ANOTHER HIGHER ORDER LANGEVIN ALGORITHM FOR QCD

Andreas S. KRONFELD

Deutsches Elektronen-Synchrotron DESY, Notkestrasse 85, D-2000 Hamburg 52, Fed. Rep. Germany

Received 31 December 1985

An algorithm is provided for integrating the Langevin equation which is second order. It introduces a term into the drift force which is a product of the gaussian noise and a second derivative of the action. The specific application presented here is for nonabelian gauge theories interacting with fermions, e.g. QCD, for which it requires less memory than the Runge–Kutta algorithm of the same order. The memory and computational requirements of Euler, Runge–Kutta, and the present algorithm are compared.

Since the introduction of stochastic quantization [1] there has been much interest in using the Langevin equation as a basis for numerical simulations of quantum field theories. Especially interesting are applications to QCD [2,3]. In stochastic quantization field variables evolve continuously in λ, a parameter variously called Langevin time or fifth time, via a Langevin equation. In Langevin simulations one discretizes λ; for example, one uses simple Euler or Runge–Kutta schemes familiar from deterministic differential equations. The alternative presented here attains $O(\Delta\lambda^2)$ accuracy by introducing terms in the drift force with higher derivatives of the action. It is, however, peculiar to stochastic differential equations because the new terms are proportional to noise. Avoiding the task of evaluating higher derivatives was the original motivation for developing Runge–Kutta algorithms for stochastic differential equations [4]. However, when the action contains only local interactions between the degrees of freedom, as in the case of QCD, the higher derivatives are not problematic. Indeed, when fermions are present, as in QCD, the new algorithm may be even more efficient.

The simple Euler update rule [2] (hereafter referred to as algorithm A)

$$U_{x,\mu}^{(\lambda+1)} = \exp(-f \cdot T)\, U_{x,\mu}^{(\lambda)} , \tag{1a}$$

$$f_i = \sqrt{\epsilon_{ij}}\,\eta_j + \epsilon_{ij}[\partial_j S_g - \tfrac{1}{2}\mathcal{R}(\xi^\dagger \partial_j \mathcal{M}\mathcal{M}^{-1}\xi)] , \tag{1b}$$

generates a sequence of configurations $\{U\}^{(\lambda)}$ labelled by Langevin time λ. The configurations are distributed according to $P = \exp(-\bar{S})$, where the equilibrium action

$$\bar{S} = S_g - \mathrm{Tr}\ln \mathcal{M} + O(\epsilon) . \tag{2}$$

In eq. (1b) η and ξ are gaussian noise: $\langle\eta\rangle = 0 = \langle\xi\rangle$ and $\langle\eta\eta\rangle = 2 = \langle\xi^\dagger\xi\rangle$; i, j, etc., are multi-indices: $i = (a, x, \mu)$; $\mathcal{R}$ denotes the real part. Also, $\mathcal{M}$ is the fermion hopping matrix γ_5("D $\cdot\ \gamma - m$"), and S_g is the pure gauge action. The algebra generators T_a and the "field derivatives" [5] ∂_t obey

$$\mathrm{tr}(T_a T_b) = -\tfrac{1}{2}\delta_{ab}, \quad [T_a, T_b] = -f_{abc}T_c ,$$

$$[\partial_{(a,x,\mu)}, \partial_{(b,y,\nu)}] = -f_{abc}\partial_{(c,x,\mu)}\delta_{\mu\nu}\delta_{xy} , \tag{3}$$

where the f_{abc} are the structure constants of the group to which the U belong.

Langevin updating, as in eqs. (1), has several advantages. One is the efficient way in which the bilinear noise term in eq. (1b) includes fermions. Another is the possibility of choosing ϵ_{ij} nonlocal in space, which can mitigate critical slow down [2,6]. The main disadvantage is the presence of the $O(\epsilon)$ terms in the equilibrium action. This is not a serious disadvantage when these terms do not effect the continuum limit ($\beta \to \infty, a \to 0$). However, when ϵ_{ij} is nonlocal, the arguments that indicate that the $O(\epsilon)$ corrections do not affect the continuum limit are based on weak coupling perturbation theory; in strong coupling ($\beta \to 0$) they no longer apply. Since present simulations are far from the continuum limit, it is worthwhile to formulate alternatives to eq. (1) whose equilibrium action differs from the simulation action, $S_g - \mathrm{Tr}\ln \mathcal{M}$ for QCD, by terms of $O(\epsilon^2)$.

Although eq. (1b) includes fermions most elegantly, the Wick contraction properties of ξ make difficult the implementation of Runge–Kutta algorithms as in ref. [4]. Batrouni [7] has solved this problem by introducing a second fermionic noise, ζ. His Runge–Kutta process replaces the drift force of eq. (1b) by

$$f_i = \sqrt{\epsilon_{ij}}\eta_j + \tfrac{1}{2}(1 + \tfrac{1}{6}\bar{\epsilon}C_A)\epsilon_{ij}$$
$$\times (\partial_j S_g + \partial_j \tilde{S}_g - \tfrac{1}{4}\xi^\dagger \mathcal{A}_j \xi - \tfrac{1}{4}\zeta^\dagger \tilde{\mathcal{A}}_j \zeta)$$
$$- \tfrac{1}{128}\eta_k \sqrt{\epsilon_{kl}}\epsilon_{ij}\,\mathcal{R}(\xi^\dagger \mathcal{A}_l \zeta\zeta^\dagger A_j \xi)\ , \qquad (4)$$

where [‡1] $\mathcal{A}_j \equiv \mathcal{M}^{-1}\partial_j \mathcal{M}^2 \mathcal{M}^{-1}$, and $\bar{\epsilon}$ is the diagonal component of $\epsilon_{ij} = \delta_{ab}\epsilon_{\mu\nu}(x-y)$. The Casimir operator of the adjoint representation C_A appears because of the commutation relations of the ∂_i. A tilde on S_g and $\mathcal{A}_j$ implies evaluation using the "tentative update" $\tilde{U}$, which is obtained from $U^{(\lambda)}$ by eqs. (1). This will be called algorithm B.

The $O(\epsilon^{3/2})$ term in eq. (4) is required to remove a nonintegrable term from the Fokker–Planck equation, which is used to determine the probability distribution. Nonintegrable terms are annoying because they prevent one from defining the equilibrium action. For finite time steps the equilibrium Fokker–Planck equation is

$$0 = \sum_{n=1}^{\infty} \frac{1}{n!}\partial_{i_1}\ldots\partial_{i_n}[\langle f_{i_1}\ldots f_{i_n}\rangle P]\ . \qquad (5)$$

Expand the $\tilde{U}$ dependent terms in eq. (4) to $O(\epsilon^2)$ to obtain an effective drift force. Ignoring the $O(\epsilon^{3/2})$ term in eq. (4) one finds in $\langle f_i f_j\rangle$ the following term:

$$\tfrac{1}{8}\epsilon_{ik}\epsilon_{jl}\,\mathrm{Tr}(\mathcal{A}_k\mathcal{A}_l)\ , \qquad (6a)$$

which when inserted into eq. (5) yields a contribution that is not integrable. However, the last term in eq. (4) contributes to $\langle f_i f_j\rangle$:

$$-\tfrac{16}{128}\epsilon_{ik}\epsilon_{jl}\,\mathrm{Tr}(\mathcal{A}_k\mathcal{A}_l)\ , \qquad (6b)$$

so it cancels the nonintegrable contribution of eq. (6a).

Batrouni's trick teaches us a lesson. If drift forces proportional to the gauge field noise can cancel nonintegrable terms in the Fokker–Planck equation, perhaps they can cancel integrable ones as well. Study of the Fokker–Planck equation for Runge–Kutta processes then impels one to consider an update rule [‡2]

$$U^{(\lambda+1)}_{x,\mu} = \exp(-f\cdot T)U^{(\lambda)}_{x,\mu}\ , \qquad (7a)$$

$$f_i = \sqrt{\epsilon_{ij}}\eta_j + (1+\alpha_1\bar{\epsilon}C_A)\epsilon_{ij}$$
$$\times [\partial_j S_g - \tfrac{1}{2}\mathcal{R}(\xi^\dagger \partial_j \mathcal{M}\mathcal{M}^{-1}\xi)]$$
$$- \alpha_2\eta_k\sqrt{\epsilon_{kl}}\epsilon_{ij}\partial_l[\partial_j S_g - \tfrac{1}{2}\mathcal{R}(\xi^\dagger\partial_j\mathcal{M}\mathcal{M}^{-1}\xi)]$$
$$- \alpha_3\eta_k\sqrt{\epsilon_{kl}}\epsilon_{ij}$$
$$\times \mathcal{R}\,\xi^\dagger(\partial_j\mathcal{M}\mathcal{M}^{-1}\partial_l\mathcal{M}\mathcal{M}^{-1}$$
$$+ \mathcal{M}^{-1}\partial_j\mathcal{M}\,\partial_l\mathcal{M}\mathcal{M}^{-1})\xi\ . \qquad (7b)$$

The α_2 term in this drift force is (except for its coefficient) identical to one of three terms that appear in the effective drift force for the Runge–Kutta algorithm. (The other two terms turn out to be inessential.) The α_3 term is essentially the same as the last term of eq. (4). With the assignments for the α_i in eq. (10) (below), eqs. (7) define algorithm C, which is the main result of this letter.

To determine the coefficients α_i in eq. (7b), one must use the Fokker–Planck equation. Define $S \equiv S_g - \mathrm{Tr}\ln\mathcal{M}$. To $O(\epsilon^2)$ the expressions for $\langle f_i f_j f_k\rangle$ and $\langle f_i f_j f_k f_l\rangle$ in algorithm C are the same as for algorithm A. However $\langle f_i\rangle_C = (1+\alpha_1 C_A)\langle f_i\rangle_A$, and

$$\langle f_i f_j\rangle_C = \langle f_i f_j\rangle_A$$
$$- 4\epsilon_{il}\epsilon_{jk}[\alpha_2\partial_k\partial_l S + 2\alpha_3\,\mathrm{Tr}(\mathcal{A}_l\mathcal{A}_k)]\ . \qquad (8)$$

When the expressions are inserted into eq. (5) one find, using $\partial_i P = -(\partial_i S)P + O(\epsilon)$ to simplify the $O(\epsilon)$ terms,

[‡1] N.B. $\tfrac{1}{4}\xi^\dagger\mathcal{A}_j\xi = \tfrac{1}{2}\mathcal{R}\xi^\dagger\partial_j\mathcal{M}\mathcal{M}^{-1}\xi$.

[‡2] Ref. [5] considers algorithms similar to eqs. (7) for systems without fermions.

$$
\begin{aligned}
0 = {} & \partial_i P + (1 + \alpha_1 \bar{\epsilon} C_{\mathrm{A}})(\partial_i S)P \\
& + \tfrac{1}{4}\partial_i[\tfrac{1}{3}\bar{\epsilon} C_{\mathrm{A}} S + \epsilon_{jk}(2\partial_j\partial_k S - \partial_j S\,\partial_k S)]P \\
& + \tfrac{1}{4}\partial_j \epsilon_{jk}\,[\mathrm{Tr}(\mathcal{A}_i \mathcal{A}_k)P] \\
& - \alpha_2 \partial_i[\tfrac{1}{2}\bar{\epsilon} C_{\mathrm{A}} S + \epsilon_{jk}(2\partial_j\partial_k S - \partial_j S\,\partial_k S)]P \\
& - 4\alpha_3 \partial_j \epsilon_{jk}\,[\mathrm{Tr}(\mathcal{A}_i \mathcal{A}_k)P] + \mathrm{O}(\epsilon^2) \, . \qquad (9)
\end{aligned}
$$

To arrive at eq. (9) I have also performed one integral, set the associated integration constant to zero and, multiplied by ϵ^{-1}. As expected, the $\mathrm{O}(\epsilon)$ terms can all be cancelled if

$$\alpha_2 = \tfrac{1}{4}, \quad \alpha_3 = \tfrac{1}{16}, \quad \alpha_1 = \tfrac{1}{24} \, . \qquad (10)$$

With these choices it is easy to integrate the Fokker–Planck equation and see that the equilibrium action is $\bar{S} = S + \mathrm{O}(\epsilon^2)$. In principle one could now evaluate the new leading corrections; however, this calculation is rather tedious, and the corrections are apparently neither simple nor instructive.

Let us examine the extra work involved in computing the drift force of eq. (7b). In general the pure gauge action S_{g} will be a sum of terms, and each term will be proportional to a Wilson loop W. The second derivative of the Wilson loop, $\partial_l \partial_j W$, is nonvanishing only when l and j describe links in the same loop. To discuss the fermion terms let $\psi \equiv \mathcal{M}^{-1}\xi, \mathcal{N} = \eta_k \sqrt{\epsilon_{kl}}\partial_l \mathcal{M}$, and $\chi = \mathcal{M}^{-1}\mathcal{N}\psi$. Then the new fermion terms are $\xi^\dagger(\partial_j\partial_l\mathcal{M})\psi$, $\psi^\dagger(\partial_j\mathcal{M}\,\partial_l\mathcal{M})\psi$, and $\xi^\dagger\partial_j\mathcal{M}\chi$. The last quantity appears in both the α_2 and α_3 terms. Because $\mathcal{M}$ is local, $\mathcal{N}\psi$ is reasonably simple and $(\partial_j\partial_l\mathcal{M})\psi$ is very simple. The most serious price is that algorithm C requires two matrix inversions ψ and χ, whereas algorithm A requires only ψ. However, since $\bar{\epsilon}$ can be made substantially larger, this is a quite tolerable.

Table 1 contains a summary of the memory and fermionic matrix inversion requirements of the three algorithms. Matrix inversions warrant emphasis because they are the most time consuming part of the computation. Algorithm A needs one fermionic matrix inversion per sweep, ψ, and one must store [‡3] $U^{(\lambda)}, F \equiv f \cdot T$, ξ, and ψ before one can construct $U^{(\lambda+1)}$. Both second-order algorithms require more storage and computation per update than algorithm A, but they allow larger step sizes $\bar{\epsilon}$. Since the number of updates needed to decorrelate a configuration is inversely proportional to $\bar{\epsilon}$, they will therefore reduce the total amount of computer time needed to perform a simulation. Algorithm B needs three fermionic matrix inversions: ψ, $\phi \equiv \mathcal{M}^{-1}\zeta$, and $\tilde{\mathcal{M}}^{-1}\zeta$; one must also store $\tilde{U}$, η, ζ, and ϕ. Algorithm C needs two matrix inversions, ψ and χ; one must also store χ and η. The latter is needed alone and in the computation of the $\eta_j\sqrt{\epsilon_{jl}}\partial_l$ terms. Thus, the elimination of the tentative update reduces by one the number of matrix inversions, gauge fields, and fermion fields. The price for algorithm C compared to algorithm B is the need for computing second derivatives of the action, but as discussed above, this is not too difficult when the couplings are local.

Table 1
Summary of memory and matrix inversion needs of algorithms A, B and C.

Algorithm	Gauge fields	Fermion fields	Matrix inversions
A	$2: U^{(\lambda)}, F$	$2: \xi, \psi$	$1: \psi = \mathcal{M}^{-1}\xi$
B	$4: U^{(\lambda)}, F, \tilde{U}, \eta$	$4: \xi, \psi, \zeta, \phi$	$3: \psi, \mathcal{M}^{-1}\zeta, \phi = \mathcal{M}^{-1}\zeta$
C	$3: U^{(\lambda)}, F, \eta$	$3: \xi, \psi, \chi$	$2: \psi, \chi = \mathcal{M}^{-1}\mathcal{N}\psi$

A potentially crucial point in determining the overall efficiency of the algorithms is Fourier acceleration. The step size ϵ_{ij} has been written as a matrix with this possibility in mind [‡4]. To Fourier accelerate one inserts fast Fourier transforms into the updating, which is like changing the basis for the (ij) matrix multiplication. Then $\epsilon_{ij} \to \epsilon(p)$. As argued in ref. [2], $\epsilon(p)$ can be adjusted to reduce the correlations in λ of the long-wavelength components of the gauge field configurations [‡5]. On the one hand, it is conceivable that algorithm A will be the easiest to Fourier accelerate, because its drift force is the simplest; algorithms B and C will require more calls to a FFT routine because ϵ_{ij} enters the drift force in a more baroque way.

[‡3] Of course, workspace (e.g. for the conjugate gradient) introduces additional memory requirements.

[‡4] And not because the author is fond of indices.

[‡5] In gauge theories, like QCD, this can only work if the gauge is fixed. This entails some subtleties and will be discussed elsewhere.

On the other hand, if the $O(\epsilon)$ terms in the equilibrium action of algorithm A turn out to be serious, one will be forced to use values of ϵ that will defeat the acceleration. Then higher-order algorithms will be necessary. Furthermore, it is clear that without Fourier acceleration, or with Fourier acceleration in matrix inversion only, the higher-order algorithms will provide more efficient simulations.

All three algorithms are currently being tested, both with and without Fourier acceleration. Numerical comparisons will be published when available. I anticipate that for systems without fermions Runge–Kutta algorithms are probably preferable, simply because the tentative update is easier to compute than the second derivative of the action. However, in QCD algorithm C is probably better, because it saves one matrix inversion per sweep.

References

[1] G. Parisi and Y.-S. Wu, Sci. Sin. 24 (1981) 483.
[2] G.G. Batrouni, G.R. Katz, A.S. Kronfeld, G.P. Lepage, B. Svetitsky and K.G. Wilson, Phys. Rev. D32 (1985) 2736.
[3] A. Ukawa and M. Fukugita, Phys. Rev. Lett. 55 (1985) 1854.
[4] E. Helfand, Bell System Techn. J. 58 (1979) 2289; H.S. Greensite and E. Helfand, Bell System Techn. J. 60 (1981) 1927.
[5] I.T. Drummond, S. Duane and R.R. Horgan, Nucl. Phys. B220 [FS8] (1983) 119.
[6] G. Parisi, in: Progress in gauge field theory, eds. G. 't Hooft et al. (Plenum, New York, 1984); A.S. Kronfeld, talk Wuppertal meeting on Lattice gauge theory – a challenge to large scale computing; and DESY report, in preparation.
[7] G.G. Batrouni, Cornell University report CLNS-85/665.