal{O}(N n_Y^2)$ and the solution of the system of equations in (13) is $\mathcal{O}(n_Y^3)$ with a direct method, but the system can be solved considerably faster with an iterative solver if necessary. We can see that there is extra work in the hybrid solver in proportion to $N \log_2(M)$, while we still need to perform SSA with work of order M. We assume that n_X and n_Y are small in comparison to N and M for most practical applications. From this we can conclude that the hybrid method could outperform SSA only if the reduction greatly affect the constant implicit in the

work of SSA. For example, consider a chemical system consisting of 100 reacting species, and it is possible to split the system so that only three of them need a stochastic treatment. Then both the number of reactions R and the constant of the SSA algorithm could be considerably reduced. However, the work would still be $\mathcal{O}(M)$ for M trajectories, but if the reduction of the constant is large enough to compensate for the increased work of $\mathcal{O}(N \log_2 M) + \mathcal{O}(M \log_2 M)$ (with a hopefully much smaller constant) a speedup could still be obtained. It is evident however, that the performance of the algorithm will be system dependent, and that the nature of the splitting will be important.

4 Numerical results

4.1 Coupled flows

The system of coupled flows discussed in Sect. 2.1.2 has been simulated using the hybrid algorithm. The variables have been partitioned so that the metabolites A, B are treated as stochastic variables, and the enzymes E_1 and E_2 are treated deterministically. With this partitioning, we end up with the following set of reactions and equations

$$\begin{array}{ll} \emptyset \xrightarrow{\frac{k_a[E_1]}{1+\frac{[A]}{K_i}}} A & \emptyset \xrightarrow{\frac{k_b[E_1]}{1+\frac{[B]}{K_i}}} B \\ A \xrightarrow{\mu} \emptyset & B \xrightarrow{\mu} \emptyset \\ A + B \xrightarrow{k2} \emptyset & \end{array} \qquad \begin{cases} \dfrac{d[E_1]}{dt} = \dfrac{k_{ea}}{1+\frac{[A]}{K_r}} - \mu[E_1] \\ \dfrac{d[E_2]}{dt} = \dfrac{k_{eb}}{1+\frac{[B]}{K_r}} - \mu[E_2] \end{cases}$$

This means that 9 reactions have been reduced to 5 in SSA, and two linear ODEs have to be solved for the mean of the enzyme concentrations. The values of the reaction rate constants that are used in the simulation are found in table 1.

k_a	k_b	k_2	K_i	μ	k_{ea}	k_{eb}	K_r
0.3	0.3	0.001	60	0.002	0.02	0.02	30

Table 1: The parameters for the coupled flows

The problem is numerically simple to solve, since the ODEs are linear and not stiff. From the equations describing this system, we can also note that if A, B and E_1, E_2 are given the same initial values, the solution for the metabolites and the enzymes will be identical. In all simulations, the initial values were $A, B, E_1, E_2 = 10$. Simulation of the system with the Gillespie algorithm (Fig. 4 and Fig. 5) reveals that the metabolites A and B display relative large fluctuations around

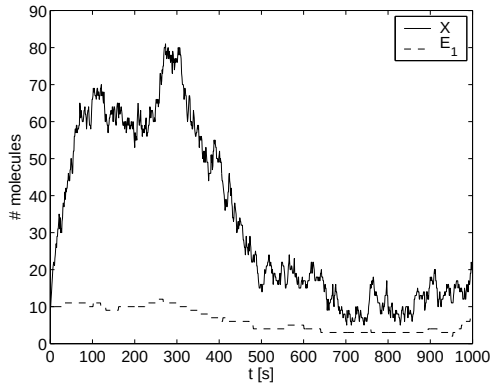

Fig. 4: One realization of a simulation of the system. Metabolite species X and enzyme E_1.

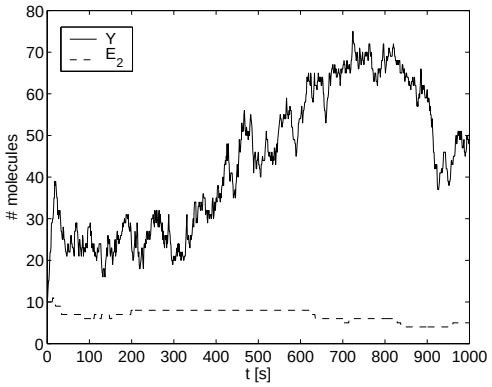

Fig. 5: Metabolite Y and enzyme E_1

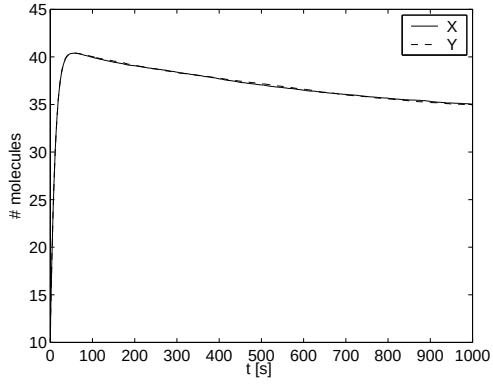

Fig. 6: Expected values of metabolite species X and Y based on 10^5 independent trajectories.

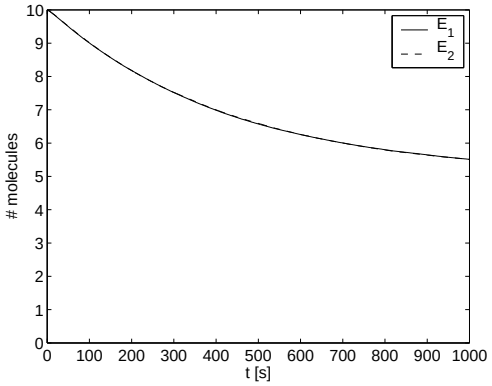

Fig. 7: Expected values of enzymes E_1 and E_2 based on 10^5 trajectories.

their mean, while the enzymes vary to a lesser extent. This motivates the splitting of the state space, even though the copy number of the enzymes are lower than for the metabolites.

Figs. 4 to 9 show the results from a stochastic simulation of the system up to $t = 1000$ s. The number of trajectories used to form expected values and approximations of the PDFs were taken to be 10^5. At this time, the system has not reached steady state, but very small changes take place if the system were to be simulated longer. Also in Fig 8 we see that the fluctuations of species X, Y is larger than for the enzymes, since the distribution is much smoother for the enzymes for the same number of trajectories.

For the hybrid solver, different integration techniques have been evaluated. In Fig. 10 we see the result from integration with the probability distribution taken at $t = 1000s$ which is the endpoint of a hybrid simulation. Raw Monte Carlo with

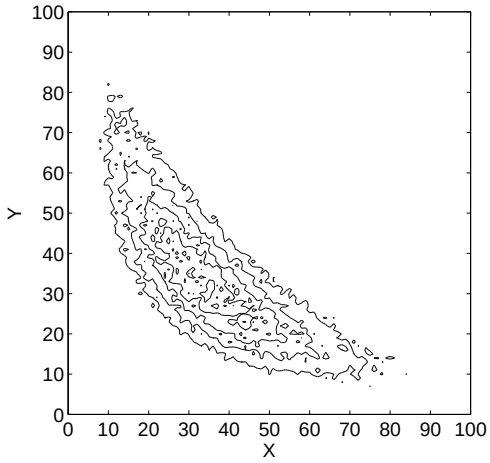

Fig. 8: Isolines of the probability density based on 10^5 trajectories. The metabolites X and Y

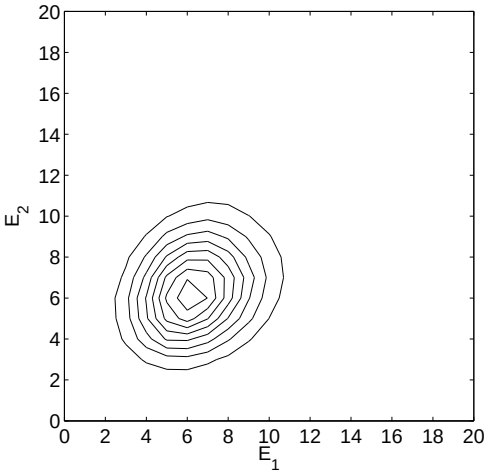

Fig. 9: Isolines of the probability density of the two enxymes E_1 and E_2 based on 10^5 trajectories.

pseudorandom numbers (i.e. no sampling from $p_0(\mathbf{x}, t)$) has been compared to raw Monte Carlo using numbers from the Faure sequence generated with [13]. Also, different smoothing alternatives have been tested. One important consideration is the fact that when for example the rejection method is used to generate numbers from the distribution, fewer evaluation points will be accepted and used than in e.g. raw Monte Carlo. This means that the number of quadrature points N in (8) will be different for different methods. So the errors reported in Fig. 10 refer to the error obtained with a fixed number of trial points N. This is not the usual way to compare different methods, but what we are interested in is which method gives the smallest error per generated random number, since there is a cost in generating the numbers and most of all in evaluating the distribution. It has to be evaluated for all trial points, not just the accepted ones. The probability distribution function has been computed based on 10^5 trajectories.

From Fig. 10 we see that quasirandom numbers give a much better result than pseudorandom numbers also in this case. However, rejection sampling, B-splines weighted sampling and linearly smoothed rejection sampling all perform badly for this problem. In fact, QMC with either of these methods have no smaller error nor higher convergence rate than raw Monte Carlo with pseudorandom numbers. The smoothing width $2h$ has been chosen to be $2e-3$ for both B-splines and linear smoothing. This parameter has been chosen to be the best in a number of tests with different values. Apparently, the characteristic function could not be smoothed sufficiently for the QMC estimator to regain convergence rate. Also, one problem is the fact that the sampling distribution $p_0(\mathbf{x}, t)$ is not smooth, which is a requirement to assure the higher convergence rate. However, uniform weighted sampling (UWF) performs well on this problem, with as high conver-

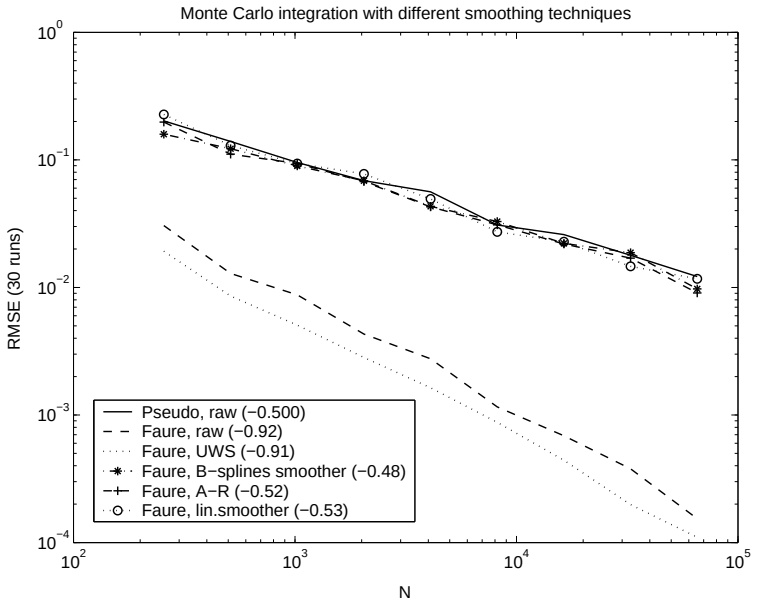

Fig. 10: The relative RMSE for some different variants of Monte Carlo integration.

gence rate as raw quasi-Monte Carlo, and even smaller relative error. Therefore, this method has been chosen for the time integration in the hybrid solver.

We would also expect the integration error to be dependent on the number of trajectories used to construct the sampling distribution, i.e. the error of the probability density function. Fig. 11 shows the RMSE for uniform weighted sampling for some different number of trajectories. The distribution function $p(\mathbf{x}, t)$ was taken from simulations with the hybrid solver at $t = 103s$.

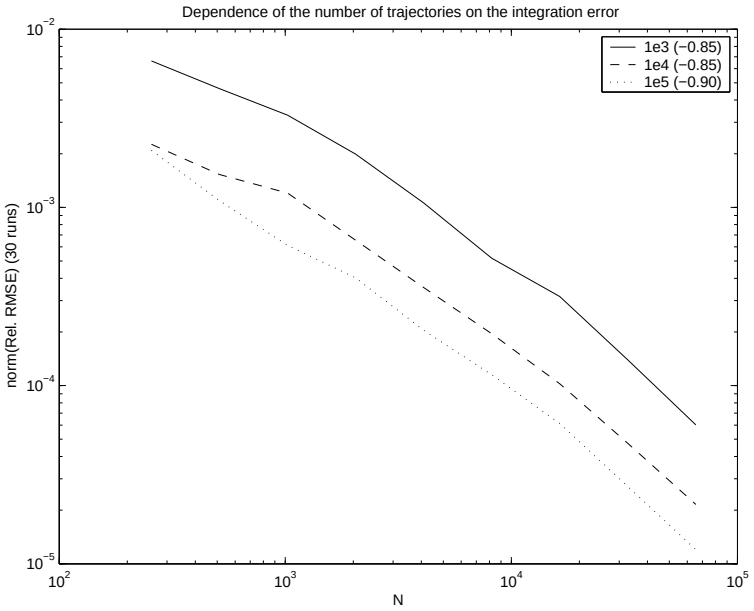

Fig. 11: Relative RMSE for varying number of trajectories used to construct $p_0(\mathbf{x}, t)$ (the numbers in the ledgend, $10^3 - 10^5$). N is the number of points from the Faure sequence. Uniform weighted sampling (UWF) has been used.

In Figs. 12 to 18 we see the results of a simulation with the hybrid solver. The time steps were chosen adaptively with an relative error tolerance of 0.1%. As a maximal time step $\Delta t = 5s$ was chosen. Even if the local error tolerance for the deterministic variables can be met for even larger time step, since the values of these variable will be treated as constants during the simulation with SSA, an upper limit is probably appropriate for the time step.

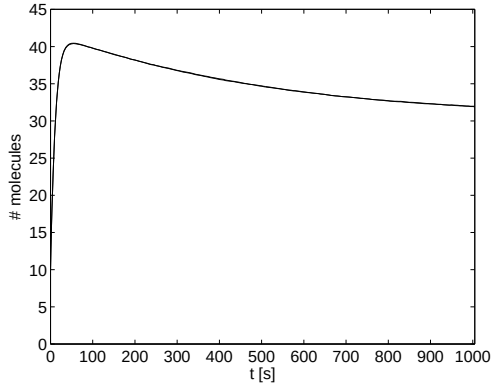

Fig. 12: Expected values of X and Y computed by simulation with the hybrid solver. Stochastic variables.

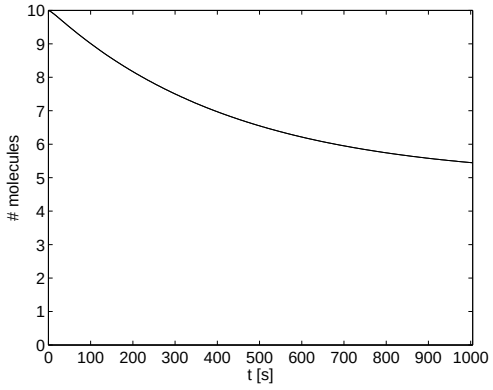

Fig. 13: Expected values of E_1 and E_2, the deterministic variables in the reduction.

The mean values computed with the hybrid solver compares well with those of the full SSA (Fig. 6). The same number of trajectories, $M = 10^5$, has been used to approximate the probability distribution.

For this problem, meeting the error tolerances is relatively easy. In this simulation not a single time step was rejected. Figs. 14 and 15 shows the time step and the relative local error measured in this simulation.

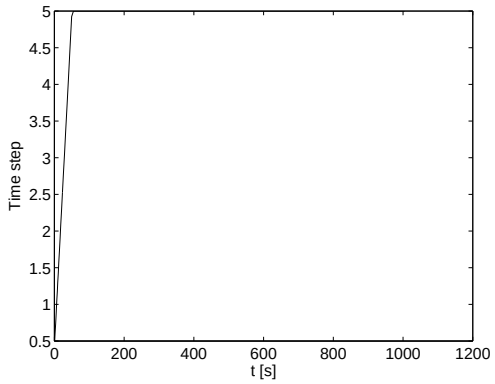

Fig. 14: Time step chosen by the adaptive time stepping scheme.

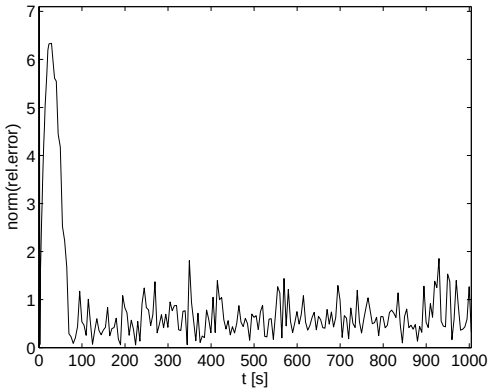

Fig. 15: Relative local error.

The time step quickly reaches its maximal value, and the relative local error of the ODE solution is kept small. We can also look at the relative integration error. The integration within the solver is performed by using ten scrambled sequences. They are generated and stored initially and then used to form estimates of the integral and error as in (7) and (8). If the RMSE is smaller than 10% of the local error tolerance of the ODE solver no more points are used. In practice, the tolerance has proven to be difficult to meet for small values. The weights generated are stored and used throughout the Newton iteration in order for it to converge. In the simulations reported here, $2^{16} \approx 65.000$ quasirandom points were used for each sequence. Figs. 16 and 17 show the relative and absolute RMSE for the simulation. As mentioned, uniform weighted sampling was used as integration method.

Unfortunately, while the hybrid method captures the behavior of the expected values, it does not provide as good an approximation of the PDF. Figure 18 shows isolines of the probability density function at $t = 1000s$ computed with the hybrid solver (10^5 trajectories). If we compare with fig 8 we see that some of the characteristic shape of the coupled flows has been lost in the approximations made. The distribution is still centered on the same expected values, but some synchronization of the trajectories is inevitable in the hybrid approach. However, the distribution obtained with this solver compares well with the results in [21] where a Fokker-Planck approach is used to solve the same problem.

To address the issue of efficency of the method, the MATLAB tool *Profiler*

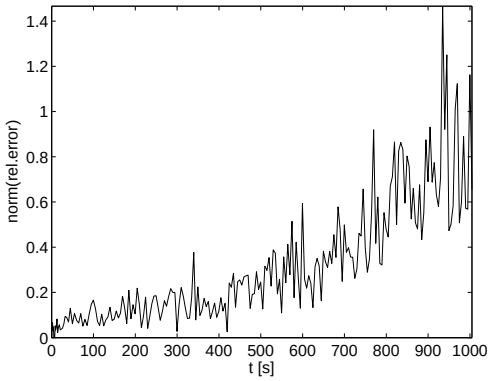

Fig. 16: Relative integration error. Each sequence used consisted of 2^{16} points. (See the text for details.)

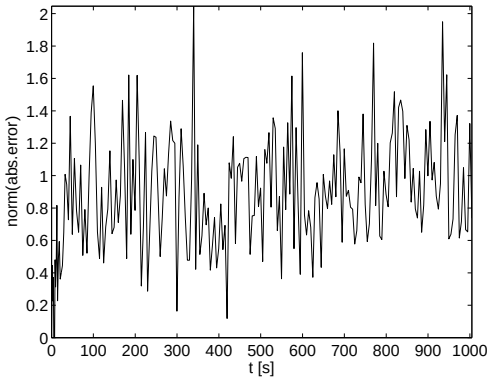

Fig. 17: Absolute integration error.

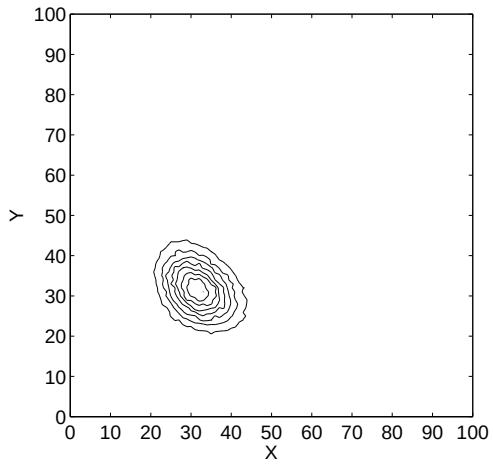

Fig. 18: The probability density at $t = 1000s$ computed with the hybrid solver.

has been used. It provides a detailed list of the time spent on each line in the code and the relative amount of time taken by all subroutines. Since the major components of the hybrid solver (SSA, evaluation of the distribution, integration and computation of the Jacobian) have been implemented in C and wrapped as mex-files, it is easy to compare their contribution to the time needed to solve the system for a given set of parameters. The generation of the quasi sequence is written as a mex-file calling a fortran routine [13]. Even if this is also time consuming, it is only done once initially and contributes little to the total time if the number of time steps needed to compute the solution is sufficiently large. The number of evaluation points for the integration is $2^{15} \approx 32.000$ and can be considered to be a moderate value for this problem.

In Table 2 we see the relative amount of time spent in SSA within the hybrid solver for varying number of trajectories when the system is solve to $t = 103s$. The values reported are the percentage of the total execution time.

Number of trajectories	10^3	10^4	10^5	10^6
Time spent in SSA [%]	3.3	18.7	44.7	50.9
Total time [s]	35	48	181	1615

Table 2: Time spent in SSA.

We see that for many trajectories the SSA takes about 50% of the time. The other major part is the sorting algorithms provided by Matlab, and needed in order to evaluate the probability density function. They take almost 40% of the execution time for 10^6 trajectories. It is evident that this part of the algorithm needs to be improved. However, what is interesting here is the fact that if SSA for the full system would take more than twice the time of the corresponding split system, the hybrid solver would be faster already for a four dimensional problem. For this particular problem though, the gain in time is not enough to compensate for the additional work related to the deterministic part of the solver. Indeed, for 10^6 trajectories the time spent in SSA in the hybrid solver was 821 s compared to 872 s for the full system.

4.2 The Vilar oscillator

The Vilar oscillator [35] is a model for circadian rhythms, illustrating some common control components that have been observed in such systems. This kind of control system is designed to assure periodic oscillations of certain molecular species in order to establish a circadian rhythm in the organism. Obviously, a system of this kind would be very complicated in an actual organism, and this model system is artificial. The model considers nine molecular species. Two genes D_a and D_r and their corresponding mRNA M_a and M_r are controlled by an activator and a repressor A and R, synthesized from the respective mRNA. Furthermore,

the activator and repressor can associate and form a complex C, in which the activator A is degraded. The deterministic set of reaction rate equations for this system are

$$\begin{cases} \frac{dD_A}{dt} &= \Theta_A D'_A - \gamma_A D_A A \\ \frac{dD_R}{dt} &= \Theta_R D'_R - \gamma_R D_R A \\ \frac{dD'_A}{dt} &= \gamma_A D_A A - \Theta_A D'_A \\ \frac{dD'_R}{dt} &= \gamma_R D_R A - \Theta_R D'_R \\ \frac{dM_A}{dt} &= \alpha'_A D'_A + \alpha_A D_A - \delta_{M_A} M_A \\ \frac{dA}{dt} &= \beta_A M_A + \Theta_A D'_A + \Theta_R D'_R - A(\gamma_A D_A + \gamma_R D_R + \gamma_C R + \delta_A) \\ \frac{dM_R}{dt} &= \alpha'_R D'_R + \alpha_R D_R - \delta_{M_R} M_R \\ \frac{dR}{dt} &= \beta_R M_R - \gamma_C AR + \delta_A C - \delta_R R \\ \frac{dC}{dt} &= \gamma_C AR - \delta_A C \end{cases}$$

The variables D'_A and D'_R are the genes D_A and D_R with bound activator. In the model it is assumed that there are only one gene coding for the repressor and activator. Thus $D_A + D'_A = 1$, and the same is true for the repressor gene. For the parameters given in Table 3 the system displays a limit cycle as in Figs. 19,20. However, if the parameter δ_R is sufficiently small the fixed point becomes stable and the system stops oscillating (Fig. 21). It was shown in [35] that a mesoscopic description of the system continues to produce reliable oscillations even when the fixed point is stable in the deterministic sense. The noise is obviously sufficient to perturb the trajectories far enough from the fixed point to initiate new cycles.

α_A	α'_A	α_R	β_A	β_R	δ_{M_A}	δ_{M_R}
50.0	500.0	0.01	50.0	5.0	10.0	0.5
δ_A	δ_R	γ_A	γ_R	γ_C	Θ_A	Θ_R
1.0	0.2	1.0	1.0	2.0	50.0	100.0

Table 3: Parameters for the Vilar oscillator.

Even for the deterministic solution (Figs. 19, 20) the cycles are not completely identical. This is probably due to numerical errors of the time integration scheme, and thus the system is rather hard to solve. Fig. 21 shows the behavior of the system with the parameters as in Table 3 but with $\delta_R = 0.08$. In the deterministic sense, the fixed point is stable and no oscillations can occur.

In order to apply the hybrid solver, the variables were divided into two subsets. A, R and C were treated stochastically while the other variables were treated

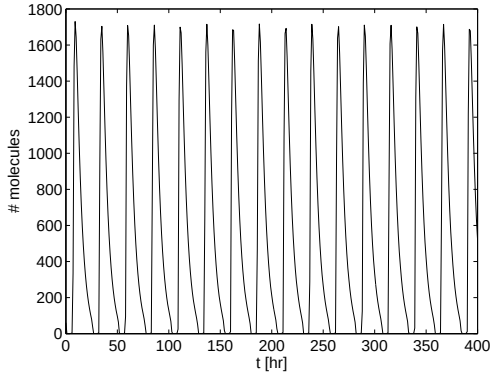

Fig. 19: Time evolution of the repressor. The system was solved with Matlab's solver `ode15s` with the default relative error tolerance $1e-3$. All parameters are as given in table3

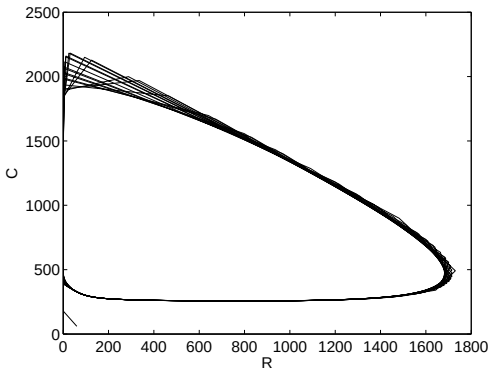

Fig. 20: Phase portrait of the repressor and the complex.

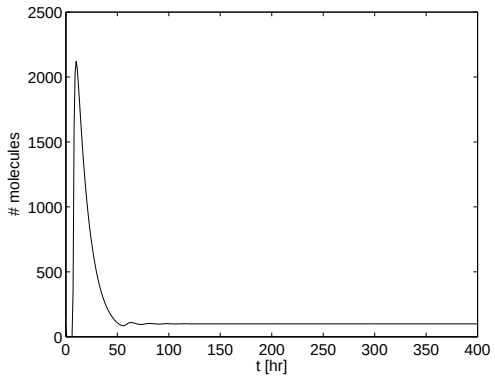

Fig. 21: Time evolution of the repressor when $\delta_R = 0.08$. The fixed point is now stable and no oscillations occur.

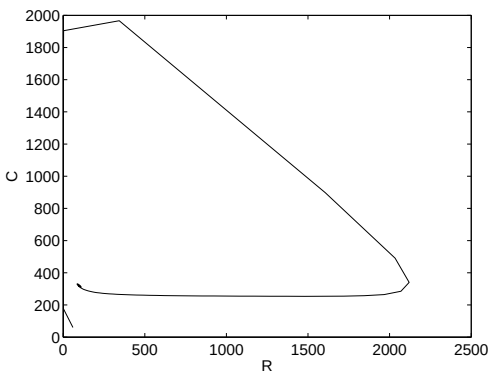

Fig. 22: Phase portrait of the repressor and the complex when $\delta_R = 0.08$.

deterministically. This may at first seem to be a bad splitting, since the genes are binary variables and should at a first glance be treated stochastically. However, these variables are effectively averaged [35] and this motivates a deterministic treatment. Once again however, the copy number is no way nearly as large as required for the variables to be assumed to be normally distributed, even if the variance might be small. One need to keep in mind though, that the same assumption is made in the reaction rate equations.

This splitting gives three stochastic variables participating in 10 chemical reactions (compared to 16 of the full system), and six deterministic variables. The set of reactions and equations describing the system is

$$
\begin{array}{ll}
A + D_A \xrightarrow{\gamma_A D_A A} D'_A & D'_A \xrightarrow{\Theta_A D'_A} A + D_A \\
\emptyset \xrightarrow{\beta_A M_A} A & A \xrightarrow{\delta_A A} \emptyset \\
A + R \xrightarrow{\gamma_C AR} C & C \xrightarrow{\delta_A R} R \\
\emptyset \xrightarrow{\beta_R M_R} R & R \xrightarrow{\delta_R} \emptyset \\
A + D_R \xrightarrow{\gamma_R A D_R} D'_R & D'_R \xrightarrow{\Theta_R D'_R} A + D_R
\end{array}
$$

$$
\begin{cases}
\frac{dD_A}{dt} &= \Theta_A D'_A - \gamma_A D_A A \\
\frac{dD_R}{dt} &= \Theta_R D'_R - \gamma_R D_R A \\
\frac{dD'_A}{dt} &= \gamma_A D_A A - \Theta_A D'_A \\
\frac{dD'_R}{dt} &= \gamma_R D_R A - \Theta_R D'_R \\
\frac{dM_A}{dt} &= \alpha'_A D'_A + \alpha_A D_A - \delta_{M_A} M_A \\
\frac{dM_R}{dt} &= \alpha'_R D'_R + \alpha_R D_R - \delta_{M_R} M_R
\end{cases}
$$

Figs. 23 to 26 shows a solution to this system with the hybrid solver. For this problem, adaptivity is not a very good alternative, since many time steps have to be recomputed and there is a considerable cost associated with this, especially when the number of trajectories is large. Here, 10^6 trajectories were used to construct the probability density, and 2^{15} quasi-random points used in each sequence in the integration algorithm. There has been a passive monitoring of the local time integration error, but no attempt to control it has been made in this case. The error is unacceptably large in some time intervals and this leads to a drift of the phase of the oscillations. However, the solution captures the general, oscillatory properties of the original system. A fixed time step $\Delta t = 0.5$ was used when solving this system. Initial conditions for the deterministic variables were 0.2 for the genes and zero for the other species. The stochastic variables were initiated as a normal distribution centered around $\{A, B, C\} = \{60, 60, 60\}$.

As can be seen in Fig. 26 the genes have a copy number below one. This can at first seem to be biologically irrelevant, but is in fact merely a scaling in order

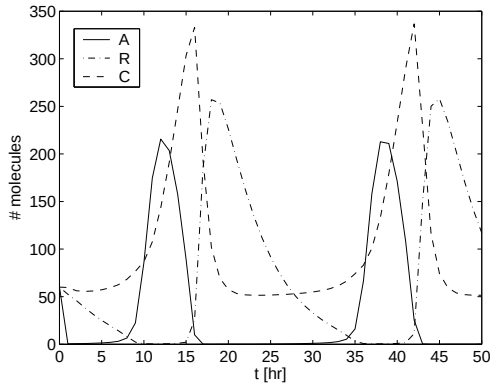

Fig. 23: Time evolution of the stochastic variables.

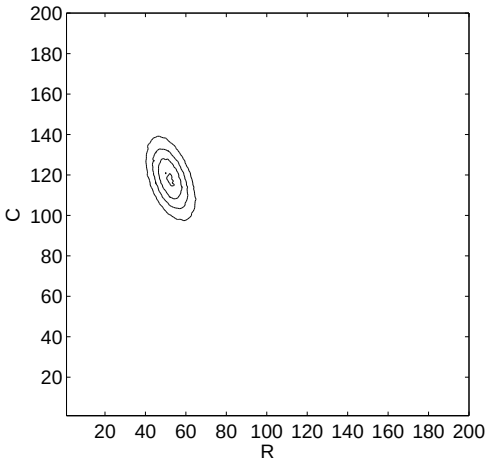

Fig. 24: Isolines of the distribution of the repressor and the complex computed with the hybrid solver.

to reduce the state space. Obviously, the amplitude of the oscillations is affected but the general behavior of the system is not.

The contribution of the SSA algorithm to the total time required to solve the system has been studied in the same manner as for the coupled flows. The results can be seen in table 4. For this system, the stochastic simulations are more demanding than for the coupled flows, and for the highest number of trajectories it consumes more than 90% of the CPU time. This system consists of only nine variables, and it should be clear that for even larger systems there is much time to save if a good splitting can be chosen.

Number of trajectories	10^3	10^4	10^5	10^6
Time spent in SSA [%]	18.6	53.1	83.9	91.8
Total time [s]	178	590	$3.5e3$	$29.7e3$

Table 4: Time spent in SSA.

In Fig. 21 we have seen that the oscillations stops in the deterministic simulation of the system if the parameter δ_R is reduced to 0.08. However, the hybrid solver still gives rise to oscillations for this value. Unfortunately, the presence of stable oscillations requires that the local error in the time integration of the deterministic variables is kept small for this particular system. As already discussed, adaptivity might not be a good option for this system, but here we are forced to control the error. Fig. 27 shows the solution with the hybrid solver when the relative local error is kept below 0.5%. Ideally, for the hybrid solver to be as effective as possible, the splitting needs to be done in such a way that the

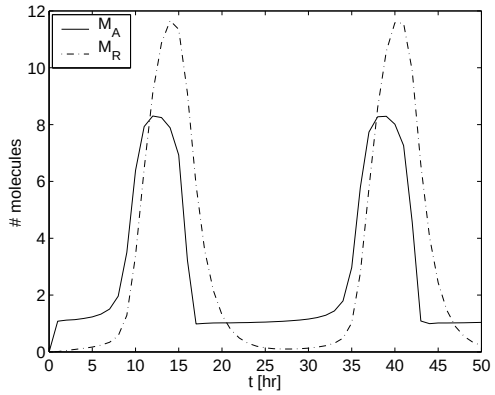

Fig. 25: Time evolution of the deterministic variables M_A and M_R.

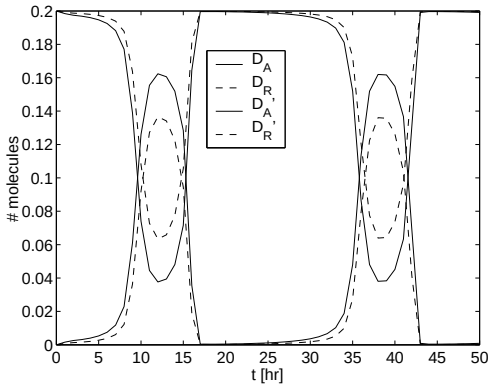

Fig. 26: Time evolution of the genes.

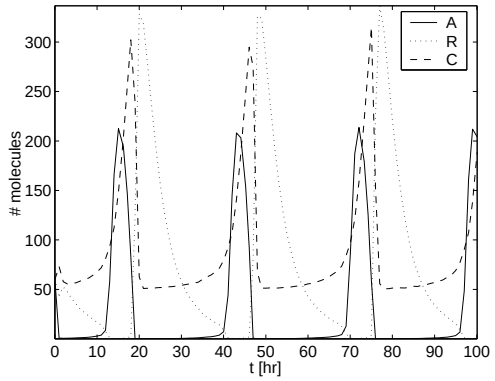

Fig. 27: The expected values of species A,R,C when $\delta_R = 0.08$. The hybrid solver produces sustained oscillations.

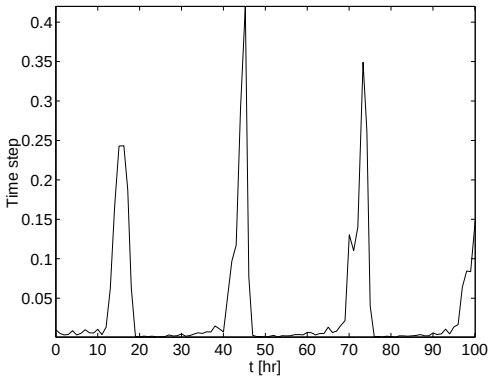

Fig. 28: Time steps chosen in the adaptive algorithm. In most time intervals, the time steps need to be chosen very small.

deterministic subset yields equations for which large time steps are permitted. This condition can not be met in this particular case, as can be seen in Fig. 28.

Finally, we consider the integration error of the QMC quadrature for this problem. Fig. 29 shows the results from an integration with the probability distributions at time $t = 100s$, taken from the endpoints of the executions in Table 4. The actual values are not directly comparable, since due to numerical errors and the stochastic influence, the solution differs slightly. However, we are interested in how the convergence rate of the integration depends on the number of trajectories used to construct the distribution. As can be seen, the convergence rate increases with increasing number of trajectories. This is to be expected, since this means a smaller error and increased smoothness of the distribution $p_0(\mathbf{x}, t)$.

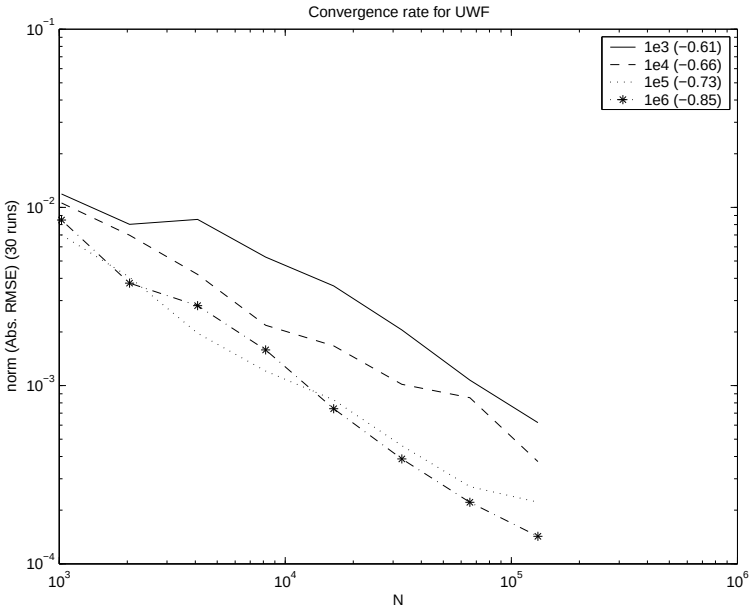

Fig. 29: Absolute RMSE for varying number of trajectories, uniform weighted sampling using the Faure sequence.

4.3 A mitogen-activated protein kinase signaling cascade

As a final example we will consider a model of a mitogen-activated protein kinase (MAPK) signaling cascade [17]. These receptor mediated signal transduction pathways are conserved regulatory systems, and consist of three sequentially acting kinases. Kinases are proteins that modify other proteins by the phosphorylation of certain amino acid residues. This modification has a different effect on different proteins, and could lead to e.g. changes in binding properties to DNA/RNA or to other proteins. In this case, the first protein in the cascade, RAF, induces two phosphorylations of MEK, the second component in the chain. Doubly phosphorylated MEK, MEKpp, in turn induces two phosphorylations of MAPK. The output signal in the scheme, doubly phosphorylated MAPK, MAPKpp, can dimerize and in this form be transported into the nucleus where it phosphorylates a number of transcription factors,proteins directly involved in the regulation of the transcription of genes to mRNA [20]. The model in [17] also takes into account the effect of scaffolds, protein complexes that facilitate signal transduction by the binding of several components of the chain (bringing reacting species closer to each other). The model we simulate here is the corresponding system without scaffolds, i.e. a MAPK cascade in solution.

The model includes 22 variables, the kinases RAF, MEK and MAPK, the dephosphatases RAFPH, MEKPH which removes phosphate groups from their corresponding kinase, and the possible dimers formed in the reactions. These 22

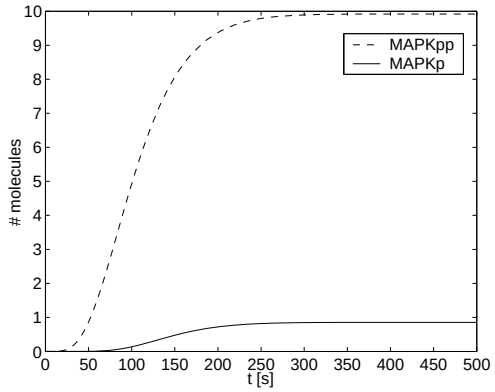

Fig. 30: Deterministic simulation of the MAPK cascade. Solution is computed with MATLAB `ode15s`.

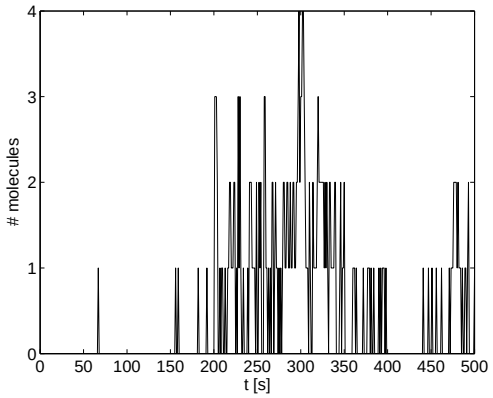

Fig. 31: Single trajectory SSA showing the fluctuations in activated MAPK kinase, MAPKpp.

variables take part in 30 chemical reactions, so this model is larger than the ones previously considered.

Fig. 30 shows a deterministic simulation of the system with Matlab's ode15s. Only the singly and doubly phosphorylated MAPK are displayed. Except for a few species, the components vary slowly in time, but with the parameters given in the supplementary material of [20] the number of MAPKpp molecules at maximal output signal is rather small. In fact, the mean value is below one. A stochastic simulation with SSA reveals that the number of molecules of doubly phosphorylated MAPK vary from zero to a few molecules after an initial time when no active MAPK is present (Fig. 31).

Suppose that we want to make a larger model where a control system like this is one component. We might then be interested in the stochastic variation of the output signal (MAPKpp), while the detailed stochastic information regarding the other components is less important. This is a scenario where the hybrid approach could provide a speedup compared to the full SSA. To evaluate the performance, the state space have here been separated so that singly and doubly phosphorylated MAPK (MAPKp, MAPKpp) are treated as discrete stochastic variables, and the other as deterministic. In this way, 30 chemical reactions for 22 chemical species are reduced to nine reactions and 20 integro-differential equations. Figs. 32 and 33 show isolines of the distributions at $t = 500s$ computed with SSA for the full system and the hybrid solver. For both methods, 10^5 trajectories was used to approximate the distribution.

As can be seen, the densities compare well for the two methods, and the hybrid solver is in this case able to capture the stochastic properties of the activated MAPK species. Since we have only two stochastic variables and their copy numbers are small, a relatively small number of quadrature points needs to be used, and the SSA of the hybrid solver is also much cheaper than that of the

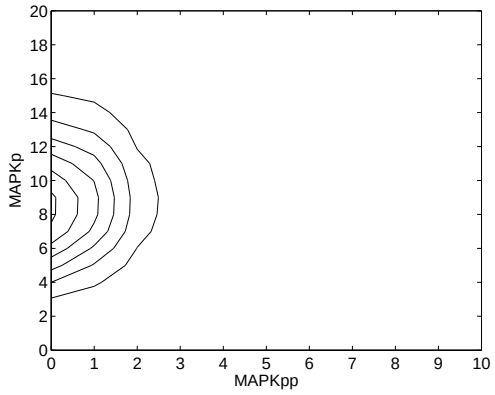

Fig. 32: Isolines of the distribution computed with SSA for the full system.

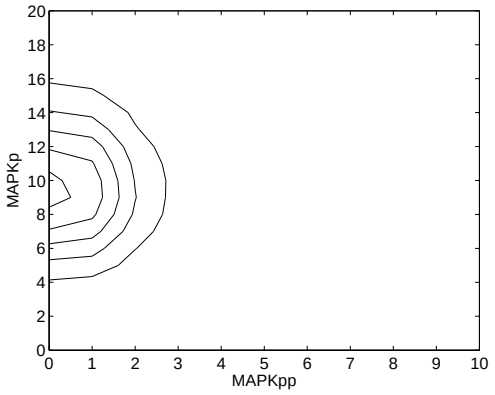

Fig. 33: Isolines of the distribution computed with the hybrid solver.

full system. With 10^4 trajectories (which can be considered to be a relatively low number), the hybrid solver computes the solution up to $t = 200s$ five to six times faster than full SSA. The relative efficiency of the hybrid solver compared to SSA is evidently dependent on the choice of the number of trajectories and the number of quadrature points. For example, if we want better resolution of the distribution and lower relative error in the value of the integral, we could choose e.g. 10^5 trajectories and 2^{15} quadrature points in each sequence. With these parameters, the hybrid solver is also five to six times faster than full SSA. One need to keep in mind that this is still a small system. For example, the same model extended to include scaffolds [20] have 89 variables participating in over 300 chemical reactions. For a system of this size, the hybrid solver is predicted to be a lot faster than SSA if a good separation can be chosen.

5 Conclusions

In this thesis, a hybrid solver for coupled macroscales and mesoscales have been implemented and evaluated. From the numerical experiments it can be seen that this solver is able to capture important features of the fully mesoscopic description, while keeping the number of stochastically treated variables at a manageable level.

For some of the test systems considered, the number of reactions are too few and the splitting done in such a way that an improvement in execution time over the full SSA algorithm can not be obtained. It is shown however, that for systems where this splitting can give a sufficient reduction in the number of reactions and the rate constants involved in those reactions, a considerable speedup compared to a fully stochastic simulation can be obtained. For a larger system, the hybrid solver is shown to execute five to six times faster than the full SSA with the chosen splitting, while it is still able to retain the stochastic properties of the original

model. Alternatively, the hybrid solver can be viewed upon as a way of improving the macroscopic model by introducing stochasticity in some components. With this viewpoint, the hybrid solver is more computationally demanding than the ODE models, but gives more realistic results to a rather low additional cost.

The major bottleneck in the time stepping scheme of the hybrid solver is, apart from the simulation, the evaluation of the probability distribution function $p_0(\mathbf{x}, t)$. The number of evaluation points is determined by the performance of the integration algorithm. It is therefore crucial to develop a scheme that gives a small error with few evaluated quadrature points. Here, the integration is done with a quasi-Monte Carlo method. Monte Carlo methods are well suited for high dimensional integration, and even if the dimensions are not particularly high for the systems considered here, a generally applicable method is required if the intention is to be able to simulate different systems without extensive reprogramming by the user. One problem with QMC methods is that the higher convergence rate compared to pseudorandom numbers depends on the integrand being sufficiently smooth. Since we are dealing with a discrete distribution this is not the case and we have seen that a large number of trajectories are needed. Even when this requirement can be met, the convergence rate is still not as high as in the ideal smooth case.

5.1 Future work

In this implementation, no optimization of the code for speed has been made. The SSA algorithm is an implementation of Gillespie's direct method. However, there are faster modifications of this algorithm, and any attempt to implement a generally applicable solver should consider the use of some of these methods. Gibson and Bruck [7] have shown that their 'Next reaction' method performs much better than the original SSA algorithm. Furthermore, Gillespie has proposed the 'tau-leap' method [14] which is an approximation of the original algorithm in which one 'leaps' over some time steps. Perhaps one should use this approximation for intermediate variables and the exact algorithm for the other variables in order to reduce the computational burden even more. One attempt to use this approach has been made in [29].

We have seen that the discrete nature of the probability distribution imposes some problems in the integration schemes. Since the evaluation of the distribution is the most important part to improve, something should be done in order to enhance the convergence rate. One possibility would be to try to make a continuous approximation of the discrete distribution by smoothing out the discontinuities by interpolation. There are also a few adaptive Monte Carlo software packages avaliable, e.g. [17]. They are provided as open source, and could possibly be modified to deal with the special requirements of this solver.

Perhaps the most difficult task when setting up a hybrid scheme is the splitting of the state space. As for now, this has to be done manually and requires some

previous knowledge of the system. If the system is large, a trial and error approach can be tedious due to the time taken to solve the system. If some method to automatically partition the variables could be implemented a lot would be gained, since more of the research time could be spent on drawing conclusions concerning the system at hand than on setting up the computation. Such a partitioning algorithm would also make it possible to treat variables differently at different times, for example if the chemical species are present in large copy numbers for long times and then drop to small values.

Obviously, there is a great interest in extending the stochastic models to include spatial dimensions. For example, in the model of the MAPK cascade, a clear spatial dependence can be seen. Some proteins in the cascade are located preferentially to the plasma membrane, either by direct interaction with some receptor or via scaffolds. Dimerized MAPKpp in turn, can be transported trough the nuclear membrane and has an important function inside the nucleus. In this case, it could be interesting to study how the distribution of activated MAPK in different locations and compartments of the cell changes over time following the binding of signal molecules to the corresponding receptor.

Within the hybrid approach, adding diffusion and convection terms to the deterministic equations would be possible. How to efficiently handle these issues for the stochastic variables are an area where an extensive research effort is necessary.

Finally, there is already software developed for the simulation of chemical reactions. With the need to be able to share models within the research community, a markup language has been developed [14]. SBML, or *Systems Biology Markup Language*, is avaliable for example as a toolbox for Matlab. In order to be able to try different systems more quickly, the hybrid solver should be extended to handle SBML.

6 Acknowledgments

I would like to thank my supervisor, Prof. Per Lötstedt, for the time invested in this project. I am also greatful to the Department of Information Technology for willingly providing the computer resources to make this work easier.

References

[1] Bratley P., Fox B.L., Niederreiter H., Implementation and tests of low-discrepancy sequences, ACM Trans. Model. Comp. Sim., 2, 1992, pp. 195-213.

[2] Caflisch R.E, Monte Carlo and quasi-Monte Carlo methods, Acta Numerica, 1998, pp.1-49.

[3] Faure H., Discrépance de suites associées à un système de numération (en dimesion s), Acta Arithmetica, 41, 1982, pp. 337-351.

[4] Faure, H., Tezuka, S., Another random scrambling of digital (t,s)-sequences, Monte Carlo and quasi-Monte Carlo methods, 2000, (Hong Kong), pp. 242-256, Springer, Berlin, 2002.

[5] Ferm L., Lötstedt P., Numerical method for coupling the macro and meso scales in stochastic chemical kinetics, Dept. of Information Technology, Uppsala University, Uppsala, Sweden, 2006.

[6] Gavin, A.C., Bosche, M., Krause, R., Grandi, P., Marzioch, M., Bauer, A., Schultz, J., Rick, J.M., Michon, A.M., Cruciat, C.M., Remor, M., Hofert C., Schelder, M., Brajenovic, M., Ruffner, H., Merino, A., Klein, K., Hudak, M., Dickson, D., Rudi, T., Gnau, V., Bauch, A., Bastuck, S., Huhse, B., Leutwein, C., Heurtier, M.A., Copley, R.R, Edelmann, A., Querfurth, E., Rybin, V., Drewes, G., Raida, M., Bouwmeester, T., Bork, P., Seraphin, B., Kuster, B., Neubauer, G., Superti-Furga, G., Functional organization of the yeast proteome by systematic analysis of protein complexes, Nature, 415(6868):123-4, 2002.

[7] Gibson M.A.,Bruck, J., Efficient exact stochastic simulation of chemical systems with many species and many channels, J. Phys. Chem. A, 104, 2000, pp. 1876-1889.

[8] Gillespie D.T., A general method for numerically simulationg the stochastic time evolution of coupled chemical reactions, J. Comput. Phy., 22, 1976, pp. 403-434.

[9] Gillespie, D.T., Approximate accelerated stochastic simulation of chemically reacting systems, J. Chem. Phys., 115, 2001, pp. 1716-1733.

[10] Gillespie, D.T., Markov Processes: An Introduction for Physical Scientists, Academic Press, San Diego, 1992.

[11] Goldbeter, A., Computational approaches to cellular rhythms, Nature, 420, 2002.

[12] Haseltine E.L., Rawlings J.B., Approximate simulation of coupled fast and slow reactions for stochastic chemical kinetics, J. Chem. Phys., 117, 2002.

[13] Hong H.S., Hickernell F.J., Algorithm 823: Implementing scrambled digital sequences, ACM Trans. Math. Softw., 29, 2003, pp. 95-109.

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

[15] Kitano, H., Computational systems biology, Nature, 420, 2002, pp. 206-210.

[16] Lemieux C., Randomized quasi-Monte Carlo: A tool for improving the efficiency of simulations in finance, Proceedings of the 2004 Winter Simulation Conference, 2004.

[17] Levchenko, A., Bruck, J., Sternberg, P.W., Scaffold proteins may biphasically affect the levels of mitogen-activated protein kinase signaling and reduce its threshold properties, Proc Natl Acad Sci USA, 97, 2000, pp. 5818-5823

[18] Ley, G., B-splines smoothed rejection sampling method and its applications in quasi-Monte Carlo integration, Journal of Zhejiang University, 3, 2002, pp. 339-343.

[19] Li, S., Armstrong, C.M, Bertin, N., Ge, H., Milstein, S., Boxem, M., Vidalain, P.O., Han, J.D., Chesneau, A., Hao, T., Goldberg, D.S., Li, N., Martinez, M., Rual, J.F., Lamesch, P., Xu, L., Tewari, M., Wong, S.L., Zhang, L.V., Berriz, G.F., Jacotot, L., Vaglio, P., Reboul, J., Hirozane-Kishikawa, T., Li, Q., Gabel, H.W., Elewa, A., Baumgartner, B., Rose, D.J., Yu, H., Bosak, S., Sequerra, R., Fraser, A., Mango, S.E., Saxton, W.M., Strome, S., Van Den Heuvel, S., Piano, F., Vandenhaute, J., Sardet, C., Gerstein, M., Doucette-Stamm, L., Gunsalus, K.C., Harper, J.W., Cusick, M.E., Roth, F.P., Hill, D.E., Vidal, M., A map of the interactome network of the metazoan C.elegans, Science, 23;303(5657):540-3, 2004.

[20] Lodish, H., Berk, A., Matsudaira, P., Kaiser, C.A., Krieger, M., Scott, M.P., Zipursky, S.L.,Darnell, J., Molecular Cell Biology Fifth Edition, W.H. Freeman and Company, New York, 2004.

[21] Lötstedt P., Ferm L., Dimensional reduction of the Fokker-Planck equation for stochastic chemical reactions, Technical Report 2005-023, Dept. of Information Technology, Uppsala University,Uppsala, Sweden, 2005.

[22] Lötstedt P., Söderberg S., Ramage A., Hemmingsson-Fränden L., Implicit solution of hyperbolic equations with space-time adaptivity, BIT, 42, 2002, pp. 134-158.

[23] Moskowitz, B., Caflisch, R.E., Smoothness and dimension reduction in quasi-Monte Carlo methods, J. Math. Comput. Modeling, 23, 1996, pp.37-54.

[24] Owen, A.B., Monte Carlo extension of quasi-Monte Carlo, Proceedings of the 30th conference on Winter simulation, Washington, D.C., United States, 1998, pp. 571-578.

[25] Owen, A.B., Monte Carlo variance of scrambled net quadrature, SIAM J. Numer. Anal, 34, 1997, pp. 1884-1910.

[26] Owen, A.B., Scrambling Sobol´ and Niederriter-Xing Points, J. Complexity, 14, 1998, pp. 466-489.

[27] Paulsson, J., Berg, O.G., Ehrenberg, M., Stochastic focusing: Fluctuation-enhanced sensitivity of intracellular regulation, Proc Natl Acad Sci USA, 97, 2000, pp. 7148-7153.

[28] Press, W.H., Farrar, G.R., Recursive stratified sampling for multidimensional Monte Carlo integration, Computers in Physics, 1990, pp. 190-195.

[29] Puchalka J., Kierzek A.M., Bridging the gap between stochastic and deteriministic regimes in the kinetic simulations of the biochemical reaction networks, Biophys.,86, 2004, pp. 1357-1372.

[30] Rao, C.V., Wolf, D.M., Arkin, A.P., Control, exploitation and tolerace of intracellular noise, Nature, 420, 2002.

[31] Sjöberg P., Numerical Solution of the Fokker-Planck Approximation of the Chemical Master Equation, IT Licentiate thesis, Dept. of Information technology, Uppsala University, Uppsala, Sweden, 2005-010, 2005.

[32] Sjöberg P., Numerical Solution of the master equation in molecular biology, Master's thesis, Dept. of Information Technology, Uppsala University, 2002.

[33] Thattai, M, van Oudenaarden, A, Intrinsic noise in gene regulatory networks, Proc Natl Acad Sci USA, 98, 2001, pp. 8614-8619.

[34] Van Kampen, N.G., Stochastic Processes in Physics and Chemistry, North-Holland Personal Library,1992.

[35] Vilar J.M.G., Kueh H.Y., Barkai N., Leibler S., Mechanism of noise resistance in genetic oscillators, Proc Natl Acad Sci USA, 99, 2002, pp. 5988-5992.

[36] Werner M., Numerical solution of the master equation using linear noise approximation, Master's thesis, Dept. of Information Technology, Uppsala University, 2004.