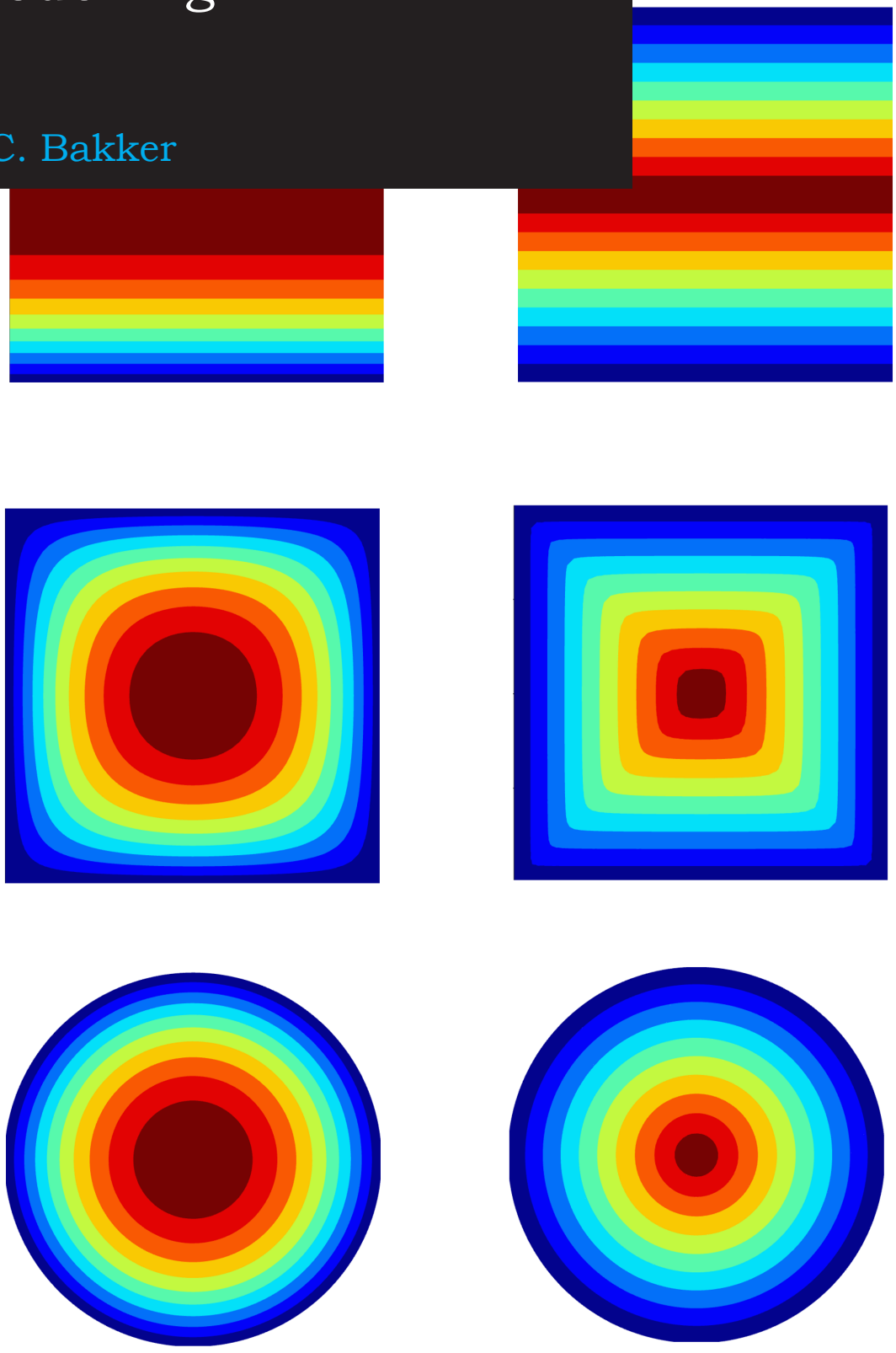
Wall-Distance Calculation for Turbulence Modelling

J.C. Bakker



Technische Universiteit Delft

TU Delft Delft University of Technology

Challenge the future

Wall-Distance Calculation for Turbulence Modelling

by

J.C. Bakker

in partial fulfillment of the requirements for the degree of

Bachelor of Science
in Applied Physics

at the Delft University of Technology,
to be defended publicly on Tuesday July 26, 2018 at 10:00.

Student number:	4478193	
Project duration:	March 23, 2018 – July 26, 2018	
Thesis committee:	Dr. ir. D. Lathouwers,	TU Delft, TNW, Supervisor
	Dr. Z. Perkó,	TU Delft, TNW
	M. Tiberga,	TU Delft, TNW

TU Delft Delft University of Technology

Abstract

The distance to the nearest wall is necessary for the Reynolds-Averaged Navier–Stokes (RANS) turbulence model. The distance to the nearest wall, which is described by the distance field function, is implemented in an existing code that can solve convection–diffusion equations, so that it can be used for a RANS model.

The chosen distance field solving manner is the p-Poisson method, which consists of solving the p-Poisson equation and normalizing the solution afterwards. This method is chosen because the p-Poisson equation is a convection–diffusion equation that can be solved by the code. A characteristic parameter of this method is the p-Poisson parameter p. The higher the parameter, the better the solution to the p-Poisson equation approaches the analytic distance field.

For $p = 2$, the solution of the p-Poisson equation has an L_2 error lower than 1×10^{-3} compared to the analytic solution of that equation, for a 1D and circular domain. If $p > 2$, the p-Poisson equation becomes non-linear. Since the convection–diffusion equation solver available in the code can only solve for linear cases, a Picard iteration is used to overcome the problem. For $p = 3$, oscillation of the solutions between the Picard iterations is observed for all tested domains. This problem is resolved by damping of the solution, which means that after every Picard iteration a weighted average is taken from the newly calculated solution and the weighted solution of the previous iteration. With this included, numerical p-Poisson equation solutions approach their corresponding analytic solution with an relative L_2 error less than 1×10^{-3}, for $p > 2$.

When the normalized solution of the p-Poisson equation is compared to the analytic distance field for a 1D, square and circular domain, it is concluded that p must be at least 8 in order to obtain a relative L_2 error lower than $5\,\%$. Therefore the minimal suggested p-Poisson parameter is 8, and 10 is advised, since complex domains may be harder to solve for. The p-Poisson method has been used to calculate the distance field in a geometry representing $1/16^{\text{th}}$ of the core of the MSFR. Using the minimal and recommended settings, almost the same results are obtained. For nearly the whole domain the calculated distance field values are as expected, except for at the pump and heat exchanger. When local mesh refinement is applied on that location. So, when the distance field does not behave as expected using the recommended settings, mesh refinement is recommended.

Acknowledgements

First, I want to thank Marco Tiberga for his guidance of this thesis. Not only for the quick answering of all my questions, but also for all the time and effort he has put into this project. Next I want to thank Aldo Hennink for his great support on helping me getting familiar with the code I needed to work with and all the aftercare. I also want to thank Danny Lathouwers for supervising the project and for his useful input at some critical points. Furthermore, I want to thank David van Nijen for supplying me with the Molten Salt Fast Reactor mesh. Last, but not least, I want to thank everyone from the department, that, directly or indirectly, helped me in the time present.

J.C. Bakker
Delft, July 2018

Nomenclature

α	Parameter of the smooth clipping method
β	Parameter of the smooth clipping method
γ	Damping coefficient
$\hat{u}_p$	p-Poisson equation, numerical, damped solution
$\hat{v}_p$	p-Poisson equation, numerical, damped normalized solution
$\mathbf{n}$	Unit inner normal
Ω	A domain
$\partial\Omega$	Boundary of a domain
$\partial\Omega_D$	Boundary of a domain where the Dirichlet boundary condition holds
$\partial\Omega_N$	Boundary of a domain where the Neumann boundary condition holds
$\tilde{u}_p$	p-Poisson equation, analytic, solution
$\tilde{v}_p$	p-Poisson equation, analytic, normalized solution
C	Constant
d	True distance field
K	Diffusion coefficient of a convection-diffusion equation
K_1	Integration constant
K_2	Integration constant
p	p-Poisson parameter
p_s	Spacial polynomial used for the discontinuous Galerkin method
R	Radius of circle
u	Solution to the Eiknonal equation
u_p	p-Poisson equation, numerical, solution
v_p	p-Poisson equation, numerical, normalized solution

Contents

List of Figures

1

Introduction

This chapter is an introduction to the thesis, where first the reason for computing the distance field is given. Second the research question is described and afterwards the outline of the thesis is given.

1.1. Intention of Thesis

The Molten Salt Fast Reactor (MSFR) is a generation IV nuclear reactor. The advantage of nuclear reactors, compared to other power plants, is that they do not emit CO_2. Moreover, the MSFR uses thorium as fuel, which leads to smaller production of long-lived activities than uranium. In addition to this, the fuel of the MSFR is liquid instead of the solid fuel, which is used in conventional nuclear reactors. This makes a meltdown accident impossible, because the liquid fuel, in case of an emergency, is drained into an external cooling bath [1].

Research on developing the MSFR is still ongoing and a part of this research focuses on the flow of the liquid fuel inside the MSFR. For this flow modeling simulations, turbulence models are used, and in general it is hard to obtain stable computations near a wall for turbulence models, due to the complex behavior of the fluid near the boundary. Therefore, several methods have been developed to overcome this problem, such as the advanced Reynolds stress model [2], the wall function approach [3, 4] and the low-Reynolds models [5, 6]. Other ways for dealing with this instability involve making use of the distance to the nearest wall, which is done by the Spalart-Allmaras and Shear-Stress Transport turbulence models [7]. The distance to the nearest wall is referred to as the wall-distance field or just distance field.

For flow research on the MSFR, our research group, RPNM, has worked on the Reynolds-Averaged Navier–Stokes (RANS) equations for turbulence modeling. These equations also make use of the distance field to overcome the stability issue [8]. The research group that investigates the flow in the MSFR currently can not produce a wall-distance field that can be used as input of the RANS equations. The wall-distance field solving method was implemented in a codebase that already can solve convection–diffusion equations for given geometries, which is solved using the discontinuous Galerkin method.

1.2. Research Question

This thesis is about implementing a method for generating the distance field of a given geometry, which is also referred to as the domain. Therefore, the research question reads:

1. What numerical solvers can be used to calculate the distance field to be utilized for the RANS equations?

2. Which of these methods is most suited?

Subsequently, the most suited method is implemented in the code, and this implementation is tested and improved.

1.3. Outline of Thesis

First, some background information about the discontinuous Galerkin method, used to solve convection–diffusion and the RANS equations, and an equation that yields the wall-distance, is given in Chapter 2. Moreover, several numerical methods for solving the distance field are given, from which one is chosen and a analysis of this method is given. In Chapter 3, the chosen numerical method is tested for some simple domains, and, where needed, improvements to the code are made. With these tests, an error analysis is done to determine for which settings the numerical method is able to produce a satisfying distance field. In Chapter 4, the distance field of an approximate MSFR is calculated, using the improved settings suggested in Chapter 3. Finally, Chapter 5 contains the conclusions and recommendations for further research.

2

Choosing the Appropriate Wall-Distance Solving Method

In this chapter different methods are compared for calculating the wall-distance field, and subsequently one of these methods is chosen to be implemented in the code. Before this comparison is made, some background information about the Discontinuous Galerkin (DG) and the Eikonal equation is provided. The Eikonal equation is treated, because the solution satisfies the wall-distance field for some particular parameters.

2.1. Discontinuous Galerkin

As mentioned in the Introduction, the code in which the distance field was implemented is able to solve convection–diffusion equations with the use of DG. DG is a Finite Element method, which means that the problem domain is subdivided into elements. Within each element, the quantity to be solved is approximated by a linear combination of a set of basis functions. These basis functions are known in advance, although the corresponding weights are not known. With this approximation the problem consists of solving a system of linear equations for the weights instead of solving a continuous function. Note that this thesis uses the term spatial polynomials p_s, which is linked to the number of basis functions. So, for example, for $p_s = 2$, a constant, linear and quadratic basis function is meant.

In order to solve for these weights, the Galerkin method multiplies the equation of the problem to be solved with a test function that can represent every local basis function. Afterwards, the equation is integrated over the domain, and via integration by parts the highest order derivative of the equation is one lower than the original order of derivative. Hence, for convection–diffusion problems, the highest order of derivative is one. Due to the test function, the same number of linearly independent equations as unknown weights can be obtained, so that there exists an unique solution for the weights. This is due to the fact that the test function can represent every local basis function, and the number of local basis functions is the same as the number of weights.

The method of solving a problem described in the previous Paragraph is called the Galerkin method. A downside of this method, is that it can induce instability due to the discontinuous test function, since it is only defined locally [9]. Furthermore, there is no local, element-based, conservation of quantities, which is present for some other methods, e.g. Finite Volume.

The DG method is appropriate for solving convection–diffusion equations, because of its stability and local conservation. DG is almost the same as standard Galerkin, except that for this method the flux term is replaced by a function that governs consistency, coercivity and conservation [10]. With the flux term, the vector field inner product with the outer normal, multiplied by the to be solved quantity, term of the integrand of the integral over the element's boundary is meant. Moreover, due to these features, DG supports local refinement of the mesh and increase of p_s [11]. The name of DG derived from the allowed discontinuity of the solution to the problem. This means that neighbouring elements are allowed to have different values, thus the solution can be discontinuous. Although discontinuity is allowed, via penalties and/or flux functions this is minimized. The standard Galerkin method does not allow this discontinuity, which then can induce the instability and lack of local conservation.

2.2. Eikonal Equation

In this section, first the definition of the Eikonal equation is given, then it is shown that the solution of a specific Eikonal equation represents the distance field. Finally, a simple derivation of this formula, which has the distance field as solution, is given.

2.2.1. Definition of the Eikonal Equation

The Eikonal equation is defined as [12]

$$\begin{aligned} |\nabla u(\mathbf{x})| &= \frac{1}{f(\mathbf{x})}, \quad \mathbf{x} \in \Omega, \\ u(\mathbf{x}) &= q(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega, \end{aligned} \tag{2.1}$$

where $u(\mathbf{x})$ can be interpret as the "time" it takes to get to the boundary $\partial\Omega$ from the point $\mathbf{x} \in \Omega$, $f(\mathbf{x})$ can be seen as the "speed" at $\mathbf{x}$, and $q(\mathbf{x})$ represents the "penalty" from starting on the boundary at point $\mathbf{x} \in \partial\Omega$.

2.2.2. Eikonal Equation Satisfying the Distance Field

When one sets the boundary penalty to zero, i.e. $q(\mathbf{x}) = 0$, and the speed to one, i.e. $f(\mathbf{x}) = 1$, the time $u(\mathbf{x})$ will then represent the wall-distance field on the domain Ω.

It is logical that the penalty $q(\mathbf{x})$ is set to 0, since the distance field is equal to 0 at the boundary. In addition to this, the speed $f(\mathbf{x})$ should be constant. In the first place, because the distance field does not have any preference where to increase on the domain. Furthermore, this constant should be one, because the distance field gradient, which is perpendicular to contour lines, is one. This is due to the fact that locally the distance field increases at the same rate as the the Euclidean distance between two points that lay on or in the extension of the gradient.

The absolute value will be used for the reason that the gradient of the distance field will also be negative on the domain Ω, since the distance field can not only increase. The use of the absolute value of the gradient leads to two solutions, one of which is the negative of the other. In order to obtain a unique solution, one uses the restriction that the distance field must be positive.

Concluding, the Eikonal equation, whose solution satisfies the distance field, is given by

$$\begin{aligned} |\nabla u(\mathbf{x})| &= 1, \quad \mathbf{x} \in \Omega, \text{ with} \\ u(\mathbf{x}) &= 0, \quad \mathbf{x} \in \partial\Omega. \end{aligned} \tag{2.2}$$

It should be noted that when a pure mathematical domain is used, the distance field has no dimension, but when a physical domain is used, like the MSFR in Chapter 4, the distance field has the same dimension in which the physical domain is defined.

2.2.3. Derivation of the Wall-Distance Formula

When one takes an arbitrary domain Ω, in which the points $\mathbf{x}$ and $\mathbf{x} + \delta\mathbf{x}$ are defined, with $|\delta\mathbf{x}|$ small, the difference in wall-distance between these two points is

$$d(\mathbf{x} + \delta\mathbf{x}) - d(\mathbf{x}), \tag{2.3}$$

where d is the wall-distance. With the use of Taylor expansion, it can be shown that equation (2.3) is approximately equal to

$$d(\mathbf{x} + \delta\mathbf{x}) - d(\mathbf{x}) \approx \nabla d(\mathbf{x}) \cdot \delta\mathbf{x}. \tag{2.4}$$

Let the angle between the gradient $\nabla d d(\mathbf{x})$ and the vector $\delta\mathbf{x}$ be θ, then the right-hand side of Equation (2.4) can be written as

$$\nabla d(\mathbf{x}) \cdot \delta\mathbf{x} = |\nabla d(\mathbf{x})|\,|\delta\mathbf{x}| \cos\theta. \tag{2.5}$$

In Figure 2.1 a visualization of a possible set-up of the variables used in this subsection is shown.

Returning to equation (2.3), the left-hand side can be approximated as

$$d(\mathbf{x} + \delta\mathbf{x}) - d(\mathbf{x}) \approx |\delta\mathbf{x}| \cos\theta, \tag{2.6}$$

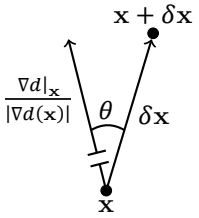


Figure 2.1: A visualization of a possible set-up of the points $\mathbf{x}$ and $\mathbf{x}+\delta\mathbf{x}$, vectors $\delta\mathbf{x}$ and $\frac{\nabla d|_{\mathbf{x}}}{|\nabla d(\mathbf{x})|}$ and angle θ between the two vectors in domain Ω.

because the first term of the r.h.s. $|\delta\mathbf{x}|$ represents the Euclidean distance between the two points $\mathbf{x}$ and $\mathbf{x}+\delta\mathbf{x}$. Furthermore, the cosine ensures that the wall-distance only changes in the direction of the wall-distance gradient.

Combining equations (2.5) and (2.6), and using equation (2.4), one obtains

$$1 = |\nabla d\,(\mathbf{x})|\,. \tag{2.7}$$

Using the boundary condition $d\,(\mathbf{x}) = 0, \mathbf{x} \in \partial\Omega$ and the fact that equation (2.7) holds for every point $\mathbf{x}$ in the domain Ω, it can be concluded that the derived equation (2.7) is the Eikonal equation (2.2) Q.E.D.

2.3. Some Wall-Distance Solving Methods

In Section 2.2, an equation has been derived which satisfies the distance field. There are multiple ways of solving the distance field, either with or without the use of the wall-distance equation (2.2). In this section, some common methods for solving the wall-distance are discussed and afterwards a appropriate method is chosen. At last, some additional information is given about the chosen method.

2.3.1. Straightforward Implementation

One obvious method would be to calculate for every point on the grid, the distance with all the points on the boundary and assign the minimal distance to the point on the grid. This is a very simple method, but very inefficient method of calculating the distance field w.r.t. other existing methods, as described in Jones and Chen [13]. This can be simply explained by the fact that, for this method, no data is reused from other points when calculating the distance field of some point on the grid.

2.3.2. Fast Marching Method

Another method for calculating the distance field is the Fast Marching Method (FMM). The FMM is specifically made for solving the Eikonal equation as defined in Equation (2.1), with the boundary penalty $q\,(\mathbf{x})$ equal to 0 [14]. For this reason this method is expected to be competitive for solving the Eikonal equation and the distance field, which is supported by the use of it in the literature, e.g. in [15–17].

2.3.3. Solving the p-Poisson Equation

In Belyaev and Fayolle [18], a total of five methods for obtaining the distance field are described. These methods are compared with each other via the analytic (true) distance field. It is concluded that the p-Poisson equation has the smallest amount of relative error, but it is not the fastest method to compute distance field. Another quality of the p-Poisson method is that it can be implemented using existing convection–diffusion equation solvers, as shown in Wukie and Orkwis [7].

2.4. Choosing Appropriate Wall-Distance Solving Method

In this section, first a method for solving the distance field is chosen, which was implemented in the code, and afterwards some information about this chosen method will be given.

2.4.1. Choosing the Appropriate Method

As mentioned in the introduction, the chosen wall-distance solving method was implemented in an existing codebase, which can solve convection–diffusion equations. That is why the p-Poisson method is chosen instead of other methods, whereby the solvers would need to be implemented entirely from scratch in the codebase. In addition to this, the p-Poisson method can produce the most precise distance field from all the methods mentioned in Belyaev and Fayolle [18].

2.4.2. Additional Information About the Appropriate Method

The p-Poisson method boils down to solving the following equation, given by [18]:

$$\begin{aligned} \nabla \cdot \left(\left| \nabla u_p \right|^{p-2} \nabla u_p \right) &= -1, \quad \mathbf{x} \in \Omega, \\ u_p &= 0, \qquad \mathbf{x} \in \partial\Omega_D, \end{aligned} \tag{2.8}$$

where $u_p\,(\mathbf{x})$ is the approximate distance field, Ω is the domain for which the distance field is calculated, $\partial\Omega_D$ is the boundary of the domain where the Dirichlet boundary condition holds and p is the p-Poisson parameter. In this case $\partial\Omega_D = \partial\Omega$. The p-Poisson parameter p must be larger or equal to 2 and the higher the parameter is, the better the distance field is approximated. Note that the term $\left|\nabla u_p\right|^{p-2}$ from equation (2.8) represents a diffusion coefficient.

In contrast to the boundary condition from equation (2.8), it is also possible to define the Neumann boundary condition

$$\frac{\partial u_p}{\partial \mathbf{n}} = 0, \ \mathbf{x} \in \partial\Omega_N, \tag{2.9}$$

where $\mathbf{n}$ is the unit inner normal $\nabla u_p / \left|\nabla u_p\right|$ and $\partial\Omega_N$ is the part of the boundary where the Neumann condition governs instead of the Dirichlet condition stated in equation (2.8). The Neumann boundary condition can be used to mimic open boundaries, i.e. symmetry and inflow/outflow, as described in Wukie and Orkwis [7]. Although the Neumann boundary condition can be used on parts of the boundary, it can not be used on the whole boundary, i.e $\partial\Omega_N \neq \partial\Omega$, because otherwise there is no unique solution.

In Belyaev and Fayolle [18], it is stated that if $p \to \infty$, the approximate distance field approaches the true distance field. A simple, yet very insightful, way of proving this goes as follows: on the domain Ω, one integrates equation (2.8) and obtains

$$\int_\Omega \nabla \cdot \left(\left| \nabla u_p \right|^{p-2} \nabla u_p \right) d\mathbf{r} = \int_{\partial\Omega} -\mathbf{n} \cdot \left| \nabla u_p \right|^{p-2} \nabla u_p d\mathbf{l} = C, \tag{2.10}$$

where C is a constant. When one takes $p \to \infty$, Equation 2.10 only holds a non-zero or -infinity solution for u_∞ if

$$\left|\nabla u_\infty\right| = 1. \tag{2.11}$$

When Equation (2.11) is recombined with the boundary condition from equation (2.8), the same problem is encountered as in Equation (2.2) Q.E.D.

Naturally, when the p-Poisson parameter does not approach infinity, only an approximation of the distance field can be calculated. Hence, a normalization is introduced in Belyaev and Fayolle [18] to improve the approximate distance field:

$$v_p\,(\mathbf{x}) = -\left|\nabla u_p\right|^{p-1} + \left[\frac{p}{p-1} u_p + \left|\nabla u_p\right|^p \right]^{\frac{p-1}{p}}, \tag{2.12}$$

where $v_p\,(\mathbf{x})$ is the normalization of the the approximate distance field $u_p\,(\mathbf{x})$. Notice that when $p \to \infty$, the normalization boils down to

$$v_p\,(\mathbf{x}) = u_p\,(\mathbf{x}),$$

as one would expect. It is easy to see that on the boundary $\partial\Omega$ the normalization v_p is equal to 0, because the approximate distance field u_p will be 0 at the boundary and hence, the two absolute gradients $\left|\nabla u_p\right|$ cancel each other. Additionally, the normalization's directional derivative w.r.t. the unit inner normal $\mathbf{n} = \nabla u_p / \left|\nabla u_p\right|$ is equal to 1 on the boundary $\partial\Omega$. This can be simply verified by taking

the gradient of the normalization v_p, using the fact that the distance field approximation u_p is 0 on the boundary and performing an inner product with the inner normal $\mathbf{n}$.

Similarly to the normalization v_p, the distance field d is equal to 0 at the boundary $\partial\Omega$ and the directional derivative of the distance field d w.r.t. the inner normal $\mathbf{n} = \nabla d/|\nabla d|$ is equal to 1 on the boundary $\partial\Omega$. Moreover the higher order directional derivatives w.r.t. the inner normal $\mathbf{n}$ of the distance field d are all equal to 0. This can be shown with ease using the distance field equation defined in equation (2.2). The first order directional derivative is given by

$$\frac{\partial d}{\partial \mathbf{n}} = \nabla d \cdot \frac{\nabla d}{|\nabla d|} = |\nabla d| = 1, \mathbf{x} \in \partial\Omega, \tag{2.13}$$

where it is used that $|\nabla d| = 1$. The second-order directional derivative is given by

$$\frac{\partial^2 d}{\partial^2 \mathbf{n}} = \nabla\left(\nabla d \cdot \frac{\nabla d}{|\nabla d|}\right) \cdot \frac{\nabla d}{|\nabla d|} = \nabla 1 \cdot \frac{\nabla d}{|\nabla d|} = 0, \mathbf{x} \in \partial\Omega. \tag{2.14}$$

In the same manner as in equation (2.14) it can be shown that all higher order directional derivatives w.r.t. the inner normal $\mathbf{n}$ of the distance field d are equal to 0 on the boundary $\partial\Omega$.

The approximate distance field u_p only confirms to the Dirichlet boundary condition of the distance field d, whereas the normalization v_p also confirms to the first order directional derivative w.r.t. the inner normal $\mathbf{n}$ equal to 1 boundary condition. This supports the fact that the normalization v_p is a better approximation to the distance field than the approximate distance field u_p.

3

Testing and Improvement of the Wall-Distance Implementation

In the first section, the solution of the p-Poisson equation with $p = 2$ is evaluated for different domains. In the second section, the p-Poisson equation will be solved for $p > 2$, using some improvements to the implementation so that converged solutions are obtained if needed. In the third section an error analysis w.r.t. the true distance field is done for several domains to conclude which improvement, or combination of improvements, provides the ”best” solution (where best is defined there).

3.1. Solving the p-Poisson Equation Using Parameter Two

In this section, first some information is given about the p-Poisson equation using $p = 2$. Afterwards the results of the distance field approximation u_2 are shown and discussed, including a comparison with the normalization v_2.

3.1.1. General Comment

When using $p = 2$, the diffusion term $\left|\nabla u_p\right|^{p-2}$ from equation (2.8) will be equal to 1. Therefore, the p-Poisson equation to be solved is linear. The solver for convection–diffusion equations available in the codebase is only able to solve linear equations, so the approximate distance field u_2 can be solved with a straightforward implementation.

3.1.2. Results for the Approximate Distance Field and Normalization

In this subsection, the p-Poisson equation using $p = 2$ is solved for a 1D, a square and a circular domain. Along with the presentation of the approximate distance field u_2 and normalization v_2, a comparison will be made with either the analytic solution to the p-Poisson equation using $p = 2$ or results obtained elsewhere. Afterwards a conclusion will be made about the accuracy of the implementation in the code of the p-Poisson equation using parameter 2.

One-Dimensional

The boundaries of the 1D problem satisfy the Dirichlet boundary condition stated in equation (2.8). Here, the 1D problem is represented by a square domain $[-1, 1] \times [-1, 1]$, where two opposite boundaries confirm to the Dirichlet boundary condition, whereas the other two opposite boundaries confirm to the Neumann boundary condition defined in equation (2.9). As described in the introduction, the code uses the discontinuous Galerkin method and for all the examples in this section $p_s = 2$ is used. This is because the code already uses spatial polynomial 2 for solving other numerical problems. Higher order polynomials are avoided to limit the calculation cost. Furthermore, a structured mesh of 50x50 squares is used for the square cases and for the circular cases the mesh consists of 9356 triangles. The 1D results are shown in Figure 3.1 (where on the left the approximate distance field u_2 and on the right the normalization v_2 is displayed).

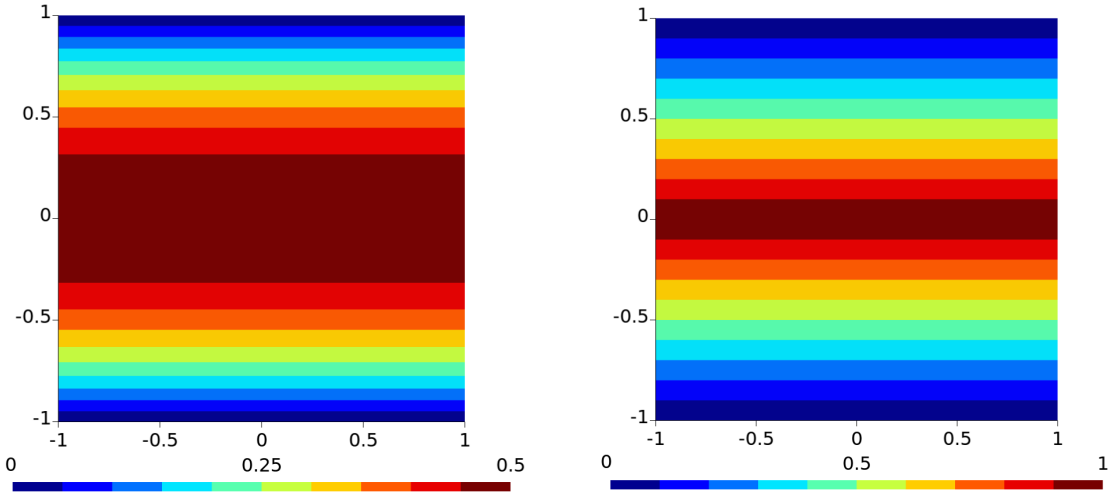


Figure 3.1: The one-dimensional numerical solution of the p-Poisson equation using $p = 2$ in figure (a) and its normalization in figure (b).

Since the linear p-Poisson equation can be solved easily for the 1D case, the relative L_2 error between the solutions shown in Figure 3.1 and the analytic solution can be calculated. The relative L_2 error between the to be compared function $f(\mathbf{x})$ and the reference function $g(\mathbf{x})$ is

$$\sqrt{\frac{\int_\Omega (g(\mathbf{x}) - f(\mathbf{x}))^2 \, d\mathbf{r}}{\int_\Omega (g(\mathbf{x}))^2 \, d\mathbf{r}}}. \tag{3.1}$$

The analytic solution of the 1D p-Poisson equation using parameter 2 is given by

$$\tilde{u}_2 = -\frac{x^2}{2} + \frac{1}{2}, \quad x \in [-1, 1], \tag{3.2}$$

where $\tilde{u}_p$ is the analytic solution to the p-Poisson equation using parameter p, and the corresponding normalization is given by

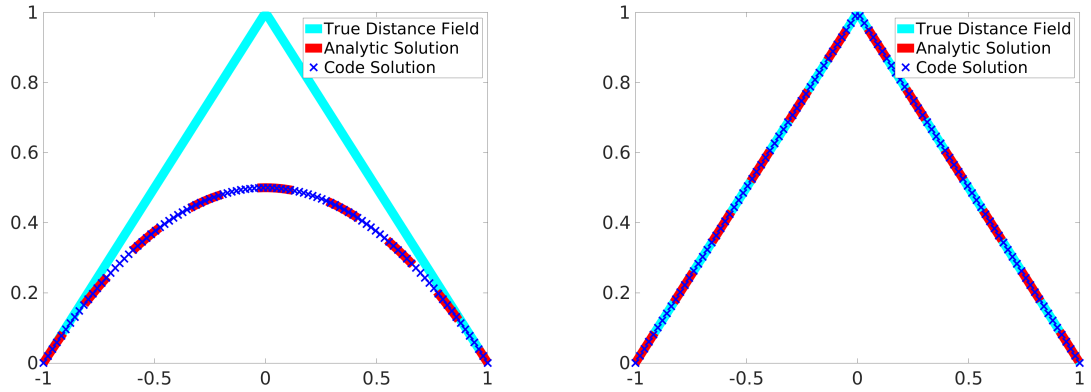
$$\tilde{v}_2 = -|x| + 1, \quad x \in [-1, 1], \tag{3.3}$$

where $\tilde{v}_p$ is the analytic solution of the normalization. In Figure 3.2, a cross section of the approximate distance field u_2 and normalization v_2 from Figure 3.1 are compared with their analytic counterpart, together with the true distance field. One can see from Figure 3.2 that the numerical calculated results are in accordance with their analytic counterparts, which is also confirmed ' calculating the L_2 relative error of the solutions shown in Figure 3.1 w.r.t. their analytic solution. Namely the L_2 relative error of the approximate distance field displayed in Figure 3.1(a) is 7×10^{-8} and for the normalization shown in 3.1(b) it is 1×10^{-7}. Notice that the normalization v_2 equals the true distance field, though there is no proof that this will happen for all domains.

Square

The domain $[-1, 1] \times [-1, 1]$ is tested, where all the boundaries satisfy the Dirichlet boundary condition. In Figure 3.3 the solutions are shown, where on the left side the approximate distance field u_2 is displayed and on the right side the normalization v_2. From the normalization v_2 shown in Figure 3.3(b) it can be concluded that the calculated normalization can not be equal to the true distance field, because the true distance field at point $(0, 0)$ is 1, where the normalization v_2 is equal to 0.767. So the idea that the normalization of the p-Poisson equation using parameter 2 is always equal to the true distance field is hereby contradicted.

In order to confirm that the results displayed in Figure 3.3 are the solution of the p-Poisson equation using parameter 2 and its normalization, a comparison between the results of the same problem generated by MATLAB [19] shown in Figure 3.4 can be done. When the results of Figure 3.3 are compared



(a) (b)

Figure 3.2: The cross sections from the numerical solutions shown in figure 3.1, compared with their analytic counterpart and true distance field. On the left the blue crosses represent the numerical data of the approximate distance field, the red dashed line is the analytic solution and the cyan continuous line represents the true distance field. On the right the same setup is used, only there the blue crosses represent the numerical data of the normalization.

with the results of Figure 3.4, no difference can be seen except some very small numerical deviation, e.g. on the boundaries. In addition to this, the normalization from Figure 3.3(b) is in good agreement with the result from Wukie and Orkwis [7].

Circle

Below the circular domain is evaluated, where the mesh is a circle with radius $R = 1$. The edge of the circle obeys the Dirichlet boundary condition stated in equation (2.8).The analytic solution of the p-Poisson equation with $p = 2$ in this domain is given by [20]

$$\tilde{u}_2 = \frac{1}{4}\left(1 - \mathrm{x}^2\right), \quad |\mathrm{x}| \leq 1. \tag{3.4}$$

The approximate distance field u_2 and its normalization v_2 of the circular case are shown in Figure 3.5. The relative L_2 error of the approximate distance field shown in Figure 3.5(a) is 3×10^{-4} and for the normalization v_2 shown in Figure 3.5(b) it is 2×10^{-4}.

With three cases analyzed, qualitatively as well as quantitatively, where the quantitative relative L_2 error is smaller than $0.1\,\%$, which is considered as a threshold error to confirm that the numerical implementation approximated the analytic solution, it can be concluded that the implementation works as intended for $p = 2$.

3.2. Solving the p-Poisson Equation for any Possible Value of the p-Poisson Parameter

In this section the implementation for solving the p-Poisson equation will be tested and improved for $p > 2$. First, a problem with solving the p-Poisson equation using $p = 3$ is shown, whereby no converged solution can be obtained. Next the cause is determined and possible solutions for this problem are presented. The solutions will be tested by solving the p-Poisson equation using $p = 4$ and look if they converge fast enough compared to other solutions. Finally, the successful methods for solving the problem will be tested via the relative L_2 error with analytic solutions to the p-Poisson equation. A competitive method is defined as a method that does not require more than ten times the number of Picard iterations as other methods solving the oscillation problem to obtain the same result.

3.2.1. Description of Problem Encountered when Solving the p-Poisson Equation Using Parameter Three

With the $p > 2$, the p-Poisson equation defined in equation (2.8) has a non-constant diffusion coefficient $\left|\nabla u_p\right|^{p-2}$, which causes the equation to be non-linear. Since the provided solver in the code can only

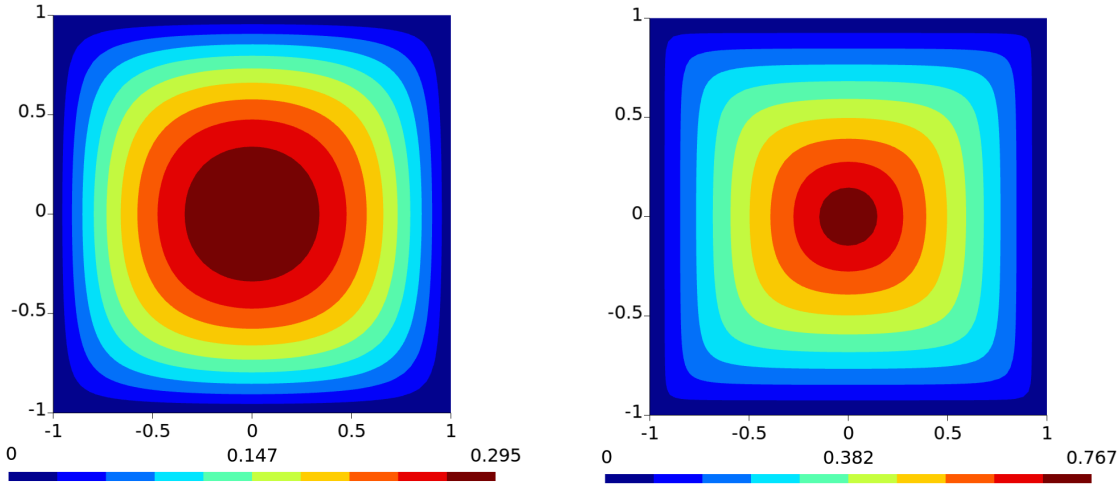


Figure 3.3: The numerical solution of the p-Poisson equation using $p = 2$ applied to a square domain in figure (a) and its normalization in figure (b).

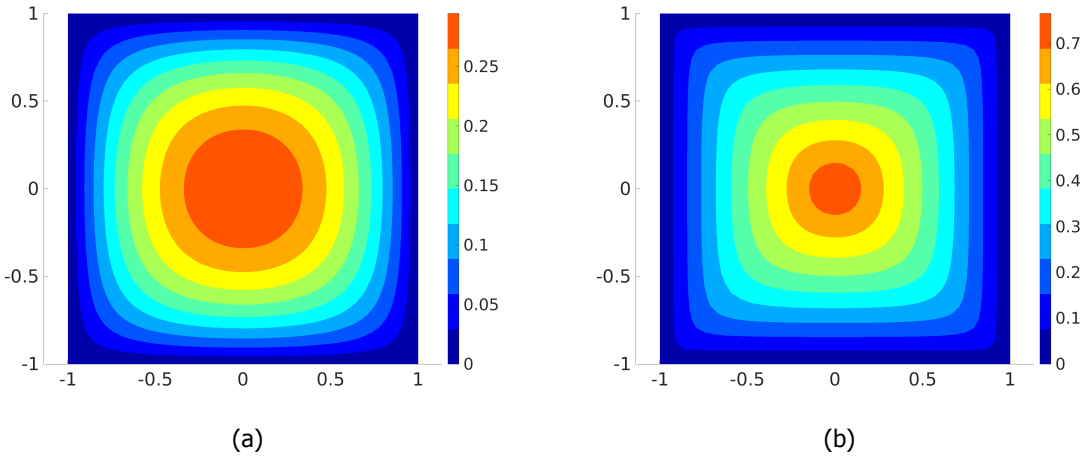


Figure 3.4: The numerical solution of the p-Poisson equation using $p = 2$ applied to a square domain on the left and its normalization on the right. These solutions where generated by MATLAB [19].

solve linear equations, Picard iteration is used to tackle the non-linearity, where the initial solution is the solution of the p-Poisson equation using a parameter 1 lower than the to be solved p-Poisson parameter. The Picard iteration, also called the fixed point iteration, uses a function as input of every iteration, that is used to fill in the non-linear part. With the non-linear part assigned a value, only the linear part remains to be solved. When this is solved, a solution is obtained that is then used as input of the next Picard iteration. This recursion is repeated until the obtained solution does not deviate more than a pre-defined error with the previous iteration.

When the p-Poisson equation is solved using $p = 3$, no convergence in the Picard iteration is observed. The solutions generated by each iteration oscillate back and forth between two identical solutions. In order to visualize this behavior, in Figure 3.6 the solutions after 10 and 11 iterations are shown, where the same settings are used as set for the result from Figure 3.1(a) (except the p-Poisson parameter). In order to be sure that the oscillation does not fade away, the same is done as for Figure 3.6, only now the Picard iterations 100 and 101 are shown in Figure 3.7. To be sure that the oscillation does not only occur when solving for a 1D domain, the same test is run on a circular domain, where the same settings are used as set for the result from Figure 3.5(a), except the p-Poison parameter, which is 3 here. From this test it can be concluded that also for a circular domain oscillation occurs.

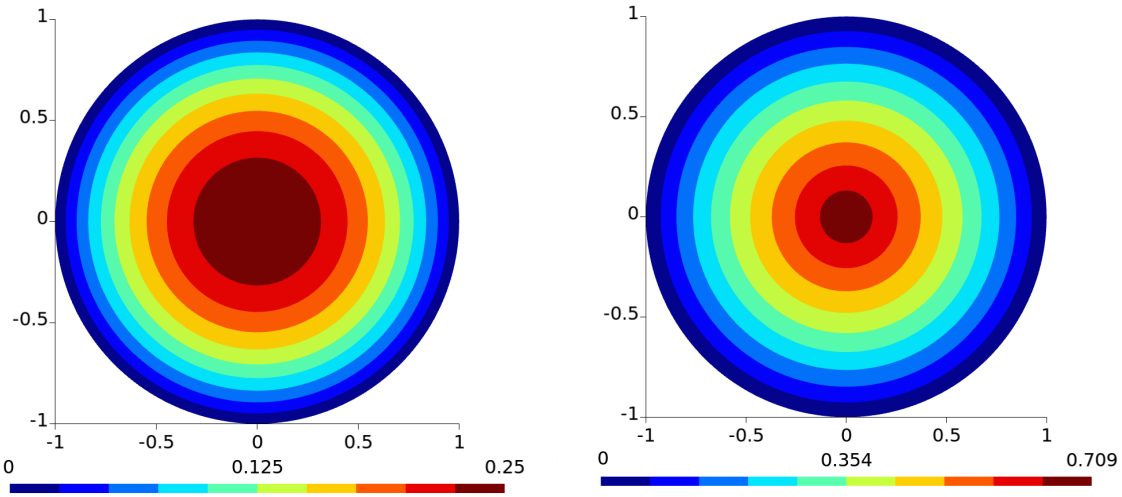


Figure 3.5: The numerical solution of the p-Poisson equation using $p = 2$ applied to a circular domain on the left and its normalization on the right.

3.2.2. Cause of the Problem

A possible cause of the problem could be that the initial diffusion coefficient induces oscillation in the Picard iteration. To test this, a different initial condition for the Picard iteration was used. First, the initial condition $|x|, x \in [-1, 1]$ is used. Once more, the results shown in Figure 3.6, the same results are obtained as the ones shown in Figure 3.6. When $\sin x, x \in [-1, 1]$ is used as initial condition, oscillation is still observed, only in this case it oscillates between two other identical solutions than the solutions shown in Figure 3.6. Thus, it is clear that the cause of the oscillation does not lie in the initial condition.

Another cause of the problem could lie in the use of discontinuous Galerkin. Therefore, a mockup code based on Finite Volume (FV) method was developed to solve the same problem. The solution of the 1D domain, generated by the FV code, p-Poisson equation using $p = 3$ is calculated for a line of length 2 consisting of 100 elements. It is observed that solutions of successive iterations also oscillate. Hence, the choice of the numerical scheme does not influence the oscillation.

Since the only thing that changed in the implementation for solving the p-Poisson equation using $p = 3$ compared to the $p = 2$ case, is the addition of the Picard iteration, maybe the Picard iteration itself is the cause of the oscillation. To test this, the first Picard iteration of the 1D case was calculated by hand to see what the solution should be. The solution of Equation (2.8) in the domain $[-1, 1]$, using Dirichlet boundary condition, is given by

$$\tilde{u}_{3,\text{Pic}=1} = K_1 \ln |x| + |x| - K_2, \tag{3.5}$$

where K_1 and K_2 are integration constants. It is known that the calculated solution $\tilde{u}_{3,\text{Pic}=1}$ should be 0 on $|x| = 1$, so integration constant $K_2 = -1$. However, for the integration constant K_1, there is no explicit condition. It should be noted that the calculated function $\tilde{u}_{3,\text{Pic}=1}$ diverges at $x = 0$. Its derivative also diverges at $x = 0$, so a possible solution to overcome this would be setting the integration constant K_1 to 0. Then, a linear function is obtained for the first Picard iteration solution $\tilde{u}_{3,\text{Pic}=1}$, where the weight of the linear term is 1. Therefore, the derivative of the first Picard iteration solution $\tilde{u}_{3,\text{Pic}=1}$ will be equal to 1. Notice that for $p = 2$ the diffusion coefficient is 1, which is also the case for the second Picard iteration $\tilde{u}_{3,\text{Pic}=2}$. Hence, the solution of the second Picard iteration of $p = 3$ is equal to the solution of the p-Poisson equation using parameter 2.

Concluding, due to the divergence in the first Picard iteration when solving for the p-Poisson equation using parameter 3, the next Picard iteration will calculate the initial condition again. The divergence happens at the place where the initial solution has a zero derivative. In order to check if this zero derivative causes the divergence, the same calculations where done as done for the results shown in Figure 3.7, only now the diffusion term is lifted by a value of 0.1. This means that instead of

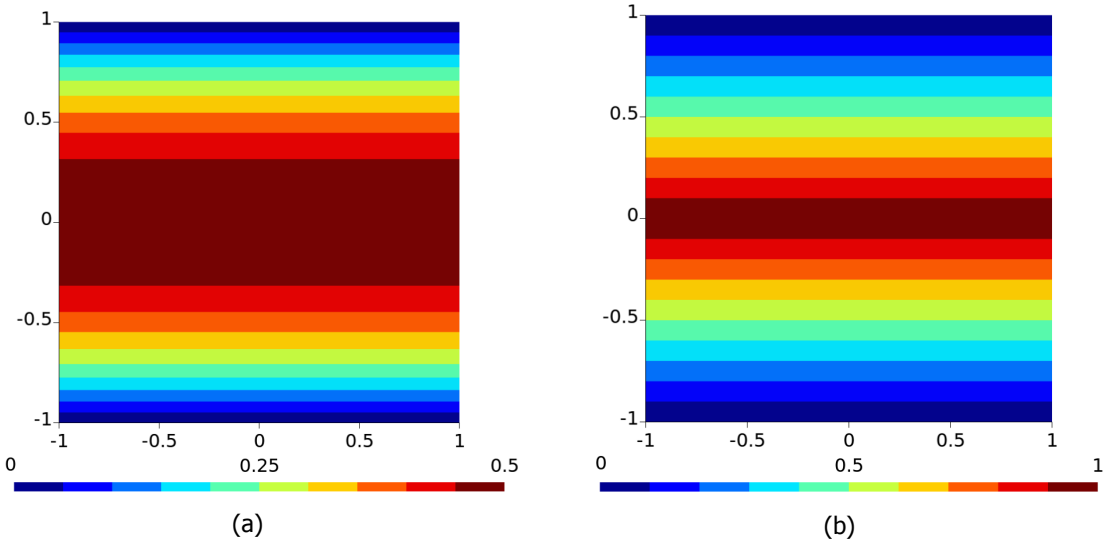


Figure 3.6: The numerical solution of the p-Poisson equation using $p = 3$ applied to a square domain after 10 Picard iterations on the left and 11 on the right.

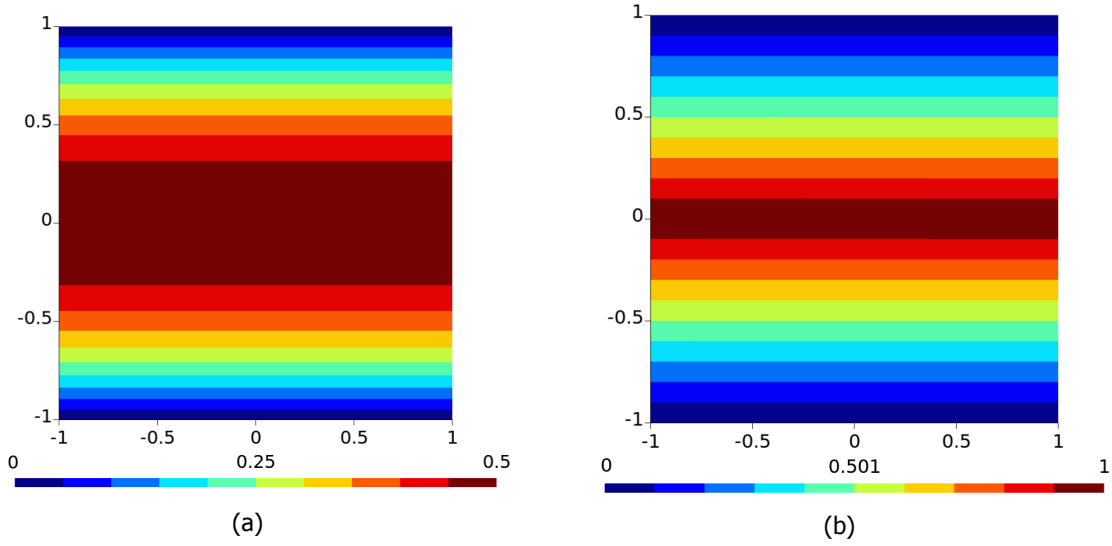


Figure 3.7: The numerical solution of the p-Poisson equation using $p = 3$ applied to a square domain after 100 Picard iterations on the left and 101 on the right.

$K = |\nabla u_p|^{p-2}$, $K = |\nabla u_p|^{p-2} + 0.1$ is used. The results are shown in Figure 3.8. It can be observed that there is practically no difference between two consecutive Picard iterations. Moreover, the relative L_2 error between these two solutions, where the highest iteration is considered as the exact solution, is 4×10^{-7}, which is considered sufficient as everything below $0.1\,\%$ is labelled converged.

In order to be sure that the convergence of the Picard iteration not only works for the 1D domain, the same calculations are executed as done for the results shown in Figure 3.8, only now a circular domain is used. The results are shown in Figure 3.9. As can be seen, the Picard iteration converges again, which can also be concluded from the relative L_2 error, which is 1×10^{-7}.

From the last two paragraphs it can be concluded that if Picard iteration is used to solve the p-Poisson equation using parameter two, zero values in the derivative cause oscillation of the Picard iteration solution, whereas when the zero values are bypassed, the Picard iteration converges.

3.2.3. Possible Solutions to the Problem

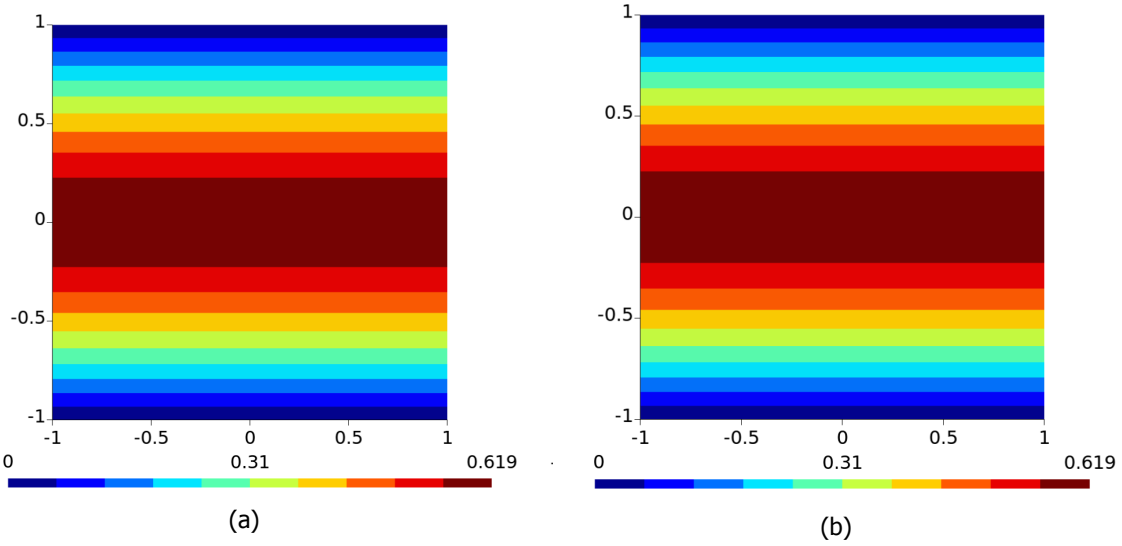


Figure 3.8: The numerical solution of the p-Poisson equation, where the diffusion coefficient $|\nabla u_p|^{p-2}$ is lifted by 0.1, using $p = 3$ applied to a square domain after 100 Picard iterations on the left and 101 on the right.

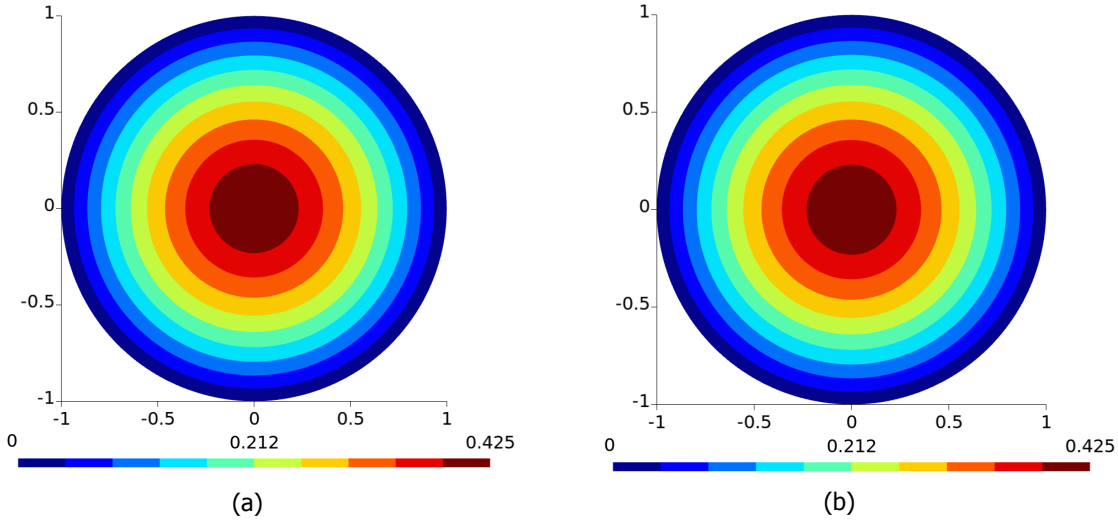


Figure 3.9: The numerical solution of the p-Poisson equation, where the diffusion coefficient $|\nabla u_p|^{p-2}$ is lifted by 0.1, using $p = 3$ applied to a circular domain after 100 Picard iterations on the left and 101 on the right.

Lifting the Diffusion Coefficient

The first solution evaluated is the lifting of the diffusion coefficient $|\nabla u_p|^{p-2}$. In the previous section, it was already shown that this method solves the oscillation problem. To test if the lifting method is also able to solve the p-Poisson equation using $p = 4$, the same calculations are executed as done for the results shown in Figure 3.8. Oscillation is observed in this case and also when the calculations are repeated utilizing an offset of 0.5. On the other hand, with an offset of 0.7 the relative L_2 error is 1×10^{-6}; thus this method is labeled as succeeded.

Gradual Increasing of the p-Poisson parameter

The next method tries to increase p more gradually, so instead of increasing it with 1, it is increased with an decimal number e.g. 0.1. With this method, the initial solution of the Picard iterations will be the solution to the previous iteration. When the same case is considered as is done for the results shown in Figure 3.6, only now p is increased with steps of 0.1 from 2 up to and including 3, the Picard iterations converge for $p < 3$, while it still oscillates for $p = 3$, only now between two solutions that are

relatively close, since the relative L_2 error between them is 2×10^{-2}. When the upper limit of the p is set to 4 instead of 3, it is observed that for the solutions of $p > 3$ the Picard iteration diverges. Even when the number of Picard iterations is set to 100, the solution of the last iteration of $p = 2.9$ has a relative L_2 error of 8×10^{-7} when compared to the solution of the previous iteration, while the Picard iteration of $p = 3$ oscillates and $p > 3$ the solution diverges. When the same procedure is repeated, only now the first Picard iteration cycle with a $p > 3$ uses the parameter value 3.01, divergence is observed. So the technique of gradually increasing p does not solve the oscillation problem and it is also not able to produce solutions using a p larger than 3.

Clipping the Diffusion Coefficient

Inspired by the lifting of the diffusion coefficient, a clipping method was investigated, which means that the diffusion coefficient has a minimal value it can attain. This method is also used in Wank [21] for obtaining a stable p-Poisson solver. When the same is done as for Figure 3.8, only now the diffusion coefficient is clipped instead of lifted by 0.1, the relative L_2 error between the 100-th and 101-th Picard iteration is 4×10^{-1}, which is not considered as converged. The 100-th and 101-th Picard iteration using the clipping method are shown in Figure 3.10.

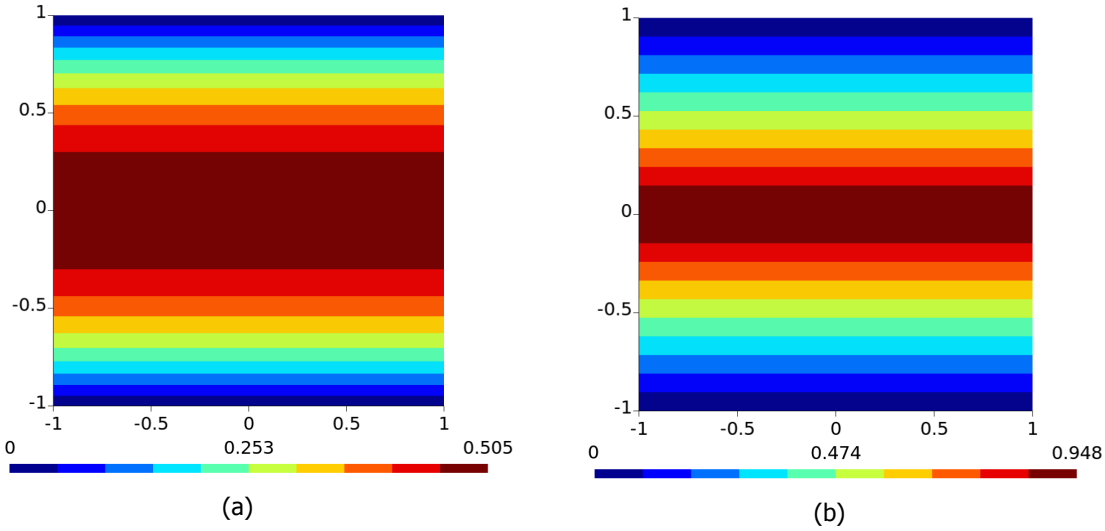


Figure 3.10: The numerical solution of the p-Poisson equation, where the diffusion coefficient $|\nabla u_p|^{p-2}$ is clipped at the value 0.1, using $p = 3$ applied to a square domain after 100 Picard iterations on the left and 101 on the right.

So the clipping method does not converge at the same rate as the lifting of the diffusion coefficient, which could possibly be due to the lack of smoothness introduced by clipping. A more sophisticated clipping method is given by

$$K = \begin{cases} \dfrac{|\nabla u_p|^{(p-2)\alpha}}{\alpha \beta^{\alpha-1}} + \beta \dfrac{\alpha - 1}{\alpha}, & |\nabla u_p|^{(p-2)} < \beta \\ |\nabla u_p|^{(p-2)}, & |\nabla u_p|^{(p-2)} \geq \beta, \end{cases} \tag{3.6}$$

where K is the new diffusion coefficient which is used for solving the p-Poisson equation and α and β are parameters. The parameter α must be larger than 1 and the parameter β must be larger than 0. Using this smooth clipping method, only values of $|\nabla u_p|^{p-2}$ below parameter β will be affected and the minimal value that the diffusion coefficient K can obtain is

$$\beta \frac{\alpha - 1}{\alpha}$$

at $|\nabla u_p|^{p-2}$ equals 0. This clipping method is chosen because when the diffusion coefficient $|\nabla u_p|^{p-2}$ equals the parameter β, the diffusion coefficient K also equals the parameter β and the derivative

of K w.r.t. $|\nabla u_p|^{p-2}$ equals 1, which provides smoothness of K. Furthermore, when $|\nabla u_p|^{p-2} = 0$, $\partial K / \partial |\nabla u_p|^{p-2} = 0$.

When the calculations done for Figure 3.10 are repeated, utilizing $\alpha = 2$ and $\beta = 0.1$, a relative L_2 error of 4×10^{-1} is obtained. With $\beta = 0.5$, the relative L_2 error is 3×10^{-1} and when the case of parameter $\beta = 1$ is calculated, shown in Figure 3.11, the relative L_2 error is 8×10^{-4}, which barely satisfies the convergence restriction. For $\alpha = 3$ and $\beta = 0.5$, the relative L_2 error is 3×10^{-1}.

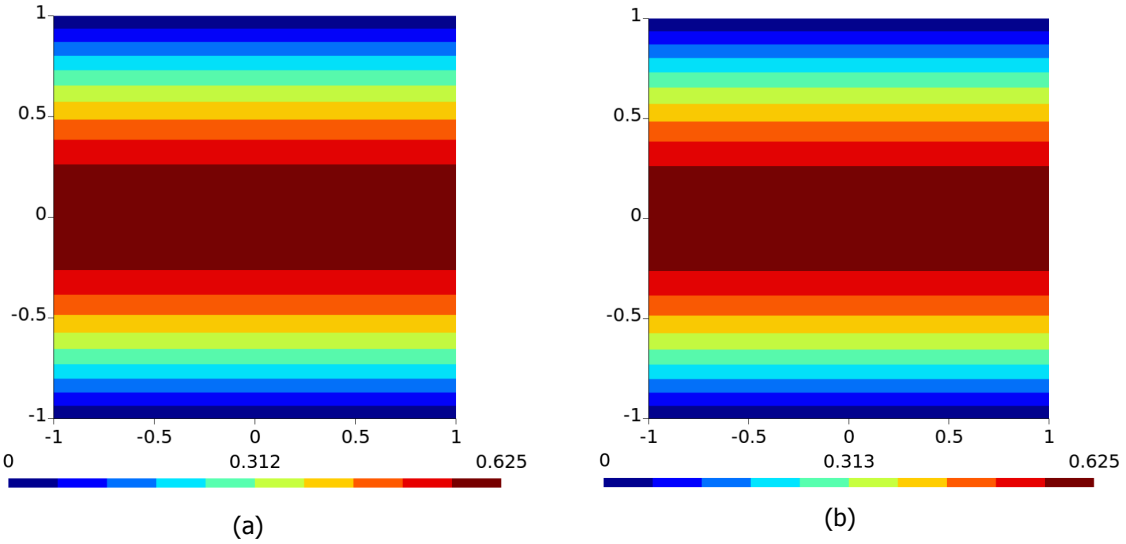


Figure 3.11: The numerical solution of the p-Poisson equation, where the diffusion coefficient $|\nabla u_p|^{p-2}$ is smoothly clipped using $\alpha = 2$ and $\beta = 1$ and $p = 3$ applied to a square domain after 100 Picard iterations on the left and 101 on the right.

To check if using the smooth clipping can generate a converged solution of the p-Poisson equation using $p = 4$, the same calculations done for Figure 3.11 are repeated. For this calculation, the solutions of the Picard iteration oscillate and when the calculations are repeated, only now using parameter $\alpha = 4$ and $\beta = 1.19$, the relative L_2 error is 2×10^{-5}. So the smooth clipping method is able to generate converged solutions to the p-Poisson equation using parameter 4.

Relaxation of the p-Poisson equation

Another method to possibly solve the oscillation problem is the relaxation of the p-Poisson equation by adding a time derivative of the approximate distance field u_p to the p-Poisson equation defined in equation (2.8). When the calculations done for Figure 3.6 are repeated, only now the 20-th and 21-th Picard iterations are compared, one obtains the results shown in Figure 3.12. For this calculation a time step of $0.1\,\mathrm{s}$ is used. The relative L_2 error between the two solutions shown in Figure 3.12 is 4×10^{-4}.

The relaxation problem can produce a converged solution to the p-Poisson equation using $p = 3$ and to test if it can also produce a converged solution using $p = 4$, the calculations done for Figure 3.12 are repeated. The relative L_2 error of the results form this calculation is 9×10^{-2}. When the calculations are repeated again, only now 501 Picard iterations are executed, the relative L_2 error is 7×10^{-2} and when the same is repeated with a time step of $1 \times 10^{-4}\,\mathrm{s}$ and 501 Picard iterations, the relative L_2 error is 1×10^{-3}. Therefore the relaxation method is able to produce a converged solution to the p-Poisson equation using $p = 4$.

Damping of the Picard Iteration Solutions

The last method described to possibly solve the oscillation problem is the damping of the approximate distance field u_p. The damping means that the solution generated by a Picard iteration is the weighted sum of the previous Picard iteration solution and the calculated solution of the Picard iteration itself, which is mathematically given by

$$\hat{u}_{p,\mathrm{Pic}=i+1} = \gamma u_{p,\mathrm{Pic}=i+1} + (1-\gamma)\,\hat{u}_{p,\mathrm{Pic}=i}, \tag{3.7}$$

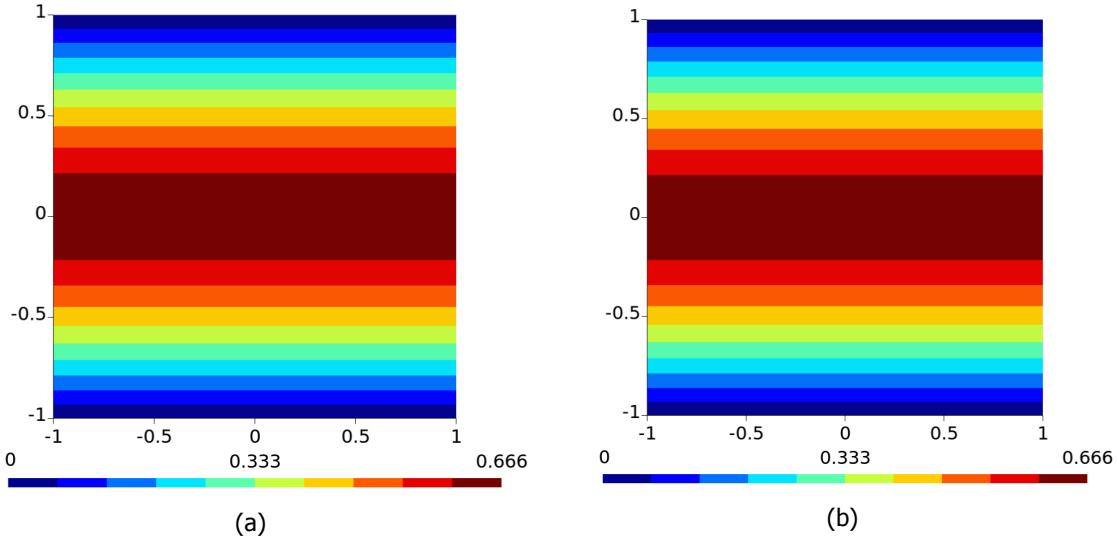


Figure 3.12: The numerical solution of the p-Poisson equation, where a time derivative is added using a time step of $0.1\,\mathrm{s}$ and $p = 3$ of a square domain after 20 Picard iterations on the left and 21 on the right.

where γ is the damping coefficient ($\gamma \in [0, 1]$) and $\hat{u}_{p,\mathrm{Pic}=i}$ is the damped solution of the i-th Picard iteration, which is used as input for the $i+1$-th Picard iteration. When the calculations done for Figure 3.6 are repeated, with $\gamma = 0.5$, one obtains the solutions shown in Figure 3.13. The relative L_2 error

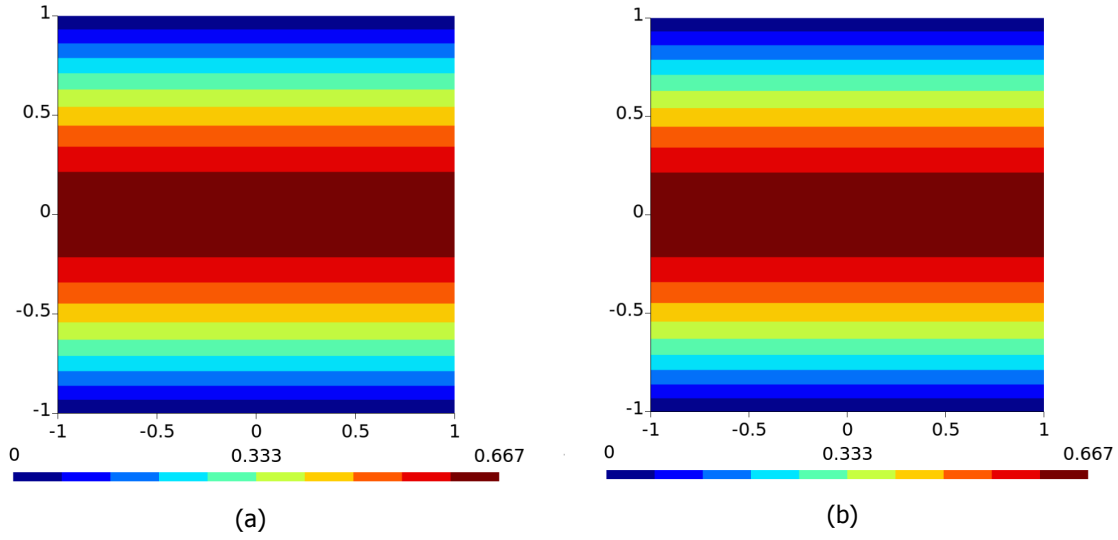


Figure 3.13: The numerical solution of the p-Poisson equation using $p = 3$ applied to a square domain after 10 Picard iterations on the left and 11 on the right. For the solution output of every Picard iteration damping is applied using $\gamma = 0.5$.

between the two solutions shown in Figure 3.13 is 3×10^{-7}.

To test if the damping method is able to produce a converged solution to the p-Poisson equation using $p = 4$, the calculations done for Figure 3.13 are repeated. The relative L_2 error between the 10-th and 11-th Picard iteration of the calculations is 2×10^{-4}, so the damping method is able to produce a converged solution to the p-Poisson equation using $p = 4$.

3.2.4. Testing the solutions to the Oscillation Problem

In Subsection 3.2.3, several methods for solving the the oscillation problem were described and, to recap, the methods that were labelled as succeeded are the lifting, clipping, relaxation and damping method.

Damping of the Picard Iteration Solutions

First the damping method was tested. This is done by calculating the relative L_2 error between the results shown in Figure 3.13(b) and the analytic solution to the p-Poisson equation using parameter 3, which is given by [20]

$$\tilde{u}_p = \frac{2}{3}\left(1 - |x|^{\frac{3}{2}}\right), \quad x \in [-1, 1]. \tag{3.8}$$

When this comparison is made, the calculated relative L_2 norm is 2×10^{-5}, which is considered as a good enough numerical approximation to the analytic solution. When the numerical solution to the p-Poisson equation using parameter 4 is compared with the analytic solution given by [20]

$$\tilde{u}_p = \frac{3}{4}\left(1 - |x|^{\frac{4}{3}}\right), \quad x \in [-1, 1], \tag{3.9}$$

the relative L_2 error is 4×10^{-5}, where the same settings are used as for the $p = 3$ case, only now the $p = 4$. When these calculations are repeated on a circular domain instead of a square domain, the relative L_2 error is 2×10^{-4}. The numerical solution of the last, 11-th, Picard iteration is shown in Figure 3.14 on the left along with the analytic solution to the p-Poisson equation using parameter 4 applied to a circular domain on the right, which is given by [20]

$$\tilde{u}_p = \frac{3}{4\sqrt[3]{2}}\left(1 - |x|^{\frac{4}{3}}\right), \quad x \in [-1, 1]. \tag{3.10}$$

With these three tests conducted it can be concluded that the damping method approached the analytic

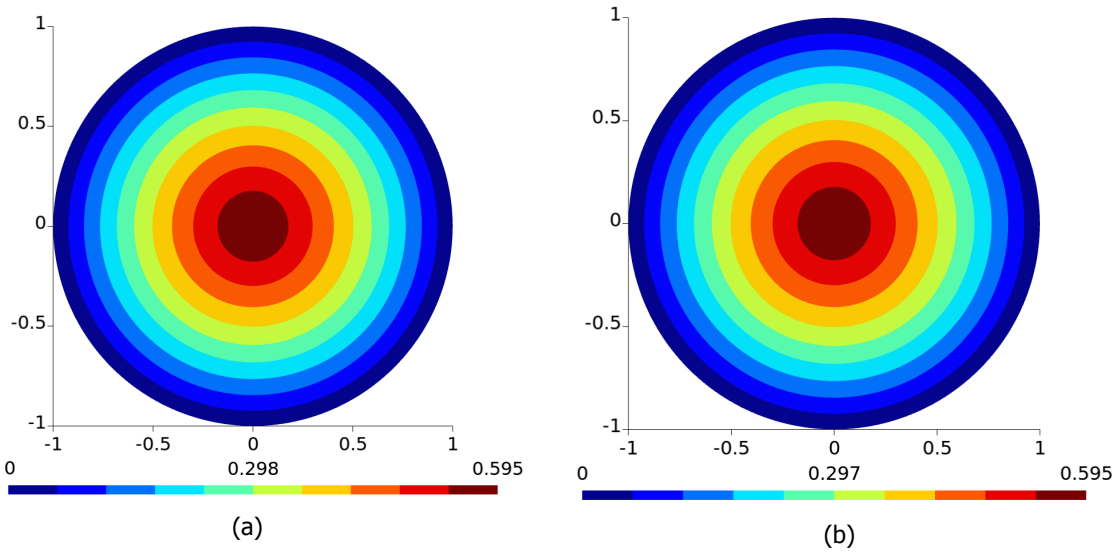


Figure 3.14: The numerical solution of the p-Poisson equation using $p = 4$ applied to a circular domain after 11 Picard iterations on the left. For the solution output of every Picard iteration damping is applied using $\gamma = 0.5$. On the right the analytic solution to this problem is shown.

solution with a small enough relative L_2 error, namely below $0.1\,\%$. Notice that the maximum number of Picard iterations allowed for other methods to be successful is 110, since only competitive methods are investigated.

Relaxation of the p-Poisson equation

Next, the relaxing method is tested. When the same calculations are done as for Figure 3.12, with $p = 4$ and 110 Picard iterations, no time step value can be found for which a converged solution is obtained to compare with the analytic result shown in equation (3.8). Either the code crashed due to a divergence in the solver or the relative L_2 error between two consecutive Picard iterations was larger than $0.1\,\%$. Thus, this solving method was discarded.

Clipping the Diffusion Coefficient

The third method that will be tested is the clipping method. When the same calculations are done as for Figure 3.11, with $p = 4$, 110 Picard iterations, $\alpha = 4$ and $\beta = 1.181$, the relative L_2 error between the 110-th Picard iteration and the analytic solution to this problem is 2×10^{-1}. The relative L_2 error between the solutions generated by the two last Picard iterations is 1×10^{-3}, which just meets the convergence restriction. Thus for these settings the numerical solution to the p-Poisson equation does not approach the analytic solution according to the restrictions. The solution belonging to the 110-th Picard iteration and the analytic solution to this problem are shown in Figure 3.15.

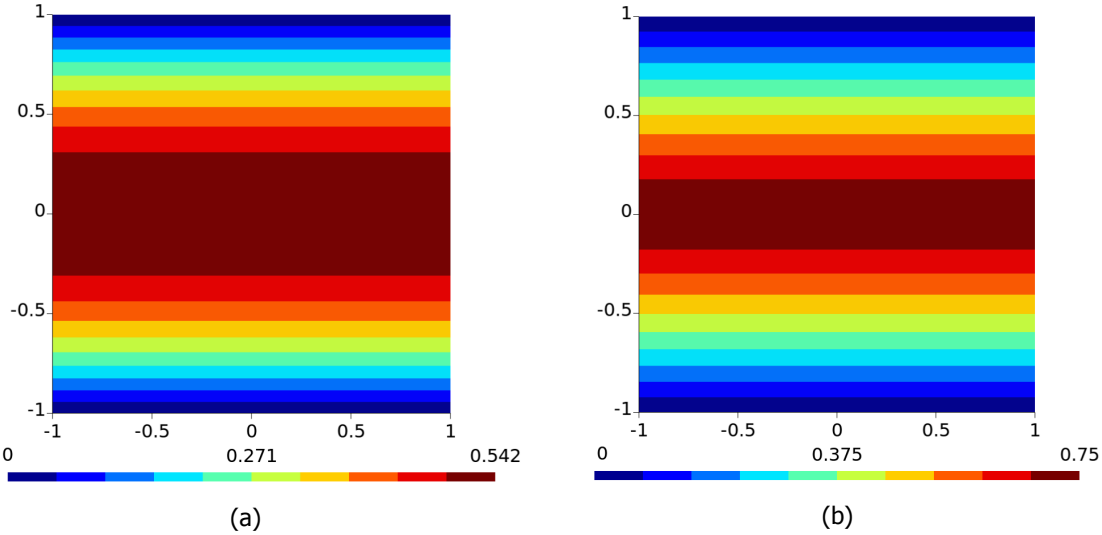


Figure 3.15: The numerical solution of the p-Poisson equation, where the diffusion coefficient $|\nabla u_p|^{p-2}$ is smoothly clipped using the parameters $\alpha = 4$ and $\beta = 1.181$ and $p = 4$ applied to a square domain after 110 Picard iterations on the left. The analytic solution to this problem is shown on the right.

When the calculations are repeated, with $\alpha = 3$ and $\beta = 1.276$, the relative L_2 error between the converged solution of Picard iteration 110 and the analytic solution to this problem, defined in equation (3.9), is 2×10^{-1}. Due to this relative L_2 error it can be concluded that the numerical approximation did not approach the analytic solution. Also when these calculations are repeated with the parameters $\alpha = 2$ and $\beta = 1.565$, the 110-th Picard iteration solution does not approach the analytic solution, because the relative L_2 error is 2×10^{-1}. Hereby it is concluded that the clipping method is not considered as succeeded.

Lifting the Diffusion Coefficient

The last method that will be considered is the lifting method. To test this method, the calculations done for Figure 3.8 are repeated, only now 110 Picard iterations are executed and $p = 4$. When the calculations are executed no converged solution is obtained and when the calculations are repeated again, only now the lifting value is set to 0.643, a relative L_2 error of 3×10^{-1} is obtained between the 110-th Picard and the analytic solution, defined in equation (3.9). Here the relative L_2 error between the 109-th and 110-th Picard iteration solution is 1×10^{-3}, so no lower lifting value can be used without violating the convergence restraint. Therefore the smooth clipping method is not considered as succeeded.

3.2.5. Conclusion

Concluding, only the damping method is considered as a successful, competitive, solution to the oscillation problem.

3.3. Error Analysis of the Wall-Distance Implementation

In this section the code implementation using the damping method for solving the distance field is tested, which means that the normalized numerical solution of the distance field generated by the

code is compared with the true distance field using the relative L_2 error. This will be done for multiple domains: a square, circular, and 1D domain. When potential errors are encountered, they are investigated and fixed. Finally, recommendations are given regarding which settings to choose, based on the results obtained.

3.3.1. One-Dimensional Domain

The true distance field of the 1D problem is given by

$$d(x) = 1 - |x|, \quad x \in [1-, 1]. \tag{3.11}$$

When this test is executed for $p = 2$, where one Picard iteration is used and damping is disabled, the same result is obtained as in Figure 3.1(b) and the relative L_2 error with the analytic solution is 1×10^{-7}. Note that here the solution is compared to the true distance field, whereas in Subsection 3.1.2 it was compared to the analytic solution of the normalized p-Poisson equation solution. When the $p = 3$ case is calculated, where 11 Picard iterations are executed along with $\gamma = 0.5$, the result shown in Figure 3.16(a) is obtained. In Figure 3.16 the result of the $p = 4$ calculation and true distance field is shown. The relative L_2 error between the solution shown in Figure 3.13(b) and the true distance field

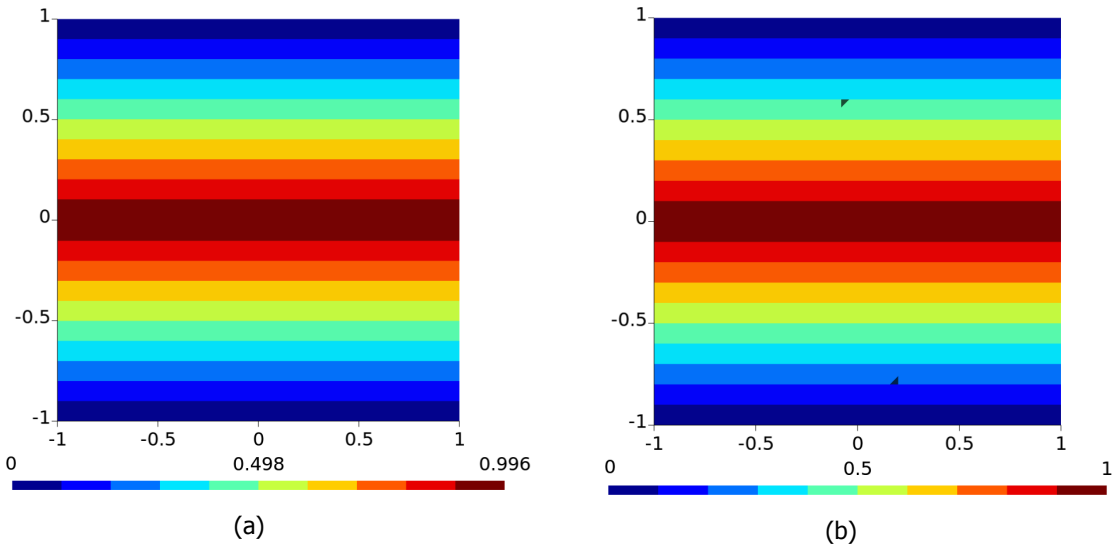


Figure 3.16: The normalization of the numerical solution of the p-Poisson equation using $p = 4$ on the left applied to a square domain after 11 Picard iterations. On the right the true distance field of a 1D domain is shown. For the solution output of every Picard iteration damping is applied using $\gamma = 0.5$.

is 5×10^{-4} and for the solution shown in Figure 3.16(a), it is 6×10^{-4}. Therefore it can be concluded that the code is able to reproduce the true distance field, because the relative L_2 error is smaller than $5\,\%$, which is considered as minimal relative L_2 error allowed when approximating the true distance field with a numerical solution.

Notice that there are two small triangles present in Figure 3.16(b), which are not expected, since the the true distance field given by equation (3.11) does not have them. When the data is studied no strange values are found around the triangles. Therefore it is suspected that the program Gmsh [22], with which the plots are produced, introduce this artificial effect.

3.3.2. Circular Domain

When the code is run for the circular domain, utilizing $p = 2$, where damping is disabled and 1 Picard iteration is executed, a relative L_2 error with the analytic solution, given by

$$d(\mathrm{x}) = 1 - |\mathrm{x}|, \quad |\mathrm{x}| \leq 1, \tag{3.12}$$

is 2×10^{-1}. The numerical solution produced by the code is the same as shown in Figure 3.5(b).

When the code is run with damping enabled, where $\gamma = 0.5$ is used, along with 11 Picard iterations and $p = 3$, the relative L_2 error with the true distance field is 1×10^{-1}. This is not considered as a small

enough error and when the same is done, only now with $p = 8$, 21 Picard iterations and $\gamma = 0.2$, the relative L_2 error is 5×10^{-2}, which confirms to the error requirement of $5\,\%$. $\gamma = 0.2$ is chosen for the reason that otherwise the Picard iteration would not converge for $p = 8$. The numerical solution to the distance field of the latter calculations is shown on the left of Figure 3.17 along with the true distance field on the right. One can see from Figure 3.17 that the numerical approximation shown on the left

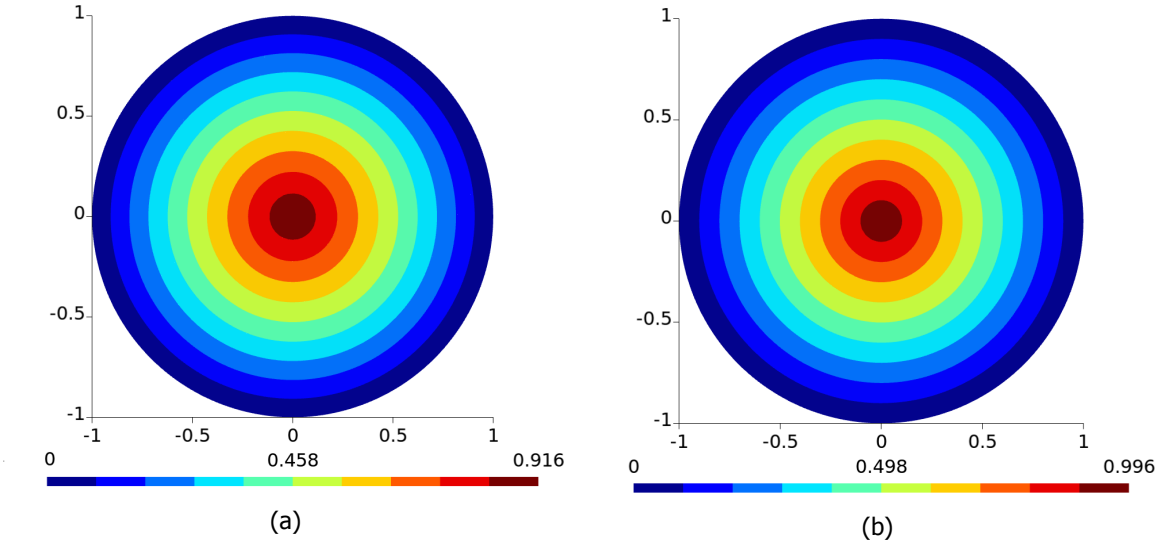


Figure 3.17: The normalization of the numerical solution of the p-Poisson equation using $p = 8$ on the left applied to a circular domain after 21 Picard iterations. On the right the true distance field of a circular domain is shown. For the solution output of every Picard iteration damping is applied using $\gamma = 0.2$.

deviates from the true distance field on the right, and that this difference in deviation increases further from the boundary. This is desirable for the application to the RANS model, because the distance close to the boundary is important and further away it is of less importance [23].

When the same calculations are done as for Figure 3.17(a), with 101 Picard iterations, $p = 20$ and damping coefficient 0.01, the relative L_2 error with the true distance field is 2×10^{-2}. To test if a change in spatial polynomial has any effect on the relative L_2 error, the $p_s = 4$ case is tested, where the relative L_2 error is 2×10^{-2}. Therefore it is assumed that the change in spatial polynomials does not affect the numerical solution significantly.

3.3.3. Square Domain

First the $p = 2$ case is tested, where the number of Picard iterations is set to 1 and damping is disabled. With the test executed, the same result as in Figure 3.3(b) is obtained and the relative L_2 difference between the true distance field is 2×10^{-1}. This error is higher than the relative L_2 error allowed.

When the calculations are repeated, only now utilizing $p = 3$ and 11 Picard iterations, the relative L_2 error is 8×10^{-2} and when $p = 4$ is used, the relative L_2 error is 5×10^{-2}. For $p = 3$ the error was lower than the requirement, but for the $p = 4$ case it satisfies this requirement. The numerical solution corresponding to the $p = 4$ case is shown in Figure 3.18 along with the true distance field.

Just like in the case of the true distance field shown in Figure 3.16(b), there are also graphical defects present, like asymmetry, in the results shown in Figure 3.18, which are introduced by Gmsh [22]. Furthermore, from the numerical solution shown in Figure 3.18(a) it can be seen that in this case the deviation of the numerical solution from the true distance field shown in 3.18(b) increases further from the boundary. This was also the case for the numerical solution for a circular domain shown in Figure 3.17(a).

3.3.4. Recommended Settings

With three cases analyzed, the minimal required parameter values are 8 for the p-Poisson parameter, 0.2 for the damping coefficient and 20 for the number of Picard iterations. The recommended parameter values are 10 for the p-Poisson parameter, 0.1 for the damping coefficient and 30 for the number of

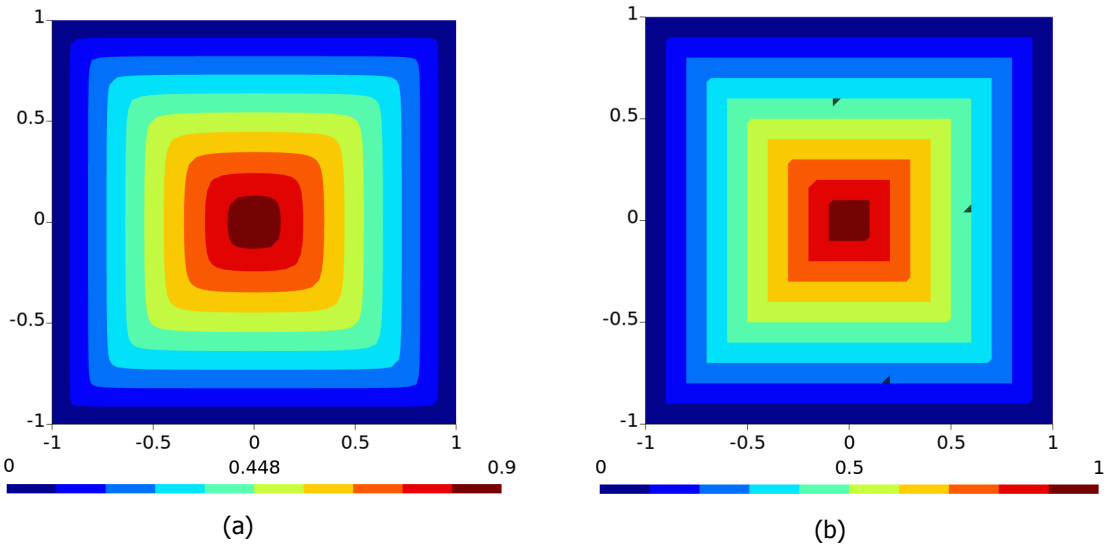


Figure 3.18: The normalization of the numerical solution of the p-Poisson equation using $p = 4$ on the left applied to a square domain after 21 Picard iterations. On the right the true distance field of a square domain is shown. For the solution output of every Picard iteration damping is applied using $\gamma = 0.2$.

Picard iterations, to better safeguard converged solutions that approach the true distance field within the $5\,\%$ relative L_2 error requirement.

4

Distance Field of Molten Salt Fast Reactor

In this chapter, the distance field of the MSFR is calculated for different settings. Firs, a test is done with the minimal requirements suggested in Subsection 3.3.4 and, afterwards, with the recommended settings stated there. In addition to the minimal suggested settings, a mesh refinement study is preformed. To study the MSFR, $1/16$-th of the core is studied, of which the mesh is from van Nijen [24]. A graphical approximation of the MSFR is displayed in Figure 4.1. The mesh to mimic the MSFR

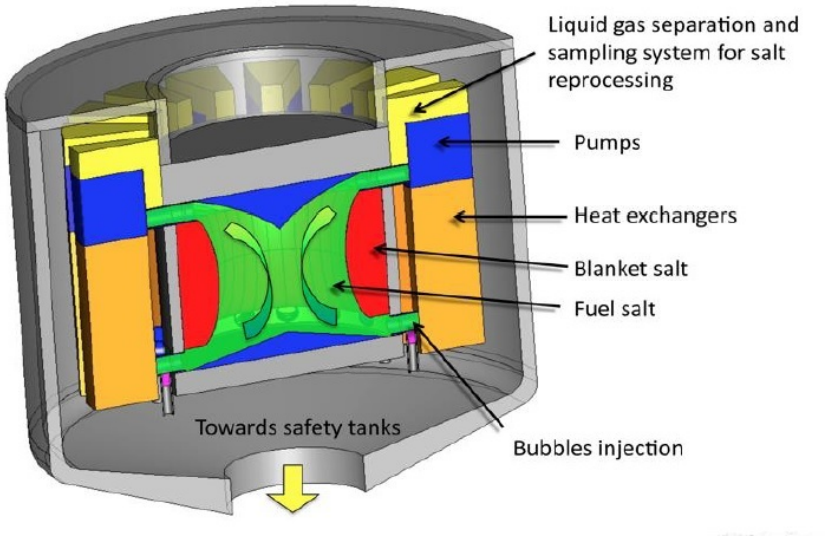


Figure 4.1: The molten salt fast reactor core design, where 16 pumps together with heat exchangers are present [24].

is 3D, instead of all the 2D meshes used up till now in this thesis. Furthermore, all the boundaries of the MSFR domain will satisfy the Dirichlet boundary condition stated in equation (2.8), except the sides of the inner core. They confirm to the Neumann boundary condition stated in equation (2.9), because symmetry is imposed there. At last, a comparison and conclusion is made between the three distance fields calculated in this chapter.

4.1. Minimal Suggested Settings

In this section the distance field of the MSFR is calculated using the minimal suggested settings stated in 3.3.4. The numerical result is shown in Figure 4.2 and the mesh used for this calculations consists of 27871 tetrahedra. The relative L_2 error between the two last Picard iterations is 4×10^{-5}. Additionally, a cut plane of the results shown in Figure 4.2 is shown in Figure 4.3.

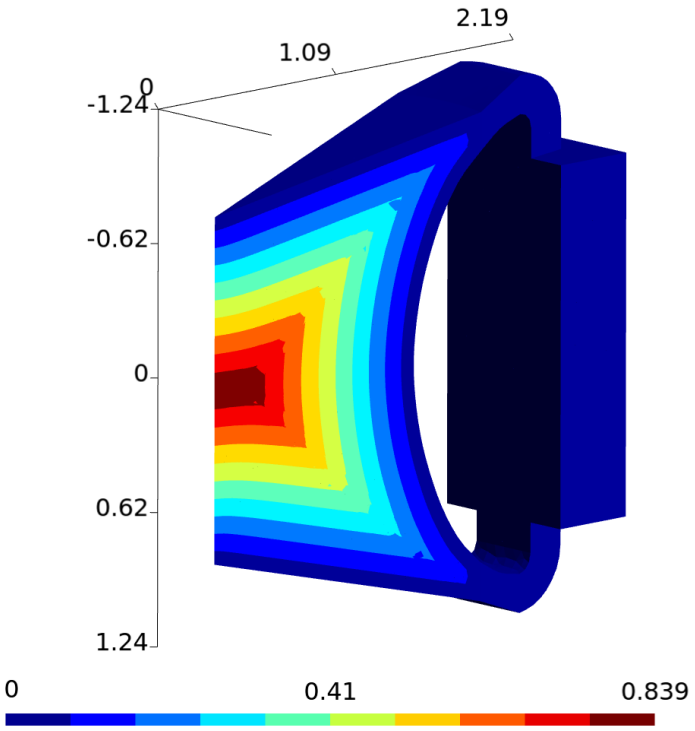


Figure 4.2: The 3D view of the numerical solution of the distance field applied to the molten salt fast reactor. This numerical distance field is created with the p-Poisson equation using $p = 8.\ 20$ Picard iterations and $\gamma = 0.2$ are used to solve the numerical solution.

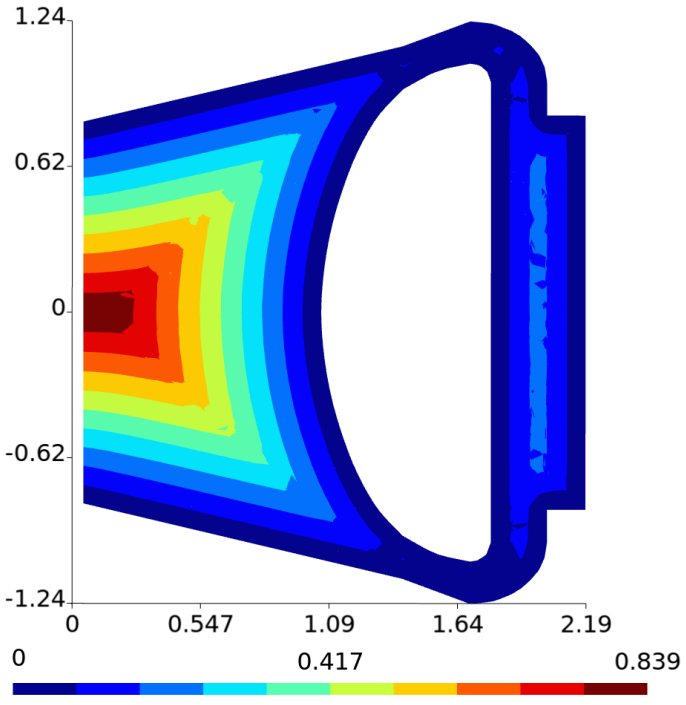


Figure 4.3: A 2D cut plane of the 3D numerical solution of the distance field applied to the molten salt fast reactor. This numerical distance field is created with the p-Poisson equation using $p = 8.\ 20$ Picard iterations and $\gamma = 0.2$ are used to solve the numerical solution.

When inspecting the plane shown in Figure 4.3, it is concluded that in the pump and heat exchanger the distance field is not as expected. This is due to the lack of straight contour lines that are expected,

because the pump and heat exchanger are an extracted trapezoid. To show the variation of the distance field in the pump and heat exchanger, the iso-value $0.16\,\mathrm{m}$ within them is shown in Figure 4.4. It can indeed be seen that the contour graph supports the assumption that the contour lines shown in Figure 4.3 are not straight. However, the core of the MSFR shows a somewhat correct distance field, since the shape of the distance field is as expected, but the numerical values are not quite what they should be. For example, in the centre of the core the distance field should be $0.801\,\mathrm{m}$, but it is $0.82\,\mathrm{m}$ and at the point $(0, 0, 0.62)$, the distance field should be $0.1801\,\mathrm{m}$, but it is $0.195\,\mathrm{m}$.

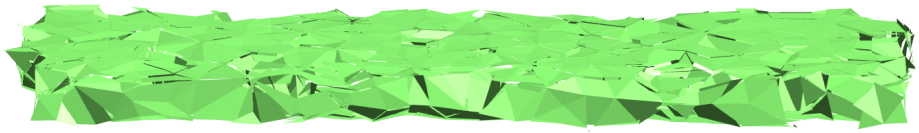


Figure 4.4: The iso-value $0.16\,\mathrm{m}$ of the pump and heat exchanger of the 3D numerical solution of the distance field applied to the molten salt fast reactor. This numerical distance field is created with the p-Poisson equation using $p = 8.\ 20$ Picard iterations and $\gamma = 0.2$ are used to solve the numerical solution.

4.2. Recommended Settings

Here the distance field of the MSFR is calculated utilizing the recommended settings from Subsection 3.3.4. When this calculation is done, the result shown in Figure 4.5 is obtained. The relative L_2 error of the solution of the last and second Picard iteration is 6×10^{-5}. Furthermore, a cut plane is showed in Figure 4.6.

With the plane inspected, shown in Figure 4.6, the same behavior is observed as for the required settings. Here the distance field is $0.819\,\mathrm{m}$ at the centre and $0.194\,\mathrm{m}$ at $(0, 0, 0.62)$.

4.3. Refined Mesh

Here first the distance field of the MSFR is calculated using the minimal suggested settings along with a finer mesh compared to the mesh used in Section 4.1. Second only within the pump and heat exchanger the mesh will be refined, so a greater refinement can be obtained in comparison with the previous mentioned mesh refinement, without running into memory issues.

Using the minimal suggested settings along with global mesh refinement, the calculated distance field is shown in Figure 4.7 and a cut plane is shown in Figure 4.8, where the mesh used consists of 46882 tetrahedra. The relative L_2 error between the last two Picard iterations is 4×10^{-5}. From the plane shown in Figure 4.8 it is concluded that here the same behavior is observed as for the non-refined mesh. The distance field at the centre is $0.816\,\mathrm{m}$ and at $(0, 0, 0.62)$ it is $0.192\,\mathrm{m}$. The iso-value of $0.16\,\mathrm{m}$ in the pump and heat exchanger is shown in Figure 4.9.

When only the pump and heat exchanger are refined compared to the mesh used in Section 4.1, using the suggested minimal settings, the result shown in Figure 4.10 is obtained and a cut plane is shown in Figure 4.11. The mesh used for this calculation consists of 47954 tetrahedra and the maximum L_2 error between the two last Picard iterations is 3×10^{-5}. From the plane shown in Figure 4.11 it can be observed that here the distance field is closer to the expected distance field within the pump and heat exchanger, compared to the result shown in Figure 4.3. The iso-value of $0.16\,\mathrm{m}$ of the distance field within the pump and heat exchanger is shown in Figure 4.12

4.4. Comparison Between Settings

With the numerical calculations done for four different cases, it can quantitatively be concluded that they show the same behavior. They all have the correct distance field in the tub, but they have a non-expected distance field in the pump and heat exchanger. The maximum value of the numerical distance fields shown in Figure 4.2 and 4.5 differ less than $1\,\%$, even though they do not share the same p-Poisson parameter at all. Also the defect of these distance fields in the pump and heat exchanger is practically the same. Therefore it is concluded that the minimal and recommended settings produce the same result in the case of the MSFR.

Regarding the refined mesh solution shown in Figure 4.7, the strange behavior of the distance field

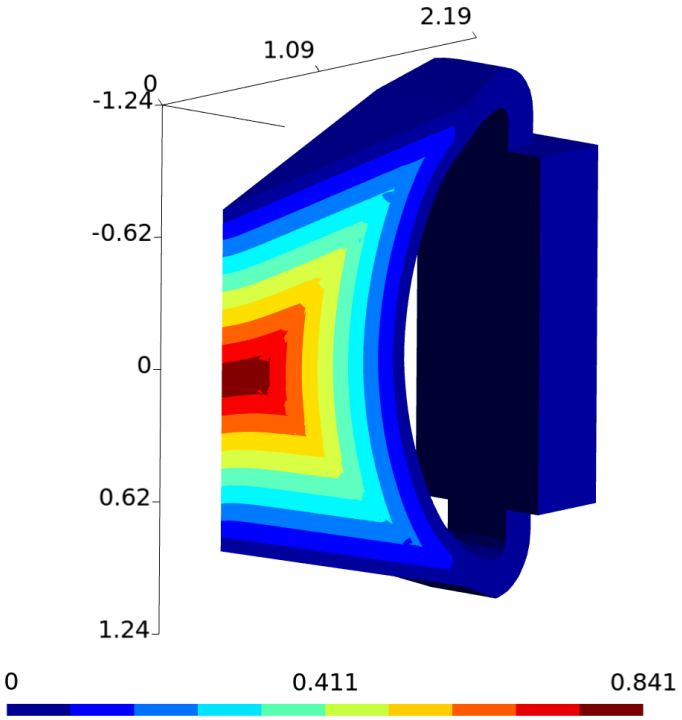


Figure 4.5: The 3D view of the numerical solution of the distance field applied to the molten salt fast reactor. This numerical distance field is created with the p-Poisson equation using $p = 10.30$ Picard iterations and d$\gamma = 0.1$ are used to solve the numerical solution.

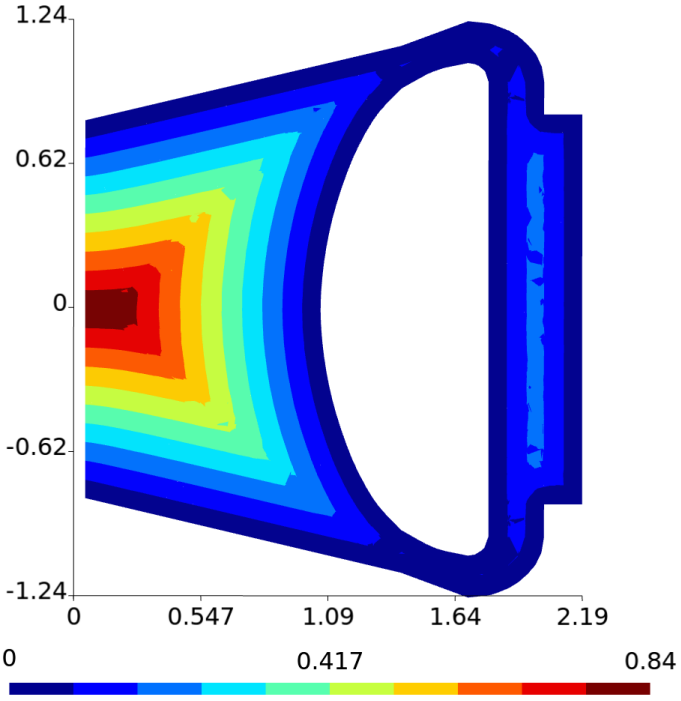


Figure 4.6: A 2D cut plane of the 3D numerical solution of the distance field applied to the molten salt fast reactor. This numerical distance field is created with the p-Poisson equation using $p = 10.30$ Picard iterations and $\gamma = 0.1$ are used to solve the numerical solution.

in the pump and heat exchanger is present, but is somewhat different than the strange behavior of the distance field presented in Figure 4.2. When only the pump and heat exchanger are refined, at a greater level than the global refinement used for the previous case, the strange behavior is suppressed.

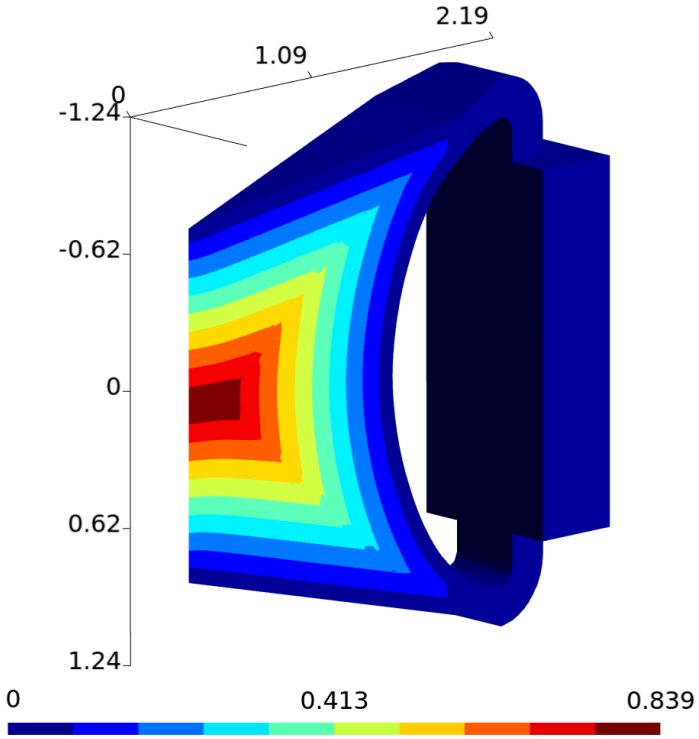


Figure 4.7: The 3D view of the numerical solution of the distance field applied to the molten salt fast reactor. This numerical distance field is created with the p-Poisson equation using $p = 8.\ 20$ Picard iterations and $\gamma = 0.2$ are used to solve the numerical solution.

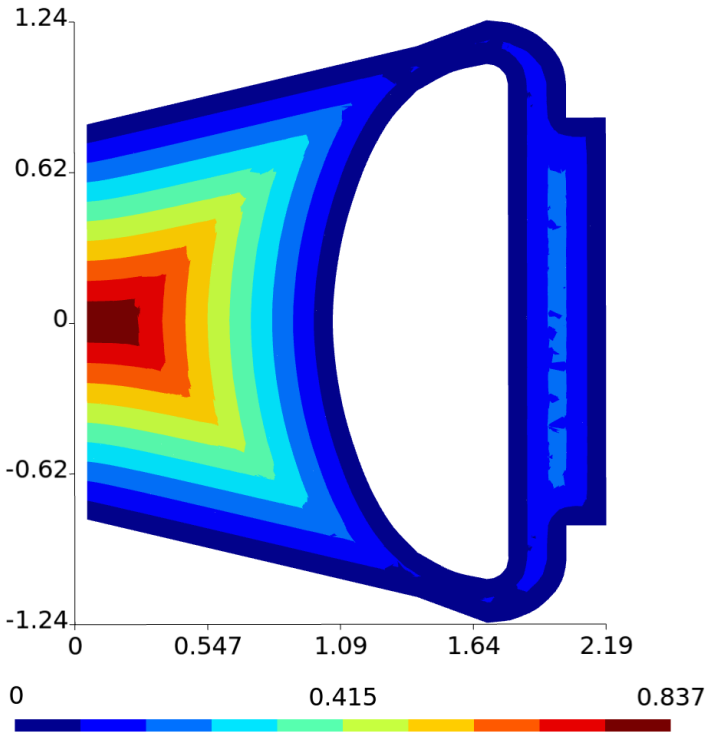


Figure 4.8: A 2D cut plane of the 3D numerical solution of the distance field applied to the molten salt fast reactor. This numerical distance field is created with the p-Poisson equation using $p = 8.\ 20$ Picard iterations and damping using coefficient 0.2 are used to solve the numerical solution.

Therefore it is advised to locally increase the mesh if there exist unexpected behavior.

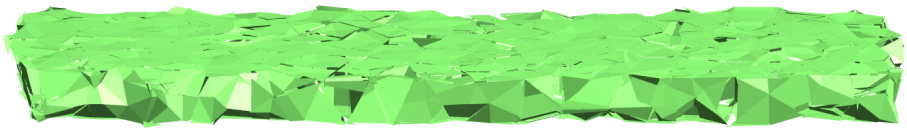


Figure 4.9: The iso-value $0.16\,\mathrm{m}$ of the pump and heat exchanger of the 3D numerical solution of the distance field applied to the molten salt fast reactor. This numerical distance field is created with the p-Poisson equation using $p = 8.\ 20$ Picard iterations and $\gamma = 0.2$ are used to solve the numerical solution.

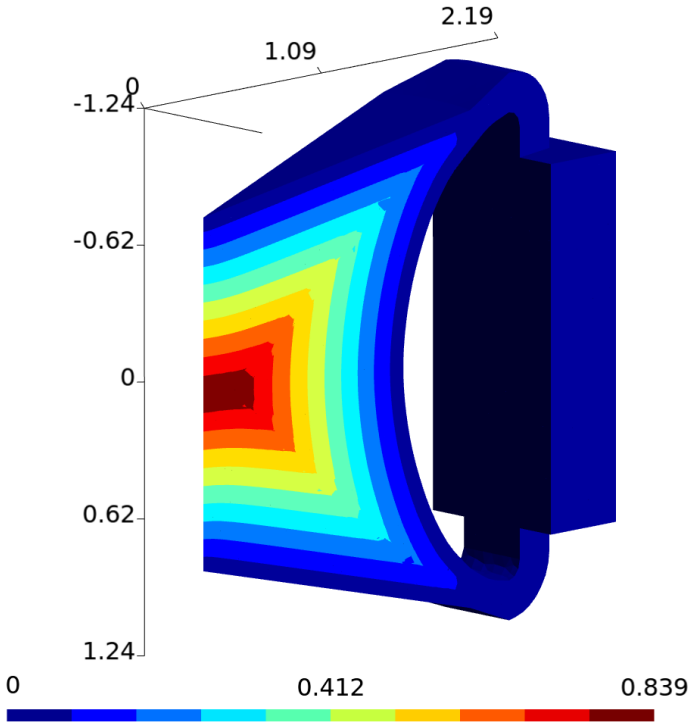


Figure 4.10: The 3D view of the numerical solution of the distance field applied to the molten salt fast reactor. This numerical distance field is created with the p-Poisson equation using $p = 8.\ 20$ Picard iterations and $\gamma = 0.2$ are used to solve the numerical solution.

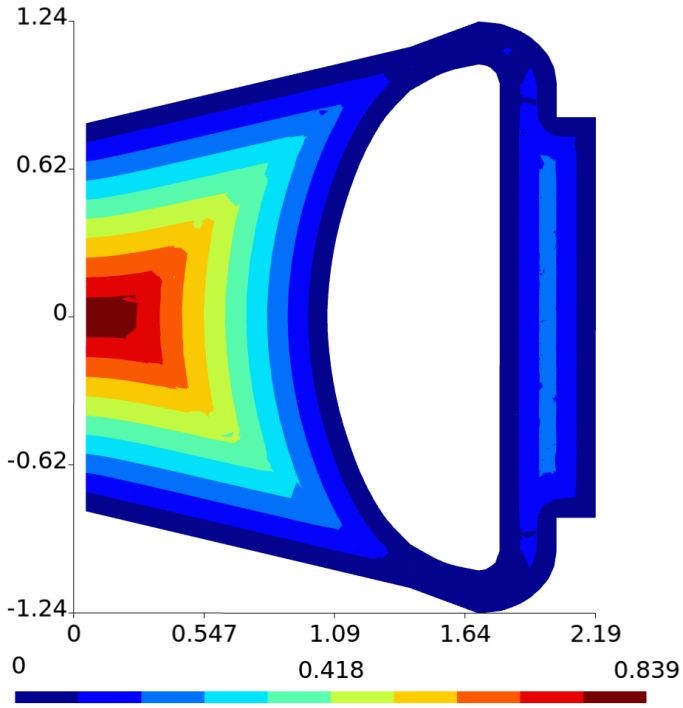


Figure 4.11: A 2D cut plane of the 3D numerical solution of the distance field applied to the molten salt fast reactor. This numerical distance field is created with the p-Poisson equation using $p = 8.\ 20$ Picard iterations and $\gamma = 0.2$ are used to solve the numerical solution.

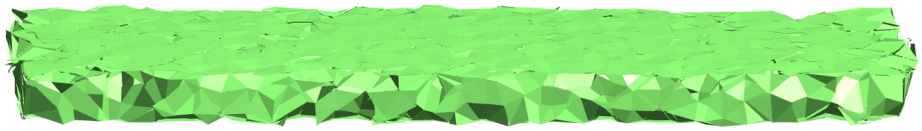


Figure 4.12: The iso-value $0.16\,\mathrm{m}$ of the pump and heat exchanger of the 3D numerical solution of the distance field applied to the molten salt fast reactor. This numerical distance field is created with the p-Poisson equation using $p = 8.\ 20$ Picard iterations and $\gamma = 0.2$ are used to solve the numerical solution.

5

Conclusions and Recommendations

In this chapter, first the research goal is answered, along with findings made when implementing and testing the code. Afterwards recommendations are given for further research.

5.1. Research Question

The research question stated in Chapter 1 was:

1. What numerical solvers can be used to calculate the distance field to be utilized for the RANS equations?
2. Which of these methods is most suited?

Subsequently, the most suited method is implemented in the code, and this implementation is tested and improved. In this section the answers, findings and results of this research question are briefly described.

Several numerical distance field solvers where investigated, such as the fast marching and p-Poisson method, which have among the highest accuracy approximating the distance field. The p-Poisson method is chosen to implement, since it consists of solving a convection-diffusion equation, called the p-Poisson equation and a normalization.

When the p-Poisson equation using parameter 2 is solved for a 1D and circular domain, the relative L_2 errors compared with the corresponding analytic solutions to the p-Poisson equation is smaller than $0.1\,\%$. This is considered as a small enough error for the numerical solution to an equation. On the square domain the p-Poisson equation using parameter 2 also produced the anticipated result.

For the case where the p-Poisson parameter is larger than 2, the p-Poisson equation becomes non-linear. Since the solver in the basecode is only able to solve for linear equations, Picard iteration is used in order to solve the equation. When the p-Poisson parameter 3 is used, oscillation is observed. To solve this problem, damping of the solution is used.

When comparing the normalized solution to the p-Poisson equation with the true distance field via the relative L_2 error on simple domains, it is concluded that the p-Poisson parameter 8, 20 Picard iterations and damping coefficient 0.2 should be minimally used. With these minimal settings, a relative L_2 error smaller than $5\,\%$ is obtained, which is considered as good enough. Because complex domains are possibly harder to solve than simple domains, it is suggested to use the p-Poisson parameter 10 along with 30 Picard iterations and damping coefficient 0.1.

When the distance field of an approximate MSFR is calculated, the minimal and suggested settings produce practically the same result. After inspection it was concluded that the distance fields were as expected, except for one place in the domain. There the distance fields were rough, whereas it should be smooth. Therefore local refinement of the mesh was applied at this part of the domain, which resulted in a distance field that conforms to the expectations, when the minimal settings calculation was repeated. Therefore it is suggested to use local refinement at spots where the distance field does not behave as expected.

5.2. Recommendations

There where several things that were not investigated in this thesis that could be investigated. One of these is the use of the fast marching method to calculate the distance field instead of the p-Poisson method. This method could possibly lead to less computational effort and better numerically calculated distance fields. The implementation of Newton's method instead of Picard iteration could also be investigated, to look whether this decreases the computational effort and increases the accuracy of the numerical distance field.

For this thesis, there was no substantiated reasoning to choose for the $5\,\%$ error requirement. Maybe the RANS model must have a distance field with a relative L_2 error lower than $1\,\%$ or should the error allowed near the wall be different from far away from it.

In this thesis there is no full quantitative error analysis of a complex domain. Therefore, the performance of the p-Poisson method applied to complex domains could be relative poor compared to the simple domains. This implies that the p-Poisson method could not be suitable for complex domains, like the MSFR, although it looks like that the p-Poisson method is able to produce a satisfying distance field of the MSFR. Thus, one can look for/make a complex domain from which the true distance field is known in order to test the p-Poisson method.

In the implementation of the p-Poisson method created for this thesis, the parameters like the damping coefficient and number of Picard iterations need to be inserted in advance of the calculations. With the relative L_2 error available, it should be possible to automate this. Then the code will look if a converged solution is obtained on the basis of the error, so no manual input is required.

A far-fetched addition to the implementation of the code is the addition of an algorithm that is able to spot where to refine the mesh, so that the mesh can be refined at that point. With information of the boundary, the implementation could search for strange behavior of the distance field near the boundary, because there the distance field should have approximately the same shape as the boundary. An example of this is the MSFR, where in the pump and heat exchanger the distance field does not have straight contour lines, whereas it is expected there due to the straight, neighbouring, boundaries.

Concerning the undesired behavior of the distance field withing the heat exchanger and pump of the MSFR, a greater mesh refinement than done is this thesis could not be executed, because the computer available did not had the resources to calculate it. Since no further refined mesh could be used, it is not demonstrated that the distance field really converges to straight contour lines, although it is expected, looking at the presented mesh refinement. So with a computer capable of further mesh refinement, one could look at the behavior of the distance field at certain locations using a finer mesh.

Bibliography

[1] P. Rubiolo, D. Heuer, E. Merle, M. Brovchenko, V. Ghetta, M. Allibert, and A. Laureau, *Overview and Perspectives of the Molten Salt Fast Reactor (MSFR) Concept,* (2013).

[2] P. A. Durbin, *A Reynolds stress model for near-wall turbulence,* Journal of Fluid Mechanics **249**, 465 (1993).

[3] B. Blocken, T. Stathopoulos, and J. Carmeliet, *CFD simulation of the atmospheric boundary layer: wall function problems,* Atmospheric Environment **41**, 238–252 (2007).

[4] D. M. Hargreaves and N. G. Wright, *On the use of the k– model in commercial CFD software to model the neutral atmospheric boundary layer,* Journal of Wind Engineering and Industrial Aerodynamics **95**, 355–369 (2007).

[5] S. Fan, B. Lakshminarayana, and M. Barnett, *Low-Reynolds-number k-epsilon model for unsteady turbulent boundary-layer flows,* AIAA Journal **31**, 1777–1784 (1993).

[6] W. Rodi, *Experience with two-layer models combining the k-epsilon model with a one-equation model near the wall,* 29th Aerospace Sciences Meeting (1991), 10.2514/6.1991-216.

[7] N. A. Wukie and P. D. Orkwis, *A p-Poisson wall distance approach for turbulence modeling,* in *23rd AIAA Computational Fluid Dynamics Conference* (2017).

[8] P. G. Tucker, *Differential equation-based wall distance computation for DES and RANS,* Journal of Computational Physics **190**, 229–248 (2003).

[9] M. Tiberga, *Three-dimensional, multi-physics simulation of a Molten Salt Fast Reactor,* (2016), first year report.

[10] R. Hartmann, *Numerical Analysis of Higher Order Discontinuous Galerkin Finite Element methods,* (2008).

[11] K. Shahbazi, *A parallel high-order discontinuous Galerkin solver for the unsteady incompressible Navier-Stokes equations in complex geometries*, Ph.D. thesis, University of Toronto (2007).

[12] J. A. Sethian and A. Vladimirsky, *Fast methods for the Eikonal and related Hamilton- Jacobi equations on unstructured meshes,* Proceedings of the National Academy of Sciences **97**, 5699–5703 (2000).

[13] M. W. Jones and M. Chen, *A New Approach to the Construction of Surfaces from Contour Data,* Computer Graphics Forum **13**, 75–84 (1994).

[14] J. A. Sethian, *A fast marching level set method for monotonically advancing fronts,* Proceedings of the National Academy of Sciences **93**, 1591–1595 (1996).

[15] A. Telea, *An Image Inpainting Technique Based on the Fast Marching Method,* Proceedings of the National Academy of Sciences **9**, 25 (2004).

[16] A. Telea and J. J. v. Wijk, *An Augmented Fast Marching Method for Computing Skeletons and Centerlines,* in *Eurographics / IEEE VGTC Symposium on Visualization*, edited by D. Ebert, P. Brunet, and I. Navazo (The Eurographics Association, 2002).

[17] S. Fisher and M. C. Lin, *Deformed Distance Fields for Simulation of Non-Penetrating Flexible Bodies,* in *Computer Animation and Simulation 2001*, edited by N. Magnenat-Thalmann and D. Thalmann (Springer Vienna, Vienna, 2001) pp. 99–111.

[18] A. G. Belyaev and P.-A. Fayolle, *On Variational and PDE-Based Distance Function Approximations,* Computer Graphics Forum **34**, 104 (2015), https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12611 .

[19] *MATLAB*, The MathWorks Inc., Natick, MA, 2000 (2018), R2017b.

[20] A. Belyaev and P.-A. Fayolle, *On modified Gordon-Wixom interpolation schemes and their applications to nonlinear and exterior domain problems,* Numerical Algorithms **77**, 691 (2018).

[21] M. Wank, *A Kačanov Type Iteration for the p-Poisson Problem*, dissertation, Universität Osnabrück (2016).

[22] C. Geuzaine and J.-F. Remacle, *Gmsh* (2018), 2.10.1.

[23] P. R. Spalart and S. R. Allmaras, *A one-equation turbulence model for aerodynamic flows,* 30th Aerospace Sciences Meeting and Exhibit (1992), 10.2514/6.1992-439.

[24] D. Nijen, *Investigation of natural circulation capabilities of the Molten Salt Fast Reactor*, Bachelor's thesis, Delft University of Technology (2018).