Introduction to numerical techniques for stochastic systems with multiple time-scales

Eric Vanden-Eijnden

Courant Institute

Consider

$$\begin{cases} \dot{X} = f(X,Y), & X_{t=0} = x \in \mathbb{R}^n \\ \dot{Y} = \varepsilon^{-1} g(X,Y), & Y_{t=0} = y \in \mathbb{R}^m. \end{cases} \tag{25}$$

Recall that if the equation for Y at fixed $X = x$ defines an Markov process which is ergodic for every x with respect to the probability distribution

$$d\mu_x(y)$$

and

$$F(x) = \int_{\mathbb{R}^m} f(x,y) d\mu_x(y) \quad \text{exists}$$

then in the limit as $\varepsilon \to 0$ the evolution for X solution of (25) is governed by

$$\dot{X} = F(X) \qquad X_{t=0} = x. \tag{26}$$

In addition

$$F(x) = \int_{\mathbb{R}^m} f(x,y) d\mu_x(y) = \lim_{T\to\infty} \frac{1}{T} \int_0^T f(x, Y_t^x) dt \tag{27}$$

where

$$\dot{Y}^x = g(x, Y^x)$$

Basic multiscale algorithm

Use

$$X_{n+1} = X_n + \Delta t\ \tilde{F}_n, \qquad X_0 = X_{t=0} \tag{28}$$

Here $\tilde{F}_n$ is an approximation of $F(X_n)$ obtained as

$$\tilde{F}_n = \frac{1}{M} \sum_{m=M_T}^{M+M_T-1} f(X_n, Y_{n,m}) \tag{29}$$

where

$$Y_{n,m+1} = Y_{n,m} + \Delta\tau\ g(X_n, Y_{n,m}), \quad Y_{0,0} = Y_{t=0},\ Y_{n+1,0} = Y_{n,M+M_T-1} \tag{30}$$

Why is this better?

Basically, because M and M_T are $O(1)$ in ε! In other words, one can reach an $O(1)$ time-scale with a $O(1)$ number of steps.

In contrast, the direct scheme

$$\begin{cases} X_{p+1} = X_p + \delta t\ f(X_p, Y_p), & X_0 = X_{t=0} \\ Y_{p+1} = Y_p + \dfrac{\delta t}{\varepsilon}\ g(X_p, Y_p), & Y_0 = Y_{t=0} \end{cases} \tag{31}$$

takes $O(\varepsilon^{-1})$ steps to reach an $O(1)$ time-scale.

Efficiency depends on how large M_T and M needs to be.
Recall that

$$\tilde{F}_n = \frac{1}{M} \sum_{m=M_T}^{M+M_T-1} f(X_n, Y_{n,m}) \tag{32}$$

M_T tampers the relaxation errors;

M controls the size of the time-averaging window.

The nice surprise: the scheme works even if $M = 1$ (no time averaging) provided only that $\Delta t \ll 1$, $\Delta\tau \ll 1$ and $\Delta\tau M_T/\Delta t \gg 1$.

In fact, to achieve an error tolerance λ on X_t, i.e. in order that

$$\sup_{0\le n\le T/\Delta t} \mathbb{E}|\phi(X(n\Delta t)) - \phi(X_n)| \le \lambda$$

one must take

$$\Delta t = C_{\phi,T}\lambda, \qquad \Delta\tau = C_{\phi,T}\lambda, \qquad M_T = C'_{\phi,T}\frac{\Delta t}{\Delta\tau}\min(\lambda^{-1}, \varepsilon^{-1})$$

Why is time-averaging unnecessary (since in this case the approximation on $\tilde{F}_n$ is very bad at each time step)?

A simple illustrative example:

$$\begin{cases} \dot{X} = -Y \\ \dot{Y} = -\frac{1}{\varepsilon}(Y - X) + \frac{1}{\sqrt{\varepsilon}}\dot{W}(t) \end{cases} \tag{33}$$

Here

$$d\mu_x(y) = \frac{e^{-(y-x)^2}}{\sqrt{\pi}} \tag{34}$$

and so the limiting equation is

$$\dot{X} = -X \tag{35}$$

Use

$$X_{n+1} = X_n + \Delta t\big(- X_n + \frac{\xi_n}{\sqrt{2}}\big),$$
$$\xi_n = \text{independent } N(0,1)$$

Then

$$X_n = x(1 - \Delta t)^n + \frac{\Delta t}{\sqrt{2}} \sum_{j=1}^{n-1} (1 - \Delta t)^j \xi_{n-j}$$

and

$$\mathbb{E}|X_n - x(1 - \Delta t)^n|^2 = \frac{\Delta t^2}{2} \sum_{j=1}^{n-1} (1 - \Delta t)^{2j} = O(\Delta t) \qquad [n = O(\Delta t^{-1})]$$

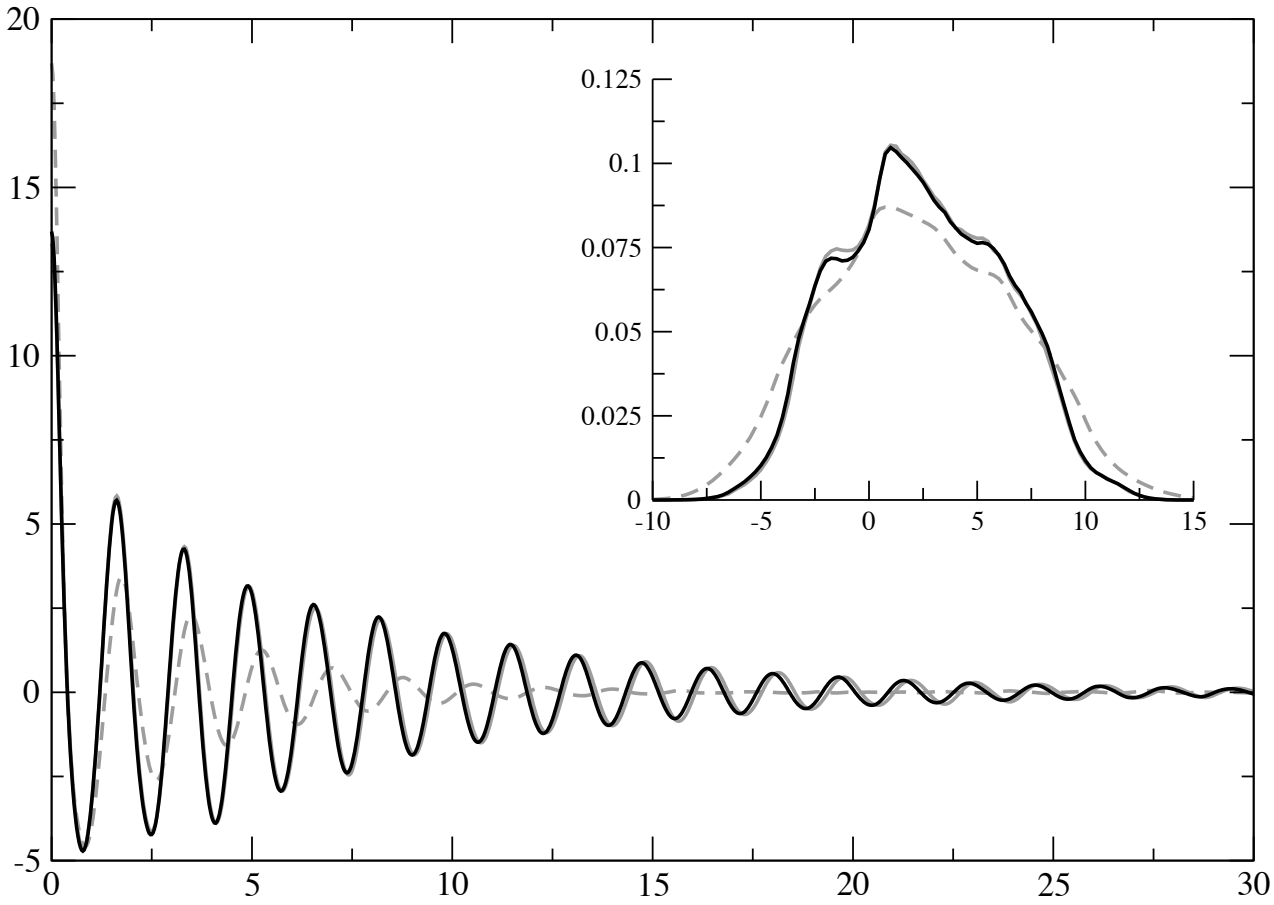
Example: the Lorenz 96 (L96) model

$$\begin{cases} \dot{X}_k = -X_{k-1}(X_{k-2} - X_{k+1}) - X_k + F_x + \dfrac{h_x}{J}\displaystyle\sum_{j=1}^{J} Y_{j,k} \\ \dot{Y}_{j,k} = \dfrac{1}{\varepsilon}\left(-Y_{j+1,k}(Y_{j+2,k} - Y_{j-1,k}) - Y_{j,k} + h_y X_k\right). \end{cases} \tag{36}$$

with $F_x = 10$, $h_x = -0.8$, $h_y = 1$, $K = 9$, $J = 8$, and $\varepsilon = 1/128$.

M_T	R	Δt	**Gain**	ν	**error(%)**	ω	**error(%)**
$\varepsilon = 2^{-7}$		2^{-11}	—	0.135	—	3.81	—
Truncated		2^{-6}	—	0.287	113	3.57	6.3
1	4	2^{-4}	32	0.201	49	3.76	1.3
1	2	2^{-5}	32	0.171	27	3.89	2.1
1	1	2^{-6}	32	0.162	20	3.88	1.9
2	1	2^{-5}	32	0.162	20	3.88	1.9
4	1	2^{-4}	32	0.158	17	3.87	1.6
1	1	2^{-7}	16	0.137	2	3.84	0.8
2	1	2^{-7}	8	0.135	0	3.83	0.5

Gain in efficiency of the multiscale scheme over a direct solver for various values of the control parameters in the multiscale algorithm. The error and the gain are calculated relative to the simulation with $\varepsilon = 2^{-7}$. The parameters ν and ω are obtained by fitting by $C_0 \cos(\omega t) e^{-\nu t}$ the ACF produced by the simulations.

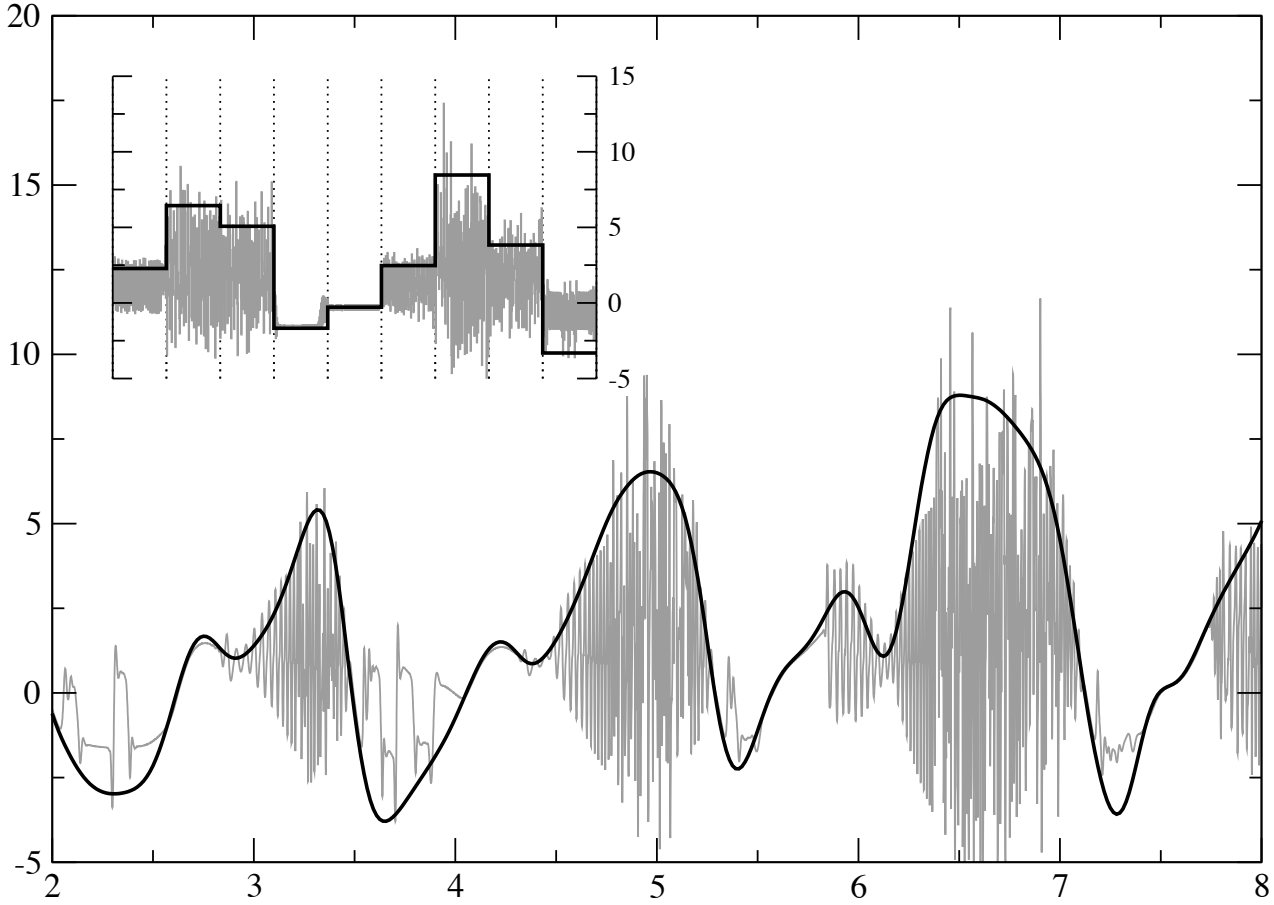


Comparison between the ACFs and PDFs; black line: multiscale solver; full grey line: direct solver with $\varepsilon = 1/128$. Efficiency gain: 16. Also shown in dashed grey are the corresponding ACF and PDF produced by the truncated dynamics where the coupling of the slow modes X_k with the fast ones, $Y_{j,k}$, is artificially switched off.

Example: L96 model with space-time scale separation

$$\begin{cases} \dot{X}_k = -X_{k-1}(X_{k-2} - X_{k+1}) - X_k + F_x + \dfrac{h_x}{J}\displaystyle\sum_{j=1}^{J} Y_{j,k} \\ \dot{Y}_{j,k} = \dfrac{1}{\varepsilon}\left(-Y_{j+1,k}(Y_{j+2,k} - Y_{j-1,k}) - Y_{j,k} + h_y X_k\right). \end{cases} \tag{37}$$

with $F_x = 10$, $h_x = -0.8$, $h_y = 1$, $K = 9$, and $J = 1/\varepsilon = 128$.



Typical time-series of the slow (black line) and fast (grey line) modes; $K = 9$, $J = 1/\varepsilon = 128$. The subplot displays a typical snapshot of the slow and fast modes at a given time.

Working assumption: spatial interaction between the $Y_{j,k}$ are sufficiently weak and short-range on the average.

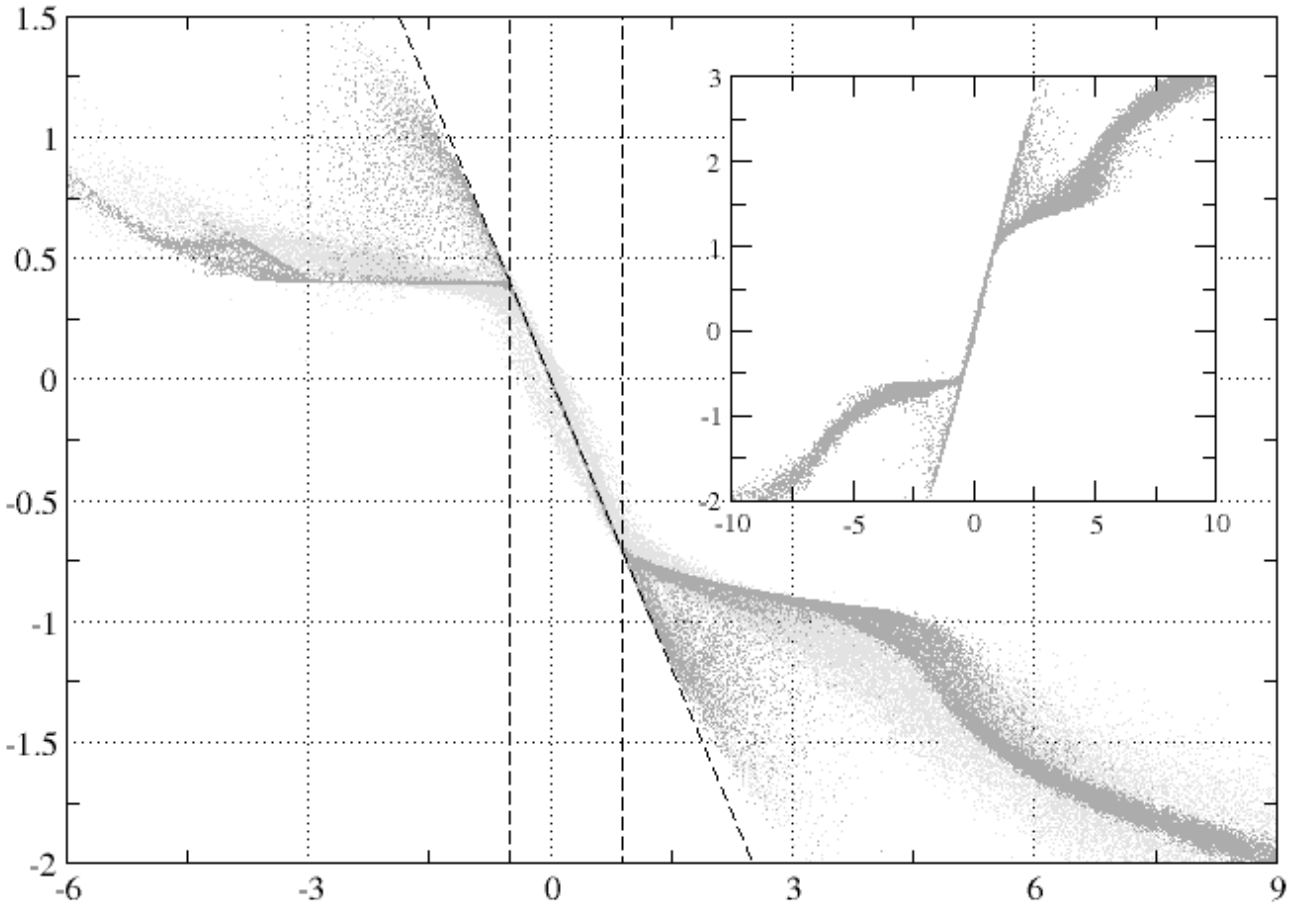
Then, at any given time, the term $(h_x/J)\sum_{j=1}^{J} Y_{j,k}$ self-averages in the limit as $J \to \infty$ (i.e. $\varepsilon \to 0$ since $J = \lfloor 1/\varepsilon \rfloor$) to a limit which, by the law of large numbers, satisfies

$$\lim_{\varepsilon \to 0} \varepsilon h_x \sum_{j=1}^{\lfloor 1/\varepsilon \rfloor} Y_{j,k} = h_x \mathbb{E}_{X_k} Y_{j,k} \equiv F(X_k), \tag{38}$$

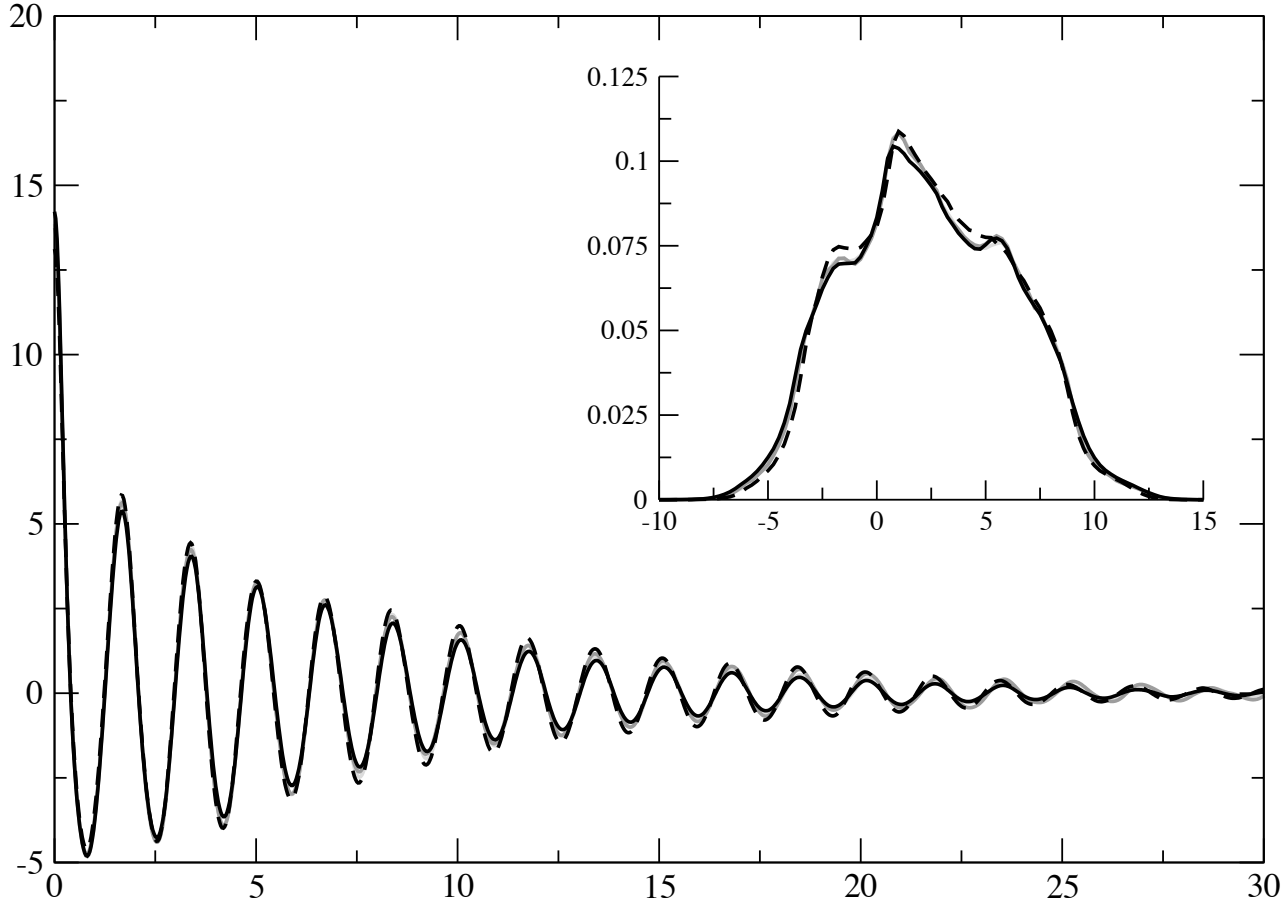
Here $\mathbb{E}_{X_k} Y_{j,k}$ is the conditional average of any given $Y_{j,k}$ at fixed X_k.

Notice that this implies $F(X_k)$ instead of $F(X_1, \ldots, X_K)$.

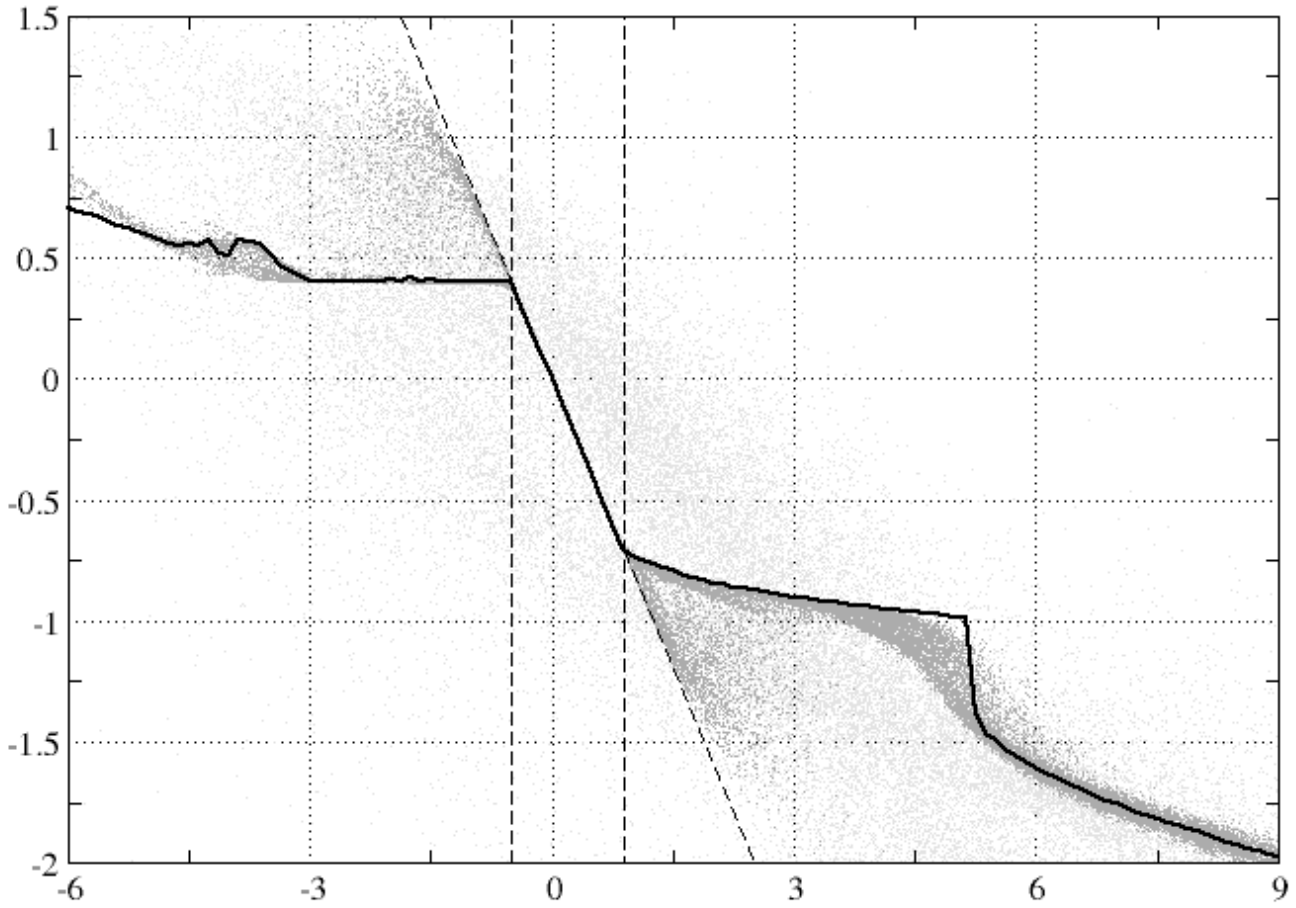
This assumption is corroborated by numerical experiments.



Scatterplots of the bare forcing $\varepsilon h_x \sum_{j=1}^{\lfloor 1/\varepsilon \rfloor} Y_{j,k}$ (no averaging here); light grey: $J = 1/\varepsilon = 128$; dark grey: $J = 1/\varepsilon = 4096$ ($K = 9$). The sharpness of these graphs confirm that bare forcing self-averages consistent with (38). The subplot: $J = 1/\varepsilon = 1024$, $h_x = 1.2$ (instead of $h_x = -0.8$ taken otherwise); it can be obtained by rescaling the forcing in the main plot respectively by 1.2/(-0.8).



Comparison of ACFs and PDFs (subplot) of the slow mode for various simulations in $J = 1/\varepsilon$ regime. Light grey: $J = 128$; dark grey: $J = 256$ (practically indistinguishable from the previous one); solid black: result from the multiscale scheme with tabulated forcing ($J = 16$; gain $= \infty$); dashed black: result from the multiscale scheme with $J = 8$, $\Delta t = 2^{-7}$, $M_T = R = 1$ (gain $= 256$).



Dark grey: scatterplots of the bare forcing $\varepsilon h_x \sum_{j=1}^{\lfloor 1/\varepsilon \rfloor} Y_{j,k}$ with $J = 1/\varepsilon = 4096$ ($K = 9$). Light grey: effective forcing $F_k(X)$ produced on-the-fly by the multiscale scheme with $K = 9$, $J = 8$, $M_T = 1$, $R = 64$, $\Delta t = 2^{-7}$. Solid black line: tabulated effective forcing computed via the multiscale scheme with $K = 9$, $J = 16$.

Warning: L96 model with hidden slow variables

$$\begin{cases} \dot{X}_k = -X_{k-1}(X_{k-2} - X_{k+1}) - \dfrac{1}{\sqrt{J}} \displaystyle\sum_{j=1}^{J} \left(Y_{j,k+1}^2 - Y_{j,k-1}^2\right) \\ \dot{Y}_{j,k} = -\dfrac{1}{\varepsilon} Y_{j+1,k}(Y_{j+2,k} - Y_{j-1,k}) - \dfrac{1}{\sqrt{J}} Y_{j,k}(X_{k+1} - X_{k-1}). \end{cases} \tag{39}$$

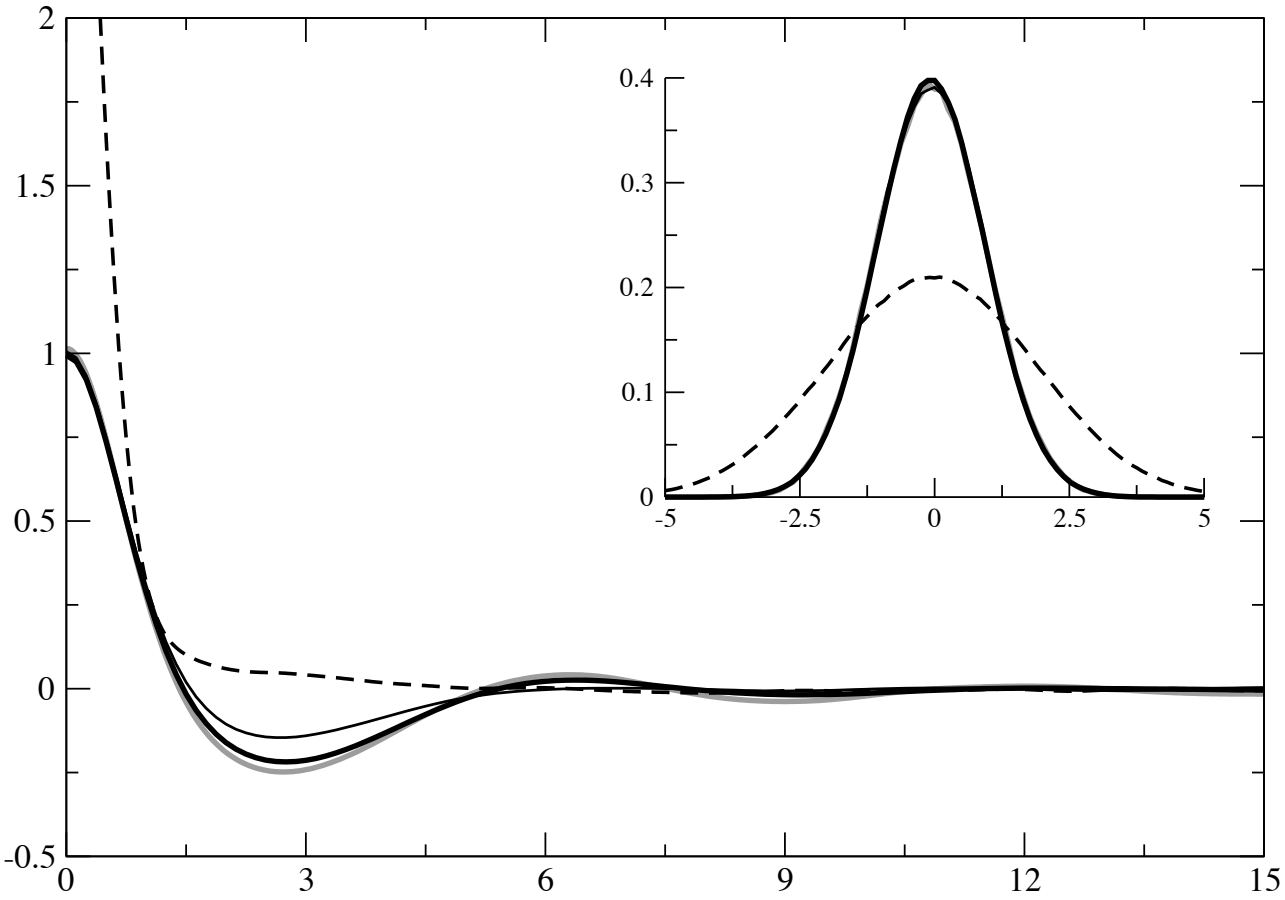
with $J = 1/\varepsilon$.

The following is an additional slow variables (hidden in (39)):

$$\bar{B}_k = \frac{1}{\sqrt{J}} \sum_{j=1}^{J} Y_{j,k}^2. \tag{40}$$

Due to the absence of forcing and damping in (39), this equation conserves the energy

$$E = \sum_{k=1}^{K} \left(X_k^2 + \sum_{j=1}^{J} Y_{j,k}^2 \right). \tag{41}$$



ACFs and PDFs of X_k. Grey line: direct simulation with $\varepsilon = 1/64$, $J = 512$ ($\delta t = 2^{-10}$); thick solid black line: multiscale scheme with $\Delta t = 2^{-8}$, $J = 32$ (efficiency gain: 64); thin black line: multiscale with $\Delta t = 2^{-7}$, $J = 8$ (efficiency gain: 512). Dashed black line: multiscale scheme with $\Delta t = 2^{-8}$, $J = 32$ (efficiency gain: 64) where the hidden slow variables B_k are not accounted for. The discrepancy clearly indicates that accounting for the B_k's is necessary. In all the multiscale computations $M_T = R = 1$.

Toward seamless multiscale schemes?

The last example shows that we must know the slow variables to make the basic multiscale algorithm work.

Can we bypass this?

Recall that in the case of stiff ODEs, when there exists a slow manifold, we can:

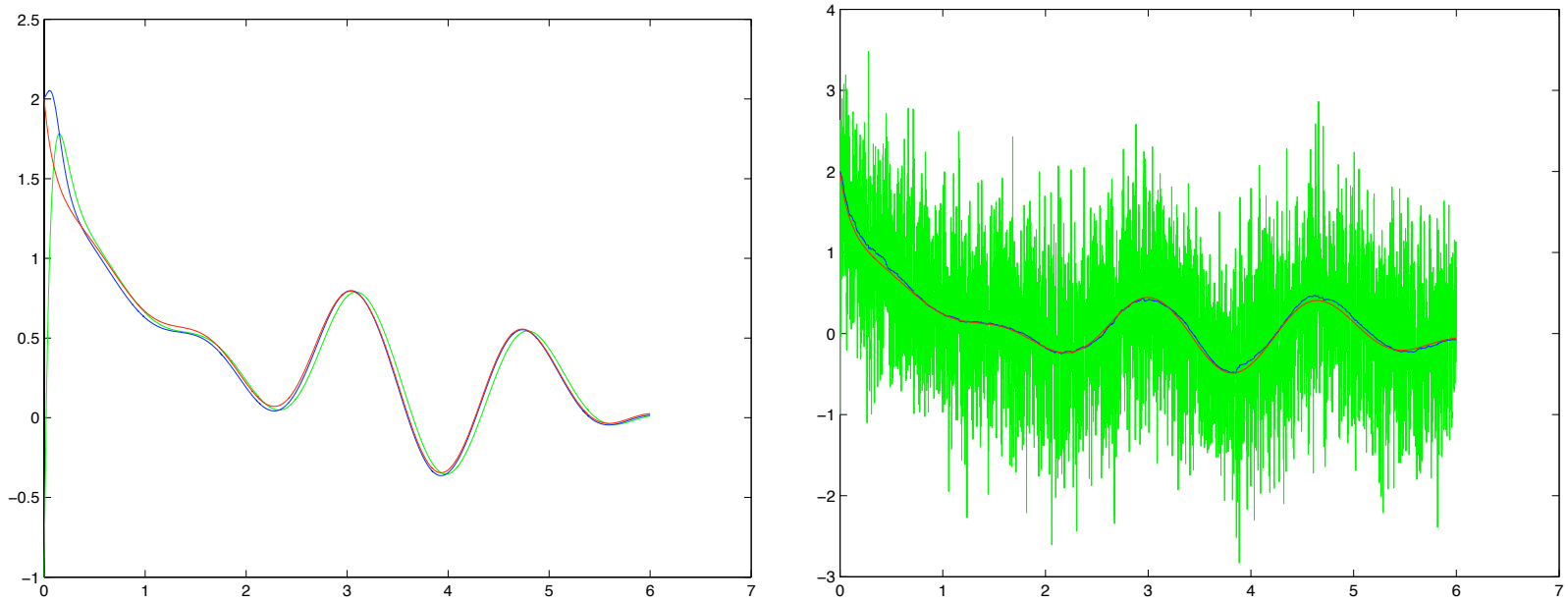
1. Take few steps with the full system using a small $O(\varepsilon)$ time-step to reach the manifold (on which the system is not stiff anymore);

2. Take then a step with the full system using a bigger $O(1)$ time-step (which may bring the system off the manifold again);

3. Repeat

This is not the most efficient strategy (going implicit is better), but it has the advantage of being very simple (one does not need what is fast what is slow, simply that some variables evolve fast on an $O(\varepsilon)$ time-scale, and some evolve slowly on an $O(1)$ time-scale), and it can be optimized (these are the so-called Chebychev methods).

Unfortunately, in most situations of interest where stochastic effects matter, there is no slow manifold, so the above strategy will not work!



A situation with a slow manifold (left panel), and one without any (right panel).

Reformulating the main theorem:

Consider the system

$$\dot{Z}_t = \frac{1}{\varepsilon} f(Z_t) + g(Z_t) \tag{42}$$

for some variable $Z_t \in \mathbb{R}^n$. Assume that there exists a vector valued function $\varphi : R^n \to R^m$ $(m < n)$ such that:

1. We have
$$f(z) \cdot \nabla \varphi(z) = 0; \tag{43}$$
2. The dynamics
$$\dot{Z}_t^x = f(Z_t^x) \tag{44}$$
is ergodic on every component indexed by $\varphi(z) = x \in \mathbb{R}^m$ with respect to the equilibrium distribution $d\mu_x(z)$.

Then $X_t = \varphi(Z_t)$ are slow variables satisfying the following equation

$$\dot{X}_t = H(X_t) \tag{45}$$

where

$$H(x) = \int_{\mathbb{R}^n} g(z) \cdot \nabla \varphi(z) d\mu_x(z). \tag{46}$$

The previous result is more general, but unfortunately it does not lead to a seamless multiscale algorithm because the mapping φ defining the slow variable is usually nonlinear.

In particular, it is easy to see that averaging the original equation, i.e. using

$$\dot{\bar{Z}}_t = G(\varphi(\bar{Z}_t)) \tag{47}$$

where

$$G(z) = \int_{\mathbb{R}^n} g(z) d\mu_x(z), \tag{48}$$

will, in general not lead to (45), in the sense that

$$\varphi(\bar{Z}_t) \neq X_t$$

unless $\nabla\varphi(z)$ is a function of x alone, i.e.

$$\nabla\varphi(z) = J(\varphi(z)) \tag{49}$$

for some $J : \mathbb{R}^m \to \mathbb{R}^n \times \mathbb{R}^m$.

Only if (49) is satisfied do we have

$$F(z) = G(z)J(z) \quad \text{and} \quad \varphi(\bar{Z}_t) = X_t$$

When this is the case, we can build a seamless multiscale algorithm based only on our ability to decompose the velocity field in its fast $O(\varepsilon^{-1})$ component and its slow $O(1)$ component.

Seamless multiscale algorithm

If

$$\nabla\varphi(z) = J(\varphi(z)) \tag{50}$$

for some $J : \mathbb{R}^m \to \mathbb{R}^n \times \mathbb{R}^m$, then we can do the following to integrate

$$\dot{Z}_t = \frac{1}{\varepsilon} f(Z_t) + g(Z_t) \tag{51}$$

Use

$$\tilde{Z}_{m+1,n} = \tilde{Z}_{m,n} + \frac{\delta t}{\varepsilon}\, f(\tilde{Z}_{m,n}), \quad \tilde{Z}_{0,0} = Z_{t=0}, \quad \tilde{Z}_{0,n+1} = \tilde{Z}_{M+M_T,n} \tag{52}$$

and compute

$$\tilde{G}_n = \frac{1}{M} \sum_{m=M_T}^{M+M_T-1} g(\tilde{Z}_{n,m}) \tag{53}$$

Then use

$$Z_{n+1} = Z_n + \Delta t\ \tilde{G}_n, \qquad Z_0 = Z_{t=0} \tag{54}$$

and repeat.

Here $\varphi(Z_n) \approx X(n\Delta t)$.

Notice that the algorithm is totally seamless, i.e. one does not need to know φ nor $\nabla\varphi$.

Application to Kinetic Monte-Carlo (KMC)

Suppose we are given a KMC system on $x \in S$ with reaction channels

$$R_j = (a_j(x), \nu_j), \qquad R = \{R_1, \ldots, R_{M_R}\}. \tag{55}$$

Given state x, the occurrences of the reactions on an infinitesimal time interval dt are independent of each other and the probability for reaction R_j to happen during this time interval is given by $a_j(x)dt$. The state of the system after reaction R_j is $x + \nu_j$.

Gillespie's Stochastic Simulation Algorithm (SSA)

Let

$$a(x) = \sum_{j=1}^{M_R} a_j(x). \tag{56}$$

Assume that the current time is t_n, and the system is at state X_n. We perform the following steps:

1. Generate independent random numbers r_1 and r_2 with uniform distribution on the unit interval $(0, 1]$. Let

   $$\delta t_{n+1} = -\frac{\ln r_1}{a(X_n)}, \tag{57}$$

   and k_{n+1} be the natural number such that

   $$\frac{1}{a(X_n)} \sum_{j=0}^{k_{n+1}-1} a_j(X_n) < r_2 \leq \frac{1}{a(X_n)} \sum_{j=0}^{k_{n+1}} a_j(X_n), \tag{58}$$

   where $a_0 = 0$ by convention.

2. Update the time and the state of the system by

   $$t_{n+1} = t_n + \delta t_{n+1} \ , \qquad X_{n+1} = X_n + \nu_{k_{n+1}}. \tag{59}$$

Then repeat.

Suppose that are fast and slow reactions:

$$R_j^s = (a_j^s(x), \nu_j^s), \qquad R_j^f = (\varepsilon^{-1} a_j^f(x), \nu_j^f). \tag{60}$$

Then $v(x) = b \cdot x$ is a slow variable if

$$b \cdot \nu_j^f = 0 \tag{61}$$

for all ν_j^f. The set of such vectors form a linear subspace in R^{N_S}.

Let $b_1, b_2, \ldots, b_J$ be a set of basis vectors of this subspace, and define

$$y_j = b_j \cdot x \quad \text{for} \quad j = 1, \ldots, J, \tag{62}$$

then $y_1, y_2, \cdots, y_J$ defines a complete set of slow variables, i.e. all slow observables can be expressed as functions of $y_1, y_2, \cdots, y_J$.

In other words, in the present case, the slow variables are linear functions of the original variables!

Nested Stochastic Simulation Algorithm (SSA)

1. **Inner SSA:** Run N independent replicas of SSA with the fast reactions $R^f = \{(\epsilon^{-1}a^f, \nu^f\})\}$ only, for a time interval of $T_0 + T_f$.

   During this calculation, compute the modified slow rates for $j = 1, \cdots, M_s$

   $$\tilde{a}_j^s = \frac{1}{N}\sum_{k=1}^{N}\frac{1}{T_f}\int_{T_0}^{T_f+T_0} a_j^s(X_\tau^k)d\tau, \tag{63}$$

   where X_τ^k is the result of the k-th replica of this auxiliary virtual fast process at virtual time τ whose initial value is $X_{t=0}^k = X_n$, and T_0 is a parameter we choose in order to minimize the effect of the transients to the equilibrium in the virtual fast process.

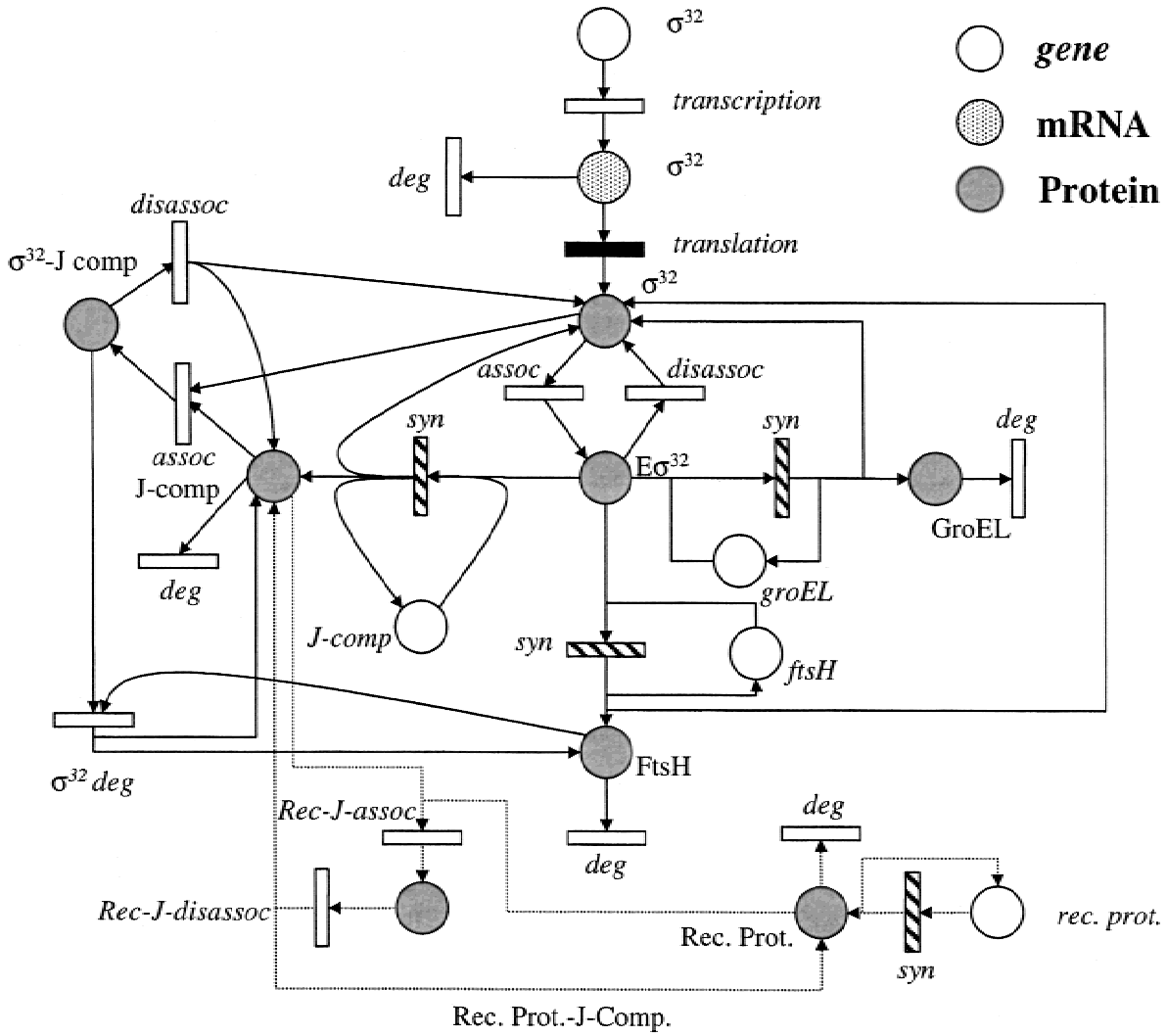
2. **Outer SSA:** Run one step of SSA for the modified slow reactions $\tilde{R}^s = (\tilde{a}^s, \nu^s)$ to generate (t_{n+1}, X_{n+1}) from (t_n, X_n).

Then repeat.

Example: heat shock response of Escherichia Coli

Reaction	Rates magnitude
DNA.σ^{32} $\rightarrow$ mRNA.σ^{32}	1.4×10^{-3}
mRNA.σ^{32} $\rightarrow$ σ^{32} + mRNA.σ^{32}	1.19
mRNA.σ_{32} $\rightarrow$ degradation	2.38×10^{-5}
σ_{32} $\rightarrow$ RNAPσ^{32}	10.5
RNAPσ^{32} $\rightarrow$ σ^{32}	9.88
σ^{32} + DnaJ $\rightarrow$ σ^{32}.DnaJ ($\star\star$)	25.2
DnaJ $\rightarrow$ degradation ($\star\star$)	2.97×10^{-6}
σ^{32}.DnaJ$\rightarrow$ σ^{32} + DnaJ	1.30
DNA.DnaJ + RNAPσ^{32} $\rightarrow$ DnaJ + DNA.DnaJ + σ^{32}	3.71
DNA.FtsH + RNAP.σ^{32} $\rightarrow$ FtsH + DNA.FtsH + σ^{32}	0
FtsH $\rightarrow$ degradation	1.48×10^{-8}
GroEL $\rightarrow$ degradation	7.76×10^{-5}
σ^{32}.DnaJ + FtsH $\rightarrow$ DnaJ + FtsH	8.4
DNA.GroEL + RNAPσ^{32} $\rightarrow$ GroEL + DNA.GroEL +σ^{32}	4.78
Protein $\rightarrow$ UnfoldedProtein ($\star$)	10^6
DnaJ+ UnfoldedProtein $\rightarrow$ DnaJ.UnfoldedProtein ($\star$)	10^7
DnaJ.UnfoldedProtein $\rightarrow$ DnaJ+ UnfoldedProtein ($\star$)	10^6

Reaction list for the heat shock model of E. Coli proposed in: R. Srivastava, M. S. Peterson and W. E. Bently, *Biotech. Bioeng.* **75**, 120–129 (2001). The rate magnitude is the value of $a_i(x)$ evaluated at initial time or equilibrium. The last three reactions marked with a ($\star$) in the table are fast: they are used in the Inner SSA. All the other reactions are used in the Outer SSA, and the rates of the reactions marked with a ($\star\star$) are averaged.



Stochastic petri net diagram for the model of heat shock of E. Coli [from R. Srivastava, M. S. Peterson and W. E. Bently, *Biotech. Bioeng.* **75**, 120–129 (2001)]

$(N, T_f/10^{-6})$	$(1,1)$	$(1,4)$	$(1,16)$	$(1,64)$	$(1,256)$	$(1,1024)$
CPU	0.62	1.32	2.98	9.56	35.81	142.08
$\overline{\sigma^{32}}$	4.60	8.66	13.60	14.52	14.98	15.00
var(σ^{32})	4.41	8.11	12.22	13.13	13.73	14.66

Efficiency of nested SSA when $N = 1$. Since we used $N_0 = 1000$ realizations of the Outer SSA to compute $\overline{\sigma^{32}}$ and var(σ^{32}), the statistical errors on these quantities is about 0.2. For comparison, the actual values of these quantities are

$$\overline{\sigma^{32}} = 14.8 \pm 0.2, \qquad \text{var}(\sigma^{32}) = 14.2 \pm 0.2.$$

and the direct SSA took 19719.2 seconds of CPU time to compute them.

Multiscale numerical techniques for stiff ODEs:

E. Hairer and G. Wanner, *"Solving Ordinary Differential Equations II, Stiff and Differential-Algebraic Problems"*, Springer-Verlag, 1991

C.W. Gear and I.G. Kevrekidis, "Projective methods for stiff differential equations: problems with gaps in their eigenvalue spectrum," *SIAM J. Sci. Comp.* **24**(4):109-110 (2003).

V. I. Lebedev and S. I. Finogenov, "Explicit methods of second order for the solution of stiff systems of ordinary differential equations", *Zh. Vychisl. Mat. Mat Fiziki*, **16**:895–910 (1976).

A. Abdulle, "Fourth order Chebychev methods with recurrence relations", *SIAM J. Sci. Comput.* **23**:2041–2054 (2002).

Multiscale numerical techniques for stochastic systems

E. Vanden-Eijnden, "Numerical techniques for multiscale dynamical systems with stochastic effects", *Comm. Math. Sci.*, **1**: 385–391 (2003).

W. E, D. Liu and E. Vanden-Eijnden, "Analysis of Multiscale Methods for Stochastic Differential Equations," Comm. Pure App. Math. **58**: 1544–1585 (2005).

Application to L96:

I. Fatkullin and E. Vanden-Eijnden, A computational strategy for multiscale systems with applications to Lorenz 96 model, *J. Comp. Phys.* **200**:605–638 (2004).

Seamless scheme for Kinetic Monte-Carlo

W. E, D. Liu, E. Vanden-Eijnden, "Nested stochastic simulation algorithm for chemical kinetic systems with disparate rates," J. Chem. Phys. **123**: 194107 (2005)

W. E, D. Liu, E. Vanden-Eijnden, "Nested stochastic simulation algorithm for chemical kinetic systems with multiple time-scales," submitted: J. Comp. Phys

Related works:

Y. Cao, D. Gillespie, L. Petzold, "Multiscale Stochastic Simulation Algorithm with Stochastic Partial Equilibrium Assumption for Chemically Reacting Systems," *J. Comp. Phys.* **206**(2):395–411 (2005).

A. Samant and D. G. Vlachos, "Overcoming stiffness in stochastic simulation stemming from partial equilibrium: A multiscale Monte Carlo algorithm.," *J. Chem. Phys.* **123**, 144114 (2005)