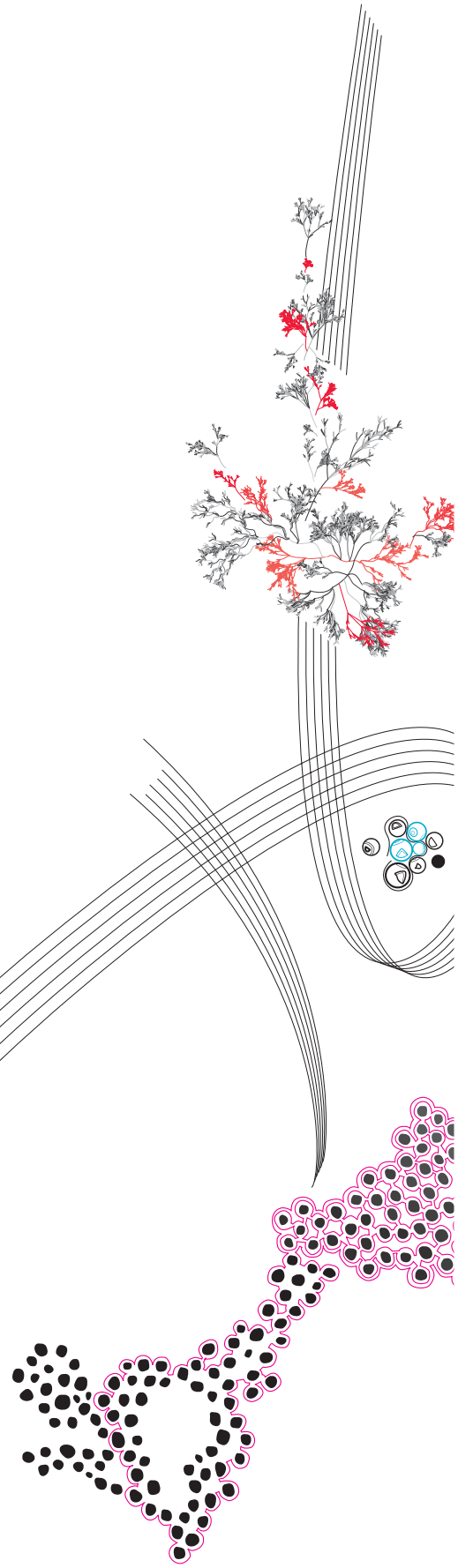

BSc Thesis Applied Mathematics

Solving the obstacle problem: Finite element simulation with a study on exact solutions

Abraham Jilles Jonkheer

Supervisor: Dr. D. Gallistl (Dietmar)

January 29, 2020

Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

Acknowledgements

This space I would like to use to thank everyone who has helped me during the time I worked on this BSc thesis.

First of all, I want to thank God for the way he has given me strength and perseverance throughout this past year, which has not been easy. He has really shown his faithfulness in this year, which I never in my life needed more than last year.

I want to thank Dietmar for his help on this research, because I was clueless on both the obstacle problem and finite element method before starting this. He coached me and guided me in the right way. In the end I have learned so much through this research, and I could not have completed it this successfully without him.

I also want to thank Roos, my wife, for supporting me through all the weeks of hard work I have put into this research. For just being there and pushing me to work on this thesis, especially when my motivation lacked.

Moreover, I want to thank my parents for always being supportive and always encouraging me to get the best out of myself. Also, I want to thank my parents-in-law for all the good times in Lelystad, where I could relax and always feel at home.

Finally, I want to thank all the people I worked with at the university. Peter, Anna, Arnout, Maik, Jasper, Rutger and Huan, thank you for just being there to work together and sometimes give a distraction from the work by playing a game or going for a walk.

Solving the obstacle problem: Finite element simulation with a study on exact solutions

A. J. Jonkheer (Bram)*

January 29, 2020

Abstract

In this paper, we focus on the elliptic and the parabolic obstacle problem. We will discuss the elliptic obstacle problem in both one and two dimensions. On the other hand, we will consider the parabolic obstacle problem in one dimension only. We use the finite element method and quadratic programming in MATLAB to simulate solutions. We provide the corresponding numerical implementation and show that there exist some elegant exact solutions to the elliptic obstacle problem in specific cases. The method we use could be a step towards finding more exact solutions of obstacle problems.

Keywords: Elliptic Obstacle Problem, Finite Element Method, Elasticity Theory, Modeling Non-Linear PDEs, Parabolic Obstacle Problem, Free Boundary Problem

1 Introduction

The obstacle problem is a classical problem in elasticity theory, in which a partial differential equation needs to be solved, while satisfying some given constraints. This problem has applications in structural engineering (theory of elastic bodies), control theory and other contact or impact problems. For example, think of an elastic membrane which, under the application of some force, will adopt a certain shape. If then there is an obstacle close enough to the membrane, and it is in the direction of the applied force, the membrane will deform because of the obstacle. This is a typical example of an elliptic problem, which is independent of time.

Uniqueness of solutions to the obstacle problem was already shown in 1980 by Kinderlehrer and Stampacchia [8]. Shortly after that, Nachbin wrote 'Obstacle Problems in Mathematical Physics' [11], where many practical examples and applications of the obstacle problem are discussed. There and in [6], the elliptic obstacle problem is discussed in detail, and we will investigate the problem in similar way.

The parabolic obstacle problem is mainly used in two ways: either to describe melting and crystallization processes, or it is applied to financial problems (see for example [4],[12]).

First of all, in this paper we want to answer the question: "What is the equilibrium position and contact area of an elastic membrane whose boundary is held fixed, and which is constrained to lie above a given obstacle?". To answer this question, we must solve

*Email: a.j.jonkheer@student.utwente.nl

the elliptic obstacle problem. We will explain how to find exact solutions in 1D and 2D, and we will use the finite element method (FEM) and quadratic programming to solve the problem numerically. Our second objective is to solve the parabolic obstacle problem in one dimension. To achieve this, we will also use the finite element method. Furthermore, we want to provide availability of code to further study the elliptic and parabolic obstacle problem.

1.1 Problem sketch

Investigating the obstacle problem, we are looking for the solution to a partial differential equation. This we will describe as the function $u : \Omega \to \mathbb{R}$, where $\Omega \subset \mathbb{R}^n$ is the domain on which the partial differential equation is defined. Now the value of u is constrained by some obstacle $\psi : \Omega \to \mathbb{R}$, hence the name of this problem. As stated above, different partial differential equations will have different meanings in real life, and thus different kinds of obstacle.

We make the distinction between two sets on Ω. Let the coincidence set, where the membrane touches the obstacle, be defined as

$$I := \{\mathbf{x} \in \Omega : u(\mathbf{x}) = \psi(\mathbf{x})\}.$$

Furthermore, there is the non-coincidence set, which we define as

$$\Lambda := \{\mathbf{x} \in \Omega : u(\mathbf{x}) > \psi(\mathbf{x})\}.$$

2 Modeling

In this section, we will demonstrate the methods used to solve both the elliptic and the parabolic obstacle problem.

2.1 Elliptic equations

We will start with elliptic equations. These equations are a good starting point for numerically solving the obstacle problem, because they only have spatial variables. This makes it relatively easy to check whether we have implemented the finite element method correctly.

We consider a membrane as a thin plate offering no resistance to bending, only having tension. Modeling an elliptic obstacle problem usually comes with forces acting on the membrane. We will describe these forces by some function $f : \Omega \to \mathbb{R}$. We fix the membrane at the boundary $\partial\Omega$ of the domain, or:

$$u = g \text{ on } \partial\Omega$$

for some function $g = g(\mathbf{x})$. It can be shown ([6],[11]) that the potential energy $E(u)$ of such a membrane is approximately equal to

$$E(u) = \int_\Omega \Big(\frac{\lambda}{2}|\nabla u|^2 - fu\Big) d\Omega, \tag{2.1}$$

with the assumption that $|\nabla u|$ is small on Ω. Minimization of this potential energy is found [6] when u satisfies the following equation:

$$\nabla \cdot (\lambda \nabla u) + f = 0$$

For simplicity we will use $\lambda = 1$. Therefore, we will look for a solution to $-\Delta u = f$, also known as Poisson's equation. Here Δ is the Laplace operator ($= \nabla^2$).
Now we can describe the elliptic obstacle problem as the following system of equations.

$$\begin{cases} -\Delta u = f \text{ on } \Lambda \\ u \geq \psi \text{ on } \Omega \\ u = \psi \text{ on } I \end{cases} \tag{2.2}$$

2.1.1 Exact solutions

1D exact

Let us look at how to find exact solutions for the 1D elliptic obstacle problem. Let $\Omega = [-1, 1]$ and let $\psi(x) \in C^1(\Omega)$ be a concave obstacle. If the forces acting on the membrane are large enough (we will get back to this later on), $I \neq \varnothing$. For now, we will focus on problems where I consists of one interval, such that the domain Ω is split up into three sections. Let us call the left and right touching point x_L and x_R, respectively, where $x_L \leq x_R$. Using the system of equations 2.2, we get:

$$\begin{cases} -\frac{\partial^2 u}{\partial x^2} = f \text{ on } [-1, x_L) \cup (x_R, 1] \\ u \geq \psi \text{ on } [-1, 1] \\ u = \psi \text{ on } [x_L, x_R] \end{cases} \tag{2.3}$$

Now we have given that Poisson's equation has to hold on the interval $[-1, x_L)$ and on the interval $(x_R, 1]$. Furthermore, as long as $\psi(x) \in C^1(\Omega)$ and $f(x) \in C^1(\Omega)$, we know that $u \in C^1(\Omega)$ [6]. We can now set up two systems of equations to find the solution.

$$\begin{cases} u_L(-1) = 0 \\ u_L(x_L) = \psi(x_L) \\ \frac{\partial u_L}{\partial x}(x_L) = \frac{\partial \psi}{\partial x}(x_L) \end{cases} \qquad \begin{cases} u_R(1) = 0 \\ u_R(x_R) = \psi(x_R) \\ \frac{\partial u_R}{\partial x}(x_R) = \frac{\partial \psi}{\partial x}(x_R) \end{cases}$$

From Poisson's equation we can find the general solutions of u_L and u_R, which resembles the real solution up to some linear function with two unknowns. This gives us six equations with six unknowns, which gives us one unique solution to the obstacle problem 1D.

2D exact

For the 2D situation, things get a little more complicated. Let $\Omega = [-1, 1]^2$ and let $\psi \in C^1(\Omega)$ be a concave obstacle. Consequently, 2.2 becomes

$$\begin{cases} -\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} = f \text{ on } \Lambda \\ u \geq \psi \text{ on } [-1, 1]^2 \\ u = \psi \text{ on } I \end{cases} \tag{2.4}$$

If we try to solve the obstacle problem in the same way we did before, we notice we cannot integrate twice to get a general solution, because we are dealing with partial derivatives. In this case, the general solution would be an infinite sum, which would not allow us to solve this system in the same manner. There are two approaches with which we can solve this problem.

1. $\frac{\partial^2 u}{\partial y^2} = 0$

Because of symmetry, it does not matter whether we look at $\frac{\partial^2 u}{\partial x^2} = 0$ or at $\frac{\partial^2 u}{\partial y^2} = 0$. Therefore, let us investigate the latter. Consequently, the only solution is of the form $u(x, y) = \zeta(x)(c_1 y + c_2)$, where $\zeta(x)$ is the general solution to Poisson's equation in 1D. As a consequence of this, as soon as we define the points on the boundary at $y = \pm 1$, we fix the whole membrane. Therefore, this is only an interesting problem if we do not define $g(x, \pm 1)$. If $g(-1, y)$ and $g(1, y)$ are constant on $\partial\Omega$, c_1 has to be equal to 0. Hence we arrived back at the 1D problem. Alternatively, if $g(-1, y)$ and $g(1, y)$ are linear in y, we can get exact solutions for a slightly more difficult problem. However, since these problems match the one-dimensional problem for the most part, we will not focus on this.

2. $\frac{\partial^2 u}{\partial \theta^2} = 0$

We decided to look only at obstacle problems radially symmetric around $(0, 0)$, where both the obstacle and the boundary is symmetric. Also, the function f can not depend on θ, since then the solution would not be symmetric. Now it makes more sense to use the Poisson's equation in polar form, because $\frac{\partial u}{\partial \theta} = 0$ and therefore $\frac{\partial^2 u}{\partial \theta^2} = 0$ for a symmetric membrane. This gives:

$$-\frac{\partial^2 u}{\partial r^2} - \frac{1}{r}\frac{\partial u}{\partial r} = f(r) \text{ on } \Lambda, \tag{2.5}$$

where we are now looking for a solution $u(r)$. The easiest way to get a completely symmetric membrane, is to fix $u(1, \theta) = 0$ and let the membrane be suspended above an obstacle symmetric around $(0, 0)$.

We can now rewrite equation 2.5 to the following equation.

$$-\frac{\partial u}{\partial r}\Big(r\frac{\partial u}{\partial r}\Big) = rf(r) \text{ on } \Lambda \tag{2.6}$$

For any function $f(r) = \sum_{n=-\infty}^{\infty} c_n x^n$, where c_n is a scalar, this equation gives an analytic general solution. The easiest force distribution would be $f(r) = -1/r$, but since constant forces are more practical, we will focus on that.
Let $f(r) := -\gamma$. Integrating twice gives us the general solution for $u(r)$.

$$u(r) = \frac{\gamma}{4}r^2 + c_1 \ln r + c_2 \text{ on } \Lambda \tag{2.7}$$

When $I \neq \varnothing$, it means there is a value r^* at which the membrane touches the obstacle all around. Again, because $\psi(x) \in C^1(\Omega)$ and $f(x) \in C^1(\Omega)$, we also know $u \in C^1(\Omega)$. We can thus find the unknown constants with the use of the following system of equations.

$$\begin{cases} u(1) = 0 \\ u(r^*) = \psi(r^*) \\ \frac{\partial u}{\partial r}(r^*) = \frac{\partial \psi}{\partial r}(r^*) \end{cases} \tag{2.8}$$

We now have three equations with three unknowns, which we can solve for certain cases of ψ.

2.1.2 The Finite Element Method

The finite element method has proven itself to be a very helpful tool to solve the obstacle problem. There are two papers in particular which have been very helpful for our research ([1],[9]). We will cover the basis of it here.

First of all, we find the weak formulation of Poisson's equation. For arbitrary functions $r(\mathbf{x}) : \Omega \to \mathbb{R}$, we get

$$\int_\Omega (\Delta u + f)\, r \, d\Omega = 0 \quad \forall r \tag{2.9}$$

This can be rewritten, using that $\nabla \cdot \big(\nabla u\, r\big) = \Delta u\, r + \nabla u \cdot \nabla r$:

$$\int_\Omega (\nabla u \cdot \nabla r) d\Omega - \int_\Omega \nabla \cdot \big(\nabla u\, r\big) d\Omega = \int_\Omega f r d\Omega \quad \forall r \tag{2.10}$$

Using Gauss' theorem, we can rewrite the second integral:

$$\int_\Omega \nabla \cdot \big(\nabla u\, r\big) d\Omega = \oint_{\partial\Omega} \Big(\frac{\partial u}{\partial n}\, r\Big) d(\partial\Omega), \tag{2.11}$$

where n is the normal direction on $\partial\Omega$. We will only look at membranes with fixed boundaries. Then r has to be equal to 0 on the boundaries, so this integral will completely disappear.
Now we will approximate the solution of equation 2.10 by discretizing Ω into a finite amount of elements. We will call this solution $u_h(\mathbf{x})$, where h is the maximal h of all elements. Let us define N as the number of nodes in the discretization of Ω.

$$u_h(\mathbf{x}) = \sum_{i=1}^{N} u_i \phi_i(\mathbf{x}), \tag{2.12}$$

where u_i is the value of the solution at node i, and $\phi_i(\mathbf{x})$ is the basis function for node i. We will use linear hat functions, which are defined as

$$\phi_i(\mathbf{x_j}) = \delta_{i,j}, \quad \delta_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \tag{2.13}$$

Define S as the set of free, unfixed nodes in Ω. For the elliptic problems we will investigate, with fixed boundaries, this is the same set as the nodes which are not on $\partial\Omega$. Moreover, we will use the projection of r into the function space spanned by the basis functions on S,

$$r(\mathbf{x}) = \sum_{\{j | \mathbf{x}_j \in S\}} c_j \phi_j(\mathbf{x}) \quad \text{for arbitrary constants } c_j. \tag{2.14}$$

This transforms equation 2.10 to the following equation.

$$\sum_{i=1}^{N} u_i \int_\Omega \nabla\phi_i \cdot \nabla\phi_j \, d\Omega = \int_\Omega f\phi_j \, d\Omega \quad \text{for } j \in \{k \,|\, \mathbf{x_k} \in S\} \tag{2.15}$$

Finally, we can write the equation above as a system of $|S|$ equations.

$$\sum_{\{i \,|\, \mathbf{x_i} \in S\}} u_i \int_\Omega \nabla\phi_i \cdot \nabla\phi_j \, d\Omega = \int_\Omega f\phi_j \, d\Omega \; - \sum_{\{l \,|\, \mathbf{x_l} \notin S\}} u_l \int_\Omega \nabla\phi_l \cdot \nabla\phi_j \, d\Omega \quad \text{for } j \in \{k \,|\, \mathbf{x_k} \in S\} \tag{2.16}$$

Error analysis

There have been extensive studies on the convergence rates of the finite element approximation of elliptic partial differential equations. The objective is to say something about the behaviour of the error between the real solution and the finite element solution, $u - u_h$. We will not prove the error estimates, but there are many studies the reader can refer to (for example [2],[9]). To estimate the error, two norms are normally used. We use |...| to denote the Euclidean norm.

- $||u||^2_{L^2(\Omega)} = \int_\Omega u^2 d\Omega$
- $||u||^2_{H^1(\Omega)} = ||\nabla u||^2_{L^2} = \int_\Omega |\nabla u|^2 d\Omega$

For the elliptic problem without an obstacle, we have the following results for the error between the exact solution u and the approximation u_h [2]. These results hold, provided that the problem is H^2-regular, which means that the solution to the problem contains two generalized derivatives in L^2. This assumption is satisfied for convex domains [3].

- $||u - u_h||_{L^2(\Omega)} \leq C h^2_{\max} ||u||_{H^2(\Omega)}$
- $||u - u_h||_{H^1(\Omega)} \leq C h_{\max} ||u||_{H^2(\Omega)}$

Here C is a constant independent of h, $h_{\max}$ is the maximum value of h on the discretized grid, and $||u||_{H^2(\Omega)}$ denotes the H^2-norm of the exact solution u, which naturally does not depend on h. This means we can expect second order convergence for the L^2-norm and first order convergence for the H^1-norm. Generally, for smooth obstacles and boundary values, these results will also hold for elliptic obstacle problems.

1D

In one dimension, there are only two boundary nodes, x_1 and x_N. Let h be the distance between two adjacent nodes. The basis functions we are using are

$$\phi_i(x) = \begin{cases} (x - x_{i-1})/h & \text{if } x \in [x_{i-1}, x_i] \\ (x_{i+1} - x)/h & \text{if } x \in [x_i, x_{i+1}] \\ 0 & \text{otherwise} \end{cases} \tag{2.17}$$

It can be easily seen this satisfies equation 2.13. At the boundaries, these hat functions are halved. We can observe right away that hat functions ϕ_i and ϕ_j only have common support when $|i - j| \leq 1$.
Now, equation 2.16 reduces to

$$\sum_{i=2}^{N-1} u_i \int_\Omega \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x}\, dx = \int_\Omega f \phi_j\, dx \; - \; u_1 \int_\Omega \frac{\partial \phi_1}{\partial x} \frac{\partial \phi_j}{\partial x}\, dx - u_N \int_\Omega \frac{\partial \phi_N}{\partial x} \frac{\partial \phi_j}{\partial x}\, dx$$
$$\text{for } j = 2, \ldots, N-1.$$

All that is left to do now is finding the value of the integrals. This is quite straightforward. For fixed values $i, j \in \{2, \ldots, N-1\}$:

$$\int_\Omega \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x}\, dx = \int_{x_{i-1}}^{x_{i+1}} \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x}\, dx = \begin{cases} -1/h & \text{if } j = i - 1 \\ 2/h & \text{if } j = i \\ -1/h & \text{if } j = i + 1 \end{cases} \tag{2.18}$$

Secondly, the force integral, we can approximate using the values of f at the nodes.

$$\int_\Omega f\phi_j\,dx = \int_{x_{j-1}}^{x_{j+1}} f\phi_j\,dx \approx f(x_j)(h+h)/2 = f(x_j)h \quad \text{for } j \in \{2,\dots,N-1\} \tag{2.19}$$

In conclusion, we write down the complete system of equations in matrix form:

$$\frac{1}{h}\begin{bmatrix} 2 & -1 & 0 & & \dots & & 0 \\ -1 & 2 & -1 & & & & \\ 0 & -1 & 2 & -1 & \ddots & & \vdots \\ & & \ddots & \ddots & \ddots & & \\ \vdots & & \ddots & \ddots & 2 & -1 & 0 \\ & & & & -1 & 2 & -1 \\ 0 & & \dots & & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} u_2 \\ \\ \vdots \\ \\ u_{N-1} \end{bmatrix} = \begin{bmatrix} f(x_2)h \\ \\ \vdots \\ \\ f(x_{N-1})h \end{bmatrix}$$

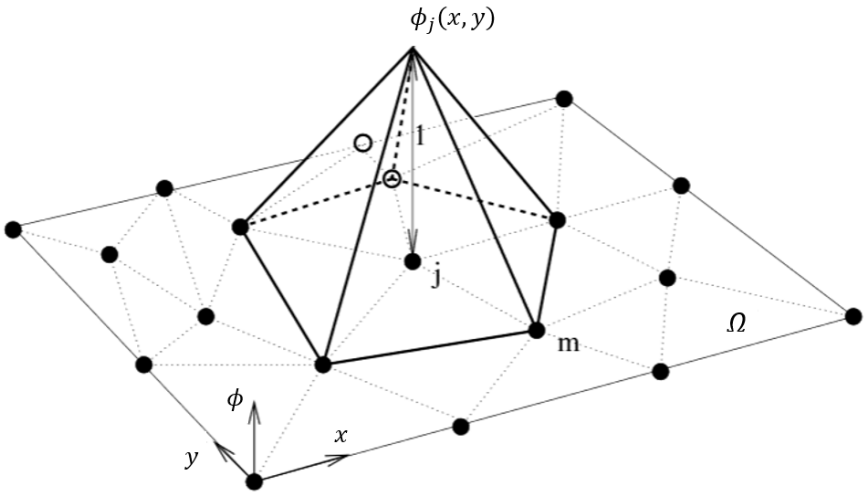

FIGURE 1: Two-dimensional piece-wise linear basis function for node j.

2D

We will again rewrite equation 2.16 into a system of equations. In two dimensions, calculating the integrals is a bit more complicated. This time we have different basis functions, which will spread over multiple triangles, as can be seen in Figure 1. This means we have to split up the first integral of equation 2.16 into a sum over the different triangles that the basis function covers. Let T_k be the domain of triangle k and let $M_{i,j}$ be the set of triangle indices where ϕ_i and ϕ_j have common support. The reader should remember that S is the set of free nodes in Ω. Let us define the value $A_{i,j}$ as follows.

$$A_{i,j} = \int_\Omega \nabla\phi_i \cdot \nabla\phi_j\,d\Omega = \sum_{k\in M_{i,j}} \iint_{T_k} \left(\frac{\partial\phi_i}{\partial x}\frac{\partial\phi_j}{\partial x} + \frac{\partial\phi_i}{\partial y}\frac{\partial\phi_j}{\partial y} \right) dxdy \quad \text{for } i,j \in \{p|(x_p,y_p) \in S\}$$

Per triangle, there are three basis functions, one associated to each node of the triangle. Again, these basis functions should satisfy equation 2.13. We consider piece-wise linear

functions. Therefore, on a single triangle, the basis function is a linear function of the form

$$\phi_l(x,y) = \alpha_l x + \beta_l y + \gamma_l \tag{2.20}$$

We can easily find the coefficients using equation 2.13.
To make sure we do not have to integrate over each triangle multiple times, we consider the local matrix $\tilde{A}$ which is associated with a single triangle, and then sum up all these matrices to find $A_{i,j}$. The matrix elements $\tilde{A}_{lm}$ are now given by

$$\begin{aligned}\tilde{A}_{lm} = \tilde{A}_{ml} &= \iint_{T_k} \left(\frac{\partial \phi_l}{\partial x}\frac{\partial \phi_m}{\partial x} + \frac{\partial \phi_l}{\partial y}\frac{\partial \phi_m}{\partial y} \right) dxdy \\ &= (\alpha_l \alpha_m + \beta_l \beta_m) \iint_{T_k} dxdy \\ &= (\alpha_l \alpha_m + \beta_l \beta_m)\text{Area}(T_k)\end{aligned}$$

The actual values $A_{i,j}$ we can now be found by summing all local matrices. Note that we need to keep track of which nodes are associated to which triangle, and thus to which local matrix.

For the right-hand side vector $\mathbf{b}$, we will first evaluate the force integral.

$$\int_\Omega f\phi_j d\Omega = \sum_{k \in M_{j,j}} \iint_{T_k} f\phi_j \, dxdy \approx \sum_{k \in M_{j,j}} f(\bar{x}_k, \bar{y}_k)\frac{1}{3}\text{Area}(T_k) \quad \text{for } j \in \{p | (x_p, y_p) \in S\},$$

where $(\bar{x}_k, \bar{y}_k)$ denotes the coordinates of the midpoint of triangle T_k, which we can easily find by dividing the sum of the coordinates of the nodes of the triangle by three.
Furthermore, the rest of the right-hand side can easily be found. We can use the values of matrix A which we have calculated above. Finally, we have arrived at the system of equations which we will solve.

$$\sum_{\{i \,|\, (x_i,y_i) \in S\}} u_i A_{i,j} = \frac{1}{3} \sum_{k \in M_{j,j}} f(\bar{x}_k, \bar{y}_k)\text{Area}(T_k) \; - \sum_{\{l \,|\, (x_l,y_l) \notin S\}} u_l A_{l,j} \quad \text{for } j \in \{p \,|\, (x_p, y_p) \in S\}$$

2.2 Parabolic equations

For the parabolic obstacle problem, we were not able to find exact solutions. Moreover, we were not able to find exact solutions in previous research. Therefore, we will focus only on the finite element method. However, convergence will be shown for the parabolic problem without an obstacle.
We will focus on the one-dimensional problem. Elaboration on the two-dimensional FEM can be found in for example [1] or [10].

We are looking for the function $u(x,t)$ as a solution to the heat equation with thermal diffusivity equal to 1.

$$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 0 \tag{2.21}$$

2.2.1 The Finite Element Method

Similar to the elliptic problem, we first write this into the weak formulation. For arbitrary functions $r(x) : \Omega \to \mathbb{R}$, we get

$$\int_\Omega \left(\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} \right) r d\Omega = 0 \quad \forall r. \tag{2.22}$$

As before, we rewrite the spatial second derivative, followed by applying Gauss' theorem, which gives

$$\int_\Omega \left(\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} \right) r = \left[\frac{\partial u}{\partial n} r \right]_{x_1}^{x_N}. \tag{2.23}$$

This time, we define $u_h(x,t) = \sum_{i=1}^N u_i(t)\phi_i(x)$, where $u_i(t)$ denotes the value of u at node i at time t. Let r again be spanned by the basis functions ϕ_j.
As a result, we have obtained the following system of equations.

$$\sum_{i=1}^N \frac{\partial u_i}{\partial t} \int_\Omega \phi_i \phi_j \, d\Omega + \sum_{i=1}^N u_i \int_\Omega \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x} d\Omega = \left[\sum_{j=1}^N \frac{\partial \phi_i}{\partial n} \phi_j \right]_{x_1}^{x_N} \tag{2.24}$$

In matrix form this can be described as follows.

$$M \frac{\partial \mathbf{u}}{\partial t} + A\mathbf{u} = \mathbf{b} \tag{2.25}$$

We again use the piece-wise linear basis functions as defined in 2.17. This makes it simple to find the values of the first integral. Note again that ϕ_i and ϕ_j only overlap whenever $|i-j| \leq 1$. That said, when there is a Neumann boundary condition, the values at the boundary nodes are not fixed, and we use only half of the basis function. That gives us the following result.

$$M_{i,j} = \int_\Omega \phi_i \phi_j \, d\Omega = \begin{cases} \frac{2}{3}h & \text{if } i = j \text{ on interior nodes} \\ \frac{1}{3}h & \text{if } i = j \text{ on boundary nodes} \\ \text{if } \frac{1}{6}h & |i-j| = 1 \end{cases} \tag{2.26}$$

The entries of matrix A we already calculated before (2.18), with the addition of the values at the boundary nodes in the case of Neumann boundary conditions, which gives

$$A_{i,j} = \int_\Omega \frac{\partial \phi_i}{\partial x} \frac{\partial \phi_j}{\partial x} \, dx = \begin{cases} 2/h & \text{if } j = i \text{ on interior nodes} \\ 1/h & \text{if } j = i \text{ on boundary nodes} \\ -1/h & \text{if } |i-j| = 1 \end{cases} \tag{2.27}$$

We have now only discretized in space, and we still have a continuous time derivative in our system of equations. We will use the backward (or implicit) Euler method to approximate $\frac{\partial \mathbf{u}}{\partial t}$. We use the backward Euler method, because this gives unconditional stability of the finite element solution [10].

$$\frac{\partial \mathbf{u}}{\partial t} \approx \frac{\mathbf{u}^t - \mathbf{u}^{t-1}}{\Delta t}, \tag{2.28}$$

where $\mathbf{u}^t$ is the vector with the values of u at time t. This gives

$$M \frac{\mathbf{u}^t - \mathbf{u}^{t-1}}{\Delta t} + A\mathbf{u}^t = \mathbf{b} \tag{2.29}$$

$$\Rightarrow \quad (M + A\Delta t)\mathbf{u}^t = M\mathbf{u}^{t-1} + \mathbf{b}\Delta t \tag{2.30}$$

Error analysis

Since now we are working with a time variable and a spatial variable, we cannot use the L^2-norm or the H^1-norm to define convergence of the finite element solution towards the actual solution. Therefore, a typical error measure is introduced, which we will call the L^2H^1-norm.

$$||u||^2_{L^2H^1(\Omega)} = \int_0^T \int_\Omega \left|\frac{\partial u}{\partial x}\right|^2 dxdt, \tag{2.31}$$

where T is the time until which we solve the equation, and $|\dots|$ denotes the Euclidean 2-norm. Now, we compute the squared L^2H^1-norm of the error $e := u - u_h$.

$$||e||^2_{L^2H^1(\Omega)} = \int_0^T \int_\Omega \left|\frac{\partial e}{\partial x}\right|^2 dxdt = \sum_{j=1}^N \int_{t_{j-1}}^{t_j} \underbrace{\left[\int_\Omega \left|\frac{\partial e}{\partial x}(t)\right|^2\right] dx}_{:=g(t)}\, dt$$

At each time instant t_j, we can evaluate $g(t)$, using the discretized solution u_h and the gradient of the exact solution u, for example by using trapezoidal integration approximation. Furthermore, we can also use the trapezoidal rule to integrate over the time interval $[t_{j-1}, t_j]$:

$$\int_0^T g(t)dt = \sum_{j=1}^N \int_{t_{j-1}}^{t_j} g(t)dt \approx \sum_{j=1}^N \frac{1}{2}(t_j - t_{j-1})(g(t_j) + g(t_{j-1}))$$

Because of the use of backward Euler integration in the finite element method, we expect first order convergence in time [10]. Secondly, because of the first order finite difference approximation we used for the integration over Ω, we expect first order convergence in space as well. Furthermore, we assume again the solution to our problem is H^2-regular. This gives us the following error estimate.

$$||u - u_h||_{L^2H^1(\Omega)} \leq C(\Delta t + h) \tag{2.32}$$

2.3 Quadratic programming

In conclusion, we need a tool to solve the obstacle problem using the finite element method. Until now, we have only found a system of equations that solves the partial differential equation. We need to write this as a linear program with some constraints. We can easily do this:

$$\begin{aligned} &\min_{\mathbf{u}} \ \frac{1}{2}\mathbf{u}'A\mathbf{u} + \mathbf{b}'\mathbf{u} \\ &\text{subject to:} \quad u_i \geq \psi(x_i) \quad \forall i \end{aligned}$$

The minimization is equivalent to solving $A\mathbf{u} + \mathbf{b} = 0$. This is exactly the shape of equation we have found in rewriting the different PDEs to systems of equations using the finite element method. In this paper, we will not go into detail how the quadratic programming is done. We will use the MATLAB function called `quadprog`. The variables used in MATLAB are stated below.

```
min 0.5*x'*H*x + f'*x   subject to:  A*x <= b
 x
```

For the obstacle problem, we will therefore use `A = -speye(dof)`, where `dof` is the number of free nodes (degrees of freedom) in the discretization. Furthermore, we use `b = -psi`, where `psi` are the values of the obstacle function ψ at the free nodes. The matrix `H` and the vector `f` depend on the partial differential equation that we solve. These can be found in the sections above.

3 Results

In this section, we will discuss all of our findings regarding the topics discussed in Section 2. First, we will focus on the exact solutions we found for the one-dimensional problem. Secondly, we will focus on the exact solutions in two dimensions and lastly, the results of the finite element method are shown regarding the elliptic obstacle problem.

3.1 Exact solutions to the elliptic obstacle problem: 1D

3.1.1 ψ linear

Let us first look at the situation where $\psi(x) = \alpha x + \beta$ and $f(x) = -\gamma$ is constant, where $\gamma > 0$. From $-\frac{\partial^2 u}{\partial x^2} = f$ we obtain as a general solution $u(x) = \frac{1}{2}\gamma x^2 + c_1 x + c_2$ by integrating twice. In Figure 2 we can see all different possibilities for a solution to the problem.

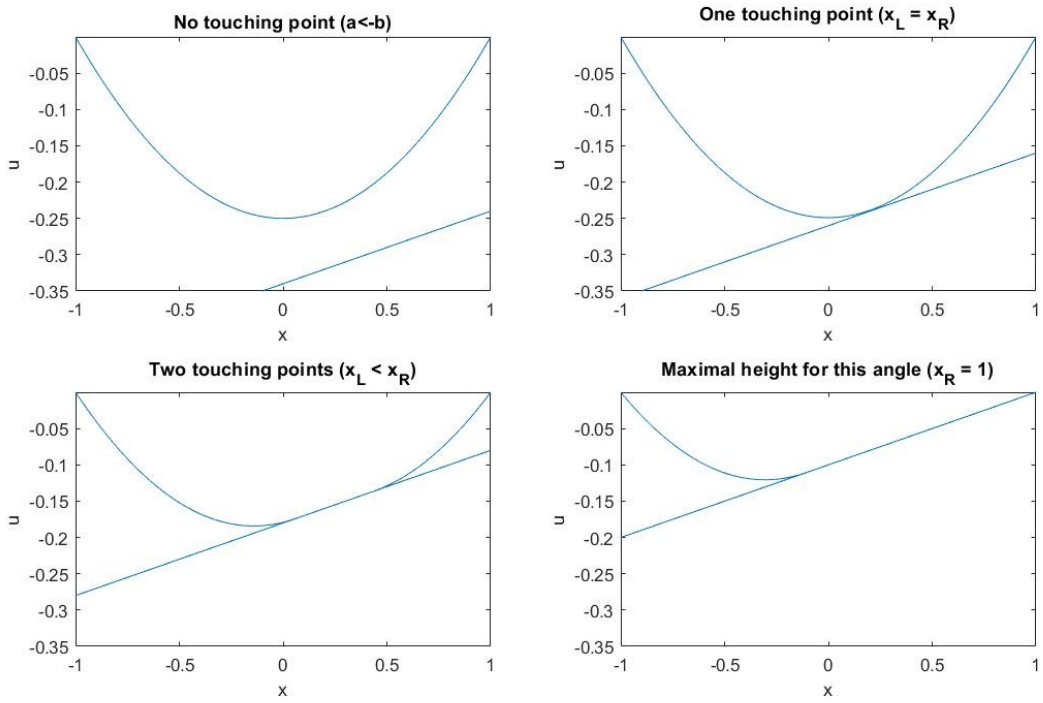

FIGURE 2: All possible situations for fixed α and varying β.

First, let us solve the system of equations on $[-1, x_L]$. $u_L(-1) = 0$ gives $c_2 = c_1 - 0.5\gamma$. $u_L'(x_L) = \alpha$ gives $c_1 = \alpha - \gamma x_L$. Now we obtain the following expression for x_L.

$$\begin{aligned} u_L(x_L) &= \frac{1}{2}\gamma x_L^2 + (\alpha - \gamma x_L)x_L + (\alpha - \gamma x_L) - \frac{1}{2}\gamma = \alpha x_L + \beta \\ &\Rightarrow -\frac{1}{2}\gamma x_L^2 - \frac{1}{2}\gamma + \alpha - \gamma x_L = \beta \\ &\Rightarrow x_L^2 + 2x_L + 1 + 2\frac{\beta - \alpha}{\gamma} = 0 \\ &\Rightarrow x_L = -1 + \sqrt{2\frac{\alpha - \beta}{\gamma}} \qquad \text{if } \beta - \alpha < 0 \end{aligned}$$

The limitation on α and β make sense, because this problem is not feasible when the height of the obstacle at the boundary ($\psi(-1) = \beta - \alpha$) is greater than 0. c_1 and c_2 follow from the equations above. Similarly, we can solve the system of equations on $[x_R, 1]$. This gives the following results:

$$u_R(x) = \frac{1}{2}\gamma x^2 + (\alpha - \gamma \cdot x_R)x - 0.5\gamma - (\alpha - \gamma \cdot x_R) \tag{3.1}$$

$$x_R = 1 - \sqrt{-2\frac{\alpha + \beta}{\gamma}} \qquad \text{if } \alpha + \beta < 0 \tag{3.2}$$

Again, the limitation on α and β is identical to saying that the height of the obstacle at the boundary ($\psi(1) = \alpha + \beta$) is below 0, which it should be for the problem to be feasible.

Now it is possible to find the force at which the membrane has one touching point, that is, where $x_L = x_R$. Solving for γ gives us an interesting result, where $\gamma_{\min}$ is the minimal force from above needed to touch the obstacle.

$$\gamma_{\min} = \sqrt{\beta^2 - \alpha^2} - \beta = -\beta\left(1 + \sqrt{1 - \frac{\alpha^2}{\beta^2}}\right) \tag{3.3}$$

Consequently, we observe that $f(x) = 2\beta$ gives a single touching point for a horizontal obstacle. Furthermore, in the limit cases ($\alpha = \pm b$), we get a single touching point for $f(x) = \beta$.

3.1.2 Two concave obstacles

In an article from 1980 [5], research is done on the obstacle problem with one or two peaks with $f = 0$. We can apply the same technique as before to find exact solutions to the obstacle problem with peaks. Suppose there are multiple peaks which are continuously differentiable in the interval $[-1, 1]$. The solution would look like Figure 3.

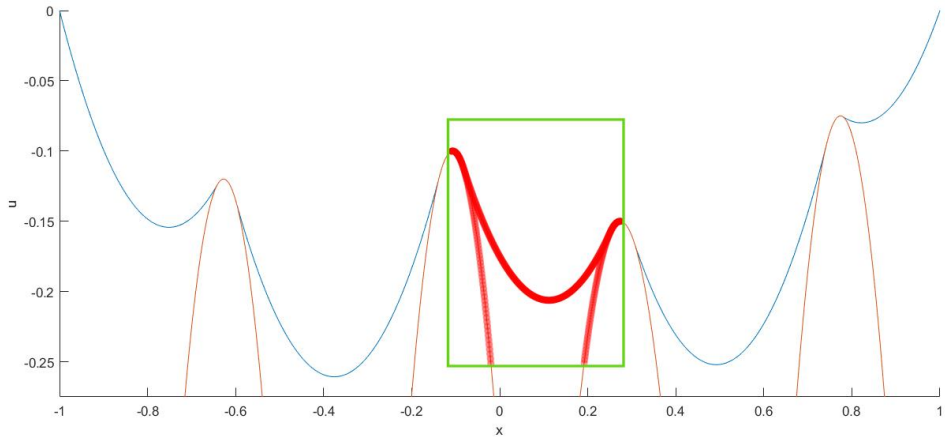

FIGURE 3: Obstacle problem with multiple peaks.

We can find a solution for in between the peaks by adding one more equation to the system of equations. We need one more equation, because now, in each section where the membrane is not in contact with the obstacle, there are four unknowns. Let us focus on one of those sections. Zooming in on the green region of Figure 3, we get Figure 4. We call the obstacle on the left and right ψ_L and ψ_R, respectively. We don't know in advance where the membrane will stop touching either sides; let us call these points x_L and x_R. Furthermore, we have two unknown integration constants from Poisson's equation, which again holds in the section in between. The needed system of equations follows naturally.

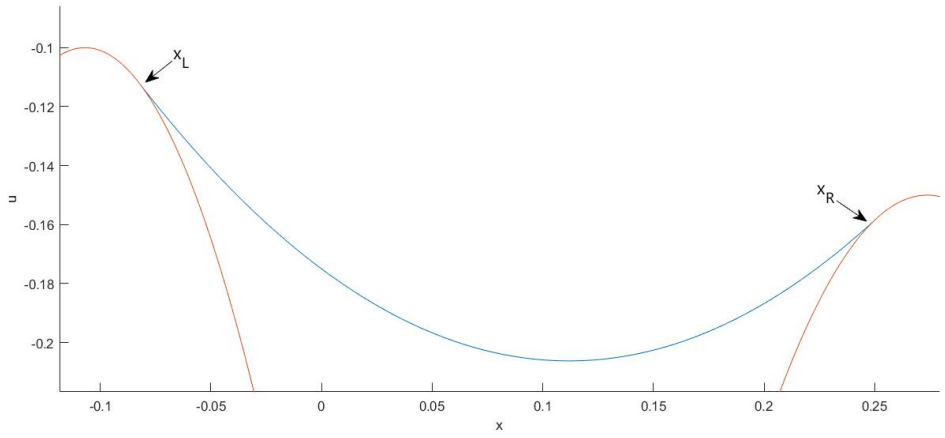

FIGURE 4: Section in between two peaks.

$$\begin{cases} u(x_L) = \psi_L(x_L) \\ u(x_R) = \psi_R(x_R) \\ \frac{\partial u}{\partial x}(x_L) = \frac{\partial \psi_L}{\partial x}(x_L) \\ \frac{\partial u}{\partial x}(x_R) = \frac{\partial \psi_R}{\partial x}(x_R) \end{cases} \tag{3.4}$$

In conclusion, we can find the coincidence sets on all the different peaks and thus finding a unique, continuous solution u on Ω.

3.2 Exact solutions to the elliptic obstacle problem: 2D

As explained in Section 2.1.1, we can find the exact solution on Λ for radially symmetric problems from equations 2.7 and 2.8. But let us first look at the minimum force needed to touch the obstacle.
Because we are solving for a concave symmetric obstacle, only the value of $\psi(r)$ at 0 matters. If there is no obstacle, the general solution is straightforward to find. From equation 2.7, we see that $c_1 = 0$, because $u \in C^1(\Omega)$. Furthermore, from $u(1) = 0$, we find $c_2 = -\gamma/4$. This gives

$$u(r) = \frac{\gamma}{4}r^2 - \frac{\gamma}{4}. \tag{3.5}$$

Let $\psi(0) = \delta$, $\delta \in \mathbb{R}$. If $\delta > 0$, then naturally the membrane will always touch the obstacle, no matter the value of γ. On the other hand, if $\delta < 0$, the minimum force needed to touch the obstacle is now found by $u(0) = \psi(0)$, which gives the following results.

$$\begin{cases} \gamma_{\min} = 4\delta \text{ for } \delta \leq 0 \\ \gamma_{\min} = 0 \text{ for } \delta > 0 \end{cases} \tag{3.6}$$

3.2.1 $\psi(r)$ is constant

Let us define $\psi(r) := -\alpha$, where $\alpha \in \mathbb{R}^+$. First, using $\frac{\partial u}{\partial r}(r^*) = \frac{\partial \psi}{\partial r}(r^*)$, we find

$$c_1 = -\frac{\gamma}{2}(r^*)^2. \tag{3.7}$$

Moreover, we again use the fact that $u(1) = 0$ to find $c_2 = -\gamma/4$. Lastly, using $u(r^*) = \psi(r^*)$, we can find r^*.

$$\begin{aligned} & \frac{\gamma}{4}(r^*)^2 - \frac{\gamma}{2}(r^*)^2 \ln r^* - \frac{\gamma}{4} = -\alpha \\ \Rightarrow\ & (r^*)^2(1 - 2\ln r^*) = 1 - 4\frac{\alpha}{\gamma} \\ r^* := e^p \Rightarrow\ & e^{2p}(1-2p) = 1 - 4\frac{\alpha}{\gamma} \\ \Rightarrow\ & (2p-1)e^{2p-1} = \frac{4\frac{\alpha}{\gamma} - 1}{e} \\ \Rightarrow\ & p = \frac{1}{2}\left[1 + W_{-1}\left(\frac{1}{e}\left(4\frac{\alpha}{\gamma} - 1\right)\right)\right] \end{aligned}$$

Here W_{-1} stands for the lower branch of the Lambert W function. We need the lower branch, because r^* is less than 1 only for $p < 0$. It is interesting to observe that the coincidence set only depends on the fraction of the height of the obstacle and the force on the membrane.
As a result, we have found an exact solution to the obstacle problem for a constant force and horizontal obstacle:

$$\begin{cases} u(r) = \frac{\gamma}{4}r^2 - \frac{\gamma}{2}(r^*)^2 \ln r - \frac{\gamma}{4} \text{ for } r \geq r^* \\ u(r) = -\alpha \text{ for } r \leq r^* \end{cases} \tag{3.8}$$

3.2.2 Non-constant $\psi(r)$

A more challenging and variable obstacle is found in the obstacle defined as $\psi(r) = -\alpha r^2 + \beta$, with $\alpha \in \mathbb{R}^+$ to create a concave obstacle. Then, using $\frac{\partial u}{\partial r}(r^*) = \frac{\partial \psi}{\partial r}(r^*)$, we find

$$c_1 = -(\frac{\gamma}{2} + 2\alpha)(r^*)^2 \tag{3.9}$$

And finally, using $u(r^*) = \psi(r^*)$, we can find r^*.

$$\begin{aligned} &\frac{\gamma}{4}(r^*)^2 - (\frac{\gamma}{2} + 2\alpha)(r^*)^2 \ln r^* - \frac{\gamma}{4} = -\alpha(r^*)^2 + \beta \\ \Rightarrow\ &(r^*)^2(1 - 2\ln r^*) = \frac{\gamma + 4\beta}{\gamma + 4\alpha} \end{aligned}$$

As before, we take $r^* = e^p$, where $r^* < 1$ and therefore $p < 0$. Then we obtain

$$p = \frac{1}{2}\left[1 + W_{-1}\left(-\frac{1}{e}\frac{\gamma + 4\beta}{\gamma + 4\alpha}\right)\right]. \tag{3.10}$$

We have thus found the exact solution to this problem.

$$u(r) = \begin{cases} \frac{\gamma}{4}r^2 - (\frac{\gamma}{2} + 2\alpha)(r^*)^2 \ln r - \frac{\gamma}{4} \text{ for } r \geq r^* \\ -\alpha r^2 + \beta \text{ for } r \leq r^* \end{cases} \tag{3.11}$$

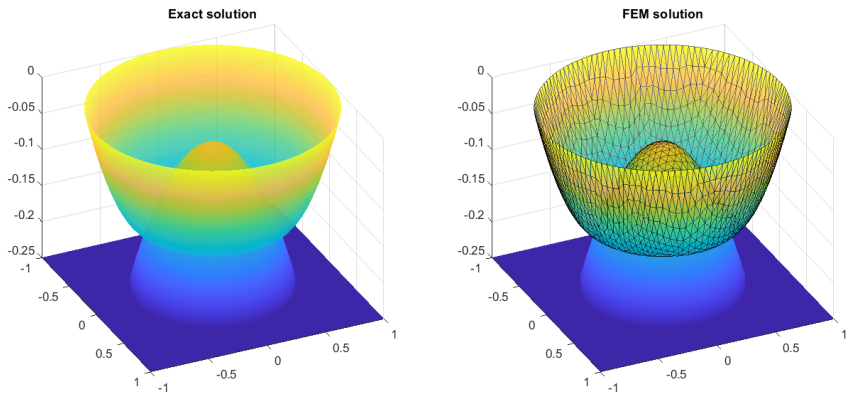

FIGURE 5: The exact solution and FEM solution for a 2D symmetric obstacle, $\alpha = 0.5$, $\beta = -0.05$, $\gamma = 2$.

3.3 The Finite Element Method

In this section, we will show some results found using the finite element method. One thing we should note up front: for the function `quadprog`, we have set the property 'OptimalityTolerance' to 10^{-15} using `optimoptions`. This is because we found that for a higher value, the error norms did not decrease as expected for small values of $h_{\max}$, but when we set 'OptimalityTolerance' to this value, it did exactly as it should, also for very small values of h.

3.3.1 Exact solutions of the elliptic obstacle problem

Since we have found exact solutions, we can check convergence of the solution found by the finite element method (in combination with `quadprog`) towards the exact solution. Let us look at Poisson's equation in 2D. For example, let us take a symmetric parabolic obstacle with $\psi(r) = 0.5r^2 - 0.05$ and a force acting on the membrane $f = -2$. The exact solution can be found using equations 3.10 and 3.11. The results are shown in Figures 5 and 6.

As expected, we see that the L^2-norm of the error is $O(h_{\max}^2)$; for each factor 10 that $h_{\max}$ decreases, the L^2-norm decreases by a factor 10^2. Furthermore, we see that the H^1-norm of the error is $O(h_{\max})$; for each factor 10 that $h_{\max}$ decreases, the H^1-norm decreases by a factor 10 as well. Figures A.1 and A.2 in the appendix show the results of the 2D case where $\frac{\partial u}{\partial y} = 0$, which is similar to the 1D equation. We observe the same convergence rates, which means that this method of combining the finite element method with the function `quadprog` works really well, and the solution converges nicely towards the exact solution. This gives us confidence that this method also works with non-symmetric obstacle problems, of which we do not have exact solutions.

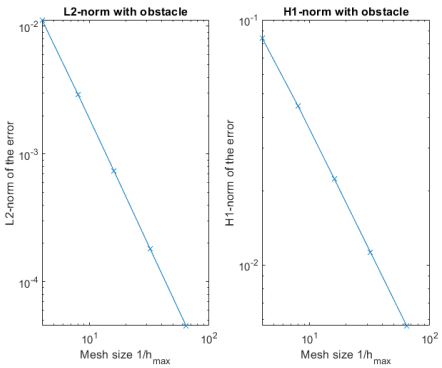

FIGURE 6: L^2-norm and H^1-norm for different values of $h_{\max}$, showing convergence of the problem as shown in Figure 5.

3.3.2 Other solutions of the elliptic obstacle problem

We now have a method for solving the elliptic problem for any given boundary values $g(x, y)$ on $d\Omega$, any given obstacle $\psi(x, y)$, and any force $f(x, y)$ working on the membrane. This gives a lot of freedom on what to investigate. We will just show one solution which can be a start for further research. Figure 7 shows the solution to the elliptic obstacle problem with a linear obstacle $\psi(x, y) = \alpha x + \beta$ ($\alpha = 0.3, \beta = -0.42$), with force $f = -2$. The black dots are the nodes where the membrane touches the obstacle.

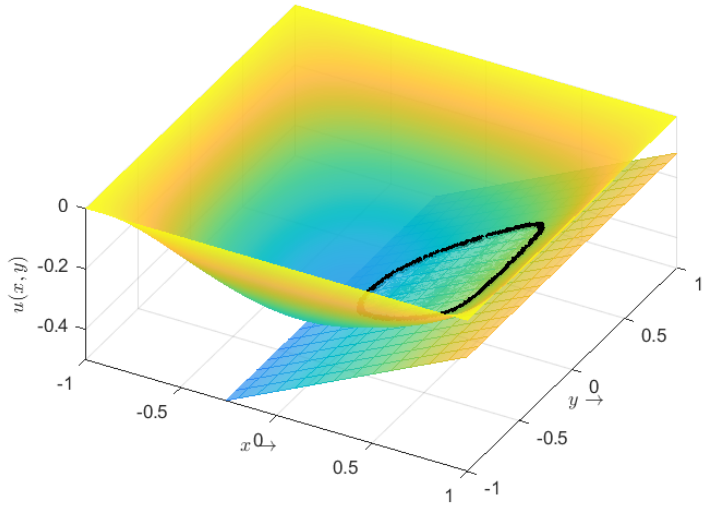

FIGURE 7: Elliptic obstacle problem with $\psi(x, y) = 0.3x - 0.42$ and force $f = -2$.

For example, we can investigate what happens when one of two things happen:

- The obstacle can move upwards and downwards.
- The force can become stronger and weaker.

In both cases, the shape and size of the touching region will change. The obstacles used to create Figure 8 have the same slope $\alpha = 0.3$, but are shifting between $\beta = -0.54$ (inner curve) and $\beta = -0.3$ (outer curve). The forces used to create Figure 9 vary between $f = -1.5$ (inner curve) and $f = -3.5$ (outer curve).

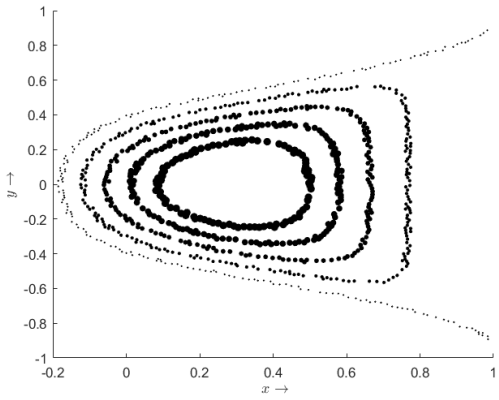

FIGURE 8: Touching region when the obstacle of Figure 7 moves upwards and downwards.

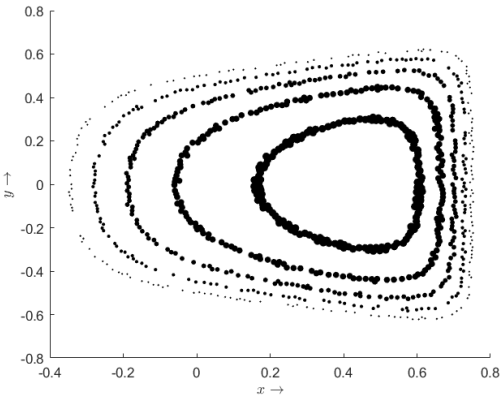

FIGURE 9: Touching region when the force becomes stronger and weaker in the situation described in Figure 7.

3.3.3 Exact solution of the heat equation

Subsequently, we will solve the heat equation with the finite element method.We use the initial distribution $u(x, 0) = \sin(\pi x)$ over the domain $\Omega = [0, 1]$ and time period [0,0.5], with Dirichlet boundary conditions $x(0, t) = x(1, t) = 0$. We can easily find the exact solution (see for example [7]), which is

$$u(x, t) = \sin(\pi x)e^{-\pi^2 t}$$

The finite element solution is shown in Figure 10. Furthermore, we can see the convergence of the error in Figure 11. We obtain first order convergence, as we had expected. For every factor 10 that h decreases, the L^2H^1-norm also decreases by a factor 10. It is good to see that it works the way it should, because then we can move on to the parabolic obstacle problem.

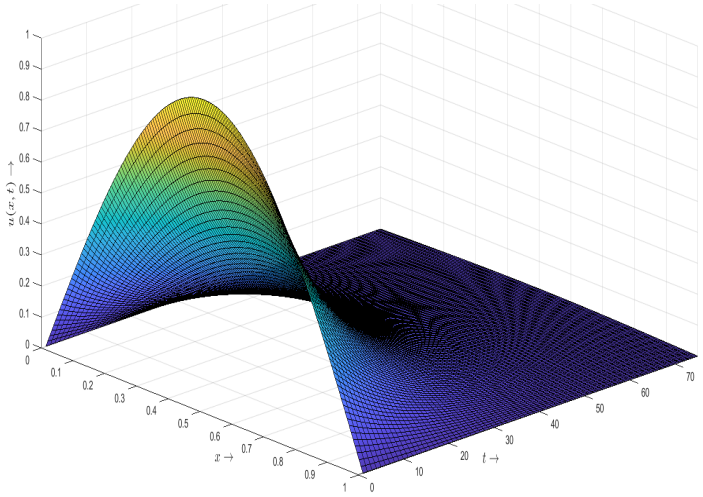

FIGURE 10: The finite element solution of the heat equation with initial state $u(x,0) = \sin(\pi x)$.

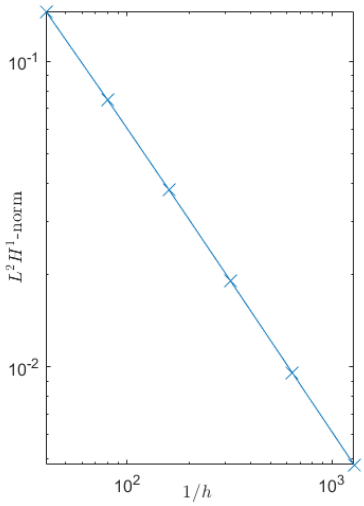

FIGURE 11: L^2H^1-norm for different values of h, convergence rate of the problem as shown in Figure 10.

3.3.4 The parabolic obstacle problem

Last but not least, we will consider the parabolic obstacle problem. Therefore, we will solve the heat equation, but now, with the help of `quadprog`, we constrain the solution u_h to lie above a parabolic obstacle $\psi(x) = -10(x-0.5)^2 + 0.5$. The result is shown in Figure 12.

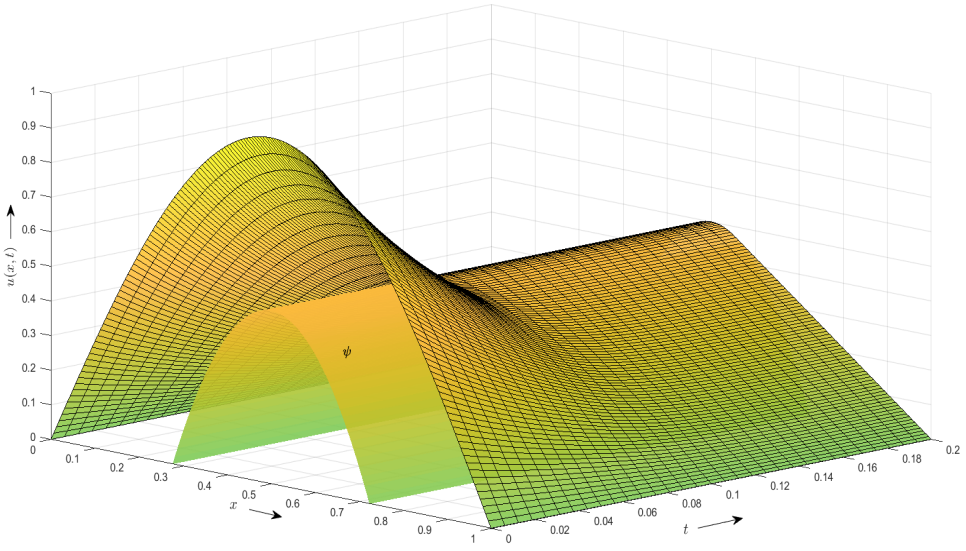

FIGURE 12: The finite element solution of the heat equation with initial state $u(x,0) = \sin(\pi x)$, constrained by the obstacle ψ.

We can see how the distribution nicely settles over the obstacle, going towards an equilibrium. We observed that the equilibrium position of the solution converges to a C^1 solution on Ω, namely the solution we get when we solve the 1D elliptic obstacle problem with $f = 0$, for which the exact solution can be found using the methods described before. Altogether, we have come full circle, which is a nice way to end this section.

4 Discussion

In this paper, we have covered many different subjects. Therefore, we will also split up this discussion into multiple parts.

4.1 Exact solutions

We have acquired a method to solve elliptic obstacle problems in a relatively simple way, provided that the membrane has Dirichlet boundary conditions, and the obstacle and the force acting on the membrane are part of a very specific set of functions. For simple cases, or very much simplified cases, this method could achieve results really quickly and give exact results, where one does not need to worry about stability of solutions, or how quickly a numerical solution converges. That said, it will probably be too restricted for practical uses, because we simply do not work with ideal environments. A practical example could be a simplified model of a circular trampoline, if one needs to calculate the minimal depth of the hole in the ground, when a given weight is allowed on. Moreover, it is not very complicated to make a one- or two-dimensional finite element method script that solves the actual problem very accurately. On the other hand, in large-scale problems, the need for accurate and efficient numerical methods is always there, in particular considering limited computational power and storage space.

In conclusion, maybe this method will inspire others to look for exact solutions to obstacle problems, because we do think there is still some ground to be won. One could for instance investigate exact solutions when the force is a space-dependent function, other than a constant function. However, the question is whether it is meaningful to investigate this further, other than a fun mathematical puzzle.

4.2 Finite element method

Regarding the finite element method, we have not achieved any new insights, other than another paper which explains how it works. We have exhibited that combined with the MATLAB function `quadprog`, it is a powerful tool to solve the obstacle problem. In addition, there is plenty to be gained. In general, the adaptive finite element method could be implemented, to make it possible to make even more accurate approximations. Because we used elements of equal size, we had relatively large errors for a big amount of elements. The rest is related to the specific form of partial differential equation.

4.2.1 Elliptic equations

First of all, we believe it is possible to find an exact formula for the minimal force needed for the membrane to touch a horizontal obstacle, when the membrane boundary is fixed

at the value 0 on the square domain $[-1, 1]^2$. Because it is not easy (maybe impossible) to find an exact solution, we can try finding a relationship between the height of the obstacle and the minimal force needed to create a coincidence set, using MATLAB. Plotting these values for different heights, we can most probably find an (empirical) relationship, for example through the MATLAB function `fittype`. This, in turn, could be an incentive to find the exact formula and maybe give new insights on how to solve problems like this in an analytic way.

Secondly, we could implement the use of Neumann and Robin boundary conditions, which would allow a much broader range of problems to be solved. However, this is done in many other papers, so for further research, this should not be a problem.

4.2.2 Parabolic equations

Regarding the parabolic obstacle problem, we have three notes. Firstly, one could investigate what happens with a moving obstacle and if it is possible to create a self-repeating pattern over time. This maybe has some practical applications, which can be explored using the code provided in the appendix.

Secondly, the time integration scheme can be made more accurate by using the method of Crank-Nicholson, Heun, Simpson, or Runge-Kutta, which give more and more accurate integration schemes. This means we could get faster convergence, which, in turn, implies we get closer to the exact solution with the same values of Δt.

Finally, at the end of Section 3.3.4, we observed that the parabolic obstacle problem converges towards the C^1 solution of an elliptic obstacle problem with the same obstacle. We expect that most of the combinations of obstacles and initial states will converge similarly, but we did not have time to investigate this. So this could be another topic of study.

4.2.3 Hyperbolic equations

Although we have not covered the subject of hyperbolic equations, we do think it is worth noting that this form of equations can be solved in a similar way. These problems have many practical applications which the other two do not have, and therefore is very interesting to study as well. Dynamical contact problems are however much more complicated in terms of theory and practice than the problems covered in this thesis.

4.3 Conclusion

All in all, we have acquired a well-working algorithm to answer our research question, but there is still a lot of room for improvement in this research regarding the obstacle problem. However, during this research, we have learned a lot, and maybe something in this research paper will inspire others to discover something new in this field.

References

[1] J. Alberty, C. Carstensen, and S. A. Funken. Remarks around 50 lines of matlab: short finite element implementation. *Numerical Algorithms*, 20(2-3):117–137, 1999.

[2] S. Bartels. *Numerical Methods for Nonlinear Partial Differential Equations*, volume 47. Springer International Publishing, 01 2015.

[3] Dietrich Braess. *Finite Elements: Theory, Fast Solvers, and Applications in Solid Mechanics*. Cambridge University Press, 3 edition, 2007.

[4] N. Calvo, J. I. Díaz, J. Durany, E. Schiavi, and C. Vázquez. On a doubly nonlinear parabolic obstacle problem modelling ice sheet dynamics. *SIAM Journal on Applied Mathematics*, 63(2):683–707, 2003.

[5] S. Chow and J. Mallet-Paret. The parameterized obstacle problem. *Nonlinear Analysis: Theory, Methods & Applications*, 4(1):73–91, 1980.

[6] C. Eck, H. Garcke, and P. Knabner. *Free Boundary Problems*, pages 427–487. Springer International Publishing, Cham, 2017.

[7] M. Hancock. The 1-D heat equation. MIT Course 18.303 Linear Partial Differential Equations, 2006.

[8] D. Kinderlehrer and G. Stampacchia. *An Introduction to Variational Inequalities and their Applications*, volume 31. Siam, 1980.

[9] M. G. Larson and F. Bengzon. *The Finite Element Method: Theory, Implementation, and Applications*, volume 10. Springer Science & Business Media, 2013.

[10] WJ Minkowycz, Ephraim M Sparrow, GE Schneider, and RH Pletcher. Handbook of numerical heat transfer. *New York, Wiley-Interscience, 1988, 1035 p. No individual items are abstracted in this volume.*, 1988.

[11] L. Nachbin. Chapter 1 the obstacle problem. In *Obstacle Problems in Mathematical Physics*, volume 134 of *North-Holland Mathematics Studies*, pages 1–21. North-Holland, 1987.

[12] A. Petrosyan and H. Shahgholian. Parabolic obstacle problems applied to finance. *Recent developments in nonlinear partial differential equations*, 439:117–133, 2007.

A Figures

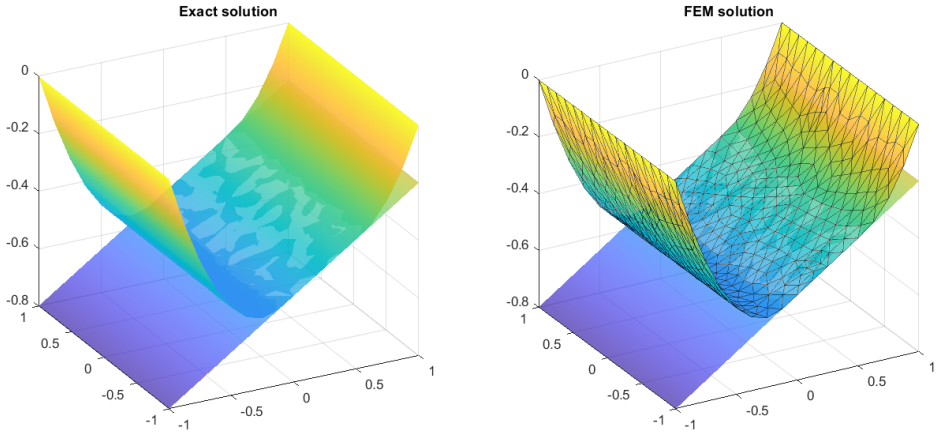

FIGURE A.1: Exact solution and FEM solution of the 2D elliptic obstacle problem with $\frac{\partial u}{\partial y} = 0$, constant f, linear ψ.

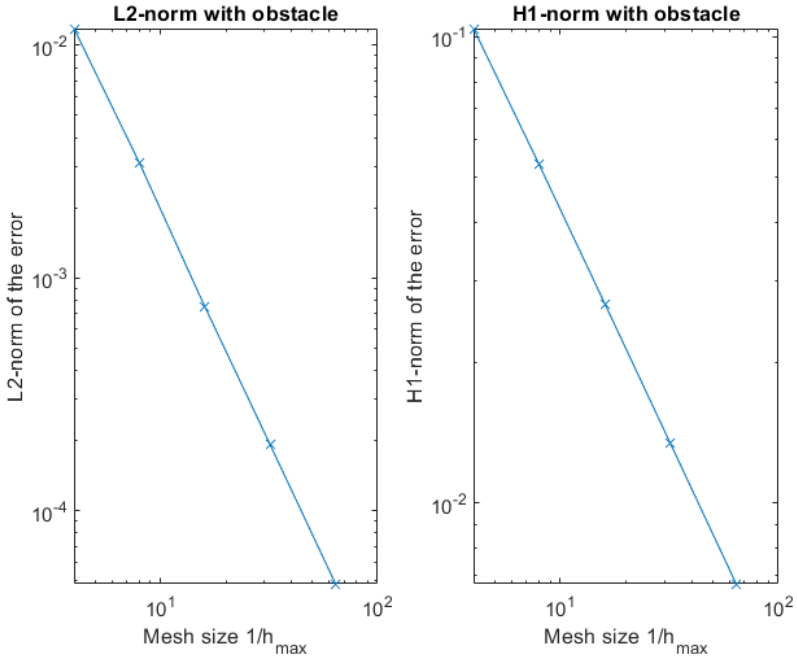

FIGURE A.2: L^2-norm and H^1-norm for different values of h_{max}, convergence rates of the problem as shown in figure A.1.

B MATLAB code

B.1 Elliptic obstacle problem

B.1.1 1D

Main code

```
options = optimset('Display', 'off');  %Needed settings for faster code, no message displayed
f = @(x) -1;

stepsizes = 1+0.5*[1];
alpha = 0.1;
beta = -0.1;
for k = 1:length(stepsizes)
    h = 1/1001;
    coord = -1:h:1;
    u_h = zeros(length(coord),1);
    nnodes = length(coord);
    dof = [2:nnodes-1];
    A = StiffMat1D(length(dof),h);
    b = LoadVec1D(coord,f);
%     u_h(dof) = A(dof,dof)\b(dof);    %solution without obstacle
    psi = @(x) alpha*x+ beta;
    u_h(dof) = quadprog(A(dof,dof),-b(dof),-speye(length(dof)),-psi,[],[],[],[],[],options);

    % plot solution with obstacle
    hold on
    plot(coord,u_h)
    line([-1 1],[obst-a obst+a])

    fmin(iter) = 2*(a+2*-0.1-2*sqrt(-0.1*(a-0.1)));
    fmin(iter) = obst-sqrt(obst^2-a^2);
end
```

StiffMat1D.m

```
function A = StiffMat1D(N,h)
A = zeros(N,N);
for i = 1:N
    if i == 1
        A(1,1:2)=[2 -1];
    elseif i == N
        A(N,N-1:N)=[-1 2];
    else
        A(i,i-1:i+1) = [-1 2 -1];
    end
end
A = A/h;
```

LoadVec1D.m

```
function b = LoadVec1D(x,f)
n = length(x)-1;
```

```
b = zeros(n+1,1);
for i = 1:n
h = x(i+1) - x(i);
b(i) = b(i) + f(x(i))*h/2;
b(i+1) = b(i+1) + f(x(i+1))*h/2;
end
```

B.1.2 2D

Main code

```
options = optimoptions('quadprog','OptimalityTolerance',1e-15);

%% Different exact solutions
index = 3; %see exactsol.m for specifications of indices
[f, uex, graduex, psi, nonhom] = exactsol(index);

%% Finite element method
[p,e,t] = initmesh('circleg','hmax',1/4); %define triangular grid and triangle measure
ref = 5;                                  %number of refinements
hmax = zeros(ref,1); h1norm = zeros(ref,1); l2norm = zeros(ref,1); %initialization

for i=1:ref
    if i>1
        [p,e,t] = refinemesh('circleg',p,e,t);  %refinement of the grid
    end
    hmax(i) = 1/2/2^i;                          % each refinement, 'hmax' reduces by factor 2
    elements = t(1:3,:)';
    coord = p';
    nnodes = length(coord);
    nt = size(t,2);

    x=zeros(nt,3); y=zeros(nt,3); %init. x and y
    for j=1:nt
        x(j,:) = coord(elements(j,:),1)';
        y(j,:) = coord(elements(j,:),2)';
    end
    areas = 0.5*(x(:,2).*y(:,3)+x(:,3).*y(:,1)+x(:,1).*y(:,2)...
        -x(:,2).*y(:,1)-x(:,3).*y(:,2)-x(:,1).*y(:,3)); %areas of all triangles

    u_h = zeros(nnodes,1);
    boundnodes = unique(e(1:2,:));              %find the nodes which are on the boundary
    Freenodes = setdiff(1:nnodes,boundnodes);
    if nonhom
        u_h(boundnodes) = uex(coord(boundnodes,:)); %homogenous or inhomogenous boundaries
    end
    A = StiffMat2D(coord, elements, areas);
    b = LoadVec2D(coord, elements, f, areas);
    b = b - A*u_h;

%     u_h(Freenodes) = A(Freenodes,Freenodes)\b(Freenodes); %solution without obstacle
    u_h(Freenodes) = quadprog(A(Freenodes,Freenodes),-b(Freenodes),...
        -speye(length(Freenodes)),-psi(coord(Freenodes,:)),[],[],[],[],[],options);
    l2norm(i,1) = L2norm(u_h,uex,coord,elements,areas);
    h1norm(i,1) = H1norm(u_h,graduex,coord,elements,areas);
```

```
    show(elements,coord,u_h,uex)     %show FEM solution and exact solution side by side
end

%% Display norms
figure
subplot(1,2,1)
loglog(1./hmax,l2norm,'-x')
title('L2-norm with obstacle'), xlabel('Mesh size 1/h_{max}'), ylabel('L2-norm of the error')
subplot(1,2,2)
loglog(1./hmax,h1norm,'-x')
title('H1-norm with obstacle'), xlabel('Mesh size 1/h_{max}'), ylabel('H1-norm of the error')
```

L^2-norm

```
function l2norm = L2norm(u,uex,coord,elem,areas)
l2norm2 = 0;
for i = 1:size(elem,1)
    nodes = elem(i,:); % local-to-global map
    x = coord(nodes,1); % node x-coordinates
    y = coord(nodes,2); % node y-coordinates
    localcoord = coord(nodes,:);
    mid = 1/3*sum(localcoord,1); % mid-point of triangles
    umid = 1/3*sum(u(nodes));
    l2norm2 = l2norm2 + areas(i)*(norm(uex(mid)-umid))^2;
end
l2norm = sqrt(l2norm2);
```

H^1-norm

```
function h1norm = H1norm(u,graduex,coord,elem,areas)
h1norm2 = 0;
for i = 1:size(elem,1)
    nodes = elem(i,:); % local-to-global map
    x = coord(nodes,1); % node x-coordinates
    y = coord(nodes,2); % node y-coordinates
    localcoord = coord(nodes,:);
    mid = 1/3*sum(localcoord,1); % mid-point of triangles
    [b,c] = Gradients(x,y,areas(i));
    graduh = [0;0];
    for j=1:3
        graduh = graduh + u(nodes(j)).*[b(j);c(j)];
    end
    h1norm2 = h1norm2 + areas(i)*(norm(graduex(mid)-graduh))^2;
end
h1norm = sqrt(h1norm2);
```

StiffMat2D.m

```
function A = StiffMat2D(p,t,areas)
np = size(p,1);
nt = size(t,1);

Alocal = zeros(3,3,nt);
I =zeros(9,nt); %row indices
J =zeros(9,nt); % column indices

for K = 1:nt
nodes = t(K,:); % local-to-global map
x = p(nodes,1); % node x-coordinates
y = p(nodes,2); % node y-coordinates
[b,c] = Gradients(x,y,areas(K));

Alocal(:,:,K) = (b*b'+c*c')*areas(K);
I(:,K) = [nodes,nodes,nodes]';
J(:,K) = reshape(repmat(nodes,3,1),[],1);
end

A = sparse(I(:),J(:),Alocal(:));
```

LoadVec2D.m

```
function b = LoadVec2D(coord,elem,f,areas)
nt = size(elem,1);
b = zeros(size(coord,1),1);
for i=1:nt
    localnodes = elem(i,:);
    localcoord = coord(localnodes,:);
    mid = 1/3*sum(localcoord,1);
    b(localnodes)= b(localnodes) + f(mid)/3*areas(i);
end
```

show.m

```
function show(elements,coord,u,uex)
figure('Name','Example graph and FEM graph','Units','inches','Position',[3,3,12,5]);
subplot(1,2,1)
hold on; grid on
trisurf(elements,[coord(:,1)],[coord(:,2)],...
    uex(coord)','facecolor','interp','edgecolor','none','edgealpha',0,'facealpha',0.8)
title('Exact solution')
view(65,35)

subplot(1,2,2)
hold on; grid on
trimesh(elements,[coord(:,1)],[coord(:,2)],...
    u','facecolor','interp','edgecolor','k','edgealpha',0.5,'facealpha',0.8)
title('FEM solution')
view(65,35)
```

gradients.m

```
function [b,c] = gradients(x,y,area)
b=[y(2)-y(3); y(3)-y(1); y(1)-y(2)]/2/area;
c=[x(3)-x(2); x(1)-x(3); x(2)-x(1)]/2/area;
```

exactsol.m

```
function [f,uex,graduex,psi,nonhom] = exactsol(index)
nonhom = false;

if index == 1
    % Simplest checks without obstacle
    % (u = 0 on x^2+y^2=1)
    f = @(x) -1;
    uex = @(x) 1/4*(x(:,1).^2+x(:,2).^2-1);
    graduex = @(x) [x(:,1)/2;x(:,2)/2];
    psi = @(x) -10*ones(size(x,1),1);

elseif index == 2
    % (u = 0 on boundary [-1,1]^2)
    f = @(x) -(2*((1-x(:,1).^2)+(1-x(:,2).^2)));
    uex=@(x) -(1+x(:,1)).*(1-x(:,1)) .* (1+x(:,2)).*(1-x(:,2));
    graduex=@(x) [2*x(:,1).*(1-x(:,2).^2); 2*x(:,2).*(1-x(:,1).^2)];
    psi = @(x) -10*ones(size(x,1),1);

elseif index == 3
    % constant or parabolic obstacle
    nonhom = true;
    gamma = 2;            %force value
    f = @(x) -gamma;
    alpha = 0.5; beta = -0.05;
    constant = false;         %specify whether it is a constant obstacle

    if constant
        psi = @(x) -alpha*ones(size(x,1),1);
        gradpsi = @(x) [0;0];
        epwr = 0.5*(1+lambertw(-1,1/exp(1)*(4*alpha/gamma-1)));
        rstar = exp(epwr);
        c_1 = -rstar^2*gamma/2;
    else                          % parabolic obstacle
        psi = @(x) -alpha*(x(:,1).^2+x(:,2).^2)+beta;
        gradpsi = @(x) -2*alpha*[x(:,1);x(:,2)];
        epwr = 0.5*(1+lambertw(-1,-1/exp(1)*(gamma+4*beta)/(gamma+4*alpha)));
        rstar = exp(epwr);
        c_1 = -rstar^2*(gamma/2+2*alpha);
    end

    uex = @(x) (sqrt(x(:,1).^2+x(:,2).^2)>rstar).*(gamma/4.*(x(:,1).^2+x(:,2).^2) ...
        +c_1/2*log(x(:,1).^2+x(:,2).^2)-gamma./4) + ...
        (sqrt(x(:,1).^2+x(:,2).^2)<=rstar).*psi(x);
    graduex = @(x) (sqrt(x(:,1).^2+x(:,2).^2)>rstar).*(gamma/4 ...
        +c_1/2/(x(:,1).^2+x(:,2).^2)).*[2*x(:,1);2*x(:,2)] + ...
        (sqrt(x(:,1).^2+x(:,2).^2)<=rstar).*gradpsi(x);
```

```
elseif index == 4
    % Solution with linear obstacle with du/dy=0 (extraction from 1D problem)
    nonhom = true;
    gamma = 2;            %force value
    f = @(x) -gamma;
    alpha=0.3; beta=-0.5;
    left_touch = -1+sqrt(-2*(alpha-beta)/f(1));
    right_touch = 1-sqrt(2*(alpha+beta)/f(1));
    c1_1=alpha+f(1)*left_touch;
    c1_2=alpha+f(1)*right_touch;
    psi = @(x) alpha*x(:,1) + beta;

    uex = @(x) (x(:,1)<=left_touch).*(-0.5*f(1)*x(:,1).^2+c1_1*x(:,1)+...
        0.5*f(1)+c1_1)+(x(:,1)>left_touch).*(x(:,1)<right_touch).*psi(x)+...
        (x(:,1)>=right_touch).*(-0.5*f(1)*x(:,1).^2+c1_2*x(:,1)+0.5*f(1)-c1_2);
    graduex = @(x) (x(:,1)<=left_touch).*[-f(1)*x(:,1)+c1_1;0]+...
        (x(:,1)>left_touch).*(x(:,1)<right_touch).*[alpha;0]+...
        (x(:,1)>=right_touch).*[-f(1)*x(:,1)+c1_2;0];
end
```

B.2 Parabolic obstacle problem

Main code

```
stepsizes = 0.1./2.^[4];     % Define different values of h to loop over

for k = 1:length(stepsizes)
    f = @(x) 0;
    h = stepsizes(k);
    x = 0:h:1;               % Define Omega
    dt = h;                  % Use dt=h to get same descent of dt as h
    T = 0.3;
    iter = round(T/dt);      % Solve until time T

    uex = @(x,t) sin(pi*x)*exp(-pi^2*t*dt);
    graduex = @(x,t) pi*cos(pi*x)*exp(-pi^2*t*dt);

    nnodes = length(x);
    u = zeros(iter,nnodes);
    u(:,1)=0; u(:,end)=0;

    freenodes = 2:nnodes-1;
    dof = length(freenodes);
    b = LoadVec1D(x(freenodes),f);
    b(1) = 0;                % =alpha/h for u(0)=alpha
    b(end) = 0;              % =beta/h for u(1)=beta
    BC = [0 0];              % [0] means Dirichlet, [1] means Neumann BC

    u0 = uex(x(freenodes),0)';
    u(1,freenodes) = u0;

    A = StiffMat1Ddjdk(dof,h,BC);
    M = StiffMat1Djk(dof,h,BC);

    %Obstacle
```

```
    options = optimset('Display', 'off');  %No help message displayed
    psi = @(x) -10*(x-0.5).^2+0.5;
    for i=1:iter+1
        psii(i,1:nnodes) = psi(x);          %Define psi for all time for plot
    end

    for i=1:iter
%         u(i+1,freenodes) = (M+A*dt)\(M*u0+b*dt);            %Solution without obstacle
        u(i+1,freenodes) = quadprog(M+A*dt,-(M*u0+b*dt),-speye(dof),...
        -psi(x(freenodes)),[],[],[],[],[],options);    %Solution with obstacle
        u0 = u(i+1,freenodes)';
    end

    l2h1norm(k) = sqrt(spacetimenorm(u,h,dt,iter,graduex));

    for i=1:iter+1
        plot(x,u(i,:))
        xlim([0 1])             %plot solution over time
        ylim([0 1])
        pause(0.01)
    end

    figure()
    hold on, grid on
    surf(x,0:dt:dt*iter,psii,'edgecolor','none','facealpha',0.9) %plot obstacle
    surf(x,0:dt:dt*iter,u,'facecolor','interp','facealpha',0.9) %plot complete solution
    zlabel('u(x,t)')
    xlabel('x')
    ylabel('t')
    zlim([0 1])
    ylim([0 iter*dt])
    view(45,20)
end

%% Error convergence
figure()
loglog(1./stepsizes,l2h1norm,'x-')
title('Step size versus spacetimenorm')
xlabel('1/h')
ylabel('l2h1norm')
```

StiffMat1Ddjdk.m

```
function A = StiffMat1Ddjdk(N,h,BC)
A = zeros(N,N);
for i = 1:N
    if i == 1
        if BC(1) == 0               %Dirichlet
            A(1,1:2)=[2 -1];
        else                        %Neumann
            A(1,1:2)=[1 -1];
        end
    elseif i == N
        if BC(2) == 0               %Dirichlet
            A(N,N-1:N)=[-1 2];
```

```
        else                        %Neumann
            A(N,N-1:N)=[-1 1];
        end
    else
        A(i,i-1:i+1) = [-1 2 -1];
    end
end
A = A/h;
```

StiffMat1Djk.m

```
function M = StiffMat1Djk(N,h,BC)
M = zeros(N,N);
for i = 1:N
    if i == 1
        if BC(1) == 0               %Dirichlet
            M(1,1:2)=[4 1];
        else                      %Neumann
            M(1,1:2)=[2 1];
        end
    elseif i == N
        if BC(2) == 0               %Dirichlet
            M(N,N-1:N)=[1 4];
        else                      %Neumann
            M(N,N-1:N)=[1 2];
        end
    else
        M(i,i-1:i+1) = [1 4 1];
    end
end
M = M*h/6;
```

spacetimenorm.m

```
function R = spacetimenorm(u,h,dt,iter,graduex)
errorintegral = zeros(iter,1);
R=0;
errorintegral(1) = 0;
for t = 1:iter
    for i = 1:size(u,2)-1
        graduh(i) = (-u(t,i)+u(t,i+1))/(h);
    end
     guex = graduex([0:h:1],t)';
     e_right = guex(2:end) - graduh';                    %Trapezoidal rule in space
     e_left = guex(1:end-1)- graduh';
     errorintegral(t+1) = h/2*sum(e_right.^2+e_left.^2);
end

for j = 1:iter
    R = R + 1/2*dt*(errorintegral(j+1)+errorintegral(j));
end
```