A Study of the Motion of the Three-Linked Swimmer in Viscous Fluid Using Computational and Experimental Methods

by

Susan YeYoung Ji

Submitted to the Department of Mechanical Engineering in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Science

at the

Massachusetts Institute of Technology

June 2004

Signature of Author........................[illegible]
Department of Mechanical Engineering
May 6, 2004

Certified by........................[illegible]
Anette Hosoi
Assistant Professor of Mechanical Engineering
Thesis Supervisor

Accepted by........................[illegible]
Ernest G. Cravalho
Chairman, Undergraduate Thesis Committee

A Study of the Motion of the Three-Linked Swimmer in Viscous Fluid Using Computational and Experimental Methods

by

Susan YeYoung Ji

Submitted to the Department of Mechanical Engineering
on May 6, 2004 in Partial Fulfillment of the
Requirements for the Degree of Bachelor of Science in
Mechanical Engineering

ABSTRACT

An experimental and numerical study was conducted to better understand the mechanics of motion of the three-linked swimmer in viscous fluid. Numerical studies in C++ were used to predict the velocity and motion of the swimmer using a modified analytical model that considers the dynamical and orientational effects of rods at the surface of a fluid. Computational simulations were graphically used to understand the pressure and velocity distributions of the fluid-structure interactions of the three-linked swimmer, with slightly different movement, using an immersed boundary method computer program. Experiments using a mechanical prototype of a three-linked swimmer were conducted to validate and compare the numerical predictions derived from computational studies.

Experimental results indicate that a flexible armed swimmer is more than three times as fast as its rigid armed counterpart. The analytical model presented in this study and the corresponding computational numerical simulations were found to capture the trends of the motion, and the predicted horizontal velocity came within 66% in value of the recorded experimental data.

Thesis Supervisor: Anette Hosoi
Title: Assistant Professor of Mechanical Engineering

1. INTRODUCTION

This section outlines the background and purpose of this study.

1.1 Scale and Size Effects

This study was inspired by the locomotion of small organisms found in nature. It is important to consider the efficiency at which organisms in nature move because the movement of natural organisms can provide the inspiration for the design of human-made devices.

At the small scale, the effects of viscosity, cohesion, and diffusion become extremely important, while the effects of gravity become insignificant. To accurately recreate the effects of the swimming of microorganisms at a scale possible for experimentation, one must use viscous fluids, i.e. fluids at low Reynolds numbers because of the geometrically similar principle [3]. The small size and cruising speed of the bacteria makes the viscous forces dominant when these kinds of organisms move. The Reynolds number for larger organisms is of higher magnitude than for smaller creatures because the Reynolds number reflects the relative importance of viscosity and inertia. This indicates why smaller organisms must rely on the physics of viscous fluids for propulsion.

1.2 Characteristics of flow at Low Reynolds Numbers

G.I. Taylor demonstrated with dye in glycerin that at low Reynolds numbers, flow is reversible [3]. If an organism moves backwards the same amount it moved forward, the net amount of movement is zero. To move in the limit of small Reynolds number, an organism needs to produce time-irreversible motion, that is to say, you cannot change your body into a certain shape, and then go back into the original shape by going through the sequence in reverse. If an animal tries to swim by a reciprocal movement, motion is not possible; the organism ends up back where it started (Purcell). For example, in a low Reynolds number limit, a scallop cannot swim because the movement for forward movement is the same as for that as backwards movement, as demonstrated in figure 1.

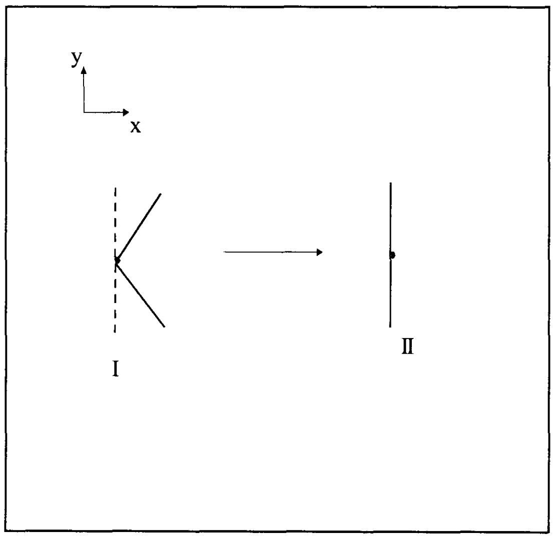


Figure 1: "Scallop" configuration, which is an example of reciprocal motion. This type of behavior results in no net motion in viscous fluid. Schematics I and II describe the respective positions. [1]

The position of an organism depends only on the configuration, not the instantaneous velocity.

1.3 Types of Swimmers at Low Reynolds Numbers

A more succinct summary of the characteristics of viscous fluid swimmers can be found in Appendix A.

It is important to discuss the various types of swimmers at low Reynolds numbers in order to better understand the context of the three-link swimmer under particular scrutiny in this study.

1.3.1 Three-Link Configuration

E.M. Purcell introduced the concept of the three-link swimmer as the simplest organism that can move in the low Reynolds number limit [12]. The analytical modeling of this configuration models the motion in terms of the translation of a point, the velocity due to rigid body rotation, and the stretching motion, and then substituting into the resistance matrix for the force, velocity, and stretching components [1]. It is possible to model the large-amplitude motion of the linkages driven by internal motors acting at the joints, as well as determine the optimal swimmer for conditions. The biological analogy to the three-linked swimmer is the spiroplasma bacteria [2]. It was found at moderate stroke angles, translation is opposite to that of the wave velocity, and that for larger stroke amplitudes, the direction of net translation reverses, that is, for moderate stroke angles, the direction is the same to that of the wave velocity. Numerical studies were also conducted where the displacements in the x-direction, displacements of the center and joint, the velocity, and efficiency were solved for graphically. Graduate student Brian Chan at MIT has developed a mechanical model of this configuration. Purcell's swimmer presents possibilities in the use of micro-robots, such as those deployed in the human body for minimally invasive therapeutic treatments. Figure 2 describes the sequence of movement for this swimmer.

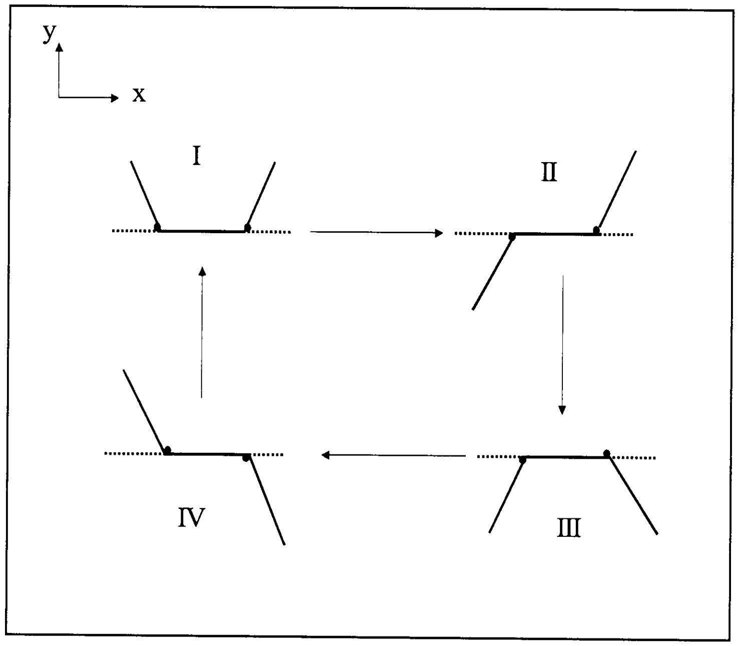


Figure 2:Configurations of the Three-Link Swimmer. The respective positions and the order in which the arms move. [1]

1.3.2 Rotating Helix

This configuration is similar to the geometry of the *Escherichia coli.* It is a round in diameter, and twisted in a helical form [13]. It was analytically determined that at low Reynolds numbers, a translating force will cause the helix to rotate even without the presence of an externally applied torque. By the linearity of the Stokes equations, an applied torque will also cause the helix to translate in position.

1.3.3 Waving Sheet

This configuration is similar to the microscopic bacteria with tails. A previous study done by G.I. Taylor examines the 2-D behavior, modeling the waving tail as a sheet in viscous fluid [15]. The waving surface was modeled as a sine wave, and the stream function was utilized to model the flow near a sheet of small amplitudes. The microscopic organism *spirochaeta balbianii* exhibits this type of behavior for movement.

A three-dimensional model, which takes into account the hydrodynamic interaction effects, has not been completed at this date, and the immersed boundary method (IBM) probably can present an effective solution to this problem.

1.3.4 One-Dimensional Flags in a 2-D Wind

By using nylon threads in tension in a soap film, the effects of flapping flags in a two-dimensional wind were experimentally determined. A similar kind of behavior is demonstrated in the flapping fins of swimming fish. It was shown that below a critical filament length, the filament is not affected by the flow [18]. At the transition to flapping, the mass of the fluid balances the mass of the filament, while the elastic energy of the filament balances the kinetic energy of the fluid.

1.3.5 Elastic Filament with Internally Generated Stresses

Filaments with internally generated stresses are the type of behavior that is demonstrated in cilia and flagella [9]. The three types of possible configurations include the clamped head, with the position and slope fixed; the fixed head, where position is fixed only; and the free head that is subjected to viscous load. The refined slender body theory was used to model the movement of the cilia. It was determined that the effective and recovery strokes encounter different resistances.

The case of nonlinearities and torsional motion in three-dimensions has not been determined. The situations outlined in earlier literature take into account only internal stresses; the case for external forces has not yet been ascertained.

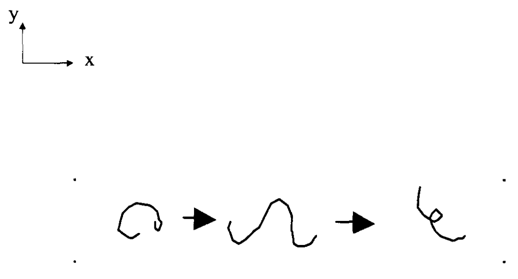


Figure 3: Supercoiling of twisted elastic filaments. The kinematics of supercoiling are dictated by geometry and viscous dynamics and are also governed by elasticity.

1.3.6 Twisted Elastic Filament in Viscous Fluid

The motion of the bacterium *B. subtilis* was the inspiration for the study of twisted elastic filaments, and this kind of behavior demonstrates that the supercoiling of elastic elements can be described by the kinematics dictated by geometry, and the viscous dynamics governed by elasticity [7]. Figure 3 shows the geometry of supercoiling. The details of the geometric untwisting and twist density evolution have been numerically determined.

1.3.7 Rotating Rod with a Kink

An analytical study of this geometry showed that the elements of viscous drag, twisting, and bending in dynamics of rotating filaments are related [10]. The formulaic analysis was completed by using elastic internal stresses to find moment and force, and then doing a force balance to find the dimensional shape of the rod. This type of natural curvature is found in bacteria and protein binding DNA. A mechanical model of this structure of the rods with a small to moderate angle (L-shaped) will extend, and rods with a larger angle (V-shaped) will fold. The pictoral depiction of this configuration is described in figure 4.

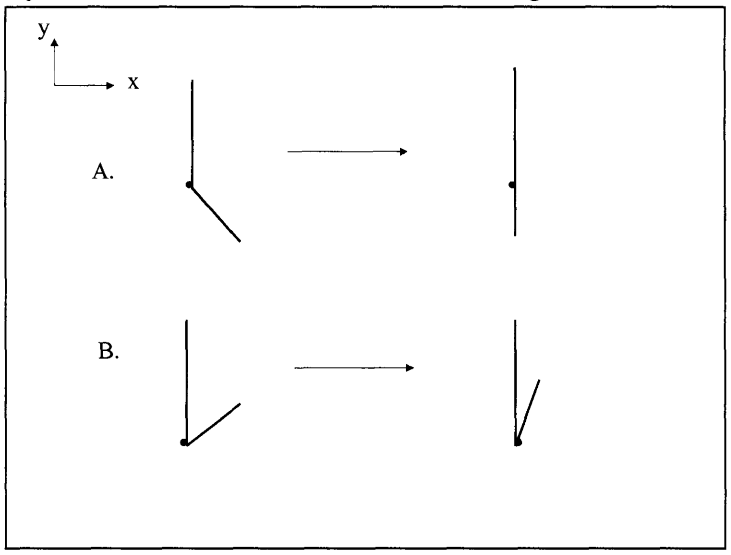


Figure 4: Schematic (A) describes the configuration of a rotating rod when the kink is a small-moderate angle. The rods with a larger angle will fold in the same situation, as illustrated in schematic (B).

1.4 Types of Modeling

This study is concerned with two types of modeling, computational numerical analysis and the immersed boundary method. Each of these methods examines different aspects of the same movement. .

1.4.1 Modeling of the Three-Linked Swimmer

The analytical model used in this study follows the methods developed to examine a series of rods floating in a liquid-gas interface [16]. The study especially considers the dynamical and orientational effects of rods at the surface of a fluid. The equations of motion presented that described the dynamics between a rod and a wall were of particular interest and this study focused on modified versions of these equations. A Raleigh dissipation function was used to describe the viscous effects and drag on the rod, and a Lagrangian multiplier method was utilized to solve the equations of motion to find the position of the rods as a function of time.

1.4.2. Immersed Boundary Method

The Immersed Boundary Method (IBM), a computational method and mathematical formulation, was developed to study the interaction between the fluid and the boundary of an immersed object, created by Charlie Peskin at the New York University [11]. The technique models the biological tissue as a part of the fluid on a lattice, intertwining Lagrangian and Eulerian coordinate systems. The Immersed Boundary Method provides an effective solution to the coupled nature of this situation; the fluid pushes against the boundary of the system the same time the boundary pushes back against the fluid. The technique also addresses the mathematical problem of solving the Navier-Stokes equations in a complex domain. This method has been used to describe the behavior of biological systems such as the heart and its valves, the inner ear, and the deformation and locomotion of cells.

The immersed boundary method is useful because the data derived from this program can be used to study the fluid-structure interactions in an object. This kind of knowledge is important in understanding the industrial applications of such devices as the three-linked swimmer. This type of close study of such small-scale viscous swimmers can assist in the construction and design of such applications as small medical applications that could be used for direct drug delivery in the bloodstream. The immersed boundary method is useful because the program makes it possible to understand what cannot be measured in experimentation.

1.5 Purpose

The purpose of this study is to computationally examine the mechanism of movement of low-Reynolds number organisms, specifically the 3-link swimmer. One computational method will simulate the motion of the 3-link swimmer using a modified analytical model [16], and the other method will study the pressure and velocity distributions through the immersed boundary method. Results will be validated and compared with experimental data of a rigid armed and flexible armed three-linked swimmer.

2. THEORETICAL ANALYSIS

2.1 Immersed Boundary Method

The Immersed Boundary Method describes an object and its boundary on a background stationary Cartesian grid, while allowing the object itself to be flexible in movement [11]. The object is defined in terms of the stationary grid through a fixed number of Lagrangian markers. The kinematic boundary condition is that the filament must move with the fluid velocity.

The Navier-Stokes equations are given by equation (1):

$$\rho \frac{D\vec{u}}{Dt} = -\nabla p + \mu \nabla^2 \vec{u} + \vec{F}_{ext}, \qquad (1)$$

where the force term $\vec{F}_{ext}$ refers to all of the external forces in the system, including elastic, gravitational, and bending, and where $\frac{D}{Dt}$ is the material derivative.

The restrictive condition for continuity states that:

$$\nabla \bullet \vec{u} = 0. \qquad (2)$$

If we take the divergence of (1), the following relation is derived:

$$\vec{\nabla}(\rho \frac{D\vec{u}}{Dt} = -\nabla p + \mu \nabla^2 \vec{u} + F_{ext}), \qquad (3)$$

$$0 + \rho \nabla \cdot ((\vec{u}\nabla)\vec{u}) = \nabla^2 p + \vec{\nabla} \cdot F_{ext}. \qquad (4)$$

If we discretize equation (1) in time:

$$\rho(\frac{u^{n+1} - u^n}{\Delta t} + \vec{u}^n \nabla u^n) = -\nabla p^{n+1} + \mu \nabla^2 u^n + F_{ext}^n, \qquad (5)$$

where $u^n = u(t = n\Delta t)$.

Through further manipulation, (5) can be broken down into the following two equations:

$$\rho(\frac{u^{\sim} - u^n}{\Delta t} + \vec{u}^n \nabla u^n) = v \nabla^2 u^n + F_{ext}^n, \qquad (6)$$

$$\rho^n(\frac{u^{n+1} + u^{\sim}}{dt}) = -\nabla p^{n+1}. \qquad (7)$$

When (6) and (7) are added together, it forms (5), and thus is the equivalent. From the condition in (2), the following relation can be determined:

$$\vec{\nabla}\frac{\hat{u}\rho^{n}}{dt}=\nabla^{2}p^{n+1}. \qquad (8)$$

From the relation in (8), we can solve for the pressures in the system (p^{n+1}).

Once the pressures, (p^{n+1}), in the system have been solved, these can be substituted into (7) to solve for the velocities for the system at the specific time.

In order to count the effects of the external forces on the object, we must first examine the energy of the system,

$$\begin{aligned} E_e &= E_b + E_s \qquad &(9)\\ &= \Sigma B, \qquad &(10)\end{aligned}$$

where the subscripts *e, b,* and *s* in (9) refer to elastic, bending, and stretching energies respectively.

From this relation, the external force can be described as,

$$f_{ext}=-d_x^4X(B)-d_x[(|X_x|-1)\frac{X_s}{|X_s|},-Bd_x^4X-d_s[(|X_s|-1)\frac{X_s}{|X_s|}], \qquad (11)$$

where the subscript *s* refers to the arc length along the membrane. These terms consider the amount of stretching of the membrane, and the direction of the stretching.

However, the relation in (11) makes no distinction between the fluid and the object. Thus, from relation (11), the forces for each point in the object can be updated in the following manner,

$$F_{ext}(x,y)=\int_0^L f_{ext}+\delta(x-X(s),y-Y(s))ds, \qquad (12)$$

where the lowercase *x* and *y* refer to the position on the fluid grid. By using the Dirac delta function in (12) ensures that if the numerical function is not in the vicinity of a membrane point, the function will not be updated.

In the same manner, the density at each point in the system is updated,

$$\vec{\rho}(x,y)=\rho_f+\int_0^L \rho_l\ \delta(X^n-x,Y^n-y)ds. \qquad (13)$$

Finally, the position of each of the points in the membrane is updated based on the velocity calculations completed above using the concept of the delta Dirac function:

$$\frac{x^{n+1}-x^{n}}{dt}=u^{n}=\iint u^{n}\delta(X^{n}-x,\ Y^{n}-y)dxdy, \qquad (14)$$

where the variable in question is x^{n+1}. Thus equation (9) ensures that all values in the grid that are not near the objects are set to zero.

2.2 Three-Linked Swimmer

For a rod in viscous fluid, the following Lagrangian equation can be used to describe the behavior [16]:

$$\ell = \frac{1}{2}m(\dot{x}^2 + \dot{y}^2) + \frac{1}{2}I\dot{\Phi}^2, \tag{15}$$

where the first term corresponds to the translational behavior, and the second term describes the rotational kinetic energy associated with a rod. The mass of the rod is denoted with the variable *m,* the moment of inertia with *I*, while the coordinates of the rod at a given time are described with *x* and *y*. The angular orientation of the rod is reflected by the variable *Φ*.

The viscous effects of the rod are included in this approximation through the use of the Raleigh dissipation function:

$$\Re = \frac{1}{2}\sigma_{\Pi}(\dot{x}^2\cos^2\phi + \dot{y}^2\sin^2\phi) + \frac{1}{2}\sigma_{\perp}(\dot{x}^2\cos^2\phi + \dot{y}^2\sin^2\phi) + \frac{1}{2}\sigma_o\dot{\phi}^2 - (\sigma_{\perp} - \sigma_{\Pi})\dot{x}\dot{y}\sin\phi\cos\phi \tag{16}$$

Thus, the Euler-Lagrange equations thus modified become:

$$\frac{d}{dt}\left(\frac{d\ell}{d\dot{q}}\right) - \frac{d\ell}{dq} + \frac{d\Re}{d\dot{q}} = 0, \tag{17}$$

and thus the equations of motion become,

$$m\ddot{x} = -\dot{x}(\sigma_{\Pi}\cos^2\phi + \sigma_{\perp}\sin^2\phi) + \dot{y}(\sigma_{\perp} - \sigma_{\Pi})\sin\phi\cos\phi, \tag{18}$$

$$m\ddot{y} = \dot{x}(\sigma_{\perp} - \sigma_{\Pi})\sin\phi\cos\phi - \dot{y}(\sigma_{\Pi}\sin^2\phi + \sigma_{\perp}\cos^2\phi), \tag{19}$$

$$I\ddot{\phi} = -\sigma_o\dot{\phi}. \tag{20}$$

Thus, if *q* is used as a vector to describe the position and orientation of the rods, the equations of motions may be simplified to be written as:

$$\ddot{q} = M^{-1}(-J\lambda - F_D), \tag{21}$$

in matrix format. The form in (21) is used in order to facilitate the inclusion of constraints regarding the interactions between the links of the swimmer. *J* is the Jacobian matrix of the constraints $J_{i\alpha} = \frac{df_\alpha}{dq_i}$, where the constraints *f* can be defined in the following manner:

$$f_{2i-1}(q) = 0 = q_{3i+1} - q_{3i-2} - a(\cos q_{3i} + \cos q_{3i+3}), \tag{22a}$$

$$f_{2i} = 0 = q_{3i+2} - q_{3i-1} - a(\sin q_{3i} + \sin q_{3i+3}), \tag{22b}$$

$$f_{3i} = \textit{time dependant coordinates required to define positions of each link}, \tag{22c}$$

where i =(1….n), where n is the number of coordinates. In the particular case of the 3-linked swimmer in question, n is 9, and alpha becomes 6.

Following the definition of the matrix J, the dimensions of the matrix are 4x9. Using the chain rule in equation (21), we find:

$$0 = \frac{d^2 f_\alpha}{dt^2} = \sum_{j=1}^{3n} J_{j\alpha}\ddot{q}_j + \sum_{i=1}^{3n}\sum_{j=1}^{3n} \ddot{q}_i \frac{dJ_{i\alpha}}{dq_j}\dot{q}_j, \tag{23}$$

Substituting expression (23) into (21) results in an expression for the Lagrange multipliers, and thus the positions can be solved for numerically.

In order to solve for the Lagrangian multipliers lambda, equation (23) is simplified to the following expression:

$$J\ddot{q} = -\vec{b} = \sum_{i=1}^{3n}\sum_{j=1}^{3n} \dot{q}_i \frac{dJ_{i\alpha}}{dq_j}\dot{q}_j \ , \tag{24}$$

in matrix format, and this expression is used to find the equivalent expression for the acceleration in terms of the positions q. This expression for the acceleration in terms of J and b is substituted into relation (21) to determine a direct expression for the Lagrangian multipliers:

$$\vec{b} = (M^{-1}(J\lambda))^T J, \tag{25}$$

where M is the identity matrix of the mass and inertia terms.

In order the facilitate the matrix multiplication, the following matrix relations were used:

$$(A^T)^T = A, \tag{26}$$
$$(AB)^T = B^T A^T, \tag{27}$$

where A and B are arbitrary matrices, and the superscript T indicates the transpose of a matrix.

Manipulating relations (26) and (27) for equations (25) and (21), an expression for lambda is determined to be,

$$\lambda = b^T (J^T M^{-1} J)^{-1}, \tag{28}$$

where the superscript T indicates the transpose of a matrix.

Using a Runge-Kutta method to integrate the equations of motion, the positions at a specific time can be calculated.

3. METHODS

In the following section, the system setup and the procedural steps that were taken to complete this study will be outlined.

3.1 Computer Setup

To complete this study, a Toshiba Satellite 5005 laptop with a Pentium III processor was utilized. Version 6.0 of MATLAB was used, and the Dev C++ program was the compiler program operated.

3.2 Computer Code for Numerical Simulation of Three-Linked Swimmer

The equations of motions were simplified using a Lagrangian multiplier method, and a combination of MATLAB and C++ was used to solve the equations of motion numerically.

The actual code utilized during this study can be found in section A.2 of the Appendix section. Four codes were used: code1.cpp was used to initialize the variables, followed by gauss.m, a MATLAB file, and then code3.cpp was used to find the new positions. After the first run, code2.cpp was used in place of code1.cpp, followed by gauss.m, then code3.cpp.

3.3 Rigid Three-Linked Swimmer Apparatus

The three-linked swimmer device that was used in this study was a machine designed and created by the graduate student Brian Chan. In a resting position, the swimmer is 9 cm in total length, and 5 cm in height for the end links. The middle link was found to have a height of 4.5 cm. Each link was found to be 2.5 cm in width individually. Figure 5 illustrates the shape and size of the swimmer.

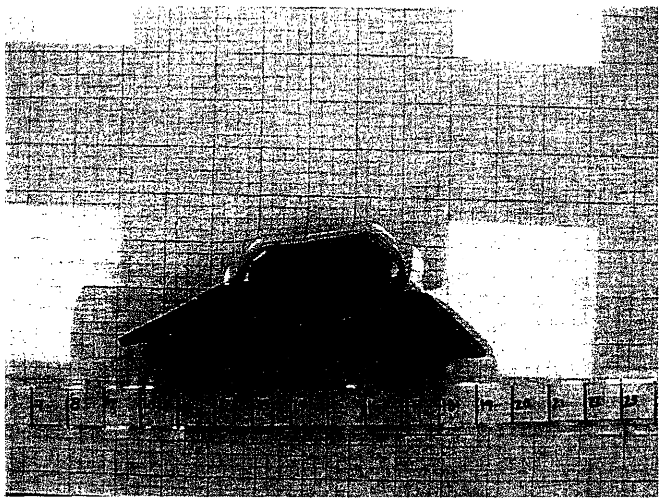


Figure 5: Three-Linked Swimmer (top view). This figure is a photo of the three-linked swimmer in viscous fluid. The ruler and graph paper indicate the size of the swimmer. The ruler at the bottom of the figure is measured in centimeters. The grid in the background is divided into measurements of 1/8".

Figure 6 illustrates the swimmer face side up to illustrate the features that are hidden during movement.

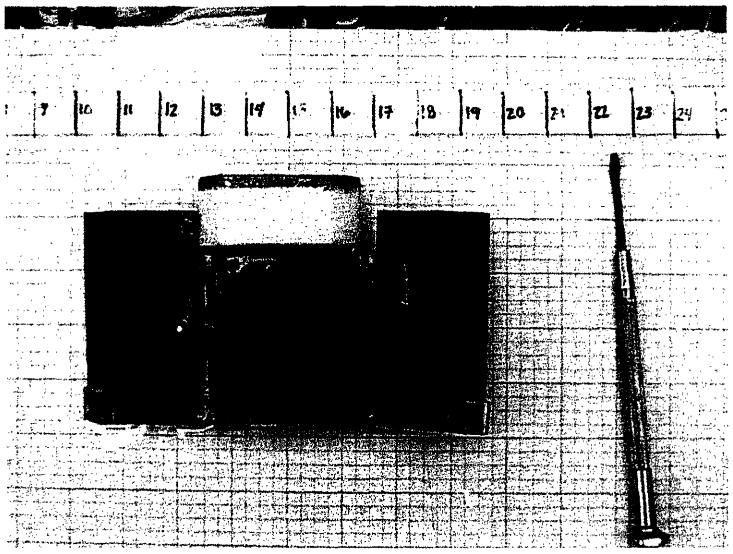


Figure 6: Three-linked swimmer (face up). This figure illustrates the height dimensions of the swimmer.

Figure 7 shows the manner in which the three-linked swimmer was wound up before it was set in motion.

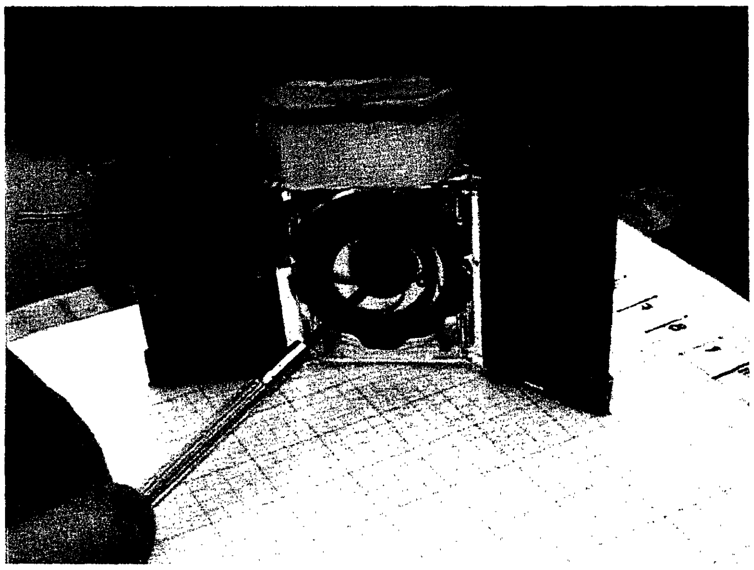


Figure 7: Winding up of the three-linked swimmer. A small screwdriver was used to set the swimmer in motion as illustrated in the figure.

The movement of the three-linked swimmer was analyzed through the use of a video camera that was setup on a tripod above the fluid and swimmer. Figure 8 illustrates the apparatus setup.

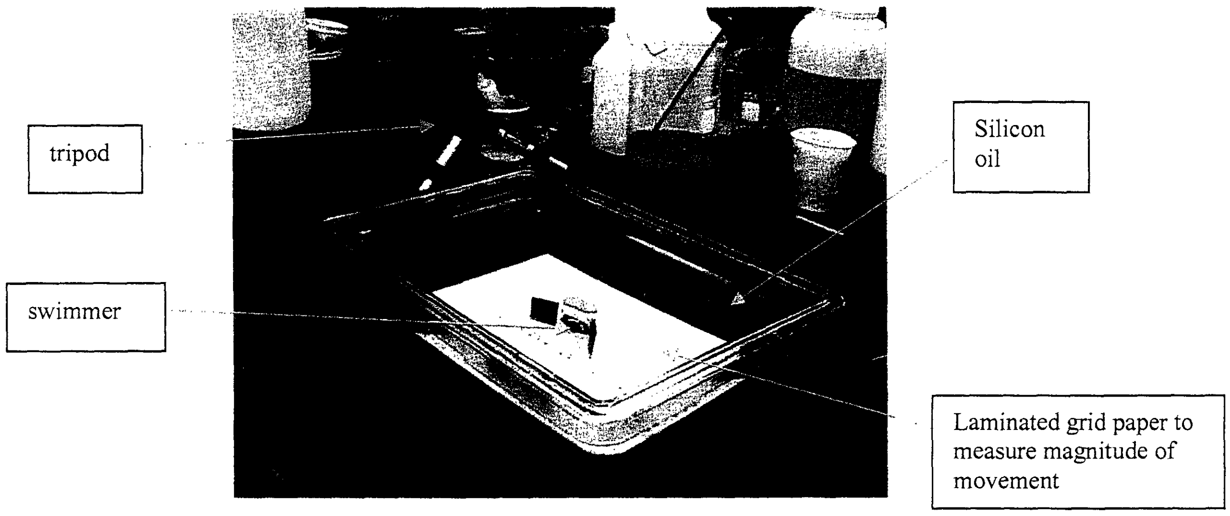


Figure 8: Experimental apparatus setup. Note that the graph paper is immersed within the silicon oil fluid.

3.3.Flexible Arm Three-Linked Swimmer Apparatus

A variation on the rigid arm three-linked swimmer device was also used during this study to examine the effects of arm rigidity upon swimming performance. These experiments were also conducted in order the correlate with the immersed boundary method effort in this study, because the immersed boundary method simulates the movement of a flexible membrane. The arms were made of pliable rubber, and were bendable, as demonstrated by figure 9.

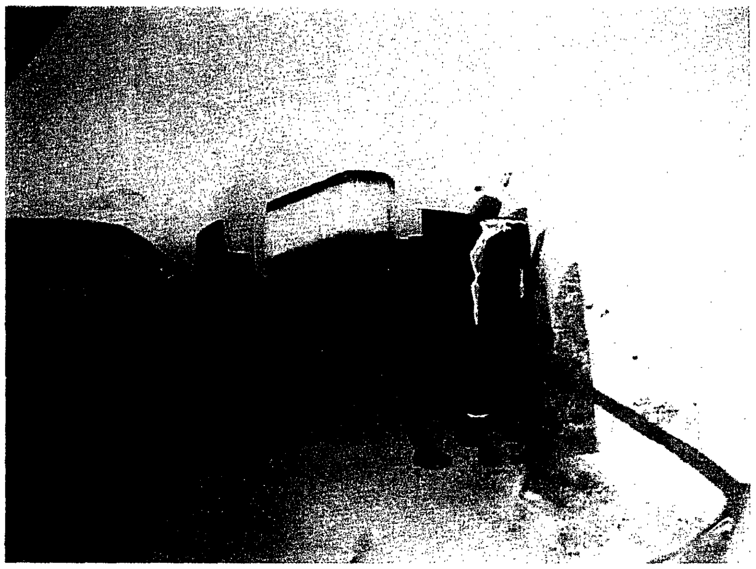


Figure 9: Demonstration of the flexible arms of the swimmer. Note that the arms can be easily bent.

Figure 10 shows the flexible three-linked swimmer in an upright position to show the similarities with the other design.

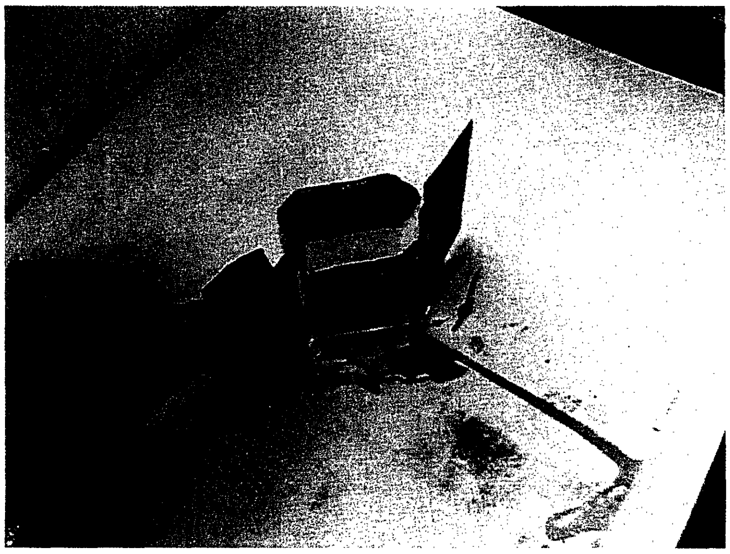


Figure 10: Demonstration of flexibility of membrane when swimmer is in upright position. This figure underscores the similarity with the rigid three-linked swimmer.

Figure 11 illustrates the three-linked swimmer in fluid. Once again, grid paper was used to measure the distance of travel of the three-linked swimmer.

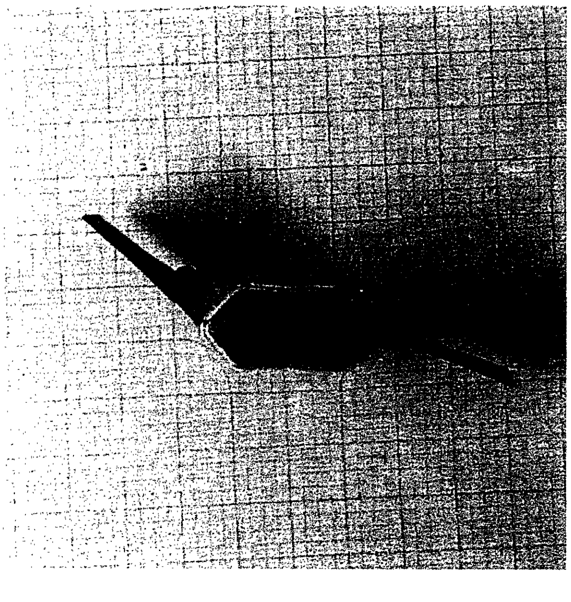


Figure 11: Flexible Three-Linked Swimmer in Fluid. Note the clarity of the graph paper lines in the background. The degree of clarity enabled ease of data analysis after the motion was recorded through a video camera.

3.3.2 Three-Linked Swimmer Experimental Procedure

In order to analyze the velocity and magnitude of movement of the swimmer, several experiments were completed, which consisted of recording on videotape the movement of the swimmer in silicon oil fluid. The Canon NTSC 2R65 MC Digital Video camcorder was used to take both still and video pictures. The swimmer was set into motion through winding it up with a screwdriver. Through carefully analyzing the film recorded, parameter such as the velocity and distance traveled per unit time, were determined.

3.4 Immersed Boundary Code

The Immersed Boundary code utilized in this study was developed by Mederic Argentina, a post-doctorate student with Professor L.Mahadevan at Harvard University. This program runs in the Macintosh environment, and was written in the C language. The simulation for the three-linked swimmer consisted of a torque of a given magnitude applied at two points of a horizontal rod.

There are a variety of Immersed Boundary programs, including the original Immersed Boundary Code constructed by N. Cowen, D.M. McQueen, and C.S. Peskin at New York University, Titanium from the University of California at Berkeley, and the graphical IBIS program written by David Eyre at the University of Utah. IBIS has a graphical interface and requires little programming ability from the user. This particular program was used because of the detailed graphical output at each time step.

4. RESULTS AND DISCUSSION

In this section, both the graphical representations of the data collected during computational simulation and experimentation is presented. The significance of the numerical simulations and experimental data with respect to the established theory is also reviewed.

4.1 Numerical Analysis of Three-Linked Swimmer

The drag coefficients for the 3-linked swimmer were estimated using a Reynolds number of about 0.1 [17]. The specific values utilized were: 44 for the drag parallel to the flow, 55 for the drag perpendicular to the flow, and 50 for the rotational drag.

Figure 12 illustrates the progression of movement for each of the links. The time dependant angle constraint has not been considered in these numerical studies. However, figures 12 and 13 do indicate the magnitude of movement for a given time period.

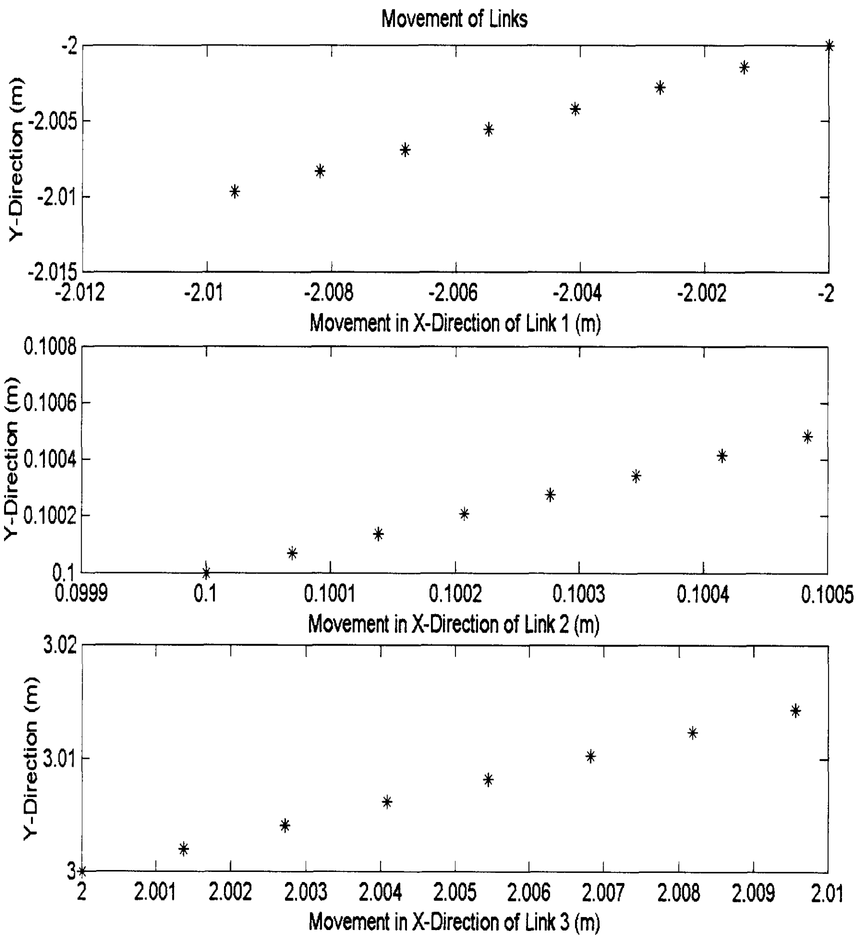


Figure 12: Comparison of Link Movement. The top graph shows the amount of link movement for the left most link (link one), the middle graph shows the movement for the middle link, and

the bottom graph shows the link movement progression for the bottom link. The starting position for link 1 in the horizontal direction is –2 m, for the middle link is 0.1m, and for link 3 is 2 m. In the vertical direction, link 1 moves 0.01 m downwards, link 2 moves 0.00483 m upwards, and link 3 moves 0.01442 m the upward direction.

Figure 1 indicates that the net motion of the swimmer in the horizontal direction for 7 seconds is 0.00483 meters, presenting a velocity of 0.069 cm/s, with movement in the right hand direction. This figure shows that link 1 and link 2 move in opposite directions of equal magnitude and thus the net movement is due to only the motion of the middle link, link 2.

From the analysis of figure 1, it can be determined that the net vertical motion is 0.004903 m in the upwards direction. Once again, the vertical motion of the right most and left most links cancel each other, and the motion is mostly due to the dynamics of the middle link.

Figure 13 illustrates the angular movement of the links for the same time period illustrated in figure 12.

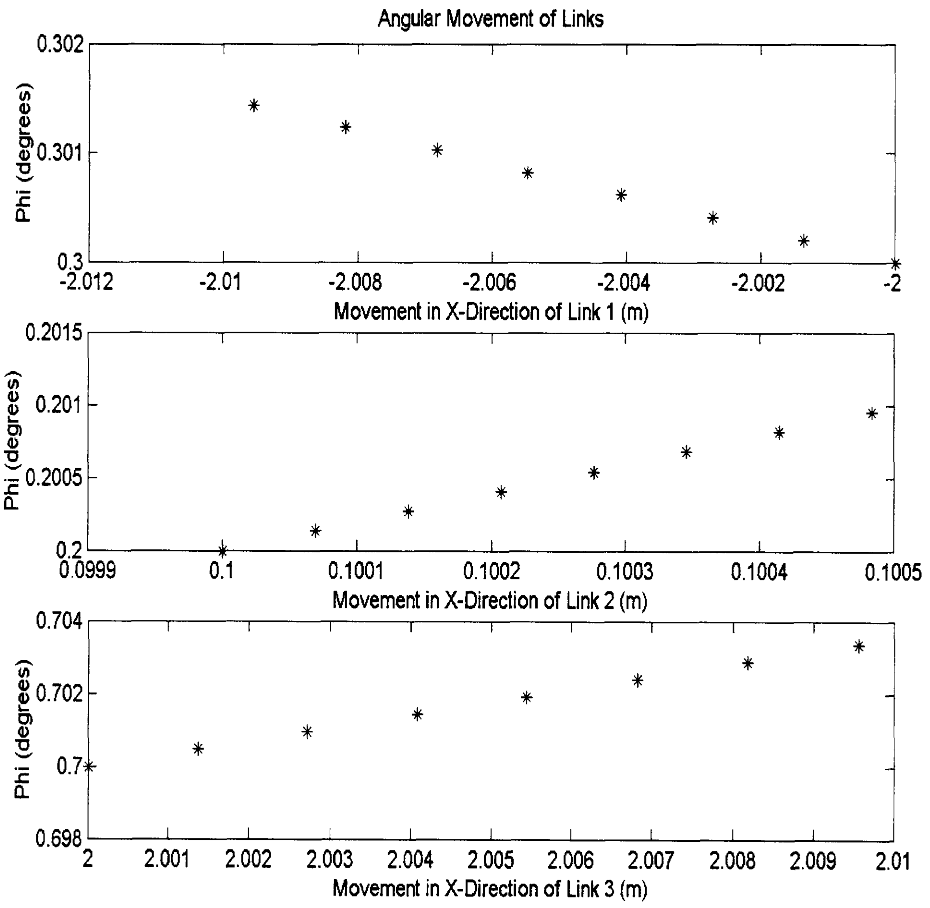


Figure 13: Comparison of Angular Movement. The top graph shows the angular movement for x position for link 1, the middle graph for link 2, and the bottom graph for link 3. All three data subplots plot data with phi increasing with increasing time.

A constraint that specified the time-dependant angle change between the links was not implemented in this version of the code. However, figures 12 and 13 show the relationship between the angles and the approximate step size change in position and angle for increasing time.

The numerical simulations show a uniform amount of change per time step. Table 1 summarizes the magnitude of change for each link in each direction for a given time step.

Table 1: Magnitude of change for one time step in each direction for each link

Link	Net Amount of Change (m)
Link 1, x direction	-0.00136
Link 1, y direction	-0.00137
Link 1, phi direction	0.000205
Link 2, x direction	6.9e-5
Link 2, y direction	6.9e-5
Link 2, phi direction	0.000136
Link 3, x direction	1.80136
Link 3, y direction	0.00206
Link 3, phi direction	0.000478

4.2 Experiments with Three-Linked Swimmer

For one flap of the swimmer, the center link was found to move about 1.27 cm (0.5") in the vertical direction, and about 0.635-0.953 (1/4"- 3/8") cm in the horizontal direction. The horizontal velocity of the swimmer was found to be 0.106 cm/s, moving 2.54 cm in 24 seconds.

4.3 Comparison Between Experimental and Numerical Analysis

It is difficult to make a direct comparison between the numerical data and the experimental data because the numerical simulation did not take into account the time dependant angle constraints. However, the magnitude of movement of the swimmer for one flap of the swimmer, 0.01 m in the horizontal direction, was similar to the magnitudes of movement predicted by the simulation. This similarity indicates that the simulations present practical data that mimics the actual movement observed during experimentation.

The numerical prediction for the velocity was found to be only 66% of the actual value. However, this discrepancy is probably due to the estimation of the drag coefficients and mass parameters. An inclusion of the time-dependant constraints would also result in less of a difference. However, the computationally predicted velocity captures the scheme of behavior very well, and provides a reliable understanding and prediction of the swimming behavior. Further refinement of input parameters would assist in the creation of an even more accurate numerical model.

4.4 Immersed Boundary Method of Analyzing Three-Linked Swimmer

The immersed boundary method program used in this study used a graphical interface to indicate the magnitude of pressure and velocity at different points along the swimmer. Blue and colors close to the blue spectrum were indicative of high magnitude and red and colors closer to red represented lower magnitudes. Green represents zero magnitude. Although the captions were written with a color copy of this study in mind, there are also included parenthetical references to the black and white image of the figures. In a black and white copy of the immersed boundary images, red appears as dark gray, while the blue spectrum appears in lighter shades of gray. Thus, in the case of a black and white copy of figures 14- 26, the reader can discern points of highest magnitude by the intensity of darkness, as the darkest areas would correspond to the areas of highest magnitude, and the lighter areas indicate areas of lower intensity.

Graphical data was collected and saved at intervals of 60 iterations in calculations. The time referenced to in the figure caption refers to the number of 60 iterations completed.

The behavior modeled in these computational simulations deviates slightly from the actual mechanical movement in that the two arms are moving simultaneously. In the actual mechanical apparatus, one link moves after the other. The computational immersed boundary program simulates the movement of the swimmer by applying a specified torque at one-third, and two-thirds of the length of the rod.

4.4.1 Pressure Distribution

Figures 14-20 illustrate the pressure distribution of the three-linked swimmer when a perturbation in the fluid is applied. Figure 14 shows the pressure distribution in the fluid around the three-linked swimmer at time=0.

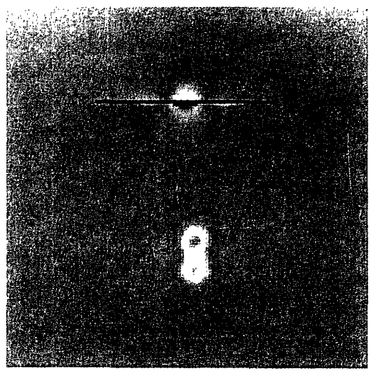


Figure 14: Pressure distribution of three-linked swimmer at t=0 when a perturbation to the fluid is applied. The red dots (the darkest points) in the middle of the figure indicate the point of perturbation.

Figure 15 shows the pressure distribution after 5 (times 60) iterations. The immediate effects of the perturbation pressure are no longer evident, but the perturbation creates torque and ensuing movement in the swimmer.

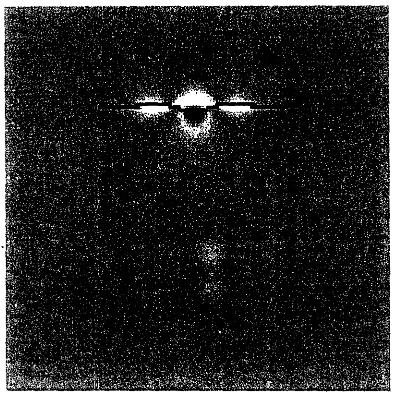


Figure 15: Pressure distribution of three-linked swimmer at t=5. At this point, the greatest pressure seems to be underneath the middle link, with some pressure in the fluid above the middle link. There is no pressure distribution in either of the two arms at this point.

Figure 16 shows the pressure distribution after 2 more iterations.

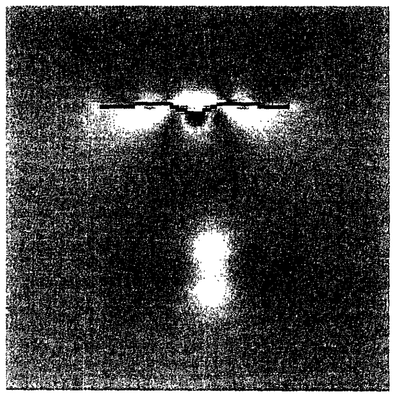


Figure 16: Pressure distribution at t=7. As an increasing amount of torque is applied at two points, small amounts of pressure are created at the undersides of the arms, as indicated by the red coloring. However, even as the arms begin to move, the greatest amount of pressure occurs on the bottom side of the middle link.

Figure 17 illustrates the pressure distribution after 8 iterations.

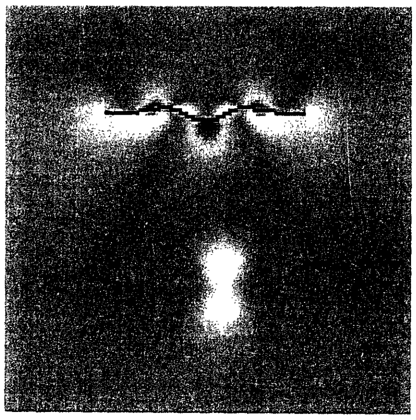


Figure 17: Pressure distribution at t=8. As the arms continue to move, the pressure distribution increases and moves toward the upper portion of the arms of the swimmer. The link on the left and right sides are moving upwards at this point.

Figure 18 is a depiction of the pressure distribution after 9 iterations. At this point in the sequence, there seems to be the most dramatic increase and change in the pressure distribution.

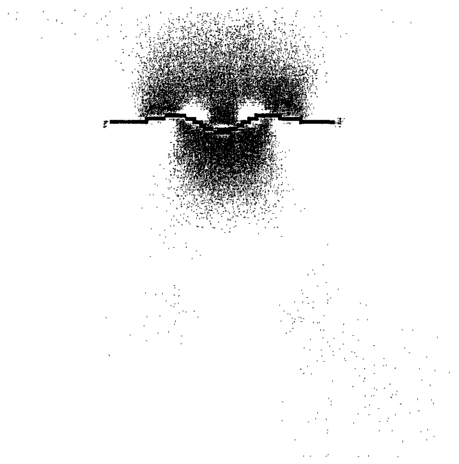


Figure 18: Pressure distribution at t=9. This figure demonstrates the continuously varying pressure distribution through the three links. The magnitude of the pressure seems to have decreased through the entire structure as the swimmer reaches a more stable configuration, as indicated by the red color (dark gray) through the arms.

Figure 19 shows the pressure distribution after 13 iterations, and figure 20 is the final graphic in this sequence, and this figure shows the pressure distribution when the swimmer reaches a steady state configuration.

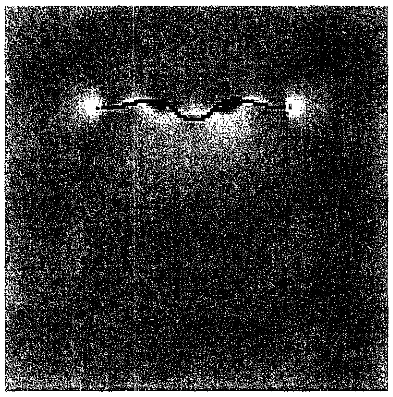


Figure 19: Pressure distribution at t=13. As the swimmer arms continue to progress upward, the pressure intensity increases, as indicated by the blue in the underside of the links..

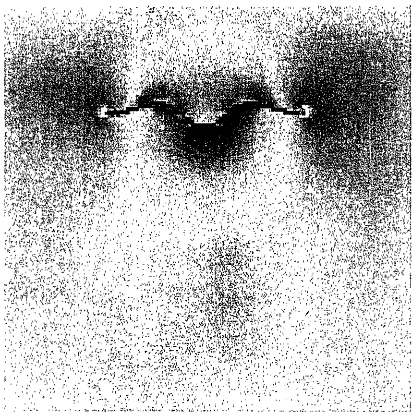


Figure 20: Pressure distribution at t=15. At this point, which is similar to the final position of the three-link swimmer, it is evident that the majority of the pressure is centered at the middle link.

The graphical data presented in this section suggests that the pressure becomes the most intense in movement, and that the center link would play an important part in determining the direction and position of motion of the entire swimmer.

4.4.2 Velocity Distribution in Horizontal Direction

The distribution of velocity in the horizontal direction is captured in figures 21-23.

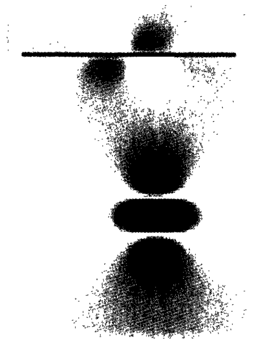


Figure 21: Horizontal velocity distribution at t=2. The blue dot in the bottom half of the figure indicates a perturbation at high velocity. The red dots on either side of the rod indicate the increasing horizontal velocity as the torque at the rods increases.

Figure 22 illustrates the velocity distribution after 9 iterations, and this figure is interesting because it shows the rapid distribution of velocity around the structure.

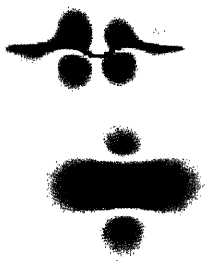


Figure 22: Horizontal velocity distribution at t=9. As the end links continue to move, horizontal velocities on alternate sides of the swimmer increase, as indicated by the red (dark gray) coloring.

Figure 23 is the last figure in this sequence, and this figure illustrates the horizontal velocity distribution at steady state conditions.

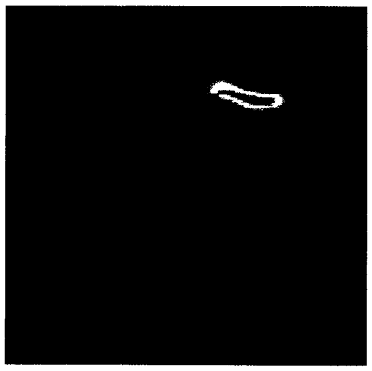


Figure 23: Horizontal velocity distribution at t=15. As the swimmer reaches a stable configuration, the swimmer is found to lean in horizontal velocity in one direction, as the right link has a higher velocity than the right.

The velocity distribution for steady state position shown in figure 10 indicates that the swimmer would move in the right hand direction, and this prediction agrees with the numerical simulation data presented in section 4.1 of this study.

4.4.3 Distribution of vertical velocity

Figures 24-26 describe the velocity distribution in the vertical direction.

Figure 25 illustrates the velocity distribution after 9 iterations.

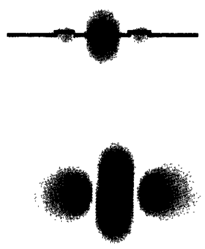


Figure 24: Velocity distribution in the vertical direction at t=3. This figure shows a parallel distribution of vertical velocity on either of the end links. There is a comparatively smaller horizontal velocity on the middle ink. The blue dot (gray) in the center indicates the perturbation in the fluid.

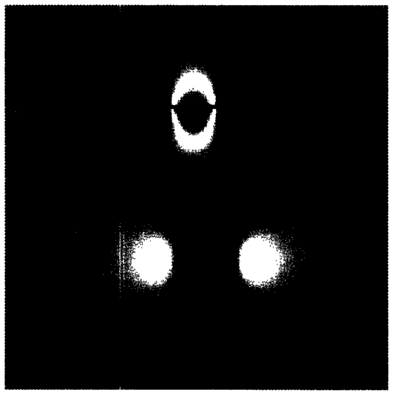


Figure 25: Velocity distribution in the vertical direction at t=9. The vertical velocity on the end links is shown to increase for progressing time as the color has turned blue, while the vertical velocity in the middle link does not increase in magnitude, but the sphere of the fluid affected by the middle link gets larger.

Figure 26 shows the vertical velocity distribution when the structure has reached a steady state position.

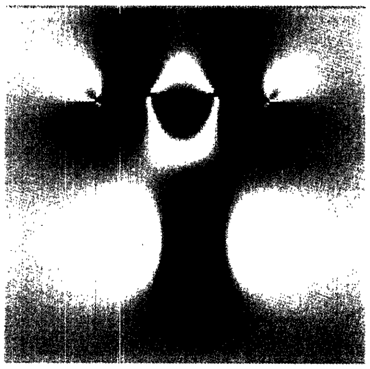


Figure 26: Vertical velocity distribution at t=15. At the stable configuration, the vertical velocity at the ends is shown to increase in sphere as well as magnitude, and the middle link is also shown to have an increasing influence on the surrounding fluid.

4.5 Experiments with Flexible Armed Three-Linked Swimmer

The flexible armed three linked swimmer was found to be much more efficient in performance than the rigid three-linked swimmer. On average, this construction was found to swim at an average speed of 0.332 cm/s in the horizontal direction. The distance of travel in the vertical direction was once again found to reach a maximum of 2.54 cm.

However, even this apparatus contained some rigidity in its construction. The arms were attached by rigid clips, and because of the overall small size of the swimmer, this accounted for about one-fourth of the total surface area.

4.6 Comparison of Flexible Armed, Rigid Armed Three-Linked Swimmer, and Numerical Predictions

The flexible armed swimmer was found to be 3.13 times faster than the rigid armed swimmer. The speed of the flexible armed swimmer was 4.8 times faster than the numerical simulation data presented. The large difference between the numerical simulation data and the flexible armed swimmer experimental values indicate that the model presented in this study applies only to the rigid armed swimmer. The experimental data suggests that the analytical model presented in this study must be modified with other constraints and conditions in order to pertain to the behavior of the flexible armed swimmer.

Figure 27 illustrates the difference between the movement of the flexible armed and rigid armed three-linked swimmer.

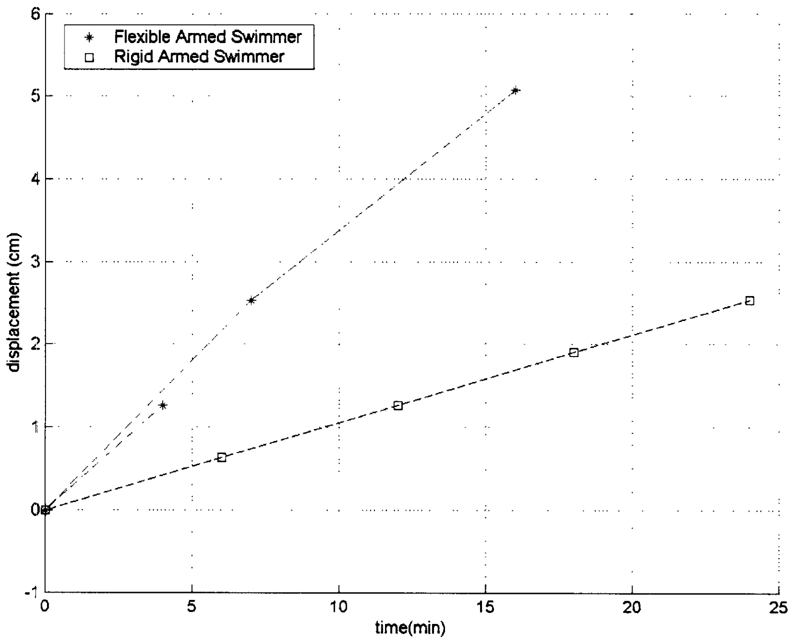


Figure 27: Difference between rigid armed and flexible armed swimmer. The dotted lines represent the linear regression lines, which were added to underscore the dramatic difference in slope between the two lines. The red stars represent data collected for the flexible armed

swimmer, and data from two separate collections is given to corroborate the repeatability of data point collection. The rigid armed swimmer is shown to have more uniform velocity, while the flexible armed swimmer is more susceptible to change.

4.7 Sources of Error

The data presented in this section suggest that the theoretical model and the experimentation conducted are in harmony with each other. However, the numerical model could be refined to greater accuracy if the drag coefficients had been determined with a greater degree of accuracy. Although the drag coefficients used in the numerical code are fairly accurate estimates, using experimental and analytical techniques to find drag coefficients could result in a model with even higher accuracy.

As mentioned earlier, an inclusion of constraints that consider the time-dependant angle change between the links, as indicated in the theoretical analysis, would also result in a model with even higher accuracy.

As previously discussed, the immersed boundary computer program used in this study simulated the motion of the three-linked swimmer by applying a torque simultaneously at one-third and two-thirds of the length of the rod. In other words, the graphical simulation presented in this study considers a three-linked swimmer that moves by flapping both links at the same time, which is a behavior that differs from the mechanical apparatus used in this study. Although the pressure and velocity distributions obtained in this study would vary slightly from the fluid-structure interactions seen when only one link is moving, the graphical data procured does apply because it still illustrates the overall trend of movement.

The experimental data was difficult to collect in some respects for long periods of time because the mechanical apparatus had to be wound up with a screwdriver, and often quickly ran out of "power." Future experiments examining the swimmer for longer periods of time or many more sets of collections would add validity to the experimental data presented in this study.

5. CONCLUSIONS

The analytical model presented seems to capture the overall trends of the three-linked swimmer described in this study. Predictions for movement in the x, y, and phi angle directions for each link were determined using numerical simulations. The numerical prediction for velocity in the x-direction, 0.069 cm/s was found to be 66% of the value derived from the experimental data, 0.106 cm/s. However, a close examination of the numerical trends in the computational model validates the use of this model to study the behavior of the three-linked swimmer. The numerical data also indicates that the motion of the middle link is responsible for net motion. The movement of the end links was shown to move in nearly equal and opposite directions, canceling the movement of the end links. The flexible armed swimmer model proved to be the more efficient swimmer than a rigid construction, being more than three times faster than the rigid armed swimmer, with a velocity of 0.332 cm/s. The immersed boundary computer program enabled a close study of the pressure and velocity distributions at various points in movement.

A further refinement of this analytical model would include the addition of the time-dependant angle constraints and more precise drag coefficients. The data in this study suggest that flexible material work much more efficiently in viscous fluid than their rigid counterparts. It would be interesting to conduct future experiments with flexible arms in the three-linked swimmer using arms of varying sizes and thickness to further understand the mechanics of these phenomena and to find the conditions for optimal performance. It would be interesting to continue further study on the differences between the two types of arms, flexible and rigid, because of the industrial applications to which these conclusions could be applied, which include airplane wings.

REFERENCES

[1] Becker, L.E., Koehler, S.A., & Stone, H.A. 2003 On Self-Propulsion of Micro-Machines at Low Reynolds Number: Purcell's Three-Link Swimmer. *J. Fluid Mech.* **490**, 15-35.

Analyzes the three-link swimmer first discussed by Purcell; the simplest organism that can swim in a viscous medium.

[2] Berg, H.C. 2002 How Spiroplasma might swim. *J. Bacteriol.* **184,** 2063-2064.

"Proposes that small helical bacteria called Spiroplasmas might propel themselves in a manner similar to Purcell's swimmer but with non-parallel axes of bending for the two idealized joints."

[3] Bonner, J.T. and T.A. McMahon. (1983). *On Size and Life.* New York: Scientific American Library.

Discusses effects of organism size in movement and other facets of life.

[4] Camalet, S., Julicher, F. & Prost, J. 1999 Self-organized beating and swimming of internally driven filaments. *Phys. Rev. Lett.* **82,** 1590-1593.

Studies flapping filaments with various boundary conditions to understand the mechanics of motion

[5] Chan, Brian. "Propulsion in Viscous Fluids: Purcell's Three-Link Swimmer," [Online document], 2002, [2003 September 5], Available HTTP: http://web.mit.edu/chosetec/www/robo/3link.

Mechanical model of Purcell's three link swimmer

[6] Eyre, D.J. and Fogelson, A.L. "IBIS: Immersed Boundary and Interface Software, A User Guide," [Online document], 1997, [2003 November 3], Available HTTP: http://www.math.utah.edu/IBIS.

An implementation to perform the immersed boundary method which requires little knowledge of fluid dynamics on the part of the user.

[7] Goldstein, R.E., Powers, T.R. & Wiggins, C.H. 1998. Viscous nonlinear dynamics of twist and writhe. *Phys. Rev. Lett.* **80,** 5232-5235.

Using geometric kinematics, viscous dynamics governed by elasticity, examines the mechanisms of supercoiling, as those exhibited by the *B. subtilis* bacteria.

[8] Gray, J. *How Animals Move.* (1953). Cambridge University Press: New York.

Discusses through illustrations and basic physics the reasons behind animal movement.

[9] Gueron, S & Levit-Gurevich, K. 1999. Energetic considerations of ciliary beating and the advantage of metachronal consideration. *Proc Natl Acad Sci USA* **22,** 12240-12245.

[10] Koehler, S.A. & Powers, T.R. 2000 Twirling elastica: Kinks, viscous drag and torsional stress. *Phys. Rev. Lett.* **85,** 4827-4830.

Looks at the relationships between viscous drag, twisting, and bending, as affecting a rotating rod with a kink submerged in fluid.

[11] Peskin, C.S. 2002. The immersed boundary method. *Acta Numerica*, 1-39.

Discusses the basic principles and detailed mathematics of the immersed boundary method.

[12] Peskin, C.S. and McQueen, D.M. A general method for the computer simulation of biological systems interacting with fluids. Biological Fluid Dynamics, The Company of Biologists Limited, Cambridge UK, 1995, pp. 265-276.

A tutorial for a user of the immersed boundary method.

[13] Purcell, E.M. *Life at Low Reynolds Number.* 1976 Lyman Laboratory, Harvard University, Cambridge, Mass 02138

Describes the non-reciprocal motion necessary for movement in viscous fluids, proposes that the simplest organism that can move in such an environment is one that has a three-linked mechanism.

[14] Purcell, E.M. 1997 The efficiency of propulsion by a rotating flagellum. *Proc Natl Acad Sci* **94,** 11307-11311

Article regarding helical propulsion.

[15] Taylor, G.I. 1951 Analysis of the swimming of microscopic organisms. *Proc. R. Soc. Lond.* **209** 447-461

Discusses motion of a waving sheet in fluid to model the movement of microscopic organisms with tails.

[16] Vella, D., Kim, H.Y., and Mahadevan, L. The wall-induced motion of a floating flexible train. *J. Fluid Mech.* **772**, 1-10.

Paper from which the model for the three-linked swimmer in this study was developed.

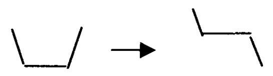
[17] White, F.M. (1999). *Fluid Mechanics.* Boston: McGraw-Hill.

Basic fluid mechanics reference

[18] Zhang, Childress, Libchaber, and Shelley. 2000. Flexible filaments in a flowing soap film as a model for 1-D flags in a 2-D wind. *Nature* **408,** 835.

APPENDIX

A1. Table of Literature Research on Viscous Fluid Swimmers

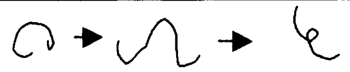
Geometry	Types of Studies Conducted	Formulas	Findings	Reference
Purcell's 3-Link	Analytical	**(1) Up=Uj+Ω x r + ω x r**; solve into Stokes eqns.; (2) substitute into resistance matrix for force, (3) velocity, and stretching components (4); substitute certain values to find points at certain conditions	(1) slender-body approximation;(2) at moderate stroke angles, direction is opposite to that of the wave velocity; (3) for larger stroke amplitudes, the direction of net translation reverses	Becker, Koehler, Stone "On Self-Propulsion of micro-machines at low Reynolds numbers: Purcell's three-link swimmer"
	Numerical		Displacements in x-direction (fig. 5); displacements of centre and joint for certain angles (fig 6); Velocity (fig 9); Efficiency (fig 10,11)	Becker, Koehler, Stone "On Self-Propulsion of micro-machines at low Reynolds numbers: Purcell's three-link swimmer"
	Efficiency	**ε=(η+2)ξaV^2/Φ** (power to pull the straightened swimmer at average speed/ average mechanical power to achieve the speed) [η=(b/a), ζ=resistance ratio, ϕ=power]	.77%; compared to undulating rod (7.4%) and rotating helix (8.6%)	Becker, Koehler, Stone "On Self-Propulsion of micro-machines at low Reynolds numbers: Purcell's three-link swimmer"
	Mechanical./ Experimental		Mechanical 3-link swimmer	Brian Chan, MIT
	Biological Analogy	spiroplasmas		Berg, "How Spiroplasma might swim"
	Pictoral			

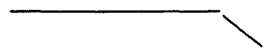
Rotating Helix	Analytical	Similar to the manner in which the Becker, Koehler, Stone paper solved for components. From Stokes equations, derived a resistance (propulsion) matrix for the coefficients of a system of two equations with force and torque. Can solve for the swimming (dropping) speed and speed of rotation in this way	At low Reynolds numbers, a translating force will causes a helix to rotate even without an applied external torque, and by the linearity of the Stokes equations, an applied torque will cause the helix to rotate	Purcell, "The efficiency of propulsion by a rotating flagellum"[14]
	Efficiency	Compare power output of the motor to the least power that would be req'd to move the cell at a speed v	ε=Av/NΩ^2; for specific cases, can drop certain terms to approximate	Purcell, "The efficiency of propulsion by a rotating flagellum"[14]
	Mechanical./ Experimental	Create a model (twisted steel wire) of varying lengths, and let drop, with its own weight, in a fluid of silicon oil. Measure sinking speed and speed of rotation.	The most efficient is when it sinks at W/"A" where W is the weight of the model	Purcell, "The efficiency of propulsion by a rotating flagellum"[14]
	Biological Analogy	*Escherichia coli*		
	Pictoral			

Waving Sheet	Analytical	(1) waving surface: **y=bsin(kx-st)**; (2) use stream fnct to describe flow near a sheet down small amplitudes : ψ=− **bσ/k(1+ky)e-ky*sin(kx-σt)**; (3) work/area: **W=2b^2σ^2*kμ**; (4) total mean force zero; (5) paper expands for higher orders	(1) non-reciprocal motion explains self-propulsion; (2) when these small waves travel down, the sheet is propelled at a rate 2π^2*b^2/λ^2 times the wave velocity and in the opposite direction to that of the propulsion of the waves; (3) two tails close to each other will swim in sync (the effort req'd to make the tails wave in unison is 1/1000 that to make them swim if out of phase); (4) however, this is only a 2D model to describe 3D behavior, which has hydrodynamic interaction	Taylor, "Analysis of the swimming of microscopic organisms." [15]
	Biological Analogy	*Spirochaeta balbianii* (microscopic organisms with tails)		Grey. *Ciliary movement* (1928) [8]
	Pictoral			
	Future Possible Work /Notes		3D behavior using immersed boundary methods? (the fluid motion coupled to the organism motion)	

1-D Flags in 2-D Wind	Mechanical./ Experimental	Nylon threads in tension in a stream of soap film	At a certain critical filament length, the filament is not affected by the flow. However, past this point, the oscillatory state can occur. And then when L becomes very long, the waving becomes an end effect. **Paper discusses the interaction of the fluid with the filament. "At the transition point to flapping, the mass of the filament balances the mass of the interacting fluid, and the elastic energy of the filament balances the kinetic energy of the fluid.	Zhang, Childress, Libchaber, and Shelley. "Flexible filaments in a flowing soap film as a model for 1-D flags in a 2-D wind." [18]
	Biological Analogy	Swimming fish, flapping flags		

Elastic Filament subject to internally generated stresses; wavelike propagating stresses	Analytical	(1) Enthalpy functional **G=int(curvature+internal force density/sliding displacement +incompressibility)** (2) Solving eqns for an oscillating force density, in terms of the displacements *h* h(x.t)=H(x)cos(ωt-ϕ(x))	(1) Only filament rigidity and hydrodynamic friction (elastic properties) responsible for motion, not internal mechanisms of force generation; (2) boundary conditions imposed at the end dictate the type of motion (beating pattern)	Camalet, Julicher, Proust "Self-Organized Beating and Swimming of Internally Driven Filaments" [4]
	Numerical		Plot amplitude as a function of the position x along the filament axis	Camalet, Julicher, Proust "Self-Organized Beating and Swimming of Internally Driven Filaments" [4]
	Biological Analogy	Cilia and flagella; using refined slender body theory for cilia moving in Stokes flow and take hydrodynamical effects into account. Geometric equations for motion of the ciliary as a function of velocity. Internal bend mechanism equations.	Used experimental data from Sleigh to compare numerical simulations of position. Effective/recovery stroke encounter different resistances. The mechanical work done for a single cilium, the effective stroke is 5 times more work than the recovery stroke. Mulitiluciliary configurations are advantageous and perhaps necessary for movement. (calculate energy expenditure by W=Fx relation)	Gueron and Levi-Gurevich. "Energetic considerations of ciliary beating and the advantage of metachronal consideration" [9]
	Pictoral	3 types: (A) Clamped head, position and sloped fixed. (B) Fixed head; position fixed only (C) Free head subject to viscous load		Camalet, Julicher, Proust "Self-Organized Beating and Swimming of Internally Driven Filaments" [4]

	Future Possible Work /Notes		(1) Think of case of nonlinearities and torsional motion for 3D (2) only internal stresses are considered in this study; other papers cited however look at semiflexible filaments subject to external forces; (3) experiments dealing with applying external forces and torques to ends of flagella could test the theory presented here	
Twisted Elastic Filament in Viscous Fluid	Analytical	(1) Elastic energy: bend+twist-extensibility, **A/2int(ds(Ω1^2+Ω2^2))+C/2int(dsΩ3^2)-int(Λds)**; (2) then use force and moment relns to energy to find those eqns. (3) Drag forces and moments balance elastic forces and moments, thus find complex curvature and twist. (4) Link=Twist+Writhe	Kinematics dictated by geometry --> viscous dynamics governed by elasticity --> describes supercoiling.	Goldstein, Powers, Wiggins, "Viscous Nonlinear Dynamics of Twist and Writhe" [7]
	Numerical	fig (2,3) integrate curvature and twist equations for different velocities to find physical condition	(fig 4) Details of geometric untwisting. Twist density evolution; time evolution of twist, bend, and total elastic energy. (fig 2.,3) illustrations of filament shapes at various twist densities, and geometric untwisting	Goldstein, Powers, Wiggins, "Viscous Nonlinear Dynamics of Twist and Writhe" [7]
	Biological Analogy	Bacterium *B. subtilis*		N.H. Mendelson *et al*, J. Bacteriolo. **177**, 7060 (1995)
	Pictoral			

Rotating Rod with a Kink	Analytical	Use elastic internal stresses to find moment and force: force balances to find dimensional shape of the rod	Interplay between viscous drag, twisting, and bending in dynamics of rotating filaments	Koehler, Powers. "Twirling Elastica: Kinks, Viscous Drag, and Torsional Stress" [10]
	Numerical		Phase diagram as a function of kink angle, kink position, and rotation	Koehler, Powers. "Twirling Elastica: Kinks, Viscous Drag, and Torsional Stress" [10]
	Mechanical / Experimental	Steel compression spring immersed in glycerol rotating with motor speeds of 5-500 rpm	As the rotation rate increases, rods with a small-moderate angle (L-shaped) will extend, and rods with a larger angle (V-shaped) will fold	Koehler, Powers. "Twirling Elastica: Kinks, Viscous Drag, and Torsional Stress" [10]
	Biological Analogy	natural curvature in bacteria, protein-binding in DNA, *B. subtilis*		
	Pictoral			
	Future Possible Work /Notes	Look at Brownian dynamic simulations with a rotating flexible rod with many kinks		

A2. C++ Code used to Initialize Variables and Determine Equations of Motion only for first time the simulation is run

```
//code1.cpp

#include <iostream.h>
#include <stdlib.h>
#include<conio.h>
#include<math.h>
#include<stdio.h>
#include<fstream.h>

//link 1: the left, link2: the center, link 3: the right

void defineInitial(double q[9]);
void findForce(double f[9], double q[9]); //calculates the
drag force on the links
void findB(double b[9], double q[9]);
void createJ(double q[9], double J[9][4]);
void makeM(double M[9][9]);

double dt=1.5; //change in time to determine the velocity and
acceleration for the given position
double dragpar=44, dragperp=55, dragrot=50; //the force drag
coefficients
double a=0.3;
double m=0.5, I=0.6;

int main() //make size of vectors x*y in for loops (from
smaller functions)
{
    double q[9]={0}; //describes the positions and
orientations of the rods
    double f[9]={0}; //describes the forces
    double lambda[4]={0}; //defines lambdas
    double J[9][4]={0};
    double b[9]={0};
    double qddot[9]={0};
    double M[9][9]={0};
    defineInitial(q);

    findForce(f,q);
    findB(b,q);
    createJ(q,J);

    ofstream output;
    output.open("c:/THESIS/3Link/q.txt", ios::out);

    for(int x=1; x<10; x++)
    {
     output << q[x] << endl;
     cout << q[x] << endl;
    }

    getch();
    return 0;
}

void makeM(double M[9][9])
{

 M[1][1]=m;
 M[2][2]=m;
 M[3][3]=I;
 M[4][4]=m;
 M[5][5]=m;
 M[6][6]=I;
 M[7][7]=m;
 M[8][8]=m;
 M[9][9]=I;

}

void createJ(double q[9], double J[9][4])
{
 ofstream outputa;
 outputa.open("c:/THESIS/3Link/J1.txt", ios::out);
 ofstream outputb;
 outputb.open("c:/THESIS/3Link/J2.txt", ios::out);
 ofstream outputc;
 outputc.open("c:/THESIS/3Link/J3.txt", ios::out);
 ofstream outputd;
 outputd.open("c:/THESIS/3Link/J4.txt", ios::out);
 ofstream outpute;
 outpute.open("c:/THESIS/3Link/J5.txt", ios::out);
 ofstream outputf;
 outputf.open("c:/THESIS/3Link/J6.txt", ios::out);
 ofstream outputg;
 outputg.open("c:/THESIS/3Link/J7.txt", ios::out);
 ofstream outputh;
 outputh.open("c:/THESIS/3Link/J8.txt", ios::out);
 ofstream outputi;
 outputi.open("c:/THESIS/3Link/J9.txt", ios::out);

 J[1][1]=-1;
 J[3][1]=a*sin(q[3]);
 J[4][1]=1;
 J[6][1]=a*sin(q[6]);
 J[2][2]=-1;
 J[3][2]=-1*a*cos(q[3]);
 J[5][2]=1;
 J[6][2]=-1*a*cos(q[6]);
 J[4][3]=-1;
 J[6][3]=a*sin(q[6]);
 J[7][3]=1;
 J[9][3]=a*sin(q[9]);
 J[5][4]=-1;
 J[6][4]=-1*a*cos(q[6]);
 J[8][4]=1;
 J[9][4]=-1*a*cos(q[9]);
 for(int x=1; x<5; x++)
 {
  outputa << J[1][x] << endl;
 }
 for(int x=1; x<5; x++)
 {
  outputb << J[2][x] << endl;
 }
 for(int x=1; x<5; x++)
 {
  outputc << J[3][x] << endl;
 }
 for(int x=1; x<5; x++)
 {
  outputd << J[4][x] << endl;
 }
 for(int x=1; x<5; x++)
 {
  outpute << J[5][x] << endl;
 }
  for(int x=1; x<5; x++)
 {
  outputf << J[6][x] << endl;
 }
  for(int x=1; x<5; x++)
 {
  outputg << J[7][x] << endl;
 }
  for(int x=1; x<5; x++)
 {
  outputh << J[8][x] << endl;
 }
  for(int x=1; x<5; x++)
 {
  outputi << J[9][x] << endl;
 }
}

void findB(double b[9], double q[9])
{
 for(int i=1; i<=9; i++)
 {

b[1]+=((q[i]/dt)*(a*cos(q[3]))*(q[3]/dt))+((q[i]/dt)*(a*cos(q[
6]))*(q[6]/dt)); //for when alpha is one

b[2]+=((q[i]/dt)*(a*sin(q[3]))*(q[3]/dt))+((q[i]/dt)*(a*sin(q[
6]))*(q[6]/dt));

b[3]+=((q[i]/dt)*(a*cos(q[6]))*(q[6]/dt))+((q[i]/dt)*(a*cos(q[
9]))*(q[9]/dt));

b[4]+=((q[i]/dt)*(a*sin(q[6]))*(q[6]/dt))+((q[i]/dt)*(a*sin(q[
9]))*(q[9]/dt));
 }
 ofstream outpute;
 outpute.open("c:/THESIS/3Link/b.txt", ios::out);
 for(int x=1; x<5; x++)
 {
  outpute << b[x] << endl;
 }
}

void findForce(double f[9], double q[9])
{
 f[1]=-
1*(q[1]/dt)*(dragpar*cos(q[3])*cos(q[3])+dragperp*sin(q[3])*si
n(q[3]))+(q[2]/dt)*((dragperp-dragpar)*sin(q[3])*cos(q[3]));
 f[2]=(q[1]/dt)*((dragperp-dragpar)*sin(q[3])*cos(q[3]))-
(q[2]/dt)*(dragpar*sin(q[3])*sin(q[3])+dragperp*cos(q[3])*cos(
q[3]));
 f[3]=-1*dragrot*(q[3]/dt);
 f[4]=-
1*(q[4]/dt)*(dragpar*cos(q[6])*cos(q[6])+dragperp*sin(q[6])*si
n(q[6]))+(q[5]/dt)*((dragperp-dragpar)*sin(q[6])*cos(q[6]));
 f[5]=(q[4]/dt)*((dragperp-dragpar)*sin(q[6])*cos(q[6]))-
(q[5]/dt)*(dragpar*sin(q[6])*sin(q[6])+dragperp*cos(q[6])*cos(
q[6]));
 f[6]=-1*dragrot*(q[6]/dt);
 f[7]=-
1*(q[7]/dt)*(dragpar*cos(q[9])*cos(q[9])+dragperp*sin(q[9])*si
n(q[9]))+(q[8]/dt)*((dragperp-dragpar)*sin(q[9])*cos(q[9]));
 f[8]=(q[7]/dt)*((dragperp-dragpar)*sin(q[9])*cos(q[9]))-
(q[8]/dt)*(dragpar*sin(q[6])*sin(q[9])+dragperp*cos(q[9])*cos(
q[9]));
 f[9]=-1*dragrot*(q[9]/dt);

 ofstream outputf;
 outputf.open("c:/THESIS/3Link/f.txt", ios::out);
 for(int x=1; x<10; x++)
 {
  outputf << f[x] << endl;
 }
}

void defineInitial(double q[9]) //set initial conditions, in
the order x1, y1, phi1, x2, y2,phi2, x3,y3,phi3 stored in the
vector q
{
  q[1]=-2;
  q[2]=-2;
  q[3]=0.3; //degrees must be in radians
  q[4]=0.1;
  q[5]=0.1;
  q[6]=0.2;
  q[7]=2;
  q[8]=3;
  q[9]=0.7;
}
```

A3. C++ Code used to Initialize Variables and Determine Equations of Motion after first run (used in place of code1.cpp)

```
//code2.cpp

#include <iostream.h>
#include <stdlib.h>
#include<conio.h>
#include<math.h>
#include<stdio.h>
#include<fstream.h>

void readFile(double temp[9]);
void defineInitial(double q[9]);
void increase(double Y[18], double F[18], double temp[9]);
//solve ODE using 2nd order Runge-Kutta Method
void function(double F[18], double temp[9], double Y[18]);
void defineInitial2(double q[9]);

double dt=1.5;
double dragpar=44, dragperp=55, dragrot=50; //the force drag
coefficients

int main()
{
 //define variables
 double temp[9]={0};
 double q[9]={0}, Y[18]={0}, F[18]={0};

// defineInitial(q); //first time
 defineInitial2(q); //second time
 readFile(temp);

 //determine initial conditions
 for(int x=1; x<10; x++)
 {
  Y[x]=q[x]/dt;
 }
 ofstream outfiler;
 outfiler.open("c:/THESIS/3Link/position.txt", ios::out);

 increase(Y,F,temp);

 cout << "Position is: ";
 for(int x=10; x<19; x++)
 {
  cout << q[x-9] <<" " <<  Y[x] << " " << q[x-9]+Y[x] <<
endl;
  outfiler << Y[x]+q[x-9] << endl;
 }

 getch();
 return 0;
}
```

```
void readFile(double temp[9])
{
 ifstream infile("c:/THESIS/3Link/data.txt");
 for(int i=0; i<9; i++)
 {
  infile >> temp[i];
 }
}

void defineInitial2(double q[9])
{
 ifstream infile("c:/THESIS/3Link/q.txt");
 for(int i=1; i<10; i++)
 {
  infile >> q[i];
 }
}

void defineInitial(double q[9]) //set initial conditions, in
the order x1, y1, phi1, x2, y2,phi2, x3,y3,phi3 stored in the
vector q
{
  q[1]=-2;
  q[2]=-2;
  q[3]=0.3; //degrees must be in radians
  q[4]=0.1;
  q[5]=0.1;
  q[6]=0.2;
  q[7]=2;
  q[8]=3;
  q[9]=0.7;
}

void function(double F[18], double temp[9], double Y[18])
{
 double h=0.001;

 F[1]=h*(temp[1]-(-
1*Y[1]*(dragpar*cos(Y[12])*cos(Y[12])+dragperp*sin(Y[12])*sin(
Y[12])))+(Y[2]*((dragperp-dragpar)*sin(Y[12])*cos(Y[12]))));
 F[2]=h*(temp[2]-((Y[1]*((dragperp-
dragpar)*sin(Y[12])*cos(Y[12]))-
(Y[2])*(dragpar*sin(Y[12])*sin(Y[12])+dragperp*cos(Y[12])*cos(
Y[12])))));
 F[3]=h*(temp[3]-(-1*dragrot*(Y[3])));
 F[4]=h*(temp[4]-((-
1*(Y[4])*(dragpar*cos(Y[15])*cos(Y[15])+dragperp*sin(Y[15])*si
n(Y[15]))+(Y[5])*((dragperp-
dragpar)*sin(Y[15])*cos(Y[15])))));
 F[5]=h*(temp[5]-((Y[4])*((dragperp-
dragpar)*sin(Y[15])*cos(Y[15]))-
(Y[5])*(dragpar*sin(Y[15])*sin(Y[15])+dragperp*cos(Y[15])*cos(
Y[15]))));
 F[6]=h*(temp[6]-(-1*dragrot*(Y[6])));
 F[7]=h*(temp[7]-(-
1*(Y[7])*(dragpar*cos(Y[18])*cos(Y[18])+dragperp*sin(Y[18])*si
n(Y[18]))+(Y[8])*((dragperp-dragpar)*sin(Y[18])*cos(Y[18]))));
 F[8]=h*(temp[8]-((Y[7])*((dragperp-
dragpar)*sin(Y[18])*cos(Y[18]))-
(Y[8])*(dragpar*sin(Y[18])*sin(Y[18])+dragperp*cos(Y[18])*cos(
Y[18]))));
 F[9]=h*(temp[9]-(-1*dragrot*(Y[9])));
 F[10]=h*Y[1];
 F[11]=h*Y[2];
 F[12]=h*Y[3];
 F[13]=h*Y[4];
 F[14]=h*Y[5];
 F[15]=h*Y[6];
 F[16]=h*Y[7];
 F[17]=h*Y[8];
 F[18]=h*Y[9];
}

void increase(double Y[18], double F[18], double temp[9])
//solve ODE using 2nd order Runge-Kutta Method
{
 double Nsteps=6, notconverged=0; //the number of steps,
convergence test
 double t=0, tf=3, h; //the starting time, the ending time,
and the increment variable
 double Z[18]={0}; //holds temporary values
 double Yold[18]={0}; //old Y
 function(F,temp,Y);
 for(int i=1; i<19; i++)
 {
  Z[i]=Y[i]+0.5*F[i]; //Y is the initial condition; F is the
force
 }
 function(F,temp,Z);
 for(int i=1; i<=18; i++)
 {
  Y[i]=Y[i]+F[i]; //new velocity
 }
}
```

A4. C++ Code Used to Solve Equations of Motion Using Runge-Kutta Method

```
//code3.cpp

#include <iostream.h>
#include <stdlib.h>
#include<conio.h>
#include<math.h>
#include<stdio.h>
#include<fstream.h>

//link 1: the left, link2: the center, link 3: the right
```

```
void findForce(double f[9], double q[9]); //calculates the
drag force on the links
void findB(double b[9], double q[9]);
void createJ(double q[9], double J[9][4]);
void makeM(double M[9][9]);

double dt=1.5; //change in time to determine the velocity and
acceleration for the given position
double dragpar=44, dragperp=55, dragrot=50; //the force drag
coefficients
double a=0.3;
double m=0.5, I=0.6;

int main() //make size of vectors x*y in for loops (from
smaller functions)
{
    double q[9]={0}; //describes the positions and
orientations of the rods
    double f[9]={0}; //describes the forces
    double lambda[4]={0}; //defines lambdas
    double J[9][4]={0};
    double b[9]={0};
    double qddot[9]={0};
    double M[9][9]={0};
    double t=0;

    ifstream infile("c:/THESIS/3Link/position.txt");
    for(int x=1; x<10; x++)
    {
     infile >> t;
     q[x]+=t;
    }

    findForce(f,q);
    findB(b,q);
    createJ(q,J);

    ofstream output;
    output.open("c:/THESIS/3Link/q.txt", ios::out);

    for(int x=1; x<10; x++)
    {
     output << q[x] << endl;
     cout << q[x]<< endl;
    }

    getch();
    return 0;
}

void makeM(double M[9][9])
{

 M[1][1]=m;
 M[2][2]=m;
 M[3][3]=I;
 M[4][4]=m;
 M[5][5]=m;
 M[6][6]=I;
 M[7][7]=m;
 M[8][8]=m;
 M[9][9]=I;

}

void createJ(double q[9], double J[9][4])
{
 ofstream outputa;
 outputa.open("c:/THESIS/3Link/J1.txt", ios::out);
 ofstream outputb;
 outputb.open("c:/THESIS/3Link/J2.txt", ios::out);
 ofstream outputc;
 outputc.open("c:/THESIS/3Link/J3.txt", ios::out);
 ofstream outputd;
 outputd.open("c:/THESIS/3Link/J4.txt", ios::out);
 ofstream outpute;
 outpute.open("c:/THESIS/3Link/J5.txt", ios::out);
 ofstream outputf;
 outputf.open("c:/THESIS/3Link/J6.txt", ios::out);
 ofstream outputg;
 outputg.open("c:/THESIS/3Link/J7.txt", ios::out);
 ofstream outputh;
 outputh.open("c:/THESIS/3Link/J8.txt", ios::out);
 ofstream outputi;
 outputi.open("c:/THESIS/3Link/J9.txt", ios::out);

 J[1][1]=-1;
 J[3][1]=a*sin(q[3]);
 J[4][1]=1;
 J[6][1]=a*sin(q[6]);
 J[2][2]=-1;
 J[3][2]=-1*a*cos(q[3]);
 J[5][2]=1;
 J[6][2]=-1*a*cos(q[6]);
 J[4][3]=-1;
 J[6][3]=a*sin(q[6]);
 J[7][3]=1;
 J[9][3]=a*sin(q[9]);
 J[5][4]=-1;
 J[6][4]=-1*a*cos(q[6]);
 J[8][4]=1;
 J[9][4]=-1*a*cos(q[9]);

 for(int x=1; x<5; x++)
 {
  outputa << J[1][x] << endl;
 }
 for(int x=1; x<5; x++)
 {
  outputb << J[2][x] << endl;
 }
 for(int x=1; x<5; x++)
 {
  outputc << J[3][x] << endl;
 }
 for(int x=1; x<5; x++)
 {
  outputd << J[4][x] << endl;
 }
 for(int x=1; x<5; x++)
 {
  outpute << J[5][x] << endl;
 }
  for(int x=1; x<5; x++)
 {
  outputf << J[6][x] << endl;
 }
  for(int x=1; x<5; x++)
 {
  outputg << J[7][x] << endl;
 }
  for(int x=1; x<5; x++)
 {
  outputh << J[8][x] << endl;
 }
  for(int x=1; x<5; x++)
 {
  outputi << J[9][x] << endl;
 }
}

void findB(double b[9], double q[9])
{
 for(int i=1; i<=9; i++)
 {

b[1]+=((q[i]/dt)*(a*cos(q[3]))*(q[3]/dt))+((q[i]/dt)*(a*cos(q[
6]))*(q[6]/dt)); //for when alpha is one

b[2]+=((q[i]/dt)*(a*sin(q[3]))*(q[3]/dt))+((q[i]/dt)*(a*sin(q[
6]))*(q[6]/dt));

b[3]+=((q[i]/dt)*(a*cos(q[6]))*(q[6]/dt))+((q[i]/dt)*(a*cos(q[
9]))*(q[9]/dt));

b[4]+=((q[i]/dt)*(a*sin(q[6]))*(q[6]/dt))+((q[i]/dt)*(a*sin(q[
9]))*(q[9]/dt));
 }
 ofstream outpute;
 outpute.open("c:/THESIS/3Link/b.txt", ios::out);
 for(int x=1; x<5; x++)
 {
  outpute << b[x] << endl;
 }
}

void findForce(double f[9], double q[9])
{
 f[1]=-
1*(q[1]/dt)*(dragpar*cos(q[3])*cos(q[3])+dragperp*sin(q[3])*si
n(q[3]))+(q[2]/dt)*((dragperp-dragpar)*sin(q[3])*cos(q[3]));
 f[2]=(q[1]/dt)*((dragperp-dragpar)*sin(q[3])*cos(q[3]))-
(q[2]/dt)*(dragpar*sin(q[3])*sin(q[3])+dragperp*cos(q[3])*cos(
q[3]));
 f[3]=-1*dragrot*(q[3]/dt);
 f[4]=-
1*(q[4]/dt)*(dragpar*cos(q[6])*cos(q[6])+dragperp*sin(q[6])*si
n(q[6]))+(q[5]/dt)*((dragperp-dragpar)*sin(q[6])*cos(q[6]));
 f[5]=(q[4]/dt)*((dragperp-dragpar)*sin(q[6])*cos(q[6]))-
(q[5]/dt)*(dragpar*sin(q[6])*sin(q[6])+dragperp*cos(q[6])*cos(
q[6]));
 f[6]=-1*dragrot*(q[6]/dt);
 f[7]=-
1*(q[7]/dt)*(dragpar*cos(q[9])*cos(q[9])+dragperp*sin(q[9])*si
n(q[9]))+(q[8]/dt)*((dragperp-dragpar)*sin(q[9])*cos(q[9]));
 f[8]=(q[7]/dt)*((dragperp-dragpar)*sin(q[9])*cos(q[9]))-
(q[8]/dt)*(dragpar*sin(q[6])*sin(q[9])+dragperp*cos(q[9])*cos(
q[9]));
 f[9]=-1*dragrot*(q[9]/dt);

 ofstream outputf;
 outputf.open("c:/THESIS/3Link/f.txt", ios::out);
 for(int x=1; x<10; x++)
 {
  outputf << f[x] << endl;
 }
}
```

A5. MATLAB Code Used for Matrix Manipulation and Solving

```
%gauss.m - Perform matrix functions to be read into C++ code

%define m
m=0.5;
I=0.7;
M=[m 0 0 0 0 0 0 0 0;
     0 m 0 0 0 0 0 0 0;
     0 0 I 0 0 0 0 0 0;
     0 0 0 m 0 0 0 0 0;
     0 0 0 0 m 0 0 0 0;
     0 0 0 0 0 I 0 0 0;
     0 0 0 0 0 0 m 0 0;
     0 0 0 0 0 0 0 m 0;
     0 0 0 0 0 0 0 0 I];

%define q
n=4;
fid=fopen('c:\THESIS\3Link\q.txt','r');
q=fscanf(fid,'%f', n);

%define J matrix
fid=fopen('c:\THESIS\3Link\J1.txt','r');
J1=fscanf(fid,'%f', n);
fid=fopen('c:\THESIS\3Link\J2.txt','r');
```

```
J2=fscanf(fid,'%f', n);
fid=fopen('c:\THESIS\3Link\J3.txt','r');
J3=fscanf(fid,'%f', n);
fid=fopen('c:\THESIS\3Link\J4.txt','r');
J4=fscanf(fid,'%f', n);
fid=fopen('c:\THESIS\3Link\J5.txt','r');
J5=fscanf(fid,'%f', n);
fid=fopen('c:\THESIS\3Link\J6.txt','r');
J6=fscanf(fid,'%f', n);
fid=fopen('c:\THESIS\3Link\J7.txt','r');
J7=fscanf(fid,'%f', n);
fid=fopen('c:\THESIS\3Link\J8.txt','r');
J8=fscanf(fid,'%f', n);
fid=fopen('c:\THESIS\3Link\J9.txt','r');
J9=fscanf(fid,'%f', n);

J=[J1'; J2'; J3'; J4'; J5'; J6'; J7'; J8'; J9'];

fid=fopen('c:\THESIS\3Link\b.txt','r');
b=fscanf(fid,'%f', 4);

%values for the force
fid=fopen('c:\THESIS\3Link\f.txt','r');
f=fscanf(fid,'%f', 9);

lambda=b'*inv(J'*inv(M)*J)

temp=-1*inv(M)*J*lambda'

%write the data to a file

fid=fopen('c:\THESIS\3Link\data.txt','w')
fprintf(fid, '%6.4f\n',temp);
fclose(fid)
```