Locomotion of a Three Dimensional Variable Shape Body in Inviscid Fluid

Dzhelil S. Rufat '07

May 2, 2005

Submitted to
the Department of Mechanical and Aerospace Engineering,
Princeton University,
in partial fulfillment of the requirements of Undergraduate Independent Work

Final Report

Advisor:	Clarence Rowley
Reader:	Maria Martin
MAE 340	
65 pages	

Abstract

In this work we compute the motion of a general 3D variable shape body immersed in an incompressible inviscid fluid. The body achieves locomotion by changing its shape. The relationship between the surface undulations and the motion is expressed in terms of a hydrodynamical connection $\mathcal{A}$, that relates the shape variables to the group variables describing the shape and position of the body respectively. We compute the matrix form of the connection analytically and numerically. The analytical method is not exact and essentially constitutes a boundary perturbation method that involves a series expansion for small shape variations. The numerical method is a finite difference technique, which involves discretizing the body into triangular panels, and solving for the flow around the body by assigning the appropriate boundary conditions on each panel. The two methods yield the same results in the appropriate limit of small perturbations to a spherical body, thus confirming their correctness. A check that the 3D code for an infinitely long body yields the same results as expected in the 2D case is an additional verification of the correctness of the code.

Having verified the correctness of the code, we describe forward and turning gaits of motion, and the possibility of control.

Contents

1 Introduction

The purpose of this work is to derive the motion due to the variation of the surface of a general variable body immersed in an inviscid and irrotational fluid.

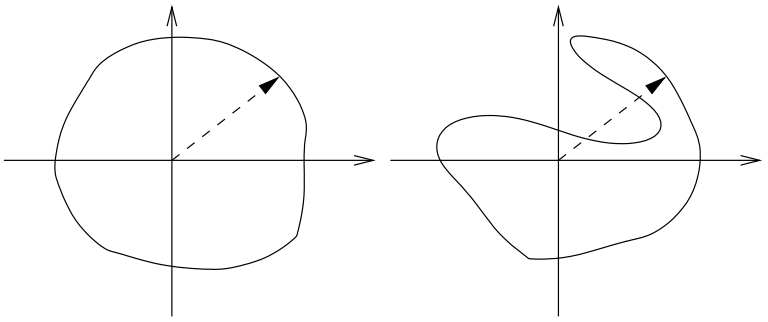
The primary motivation is the interest in underwater robotic vehicles and the search for innovative propulsion mechanisms beyond the standard propellers. Although surface variations may not lead to a very high speed motion, they provide higher maneuverability and efficiency and are the primary source of motion of aquatic animals.

An analytical method will be outlined, but due to its high complexity it will not be pursued to completion. Instead, the numerical programming approach will be emphasized. The strength of the numerical method is that it is not limited to small variations of the surface and allows for arbitrary body geometry. (The triangulation scheme in this paper relies on the parametrization of the body in spherical coordinates, which means that the surface should be star-shaped (Figure 1), however, it is easy to overcome this limitation.)

The main limitation of the model is neglecting viscous effects, which play a very important role in the motion of aquatic animals for instance. Hence, the model is mainly applicable to smooth bodies without any sharp edges, as the model relies on the body not being able to generate vorticity in the fluid.

Mason & Burdick studied a similar problem in a paper published in 1999 [3] and found an approximate analytical solution for a two dimensional amoeba-like body. The results derived by them will be extended to the three dimensional case. Additionally, Melli-Huber, Kanso and Rowley developed a model of a three link swimmer consisting of rigid ellipses, which can move relative to each other, and showed that such a system can achieve motion [2].

Any realistic robotic vehicle would have a finite number of actuators that would deform its surface. Thus it is reasonable to divide the surface variation into several modes corresponding to each actuator instead of allowing the surface to be infinitely variable.



(a) (b)

Figure 1: The only requirement imposed by the triangulation scheme on the shape in this model is that it is star-shaped, i.e. at every direction we cross the surface at a unique point (a). If any ray emanating from the center crosses the surface more than once (b) the surface is not star-shaped.

2 Background

We parameterize the surface of the body

$$F(\theta, \varphi, s_1, ..., s_n) \tag{1}$$

where $s_i(t)$ are the shape variables as a function of time that define each mode of surface deformation. θ and φ are the azimuthal and polar angles respectively.

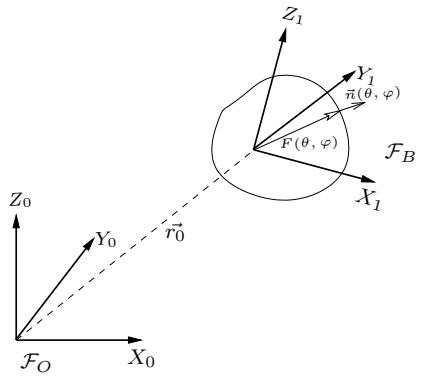


Figure 2: Body and Observer frames

2.1 Reference Frame

We fix a frame $\mathcal{F}_B$ to the body of the swimmer, and let $\mathcal{F}_O$ denote an inertial reference frame (Figure 2). The position in the inertial reference frame will be

$$[\vec{r}]_O = R[\vec{r}]_B + \vec{r_0} \tag{2}$$

where $R \in SO(3)$ is a rotation matrix describing the orientation of $\mathcal{F}_B$ with respect to $\mathcal{F}_O$, and $\vec{r_0}$ is $\mathcal{F}_B$'s origin in the inertial frame.

Given the linear velocity $\vec{\xi}(t)$ and the angular velocity $\vec{\omega}(t)$ in $\mathcal{F}_\mathcal{B}$, we would like to compute the time evolution of the motion of the body frame, i.e. the time evolution of R and $\vec{r_0}$.

Let $\mathcal{F}_B$ be instantaneously fixed. Then the velocity of a point $\vec{r}$ with respect to $\mathcal{F}_\mathcal{B}$

$$\vec{v} = \vec{\xi} + \vec{\omega} \times \vec{r} = \vec{\xi} + \Omega\vec{r}$$

where

$$\Omega = \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix}$$

is an antisymmetric matrix. Since R and $\vec{r_0}$ are fixed, the velocity in the observer frame will be

$$\frac{d}{dt}[\vec{r}]_O = R\frac{d}{dt}[\vec{r}]_B = R\vec{v}$$

Substituting $\vec{v}$

$$\frac{d}{dt}[\vec{r}]_O = R\Omega[\vec{r}]_B + R\vec{\xi} \tag{3}$$

Now we let $\mathcal{F}_\mathcal{B}$ move with the body, i.e. we keep $[\vec{r}]_B$ fixed. Then

$$\frac{d}{dt}[\vec{r}]_O = \frac{dR}{dt}[\vec{r}]_B + \frac{d\vec{r_0}}{dt} \tag{4}$$

Since $[\vec{r}]_B$ is an arbitrary point on the body, for Equation 3 and 4 to hold we require that corresponding terms be equal

$$\dot{R} = R\Omega$$

$$\dot{\vec{r_0}} = R\vec{\xi}$$

Solving the above differential equation for the matrix R is equivalent to solving the following integral equation

$$R(t) = 1 + \int_0^t dt' R(t')\Omega(t')$$

A formal solution of the integral equation can be constructed iteratively (starting with $R^0(t) = 1$)

$$R^{(i+1)}(t) = 1 + \int_0^t dt' R^{(i)}(t')\Omega(t')$$

Repeating the iteration infinitely many times we may expect to obtain the solution

$$R(t) = 1 + \sum_{n=1}^{\infty} \int_0^t dt_1 \int_0^{t_1} dt_2 ... \int_0^{t_{n-1}} dt_n \Omega(t_n)...\Omega(t_1)$$

which is known as the Dyson series. The Dyson series can be shown to be equal to

$$R(t) = Pe^{\int_0^t dt' \Omega(t')}$$

where P is the time-ordering operator. For two time dependant operators the time-ordered operator is defined by

$$P\{A(t)B(t'\} = \begin{cases} A(t)B(t'), \text{ if } t > t' \\ B(t)A(t'), \text{ if } t < t' \end{cases}$$

Thus the position of the body at any instance of time will be described by

$$R(t) = Pe^{\int_0^t \Omega(t')dt'} \quad (5)$$

$$\vec{r_0}(t) = \int_0^t R(t')\vec{\xi}(t')dt' \quad (6)$$

If Ω commutes with itself at different times P will be the identity, but this is not the case in general, so we cannot simplify the solution further. It will be more convenient to evolve the differential equations in time numerically.

Formally the location of $\mathcal{F}_B$ will be given by $g \in SE(3)$.

$$g = \begin{pmatrix} R & \vec{r_0} \\ \vec{0}^T & 1 \end{pmatrix} \qquad g^{-1}\dot{g} = \begin{pmatrix} \Omega & \vec{\xi} \\ \vec{0}^T & 0 \end{pmatrix} \quad (7)$$

$SE(3)$, the Special Euclidian group, is a continuous group under the binary operation of matrix multiplication. $SE(3)$ is a differentiable manifold, and the quantity $g^{-1}\dot{g}$ is an element of the Lie algebra of $SE(3)$, se(3). There is a 1-1 correspondence between $g^{-1}\dot{g}$ and $\mathbb{R}^6$, i.e. the components of linear $\vec{\xi}$ and angular $\vec{\omega}$ body velocity.

2.2 Potential Flow

Given our assumption of an irrotational flow $\nabla \times \vec{v} = 0$, the velocity field of the fluid can be expressed as the potential of a scalar function ϕ.

$$\vec{v} = \nabla\phi$$

The assumption of incompressibility and conservation of mass imply

$$\nabla \cdot \vec{v} = 0$$

then

$$\nabla^2\phi = 0$$

This is Laplace's equation, and it has a unique solution given a Neumann boundary condition for the normal derivative $\partial\phi/\partial n$ on the surface. The fluid motion around the body is described by an unbounded potential flow. We will assume that the fluid velocity at infinity is zero. We solve the Laplace equation with Neumann boundary conditions by separation of variables. By the Kirchhoff principle for potential flow around a rigid body, the general fluid potential can be expressed as

$$\phi = \sum_{i=1}^{3}(\xi_i\phi_i^g + \omega_i\phi_{i+3}^g) + \sum_{j=1}^{n}\phi_j^s\dot{s_j} \tag{8}$$

The terms ϕ_i^g are Kirchhoff potentials, defined with respect to a certain coordinate, to represent the potential for the body moving at unit speed along that coordinate with every other coordinate held constant.

The instantaneous velocity in the body frame of a surface point will be $\vec{\xi} + \omega \times \vec{F}(\theta, \varphi, s) + \frac{\partial}{\partial s_i}\vec{F}(\theta, \varphi, s)\dot{s_i}$, where $\vec{F} = F\hat{r}$ in body frame. At the surface the velocity

of the body and the fluid must match

$$\begin{aligned} \nabla\phi_i^g \cdot \vec{n} &= n_i & i &= 1,2,3 \\ \nabla\phi_i^g \cdot \vec{n} &= (\hat{e}_i \times \vec{F}) \cdot \vec{n} = (\vec{F} \times \vec{n})_i & i &= 4,5,6 \\ \nabla\phi_i^s \cdot \vec{n} &= \frac{\partial \vec{F}}{\partial s_i} \cdot \vec{n} & \forall s_i & \end{aligned} \tag{9}$$

2.3 Hydrodynamical Connection

To compute the Lagrangian we find total kinetic energy of the system body-fluid

$$\begin{aligned} T &= T_B + T_F \\ &= \frac{1}{2}\dot{q}^T \Lambda(q)\dot{q} + \frac{\rho_0}{2}\int_{\mathcal{F}} \|\nabla\phi\|^2 dV \end{aligned} \tag{10}$$

where ρ_0 is the fluid density and Λ is the kinetic energy metric in the absence of the fluid.

The integral over the volume of the fluid can be simplified by invoking Stokes theorem and using the identity $\nabla \cdot (\phi\nabla\phi) = \nabla\phi \cdot \nabla\phi + \phi\nabla^2\phi$.

$$\int_{\mathcal{F}} \|\nabla\phi\|^2 dV = \int_{\mathcal{F}} \nabla \cdot (\phi\nabla\phi) dV = -\int_{\Sigma} \phi(\nabla\phi \cdot n) dA \tag{11}$$

where we have taken into account that the surface integral at infinity is zero. The minus sign is introduced by the fact that the normal vectors on the boundary n point towards the fluid - the region of integration.

The potential ϕ depends only on the geometry and velocity of the submerged body. Expanding the potential in terms of the Kirchhoff potentials (Equation 8), we can express the kinetic energy of the fluid in terms of the velocity of the body.

Then

$$T = \frac{1}{2}\begin{pmatrix} \dot{g} \\ \dot{s} \end{pmatrix}^T \begin{pmatrix} \Lambda^{gg} + A^{gg} & \Lambda^{gs} + A^{gs} \\ (\Lambda^{gs})^T + (A^{gs})^T & \Lambda^{ss} + A^{ss} \end{pmatrix} \begin{pmatrix} \dot{g} \\ \dot{s} \end{pmatrix}$$

A^{gg} and A^{gs} are the added inertia terms due to effect of the fluid, calculated by substituting the Kirchhoff potentials (Equation 8) in the integral for the fluid kinetic energy

(Equation 11).

$$A^{gg}_{ij}(s) = -\frac{\rho_0}{2}\int_\Sigma (\phi^g_i(\nabla\phi^g_j \cdot n) + \phi^g_j(\nabla\phi^g_i \cdot n))dS$$
$$A^{gs}_{ij}(s) = -\frac{\rho_0}{2}\int_\Sigma (\phi^g_i(\nabla\phi^s_j \cdot n) + \phi^s_j(\nabla\phi^g_i \cdot n))dS \tag{12}$$

We can derive the equations governing the mechanics of the system from the Euler-Lagrange equations. We will seek to find the connection between the shape variables s_i and the group variables g.

$$g^{-1}\dot{g} = -\mathcal{A}(s)\dot{s} \tag{13}$$

Our aim in this paper is to solve Equation 13. Numerically, we solve for g by evolving in time through a Runge-Kutta scheme. To find an explicit form of $\mathcal{A}(s)$ at each time step one needs to employ the methods of geometric mechanics. Mason and Burdick [3] consider this problem in detail and conclude that in the case of a fluid-body system with zero momentum the connection has the following form:

$$\mathcal{A}(s) = I_d^{-1}(s)(I_d A_d)(s) \tag{14}$$

where

$$I_d(s) = \Lambda^{gg}(s) + A^{gg}(s) \tag{15}$$
$$(I_d A_d)(s) = \Lambda^{gs}(s) + A^{gs}(s) \tag{16}$$

We will compute $\mathcal{A}(s)$ analytically and numerically in the next sections. Although we will touch upon the analytical means briefly in the next section by considering the method of boundary perturbation, we will mainly emphasize the numerical method. The numerical method does not suffer from the shortcoming of the analytical method of being applicable only to small amplitude motion.

In Section 3 we describe the Boundary Perturbation method to solve for the flow around the body. Then in Section 4 we numerically compute the intrinsic inertia matrix Λ and the added inertia matrix A, and obtain a numerical algorithm to solve for the connection $\mathcal{A}$. In Section 5 we consider an examples where we compare the results of the two methods in the previous sections.

3 Boundary Perturbation Method

We define the radius-vector (the shape) as the sum of two components - an unperturbed part and small perturbation.

$$F(\theta, \varphi, s_1, ..., s_n) = F_0(\theta, \varphi) + \epsilon F_1(\theta, \varphi, s_1, ..., s_n), \tag{17}$$

where

$$F_1 = \sum_{i=1}^{n} s_i(t)\sigma_i(\theta, \varphi), \tag{18}$$

s_i and σ_i are the temporal and the spacial components, that describe the different modes of surface variation corresponding to actuator mode i. F_0 is the unperturbed surface and $F_0 = \text{const.}$ corresponds to a sphere. The requirement for small perturbations is satisfied by letting $\epsilon \ll 1$.

The fluid surrounding the body is perfectly irrotational . We can determine the potential from applying the boundary conditions on the surface and from the requirement that $u = \nabla\phi$ approaches zero at infinity.

In the case of spherical symmetry the solution to the Laplace equation is

$$\phi(r, \varphi, \theta) = \sum_{k=0}^{\infty} \big(A_k r^k + \frac{B_k}{r^{k+1}}\big) P_k(\varphi, \theta) \tag{19}$$

where $P_k(\varphi, \theta)$ is the angular dependence [1] We must take $A_l = 0$ since the first term would cause the velocity to tend to infinity for infinitely large r. Then solution acquires the simpler form

$$\phi(r, \varphi, \theta) = \sum_{k=0}^{\infty} a_k \frac{P_k(\varphi, \theta)}{r^{k+1}} \equiv \sum_{k=0}^{\infty} a_k \Phi_k \tag{20}$$

Given a solution matching the unperturbed boundary we need to find a solution matching the perturbed boundary conditions. We will use a method very similar to perturbation theory in Quantum Mechanics. It will be different in the fact that the perturbation will be on the boundary and not on the operator. Let us expand the potential in a

[1] In the general case P_k will be the spherical harmonics, but in our case we will have azimuthal symmetry so it will be the Legendre polynomials in terms of $\cos(\theta)$

power series in terms of ϵ .

$$\phi = \phi^{(0)} + \epsilon\phi^{(1)} + \epsilon^2\phi^{(2)} + ... + \epsilon^i\phi^{(i)} + ... \tag{21}$$

where

$$\phi^{(i)} = \sum_{k=0}^{\infty} a_k{}^{(i)}\Phi_k \tag{22}$$

Our purpose is to express the coefficients $a_k{}^{(i)}$ in terms of the perturbation.
The potential should satisfy the new boundary condition on the perturbed surface Σ

$$\nabla\phi \cdot \vec{n}|_{\Sigma} = v_n \tag{23}$$

We expand both sides of Equation 23 in terms of ϵ and equate terms of equal power.

First we need to compute the normal $\vec{n}$ at each point on the surface. The two tangential components on the surface will be

$$\vec{t_1} = \frac{\partial}{\partial\varphi}(F\hat{r}) = \begin{bmatrix} \frac{\partial F}{\partial\varphi} \\ F \\ 0 \end{bmatrix} \tag{24}$$

$$\vec{t_2} = \frac{\partial}{\partial\theta}(F\hat{r}) = \begin{bmatrix} \sin(\varphi)\frac{\partial F}{\partial\theta} \\ 0 \\ F \end{bmatrix} \tag{25}$$

The normal component will then be their cross product.

$$\vec{n} = \vec{t_1} \times \vec{t_2} = \begin{bmatrix} F^2 \\ -F\frac{\partial F}{\partial\varphi} \\ -F\sin(\varphi)\frac{\partial F}{\partial\theta} \end{bmatrix} \tag{26}$$

Note that we need not normalize the above expression since it occurs on both sides of the boundary condition equation (Equation 23). Since $F = F_0 + \epsilon F_1$ and since $F_0 = \text{const.}$

$$\vec{n} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + 2\epsilon \begin{bmatrix} \frac{F_1}{F_0} \\ -\partial_\varphi\frac{F_1}{F_0} \\ -\sin(\varphi)\partial_\theta\frac{F_1}{F_0} \end{bmatrix} + \epsilon^2 \begin{bmatrix} (\frac{F_1}{F_0})^2 \\ -\frac{F_1}{F_0}\partial_\varphi\frac{F_1}{F_0} \\ -\sin(\varphi)\frac{F_1}{F_0}\partial_\theta\frac{F_1}{F_0} \end{bmatrix} = \vec{n}^{(0)} + \epsilon\vec{n}^{(1)} + \epsilon^2\vec{n}^{(2)} \tag{27}$$

where we have divided by F_0^2 to non-dimensionalise $\vec{n}$.

Next we need to expand the gradient of the potential in terms of ϵ.

$$\nabla\phi^{(i)}|_\Sigma = \sum_{k=0}^{\infty} a_k{}^{(i)} \nabla \frac{P_k(\varphi,\theta)}{r^{k+1}}|_{r=F}$$

$$\nabla\phi^{(i)}|_\Sigma = \sum_{k=0}^{\infty} a_k{}^{(i)} \frac{1}{F^{k+2}} \vec{\psi_k}(\varphi,\theta)$$

where

$$\vec{\psi_k} = \begin{bmatrix} -(k+1)P_k \\ \frac{\partial P_k}{\partial \varphi} \\ \frac{1}{\sin(\varphi)} \frac{\partial P_k}{\partial \theta} \end{bmatrix} \tag{28}$$

We may expand $\frac{1}{F^{k+2}} = \frac{1}{(F_0+\epsilon F_1)^{k+2}}$ to obtain

$$\frac{1}{F^{k+2}} = \sum_{j=0}^{\infty} \epsilon^j f_k{}^{(j)} \tag{29}$$

where

$$f_k{}^{(j)} = \frac{1}{F_0^{k+2}} \frac{(k+1+j)!}{j!(k+1)!} (-\frac{F_1}{F_0})^j$$

Then

$$\nabla\phi^{(i)}|_\Sigma = \sum_{k,j=0}^{\infty} a_k{}^{(i)} \vec{\psi_k} f_k{}^{(j)} \epsilon^j$$

Now taking into consideration the fact that

$$\nabla\phi|_\Sigma = \nabla \left(\sum_{i=0}^{\infty} \epsilon^i \phi^{(i)} \right) |_\Sigma = \sum_{i=0}^{\infty} \epsilon^i \nabla\phi^{(i)}|_\Sigma$$

we obtain the power series expansion of the gradient in terms of ϵ

$$\nabla\phi|_\Sigma = \sum_{k=0}^{\infty} \vec{\psi_k} \sum_{i,j=0}^{\infty} a_k{}^{(i)} f_k{}^{(j)} \epsilon^{i+j}$$

Dotting the gradient into the normal vector on the surface

$$\nabla\phi|_\Sigma \cdot (\vec{n}^{(0)} + \epsilon\vec{n}^{(1)} + \epsilon^2\vec{n}^{(2)}) = v^{(0)} + \epsilon v^{(1)} + \epsilon^2 v^{(2)} + ... \tag{30}$$

Equating terms of equal order in ϵ and using Einstein's summation convention for repeated indices

$$\begin{aligned} &a_k^{(0)} u_k^{(0)} = v^{(0)} \\ &a_k^{(1)} u_k^{(0)} + a_k^{(0)} u_k^{(1)} = v^{(1)} \\ &a_k^{(2)} u_k^{(0)} + a_k^{(1)} u_k^{(1)} + a_k^{(0)} u_k^{(2)} = v^{(2)} \\ &a_k^{(3)} u_k^{(0)} + ... \end{aligned}$$

where

$$\begin{aligned} u_k^{(0)} &= \vec{\psi}_k \cdot \vec{n}^{(0)} {f_k}^{(0)} \\ u_k^{(1)} &= \vec{\psi}_k \cdot (\vec{n}^{(0)} {f_k}^{(1)} + \vec{n}^{(1)} {f_k}^{(0)}) \\ u_k^{(2)} &= \vec{\psi}_k \cdot (\vec{n}^{(0)} {f_k}^{(2)} + \vec{n}^{(1)} {f_k}^{(1)} + \vec{n}^{(2)} {f_k}^{(0)}) \\ u_k^{(3)} &= ... \end{aligned} \tag{31}$$

Thus

$$\begin{aligned} a_k^{(0)} u_k^{(0)} &= v^{(0)} \\ a_k^{(1)} u_k^{(0)} &= v^{(1)} - a_k^{(0)} u_k^{(1)} \\ a_k^{(2)} u_k^{(0)} &= v^{(2)} - a_k^{(1)} u_k^{(1)} - a_k^{(0)} u_k^{(2)} \\ a_k^{(3)} u_k^{(0)} &= ... \end{aligned} \tag{32}$$

and we may obtain the coefficients $a_k^{(i)}$ by expanding the right sides in terms of the complete basis of angular functions $u_k^{(0)}(\varphi, \theta)$. Note that each successive approximation $a_k^{(i)}$will depend on the lower order approximations $(a_k^{(i-1)}, a_k^{(i-2)}, \cdots, a_k^{(0)})$. The task of finding these coefficients will be additionally complicated by the fact that in general the set of functions $u_k^{(0)}$ will not be orthogonal so we will not be able to isolate each component by taking a dot product. Instead we will have to solve a system of linear equations.

The problem may be simplified by expanding $u_k^{(0)}$ in terms of a complete basis of ortho-normal functions $\{e_l(\varphi, \theta)\}$.

$$u_k^{(0)} = B_{kl} e_l$$

where B_{kl} are the expansion coefficients, which can be easily found by taking the dot product with e_l of both sides of the equation and exploiting the orthogonality of the functions $(e_i \cdot e_j = \delta_{ij})$. Then

$$B_{kl} = e_l \cdot u_k^{(0)}$$

The dot product, depending on the type of orthogonal functions, is defined as a weighted integral over a unit sphere (the domain of $[\varphi, \theta]$). Substituting $u_k^{(0)} = B_{kl} e_l$ in Equation 32 we obtain

$$a_k^{(i)} B_{kl} e_l = \rho^{(i)}$$

For convenience we have renamed the right sides of Equation 32 to $\rho^{(i)}$. Dotting both sides with e_l

$$B_{kl}a_k^{(i)} = e_l \cdot \rho^{(i)}$$

This is a system of linear equations with constant known coefficients B_{kl} that needs to be solved for the unknown $a_k^{(i)}$. This task may seem to be prohibitive at first since the function space spanned by e_l is infinite dimensional, so we will have an infinite number of equations - one for each value of the index l. The problem can be significantly simplified if we make the following assumptions.

1. Assume $B_{kl} = 0$ if $l > k$, i.e. B_{kl} is a lower triangular matrix with all super-diagonal entries equal to zero. This means, by the definition $B_{kl} = e_l \cdot u_k^{(0)}$, that the basis functions $u_k^{(0)}$ is a linear combination only of the first k members of the orthonormal basis $\{e_l\}$. As an example, the Legendre polynomials $P_k(\cos(\theta))$ satisfy this condition in terms of the basis $\{\cos^l(\theta)\}$ or the basis $\{\cos(l\theta)\}$.

2. Assume that for every finite i there exists $\lambda^{(i)}$ such that for any $l > \lambda^{(i)}$, $e_l \cdot \rho^{(i)}$ vanishes.This means that $\rho^{(i)}$ is a linear combination of at most $\lambda^{(i)}$ of functions e_l. This will be true in our case since we express the surface undulation in terms of a finite number of surface variation modes [2]. If the body's surface perturbation is not divisible into a finite number of modes, the current assumption would be violated, and then we will have to deal with an infinite system of equations.

Based on these two assumptions, all terms on the left hand side with B_{kl}, for which $k < l$, and all terms on the right hand side with $l > \lambda$ will vanish. This simplification allows us to construct the solution iteratively starting from a_λ until a_0 by only considering

[2]The existence of a maximal $\lambda^{(i)}$ for $\forall i$ follows by induction on i For the base case $i = 0$ the statement will be true since $\rho^{(0)} = v^{(0)}$ and $v^{(0)}$ consist of finitely many linear combinations of the basis functions. For the induction step we note that $\rho^{(i)}$ depends on lower order terms $\rho^{(i')}$ with $i' < i$. Then if the statement holds for $i' < i$ it will hold for i too. By induction we prove that for any i $\rho^{(i)}$ will be a linear combination of a finite number of basis functions, meaning there should exist a maximal $l_{max} = \lambda^{(i)}$ for $\forall i$. QED

the non-vanishing terms.

$$\begin{aligned} a_\lambda &= \frac{e_\lambda \cdot \rho}{B_{\lambda\lambda}} \\ a_{\lambda-1} &= \frac{e_{\lambda-1} \cdot \rho - B_{\lambda,\lambda-1} a_\lambda}{B_{\lambda-1,\lambda-1}} \\ &\vdots \\ a_1 &= \frac{e_1 \cdot \rho - B_{21} a_2 - \cdots - B_{\lambda 1} a_\lambda}{B_{11}} \\ a_0 &= \frac{e_0 \cdot \rho - B_{10} a_1 - \cdots - B_{\lambda 0} a_\lambda}{B_{00}} \end{aligned} \tag{33}$$

where we have left out the superscript (i) on a, ρ and λ for conciseness. The explicit form of the coefficients in Equation 33 completes our derivation of the potential around an undulating body.

$$\phi^{(i)} = \sum_{k=0}^{\lambda^{(i)}} a_k^{(i)} \Phi_k \tag{34}$$

where Φ_k are the known eigen-solutions to the Laplace equation.

Essentially, what we have described so far, is a method that will output the coefficients $a_k^{(i)}$ given the boundary conditions for the normal velocity v. which will allow us to construct the Kirchhoff potentials and the added inertia terms.

The perturbation method permits us to obtain the potential function around any body in a fluid to the desired order of accuracy.The calculations for higher order perturbations are quite tedious and we will go no higher than first order perturbation theory in this paper.

Having evaluated the Kirchhoff potential functions, it is possible in principle to calculate the added mass terms, the hydrodynamical connection and the motion of the body as a whole. However, we will go no further than computing the potentials, since deriving expressions for the added inertias involves integration over the irregular surface of the body and our expressions have already become quite cumbersome and heavy. Although we will abandon the analytical method at this point, later it will play a very important role in validating our numerical algorithm in Section 5.

4 Numerical Method

The perturbation approach allowed us to compute the potentials approximately. Calculating the motion of the body requires that we evaluate the added and intrinsic inertia terms at different time steps, and that we use these to evaluate the connection matrix.To compute all these quantities numerically, first of all we will divide the body surface into finite elements. To keep things as simple as possible we chose the panels to be triangular.

Next we will compute the added inertia matrix describing the effect of the fluid. We compute the motion of the fluid around the body by assigning source-density distribution to the panels, which would lead to a flow equivalent to that obtained by satisfying the Neumann boundary condition.

Finally, we will compute the intrinsic inertia matrix for an arbitrarily shaped body by subdividing the body into small volume elements and summing the inertia matrices of all the elements. We obtain each element by connecting each of the panels on the surface to the origin, from which we obtain volume elements with 4 triangular faces, i.e. tetrahedrons. To evaluate the sum we need to compute the intrinsic inertia for each individual tetrahedron.

4.1 Triangulation

This section deals with the division of a body described in spherical coordinates into discrete elements. We will seek to divide the body into relatively uniform triangles.

It is easy to see that the problem will reduce to the discretization of a unit sphere, since once we discretize a unit sphere we can extend the solid angle subtended by each panel to any arbitrarily shaped surface defined in polar coordinates.

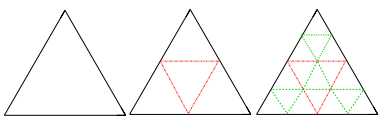


Figure 3: Recursive Division of triangles

As a first step we inscribe a regular Platonic solid with triangular faces into a unit sphere. There are only three regular convex Platonic figures with triangular faces: a tetrahedron (4 faces), an octahedron (8 face) and an icosahedron (20 faces). The inscribed Platonic figure provides the basic triangulation, then in order to refine the triangulation further we divide each triangular face into four smaller triangles by taking the midpoints of each side and interconnecting them (Figure 3). Repeating this recursively yields higher and higher depths of triangulation and precision. At each step of refinement s the number of triangles will grow exponentially (4^s) and we may not be able to afford too high refinements since this becomes computationally expensive. In planar geometry when we divide an equilateral triangle in this way we obtain four identical triangles. Yet, this will not be true on a spherical surface and the triangle will not be identical. The icosahedron yields smaller triangles and a more equal subsequent division, since locally the sphere can be approximated by a plane. Therefore, we will use an icosahedron as a basis triangulation in our program. No Platonic figure has more triangular faces than the icosahedron, so it is impossible to obtain a more uniform triangulation by starting from a different figure.

Refer to Figure 4 for a comparison of triangulation starting from an octahedron and an icosahedron.

In the case of a non-spherical body the distance from the center will lead to another source of non-uniformity among the triangle. All triangles will subtend roughly the same solid angles, and their area will be proportional to the distance from the center. For small perturbations all triangles will be roughly the same distance from the center and the variation in size will be negligible.

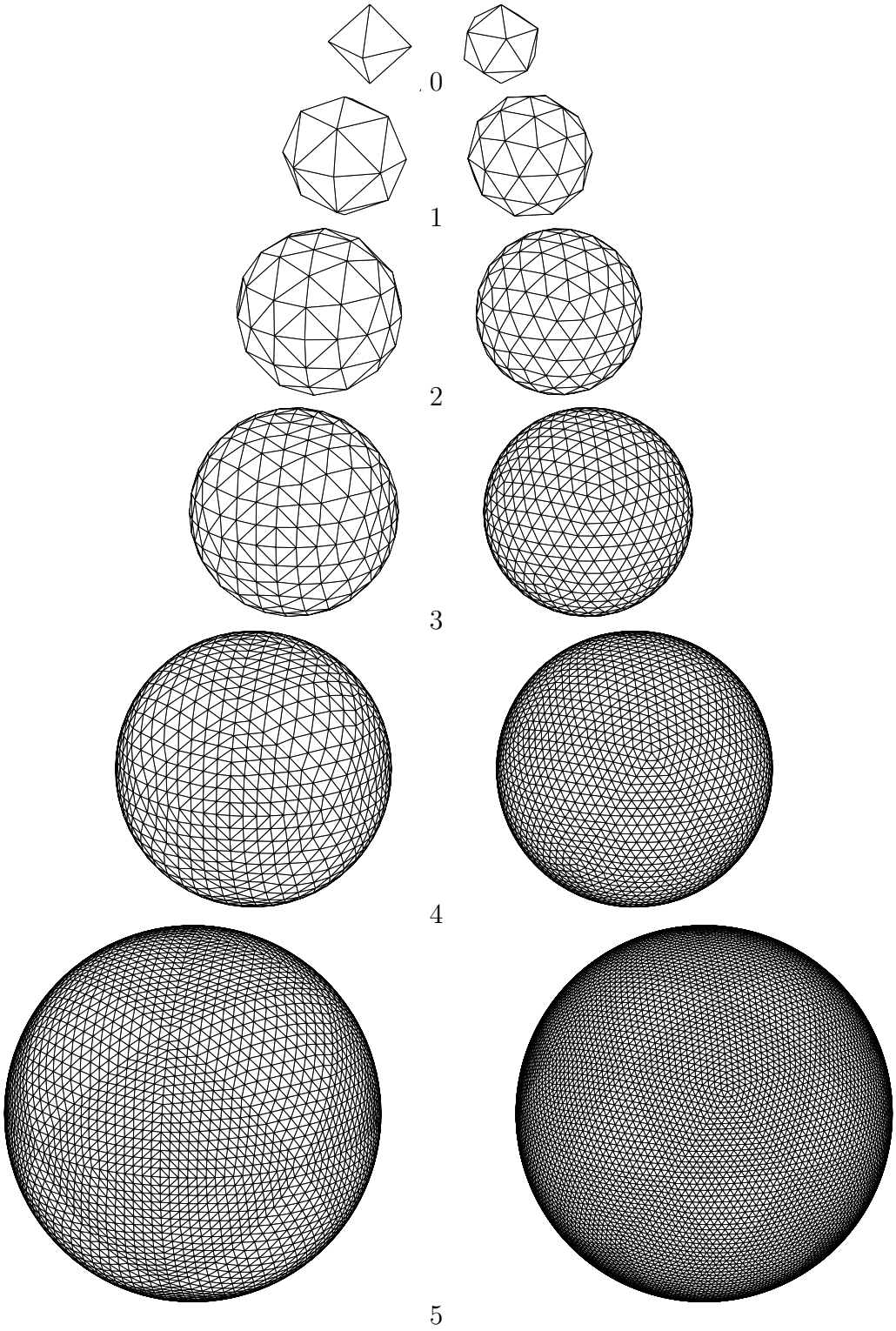


Figure 4: 0,1,2,3,4,5 order triangulation starting from a octahedron and icosahedron respectively

4.2 Fluid Inertia

In this section we compute the contribution of the fluid to the connection. The added inertia matrix A reflects the increase in inertness due to the presence of the fluid. It is related to the total kinetic energy of the ambient flow by

$$T_{\mathcal{F}} = \frac{1}{2} \begin{pmatrix} \dot{g} \\ \dot{s} \end{pmatrix}^T \begin{pmatrix} A^{gg} & A^{gs} \\ (A^{gs})^T & A^{ss} \end{pmatrix} \begin{pmatrix} \dot{g} \\ \dot{s} \end{pmatrix} \tag{35}$$

To compute the added inertia terms we need to know how the flow behaves when the surface varies. The added inertia terms depend on the Kirchhoff potentials corresponding to each group or shape variable (Equation 12 gives the explicit form). To compute these terms we solve for the ambient flow described in terms of a potential function.

The exact solution of the problem of potential flow around an arbitrarily shaped body can be approached in a variety of ways. One method is the boundary perturbation theory which gives an approximate solution to the problem for small perturbations. Yet, if we are interested in solutions for more complex situations we must eventually resort to numerical methods. Since we are aiming to solve Laplace's equation we use a finite-difference approximation over the fluid volume. This brute force approach would be too inefficient though. Instead, the Laplace equation may be reduced to an integral equation for a source-density distribution on the surface of the body (Hess and Smith [1]). Assigning source density distribution on the surface reduces the problem domain from a 3D domain to a 2D one.

Given a source-density distribution $\sigma(p')$ at point p' we may express the potential at any point P as

$$\phi(P) = \oiint_{\partial\mathcal{B}} \frac{\sigma(p')}{r(P,p')} dA' \tag{36}$$

where primed quantities indicate integration over the source surface $\partial\mathcal{B}$ and $r(P,p')$ represents the distance between the two points. The objective is to compute a source distribution on the surface that fulfills the Neumann boundary conditions. To that purpose we apply the Neumann boundary condition to Equation 36 to obtain an integral

equation involving the unknown source distribution $\sigma(p), p \in \partial\mathcal{B}$

$$-2\pi\sigma(p) + \iint_{\partial\mathcal{B}} \frac{\partial}{\partial n}\Big(\frac{1}{r(p,p')}\Big)\sigma(p')dA' = v_n(p) \tag{37}$$

known as the Fredholm integral equation [3] of the second kind over the boundary surface $\partial\mathcal{B}$ for which the unknown function $\sigma(p)$ appears both inside and outside the integral.

We solve the integral equation numerically by dividing the surface into small panels and we adjust the source density on each element in such a way that the Neumann boundary condition is satisfied at the center of each panel.

Consider two panels i and j on the surface. Assign unit source-density to each panel. The potential and hence the velocity induced on any point can be calculated using Equation 36. Call the normal velocity induced by panel i on the midpoint of panel j A_{ij}. We seek to adjust the source-density distribution in such a way that the normal velocity of the fluid is equal to the normal velocity of the body. If the unknown source-density on panel i is not 1 but σ_i the velocity induced on the center of panel j will scale proportionally from $A_{ij}.1$ to $A_{ij}\sigma_i$. The sum of all velocities induced at the center of panel j must match the normal component of the body velocity v_{nj} at that point , i.e.

$$\begin{aligned} v_{nj} &= A_{ij}\sigma_i \\ v_n &= A\sigma \end{aligned} \tag{38}$$

Hence the source-density on each panel must be

$$\sigma = A^{-1}v_n \tag{39}$$

As the number of panels becomes large the discrete solution σ_i will tend to the actual continuous solution of Equation 37.

The numerical method requires inverting a matrix, which takes quadratic time in the dimension of the matrix. As the number of panels becomes large this will be the main bottleneck, and the most time demanding part of the algorithm.

[3] Note that the integral appears with inverted signs in [1] since they define the velocity as the *negative* gradient of the potential

To compute the added inertia matrices we will make use of the numerical solution for the source-density distribution (Equation 39). Let Φ_{ij} describe the potential induced by panel i on panel j and let v_{nj} describe the boundary velocity at the center of panel j. Then the added inertia relies on computing terms of the form (see Equation 12)

$$\int_{\Sigma} \phi(\nabla\phi \cdot n)dS \approx \sum_{i,j=1}^{N} \Phi_{ij} v_{nj} \Delta S_j \tag{40}$$

where ΔS_j is the area of panel j and v_{nj} is the normal velocity at the center of panel j (boundary condition). By appropriately adjusting the normal components of velocity for each case we may compute the relevant added inertias terms.

Equation 40 relies on computing the potential Φ_{ij} induced by panel i on the center of panel j. Next we will show how to compute the potential Φ_{ij} and velocity V_{ij} induced by each panel on every other.

The potential induced by a triangle with uniform unit source density is

$$\phi = \iint_{\Delta} \frac{dA'}{r} \tag{41}$$

where dA is an area element of the triangle $\triangle$. The primed quantities imply integration over the source. For convenience we will switch to a frame of reference in which the triangle lies in the xy-plane and the point P at which we compute the influence will have coordinates $\vec{r} = (x, y, z)$ (Figure 5).

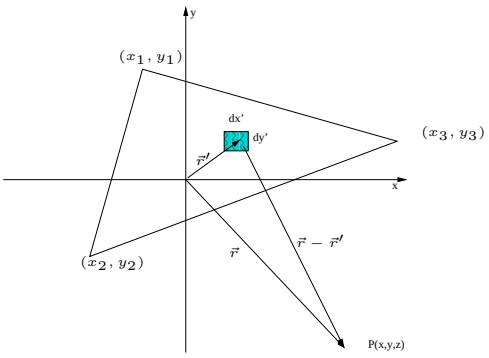


Figure 5: A plane triangle lying in the xy-plane.

Hess [1] solves the integral in Equation 41 by switching to cylindrical coordinates with an origin $(x, y, 0)$, i.e. the foot of the perpendicular from P to the plane of the

triangle (see Figure 6). In this frame the integral reduces to

$$\phi = \oint_{\partial\Delta} \int_0^R \frac{R\,dR\,d\theta}{\sqrt{R^2+z^2}} \tag{42}$$

where $R^2 = x^2 + y^2$ and using the fact that

$$r^2 = R^2 + z^2 \qquad dr = \frac{R\,dR}{\sqrt{R^2+z^2}}$$

the integral is further reduced to the form

$$\phi = \oint_{\partial\Delta} r\,d\theta - |z|\,\Delta\theta \tag{43}$$

Note that

$$\Delta\theta = \begin{cases} 0, & \text{outside}\triangle \\ 2\pi, & \text{inside}\triangle \end{cases}$$

It seems as if the term is discontinuous as we cross the side of the triangle, but as Hess argues, this will be matched by an equal and opposite discontinuity of the first term in Equation 43 related to the sign change of $d\theta$ and hence overall the potential will be continuous.

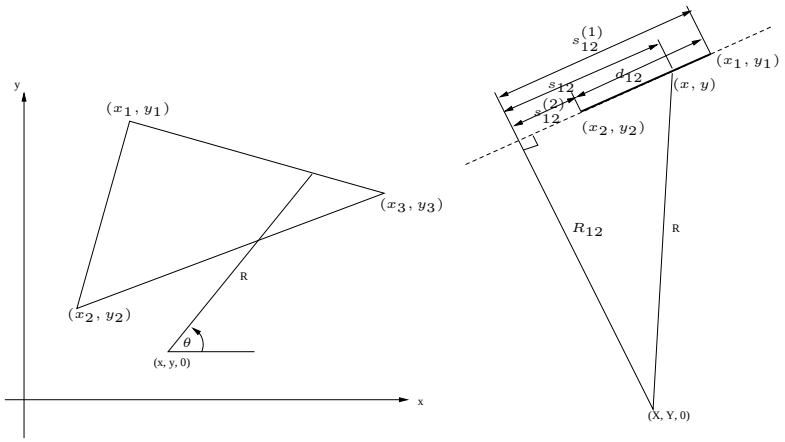


Figure 6: Introduction of cylindrical coordinates.

Computing the potential from Equation 43 involves the following quantities, all of which are labelled on Figure 6.

$$\begin{aligned} d_{12} &= \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2} \\ R_{12} &= -(x-x_1)S_{12} + (y-y_1)C_{12} \end{aligned}$$

d_{12} is the distance between nodes 1 and 2, and R_{12} is a signed perpendicular distance to the side 12.

Now define

$$\begin{array}{ll} C_{12} = \frac{x_2 - x_1}{d_{12}} & S_{12} = \frac{y_2 - y_1}{d_{12}} \\ s_{12}^{(1)} = (x_1 - x)C_{12} + (y_1 - y)S_{12} & s_{12}^{(2)} = (x_2 - x)C_{12} + (y_2 - y)S_{12} \\ r_1 = \sqrt{(x - x_1)^2 + (y - y_1)^2 + z^2} & r_2 = \sqrt{(x - x_2)^2 + (y - y_2)^2 + z^2} \end{array}$$

and the final solution for the integral can be expressed using the following quantities

$$\begin{aligned} R_{12} &= (x_1 - x)S_{12} - (y_1 - y)C_{12} \\ Q_{12} &= \ln\Big(\frac{r_1 + r_2 + d_{12}}{r_1 + r_2 - d_{12}}\Big) \\ J_{12} &= \tan^{-1}\Big(\frac{R_{12}|z|(r_1 s_{12}^{(2)} - r_2 s_{12}^{(1)})}{r_1 r_2 R_{12}^2 + z^2 s_{12}^{(1)} s_{12}^{(2)}}\Big) \end{aligned}$$

where $\tan^{-1}$ is defined over the range from $-\pi$ to π by taking into consideration the signs of the numerator and denominator. The potential and the velocities are then given by

$$\phi = R_{12}Q_{12} + R_{23}Q_{23} + R_{31}Q_{31} + |z|(J_{12} + J_{23} + J_{31} - \Delta\theta) \tag{44}$$

$$v_x = -S_{12}Q_{12} - S_{23}Q_{23} - S_{31}Q_{31} \tag{45}$$

$$v_y = +C_{12}Q_{12} + C_{23}Q_{23} + C_{31}Q_{31} \tag{46}$$

$$v_z = \mathrm{sgn}(z)(\Delta\theta - J_{12} - J_{23} - J_{31}) \tag{47}$$

Note that on the surface of the triangle all J's vanish and

$$v_z = 2\pi\ \mathrm{sgn}(z)$$

This complete our derivation of the added inertia matrix in terms of the fluid potential around the body.

Figure 7 shows a few examples of an ambient flow, computed by assigning source density distribution to the triangular elements of the surface of variously shaped bodies.

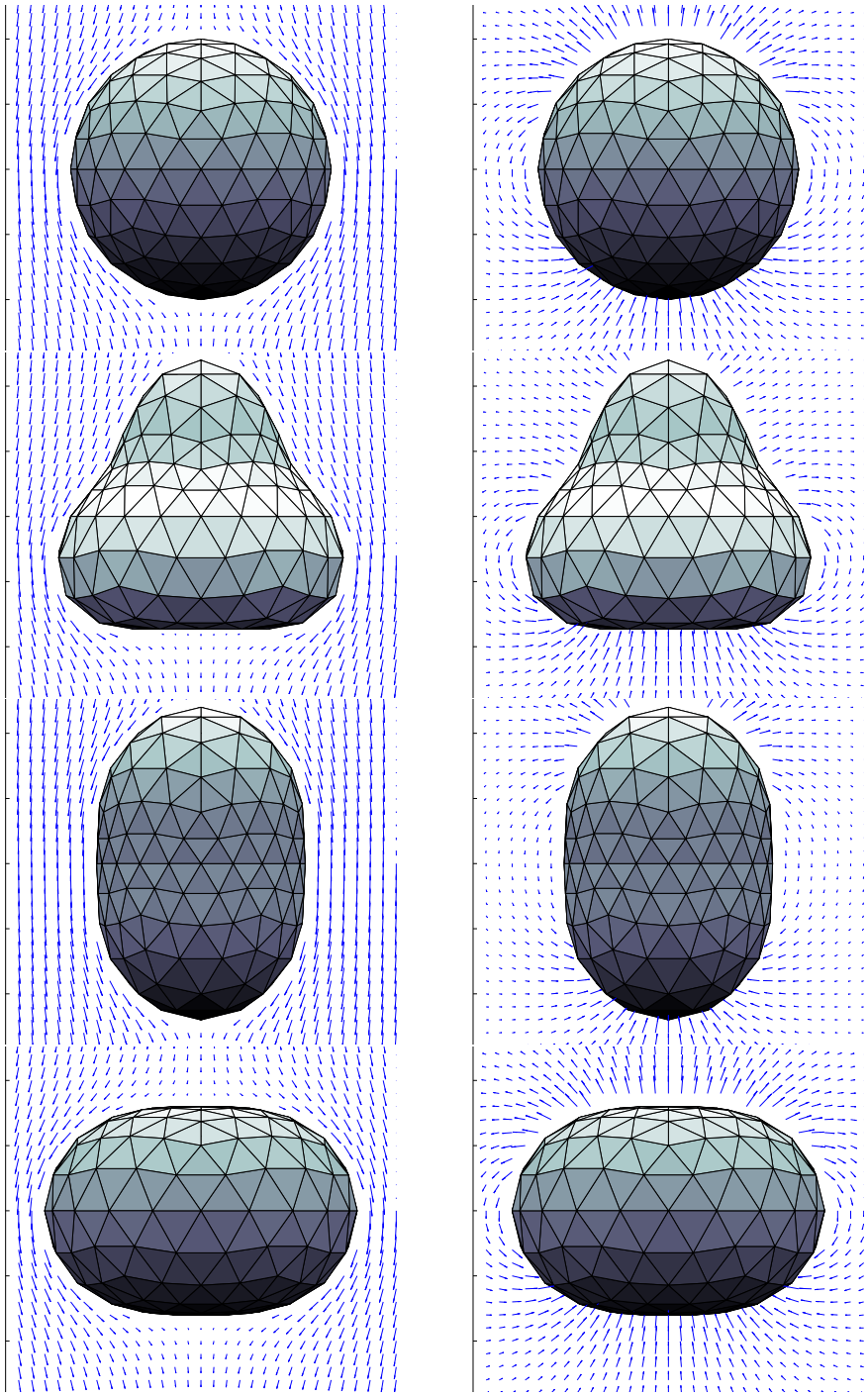
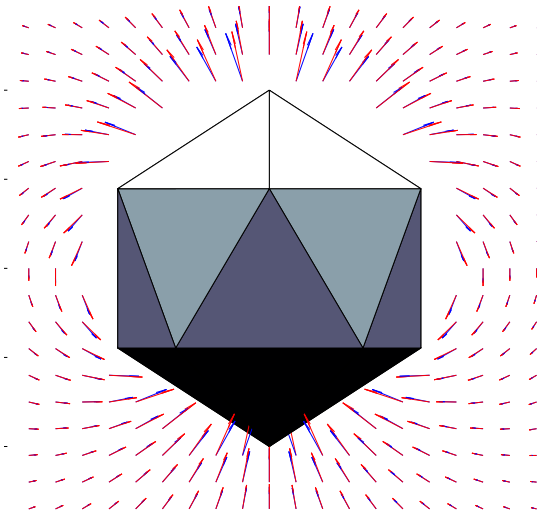
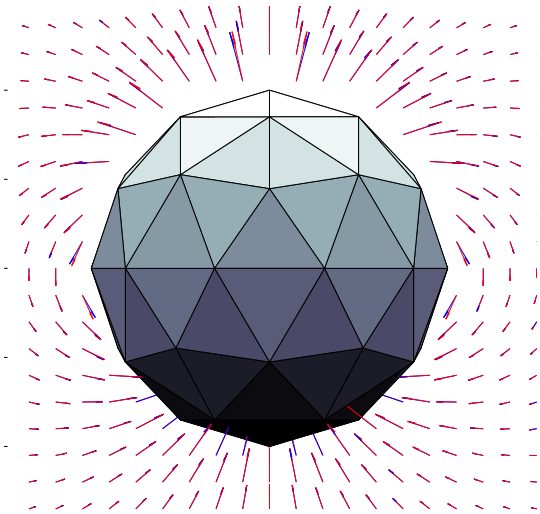


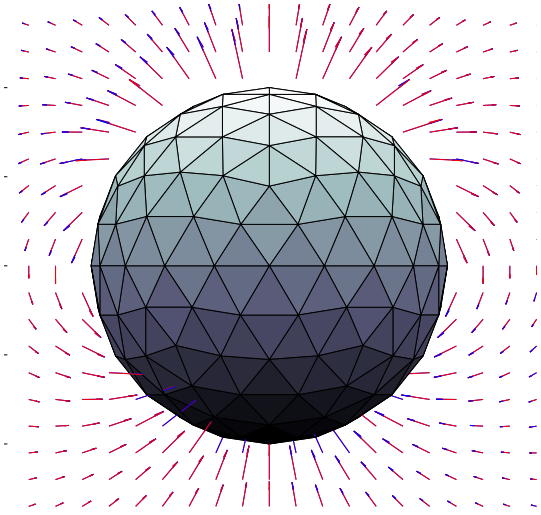
Figure 7: The flow around a body moving at unit velocity along the z axis in $\mathcal{F}_B$ and $\mathcal{F}_O$ respectively. The coloring shows the source distribution on the surface that would induce the velocity field. Darker colors correspond to regions which act as sinks and lighter ones to regions which act as sources.



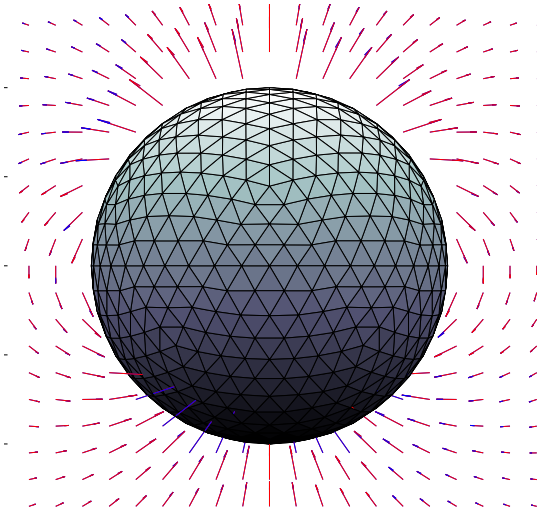
(a) 0



(b) 1



(c) 2



(d) 3

Figure 8: The numerically computed velocity (blue) deviates from actual velocity (red) due to the discrete nature of the triangulation. Higher orders of triangulations lead to an improvement in the fit.

4.3 Body Inertia

In this section we calculate the intrinsic inertia of the body in the absence of the fluid. The intrinsic inertia matrix Λ in the absence of the fluid satisfies

$$T_{\mathcal{B}} = \frac{1}{2} \begin{pmatrix} \dot{g} \\ \dot{s} \end{pmatrix}^T \begin{pmatrix} \Lambda^{gg} & \Lambda^{gs} \\ (\Lambda^{gs})^T & \Lambda^{ss} \end{pmatrix} \begin{pmatrix} \dot{g} \\ \dot{s} \end{pmatrix} \tag{48}$$

where $\dot{g} \equiv (\vec{v}, \vec{\omega})^T \in \mathbb{R}^6$ is the rate of translation and rotation of the body frame $\mathcal{F}_{\mathcal{B}}$. To compute the locked inertia matrix Λ^{gg} and the cross term matrix Λ^{gs} we need to make assumptions about the body's internal structure. We will assume that the amoeba is homogenous and that its kinetic energy is equal to that of an instantaneously rigid body with the same center-of-mass velocity and angular rotation.

$$T_{\mathcal{B}} = \frac{m\vec{v_c}^2}{2} + \frac{1}{2}\vec{\omega}^T I_c \vec{\omega} \tag{49}$$

I_c is the moment of inertia with respect to the center of mass.

There are three contributions to the center of mass motion

$$\vec{v_c} = \vec{v} + \vec{\omega} \times \vec{r} + \sum_i \vec{c_i}\dot{s}_i$$

where $\vec{v}$ is the motion due to the translation of the frame, $\vec{\omega} \times \vec{r}$ is the motion due to the rotation of the frame and $\sum_i \vec{c_i}\dot{s}_i$ is the motion due to the shape changes. $\vec{c_i}$ is the velocity of the center of mass when the body is perturbing its surface at unit speed $\dot{s}_i = 1$ with only mode i being active. Expanding

$$\begin{aligned} \vec{v_c}^2 = \quad & v^2 + (\vec{\omega} \times \vec{r}) \cdot (\vec{\omega} \times \vec{r}) + (\textstyle\sum_i \vec{c_i}\dot{s}_i)^2 + \\ & 2\vec{v} \cdot (\vec{\omega} \times \vec{r}) + 2\vec{v} \cdot \textstyle\sum_i \vec{c_i}\dot{s}_i + 2(\vec{\omega} \times \vec{r}) \cdot \textstyle\sum_i \vec{c_i}\dot{s}_i \end{aligned} \tag{50}$$

Consider the identity

$$(\vec{\omega} \times \vec{r}) \cdot (\vec{\omega} \times \vec{r}) = \vec{r} \cdot ((\vec{\omega} \times \vec{r}) \times \vec{\omega}) = \omega^2 r^2 - (\vec{\omega} \cdot \vec{r})^2 = \omega^2 r_\perp^2$$

and define

$$\Sigma = \begin{pmatrix} 0 & z & -y \\ -z & 0 & x \\ y & -x & 0 \end{pmatrix} \qquad \Pi = \begin{pmatrix} | & & | \\ \vec{c_1} & ... & \vec{c_n} \\ | & & | \end{pmatrix} \tag{51}$$

where x, y and z are the components of $\vec{r}$. Then

$$T_{\mathcal{B}} = \frac{m}{2}\left(v^2 + \omega^T \frac{I_c + mr_\perp^2}{m}\omega + 2(\Pi\dot{s})^2 + 2v^T\Sigma\omega + 2v^T\Pi\dot{s} + 2\omega^T\Sigma^T\Pi\dot{s}\right) \tag{52}$$

By the parallel axis theorem, $I = I_c + mr_\perp^2$ is the moment of inertia with respect to the origin of the $\mathcal{F}_{\mathcal{B}}$ frame.

Comparing Equation 48 with 52

$$\Lambda^{gg} = m\begin{pmatrix} \mathbf{1}_3 & \Sigma \\ \Sigma^T & I/m \end{pmatrix} \qquad \Lambda^{gs} = m\begin{pmatrix} \Pi \\ \Sigma^T\Pi \end{pmatrix} \tag{53}$$

Π is straightforward to evaluate - we simply make the body change its surface at unit speed and compute the displacement of the center of mass by averaging the centers of mass of all volume elements. It is slightly more complicated to compute I. We will now describe a numerical method to find the moment of inertia matrix for an arbitrary body. The moment of inertia matrix is a second order tensor defined by the following equation for the rotational kinetic energy

$$T_r = \frac{1}{2}\omega^T I\omega \tag{54}$$

where T_r is the rotational kinetic energy and ω is the angular velocity.

$$I = \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix} \tag{55}$$

The components of the inertia matrix are given by integrating the appropriates moments over the whole volume

$$\begin{aligned} I_{zz} &= \textstyle\int_V x^2 + y^2 dm \\ I_{xy} = I_{yx} &= -\textstyle\int_V xy dm \end{aligned} \tag{56}$$

Connecting the vertices of all triangular elements on the surface to the origin will break up the body into tetrahedra, with one vertex at the origin, and the other three on the surface of the body. The tetrahedra will emanate from the origin. Finding the moment of inertia of the body then reduces to finding the sum of the inertia matrices of each

tetrahedron. In the limit where the number of tetrahedra N tends to infinity, the sum of all moments of inertia will tend to the actual moment of inertia of our body.

$$I = \lim_{N\to\infty} \sum_{i=1}^{N} I_i \tag{57}$$

The problem of finding the moment of inertia for a general 3D body is then reduced to finding the sum of the inertia matrix for its composite tetrahedra.

Consider tetrahedron with vertices at $(\vec{v_0}, \vec{v_1}, \vec{v_2}, \vec{v_3})$ (Figure 9). Let us find its moment of inertia as a function of the coordinates of the vertices. First consider the diagonal element.

$$I_{zz} = \rho \int_{\mathcal{V}} x^2 + y^2 dv = \rho \oint_{\mathcal{A}} \vec{F} \cdot d\vec{a}$$

where we have used the divergence theorem and where $\vec{F}$ is such that $\nabla \cdot \vec{F} = x^2 + y^2$. One such $\vec{F}$ that satisfies this equation is

$$\vec{F} = \begin{pmatrix} 0 \\ 0 \\ (x^2+y^2)z \end{pmatrix}$$

Similarly in evaluating $\int_{\mathcal{V}} xy\, dv$ we use

$$\vec{F} = \begin{pmatrix} 0 \\ 0 \\ xyz \end{pmatrix}$$

Hence for a tetrahedron

$$I_{zz} = \rho \sum_{t=1}^{4} n_{zt} \oint_{\mathcal{A}_t} (x^2+y^2) z dA \tag{58}$$

$$I_{xy} = -\rho \sum_{t=1}^{4} n_{zt} \oint_{\mathcal{A}_t} xyz dA \tag{59}$$

where the index t runs over the 4 sides of the tetrahedron and n_z represents the z competent of the normal to the surface. We have to evaluate the four integrals in Equation 58, so we parameterize the triangle $\triangle_{ijk}$, described by vertices $\{\vec{v_i}, \vec{v_j}, \vec{v_k}\}$, in terms of s and t

$$\vec{r} = \vec{v_i} + s(\vec{v_i} - \vec{v_j}) + t(\vec{v_i} - \vec{v_k}), \text{ where} \begin{cases} s+t \leq 1 \\ s,t \geq 0 \end{cases}$$

or

$$\vec{r} = \vec{o} + s\vec{a} + t\vec{b}, \text{ where} \begin{cases} s+t \leq 1 \\ s,t \geq 0 \end{cases}$$

Thus

$$I_{zz}^{(i)} = \rho|\vec{a}\times\vec{b}|n_{zi}\int_0^s ds\int_0^{1-s} dt(\vec{r}^2 - z^2)z$$

$$I_{xy}^{(i)} = -\rho|\vec{a}\times\vec{b}|n_{zi}\int_0^s ds\int_0^{1-s} dtxyz$$

A more explicit form of the above integrals is then

$$\begin{aligned} I_{zz}^{(i)} = \rho|\vec{a}\times\vec{b}|\Big[& \frac{1}{60}(a_z b_x^2 + a_z b_y^2 + a_x^2 b_z + a_y^2 b_z) \\ + & \frac{1}{30}(a_x a_z b_x + a_y a_z b_y + a_x b_x b_z + a_y b_y b_z) \\ + & \frac{1}{20}(a_x^2 a_z + a_y^2 a_z + b_x^2 b_z + b_y^2 b_z) \\ + & \frac{1}{6}(a_x a_z o_x + b_x b_z o_x + a_z o_x^2 + b_z o_x^2 + a_y a_z o_y + b_y b_z o_y + a_z o_y^2 + b_z o_y^2) \\ + & \frac{1}{12}(a_z b_x o_x + a_x b_z o_x + a_z b_y o_y + a_y b_z o_y + a_x^2 o_z + a_y^2 o_z + a_x b_x o_z + b_x^2 o_z + a_y b_y o_z + b_y^2 o_z) \\ + & \frac{1}{3}(a_x o_x o_z + b_x o_x o_z + a_y o_y o_z + b_y o_y o_z) \\ + & \frac{1}{2}(o_x^2 o_z + o_y^2 o_z)\Big] \end{aligned} \tag{60}$$

$$\begin{aligned} I_{xy}^{(i)} = -\rho|\vec{a}\times\vec{b}|\Big[& \frac{1}{20}(a_x a_y a_z + b_x b_y b_z) \\ + & \frac{1}{60}(a_y a_z b_x + a_x a_z b_y + a_z b_x b_y + a_x a_y b_z + a_y b_x b_z + a_x b_y b_z) \\ + & \frac{1}{12}(a_y a_z o_x + b_y b_z o_x + a_x a_z o_y + b_x b_z o_y + a_x a_y o_z + b_x b_y o_z) \\ + & \frac{1}{24}(a_z b_y o_x + a_y b_z o_x + a_z b_x o_y + a_x b_z o_y + a_y b_x o_z + a_x b_y o_z) \\ + & \frac{1}{6}(a_z o_x o_y + b_z o_x o_y + a_y o_x o_z + b_y o_x o_z + a_x o_y o_z + b_x o_y o_z) \\ + & \frac{1}{2}o_x o_y o_z\Big] \end{aligned} \tag{61}$$

The other matrix elements of $I^{(i)}$ can be obtained by cyclical permutations of the subscript indices $\{x, y, z\}$. The integrals over the other faces of the tetrahedron may be obtained by suitably replacing $\vec{a}$, $\vec{b}$ and $\vec{o}$ with the appropriate vertices. Finally we obtain the total moment of inertia matrix of the tetrahedron by summing the integral over all faces (Figure9),i.e.

$$I^{(i)} = I_{\triangle 102}^{(i)} + I_{\triangle 203}^{(i)} + I_{\triangle 301}^{(i)} + I_{\triangle 123}^{(i)} \tag{62}$$

where we identify the triangle $\triangle_{ijk}$ with $\triangle_{\vec{o},\vec{o}+\vec{a},\vec{o}+\vec{b}}$. To obtain the total moment

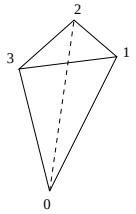


Figure 9: Faces of tetrahedron.

of inertia for the body we must additionally sum over all tetrahedra.

$$I = \sum_{i=1}^{N} I^{(i)} \tag{63}$$

Since each of the interior faces ($\triangle_{102}$, $\triangle_{203}$, $\triangle_{301}$) occur twice with opposite signs - once for each of two neighboring volume elements - their contribution to the sum will cancel out and the sum will simplify to

$$I = \sum_{i=1}^{N} I^{(i)}_{\triangle 123} \tag{64}$$

5 Amoeba Example

So far we have described two methods to find the potential around a deformable body - analytically through the boundary perturbation method and numerically through the panel method. In this section there will be frequent references to the boundary perturbation section. Here we will compute the potential to 1st order explicitly for simple motion given by

$$F = 1 + \epsilon(s_1 \cos(2\varphi) + s_2 \cos(3\varphi)) \tag{65}$$

The normal component of the velocity of the body on the boundary will be

$$v = \left[R \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} + \epsilon S \begin{bmatrix} \dot{s_1} \\ \dot{s_2} \\ \dot{s_3} \end{bmatrix} \right] \cdot \vec{n} \tag{66}$$

Here R is a rotation matrix, which converts from cartesian to spherical coordinates. An explicit form of R is given by Equation 67

$$R = \begin{pmatrix} \sin(\varphi)\cos(\theta) & \sin(\varphi)\sin(\theta) & \cos(\varphi) \\ \cos(\varphi)\cos(\theta) & \cos(\varphi)\sin(\theta) & -\sin(\varphi) \\ -\sin(\theta) & \cos(\theta) & 0 \end{pmatrix} \tag{67}$$

And S describes the surface variations (Equation 68)

$$S = \begin{pmatrix} \cos(2\varphi) & \cos(3\varphi) & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{68}$$

The Laplace equation with the above boundary condition may be solved by separation of variables

$$\phi = \phi_z \dot{z} + \phi_1^s \dot{s_1} + \phi_2^s \dot{s_2} \tag{69}$$

where ϕ_i are the Kirchhoff potentials. To simplify the problem we correctly assume that $\phi_x = \phi_y = 0$, which is implied by the azimuthal symmetry. Due to the azimuthal symmetry the solution will involve the Legendre polynomials.

$$\phi(r, \varphi) = \sum_{k=0}^{\infty} a_k \frac{P_k(\cos\varphi)}{r^{k+1}}$$

P_l are the Legendre polynomials defined by the Rodriguez formula.

$$P_l(x) = \frac{1}{2^l l}\left(\frac{d}{dx}\right)^l (x^2 - 1)^l \tag{70}$$

By Equation 27

$$\vec{n} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + 2 \begin{bmatrix} \eta(\varphi) \\ -\dot{\eta}(\varphi) \\ 0 \end{bmatrix} \epsilon + \begin{bmatrix} \eta(\varphi)^2 \\ -\eta(\varphi)\partial_\varphi \eta(\varphi) \\ 0 \end{bmatrix} \epsilon^2 \tag{71}$$

where for brevity $\eta \equiv s_1 \cos(2\varphi) + s_2 \cos(3\varphi)$ and $\partial_\varphi \eta \equiv -2s_1 \sin(2\varphi) - 3s_2 \sin(3\varphi)$.

By Equation 28

$$\vec{\psi_k} = \begin{bmatrix} -(k+1)P_k \\ \frac{\partial}{\partial\varphi}P_k(\cos\varphi) \\ 0 \end{bmatrix} \tag{72}$$

Now we have all the information we need to solve the system of equations for a_k and to find the potentials. Computing the first few functions $u_k^{(0)}$ yields the result below[4]

$$\begin{aligned}
{u_0}^{(0)} &= -1 \\
{u_1}^{(0)} &= -2c(\varphi) \\
{u_2}^{(0)} &= -\frac{3}{4}(1+3c(2\varphi)) \\
{u_3}^{(0)} &= -\frac{1}{2}(3c(\varphi)+5c(3\varphi)) \\
{u_4}^{(0)} &= -\frac{5}{64}(9+20c(2\varphi)+35c(4\varphi)) \\
{u_5}^{(0)} &= -\frac{3}{64}(30c(\varphi)+35c(3\varphi)+63c(5\varphi)) \\
{u_6}^{(0)} &= -\frac{7}{512}(50+105c(2\varphi)+126c(4\varphi)+231c(6\varphi)) \\
{u_7}^{(0)} &= -\frac{1}{128}(175c(\varphi)+189c(3\varphi)+231c(5\varphi)+429c(7\varphi))
\end{aligned}$$

or in matrix form

$$B_{\{kl\}} = \begin{pmatrix}
-1 & & & & & & & \\
 & -2 & & & & & & \\
-\frac{3}{4} & & -\frac{9}{4} & & & & & \\
 & -\frac{3}{2} & & -\frac{5}{2} & & & & \\
-\frac{45}{64} & & -\frac{25}{16} & & -\frac{175}{64} & & & \\
 & -\frac{45}{32} & & -\frac{105}{64} & & -\frac{189}{64} & & \\
-\frac{175}{256} & & -\frac{735}{512} & & -\frac{x441}{256} & & -\frac{1617}{512} & \\
 & -\frac{175}{128} & & -\frac{189}{128} & & -\frac{231}{128} & & -\frac{429}{128}
\end{pmatrix}$$

where $u_k^{(0)} = B_{kl}\cos(l\varphi)$

Using Equation 33 we obtain the potentials [5]

$$\begin{aligned}
\phi_z &= -\frac{1}{2}\frac{P_1}{r^2} + \epsilon\Big(\frac{9s_1}{10}\frac{P_1}{r^2} + \frac{9s_2}{7}\frac{P_2}{r^3} - \frac{6s_1}{5}\frac{P_3}{r^4} - \frac{48s_2}{35}\frac{P_4}{r^5}\Big) \\
{\phi_1}^s &= \epsilon\Big(\frac{1}{3}\frac{P_0}{r^1} - \frac{4}{9}\frac{P_2}{r^3}\Big) \\
{\phi_2}^s &= \epsilon\Big(\frac{3}{10}\frac{P_1}{r^2} - \frac{2}{5}\frac{P_3}{r^4}\Big)
\end{aligned}$$

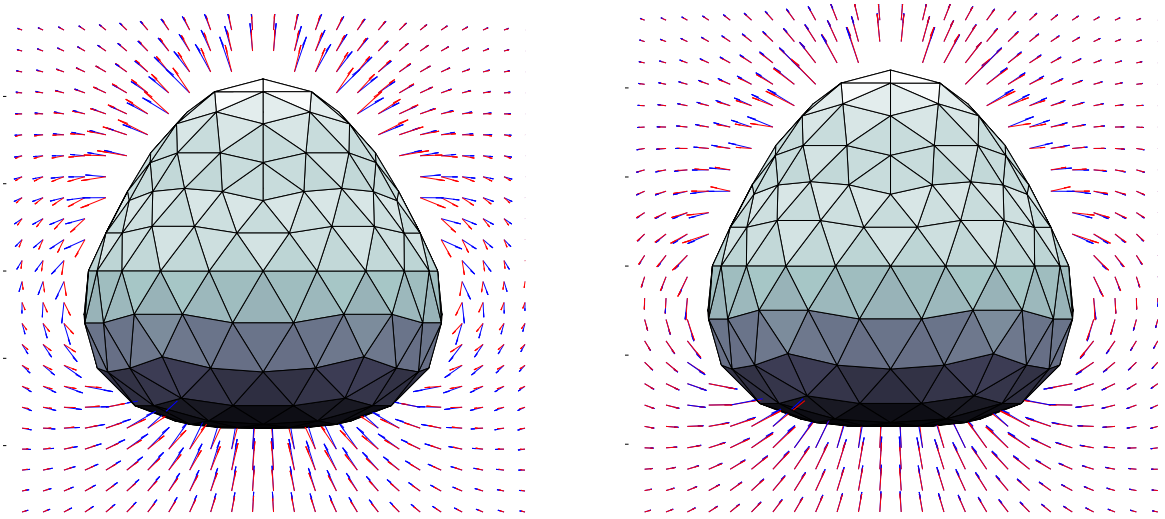
[4]For convenience we denote $\cos(\varphi)$ by $c(\varphi)$.

[5]We use the abbreviation $P_l(cos\varphi) = P_l$

or more explicitly

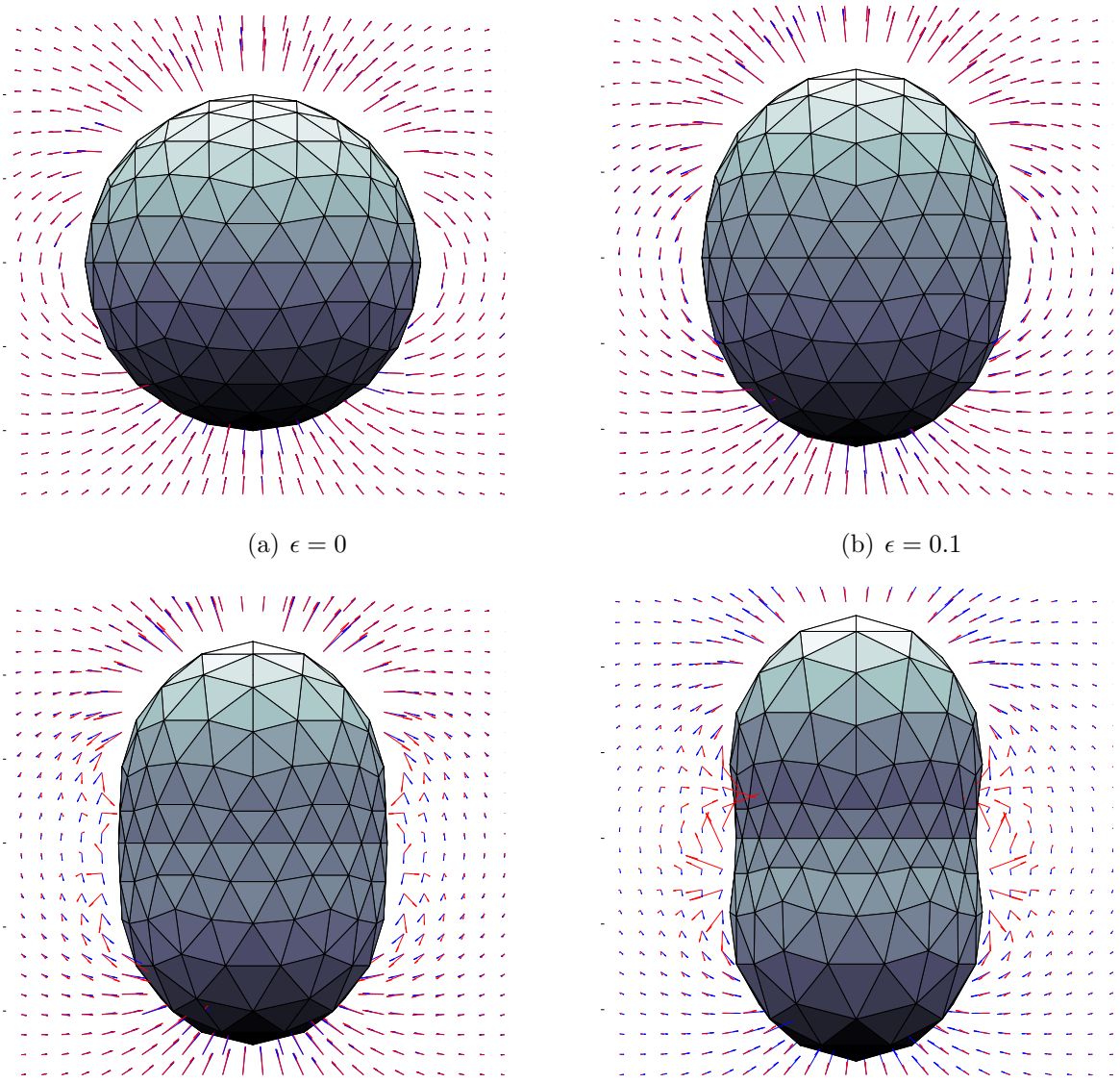
$$
\begin{aligned}
\phi_z &= -\frac{c(\varphi)}{2r^2} + \epsilon\Big(\frac{9}{10r^2}c(\varphi)s_1 + \frac{9}{28r^3}(1+3c(2\varphi))s_2 \\
&- \frac{3}{20r^4}(3c(\varphi)+5c(3\varphi))s_1 - \frac{3}{140r^5}(9+20c(2\varphi)+35c(4\varphi))s_2\Big) \\
{\phi_1}^s &= \epsilon\Big(\frac{1}{3r} - \frac{1+3c(2\varphi)}{9r^3}\Big) \\
{\phi_2}^s &= \epsilon\Big(\frac{3c(\varphi)}{10r^2} - \frac{3c(\varphi)+5c(3\varphi)}{20r^4}\Big)
\end{aligned}
$$

To see a comparison of this analytical result with the numerical result of the panel method refer to Figure 12, 13 , 14. The agreement of the results is a good validation of both the equations for boundary perturbation and the numerical method developed in the previous sections.



(a) $v^{(0)}$ (b) $v^{(0)} + \epsilon v^{(1)}$

Figure 10: An illustration of a perturbed sphere ($s_1 = 0, s_2 = 1$) moving along the z axis. The blue arrows represent the exact velocity v computed numerically. The red arrows show the uncorrected velocity field $v^{(0)}$ corresponding to a sphere (a), and the first order correction to the velocity field $v^{(0)} + \epsilon v^{(1)}$ due to the perturbation($\epsilon = 0.1$)(b). Adding the first order correction improves the fit with the numerically computed velocity.



(a) $\epsilon = 0$

(b) $\epsilon = 0.1$

(c) $\epsilon = 0.2$

(d) $\epsilon = 0.3$

Figure 11: First order boundary perturbation theory (red) fails for large perturbations ($s_1 = 1, s_2 = 0$).

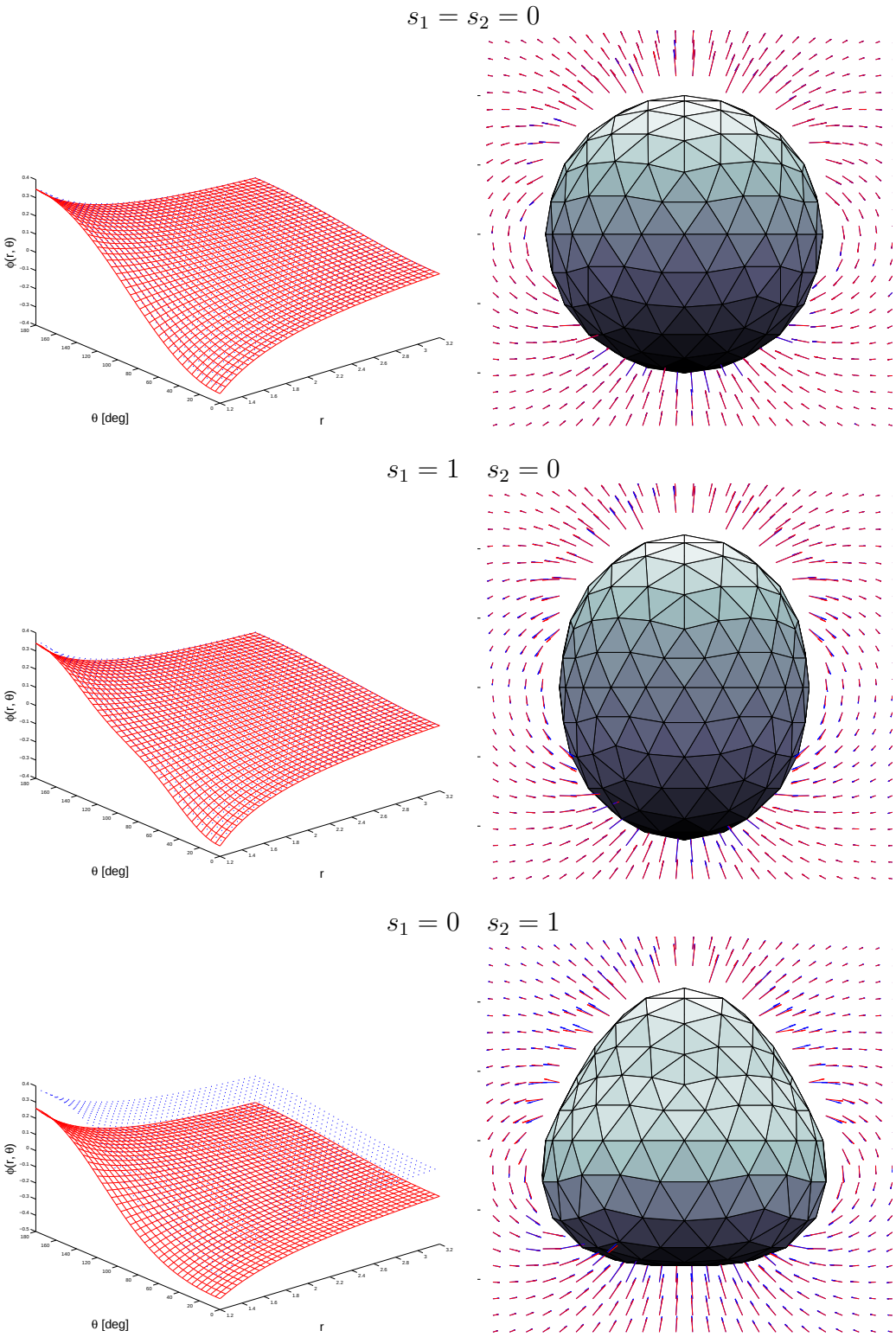


Figure 12: Comparison of the results of 1st order perturbation (red) and the numerical method(blue) for ϕ_z ($\dot{z} = 1$).

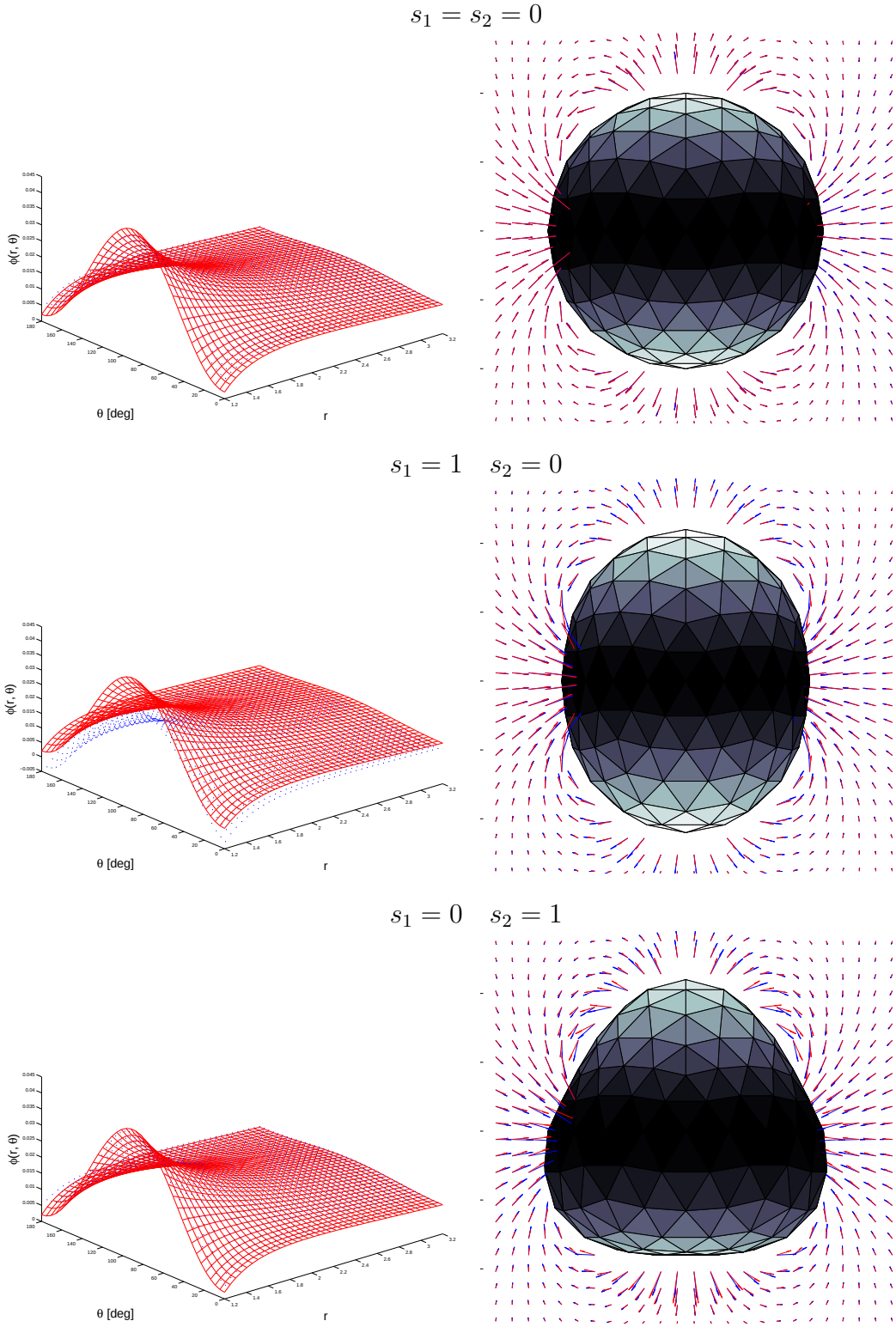


Figure 13: Comparison of the results of 1st order perturbation (red) and the numerical method(blue) for ϕ_1^s ($\dot{s}_1 = 1$).

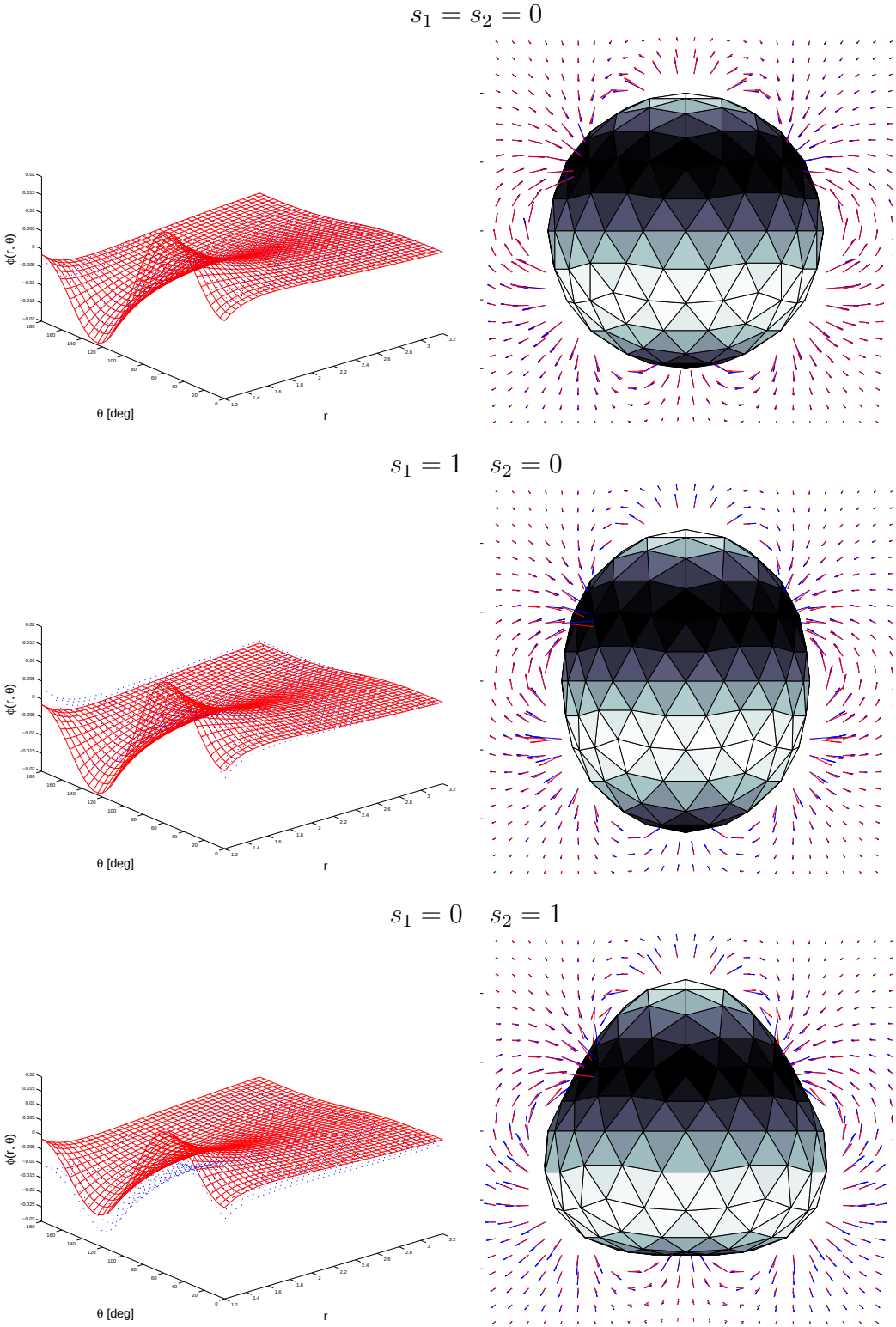


Figure 14: Comparison of the results of 1st order perturbation (red) and the numerical method(blue) for ϕ_2^s ($\dot{s}_2 = 1$).

6 Gaits, Displacement and Rotation

Mason and Burdick [3] have found that for a two dimensional amoeba-like body with surface undulations governed by

$$F(\theta, s) = 1 + \epsilon \left(s_1 \cos 2\theta + s_2 \cos 3\theta \right) \tag{73}$$

the velocity of the centroid is

$$v_c = \epsilon^2 \mu \dot{s}_1 s_2 \hat{x} + \mathcal{O}(\epsilon^3) \tag{74}$$

where $\mu = (2\pi r_0^2 \rho)/(M + \pi r_0^2 \rho)$

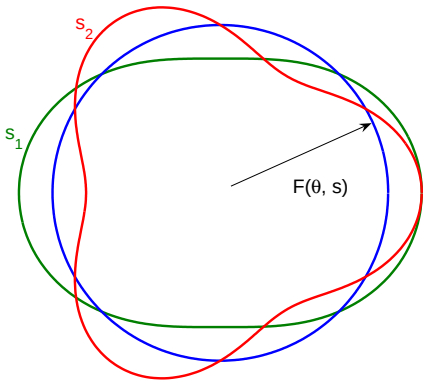


Figure 15: 2D amoeba surface variation modes

It is much harder to find an explicit form of the governing motion in the three dimensional case due to the complexity of the potentials computed using the boundary perturbation method.It will suffice to considering only the numerical results.

It is expected that a 3D body moves slower than a two dimensional one, since the fluid is not restricted to a single plain but can "escape" by flowing in the extra dimension available, thus reducing its propulsive effect on the body.

We will study forward motion gaits and turning gaits for a three dimensional amoeba-like object. Note that a 2D amoeba corresponds to several 3D amoebas in

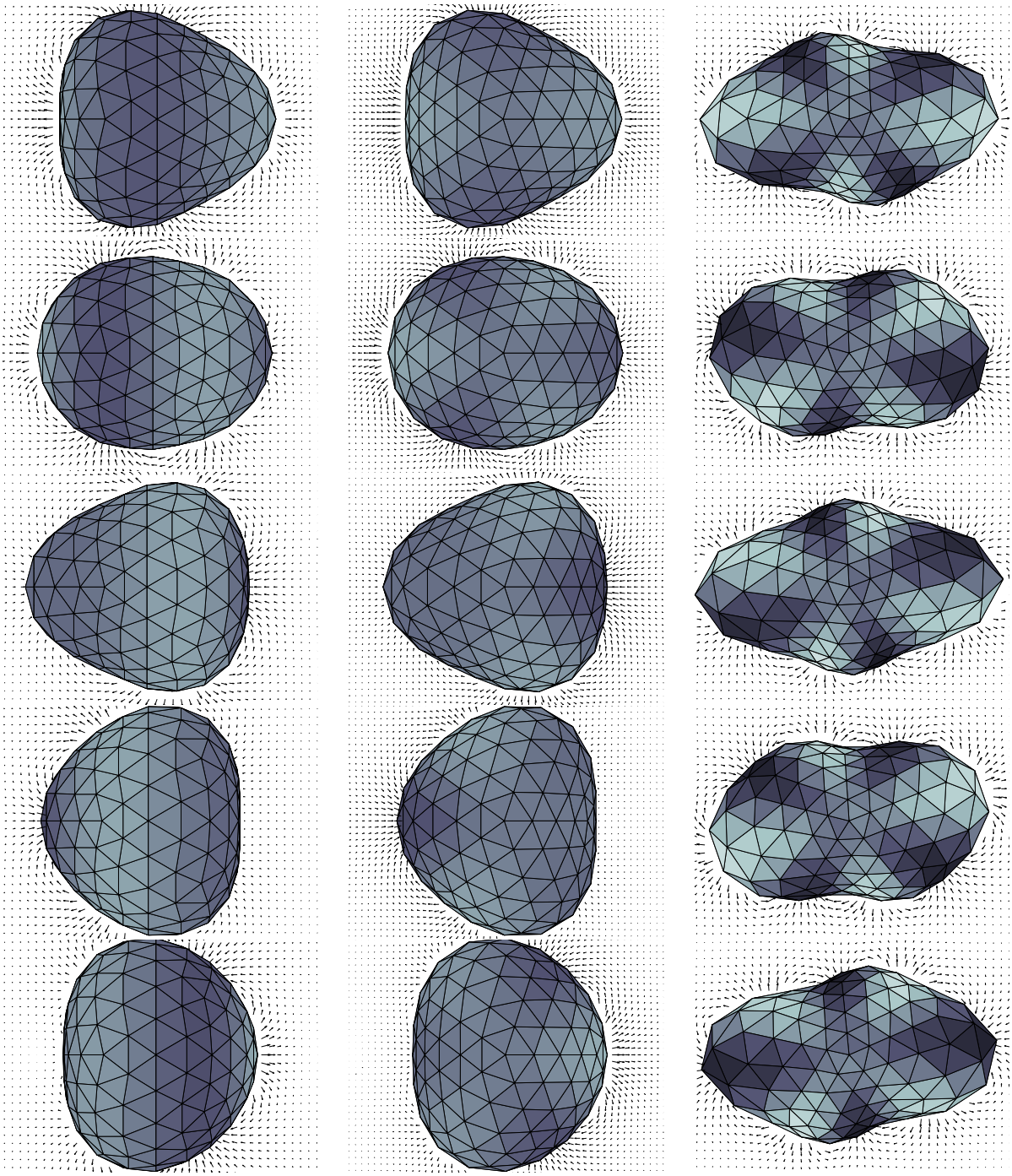


Figure 16: Computer animation of the forward gait for rotationally symmetric amoeba, forward gait for planar amoeba and turning gait with 5 frames over a single oscillation.

the sense that there are more than one modes of surface variation corresponding to the same 2D mode. Here are two examples:

1. Planar amoeba

$$F = 1 + \epsilon \sin\varphi \ (s_1 \cos 2\theta + s_2 \cos 3\theta) \tag{75}$$

2. Rotationally symmetric amoeba:

$$F = 1 + \epsilon(s_1 \cos 2\varphi + s_2 \cos 3\varphi) \tag{76}$$

where s_1 and s_2 evolve in time like

$$s_1 = \sin t \qquad s_2 = \cos t$$

The two possibilities and the resulting displacement over a single period of 2π are shown on Figure 17. The rotationally symmetric amoeba makes a huge backward dip before moving forward. This oscillation is related to the fact that the deformation of the body shifts the center of mass. The origin of $\mathcal{F}_{\mathcal{B}}$ moves so as to conserve the position of the center of mass. In the case of the rotational symmetry there is more moving mass concentrated around the periphery and hence the effect is much more pronounced.

A turning gait may be designed by propagating waves around the surface of the body. Instead of using a spherical body we will choose one that has an ellipsoid surface in order to make turning apparent:

$$F_0 = (\cos^2\theta \ \sin^2\varphi + 0.3 \sin^2\theta \ \sin^2\varphi + \cos^2\varphi)^{-\frac{1}{2}} \tag{77}$$

Turning gait:

$$F = F_0 + \epsilon \sin(s_1 + 4\theta) \sin\phi \tag{78}$$

and let $s_1 = 3t$. The resulting motion is shown on Figure 18.

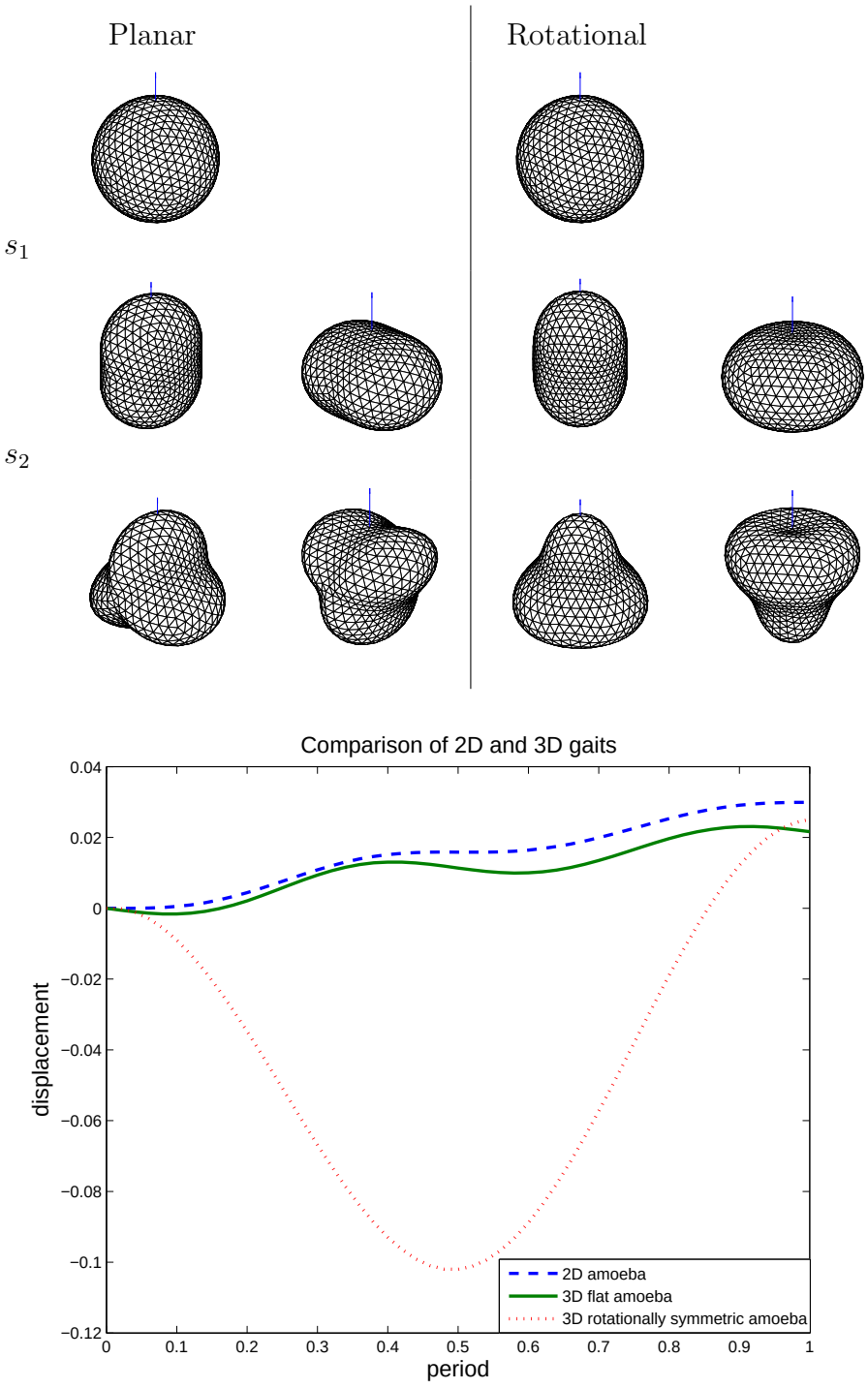


Figure 17: Planar and rotationally symmetric modes of forward motion. The figure depicts the two modes of surface variation of a spherical amoeba. The plot compares the displacement of the two 3D gaits with the 2D gait for $\epsilon = 0.1$.

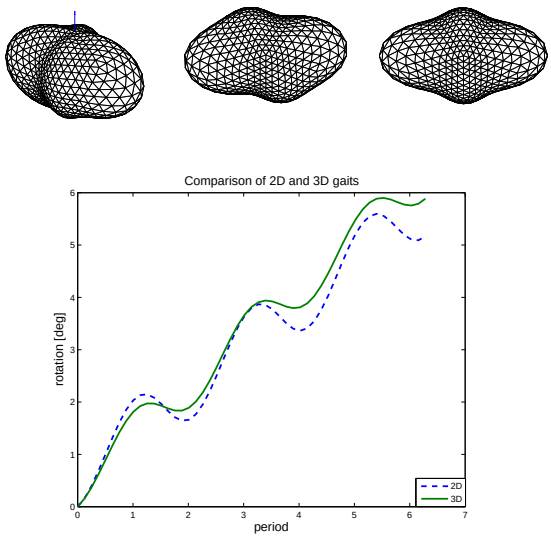


Figure 18: Rotation in the case of a 2D (dashed) and 3D body.

Surprisingly the rotational displacement in the three dimensional case is larger, hinting that the fluid alone cannot account for this rotation. The effect of turning without any bearing point conserving angular momentum at zero, is similar to the ability of cats to turn around in midair and always fall on their feet. The effect is stronger in three dimensions since there is more mass that can be moved in the periphery than in two dimensions.

It is important to mention the issue of convergence here - the program requires at least a second order of triangulation in order to lead to reasonable results.

In the 3D case the number of panels on the surface of the body is quadratic in their characteristic size. Therefore, reducing the size of the panel leads to a quadratic increase in the number of panels, which on the other hand leads to a quadruple increase in running time (as mentioned earlier our algorithm is $\mathcal{O}(N^2)$ where N is the number of surface triangles). Thus in the three dimensional case it is hard to obtain results as accurate in the 2D case. Figure 6 illustrates the sequence of displacement curves in the case of the rotationally symmetric amoeba and the planar amoeba, obtained by applying higher order of triangulation each

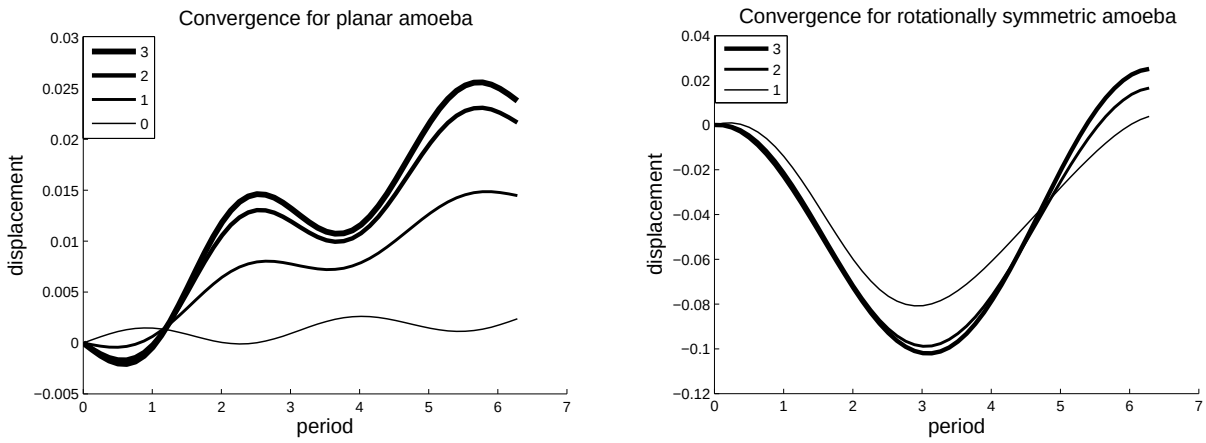


Figure 19: A sequence of displacement curves for planar and rotational amoeba generated by increasing the number of surface triangles.

time. It seems that in our case, 2nd or 3rd order triangulation is enough to obtain reasonably accurate solutions.

We have already validated the code by comparing its output with the analytical solution of perturbation theory. Since we know the solutions to the 2D problem [3], we have another possibility to validate the 3D code by checking if it leads to the same results in the limit, where we have a very long cylindrical amoeba. In this case, the results we obtain must be the same as the results of the two dimensional code. We stretch the amoeba so that the solution tends to the 2D case by converting it into a cylinder and letting the height to diameter ratio H/D go to infinity (Figure 20).

Unfortunately, the simulations depicted on these figures suffer from a serious deficiency. The triangulations scheme is inadequate for elongated bodies, triangles further from the center are much bigger than those close to the center.

In the case of the cylindrical amoeba, we obtain what we would expect, as the height of the amoeba grows the displacement tends to that of a 2D amoeba. Figure 20 shows the convergence of the trajectory of the displacement of a cylinder to the 2D amoeba solution as H/D increases. This is a separate and independent

confirmation of the validity of our code.

Another way to elongate the amoeba is to stretch it into an ellipsoid (Figure 21).

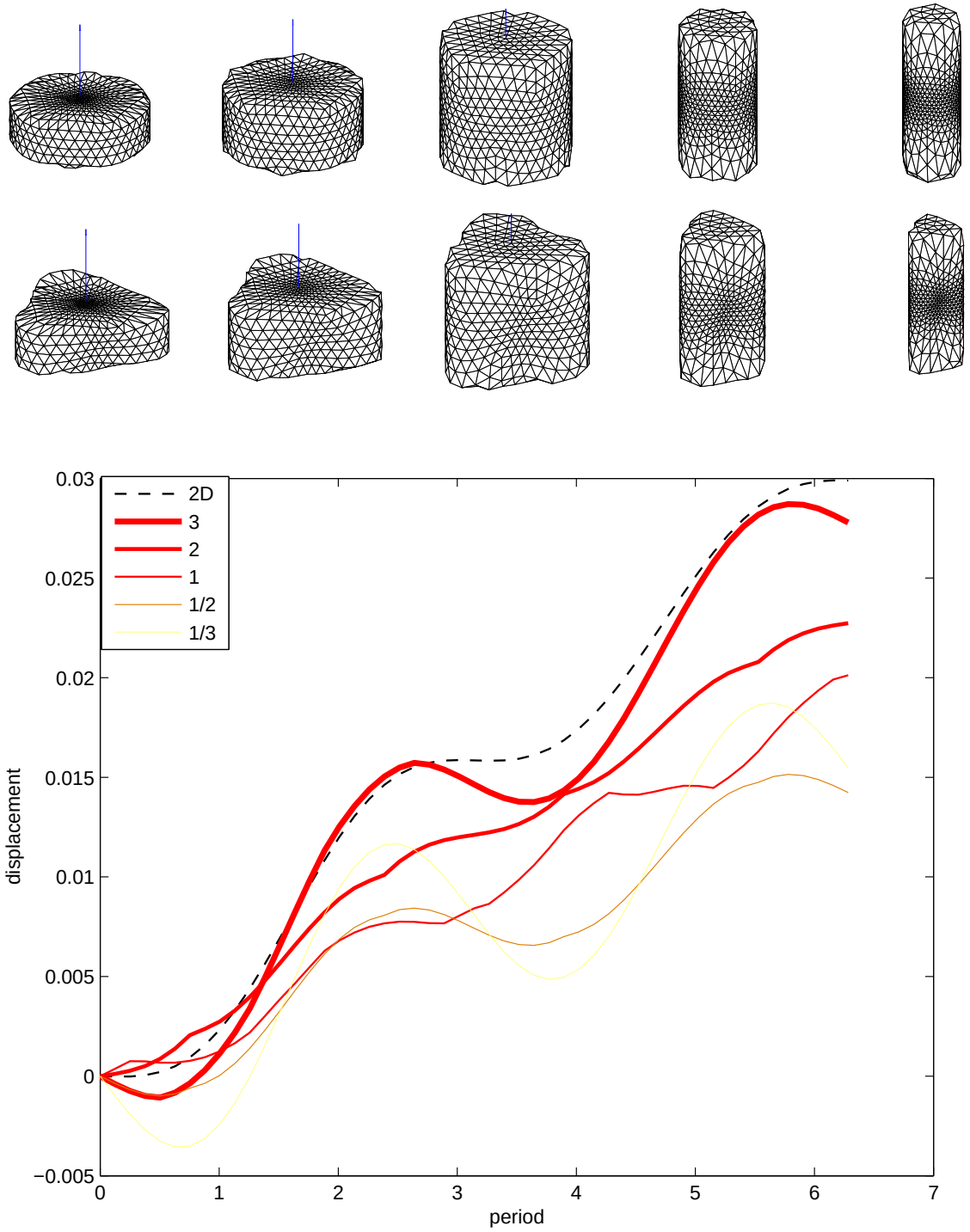


Figure 20: Comparison of the 2D displacement curve (dashed) with the 3D displacement for a cylindrical amoeba of height to diameter ratios 1/3, 1/2, 1, 2, 3.

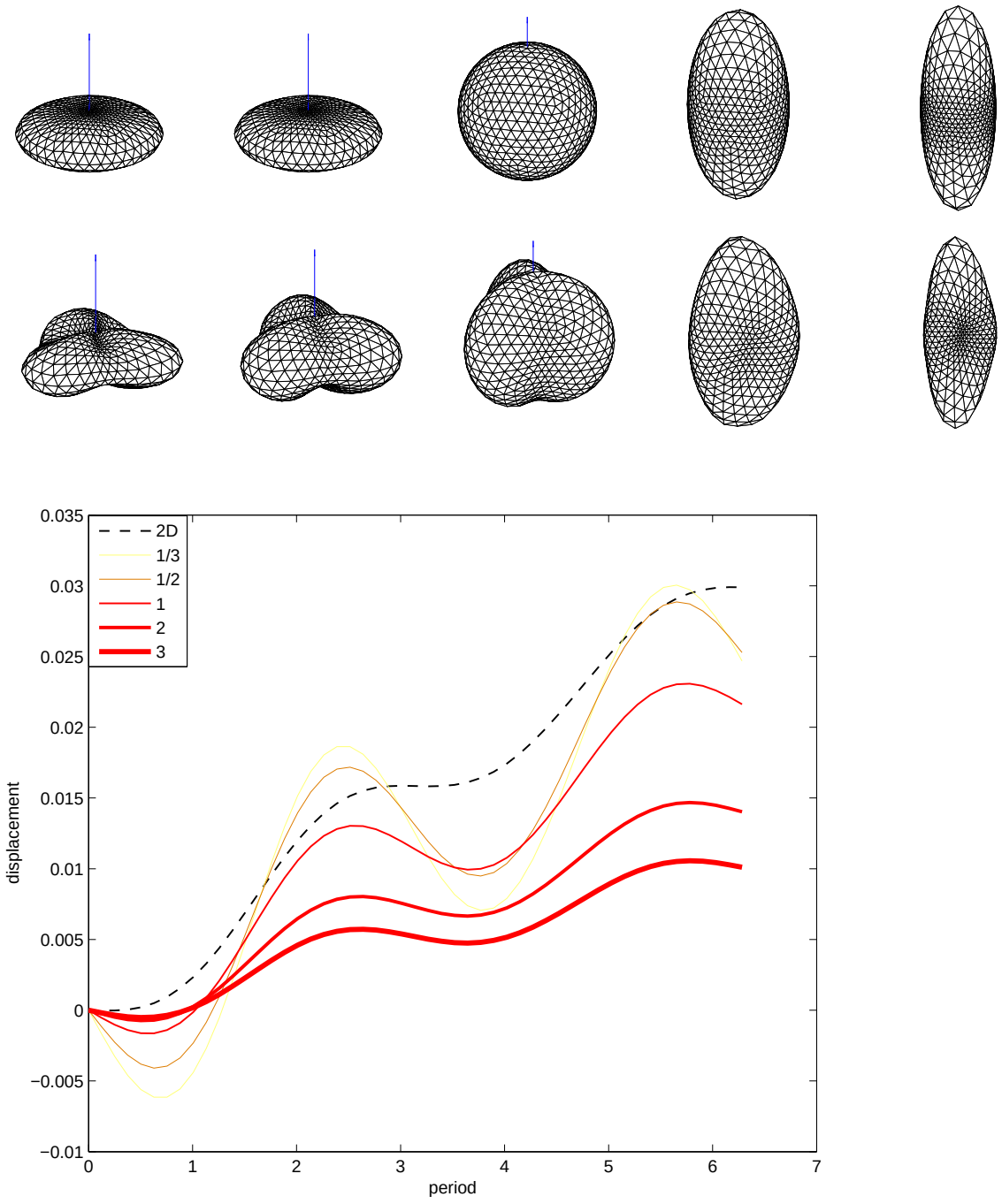


Figure 21: Comparison of the 2D displacement curve (dashed) with the 3D displacement for an ellipsoidal amoeba of eccentricity 1/3, 1/2, 1, 2, 3. In the case of the ellipsoidal shape elongating the body does not lead to convergence with the 2D solution - elongating the body makes its displacement lower. The reason for this is that the diameter decreases with height.

7 Control

So far we have focused on designing a program that outputs the motion of the swimmer given how its shape varies. In this part we will briefly outline suggestions for a method that will allow us to do the reverse - output the shape changes of the body given the desired trajectory. The connection is given by:

$$g^{-1}\dot{g} = -\mathcal{A}\dot{s} \tag{79}$$

In the case where $\mathcal{A}$ is a square non-singular matrix it may be inverted to obtain:

$$\begin{aligned} \dot{s} &= \mathcal{A}^{-1}(s)g^{-1}\dot{g} \\ \dot{s} &= (g\mathcal{A}(s))^{-1}\dot{g} \end{aligned} \tag{80}$$

$\mathcal{A}$ will be a square matrix if and only if the number of shape variables is equal to the number of group variables. The above differential equation may then be evolved in time to obtain how the shape variables should change in order for the body to follow a trajectory defined by $g(t)$. This simple method will suffer from two limitations.

First, the connection may become singular at some point. Second, the differential equation that we will need to solve may lead to values of s that are not allowed and lead to non-physical solutions, for instance the radius of the body cannot be negative. These limitations reduce the applicability of the above formula.

Another possibility to control is to design turning and forward gaits that would allow us to roughly steer the body according to the changes in the direction of the trajectory. We will steer the body in a plane by expressing the shape changes as the superposition of two components.

$$s(t) = s_f(t) + s_r(t) \tag{81}$$

where s_f is the forward motion gait which leads to forward displacement over a single undulation, and s_r is the turning gait which leads to a net rotation

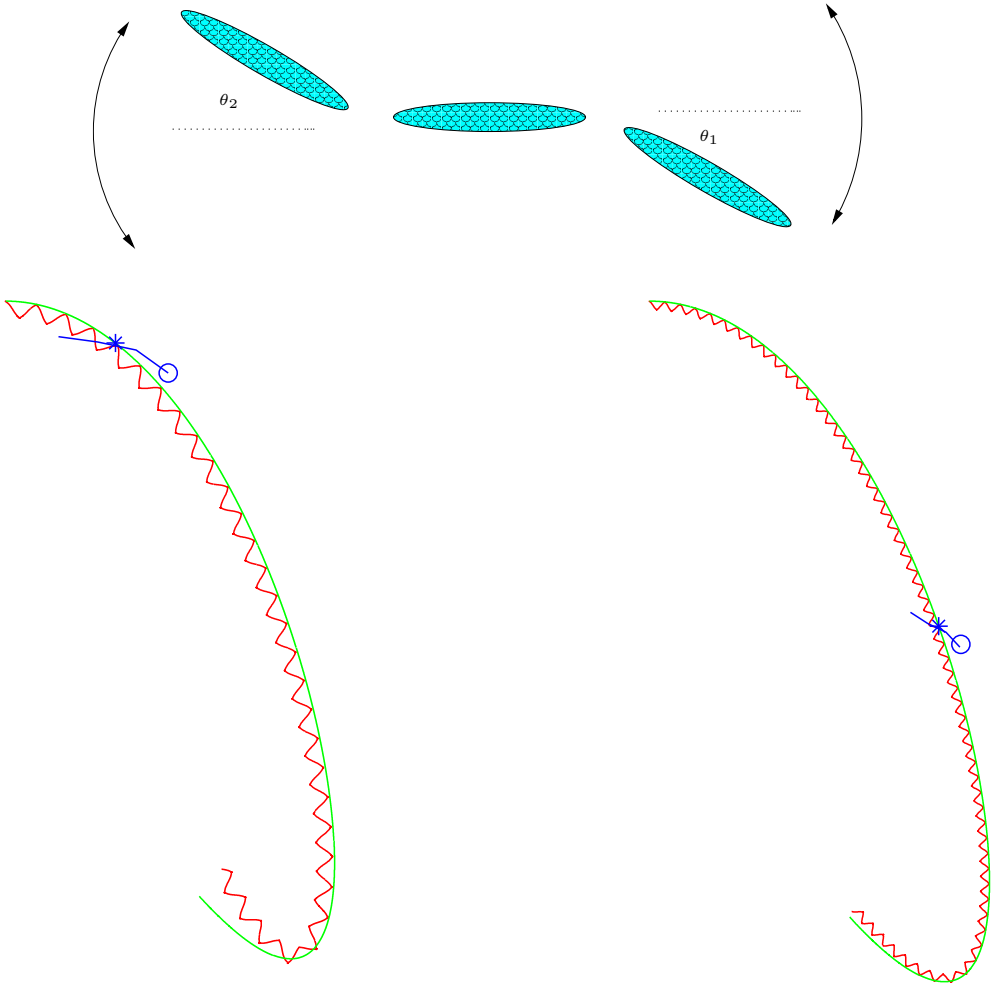


Figure 22: Implementation of the adiabatic approximation for a three-link simmer consisting of three ellipses that can move relative to each other [2].The swimmer achieves motion by varying the two angles θ_1 and θ_2. The desired trajectory is the green curve, and the actual trajectory of the body center of mass is the red curve. It is clear from the graphs that as the scale of each oscillation decreases relative to the scale of the trajectory, we obtain a better fit with the desired trajectory.

over a single undulation.Unfortunately due to the non-linearity of the problem a superposition of the two gaits will not lead to motion in the desired direction.

Instead we will employ an adiabatic approximation. Let the forward gait be

given by the real part of the rapidly oscillating function

$$s_f = Ae^{i\omega t} \tag{82}$$

By introducing some constant deformation B to the body we cause a deviation from the forward direction.

$$s = Ae^{i\omega t} + B \tag{83}$$

Now instead of letting B be constant in time, we will start changing it in such a way that we follow a curved trajectory. In the adiabatic approximation we assume that $\dot{B}$ is negligible, i.e. over the period of a single surface undulation B may be assumed constant.

In the case of potential flow the motion is time scale invariant. Thus it is convenient to introduce $d\tau = \omega\, dt$

$$s = Ae^{i\tau} + B \tag{84}$$

Set B at some constant value and compute the displacement δ and rotation β from the forward direction (defined as the direction in which the body moves at $B = 0$). The average velocity and curvature will then be

$$< v >= \frac{\delta}{T} = \frac{\omega\delta(B)}{2\pi} \qquad < \kappa >= \frac{\beta(B)}{\delta(B)}$$

We compute $< \kappa >$ for several values B, and applying a polynomial fit we may compute the inverse function, i.e. B as a function of $< \kappa >$.

$$B = P(< \kappa >)$$

We want the body to follow a certain predefined trajectory

$$\vec{r}(t) = (x(t), y(t))$$

where $v(t)$ and $\kappa(t)$ are the velocity and the curvature along the trajectory.

$$v(t) = (\dot{x}^2 + \dot{y}^2)^{1/2} \quad \kappa(t) = \frac{\dot{x}\ddot{y} - \dot{y}\ddot{x}}{(\dot{x}^2 + \dot{y}^2)^{3/2}}$$

In the adiabatic approximation we may ignore the contribution of $\dot{B}$ to $\dot{s}$, so we expect the same behavior as if $\dot{B}$ were zero and match $< v >= v$ and $< \kappa >= \kappa$ and solve for ω and B.

$$B(t) = P(\kappa) \quad \omega(t) = 2\pi \frac{v}{\delta(B)}$$

This completes the computation of the shape variables as a function of time

$$s = Ae^{i\int_0^t \omega(t')\,dt'} + B(t) \tag{85}$$

This method will only work well for smoothly varying trajectories, where the specific dimension of the trajectory is much larger than the displacement per unit period. Although small deviations from the trajectory during each oscillation will exist, we prefer this method since in general we are only interested reaching the destination point from the departure point without worrying about the trajectory.

In the adiabatic approximation we have assumed that the body oscillates rapidly in a time scale too small for any changes along the trajectory to take place. Unfortunately, rapidly swimming macroscopic bodies shed vorticity, and it is necessary to include this effect to develop a more realistic model of the motion.

8 Conclusion

We have shown how boundary perturbation theory can be used to compute the flow around a general class of deformable bodies. We have implemented a numerical algorithm to compute the motion of a deformable body in three dimensions. We have looked at several types of surface undulations that lead to forward or turning motion. We have suggested ways to control the motion of the body by varying the shape in such a way that the body follows a predefined trajectory.

In our analysis we have assumed potential flow, which is the solution to the Navier-Stokes equation for very high Reynolds numbers ($Re \gg 1$). It would

be illuminating to analyze fluids on the other side of the spectrum ($Re \ll 1$), where the Stokes flow approximation holds and compare the results with those of potential flow. For a microscopic amoeba the relevant equations are those of creeping flow. Creeping flow can be analyzed using similar techniques to those used for potential flow in this paper, and it may be a first step in understanding the role of viscosity on the motion.

A Code

We implement the numerical method in MATLAB. The code consists of 12 files. The main file that we use is driver. The generic form of running driver is

```
>>driver(order of triangulation, number of periods, 'filename');
```

For example the command

```
>>driver(2, 1, 'filename');
```

would output the results in 'filename' using 2nd order triangulation over a single period. To encode the surface variation we need to change the files surface.m for the spacial variation and shape_var.m for the temporal variation. The output is given in the form

$$t \quad x \quad y \quad z \quad R_{xx} \quad R_{xy} \quad R_{xz} \quad R_{yx} \quad R_{yy} \quad R_{yz} \quad R_{zx} \quad R_{zy} \quad R_{zz} \quad \sigma_1 ... \sigma_N$$

where x, y and z indicate the position and R the rotation of the body at time t. The elements σ_i indicate the source density on panel i. σ is used to visualize the flow.

All the files and the hierarchy of the code are shown on Figure 23.

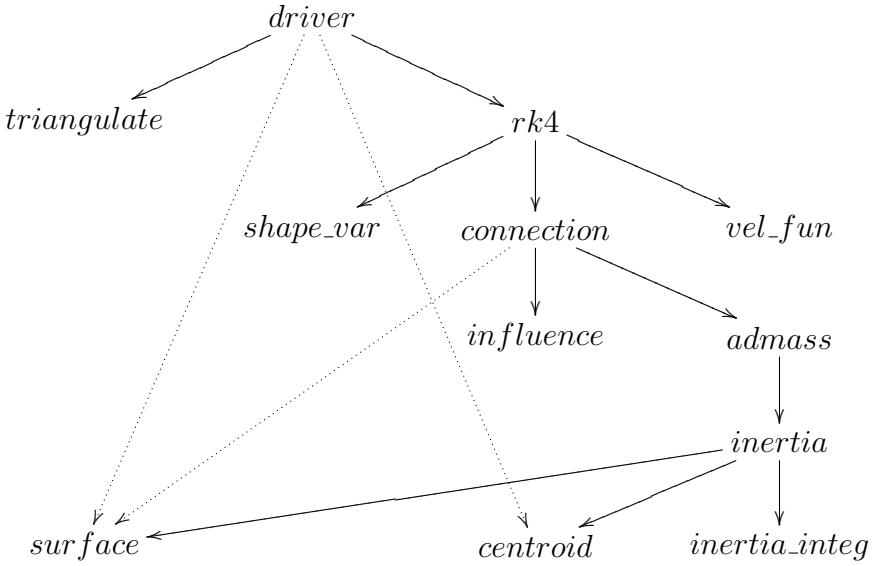


Figure 23: Hierarchy of the MATLAB code

A.1 admass.m

```
function [Id, IdAd] = admass(An, Phi, s, n, zcg, face, verts, del, M, Nf)
% Dzhelil Rufat 31 March 2005
% -----------------INPUT-------------------------------------------------
%
% An(i,j)                  normal velocity induced at Ci due
%                          to a constant source distribution at panel j
%                          Expressed w.r.t a frame attached to the panel i
%
% Phi                      potential function
%
% s                        shape variables
%
% n                        normal vectors to panels
%
% zcg                      position of the control points relative to c.o.m
%
% face, verts              verts is in spherical coordinates
%
% del                      area of panels
%
% M                        mass of body (volume*relative density)
%
% Nf                       number of panels(faces)
%
% -----------------------------------------------------------------------
% -----------------INTERNAL VARIABLES----------------------------------
%
% vfn                    boundary conditions, i.e., normal velocity
%                        of the fluid at the control points
%
% sigma                  source distribution due to a given vfn
%
%-----------------------------------------------------------------------
% -----------------OUTPUT-----------------------------------------------
%
% Id, IdAd               added inertia matrices
%
% -----------------------------------------------------------------------


%calculating boundary conditions

% translations at unit speed in the BODY fixed frame
vel_x = n(:,1);
vel_y = n(:,2);
vel_z = n(:,3);

%angular rotation at unit speed
vel_u = zcg(:,2).*n(:,3) - zcg(:,3).*n(:,2); %around x axis
vel_v = zcg(:,3).*n(:,1) - zcg(:,1).*n(:,3); %around y axis
vel_w = zcg(:,1).*n(:,2) - zcg(:,2).*n(:,1); %around z axis

% shape changes at unit speed
[F, d1F,d2F, d3F] =  surface(verts(:,1), verts(:,2), s);

for i = 1:Nf
    velc_1(i,:) = (d1F(face(i,1),:) + d1F(face(i,2),:) + d1F(face(i,3),:))/3;
    velc_2(i,:) = (d2F(face(i,1),:) + d2F(face(i,2),:) + d2F(face(i,3),:))/3;
    velc_3(i,:) = (d3F(face(i,1),:) + d3F(face(i,2),:) + d3F(face(i,3),:))/3;
end
vel_s1 = velc_1(:,1).*n(:,1) +  velc_1(:,2).*n(:,2) + velc_1(:,3).*n(:,3);
vel_s2 = velc_2(:,1).*n(:,1) +  velc_2(:,2).*n(:,2) + velc_2(:,3).*n(:,3);
vel_s3 = velc_3(:,1).*n(:,1) +  velc_3(:,2).*n(:,2) + velc_3(:,3).*n(:,3);


% Source density distribution
inv_An = inv(An);


%-------------------------------------------------
% make source distributions global to compute field
% see rk4 file
global sigma_x;  global sigma_y;  global sigma_z;
global sigma_u;  global sigma_v;  global sigma_w;
global sigma_s1; global sigma_s2; global sigma_s3;
%-------------------------------------------------

%compute source distributions
sigma_x  = inv_An*vel_x;
sigma_y  = inv_An*vel_y;
sigma_z  = inv_An*vel_z;
sigma_u  = inv_An*vel_u;
sigma_v  = inv_An*vel_v;
sigma_w  = inv_An*vel_w;
sigma_s1 = inv_An*vel_s1;
```

```
sigma_s2 = inv_An*vel_s2;
sigma_s3 = inv_An*vel_s3;


% compute potential functions
phi_x  = Phi*sigma_x;
phi_y  = Phi*sigma_y;
phi_z  = Phi*sigma_z;
phi_u  = Phi*sigma_u;
phi_v  = Phi*sigma_v;
phi_w  = Phi*sigma_w;
phi_s1 = Phi*sigma_s1;
phi_s2 = Phi*sigma_s2;
phi_s3 = Phi*sigma_s3;

% compute the added masses

% BODY fixed frame

m_xx = sum(phi_x.*vel_x.*del);
m_yy = sum(phi_y.*vel_y.*del);
m_zz = sum(phi_z.*vel_z.*del);
m_uu = sum(phi_u.*vel_u.*del);
m_vv = sum(phi_v.*vel_v.*del);
m_ww = sum(phi_w.*vel_w.*del);

m_xy = 0.5*sum((phi_x.*vel_y + phi_y.*vel_x).*del);
m_xz = 0.5*sum((phi_x.*vel_z + phi_z.*vel_x).*del);
m_xu = 0.5*sum((phi_x.*vel_u + phi_u.*vel_x).*del);
m_xv = 0.5*sum((phi_x.*vel_v + phi_v.*vel_x).*del);
m_xw = 0.5*sum((phi_x.*vel_w + phi_w.*vel_x).*del);

m_yz = 0.5*sum((phi_y.*vel_z + phi_y.*vel_z).*del);
m_yu = 0.5*sum((phi_y.*vel_u + phi_y.*vel_u).*del);
m_yv = 0.5*sum((phi_y.*vel_v + phi_y.*vel_v).*del);
m_yw = 0.5*sum((phi_y.*vel_w + phi_y.*vel_w).*del);

m_zu = 0.5*sum((phi_z.*vel_u + phi_u.*vel_z).*del);
m_zv = 0.5*sum((phi_z.*vel_v + phi_v.*vel_z).*del);
m_zw = 0.5*sum((phi_z.*vel_w + phi_w.*vel_z).*del);

m_uv = 0.5*sum((phi_u.*vel_v + phi_v.*vel_u).*del);
m_uw = 0.5*sum((phi_u.*vel_w + phi_w.*vel_u).*del);

m_vw = 0.5*sum((phi_v.*vel_w + phi_w.*vel_v).*del);

m_x1 = 0.5*sum((phi_x.*vel_s1 + phi_s1.*vel_x).*del);
m_y1 = 0.5*sum((phi_y.*vel_s1 + phi_s1.*vel_y).*del);
m_z1 = 0.5*sum((phi_z.*vel_s1 + phi_s1.*vel_z).*del);
m_u1 = 0.5*sum((phi_u.*vel_s1 + phi_s1.*vel_u).*del);
m_v1 = 0.5*sum((phi_v.*vel_s1 + phi_s1.*vel_v).*del);
m_w1 = 0.5*sum((phi_w.*vel_s1 + phi_s1.*vel_w).*del);

m_x2 = 0.5*sum((phi_x.*vel_s2 + phi_s2.*vel_x).*del);
m_y2 = 0.5*sum((phi_y.*vel_s2 + phi_s2.*vel_y).*del);
m_z2 = 0.5*sum((phi_z.*vel_s2 + phi_s2.*vel_z).*del);
m_u2 = 0.5*sum((phi_u.*vel_s2 + phi_s2.*vel_u).*del);
m_v2 = 0.5*sum((phi_v.*vel_s2 + phi_s2.*vel_v).*del);
m_w2 = 0.5*sum((phi_w.*vel_s2 + phi_s2.*vel_w).*del);

m_x3 = 0.5*sum((phi_x.*vel_s3 + phi_s3.*vel_x).*del);
m_y3 = 0.5*sum((phi_y.*vel_s3 + phi_s3.*vel_y).*del);
m_z3 = 0.5*sum((phi_z.*vel_s3 + phi_s3.*vel_z).*del);
m_u3 = 0.5*sum((phi_u.*vel_s3 + phi_s3.*vel_u).*del);
m_v3 = 0.5*sum((phi_v.*vel_s3 + phi_s3.*vel_v).*del);
m_w3 = 0.5*sum((phi_w.*vel_s3 + phi_s3.*vel_w).*del);


% obtain inertia without fluid
[L_gg, L_gs] = inertia(verts, face, Nf, s);

%assign total inertia
Id    = M*L_gg + [m_xx m_xy m_xz m_xu m_xv m_xw; ...
                  m_xy m_yy m_yz m_yu m_yv m_yw; ...
                  m_xz m_yz m_zz m_zu m_zv m_zw; ...
                  m_xu m_yu m_zu m_uu m_uv m_uw; ...
                  m_xv m_yv m_zv m_uv m_vv m_vw; ...
                  m_xw m_yw m_zw m_uw m_vw m_ww; ...
                  ];

IdAd  = M*L_gs + [m_x1, m_x2, m_x3; ...
                  m_y1, m_y2, m_y3; ...
                  m_z1, m_z2, m_z3; ...
                  m_u1, m_u2, m_u3; ...
                  m_v1, m_v2, m_v3; ...
                  m_w1, m_w2, m_w3; ...
                 ];
```

A.2 centroid.m

```
function [cntr, vtot, del, cg, n, t1, t2] = centroid(vert, face, f);
%Dzhelil Rufat, October 28th, 2004
%[cntr, vtot, del, cg, n, t1, t2] = centroid(vert, face, f)
%Calculates motion of centroid
%-----------------INPUT-------------------------------------------------
% vert - vertice coordinates relative to body frame
% face - the face matrix
% f    - number of faces
%-----------------OUTPUT------------------------------------------------
% cntr      coordinates of centoroid relative to body frame
% vtot      volume of body
% del       panel areas
% cg        panel centers
% t1, t2    tangentials to panel surfaces
% n         normal to panel surfaces
%-----------------------------------------------------------------------

for i=1:f,
    TRI=[vert(face(i,1),:);vert(face(i,2),:);vert(face(i,3),:)];

    %collocation points
    cg(i,:)= mean(TRI);
    %side of triangle
    A = TRI(2,:)-TRI(3,:); a=norm(A);
    B = TRI(3,:)-TRI(1,:); b=norm(B);
    C = TRI(1,:)-TRI(2,:); c=norm(C);
    %normal vectors
    N = cross(A,B);
    n(i,:) = N/norm(N);
    %tangential vectors
    t1(i,:) = A/a;
    t2(i,:) = cross(n(i,:),t1(i,:));
    %area
    del(i,:) = 1/4*sqrt((a+(b+c))*(c-(a-b))*(c+(a-b))*(a+(b-c)));
end

CNTR = 3/4*cg;
VOLUME = 1/3*del.*dot(n,cg,2);

vtot = sum(VOLUME);

for i=1:3
    cntr(i) = sum(CNTR(:,i).*VOLUME)/vtot;
end
```

A.3 connection.m

```
function A = connection(s, sdot, M, face, zcg, t1, t2, n, del, Nf, verts, Nv);
% Computes the hydromechanical connection which relates the shape changes
% to the group motion

%vertice coordinates in cartesian coordinates
vertc = surface(verts(:,1),verts(:,2), s);

%compute the influence of panel i to panel j
 [An, Phi] = influence(zcg, face, Nf, vertc, Nv,  n, t1, t2);

% non-dimensional added inertias
[Id, IdAd] = admass(An, Phi, s, n, zcg, face, verts, del, M, Nf);

% connection
  A = inv(Id)*IdAd * sdot';
```

A.4 driver.m

```
function driver(d, P, filename )
% D Rufat, Feb 14, 2005
% d - depth of discretization
% P - number of periods
% filename - name under which to save results in current director

tic
% solve rk4 - fixed time steps

% time span
```

```
h = 2*pi/50;  t = 0:h:2*pi*P;  nbt = length(t);

% initial conditions
g = [ 0 0 0, ...
            1 0 0, ...
            0 1 0, ...
            0 0 1];


% source distribution
%sigma = zeros(nbt, N);

%density of body relative to fluid
density = 1;

% M - mass of body
[verts, face, Nv, Nf] = triangulate(d);
 vertc0 = surface(verts(:,1),verts(:,2), [0,0,0]);
[cntr, vtot, del, zcg0, n, t1, t2] = centroid(vertc0, face, Nf);
M = density*vtot;

% call rk4
for i = 1:nbt-1
    i
    %shape changes
    [s, sdot] =  shape_var(t(i));
    % discretization of body
    vertc = surface(verts(:,1),verts(:,2), s);
    [cntr, vtot, del, zcg, n, t1, t2] = centroid(vertc, face, Nf);

    [g(i+1,:) sigma(i+1,:)]= ...
        rk4(t(i),g(i,:),h,M,face,zcg,t1,t2,n,del,Nf,verts,Nv);
end


%saveresults
data = [t' g sigma];
eval([ 'save ' filename '.txt data -ASCII' ]);
toc
```

A.5 inertia.m

```
function [L_gg, L_gs] = inertia(verts, face, Nf, s);
%[L_gg, L_gs] = inertia(verts, face, Nf, s);
%Dzhelil Rufat, March 15th, 2005
%Calculates moment of inertia with respect to Body frame
%-----------------INPUT-------------------------------------------------
% verts - vertice spherical coordinates [theta, phi]
% face  - face matrix
% Nf     - number of collocation points
% s     - shape variables
%-----------------INTERNAL VARIABLES-----------------------------------
% I   - moment of inertia with respect to Body frame
%-----------------OUTPUT-----------------------------------------------
% L_gg, L_gs - locked inertia matrix of the deformable body in the
% absence of the fluid
%----------------------------------------------------------------------

%The moment of inertia of a tetrahedron
%    I = [Ixx, Ixy, Ixz;
%         Iyx, Iyy, Iyz;
%         Izx, Izy, Izz;]

the = verts(:,1); phi=verts(:,2);
[P, d1P, d2P, d3P] = surface(the, phi, s);

I = [0 0 0 0 0 0];
for i=1:Nf,
    TRI=[P(face(i,1),:);P(face(i,2),:);P(face(i,3),:)];
    %vertices of tetrahedron
    v0=[0, 0, 0];
    v1=TRI(1,:); v2=TRI(2,:); v3=TRI(3,:);
    %evaluate the inertia integral due to each surface panel
    I = I + inertia_integ(v1,v2,v3);
end

[cntr, V] = centroid(P, face, Nf);
I = I/V; %moment of inertia per unit volume

Ixx=I(1);Iyy=I(2);Izz=I(3);Iyz=I(4);Ixz=I(5);Ixy=I(6);

x=cntr(1);y=cntr(2);z=cntr(3);

%calculate iduced velocity of centroid due to changes in shape variables
```

```
h = 1e-6;

%fix s1_dot at 1
CNTR1 = centroid(P + h*d1P, face, Nf);
%fix s2_dot at 1
CNTR2 = centroid(P + h*d2P, face, Nf);
%fix s3_dot at 1
CNTR3 = centroid(P + h*d3P, face, Nf);


%V = xidot + SUM(ai*wi) + SUM(bi*si);

a1=[ 0  -z   y];
a2=[ z   0  -x];
a3=[-y   x   0];

b1 = (CNTR1 - cntr)/h;
b2 = (CNTR2 - cntr)/h;
b3 = (CNTR3 - cntr)/h;

%locked inertia matrix
L_gg =...
[
    1    0    0       a1(1)  a2(1)  a3(1)
    0    1    0       a1(2)  a2(2)  a3(2)
    0    0    1       a1(3)  a2(3)  a3(3)

   a1(1) a1(2) a1(3)  Ixx    Ixy    Ixz
   a2(1) a2(2) a2(3)  Ixy    Iyy    Iyz
   a3(1) a3(2) a3(3)  Ixz    Iyz    Izz
];

L_gs = ...
[
    b1(1)        b2(1)       b3(1)
    b1(2)        b2(2)       b3(2)
    b1(3)        b2(3)       b3(3)

    dot(a1,b1)  dot(a1,b2)   dot(a1,b3)
    dot(a2,b1)  dot(a2,b2)   dot(a2,b3)
    dot(a3,b1)  dot(a3,b2)   dot(a3,b3)
];
```

A.6 inertia_integ.m

```
function Int=inertia_integ(v1,v2,v3)
%evaluate the integrals S (x^2+y^2)z dA  &  S xyz dA
% INPUT
% v1, v2, v3 - vertices
%OUTPUT
% Int: [Ixx Iyy Izz Iyz Ixz Ixy]
%
o=v1; a=v2-v1; b=v3-v1;

axb = cross(a,b);
n = axb/norm(axb);
aa=dot(a,a); bb=dot(b,b); ab=dot(a,b);
oo=dot(o,o); oa=dot(o,a); ob=dot(o,b);
for i=1:3
    A3 = aa*a(i)-a(i)^3;
    A2 = aa*b(i)+2*ab*a(i)-3*a(i)^2*b(i);
    A1 = bb*a(i)+2*ab*b(i)-3*b(i)^2*a(i);
    A0 = bb*b(i)-b(i)^3;

    B2 = (aa-3*a(i)^2)*o(i)+2*oa*a(i);
    B1 = 2*ab*o(i)-6*a(i)*b(i)*o(i)+2*oa*b(i)+2*ob*a(i);
    B0 = (bb-3*b(i)^2)*o(i)+2*ob*b(i);

    C1 = oo*a(i)+2*oa*o(i)-3*a(i)*o(i)^2;
    C0 = oo*b(i)+2*ob*o(i)-3*b(i)*o(i)^2;

    D0 = oo*o(i)-o(i)^3;

    Int_diag(i)= n(i)*norm(axb)*(...
         1/60*(3*A3+A2+A1+3*A0)...
        +1/24*(2*B2+B1+2*B0)...
        +1/6*(C1+C0)...
        +1/2*D0);
end

A3 = a(1)*a(2)*a(3);
A2 = a(1)*a(2)*b(3)+a(1)*b(2)*a(3)+b(1)*a(2)*a(3);
```

```
A1 = b(1)*b(2)*a(3)+b(1)*a(2)*b(3)+a(1)*b(2)*b(3);
A0 = b(1)*b(2)*b(3);

B2 = a(1)*a(2)*o(3)+a(1)*o(2)*a(3)+o(1)*a(2)*a(3);
B1 = (a(1)*b(2)+a(2)*b(1))*o(3)...
    +(a(2)*b(3)+a(3)*b(2))*o(1)...
    +(a(3)*b(1)+a(1)*b(3))*o(2);
B0 = b(1)*b(2)*o(3)+b(1)*o(2)*b(3)+o(1)*b(2)*b(3);

C1 = o(1)*o(2)*a(3)+o(1)*a(2)*o(3)+a(1)*o(2)*o(3);
C0 = o(1)*o(2)*b(3)+o(1)*b(2)*o(3)+b(1)*o(2)*o(3);

D0 = o(1)*o(2)*o(3);

Int_side= - n*norm(axb)*(...
     1/60*(3*A3+A2+A1+3*A0)...
    +1/24*(2*B2+B1+2*B0)...
    +1/6*(C1+C0)...
    +1/2*D0);

%-----DEBUGGING - should be equal to volume
%Int_side(4) = ...
%    n(1)*norm(axb)*[o(1)/2+1/6*(a(1)+b(1))];
%-----

Int = [Int_diag, Int_side];
```

A.7 influence.m

```
function [An, Phi] = influence(zc, face, Nf, vert, Nv,  n, t1, t2)
% ---------------------------------------------
% D Rufat 29/01/2005
% [An,Phi] = influence(zc,t,n,del,N)
% -----------------INPUT----------------------
%
% zc     position of collocation pts
% face   panels
% vert  vertice coordinates
% n      components of outward normal vectors
% N      number of panels
%
% zc, vert and n  are w.r.t inertial frame
%
% -----------------OUTPUT-----------------
%
% An(i,j)                 normal velocities induced at Ci
%                         due to a constant source distribution at panel j
%                         Expressed w.r.t a frame attached to the panel i
%
% Phi(i,j)                potential function iduced by panel j on panel i
%
% ----------------------------------------

for j=1:Nf

%------------TRANSFORM TO A COORDINATE SYTEM ATTACHED TO PANEL J----------

    %translate origin to the collocation point at j
        %find vertices of jth triangle
    vj= [vert(face(j,1),:);
         vert(face(j,2),:);
         vert(face(j,3),:);];
        %translate vertices of triangle
    vj = vj-ones(3,1)*zc(j,:);
        %translate collocation points
    cj = zc-ones(Nf,1)*zc(j,:);

    %rotate to a coordinate sytem of the j-th panel
        %vertices (quantities below are scalars)
    x1=t1(j,1)*vj(1,1)+t1(j,2)*vj(1,2)+t1(j,3)*vj(1,3);
    y1=t2(j,1)*vj(1,1)+t2(j,2)*vj(1,2)+t2(j,3)*vj(1,3);
    x2=t1(j,1)*vj(2,1)+t1(j,2)*vj(2,2)+t1(j,3)*vj(2,3);
    y2=t2(j,1)*vj(2,1)+t2(j,2)*vj(2,2)+t2(j,3)*vj(2,3);
    x3=t1(j,1)*vj(3,1)+t1(j,2)*vj(3,2)+t1(j,3)*vj(3,3);
    y3=t2(j,1)*vj(3,1)+t2(j,2)*vj(3,2)+t2(j,3)*vj(3,3);
         %collocation points (quantities are Nf sized vectors)
    x = t1(j,1)*cj(:,1)+t1(j,2)*cj(:,2)+t1(j,3)*cj(:,3);
    y = t2(j,1)*cj(:,1)+t2(j,2)*cj(:,2)+t2(j,3)*cj(:,3);
    z =  n(j,1)*cj(:,1)+ n(j,2)*cj(:,2)+ n(j,3)*cj(:,3);
        %normal vectors
    nx = t1(j,1)*n(:,1)+t1(j,2)*n(:,2)+t1(j,3)*n(:,3);
    ny = t2(j,1)*n(:,1)+t2(j,2)*n(:,2)+t2(j,3)*n(:,3);
    nz =  n(j,1)*n(:,1)+ n(j,2)*n(:,2)+ n(j,3)*n(:,3);
```

```
    z(j)=z(j)+1e-16;
%--------------CALCULATES THE INFLUENCE IN NEW COORDINATE SYSTEM-----------

    %check if point is inside the triangle
    xa=x2-x1; xb=x3-x1; xp=x-x1;
    ya=y2-y1; yb=y3-y1; yp=y-y1;
    det=xa*yb-xb*ya;
    S = ( yb*xp-xb*yp)/det;
    T = (-ya*xp+xa*yp)/det;
    for i=1:Nf
        s=S(i); t=T(i);
        %points are inside
        if (s>0 && t>0 && s+t<1) delta(i,:)=2*pi;
        %points are outside
        else delta(i,:)=0;
        end;
    end;

    %side lengths
    d12 = sqrt((x2-x1)^2 + (y2-y1)^2);
    d23 = sqrt((x3-x2)^2 + (y3-y2)^2);
    d31 = sqrt((x1-x3)^2 + (y1-y3)^2);

    %side slopes
    C12 = (x2-x1)/d12;  S12=(y2-y1)/d12;
    C23 = (x3-x2)/d23;  S23=(y3-y2)/d23;
    C31 = (x1-x3)/d31;  S31=(y1-y3)/d31;

    %arcs associated with perpendiculars P to each side
    s1_12=(x1-x)*C12+(y1-y)*S12;
    s1_23=(x2-x)*C23+(y2-y)*S23;
    s1_31=(x3-x)*C31+(y3-y)*S31;

    s2_12=(x2-x)*C12+(y2-y)*S12;
    s2_23=(x3-x)*C23+(y3-y)*S23;
    s2_31=(x1-x)*C31+(y1-y)*S31;

    %signed perpendicular distances
    R12 = (x1-x)*S12 -(y1-y)*C12;
    R23 = (x2-x)*S23 -(y2-y)*C23;
    R31 = (x3-x)*S31 -(y3-y)*C31;

    %distances from point P to the corner points of the triangle
    r1 = sqrt((x-x1).^2 + (y-y1).^2 + z.^2);
    r2 = sqrt((x-x2).^2 + (y-y2).^2 + z.^2);
    r3 = sqrt((x-x3).^2 + (y-y3).^2 + z.^2);

    Q12 = log((r1+r2+d12)./(r1+r2-d12));
    Q23 = log((r2+r3+d23)./(r2+r3-d23));
    Q31 = log((r3+r1+d31)./(r3+r1-d31));

    J12 = atan2(R12.*abs(z).*(r1.*s2_12-r2.*s1_12),r1.*r2.*R12.^2+z.^2.*s2_12.*s1_12);
    J23 = atan2(R23.*abs(z).*(r2.*s2_23-r3.*s1_23),r2.*r3.*R23.^2+z.^2.*s2_23.*s1_23);
    J31 = atan2(R31.*abs(z).*(r3.*s2_31-r1.*s1_31),r3.*r1.*R31.^2+z.^2.*s2_31.*s1_31);

    % BUGG found on Feb-15-2005
    %V is defined as  -grad phi and positive rotations are anticlockwise
    phi = +R12.*Q12 + R23.*Q23 + R31.*Q31 + abs(z).*(J12+J23+J31-delta);
    vx  = +S12.*Q12 + S23.*Q23 + S31.*Q31;
    vy  = -C12.*Q12 - C23.*Q23 - C31.*Q31;
    vz  = -sign(z).*(J12+J23+J31-delta);


    %--------------------RESULTS-----------------------------------------

    Phi(:,j)= phi;
    An (:,j)= nx.*vx+ ny.*vy+ nz.*vz;

end
```

A.8 rk4.m

```
function [g1, sigma] = rk4(t0,g0,h,M,face,zcg,t1,t2,n,del,Nf,verts,Nv)
% D Rufat, February 16, 2005
% Fourth order fixed time step Runge-Kutta scheme to advance motion in
% time.

% shape variables
[s_0, sdot_0] = shape_var(t0);
[s_1, sdot_1] = shape_var(t0+h/2);
[s_2, sdot_2] = shape_var(t0+h);
```

```
% connection
A0 = connection(s_0, sdot_0, M, face, zcg, t1, t2, n, del, Nf, verts, Nv);
A1 = connection(s_1, sdot_1, M, face, zcg, t1, t2, n, del, Nf, verts, Nv);
A2 = connection(s_2, sdot_2, M, face, zcg, t1, t2, n, del, Nf, verts, Nv);


K1 = vel_fun(A0,g0);
K2 = vel_fun(A1,g0 + h*K1/2);
K3 = vel_fun(A1,g0 + h*K2/2);
K4 = vel_fun(A2,g0 + h*K3);

g1 = g0 + h*(K1+2*K2+2*K3+K4)/6;

%-------------import source distributions----------------
global sigma_x;  global sigma_y;  global sigma_z;
global sigma_u;  global sigma_v;  global sigma_w;
global sigma_s1; global sigma_s2; global sigma_s3;
sigma = (0*sigma_x*A2(1) +  0*sigma_y*A2(2) +  0*sigma_z*A2(3) + ...
         0*sigma_u*A2(4) +  0*sigma_v*A2(5) +  0*sigma_w*A2(6) +...
         sigma_s1*sdot_2(1) + sigma_s2*sdot_2(2) + sigma_s3*sdot_2(3))';
```

A.9 shape_var.m

```
function [s, sdot] = shape_var(t)
% Prescribe shape changes
% ---------------INPUT-----------
%   t  - time
% --------------OUTPUT------------
%   s   = [s1, s2, s3] - shape variables
%   sdot= [s1dot, s2dot, s3dot] - time derivatives
%---------------------------------


s1 = sin(t);
s2 = cos(t);
s3 = 0*t;

s1dot = cos(t);
s2dot = -sin(t);
s3dot =  0*t;

s    = [s1,    s2,    s3];
sdot = [s1dot, s2dot, s3dot];
```

A.10 surface.m

```
function [P, d1P, d2P, d3P] = surface(theta, phi, s)
% Prescribe shape changes
% [F, d1F, d2F, d3F] = surface(theta, phi, s1, s2, s3)
% ---------------INPUT-----------
%   s1, s2,s3 - shape variables
%   theta, phi - angle parameter in polar coordinates fixed to center
%                of body frame
% --------------OUTPUT------------
%   P    - coordinates in Body Frame
%   d1P  - derivative with respect to 1st mode variable
%   d2P  - derivative with respect to 2nd mode variable
%   d3P  - derivative with respect to 3rd mode variable
%---------------------------------
s1=s(:,1);s2=s(:,2);s3=s(:,3);
%---------------------------------


%MODES:
F0 = 1;
F1 = 0.1*s1*cos(2*phi);
F2 = 0.1*s2*cos(3*phi);
F3 = 0;

%SHAPE
F = F0 + F1 + F2 + F3;

%DERIVATIVES
d1F = 0.1*cos(2*phi);
d2F = 0.1*cos(3*phi);
d3F = 0;

%-----------------------
[  P(:,1),  P(:,2),  P(:,3)]=sph2cart(theta,pi/2-phi,   F);
```

```
[d1P(:,1),d1P(:,2),d1P(:,3)]=sph2cart(theta,pi/2-phi, d1F);
[d2P(:,1),d2P(:,2),d2P(:,3)]=sph2cart(theta,pi/2-phi, d2F);
[d3P(:,1),d3P(:,2),d3P(:,3)]=sph2cart(theta,pi/2-phi, d3F);
%----------------------
```

A.11 triangulate.m

```
function [vert, face, v, f] = triangulate(d)
% ------------------------------------------------------------
% D Rufat, 30 March 2005
% -----------------INPUT--------------------------------
%
% d    depth of discretization
%
% -----------------OUTPUT-------------------------------
% vert - coordinates of vertices in spherical [the, phi]
% face - faces
% v    - number of vertices
% f    - number of faces
% ------------------------------------------------------------

%icosahedron
GoldRatio = (1+sqrt(5))/2; PHI=2*atan(1/GoldRatio);
for i = 1:10,
    if rem(i,2)==1 phi=PHI;
    else phi=pi-PHI;
    end
    vert(i+2,:) = [(i-1)*pi/5, phi];
end
vert(1,:) = [0, 0];
vert(2,:) = [0, pi];

for i=1:4
    face(4*i-3,:)=[2*i+1, 2*i+2, 2*i+3];
    face(4*i-2,:)=[2*i+2, 2*i+4, 2*i+3];
    face(4*i-1,:)=[1,2*i+1,2*i+3];
    face(4*i-0,:)=[2,2*i+4,2*i+2];
end
face = [face;
        [11, 12,  3];
        [3,  12,  4];
        [1,  11,  3];
        [2,   4, 12];
        ];
f=20; v=12;


%octahedron
%for i = 1:4,
%    vert(i+2,:) = [(i-1)*pi/2, pi/2];
%end
%vert(1,:) = [0, 0];
%vert(2,:) = [0, pi];
%face = [...
%    1 3 4;
%    1 4 5;
%    1 5 6;
%    1 6 3;
%    2 4 3;
%    2 5 4;
%    2 6 5;
%    2 3 6;
%    ];
%f=8; v=6;


for depth=1:d,
    N=f;
    for i=1:N,
        TRI=[vert(face(i,1),:);vert(face(i,2),:);vert(face(i,3),:)];

        TRI(:,2)=pi/2-TRI(:,2);
        [A(1),A(2),A(3)] = sph2cart(TRI(1,1),TRI(1,2),1);
        [B(1),B(2),B(3)] = sph2cart(TRI(2,1),TRI(2,2),1);
        [C(1),C(2),C(3)] = sph2cart(TRI(3,1),TRI(3,2),1);
        c1 = B+C; [p1(1), p1(2),r]=cart2sph(c1(1),c1(2),c1(3));
        c2 = C+A; [p2(1), p2(2),r]=cart2sph(c2(1),c2(2),c2(3));
        c3 = A+B; [p3(1), p3(2),r]=cart2sph(c3(1),c3(2),c3(3));
        TRI(:,2)=pi/2-TRI(:,2);
        p1(2)=pi/2-p1(2);p2(2)=pi/2-p2(2);p3(2)=pi/2-p3(2);
        %
        if face(i,1)==1 | face(i,1)==2
            p2(1) = TRI(3,1);
```

```
            p3(1) = TRI(2,1);
        end
        vert(v+1,:)=p1;
        vert(v+2,:)=p2;
        vert(v+3,:)=p3;

        temp(4*i-3,:)= [face(i,1), v+3, v+2];
        temp(4*i-2,:)= [face(i,2), v+1, v+3];
        temp(4*i-1,:)= [face(i,3), v+2, v+1];
        temp(4*i-0,:)= [v+1, v+2, v+3];
        v = v+3; f=f+3;
    end
    face = temp;
end
```

A.12 vel_fun.m

```
function gdot = vel_fun(A,g)
%compute velocity gdot given connection A and current position g
% g = ( x,  y, z, R11, R12, R13, R21, R22, R23, R31, R32, R33)
% A = (Ax, Ay, Az, Au, Av, Aw) (u,v,w angular rotation around x,y,z)


R = [ ...
    g(4),  g(5),  g(6);
    g(7),  g(8),  g(9);
    g(10), g(11), g(12);
    ];

wx=A(4); wy=A(5); wz=A(6);

O = [...
    0  -wz   wy;
   wz    0  -wx;
  -wy   wx    0];

%derivative of R
D = R*O;

% differential equations
gdot = - [A(1), ...
          A(2), ...
          A(3), ...
          D(1,1), ...
          D(1,2), ...
          D(1,3), ...
          D(2,1), ...
          D(2,2), ...
          D(2,3), ...
          D(3,1), ...
          D(3,2), ...
          D(3,3)];
```

References

[1] J.L. Hess and M. O. Smith. Calculation of potential flow about arbitrary bodies. *Progress in Aeronautical Sciences*, 8:1–59, 1967.

[2] J.Melli-Huber, E.Kanso, and C.Rowley. Numerical model for fish-like locomotion. *1000 Islands Fluid Mechanics Meeting*, 2004.

[3] R Mason and J. Burdick. Propulsion and control of deformable bodies in an ideal fluid. *Proceedings. 1999 IEEE International Conference on Robotics and Automation*, 1:773–780, 1999.