Slovak University of Technology in Bratislava
Faculty of Civil Engineering

Reg. No.: SvF-16646-104364

Basic hydrological modeling in software NaturaSat

Master's thesis

2024 **Bc. Ivana Piačková**

Slovak University of Technology in Bratislava
Faculty of Civil Engineering

Basic hydrological modeling in software NaturaSat

Master's thesis

Study programme:	Mathematical and Computational Modeling
Study field:	9.1.9. Applied Mathematics
Training workplace:	Department of Mathematics and Descriptive Geometry
Supervisor:	Ing. Michal Kollár, PhD.
Consultant:	Prof. RNDr. Karol Mikula, DrSc.

Bratislava 2024 **Bc. Ivana Piačková**

Slovak University of Technology in Bratislava
Department of Mathematics and Descriptive Geometry

Faculty of Civil Engineering
Academic year: 2023/2024
Reg. No.: SvF-16646-104364

MASTER THESIS TOPIC

Student:	**Bc. Ivana Piačková**
Student's ID:	104364
Study programme:	Mathematical and Computational Modeling
Study field:	Mathematics
Thesis supervisor:	Ing. Michal Kollár, PhD.
Head of department:	Ing. Marek Macák, PhD.
Consultant:	prof. RNDr. Karol Mikula, DrSc.

Topic: **Basic hydrological modeling in software NaturaSat**

Language of thesis: English

Specification of Assignment:

The thesis will deal with the implementation of hydrological modeling into the NaturaSat software. Hydrological modeling will be performed by solving the Laplace equation on a computational domain bounded by distributaries, representing the potential wetland area. Dirichlet boundary conditions corresponding to the water level will be specified at the boundary. By solving the Laplace equation, groundwater levels within the computational domain will be obtained, which will be compared with a digital terrain model to determine the optimal water levels in river branches for wetland occurrence. The Laplace equation will be solved on a real computational domain obtained by segmentation from orthophoto images and will be solved using the complementary volume method on an irregular triangular mesh.

Deadline for submission of Master thesis:	09. 05. 2024
Approval of assignment of Master thesis:	09. 05. 2024
Assignment of Master thesis approved by:	prof. RNDr. Karol Mikula, DrSc. – study programme supervisor

Declaration

I declare that this thesis has been composed solely by myself under supervision of my supervisor and using the literature stated in Bibliography.

Bratislava 9. 5. 2024

Ivana Piačková

Acknowledgement

First and foremost, I would like express my sincere gratitude towards my supervisor, Ing. Michal Kollár, PhD., for his help, support, and invaluable guidance. His accessibility and insightful advice were instrumental during the work on the thesis.

I am also grateful to my consultant, Prof. RNDr. Karol Mikula, DrSc., whose introduction to the topic and valuable insights significantly contributed to the development of this thesis.

Last but not least, I would like to express special thanks to my family and my boyfriend for their unending support, encouragement and constructive feedback.

Bratislava 9. 5. 2024

Ivana Piačková

Abstract

Title: Basic hydrological modeling in software NaturaSat
Abstract: The thesis deals with the implementation of hydrological modeling into the NaturaSat software. Hydrological modeling is performed by solving the Laplace equation on a computational domain bounded by distributaries, which represents the area of potential wetland occurrence. Dirichlet boundary conditions corresponding to the water level are specified at the boundary. By solving the Laplace equation, we obtain groundwater levels within the computational domain, which will be compared with a digital terrain model to determine the optimal water levels in river branches for wetland occurrence. The Laplace equation is solved on a real computational domain obtained by segmentation from orthophoto images and is solved using the complementary volume method on an irregular triangular mesh.

Keywords: hydrological modeling, groundwater level, Laplace equation, complementary volume method, triangular grid

Abstrakt

Názov práce: Základné hydrologické modelovanie v softvéri NaturaSat
Abstrakt: Práca sa zaoberá implementáciou hydrologického modelovania do softvéru NaturaSat. Hydrologické modelovanie je vykonávané riešením Laplaceovej rovnice na výpočtovej oblasti ohraničenej riečnymi ramenami, ktorá predstavuje oblasť možného výskytu mokrade. Na hranici sú zadané Dirichletove okrajové podmienky zodpovedajúce výške vodnej hladiny. Riešením Laplaceovej rovnice získame hladiny podzemnej vody vo vnútri výpočtovej oblasti, ktoré budú porovnané s digitálnym modelom terénu na určenie optimálnej hladiny vody v riečnych ramenách pre výskyt močiarov. Laplaceova rovnica je riešená na reálnej výpočtovej oblasti získanej segmentáciou z ortofotosnímok, a je riešená pomocou metódy komplementárnych objemov na nepravidelnej trojuholníkovej sieti.

Kľúčové slová: hydrologické modelovanie, hladina podzemnej vody, Laplaceova rovnica, metóda komplementárnych objemov, trojuholníková sieť

Contents

1 Introduction

Wetlands, regions that are permanently or seasonally flooded or saturated by water, belong among the most productive and biodiverse ecosystems in the world. They fulfill many important functions for both nature and our civilization, including providing habitats for 40% of the world's plant and animal species [22], storing and purifying water, and helping with erosion and flood control [15]. However, wetlands are disappearing at an alarming rate due to human activities, climate change, pollution, and various other factors.

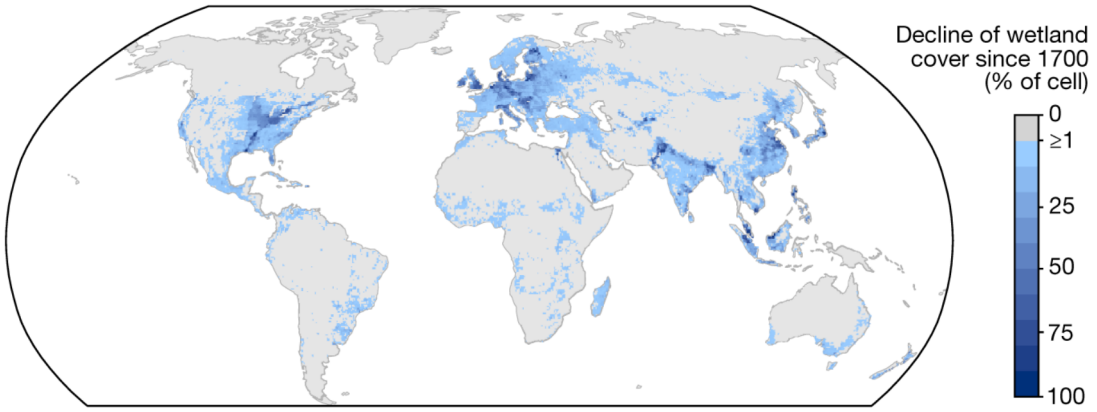


Figure 1.1: Decline of wetland cover since 1700 [3].

Maintaining the health of wetlands requires continuous monitoring, analysis of their conditions, and, if necessary, the application of restoration mechanisms. However, the analysis of wetland conditions through fieldwork alone is both challenging and time inefficient. Our goal is to help with this process by implementing basic hydrological modeling into the environmental software NaturaSat [14], which serves for identifying and monitoring Natura2000 habitats.

One of the wetlands' characteristics indicating their state is the water level. We consider the surface of groundwater to behave as an elastic membrane connecting boundary conditions, so it can be described using a non-linear minimal surface equation. We use the Laplace equation, as it is a good approximation of the minimal surface equation in cases where the gradients of the solution are small. The primary objective of hydrological modeling is to determine the water level within a selected computational region.

The region is defined by a segmentation polygon, which can be obtained through semi-automatic or automatic segmentation within the NaturaSat software [13, 12]. On the boundary of the region we impose Dirichlet boundary conditions, representing the water level at specific vertices of the segmentation polygon.

To solve this boundary problem, we apply the complementary volume method on an irregular triangular grid. Firstly, we discretize the computational region into a triangular grid using the triangulation of a polygonal region (see Chapter 2). Next, we implement the derived numerical scheme of the complementary volume method for the Laplace equation (see Chapter 3). The resulting solution can then be visualized for the user within the NaturaSat software (see Chapter 4). Subsequently, we test the implementation on a synthetic experiment to determine the EOC, as well as on a real experiment involving the region in the Danube basin (see Chapter 5).

The next goal is to compare the solution of the Laplace equation with the digital terrain model (DTM) to identify the relationship between the ground and the water surface (see Chapter 6). This comparison serves to assess the wetland's conditions and determine the optimal water level in surrounding water bodies, especially rivers and streams, for the wetland occurrence, which can help with their restoration.

2 Triangulation

Triangulation, in general, is a subdivision of a planar object into non-overlapping triangles. Within the scope of our application, we specifically employ triangulation for polygons. This process involves partitioning a polygonal area into a set of triangles. The set of triangular facets adheres to the following principles:

- two facets are either disjoint or share a lower-dimensional face, such as an edge or a vertex,
- the union of all facets reconstructs the original area,
- the vertices of each triangle coincide with the vertices of the original polygon.

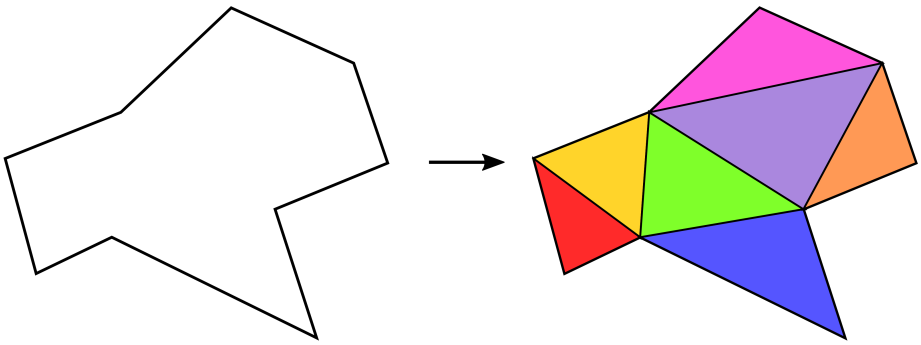


Figure 2.1: Polygon triangulation

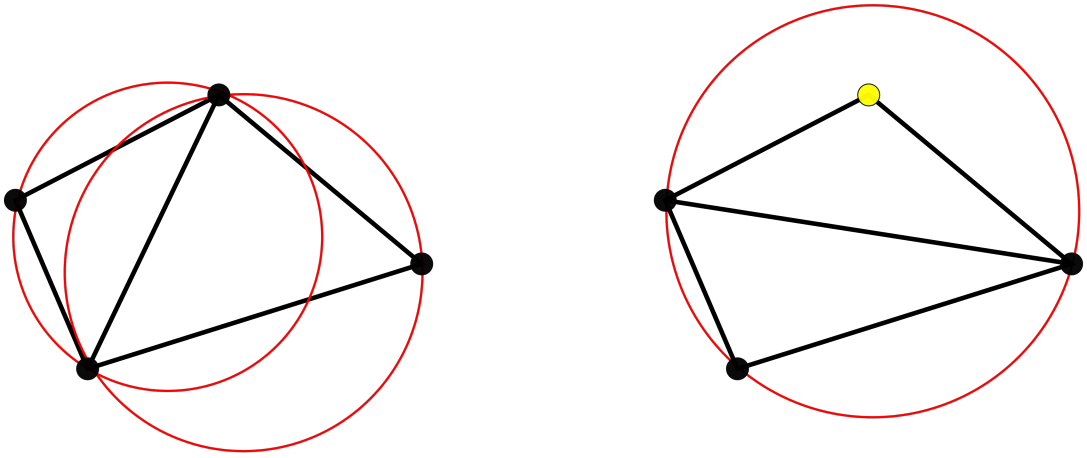
Facets of triangulation can be given an orientation, which subsequently influences the orientation of the edges of the facet. The orientation of two adjacent facets is termed consistent if they induce opposite orientations on their shared incident edge. The entire triangulation can be characterized as orientable when this condition holds for each pair of adjacent facets. [26]

There are several types of triangulations suitable for different uses, such as Delaunay, regular, or constrained triangulation. In our context, we focus on Delaunay triangulation, more specifically constrained Delaunay triangulation, because it consists of only acute triangles of similar size and shape. These characteristics guarantee an easy construction of co-volumes, which is described in the next chapter and there are

no singularities or problematic geometrical features, which leads to efficient numerical calculations.

2.1 Delaunay triangulation

The Delaunay triangulation is a triangulation of a set of points in a plane, which satisfies so-called *Delaunay property* (or *empty circumcircle property*): the circumcircle of any facet contains no point from the given set (Fig.2.2). The circumcircle of a triangle is defined as the circle passing through all three vertices of this triangle. Additionally, Delaunay triangulation is designed to maximize the minimum angle of all triangles which leads to "well-shaped" triangles.



(a) Delaunay triangulation

(b) Non-Delaunay triangulation

Figure 2.2: Two different triangulations on the same set of points showing the Delaunay property. The yellow vertex indicates a violation of this property.

2.1.1 Constrained Delaunay triangulation

Constrained, not necessarily Delaunay triangulation is a type of triangulation that forces specific polylines to be included among the resulting edges (Fig.2.3). These polylines are referred to as *constraints* and the corresponding edges in the triangulation are called *constrained edges*. The basic version of the triangulation process assumes that input constraints do not intersect, except at their endpoints. While it is possible to generate a constrained triangulation even when constraints intersect, such cases require the introduction of new vertices at the intersection points. [26]

Specifically, constrained Delaunay triangulation presents a triangulation method that simultaneously fulfills the empty circumcircle property of Delaunay triangulation and includes the specified input constraints into the resulting edges.

This method aligns with our application objectives, as we want to create a primal triangular grid on the area defined by a segmentation polygon. Segmentation polygons are closed polygons that result from semi-automatic and automatic segmentation of the observed biotopes, particularly wetlands, using the NaturaSat software [13, 12]. It is necessary to preserve the edges given by segmentation polygons within the grid.

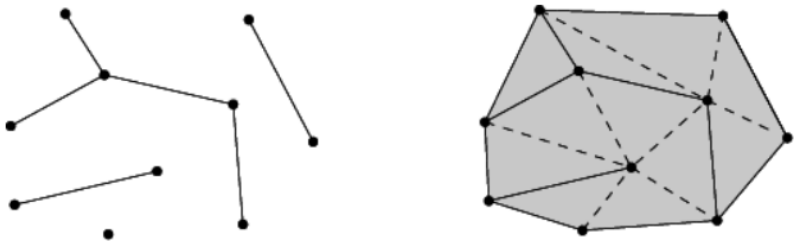


Figure 2.3: Set of points and constraints (left) and the resulting constrained triangulation (right). [26]

2.2 Implementation using CGAL library

Numerous algorithms exist for triangulation and there are many libraries that implement them, including CGAL [21], CDT [11], Triangle [20] or Fade2D [4]. In our case, CGAL appears as the most fitting choice, so we use it for the triangulation process. CGAL (The Computational Geometry Algorithms Library) is an open-source library that provides computational geometry algorithms in C++. The library covers a wide range of topics, with different triangulations among them.

CGAL provides complete triangulations which means that they cover the convex hull of the set of vertices. This may be problematic when creating a grid over concave areas or regions with holes inside. To overcome this issue, the domain of triangulation is introduced. By defining a domain, we can access only the triangles within it and ignore ones outside the domain.

Triangulation in CGAL is represented by faces and vertices rather than by edges. This approach saves storage space and results in faster algorithms. Each face gives access to its three incident vertices and its three adjacent faces. Each vertex gives access to one of its incident faces and through this face to the circular list of its all incident faces.

For manipulating the triangulation, we address the triangulation classes which provide various functionalities, such as the location of a point, insertion, removal, or displacement of a point. These classes have two template parameters that separate the geometric and combinatorial aspect of the triangulation:

- a **geometric traits** class providing the geometric primitives and the elementary operations,

- a **triangulation data structure** class, which defines the faces and vertices of the triangulation and enables us to access them. [17]

2.2.1 Generating constrained Delaunay triangulation

Generating any triangulation in CGAL requires the definition of an object of the corresponding triangulation class. For the constrained Delaunay triangulation, we opt for **Constrained_Delaunay_triangulation_2** with the following template parameters: *Traits* for the geometric aspect, *Tds* for the triangulation data structure, and *Itag*, which allows the user to select if intersecting constraints are supported.

Subsequently, the user needs to input the points and the constraints into this object. The points are inserted into triangulation using the *insert* function, which supports inserting individual points and also an array of points specified by their iterator range. The constraints of the triangulation can be defined using the function *insert_constraint*. This function allows the insertion of individual line segments, given by their endpoints in the form of vertex handles. Additionally, it enables the user to insert constraints as a pre-defined polyline, such as the **Polygon_2** type of CGAL [5]. This approach is particularly relevant for our application, as we are dealing with segmentation polygons. The **Polygon_2** object is filled with points of a polygon using the function *push_back*.

The insertion of these basic geometric entities into the triangulation object automatically leads to a triangulation generation. It means that this process does not require calling any additional function. The created triangulation can be visualized using the *draw* function, which takes the triangulation object as an argument. A showcase of the output of this function is presented in Fig. 2.4, where we demonstrate constrained Delaunay triangulation on both a square and a discretized circle.

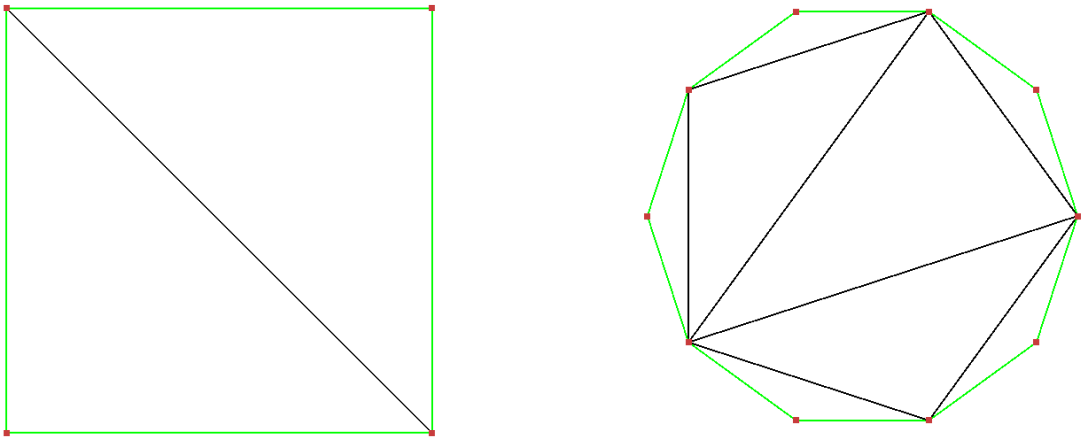


Figure 2.4: Examples of constrained Delaunay triangulation. Red points correspond to given vertices and green edges to given constraints.

Note: CGAL's constrained Delaunay triangulation tries to be as much Delaunay as possible. However, the resulting triangles do not necessarily fulfill *the empty circumcircle property*. Instead, they fulfill a weaker condition called *the constrained empty circumcircle property*: the circumcircle of any triangle does not enclose any vertex visible from the interior of the triangle. In this context, visibility is obstructed by constrained edges, which we consider as blocking the view.

2.2.2 Mesh refinement

At this point, we can establish a basic triangulation on a set of points and constraints. For the application of the complementary volume method on this discretization, a significantly denser primal grid is required. Mesh density is very important in numerical methods as increased mesh density leads to more accurate results.

In CGAL, a mesh is defined as a partition of a specified region into simplices, satisfying several criteria related to shape and size. The region intended for meshing is referred to as the domain and it has to be a bounded region. The domain is defined by a planar straight line graph - its segments are constraints that will be represented as edges in the mesh. Additionally, it can contain isolated points that will be represented as vertices of the mesh.

The user can define various criteria that the generated mesh should satisfy. The first criterion is to set which specified components of the region are to be meshed or on the other side, not meshed. This proves useful particularly when dealing with regions containing internal holes that are not intended for meshing. This criterion can be set either by specifying the domain using the *mark_domain_in_triangulation* function or by specifying a set of seed points (see Fig.2.5).

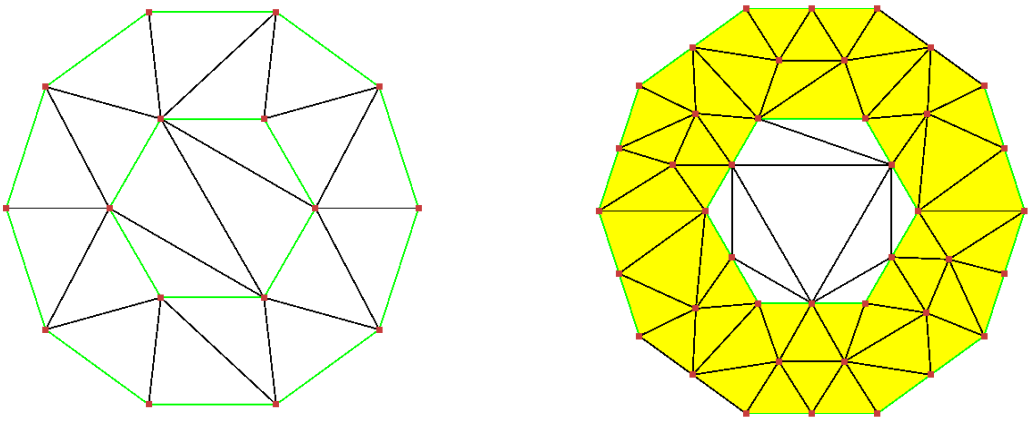
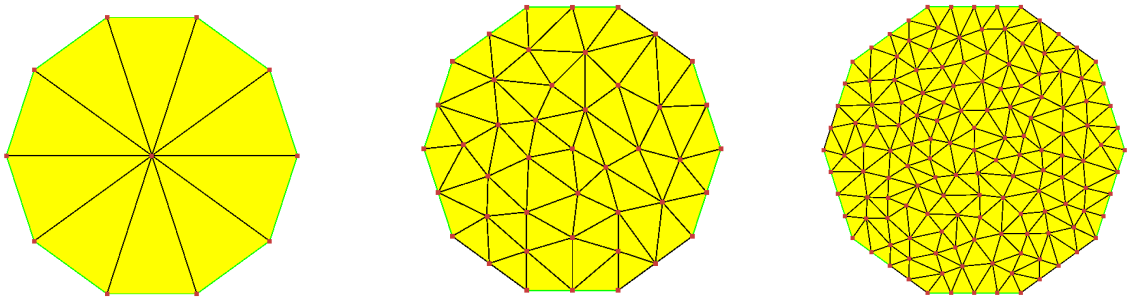


Figure 2.5: Triangulation and mesh generated on a discretized unit circle with an inside hole. The hole is identified by setting the domain (yellow) and new vertices are created only inside it.



(a) Default shape and size criteria

(b) Default shape criterion and size bound $S = 0.5$

(c) Default shape criterion and size bound $S = 0.25$

Figure 2.6: Mesh generated on a discretized unit circle with different size criteria.

The second criterion deals with the shape of triangles. It sets a lower bound B on the ratio between the circumradius and the shortest edge length. This implies a lower bound of $\arcsin(\frac{1}{2B})$ on the minimum angle for each triangle.

Lastly, the user can influence the size of triangles, directly impacting mesh density. The size criterion sets an upper bound S for the length of all segments of all triangles (see Fig.2.6).

The mesh generation algorithm in CGAL [18] starts with a constrained Delaunay triangulation and then produces the final mesh using the Delaunay refinement method. This method introduces new vertices to the triangulation and stops once the specified criteria are satisfied. In our case, we already have the constrained Delaunay triangulation which serves as an input for the mentioned refinement method. It can be accessed through the *refine_Delaunay_mesh_2* function with two arguments: the triangulation and the criteria. Alternatively, the user can choose to use the **Delaunay_mesher_2** class, which implements a 2D mesh generator. The object of this class is initiated with a triangulation and using functions *set_seeds* and *set_criteria* allows for the specification of mesh criteria.

3 Complementary volume method for solving the Laplace equation

Laplace equation is a second-order partial differential equation

$$\Delta u(x) = 0 \quad x \in \Omega, \tag{3.1}$$

where Δ represents the Laplace operator, u is a twice-differentiable function, and Ω is the domain for solving the equation. In our case, Ω corresponds to the region defined by the segmentation polygon. When Ω is a closed region, solving the equation requires specifying boundary conditions

$$u(x) = u_0(x) \quad x \in \partial\Omega, \tag{3.2}$$

where $u_0(x)$ is a given function and $\partial\Omega$ is the boundary of Ω.

3.1 Spatial discretization

The Laplace equation (3.1) can be numerically solved using the complementary volume method, which combines the finite volume method [2] and the finite element method. The initial step is to discretize the domain Ω into co-volumes. The domain Ω is already approximated by a triangulation (see Chapter 2). As mentioned in the previous chapter, we use the Delaunay triangulation, which is composed solely of acute triangles of similar size and shape and thus is suitable for the complementary volume method, guaranteeing both efficiency of numerical calculations and easy construction of co-volumes. The triangulation contains a finite number of nodes

$$X_i, \quad X_i \in \Omega, \quad i = 1, \ldots, N,$$

which correspond to the vertices of the triangular grid. The value u_i represents the solution at node X_i. The triangles of this grid with vertex X_i are referred to as

$$T_{iq}, \quad i = 1, \ldots, N, \quad q = 1, \ldots, Q_i,$$

where Q_i is the number of triangles sharing the vertex X_i. The remaining two vertices of the triangle T_{iq} are denoted by X_i^{q1} and X_i^{q2}.

Now, we can construct a co-volume grid based on the triangulation [9]. For each vertex X_i, we define a co-volume V_i (Fig.3.1) as a polygon with vertices at:

- the centers of mass of all triangles sharing the vertex X_i,
- the midpoints of line segments between the vertex X_i and the neighboring vertices X_i^{q1} and X_i^{q2}.

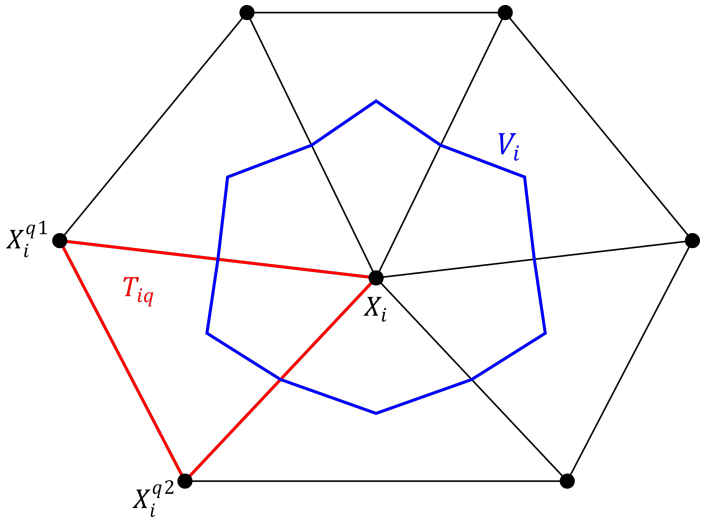


Figure 3.1: Co-volume V_i

3.2 Numerical scheme

By integrating equation (3.1) over the co-volume V_i, we get

$$\int_{V_i} \Delta u \, dx = 0, \tag{3.3}$$

which can be further adjusted by applying Green's theorem:

$$\int_{\partial V_i} \nabla u \cdot \vec{\eta}_i \, ds = 0, \tag{3.4}$$

where ∇u is the gradient of the function u and $\vec{\eta}_i$ is the outer unit normal vector to the finite volume boundary ∂V_i.

Considering the geometry of the boundary ∂V_i, it can be divided into separate boundary components ∂V_{iq} corresponding to individual triangles T_{iq}, where V_{iq} denotes

$V_i \cap T_{iq}$. The integral in (3.4) is then expressed as

$$\sum_{q=1}^{Q_i} \int_{\partial V_{iq}} \nabla u \cdot \vec{\eta}_{iq}\, ds = 0, \tag{3.5}$$

where $\vec{\eta}_{iq}$ is the outer unit normal vector to the boundary component ∂V_{iq}.

As depicted in Fig. 3.2, there are two edges e_{iq}^1 and e_{iq}^2 of the co-volume V_i inside the triangle T_{iq} with different outer unit normal vectors $\vec{\eta}_{iq}^{\,j}$, $j = 1, 2$. Therefore, the integral in (3.5) can be further decomposed into two integrals, resulting in:

$$\sum_{q=1}^{Q_i} \sum_{j=1}^{2} \int_{e_{iq}^j} \nabla u \cdot \vec{\eta}_{iq}^{\,j}\, ds = 0. \tag{3.6}$$

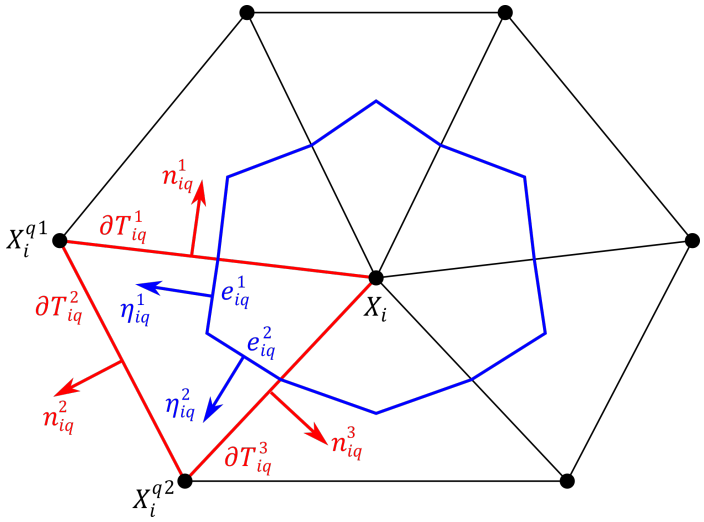


Figure 3.2: Portion of the co-volume V_i corresponding to the triangle T_{iq} with highlighted edges of co-volume e_{iq}^1 and e_{iq}^2, outer unit normal vectors to the co-volume edges $\vec{\eta}_{iq}^1$ and $\vec{\eta}_{iq}^2$, the triangle edges ∂T_{iq}^1, ∂T_{iq}^2 and ∂T_{iq}^3, and outer unit normal vectors to the triangle edges $\vec{n}_{iq}^1$, $\vec{n}_{iq}^2$ and $\vec{n}_{iq}^3$.

Considering a linear representation of a solution u on individual triangles T_{iq}, the gradient ∇u is a constant vector on each triangle. The gradient can be approximated by the mean value

$$\nabla u \approx \frac{1}{m(T_{iq})} \int_{T_{iq}} \nabla u\, dx, \tag{3.7}$$

where $m(T_{iq})$ represents the area of triangle T_{iq}. Applying Green's theorem on the previous equation (3.7) we obtain:

$$\nabla u \approx \frac{1}{m(T_{iq})} \int_{\partial T_{iq}} u \vec{n}_{iq}\, ds, \tag{3.8}$$

where $\vec{n}_{iq}$ is the outer unit normal vector to the boundary of the triangle T_{iq}.

The integral in the equation (3.8) can be divided into three integrals over each edge of the triangle. Thus, we get the equation with a sum:

$$\nabla u \approx \frac{1}{m(T_{iq})} \sum_{j=1}^{3} \int_{\partial T_{iq}^j} u \vec{n}_{iq}^j \, ds, \tag{3.9}$$

where ∂T_{iq}^j is the j-th edge of the triangle and $\vec{n}_{iq}^j$ is the normal vector to this edge, as shown in Fig. 3.2. The normal vector to the edge of the triangle is constant, thus it can be put outside of the integral:

$$\nabla u \approx \frac{1}{m(T_{iq})} \sum_{j=1}^{3} \vec{n}_{iq}^j \int_{\partial T_{iq}^j} u \, ds. \tag{3.10}$$

The remaining integral can be, as a consequence of the linear representation of u, expressed as an average value on the corresponding edge ∂T_{iq}. For example, for the first edge ∂T_{iq}^1 between the vertices X_i and X_i^{q1} it is:

$$\int_{\partial T_{iq}^1} u \, ds = \frac{u_i + u_{q1}}{2} m(\partial T_{iq}^1). \tag{3.11}$$

Let denote the length of the triangle edges $m(\partial T_{iq}^j)$ as d_{iq}^j. Then, we get a constant representation of the gradient ∇u on the triangle T_{iq}, denoted as $\vec{P}_{T_{iq}}$:

$$\vec{P}_{T_{iq}} = \frac{1}{m(T_{iq})} \left(\frac{u_i + u_{q1}}{2} d_{iq}^1 \vec{n}_{iq}^1 + \frac{u_{q1} + u_{q2}}{2} d_{iq}^2 \vec{n}_{iq}^2 + \frac{u_i + u_{q2}}{2} d_{iq}^3 \vec{n}_{iq}^3 \right), \tag{3.12}$$

where u_i, u_{q1} and u_{q2} denote the solution in the vertices of the triangle T_{iq}. Values d_{iq}^1, d_{iq}^2 and d_{iq}^3 represent the lengths of the triangle edges and vectors $\vec{n}_{iq}^1$, $\vec{n}_{iq}^2$ and $\vec{n}_{iq}^3$ are the outer unit normal vectors to each side of the triangle T_{iq} (see Fig. 3.2).

Now, we substitute the gradient with its approximation (3.12) in the equation (3.6):

$$\sum_{q=1}^{Q_i} \sum_{j=1}^{2} \int_{e_{iq}^j} \vec{P}_{T_{iq}} \cdot \vec{\eta}_{iq}^j \, ds = 0. \tag{3.13}$$

As $\vec{P}_{T_{iq}}$ is a constant vector, it can be factored out of the integral. The remaining integral of the outer unit normal vector to the co-volume boundary part e_{iq}^j is expressed as:

$$\int_{e_{iq}^j} \vec{\eta}_{iq}^j \, ds = m(e_{iq}^j) \vec{\eta}_{iq}^j, \tag{3.14}$$

where $m(e_{iq}^j)$ is the length of the co-volume edge e_{iq}^j. Consequently, in (3.13), we

eliminate the integral, resulting in the equation

$$\sum_{q=1}^{Q_i} \sum_{j=1}^{2} m(e_{iq}^j) \vec{P}_{T_{iq}} \cdot \vec{\eta}_{iq}^j = 0. \tag{3.15}$$

It represents the numerical scheme for solving Laplace equation.

3.3 System of linear equations

The obtained numerical scheme (3.15) gives us for $i = 1, \ldots, N$ a system of linear equations in the form:

$$A\vec{u} = \vec{0}, \tag{3.16}$$

where A denotes the system matrix and $\vec{u} = [u_1, u_2, \ldots, u_N]$ is the vector of nodal values of the solution. In (3.15), these unknown nodal values are hidden inside the gradient approximation $\vec{P}_{T_{iq}}$.

To determine the individual coefficients of the matrix A, we need to consider that the nodal value of the solution at a specific vertex must be consistent across all triangles sharing that vertex. The definition of the gradient approximation (3.12) contains nodal values in local numbering based on triangles (e.g. u_{q1} or u_{q2}) and we need to identify values corresponding to the same node. According to the geometry of the co-volume V_i, we get:

$$u_{iq1} = \begin{cases} u_{iQ_i2}, & \text{if } q = 1 \\ u_{i(q-1)2}, & \text{otherwise} \end{cases} \tag{3.17}$$

To simplify the numbering, nodal values for the co-volume V_i are denoted as u_i for the central vertex and $u_{i1}, \ldots, u_{iQ_i}$ for the boundary vertices (see Fig. 3.3). Using this numbering, we can transform the equation (3.15) and isolate each nodal value:

$$a_i u_i + \sum_{q=1}^{Q_i} a_{iq} u_{iq} = 0, \tag{3.18}$$

where a_i and a_{iq} for $q = 1, \ldots, Q_i$ are non-zero coefficients present in the i-th row of the system matrix A. These coefficients are assigned to specific columns according to the global numbering of nodes in the whole grid.

The diagonal coefficient a_i can be expressed as

$$a_i = \sum_{q=1}^{Q_i} \frac{1}{2m(T_{iq})} \sum_{j=1}^{2} m(e_{iq}^j) \vec{\eta}_{iq}^j \cdot (d_{iq}^1 \vec{n}_{iq}^1 + d_{iq}^3 \vec{n}_{iq}^3). \tag{3.19}$$

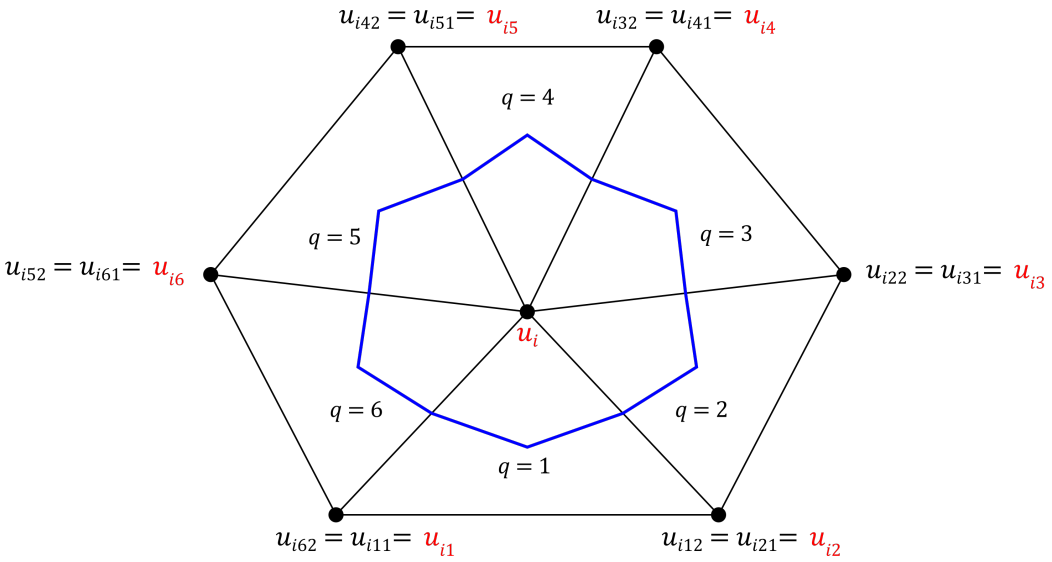


Figure 3.3: Local numbering of nodal values based on triangles numbered $q = 1, \ldots, Q_i$, where $Q_i = 6$ for the co-volume V_i and its simplification (red numbering).

The non-diagonal coefficient a_{iq} corresponds to the nodal value u_{iq} in the vertex, which neighbors with two triangles indexed q and k, where

$$k = \begin{cases} Q_i, & \text{if } q = 1 \\ q - 1, & otherwise \end{cases}$$

and it can be expressed as

$$\begin{aligned} a_{iq} = & \frac{1}{2m(T_{iq})} \sum_{j=1}^{2} m(e_{iq}^j) \vec{\eta}_{iq}^j \cdot (d_{iq}^1 \vec{n}_{iq}^1 + d_{iq}^2 \vec{n}_{iq}^2) \\ & + \frac{1}{2m(T_{ik})} \sum_{j=1}^{2} m(e_{ik}^j) \vec{\eta}_{ik}^j \cdot (d_{ik}^3 \vec{n}_{ik}^3 + d_{ik}^2 \vec{n}_{ik}^2). \end{aligned} \tag{3.20}$$

Finally, we can include the boundary conditions into the system, since the vertices lying on the boundary of the region have known values of the solution. The respective rows of the system matrix have their diagonal elements equal to 1 and zeroes elsewhere. On the right-hand side, we insert the boundary conditions for the corresponding vertices. Now, we can solve the system for the unknown values at the inner vertices of the grid.

3.4 Implementation

After the triangulation process, the software creates a co-volume for each vertex. Therefore, we introduce the **FiniteVolume** class, designed to compute and store all information about each co-volume and also the supporting **Triangle** class.

The **Triangle** class is used to simplify the **FiniteVolume** class. It contains characteristics corresponding to the specific face around the central vertex:

- all vertices of the face (*vertices*),
- the lengths of the edges of the triangle (*edgeLengths*),
- the normal vectors to the triangle edges (*edgeNormals*),
- the lengths of the co-volume edges - only the edges internal to the face (*volumeLengths*),
- the normal vectors to the co-volume edges - only the edges internal to the face (*volumeNormals*),
- the area of the face (*area*).

These attributes are essential for computing the matrix coefficients (see equations (3.19) and (3.20)), and are initialized within the constructor of this class.

The parametrized constructor requires the input of the triangle's vertices, which are subsequently assigned to the *vertices* variable. They are then used within the functions *computeEdgeCharasteristics* and *computeVolumeCharacteristics* for computing the remaining variables.

Calculating the length of a line segment defined by two vertices with coordinates (x_1, y_1) and (x_2, y_2) is done by the formula

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

This formula applies to both the lengths of the triangle edges and the co-volume edges. Calculating the former is straightforward since the coordinates of the triangle vertices are known. However, the latter requires a calculation of the coordinates of the midpoint between two vertices and the centroid beforehand. The midpoint is computed as the average of vertices' coordinates of the corresponding edge and the centroid is computed as the average of the coordinates of all triangle's vertices:

$$T = \left(\frac{x_1 + x_2 + x_3}{3}, \frac{y_1 + y_2 + y_3}{3} \right).$$

Now, we know the coordinates of points of both triangle edges and co-volume edges and we determine their respective tangent vectors. Since the normal vectors are orthogonal to the tangent vectors, we simply rotate the tangent vectors by 90 degrees. We need to ensure that the normal vectors are unit vectors, so we divide them by the length of the corresponding edge. Moreover, the normal vectors must point outward

from the triangle or the co-volume. As the triangulation is orientable, rotating the tangent vector by +90 degrees always gives an outer normal vector.

Finally, we compute the area of the triangle using Heron's formula [25]. Assuming the lengths of the triangle edges d_1, d_2, d_3, we begin by calculating the triangle's semiparameter:

$$s = \frac{d_1 + d_2 + d_3}{2},$$

and then determine the area as:

$$area = \sqrt{s(s - d_1)(s - d_2)(s - d_3)}.$$

The class **FiniteVolume** deals with a co-volume as a whole and enables us to compute diagonal and non-diagonal coefficients of the system matrix corresponding to a specific vertex and its respective row. It contains

- a vertex handle for the central vertex (v),
- the number of faces sharing the central vertex (*facesNumber*),
- the handles to the faces sharing the central vertex (*faces*),
- the **Triangle** objects corresponding to each face of *faces* (*triangles*).

To initialize these variables, the *initializeFiniteVolume* function is provided, which requires the generated triangulation as an argument. The triangulation offers information about the incident faces to the central vertex of the co-volume. Subsequently, we determine the value of *facesNumber* and the coordinates of vertices for each face. The coordinates are used to create a **Triangle** object for each face, as described earlier.

This class involves two functions to compute the matrix coefficients, namely *computeDiagonalElement* and *computeOffDiagonalElements*. These functions use the precalculated characteristics of each triangle in the co-volume in formulas (3.19) and (3.20).

For solving a system of equations, we utilize the Eigen library, a C++ template library for linear algebra [8]. Eigen offers various methods for solving linear systems. We specifically choose the BiCGSTAB - biconjugate gradient stabilized method for sparse square problems [23]. The corresponding Eigen function requires the system matrix in the form of a **SparseMatrix** object from the library, along with the right-hand side of the system in the form of a vector.

The **SparseMatrix** object is constructed from triplets $(row, column, value)$. The triplets are determined by iterating through all finite volumes, computing their matrix coefficients using previously mentioned functions of **FiniteVolume** class, and assigning them to the specific row and column based on the global numbering of vertices. At the same time, we fill the right-hand side vector with zeros. However, we need to consider

the boundary co-volumes with predefined boundary conditions. For these co-volumes, the only non-zero value in the row is the diagonal coefficient with a value of 1, and the corresponding element of the right-hand side is filled with the given boundary condition.

Insertion of the matrix and the right-hand side vector into the *BiCGSTAB* function of Eigen results in the solution vector. The solution can be visualized through various means, such as using the VTK format or by interpolating the solution onto the regular grid (see Chapter 4).

4 Visualization

The visualization of the solution to the problem described in Chapter 3 aids in better understanding the output. Various methods can be employed to visualize the vector of solution on the triangular grid. Our approach includes visualization using VTK files [24] and interpolation of the solution onto a regular grid.

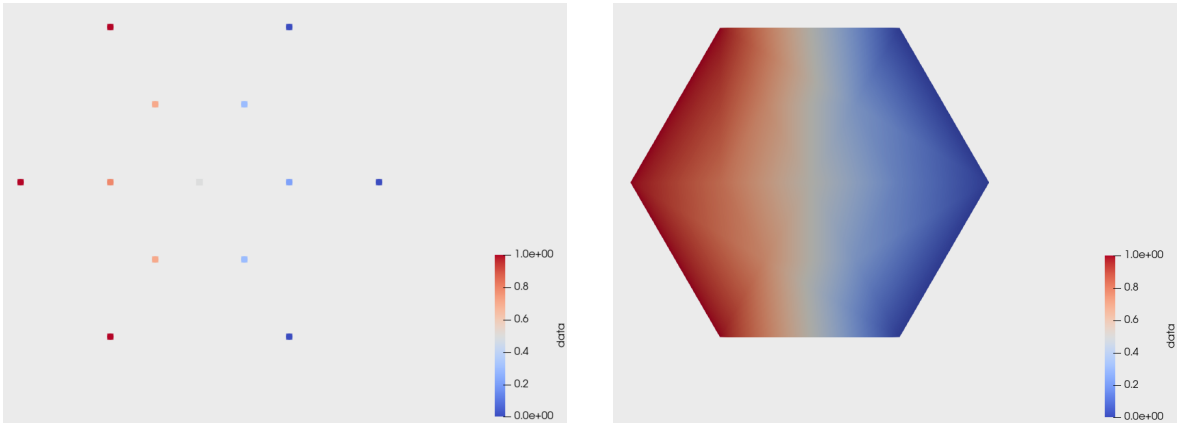
4.1 Using VTK file

We initially explored visualization through VTK files. VTK (The Visualization Toolkit) is an open-source software system for 3D computer graphics, image processing, modeling, volume rendering, and scientific visualization [19]. It utilizes a VTK file format, compatible with visualization applications like ParaView. ParaView is an open-source post-processing visualization engine that uses VTK as the data processing and rendering engine [16].

The most important aspect is correctly setting up the VTK file within the program. Firstly, we set a count of points and then specify their coordinates. Then, we define the triangles of the triangular grid as polygons, represented by the indices of their vertices. This process captures the geometry of the region within the VTK file. Finally, we insert information about the solution, which can be visualized by assigning different colors to each point on the grid.

When loading the file into ParaView, various representations of the geometry can be displayed (Fig. 4.1). When we choose the surface representation, ParaView uses color data present at the geometry's points and interpolates it across the remaining internal points of the geometry.

However, utilizing this method requires access to a visualization engine compatible with VTK files. To overcome this disadvantage, we decided to implement interpolation of the solution onto the regular grid, which can be saved as an image, because the image is a set of pixels lined up in a regular grid. The advantage of image format is that it is much more accessible because it can be viewed through more commonly used applications and also we can easily display it within NaturaSat software.



(a) The point representation in ParaView (b) The surface representation in ParaView

Figure 4.1: The Laplace equation solved on a discretized circle with half of the boundary given boundary condition with value of 0 and the remaining part with value of 1. Example of the visualization using VTK file inside ParaView application.

4.2 Interpolation onto the regular grid

Interpolation is the process of estimating unknown values that lie between existing data points. In our case, these data points correspond to the vertices of the triangular grid, where we know the solution to the Laplace equation. Through interpolation, we can determine the value of the solution at other points within the considered region and visualize them as an image.

To encompass the data within an image, we first need to define a regular grid corresponding to the set of pixels. This rectangular grid must cover the entire region defined by the segmentation polygon, serving as a rectangular hull. Pixels outside of this region are assigned the minimal value associated with the black color in the image. Conversely, pixels located within the triangles of the triangular grid are assigned varying shades of grey based on the interpolated value.

The final image does not have to be grayscale, as we can use a color gradient in RGB channels. In this case, we choose specific colors representing specific values of the solution and when interpolating the solution data on the regular grid, we assign corresponding values to all three RGB channels.

4.2.1 The regular grid

The first step is to define the regular grid, specifically its pixel size. The pixel size needs to be chosen carefully, as too large sizes can result in the loss of details and too small sizes increase computational time. In initial tests, we chose to set the pixel size based on the shortest edge between two adjacent vertices of the triangulation.

Next, we determine the dimensions of the rectangular grid. Using CGAL functionalities which enable us to find boundary points of a polygon, we identify the coordinates

of the top-left and bottom-right points of the regular grid. Now, we can compute the number of pixels in both the $x-$ and $y-$directions:

$$envWidth = \frac{bottomRight.x - topLeft.x}{pixelSize}, \tag{4.1}$$

$$envHeight = \frac{topLeft.y - bottomRight.y}{pixelSize}. \tag{4.2}$$

These characteristics of the regular grid are used to iterate over each pixel of the image and identify its color according to the calculated solution through interpolation.

4.2.2 Pixel localization

To correctly assign colors to each pixel of the regular grid, it is necessary to determine whether a pixel lies outside or inside the region of interest. Outer pixels are simply assigned the minimal value, behaving as a black background in the image. On the other hand, assigning the color to the inner pixels is more challenging as we need to identify to which triangle of the triangular grid each inner pixel belongs. Subsequently, we get the final color of the pixel by applying interpolation based on the identified triangle.

For this purpose, we iterate through each pixel in a cycle. A pixel does not have to entirely lie within one triangle, therefore we localize it based on its center given as:

$$pixelCentre = \big((i + 0.5) * pixelSize, (j + 0.5) * pixelSize\big), \tag{4.3}$$

where $i = 0, \ldots, envWidth$ and $j = 0, \ldots, envHeight$. We then check if this point belongs to any face of the triangular grid. Many methods exist that help us determine if a point lies inside a triangle. We specifically implemented an algorithm that assesses on which side of the half-plane created by the triangle's edges the point is located.

Assume the triangle ABC and point P whose location we want to check. For the triangle's edge AB, we determine if P lies to the left or right of the line defined by points A and B. Let's define two vectors: u, lying on the edge AB in any orientation, and v, starting from the same point and aiming to the point P (see Fig. 4.2). The orientation of u is arbitrary but must be consistent for all edges of the triangle. We consider these vectors as 3D vectors with zero z-coordinate:

$$\vec{u} = A - B = (A_x - B_x, A_y - B_y, 0),$$
$$\vec{v} = P - B = (P_x - B_x, P_y - B_y, 0),$$

because the half-plane containing the point P is determined using the cross product of these vectors.

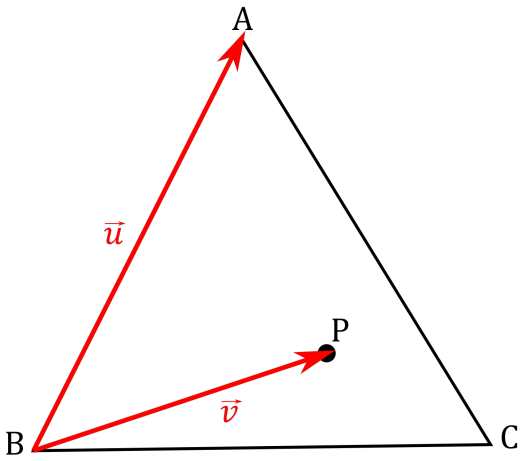


Figure 4.2: The determination of the point P's position relative to the edge AB using vectors $\vec{u}$ and $\vec{v}$.

For these vectors, the cross product yields non-zero value only in the z-coordinate:

$$(\vec{u} \times \vec{v})_z = (A_x - B_x)(P_y - B_y) - (A_y - B_y)(P_x - B_x)$$

and we are interested in its sign.

Algorithm 1: Pixel localization - check if the pixel's center lies inside the triangle of the triangular grid

Input: triangle's vertices A, B and C, the point of interest P

`/* Compute` z`-coordinate of the cross product between a vector corresponding to triangle's edge and a vector pointing to the point` P `*/`

$sgn1 \leftarrow (A_x - B_x) * (P_y - B_y) - (A_y - B_y) * (P_x - B_x)$

$sgn2 \leftarrow (B_x - C_x) * (P_y - C_y) - (B_y - C_y) * (P_x - C_x)$

$sgn3 \leftarrow (C_x - A_x) * (P_y - A_y) - (C_y - A_y) * (P_x - A_x)$

`/* Check for the same signs of cross products */`

$plus \leftarrow (sgn1 \geq 0) \wedge (sgn2 \geq 0) \wedge (sgn3 \geq 0)$

$minus \leftarrow (sgn1 \leq 0) \wedge (sgn2 \leq 0) \wedge (sgn3 \leq 0)$

$isInsideTriangle \leftarrow plus \vee minus$

We repeat the process and get three signs for all three edges of the triangle. If the signs are the same (either negative or positive), the point lies within the triangle. Otherwise, it lies outside the triangle. It is important to account for cases where the point lies directly on the triangle's boundary, indicated by a zero z-coordinate in the cross product. The whole algorithm is shown in Alg. 1.

Once we establish that the point is located inside the triangle, the next step involves interpolating the solution at the triangle's vertices to this point.

4.2.3 Barycentric interpolation

There are various types of interpolation, such as:

- nearest neighbor interpolation,
- bilinear interpolation,
- bicubic interpolation,
- barycentric interpolation (only for triangles), …

Since the solution is given in the vertices of the triangular grid, we choose the barycentric interpolation.

The barycentric interpolation is based on barycentric coordinates on a triangle. Assume triangle ABC and barycentric coordinates λ_0, λ_1, λ_2 defined on this triangle. Then the coordinates of an arbitrary point P inside the triangle or on its boundary can be expressed as

$$P = \lambda_0 A + \lambda_1 B + \lambda_2 C, \tag{4.4}$$

where $\lambda_0, \lambda_1, \lambda_2 \geq 0$. Additionally, barycentric coordinates fulfill property:

$$\lambda_0 + \lambda_1 + \lambda_2 = 1. \tag{4.5}$$

The barycentric coordinates λ_0, λ_1, λ_2 of point P are equivalent to the ratios of the areas of PBC, PCA and PAB to the area of the reference triangle ABC (Fig. 4.3). Let denote:

- the area of a triangle ABC as A,
- the area of a triangle PBC as A_0,
- the area of a triangle PCA as A_1,
- the area of a triangle PAB as A_2,

then:

$$\lambda_0 = \frac{A_0}{A}, \quad \lambda_1 = \frac{A_1}{A}, \quad \lambda_2 = \frac{A_2}{A}. \tag{4.6}$$

The problem of determining the barycentric coordinates of a point inside the triangle is reduced to finding the areas of individual triangles created inside the reference triangle. To solve this problem we need to compute the area using available information about the triangle.

Let define two vectors $\vec{u} = (B - A)$ and $\vec{v} = (C - A)$ as shown in Fig. 4.3. Applying the cross product on these vectors we get a normal vector to the triangle ABC. The magnitude of the cross product equals to the area of a parallelogram with

the vectors as its sides. As we are working with triangle, its area is a half of the area of a parallelogram:

$$A = \frac{\|\vec{u} \times \vec{v}\|}{2}. \tag{4.7}$$

The cross product in (4.7) is a normal vector with only non-zero z-coordinate, because the triangle lies in a plane $z = 0$:

$$\vec{u} \times \vec{v} = (0, 0, u_x v_y - u_y v_x). \tag{4.8}$$

Then the formula for the area can be adjusted as:

$$\begin{aligned} A &= \frac{\sqrt{(u_x v_y - u_y v_x)^2}}{2} = \frac{|u_x v_y - u_y v_x|}{2} \\ &= \frac{|(B_x - A_x)(C_y - A_y) - (B_y - A_y)(C_x - A_x)|}{2}. \end{aligned} \tag{4.9}$$

Analogically, we compute the areas of inner triangles A_0, A_1, and A_2. It is sufficient to compute two of these areas because the third barycentric coordinate can be determined using the property (4.5).

$$A_0 = \frac{|(B_x - P_x)(C_y - P_y) - (B_y - P_y)(C_x - P_x)|}{2} \tag{4.10}$$

$$A_1 = \frac{|(A_x - P_x)(C_y - P_y) - (A_y - P_y)(C_x - P_x)|}{2} \tag{4.11}$$

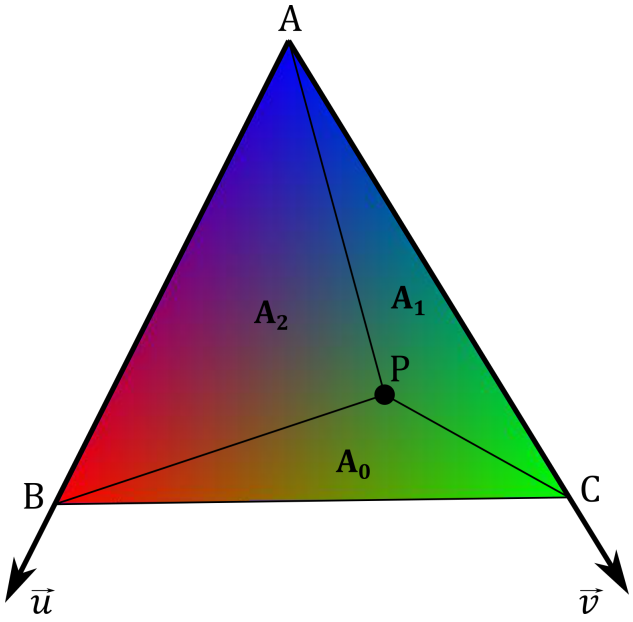


Figure 4.3: Barycentric interpolation on a triangle ABC with inner point P and marked area of inner triangles PBC, PCA and PAB. Vectors $\vec{u}$ and $\vec{v}$ are used to compute the area of the reference triangle ABC.

Inputting the areas into the formulas for barycentric coordinates (4.6), we obtain λ_0 and λ_1. The third coordinate λ_3 is determined as

$$\lambda_2 = 1 - \lambda_0 - \lambda_1. \tag{4.12}$$

Now, since we know the barycentric coordinates of the point P inside the reference triangle ABC, we want to determine the value of the solution at this point. Assume three values of solution u_0, u_1 and u_2 given in the vertices of the triangle, then the value in point P can be computed as

$$u_P = \lambda_0 u_0 + \lambda_1 u_1 + \lambda_2 u_2. \tag{4.13}$$

Finally, we know the values of the solution for the entire regular grid, both for background pixels and pixels inside the region of interest. The program uses these values to create an image that is easy to display.

5 Numerical experiments

In this chapter, we present the application of the basic hydrological modeling implementation in NaturaSat software through different numerical experiments. These experiments progressively transition from synthetic scenarios to real situations. We focus specifically on defining a region of interest, the result of triangulation and the visualization of the solution.

For the implementation, we first need to provide the region as a set of vertices along its boundary. The vertices can either be synthetic for a specific type of boundary or obtained by segmentation within NaturaSat software. Subsequently, we proceed to create a triangulation of the region with predefined criteria. By default, we set the size criterion for the length of the triangles' edges as the distance between two neighboring vertices on the boundary. In some cases, we may adjust the size criterion to create a denser grid. The final triangulation is not directly accessible to the user but can be exported and saved into a VTU file for visualization in ParaView. Finally, after computing the solution, we interpolate the values of the solution to RGB channels for color visualization using a linear color gradient. This color gradient is configured with three nodes of blue, white, and red colors, where blue corresponds to the lowest water level and red to the highest water level.

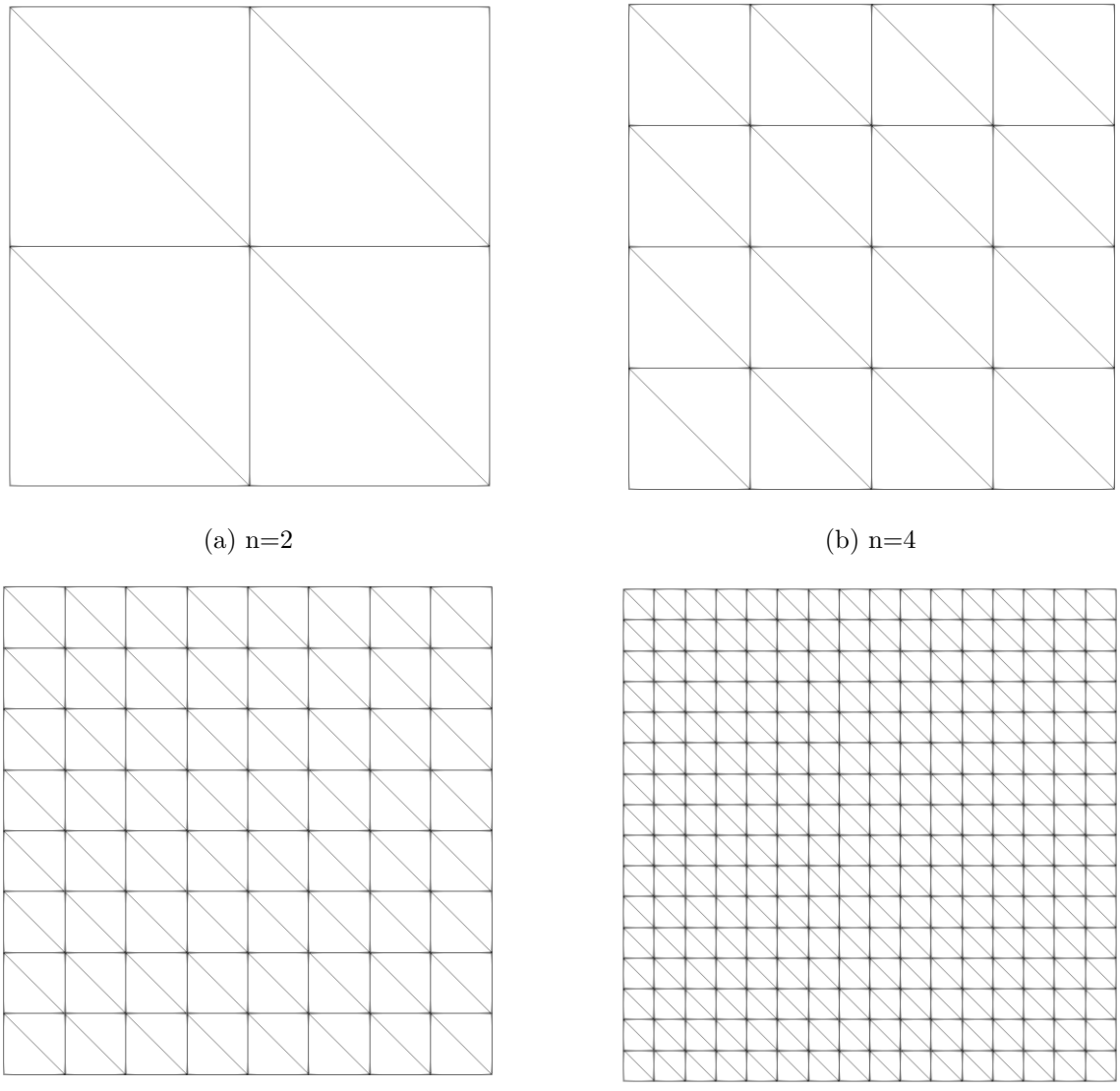
5.1 Synthetic region - square with the Poisson equation

In the first experiment, we want to check whether the implementation is correct by computing the experimental order of convergence (EOC). To compute the EOC we need to know the exact solution of the boundary problem. In this specific case, we decided to solve the Poisson equation

$$\begin{aligned} -\Delta u = &- y^2(1-y)^2\left(-8x(1-x)+2(1-x)^2+2x^2\right) \\ &- x^2(1-x)^2\left(-8y(1-y)+2(1-y)^2+2y^2\right) \end{aligned} \tag{5.1}$$

with the exact solution $u_{exact} = x^2y^2(1-x)^2(1-y)^2$ on a simple domain given as a unit square $\Omega = [0,1] \times [0,1]$. The boundary conditions are derived from the exact solution for the points on the boundary of the domain. Solving the Poisson equation instead of the Laplace equation required small changes in the right-hand side of the system of linear equations.

We apply the derived numerical method to solve this boundary problem with a changing triangular grid, which is getting progressively denser, as shown in Fig. 5.1. In this experiment, we manually set the vertices of the triangular grid to obtain a regular grid, because CGAL triangulation does not guarantee the regularity. The parameter n, representing the number of line segments on each edge of the unit square, is chosen as $n = 2^k$, where $k = 1, 2, 3, \ldots$. Subsequently, we visualize the solution for each case of the triangular grid using VTK files and ParaView software (see Fig. 5.2).



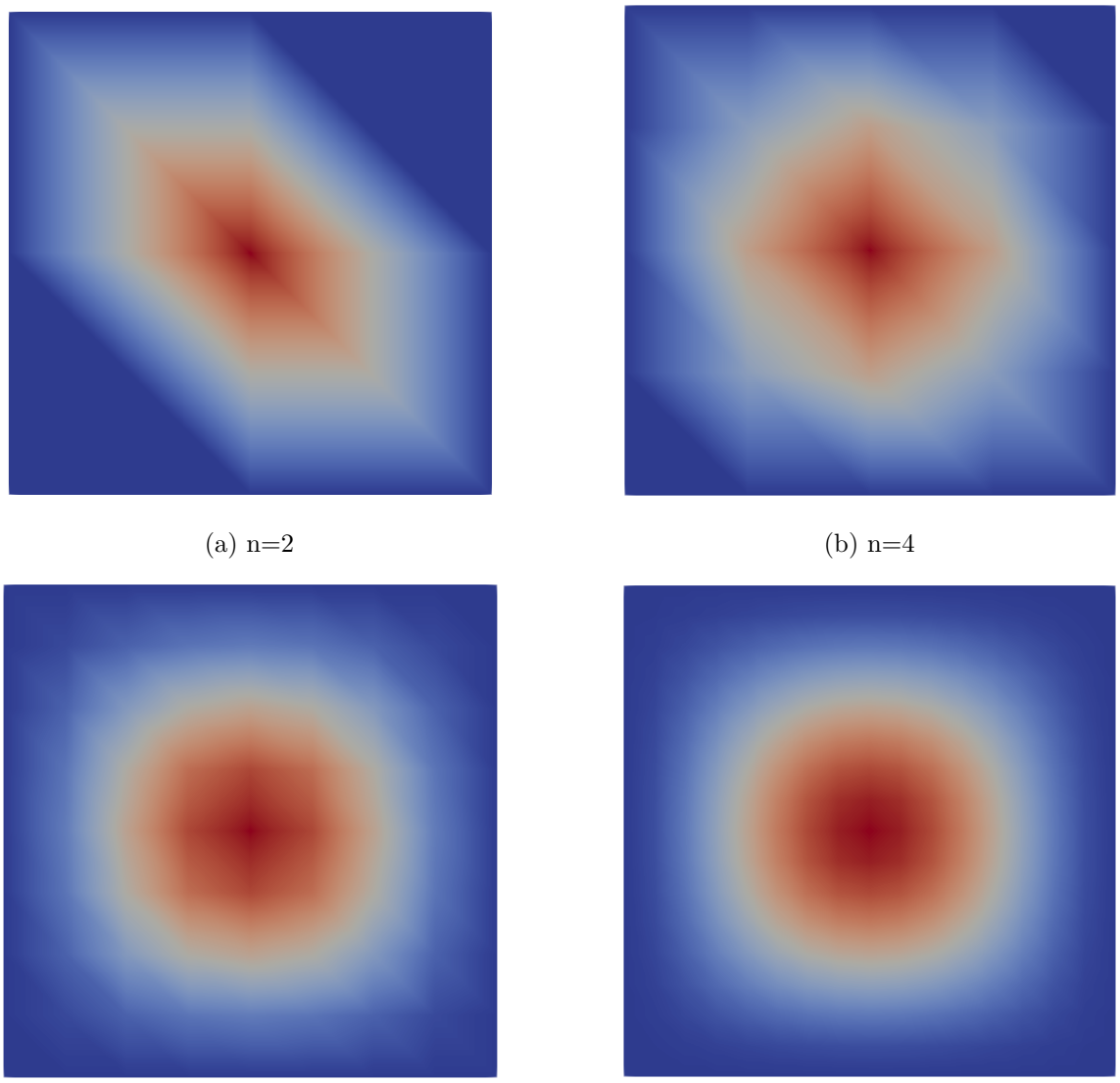
(a) n=2

(b) n=4

(c) n=8

(d) n=16

Figure 5.1: The triangulation of the domain, which is getting progressively denser.



(a) n=2 (b) n=4

(c) n=8 (d) n=16

Figure 5.2: The solution of the Poisson equation on the domain with a changing density of a triangular grid.

For each numerical solution u, we compute the L^2 norm error as

$$error = \sqrt{\left(\frac{1}{n}\right)^2 \sum_{i=0}^{N} \left(u_{exact}(X_i) - u_i\right)^2}. \tag{5.2}$$

The EOC is then computed from the ratio of errors of two consecutive cases as

$$EOC = \log_2 \left(\frac{error_k}{error_{k+1}}\right), \tag{5.3}$$

where $error_k$ corresponds to the case with $n = 2^k$.

n	Error	EOC
2	0.00195312000	
4	0.00045168300	2.112400
8	0.00011233800	2.007460
16	0.00002807700	2.000390
32	0.00000701919	2.000010
64	0.00000175480	1.999998

Table 5.1: Error of the numerical solution and the EOC for progressively denser mesh of the domain.

The values of EOC for our experiment are presented in Tab. 5.1. We computed the error of the numerical solution for six cases with gradually denser triangular grids. As the value of EOC approaches the number 2, it suggests that the numerical method is of the second order of convergence.

5.2 Synthetic region - discretized circle

In the next experiment, we focus on the original boundary problem involving the Laplace equation. The domain is represented by a discretized circle defined by 10 evenly spaced vertices along its boundary (Fig. 5.3). The coordinates of the vertices are initially computed in UTM (Universal Transverse Mercator) coordinates [10] and then converted to GPS coordinates for application within NaturaSat software. We chose the radius of the circle to be $R = 500\,\mathrm{m}$ for detailed visualization of the solution.

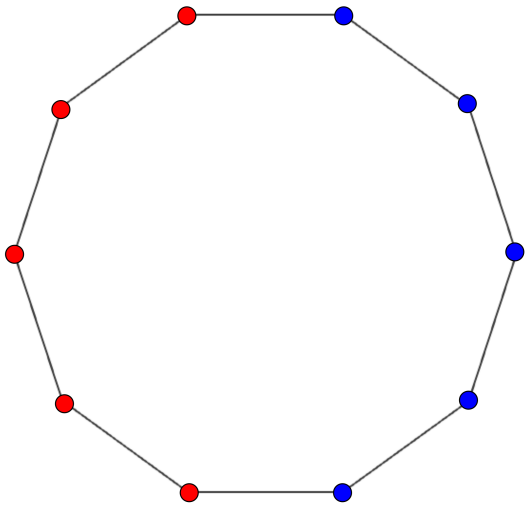
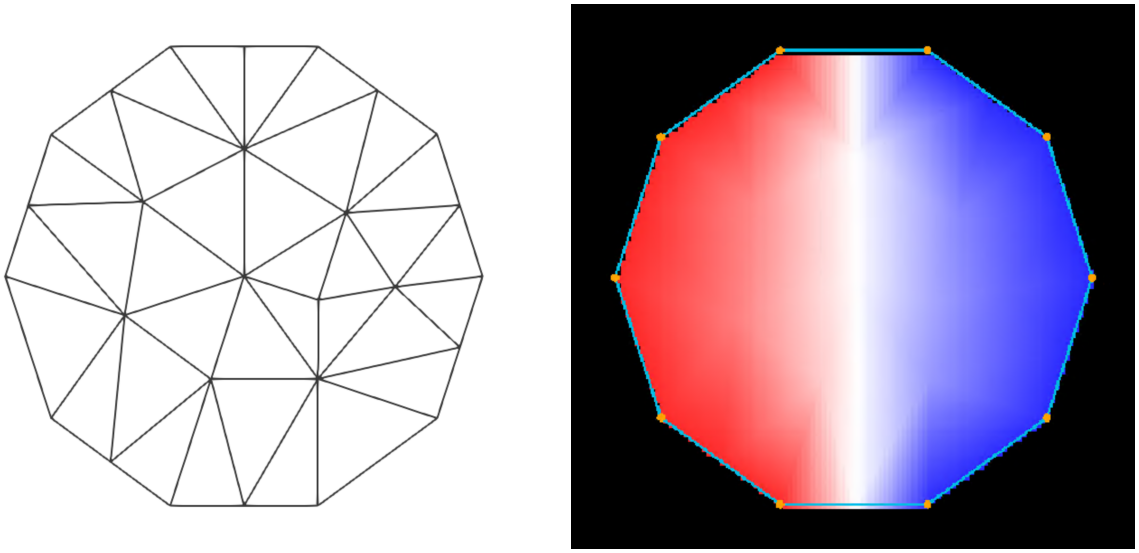


Figure 5.3: The computational domain given as a discretized circle with 10 vertices on the boundary. Red vertices indicate the boundary conditions with a value of 1, while blue vertices indicate the boundary conditions with a value of 0.

All vertices of the circle are given synthetic Dirichlet boundary conditions. Specifically, we manually assign boundary conditions such that the left half of the circle is set to the value of 1, while the right half is set to the value of 0 (see Fig. 5.3). These boundary conditions enable us to validate the correctness of the solution of the Laplace equation.

Initially, we keep the default triangulation criteria. The distance of two neighboring vertices is 309.017 and it serves as the size criterion. Even though this criterion implies the generation of a symmetrical regular grid, the triangulation process via the CGAL library usually results in an irregular grid, as depicted in Fig. 5.4a.

The triangular grid serves as the basis for constructing a co-volume grid which enters into the complementary volume method. The visualization of its solution is presented in Fig. 5.4b, where red denotes the highest water level and blue represents the lowest water level. The water level range in the domain is $[0, 1]$. We observe minor color distortions in certain areas of the domain, which could be reduced by choosing a denser triangular grid.



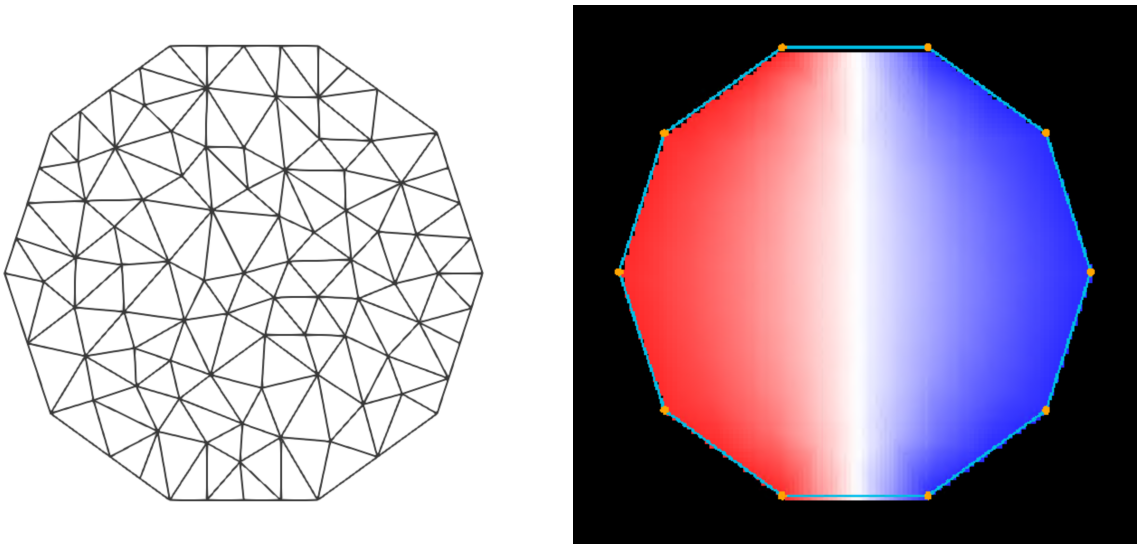
(a) Triangulation of the domain.

(b) The solution representing the water level.

Figure 5.4: Results of the applied implementation on the domain of a discretized circle with the default size criterion for the triangular grid.

The density of the triangular grid is determined by the size criterion used for triangulation. We tried halving the size criterion, resulting in a value of 154.5085. As depicted in Fig. 5.5a, the obtained triangular grid is denser compared to the previous configuration. This grid results in the solution with reduced color distortions (see Fig. 5.5b).

The results align with expectations when solving the Laplace equation: the left half of the circle appears in shades of red, while the right half is depicted in shades of blue, with a gradual transition to white along the vertical line crossing the circle's center.



(a) Triangulation of the domain.

(b) The solution representing the water level.

Figure 5.5: Results of the applied implementation on the domain of a discretized circle with the size criterion set as the half of the default value for the triangular grid.

5.3 Real region

In the following experiments, the computational domain is based on a real region located in the area of CHKO Dunajské luhy [1], near the municipality of Bodíky (see Fig. 5.6). The region is bounded by two distributaries, Bodické rameno, and Bačianske rameno, which naturally provide elevation data for boundary conditions. Furthermore, the southern part of this region falls within the Foráš nature reserve, which is protected as part of the floodplain forests and wetlands by the State Nature Conservancy of the Slovak Republic [27]. This suggests possible occurrences of wetlands, making it particularly suitable for testing the implementation.

To solve the boundary problem and compute the water level inside the region, we experimented with various sets of boundary conditions, and the results are discussed in further sections.

5.3.1 Synthetic boundary conditions

The segmentation polygon for this experiment was obtained using a segmentation tool within NaturaSat software on orthomosaic images [7], which are characterized by high resolution. The polygon consists of 108 unique points on the boundary, connected by straight lines. Due to the proximity of these points, the default size criterion for triangulation is sufficient for generating a suitably dense triangular grid (see Fig. 5.7). (Reminder: the default size criterion is defined as the distance between two neighboring vertices on the boundary).

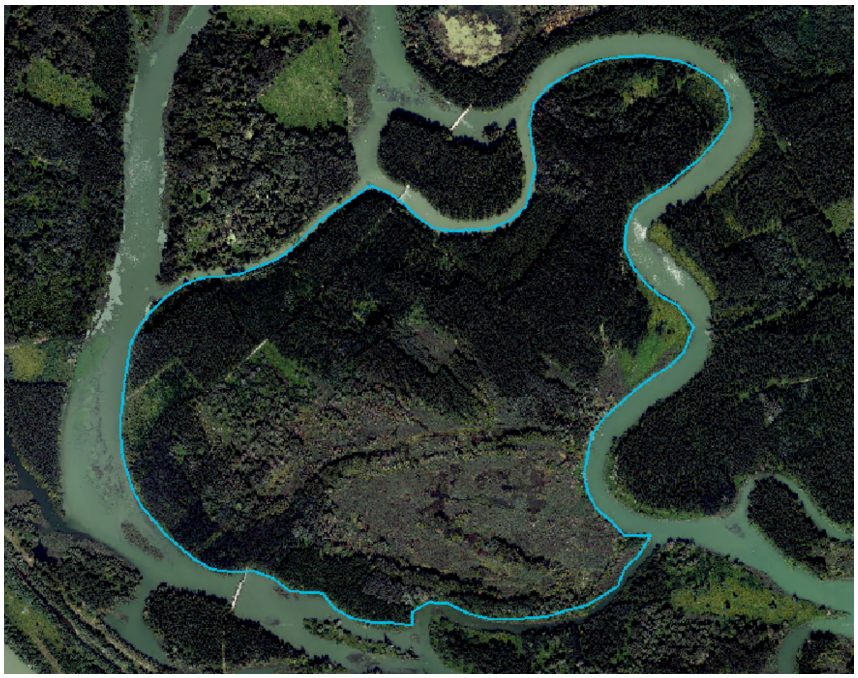


Figure 5.6: The segmentation polygon (cyan polyline) corresponding to the computational domain, based on the real region in CHKO Dunajské Luhy near Bodíky displayed on an orthomosaic image.

In this experiment, we define synthetic boundary conditions for the region, similar to those in the first experiment. Figure 5.8 depicts all vertices of the segmentation polygon, with the highlighted first vertex, along with the polyline orientation. For the first half of the boundary, starting with the yellow vertex, the boundary condition value is set to 1, while for the second half, it is set to 0.

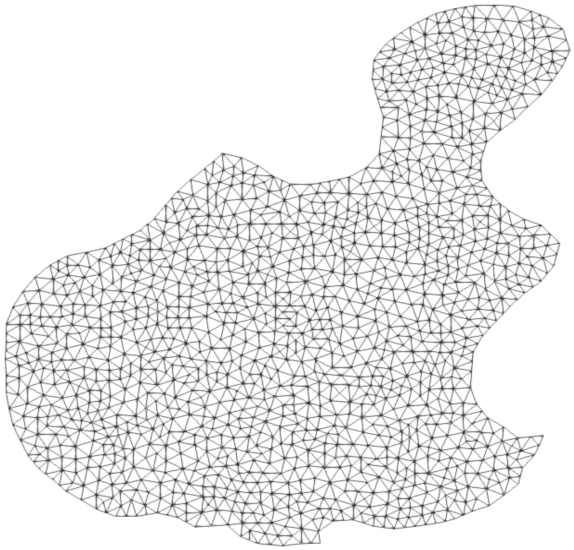


Figure 5.7: Triangulation of the domain with the default size criterion.

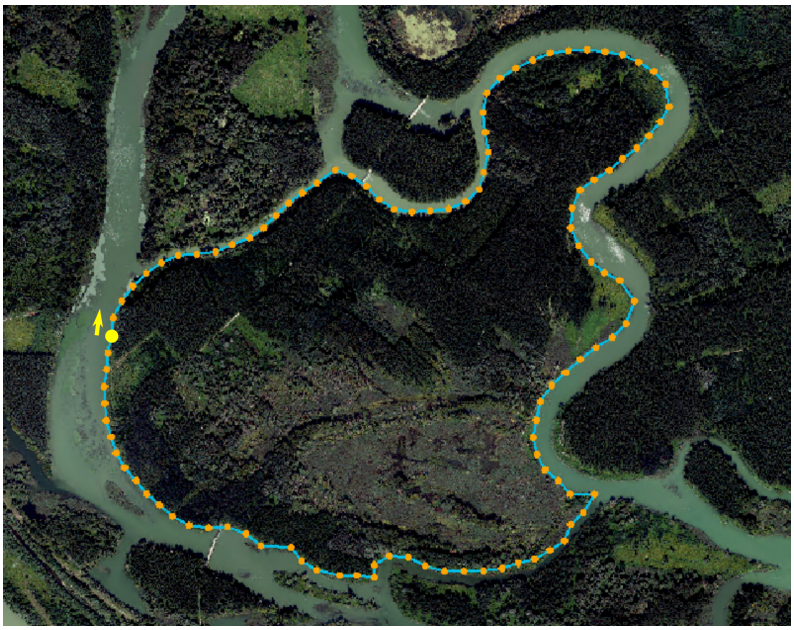


Figure 5.8: The segmentation polygon with all vertices (orange dots). The yellow vertex represents the first vertex and the arrow indicates the polyline orientation.

These boundary conditions lead to the solution depicted in Fig. 5.9. As in the first experiment, the expected outcome is achieved, with the highest water level appearing in the half of the region with a boundary condition value of 1, and the lowest in the half with a boundary condition value of 0.

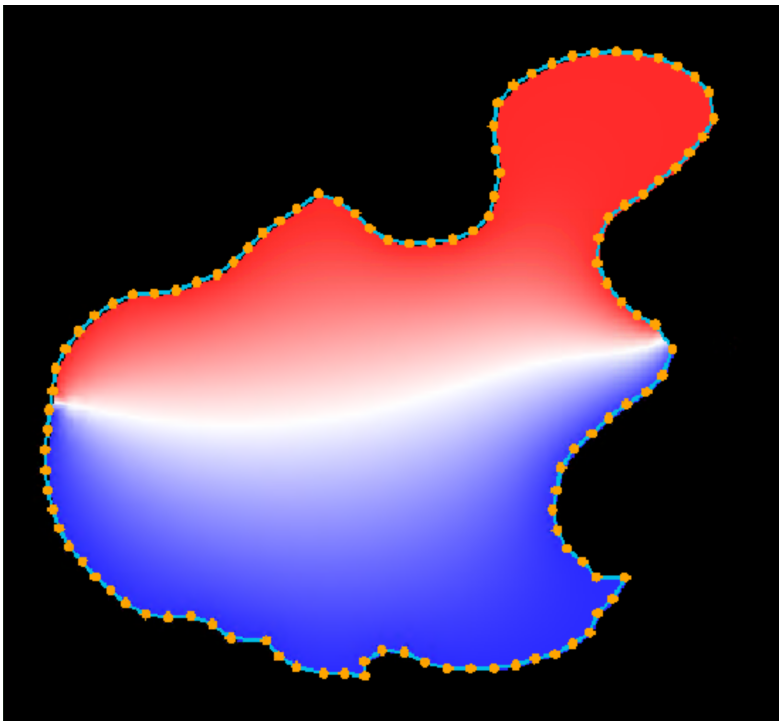


Figure 5.9: The solution of the boundary problem. Red represents a value of 1 and blue corresponds to a value of 0.

5.3.2 Boundary conditions from elevation data

In the next experiment, we explore the same region but with different boundary conditions. This time, we choose more realistic values acquired from the DTM (Digital terrain model) available via Geodetický a kartografický ústav Bratislava (GKÚ) [6], as shown in Fig. 5.10. To obtain more accurate water levels from the DTM we need to adjust the segmentation polygon, so its vertices lie in the middle of distributaries (see Fig. 5.11). The polygon is manually defined by linear segments, which contain a new vertex every 20 m. The boundary conditions are set only for the vertices at the linear segments' ends and their range is $[117.161, 117.628]$ meters.

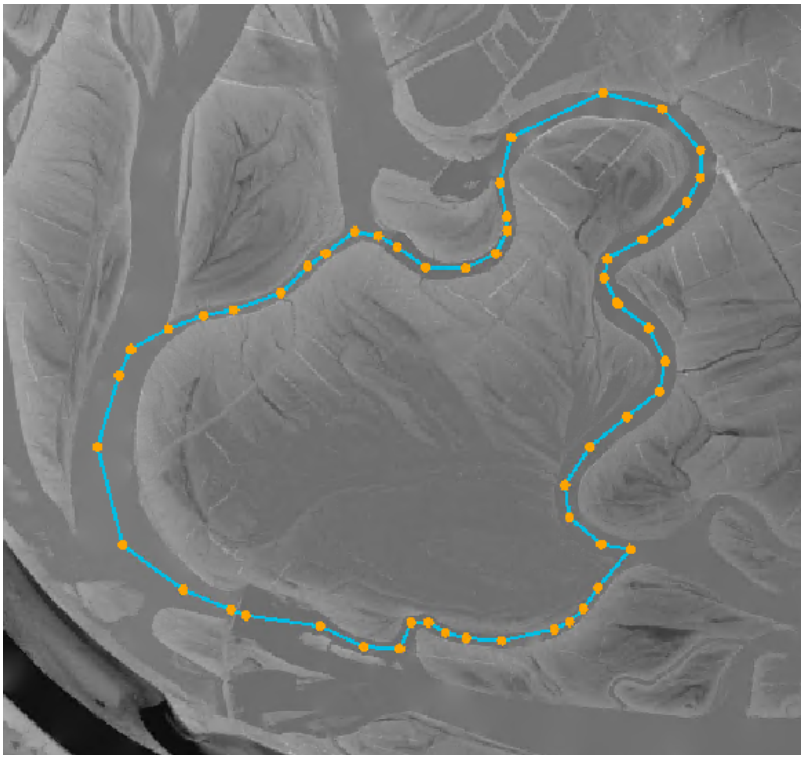


Figure 5.10: The digital terrain model data in the area of the region of interest.

Another change compared to the previous experiment involves the frequency of given boundary conditions. As previously mentioned, they are not defined at each vertex on the boundary. It deals with cases when the user has limited measurements of the water level (only in a few vertices on the boundary). However, the complementary volume method requires boundary conditions in all vertices on the boundary, so it is a necessity to define the values for other vertices. This problem is solved by interpolation of given values for vertices without defined boundary conditions.

Simultaneously, as the segmentation polygon differs from the one in the previous experiment, we also get a different triangulation. The triangular grid is much denser, because this region is defined by more vertices on the boundary (see Fig. 5.12).

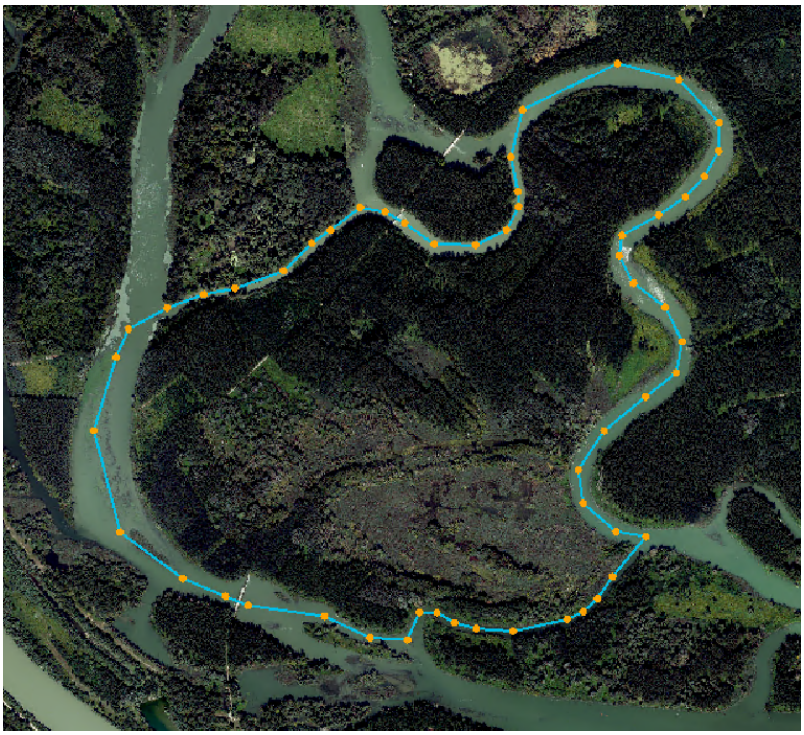


Figure 5.11: The segmentation polygon with the vertices, in which are defined boundary conditions (orange dots). The vertices are located in the middle of the distributaries.

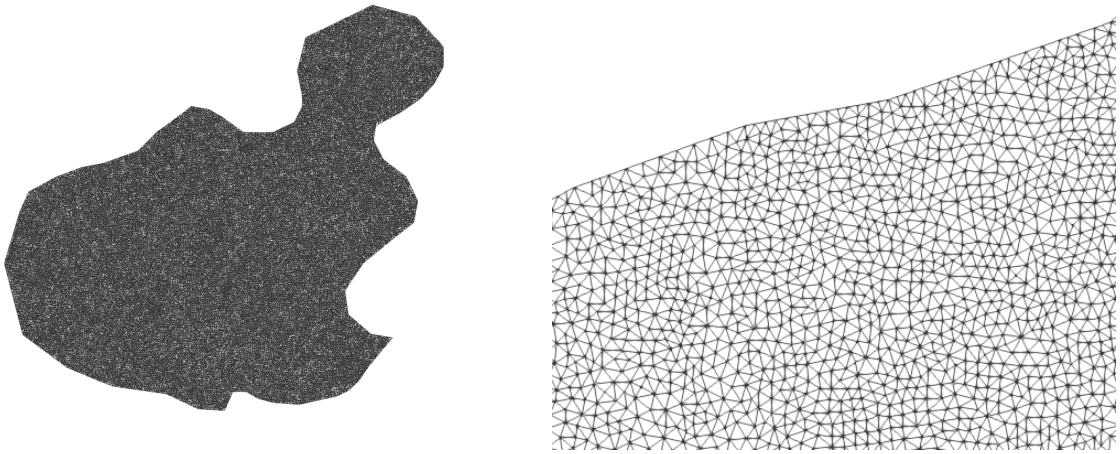


Figure 5.12: Triangulation of the domain with default size criterion and its detail (on the right). The grid is denser because the segmentation polygon contains more vertices that are closer to each other.

The solution is depicted in Fig. 5.13. We see that the highest water level occurs in the west of the region and the lowest in the east. This result is expected because there are two weirs located within the distributaries which lower the water level by a few tens of centimeters.

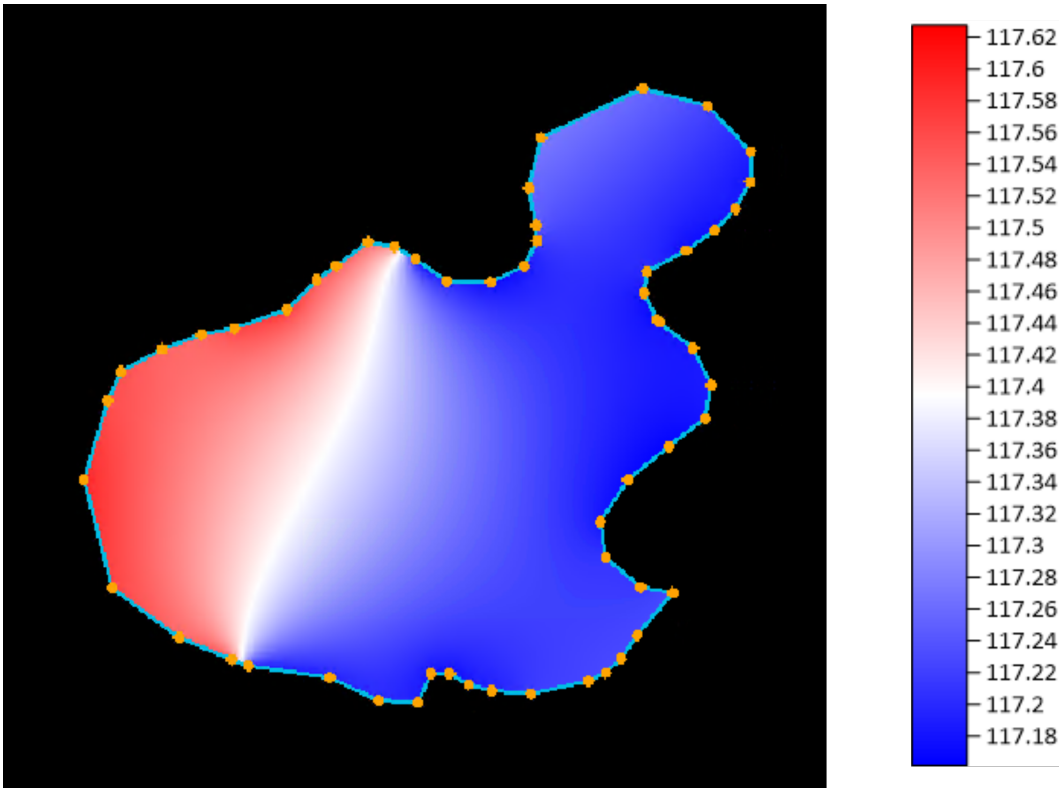


Figure 5.13: The solution of the boundary problem. Red represents a maximal value of 117.628 m and blue corresponds to a minimal value of 117.161 m.

6 Comparison of the solution of the Laplace equation with the DTM

The information about the water level inside the region of interest is not sufficient for assessing wetland conditions alone. The saturation of a region with water depends on the elevation of the ground. Therefore, we are interested in the relationship between the water surface and the ground. This way, we can identify areas where water is either above or close to the terrain. These areas may indicate potential wetland occurrences.

To compare the water level with the ground elevation, we use the already mentioned Digital terrain model (DTM) obtained from GKÚ. The DTM is derived from airborne laser scanning, which generates a point cloud representing the Earth's surface. The DTM is created by interpolation from the classified point cloud. We specifically use the data from the second cycle of the project, which is currently in progress, with five regions already completed (see Fig. 6.1) [6]. One of them is the Danube basin, which we are particularly interested in. We choose a part of the data corresponding to the real region we are working with.

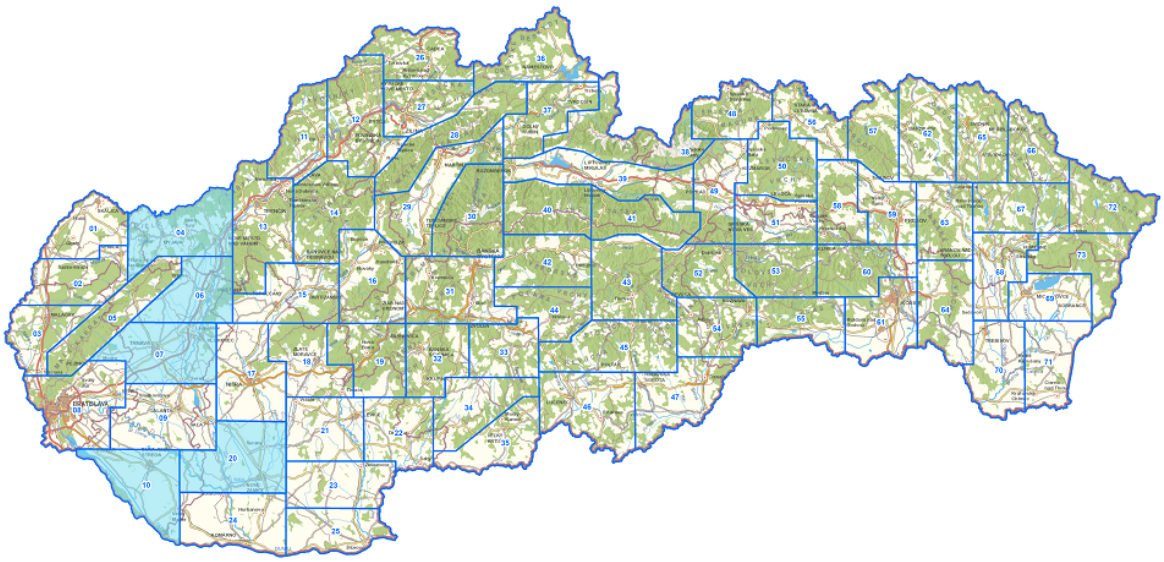


Figure 6.1: Slovakia is divided into 73 regions which are gradually processed in the second cycle of the DTM acquisition. The regions highlighted in blue color are already available.

The DTM contains the elevation of the ground. In the case of water bodies, the data represents the elevation of the water surface at the time of scanning, interpolated from the points on the water surface and the points on the shore. This means that the elevation of the water surface in water bodies can be affected by errors caused by interpolation from the points on the shore. When creating the segmentation polygon for the real region presented in the previous chapter, we attempted to set the polygon's vertices in a way that would minimize the influence of such errors. Additionally, we need to keep in mind that the DTM data for the water surface are influenced by the time of scanning. In the case of our region of interest, the airborne laser scanning was done in the winter, so the water level is lower than in other seasons.

The solution of the Laplace equation is computed in a resolution $0.5\,\mathrm{m} \times 0.5\,\mathrm{m}$, which is identical to the resolution of the DTM. The comparison is then done in all pixels within the computation domain as:

$$comparison = DTM - solution. \tag{6.1}$$

To visualize the comparison in color, we use a 5-node linear color gradient. We choose more colors to better distinguish the areas of possible wetland occurrence:

- blue - areas where the water surface is significantly above the ground,
- green - areas where the water surface is slightly above the ground,
- white - areas where the water level matches the ground elevation,
- yellow - areas where the water surface is slightly below the ground,
- red - areas where the water surface is significantly below the ground.

We also enable the user to set a parameter to lower the elevation of the ground. This feature can be used to highlight areas with a water level slightly below the ground because it can indicate the saturation of soil by water and also can eliminate the errors of the DTM data (the accuracy of the DTM is $\pm 5\,\mathrm{cm}$). It can also help with simulated scenarios of increased water levels in surrounding distributaries caused by intense precipitation or water regulation.

6.1 Results for the real region

The comparison of the solution with the DTM is performed on the same region in CHKO Dunajské Luhy as in real scenarios of the previous chapter, and the result is depicted in Fig. 6.2. The range of the comparison values is $[-1.508, 2.782]$ meters. Negative values represent situations where the water surface is above the ground, while

positive values indicate situations where the water surface is below the ground. We are particularly interested in the negative values, which are visualized in shades ranging from blue, through green, to white.

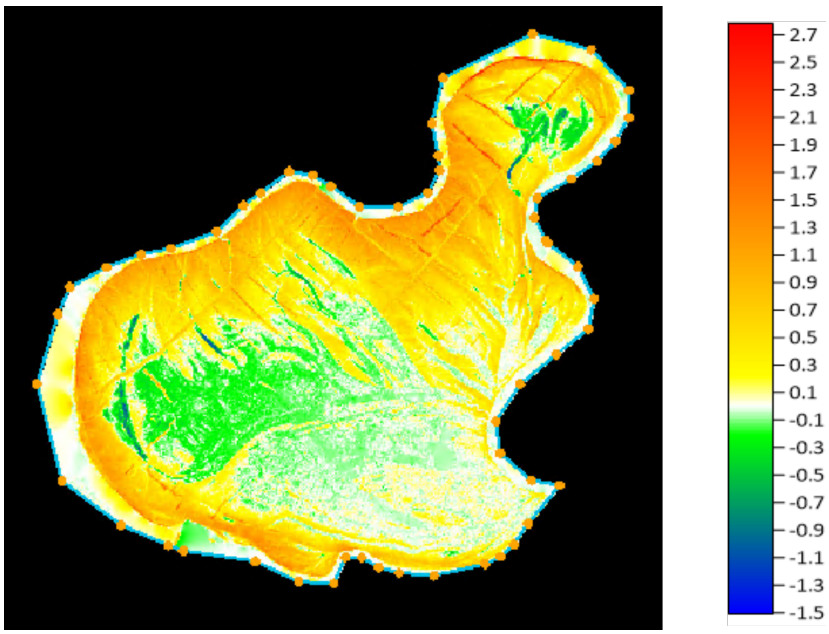


Figure 6.2: The comparison of the solution of the Laplace equation with the DTM for the region of interest.

Both maximum and minimum values of the comparison are observed in the northern part of the region, as shown in Fig. 6.3. There is an area visualized mainly in green, so it suggests the possible occurrence of a shallow lake or swamp. The curved blue segment indicates the possible occurrence of a stream. However, due to vegetation cover, we are not able to distinguish these possible water bodies from the orthophoto image and check the correctness of our assumptions. It would require a terrain inspection.

In the western part of the region, we observe another case of negative values near the minimum (see Fig. 6.4). There seems to be a continuous land depression, which should be flooded by water. However, in the orthophoto images, we are not able to identify the land depression or water presence, even though there are areas with sparse vegetation. This may be again checked by the terrain inspection.

The region includes the Foráš nature reserve in the southeast, where the orthophoto image suggests the presence of a lake or multiple smaller lakes. As the DTM for water bodies provides elevation data for their surfaces, we expect values close to zero. In Fig. 6.5, we see that the area is mostly displayed in light green, white, and light yellow. The corresponding values are close to zero, aligning with our expectations. One of the reasons why the area is not fully white is the accuracy of the DTM.

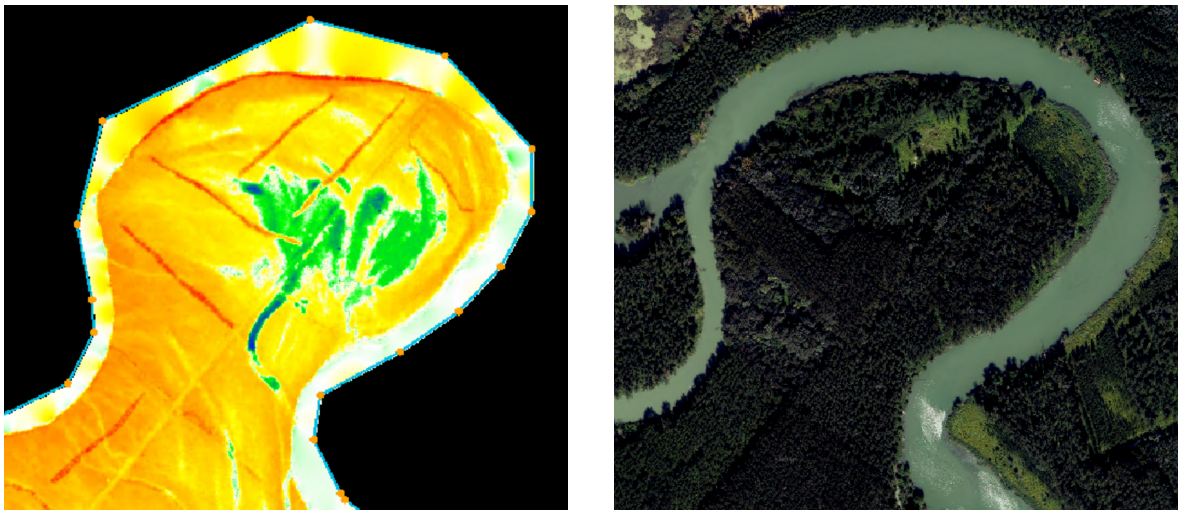


Figure 6.3: Detail of the comparison and the orthophoto image for the northern part of the region. There are located both minimum (blue) and maximum (red) values of the comparison.

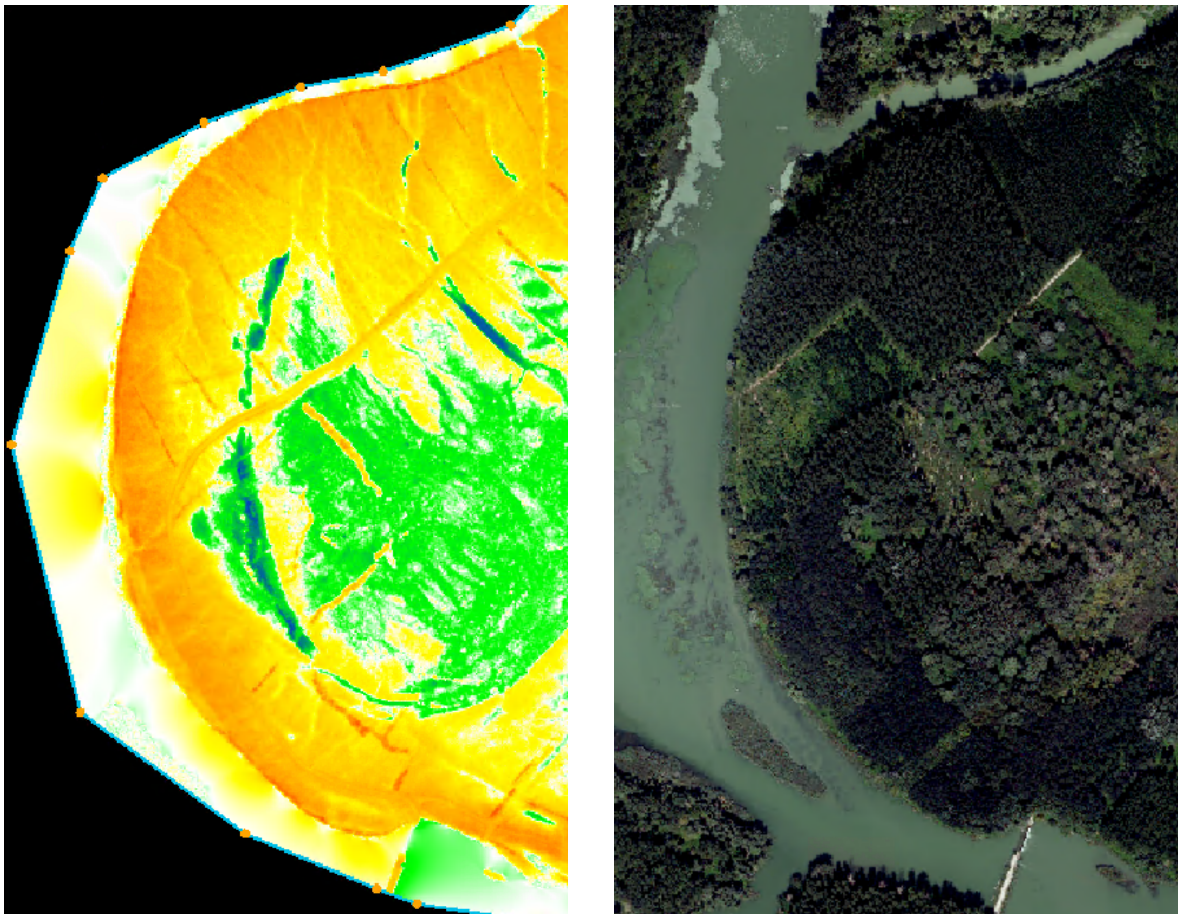


Figure 6.4: Detail of the comparison and the orthophoto image for the western part of the region. There are values close to the minimum (blue) suggesting a flooded land depression.

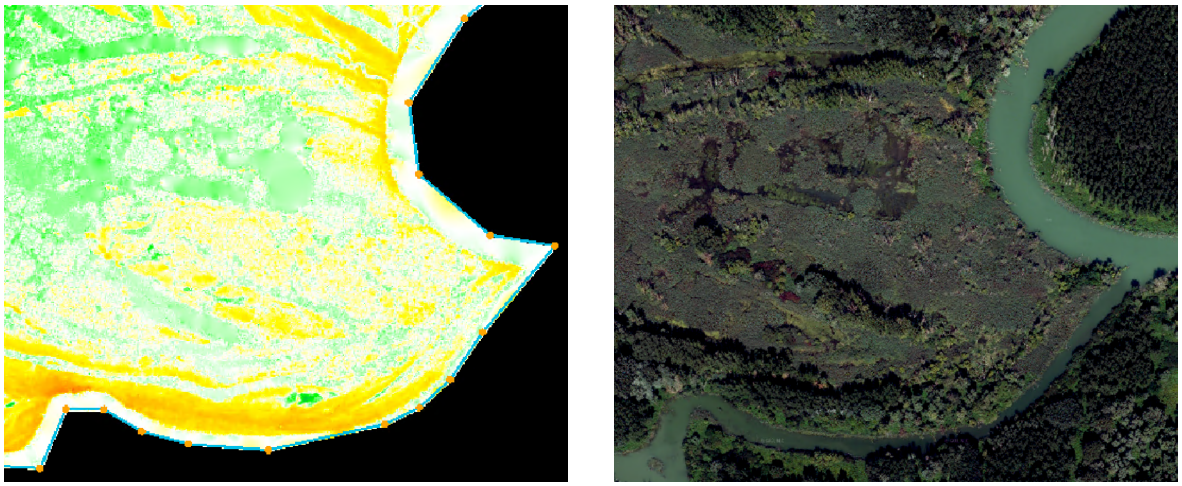
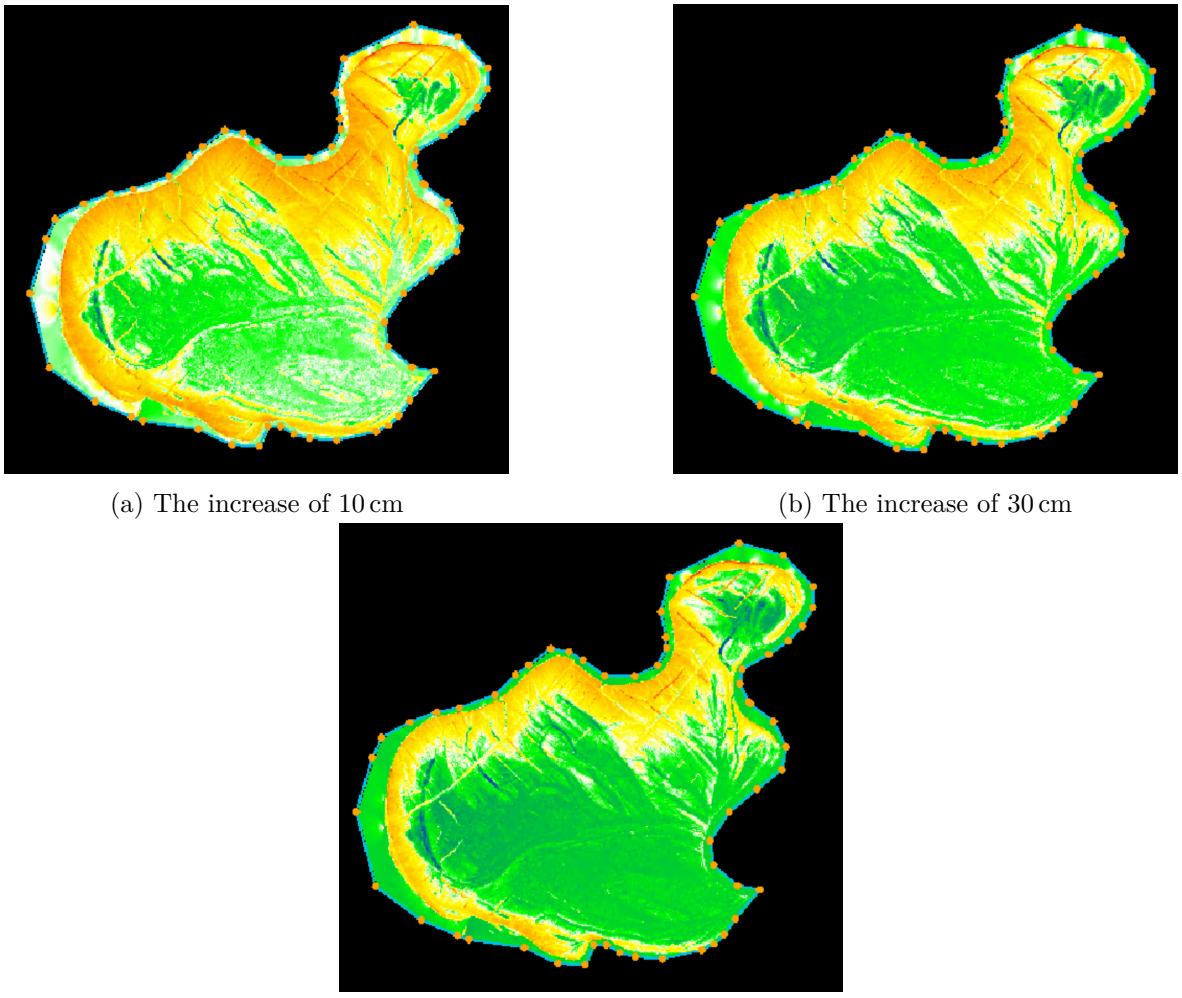


Figure 6.5: Detail of the comparison and the orthophoto image for the southeast part of the region. In the orthophoto image, we observe a lake and in the comparison, this area is displayed in light green, white, and light yellow.



(a) The increase of 10 cm

(b) The increase of 30 cm

(c) The increase of 50 cm

Figure 6.6: The comparison of the solution of the Laplace equation in the case of increased water levels in surrounding distributaries.

We also simulated scenarios where water levels in surrounding distributaries rise to identify areas theoretically flooded by water. Flow of water in the distributaries can be regulated leading to changes in the water level. We are interested in how to raise the water level to keep wetlands in a healthy state or to create conditions for the formation of wetlands. In Fig. 6.6, we can see the effect of water level rises of 10, 30, and 50 cm. We observe that in such cases the majority of the region should be flooded by water.

The areas depicted in blue and green in all results of this section suggest potential wetland occurrences, assuming the land is a homogeneous environment. However, this is generally not true and we may at least suppose that these areas represent the land depressions, which in case of increased water levels in the surrounding distributaries could become flooded, potentially transforming into wetlands.

7 Conclusions

The thesis dealt with basic hydrological modeling in an effort to support ecologists, botanists, and other specialists in the analysis and restoration of wetlands. It focused on determining one of the wetlands' characteristics, the water level, in a specified computational domain by solving the Laplace equation with given Dirichlet boundary conditions.

The first step involved creating a triangular grid over the computational domain, for which we explored different libraries for triangulation. Then we derived the complementary volume method scheme for solving the Laplace equation. Finally, we tried two types of visualization of the solution - using VTK files and interpolation onto a regular grid. This method was implemented into the NaturaSat software.

We proceeded to test the implementation through various experiments, from synthetic scenarios to real-world applications. Initially, we conducted synthetic experiments to verify the correctness of the solution of the Laplace equation and compute its EOC. The proposed method worked successfully with the artificial data, both for defining the region and specifying boundary conditions. Additionally, the method proved effective also for other experiments, where we focused on a real segmented region with both synthetic and real boundary conditions derived from the digital terrain model. In the result obtained with real boundary conditions, we successfully identified the jump in water level caused by the presence of weirs within the distributaries.

Lastly, we analyzed the relationship between the water surface and the ground by comparing the solution of the Laplace equation with the digital terrain model. Such analysis is necessary to comprehensively assess wetland conditions because the water level alone is not sufficient. It revealed two particularly promising areas of potential wetland occurrence within the region of interest in the Danube basin. However, they were not identifiable on the orthophoto images, so further terrain inspection is required. Concurrently, we identified existing lakes within the Foráš nature reserve, which are characterized by comparison values close to zero.

This work can be expanded in various ways. One of them is to verify the accuracy of the Laplace equation in describing the behavior of groundwater through comparison with real measurements of water level within the region. Additionally, as our colleagues have done field measurements of the water level around the region of in-

terest, it would be interesting to apply the method to this real data. Furthermore, we can continue by further refinement of the hydrological module in the NaturaSat software. This includes the derivation of a numerical scheme for solving the minimal surface equation, potentially leading to more accurate results, as well as calculating other wetland characteristics.

Resumé

Mokrade sú trvalo alebo sezónne zaplavené oblasti, ktoré sa vyznačujú výraznou biodiverzitou a plnia viacero významných funkcií, čím sú pre prírodu a ľudstvo veľmi dôležité. Avšak mokrade sa veľmi rýchlym tempom vytrácajú. Dobrý stav mokradí je možné zabezpečiť ich dlhodobým monitorovaním, analýzou ich vlastností a v prípade potreby aplikovaním revitalizačných procesov.

Cieľom tejto diplomovej práce bolo pomôcť odborníkom pri analýze a revitalizácii mokradí formou hydrologického modelovania založeného na numerických metódach. Základné hydrologické modelovanie predstavuje rozšírenie environmentálneho softvéru NaturaSat [14], ktorý slúži na identifikáciu a monitorovanie biotopov Natura2000. V rámci neho sme sa zamerali na výšku hladiny podzemnej vody ako jednu z vlastností mokradí, ktorá sa dá použiť na posúdenie ich stavu. Správanie podzemnej vody sa dá popísať rovnicou minimálnej plochy, ktorá sa pre homogénne prostredie a malé hodnoty gradientov riešenia aproximuje Laplaceovou rovnicou. Základnou myšlienkou teda bolo vypočítať výšku hladiny vo zvolenej výpočtovej oblasti pomocou riešenia Laplaceovej rovnice metódou komplementárnych objemov so zadanými Dirichletovými okrajovými podmienkami na hranici oblasti.

Prvým krokom je diskretizovať výpočtovú oblasť. Výpočtová oblasť je daná vo forme segmentačného polygónu, ktorý je možné vytvoriť pomocou semi-automatickej alebo automatickej segmentácie v softvéri NaturaSat [12, 13]. Na oblasti vytvoríme nepravidelnú trojuholníkovú sieť pomocou triangulácie cez knižnicu CGAL [21] (pozri kapitolu 2). Používame Delaunayovú trianguláciu, ktorá sa vyznačuje ostrými trojuholníkmi podobnej veľkosti a tvaru, vďaka čomu je zabezpečená efektivita numerických výpočtov a jednoduchá konštrukcia komplementárnych objemov. Sieť komplementárnych objemov vybudujeme na trojuholníkovej sieti.

Následne je potrebné odvodiť numerickú schému metódy komplementárnych objemov pre Laplaceovu rovnicu, ktorú popisujeme v kapitole 3. Pri odvádzaní využívame techniky metódy konečných objemov a metódy konečných prvkov, keďže metóda komplementárnych objemov je ich kombináciou. Výsledná numerická schéma pre konkrétny komplementárny objem vedie na systém lineárnych rovníc. Po dosadení Dirichletových okrajových podmienok sme schopní tento systém riešiť. Na riešenie sme použili metódu BiCGSTAB [23] z knižnice EIGEN [8].

Riešenie Laplaceovej rovnice predstavuje výšku hladiny podzemnej vody v jednotlivých komplementárnych objemoch a jednotlivých vrcholoch trojuholníkovej siete. Pre lepšie porozumenie výsledkov je vhodné ich vizualizovať. Na vizualizáciu sme použili dva prístupy (pozri kapitolu 4). Prvý prístup ukladá geometriu a riešenie Laplaceovej rovnice do VTK súboru [24], ktorý je možné si externe otvoriť napr. v softvéri ParaView [16]. Druhý prístup slúži na zobrazenie vizualizácie priamo v softvéri NaturaSat a je založený na interpolácii riešenia do rovnomernej mriežky. Takýto typ vizualizácie vedie k šedotónovému obrázku, ale zavedením lineárneho farebného prechodu je možné riešenie vizualizovať aj farebne.

Takto navrhnutú metódu sme implementovali do softvéru NaturaSat v programovacom jazyku C++. Otestovali sme ju najprv na umelej oblasti pre prípad Poissonovej rovnice, ktorej riešenie poznáme, aby sme mohli určiť experimentálny rád presnosti metódy. Podarilo sa nám ukázať, že numerická metóda je druhého rádu presnosti. Následne sme ju ďalej aplikovali aj na reálnej oblasti nachádzajúcej sa v CHKO Dunajské Luhy, ktorá je ohraničená riečnymi ramenami Dunaja. V experimente sme použili umelé aj reálne okrajové podmienky. Reálne Dirichletove okrajové podmienky sme určili z digitálneho modelu reliéfu (DMR), ktorý poskytuje Úrad geodézie, kartografie a katastra SR [6]. Experiment dopadol úspešne, keďže sme z výsledkov dokázali identifikovať dve miesta skoku výšky vodnej hladiny, ktorý je spôsobený prítomnosťou hrádzí v riečnych ramenách, a teda spôsobuje pokles hladiny v desiatkach centimetrov. Všetky experimenty sú detailne popísané v kapitole 5.

Samotná výška hladiny podzemnej vody však nie je postačujúca pre posúdenie stavu mokradí. Je potrebné ju porovnať s výškou terénu. Tomuto sme sa venovali v poslednej časti diplomovej práce (pozri kapitolu 6), v ktorej identifikujeme vzťah medzi vodnou hladinou a terénom. Výsledok Laplaceovej rovnice pre reálnu oblasť sme porovnali s digitálnym modelom reliéfu a na základe toho sme identifikovali miesta, ktoré predstavujú možný výskyt mokrade. Pri porovnávaní s ortofoto snímkami sme ich však nevedeli priamo potvrdiť, preto je potrebná terénna inšpekcia týchto miest. Súčasne sme sa sústredili aj na jazero, ktoré sa nachádza v juhovýchodnej časti oblasti a určili miesto jeho výskytu vďaka hodnotám porovnania blízkym nule.

V našej práci sa dá ďalej pokračovať verifikáciou presnosti Laplaceovej rovnice pri popisovaní správania podzemnej vody. Môžeme to zhodnotiť pomocou porovnania riešenia a reálnych nameraných hodnôt výšky hladiny vo vnútri výpočtovej oblasti. Ďalším posunom v tejto téme by mohla byť aplikácia metódy na reálne merania výšky hladiny podzemnej vody v teréne, ktoré robili naši kolegovia z katedry. Na takto reálnych dátach by mohli výsledky lepšie zodpovedať realite. Navyše môžeme pokračovať aj v rozširovaní hydrologického modelovania v softvéri NaturaSat, napr. o riešenie rovnice minimálnej plochy, či o výpočet ďalších vlastností mokradí, ktoré by mohli byť použité na posúdenie ich stavu.

Bibliography

1. CHRÁNENÁ KRAJINNÁ OBLASŤ DUNAJSKÉ LUHY. [N.d.]. Available also from: https://chkodunajskeluhy.sopsr.sk/. Online; accessed March 2024.
2. EYMARD, R.; GALLOUËT, T.; HERBIN, R. Finite volume methods. In: *Solution of Equation in Rn (Part 3), Techniques of Scientific Computing (Part 3)*. Elsevier, 2000, vol. 7, pp. 713–1018. Handbook of Numerical Analysis. ISSN 1570-8659. Available from DOI: https://doi.org/10.1016/S1570-8659(00)07005-8.
3. FLUET-CHOUINARD, E.; STOCKER, B.; ZHANG, Z., et al. Extensive global wetland loss over the past three centuries. *Nature*. 2023, vol. 614, pp. 281–286. Available from DOI: 10.1038/s41586-022-05572-6.
4. GEOM SOFTWARE. *Fade2D Documentation*. 2024. Available also from: https://www.geom.at/fade2d/html/index.html.
5. GIEZEMAN, G.-J.; WESSELINK, W. 2D Polygons. In: *CGAL User and Reference Manual*. 5.6.1. CGAL Editorial Board, 2024. Available also from: https://doc.cgal.org/5.6.1/Manual/packages.html#PkgPolygon2.
6. GKÚ BRATISLAVA. *Letecké laserové skenovanie*. [N.d.]. Available also from: https://www.geoportal.sk/sk/zbgis/lls/.
7. GKÚ BRATISLAVA, NLC. *Ortofotomozaiky SR*. [N.d.]. Available also from: https://www.geoportal.sk/sk/zbgis/ortofotomozaika/2-cyklus/.
8. GUENNEBAUD, G.; JACOB, B., et al. *Eigen: A C++ template library for linear algebra*. 2021. Available also from: http://eigen.tuxfamily.org.
9. KOLLÁR, M. *Solving partial differential equations on surfaces with applications to geodetic data analysis*. 2018. PhD thesis. Slovak University of Technology in Bratislava.
10. LANGLEY, R. B. The UTM grid system. *GPS world*. 1998, vol. 9, no. 2, pp. 46–50.
11. LEICA GEOSYSTEMS. *CDT: C++ library for constrained Delaunay triangulation*. 2019. Available also from: https://artem-ogre.github.io/CDT/index.html.

12. MIKULA, K.; URBÁN, J.; KOLLÁR, M.; AMBROZ, M.; JAROLÍMEK, I.; SIBIK, J.; ŠIBÍKOVÁ, M. An automated segmentation of NATURA 2000 habitats from Sentinel-2 optical data. *Discrete & Continuous Dynamical Systems - S*. 2021, vol. 14, pp. 1017–1032. Available from DOI: 10.3934/dcdss.2020348.

13. MIKULA, K.; URBÁN, J.; KOLLÁR, M.; AMBROZ, M.; JAROLÍMEK, I.; SIBIK, J.; ŠIBÍKOVÁ, M. Semi-automatic segmentation of NATURA 2000 habitats in Sentinel-2 satellite images by evolving open curves. *Discrete & Continuous Dynamical Systems - S*. 2021, vol. 14, pp. 1033–1046. Available from DOI: 10.3934/dcdss.2020231.

14. MIKULA, K.; ŠIBÍKOVÁ, M.; AMBROZ, M.; KOLLÁR, M.; OŽVAT, A. A.; URBÁN, J.; JAROLÍMEK, I.; ŠIBÍK, J. NaturaSat—A Software Tool for Identification, Monitoring and Evaluation of Habitats by Remote Sensing Techniques. *Remote Sensing*. 2021, vol. 13, no. 17. ISSN 2072-4292. Available from DOI: 10.3390/rs13173381.

15. NC WETLANDS. *Why Our Wetlands Matter: Functions and Benefits of NC's Wetlands* [https://www.ncwetlands.org/learn/functions-benefits/]. [N.d.]. Online; accessed February 2024.

16. PARAVIEW. 2024. Available also from: https://www.paraview.org/.

17. PION, S.; YVINEC, M. 2D Triangulation Data Structure. In: *CGAL User and Reference Manual*. 5.6.1. CGAL Editorial Board, 2024. Available also from: https://doc.cgal.org/5.6.1/Manual/packages.html#PkgTDS2.

18. RINEAU, L. 2D Conforming Triangulations and Meshes. In: *CGAL User and Reference Manual*. 5.6.1. CGAL Editorial Board, 2024. Available also from: https://doc.cgal.org/5.6.1/Manual/packages.html#PkgMesh2.

19. SCHROEDER, W.; MARTIN, K.; LORENSEN, B. *The Visualization Toolkit (4th ed.)* Kitware, 2006. ISBN 978-1-930934-19-1.

20. SHEWCHUK, J. R. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In: LIN, M. C.; MANOCHA, D. (eds.). *Applied Computational Geometry Towards Geometric Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 203–222. ISBN 978-3-540-70680-9.

21. THE CGAL PROJECT. *CGAL User and Reference Manual*. 5.6.1. CGAL Editorial Board, 2024. Available also from: https://doc.cgal.org/5.6.1/Manual/packages.html.

22. THE EUROPEAN SPACE AGENCY (ESA). *Space key to wetland conservation* [https://www.esa.int/Applications/Observing_the_Earth/Space_key_to_wetland_conservation]. [N.d.]. Online; accessed February 2024.

23. VAN DER VORST, H. A. Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems. *SIAM Journal on Scientific and Statistical Computing*. 1992, vol. 13, no. 2, pp. 631–644. Available from DOI: 10.1137/0913035.

24. VTK DOCUMENTATION. *VTK file formats*. [N.d.]. Available also from: https://docs.vtk.org/en/latest/design_documents/VTKFileFormats.html. Online; accessed February 2024.

25. WEISSTEIN, E. W. *Heron's Formula.* [N.d.]. Available also from: https://mathworld.wolfram.com/HeronsFormula.html. From MathWorld–A Wolfram Web Resource.

26. YVINEC, M. 2D Triangulations. In: *CGAL User and Reference Manual*. 5.6.1. CGAL Editorial Board, 2024. Available also from: https://doc.cgal.org/5.6.1/Manual/packages.html#PkgTriangulation2.

27. ŠTÁTNA OCHRANA PRÍRODY SLOVENSKEJ REPUBLIKY. *Foráš*. [N.d.]. Available also from: https://data.sopsr.sk/chranene-objekty/chranene-uzemia/detail/1145. Online; accessed March 2024.