The University of Queensland

A Krylov-based FSP algorithm for solving the chemical master equation arising in the discrete modelling of biological systems

Roger B. Sidje
(rbs@maths.uq.edu.au)

Advanced Computational Modelling Centre
Department of Mathematics
University of Queensland
Brisbane, Australia

Contents

- Stochastic models for biochemical systems
- The Finite State Projection Algorithm (FSP)
- The Krylov FSP
- Numerical Results

Joint work with

- Kevin Burrage (UQ)
- Markus Hegland (ANU)
- Shev MacNamara (UQ)

Chemical reaction systems

- Well-stirred mixture.
- N molecular species: $S_1, \ldots, S_N$.
- Constant temperature, fixed volume Ω
- M reaction channels: $R_1, \ldots, R_N$

  $R : c_1 S_1 + \ldots c_N S_N \xrightarrow{\kappa} b_1 S_1 + \ldots b_N S_N$

  κ: reaction rate constant.
  c_i, b_i: coefficients of reactants&products
- Dynamical state: $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$, $\boldsymbol{x}(t) = (x_1(t), \ldots, x_N(t))^T$, where $x_i(t)$ is the number of S_i molecules in the system.
- Propensity function: $\alpha_j(\boldsymbol{x})dt$ = the probability, given $\boldsymbol{x}(t) = \boldsymbol{x}$, that one reaction R_j will occur in Ω in $[t, t+dt)$.
- Stochiometric vector: $\boldsymbol{\nu}_j$. When R_j occurs, $\boldsymbol{\nu}_j$ is used to update the state of the system by $\boldsymbol{x}(t+dt) = \boldsymbol{x}(t) + \boldsymbol{\nu}_j$. $\boldsymbol{\nu} = (b_1 - c_1, \ldots, b_N - c_N)^T$.

Chemical Master Equation (CME)

Let $P(\boldsymbol{x};t)$ be the probability of being in state $\boldsymbol{x}$ at time t. Then it satisfies the discrete parabolic PDE

$$\frac{\partial P(\boldsymbol{x};t)}{\partial t} = \sum_{j=1}^{M} \alpha_j(\boldsymbol{x}-\boldsymbol{\nu}_j)P(\boldsymbol{x}-\boldsymbol{\nu}_j;t) - P(\boldsymbol{x};t)\sum_{j=1}^{M}\alpha_j(\boldsymbol{x})$$

ODE formulation

Let $\boldsymbol{p}(t)$ be a vector of probabilities $P(\boldsymbol{x};t)$, indexed by the states $\boldsymbol{x}$. Then

$$\dot{\boldsymbol{p}}(t) = \boldsymbol{A}\boldsymbol{p}(t)$$

where $\boldsymbol{A} = [a_{ij}]$ is the transition matrix populated by the propensities. The rows and columns of the matrix are indexed by the states, so the states have now been enumerated.

Solution techniques

- Stochastic Simulation Algorithm (SSA): evolve $\boldsymbol{x}(t)$ by $\boldsymbol{x}(t+dt) = \boldsymbol{x}(t) + \boldsymbol{\nu}_j$. Trajectories.
- CME Solver: $\boldsymbol{p}(t) = \exp(t\boldsymbol{A})\boldsymbol{p}_0$.

Issues

- Large state space
- Long integration domain

Finite State Projection Algorithm

FSP [Munsky/Khammash (J. Chem. Phys., 2006)]

$$\boldsymbol{A} = \left(\begin{array}{c|c} \boldsymbol{A}_k & * \\ \hline * & * \end{array}\right)$$

$$e^{t_f \boldsymbol{A}} \boldsymbol{v} \approx e^{t_f \boldsymbol{A}_k} \boldsymbol{v}_k$$

The states indexed by $\{1, \ldots, k\}$ form the *finite state projection*, denoted $\boldsymbol{X}_k$.

Finite State Projection Algorithm

FSP [Munsky/Khammash (J. Chem. Phys., 2006)]

$$\boldsymbol{A} = \left(\begin{array}{c|c} \boldsymbol{A}_k & * \\ \hline * & * \end{array}\right)$$

$$e^{t_f \boldsymbol{A}} \boldsymbol{v} \approx e^{t_f \boldsymbol{A}_k} \boldsymbol{v}_k$$

The states indexed by $\{1, \ldots, k\}$ form the *finite state projection*, denoted $\boldsymbol{X}_k$.

Theorem. Let
$\Gamma_k = \mathbb{1}^T e^{t_f \boldsymbol{A}_k} \boldsymbol{p}_k(0), \quad \mathbb{1} = (1, ..., 1)^T$
Then

- $\Gamma_k \leq \Gamma_{k+1} \leq ... \leq 1.$
- If $\Gamma_k \geq 1 - \epsilon$ then

$$\left(\begin{array}{c} e^{t_f \boldsymbol{A}_k} \boldsymbol{p}_k(0) \\ \mathbf{0} \end{array}\right) \leq \boldsymbol{p}(t_f) \leq \left(\begin{array}{c} e^{t_f \boldsymbol{A}_k} \boldsymbol{p}_k(0) \\ \mathbf{0} \end{array}\right) + \epsilon \mathbb{1}.$$

FSP Algorithm

ALGORITHM: FSP$(\boldsymbol{A}, \boldsymbol{p}_0(0), t_f, \epsilon, \boldsymbol{X}_0)$

$\boldsymbol{A}_0 := submatrix(\boldsymbol{X}_0)$;

$\Gamma_0 := \mathbb{1}^T \exp(t_f \boldsymbol{A}_0) \boldsymbol{p}_1(0)$;

for $k := 1, 2, \ldots$ **until** $\Gamma_k \geq 1 - \epsilon$ **do**

$\quad \boldsymbol{X}_k := expand(\boldsymbol{X}_{k-1})$;

$\quad \boldsymbol{A}_k := submatrix(\boldsymbol{X}_k)$;

$\quad \Gamma_k := \mathbb{1}^T \exp(t_f \boldsymbol{A}_k) \boldsymbol{p}_k(0)$;

endfor

return $\exp(t_f \boldsymbol{A_k}) \boldsymbol{p}_k(0)$;

FSP Algorithm

ALGORITHM: FSP$(\boldsymbol{A}, \boldsymbol{p}_0(0), t_f, \epsilon, \boldsymbol{X}_0)$
$\boldsymbol{A}_0 := submatrix(\boldsymbol{X}_0)$;
$\Gamma_0 := \mathbb{1}^T \exp(t_f \boldsymbol{A}_0)\boldsymbol{p}_1(0)$;
for $k := 1, 2, \ldots$ **until** $\Gamma_k \geq 1 - \epsilon$ **do**
    $\boldsymbol{X}_k := expand(\boldsymbol{X}_{k-1})$;
    $\boldsymbol{A}_k := submatrix(\boldsymbol{X}_k)$;
    $\Gamma_k := \mathbb{1}^T \exp(t_f \boldsymbol{A}_k)\boldsymbol{p}_k(0)$;
endfor
return $\exp(t_f \boldsymbol{A_k})\boldsymbol{p}_k(0)$;

Issues

- Implemented via MATLAB's *expm*
- No intermediate time points
- Repeated $e^{t_f \boldsymbol{A}_k}\boldsymbol{p}_k(0)$ with the *same* t_f

The new Krylov FSP

- Krylov-based instead of *expm*

$$e^{\tau \boldsymbol{A}}\boldsymbol{v} \approx \beta \boldsymbol{V}_{m+1} \exp\left(\tau\, \overline{\boldsymbol{H}}_{m+1}\right)\boldsymbol{e}_1$$

- Embbeded exponentials

$$e^{t\boldsymbol{A}}\boldsymbol{v} \approx e^{\tau_k \boldsymbol{A}_k} \ldots e^{\tau_0 \boldsymbol{A}_0}\boldsymbol{v}$$

- Inexact/relaxed matrix-vector products

Embbeded exponentials

$$\boldsymbol{p}(t_f) \approx e^{\tau_K \boldsymbol{A}_K} \ldots e^{\tau_0 \boldsymbol{A}_0} \boldsymbol{p}(0), t_f = \sum_{k=0}^{K} \tau_k$$

- Harness Expokit's step-by-step scheme.
- The $\boldsymbol{A}_k$ changes between internal steps!

Integration scheme (dense output)

$0 \equiv t_0 < t_1 < \cdots < t_K < t_{K+1} \equiv t_f$

$$\boldsymbol{p}(t_{k+1}) = \boldsymbol{p}(t_k + \tau_k) = e^{\tau_k \boldsymbol{A}_k} \boldsymbol{p}(t_k).$$

Embbeded exponentials

$$\boldsymbol{p}(t_f) \approx e^{\tau_K \boldsymbol{A}_K} \ldots e^{\tau_0 \boldsymbol{A}_0} \boldsymbol{p}(0), t_f = \sum_{k=0}^{K} \tau_k$$

- Harness Expokit's step-by-step scheme.
- The $\boldsymbol{A}_k$ changes between internal steps!

Integration scheme (dense output)

$0 \equiv t_0 < t_1 < \cdots < t_K < t_{K+1} \equiv t_f$

$$\boldsymbol{p}(t_{k+1}) = \boldsymbol{p}(t_k + \tau_k) = e^{\tau_k \boldsymbol{A}_k} \boldsymbol{p}(t_k).$$

Krylov FSP criteria

- $\Gamma_k = \mathbb{1}^T \exp(\tau_k \boldsymbol{A}_k) \boldsymbol{p}(t_k) \geq 1 - \epsilon \frac{t_k}{t_f}$

  $\tau_k := \tau_k/2$ until the criteria is met.
- Expand the FSP

Embbeded exponentials

$$\boldsymbol{p}(t_f) \approx e^{\tau_K \boldsymbol{A}_K} \dots e^{\tau_0 \boldsymbol{A}_0} \boldsymbol{p}(0), t_f = \sum_{k=0}^{K} \tau_k$$

- Harness Expokit's step-by-step scheme.
- The $\boldsymbol{A}_k$ changes between internal steps!

Integration scheme (dense output)

$0 \equiv t_0 < t_1 < \cdots < t_K < t_{K+1} \equiv t_f$

$$\boldsymbol{p}(t_{k+1}) = \boldsymbol{p}(t_k + \tau_k) = e^{\tau_k \boldsymbol{A}_k} \boldsymbol{p}(t_k).$$

Krylov FSP criteria

- $\Gamma_k = \mathbb{1}^T \exp(\tau_k \boldsymbol{A}_k) \boldsymbol{p}(t_k) \geq 1 - \epsilon \frac{t_k}{t_f}$

  $\tau_k := \tau_k/2$ until the criteria is met.
- Expand the FSP

Accuracy criteria

- Select τ_k for a good approximation
- Integrate without expanding the FSP

Inexact matrix-vector product

Don't use the finite state projection $\boldsymbol{X}_k$ entirely. Only capture an approximation:

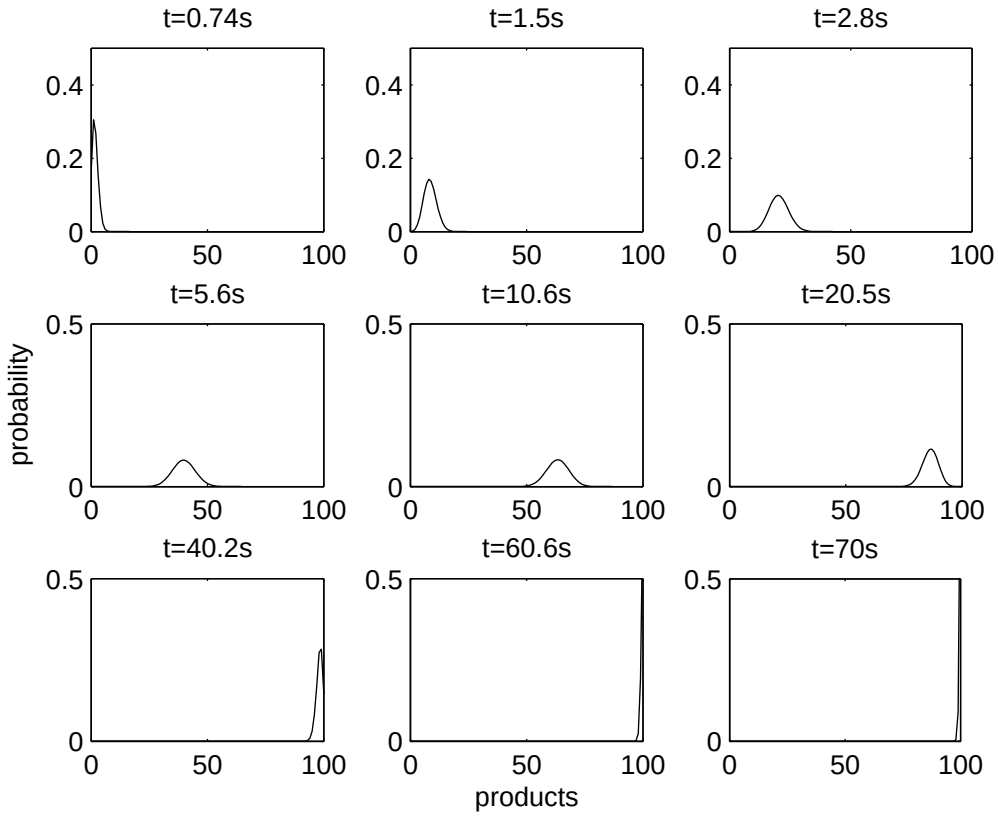
$$\begin{aligned}
\boldsymbol{A}_k\boldsymbol{v} &= \begin{bmatrix} \boldsymbol{a}_1^{(k)} & \boldsymbol{a}_2^{(k)} & \dots \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \end{bmatrix} \\
&\approx \sum_{i \in \hat{\boldsymbol{X}}_k} v_i \boldsymbol{a}_i^{(k)} \\
&= \hat{\boldsymbol{A}}_k\boldsymbol{v}.
\end{aligned}$$

Inexact matrix-vector product

Do no the entire finite state projection $\boldsymbol{X}_k$.

Capture only an approximation:

$$\begin{aligned}\boldsymbol{A}_k\boldsymbol{v} &= \begin{bmatrix}\boldsymbol{a}_1^{(k)} & \boldsymbol{a}_2^{(k)} & \dots\end{bmatrix}\begin{bmatrix}v_1\\ v_2\\ \vdots\end{bmatrix}\\ &\approx \sum_{i\in\hat{\boldsymbol{X}}_k} v_i\boldsymbol{a}_i^{(k)}\\ &= \hat{\boldsymbol{A}}_k\boldsymbol{v}.\end{aligned}$$



Krylov FSP tracks the support.

Krylov FSP Algorithm

ALGORITHM: KrylovFSP($\boldsymbol{A}, \boldsymbol{p}_0(0), t_f, \epsilon, \boldsymbol{X}_0, m, tol$)

$t_0 := 0$; $\hat{\boldsymbol{X}}_0 := \boldsymbol{X}_0$;

$\boldsymbol{A}_0 := submatrix(\boldsymbol{X_0})$; $\hat{\boldsymbol{A}}_0 := \boldsymbol{A}_0$;

expandFSP := FALSE;

for $k := 0, 1, 2, ...$ **until** $t_k = t_f$ **do**

  $[\boldsymbol{V}_{m+1}, \overline{\boldsymbol{H}}_{m+1}] := Arnoldi\ (\hat{\boldsymbol{A}}_k, \boldsymbol{p}_k(t_k), m)$;

  repeat

    $\tau_k :=$ step-size;

    $\boldsymbol{p}_k(t_k) := \beta \boldsymbol{V}_{m+1} \exp(\tau_k \overline{\boldsymbol{H}}_{m+1}) \boldsymbol{e}_1$;

    err := numerical-error-estimate;

  until err $\leq 1.2\ tol$;

  $\Gamma_k := \mathbb{1}^T \boldsymbol{p}_k(t_k)$;

  while $\Gamma_k < 1 - \epsilon \frac{t_k + \tau_k}{t_f}$ **do**

    expandFSP := TRUE;

    $\tau_k := \frac{1}{2} \tau_k$;

    $\boldsymbol{p}_k(t_k) := \beta \boldsymbol{V}_{m+1} \exp(\tau_k \overline{\boldsymbol{H}}_{m+1}) \boldsymbol{e}_1$;

    $\Gamma_k := \mathbb{1}^T \boldsymbol{p}_k(t_k)$;

  endwhile

  if expandFSP

    $\boldsymbol{X}_{k+1} := expand(\boldsymbol{X}_k)$; $\hat{\boldsymbol{X}}_{k+1} := select(\boldsymbol{X}_{k+1})$;

    $\boldsymbol{A}_{k+1} := submatrix(\boldsymbol{X}_{k+1})$; $\hat{\boldsymbol{A}}_{k+1} := submatrix(\hat{\boldsymbol{X}}_{k+1})$;

    expandFSP := FALSE;

  endif

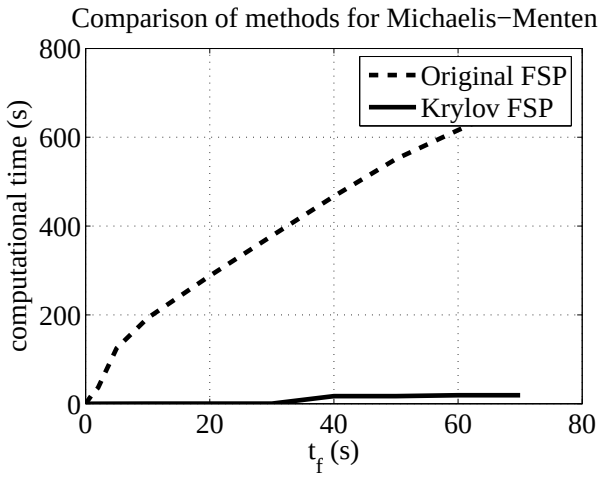
  $t_{k+1} := t_k + \tau_k$;

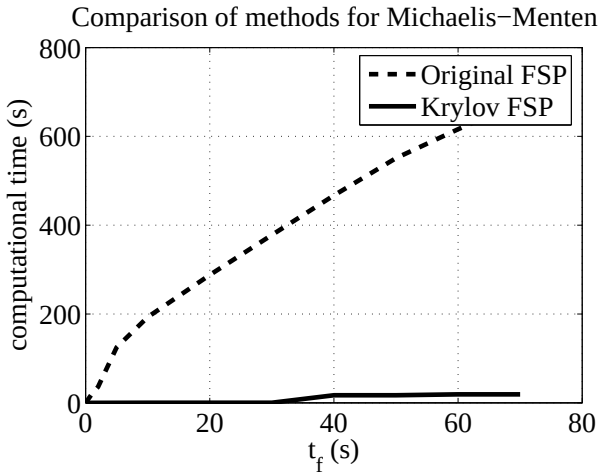
endfor

return $\boldsymbol{p}(t_k)$

Numerical Results

Tests on a SGI Altix (with $tol = 10^{-5}$)





The Michaelis-Menten enzyme kinetics involve four species, substrates (S), enzymes (E), enzyme-substrate complexes (ES) and products (P), interacting through three chemical reactions:

$$\mathrm{S} + \mathrm{E} \underset{\kappa_2}{\overset{\kappa_1}{\rightleftarrows}} \mathrm{ES} \xrightarrow{\kappa_2} \mathrm{E} + \mathrm{P}$$

Conlcusion

- Krylov FSP is *much* faster, albeit sometimes less accurate.
- It is better at memory management.
- Dense output is readily available.
- Best suited for problems where the support
  - spreads slowly
  - confined.
- Large t_f is a challenge for all methods.