1 Stokes flow

This repository concerns the solution of Stokes equations,

$$\eta \nabla^2 \boldsymbol{u} - \nabla p = -\boldsymbol{f}, \tag{1.1}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{1.2}$$

on some domain $\Omega := (x, y, z) \in [0, L_x] \times [0, L_y] \times [0, L_z]$ subject to some boundary conditions (BCs). Here, the velocity $\boldsymbol{u} = \begin{pmatrix} u & v & w \end{pmatrix}^T$, and $\boldsymbol{f} = \begin{pmatrix} f & g & h \end{pmatrix}^T$ are smeared δ-function sources representing particles with an effective hydrodynamic radius of R_h. The boundary conditions, imposed on the faces of Ω, can be combinations of `open`, `periodic`, `mirror wall` or `inverse mirror wall`. The latter two correspond to a confined geometry, and some BC must be prescribed at both endpoints of each axis.

The source term $\boldsymbol{f}$ consists of monopole and dipole terms with "exponential of a semicircle" (ES) kernel [1] envelopes Δ;

$$\Delta(z; \alpha, \beta) = \frac{1}{\displaystyle\int_{-\alpha}^{\alpha} e^{\beta(\sqrt{1-(\frac{z}{\alpha})^2}-1)} dz} \begin{cases} e^{\beta(\sqrt{1-(\frac{z}{\alpha})^2}-1)}, & |\frac{z}{\alpha}| \leq 1 \\ 0, & \text{otherwise.} \end{cases} \tag{1.3}$$

Here, $\alpha = w_i h_i / 2$, where h_i is the grid spacing in direction i (x, y or z), w_i is the number of points to which we spread in that direction, and we've normalized the kernel so that integrating it over its support gives 1. The monopole term Δ_f represents the force exerted by particles on the fluid, while the dipole term Δ_τ relates to external torques $\boldsymbol{\tau}$ on the particles, given in [2] as

$$\boldsymbol{f}(\boldsymbol{x}) = \sum_{k=1}^{M} \boldsymbol{f}(\boldsymbol{y}_k) \Delta_f(\boldsymbol{x} - \boldsymbol{y}_k) + \frac{1}{2} \nabla \times \big(\boldsymbol{\tau}(\boldsymbol{y}_k) \Delta_\tau(\boldsymbol{x} - \boldsymbol{y}_k) \big), \tag{1.4}$$

where $\boldsymbol{y}_k$ is the position of particle k. Once we've solved for the fluid velocity $\boldsymbol{u}$, we can obtain the linear and angular velocities of the particles through

$$\boldsymbol{V}(\boldsymbol{y}_k) = \int_\Omega \boldsymbol{u}(\boldsymbol{x}) \Delta_f(\boldsymbol{x} - \boldsymbol{y}_k) d\boldsymbol{x}, \tag{1.5}$$

$$\boldsymbol{\Omega}(\boldsymbol{y}_k) = \frac{1}{2} \int_\Omega \big(\nabla \times \boldsymbol{u}(\boldsymbol{x}) \big) \Delta_\tau(\boldsymbol{x} - \boldsymbol{y}_k) d\boldsymbol{x}. \tag{1.6}$$

Note, the kernel parameters given above correspond to uniform grids. When the grid is not uniform in a given direction, we use the relation

$$\alpha = \frac{w R_h}{2 c_{f|\tau}(w)}, \tag{1.7}$$

where w and $c_{f|\tau}(w)$ are the width and dimensionless radius (of monopole or dipole) of the ES kernel found in tables listed in the report deriving the DP-wall solver (ADD HERE).

2 Describe the SpreadInterp library

2.1 How to handle various BCs

3 Describe the Python interface

3.1 A note on memory layout

Data on the grid are stored so that the fastest index is $l = 0, \cdots, \texttt{dof} - 1$, followed by $i = 0, \cdots, N_x - 1$, then $j = 0, \cdots, N_y - 1$, and with $k = 0, \cdots, N_z - 1$ the slowest. This means that the `Grid` member `fG` is such that `fG`(k,j,i,l) = `fG[l + dof * (i + Nx * (j + Ny * k))]`. The reason for this has to do with parallelization and vectorization in the underlying spreading algorithm, where it's better if x and y vary faster than z. However, this isn't necessarily true for the solve stage.

In the Python interface, when we execute

```
fG = Spread(species, grid, Nx * Ny * Nz * dof),
```

the returned variable `fG` has been marshalled into a flat `numpy` array. If it makes sense (based on the solver) to *copy* the memory layout to a new one, say in reverse order, one can use

```
fG = np.transpose(np.reshape(fG, (Nz,Ny,Nx,dof)), (3,2,1,0)).copy()
```

Then, the `fG` variable will be indexed in memory like `fG[l,i,j,k]` with k the fastest index. Note, without the `copy()` operation appended at the end, we will only get a new *view* of the memory layout, while the actual layout will remain unchanged.

4 Triply periodic solver

Suppose now that Ω is a triply periodic domain and we impose periodic boundary conditions on $\boldsymbol{u}$ and p. Taking the divergence of Eq. 1.1 and using Eq. 1.2, we find the pressure p satisfies a Poisson equation

$$\nabla^2 p = -\nabla \cdot \boldsymbol{f} \tag{4.1}$$

Then, taking the Fourier transform of Eq. 4.1 and using Fourier derivative symbols, we find

$$\hat{p} = -\frac{i\boldsymbol{k} \cdot \hat{\boldsymbol{f}}}{k^2}, \tag{4.2}$$

where $\boldsymbol{k} = \begin{pmatrix} k_x & k_y & k_z \end{pmatrix}^T$ is the vector of wave numbers in x, y and z, $k^2 = ||\boldsymbol{k}||^2$, and $\hat{\cdot}$ denotes the Fourier transform. Now, if take the Fourier transform of Eq. 1.1 and use Eq. 4.2, we find that

$$\hat{u} = \frac{1}{\eta k^2}\Big(\hat{f} + \frac{ik_x}{k^2}\big(ik_x\hat{f} + ik_y\hat{g} + ik_z\hat{h}\big)\Big), \tag{4.3}$$

$$\hat{v} = \frac{1}{\eta k^2}\Big(\hat{g} + \frac{ik_y}{k^2}\big(ik_x\hat{f} + ik_y\hat{g} + ik_z\hat{h}\big)\Big), \tag{4.4}$$

$$\hat{w} = \frac{1}{\eta k^2}\Big(\hat{h} + \frac{ik_z}{k^2}\big(ik_x\hat{f} + ik_y\hat{g} + ik_z\hat{h}\big)\Big). \tag{4.5}$$

The above equations hold for $k \neq 0$. Since the net force on the unit cell must be 0, we accept 0 as the solution for the $k = 0$ mode of the fluid velocity.

In our uniform discretization, $k_i = \frac{2\pi j}{L_i}$, $j = -\frac{N_i}{2}, \cdots, \frac{N_i}{2} - 1$, with N_i the number of points in direction $i = x, y, z$, and L_i the length.

4.1 Validation (translation-only)

Here, we will validate the efficacy of our method and implementation in computing particle velocities against Hasimoto's correction to Stokes drag law for cubic lattices of spheres [3], given by

$$\frac{F}{\eta V} = \frac{6\pi R_h}{1 - 2.8373(R_h/L) + 4.19(R_h/L)^3 - 27.4(Rh/L)^6 + \text{h.o.t}}, \tag{4.6}$$

where F is the force on the particle and V is its velocity in one direction. We place a force of $\begin{pmatrix}1 & 0 & 0\end{pmatrix}$ on a particle of radius $R_h = 1.5539h$ located randomly in the unit cell $[0, L]^3$. This R_h corresponds to ES kernel settings $w = 6, \beta = 1.714w$ with 0.02% error from translational invariance. After spreading the force on the particle to the grid with the first term in Eq. 1.4, solving Eq. 1.1 and interpolating the velocity on the particles with Eq. 1.5, we evaluate agreement with Eq. 4.6. The results, collected over several random placements of the particle and lattice lengths, are shown in the following figure.

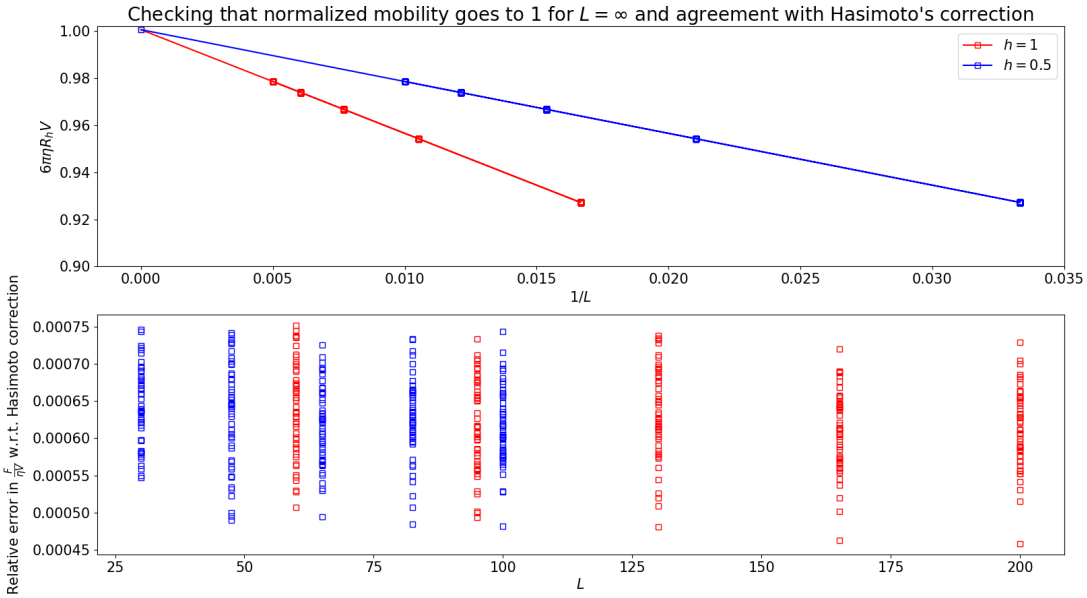

Figure 1: In the top panel, we check that the normalized mobility extrapolates to approximately 1 for unit (blue) and non-unit (red) grid spacing h as $L \to \infty$. The bottom figure assesses the relative agreement with Hasimoto's periodic correction to Stokes drag law Eq. 4.6 for unit (blue) and non-unit (red) h. We achieve around 3 digits of accuracy, and the spread in the data corresponds well to the 0.02% error due to loss in translational invariance of the ES kernel.

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, 2018.

[2] Sune Lomholt and Martin Maxey. Force-coupling method for particulate two-phase flow: Stokes flow. *Journal of Computational Physics*, 184:381–405, 01 2003.

[3] H. Hasimoto. On the periodic fundamental solutions of the stokes equations and their application to viscous flow past a cubic array of spheres. *Journal of Fluid Mechanics*, 5(2):317328, 1959.