

Article

Efficient Methods to Calculate Partial Sphere Surface Areas for a Higher Resolution Finite Volume Method for Diffusion-Reaction Systems in Biological Modeling

Abigail Bowers, Jared Bunn and Myles Kim * 

Department of Applied Mathematics, Florida Polytechnic University, Lakeland, FL 33805, USA; abowers@floridapoly.edu (A.B.); jbunn@floridapoly.edu (J.B.)

* Correspondence: mkim@floridapoly.edu

Received: 14 November 2019; Accepted: 20 December 2019; Published: 23 December 2019



Abstract: Computational models for multicellular biological systems, in both in vitro and in vivo environments, require solving systems of differential equations to incorporate molecular transport and their reactions such as release, uptake, or decay. Examples can be found from drugs, growth nutrients, and signaling factors. The systems of differential equations frequently fall into the category of the diffusion-reaction system due to the nature of the spatial and temporal change. Due to the complexity of equations and complexity of the modeled systems, an analytical solution for the systems of the differential equations is not possible. Therefore, numerical calculation schemes are required and have been used for multicellular biological systems such as bacterial population dynamics or cancer cell dynamics. Finite volume methods in conjunction with agent-based models have been popular choices to simulate such reaction-diffusion systems. In such implementations, the reaction occurs within each finite volume and finite volumes interact with one another following the law of diffusion. The characteristic of the reaction can be determined by the agents in the finite volume. In the case of cancer cell growth dynamics, it is observed that cell behavior can be different by a matter of a few cell size distances because of the chemical gradient. Therefore, in the modeling of such systems, the spatial resolution must be comparable to the cell size. Such spatial resolution poses an extra challenge in the development and execution of the computational model due to the agents sitting over multiple finite volumes. In this article, a few computational methods for cell surface-based reaction for the finite volume method will be introduced and tested for their performance in terms of accuracy and computation speed.

Keywords: partial sphere surface areas; finite volume method; diffusion reaction equation; mathematical model for biology

1. Introduction

Agent-based models (also called single-cell based models or individual-based models) have been widely popular in various fields as tools to build computational models. They have been adopted in biological system modeling [1–3], fluid dynamics simulation [4,5], ecosystem modeling [6,7], human organizational simulation [8,9], business simulation [10], etc. In agent-based modeling, a system is modeled as a collection of autonomous self-decision-making entities called agents. Each agent individually assesses its situation and makes decisions on the basis of a set of rules. These rules can be designed to make internal decisions altering the fate of an agent or can be written to define interactions among spatially neighboring agents. The rule set can also be identical to each agent or can be designed differently for different groups of agents. Agent-based models were proven to be an effective modeling

approach for complex heterogeneous systems. Though all rules are usually written for small scale, agent level or local interaction level, agent-based models are capable of showing population level emerging characteristics that are often unpredictable from the local rule sets. Due to these reasons, cancer biology is one of those areas where agent-based models have been successful [3,5,11]. In a biological system, cells interact with one another in various ways, through physical interaction and/or through chemical signaling. Cells uptake molecules from the surrounding medium, and molecules may come from nearby cells because they synthesize and secrete molecules. Due to the diffusive nature of the molecules in media, the mathematical model for the biological cells that consume and secrete molecules in media would be a diffusion-reaction equation and finite volume methods are a popular choice for the numerical implementation [11,12].

It has been experimentally observed that cancer cells' fates can be different by a matter of one or two cell lengths due to the spatial nutrient concentration change [13]. Thus, a computational model for cancer cells with physiology including consumption and secretion of molecules will require a spatial resolution of single cell length. Uptake of molecules by cells may involve receptors on the cell surface and endocytosis process. If a 3D finite volume implementation is designed with a cell surface reaction, one cell length spatial resolution, and cells represented by spheres (agents), then a snapshot around one cell would be like Figure 1. The cell in Figure 1 overlaps with eight finite volumes. The contribution of this cell to the whole diffusion-reaction system needs to be divided into eight parts proportional to the cell surface area in each finite volume that the cell overlaps; S_1, S_2, \dots, S_8 in Figure 1. In a diffusion-reaction equation like Equation (1), the reaction coefficient, λ , would depend, as a function of spatial variables, on the values of S_i 's from all cells in the system. Thus, it is important to calculate the precise values of S_i 's to capture the correct rate of reaction.

$$\frac{\partial c}{\partial t} = D\nabla^2 c + \lambda c \tag{1}$$

$$S = \iint_R \frac{r}{\sqrt{r^2 - (x-a)^2 - (y-b)^2}} dA \tag{2}$$

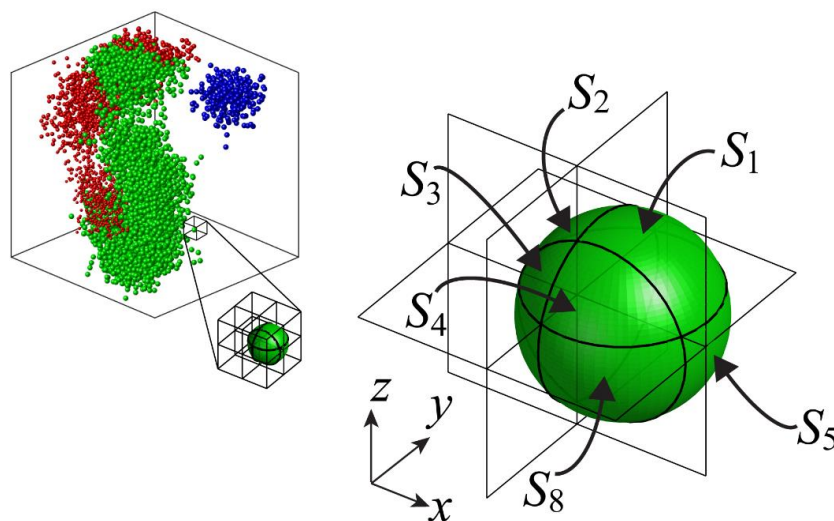


Figure 1. Schematic diagram of an agent-based model for 3D cell proliferation (left) and a spherical agent representing a cell overlapping with surrounding finite volumes (right). There are more than 4600 agents in the snapshot of a simulation (left). A small zoomed-in view of a cell with surrounding finite volumes (rectangular boxes) is shown in the middle. A further zoomed-in view of the cell is displayed on the right. When the agent's diameter is smaller than the edge of the rectangular box, the agent can overlap with up to eight finite volumes. $S_1, S_2, \dots,$ and S_8 are surface areas that overlap eight finite volumes.

Analytic formulas are not readily available for the calculation of these areas even when all geometric specifications are given. An attempt to calculate these surface areas, S_i 's, can be made by using the function $f(x, y) = \sqrt{r^2 - (x - a)^2 - (y - b)^2}$, which results in a double integral as in Equation (2). This integral, however, is possible analytically only when the domain R is extremely simple in polar coordinates. Therefore, numerical calculation methods need to be deployed. Diffusions in biological contexts are usually fast. The stability of numerical schemes for a fast diffusion requires small time stepping. Furthermore, typical cancer cell simulations consider thousands of cells and use durations of days and weeks of their lives [3,5]. Considering all these, we need to find an efficient numerical method to calculate the cell surface area in finite volumes for thousands of cells repeatedly for the duration of simulation not to burden the overall numerical simulation. In this article, we suggest three different computational methods and test them for their performance in terms of the accuracy and the computational speed. The first one is a surface area calculation based on the triangulation. The second and the third methods are two different Monte Carlo methods. All these three methods are improved by using a linear system of equations for $S_i, i = 1, 2, \dots, 8$.

2. Methods

Here, we suggest three different computational methods and test their performance in terms of accuracy and computation speed. The goal of each method is to calculate the values of S_i for $1 \leq i \leq 8$. In general, it is not possible to calculate those values analytically unless they fall into special cases such as when they are all identical. However, the sum of $S_1, S_2, S_3,$ and S_4 can be calculated analytically because it is a surface of revolution that can be obtained by rotating an arc on a circle around an axis. The same can be computed for $S_5, S_6, S_7,$ and S_8 . The sum of $S_1, S_2, S_3,$ and $S_4,$ and the sum of $S_5, S_6, S_7,$ and S_8 are also areas of two surfaces that we can get by cutting the sphere with a plane. It only requires knowing the radius of the sphere and the distance from the center of the sphere to the plane that cuts through it. If the distance between the center and the plane is d and the sphere radius is R , then

$$S_1 + S_2 + S_3 + S_4 = 2\pi \int_d^R \sqrt{R^2 - t^2} \sqrt{1 + \frac{t^2}{R^2 - t^2}} dt = 2\pi R(R - d) \tag{3}$$

and

$$S_5 + S_6 + S_7 + S_8 = 2\pi \int_{-R}^d \sqrt{R^2 - t^2} \sqrt{1 + \frac{t^2}{R^2 - t^2}} dt = 2\pi R(d + R). \tag{4}$$

Here, we assume that $S_1 + S_2 + S_3 + S_4$ is the area of the smaller cut. There are two sets of similar calculations that can be done. They are the set of $S_1 + S_4 + S_5 + S_8$ and $S_2 + S_3 + S_6 + S_7,$ and the set of $S_1 + S_2 + S_5 + S_6$ and $S_3 + S_4 + S_7 + S_8$. These can be analytically calculated when the radius and the center of sphere are known. It appears that there is no other analytical closed form equation other than these. By collecting these, we can form a linear system of equations,

$$\begin{cases} S_1 + S_2 + S_3 + S_4 = \sigma_1 \\ S_5 + S_6 + S_7 + S_8 = \sigma_2 \\ S_1 + S_4 + S_5 + S_8 = \sigma_3 \\ S_2 + S_3 + S_6 + S_7 = \sigma_4 \\ S_1 + S_2 + S_5 + S_6 = \sigma_5 \\ S_3 + S_4 + S_7 + S_8 = \sigma_6 \end{cases} \tag{5}$$

Here, the values of σ_i are knowable. This linear system, however, is redundant. A version without redundancy will be

$$\begin{cases} S_1 + S_2 + S_3 + S_4 = \sigma_1 \\ S_5 + S_6 + S_7 + S_8 = \sigma_2 \\ S_1 + S_4 + S_5 + S_8 = \sigma_3 \\ S_1 + S_2 + S_5 + S_6 = \sigma_5 \end{cases} \quad (6)$$

Since we have eight variables in four equations, we need to have more equations to find the solution. To add more equations, we will use numerical methods. In fact, we will calculate some of the S_i values as additional equations. We will deploy numerical methods for this. One method is just a numerical calculation of the surface area using triangulation. The surface area calculations using triangulation are easier and simpler for areas that belong to the smaller piece when the sphere is cut by a plane. For example, $S_1, S_2, S_3,$ and S_4 in Figure 1 make the smaller cut, and they can be seen as function graphs from a same function (of x and y) with different domains on the horizontal plane (xy plane). The surface for S_5 in Figure 1, however, may not be seen as a function graph with a domain on a plane. This makes the calculation for S_5 difficult. We can solve S_5 by adding more equations to Equation (6). The other two methods are Monte Carlo methods. We will generate points on the sphere. One method will generate sample points from a uniform distribution on a sphere and the other method will design a set of points that are almost uniform on a sphere. Monte Carlo methods will count points that fall onto the region for S_i and calculate S_i by measuring the proportion of those points. In fact, Monte Carlo methods do not necessarily need to use Equation (6). However, we choose to use Equation (6) as well for the purpose of comparison and to improve the computation speed. Details of three methods are described in the following three subsections.

2.1. Surface Area Calculation Using Triangulation

We will use the settings in Figure 1 to explain this method. It is assumed that $S_1, S_2, S_3,$ and S_4 in the configuration presented make the smaller piece when the sphere is cut by a horizontal plane. It is also assumed that the sphere center is in the fifth octant where the surface of S_5 belongs. As was mentioned earlier, the surfaces of $S_1, S_2, S_3,$ and S_4 are function graphs with the same function formula but with different domains on the horizontal plane. Thus, the surface area calculations can be done by triangulating domains and approximating the surface with triangles. Approximated values for S_i for $1 \leq i \leq 4$ will replace the first equation in Equation (6). S_8 will be calculated by the same method and inserted to Equation (6). Then, the new system equation is

$$\begin{cases} S_1 = s_1 \\ S_2 = s_2 \\ S_3 = s_3 \\ S_4 = s_4 \\ S_8 = s_8 \\ S_5 + S_6 + S_7 + S_8 = \sigma_2 \\ S_1 + S_4 + S_5 + S_8 = \sigma_3 \\ S_1 + S_2 + S_5 + S_6 = \sigma_5 \end{cases} \quad (7)$$

This equation is consistent and has a unique solution. Bringing the first five equations to the seventh equation, S_5 can be solved. Applying the first five equations and the S_5 value to the last equation, we get S_6 . The sixth equation can be solved for the last unknown S_7 . We can avoid the calculation of S_5 . While the method is explained based on the setting depicted in Figure 1, in general, any situation can be like Figure 1 with a maximum of two reflections in the respective planes. If $S_1, S_2, S_3,$ and S_4 make the bigger piece, the sphere can be reflected with respect to the horizontal cutting surface. Then, the surface above the horizontal cutting plane will be the smaller piece. If S_8 belongs to the bigger piece than S_5 , the reflection with respect to the vertical plane between S_5 and S_8 will switch the roles of S_5 and S_8 . Reflections do not complicate the algorithm too much and it is not computationally

costly. Furthermore, the S_8 calculation can be computed with the domain on the horizontal domain after a simple linear transformation. If the linear transformation, $(x, y, z) \mapsto (z, -x, -y)$, is applied to the sphere, the region in the eighth octant comes up to the third octant (Figure 2). Thus, S_1, S_2, S_3, S_4 , and S_8 , all can be calculated as graphs of functions of x and y , which we found convenient.

We discovered that the triangulation of the domains is one of the computationally expensive parts in the algorithm. To make the algorithm more efficient, we adopted a measure to reduce the burden of domain triangulation. Once we identify a desired resolution for the triangulation in terms of the number of vertices, a triangulation of a disk is generated using the mesh generation package distmesh [14]. The union of the domains for S_1, S_2, S_3 , or S_4 as function graphs is a disk. The pre-generated triangulation vertices can be scaled and translated to be used for the domain of S_1, S_2, S_3 , or S_4 together. For each domain, we identify vertices that are on the domain (with an appropriate margin along the straight-line borders) and add vertices with similar spacings along the straight-line borders. A triangulation is generated based on this collection of vertices using the Delaunay triangulation (Figure 3).

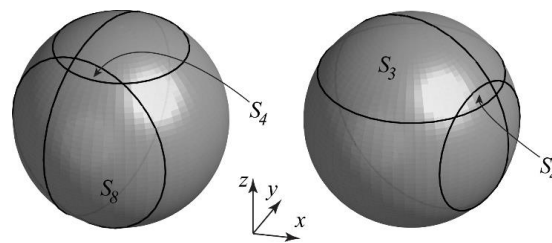


Figure 2. A linear transformation, $(x, y, z) \mapsto (z, -x, -y)$, converts S_8 to S_3 without changing the size and shape. The sphere on the left becomes the sphere on the right after the linear transformation. All circles are intersections with xy, yz, xz planes. Note that the surface for S_8 on the left is in the eighth octant. The same size and shape surface is in the third octant of the sphere on the right after the linear transformation.

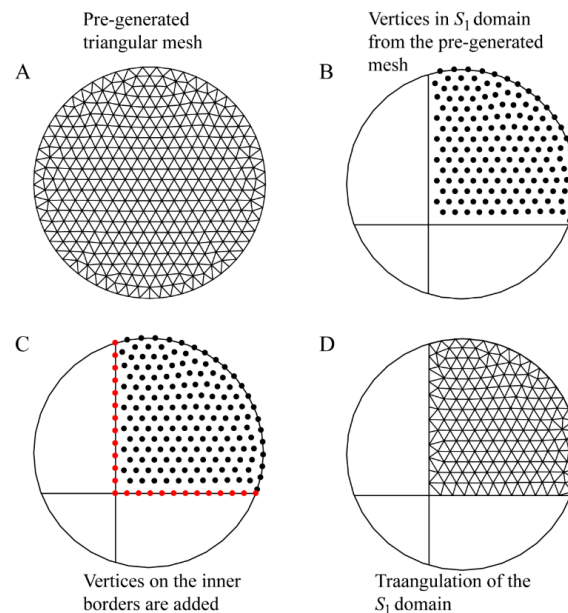


Figure 3. The triangulation for the domain of S_1 . Panel (A) shows the pre-generated triangulation of a disk with a desired resolution. In panel (B), vertices in the domain for S_1 are identified and collected. In this process, those vertices that are too close to the inner border lines are excluded. Panel (C) shows newly added vertices on the inner border lines (red dots) with a similar spatial resolution. The triangulation based on the black and red vertices is done in panel (D). The value of S_1 is calculated using this triangulation.

2.2. Monte Carlo Using Sample Points from a Uniform Distribution on Spheres

There are several different algorithms that generate uniformly distributed sample points on spheres [15–19]. We used normally distributed random vectors to generate uniformly distributed sample points on spheres [20,21]. When three-dimensional vectors, $\mathbf{X} = (x_1, x_2, x_3)$, are generated with all x_i values from $N(0, 1)$, the standard normal distribution, the collection of $\mathbf{X}/|\mathbf{X}|$ are uniformly distributed on the unit 2-sphere as vertices (points). Here, $|\mathbf{X}| = \sqrt{x_1^2 + x_2^2 + x_3^2}$. A set of these random points will be generated. For any sphere representing a cell, these random points will be scaled and translated to fit on the surface of a cell-representing sphere. Five areas, S_1, S_2, S_3, S_4, S_8 in Figure 1, for example, will be approximately calculated using a Monte Carlo method, that is, by counting points on the corresponding regions. The remaining three values will be calculated by Equation (7).

2.3. Monte Carlo Using the Icosahedron-Based Vertices on Spheres

The method in the previous section was to generate dense enough random samples that converge to the uniform distribution over the sphere and classify those points using the regions in eight octants. Since the sample points are randomly generated, the uniformity may vary, and the computational performance may fluctuate. In this section, we will introduce another method that uses almost uniform vertices distributed on the sphere. We generate almost uniform vertices on the sphere using icosahedral triangulation [22]. A regular icosahedron has twelve vertices, thirty edges, and twenty faces. Each face is an equilateral triangle. A regular icosahedron can be embedded in a sphere in a way that all vertices are on the surface of sphere. These twelve vertices are vertices of the zeroth-generation almost uniform vertices on the sphere. The first-generation almost uniform vertices on the sphere are built based on the zeroth-generation by finding the midpoint of each edge and projecting (from the center of the sphere) those midpoints on the sphere surface. These projected points are the new vertex addition. Since there are thirty edges on the icosahedron, the first-generation almost uniform vertices consist of forty-two vertices. To make the second-generation, it is required to triangulate those forty-two vertices, and the projections of each edge midpoint onto the sphere surface are added as new vertices. Any generation can be found iteratively based on the previous generation. Figure 4 shows the triangulation based on the almost uniform icosahedron-based vertices.

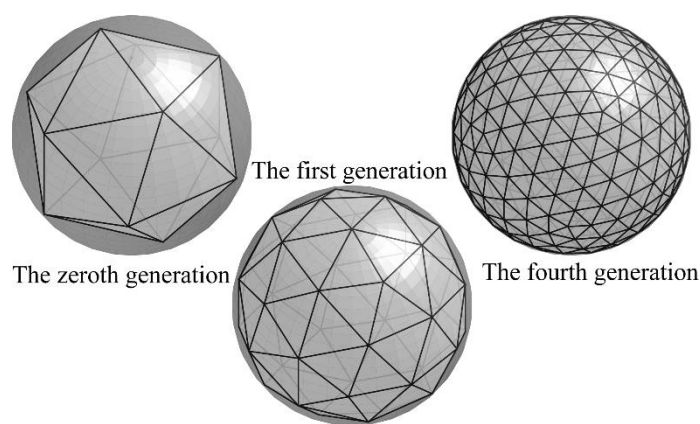


Figure 4. Icosahedron-based vertices on a sphere. The zeroth, first, and fourth-generation vertices are shown with triangulation.

As the generation number increases, the number of total vertices increases. If V_i, E_i , and F_i denote the number of vertices, edges, and faces, respectively, of the i th-generation, we have $V_0 = 12, E_0 = 30, F_0 = 20$. It is also obvious that $V_{i+1} = V_i + E_i$ and $F_{i+1} = 4F_i$. Since the Euler characteristic of a genus zero surface is 2, $E_{i+1} = V_i + E_i + 4F_i - 2$. Thus, we can only choose the number of vertices from the sequence of $\{V_i\}_{i=1}^{\infty} = \{V_0 = 12, V_1 = 42, V_2 = 162, V_3 = 642, V_4 = 2562, V_5 = 10,242, \dots\}$. We will pick a vertex set from a generation and use them as random sample points for the Monte Carlo

method. Figure 5 compares vertices on a matching region from the random 40,962 sample points from the uniform distribution in Section 0 and vertices from the sixth-generation icosahedral vertices. The vertices collected from random sample points show stochasticity, whereas the points from the icosahedron-based vertices are spaced regularly.

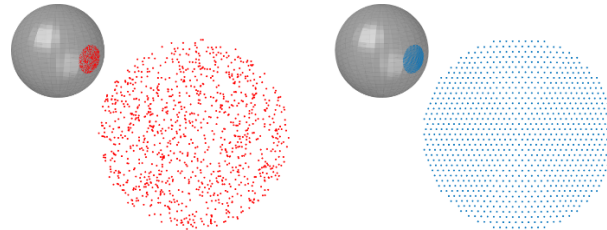


Figure 5. Comparison of points on a matching region that are collected from 40,962 random sample points from the uniform distribution (red, **left**) described in 0 and from 40,962 vertices in the sixth-generation icosahedron-based vertices (blue, **right**). The dots in the region on the sphere and the front views are shown. There are 1086 red dots and 1015 blue dots.

2.4. Implementations for Comparison

We will compare the three methods in different spatial resolutions in terms of computational speed and accuracy. A set of randomly generated one hundred cells (spheres) is used to test the performance of three methods. The number of vertices in the icosahedron-based vertex set is fixed at each generation. The icosahedron-based vertices from the first-generation to the eighth-generation are used. To make fair comparisons, sets of sample points from the uniform distribution on the surface of the sphere are generated (middle column in Table 1) following the number of vertices in icosahedron-based methods. The vertices in the sets of sample points from the uniform distribution are incremental because the icosahedron-based vertices are incremental. For example, 162 vertices in case number 2 contain 42 vertices in case number 1. While the icosahedron-based vertices and sample point vertices from the uniform distribution are on the surface of a sphere, the vertices used for the triangulation method are on a disk and used to triangulate areas on a sphere, but not the whole, half-sphere at most. Thus, we consider a half number of vertices in the triangulation method to be equivalent to, in terms of the number of vertices, the Monte Carlo methods with double vertices. The cases used for the comparison with the number of vertices are listed in Table 1. There are only six cases for the triangulation method whereas there are eight cases for the other two methods. We did not include the two cases that correspond to cases 7 and 8 of the other two methods because the triangulation method takes too long to compute at those resolutions. Instead of including those cases, we will use the result from the triangulation with 333,065 vertices, which correspond to case number 8, as the accuracy reference (true values). It is the eight hundred area calculations ($8 \text{ pieces} \times 100 \text{ cells}$). The average percentage errors will be calculated using the result from the triangulation with 333,065 vertices as the true values. In fact, the triangulation method is a numerical calculation of the integral in Equation (2). Thus, it is certain that the values from the triangular method converge to the true value.

Cells represented by spheres can be anywhere in the computational domain. Monte Carlo methods may perform differently not only depending on the sample point used but also depending on the center location and the radius of the sphere due to the fact that those eight pieces cut by the finite volumes around the sphere will be different and random. The set of one hundred randomly generated cells (spheres) that we use for the test has a few conditions. Since we set the dimension of each finite volume by $7 \mu\text{m} \times 7 \mu\text{m} \times 7 \mu\text{m}$, each cell radius is limited between 5 and $7 \mu\text{m}$. So, a cell can overlap with a maximum of eight finite volumes, as shown in in Figure 1. Therefore, generating random spheres having centers around the origin and overlapping only with a maximum of eight finite volumes around the origin will be enough to represent random cells in the whole computational domain. Tests done by using these randomly generated spheres will also eliminate the necessity to consider other sample points from the uniform distribution on a sphere. Three methods will be used to calculate five areas

($S_1, S_2, S_3, S_4,$ and S_8 for a cell in Figure 1) of each cell and the remaining three areas will be solved through system equations such as Equation (7). The time to perform all calculations for one hundred spheres is measured for the computation speed. The hundred-cell calculation is repeated thirty times and the average over thirty repetitions is presented. The basic flowchart for the computation is in Figure 6 and MATLAB® is used for programming.

Table 1. The number of vertices.

Case Number	Triangulation	Uniform Distribution	Icosahedron-Based
1	21	42	42
2	82	162	162
3	320	642	642
4	1243	2562	2562
5	5120	10,242	10,242
6	20,504	40,962	40,962
7	N/A	163,842	163,842
8	N/A	655,362	655,362

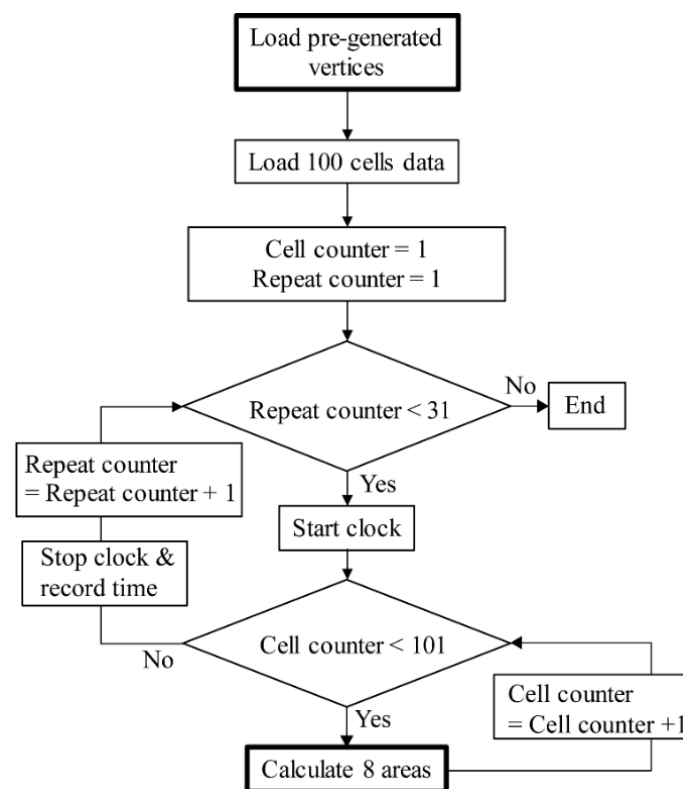


Figure 6. Flow chart for computations in three methods. This flow chart describes the basic structure of the computations performed by three methods. There are two parts that are method specific. They are indicated with thickened boxes. A set of pre-generated 100 cells is commonly used. Each method computed eight areas of every cell—a total of 800 area calculations. These calculations are repeated 30 times to measure the average computation time.

3. Results

Figure 7 shows the average percentage error and the average computation time of all three methods with up to eight cases as listed in Table 1. The blue lines in each graph show the average percentage error and the red graphs are the average calculation time. All methods show increasing computational time as the error decreases. The patterns of increment and decrement are approximately exponential. The exponential increment and decrement are expected since the number of vertices in the icosahedron-based vertex generation approximately goes up by four-fold from one generation to the next generation, while other methods followed the number of vertices in the icosahedron-based method. The triangulation method always showed better accuracy for equivalent vertices counts. The worst error is about 13% whereas the other methods recorded higher than 17% error. The lowest average error for the triangulation method was 0.015% but the other two methods could not achieve that low-level error even with a larger number of vertices. While the triangulation method achieved the best accuracy, it took much longer to produce more accurate results. Monte Carlo methods finished calculations in about 3 s in case number 8, whereas the triangulation method took approximately 350 s in case number 6. In the algorithm of the triangulation method, the triangulation of the domain depicted by D in Figure 3 is done by the Delaunay algorithm. This turned out to be computationally expensive. Comparing two Monte Carlo methods, the icosahedron-based method was superior to the sampling from a uniform distribution. Since the vertices themselves in the icosahedron-based method are almost uniform, the icosahedron-based method achieved better average accuracy than the Monte Carlo method using sample points from a uniform distribution. The stochasticity in the sample points presumably contributed to the bigger average error even though two methods use the same number of vertices. Interestingly, the computation time was also slightly but consistently longer in the case of the sample points method.

Figure 8 has the computational time against the average percentage error in the common logarithm scale for the three methods. The data points are from Figure 7. All plots show approximately decreasing linear graphs. Stochasticity of the sample points in the Monte Carlo method negatively impacts the convergence rate, resulting in the graph being more deviated from a linear graph than the other two methods in this log scale. These graphs show the performance of the three methods more clearly and allow us to compare them. According to these graphs, at the average error of 0.1% or higher, the icosahedron-based method provides quick ways to perform calculations. However, this may not be true at a lower average error level. The trend shown along the graph of the icosahedron-based method suggests this. If there were one or two more points to the left end of the current graph (red), the shape of the graph suggests that it will cross the blue plot. This means that the icosahedron-based method can be more costly than the triangulation method for the same average accuracy. Adding two more data points on the red graph requires the ninth and tenth-generation icosahedron-based vertices. We did not include those generation vertices because generating those vertices was extremely computationally costly. Judging by the patterns of graphs in Figure 8, using the ninth-generation or the tenth-generation vertices will be a not efficient method to adopt for finite volume simulations.

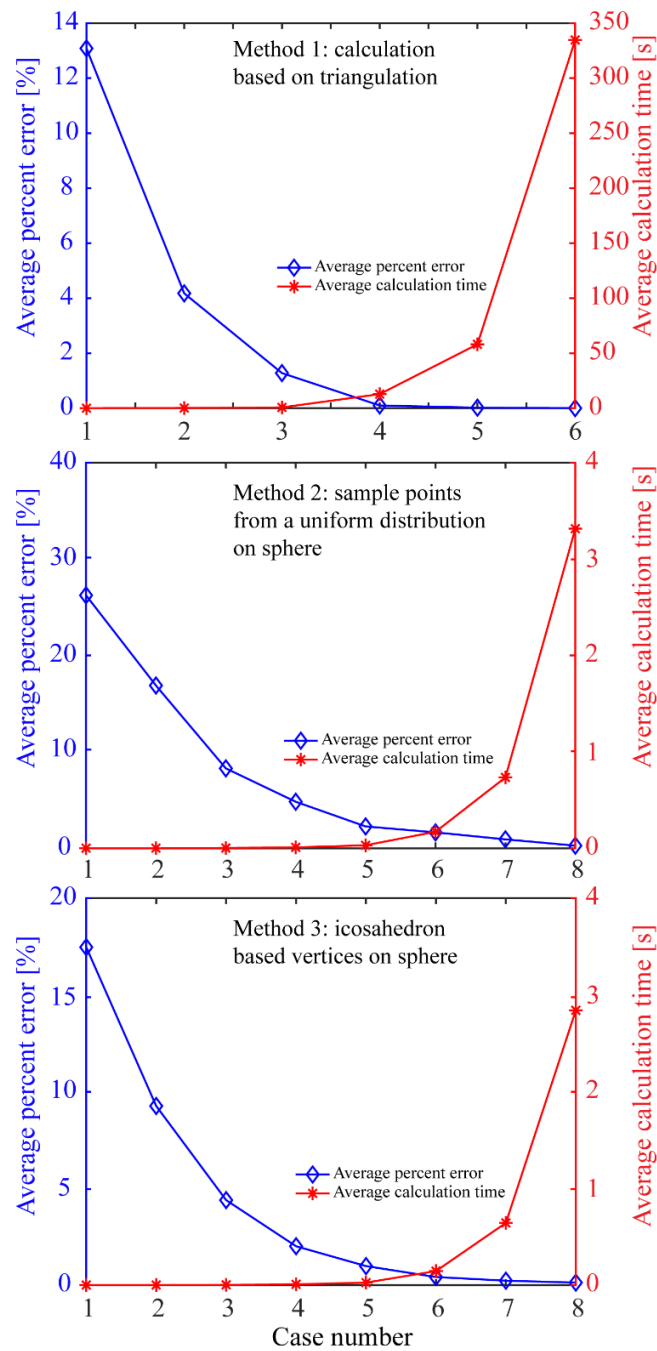


Figure 7. Average percent error and average calculation time from three methods. The blue plots with diamond markers are the average percent error which follows the vertical axis on the left side. Each data point is the average over 800 area calculations for 100 cells. The values from the triangulation method with 333,064 vertices on the disk domain are used as the exact values. The red plots with asterisks are the average calculation time that uses the axis on the right side. The average time is calculated from 30 repetitions of the 800 area calculations for 100 cells. The horizontal axes are all case numbers which are listed in Table 1.

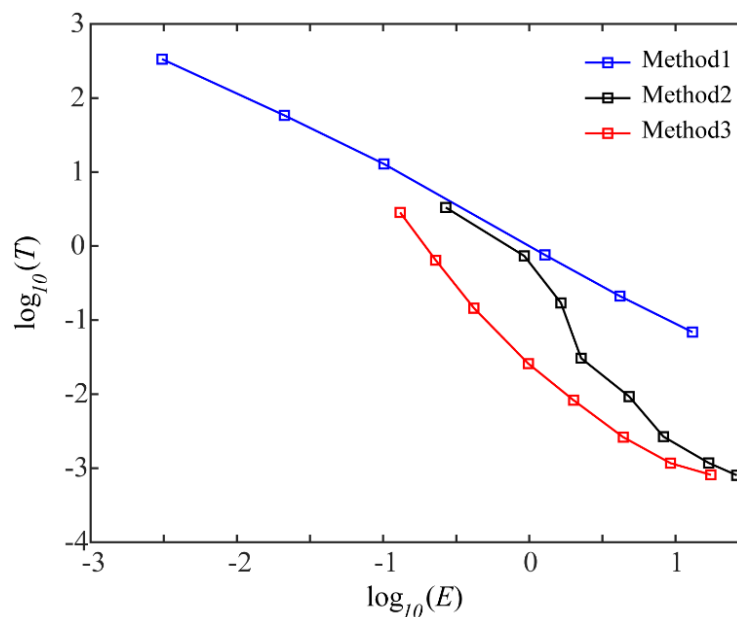


Figure 8. The average computation time (T) vs. the average percent error (E) in the common logarithm scale. The method 1 is the calculation based on triangulation. The method 2 is the Monte Carlo method using sample points from a uniform distribution on a sphere. The method 3 is the Monte Carlo method using the icosahedron-based vertices. The data points are from Figure 7.

4. Discussion

In this article, we have shown that the quasi-Monte Carlo method using the icosahedron-based almost uniform vertices on spheres can be a well-balanced method to calculate areas on a sphere intersecting with rectangular finite volumes. The quasi-Monte Carlo method has been reported to be a more efficient tool in various contexts [23,24]. We also showed that the triangulation method, in which the partial sphere surfaces need to be a graph of a two-variable function, can be used in conjunction with a linear system of equations to calculate the area on a sphere intersecting with rectangular finite volumes. In all methods discussed in this article, we utilized pre-generated vertices for efficiency and avoided the full generation of vertices for each sphere. Generating uniformly distributed vertices, especially with neighborhood relation, is computationally expensive. We tested three methods to capture the cell surface reaction contribution in a finite volume implementation. The icosahedron-based method would be our choice because it is capable of obtaining good accuracy in a short time. Moreover, the coding is simple to produce. These three methods are compared by point-wise comparison. Yet, the total (global) reaction rate remains the same across these methods because each cell's reaction rate contribution is proportional to the cell surface area (sphere area) and the cell surface was kept to be the exact value through the linear system of equations. The impact of these different methods on the diffusion-reaction system in the whole domain and in the long term was not within the scope of this article and we intend to address this in future work.

There is another potential advantage of icosahedron-based methods. Reactions on the cell surface include the reaction through cell surface receptors. When surface receptor locations are equally likely on the cell surface, the cell area in each finite volume can be used as the reaction contribution from the cell to the finite volume. It is, however, known that cell surface receptors can move around and be localized on the cell surface [25,26]. The icosahedron-based method is capable of tracing any region on the surface and measuring the area. Therefore, the icosahedron-based method will be suitable in cases where unequally distributed or localized surface receptors need to be considered.

The icosahedron-based vertices are generated by adding the projection of the midpoint at each edge of the previous generation. This is why almost uniformity is maintained. It is also why the number of vertices increases only by four-fold from the previous generation, whereas the number of

sample points from a uniform distribution can be arbitrary. There is a variation we may introduce to make icosahedron-based vertices other than those fixed numbers by the generation. Instead of putting the midpoint on each edge, two trisection points can be added on each edge. This will generate a different sequence of numbers for the vertex counts. Or, we can even proceed by mixing bisection generation and trisection generation. Trisection, however, appears to affect the uniformity a bit more.

Author Contributions: Conceptualization, M.K.; methodology, A.B., J.B., and M.K.; software, A.B. and M.K.; validation, A.B., J.B., and M.K.; writing—original draft preparation, A.B., J.B., and M.K.; writing—review and editing, A.B., J.B., and M.K.; visualization, M.K. All authors have read and agreed to the published version of the manuscript.

Acknowledgments: The abstract was presented as a platform in Platform: Computational Methods and Bioinformatics, Biophysical Journal.

Conflicts of Interest: The authors declare no conflict of interest.

Data Availability: The data used to support the findings of this study are available from the corresponding author upon request.

References

1. Rejniak, K.A. Homeostatic Imbalance in Epithelial Ducts and Its Role in Carcinogenesis. *Scientifica* **2012**, *2012*, 132978. [[CrossRef](#)] [[PubMed](#)]
2. Kim, M.J.; Reed, D.; Rejniak, K.A. The formation of tight tumor clusters affects the efficacy of cell cycle inhibitors: A hybrid model study. *J. Theor. Biol.* **2014**, *352*, 31–50. [[CrossRef](#)] [[PubMed](#)]
3. Kim, E.; Rebecca, V.; Fedorenko, I.V.; Messina, J.L.; Mathew, R.; Maria-Engler, S.S.; Basanta, D.; Smalley, K.S.M.; Anderson, A.R.A. Senescent fibroblasts in melanoma initiation and progression: An integrated theoretical, experimental, and clinical approach. *Cancer Res.* **2013**, *73*, 6874–6885. [[CrossRef](#)] [[PubMed](#)]
4. Müller, M.; Charypar, D.; Gross, M. Particle-Based Fluid Simulation for Interactive Applications. Available online: <https://matthias-research.github.io/pages/publications/sca03.pdf> (accessed on 23 December 2019).
5. Auer, S. Realtime particle-based fluid simulation. Ph.D. Thesis, Technische Universität München, München, Germany, 2009.
6. Goedhart, P.W.; Lof, M.E.; Bianchi, F.J.J.A.; Baveco, H.J.M.; van der Werf, W. Modelling mobile agent-based ecosystem services using kernel-weighted predictors. *Methods Ecol. Evol.* **2018**, *9*, 1241–1249. [[CrossRef](#)]
7. McLane, A.J.; Semeniuk, C.; McDermid, G.J.; Marceau, D.J. The role of agent-based models in wildlife ecology and management. *Ecol. Model.* **2011**, *222*, 1544–1556. [[CrossRef](#)]
8. Kennedy, W.G. Modelling Human Behaviour in Agent-Based Models. In *Agent-Based Models of Geographical Systems*; Springer: Dordrecht, The Netherlands, 2012; pp. 167–179.
9. Bonabeau, E. Agent-based modeling: Methods and techniques for simulating human systems. *Proc. Natl. Acad. Sci. USA* **2002**, *99*, 7280–7287. [[CrossRef](#)] [[PubMed](#)]
10. North, M.J.; Macal, C.M. *Managing Business Complexity: Discovering Strategic Solutions with Agent-Based Modeling and Simulation*; Oxford University Press: New York, NY, USA, 2007.
11. Ghaffarizadeh, A.; Friedman, S.H.; MacKlin, P. BioFVM: An efficient, parallelized diffusive transport solver for 3-D biological simulations. *Bioinformatics* **2015**, *32*, 1256–1258. [[CrossRef](#)] [[PubMed](#)]
12. Kim, M.J.; Baek, S.; Jung, S.H.; Cho, K.H. Dynamical characteristics of bacteria clustering by self-generated attractants. *Comput. Biol. Chem.* **2007**, *31*, 328–334. [[CrossRef](#)] [[PubMed](#)]
13. Kim, B.J.; Forbes, N.S. Single-cell analysis demonstrates how nutrient deprivation creates apoptotic and quiescent cell populations in tumor cylindroids. *Biotechnol. Bioeng.* **2008**, *101*, 797–810. [[CrossRef](#)] [[PubMed](#)]
14. Persson, P.-O.; Strang, G. A Simple Mesh Generator in MATLAB. *SIAM Rev.* **2004**, *46*, 329–345. [[CrossRef](#)]
15. Hicks, J.S.; Wheeling, R.F. An efficient method for generating uniformly distributed points on the surface of an n-dimensional sphere. *Commun. ACM* **1959**, *2*, 17–19. [[CrossRef](#)]
16. Muller, M.E. A note on a method for generating points uniformly on n-dimensional spheres. *Commun. ACM* **1959**, *2*, 19–20. [[CrossRef](#)]
17. Tashiro, Y. On methods for generating uniform random points on the surface of a sphere. *Ann. Inst. Stat. Math.* **1977**, *29*, 295–300. [[CrossRef](#)]

18. Sibuya, M. A method for generating uniformly distributed points on N-dimensional spheres. *Ann. Inst. Stat. Math.* **1962**, *14*, 81–85. [[CrossRef](#)]
19. Harman, R.; Lacko, V. On decompositional algorithms for uniform sampling from n-spheres and n-balls. *J. Multivar. Anal.* **2010**, *1*, 2297–2304. [[CrossRef](#)]
20. Poland, J. Three Different Algorithms for Generating Uniformly Distributed Random Points on the N-Sphere. Available online: <https://pdfs.semanticscholar.org/467c/634bc770002ad3d85ccfe05c31e981508669.pdf> (accessed on 21 December 2019).
21. Knuth, D.E. *The Art of Computer Programming. Volume 2: Seminumerical Algorithms*; Addison-Wesley: Boston, MA, USA, 1997.
22. Baumgardner, J.R.; Frederickson, P.O. Icosahedral Discretization of the Two-Sphere. *SIAM J. Numer. Anal.* **1985**, *22*, 1107–1115. [[CrossRef](#)]
23. Caflisch, R.E. Monte Carlo and quasi-Monte Carlo methods. *Acta Numer.* **1998**, *7*, 1–49. [[CrossRef](#)]
24. Schürer, R. A comparison between (quasi-)Monte Carlo and cubature rule based methods for solving high-dimensional integration problems. *Math. Comput. Simul.* **2003**, *62*, 509–517. [[CrossRef](#)]
25. Hislop, J.N.; Von Zastrow, M. Analysis of GPCR localization and trafficking. *Methods Mol. Biol.* **2011**, *746*, 425–440. [[PubMed](#)]
26. Steagall, R.J.; Yao, F.; Shaikh, S.R.; Abdel-Rahman, A.A. Estrogen receptor α activation enhances its cell surface localization and improves myocardial redox status in ovariectomized rats. *Life Sci.* **2017**, *182*, 41–49. [[CrossRef](#)] [[PubMed](#)]



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).