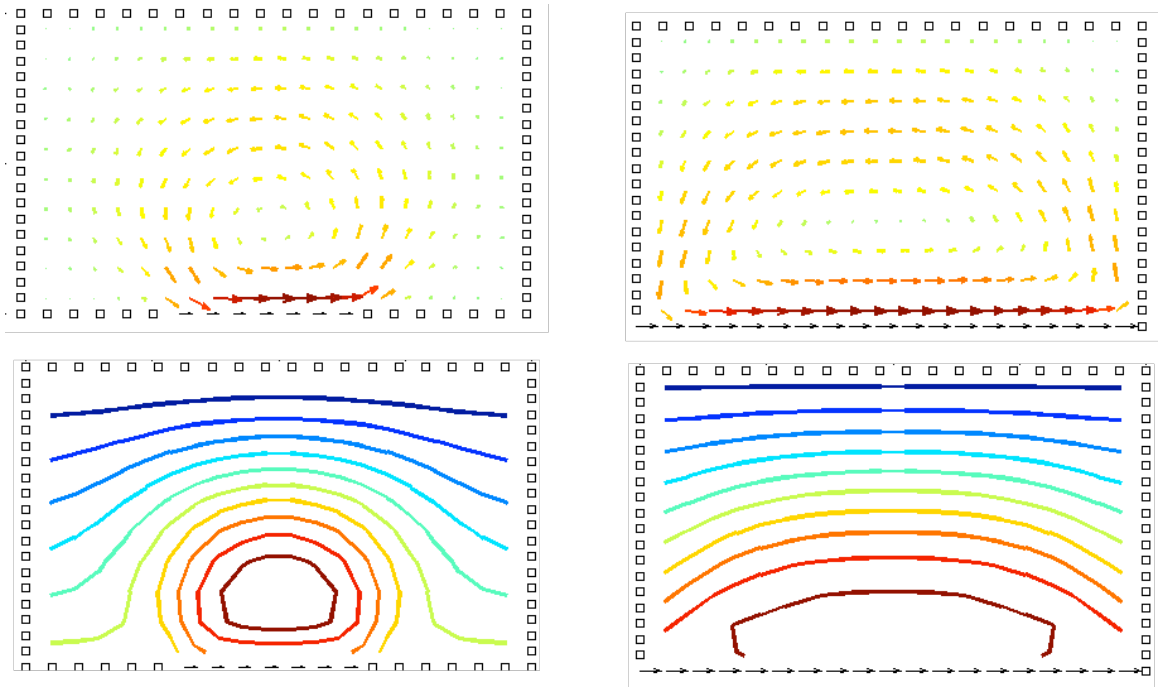
# STOKES FLOW SIMULATION IN MATLAB

BRENDAN HUANG



The simulation of microfluidic flows is of interest in numerous medical areas including microfluidic device design, the investigation of basic physiology, and biomechanical engineering. In particular, low Reynolds number flow, or Stokes flow, is a common fluidic regime that defines the dynamics of bacteria, small organisms, and specific regions within the human body. This document describes the theory and implementation of two methods of numerical fluid simulations of Stokes flow: the method of fundamental solutions (MFS) and the boundary element method (BEM).

This document is adapted from Appendix D of my PhD Thesis [1]. The methods discussed here can be found in more detail in two of Pozrikidis' texts [2, 3]. In this document, I have tried to highlight the most essential aspects of the methodology. Additionally, I have adapted some portions of code from a tutorial on boundary element methods by Kirkup and Yazdani [4]. All of these sources will provide more explicit detail on certain elements of the methods, should the reader be interested.

The layout of this document is as follows. I will first introduce the general problem of simulating Stokes flow. I will then discuss the theory and implementation of two numerical methods. The first, the method of fundamental solutions (MFS), is easy to implement but often less numerically precise. The second, the boundary element method (BEM), offers better precision but is more complicated and computationally intense to implement. The goal of this document is for users to be able to reuse as much code as possible for their own purposes.

## 1. Numerical Solutions of Navier-Stokes

The goal of our fluidic simulations will be to find numerical solutions to the Navier-Stokes equations. The incompressible Navier-Stokes equations are:

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = \mathbf{f} - \frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla^2 \mathbf{u} \tag{1}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{2}$$

As written above, the Navier-Stokes equation is actually a compact way of writing three coupled partial differential equations in addition to the incompressibility equation. From a mathematical perspective, in order to solve these coupled differential equations, we need to specify:

- The geometry of the problem, including: the domain of where the problem is to be solved, as well as the boundaries or edges of the domain.
- The boundary conditions for either $u$, $v$, and $w$, defined on the boundary (Dirichlet boundary conditions), the traction $\mathbf{f}$ on the surface (Neumann boundary conditions), or some combination of the two (mixed and Robin boundary conditions).
- The initial conditions of the problem, if the flow is non-steady state.

From a very high level perspective, many computational fluid dynamic (CFD) models solve Navier-Stokes (or other constitutive fluid equations) by dividing the domain into a mesh of volume elements and then using numerical methods to solve the resulting equations. Common techniques such as the finite difference method, finite volume method, and finite element method work by solving the equations of motion on the domain of the fluid. That is, for a volume of fluid flow, the volume is discretized up into small sub-volumes. The equations of motion are solved in these sub-volumes.

In another set of CFD schemes, special properties of Navier-Stokes are taken advantage of in order to avoid having to discretize the entire domain. The advantage is that these methods take three-dimensional mesh structures and recast them as

lower dimensionality problems, making computation less intensive. The disadvantage of these methods is that they typically work in more restricted circumstances. For example, the methods presented here only work for low Reynolds number approximations.

Here, I discuss the two methods we have implemented: the method of fundamental solutions (MFS), and the boundary element method (BEM). Before describing the methods, however, I will briefly discuss some of the mathematical background on which they rely.

## 2. Interlude: Linear Differential Equations

In the context of solving the full Navier-Stokes versus low-Reynolds number flow, the concept of linear differential equations is important. A linear differential equation is a differential equation in which separate solutions can be added together to yield a third valid solution. Typically, linear differential equations are written as:

$$Lu = f \tag{3}$$

where $L$ is a linear operator, $u$ is an unknown function, and $f$ is a known function. Thus, if we have two solutions, $u_1$, and $u_2$, we know that

$$\begin{aligned} Lu_1 &= f_1 \\ Lu_2 &= f_2 \\ Lu_1 + Lu_2 &= f_1 + f_2 \\ L(u_1 + u_2) &= (f_1 + f_2) \end{aligned}$$

and thus we see that if $L(u_1 + u_2) = Lu_1 + Lu_2$, then $u_1 + u_2$ is also a valid solution. Linear superposition is a very useful mathematical tool because we can solve for simple solutions to our differential equations, usually implicitly choosing a mathematical basis, and then reconstruct any arbitrary solution in terms of that mathematical basis. Such linear superposability forms the mathematical basis for constructive and destructive interference in Maxwell's equations or the Schrödinger equation, for example.

In addition to its important physical ramifications, linearity also has important properties for mathematical analysis. Specifically, if we consider the term $f$ from Eq 3 often termed our source function, to be an impulse $\delta(\mathbf{x} - \mathbf{x}_0)$ (where $\delta$ is the Dirac delta function), then $y$ is known as our Green's function:

$$Ly = \delta(\mathbf{x} - \mathbf{r}_0) \tag{4}$$

$$y \equiv G(\mathbf{x}, \mathbf{r}_0) \tag{5}$$

The physical interpretation of a Green's function $G(\mathbf{x}, \mathbf{x}_0)$ is as the function that specifies the resulting field or action in response to a point source. It is often known as a spatial or temporal impulse response. By summing up the response to many impulses, we can get the total resulting field from a complex distribution. For example, in electrostatics a single point charge generates an electric field given by Gauss' Law. If we want to know what the electric field is in a geometry with a distribution of charges, we merely have to place a distribution of point sources in the proper locations, and sum up all of their contributions. As will be discussed shortly, properties of Green's functions are central to both of our computational schemes, but an important underlying assumption is that the Green's function solves a linear differential equation.

The concept of linearity is important because Navier-Stokes is a non-linear differential equation. More explicitly, if we have a solution $\mathbf{u}_1$ and $\mathbf{u}_2$, $\mathbf{u_1} + \mathbf{u_2}$ is not a solution. This property arises as a result of the convective acceleration term in Eq. 1:

$$\begin{aligned} \text{Conv Accel} &\equiv (\mathbf{u} \cdot \nabla)\mathbf{u} \\ &= (\mathbf{u}_1 + \mathbf{u}_2) \cdot \nabla(\mathbf{u}_1 + \mathbf{u}_2) \\ &(\mathbf{u}_1 \cdot \nabla)\mathbf{u}_1 + (\mathbf{u}_2 \cdot \nabla)\mathbf{u}_2 + (\mathbf{u}_1 \cdot \nabla)\mathbf{u}_2 + (\mathbf{u}_2 \cdot \nabla)\mathbf{u}_1 \\ &\neq (\mathbf{u}_1 \cdot \nabla)\mathbf{u}_1 + (\mathbf{u}_2 \cdot \nabla)\mathbf{u}_2 \end{aligned}$$

In the case of low-Reynolds number flow, the convective acceleration term becomes negligible. Under the additional assumption of the absence of body forces ($\mathbf{f} = 0$), the Navier-Stokes then simplifies to the unsteady Stokes equation:

$$\frac{\partial \mathbf{u}}{\partial t} = -\frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla^2 \mathbf{u} \tag{6}$$

Finally, in the case of steady state flow ($\frac{\partial \mathbf{u}}{\partial t} = 0$), we recover the Stokes equation:

$$\nabla p = \mu \nabla^2 \mathbf{u} \tag{7}$$

We have highlight these assumptions because in our numerical simulations, we will not be solving the full Navier-Stokes equation, but rather the Stokes equation. Our methods rely heavily on Green's functions, which require linearity. Thus, it

is important to keep in mind that our simulations will only be valid for (a) Low reynolds number and (b) steady state flow.

With the mathematical background introduced, we can now proceed to our first simulation method: the method of fundamental solutions.

## 3. The Method of Fundamental Solutions

The underlying principle of the method of fundamental solutions, or MFS, is as follows. We first find the solutions to a particular differential equation ignoring of boundary conditions. (In this particular case, the fundamental solutions will be the Green's functions associated with a point force.) We then superpose those solutions in a way such that that they satisfy the boundary conditions of the problem. Because each basis function satisfies the Stokes equation, and because Stokes equation is linear, any linear combination of functions also satisfies the Stokes equation. Adding them in such a way that the boundary conditions are satisfied then completes the requirements for solving our partial differential equation.

The major advantages of MFS is that it is (a) mesh-free, (b) avoids integrating over singularities, and (c) requires no numerical integration, making it computationally low-cost.

MFS has several disadvantages, too. First, the method fundamentally requires the placement of fictitious forces outside the domain on which the solution is solved, and it is unclear what physical interpretation these forces should have. Additionally, because singular functions are not directly embedded in the boundary, the method does not estimate higher-derivatives (i.e. traction) well. Finally, the way in which we set up our equations to solve, we are often left with an ill-posed inverse problem which may be sensitive to our placement of the point forces. Given that the forces are a somewhat fictitious construct, and and their specific location is arbitrary and not dictated by the physical problem, an ideal simulation method would be insensitive to their location. The flow fields generated by MFS, however, can be sensitive to the placement of these forces.

3.0.1. *Theory of MFS.* In this section, we will describe the application of MFS, a general technique to numerically solve differential equations, towards Stokes flow. For the purposes of this explanation, we will consider the 2D case, although it will be generalizable to 3D as well.

Our goal is to solve Stokes equation, Eq 7, while satisfying specific boundary conditions for $u$ and $v$. In order to do so, we consider the basis function associated with a point source $\mathbf{f}$, which is called a *Stokeslet*:

$$-\frac{1}{\rho}\nabla p + \frac{\mu}{\rho}\nabla^2 \mathbf{u} = \mathbf{f}\delta(\mathbf{x} - \mathbf{x}_0) \tag{8}$$

where $\mathbf{x} = (x, y)$ is the spatial coordinate parameterizing the flow field, and $\mathbf{x}_0 = (x_0, y_0)$ is the spatial location of our Stokeslet point force $\mathbf{f}$. The solution of this equation tells us that if we pick a location $\mathbf{x}_0$ to place a Stokeslet, what the resulting flow field and pressure will be at any other arbitrary location $\mathbf{x}$.

The specific form of the Stokeslet, also called the Oseen-Burgers tensor is (in 2D):

$$u_i(\mathbf{x}) = \frac{1}{8\pi\mu} G_{ij}(\mathbf{x}, \mathbf{x}_0) f_j(\mathbf{x_0}) \tag{9}$$

$$G_{ij}(\mathbf{x}, \mathbf{x}_0) = -\delta_{ij} \ln r + \frac{\hat{x}_i \hat{x}_j}{r^2} \tag{10}$$

where $\hat{\mathbf{x}} = \mathbf{x} - \mathbf{x}_0$ and $r \equiv |\hat{\mathbf{x}}|$. The interpretation of $G_{ij}(\mathbf{x}, \mathbf{x}_0)$ is a tensor quantity that describes how the flow field at a location is induced by a point force located $\hat{\mathbf{x}}$ away.

Because of the linearity of Stokes equation, we can add many of these Stokeslets together. The basic idea of MFS, then, is to specify our boundary conditions at $N$ discrete locations at our boundary. We then place an equivalent number of Stokeslets in the vicinity of each of these locations. Although all Stokeslets will influence the flow at every location, the magnitude of influence that any given Stokeslet has on a location at the boundary will depend on its proximity to that location. Mathematically speaking, if we have $N$ boundary points with $d$ values of vectorial flow components, each of those $N$ Stokeslets will be coupled to every boundary point, leading to $N \times d$ coupled equations. In these equations, the magnitude of each Stokeslet unknown. We can then solve for the magnitude of each of those Stokeslets such that the boundary conditions are satisfied.

The MFS process yields a solution because (1) each of the Stokeslets is a solution to the Stokes equation, (2) because they can be added in a linear fashion, and (3) because we are setting them up in a way such that our boundary conditions are satisfied. Thus, any solution that we get out of adding the Stokeslets together satisfies our original criteria for solving the Stokes equation.

More explicitly, we start by discretizing our boundary at a set of locations $\mathbf{x}_m$. We then specify a boundary condition $u(\mathbf{x}_m)$ and $v(\mathbf{x}_m)$ at each of these locations. For each of the $N$ boundary conditions, we place a nearby Stokeslet $\mathbf{f}$ at a location $\mathbf{x}_n$. At this point in the process, we specify the location of the Stokeslet, but not the magnitude of the force $\mathbf{f} = (f, g)$ If we then calculate the flow field $u_i$ at a given boundary point $\mathbf{x}_m$ due to the contribution of all the Stokeslets, we get:

$$u_i(\mathbf{x}_m) = \sum_n G_{ij}(\mathbf{x}_m, \mathbf{x}_n) f_j(\mathbf{x}_n)$$

and written out explicitly:

$$u(\mathbf{x}_m) = \frac{1}{8\pi\mu}\sum_n f_n\left[-\ln(r_{mn}) + \frac{(x_m - x_n)^2}{r_{mn}}\right] + \frac{1}{8\pi\mu}\sum_n g_n\frac{(y_m - y_n)^2}{r_{mn}}$$

$$v(\mathbf{x}_m) = \frac{1}{8\pi\mu}\sum_n f_n\frac{(x_m - x_n)^2}{r_{mn}} + \frac{1}{8\pi\mu}\sum_n g_n\left[-\ln(r_{mn}) + \frac{(y_m - y_n)^2}{r_{mn}}\right]$$

If we then collect these equations into a large matrix, we get $2N$ equations relating $u$ and $v$ to $f$ and $g$:

$$\begin{pmatrix} u(\mathbf{x}_1) \\ \vdots \\ u(\mathbf{x}_m) \\ v(\mathbf{x}_1) \\ \vdots \\ v(\mathbf{x}_m) \end{pmatrix} = \left(\begin{array}{c|c} G_{xx}(\mathbf{x}_m, \mathbf{x}_n) & G_{xy}(\mathbf{x}_m, \mathbf{x}_n) \\ \hline G_{yx}(\mathbf{x}_m, \mathbf{x}_n) & G_{yy}(\mathbf{x}_m, \mathbf{x}_n) \end{array}\right) \begin{pmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_n) \\ g(\mathbf{x}_1) \\ \vdots \\ g(\mathbf{x}_n) \end{pmatrix}$$

or, more compactly written:

$$u_{i,m} = G_{ij,mn} f_{j,n} \tag{11}$$

$$\mathbf{u} = \underline{\underline{G}}\mathbf{f} \tag{12}$$

For clarity, we have now divided our block matrix $\underline{\underline{G}}$ into 4 submatrices $G_{xx}$, $G_{xy}$, $G_{yx}$, and $G_{yy}$ each of which connect the $x$ and $y$ components of force to the $x$ and $y$ components of flow. We have written $\underline{\underline{G}} \equiv G_{ij,mn}$ to explicitly denote that our matrix is 4 dimensional, with two dimensions of spatial location encoded by $m$ and $n$, and two dimensions of vectorial ($x$,$y$) susceptibility encoded by $i, j$. The implied summation contracts the $j$ and $n$ indices.

We can calculate the matrix $\underline{\underline{G}}$ based on our Green's function, and we specify $\mathbf{u}$ as part of our original problem formulation. Thus, solving for $\mathbf{f}$ simply comes down to inverting our matrix $\underline{\underline{G}}$

$$\mathbf{f} = \underline{\underline{G}}^{-1}\mathbf{u} \tag{13}$$

We note that this equation is a Fredholm integral of the first kind. It is well known that this inverse problem is ill-posed. Because we are solving the inverse problem, MFS may often become sensitive to small changes in where we locate our Stokeslets, a definite limitation of the technique.

To complete the method, we note that up until now, we have only considered flow on the boundary. However, our goal was to solve for flow in the domain of our problem. Once we have solved for the magnitude all of our Stokeslets, however, we can compute the flow $\mathbf{u}^{int}$ at any arbitrary location inside the domain of our

problem. This can be accomplished in exactly the same way as solving for Eq 12, except instead of calculating the inverse problem, we compute the forward equation:

$$\mathbf{u}^{int} = \underline{\underline{G}}^{int}\mathbf{f} \tag{14}$$

Of note, the number of Stokeslets need not be equal to the number of interior points. If there are $N$ Stokeslets and $L$ interior points, for example, then $\underline{\underline{G}}^{int}$ will be an $2L \times 2N$ matrix.

## 4. Matlab Scripts for MFS

The Matlab implementation of MFS is summarized in the flow diagram (Fig. 1):

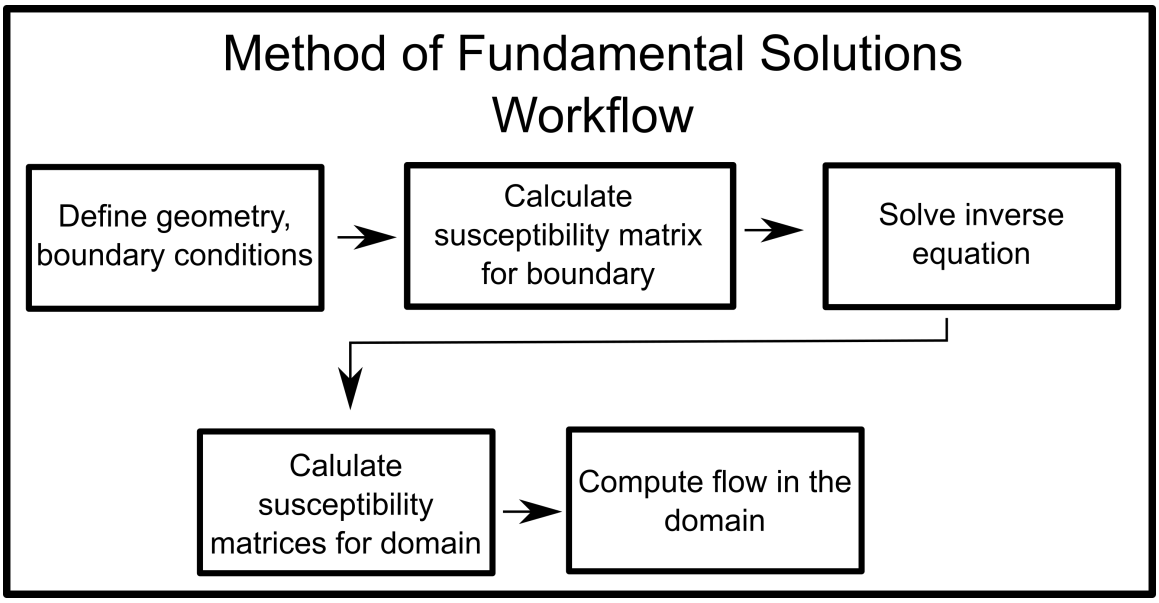



Figure 1. Workflow for method of fundamental solutions (MFS) numerical simulations.

4.0.1. *Overall execution script.* The script to run the entire MFS is given by the "doit" script. The current implementation is only in 2D. (The 3D problem becomes highly ill-conditioned with very boundary points specified.). The sample script simulates a problem similar to the lid-driven cavity, but with only a portion of the lid

(surface) moving.

| MFS Doit Scripts | |
|---|---|
| **Script** | doit_sim_MFS.m |
| **Inputs** | 2D only |
| **Outputs** | Vectorial flow field, pressure, streamfunction |

4.0.2. *Defining geometry and boundary conditions.* In the first step, the geometry of the problem is defined. We can have open boundary conditions, where the there is no enclosure of the fluid, or closed boundary conditions, where the fluid flow is constrained to a chamber. We can also specify the relative distance of the Stokeslets from the boundary. In this simulation, a Stokeslet is paired next a boundary point and placed along a line connecting the boundary point to the center of mass of the problem.

| Generating Boundary and Boundary Conditions | |
|---|---|
| **Script** | generateBC_MFS.m |
| **Inputs** | Open versus closed boundaries, size of domain, value of speed, fraction of surface that is "moving" |
| **Outputs** | *scx,scy* location of Stokeslets; *bcx, bcy, bcu, bcv* location of boundary element points and value of velocity at those locations |

4.0.3. *Calculating the susceptibility matrices.* The next step is to calculate the effect of each Stokeslet on each point in the boundary.

| Calculating elements of susceptibility matrix | |
|---|---|
| **Script** | calc_Amat_MFS_2D.m |
| **Inputs** | *scx,scy* location of Stokeslets; *bcx, bcy* location of boundary points |
| **Outputs** | *Amat* Susceptibility matrix relating each Stokeslet to each boundary point; **optional** *Apres* Susceptibility matrix of pressure due to Stokeslets; *Apsi* Susceptibility matrix of streamfunction due to Stokeslets |

4.0.4. *Solving for the flow in the interior domain.* To solve the inverse equation, we use a built in function from Matlab. Once each Stokeslet has a magnitude from solving the inverse equation, a second susceptibility matrix can be calculated to compute

the flow. Conveniently, we can also propagate the Stokeslets in order directly estimate the pressure and streamline inside the domain.

| Computing quantities in the domain | |
|---|---|
| **Script** | calcUV_MFS.m, calcP.m |
| **Inputs** | *qcoeff* strength of Stokeslets; *Amat* Susceptibility matrix relating each Stokeslet to interior points; *xpts, ypts*: number of points to calculate in the interior; *Apres, Apsi* Susceptibility matrices for pressure and streamfunction. |
| **Outputs** | *u,v* vectorial flow field ; **optional** *pres, psi* pressure field, streamfunction |

## 5. Boundary Element Method

The boundary element method (BEM), our second simulation method, has a number of similarities to the MFS. Much as in MFS, one specifies the geometry and boundary conditions of the problem and then embeds Green's functions in order to satisfy the Stokes equation on the domain. Enforcement of the boundary conditions the allows us to solves for the magnitude of the Green's functions. Once the coefficients of the Green's functions have been estimated, flow can then be calculated at any arbitrary location in the domain of the problem. The overall flow of BEM is shown in Fig.2.

The differences between the two methods are that instead of the forces being located outside the domain as in MFS, in BEM the forces are located on the boundary of the problem (hence the name *boundary element* method). Because the forces are located on the boundary, then, instead of solving Eq. 12, we solve the boundary integral (equation 38). The boundary integral equation arises when we convert our volume equation into a boundary equation by invoking the divergence theorem, as will be shown below.

Because the divergence theorem requires integrating over the surface of the boundary, we have to compute the response of elements by integrating over surface areas in BEM (as opposed to summing discrete points in MFS), a process that requires additional numerical and analytic work. In particular, because the point forces are collocated on the boundary, and because the Green's functions have singularities, the BEM requires methods to integrate over singularities.

Overall, BEM is more complicated to implement than MFS but has several advantages. First of all, there is no arbitrariness of locating point forces. The forces in BEM are located on the boundary, and instead of being fictitious, they have the direct physical interpretation of specifying the traction on the surface. Secondly, the boundary integral equation is slightly different than Eq. 12, and in certain cases, it is a Fredholm integral of the second kind. The Fredholm integral of the second kind

is not ill-posed in the same manner as the first kind. As a result, BEM simulations are often less sensitive to small numerical errors.

In this section, we will first motivate the underlying mathematical principle of BEM. We will then discuss our specific implementation of the procedure.

5.1. **Interlude: The Reciprocal Relations.** The mathematical basis of BEM lies in Green's second identity, or the reciprocal identity. In this section, we will give some underlying intuition to the procedure by considering the reciprocal identity as applied towards a simpler problem: Laplace's Equation. Laplace's equation is defined by:

$$\nabla^2 f = 0 \tag{15}$$

Let's consider two separate functions $f$ and $\phi$ that are not the same but both satisfy Laplace's equation. Specifically, let's consider the quantity $\phi\nabla^2 f$. We can write the identity:

$$\phi\nabla^2 f = \nabla\cdot(\phi\nabla f) - \nabla\phi\cdot\nabla f \tag{16}$$

$$= \nabla\phi\cdot\nabla f - \nabla\phi\cdot\nabla f + \phi\nabla^2 f \tag{17}$$

$$= \phi\nabla^2 f \tag{18}$$

and equivalently:

$$f\nabla^2\phi = \nabla\cdot(f\nabla\phi) - \nabla f\cdot\nabla\phi \tag{19}$$

and subtracting Eq. 19 from Eq. 16, while noting that $\nabla^2 f = 0$ and $\nabla^2\phi = 0$, we get:

$$\phi\nabla^2 f - f\nabla^2\phi = \nabla\cdot(\phi\nabla f - f\nabla\phi) \tag{20}$$

$$0 = \nabla\cdot(\phi\nabla f - f\nabla\phi) \tag{21}$$

If we now apply the divergence theorem to Eq. 21:

$$\int\int(\phi\nabla f - f\nabla\phi)\cdot\hat{n}\,dA = 0$$

Finally, if $\phi$ is a Green's function satisfying the equation:

$$\nabla^2\phi = f\delta(\mathbf{x}-\mathbf{x}_0) \tag{22}$$

$$\phi \equiv G(\mathbf{x},\mathbf{x}_0) \tag{23}$$

We recover the boundary integral equation:

$$\int (G\nabla f - f\nabla G)\cdot \hat{n}\, dA = \int\int f(\mathbf{x})\delta(\mathbf{x}-\mathbf{x}_0) \tag{24}$$

$$\int (G\nabla f - f\nabla G)\cdot \hat{n}\, dA = f(\mathbf{x}_0) \tag{25}$$

The interpretation of Eq 25 is as follows. Suppose we would like to construct an arbitrary function $f(\mathbf{x}_0)$, where $x_0$ is in the domain. We can do so by specifying the function $f$ along the boundary, and combining that with the knowing of how the function on the boundary is propagated to the interior of the problem via our Green's function $G$.

As an aside, one intuitive physical explanation of Eq 25 comes from the field of optics. In optics, the Huygens-Fresnel principle tells us that if we want to know the field at an arbitrary location $\mathbf{x}_0$ in space, we can propagate our wave front from a boundary. For example, in order to generate a sharp electric field at a microscope focus, we specify the electric field at a lens, and that electric field then propagates and interferes at the focus. In the most extreme case as the angular aperture of the system reaches $2\pi$, we specify an entire closed surface around the point. In that case, we can generate any arbitrary electric field in our domain by propagating the field from points on the boundary [1]. Similarly, in the boundary element method, we reconstruct the flow field at any point by propagating our Green's functions from the boundary to to the domain.

5.2. **The Reciprocal Relations in Stokes Flow.** In the previous section, we gave a simple example of Green's second identity in order to motivate its use and provide a physical interpretation. Using a similar analysis, we can derive the reciprocal relations for the Stokes equation. The goal of the reciprocal relation will be to derive the boundary integral equations, Eqs 38 and 39. Analogous to Eq. 13 in MFS, the boundary integral equation will be the equation we actually numerically solve in order to simulate Stokes flow in our domain.

We begin with a statement of Stokes flow in terms of the Cauchy stress tensor $\underline{\underline{\sigma}} \equiv \sigma_{ij}$:

$$\partial_i \sigma_{ij} = 0 \tag{26}$$

$$\sigma_{ij} = p\delta_{ij} + \mu(\partial_i u_j + \partial_j u_i) \tag{27}$$

[1] In fact, the ability to reconstruct an electric field at a point in the domain by specifying the boundary is taken advantage of in Kirchoff scalar diffraction theory. The Kirchoff integral, a construct used to numerically simulate electromagnetic diffraction, takes advantage of Eq 25. Notably, the use of Eq 25 leads to the celebrated Rayleigh-Sommerfeld diffraction integral.

As previously, if we left multiply Eq 26 by a second valid solution to Stokes equation $u_i'$, we can write Green's second identity as:

$$u_i' \partial_j \sigma_{ij} = \partial_j (u_i' \sigma_{ij}) - \sigma_{ij} \partial_j u_i' \tag{28}$$

$$= \partial_j (u_i' \sigma_{ij}) - [p\delta_{ij} + \mu(\partial_i u) j + \partial_j u_i)] \partial_j u_i' \tag{29}$$

and we can eliminate the pressure term $p$ by recalling that $\delta_{ij} \partial_j u_i' = \partial_i u_i' = 0$, where $\partial_i u_i = 0$ is a statement of incompressibility.

Writing equivalent equations for $u_i'$ and $u_i$:

$$u_i' \partial_j \sigma_{ij} = \partial_j (u_i' \sigma_{ij}) - \mu(\partial_i u_j + \partial_j u_i) \partial_j u_i' \tag{30}$$

$$u_i \partial_j \sigma_{ij}' = \partial_j (u_i \sigma_{ij}') - \mu(\partial_i u_j' + \partial_j u_i') \partial_j u_i \tag{31}$$

and subtracting the two equations as before:

$$\partial_i (u_i' \sigma_{ij} - u_i \sigma_{ij}') = 0 \tag{32}$$

$$\nabla \cdot (\mathbf{u}' \cdot \underline{\underline{\sigma}} - \mathbf{u} \cdot \underline{\underline{\sigma}}') = 0 \tag{33}$$

where now Eq 33 defines the reciprocal relation for Stokes flow. As in the case of Laplace's equation, we can associate $\mathbf{u}'$ instead with the Green's function of the Stokes equation, i.e.

$$\nabla p - \mu \nabla^2 \mathbf{u} = \mathbf{f} \delta(\mathbf{x} - \mathbf{x}_0) \tag{34}$$

$$u_i = \frac{1}{8\pi\mu} G_{ij} f_j \tag{35}$$

We also need to associate a Green's function with the stress tensor $\underline{\underline{\sigma}}'$. The stress tensor Green's function, denoted as $T_{ijk}$ is defined by the relationship:

$$\sigma_{ik} = \frac{1}{8\pi} T_{ijk} f_j \tag{36}$$

Now, plugging in our Green's functions into the Lorentz reciprocal relation, we recover:

$$\partial_i \left( \frac{1}{8\pi\mu} G_{jm} f_m \sigma_{ij} - \frac{1}{8\pi} u_i T_{imj} f_m \right) = f_m u_m \delta(\mathbf{x} - \mathbf{x}_0) \tag{37}$$

5.3. **The boundary integral equations.** Much like Eq. 25, in order to convert Eq. 37 into a more useful form, we now take the volume integral of the right side of Eq. 37. The integral will have different values in two scenarios that we must consider: when the delta function $\mathbf{f}\delta(\mathbf{x}-\mathbf{x}_0)$ is placed inside the domain, and when it is placed on the boundary.

If we consider a point $\mathbf{x}_0$ on the boundary, then we have to take the principal value $PV$ of the integral, leading to:

(38)
$$u_j(\mathbf{x}_0) = -\frac{1}{4\pi\mu}\int_D G_{ji}(\mathbf{x}_0,\mathbf{x})f_i(\mathbf{x})dS(\mathbf{x}) + \frac{1}{4\pi}\int_D^{PV} u_i(\mathbf{x})T_{ijk}(\mathbf{x}_0,\mathbf{x})n_k(\mathbf{x})dS(\mathbf{x})$$

where $dS(\mathbf{x})$ is a differential surface element, $D$ is the boundary. If we consider a point in the domain, we get:

$$u_j(\mathbf{x}_0) = -\frac{1}{8\pi\mu}\int_D G_{ji}(\mathbf{x}_0,\mathbf{x})f_i(\mathbf{x})dS(\mathbf{x}) + \frac{1}{8\pi}\int_D u_i(\mathbf{x})T_{ijk}(\mathbf{x}_0,\mathbf{x})n_k(\mathbf{x})dS(\mathbf{x}) \quad (39)$$

The implications of Eqs. 38 and 39 are as follows. Like Eq. 25, Eq. 39 states that we we can solve for our flow anywhere in the domain. In order to do so, we just need to know what the flow $u_i$ and traction $f_i$ are on the boundary, as well as how they propagate into the domain via our Green's functions $G_{ji}$ and $T_{ijk}$. More practically speaking, much as is the case with MFS, the general procedure of BEM is that we specify the velocity or the traction along the boundary, and solve for unknown coefficients in Eq. 38. Once we have solved for these unknown coefficients in Eq. 38, we plug them into Eq. 39 in order to compute the flow inside the domain.

More specifically, if we specify the velocity, then we need to solve for the strength of the Stokeslets (single layer) that are embedded in the boundary. Conversely, if we specify the traction, then we need to solve for the strength of the Stresslets (double layer) that are embedded in boundary. This is in contrast to MFS, where we only specify velocity, and only solve for the strength of Stokeslets.

In our own numerical simulations, we actually solve for a combination of both velocity and stress (mixed boundary conditions). We specify the velocity in the no-slip boundaries, while we specify either velocity or traction in the moving surface (see Ch. 8).

5.3.1. *Equations of BEM.* Like in MFS, in BEM we end up generating a system of coupled equations. In an analogous fashion, we discretize the boundary into $N$ points. If we redefine: $T_{ij} = T_{ijk}n_k$, where $n_k$ is the unit vector normal to the surface of the boundary, the resulting system can be written as:

$$\begin{pmatrix} u(\mathbf{x}_1) \\ \vdots \\ u(\mathbf{x}_m) \\ v(\mathbf{x}_1) \\ \vdots \\ v(\mathbf{x}_m) \end{pmatrix} = \left(\begin{array}{c|c} G_{xx}(\mathbf{x}_m,\mathbf{x}_n) & G_{xy}(\mathbf{x}_m,\mathbf{x}_n) \\ \hline G_{yx}(\mathbf{x}_m,\mathbf{x}_n) & G_{yy}(\mathbf{x}_m,\mathbf{x}_n) \end{array}\right) \begin{pmatrix} f(\mathbf{x}_1) \\ \vdots \\ f(\mathbf{x}_n) \\ g(\mathbf{x}_1) \\ \vdots \\ g(\mathbf{x}_n) \end{pmatrix} +$$

$$+ \left(\begin{array}{c|c} T_{xx}(\mathbf{x}_m,\mathbf{x}_n) & T_{xy}(\mathbf{x}_m,\mathbf{x}_n) \\ \hline T_{yx}(\mathbf{x}_m,\mathbf{x}_n) & T_{yy}(\mathbf{x}_m,\mathbf{x}_n) \end{array}\right) \begin{pmatrix} u(\mathbf{x}_1) \\ \vdots \\ u(\mathbf{x}_n) \\ v(\mathbf{x}_1) \\ \vdots \\ v(\mathbf{x}_n) \end{pmatrix}$$

or, more compactly written:

$$u_{i,m} = G_{ij,mn} f_{j,n} + T_{ij,mn} u_{j,n} \tag{40}$$

$$\mathbf{u} = \underline{\underline{G}}\mathbf{f} + \underline{\underline{T}}\mathbf{u} \tag{41}$$

If we specify $\mathbf{u}$ along our boundary, then we end up solving a Fredholm integral of the first kind with solution:

$$\mathbf{f} = \underline{\underline{G}}^{-1}(\mathbb{1} - \underline{\underline{T}})\mathbf{u} \tag{42}$$

On the other hand, if we specify the traction, then we solve for $\mathbf{u}$, and it becomes a Fredholm integral of the second kind with solution:

$$\mathbf{u} = (\mathbb{1} - \underline{\underline{T}})^{-1}\underline{\underline{G}}\mathbf{f} \tag{43}$$

Similarly to MFS, once we have solved for $\mathbf{u}$ and $\mathbf{f}$ on the boundary, we can then solve for the flow field on the domain:

$$\mathbf{u}^{int} = \underline{\underline{G}}^{int}\mathbf{f} + \underline{\underline{T}}^{int}\mathbf{u} \tag{44}$$

The number of points on the interior $L$ is again uncoupled from the number of boundary points $N$. $\underline{\underline{G}}^{int}$ and $\underline{\underline{T}}^{int}$ are $2L \times 2N$ matrices.

## 6. BEM Matlab Scripts

As shown in Fig 2 , there is a well-defined set of steps to simulate Stokes flow using BEM, and it is somewhat similar to MFS.

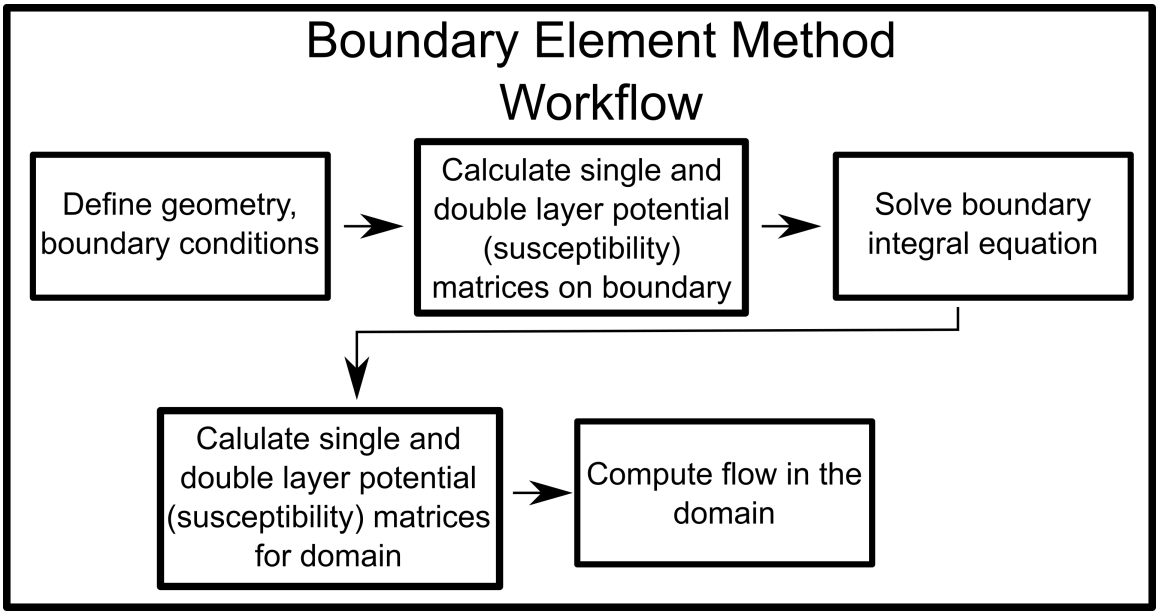



Figure 2. Workflow for boundary element method (BEM) numerical simulations.

The overall idea is to:

- Define the geometry of the problem and the boundary conditions.
- Calculate the "susceptibility" matrices relating various points on the boundary affect each other through the single and double layer potential.
- Solve the boundary integral equation.
- Calculate the "susceptibility" matrices between boundary and interior points, and calculate various quantities in the domain of the solution

6.0.1. *Execution script.* The first script is the “doit script,” the script that runs the entire simulation. There are two: one that runs a two-dimensional simulation, and one that runs a three-dimensional simulation. Both are capable of outputing the simulated vectorial flow field, pressure, and shear stress tensor. Automated computation of the streamfunction is possible but has not yet been implemented.

| BEM Doit Scripts | |
|---|---|
| **Script** | doit_sim_BEM_2D.m, doit_sim_BEM_3D.m |
| **Inputs** | Choice between 2D vs 3D |
| **Outputs** | Vectorial flow field, pressure, shear stress tensor, (streamfunction) |

6.0.2. *Defining geometry and boundary conditions.* In the next script, we define the geometry and boundary conditions. Again, our goal is to simulate a small patch of moving surface that is embedded in no-slip surface (analogous to lid-driven cavity). As described in the MFS section, we have the choice between an open or closed boundary condition. In addition, however, we also the choice between specifying the force (traction) or the velocity on the moving surface. In this implementation, we finally can choose whether to simulate Stokes flow in 2D or 3D.

| Generating Boundary and Boundary Conditions | |
|---|---|
| **Script** | generateBC_BEM_2D.m, generateBC_BEM_3D.m |
| **Inputs** | Traction vs. velocity boundary condition, open versus closed boundaries, size of domain, value of speed or traction, fraction of surface that is "moving" |
| **Outputs** | *vertpts* location of vertices of line (2D) or triangulation (3D); *elemvert*; *p* collocation points where boundary elements are calculated; *bcx, bcy, bcu, bcv* values of boundary element points, both location and value of either velocity or traction; *alpha* index specifying at each of *bcu*, *bcv*, whether it is a velocity or traction condition; *normv* normal vector of each boundary element |

6.0.3. *Generating susceptibility matrices.* In this script, we calculate the susceptibility matrix between a set of points and a set of boundary elements. In the language

of BEM, we calculate single and double layer potential.

| Generating susceptibility matrices | |
|---|---|
| **Script** | calc_Amat_BEM_2D.m, calc_Amat_BEM_3D.m |
| **Inputs** | *vertpts, elemvert* Boundary elements to calculate the influence of (number $n$); $p$ points to calculate the influence to (number $m$); *normv* normal vectors for each element used to calculate double layer potential; *p_on* specifies whether the elements can be collocated with points $p$ (not true, for example, when calculating interior points) |
| **Outputs** | *SLmat* single layer potential susceptibility matrix; *DLmat* double layer potential susceptibility matrix. Both have dimensions of $2m \times 2n$ (2D) or $3m \times 3n$ (3D) |

6.0.4. *Numerical routines for calculating susceptibility matrices.* In order to calculate the susceptibility matrices, we calculate each element of the matrix in a point-wise fashion. For non-collocalized points, Gaussian quandrature is used to numerically integrate. When a point collocalized with a boundary element, we end up integrating over a singularity. The process of integrating over the singularity often requires some analytical work. Of note, there is a standard formula used in 2D to integrate over singularities analytically. In 3D the analytical formula depends on the shape of the boundary element used. In this Matlab code, I analytically computed the integral for a right triangle.

| Calculating elements of susceptibility matrix | |
|---|---|
| **Script** | calc_SLDL_2D.m, calc_SLDL_3D.m |
| **Inputs** | *p1* point to calculate influence to; *r1, r2, [r3]* vertices of line segment (2D) or triangle (3D) that we are calculating the influence of; *normv* normal vectors of element; *p_on* is $\mathbf{x} = \mathbf{x}_0$ |
| **Outputs** | *SLxx, SLxy, SLyx, SLyy* single layer potential coupling coefficients; *DLxx, DLxy, DLyx, DLyy* double layer potential coupling coefficients. For both, there are 4 components in 2D and 9 components in 3D |
| **Subfunctions** | calcsingint.m: analytic expression for calculating singular single-layer integral in 3D; other subfunctions are called to numerically integrate by using Gaussian quadratures in 1D and 2D |

6.0.5. *Solving the boundary integral equation.* Here the boundary integral equation is solved. The important caveat is that in the case of mixed boundary conditions (i.e. specifying traction on the flow portion of the surface, but specifying zero velocity

at the no slip boundaries), we have to exchange columns in our matrices of our boundary integral equation.

| Solving the bondary integral equation | |
|---|---|
| **Script** | solve_mixedBC.m |
| **Inputs** | *BCtotal*: known velocity / traction values on boundary, all three dimensions in a single column; *alpha* index specifying which boundary element is traction versus velocity; *SLmat* single layer potential; *DLmat*: double layer potential; *visc* viscosity; *dimen*: 2D vs. 3D |
| **Outputs** | *u0* velocity values of boundary, all in a single column; *f0* traction values on boundaries, all in a single column; *bcu, bcv, [bcw]*: u0 split into proper dimensions; *fx, fy, [fz]*: f0 split into proper dimensions |

6.0.6. *Calculating quantities in the interior.* In addition to being able to compute the flow field in the interior of the domain, BEM also provides a method to calculated pressure, stress, and in principle, the streamfunction, all directly from the boundary. Much like the velocity calculation, we have to calculate a single and double layer potential for each.

| Generating pressure and stress susceptibility matrices | |
|---|---|
| **Script** | calcPSmat_BEM_2D.m, calcPSmat_BEM_3D.m |
| **Inputs** | *vertpts, elemvert* Boundary elements to calculate the influence of (number $n$); $p$ points to calculate the influence to (number $m$); *normv* normal vectors for each element; *p_on* specifies whether the elements can be collocated with points $p$. This should always be false for pressure and stress, because they will always be interior points |
| **Outputs** | *PSL*: pressure single layer matrix; *PDL*: pressure double layer matrix; *SSL*: stress single layer matrix; *SDL*: stress double layer matrix; |

| Computing quantities in the domain | |
|---|---|
| **Script** | calcUV_BEM.m, calcUV_BEM_3D.m |
| **Inputs** | *f0, u0* traction and velocity on boundaries ; *SLmat, DLmat* single and double layer potential susceptibility matrices linking elements on boundary with interior points: *viscosity*; *xpts, ypts, [zpts]*: number of points to calculate in the interior; **optional** *PSL, PDL, SSL, SDL* single and double layer potentials for pressure and stress, if would like to directly calculate these quantities on the interior |
| **Outputs** | *u,v, [w]* vectorial flow field ; **optional** *pres, stress* pressure field, shear stress tensor |

## 7. Conclusion

In this document, we have detailed two methods, MFS and BEM in the simulation of Stokes flow. We have given an overview of the theory of both methods, as well as the flow of the practical implementation in Matlab. A version of this tutorial appears in the PhD Thesis of Brendan Huang [1], the full version of which can be downloaded at: .

## References

[1] B.K. Huang. *All optical quantification of ciliary physiology.* PhD thesis, Yale University, 2015.

[2] C. Pozrikidis. *Boundary integral and singularity methods for linearized viscous flow.* Cambridge University Press, Cambridge England ; New York, 1992.

[3] C. Pozrikidis. *A practical guide to boundary element methods with the software library BEMLIB.* Chapman & Hall/CRC, Boca Raton, 2002.

[4] Stephen Kirkup, Javad Yazdani, NE Mastorakis, M Poulos, V Mladenov, Z Bojkovic, D Simian, S Kartalopoulos, A Varonides, and C Udriste. A gentle introduction to the boundary element method in matlab/freemat. In *WSEAS International Conference. Proceedings. Mathematics and Computers in Science and Engineering*, number 10. WSEAS, 2008.