Preprints of the 10th IFAC International Symposium on Dynamics and Control of Process Systems
The International Federation of Automatic Control
December 18-20, 2013. Mumbai, India

Projective Integration with an Adaptive Projection Horizon

Max A. Fahrenkopf, James W. Schneider, B. Erik Ydstie

Department of Chemical Engineering
Carnegie Mellon University, Pittsburgh, PA 15213 USA
mfahrenk@andrew.cmu.edu, schneider@cmu.edu, ydstie@cmu.edu

Abstract: We present an algorithm for projective integration that is computationally efficient for integrating systems of differential equations with multiple time-scales. Adaptive projective integration is a technique that uses a few inner integration steps to generate data to fit to a local reduced-order model. This reduced-order model is then used to extrapolate forward in time to estimate the states at some future time. This inner-outer integration is iterated until the desired integration is complete. The method uses an adaptive projective horizon to control for error generation during the integration. By examining an example Brusselator system, consisting of three non-linear differential equations, we show two orders of magnitude savings in computational time using adaptive projective integration over explicit Euler's method.

Keywords: Multi-time-scales, Large Scale Systems, Stiff differential equations, Integration, Computer Simulation, Reduced-order modeling

1. INTRODUCTION

Large systems of non-linear differential equations are typically very computationally expensive to solve. This expense often precludes further analysis such as optimization or fixed point identification. Stiff differential equations are particularly expensive to solve as they demand stringent conditions to be met for numerical stability. Stiffness arises when the differential equations describe both fast and slow dynamics and frequently occur in chemical process systems due to chemical reactor kinetics and recycle streams (Vora and Daoutidis, 2001; Baldea and Daoutidis, 2007). Proper exploitation of multiple time-scales in process systems has led to efficient control strategies, such as adaptive control, that stems from the use of reduced-order models (Saksena et al., 1984).

Projective integration is a computationally efficient method for solving differential equations with both fast and slow dynamics. Projective integration works by utilizing two coupled integration methods with very different integration time steps. An inner integration is performed at small time steps to damp out fast dynamics. After a few inner integration steps an extrapolation is made over a large number of time steps which serves as the outer integration over the slow dynamics. This process is then repeated until the desired integration is completed (Kevrekidis and Samaey, 2009). Projective integration is a particularly efficient method to integrate stiff differential equations because it avoids costly implicit integration methods (Gear and Kevrekidis, 2003). Efficient integration of stiff differential equations represents just one example of the accelerating power of this method. Additional examples include accelerating stochastic simulation of nematic liquid crystals (Siettos et al., 2003), accelerating kinetic Monte Carlo simulations of adsorption onto a metal substrate (Rico-Martínez et al., 2004), and projective integration over space and time for accelerating the integration of partial differential equation (Kevrekidis et al., 2003).

In this paper we present a projective integration scheme that uses an adaptive projection horizon to control the error generated. We employ a affine model [1] to make optimal predictions for the outer integration. By applying this additional structure, important properties such as stability and prediction error can be estimated and the projection horizon can be adjusted to balance the trade-off between acceleration and accuracy. Related work in adaptive control has led to a number of stable and robust algorithms for extended-horizon adaptive control (Ydstie et al., 1985, 1988). In the following sections we outline the algorithm for adaptive projective integration and show two test problems to show the utility of our approach. The first test problem examines a stiff system called the Brusselator that models a chemical reaction with oscillating states. Adaptive projective integration can integrate the Brusselator test problem two orders of magnitude faster than using explicit Euler's method. The second test problem is a stochastic simulation of DNA electrophoresis through a microfabriacted obstacle course. Adaptive projective integration is only able to make modest improvements in CPU time for the DNA simulation problem.

2. ADAPTIVE PROJECTIVE INTEGRATION

This section describes the algorithm for adaptive projective integration. The system to be integrated contains both fast and slow dynamics represented by

$$\frac{dx}{dt} = g_1(x,t) + \frac{1}{\epsilon} g_2(x,t) \tag{1}$$

[1] An affine function has the form $y = Ax + b$ where A is a matrix and b is a vector.

where $x \in \mathbb{R}^n$ and ϵ is a small number which indicates that the first term in the differential equation describes slow dynamics and the second term describes fast dynamics. Projective integration uses a detailed inner integration to damp out the fast dynamics and then uses an outer integration to extrapolate over a long time horizon (Gear and Kevrekidis, 2003). One such inner integration is explicit Euler's method with an integration time step, δt, at least as small as ϵ

$$x_{k+1} = x_k + \delta t\, g_1(x_k, t_k) + \frac{\delta t}{\epsilon} g_2(x_k, t_k) \tag{2}$$

which can be more simply written as $x_{k+1} = f(x_k)$ if we include t_k as a state variable. Because the integration time step, δt, is required to be small for numerical stability, integrating $x_{k+1} = f(x_k)$ to a long time horizon can be prohibitively expensive depending on the size and structure of $f(x)$. The Projective integration technique therefore integrates a small number of steps, then it fits an affine function to the simulation data, and finally it makes a long time horizon projection. This affine approximation of the system $f(x)$ takes on the form

$$y_{k+1} = Ay_k + a_0 + e_k \tag{3}$$

where $y_k = x_k$, $y_{k+1} \approx x_{k+1}$, A is a constant matrix where the entries are found by the method of least-squares, a_0 is a vector also fit using least-squares, and e_k contains the fitting errors. A projection N steps into the future can be made using

$$y_{k+N} = A^N y_k + \sum_{i=0}^{N-1} A^i \left(a_0 + e_{k+N-1-i}\right) \tag{4}$$

where the future errors $e_{k+1}, e_{k+2}, \ldots, e_{k+N-1}$ are unknown. In order to make the projection, we assume that the error is time-invariant so that $e = e_k \approx e_{k+1} \approx \ldots \approx e_{k+N-1}$. Using the identity for a geometric series we arrive at

$$y_{k+N} = A^N y_k + (a_0 + e)\left(I - A\right)^{-1}\left(I - A^N\right) \tag{5}$$

where the term $e\left(I - A\right)^{-1}\left(I - A^N\right)$ is the estimate of the error $x_{k+N} - y_{k+N}$.

Eq. (5) serves as the outer integration in the projective integration method. Once the projection y_{k+N} is made using Eq. (5) the inner integration is restarted taking $x_{k+N} = y_{k+N}$. Because the system f is approximated by the affine function (3) in order to make the long time horizon projections, the projections introduce some additional integration error beyond the discretization error associated with Euler's method. Before we summarize the projective integration method in an algorithm we will first outline theorem 1 which is used to determine the projection horizon as a function of the user specified error tolerance.

Theorem 1. Let κe be the user specified error tolerance and $\lambda_{\max} = \max_i\{|\lambda_i|\}$ where λ_i is an eigenvalue of the matrix A in Eq. (3). The projection horizon N is bounded by

$$N \leq \frac{\log\left(\kappa\lambda_{\max} - \kappa + 1\right)}{\log\left(\lambda_{\max}\right)} \tag{6}$$

so that the outer integration (5) does not introduce more error than κe.

Proof. A bound on the projection horizon N can be derived from bounding the projection error estimate by $e\kappa I$ in Eq. (5) which requires

$$\kappa I - \left(I - A\right)^{-1}\left(I - A^N\right) \geq 0 \tag{7}$$

where $M \geq 0$ means M is positive-semidefinite. If the matrix A in (5) is stable then $(I - A) \geq 0$ and N can be arbitrarily large, otherwise condition (7) is premultiplied by $(I - A)$ to yield

$$\left(\kappa - 1\right) I - \kappa A + A^N \leq 0 \tag{8}$$

using $(I - A) < 0$. The matrix A can then be diagonalized, $A = PDP^{-1}$, where $D_{ii} = \lambda_i$, $D_{ij} = 0$, to yield

$$\left(\kappa - 1\right) I - \kappa D + D^N \leq 0. \tag{9}$$

The maximum absolute eigenvalue $\lambda_{\max} = \max\{|\lambda_i|\}$ must then satisfy the condition

$$\kappa - 1 - \kappa\lambda_{\max} + \lambda_{\max}^N \leq 0 \tag{10}$$

which can be rearrange to give the bound on the projection horizon N in Theorem 1 $\quad\square$.

The algorithm for adaptive projective integration is as follows:

(1) Starting at x_k, integrate h steps forward using the inner integrator, $x_{k+1} = f(x_k)$, to generate data $x_k, x_{k+1}, \ldots, x_{k+h+1}$.

(2) Let $\phi = [x_k, \ldots, x_{k+h}]$ and $\Psi = [x_{k+1}, \ldots, x_{k+h+1}]$. Append a row vector of ones, $\mathbf{1}$, to the matrix ϕ so that $\Phi = [\phi; \mathbf{1}]$ where a semi-colon denotes a new row in the matrix. Fit the affine model $y_{k+1} = Ay_k + a_0$ using least-squares so that $\Theta = \Psi\Phi^+$ where Φ^+ is the pseudoinverse of Φ, found efficiently using a SVD, and $\Theta = [A, a_0]$.

(3) Diagonalize A and calculate $N^* = \left\lfloor \frac{\log(\gamma_{\max})}{\log(\lambda_{\max})} \right\rfloor$ where $\lfloor\cdot\rfloor$ is the floor operator, $\gamma_{\max} = \kappa\lambda_{\max} - \kappa + 1$, and $\lambda_{\max} = \max\{|\lambda_i|\}$ is maximum absolute eigenvalue of A. If $\lambda_{\max} > 1$ set the projection horizon to $N = \min\{N^{\text{spec}}, N^*\}$, where N^{spec} is the user specified projection horizon, otherwise $\lambda_{\max} \leq 1$ and set $N = N^{spec}$.

(4) Project forward N steps to $y_{k+h+1+N} = A^N y_{k+h+1} + a_0 \sum_{i=0}^{N-1} A^i$, set $x_{k+h+1+N} = y_{k+h+1+N}$, reset the index $k \leftarrow k + h + N + 1$, and go to step 1.

The emphasis of this approach is fast and cheap computations to accelerate long simulations. In certain cases it may be advantageous to replace the matrix A in step 2 with a strictly diagonal matrix B so that the cost of diagonalization can be avoided. This introduces a tradeoff with accuracy, however, as the diagonal matrix B contains less information than the full least-squares solution A and the projection $y_{k+h+1+N}$ is correspondingly less accurate. Ultimately using a strictly diagonal matrix B will require smaller projection horizons N which can lead to increased CPU time yet again. Another approach is to omit step 3 in the algorithm completely. This approach requires some experimentation to identify a projection horizon N^{spec} that appropriately balances accuracy and acceleration of the simulation.

3. TEST PROBLEMS

In this section we outline two useful example to illustrate the algorithm for adaptive projective integration. The first example problem is a Brusselator with rapidly replenished source which is a non-linear system that describes an oscillating chemical reaction. This example was also analyzed by Gear and Kevrekidis (2003). The following differential equations for the Brusselator with a rapidly replenishing source are

$$\begin{aligned} \frac{dx_1}{dt} &= \frac{p_1 - x_1}{p_2} - x_1 x_2 \\ \frac{dx_2}{dt} &= p_3 - (x_1 + 1)\, x_2 + x_2^2 x_3 \\ \frac{dx_3}{dt} &= x_1 x_2 - x_2^2 x_3 \end{aligned} \tag{11}$$

where the terms x_1 and p_3 represent the concentration of the reagents and the terms x_2 and x_3 represent the concentration of the products. The chemical reaction takes place in a large reservoir of reagents leading to the concentration, p_3, to be constant. The second reagent is rapidly replenished to its set point p_1 with a time scale p_2. The system has an unstable stationary point at $x_1 = p_1$, $x_2 = p_3$ and $x_3 = p_1/p_3$ and all other points lead to a stable limit cycle. The terms p_1, p_2 and p_3 are constant parameters with values $p_1 = 3$, $p_2 = 10^{-4}$, and $p_3 = 1$. The initial conditions are $x_1(0) = p_1$, $x_2(0) = p_3 + 0.1$, and $x_3(0) = p_1/p_3 + 0.1$. The system of differential of equations is stiff and is integrated using explicit Euler's method to $t_k = 10$ with a time step $\delta t = p_2 = 10^{-4}$. The results from explicit Euler's method are used as the standard to compare against the results from Adaptive Projective Integration.

Results are shown in figure 1 for different error factors $\kappa = 10^3$ and $\kappa = 10^6$. In step 1 of the algorithm the full simulation is integrated forward $h = 4$ steps. The affine model is then fit and the projection horizon is specified according to the error factor. When the linear model $y_{k+h+1} = Ay_{k+h} + a_0$ is stable the projection horizon is $N^{spec} = 10240$ otherwise the projection horizon is bounded by Eq. (6). In figure 1 we see that adaptive projective integration with an error factor $\kappa = 10^3$ leads to good agreement with data produced using Euler's method alone with correlation coefficients $r^2 = 0.79$, 0.81, and 0.79 for the states x_1, x_2, and x_3, respectively. When the error factor is set to $\kappa = 10^6$ the error increases substantially and the correlation coefficients drop to $r^2 = 0.01$, 0.02, and 0.01. Regardless of the large error introduced during the adaptive projective integration, we can see that the integration recovers quickly to the correct trajectory so that the error in the states at $t_k = 10$ is commensurate to when $\kappa = 10^3$. In this example the stable limit cycle helps to correct any over projections that occur. The CPU times for adaptive projective integration with $\kappa = 10^3$ and $\kappa = 10^6$ are 0.049 s and 0.016 s compared to 0.835 s using Euler's method alone. These computations were performed in MATLAB(R) using a desktop PC equipped with an Intel(R) i7 2.93 GHz quad-core processor ran in serial.

As the emphasis is on efficient computations, the algorithm for adaptive projective integration may be modified to better suit these needs. One approach is to omit step 3 of the

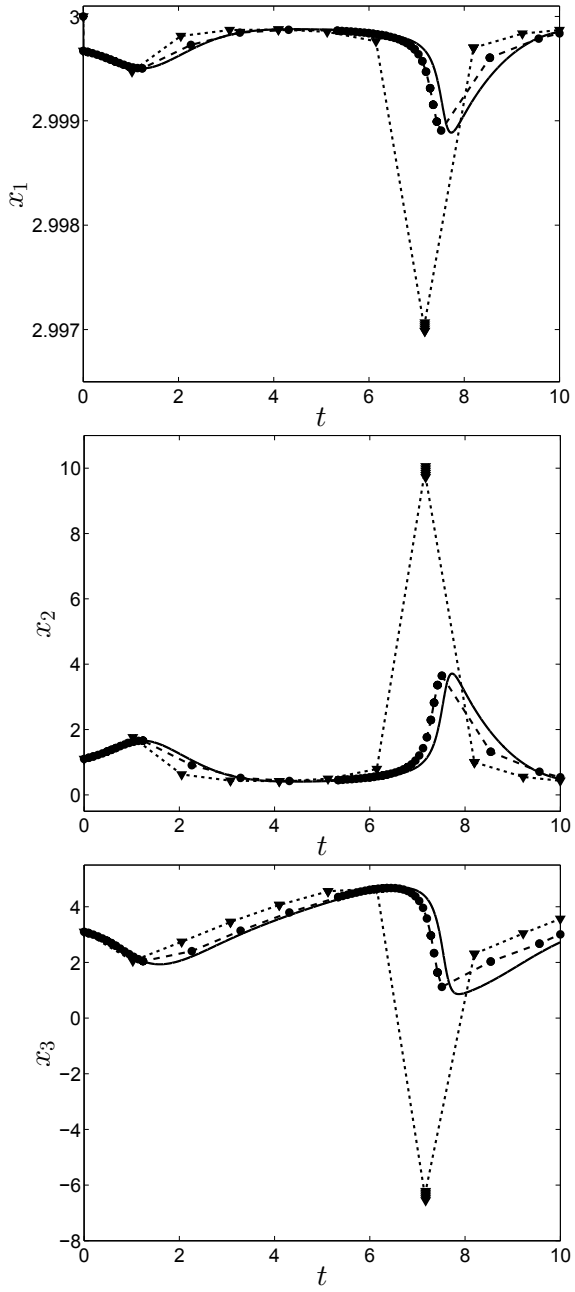

Fig. 1. Brusselator example results. Euler's method is compared to the results from adaptive projective integration. The lines (——), (− • −), and (· · ▾ · ·) corresponds to Euler's method, projective integration with error factor $\kappa = 10^3$, and projective integration with error factor $\kappa = 10^6$, respectively. The specified projection horizon is $N^{spec} = 10240$ for both cases.

algorithm which adjusts the projection horizon according to the user specified error factor κ and the eigenvalues of the fit matrix A. By omitting this step, the diagonalization of A can be avoided but the projection horizon cannot be corrected. The user must typically specify a smaller projection horizon so that the projective integration algorithm does not introduce too much error. Figure 2 shows the results from using projective integration with fixed projection horizons $N^{spec} = 2560$ and $N^{spec} = 10240$. The integration results from using projective integration with an integration horizon $N^{spec} = 2560$ are nearly indis-

tinguishable from the results generated by Euler's method with correlation coefficients $r^2 = 0.999$, 0.996 and 0.999 for states x_1, x_2, and x_3, respectively. As the projection horizon is increased to $N^{spec} = 10240$ the correlation coefficients drop to $r^2 = 0.010$, 0.026, and 0.013. The CPU times for projective integration with a fixed horizon are 0.006 s and 0.002 s using $N^{spec} = 2560$ and $N^{spec} = 10240$, respectively, compared to 0.835 s using Euler's method alone. Projective integration also outperforms commercial integrators designed for stiff differential equations such as ode23s in MATLAB(R) which requires 0.02 s of CPU time to integrate Eq. (11) to the default accuracy.

Another useful example simulates DNA migrating through a microfabricated obstacle course under the action of electrophoresis. Here DNA is represented by N_b beads connected by springs. A momentum balance yields the stochastic differential equation

$$dr = \left(F^{elec}(r) + F^{EV}(r) + F^{s}(r)\right) dt + \sqrt{2}\, dW \qquad (12)$$

where $r = [x_1, y_1, z_1, x_2, y_2, z_2, \ldots]^T$ is a vector containing the (x, y, z) coordinates of each bead, $F^{elec}(r)$ is a look up table containing the values for the force applied by the electric field in the obstacle course, $F^{EV}(r)$ is the excluded volume term that prevents the beads from overlapping, $F^{s}(r)$ is the spring force term that keeps the beads connected, and dW is a Wiener process represented by Gaussian white noise with zero mean and variance dt which accounts for Brownian motion.

The look up table for $F^{elec}(r)$ is generated by solving Laplace's equation $\partial^2 V/\partial x^2 + \partial^2 V/\partial y^2 = 0$ over the interior of the obstacle course, shown in figure 3, with homogeneous Neumann boundary conditions on the walls and Dirichlet boundary conditions of $V(0, y) = V_{app}$ and $V(4, y) = 0$. Laplace's equations is solved using MATLAB(R) PDE toolbox to generate $V(x, y)$ which is then numerical differentiated to yield the vector field $\zeta F^{elec}(x, y)$ where ζ is a unit fixing constant the places that force field in dimensionless units. The magnitude of the spring force term is given by

$$f_i = \frac{\nu q_i}{\left(1 - q_i^2\right)^2} - \frac{7\nu q_i}{\nu\left(1 - q_i^2\right)} + C q_i + D q_i \left(1 - q_i^2\right) \qquad (13)$$

where $q_i = \|r_{i+1} - r_i\|$ is the distance between bead $i + 1$ and bead i, ν is a unit fixing constant, C and D are constants, and the total spring force for each bead i is calculated by $F_i^s(r) = f_i\left(r_{i+1} - r_i\right)/q_i - f_{i-1}\left(r_i - r_{i-1}\right)/q_{i-1}$ (Underhill and Doyle, 2006). The excluded volume term in Eq. (12) prevents the beads from overlapping and is given by

$$F_i^{EV} = -\sum_j \frac{9\alpha\nu^{4.5}}{2}\left(\frac{3}{4\sqrt{\pi}}\right)^3 e^{-2.25\nu\|r_j - r_i\|^2}\left(r_j - r_i\right) \qquad (14)$$

where α is constant parameter that specifies the strength of the repulsion between beads (Jendrejack et al., 2002).

The stochastic differential equation (12) has $3N_b$ equations that are solved using semi-implicit Euler's method (Somasi et al., 2002) with N_t integration steps and is solved N_e multiple times to yield estimates of the first two moments of r. In our example we use $N_b = 12$, $N_t = 5 \times 10^5$, and $N_e = 100$ with an integration time step $\delta t = 5 \times 10^{-4}$. The simulation requires 70.14 minutes of wall-clock time

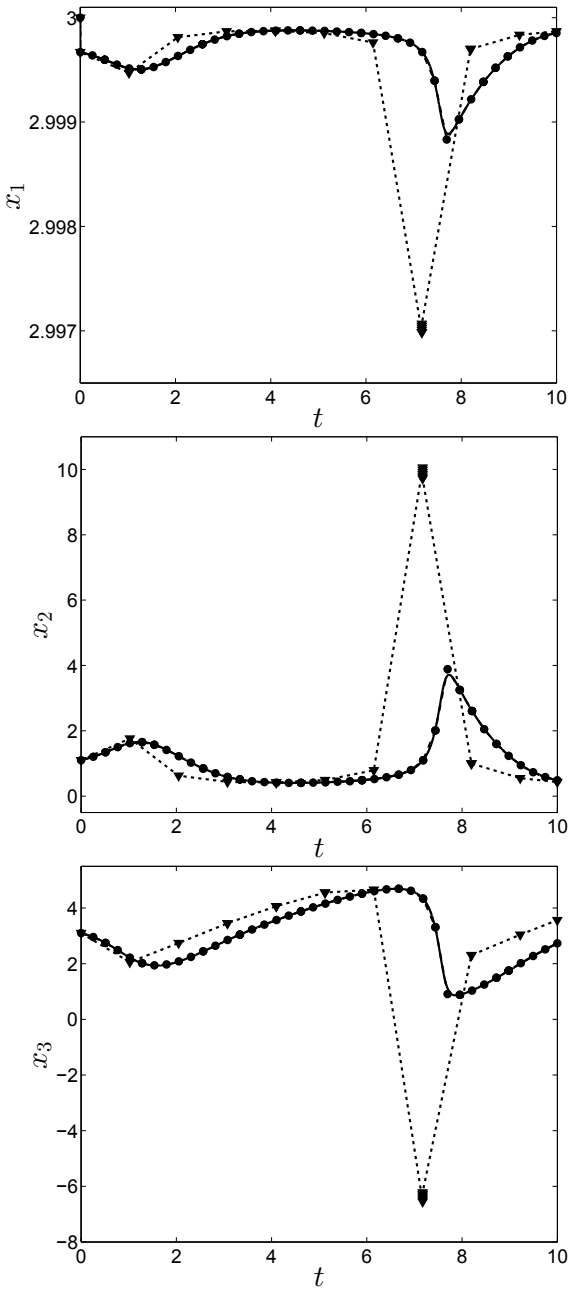

Fig. 2. Brusselator example results. Euler's method is compared to the results from adaptive projective integration. The lines (——), (− • −), and (· · ▾ · ·) corresponds to Euler's method and projective integration with fixed projective horizons $N^{spec} = 2560$ and $N^{spec} = 10240$, respectively.

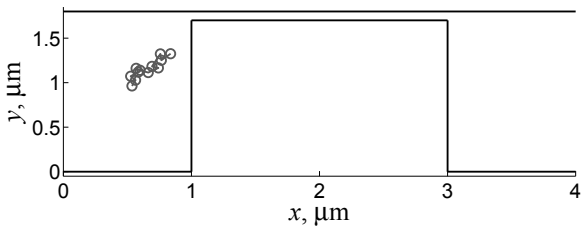

Fig. 3. DNA approximated by a bead-spring model in an obstacle course.

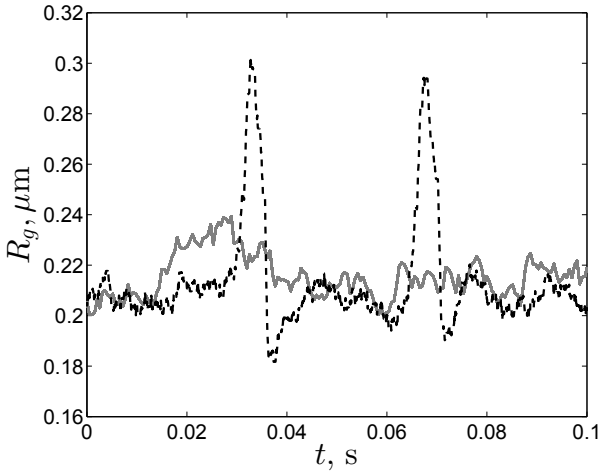

Fig. 4. Radius of gyration of DNA as it migrates through the obstacle course. Integration results of Eq. (12) using semi-implicit Euler's method (—) are compared against adaptive projective integration (− −).

to complete the simulation using a desktop PC equipped with an Intel(R) i7 2.93 GHz quad-core processor ran in parallel using MATLAB(R).

The simulation of DNA electrophoresis in an obstacle course generates the (x, y, z) coordinates for each bead as they evolve through time. The size of the DNA as is moves through the obstacle course is indicated by the radius of gyration,

$$R_g^2 = \frac{1}{N_b} \sum_{i=1}^{N_b} \left\langle (r_i - r_{cm})^2 \right\rangle \tag{15}$$

where r_{cm} is the center of mass of the DNA and the brackets $\langle \cdot \rangle$ indicate an ensemble average over the N_e simulation realizations. The radius of gyration from the simulation is shown in figure 4. We apply the version of our adaptive projective integration method which fits a strictly diagonal matrix B for outer-integration model $y_{k+1} = By_k + b_0$ where $y_k = \langle r(t_k) \rangle$. The projection horizon is set at $N^{spec} = 5000$ and step 3 of the algorithm adaptive projective integration algorithm is omitted. The results from figure 4 show that projective integration results in overshooting the actual trajectory of the simulation, but the stability and dissipative properties of the simulation quickly correct the overshoot and bring the results from the different integration techniques into qualitative agreement with each other. In general, quantitative agreement from stochastic simulations cannot be achieved without using a large number of realizations. The wall-clock time using adaptive projective integration is 57.48 minutes which is small decrease over the full simulation time of 70.14 minutes.

4. CONCLUSIONS

Adaptive projective integration is a method for computationally inexpensive integration of differential equations. In the algorithm, the differential equation is integrated forward for a small number a steps which generates data that is used to construct an affine model. The linear model is used to project forward a large number of steps. Based on the eigenvalues of the linear model the projection horizon can be adjusted to help avoid large errors during the integration. We used the Brusselator problem as an example to highlight the different features of our approach. We found that using projective integration with a fixed projective horizon yields the best tradeoff between computational speed up and accuracy giving nearly an identical answer to Euler's method but with two orders of magnitude speed up in CPU time. Less impressive speed up is observed using the stochastic simulation of DNA.

REFERENCES

Baldea, M. and Daoutidis, P. (2007). Control of integrated process networksa multi-time scale perspective. *Computers & chemical engineering*, 31(5), 426–444.

Gear, C.W. and Kevrekidis, I.G. (2003). Projective methods for stiff differential equations: problems with gaps in their eigenvalue spectrum. *SIAM Journal on Scientific Computing*, 24(4), 1091–1106.

Jendrejack, R.M., de Pablo, J.J., and Graham, M.D. (2002). Stochastic simulations of DNA in flow: Dynamics and the effects of hydrodynamic interactions. *The Journal of chemical physics*, 116, 7752.

Kevrekidis, I.G., Gear, C.W., Hyman, J.M., Kevrekidid, P.G., Runborg, O., and Theodoropoulos, C. (2003). Equation-free, coarse-grained multiscale computation: Enabling mocroscopic simulators to perform system-level analysis. *Communications in Mathematical Sciences*, 1(4), 715–762.

Kevrekidis, I.G. and Samaey, G. (2009). Equation-free multiscale computation: Algorithms and applications. *Annual review of physical chemistry*, 60, 321–344.

Rico-Martínez, R., Gear, C.W., and Kevrekidis, I.G. (2004). Coarse projective kMC integration: Forward/reverse initial and boundary value problems. *Journal of Computational Physics*, 196(2), 474–489.

Saksena, V.R., O'reilly, J., and Kokotovic, P.V. (1984). Singular perturbations and time-scale methods in control theory: survey 1976–1983. *Automatica*, 20(3), 273–293.

Siettos, C.I., Graham, M.D., and Kevrekidis, I.G. (2003). Coarse Brownian dynamics for nematic liquid crystals: Bifurcation, projective integration, and control via stochastic simulation. *Journal of Chemical Physics*, 118(22), 10149–10156.

Somasi, M., Khomami, B., Woo, N.J., Hur, J.S., and Shaqfeh, E.S. (2002). Brownian dynamics simulations of bead-rod and bead-spring chains: numerical algorithms and coarse-graining issues. *Journal of Non-Newtonian Fluid Mechanics*, 108(1), 227–255.

Underhill, P.T. and Doyle, P.S. (2006). Alternative spring force law for bead-spring chain models of the worm-like chain. *Journal of Rheology*, 50, 513.

Vora, N. and Daoutidis, P. (2001). Nonlinear model reduction of chemical reaction systems. *AIChE Journal*, 47(10), 2320–2332.

Ydstie, B.E., Kemna, A.H., and Liu, L.K. (1988). Multivariable extended-horizon adaptive control. *Computers & Chemical Engineering*, 12(7), 733–743.

Ydstie, B.E., Kershenbaum, L.S., and Sargent, R.W.H. (1985). Theory and application of an extended horizon self-tuning controller. *AICHE journal*, 31(11), 1771–1780.