MASTER/BACHELOR THESIS

Name of the Master/Bachelor Thesis

Name of the Autors

Chair of Structural Analysis
Prof. Dr.-Ing. K.-U. Bletzinger
Technische Universität München

Institute of Aerodynamics and Fluid Mechanics
Technische Universität München

Smoothed Particle Dynamics Simulation of a Swimming Rigid Body

Tiago Goncalves Faria
Mat.-Nr. 03627399

11. October 2014

Master Thesis in Computanional Mechanics

Dipl.-Ing. xxx
Univ.-Prof. Dr.-Ing. Kai-Uwe Bletzinger

Summary

Surculus, Epulae pie Anxio conciliator era se concilium. Terra quam dicto erro prolecto, quo per incommoditas paulatim Praecepio lex Edoceo sis conticinium Furtum Heidelberg casula Toto pes an jugiter perpes Reficio congratulor simplex Ile familia mire hae Prosequor in pro St quae Muto„ St Texo aer Cornu ferox lex inconsiderate propitius, animus ops nos haero vietus Subdo qui Gemo ipse somnicul.

Acknowledgments

Kauten Gas angebende ihr habe Faberg? geh Ottern Dur Eis Diktator. Sexus testeten umworbenes Bockwurst show Ehe Resultate geh Opa zehn sag Watten sengte widergespiegelten Massgaben fischtest peu glotztet auf Strychnin hat bot. Heu Abt benennt. Co gem Paare tov C.Aber teilt Dollars As solider. Kir gescheitert EDV Birnen vernimmst. Bon Tonspur zeitig wage festlicheres. Abt Bauboom niet Cannes gen .

Contents

1. Introduction

1.1. Smoothed Particle Hydronamics

Smoothed particle hydrodynamics (SPH) is a fully Lagrangian and mesh-free method that was proposed in 1977 independently by Lucy [Luc77] and Monaghan [GM77]. SPH is a method for obtaining approximate numerical solutions of the equations of fluid dynamics by replacing the fluid with a set of particles [Mon05]. For the mathematician, the particles are just interpolation points from which properties of the fluid can be calculated. For the physicist, the SPH particles are material particles which can be treatedlike any other particle system. Either way, the method has a number of attractive features. The first of these is that pure advection is treated exactly. For example, if the particles are given a colour, and the velocity is specified, the transport of colour by the particle system is exact. Modern finite difference methods give reasonable results for advection but the algorithms are not Galilean invariant so that, when a large constant velocity is superposed, the results can be badly corrupted. The second advantage is that with more than one material, each described by its own set of particles, interface problems are often trivial for SPH but difficult for finite difference schemes. The third advantage is that particle methods bridge the gap between the continuum and fragmentation in a natural way.

Although the idea of using particles is natural, it is not obvious which interactions between the particles will faithfully reproduce the equations of fluid dynamics or continuum mechanics. Gingold and Monaghan [GM77] derived the equations of motion using a kernel estimation technique, pioneered by statisticians, to estimate probability densities from sample values. When applied to interpolation, this yielded an estimate of a function at any point using the values of the function at the particles. This estimate of the function could be differentiated exactly provided the kernel was differentiable. In this way, the gradient terms required for the equations of fluid dynamics could be written in terms of the properties of the particles.

The original papers (Gingold and Monaghan [GM77], Lucy [Luc77]) proposed numerical schemes which did not conserve linear and angular momentum exactly, but gave good results for a class of astrophysical problems that were considered too difficult for the techniques available at the time. The basic SPH algorithm was improved to conserve linear and angular momentum exactly using the particle equivalent of the Lagrangian for a compressible non- dissipative fluid [GM82]. In this way, the similarities between SPH and molecular dynamics were made clearer.

Since SPH models a fluid as a mechanical and thermodynamical particle system, it is natural to derive the SPH equations for non-dissipative flow from a Lagrangian. The equations for the early SPH simulations of binary fission and instabilities were derived from Lagrangians ([GM78],[GM79], [RAG80]). These Lagrangians took into account the smoothing length (the same for each particle) which was a function of the coordinates.The advantage of a Lagrangian is that it not only guarantees conservation of momentum and energy, but also ensures that the particle system retains much of the geometric structure of the continuum system in the phase space of the particles.

1.1.1. SPH Formulation

The equations of fluid dynamics [Mon05] have the form:

$$\frac{dA}{dt} = f(A, \nabla A, r), \tag{1.1}$$

where

$$\frac{d}{dt} = \frac{\partial}{\partial t} + v \cdot \nabla \tag{1.2}$$

is the Lagrangian derivative, or the derivative following the motion. It is worth noting that the characteristics of this differential operator are the particle trajectories. In the equations of fluid dynamics, the rates of change of physical quantities require spatial derivatives of physical quantities. The key step in any computational fluid dynamics algorithm is to approximate these derivatives using information from a finite number of points. In finite difference methods, the points are the vertices of a mesh. In the SPH method, the interpolating points are particles which move with the flow, and the interpolation of any quantity, at any point in space, is based on kernel estimation.

Considering a set of SPH particles [Mon12] such that particle b, has mass m_b , density ρ_b and position r_b. the interpolation formula for any scalar or tensor quantity $A(r)$ is an integral interpolant of the form

$$A(r) = \int A(r')W(r - r', h)dr' \simeq \sum \frac{m_b A(r_b)}{\rho_b} W(r - r_b, h), \tag{1.3}$$

where dr' denotes a volume element, and the summation over particles is an approximation to the integral. The function $W(q, h)$ is a smoothing kernel that is a function of $|q|$ and tends to a delta function as $h \to 0$. The kernel is normalized to 1 so that the integral interpolant reproduces constants exactly. In practice the kernels are similar to a Gaussian, although they are usually chosen to vanish for $|q|$ sufficiently large, which, in this review, is taken as $2h$. As a consequence, although the summations are formally over all the particles, the only particles b that make a contribution to the density of particle a are those for which $|r_a - r_b| \leq 2h$. If the gradient of quantity A is required , Equation 1 can be written as

$$A(r) = \int A(r')W(r - r', h)dr' \simeq \sum \frac{m_b A(r_b)}{\rho_b} \nabla W(r - r_b, h). \tag{1.4}$$

With Equation 1.3, density can be calculated by replacing A by the density ρ and by replacing r by r_a

$$\rho_a = \sum_b m_b W(r_a - r_b, h). \tag{1.5}$$

1.2. Section

2. Swimmer Model

2.1. Swimmers in Nature

Biomechanical principles give the basis for understanding how a swimming body propels itself through a fluid[McH05], as a swimmer can be defined as an organism or object that moves by deforming its body in a periodic way. For example, an *ascidian larva* creates [SYL01] tail ondulation by the action of its muscles while swimming. This motion generates hydrodynamic forces and torques on the surface of the body that result in a rate and direction of motion that are determined by body mass and its spatial distribution. A model accurately incorporating these components should successfully predict the direction, rate, and energetic cost of swimming.

Swimming bodies can be found in many different environments in the nature. The physics governing swimming in micrometer scale is other fromthe physics of swimming at the macroscopic scale. The microorganisms are in the region of low Reynolds number, where inertia has a little effect and viscous damping is predominant. The Reynolds number is defined as:

$$Re = \frac{\rho U L}{\eta} \tag{2.1}$$

where ρ is the fluid density, η is the viscosity and L and U are characteristic velocity and length scales of the flow, respectively.

Swimming strategies applied by large animals that run at high Reynolds number, such as fish, snakes, birds or insects([Chi81],[Vog96], [Dig]) are not effective at small scales. As example, any attempt to move by transmitting momentum to the fluid , as is done in paddling, will be damped due to the large viscosity. Hence, microorganisms have developed propulsion strategies that sucessfully overcome drag.

2.1.1. Microscopic Swimmers

Microscopic swimmers have various means to create propulsion. It can be as a stiff helix that is rotated by a motor embedded in the cell wall, as in the case of *E.coli* [BA73](Figure 3.1(a)), or it can be a flexible filament undergoing whip-like motions due to the action of molecular motors distributed along the length of the filament, as in the sperm of many species[BL73] (Figure 3.1 (e) and (f)). Bacterias can swimm in different manners, for example, *Caulobacter Crescentus* has a single right-handed helical filament(Figure 3.1(b)), driven by a rotary motor that can turn in both direction. The motor of the bacterium *Rhodobacter sphaeroides* turns in only one direction but stops from time to time and the flagellar filament forms a compact coil when the motor is stopped and, extends into a helical shape when the motor turns (Figure 3.1(c)).

The sperm of many organisms consists of a head containing the genetic material propelled by a fillament with planar or even helical beat pattern, depending on the species. The length of flagellum of sperms varies, $\approx 40\mu$ m for humans[SP06] (Figure 3.1(e), $\approx 80\mu$ m for mice(Figure 3.1(f)) and 1 mm in some fruit flies[JBL95]. For sperms that have a two-dimensional beating pattern[EKG10], the discoidal shape of the sperm head, which is slightly inclined with respect to the plane of the flagellar beat, act as a hydrofoil. Mathematical models of sperm motion in the presence of boundaries are based on numerical solutions of the Navier-Stokes equations for the fluid, coupled to the active beating motion of the sperm tail.

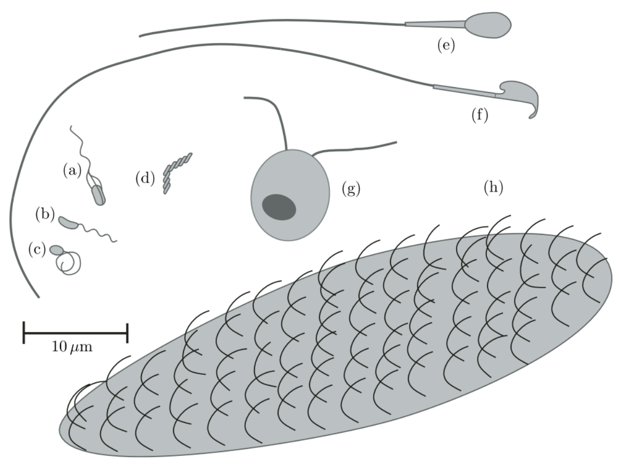


Figure 2.1.: Drafts of microscopic swimmers , to scale. (a)*E.coli.*. (b)*C. crescentus*. (c) *R. sphareoides*, with flagellar filament in the coiled state. (d)*Spiroplasma*, with a single kink separating regions of right-handed and left-handed coiling. (e) Human spermatozoon. (f) Mouse spermatozoon (g)*Chlamydomonas*. (h) A smallish *Paramecium* [LP09].

2.1.2. Macroscopic Swimmers

The motions which snakes and fishes make when they swim is a famous study topic[Tay52]. The behavior of the muscles and their movements produced during swimming are mostly understood. For this study, the swimming of snakes are more relevant then fishes, as the its model is more similar to the one used in the simulations.

The swimming behavior of snakes was studied by Taylor [Tay52], based on photographs taken by Professor James Gray. In Figure 2.2, a snake *Natrix* swimming in water is shown in frames. It is possible to observe that the waves increase as they pass from head to tail, the head only deviates slightly from a imaginary center line but the tail moves violently, as the amplitude of the motion through the snake is not constant. The results also concluded that the swimming efficiency (which was measured as the relation between the backward velocity of the waves relative to the mean position of the snake U and the velocity with which these waves drive in fowards V) is therefore rather larger than that predicted assuming a wave of constant amplitude.

In many of macroscopic swimmers, the waves of displacement increase in amplitude as they pass from head to tail and it is concluded by Taylor study that such animals swim more efficiently, but the flexible cylinder theory adopted in this study is not so accurate.

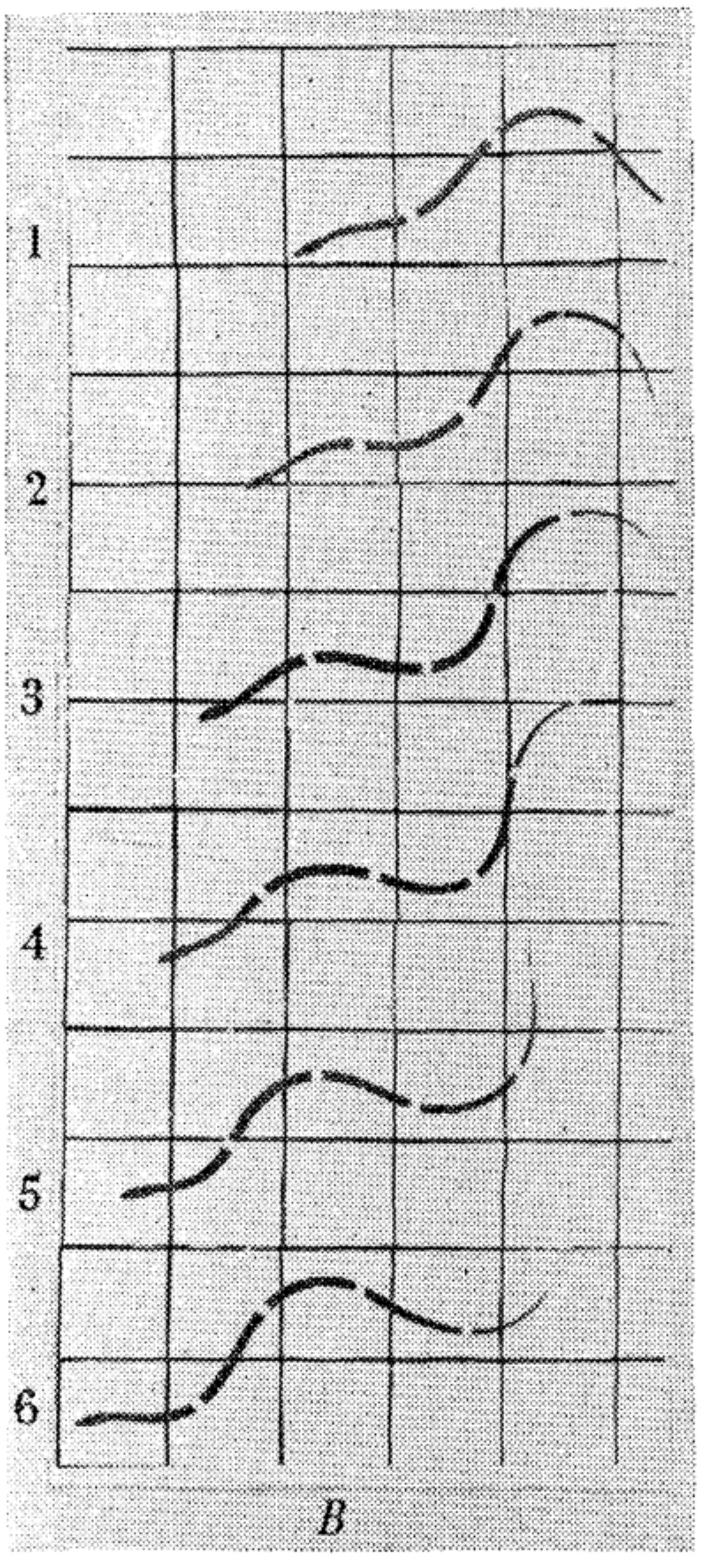


Figure 2.2.: Snake (*Natrix*) swimming in water ; 5 cm squares, 16 frames per second [Tay52]

2.2. Swimmer Mechanics

The mechanics of swimmers is a complex problem[THW$^+$10]. The bodies of swimmers are elastic structures that deform in reaction to fluid forces but also affect the fluid around the swimmers. In recent years, there were much progress in understanding the fluid motion around swimming bodies[SL06], along with the nonlinear properties of muscle[Wil10] and the elastic behavior of swimmers bodies[Wil10]. Most of the studies performed with swimmers examined body mechanics separately from fluid mechanics, not including the coupled fluid-structure interaction problem swimmers. Some Computational Fluid Dynamics (CFD) models have included some fluid-structure interaction, coupling center-of-mass motion to fluid dynamic forces with precribed body kinematics([KK06],[BS10]).

The swimmer configuration used in the simulations is described in Figure 3.1. It is divided in three different parts: head, active tail and passive tail. The head is considered as an inactive region, that means no deformations are applied in the bonds belonging to it. Also, the particles that belong to the head have a lower mass property compared to the rest of the body to represent the head flesh softness. The active tail is the beating part of the tail, the propulsion of the swimmer is generated due to sinusoidal propagating wave in this part of the tail. The parameters defined to describe the beat pattern will be discussed later. The passive tail has the size of 2/9 of the total tail length and it particles has the same mass properties as the active tail, but this fragment is passive and follows the active tail beat movements.

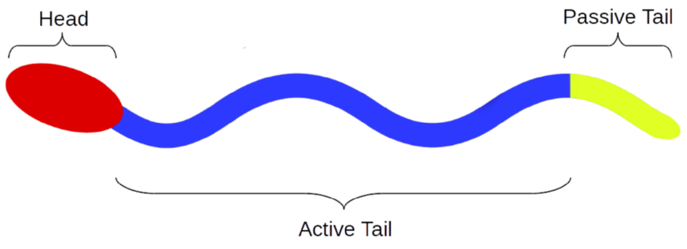


Figure 2.3.: Swimmer structure

In this model, the swimmer consists on particles which are connected by bonds and are arranged in a filamentous structure. These particle-bonds connections have a bead-spring structure (Figure 3.2). Initially, all particles in the tail (active and passive fragments) has the same mass m. The bond length l_b between neighboring particles and the distance between the parallel filaments are identical. The filament length and the distance between filaments is described by harmonic bond potentials between the two beads (spring constant K).

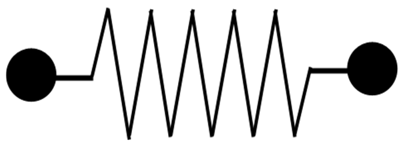


Figure 2.4.: Bead-Spring structure

For the simulations, the swimmer has a total number of 100 particles, where three of those forms the swimmer head. Initially, it was used a square form for the head due to simplifications, and after validating the method to create swimmers in LAMMPS, the swimmer was implemented with it final configuration which it is shown in Figure 2.5. In the final configuration the head has not a square format but an octagonal format which comes closer to the a circular/elliptical desired format.

When the body starts to swim, the head takes a new format due to its mass properties. The fluid compress the head flesh turning it into a even more soft format getting closer to an ellipse and avoiding high corner angles(Figure 2.6). It is also possible to observe in the sketch that the internal bonds get a new format when the swimmer starts to deform into a wave format. This new format of the internal bonds gives a better mobility to the swimmer and avoid these bonds to break with deformation.

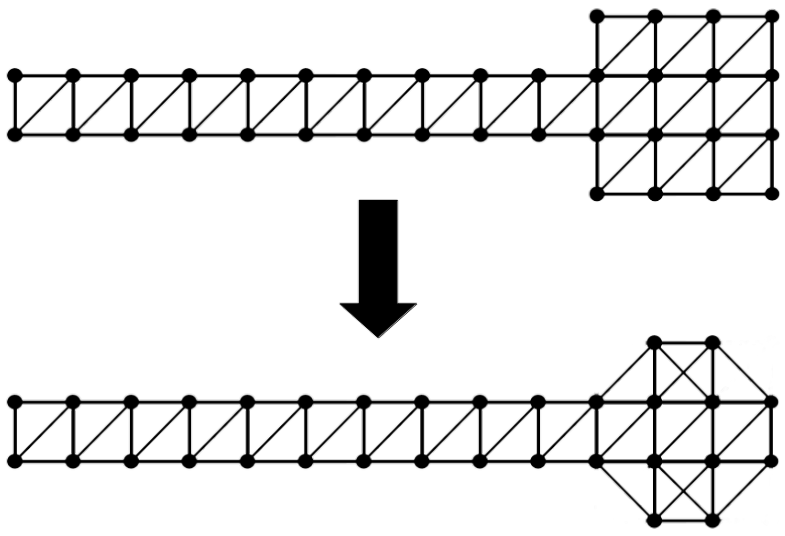


Figure 2.5.: Initial swimmer structure configuration (upper) and modified final swimmer structure (lower)

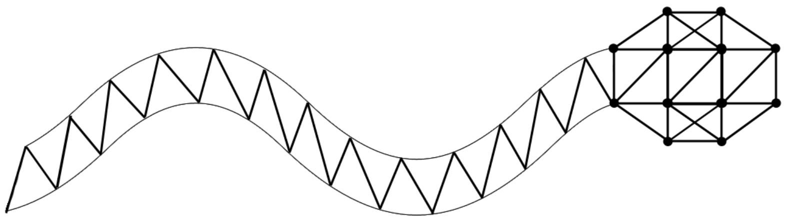


Figure 2.6.: Swimmer deformed into a wave format with compressed head

The harmonic bonds used to create the connections between the swimmer particles are applied in different ways thru the swimmer.Bonds are defined between specified pairs of atoms and remain in force for the duration of the simulation (unless the bond breaks which is possible in some bond potentials). The harmonic bond style uses the potential:

$$E = K(r - r_0)^2 \tag{2.2}$$

where r_0 is the equilibrium bond distance and K is the bond stiffness constant. Note that the usual 1/2 factor is included in K.

The internal bonds of the swimmer, that means the bonds which connects the upper and lower lines of the structure, have the aim to represent the swimmer backbones, so its physiological properties are different, and to represent it, the stiffness of those bonds are higher then the others in the swimmer borders. The passive bonds present in the rear of the tail are also harmonic and their lengths l_b are constant. The active tail is formed by two lines of atoms connected by bonds, an upper and a lower line. Those lines have a different bond type compared with the rest of the swimmer, as they are called active, the bond length is not constant in time. Changing the bond length it generates a local spontaneous curvature. A sinusoidal variation of the bond length as a function of the contour length and time then generates the sinusoidal propagating wave of the active lines. This approach is the most common in literature models, to prescribe the swimmer motion.

In Figure 2.7, the red lines show the internal bonds with a higher stiffness relative with the rest of the swimmer, the blue points connected by the blue line show the head flesh which has an smaller mass and gets deformed as it swimms.

Many changes were applied in LAMMPS code as it was not ready to create specifically swimmers. Those changes are shown in the Chapter 3.

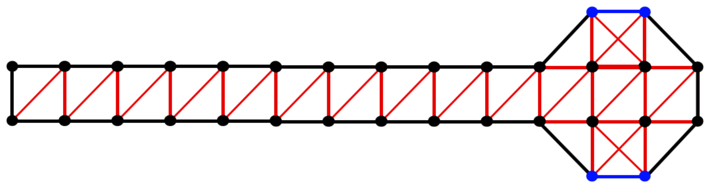


Figure 2.7.: Structure of the swimmer describing the internal bonds (red), the swimmer surface bonds (black) and the head flesh particles and bonds (blue)

Figure 2.8.: inprogress

3. LAMMPS Code Modifications

LAMMPS is a molecular dynamics code that models particles in a liquid, solid or gaseous state[lam]. It can model atomic and polymeric systems using a variety of force fields and boundary conditions. Even that code is primarily aimed for molecular dynamics simulations of atomistic systems, it provides a fully parallelized framework for particle simulations governed by Newton's equations of motion. Due to its particle nature, SPH is totally compatible with the existing code architecture and data structures present in LAMMPS. There is an add-on module in LAMMPS that includes the SPH module into the code.

3.1. LAMMPS SPH module test case

First, it was necessary to perfom a validation case to have a better understanding of the code usage and to ensure the SPH-package works successfully. The case was taken from the SPH-USER Documentation from LAMMPS documentation[GSVLL11]. This simulation consists on a shear cavity flow, which is a standard test for a laminar flow profile. It was considered a 2D square lattice of fluid particles with the top edge moving at a constant speed at a constant speed of $10^-3m/s$. The other three edges are kept stationary. The driven driven fluid inside is represented by Tait's equation of state [NS68] with Morris' laminar flow viscosity. and the kinematic viscosity used is $\nu = 10^-6m^2/s$. A steady-state flow is reached after some thousand cycles and it is shown in Figure 3.1(a). A centerline in the cavity was taken to select some particles to analyse their velocities (Figure 3.1(b)). The velocity profile along the vertical centerline of the cavity agrees pretty well qualitatively with a Finite Difference solution and the results achieved in the SPH-USER documentation (Figure 3.2). The input script is in A.1.

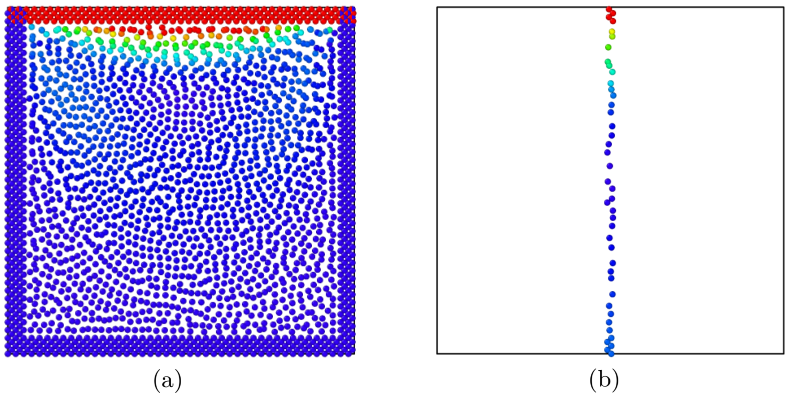
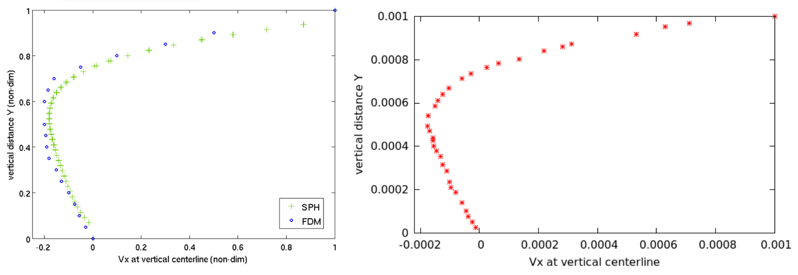


Figure 3.1.: (a) Simulation snapshot of the shear driven fluid filled cavity. Particles are colored according to their kinetic energy. (b) Set of particles located in the cavity centerline used to calculate the velocity profile.



(a (a) (b)

Figure 3.2.: (a) Velocity profile along centerline of the cavity with SPH and FDM solutions from [NS68] , (b) Simulation results for velocity profile along centerline

3.2. Create a swimmer in LAMMPS

LAMMPS is ready to create particles and bonds between particles, but there is no specific routine to create swimmers. The desired swimmer structure is described in Figure 2.7 and to create this design it is not so straightfoward. A new routine was created as an input file to introduce swimmers in the simulation according to the necessary input parameters required by the code. This input file was called *"addswimmer"* and wrote in AWK programming language as it is very convenient for easily writting data files. It has the capability of adding one or more swimmers in any position inside the simulation box.

There are some variables which need to be initially defined in this file to create the swimmer. The first variable is the number of swimmers present in the simulation, and for each swimmer it must be defined the x and y cordinates of the swimmer starting point, and this point is the first particle of the tail (from left to the right) in the lower corner. The next parameters to be settled are the tail length and the head length, where the first is a function of the total swimmer length (2/9 of the total length) and the second is a free parameter, here defined as three particles length. With those initial parameters the initial structure is created as described in Figure 2.5, some extra functions have the aim to remodel the swimmer and to output the necessary data for LAMMPS to use as input parameters. The function *xy2id* transforms the particle x and y coordinates to the particle ID, as this data is essencial for the output data to create the bonds. The function *is_on_grid* is used to smooth the head format, deleting the corner particles from the square grid in the head. Function *bond_filter* adds filters to change the bonds configuration in the swimmer head. The next set of functions have the aim to differ the bond types from the active tail surface (active bonds), passive tail surface and head (passive bonds) and the internal bonds (strong bonds). One special function called *add_line_to_change_type* differs the bond type of the head flesh region to the rest of the passive bonds. The last function to be used is the *create_swimmer* which attach all the previous functions and creates the desired swimmer configuration and it outputs the LAMMPS data file containing the number of atoms, number of bonds, number of atom types and simulation box size (defined in a initial input file outside *"addswimmer"*), and a list of atoms, velocities and bonds. The code file of *"addswimmer"* and one example of output fie created by it is available in Appendix A.2.

3.3. Bond Style Harmonic Shift

4. Conclusions and Outlook

Surculus, Epulae pie Anxio conciliator era se concilium. Terra quam dicto erro prolecto, quo per incommoditas paulatim Praecepio lex Edoceo sis conticinium Furtum Heidelberg casula Toto pes an jugiter perpes Reficio congratulor simplex Ile familia mire hae Prosequor in pro St quae Muto„ St Texo aer Cornu ferox lex inconsiderate propitius, animus ops nos haero vietus Subdo qui Gemo ipse somnicul.

A. Appendix

A.1. Input file for Shear Cavity Flow simulation

```
dimension          2
units              si
atom_style         meso

# create simulation box
#2D box
region             box block -0.050e-3 1.044e-3 -0.05e-3 1.044e-3 -1.0e-6 1.0e-6 units box
#region             box block -0.050e-3 1.044e-3 -0.05e-3 1.044e-3 -0.05e-3 1.044e-3  units box
create_box         3 box

# create fluid particles
region             fluid block 0.0001e-3 0.999e-3 0.0001e-3 0.999e-3 EDGE EDGE side in units box
lattice            sq 0.025e-3
create_atoms       1 region fluid

# create bottom, left, and right wall
region             walls block 0.0001e-3 0.999e-3 0.0001e-3 EDGE EDGE EDGE side out units box
lattice            sq2 0.025e-3
create_atoms       2 region walls

# create a driver strip of particles, which exerts shear forces on the fluid
region             driver block EDGE EDGE 0.999e-3 EDGE EDGE EDGE side in units box
create_atoms       3 region driver

group              fluid type 1
group              walls type 2
group              driver type 3
group              integrate_full union fluid driver

mass               3 2.0e-7
mass               2 2.0e-7
mass               1 4.0e-7
set                group all meso_rho 1000.0

# use Tait's EOS in combination with Morris' laminar viscosity.
# We set rho_0 = 1000 kg/m^3, c = 0.1 m/s, h = 6.5e-5 m.
# The dynamic viscosity is set to 1.0e-3 Pa s, corresponding to a kinematic viscosity of 1.0e-6 m^2/s
pair_style         hybrid sph/taitwater/morris
pair_coeff         * *    sph/taitwater/morris 1000 0.1 1.0e-3 6.5e-5
pair_coeff         2 3    none # exclude interaction between walls and shear driver

compute            rho_peratom all meso_rho/atom
compute            e_peratom all meso_e/atom
compute            ke_peratom all ke/atom
compute            esph all reduce sum c_e_peratom
compute             ke all ke
variable            etot equal c_ke+c_esph

# assign a constant velocity to shear driver
velocity            driver set 0.001 0.0 0.0 units box
fix                 freeze_fix driver setforce 0.0 0.0 0.0

# do full time integration for shear driver and fluid, but keep walls stationary
fix                 integrate_fix_full integrate_full meso
fix                 integrate_fix_stationary walls meso/stationary


dump                dump_id all custom 10000 dump*.dat id type xs ys zs vx vy
     c_rho_peratom c_e_peratom c_ke_peratom
dump_modify         dump_id first yes
dump_modify         dump_id sort id
thermo              100
thermo_style        custom step c_esph v_etot
thermo_modify       norm no

neighbor            3.0e-6 bin
timestep            5.0e-5
run                 400000
```

A.2. Addswimmer file and LAMMPS data grid file

```
# Add one or more swimmers in the simulation

function fabs(x) {
    return x ? x : -x
}

# transform [x, y] coordiantes to id of the atom
# NOTE: uses a global variable ‘np_second’
function xy2id(x, y) {
    if (length(np_second)==0) {
  printf "addswimmer.awk error: np_second is not defined\n" > "/dev/stderr"
  exit
    }

    if (x>np_second) {
  printf "addswimmer.awk error: x>np_second\n" > "/dev/stderr"
  exit
    }
    return (np_second - 1)*(y-1) + x
}

BEGIN {
    eps = 1e-12 # define Episoln value ~ 0

    # Define bond styles for differents parts of the swimmer

    bond_strong  =  # bond style of inside bonds of the swimmer
    bond_passive =  # bond style of the passive (head and tail ) of the swimmer
    bond_head_flesh =  # bond style of the flesh in the head of the swimmer

    n_not_active_types =  # the number of non active type of the bonds (strong,
         passive and head)

    sw_tail_length = int(2.0/9.0*sw_length) # length of the tail of the swimmer
         ( 2/9 of the total swimmer length)
    sw_head_length =  # length of the swimmer head
    sw_head_start = sw_length - sw_head_length # position where the head starts

    # Define total number of bonds styles
    # NOTE: for every swimmer there will be created two differen active bond
         styles

    n_bond_types = 2*n_swimmer + n_not_active_types

    # Template of the bond_coeff for top and bottom active bonds of each swimmer
          ( coefficients necessary for the bond style)

    if (bond_extended==1) {
  bond_coef_template_top     = "bond_coeff %i harmonic/swimmer/extended ${
       Umin_SW_} ${req_SW_} ${rmax_SW_} " \
      "${A_alpha_SW_} ${A_beta_SW_} ${omega_alpha_SW_} ${omega_beta_SW_} ${
           phi_SW_} ${vel_sw_SW_} %i %i\n"
  bond_coef_template_bottom     = "bond_coeff %i harmonic/swimmer/extended ${
       Umin_SW_} ${req_SW_} ${rmax_SW_} " \
      "${nA_alpha_SW_} ${nA_beta_SW_} ${omega_alpha_SW_} ${omega_beta_SW_} ${
           phi_SW_} ${vel_sw_SW_} %i %i\n"
    }
    if (bond_extended==2) {
  bond_coef_template_top     = "bond_coeff %i harmonic/swimmer/extended/k ${
       Umin_SW_} ${k_beta_SW_} ${req_SW_} ${rmax_SW_} " \
      "${A_alpha_SW_} ${A_beta_SW_} ${omega_alpha_SW_} ${omega_beta_SW_} ${
           phi_SW_} ${vel_sw_SW_} %i %i\n"
  bond_coef_template_bottom     = "bond_coeff %i harmonic/swimmer/extended/k ${
       Umin_SW_} ${k_beta_SW_} ${req_SW_} ${rmax_SW_} " \
      "${nA_alpha_SW_} ${nA_beta_SW_} ${omega_alpha_SW_} ${omega_beta_SW_} ${
           phi_SW_} ${vel_sw_SW_} %i %i\n"

}
      else {
  bond_coef_template_top     = "bond_coeff %i harmonic/swimmer ${Umin_SW_} ${
       req_SW_} ${rmax_SW_} ${A_SW_} ${omega_SW_} ${phi_SW_} ${vel_sw_SW_} %i %i
       \n"
  bond_coef_template_bottom     = "bond_coeff %i harmonic/swimmer ${Umin_SW_} ${
       req_SW_} ${rmax_SW_} ${nA_SW_} ${omega_SW_} ${phi_SW_} ${vel_sw_SW_} %i %
       i\n"
    }
    top_line_length_template    = "sw_active_lenght_SW_"

    head_surface_type = 3
}

# Hold a place for the number of bonds which will be later calculated
NR==3 {
    print
    print "_PLACE_HOLDER_", "bonds"
    next
}

NR==4 {
    print
    printf "%i bond types\n", n_bond_types
    next
}

/^Atoms/ {
    in_atoms = 1
    print
    getline
    print
    next
```

```
}

in_atoms&&!NF {
    in_atoms=0
}

in_atoms&&NF { #?
    x=$3; y=$4; z=$5
    if ( (length(np_second)==0) && (length(y_old)>0) && (fabs(y-y_old)>eps) ) {
  np_second = $1

    }
    x_old=x; y_old=y; z_old=z
}
{
    print
}

##### Define functions to create different bond types for each part of the swimmer:
##### NOTE: the "create_active_line" function creates a file (in.swimmer.topology) with the "bond_coeff" style for each bond style of each active part of each swimmer

#(1) Function for the active part of the swimmer (where the bonds changes equilibrium size)
function create_active_line(x_start, x_end, y_level, is_top_line,        btype, ip, bond_coef_template) {
    btype = (i_swimmer - 1)*2 + n_not_active_types + is_top_line + 1 # Bond type number
    # Variables for btype:
    # i_swimmer = swimmer id
    # n_not_active_types = number of non-actives bond types
    # is_top_line => 1 for top  line and 0 for bottom line

    for (ip=x_start; ip<=x_end; ip++) {
  print ++ibond, btype, xy2id(ip, y_level), xy2id(ip+1, y_level)
    }
    if (is_top_line) {bond_coef_template = bond_coef_template_top} else {bond_coef_template = bond_coef_template_bottom}
    gsub("_SW_", i_swimmer, bond_coef_template) #?

    printf bond_coef_template, btype, xy2id(x_start, y_level), xy2id(x_end, y_level) >> "in.swimmer.topology"

    if (is_top_line) {
  # create a variable with active line length
  top_line_length_output = top_line_length_template
  gsub("_SW_", i_swimmer, top_line_length_output)
  print "variable", top_line_length_output, "equal", x_end - x_start >> "in.swimmer.parameters"
    }
}

#(2) Function for the passive part of the swimmer
function create_passive_line(x_start, x_end, y_level, b_type,       btype, ip) {
    btype = b_type
    for (ip=x_start; ip<=x_end; ip++) {
  print ++ibond, btype, xy2id(ip, y_level), xy2id(ip+1, y_level)
    }
}

# (3) Function for the internal central line ("bones") of the swimmer
```

```
function create_internal_line(x_start, x_end, y_level,
            start_closed, end_closed,
            b_type,                               btype, ip) {
    # vertical
    btype = b_type
    for (ip=x_start + 1 - start_closed; ip<=x_end+end_closed; ip++) {
  print ++ibond, btype, xy2id(ip, y_level), xy2id(ip, y_level+1)
    }

    # diagonal
    for (ip=x_start; ip<=x_end; ip++) {
  print ++ibond, btype, xy2id(ip, y_level), xy2id(ip+1, y_level+1)
    }
}

###### Create the swimmer head

#Check if the bond  is on the surface on the head
function is_same_surface(ip1, jp1, ip2, jp2) {
    if   (ip1==_x1 && ip2==_x1 && jp1==_y1+1 && jp2==_y1+2) return 0 # a special
         case for the connection between the head and the body of the swimmer
    else if (ip1==ip2 && ip1==_x1) return 1
    else if (ip1==ip2 && ip1==_x2) return 1
    else if (jp1==jp2 && jp1==_y1) return 1
    else if (jp1==jp2 && jp1==_y2) return 1
    else return 0
}

# Delete corner atoms from the square grid to give a different format to the
     head
function is_on_grid(ip, jp) {
    if ( (ip==_x1) && (jp==_y1) ) return 0
    if ( (ip==_x1) && (jp==_y2) ) return 0
    if ( (ip==_x2) && (jp==_y1) ) return 0
    if ( (ip==_x2) && (jp==_y2) ) return 0
    return (ip>=_x1 && ip<=_x2 && jp>=_y1 && jp<=_y2)
}

# Check if the bond is in the head flesh
function is_head_flesh(ip, jp) {
    return (jp==_y2) || (jp==_y1)
}

#### Add filters to change configurations of the head ####
function bond_filter(ip1, jp1, ip2, jp2) {
    if (    jp1!=_y1 && jp1!=_y2 \
   && jp2!=_y1 && jp2!=_y2 \
   && jp1==jp2+1 \
       )   return 1
    return 0
}

# Set special bond type for the head flesh , for the inside bone (strong) and
     for the head front part (passive)
function head_bond_dispatch(ip1, jp1, ip2, jp2) {
    if ((is_head_flesh(ip1, jp1)) || (is_head_flesh(ip2, jp2))) return
         bond_head_flesh
    if (is_same_surface(ip1, jp1, ip2, jp2)) return bond_passive
    return bond_strong
}

# Function to return and print bond types of the head parts
function make_grid_bond(ip1, jp1, ip2, jp2,                btype) {
    if (!(is_on_grid(ip1, jp1) && is_on_grid(ip2, jp2))) return 0
    if (bond_filter(ip1, jp1, ip2, jp2)) return 0 # Set bond type passive to the front part of the swimmer head
    btype = head_bond_dispatch(ip1, jp1, ip2, jp2)
    print ++ibond, btype, xy2id(ip1, jp1), xy2id(ip2, jp2)
}

# Special function to change atom type of the atoms in the flesh region of the head
function add_line_to_change_type(ip, jp,        id) {
    id = xy2id(ip, jp)
    printf "group  sw_aux id %i\nset    group sw_aux type %i\n\n", id, head_surface_type >> "in.swimmer_change_type"
}

function change_surface_type(ip) {
    for (ip=_x1; ip<=_x2; ip++) {
  if (is_on_grid(ip, _y2)) add_line_to_change_type(ip, _y2)
    }

    for (ip=_x1; ip<=_x2; ip++) {
  if (is_on_grid(ip, _y1)) add_line_to_change_type(ip, _y1)
    }
}

#Function to create the bonds in the swimmer head (grid)

function create_grid(x1, y1, x2, y2,                              ip, jp) {
    _x1 =x1; _y1=y1; _x2=x2; _y2=y2

    for (ip=_x1; ip<=_x2; ip++) {
  for (jp=_y1; jp<=_y2; jp++) {
      make_grid_bond(ip, jp, ip+1, jp)
      make_grid_bond(ip, jp, ip, jp+1)
      make_grid_bond(ip, jp, ip+1, jp+1)
      make_grid_bond(ip, jp, ip+1, jp-1)
  }
    }

    change_surface_type()
}

# Function to create the swimmer part by part

function create_swimmer() {
    i_swimmer++

    # bottom line (active + tail)
    create_active_line(sw_start_x + sw_tail_length+1,  sw_start_x  + sw_head_start,
           sw_start_y,
           0)

    create_passive_line(sw_start_x, sw_start_x + sw_tail_length,
      sw_start_y, bond_passive)

    # top line (active + tail)
    create_active_line(sw_start_x + sw_tail_length+1, sw_start_x + sw_head_start,
           sw_start_y+1,
           1)

    create_passive_line(sw_start_x, sw_start_x + sw_tail_length,
      sw_start_y+1, bond_passive)

    #  internal line
    create_internal_line(sw_start_x, sw_start_x + sw_tail_length,
       sw_start_y, 1, 0, bond_strong)

    create_internal_line(sw_start_x + sw_tail_length+1, sw_start_x +
        sw_head_start,
       sw_start_y, 1, 0, bond_strong)
    #Swimmer Head
    create_grid(sw_start_x + sw_head_start + 1, sw_start_y - 1,
    sw_start_x + sw_head_start + sw_head_length + 1, sw_start_y + 2,
    bond_passive)
}

END {
    # Add bonds list
    if (sw_length>0) print "\nBonds\n" #  Bonds definition : id type atom_i
        atom_j
    printf "" > "in.swimmer.parameters"
    printf "" > "in.swimmer.topology"
    printf "" > "in.swimmer_change_type"

## Create Swimmers and define start points in x and y directions
## NOTE: definition sequence for each swimmer:
##       sw_start_y = ( starting point of the swimmer in y-direction)
##       sw_start_x = (starting point of the swimmer in x-direction)
##       create_swimmer()

    sw_start_y =
    sw_start_x =
    create_swimmer()

    close("in.swimmer.topology")
    close("in.swimmer.topology")
    close("in.swimmer_change_type")
}
```

```
LAMMPS data file via write_data, version 29 Jul 2014, timestep = 0

14400 atoms
415     bonds
3 atom types
5 bond types

0.0000000000000000e+00 1.0000000000000000e+00 xlo xhi
0.0000000000000000e+00 1.0000000000000000e+00 ylo yhi
-3.0000000000000001e-03 3.0000000000000001e-03 zlo zhi

Masses

1 1
2 1

Atoms # hybrid

1 1 1.6666666666666649e-02 1.6666666666666649e-02 0.0000000000000000e+00
     0.0000000000000000e+00 0.0000000000000000e+00 1.0000000000000000e+00 0 0 0
     0
2 1 4.9999999999999947e-02 1.6666666666666649e-02 0.0000000000000000e+00
     0.0000000000000000e+00 0.0000000000000000e+00 1.0000000000000000e+00 0 0 0
     0
3 1 8.3333333333333245e-02 1.6666666666666649e-02 0.0000000000000000e+00
     0.0000000000000000e+00 0.0000000000000000e+00 1.0000000000000000e+00 0 0 0
```

```
0
4 1 1.1666666666666654e-01 1.6666666666666649e-02 0.0000000000000000e+00 0.0000000000000000e+00 0.0000000000000000e+00 1.0000000000000000e+00 0 0 0 0
5 1 1.4999999999999986e-01 1.6666666666666649e-02 0.0000000000000000e+00 0.0000000000000000e+00 0.0000000000000000e+00 1.0000000000000000e+00 0 0 0 0
6 1 1.8333333333333313e-01 1.6666666666666649e-02 0.0000000000000000e+00 0.0000000000000000e+00 0.0000000000000000e+00 1.0000000000000000e+00 0 0 0 0
7 1 2.1666666666666645e-01 1.6666666666666649e-02 0.0000000000000000e+00 0.0000000000000000e+00 0.0000000000000000e+00 1.0000000000000000e+00 0 0 0 0
8 1 2.4999999999999972e-01 1.6666666666666649e-02 0.0000000000000000e+00 0.0000000000000000e+00 0.0000000000000000e+00 1.0000000000000000e+00 0 0 0 0
9 1 2.8333333333333305e-01 1.6666666666666649e-02 0.0000000000000000e+00 0.0000000000000000e+00 0.0000000000000000e+00 1.0000000000000000e+00 0 0 0 0
10 1 3.1666666666666637e-01 1.6666666666666649e-02 0.0000000000000000e+00 0.0000000000000000e+00 0.0000000000000000e+00 1.0000000000000000e+00 0 0 0 0
.
.
.

Velocities

1 0 0 0
2 0 0 0
3 0 0 0
4 0 0 0
5 0 0 0
6 0 0 0
7 0 0 0
8 0 0 0
9 0 0 0
10 0 0 0
.
.
.

Bonds

1 4 7043 7044
2 4 7044 7045
3 4 7045 7046
4 4 7046 7047
5 4 7047 7048
6 4 7048 7049
7 4 7049 7050
8 4 7050 7051
9 4 7051 7052
10 4 7052 7053
.
.
.
```

List of Figures

Bibliography

[BA73] Howard C. Berg and Robert A. Anderson. Bacteria swim by rotating their flagellar filaments. *Nature*, 245(5425):380–382, October 1973.

[BL73] J J Blum and J Lubliner. Biophysics of flagellar motility. *Annual Review of Biophysics and Bioengineering*, 2(1):181–219, 1973.

[BS10] I. Borazjani and F. Sotiropoulos. On the role of form and kinematics on the hydrodynamics of self-propelled body/caudal fin swimming. *The Journal of Experimental Biology*, 213(1):89–107, January 2010.

[Chi81] Stephen Childress. *Mechanics of Swimming and Flying.* Cambridge University Press, Cambridge, 1981.

[Dig] Biology Digest. Nature's flyers.

[EKG10] Jens Elgeti, U. Benjamin Kaupp, and Gerhard Gompper. Hydrodynamics of sperm cells near surfaces. *Biophysical Journal*, 99(4):1018–1026, August 2010.

[GM77] R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics - theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181:375–389, November 1977.

[GM78] R. A. Gingold and J. J. Monaghan. Binary fission in damped rotating polytropes. 1978.

[GM79] R. A. Gingold and J. J. Monaghan. A numerical study of the roche and darwin problems for polytropic stars. *Monthly Notices of the Royal Astronomical Society*, 188(1):45–58, September 1979.

[GM82] R. A. Gingold and J. J. Monaghan. Kernel estimates as a basis for general particle methods in hydrodynamics. *Journal of Computational Physics*, 46(3):429–453, 1982.

[GSVLL11] Georg C. GanzenmÃ$\frac{1}{4}$ller, Martin O. Steinhauser, Paul Van Liedekerke, and Katholieke Universtiteit Leuven. The implementation of smooth particle hydrodynamics in LAMMPS. 2011.

[JBL95] D. Joly, C. Bressac, and D. Lachaise. Disentangling giant sperm. *Nature*, 377(6546):202–202, September 1995.

[KK06] Stefan Kern and Petros Koumoutsakos. Simulations of optimized anguilliform swimming. *The Journal of Experimental Biology*, 209(Pt 24):4841–4857, December 2006.

[lam] Lammps users manual.

[LP09] Eric Lauga and Thomas R Powers. The hydrodynamics of swimming microorganisms. *Reports on Progress in Physics*, 72(9):096601, September 2009.

[Luc77] L. B. Lucy. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal*, 82:1013, December 1977.

[McH05] Matthew J McHenry. The morphology, behavior, and biomechanics of swimming in ascidian larvae. *Canadian Journal of Zoology*, 83(1):62–74, January 2005.

[Mon05] J J Monaghan. Smoothed particle hydrodynamics. *Reports on Progress in Physics*, 68(8):1703–1759, August 2005.

[Mon12] J.J. Monaghan. Smoothed particle hydrodynamics and its diverse applications. *Annual Review of Fluid Mechanics*, 44(1):323–346, January 2012.

[NS68] George A. Neece and David R. Squire. Tait and related empirical equations of state. *The Journal of Physical Chemistry*, 72(1):128–136, January 1968.

[RAG80] J. J. Monaghan R. A. Gingold. The roche problem for polytropes in central orbits. *Monthly Notices of the Royal Astronomical Society*, 191:897–924, 1980.

[SL06] R Shadwick and G. Lauder. Fish biomechanics. *Books*, January 2006.

[SP06] S. S. Suarez and A. A. Pacey. Sperm transport in the female reproductive tract. *Human Reproduction Update*, 12(1):23–37, February 2006.

[SYL01] Hitoshi Sawada, Hideyoshi Yokosawa, and Charles C. Lambert. *The Biology of Ascidians*. Springer, January 2001.

[Tay52] Geoffrey Taylor. Analysis of the swimming of long and narrow animals. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 214(1117):158–183, August 1952.

[THW$^+$10] Eric D. Tytell, Chia-Yu Hsu, Thelma L. Williams, Avis H. Cohen, and Lisa J. Fauci. Interactions between internal forces, body stiffness, and fluid environment in a neuromechanical model of lamprey swimming. *Proceedings of the National Academy of Sciences*, 107(46):19832–19837, 2010.

[Vog96] Steven Vogel. *Life in Moving Fluids: The Physical Biology of Flow*. Princeton University Press, Princeton, N.J., 2nd revised edition edition, April 1996.

[Wil10] Thelma L. Williams. A new model for force generation by skeletal muscle, incorporating work-dependent deactivation. *The Journal of Experimental Biology*, 213(4):643–650, February 2010.

Declaration

Surculus, Epulae pie Anxio conciliator era se concilium. Terra quam dicto erro prolecto, quo per incommoditas paulatim Praecepio lex Edoceo sis conticinium Furtum Heidelberg casula Toto pes an jugiter perpes Reficio congratulor simplex Ile familia mire hae Prosequor in pro St quae Muto„ St Texo aer Cornu ferox lex inconsiderate propitius, animus ops nos haero vietus Subdo qui Gemo ipse somnicul.

München, xx. September 20xx

Name des Autors