Parallelising the finite state projection method

Vikram Sunkara[1] Markus Hegland[2]

(Received 31 January 2011; revised 26 August 2011)

Abstract

Many realistic mathematical models of biological and chemical systems, such as enzyme cascades and gene regulatory networks, need to include stochasticity. These systems are described as Markov processes and are modelled using the Chemical Master Equation. The Chemical Master Equation is a differential-difference equation (continuous in time and discrete in the state space) for the probability of a certain state at a given time. The state space is the population count of species in the system. A successful method for computing the Chemical Master Equation is the Finite State Projection Method. We give a new algorithm to distribute the Finite State Projection Method method onto multi-core systems. This method is called the Parallel Finite State Projection method. This article also analyses the theory needed for parallelisation of the Chemical Master Equation.

http://journal.austms.org.au/ojs/index.php/ANZIAMJ/article/view/3958 gives this article, Published October 13, 2011. ISSN 1446-8735. (Print two pages per sheet of paper.)

Contents

1 Introduction

The Chemical Master Equation (CME) describes systems in which multiple species are interacting with each other with some propensity or intensity. Recently the field of biology has found applications of the CME to model many biological processes where proteins and species interact in small numbers. Due to the small number of interactions, stochasticity plays a key role. Even though the individual populations are small, that is, $\ll 1000$, having many different species grows the set of possible states the system can be in to a very large number. Until 2005, it was considered only feasible to take stochastic simulation methods where we average over realisations to find the solution to the CME. Then we had the introduction of the Finite State Projection method, which helped compute problems with larger state spaces [5, 2, 6, 7].

The main focus of this article is to give the framework and theory for distributing the CME problem onto large computational infrastructure. Clusters are now accessible to most researchers. We want to be able to utilise this infrastructure to compute faster the CME based models. Showing success in demonstrating that we can compute systems with a few million possible states, would make larger problems more feasible. Hence the main focus is on

distributing the CME problem onto multiple *cores*. A single core is technically a single logic chipset.

The following sections show the theoretical foundation and theorems important for the design of multi-core based computation of the CME. Section 2 gives a brief background to the CME problem and the popular way of solving it, using the FSP. Section 3 introduces the Parallel Finite State Projection method (PFSP) and gives theorems necessary for that method. Section 4 finishes with a discussion on what we expect and should analyse from the current implementation of PFSP using the CMEPy [1], a Python based CME library.

We do not discuss time comparisons of the PFSP method. This is because the focus of the article is to give the framework for expanding the CME for multi-core computation. We implemented the PFSP method in code; however, it is not at a stage for benchmarking. This is further discussed in the last section. Let us start by looking into the CME problem and the FSP.

2 The finite state projection method

Without any loss of generality the Chemical Master Equation is described as the ODE problem

$$\frac{dp_t}{dt} = Ap_t\,, \tag{1}$$

with initial condition $p_0 = (1, 0, \ldots)$. The generator matrix A has the properties

- the diagonal elements are negative,
- the off-diagonal elements are all positive,
- the columns sum up to zero.

The dimension of A is the size of the state space, Ω. The vector p_t is a probability vector where each coordinate corresponds to a state in the state

space and its value describes the probability of the system being in that state at time $t > 0$.

In many applications we find that state spaces are very large; however, the number of states which contribute significantly in probability is far fewer. The Finite State Projection method (FSP), formulated by Mansky and Khammash [5], takes advantage of this effect.

For the purpose of this article we generalise the FSP to the following form: let Ω be the state space of problem (1). Given a tolerance $\varepsilon > 0$, the FSP method finds a sub matrix A^J of A such that $\|e^{At}p_0 - e^{A^J t}p_0\|_1 < \varepsilon$. Furthermore, the FSP approximation has the following key properties: let $\bar{p}_t$ be the FSP approximation at time t of p_t, then

- $\bar{p}_t$ has finite support, and
- $0 \leqslant \bar{p}_t(x) \leqslant p_t(x)$ for all $x \in \Omega$.

The Finite State Projection method has been shown to be a successful method for solving the CME and was found to be faster then the conventional stochastic simulation method [6, 4]. However, there was the following problem: the FSP ensures the existence of a finite state space we can project onto; but it does not inform us where it is. In applications where the factors that generate A are non-linear, predicting the state space is difficult, at times requires brute force. This problem led to the work by Sunkara and Hegland [7] giving the Optimal Finite State Projection method (OFSP), which is known to be *order optimal* [7]. The OFSP gives control over where the state space is and where it might be drifting, helping us tackle bigger problems where the location of the states with probabilities away from zero is unknown.

The major motivation for computing the CME is the presence of new applications arising in biology. For example, signalling cascades, tissue pigmentation, chlamydia and neuron networks, where researchers are interested in the stochastic interactions of small groups of particles. Moreover, they are interested in the probability distributions over what is possible, rather than expectations. These problems have more than a few million possible states. In

such applications it becomes impractical to compute the solution on a single core. In the next section 3 we introduce a new method called the Parallel Finite State Projection method (PFSP), which describes a way of parallelising the FSP method onto multiple computers. We also provide theory which can be used to extend all other possible methods for solving the FSP, into a parallel setting.

3 The parallel finite state projection method

We describe a new algorithm for solving the CME, The Parallel Finite State Projection method (PFSP), where we distribute the computation over several cores. We utilise the linearity of the CME to distribute smaller problems onto multiple cores, which are solved and gathered, not introducing any approximation to the CME structure.

We introduce some general notation to describe parts of the CME on different cores. We use the notation superscript k when we refer to vectors, state spaces and errors on the kth core. To distribute the problem onto multiple cores we define a partitioning map $Z : \Omega \mapsto \{\Omega^k : k = 1, \ldots, K\}$ where $\cup_{k=1}^{K} \Omega^k = \Omega$ with the conditions $\Omega^k \cap \Omega^j = \emptyset$ if $k \neq j$, and $\Omega^k \subset \Omega$ for all k. Now to map the CME problem onto multiple cores, we define a *projection* of p_t, the probability vector, onto Ω^k by

$$p_t^k(\mathbf{x}) := \begin{cases} p_t(\mathbf{x}), & \text{if } \mathbf{x} \in \Omega^k, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Clearly $p_t = \sum_{k=1}^{K} p_t^k$. A simple application of the linearity of the CME (1) shows that if we evolve p_t^k for a time step of Δt by solving $dp_t^k/dt = Ap_t^k$ to obtain $p_{t+\Delta t}^k = e^{A\,\Delta t} p_t^k$, separately for each k, then the $\sum_{k=1}^{K} e^{A\Delta t} p_t^k$ is the same as evolving p_t to $e^{A\Delta t} p_t$. This is seen in

$$p_{t+\Delta t} = e^{A\Delta t} p_t = e^{A\Delta t} \sum_{k=1}^{K} p_t^k = \sum_{k=1}^{K} p_{t+\Delta t}^k \,.$$

Algorithm 1: PFSP Master Algorithm

input : p_0, K, ϵ, $\epsilon^1, \ldots, \epsilon^K$, ϵ_c, t, Δt, h
output: $\tilde{p}_{t+h\Delta t}$

begin
  $\bar{p}_t, \bar{\Omega}_t \leftarrow \text{OFSP}(t, \epsilon)$
  for $i \leftarrow 1$ ***to*** h **do**
    $\bar{p}_t^1, \bar{p}_t^2, \ldots \bar{p}_t^K \leftarrow \texttt{projection}(\bar{p}_t, Z(\bar{\Omega}_t, K))$
    for $k \leftarrow 1$ ***to*** K **do**
      `send` $(\bar{p}_t^k, \epsilon^k, \Delta t)$ to Core k
      $\hat{p}_{t+\Delta t}^k \leftarrow$ `receive` from Core k
    end
    $t \leftarrow t + i\Delta t$
    $\bar{p}_t, \bar{\Omega}_t \leftarrow \texttt{compress}\ (\sum_{k=1}^K \hat{p}_{t+\Delta t}^k, \epsilon_c)$
  end
  `return` $\bar{p}_t$
end

We see that the support of any two $p_{t+\Delta t}^k$ vectors can overlap. Now we have split the problem into K pieces without any change to the CME. Hence a simple way of parallelising the CME is to partition the state space into non-intersecting subsets, Ω^k, and project the CME problem (1) onto each Ω^k. Then we compute these smaller problems independently and collate the probabilities at the end by simply adding all the vectors together. Algorithm 1 and Algorithm 2 describe the Parallel Finite State Projection method. The rest of this article is an in-depth construction of the PFSP and a demonstration of how the error evolves in this method.

We want to ensure that on each core, p^k will bound our approximation $\hat{p}^k$. We demonstrate this in the following lemma and theorems. The key structure we repeatedly use is the linearity of the CME.

Lemma 1. *Let* p *be a probability vector which is a solution to* (1)*. If* $\hat{p}$ *is a*

Algorithm 2: PFSP Slave Algorithm

input : $\hat{p}_t^k, \epsilon^k, \Delta t$
output: $\hat{p}_{t+\Delta t}^k$

```
1  begin
2      find A^k, a submatrix of A
3      Γ ← e^{A^k Δt} p̂_t^k
4      if Γ > ||p̂_t^k||_1 − ε^k then
5          stop
6      else
7          Pick a bigger A^k and go to 2
8      end
9      return p̂_{t+Δt}^k
10 end
```

vector which is positive and bounded element-wise by p *and* e^{At} *is monotone, then for any* $t > 0$, $e^{At}(p - \hat{p})$ *is positive element-wise.*

Proof: Write p as the sum of two positive vectors, where one is $\hat{p}$. Then normalise the vectors and use the property of e^{At}. ♠

Extending the above lemma gives

$$0 \leqslant (e^{At}\hat{p})(\mathbf{x}) \leqslant (e^{At}p)(\mathbf{x}) \quad \text{for all } \mathbf{x} \in \Omega\,. \tag{3}$$

Theorem 2. *Let* p_t *be a solution to the* CME *at some time* t, *and let* $\hat{p}_t$ *be its approximation, where* $0 \leqslant \hat{p}_t \leqslant p_t$ *element-wise. Applying the* Z *mapping on both vectors we get* $\hat{p}_t = \sum_{i=1}^{K} \hat{p}_t^k$ *and* $p_t = \sum_{i=1}^{K} p_t^k$. *If* A *is the infinitesimal generator as in* (1), *then for any* $\Delta t > 0$,

$$\left\| e^{A\Delta t}p_t - e^{A\Delta t}\hat{p}_t \right\|_1 = \sum_{i=1}^{K} \left\| e^{A\Delta t}p_t^k - e^{A\Delta t}\hat{p}_t^k \right\|_1, \tag{4}$$

where p^k *is defined as in* (2).

Proof: Using Lemma 1, $e^{At}p^k - e^{At}\hat{p}^k$ is positive element-wise for each k. ♠

Theorem 3. *Let* $p \in \ell_1$ *with* $\|p\|_1 = 1$. *If* A *is a matrix whose columns sum to zero with only diagonal negative terms and* $e^{At} : \ell_1 \mapsto \ell_1$ *is a matrix with the property that its columns have positive real numbers and sum up to one, then*

$$\left\|e^{At}p\right\|_1 \leqslant \|p\|_1 .$$

Proof: This theorem was proved by Engblom [3]. ♠

We look at the error associated with distributing the probability over multiple cores and computing them independently. We are interested in errors which arise in steps 2, 10 and 6 of Algorithm 1, and in steps 2 and 4 of Algorithm 2.

The first step of the PFSP method is that we compute the problem linearly on a single core until some time $t > 0$. Let $\bar{p}_t$ denote the OFSP solution of (1) for some error ϵ, that is,

$$\|p_t - \bar{p}_t\|_1 < \epsilon . \tag{5}$$

Now let $\operatorname{supp} \bar{p}_t = \bar{\Omega}_t$. Applying $Z(\bar{\Omega}_t, K)$ gives K distinct subsets, $\bar{\Omega}_t^k$, where $k = 1, \ldots, K$. Define $\bar{p}_t^k$, for $k = 1, \ldots, K$, as the projection (2) of $\bar{p}_t$ onto $\bar{\Omega}_k^t$.

We distribute the $\bar{p}_t^k$ vectors onto K different cores and on each core solve the problem

$$\frac{d\hat{p}_s^k}{ds} = A^k \hat{p}_s^k , \tag{6}$$

with initial condition $\bar{p}_t^k$. If $0 < \epsilon^k < 1$, then [5] there exists a FSP approximation A_k such that

$$\left\| e^{A\Delta t}\bar{p}_t^k - e^{A_k\Delta t}\bar{p}_t^k \right\|_1 < \epsilon^k. \tag{7}$$

Consider the error if (6) is evolved on each of the cores and then was collated. Let $p_{t+\Delta t}$ be the real solution to (1) and $\hat{p}_{t+\Delta t} := \sum_{k=1}^{K} \hat{p}_{t+\Delta t}^k$. Then

$$\begin{aligned}
\|p_{t+\Delta t} - \hat{p}_{t+\Delta t}\|_1 &\leqslant \left\| e^{A\Delta t}p_t - e^{A\Delta t}\bar{p}_t \right\|_1 + \left\| e^{A\Delta t}\bar{p}_t - \hat{p}_{t+\Delta t} \right\|_1 \\
&= \left\| e^{A\Delta t}p_t - e^{A\Delta t}\bar{p}_t \right\|_1 + \left\| e^{A\Delta t}\bar{p}_t - \sum_{k=1}^{K} \hat{p}_{\Delta t}^k \right\|_1 \\
&= \left\| e^{A\Delta t}p_t - e^{A\Delta t}\bar{p}_t \right\|_1 + \left\| \sum_{k=1}^{K} e^{A\Delta t}\bar{p}_t^k - \sum_{k=1}^{K} \hat{p}_{\Delta t}^k \right\|_1 .
\end{aligned}$$

Using Theorem 3, (5), (6) and (7) the above simplifies to

$$\|p_{t+\Delta t} - \hat{p}_{t+\Delta t}\|_1 \leqslant \sum_{k=1}^{K} \epsilon^k + \epsilon\,. \tag{8}$$

When computing, ϵ^k and ϵ are found by introducing a *sink state* [5]. A sink state is a fictitious state into which the probability on the boundary of the truncated state space flows.

If (6) is evolved for too large a time step Δt, then $\operatorname{supp} \hat{p}_{\Delta t}^k$ starts to become as big as $\Omega_{t+\Delta t}$. This implies a slow down in the speed of computation. To prevent this from happening a *trading step* is introduced.

In the trading step the master core collates all the solution vectors computed on the slave cores. Once that is done, the state space of the support of the collated vectors is then reclustered into new non-intersecting subsets of the state space. The problem is reprojected onto these new clusters and all overlaps have been removed. We now demonstrate the trading step.

Let us trade at time $t + \Delta t$, where we have evolved on a single core to time t using the OFSP method and computed time step Δt on the separate cores. To trade we define a new partition over the union of the state spaces on all the cores. Let

$$\hat{\Lambda}_{\Delta t} := \bigcup_{k=1}^{K} \operatorname{supp} \hat{p}^k_{\Delta t}.$$

Applying Z to $\hat{\Lambda}_{\Delta t}$ gives us K non-intersecting clusters of the state space, $\hat{\Lambda}^k_\Delta$, where $k = 1 \dots K$. We reset problem (6) to having the initial condition as

$$\bar{p}^k_{t+\Delta t}(\mathbf{x}) = \begin{cases} \sum_{k=1}^{K} \hat{p}^k_{\Delta t}(\mathbf{x}), & \text{if } \mathbf{x} \in \hat{\Lambda}_{\Delta t}, \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

on each of the K cores.

The final step is to compress each $\hat{p}_{t+\Delta t}$ to its best N-term approximation for error ϵ_c, that is,

$$\|\hat{p}_{t+\Delta t} - \tilde{p}_{t+\Delta t}\| < \epsilon_c, \tag{10}$$

where $\tilde{p}_{t+\Delta t}$ is our N-term approximation of $\hat{p}_{t+\Delta t}$.

4 Discussion

Figure 1 shows the number of computational states used in the OFSP and PFSP methods. The system was a two dimensional birth process. We computed up to a final time of 1000 seconds. In the PFSP we traded every 32 steps and we distributed the problem over four cores. The error at each point in time is the same for the OFSP method and the PFSP method. The key features are that on every core the state space being computed is much smaller. This gives us some evidence of the type of reduction that we can get. However, this was only 1.5 times faster than the single core method. The implementation of the PFSP used the CMEPy library [1] which is designed for fast computing but on a

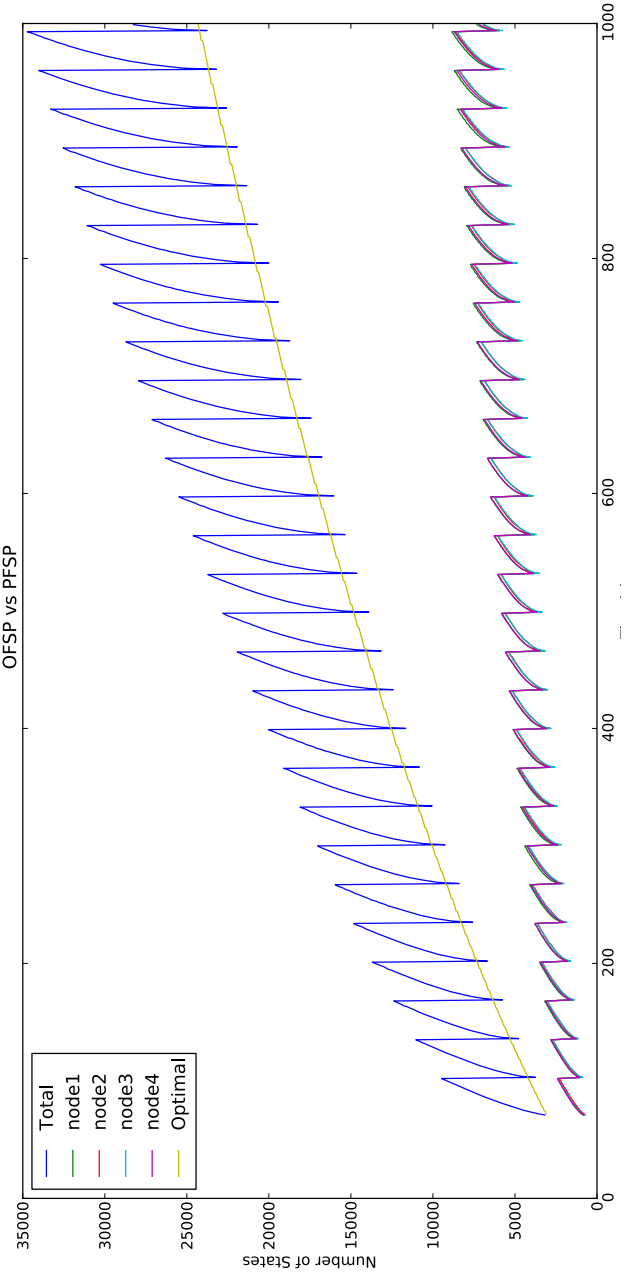

Figure 1: OFSP versus PFSP.

single core; however, it is not designed to distribute the state space to multiple cores, hence slowing down the solver. For future work we would like to build a CMEPy module whose data structures are customised for parallel computing. Then we would look at the impact and speedup when we distribute the state space over multiple cores. Also for further research we would use the theory given in this article to extend some other existing CME solution methods into solvers which compute over large computing infrastructures.

References

[1] CMEPy, Jan 2011. https://github.com/fcostin/cmepy C855, C862

[2] K. Burrage, M. Hegland, S. MacNamara, and R.B. Sidje. A Krylov-based finite state projection algorithm for solving the chemical master equation arising in the discrete modeling of biological systems. *In: Langville, A. N., Stewart, W. J. (Eds.), Proceedings of the 150th Markov Anniversary Meeting, Boson Books*, pp. 21–38, 2006. C854

[3] S. Engblom. *Numerical Solution Methods in Stochastic Chemical Kenetics*. PhD thesis, Uppsala University, 2008. C860

[4] M. Hegland, A. Hellander, and P. Lotstedt. Sparse grid and hybrid methods for the chemical master equation. *BIT Numerical Mathematics*, 48(2):265–283, 2008. doi:10.1007/s10543-008-0174-z C856

[5] M. Khammash and B. Munsky. The finite state projection algorithm for the solution of the chemical master equation. *Journal of Chemical Physics*, 124(044104):1–12, 2006. doi:10.1063/1.2145882 C854, C856, C861

[6] S. MacNamara, A.M. Bersani, K. Burrage, and R.B. Sidje. Stochastic chemical kinetics and the total quasi-steady-state assumption: application to the stochastic simulation algorithm and chemical master

equation. *Journal of Chemical Physics*, 129(095105):1–13, 2008. doi:10.1063/1.2971036 C854, C856

[7] V. Sunkara and M. Hegland. An optimal finite state projection method. *Procedia Computer Science*, 1(1):1579–1586, 2010. ICCS 2010. http://www.sciencedirect.com/science/article/pii/S187705091000178X, doi:10.1016/j.procs.2010.04.177 C854, C856

Author addresses

1. **Vikram Sunkara**, Centre of Mathematics and its Applications, Mathematical Sciences Institute, The Australian National University, Australia.
mailto:Vikram.Sunkara@anu.edu.au

2. **Markus Hegland**, Centre of Mathematics and its Applications, Mathematical Sciences Institute, The Australian National University, Australia.
mailto:Markus.Hegland@anu.edu.au