MODELING AND SIMULATION OF DISSOLUTION AND EROSION OF POROUS SOLIDS

BY DANIEL BRAIDO

A thesis submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Mechanical and Aerospace Engineering

Written under the direction of

Professor Alberto Cuitino

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

January, 2011

ABSTRACT OF THE THESIS

Modeling and Simulation of Dissolution and Erosion of Porous Solids

by Daniel Braido

Thesis Director: Professor Alberto Cuitino

Tablet dissolution modeling has seen multiple efforts from both empirical and physical approaches. We present an expandable 3-D Cartesian framework for modeling many of the physical processes involved in tablet dissolution, which allows for powerful model manipulation. The primary focus of this framework is the way in which the moving boundaries are handled, while the internal mechanisms represent one possible set of physical process interaction. The effects of external fluid shear and internal density profiles are compared for several cases including active and inactive internal processes.

Acknowledgements

I am grateful to Professor Alberto Cuitino for being patient with my progress, and helping to put me back on the right track when my course became too tangential to the work.

Dedication

I dedicate this thesis to my parents Louis and Susan Braido, and my wife Qin Braido. Nothing would be possible without their continued support.

Table of Contents

List of Abbreviations

C_w is the concentration of solvent
$C_w = 0$ at $\phi = 0$, the tablet/bulk fluid interface
t is the time in seconds
α_w is the penetration/diffusion coefficient. This value is assigned at initialization, and can vary throughout the tablet.
x is the position in the x direction
C_a is the volume based concentration of active particles
C_s is the concentration of dissolved active in the fluid in the cell
$\frac{\partial C_a}{\partial t}$ becomes the "source" term for the solute diffusion equation in the next section
t is the time
α_C is the dissolution coefficient which combines the effects of surface area, solvent concentration, and a solvent/solute specific coefficient
$\alpha_C = \alpha_{API} * C_w$
α_{API} is the dissolution coefficient of the specific API/solvent system being considered. It is an assigned value which represents the affinity of the API to enter solution.
C_s is the concentration of drug in solution, normalized from 0 to 1.
$C_s = 0$ at $\phi = 0$, the tablet/bulk fluid interface
t is the time in seconds
β is the diffusion coefficient, this value is assigned at initialization and can vary throughout the tablet
x is the position in the x direction in meters

Chapter 1

Introduction

Precision of dosage, mechanical and chemical stability and ease of storage and distribution make tablets the most popular drug delivery dosage form currently in use. A typical pharmaceutical tablet consists of several different powders, (such as actives, excipients, lubricants, glidants, disintegrants, etc.) compressed into a solid body. Usually administered orally, tablets disintegrate and dissolve in the gastro-intestinal tract, allowing for the pharmaceutical active to be absorbed in the body. It is the properties of this dissolution, which govern a tablet's performance as a drug delivery form. The ability to explicitly manipulate the factors involved in tablet dissolution is key to developing the necessary understanding for the next generation of engineered delivery systems. Despite its importance for the process of time-controlled delivery, the complexity of the phenomena and the multiple scales involved cause drug dissolution to remain poorly understood. In fact, currently, formulation design targeted at preset release profiles consists in large parts of trial and error with feed-back provided by large amounts of dissolution testing. The most common approach to designing a new formulation is to start with what is already being used and insert the new API. This limits the ability of designers to create a truly new or more effective product. In an attempt to ameliorate the state of the art, over the last several years, the FDA has championed the Process Analytical Technologies and Quality by Design initiatives. Aiming to provide a novel systematic approach for the design, analysis and control of manufacturing processes. QbD requires an in-depth, model-based understanding of the engineering and scientific principles involved and the identification of the variables, which affect product quality. After setting the targeted product profile Critical Quality Attributes need to be identified. These are the product qualities affecting its performance as a time-controlled drug

delivery device. The set of material characteristics and independent process parameters capable of affecting the CQAs is usually referred to as the set of critical process parameters. Determination of the range of each CPP that produces acceptable product allows for the establishment of a process design space. While experimental methods, such as dissolution testing can still serve an important role as a control mechanism, product quality cannot be tested in - it needs to be designed along with the product, on the foundation of physics and engineering-based models of the system. The last several decades have seen tremendous progress in the implementation of numerical models for the study of pharmaceutical tablet dissolution. Of these, the simplest ones were based on zero-order kinetics, approximating a slow release system with no disagregation, which never reaches equilibrium conditions [1]. The full history of dissolution modeling can be read in recent review papers[2, 3], but a brief recount is given here. As the need to consider additional phenomena became apparent, first-order kinetics models were devised, taking into account the effects of the concentration of the dissolved drug in a diffusion layer around the tablet - the Noyes-Whitney equation, dissolution rate dependence on the solid area available for dissolution - the Brunner equation and the proportionality of the release rate to the amount of drug remaining in the solid (accounting for diminishing tablet dimensions) - the Hixson Crowell equation [4, 5]. Further insight into the importance of tablet microstructure lead to the inclusion of the location of individual active particles inside the solid form. The Higuchi equation (modified by Cobby to apply to cylindrical tablets) models the dissolution of drug particles dispersed in a uniform, homogeneous matrix. It accounts for tablet porosity, volume accessible to the dissolution media and an effective diffusion coefficient through the pore channels. The model was augmented by Seki [6] to handle non-homogeneous matrices. Progression in model development has recently yielded simulation tools accounting for tablet structure and geometric characteristics - the Korsmeyer-Peppas model[7], as well as surface erosion (handled as a kinetic process) - the Hopfenberg model[8]. Current dissolution models have attained high levels of sophistication, simulating the change in diffusivity caused by the gradual penetration of the solvent into the tablet and the subsequent swelling of the polymer matrix. The phenomena considered also include, active

dissolution and the changes in porosity it produces as well as tablet surface erosion [9, 10, 11]. We look to expand on these models by incorporating all the elements they do into an expandable 3-D framework which allows for a more discrete representation of the tablet and environmental conditions. Such a framework allows for simulation of changing dissolution environments like and in vivo process. Other research in the field has placed increased interest in the effects of the fluid flow around the tablet, with an emphasis on the effects of shear disintegration[12, 13]. The work herein provides a basis for that expansion.

1.1 Overview of Thesis

The goal of our research work is to develop a numerical scheme for studying the evolution of porous solids in a dissolving media. This model is developed as a combination of the different processes involved in the disintegration and dissolution of a porous solid. The thesis illustrates the contribution of each of these phenomenon separately and then details the tools and methodology used in superposing individual contribution to obtain the overall model of dissolution. The first chapter of this thesis entitled **Introduction** gives an overview of the numerical scheme and provides background information about the previous work done to model the different processes involved as well as equations which empirically model tablet dissolution as a whole. The second chapter, which is titled **Formulation** describes the separate components used in our numerical scheme. The first section is a detailed discussion of the level set representation that combines velocity components from diffusion and erosion to obtain the total velocity driving the solid/fluid interface, including how boundary conditions are applied over an irregular boundary using a ghost node based method. This is followed by a description of the processes of solvent penetration, particle dissolution and solute diffusion which occur past this interface. This chapter outlines the general framework which is later used to solve specific problems. The third chapter entitled **Applications** is where problems involving different types of tablets are solved using the model. The main focus is placed on the importance of the moving boundaries in tablet dissolution. Also, the effect of non-homogeneous density and surface shear profiles on drug release are compared. In

this section the release profiles of a model tablet is compared under different erosion conditions; homogenous surface erosion, a density based erosion profile, fluid shear based erosion profile and a combined response. A detailed analysis of obtained results can be found in the final chapter entitled **Conclusion and Future Scope** followed by a discussion about the scope of current research for future work and its applications to various problems of practical importance. Finally, an **Appendix** is included where the algorithm and code for the numerical model can be found.

Chapter 2

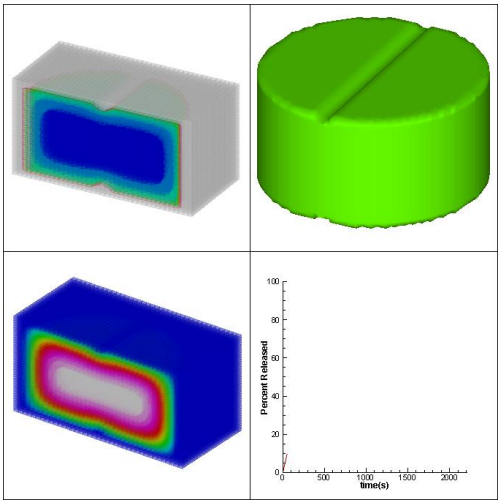
A modeling framework for predicting release profiles due to coupled phenomena: surface erosion and bulk diffusion

2.1 Drug (active substance) release in physiological environments

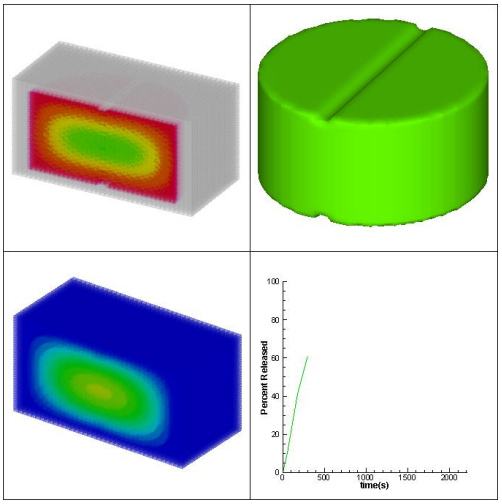
When pharmaceutical solids such as tablets are ingested a set of processes is triggered that results in the release of the active substance component. Once has reached the stomach, unless there is a coating layer, the surface of the tablet immediately begins absorbing the surrounding fluid. As the excipient matrix absorbs the fluid, active drug particles are dissolving from inside this matrix. The resulting drug solute then diffuses out past the tablet surface until it reaches the surrounding bulk fluid and becomes mixed. As this is occurring, the excipient matrix often swells and fractures resulting in movement of the tablet surface and also the formation of new surfaces. A side by side comparison of the different processes is shown in Figure 2.1 with time based progression.

2.2 Modeling Framework

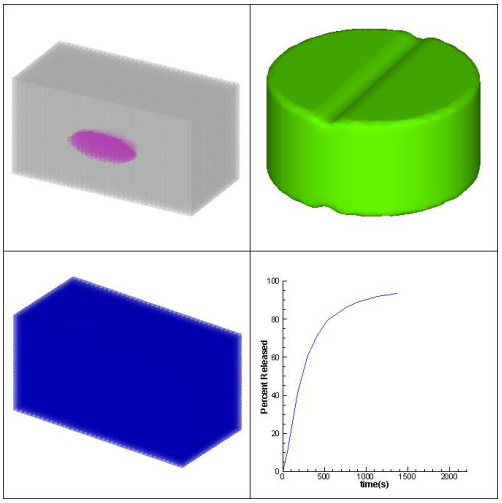
The simulation tools presented in this thesis allow for considering coupled mechanisms of dissolution and disintegration as observed in actual experiments. The boundary conditions for these processes are governed by the location of the tablet/bulk fluid interface. Tablet erosion makes this a moving boundary problem. A level set formulation is used to track to evolution of the interface and calculate the necessary boundary conditions. In this study we consider solvent penetration governed by diffusion, with a predetermined solvent penetration coefficient which controls the rate at which the fluid progresses. This process is fully described in the Solvent Penetration section. Once fluid has reached a cell, the active particles inside that cell begin dissolving as



(a) 60s



(b) 300s



(c) 1380s

Figure 2.1: Snapshots of simulated tablet evolving shape, solvent concentration, active concentration and drug release profiles.

described in the Active Dissolution section. Although the active concentration value is uniform throughout the cell, the active particles' rate of dissolution is calculated as one dimensional diffusion from a sphere using the current fluid concentration, total surface area of active particles and a prescribed particle dissolution coefficient. As the active particles enter solution, the concentration of solute increases. The solute diffusion is then calculated using Fick's Second Law, and is governed by a prescribed solute diffusion coefficient. The details of this process are described in the Active Diffusion section.

2.2.1 Modeling Evolving Topologies

One of the primary concerns of the numerical simulations is proper evolution of the tablet fluid interface. Previous work has shown scaling between identical formulations compacted in different geometries[14]. There are numerous schemes which have been developed, both Lagrangian and Eularian, to track changes of a surface. As the problem of tablet dissolution entails tracking complex geometry changes based on various external and internal effects, it is important to use a surface tracking method which is both stable and inexpensive. Lagrangian methods often rely on complicated meshes which are adapted over time. This remeshing can result in a highly distorted grid shape, which can in turn negatively influence the time-step which is based on the smallest grid size. The simplicity and consistency of fixed-grid method has focused our attention on Eularian methods. The most commonly used Eularian methods of surface tracking are the Volume-of-Fluid method by Hirts and Nichols[15], the phase-field method and the level set method by Osher and Sethian. Level set functions have been shown to produce smooth evolution of surfaces with complex geometry and are extremely well suited for use on cartesian based grid structures. There has been extensive work[16] employing levels sets to track surfaces for the purpose of modeling fluid dynamics, crystal growth and chemical deposition. The ability of level sets to operate based on a single velocity function makes this framework easy to expand as well, since all of the effects considered can be combined into a single function representing the velocity of the surface. Inclusion of additional factors will have no effect on the level set implementation and can be

incorporated directly into the velocity function.

In addition, the formation of new surfaces are handled implicitly by this function, which will become important as the internal effects of swelling are added to the model and fractures and cracks develop. The methodology used was adapted from a framework developed by Kinjal Dhruva.[17]

Level Set Method

The Level Set Method is implemented using an explicitly defined signed distance function. The tablet fluid interface, herein referred to as Γ exists where $\phi(\dot{\mathbf{X}}, t) = 0$. A basic 2-dimensional representation of the evolution of an object with a circular cross section is shown in Figure 2.3. The function is defined in such a way that:

$\phi = 0$ at Γ

$\phi > 0$ inside Γ

$\phi < 0$ outside Γ

The function values range from positive to negative, with negative values being exterior and positive values being interior. It is important to note that none of the actual discretized points will be equal to 0. That is to say that the interface always lies between nodes, and never specifically at any node on the fixed grid. This characteristic is inherent to the level set method and will later be used in proper implementation of the boundary conditions.

The evolution of the interface is based on the function:

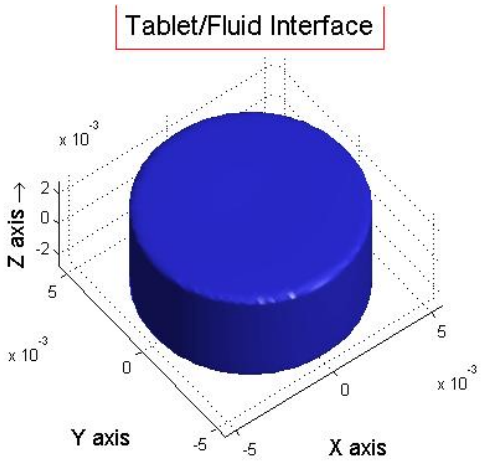
$$\phi(\dot{\mathbf{X}}, t) = 0 \tag{2.1}$$

The derivative of this equation with respect to time can be determined using the chain rule to be:

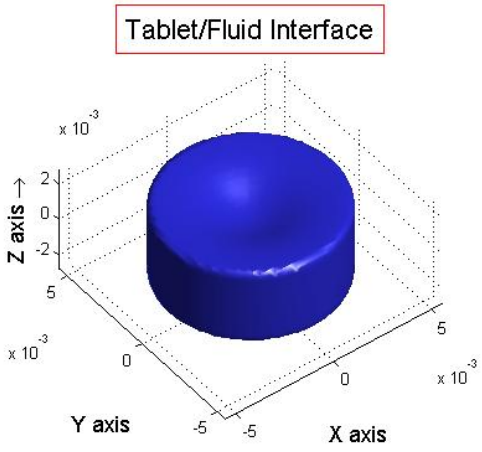
$$\frac{d\phi}{dt} + \frac{d\phi}{d\mathbf{X}} \cdot \frac{\mathbf{X}}{dt} = 0 \tag{2.2}$$

Which can be rewritten as:

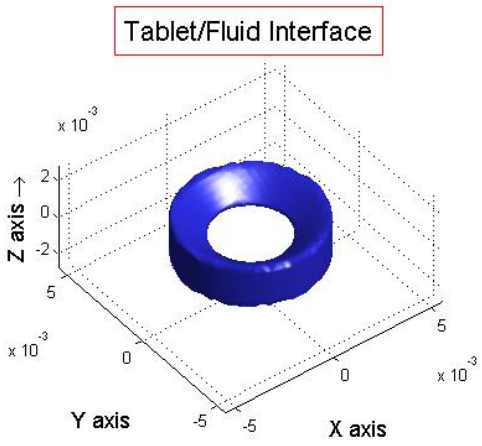
$$\phi_t + \nabla\phi \cdot \dot{\mathbf{X}} = 0 \tag{2.3}$$



(a) Dense Shell Erosion Beginning



(b) Dense Shell Erosion Intermediate



(c) Dense Shell Erosion Final

Figure 2.2: The tablet shapes shown in the figures above are the location of the zero level set as time progresses. The formation of the central hole is handled implicitly.

For our purposes it is only necessary to move the surface in the normal direction. All tangential motion is neglected. Assuming a normal velocity has been previously calculated, in our case one representing tablet erosion, it can be expressed as v_n. It can then be shown that:

$$\dot{\mathbf{X}} \cdot \mathbf{N} = v_n \tag{2.4}$$

Where:

$$\mathbf{N} = \frac{\nabla \phi}{|\nabla \phi|} \tag{2.5}$$

Therefore:

$$\mathbf{v} = \dot{\mathbf{X}} \cdot \frac{\nabla \phi}{|\nabla \phi|} \tag{2.6}$$

This representation means that equation(1.2.3) can be rewritten as:

$$\phi_t + v_n |\nabla \phi| = 0 \tag{2.7}$$

Which can in turn be adapted to our present use in a timestep based discretization as:

$$d\phi = -v_n |\nabla \phi| \triangle t = 0 \tag{2.8}$$

This methodology assumes v_n exists at all points inside the computational boundary. In most cases of tablet dissolution the velocity function will only really have meaningful, non-zero values near the interface, but this only serves to simplify the calculations, not render this assumption false. v_n has been explicitly defined in the presented models. The values assigned range from representing uniform erosion to density based erosion as well as an erosion profile based on fluid shear calculations from the fluid dynamics of the bulk solvent, and can also incorporate swelling of the excipient matrix as a result of solvent penetration or increasing porosity as a result of particle dissolution, though this aspect is not explored here.

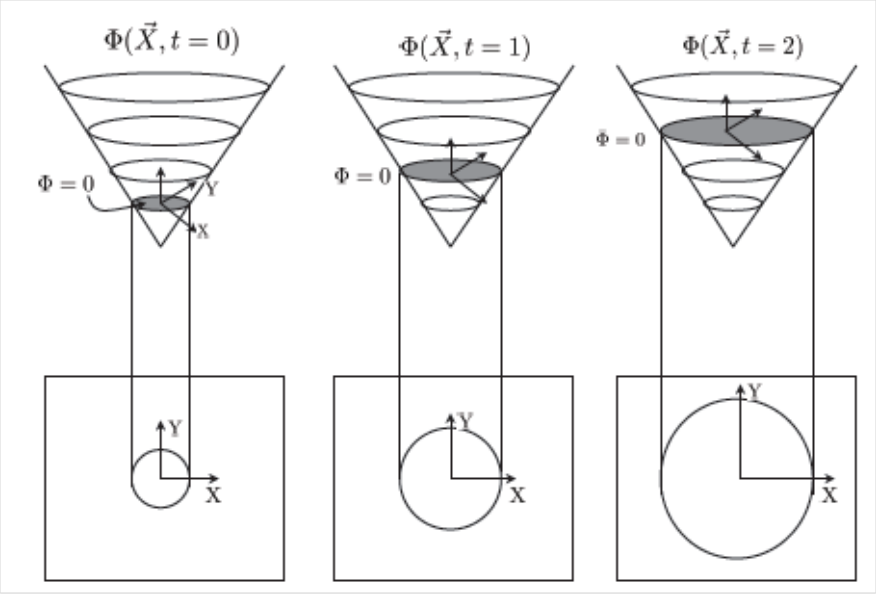


Figure 2.3: The figure above is a simple representation of the level set evolution of a circular tablet in two-dimensions. If One were to progress from left to right, the operations would represent a swelling of the interface. The two-dimensional figure(circle) is represented by a three-dimensional object(cone). $\phi = 0$ represents the interface, evident from the resulting projection on the plane.

Normal and Curvature

When using level sets to define interface locations it is necessary to determine the normals to the surface and the curvature. These values are important for proper evolution of the level set function, as well as for proper application of the boundary conditions of the nested equations. Fortunately, the use of a fixed cartesian grid makes these calculations extremely simple. The normal of the level set function can be expressed as:

$$n_{x,y,z} = \frac{\nabla\phi}{|\nabla\phi|} = \frac{\phi_x, \phi_y, \phi_z}{(\phi_x^2 + \phi_y^2 + \phi_z^2)^{\frac{1}{2}}} \tag{2.9}$$

and the curvature is given by the equation: While these formulas may appear complex, they are easily and inexpensively computed using a few lines of matrix algebra. This represents a great savings over any lagrangian method, as such computations are often complex and expensive.

Indexing

As the problem will be represented on a fixed cartesian grid, and the interface exists between grid nodes, it is necessary to determine which nodes are closest to this interface. As mentioned before, the tablet/fluid interface is considered to exist everywhere the level set function is equal to 0. Thus the nodes of interest are those where any nearest neighbor has a level set value of opposite sign. While the majority of nodes, 'regular' nodes, exist purely inside or outside the tablet interface, some nodes lie within less than a grid space from the interface. These 'irregular' nodes must be identified and properly indexed. These are the nodes which need to be given special attention, especially when it comes to proper implementation of the boundary conditions. The use of a fixed cartesian grid makes the determination of such nodes quite simple. A discretized scheme in 2 dimensions can be simply stated as:

$$\phi_{max} \cdot \phi_{min} \{ < 0, irregular > 0, regular$$

where,

$\phi_{max} = max\{\phi_{i,j+1}, \phi_{i,j-1}, \phi_{i+1,j}, \phi_{i-1,j}\}$

$\phi_{min} = min\{\phi_{i,j+1}, \phi_{i,j-1}, \phi_{i+1,j}, \phi_{i-1,j}\}$.

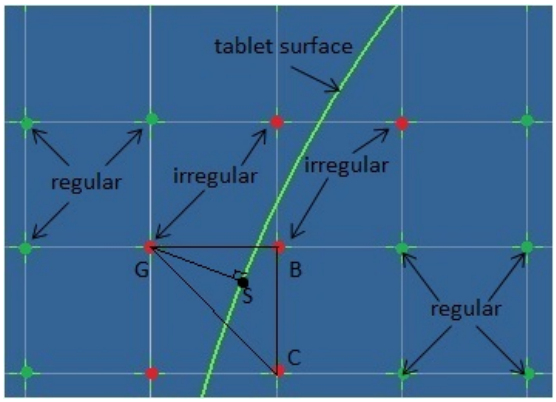


Figure 2.4: A simplified portion of a 2D mesh noting areas of regular and irregular grid nodes as considered by indexing algorithm.

For both regular and irregular points, their relative position of inside or outside the surface can still be determined simply by the sign of the level set function value at that location. Figure2.4 shows how the grid nodes are defined surrounding the interface location.

Immersed Boundary Conditions

In order to properly update all of the equations, the boundary conditions associated with the problem statement must be properly implemented. The level set tracks the propagation of the interface, as well as information about its shape and normal vectors pointing to the interface.

Boundary conditions were handled by assigning temporary values at certain specialized nodes in each of the concentration matrices. This process, known as ghosting, was to ensure that the concentration gradient remained continuous across the tablet fluid interface. By assigning temporary values to the set of nodes located just outside the interface, it is hoped that the concentration can be calculated more effectively. Three different methods of boundary condition implementation were compared: simple

value assignment, one-dimensional interpolation, and a layered three-dimensional to one-dimensional interpolation method.

For the value assignment method, if a node is identified as a ghost node it is assigned a value. This value was either the concentration value at the interface, or the concentration of the bulk fluid. For the one-dimensional method, the ghost node value is determined based on the value of the interface concentration and the concentration of the internal neighboring node nearest to the surface. The value at the ghost node is a simple linear interpolation using the two values.

The most complicated method we used to assign ghost node values was actually a layered method. The process began when a node in need of ghosting was identified. The vector from that node normal to the surface was first analyzed to determine the most appropriate style of interpolation with which to begin. If the normal vector pointed entirely in one dimension, the one-dimensional interpolation method previously described was used. However, if the vector pointed in two or three dimensions, a form of triangular interpolation was utilized. The two methods are fairly similar, with the three-dimensional method being an extension of the two-dimensional method. All of these methods assume that the concentration is known at the tablet/fluid interface.

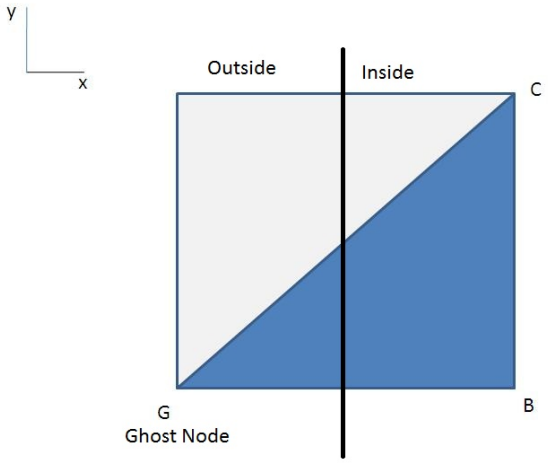


Figure 2.5: The edges of the square represent the edges of a 2d cell. The triangular area is the area used for 2-D ghosting interpolation.

If it was determined that the normal vector lies in only two of the three dimensions being considered, the two-dimensional interpolation method is used. The two non zero dimensions of the normal vector are used to determine the associated nodes used for interpolation. The three nodes make up a triangle. The cartesian coordinates of these nodes are used in conjunction with the cartesian coordinates of the point on the surface closest to the ghost node. The location of the surface point is determined using the level set value and cartesian coordinates at the ghost node as well as the associated normal vector. These cartesian coordinates are then used to form the "natural coordinates" of the point on the surface being considered, which serve both as interpolation coefficients and ensure the surface point lies within the area of interpolation. The natural coordinates are calculated as a series of vector dot product operations, as indicated below:

$$S_B = GC \cdot GS, \quad S_C = GB \cdot GS, \quad S_G = BC \cdot BS$$

$$\triangle_B = GC, \quad \triangle_C = GB, \quad \triangle_G = BG \cdot BC$$

$$N_B = \frac{S_B}{\triangle_B}, \quad N_B = \frac{S_C}{\triangle_C}, \quad N_B = \frac{S_G}{\triangle_G},$$

where N_B is the natural coordinate for node B. If this value is positive, or more directly, if the vectorS_B points in the same direction as the vector$\triangle_B$, the point lies inside the interpolation area. N_B represents the distance of the point on the surface normal to the opposing face of the triangle divided by the distance of point B normal to the opposing face of the triangle. The distance of B to the opposing face is used as a normalizing factor, so the coordinates actually act as interpolation coefficients. The value at the ghost node is determined by the equation:

$$C_S = N_B C_B + N_C C_C + N_G C_G \tag{2.10}$$

Since the concentration at the tablet fluid interface is already known, the equation can

be rewritten as

$$C_G = \frac{C_S - N_B C_B - N_C C_C}{N_G}, \tag{2.11}$$

thus, solving for the ghost node concentration.

If it is determined that the normal vector points in all three dimensions, a three-dimensional interpolation scheme is used. This method is a direct extension of the previously described two-dimensional method. Again using the ghost node as the base point, the vector normal to the surface is used to select the other three nodes whose values will be used to interpolate the temporary concentration for the ghost node, as shown below:

$$S_G = (CB \times CD) \cdot CS, \quad S_B = (CD \times CG) \cdot CS, \quad S_C = (GD \times GB) \cdot GS, \quad S_D = (CG \times CB) \cdot CS$$

$$\triangle_G = CG, \quad \triangle_B = CB, \quad \triangle_D = CD,$$

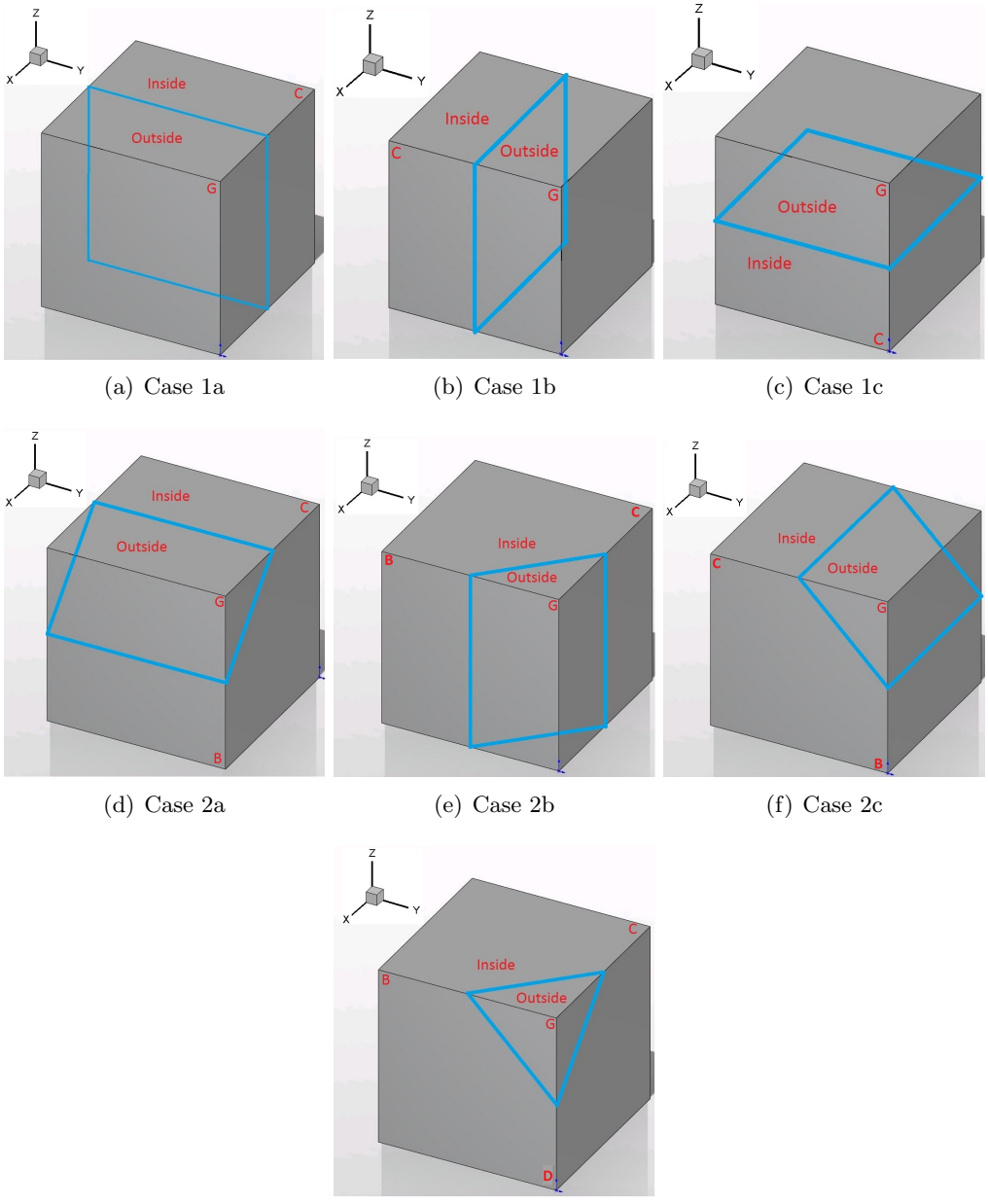
$\triangle_C$ is computed by calculating the equation of the plane across from node C, followed by calculating the distance to point C along the normal.

$$N_G = \frac{S_G}{\triangle_G}, \quad N_B = \frac{S_B}{\triangle_B}, \quad N_C = \frac{S_C}{\triangle_C}, \quad N_D = \frac{S_D}{\triangle_D}$$

These normal coordinates are then used in a similar fashion to determine the ghost node concentration, with the reconfigured equation being:

$$C_G = \frac{C_S - N_B C_B - N_C C_C - N_C C_D}{N_G} \tag{2.12}$$

The goal of the layered three-dimensional method is to provide the most accurate representation of the concentration gradient across the surface in the quickest way possible. Figure 2.6 is a table showing the different interface situations and their associated ghosting schemes.



(a) Case 1a (b) Case 1b (c) Case 1c

(d) Case 2a (e) Case 2b (f) Case 2c

(g) Case 3

Figure 2.6: This table shows the 7 different interface situations possible in the model. Each one is associated with its respective ghosting type. G indicates the node for which a ghost value is to be calculated.

2.2.2 Solvent Penetration

An important factor in tablet dissolution is the penetration of solvent into the excipient matrix. In the physical process of dissolution, the surrounding solvent enters the excipient matrix of the tablet similar to a sponge soaking up water. While the process of solvent entering the tablet takes longer than water entering a sponge, the process by which both happen is very similar. The small voids between particles in a tablet allow for water to enter and proceed through the tablet, in many cases, to the center of the tablet. As the tablet enters the solvent media, or after the simulation is started, the tablet begins to soak up the surrounding fluid. In many cases this will result in the tablet swelling and possibly fracturing. While the present iteration of the numerical simulation does not incorporate swelling or fracture, the bulk fluid is still considered to penetrate the excipient matrix. The model considers the natural porosity of the tablet to exist as a network of channels, some open and some closed. Solvent penetration is represented as a diffusion based process, whose rate is manipulated via a diffusion coefficient which represents the porosity, hydrophobicity, and tortuosity of the compacted particles. The method is based on Fick's second law, and incorporates the effects of a non-uniform diffusion coefficient:

$$\frac{\partial C_w}{\partial t} = \nabla . \alpha_w \nabla C_w \tag{2.13}$$

This equation can be expanded in one dimension using the chain rule to:

$$\frac{\partial C_w}{\partial t} = \frac{d\alpha_w}{dx}\frac{dC_w}{dx} + \alpha_w \frac{d^2 C_w}{dx^2} \tag{2.14}$$

Where:

C_w is the concentration of solvent

$C_w = 0$ at $\phi = 0$, the tablet/bulk fluid interface

t is the time in seconds

α_w is the penetration/diffusion coefficient. This value is assigned at initialization, and can vary throughout the tablet.

x is the position in the x direction

Full expansion results in a summation of the right side of the equation for all included dimensions, in our case x, y and z. The incorporation of a non-uniform diffusion coefficient allows for better tablet definition especially when exploring the effects of channeling. The diffusion is modeled using a centrally based finite-differencing method. The equation uses the current solvent concentration values of the six surrounding nodes and a prescribed coefficient to determine the rate of change of the solvent concentration with respect to time. The prescribed diffusion coefficient represents the aggregate effects of porosity, hydrophobicity and tortuosity of the excipient matrix. Since physical tablet structure may not have uniform porosity, the function is so defined that each cell represented has its own solvent diffusion coefficient. This diffusion coefficient represents a combination of the physical factors affecting the rate of solvent entering the tablet. The variation of α_w is explored later in the paper. It has been argued that solvent penetration into a polymer matrix would be better represented by Case II diffusion, but in the case of the referenced physical experiments, solvent penetration of the tablets would best be described as anomalous diffusion.([18, 19]) This type of diffusion can be modeled by augmenting the local coefficients of solvent penetration in proportion to their current solvent concentration. By increasing the local penetration coefficients a sharper penetration front is effected more closely matching the observed physical response. While this is still not the exact solution, it is a quick way to get useful results. Furthermore, depending on the specific physical system, some solvent penetration cases are best handled by Fick's Law based modeling.[20] It is important to note that other modes of solvent transport can easily be added, as the solvent penetration portion is represented by its own module in the algorithm. In fact, this interchangeability is part of the overall goal of the modeling platform presented here. Some early simulations were performed showing the effects of the different diffusion types on the model output was nominal at relatively fast penetration rates, especially when paired with an eroding interface and no swelling. As this was the case, Case 1 diffusion was implemented for the simulations as this reduced the number of variables to be estimated. The

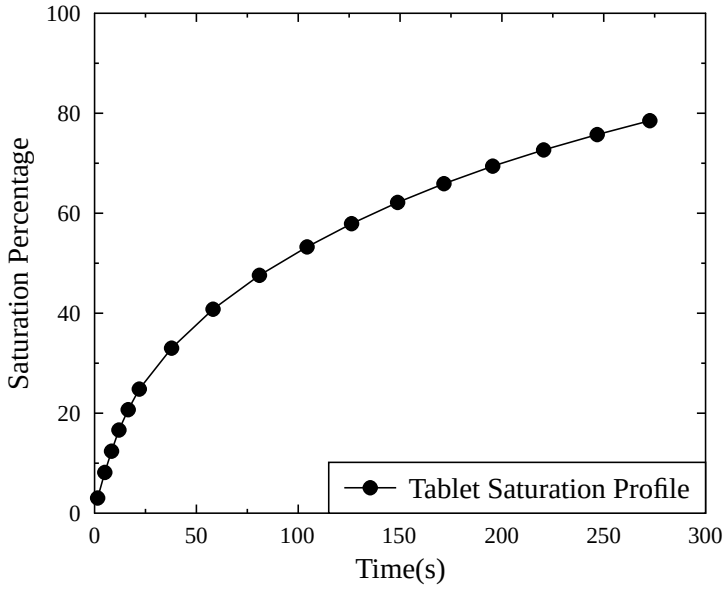


Figure 2.7: Time evolution of the solvent penetration into the matrix. Scale is from completely dry(0) to completely wet(100). In this simulation case 1 solvent diffusion with s$\alpha_w of .001 \frac{mm^2}{sec}$ is considered. The corresponding time evolution of the active substance dissolution is presented in 2.8.

evolution of solvent penetration through the excipient matrix is for a specific case, $\alpha_w = .001 \frac{mm^2}{sec}$, is shown in Figure 2.7 At the onset of the simulation, the tablet is assumed to be completely dry, and everywhere outside the tablet is pure solvent. The initial conditions assign a concentration of 0 in the inside and of 1 at the boundary. The conditions at the boundary are imposed using the ghosting scheme described above, where a concentration larger than 1 is specified at the ghost (external to the domain of analysis) to property enforce the concentration of 1 at the boundary. Note that grid nodes are not necessarily on the boundary. During the time evolution, the external concentration is held constant at 1, as it is assumed the tablet is surrounded by pure solvent.

2.2.3 Dissolution and Diffusion of Active Substance

The dissolution and release of drug form the excipient matrix is handled via two separate processes, active dissolution and active diffusion. The former handles the physics which describe the dissolution of drug particles from their solid form into solute. The diffusion of this solute is handled by a second set of equations which tracks the localized concentration of the solute until it has diffused past the tablet fluid interface. The API solid concentrations are what are used when calculating the amount of drug which has been released from the system at any given time, while the active diffusion is more of a physically relevant clearing mechanism. The rate at which solute diffuses from the system can be release rate limiting, but such cases were not considered in this thesis.

2.2.4 Active Dissolution

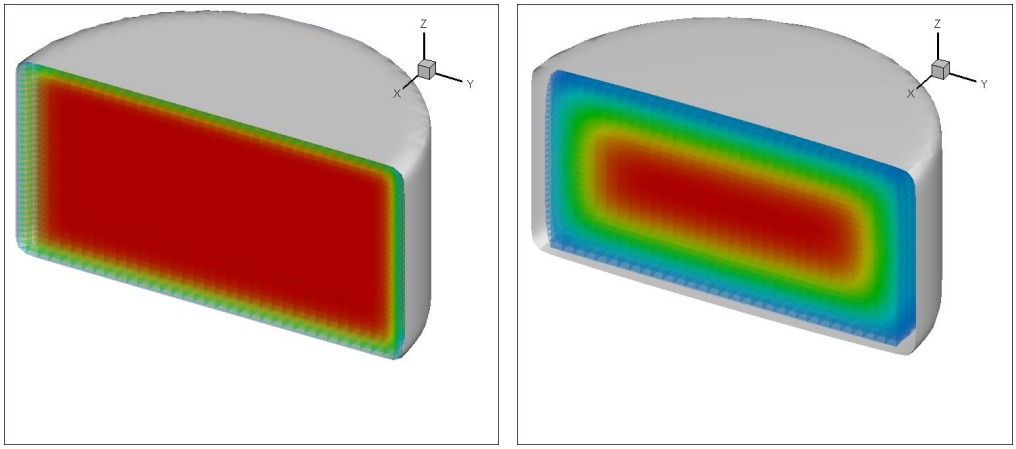
Particle dissolution is calculated using a set of parameters which represent the average values of particle size, shape, and concentration for each cell. The size, shape and number of active particles is initially defined for every cell located inside the tablet. At the beginning of the simulation, each grid cell inside the tablet is assigned a number of active particles of known size. The number and size of the active particles is used to produce a volume based concentration of active particles. While the concentration flux is determined based on the bulk active concentration of each cell, the volume of the individual particles is considered to decrease with this bulk concentration. This results in a decrease in surface area of the particles which affects their bulk dissolution rate. The representative equation is similar to the Brunner equation[21]. The rate of particle dissolution is based on another extrapolation of Fick's Law which incorporates the solvent penetration, and operates on the total particle volume inside the cell:

$$\frac{\partial C_a}{\partial t} = \alpha_C (C_a - C_s), \tag{2.15}$$

where C_a is the volume based concentration of active particles, C_s is the concentration of dissolved active in the fluid, $\frac{\partial C_a}{\partial t}$ becomes the "source" term for the solute diffusion equation in the next section, t is the time, $\bar{\alpha} = \alpha_{\text{API}} S_p C_w$ is the dissolution coefficient

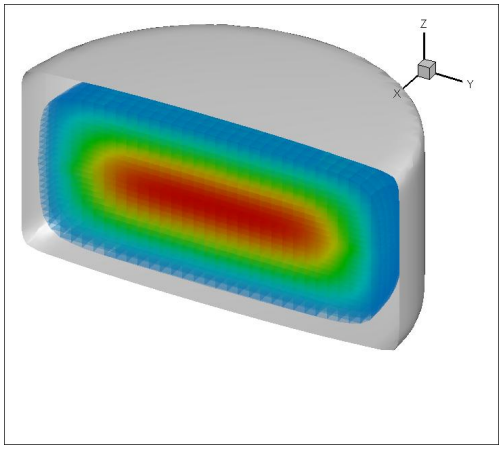
which combines the effects of surface area, solvent concentration, and a solvent/solute specific coefficient, with S_p the surface area of the particle, C_w the concentration of solvent and α_{API} the dissolution coefficient of the specific API/solvent system being considered, which is a value accounting for the affinity of the API to enter solution. The units of α_{API} are $\frac{mm^{-2}}{s}$, which may seem odd, but is a result of the dimensionless formats in which the solvent and API concentrations are considered. As more realistic systems are considered, with experimentally determinable physical constants, the concentration units can be adjusted to mass/volume.

Although the concentration flux calculations are made using the bulk concentration of each cell, the evolution of the individual particles is still considered. As the concentration decreases, the size of the particles in each cell must also decrease. The change in API is used both to update the remaining volume of API in each cell and the local solute concentration. Solute concentration will be further explained in the next section. This equation assumes that the active particles remain in the grid cell in which they were initially defined. That is not to say that solid drug particles are not be able to move in dissolving tablets, simply that such an effect is not handled via this particular model. In decreasing the volume of the individual particles in a cell, we assume that all of the particles of a given size in that cell dissolve at the same rate. Therefore in a monosized active distribution, all of the active particles in a given internal cell will be reduced by the same volume. This reduction in volume has a large effect on the rate of dissolution as the dissolution coefficient takes into account the surface area of the particles. In this equation, α_C is determined by three factors; combined surface area of active particles in the cell, solvent concentration in the cell and an active ingredient/solvent system based dissolution constant. This results in a diffusion coefficient which can change not only based on the location inside the tablet, but also with every timestep. As time progresses the concentration of active particles decreases and particle volume is reduced until the particles in a cell have decreased below a threshold volume. Once the particles in a given cell have reached the threshold volume, they are then considered completely dissolved and are removed from the simulation. Figure **??** shows the resultant concentration distributions of a model tablet as time progresses.

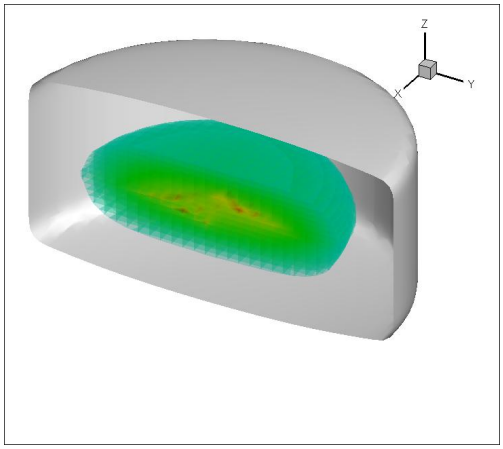


(a) initial

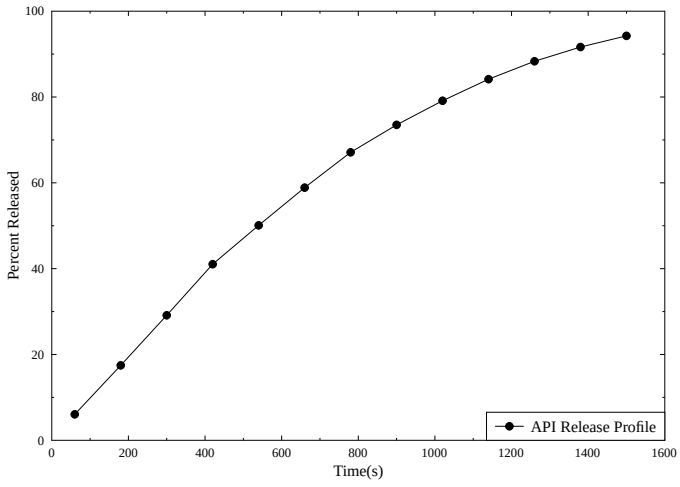
(b) intermediate



(c) intermediate2



(d) final



(e) API release profile

Figure 2.8: The evolution of API concentration inside a tablet undergoing uniform surface erosion.

Diffusion of Active Substance

Once the active particles have entered solution the solute proceeds to diffuse out of the tablet and into the bulk solution. As the API particles dissolve, the associated volume of drug dissolved is converted to a solute concentration inside the tablet. This solute concentration must diffuse out of the excipient matrix to the bulk solution. The process is very similar to the solvent penetration into the tablet, and thus is handled in much the same mathematical formulation.[22]

$$\frac{\partial C_s}{\partial t} = \nabla . \alpha_s \nabla C_s + \text{Source} \tag{2.16}$$

This equation can be expanded in one dimension using the chain rule to:

$$\frac{\partial C_s}{\partial t} = \frac{d\beta}{dx}\frac{dC_s}{dx} + \beta\frac{d^2C_s}{dx^2} + \text{Source} \tag{2.17}$$

Where:

C_s is the concentration of drug in solution, normalized from 0 to 1.

$C_s = 0$ at $\phi = 0$, the tablet/bulk fluid interface

t is the time in seconds

β is the diffusion coefficient, this value is assigned at initialization and can vary throughout the tablet

x is the position in the x direction in meters

Like the solvent penetration, the solute diffusion is solved using a modified version of Fick's diffusion equation. Each cell inside the tablet represents a generating source, while the bulk solution outside the tablet is treated as a perfect sink. This assumption has been previously used and verified by Feldman in 1967 [4]. At every timestep the contributions from the particle diffusion step are added to the solute concentrations, then the solute concentrations are updated using a basic finite-diffencing scheme like the solvent concentrations. Currently, the solute concentration is calculated only as a rational clearing mechanism for the dissolved active, which results in some mild rate

limiting of particle dissolution, as opposed to defining the drug release profile.

2.3 Summary

The framework described herein is capable of handling dissolution modeling for a fairly large range of cases. The use of level sets to track the tablet/fluid interface allow for multiple model geometries of varying complexity. The formation of increasingly complex geometries as a result of moving boundaries are also well handled. In addition, there is no need to directly define the governing process, only the rate constants of the individual processes. The governing process can be revealed through small changes in these coefficients. The different processes; moving boundaries, solvent penetration, API dissolution and solute diffusion are modular and easy to replace with new governing equations. For example, the solute diffusion equation can be changed to represent Case II diffusion, or the API dissolution could consider individual particles directly without need for cell based concentrations(other than the need for more detailed model construction). The data output is scalable depending on desired resolution. The output can be as little as a release profile, to detailed contour maps of solvent concentration, API concentration, solute concentration, etc. at every timestep.

Chapter 3

Case Study: The role of erosion patterns on the active substance release profiles

3.1 Introduction

The dissolution performance of pharmaceutical tablets varies from tablet to tablet, even when those tablets are from the same batch. These changes can be the result of differences in active particle distribution, internal structure or particle properties. In order to determine the importance of the roles played by each of the governing physical processes of the model, a series of simulations were performed to compare change in operating coefficients to changes in the release profile. The design space was restricted to an area which produced release profiles similar to those of physical tablets which were previously dissolved. The physical results will not be displayed here as the tablets in question were subject to physical processes not currently incorporated in this model.

3.2 Erosion Patterns

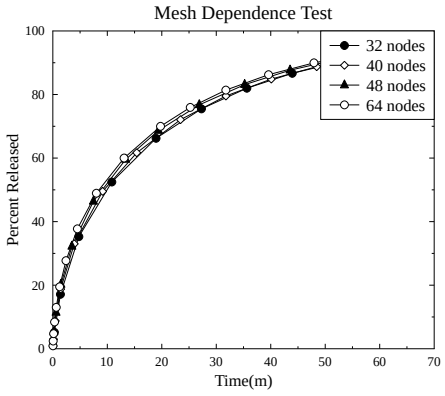
A moving tablet fluid interface is necessary to capture the effects of changing geometry on tablet dissolution. The model focuses on smooth interface tracking over a standardized grid structure in order to allow for additional complexities to be added to the model framework. It has been shown that evolving geometries have an effect on the rate of drug release and thus overall tablet performance [23, 13] In order to prove the feasibility and effectiveness of our method, four different types of erosion controlling profiles have been performed using the model. These different erosion patterns result from the specific evolution law provided for the normal component of the surface velocity v_n in Eq. 2.7 of the dissolving solid. The four cases are: homogeneous erosion,

heterogeneous erosion due to inhomogeneities in density distributions, heterogeneous erosion due to inhomogeneities fluid flow profiles, and heterogeneous erosion due to variations in both density and flow. Finally, a case of coupled surface erosion and bulk diffusion is presented.

To ensure that a proper spatial representation is utilized, a mesh convergence test was performed. The results for this analysis is shown in Figure 3.1 for a solid without erosion. As expected, the simulated release profiles are smoother for more resolved meshes, however the overall behavior shows a converged response for all meshes studied: 32^3, 40^3, 48^3 and 64^3. A resolution of 40^3 was deemed sufficient for all subsequent simulations.

	NE 32^3	NE 40^3	NE 48^3	NE 64^3
API Disso Coeff(mm^-2/s)	2E-07	2E-07	2E-07	2E-07
Pen. Coeff(mm^2/s)	1E-09	1E-09	1E-09	1E-09
Erosion factor (mm/s)	1E+08	1E+08	1E+08	1E+08
Erosion Type	Uniform	Uniform	Uniform	Uniform
Solute Diff. Coeff(mm^2/s)	1.00E-06	1.00E-06	1.00E-06	1.00E-06
Tablet Radius(mm)	5	5	5	5
Tablet Thick.(mm)	5	5	5	5
Grid Nodes	32x32x32	40x40x40	48x48x48	64x64x64
Density Dist. Amp	0	0	0	0

(a) Mesh Test Parameters



(b) Mesh Test Release Profiles

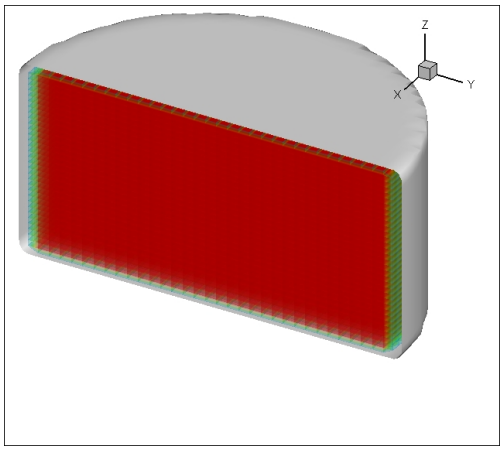
Figure 3.1: Mesh convergence test

3.2.1 Homogeneous Erosion

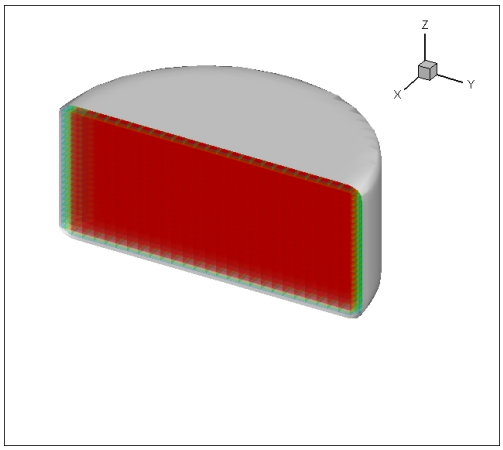
The first erosion case explored is simple uniform erosion of the tablet surface, where v_n is equal to a constant $v_0 = 1\frac{\mathrm{m}}{\mathrm{s}}$. This scenario corresponds to the erosion of a completely homogeneous and isotropic solid in a motionless fluid acting as a perfect sink. The boundary conditions at the tablet fluid interface are set slightly differently for each of the processes. In the case of the solvent penetration equations, everything outside the tablet surface is considered to be pure solvent at all times, and everything inside the tablet is considered to be perfectly dry initially. For the solute diffusion equations, everywhere outside the tablet is initially pure solvent, but as time progresses the formation of a boundary layer is considered. The boundary layer thickness is determined based on the current average concentration of solute at the tablet surface. This is discussed in more detail in the previous section on Solute Diffusion. These conditions represent at $t = 0$ a completely dry solid submerged into a fluid medium. These boundary and initial conditions are utilized for all subsequent cases. The erosion evolution, shown in Fig. 3.2, exhibits a uniform shrinkage over time and the shape remains self-similar. It should be noted that no numerical artifact effects are introduced on the evolving tablet shape even when the level set was never reinitialized.

3.2.2 Heterogeneous Erosion Driven by Solid Properties: Density Distribution

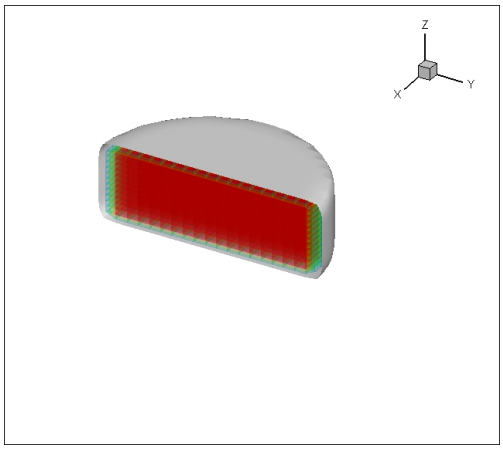
We computationally examine the effect of non-homogenous density distribution on the active release profiles. The distribution has been shown to alter the drug release performance of compacted tablets [5]. In this study, we limit our analysis to density distributions with radial symmetry which is one of the most prevalent cases due to the effects of friction during compression. These profiles are exemplified in the 3D density mappings shown in Fig. 3.3 which were obtained by X-ray μ-computer tomography. Other studies have shown similar heterogeneous density distributions in pharmaceutical tablets[24, 25]. It should be noted that the proposed methodology allows for incorporating more general density profiles with similar level of complexity.



(a) Uniform Erosion Beginning



(b) Uniform Erosion Intermediate



(c) Uniform Erosion Final

Figure 3.2: The progression of tablet geometries show the evolution of the tablet-fluid interface due to a uniform erosion profile. The shape remains completely the same as the tablet erodes. There is no solvent penetration or API dissolution taking place. This would represent a completely isotropic tablet floating in solvent with no fluid motion.

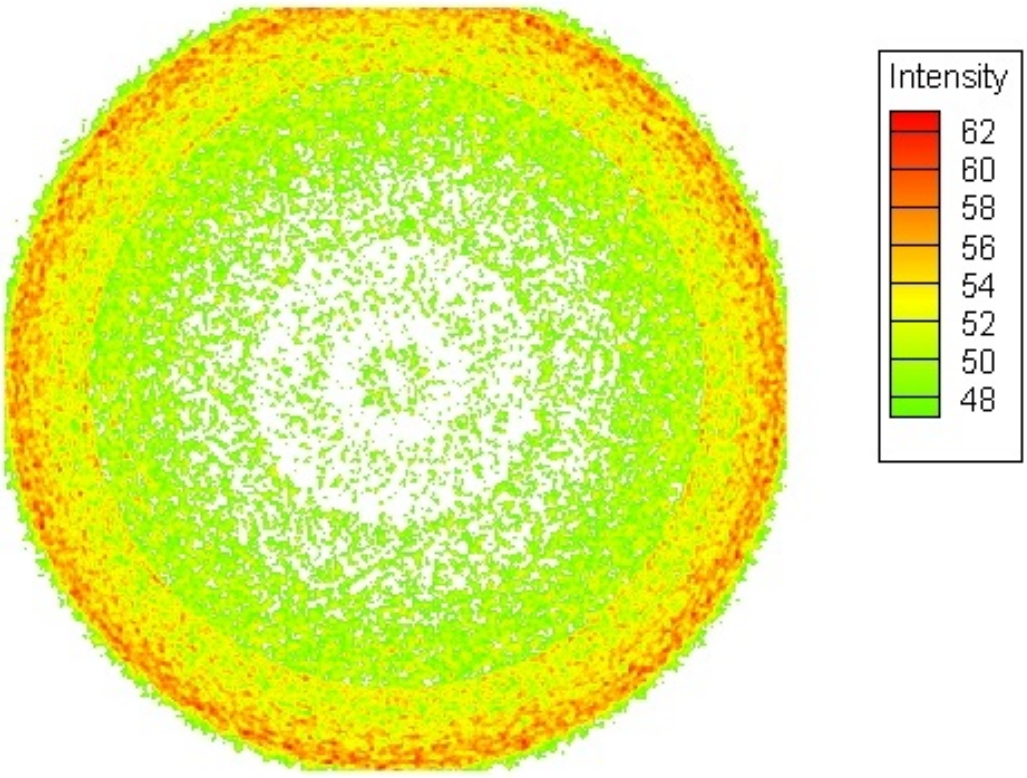


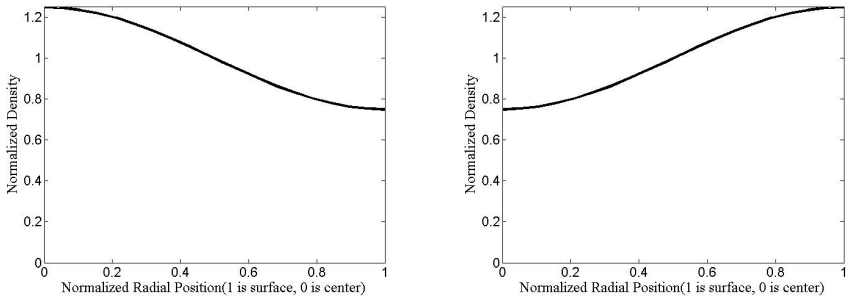
Figure 3.3: Density Profile of tablet obtained from X-ray μCT. Intensity represents relative density. White areas have values below the cutoff threshold. An exaggerated form of this density profile is investigated in the light core case.

The erosion profiles are considered to be proportional to the tablet density which are described by equation for the normal velocity at the interface:

$$v_n^{\text{density}}(X,Y) = K_{\text{density}} \frac{v_0}{\left(1 + A\sqrt{\frac{X^2+Y^2}{R^2}}\right)}$$

where v_0 is the erosion rate for the uniform case and A is constants selected to account for the variation of the erosion rate due to density distributions. Values of $A < 0$ corresponds to light cores and $A > 0$ to dense cores. Two different cases have been considered, one with $A = -0.5$ another with $A = 0.5$. The erosion profiles for each case are shown in Fig. 3.4. Finally, K_{density} is a constant selected in such a way that the initial dissolution rate of the active substance, denoted as ξ, is the same to the homogeneous case, i.e.

$\text{K}_{\text{density}}$ is selected such that $\xi^{\text{homogeneous}}|_{t=0} = \xi^{\text{heterogeneous}}|_{t=0}$
which gives $K_{\text{density light core}} = 1.1984$ for light core case ($A < 0$) and $K_{\text{density dense core}} = 0.6585$ for dense core case ($A > 0$). Different erosion profiles are also derived from heterogeneities in composition, for example, experiments with HPMC tablets have shown that higher API concentrations result in faster erosion of the tablet matrix, more so in the case of low solubility API [26, 27]. This type of variability is also amenable within the current modeling framework.



(a) Dense Core (b) Light Core

Figure 3.4: Density Distributions for Dense Core and Light Core Cases.

Light Core Case

In this case, the erosion produces a smooth tablet surface which dips in the center of the top and bottom of the tablet as expected. As time progresses this dip becomes more pronounced, and eventually would lead to a hole forming in the center of the tablet. Notice that the numerical formulation can follow the evolving topology of solid without the need of any ad-hoc schemes which requires any a priori assumptions about the erosion pattern.

Dense Core Case

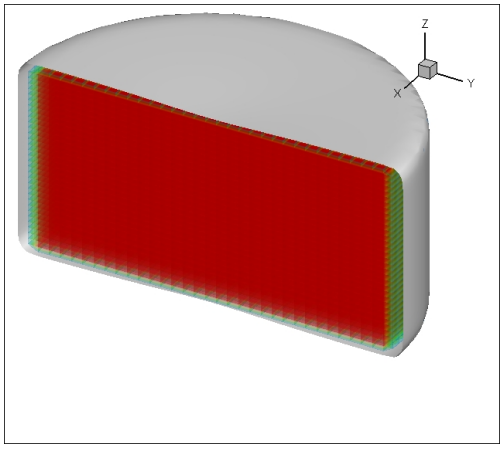
Alternatively, the tablet might have a density distribution with a dense core. The tablet might be more dense in the center than at the radial surface. The resulting tablet erosion from the density profile in Fig. 3.4 produces a smooth tablet surface which becomes increasingly spherical over time as shown in Fig. 3.6 due to the preferential erosion of the outer surface.

3.2.3 Heterogeneous Erosion Driven by Fluid Conditions: Non-uniform Flow

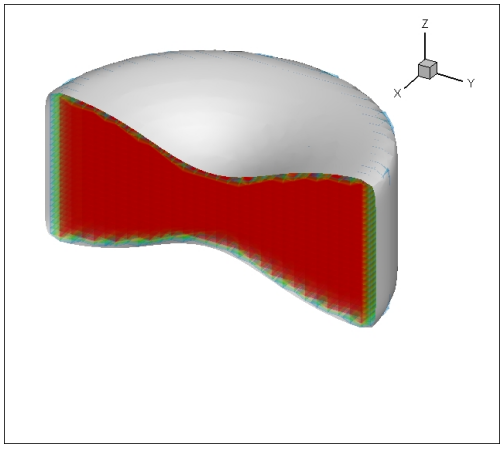
The third case corresponds to the erosion due to fluid flow where we assumed that the erosion is proportional to the tangential component of the flow velocity. Notice that the surface is evolving in time, so does the tangential direction of the tablet surface. In fully coupled fluid-solid simulations, the evolving shape of the tablet changes the flow patterns. In the present study, we assumed that the fluid flow remains unaltered by the evolving tablet shape. In particular, we consider an approximation for the flow velocity in a 2D channel as shown in Fig. 3.7, which results in the following expression for the erosion rate:

$$v_n^{\text{flow}}(X, Y, Z) = K_{\text{flow}} \left[a \; sin\theta + b \; sin(c \; \theta) + v_0 \right],$$

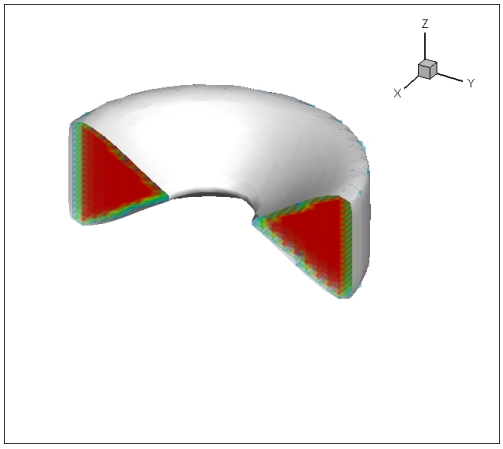
where θ is the angle between the evolving surface normal $\mathbf{n}$ which is computed by evaluating the gradient of the level set ϕ at the interface ($\phi = 0$) properly normalized



(a) Density Based Erosion Beginning

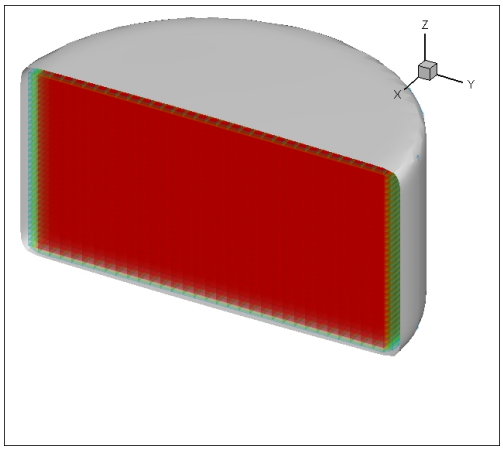


(b) Density Based Erosion Intermediate

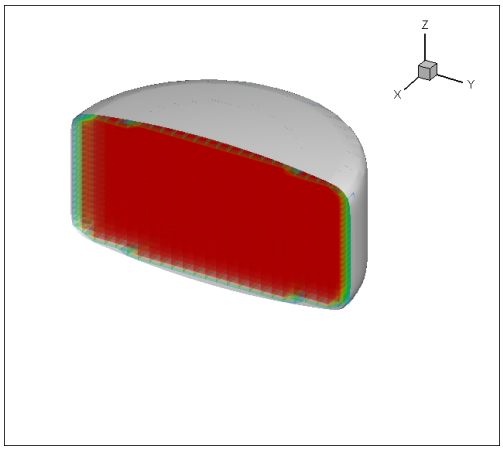


(c) Density Based Erosion Final

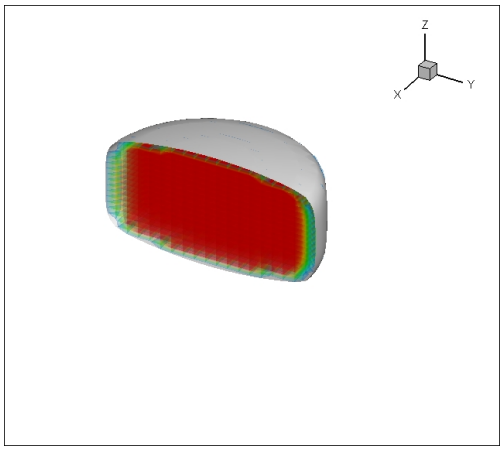
Figure 3.5: Evolved tablet surface geometries accounting for density profile which decreases wrt. radial location. The tablet forms a hole in the center due to its lower central density. The formation of this new surface is handled implicitly via the level set.



(a) Density Based Erosion Beginning



(b) Density Based Erosion Intermediate



(c) Density Based Erosion Final

Figure 3.6: Evolved tablet surface geometries accounting for density profile which decreases wrt. radial location. Tablet becomes increasingly spherical over time.

and the z-axis, i.e.

$$\theta(X, Y, Z) = \arccos\left[\mathbf{n}(X, Y, Z) \cdot \mathbf{k}\right].$$

The constants a,b and c are set to 0.5, 0.25, and 2 respectively. The resulting shear erosion profile is shown in Fig.3.8. Similar to the density case, the constant K_{fluid} is selected so the dissolution rate of the active substance is the same to the homogeneous case, i.e.

K_{flow} is selected such that $\xi^{\text{homogeneous}}|_{t=0} = \xi^{\text{heterogeneous}}|_{t=0}$,
which gives $K_{\text{flow}} = .6654$

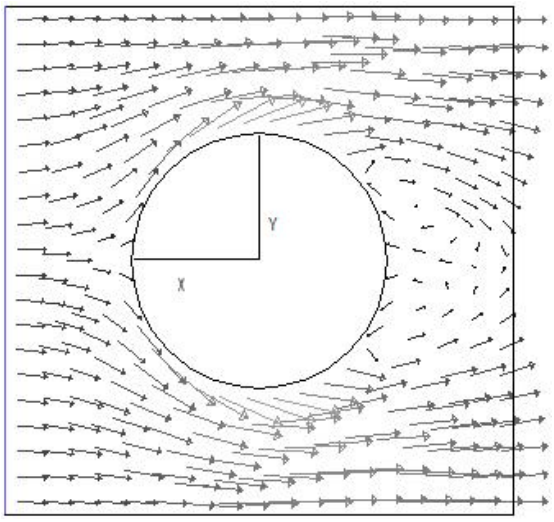


Figure 3.7: simulated shear profile around cylindrical solid

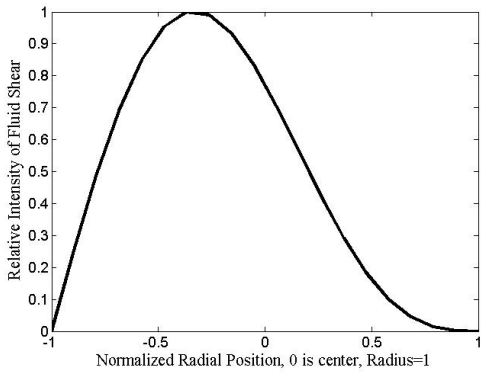


Figure 3.8: Imposed erosion rate as a function of θ.

The simulations capture the directionality imposed by the channel flow conditions (in addition to the uniform erosion), showing a preferential erosion in regions of higher tangential velocity. The initial cylindrical shape evolves into an airfoil or a teardrop one as shown in 3.9. Other flow patterns can also be considered such as the one produced by the USP type II apparatus. However, it has been shown [28, 29] that the flow profile acting on the surface of the solid is highly sensitive on positioning of the solid, in particular on misalignments of the axis of the tablet with the axis of rotation.

3.2.4 Heterogeneous Erosion Driven by Density and Flow

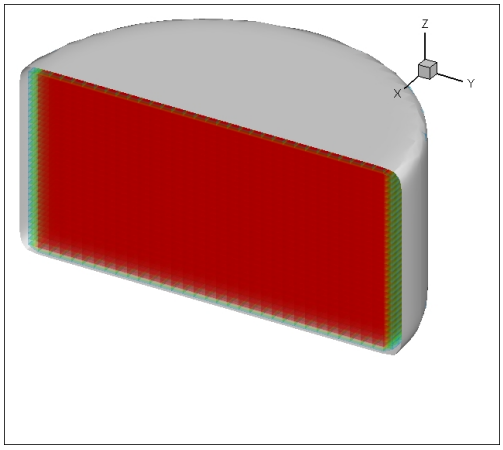
A more complex scenario is the drug release profiles of heterogeneous solid in a heterogeneous environments, where erosion is dictated by both solid and fluid properties and conditions. Two different conditions have been considered with the same imposed flow, light core and dense core. In the first case, shown in 3.10, the solid surface becomes less symmetric, owing to the combined effects of shear based erosion and tablet density. As the front portion of the solid has a larger levels of tangential velocities and tablet density decreases with the radial location, the front portion of the solid shows a noticeable shape variation from the rear end. In the second case, shown in 3.11, there is also noticeable change in shape where the tendency towards sphericity due density effects dominates the surface topology. This modeling strategy can be expanded to include other modalities of interfacial movement such as additional shear conditions(ie. solid/solid interactions) or internal swelling. In this case, the erosion rate is controlled by:

$$v_n^{\text{density+flow}}(X,Y,Z) = K_{\text{density+flow}} \left\{ \left(A\sqrt{\frac{X^2+Y^2}{R^2}} \right) + [a\ sin\theta + b\ sin(c\ \theta)] + v_0 \right\},$$

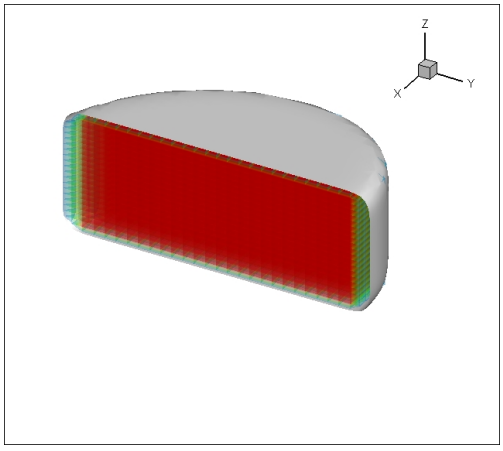
where again $K_{\text{density+flow}}$ has been modulated to enforced that

$$\xi^{\text{homogeneous}}|_{t=0} = \xi^{\text{heterogeneous}}|_{t=0},$$

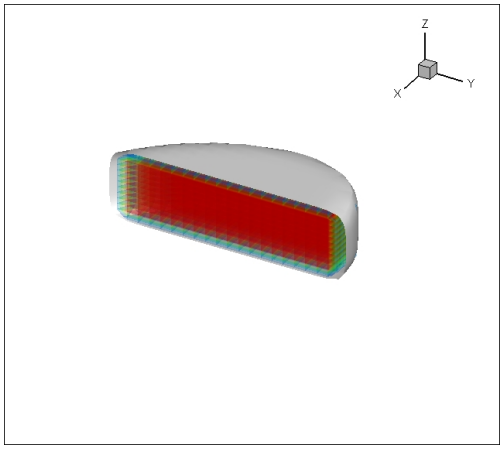
which give $K_{\text{flow+density light core}} = .6898$ and $K_{\text{flow+density dense core}} = .3978$.



(a) Shear Based Erosion Beginning

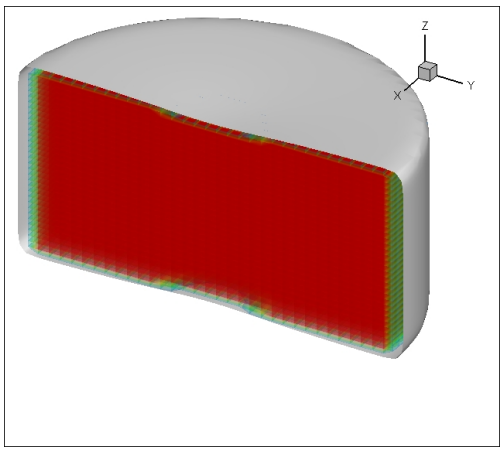


(b) Shear Based Erosion Intermediate

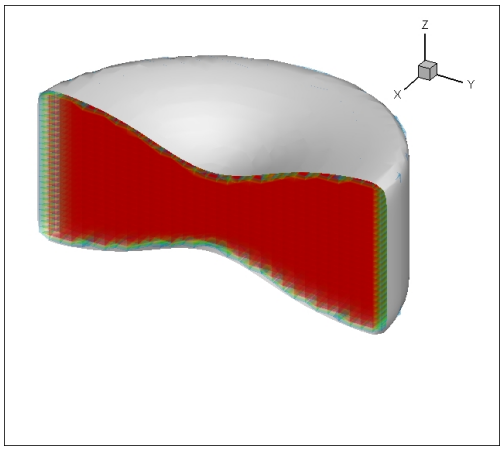


(c) Shear Based Erosion Final

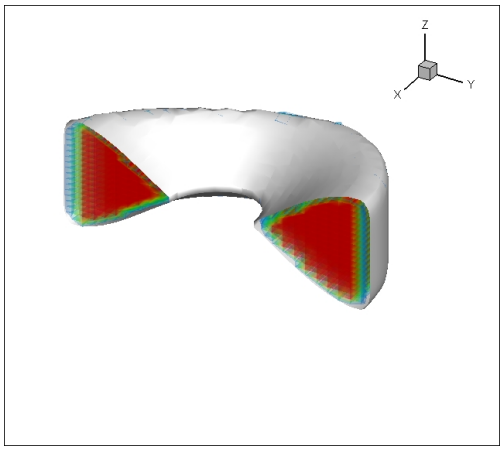
Figure 3.9: Initial and intermediate tablet surface geometries for implied shear based erosion. There are no dissolution processes occurring besides surface erosion. This case represents an augmented version of the uniform erosion case.



(a) Shear and Density Based Erosion Beginning

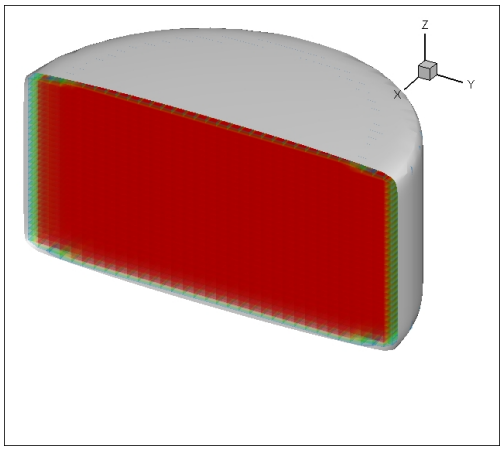


(b) Shear and Density Based Erosion Intermediate

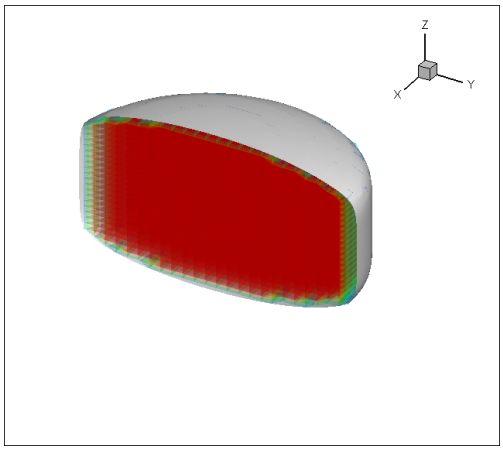


(c) Shear and Density Based Erosion Final

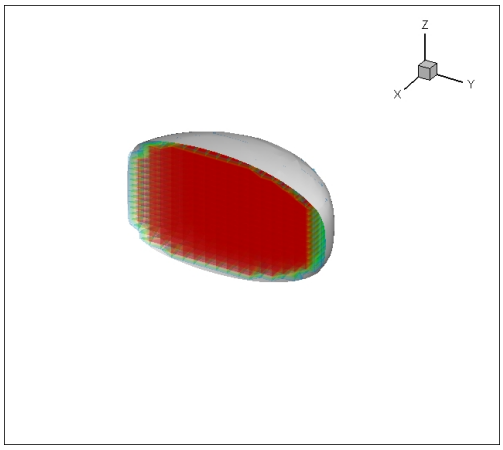
Figure 3.10: Initial and intermediate tablet surface geometries for implied shear based erosion combined with the imposed density profile for tablets with denser edges than cores.



(a) Shear and Density Based Erosion Beginning



(b) Shear and Density Based Erosion Intermediate



(c) Shear and Density Based Erosion Final

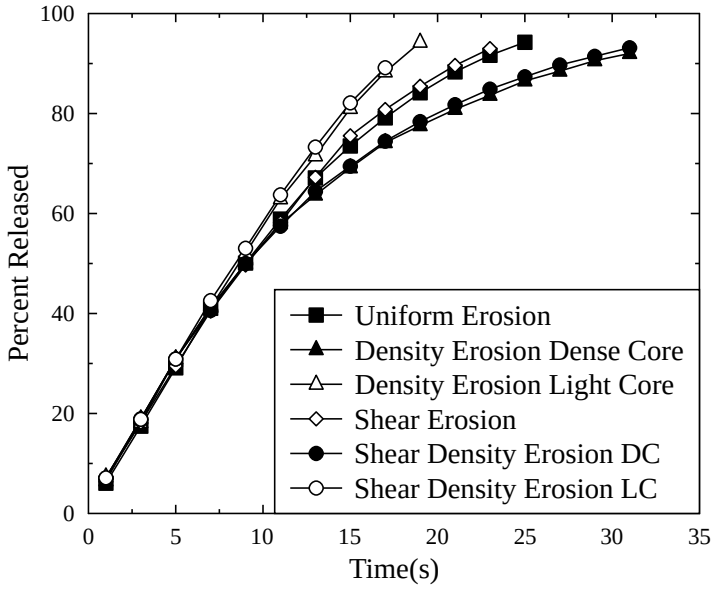
Figure 3.11: Initial and intermediate tablet surface geometries for implied shear based erosion combined with the imposed density profile for tablets with denser cores than edges.

Effect of Erosion Mechanisms on Release Profiles

In order to study the imprint of the erosion mechanisms on the release profile of the active substances, we compare the homogeneous release profile to all heterogeneous cases studied in the previous sections. In order to compare the characteristic features of the release profiles, the initial rate for all heterogeneous cases is set to be equal to the homogeneous case. These profiles are presented in Figure3.12 , where the effects are clearly observed.

	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6
API Disso Coeff(mm^-2/s)	2.00E-07	2.00E-07	2.00E-07	2.00E-07	2.00E-07	2.00E-07
Pen. Coeff(mm^2/s)	1.00E-09	1.00E-09	1.00E-09	1.00E-09	1.00E-09	1.00E-09
Erosion factor (mm/s)	1.30E-03	1.30E-03	1.30E-03	1.30E-03	1.30E-03	1.30E-03
Erosion Type	Uniform	Density	Density	Flow	Flow and Density	Flow and Density
Solute Diff. Coeff(mm^2/s)	1.00E-06	1.00E-06	1.00E-06	1.00E-06	1.00E-06	1.00E-06
Tablet Radius(mm)	5	5	5	5	5	5
Tablet Thick.(mm)	5	5	5	5	5	5
Grid Nodes	40x40x40	40x40x40	40x40x40	40x40x40	40x40x40	40x40x40
Density Dist. Amp	0	0.5	-0.5	0	0.5	-0.5

(a) Model parameters



(b) Model Release Profiles

Figure 3.12: Release Profiles of model tablets with different erosion conditions

To quantify these differences a similarity factor f_2 is used, which is defined as [30]:

$$\kappa = 50 \log \left\{ 100 \left[1 + \frac{1}{n} \sum_{t=1}^{n} (R_t - T_t)^2 \right]^{-\frac{1}{2}} \right\},$$

where n is the number of points used R_t is the simulated percent active released at time t of the reference (homogeneous) case and T_t is the percent active released at time t of the test (heterogeneous) case. If the value of f_2 is 100, the profiles are identical. The values computed from all cases at $t \to 0$, $t = 9$ and 17 minutes (when about 80% release is observed for all cases) are presented in Table 3.1. It is common to cap the release percentage when using this comparison technique so as to reduce the effects of a trailing difference as the tablets have neared the end of their release.

Table 3.1: Similarity factors for erosion controlled release

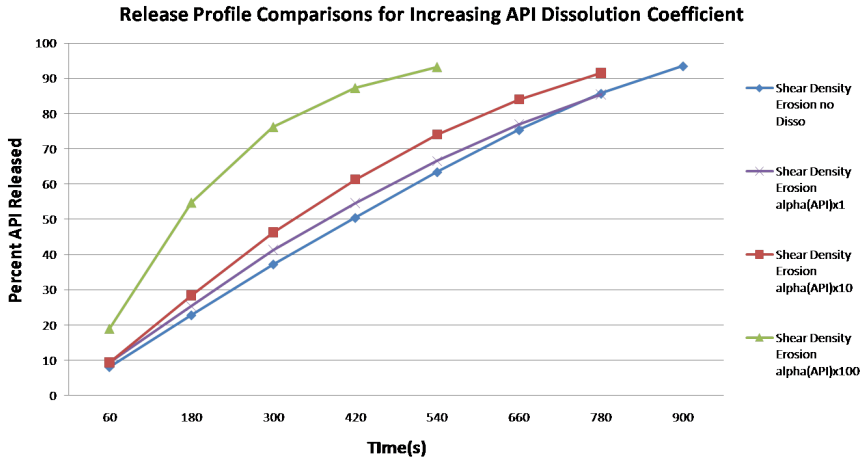
Case	$f_2 \to 0$	f_2 at 9 minutes	f_2 at 17 minutes
Density with dense core	100	78	92
Density with light core	100	66	89
Flow	100	92	96
Flow + Density with dense core	100	78	90
Flow + Density with light core	100	63	84

It is interesting to notice that the larger difference in active release profiles are found for the set of parameter of these study when the density of the core is relatively high which shows a closer shape evolution to the reference homogeneous case. It can be noted that the lowest values in the table for 17 minutes are for comparisons of the tablets with two different density profiles. The difference between dense exterior and dense interior and even between both and the uniform erosion case is attributable to the different ways in which the surface evolves. The case with denser exteriors have a divot form in the middle of the tablet as time increases. This divot allows for faster solvent penetration and solute diffusion from the tablet as it increases the surface area to volume ratio as the tablet dissolves. This effect clearly produces greater differences than a change in the flow profile around the tablet, as long as the overall erosion rates are similar. The tablets with denser cores are the slowest to dissolve. Part of this is

due to the normalization of the erosion rates, but also because these tablets have the smallest surface area to volume ratio of any of the groups as they tend towards a more spherical shape as time progresses.

3.3 Coupling Active Dissolution to Surface Erosion Modes

As the simulation framework allows to consider the interaction of several concurrent processes, we finally consider the interaction of the surface erosion modes with the rate of dissolution of the active which depends on the current value of the solvent concentration(increasing as time progresses), the solubility of the active(defined in advance) and the particle size distribution(defined in advance). Four specific cases have been considered in which the solvent penetration rates and API dissolution rates were manipulated via the main governing coefficients. The cases range from no dissolution, to fast dissolution. The cases which include internal dissolution have API dissolution coefficients which are each a factor of 10 greater than the previous slower case. In the previous sections, the model parameters, with the exception of erosion type and density distribution amp, are those of the Slow Dissolution case. The difference between this case and the case where all dissolution is the effect of erosion is very small, as shown in Fig 3.13. This is because the surface erosion is still the rate governing process for these parameter values. As the API dissolution coefficient is increased, it plays a larger and larger role in the drug release rate. The shape of the release profile for the Fast Dissolution case is much more curved than that of the No Dissolution case. Since the API particle dissolution rate is handled by calculating spherical diffusion, as the particles dissolve, their surface area decreases, and their individual dissolution rates decrease. That is why the release rate of the Fast Dissolution cases tailors off after the inital fast release.



(a) Release Increase Dissolution Coefficient

	No Disso	Slow Disso	Medium Disso	Fast Disso
API Disso Coeff(mm^-2/s)	0	2.00E-07	2.00E-06	2.00E-05
Pen. Coeff(mm^2/s)	1.00E-09	1.00E-09	1.00E-09	1.00E-09
Erosion factor (mm/s)	1.30E-03	1.30E-03	1.30E-03	1.30E-03
Solute Diff. Coeff(mm^2/s)	1.00E-06	1.00E-06	1.00E-06	1.00E-06
Tablet Radius(mm)	5	5	5	5
Tablet Thick.(mm)	5	5	5	5
Grid Nodes	40x40x40	40x40x40	40x40x40	40x40x40
Density Dist. Amp	0.5	0.5	0.5	0.5

(b) Model Parameters

Figure 3.13: Release Profiles of model tablets with different API for tablets whose surface erosion is influenced by fluid shear and density distribution, dense core.

Chapter 4

Conclusions and Future Work

The model shows at least a core competency in effectively modeling basic release processes. The movement of the surface is properly handled, and the numerical portion is handled in such a way as to allow for full customization of the surrounding fluid shear environment. This is evident in the multiple geometry changes handled by the model, and the resulting smoothness of the release profiles from these various models. The changes to the location of the tablet fluid interface had the expected effects for the simple cases considered. There were clear differences in the resultant release from tablets with different density distributions. At the same time, the effects of a differently shaped fluid flow profile did not have as great an effect on the system. The model system presented here is meant to show feasibility as a platform for the creation of new models and for studying environmental effects. Before this model is complete more physical processes need to be added and the format needs to be ported to C and converted to take advantage of parallel processing techniques.

4.1 Future Work

Swelling of the excipient matrix has been shown to be an extremely important part of tablet dissolution for certain excipient formulations. Most commonly, the swelling response of HPMC(hydroxy propyl methyl cellulose)[31, 32, 33]. Other excipients, including micro-crystalline cellulose and lactose, can also swell when exposed to solvents. Swelling of the matrix can cause changes in the location of the bulk interface, the rate of solvent uptake, rate of solute diffusion and contribute to internal stresses. We have already begun to perform experiments using video processing to characterize 1-D

swelling and solvent penetration parameters. We also plan to measure the swelled volume of individual particles and use simulations to relate this to swelling of the tablet as a whole.

In addition to measuring the swelling parameters physically, it will be necessary to make adjustments to the model to include swelling. A cellular automata model which captures the swelling response of HPMC has been recently published[34], which demonstrates a 2-D model of HPMC swelling verified with physical experiments. Incorporating a high resolution 3-D cellular automata model inside the existing grid space would allow for swelling and solvent penetration to be simulated with an increased relevance at an increased resolution. These processes are less computationally expensive than updating the level set function or calculating the ghosting coefficients, thus increasing their resolution would have much less of an effect than increasing overall grid resolution. As previously described we are focusing on a meso-scale model which allows for non homogeneous tablet descriptions, and part of the goal is to resolve the processes involved in tablet dissolution at meaningful resolutions. A mechanistic 1-D model which calculates tablet swelling in response to stress relaxation of individually swelled particles as a result of solvent penetration using a level set technique has been tested, but scaling to 3-D would cause the model to run orders of magnitude slower than the present framework.

Once swelling effects have been incorporated, we also plan to allow for more methods of model input. Currently, models are built using a simple program which does not have the power of a DEM based approach. It would be extremely useful to directly incorporate results from models of tablet compaction. As another group member is working directly with such models[35], the data produced by these will be the obvious starting point. Once a model can be built using this data, more common formats such as .stl will be considered, as this greatly broadens the range of programs which can then build the initial model. Part of the drive to add recognition of common formats is due to our participation in the Pharmahub project. We are also looking for new ways to extract parameters from physical tablets using techniques such as X-ray tomography and LIBS(Laser Induced Breakdown Spectroscopy).

.1 Model Builder

The models used in this study were built in groups using this script where the values of each of the user defined parameters are entered. The function handles the recursion of building different models and names files from a base group with a numbered counter. The actual model building is done by the referenced function "build tablet model3" which is described in the next section.

```
clear;


Grid_nodes              = 30;
Tablet_thickness        = 5;
Tablet_radius           = 5;


Tstep                   = 150000;
Partr                   = .0000215;
Conc_Active             = 9;
RReducer=[10000];
Pen_Coeff=[100];
Dissolution_Coefficient=[2000];
Porosity_factor=[0];
active_slope=[0];
DD_solute= [1e-6];
PPen_slope=[0];
Amp=[1]
fudge=1;
erosion_type=4;
pillset=['Uniform_Erosion_w_Internal_Disso_mesh_30'];
pillstyle=[pillset,num2str(fudge)];
final_time=40*90;
for tt=1:length(RReducer)
```

```
    for vv=1:length(Pen_Coeff)
        for qq=1:length(Dissolution_Coefficient)
            for ff=1:length(Porosity_factor)
                for uu=1:length(active_slope)
                    for bb=1:length(DD_solute)
                        for cc=1:length(PPen_slope)
                            for dd=1:length(Amp)
                coords=[tt,vv,qq,ff,uu];
            [D_solute]=build_tablet_model3(RReducer(tt),Pen_Coeff(vv),
            Dissolution_Coefficient(qq),Porosity_factor(ff),active_slope(uu),
            DD_solute(bb),PPen_slope(cc),pillstyle,coords,Grid_nodes,
            Tablet_thickness,Tablet_radius,Tstep,Partr,Conc_Active,final_time,
            erosion_type,Amp(dd));
            fudge=fudge+1;
            pillstyle=[pillset,num2str(fudge)];
                            end
                        end
                    end
                end
            end
        end
    end
end

Amp =

     1



fname =
```

```
Load_Files\Uniform_Erosion_w_Internal_Disso_mesh_301_setup
```

.2 build tablet model3

This function builds the models used in the simulation engine. It takes the user inputs and build all the necessary matrices including the level set solvent concentration, active concentration, solute concentration and other helper matrices.

```
function[D_solute]=build_tablet_model3(reducer,Pen_coefff,Dissolution_coefficient,
porosity_factor,active_slope,D_solute,Pen_slope,pillstyle,coords,grid_nodes,
tablet_thickness,tablet_radius,tstep,partr,Conc_active,final_time,erosion_type,Amp)


interface_factor=.9;
set_step = 1000;
form_factor=(1/.9);
% % active_slope=0;
M=Dissolution_coefficient;
dte=2e-6;
dt=5e-14;
dtmax=1;
warning('off','all');
stepp=1;
tablet_thickness=tablet_thickness*.001;
tablet_radius=tablet_radius*.001;
Conc_active=Conc_active*.01;
release_step=100;
safety=50;
vol_threshold=.1;
solvent_threshold=.3;
percolation_threshold=.5;
```

```
Cw_interface=1;
C_solute_interface=0;
%reducer=reducer*1e10;
Pen_coeff = Pen_coefff*1e-11;
% % porosity_factor=porosity_factor*1e-11;
percent_released=0;
max_release=99.8;
max_ite=10;
D=1;
oops=0;

dtplus=0;
relaxor=0.5;
dte=2e-8;%Fixed timestep
step=1;%initialize step counter

%uses user input values to determine state space parameters
x_length=2*form_factor*tablet_radius;
y_length=2*form_factor*tablet_radius;
z_length=form_factor*tablet_thickness;
xmin =-form_factor*tablet_radius; ymin =-form_factor*tablet_radius; zmin=...
 -.5*form_factor*tablet_thickness;
xmax = form_factor*tablet_radius; ymax = form_factor*tablet_radius; zmax=...
  .5*form_factor*tablet_thickness;

nx=grid_nodes;
ny=grid_nodes;
nz=grid_nodes;
%uses physical limits and defined number of nodes in each direction to
%determine cell dimensions
```

```
dx=(xmax - xmin)/(nx - 1); dy=(ymax - ymin)/(ny - 1);  dz=(zmax - zmin)/(nz - 1);
%dd =min(min(dx,dy),dz);
dd=(dx*dx + dy*dy + dz*dz )^(0.5);
%tb=0.05*tstep*dt;
timenow=0;
shift=1000000;
%defines the percentage of nodes/100 which are actives


%total number of nodes
nump=ny*nx*nz;
%particles=100000;
particle_volume=(4/3)*pi*partr^3;
cell_volume=dx*dy*dz;
Tablet_volume=(tablet_radius^2)*pi*(tablet_thickness);
total_active_volume=Conc_active*Tablet_volume;
num_part=ones(ny,nx,nz).*total_active_volume/particle_volume;

%uses physical limits and defined number of nodes in each direction to
%determine cell dimensions

C_active=zeros(ny,nx,nz);C_BC=zeros(ny,nx,nz);
pathway=C_active;
%Cxp=zeros(ny,nx,nz);Cxm=zeros(ny,nx,nz);Cyp=zeros(ny,nx,nz);
%Cym=zeros(ny,nx,nz);Czp=zeros(ny,nx,nz);Czm=zeros(ny,nx,nz);
%Positional Matrices
x=linspace(xmin,xmax,nx); y=linspace(ymax,ymin,ny); z=linspace(zmin,zmax,ny);
[X,Y,Z] = meshgrid(x,y,z);
%Uno
ONE=ones(nx,ny,nz);
```

```
%Diffusion matrix;  Used in conjuction with Cw matrix to determine
%diffusibility of active through the scaffold
%Dw=Pen_coeff*ONE;

%Concentration of water in scaffold
Cw=zeros(ny,nx,nz);



% %~~~~ INITIALIZE MATRICES  ~~~~~%
Cxp=zeros(ny,nx,nz);Cxm=zeros(ny,nx,nz);Cyp=zeros(ny,nx,nz);Cym=zeros(ny,nx,nz);
Czp=zeros(ny,nx,nz);Czm=zeros(ny,nx,nz);
Lxp=zeros(ny,nx,nz);Lxm=zeros(ny,nx,nz);Lyp=zeros(ny,nx,nz);Lym=zeros(ny,nx,nz);
Lzp=zeros(ny,nx,nz);Lzm=zeros(ny,nx,nz);



%~~~~ INITIALIZE COUNTERS  ~~~~~%
k=1;m=1;t=0;p=1;pp=2;ite=1;
So=zeros(ny,nx,nz);
%~~~~ THE CARTESIAN GRID  ~~~~~%
x=linspace(xmin,xmax,nx); y=linspace(ymax,ymin,ny); z=linspace(zmin,zmax,ny);
[X,Y,Z] = meshgrid(x,y,z);
x2 = linspace(2*xmin,2*xmax,nx);
y2 = linspace(2*ymin,2*ymax,ny);
z2 = linspace(2*zmin,2*zmax,nz);
[X2,Y2,Z2] = meshgrid(x2,y2,z2);
ZZ = zeros(nx,ny,nz);
ONE=ones(nx,ny,nz);
EPS =eps*ones(nx,ny,nz);
k=1;
```

```
%~~~~ LOCATION AND SIZE OF INTIAL PILLS  ~~~~~%
R=sqrt(X.^2+Y.^2);
%z=p(:,3);
Rs=sqrt(X.^2+Y.^2+Z.^2);
d1=R-.9*xmax;
d2=Z-.9*zmax;
d3=-Z+.9*zmin;

d=dintersect(dintersect(d1,d2),d3);

L=d;
%~~~~ INTIALIZE GRAPHICS OUTPUT  ~~~~~%

Lxp(:,1:nx-1,:) = L(:,2:nx,:);
Lxp(:,nx,:) = 2*L(:,nx,:)-L(:,nx-1,:);
Lxm(:,2:nx,:) = L(:,1:nx-1,:);
Lxm(:,1,:) = 2*L(:,1,:)-L(:,2,:);

Lyp(2:ny,:,:) = L(1:ny-1,:,:);
Lyp(1,:,:) = 2*L(1,:,:)-L(2,:,:);
Lym(1:ny-1,:,:) = L(2:ny,:,:);
Lym(ny,:,:) = 2*L(ny,:,:)-L(ny-1,:,:);

Lzp(:,:,2:nz) = L(:,:,1:nz-1);
Lzp(:,:,1) = 2*L(:,:,1)-L(:,:,2);
Lzm(:,:,1:nz-1) = L(:,:,2:nz);
Lzm(:,:,nz) = 2*L(:,:,nz)-L(:,:,nz-1);
```

```
GradLx = (Lxp - Lxm)/(2*dx);
GradLy = (Lyp - Lym)/(2*dy);
GradLz = (Lzp - Lzm)/(2*dz);
Norm = (GradLx.^2 + GradLy.^2 + GradLz.^2).^(1/2) + EPS;

P1=GradLx./Norm;
P2=GradLy./Norm;
P3=-GradLz./Norm;



OUT_level = (max(L,ZZ)./L);
IN_level  = (min(L,ZZ)./L);
L_min     = min(min(min(L)));

[index]=index_irregular3(L,Lxp,Lxm,Lyp,Lym,Lzp,Lzm,ny,nx,nz);

active_level=max((IN_level-abs(index)),ZZ);
[C_active,C_cell]=active(active_level,active_slope,nx,ny,nz,index,ZZ,L,...
total_active_volume,cell_volume);
Dw=(OUT_level+(Pen_coeff.*IN_level))+IN_level.*((L-L_min/2)./(L_min/2))...
.*(Pen_slope/100).*Pen_coeff;
SV=C_active.*cell_volume;
total_active_volume=sum(sum(sum(C_active*cell_volume)));
C_solute=zeros(ny,nx,nz);
fname=['Load_Files\',pillstyle,'_setup']
%Tablet_conc=sum(sum(sum(C_active)));
save (fname);

Input argument "Dissolution_coefficient" is undefined.


Error in ==> build_tablet_model4 at 8
```

```
M=Dissolution_coefficient;
```

.3 Tablet Dissolution

This function is used to perform batches of simulations, enacting the main engine for each of the models in a defined series.

```
pillstyle = ;
start_tab = ;
tot_tab = ;

for pp=start_tab:tot_tab
    newTablet=[pillstyle,num2str(pp)];
        [release_profile,Run_name,flood_profile,C_active,TimeTaken,C_solute,L]=...
        Cylinder_dissolve_3D(pillstyle,pillstyle, grid_nodes, tablet_thickness,...
        tablet_radius, reducer, tstep, partr, Conc_active, M, Pen_coeff, D_solute,...
        erosion_type,Amp );

end

quit;
```

.4 group dissolve33

This is the main function of the simulation engine. This portion of code loads the previously constructed models and simulates their dissolution. Once the models are loaded and several helper matrices constructed, the first step of the simulation process is completed. This includes an update of the level set, the ghosting values, solvent penetration, API dissolution, solute diffusion. Once all the values have been updated to the first step, a repeat loop is entered for the remaining simulation. This loop includes all the previous operations to some degree, ie. some are calculated more often than others depending on the calculation of error times. there are also progress checks which

save the workspace every so many steps or time and which can end the simulation if certain criteria have been met. The helper functions used here will be described following this section in order of appearance.

```
function[release_profile,Run_name,flood_profile,C_active,TimeTaken,C_solute]=...
group_dissolve33(pillstyle)


tic
Run_name=pillstyle; %#ok<NASGU>
loadfilename=['Load_files/',pillstyle,'_setup.mat'];
%filename of active matrix to be loaded
load(loadfilename)
maxN=5;
max_release=94;
movie_counter=1
%-----------------Initialize constants------------
C_active1=C_active; %#ok<NODEF>
Outer_layer=abs(index); %#ok<NODEF,NASGU>
max_conc=max(max(max(C_active)));
% Dw=Pen_coeff*ONE;
Run_name=pillstyle;
SV_original=SV; %#ok<NODEF>
total_active_conc=sum(sum(sum(C_active)));
%percolation_prob=zeros(ny,nx,nz);
%-----------------Initialize Matrices------------
index2=zeros(ny,nx,nz);
Ng=zeros(ny,nx,nz);Nb=zeros(ny,nx,nz);Nc=zeros(ny,nx,nz);Nd=zeros(ny,nx,nz);
pointg=zeros(ny,nx,nz,3);pointb=zeros(ny,nx,nz,3);pointc=zeros(ny,nx,nz,3);
pointd=zeros(ny,nx,nz,3);
release_profile=zeros((tstep/release_step),2);
release_profile2=zeros((tstep/release_step),2);
```

```
flood_profile=zeros((tstep/release_step),2);
flood_profile2=zeros((tstep/release_step),2);
Tablet_Volume=zeros(tstep/release_step,3);
Tablet_Volume2=zeros(tstep/release_step,3);
%~~~~ INTIALIZE GRAPHICS OUTPUT  ~~~~~%
stepp=1;
release_time=90;
release_time2=60;
time_step=1;
time_step2=1;
now_time=0;
newtime=0;
curve_shift=1.5;
%final_time=91*60;



%~~~~ INDEXING FOR BOUNDARY VARIABLES ~~~~~%
%id(i)=0 for interior nodes
%id(i)=1 for boundary nodes
Cid=zeros(ny,nx,nz);
Cid(:,:,nz)=1;   % XY+(FRONT)
Cid(:,:,1)=1;    % XY-(BACK)

Cid(1,:,:)=1;    % XZ+(TOP)
Cid(ny,:,:)=1;   % XZ-(BOTTOM)

Cid(:,nx,:)=1;   % YZ+(RIGHT)
Cid(:,1,:)=1;    % YZ-(LEFT)

Lxp(:,1:nx-1,:) = L(:,2:nx,:);
```

```
Lxp(:,nx,:) = 2*L(:,nx,:)-L(:,nx-1,:);
Lxm(:,2:nx,:) = L(:,1:nx-1,:);
Lxm(:,1,:) = 2*L(:,1,:)-L(:,2,:);

Lyp(2:ny,:,:) = L(1:ny-1,:,:);
Lyp(1,:,:) = 2*L(1,:,:)-L(2,:,:);
Lym(1:ny-1,:,:) = L(2:ny,:,:);
Lym(ny,:,:) = 2*L(ny,:,:)-L(ny-1,:,:);

Lzm(:,:,2:nz) = L(:,:,1:nz-1);
Lzm(:,:,1) = 2*L(:,:,1)-L(:,:,2);
Lzp(:,:,1:nz-1) = L(:,:,2:nz);
Lzp(:,:,nz) = 2*L(:,:,nz)-L(:,:,nz-1);

GradLx = (Lxp - Lxm)/(2*dx);
GradLy = (Lyp - Lym)/(2*dy);
GradLz = (Lzp - Lzm)/(2*dz);
Norm = (GradLx.^2 + GradLy.^2 + GradLz.^2).^(1/2) + EPS;

P1=-GradLx./Norm;
P2=GradLy./Norm;
P3=-GradLz./Norm;

OUT_level = (max(L,ZZ)./L);
IN_level  = (min(L,ZZ)./L);

L_max=max(max(max(max(max(Lxp,Lxm),Lyp),Lym),Lzp),Lzm);
L_min=min(min(min(min(min(Lxp,Lxm),Lyp),Lym),Lzp),Lzm);

[CC]=Curvature3D(P1,P2,P3,nx,ny,nz,dx,dy,dz);
```

```
[CC_fac]=Curv_factor(CC,curve_shift,index);

%~~~~ SOLUTION OF PARABOLIC EQUATION  ~~~~~%
Cw=OUT_level;
initial_in=sum(sum(sum(IN_level)));
initial_out=sum(sum(sum(OUT_level)));



%~~~~ LEVEL SET UPDATE UNDER VELOCITY FIELD ~~~~~%

%%F represents the rate at which the surface will erode or swell.
F=-ONE/reducer;
[F2]=changeFtype(F,X,Y,tablet_radius,P1,P2,P3,nx,ny,nz,erosion_type,CC_fac,ONE,Amp);
normalizer=sum(sum(sum(F)))/sum(sum(sum(F2)));
F2=F2.*normalizer;

if  ( rem(stepp,1)==0)
    [critical,timenow,dL,L,k,IN_level,OUT_level,dt,C_active]=Level_update_3D(dd,F2,...
    timenow,dt,L,Lxm,Lym,Lzm,Lxp,Lyp,Lzp,ZZ,ONE,k,dx,dy,dz,safety,C_active,index);
end
OUT_level = (max(L,ZZ)./L);
IN_level  = (min(L,ZZ)./L);
%~~~~ INDEX FOR REGULAR/IRREGULAR ~~~~~%

%index=1  ->Irregular Inside
%index=-1 ->Irregular Outside
%index=0  ->Regular
[index]=index_irregular3(L,Lxp,Lxm,Lyp,Lym,Lzp,Lzm,ny,nx,nz);
[Ng,Nb,Nc,Nd,pointg,pointb,pointc,pointd,index2]=boundary_ghost3D_coord(index,L,x,y,z,..
P1,P2,P3,maxN,nx,ny,nz,index2,Ng,Nb,Nc,Nd,pointg,pointb,pointc,pointd,dx,dy,dz);
```

```
%%[X1,X5,Y1,Y5,Z1,Z5,D1,D2,D3,Xp,Yp,Zp,Iym,Ixm,Izm,Iyp,Ixp,Izp,DB]=bilinear_coeff_3D...
(dd,P1,P2,P3,X,Y,Z,nx,ny,nz,index,dx,dy,dz,L,x,y,z);

[C_interface]=determine_interface(C_solute,index,interface_factor,ZZ);
% [Cw,oops]=boundary_ghost2(nz,ny,nx,index,Cw_interface,Cw,L,Cw_bulk_fluid,oops);
[Cw,C_solute]=boundary_ghost3D(nz,ny,nx,index,index2,Cw_interface,Cw,C_interface,...
C_solute,Ng,Nb,Nc,Nd,pointg,pointb,pointc,pointd);
%[C_solute,oops]=boundary_ghost3a(nz,ny,nx,index,C_interface,C_solute,L,x,y,z,P1,...
P2,P3,dx,dy,dz,oops,C_bulk_fluid);

%Steps ahead water level through scaffold
[Cw,dtcw,dCw]=water_level2(Cw,Dw,dd,dx,dy,dz,nx,ny,nz,dte,IN_level,OUT_level,dt,ONE);
Cw=max(Cw,OUT_level);
Cw=min(ONE,Cw);

dt;
[Surf]=Surf_calc(C_active,cell_volume,particle_volume);
%[dt_conc,C_active,dC_active]=Active_conc(C_active,Cw,dd,dx,dy,dz,nx,ny,nz,dte,...
IN_level,OUT_level,dt,ONE,ZZ,M,max_conc,Surf);
[C_solute,C_active,dC_active,Dw]=Active_conc4(C_solute,C_active,Cw,IN_level,dt,...
ONE,ZZ,M,max_conc,Surf,Pen_coeff,porosity_factor,Conc_active,Dw);
[dt_solute,C_solute,dc_solute]=Solute_conc(C_solute,Cw,dd,dx,dy,dz,nx,ny,nz,dte,...
IN_level,OUT_level,dt,ONE,ZZ,M,max_conc,D_solute);
C_active=C_active.*IN_level;

% %saves the values from the first step of the simulation
% fname=[pillstyle,'step_',num2str(stepp+shift)];
% save (fname);
```

```
stepp=2;
newstep=2;
for stepp=2:tstep

    OUT_level = (max(L,ZZ)./L);
    IN_level  = (min(L,ZZ)./L);
    dtplus=dtplus+dt;



    if dtplus>=newtime
        Lxp(:,1:nx-1,:) = L(:,2:nx,:);
        Lxp(:,nx,:) = 2*L(:,nx,:)-L(:,nx-1,:);
        Lxm(:,2:nx,:) = L(:,1:nx-1,:);
        Lxm(:,1,:) = 2*L(:,1,:)-L(:,2,:);

        Lyp(2:ny,:,:) = L(1:ny-1,:,:);
        Lyp(1,:,:) = 2*L(1,:,:)-L(2,:,:);
        Lym(1:ny-1,:,:) = L(2:ny,:,:);
        Lym(ny,:,:) = 2*L(ny,:,:)-L(ny-1,:,:);

        Lzm(:,:,2:nz) = L(:,:,1:nz-1);
        Lzm(:,:,1) = 2*L(:,:,1)-L(:,:,2);
        Lzp(:,:,1:nz-1) = L(:,:,2:nz);
        Lzp(:,:,nz) = 2*L(:,:,nz)-L(:,:,nz-1);

        GradLx = (Lxp - Lxm)/(2*dx);
        GradLy = (Lyp - Lym)/(2*dy);
        GradLz = (Lzp - Lzm)/(2*dz);
        Norm = (GradLx.^2 + GradLy.^2 + GradLz.^2).^(1/2) + EPS;
```

```
P1=-GradLx./Norm;
P2=GradLy./Norm;
P3=-GradLz./Norm;
[CC]=Curvature3D(P1,P2,P3,nx,ny,nz,dx,dy,dz);
[CC_fac]=Curv_factor(CC,curve_shift,index);
[F2]=changeFtype(F,X,Y,tablet_radius,P1,P2,P3,nx,ny,nz,erosion_type,...
CC_fac,ONE,Amp);
F2=F2.*normalizer;

%~~~~ SOLUTION OF PARABOLIC EQUATION  ~~~~~%

%~~~~ LEVEL SET UPDATE UNDER VELOCITY FIELD ~~~~~%


% Erosion rate of the surface

% Changes to surface location are made with this function.  It uses
% the matrices built above to calculate the necessary changes for
% the level set values and then implements them.  It also reduces
% the active concentration to correspond to the loss of volume of
% the outermost layer of the tablet.
[critical,timenow,dL,L,k,IN_level,OUT_level,dt,C_active]=...
    Level_update_3D(dd,F2,timenow,dtplus,L,Lxm,Lym,Lzm,Lxp,Lyp,...
    Lzp,ZZ,ONE,k,dx,dy,dz,safety,C_active,index);




OUT_level = (max(L,ZZ)./L);
IN_level  = (min(L,ZZ)./L);
```

```
%~~~~ INDEX FOR REGULAR/IRREGULAR ~~~~~%

%index=1  ->Irregular Inside
%index=-1 ->Irregular Outside
%index=0  ->Regular

[index]=index_irregular3(L,Lxp,Lxm,Lyp,Lym,Lzp,Lzm,ny,nx,nz);



SV=C_active.*cell_volume;
vol_dissolved=1-(SV./SV_original)-OUT_level;

[Ng,Nb,Nc,Nd,pointg,pointb,pointc,pointd,index2]=...
    boundary_ghost3D_coord(index,L,x,y,z,P1,P2,P3,maxN,nx,ny,nz,...
    index2,Ng,Nb,Nc,Nd,pointg,pointb,pointc,pointd,dx,dy,dz);

dtplusnow=dtplus;
dtplus=0;
if critical>dtmax
    critical=dtmax;
end
smaller_dt=min(dtmax,dtcw);
if critical>smaller_dt
    factor=min(200,round(.9*(critical/smaller_dt)));
else
    smaller_dt=critical;
    factor=1;
end
```

```
    newstep=stepp+factor;
    newtime=factor*smaller_dt;
end

[C_interface]=determine_interface(C_solute,index,interface_factor,ZZ);

[Cw,C_solute]=boundary_ghost3D(nz,ny,nx,index,index2,Cw_interface,...
    Cw,C_interface,C_solute,Ng,Nb,Nc,Nd,pointg,pointb,pointc,pointd);



[Cw,dtcw,dCw]=water_level2(Cw,Dw,dd,dx,dy,dz,nx,ny,nz,dte,IN_level,OUT_level,dt,ONE)
Cw=max(Cw,OUT_level);
Cw=min(Cw,ONE);

[Surf]=Surf_calc(C_active,cell_volume,particle_volume);
%[dt_conc,C_active,dC_active]=Active_conc(C_active,Cw,dd,dx,dy,dz,nx,ny,nz,dte,...
IN_level,OUT_level,dt,ONE,ZZ,M,max_conc,Surf);
[C_solute,C_active,dC_active,Dw]=Active_conc4(C_solute,C_active,Cw,...
    IN_level,dt,ONE,ZZ,M,max_conc,Surf,Pen_coeff,porosity_factor,Conc_active,Dw);
[dt_solute,C_solute,dc_solute]=Solute_conc(C_solute,Cw,dd,dx,dy,dz,...
    nx,ny,nz,dte,IN_level,OUT_level,dt,ONE,ZZ,M,max_conc,D_solute);

C_active=C_active.*IN_level;



now_time=now_time+dt;

dt=min(min(critical,dtcw),dtmax);
```

```
%dt=min(dt,dt_solute);
%dumps necessary data into a save file
if  rem(stepp,500)==0
    percent_released=100*(1-(sum(sum(sum(C_active)))/total_active_conc));
    percent_flood=100*(((sum(sum(sum(Cw)))-initial_out)/initial_in));
    release_profile(time_step,1)=percent_released;
    release_profile(time_step,2)=now_time;
    flood_profile(time_step,1)=percent_flood;
    flood_profile(time_step,2)=now_time;
    [Tablet_Volume(time_step,1)]=Level_Set_Volume(IN_level,index,ZZ,...
        nx,ny,nz,cell_volume,dx,dy,dz,L,P1,P2,P3);
    Tablet_Volume(time_step,2)=Tablet_volume-Tablet_Volume(time_step,1);
    Tablet_Volume(time_step,3)=Tablet_Volume(time_step,2)/Tablet_volume;
    release_time=release_time+120;
    time_step=time_step+1;

    %         rpUtilsProgress((stepp/tstep*100),'Iterating');
end
if  now_time>release_time2
    percent_released=100*(1-(sum(sum(sum(C_active)))/total_active_conc));
    percent_flood=100*(((sum(sum(sum(Cw)))-initial_out)/initial_in));
    release_profile2(time_step2,1)=percent_released;
    release_profile2(time_step2,2)=now_time;
    flood_profile2(time_step2,1)=percent_flood;
    flood_profile2(time_step2,2)=now_time;
    [Tablet_Volume2(time_step2,1)]=Level_Set_Volume(IN_level,index,...
        ZZ,nx,ny,nz,cell_volume,dx,dy,dz,L,P1,P2,P3);
    Tablet_Volume2(time_step2,2)=Tablet_volume-Tablet_Volume2(time_step,1);
    Tablet_Volume2(time_step2,3)=100*Tablet_Volume2(time_step,2)/Tablet_volume;
    fname=[Run_name,'step_',num2str(release_time2+shift)];
```

```
    save (fname, 'release_profile2', 'C_active', 'C_solute' ,'Cw', 'L');
    tecplot_Level_Set(x,y,z,nx,ny,nz,L,pillstyle,t);
    release_time2=release_time2+120;
    time_step2=time_step2+1;
    %  rpUtilsProgress(percent_released,'Iterating');
end
if now_time>movie_counter
            fname=[Run_name,'movie_step_',num2str(round(now_time)+shift)];
    save (fname, 'release_profile2', 'C_active', 'C_solute' ,'Cw', 'L');
    movie_counter=movie_counter+60;
end
if percent_released>max_release %||percent_released<0

    percent_released=100*(1-(sum(sum(sum(C_active)))/total_active_conc));
    fakestep=ceil(stepp/release_step);
    release_profile(fakestep,1)=percent_released;
    release_profile(fakestep,2)=timenow;
    fname=[Run_name,'step_',num2str(stepp+shift)];
    save (fname)
    break;
elseif timenow>final_time %||percent_released<0

    percent_released=100*(1-(sum(sum(sum(C_active)))/total_active_conc));
    fakestep=ceil(stepp/release_step);
    release_profile(fakestep,1)=percent_released;
    release_profile(fakestep,2)=timenow;
    fname=[Run_name,'step_',num2str(stepp+shift)];
    save (fname)
    break;
end
```

```
end


endstep=length(release_profile);









TimeTaken=toc/60
fname=['Output/',Run_name]
save (fname)



Error in group_dissolve33 at 4
Run_name=pillstyle; %#ok<NASGU>
```

.5 Helper Functions

.5.1 Curvature3D

This function calculates the curvature of the tablet/bulk fluid interface across the whole level set.

```
function [K]=Curvature3D(P1,P2,P3,nx,ny,nz,dx,dy,dz)
%~~~~~~ CURVATURE  ~~~~~%
        P1p(:,1:nx-1,:) = P1(:,2:nx,:);
        P1p(:,nx,:) = 2*P1(:,nx,:)-P1(:,nx-1,:);
        P1m(:,2:nx,:) = P1(:,1:nx-1,:);
        P1m(:,1,:) = 2*P1(:,1,:)-P1(:,2,:);
        dP1dx=(P1p-P1m)./(2*dx);
```

```
P2p(2:ny,:,:) = P2(1:ny-1,:,:);
P2p(1,:,:) = 2*P2(1,:,:)-P2(2,:,:);
P2m(1:ny-1,:,:) = P2(2:ny,:,:);
P2m(ny,:,:) = 2*P2(ny,:,:)-P2(ny-1,:,:);
dP2dy=(P2p-P2m)./(2*dy);

P3m(:,:,2:nz) = P3(:,:,1:nz-1);
P3m(:,:,1) = 2*P3(:,:,1)-P3(:,:,2);
P3p(:,:,1:nz-1) = P3(:,:,2:nz);
P3p(:,:,nz) = 2*P3(:,:,nz)-P3(:,:,nz-1);
dP3dz=(P3p-P3m)./(2*dz);

K=dP1dx+dP2dy+dP3dz;
```

.5.2 Curv factor

Calculates the curvature factor based on the current interface curvature.

```
function[K_fac]=Curv_factor(K,curve_shift,index)

K_fac=((erf(K+curve_shift)+1)/2).*abs(index);
```

.5.3 changeFtype

This function is responsible for calculating the velocity of the interface based on the assigned uniform erosion speed and the type of erosion being considered; Uniform, Density Based, Shear Based, Shear Density Based.

```
function [F2]=changeFtype(F,X,Y,Ri,P1,P2,P3,nx,ny,nz,type,index,ONE,Amp)

if type==1
F2=F;
```

```
elseif type==2
%density based erosion only
pos=(pi*sqrt((X.^2+Y.^2)./Ri^2));
density_factor=ONE+Amp*cos(pos);
F2=F./density_factor;

elseif type==3
%fluid shear based erosion only
a=.5;
b=.25;
c=2;
theta=atan2(P2,P1);
erosion_factor=(abs(a.*sin(theta)+b.*sin(c.*theta)));
normalizere=max(max(max(erosion_factor)));
erosion_factor=erosion_factor/normalizere;
F2=F+(F.*erosion_factor.*(P1.^2+P2.^2)+a.*F.*(P3.^2));

elseif type==4
%combined fluid shear, uniform and density based erosion
a=1;
b=.5;
c=2;
theta=atan2(P2,P1);
erosion_factor=(abs(a.*sin(theta)+b.*sin(c.*theta)));
normalizere=max(max(max(erosion_factor)));
erosion_factor=erosion_factor/normalizere;
pos=(pi*sqrt((X.^2+Y.^2)./Ri^2));
density_factor=ONE+Amp*cos(pos);
F2=(F+(F.*erosion_factor.*(P1.^2+P2.^2)+a.*F.*(P3.^2)))./density_factor;
```

```
end
```

.5.4 Level update 3D

This function updates the level set using the previously calculated interfacial velocity, also modifying the current API concentration due to volume loss of the outermost cells.

```
function [critical,timenow,dL,L,k,IN_level,OUT_level,dt,C_active]=...
    Level_update_3D(dd,F,timenow,dt,L,Lxm,Lym,Lzm,Lxp,Lyp,Lzp,ZZ,ONE,k,...
    dx,dy,dz,safety,C_active,index)

    critical = dd/(max(max(max(abs(F))))*safety);
    %dt=critical*.5;
    timenow=timenow+dt;

    Deltap = sqrt(...
        ((max(L-Lxm,ZZ)).^2 + (min(Lxp-L,ZZ)).^2)/dx ...
        + ((max(L-Lym,ZZ)).^2 + (min(Lyp-L,ZZ)).^2)/dy ...
        + ((max(L-Lzm,ZZ)).^2 + (min(Lzp-L,ZZ)).^2)/dz);

    Deltam = sqrt(...
        ((min(L-Lxm,ZZ)).^2 + (max(Lxp-L,ZZ)).^2)/dx ...
        + ((min(L-Lym,ZZ)).^2 + (max(Lyp-L,ZZ)).^2)/dy ...
        + ((min(L-Lzm,ZZ)).^2 + (max(Lzp-L,ZZ)).^2)/dz);

    dL = max(F,ZZ).*Deltap + min(F,ZZ).*Deltam;

%     ADD = (max(L,ZZ)./L);
    %   ADD=ones(ny,nx);
```

```
    %     L = (L - dt*dL.*ADD);
    DL=dt*dL;
    L = (L - DL);
    L=min(L,ONE);
%     MIN=-ONE;
    L=max(L,-ONE);

    C_active=C_active+max(index,ZZ).*(DL/dd);

    %mesh(X,Y,L)
    %       contour(X,Y,L,[0 0])
    %       pause(0.2);
    %      N(k)=getframe;
    inside=min(L,ZZ);
    IN_level=inside./L;
    OUT_level=ONE-IN_level;
    k=k+1;
```

.5.5 index irregular3

This function is used to populate and update the matrix storing the locations of the irregular points, those immediately adjacent to the tablet/bulk fluid interface.

```
function [index]=index_irregular3(L,Lxp,Lxm,Lyp,Lym,Lzp,Lzm,ny,nx,nz);



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

%%%INDEXING IRREGULAR POINTS %%%%%%%%%%%%%
```

```
L_max=max(max(max(max(max(Lxp,Lxm),Lyp),Lym),Lzp),Lzm);
L_min=min(min(min(min(min(Lxp,Lxm),Lyp),Lym),Lzp),Lzm);

for k=1:nz
  for i=1:ny
      for j=1:nx
              if (L_max(i,j,k)*L_min(i,j,k)<=0 & L(i,j,k)<0)
                  index(i,j,k)=1; % +1  points inside,
              end
              if (L_max(i,j,k)*L_min(i,j,k)<=0 & L(i,j,k)>0)
                  index(i,j,k)=-1; % -1 for points outside,
              end
              if (L_max(i,j,k)*L_min(i,j,k)>=0)
                  index(i,j,k)=0; % 0 for regular points
              end
          end
      end
  end
```

.5.6 boundary ghost3D coord

This function calculates the ghosting coefficients for the irregular points. The coordinates here are updated whenever the level set is updated, but not at every iteration. The ghosting values are calculated for the solvent penetration and solute diffusion at every step in a separate function using the coordinates calculated here.

```
function [Ng,Nb,Nc,Nd,pointg,pointb,pointc,pointd,index2]=boundary_ghost3D_coord...
(index,L,x,y,z,P1,P2,P3,maxN,nx,ny,nz,index2,Ng,Nb,Nc,Nd,pointg,pointb,pointc,...
```

```
pointd,dx,dy,dz)


for k=1:nz
    for i=1:ny
        for j=1:nx
            if index(i,j,k)==-1

                PIy=-P2(i,j,k)*L(i,j,k)+y(i);
                PIx=P1(i,j,k)*L(i,j,k)+x(j);
                PIz=P3(i,j,k)*L(i,j,k)+z(k);
                points=[PIx,PIy,PIz];

                corner1=[i,j,k];
                corner2=[i+sign(P2(i,j,k)),j,k];
                corner3=[i,j+sign(P1(i,j,k)),k];
                corner4=[i+sign(P2(i,j,k)),j+sign(P1(i,j,k)),k];
                corner5=[i,j,k+sign(P3(i,j,k))];
                corner6=[i+sign(P2(i,j,k)),j,k+sign(P3(i,j,k))];
                corner7=[i,j+sign(P1(i,j,k)),k+sign(P3(i,j,k))];
                corner8=[i+sign(P2(i,j,k)),j+sign(P1(i,j,k)),k+sign(P3(i,j,k))];

                if ((abs(sign(P1(i,j,k))))+(abs(sign(P2(i,j,k))))+...
                (abs(sign(P3(i,j,k)))))<2
                    [Ng(i,j,k),pointb(i,j,k,:),index2(i,j,k)]=natural_coordinates1D...
                    (L,P1,P2,P3,i,j,k);
                elseif ((abs(sign(P1(i,j,k))))+(abs(sign(P2(i,j,k))))+...
                (abs(sign(P3(i,j,k)))))<3
                    Pmax=max(abs(P1(i,j,k)),max(abs(P2(i,j,k)),abs(P3(i,j,k))));
                    Pmin=min(abs(P1(i,j,k)),min(abs(P2(i,j,k)),abs(P3(i,j,k))));
                    if P1(i,j,k)==Pmax
```

```
        if abs(P2(i,j,k))==Pmin
            [Ng(i,j,k),Nb(i,j,k),Nc(i,j,k),pointg(i,j,k,:),...
            pointb(i,j,k,:),pointc(i,j,k,:),index2(i,j,k)]=...
            natural_coordinates2D(corner1,corner3,corner7,points,...
            x,y,z,maxN);
        else
            [Ng(i,j,k),Nb(i,j,k),Nc(i,j,k),pointg(i,j,k,:),...
            pointb(i,j,k,:),pointc(i,j,k,:),index2(i,j,k)]=...
            natural_coordinates2D(corner1,corner3,corner4,points,...
            x,y,z,maxN);
        end
    elseif abs(P2(i,j,k))==Pmax
        if abs(P1(i,j,k))==Pmin
            [Ng(i,j,k),Nb(i,j,k),Nc(i,j,k),pointg(i,j,k,:),...
            pointb(i,j,k,:),pointc(i,j,k,:),index2(i,j,k)]=...
            natural_coordinates2D(corner1,corner2,corner6,points,...
            x,y,z,maxN);
        else
            [Ng(i,j,k),Nb(i,j,k),Nc(i,j,k),pointg(i,j,k,:),...
            pointb(i,j,k,:),pointc(i,j,k,:),index2(i,j,k)]=...
            natural_coordinates2D(corner1,corner2,corner4,points,...
            x,y,z,maxN);
        end
    else
        if abs(P2(i,j,k))==Pmin
            [Ng(i,j,k),Nb(i,j,k),Nc(i,j,k),pointg(i,j,k,:),...
            pointb(i,j,k,:),pointc(i,j,k,:),index2(i,j,k)]=...
            natural_coordinates2D(corner1,corner5,corner7,points,...
            x,y,z,maxN);
        else
```

```
                    [Ng(i,j,k),Nb(i,j,k),Nc(i,j,k),pointg(i,j,k,:),...
                    pointb(i,j,k,:),pointc(i,j,k,:),index2(i,j,k)]=...
                    natural_coordinates2D(corner1,corner5,corner6,points,...
                    x,y,z,maxN);
                end
            end
        else
            Pmax=max(abs(P1(i,j,k)),max(abs(P2(i,j,k)),abs(P3(i,j,k))));
            if abs(P1(i,j,k))==Pmax
                [Ng(i,j,k),Nb(i,j,k),Nc(i,j,k),Nd(i,j,k),pointg(i,j,k,:),...
                pointb(i,j,k,:),pointc(i,j,k,:),pointd(i,j,k,:),index2(i,j,k)]=.
                natural_coordinates3D(corner1,corner4,corner3,corner7,points,...
                x,y,z,maxN,dx,dy,dz);
            elseif abs(P2(i,j,k))==Pmax
                [Ng(i,j,k),Nb(i,j,k),Nc(i,j,k),Nd(i,j,k),pointg(i,j,k,:),...
                pointb(i,j,k,:),pointc(i,j,k,:),pointd(i,j,k,:),index2(i,j,k)]=.
                natural_coordinates3D(corner1,corner6,corner2,corner4,points,...
                x,y,z,maxN,dx,dy,dz);
            else
                [Ng(i,j,k),Nb(i,j,k),Nc(i,j,k),Nd(i,j,k),pointg(i,j,k,:),...
                pointb(i,j,k,:),pointc(i,j,k,:),pointd(i,j,k,:),index2(i,j,k)]=.
                natural_coordinates3D(corner1,corner7,corner5,corner6,points,...
                x,y,z,maxN,dx,dy,dz);
            end
        end
    end
  end
end
```

.5.7 bilinear coeff 3D

```
function [X1,X5,Y1,Y5,Z1,Z5,D1,D2,D3,Xp,Yp,Zp,Iym,Ixm,Izm,Iyp,Ixp,Izp,DB]=...
bilinear_coeff_3D(dd,P1,P2,P3,X,Y,Z,nx,ny,nz,index,dx,dy,dz,L,x,y,z)
% Xp=dd*P1+X;
% Yp=dd*P2+Y;
% Zp=dd*P3+Z;
                Xp=2*L.*P1+X;
                Yp=2*L.*P2+Y;
                Zp=2*L.*P3+Z;
                pxb=L.*P1+X;
                pyb=L.*P2+Y;
                pzb=L.*P3+Z;


DB=zeros(ny,nx,nz);
for k=1:nz
    for i=1:ny
        for j=1:nx
            if index(i,j,k)==-1
% %                 px=-2*L(i,j,k)*P1(i,j,k)+X(i,j,k);
% %                 py=-2*L(i,j,k)*P2(i,j,k)+Y(i,j,k);
% %                 pz=-2*L(i,j,k)*P3(i,j,k)+Z(i,j,k);
                pbx=L(i,j,k)*P1(i,j,k)+X(i,j,k);
                pby=L(i,j,k)*P2(i,j,k)+Y(i,j,k);
                pbz=L(i,j,k)*P3(i,j,k)+Z(i,j,k);


                px=Xp(i,j,k);
                py=Yp(i,j,k);
                pz=Zp(i,j,k);
                [ixm,ixp,iyp,iym,izp,izm]= neighbour3(pbx,pby,pbz,x,y,z,nx,ny,nz,...
                dx,dy,dz);
```

```
Ixm(i,j,k)=ixm;
Ixp(i,j,k)=ixp;
Iyp(i,j,k)=iyp;
Iym(i,j,k)=iym;
Izp(i,j,k)=izp;
Izm(i,j,k)=izm;



X1(i,j,k)=X(Iym(i,j,k),Ixm(i,j,k),Izp(i,j,k));
X2(i,j,k)=X(Iym(i,j,k),Ixp(i,j,k),Izp(i,j,k));
X3(i,j,k)=X(Iyp(i,j,k),Ixp(i,j,k),Izp(i,j,k));
X4(i,j,k)=X(Iyp(i,j,k),Ixm(i,j,k),Izp(i,j,k));
X5(i,j,k)=X(Iym(i,j,k),Ixm(i,j,k),Izm(i,j,k));
X6(i,j,k)=X(Iym(i,j,k),Ixp(i,j,k),Izm(i,j,k));
X7(i,j,k)=X(Iyp(i,j,k),Ixp(i,j,k),Izm(i,j,k));
X8(i,j,k)=X(Iyp(i,j,k),Ixm(i,j,k),Izm(i,j,k));

Y1(i,j,k)=Y(Iym(i,j,k),Ixm(i,j,k),Izp(i,j,k));
Y2(i,j,k)=Y(Iym(i,j,k),Ixp(i,j,k),Izp(i,j,k));
Y3(i,j,k)=Y(Iyp(i,j,k),Ixp(i,j,k),Izp(i,j,k));
Y4(i,j,k)=Y(Iyp(i,j,k),Ixm(i,j,k),Izp(i,j,k));
Y5(i,j,k)=Y(Iym(i,j,k),Ixm(i,j,k),Izm(i,j,k));
Y6(i,j,k)=Y(Iym(i,j,k),Ixp(i,j,k),Izm(i,j,k));
Y7(i,j,k)=Y(Iyp(i,j,k),Ixp(i,j,k),Izm(i,j,k));
Y8(i,j,k)=Y(Iyp(i,j,k),Ixm(i,j,k),Izm(i,j,k));

Z1(i,j,k)=Z(Iym(i,j,k),Ixm(i,j,k),Izp(i,j,k));
Z2(i,j,k)=Z(Iym(i,j,k),Ixp(i,j,k),Izp(i,j,k));
```

```
                Z3(i,j,k)=Z(Iyp(i,j,k),Ixp(i,j,k),Izp(i,j,k));
                Z4(i,j,k)=Z(Iyp(i,j,k),Ixm(i,j,k),Izp(i,j,k));
                Z5(i,j,k)=Z(Iym(i,j,k),Ixm(i,j,k),Izm(i,j,k));
                Z6(i,j,k)=Z(Iym(i,j,k),Ixp(i,j,k),Izm(i,j,k));
                Z7(i,j,k)=Z(Iyp(i,j,k),Ixp(i,j,k),Izm(i,j,k));
                Z8(i,j,k)=Z(Iyp(i,j,k),Ixm(i,j,k),Izm(i,j,k));

                D1(i,j,k)=(Xp(i,j,k)-X(Iym(i,j,k),Ixm(i,j,k),Izm(i,j,k)))/dx;
                D2(i,j,k)=(Yp(i,j,k)-Y(Iym(i,j,k),Ixm(i,j,k),Izm(i,j,k)))/dy;
                D3(i,j,k)=(Zp(i,j,k)-Z(Iym(i,j,k),Ixm(i,j,k),Izm(i,j,k)))/dz;
                D4=abs(pbx-px)/dx;
                D5=abs(pby-py)/dy;
                D6=abs(pbz-pz)/dz;
                DB(i,j,k)=(1-D4)*(1-D5)*(1-D6);

            end
        end
    end
end
```

.5.8 determine interface

This function determines the concentration of solute which will be used as the surface concentration when calculating the ghosting values.

```
function[C_interface]=determine_interface(C_solute,index,interface_factor,ZZ)

index_factor=max(index,ZZ);
avg_outer_solute_conc=sum(sum(sum(index_factor.*C_solute)))/sum(sum(sum(index_factor)));
C_interface=avg_outer_solute_conc*interface_factor;
```

.5.9 boundary ghost3D

This function creates the ghosting matrix at every iteration. The values created are for both the solvent concentration(C1) and solute concentration(C2).

```
function [C1,C2]=boundary_ghost3D(nz,ny,nx,index,index2,C1_interface,C1,...
C2_interface,C2,Ng,Nb,Nc,Nd,pointg,pointb,pointc,pointd);



for k=1:nz
    for i=1:ny
        for j=1:nx
            if index(i,j,k)==-1
                if index2(i,j,k)==1
                    C1(i,j,k)=(C1(pointb(i,j,k,1),pointb(i,j,k,2),...
                    pointb(i,j,k,3))-C1_interface)*Ng(i,j,k)+C1_interface;
                    C2(i,j,k)=(C2(pointb(i,j,k,1),pointb(i,j,k,2),...
                    pointb(i,j,k,3))-C2_interface)*Ng(i,j,k)+C2_interface;
                elseif index2(i,j,k)==2
                    C1(i,j,k)=(C1_interface-Nb(i,j,k)*C1(pointb(i,j,k,1),...
                    pointb(i,j,k,2),pointb(i,j,k,3))-Nc*C1(pointc(i,j,k,1),...
                    pointc(i,j,k,2),pointc(i,j,k,3)))/Ng;
                    C2(i,j,k)=(C2_interface-Nb(i,j,k)*C2(pointb(i,j,k,1),...
                    pointb(i,j,k,2),pointb(i,j,k,3))-Nc*C2(pointc(i,j,k,1),...
                    pointc(i,j,k,2),pointc(i,j,k,3)))/Ng;
                elseif index2(i,j,k)==3
                    C1(i,j,k)=(C1_interface-Nb(i,j,k)*C1(pointb(i,j,k,1),...
                    pointb(i,j,k,2),pointb(i,j,k,3))-Nc*C1(pointc(i,j,k,1),...
                    pointc(i,j,k,2),pointc(i,j,k,3))-Nd*C1(pointd(i,j,k,1),...
                    pointd(i,j,k,2),pointd(i,j,k,3)))/Ng;
                    C2(i,j,k)=(C2_interface-Nb(i,j,k)*C2(pointb(i,j,k,1),...
```

```
                    pointb(i,j,k,2),pointb(i,j,k,3))-Nc*C2(pointc(i,j,k,1),...
                    pointc(i,j,k,2),pointc(i,j,k,3))-Nd*C2(pointd(i,j,k,1),...
                    pointd(i,j,k,2),pointd(i,j,k,3)))/Ng;
                else
                    C1(i,j,k)=C1_interface;
                    C2(i,j,k)=C2_interface;
                end
            end
        end
    end
end
```

.5.10 water level2

this function calculates the solvent penetration of the tablet using a Fick's Second Law approach.

```
function [Cw,dtcw,dCw,dcw]=water_level2(Cw,Dw,dd,dx,dy,dz,nx,ny,nz,dte,IN_level,...
OUT_level,dt,ONE)
%Creates matrices for use with Taylor series approximation
Cwxp(:,1:nx-1,:)=Cw(:,2:nx,:);Cwxp(:,nx,:)=1;
Cwxm(:,2:nx,:)=Cw(:,1:nx-1,:);Cwxm(:,1,:)=1;
Cwyp(2:ny,:,:)=Cw(1:ny-1,:,:);Cwyp(1,:,:)=1;
Cwym(1:ny-1,:,:)=Cw(2:ny,:,:);Cwym(ny,:,:)=1;
Cwzp(:,:,1:nz-1)=Cw(:,:,2:nz);Cwzp(:,:,nz)=1;
Cwzm(:,:,2:nz)=Cw(:,:,1:nz-1);Cwzm(:,:,1)=1;

%calculates change of water concentration in pill per unit time
dcw=-Dw.*(Cwxp/dx^2+Cwxm/dx^2+Cwyp/dy^2+Cwym/dy^2+Cwzp/dz^2+Cwzm/dz^2 ...
    -2*Cw*(1/dx^2 + 1/dy^2+ 1/dz^2));
```

```
%calculates critical timestep size
critical = dd/max(max(max(abs(IN_level.*dcw))));
dtcw=.95*critical;

%timestep change of water concentration
dCw=dt*dcw;

%new water concentration in pill
Cw=min((IN_level.*(Cw-dCw)+OUT_level),ONE);
```

.5.11 Surf calc

This function calculates the combined surface area of the particles in each cell.

```
function [SA]=Surf_calc(C_active,cell_volume,particle_volume)
num_part=C_active.*cell_volume./particle_volume;
SA=4*pi*((3*C_active*cell_volume)./(4*pi*num_part)).^(2/3).*num_part;
SA(isnan(SA))=0;
```

.5.12 Active conc4

This function calculates the volume released form the active particles and updates the API concentration as well as the solvent penetration coefficient in cases where dissolution of the actives results in faster absorption of solvent into the tablet.

```
function [C_solute,C_active,dc,Dw]=Active_conc4(C_solute,C_active,Cw,...
IN_level,dt,ONE,ZZ,M,max_conc,Surf,Pen_coeff,porosity_factor,Conc_active,Dw)
%Creates matrices for use with Taylor series approximation

%calculates change of water concentration in pill per unit time
dc=IN_level.*Cw.*M.*Surf.*(C_active-C_solute);
```

```
% % %calculates critical timestep size
% % critical = dd/max(max(max(abs(IN_level.*dc))));
% % dt_conc=.95*critical;

%timestep change of water concentration
dC=dt*dc;
dC=max(dC,ZZ);

%new water concentration in pill
C_active=min(max_conc,max(IN_level.*(C_active-dC),ZZ));
C_solute=IN_level.*(C_solute+dC);

Pen_increase=-porosity_factor.*dc;
Dw=max(ZZ,(Dw-isnan(Pen_increase)));
```

.5.13 Solute conc

This function calculates the evolution of the drug solute. It updates the concetration of solute inside the tablet based on the volume of solvent added from dissolving API and the volume which moves from or to another cell and for the irregular points, the volume released from the tablet.

```
function [dt_conc,C,dc]=Solute_conc(C,Cw,dd,dx,dy,dz,nx,ny,nz,dte,...
IN_level,OUT_level,dt,ONE,ZZ,M,max_conc,D_solute)

%the effective diffusion coefficient(alpha) is calculated as a function of solvent
%concentration, active particle surface area and a prescribed diffusion
%constant representing the chosen API-solvent system.
alpha=Cw.*D_solute;
```

```
%Creates matrices for use with Taylor series approximation
alphaxp(:,1:nx-1,:)=alpha(:,2:nx,:);alphaxp(:,nx,:)=0;
alphaxm(:,2:nx,:)=alpha(:,1:nx-1,:);alphaxm(:,1,:)=0;
alphaym(2:ny,:,:)=alpha(1:ny-1,:,:);alphaym(1,:,:)=0;
alphayp(1:ny-1,:,:)=alpha(2:ny,:,:);alphayp(ny,:,:)=0;
alphazp(:,:,1:nz-1)=alpha(:,:,2:nz);alphazp(:,:,nz)=0;
alphazm(:,:,2:nz)=alpha(:,:,1:nz-1);alphazm(:,:,1)=0;

Cxp(:,1:nx-1,:)=C(:,2:nx,:);Cxp(:,nx,:)=0;
Cxm(:,2:nx,:)=C(:,1:nx-1,:);Cxm(:,1,:)=0;
Cym(2:ny,:,:)=C(1:ny-1,:,:);Cym(1,:,:)=0;
Cyp(1:ny-1,:,:)=C(2:ny,:,:);Cyp(ny,:,:)=0;
Czp(:,:,1:nz-1)=C(:,:,2:nz);Czp(:,:,nz)=0;
Czm(:,:,2:nz)=C(:,:,1:nz-1);Czm(:,:,1)=0;



%calculates change of water concentration in pill per unit time
dc=IN_level.* ...
(((alphaxp-alphaxm).*(Cxp-Cxm))./(2*dx)^2+ ...
((alphayp-alphaym).*(Cyp-Cym))./(2*dy)^2+ ...
    ((alphazp-alphazm).*(Czp-Czm))./(2*dz)^2)+ ...
    alpha.*(Cxp/dx^2+Cxm/dx^2+Cyp/dy^2+Cym/dy^2+ ...
Czp/dz^2+Czm/dz^2-2*C*(1/dx^2 + 1/dy^2+ 1/dz^2));



%calculates critical timestep size
critical = dd/max(max(max(abs(IN_level.*dc))));
dt_conc=.95*critical;
```

```
%timestep change of water concentration
dC=dt*dc;

%new water concentration in pill
C=min(max_conc,max(IN_level.*(C+dC),ZZ));
```

.5.14 Level Set Volume

This function uses the current level set to determine the remaining volume of intact tablet.

```
function[Tablet_Volume]=Level_Set_Volume(IN_level,index,ZZ,nx,ny,nz,...
cell_volume,dx,dy,dz,L,P1,P2,P3)

Tablet_Volume=0;
Tablet_Volume=sum(sum(sum(IN_level-max(index,ZZ)/2)))*cell_volume;
avg_dist=(dy+dx+dz)/3;
for i=1:ny
    for j=1:nx
        for k=1:nz
            if index(i,j,k)==1
% %                 dist=abs(P1(i,j,k)*dx)+abs(P2(i,j,k)*dy)+abs(P3(i,j,k)*dz);
                dist=sqrt((P1(i,j,k)*dx)^2+(P2(i,j,k)*dy)^2+(P3(i,j,k)*dz)^2);
                Tablet_Volume=Tablet_Volume+(cell_volume*sqrt((L(i,j,k))^2/(dist)^2));
            end
        end
    end
end
```

References

[1] Charalambos G. Varelas, David G. Dixon, and Carol A. Steiner. Zero-order release from biphasic polymer hydrogels. *Journal of Controlled Release*, 34(3):185 – 192, 1995.

[2] Lobo J.M.S. Costa, P. Modeling and comparison of dissolution profiles. *European Journal of Pharmaceutical Sciences*, 13:123–133, 2001.

[3] Aristides Dokoumetzidis and Panos Macheras. A century of dissolution research: From noyes and whitney to the biopharmaceutics classification system. *International Journal of Pharmaceutics*, 321(1-2):1 – 11, 2006.

[4] Stuart Feldman Milo Gibaldi. Establishment of sink conditions in dissolution rate determinations. theoretical considerations and application to nondisintegrating dosage forms. *Journal of Pharmaceutical Sciences*, 56(10):1238–1242, 1967.

[5] Ito Y Teramura S Okado J. Kitazawa S, Johno I. Effects of hardness on the disintegration time and the dissolution rate of uncoated caffeine tablets. *J Pharm Pharmacol.*, 27(10):765–70, October 1975.

[6] Hirotaka Endoh Kozo Ishikawa Kazuhiko Juni Masahiro Nakano Toshinobu Seki, Takeo Kawaguchi. Controlled release of 3prime, 5prime-diester prodrugs of 5-fluoro-2prime-deoxyuridine from poly-2-lactic acid microspheres. *Journal of Pharmaceutical Sciences*, 79(11):985–987, 1990.

[7] Richard W. Korsmeyer, Robert Gurny, Eric Doelker, Pierre Buri, and Nikolaos A. Peppas. Mechanisms of solute release from porous hydrophilic polymers. *International Journal of Pharmaceutics*, 15(1):25 – 35, 1983.

[8] Goldberger A. Friedman M. Katzhendler I., Hoffman A. Modeling of drug release from erodible tablets. *Journal of Pharmaceutical Sciences*, 86(1):110–115, 1997.

[9] P.A.C. Gane, C.J. Ridgway, and E. Barcel. Analysis of pore structure enables improved tablet delivery systems. *Powder Technology*, 169(2):77 – 83, 2006.

[10] Per Borgquist, Anna Krner, Lennart Piculell, Anette Larsson, and Anders Axelsson. A model for the drug release from a polymer matrix tablet–effects of swelling and dissolution. *Journal of Controlled Release*, 113(3):216 – 225, 2006.

[11] Loo A. Lim T.S. Stepanek, F. Multiscale modelling methodology for virtual prototyping of effervescent tablets. *Journal of Pharmaceutical Sciences*, 95:1614–1625, 2006.

[12] Pal A. Sjoberg M. Carlsson M. Laurell E. Brasseur J.G. Abrahamsson, B. A novel in vitro and numerical analysis of shear-induced drug release from extended-release tablets in the fed stomach. *Pharmaceutical Research*, 22:1215–1226, 2005.

[13] Hurley N.J. Crane L. Healy A.M. Corrigan O.I. Gallagher K.M. McCarthy L.G. Crane, M. Simulation of the usp drug delivery problem using cfd: experimental, numerical and mathematical aspects. *Simulation Modelling Practice and Theory*, 12:147–158, 2003.

[14] Mayersohn M. Walker G.C. Cobby, J. Influence of shape factors on kinetics of drug release from matrix tablets. *Experimental. J. Pharm. Sci.*, 63:732737, 1974.

[15] C. W. Hirt and B. D. Nichols. Volume of fluid (vof) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39(1):201 – 225, 1981.

[16] J.A. Osher, Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science.* Cambridge University Press, 1996.

[17] Kinjal Dhruva. Modeling dynamic void growth and coalescence by plasticity and diffusion using a level set approach. Master's thesis, Rutgers University, Dept. of MAE, 2004.

[18] Franson N.M. Peppas, N.A. The swelling interface number as a criterion for prediction of diffusional solute release mechanisms in swellable polymers. *Journal of Polymer Science: Polymer Physics Edition*, 21(6):983–997, 2003.

[19] Peppas N.A. Siepmann, J. Hydrophilic matrices for controlled drug delivery: An improved mathematical model to predict the resulting drug release kinetics (the sequential layer model). *Pharmaceutical Research*, 17:1290–1298, 2000.

[20] J. Tritt-Goc, J. Kowalczuk, and N. Pislewski. Mri study of fickian, case ii and anomalous diffusion of solvents into hydroxypropylmethylcellulose. *Applied Magnetic Resonance*, 29:605–615, 2005. 10.1007/BF03166337.

[21] E. Brunner. Reaktionsgeschwindigkeit in heterogenen systemen. *Z. Phys. Chem.*, 47:56–102, 1904.

[22] Peppas N.A. Ritger, P.L. A simple equation for description of solute release i. fickian and non-fickian release from non-swellable devices in the form of slabs, spheres, cylinders or discs. *Journal of Controlled Release*, 5(1):23–35, 1986.

[23] Crane M. Ruskin H.J. Crane L. McMahon, N. The importance of boundary conditions in the simulation of dissolution in the usp dissolution apparatus. *Simulation Modelling Practice and Theory*, 15:247–255, 2007.

[24] I. C. Sinka, S. F. Burch, J. H. Tweed, and J. C. Cunningham. Measurement of density variations in tablets using x-ray computed tomography. *International Journal of Pharmaceutics*, 271(1-2):215 – 224, 2004.

[25] Virginie Busignies, Bernard Leclerc, Patrice Porion, Pierre Evesque, Guy Couarraze, and Pierre Tchoreloff. Quantitative measurements of localized density variations in cylindrical tablets using x-ray microtomography. *European Journal of Pharmaceutics and Biopharmaceutics*, 64(1):38 – 50, 2006.

[26] R. Bettini, P. L. Catellani, P. Santi, G. Massimo, N. A. Peppas, and P. Colombo. Translocation of drug particles in hpmc matrix gel layer: effect of drug solubility and influence on release rate. *Journal of Controlled Release*, 70(3):383 – 391, 2001.

[27] Paolo Colombo, Ruggero Bettini, and Nikolaos A. Peppas. Observation of swelling process and diffusion front position during swelling in hydroxypropyl methyl cellulose (hpmc) matrices containing a soluble drug. *Journal of Controlled Release*, 61(1-2):83 – 91, 1999.

[28] J. Kukura, J. L. Baxter, and F. J. Muzzio. Shear distribution and variability in the usp apparatus 2 under turbulent conditions. *International Journal of Pharmaceutics*, 279(1-2):9 – 17, 2004.

[29] Jennifer L. Baxter, Joseph Kukura, and Fernando J. Muzzio. Hydrodynamics-induced variability in the usp apparatus ii dissolution test. *International Journal of Pharmaceutics*, 292(1-2):17 – 28, 2005.

[30] J.W.Moore and H.H.Flanner. Mathematical comparison of curves with an emphasis on in vitro dissolution profiles. *Pharm. Tech.*, 20(6):64–74, 1996.

[31] Philip L. Ritger and Nikolaos A. Peppas. A simple equation for description of solute release ii. fickian and anomalous release from swellable devices. *Journal of Controlled Release*, 5(1):37 – 42, 1987.

[32] Nikolaos A. Peppas and Jennifer J. Sahlin. A simple equation for the description of solute release. iii. coupling of diffusion and relaxation. *International Journal of Pharmaceutics*, 57(2):169 – 172, 1989.

[33] Kosmas Kosmidis, Eleni Rinaki, Panos Argyrakis, and Panos Macheras. Analysis of case ii drug transport with radial and axial release from cylinders. *International Journal of Pharmaceutics*, 254(2):183 – 188, 2003.

[34] Hannu Laaksonen, Jouni Hirvonen, and Timo Laaksonen. Cellular automata model for swelling-controlled drug release. *International Journal of Pharmaceutics*, 380(1-2):25 – 32, 2009.

[35] Athanas Koynov. Correlation between excipient characteristics and product properties through numerical simulations of tablet compaction. AIChE, November 2008.

Vita

Daniel Braido

2004	Graduated from Carnegie Mellon University
2004-2011	**Attended Rutgers University** Majors: Mechanical Engineering.
2005 - 2011	Graduate Assistant.