CPU and GPU Parallel Kramers-Klein Calculations

Bogdan I. Iaparov, Anton N. Karmatsky, and Alexander S. Moskvin

Ural Federal University, Yekaterinburg, Russia
bogdan.iaparov@urfu.ru

Abstract. The Kramers-Klein equation is a special Fokker-Planck equation describing the brownian motion in a potential. This equation is used in different areas of biology, chemistry and physics. In this paper we present the comparison between numerical and analytical solutions in case of a linear force and usage of multi-core CPU and GPU for solving Kramers-Klein equation with different boundary conditions. It was shown that the GPU calculations under reflecting boundary conditions are slower as compared with other boundary conditions because in the first case we need to call two kernels sequentially.

Keywords: Kramers-Klein equation · parallel algorithm · CUDA · OpenMP

1 Introduction

Brownian motion(BM) is the random motion of a particle in a fluid. Hereafter we're discussing the one-dimensional case. BM can be mathematically described in two ways. The first one is a Langevin equation, a stochastic differential equation for the particle's coordinate:

$$\begin{aligned} m\ddot{x} + m\gamma\dot{x} + mf'(x) &= m\Gamma(t) \\ < \Gamma(t) > &= 0 \\ < \Gamma(t)\Gamma(t') > &= 2\gamma k_B T\delta(t'-t) \end{aligned} \tag{1}$$

here m is the mass of the particle, γ is the damping constant, $f'(x)$ is the stationary potential force per mass due to the potential $-f(x)$, $\Gamma(t)$ – delta-correlated Gaussian random process. It's clearly seen from (1) that $x(t)$ and $v(t) = \dot{x}(t)$ are stochastic quantities due to $\Gamma(t)$ in the equation. Therefore we may ask for probability density function (PDF)($\rho(x, v, t)$) of finding particle in a coordinate x with a velocity v at time t. An equation for PDF in case of BM in an external potential field is called Kramers-Klein (KK) equation:

$$\frac{\partial\rho}{\partial t} = -v\frac{\partial\rho}{\partial x} + f'(x)\frac{\partial\rho}{\partial v} + \gamma\frac{\partial v\rho}{\partial v} + \frac{\gamma k_B T}{m}\frac{\partial^2\rho}{\partial v^2} \tag{2}$$

This equation was used first to describe reaction kinetics [1], but later it turned out that it can be used in very different fields of physics such as relaxation of dipoles, superionic conductors and so on [2]. In the paper we address parallel numerical solutions of (2).

2 Numerical Solution

Despite intensive analytical studies of KK equation [2], analytic solution can be found only in special cases. This makes numerical solutions as a significant resource for studying stochastic processes. Difficulties of studying KK equation numerically come from the fact, that this partial differential equation is of a mixed type. The problem behaves like a parabolic equation in the v-direction and like a hyperbolic equation in the x-direction. Moreover, we have different types of hyperbolic equations in the x-directions for $v > 0$ and $v < 0$. We used a numerical method, computational domain and the uniform grid as in [3].Central differences scheme were used for the discretization in v - direction of first and second derivatives, for x - direction the upwind scheme discretization was used, for t - direction forward Euler scheme was used. It is an explicit method that efficiently deals with different types of boundary conditions and potential fields. Convergence and stability of the method is discussed in Ref.[3]. The accuracy of the method is defined as $O(\Delta x + \Delta v^2 + \Delta t)$.

3 Boundary Conditions

Say we have a rectangular 2D computational domain(in x and v direction, $x_L \leq x \leq x_R$,$v_L \leq v \leq v_R$) with uniform grid. In the v-direction the problem doesn't have boundaries and we assume that computational domain is so large that we can assume the BC as follows:

$$\rho(x, v_L, t) = \rho(x, v_R, t) = 0 \tag{3}$$

for all x, t.

In the x-direction we consider three types of BC:

- Dirichlet. In analogy with v-direction we assume that computational domain is so large that particle can't be in boundaries of domain:

$$\rho(x_L, v, t) = \rho(x_R, v, t) = 0 \tag{4}$$

  for all v, t.
- Absorbing. In this case we only have physical boundaries defined at x_L for $v < 0$ and at x_R for $v > 0$:

$$\begin{aligned} \rho(x_L, v, t) = 0, v > 0 \\ \rho(x_R, v, t) = 0, v < 0 \end{aligned} \tag{5}$$

  The physical meaning of the absorbing BC is that there is no particle flux from the boundary into the interior.
- Reflecting. In this case we have the BC as follows:

$$\begin{aligned} \rho(x_L, v, t) = \rho(x_L, -v, t) \\ \rho(x_R, v, t) = \rho(x_R, -v, t) \end{aligned} \tag{6}$$

  The physical meaning of the reflecting BC is that no particle would be lost at the boundary and all particles will be reflected back into the interior.

4 Parallelization

Parallel CPU calculations were implemented using OpenMP, the GPU calculations were done using CUDA. The numerical method we used [3] is explicit, the PDF inside the domain is fully calculated from PDF at previous time, as a consequence, calculation of PDF in each interior's node is independent of another nodes at the same time:

$$\rho(x, v, t_{i+1}) = \rho(x, v, \rho(x, v, t_i)) \tag{7}$$

hence the method can be easily parallelized using "stencil" parallel pattern.

In case of CPU parallelization, the boundaries were calculated separately and without parallelization because the highest performance was shown in this case. In case of GPU parallelization the boundaries were calculated in one CUDA kernel with interior in case of the Dirichlet and absorbing BC.

In case of the reflecting BC two CUDA kernels run sequentially: the first one calclulates interior and half of boundaries in x-direction with $v > 0$ and the second one calculates half of the boundaries in x-direction with $v < 0$. We did it because in this case PDF on the boundary depends on PDF in another points at the same time(see (6)). As a result, GPU calculations with reflecting BC are slower than with the Dirichet or absorbing ones.

In CUDA calculations only global memory was used.

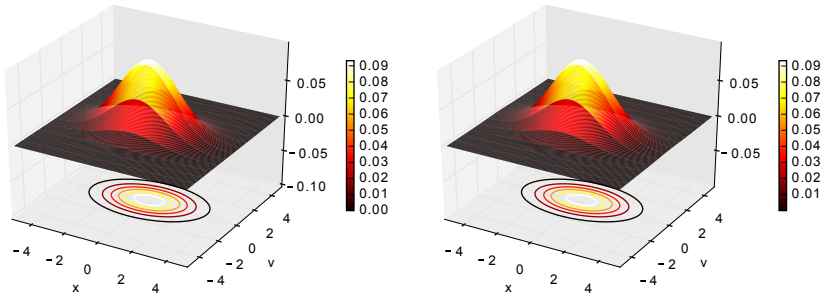
5 Results

Program for CPU was written in C++ and compiled with gcc (g++ -O3) compiler. GPU implementation is also in C++ and CUDA part was compiled with proprietary nvcc compiler while g++ has been used for the host code. CPU calculations were done on PC(4x Intel Core i5 3.2 GHz, NVIDIA GTX 750), GPU calculations were done on PC and Uran supercomputer(NVIDIA Tesla M2050) from Institute of Mathematics and Mechanics UrB RAS. Calculations were done using double precision floating point numbers.

First, we compared results of the numerical solution with the analytic one in case of a linear force (parabolic potential). The analytic solution in this case was taken from [2] and is a two-variable Gaussian distribution of coordinate and velocity with the time-dependent means and variances. The error was calculated as a discrete norm:

$$\varepsilon(t) = \left(\Delta x \Delta v \sum_{i,j} (\rho(x_i, v_j, t) - \rho_a(x_i, v_j, t))^2 \right)^{0.5} \tag{8}$$

where ρ_a is an analytic solution. In Figure 1 the numerical and analytic PDFs are shown at $t = 12$ (dimensionless units). Number of nodes in the x-direction is 2049 and in the v-direction is 129. It is clearly seen that numerical solution does approximate the analytic one quite accurately.



(a) Numerical solution (b) Analytic solution

Fig. 1: PDF in case of potential $0.18x^2$, $\gamma = 3$. 1a is a numerical and 1b is an analytic solution at $t = 12$. At $t = 0$, the particle has zero velocity and $x = 0$. $\varepsilon(12) = 8.3e - 4$. All variables are in dimensionless units.

Then we looked on the performance of parallel CPU and GPU calculations comparing with the serial CPU ones with different BC. Calculations were done on an uniform grid with 2049 nodes in x-direction and with 257 nodes in v-direction, total of 526593 nodes. Time step was $1.4e - 4$ and calculations were stopped when $t = 10$. The results are shown in the Table 1. Parallel solving of KK equation provides a significant speed up of the calculations even in case of the parallel CPU solution.

Table 1: Duration of solving KK equation on a 2049x257 grid with $\Delta t = 1.4e-4$ to $t = 10$. In brackets is written the speed up factor comparing with serial CPU calculations(1x CPU).

Type of BC / Hardware	Dirichlet	Absorbing	Reflecting
1x CPU	592.6 s	601.2 s	600.8 s
4x CPU	218.3 s (2.7x)	226.74 s (2.7x)	220 s (2.7x)
1x GPU(GTX 750)	114.7 s (5.2x)	116.6 s (5.2x)	122 s (4.9x)
1x GPU(Tesla M2050)	39.8 s (14.9x)	40 s (15x)	43.9 s (13.7x)

Calculations on grids with different N_p and N_x have shown that duration of solving KK equation depends linearly on $N = N_p \cdot N_x$(data not shown).

6 Discussion

KK equation is a partial differential equation of mixed type which describes the time evolution of a PDF of a brownian particle in an external potential. In

literature the Kramers problem(find the escape rate of a Brownian particle from a potential well) is usually solved analytically using some assumptions(harmonic approximation near minima [1]), piecewise-linear approximation [4] and high energy barriers approximation[5].

Finding PDF is a much harder task and can be solved analytically only in a few cases. In this paper we solved the KK equation numerically. Solving KK equation numerically allows us to get a solution for both finite and infinite geometry and arbitrary potential field. We showed that both CPU and GPU parallel computing provide a significant gain in the solving time. Even home PC's GPU can solve the equation for an acceptable time.

BC type is important from a physical viewpoint on the problem when it has a finite geometry. Different types of collisions of brownian particle with a wall: the elastic(reflecting BC) and inelactic (absorbing BC) ones influence the PDF very significantly[3].

From a viewpoint of computations, in case of the reflecting BC, the GPU parallelization is less effective because in one iteration we need to call two kernels sequentially and Table 1 shows quite clearly, that the GPU acceleration is smaller in case of the reflecting BC.

From a viewpoint of computations, further works should include more accurate and efficient schemes for solving the KK equation and its parallelization[7]. Further physical researches should include numerical studies of the KK equation for the ion channel kinetics[6] with different physically reasonable potentials, infinite and finite geometries.

Acknowledgements. Supported by the Russian Scientific Foundation, project # 14-35-00005.

References

1. Kramers, H.A.: Brownian motion in a field of force and the diffusion model of chemical reactions, Physica. 7: 284-304 (1940)
2. Risken, H.: The Fokker-Planck Equation, Springer, Berlin (1989)
3. Araújo, A., Das, A.K., Sousa, E.: A numerical approach to study the Kramers equation for finite geometries: boundary conditions and potential fields, J. Phys. A: Math. Theor. 48:045202 (2015)
4. Goychuk, I., Hänggi, P.: The role of conformational diffusion in ion channel gating, Physica A. 325:9–18 (2003)
5. Mel'nikov, V.I., Meshkov, S.V.: Theory of activated rate processes: Exact solution of the Kramers problem, J. Chem. Phys. 85:1018–1027 (1986)
6. Sigg, D.: Modeling ion channels: Past, present, and future, J Gen Physiol., 144: 726 (2014)
7. Shu, C.-W.: Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws, Advanced Numerical Approximation of Nonlinear Hyperbolic Equations,Lecture Notes in Mathematics series, 1697: 325432 (2006)