A FFT-based convolution method for mixed-periodic Boundary Condition

Zhe Chen*[1]

[1]Courant Institute of Mathematical Sciences, New York University

May 2020

1 Introduction

Convolutions are widely used to simulate colloidal system. The interactions between colloidal particles in Stokes flow can be described by their mobility, which is a function of the displacement between them. If the force applied on particles are given, we can compute velocity of particles by convolution of the mobility kernel and the force. This convolution is expensive and critical for simulating dynamics of large scale colloidal particles, which makes it essential that we have a cheap and accurate numerical method for computing them. Let's assume we compute a discrete convolution of two periodic sequences with length N in each direction in d dimensions. If we directly compute the convolution, it would cost $O(N^{2d})$, which is generally not affordable in high dimensions. One of the most popular and promising method that we could use to compute them faster is the Fast Fourier Transform (FFT), which cuts down complexity dramatically to $O(N^d \log(N))$, since a convolution becomes dot production in Fourier space. In most colloidal system, nevertheless, it's not trivial to use finite sequence convolutions to represent convolutions of the force and the mobility kernel. For example, the force field is usually considered as finitely supported or infinitely periodic, but the periodicity can be different in different directions, which we call mixed-periodicity, and the kernel can be long-ranged, which means slow-decaying. Moreover, sample points of velocity might be non-uniform since we only need velocity of the particles in most cases. In this report, we'll first introduce a Stokesian system of suspension colloidal particles above a wall, where x and y direction is periodic but z direction is aperiodic, in section 2. The kernel in this problem has most of the challenges we meet in practice to do convolutions. Then, a general library to compute convolutions with mixed periodicity boundary condition, uniform/non-uniform grid and any kind of kernel is developed in section 3. In the end, we'll use this library to compute convolutions in the this colloidal system in section 4.

2 Colloidal particles near a wall and splitting method

For a colloidal system near a wall, many interesting collective phenomenon could happen as a result of long-range hydrodynamic interactions. For example, if we apply uniform force in x-direction, which is component of gravity tangent to the inclined plane, on uniform-distributed particles, it will develop finger-shape front gradually (here's an intuitive video[1] in real physical experiments). Those very small colloidal particles have equal radius and almost at the same gravitational height, which has the same magnitude with radius, as a result of balance between Brownian motion and gravity. At $t = 0$, they are uniformly distributed in a rectangle strip and given a uniform force in x-direction. Then they moves in this viscous fluid with low Reynold number. Although it's uniform in y at the beginning, it gradually develop some special structure in y, which interests us to simulate it numerically.

In our numerical model, we will suppose that the boundary condition is periodic in y but aperiodic in x. In the domain $L_x \times L_y$, it's equal-space meshed. Particle density ρ is given on center point $x_m =$

*zc1291@cims.nyu.edu

[1]https://cims.nyu.edu/~zc1291/public/conv/experiment/PureWater_unaligned.mp4

$(m-1/2)h,\ y_n = (n-1/2)h,\ m,n = 1,...,N$. Initial condition is given that $\rho = \rho_0$ in a strip $L_x^0 \times L_y$ at the left of the domain. Here force f is proportional to particle density ρ in +x direction. Thus velocity v can be computed by the convolution of ρ and mobility of the particles, i.e. RPY-Swan kernel[2] K.

$$v = K * \rho \tag{1}$$

However, this so-called RPY-Swan kernel is extremely complicated and long-ranged. Periodicity of this problem is mixed. It needs to be carefully treated to use FFT-based convolutions.

Our group used to use direct convolution method, paralleled by GPU, to compute velocity. The periodic BC in y is mimicked by adding finitely many periodic boxes on both side in y of original domain. However, this method is expensive, $O((MN^2)^2)$ for M image boxes, and converges slowly because of slow decay of RPY-Swan kernel. What's worrying is the slow decay of K, which makes it difficult to simulate large amount of particles. In this report, I'm trying to take advantage of FFT to build a fast and cheap method to compute this convolution that has long-range kernel and mixed-periodicity.

3 Set up and basic algorithm

In this section, we'll develop a general library to compute convolutions with FFT. For simplicity, we first use one-dimension convolutions to introduce the algorithm, with higher dimension convolutions built on these elements. For functions f and g, denote $h = f * g$, by the convolution theorem,

$$h = f * g = \mathcal{F}^{-1}(\mathcal{F}(f)\ \cdot\ \mathcal{F}(g)), \tag{2}$$

where $\cdot$ denotes inner product, $\mathcal{F}(\cdot)$ denote Fourier transform, $\mathcal{F}^{-1}(\cdot)$ denotes the inverse Fourier transform and $*$ denotes convolutions.

In computational perspective, generally, we can only compute convolutions of discrete finite sequences, i.e. evaluations of functions on grids. Here we first assume these evaluations are on uniform grids, then we'll explore the non-uniform grid case, i.e. only particles' velocities are needed, in section 3.3. Mostly, one of the functions should be finitely supported or infinite but periodic. There are basically two classifications as following and we will use these two as fundamental elements to use:

1. One of the two functions is periodic, say f is periodic but g is aperiodic,
2. Both f and g are aperiodic, but one of them is, say f, is finitely supported.

3.1 Case 1. One function is periodic

Assume f is periodic in $[0, L]$ and g is aperiodic in $\mathbb{R}$. We evaluate f and g on uniform grids $f_n = \rho(L \cdot n/N)$ and $g_n = g(L \cdot n/N)$. By periodicity of f, the convolution of these two functions would be written as form of periodic summation,

$$h_n = (f * g)_n = \sum_{l=-\infty}^{+\infty} f_l g_{n-l} = \sum_{l=0}^{N-1} f_l g_{n-l}^N, \tag{3}$$

where $g_n^N = \sum_{p=-\infty}^{\infty} g(n - pN),\ n = 0, ..., N-1$ is defined as periodic summation. Now, convolutions of g and f in $\mathbb{R}$ is discretized as circular discrete convolutions (eq.3) of two N length sequences f_n and g_n^N.

Whether or not we can compute the periodic summation g^N easily depends on how fast g decays. We shall thus consider two cases: in the first we suppose that g_n decays very quickly as $n \to \pm\infty$ and only several images are needed to compute it, and in the second we suppose that it decays slowly and it cannot be computed by truncating the periodic summation.

3.1.1 Case 1.(a) Short-ranged function g

For g that is short-ranged, which means g decays fast and usually exponentially, the circular summation can be truncated as

$$g^N \approx \sum_{p=-p_{max}}^{p_{max}} g_{n-pN}, \tag{4}$$

[2]https://github.com/CecilMartin/MyResearchRecord/blob/master/doc/Kernel_sedimentation_1.pdf

where p_{max} is some threshold chosen based on the decay of g_n. If g_n really does decay quickly, then we can choose p_{max} to be small without incurring a large error. By making this approximation, we may compute g^N in $O(Np_{max})$ operations, which is small enough but still gives good accuracy. We can then compute $f * g$ in $O(N\log(N) + Np_{max})$ operations.

3.1.2 Case 1.(b) Long-ranged function g

However, if g is long-ranged, which means it decays slowly. The cutoff has to be really large to avoid big error, which requires large computational efforts and makes it impossible to reach $O(N\log(N))$. If we are lucky enough to know analytic Fourier transform of g, however, we could evaluate the continuous Fourier transform at discrete Fourier transform modes $k_n = 2\pi n/L$ and then we don't need to deal with slow-decaying periodic summation.

$$n \in \mathcal{I} = \begin{cases} -N/2, \dots, -1, 0, 1, \dots, N/2-1, & if\ N\ is\ even \\ -(N-1)/2, \dots, -1, 0, 1, \dots, (N-1)/2, & if\ N\ is\ odd \end{cases} \tag{5}$$

But that's not always the case since it's difficult to get analytic Fourier transform. In section 2, we'll introduce a splitting method to help us out through this.

3.2 Case 2. Both functions are aperiodic, but one of them is finitely supported

For aperiodic f and g, the circular discrete convolution does not work. However, a zero-padding trick can make it happen. Consider f is finitely supported in $[0, L]$ and g is not finitely supported. If we pad N-length $[f_0, ..., f_{N-1}]$ with zeros to be $(2N-1)$-length vector $\tilde{f} = [f_0, ..., f_{N-1}, 0, ..., 0]$, let $\tilde{g} = [g_0, ..., g_{N-1}, g_{-(N-1)}, ..., g_{-1}]$. Then we compute a circular discrete convolution 2 $\tilde{f} * \tilde{g}$ and take the first N components. That's exactly the needed convolution h since the right-most non-zero component of f would not meet the left-most. By this zeros-padding trick, the convolution of two aperiodic functions, one of which is finitely supported, could be solved by a circular discrete convolution and then solved quickly by FFT. Actually, in implementation, I double zero-padding it to $2N$ length rather than $2N-1$ to keep length of vector to be power of 2 to benefit FFT algorithm, so that $\tilde{g} = [g_0, ..., g_{N-1}, g_{-N}, g_{-(N-1)}, ..., g_{-1}]$. Thus, if one of the functions is finitely supported, the convolutions can be done really quickly by zero-padding and FFT.

3.3 Case 3. Non-uniform grids, particle-mesh

In simulations, we usually care about the evolution of particles only. These particles are not generally uniform distributed. So, in this case, f is given on non-uniform locations x_k with $f(x_k) = f_k$, $k = 1, \cdots, n$. Assume $g \in \mathcal{C}^\infty \cap \mathcal{L}^2 \cap \mathcal{L}^1$, we need to evaluate $h = f * g$ on x_k.

If f is periodic and finitely supported and analytic Fourier transform of g is known, we can easily get $h_k = h(x_k)$ through NUFFT with spectral accuracy. However, we are unable to get analytic Fourier transform of g in most cases and we want to switch it to aperidoc case. Then we turn to periodic summation similar to eq. 3. Function f can be written as $f(x) = \sum_{k=1}^n f_k \delta(x - x_k)$. Suppose periodic box of f is $[0, L]$, then $h(x) = \int_0^L f(y) g^L(x-y)dy$. The circular summation can be truncated similar to eq. 4:

$$g^L(x) = \sum_{p=-\infty}^{\infty} g(x - pL) \approx \sum_{p=-p_{max}}^{p_{max}} g(x^* - pL) \triangleq \widetilde{g^{L,p_{max}}}(x),\ x \equiv x^* \in [0, L] \mod L \tag{6}$$

Then $\tilde{h}(x) = \int_0^L f(y)\widetilde{g^{L,p_{max}}}(x-y)dy = \sum_{k=1}^n f_k \widetilde{g^{L,p_{max}}}(x - x_k)$ can be computed directly as reference. The truncation error here $||\tilde{h}(x) - h(x)||$ depends on tail behavior of g, e.g. exponential decaying for Gaussian. However, unlike the uniform grid case, NUFFT method to compute convolution of aperiodic and finitely supported function f and $\widetilde{g^{L,p_{max}}}$ will not be machine accurate but spectral accurate, which is good enough. The framework of NUFFT method here is pretty much similar to uniform grid case. First, we double zero pad function f to periodic function $\tilde{f}$ on $[0, 2L]$, and then evaluate the convolution $\tilde{h}(x) = \int_0^{2L} \tilde{f}(y)\widetilde{g^{L,p_{max}}}(x-y)dy$ on $x \in [0, L]$. This integral has no error with $\tilde{h}$ on $[0, L]$.

We first use NUFFT (FINUFFT [1] package) to compute Fourier transform of $\tilde{f}(x) = \sum_{k=1}^{n} f_k \delta(x - x_k)$ on sets $\mathcal{I}$ (eq. 5). The error is controlled by grid points within support of windows function, which is described in Barnett and Magland [1]. For given N modes in $\mathcal{I}$ and error ϵ, FINUFFT automatically gives $\hat{\bar{f}}$ such that $||\hat{\bar{f}} - \hat{\tilde{f}}||_\infty \leq \epsilon ||\tilde{f}||_\infty = \epsilon ||f||_\infty$. Then, we compute product in Fourier space $\hat{\bar{h}} = \widehat{\widetilde{g^{L,p_{max}}}} \cdot \hat{\bar{f}}$. And another NUFFT of $\hat{\bar{f}}$ gives $\bar{h}_k = \bar{h}(x_k)$ with error ϵ. By convolution theory, $\bar{h}(mh) = h \sum_{k=1}^{N} \bar{f}(kh) \widetilde{g^{L,p_{max}}}(mh - kh)$, $h = 2L/N$, which is circular discrete convolution on equi-spaced grid. Since $\bar{f}$ is given by Fourier modes $\hat{\bar{f}}$, it's periodic and $\mathcal{C}^\infty$. Thus, this trapezoidal rule of integral is exponentially accurate. Since the Fourier modes are determined by $\bar{h}(mh)$, $||\hat{\bar{h}} - \hat{\tilde{h}}|| = ||\bar{h} - \tilde{h}|| \leq C_1 \exp(-C_2 N)$. Then, by NUFFT, the evaluation of Fourier modes $\hat{\bar{h}}$ on x_k gives $\bar{h}_k$ and the vector error

$$||\bar{h}_k - \tilde{h}_k|| / ||\tilde{h}_k|| \leq C_1 \exp(-C_2 N) + C_3 \epsilon \tag{7}$$

In conclusion, the NUFFT based method will give spectral accuracy $||\bar{h}_k - \tilde{h}_k||$ compared with direct summation of $\tilde{h}_k$. The truncation error $||h_k - \tilde{h}_k||$ induced by truncated periodic summation (eq. 6) depends on decaying of function g

4 Application of this convolution library on the colloidal system

To simulate the colloidal system in section 2, we need to compute the convolution (eq. 1). We introduce two methods in this section using the convolution library.

4.1 Truncate the periodic summation

First, we can easily develop a method that is equivalent with direct convolutions that uses extra periodic boxes, but uses FFT instead.

In y-direction, it's periodic for ρ but aperiodic for K. Hence we can use 3.1 in the convolution library. If we truncate periodic summation in eq.(3) at M on both sides, $K_n^N = \sum_{p=-M}^{M} K(n - pN)$, then this circular discrete convolution in eq.(3) is equivalent with the direct convolution that has M image boxes on both sides. In x direction, it's aperiodic and we use 3.2 in the library. Thus, the algorithm is to first compute truncated periodic summation in y direction and then compute 2D FFT to get the convolution. We verify with numerical tests that this is equivalent to direct convolution method but with much less computation.

4.2 Split the long-range kernel

In this sub-section, we consider how to deal with the slowly-decaying kernel. As we stated in 3.1.2, if the analytic Fourier transform of the Kernel is known, the FFT can be used to compute convolutions of periodic ρ even with a long-ranged K. Unfortunately, the RPY-Swan kernel is too complicated to find its analytic Fourier transform. It is, however, possible to find the analytic Fourier transform of the RPY-Swan kernel in the limit as $a \to 0^+$. This limiting kernel K_{RB} is known as the Rotne-Blake kernel. The RPY-swan and Rotne-Blake kernels are asymptotically the same as $(x, y) \to \infty$ because the radius a makes no difference to the generated velocity at long distances. The way we develop its FT in y direction is shown in this Maple sheet [3]. Hence, we can split the kernel into a long-ranged, slowly decaying part, the Rotne-Blake kernel, and a short-ranged, quickly decaying correction:

$$K_{RPY} = K_{RB} + (K_{RPY} - K_{RB}).$$

We can then evaluate v as $v_l + v_s$, where

$$v_l = K_{RB} * \rho, \qquad \text{and} \qquad v_s = (K_{RPY} - K_{RB}) * \rho.$$

This is the so-called splitting method. Then it can be computed by the convolution library after splitting. The convolution in v_l is in 3.1.1 in the y direction and the convolution in v_s is in 3.1.2 in the y direction. Both convolutions are in 3.2 in the x direction, as the density ρ has finite support in the x direction.

[3]https://github.com/CecilMartin/MyResearchRecord/blob/master/doc/RotneBlakeFourier.pdf

4.3 Conclusions

By using this convolution library, we can compute the velocity of the particles in mixed-periodic condition and uniform/non-uniform grids. It is equivalent with direct convolution mathematically on uniform grids, i.e. within machine error, but much more quickly in $O(N\log(N))$ for truncated periodic boxes. It's spectral accurate with direct truncated periodic summation for non-uniform grids. If analytic Fourier transformation of the kernel or its long-range part is provided, we can compute the velocity with spectral accurate but still $O(N\log(N))$ complexity. Code for this project is on Github[4].

References

[1] Alex H. Barnett, Jeremy F. Magland, and Ludvig af Klinteberg. "A parallel non-uniform fast Fourier transform library based on an "exponential of semicircle" kernel". In: *arXiv:1808.06736 [cs, math]* (Apr. 2019). arXiv: 1808.06736. URL: http://arxiv.org/abs/1808.06736 (visited on 11/05/2019).

[4] https://github.com/CecilMartin/convolution_fft