TACTIC BEHAVIORS IN BACTERIAL DYNAMICS

by

MICHAEL DAVID SEKORA

Submitted to the Department of Physics
in Partial Fulfillment of the Requirements for the Degree of

BACHELOR OF SCIENCE

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2005 [June 2005]

Signature of Author _____

Michael David Sekora
Department of Physics
(6 May 2005)

Certified by _____

Dr. Roman Stocker
Thesis Supervisor

Department of Mathematics
(If supervisor is not in Physics, list correct department)

Accepted by _____

Professor David E. Pritchard
Senior Thesis Coordinate, Department of Physics

Tactic Behaviors in Bacterial Dynamics

by

Michael David Sekora

Submitted to the Department of Physics on 6 May 2005
in Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science

Abstract

The locomotion of a wide class of motile bacteria can be mathematically described as a biased random walk in three-dimensional space. Fluid mechanics and probability theory are invoked to model the dynamics of bacteria swimming using tactic behaviors (movements or reorientation in response to chemical, physical or environmental stimuli) in flowing, viscous media. Physical descriptions are developed for bacterial chemotaxis (response to chemical agents) near particles exuding attractants, a small-scale process with global-scale implications for the biogeochemistry of the oceans. Three cases were investigated: a stationary particle, a slowly moving particle and a particle that generates a hydrodynamic wake in the form of attached vortices. The key finding of this thesis consists in the discovery of several scenarios in which motile bacteria swimming via random walks put themselves at a disadvantage in their quest for food with respect to non-motile pacteria. Thus, there exist threshold values in nutrient gradients and bacterial chemosensory ability below which bacteria would be better served if they did not swim. In the presence of vortices, it was discovered that bacteria can exploit the recirculating flow field to vastly increase their nutrient supply, but only if they alter their swimming behavior as a function of the concentration field. Otherwise, slow bacteria completely miss the hydrodynamic wake (and the high nutrient region) behind a nearby moving particle, while fast bacteria end up colonizing the particle (i.e. clustering around the particle and potentially anchoring themselves to it). These processes are currently under investigation in laboratory experiments using high-speed digital photography, for which software (BacTrack^{TM}) was written that can locate and track multiple bacteria over time, with the aim of providing trajectories and their statistics and ultimately establish the importance of these phenomena for marine ecology and biogeochemistry. Preliminary experiments were conducted with *Escherichia coli* being exposed to ultraviolet radiation, documenting the known result of *E. coli* being repelled by UV radiation and providing a successful test bed for the reliability of the tracking software.

Thesis Supervisor: Roman Stocker
Title: Applied Mathematics Instructor

For my Mom and Dad

Table of Contents

1. Introduction

1.1. Tactic Behaviors

Tactic responses occur in organisms and are orientations or movements that result from forces or agents acting on those organisms. Tactic responses occur in bacteria when the intensity of a spatially uniform stimulus changes with time. Examples of tactic behaviors are phototaxis (response to electromagnetic radiation), chemotaxis (response to chemical agents), thermotaxis (response to temperature), and magnetotaxis (response to magnetic fields). Such behaviors define bacterial motion, which can be described as a biased three-dimensional random walk.

The first studies of tactic responses were carried out by Engelmann in 1883 and Pfeffer in 1884. Engelmann first examined phototaxis and found that when *Bacterium photometricum* (species of bacteria responsive to stimulation by visible radiation) were uniformly illuminated and then briefly darkened, every bacterium backed up, stopped, and then resumed normal motion. An identical effect was found for chemotaxis when Engelmann briefly exposed *Bacterium photometricum* to CO_2 [2, 3]. Pfeffer investigated chemotaxis by inserting a capillary tube containing some chemical attractant into a suspension of motile bacteria. Pfeffer observed that the bacteria accumulated near the mouth of the capillary tube, where the concentration was relatively high [4]. From these early experiments, one understands that chemotaxis results from the abundance or scarcity of certain chemical agents, where certain microorganisms use chemosensory abilities to find and stay in microenvironments that are rich in chemical attractants or scarce in chemical repellants. These chemosensory abilities are also used by some dinoflagellates and bacteria to sense the chemical field generated by phytoplankton that leak organic matter [5, 6].

Berg and Brown (1972) observed that microorganisms move toward regions of higher attractant (or lower repellant) concentrations by sensing the gradient of the chemical field [4]. However, theoretical and experimental analysis has shown that bacteria are unable to sense the gradient of the chemical field over their body length because of their small size (length ~ 0.5–50 μm) [5, 7]. Nevertheless, bacteria are able to detect and respond to chemical gradients by utilizing a temporal memory that is at most a few seconds [2]. Bacteria use this temporal memory to compare the chemical concentration for which they presently experience with the chemical concentration for which they experienced only a moment ago.

Bacteria use their temporal memory to sense changes in chemical concentration by moving in two modes: run mode and tumble mode [5]. In run mode, bacteria move in approximately a straight line at a nearly uniform speed. In tumble mode, bacteria stop moving and abruptly orient themselves in a new direction [2, 4, 5]. For every tumble that is taken, there must be a run. While in run mode, bacteria take small steps in the concentration field. If movement with respect to the concentration gradient is favorable, bacteria remain in run mode and continue taking small steps, while maintaining their

orientation. However, if movement with respect to the concentration gradient is unfavorable, bacteria enter tumble mode and reorient themselves.

In many bacteria, movement and reorientation with respect to the concentration field is accomplished through the action of flagella. Flagella are approximately 15–50 μm in length and are threadlike projections extending outward from bacterial cells, which have an ovular (i.e., basal body) shape. Certainly, not all microorganisms have flagella. However, many studies of bacterial chemotaxis including studies by Berg and Brown and this work have used flagellated bacteria, particularly *Escherichia coli*. There are three main types of flagella distributions in bacteria: monotrichous (single flagella at one end of bacterial cell, e.g., *Escherichia coli*), lophotrichous (tuft of flagella at one or both ends of bacterial cell), and peritrichous (many flagella distributed uniformly over bacterial surface). Figure 1.1 shows images of monotrichous, lophotrichous, and peritrichous flagella distributions (taken from Reference 8).

Figure 1.1: Images of monotrichous, lophotrichous, and peritrichous flagella distributions (taken from Reference 8)

The cellular mechanics of flagellar motion are complicated. But essentially, when flagella rotate counter-clockwise (forming a corkscrew pattern), bacteria enter run mode and move forward in a manner similar to how submarines move forward by use of propellers. When flagella rotate clockwise, bacteria stop swimming and enter tumble mode. Here, the flagella spring outward from the bacteria, causing the bacteria to reorient themselves. Figure 1.2 diagrams run and tumble modes (taken from Reference 8).

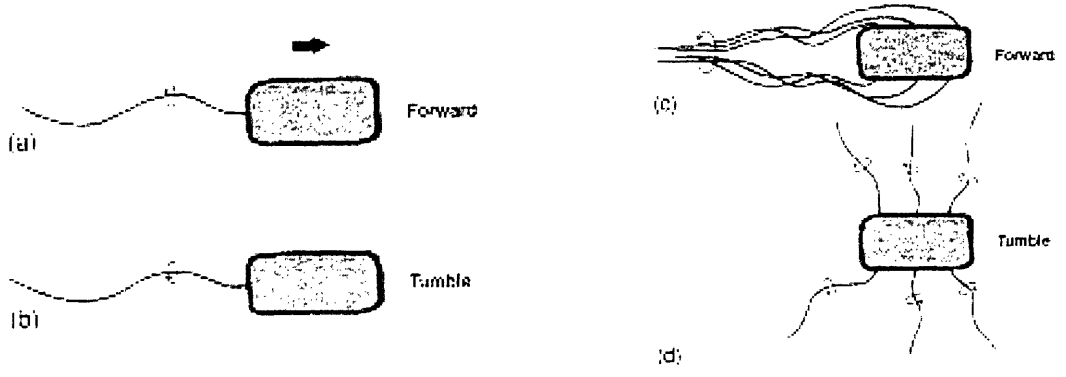

Figure 1.2: Run and tumble modes for monotrichous and peritrichous flagella distributions (taken from Reference 8)

Since movement is with respect to the chemical gradient, the length of a run (i.e., the straight line distance) is a random variable whose average increases when concentrations increase and decreases when concentrations decrease. Likewise, the change in direction (i.e., orientation) that a bacterium undergoes during a tumble is a random variable with a certain probability distribution. Therefore, the combination of run and tumble modes result in a three-dimensional random walk that is biased toward regions of higher chemical attractant (or lower chemical repellant) concentrations [5].

When bacteria move up the attractant gradient (or down the repellant gradient), the probability per unit time of the termination of a run decreases and the bacteria change direction less frequently. This motion results from an increase in the time rate of change of the fractional amount of chemoreceptor (protein) bound [2, 4]. When bacteria move down the attractant gradient (or up the repellant gradient), the probability per unit time of the termination of a run reverts to the value which is appropriate for an isotropic solution of similar concentration and the bacteria changes direction more frequently. This motion results from a decrease in the time rate of change of the fraction amount of chemoreceptor (protein) bound [2, 4].

The process by which microorganisms navigate through chemical fields can be summarized in the following crude outline:

1. Sample attractant (or repellant) concentration at current position
2. Take a small step within the concentration field and sample the attractant (or repellant) concentration at new position
3. Compare the attractant (or repellant) concentrations prior to and after taking the small step in the concentration field
4. Choose dynamical mode
    - 4a. If the attractant increases (or repellant decreases) between steps: continue moving in the present direction (run mode)
    - 4b. If the attractant decreases (or repellant increases) between steps: change direction (tumble mode)
5. Repeat this process starting from Step 2

Despite the complexity of molecular interaction networks, reaction pathways, and cellular processes involved in chemotaxis, the above conceptual model captures the essential physics. Some of this biological complexity can be accounted for by the strengths of certain chemosensory parameters appearing in the model. For completeness, the molecular pathways resulting in bacterial chemotaxis and flagellar assembly are given without explanation in Figures 1.3 and 1.4.

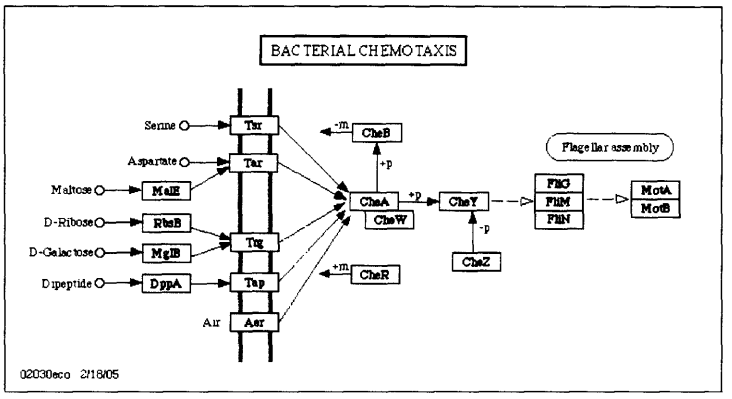

Figure 1.3: Molecular pathways associated with chemotaxis in *E. coli* (taken from Reference 9).

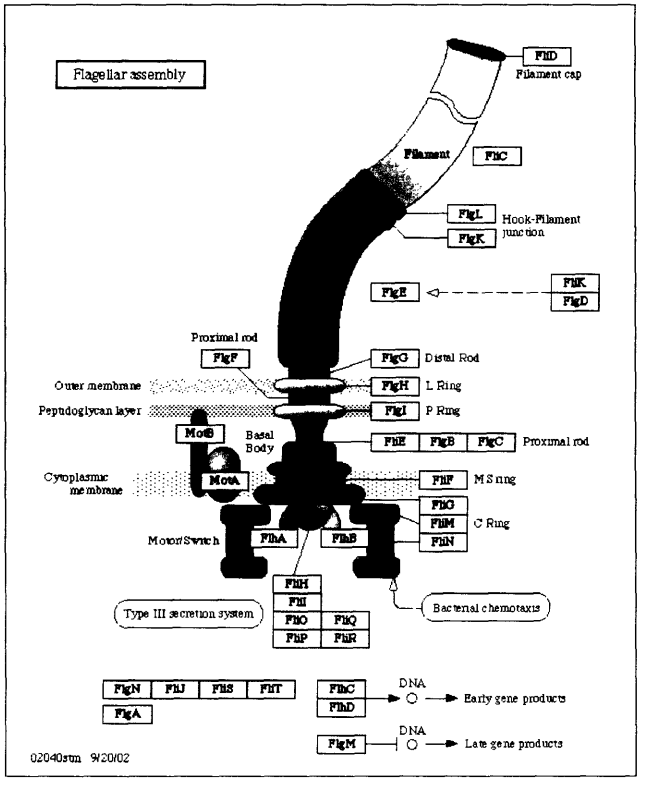

Figure 1.4: Molecular pathways associated with flagellar assembly in *E. coli* (taken from Reference 9).

1.2. Nutrient Sources and Ecological Consequences

Microscale processes significantly influence energy and nutrient fluxes in marine environments, where bacteria and their predators have a critical role in maintaining ecological productivity and explaining how life formed at greater aquatic depths. Driving this system is the growth of bacteria which are dependent upon dissolved organic matter, which originates from point sources such as stationary or moving leaky algae, bacterial

hydrolysis products diffusing from aggregates, or fecal pellets. The bacterial biomass is consumed by protozoan predators, which then enter the classic food web. Therefore, ecological productivity is directly proportional to the concentration of dissolved organic matter. It is known that bacteria near point sources outgrow bacteria utilizing ambient nutrient concentrations. Thus, microscale heterogeneity in the concentration of dissolved organic matter enhances bacterial growth and consequently the growth of protozoan predators. Overall, these processes lead to significantly higher rates of ecosystem productivity that can be explained from bulk nutrient concentration. It is crucial to estimate the extent that microscale interations structure microbial communities on ecological and evolutionary timescales.

1.3. Random Walks

Random walks and stochastic processes are probabilistic in nature. These dynamics are active areas of research in many fields and are significantly impacting science's understanding of how macroscopic behavior results from microscopic processes.

The following simple example is presented to motivate this work's treatment and application of random walks. Consider the motion of particles in one dimension, which start at time $t = 0$ and $x = 0$. The random walk is defined by the following rules:

1. Each particle steps to the right or left once every τ seconds (i.e., the time step), moving a distance δ (i.e., the step size).
2. The probability for a particle going left or right is 1/2.
3. Each particle moves independently of all other particles.

If there are N_0 particles and $x_i(n)$ is the position of the i^{th} particle after n steps, then the mean displacement after n steps is:

$$\langle x_i(n)\rangle = \frac{1}{N_0}\sum_{i=1}^{N_0} x_i(n) = \frac{1}{N_0}\sum_{i=1}^{N_0} x_i(n-1) \pm \delta = \frac{1}{N_0}\sum_{i=1}^{N_0} x_i(n-1) = \langle x_i(n-1)\rangle = 0. \quad 1.1$$

Now consider a distribution of random walkers. The number of particles at a given position and time is $N(x, t) = N_0P(x, t)$, where $P(x, t)$ is the probability that a particle will be at position x at time t. After a discrete change is position and time, $P(x, t)$ is given by:

$$P(x,t+\tau) = \frac{1}{2}\left(P(x+\delta,t) + P(x-\delta,t)\right). \quad 1.2$$

Performing a Taylor expansion about x and t in the limit as $\delta, \tau \to 0$ shows that:

$$P(x,t) + \frac{\partial P}{\partial t}\tau \approx \frac{1}{2}\left(P(x,t) + \frac{\partial P}{\partial t}\delta + \frac{\partial^2 P}{\partial t^2}\frac{\delta^2}{2} + P(x,t) - \frac{\partial P}{\partial t}\delta + \frac{\partial^2 P}{\partial t^2}\frac{\delta^2}{2}\right). \quad 1.3$$

If one disregards higher order terms and works in the continuum limit, the discrete probability $P(x, t)$ becomes the probability density $p(x, t)$ and Equation 1.3 becomes:

$$\frac{\partial p}{\partial t} \approx \frac{\delta^2}{2\tau} \frac{\partial^2 p}{\partial t^2}, \tag{1.4}$$

which is the diffusion equation for the probability distribution [1]. This example shows how macroscopic behavior results from microscopic processes. Modifications can be made to include a bias, boundary constraints, periodicity, etc.

Random walks are very useful for understanding biophysical mechanisms and have been applied to the study of bacterial motion, where each bacterium is treated as an independent random walker. Bacterial motion is an attractive area of research because it integrates principles from:

1. Biology – understanding processes associated with a single cell and how these processes evolved for use in more complex organisms
2. Physics – elegantly modeling biologically motivated problems
3. Mathematics – exploring complicated systems and associations between dynamics and decision theory
4. Ecology – incorporating microbial dynamics into marine food web structure and climate change models

1.4. Overview

This work is meant to illustrate some of the physical and biological effects associated with tactic behaviors, in particular, chemotaxis. The stationary particle problem is discussed in Chapter 2, where bacteria seek out higher nutrient concentrations emanating from a spherical particle that is fixed in time and space. The moving particle problem, which is presented in Chapter 3, is similar to the stationary particle problem, except that the particle descends vertically through viscous fluid. In Chapter 4, the moving particle problem is generalized such that vortex formation is considered. Solutions are found using computational fluid dynamics and the finite element method. Preliminary results are given in Chapter 5 for experiments with *Escherichia coli* being exposed to ultraviolet radiation. These results show that *E. coli* is repelled by UV and that the image processing software (BacTrack™) that was developed is effective at locating and tracking numerous bacteria over many exposures. Further research and conclusions about this work are explained in Chapter 6. Samples of chemotaxis simulation code as well as image processing software are given in the appendix.

2. Stationary Particle Problem

Bacterial chemotaxis toward a stationary, attracting particle is discussed in this chapter. Although this scenario represents the simplest case for bacterial chemotaxis, it has profound micro-ecological consequences, where the stationary particle can be thought of as a spherical alga cell floating in an aqueous environment and leaking nutrients. The model and results presented in this chapter follow work done by Brown and Berg (1974) and Jackson (1987).

2.1. Model

Figure 2.1 illustrates the stationary particle problem. A spherical particle, perhaps an alga cell, is centered at the origin of a Cartesian coordinate system. The alga has a radius a and nutrient concentration C_0 at its surface. At some distance r_0 from the origin there is a given number of bacteria with swimming speed v_0 that prefer the higher nutrient concentrations found near the alga. After some time t, the bacteria are located at a distance $r(t)$ from the origin. This distance is contained in a sphere of radius $v_0 t$ centered on the origin.

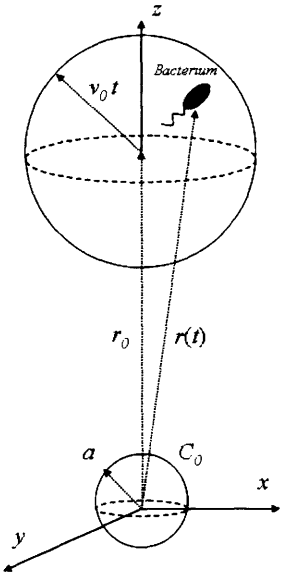

Figure 2.1: Stationary particle problem. A spherical alga is centered at the origin and has radius a and nutrient concentration C_0 at its surface. Initially, a given number of bacteria are located at $(0,0,r_0)$.

The alga cell leaks an organic chemical, which forms a spherically symmetric concentration field that is determined by molecular diffusion. The effects of rotational diffusion are not considered in the model even though rotational diffusion could slightly alter the direction that bacteria swim [6]. Diffusion from a spherical particle in the absence of fluid motion or diffuse sources or sinks is given by:

$$\frac{\partial C}{\partial t} = D\nabla^2 C, \tag{2.1}$$

where C is concentration, t is time, and D is diffusivity. For a sphere of radius a with concentration C_∞ at infinity and concentration C_0 at its surface, the concentration at a distance r from the sphere's center is given by:

$$C = (C_0 - C_\infty)ar^{-1} + C_\infty. \tag{2.2}$$

After considering the loss of molecules from the cell per unit time, which is given by:

$$\delta = 4\pi Da(C_0 - C_\infty), \tag{2.3}$$

Equation 2.2 simplifies to:

$$C = \left(\frac{ba^{2.28}\mu f}{4\pi Dn}\right)r^{-1} + C_\infty = \left(\frac{\varphi L}{4\pi D}\right)r^{-1} + C_\infty. \tag{2.4}$$

$ba^{2.28} = \varphi$ (the carbon content for planktonic microalgae), L is the specific molecular leakage rate of substrate, μ is the growth rate of an alga cell, f is the fraction of primary production for the organic matter leakage rate from phytoplankton cells, and n is the average number of carbon atoms per compound that the alga leaks (n does not necessarily have to be associated with carbon atoms, but rather any desired atom or molecule). For a bacterium in the run phase, there is a probability P_t that a run will end within the time interval Δt. Therefore:

$$P_t = \Delta t / \tau, \tag{2.5}$$

where τ is the mean run time and is given by:

$$\tau = \exp\left[\ln \tau_0 + \alpha \overline{\frac{dP_b}{dt}}\right]. \tag{2.6}$$

τ_0 is the mean run length in the absence of concentration gradients, α is a chemosensory constant of the system (i.e., a measure of how strongly attracted bacteria are to concentration gradients), and $\overline{(dP_b/dt)}$ is a weighted rate of change of P_b, where P_b is the fractional amount of cellular protein surface receptor bound by the substrate. $\overline{(dP_b/dt)}$ is given by:

$$\overline{\frac{dP_b}{dt}} = T_m^{-1}\int_{-\infty}^{t}\frac{dP_b}{dt'}\exp\left(\frac{t'-t}{T_m}\right)dt', \tag{2.7}$$

where T_m is a time constant of the bacterial system (i.e., decay time associated with temporal memory mechanism) and (dP_b/dt) is given by:

$$\frac{dP_b}{dt} = \frac{K_D}{(K_D + C)^2}\frac{dC}{dt}. \qquad 2.8$$

K_D is the half-saturation (i.e., dissociation) constant of the surface receptor binding to the compound of interest and (dC/dt) is the time rate of change in the concentration as a bacterium moves through the chemical field, which is approximated by:

$$\frac{dC}{dt} = \frac{C_{final} - C_{initial}}{\Delta t}, \qquad 2.9$$

$C_{initial}$ is the concentration felt by a bacterium before taking a step within time Δt and C_{final} is the concentration felt by a bacterium after taking a step within time Δt.

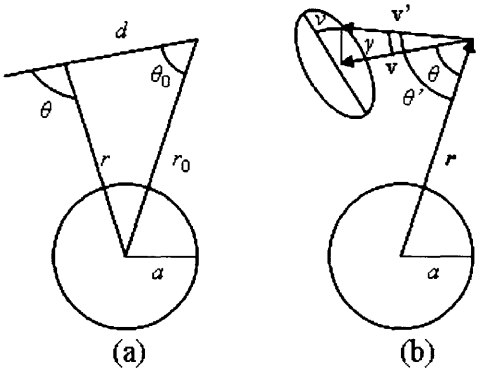

Figure 2.2: Alga-centered coordinate system. A: Change in θ and r relative to the alga during a run of length d. B: Angles used to define a tumble. (Reproduction of Figure 1 used in Reference 5).

Berg and Brown (1972) determined experimentally that the tumble angle γ of *Escherichia coli* is slightly biased. To account for this angular bias, Jackson (1987) adapted a unique, alga-centered coordinate system that used the spherical symmetry of the fixed alga problem. The position and motion of a bacterium is given by the distance from the center of the alga r and the angle of movement relative to the alga. If a bacterium moves with a constant speed v_0 on a straight run starting at r_0 with an angle θ_0 away from the alga center, then at a time t into the run, the bacterium will be at a distance $d = v_0 t$ from its starting position. Also, the bacterium will be at a distance r away from the alga center and heading at an angle θ from the alga. These values are given by:

$$\begin{aligned} r &= \left(d^2 + r_0^2 - 2r_0 d\cos\theta_0\right)^{1/2} \\ \cos\theta &= r^{-1}\left(r_0\cos\theta_0 - d\right) \end{aligned}. \qquad 2.10$$

During a tumble, a bacterium stops and changes its direction. The angle of the bacterium's new direction relative to the bacterium's old direction is the tumble angle γ, which is a random variable. However, a complete description of bacterial motion requires a second angle v which describes the angle around the old direction vector relative to the algal cell. After some trigonometric calculations, the direction of a new run relative to the alga is given by:

$$\theta' = \cos^{-1}(\cos\theta\cos\gamma - \sin\theta\sin\gamma\cos v). \tag{2.11}$$

Since any value of v is equally likely, v is calculated from a uniform distribution ranging from 0 to 2π. From symmetry arguments, if γ has no directional bias, then θ' is independent of θ and is given by the following equation, where R is a random number with a uniform distribution from 0 to 1:

$$\theta' = \cos^{-1}(2R - 1), \tag{2.12}$$

With the above model, one is able to simulate bacterial chemotaxis by starting a bacterium at a given value of r with $\overline{(dP_b/dt)} = 0$ and randomly choosing a value of θ_0. The simulation followed bacterium through time and calculated values of r, C, and P_t at intervals of Δt. When $P_t > R$, a tumble was initiated. During a tumble, the simulation calculated new values of r and θ using random numbers. Tumbles were assumed to be instantaneous [5].

For the simulations presented in Section 2.2.1, bacteria were confined to a region $a < r < R_{max}$. If $r < a$ or $r > R_{max}$, then that bacterium was reflected and a tumble was initiated [5]. Reflections initiated at R_{max} were implemented by Jackson to isolate the effect of a single alga from a larger volume by assuming that there are other alga cells spaced a distance of $2R_{max}$. Some issue arise with this assumption. First, if there was a distribution of alga cells, then the concentration field would be greatly altered, particularly the background substrate concentration. Second, there is no geometric configuration that allows each alga cell to have a sphere of influence of radius R_{max} (i.e., one cannot pack spherical volumes of equal radius into a three-dimensional space without leaving gaps). Therefore, the assumption that two adjacent alga cells have an equal effect on a bacterium position a distance R_{max} from the center of one of the alga cells is invalid. Moreover, this boundary reflection at R_{max} has a significant impact on the bacterial random walks and in Section 2.2.2, the constraint of initiating reflections at R_{max} is lifted. Simulations were carried out by software written in MATLAB (MathWorks). A sample of the simulation code is presented in Section A.1.

Table 2.1 summarizes the parameters used in modeling the stationary particle problem. These values were derived from studies conducted by Berg and Brown (1972, 1974) and are the same parameters used in Jackson's simulations (1987). The same parameter values are used throughout this section unless otherwise specified.

parameter	meaning	standard value
a	Alga radius	10×10^{-4} cm
R_{max}	Maximum distance from center of alga	$10a$
r_0	Initial distance from center of alga	75×10^{-4} cm
D	Substrate diffusivity	10^{-5} cm^2 s^{-1}
μ	Growth rate of alga	0.5 d^{-1}
f	Fraction of primary production for leakage	1
n	Number of desired atoms per compound	1
L	Specific molecular leakage rate, $(\mu f / n)$	0.5 d^{-1}
b	Curve fitting constant for alga	1.67×10^{-4} mol cm$^{-2.28}$
C_∞	Background substrate concentration	0 mol cm^{-3}
C_0	Substrate concentration at $r = a$	1.11×10^{-9} mol cm^{-3}
K_D	Chemosensory binding constant	1×10^{-7} mol cm^{-3}
v_0	Bacterial swimming speed	12.5×10^{-4} cm s^{-1}
τ_0	Unbiased mean run time	0.67 s
α	Chemosensory constant of bacterial system	660 s
T_m	Time constant of bacterial system	1 s
Δt	Numerical time step	0.067 s

Table 2.1: Parameter values for the stationary particle problem (Values taken from Reference 5)

To determine the angular bias in γ, Berg and Brown (1972) recorded 1,166 tumbling events for *E. coli* and constructed a distribution for the number of events versus γ. In Berg and Brown's analysis, the mean value for γ was 62 ± 26 degrees [4]. To determine the distribution of γ for simulations, Berg and Brown's original data was fit with four distributions (Rayleigh, Maxwellian, normal, and log-normal). Figure 2.3 shows these fits and Table 2.2 compares the χ^2 (i.e., the goodness of fit) for each fit.

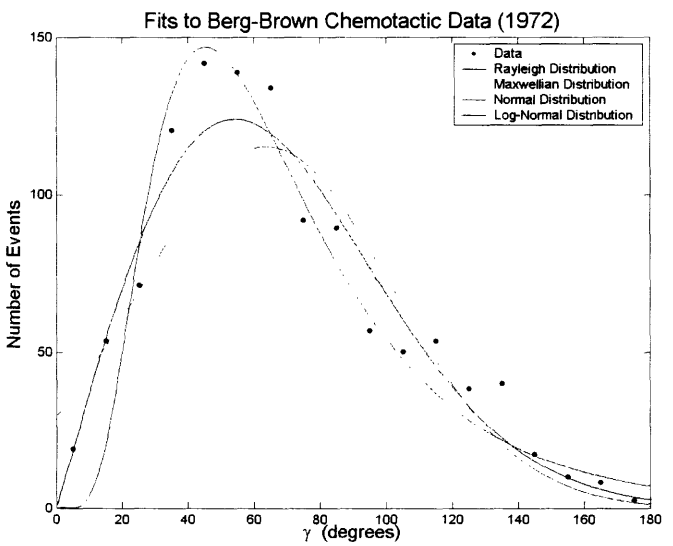

Figure 2.3: Fits to Berg and Brown's original data on γ [5].

Distribution	χ^2
Rayleigh	2.11
Maxwellian	7.43
Normal	5.31
Log-Normal	5.05

Table 2.2: Distributions and their relative χ^2 used in fitting Berg and Brown's original γ data [4].

The Rayleigh distribution provided the best fit to the data and is given by:

$$P(x) = \frac{x}{\beta^2} \exp\left(\frac{-x^2}{2\beta^2}\right), \tag{2.13}$$

where β was found to equal 54.2 degrees. The mean angle is:

$$\mu = \beta\sqrt{\frac{\pi}{2}}, \tag{2.14}$$

and the variance is:

$$\sigma^2 = \frac{4-\pi}{2}\beta^2. \tag{2.15}$$

Therefore, the mean angle is 67.9 ± 35.5 degrees. This Rayleigh distribution is defined as the Berg-Brown distribution of γ.

2.2. Results

2.2.1. Background

Figure 2.4 shows a typical chemotaxis trajectory for a single bacterium starting a distance $r_0 = 75\ \mu m$ from the center of an alga cell and traveling for 50 seconds (i.e., 747 total time steps). The downward, jagged trajectory is characteristic of a biased random walk. When initially positioned farther from the center of the alga, bacteria take a longer time to descend to the alga's surface because the attractant concentration is weaker and there is a greater distance to traverse.

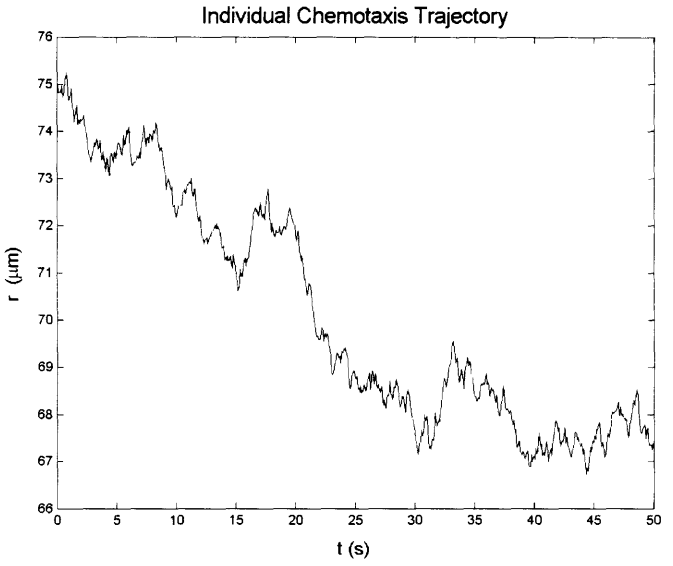

Figure 2.4: Individual chemotaxis trajectory characterizing a biased random walk. The bacterium started a distance $r_0 = 75\ \mu m$ from the center of the alga and was simulated using standard parameter values.

When the positions of many bacterial trajectories are averaged over time:

$$\left\langle \vec{r}(t) \right\rangle = \frac{1}{N} \sum_{i=1}^{N} \vec{r}(t), \tag{2.16}$$

steady movement toward the alga is observed. How quickly a bacterium approaches the alga significantly depends upon the magnitude of L. Figure 2.5 shows the average distance from the alga $\langle \vec{r} \rangle$ as a function of time for three cases (biased γ, random γ, and fixed $\gamma = 80^\circ$) at different values of L. In the biased case, γ was sampled from the Berg-Brown distribution. In the random case, γ was sampled from a uniform distribution defined from 0 to π. And in the fixed case, γ was held constant at 80°. Overall, there is no qualitative difference between the average trajectories for the different cases. The small hump in the trajectories seen between $t = 0$ and $t = 10$ seconds occurs for two reasons. The principal reason is some individual bacteria trajectories traveled away from the alga and were reflected back toward the alga. This reason is plausible since it would take 4 seconds for a bacterium to move from r_0 to R_{max} and back to r_0 given the bacterium travels at v_0 along a straight line without tumbling. Another reason for the small hump is it takes a small amount of time (approximately T_m) for chemotaxis to work at full capability. This initial down time results from the temporal memory mechanism involved in chemotaxis, where a bacterium must maintain a brief history of moving towards favorable concentration gradients. The roundedness in the average trajectories near $\langle r \rangle = 20 \times 10^{-4}$ cm results from individual bacteria trajectories reflecting off of the surface of the alga. Lastly, between $t = 10$ and $t = 30$ seconds $\langle \vec{r} \rangle$ is approximately linear. Therefore, the rate of movement inward (i.e., approach velocity v_a) is approximately constant – making v_a a useful indicator of chemotactic response.

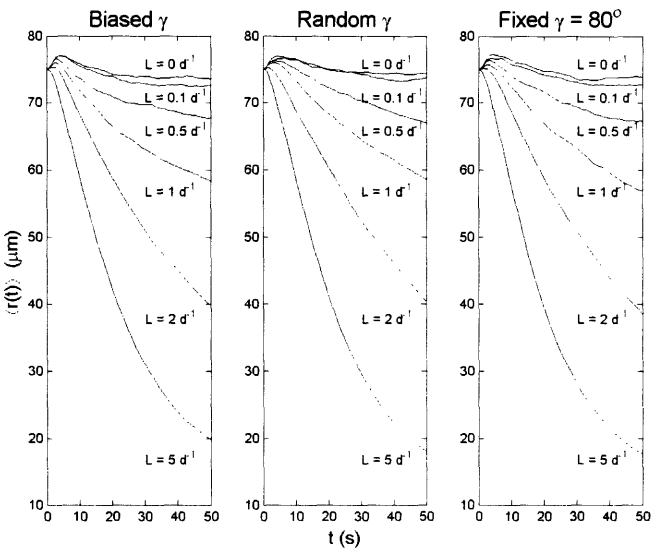

Figure 2.5: Average distance from the alga $\langle \overset{\omega}{r} \rangle$ as a function of time for biased γ, random γ, and fixed γ = 80° at different values of L. Bacteria are steadily attracted to the alga. $\langle \overset{\omega}{r} \rangle$ was obtained by averaging 5,000 bacteria trajectories over the 50 seconds that the bacteria traveled.

v_a associated with $\langle \overset{\omega}{r}(t) \rangle$ was calculated using linear recursion for $\tilde{t} = \{t = 10, \mathrm{K}\ , 30\}$:

$$v_a = \frac{1}{\dim\left(\overset{\omega}{r}(\tilde{t})\right)} \sum_{i=1}^{\dim\left(\overset{\omega}{r}(\tilde{t})\right)} \frac{\left| \left\langle \overset{\omega}{r}(\tilde{t}_i) \right\rangle - \left\langle \overset{\omega}{r}(\tilde{t}_{i+1}) \right\rangle \right|}{\Delta t}. \qquad 2.17$$

The uncertainty in v_a was obtained by calculating the standard error:

$$\sigma_{v_a} = \left[\frac{1}{\dim\left(\overset{\omega}{r}(\tilde{t})\right)\left(\dim\left(\overset{\omega}{r}(\tilde{t})\right) - 1\right)} \sum_{i=1}^{\dim\left(\overset{\omega}{r}(\tilde{t})\right)} \left(\frac{\left| \left\langle \overset{\omega}{r}(\tilde{t}_i) \right\rangle - \left\langle \overset{\omega}{r}(\tilde{t}_{i+1}) \right\rangle \right|}{\Delta t} - v_a \right)^2 \right]^{1/2}. \qquad 2.18$$

Calculating v_a for biased and random γ at different values of a allows one to compare the overall chemotactic response between these two distributions of γ. Figure 2.6 illustrates that there is little difference in v_a at different values of a for either a biased or random distribution of γ. Moreover, Figure 2.6 demonstrates an approximately linear relation between v_a and a. Because there is little difference in v_a for a biased or random distribution of γ and because simulations involving a random distribution of γ were computationally easier to perform; subsequent simulations were calculated using a random distribution of γ unless otherwise specified. Also, by comparing v_a for fixed values of γ, one determines if there is a favorable fixed tumbling angle. Figure 2.7 demonstrates that fixed (non-zero) values of γ produce approximately the same approach velocity ($v_a = 0.123 \pm 0.018\ \mu\mathrm{m\ s^{-1}}$, $\chi^2 = 0.965$).

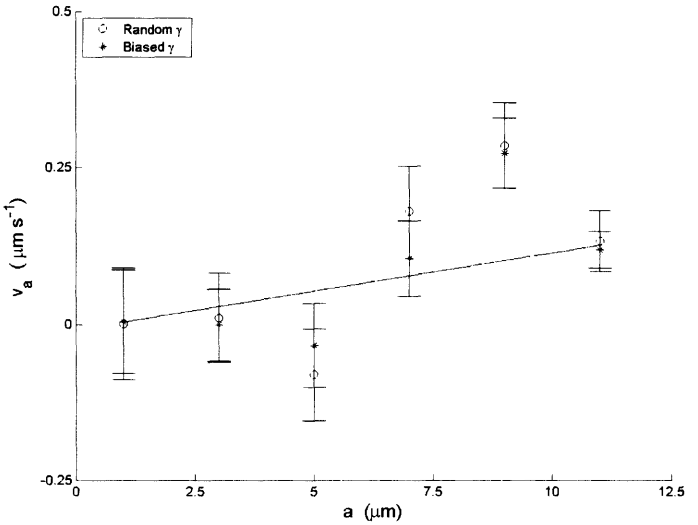

Figure 2.6: v_a as a function of a for biased and random distributions of γ. There is little difference in v_a for either distribution. 5,000 bacteria were simulated for 50 seconds at each value of a for both distributions.

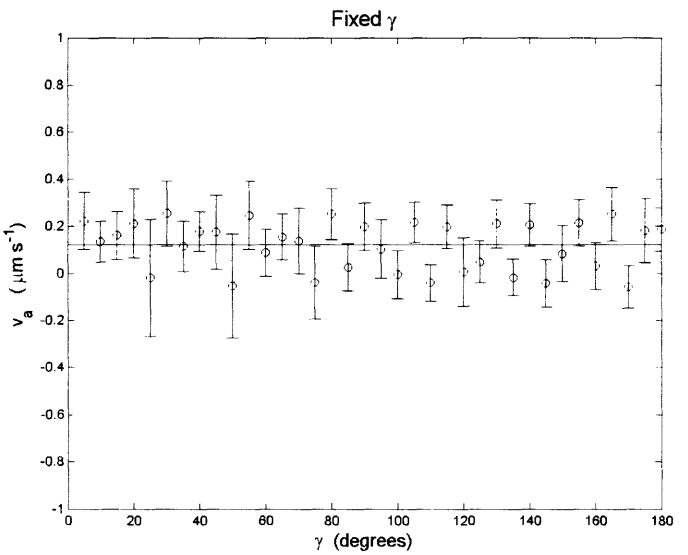

Figure 2.7: v_a as a function of fixed γ. Fixed values of γ produce approximately the same approach velocity ($v_a = 0.123 \pm 0.018$ μm s^{-1}, $\chi^2 = 0.965$). 5,000 bacteria were simulated for 50 seconds at each value of γ.

As one increases the background concentration C_∞, bacteria approach the alga cell more slowly because the increased background concentration decreases the concentration gradient, which decreases chemosensory response. Figure 2.8 shows the average distance from the alga $\langle \overset{\omega}{r} \rangle$ as a function of time for different background concentrations. Increasing background concentration has a dramatic effect because it decreases chemosensory response.

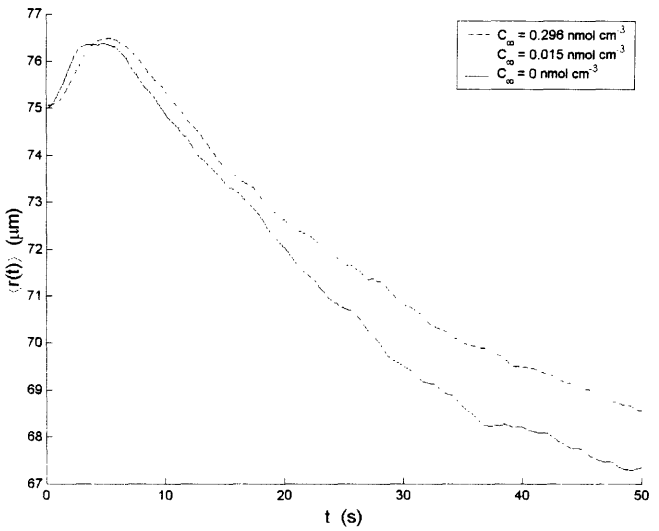

Figure 2.8: $\langle \overset{\omega}{r} \rangle$ as a function of t for different values of C_∞. Increased background concentration decreases the concentration gradient, which decreases chemosensory response. $\langle \overset{\omega}{r} \rangle$ was obtained by averaging 5,000 bacteria trajectories over the 50 seconds that the bacteria traveled.

Figure 2.9 shows the relationship between parameters α and T_m. Increased values of T_m increase the time over which bacteria average concentration gradients and decrease bacterial sensitivity to concentration gradients. Increased values of α increase bacterial sensitivity to a given stimulus and lead to larger values of v_d. The relationship between α and T_m can be represented by the dimensionless factor β, which characterizes the overall chemotactic response [5]: $\beta = \alpha / T_m$.

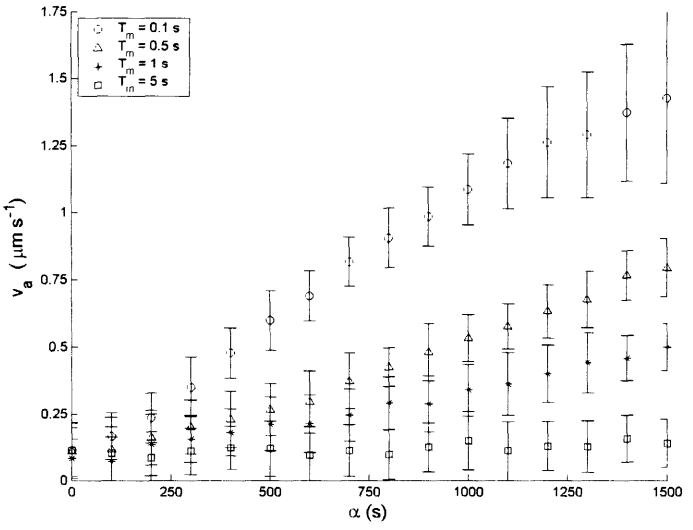

Figure 2.9: Relationship between parameters α and T_m. Increased values of T_m increase the time over which bacteria average concentration gradients and decrease bacterial sensitivity to concentration gradients. Increased values of α increase bacterial sensitivity to a given stimulus and lead to larger values of v_a. 5,000 bacteria were simulated for 50 seconds at each value of α for all values of T_m.

Bacteria sense alga cells with larger a more easily because larger alga cells have larger C_0 and the substrate concentration has a larger distance over which to decrease, giving bacteria more time to react to larger concentration fields. Therefore, as a increases v_a increases. Moreover, increasing the value of L also increases v_a because larger values of L result in larger values of C_0. Figure 2.10 illustrates these two effects. Furthermore, for $a < 5$ μm, v_a is approximately zero for relevant values of L, which implies that there is a minimum radius for which algal cells can be detected by bacteria. As mentioned earlier, increasing α increases bacterial sensitivity to a given stimulus and results in larger values of v_a. However, Figure 2.11 demonstrates that such an effect is only observed when the alga radius is above a minimum size ($a > 5$ μm).

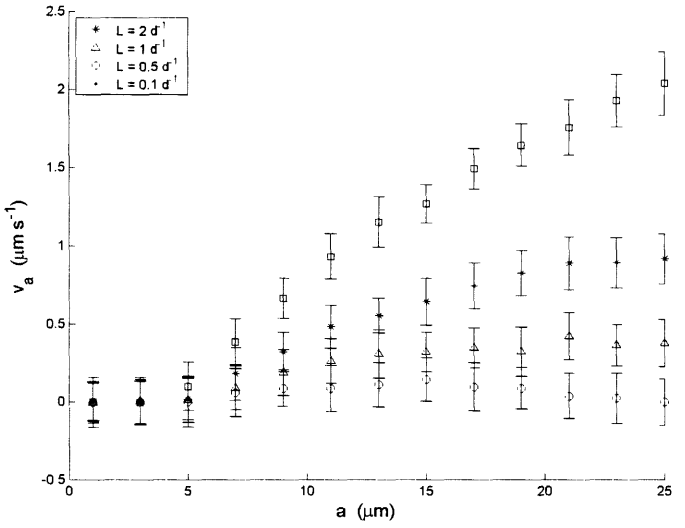

Figure 2.10: v_a increases as a or L increases. For $a < 5$ μm, v_a is approximately zero for relevant values of L, which implies that there is a minimum radius for which algal cells can be detected by bacteria. 5,000 bacteria were simulated for 50 seconds at each value of a for all values of L.

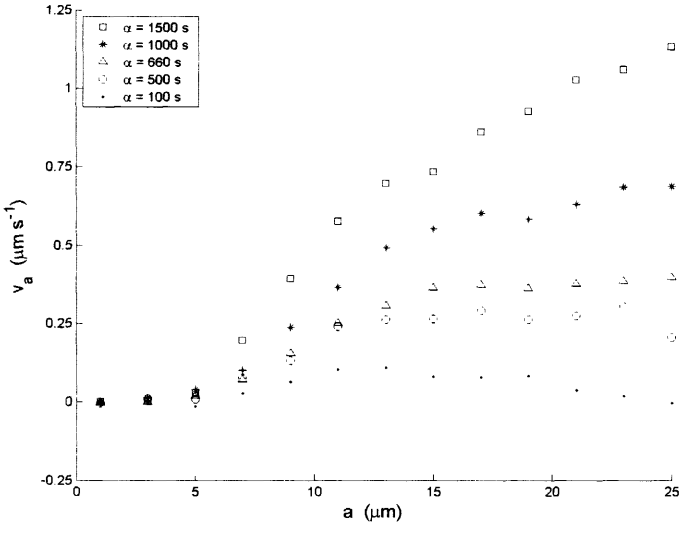

Figure 2.11: v_a increases as a or α increases. For $a < 5$ μm, v_a is approximately zero for relevant values of α, which implies that there is a minimum radius for which algal cells can be detected by bacteria. 5,000 bacteria were simulated for 50 seconds at each value of a for all values of α. Error bars were not displayed in order to improve clarity in the figure.

2.2.2. Geometric Considerations and Chemotactic Thresholds

In the following simulations the boundary constraint at R_{max} was lifted. Bacterial responses were simulated for an alga cell with a radius $a = 10$ μm and bacteria with a chemosensory constant $\alpha = 660$ s. Figure 2.12 shows the threshold effect associated with

the stationary particle problem when bacteria are not constrained by an upper distance bound. As the leakage rate L increases, the average position $\langle r(t) \rangle$ of a large number of bacteria decreases, which means that there is steady movement toward the alga cell. However, bacteria are not always attracted to the alga cell. In fact, if the attractant (i.e., the leakage rate L) is not above a certain threshold for a bacterium's chemosensory constant α, then the bacteria will move away from the alga cell. The critical value for the attractant is $L_{cr} = 2.2\ d^{-1}$ when $\alpha = 660$ s. At this threshold value, bacteria do not change their average distance from the alga. Above this threshold, the average distance decreases with time; while below this threshold, the average distance increases with time. Therefore, unless the attractant and chemosensory strength are above certain threshold, chemotaxis is disadvantageous. In Figure 2.12, 5,000 bacteria were started at $r_0 = 75\ \mu m$ for each chosen value of L and allowed to propagate for 50 seconds. Error bars associated with the data points were not plotted because they were approximately the size of the point markers.

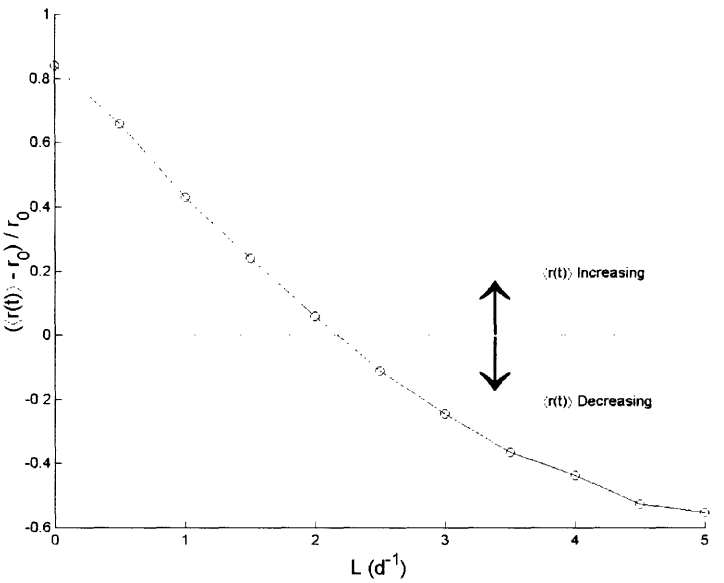

Figure 2.12: Threshold effect associated with bacteria propagating freely in three-dimensional space around a stationary particle. On average, bacteria travel away from the alga unless a chemotactic threshold is met. For $\alpha = 660\ s$, $L_{cr} = 2.2\ d^{-1}$. 5,000 bacteria were simulated for 50 seconds at chosen values of L. Error bars were approximately the size of the marker.

Even if one changes the distance that bacteria are initially positioned with respect to the algal cell, movement away from the alga is present for certain chemotactic parameters. In Figure 2.13 for $\alpha = 660$ s and $L = 1\ d^{-1}$, one notices that the average distance a population of bacteria move away by decays exponentially as r_0 increases. The decaying exponential shown in Figure 2.13 is given by: $(4.587 \pm 0.078)\exp[(-0.059 \pm 0.001)r_0]$, which has $\chi^2 = 4.226$. Essentially, as r_0 goes to infinity the average distance that a population of bacteria move away by goes to zero because at an infinite distance away from the alga cell the concentration becomes the background concentration, making $dC/dt = 0$ and causing

bacteria to move randomly without attraction in any direction. Therefore, unless $L > L_{cr}$ for a given α swimming is disadvantageous and a population of bacteria, on average, increases its distance from the attracting alga cell, which looks like repulsion. This effect is purely geometrical, such that bacteria distributed with respect to a fixed point randomly walk, expand the distribution, and on average increase the distribution's distance from the fixed point since chemotactic attraction is weak. In cases where these thresholds aren't met, bacteria would be better served if they were inert.

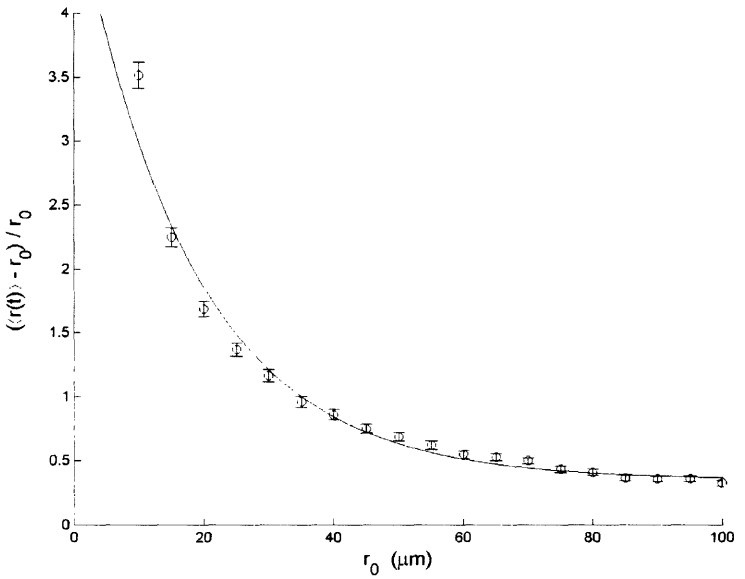

Figure 2.13: Inherent repulsion from alga. Regardless of the initial proximity to the alga cell, swimming causes the bacteria to move further away from the alga unless $L > L_{cr}$ for a given α. 5,000 bacteria with α = 660 s and L = 1 d^{-1} were simulated for 50 seconds at chosen values of r_0. This data was fit with a decaying exponential (χ^2 = 4.226).

2.3. Conclusions

Parameters used in simulations followed known results for *E. coli*. Some results that were first presented by Jackson (1987) were reproduced, leading to the following conclusions:

1. There is little difference in approach velocity v_a for either a biased or random distribution of tumble angle γ and fixed (non-zero) values of γ produce approximately the same v_a (0.123±0.018 μm s^{-1}, χ^2 = 0.965).

2. Increasing the background concentration C_r decreases the concentration gradient, which decreases chemosensory response and causes bacteria to approach the fixed alga cell more slowly.

3. The overall chemotactic response can be characterized by: $\beta = \alpha / T_m$, which bacteria respond to in the following manner. Increasing T_m (i.e., decay time associated with the temporal memory mechanism) increases

the time over which bacteria average concentration gradients and decrease bacterial sensitivity to concentration gradients. Increasing α (i.e., measure of how strongly attracted bacteria are to concentration gradients) increases bacterial sensitivity to a given stimulus and leads to larger values of v_a (an indication of chemosensory response).

4. v_a increases as a, L, or α increases. However, for $a < 5$ μm, v_a is approximately zero for relevant values of L and α, which implies that there is a minimum radius for which algal cells can be detected by bacteria.

Additionally, simulations were run with the boundary constraint at R_{max} lifted, which identified a threshold effect associated with the stationary particle problem. In cases where chemotactic thresholds for α and L aren't met, bacteria would be better served if they were inert. The following conclusions pertain to this threshold effect:

1. If the attractant (i.e., the leakage rate L) is not above a certain threshold for a bacterium's chemosensory ability α, the bacteria will move away from the alga cell. This effect is purely geometrical, such that bacteria distributed with respect to a fixed point randomly walk, expand the distribution, and on average increase the distribution's distance from the fixed point since chemotactic attraction is weak. This movement away from the fixed alga looks like repulsion.

2. Even if one changes the distance that bacteria are initially positioned with respect to the algal cell, movement away from the alga is present for certain chemotactic parameters. The average distance that a population of bacteria move away from a fixed alga cell decays exponentially as r_0 increases ($(4.587 \pm 0.078)\exp[(-0.059 \pm 0.001)r_0]$, $\chi^2 = 4.226$).

3. Analytical Moving Particle Problem

Bacterial chemotaxis toward an attracting particle that is moving through viscous fluid is a simple extension of the stationary particle problem. The moving particle can be thought of as a spherical alga cell or nutrient aggregate falling through fluid and leaking nutrients, thereby creating a concentration wake that attracts bacterial populations and affects the micro-environment. The model and results presented in this section follows work done by Jackson (1989).

3.1. Model

Mitchell et al. (1985) argued that bacterial chemotaxis, particularly for *E. coli*, is too slow for bacteria to collect around falling algae and estimated that the net rate of movement for chemotactic bacteria following a falling alga is about 10% of the bacterial run-mode swimming velocity, where the typical alga fall velocity is ~ 10 μm s^{-1} and bacterial swimming speed is ~ 10-30 μm s^{-1} [11]. However, the velocity of fluid around the falling alga is not constant but decreases near the particle because of viscosity. Therefore, if the fluid velocity relative to the falling alga is small, bacteria do not need to move as fast to stay near the alga. Moreover, bacteria can collect a high concentration of nutrients by positioning themselves in and traveling with the concentration wake. Figure 3.1 illustrates the moving particle problem. Notice that the concentration increases as one approaches the alga (radius a = 50 μm). Furthermore, there is a long wake in the concentration field produced by the fluid flow. The white lines in the figure are streamlines of the velocity field, which are equivalent to trajectories of inert bacteria (i.e., bacteria moving passively with the fluid).

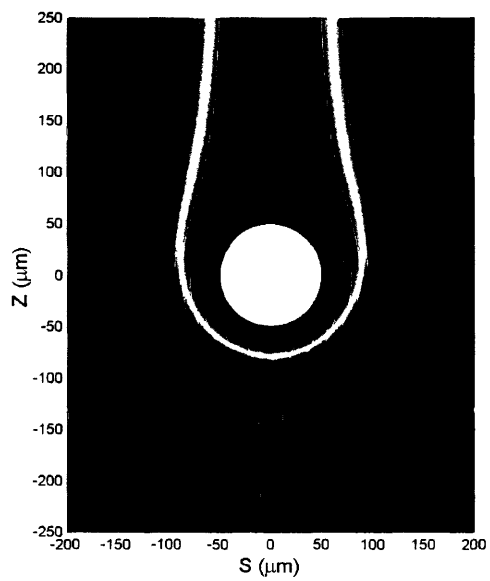

Figure 3.1: Moving particle problem. A spherical alga is centered at the origin and has radius a = 50 μm and nutrient concentration C_0 at its surface. Fluid flows positively along the z-axis passing the alga and creating a wake in the concentration field. The white lines are streamlines of the velocity field, which are equivalent to trajectories of inert bacteria moving passively with the fluid.

The difference between the stationary and moving particle problems is how the concentration and velocity fields are defined, which are affected by the alga fall speed (i.e., the settling velocity) and the size of the alga. The settling velocity of a spherical particle at low Reynolds number, $\mathrm{Re} = 2aw_s\nu^{-1}$ is given by:

$$w_s = \frac{2ga^2\Delta\rho}{9\nu\rho_0}, \tag{3.1}$$

where a is the spherical radius of the alga cell, ρ_0 is the fluid density, $\Delta\rho$ is the difference between the particle and fluid densities, ν is the kinematic viscosity, and g is the gravitational acceleration [11]. Re is a non-dimensional number relating a, w_s, and ν. Re gives a rough indication of the effects of inertia and viscosity and characterizes flow behavior. Re is understood when one considers the Navier-Stokes equations, which are equations of motion for an incompressible fluid of constant density (ρ_0) and viscosity (ν):

$$\begin{aligned} &\frac{\partial \vec{u}}{\partial t} + (\vec{u}\cdot\nabla)\vec{u} = -\frac{1}{\rho_0}\nabla p + \nu\nabla^2\vec{u} + \vec{g} \\ &\nabla\cdot\vec{u} = 0 \end{aligned}. \tag{3.2}$$

$\vec{u}$ is velocity, p is pressure, and $\vec{g}$ is gravitational acceleration. From the Navier-Stokes equations, one can derive Re by examining the inertia and viscosity terms and doing dimensional analysis [12]:

$$\frac{|inertia\ term|}{|viscosity\ term|} = \frac{|(\vec{u}\cdot\nabla)\vec{u}|}{|\nu\nabla^2\vec{u}|} = \frac{O\left((speed)^2/(length)\right)}{O\left((viscosity)(speed)/(length)^2\right)} = O\left(\frac{2aw_s}{\nu}\right). \tag{3.3}$$

Jackson obtained the following results by fitting data assembled by Smayda (1970) that pertained to phytoplankton size and fall velocities [10, 13]:

$$w_s = \xi f_\rho a^{1.17}, \tag{3.4}$$

where ξ is a fitting constant and f_ρ is a scaling parameter. Jackson accounted for differences in Equation 3.1 and 3.4 by assuming:

$$\Delta\rho = \frac{9\nu\rho_0\xi f_\rho}{2ga^{0.83}}. \tag{3.5}$$

Fluid flow for a alga falling through a still medium is equivalent to the alga being fixed in space and fluid moving past. Calculations are done in the alga's reference frame. The z-axis is the alga fall line, where flow travels from $-z$ to $+z$. Coordinates are (x, y, z), where $r = (x^2 + y^2 + z^2)^{1/2}$ is the distance from a bacterium to the alga center and $s = (x^2 + y^2)^{1/2}$ is the distance from the alga fall line. For $\mathrm{Re} < 1$, the flow around a sphere is given by the Stokes relationship in Cartesian coordinates:

$$u_x = w_s \frac{3axz}{4r^3}\left(\frac{a^2}{r^2}-1\right)$$
$$u_y = w_s \frac{3ayz}{4r^3}\left(\frac{a^2}{r^2}-1\right) \quad . \tag{3.6}$$
$$u_z = w_s\left[1-\frac{3a}{4r}\left(1+\frac{a^2}{3r^2}+\frac{z^2}{r^2}-\frac{a^2z^2}{r^4}\right)\right]$$

For low Péclet numbers (Pe = $2aw_sD^{-1} > 1$), which corresponds to lower alga fall velocities, molecular diffusion determines the concentration field. Pe relates forced convection (i.e., chemical emission driven by fluid flowing passed the alga) and diffusion (i.e., chemical emission driven by concentration gradients) and gives a rough indication of wake length where there is high chemical concentration. Pe can be derived in a manner similar to how Re was derived. Acrivos and Goddard (1962) used a singular perturbation expansion of Pe to calculate the concentration near the sphere (inner solution) and far from the sphere (outer solution) [10, 14]. To second order in Pe, the inner solution is:

$$C = C_0\left[\frac{a}{r}+\frac{Pe}{2}\left[\left(\frac{z}{r}-1\right)+\frac{a}{2r}\left(1-\frac{3z}{2r}\right)\right]\right], \tag{3.7}$$

and the outer solution is:

$$C = C_0\frac{a}{r}\exp\left[\frac{Pe(z-r)}{4a}\right]. \tag{3.8}$$

Because the full solution to the concentration field with Pe < 1 is defined by an inner and outer solution, one has to locate the region where the two solutions are approximately valid. This region occurs at a distance $r = 1.25a$ from the alga. The concentration field for Pe < 1 is shown in Figure 3.2. Near the alga, the concentration contours are approximately circular. But as one moves farther away, the contours become ovular.

For high Péclet numbers (Pe = $2aw_sD^{-1} > 1$), which corresponds to higher alga fall velocities fluid advection becomes more important than molecular diffusion in determining the concentration field. Acrivos and Goddard (1965) used an asymptotic method to calculate concentrations and fluxes around falling spheres for Pe > 1 using the incomplete gamma function and the gamma function [15]:

$$C = C_0\frac{\Gamma(1/3,\zeta)}{\Gamma(1/3)}$$
$$\zeta = \frac{Pe(r-a)^3\sin^3\psi}{3a^3\left(\psi-\frac{\sin 2\psi}{2}\right)}. \tag{3.9}$$
$$\psi = \cos^{-1}(-z/r)$$

Unlike the low Pe case, calculating the concentration field for the high Pe case is computationally more difficult because one has to reference the gamma function but is conceptually easier because one does not have to consider regions where only certain solutions are valid. Figure 3.3 illustrates the long concentration wake behind the alga, which characterizes the high Pe case.

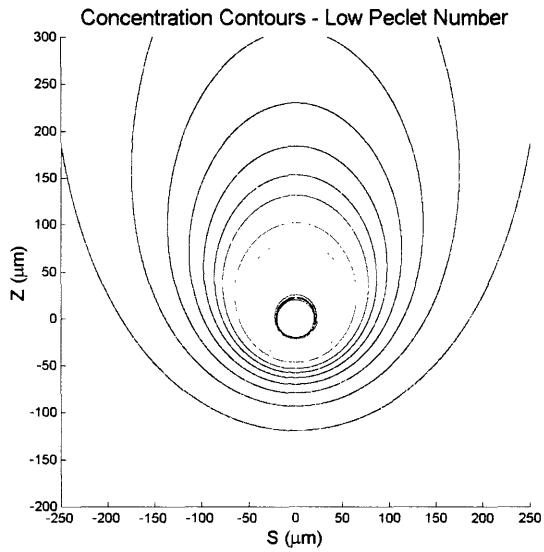

Figure 3.2: Concentration contour for low Pe case (Pe = 0.69).

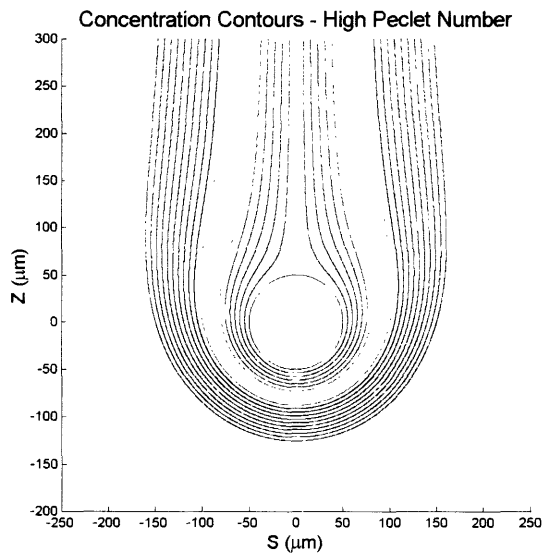

Figure 3.3: Concentration contours for high Pe case (Pe = 5.05).

C_0 is the concentration at the surface of the alga and is similar to the surface concentration in stationary particle problem. However, one must account for fluid motion increasing substrate flux from the alga by including the Sherwood number (Sh), which

modifies the relationship between C_0 and the alga loss rate. For low Pe, Sh = 1 + 0.25Pe; but for high Pe, Sh = 0.461 + 0.496 $Pe^{1/3}$. The variables φ, L, and D have the same meaning as in the stationary particle problem.

$$C_0 = \left(\frac{\phi L}{4\pi DSh} \right) a^{-1}. \tag{3.10}$$

The chemotaxis model that was applied to the stationary particle problem also applies to the moving particle problem. Because it was shown in the previous chapter that there is little difference in chemotactic response for either a biased or random γ, all simulations presented in the remainder of this work used a uniform distribution for γ because calculations were computationally easier to perform. Furthermore, the new direction a bacterium travels after a tumble can be defined by randomizing the angles of a spherical coordinate system: $\theta = 2\pi R$, $\varphi = \cos^{-1}(2R - 1)$. This allows one to work in a simple, spherical coordinate system instead of the alga-centered coordinate system defined by Jackson. After some time interval Δt, the change in a bacterium's position is given by $(\Delta x, \Delta y, \Delta z) = (v_0 \Delta t \cos\theta \sin\varphi, v_0 \Delta t \sin\theta \sin\varphi, v_0 \Delta t \cos\varphi)$. This change in position can be obtained by randomizing a bacterium's velocity $(\Delta x, \Delta y, \Delta z) = \vec{v}\Delta t$, where:

$$\vec{v} = v_0 \frac{\vec{R} - \frac{1}{2}}{\left\| \vec{R} - \frac{1}{2} \right\|}. \tag{3.11}$$

When determining the velocity of a bacterium, one must consider the velocity resulting from fluid flow. If the fluid velocity is given by $\vec{u} = (u_x, u_y, u_z)$ and the bacterial chemotactic velocity is given by $\vec{v} = (v_x, v_y, v_z)$, then the net bacterial velocity relative to the falling alga $\vec{V}$ is the sum of the fluid and bacterial chemotactic velocities:

$$\vec{V} = \vec{u} + \vec{v}. \tag{3.12}$$

Simulations started bacteria with $dC/dt = dP_b/dt = \overline{dP_b/dt} = 0$ in front of the falling alga at position $(0, y_i, -5a)$ and $t = 0$, where y_i is the initial position along the y-axis for the i^{th} bacterium. Values for P_t, r, s, and C were calculated at intervals of Δt. Due to the cylindrical symmetry of the problem, x can initially equal 0 without loss of generality, which makes $s = y_i$. Simulations followed a large number of bacteria for 50 seconds. If bacteria collided with the alga cell (i.e., $r < a$) they were reflected. There were no upper bounds on the distance a bacterium could travel away from the alga. A sample of the simulation code is presented in Section A.1.

Table 3.1 summarizes the parameters used in modeling the moving particle problem. These values were derived from studies conducted by Berg and Brown (1972, 1974) and are the same parameters used in Jackson's simulations (1989). The same parameter values are used throughout this chapter unless otherwise specified.

parameter	meaning	Low Pe Value	high pe value
a	Alga radius	20×10^{-4} cm	50×10^{-4} cm
w_s	Settling velocity	17.3×10^{-4} cm s^{-1}	50.5×10^{-4} cm s^{-1}
Re	Reynold's number	6.9×10^{-4}	50.5×10^{-4}
Pe	Péclet number	0.69	5.05
Sh	Sherwood number	1.17	1.31
C_0	Substrate concentration at $r = a$	2.7×10^{-9} mol cm^{-3}	8.7×10^{-9} mol cm^{-3}
C_∞	Background substrate concentration	0 mol cm^{-3}	0 mol cm^{-3}
L	Specific molecular leakage rate	0.5 d^{-1}	0.5 d^{-1}
φ	Carbon content of an alga	0.117×10^{-9} mol $cell^{-1}$	0.947×10^{-9} mol $cell^{-1}$
α	Chemosensory constant of bacterial system	660 s	660 s
T_m	Time constant of bacterial system	1 s	1 s
b	Curve fitting constant for alga	1.67×10^{-4} mol $cm^{-2.28}$	1.67×10^{-4} mol $cm^{-2.28}$
D	Substrate diffusivity	10^{-5} cm^2 s^{-1}	10^{-5} cm^2 s^{-1}
K_D	Chemosensory binding constant	1×10^{-7} mol cm^{-3}	1×10^{-7} mol cm^{-3}
τ_0	Unbiased mean run time	0.67 s	0.67 s
v_0	Bacterial swimming speed	12.5×10^{-4} cm s^{-1}	12.5×10^{-4} cm s^{-1}
Δt	Numerical time step	0.05 s	0.05 s
f_p	Fall velocity scale parameter	2	2
g	Gravitational acceleration	980 cm s^{-2}	980 cm s^{-2}
ν	Kinematic viscosity	0.01 cm^2 s^{-1}	0.01 cm^2 s^{-1}
ρ_0	Fluid density	1.03 g cm^{-3}	1.03 g cm^{-3}
ξ	Curve fitting constant for w_s	1.24	1.24

Table 3.1: Parameter values for the moving particle problem (Values taken from Reference 10)

3.2. Results

3.2.1. Background

Figures 3.4 and 3.5 show bacterial trajectories for three cases (inert, non-sensing, and sensing) simulated with low and high Pe conditions, respectively. In both figures, bacteria were initialized at 25 evenly spaced positions between $s = 0$ and $s = 5a$ and started at $z = -5a$. The inert case applies to bacteria that move passively with the fluid flow (i.e., $v_0 = 0$ cm s^{-1}, $\alpha = 0$ s). Inert trajectories represent streamlines in the velocity field. The rate at which inert bacteria move relative to the alga depends upon their initial position relative to the alga fall line. The non-sensing case applies to bacteria with no chemosensory ability (i.e., $\alpha = 0$ s). Non-sensing bacteria are unaffected by concentration gradients, thereby having no bias in run length or direction. The sensing case applies to chemotactic bacteria that seek out higher nutrient concentrations. For sensing bacteria, being initially positioned near the alga fall line not only moves bacteria through the concentration field more slowly but also exposes bacteria to higher substrate concentrations. Sensing bacteria for both low and high Pe are able to detect and move toward a falling alga; but given the current physical and chemotactic parameters, they are unable to swim sufficiently fast to remain near the alga indefinitely. Nevertheless, bacterial migration towards the falling alga and the trailing concentration wake increases exposure to higher substrate concentrations. Lastly, the length scales between Figures 3.4 and 3.5 are normalized by defining z from $-5a$ to $5a$ and s from 0 to $5a$.

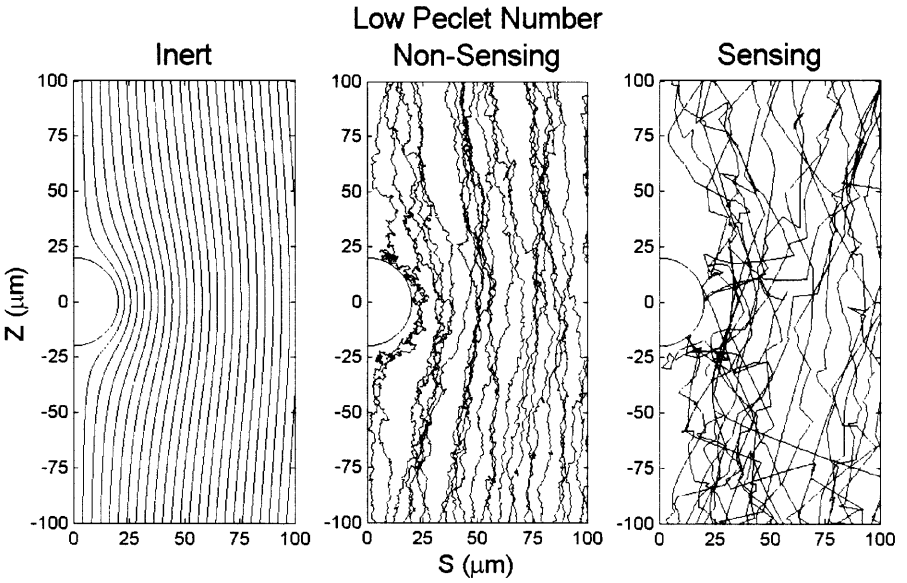

Figure 3.4: Trajectories for inert, non-sensing, and sensing bacteria for Pe = 0.69. Bacteria were initialized at at $z = -5a$ for 25 evenly spaced positions between $s = 0$ and $s = 5a$. Each bacterium was simulated for 100 seconds.

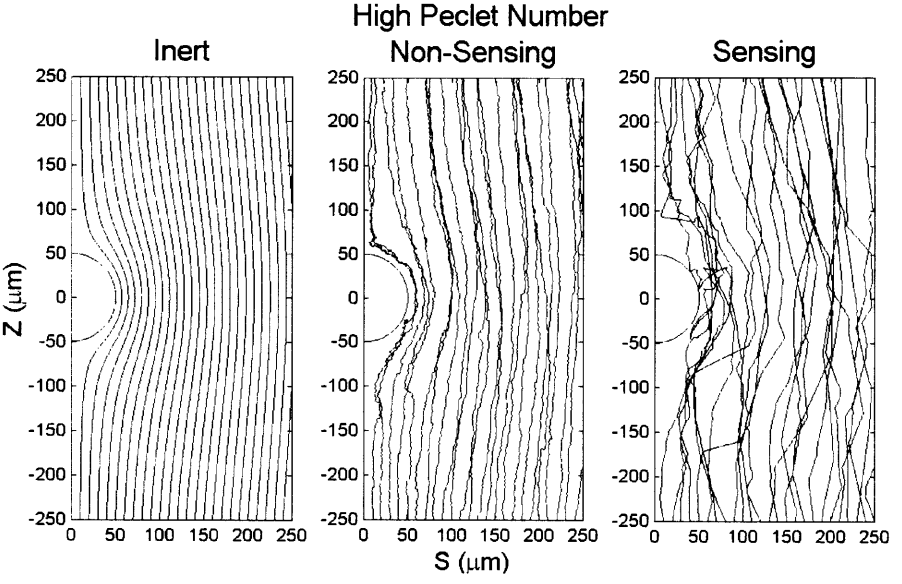

Figure 3.5: Trajectories for inert, non-sensing, and sensing bacteria for Pe = 5.05. Bacteria were initialized at at $z = -5a$ for 25 evenly spaced positions between $s = 0$ and $s = 5a$. Each bacterium was simulated for 100 seconds.

Figures 3.6 and 3.7 show average bacterial velocity $\langle v \rangle$ as a function of initial position from the alga fall line s_0. $\langle v \rangle$ was calculated by first determining the average time it takes for 1,000 bacteria, initially at a distance $z_0 = -5a$ below the alga and a distance s_0 from the alga fall line, to reach $z = 5a$ above the alga. Then, the distance $10a$ was divided by this

average time. For all cases (inert, non-sensing, and sensing), bacteria move more slowly relative to the alga the closer the bacteria are initially positioned. Moreover, for inert bacteria, $\langle v \rangle \to 0$ as $s_0 \to 0$. Near the alga fall line, $\langle v \rangle$ is larger for non-sensing and sensing bacteria than for inert bacteria because non-sensing and sensing bacteria have a random component to their velocity which causes them to drift outward and sample higher fluid flow velocities. It is favorable for bacteria to have a small $\langle v \rangle$ in the z-direction because it results in the bacteria having a longer exposure time to the concentration field. For all values of s_0, $\langle v_{sensing} \rangle < \langle v_{non-sensing} \rangle$ because sensing bacteria have a chemotactic mechanism that allows for the detection and migration to higher chemical concentrations, where the sensing bacteria are able to give some resistance to the fluid flow moving them away from the alga. Lastly, as s_0 increases, $\langle v \rangle$ for non-sensing and sensing bacteria is less than $\langle v \rangle$ for inert bacteria. There are some subtle physics associated with this last observation that will be investigated in Section 3.2. Figures 3.8 and 3.9 show how much greater $\langle v \rangle$ is for non-sensing and sensing bacteria when compared to $\langle v \rangle$ for inert bacteria. s associated with $\langle v \rangle / \langle v_{inert} \rangle = 1$ represents a critical value of the initial distance from the alga fall line s_{cr}. For $s_0 > s_{cr}$, $\langle v_{inert} \rangle > \langle v \rangle$ for either non-sensing or sensing bacteria. $\langle v \rangle / \langle v_{inert} \rangle$ for non-sensing and sensing bacteria decays exponentially as s_0/a increases for both low and high Pe.

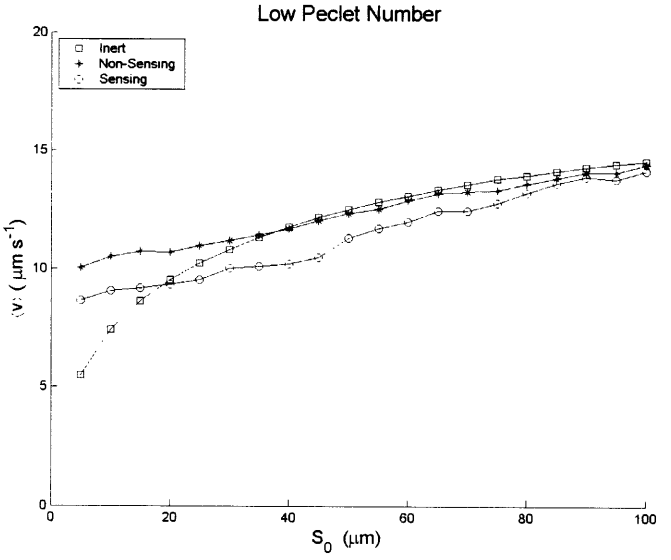

Figure 3.6: $\langle v \rangle$ for inert, non-sensing, and sensing bacteria for Pe = 0.69. 1,000 bacteria were initially positioned a distance $z_0 = -5a$ below the alga and a distance s_0 from the fall line. Each bacterium was simulated for 100 seconds. Error bars were approximately the size of the data marker.

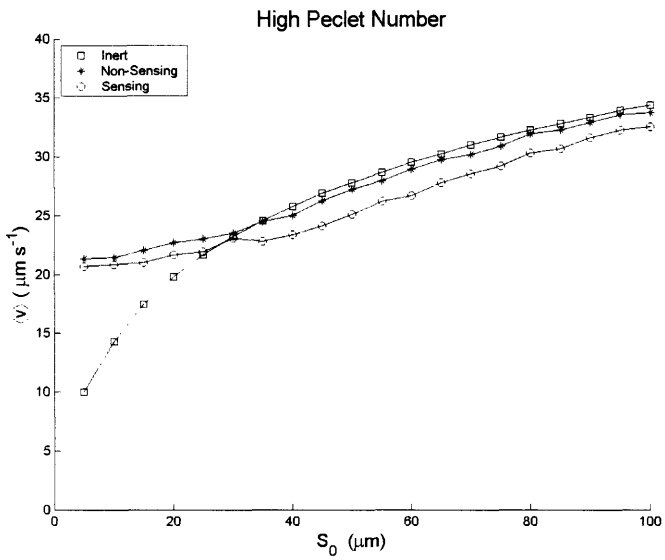

Figure 3.7: $\langle v \rangle$ for inert, non-sensing, and sensing bacteria for Pe = 5.05. 1,000 bacteria were initially positioned a distance z_0 = -5a below the alga and a distance s_0 from the fall line. Each bacterium was simulated for 100 seconds. Error bars were approximately the size of the data marker.

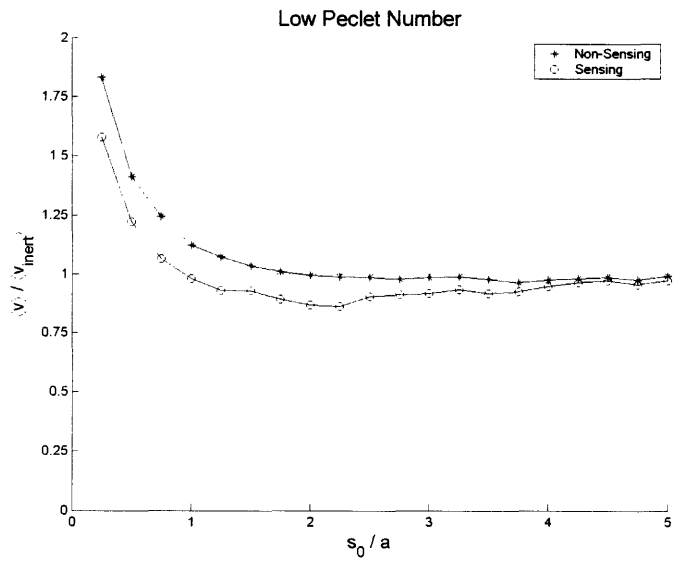

Figure 3.8: $\langle v \rangle / \langle v_{inert} \rangle$ for inert, non-sensing, and sensing bacteria for Pe = 0.69. 1,000 bacteria were initially positioned a distance z_0 = -5a below the alga and a distance s_0 from the fall line. Each bacterium was simulated for 100 seconds. Error bars were approximately the size of the data marker.

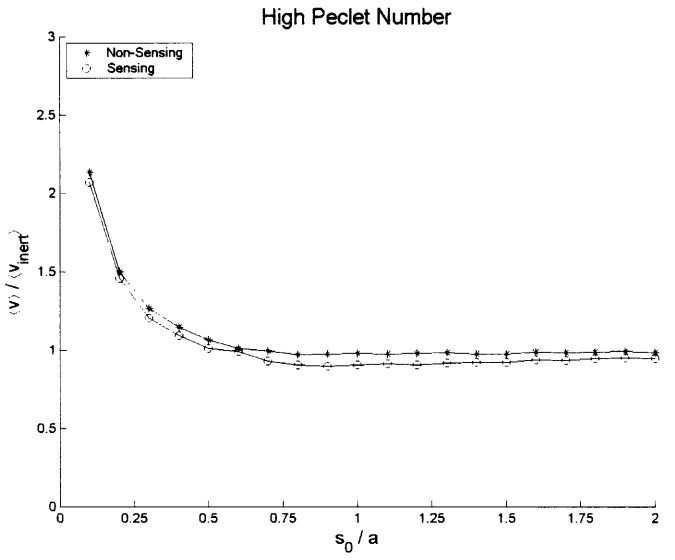

Figure 3.9: $\langle v \rangle / \langle v_{inert} \rangle$ for inert, non-sensing, and sensing bacteria for Pe = 5.05. 1,000 bacteria were initially positioned a distance $z_0 = -5a$ below the alga and a distance s_0 from the fall line. Each bacterium was simulated for 100 seconds. Error bars were approximately the size of the data marker.

Figures 3.10 and 3.11 illustrate that the average concentration $\langle C \rangle$ sampled by bacteria increases the closer the bacteria start with respect to the alga fall line. Close to $s_0 = 0$, $\langle C \rangle$ increases because the bacteria pass through regions of higher substrate concentration close to the alga and because the fluid moves slower close to the alga, keeping them in those higher concentration regions for longer durations. $\langle C \rangle$ was calculated by first summing the concentrations sampled by each bacterium as it traveled between $z = -5a$ and $5a$, multiplying by Δt, and dividing by the time t_{travel} it took a particular bacterium to travel between $z = -5a$ and $5a$. Then, these sample concentrations were averaged over the number of bacteria N starting at a certain s_0. See Equation 3.12. As s_0 increases, $\langle C \rangle$ for the non-sensing and sensing bacteria overtake $\langle C \rangle$ for the inert bacteria. Similar to what was seen with $\langle v \rangle$, there are critical values of s_{cr}, where $\langle C \rangle$ for inert bacteria is always less than $\langle C \rangle$ for either non-sensing or sensing bacteria $s_0 > s_{cr}$. There are some subtle physics associated with this last observation that will be investigated in Section 3.2.

$$\langle C \rangle = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{t_{travel}} \sum_{z=-5a}^{5a} C_i(z) \Delta t \,. \qquad 3.12$$

Figures 3.12 and 3.13 show how much greater $\langle C \rangle$ is for non-sensing and sensing bacteria when compared to $\langle C \rangle$ for inert bacteria. s associated with $\langle C \rangle / \langle C_{inert} \rangle = 1$ represents a critical value of the initial distance from the alga fall line s_{cr}. For $s_0 > s_{cr}$, $\langle C_{inert} \rangle < \langle C \rangle$ for either non-sensing or sensing bacteria. There is a qualitative difference

between Figures 3.12 and 3.13. In Figure 3.12, as s_0/a increases $\langle C \rangle / \langle C_{inert} \rangle$ also increases but then levels off after passing $s_0/a = 2$. Yet in Figure 3.13, as s_0/a increases, $\langle C \rangle / \langle C_{inert} \rangle$ increases and seems to have leveled off until $s_0/a = 1.5$, where $\langle C \rangle / \langle C_{inert} \rangle$ increases exponentially. It is possible that the exponential growth in $\langle C \rangle / \langle C_{inert} \rangle$ at large values of s_0/a for the high Pe case could also be seen for the low Pe case. Nevertheless, non-sensing and sensing bacteria obtain much higher average concentrations than inert bacteria far away from the alga, which seems to be a likely case in ecosystems where nutrients (i.e., alga cells) are diffuse.

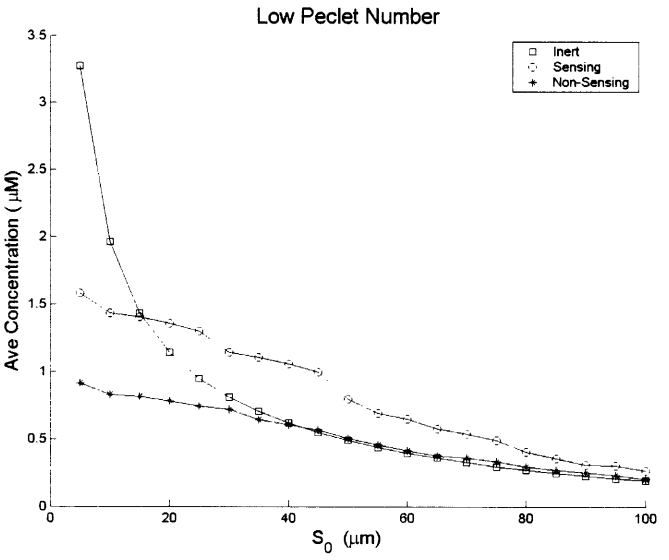

Figure 3.10: $\langle C \rangle$ for inert, non-sensing, and sensing bacteria for Pe = 0.69. 1,000 bacteria were initially positioned a distance $z_0 = -5a$ below the alga and a distance s_0 from the fall line. Each bacterium was simulated for 100 seconds. Error bars were approximately the size of the data marker.

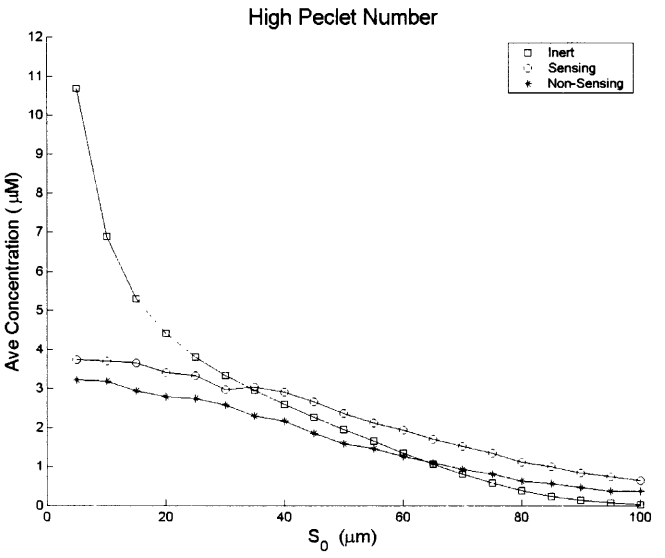

Figure 3.11: $\langle C \rangle$ for inert, non-sensing, and sensing bacteria for Pe = 5.05. 1,000 bacteria were initially positioned a distance z_0 = -5a below the alga and a distance s_0 from the fall line. Each bacterium was simulated for 100 seconds. Error bars were approximately the size of the data marker.

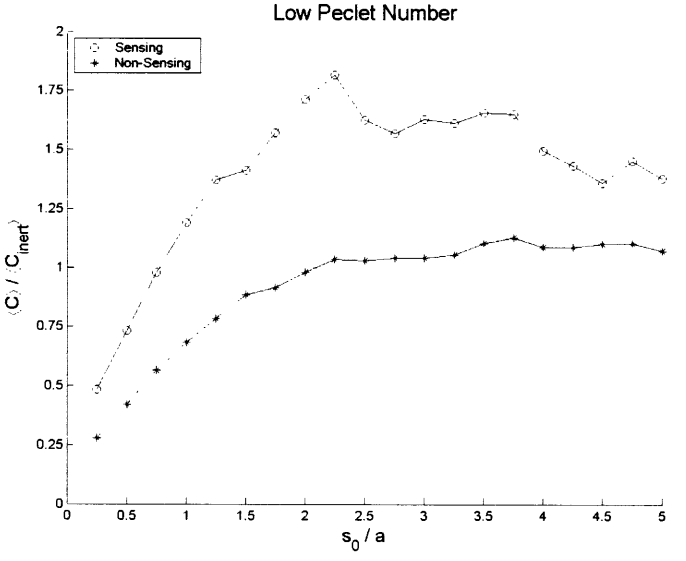

Figure 3.12: $\langle C \rangle / \langle C_{inert} \rangle$ for inert, non-sensing, and sensing bacteria for Pe = 0.69. 1,000 bacteria were initially positioned a distance z_0 = -5a below the alga and a distance s_0 from the fall line. Each bacterium was simulated for 100 seconds. Error bars were approximately the size of the data marker.

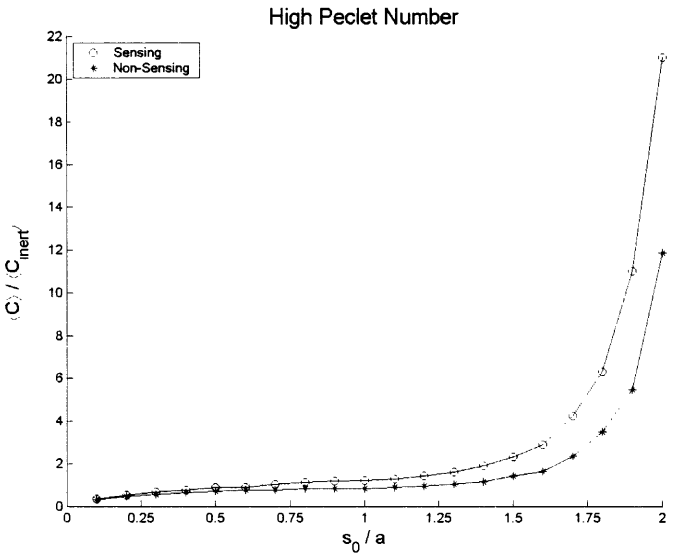

Figure 3.13: $\langle C \rangle / \langle C_{inert} \rangle$ for inert, non-sensing, and sensing bacteria for Pe = 5.05. 1,000 bacteria were initially positioned a distance $z_0 = -5a$ below the alga and a distance s_0 from the fall line. Each bacterium was simulated for 100 seconds. Error bars were approximately the size of the data marker.

Contact time T_c is a measure of nutritional enhancement and is equivalent to the time interval that an average bacterium spends next to the alga cell. By maintaining a position near the falling alga, sensing bacteria achieve a higher average substrate concentration. T_c was calculated by first summing the concentrations sampled by each bacterium as it traveled between $z = -5a$ and $5a$ and multiplying by Δt. Then, these sample concentrations were averaged over the number of bacteria N and divided by C_0 to obtain units of time.

$$T_c = \frac{1}{NC_0} \sum_{i=1}^{N} \sum_{z=-5a}^{5a} C_i(z) \Delta t \,. \tag{3.13}$$

In the following simulations, bacteria were dispersed over different values of s. The initial position for bacteria: $(0, y_i, -5a)$ was far enough upstream for bacteria to be relatively unaffected by the substrate concentration field, while being close enough so that bacteria would not drift too far from the alga before sensing it. S_{max} is the maximum distance that bacteria could swim to intercept the falling alga before the alga passes below the bacteria. For the following simulations, 1,000 bacteria were dispersed in groups of 20 bacteria at 50 evenly spaced intervals between $y = 0$ and $y = S_{max}$.

$$S_{\max} = \frac{5av_0}{w_s} \,. \tag{3.14}$$

Figure 3.14 shows that as a or L increases, T_c increases. These results make sense because T_c is proportional to C, which is proportional to L and $\varphi/a = ba^{1.28}$. In the

previous section, approach velocity v_a was used as an indicator of chemosensory ability that increased with increasing L. Here, T_c is also an indicator of chemosensory ability that increases with increasing L. However, a is the dominant factor for increasing T_c. Although both a and L increase C and subsequently the concentration gradients, increasing a also affects the fluid flow. The larger the value of a, the farther out from the alga fall line there exists slower moving fluid, which bacteria can use to maneuver near the falling alga.

Figure 3.15 illustrates that as a increases or v_0 decreases, T_c increases. Part of this effect results from the smaller S_{max} associated with smaller v_0, which moves the initial position of bacteria closer to the alga and increases T_c. These results are also associated with the data presented in Figures 3.10 and 3.11, where the average concentration was compared for inert, non-sensing, and sensing bacteria. It was found that at some values of s from the alga fall line, inert bacteria obtained higher average concentrations. It would seem that in some cases smaller v_0 is favorable to sensing bacteria even if it prevents them from migrating to regions of higher concentration more effectively. It can be assumed that bacteria initially positioned far away from the alga fall line do not contribute as much as bacteria initially positioned close to the alga fall line. Therefore, the dominant factor in determining T_c is the concentration felt by bacteria initially positioned close to the alga fall line, which is the region where inert bacteria obtain higher average concentrations than non-sensing or sensing bacteria. Intuitively, this does not make sense. Why do inert bacteria or bacteria with small v_0 ever have higher average concentrations than sensing bacteria? This question will be investigated in Section 3.2.

Figure 3.16 illustrates that as a increases or f_ρ decreases, T_c increases. This is obvious because $w_s \sim f_\rho$. Slower alga fall speeds give bacteria more of an opportunity to detect, migrate, and remain near the alga, which increases bacterial nutrient exposure and T_c.

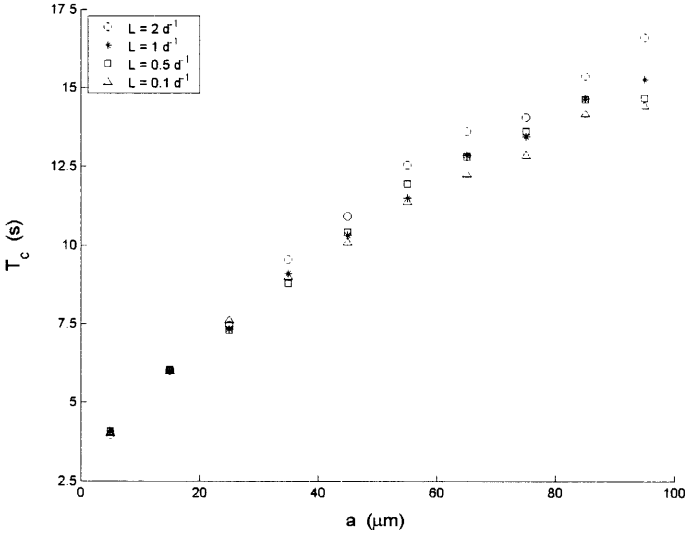

Figure 3.14: T_c increases as a or L increases. 1,000 bacteria were dispersed in groups of 20 bacteria at 50 evenly spaced intervals between $y = 0$ and $y = S_{max}$.

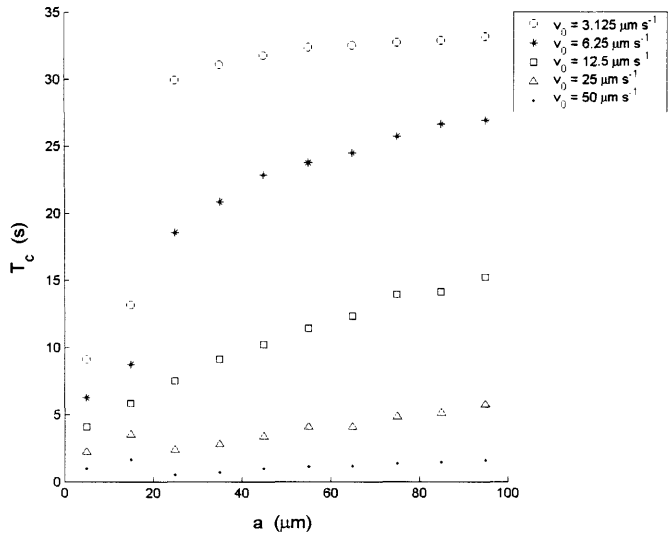

Figure 3.15: T_c increases as a increases but decreases as v_0 increases. Smaller v_0 results in larger T_c, which corresponds with what was observed in Figures 2.24 and 2.25. 1,000 bacteria were dispersed in groups of 20 bacteria at 50 evenly spaced intervals between $y = 0$ and $y = S_{max}$.

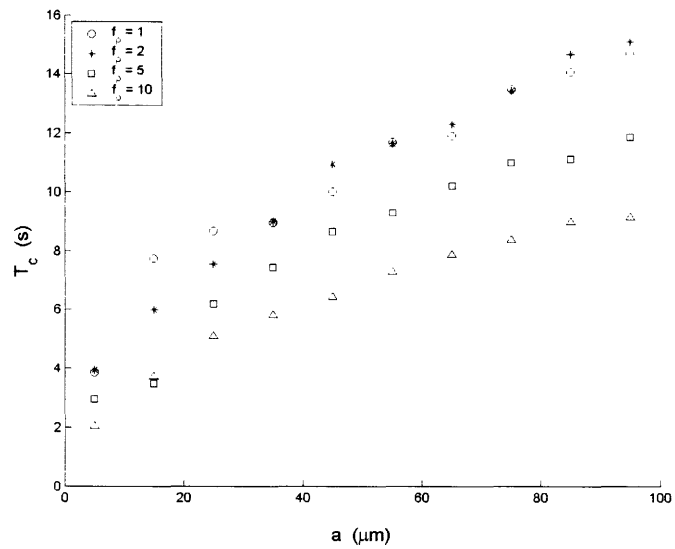

Figure 3.16: T_c increases as a increases but decreases as f_ρ increases. This is obvious since $w_s \sim f_\rho$. 1,000 bacteria were dispersed in groups of 20 bacteria at 50 evenly spaced intervals between $y = 0$ and $y = S_{max}$.

3.2.2. Flow Considerations and Chemotactic Thresholds

There are interesting physics associated with the velocity field. For the results presented in this section, only the high Péclet number case (Pe = $2aw_sD^{-1} > 1$), which corresponds to high alga fall velocities is investigated because it is computationally straightforward and does not rely on inner and outer solutions to the concentration field. Figure 3.17 shows the velocity profile for u_z/w_s as a function of distance from the alga fall line for different values of z/a. The velocity profile in the z direction is not linear going from convex to concave to w_s in the limit as $s \to \infty$. Figure 3.18 shows inflection points of u_z (i.e., where u_z goes from convex to concave) as a function of s/a. Since bacteria tumble randomly, they have equal probability of sampling any portion of the fluid velocity field. However, because u_z/w_s is convex near the alga, a small change in s produces a large change in the fluid velocity felt by bacteria. This effect is evidenced in the zoomed-in illustration of the velocity profile for $z/a = 2$. If bacteria move to larger values of s, they will feel higher fluid velocities and be pushed away from the alga more quickly. Additionally, the higher the fluid velocity in the z direction, the less time there is for bacteria to sample nutrient concentrations. In the zoomed-in illustration, if a bacterium moves to the left the transit time increases but if a bacterium moves to the right the transit time decreases. In such cases where the velocity field changes so dramatically, it would be more advantageous if the bacteria had not moved at all. The shape of u_z results from the alga's spherical geometry.

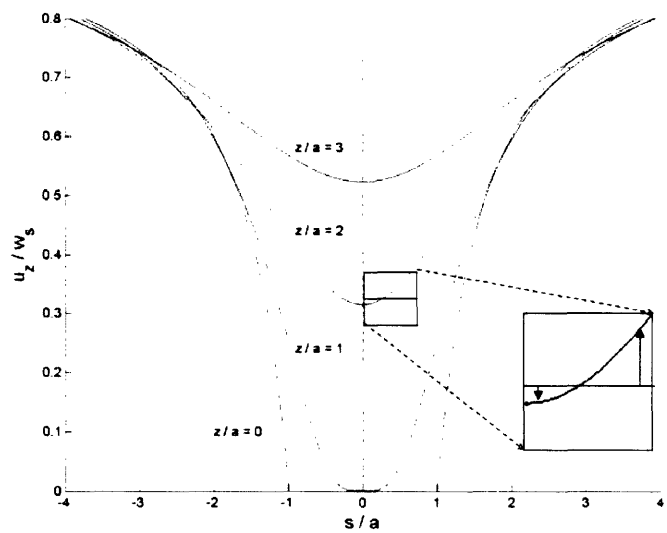

Figure 3.17: Profile of u_z/w_s for z/a = 0, 1, 2, and 3 μm at different values of s/a. By randomly swimming, bacteria sample equal portions of the velocity field. The convexity in u_z/w_s at small values of s/a means a minute change in s/a produces a large change in u_z/w_s, where the larger the velocity the less time bacteria have to sample nutrients. The shape of u_z/w_s results from the alga's spherical geometry.

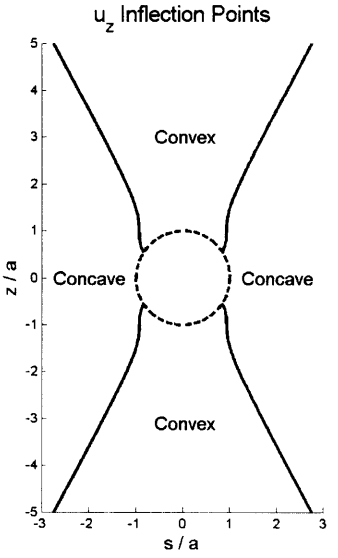

Figure 3.18: Inflection points of u_z (shown in solid black) as a function of s/a. The inflection points separate regions of convexity and concavity. The radius of the alga is drawn with a dashed line. The inflection points were solved for numerically by finding where the second derivative of u_z in Equation 3.6 equaled zero.

A useful quantity for examining chemotactic response in the moving particle problem is the non-dimensional average integrated nutrient exposure I, which is plotted in Figure 3.19 as a function of non-dimensional initial position s_0/a. Here, one observes the dramatic effect of the velocity field u_z going from convex to concave as the distance from the alga fall line s increases. I is a measure of the amount of substrate N bacteria have been exposed to as those bacteria travel between $z = -5a$ and $z = +5a$ and is given by:

$$I = \frac{w_s}{10aC_0}\frac{1}{N}\sum_{i=1}^{N}\int_t C_i dt = \frac{w_s}{10aC_0}\frac{1}{N}\sum_{i=1}^{N}\sum_{z=-5a}^{+5a} C_i(z)\Delta t\,. \qquad 3.15$$

C is integrated over the time that the i^{th} bacterium takes to travel between $z = -5a$ and $z = +5a$. I is non-dimensionalized by multiplying by w_s and dividing by C_0 and $10a$ (the length scale over which a bacterium's sampled concentration is being summed). Very close to the alga (e.g., $s_0/a < 0.5$), I_{inert} dominates $I_{sensing}$ and $I_{non\text{-}sensing}$. However, as s_0/a increases, I_{inert} decreases exponentially and is overtaken by $I_{sensing}$ at $s_0/a = 0.680 \pm 0.100$ and by $I_{non\text{-}sensing}$ at $s_0/a = 1.272 \pm 0.100$. Later at $s_0/a = 2$, $I_{sensing}$ is ~20 times greater than I_{inert} and $I_{non\text{-}sensing}$ is ~12 times greater than I_{inert}. Again, inert bacteria do as much as 3 times better than motile bacteria when near the alga because inert bacteria only move passively with the fluid flow and never sample u_z for different values of s. Therefore, just by moving, motile chemotactic bacteria put themselves at a disadvantage when close to an algal cell. This disadvantage illustrates that there are thresholds for certain chemotactic parameters. For the data presented in Figure 3.19, 1,000 bacteria were simulated at each value of s_0/a for 50 seconds. Error bars were approximately the size of the data markers.

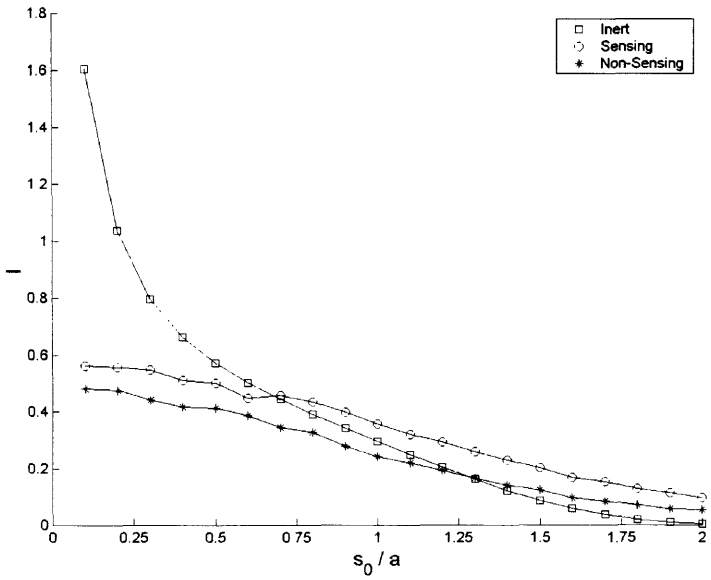

Figure 3.19: Non-dimensional average integrated nutrient exposure I as a function of initial position s_0/a. I was calculated as bacteria moved between -5a and 5a. Sensing bacteria do better than inert bacteria for $s_0/a \geq 0.680 \pm 0.100$ and collect approximately 20 times more nutrients than inert bacteria at $s_0/a = 2$. Likewise, non-sensing bacteria do better than inert bacteria for $s_0/a \geq 1.272 \pm 0.100$ and collect approximately 12 times more nutrients than inert bacteria at $s_0/a = 2$. This effect is a result of u_z going from convex to concave. 1,000 bacteria were simulated for each value of s_0/a. Error bars were approximately the size of the data markers.

The chemotactic threshold for the moving particle problem is examined in Figure 3.20, where $s_0/a = 0.75$ and $L = 0.5\ d^{-1}$ for sensing bacteria. As α increases, I increases until it overtakes I_{inert} at some critical value of α (α_{cr}), which is 430 s for this case. Below $I - I_{inert} = 0$, sensing bacteria are less effective than inert bacteria and chemotaxis shows to be disadvantageous. When $\alpha = 0$, the sensing bacteria become non-sensing. Therefore, one can also examine I for non-sensing bacteria by extending a horizontal line across all values of α for $I - I_{inert}$ when $\alpha = 0$. The value of $I - I_{inert}$ when $\alpha = 0$ gives the initial disadvantage chemotactic bacteria face just by moving. As mentioned before, it takes a minimum chemosensory strength to overcome this initial disadvantage. The data shown in Figure 3.20 was fit with: $(0.027 \pm 0.001)\ln[(0.003 \pm 0.001)(\alpha - (50.198 \pm 15.161))]$, which had $\chi^2 = 0.985$. 1,000 bacteria were simulated at each value of α for 50 seconds. Error bars were calculated at each value of α by taking the standard associated with $I - I_{inert}$.

By determining α_{cr} for different values of s_0/a with L fixed ($L = 0.5\ d^{-1}$ in the following simulation), one obtains the plot in Figure 3.21, which shows three operational regimes. First, for $s_0/a < 0.5$ sensing bacteria are always less effective than inert bacteria. Second, for $0.5 < s_0/a < 1.375$ sensing bacteria are more effective than inert bacteria provided their chemosensory ability exceeds a threshold ($\alpha > \alpha_{cr}$). Third, for $1.375 < s_0/a$ sensing bacteria are always more effective than inert bacteria. The data shown in Figure 3.20 was fit with: $(4.078 \pm 0.249 \times 10^5)\exp[(-8.919 \pm 0.095)s_0/a]$, which had $\chi^2 = 0.394$. This data was obtained by measuring α_{cr} for plots like Figure 3.20 but with different values of s_0/a. Error bars corresponded by our ability to visually determine α_{cr} for certain values of s_0/a.

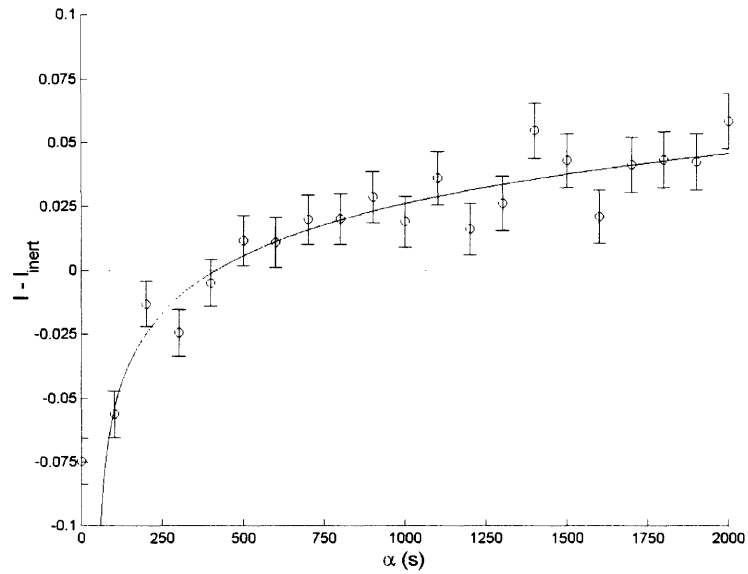

Figure 3.20: Determining chemotactic threshold. Sensing bacteria do better than inert bacteria when $\alpha > \alpha_{cr}$, chemotaxis is not beneficial. The value of $I - I_{inert}$ associated with $\alpha = 0$, corresponds to non-sensing bacteria. $\alpha_{cr} = 430$ for $s_0/a = 0.75$ and $L = 0.5\ d^{-1}$. 1,000 bacteria were simulated at each value of α. This data was fit with a logarithmic function ($\chi^2 = 0.985$).

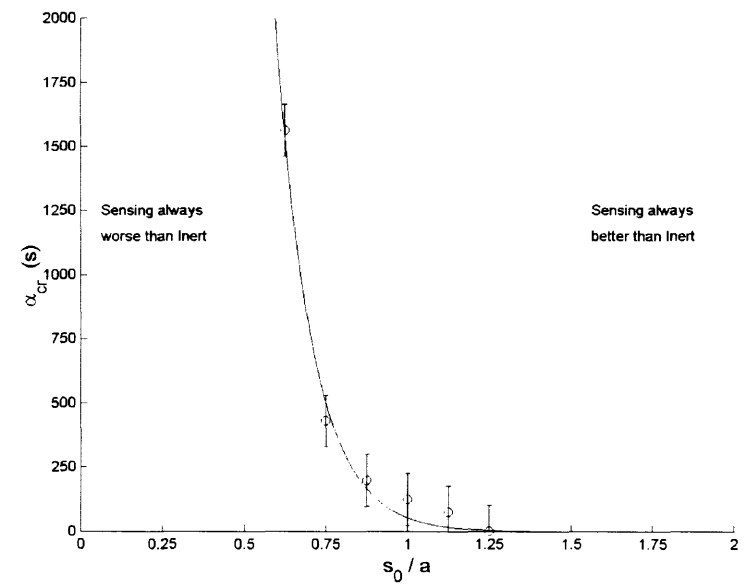

Figure 3.21: α_{cr} as a function of initial position. There are three dynamical regimes associated with sensing bacteria: i) sensing bacteria always perform worse than inert bacteria, ii) sensing bacteria perform better than inert bacteria when a chemotactic threshold is met, and iii) sensing bacteria always perform better than inert bacteria. Here, these three regimes are separated by dotted in lines. This data was fit with a decaying exponential ($\chi^2 = 0.394$).

3.3. Conclusions

The model for bacterial chemotaxis toward a particle moving through viscous fluid was developed from the model for chemotaxis toward a stationary particle. The essential difference between these two problems is how the concentration and velocity fields are defined, which is affected by the alga's sinking speed and size. Simulations were carried out for cases with high and low Pe. Parameters used in simulation followed known results for *E. coli*. Some results that were first presented by Jackson (1989) were reproduced, leading to the following conclusions:

1. Chemotaxis is too slow for bacteria to collect around falling algae. However, if the fluid velocity decreases near the alga because of viscosity, then bacteria do not need to move as fast to stay near the alga. Moreover, bacteria can collect a high concentration of nutrients by positioning themselves in and traveling with the concentration wake.

2. It is favorable for bacteria to have a small $\langle v \rangle$ in the z-direction because it results in the bacteria having a longer exposure time with respect to the concentration field. There is a critical value of the initial distance from the alga fall line s_{cr}, associated with $\langle v \rangle / \langle v_{inert} \rangle = 1$, such that for $s_0 > s_{cr}$, $\langle v_{inert} \rangle > \langle v \rangle$ for either non-sensing or sensing bacteria. $\langle v \rangle / \langle v_{inert} \rangle$ for non-sensing and sensing bacteria decays exponentially as s_0/a increases for both low and high Pe. For all values of s_0, $\langle v_{sensing} \rangle < \langle v_{non-sensing} \rangle$.

3. There is a critical value of the initial distance from the alga fall line s_{cr}, associated with $\langle C \rangle / \langle C_{inert} \rangle = 1$, such that for $s_0 > s_{cr}$, $\langle C_{inert} \rangle < \langle C \rangle$ for either non-sensing or sensing bacteria. Non-sensing and sensing bacteria obtain much higher average concentrations than inert bacteria far from the alga, which seems a likely case in ecosystems where nutrients (i.e., alga cells) are diffuse. The s_{cr} associated with $\langle C \rangle / \langle C_{inert} \rangle = 1$ is not the same as the s_{cr} associated with $\langle v \rangle / \langle v_{inert} \rangle = 1$.

4. Contact time T_c is a measure of nutritional enhancement and is equivalent to the time interval that an average bacterium spends next to the alga cell. T_c increases with increasing a, L and decreasing f_ρ, v_0. Increasing a has a dominant effect on increasing T_c because the larger the value of a, the farther out from the alga fall line there exists slower moving fluid, which bacteria can use to maneuver near the falling alga. Increasing L increases T_c because $L \sim C \sim T_c$. a also increases T_c for this reason ($a^{1.28} \sim C \sim T_c$). Decreasing f_ρ increases T_c because $w_s \sim f_\rho$ and alga sinking speeds give bacteria more of an opportunity to detect, migrate, and remain near the alga. Lastly, decreasing v_0 increases T_c because s_{max} is associated with v_0 and smaller v_0 moves the initial position of bacteria closer to the alga.

However, this last effect is associated with how bacteria sample u_z and will be explained in the conclusions below.

One particularly interesting aspect of the moving particle problem is the velocity field u_z goes from convex to concave as the distance from the alga fall line s increases. If bacteria move to larger values of s, they feel higher fluid velocities and are pushed away from the alga more quickly. Additionally, the higher the fluid velocity in the z direction (u_z), the less time there is for bacteria to sample nutrient concentrations. In such cases where the velocity field changes so dramatically, it would be more advantageous for bacteria had not move at all. However, there is a threshold effect for α associated with chemotaxis toward a falling alga that explains when the mechanism is advantageous and is supported by the following conclusions:

1. Under standard conditions, inert bacteria do much better than motile bacteria when near the alga because inert bacteria only move passively with the fluid flow and never sample u_z for different values of s. Just by moving, motile chemotactic bacteria put themselves at a disadvantage because u_z is convex near the alga. Overcoming this disadvantage means meeting a chemotactic threshold for α.

2. When $\alpha = 0$, $I - I_{inert}$, the non-dimensional average integrated nutrient exposure, gives the initial disadvantage chemotactic bacteria face just by moving. As α increases, $I - I_{inert}$ increases. α_{cr} is the chemotactic threshold for the moving particle problem and is defined when $I - I_{inert} = 0$. However, below $I - I_{inert} = 0$, sensing bacteria are less effective than inert bacteria and chemotaxis shows to be disadvantageous.

3. By determining α_{cr} for different values of s_0/a with L fixed, one obtains the three operational regimes associated with this threshold effect. These three operational regimes are defined by: i) sensing bacteria always being less effective than inert bacteria, ii) sensing bacteria being more effective than inert bacteria provided α_{cr} is met, or iii) sensing bacteria always being more effective than inert bacteria. This suggests that bacteria must have evolutionarily developed chemotaxis such that these thresholds were met and the mechanism was biologically successful.

4. Vortices and Numerical Moving Particle Problem

4.1. Computational Fluid Dynamics and Finite Element Method

Computational fluid dynamics (CFD) is a very powerful method that uses numerical analysis to predict the physical properties of complicated fluid flows. CFD allows one to investigate a wide variety of problems that are analytically intractable but computationally feasible. Here, the concentration, velocity, vorticity, and pressure fields were computed as a function of time until a steady state solution (i.e., solution does not change with time) was reached. Then, these fields were imported into the chemotaxis simulator to see how bacteria responded to more complicated flows with specific geometric and boundary conditions.

The Finite Element Method (FEM) was employed via FEMLAB (Finite Element Method Laboratory), which is a software package produced by COMSOL that interfaces easily with MATLAB and allows for the modeling of multiple, simultaneous physical processes. FEM gives CFD solutions by making discrete approximations to the physical model of a system defined by certain parameters, equations, and boundary conditions. Figure 4.1 shows an example of this discretization. The perimeter of a circle is approximated by dividing the circle (i.e., the physical system) into a mesh (i.e., a collection) of three-sided polygons (i.e., discrete approximations). The sides of the polygons are elements and points where sides meet are nodes [16].

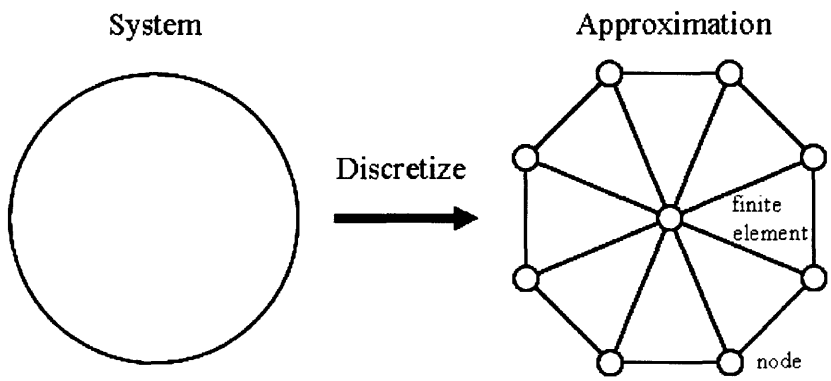

Figure 4.1: The rudiments of the finite element method, where a continuous system is discretized into a mesh of finite elements. The sides of each finite element are known as elements and nodes are points where elements intersect [16].

After a system is discretized, the mesh is disassembled into separate polygons (i.e., finite elements) so the solution to the boundary value problem can be calculated over the domain of each polygon. Each finite element is characterized by its degrees of freedom, represented by the set of node points. The solution to a CFD problem is calculated at the node points and approximated locally over the domain of each finite element by interpolation. The union of discretized solutions over adjacent finite elements are patched together to form a trial solution [16], which is tested for convergence. There are errors

associated with numerical analysis. However, a more accurate solution is obtained by using a finer mesh (i.e., implementing more node points).

For the results investigated, a model was generated to reflect vortices in an experimental setup, where fluid with a certain speed enters a wide channel, passes over a stationary particle (i.e., an alga cell or aggregate leaking nutrients), and leaves the channel which has zero exit pressure [17]. Results for these simulations are presented in Section 4.4 and 4.5. To simplify the modeling, computation was done in two-dimensions (x and y, i.e., the horizontal and vertical axis). The FEMLAB modeling is outlined below:

1. Create a multi-physics model defined by the incompressible Navier-Stokes equation and convection/diffusion.
2. Enter parameters associated with the physical system, such as fluid density, kinematic viscosity, concentration at the surface of the particle, inlet speed of fluid, and diffusivity.
3. Draw the model geometry. The cylinder of fluid is modeled by a rectangle and the particle is modeled by a circle.
4. Define the boundary and sub-domain conditions for both the Navier-Stokes and convection/diffusion sub-systems.
5. Initialize and refine the mesh around the alga cell. The mesh used in solving the CFD problem is shown in Figure 4.2.
6. Solve the boundary value problem.

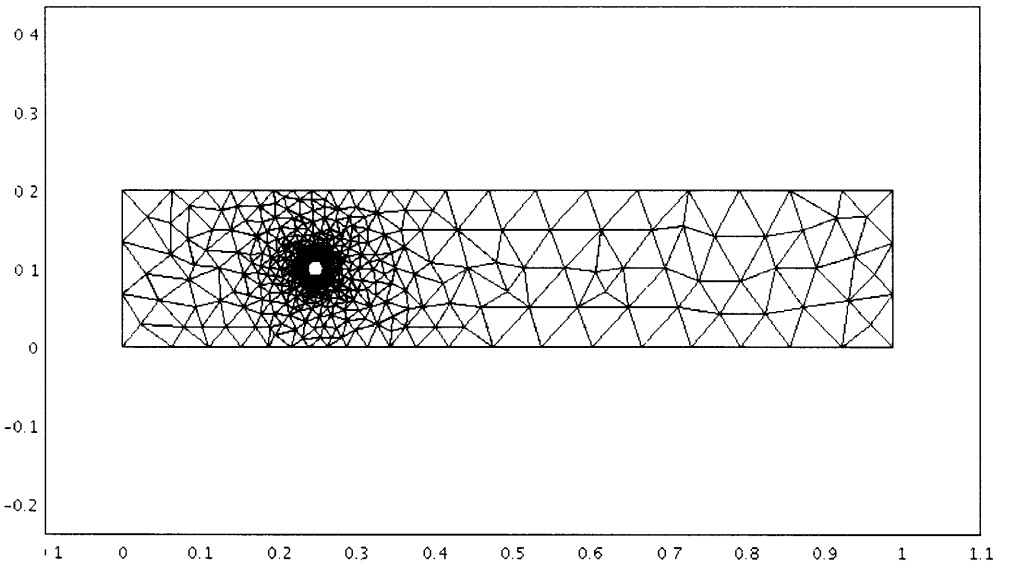

Figure 4.2: Mesh used to solve the CFD problem with FEMLAB.

4.2. Vortices

Vorticity gives a measure of the local rotation or spin of fluid elements. Vortices can exist in the wake of particles falling through fluid and are shown in Figure 4.3.

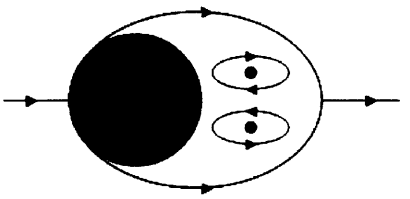

Figure 4.3: Vortices in the wake of a sphere moving through fluid. The large circle is the moving sphere and the two smaller circles are origins of vortices. The curves represent streamlines of the velocity field flowing in the direction of the arrows.

Vortices only form when Re is sufficiently high (Re > 1). Vorticity is defined as [12]:

$$\underset{\sim}{\omega} = \nabla \times \underset{\sim}{u}, \tag{4.1}$$

where u is the velocity field. In the two-dimensional modeling of an alga cell falling through a cylinder of fluid, the vorticity is given by:

$$\underset{\sim}{\omega} = \left(\frac{\partial u_y}{\partial x} - \frac{\partial u_x}{\partial y} \right) \hat{z}. \tag{4.2}$$

Using Euler's equation (the equation of motion for an ideal fluid), one can derive two important relationships about vorticity. Here, an ideal fluid is defined as a fluid having the following properties:

1. Incompressibility: no element of fluid changes volume as it moves.
2. Density is constant.
3. Inviscid: viscosity is zero or negligible. The force exerted across a geometrical surface element $\boldsymbol{n}\delta S$ is $p\boldsymbol{n}\delta S$, where p is pressure, $\boldsymbol{n}$ is the normal vector to the surface, and δS is the differential surface element.

Euler's equation is given by:

$$\frac{\partial \underset{\sim}{u}}{\partial t} + \underset{\sim}{\omega} \times \underset{\sim}{u} = -\nabla H, \tag{4.3}$$

where H is the energy per mass term. In a two-dimensional flow of an ideal fluid subjected to a conservative body force, one can derive the following two relations, which have significant importance to the application of vorticity [12]:

$$\begin{matrix} \frac{D\vec{\omega}}{Dt} = 0 \\ (\vec{u} \cdot \nabla)\vec{\omega} = 0 \end{matrix} \quad . \tag{4.4}$$

The first relationship indicates that $\vec{\omega}$ for each individual fluid element is conserved over time. This is important because once the boundary value problem has reached a steady-state solution vortices will persist. The second relationship indicates that $\vec{\omega}$ is constant along a streamline (i.e., a curve having he same direction as $\vec{u}(\vec{x},t)$ at each point). This is important because a vortex originates and develops outwardly from a certain point. Moreover, the streamlines around a vortex's origin are closed and have a constant velocity. As one approaches the vortex's origin, the velocity goes to zero. To arrive at these two relationships, it was assumed that the fluid was inviscid.

Vortices have a significant application to bacterial locomotion because as bacteria chemotaxis towards a falling alga they may be drawn to the alga's wake. In some cases, the bacteria may enter the vortices which have formed behind the falling alga cell and recirculate around the closed velocity streamlines, collecting nutrients. This is beneficial for bacteria because they are exposed to very high nutrient concentrations while being shielded from the outside flow (u_x) that would move them away from the alga. Once inside a vortex, bacteria don't need to expend energy by having to move with the falling alga, but rather can move passively with the alga cell.

There are three conditions that apply to bacteria entering vortices provided that bacteria do not alter their swimming speed or chemotactic mechanism:

1. Chemosensory ability has to be sufficiently high so bacteria may quickly respond to the attractant gradients. As the alga cell falls, bacteria have roughly a time $t_{cr} = l_{vortex}/w_s$ to become trapped in a vortex, where w_s is the alga fall speed and l_{vortex} is the distance along the alga fall line from the center of the alga to the edge of the vortex farthest from the alga.

2. The bacterial run speed v_0 must be on the order of the alga fall speed w_s ($v_0/w_s \sim 1$).

3. The bacteria run speed v_0 must be on the order of the average velocity inside the vortex v_{vortex} ($v_0/v_{vortex} \sim 1$). Otherwise, the bacteria will cross the velocity streamlines and move out of the vortex region.

4.3. Model

The chemotaxis model that was applied to the analytical moving particle problem was applied to the numerical moving particle problem. However, values for the concentration, velocity, vorticity, and pressure fields were imported from FEMLAB. As before, simulations started bacteria with $dC/dt = dP_b/dt = \overline{dP_b/dt} = 0$ in front of the falling alga at position $(0,y_i)$ and $t = 0$, where y_i is the initial position along the y-axis for the i^{th} bacterium. Simulations followed a large number of bacteria as they moved over the domain of the channel. If bacteria collided with the alga cell (i.e., $r < a$) or channel sides they were reflected. A simulation terminated once a bacterium exited the channel downstream. Each bacterium was simulated for a maximum of 1,000 seconds. A sample of the simulation code is presented in Section A.1.

Two sets of fluid dynamic properties were investigated. Both cases used $\mathrm{Re} = 2aw_s\nu^{-1} = 20$, which is in a regime where vortices form. When Re is held constant, contour plots of the velocity, vorticity, and pressure fields are qualitatively identical. Pe was different between the two cases. The first case had very low viscosity ($\nu = 1\times10^{-6}$ $\mathrm{cm^2\ s^{-1}}$) and investigated aggregate colonization. The second case had a realistic viscosity ($\nu = 1\times10^{-2}$ $\mathrm{cm^2\ s^{-1}}$) and explored vortex recirculation.

Table 4.1 summarizes the parameters used in modeling the numerical moving particle problem and are used throughout this chapter unless otherwise specified. These values were derived from earlier studies [4, 10, 18]. For the low viscosity case, Equations 3.4 and 3.10 related a, C_0, and w_s. For the realistic viscosity case, the following equation defined w_s [18]:

$$w_s = 0.13a^{0.26}.$$

This relation was empirically derived by Kiorboe and Jackson (2001) for large larger particles ($0.02 < a < 1.5$ cm).

parameter	meaning	artificial case	realistic case
a	Alga radius	40×10^{-4} cm	8.000×10^{-4} cm
w_s	Settling velocity	25×10^{-4} cm s^{-1}	1.250×10^{-4} cm s^{-1}
Re	Reynold's number	20	20
Pe	Péclet number	2	20,000
C_0	Substrate concentration at $r = a$	5.0×10^{-9} mol cm^{-3}	4.500×10^{-9} mol cm^{-3}
C_∞	Background substrate concentration	0 mol cm^{-3}	0 mol cm^{-3}
L	Specific molecular leakage rate	0.5 d^{-1}	0.5 d^{-1}
φ	Carbon content of an alga	0.569×10^{-9} mol $cell^{-1}$	10.000×10^{-9} mol $cell^{-1}$
α	Chemosensory constant of bacterial system	660 s	660 s
T_m	Time constant of bacterial system	1 s	1 s
b	Curve fitting constant for alga	1.67×10^{-4} mol $cm^{-2.28}$	1.67×10^{-4} mol $cm^{-2.28}$
D	Substrate diffusivity	10^{-5} cm^2 s^{-1}	10^{-5} cm^2 s^{-1}
K_D	Chemosensory binding constant	1×10^{-7} mol cm^{-3}	1×10^{-7} mol cm^{-3}
τ_0	Unbiased mean run time	0.67 s	0.67 s
v_0	Bacterial swimming speed	12.5×10^{-4} cm s^{-1}	200×10^{-4} cm s^{-1}
Δt	Numerical time step	0.05 s	0.05 s
f_p	Fall velocity scale parameter	2	2
g	Gravitational acceleration	980 cm s^{-2}	980 cm s^{-2}
ν	Kinematic viscosity	1×10^{-6} cm^2 s^{-1}	0.01 cm^2 s^{-1}
ρ_0	Fluid density	1.03 g cm^{-3}	1.03 g cm^{-3}
ξ	Curve fitting constant for w_s	1.24	1.24
$l_{cylinder}$	Length of fluid cylinder	0.40 cm	80 cm
$w_{cylinder}$	Width of fluid cylinder	0.08 cm	16 cm
$R_{cylinder}$	Ratio: aggregate diameter to cylinder width	0.1	0.1
$x_{aggregate}$	Position of aggregate along x-axis	0.10cm	20 cm
$y_{aggregate}$	Position of aggregate along y-axis	0.04 cm	8 cm
v_{vortex}	Average speed in vortex forming region	1×10^{-4} cm s^{-1}	60×10^{-4} cm s^{-1}

Table 4.1: Parameters for the numerical moving particle problem (Values taken from References 4, 10, 18)

4.4. Low Viscosity Case – Aggregate Colonization

The main difference between the low viscosity case used in the numerical moving particle problem and the high Pe case used in the analytic moving particle problem is the kinematic viscosity. In the low viscosity case $\nu = 1\times10^{-6}$ cm^2 s^{-1}, while in the high Pe case $\nu = 1\times10^{-2}$ cm^2 s^{-1}.

Figures 4.4 – 4.7 show contour curves of physical properties associated with fluid flowing through a channel. Notice that all non-closed contour curves terminate at right-angles to the edges of the cylinder and results from the cylinder's boundary conditions:

1. Fluid cannot penetrate the boundary. Therefore, the component of velocity normal to the solid boundary vanishes.

2. The tangential component of the velocity vanishes at the solid boundary (i.e., no-slip boundary condition).

Figure 4.4 diagrams contour curves of the velocity field. The velocity field is highest where the fluid flows tangent to the surface of the moving particle but goes to zero as one approaches the particle. Also, the velocity is low in the region directly behind the aggregate (i.e., the wake).

Streamlines of the velocity field are drawn in Figure 4.5. Far from the aggregate, the velocity streamlines are parallel to the length of the cylinder. Notice that the streamlines around the vortices are closed, as was indicated in Figure 4.3. The existence of closed streamlines around vortices is a good indication that an accurate solution was reached for the concentration, velocity, vorticity, and pressure fields using FEMLAB. If a bacterium were to move passively around one of these vortex streamlines, the bacterium would recirculate indefinitely in a high concentration region.

Figure 4.6 shows the vorticity field. The regions with a high magnitude of vorticity are in front of the falling aggregate where the fluid moves parallel to the aggregate surface. From this diagram one understands that vortices will form in regions where velocity is low, which is directly behind the falling aggregate.

The pressure field is illustrated in Figure 4.7. Notice that the contour curves of the pressure field are tangent to the contour curves of the velocity field. This result occurs because the pressure field is related to the velocity field through the Navier-Stokes equation. The region of highest pressure occurs directly in front of the falling aggregate because fluid hits the aggregate head-on. In this high pressure region, fluid and bacteria moving with the fluid are directed away and to the sides of the aggregate. However, there are also regions of low pressure behind the aggregate. Fluid and bacteria moving with the fluid are able to flow into these low pressure regions. Since these low pressure regions exist in the vicinity of the vortices, it is possible for bacteria to flow into the vortices and become trapped.

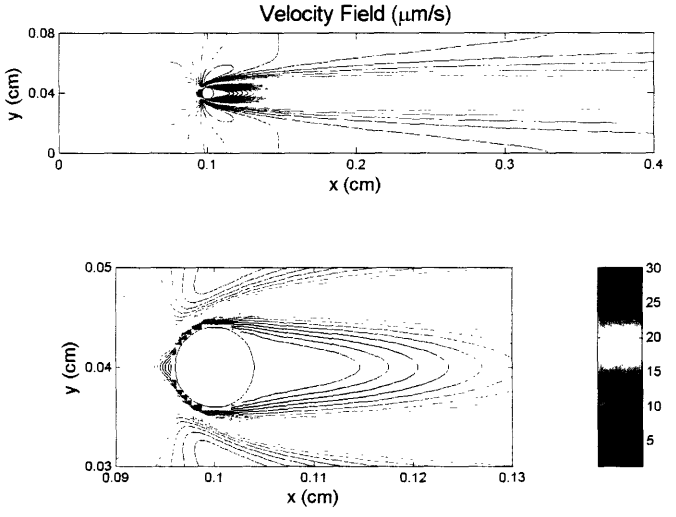

Figure 4.4: Velocity field (Re = 20, Pe = 2).

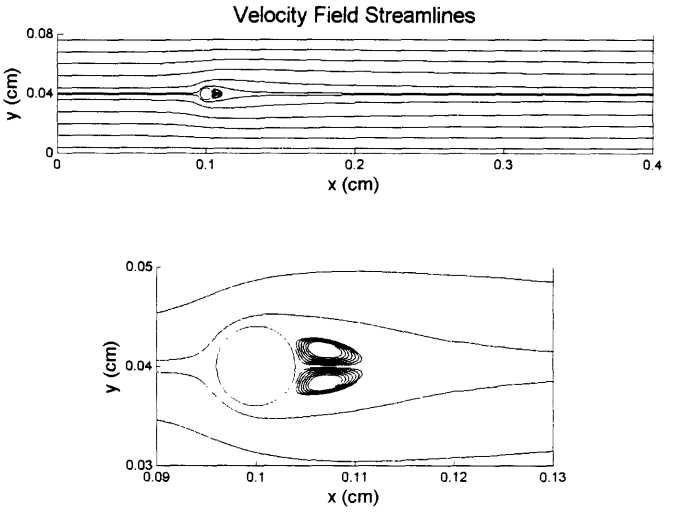

Figure 4.5: Streamlines of the velocity field (Re = 20, Pe = 2).

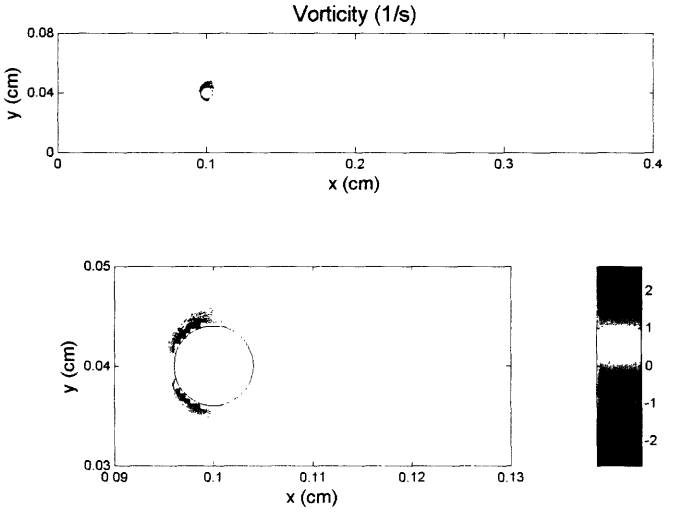

Figure 4.6: Vorticity Field (Re = 20, Pe = 2).

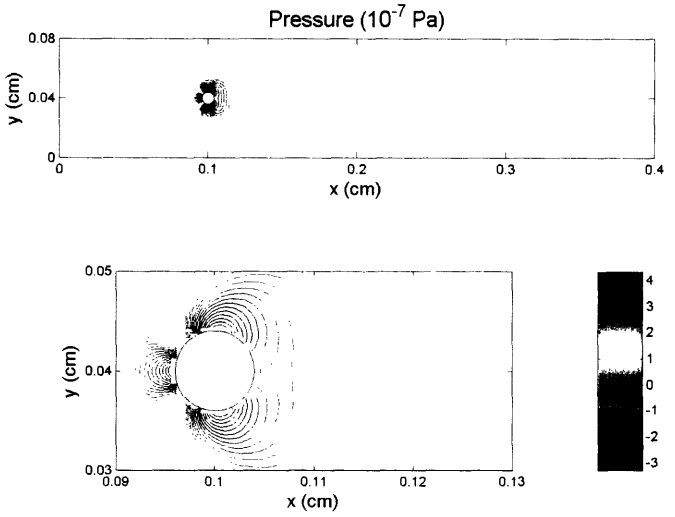

Figure 4.7: Pressure field (Re = 20, Pe = 2).

Figures 4.8 and 4.9 show surface and contour plots of the concentration field. The highest concentration is near the aggregate. However, there is a large concentration plume in the aggregate's wake. This large plume is easily distinguished when examining the contour curves of the concentration field. Often, bacteria cannot swim against the flow of fluid. In such cases, bacteria cannot maintain position near the falling aggregate. Therefore, bacteria may try to seek out regions of low fluid velocity and high concentration, such as in the region directly behind the aggregate and in the aggregate's wake.

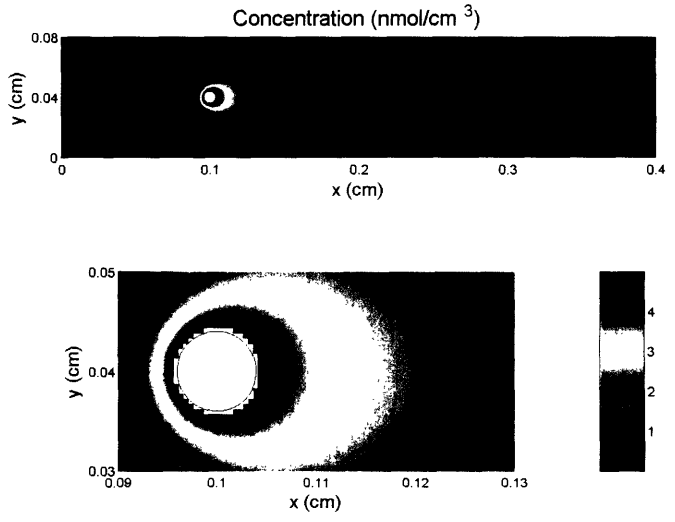

Figure 4.8: Surface plot of the concentration field (Re = 20, Pe = 2).

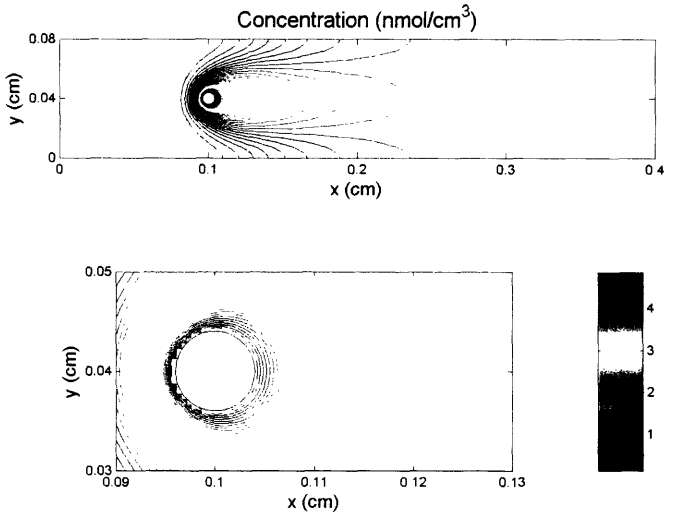

Figure 4.9: Contour plot of the concentration field (Re = 20, Pe = 2).

Now that the physical properties are known for Re = 20 and Pe = 2, one can investigate bacterial chemotaxis arising from this particular fluid flow. As was done with the analytical treatment of the moving particle problem, three cases of inert, non-sensing, and sensing bacteria are examined. However, the analysis of the low and realistic viscosity cases is meant to elucidate qualitative effects associated with chemotaxis and vortices.

Figure 4.10 shows inert bacteria traveling passively with the fluid. There are no bacteria circulating in the vortices. The lack of inert bacteria in vortices results from vortices having closed velocity streamlines. Unless bacteria move with a velocity that has a component normal to the streamlines shown in Figure 4.5, no bacteria can enter the vortex region.

The case of non-sensing bacteria is illustrated in Figure 4.11. Here, bacteria move actively but without directional bias. Therefore, bacteria can move across streamlines of the velocity field. However, they will not remain in the low velocity region because they are not inclined to find higher concentrations. Moreover, even if the non-sensing bacteria happen to move directly through a vortex they will not remain in the region because the bacterial swimming speed is much greater than the speed in the vortices.

Figure 4.12 demonstrates sensing bacteria, which is dramatically different than the cases for inert and non-sensing bacteria. Sensing bacteria are able to actively seek-out high concentration regions in the wake and cluster behind the falling aggregate. The dense clustering near the aggregate is more related to colonization rather than the aforementioned vortex recirculation. This clustering can be understood by examining the ratios of the bacterial swimming speed v_0 to the aggregate sinking speed w_s and to the average speed in the vortices v_{vortex}. Since $v_0/w_s = 0.5$, bacteria are able to overcome the fluid flow and move across the velocity streamlines, arriving in the vortex region.

However, since the bacteria swim much faster than the speed of the vortices (v_0/v_{vortex} = 12.5), the vortices are not strong enough to hold the bacteria for recirculation. Instead the bacteria move next to the aggregate and remain near the aggregate surface because of continuously swimming in a low fluid velocity region and being attracted to high concentration gradients. Such a scenario is most likely characterized by aggregate colonization in nature.

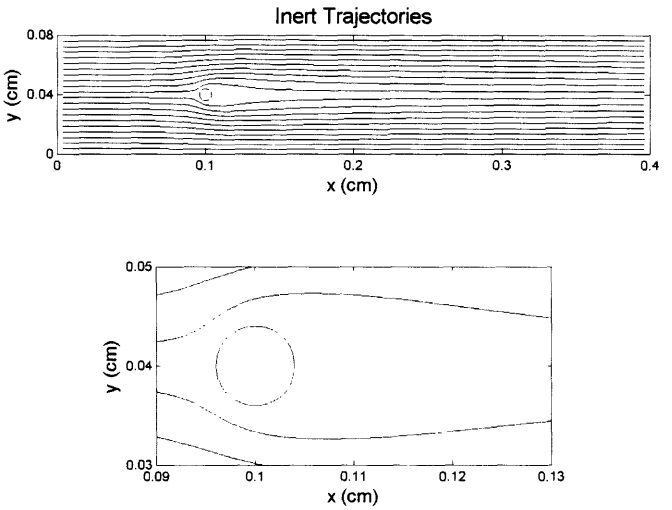

Figure 4.10: Inert bacteria trajectories for the low viscosity case (Re = 20, Pe = 2). The trajectories follow the velocity streamlines.

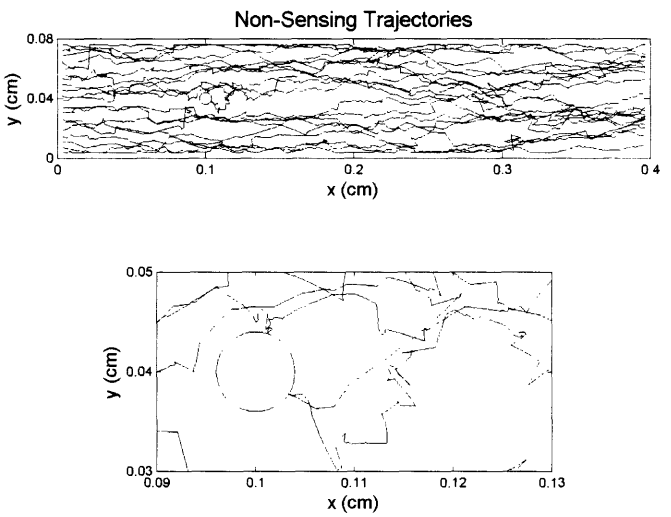

Figure 4.11: Non-sensing bacteria trajectories for the low viscosity case (Re = 20, Pe = 2). Even if the non-sensing bacteria happen to move directly through a vortex they will not remain in the region because the bacterial swimming speed is much greater than the speed in the vortices.

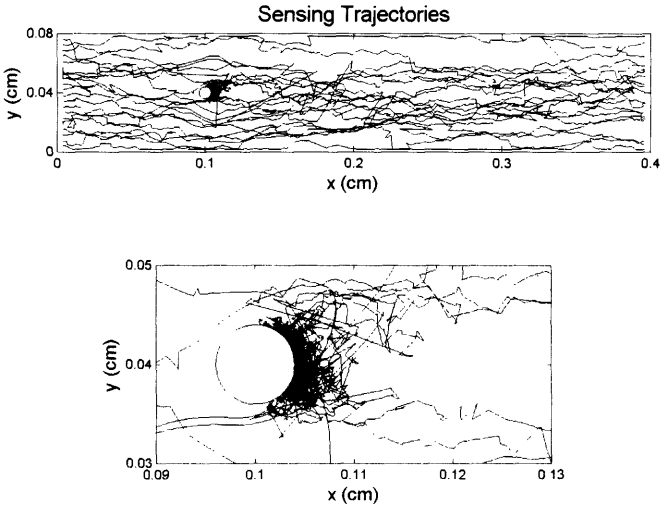

Figure 4.12: Sensing bacteria trajectories for the low viscosity case (Re = 20, Pe = 2). There is dense clustering of bacteria near the aggregate (here called aggregate colonization).

The effect of aggregate colonization is better understood when one considers the time during which bacteria are exposed to the highest nutrient concentration. This exposure time is the time it takes a bacterium to travel from $x = 0$ to $x = 50a$. For the low viscosity case (Re = 20, Pe = 2, and $a = 40\times10^{-4}$ cm), the exposure time is calculated from $x = 0$ to $x = 0.2$ cm. Figure 4.13 shows histograms of exposure times for 2,000 bacteria, where groups of 20 bacteria were started at 100 equally spaced positions along the y-axis of the channel at $x = 0$ and ran for a maximum of 1,000 s. For the inert case, there is little deviation in the exposure time. The largest peak is centered on an exposure time of 80 s, which is equal to $50a/w_s$ and results from fluid flow far away from the aggregate surface. The small deviation toward larger exposure times results from the fluid slowing down near the aggregate surface. For the non-sensing case, the exposure times are more spread out but the largest peak is still centered on an exposure time of 80 s. The larger spread in exposure time not only results from fluid flowing more slowly near the aggregate surface but also a random component of velocity which causes bacteria to move with, against, and tangent to the fluid flow. The third panel in Figure 4.13 shows a histogram of exposure times for sensing bacteria. Similar to the non-sensing case, there is a clustering of exposure times centered on 80 s. Additionally, there is a large peak in exposure time centered on 1,000 s. Between these large peaks at 80 and 1,000 s, there are many small counts. The large peak at 1,000 s results from the effect of aggregate colonization. The presence of small peaks between 80 and 1,000 s means that some bacteria approached the aggregate surface but did not reside in the low fluid velocity, high concentration region for a prolonged time. Because of the low number of counts, such occurrences are unlikely. Since the peak at 1,000 s is large and solitary, whenever bacteria get close to the aggregate surface it is very likely that bacteria will colonize and reside near the aggregate surface for a significant duration.

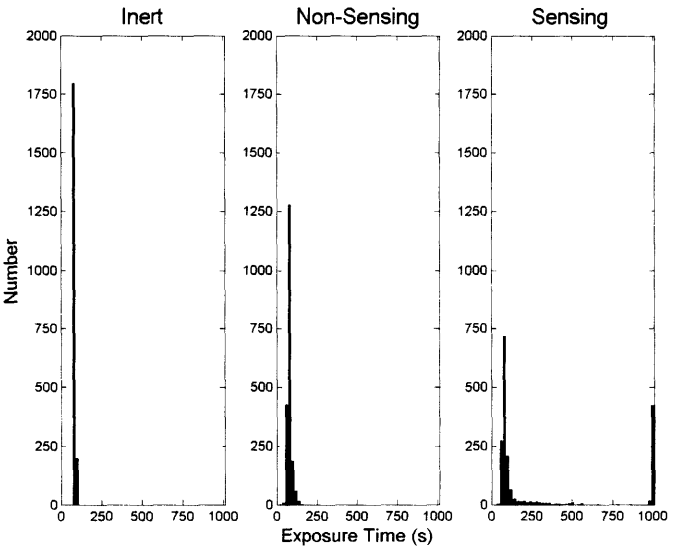

Figure 4.13: Histograms of exposure times for inert, non-sensing, and sensing bacteria for the low viscosity case (Re = 20, Pe = 2). In each histogram 2,000 bacteria in groups of 20 bacteria started at 100 equally spaced positions along the y-axis of the fluid cylinder at $x = 0$ ran for a maximum of 1,000 s.

Figure 4.14 illustrates the differences of starting 20 bacteria upstream at $x = 0$ ("Outside" the vortex region) and starting 4 bacteria downstream, behind the falling aggregate at $x =$ 0.105 cm ("Inside" the vortex region). Initial values for y were defined by an equally spaced number of intervals between $\pm$ 1.25a around the aggregate fall line. Trajectories for inert bacteria are shown in the first two panels. One sees that when bacteria are started outside the vortex region, they are unable to move across the velocity streamlines and approach the rear of the aggregate. However, when inert bacteria are started inside the vortex region, only the bacteria closest to the aggregate fall line enter into recirculation around a vortex, while inert bacteria initiated at y = 0.036 and 0.044 are just on the periphery for undergoing recirculation. Trajectories for non-sensing bacteria are shown in the second two panels. When bacteria are started inside or outside the vortex region, some bacteria reside briefly in the low velocity, high concentration region directly behind the aggregate because the low pressure draws them inward. However, because their swimming speed is unbiased and greater than the average speed in the vortices, the bacteria quickly move outside this region. In the bottom two panels, trajectories for sensing bacteria are shown. Many bacteria reside in the low velocity, high concentration region directly behind the aggregate. The greatest difference in trajectories is for inert bacteria starting inside or outside the vortex region. There is little difference between the trajectories of non-sensing or sensing bacteria starting inside or outside the vortex region because these bacteria are swimming at a speed approximately equal to the aggregate sinking speed, which is greater than the average speed in the vortices.

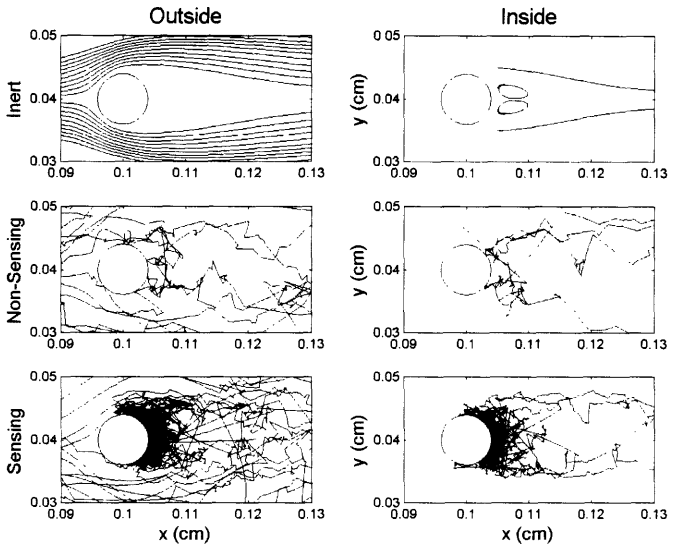

Figure 4.14: Comparison of trajectories for inert, non-sensing, and sensing bacteria starting outside and inside the vortex region for the very low viscosity case (Re = 20, Pe = 2).

4.5. Realistic Viscosity Case – Vortex Recirculation

In the realistic viscosity case, Re = 20 but is achieved with $v = 0.01$ cm^2 s^{-1} (the viscosity of water) and rescaling the dimensions of the problem. Although the choice of v and scaling maintains the Reynolds number, the Péclet number changes dramatically (Pe = 20,000). This realistic viscosity case reflects marine bacteria seeking out nutrient concentrations around a marine snow aggregate. Marine bacteria travel at swimming speeds between 100 – 400 μm s^{-1} [18], which is much faster than *E. coli* (10 – 30 μm s^{-1}) [2, 4]. Marine snow aggregates are organic particles with $a = 0.02 - 1.5$ cm that fall into the deep sea from the sunlit surface waters. Since Re is maintained between the two viscosity cases, the velocity, vorticity, and pressure fields as well as their respective contour and streamline plots are qualitatively identical. However, the scaling of the plots change: length goes from 0.4 to 80 cm, width goes from 0.08 to 16 cm, and values along contours change according to the prescribed velocity and pressure. Since there is a dramatic difference in Pe for the two cases, there is also a dramatic difference in the concentration field, which is evidenced in Figures 4.8 and 4.15. In Figure 4.15, there exists a very long and slender concentration wake, which results from advection dominating diffusion. Additionally, there is a very steep gradient in concentration near the aggregate and wake.

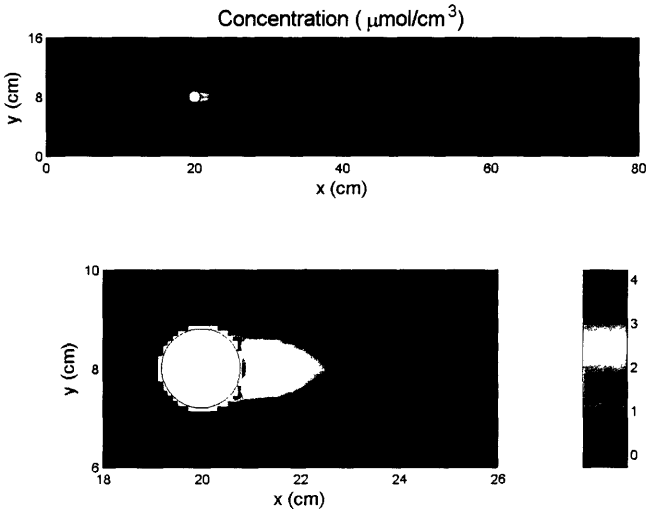

Figure 4.15: Surface plot of the concentration field (Re = 20, Pe = 20,000).

Figure 4.16 shows inert bacterial trajectories, which are qualitatively identical to the trajectories shown in Figure 4.10. Non-sensing bacteria are illustrated in Figure 4.17. Even though non-sensing bacteria can move across streamlines of the velocity field, it seems that the fluid speed might be too high for bacteria to traverse the concentration wake without actively sensing chemical gradients. For this realistic viscosity case $v_0/w_s = 0.16$, which is a factor of 3 smaller than in the low viscosity case. With decreasing v_0/w_s, bacteria are less able to swim against the fluid flow, causing bacterial trajectories to follow streamlines more closely.

Figure 4.18 demonstrates sensing bacteria. Notice that the dense clustering directly behind the aggregate is absent in this realistic viscosity case, which suggests that vortex recirculation may have a greater effect on determining chemotactic response. As mentioned before $v_0/w_s = 0.16$, which means that sensing bacteria have greater difficulty moving across velocity streamlines and into regions of high concentration. However, $v_0/v_{vortex} = 3.33$, which means that the vortices have a stronger effect than they did in the low viscosity case. Therefore, if the bacteria are able to overcome the speed of the fluid and enter the region behind the falling aggregate, then the vortices are strong enough to trap bacteria and allow for some recirculation in a high concentration region.

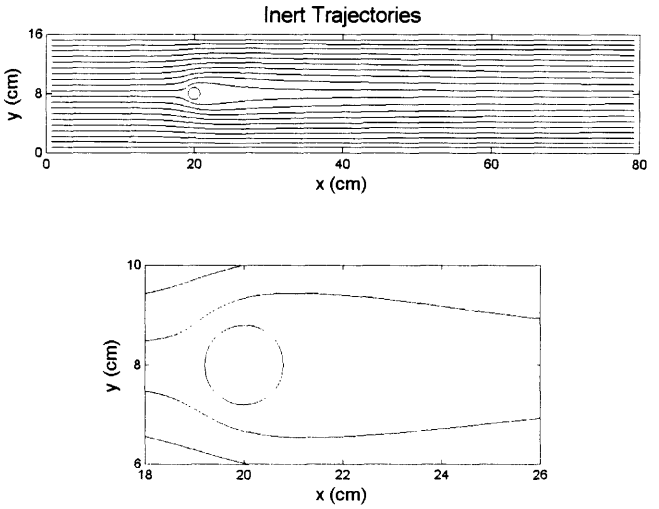

Figure 4.16: Inert bacteria trajectories for the realistic viscosity case (Re = 20, Pe = 20,000). The trajectories follow the velocity streamlines and are identical to the inert bacteria trajectories for the low viscosity case.

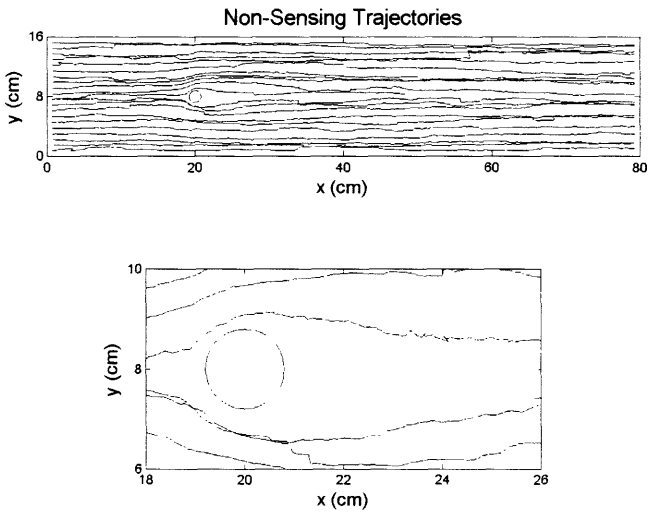

Figure 4.17: Non-sensing bacteria trajectories for the realistic viscosity case (Re = 20, Pe = 20,000).

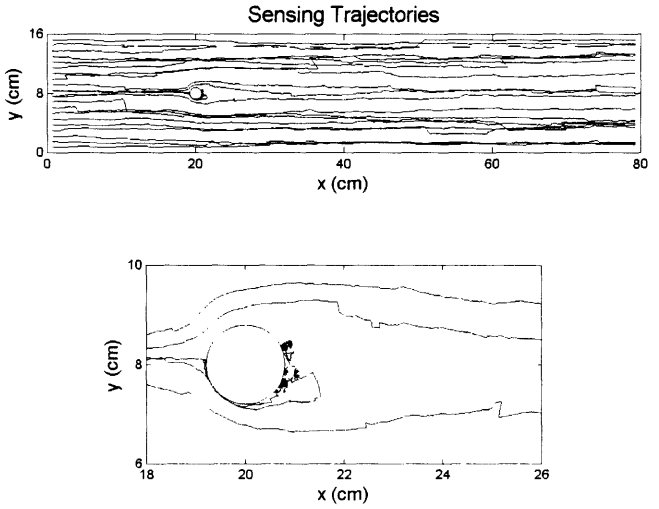

Figure 4.18: Sensing bacteria trajectories for the realistic viscosity case (Re = 20, Pe = 20,000). Dense clustering of bacteria near the aggregate (i.e., colonization) is absent. Vortex recirculation has a stronger effect on chemotaxis causing some trajectories to bend inward.

For the realistic viscosity case (Re = 20, Pe = 20,000, and a = 0.8 cm), the exposure time is calculated from $x = 0$ to $x = 40$ cm. Figure 4.19 shows histograms of exposure times for 2,000 bacteria, where groups of 20 bacteria were started at 100 equally spaced positions along the y-axis of the fluid cylinder at $x = 0$ and ran for a maximum of 1,000 s. As was seen in the low viscosity case, there is little deviation in the exposure time for inert bacteria. The largest peak is centered on an exposure time of 300 s, which is approximately equal to $50a/w_s$ = 320 s and results from fluid flow far away from the aggregate surface. The small deviation toward larger exposure times results from the fluid slowing down near the aggregate surface. For the non-sensing case, the exposure times are more spread out but the largest peak is still centered on an exposure time of 300 s. The larger spread in exposure time not only results from fluid flowing more slowly near the aggregate surface but also a random component of velocity which causes bacteria to move with, against, and tangent to the fluid flow. The third panel in Figure 4.19 shows a histogram of exposure times for sensing bacteria. Similar to the non-sensing case, there is a clustering of exposure times centered on 300 s. Additionally, there is diffuse peak in exposure time near 1,000 s. Between these large peaks at 300 and 1,000 s, there are many small counts. The large peak near 1,000 s results from bacteria entering and remaining in the region behind the sinking aggregate but is more spread out than the peak at 1,000 s seen in Figure 4.13. Dispersion around 1,000 s, results from some vortex recirculation. However, since bacteria swim continuously and the fluid speed in the vortex is on the order of the bacterial swimming speed, bacteria can inadvertently swim out of the vortex region. The presence of small peaks between 300 and 1,000 s means that some bacteria which approached the aggregate surface did not reside in the low fluid velocity, high concentration region for a prolonged time. However, because of the low number of counts, such occurrences are unlikely. Since the peak at 1,000 s is large but dispersed, whenever bacteria get close to the aggregate surface they try to colonize and

reside near the aggregate but cannot since the speed of the sinking aggregate is large. Although the vortices have a more pronounced effect, they are not strong enough to hold the bacteria indefinitely, causing some bacteria to leave the vortex region.

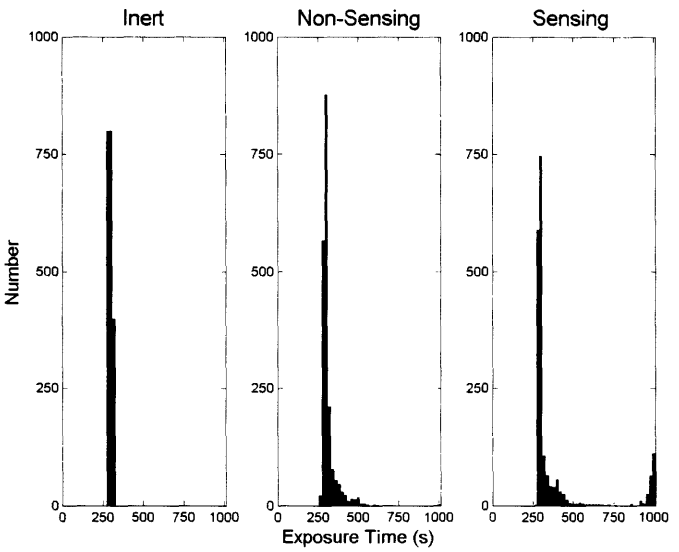

Figure 4.19: Histograms of exposure times for inert, non-sensing, and sensing bacteria for the realistic viscosity case (Re = 20, Pe = 20,000). In each histogram 2,000 bacteria in groups of 20 bacteria started at 100 equally spaced positions along the y-axis of the channel at $x = 0$ ran for a maximum of 1,000 s

As before, Figure 4.20 illustrates the differences of starting 20 bacteria upstream at $x = 0$ ("Outside" the vortex forming region) and starting 4 bacteria downstream, behind the falling aggregate at $x = 21$ cm ("Inside" the vortex forming region). Initial values for y were defined by an equally spaced number of intervals between $\pm$ 1.25a around the aggregate fall line. Trajectories for inert bacteria are shown in the first two panels. One sees that when bacteria are started outside the vortex region, they are unable to move across the velocity streamlines and approach the rear of the aggregate. However, when inert bacteria are started inside the vortex region, only the bacteria closest to the aggregate fall line enter into recirculation around a vortex, while inert bacteria initiated at $y = 7.2$ and 8.8 cm are just on the periphery for undergoing recirculation. In the second two panels, when non-sensing bacteria are started inside or outside the vortex region, some bacteria reside briefly in the low velocity, high concentration region directly behind the aggregate because the low pressure draws them inward. However, because their swimming speed is unbiased, the bacteria move outside of this region. Nevertheless, one notices that some of the trajectories are influenced by the vortices. In the bottom two panels, few sensing bacteria are able to enter and reside in the low velocity, high concentration region directly behind the aggregate. This low number of bacteria results from the fluid flow being too strong for the bacteria to overcome. Again, one can see that some of the trajectories are influenced by the vortices. Vortex recirculation is having an effect. However, some aggregate colonization exists for these chemotaxis parameters and is observed near the aggregate surface. It seems that aggregate colonization is a result of

bacteria continuously swimming at a speed that allows for them to move across velocity streamlines and into the low velocity, high concentration region directly behind the aggregate. Furthermore, it seems that vortex recirculation only occurs when the bacterial swimming speed is on the order of the average speed inside the vortices. However, with such a small swimming speed, bacteria are unable to effectively move across velocity streamlines and into the low velocity, high concentration region directly behind the aggregate. Therefore, for some choice of chemotaxis parameters for continuously swimming bacteria, some vortex recirculation is observed but aggregate colonization is not completely absent from the dynamics. In order to exploit vortex recirculation such that aggregate colonization is absent, bacteria must reduce their swimming speed when inside the vortex region. This altering of swimming speed could be done in relation to concentration gradients such that swimming speed decreases as the concentration gradient increases sharply in the vicinity of the vortex region.

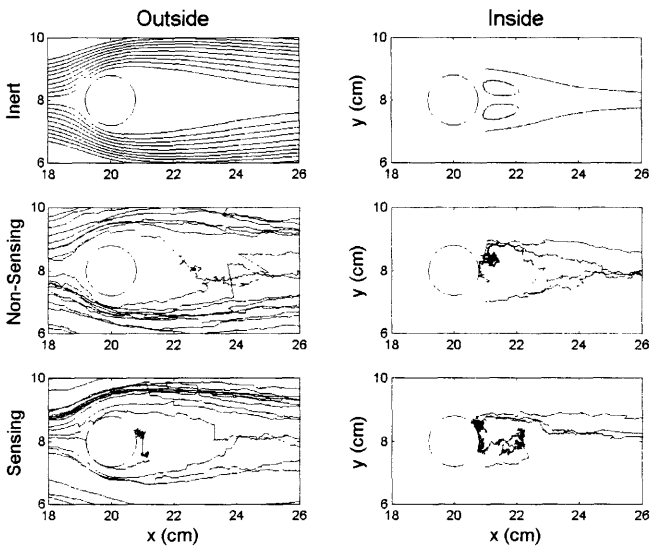

Figure 4.20: Comparison of trajectories for inert, non-sensing, and sensing bacteria starting outside and inside the vortex region for the realistic viscosity case (Re = 20, Pe = 20,000).

4.6. Conclusions

In some cases, bacteria could enter vortices which have formed behind a moving particle and recirculate around closed velocity streamlines, collecting nutrients. Once inside a vortex, bacteria could just move passively as the particle travels to greater aquatic depths. The following conclusions were made when investigating effects associated with chemotaxis and vortices:

1. There are three conditions that apply to bacteria entering vortices provided bacteria do not alter their swimming speed or chemotactic mechanism: chemosensory ability has to be sufficiently high so bacteria may respond to the attractant gradients, $v_0/w_s \sim 1$, and $v_0/v_{vortex} \sim 1$. Additionally,

bacteria must move with a velocity that has a component normal to the velocity streamlines.

2. Colonization was defined as the clustering of bacteria near an aggregate and is characterized by: $v_0/w_s \approx 1$, $v_0/v_{vortex} > 1$, and bacteria continuously swimming and sensing concentration gradients. Under this mechanism, whenever bacteria get close to the aggregate surface they have the opportunity to colonize and reside near the surface for a long time.

3. When $v_0/w_s = 0.16$ and $v_0/v_{vortex} = 3.33$, vortex recirculation effects chemotactic response. However, aggregate colonization is still present. Whenever bacteria get close to the aggregate surface they try colonize and reside near the aggregate but cannot since the speed of the sinking aggregate is large. Although the vortices have a more pronounced effect, they are not strong enough to hold the bacteria indefinitely.

4. Aggregate colonization results from bacteria continuously swimming at a speed that allows for movement across velocity streamlines and into the vortex region (i.e., low velocity, high concentration region directly behind the aggregate). True vortex recirculation only occurs when the bacterial swimming speed is on the order of the average speed inside the vortices. However, with a small swimming speed, bacteria are unable to move across velocity streamlines and into the vortex region. For some chemotactic parameters for continuously swimming bacteria, vortex recirculation is observed. However, the effect of aggregate colonization is not completely absent. To exploit vortex recirculation, bacteria must reduce their swimming speed when inside the vortex region. This altering of swimming speed can be done in relation to concentration gradients, where swimming speed decreases as the concentration gradient increases sharply in the vicinity of the vortex region.

5. The altering of bacterial swimming speed is important in the context of a standing debate in marine microbial ecology about free-living versus attached bacteria. Free-living bacteria are characterized by constantly searching for nutrients, while attached bacteria are characterized by anchoring themselves to a source rich in nutrients. If bacteria can alter their swimming speed in order to obtain a high concentration of nutrients, then it is reasonable to suspect that the chemotactic mechanism between free-living and attached bacteria is different. For free-living bacteria, swimming speed might be a function of preferred concentration, such that free-living bacteria are stationary when the nutrient concentration is high and motile when the nutrient concentration is low. For attached bacteria, swimming speed might be a function of dC/dt, such that attached bacteria are motile when the gradient changes. To investigate these connections, one must consider the following cases: v_0/w_s and $v_0/v_{vortex} < 1$, ≈ 1, or > 1; swimming speed is constant or a function of C and/or dC/dt.

5. Image Processing and Experimental Results

5.1. Motivation

Although theoretical models and simulations give understanding about cellular dynamics, it is essential that experiments be carried out to obtain biological parameters, identify mechanisms and their ecological consequences, and motivate further theoretical investigation. With the use of a high speed camera, one is able to record images of bacterial locomotion and associated fluid mechanics.

5.2. Image Processing (BacTrack^{TM})

BacTrack^{TM} is a Java program written by Scott Stransky and developed by Michael Sekora and Roman Stocker. BacTrack^{TM} was designed to locate and track the motion of numerous bacteria recorded over many exposures. There are two phases associated with image processing. First, the positions of many particles (i.e., bacteria) are located over a number of frames. Second, trajectories are drawn between particle positions. The BacTrack^{TM} user interface is shown in Figure 5.1.

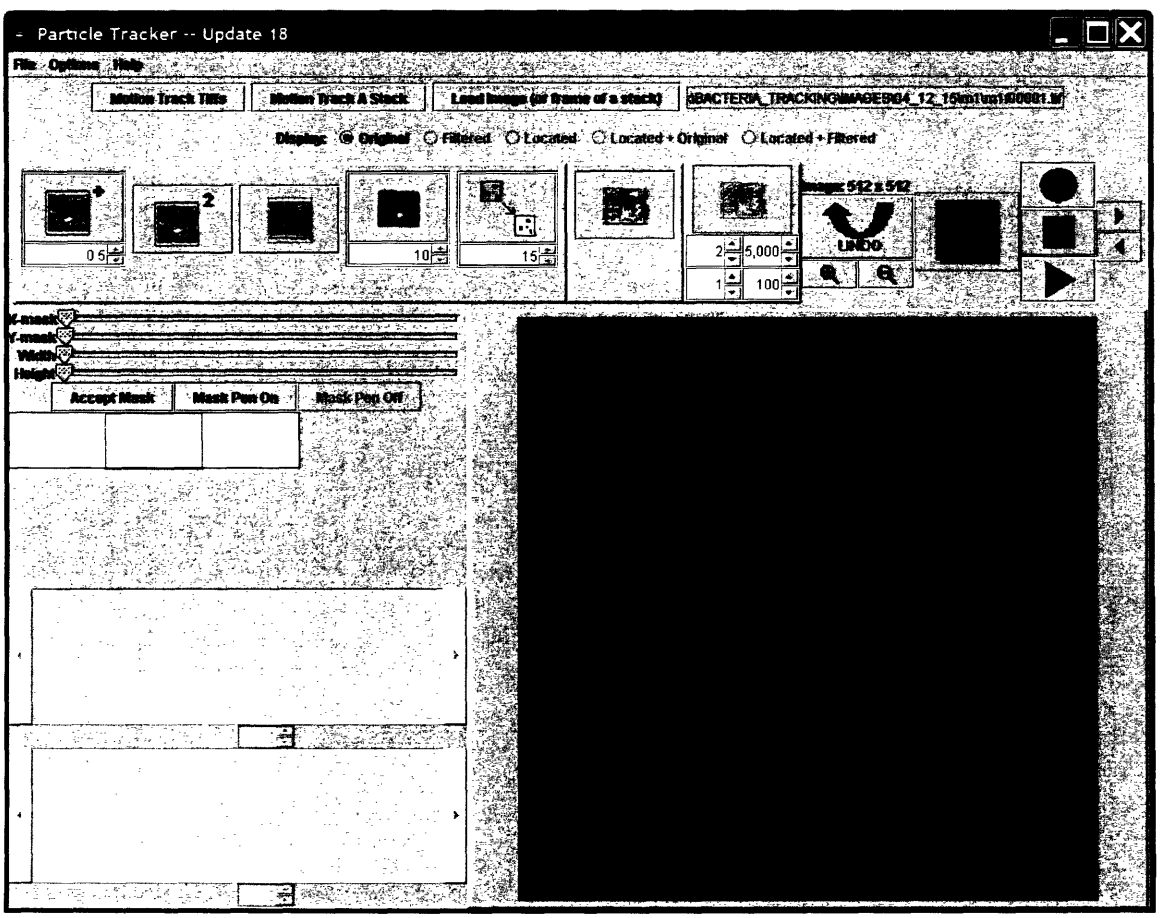

Figure 5.1: BacTrack^{TM} user interface.

5.2.1. Locating Particles

There are three essential image processing tools one should consider when analyzing images of bacterial motion:

1. Mask out certain regions in images
2. Remove background
3. Define threshold bounds

These tools were listed in the order that BacTrack™ applies them to image processing. Masking out certain regions of an image or images is an important tool especially when some physical obstruction exists in the image such as walls of a fluid channel, large nutrient aggregate, or some microfluidic device. With this tool, one is able to select preferred regions to analyze.

Removing the image background counteracts the effects of non-uniform illumination of an object. Even with delicate alignment, the intensity from a light source falls off as a Gaussian from the center of the source, thus causing particles at the edges of an image to be obscured. Therefore, creating a uniform background significantly affects how well particles are located. The Rolling Ball algorithm was employed for background removal and works as a geometric shape filter. The ball represents a structuring shape that does not fit into the geometric shape (i.e., mold) but fits well into the background clutter. The diameter of the ball is equal to the length of the largest particle that is not part of the image background. The algorithm works in the following manner. Initially, one has some measurement of intensity at points (x, y). Figure 5.2a represents a one-dimensional analogue of a two-dimensional slice perpendicular to the (x, y) plane. If one rolls the ball underneath the surface of the signal in such a way that the ball is always in contact with the undersurface of the signal, then one produces the dashed line shown in Figure 5.2b, which is the set of points consisting of all possible locations of the center of the ball. Now, if one rolls the ball such that its center is always in contact with the dashed line shown in Figure 5.2b, then the top of the ball generates the solid line shown in Figure 5.2c. The background is removed by subtracting the solid line shown in Figure 5.2c from the initial signal shown in Figure 5.2a, which results in the processed signal shown in Figure 5.2d [19]. Figure 5.3 shows an image before and after the rolling ball algorithm was applied.

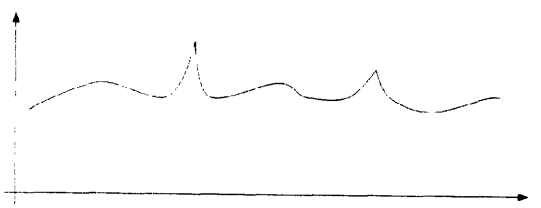

Figure 5.2a: Surface of initial signal. (Taken from Reference 19).

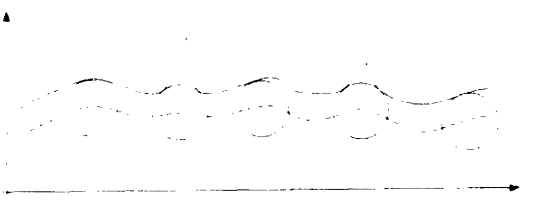

Figure 5.2b: Surface generated by the center of the rolling ball. (Taken from Reference 19).

Figure 5.2c: Surface generated by the top of the rolling ball. (Taken from Reference 19).

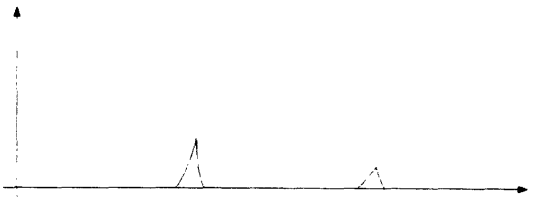

Figure 5.2d: Processed signal. (Taken from Reference 19).

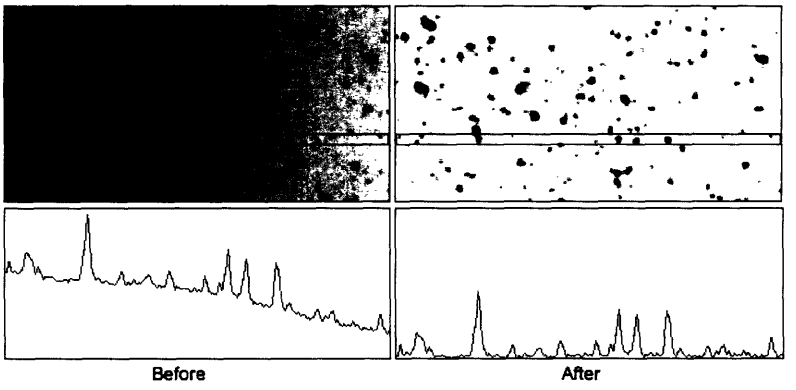

Figure 5.3: Image before and after the rolling ball algorithm was applied. Notice that there is a gradient in the image intensity before it was processed. (Taken from Reference 20).

After the background has been removed from the images, intensity thresholds are applied. Here, the intensity must be between minimum and maximum grayscale values, where intensity is related to the grayscale volume of an image. Figure 5.4 shows Figure 5.2d with threshold bounds being applied. Under this analysis, the original signal shown in Figure 5.2 would produce two particles with diameters equal to the length of the dashed line (i.e., lower threshold) intersected by the signal's surface.

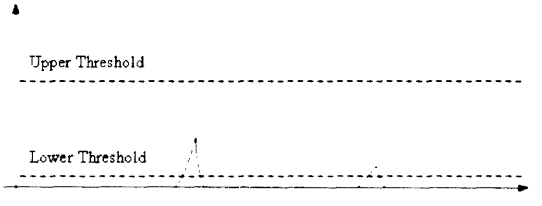

Figure 5.4: Thresholds being applied to processed signal. The diameters of the processed particles will equal the length of the dashed line (i.e., lower threshold) intersected by the processed signal's surface. (Adapted from Figure 5.2d).

There are many other features in BacTrack™ that one can apply to adjust the quality of images, such as:

- Increasing the contrast
- Squaring the intensity of each pixel
- Inverting the grayscale color scheme
- Blurring pixels (i.e., taking an average over pixels in a region and giving them an average intensity)
- Restricting particle location to only those particles that have major and minor axes between defined minimum and maximum values
- Zoom in, zoom out

5.2.2. Tracking Particles

BacTrack™ is able to save data pertaining to the particle positions found for each given frame. This data is useful if one wants to plot bacterial distributions in space but does not care about the motion of each particle. Additionally, BacTrack™ can draw trajectories between positions of the same particle over many exposures.

First, a tracking algorithm is chosen from the six available in BacTrack™:

1. Particle-by-particle, no velocity
2. All particles at once, no velocity
3. Conservative search radius
4. Biased search radius
   A. Closest
   B. Velocity
   C. Dual (closest and velocity)

These different algorithms are explained in Section A.2.

5.3. Image Analysis

After processing frames with BacTrack™, image analysis is done in MATLAB. A sample of the image analysis code is presented in Section A.3. Before graphical plots can be generated, some restrictions on the data must be applied so that accurate information can be extracted. Prior to the analysis, all bacteria trajectories are indexed. Those trajectories not meeting one or more of the restrictions are taken out of consideration. The steps taken during image analysis are outlined below:

1. Load data for analysis and define image analysis parameters.
2. Restrict based on trajectory duration. Keep trajectories that have been tracked for a minimum amount of time.

3. Restrict based on the maximum distance traveled over an entire trajectory. Disregard particles that are stationary or move little over time.
4. Apply the median filter algorithm to smooth out trajectories.
5. Compute the *SPD* and *RCD* (defined below) for the following three cases: each frame of each trajectory, averaged over each trajectory, and averaged over all trajectories (i.e., global average).
6. Restrict based on the mean speed of the particles. Disregard particles that do not have an *SPD* (averaged over entire trajectory) that is above a minimum value.
7. Compute the bacterial run speed v_0 as the mean of the top p percent of *SPD*'s (averaged over entire trajectory). Typically, $p = 10$ [21].
8. Detect when tumbles occur during each trajectory.
9. Apply tumble sharpening filter to differentiate between the run and tumble modes of a trajectory.
10. Compute tumble frequency and duration.
11. Restrict based on tumble frequency. Discard the top and bottom q percent of tumble frequencies. Typically, q equals 10 [21].
12. Compute global average tumble frequency and duration.

The median filter algorithm is applied by taking the first n elements starting with the first indexed value of a trajectory's x and y components and sectioning them into a group. n is the number of median filter points and typically equals the average time it takes for a bacterium to tumble divided by the frame rate of the high speed camera. The n elements are sorted into increasing order and the first and last elements of the sorted set are disregarded. The resulting n-2 elements are averaged to obtain a central x and y value over the first n elements. The process continues with the next set of n elements starting with the second indexed value of a trajectory's x and y components.

The *SPD* is the absolute instantaneous linear speed and is obtained from the frame-to-frame displacement. The *RCD* is the absolute angular change in direction of a trajectory from frame to frame. A pictorial definition of how the *RCD* is computed is shown in Figure 5.4. The *SPD* and *RCD* for frame j in trajectory i is calculated by:

$$SPD_{i,j} = \left(\left(x_{i,j+1} - x_j\right)^2 + \left(y_{i,j+1} - y_{i,j}\right)^2\right)^{1/2}$$
$$RCD_{i,j} = \left|\mathrm{mod}\left(\left|\tan^{-1}\left(\frac{y_{i,j+2} - y_{i,j+1}}{x_{i,j+2} - x_{i,j+1}}\right) - \tan^{-1}\left(\frac{y_{i,j+1} - y_{i,j}}{x_{i,j+1} - x_{i,j}}\right)\right| + \pi, 2\pi\right) - \pi\right|. \quad 5.1$$

The average *SPD* and *RCD* for trajectory i of length N frames is calculated by:

$$\overline{SPD_i} = \frac{1}{N}\sum_j SPD_{i,j}$$
$$\overline{RCD_i} = \frac{1}{N}\sum_j RCD_{i,j} \quad 5.2$$

The global average *SPD* ($\overline{\overline{SPD}}$) and *RCD* ($\overline{\overline{RCD}}$) is computed in a manner similar to the above but instead averaged over all frames of all trajectories. The $\overline{\overline{SPD}}$ is used to restrict trajectories that do not have a $\overline{SPD_i}$ which is above a minimum value. To do so, one disregards trajectories that do not have a $\overline{SPD_i}$ greater than or equal to the $\overline{\overline{SPD}}$ times some factor $f_{mean\ speed}$ ($0 \leq f_{mean\ speed} \leq 1$). Typically, $f_{mean\ speed} = 0.75$ for *E. coli* [21].

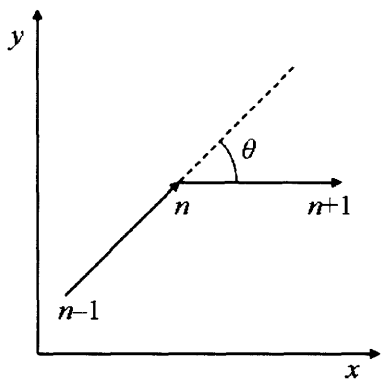

Figure 5.4: Pictorial definition of how the *RCD* is calculated by determining the angular change of direction between frames. Three positions for a bacterium are shown at n-1, n, and n+1.

Tumbles were detected by noting the points in particle *i*'s trajectory where the following two conditions were satisfied.

1. $SPD_{i,j} \leq f_{tumble\ SPD}\ v_0$
2. $RCD_{i,j} \geq f_{tumble\ RCD}$

Here, $0 \leq f_{tumble\ SPD} \leq 1$ is a factor applied to the bacterial swimming speed v_0, which was determined in Step 7 and $f_{tumble\ RCD}$ is the minimum number of degrees per second that a bacterium can tumble for that part of its trajectory to be considered in tumble mode. When a tumble is detected a 1 is placed at position *j* along trajectory *i*. However, if no tumble is detected, then a 0 is placed at the same position. These gives a binary sequence of run and tumble modes (e.g., 10100001001000010001).

During a trajectory, a particle may go into or out of focus such that the entire body of the particle may not be fully illuminated. Such distortion may cause some inaccuracy in the run-tumble sequence. However, applying the tumble sharpening filter allows one to differentiate more easily between the run and tumble modes of a trajectory. In Step 1, the tumble sharpening parameter is defined, which is the maximum length of time (or the number of frames during this length of time) overlooked when defining run and tumble modes. The tumble sharpening filter works in the following manner: if 1 represents a tumble, 0 represents a run, and 10100001001000010001 is the run-tumble sequence; then if the tumble sharpening parameter equals 3 frames, the run-tumble sequence becomes 11100001111000011111. Note, the tumble sharpening parameter should be less than the

average time it takes a bacterium to undergo one tumble. Essentially, the tumble sharpening filter looks for a set of elements, where one is the first and last element. If the number of elements between the first and last element is less than or equal to the number of elements defined by the tumble sharpening parameter, then those in-between elements are defined as tumbles.

The tumble duration (i.e., the time it takes for a bacterium to undertake one tumble) and tumble frequency (i.e., the number of tumbles per second) are computed by analyzing run-tumble sequences. Since tumbles in a run-tumble sequence are designated by 1's, the duration of one tumble is simply the number of frames over which tumbles occur multiplied by the number of frames per second recorded by the camera. The average tumble duration over an entire trajectory is calculated by averaging the durations for individual tumbles for a particular particle. The tumble frequency can only be defined for an entire trajectory, not for individual tumbles. The tumble frequency is found by summing the number of contiguous segments of tumbles occurring throughout a trajectory and dividing this sum by the length of the trajectory. Trajectories are further restricted by discarding the top and bottom q percent of tumble frequencies. Typically, q equals 10 [21]. Lastly, the global average tumble frequency and global average tumble duration are computed by averaging all resulting trajectories.

5.4. Experimental Results

A significant amount of effort was put into developing BacTrack™ and MATLAB image analysis software. In the research leading up to this work, theoretical modeling of chemotaxis has led experiment. It is beyond the scope of this work to experimentally research and make conclusions about bacterial chemotaxis. However, results for *E. coli* bacteria moving in a controlled environment with and without ultraviolet radiation being shined upon them are presented to illustrate the capabilities of BacTrack™ and the MATLAB image analysis software. Moreover, it is experimentally well known that *E. coli* are repelled by ultraviolet radiation. Therefore, the data presented in the following sections are preliminary results for the experimental setup.

5.4.1. Setup

The experimental setup consisted of an optical microscope with a stage for viewing bacteria on a slide. Attached to the microscope was a high speed camera (Phantom V5.1 by Vision, capable of several thousand frames per second). Exposures of 512×512 pixels were taken at 50 frames per second at a magnification of 20× and were saved as 8bit TIFF images. The length scale for exposures was 0.625 μm/pixel and was obtained from a reference magnification of 100× and a reference length scale of 0.125 μm/pixel. For the following results, movies were taken for approximately 80 seconds. In one set of frames, UV radiation was shined upon a specimen of *E. coli* bacteria. The parameters used for processing and analyzing images are given in Table 5.1.

image processing Parameter	value
Remove background parameter	5 pixels
Lower threshold	55
Upper threshold	255
Particle tracking algorithm	Conservative Search Rad.
Search radius	5 pixels
IMAGE ANALYSIS PARAMETER	VALUE
Minimum trajectory duration	2 s
Minimum for maximum distance traveled	7 μm
Number of median filter points	6
Minimum mean speed factor	0.75
p	10 percent
$f_{tumble\ SPD}$	0.3
$f_{tumble\ RCD}$	1,000 deg s^{-1}
Tumble sharpening parameter	0.08 s
q	10 percent

Table 5.1: Parameters used for processing and analyzing images where UV radiation was and was not applied. The image processing parameters were used in BacTrack™ and the image analysis parameters were used in the MATLAB software.

5.4.2. Individual Bacteria Trajectories

In the MATLAB image analysis software, one can investigate properties of individual trajectories. The following plots illustrate three aspects of bacterial trajectories:

1. Run mode, never undergoing a tumble
2. Tumble mode, undergoing one tumble
3. Moving in a circular trajectory while never undergoing a tumble

These plots are important because they experimentally define run and tumble modes. The existence of circular trajectories is a newly discovered biological phenomenon. It is impressive that the experimental setup could detect these circular trajectories. In each plot the following information about a trajectory is displayed: picture of trajectory with respect to the entire exposure area, global properties, zoomed picture of trajectory showing finer details, and *SPD* and *RCD* as functions of time over the duration of the entire trajectory.

Figure 5.5 shows information pertaining to a typical bacterial trajectory defined only by run mode, notice the straight line trajectory. For a bacterium in run mode, the *SPD* and *RCD* stay relatively constant at 15.6 μm s^{-1} and 386.1 deg s^{-1}, respectively. Figure 5.6 shows information pertaining to a bacteria undergoing one tumble over the duration of its trajectory. When the tumble occurs, the *SPD* dips downward while the *RCD* spikes upward and occurs between a time of 1.72 and 1.86 s. Examining the zoomed in picture

of the bacterium's trajectory in Figure 5.6, one notices a very sharp turn about three-fourths of the way along the trajectory, which is where the tumble occurs. Figure 5.7 shows information pertaining to a bacterium moving in a circular trajectory. Such a trajectory occurs with little or no tumbles, just a continuous change in direction. This circular trajectory is a recently understood biological phenomenon, which results from a torque balance between a bacterium's body and flagellum. Since these two parts of a bacterium have a different distance from some planar surface (i.e., the wall of the specimen container), where the bacteria are swimming parallel to this surface, there is a net torque that makes bacteria swim in circles [22].

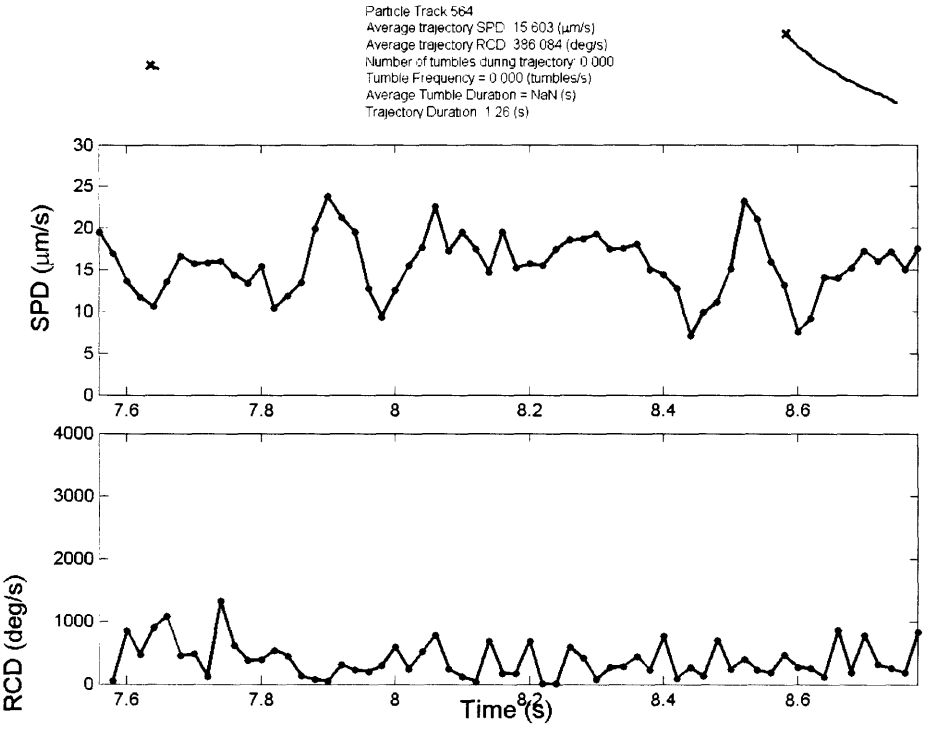

Figure 5.5: *SPD* and *RCD* for a bacterium moving only in run mode. The *SPD* and *RCD* remain relatively constant throughout the trajectory's duration.

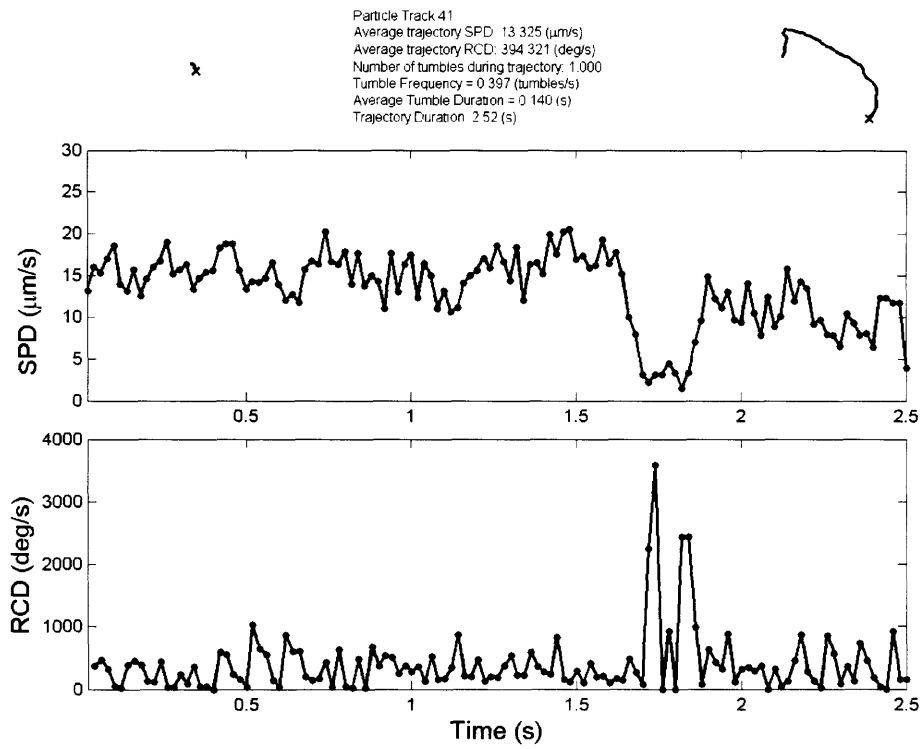

Figure 5.6: *SPD* and *RCD* for a bacterium undergoing one tumble during its trajectory. When the tumble occurs (at $t \sim 1.75$ s), the *SPD* dips downward while the *RCD* spikes upward.

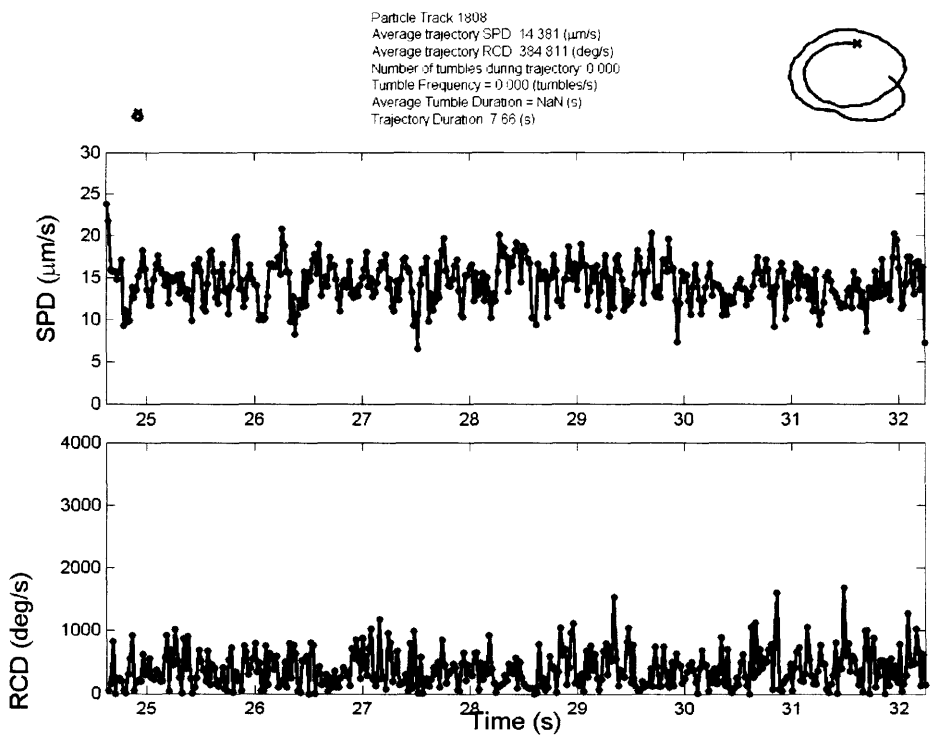

Figure 5.7: *SPD* and *RCD* for a bacterium moving in circular motion during its entire trajectory. The bacterium never undergoes a tumble.

5.4.3. No Ultraviolet Radiation

Figure 5.8 shows bacterial trajectories color-coded to illustrate values for trajectory duration. However, trajectories can also be color-coded according to $\overline{SPD_i}$, $\overline{RCD_i}$, etc. The four colors used distinguish intervals of 25% of the maximum value. The coloring scheme going in increasing order is: Red, Green, Blue, and Black. Color-coding trajectories is useful for determining if there is a pattern in the distribution of bacteria over the viewable surface and allows one to investigate bacterial clustering as a result of trajectory duration, $\overline{SPD_i}$, $\overline{RCD_i}$, etc. Therefore, color-coding bacterial trajectories would be very useful once experiments on sinking aggregates are undertaken. Here, there is no exposure to ultraviolet radiation and bacterial trajectories are equally distributed over the viewable surface. Also, most of the trajectories have short durations (i.e., colored Red).

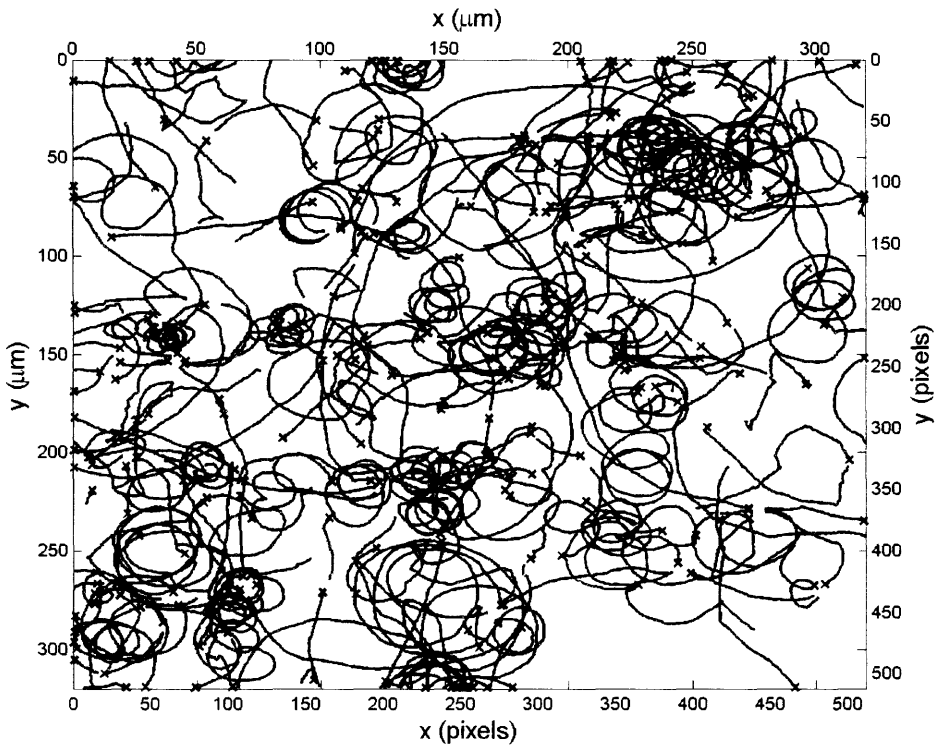

Figure 5.8: Bacterial trajectories color-coded to illustrate values for trajectory duration. There is no exposure to UV radiation and bacterial trajectories are equally distributed over the viewable surface. The four colors used distinguish intervals of 25% of the maximum value. The coloring scheme going in increasing order is: Red, Green, Blue, and Black.

Figures 5.9 – 5.11 show histograms of trajectory duration, $SPD_{i,j}$, and $RCD_{i,j}$. Figure 5.9 shows that the number of trajectories decreases exponentially as the duration increases, where the average duration over all trajectories is 6.81 s and the minimum and maximum durations are 0.92 and 55.1 s, respectively. There are two reasons for why some

trajectories were tracked for such short durations: i) those particles went out of focus by moving vertically with respect to the camera and ii) stopped swimming all together. The first reason is most likely, since much of the image processing was developed to account for trajectories going out of focus. Figure 5.10 shows that values for $SPD_{i,j}$ are normally distributed, where $\overline{\overline{SPD}}$ is 12.0 μm s^{-1} and the minimum and maximum $SPD_{i,j}$ are 0 and 33.9 μm s^{-1}, respectively. The Central Limit Theorem says that for a set of N independent random variables each with an arbitrary probability distribution having a mean and finite variance, the normal form variable has a limiting cumulative distribution function which approaches a normal distribution. Therefore, the swimming speed for *E. coli* in a controlled, stable environment is a random variable. Figure 5.11 shows that the number of counts decreases exponentially as the $RCD_{i,j}$ increases, where $\overline{\overline{RCD}}$ is 479.7 deg s^{-1}. Therefore, small changes in direction are expected from tumbling.

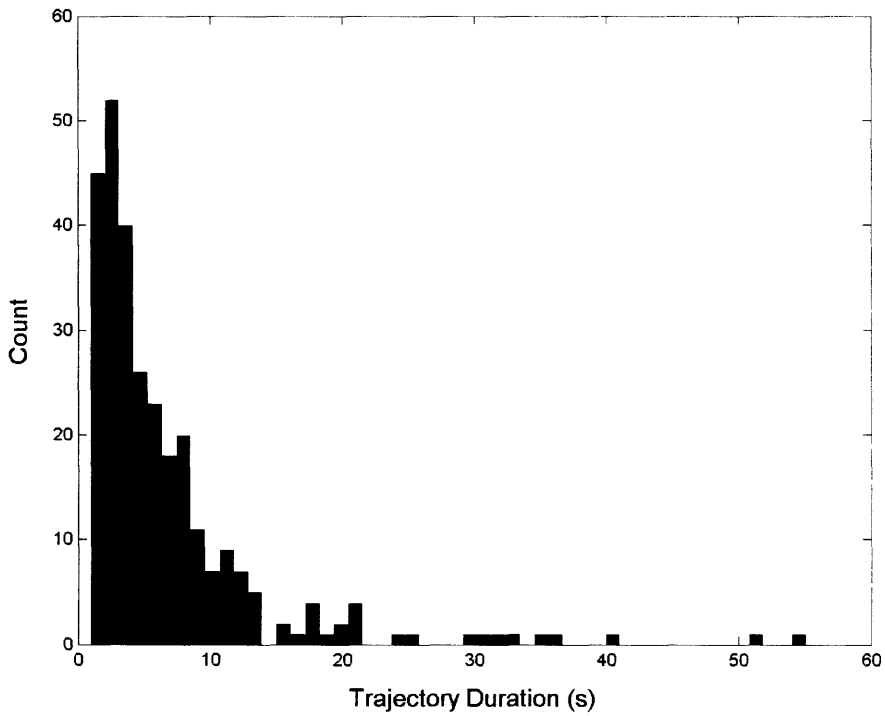

Figure 5.9: Histogram of trajectory durations for bacteria not exposed to UV radiation. The number of trajectories decreases exponentially as duration increases. Movies were taken for 80 s.

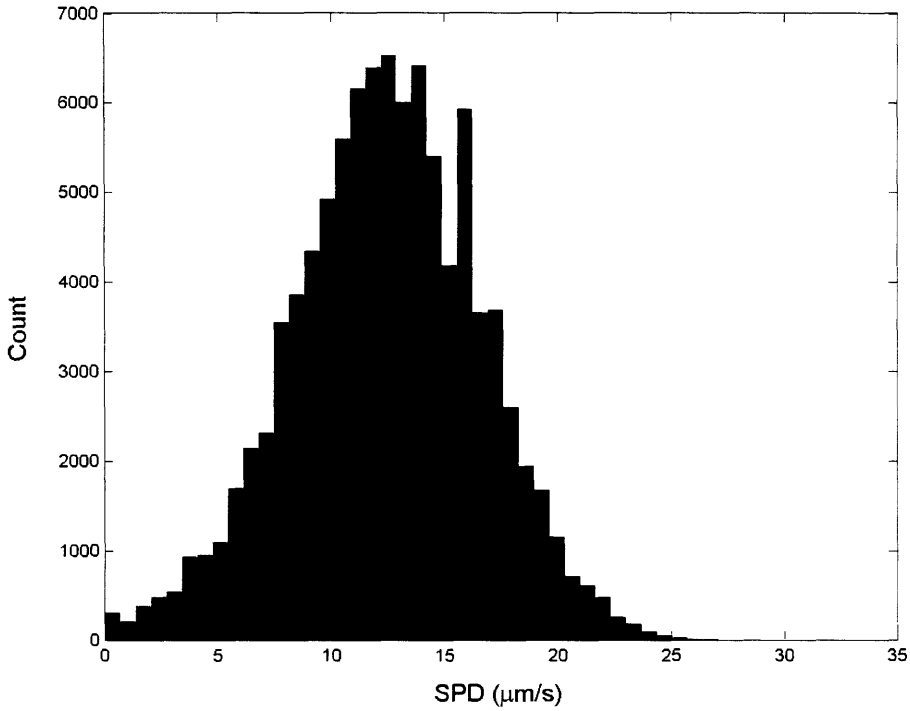

Figure 5.10: Histogram of *SPD* for bacteria not exposed to UV radiation. *SPD* is normally distributed, suggesting that the swimming speed for *E. coli* in a controlled, stable environment is a random variable.

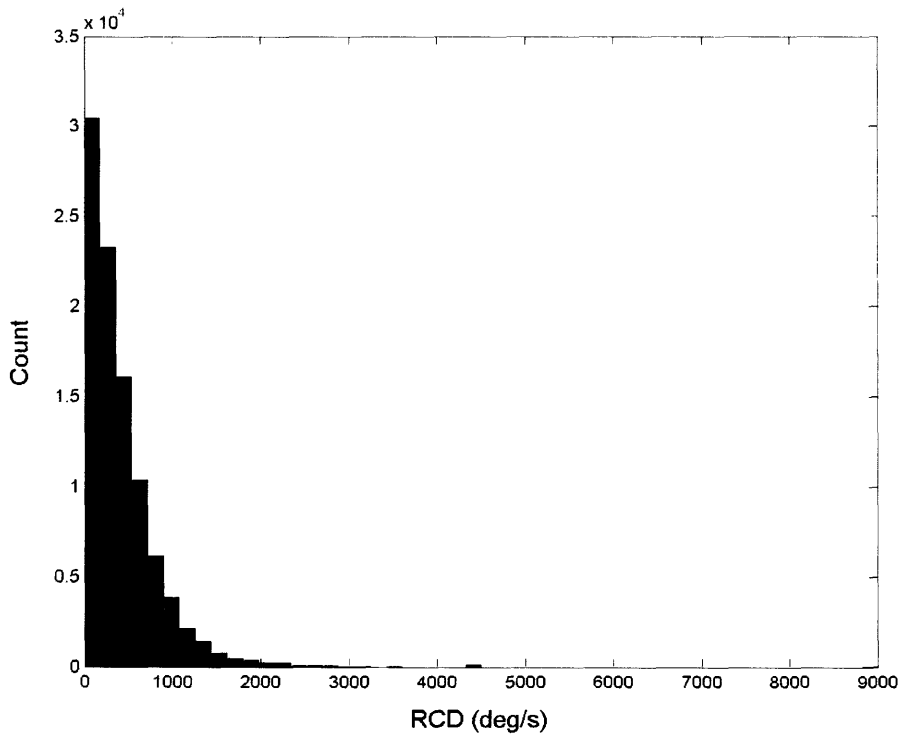

Figure 5.11: Histogram of *RCD* for bacteria not exposed to UV radiation. The number of trajectories decreases exponentially as *RCD* increases. Small changes in direction are expected from tumbling.

5.4.4. Ultraviolet Radiation

Figure 5.12 shows bacterial trajectories color-coded to illustrate values for trajectory duration. This is similar to what was done in Figure 5.8. Here, bacteria were exposed to ultraviolet radiation during the entire time images were recorded. One notices that bacterial trajectories are not equally distributed over the viewable surface but rather bacteria move toward the edges of the surface and away from the center where the UV radiation has the highest intensity. This plot is dramatically different from what was observed in Figure 5.8 and demonstrates that *E. coli* is repelled by UV radiation.

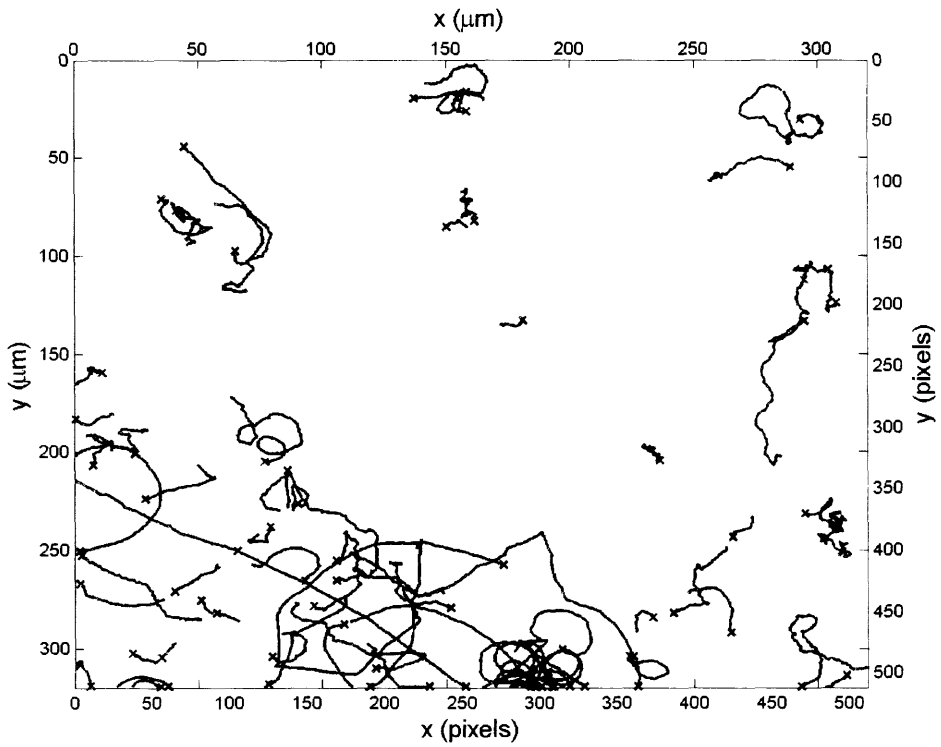

Figure 5.12: Bacterial trajectories color-coded to illustrate values for trajectory duration. Bacteria were exposed to UV radiation. Bacterial trajectories are not equally distributed over the viewable surface but rather bacteria move toward the edges of the surface and away from the center where the UV radiation has the highest intensity. The four colors used distinguish intervals of 25% of the maximum value. The coloring scheme going in increasing order is: Red, Green, Blue, and Black.

Figures 5.13–5.15 show histograms of trajectory duration, $SPD_{i,j}$, and $RCD_{i,j}$. Figure 5.13 shows that the number of trajectories decreases exponentially as the duration increases. Here, the average duration over all trajectories is 10.3 s and the minimum and maximum durations are 0.9 and 78.4 s, respectively. Figure 5.14 shows that values for $SPD_{i,j}$ have a decreasing linear distribution, where $\overline{\overline{SPD}}$ is 6.5 μm s^{-1} and the minimum and maximum $SPD_{i,j}$ are 0 and 30.1 μm s^{-1}, respectively. This distribution for $SPD_{i,j}$ is

dramatically different than the normal distribution that was observed when no UV radiation illuminated the bacteria. Figure 5.15 shows that the number of counts decreases exponentially as the $RCD_{i,j}$ increases, which is what occurred in Figure 5.11. However, $\overline{\overline{RCD}}$ is 1,318.9 deg s^{-1}, a factor of three larger, and there are many more counts at higher *RCD*. Therefore, larger changes in direction are expected from tumbling.

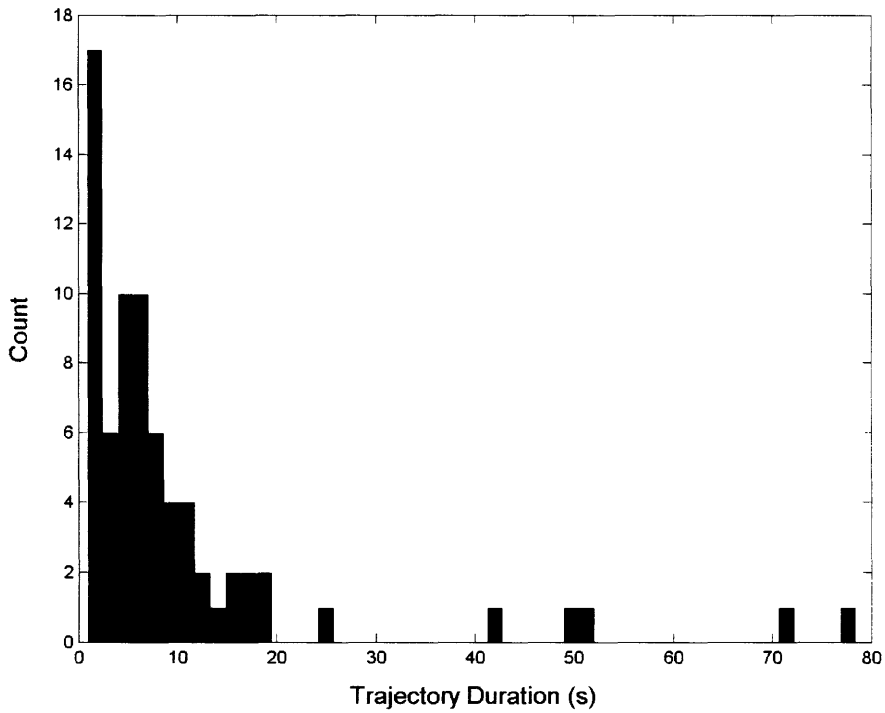

Figure 5.13: Histogram of trajectory durations for bacteria exposed to UV radiation. The number of trajectories decreases exponentially as duration increases. Movies were taken for 80 s.

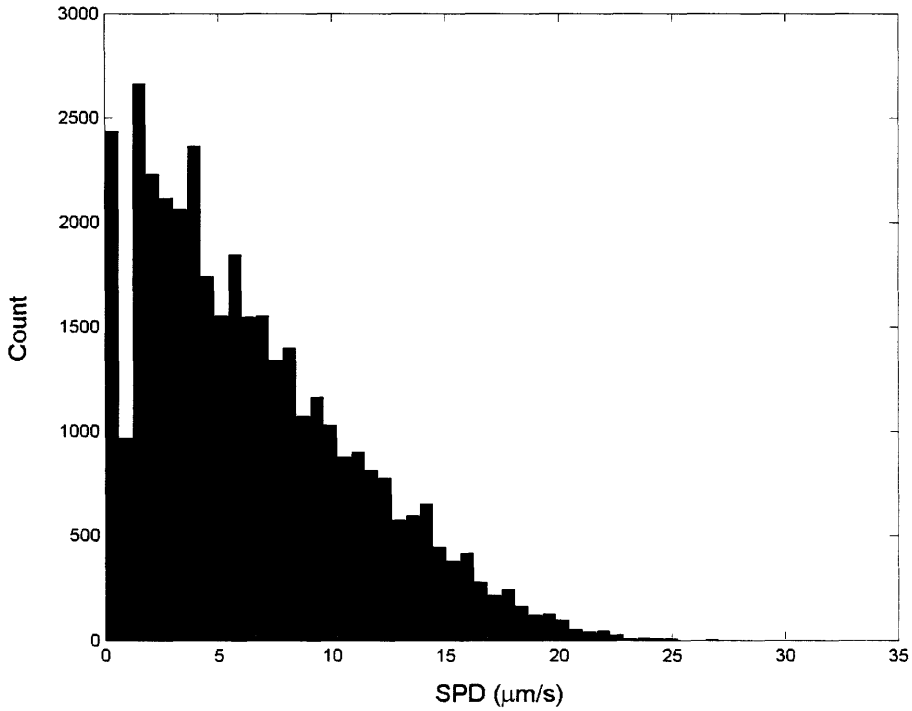

Figure 5.14: Histogram of *SPD* for bacteria exposed to UV radiation. *SPD* follows a decreasing linear distribution.

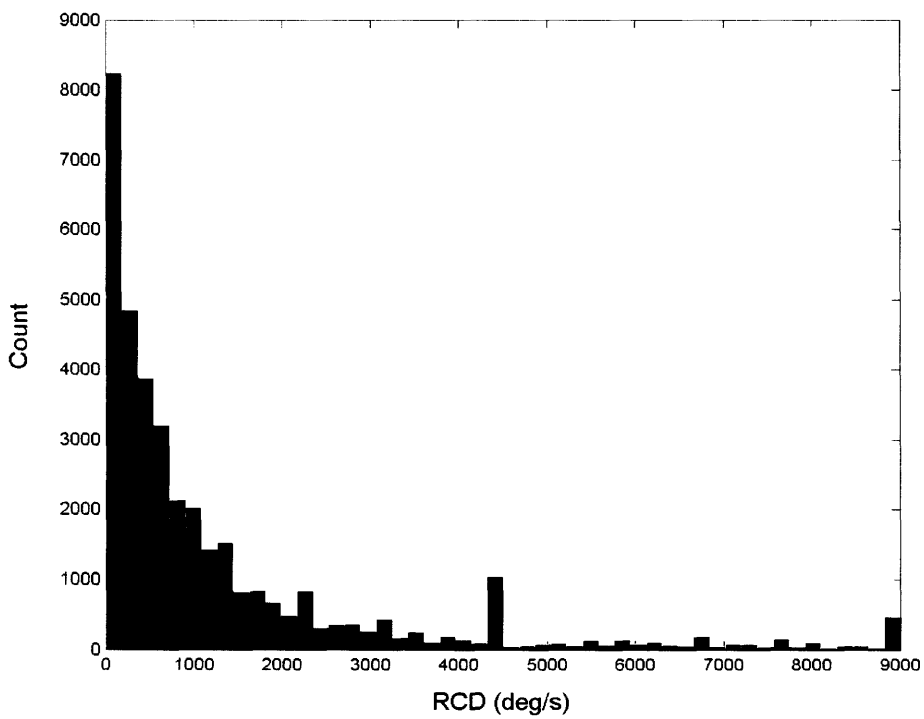

Figure 5.15: Histogram of *RCD* for bacteria exposed to UV radiation. The number of trajectories decreases exponentially as *RCD* increases. Larger changes in direction are expected from tumbling.

5.5. Conclusions

BacTrack™ was shown to be a reliable and effective piece of software for locating and tracking numerous bacteria recorded over many exposures. BacTrack™ was tested by obtaining data of bacteria dynamics when exposed/not exposed to ultraviolet radiation (global run/tumble properties summarized in Table 5.2). From Figures 5.8 and 5.12, one can easily identify that *E. coli* are repelled by UV. Furthermore, when no UV is present, *SPD* is normally distributed with a global average of 12.0 μm s^{-1}, but when UV is present, *SPD* follows a decreasing linear distribution with a global average of 6.5 μm s^{-1}. Additionally, the number of counts decreases exponentially as the *RCD* increases for cases when UV was and was not present. However, when UV was present there were many more counts at higher *RCD* values and the global average *RCD* was 1,318.9 deg s^{-1}. Yet, when UV was not present the global average *RCD* was 479.7 deg s^{-1}. Since changes in *SPD* and *RCD* are inversely proportional when tumbling occurs, one concludes that the bacteria exposed to UV are essentially in a persistent state of tumbling, trying to move in a direction away from the adverse effects of UV radiation. But, since UV illuminates the entire viewable surface, bacteria move toward regions of lowest UV intensity (i.e., away from the center). This conclusion is further supported by the global average tumbling frequency and duration increasing when UV illuminates the bacteria.

value	no uV	UV
Global average *SPD*	11.959 μm s^{-1}	6.462 μm s^{-1}
Global average *RCD*	479.651 deg s^{-1}	1,318.864 deg s^{-1}
Swimming speed	16.572 μm s^{-1}	14.854 μm s^{-1}
Global average tumble frequency	0.601 tumbles s^{-1}	1.474 tumbles s^{-1}
Global average tumble duration	0.087 s	0.343 s
Number of trajectories tracked initially	6,342	5,055
Number of trajectories satisfying all restrictions	288	72

Table 5.2 Global run/tumble properties associated with *E. coli* exposed/not exposed to ultraviolet radiation.

6. Conclusions

6.1. Further Research

Bacterial locomotion remains an exciting research topic both theoretically and experimentally. There are many issues in this field awaiting exploration which have vast application to biology, physics, mathematics, and ecology. Some ideas that would be interesting to investigate are:

1. Modeling chemotactic response to more complicated fluid dynamics such as stratified fluids and unsteady, periodic, rotating, and turbulent flows

2. Considering significant biological issues like bacteria adhering to an aggregate's surface; variable or periodic attractants; multiple or coupled tactic behaviors; and colony dynamics where individual bacteria communicate, group together, or isolate themselves

3. Confronting mathematical and dynamical systems questions about random walks, complicated random and continuous systems, and stochastic partial differential equations

Furthermore, there is much experimental work to be done especially with aggregates falling through viscous fluid. In nature, these sinking aggregates and their wake of enhanced concentration have an important role in deep-sea carbon export. Experiments pertaining to the falling alga problem could be carried out with microfluidic devices, which are small chips (3 cm in length) made of polydimethylsiloxane (PDMS) on which complex patterns of microchannels can be rapidly and accurately constructed using soft lithography. The experimental setup for a microfluidic device is shown in Figure 6.1. Microfluidic devices are useful because they allow for exploration of microscale environmental processes. Microfluidics has two advantages. First, microfluidics allows for control of geometries, flow rates, and concentration gradients. Second, the size and transparency of microchannels are suited for microscopy. Figure 6.2 shows an example of a microfluidic device mimicking a sinking particle. Similar to what was done in Chapters 3 and 4, Figure 6.3 shows sensing bacteria clustering in the concentration wake. The system presented in this figure was modeled to reflect an actual microfluidic device.

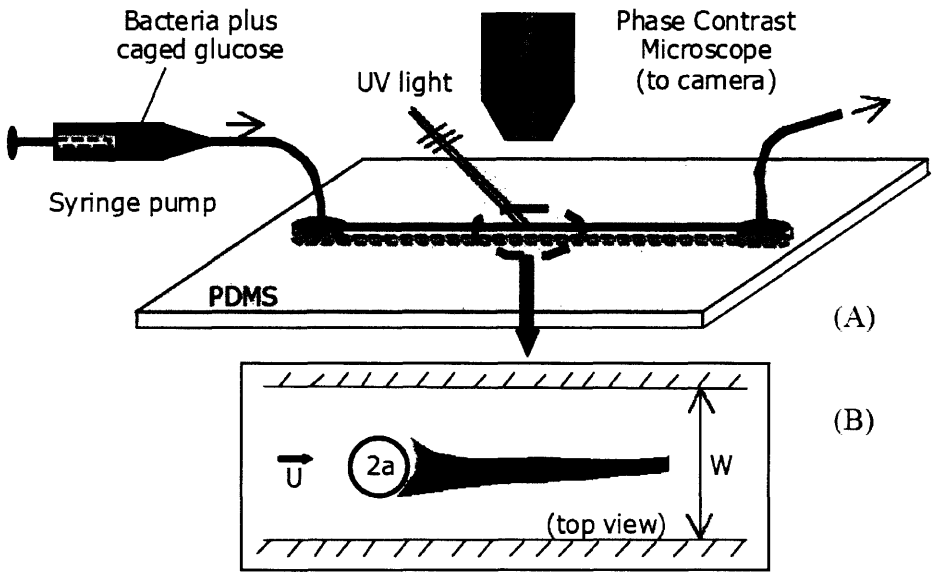

Figure 6.1: Experimental setup using a microfluidic device. (A) Flow is generated by a syringe pump. Nutrients enter via diffusion from a cylindrical post made of PDMS. (B) View from above the experimental setup. The post is shown as a solid-lined circle and represents the moving aggregate. There is a concentration wake (dark gray) leaving the post.

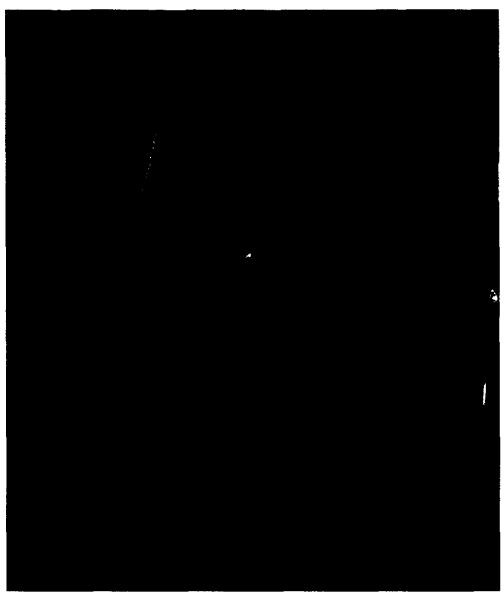

Figure 6.2: Example of a microfluidic device mimicking a moving particle. The black circle is the post. The streamlines were obtained using 2.1 μm fluorescent beads and a 1 s exposure.

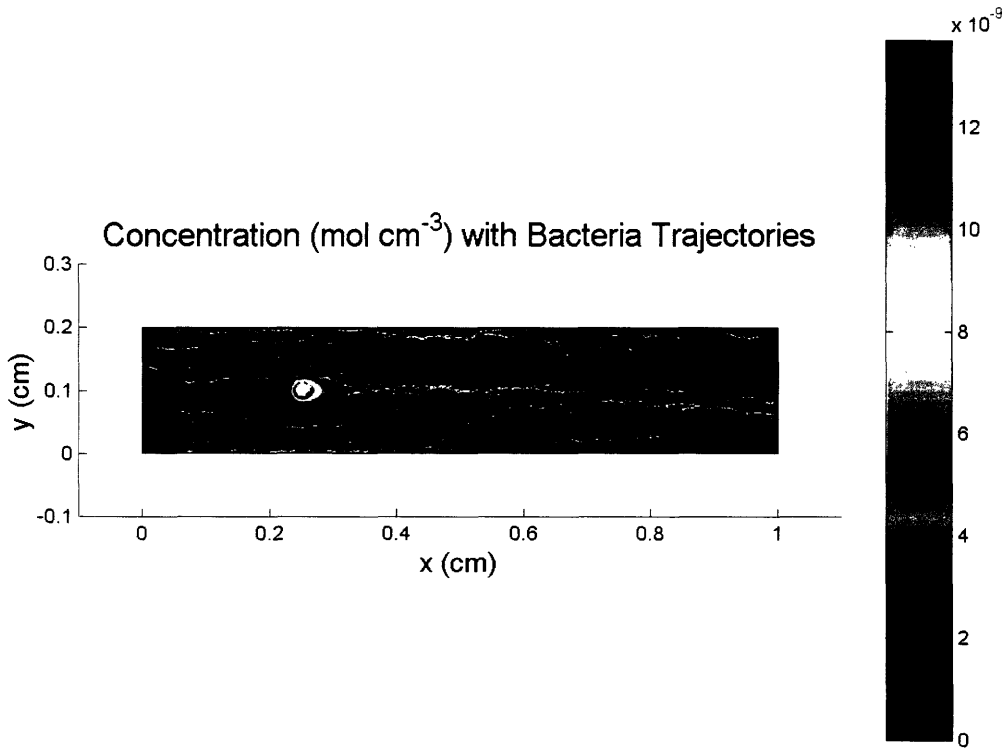

Figure 6.3: Sensing bacteria moving through microfluidic device. Bacteria cluster in the concentration wake. Re = 0.02, Pe = 23, $w_s = 25\times10^{-4}$ μm s^{-1}, and $v_0 = 12.5\times10^{-4}$ μm s^{-1}.

6.2. Remarks

In this work, models for bacterial chemotaxis in the stationary and moving particle problems were developed from earlier research. Chemotaxis associated with vortices was also investigated. Lastly, software was built to track bacteria over many exposures. Preliminary experiments were conducted with *E. coli* being exposed to ultraviolet radiation. This research found the following conclusions:

1. In the stationary particle problem, if the attractant (i.e., the leakage rate L) is not above a certain threshold for a given chemosensory constant α, a bacterial population, on average, increases its distance from the attracting particle, which looks like repulsion. This effect is purely geometrical, such that bacteria distributed with respect to a fixed point randomly walk, expand the distribution, and on average increase the distribution's distance from the fixed point since chemotactic attraction is weak. In cases where chemotactic thresholds for α and L aren't met, bacteria would be better served if they were inert.

2. In the moving particle problem, just by moving, motile bacteria put themselves at a disadvantage. The gain in the non-dimensional average integrated nutrient exposure $I - I_{inert}$ when $\alpha = 0$ gives the initial disadvantage chemotactic bacteria face. As α increases, $I - I_{inert}$ increases. α_{cr} is the chemotactic threshold for the moving particle problem and is defined when $I - I_{inert} = 0$. By determining α_{cr} for different values of s_0/a wit L fixed, one obtains three operational regimes defined by: i) sensing bacteria always being less effective than inert bacteria, ii) sensing bacteria being more effective than inert bacteria provided α_{cr} is met, or iii) sensing bacteria always being more effective than inert bacteria. This observation results from the velocity field u_z not being linear and going from convex to concave, such that as bacteria move to larger values of s they feel higher fluid velocities and are pushed away from the alga more quickly, resulting in smaller transit times.

3. When examining chemotaxis associated with vortices in the moving particle problem, there are three conditions that apply to bacteria entering vortices provided that bacteria do not alter their swimming speed or chemotactic mechanism: chemosensory ability has to be sufficiently high, $v_0/w_s \sim 1$, and $v_0/v_{vortex} \sim 1$. Aggregate colonization results from bacteria continuously swimming at a speed that allows them to move across velocity streamlines and into the vortex region (low velocity, high concentration region directly behind the aggregate). Vortex recirculation in the absence of aggregate colonization only occurs when the bacterial swimming speed is on the order of the average speed inside the vortices. However, with such a small swimming speed, bacteria are unable to effectively move across velocity streamlines and into the vortex region. When vortex recirculation is observed, aggregate colonization is not completely absent. To exploit vortex recirculation, bacteria must reduce their swimming speed when inside the vortex region. This altering of swimming speed can be done in relation to concentration gradients, where swimming speed decreases as the concentration gradient increases sharply in the vicinity of the vortex region.

4. BacTrack™ is software designed for the locating and tracking of numerous bacteria recorded over many exposures. BacTrack™ was tested and showed that *E. coli* are repelled by ultraviolet radiation. Bacteria constantly exposed to UV are essentially in a persistent state of tumbling, trying to move away from the adverse effects of UV radiation. But, since UV illuminates the entire viewable surface, the bacteria move toward regions of lowest UV intensity (i.e., away from the center).

7. Acknowledgement

The author graciously thanks Roman Stocker for his friendship and guidance during the researching and writing of this work, many delightful conversations, and everything he has taught the author. The author also thanks Scott Stransky for his effort in developing BacTrack™.

8. References

1. Stocker, R. (2004). Course Notes for Nonlinear Dynamics: Continuum Mechanics. Department of Mathematics, Massachusetts Institute of Technology.

2. Brown, D. A. and H. C. Berg. (1974). Proc. Nat. Acad. Sci. USA Vol 71, No. 4, pp. 1388-1392.

3. Engelmann, T. W. (1883) Pflugers Arch. Gesamte Physiol. Menschen Tiere 30, 95-124.

4. Berg, H. C. and D. A. Brown. (1972). Nature Vol. 239, pp. 500-504.

5. Jackson, G. A. (1987). Limnol. Oceanogr., 32 (6), 1253-1266.

6. Fitt, W. K. (1985). J. Phycol., 21: 62-67.

7. Berg, H. C. and E. M. Purcell. (1977). Biophys. J. 20: 193-219.

8. Johnson, M. (2005). Mechanisms of Bacterial Motility. Medical Microbiology, Indiana State University School of Medicine. http://www.indstate.edu/thcme/micro/flagella.html

9. Kyoto Encyclopedia of Genes and Genomes. (2005). Kanehisa Laboratories (Bioinformatics Center, Institute for Chemical Research, Kyoto University; Human Genome Center, Institute of Medical Science, University of Tokyo). http://www.genome.jp/kegg/pathway.html.

10. Jackson, G. A. (1989). Limnol. Oceanor. 34 (3): 514-530.

11. Mitchell, J. G., A. Okubo, and J. A. Furman. (1985). Nature 316: 58-59.

12. Acheson, D. J. (1990). Elementary Fluid Dynamics. Oxford University Press.

13. Smayda, T. J. (1970). Oceanography and Marine Biology an Annual Review Vol. 8 pp. 353-414.

14. Acrivos, A. and T. D. Taylor. (1962). Phys. Fluids. 5: 387-394.

15. Acrivos, A. and J. D. Goddard. (1965). J. Fluid Mech. 23: 273-291.

16. Fellipa, C. A. (Fall 2004). Course Notes for Introduction to Finite Element Methods. Department of Aerospace Engineering Sciences, University of Colorado.

17. COMSOL. (2005). FEMLAB Version 3.1 Product Manual.

18. Kiorboe, T. and G. A. Jackson. (2001). Limnol. Oceanogr., Vol. 46, No. 6, pp. 1309 – 1318.

19. Ritter, G. X. (2000). Handbook of Computer Vision Algorithms in Image Algebra. Second Edition. CRC Press LLC, Boca Raton, Florida.

20. Research Services Branch, National Institute of Mental Health and National Institute of Neurological Disorders and Stroke. (2005). Image Processing and Analysis in Java. http://rsb.info.nih.gov/ij/docs/menus/process.html

21. Mittal, N., E. O. Budrene, M. P. Brenner, and A. van Oudenaarden. (2003). Proc. Nat. Acad. Sci. USA, Vol. 100, No. 23, pp. 13259 – 13263.

22. Frymier, P. D., R. M. Ford, H. C. Berg, and P. T. Cummings. (1995). Three-Dimensional Tracking of Motile Bacteria near a Solid Planar Surface. Proc. Nat. Acad. Sci. USA, Vol. 92, No. 13, pp. 6195 – 6199.

A. Appendix

The sample codes presented in this appendix were written in MATLAB, a software package produced by The MathWorks. These codes are provided as a reference for future theoretical research.

A.1. Samples of Chemotaxis Simulation Codes

A.1.1. Stationary Particle Problem

The following code investigates variation in leakage rate L for sensing bacteria in the stationary particle problem. There is no upper boundary constraint on the distance that a bacterium can travel away from the alga (R_{max}).

```
% Copyright Michael David Sekora, 7 May 2005

clear all; close all;

matrix = [{'L_0.mat' ...
           'L_0_5.mat' ...
           'L_1.mat' ...
           'L_1_5.mat' ...
           'L_2.mat' ...
           'L_2_5.mat' ...
           'L_3.mat' ...
           'L_3_5.mat' ...
           'L_4.mat' ...
           'L_4_5.mat' ...
           'L_5.mat'}];

L  = [0:0.5:5];

for g = 1:length(matrix);

   leakd          = L(g);
   a              = 10e-4;
   alpha          = 660;            % sec
   T_m            = 1;              % sec

   batch          = 50;
   bacteria       = 100;
   time           = 50;             % sec
   dir_bias       = 0;
   dist_bound     = 1;
   min_dist       = a;

   v              = 12.3e-4;        % cm*sec^-1
   dt             = 0.067;          % sec
   dd             = v*dt;           % cm
   D              = 1e-5;           % diffusivity, cm^2/s
   K_D            = 1e-7;           % mol*cm^-3
```

```
tau_0          = 0.67;           % sec
nmolec         = 1;

leak           = leakd/8.64e4;   % leakage rate, s^-1
C_const        = 1.67e-4*a^2.28*leak/(4*pi*D*nmolec);    % mol*cm^-3

for h = 1:batch;

   for i = 1:bacteria;

      j           = 1;           % index for total number of steps
      t           = 0;
      pos(:,j)    = [0;0;7.5*a];
      r(j)        = norm(pos(:,j));
      v_random    = (rand(3,1) - 0.5)/norm(rand(3,1) - 0.5)*v;
      C(j)        = C_const/r(j);
      dCdt(j)     = 0;
      dPbdt(j)    = 0;
      avedPbdt(j) = 0;
      dPconst     = exp(-dt/T_m);
      lntau_0     = log(tau_0);
      tau         = tau_0;

      while t <= time;

         if dt/tau   > rand;     % if it tumbles
            if dir_bias == 1;
               gamma = (pi/180)*raylrnd(54.1896);
               nu    = 2*pi*rand;
               theta = acos(cos(theta)*cos(gamma) - ...
                           sin(theta)*sin(gamma)*sin(nu));
            else
               v_random = (rand(3,1) - 0.5)/norm(rand(3,1) - 0.5)*v;
            end
         end

         j = j+1;  t = j*dt;
         pos(:,j)    = pos(:,j-1) + v_random*dt;
         r(j)        = norm(pos(:,j));
         if dist_bound == 1;
            if r(j) <= a;
               v_random = -v_random;
               pos(:,j) = pos(:,j-1) + v_random*dt;
               r(j)     = norm(pos(:,j));
            end
         end

         C(j)        = C_const/r(j);
         dCdt(j)     = (C(j) - C(j-1))/dt;
         dPbdt(j)    = K_D/((K_D + C(j))^2)*dCdt(j);
         avedPbdt(j) = dPconst*avedPbdt(j-1) + dPbdt(j)*dt/T_m;
         lntau       = lntau_0 + alpha*avedPbdt(j);
         tau         = exp(lntau);

      end

      r_temp(i,:) = real(r);
```

```
        s_temp(i,:) = real(sqrt(pos(1,:).^2 + pos(2,:).^2));
        x_temp(i,:) = real(pos(1,:));
        y_temp(i,:) = real(pos(2,:));
        z_temp(i,:) = real(pos(3,:));
        C_temp(i,:) = real(C);
        clear r pos C dCdt dPbdt avedPbdt;

    end

    R((1:bacteria)+(h-1)*bacteria,:) = r_temp;
    S((1:bacteria)+(h-1)*bacteria,:) = s_temp;
    X((1:bacteria)+(h-1)*bacteria,:) = x_temp;
    Y((1:bacteria)+(h-1)*bacteria,:) = y_temp;
    Z((1:bacteria)+(h-1)*bacteria,:) = z_temp;
    Conc((1:bacteria)+(h-1)*bacteria,:) = C_temp;
    clear r_temp s_temp x_temp y_temp z_temp C_temp;

end

T = [0:dt:t-dt];

save(char(cellstr(matrix(g))));

end
```

A.1.2. Analytical Moving Particle Problem

The following code investigates variation in the chemosensory constant α for sensing bacteria in the analytical moving particle problem with high Pe.

```
% Copyright Michael David Sekora, 7 May 2005

clear all; close all;

folder = [{'C:\CONCENTRATION_VS_ALPHA\SENSING\ND_S_0_625' ...
           'C:\CONCENTRATION_VS_ALPHA\SENSING\ND_S_0_875' ...
           'C:\CONCENTRATION_VS_ALPHA\SENSING\ND_S_1_125' ...
           'C:\CONCENTRATION_VS_ALPHA\SENSING\ND_S_1_375'}];

matrix = [{'ALPHA_0.mat' ...
           'ALPHA_100.mat' ...
           'ALPHA_200.mat' ...
           'ALPHA_300.mat' ...
           'ALPHA_400.mat' ...
           'ALPHA_500.mat' ...
           'ALPHA_600.mat' ...
           'ALPHA_700.mat' ...
           'ALPHA_800.mat' ...
           'ALPHA_900.mat' ...
           'ALPHA_1000.mat' ...
           'ALPHA_1100.mat' ...
           'ALPHA_1200.mat' ...
           'ALPHA_1300.mat' ...
```

```
            'ALPHA_1400.mat' ...
            'ALPHA_1500.mat' ...
            'ALPHA_1600.mat' ...
            'ALPHA_1700.mat' ...
            'ALPHA_1800.mat' ...
            'ALPHA_1900.mat' ...
            'ALPHA_2000.mat'}];

ALPHA       = [0:100:2000];
a           = 50e-4;
Y_initial   = [0.625:0.25:1.375]*a;

for f = 1:length(folder);

   cd(char(cellstr(folder(f))));

   for g = 1:length(matrix);

      a              = 50e-4;
      leakd          = 0.5;            % d^-1
      v0             = 12.5e-4;        % cm/s
      f_rho          = 2;

      disperse_b     = 0;
      batch          = 10;
      bacteria       = 100;
      time           = 50;             % s
      low_Pe_inner_r = 1.25*a;

      xi             = 1.24;
      viscosity      = 0.01;           % kinematic, cm^2/s
      alpha          = ALPHA(g);       % s
      T_m            = 1;              % s
      dt             = 0.05;           % s
      b              = 1.67e-4;
      D              = 1e-5;           % diffusivity, cm^2/s
      K_D            = 1e-7;           % mol cm^-3
      tau_0          = 0.67;           % s

      leak           = leakd/8.64e4;   % leakage rate, s^-1
      w_s            = xi*f_rho*a^1.17;
      Reynolds       = 2*a*w_s/viscosity;
      Peclet         = 2*a*w_s/D;
      S_max          = 5*a*v0/w_s;
      if Peclet      < 1;
         Sh          = 1 + 0.25*Peclet;
      else
         Sh          = 0.461 + 0.496*Peclet^0.3333;
      end
      if disperse_b == 1;
         Y_initial   = [S_max/batch:S_max/batch:S_max];
      end
      C_const        = b*a^2.28*leak/(4*pi*D*Sh*a);     % mol*cm^-3

      for h = 1:batch;

         for i = 1:bacteria;
```

```
j           = 1;     % index for total number of steps
t           = 0;
pos(:,j)    = [0;Y_initial(f);-5*a];
r(j)        = norm(pos(:,j));
v_random    = (rand(3,1) - 0.5)/norm(rand(3,1) - 0.5)*v0;

theta_p     = real(acos(-pos(3,j)/r(j)));
zeta        = abs(real(Peclet*(r(j) - a)^3* ...
                  (sin(theta_p))^3/ ...
                  (3*a^3*(theta_p - 0.5*sin(2*theta_p)))));
C(j)        = C_const*gammainc(0.3333,zeta)/gamma(0.3333);

dCdt(j)     = 0;
dPbdt(j)    = 0;
avedPbdt(j) = 0;
dPconst     = exp(-dt/T_m);
lntau_0     = log(tau_0);
tau         = tau_0;

while t       <= time;

   if dt/tau   > rand;     % if it tumbles
      v_random = (rand(3,1) - 0.5)/ ...
                 norm(rand(3,1) - 0.5)*v0;
   end
   u_flow      = [w_s*3*a*pos(1,j)*pos(3,j)/ ...
                     (4*r(j)^3)*(-1 + a^2/r(j)^2); ...
                  w_s*3*a*pos(2,j)*pos(3,j)/ ...
                     (4*r(j)^3)*(-1 + a^2/r(j)^2); ...
                  w_s*(1 - 3*a/(4*r(j))* ...
                     (1 + a^2/(3*r(j)^2) + ...
                     pos(3,j)^2/r(j)^2 - ...
                     a^2*pos(3,j)^2/r(j)^4))];
   v_b         = u_flow + v_random;

   j           = j + 1;
   t           = j*dt;
   pos(:,j)    = pos(:,j-1) + v_b*dt;
   r(j)        = norm(pos(:,j));
   if r(j)    <= a;
      v_b      = -v_b;
      pos(:,j) = pos(:,j-1) + v_b*dt;
      r(j)     = norm(pos(:,j));
   end

   theta_p     = real(acos(-pos(3,j)/r(j)));
   zeta        = abs(real(Peclet*(r(j) - a)^3* ...
                     (sin(theta_p))^3/ ...
                     (3*a^3*(theta_p - ...
                     0.5*sin(2*theta_p)))));
   C(j)        = C_const*gammainc(0.3333,zeta)/ ...
                 gamma(0.3333);

   dCdt(j)     = (C(j) - C(j-1))/dt;
   dPbdt(j)    = K_D/((K_D + C(j))^2)*dCdt(j);
   avedPbdt(j) = dPconst*avedPbdt(j-1) + dPbdt(j)*dt/T_m;
```

```
            lntau       = lntau_0 + alpha*avedPbdt(j);
            tau         = exp(lntau);

        end

        r_temp(i,:) = real(r);
        s_temp(i,:) = real(sqrt(pos(1,:).^2 + pos(2,:).^2));
        x_temp(i,:) = real(pos(1,:));
        y_temp(i,:) = real(pos(2,:));
        z_temp(i,:) = real(pos(3,:));
        C_temp(i,:) = real(C);
        clear r pos C dCdt dPbdt avedPbdt;

    end

    R((1:bacteria)+(h-1)*bacteria,:) = r_temp;
    S((1:bacteria)+(h-1)*bacteria,:) = s_temp;
    X((1:bacteria)+(h-1)*bacteria,:) = x_temp;
    Y((1:bacteria)+(h-1)*bacteria,:) = y_temp;
    Z((1:bacteria)+(h-1)*bacteria,:) = z_temp;
    Conc((1:bacteria)+(h-1)*bacteria,:) = C_temp;
    clear r_temp s_temp x_temp y_temp z_temp C_temp;

end

T = [0:dt:t-dt];

save(char(cellstr(matrix(g))));

end

end
```

A.1.3. Numerical Moving Particle Problem

The following code simulates particle paths for sensing bacteria in the realistic case of the numerical moving particle problem. This code imports data for the concentration and velocity fields, which were solved for in FEMLAB.

```
% Copyright Michael David Sekora, 7 May 2005

clear all; close all;

coarse_matrix  = [{'C:\Re_20_REALISTIC\coarse_data.mat'}];
fine_matrix    = [{'C:\Re_20_REALISTIC\fine_data.mat'}];

save_folder    = [{'C:\Re_20_REALISTIC\PARTICLE_PATHS\SENSING'}];
save_matrix    = [{'sensing.mat'}];

A           = [0.8];
ALGA_CENTER = [20;8];

for g = 1:length(coarse_matrix);
```

```
a               = A(g);
leakd           = 0.5;              % d^-1
v0              = 200e-4;           % cm/s
f_rho           = 2;

Y_initial       = 0;
disperse_b      = 1;
batch           = 4;
bacteria        = 1;
time            = 1000;             % s
low_Pe_inner_r  = 1.25*a;

xi              = 1.24;
viscosity       = 1e-2;             % kinematic, cm^2/s
alpha           = 660;              % s
T_m             = 1;                % s
dt              = 0.05;             % s
b               = 1.67e-4;
D               = 1e-5;             % diffusivity, cm^2/s
K_D             = 1e-7;             % mol cm^-3
tau_0           = 0.67;             % s

leak            = leakd/8.64e4;     % leakage rate, s^-1
w_s             = 0.125;
Reynolds        = 2*a*w_s/viscosity;
Peclet          = 2*a*w_s/D;
if Peclet       < 1;
   Sh           = 1 + 0.25*Peclet;
else
   Sh           = 0.461 + 0.496*Peclet^0.3333;
end

S_max           = 10*a;
if disperse_b == 1;
   Y_initial    = [2*S_max/(batch+1):2*S_max/(batch+1): ...
                                    2*S_max*(1-1/(batch+1))];
end
C_const         = 4.5e-6;           % mol*cm^-3

alga_center     = ALGA_CENTER(1:2,g);
epsilon         = 1e-9;
magnitude       = 2;

load(char(cellstr(coarse_matrix(g))));
coarse_concentration = concentration;
coarse_x_velocity    = x_velocity;
coarse_y_velocity    = y_velocity;
coarse_x_coordinate  = x_coordinate;
coarse_y_coordinate  = y_coordinate;
clear concentration x_velocity y_velocity ...
      x_coordinate y_coordinate;

load(char(cellstr(fine_matrix(g))));
fine_concentration   = concentration;
fine_x_velocity      = x_velocity;
fine_y_velocity      = y_velocity;
```

```
fine_x_coordinate    = x_coordinate;
fine_y_coordinate    = y_coordinate;
clear concentration x_velocity y_velocity ...
      x_coordinate y_coordinate;

[x_vel_nan_row,x_vel_nan_column] = find(isnan( ...
                                    coarse_x_velocity)==1);
[y_vel_nan_row,y_vel_nan_column] = find(isnan( ...
                                    coarse_y_velocity)==1);
[conc_nan_row,conc_nan_column]   = find(isnan( ...
                                    coarse_concentration)==1);
if x_vel_nan_row  > 0;
   for k = 1:length(x_vel_nan_row);
      coarse_x_velocity(x_vel_nan_row(k), ...
                        x_vel_nan_column(k))  = w_s/25;
   end
end
if y_vel_nan_row  > 0;
   for k = 1:length(y_vel_nan_row);
      coarse_y_velocity(y_vel_nan_row(k), ...
                        y_vel_nan_column(k))  = w_s/25;
   end
end
if conc_nan_row   > 0;
   for k = 1:length(conc_nan_row);
      coarse_concentration(conc_nan_row(k), ...
                           conc_nan_column(k)) = C_const;
   end
end
clear x_vel_nan_row x_vel_nan_column ...
      y_vel_nan_row y_vel_nan_column ...
      conc_nan_row conc_nan_column;

[x_vel_nan_row,x_vel_nan_column] = find(isnan( ...
                                    fine_x_velocity)==1);
[y_vel_nan_row,y_vel_nan_column] = find(isnan( ...
                                    fine_y_velocity)==1);
[conc_nan_row,conc_nan_column]   = find(isnan( ...
                                    fine_concentration)==1);
if x_vel_nan_row  > 0;
   for k = 1:length(x_vel_nan_row);
      fine_x_velocity(x_vel_nan_row(k), ...
                      x_vel_nan_column(k))  = w_s/25;
   end
end
if y_vel_nan_row  > 0;
   for k = 1:length(y_vel_nan_row);
      fine_y_velocity(y_vel_nan_row(k), ...
                      y_vel_nan_column(k))  = w_s/25;
   end
end
if conc_nan_row   > 0;
   for k = 1:length(conc_nan_row);
      fine_concentration(conc_nan_row(k), ...
                         conc_nan_column(k)) = C_const;
   end
end
```

```
clear x_vel_nan_row x_vel_nan_column ...
      y_vel_nan_row y_vel_nan_column ...
      conc_nan_row conc_nan_column;

coarse_x_grid_step   = coarse_x_coordinate(2) - ...
                           coarse_x_coordinate(1);
coarse_y_grid_step   = coarse_y_coordinate(2) - ...
                           coarse_y_coordinate(1);
fine_x_grid_step     = fine_x_coordinate(2) - ...
                           fine_x_coordinate(1);
fine_y_grid_step     = fine_y_coordinate(2) - ...
                           fine_y_coordinate(1);

for h = 1:batch;

   if disperse_b == 1;
      y_initial   = Y_initial(h);
   else
      y_initial   = Y_initial(g);
   end

   for i = 1:bacteria;

      j           = 1;     % index for total number of steps
      t           = 0;
      pos(:,j)    = [coarse_x_coordinate(2);y_initial];
      r(j)        = sqrt((pos(1,j) - alga_center(1))^2 + ...
                        (pos(2,j) - alga_center(2))^2);
      v_random    = (rand(2,1) - 0.5)/norm(rand(2,1) - 0.5)*v0;

      if pos(1,j) > coarse_x_coordinate(end-1);
         break;
      elseif pos(1,j)   > fine_x_coordinate(2) & ...
             pos(1,j)   < fine_x_coordinate(end-1) & ...
             pos(2,j)   > fine_y_coordinate(2) & ...
             pos(2,j)   < fine_y_coordinate(end-1);
         fine_x_grid_index  = ceil((pos(1,j) - ...
                                 fine_x_coordinate(1))/ ...
                                 fine_x_grid_step);
         fine_y_grid_index  = ceil((pos(2,j) - ...
                                 fine_y_coordinate(1))/ ...
                                 fine_y_grid_step);
         C(j)  = sum([fine_concentration(fine_y_grid_index, ...
                        fine_x_grid_index)/sqrt(( ...
                        fine_x_coordinate( ...
                        fine_x_grid_index) - pos(1,j))^2 + ...
                        (fine_y_coordinate( ...
                        fine_y_grid_index) - pos(2,j))^2 + ...
                        epsilon)^magnitude,...
                     fine_concentration(fine_y_grid_index, ...
                        fine_x_grid_index-1)/sqrt(( ...
                        fine_x_coordinate( ...
                        fine_x_grid_index-1) - pos(1,j))^2 + ...
                        (fine_y_coordinate( ...
                        fine_y_grid_index) - pos(2,j))^2 + ...
                        epsilon)^magnitude,...
                     fine_concentration(fine_y_grid_index-1, ...
```

```
                        fine_x_grid_index)/sqrt(( ...
                        fine_x_coordinate( ...
                        fine_x_grid_index) - pos(1,j))^2 + ...
                        (fine_y_coordinate( ...
                        fine_y_grid_index-1) - pos(2,j))^2 + ...
                        epsilon)^magnitude,...
                    fine_concentration(fine_y_grid_index-1, ...
                        fine_x_grid_index-1)/sqrt(( ...
                        fine_x_coordinate( ...
                        fine_x_grid_index-1) - pos(1,j))^2 + ...
                        (fine_y_coordinate( ...
                        fine_y_grid_index-1) - pos(2,j))^2 + ...
                        epsilon)^magnitude])/...
                sum([1/sqrt((fine_x_coordinate( ...
                            fine_x_grid_index) - ...
                            pos(1,j))^2 + (fine_y_coordinate( ...
                            fine_y_grid_index) - pos(2,j))^2 + ...
                            epsilon)^magnitude,...
                    1/sqrt((fine_x_coordinate( ...
                            fine_x_grid_index-1) - ...
                            pos(1,j))^2 + (fine_y_coordinate( ...
                            fine_y_grid_index) - pos(2,j))^2 + ...
                            epsilon)^magnitude,...
                    1/sqrt((fine_x_coordinate( ...
                            fine_x_grid_index) - ...
                            pos(1,j))^2 + (fine_y_coordinate( ...
                            fine_y_grid_index-1) - ...
                            pos(2,j))^2 + epsilon)^magnitude,...
                    1/sqrt((fine_x_coordinate( ...
                            fine_x_grid_index-1) - ...
                            pos(1,j))^2 + (fine_y_coordinate( ...
                            fine_y_grid_index-1) - ...
                            pos(2,j))^2 + epsilon)^magnitude]);
    else
        coarse_x_grid_index  = ceil((pos(1,j))/ ...
                                coarse_x_grid_step);
        coarse_y_grid_index  = ceil((pos(2,j))/ ...
                                coarse_y_grid_step);
        C(j)  = sum([coarse_concentration( ...
                        coarse_y_grid_index, ...
                        coarse_x_grid_index)/sqrt(( ...
                        coarse_x_coordinate( ...
                        coarse_x_grid_index) - pos(1,j))^2 + ...
                        (coarse_y_coordinate( ...
                        coarse_y_grid_index) - pos(2,j))^2 + ...
                        epsilon)^magnitude,...
                    coarse_concentration( ...
                        coarse_y_grid_index, ...
                        coarse_x_grid_index-1)/sqrt(( ...
                        coarse_x_coordinate( ...
                        coarse_x_grid_index-1) - pos(1,j))^2 + ...
                        (coarse_y_coordinate( ...
                        coarse_y_grid_index) - pos(2,j))^2 + ...
                        epsilon)^magnitude,...
                    coarse_concentration( ...
                        coarse_y_grid_index-1, ...
                        coarse_x_grid_index)/sqrt(( ...
```

```
                    coarse_x_coordinate( ...
                    coarse_x_grid_index) - pos(1,j))^2 + ...
                    (coarse_y_coordinate( ...
                    coarse_y_grid_index-1) - pos(2,j))^2 + ...
                    epsilon)^magnitude,...
                coarse_concentration( ...
                    coarse_y_grid_index-1, ...
                    coarse_x_grid_index-1)/sqrt(( ...
                    coarse_x_coordinate( ...
                    coarse_x_grid_index-1) - pos(1,j))^2 + ...
                    (coarse_y_coordinate( ...
                    coarse_y_grid_index-1) - pos(2,j))^2 + ...
                    epsilon)^magnitude])/...
            sum([1/sqrt((coarse_x_coordinate( ...
                        coarse_x_grid_index) - ...
                        pos(1,j))^2 + ...
                        (coarse_y_coordinate( ...
                        coarse_y_grid_index) - ...
                        pos(2,j))^2 + epsilon)^magnitude,...
                1/sqrt((coarse_x_coordinate( ...
                        coarse_x_grid_index-1) - ...
                        pos(1,j))^2 + ...
                        (coarse_y_coordinate( ...
                        coarse_y_grid_index) - ...
                        pos(2,j))^2 + epsilon)^magnitude,...
                1/sqrt((coarse_x_coordinate( ...
                        coarse_x_grid_index) - ...
                        pos(1,j))^2 + ...
                        (coarse_y_coordinate( ...
                        coarse_y_grid_index-1) - ...
                        pos(2,j))^2 + epsilon)^magnitude,...
                1/sqrt((coarse_x_coordinate( ...
                        coarse_x_grid_index-1) - ...
                        pos(1,j))^2 + ...
                        (coarse_y_coordinate( ...
                        coarse_y_grid_index-1) - ...
                        pos(2,j))^2 + epsilon)^magnitude]);
end

dCdt(j)      = 0;
dPbdt(j)     = 0;
avedPbdt(j)  = 0;
dPconst      = exp(-dt/T_m);
lntau_0      = log(tau_0);
tau          = tau_0;

while t       <= time;

   if dt/tau   > rand;     % if it tumbles
      v_random = (rand(2,1) - 0.5)/norm(rand(2,1) - 0.5)*v0;
   end

   if pos(1,j) > fine_x_coordinate(2) & ...
      pos(1,j) < fine_x_coordinate(end-1) & ...
      pos(2,j) > fine_y_coordinate(2) & ...
      pos(2,j) < fine_y_coordinate(end-1);
      x_flow   = sum([fine_x_velocity( ...
```

```
                fine_y_grid_index, ...
                fine_x_grid_index)/sqrt(( ...
                fine_x_coordinate( ...
                fine_x_grid_index) - ...
                pos(1,j))^2 + ...
                (fine_y_coordinate( ...
                fine_y_grid_index) - ...
                pos(2,j))^2 + ...
                epsilon)^magnitude,...
            fine_x_velocity( ...
                fine_y_grid_index, ...
                fine_x_grid_index-1)/sqrt(( ...
                fine_x_coordinate( ...
                fine_x_grid_index-1) - ...
                pos(1,j))^2 + ...
                (fine_y_coordinate( ...
                fine_y_grid_index) - ...
                pos(2,j))^2 + ...
                epsilon)^magnitude,...
            fine_x_velocity( ...
                fine_y_grid_index-1, ...
                fine_x_grid_index)/sqrt(( ...
                fine_x_coordinate( ...
                fine_x_grid_index) - ...
                pos(1,j))^2 + ...
                (fine_y_coordinate( ...
                fine_y_grid_index-1) - ...
                pos(2,j))^2 + ...
                epsilon)^magnitude,...
            fine_x_velocity( ...
                fine_y_grid_index-1, ...
                fine_x_grid_index-1)/sqrt(( ...
                fine_x_coordinate( ...
                fine_x_grid_index-1) - ...
                pos(1,j))^2 + ...
                (fine_y_coordinate( ...
                fine_y_grid_index-1) - ...
                pos(2,j))^2 + ...
                epsilon)^magnitude])/...
        sum([1/sqrt((fine_x_coordinate( ...
                    fine_x_grid_index) - ...
                    pos(1,j))^2 + ...
                    (fine_y_coordinate( ...
                    fine_y_grid_index) - ...
                    pos(2,j))^2 + ...
                    epsilon)^magnitude,...
            1/sqrt((fine_x_coordinate( ...
                    fine_x_grid_index-1) - ...
                    pos(1,j))^2 + ...
                    (fine_y_coordinate( ...
                    fine_y_grid_index) - ...
                    pos(2,j))^2 + ...
                    epsilon)^magnitude,...
            1/sqrt((fine_x_coordinate( ...
                    fine_x_grid_index) - ...
                    pos(1,j))^2 + ...
                    (fine_y_coordinate( ...
```

```
                                fine_y_grid_index-1) - ...
                                pos(2,j))^2 + ...
                                epsilon)^magnitude,...
                          1/sqrt((fine_x_coordinate( ...
                                fine_x_grid_index-1) - ...
                                pos(1,j))^2 + ...
                                (fine_y_coordinate( ...
                                fine_y_grid_index-1) - ...
                                pos(2,j))^2 + ...
                                epsilon)^magnitude]);
y_flow    = sum([fine_y_velocity( ...
                     fine_y_grid_index, ...
                     fine_x_grid_index)/sqrt(( ...
                     fine_x_coordinate( ...
                     fine_x_grid_index) - ...
                     pos(1,j))^2 + ...
                     (fine_y_coordinate( ...
                     fine_y_grid_index) - ...
                     pos(2,j))^2 + ...
                     epsilon)^magnitude,...
                   fine_y_velocity( ...
                     fine_y_grid_index, ...
                     fine_x_grid_index-1)/sqrt(( ...
                     fine_x_coordinate( ...
                     fine_x_grid_index-1) - ...
                     pos(1,j))^2 + ...
                     (fine_y_coordinate( ...
                     fine_y_grid_index) - ...
                     pos(2,j))^2 + ...
                     epsilon)^magnitude,...
                   fine_y_velocity( ...
                     fine_y_grid_index-1, ...
                     fine_x_grid_index)/sqrt(( ...
                     fine_x_coordinate( ...
                     fine_x_grid_index) - ...
                     pos(1,j))^2 + ...
                     (fine_y_coordinate( ...
                     fine_y_grid_index-1) - ...
                     pos(2,j))^2 + ...
                     epsilon)^magnitude,...
                   fine_y_velocity( ...
                     fine_y_grid_index-1, ...
                     fine_x_grid_index-1)/sqrt(( ...
                     fine_x_coordinate( ...
                     fine_x_grid_index-1) - ...
                     pos(1,j))^2 + ...
                     (fine_y_coordinate( ...
                     fine_y_grid_index-1) - ...
                     pos(2,j))^2 + ...
                     epsilon)^magnitude])/...
                sum([1/sqrt((fine_x_coordinate( ...
                           fine_x_grid_index) - ...
                           pos(1,j))^2 + ...
                           (fine_y_coordinate( ...
                           fine_y_grid_index) - ...
                           pos(2,j))^2 + ...
                           epsilon)^magnitude,...
```

```
                        1/sqrt((fine_x_coordinate( ...
                               fine_x_grid_index-1) - ...
                               pos(1,j))^2 + ...
                               (fine_y_coordinate( ...
                               fine_y_grid_index) - ...
                               pos(2,j))^2 + ...
                               epsilon)^magnitude,...
                        1/sqrt((fine_x_coordinate( ...
                               fine_x_grid_index) - ...
                               pos(1,j))^2 + ...
                               (fine_y_coordinate( ...
                               fine_y_grid_index-1) - ...
                               pos(2,j))^2 + ...
                               epsilon)^magnitude,...
                        1/sqrt((fine_x_coordinate( ...
                               fine_x_grid_index-1) - ...
                               pos(1,j))^2 + ...
                               (fine_y_coordinate( ...
                               fine_y_grid_index-1) - ...
                               pos(2,j))^2 + ...
                               epsilon)^magnitude]);
else
   x_flow   = sum([coarse_x_velocity( ...
                      coarse_y_grid_index, ...
                      coarse_x_grid_index)/sqrt(( ...
                      coarse_x_coordinate( ...
                      coarse_x_grid_index) - ...
                      pos(1,j))^2 + ...
                      (coarse_y_coordinate( ...
                      coarse_y_grid_index) - ...
                      pos(2,j))^2 + ...
                      epsilon)^magnitude,...
                   coarse_x_velocity( ...
                      coarse_y_grid_index, ...
                      coarse_x_grid_index-1)/sqrt(( ...
                      coarse_x_coordinate( ...
                      coarse_x_grid_index-1) - ...
                      pos(1,j))^2 + ...
                      (coarse_y_coordinate( ...
                      coarse_y_grid_index) - ...
                      pos(2,j))^2 + ...
                      epsilon)^magnitude,...
                   coarse_x_velocity( ...
                      coarse_y_grid_index-1, ...
                      coarse_x_grid_index)/sqrt(( ...
                      coarse_x_coordinate( ...
                      coarse_x_grid_index) - ...
                      pos(1,j))^2 + ...
                      (coarse_y_coordinate( ...
                      coarse_y_grid_index-1) - ...
                      pos(2,j))^2 + ...
                      epsilon)^magnitude,...
                   coarse_x_velocity( ...
                      coarse_y_grid_index-1, ...
                      coarse_x_grid_index-1)/sqrt(( ...
                      coarse_x_coordinate( ...
                      coarse_x_grid_index-1) - ...
```

```
                        pos(1,j))^2 + ...
                        (coarse_y_coordinate( ...
                        coarse_y_grid_index-1) - ...
                        pos(2,j))^2 + ...
                        epsilon)^magnitude])/...
                sum([1/sqrt((coarse_x_coordinate( ...
                            coarse_x_grid_index) - ...
                            pos(1,j))^2 + ...
                            (coarse_y_coordinate( ...
                            coarse_y_grid_index) - ...
                            pos(2,j))^2 + ...
                            epsilon)^magnitude,...
                     1/sqrt((coarse_x_coordinate( ...
                            coarse_x_grid_index-1) - ...
                            pos(1,j))^2 + ...
                            (coarse_y_coordinate( ...
                            coarse_y_grid_index) - ...
                            pos(2,j))^2 + ...
                            epsilon)^magnitude,...
                     1/sqrt((coarse_x_coordinate( ...
                            coarse_x_grid_index) - ...
                            pos(1,j))^2 + ...
                            (coarse_y_coordinate( ...
                            coarse_y_grid_index-1) - ...
                            pos(2,j))^2 + ...
                            epsilon)^magnitude,...
                     1/sqrt((coarse_x_coordinate( ...
                            coarse_x_grid_index-1) - ...
                            pos(1,j))^2 + ...
                            (coarse_y_coordinate( ...
                            coarse_y_grid_index-1) - ...
                            pos(2,j))^2 + ...
                            epsilon)^magnitude]);
y_flow   = sum([coarse_y_velocity( ...
                    coarse_y_grid_index, ...
                    coarse_x_grid_index)/sqrt(( ...
                    coarse_x_coordinate( ...
                    coarse_x_grid_index) - ...
                    pos(1,j))^2 + ...
                    (coarse_y_coordinate( ...
                    coarse_y_grid_index) - ...
                    pos(2,j))^2 + ...
                    epsilon)^magnitude,...
                coarse_y_velocity( ...
                    coarse_y_grid_index, ...
                    coarse_x_grid_index-1)/sqrt(( ...
                    coarse_x_coordinate( ...
                    coarse_x_grid_index-1) - ...
                    pos(1,j))^2 + ...
                    (coarse_y_coordinate( ...
                    coarse_y_grid_index) - ...
                    pos(2,j))^2 + ...
                    epsilon)^magnitude,...
                coarse_y_velocity( ...
                    coarse_y_grid_index-1, ...
                    coarse_x_grid_index)/sqrt(( ...
                    coarse_x_coordinate( ...
```

```
                        coarse_x_grid_index) - ...
                        pos(1,j))^2 + ...
                        (coarse_y_coordinate( ...
                        coarse_y_grid_index-1) - ...
                        pos(2,j))^2 + ...
                        epsilon)^magnitude,...
                    coarse_y_velocity( ...
                        coarse_y_grid_index-1, ...
                        coarse_x_grid_index-1)/sqrt(( ...
                        coarse_x_coordinate( ...
                        coarse_x_grid_index-1) - ...
                        pos(1,j))^2 + ...
                        (coarse_y_coordinate( ...
                        coarse_y_grid_index-1) - ...
                        pos(2,j))^2 + ...
                        epsilon)^magnitude])/...
                sum([1/sqrt((coarse_x_coordinate( ...
                            coarse_x_grid_index) - ...
                            pos(1,j))^2 + ...
                            (coarse_y_coordinate( ...
                            coarse_y_grid_index) - ...
                            pos(2,j))^2 + ...
                            epsilon)^magnitude,...
                    1/sqrt((coarse_x_coordinate( ...
                            coarse_x_grid_index-1) - ...
                            pos(1,j))^2 + ...
                            (coarse_y_coordinate( ...
                            coarse_y_grid_index) - ...
                            pos(2,j))^2 + ...
                            epsilon)^magnitude,...
                    1/sqrt((coarse_x_coordinate( ...
                            coarse_x_grid_index) - ...
                            pos(1,j))^2 + ...
                            (coarse_y_coordinate( ...
                            coarse_y_grid_index-1) - ...
                            pos(2,j))^2 + ...
                            epsilon)^magnitude,...
                    1/sqrt((coarse_x_coordinate( ...
                            coarse_x_grid_index-1) - ...
                            pos(1,j))^2 + ...
                            (coarse_y_coordinate( ...
                            coarse_y_grid_index-1) - ...
                            pos(2,j))^2 + ...
                            epsilon)^magnitude]);
end

u_flow      = [x_flow; y_flow];
v_b         = u_flow + v_random;

j           = j + 1;
t           = j*dt;
pos(:,j)    = pos(:,j-1) + v_b*dt;
r(j)        = sqrt((pos(1,j) - alga_center(1))^2 + ...
                (pos(2,j) - alga_center(2))^2);

if r(j)          <= a;
   v_b            = -v_b;
```

```
   pos(:,j)       = pos(:,j-1) + v_b*dt;
elseif pos(1,j)  <= coarse_x_coordinate(2);
   pos(1,j)       = coarse_x_coordinate(2);
elseif pos(2,j)  <= coarse_y_coordinate(2);
   pos(2,j)       = coarse_y_coordinate(2);
elseif pos(2,j)  >= coarse_y_coordinate(length( ...
                       coarse_y_coordinate) - 1);
   pos(2,j)       = coarse_y_coordinate(length( ...
                       coarse_y_coordinate) - 1);
end
r(j)              = sqrt((pos(1,j) - ...
                       alga_center(1))^2 + ...
                       (pos(2,j) - alga_center(2))^2);

if pos(1,j) > coarse_x_coordinate(end-1);
   break;
elseif pos(1,j)   > fine_x_coordinate(2) & ...
       pos(1,j)   < fine_x_coordinate(end-1) & ...
       pos(2,j)   > fine_y_coordinate(2) & ...
       pos(2,j)   < fine_y_coordinate(end-1);
   fine_x_grid_index  = ceil((pos(1,j)- ...
                            fine_x_coordinate(1))/ ...
                            fine_x_grid_step);
   fine_y_grid_index  = ceil((pos(2,j)- ...
                            fine_y_coordinate(1))/ ...
                            fine_y_grid_step);
   C(j)  = sum([fine_concentration(fine_y_grid_index, ...
                   fine_x_grid_index)/sqrt(( ...
                   fine_x_coordinate( ...
                   fine_x_grid_index) - ...
                   pos(1,j))^2 + (fine_y_coordinate( ...
                   fine_y_grid_index) - pos(2,j))^2 + ...
                   epsilon)^magnitude,...
                fine_concentration(fine_y_grid_index, ...
                   fine_x_grid_index-1)/sqrt(( ...
                   fine_x_coordinate( ...
                   fine_x_grid_index-1) - ...
                   pos(1,j))^2 + (fine_y_coordinate( ...
                   fine_y_grid_index) - pos(2,j))^2 + ...
                   epsilon)^magnitude,...
                fine_concentration( ...
                   fine_y_grid_index-1, ...
                   fine_x_grid_index)/sqrt(( ...
                   fine_x_coordinate( ...
                   fine_x_grid_index) - pos(1,j))^2 + ...
                   (fine_y_coordinate( ...
                   fine_y_grid_index-1) - ...
                   pos(2,j))^2 + epsilon)^magnitude,...
                fine_concentration( ...
                   fine_y_grid_index-1, ...
                   fine_x_grid_index-1)/sqrt(( ...
                   fine_x_coordinate( ...
                   fine_x_grid_index-1) - ...
                   pos(1,j))^2 + (fine_y_coordinate( ...
                   fine_y_grid_index-1) - ...
                   pos(2,j))^2 + epsilon)^magnitude])/...
             sum([1/sqrt((fine_x_coordinate( ...
```

```
                        fine_x_grid_index) - ...
                        pos(1,j))^2 + ...
                        (fine_y_coordinate( ...
                        fine_y_grid_index) - ...
                        pos(2,j))^2 + ...
                        epsilon)^magnitude,...
                    1/sqrt((fine_x_coordinate( ...
                        fine_x_grid_index-1) - ...
                        pos(1,j))^2 + ...
                        (fine_y_coordinate( ...
                        fine_y_grid_index) - ...
                        pos(2,j))^2 + ...
                        epsilon)^magnitude,...
                    1/sqrt((fine_x_coordinate( ...
                        fine_x_grid_index) - ...
                        pos(1,j))^2 + ...
                        (fine_y_coordinate( ...
                        fine_y_grid_index-1) - ...
                        pos(2,j))^2 + ...
                        epsilon)^magnitude,...
                    1/sqrt((fine_x_coordinate( ...
                        fine_x_grid_index-1) - ...
                        pos(1,j))^2 + ...
                        (fine_y_coordinate( ...
                        fine_y_grid_index-1) - ...
                        pos(2,j))^2 + ...
                        epsilon)^magnitude]);
else
   coarse_x_grid_index  = ceil((pos(1,j))/ ...
                              coarse_x_grid_step);
   coarse_y_grid_index  = ceil((pos(2,j))/ ...
                              coarse_y_grid_step);
   C(j)  = sum([coarse_concentration( ...
                  coarse_y_grid_index, ...
                  coarse_x_grid_index)/sqrt(( ...
                  coarse_x_coordinate( ...
                  coarse_x_grid_index) - ...
                  pos(1,j))^2 + ...
                  (coarse_y_coordinate( ...
                  coarse_y_grid_index) - ...
                  pos(2,j))^2 + epsilon)^magnitude,...
               coarse_concentration( ...
                  coarse_y_grid_index, ...
                  coarse_x_grid_index-1)/sqrt(( ...
                  coarse_x_coordinate( ...
                  coarse_x_grid_index-1) - ...
                  pos(1,j))^2 + ...
                  (coarse_y_coordinate( ...
                  coarse_y_grid_index) - ...
                  pos(2,j))^2 + epsilon)^magnitude,...
               coarse_concentration( ...
                  coarse_y_grid_index-1, ...
                  coarse_x_grid_index)/sqrt(( ...
                  coarse_x_coordinate( ...
                  coarse_x_grid_index) - ...
                  pos(1,j))^2 + ...
                  (coarse_y_coordinate( ...
```

```
                        coarse_y_grid_index-1) - ...
                        pos(2,j))^2 + epsilon)^magnitude,...
                    coarse_concentration( ...
                        coarse_y_grid_index-1, ...
                        coarse_x_grid_index-1)/sqrt(( ...
                        coarse_x_coordinate( ...
                        coarse_x_grid_index-1) - ...
                        pos(1,j))^2 + ...
                        (coarse_y_coordinate( ...
                        coarse_y_grid_index-1) - ...
                        pos(2,j))^2 + epsilon)^magnitude])/...
                sum([1/sqrt((coarse_x_coordinate( ...
                        coarse_x_grid_index) - ...
                        pos(1,j))^2 + ...
                        (coarse_y_coordinate( ...
                        coarse_y_grid_index) - ...
                        pos(2,j))^2 + ...
                        epsilon)^magnitude,...
                    1/sqrt((coarse_x_coordinate( ...
                        coarse_x_grid_index-1) - ...
                        pos(1,j))^2 + ...
                        (coarse_y_coordinate( ...
                        coarse_y_grid_index) - ...
                        pos(2,j))^2 + ...
                        epsilon)^magnitude,...
                    1/sqrt((coarse_x_coordinate( ...
                        coarse_x_grid_index) - ...
                        pos(1,j))^2 + ...
                        (coarse_y_coordinate( ...
                        coarse_y_grid_index-1) - ...
                        pos(2,j))^2 + ...
                        epsilon)^magnitude,...
                    1/sqrt((coarse_x_coordinate( ...
                        coarse_x_grid_index-1) - ...
                        pos(1,j))^2 + ...
                        (coarse_y_coordinate( ...
                        coarse_y_grid_index-1) - ...
                        pos(2,j))^2 + ...
                        epsilon)^magnitude]);
    end

    dCdt(j)      = (C(j) - C(j-1))/dt;
    dPbdt(j)     = K_D/((K_D + C(j))^2)*dCdt(j);
    avedPbdt(j) = dPconst*avedPbdt(j-1) + dPbdt(j)*dt/T_m;
    lntau        = lntau_0 + alpha*avedPbdt(j);
    tau          = exp(lntau);

end

data(i+(h-1)*bacteria).R = real(r);
data(i+(h-1)*bacteria).S = real(pos(2,:));
data(i+(h-1)*bacteria).X = real(pos(1,:));
data(i+(h-1)*bacteria).Y = real(pos(2,:));
data(i+(h-1)*bacteria).Conc = real(C);
data(i+(h-1)*bacteria).T = real([0:dt:t-dt]);
clear r pos C t dCdt dPbdt avedPbdt;
```

```
        end

    end

    cd(char(cellstr(save_folder(g))));
    save(char(cellstr(save_matrix(g))),'data');

end
```

A.2. Particle Tracking Algorithms

Particle-by-Particle, No Velocity

Basic Rule:

1. Match one particle in frame n with the closest particle in frame n+1. Arbitrarily choose the next particle to match in frame n.
2. Continue matching until there are no more particles in frame n+1. Trajectories end for unmatched particles in frame n.
3. There is no need to define a search radius and the distance between particle matches is minimized on a particle-by-particle basis.

All Particles at Once, No Velocity

Basic Rule:

1. Match all particles in frame n with the closest particles in frame n+1. Find the best matches for particles in frame n over all possibilities.
2. Trajectories end for unmatched particles in frame n.
3. There is no need to define a search radius and the distance between particle matches is minimized over all possibilities

Conservative Search Radius

Basic Rule:

1. Each child (particle in frame n+1) comes from one parent (particle in frame n).
2. Each parent begets one child.

Algorithm:

1. Locate particles in frame n (parent particles) and frame n+1 (child particles).
2. Define a hard cutoff radius (the search radius): R, the maximum distance an object can move between successive frames.

3. Consider all particle pairs (parent from frame n, child from frame n+1). Compute distances for all pairs. Throw out all pairs whose distance exceeds R.
4. Consider one parent particle at a time. There are three possibilities:
   - 4a. Parent has no children: when no child-particle is within the search radius of that parent (for example the particle could have gone out of focus). The trajectory ends for that parent.
   - 4b. Parent has one and only one child: when one and only one child-particle is within the search radius of that parent. In this case, one needs to check if the child could have more than one parent. There are two possibilities:
     - $4b_1$. Child has only one parent: consider this a match. The parent and the child are the same particle.
     - $4b_2$. Child has more than one parent: end the trajectory of that parent.
   - 4c. Parent has more than one child: when more than one child-particle is within the search radius of that parent. The trajectory ends for that parent. This case is analogous to $4b_2$.
5. After one cycles through all of the parents, there will be some children left without a parent. Those children are new trajectories (for examples, particles coming in focus).
6. Repeat the process for the next exposure starting from Step 2.

Biased Search Radius (Distance and/or Velocity)

Basic Rule:

1. Each child comes from only one parent.
2. Each parent begets only one son.
3. Treat the case in which a parent has more than one child within its search radius, or the vice versa.

Algorithm:

1. Locate particles in frame n and frame n+1. Also, assume one knows the velocity of each particle in frame n
2. Define a hard cutoff radius (the search radius): R, the maximum distance an object can move between successive frames.
3. Consider all particle pairs. Compute distances for all pairs. Throw out all pairs whose distance exceeds R.
4. Consider one parent particle at a time. There are three possibilities:
   - 4a. Parent has no children: when no child-particle is within the search radius of that parent (for example the particle could have gone out of focus). The trajectory ends for that parent.
   - 4b. Parent has one and only one child: when one and only one child-particle is within the search radius of that parent. In this case, one needs to check if the child could have more than one parent. There are two possibilities:

$4b_1$. Child has only one parent: consider this a match. The parent and the child are the same particle.

$4b_2$. Child has more than one parent. Determine the true parent by using one of the following algorithms, chosen by the user:

$4b_{2a}$. The correct parent has a position closest position to the child's position (distance criterion).

$4b_{2b}$. The correct parent has a velocity vector pointing closest to the child's position (velocity criterion). This algorithm does not work in the first two frames, nor does it work if one of the parents does not have a velocity, for example because it just appeared in frame *n*. In such cases, resort to $4b_{2a}$.

$4b_{2c}$. The correct parent satisfies both distance and velocity criterion (dual criterion; more conservative approach). In case there is no such parent, consider that child to be an orphan and start a new trajectory. In case one of the parents does not have a velocity, disregard the velocity criterion and apply only the distance criterion.

4c. Parent has more than one child: when more than one child-particle is within the search radius of that parent. This case is analogous to $4b_2$. Determine the true child by using one of the following algorithms, chosen by the user:

$4b_1$. The correct child has a position closest position to the parent's position (distance criterion).

$4b_2$. The correct child has a velocity vector pointing closest to the parent's position (velocity criterion). This algorithm does not work in the first two frames, nor does it work if one of the parents does not have a velocity, for example because it just appeared in frame *n*. In such cases, resort to $4b_1$.

$4b_3$. The correct child satisfies both distance and velocity criterion (dual criterion; more conservative approach). In case there is no such child, the trajectory ends for that parent. In case one of the parents does not have a velocity, disregard the velocity criterion and apply only the distance criterion.

5. After one cycles through all of the parents, there will be some children left without a parent. Those children are new trajectories (for examples, particles coming in focus).

6. Repeat the process for the next exposure starting from Step 2.

A.3. Image Analysis Software

A.3.1. Data Reduction

The following code takes data from BacTrack™ and puts it in a format that is easily handled by MATLAB.

```
% Copyright Michael David Sekora, 7 May 2005

clear all; close all;

%-----------------------------------------------------------------------
% DIRECTORY/FILE NAMES - CHANGE ACCORDINGLY
open_dir  = 'C:\TRACKING\04_12_15_IMAGES_1';
open_file = 'm1_im_1_4076_back_5_rad_5.txt';
save_dir  = 'PROCESSED';
save_file = 'm1_im_1_4076.mat';
%-----------------------------------------------------------------------

cd(open_dir);
format long g;
fid = fopen(open_file, 'rt');
if fid < 0
    message = sprintf('Unable to open file %s: %s\n', open_file);
    error(message);
end
number = 0;
while 1
    line = fgets(fid);
    if strcmpi(line(1),'f') == 1
        continue;
    end
    if line == -1
        break;
    end
    line   = str2num(line);
    number = number + 1;
    data(number).x      = line(2:2:end);
    data(number).y      = line(3:2:end);
    data(number).start  = line(1);
    data(number).end    = (length(line) - 1)/2;
    data(number).length = length(line) - 1;
end
if isdir(save_dir)==0
    mkdir(save_dir);
end
cd(save_dir);
save(save_file);
```

A.3.2. Image Analysis

The code presented in this section works in the following manner. First, data pertaining to bacteria trajectories is loaded. Then, restrictions (described in Section 5.3) are applied. Lastly, statistics are displayed and plots are drawn according to user specified parameters.

```
% Copyright Michael David Sekora, 7 May 2005

clear all; close all;

%-----------------------------------------------------------------------
% PARAMETERS - CHANGE ACCORDINGLY
%-----------------------------------------------------------------------

% DIRECTORY/FILE NAMES
open_data_directory       = 'C:\TRACKING\04_12_15_IMAGES_1\PROCESSED';
open_data_file            = 'm1_im_1_4076.mat';
open_processing_directory = 'C:\TRACKING\DATA_ANALYSIS';

% RUN INFORMATION
run_name = 'M1';
run_date = ' ';

% IMAGE PARAMETERS
image_width             = 512;    % width in pixels
image_height            = 512;    % height in pixels
frames_per_sec          = 50;     % camera speed
ref_microns_per_pixel   = 0.125;  % at reference magnification
ref_magnification       = 100;
magnification           = 20;

% RESTRICTION/PROCESSING CONTROL PANEL PARAMETERS
min_traj_duration_in_sec         = 2;
min_for_max_distance_in_microns  = 7;
number_of_median_filter_points   = 6;
min_mean_speed_factor            = 0.75;
run_speed_top_percent            = 10;
detect_tumbles_run_speed_factor  = 0.3;
detect_tumbles_rcd_per_sec       = 1000;
tumble_skip_parameter            = 4;
restrict_tumble_freq_percent     = 10;

% RESTRICTION/PROCESSING CONTROL PANEL: 1 MEANS ON, 0 MEANS OFF
restrict_traj_duration           = 1;
restrict_max_distance            = 1;
apply_median_filter              = 1;
restrict_mean_speed              = 0;
sharpen_run_and_tumble_modes     = 1;
restrict_tumble_freq             = 1;

% STATISTICS DISPLAY CONTROL PANEL: 1 MEANS ON, 0 MEANS OFF
list_traj_tumble_times           = 0;

% TRAJECTORY PLOTTING CONTROL PANEL: 1 MEANS ON, 0 MEANS OFF
```

```
plot_traj_based_on_duration      = 1;
plot_traj_based_on_ave_traj_spd  = 0;
plot_traj_based_on_ave_traj_rcd  = 0;
use_phys_and_nonphys_axes        = 1;

% SPD AND RCD PLOTTING CONTROL PANEL: 1 MEANS ON, 0 MEANS OFF
plot_spd_rcd_all_traj_together   = 0;
plot_spd_rcd_each_traj_separate  = 0;

% HISTOGRAM CONTROL PANEL: 1 MEANS ON, 0 MEANS OFF
duration_histogram   = 1;

% SPD HISTOGRAM CONTROL PANEL: 1 MEANS ON, 0 MEANS OFF
spd_histogram_for_all_traj       = 1;
spd_histogram_for_each_traj      = 0;
ave_spd_histogram_for_each_traj  = 0;

% RCD HISTOGRAM CONTROL PANEL: 1 MEANS ON, 0 MEANS OFF
rcd_histogram_for_all_traj       = 1;
rcd_histogram_for_each_traj      = 0;
ave_rcd_histogram_for_each_traj  = 0;

% TRAJECTORY PLOTTING PARAMETERS
traj_markersize   = 6;
traj_linewidth    = 1.5;

% SPD/RCD PLOTTING PARAMETERS
spd_rcd_markersize   = 12;
spd_rcd_linewidth    = 1.5;
spd_rcd_text_font    = 8;

% HISTOGRAM PARAMETERS
duration_bins  = 50;
spd_bins       = 50;
rcd_bins       = 50;
facecolor      = 'k';
edgecolor      = 'w';

% FONT/COLORING/LABELING PARAMETERS
fontsize       = 14;
title_font     = 16;
traj_coloring  = 'rgbk';

%-------------------------------------------------------------------------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
% RESTRICTION AND ANALYSIS - DO NOT CHANGE ANYTHING
%-------------------------------------------------------------------------

% LOAD DATA FOR ANALYSIS
cd(open_data_directory);
load(open_data_file);
cd(open_processing_directory);

% COMPUTE NEEDED VALUES FROM PARAMETERS
```

```
number_of_traj_colors=length(traj_coloring);
number_of_traj_initially=length([data.length]);
min_traj_duration_in_frames=min_traj_duration_in_sec*frames_per_sec;
microns_per_pixel=ref_microns_per_pixel* ...
                    ref_magnification/magnification;
min_for_max_distance_in_pixels=min_for_max_distance_in_microns/ ...
                                  microns_per_pixel;
detect_tumbles_rcd_per_frame=detect_tumbles_rcd_per_sec/ ...
                                 frames_per_sec;

% RESTRICT BASED ON TRAJECTORY DURATION (FRAME NUMBER)
restrict_traj_duration_turned='off';
if restrict_traj_duration==1
    restrict_traj_duration_turned='on';
    index=find([data.length]>=min_traj_duration_in_frames);
else
    index=[1:1:length([data.length])];
end

% RESTRICT BASED ON MAXIMUM DISTANCE TRAVELED
restrict_max_distance_turned='off';
if restrict_max_distance==1
    restrict_max_distance_turned='on';
    i=1;
    for j=1:1:length(index);
        if max([max(data(index(j)).x)-min(data(index(j)).x),...
                max(data(index(j)).y)-min(data(index(j)).y)])>= ...
                   min_for_max_distance_in_pixels
            temp_index(i)=index(j);
            i=i+1;
        end
    end
    index=temp_index;
    clear temp_index;
end

% APPLY ALON'S MEDIAN FILTER
median_filter_turned='off';
if apply_median_filter==1
    median_filter_turned='on';
    for i=1:1:length(index)
        for j=1:1:(length(data(index(i)).x)- ...
                    (number_of_median_filter_points-1))
            sort_x=sort(data(index(i)).x ...
                      (j:1:(number_of_median_filter_points+j-1)));
            sort_y=sort(data(index(i)).y ...
                      (j:1:(number_of_median_filter_points+j-1)));
            central_x(j)=mean(sort_x ...
                            (2:1:(number_of_median_filter_points-1)));
            central_y(j)=mean(sort_y ...
                            (2:1:(number_of_median_filter_points-1)));
        end
        data(index(i)).x=central_x;
        data(index(i)).y=central_y;
        data(index(i)).length=length(data(index(i)).x);
        data(index(i)).end=data(index(i)).start+ ...
                             data(index(i)).length-1;
```

```
        central_x=[]; central_y=[];
    end
end

% COMPUTE SPD AND RCD
% COMPUTE SPD AND RCD FOR EACH FRAME IN EACH TRAJECTORY
for i=1:1:length(index)
    for j=1:1:(length(data(index(i)).x)-1)
        data(index(i)).spd(j)=sqrt((data(index(i)).x(j+1)- ...
                                    data(index(i)).x(j))^2+ ...
                                   (data(index(i)).y(j+1)- ...
                                    data(index(i)).y(j))^2);
    end
    for j=1:1:(length(data(index(i)).x)-2)
        data(index(i)).rcd(j)=(180/pi)*abs(mod(abs( ...
                                  atan2(data(index(i)).y(j+2)- ...
                                        data(index(i)).y(j+1), ...
                                        data(index(i)).x(j+2)- ...
                                        data(index(i)).x(j+1))- ...
                                  atan2(data(index(i)).y(j+1)- ...
                                        data(index(i)).y(j), ...
                                           data(index(i)).x(j+1)- ...
                                        data(index(i)).x(j)))+ ...
                                  pi,2*pi)-pi);
    end
end
% COMPUTE AVERAGE SPD AND RCD FOR EACH TRAJECTORY
for i=1:1:length(index)
    data(index(i)).average_traj_spd=mean(data(index(i)).spd);
    data(index(i)).average_traj_rcd=mean(data(index(i)).rcd);
end
% COMPUTE AVERAGE SPD AND RCD OVER ALL TRAJECTORIES
global_average_spd=mean([data.spd]);
global_average_rcd=mean([data.rcd]);

% RESTRICT BASED ON MEAN SPEED
restrict_mean_speed_turned='off';
if restrict_mean_speed==1
    restrict_mean_speed_turned='on';
    i=1;
    for j=1:1:length(index)
        if data(index(j)).average_traj_spd>=min_mean_speed_factor*...
                                                global_average_spd
            temp_index(i)=index(j);
            i=i+1;
        end
    end
    index=temp_index;
    clear temp_index;
end

% COMPUTE RUN SPEED
temp_average_traj_spd=fliplr(sort([data(index).average_traj_spd]));
run_speed=mean(temp_average_traj_spd(1:1:floor ...
             (length(temp_average_traj_spd)* ...
             run_speed_top_percent/100)));
clear temp_average_traj_spd;
```

```
% DETECT TUMBLES
for i=1:1:length(index)
    for j=1:1:length(data(index(i)).rcd)
        if data(index(i)).spd(j+1)<=detect_tumbles_run_speed_factor...
                                        *run_speed & ...
           data(index(i)).rcd(j)>=detect_tumbles_rcd_per_frame
            data(index(i)).tumble_value(j)=1;
        else
            data(index(i)).tumble_value(j)=0;
        end
    end
end

% SHARPEN RUN/TUMBLE MODES
sharpen_run_tumble_modes_turned='off';
if sharpen_run_and_tumble_modes==1
    sharpen_run_tumble_modes_turned='on';
    for i=1:1:length(index)
        for tumble_skip_parameter_index=1:1:tumble_skip_parameter
            for k=1:1:(length(data(index(i)).tumble_value)- ...
                        tumble_skip_parameter_index-1)
                if data(index(i)).tumble_value(k)==1 & ...
                   data(index(i)).tumble_value ...
                      (k+tumble_skip_parameter_index+1)==1 & ...
                   sum(data(index(i)).tumble_value ...
                      (k:1:(k+tumble_skip_parameter_index+1)))~= ...
                      tumble_skip_parameter_index+2
                    data(index(i)).tumble_value ...
                       (k:1:(k+tumble_skip_parameter_index+1))=1;
                end
            end
        end
    end
end

% COMPUTE TUMBLE FREQUENCY AND DURATION
for i=1:1:length(index)
    tumble_data=imfeature(bwlabel(data(index(i)).tumble_value), ...
                  'Area','Centroid');
    data(index(i)).number_of_traj_tumbles=length([tumble_data.Area]);
    data(index(i)).traj_tumble_freq=length([tumble_data.Area])/ ...
                                    length(data(index(i)).x);
    data(index(i)).traj_tumble_duration=[tumble_data.Area];
    data(index(i)).average_traj_tumble_duration=mean([tumble_data. ...
                                                    Area]);
    temp_traj_tumble_times=[tumble_data.Centroid];
    data(index(i)).traj_tumble_times=data(index(i)).start+ ...
                                     temp_traj_tumble_times ...
                                     (1:2:length([tumble_data. ...
                                     Centroid]))+1;
end
clear tumble_data temp_traj_tumble_times;

% RESTRICT BASED ON TUMBLE FREQUENCY
restrict_tumble_freq_turned='off';
if restrict_tumble_freq==1
```

```
    restrict_tumble_freq_turned='on';
    temp_traj_tumble_freq=sort([data(index).traj_tumble_freq]);
    i=1;
    for j=1:1:length(index)
        if data(index(j)).traj_tumble_freq ...
              >=temp_traj_tumble_freq(ceil(length ...
                   (temp_traj_tumble_freq)* ...
                   restrict_tumble_freq_percent/100)) & ...
           data(index(j)).traj_tumble_freq ...
              <=temp_traj_tumble_freq(floor(length ...
                   (temp_traj_tumble_freq)* ...
                   (100-restrict_tumble_freq_percent)/100))
            temp_index(i)=index(j);
            i=i+1;
        end
    end
    index=temp_index;
    clear temp_traj_tumble_freq temp_index;
end

% COMPUTE GLOBAL AVERAGE TUMBLE FREQUENCY AND DURATION
global_average_tumble_freq=mean([data(index).traj_tumble_freq]);
global_average_tumble_duration=mean([data(index). ...
                                  traj_tumble_duration]);

%--------------------------------------------------------------------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%--------------------------------------------------------------------

%--------------------------------------------------------------------
% DISPLAY STATISTICS
%--------------------------------------------------------------------

disp([' ']);
disp(['Information about Particle Tracks: ']);
clock_time=clock;
disp(['Date Processed: ' date ', ' num2str(clock_time(4)) ':' ...
   num2str(clock_time(5)) ':' num2str(clock_time(6))]);
disp(['Run Name: ' run_name]);
disp(['Run Date: ' run_date]);
disp([' ']);
disp(['Global Properties: ']);
disp(['Global Average SPD: ' num2str(global_average_spd* ...
   microns_per_pixel*frames_per_sec,'%11.3f') ' (microns/sec)']);
disp(['Global Average RCD: ' num2str(global_average_rcd* ...
   frames_per_sec,'%11.3f') ' (deg/sec)']);
disp(['Run Speed: ' num2str(run_speed*microns_per_pixel* ...
   frames_per_sec,'%11.3f') ' (microns/sec)']);
disp(['Global Average Tumble Frequency: ' num2str ...
   (global_average_tumble_freq* ...
   frames_per_sec,'%11.3f') '(tumbles/sec)']);
disp(['Global Average Tumble Duration: ' num2str ...
   (global_average_tumble_duration/ ...
   frames_per_sec,'%11.3f') ' (sec)']);
disp([' ']);
disp(['Trajectories Tracked: ' ]);
disp(['Trajectories tracked initially: ' num2str ...
```

```
   (number_of_traj_initially)]);
disp(['Trajectories satisfying all restrictions: ' num2str ...
   (length(index))])
disp(['Colors distinguishing trajectories define intervals of ' ...
   num2str(100/number_of_traj_colors) '% maximum value']);
disp(['Color of intervals (in increasing order): ' traj_coloring]);
disp([' ']);
disp(['Restrictions Applied:']);
disp(['Trajectory duration restriction was turned: ' ...
   restrict_traj_duration_turned]);
disp(['Maximum distance traveled restriction was turned: ' ...
   restrict_max_distance_turned]);
disp(['Median filter was turned: ' median_filter_turned]);
disp(['Mean speed restriction was turned: ' ...
   restrict_mean_speed_turned]);
disp(['Tumble frequency restriction was turned: ' ...
   restrict_tumble_freq_turned]);
disp(['Sharpening differences between run/' ...
   'tumble modes was turned: ' sharpen_run_tumble_modes_turned]);
disp([' ']);
disp(['Restriction Parameters: ']);
disp(['Minimum trajectory duration: ' num2str ...
   (min_traj_duration_in_sec) ' (sec)']);
disp(['Minimum value for maximum distance traveled: ' num2str ...
   (min_for_max_distance_in_microns) ' (microns)']);
disp(['Number of median filter points: ' num2str ...
   (number_of_median_filter_points)]);
disp(['Minimum mean speed factor: ' num2str ...
   (min_mean_speed_factor)]);
disp(['Compute run speed by keeping top ' num2str ...
   (run_speed_top_percent) '% from each average trajectory spd']);
disp(['Tumble detected when spd <= ' num2str ...
   (detect_tumbles_run_speed_factor) ' * run speed']);
disp(['Tumble detected when rcd >= ' num2str ...
   (detect_tumbles_rcd_per_sec) ' (deg/sec)']);
disp(['Maximum time segment overlooked when defining run/' ...
   'tumble modes = ' num2str(tumble_skip_parameter/ ...
   frames_per_sec) ' (sec)']);
disp(['Discard top and bottom ' num2str ...
   (restrict_tumble_freq_percent) ...
   '% when computing global-average tumble frequency']);
disp([' ']);
disp(['Image Parameters: ']);
disp(['Image width: ' num2str(image_width) ' (pixels)']);
disp(['Image height: ' num2str(image_height) ' (pixels)']);
disp(['Frame Rate: ' num2str(frames_per_sec) ' (frames/sec)']);
disp(['Magnification: ' num2str(magnification)]);
disp(['Length Scale: ' num2str(microns_per_pixel) ...
   ' (microns/pixel)']);
disp([' ']);
disp(['Minimum tracked duration: ' num2str ...
   (min([data(index).length])/frames_per_sec,'%11.3g') ' (sec)']);
disp(['Maximum tracked duration: ' num2str ...
   (max([data(index).length])/frames_per_sec,'%11.3g') ' (sec)']);
disp(['Average tracked duration: ' num2str ...
   (mean([data(index).length])/frames_per_sec,'%11.3g') ' (sec)']);
disp([' ']);
```

```
disp(['Minimum SPD taken from all trajectories: ' num2str ...
   (min([data(index).spd])*microns_per_pixel*frames_per_sec, ...
   '%11.3f') ' (microns/sec)']);
disp(['Maximum SPD taken from all trajectories: ' num2str ...
   (max([data(index).spd])*microns_per_pixel*frames_per_sec, ...
   '%11.3f') ' (microns/sec)']);
disp(['Average SPD taken from all trajectories: ' num2str ...
   (mean([data(index).spd])*microns_per_pixel*frames_per_sec, ...
   '%11.3f') ' (microns/sec)']);
disp([' ']);
disp(['Minimum SPD taken from SPDs averaged over each trajectory: '...
   num2str(min([data(index).average_traj_spd])*microns_per_pixel* ...
   frames_per_sec,'%11.3f') ' (microns/sec)']);
disp(['Maximum SPD taken from SPDs averaged over each trajectory: '...
   num2str(max([data(index).average_traj_spd])*microns_per_pixel* ...
   frames_per_sec,'%11.3f') ' (microns/sec)']);
disp(['Average SPD taken from SPDs averaged over each trajectory: '...
   num2str(mean([data(index).average_traj_spd])*microns_per_pixel* ...
   frames_per_sec,'%11.3f') ' (microns/sec)']);
disp([' ']);
disp(['Minimum RCD taken from all trajectories: ' num2str ...
   (min([data(index).rcd])*frames_per_sec,'%11.3f') ' (deg/sec)']);
disp(['Maximum RCD taken from all trajectories: ' num2str ...
   (max([data(index).rcd])*frames_per_sec,'%11.3f') ' (deg/sec)']);
disp(['Average RCD taken from all trajectories: ' num2str ...
   (mean([data(index).rcd])*frames_per_sec,'%11.3f') ' (deg/sec)']);
disp([' ']);
disp(['Minimum RCD taken from RCDs averaged over each trajectory: '...
   num2str(min([data(index).average_traj_rcd])*frames_per_sec,...
   '%11.3f') ' (deg/sec)']);
disp(['Maximum RCD taken from RCDs averaged over each trajectory: '...
   num2str(max([data(index).average_traj_rcd])*frames_per_sec, ...
   '%11.3f') ' (deg/sec)']);
disp(['Average RCD taken from RCDs averaged over each trajectory: '...
   num2str(mean([data(index).average_traj_rcd])*frames_per_sec,...
   '%11.3f') ' (deg/sec)']);
disp([' ']);
disp(['Minimum number of tumbles taken from all trajectories: ' ...
   num2str(min([data(index).number_of_traj_tumbles]),'%11.3f')]);
disp(['Maximum number of tumbles taken from all trajectories: ' ...
   num2str(max([data(index).number_of_traj_tumbles]),'%11.3f')]);
disp(['Average number of tumbles taken from all trajectories: ' ...
   num2str(mean([data(index).number_of_traj_tumbles]),'%11.3f')]);
disp([' ']);
disp(['Minimum tumble frequency taken from all tumble frequencies:'...
   num2str(min([data(index).traj_tumble_freq])*frames_per_sec, ...
   '%11.3f') ' (tumbles/sec)']);
disp(['Maximum tumble frequency taken from all tumble frequencies:'...
   num2str(max([data(index).traj_tumble_freq])*frames_per_sec, ...
   '%11.3f') ' (tumbles/sec)']);
disp(['Average tumble frequency taken from all tumble frequencies:'...
   num2str(mean([data(index).traj_tumble_freq])*frames_per_sec, ...
   '%11.3f') ' (tumbles/sec)']);
disp([' ']);
disp(['Minimum tumble duration taken from all tumble durations: ' ...
   num2str(min([data(index).traj_tumble_duration])/frames_per_sec, ...
   '%11.3f') ' (sec)']);
```

```
disp(['Maximum tumble duration taken from all tumble durations: ' ...
   num2str(max([data(index).traj_tumble_duration])/frames_per_sec, ...
   '%11.3f') ' (sec)']);
disp(['Average tumble duration taken from all tumble durations: ' ...
   num2str(mean([data(index).traj_tumble_duration])/frames_per_sec,...
   '%11.3f') ' (sec)']);
disp([' ']);
disp(['Minimum tumble duration taken from tumble durations '...
   'averaged over each trajectory: ' num2str( ...
   min([data(index).average_traj_tumble_duration])/ ...
   frames_per_sec,'%11.3f') ' (sec)']);
disp(['Maximum tumble duration taken from tumble durations ' ...
   'averaged over each trajectory: ' num2str( ...
   max([data(index).average_traj_tumble_duration])/ ...
   frames_per_sec,'%11.3f') ' (sec)']);
disp(['Average tumble duration taken from tumble durations ' ...
   'averaged over each trajectory: ' num2str( ...
   mean([data(index).average_traj_tumble_duration])/ ...
   frames_per_sec,'%11.3f') ' (sec)']);
disp([' ']);
if list_traj_tumble_times==1
    for i=1:1:length(index)
        disp(['Particle Track ' num2str(sprintf(['%04d'], ...
           index(i))) ': tumble frequency = ' num2str( ...
           data(index(i)).traj_tumble_freq*frames_per_sec, ...
           '%11.3f') ' (tumbles/sec), average tumble duration = ' ...
           num2str(data(index(i)).average_traj_tumble_duration/ ...
           frames_per_sec,'%11.3f') ' (sec), time of tumbles: ' ...
           num2str(data(index(i)).traj_tumble_times/ ...
           frames_per_sec,'%11.3f') ' (sec), tumble durations: ' ...
           num2str(data(index(i)).traj_tumble_duration/ ...
           frames_per_sec,'%11.3f') ' (sec)']);
    end
end

%-------------------------------------------------------------------------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-------------------------------------------------------------------------

%-------------------------------------------------------------------------
% PLOTTING
%-------------------------------------------------------------------------

% TRAJECTORY PLOTTING
% PLOT TRAJECTORIES BASED ON DURATION
if plot_traj_based_on_duration==1
    figure;
    if use_phys_and_nonphys_axes==1
        set(gca,'XaxisLocation','bottom','YAxisLocation','right');
        axis([0 image_width 0 image_height]);
        axis ij;
        xlabel(['x (pixels)'],'FontSize',fontsize);
        ylabel(['y (pixels)'],'FontSize',fontsize);
        hold on;
    end
    axes('XAxisLocation','top','YAxisLocation','left');
    for i=1:1:length(index)
```

```
        for j=1:1:number_of_traj_colors
            if data(index(i)).length>(j-1)*max([data.length])/ ...
                  number_of_traj_colors & ...
               data(index(i)).length<=(j)*max([data.length])/ ...
                  number_of_traj_colors
                h=plot(microns_per_pixel*data(index(i)).x(1), ...
                       microns_per_pixel*data(index(i)).y(1),'x-',...
                       microns_per_pixel*data(index(i)).x, ...
                       microns_per_pixel*data(index(i)).y,'-',...
                       'MarkerSize',traj_markersize,'LineWidth', ...
                       traj_linewidth);
                set(h,'Color',traj_coloring(j));
                hold on;
            end
        end
    end
    set(gca,'XAxisLocation','top','YAxisLocation','left');
    axis(microns_per_pixel*[0 image_width 0 image_height]);
    axis ij;
    xlabel(['x (\mum)'],'FontSize',fontsize);
    ylabel(['y (\mum)'],'FontSize',fontsize);
end
% PLOT TRAJECTORIES BASED ON AVERAGE TRAJECTORY SPD
if plot_traj_based_on_ave_traj_spd==1
    figure;
    if use_phys_and_nonphys_axes==1
        set(gca,'XaxisLocation','bottom','YAxisLocation','right');
        axis([0 image_width 0 image_height]);
        axis ij;
        xlabel(['x (pixels)'],'FontSize',fontsize);
        ylabel(['y (pixels)'],'FontSize',fontsize);
        hold on;
    end
    axes('XAxisLocation','top','YAxisLocation','left');
    for i=1:1:length(index)
        for j=1:1:number_of_traj_colors
            if data(index(i)).average_traj_spd>(j-1)*max( ...
                  [data.average_traj_spd])/number_of_traj_colors & ...
               data(index(i)).average_traj_spd<=(j)*max( ...
                  [data.average_traj_spd])/number_of_traj_colors
                h=plot(microns_per_pixel*data(index(i)).x(1), ...
                       microns_per_pixel*data(index(i)).y(1),'x-',...

microns_per_pixel*data(index(i)).x, ...
                       microns_per_pixel*data(index(i)).y,'-',...
                       'MarkerSize',traj_markersize,'LineWidth', ...
                       traj_linewidth);
                set(h,'Color',traj_coloring(j));
                hold on;
            end
        end
    end
    set(gca,'XAxisLocation','top','YAxisLocation','left');
    axis(microns_per_pixel*[0 image_width 0 image_height]);
    axis ij;
    xlabel(['x (\mum)'],'FontSize',fontsize);
    ylabel(['y (\mum)'],'FontSize',fontsize);
```

```
end
% PLOT TRAJECTORIES BASED ON AVERAGE TRAJECTORY RCD
if plot_traj_based_on_ave_traj_rcd==1
    figure;
    if use_phys_and_nonphys_axes==1
        set(gca,'XaxisLocation','bottom','YAxisLocation','right');
        axis([0 image_width 0 image_height]);
        axis ij;
        xlabel(['x (pixels)'],'FontSize',fontsize);
        ylabel(['y (pixels)'],'FontSize',fontsize);
        hold on;
    end
    axes('XAxisLocation','top','YAxisLocation','left');
    for i=1:1:length(index)
        for j=1:1:number_of_traj_colors
            if data(index(i)).average_traj_rcd>(j-1)*max( ...
                  [data.average_traj_rcd])/number_of_traj_colors & ...
               data(index(i)).average_traj_rcd<=(j)*max( ...
                  [data.average_traj_rcd])/number_of_traj_colors
                h=plot(microns_per_pixel*data(index(i)).x(1), ...
                       microns_per_pixel*data(index(i)).y(1),'x-',...
                       microns_per_pixel*data(index(i)).x, ...
                       microns_per_pixel*data(index(i)).y,'-',...
                       'MarkerSize',traj_markersize,'LineWidth', ...
                       traj_linewidth);
                set(h,'Color',traj_coloring(j));
                hold on;
            end
        end
    end
    set(gca,'XAxisLocation','top','YAxisLocation','left');
    axis(microns_per_pixel*[0 image_width 0 image_height]);
    axis ij;
    xlabel(['x (\mum)'],'FontSize',fontsize);
    ylabel(['y (\mum)'],'FontSize',fontsize);
end

% HISTOGRAM BASED ON DURATION
if duration_histogram==1
    figure;
    hist([data(index).length]/frames_per_sec,duration_bins);
    h=findobj(gca,'Type','patch');
    set(h,'FaceColor',facecolor,'EdgeColor',edgecolor);
    %axis([]);
    xlabel(['Trajectory Duration (s)'],'FontSize',fontsize);
    ylabel(['Count'],'FontSize',fontsize);
    title(['Histogram of Trajectory Durations'],'FontSize', ...
       title_font);
end

% SPD AND RCD SUBPLOTTING
% PLOT SPD's AND RCD's FOR ALL TRAJECTORIES TOGETHER
if plot_spd_rcd_all_traj_together==1
    figure;
    for i=1:1:length(index)
        subplot(2,1,1);
        plot([(data(index(i)).start+1):1:data(index(i)).end]/ ...
```

```
            frames_per_sec, [data(index(i)).spd]* ...
            microns_per_pixel*frames_per_sec,'k.-','MarkerSize', ...
            spd_rcd_markersize,'LineWidth',spd_rcd_linewidth);
        hold on;
            subplot(2,1,2);
        plot([(data(index(i)).start+2):1:data(index(i)).end]/ ...
            frames_per_sec,[data(index(i)).rcd]*frames_per_sec, ...
            'k.-','MarkerSize',spd_rcd_markersize,'LineWidth', ...
            spd_rcd_linewidth);
        hold on;
    end
    subplot(2,1,1); axis tight;
    xlabel(['Time (s)'],'FontSize',fontsize);
    ylabel(['SPD (\mum/s)'],'FontSize',fontsize);
    subplot(2,1,2); axis tight;
    xlabel(['Time (s)'],'FontSize',fontsize);
    ylabel(['RCD (deg/s)'],'FontSize',fontsize);
    title(['All Trajectories'],'FontSize',title_font);
end
% PLOT SPD's AND RCD's FOR ALL TRAJECTORIES SEPARATELY
if plot_spd_rcd_each_traj_separate==1
    for i=1:1:length(index)
        figure;
        axes('position',[0.1 0.1 0.8 0.325]);
        plot([(data(index(i)).start+2):1:data(index(i)).end]/ ...
            frames_per_sec,[data(index(i)).rcd]*frames_per_sec, ...
            'k.-','MarkerSize',spd_rcd_markersize,'LineWidth', ...
            spd_rcd_linewidth);
        axis tight;
        set(gca,'XLim',[(data(index(i)).start+1) ...
           data(index(i)).end]/frames_per_sec);
        ylabel(['RCD (deg/s)'],'FontSize',fontsize);
        xlabel(['Time (s)'],'FontSize',fontsize);
        axes('position',[0.1 0.475 0.8 0.325]);
        plot([(data(index(i)).start+1):1:data(index(i)).end]/ ...
            frames_per_sec,[data(index(i)).spd]*microns_per_pixel*...
            frames_per_sec,'k.-','MarkerSize',spd_rcd_markersize, ...
            'LineWidth',spd_rcd_linewidth);
        axis tight;
        set(gca,'XLim',[(data(index(i)).start+1)
data(index(i)).end]/frames_per_sec);
        ylabel(['SPD (\mum/s)'],'FontSize',fontsize);
        axes('position',[0.1 0.825 0.15 0.15]);
        plot(data(index(i)).x(1),data(index(i)).y(1),'kx-',...
            data(index(i)).x,data(index(i)).y,'k-',...
            'MarkerSize',traj_markersize,'LineWidth',traj_linewidth);
        axis([0 image_width 0 image_height]);
        axis ij;
        axis off;
        axes('position',[0.3 0.825 0.4 0.15]);
        text(0.15,0,['Particle Track ' num2str(index(i))], ...
            'FontSize',spd_rcd_text_font);
        text(0.15,(1+1/6)/8,['Average trajectory SPD: ' ...
            num2str(data(index(i)).average_traj_spd* ...
            microns_per_pixel*frames_per_sec, ...
            '%11.3f') ' (\mum/s)'],'FontSize',spd_rcd_text_font);
        text(0.15,(2+2/6)/8,['Average trajectory RCD: ' num2str( ...
```

```
            data(index(i)).average_traj_rcd*frames_per_sec, ...
            '%11.3f') ' (deg/s)'],'FontSize',spd_rcd_text_font);
        text(0.15,(3+3/6)/8,['Number of tumbles during trajectory:'...
            num2str(data(index(i)).number_of_traj_tumbles, ...
            '%11.3f')],'FontSize',spd_rcd_text_font);
        text(0.15,(4+4/6)/8,['Tumble Frequency = ' num2str( ...
            data(index(i)).traj_tumble_freq*frames_per_sec, ...
            '%11.3f') ' (tumbles/s)'],'FontSize',spd_rcd_text_font);
        text(0.15,(5+5/6)/8,['Average Tumble Duration = ' num2str( ...
            data(index(i)).average_traj_tumble_duration/ ...
            frames_per_sec,'%11.3f') ' (s)'],'FontSize', ...
            spd_rcd_text_font);
        text(0.15,7/8,['Trajectory Duration: ' num2str( ...
            data(index(i)).length/frames_per_sec, ...
            '%11.3g') ' (s)'],'FontSize',spd_rcd_text_font);
        axis([0 1 0 1]);
        axis ij;
        axis off;
        axes('position',[0.75 0.825 0.15 0.15]);
        plot(data(index(i)).x(1),data(index(i)).y(1),'kx-',...
            data(index(i)).x,data(index(i)).y,'k-',...
            'MarkerSize',traj_markersize,'LineWidth', ...
            traj_linewidth);
        axis([min(data(index(i)).x)-5 max(data(index(i)).x)+5 ...
             min(data(index(i)).y)-5 max(data(index(i)).y)+5]);
        axis ij;
        axis off;
    end
end

% SPD HISTOGRAMS
% HISTOGRAM BASED ON SPD's FOR ALL TRAJECTORIES
if spd_histogram_for_all_traj==1
    figure;
    hist([data(index).spd]*microns_per_pixel*frames_per_sec,spd_bins);
    h=findobj(gca,'Type','patch');
    set(h,'FaceColor',facecolor,'EdgeColor',edgecolor);
    %axis([]);
    xlabel(['SPD (\mum/s)'],'FontSize',fontsize);
    ylabel(['Count'],'FontSize',fontsize);
    title(['All Trajectories'],'FontSize',title_font);
end
% HISTOGRAM BASED ON SPD's FOR EACH TRAJECTORY
if spd_histogram_for_each_traj==1
    for i=1:1:length(index)
        figure;
        hist([data(index(i)).spd]*microns_per_pixel* ...
            frames_per_sec,spd_bins);
        h=findobj(gca,'Type','patch');
        set(h,'FaceColor',facecolor,'EdgeColor',edgecolor);
        %axis([]);
        xlabel(['SPD (\mum/s)'],'FontSize',fontsize);
        ylabel(['Count'],'FontSize',fontsize);
        title(['Particle Track ' num2str(index(i))],'FontSize', ...
             title_font);
    end
end
```

```
% HISTOGRAM BASED ON AVERAGE SPD FOR EACH TRAJECTORY
if ave_spd_histogram_for_each_traj==1
    figure;
    hist([data(index).average_traj_spd]*microns_per_pixel* ...
         frames_per_sec,spd_bins);
    h=findobj(gca,'Type','patch');
    set(h,'FaceColor',facecolor,'EdgeColor',edgecolor);
    %axis([]);
    xlabel(['Average SPD (\mum/s)'],'FontSize',fontsize);
    ylabel(['Count'],'FontSize',fontsize);
    title(['Averages Taken for Each Trajectory'],'FontSize', ...
          title_font);
end

% RCD HISTOGRAMS
% HISTOGRAM BASED ON RCD's FOR ALL TRAJECTORIES
if rcd_histogram_for_all_traj==1
    figure;
    hist([data(index).rcd]*frames_per_sec,rcd_bins);
    h=findobj(gca,'Type','patch');
    set(h,'FaceColor',facecolor,'EdgeColor',edgecolor);
    %axis([]);
    xlabel(['RCD (deg/s)'],'FontSize',fontsize);
    ylabel(['Count'],'FontSize',fontsize);
    title(['All Trajectories'],'FontSize',title_font);
end
% HISTOGRAM BASED ON RCD's FOR EACH TRAJECTORY
if rcd_histogram_for_each_traj==1
    for i=1:1:length(index)
        figure;
        hist([data(index(i)).rcd]*frames_per_sec,rcd_bins);
        h=findobj(gca,'Type','patch');
        set(h,'FaceColor',facecolor,'EdgeColor',edgecolor);
        %axis([]);
        xlabel(['RCD (deg/s)'],'FontSize',fontsize);
        ylabel(['Count'],'FontSize',fontsize);
        title(['Particle Track ' num2str(index(i))],'FontSize', ...
              title_font);
    end
end
% HISTOGRAM BASED ON AVERAGE RCD FOR EACH TRAJECTORY
if ave_rcd_histogram_for_each_traj==1
    figure;
    hist([data(index).average_traj_rcd]*frames_per_sec,rcd_bins);
    h=findobj(gca,'Type','patch');
    set(h,'FaceColor',facecolor,'EdgeColor',edgecolor);
    %axis([]);
    xlabel(['Average RCD (deg/s)'],'FontSize',fontsize);
    ylabel(['Count'],'FontSize',fontsize);
    title(['Averages Taken for Each Trajectory'],'FontSize', ...
          title_font);
end

%-----------------------------------------------------------------------
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%-----------------------------------------------------------------------
```