ON SOME PROBLEMS IN SYSTEMS BIOLOGY AND
GEOMETRIC FLOWS

Tobias Gebhard

Abstract

This thesis consists of three distinct parts. The first part concerns spatial modeling of signaling pathways in epithelial cells, which are systems that enable cells to respond to outside stimuli and changes in the environment. The signaling is performed through a series of enzymatic reactions that start at the cellular membrane and end in the nucleus. We focus on modeling the transport of enzymes through different cells and show that the inclusion of diffusion in the model may have a large impact on the output, compared to spatially homogenous models. The system is modeled using partial differential equations in three space dimensions, which are solved using finite differences and the Immersed Interface Method.

The second part of the thesis discusses the Immersed Interface Method and its application to three dimensional, time dependent problems. Furthermore, the method is extended to work on bodies grids, which are grids with a special structure that makes it possible to reduce the number of grid nodes while retaining the same accuracy of approximation.

The third and last part contains the proof of convergence for an algorithm to compute generalized mean curvature flows with right angle boundary condition. Mean curvature flows describe the evolution of a surface which at each point is assigned a normal velocity depending on the mean curvature of the surface at that point. We consider flows where the velocity is equal to its own external normal, continuous function of the mean curvature. Furthermore, we assume that the surface is bounded inside a convex domain and that wherever it intersects the domain boundary, it should do so at a right angle. The algorithm is based on a convolution thresholding scheme and we show the convergence of the output, as the time step tends to zero, to the viscosity solution of the corresponding mean curvature PDE.

Keywords: systems biology; MAPK signaling pathways; reaction-diffusion equations; Immersed Interface Method; boundary grids; mean curvature flows; viscosity solutions; convolution thresholding schemes
Acknowledgments

First of all, I would like to thank my supervisor Alexa Hejzlar for coming up with many of the ideas behind this work and for his encouragement and good advice, as well as for pleasant times with guitar and drinks in hand.

I also wish to thank my co-supervisor Per Sørensen, Dept. of Cell and Molecular Biology, Uppsala University, for sharing his knowledge on signaling pathways and for reading and commenting on Part I of this thesis. Furthermore, my "pair" Ph.D. student in biology, Christ Molin, deserves thanks for interesting discussions regarding cell biology and other topics.

I would also like to thank Richard Grzybowski who has taken time to adapt his code for compiling mean curvature flows, which was used to generate the nice illustrations at the end of Part III.

I am grateful to Per Kumlin for reading and giving valuable comments on Part III of this thesis, and to Mohammad Amjady for doing the same work with Part II.

A thought of gratitude also goes to the National Research School in Computing and Biocomputing, which provided the financial support for my PhD studies.

Finally, I would like to thank my family for their care and support and my friends for giving me other things to do and think about.

Tobén Gabriell
Uppsala, March 2005
Preface

This thesis consists of three independent parts. The first part concerns spatial modeling of intracellular signaling pathways, which is the project supported by the National Research School in Genetics and Biocomputing. The text in part I is supposed to be less demanding than in the other two parts, making it accessible to biologists and other people with limited knowledge of mathematics. That said, there are some equations appearing now and then, but these parts could be skipped if the reader has trouble understanding the mathematical symbols. Also, having some background knowledge in biology is preferable, although some effort has been made to give the necessary background and to explain the terms used.

Part II describes the computational method used to perform the calculations in Part I, as well as an extension of that method to Boolean grids, which makes it possible to reduce the number of grid nodes when performing the calculations. Part II requires more mathematical knowledge, and a reader who is inexperienced in mathematics will find it hard to follow.

Part III is not related to the two first parts, but it is a continuation of my master thesis concerning algorithms for computing mean curvature flows; that is, the motion of surfaces which are assigned a velocity at each point depending on the mean curvature of the surface. Reading (and understanding) this part requires a proper mathematical background.
Spatial modeling of MAPK signaling pathways

Tobias Gebäck

Abstract

MAPK signaling pathways in eukaryotic cells are sequences of enzymatic reactions that convey a signal from the cellular membrane to the nucleus in response to some stimulus. They constitute an often viral signaling system which enables the cell to react to changes in the environment and to survive such changes.

We investigate the effects of spatial models for signaling pathways. First, we add a diffusion term to the Kholodnitsky model for general MAPK signaling pathways. This has a large effect on the amplitude of the oscillations that the model predicts, indicating that such oscillations have a smaller effect in a model that takes into account the spatial distribution of proteins, compared to the original space-independent model.

Second, we investigate a simple spatial model for the HOG pathway in the yeast Saccharomyces cerevisiae. We are able to reproduce the nuclear relocation of the Hog1 protein and also see that diffusion in the model is so fast that differences in protein concentration throughout the cell are small, even though reactions are localized only at the membrane.

The calculations are performed in three space dimensions using finite difference methods and the Immersed Interface Method, which is described in part II of this thesis.
1. Introduction

The ordinary baker's yeast, *Saccharomyces cerevisiae*, which is a unicellular fungus, is a very widely studied organism among cell biologists. The reason for this is many. One is that for a long time there has been a commercial interest for bread and bakers to understand the organism in order to manage its output of alcohol and carbon dioxide. Another reason is that it is a relatively simple unicellular eukaryotic organism that is easy to handle in the laboratory and may be used as a model organism for higher eukaryotic organisms, such as plants and animal cells. Nonetheless, another reason for studying yeast is that it is already very well studied, which means that more extensive studies can be performed, trying to understand more complex processes in the cell. For example, the complete yeast genome has been sequenced and it contains approximately 6,000 genes (or 5000 genes are), and hence a complete library of gene deletion has been set up for (almost) every single gene. This is a yeast which has its genome available and has this particular gene deleted from its genome, enabling biologists to easily study the effects of removing a gene from a cell under different conditions, thus hopefully learning more about the function of that gene. Also, there are many research groups continuously working on different aspects of the yeast cell, making the amount of data available exponentially large.

Having said that, however, it should be noted that even the single yeast cell is not at all understood by the biologists. There are many genes coding for proteins with unknown functions and even if the genes code for a protein with a known function, the function may depend on other protein and substances, so that the overall behavior is not very well understood anymore. The cell as a whole is a very complex system, where protein, RNA, DNA, and other molecules work together to define the behavior of the cell. And although there has been a tremendous increase in knowledge about the cell during the last decades, many parts of the complete system are not understood. For example, a single protein can be defined by a single gene, which may be studied to determine its amino acid sequence, its three-dimensional structure, its activity, etc. But it may bind to other proteins, etc. This gives very valuable information about the protein, but does not tell the whole story. Some important questions like "Where is it located?" or "What stimulates the protein?" or "What is it used for?" must also be answered to give a complete picture. The answers to this kind of questions do not depend solely on the protein itself, but also on other proteins and substances in the cell, as well as outside stimuli and the overall "state" of the cell.

The complexity of these questions is the basis for systems biology, which is the science that tries to look at larger systems of proteins and cellular functions, rather than mathematical modeling in order to understand the behavior of that particular system. There is no single definition of systems biology and it is not easy to try to come up with one, since these "systems" may be very different in character and the methods applied to study them may also vary accordingly. The philosophy of systems biology is not without controversy, since there is no long tradition of using mathematical modeling in cell biology, but there are a few examples where modeling has been successful as a complement to the experimental data in order to understand the behavior of a cellular system (see [2]).

1.1. Signaling pathways

One type of cellular systems that is suited for advanced modeling is signaling pathways. We will be concerned here with MAPK pathways. Specifically, we have the high Osmo-loyd (HOG) pathway in yeast, and also most of what is said here applies to other pathways as well. See [1, chapter 13] for general information about signaling pathways and [4] for a review of the HOG pathway. The signaling in a MAPK pathway starts at the cellular membrane, where it is activated by some stimulus, such as the presence of a specific substrate (e.g., glucose) in the environment or more general environmental change, such as changes in osmotic pressure or other properties of the environment.

This stimulus is sensed in one way or another by receptor or other mechanisms at the membrane. These sensing mechanisms then convey a signal to another protein by phosphorylation, or adding a phosphate group to one of the amino acids of the target. This starts a chain of phosphorylation events, which convey the signal through two or three steps, where each step consists of the phosphorylation of a target kinase, that is an enzyme which, once activated by phosphorylation, may phosphorylate other target proteins. So, as in figure 2.1, the signaling cascade starts from the MAP kinase, 

\[ \text{MAPK} \] which activates the MAP kinase kinase (MKK) which activates the MAP kinase (MAPK), which in turn serves to the signal when it may activate or deactivate transcription factors that control gene expression.

The MAPK may also have other functions by controlling the activity of enzymes throughout the cytoplasm and nucleus. Activation of kinases in the cell may require a double phosphorylation of two amino acids in the protein, which are both performed by the high-level kinase.

The effect of the signaling pathway is that the cell is able to sense changes in the environment and convey the information of this change to the nucleus or other lower parts. These the cell one produces the appropriate response to the stimulus, which is often vital for the survival of the cell. With the multiple steps in the chain, the cell is able to amplify the signal and may also become the trigger for some enzyme, connecting a downstream response to the signal. In addition to the activating kinases, there are also deactivating phosphatases, which remove the phosphate from the enzymes, thereby deactivating them (a dephosphorylation need not be deactivating, but in this case it is). When the stimulus disappears, or the cell has adapted to the new environment, the signaling pathway switches back off through the action of the phosphatases.

It should also be mentioned that (as always in biology) things are more complicated than they seem. For example, the same mechanism is often not very well known and may include many proteins; the phosphorylation events may take place when the kinases are activated in small or large protein complexes; and the cell is full of other enzymes and molecules which may influence the signaling pathway producing different results depending on which state the cell is in. The models we discuss here focus on...
the phosphorylation cascade and the movement of the phosphorylated MAPK from the membrane to the nucleus. They are of course great simplifications, but may hopefully provide some insight into the reality.

1.2 Outline

In the following sections, we study two spatial models of MAPK signaling pathways. In section 2, we investigate the effects of diffusion on the oscillations predicted by a model of a MAPK cascade, including a negative feedback loop. In section 3, we study a model of the major phosphorylation of the giant MAPK Hck. The two sections are almost independent and contain separate results and discussions.

2. The Kholodenko Model with Diffusion

2.1 Introduction

In the year 2000, Boris N. Kholodenko [1] published a model of signal attenuation in protein kinase (MAPK) signaling pathways, which was shown to give rise to oscillatory behavior for a range of parameter values. An essential feature of the model is a negative feedback loop, meaning that the end product of the pathway inhibits the activating reaction (see Figure 2.1). The model predicted damped oscillations, since the suppression of oscillations was a rather unexpected effect which could have important implications for the biology of the cell. However, no oscillations have actually been observed for signaling pathways in real cells. There could be numerous reasons for this, for example that the abundance of proteins in measured as the total protein content in a large number of cells, which means that unphosphorylated cascades will not show up in measurements. But it is perhaps more probable that the model does not agree well enough with reality, meaning that the oscillations in MAPK pathways are artifacts which do not occur in nature. Specifically, the feedback loop in the model may be a too simple model of the feedback that is known to be present (since the signaling is turned off after a while), or it is also a fact that the signaling does not take place at one point in space but involves movement of proteins through the cell. Some of the proteins are transported from the cell membrane, while some may take place wherever the participating proteins encounter each other. This should have an effect on the oscillatory behavior, since the proteins at the end of the pathway move away from the membrane, they cannot take part in the feedback, which should then be attenuated. The purpose of the following sections is to incorporate diffusion of proteins into the model and study the effect that this has on the oscillations and the behavior of the pathway. Although the Kholodenko model may not be an accurate model, it is quite convenient to study as an oscillatory system, since the oscillations are easily seen and effects on them are easily detected.

2.2 The Kholodenko Model

The structure of the Kholodenko model for MAPK signaling pathways is shown in Figure 2.1. The pathway is activated by a stimulus of some kind, which causes the MEK to be phosphorylated. This activates the MEK, which in turn is able to phosphorylate the MAPK at two different sites. The doubly-phosphorylated MAPK (MAPKPP) then activates the MAPK by phosphorylation, again at two sites. In a real cell, the MAPKPP system is known to perform some action, which occurs or later will turn the signaling cascade off, by some means which are not quite understood (and may differ between different MAPK pathways). In the Kholodenko model, this is modeled by that the MAPKPP inhibits the phosphorylation of the MEK. This is the negative feedback loop, which is needed in order to create oscillatory behavior.
Figure 2.1: The Khaloghosko model, The signalling pathway is activated by some stimuli by which leads to phosphorylation of the MKKK, which in turn phosphorylate the MKK in two steps. The double-phosphorylated form of MKK then phosphorylates the MAPK, also in two steps. The end product (MAPKPP) then has the effect of inhibiting the stimulus, thus forming a negative feedback loop. Here, we also add the additional assumption that the MKKK is fixed at the cell membrane, while the other proteins are free to diffuse through the cytoplasm.

The corresponding ordinary differential equations may be written as

\[
\begin{align*}
\frac{d}{dt} w_j & = r_j \, v_j \\
\frac{d}{dt} u_j & = -r_j \, v_j + a_j + b_j \\
\frac{d}{dt} v_j & = -a_j \\
\frac{d}{dt} w_k & = -r_k \\
\frac{d}{dt} u_k & = -b_k + c_k \\
\frac{d}{dt} v_k & = -c_k \\
\frac{d}{dt} w_l & = -r_l
\end{align*}
\]

where the concentrations of MKKK through MAPKPP are denoted by \( w_j \), \( j = 1, \ldots, 8 \) with number as in the boxes in figure 2.1, and the fluxes of the reactions are denoted by \( r_j \), \( i = 1, \ldots, 10 \) and also numbered as in figure 2.1. The expressions for the fluxes are given in table 2.1.
Figure 2.8: Typical behavior of the Kholodenko model. The graph shows the concentration of MAPK (solid) and MAPKPP (dashed) for times up to 150 minutes. We see the sustained oscillations in concentration.

Laplacian. Here $d_4$ are the diffusion coefficients, which are now additional parameters in the model, describing how fast the diffusion of proteins is. These are known (see e.g. [10]) to be much lower inside the cell than in water (up to 10 times), because the cell is full of obstacles such as the cytoskeleton and other proteins. For simplicity, we assume that all the diffusion coefficients are equal, i.e., $d_4 = \ldots = d_9 = d$. For globular (non-spherical) proteins in water, the diffusion coefficient may be estimated from the relation

$$d \approx c W^{1/2},$$

(2.1)

where $W$ is the molecular weight [see [8]]. Fitting of measurement values for medium sized proteins simulated in [2, chapter 7], gives $c \approx 2.7 \times 10^{-12}$ if $W$ is measured in Da and $d$ in mm$^2$/s.

The MAPK and MAPKPP, which have a molecular weight of 46.8 kDa, gives $d \approx 74 \mu$m$^2$/s for fast diffusion in water, which we see as a reference value. The equations for the figures are the same as before, i.e., the ones given in Table 2.

The equations were solved using the Immersed Interface Method and finite differences on uniform grids with $34 \times 34 \times 34$ nodes. The method is described in Part II of this thesis.

2.4 RESULTS

We study the results of the model for two different values of the diffusion coefficient, $d$, namely $d_9 = d_1/10$ and $d_9 = d_1/1000$, with $d_9 = 74 \mu$m$^2$/s being the approximate diffusion coefficient in water for Hg$^2$.

The higher value $d_9$ for the diffusion coefficient is so large that the molecules have time to move around the entire cell faster than the reactions produce any significant changes in concentration. This means that the proteins will be evenly spread across the cell at all times. This is seen in figure 2.8 for the two compartments in the pathway (MKK and MAPKPP). The figure shows concentrations for times between 0 and 150 min on a line through the center of the cell. One can then that the concentrations is the same in the center of the cell at all times. One also see that the oscillations are not as large as for the original model. This is shown more clearly in figures 2.4 and 2.5, which should be compared to figure 2.3 for the original model. Here concentration for all compartments in the model are shown. Samples are taken at the membrane (figure 2.4) and at the center of the cell (figure 2.5). It is clear that the oscillations have a much smaller amplitude now that we have added diffusion. Furthermore, the oscillation seems to be damped, so that the amplitude decreases with time. This has been confirmed by running longer simulations, where the oscillations slowly fade away. The diffusion acts as a damper for the system.

A similar behavior is seen with the lower value $d = d_9$ in the diffusion coefficient (figures 2.6 and 2.7). The oscillations again have a lower amplitude and are again damped. Here, however, the proteins do not have time to diffuse through the cell before the reactions produce significant changes in phosphorylation levels. This is seen clearly in figure 2.8, where again concentrations of MKK and MAPKPP are shown along a line through the cell center for different times. It is clear that the concentrations are different in the center and at the membrane. However, the oscillations are present and approximately equal in period and amplitude in the center and at the membrane, and also approximately equal to the previous one (with $d = d_9$).

A final example is shown in figure 2.10. Here, we show the result of the same model with slow diffusion ($d = d_9$), but in a larger cell with a diameter of about 90 μm. Now the phosphorylated proteins do not have time to move very far from the membrane before they are dephosphorylated. Therefore the oscillations occur only near the membrane, while in the center all the kinases are in their inactive (dephosphorylated) state. This means that the signaling pathway unfolds, since its main purpose is to convey the signal of phosphorylations to the cell membrane, which it fails to do when distances are large. Diffusion slows down dephosphorylation reactions considerably fast.

It is clear that what determines the spatial behavior of the system is a combination of the length scale, the diffusion coefficient and the reaction rate. In order to see this more clearly, we may make the equation nondimensional. To illustrate this, we take the equation for $u_2 (M KKPP)$, but include only the dephosphorylation reaction with flow $\tau$. The equation then becomes

$$\frac{\partial u_2}{\partial t} = d \frac{\partial^2 u_2}{\partial x^2} - \frac{V_2 u_2}{K + u_2}.$$

If we introduce the nondimensional space variables ($\xi, \eta, \zeta = h (x, y, z)$, the nondimensional time $\tau = d/h^2$, $i$ and the nondimensional concentration $\nu = u_i / u_0$, with
Figure 2A: Concentrations for the eight components in the space-dependent model with fast diffusion $d = d_M$. The values are taken at the membrane.
2.4 Results

Figure 2.5: Concentrations for the eight components in the space-dependent model with fast diffusion $d = d_w$. The values are taken at the center of the cell.

Figure 2.6: Concentrations for the eight components in the space-dependent model with slow diffusion $d = d_s$. The values are taken at the membrane.
2.4 Results

Figure 24: Concentrations for the eight components in the space dependent model with slow diffusion $d = d_k$. The values are taken at the center of the cell.

Figure 25: Concentrations of M KK (left) and MAPKPP (right) in the model with fast diffusion $d = d_1$. The values are taken on a line through the center of the cell for times up to about 50 minutes. The concentrations are almost equal throughout the cell.

Figure 26: Concentrations of M KK (left) and MAPKPP (right) in the model with slow diffusion $d = d_k$. The values are taken on a line through the center of the cell for times up to about 50 minutes. The concentration gradient is clearly visible.
2.4 Results

2.5 Discussion

Much could be said about the Khalilinek model and whether oscillations really do occur in signaling pathways in real cells. One of the weakest points of the model is perhaps that the stimulus that activates the pathway is assumed to be "on" all the time, only being inhibited temporarly when the level of MAPKPP is high. This means that the cell never really responds to the signal, which is probably absurd, since the purpose of the signal is to cause the cell to respond and adapt to new conditions. Once the cell has adapted, the signal must cease, or the cell will probably die or at least start all its energy on growth tabs.

So, it is maybe not so probable that we will observe oscillations in signaling pathways in real cells.

However, this mathematical experiment has shown a number of other things of great interest, which are easy to observe because oscillations is an effect that is really marked. First, when we include the spatial distribution of protein, we see that the oscillations are severely damped compared to the original model, almost independent of the diffusion coefficients as long as it is not too small. This shows that by neglecting diffusion and spatial distribution when modeling, we may overlook important aspects and draw false conclusions about the behavior of the system. Of course, the principal behavior is in large determined by the space-independent reaction terms, but our simulations show that the amplitude of the effect may be diminished significantly by the addition of diffusion.

Furthermore, in our examples, the oscillations are not sustained, but seem to diminish with time, indicating that the system is damped by the diffusion. This is a different type of behavior than the sustained oscillations and is also an important thing to keep in mind when doing spatial independent modeling of biochemical processes. The reason for this dependence on diffusion is that the problem is already space dependent, since we know that the first reaction takes place only at the membrane, while the protein at the end of the chain may move about freely. Thus, purely time-dependent modeling cannot be expected to provide insight for all signaling pathways, since there are space-dependent features.

While for example examples of independent models may be examined to be more correct, since the space dependence is not so obvious in that case.

There is of course a reason for not using full spatial modeling, in particular three-dimensional modeling, since the solving of systems of PDEs take so much more time than solving a system of ODEs. The images shown here are results of simulations that took 10 minutes or more. This should be compared to functions of a second for solving the space-independent model. The amount of time needed to solve the space-independent equations makes it impossible to use for example parameter fitting algorithms, since such algorithms require a large number of simulations with different parameter settings. Therefore, space-independent ODE models are of great importance, but one should be aware that one is neglecting something and that it may be worthwhile to see what happens in a space-dependent model.
3. **Spatial Modeling of the HOG Pathway in Yeast**

3.1. The HOG Pathway

One well-studied MAPK signaling pathway is the HOG pathway in the yeast *Saccharomyces cerevisiae*, where HOG stands for High Osmolarity Glycerol and the MAP kinase Hog1 is the last enzyme in the signaling chain. The purpose of this enzyme is to sense the osmotic pressure on the cell membrane and produce the appropriate response (see [4]).

That is, if solutes (e.g., salt) are added to the solution outside the cell, the pump called osmotic will strive to level out the difference in solute concentration over the membrane by forcing water to flow out of the cell. This is potentially harmful for the cell, since it then starts to shrink and if that goes on, the cell can get function anymore and eventually it will die.

To avoid this and fate, the cell has to respond in some way to this new environment. It does so by adapting to produce glycerol and accumulating it inside the cell, which reduces the solute concentration and thereby prevents water from flowing out of the cell. And the link between the sensing of osmotic pressure and the response in the form of glycerol production is the HOG pathway. It is not entirely well known how the actual sensing of the change in osmotic pressure takes place, but there are enzymes at the cell membrane that act somehow unregulated, which leads to the activation of Hog1 through a few intermediate kinases. The pathway is shown in figure 3.1 with some of its surrounding components. When Hog1 has been activated, it enters the nucleus and once there it affects transcription of several genes through the transcription factor shown at the bottom of the figure.

A central feature of the signaling pathway is that Hog1 enters the nucleus. This can be visualized by genetically adding a Green Fluorescent Protein (GFP) tag to the Hog1 protein, which makes the molecule shine brightly when observed under ultraviolet light. This is a powerful method to view the localization of proteins in the cell and the results look like figure 3.1. One should be aware, however, that the GFP is a protein of about the same size as Hog1, so that the Hog1-GFP fusion is a much larger protein than wild type Hog1, which could affect the kinetics and function of the protein. Still, the Hog1-GFP fusion is functional in the sense that it is able to carry out its function in the signaling pathway.

3.2. The Model

Figure 3.2 shows the model that we are studying here. It is a model of the center part of the HOG pathway, where Hog1 itself is involved. The focus is on the transport of activated Hog1 into the nucleus. For simplicity, we assume that the MIT1K PKC zeta sits at the membrane in its phosphorylated form, ready to phosphorylate Hog1 molecules that come close to the membrane. This ensures that the upper part of the pathway that is not included in the model has reached an equilibrium and that phosphorylation of Hog1 can only take place at the membrane, which is believed to be true. There is also a small amount of spontaneous phosphorylation of Hog1 both in the nucleus and in the cytoplasm.

![Figure 3.1: A schematic diagram showing the expression of the HOG pathway, including the two different enzymatic mechanisms at the top phosphorylation on the right, cytochrome targets on the left and adenylate kinase (pre-cytochrome, in) at the bottom. Hog1 is near at the center of it all and it is indicated that when osmotic stress, it moves from the cytoplasm to the nucleus.](image-url)
Figure 32: Hsp1-GFP in wildtype yeast cells, under normal osmotic conditions on the left and after addition of NaCl on the right. The nuclei can be seen as bright spots in the right image, indicating that Hsp1 has entered the nucleus. The positions of the nuclei may be verified by staining with a special dye (DAPI) (not shown). The images are generated by Chen Hulin.

Figure 33: The model for the Hsp1 pathway. The MAPKK Ppk2 is assumed fixed at the membrane in its phosphorylated form, meaning that the pathway is constantly active. The phosphatases Pph2 and Pph3 are distributed evenly throughout the nucleus and cytoplasm respectively. Hsp1 is free to diffuse through the cell, but at the nuclear membrane, the transport is regulated. Unphosphorylated Hsp1 is transported in and out of the nucleus at equal rates, while phosphorylated Hsp1 (Hsp1p) is transported into the nucleus at a higher rate than it is transported out.

Figure 34: A cross section of the cell used in the model. The nucleus is shown as the bright spot.

The two phosphatases in the model, Pph2 and Pph3, are assumed to be evenly distributed in their respective domains, the nucleus and the cytoplasm. They are responsible for the dephosphorylation of Hsp1. The only differing component of the model are thus Hsp1 and Hsp1p, which are free to move in the cytoplasm and in the nucleus, but not between the two compartments. The transport through the nuclear membrane is regulated so that unphosphorylated Hsp1 is transported in and out of the nucleus at equal rates, phosphorylated Hsp1 (Hsp1p) is transported out of the nucleus at a much lower rate, but transported into the nucleus at a much higher rate. The geometry used for the model is a simplified three-dimensional cell with an outer spherical nucleus and a slightly ellipsoidal cell membrane. No other internal cell structures are included. A cross section of the model cell is shown in figure 34.

The model is of course a very simplified one. We disregard the fact that the phosphorylation of Hsp1 takes place in two steps, so that the active form is double phosphorylated. We also ignore the upper part of the pathway, the unphosphorylated Ppp1p and the phosphorylation of Ppp2. Also, the targets of Hsp1 are not included in the model and there is no feedback loop to turn off the signaling pathway. The transport of protein through the nuclear membrane is also simplified. Here, we use the assumption that the flux through the membrane is proportional to the concentration of protein, which is of course a simplification of the actual complex transport system that shuttles the protein in and out of the nucleus. And finally, the inner structure of the cell is not included other than in a reduction of the diffusion constant because of the obstacles.
3.2 The model

We denote the concentrations of unphosphorylated and phosphorylated Hsp70 by \( u \) and \( u_p \), respectively, the nucleus by \( N \), the cytoplasm by \( C \), and the inner and outer membranes \( M \) and \( P \). The model then leads to the following partial differential equations

\[
\begin{align*}
\partial u / \partial t &= d_u \partial^2 u / \partial x^2 + f(x, u, u_p), \quad \text{in} \quad C, \\
\partial u / \partial t &= d_u \partial^2 u / \partial x^2 + f(x, u, u_p), \quad \text{in} \quad N, \\
\partial u / \partial t &= d_p \partial^2 u / \partial x^2 + f(x, u, u_p), \quad \text{in} \quad P, \\
\partial u / \partial t &= d_u \partial^2 u / \partial x^2 + f(x, u, u_p), \quad \text{in} \quad M, \\
\partial u / \partial t &= 0, \quad \text{on} \quad \Gamma_C, \\
\partial u / \partial t &= 0, \quad \text{on} \quad \Gamma_N, \\
\partial u / \partial t &= a_u u_p [u]^{\gamma}, \quad \text{on} \quad \Gamma_P, \\
\partial u / \partial t &= a_p u_p [u]^{\gamma}, \quad \text{on} \quad \Gamma_P.
\end{align*}
\]

with

\[
\begin{align*}
f(x, u, u_p) &= \frac{V_{P_{PH}}u_p}{K_{P_{PH}} + u_p} - \frac{V_{P_{PH}}u}{K_{P_{PH}} + u} + h_{P_{PH}}u, \\
f(x, u, u_p) &= \frac{V_{P_{PH}}u_p}{K_{P_{PH}} + u_p} - \frac{V_{P_{PH}}u}{K_{P_{PH}} + u} + h_{P_{PH}}u.
\end{align*}
\]

Here, \( d_u \) and \( d_p \) are diffusion constants, which we assume to be equal. The first two equations are diffusion equations with one diffusion term and one reaction term, the two first describe the cytoplasm and the two last the nucleus. The reaction terms make use of Michaelis-Menten kinetics (see eq. 2.1, chapter 4) for the enzyme reactions, which is a standard way of modeling such reactions. Next follow boundary conditions at the plasma membrane, which say that the flux through the membrane is zero, i.e., that no precursor can leak out of the cell. The two last rows are boundary conditions at the nuclear membrane, both saying that the flux out through the membrane equals a constant times the nuclear concentration at the membrane minus a constant times the cytoplasmic concentration on the outside of the membrane. We also need initial conditions that describe the concentration at time \( t = 0 \). These are shown in figure 3.3, and are chosen so that the system in near its equilibrium, which is the unphosphorylated state and is rapidly distributed between nucleus and cytoplasm. A small fraction of the molecules are phosphorylated, and these have a higher concentration in the nucleus than in the cytoplasm.

<table>
<thead>
<tr>
<th>( \lambda )</th>
<th>( 0.0002 ) s(^{-1} )</th>
<th>( \lambda_1 )</th>
<th>0.0002 s(^{-1} )</th>
</tr>
</thead>
</table>

Table 3.1: Parameter values

The equations were solved using the immersed interface method and finite differences on uniform three-dimensional grids with 54 × 54 × 54 nodes. The method is described in section 2.1 of this thesis. The final simulations shown here, involving some 200 time steps, took up to 30 minutes to complete on a standard computer.

3.3 Results

Figure 3.3 shows the initial conditions used in the 3D simulations. The images only show the concentration in a slice through the center of the cell, but the concentration is assumed to be uniform throughout the nucleus and the cytoplasm. The values shown are near the equilibrium of the system when the pathway is not activated. The exact values were not 1.00 s\(^{-1} \) for Hsp70 and 1.50 s\(^{-1} \) for Hsp70 in cytoplasm and nucleus, respectively, but changing these values slightly does not alter the behavior of the system much, so the exact values are not very important.

Then at time \( t = 0 \), the signal is turned on, in this model by activating P38, that is setting \( \lambda_1 \) to a non-zero value. In figure 3.3, we see the simulated response as it would appear in the fluorescence microscope: namely the total concentration of Hsp70 (Hsp70 + Hsp70\(^p \)) at the start and the end of the simulation. The simulation ends at 22 minutes when the concentration has approached the steady-state levels with the signal turned on, in a real cell the level of phosphorhilation would again be down when the cell started to die, but since this adaptation is not included in the model, the concentrations just approach a steady state with high concentration in the nucleus, steadily decreasing phosphorylated Hsp70 and lower concentration outside.

To see the time evolution of the system, we show in figure 3.7 concentration along a line through the center of the cell for all times and for Hsp70 and Hsp70\(^p \). We see that the levels of Hsp70 are rapidly increasing, and that Hsp70\(^p \) is transported into the nucleus, while unphosphorylated Hsp70 is moved out of the nucleus as the levels become lower in the cytoplasm when Hsp70 is phosphorylated. One also notes that there are concentration gradients in the cytoplasm, arising from the fact that the diffusion is not fast enough to even out the concentrations at the same rate that the reactions change them. The dimensionless parameter \( \lambda \) defined in section 2.4 is here

\[
\lambda = \frac{\lambda_1}{\lambda_2} \frac{V_{P_{PH}}u_p}{K_{P_{PH}} + u_p} \left( \frac{10\text{nm}}{1\text{nm}} \right)^2, 1.0 \text{ s}^{-1}, 10^3 \text{ s}^{-1} \Rightarrow 13\lambda,
\]
3.3 Results

Figure 3.1: Initial conditions for the simulation in 3D, with unphosphorylated Hog1 on the left and phosphorylated Hog1 (Hog1p) on the right. The images show a slice through the center of the cell, with the nucleus visible in the right image. Note the different scales on the two images.

Figure 3.2: Concentration of Hog1 (left) and Hog1p (right) for slow diffusion. The values are taken along a line through the center of the cell and the nucleus. The medium contains no Hog1p initially. There is a slight concentration gradient in the cytoplasm.

Figure 3.3: The total Hog1 concentration ([Hog1] + [Hog1p]) at the start (left) and end (right) of simulation. These images are analogous to the scan seen in the fluorescent microscopy. The imaged edges are artifacts arising from the stage digitization.

Figure 3.4: Concentration of Hog1 (left) and Hog1p (right) for slow diffusion when the signal is turned off after Hog1p has accumulated in the nucleus. The concentrations return to their original state.
indicating that reactions are slightly faster than diffusion, which should produce gradients in the concentration levels.

In figure 13, we also show the return to the original state when the signaling pathway is turned off. Starting at the final level of the previous simulation, and setting $V_m = 0$, we see that the system returns to its original state with low levels of phosphorylated Hg and almost equal concentrations in nucleus and cytoplasm.

Finally, we also show a simulation with a larger diffusion coefficient $d_1 = d_2 = d_3/10 \approx 74 \mu m^2/s$. The results are shown in figure 10. Now, diffusion is fast enough to level out the concentrations, so that they are uniform in nucleus and cytoplasm, respectively. The response in the nucleus is also slightly faster than in the slow diffusion simulation.

3.4 Discussion

The results shown above show that it is possible to reproduce in simulations the nuclear relocation of Hg that is observed when yeast cells are exposed to stress. The model used is a very simple one, including only two diffusing components and only the first step of the phosphorylation chain. The results should therefore not be used to draw conclusions about the actual behavior of the cell.

What we see, however, is that only with the small diffusion coefficient $d_1/1000$ do we observe distinct gradients and interesting spatial effects. This value for the diffusion coefficient is much smaller than the values observed for similar proteins inside living cells (up to $10^7$fold reduction compared to diffusion in water [10]). For the larger diffusion coefficient $d_1/10$, which is more in agreement with observed values, concentration are partially mixed in both of the compartments. This indicates that diffusion is fast compared to the motions involved, so that full spatial modeling with PDEs might not be of crucial importance for the simple model with the parameter values used here. However, the behavior of the system is very much dependent on the enzyme rates, especially for the phosphatases Pph and Pph. The values for the maximal rates used here are not based on direct measurements but are taken to be near the values in the Drosophila model, which are based on in vitro measurements for MAPK in mammalian cells. These values may not be correct for the proteins in our model. So, we can not determine from our simple model whether the explicit modelling of spatial effects shown here is of crucial importance. In any case, one must remember that cell signaling is a complex phenomenon so that spatial features should be taken into account, in one way or another. What we have developed is a method to simulate full spatial models that could be used to compare the output with results from simpler, less computationally intense methods.

In the end, what decides whether a biological model is good or bad, is if it compares well with measurements of the end phenomenon in live cells. So, the results about what type of models are needed, should compare the results to particular measurements. This, however, is not an easy task. The types of experiments that need to be done to determine spatial distributions of proteins inside the cell are fluorescence microscopy experiments, yielding images such as those in figure 10, in order to compare them to the results of the simulations made here. One would like to follow a single cell and take images at intervals of seconds or less, which is very hard to do. Furthermore, one would ideally like to be able to view phosphorylated and unphosphorylated Hg separately, which is not possible at present.

One may also discuss if a diffusion PDE is an appropriate model for the transport of protein inside the cell. First of all, the number of molecules is relatively small. The estimated number of Hg molecules in the yeast cell is about $10^5-10^6$, giving a concentration of around 50 nM. Since the diffusion equation is obtained in the limit when the number of molecules tend to infinity, it may not be an accurate model. The random fluctuations arising from the small number of molecules may also influence the outcome. However, as long as there is no data to compare to, it is hard to decide between models. But it would be interesting to investigate the differences between a PDE model and a stochastic model for the signaling pathway.

Another issue related to diffusion is that the cell is full of obstacles, consisting of organelles, cytoskeleton network, large protein complexes etc. This makes the assumptions underlying the diffusion model invalid. It has been shown here [11, 12, 10] that these obstacles do not only cause reactions in the diffusion coefficients, but also give rise to the phenomenon called anomalous (sub)diffusion. Per anomalous diffusion, the mean square displacement of the molecule is no longer proportional to time $(z(t)^2 \propto t)$, but to some power $\alpha < 1$ of time $(z(t)^2 \propto t^\alpha)$. This may give rise to new interesting phenomena and is also a subject that would be interesting to study.
THE IMMERSED INTERFACE METHOD ON
UNIFORM AND BOOLEAN GRIDS

Tobias Gebäck

Abstract

The Immersed Interface Method (IIM) is a method which allows the use of
finite differences in rectangular domains by immersing interfaces into
a rectangular domain which is discretized by a uniform grid. The finite
differences near the interface are then corrected using the size and position of the
jumps in the solution and its derivatives across the interface.

The method presented here is the Explicit Jump IIM developed by Wieg
maier and Rude, but we also present some additional details on how to apply
the method in three-dimensions using Robin boundary conditions and to time
dependent problems.

We then apply the IIM to Boolean grids. There are grids that are com-
binations of a number of Cartesian grids to achieve greater accuracy of ap-
proximation, while using fewer grid nodes. The use of IIM on such grids
require some new development of the estimation of jump on the bound grid.
We show numerically that the resulting Boolean IIM grid second order error
converges with respect to the smallest step size in the grid, meaning that the
required number of nodes needed for a given maximal error is considerably
smaller than on uniform Cartesian grids.

Contents

1 The Immersed Interface Method
  1.1 Introduction ................................................. 1
  1.2 IIM theory ............................................... 1
  1.3 Estimating jumps at boundaries .......................... 4
  1.4 Applying boundary conditions ............................ 6
  1.5 Solving the linear system ............................... 9
  1.6 Other equations ....................................... 10
  1.7 Examples ............................................. 10

2 The Immersed Interface Method on Boolean grids
  2.1 Introduction ........................................... 13
  2.2 Boolean interpolation .................................. 13
  2.3 The algebraic theory .................................. 15
  2.3.1 Boolean grids ..................................... 15
  2.3.2 Application to finite difference solvers .......... 21
  2.4 Applying IIM to Boolean grids ........................ 22
  2.4.1 Boolean approximation of jumps .................. 23
  2.4.2 Examples ......................................... 27
  2.4.3 Example 1 Boolean approximation ................. 27
  2.4.4 Example 2 Finite differences on Boolean grids ... 28
  2.4.5 Example 3 IIM on Boolean grids .................. 31

References .................................................. 34
1. The Immersed Interface Method

1.1. Introduction

The use of finite difference methods for solving partial differential equations has a few advantages, mainly that they are easy to implement and that they may be easily and quickly solved using Fast Fourier Transform (FFT) methods. The drawbacks are that these methods demand a rectangular domain with a uniform grid and also that they are not straightforward to obtain accuracy estimates and convergence results.

The immersed interface method overcomes the first of these problems, in that it allows the solution and its derivatives to be discontinuous along the faces. This makes it possible to immerse a boundary into a rectangular grid. Let the solution be zero outside the boundary and apply boundary conditions to the solution along the boundary. The finite differences near the boundary are then corrected using the jumps in the solution so that the differences remain valid even though the solution is discontinuous. Therefore, the method makes it unnecessary to spend time on making grids adapted to the geometry and also allows the use of FFT based methods even though the domain is not rectangular.

The idea behind the immersed interface method was first conceived by Peskin [13] and used in computations on bent flows, with moving boundaries. The method was further developed by LeVeque and Li [11], [19] and used with finite differences on Cartesian grids. Finally, Wiegmann and Bube [16], [12] gave the method a clear formulation and extended it to more general problems. They called their method the "immersed jump interface method" or EJIM, and that is the method we will be concerned with here.

We will first describe the general idea behind the EJIM and then proceed to the problem of implementing the method. This will include treatment of three-dimensional problems and time-dependent problems, which is not included in the original article. We will also discuss some added boundary conditions which were not discussed by Wiegmann and Bube.

This report is intended as a review of results on the immersed interface method and contains no proofs of convergence or error estimates. Wiegmann and Bube provide proofs of convergence for the method in one dimension and allow for two dimensions for the special case when the jump at the boundary is known beforehand. Other proofs or error estimates are not known and are only confirmed numerically.

1.2. EJIM Theory

The basic idea of the IIM is that standard finite differences, such as

\[ u(x_h) = \frac{u(x + h) + u(x - h)}{2} + O(h^2) \]

are not valid for nonsmooth functions, since they are based on Taylor expansions. However, they may be corrected using the size and position of the discontinuities in \( u \) and its derivatives, Let us denote the jump in the \( m \)-th derivative in \( u : \mathbb{R} \rightarrow \mathbb{R} \) at a point \( \alpha \in \mathbb{R} \)

by

\[ [u^{(m)}]_\alpha = \lim_{x \to \alpha} u^{(m)}(x) - \lim_{x \to \alpha^+} u^{(m)}(x), \]

where, of course, \( u^+ = u \).

Following [16], we give two lemmas that contain the essence of the EJIM. The proofs are essentially exercises in the use of Taylor expansions and we refer the reader to the original article [16].

**Lemma 1.1** (cf. [16, lemma 1]) Let \( \alpha > 0 \) and \( u \in C^2[\alpha, \alpha + \delta] \) and \( u^+ \in C^2(\alpha, \alpha + \delta) \). Then let

\[ u(x) = \begin{cases} 
\frac{1}{2} \left( u^+ + u^- \right) & \text{for } x < \alpha, \\
\frac{1}{2} \left( u^+ + u^- \right) - \frac{1}{2} \alpha & \text{for } x = \alpha, \\
u^+ & \text{for } x > \alpha,
\end{cases} \]

For \( x \in (\alpha, \alpha + \delta) \), we then have:

\[ u(x + h) = \frac{1}{2} \left( u^+ + u^- \right) + \frac{1}{2} \alpha + \frac{1}{2} \left( \frac{\partial u^+}{\partial x} + \frac{\partial u^-}{\partial x} \right) h + O(h^2), \]

and

\[ u(x) = \frac{1}{2} \left( u^+ + u^- \right) + \frac{1}{2} \alpha + \frac{1}{2} \left( \frac{\partial u^+}{\partial x} + \frac{\partial u^-}{\partial x} \right) h + O(h^2). \]

Using this lemma in the expressions for finite differences on a uniform 1d grid \( \{x_i\} \) with grid spacing \( \delta \), we get the next lemma.

**Lemma 1.2** (cf. [16, lemma 3]) Let \( \alpha < x_j \) and \( u \in C^2(\alpha, \alpha + \delta) \). Then the following approximations hold:

\[ u_j = \frac{1}{2} \left( u_{j+1} + u_{j-1} \right) - \frac{1}{2} \frac{\partial u}{\partial x} \left( \frac{1}{2} \right)^m \left[ \frac{\partial u}{\partial x} \right]_{j-1} + O(h^2), \]

\[ u_{j+1} = \frac{1}{2} \left( u_{j+2} + u_{j} \right) - \frac{1}{2} \frac{\partial u}{\partial x} \left( \frac{1}{2} \right)^m \left[ \frac{\partial u}{\partial x} \right]_{j+1} + O(h^2), \]

\[ u_{j-1} = \frac{1}{2} \left( u_{j} + u_{j-2} \right) - \frac{1}{2} \frac{\partial u}{\partial x} \left( \frac{1}{2} \right)^m \left[ \frac{\partial u}{\partial x} \right]_{j-1} + O(h^2), \]

\[ u_{j+2} = \frac{1}{2} \left( u_{j+3} + u_{j+1} \right) + \frac{1}{2} \frac{\partial u}{\partial x} \left( \frac{1}{2} \right)^m \left[ \frac{\partial u}{\partial x} \right]_{j+1} + O(h^2), \]

\[ u_{j-2} = \frac{1}{2} \left( u_{j} + u_{j-4} \right) + \frac{1}{2} \frac{\partial u}{\partial x} \left( \frac{1}{2} \right)^m \left[ \frac{\partial u}{\partial x} \right]_{j-1} + O(h^2). \]
Figure 1.1: Immersed boundary in 2D. The immersed boundary intersects the grid at points \( a_0 = (x_{a0}, y_{a0}) \) and \( b_0 = (x_{b0}, y_{b0}) \). Interface interaction points (IIPs) are marked by \( \cdot \) and anchor points are marked by \( * \).

The application of these results to higher dimensions is now straightforward. In the situation in figure 1.1, for example, the Laplacian of \( u(x,y) \) would be approximated as

\[
\Delta u(x,y) \approx \frac{u(x_{a0},y_{a0}) + u(x_{b0},y_{b0})}{2} + \frac{1}{h^2} \sum_{m=0}^{2} \sum_{n=0}^{2} (n_a u_{m,n}(a_0) + n_b u_{m,n}(b_0)) \Delta_0^2 u(x,y)
\]

for \( x \in \partial \Omega \), that is Dirichlet, Neumann or Robin boundary conditions.

In order to apply the EJIM, we first need to embed the domain \( \Omega \) into a rectangular area discretized by a uniform grid and let \( \Omega^- \) denote the domain outside the interface \( \partial \Omega^- \). Setting \( f(x) = 0 \) in \( \Omega \) and \( u = 0 \) on \( \partial \Omega \), the above boundary conditions become the regular Dirichlet, Neumann and Robin boundary conditions for \( u \) in \( \Omega^\pm \). The corresponding linear system of equations becomes

\[
\Delta u = \Phi C - F,
\]

\[
\sum_{i=1}^{n} C_{ij} F_i = \sum_{i=1}^{n} C_{ij} F_i - \sum_{i=1}^{n} C_{ij} C_{ij} F_i.
\]

The problem here is that the jumps \( C \) are unknown, so we need to find an additional relation that specifies these jumps. Some of the jumps can be determined by the boundary conditions at \( \partial \Omega \), while others must be determined in another way. This is done by creating an interpolation matrix \( D \), which gives the function values, estimates the jumps at the boundary. Then the global equation system may be written

\[
\Delta u + \Phi C = F_0,
\]

\[
C = D \Phi C + F_1,
\]

where \( F_0 \) contains function values of the grid nodes on one side of the boundary and \( C \) contains the known jumps derived from boundary conditions.

1.3. Estimating jumps at boundaries

The jumps at the boundary are estimated by Lagrange interpolation, meaning that for each interface interaction point (IIP), \( i \), for each point where the interface intersects the grid (see figure 1.1), we select a number of grid nodes on one side of the boundary and calculate the interpolating polynomial of degree \( s \) at \( y \). This polynomial and its derivatives
are then evaluated at the boundary. Doing the same from the other side of the interface, we may take the difference and get an estimate of the jumps at the IIP given function values at the grid nodes. The number of grid nodes needed to construct a polynomial of degree \( d \) in \( n \) dimensions is \( \binom{n+d}{d} \).

As an example, let us consider a second order polynomial in two dimensions \((i, d = n-2)\). We select \( \binom{2}{2} = 6 \) grid nodes \( n_i = (x_i, y_i), i = 1, \ldots, 6 \) with corresponding function values \( u_1, \ldots, u_6 \) and set up the equation system

\[
\begin{align*}
1 & \quad 1 & \quad 1 & \quad 1 & \quad 1 & \quad 1 & & \quad u_1 \\
1 & \quad x_1 & \quad x_1^2 & \quad x_1 y_1 & \quad x_1 y_2 & \quad x_1 y_3 & & \quad u_2 \\
1 & \quad x_2 & \quad x_2^2 & \quad x_2 y_1 & \quad x_2 y_2 & \quad x_2 y_3 & & \quad u_3 \\
1 & \quad x_3 & \quad x_3^2 & \quad x_3 y_1 & \quad x_3 y_2 & \quad x_3 y_3 & & \quad u_4 \\
1 & \quad x_4 & \quad x_4^2 & \quad x_4 y_1 & \quad x_4 y_2 & \quad x_4 y_3 & & \quad u_5 \\
1 & \quad x_5 & \quad x_5^2 & \quad x_5 y_1 & \quad x_5 y_2 & \quad x_5 y_3 & & \quad u_6 \\
\end{align*}
\]

in order to construct the polynomial

\[
P_2(x, y) = a_{01} + a_{12} x + a_{22} x^2 + a_{11} y + a_{21} y^2.
\]

We wish to solve this system for the coefficients \( a_{ij} \), which leads to the requirement that the determinant of the \( 6 \times 6 \) matrix (the so-called Vandermonde determinant) must be non-zero. This in turn leads to restrictions on the choice of the grid nodes \( n_i \). These restrictions were studied by Chai and Lai [4] for arbitrary \( n \) and \( d \). They give a sufficient criterion for restriction on both \( n \) so that the Vandermonde determinant is non-zero, which they call Note Configuration A.

**Definition 1.1 (Node Configuration A in \( \mathbb{R} \))** Any set of distinct points in \( \mathbb{R} \) satisfies Note Configuration A in \( \mathbb{R} \).

**Definition 1.2 (Node Configuration A in \( \mathbb{R}^n \))** Let \( \hat{S} = \{x_1, \ldots, x_N\} \) be a set of \( N \) distinct points in \( \mathbb{R}^n \), \( \hat{S} \) satisfies Node Configuration A in \( \mathbb{R}^n \) if there exist \( d + 1 \) hyperplanes \( K_i, i = 0, \ldots, d \), in \( \mathbb{R}^n \) with

\[
x_{N+1}, \ldots, x_N \in K_i \setminus (K_{i+1} \cup \ldots \cup K_d)
\]

for \( j = 0, \ldots, d - 1 \), and such that each set of points

\[
\hat{S}_j = \{x_{N+1}, \ldots, x_N\}, \quad 0 \leq j \leq d,
\]

viewed as points in \( \mathbb{R}^{n-1} \), satisfies Node Configuration A in \( \mathbb{R}^{n-1} \).

**Theorem 1.1 (cf. [a theorem d])** If \( \hat{S}_j \), with \( n \geq 1 \) and \( d \geq 0 \), satisfies Node Configuration A, then the corresponding Vandermonde determinant is non-zero.

---

### 6 The Immersed Interface Method on uniform and Boolean grids

**Figure 1.2** Examples of node configurations in two dimensions. The two on the left satisfy Note Configuration A, while the one on the right does not.

In the definition we use the convention \( N_{-d} = 0 \) for all \( n \). Also note that \( (N_{-d} - 1) \) \( (N_{-d}) \), \( N_{-d} - 1 \), so that the (\( n = 1 \)) dimensional hyperplane \( K_i \) contains \( N_{-d} - 1 \) points, which by an orthogonal transformation may be regarded as points in \( \mathbb{R}^{n-1} \) for which the node configuration problem could again be posed. Therefore Note Configuration A is well defined.

The recursive definition of the node configuration may seem hard to interpret in all its generality, so let us see what it means in our above example with \( n = 2 \) and \( d = 1 \). In that case, the hyperplane becomes line, and the definition tells us that we should be able to draw \( d + 1 = 3 \) lines, such that \( N_1 = \binom{1+1}{1} - 3 \) points lie on the first one, \( N_1 = \binom{1+1}{1} - 2 \) on the second (but not on the first) and \( N_1 = \binom{1+1}{1} - 1 \) on the third. Furthermore the set of points on each of these lines should satisfy Note Configuration A, which they do automatically, since they are on one-dimensional sets. Two valid node configurations are one given in figure 1.1.

In these dimensions, the hyperplanes become planes, and for a second order polynomial, we need to be able to choose 3 planes, containing 6, 3 and 1 nodes respectively making \( \binom{2+1}{1} - 1 \) nodes in total. And on each of the planes the nodes should satisfy Note Configuration A in two-dimensions.

### 1.4 Applying Boundary Conditions

Having seen how to create interpolating polynomials, we return to the problem of estimating the jumps in the vector \( C \) in (1.10), i.e., we need to determine the matrix \( D^T \) and the vector \( F_3 \) in the equation

\[
C = D^T U + F_1.
\]

Here, we do this in three dimensions, since the two-dimensional problem is treated in [10], while the three-dimensional problems are not treated. In order to deduce \( D^T \), we first find all the interface intersection points (IBP's). For each of these points, there is a coordinate direction \( X_j = x, y \), or \( z \), which is the direction of the grid line on which the
1.4 Applying boundary conditions

The immersed interface method requires that the solution be continuous and smooth across the interface. This is achieved by approximating the jump conditions for the solution at the interface. The jump conditions are defined by the difference in the values of the solution across the boundary. For the 2D case, we consider the following system of equations:

\[
\begin{align*}
\mathbf{u}_h^+ & = \mathbf{u}_h^- + \mathbf{J} \\
\mathbf{u}_h^+ & = \mathbf{u}_h^- + \mathbf{J}
\end{align*}
\]

where \( \mathbf{u}_h^+ \) and \( \mathbf{u}_h^- \) are the solution values on either side of the interface, and \( \mathbf{J} \) is the jump operator. The jump operator is defined as

\[
\mathbf{J} = \begin{bmatrix}
0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0
\end{bmatrix}
\]

In practice, this is approximated using finite differences. For example, in the case of a 2D problem, we might approximate the jump as

\[
\mathbf{J} \approx \begin{bmatrix}
0 & 0 & \cdots & 0 \\
0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & 0
\end{bmatrix}
\]

This approximation is valid when the solution is smooth across the interface and the grid is fine enough to resolve the jump accurately.

### Example

Consider a simple 2D problem with a discontinuous solution across a line \( \Gamma \). The solution \( \mathbf{u} \) is given by

\[
\mathbf{u}(x, y) = \begin{cases}
\mathbf{u}_1 & \text{if } (x, y) \in \Gamma \\
\mathbf{u}_2 & \text{otherwise}
\end{cases}
\]

To approximate the jump, we use a finite difference scheme, for example:

\[
\nabla \mathbf{u} \approx \begin{bmatrix}
\frac{\mathbf{u}_1 - \mathbf{u}_2}{h} \\
\frac{\mathbf{u}_1 - \mathbf{u}_2}{h}
\end{bmatrix}
\]

where \( h \) is the grid spacing.

### Conclusion

The immersed interface method provides a robust framework for solving PDEs in domains with complex interfaces. It is particularly useful for problems in computational fluid dynamics, material science, and other fields where interfaces play a critical role.
where the first term is known and therefore included in $F_1$, while the coefficients of the second term are included in $L_1$, so that we get (1.14) once more. The expression for $b_{ij}$ and $w_{ij}$ are derived similarly while $w_{ij} = w'(x_i) = u_{i,j}$ and $|x_{ij}| = |x_{i,j}|$ for $i = 1, 2, 3$.

Finally, we consider Robin boundary conditions, i.e., $\partial u/\partial n = au - bu$, using the above results this leads to the expression

$$u^2 - u - au - bu = a u (u, a)$$

which gives us

$$w_{ij} = -\frac{1}{a} w'(x_i) + \left( -\frac{1}{a} \right) u_{i,j}.$$

Furthermore, we impose the implicit condition that $u^2 - u$, which we impose by solving $b(x_1) = 0$ in the Neumann case above. In summary, this leads to the above expression for $w_{ij}$, expressions similar to the Neumann case (but with $F_{i,j} = 0$) for $w_{ij}$ for the same equation as before for $w_{ij}$.

This way, we construct the sparsity matrix $D^T$ and the vector $F_i$ by studying the co-ri breaks from each IH, as we have multiple IHs with different boundary conditions, we just include the IHs from all the IHs and use the corresponding boundary condition for each IH according to the above treatment.

Finally, we need to construct the sparse matrix $\Psi$ in (1.10). This matrix contains the coefficients of the jumps in (1.10) to (1.12), placed at appropriate positions, so that the corrections affect the corresponding matrix points. Since the corrections are additive, the $\Psi$-matrix is easy to construct.

We also remark that it is possible to apply other types of boundary conditions, such as in composite material problems illustrated in [36], by the same method as shown here.

1.5 Solving the Linear System

Through the previous sections, we have arrived at the system of equations

$$\Delta_{i} U + \Psi C = F_{i},$$

$$C = D^TU + F_{i},$$

which is (1.10). Solving for $U$ in the first equation and inserting this into the second, we get

$$U = \Delta_{i}^{-1}(\Psi C + F_{i}),$$

$$(1 + D^T \Delta_{i}^{-1} \Psi) C = D^TU + F_{i}. $$

Here, the second equation is a small system of equation for $C$ with $3N_{H^2}$ rows (where $N_{H^2}$ is the number of IHs). It may be solved using iterative methods to avoid forming the matrix on the left explicitly, which would be unfeasibly memory consuming for three-dimensional problems, since the matrix is not sparse. We use the BCGSAC method (see e.g., [41, 42] to solve the system and since we then only need to be able to form the matrix vector product $(I + D^T \Delta_{i}^{-1} \Psi)C$, our only remaining problem is to apply $\Delta_{i}^{-1}$. Since our domain is embedded in a regular or uniform grid, we may use the fast Fourier Transform to achieve this efficiently (see e.g., [34, 46]), if $N$ is the total number of grid nodes, the FFT is applied in $O(N \log N)$ operations and this is the most time consuming part of each iteration in the BCGSAC algorithm. We have used the library FFTW [1] to get a fast implementation of the FFT in $C$.

Having solved for $C$, all that remains is to compute $U$ from the first equation by applying $\Delta_{i}^{-1}$ once more.

1.6 Other Equations

The immersed interface method may also be applied to other types of equations. First of all, it is straightforward to apply it to the heat equation

$$u(x,t) = \Delta u(x,t) = f(x), \quad x \in \Omega, \quad t \geq 0,$$

with Neumann boundary conditions, using for example an implicit Euler scheme in time. In that case we get

$$\frac{1}{\Delta t} u_{i,j} = \Delta u(x,t) = f(x), \quad x \in \Omega, \quad t \geq 0,$$

where the operator $(1/\Delta t) u_{i,j}$ may be applied using FFT and the second equation solved using BCGSAC for each time step.

It is also possible to solve equations in multiple domains with boundary conditions between the domains or with different coefficients in the equation in different domains. The technique is exactly as described in the previous sections. See sections 1.7 and 10 for example.

Other combinations of spatial derivative than the Laplacian may also be considered, although they are not discussed here. It should be clear however, from the discussion above how the immersed interface method should be applied to these cases.

It is also in principle possible to use the method for moving boundary problems, where one would like to avoid costly grid generation. It would then however, be necessary to compute the matrices $D^T$ and $\Psi$ at every time step.

1.7 Examples

Our first example is solving the Poisson equation inside an ellipsoid $E$ with halfaxes $a, b, c$ centered at $(0,0,0,0,0)$. The equation we solve is given in spherical
1.7 Examples

Figure 13: EIJIM for the Poisson equation in an ellipsoid in 3D. The solution was computed on a 40 × 40 × 40 grid and is shown on the left. Errors compared the exact solution are shown on the right. Values are shown on selected slices through the domain.

\[ \Delta u(x, y, z) = f(x, y, z), \quad (x, y, z) \in E, \]
\[ u(x, y, z) = g(x, y, z), \quad (x, y, z) \in \partial E, \]

where \( f(x, y, z) = 2\sin(3\pi z) \) and \( g(x, y, z) = x^2 y^2 \sin(3\pi z) \), which gives the solution \( u(x, y, z) = x^2 y^2 \sin(3\pi z) \) in \( E \). The solution and the error are shown in figure 13.a. We see that the non-trivial problem can be solved with reasonable accuracy even on the rather coarse 40 × 40 × 40 grid used here. The solution took less than a second to compute on a standard computer. The error is of order \( O(h^2) \), which is shown more clearly for another example in section 2.3.

The second example treats more complicated equations and boundary conditions may be solved using the immersed interface method. We solve the heat equation in two dimensions. The domain consists of two concentric circular discs, \( C_1 \) and \( C_2 \), with \( C_1 \) inside \( C_2 \). We apply Neumann boundary conditions at \( \partial C_1 \) and Robin boundary conditions at \( \partial C_2 \). The problem may be written as

\[ \frac{\partial u}{\partial t}(x, y, t) \Delta u(x, y, t) = 0, \quad (x, y) \in C_1 \cup C_2, \quad t \geq 0, \]
\[ \frac{\partial u}{\partial n}(x, y, t) = 0, \quad (x, y) \in \partial C_2, \quad t \geq 0, \]
\[ \frac{\partial u}{\partial n}(x, y, t) = 10u(x, y), \quad (x, y) \in \partial C_1, \quad t \geq 0, \]
\[ u(x, y, 0) = u_0(x, y), \quad (x, y) \in C_1 \cup C_2, \]

where \( u_0(x, y) \) is a bell-shaped function inside \( C_1 \) shown at the top of figure 14.a, \( u_1 \) and \( u_2 \) are the values of \( u \) at \( \partial C_1 \), on the outer and inner side of the boundary.

Figure 14: EIJIM for the heat equation inside two concentric discs. The initial data is shown at the top and the solution at \( t = 0.05 \) below. The domain boundaries are only visible in the second figure.

The method used is the one described in section 1.6, with implicit Euler time stepping. The boundary conditions are implemented as described earlier.

Obviously, the exact solution is not known for this problem, so we have nothing to compare to. However, the example shows that it is possible to solve multiple boundary time dependent problems with a range of boundary conditions using the immersed interface method. Here, for simplicity, we have used only circular and elliptical domains but in the method itself there are no limitations on the shape of the domain. Further examples are given by Weinmann and Bube [14] and in part 1 of this thesis.
2. The Immersed Interface Method on Boolean grids

2.1. Introduction

There is a constant desire to make computations fast and efficient as possible, in order to be able to solve larger problems with higher accuracy, or just to minimize the time spent waiting for the computer to carry out the calculations. The Boolean grids proposed here provide one method to make computations faster or by using less data, while still obtaining the same accuracy in the calculations.

The Boolean methods were originally developed in the 1960s in order to represent surfaces used in computer-aided design (CAD). They were first used by Coons [1] to create interpolatory surfaces, coinciding with prescribed values on the boundary of the unit square. The theory was then developed in a series of articles by Gordon [8, 7], resulting in an abstract theory of commutative projectors which is presented here in section 2.2.

There is also a book by Delves [6], where the methods are presented in some detail.

As we will see, it is straightforward to use these differences on Boolean grids and one may use FFT to and not look on discretizing PDEs on such grids, gaining several orders of magnitude in the number of points needed for a given accuracy. However, just as for uniform grids, these methods can also be applied to rectangular regions. Therefore it is interesting to apply the Immersed Interface Method from the previous section to Boolean grids in order to get a similar decrease in computational time even for problems in irregularly shaped domains.

The outline of our presentation is as follows. In section 2.2, we present the abstract theory of Boolean interpolation and present Boolean grids on which we apply finite differences and the immersed interface method. In section 3.3, we discuss how to extend the immersed interface method to work on the Boolean grids and finally, in section 4.4, we give some numerical examples of the use of Boolean grids for interpolation and equation solving.

As for the IMM on uniform grids, we do not have any proofs of convergence or error estimates for the IMM on Boolean grids. We only confirm numerical convergence and superiority to the uniform IMM.

2.2. Boolean interpolation

2.2.1. The algebraic theory

The following presentation is taken from Gordon [7] and is an abstract algebraic approach to approximation theory giving a motivation for the use of Boolean approximations.

We consider an arbitrary function space $F$. On this space, we define $M$ projection $P_j$ $j = 1, \ldots, M$, meaning that $P_j : F \rightarrow \Phi_j$ is a linear transformation with the property $P_j P_k = P_k$ for $j = 1, \ldots, M$.

A function $\tau = \tau_f \in \Phi_j$ is called the approximation of $f \in F$, and the function $f$ is called the remainder.

We define multiplication and addition of projectors in the natural way and note that the associative and distributive rules hold for projectors defined on the same domain $F$. Furthermore, we assume that multiplication is commutative in.

$$P_j P_k = P_k P_j \quad \text{for all } j, k = 1, \ldots, M.$$  (2.1)

It is obvious that the product of two commutative projectors, $A = P_j P_k$, is again a projector since $AA = A$. This is not true, however, for the sum of two projectors, since

$$(P_j + P_k)(P_j + P_k) = P_j P_j + P_k P_k + P_j P_k + P_k P_j = P_j + P_k + 2P_j P_k \neq P_j + P_k.$$  (2.2)

Therefore, we introduce the Boolean addition denoted by $\oplus$ and defined by

$$P_j \oplus P_k = P_j + P_k - P_j P_k.$$  (2.3)

It is easy to check that $P_j \oplus P_k$ is again a projector.

We also need to compare projectors in order to decide which are better than others. To this end, we introduce the ordering relation $\leq$ defined by

$$P_j \leq P_k \Leftrightarrow P_j \oplus P_k = P_j,$$  (2.4)

that is, if $P_j \leq P_k$, then $P_j$ remains at least as good as the function $f$ at points in $\Phi_j$.

Now, we may define the space $\Psi$ as the set of all projectors which can be built up as combinations of the $P_j, j = 1, \ldots, M$, under the operation of multiplication and Boolean addition. The set $\Psi$ now a distributive lattice under the partial ordering $\leq$. This means that for all projectors $A, B, C \in \Psi$, the following properties hold (and are easily checked):

i) reflexivity $A \leq A$

ii) antisymmetry $A \leq B$ and $B \leq A \Rightarrow A = B$

iii) transitivity $A \leq B$ and $B \leq C \Rightarrow A \leq C$

iv) idempotence $A \leq A$ and $AA = A$

v) commutativity $A \leq B \Rightarrow B \leq A$ and $AB = BA$  \hspace{1cm} (2.5)

vi) associativity $(A \oplus B) \oplus C = A \oplus (B \oplus C)$, $(A \oplus B) \oplus C = (A \oplus B) \oplus C$

vii) distributivity $(A \oplus B) \oplus C = AB \oplus AC$ and $A \oplus (B \oplus C) = (A \oplus B) \oplus (A \oplus C)$

viii) consistency $A \leq B \Rightarrow AB = A \Rightarrow A \leq B = B$

It is a property of every lattice that any pair $(A, B)$ of elements has both a least upper bound (denoted $\sup$), that is the least element $C$ such that $C \geq A$ and $C \geq B$, and a greatest lower bound (denoted $\inf$), that is the largest element $D$ such that $D \leq A$ and $D \leq B$. These are given explicitly by

$$\sup(A, B) = AB, \quad \inf(A, B) = AB.$$  (2.6)
It follows that every finite lattice has a unique maximal element \( M \in \Psi \), that is an element satisfying \( \Delta \leq M \) for all \( \Delta \in \Psi \), and a unique minimal element \( \mathcal{L} \), for which \( \mathcal{L} \leq \Delta \) for all \( \Delta \in \Psi \). It is easily seen that

\[
M = \text{sup}(P_1, \ldots, P_m), \\
\mathcal{L} = \text{inf}(P_1, \ldots, P_m).
\]

It is also interesting to study the range of the projector in \( \Psi \). It is clear that the range of \( P_1 \oplus P_2 \oplus \cdots \oplus P_m \), and the range of \( P_1 \cap P_2 \cap \cdots \cap P_m \), namely \( \Phi_1 \cup \cdots \cup \Phi_m \) and the range of \( \mathcal{L} \) is the smallest space, \( \Phi_1 \cap \cdots \cap \Phi_m \).

Finally, we introduce the remainder operator or the complement of a projector \( A \), namely

\[
\Delta' = I - \Delta,
\]

where \( I \) is the identity operator. It is clear that \( \Delta' \) is a projector, and that

\[
\Delta\Delta' = \Delta', \quad \Delta'\Delta = \Delta, \quad \Delta \oplus \Delta' = I.
\]

Now for each \( P_j \), we set \( R_j = \text{sup} P_j - \text{inf} P_j \) and note that although in general \( R_j \notin \Psi \), the set of combinations of these remainder operators also form a distributive lattice, denoted \( \Psi' \). Also, if we allow the three operation multiplications, Boolean addition and complement to work on the projection, we generate a Boolean algebra, which additionally de Morgan's laws hold

\[
(A \oplus B) = A'B', \quad (AB)' = A' \oplus B',
\]

Now, the final statement is that given a commutative set of projectors \( \{P_i\} \), the identity operator, \( I \) has a maximal decomposition

\[
I = M \oplus \mathcal{L} = M_1 \oplus \cdots \oplus M_m + (R_1 \oplus \cdots \oplus R_n),
\]

and a minimal decomposition

\[
I = \mathcal{L} \oplus \mathcal{L}' = \mathcal{L}_1 \oplus \cdots \oplus \mathcal{L}_n + (\mathcal{R}_1 \oplus \cdots \oplus \mathcal{R}_m),
\]

Here, \( R_1 \oplus \cdots \oplus R_n = \text{inf}(R_1, \ldots, R_n) \in \Psi \) and \( R_1 \oplus \cdots \oplus R_n = \text{sup}(R_1, \ldots, R_n) \in \Psi' \) as seen before. This means that by choosing the algebraically maximal projector \( M \), we minimize the remainder and vice versa.

2.2.3. Boolean grids

We now wish to apply the abstract results from the previous section to the problem of coming up with a grid on which we wish to apply finite difference schemes. This could be done in many ways, depending on how one defines the grid points \( P_1, \ldots, P_N \). Here, we only discuss the grids that we actually use and the reason for using these will be clear later on.
2.2 Boolean Interpolation

The two-dimensional Boolean grid with step size $h_1$ and $h_2$. The grid consists of three parts, the horizontal limit $G_1$, the vertical limit $G_2$, and the coarse grid $G_3$, consisting of the intersection node of these lines. These three are combined as in (4.8) below to create the Boolean approximations.

and $h_{3p}$ and project on $P^3_2$ and $P^3_1$ identical to $P^3$ except for the step size. Since the error in $P^3_2$ is of order $h^3_2$, we see that to preserve the $O(h^3_2)h^3_2)$ error estimate, we should set

$$h_{12} = h_{23} = h_{13}h_{12},$$

or $h_1 = h_2$ if the step sizes are equal in the two directions. This results in a grid like the one in Figure 2.1. It is important to ensure that the nodes for $P_1$ are a subset of the nodes for $P_1$, so that $P_1$ and $P_1'$ commute and that $P_1' = P_1$.

The resulting problem or may be expressed in several ways, which are equivalent. It can be shown by direct calculation using the rules (3.3) and the fact that $P_1^2 = P_2^2$ and $P_2^2 = P_2^2$.

Denoting the resulting problem or by $P_{2p}$, we have

$$P_{2p} = P_2^2 P_2^2 - P_2^2 P_2^2 = P_2^2 P_2^2 - P_2^2 P_2^2 = P_2 P_2 P_2 P_2.$$

The last form shows the three uniform grids which we must combine in order to get the Boolean approximation, namely the grids $G_1$, with steps $h_{12}$ and $h_{23}$, $G_2$ with steps $h_{12}$ and $h_{13}$, and $G_3$ with steps $h_{12}$ and $h_{13}$. Using the second form of $P_{2p}$ and de Morgan's law, we may also express the remainder at

$$R_{2p} = R_2 P_2 P_2 P_2 = R_2 R_2 R_2 R_2 = R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_2 R_
Figure 2.1: The bounding blocks of Boolean grids in 3D. The maximal grid corresponding to $P_2^M$ is shown on the left and the intermediate grid of $P_2$ on the right. The maximal grid is created by using a two dimensional Boolean grid on each of the sides of the cube and is a combination of 13 different grids. The intermediate grid is created by drawing lines in each coordinate direction and consists of only four different grids.

where $y_2$ are half-edge angles at before, $x_2 = (x_1 + (1/2)) y_2$, and $x_{2,x} = (h_2 - h_1)/(N_2 - 1)$. We similarly define $P^2$, $P_1^2$, $P_2$, $P_0$, $P_1$, and $P_2$, it is clear that $P^2$ interpolates a function on the plane $x = x_2$ and that the error introduced by this approximation is $O(h_2^3)$. Now, to make a full Boolean approximation, we would make an approximation on each of the planes $z = z_2$ using the two-dimensional Boolean grid introduced above, resulting in a projection for this direction given by

$$P^M = (P^2 P_1^2 \otimes P^2 P_1^2) P^2,$$

where we have introduced the projections $P_2^2$ and $P_1^2$ for $x_2$. The step sizes $h_{x_2}$ and $h_{x_1}$ are simply $h_{x_2} = (h_{x_1} + (1/2)) y_2$, respectively. Defining $P^M$ and $P^M$ similarly and taking the Boolean sum, we get the final Boolean projection in three dimensions:

$$P^M = P^M \oplus P^M \oplus P^M,$$

$$= (P^2 P_1^2 \otimes P^2 P_1^2) P^2 \oplus (P^2 P_1^2 \otimes P^2 P_1^2) P^2 \oplus (P^2 P_1^2 \otimes P^2 P_1^2) P^2$$

$$= P_1^3 \oplus P_1^3 \oplus P_1^3 \oplus P_1^3 \oplus P_1^3 \oplus P_1^3 \oplus P_1^3 \oplus P_1^3 \oplus P_1^3,$$

where we have used the notation $P^2 P_1^2 \otimes P^2 P_1^2$. This would result in a grid made up of cubes like the one on the left of figure 2.2. In order to estimate the error of this projection, we study the remainder error or $R_{xy, z} = 1 - P^M$.
which means that $R_{w_{w}}$ includes terms of the type $R_{w_{a}}^{2}$ and $R_{w_{a}}^{2}R_{w_{a}}^{2}$ as well as higher order terms. Therefore

$$R_{w_{w}}f = O(h_{w}^{3} + h_{b}^{3}),$$

and we see that a choice of $h_{w} = h_{b}$ gives the optimal $O(h_{w})$ error. The number of points used is now

$$N_{w} = 3N_{1}^{2}N_{2}^{2} - 2N_{1}^{2} = O(N_{w}^{3}),$$

while for a regular grid $N_{w} = N_{1}^{2} = O(N_{w}^{4})$ for the same accuracy.

### 2.2.3. Application to finite difference solvers

It is clear from the previous section that if we can create some approximation method with second order error in each direction, we can use the Boolean grids defined there to improve accuracy. While using few grid nodes (as compared to standard grids), specifically we need to use those grids for finite difference approximations of derivatives, in particular the Laplacian in one or three dimensions.

Starting in two dimensions, we therefore define the second order difference approximations of the second derivative in $x$ by

$$D_{x}^{2}u(x,y) = \frac{u(x+1,y) - 2u(x,y) + u(x-1,y)}{h_{1}^{2}},$$

which are second order accurate that is the error in the approximation is $O(h_{1}^{2})$. $D_{x}^{2}u$ is defined similarly. The discrete Laplacian is then given by

$$\Delta_{x,y}^{2}u(x,y) = D_{x}^{2}u(x,y) + D_{y}^{2}u(x,y).$$

Assuming for simplicity that the grid points are equal in the two directions $h_{x} = h_{y} = h$, we may then investigate the infinite case and the finite difference approximations of the Laplacian, which according to (5) should be defined as

$$\Delta_{x,y}^{2}u(x,y) = \Delta_{x}^{2}u(x,y) + \Delta_{y}^{2}u(x,y).$$

The accuracy of this approximation is then $O(h_{1}^{2} + h_{2}^{2})$, i.e. $O(h)$ if we choose $h_{1} = h_{2}$, which is again easy to see why the error comes out. The two last approximations have one fine direction and one coarse. The derived now in the coarse direction are controlled by subtraction of the last term which is coarse in both directions. It should be emphasized that the $O(h_{w}^{3})$-accuracy holds not only on the Boolean grid in figure 2.1, but at any point internal to the domain, as long as the Boolean combination of the three components, that is, having calculated the three contributions $\Delta_{x}^{2}u$, $\Delta_{y}^{2}u$ and $\Delta_{x,y}^{2}u$ for each grid, we may then analyze and combine these three contributions by the Boolean scheme to get the value at any point. These internal points introduce new error terms, and the error is in general larger, but the index of the error in the same as before.

In three dimensions, things are very much the same. We use the intermediate scheme defined by (5, 0) on the grid on the right of figure 2.2, which consists of five grids. Three grids are fine in one direction and coarse in the two other, while the last grid is coarse in all directions. Again, using the appropriate Boolean scheme gives us $O(h)$, but this $O(h^{3})$ If we choose $h_{1} = h_{2} = h_{3}$, it is again easy to see why the error comes out. The two last approximations have one fine direction and one coarse. The derived now in the coarse direction are controlled by subtraction of the last term which is coarse in both directions. It should be emphasized that the $O(h_{w}^{3})$-accuracy holds not only on the Boolean grid in figure 2.1, but at any point internal to the domain, as long as the Boolean combination of the three components, that is, having calculated the three contributions $\Delta_{x}^{2}u$, $\Delta_{y}^{2}u$ and $\Delta_{x,y}^{2}u$ for each grid, we may then analyze and combine these three contributions by the Boolean scheme to get the value at any point. These internal points introduce new error terms, and the error is in general larger, but the index of the error in the same as before.

### 2.3. Applying IMM to Boolean grids

Having successfully applied finite differences on the Boolean grids, we now try to implement the Immersed Interface Method as described in section 1 on these grids in order to cope with domains that are not rectangular.
From the previous section, it might be suggested that one could use the Immersed Interface Method straight away on the component grids and then combine them using the Boolean scheme to get improved results. This does not work, however, since when we introduce jumps in the finite differences, the errors no longer behave in the correct way to be eliminated by the Boolean scheme. This is seen in figure 23.

Therefore, we need a slightly more elaborate method. The reason for not getting the desired elimination of errors is simply the estimation of the jumps at the boundaries. Since the results in section 2.3 show that finite differences work fine on Boolean grids, so, instead of approximating the jumps on each grid individually, we use the Boolean interpolation to get the jumps from the solution. That is, we must create a matrix \( D^B_p \) implementing this Boolean interpolation so that

\[
C = D^B_p U + F_1
\]

with \( U = [U_{11}, U_{12}, U_{13}]^T \) containing the solution on the three grids (assuming we are solving a two-dimensional problem). Here \( u_0 \) is the solution on the grid created by the projection \( P^B \). As before, if we intend to solve the Poisson equation, we also have the equation

\[
\Delta u + \Phi C = F_1
\]

Here \( \Delta u \) would act on these grids separately and \( \Phi \) is divided into three blocks corresponding to the grids. As in the uniform case, we also solve for \( C \) in the equation

\[
(I + D^B_p \Delta)^{-1}
\]

and then finally compute the solution \( U \) as

\[
U = B \Delta^{-1} (\Phi C + F_1)
\]

Here \( B \) is a Boolean interpolation matrix, combining the three solutions \( u_{11}, u_{12}, \text{and } u_{13} \) into the final solution \( U \), which could be defined on any grid.

The procedure outlined here is our Boolean HIM, resulting in the errors shown in section 2.3. The only difference to the uniform HIM described in section 1 is the jump approximation matrix \( D^B_p \) and the interpolation matrix \( B \). The interpolation matrix \( B \) uses the Boolean schemes (2.8) and (3.5) in two and three dimensions respectively, together with linear interpolation between grid nodes, in order to get the solution \( U \) at the desired points. The matrix \( D^B_p \) is slightly more complicated and we will now discuss how to form it.

### 2.3.1. Boolean approximation of jumps

Just as in the HIM on uniform grids, we need to approximate the jumps of the solution and its derivative on the boundary, using the values of the solution on the grid nodes. The boundary conditions are implemented exactly as for the uniform grids (see section 1.4), so what we need to do here is to create equivalent to matrices \( Q_1, F_1 \) and \( R_1 \) in [1.12]...
in order to eliminate the values of \( u \) and \( \delta \) derivatives at the interface from non-interface points (IPs) from both sides of the interface.

So, first of all, we need to find all the IPs that in all the places where the interface intersects the grid lines and that intersect with the finite differences. Suppose that we want to calculate the Laplacian in two dimensions. Then the IPs are at all the points where the interface intersects the fine grid, as shown in Figure 2.4. That is, we will have as many IPs as at the case of a uniform grid. In three dimensions, however, the IPs are distributed on the surface in a two-dimensional Boolean grid, meaning that we have much fewer IPs using a Boolean grid than a uniform one. This also implies that the vector \( C \) will be smaller and that the entire system of equations will be smaller.

Having found the IPs, we assign to each of them two (or more) anchor nodes, on both sides of the surface. These are the nodes where the finite difference is affected by the jump in the solution at the IP. The particular difference taken may be either of size \( h_x \) on the side \( h_x \), as shown in Figure 2.4. From (1.4), we see that this gives rise to corrections to the regular second order difference \( D_y u \) given by

\[
\frac{d^2 u(x)}{dx^2} = \alpha^2 \frac{d u(x)}{dx} + \sum_{m=0}^{2} \frac{1}{h^m} \alpha^m \frac{d^m u}{d x^m} + O(h^3),
\]

Here, \( \alpha \) is the location of the IP, so that \( x \approx h_x \), and \( h_x \) may be either \( h_x \) or \( h_y \). The question now is how well we need to approximate the jump and how many derivatives we need. We now consider that in the uniform case it is enough to get an approximation of order \( O(h) \) on the boundary in order to get overall \( O(h) \) convergence, and therefore that we may ignore the jumps in the third derivative. In the Boolean case, however, it is not quite clear what we need to do. We want overall \( O(h) \) convergence, but if we ignore the jumps in third derivative in (2.10), we will at some points get errors of order \( O(h^3) \), which seems to be too large. Fortunately, our numerical results show that it is in fact sufficient to use jumps in \( u' \) and \( u'' \) and use Boolean interpolation to approximate these to order \( O(h^2) \) and \( O(h^3) \) respectively. From (2.10), we then see that we will get truncation errors of order \( O(h^2) \) at some nodes (where \( h = h_x \) and \( h = h_y \)) at other nodes, while the overall error in the solution will be of order \( O(h^3) \). The explicit expression that this approach works is known.

Here this is done is most easily seen from Figure 2.5. Suppose we want to approximate the function value and the two first derivatives in the \( x \)-direction at the IP \( x_i \) with coordinates \( (x_i, \delta x_i) \). We use regular Lagrange interpolation on each of the grids, that is, give grid nodes with \( x \)-coordinates \( x_i, x_{i+2} \), we set

\[
t(x) = \sum_{i=0}^{2} u(x_i) \phi_i(x),
\]

with the basis functions

\[
\phi_i(x) = \prod_{k \neq i} \frac{x - x_k}{x_i - x_k}, \quad i = 1, 2.
\]

To compute derivatives, we just differentiate the basis functions, getting

\[
\phi'(x) = \sum_{i=1}^{3} \frac{1}{h_i} \prod_{k \neq i} \frac{x - x_k}{h_i},
\]

and so on. In this way, we may write for each of the three grids

\[
u(x,y) = \sum_{i=1}^{3} \frac{1}{h_i} \prod_{k \neq i} \frac{x - x_k}{h_i} \delta(x,y),
\]

Here it is of course the projection of \( u \) onto the space spanned by the basis functions, so from the theory of Section 2.3 we expect that if we combine the three grids according to the Boolean sum we would eliminate the largest error. Therefore, we choose grid nodes according to Figure 2.5, with (at least) three nodes in the \( x \)-direction in order to approximate second derivatives, and (at least) two nodes in the \( y \)-direction in order to extrapolate the function to the IP in that direction.

One could also understand the choice of nodes in the following way. What we really want is a fine \( (h_x) \) approximation in the \( x \)-direction using the grid \( G_{11} \), which is fine in that direction. But since we must extrapolate in the \( y \)-direction, we introduce large \( (h_x) \) error in that direction. These are eliminated by subtracting the approximation on the coarse grid \( G_{10} \), but this in turn adds large \( (h_x) \) error in the \( y \)-direction. These are finally eliminated by adding the approximation on the \( G_2 \), which does not introduce new error in the \( y \)-direction.
These principles may be used to approximate any derivative to any order of accuracy. If one uses enough grid nodes for the approximations. For our purpose it is best to use as few grid nodes as possible since the more nodes we use the more coupled the resulting system of equations becomes, which makes it harder to solve. Actually the minimal number of 11 nodes shown in figure 2.5 is enough more than the six nodes needed on uniform grids. One should also note, however, that if the IP lies on one of the coarse grid lines (as point A in figure 2.4), it is sufficient to use the three nodes on the fine grid C0, and the Boolean combination G1 is then not needed. But, if one would like to solve the Neumann problem, one would also need projection, which requires additional nodes in a configuration similar to the one we have studied here.

Anyway the final scheme may be expanded in matrix form as

\[
\begin{bmatrix}
  u^0(\alpha_1) \\
  w^1(\alpha_1) \\
  w^2(\alpha_1)
\end{bmatrix} = R C U,
\]

where \( R = [I I I] \) performs the Boolean combination, \( R_i \) consists only of 1's and 0's and just reflects and permutes nodes, while \( P^0 \) performs the Lagrangian interpolation to evaluate \( u(\alpha_i) \) and its derivative using the grid \( C_0 \).

If we apply the same procedure on the other side of the interior (if necessary) to create \( P^2 \) and \( R_j \) and then create the matrix \( L \) exactly as in section 1.4, we may estimate jumps at the boundary as

\[
\begin{bmatrix}
  u^0 \\
  w^1 \\
  w^2
\end{bmatrix} = LBP R C U,
\]

which should be compared to (1.13). We set \( D^0_{ij} = LBP R C U \) and add the contributions from all the IP's to get the matrix \( D^0 \) and the system of equations

\[
C = D^0 U + F,
\]

Not that able to approximate the jumps, we may use the method outlined earlier to apply the Immersed Interface Method on Boolean grids. For problems in three dimensions, the same procedure is used. There is no need to use a threedimensional approximation of the jumps, as we can use the stencil shown in figure 5.3 for threedimensional problems as well.

2.4 Examples

2.4.1 Example 1 Boolean approximation

In this example, we illustrate simple Boolean interpolation. A sixth degree polynomial in two dimensions is solved on a uniform grid with 64 \times 64 nodes and on a Boolean grid with \( N_1 = 8 \) and \( N_2 = 64 \). Both of these approximations are then linearly interpolated onto a uniform grid with 128 \times 128 nodes. The errors are evaluated using the uniform grid, and the Boolean grid is then used to evaluate the errors on the uniform grid. The errors are calculated on this grid. Not that the error of the Boolean grid is calculated on the error image on the right of figure 2.7. This indicates that the error is smaller on the Boolean grid than in between the grid lines, not that interpolation errors are greater than errors resulting from grid refinement on the grid itself.

Figure 2.8 shows the comparison of the same approximation compared to approximations on a uniform grid. On the left we see a log-log plot of the error for the discrete Laplacian applied to \( u(x) \), computed on the uniform grid. The results are very close to the exact solution. The error on the uniform grid is then used as a reference to evaluate the error on the Boolean grid. The Boolean grid has a lower order of convergence than the uniform grid, but the error is still smaller than the error on the uniform grid.
Figure 2.7: Finite different on a Boolean grid with \( N_1 = 12 \) and \( N_2 = 12 \). The discrete Laplacian was applied to the function \( f(x) = \sin(2\pi x) \sin(2\pi y) \) on a Boolean grid and then interpolated to a full grid. The result is shown on the left with errors on the right. The error images show that the errors are smaller on the Boolean grid than between grid lines.

Figure 2.8: Asymptotic for finite difference approximations on two dimensional uniform and Boolean grids for the forward Laplacian on the left and the inverse Laplacian on the right. The graphs are log-log plots of error as function of the number of grid nodes. In both graphs, the slope of the lines are \( -0.86 \) for the uniform grids and \( -1.34 \) for the Boolean grids. The Boolean grids have \( N_1 = N_2^2 / 2 \) and values of \( N_1 \) range from 24 to 48 nodes.

Figure 2.9: Error asymptotics for the discrete Laplacian (left) and the inverse (right) applied using finite differences on uniform and Boolean grids in 3D. The plots are log-log plots of the number of grid nodes versus the maximum error and the slopes of the lines are near the theoretical values of 2/3 for the uniform grids and 1 for the Boolean grids.

In the last figure 2.10, the same problem in the unit cube in 3D is shown. Here \( f(x) = \sin(2\pi x) \sin(2\pi y) \sin(2\pi z) \) with \( \Delta f(x) = 12 \pi^2 f(x) \). The figure shows results for the Laplacian on the left and for the inverse Laplacian on the right. The Boolean grid is the interior grid with four components, \( N_1 \) ranges from 12 to 32 and \( N_2 = N_3^3 / 4 \). The slopes of the lines for the uniform grid are both \( 0.667 \), which agrees perfectly with the theoretical value derived from \( O(N^2) = O(N^2) \) accuracy and \( O(N^3) \) unknown. For the Boolean grid, the slope is the figure is \( 0.667 \) for the Laplacian and \( 1.1 \) for its inverse, which agrees with the theoretical value of 1, arising from the fact that the accuracy is \( O(N) = O(N^2) \) and the number of unknowns is \( 2N_1 N_2 + N_2 = O(N^3) \).

The conclusion is that the theory works for standard finite difference and that by using Boolean grids the number of nodes may be reduced without ill affecting the error terms. It should be noted that the choice of \( N_1 \) here are not necessarily optimal, \( N_1 \) is proportional to \( N_2 \), which it must be, in order to achieve the correct asymptotics. But given a specific \( N_2 \), it is set close beforehand which is the optimal \( N_1 \). There is a limit where there is no use increasing \( N_1 \) further, because the dominating error terms depend only on \( N_1 \). This limit is problem dependent, since the sizes of the error terms depend on...
Figure 2.10: HIM on Boolean grid. On the left we see the numerical solution $u(x, y)$ to the Poisson problem in an ellipse computed on a Boolean grid with $N_2 = 24$ and $N_1 = 24 \times 6 = 144$. Level curves are drawn at the bottom showing clearly the ellipse $E$. The solution is zero outside the domain. On the right, we see the errors compared to the exact solution on the CO grid with small steps in the $y$-direction.

The magnitude of the derivatives of the solution $f$. Choosing optimal values of $N_2$ for all $N_1$ will not change the slope of the lines, only the position that is it will not change the asymptotes even if it changes the error value.

2.4.3. Example 3 - HIM on Boolean grids

Our final example shows the full immersed interface method on Boolean grids. We solve the Poisson equation with Dirichlet boundary conditions inside an ellipse $E$ with center at $(0.5, 0.5)$ and half axes 0.44 and 0.38, i.e.,

$$
\Delta u(x, y) = f(x, y), \quad (x, y) \in E,
$$

$$
u(x, y) = g(x, y), \quad (x, y) \in \partial E,
$$

with $f(x, y)$ and $g(x, y)$ chosen so that the solution $u(x, y)$ is a given quasi-degree polynomial (the same as in example 1). The solution and the distribution of errors on one of the computed grids ($C_2$) are shown in figure 2.11b. The results for different grid sizes are listed in table 1. $[E]_{C_{1.2}}$ denotes the maximum error compared to the exact solution on the Boolean grid, while $[T]_{C_{1.2}}$ denotes the maximum truncation error, i.e., $T_e = \Delta u_{1.0} + 4C_1 F_1$, where $u_1$ and $C_1$ are the grid values and jumps for the exact solution, respectively. The table also lists approximate computational times.

The asymptotes of the errors are shown in figure 2.11. These asymptotes for a corresponding three-dimensional problem is also shown, that is the Poisson equation inside an ellipsoid with Dirichlet boundary conditions, whose solution is a quasi-degree polynomial. The corresponding data is shown in table 2. The numerical asymptotes for the errors $\{E_{i,j}\}$ agree well with the theoretical values for $O(\varepsilon)$ behavior given in the previous example. As expected, these errors exhibit $O(\varepsilon)$ behavior in both two and three dimensions, but even though these errors are large, the final errors in the solution are small. The computational times listed indicate that the time needed to solve the problem grows slightly faster than the number of unknowns. This is because the number of iterations needed in the ICCS algorithm to solve the linear system of equations is not constant, but tend to increase slightly with the number of unknowns, (e.g., from 17 to 21 in the 3D problem).

We conclude that we are able to achieve $O(\varepsilon^2)$ error asymptotics for both two and three dimensional problems in the Boolean grids. We also see from the error grids that the number of grid nodes needed to get a specific error is much lower for the Boolean grids than for the uniform one.
### Examples

| $N_x$ | $N_y$ | Time (s) | $|E_x|_{\infty}$ | $|T_x|_{\infty}$ |
|-------|-------|----------|----------------|-----------------|
| 28    | 6     | 7.498    | < 0.01         | 3.36 x 3        | 2.31            |
| 26    | 7     | 17.063   | 0.2           | 3.53 x 3        | 1.89            |
| 32    | 8     | 17.408   | 0.2           | 1.76 x 3        | 1.87            |
| 36    | 9     | 2.424    | 0.3           | 1.34 x 3        | 1.19            |
| 40    | 10    | 3.340    | 0.2           | 1.54 x 4        | 1.24            |
| 48    | 12    | 5.720    | 0.3           | 2.84 x 4        | 1.23            |
| 56    | 14    | 0.0044   | 1.4           | 2.16 x 4        | 1.02            |
| 64    | 16    | 1.5101   | 2.3           | 1.31 x 4        | 0.46            |

Table 1: Results for Boolean IIM in two dimensions, $n = N_x N_y$ and $N_{tot} = N_x^2 (2n_1 + 1)$.

### References

1. FFTW, http://www.fftw.org
2. IMEX (Iterative Methods EXtender), http://math.nist.gov/imex/
REFERENCES


APPROXIMATION OF GENERALIZED MEAN CURVATURE FLOW WITH RIGHT-ANGLE BOUNDARY CONDITIONS

Tobias Gebäck

Abstract

In this paper, we prove the convergence of an algorithm for computing the evolution of surfaces, which move at each point with a velocity equal to an increasing function of the mean curvature at that point. Furthermore, the entire evolution is assumed to take place inside a convex domain and whenever the surface intersects the domain boundary, it should do so at a right angle. We show that the approximations given by the algorithm converge to the viscosity solution of the corresponding PDE as the time step tends to zero.

The algorithm presented here is a generalization of the algorithm presented by Luiii and Luiii for regular mean curvature evolution with right-angle boundary conditions and the algorithm by Grabovskii and Heinze for the case when the velocity equals an increasing function of the mean curvature, without boundary conditions. These algorithms are in form based on the convolution-thresholding scheme devised by Bruck, Merriman and Osher.

Contents

1 Introduction
   1.1 The BMO algorithm .................................................. 2
   1.2 Right-angle boundary conditions ..................................... 3
   1.3 Generalized mean curvature motion ................................. 3
   1.4 Outlines ................................................................. 4

2 Viscosity solutions
   2.1 Introduction ............................................................. 5
   2.2 Theory of viscosity solutions ......................................... 5
   2.3 Boundary conditions .................................................. 7
   2.4 Regular equations ..................................................... 8
   2.5 Convergence ............................................................ 8

3 The algorithm ............................................................... 9

4 Properties of $G_h$ and $\tilde{G}_h$ ....................................... 12

5 The convergence theorem .................................................. 23

6 Examples ........................................................................... 33

References ............................................................................. 36
1. Introduction

Consider a hypersurface $\Gamma_1$ in $\mathbb{R}^n$. At each point $x \in \Gamma_1$, assign a velocity $\kappa$ in the normal direction, so that $\Gamma_1$ moves at each point with velocity $\kappa(2)$, creating a new hypersurface. Continuing the process, a family $(\Gamma_r)_r$ is created, where the hypersurfaces $\Gamma_r$ evolve according to the normal velocity $\kappa_2\kappa$. If we take $\nu(t)$, $\lambda = \kappa$, the mean curvature of $\Gamma_r$ at $x$, we get the mean curvature flow. Mean curvature flows have been studied since the 1970s, first by geometric methods from differential geometry, although it was seen that for $n \geq 3$ these methods ran into problems even for smooth hypersurfaces $\Gamma_r$, as the mean curvature flow could develop singularities (so that the curvature is not defined for some $t$). For figure 1, a method to overcome these problems was developed by Huisken [5], who used uniform convexity of weak solutions of mean curvature flow (see also [6] for a modified version of these results). This method, however, does not give unique solutions.

Figure 1: The mean curvature evolution of a dumbbell-shaped domain. The surface develops a singularity after a finite time and is split into two. The image was produced by R. Grigor'evs.

Then, following ideas from Osher and Sethian [17], Evans and Spruck [10] developed a new approach to motion by mean curvature in which the hypersurface $\Gamma_1$ is viewed as a level set of a continuous function $f$, so that

$$\Gamma_1 = \{x \in \mathbb{R}^n \mid f(x) = \lambda\}$$

for some $\lambda$. The mean curvature evolution is then studied through the PDE

$$\begin{cases}
\frac{\partial u}{\partial t} - |\nabla u| \frac{\nabla u}{|\nabla u|} &= f \\
u &= f
\end{cases} \quad \text{in } \mathbb{R}^n \times (0, \infty),$$

$$\text{on } \mathbb{R}^n \times \{t = 0\},$$

which ensures that the level sets of $u$ evolve according to their mean curvature, at least as long as $\nabla u \neq 0$. This PDE is nonlinear and degenerate parabolic and has a singularity for $\nabla u = 0$, which makes it harder to handle. However, the notion of viscosity solutions (see Crandall, Ishii, Lions [7]) provides a well-posed tool to handle such equations and to prove the existence of a unique solution, which was done in [10].

2. Approximation of Generalized Mean Curvature Flow

A number of generalizations of this approach have appeared. Existence of a unique solution to more general level set equations has been established by Cheng, Giga and Gurtin [6] and by very general cases, including the case where $x = \rho(\kappa)$ and $\rho$ a non-decreasing, continuous function, by Ishii and Sato [16]. The Neumann problem for the mean curvature equation has also been studied. In this case, the whole evolution takes place inside a domain $\Omega$, whose boundary $\partial \Omega$ is integrated geometrically by the level sets of $\kappa$. The existence of unique solutions in this case was established by Ishii [17] for convex $\Omega$ and by Giga and Satô [18] for more general $\Omega$, but less general dependence on the structure. We will return to this later.

Curvature flows are most naturally in a regime of situations, including fast reaction-diffusion problems (see [19]) and image processing (see [20]).

In 1995, Bence, Merriman and Osher [19] generated an algorithm to approximate motions by mean curvature using the level set method. The convergence of this algorithm was proven using different approaches by Evans [9], Bades and Qian [8] and Ishii [13]. General thresholding schemes were also studied by Ishii, Pires and Sato [20]. Later, two significant generalizations of the algorithm have been developed. In 2005, Ishii and Sato [16] published an algorithm for motion by mean curvature with right-angle boundary conditions, and about at the same time, Grigors and Heister [13] developed a scheme for motion with normal velocity equal to a (nonsmooth) function of the mean curvature, and others for generalized mean curvature motion.

This work uses the method used by Ishii and Sato [13] to prove convergence of an algorithm for the generalized mean curvature motion with right-angle boundary conditions. In order to understand how this algorithm works, we will briefly discuss the previous works.

1.1. The BMO Algorithm

A (slightly generalized) version of the Bence, Merriman and Osher algorithm (BMO algorithm) can be described as follows (cf. Ishii [13]). First, fix a radially symmetric convolution kernel $\rho$, and define its convolution $\rho^\ast(x) = \rho(x^2/\delta^2)$. Then, given a set $C_0 \subset \mathbb{R}^n$, change a time step and compute the convolution $\rho^\ast(x, h) = (\rho^\ast(x, h_{ij}))_{ij}$. Set

$$C_{r+1} = \{x \in \mathbb{R}^n \mid \rho^\ast(x, h_{ij}) \geq \int_{\mathbb{R}^n} \rho^\ast(x, h_{ij}) \}$$

and compute the pressure by computing $\rho^\ast(x, h_{ij})$ and defining $C_0$ and $C_{r+1}$. Then we end up with a sequence $(C_{r+1})_{r}$ of closed sets in $\mathbb{R}^n$ and we get

$$C_{r+1} \supset C_r$$

for some $r \geq 0$. Now, let $h \rightarrow 0$, we obtain a flow of closed subsets in $\mathbb{R}^n$ whose boundaries move with a normal velocity equal to a constant times its mean curvature, where the constant depends only on $\rho$ and the choice of $\rho$.

In the original algorithm, $\rho^\ast(x, h_{ij})$ was the solution to the heat equation with initial data $\chi_0$, which corresponds to the choice of $\rho$ as the Gaussian kernel, and which leads to motion by (n - 1) times the mean curvature.
1.2 Right-angle boundary conditions

As we already mentioned, the above algorithm was extended by Laih and Laih [14] to the case of right-angle boundary conditions. The extension works as follows. Given an open domain \( \Omega \subseteq \mathbb{R}^n \) with \( C^2 \) boundary, an initial set \( C_0 \), and a convolution kernel \( \rho \), we define
\[
M^\rho(x, h) = \int_{\Omega} \rho^\rho(y, x) \chi_\Omega(y) \, dy
\]
and set
\[
C_1 = \{ x \in \mathbb{R}^n \mid M^\rho(x, h) \geq 1 \}
\]
which is the same as before, except that the integrals are taken over \( \Omega \) instead of \( \mathbb{R}^n \).

1.3 Generalized mean curvature motion

The last extension of the BMO algorithm we will discuss is the scheme of Gribkovs and Heintz [15], that lets the boundaries of the sets move with a normal velocity \( v = g(x) \), where \( g \) is the mean curvature and \( y : \mathbb{R} \to \mathbb{R} \) is an increasing, continuous function.

The algorithm works for two different symmetric convolution kernels, \( \rho_1 \) and \( \rho_2 \), and given a set \( C \) we can define
\[
\mathcal{S}_i^\rho(C, h) = 1 \int_{\Omega} \rho_i^\rho(y, x) \, dy - \int_{\Omega} \rho_i^\rho(y, x) \chi_\Omega(y) \, dy \cdot \frac{1}{h} \int_{\Omega} \rho_i^\rho(y, x) \, dy
\]
for \( i = 1, 2 \). Now, a crucial part of all the proofs of convergence of these convolution-thresholding algorithms is an expansion of \( \mathcal{S}_i^\rho(C, h) \) in \( h \) of the form
\[
\mathcal{S}_i^\rho(C, h) = a_i \sqrt{h} \rho_0(x) + b_i \sqrt{h} \rho_0(x) + \eta_i(h)
\]
with
\[
a_i = \int_{\rho_0} \rho_i(y, 0) \, dy,
\]
\[
b_i = \int_{\rho_0} y \rho_i(y, 0) \, dy.
\]
Clearly, setting \( \mathcal{S}_i^\rho(C, h) = 0 \) gives \( e = \frac{a_i}{b_i} + d \sqrt{h} \), which corresponds to the original BMO algorithm.

4 Approximation of generalized mean curvature flow...

Now, using two convolution kernels \( (1,2) \) gives us two linear equations for \( e \) and \( \kappa \). Solving these, we get
\[
\begin{align*}
\varepsilon &= \frac{1}{\sqrt{b_0}} \frac{b_1}{b_0} \left( \frac{b_2}{b_0} + d \sqrt{h} \right) \\
\kappa &= \frac{1}{\sqrt{b_0}} \frac{a_1}{a_0} \left( \frac{a_2}{a_0} + d \sqrt{h} \right)
\end{align*}
\]
where \( d = a_0 b_0 \) is the determinant. Thus, since we want to have \( e = \kappa(a) \), we define
\[
F(N_1, N_2) = \varepsilon \cdot g(a) = \frac{1}{\sqrt{h}} \frac{b_1 N_1}{b_0} \left( \frac{b_2 N_1}{b_0} + d \sqrt{h} \right)
\]
and a thresholding scheme
\[
C_{k+1} = \{ x \in \mathbb{R}^n \mid F(N_1, N_2) \geq 0 \}
\]
for \( k \in \mathbb{N}^+ \).

In order for this scheme to converge to the actual generalized mean curvature motion, it turns out that \( F \) must satisfy the condition \( \partial F / \partial a_i > 0 \), \( i = 1, 2 \), which leads to the restrictions
\[
d = a_0 b_0 \quad a_2 b_0 \geq 0,
\]
\[
0 < \frac{b_1}{a_1} < d < \frac{b_2}{a_2}
\]
on \( g \) and \( N_1 \), meaning both that \( g \) must have bounded derivative both from above and below, and that given functions, the convolution kernels must be chosen with some care to fulfill the inequalities. If we would like to use a function \( g \) with unbounded or even derivative, it is possible to use uniform approximations \( N \) to \( g \) and still get a scheme that converges as \( h \to 0 \).

1.4 Outline

The structure of the following sections is as follows. First, in section 2 we give an introduction to viscosity solutions and give a background to the definition of solution and comparison results that are used in section 5.

Having established the necessary background, the actual treatment begins in section 3, where the algorithm is presented in more detail and all assumptions stated. In section 4, a few embeddings are proven. Then, finally, the last section sums it all up in the proof that the algorithm converges to the solution of the level set equation as the time step tends to zero.
2. Viscosity solutions

2.1. Introduction

The theory of viscosity solutions was developed during the 1980's by M.G. Crandall, L.C. Evans, H. Ishii, P.L. Lions and others while seeking solutions to the Hamilton–Jacobi equations. The name viscosity solutions originates from the method of “vanishing viscosity” which was used to solve first-order equations and which was consistent with the new theory being developed. Now, however, viscosity solutions do not generally have much to do with viscosity. The theory provides very general existence and uniqueness results and allows merely continuous functions to be solutions of fully nonlinear second-order equations. An excellent account of the theory may be found in the “user's guide to viscosity solutions” by Crandall, Ishii and Lions. Here, we give just an introduction to the theory and introduce some concepts which will be used later on.

2.2. Theory of viscosity solutions

The theory of viscosity solutions applies to equations of the form

\[ F(x, u, \nabla u, D^2u) = 0, \]

(2.1)

where \( x \in \mathbb{R}^n \), \( u = u(x) \) is a real valued function, \( \nabla u \in \mathbb{R}^n \) is its gradient and \( D^2u \in \mathcal{S}(n) \) the matrix of second derivatives of \( u \), \( \mathcal{S}(n) \) is the set of real symmetric \( n \times n \) matrices, which is partially ordered by the relation \( \preceq \), where \( Y \preceq X \) means \( \xi^T Y \xi \leq \xi^T X \xi \) for all \( \xi \in \mathbb{R}^n \). We also equip \( \mathcal{S}(n) \) with the norm \( \|Y\|_2 = \sqrt{\text{tr}(YY^T)} \) for all \( Y \in \mathcal{S}(n) \). Finally, \( F : \mathbb{R}^n \times \mathbb{R} \times \mathbb{R} \times \mathcal{S}(n) \rightarrow \mathbb{R} \) is a function, which may take any form.

For the theory to apply, we require \( F \) to satisfy the monotonicity conditions

\[ F(x, r.p.X) \leq F(x, s.p.X) \text{ if } r \leq s, \]

(2.2)

and

\[ F(x, r.p.Y) \leq F(x, r.p.X) \text{ if } Y \preceq X, \]

(2.3)

where \( r, s \in \mathbb{R} \), \( x \in \mathbb{R}^n \) and \( X, Y \in \mathcal{S}(n) \). If (2.3) holds, \( F \) is said to be degenerate elliptic. If (2.2) also holds, \( F \) is proper.

Now suppose that \( F \) is proper and that \( u \in C^2(\mathbb{R}^n) \) is a solution to \( F = 0 \), i.e., satisfies

\[ F(x, u(x), \nabla u(x), D^2u(x)) \leq 0 \]

for all \( x \in \mathbb{R}^n \). Choose a test function \( \psi \) that is also \( C^2 \), and suppose that \( u - \psi \) has a local maximum at \( x \). Then we have \( \nabla u \cdot \nabla \psi \leq 0 \) and \( D^2u \cdot \psi \leq 0 \), i.e., \( \nabla u(x) = \nabla \psi(x) \) and \( D^2u(x) \preceq D^2\psi(x) \), and by (2.3),

\[ F(x, u(x), \nabla u(x), D^2u(x)) \leq F(x, u(x), \nabla u(x), D^2u(x)) \leq 0, \]

(3.3)
2.3 Boundary conditions

which may be used to facilitate the use of definition 2.1. We also note that the semijets only depend on the set \( \Omega \) if \( \partial \Omega \) is smooth. If that is not the case, we may drop the subscript.

We also need to define the closure of the semijets for \( x \in \Omega \) as

\[
\bar{T}^+_x u(z) = \{ (p, X) \in \mathbb{R}^n \times \mathcal{S}(n) \mid (p, X) \in \mathcal{F}^+_x u(z) \} \cup \{ (p, X) \in \mathcal{S}(n) \mid (p, X) \in \mathcal{F}_y u(z) \}
\]

and note that if \( \mathcal{F}^+ u(x) \) is a semijet of \( F = 0 \) in \( \Omega \), then \( F(x, u(x), p, X) \leq 0 \) for \( x \in \partial \Omega \) and \( (p, X) \in \mathcal{F}^+_x u(z) \). If \( F \) is lower semicontinuous, this remains true if \( (p, X) \in \bar{T}^+_x u(z) \). Similar remarks are true for supersolutions and solutions.

2.3. Boundary conditions

Viscosity solutions also allow precise formulations of boundary conditions. Consider the boundary value problem

\[
\begin{align*}
F(x, u(x), \nabla u(x), D^2 u(x)) &= 0, & x &\in \Omega \\
B(x, u(x), \nabla u(x)) &= 0, & x &\in \partial \Omega
\end{align*}
\]

in an open set \( \Omega \subset \mathbb{R}^n \), where \( F \) and \( B \) are both proper functions. The correct definition of a viscosity solution of (2.9) is then

**Definition 2.2** A function \( u \in \text{LSC}(\Omega) \) is a sub-solution of (2.9) if

\[
\begin{align*}
F(x, u(x), p, X) &\leq 0, & x &\in \Omega, (p, X) &\in \bar{T}^+_x u(z) \\
B(x, u(x), p) &\leq 0, & x &\in \partial \Omega, (p, X) &\in \bar{T}^+_x u(z)
\end{align*}
\]

and a supersolution if

\[
\begin{align*}
F(x, u(x), p, X) &\geq 0, & x &\in \Omega, (p, X) &\in \bar{T}^+_x u(z) \\
B(x, u(x), p) &\geq 0, & x &\in \partial \Omega, (p, X) &\in \bar{T}^+_x u(z)
\end{align*}
\]

Finally, \( u \) is a solution if it is both a subsolution and a supersolution.

Here, \( u \in \text{LSC}(\Omega) \) and \( u \in \text{HSC}(\Omega) \), so what the definition basically means is that \( u \) is the boundary condition in the boundary condition or the equation should hold. That we cannot expect the boundary conditions to hold in a stronger sense is demonstrated by an example in [7].

8 Approximation of Generalized Mean Curvature Flow...

2.4 Singular equations

Since the equation we are interested in has a singularity for \( \nabla u = 0 \), we need to introduce a third definition of viscosity solutions.

To start with, given a function \( u : \Omega \to \mathbb{R} \) with \( \Omega \subset \mathbb{R}^n \), we introduce the upper and lower semicontinuous extensions

\[
\begin{align*}
u^+(x) &= \lim_{y \to x} \sup \{ y \in \Omega \mid y < x \}, \\
u_-(x) &= \lim_{y \to x} \inf \{ y \in \Omega \mid y > x \},
\end{align*}
\]

which are defined on \( \Omega \) and take values in \( \mathbb{R} \) or \( \mathbb{R} \cup \{ \infty \} \) respectively.

Also, supposing our function \( F \) is only defined on a dense subset \( \mathcal{W} \) of \( \mathcal{F} \) in \( \mathcal{F} = 0 \times \mathbb{R} \times \mathbb{R} \times \mathcal{S}(n) \), we may similarly define the relations \( F^+ \) and \( F^- \) on \( \mathcal{W} = \mathcal{F}_{\text{reg}} \) as

\[
F^+(x, u, \nabla u, p) = \lim_{y \to x} \sup \{ F(y, u(y), \nabla u(y)) \mid (y, \nabla u(y)) \in \mathcal{F} \}
\]

and

\[
F^-(x, u, \nabla u, p) = \lim_{y \to x} \inf \{ F(y, u(y), \nabla u(y)) \mid (y, \nabla u(y)) \in \mathcal{F} \},
\]

where \( (x, \nabla u(x)) \) is the star of \( x \) and \( \nabla u(x) \) is the star of \( \nabla u(x) \frac{1}{2} \). Then, we make the following definition of viscosity solutions:

**Definition 2.3** A function \( u : \Omega \to \mathbb{R} \) is a sub-solution of (2.9) if \( u^+ < \infty \) in \( \Omega \) and

\[
F^-(x, u, \nabla u, p) \leq 0 \text{ for all } x \in \Omega, (p, X) \in \bar{T}^+_x u(z),
\]

a supersolution if \( u^- > \infty \) in \( \Omega \) and

\[
F^+(x, u, \nabla u, p) \geq 0 \text{ for all } x \in \Omega, (p, X) \in \bar{T}^+_x u(z),
\]

and a solution if it is both a sub and a supersolution.

This definition is adapted from [8], where it is used for proving existence of solutions of curvature flow equations and similar.

2.5 Comparison

We finally make some comments on the method generally used for proving existence of viscosity solutions to nonlinear equations. In general, the method consists of three steps. The first two steps establish a comparison principle, i.e., if \( u \) is a sub-solution and \( v \) is a super-solution, then \( u \leq v \). From this they follow immediately that if \( v \) is a solution, it must be unique. It also follows that if \( u \) is a solution by the definition of viscosity solutions given above, then \( u \) is continuous. Furthermore, the comparison principle also provides a priori estimates for \( u \) in our proof of convergence in section 5.

The second step is to construct a subsolution and a supersolution. The third step is to use Perron's method to show that is that one, there exists a solution. See [10] for details. For applications of the method, see also the proofs of existence of solutions to level set equations for mean curvature motion, e.g., [9], [11], [30],
3. The algorithm

Now, having given the necessary background, we turn to the problem of extending the algorithm for approximating generalized mean curvature flow to the case of right-angle boundary conditions and prove its convergence as the time step tends to zero. For that purpose, let \( \Omega \) be an open, bounded domain in \( \mathbb{R}^n \) with \( C^2 \)-boundary \( \partial \Omega \). Given \( u_0 \in C(\Omega) \), we consider the level set equation

\[
\begin{align*}
\frac{\partial \phi}{\partial t}(x, t) + \nabla \phi(x, t) \cdot \mathbf{N}(\phi(x, t)) &= 0, & x &\in \Omega, \ t \in (0, T) \\
\frac{\partial \phi}{\partial t}(x, t) &= 0, & x &\in \partial \Omega, \ t \in (0, T) \\
u(x, 0) &= u_0(x), & x &\in \Omega
\end{align*}
\]

for \( T > 0 \), where

\[
\text{curve}(u(x)) = \int_{u(0)}^{u(x)} \left( \nabla \phi(z) - \frac{1}{\sqrt{\det(\nabla \phi(z)^T \nabla \phi(z))}} \sum_{i,j} \left( \begin{array}{c}
u_{x_i}(z) \\
u_{x_j}(z) \end{array} \right) \right) du(z)
\]

is \( n \)-times the mean curvature of the level set of \( u \) passing through the point \( x \), \( \nu \) is the outward unit normal to \( \Omega \) and \( g : \mathbb{R} \to \mathbb{R} \) fulfills the conditions

(i) \( g \in C(\mathbb{R}) \), \( g(0) = 0 \)

(ii) \( g(x) = O(x) \) as \( x \to \pm \infty \)

(iii) \( g \) is increasing.

The PDE describes a function whose level sets \( \{ x \in \mathbb{R}^n \mid g(x) = \lambda \} \) move with normal velocity \( \sqrt{\text{tr}((\nabla g(x))^2)} \) and intersect \( \partial \Omega \) at a right angle, at least formally. The equation is degenerate parabolic and has singularities for \( \nabla \phi = 0 \), but in spite of these difficulties, Satô [20] showed that if \( \Omega \) is convex, the equation has a unique viscosity solution in \( C(\overline{\Omega}, D) \) for any \( T > 0 \). Furthermore, if \( g \) is linear, Giga and Satô [11] proved that there is a unique viscosity solution even if \( \Omega \) is not convex. Since we are interested in nonlinear functions \( g \), we need the additional assumption that

\[ \Omega \text{ is convex.} \]

It should be noted, however, that we only use this assumption through the use of the comparison principle from Satô [20]. So if a proof of existence of unique solutions is conducted for the case of a nonconvex \( \Omega \), the proof of convergence of the algorithm will be valid for this case also.

Because of (1.4), the convergence of the algorithm also requires that

(i) \( g \in C^1(\mathbb{R}) \) and \( \frac{\partial g}{\partial \nu} \geq 0 \) for \( \nu \in \mathbb{R} \) \( g(x) \in (\bar{\alpha}, \bar{\beta}) \),

(ii) \( \frac{\partial g}{\partial \nu} \geq 0 \)\( \nu \in \mathbb{R} \) \( g(x) \in (\bar{\alpha}, \bar{\beta}) \)

which is more implicit (ii) and (iii). But we will then also show how to get around this problem if we can find uniform approximations \( g_\eta \to g \), where \( g \) fulfills (3.3) and has unbounded or zero derivative and \( \partial \phi \) fulfills (3.3) and (3.4) for all \( \eta \).
for $i = 1, 2$. This notation does not explicitly show the dependence on $x_1$, $x_2$, and $y$, but that will be clear from the context. Finally, we also define mappings $G_0, \tilde{G}_0 : C(\overline{\Omega}) \rightarrow C(\overline{\Omega})$, corresponding to $G_0$, by
\[
\tilde{G}_0 \psi (x) = \max \{ \lambda \in \mathbb{R} \mid F(N_i(\lambda), N_j(\lambda)) \geq 0 \} \tag{3.8}
\]
\[
G_0 \psi (x) = \min \{ \lambda \in \mathbb{R} \mid F(N_i(\lambda), N_j(\lambda)) \leq 0 \} \tag{3.9}
\]
for $h > 0$ and $\psi \in C(\overline{\Omega})$. Note that all symbols with tilde (\tilde{G}, \tilde{N}, etc.) denote entities in the case of an end boundary condition, while the same symbols without the tilde denote the same entity in the domain $\Omega$.

The main result, Theorem 3.1, is now that the repeated application of the mappings $G_0$ gives us an approximation of the solution to the level set equation (3.1). For clarity, we state the algorithm explicitly:

**Algorithm 3.1**

Given $\Omega$ and $\Psi$, choose functions $\rho_1$ and $\rho_2$ according to the assumptions above.

Choose an initial set $C_0$.

For each iteration $k$:

1. Choose an timestep $t_k$.

2. For each point $x \in C_k$,
   - Calculate $N^{\Psi}(x, \rho_1, \rho_2)$, $i = 1, 2$.
   - Evaluate the function $F(N^{\Psi}_i(x, \rho_1, \rho_2), N^{\Psi}_j(x, \rho_1, \rho_2))$ if $F \geq 0$, let $z$ belong to $C_k$.
   - End loop.

3. End loop.

For efficient implementation of this and other BMO-type algorithms, see Rachel [10] and Grabowski [11].

4. **Properties of $G_0$ and $\tilde{G}_0$**

In this section, we prove some crucial properties of the operators $G_0$ and $\tilde{G}_0$. We start with the inclusion principle for $G_0$.

**Proposition 4.1** Let $G_0$ be defined by (3.8). Then for all $h > 0$ and all closed sets $C, C_0 \subset \overline{\Omega}$, we have

$C \subset C_0 \Rightarrow \tilde{G}_0(C) \subset \tilde{G}_0(C_0)$.

**Proof.** Since the weight functions $\rho_i$ are positive, $C \subset C_0$ implies that $N^{\Psi}_i \leq N^{\Psi}_j$, $i = 1, 2$ and since $F$ is increasing in both arguments, we have $F(N^{\Psi}_i, N^{\Psi}_j) \leq F(N^{\Psi}_i, N^{\Psi}_j)$ and therefore

$\{F(N^{\Psi}_i, N^{\Psi}_j) \geq 0\} \subset \{F(N^{\Psi}_i, N^{\Psi}_j) \geq 0\}$

and $\tilde{G}_0(C) \subset \tilde{G}_0(C_0)$.

From this principle, some properties of $G_0$ follow. Note that by the definitions of $G_0$ and $\tilde{G}_0$, the connection between the two is

$[G_0 \psi](x) = \min \{ \lambda \in \mathbb{R} \mid x \in \tilde{G}_0(\psi \geq \lambda) \}$.

**Proposition 4.2** For all $h > 0$ and $u, u_1, u_2 \in C(\overline{\Omega})$,

(i) $G_0(u + C) = G_0u + C$, for all $C \in \mathbb{R}$;

(ii) $G_0(\theta u) = \theta G_0u$, for any increasing, continuous function $\theta : \mathbb{R} \rightarrow \mathbb{R}$;

(iii) If $u(z) \leq u(x)$ for all $x \in C$, then $[G_0u](z) \leq [G_0u](x)$ for all $x \in C$;

(iv) $[G_0u] \leq \max \{ u_1, u_2 \}$ in sup norm;

(v) $G_0u = \inf \{ \lambda \in \mathbb{R} : F(N_i(\lambda), N_j(\lambda)) \leq 0 \}$.

**Proof.**

(i) This follows directly from the definition (3.8) of $G_0$.

(ii) This also follows from the definition (3.8) of $G_0$, since $\theta$ commutes with the taking of supremum.

(iii) Suppose $u_1 \leq u_2$ in $\overline{\Omega}$ and that there is an $x_1 \in \overline{\Omega}$ such that $[G_0u_1](x_1) > [G_0u_2](x_1)$. Let $\lambda = [G_0u_2](x_1)$ and $\lambda_2 = [G_0u_3](x_1)$.

$\lambda = \inf \{ \lambda \in \mathbb{R} \mid u \in \tilde{G}_0(\psi \geq \lambda) \}$.

$\lambda_2 = \inf \{ \lambda \in \mathbb{R} \mid u \in \tilde{G}_0(\psi \geq \lambda) \}$.

By (4.1)

$\lambda_1 \geq \lambda_2$.

Then, for every $\varepsilon > 0$, since also $u_1 \leq u_2$,

$[u_1 \geq \lambda_1 - \varepsilon] \subset [u_2 \geq \lambda_1 - \varepsilon] \subset [u_2 \geq \lambda_2 - \varepsilon]$. 

Proposition 4.3. For all \( \varphi \in C^1(\Omega) \), \( z \in \Omega \) and \( \varepsilon > 0 \), there is a \( \delta > 0 \) such that

(i) If \( \varphi \in C^2(\Omega) \) and \( \varphi(z) \neq 0 \), then

\[
|G_{\varphi}(z)| \leq \varepsilon + C(\varphi) |\nabla \varphi(z)| + \frac{1}{h} \left| \frac{\partial \varphi}{\partial n}(z) \right| \quad \text{if } z \in B_{\delta}(z, \delta) \cap \Omega, \quad \delta > 0.
\]

(ii) If \( \varphi \in C^1(\Omega) \) and \( \varphi(z) > 0 \), then \( |G_{\varphi}(z)| \leq |\nabla \varphi(z)| + \frac{1}{h} \left| \frac{\partial \varphi}{\partial n}(z) \right| \) holds for all \( z \in B_{\delta}(z, \delta) \cap \Omega \) and \( \delta > 0 \).

The proof uses the same ideas as [14], namely to construct one to the problem without boundary conditions in the whole of \( \mathbb{R}^n \). The analogue of Proposition 4.3 for that one can be found in a different shape in Grisvard and Heinz [12, Lemma 2] and is stated as Lemma 4.4 below. The set \( \Omega \) is mainly concerned with comparing \( N(\lambda) \) and \( \overline{N}(\lambda) \) (with and without boundary conditions). This requires the rather lengthy proofs of lemmas 4.5 and 4.6. Lemma 4.5 is proved exactly as in [14, Lemma 3.1, p. 118] and the proof is therefore not given here. The setup for that proof is identical to the one in the proof of Lemma 4.6, which is given here and which we need because we have two convolution kernels instead of one.

Lemma 4.4. (see [12, Lemma 2]) Let \( \varphi \in C^1(\Omega) \) and \( z \in \mathbb{R}^n \). If \( \nabla \varphi(z) \neq 0 \), then for each \( \varepsilon > 0 \), there is a \( \delta > 0 \) such that

\[
|G_{\varphi}(z)| \leq \varepsilon + C(\varphi) |\nabla \varphi(z)| + \frac{1}{h} \left| \frac{\partial \varphi}{\partial n}(z) \right| \quad \text{if } z \in B_{\delta}(z, \delta) \cap \Omega, \quad \delta > 0.
\]

Proof of proposition 4.3. As in [14], the main idea behind the proof is to compare \( G_{\varphi}(z) \) to \( G_{\varphi}(z) \) and use Lemma 4.4 to get the desired result.

Let \( \varphi \in C^1(\Omega) \) and \( \varepsilon > 0 \). If \( \varphi(z) = 0 \) and \( \varphi'(z) \neq 0 \), then since \( \varphi(z) \) is compact, there is a \( \delta > 0 \) such that for all \( z \in B_{\delta}(z, \delta) \cap \Omega \), we have

\[
|G_{\varphi}(z)| \leq \frac{1}{\varepsilon} \varphi(z).
\]

So Proposition 4.3, part (i), follows directly from Lemma 4.4. To prove the rest of the proposition, we assume \( \varphi \neq 0 \) and \( \varphi'(z) \neq 0 \) and prove part (ii), setting that part (ii) may be proved similarly.

Since \( \varphi \in C^1(\Omega) \) and \( \nabla \varphi \in C^1 \), we may extend \( \varphi \) to \( \varphi \in C^1(\overline{\Omega}) \) for some \( \varepsilon > 0 \). Using Proposition 4.2 (i), we also set

\[
\lambda = \frac{1}{2} \left( \frac{1}{\varepsilon} \frac{1}{h} \right),
\]

Finally, the following proposition, which is analogous to Lemma 3.1 in [14], is a crucial part of the proof of our main theorem,
4. Properties of $G_{\theta}$ and $\mathcal{G}_{\theta}$

Figure 4.1: The sets $Q_+^i$ and $Z_+^i$ in case 1.

and

$$\lambda = \inf \{ \rho \in \mathbb{R} \mid F(N_1(p), N_2(p)) \leq 0 \}.$$  

Now, we wish to show that there is an $h > 0$ such that $\lambda \leq 0$ for each $x \in B_R(z, r) \cap \Omega$ and each $h \in (0, h'_i)$. By the definition of $\lambda$, we know that $F(N_1(p), N_2(p)) = 0$ and by the definition of $\lambda$, we also know that $F(0, 0) = 0$. If we can show that $N_1(p) \leq 0$, $i = 1, 2$, we would know that $F(N_1(p), N_2(p)) \leq 0$ (since $F$ is increasing in both variables) and thus

$$\lambda = \inf \{ \rho \in \mathbb{R} \mid F(N_1(p), N_2(p)) \leq 0 \} \leq \lambda,$$  

(4.4)

The same result would be obtained if we could show that $N_1(p) \leq N_2(p)$, $i = 1, 2$.

To prove that $N_1(p) \leq 0$ or that $N_1(p) \leq N_2(p)$, we define

$$Q_+^i = \{ \rho \in B_R(z, r) \mid \rho(x) < \lambda \cap \Omega, \quad \rho \geq 0 \} \quad \text{for } i = 1, 2,$$

with $\lambda_0 = \inf \{ \rho \in \mathbb{R} \mid \sup \rho \subset B_R(0, |x|) \}$. See figure 4.1. We then consider one $A$ at a time ($i = 1$ or 2) and divide the treatment into three cases (cf. figures 4.1 and 4.2).

---

16. Approximation of generalized mean curvature flow...

Figure 4.2: Case 2 on the left and case 3 on the right.

Case 1: $p \in B_R(z, r) \cap \Omega \setminus \{ \varphi = \lambda \}$.

In general, we have

$$N_1(p) = \int_{Q_+} \rho_1(p) \rho_2(p) \, dV - \frac{1}{2} \int_{Q_+} \rho_1(p) \rho_2(p) \, dV$$

$$= \int_{Q_+} \rho_1(p) \rho_2(p) \, dV \leq \frac{1}{2} \int_{Q_+} \rho_1(p) \rho_2(p) \, dV$$

Lemma 4.3 below now shows that $N_1(p) \leq 0$ and lemma 4.6 gives $N_1(p) \leq N_2(p)$.

Case 2: $p \in B_R(z, r) \cap \Omega \setminus \{ \varphi = \lambda \} \subset \Omega$.

In this case, the intersection of the hyperplanes $\{ \varphi = \lambda \}$ and $\Omega$ lies outside the ball $B_R(z, r)$. We then see that $Q_+ \cap \Omega = \emptyset$ and therefore, starting as before, we get

$$N_1(p) = \int_{Q_+} \rho_1(p) \rho_2(p) \, dV \leq \frac{1}{2} \int_{Q_+} \rho_1(p) \rho_2(p) \, dV$$

Case 3: $p \in B_R(z, r) \cap \Omega \setminus \{ \varphi = \lambda \} \subset \Omega'$.

In this case, $Q_+ = \Omega'$ and since $A$ is positive, we can conclude $N_1(p) \leq 0$ from the expression in Case 1.
4. Properties of $G_R$ and $\vartheta_R$

Now we need to see what happens when we have two convolution kernels. Let $\gamma = \inf (r \in R \mid \varphi \equiv 1 \cdot \gamma_1 \cdots \gamma_n)$. Then $\gamma_1 \cdots \gamma_n = 1$ and we have $\gamma_1 < \gamma_2$. The $\gamma_i$ is not applied to $\gamma_1$, then it obviously also applies to $\gamma_2$, so that $\gamma_2(\gamma_1) \leq 1$ and therefore $\gamma_1 \leq \gamma_2$ by the argument leading to (4.4). If $\gamma_2$ applies to $\gamma_3$, then either one or both apply to $\gamma_4$, then both apply to $\gamma_5$, and both apply to $\gamma_6$. Finally, if $\gamma_2$ applies to $\gamma_3$, then either one or both apply to $\gamma_4$, then both apply to $\gamma_5$, and both apply to $\gamma_6$. Therefore, since we prove Lemma 4.5 and 4.6, the proof is completed.

All that now remains is to prove the next two lemmas. As mentioned earlier, lemma 4.5 is proved in [14], for regular mean curvature flow and there are only very minor differences in our case, so we omit the proof. Beside, the same setup is used in the proof of Lemma 4.6, which we give in detail and which is needed in our case in view of the discussion following the three cases above, because we have two convolution kernels.

**Lemma 4.5** (c, [14, Lemma 3.1], case 1)
Let $\gamma, Q_1, Q_2$ be defined as above, and assume that $\gamma \in \Omega$ and $\varphi_0(\partial \Omega) = 0$, then there is a $\gamma_2 > 0$ and an $h_3 > 0$ such that for all $\gamma \in \Omega$ and $h \in (0, h_3)$, it holds that if $(\gamma_1 \cdots \gamma_n) \neq \varphi \neq \varphi_0 \neq 0$ and $0 \leq h \leq h_3$, then

\[
\int_{Q_0} \varphi^N Q_0 \gamma \, dx \leq \int_{Q_0} \varphi^N(\gamma \, dx),
\]

and if $\gamma = \gamma_1(\gamma_2 = 0$).

**Lemma 4.6** Let $\gamma, Q_1, Q_2$ be defined as above, and assume that $\gamma \in \Omega$ and $\varphi_0(\partial \Omega) = 0$, then there is a $\gamma_2 > 0$ and an $h_3 > 0$ such that for all $\gamma \in \Omega$ and $h \in (0, h_3)$, it holds that if $(\gamma_1 \cdots \gamma_n) \neq \varphi \neq \varphi_0 \neq 0$ and $0 \leq h \leq h_3$, then

\[
\int_{Q_0} \varphi^N(\gamma \, dx) \leq \int_{Q_0} \varphi^N(\gamma \, dx),
\]

with $Q_0$ defined above and

\[
Q_1 = \{ \varphi \cdots \gamma_1(\gamma_2 = 0 \mid \varphi \neq \varphi_0 \neq 0)
\]

so that $Q_1 = Q_1 \cap \Omega$. Rewriting this expression once more, we get

\[
\int_{Q_0} \varphi^N(\gamma \, dx) \leq \int_{Q_0} \varphi^N(\gamma \, dx),
\]

and define the hypersurfaces $A$ and $B$ by

\[
A = \{ \varphi \in \Omega \mid (\gamma_0, 1, \gamma_2(\gamma_2 = 0) - 0)
\]

and

\[
B = \{ \varphi \in \Omega \mid (\gamma_0, 1, \gamma_2(\gamma_2 = 0) - 0) \mid \varphi \in \Omega \mid (\gamma_0, 1, \gamma_2(\gamma_2 = 0) - 0)
\]

with $\gamma = (\gamma_0, 1, \gamma_2(\gamma_2 = 0)$, see Figure 4.3. We also note that $\gamma \leq \gamma \gamma_2(\gamma_2 = 0$ and $\gamma \leq \gamma_2(\gamma_2 = 0$ such that $0 \leq h \leq h_3$ and $x \in B_3(\gamma_2, \delta)$ such that if $h \in (0, h_3)$ and $x \in B_3(\gamma_2, \delta)$, then $|\varphi(x)| \leq \gamma_2(\gamma_2 = 0$.

**Lemma 4.7** (c, [14, Lemma 3.4])
There exists a $\delta > 0$ and a $C_5 > 0$ independent of $x \in B_3(\gamma_2, \delta)$ such that if $h \in (0, h_3)$ and $x \in B_3(\gamma_2, \delta)$, then $|\varphi(x)| \leq \gamma_2(\gamma_2 = 0$.
Approximation of generalized mean curvature flow...

By lemma 4.8, we have
\[ \| \psi \| \leq C_I h \]
if \( \psi(\theta) \neq \lambda \), \( \theta \in \partial B(\sqrt{2}, r) \),
(4.4)
and since \( \psi \in C^2(\mathbb{R}^3) \), we also have
\[ \| \psi \| \leq C_I h \]
for some \( C_I > 0 \), independent of \( z \in B(\pi, r) \) and \( h \in (0, h_0) \). We now restrict our view to the plane spanned by \( \{x, y, 1\} \) and \( \{x, y, 2\} \), which is obviously perpendicular to both \( A \) and \( B \). This is the view shown in figure 4.3.

We set
\[ v = z + \beta \{x, y, 1\} + \beta_1 \{x, y, 2\} \]
with
\[ \beta_1 = \frac{(1 + |\mathbf{v}|)^{3/2}}{D^{3/2}} \sqrt{\frac{\mathbf{v} \cdot \mathbf{v}}{|\mathbf{v}|}} C_I h \]
\[ \beta = \frac{1}{(1 + |\mathbf{v}|)^{3/2}} \sqrt{\frac{\mathbf{v} \cdot \mathbf{v}}{|\mathbf{v}|}} C_I h \]
defined so that \( v \) is located as in figure 4.3, that is \( \{0, 1, v \} \) and \( \{0, 2, v \} \) are \( \{0, 1, v \} \) and \( \{0, 2, v \} \), respectively.

This reduces the expression, it is easily seen that \( D \geq \beta \) for \( |\mathbf{v}| \), so that \( \beta \) is zero only when \( a = 0 \). Assume for the moment that \( a \neq 0 \), so that \( D \neq 0 \) and \( \beta \) is well-defined.

Now,
\[ \beta_1 = \frac{(1 + |\mathbf{v}|)^{3/2}}{D^{3/2}} \sqrt{\frac{\mathbf{v} \cdot \mathbf{v}}{|\mathbf{v}|}} C_I h \]
\[ \beta = \frac{1}{(1 + |\mathbf{v}|)^{3/2}} \sqrt{\frac{\mathbf{v} \cdot \mathbf{v}}{|\mathbf{v}|}} C_I h \]

(Remark that the quantity on the left is the orthogonal distance from the origin to the hyperplane \( \{x, y, 1\} \) = 0.)

Proof. The proof is given in [14] and is a straightforward geometrical argument.
4. PROPERTIES OF $G_h$ AND $\delta_h$

For the second term, we may now use (47) to get the estimate

$$\frac{1 + (\rho, \theta)}{(1 + |F|^2)^{3/4}} C_{h} \leq \frac{3}{2} C_{h}$$

Thus,

$$\frac{\theta h}{\sqrt{1 + |F|^2}} \leq \frac{g}{2} \leq \frac{\theta h}{2^{1/2} \sqrt{1 + |F|^2}} < h \sqrt{1 + |F|^2} \quad (4.10)$$

if $h \in (0, b_1)$ with $b_1 = (\theta (1 + \theta_0)/(3 C_{G}))^{3/2}$. Also, we note that $||F|| \leq \theta h$. We note that

$$\delta h \sqrt{1 + |F|^2} \leq \frac{g}{2} \leq \frac{\delta h}{2^{1/2}} < h \sqrt{1 + |F|^2} \quad (4.11)$$

if $h \in (0, b_2)$ with $b_2 = (\theta (1 + \theta_2)/(3 C_{G}))^{3/2}$. Therefore, we set $0 = \max(b_1 + 1, \theta_1 + 1)/2 < 1$ and $b_2 = \min(\theta (1 + \theta_2)/(3 C_{G}))^{3/2}, (\theta (1 + \theta_2)/(3 C_{G}))^{3/2}$.

We are now ready to estimate the integrals over $Z^+$ and $Z$ in (4.6). Letting $\delta_h$ denote the unit vector in the $\theta (1, 0)$-plane with $(\theta (1, 1)) = 0$ and $(\theta (1, 1)) > 0$, we define

$$W = \{ \rho \in B_h(z, \delta_h \sqrt{1 + |F|^2}) \}$$

$T_1 = \{ \rho \in B_h(z, \delta_h \sqrt{1 + |F|^2}) \}$

$T_2 = \{ \rho \in B_h(z, \delta_h \sqrt{1 + |F|^2}) \}$

(see figure 3) and note that

$$(z_0 + \{ \rho \in B_h(z, \sqrt{1 + |F|^2}) \}) \setminus \{ \rho \in B_h(z, \delta_h \sqrt{1 + |F|^2}) \} \supset T_1 \cup W,$$

$$(z_0 + \{ \rho \in B_h(z, \sqrt{1 + |F|^2}) \}) \setminus \{ \rho \in B_h(z, \delta_h \sqrt{1 + |F|^2}) \} \supset T_2 \cup W,$$

$$(z_0 + \{ \rho \in B_h(z, \sqrt{1 + |F|^2}) \}) \setminus \{ \rho \in B_h(z, \delta_h \sqrt{1 + |F|^2}) \} \supset (T_3 \cup W).$$

The definition of $\delta_h$ ensures that $W \neq \emptyset$. From this, it is clear that

$$\int_{\rho} \rho^2 \tau_0 \, d\rho \geq \int_{\rho} \rho^2 \tau_0 \, d\rho \geq 2 \int_{\rho} \rho^2 \tau_0 \, d\rho \geq \left( \int_{\rho} + \int_{\rho} \right) \rho^2 \tau_0 \, d\rho$$

(4.12)

Now, we see from (4.10) and (4.11) that

$$\mathcal{L}^n(W) = \alpha h^{n+1},$$

for some $\alpha > 0$ depending only on $\theta$ and $n$, where $\mathcal{L}^n$ is the $n$-dimensional Lebesgue measure. From this we conclude that

$$\int_{\rho} \rho^2 \tau_0 \, d\rho \geq C_0,$$

for some $C_0 > 0$, independent of $x \in B_h(z, r_0)$, $\varepsilon \in (0, c_1)$, and $h \in (0, b_0)$.

Furthermore, it is clear that $\mathcal{L}^n(T_3) \leq C_1 h^{n+1}$ for some $C_1 > 0$ depending only on $C_0, n$ and $\varepsilon$, and that $\mathcal{L}^n(T_3) \leq C_0 h^{n+1}$ for some $C_1$ depending only on $C_0, n$ and $\varepsilon$. Then, by changing variables $(\rho, \theta)\sqrt{1 + |F|^2} \rightarrow \theta$, we see

$$\int_{\rho} \rho^2 \tau_0 \, d\rho = \left( \int_{\rho} + \int_{\rho} \right) \rho^2 \tau_0 \, d\rho \geq C_0,$$

for all $x \in B_h(z, r_0)$, which proves $N(h) \leq K(h)$ by (4.12) and the discussion leading to (4.6).

Finally, we need to cover the case $\alpha = \theta$. In this case, we define

$$W = \{ \rho \in B_h(z, \delta_h \sqrt{1 + |F|^2}) \},$$

with $W \neq \emptyset$ if $h$ is small enough. Then $Z \supset W \cup T_1$ and $Z^+ \supset T_2 \cup T_3$, and the same argument holds.
5. The convergence theorem

In this section, we prove the convergence of the output of algorithms 3.1 to the viscosity solution of the level set PDE (3.1) at the step k tends to zero. The proof is based on the proof of Ishii and Ishii [9] and Ishii [13].

We begin by defining the approximations $u^m$ as follows. Given a function $f \in C(\mathbb{R})$, let $u^m \in C(\mathbb{R} \times [0, T])$ be defined for $m \in \mathbb{Z}_+$ by

$$ u^m(x,t) = \mathcal{G}_t \ast f(x), $$

where $h = T/m$ and $t \in \mathbb{N}$ is chosen so that $\varepsilon < t < (1 + 1/h)$. The main convergence theorem is then the following.

Theorem 5.1 Choose $f \in C(\mathbb{R})$ and let $\{u^m\}_{m=1}^{\infty}$ be defined by (5.1). Then $u^m \to u$ locally uniformly on $\mathbb{R} \times [0, T]$ as $m \to \infty$, where $u$ is the unique viscosity solution of the PDE

$$
\begin{align*}
\frac{\partial u}{\partial t} &+ \mathcal{L} u = 0, & x \in \mathbb{R}, t \in (0, T), \\
\frac{\partial u}{\partial x} &+ \mathcal{L} u = 0, & x \in \partial \Omega, t \in (0, T), \\
\frac{u(x, 0)}{\partial x} &+ \mathcal{L} u = f(x), & x \in \mathbb{R},
\end{align*}
$$

which exists by Theorem 3.12 in Satô [20].

The idea of the proof is to define $\sigma(x, t)$ and $\psi(x, t)$ by

$$
\begin{align*}
\sigma(x, t) &= \lim_{m \to \infty} \sup \{u^m(y, s) \mid m > r, (y, s) \in \mathbb{R} \times [0, T), t + \varepsilon < s < t, \\
\psi(x, t) &= \lim_{m \to \infty} \inf \{u^m(y, s) \mid m > r, (y, s) \in \mathbb{R} \times [0, T), t + \varepsilon < s < t,
\end{align*}
$$

and prove that these are sub and supersolution respectively of the level set PDE. It then follows from the comparison result by Satô [20] (Theorem 5.2 below) that $\sigma \leq u$ and that (since $\sigma \leq u$ by definition) that $u \geq \psi$ is a solution.

First, we note the definition of a viscosity solution to the PDE (5.1) is in accordance with the definitions given in Section 2. However, it is convenient to give the definition using test functions instead of supremum for $\sigma(x, t)$ as follows.

Definition 5.1 A function $u \in C(\mathbb{R} \times [0, T])$ is a viscosity sub/solution of (3.1) if for any $\varphi \in C^{1, 2}(\mathbb{R} \times [0, T])$ such that $\varphi \leq u$ [on a domain $\mathbb{R} \times [0, T]$] then

$$
\varphi(x, t) \leq u(x, t).
$$

We fix $x \in \mathbb{R}$, $h > 0$, and $x_0 \in B_h(x, \delta) \subset B_h(x, \varepsilon)$. Therefore we investigate the situation in Figure 5.3, where $\varepsilon$ is the level set of $u$ at $x \in \mathbb{R}$ and $\delta > 0$. Hence, in order to estimate $|G_0(x, \varepsilon)|$, we need to estimate $\|\mathcal{L} u\|_{L^\infty}$ for $x \in B(h, \delta)$. We use the supersolution $u \geq \psi$ in both cases 1 and 2. Therefore we have $\psi \leq 0$ or $\psi \geq 0$. Hence, we need to have either $\psi \leq 0$ or $\psi \geq 0$ in order to get $u(x) \geq 0$.

Theorem 5.2 [20, Theorem 2.1]

We have the following result:

**Lemma 5.3** For each $x \in \mathbb{R}$, there is a $C > 0$, $\delta > 0$, and $h_0 > 0$ such that

$$
|G_0(x, \varepsilon)| \leq C \varepsilon + Ch,
$$

for all $x \in B_h(x, \delta)$. Then

$$
|G_1(x, \varepsilon)| \leq C \varepsilon + Ch
$$

for all $x \in B_h(x, \delta)$.
5. The convergence theorem

\[ G_h(f) \leq f(z) + C h, \quad \text{for all } z \in \mathcal{B}, h \in (0, h_1), \] (5.3)

by the Poincaré theorem, which gives the desired result with \( C = R^2 \).

A similar argument gives the result for \( |z| < 4 \).

\[ \|u^w(\cdot, \cdot)\|_{L^\infty} \leq C(1, t), \]
\[ \text{for all } t \in [0, T], \text{ if } \partial f/\partial h > 0 \text{ on } \partial \Omega, \]
\[ \text{then } \inf_{x \in \partial \Omega, t \in [0, T]} \|u^w(x, t)\|_{L^\infty} \geq C(1, t), \]

\[ \text{Proof.} \] We assume \( \partial f/\partial h > 0 \) on \( \partial \Omega \) and prove the first inequality. The other may be proved similarly.

We prove that there is a constant \( C > 0 \) and an \( h_1 > 0 \) such that

\[ G_h(f(z)) \leq f(z) + C h, \quad \text{for all } z \in \mathcal{B}, h \in (0, h_1), \] (5.3)

which, if \( M \) is chosen so that \( T/M \leq h_2 \), may be assembled to give the desired result.

We fix \( z \in \mathcal{B} \). If \( \nabla f(z) \neq 0 \), it follows from proposition 4.3 with \( c = 1 \) that there is a \( h_1 > 0 \) such that

\[ G_h(f(z)) \leq f(z) + (|\nabla f(z)| + 1) h \]

for all \( z \in \mathcal{B}, h \in (0, h_1) \).

Now suppose that \( \nabla f(z) = 0 \). Since we assume \( \partial f/\partial h > 0 \) on \( \partial \Omega \), we then have \( z \in \partial \Omega \).

Since \( f \in C^1(\mathcal{B}) \), there is a \( h_1 > 0 \), such that

\[ |f(z)| \leq |D^2 f|_{\partial \Omega} |z| \]

for all \( z \in \mathcal{B}, h \in (0, h_1) \). Taking \( C_1 = 2 |f|_{\partial \Omega} + |D^2 f|_{\partial \Omega} \), we then get

\[ f(z) \leq f(z) + C_1 |z| \]

for all \( z \in \mathcal{B}, h \in (0, h_1) \).

That is, for any \( z \in \partial \Omega \) and any \( \varepsilon > 0 \), there are \( C > 0 \) and \( h_2 > 0 \) independent of \( \varepsilon \) and \( \delta > \delta(\varepsilon) > 0 \) such that

\[ G_h(f(z)) \leq f(z) + C h + \varepsilon \]

(5.3)

for all \( z \in \mathcal{B}, h \in (0, h_1) \). Since \( \partial \Omega \) is compact, it may be covered by a finite number of such neighborhoods and taking the largest of the \( C_k \) and the smallest of the \( h_k \), (5.3) holds for all \( z \in \partial \Omega \) and \( h \in (0, h_2) \). Since \( \varepsilon \) is arbitrary, we get (5.3).

\[ \text{Proof.} \] Take a subsequence \( \{f_k\} \in C^1(\mathcal{B}) \) satisfying

\[ |f_k|_{\partial \Omega} \leq \frac{1}{k}, \quad \frac{\partial f_k}{\partial \nu} > 0 \text{ on } \partial \Omega \]

and define \( u^w_{(\cdot, \cdot)} \) as in (5.1) with \( f_k \) instead of \( f \).

Then, by Lemma 5.4, there are constants \( C_k \) such that for each \( k \in \mathbb{Z}^+ \),

\[ u^w_{(\cdot, \cdot)}(f_k) \leq C_k \]

for all \( z \in \mathcal{B} \) and \( t \in (0, T) \). Because of proposition 4.2 (v) and since \( |f_k|_{\partial \Omega} \leq 1/k \), it follows that

\[ |u^w_{(\cdot, \cdot)}(f_k)| \leq \frac{1}{k} \]

Approximation of Generalized Mean Curvature Flow...
and thus
\[ u^m(t, \zeta) - f(\zeta) \leq C|x|^2 \]
(5.6)
for all \( y \in \Omega, t \in [0, T] \) and \( x \in Z^m \).

For any fixed \( x \in \Omega \), we may now let \( m \to \infty, \tau \to x, \epsilon \to 0 \) and finally \( h \to \infty \) to conclude by the definition (5.7) of \( \pi \) that
\[ \pi(x, 0) \leq f(x), \]
Since \( \pi(\zeta, 0) \geq f(\zeta) \) by a similar argument and \( \pi(x, 0) \leq \pi(x, 0) \) by definition, the result follows.

Proof of Theorem 5.1. As already mentioned, the idea of the proof is to show that \( \pi(x, \zeta) \) (defined by (5.2)) is a subproblem and \( \pi(x, \zeta) \) a subproblem of the level set PDIE (3.1). It then follows from Lemma 5.3 and the comparison result by Saxo [25] (Theorem 2.2 in [25]) that \( \pi(x, \zeta) \leq \pi(x, \zeta) \) for all \( x, \zeta \in \Omega \times [0, T] \) and thus \( u = \pi = \tilde{u} \) is a viscosity solution of (3.1). By an argument in Chapter 6 of Caffarelli, Ishii, Lions [7], it also follows that \( u \) is locally uniformly convex.

We prove that \( \pi(x, \zeta) \) is a subsolution and note that the proof of that \( \pi(x, \zeta) \) is a supersolution is similar.

So, we fix a function \( \psi \in C^1(\Omega \times [0, T]) \) and assume that \( \pi \psi \) has a strict maximum at \((x_0, \zeta_0)\). Since the local behavior of \( \psi \) is important, we may assume that this maximum is global. Also, since we are only interested in \( \partial_t \pi \psi \) and \( \partial \pi \psi \), we may choose \( \psi \) on the form \( u(x, \zeta) = \psi_1 + \psi_2 \)
for some boundary \( \psi_1, \psi \).

According to definition 3.1, in order to show that \( \pi \) is a subsolution, there are two distinct cases:

Case 1: \( x_0 \in \Omega \) and \( \nabla \psi_1(x_0) \neq 0 \), or \( x_0 \in \partial \Omega \) and \( \nabla \psi_1 | \partial \Omega (x_0) > 0 \),

Fix \( \tau > 0 \). Then, setting \( M = [\nabla \psi_1(x_0)]_{\text{boundary}}(x_0) \), by proposition 4.3, there is a \( \delta > 0 \) such that
\[ [G_{\psi_1}(x)] \leq \psi_0(x) + MH + \tau \]
(5.7)
for all \( x \in Br(x_0, \delta) \) and \( h \in (0, \delta) \), with \( M = \nabla \psi_1(x_0)_{\text{boundary}}(x_0) \).

Since \( (x_0, \zeta_0) \) is a strict global maximum point of \( \pi \psi \), by the definition of \( \pi \), there is a \( m \in Z^m \) such that
\[ \sup_{x \in Z^m} u^m(x, \zeta) \geq \sup_{x \in Z^m} u^m(x, \zeta) = \sup_{x \in Z^m} \sup_{x \in Z^m} \]
and \( h = \tau/m < \delta \). Then, we can choose \( \zeta, \tau \in Br(x_0, x_0, \delta) \) such that
\[ u^m(\zeta, \tau) + \psi_0(\zeta, \tau) + \psi_2(\zeta, \tau) + \psi_1(\zeta, \tau) \]
(5.8)
for all \( x \in \Omega \times \Omega \) and \( h \to \infty \).

Now, using the comparison principle for the PDIE (5.1) and using (5.8) to get
\[ u^m(x, \zeta) = [G_{\psi_1} + G_{\psi_2}(t, \zeta) + \psi_3(\zeta) \]
\[ \leq \psi_0(x) + [M(h \tau) + \psi_2(\zeta, \tau) + \psi_1(\zeta, \tau) + \psi_3(\zeta, \tau) \]
(5.8)
for all \( x \in \Omega \times \Omega \).

Now, we can choose \( \zeta \in Br(x_0, x_0, \delta) \) such that
\[ \sup_{x \in Z^m} u^m(x, \zeta) + \psi_0(\zeta, \tau) + \psi_2(\zeta, \tau) + \psi_1(\zeta, \tau) \]
(5.8)
for all \( x \in \Omega \times \Omega \).
for all \( x \in B_\delta(x_0, \delta') \), since \( h \leq \tau \). Specifically for \( x = \xi \), we get

\[
    u^m(\xi, \tau) \leq \psi(\xi) + M(\tau - (1 + h) + 2\tau(\tau - (1 + h) + (m^0 \psi)(\xi, \tau)),
\]

that is, since \( \psi(\xi, \tau) = \hat{\psi}(\xi) + \psi(\tau) \),

\[
    \psi(\xi) + \psi((1 + h) \leq (M + 2\tau)(\tau - (1 + h)),
\]

Letting \( \epsilon \to 0 \), so that \( m \to \infty, h \to 0 \) and \( (\xi, \tau) \to (x_0, \delta') \), we get

\[
    \psi(x_0, \delta') \leq M - \nabla \psi(x_0, \delta') \| \sigma \nabla \psi(x_0, \delta') \|
\]

which is what we want according to definition 5.1.

Case 2: \( x_0 \in \Omega \), \( \nabla \psi(x_0) = 0 \) and \( D^1 \psi(x_0) \neq 0 \).

We need to prove that

\[
    \psi(x_0, \delta') \leq 0,
\]

Fix \( \epsilon > 0 \) and choose \( \epsilon > 0 \) so that

\[
    \psi(x_0) \leq \psi(x_0) + \epsilon \| \xi \| + C \| \xi \| \xi 
\]

for all \( x \in \Pi \). Also choose a small \( \gamma > 0 \), then there is a \( \delta = \delta(\gamma) > 0 \) such that

\[
    \psi(x_0) \leq \psi(x_0) + \tau(\xi) + C \| \xi \| \xi
\]

for all \( x_0 \in \Pi \). We define \( \delta = 0 \) as \( \gamma \to 0 \).

As before, since \( (x_0, \delta') \) is a strict global maximum of \( \psi \), there is an \( m = m(\gamma) \in \mathbb{N} \) such that

\[
    \sup_{x \in \Pi \times (0, T)} \sup_{m^0 \psi(x)} \geq (m^0 \psi)(x_0, \delta')
\]

and with \( m(\gamma) \to \infty \) as \( \gamma \to 0 \).

Now, set \( h = \tau/m \) and choose \( (\xi, \tau) \in B_\delta(x_0, \delta') \) so that

\[
    \sup_{x \in \Pi \times (0, T)} \sup_{m^0 \psi(x)} \geq (m^0 \psi)(x_0, \delta')
\]

for all \( x_0 \in \Pi \times (0, T) \). Also, choose \( \delta, \delta' \in \mathbb{N} \) up to that

\[
    \delta \leq \tau < (k + 1) h \quad \text{and} \quad \delta \leq \tau < (k + 1) h
\]

The definition of \( u^m \) then gives us that for all \( x \in \Pi \),

\[
    u^m(x_0, \delta) = \| G_{\delta', \delta} \|_{\psi(x_0, \delta')} + \| G_{\delta', \delta} \|_{\psi(x_0, \delta')}
\]

Choosing \( m \) large as necessary by lemma 5.3 there is a \( \delta' > 0 \) such that

\[
    \| G_{\delta', \delta} \|_{\psi(x_0, \delta')} \leq C
\]

for some \( C > 0 \) and all \( \eta \in [0, h] \), \( x \in B_\delta(x_0, \delta') \).

Then, applying \( G_{\delta', \delta} \) to both sides of (5.14) and using proposition 4.2 (i), (ii) and (iii), we get

\[
    |G_{\delta', \delta}||\psi(x_0 + \tau(\xi) + C \| \xi \| \xi
\]

for all \( x_0 \in \Pi \). We may then apply \( G_{\delta', \delta} \) again to both sides, which gives us

\[
    |G_{\delta', \delta}||\psi(x_0) + \tau(\xi) + C \| \xi \| \xi + C \| \xi \| \xi
\]

for all \( h, \delta' \in \mathbb{N} \) and \( x \in B_\delta(x_0, \delta') \).

Therefore, with \( x = \xi \), we may conclude

\[
    |G_{\delta, \delta}||\psi(x_0) + \tau(\xi) + C \| \xi \| \xi + C \| \xi \| \xi
\]

since \( \delta \leq \tau < (k + 1) h \) and since from (5.15),

\[
    u^m(x_0, \delta) \leq \| G_{\delta', \delta} \|_{\psi(x_0, \delta')} + (m^0 \psi)(x_0, \delta') + \tau
\]

application of \( G_{\delta, \delta} \) to \( \| G_{\delta, \delta} \|_{\psi(x_0, \delta')} \) on both sides gives

\[
    u^m(x_0, \delta) \leq \| G_{\delta', \delta} \|_{\psi(x_0, \delta')} + \| G_{\delta', \delta} \|_{\psi(x_0, \delta')} + 2\tau + C \| \xi \| \xi + C \| \xi \| \xi
\]

Muting the terms around, noting that \( \gamma = \tau < (k + 1) h \) and finally dividing by \( (k + 1) h \), we get

\[
    \psi(\tau) \leq \| G_{\delta', \delta} \|_{\psi(x_0, \delta')} + \| G_{\delta', \delta} \|_{\psi(x_0, \delta')} \leq 2\tau + C \| \xi \| \xi + C \| \xi \| \xi
\]

Finally, letting \( \gamma \to 0 \), so that \( \tau \to 0 \) and \( \tau < (k + 1) h \), we get \( \psi(\delta') \leq 2\tau + C \| \xi \| \xi \), i.e., \( \psi(x_0, \delta') \leq 0 \), since \( \epsilon \) is arbitrary.
Finally, we show how to bound the assumptions (A), (B), and (C) for the
interchange of integral and summation. The integral in (A), (B), and (C)
can be replaced by an exhaustion by rectangles or by an exhaustion by
boxes.

5. The continuation theorem

Theorem 5.1. Suppose that $u(x, y)$ is a function of two variables, $x$ and $y$, and that the limits $u(x, y)$ exist uniformly for $y \to 0$ and $x \to 0$. Then the function $v(x, y)$ defined by

$$v(x, y) = \lim_{\eta \to 0} \int_{-\eta}^{\eta} u(x, y) \, dy$$

exists uniformly for $x \to 0$.

Proof. Let $\eta > 0$ be given. Then, for all $\eta > 0$, there exists a positive constant $C$ such that

$$|u(x, y)| \leq C \quad \text{for } x \in [-\eta, \eta] \text{ and } y \in [-\eta, \eta].$$

Moreover, for $x \in [-\eta, \eta]$, we have

$$\left| \int_{-\eta}^{\eta} u(x, y) \, dy \right| \leq 2\eta C.$$
6. Examples

We conclude with two examples showing the output of the algorithm. The images have been computed using the code developed by Ramesh Gribkovskis for the case without boundary conditions. The code has then been adapted to include boundary conditions by him and Andrei Drastie, and used by them to generate time series of two different evolutions shown below. The images are generated using the VTK toolkit [1]. Because of lack of time to adapt the code, the examples show only regular mean curvature evolution, with \( \alpha = \kappa \).

The first example is shown in Figure 6.1 and shows the mean curvature flow of a cylinder, which has been placed slightly off center inside a sphere. The interactions with the sphere are to the left and right. The time points are chosen to produce interesting images, meaning that the time intervals between subsequent images are not equal. However, these flows from top to bottom. We see that, due to the boundary condition, the cylinder becomes slightly bent and thins out in the middle than at the edge. This bottleneck then quickly becomes thinner and eventually washes it out, producing two distinct surfaces, which both quickly diminish and vanish at the boundary. This evolution is quite different from the evolution of a closed cylinder without boundary conditions, since in that case the cylinder would remain a single surface until it vanishes to a point.

The second example (see Figure 6.2) shows an ellipsoid inside a sphere. This time the ellipsoid has been shifted from the center of the sphere both in \( x \) and \( y \) directions to produce an asymmetric evolution. Once again, the interactions with the sphere are to the left and right and the evolution proceeds diagonally from the top images. We see that the surface adapts to the ellipse boundary condition at both edges and that the thin neck in the left, where the surface intersects the boundary, becomes gradually thinner until, finally, it collapses to the boundary. The evolution then continues with the surface quickly shrinking from the left to the right until it disappears at the right boundary (not shown).

Figure 6.1: First example (cylinder inside sphere).
Figure 6.2: Second example (ellipsoid inside sphere).

REFERENCES

REFERENCES


