A PARTICLE-IN-CELL FRAMEWORK FOR THE SIMULATION OF CELLULAR CHEMOTAXIS

BY

MATTHEW T. WROBEL

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2011

Urbana, Illinois

Master's Committee:

Associate Professor Luke Olson

Abstract

Cellular chemotaxis is the motion of biological cells in response to a chemical—namely its gradient. In particular, the study of bacterial chemotaxis and immune-cell chemotaxis are of interest in immunology. A possible framework for modelling cellular chemotaxis involves simultaneously advancing the chemical's fluid equations using a finite difference approach, and a particle model of the cellular motion, to which they are coupled. Results in the limited scope of bacterial (flagellated) chemotaxis show the framework to be viable. Further research is warranted to validate current components and extend the framework's capabilities through new components.

For my brother, who loves theses.

Acknowledgments

This project was made possible by the department of Chemical and Biomolecular Engineering, Assistant professor Christopher V. Rao, and the Rao Lab through the provision of funding*, motivation, and guidance for such research. Special thanks to Yuki Kimura for assistance in *E. coli* bacteria chemotactic theory and modelling.

*This project was funded in part by research grant GM083601 from the National Institutes of Health.

Table of Contents

List of Abbreviations

FD	Finite-Difference.
FTCS	Forward-Time Centered-Space.
OEH	Odd-Even Hopscotch.
PIC	Particle-in-Cell.

List of Symbols

t	Time
D	Diffusion constant
$\vec{r}$	Position vector
$\phi(\vec{r}, t)$	Time-dependent scalar chemical concentration field
$\phi^n_{i,j,k}$	Scalar concentration at timestep n and at grid point on indices i, j, and k
m	Particle index (unique label for each particle)
$\vec{r}_m$	Position of particle m
v	Particle speed (scalar)
$\hat{v}_m$	Unit vector representing particle's forward direction
$\hat{u}_m$	Additional unit vector, orthogonal to $\hat{v}_m$, fully determining particle's orientation
$s_1, s_2, \ldots$	Independent, uniform random variables on $[0, 1)$

Chapter 1

Goals

Cellular chemotaxis is the motion of a cell in response to a chemical. The chemical may be a nutrient, attractant, indicator for extracellular communication etc. The presence of such a chemical alters the behavior and movement of the cell, namely causing it to move along or against the chemical's concentration gradient. For example, an immune cell may be attracted to an infection by the metabolic by-products of bacteria, or by secondary indicators secreted by tissues under attack. To view cellular chemotaxis in a fluid, mathematically, involves the time-evolution of the chemoattractant (or repellant) distribution and the time-evolution of the cell distribution. To model both aspects requires concurrently advancing two sets of coupled equations. One set of equations describes the chemical density distribution and another set describes the cellular motion. Historically, with a high enough cell density, there is a precedent to model the cells as a continuous fluid as in the Keller-Segel model [3]. However, to investigate the effects of various suspected internal mechanisms and chemotactic behaviors it could be more revealing to treat each cell as a discrete particle and directly model individual behavior. Resolving individual behavior could be advantageous, as certain bacteria, such as *S. Tyhpi* and *E. Coli* not only respond to environmental attractants (such as nutrients) but to attractants self-excreted under certain circumstances. The resulting patterns and distributions of bacteria can be intricate, exhibiting features not resemblant of a continuous fluid, such as clumping [6]. It is also beneficial to have the capability to add a chemoattractant, a new behavior in response to it, or an additional new class of cells with entirely different locomotion and secretions, without constructing an entirely new model.

Significant advancements in computing capabilities since the development of the Keller-Segel model also make direct simulation of large populations feasible. Many tools that already exist to model chemotaxis occupy a great deal of time simulating the internal mechanisms of the cells such as individual receptors and flagellar motors, or are general purpose biophysical simulators [2]. The framework proposed herein specifically targets chemotaxis in fluid and favors parametrization of internal mechanisms in order to support large populations of cells suitable for generating statistics, while still capturing individual behavior.

The chemical distribution is considered as a scalar concentration (or density) field. The field evolves in time according to usual fluid equations, but the simplest case is diffusion only. While there may be

warrant for capturing advection for scenarios in vivo, the simplest circumstances in vitro do not involve any significant flow. So, the scalar concentration field evolves according to the following equation:

$$\frac{\partial \phi(\vec{r},t)}{\partial t} = D\nabla^2\phi(\vec{r},t) + f(\vec{r},t) \tag{1.1}$$

The function $f(\vec{r},t)$ represents the global superposition of all the local additions or subtractions to the chemical concentration by individual cells (through secretion or ingestion); $f(\vec{r},t) = \sum_m f_m(\vec{r},t)$. The f_m represent the individual addition/subtraction by a single cell. Consider these "shape" functions to be delta functions $f_m(\vec{r},t) = E_m\delta(\vec{r}-\vec{r}_m)$. The delta function integrates out to a quantity-per-time of the chemical, E_m (and not a concentration-per-time). The delta function also reflects an infinitesimal particle. This choice of shape function would, however, assuredly cause the solution to become locally negative in the continuous case. Once discretized, the issue becomes moot, since finite-size bacteria would fit inside a grid cell with a reasonable choice of grid spacing, thereby placing a lower bound on the apparent "size" of the shape function to be that of one grid cell. The classes of particles used here are all nutrient-consuming (or "ingesting") particles, but the exact mechanics of ingested chemoattractants (such as nutrients) and consumption rates are not well-discussed in the context of chemotaxis. A simple linear dependence on concentration is assumed:

$$E_m = C\phi(\vec{r}_m,t) \tag{1.2}$$

In general, the fundamental concept in chemotaxis is that a particle attempts to follow a chemical gradient. Of the few types of test particles used in development and debugging, one type of particle satisfies that very basic requirement. For the gradient-following test particle, the position updates as follows:

$$\frac{\partial \vec{r}_m}{\partial t} = v\frac{\nabla\phi(\vec{r}_m,t)}{\varepsilon + ||\nabla\phi(\vec{r}_m,t)||} \tag{1.3}$$

where ε is a very small number to prevent division-by-zero. The result is a particle that "swims" directly up the gradient at an essentially fixed speed. This overly simplified particle model is not suitable for any simulation beyond testing. Henceforth this type of particle will simply be referred to as a test particle.

For bacterial particles (with flagellated locomotion) a set of equations that describes two alternate behaviors—running (straight) and tumbling—must be used. A tumble amounts to a random reorientation, and a run is an approximately straight movement. For example, a bacterium can indirectly follow a gradient by remaining straight more often when concentration is seen increasing, and reorienting more often

when decreasing, as described by:

$$(\frac{\partial \vec{r}_m}{\partial t}, \frac{\partial \hat{v}_m}{\partial t}) = \begin{cases} (0, \omega\hat{\omega} \times \hat{v}_m) & \text{if } s_1 < p_m; \\ (v\hat{v}_m, 0) & \text{otherwise.} \end{cases} \tag{1.4}$$

The unit vector $\hat{v}_m$ represents the particle's forward direction. The upper portion of (1.4) represents a tumble, and the bottom portion represents a run, where p is the probability of finding a particle tumbling, and s_1 is a uniform random variable on $[0, 1)$ Additionally, ω is an angular velocity with unknown random distribution. There is not an adequate basis for p and ω. On the other hand, there are measurements of total angle of deviation from a tumble, and it has been observed that the runs and tumbles are a Poisson process, and thus an exponential distribution of run durations can be seen [1]. With that in mind, the following equations can be constructed:

$$q_m = \frac{1}{(1 + \phi(\vec{r}_m, t))^2} \frac{\partial \phi(\vec{r}_m, t)}{\partial t} \tag{1.5}$$

$$p[k > 0]_{\Delta t} = 1 - e^{-\Delta t \lambda e^{-\alpha q_m}} \tag{1.6}$$

where q describes the sensitivity of the particle to time-changes in concentration, with the first term in (1.5) regarding receptor binding [2]. Equation (1.6) describes the probability of a tumble occurring in a time interval of duration Δt (k is the number of tumble events in the interval) with a baseline tumble rate per unit-time of λ. Since the time step will be discretized, there is no additional challenge posed by these measurements, as will be seen in the next chapter.

The goals in developing a framework to model chemotaxis are to treat the two sets of equations—fluid and particle—in a manner that allows changes to each set with minimal changes to the framework, and the ability to model additional types of biological cells and chemicals alongside and interacting with one another. To meet these goals, an object-oriented framework was constructed in C++. The framework supports any number of scalar or vector fields, updaters (to advance the fields) and interpolators and depositors (to couple the particle equations to the fields) and any number particle sets. In the following section the presently-employed numerical methods and discretizations in the framework are introduced. Of course, besides being configurable (scriptable) and extensible, the framework must demonstrate the capability to reasonably model basic experiments. Results are discussed in the third chapter.

Chapter 2

Methods

Many methods exist to advance fluid equations in time. Because, in this framework, the fluid equations must be advanced alongside and coupled to a particle model which is often stochastic, an inexpensive method is desirable. The stochastic process precludes large time stepping, and the fluid equations should be updated roughly as often as the particles—in this framework, the fluid equations are updated at least as often. For time discretization, a simple forward Euler timestep is used. Two finite-difference schemes are presently supported, Forward-Time Centered-Space (FTCS) and Odd-Even Hopscotch (OEH, shown below):

for all $(i+j+k+n)$ even:

$$\phi^{n+1}_{i,j,k} = \phi^{n}_{i,j,k} + D\Delta t\left(\frac{\phi^{n}_{i+1,j,k} - 2\phi^{n}_{i,j,k} + \phi^{n}_{i-1,j,k}}{\Delta x^2} + \frac{\phi^{n}_{i,j+1,k} - 2\phi^{n}_{i,j,k} + \phi^{n}_{i,j-1,k}}{\Delta y^2} + \frac{\phi^{n}_{i,j,k+1} - 2\phi^{n}_{i,j,k} + \phi^{n}_{i,j,k-1}}{\Delta z^2}\right) \tag{2.1}$$

for all $(i+j+k+n)$ odd:

$$\phi^{n+1}_{i,j,k} = \frac{\phi^{n}_{i,j,k} + D\Delta t\left(\frac{\phi^{n+1}_{i+1,j,k}+\phi^{n+1}_{i-1,j,k}}{\Delta x^2} + \frac{\phi^{n+1}_{i,j+1,k}+\phi^{n+1}_{i,j-1,k}}{\Delta y^2} + \frac{\phi^{n+1}_{i,j,k+1}+\phi^{n+1}_{i,j,k-1}}{\Delta z^2}\right)}{1 + D\Delta t\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2} + \frac{1}{\Delta z^2}\right)} \tag{2.2}$$

finally, for all i, j, k :

$$\phi^{n+1}_{i,j,k} := \phi^{n+1}_{i,j,k} + \sum_m (f_m)^n_{i,j,k} \tag{2.3}$$

FTCS is simply (2.1) for all grid points. OEH is unconditionally stable for the diffusion equation, which allows increasing the grid resolution without restricting the time step (and without necessarily gaining any additional accuracy in diffusion, as truncation error will still be dependent on a Δt term) [4]. This may be useful for reducing grid cell occupancy by particles, or for placing small fixed point sources in the domain etc. All biological cells will henceforth be referred to by their representation in the framework as "particles" unless explicitly stated, in order to distinguish from grid cells (the volumes delineated by grid points.) OEH, however, would not be advantageous where advection is needed as when it is paired with advective terms the stability condition is no longer unconditional and is in fact more restrictive than FTCS [4].

A zero-diffusion (zero gradient) boundary condition is used here, although other conditions can be selected in the framework. For FTCS, the boundary condition can be set by separately updating guard points outside the boundary of the domain. For OEH, in the second step (2.2) it is simplest to change the formulation and behavior just inside the boundary when the updater advances ϕ.

For the $f(\vec{r}, t)$ term in (1.1) the interpolation/deposition scheme must be introduced. For example, bilinear interpolation is show below. The indices i and j are such that $x_i < x < x_{i+1}$ and $y_j < y < y_{j+1}$. Then:

$$\begin{aligned}\phi^n(x,y) = (1-\frac{x-x_i}{\Delta x})(1-\frac{y-y_j}{\Delta y})\phi^n_{i,j} + (\frac{x-x_i}{\Delta x})(1-\frac{y-y_j}{\Delta y})\phi^n_{i+1,j} + \\ (1-\frac{x-x_i}{\Delta x})(\frac{y-y_j}{\Delta y})\phi^n_{i,j+1} + (\frac{x-x_i}{\Delta x})(\frac{y-y_j}{\Delta y})\phi^n_{i+1,j+1}\end{aligned} \tag{2.4}$$

$$\frac{\partial \phi^n(x,y)}{\partial x} = \frac{(1-\frac{y-y_j}{\Delta y})(\phi^n_{i+1,j}-\phi^n_{i,j}) + (\frac{y-y_j}{\Delta y})(\phi^n_{i+1,j+1}-\phi^n_{i+1,j+1})}{\Delta x}$$

$$\frac{\partial \phi^n(x,y)}{\partial y} = \frac{(1-\frac{x-x_i}{\Delta x})(\phi^n_{i,j+1}-\phi^n_{i,j}) + (\frac{x-x_i}{\Delta x})(\phi^n_{i+1,j+1}-\phi^n_{i+1,j})}{\Delta y} \tag{2.5}$$

allows a particle to evaluate ϕ and $\nabla\phi$ at its location. Using the linear, concentration-dependent ingestion as an example, the total amount a particle deposits, and the amounts it deposits to its neighboring grid points can be determined by the bilinear stencil as:

$$f^n_m = \frac{C\phi^n(x,y)}{\Delta x \Delta y \Delta z} \tag{2.6}$$

$$\begin{aligned}(f_m)^n_{i,j} &= (1-\frac{x-x_i}{\Delta x})(1-\frac{y-y_j}{\Delta y})f^n_m \\ (f_m)^n_{i+1,j} &= (\frac{x-x_i}{\Delta x})(1-\frac{y-y_j}{\Delta y})f^n_m \\ (f_m)^n_{i,j+1} &= (1-\frac{x-x_i}{\Delta x})(\frac{y-y_j}{\Delta y})f^n_m \\ (f_m)^n_{i+1,j+1} &= (\frac{x-x_i}{\Delta x})(\frac{y-y_j}{\Delta y})f^n_m\end{aligned} \tag{2.7}$$

Recall from (2.3) that $(f_m)^n_{i,j}$ is used in updating to step $n+1$. Equation (2.6) is the general form; in this two-dimensional example assume $\Delta z = 1$. See Appendix A for higher-order and three-dimensional interpolation schemes. It may be default to pair an interpolation scheme with an analogous deposition scheme, but results in the following section show the need for another technique. The framework includes

a flat deposition scheme (referred to as "flat2d" when paired with bilinear interpolation, and "flat3d" when paired with trilinear interpolation.) Flat2d is as follows:

$$\begin{aligned}(f_m)^n_{i,j} &= f^n_m/4.\\ (f_m)^n_{i+1,j} &= f^n_m/4.\\ (f_m)^n_{i,j+1} &= f^n_m/4.\\ (f_m)^n_{i+1,j+1} &= f^n_m/4.\end{aligned} \tag{2.8}$$

With the interpolation, the particle equations can be advanced. The test particle equations of motion are:

$$\vec{\imath}^{n+1}_m = \vec{\imath}^n_m + v\frac{\nabla\phi^n(\vec{\imath}^n_m)}{\varepsilon + ||\nabla\phi^n(\vec{\imath}^n_m)||} \tag{2.9}$$

This equation is simply (1.3) with a forward-Euler time step and spatial derivatives acquired from interpolation (such as (2.5) if bilinear is used.)

For the bacterial particle, which runs and tumbles, slightly more complicated steps are taken. Equations (1.5) and (1.6) become:

$$q^n_m = \frac{1}{(1+\phi^n(\vec{\imath}^n_m))^2}\frac{\phi^n(\vec{\imath}^n_m) - \phi^{n-1}(\vec{\imath}^{n-1}_m)}{\Delta t} \tag{2.10}$$

$$(p[k>0]_{\Delta t})^n_m = 1 - e^{-\Delta t\lambda e^{-\alpha q^n_m}} \tag{2.11}$$

The only new requirement is the storage of an interpolated concentration from a previous time step. As implemented here, the Poisson distribution does not describe whether a particle *is* tumbling, but rather

whether it *has* tumbled. If a particle has tumbled, it will have reoriented:

$$
\begin{aligned}
&\text{if } s_1 < (p[k>0]_{\Delta t})_m^n : \\
&\qquad \hat{u}_m^{n+1} = R(2\pi s_2, \hat{v}_m^n)\hat{u}_m^n \\
&\qquad \hat{v}_m^{n+1} = R(\theta, \hat{u}_m^n)\hat{v}_m^n \\
&\text{otherwise :} \\
&\qquad \hat{u}_m^{n+1} = \hat{u}_m^n \\
&\qquad \hat{v}_m^{n+1} = \hat{v}_m^n \\
&\text{always :} \\
&\bar{r}_m^{n+1} = \bar{r}_m^n + v\Delta t\hat{v}_m^{n+1}
\end{aligned}
\tag{2.12}
$$

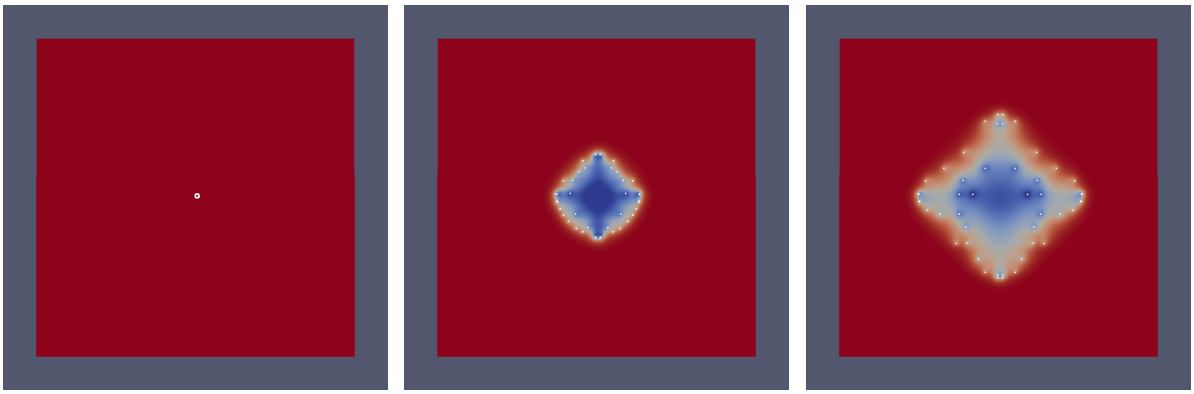
where R is a rotation matrix. For example, $R(\theta, \hat{v}_m^n)$ will rotate around an axis defined by $\hat{v}_m^n$, by an angle of θ. The angle, $\theta = cos^{-1}(2\sqrt{s_3} - 1)$. This transformation yields a density function with a mean of 67 degrees that very closely approximates the observations of Berg, which had a mean of 62 degrees [1]. The bacteria particle, as implemented, actually supports a small drift angle during a run, which is achieved in much the same way a tumble is, but for clarity it is left out of (2.12) as it has been set extremely low in the absence of clear information regarding a reasonable value. It is a deficiency of the model that it does not capture the time spent tumbling, as under certain circumstances a bacterium can spend a significant amount of time tumbling [5]. Some work was conducted using state variables for each particle, where the Poisson process would describe the probability of having left one state to another, but sufficient information regarding the distribution of tumble durations would be required. The uniform random variables s are obtained using the *drand48* pseudo-random number generator from the GNU compiler extensions. While this is adequate, better routines which are also portable are under consideration.

For all simulations in the following chapter, the particles were laid out uniformly in a small ring in the x-y plane. For bacterial particles, this initial condition is not complete. The initial $\hat{u}$ and $\hat{v}$ vectors are established by using a uniform random unit cube and rejecting points beyond the unit sphere. Periodically through the simulation the unit vectors are reorthogonalized and renormalized as was found necessary after many repeated applications of the rotation matrices.

Chapter 3

Results

Of a variety of diagnostic simulations created in testing, one produces a result of interest. The initial condition sets up a uniform concentration, and a ring of particles (which exhibits point symmetry and rotational invariance of $2\pi/N$ where N is number of particles). In this case, $N = 50$. It is reasonable to expect this "ring test" to exhibit this invariance and symmetry until the particles moving outward approach boundaries. This test used the default pairing of bilinear interpolation and bilinear interpolation mentioned in Ch. 2. For all test particle simulations, the square domain is 400 μm on edge, and grid spacing is 4 μm in both directions. See Appendix B for complete input files.



(a) 0 seconds **(b)** 5 seconds **(c)** 10 seconds

Figure 3.1: Test particles with bilinear interpolation/deposition[†]

Clearly this pairing does not produce desirable results. Figure 3.2 shows the paths the particles follow. The particles tend to follow grid lines, turning at right angles. Furthermore, the particles left toward the center in the wake of the others can be seen oscillating about their location, when viewing the results in animation. The problem is that the particles are responding to their own deposition, their own self-induced gradients as illustrated in Figure 3.3.

[†]All particle visualizations are produced using Paraview 3.6.12 by Kitware, Inc.

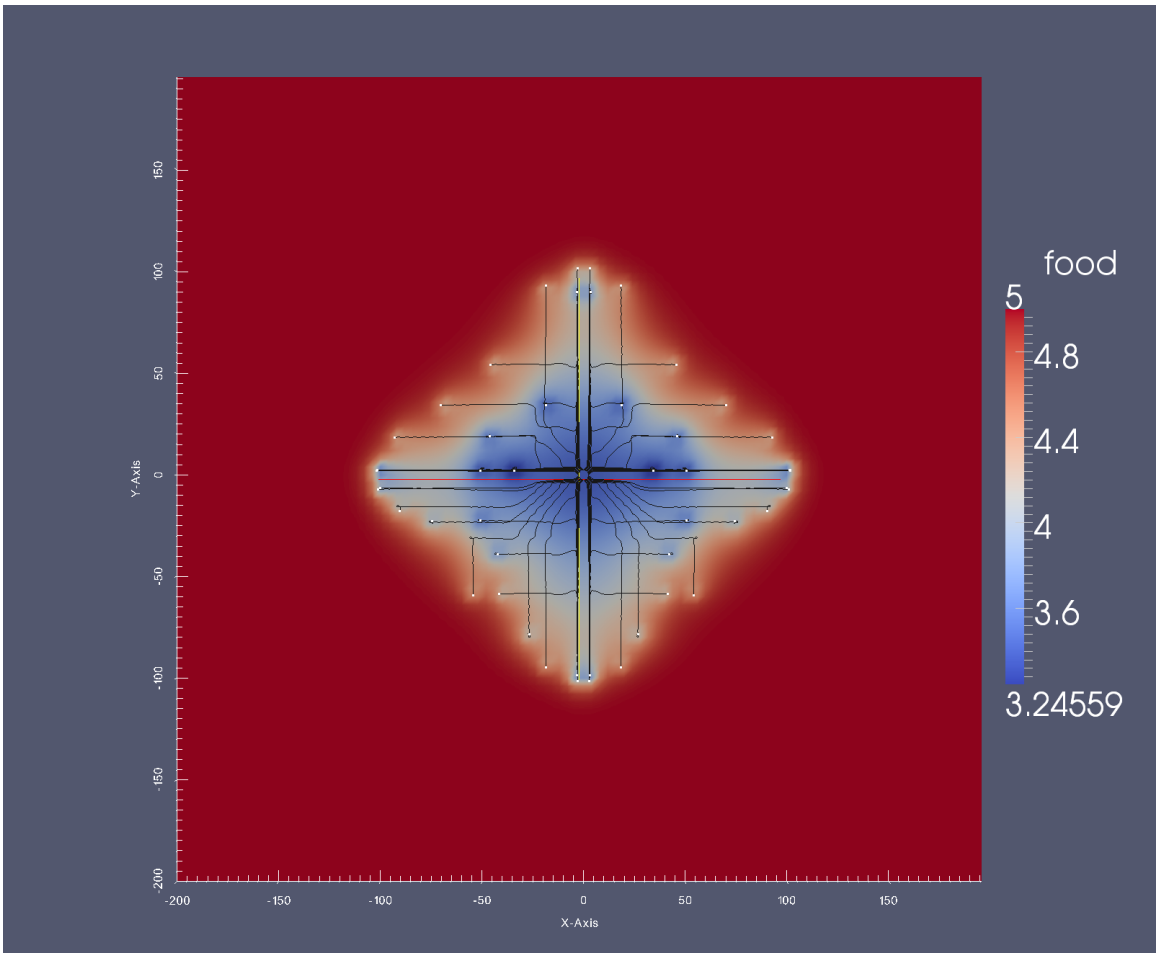
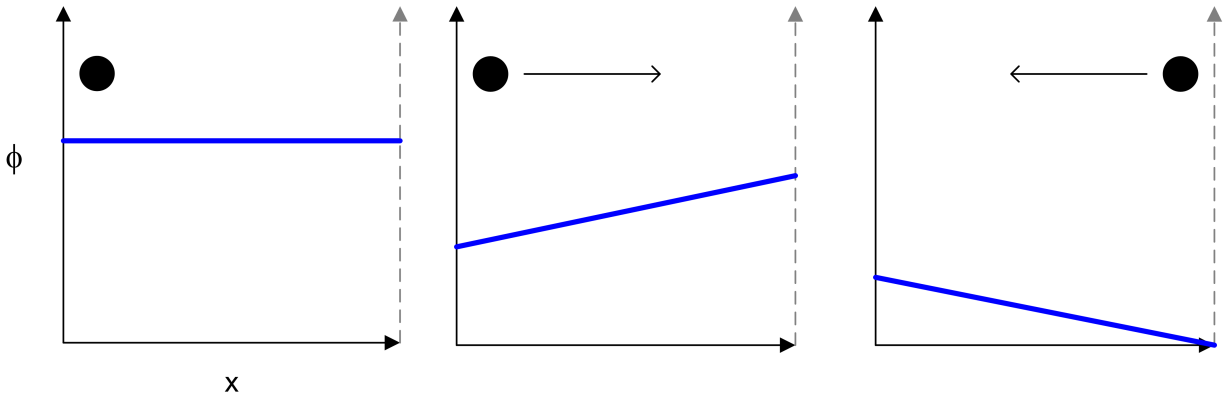


Figure 3.2: Pathlines of test particles with bilinear interpolation/deposition

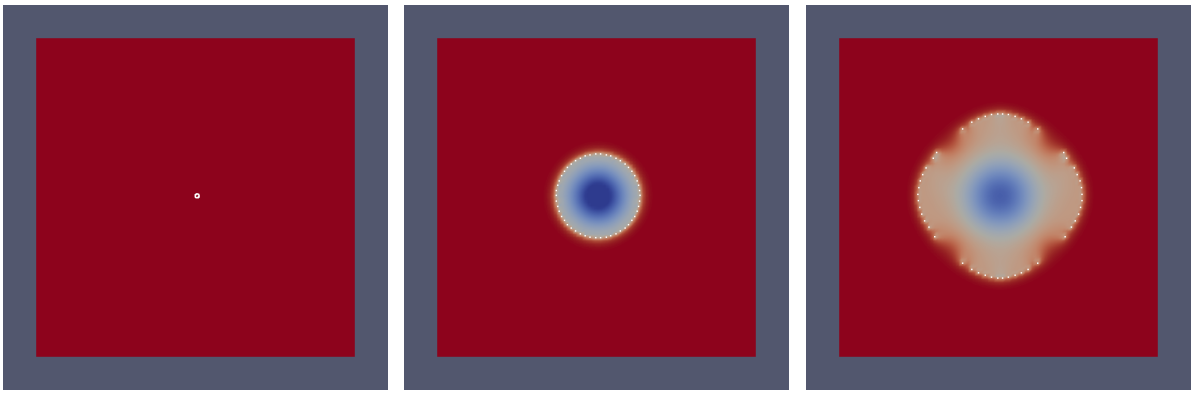


(a) A test particle place in grid cell at step 0

(b) Particle induces local gradient toward right

(c) Particle has moved right according to gradient, induces leftward gradient

Figure 3.3: Self-induced gradient



(a) 0 seconds **(b)** 5 seconds **(c)** 10 seconds

Figure 3.4: Test particles with bicubic interpolation/deposition

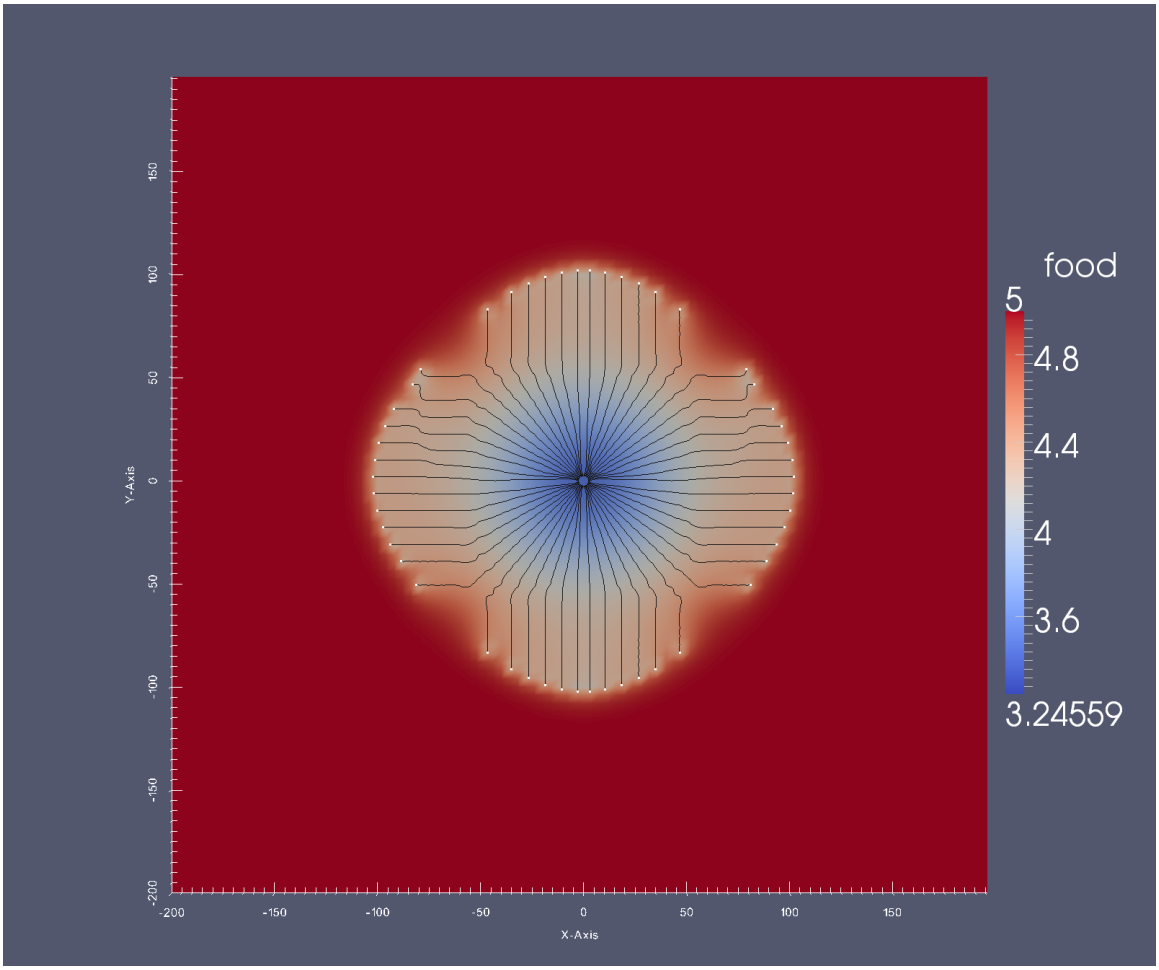


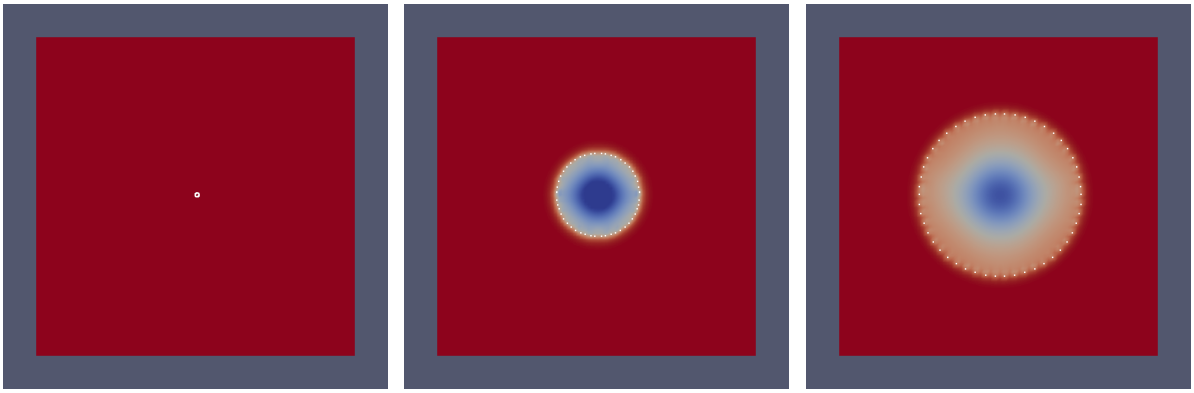
Figure 3.5: Pathlines of particles with bicubic interpolation/deposition

Figures 3.4 and 3.5 show that increasing to bicubic interpolation and deposition has an effect, but does not solve the problem. The effect of the higher-order interpolation is largely due to the increased stencil

size, which extends one grid point farther in every direction. The problem is also delayed if the particle count is increased. When viewed in motion, with grid cells outlined, it becomes clear that the problem arises when grid cell occupancy lowers, and particles begin to have empty grid cells in between each other (or multiple empty cells, in the case of bicubic, such that the stencils do not overlap.) Also observable when in motion, is that some particles even reverse direction completely, to travelling inward against what the expected gradient would be, and against the global character of the concentration field.

It was conjectured that making the Δt of the particles a multiple of that of the fields would allow more iterations and a longer duration of diffusion in between a particle depositing onto a field, and then interpolating the gradient from it. Even with that ratio being one hundred (keeping Δt for fields fixed) the results are nearly indistinguishable, and that result is omitted for brevity.

The problem requires a direct solution, and a direct solution is to disallow a particle from directly inducing a local gradient at its own location. The "flat" deposition scheme does not directly alter the gradient within a grid cell, and thereby solves the problem as seen in Figures 3.6 and 3.7. The rotational invariance is roughly preserved. With this deposition scheme there does not seem to be sensitivity to grid aliasing; in the beginning of the pathlines there is noticeable grid effect, yet the solution improves toward the expected result.



(a) 0 seconds **(b)** 5 seconds **(c)** 10 seconds

Figure 3.6: Test particles with bilinear interpolation, flat deposition

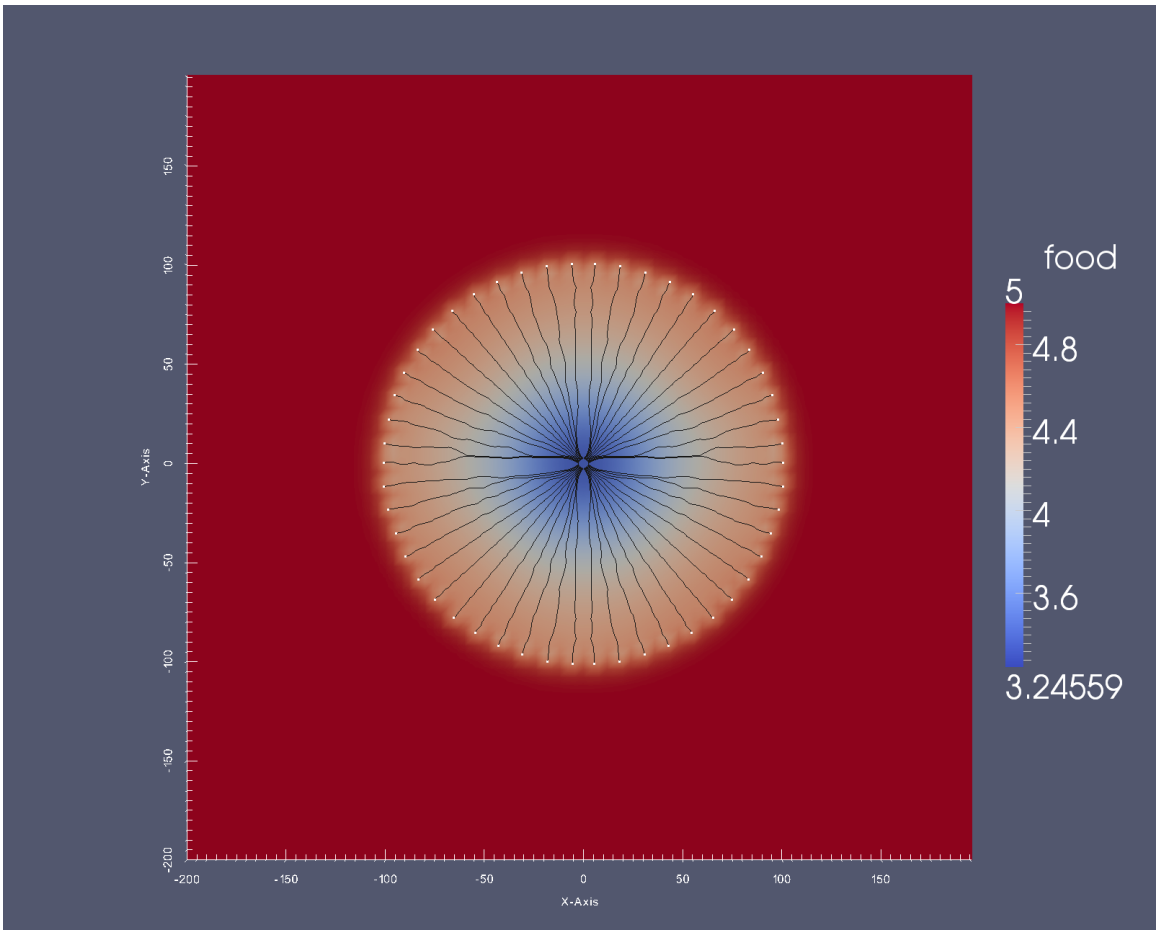


Figure 3.7: Pathlines of particles with bilinear interpolation, flat deposition

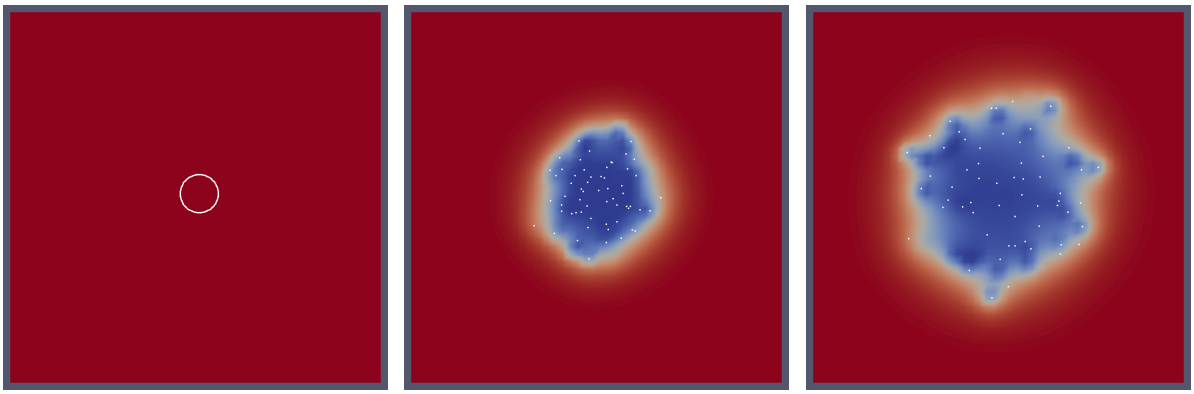
$\Delta x \& \Delta y$, Δt	8, 0.0008	4, 0.0002	2, 0.00005	1, 0.0000125
$\overline{\|\|\vec{r}_m\|\|}$	100.4680	101.3120	101.0220	100.0020
Variance($\|\|\vec{r}_m\|\|$)	0.791837	0.090815	0.088887	1.58657000

Table 3.1: Successive mesh and time step refinements with test particles

Table 3.1 shows the final mean distance-from-origin of all particles through successive refinements in the grid and time step. The solution apparently converges to a value near 101 um. The sharp rise in variance at the last refinement indicates round-off error overshadowing error from truncation. Again, the expected solution (in an infinite domain) is perfectly round, and the lowest variance found in the third column represents a very round pattern considering the underlying rectilinear grid and finite square domain.

The bacteria model is inherently three-dimensional and has not (yet) been adapted to project onto two dimensions (e.g. uniform in z-direction). The domain is 200 μm wide in x and y, and is 100 μm wide in z. Grid spacing is 4 μm in x and y, and is 10 μm in z. A similar initial distribution of 100 particles into a ring shape in the x-y plane is used, with the initial direction randomization discussed in Ch. 2. Because the

domain is a volume, it has been sliced halfway through the z dimension to reveal (some of) the particles. With previous lessons in mind, flat deposition is used, even though bacteria do not directly sense the gradient (and are not modelled to here.)



(a) 0 seconds **(b)** 5 seconds **(c)** 10 seconds

Figure 3.8: Bacteria particles with trilinear interpolation, flat deposition

$\Delta x\&\Delta y$, Δz, Δt	4, 10, 0.0002	2, 5, 0.00005
$\overline{\|\|\vec{r}_m\|\|}$	72.5912	72.4589
	72.7386	72.2696
	71.7445	72.7012
	71.7970	73.3656
	71.3473	73.2029
	72.2258	72.3921
	72.6959	72.3700
	71.6525	72.5469
	72.4768	73.4839
	71.0973	73.3534
Mean	72.0367	72.8145
Variance	0.345813	0.230766

Table 3.2: Bacteria particle statstics and refinement

The bacteria results closely resemble digitally-tracked bacteria trajectories [1]. This simulation should mimic a bacterial swarm in agar. Indeed the formation of the distribution and outermost "ring" is in agreement with descriptions of the advancement of the perimeter and bacteria left lingering in the wake of the fastest-advancing outermost bacteria [6]. While there is not necessarily an expectation of a round pattern in swarm experiments, measurements involving the radius of the pattern in the agar are often made. Table 3.2 shows reasonable agreement between successive runs of the simulation. Similar to the test particles, the table also shows the overall character of the solution is not sensitive to refinement.

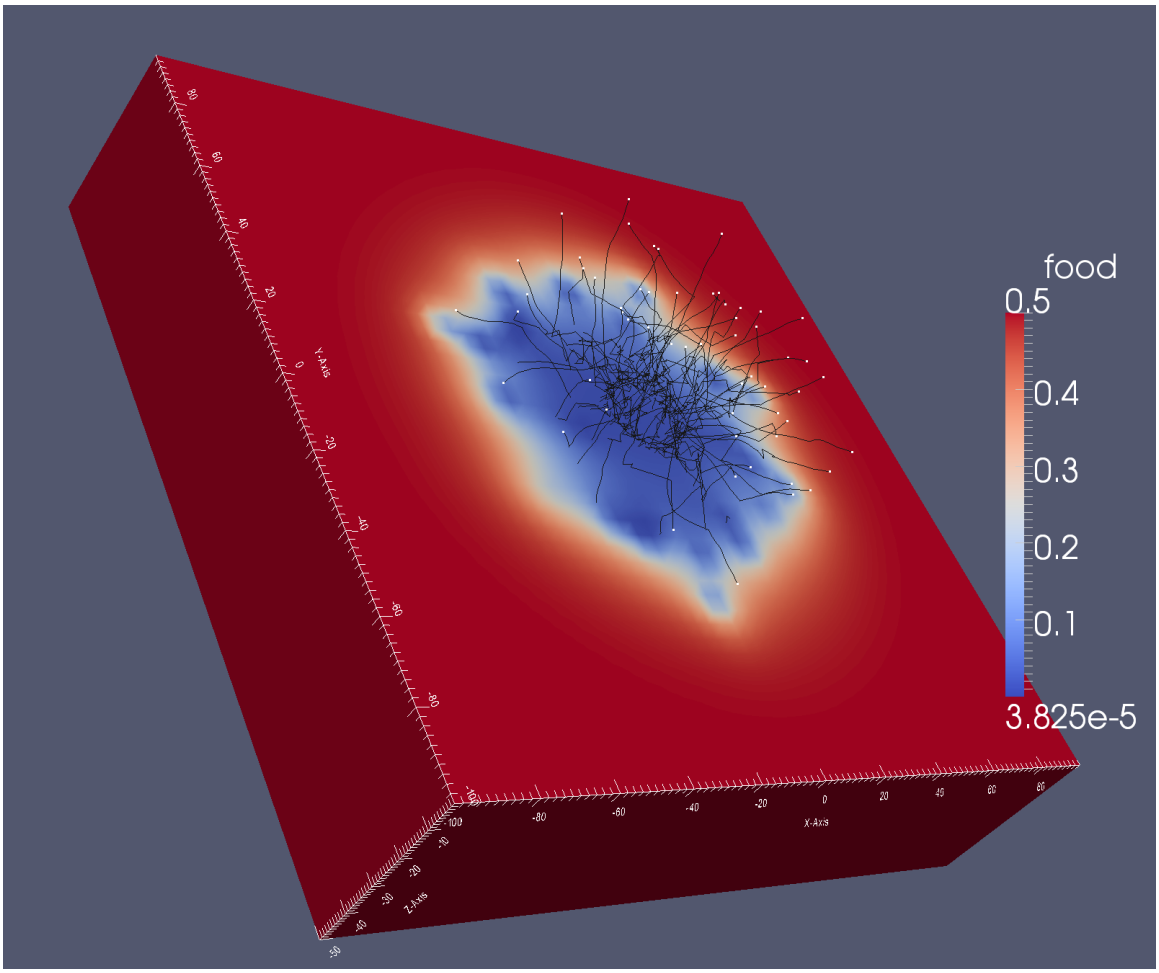


Figure 3.9: Pathlines of bacteria particles

Chapter 4

Summary and Future Directions

While the bacterial results have the expected character and particle behavior, they are not fit for experimental validation, as the input parameters are mostly inferred from different experiments. The model comprised of the particular components is not claimed to be a new method for simulating bacteria. It is meant as a demonstration of a framework in which many varying models can be built and interact with one another through fluid equations, or more generally through fields. The bacteria simulation represents a plausible result while exercising updaters, interpolators, depositors, and a particle set. Future work could see additional components, such as particle sets representing immune cells with amoeboid locomotion or updaters to support time-advancement of flow fields (advection). Static scalar fields could be used to represent obstacles or interesting topology in the domain, perhaps alongside a non-uniform diffusion coefficient field describing whether obstacles are porous. Particle sets could easily be extended to support reproduction or even death.

The software has adequate performance as well. All of the simulations shown—aside from those in the refinement tables—can be completed in roughly a minute or less on a single workstation. A judicious choice of domain decomposition would expose significant parallelism in both the field and particle updates, enhancing performance or allowing significantly larger simulations. This domain decomposition would be built on top of a Message Passing Interface (MPI) to allow large scale parallel simulations.

References

[1] H. Berg and D. Brown. Chemotaxis in escherichia coli analysed by three-dimensional tracking. *Nature*, 239(5374):500–504, 1972. cited By (since 1996) 453.

[2] T. Emonet, C. Macal, M. North, C. Wickersham, and P. Cluzel. Agentcell: A digital single-cell assay for bacterial chemotaxis. *Bioinformatics*, 21(11):2714–2721, 2005. cited By (since 1996) 40.

[3] E. Keller and L. Segel. Model for chemotaxis. *Journal of Theoretical Biology*, 30(2):225–234, 1971. cited By (since 1996) 259.

[4] J. G. Verwer and B. P. Sommeijer. Stability analysis of an odd-even-line hopscotch method for three-dimensional advection-diffusion problems. *SIAM Journal on Numerical Analysis*, 34(1):376, 1997. ISSN 00361429.

[5] A. Wolfe and H. Berg. Migration of bacteria in semisolid agar. *Proceedings of the National Academy of Sciences of the United States of America*, 86(18):6973–6977, 1989. cited By (since 1996) 125.

[6] D. Woodward, R. Tyson, M. Myerscough, J. Murray, E. Budrene, and H. Berg. Spatio-temporal patterns generated by salmonella typhimurium. *Biophysical Journal*, 68(5):2181–2189, 1995. cited By (since 1996) 86.

Appendix A: Other interpolation

Higher order interpolation schemes can be compactly stated as a Lagrange polynomial, such as the tricubic below:

$$\phi^n(x,y,z) = \sum_{i'=i-1}^{i+2} \sum_{j'=j-1}^{j+2} \sum_{k'=k-1}^{k+2} \left[\left(\prod_{\substack{i''=i-1 \\ i'' \neq i'}}^{i+2} \frac{x-x_{i''}}{x_{i'}-x_{i''}} \right) \left(\prod_{\substack{j''=j-1 \\ j'' \neq j'}}^{j+2} \frac{y-y_{j''}}{y_{j'}-y_{j''}} \right) \left(\prod_{\substack{k''=k-1 \\ k'' \neq k'}}^{k+2} \frac{z-z_{k''}}{z_{k'}-z_{k''}} \right) \phi^n_{i',j',k'} \right]$$

The first derivatives—which are all that are required so far for these models—can be computed easily using the product rule, such as below:

$$\frac{\partial \phi^n(x,y,z)}{\partial x} = \sum_{i'=i-1}^{i+2} \sum_{j'=j-1}^{j+2} \sum_{k'=k-1}^{k+2} \left[\left(\sum_{\substack{i^*=-1 \\ i^* \neq i}}^{i+2} \left(\prod_{\substack{i''=i-1 \\ i'' \neq i' \\ i'' \neq i^*}}^{i+2} \frac{x-x_{i''}}{x_{i'}-x_{i''}} \right) \right) \left(\prod_{\substack{j''=j-1 \\ j'' \neq j'}}^{j+2} \frac{y-y_{j''}}{y_{j'}-y_{j''}} \right) \left(\prod_{\substack{k''=k-1 \\ k'' \neq k'}}^{k+2} \frac{z-z_{k''}}{z_{k'}-z_{k''}} \right) \phi^n_{i',j',k'} \right]$$

(Keeping in mind that an empty production is equal to one, and not zero.) For less dimensions, such as bicubic (2d) simply remove the summation and production involving z. To obtain a trilinear, run the summations/productions from i to $i+1$ instead of from $i-1$ to $i+2$ (and the same for j and k.) For deposition, the summations are replaced by for-loops, and the productions represent the weight-coefficient for each grid point determined at a specific i', j' and k'.

Appendix B: Input files

Test particles, bilinear:

```
<?xml version="1.0" ?>
<domain>
  <time duration="10" dt="0.0002" dumpPeriod="0.04" tratio="1"/>

  <particleSet type="fakeBacteria" name="cells" n="50">
    <constants gradSens="10.0" ingestRate="12.8" randomMagn="0.0" />
    <initialCondition type="particleRing" x="0." y="0." z="0." r="2.4" />
    <interpolator type="bilinear" label="food" fieldIndex="0" />
    <depositor type="bilinear" label="food" fieldIndex="0" />

  </particleSet>

  <gridField name="food" nTimes="2" tZero="0">
    <origin x="-200" y="-200" z="0." />
    <length lx="400." ly="400." lz="1." />
    <count nx="100" ny="100" nz="1" ncomp="1" guard="3" />

    <initialCondition type="gridUniform" value="5.0"/>
  </gridField>

  <updater type="bcZeroGrad" fieldIndex="0">
    <readField index="0" timeLevel="1" />
    <writeField index="0" timeLevel="1" />
  </updater>
```

```
  <updater type="gridLaplace" fieldIndex="0" c="3e1">
    <readField index="0" timeLevel="0" />
    <writeField index="0" timeLevel="1" />
  </updater>


</domain>
```

Test particles, bicubic:

```
<?xml version="1.0" ?>
<domain>
  <time duration="10" dt="0.0002" dumpPeriod="0.04" tratio="1"/>

  <particleSet type="fakeBacteria" name="cells" n="50">
    <constants gradSens="10.0" ingestRate="12.8" randomMagn="0.0" />
    <initialCondition type="particleRing" x="0." y="0." z="0." r="2.4" />
    <interpolator type="bicubic" label="food" fieldIndex="0" />
    <depositor type="bicubic" label="food" fieldIndex="0" />

  </particleSet>

  <gridField name="food" nTimes="2" tZero="0">
    <origin x="-200" y="-200" z="0." />
    <length lx="400." ly="400." lz="1." />
    <count nx="100" ny="100" nz="1" ncomp="1" guard="3" />

    <initialCondition type="gridUniform" value="5.0"/>
  </gridField>

  <updater type="bcZeroGrad" fieldIndex="0">
    <readField index="0" timeLevel="1" />
```

```
    <writeField index="0" timeLevel="1" />
  </updater>

  <updater type="gridLaplace" fieldIndex="0" c="3e1">
    <readField index="0" timeLevel="0" />
    <writeField index="0" timeLevel="1" />
  </updater>



</domain>
```

Test particles, flat deposition, bilinear interpolation:

```
<?xml version="1.0" ?>
<domain>
  <time duration="10" dt="0.0002" dumpPeriod="0.04" tratio="1"/>

  <particleSet type="fakeBacteria" name="cells" n="50">
    <constants gradSens="10.0" ingestRate="12.8" randomMagn="0.0" />
    <initialCondition type="particleRing" x="0." y="0." z="0." r="2.4" />
    <interpolator type="bilinear" label="food" fieldIndex="0" />
    <depositor type="flat2d" label="food" fieldIndex="0" />

  </particleSet>

  <gridField name="food" nTimes="2" tZero="0">
    <origin x="-200" y="-200" z="0." />
    <length lx="400." ly="400." lz="1." />
    <count nx="100" ny="100" nz="1" ncomp="1" guard="3" />

    <initialCondition type="gridUniform" value="5.0"/>
  </gridField>
```

```
  <updater type="bcZeroGrad" fieldIndex="0">
    <readField index="0" timeLevel="1" />
    <writeField index="0" timeLevel="1" />
  </updater>

  <updater type="gridLaplace" fieldIndex="0" c="3e1">
    <readField index="0" timeLevel="0" />
    <writeField index="0" timeLevel="1" />
  </updater>



</domain>
```

Bacteria particles:

```
<?xml version="1.0" ?>
<domain>
  <time duration="10" dt="0.0002" dumpPeriod="0.04" tratio="10"/>

  <particleSet type="ecoliBacteria" name="cells" n="100">
    <constants
        v="10.0"
        alpha="660"
        lambda0="0.813"
        gamma_up="0.0"
        gamma_down="0.0"
        Lb="0.0"
        ingestRate="4800.0"
    />

    <initialCondition type="particleRing" x="0." y="0." z="0." r="10." />
    <interpolator type="trilinear" label="food" fieldIndex="0" />
    <depositor type="flat3d" label="food" fieldIndex="0" />
```

```
  </particleSet>

  <gridField name="food" nTimes="2" tZero="0">
    <origin x="-100.0" y="-100.0" z="-50." />
    <length lx="200." ly="200." lz="100." />
    <count nx="50" ny="50" nz="10" ncomp="1" guard="1" />

    <initialCondition type="gridUniform" value="0.5" timeOffset="0" />
    <initialCondition type="gridUniform" value="0.5" timeOffset="1" />
  </gridField>

  <updater type="bcZeroGrad" fieldIndex="0" />

  <updater type="gridHopscotch" fieldIndex="0" c="3e1">
    <updater type="bcZeroGrad"  timeOffset="1" />
  </updater>

</domain>
```