

Arnold Gabriel Hanser, BSc

# Shape reconstruction of particles given SAXS measurement data

## **MASTER'S THESIS**

to achieve the university degree of  
Diplom-Ingenieur  
Master's degree programme: Mathematics

submitted to

**Graz University of Technology**

## **Supervisor**

Ao. Univ. Prof. Dr. Wolfgang Ring,  
Department of Mathematics and Scientific Computing,  
University of Graz

Graz, April, 2024



# Contents

<b>1</b>	<b>The problem and important quantities</b>	<b>6</b>
1.1	SAXS . . . . .	6
1.2	Important literature about SAXS . . . . .	7
1.3	The main quantities $I_s(q)$ and $I_{s,meas}(q)$ . . . . .	8
1.4	A more advantageous representation of $I_s(q)$ . . . . .	10
1.5	The shape derivative $dI_s(q; \Omega, V)$ of $I_s(q)$ . . . . .	13
1.6	Error measure $J(\Omega)$ between $I_s(q)$ and $I_{s,meas}(q)$ and its shape derivative . . . . .	22
<b>2</b>	<b>Algorithms to evaluate <math>I_s(q)</math> and its directional derivatives</b>	<b>24</b>
2.1	Quantities of interest . . . . .	24
2.2	Finding the boundary of the region . . . . .	25
2.2.1	Structure of the variable "boundary" . . . . .	27
2.2.2	Location of $\Omega$ . . . . .	28
2.2.3	Find every boundary point of $\Omega$ . . . . .	29
2.2.4	Post calculation of the boundary . . . . .	31
2.3	Calculate the autoconvolution $\gamma(\mathbf{z})$ . . . . .	33
2.3.1	Find the boundary of $\Omega + \mathbf{z}, \mathbf{z} \in \mathbb{R}^3$ . . . . .	33
2.3.2	Calculate the volume of $\Omega \cap (\Omega + \mathbf{z})$ . . . . .	49
2.4	Calculate $I_s(q)$ . . . . .	50
2.5	Calculate $F(\mathbf{x}, q)$ for $\mathbf{x} \in \partial\Omega$ . . . . .	53
2.5.1	Transformation of $\hat{F}(z_1, z_2)$ . . . . .	55
2.5.2	Sufficient number of Gaussian points in $z$ -direction . . . . .	60
2.5.3	Using bilinear interpolation to approximate $\hat{F}(z_1, z_2)$ . . . . .	63
2.5.4	Applying the Gaussian quadrature rule . . . . .	66
2.5.5	Result and considerations . . . . .	68
<b>3</b>	<b>Finding a body of rotation that minimizes the error <math>J(\Omega)</math></b>	<b>70</b>
3.1	Limitations on $\Omega$ . . . . .	70
3.1.1	Representation of the body of rotation $\Omega$ . . . . .	70
3.1.2	Construct $\Omega$ given a polygonal line . . . . .	72
3.2	Formulation of the actual inverse problem . . . . .	73
3.2.1	Similarities of the 2d and 3d polygonal lines . . . . .	74
3.2.2	Strategy to approach $\Omega_{opt}$ . . . . .	76
3.3	Calculate $G(\mathbf{x})$ for $\mathbf{x} \in \partial\Omega$ . . . . .	77
3.4	Choice of $V$ in $dJ(\Omega, V)$ . . . . .	79
3.4.1	Smooth approximation of the normal vector on $\partial\Omega$ . . . . .	81
3.4.2	Interpolation of $G(\mathbf{x})$ on $\partial\Omega$ . . . . .	94
3.4.3	Final setting of $V(\mathbf{x})$ . . . . .	96
3.5	Deformation of $\Omega$ in direction $V(\mathbf{x})$ . . . . .	96
3.5.1	Approximation of the descent of $J(\Omega)$ . . . . .	99
3.6	Importance of scaling $\Omega$ to a specific volume . . . . .	103
3.7	Parameter settings and their influence on $\log(I_s(q))$ . . . . .	106
3.7.1	Influence of <i>nrOfGaussPoints</i> on $\log(I_s(q))$ . . . . .	107

3.7.2	Influence of <i>nrOfSpherePoints</i> on $\log(I_s(q))$ . . . . .	109
3.7.3	Influence of <i>lengthXY</i> on $\log(I_s(q))$ . . . . .	110
3.7.4	Summary . . . . .	116
3.8	Algorithm to calculate $I_s(q)$ for rotational bodies . . . . .	116
3.8.1	Parameter choices . . . . .	117
3.9	Information about the test runs for this thesis . . . . .	121
<b>4</b>	<b>Random Search ansatz to find <math>\Omega_{opt}</math></b>	<b>123</b>
4.1	Limitations on $\Omega$ for the Random Search ansatz . . . . .	123
4.2	The objective function to minimize . . . . .	125
4.3	The penalty term $\vartheta(\tilde{P})$ . . . . .	125
4.4	Changes on $\Omega$ in one step . . . . .	127
4.5	Accepting or declining the changes on $\Omega$ . . . . .	129
4.6	Unsuccessful test run to reconstruct the polygonal line for the Titan_1b particle and modifications for the algorithm . . . . .	130
4.7	Successful testrun to reconstruct the polygonal line for the Ti- tan_1b particle . . . . .	133
<b>5</b>	<b>Conclusion</b>	<b>138</b>



# 1 The problem and important quantities

For a small particle, a physical measurement method called small angle X-ray scattering (SAXS) provides us some scattering data (more precisely the intensity distribution of the scattered beams) of a given particle which holds information about the overall structure of the particle.

In the book *Small angle X-ray scattering* by O. Glatter and O. Kratky [6], a formula gets derived how to calculate the intensity data SAXS would provide us, given an arbitrary particle with shape  $\Omega \subset \mathbb{R}^3$  and constant electron density.

The goals for this thesis are:

- Deriving formulas that give information about how the intensity data change when we alter the shape of the particle.
- Providing and describing an algorithm that reconstructs the shape given a set of measured intensity data so that the calculated intensity data of the reconstructed shape gets as close as possible to the measured intensity.

## 1.1 SAXS

Small angle X-ray scattering (SAXS) is a method to determine the structure and internal organisation of supramolecular assemblies (referred to as particles in this thesis). For example at the Austrian SAXS Beamline at ELETTRA, Trieste (Italy), measurements can be made for particles which have a size in the range of about 1 nm to 60 nm. Examples of such particles are globular proteins, peptides, lipid or detergent micelles, etc.

For SAXS, only one purified type of particles in a “buffer solution” is observed. Further restrictions are:

- The particles have to be diluted. This means that (almost) every two particles are so far apart that the distance between two particles is not in the range of “inner-particle-dimensions”. This means that particle interaction effects can be neglected.
- The particles have to be monodispers. This means that all particles look exactly the same, i.e. they are copies of one reference particle.

The intensity measured by SAXS is the scattering intensity, which is a function of the scattering vector  $q = \frac{4\pi \sin(\theta)}{\lambda}$ . The quantity  $2\theta$  is the scattering angle and  $\lambda$  is the wavelength of the X-ray. (For example the SAXS Beamline at ELETTRA generates X-rays with a wavelength of  $\lambda = 0.154\text{nm}$ .) The intensity then gets measured for  $q$  in a range  $[q_{min}, q_{max}]$ . Usual values for  $q_{min}$  and  $q_{max}$  are  $[q_{min}, q_{max}] = [0.1 \frac{1}{\text{nm}}, 5.5 \frac{1}{\text{nm}}]$ . When plotting the measured intensity data, usually the logarithm of the scattering intensity is plotted over  $q$ .

The source for this section is a non-published summary of SAXS by Karin Kornmueller, who is a bio-physicist in the Medical University of Graz.

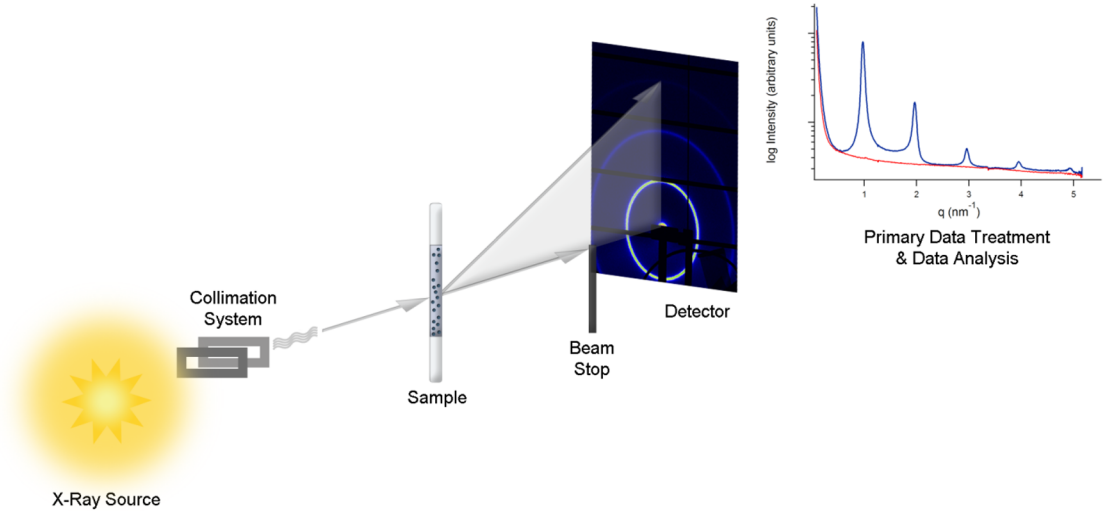


Figure 1: SAXS

## 1.2 Important literature about SAXS

SAXS studies were initiated by A. Guinier. In 1955, he published the book *Small-angle scattering of X-rays* together with G. Fournet [7]. In 1982, O. Glatter and O. Kratky published the book *Small angle X-ray scattering* [6]. This book gives a compact representation of the theory, techniques and applications of SAXS that were present at that time. The chapters were written by different specialists in their respective field. This book is still considered to be a great textbook for graduate students. In 1987, F. A. Feigin and D. I. Svergun published the book *Structure Analysis by Small-Angle X-Ray and Neutron Scattering* [2]. This book additionally discusses the theoretical and experimental work done in the Soviet Union.

In 2010, D. I. Svergun published the review *Small-angle X-ray and neutron scattering as a tool for structural systems biology* [16]. Most importantly for this thesis, he explains existing shape reconstruction algorithms for SAXS and SANS (small angle neutron scattering) for different particle systems and preconditions. Furthermore, he explains:

- the difference between SAXS and SANS,
- when the “structure factor”  $S(s)$  due to interference effects between different particles is important and under which circumstances it can be neglected,

- the differences between SAS (SAXS and/or SANS) studies of monodisperse and polydisperse systems,
- geometrical and weight parameters which can be immediately determined by SAS studies of monodisperse systems.

In 2013, D. I. Svergun published the book *Small Angle X-Ray and Neutron Scattering from Solutions of Biological Macromolecules* [15]. The most important chapters for this thesis are chapter 4 (Data Analysis - Monodisperse Systems) and chapter 6 (Static structural studies). Among other things, he explains a method to determine the shape of a particle by iteratively adding and removing volumetric elements.

In 2018, O. Glatter published the book *Scattering methods and their application in Colloid and Interface Science* [5]. In this book the author discusses the most important theoretical aspects and the most important applications of scattering methods. For this thesis, the chapter 3 is the most important chapter, where he discusses the general theory of the Inverse Scattering Problem and different scattering curves.

### 1.3 The main quantities $I_s(q)$ and $I_{s,meas}(q)$

Our first main restrictive assumption is that the electron density  $\rho(\mathbf{x})$  of the particle with shape  $\Omega \subset \mathbb{R}^3$  is considered to be constant on  $\Omega$ , this means:

$$\rho(\mathbf{x}) = \begin{cases} c_\rho & \mathbf{x} \in \Omega, \\ 0 & \text{else.} \end{cases} \quad (1)$$

The second main restriction is that the background medium has to have a negligible low electron density in comparison to the particle.

The constant electron density restriction is almost never perfectly satisfied for any given particle, but there are particles where it is at least a good approximation. Examples herefore are LDL (low density lipoproteins) like cholesterol at higher temperatures.

Our main quantity of interest is the intensity  $I_s(q)$ . In [6], it is shown on page 124 that  $I_s(q)$  can be written in a compact form with the help of the "autoconvolution of the electron density"  $\gamma(\mathbf{z})$  and the sinc-function (sinus cardinalis)  $\text{sinc}(x)$ :

$$I_s(q) = \int_{\mathbb{R}^3} \gamma(\mathbf{z}) \text{sinc}(q\|\mathbf{z}\|) d\mathbf{z}, \quad (2)$$

where

$$\gamma(\mathbf{z}) = \int_{\mathbb{R}^3} \rho(\mathbf{x}) \rho(\mathbf{x} - \mathbf{z}) d\mathbf{x} = \int_{\mathbb{R}^3} (c_\rho \mathbb{1}_\Omega(\mathbf{x})) (c_\rho \mathbb{1}_\Omega(\mathbf{x} - \mathbf{z})) d\mathbf{x} = c_\rho^2 \int_{\Omega \cap (\Omega + \mathbf{z})} d\mathbf{x} \quad (3)$$



and

$$\text{sinc}(x) = \begin{cases} \frac{\sin(x)}{x} & \text{for } x \neq 0, \\ 1 & \text{for } x = 0. \end{cases}$$

The **measured intensity data**  $I_{s,meas}(q)$  is provided and was obtained by measuring the intensity  $I_s(q)$  of a particle for  $q$  in a range  $[q_{min}, q_{max}]$  e.g. with SAXS. The **intensity data of a shape**  $I_s(q)$  is determined by evaluating  $I_s(q)$  like in equation (2) for sufficiently many values in the same range  $[q_{min}, q_{max}]$ . The goal is to find a shape that minimizes the difference between the measured and the calculated intensity data with respect to some error measure. This is done by successively changing the boundary of the shape in an effective way.

To illustrate the graph of  $I_s(q)$  for  $q$  in  $[q_{min}, q_{max}]$ , usually  $\log(I_s(q))$  is plotted over  $q$ . (The base of the logarithm does not matter – in this thesis the logarithm naturalis  $\ln(x)$  is used.)

We show that  $I_s(q) = I_s(q, \Omega)$  is rotational and translational invariant with respect to  $\Omega$ .

**Lemma 1.1.** *For  $q \in \mathbb{R}$ ,  $I_s(q) = I_s(q, \Omega)$  is rotational and translational invariant with respect to  $\Omega$ . Which means if  $\tilde{\Omega} = R\Omega + \mathbf{y}$  with  $R \in \text{SO}(3)$  and  $\mathbf{y} \in \mathbb{R}^3$ , then  $I_s(q, \Omega) = I_s(q, \tilde{\Omega})$ .*

*Proof.* Let

$$\begin{aligned} \rho(\mathbf{x}, \Omega) &= \begin{cases} c_\rho & \mathbf{x} \in \Omega, \\ 0 & \text{else,} \end{cases} \\ \gamma(\mathbf{z}, \Omega) &= \int_{\mathbb{R}^3} \rho(\mathbf{x}, \Omega) \rho(\mathbf{x} - \mathbf{z}, \Omega) d\mathbf{x}, \\ I_s(q, \Omega) &= \int_{\mathbb{R}^3} \gamma(\mathbf{z}, \Omega) \text{sinc}(q\|\mathbf{z}\|) d\mathbf{z}. \end{aligned}$$

To show the rotational invariance, let  $\Omega \subset \mathbb{R}^3$  and let  $R$  be a rotation matrix. We then have to show that  $I_s(q, \Omega) = I_s(q, R\Omega)$ . It holds that

$$\rho(\mathbf{x}, R\Omega) = \begin{cases} c_\rho & \mathbf{x} \in R\Omega \\ 0 & \text{else} \end{cases} = \begin{cases} c_\rho & R^{-1}\mathbf{x} \in \Omega \\ 0 & \text{else} \end{cases} = \rho(R^{-1}\mathbf{x}, \Omega).$$

For  $\mathbf{y} := R^{-1}\mathbf{x}$  it holds that  $d\mathbf{y} = d\mathbf{x}$  and, using the transformation theorem for integrals,

$$\begin{aligned} \gamma(\mathbf{z}, R\Omega) &= \int_{\mathbb{R}^3} \rho(\mathbf{x}, R\Omega) \rho(\mathbf{x} - \mathbf{z}, R\Omega) d\mathbf{x} = \int_{\mathbb{R}^3} \rho(R^{-1}\mathbf{x}, \Omega) \rho(R^{-1}\mathbf{x} - R^{-1}\mathbf{z}, \Omega) d\mathbf{x} = \\ &= \int_{\mathbb{R}^3} \rho(\mathbf{y}, \Omega) \rho(\mathbf{y} - R^{-1}\mathbf{z}, \Omega) d\mathbf{y} = \gamma(R^{-1}\mathbf{z}, \Omega). \end{aligned}$$

Again, for  $\mathbf{u} := R^{-1}\mathbf{z}$  it holds that  $d\mathbf{u} = d\mathbf{z}$ ,  $\|\mathbf{u}\| = \|\mathbf{z}\|$  and therefore

$$\begin{aligned} I_s(q, R\Omega) &= \int_{\mathbb{R}^3} \gamma(\mathbf{z}, R\Omega) \operatorname{sinc}(q\|\mathbf{z}\|) d\mathbf{z} = \int_{\mathbb{R}^3} \gamma(R^{-1}\mathbf{z}, \Omega) \operatorname{sinc}(q\|\mathbf{z}\|) d\mathbf{z} = \\ &= \int_{\mathbb{R}^3} \gamma(\mathbf{u}, \Omega) \operatorname{sinc}(q\|\mathbf{u}\|) d\mathbf{u} = I_s(q, \Omega). \end{aligned}$$

To show translational invariance, let  $\Omega \subset \mathbb{R}^3$  and let  $\mathbf{a} \in \mathbb{R}^3$ . We then have to show that  $I_s(q, \Omega) = I_s(q, \Omega + \mathbf{a})$ . It holds that

$$\rho(\mathbf{x}, \Omega + \mathbf{a}) = \begin{cases} c_\rho & \mathbf{x} \in \Omega + \mathbf{a} \\ 0 & \text{else} \end{cases} = \begin{cases} c_\rho & \mathbf{x} - \mathbf{a} \in \Omega \\ 0 & \text{else} \end{cases} = \rho(\mathbf{x} - \mathbf{a}, \Omega).$$

For  $\mathbf{y} := \mathbf{x} - \mathbf{a}$  it holds that  $d\mathbf{y} = d\mathbf{x}$  and

$$\begin{aligned} \gamma(\mathbf{z}, \Omega + \mathbf{a}) &= \int_{\mathbb{R}^3} \rho(\mathbf{x}, \Omega + \mathbf{a}) \rho(\mathbf{x} - \mathbf{z}, \Omega + \mathbf{a}) d\mathbf{x} = \int_{\mathbb{R}^3} \rho(\mathbf{x} - \mathbf{a}, \Omega) \rho((\mathbf{x} - \mathbf{a}) - \mathbf{z}, \Omega) d\mathbf{x} = \\ &= \int_{\mathbb{R}^3} \rho(\mathbf{y}, \Omega) \rho(\mathbf{y} - \mathbf{z}, \Omega) d\mathbf{y} = \gamma(\mathbf{z}, \Omega) \end{aligned}$$

and therefore

$$I_s(q, \Omega + \mathbf{a}) = \int_{\mathbb{R}^3} \gamma(\mathbf{z}, \Omega + \mathbf{a}) \operatorname{sinc}(q\|\mathbf{z}\|) d\mathbf{z} = \int_{\mathbb{R}^3} \gamma(\mathbf{z}, \Omega) \operatorname{sinc}(q\|\mathbf{z}\|) d\mathbf{z} = I_s(q, \Omega).$$

□

#### 1.4 A more advantageous representation of $I_s(q)$

The next step is to get yet another representation of  $I_s(q)$  which exploits that  $\gamma(\mathbf{z})$  is only dependent on  $\mathbf{z}$  and not on  $q$ . For this task we have to transform the integral in equation (2) to an integral with respect to spherical coordinates. We rewrite this integral as an integral over spheres and then we show that there exists an  $R \in \mathbb{R}$  so that  $\gamma(\mathbf{z}) = 0$  for all  $\|\mathbf{z}\| > R, \mathbf{z} \in \mathbb{R}^3$ .

We use spherical coordinates  $(r, \phi, \theta)$  like they are defined in Forster, Analysis 3, p. 109 [3] and we use the following transformation theorem to transform the integral we have in equation (2).

**Theorem 1.2** (Transformation to Spherical Coordinates). *Let*

$$\Phi : \mathbb{R}_+ \times [0, \pi] \times [0, 2\pi] \rightarrow \mathbb{R}^3, \quad \Phi(r, \theta, \varphi) := (r \sin \theta \cos \varphi, r \sin \theta \sin \varphi, r \cos \theta). \quad (4)$$

*Then  $f : \mathbb{R}^3 \rightarrow \mathbb{R}$  is integrable if and only if*

$$\mathbb{R}_+ \times [0, \pi] \times [0, 2\pi] \rightarrow \mathbb{R}, \quad (r, \theta, \varphi) \mapsto f(\Phi(r, \theta, \varphi)) r^2 \sin(\theta)$$

is integrable. Then it holds that

$$\int_{\mathbb{R}^3} f(x, y, z) dx dy dz = \int_0^{2\pi} \int_0^\pi \int_0^\infty f(\Phi(r, \theta, \varphi)) r^2 \sin \theta dr d\theta d\varphi$$

*Proof.* See Forster, Analysis 3, p. 109 [3].  $\square$

Moreover, these integrals (over  $r$ ,  $\theta$  and  $\varphi$ ) can be interchanged arbitrarily because of the Theorem of Fubini (Forster, Analysis 3, p. 88 [3]).

**Lemma 1.3.** *For the equation (4) and for  $r \in \mathbb{R}_+$ ,  $\theta \in \mathbb{R}$ ,  $\varphi \in \mathbb{R}$  and  $a \in \mathbb{R}_+$  it holds that*

$$(a) \quad \|\Phi(r, \theta, \varphi)\|_2 = r,$$

$$(b) \quad a\Phi(r, \theta, \varphi) = \Phi(ar, \theta, \varphi),$$

*Proof.* (a)

$$\begin{aligned} \|\Phi(r, \theta, \varphi)\|_2 &= \sqrt{(r \sin \theta \cos \varphi)^2 + (r \sin \theta \sin \varphi)^2 + (r \cos \theta)^2} = \\ &= \sqrt{r^2((\sin \theta)^2((\sin \varphi)^2 + (\cos \varphi)^2) + (\cos \theta)^2)} = \\ &= \sqrt{r^2((\sin \theta)^2 + (\cos \theta)^2)} = \sqrt{r^2} = r. \end{aligned}$$

(b)

$$\begin{aligned} a\Phi(r, \theta, \varphi) &= a(r \sin \theta \cos \varphi, r \sin \theta \sin \varphi, r \cos \theta) = \\ &= (ar \sin \theta \cos \varphi, ar \sin \theta \sin \varphi, ar \cos \theta) = \Phi(ar, \theta, \varphi). \end{aligned}$$

$\square$

**Definition 1.1.** Let  $S^2 = \{\mathbf{x} \in \mathbb{R}^3 \mid \|\mathbf{x}\|_2 = 1\}$  be the unit sphere in  $\mathbb{R}^3$ , let  $f : \mathbb{R}^3 \rightarrow \mathbb{R}$  be an integrable function and let  $\Phi$  be the transformation function to spherical coordinates as described in equation (4). Then the integration over the unit sphere is defined as

$$\int_{\mathbf{p} \in S^2} f(\mathbf{p}) dS(\mathbf{p}) := \int_0^{2\pi} \int_0^\pi f(\Phi(1, \theta, \varphi)) \sin \theta d\theta d\varphi.$$

This definition makes sense, because  $\|\Phi(1, \theta, \varphi)\|_2 = 1$  according to Lemma 1.3 and therefore  $\Phi(1, \theta, \varphi) \in S^2$ . (It can be shown that the integral over  $S^2$  is in fact the integral over the manifold  $S^2$ .)

**Theorem 1.4.** *Another representation for  $I_s(q)$  is given as*

$$I_s(q) = \int_0^\infty r^2 \text{sinc}(rq) \int_{\mathbf{p} \in S^2} \gamma(r\mathbf{p}) dS(\mathbf{p}) dr. \quad (5)$$

*Proof.* For  $r \geq 0$  let  $f_r : \mathbb{R}^3 \rightarrow \mathbb{R}$  be  $f_r(\mathbf{p}) := \gamma(r\mathbf{p})$ . Then for  $\mathbf{z} \in \mathbb{R}^3 \setminus \{\mathbf{0}\}$  it holds that

$$\gamma(\mathbf{z}) = \gamma\left(\|\mathbf{z}\|(1/\|\mathbf{z}\|)\mathbf{z}\right) = f_{\|\mathbf{z}\|}((1/\|\mathbf{z}\|)\mathbf{z}).$$

Starting with equation (2), applying Theorem 1.2, Theorem of Fubini, Lemma 1.3 (several times) and finally Definition 1.1, we get

$$\begin{aligned} I_s(q) &= \int_{\mathbb{R}^3} \gamma(\mathbf{z}) \operatorname{sinc}(q\|\mathbf{z}\|) d\mathbf{z} = \int_{\mathbb{R}^3} f_{\|\mathbf{z}\|}((1/\|\mathbf{z}\|)\mathbf{z}) \operatorname{sinc}(q\|\mathbf{z}\|) d\mathbf{z} = \\ &= \int_0^{2\pi} \int_0^\pi \int_0^\infty f_{\|\Phi(r,\theta,\varphi)\|_2}((1/\|\Phi(r,\theta,\varphi)\|_2)\Phi(r,\theta,\varphi)) \operatorname{sinc}(q\|\Phi(r,\theta,\varphi)\|_2) r^2 \sin \theta \, dr \, d\theta \, d\varphi = \\ &= \int_0^\infty \int_0^{2\pi} \int_0^\pi f_r((1/r)\Phi(r,\theta,\varphi)) \operatorname{sinc}(qr) r^2 \sin \theta \, d\theta \, d\varphi \, dr = \\ &= \int_0^\infty r^2 \operatorname{sinc}(rq) \int_0^{2\pi} \int_0^\pi f_r(\Phi(1,\theta,\varphi)) \sin \theta \, d\theta \, d\varphi \, dr = \\ &= \int_0^\infty r^2 \operatorname{sinc}(rq) \int_{\mathbf{p} \in S^2} f_r(\mathbf{p}) \, dS(\mathbf{p}) \, dr = \int_0^\infty r^2 \operatorname{sinc}(rq) \int_{\mathbf{p} \in S^2} \gamma(r\mathbf{p}) \, dS(\mathbf{p}) \, dr. \end{aligned}$$

□

The following lemma shows some important properties of the support of  $\gamma$  which can be used to rewrite equation (5).

**Lemma 1.5.** *Let  $\Omega$  be open and bounded and  $R := \operatorname{diag}(\Omega) = \sup_{\mathbf{x}, \mathbf{y} \in \Omega} \|\mathbf{x} - \mathbf{y}\|_2$ . Then the following statements hold true:*

- (a)  $R < \infty$ ,
- (b)  $\gamma(\mathbf{z}) \geq 0$  for all  $\mathbf{z} \in \mathbb{R}^3$ ,
- (c)  $\gamma(\mathbf{z}) = 0$  for all  $\mathbf{z} \in \mathbb{R}^3$  if  $\|\mathbf{z}\|_2 > R$ ,
- (d) For all  $\epsilon > 0$  there exists a  $\mathbf{z} \in \mathbb{R}^3$  with  $\|\mathbf{z}\|_2 > R - \epsilon$  so that  $\gamma(\mathbf{z}) > 0$ .

*Proof.* (a)  $\Omega$  is bounded, so there exists a  $K > 0$  so that for all  $\mathbf{x} \in \Omega$  it holds that  $\|\mathbf{x}\|_2 < K$ . Let  $\mathbf{x}, \mathbf{y} \in \Omega$ . Then,  $\|\mathbf{x} - \mathbf{y}\|_2 \leq \|\mathbf{x}\|_2 + \|\mathbf{y}\|_2 < 2K$ . Therefore it follows that  $R = \sup_{\mathbf{x}, \mathbf{y} \in \Omega} \|\mathbf{x} - \mathbf{y}\|_2 \leq 2K$ .

(b) Equation (3) yields  $\gamma(\mathbf{z}) = c_\rho^2 \int_{\Omega \cap (\Omega + \mathbf{z})} d\mathbf{x} \geq 0$ .

(c) Let  $\mathbf{z} \in \mathbb{R}^3$  with  $\gamma(\mathbf{z}) \neq 0$ . Statement (b) yields that  $c_\rho^2 \int_{\Omega \cap (\Omega + \mathbf{z})} d\mathbf{x} = \gamma(\mathbf{z}) > 0$ . So there exists an  $\mathbf{x} \in \mathbb{R}^3$  so that  $\mathbf{x} \in \Omega$  and  $\mathbf{x} \in \Omega + \mathbf{z}$ . This means that  $\mathbf{x} - \mathbf{z} \in \Omega$ . It follows that  $\|\mathbf{z}\|_2 = \|\mathbf{x} - (\mathbf{x} - \mathbf{z})\|_2 \leq \sup_{\mathbf{x}, \mathbf{y} \in \Omega} \|\mathbf{x} - \mathbf{y}\|_2 = R$ .

(d) For  $\epsilon > 0$  let  $\mathbf{x}, \mathbf{y} \in \Omega$  so that  $R - \epsilon < \|\mathbf{x} - \mathbf{y}\|_2 < R$  and let  $\mathbf{z} := \mathbf{x} - \mathbf{y}$ . Because  $\Omega$  is open, there is a  $\delta > 0$  so that  $B_\delta(\mathbf{x}) \subset \Omega$  and  $B_\delta(\mathbf{x} - \mathbf{z}) = B_\delta(\mathbf{y}) \subset \Omega$ . It follows that  $B_\delta(\mathbf{x}) \subset \Omega \cap (\Omega + \mathbf{z})$ . Therefore,  $\gamma(\mathbf{z}) = c_\rho^2 \int_{\Omega \cap (\Omega + \mathbf{z})} d\mathbf{x} \geq c_\rho^2 \int_{B_\delta(\mathbf{x})} d\mathbf{x} > 0$ .  $\square$

We can use Lemma 1.5 to rewrite (5):

$$I_s(q) = \int_0^{\tilde{R}} r^2 \text{sinc}(rq) \Gamma_s(r) dr \quad (6)$$

$$\text{with } \Gamma_s(r) = \int_{\mathbf{p} \in S^2} \gamma(r\mathbf{p}) dS(\mathbf{p}) \quad (7)$$

$$\text{and } \tilde{R} \geq \text{diag}(\Omega). \quad (8)$$

Remark: In the next chapters, another task is to find  $\tilde{R}$  close enough to its lower bound  $R = \sup_{\mathbf{x}, \mathbf{y} \in \Omega} \|\mathbf{x} - \mathbf{y}\|_2$  to make the integration region conveniently small.

The biggest advantage to use (6) instead of (2) is that it is possible to first evaluate  $\Gamma_s(r)$  for sufficiently many (numerical integration) points  $r_1, \dots, r_n$ , save those values in an array, and re-use these values whenever we want to evaluate  $I_s(q)$  for some  $q \in \mathbb{R}$ .

## 1.5 The shape derivative $dI_s(q; \Omega, V)$ of $I_s(q)$

For this section and all following sections, we define  $\Omega \subset \mathbb{R}^3$  as an open, bounded set so that  $\partial\Omega$  is smooth enough that Gauss' Theorem holds. Integration with respect to the 2-dimensional surface measure will be denoted by  $dS(\mathbf{x})$ .

Remember that the task at hand is to construct a shape from given measurement data. This is done by starting at an initial shape (e.g. a sphere), iteratively make small changes on the boundary, recalculate  $I_s(q_i)$  for enough values  $q_i \in \mathbb{R}$  and compare it to the given graph (by some error measure), update the shape to diminish the mismatch - and try to do this procedure iteratively in an efficient way. Therefore it is important to know how  $I_s(q)$  changes for a given  $q \in \mathbb{R}$ , when small changes to  $\Omega$  are made.

First we give a brief introduction to shape sensitivity analysis. This introduction is based on the book *Shape optimization problems* of H. Azegami [1]. In the following, our goal is to find a formula for  $dI_s(q; \Omega, V)$  (Theorem 1.15), whereas the most tricky part is to find a formula for  $d\gamma(\mathbf{z}; \Omega, V)$  (Theorem 1.13). In the lemmas preceeding the main results, we show ways to handle integrals over  $\partial(\Omega \cap (\Omega + \mathbf{z}))$  and we show how we can rewrite  $d\gamma(\mathbf{z}; \Omega, V)$  so that we can

apply Theorem 1.6.

Let

$$T_t : \mathbb{R}^N \rightarrow \mathbb{R}^N \quad t \in [0, \epsilon)$$

be a smooth family of diffeomorphisms, which describes the change of the domain  $\Omega$ :

$$X \in \Omega \mapsto x = T_t(X) \equiv x(t, X).$$

The transformed geometry is given by:

$$\Omega_t = T_t(\Omega).$$

**Definition 1.2.** The *Eulerian velocity field*  $V(t, x)$  at the point  $x$  is given by:

$$V(t, x) = \frac{\partial}{\partial t} [x(t, T_t^{-1}(x))].$$

By this definition one can see that  $x(t, X)$  satisfies the following initial value problem

$$\begin{aligned} \frac{d}{dt} x(t, X) &= V(t, x(t, X)), \\ x(0, X) &= X. \end{aligned} \tag{9}$$

This suggests that we can also start with a given vector field  $V(t, x)$  and define the family of transformations  $T_t$  via the solution map to (9). In this context we use the notation  $T_t(V)(X)$ .

**Definition 1.3** (Eulerian derivative). Let  $\Omega \subset \mathbb{R}^N$  and let  $J$  be a functional defined on a suitable set of “shapes”  $\Omega \in \mathbb{R}^N$  with  $\Omega \mapsto J(\Omega)$ . Then the *Eulerian derivative* of the functional  $J$  at  $\Omega$  in the direction of a vector field  $V$  is given by

$$dJ(\Omega; V) = \lim_{t \searrow 0} \frac{1}{t} [J(\Omega_t) - J(\Omega)] \tag{10}$$

with  $\Omega_t = T_t(V)(\Omega)$ , provided that the limit exists.

Important for this thesis is the shape derivative of the following functional:

**Theorem 1.6.** The shape derivative of  $J(\Omega) := c \int_{\Omega} d\mathbf{x}$  is given by

$$dJ(\Omega; V) = c \int_{\partial\Omega} \langle V(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle dS(\mathbf{x}). \tag{11}$$

*Proof.* Let  $\Omega_t := T_t(V)(\Omega)$ , so  $J(\Omega_t) = c \int_{\Omega_t} d\mathbf{x}$ .

Transforming the integral to an integral over  $\Omega$  (transformation formula, see Analysis 3, Forster O., page 105 [3]) leads to

$$J(\Omega_t) = c \int_{\Omega} \det(DT_t) d\mathbf{x}.$$

with Jacobian  $DT_t = (\frac{\partial T_{t,i}(V)}{\partial X_j}(X))_{i,j=1,\dots,N}$ .

With 1.3 we have

$$dJ(\Omega; V) = c \lim_{t \searrow 0} \frac{1}{t} \int_{\Omega} [\beta(t) - \beta(0)] d\mathbf{x}.$$

with  $\beta(t) := \det(DT_t)$  and  $\beta(0) = 1$ .

Assuming everything smooth enough (exchange limit and integral) we get

$$dJ(\Omega; V) = c \int_{\Omega} \lim_{t \searrow 0} \frac{1}{t} [\beta(t) - \beta(0)] d\mathbf{x} = c \int_{\Omega} \beta'(0) d\mathbf{x}.$$

In the book *Introduction to Shape Optimization* by Sokolowski and Zolesio [14], it is shown on page 76 that

$$\beta'(0) = \operatorname{div} V(0).$$

This follows from the so called Jacobi formula and the chain rule.

Therefore, by Gauss' Theorem, we get

$$dJ(\Omega; V) = c \int_{\Omega} \operatorname{div} V(0) d\mathbf{x} = c \int_{\partial\Omega} \langle V(0, \mathbf{x}), n(\mathbf{x}) \rangle dS(\mathbf{x}) = c \int_{\partial\Omega} \langle V(\mathbf{x}), n(\mathbf{x}) \rangle dS(\mathbf{x}).$$

□

**Lemma 1.7.** *Let  $A, B \subset \mathbb{R}^3$  and  $\mathbf{x} \in \partial(A \cap B)$ . Moreover, suppose that  $\mathbf{x}$  is not an isolated point of  $(A \cap B)$  or  $\mathbb{R} \setminus (A \cap B)$ . Let  $C_1 := \partial A \cap B^o$ ,  $C_2 := \partial B \cap A^o$  and  $C_3 := \partial A \cap \partial B$ . Then*

- (a) *there exists an  $i \in \{1, 2, 3\}$  so that  $\mathbf{x} \in C_i$ ,*
- (b)  *$C_i$  are pairwise disjoint for  $i \in \{1, 2, 3\}$ .*

*Proof.* (a)  $\mathbf{x} \in \partial(A \cap B)$ , so there exists a sequence  $(\mathbf{x}_n)_{n \in \mathbb{N}} \subset (A \cap B)$  and a sequence  $(\mathbf{y}_n)_{n \in \mathbb{N}}$  so that  $(\mathbf{y}_n) \notin (A \cap B)$ ,  $\mathbf{x}_n \neq \mathbf{x}$ ,  $\mathbf{y}_n \neq \mathbf{x}$  for all  $n \in \mathbb{N}$  and  $\mathbf{x} = \lim_{n \rightarrow \infty} \mathbf{x}_n = \lim_{n \rightarrow \infty} \mathbf{y}_n$ . Here, we use the assumption that  $\mathbf{x}$  is not isolated. This especially means that  $\mathbf{x}_n \in A$  and  $\mathbf{y}_n \in B$  for all  $n \in \mathbb{N}$ . Furthermore, it exists a subsequence  $(\mathbf{y}_{n_k})_{k \in \mathbb{N}} \subset (\mathbf{y}_n)_{n \in \mathbb{N}}$  so that either  $(\mathbf{y}_{n_k}) \notin A$  or  $(\mathbf{y}_{n_k}) \notin B$  for all  $k \in \mathbb{N}$ .

Case 1:  $(\mathbf{y}_{n_k}) \notin A$  for all  $k \in \mathbb{N}$ : Because there is a sequence converging to  $\mathbf{x}$  outside of  $A$   $((\mathbf{y}_{n_k})_{n_k \in \mathbb{N}})$  and a sequence converging to  $\mathbf{x}$  inside of  $A$   $((\mathbf{x}_n)_{n \in \mathbb{N}})$ ,  $\mathbf{x} \in \partial A$ .

Case 1.1: There exists a sequence  $(\mathbf{z}_n)_{n \in \mathbb{N}}$  outside  $B$  converging to  $\mathbf{x}$ . Because  $(\mathbf{x}_n)_{n \in \mathbb{N}}$  converges inside of  $B$  to  $\mathbf{x}$ . So,  $\mathbf{x} \in \partial B$  and therefore,  $\mathbf{x} \in C_3$ .

Case 1.2: such a  $(\mathbf{z}_n)_{n \in \mathbb{N}}$  does not exist, so  $\mathbf{x} \in B^o$  and therefore,  $\mathbf{x} \in C_1$ .

Case 2:  $\mathbf{y}_{n_k} \notin B$  for all  $k \in \mathbb{N}$ : Analog, either  $\mathbf{x} \in C_3$  or  $\mathbf{x} \in C_2$ .

- (b) Because  $A^o \cap \partial A = \emptyset$  and  $B^o \cap \partial B = \emptyset$  the statement (b) follows. □

**Lemma 1.8.** Let  $\mathbf{z} \in \mathbb{R}^3$  so that  $\int_{\partial\Omega} \mathbb{1}_{\partial(\Omega+\mathbf{z})}(\mathbf{x}) dS(\mathbf{x}) = 0$ .

Then, for every subset  $D \subset C_3$  with  $C_3 := \partial\Omega \cap \partial(\Omega + \mathbf{z})$  and every  $f \in L^\infty(C_3)$  we find

$$\int_D f(\mathbf{x}) dS(\mathbf{x}) = 0.$$

*Proof.*

$$\left| \int_D f(\mathbf{x}) dS(\mathbf{x}) \right| \leq \|f\|_\infty \int_D dS(\mathbf{x}) \leq \|f\|_\infty \int_{C_2} dS(\mathbf{x}) = \|f\|_\infty \int_{\partial\Omega} \mathbb{1}_{\partial(\Omega+\mathbf{z})}(\mathbf{x}) dS(\mathbf{x}) = 0.$$

Consequently,  $\int_D f(\mathbf{x}) dS(\mathbf{x}) = 0$ .  $\square$

**Lemma 1.9.** Let  $\mathbf{z} \in \mathbb{R}^3$  so that  $\int_{\partial\Omega} \mathbb{1}_{\partial(\Omega+\mathbf{z})}(\mathbf{x}) dS(\mathbf{x}) = 0$  and  $f \in L^\infty(\partial(\Omega \cap (\Omega + \mathbf{z})))$ . Then

$$\int_{\partial(\Omega \cap (\Omega + \mathbf{z}))} f(\mathbf{x}) dS(\mathbf{x}) = \int_{\partial\Omega \cap (\Omega + \mathbf{z})} f(\mathbf{x}) dS(\mathbf{x}) + \int_{\partial(\Omega + \mathbf{z}) \cap \Omega} f(\mathbf{x}) dS(\mathbf{x}).$$

*Proof.* Set  $A := \Omega$  and  $B := \Omega + \mathbf{z}$ . According to Lemma 1.7 we can split  $\partial(A \cap B)$  into the 3 disjoint sets  $C_1 := \partial A \cap B^\circ$ ,  $C_2 := \partial B \cap A^\circ$  and  $C_3 := \partial A \cap \partial B$ . Let  $C_4 := \partial A \cap B$  be the disjoint union of  $C_1 = \partial A \cap B^\circ$  and  $C_5 := \partial A \cap (B \setminus B^\circ)$ . Because  $(B \setminus B^\circ) \subset \partial B$  it holds that  $C_5 \subset C_3$ . Analog, let  $C_6 := \partial B \cap A$  be the disjoint union of  $C_2 = \partial B \cap A^\circ$  and  $C_7 := \partial B \cap (A \setminus A^\circ)$ . Because  $(A \setminus A^\circ) \subset \partial A$  it holds that  $C_7 \subset C_3$ .

It can be concluded:

$$\partial(\Omega \cap (\Omega + \mathbf{z})) = \partial(A \cap B) = C_1 \dot{\cup} C_2 \dot{\cup} C_3 = (C_4 \setminus C_5) \dot{\cup} (C_6 \setminus C_7) \dot{\cup} C_3$$

In the following we split up the integral in disjoint sets. Then we see that the integrals over  $C_5$ ,  $C_7$  and  $C_3$  vanish due to the assumption  $\int_{\partial\Omega} \mathbb{1}_{\partial(\Omega+\mathbf{z})}(\mathbf{x}) dS(\mathbf{x}) = 0$  and Lemma 1.8.

$$\begin{aligned} \int_{\partial(\Omega \cap (\Omega + \mathbf{z}))} f(\mathbf{x}) dS(\mathbf{x}) &= \int_{(C_4 \setminus C_5) \dot{\cup} (C_6 \setminus C_7) \dot{\cup} C_3} f(\mathbf{x}) dS(\mathbf{x}) = \\ &= \int_{C_4} f(\mathbf{x}) dS(\mathbf{x}) - \int_{C_5} f(\mathbf{x}) dS(\mathbf{x}) + \int_{C_6} f(\mathbf{x}) dS(\mathbf{x}) - \int_{C_7} f(\mathbf{x}) dS(\mathbf{x}) + \int_{C_3} f(\mathbf{x}) dS(\mathbf{x}) = \\ &= \int_{C_4} f(\mathbf{x}) dS(\mathbf{x}) - 0 + \int_{C_6} f(\mathbf{x}) dS(\mathbf{x}) - 0 + 0 = \\ &= \int_{\partial\Omega \cap (\Omega + \mathbf{z})} f(\mathbf{x}) dS(\mathbf{x}) + \int_{\partial(\Omega + \mathbf{z}) \cap \Omega} f(\mathbf{x}) dS(\mathbf{x}). \end{aligned}$$

$\square$



**Lemma 1.10.** Let  $\Omega \subset \mathbb{R}^3$  be bounded so that its boundary has Lebesgue measure zero, i.e.  $\int_{\mathbb{R}^3} \mathbb{1}_{\partial\Omega}(\mathbf{x}) d\mathbf{x} = 0$ .

Then it holds that  $\int_{\partial\Omega} \mathbb{1}_{\partial(\Omega+\mathbf{z})}(\mathbf{x}) dS(\mathbf{x}) = 0$  for almost every  $\mathbf{z} \in \mathbb{R}^3$ .

*Proof.* Let  $\mathbf{x} \in \partial\Omega$  and  $\mathbf{y}(\mathbf{x}) := \mathbf{x} - \mathbf{z}$ . Then,  $d\mathbf{y} = d\mathbf{z}$  and

$$\int_{\mathbb{R}^3} \mathbb{1}_{\partial(\Omega+\mathbf{z})}(\mathbf{x}) d\mathbf{z} = \int_{\mathbb{R}^3} \mathbb{1}_{\partial\Omega}(\mathbf{x} - \mathbf{z}) d\mathbf{z} = \int_{\mathbb{R}^3} \mathbb{1}_{\partial\Omega}(\mathbf{y}) d\mathbf{y} = 0$$

and therefore

$$\int_{\mathbb{R}^3} \left( \int_{\partial\Omega} \mathbb{1}_{\partial(\Omega+\mathbf{z})}(\mathbf{x}) dS(\mathbf{x}) \right) d\mathbf{z} = \int_{\partial\Omega} \left( \int_{\mathbb{R}^3} \mathbb{1}_{\partial(\Omega+\mathbf{z})}(\mathbf{x}) d\mathbf{z} \right) dS(\mathbf{x}) = \int_{\partial\Omega} 0 dS(\mathbf{x}) = 0.$$

Because  $\int_{\partial\Omega} \mathbb{1}_{\partial(\Omega+\mathbf{z})}(\mathbf{x}) dS(\mathbf{x}) \geq 0$  for every  $\mathbf{z} \in \mathbb{R}^3$  it holds that

$$\int_{\partial\Omega} \mathbb{1}_{\partial(\Omega+\mathbf{z})}(\mathbf{x}) dS(\mathbf{x}) = 0 \quad \text{for almost every } \mathbf{z} \in \mathbb{R}^3.$$

□

Because we cannot directly apply Theorem 1.6 on  $d\gamma(\mathbf{z}; \Omega, V)$ , we need the upcoming definitions and the following lemmas.

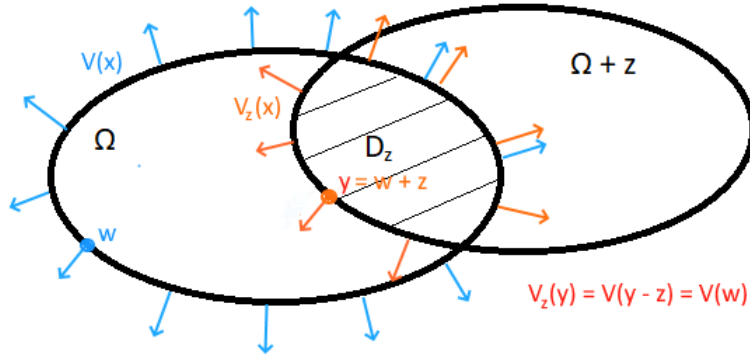


Figure 2: Sketch for the region  $D(\mathbf{z})$  and the vector field  $V_z$

**Definition 1.4** ( $D(\mathbf{z})$  and  $D_t(\mathbf{z})$ ). For  $\mathbf{z} \in \mathbb{R}^3$  we define  $D(\mathbf{z}) := \Omega \cap (\mathbf{z} + \Omega)$  and when we recall  $\Omega_t$  in Definition 1.3, we define  $D_t(\mathbf{z}) := \Omega_t \cap (\mathbf{z} + \Omega_t)$  for  $t \geq 0$ .

**Definition 1.5** ( $V_{\mathbf{z}}$  and  $V_{\mathbf{z},t}$ ). We define the vector field  $V_{\mathbf{z}}$  as

$$V_{\mathbf{z}} : \partial D(\mathbf{z}) \rightarrow \mathbb{R}^3, \quad V_{\mathbf{z}}(\mathbf{x}) = \begin{cases} V(\mathbf{x}) & \text{for } \mathbf{x} \in \partial\Omega, \\ V(\mathbf{x} - \mathbf{z}) & \text{for } \mathbf{x} \in \partial(\Omega + \mathbf{z}) \setminus \partial\Omega. \end{cases}$$

and when we recall  $V(\mathbf{x}, t)$  in Definition 1.2, for  $t \geq 0$  we define the vector field  $V_{\mathbf{z},t}$  as

$$V_{\mathbf{z},t} : \partial D(\mathbf{z}) \rightarrow \mathbb{R}^3, \quad V_{\mathbf{z},t}(\mathbf{x}) = \begin{cases} V(\mathbf{x}, t) & \text{for } \mathbf{x} \in \partial\Omega, \\ V(\mathbf{x} - \mathbf{z}, t) & \text{for } \mathbf{x} \in \partial(\Omega + \mathbf{z}) \setminus \partial\Omega. \end{cases}$$

**Definition 1.6** ( $\gamma_{\mathbf{z}}$ ). For  $\mathbf{z} \in \mathbb{R}^3$  we define

$$\gamma_{\mathbf{z}}(D(\mathbf{z})) := \gamma(\mathbf{z}, \Omega) = c_\rho^2 \int_{\Omega \cap (\mathbf{z} + \Omega)} d\mathbf{x} = c_\rho^2 \int_{D(\mathbf{z})} d\mathbf{x}. \quad (12)$$

**Lemma 1.11.** Let  $\mathbf{z} \in \mathbb{R}^3$  so that  $\int_{\partial\Omega_t} \mathbb{1}_{\partial(\Omega_t + \mathbf{z})}(\mathbf{x}) dS(\mathbf{x}) = 0$ . If  $\partial\Omega_t$  is moved by the vector field  $V_t$ , then  $\partial D_t(\mathbf{z})$  is moved by the vector field  $V_{\mathbf{z},t}$  for almost every  $\mathbf{x} \in \partial D_t(\mathbf{z})$  with respect to the 2-dimensional surface measure on  $\partial D_t(\mathbf{z})$ .

*Proof.* We have to proof this statement for almost every  $\mathbf{x} \in \partial D_t(\mathbf{z})$ . Because  $0 = \int_{\partial\Omega_t} \mathbb{1}_{\partial(\Omega_t + \mathbf{z})}(\mathbf{x}) dS(\mathbf{x}) = \int_{\partial\Omega_t \cap \partial(\Omega_t + \mathbf{z})} dS(\mathbf{x})$ , we see that  $\partial\Omega_t \cap \partial(\Omega_t + \mathbf{z})$  has 2-dimensional surface measure 0. So it is sufficient to proof the statement for all  $\mathbf{x} \in D_t := D_t(\mathbf{z}) \setminus (\partial\Omega_t \cap \partial(\Omega_t + \mathbf{z}))$ . Let  $\mathbf{x} \in D_t \subset D_t(\mathbf{z})$ . So,  $\mathbf{x}$  has to lie either in  $\partial\Omega_t$  or in  $\partial(\Omega_t + \mathbf{z})$ , but not in both because then it would not be in  $D_t$ .

Case 1:  $\mathbf{x} \in \partial\Omega_t$ : Here,  $\partial D_t(\mathbf{z})$  has to move by the same vector as any point in  $\partial\Omega_t$ . Because  $V_{\mathbf{z},t}(\mathbf{x}) = V_t(\mathbf{x})$ , this is the case.

Case 2:  $\mathbf{x} \in \partial(\Omega_t + \mathbf{z}) \setminus \partial\Omega_t$ : Here,  $\mathbf{x} \in \partial(\Omega_t + \mathbf{z}) \subset \partial D_t(\mathbf{z})$  has to move by the same vector as  $(\mathbf{x} - \mathbf{z}) \in \partial\Omega_t$ . Because  $V_{\mathbf{z}}(\mathbf{x}) = V(\mathbf{x} - \mathbf{z})$ , this is the case.

□

**Lemma 1.12.** Let  $\mathbf{z} \in \mathbb{R}^3$  so that  $\int_{\partial\Omega} \mathbb{1}_{\partial(\Omega + \mathbf{z})}(\mathbf{x}) dS(\mathbf{x}) = 0$  and there exists a  $t_0 > 0$  so that for every  $0 \leq t \leq t_0$  it holds that  $\int_{\partial\Omega_t} \mathbb{1}_{\partial(\Omega_t + \mathbf{z})}(\mathbf{x}) dS(\mathbf{x}) = 0$ . Moreover,  $\partial\Omega$  is smooth enough (so that the Gauss' Theorem holds). Then, it holds true that

$$d\gamma(\mathbf{z}; \Omega, V) = d\gamma_{\mathbf{z}}(D(\mathbf{z}); V_{\mathbf{z}}) \quad (13)$$

*Proof.* When we recall Definition 1.3 we see that

$$d\gamma(\mathbf{z}; \Omega, V) = \lim_{t \searrow 0} \frac{1}{t} [\gamma(\mathbf{z}, \Omega_t) - \gamma(\mathbf{z}, \Omega)],$$

whereas  $\Omega_t$  is moved by the vector fields  $V(\tau, \mathbf{x})$ ,  $\tau \in [0, t]$  for the time  $t \geq 0$ . For  $t = 0$ ,  $V(t, \mathbf{x}) = V(\mathbf{x})$  and moreover, we know by Lemma 1.11 that  $(\Omega_t \cap (\mathbf{z} + \Omega_t)) = D_t(\mathbf{z})$  is moved by the vector field  $V_{\mathbf{z}, t}$  for  $t \geq 0$  whenever  $\int_{\partial\Omega_t} \mathbb{1}_{\partial(\Omega_t + \mathbf{z})}(\mathbf{x}) dS(\mathbf{x}) = 0$ . Because of Lemma 1.10 this is true almost everywhere. Therefore it follows that  $D_t(\mathbf{z})$  is the set  $D(\mathbf{z})$  moved by the vector fields  $V_{\mathbf{z}, \tau}(\mathbf{x})$ ,  $\tau \in [0, t_1]$  for some time  $t \geq t_1 \geq 0$ . So we can write

$$\begin{aligned} d\gamma(\mathbf{z}; \Omega, V) &= \lim_{t \searrow 0} \frac{1}{t} [\gamma(\mathbf{z}, \Omega_t) - \gamma(\mathbf{z}, \Omega)] = \lim_{t \searrow 0} \frac{1}{t} \left[ c_\rho^2 \int_{\Omega_t \cap (\mathbf{z} + \Omega_t)} d\mathbf{x} - \gamma_{\mathbf{z}}(D(\mathbf{z})) \right] = \\ &= \lim_{t \searrow 0} \frac{1}{t} \left[ c_\rho^2 \int_{D_t(\mathbf{z})} d\mathbf{x} - \gamma_{\mathbf{z}}(D(\mathbf{z})) \right] = \lim_{t \searrow 0} \frac{1}{t} [\gamma_{\mathbf{z}}(D_t(\mathbf{z})) - \gamma_{\mathbf{z}}(D(\mathbf{z}))] = \\ &= d\gamma_{\mathbf{z}}(D(\mathbf{z}), V_{\mathbf{z}}) \end{aligned}$$

□

Unfortunately, we could not proof the last lemma without the condition that  $\int_{\partial\Omega} \mathbb{1}_{\partial(\Omega + \mathbf{z})}(\mathbf{x}) dS(\mathbf{x}) = 0$  for almost every  $\mathbf{z} \in \mathbb{R}^3$ . We needed the additional condition that there exists a  $t_0 > 0$  so that for every  $0 \leq t \leq t_0$  it holds that  $\int_{\partial\Omega_t} \mathbb{1}_{\partial(\Omega_t + \mathbf{z})}(\mathbf{x}) dS(\mathbf{x}) = 0$ .

**Theorem 1.13.** *Let  $\mathbf{z} \in \mathbb{R}^3$  so that  $\int_{\partial\Omega} \mathbb{1}_{\partial(\Omega + \mathbf{z})}(\mathbf{x}) dS(\mathbf{x}) = 0$  and suppose there exists a  $t_0 > 0$  so that for every  $0 \leq t \leq t_0$  it holds that  $\int_{\partial\Omega_t} \mathbb{1}_{\partial(\Omega_t + \mathbf{z})}(\mathbf{x}) dS(\mathbf{x}) = 0$ . Then the shape derivative of  $\gamma(\mathbf{z}) = \gamma(\mathbf{z}, \Omega) = c_\rho^2 \int_{\Omega \cap (\mathbf{z} + \Omega)} d\mathbf{x}$  is given by:*

$$d\gamma(\mathbf{z}; \Omega, V) = c_\rho^2 \int_{\partial\Omega} (\mathbb{1}_{\Omega + \mathbf{z}}(\mathbf{x}) + \mathbb{1}_{\Omega - \mathbf{z}}(\mathbf{x})) \langle V(\mathbf{x}), n(\mathbf{x}) \rangle dS(\mathbf{x}). \quad (14)$$

*Proof.* It follows from Theorem 1.6, Lemma 1.9 and Lemma 1.12 that:

$$\begin{aligned}
d\gamma(\mathbf{z}; \Omega, V) &= d\gamma_{\mathbf{z}}(D(\mathbf{z}); V_{\mathbf{z}}) = c_{\rho}^2 \int_{\partial D(\mathbf{z})} \langle V_{\mathbf{z}}(\mathbf{x}), n_{\partial D(\mathbf{z})}(\mathbf{x}) \rangle dS(\mathbf{x}) = \\
&= c_{\rho}^2 \int_{\partial(\Omega \cap (\mathbf{z} + \Omega))} \langle V_{\mathbf{z}}(\mathbf{x}), n_{\partial D(\mathbf{z})}(\mathbf{x}) \rangle dS(\mathbf{x}) = \\
&= c_{\rho}^2 \left( \int_{\partial\Omega \cap (\Omega + \mathbf{z})} \langle V_{\mathbf{z}}(\mathbf{x}), n_{\partial D(\mathbf{z})}(\mathbf{x}) \rangle dS(\mathbf{x}) + \int_{\partial(\Omega + \mathbf{z}) \cap \Omega} \langle V_{\mathbf{z}}(\mathbf{x}), n_{\partial D(\mathbf{z})}(\mathbf{x}) \rangle dS(\mathbf{x}) \right) = \\
&= c_{\rho}^2 \left( \int_{\partial\Omega} \mathbb{1}_{\Omega + \mathbf{z}}(\mathbf{x}) \langle V_{\mathbf{z}}(\mathbf{x}), n_{\partial\Omega}(\mathbf{x}) \rangle dS(\mathbf{x}) + \int_{\partial(\Omega + \mathbf{z})} \mathbb{1}_{\Omega}(\mathbf{x}) \langle V_{\mathbf{z}}(\mathbf{x}), n_{\partial(\Omega + \mathbf{z})}(\mathbf{x}) \rangle dS(\mathbf{x}) \right) = \\
&= c_{\rho}^2 \left( \int_{\partial\Omega} \mathbb{1}_{\Omega + \mathbf{z}}(\mathbf{x}) \langle V(\mathbf{x}), n(\mathbf{x}) \rangle dS(\mathbf{x}) + \int_{\partial(\Omega + \mathbf{z})} \mathbb{1}_{\Omega}(\mathbf{x}) \langle V(\mathbf{x} - \mathbf{z}), n(\mathbf{x} - \mathbf{z}) \rangle dS(\mathbf{x}) \right) = \\
&= c_{\rho}^2 \left( \int_{\partial\Omega} \mathbb{1}_{\Omega + \mathbf{z}}(\mathbf{x}) \langle V(\mathbf{x}), n(\mathbf{x}) \rangle dS(\mathbf{x}) + \int_{\partial\Omega} \mathbb{1}_{\Omega - \mathbf{z}}(\mathbf{x}) \langle V(\mathbf{x}), n(\mathbf{x}) \rangle dS(\mathbf{x}) \right) = \\
&= c_{\rho}^2 \int_{\partial\Omega} (\mathbb{1}_{\Omega + \mathbf{z}}(\mathbf{x}) + \mathbb{1}_{\Omega - \mathbf{z}}(\mathbf{x})) \langle V(\mathbf{x}), n(\mathbf{x}) \rangle dS(\mathbf{x}).
\end{aligned}$$

Remark: Because of the assumption  $\int_{\partial\Omega} \mathbb{1}_{\partial(\Omega + \mathbf{z})}(\mathbf{x}) dS(\mathbf{x}) = 0$  and Lemma 1.8 we see that the values of  $V$  and  $n$  on  $\mathbf{x} \in (\partial\Omega \cap \partial(\Omega + \mathbf{z}))$  are irrelevant. So, on  $\partial\Omega$  we can write  $V_{\mathbf{z}}(\mathbf{x}) = V(\mathbf{x})$  and on  $\partial(\Omega + \mathbf{z})$  we can write  $V_{\mathbf{z}}(\mathbf{x}) = V(\mathbf{x} - \mathbf{z})$  and moreover,  $n_{\partial\Omega}$  and  $n_{\partial(\Omega + \mathbf{z})}$  are well defined within the respective integrals.  $\square$

**Lemma 1.14.** *It holds that*

$$\int_{\mathbb{R}^3} \mathbb{1}_{\Omega - \mathbf{z}}(\mathbf{x}) \operatorname{sinc}(q\|\mathbf{z}\|) d\mathbf{z} = \int_{\mathbb{R}^3} \mathbb{1}_{\Omega + \mathbf{z}}(\mathbf{x}) \operatorname{sinc}(q\|\mathbf{z}\|) d\mathbf{z}. \quad (15)$$

*Proof.* Set  $\mathbf{u} := -\mathbf{z}$ . Then  $d\mathbf{u} = d\mathbf{z}$ ,  $\|\mathbf{u}\| = \|\mathbf{z}\|$ , and integrating over all  $\mathbf{z} \in \mathbb{R}^3$  is equivalent to integrating over all  $\mathbf{u} \in \mathbb{R}^3$ . Therefore:

$$\int_{\mathbb{R}^3} \mathbb{1}_{\Omega - \mathbf{z}}(\mathbf{x}) \operatorname{sinc}(q\|\mathbf{z}\|) d\mathbf{z} = \int_{\mathbb{R}^3} \mathbb{1}_{\Omega + \mathbf{u}}(\mathbf{x}) \operatorname{sinc}(q\|\mathbf{u}\|) d\mathbf{u} = \int_{\mathbb{R}^3} \mathbb{1}_{\Omega + \mathbf{z}}(\mathbf{x}) \operatorname{sinc}(q\|\mathbf{z}\|) d\mathbf{z}.$$

$\square$

**Theorem 1.15.** *Suppose, for almost every  $\mathbf{z} \in \mathbb{R}^3$  there exists a  $t_0 > 0$  so that for every  $0 \leq t \leq t_0$  it holds that  $\int_{\partial\Omega_t} \mathbb{1}_{\partial(\Omega_t + \mathbf{z})}(\mathbf{x}) dS(\mathbf{x}) = 0$ .*

Then, the shape derivative of  $I_s(q)$  is given by:

$$dI_s(q; \Omega, V) = 2c_\rho^2 \int_{\partial\Omega} \int_{\Omega} \text{sinc}(q\|\mathbf{z} - \mathbf{x}\|) d\mathbf{z} \langle V(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle dS(\mathbf{x}). \quad (16)$$

*Proof.* When we observe  $I_s(q) = \int_{\mathbb{R}^3} \gamma(\mathbf{z}) \text{sinc}(q\|\mathbf{z}\|) d\mathbf{z}$ , we see that the only term depending on  $\Omega$  is  $\gamma(\mathbf{z})$ . Therefore it holds that

$$dI_s(q; \Omega, V) = \int_{\mathbb{R}^3} d\gamma(\mathbf{z}; \Omega, V) \text{sinc}(q\|\mathbf{z}\|) d\mathbf{z}.$$

Because of Lemma 1.10 we know that  $\int_{\partial\Omega} \mathbb{1}_{\partial(\Omega+\mathbf{z})}(\mathbf{x}) dS(\mathbf{x}) = 0$  for almost every  $\mathbf{z} \in \mathbb{R}^3$ . Therefore we can apply Theorem 1.13 on almost every  $\gamma(\mathbf{z})$ . To simplify the integral we also can apply Lemma 1.14 in the end.

$$\begin{aligned} dI_s(q; \Omega, V) &= \int_{\mathbb{R}^3} d\gamma(\mathbf{z}; \Omega, V) \text{sinc}(q\|\mathbf{z}\|) d\mathbf{z} = \\ &= \int_{\mathbb{R}^3} c_\rho^2 \int_{\partial\Omega} (\mathbb{1}_{\Omega+\mathbf{z}}(\mathbf{x}) + \mathbb{1}_{\Omega-\mathbf{z}}(\mathbf{x})) \langle V(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle dS(\mathbf{x}) \text{sinc}(q\|\mathbf{z}\|) d\mathbf{z} = \\ &= c_\rho^2 \int_{\partial\Omega} \left( \int_{\mathbb{R}^3} (\mathbb{1}_{\Omega+\mathbf{z}}(\mathbf{x}) + \mathbb{1}_{\Omega-\mathbf{z}}(\mathbf{x})) \text{sinc}(q\|\mathbf{z}\|) d\mathbf{z} \right) \langle V(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle dS(\mathbf{x}) = \\ &= c_\rho^2 \int_{\partial\Omega} \left( 2 \int_{\mathbb{R}^3} \mathbb{1}_{\Omega+\mathbf{z}}(\mathbf{x}) \text{sinc}(q\|\mathbf{z}\|) d\mathbf{z} \right) \langle V(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle dS(\mathbf{x}) = \\ &= 2c_\rho^2 \int_{\partial\Omega} \int_{\mathbb{R}^3} \mathbb{1}_{\Omega+\mathbf{z}}(\mathbf{x}) \text{sinc}(q\|\mathbf{z}\|) d\mathbf{z} \langle V(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle dS(\mathbf{x}) = \\ &= 2c_\rho^2 \int_{\partial\Omega} \int_{\mathbb{R}^3} \mathbb{1}_{\Omega}(\mathbf{x} - \mathbf{z}) \text{sinc}(q\|\mathbf{z}\|) d\mathbf{z} \langle V(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle dS(\mathbf{x}). \end{aligned}$$

For given  $\mathbf{x}$  set  $\mathbf{u} := \mathbf{x} - \mathbf{z}$ . Then,  $\|\mathbf{z}\| = \|\mathbf{x} - \mathbf{u}\| = \|\mathbf{u} - \mathbf{x}\|$  and integrating over all  $\mathbf{z} \in \mathbb{R}^3$  is equivalent to integrating over all  $\mathbf{u} \in \mathbb{R}^3$ . Therefore we can write

$$\begin{aligned} dI_s(q; \Omega, V) &= 2c_\rho^2 \int_{\partial\Omega} \int_{\mathbb{R}^3} \mathbb{1}_{\Omega}(\mathbf{x} - \mathbf{z}) \text{sinc}(q\|\mathbf{z}\|) d\mathbf{z} \langle V(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle dS(\mathbf{x}) = \\ &= 2c_\rho^2 \int_{\partial\Omega} \int_{\mathbb{R}^3} \mathbb{1}_{\Omega}(\mathbf{u}) \text{sinc}(q\|\mathbf{u} - \mathbf{x}\|) d\mathbf{u} \langle V(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle dS(\mathbf{x}) = \end{aligned}$$

$$\begin{aligned}
&= 2 c_\rho^2 \int_{\partial\Omega} \int_{\Omega} \text{sinc}(q\|\mathbf{u} - \mathbf{x}\|) d\mathbf{u} \langle V(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle dS(\mathbf{x}) = \\
&= 2 c_\rho^2 \int_{\partial\Omega} \int_{\Omega} \text{sinc}(q\|\mathbf{z} - \mathbf{x}\|) d\mathbf{z} \langle V(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle dS(\mathbf{x}).
\end{aligned}$$

□

Setting

$$F(\mathbf{x}, q) := 2 c_\rho^2 \int_{\Omega} \text{sinc}(q\|\mathbf{z} - \mathbf{x}\|) d\mathbf{z}, \quad (17)$$

we can write

$$dI_s(q; \Omega, V) = \int_{\partial\Omega} F(\mathbf{x}, q) \langle V(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle dS(\mathbf{x}). \quad (18)$$

Introducing  $F(\mathbf{x}, q)$  makes sense, because this term is not dependent on the vector field  $V$ . When e.g. testing  $dI_s(q; \Omega, V)$  on several vector fields  $V$  we can split the task in first evaluating enough values for  $F(\mathbf{x}, q)$ ,  $\mathbf{x} \in \partial\Omega$ , saving those values in an array, and then reuse them when evaluating  $dI_s(q; \Omega, V)$ .

## 1.6 Error measure $J(\Omega)$ between $I_s(q)$ and $I_{s,meas}(q)$ and its shape derivative

The used error measure between the actual simulated intensity data of the shape  $I_s(q)$  and the provided measured intensity data  $I_{s,meas}(q)$  was chosen as:

$$\begin{aligned}
J(\Omega) &:= \int_{q_{min}}^{q_{max}} \left| \ln(I_s(q) + \epsilon) - \ln(I_{s,meas}(q) + \epsilon) \right|^2 dq = \\
&= \int_{q_{min}}^{q_{max}} \left( \ln \frac{I_s(q) + \epsilon}{I_{s,meas}(q) + \epsilon} \right)^2 dq.
\end{aligned} \quad (19)$$

The error measure was chosen like that because of the following reasons:

- $I_s(q)$  (and also  $I_{s,meas}(q)$ ) are declining very fast with increasing  $q$ , so comparing the values of  $\ln(I_s(q))$  and  $\ln(I_{s,meas}(q))$  makes more sense. Using (19), we ensure that small  $q$  have about the same contribution to the error  $J(\Omega)$  as larger  $q$  values.
- Some small constant  $\epsilon$  got added to  $I_s(q)$  and also  $I_{s,meas}(q)$  because:
  - to provide numerical stability,
  - for some shapes like e.g. a sphere,  $I_s(q)$  becomes 0 for some values — for such  $q$  the  $\ln$  would converge to  $-\infty$ , so an error measure without the added  $\epsilon$  would not be a good choice for such shapes.
- The difference of  $\ln(I_s(q) + \epsilon)$  and  $\ln(I_{s,meas}(q) + \epsilon)$  gets squared, because

- easier representation of the shape derivative of  $J(\Omega)$ ,
- proportionally bigger influence of such  $q$  where  $\ln(I_s(q) + \epsilon)$  and  $\ln(I_{s,meas}(q) + \epsilon)$  differ more.

**Theorem 1.16.** *The shape derivative of  $J(\Omega)$  is given by*

$$dJ(\Omega; V) = 2 \int_{\partial\Omega} \int_{q_{min}}^{q_{max}} F(\mathbf{x}, q) \frac{1}{I_s(q) + \epsilon} \ln \frac{I_s(q) + \epsilon}{I_{s,meas}(q) + \epsilon} dq \langle V(\mathbf{x}), n(\mathbf{x}) \rangle dS(\mathbf{x}) \quad (20)$$

with  $F(\mathbf{x}, q)$  defined in (17).

*Proof.* First we have to make sure that the usual (derivation) chain rule also applies on the shape derivative — when observing Definition 1.3 we see that the shape derivative itself is nothing more than the derivative of a composition of differentiable functions/functionals.

Furthermore when observing  $J(\Omega)$  we see that the only part dependent on  $\Omega$  is  $I_s(q)$ . We also assume everything is smooth enough to interchange integrals with derivatives. Moreover, we use the representation of  $dI_s(q; \Omega, V)$  given at (18). Therefore

$$\begin{aligned} dJ(\Omega; V) &= \int_{q_{min}}^{q_{max}} 2 \left( \ln \frac{I_s(q) + \epsilon}{I_{s,meas}(q) + \epsilon} \right) \frac{I_{s,meas}(q) + \epsilon}{I_s(q) + \epsilon} \frac{1}{I_{s,meas}(q) + \epsilon} dI_s(q; \Omega, V) dq = \\ &= \int_{q_{min}}^{q_{max}} 2 \left( \ln \frac{I_s(q) + \epsilon}{I_{s,meas}(q) + \epsilon} \right) \frac{1}{I_s(q) + \epsilon} \int_{\partial\Omega} F(\mathbf{x}, q) \langle V(\mathbf{x}), n(\mathbf{x}) \rangle dS(\mathbf{x}) dq = \\ &= 2 \int_{\partial\Omega} \int_{q_{min}}^{q_{max}} F(\mathbf{x}, q) \frac{1}{I_s(q) + \epsilon} \ln \frac{I_s(q) + \epsilon}{I_{s,meas}(q) + \epsilon} dq \langle V(\mathbf{x}), n(\mathbf{x}) \rangle dS(\mathbf{x}). \end{aligned}$$

□

Setting

$$G(\mathbf{x}) := 2 \int_{q_{min}}^{q_{max}} F(\mathbf{x}, q) \frac{1}{I_s(q) + \epsilon} \ln \frac{I_s(q) + \epsilon}{I_{s,meas}(q) + \epsilon} dq, \quad (21)$$

we can write

$$dJ(\Omega; V) := \int_{\partial\Omega} G(\mathbf{x}) \langle V(\mathbf{x}), n(\mathbf{x}) \rangle dS(\mathbf{x}). \quad (22)$$

## 2 Algorithms to evaluate $I_s(q)$ and its directional derivatives

In this chapter, the algorithm to evaluate the objective function  $I_s(q)$  and its directional derivatives are described. The algorithm has been written in *Python*.

$I_s(q)$  is dependent of the electron density  $\rho(\mathbf{x})$  which is assumed to be constant on a bounded region  $\Omega \subset \mathbb{R}^3$ , this means:

$$\rho(\mathbf{x}) = \begin{cases} c_\rho & \mathbf{x} \in \Omega, \\ 0 & \text{else.} \end{cases} \quad (23)$$

An approximate cuboidal bound  $C_\Omega$  of  $\Omega$  with  $\Omega \subset C_\Omega$  has to be provided for the algorithm. A function  $\varphi(\mathbf{x})$  that describes  $\Omega$  has to be provided. More specifically we assume that  $\varphi(\mathbf{x}) < 0$  if  $\mathbf{x} \in \Omega$  and  $\varphi(\mathbf{x}) \geq 0$  if  $\mathbf{x} \in C_\Omega \setminus \Omega$ . In other words,  $\Omega = \{\mathbf{x} \in \mathbb{R}^3 \mid \varphi(\mathbf{x}) < 0\}$ . We call  $\varphi$  a level-set function for  $\Omega$ .

### 2.1 Quantities of interest

In the last chapter we concluded the following:

- $I_s(q) = \int_{\mathbb{R}^3} \gamma(\mathbf{z}) \text{sinc}(q\|\mathbf{z}\|) d\mathbf{z}$  is our main quantity of interest, where:
  - $q = \|\mathbf{q}\|_2$  with  $\mathbf{q} \in \mathbb{R}^3$ ,
  - $\gamma(\mathbf{z}) = \int_{\mathbb{R}^3} \rho(\mathbf{x}) \rho(\mathbf{x}-\mathbf{z}) d\mathbf{x} = \int_{\mathbb{R}^3} (c_\rho \mathbb{1}_\Omega(\mathbf{x})) (c_\rho \mathbb{1}_\Omega(\mathbf{x}-\mathbf{z})) d\mathbf{x} = c_\rho^2 \int_{\Omega \cap (\Omega+\mathbf{z})} d\mathbf{x}$ ,
  - $\text{sinc}(x) = \begin{cases} \frac{\sin(x)}{x} & \text{for } x \neq 0, \\ 1 & \text{for } x = 0. \end{cases}$
- For  $R \geq \text{diam}(\Omega)$  we have  $\gamma(\mathbf{z}) = 0$  for all  $\mathbf{z} \in \mathbb{R}^3$  with  $\|\mathbf{z}\|_2 > R$ . Moreover,
 
$$I_s(q) = \int_0^R \frac{r}{q} \sin(rq) \int_{\mathbf{p} \in S^2} \gamma(r\mathbf{p}) dS(\mathbf{p}) dr$$
 is another representation of  $I_s(q)$ .
- The shape derivative  $dI_s(q; \Omega, V)$  of  $I_s(q)$  is given by

$$dI_s(q; \Omega, V) = \int_{\partial\Omega} F(\mathbf{x}, q) \langle V(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle dS(\mathbf{x}),$$

where  $F(\mathbf{x}, q) := 2c_\rho^2 \int_{\Omega} \text{sinc}(q\|\mathbf{z} - \mathbf{x}\|) d\mathbf{z}$ .

The algorithm to calculate  $I_s(q)$  consists of the following parts:

1. Find the boundary of the region  $\Omega$ .
2. For sufficiently many sampling points  $\mathbf{z} \in \mathbb{R}^3$  calculate  $\gamma(\mathbf{z})$ .



3. Numerically integrate over  $\mathbb{R}^3$  to obtain  $I_s(q)$ .

Furthermore, for our algorithm it is never necessary to directly calculate  $dI_s(q; \Omega, V)$ , it is only necessary to calculate  $F(\mathbf{x}, q)$  for  $\mathbf{x} \in \partial\Omega$  when evaluating the shape derivative  $dJ(\Omega; V)$  of our error measure  $J(\Omega)$ .

## 2.2 Finding the boundary of the region

In this section, we

1. define a grid which is laid over  $\Omega$ ,
2. define which grid points are considered to be “on the boundary” of the region  $\Omega$ ,
3. define restrictions on  $\Omega$  and we describe an algorithm that finds all grid points on the boundary (grid points in  $\Omega$  where a neighboring grid point is outside  $\Omega$ ), when those restrictions on  $\Omega$  are met,
4. proof, that this algorithm finds all grid points on the boundary,
5. save all those “boundary grid points” in a variable,
6. explain the necessary post-calculation of the boundary.

First, a cuboid has to be defined that contains  $\Omega$ . It is defined by its coordinate boundaries  $x_{min}, y_{min}, z_{min}, x_{max}, y_{max}$ , and  $z_{max}$ :

$$C_\Omega := [x_{min}, x_{max}] \times [y_{min}, y_{max}] \times [z_{min}, z_{max}]. \quad (24)$$

With the starting point  $[x_{min}, y_{min}, z_{min}]$  and the mesh sizes  $l_{x,y}$  and  $l_z$  (lengths in  $x$ -,  $y$ - and  $z$ -direction), let the grid  $G$  be

$$G := \{[x_{min} + i l_{x,y}, y_{min} + j l_{x,y}, z_{min} + k l_z] \mid i, j, k \in \mathbb{Z}\}. \quad (25)$$

Then the resulting grid  $G_{C_\Omega}$  laid over  $\Omega$  is:

$$G_{C_\Omega} := G \cap C_\Omega = \{[x_{min} + i l_{x,y}, y_{min} + j l_{x,y}, z_{min} + k l_z] \mid \begin{array}{l} i, j, k \in \mathbb{N}_0 \text{ so that } x_{min} + i l_{x,y} \leq x_{max}, \\ y_{min} + j l_{x,y} \leq y_{max}, z_{min} + k l_z \leq z_{max} \end{array}\}. \quad (26)$$

Moreover, we make the assumption that  $x_{min}$  and  $y_{min}$  are multiples of  $l_{x,y}$  and  $z_{min}$  is a multiple of  $l_z$ . This means there exist  $i_1, j_1, k_1 \in \mathbb{Z}$  so that

$$\begin{aligned} x_{min} &= i_1 l_{x,y}, \\ y_{min} &= j_1 l_{x,y}, \\ z_{min} &= k_1 l_z. \end{aligned} \quad (27)$$

With the help of assumption (27), we can rewrite  $G$  as

$$G := \{[i_2 l_{x,y}, j_2 l_{x,y}, k_2 l_z] \mid i_2, j_2, k_2 \in \mathbb{Z}\}. \quad (28)$$

**Definition 2.1** (Grid coordinates). For the grid  $G$  (and therefore also for the grid  $G_{C_\Omega}$ ) and for  $i, j, k \in \mathbb{Z}$  let:

$$[i, j, k]_G = [x_{min} + i l_{x,y}, y_{min} + j l_{x,y}, z_{min} + k l_z]. \quad (29)$$

Furthermore, for  $a, b, c \in \mathbb{R}$  let

$$[a, b, c]_G = [x_{min} + a l_{x,y}, y_{min} + b l_{x,y}, z_{min} + c l_z]. \quad (30)$$

**Definition 2.2** (Neighbors of a grid point). Neighbors of a grid point  $[i, j, k]_G$  are the grid points with only one grid-coordinate decreased or increased by 1 (e.g.  $[i - 1, j, k]_G, [i + 1, j, k]_G, [i, j - 1, k]_G, \dots$ ).

**Definition 2.3** (Nearby grid points of a grid point). The nearby grid points of a grid point  $\mathbf{g} = [i, j, k]_G$  are the points

$$\{[i + i_0, j + j_0, k + k_0]_G \mid i_0, j_0, k_0 \in \{-1, 0, 1\}\},$$

which are the 27 grid points around  $\mathbf{g}$  (including  $\mathbf{g}$  itself).

**Definition 2.4** (Boundary grid point). The grid point  $\mathbf{g} = [i, j, k]_G$  is a boundary grid point of  $\Omega$  with respect to  $G$ , if  $\mathbf{g} \in \Omega$  and  $\mathbf{g}$  has a neighbor outside  $\Omega$ . A boundary grid point is therefore a grid point where the level set function which encodes  $\Omega$  changes sign if going from  $\mathbf{g}$  to one of its neighbors.

**Definition 2.5** (Boundary part). The boundary  $\partial\Omega$  of  $\Omega$  consists of one or more connected components in  $\mathbb{R}^3$ . We call the connected components of  $\partial\Omega$  boundary parts of  $\Omega$ .

We say, a boundary grid point  $\mathbf{g}$  is a grid point of the boundary part  $B \subset \partial\Omega$ , if  $\mathbf{g}$  has a grid neighbor  $\mathbf{p}$  outside  $\Omega$  and the connecting line  $\{\lambda\mathbf{g} + (1 - \lambda)\mathbf{p} \mid \lambda \in [0, 1]\}$  intersects  $B$ .

The output of the boundary finding algorithm are all points on the grid  $G$  that are boundary grid points. The point indices of those points are saved in the variable *boundary*.

The following steps are necessary to find the boundary of  $\Omega$ :

1. Locate  $\Omega$ : For every boundary part  $B$  of  $\Omega$ , at least one boundary grid point of  $B$  has to be found.
2. Fill variable *boundary*: Recursively check nearby points of the already found points if they are also boundary points.
3. Post calculation: Important parameters like the coordinate range of  $\Omega$  and a ball (with midpoint and radius) that contains  $\Omega$  are calculated.

The three steps are explained in detail in the subsequent sections.

### 2.2.1 Structure of the variable "boundary"

Important for the performance of the program is the structure of the variables *boundary* and *boundary\_out*. The main focus lies on fast inserting, checking and iterating of the three dimensional indices  $[i, j, k]$ . The following components are used to describe the structure:

- *SortedDictionary*: A *Dictionary* is a set of  $\langle \text{key}, \text{value} \rangle$  object pairs. No two different pairs of this set are allowed to have the same key. And for the key respectively value object, usually every type of object (e.g. integer, string, floating point number, arrays, lists, other sets, ...) is allowed. For the *SortedDictionary*, the key has to be a sortable object like an integer, floating point number or a string (lexicographic order). When iterating over all pairs of a *SortedDictionary*, the pairs are returned in the ascending order of the keys. A *Dictionary* alone has no predefined order (at least no order that should be used).  
The main advantage of a *SortedDictionary* in comparison to a *Dictionary* with size  $n$  is, that checking if a key exists or returning the value of some given key takes  $O(\log(n))$  checks, whereas for a *Dictionary* it takes  $O(n)$  checks (same for removing an element).  
A *Dictionary* is usually written as *Dictionary*  $\langle \text{type\_key}, \text{type\_value} \rangle$  (e.g. *SortedDictionary*  $\langle \text{int}, \text{string} \rangle$ ).
- *SortedList*: A list of sortable objects (like numbers, strings...). To describe it, it is usually written as *SortedList*  $\langle \text{type} \rangle$ .

The structure of the variable *boundary* and *boundary\_out* is:

$$\text{SortedDictionary} \langle \text{int}, \text{SortedDictionary} \langle \text{int}, \text{SortedList} \langle \text{int} \rangle \rangle \rangle \quad (31)$$

Hereby the first integer responds to the  $x$ -coordinate, the second integer responds to the  $y$ -coordinate and the third integer responds to the  $z$ -coordinate of the point indices. So the outermost *SortedDictionary*'s keys are the different  $x$ -coordinates. Every of these  $x$ -coordinates has a *SortedDictionary* which keys are the different  $y$ -coordinates (of this  $x$ -coordinate), and every of these  $y$ -coordinates has a *SortedList* of  $z$ -coordinates.

The following *Python* code is used to add point indices to the variable *boundary*, where the input variable *Ind* is a three-dimensional array containing the point indices:

```
def add_to_boundary(Ind):
    if Ind[0] not in boundary:
        boundary[Ind[0]] = sc.SortedDict()
    if Ind[1] not in boundary[Ind[0]]:
        boundary[Ind[0]][Ind[1]] = sc.SortedList()
    boundary[Ind[0]][Ind[1]].add(Ind[2])
```

The subsequent code is used to check whether some point indices are already in *boundary*:

```
def already_checked(Ind):
    return Ind[0] in boundary and Ind[1] in boundary[Ind[0]] \
        and Ind[2] in boundary[Ind[0]][Ind[1]]
```

## 2.2.2 Location of $\Omega$

In the first step the algorithm tries to find every boundary part of  $\Omega$ . So, for every boundary part, the algorithm has to find at least one boundary point of this boundary part and add it to the dictionary *boundary*.

For the location of  $\Omega$  the following input parameters are important:

- *cuboidToFindOmega* =  $[[x_{min}, y_{min}, z_{min}], [x_{max}, y_{max}, z_{max}]]$ : The cuboid  $Q := [x_{min}, x_{max}] \times [y_{min}, y_{max}] \times [z_{min}, z_{max}]$  within which  $\Omega$  is located.
- *lengthXY* =  $l_{x,y}$ : Mesh size of  $G$  in  $x$ - and  $y$ -direction. To ensure exact calculation (of  $I_s(q)$ ,  $\gamma(\mathbf{z})$ , ...),  $l_{x,y}$  should be set as small as possible. But the evaluation time of the algorithm is dependent on  $l_{x,y}$ . Testing showed that the calculation takes  $O(\frac{1}{l_{x,y}^2})$  time, which makes sense because the boundary is 2-dimensional.
- *factorZrefinement*:  $l_z = lengthZ = l_{x,y}/factorZrefinement$  is the mesh size of  $G$  in  $z$ -direction. As a default it is set to 1 (same length in  $x$ -,  $y$ - and  $z$ -direction).
- *factorOmegaDetection*: Checking all points of  $G$  if they are in  $\Omega$  would take too much time! It is the only part of the algorithm that takes  $O(\frac{1}{l_{x,y}^3})$  time. With no further information on  $\Omega$  there is no better way to locate boundary parts of  $\Omega$  than checking all points and their neighbors, if one of them is inside and one of them is outside  $\Omega$ .  
Instead the location of  $\Omega$  can be made with a coarser grid. Only those  $[i, j, k]_G$  indices are checked where *factorOmegaDetection*  $\in \mathbb{N}$  divides  $i$ ,  $j$  and  $k$ , so only  $1/factorOmegaDetection^3$  grid points are checked in comparison with  $G$ .

A big downside of choosing *factorOmegaDetection* too large is if small parts of  $\Omega$  (when  $\Omega$  is small or consists of several small parts) or holes in  $\Omega$  are overlooked if no point of the much coarser grid lies within those parts/holes.

Now the following coarser grid is important:

$$\begin{aligned}
G_1 = \{ & [x_{min} + i l_{x,y} \textit{factorOmegaDetection}, \\
& y_{min} + j l_{x,y} \textit{factorOmegaDetection}, \\
& z_{min} + k l_z \textit{factorOmegaDetection}] \\
& \text{so that } i, j, k \in \mathbb{N}_0, x_{min} + i l_{x,y} \textit{factorOmegaDetection} \leq x_{max} \\
& y_{min} + j l_{x,y} \textit{factorOmegaDetection} \leq y_{max} \\
& z_{min} + k l_z \textit{factorOmegaDetection} \leq z_{max} \}
\end{aligned} \tag{32}$$

To **ensure that every boundary part of  $\Omega$  is detected**, a sufficient condition is that a cuboid of size  $[\textit{factorOmegaDetection} l_{x,y}, \textit{factorOmegaDetection} l_{x,y}, \textit{factorOmegaDetection} l_z]$  — which is exactly one cell of the  $G_1$  grid — can fit inside every component of  $\Omega$  and every component of any hole of  $\Omega$ .

The next step is that every two neighboring points on  $G_1$  are checked if one of the points  $\mathbf{x}_1$  is inside  $\Omega$  while the other one  $\mathbf{x}_2$  is outside  $\Omega$ . When  $\mathbf{x}_1$  is inside and  $\mathbf{x}_2$  is outside  $\Omega$ , the connecting line between them has to intersect at least one boundary part of  $\Omega$ . The following example demonstrates how a boundary point with respect to  $G$  is found by this method:

Example: Neighboring points differ in only one coordinate, so without loss of generality,  $\mathbf{x}_1 = [i, j, k]_{G_1}$  is inside and  $[i+1, j, k]_{G_1}$  is outside  $\Omega$ . It is obvious that these points also lie on  $G$  with some indices  $\mathbf{x}_1 = [i_1, j_1, k_1]_G$  and  $\mathbf{x}_2 = [i_1 + \textit{factorOmegaDetection}, j_1, k_1]_G$ . So there has to be an  $l \in \mathbb{N}_0, l \leq \textit{factorOmegaDetection} - 1$ , so that  $\mathbf{x}_3 := [i_1 + l, j_1, k_1]_G$  lies inside  $\Omega$  and  $[i_1 + l + 1, j_1, k_1]_G$  lies outside  $\Omega$ . So the boundary point  $\mathbf{x}_3$  of  $\Omega$  (with respect to  $G$ ) is found. Every boundary point indices found by the previous method are saved into *boundary* but also is appended to the list of indices *points\_to\_check*.

### 2.2.3 Find every boundary point of $\Omega$

The previous task was to find at least one boundary point (with respect to  $G$ ) on every boundary part of  $\Omega$ . These points were saved into the list of indices *points\_to\_check*. The current task is to find *every* boundary point (with respect to  $G$ ) based on the points in *points\_to\_check*.

We therefore have to state a condition on  $\Omega$ :

- If  $\mathbf{g}_1$  and  $\mathbf{g}_2$  are boundary points of the same boundary part of  $\Omega$ , there exists a sequence  $(\mathbf{g}_1 = \mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n = \mathbf{g}_2)$  of grid points so that  $\mathbf{p}_l, l = 1, \dots, n$  are boundary points with respect to  $G$  and all 2 consecutive points

$\mathbf{p}_l$  and  $\mathbf{p}_{l+1}, l = 1, \dots, (n-1)$  are nearby grid points (so each coordinate of two consecutive points only differs at most by 1).

The following code (which is explained afterwards) finds all boundary points of  $\Omega$ :

```
while (j < len(points_to_check)):
    ind0 = points_to_check[j]
    for i0 in [-1, 0, 1]:
        for j0 in [-1, 0, 1]:
            for k0 in [-1, 0, 1]:
                ind = ind0 + [i0, j0, k0]
                is_inside = phi_ind(xmin, lengthVect, ind) < 0
                if is_inside and not already_checked(ind, is_inside):
                    im00 = ind - [1, 0, 0]
                    ip00 = ind + [1, 0, 0]
                    i0m0 = ind - [0, 1, 0]
                    i0p0 = ind + [0, 1, 0]
                    i00m = ind - [0, 0, 1]
                    i00p = ind + [0, 0, 1]
                    bm00 = (phi_ind(xmin, lengthVect, im00) < 0)
                    bp00 = (phi_ind(xmin, lengthVect, ip00) < 0)
                    b0m0 = (phi_ind(xmin, lengthVect, i0m0) < 0)
                    b0p0 = (phi_ind(xmin, lengthVect, i0p0) < 0)
                    b00m = (phi_ind(xmin, lengthVect, i00m) < 0)
                    b00p = (phi_ind(xmin, lengthVect, i00p) < 0)
                    if not bm00 or not bp00 or not b0m0
                        or not b0p0 or not b00m or not b00p:
                        points_to_check.append(ind)
                        add_to_boundary(ind, is_inside)
```

For every point  $\mathbf{g} = \text{points\_to\_check}[j]$ , the following happens in the above *Python* code:

- $\text{ind0}$  are the grid indices of  $\mathbf{g}$
- Check every nearby grid point  $\mathbf{p}$  of  $\mathbf{g}$  (the 27 grid points around  $\mathbf{g}$ ), if it is inside  $\Omega$  and was not already added to *boundary* respectively to *points\_to\_check*.
- If the previous condition holds true for a nearby grid point  $\mathbf{p}$ , check every neighbor of  $\mathbf{p}$ , if one of them is outside  $\Omega$ . If this condition is also true,  $\mathbf{p}$  is a boundary point of  $\Omega$  with respect to  $G$ . So the indices of  $\mathbf{p}$  are added to *boundary* and *points\_to\_check*.

Now we have to check if the described algorithm really finds all boundary grid points of a boundary part of  $\Omega$ .

**Theorem 2.1.** *Let  $\mathbf{g}$  be a boundary grid point of  $\Omega$  and let the condition on  $\Omega$  we stated at the beginning of this section be met. Furthermore, for every boundary part of  $\Omega$ , at least one boundary grid point was added to the variable `points_to_check` beforehand. Then,  $\mathbf{g}$  is found by the algorithm stated above.*

*Proof.* Let  $\tilde{\mathbf{g}}$  be a boundary grid point of the same boundary part as  $\mathbf{g}$ , that was added to `points_to_check` beforehand. Then, the condition on  $\Omega$  at the beginning of this section states that there are boundary grid points  $\mathbf{p}_2, \dots, \mathbf{p}_{n-1}$ ,  $n \in \mathbb{N}$  so that for the sequence  $(\tilde{\mathbf{g}} = \mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n = \mathbf{g})$  it holds that  $\mathbf{p}_l$  and  $\mathbf{p}_{l+1}$ ,  $l = 1, \dots, (n-1)$  are nearby grid points.

Let  $\mathbf{p}_l = [i, j, k]_G$ ,  $l \in \{1, \dots, (n-1)\}$  be already found by the above algorithm and therefore also be added to `points_to_check`. And let it be the point that is currently checked by the algorithm, so  $ind0 = [i, j, k]$ .  $\mathbf{p}_l$  and  $\mathbf{p}_{l+1}$  are nearby grid points, so every of their coordinate values with respect to  $G$  differ only by 0 or 1, so there exist  $i_0, j_0, k_0 \in \{-1, 0, 1\}$  so that  $\mathbf{p}_{l+1} = [i + i_0, j + j_0, k + k_0]_G$ . When we analyze the algorithm we see that  $ind$  is set to  $[i + i_0, j + j_0, k + k_0]$  once. Then,  $is\_inside = True$ , because  $\mathbf{p}_{l+1} \in \Omega$ .

If  $ind = [i + i_0, j + j_0, k + k_0]$  was not already added to `points_to_check`, the 6 neighbors of  $\mathbf{p}_{l+1} = [i + i_0, j + j_0, k + k_0]_G$  are checked if one of them lies outside  $\Omega$ . This has to be true, because  $\mathbf{p}_{l+1}$  is a boundary point of  $\Omega$  with respect to  $G$ , and so at least one of its neighbors has to lie outside  $\Omega$ . Consequently,  $[i + i_0, j + j_0, k + k_0]$  is added to `points_to_check`.

This means,  $\mathbf{p}_{l+1}$  was either already added to `points_to_check` or is added now. We know that  $\mathbf{p}_1 = \tilde{\mathbf{g}}$  was added to `points_to_check` beforehand, so by induction we can conclude that for all  $l \in \{1, \dots, n\}$ ,  $\mathbf{p}_l$  are added to `points_to_check`. In particular,  $\mathbf{g} = \mathbf{p}_n$  is added to `points_to_check`. Furthermore, every point that is added to `points_to_check` is also added to the variable `boundary`.  $\square$

It is interesting to note is that in the previous loop, for every point in `points_to_check` it can happen that up to 27 new points are added to `points_to_check`. But only boundary points and their neighbors are added to `points_to_check`, and this only one time. Obviously the number of these points is bounded, so the loop will not go on forever.

#### 2.2.4 Post calculation of the boundary

At this step of the algorithm some important parameters of  $\Omega$  are calculated. Furthermore, a sparser representation of the variable `boundary` is created. For some calculations it is possible to use the sparser representation of `boundary` to save time.

The following parameters are calculated:

- $minXYZ$ ,  $maxXYZ$ : Every point in `boundary` is checked to find the minimum and maximum  $x$ -,  $y$ - and  $z$ -coordinates with respect to  $G$ .
- $midXYZ = np.floor((1.0/2)(minXYZ + maxXYZ))$ : The indices of a point on  $G$  that should be near the center of  $\Omega$ .  $(1.0/2)(minXYZ +$

$maxXYZ$ ) is exactly the mean of the maximum and minimum occurring indices on the boundary with respect to  $G$ .  $np.floor()$  rounds every coordinate down to the next smaller or equal integer - this ensures that  $midXYZ$  is again a point on  $G$ .

- *rangeXYZg*: This variable is set to  $maxXYZ - minXYZ + [2, 2, 2]$ . In the following Lemma 2.2 it is proved that every grid point inside  $\Omega$  also lies in the cuboid that is spanned by the coordinates  $midXYZ - rangeXYZg/2$  and  $midXYZ + rangeXYZg/2$ .
- *rangeXYZ*: This variable is set to  $rangeXYZg + [2, 2, 2]$ . The idea is that for a function  $f : \mathbb{R}^3 \rightarrow \mathbb{R}$  that is only non-zero in  $\Omega$ , let  $g : \mathbb{R}^3 \rightarrow \mathbb{R}$  be an interpolation of  $f$  so that  $f = g$  on every grid point. And for every grid cell,  $g$  is a linear combination of the values of  $f$  on the 8 corner points of the grid cell. Then it is obvious that the following statement holds true: If  $f$  is non-zero only on grid points within  $[x_1, y_1]_G \times [x_2, y_2]_G \times [x_3, y_3]_G$ , then  $g$  can only be non-zero on points within  $[x_1 - 1, y_1 + 1]_G \times [x_2 - 1, y_2 + 1]_G \times [x_3 - 1, y_3 + 1]_G$ .
- *midPointOmega*:  $midXYZ$  in space coordinates, not in grid coordinates with respect to  $G$ .
- *max\_radius*: Every point in *boundary* is checked and the maximum distance of those points to  $midXYZ$  is calculated, this value is multiplied by 2 and some small value *eps* is added. So it holds that all points of  $G$  that lie within  $\Omega$  are also inside the ball with midpoint *midPointOmega* and radius  $max\_radius/2$ .

The variable name “*max\_radius*” becomes clear in section 2.4 (page 50).

**Lemma 2.2.** Let  $\mathbf{l} = [l_1, l_2, l_3] := minXYZ$ ,  $\mathbf{u} = [u_1, u_2, u_3] := maxXYZ$ ,  $\mathbf{m} = [m_1, m_2, m_3] := midXYZ$  and  $\mathbf{r} = [r_1, r_2, r_3] = rangeXYZg$ .

Then, every grid point inside  $\Omega$  is contained in the cuboid

$$C = [m_1 - r_1/2, m_1 + r_1/2] \times [m_2 - r_2/2, m_2 + r_2/2] \times [m_3 - r_3/2, m_3 + r_3/2]$$

*Proof.* Let  $\mathbf{y} = [y_1, y_2, y_3]$  be a grid point inside  $\Omega$  and let  $i \in \{1, 2, 3\}$ , so it holds that

$$l_i \leq y_i \leq u_i.$$

Because  $m_i = \text{floor}(1/2(l_i + u_i))$  it holds that

$$(l_i + u_i)/2 - 1 \leq m_i \leq (l_i + u_i)/2.$$

Furthermore,

$$r_i = u_i - l_i + 2$$

and therefore

$$\begin{aligned} m_i - r_i/2 &\leq (l_i + u_i)/2 - (u_i - l_i + 2)/2 = l_i - 1 \leq l_i \leq y_i \\ \text{and } y_i &\leq u_i = ((l_i + u_i)/2 - 1) + (u_i - l_i + 2)/2 \leq m_i + r_i/2. \end{aligned}$$

This yields  $\mathbf{y} \in C$ . □



Furthermore, the variable *thinned\_out\_boundary* is created, which has the same structure as the variable *boundary*. It consists only of the points of *boundary* that are “boundary points in  $z$ -direction”. This means,  $[i, j, k]_G$  is element of *thinned\_out\_boundary* if and only if  $[i, j, k]_G$  is an element of *boundary* and either  $[i, j, k - 1]_G$  or  $[i, j, k + 1]_G$  are outside  $\Omega$ .

### 2.3 Calculate the autoconvolution $\gamma(\mathbf{z})$

With the use of the variables *boundary* and *thinned\_out\_boundary* this section describes how the algorithm calculates the autoconvolution  $\gamma(\mathbf{z})$  for a given  $\mathbf{z} \in \mathbb{R}^3$ . We recall that

$$\begin{aligned} \gamma(\mathbf{z}) &= \int_{\mathbb{R}^3} \rho(\mathbf{x}) \rho(\mathbf{x} - \mathbf{z}) d\mathbf{x} = \int_{\mathbb{R}^3} (c_\rho \mathbb{1}_\Omega(\mathbf{x})) (c_\rho \mathbb{1}_\Omega(\mathbf{x} - \mathbf{z})) d\mathbf{x} = \\ &= c_\rho^2 \int_{\Omega \cap (\Omega + \mathbf{z})} d\mathbf{x} = c_\rho^2 \text{vol}(\Omega \cap (\Omega + \mathbf{z})). \end{aligned} \quad (33)$$

Disregarding the constant charge density  $c_\rho$  the only thing to do is to calculate the volume of  $\Omega$  intersected by  $(\Omega + \mathbf{z})$ . We therefore identify the grid points lying in  $\Omega \cap (\Omega + \mathbf{z})$ , so we make the approximation

$$\text{vol}(\Omega \cap (\Omega + \mathbf{z})) \approx (\text{number of grid points in } \Omega \cap (\Omega + \mathbf{z})) \cdot (\text{volume of a cell}).$$

The main task in this section is to find the boundary of  $(\Omega + \mathbf{z})$  for  $\mathbf{z} \in \mathbb{R}^3$ .

#### 2.3.1 Find the boundary of $\Omega + \mathbf{z}, \mathbf{z} \in \mathbb{R}^3$

The first question that arises is how to get all boundary points of  $(\Omega + \mathbf{z})$  (with respect to  $G$ ) knowing the boundary of  $\Omega$  and the shift  $\mathbf{z}$ . If every coordinate of  $\mathbf{z}$  is a (integer) multiple of the grid lengths of  $G$ , i.e.  $\mathbf{z} \in \{[il_{x,y}, jl_{x,y}, kl_z] \mid i, j, k \in \mathbb{Z}\}$  then the boundary points of  $(\Omega + \mathbf{z})$  would simply be the boundary points of  $\Omega$  shifted by  $\mathbf{z}$ . But this is obviously not the general case and would be a too restrictive assumption.

Before we explain a strategy to find all boundary points of  $\Omega + \mathbf{z}$ , we have to introduce rounding functions with respect to  $G$  and some properties of them.

**Definition 2.6** (Round function with respect to  $G$ ). For  $\mathbf{z} = [z_1, z_2, z_3] \in \mathbb{R}^3$ , let the round function with respect to  $G$  be:

$$\text{round}_G(\mathbf{z}) := [\text{round}(z_1/l_{x,y}) l_{x,y}, \text{round}(z_2/l_{x,y}) l_{x,y}, \text{round}(z_3/l_z) l_z], \quad (34)$$

where  $\text{round}(\cdot)$  is the usual round function that rounds a real number to its nearest integer and rounds every number  $a = n + 1/2, n \in \mathbb{Z}$  up to  $\text{round}(a) = n + 1$ .

**Lemma 2.3.** For  $\mathbf{z} \in \mathbb{R}^3$ ,  $\mathbf{z}_r := \text{round}_G(\mathbf{z})$  is a grid point of  $G$ .

*Proof.* Because of assumption (27), there exist  $i, j, k \in \mathbb{Z}$  so that

$$[x_{min}, y_{min}, z_{min}] = [i l_{x,y}, j l_{x,y}, k l_z].$$

Setting  $i_0 := \text{round}(z_1/l_{x,y}) - i \in \mathbb{Z}$ ,  $j_0 := \text{round}(z_2/l_{x,y}) - j \in \mathbb{Z}$  and  $k_0 := \text{round}(z_3/l_z) + k \in \mathbb{Z}$  one can see that

$$\mathbf{z}_r = [x_{min} + i_0 l_{x,y}, y_{min} + j_0 l_{x,y}, z_{min} + k_0 l_z] \in G.$$

□

In the following lemma we make an estimate, how close the coordinates of  $\mathbf{z} \in \mathbb{R}^3$  and the coordinates of its rounded vector with respect to  $G$  are:

**Lemma 2.4.** *For  $\mathbf{z} = [z_1, z_2, z_3] \in \mathbb{R}^3$ , let  $\mathbf{z}_r = [z_{r,1}, z_{r,2}, z_{r,3}] := \text{round}_G(\mathbf{z})$ . Then it holds that*

$$\begin{aligned} |z_{r,1} - z_1| &\leq l_{x,y}/2, \\ |z_{r,2} - z_2| &\leq l_{x,y}/2, \\ |z_{r,3} - z_3| &\leq l_z/2. \end{aligned} \tag{35}$$

*Proof.*

$$\begin{aligned} |z_{r,1} - z_1| &= |\text{round}(z_1/l_{x,y}) l_{x,y} - (z_1/l_{x,y}) l_{x,y}| = \\ &= |\text{round}(z_1/l_{x,y}) - (z_1/l_{x,y})| l_{x,y} \leq (1/2) l_{x,y} = l_{x,y}/2. \end{aligned}$$

For the 2nd and 3rd coordinate of  $\mathbf{z}_r$  the proof is similar. □

**Definition 2.7** (Grid cell of  $G$ , Grid side of  $G$ ). A grid cell  $C$  of  $G$  is a cuboid

$$\begin{aligned} C &= [x_{min} + i l_{x,y}, x_{min} + (i+1) l_{x,y}] \times [y_{min} + j l_{x,y}, y_{min} + (j+1) l_{x,y}] \times \\ &\quad \times [z_{min} + k l_z, z_{min} + (k+1) l_z] \end{aligned}$$

with  $i, j, k \in \mathbb{Z}$ .

A set  $s \subset \mathbb{R}^3$  is a grid side of  $G$  if there exists a grid cell  $C$  of  $G$  so that  $s$  is one of the 6 faces of the cuboid  $C$ .

The grid cells are the unit cuboids of the grid  $G$  and the grid sides are the sides of these cuboids. It is clear that the 8 corner points of a grid cell are points in  $G$  and so the 4 corner points of a grid side  $s$  are also points in  $G$ .

It is also clear that the 8 grid cells that have  $\mathbf{g} = [g_1, g_2, g_3] \in G$  as a corner point are:

$$\{ [g_1 + (i-1) l_{x,y}, g_1 + i l_{x,y}] \times [g_2 + (j-1) l_{x,y}, g_2 + j l_{x,y}] \times [g_3 + (k-1) l_z, g_3 + k l_z] \mid i, j, k \in \{0, 1\} \}$$

One can see, that a grid side is shared by 2 grid cells which is also the intersection of those two grid cells.

**Definition 2.8** (Neighboring grid cells). The neighboring grid cells of a grid side  $s$  are the 2 grid cells that share  $s$  as a common boundary surface.

Later we shall need the fact, that if  $\mathbf{z} \in \mathbb{R}^3$  is a point inside or on the boundary of a grid cell  $C$  of  $G$ , it holds that  $\mathbf{z}_r := \text{round}_G(\mathbf{z})$  is a corner point of  $C$ .

**Lemma 2.5.** Let  $\mathbf{z} \in \mathbb{R}^3$  and let  $C$  be a grid cell of  $G$  so that  $\mathbf{z} \in C$ . Then,  $\mathbf{z}_r := \text{round}_G(\mathbf{z})$  is a corner point of  $C$ .

*Proof.* Because  $C$  is a grid cell of  $G$ , there are indices  $i_0, j_0, k_0 \in \mathbb{Z}$  so that

$$C = [x_{\min} + i_0 l_{x,y}, x_{\min} + (i_0 + 1) l_{x,y}] \times [y_{\min} + j_0 l_{x,y}, y_{\min} + (j_0 + 1) l_{x,y}] \times [z_{\min} + k_0 l_z, z_{\min} + (k_0 + 1) l_z].$$

Because  $\mathbf{z}_r$  is a grid point of  $G$ , there are indices  $i_1, i_2, i_3 \in \mathbb{Z}$  so that

$$\mathbf{z}_r = [x_{\min} + i_1 l_{x,y}, y_{\min} + j_1 l_{x,y}, z_{\min} + k_1 l_z].$$

By setting  $i_C := i_0 - i_1 + 1, j_C := j_0 - j_1 + 1, k_C := k_0 - k_1 + 1$  we can write

$$C = [z_{r,1} + (i_C - 1) l_{x,y}, z_{r,1} + i_C l_{x,y}] \times [z_{r,2} + (j_C - 1) l_{x,y}, z_{r,2} + j_C l_{x,y}] \times [z_{r,3} + (k_C - 1) l_z, z_{r,3} + k_C l_z].$$

Furthermore we know that the 8 grid cells that have  $\mathbf{z}_r$  as a corner point are:

$$C_{\mathbf{z}_r} := \{ [z_{r,1} + (i - 1) l_{x,y}, z_{r,1} + i l_{x,y}] \times [z_{r,2} + (j - 1) l_{x,y}, z_{r,2} + j l_{x,y}] \times [z_{r,3} + (k - 1) l_z, z_{r,3} + k l_z] \mid i, j, k \in \{0, 1\} \}.$$

Now suppose that  $\mathbf{z}_r = [z_{r,1}, z_{r,2}, z_{r,3}]$  is not a corner point of  $C$ , this means  $C \notin C_{\mathbf{z}_r}$ . So at least one of the indices  $i_C, j_C$  and  $k_C$  is neither 0 nor 1. Without loss of generality,  $i_C$  is neither 0 nor 1 and without loss of generality,  $i_C > 1$ , so  $i_C \geq 2$ .

Because  $z$  is an element of  $C$  it holds that

$$z_1 \geq z_{r,1} + (i_C - 1) l_{x,y} \geq z_{r,1} + l_{x,y} \Rightarrow |z_{r,1} - z_1| \geq l_{x,y}$$

But this is a contradiction to Lemma 2.4 which states that  $|z_{r,1} - z_1| \leq l_{x,y}/2$ . Therefore,  $\mathbf{z}_r$  has to be a corner point of  $C$ . □

Unfortunately, in later proofs we have to add up rounded vectors (with respect to  $G$ ). But there is one little problem that can be explained with the help of this example:

If  $a + b = 5, a, b \in \mathbb{R}$ , does it always hold true that  $\text{round}(a) + \text{round}(b) = 5$ ? No! E.g. for  $a = 3.5, b = 1.5$ ,  $\text{round}(a) + \text{round}(b) = 6$ .

It would be sufficient to round one of the summands up and the other down, then this statement becomes true.

**Lemma 2.6.** *Let the round function  $\text{round}(\cdot)$  be as stated in the Definition 2.6 and let the second round function be  $\text{round}_d(\cdot)$ , which also rounds every number to its nearest integer, but rounds every number  $a = n + 1/2, n \in \mathbb{Z}$  down to  $\text{round}_d(a) = n$ . Then for  $a, b \in \mathbb{R}, n \in \mathbb{Z}$  so that  $a + b = n$  it holds that*

$$\text{round}(a) + \text{round}_d(b) = n.$$

*Proof.* For the case that  $a, b \in \mathbb{Z}$  this statement is obviously true.

Let  $a \notin \mathbb{Z}$ . We know that there are unique  $n_1, n_2 \in \mathbb{Z}$  and unique  $r_1, r_2 \in [0, 1)$  so that  $a = n_1 + r_1$  and  $b = n_2 + r_2$ . Because  $a \notin \mathbb{Z}$  follows that  $r_1 \neq 0$ , therefore  $r_1 > 0$ . It holds that

$$r_1 + r_2 = a - n_1 + b - n_2 \in \mathbb{Z}$$

and

$$0 < r_1 + 0 \leq r_1 + r_2 < 1 + 1 = 2.$$

This yields

$$\begin{aligned} r_1 + r_2 &= 1, \\ n &= a + b = n_1 + r_1 + n_2 + r_2 = n_1 + n_2 + 1. \end{aligned}$$

Case 1,  $r_1 < 1/2$ : Here,  $r_2 = 1 - r_1 > 1/2$  and

$$\text{round}(a) + \text{round}_d(b) = n_1 + (n_2 + 1) = n.$$

Case 2,  $r_1 = 1/2$ : Here,  $a = n_1 + 1/2$ . It holds that

$$\text{round}(a) + \text{round}_d(b) = \text{round}(n_1 + 1/2) + \text{round}_d(n_2 + 1/2) = (n_1 + 1) + n_2 = n.$$

Case 3,  $r_1 > 1/2$ : Here,  $r_2 = 1 - r_1 < 1/2$  and

$$\text{round}(a) + \text{round}_d(b) = (n_1 + 1) + n_2 = n.$$

□

**Definition 2.9** (Second round function with respect to  $G$ ). For  $\mathbf{z} = [z_1, z_2, z_3] \in \mathbb{R}^3$ , let the second round function with respect to  $G$  be:

$$\text{round}_{d,G}(\mathbf{z}) := [\text{round}_d(z_1/l_{x,y})l_{x,y}, \text{round}_d(z_2/l_{x,y})l_{x,y}, \text{round}_d(z_3/l_z)l_z], \quad (36)$$

where  $\text{round}_d(\cdot)$  is the round function that rounds a real number to its nearest integer and rounds every number  $a = n + 1/2, n \in \mathbb{Z}$  down to  $\text{round}_d(a) = n$ .

When we check the proofs of Lemma 2.3, Lemma 2.4 and Lemma 2.5, we see that all those lemmas also hold true for  $\mathbf{z}_r = \text{round}_d(\mathbf{z})$  instead of  $\mathbf{z}_r = \text{round}(\mathbf{z})$ .

**Lemma 2.7.** *Let  $\mathbf{y} \in G$ ,  $\mathbf{x} = [x_1, x_2, x_3], \mathbf{z} = [z_1, z_2, z_3] \in \mathbb{R}^3$  so that  $\mathbf{x} + \mathbf{z} = \mathbf{y}$ . Then,*

$$\text{round}_G(\mathbf{x}) + \text{round}_{d,G}(\mathbf{z}) = \mathbf{y}.$$

*Proof.* At the equation (28) we see that for  $\mathbf{y} \in G$  there exist  $i, j, k \in \mathbb{Z}$  so that

$$\mathbf{y} = [i l_{x,y}, j l_{x,y}, k l_z].$$

Now, check every coordinate separately. For the first coordinate it holds that  $x_1 + z_1 = y_1$ , therefore

$$(x_1/l_{x,y}) + (z_1/l_{x,y}) = y_1/l_{x,y} = i \in \mathbb{Z}.$$

We can apply Lemma 2.6 to get

$$\text{round}(x_1/l_{x,y}) + \text{round}_d(z_1/l_{x,y}) = i$$

and consequently

$$\text{round}(x_1/l_{x,y}) l_{x,y} + \text{round}_d(z_1/l_{x,y}) l_{x,y} = i l_{x,y} = y_1.$$

Similarly, we can show that

$$\begin{aligned} \text{round}(x_2/l_{x,y}) l_{x,y} + \text{round}_d(z_2/l_{x,y}) l_{x,y} &= j l_{x,y} = y_2, \\ \text{round}(x_3/l_z) l_z + \text{round}_d(z_3/l_z) l_z &= k l_z = y_3 \end{aligned}$$

and therefore

$$\text{round}_G(\mathbf{x}) + \text{round}_{d,G}(\mathbf{z}) = \mathbf{y}.$$

□

We can define the 27 nearby grid points with respect to  $\mathbf{z} \in \mathbb{R}^3$  as follows:

**Definition 2.10** (Nearby grid points). For  $\mathbf{z} = [z_1, z_2, z_3] \in \mathbb{R}^3$ , let  $\mathbf{z}_r = [z_{r,1}, z_{r,2}, z_{r,3}] := \text{round}_{d,G}(\mathbf{z})$  be its rounded vector with respect to  $G$ . Then the 27 nearby grid points of  $\mathbf{z}$  we define as:

$$Z_r := \{\mathbf{z}_r + [i l_{x,y}, j l_{x,y}, k l_z] \mid i, j, k \in [-1, 0, 1]\}. \quad (37)$$

Now we can define our (first) strategy to find all boundary points of  $\Omega + \mathbf{z}$ : For every boundary point  $\mathbf{x}$  of  $\Omega$  with respect to  $G$ , check for every point  $\mathbf{y} = \mathbf{x} + \mathbf{p}$ ,  $\mathbf{p} \in Z_r$ , if  $\mathbf{y}$  is a boundary point of  $\Omega + \mathbf{z}$  with respect to  $G$ . So the following points have to be checked:

$$B_{\mathbf{z}+\Omega} := \{\mathbf{x} + Z_r \mid \mathbf{x} \text{ is a boundary grid point of } \Omega\}. \quad (38)$$

It is important to keep the "inverse" case in mind. Let  $\mathbf{y}$  be a boundary point of  $\Omega + \mathbf{z}$  with respect to  $G$ . Then  $\mathbf{y}$  is checked if at least one of the points  $\mathbf{y} - Z_r$  is a boundary point of  $\Omega$ .

In the usual case of running the algorithm it holds that  $l_z = l_{x,y}$ , the boundary has to be very "unsmooth" so that a grid point in  $\Omega + \mathbf{z}$  is not recognized, which means it is classified as a point outside  $\Omega + \mathbf{z}$ . The following 2D example shows a bad case scenario when a boundary point is not recognized:

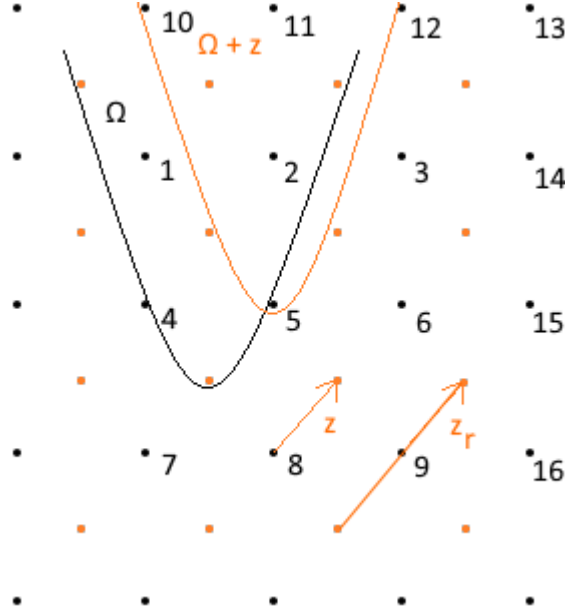


Figure 3: Example where a boundary point of  $\Omega + \mathbf{z}$  is not recognized

Here  $\mathbf{z}$  was set to  $[0.5l_{x,y}, 0.5l_{x,y}]$ , therefore the rounded  $\mathbf{z}_r$  is  $[l_{x,y}, l_{x,y}]$ . The points 4 and 5 are not inside  $\Omega$ , so the last boundary points of  $\Omega$  in downwards direction are the points 1 and 2. The points that are checked if they are in  $\Omega + \mathbf{z}$  because of the point 1 are the point 11 ((point 1) +  $\mathbf{z}_r$ ) and all 8 surrounding points (10, 12, 1, 2, 3,...). The points that are checked because of point 2 are the point 12 and all 8 surrounding points. So the point 5 is not checked if it is a boundary point of  $\Omega + \mathbf{z}$  and the algorithm classifies it as an outside point of  $\Omega + \mathbf{z}$ .

Before we state conditions on  $\Omega$  that ensure that we find every boundary point of  $\Omega + \mathbf{z}$  when we know all boundary points of  $\Omega$ , we consider a simpler case: Assuming we know all grid points that lie inside  $\Omega$ , how do we ensure to find all grid points that lie inside  $\Omega + \mathbf{z}$ ? Similar to the strategy explained at (38), the following grid points have to be checked if they lie inside  $\Omega + \mathbf{z}$ :

$$A_{\mathbf{z}+\Omega} := \{\mathbf{x} + \mathbf{z}_r \mid \mathbf{x} \in G \cap \Omega\}. \quad (39)$$

Now we can state the conditions that have to be met when we want to ensure that every grid point of  $(\Omega + \mathbf{z})$  is found by the strategy given at (39) when we know the grid points of  $\Omega$ :

- Whenever a boundary part of  $\Omega$  intersects a grid side  $s$  under the assumption that all corner points of  $s$  lie outside  $\Omega$ , the following has to hold (a

2d visualization follows):

- 1.) For one of the 2 neighboring grid cells  $C_1$  of  $s$ , a corner point of it has to lie inside  $\Omega$ . (40)

- 2.) Let  $\mathbf{m}_{C_2}$  be the midpoint of the other neighboring grid cell  $C_2$  and  $\mathbf{m}_s$  be the midpoint of  $s$ . Then  $\Omega$  is not allowed to intersect the rectangle  $s_m := s + (\mathbf{m}_{C_2} - \mathbf{m}_s)$  (which is a "bisectional rectangle of  $C_2$ "). (41)

• Moreover, it has to hold that:

- 3.) No simply connected part of  $\Omega$  lies inside only one grid cell. (42)

Remark: 1.) and 2.) can be seen as an upper bound to the curvature of  $\partial\Omega$  in relation to the grid width.

This is a 2d visualization of the conditions (40) and (41):

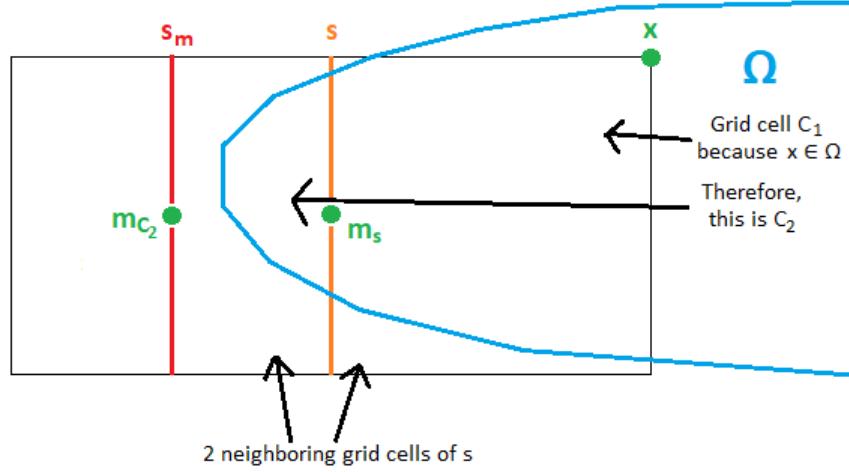


Figure 4: Visualization of (40) and (41)

In figure 4, the grid side  $s$  has to be investigated, because no corner point of  $s$  lies inside  $\Omega$ . Because one of the corners of the 2 neighboring grid cells lies inside  $\Omega$  (here:  $\mathbf{x} \in \Omega$ ), the first assumption is fulfilled. Moreover, we know that the right grid cell is  $C_1$ , the left grid cell is  $C_2$  and we can determine  $s_m$ , which is a "bisectional line of  $C_2$ " in 2d. Because  $\Omega$  does not intersect  $s_m$ , the second condition is also fulfilled.

Remark: The condition 2 states that  $\Omega$  is not allowed to reach "more than half" into the grid cell  $C_2$ .

Figure 3 shows a 2d example, where both corner points of the side between point 4 and point 5 are outside  $\Omega$  but the side is intersected by  $\Omega$ . The points 1 and 2 are points of one neighboring grid cell and lie inside  $\Omega$ , so the first of the previous conditions is fulfilled. But for the other neighboring grid cell with corner points 4, 5, 7 and 8,  $\Omega$  reaches more than half into the grid cell, so the second condition is not fulfilled - and it was possible to find an example where a point of  $\Omega + \mathbf{z}$  is not found.

**Lemma 2.8.** *Let  $\mathbf{z} = [z_1, z_2, z_3] \in \mathbb{R}^3$  and let  $\mathbf{y} \in G \cap (\Omega + \mathbf{z})$  be a grid point inside  $(\Omega + \mathbf{z})$ , let  $\mathbf{x} := \mathbf{y} - \mathbf{z} (\in \Omega)$  be inside a grid cell  $C$ , whereby no corner point of  $C$  lies inside  $\Omega$ . Furthermore the conditions (40), (41) and (42) have to be met.*

*Then, at least one grid side  $s_0$  of  $C$  is intersected by  $\Omega$  and also has  $\mathbf{x}_r := \text{round}_G(\mathbf{x}) = [x_{r,1}, x_{r,2}, x_{r,3}]$  as one of its corner points.*

*Proof.* Because of Lemma 2.5,  $\mathbf{x}_r$  is a corner point of  $C$ . We know that the 8 grid cells that have  $\mathbf{x}_r$  as a corner point are

$$C_{\mathbf{x}_r} := \{ [x_{r,1} + (i-1)l_{x,y}, x_{r,1} + i l_{x,y}] \times [x_{r,2} + (j-1)l_{x,y}, x_{r,2} + j l_{x,y}] \times [x_{r,3} + (k-1)l_z, x_{r,3} + k l_z] \mid i, j, k \in \{0, 1\} \}.$$

Without loss of generality, let  $C$  be

$$C = [x_{r,1}, x_{r,1} + l_{x,y}] \times [x_{r,2}, x_{r,2} + l_{x,y}] \times [x_{r,3}, x_{r,3} + l_z].$$

Then it is clear that 3 of the sides of  $C$  have  $\mathbf{x}_r$  as a corner point and the other 3 sides  $s_1, s_2$  and  $s_3$  do not have  $\mathbf{x}_r$  as a corner point, whereby  $s_1, s_2$  and  $s_3$  are the sets

$$\begin{aligned} s_1 &= \{ \mathbf{x}_r + [l_{x,y}, \beta l_{x,y}, \gamma l_z] \mid 0 \leq \beta, \gamma \leq 1 \}, \\ s_2 &= \{ \mathbf{x}_r + [\alpha l_{x,y}, l_{x,y}, \gamma l_z] \mid 0 \leq \alpha, \gamma \leq 1 \}, \\ s_3 &= \{ \mathbf{x}_r + [\alpha l_{x,y}, \beta l_{x,y}, l_z] \mid 0 \leq \alpha, \beta \leq 1 \}. \end{aligned}$$

Suppose, all 3 sides that have  $\mathbf{x}_r$  as a corner point are not intersected by  $\Omega$ . Because no simply connected part of  $\Omega$  lies inside only one grid cell, there has to be a path  $p$  from  $\mathbf{x} \in \Omega$  to the boundary of  $C$ , where  $p \subset C \cap \Omega$ . The starting point of  $p$  is  $\mathbf{x}$  and let the endpoint of  $p$  be  $\tilde{\mathbf{x}}$ . Because  $\Omega$  does not intersect the 3 sides that have  $\mathbf{x}_r$  as a corner point,  $\tilde{\mathbf{x}}$  has to lie on one of the sides  $s_1, s_2$  or  $s_3$ . Without loss of generality,  $\tilde{\mathbf{x}} \in s_1$ .

Because of Lemma 2.4, for the first coordinate of  $\mathbf{x}$ , it holds that

$$x_1 \leq x_{r,1} + l_{x,y}/2.$$



And because  $\tilde{\mathbf{x}} \in s_1$ , for the first coordinate of  $\tilde{\mathbf{x}}$  it holds that

$$\tilde{x}_1 = x_{r,1} + l_{x,y}.$$

Because the path  $p$  goes from  $\mathbf{x}$  to  $\tilde{\mathbf{x}}$  within  $C$  and within  $\Omega$ , there has to be a  $\hat{\mathbf{x}} = [\hat{x}_1, \hat{x}_2, \hat{x}_3] \in p \subset C \cap \Omega$  so that

$$\hat{x}_1 = x_{r,1} + l_{x,y}/2.$$

Because  $\hat{\mathbf{x}} \in C$  it holds that

$$\begin{aligned} x_{r,2} &\leq \hat{x}_2 \leq x_{r,2} + l_{x,y}, \\ x_{r,3} &\leq \hat{x}_3 \leq x_{r,3} + l_z. \end{aligned}$$

Because  $s_1$  is intersected by  $\Omega$ , either the condition (40) or (41) has to be met by  $C$ . Because (40) is not met, (41) has to be met, so  $\Omega$  is not allowed to intersect

$$\begin{aligned} s_m &= s_1 + (m_C - m_{s_1}) = s_1 + ([l_{x,y}/2, l_{x,y}/2, l_z/2] - [l_{x,y}, l_{x,y}/2, l_z/2]) = \\ &= s_1 + [-l_{x,y}/2, 0, 0] = \{\mathbf{x}_r + [l_{x,y}/2, j l_{x,y}, k l_z] \mid 0 \leq j, k \leq 1\}. \end{aligned}$$

When we check the coordinates of  $\hat{\mathbf{x}}$ , we see that  $\hat{\mathbf{x}} \in s_m$ . Because it also holds  $\hat{\mathbf{x}} \in \Omega$ ,  $\Omega$  intersects  $s_m$ , which is a contradiction to (41).

Therefore, the assumption that all 3 sides that have  $\mathbf{x}_r$  as a corner point are not intersected by  $\Omega$  cannot be true! So at least one grid side  $s_0$  of  $C$  is intersected by  $\Omega$  and also has  $\mathbf{x}_r := \text{round}_G(\mathbf{x})$  as one of its corner points.  $\square$

**Lemma 2.9.** *Let  $\mathbf{z} \in \mathbb{R}^3$ ,  $G_\Omega := \Omega \cap G$ ,  $G_{\Omega+\mathbf{z}} := (\Omega + \mathbf{z}) \cap G$  and  $Z_r$  the "grid points around  $\mathbf{z}$ " like described at (37). Furthermore, let  $\mathbf{y} \in (G_{\Omega+\mathbf{z}})$ ,  $\mathbf{x} := \mathbf{y} - \mathbf{z}$ ,  $\mathbf{x}_r = \text{round}_G(\mathbf{x})$ ,  $\mathbf{x}_1 \in G_\Omega$  and let  $C$  be a grid cell of  $\Omega$  where 2 of its corner points are  $\mathbf{x}_r$  and  $\mathbf{x}_1$ .*

*Then, proceeding from  $\mathbf{x}_1$  the strategy described at (39) finds  $\mathbf{y}$ .*

*Proof.* Let  $\mathbf{z}_r = \text{round}_{d,G}(\mathbf{z})$ . Because  $\mathbf{x} + \mathbf{z} = \mathbf{y}$  and  $\mathbf{y} \in G$  it follows from Lemma (2.7) that

$$\mathbf{x}_r + \mathbf{z}_r = \mathbf{y}.$$

$\mathbf{x}_1$  and  $\mathbf{x}_r$  are corner points of  $C$ , therefore there exist  $i_1, j_1, k_1 \in \{-1, 0, 1\}$  so that

$$\mathbf{x}_1 = \mathbf{x}_r + [i_1 l_{x,y}, j_1 l_{x,y}, k_1 l_z].$$

When we observe  $Z_r$ , we see that

$$\mathbf{z}_1 := \mathbf{z}_r - [i_1 l_{x,y}, j_1 l_{x,y}, k_1 l_z] \in Z_r.$$

So, proceeding from  $\mathbf{x}_1$  the strategy described at (39) finds  $\mathbf{y}$ , because

$$\mathbf{x}_1 + \mathbf{z}_1 = \mathbf{x}_r + [i_1 l_{x,y}, j_1 l_{x,y}, k_1 l_z] + \mathbf{z}_r - [i_1 l_{x,y}, j_1 l_{x,y}, k_1 l_z] = \mathbf{x}_r + \mathbf{z}_r = \mathbf{y}.$$

$\square$

**Theorem 2.10.** Let  $\mathbf{z} \in \mathbb{R}^3$ ,  $G_\Omega := \Omega \cap G$ ,  $G_{\Omega+\mathbf{z}} := (\Omega + \mathbf{z}) \cap G$  and  $Z_r$  the "grid points around  $\mathbf{z}$ " like described at (37). Furthermore the conditions (40), (41) and (42) hold true.

Then, every point in  $G_{\Omega+\mathbf{z}}$  is found by the strategy described at (39) applied to all grid points  $\mathbf{g} \in G_\Omega$ .

*Proof.* Let  $\mathbf{y} \in (G_{\Omega+\mathbf{z}})$ ,  $\mathbf{x} := \mathbf{y} - \mathbf{z} \in \Omega$ ,  $\mathbf{x}_r = \text{round}_G(\mathbf{x})$  and  $\mathbf{z}_r = \text{round}_{d,G}(\mathbf{z})$ . Because  $\mathbf{x} + \mathbf{z} = \mathbf{y}$  and  $\mathbf{y} \in G$  it follows from Lemma (2.7) that

$$\mathbf{x}_r + \mathbf{z}_r = \mathbf{y}.$$

Let  $C$  be a grid cell of  $G$  that contains  $\mathbf{x}$ . Lemma 2.5 states that  $\mathbf{x}_r = \text{round}_G(\mathbf{x})$  is a corner point of  $C$ . This means,  $C$  is one of the following cuboids:

$$C_{\mathbf{x}_r} := \{ [x_{r,1} + (i-1)l_{x,y}, x_{r,1} + i l_{x,y}] \times [x_{r,2} + (j-1)l_{x,y}, x_{r,2} + j l_{x,y}] \times [x_{r,3} + (k-1)l_z, x_{r,3} + k l_z] \mid i, j, k \in \{0, 1\} \}.$$

Without loss of generality, let  $C$  be

$$C = [x_{r,1}, x_{r,1} + l_{x,y}] \times [x_{r,2}, x_{r,2} + l_{x,y}] \times [x_{r,3}, x_{r,3} + l_z].$$

Now we consider 2 cases, depending on whether one of the corner points of  $C$  is inside  $\Omega$ .

**Case 1:** at least one of the corner points of  $C$  is inside  $\Omega$ . Then there exist  $i_0, j_0, k_0 \in \{0, 1\}$  so that

$$\mathbf{x}_0 := \mathbf{x}_r + [i_0 l_{x,y}, j_0 l_{x,y}, k_0 l_z] \in G_\Omega.$$

When we observe  $Z_r$ , we see that

$$\mathbf{z}_0 := \mathbf{z}_r - [i_0 l_{x,y}, j_0 l_{x,y}, k_0 l_z] \in Z_r.$$

So, proceeding from  $\mathbf{x}_0$  the strategy described at (39) finds  $\mathbf{y}$ , because

$$\mathbf{x}_0 + \mathbf{z}_0 = \mathbf{x}_r + [i_0 l_{x,y}, j_0 l_{x,y}, k_0 l_z] + \mathbf{z}_r - [i_0 l_{x,y}, j_0 l_{x,y}, k_0 l_z] = \mathbf{x}_r + \mathbf{z}_r = \mathbf{y}.$$

**Case 2:** None of the corner points of  $C$  is in  $\Omega$ . All requirements for Lemma 2.8 are satisfied, therefore at least one grid side  $s_0$  of  $C$  is intersected by  $\Omega$  and also has  $\mathbf{x}_r$  as one of its corner points. Because none of the corner points of  $C$  is in  $\Omega$ , this holds also true for  $s_0$ . Therefore, one of the neighboring grid cells  $C_2$  of  $s_0$  has to meet condition (40), so at least one of its corner points  $\mathbf{x}_1$  is in  $\Omega$ .

$\mathbf{x}_1 \in G_\Omega$  and  $\mathbf{x}_r$  are corner points of  $C_2$ , therefore we can apply Lemma 2.9 to see that proceeding from  $\mathbf{x}_1$ , the strategy described at (39) finds  $\mathbf{y}$ .  $\square$

Now we have shown that the strategy given at (39) to find all grid points inside  $\Omega + \mathbf{z}$  works, when the conditions (40), (41) and the condition that no simply connected part of  $\Omega$  lies inside only one grid cell hold true.

But we still have to find conditions, so that the strategy (38) that is used in the algorithm also works. It turns out that it is sufficient to state the same three conditions for  $\Omega^c := \mathbb{R}^3 \setminus \Omega$ .

**Theorem 2.11.** *Let  $\mathbf{z} \in \mathbb{R}^3$ ,  $G_\Omega := \Omega \cap G$ ,  $G_{\Omega+\mathbf{z}} := (\Omega + \mathbf{z}) \cap G$  and  $Z_r$  the "grid points around  $\mathbf{z}$ " like described at (37). Let the conditions (40), (41), and the condition that no simply connected part of  $\Omega$  lies inside only one grid cell hold true. Furthermore, let the conditions (40), (41) and (42) hold true for  $\Omega^c$ .*

*Then, every boundary grid point in  $G_{\Omega+\mathbf{z}}$  is found by the strategy described at (38) applied to all boundary grid points of  $\Omega$ .*

*Proof.* Let  $\mathbf{y}$  be a boundary grid point of  $\Omega + \mathbf{z}$ ,  $\mathbf{x} := \mathbf{y} - \mathbf{z} \in \Omega$ ,  $\mathbf{x}_r = \text{round}_G(\mathbf{x})$  and  $\mathbf{z}_r = \text{round}_{d,G}(\mathbf{z})$ . Because  $\mathbf{x} + \mathbf{z} = \mathbf{y}$  and  $\mathbf{y} \in G$  it follows from Lemma (2.7) that

$$\mathbf{x}_r + \mathbf{z}_r = \mathbf{y}.$$

Let  $C$  be a grid cell of  $G$  that contains  $\mathbf{x}$ . Lemma 2.5 states that  $\mathbf{x}_r = \text{round}_G(\mathbf{x})$  is a corner point of  $C$ . This means,  $C$  is one of the following cuboids:

$$C_{\mathbf{x}_r} := \{ [x_{r,1} + (i-1)l_{x,y}, x_{r,1} + i l_{x,y}] \times [x_{r,2} + (j-1)l_{x,y}, x_{r,2} + j l_{x,y}] \times \\ \times [x_{r,3} + (k-1)l_z, x_{r,3} + k l_z] \mid i, j, k \in \{0, 1\} \}.$$

Without loss of generality, let  $C$  be

$$C = [x_{r,1}, x_{r,1} + l_{x,y}] \times [x_{r,2}, x_{r,2} + l_{x,y}] \times [x_{r,3}, x_{r,3} + l_z].$$

Differently from Theorem 2.10 we now have to consider 3 cases:

1. No corner point of  $C$  lies inside  $\Omega$ .
2. At least one corner point of  $C$  lies inside  $\Omega$ , but not all corner points of  $C$  lie inside  $\Omega$ .
3. All corner points of  $C$  lie inside  $\Omega$ .

**Case 1:** No corner point of  $C$  lies inside  $\Omega$ . All requirements for Lemma 2.8 are satisfied, therefore at least one grid side  $s_0$  of  $C$  is intersected by  $\Omega$  and also has  $\mathbf{x}_r$  as one of its corner points.

Because none of the corner points of  $C$  is in  $\Omega$ , this holds also true for  $s_0$ . Therefore, one of the neighboring grid cells  $C_2$  of  $s_0$  has to meet condition (40), so at least one of its corner points  $\mathbf{x}_2$  is in  $\Omega$ .

$\mathbf{x}_2$  and  $\mathbf{x}_r$  are corner points of  $C_2$ , therefore there exist  $i_2, j_2, k_2 \in \{-1, 0, 1\}$  so that

$$\mathbf{x}_2 = \mathbf{x}_r + [i_2 l_{x,y}, j_2 l_{x,y}, k_2 l_z].$$

Let

$$\begin{aligned}\mathbf{x}_3 &= \mathbf{x}_r + [0, j_2 l_{x,y}, k_2 l_z], \\ \mathbf{x}_4 &= \mathbf{x}_r + [0, 0, k_2 l_z]\end{aligned}$$

and let

$$\mathbf{x}_1 = \begin{cases} \mathbf{x}_2 & \text{if } \mathbf{x}_3 \notin \Omega, \\ \mathbf{x}_3 & \text{else if } \mathbf{x}_4 \notin \Omega, \\ \mathbf{x}_4 & \text{else} \end{cases} \quad \text{and} \quad \mathbf{x}_0 = \begin{cases} \mathbf{x}_3 & \text{if } \mathbf{x}_3 \notin \Omega, \\ \mathbf{x}_4 & \text{else if } \mathbf{x}_4 \notin \Omega, \\ \mathbf{x}_r & \text{else.} \end{cases}$$

Then,  $\mathbf{x}_1 \in \Omega$  and  $\mathbf{x}_0 \notin \Omega$  and they are neighboring grid points on  $G$ , because they are only different in one coordinate and the difference is one corresponding grid length ( $l_{x,y}$  or  $l_z$ ). Therefore,  $\mathbf{x}_1$  is a boundary grid point of  $\Omega$  and it is a corner point of  $C$ , so there exist  $i_1, j_1, k_1 \in \{-1, 0, 1\}$  so that

$$\mathbf{x}_1 = \mathbf{x}_r + [i_1 l_{x,y}, j_1 l_{x,y}, k_1 l_z].$$

When we observe  $Z_r$ , we see that

$$\mathbf{z}_1 := \mathbf{z}_r - [i_1 l_{x,y}, j_1 l_{x,y}, k_1 l_z] \in Z_r.$$

So, proceeding from  $\mathbf{x}_1$  the strategy described at (38) finds  $\mathbf{y}$ , because

$$\mathbf{x}_1 + \mathbf{z}_1 = \mathbf{x}_r + [i_1 l_{x,y}, j_1 l_{x,y}, k_1 l_z] + \mathbf{z}_r - [i_1 l_{x,y}, j_1 l_{x,y}, k_1 l_z] = \mathbf{x}_r + \mathbf{z}_r = \mathbf{y}.$$

**Case 2:** At least one corner point of  $C$  lies inside  $\Omega$ , but not all corner points of  $C$  lie inside  $\Omega$ . So we have an  $\hat{\mathbf{x}}_2 \in \Omega$  and an  $\hat{\mathbf{x}}_3 \notin \Omega$  which are both corner points of  $C$ . Analogously to the proof for Case 1, we can find 2 neighboring grid points  $\hat{\mathbf{x}}_0 \notin \Omega$  and  $\hat{\mathbf{x}}_1 \in \Omega$  which are both on  $C$ . Therefore,  $\hat{\mathbf{x}}_1$  is a boundary grid point of  $\Omega$  and because it is a corner point of  $C$ , there exist  $i_3, j_3, k_3 \in \{-1, 0, 1\}$  so that

$$\hat{\mathbf{x}}_1 = \mathbf{x}_r + [i_3 l_{x,y}, j_3 l_{x,y}, k_3 l_z].$$

When we observe  $Z_r$ , we see that

$$\hat{\mathbf{z}}_1 := \mathbf{z}_r - [i_3 l_{x,y}, j_3 l_{x,y}, k_3 l_z] \in Z_r.$$

So, proceeding from  $\hat{\mathbf{x}}_1$  the strategy described at (38) finds  $\mathbf{y}$ , because

$$\hat{\mathbf{x}}_1 + \hat{\mathbf{z}}_1 = \mathbf{x}_r + [i_3 l_{x,y}, j_3 l_{x,y}, k_3 l_z] + \mathbf{z}_r - [i_3 l_{x,y}, j_3 l_{x,y}, k_3 l_z] = \mathbf{x}_r + \mathbf{z}_r = \mathbf{y}.$$

**Case 3:** All corner points of  $C$  lie inside  $\Omega$ . Because  $\mathbf{y}$  is a boundary grid point, a neighbor  $\tilde{\mathbf{y}}$  of  $\mathbf{y}$  has to lie outside  $\Omega$ .  $\tilde{\mathbf{y}}$  is only different from  $\mathbf{y}$  in one coordinate and the difference in this coordinate is exactly one corresponding grid length, so without loss of generality,

$$\tilde{\mathbf{y}} = \mathbf{y} + [l_{x,y}, 0, 0].$$

Let

$$\tilde{\mathbf{x}} := \tilde{\mathbf{y}} - \mathbf{z} = \mathbf{x} + [l_{x,y}, 0, 0],$$

so  $\tilde{\mathbf{x}}$  lies in the grid cell

$$\tilde{C} = [x_{r,1} + l_{x,y}, x_{r,1} + 2l_{x,y}] \times [x_{r,2}, x_{r,2} + l_{x,y}] \times [x_{r,3}, x_{r,3} + l_z].$$

Again, there are 2 cases whether at least one corner point of  $\tilde{C}$  lies outside of  $\Omega$ .

**Case 3.1** One corner point of  $\tilde{C}$  lies outside  $\Omega$ . Let this corner point be  $\tilde{\mathbf{x}}_0$ .  $\tilde{\mathbf{x}}_0$  does not have  $(x_{r,1} + l_{x,y})$  as its first coordinate, because then it would be also a corner point of  $C$  (and all corner points of  $C$  are inside  $\Omega$ ). Therefore, there exist  $j_4, k_4 \in \{0, 1\}$  so that

$$\tilde{\mathbf{x}}_0 = [x_{r,1} + 2l_{x,y}, x_{r,2} + j_4 l_{x,y}, x_{r,3} + k_4 l_z] \notin \Omega.$$

Let

$$\tilde{\mathbf{x}}_1 = [x_{r,1} + l_{x,y}, x_{r,2} + j_4 l_{x,y}, x_{r,3} + k_4 l_z],$$

then  $\tilde{\mathbf{x}}_1 \in \Omega$ , because it is a corner point of  $C$ .  $\tilde{\mathbf{x}}_1$  is also a neighbor of  $\tilde{\mathbf{x}}_0 \notin \Omega$  with respect to  $G$ , therefore  $\tilde{\mathbf{x}}_1$  is a boundary grid point of  $\Omega$ .

When we observe  $Z_r$ , we see that

$$\tilde{\mathbf{z}}_1 := \mathbf{z}_r - [l_{x,y}, j_4 l_{x,y}, k_4 l_z] \in Z_r.$$

So, proceeding from  $\tilde{\mathbf{x}}_1$  the strategy described at (38) finds  $\mathbf{y}$ , because

$$\tilde{\mathbf{x}}_1 + \tilde{\mathbf{z}}_1 = \mathbf{x}_r + [l_{x,y}, j_4 l_{x,y}, k_4 l_z] + \mathbf{z}_r - [l_{x,y}, j_4 l_{x,y}, k_4 l_z] = \mathbf{x}_r + \mathbf{z}_r = \mathbf{y}.$$

**Case 3.2** No corner point of  $\tilde{C}$  lies outside  $\Omega$ . So no corner point of  $\tilde{C}$  lies inside  $\Omega^c$ .

For  $\tilde{\mathbf{x}}_r = [\tilde{x}_{r,1}, \tilde{x}_{r,2}, \tilde{x}_{r,3}] := \text{round}_G(\tilde{\mathbf{x}})$  it holds that

$$\begin{aligned} \tilde{\mathbf{x}}_r &= \text{round}_G(\tilde{\mathbf{x}}) = \text{round}_G(\mathbf{x} + [l_{x,y}, 0, 0]) = \text{round}_G([x_1 + l_{x,y}, x_2, x_3]) = \\ &= [\text{round}((x_1 + l_{x,y})/l_{x,y}) l_{x,y}, \text{round}(x_2/l_{x,y}) l_{x,y}, \text{round}(x_3/l_z) l_z] = \\ &= [\text{round}((x_1/l_{x,y} + 1) l_{x,y}), \text{round}(x_2/l_{x,y}) l_{x,y}, \text{round}(x_3/l_z) l_z] = \\ &= [\text{round}((x_1/l_{x,y}) l_{x,y}), \text{round}(x_2/l_{x,y}) l_{x,y}, \text{round}(x_3/l_z) l_z] + [l_{x,y}, 0, 0] = \\ &= \mathbf{x}_r + [l_{x,y}, 0, 0]. \end{aligned}$$

Now we check if all requirements for Lemma 2.8 for  $\Omega^c$  instead of  $\Omega$  are met:

- $\tilde{\mathbf{y}}$  is a grid point inside  $(\Omega^c + \mathbf{z})$ ,
- $\tilde{\mathbf{x}} \in \Omega^c$  is inside a grid cell  $\tilde{C}$ ,
- No corner point of  $\tilde{C}$  lies inside  $\Omega^c$ ,

- (40), (41) and (42) are met for  $\Omega^c$ .

All those requirements are met, so at least one grid side  $\tilde{s}$  of  $\tilde{C}$  is intersected by  $\Omega^c$  and also has  $\tilde{\mathbf{x}}_r$  as one of its corner points.

Because none of the corner points of  $\tilde{C}$  is in  $\Omega^c$ , this holds also true for  $\tilde{s}$ . Therefore, one of the neighboring grid cells  $\tilde{C}_2$  of  $\tilde{s}$  has to meet condition (40), so at least one of its corner points  $\tilde{\mathbf{x}}_2$  is in  $\Omega^c$ , this means  $\tilde{\mathbf{x}}_2 \notin \Omega$ . Because  $\tilde{\mathbf{x}}_2$  and  $\tilde{\mathbf{x}}_r$  are corner points of  $\tilde{C}$ , there exist  $i_5, j_5, k_5 \in \{-1, 0, 1\}$  so that

$$\tilde{\mathbf{x}}_2 = [\tilde{x}_{r,1} + i_5 l_{x,y}, \tilde{x}_{r,2} + j_5 l_{x,y}, \tilde{x}_{r,3} + k_5 l_z].$$

Let

$$\begin{aligned}\tilde{\mathbf{x}}_3 &= [\tilde{x}_{r,1}, \tilde{x}_{r,2} + j_5 l_{x,y}, \tilde{x}_{r,3} + k_5 l_z], \\ \tilde{\mathbf{x}}_4 &= [\tilde{x}_{r,1}, \tilde{x}_{r,2}, \tilde{x}_{r,3} + k_5 l_z].\end{aligned}$$

and let

$$\tilde{\mathbf{x}}_5 = \begin{cases} \tilde{\mathbf{x}}_2 & \text{if } \tilde{\mathbf{x}}_3 \in \Omega, \\ \tilde{\mathbf{x}}_3 & \text{else if } \tilde{\mathbf{x}}_4 \in \Omega, \\ \tilde{\mathbf{x}}_4 & \text{else} \end{cases} \quad \text{and} \quad \tilde{\mathbf{x}}_6 = \begin{cases} \tilde{\mathbf{x}}_3 & \text{if } \tilde{\mathbf{x}}_3 \in \Omega, \\ \tilde{\mathbf{x}}_4 & \text{else if } \tilde{\mathbf{x}}_4 \in \Omega, \\ \tilde{\mathbf{x}}_r & \text{else.} \end{cases}$$

Then,  $\tilde{\mathbf{x}}_5 \notin \Omega$  and  $\tilde{\mathbf{x}}_6 \in \Omega$  and they are neighbors with respect to  $G$ . Therefore,  $\tilde{\mathbf{x}}_6 \in \Omega$  is a boundary grid point with respect to  $G$  and there exist  $j_6, k_6 \in \{-1, 0, 1\}$  so that

$$\tilde{\mathbf{x}}_6 = [\tilde{x}_{r,1}, \tilde{x}_{r,2} + j_6 l_{x,y}, \tilde{x}_{r,3} + k_6 l_z] = [x_{r,1} + l_{x,y}, x_{r,2} + j_6 l_{x,y}, x_{r,3} + k_6 l_z].$$

When we observe  $Z_r$ , we see that

$$\tilde{\mathbf{z}}_6 := \mathbf{z}_r - [l_{x,y}, j_6 l_{x,y}, k_6 l_z] \in Z_r.$$

So, proceeding from  $\tilde{\mathbf{x}}_6$  the strategy described at (38) finds  $\mathbf{y}$ , because

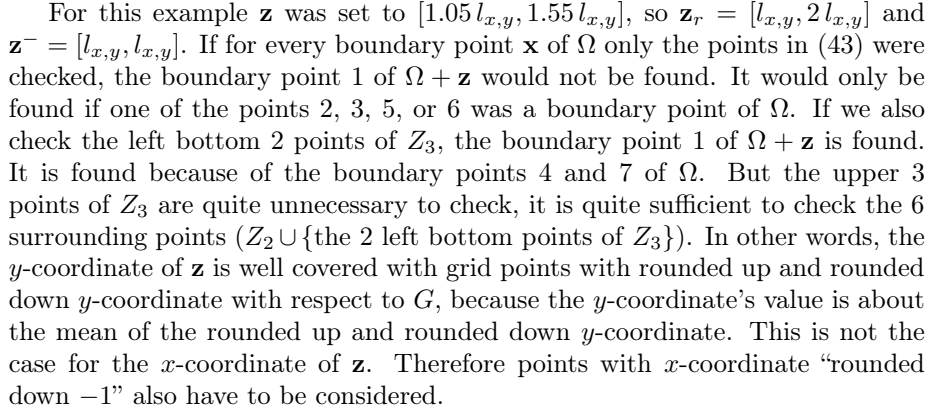
$$\tilde{\mathbf{x}}_6 + \tilde{\mathbf{z}}_6 = \mathbf{x}_r + [l_{x,y}, j_6 l_{x,y}, k_6 l_z] + \mathbf{z}_r - [l_{x,y}, j_6 l_{x,y}, k_6 l_z] = \mathbf{x}_r + \mathbf{z}_r = \mathbf{y}.$$

□

The downside of this approach is that for every boundary point of  $\Omega$ , 27 points have to be checked if they are boundary points of  $\Omega + \mathbf{z}$ . Because this part is a bottleneck of the algorithm, there was the incentive to find another approach that does not need so many checks at the cost of not letting the boundary to be that "unsmooth" without letting the algorithm make some mistakes. Let  $\mathbf{z}^-$  be  $\mathbf{z}$  rounded down to the next nearest multiple of the corresponding grid lengths of  $G$  (compare it with  $\mathbf{z}_r$ ), then for each boundary point  $\mathbf{x}$  of  $\Omega$  the 8 obvious points that always have to be checked are:

$$\begin{aligned}\mathbf{x} + \mathbf{p}, \quad \mathbf{p} \in Z_2 \text{ with} \\ Z_2 = \{\mathbf{z}^- + [i l_{x,y}, j l_{x,y}, k l_z] \mid i, j, k \in [0, 1]\}.\end{aligned} \tag{43}$$

The question that arises is whether the other  $27 - 8 = 19$  points  $Z_3 := Z_r \setminus Z_2$  necessarily have to be checked.  $Z_3$  are the remaining points on the other 7 grid cells that have  $\mathbf{z}_r$  as a corner point. The next example show  $\mathbf{z}_r$ ,  $\mathbf{z}^-$  and the points of  $Z_3$  and why it is necessary to sometimes check some of the points of  $Z_3$ :


$$l_{x,y} = l_z \quad \text{and} \quad l_G := l_{x,y} = l_z.$$

47

multiple of the grid length, but  $z_2 = 1.55l_G$  was not. So it was easy to construct an example where a boundary point of  $\Omega + \mathbf{z}$  are not recognized).

Therefore we will introduce the variable *boundarySmoothnessConst*, which can be set to a value between 0 and 1. If the value is 1, all 27 surrounding points  $Z$  are checked, and if the value is 0, only the 8 surrounding points  $Z_2$  are checked. Otherwise, introduce  $\mathbf{z}_\Delta := \text{abs}((1/l_G)(\mathbf{z} - \mathbf{z}_r))$ , which indicates how far each coordinate is away from the next grid coordinate. (For the last example  $\mathbf{z} = [1.05l_{x,y}, 1.55l_{x,y}]$ ,  $\mathbf{z}_\Delta$  would be  $[0.05, 0.45]$ ). Let  $z_{\Delta, \max} := \max(\mathbf{z}_\Delta)$  be the maximum coordinate of  $\mathbf{z}_\Delta$ . Then we have defined all variables needed to decide which points of  $Z_3$  should be considered.

Of importance are also the arrays *zxToCheck*, *zyToCheck* and *zzToCheck*, which are the grid indices around  $z$  that decide which points are checked. After filling those arrays, for every boundary point  $\mathbf{x}$  of  $\Omega$ , following points are checked:

$$\begin{aligned} & \mathbf{x} + \mathbf{p}, \quad \mathbf{p} \in Z_4 \text{ with} \\ Z_4 = \{[i l_G, j l_G, k l_G] \mid i \in \text{zxToCheck}, j \in \text{zyToCheck}, k \in \text{zzToCheck}\}. \end{aligned} \quad (44)$$

This is the code that fills the arrays *zxToCheck*, *zyToCheck* and *zzToCheck*:

```
zInd = np.multiply(lengthInvVect, z)
zRound = np.array(np.round(zInd)).astype(int)
zDeltaAbs = np.abs(zInd - zRound)
zDeltaMax = np.max(zDeltaAbs)

zxToCheck = np.array([zRound[0]])
zyToCheck = np.array([zRound[1]])
zzToCheck = np.array([zRound[2]])

if zDeltaMax != 0:
    if zInd[0] >= zRound[0] or zDeltaAbs[0]/zDeltaMax < boundarySmoothnessConst:
        zxToCheck = np.append(zxToCheck, zRound[0] + 1)
    if zInd[0] < zRound[0] or zDeltaAbs[0]/zDeltaMax < boundarySmoothnessConst:
        zxToCheck = np.append(zxToCheck, zRound[0] - 1)
    if zInd[1] >= zRound[1] or zDeltaAbs[1]/zDeltaMax < boundarySmoothnessConst:
        zyToCheck = np.append(zyToCheck, zRound[1] + 1)
    if zInd[1] < zRound[1] or zDeltaAbs[1]/zDeltaMax < boundarySmoothnessConst:
        zyToCheck = np.append(zyToCheck, zRound[1] - 1)
    if zInd[2] >= zRound[2] or zDeltaAbs[2]/zDeltaMax < boundarySmoothnessConst:
        zzToCheck = np.append(zzToCheck, zRound[2] + 1)
    if zInd[2] < zRound[2] or zDeltaAbs[2]/zDeltaMax < boundarySmoothnessConst:
        zzToCheck = np.append(zzToCheck, zRound[2] - 1)
```

There is the special case  $zDeltaMax == 0$ , which means that every coordinate of  $\mathbf{z}$  is a multiple of the grid length. In this case, the boundary of  $\Omega + \mathbf{z}$  is just every boundary point of  $\Omega$  with respect to  $G$  shifted by  $\mathbf{z}$ . Otherwise *zxToCheck* always contains the 2 grid coordinates nearest to the grid



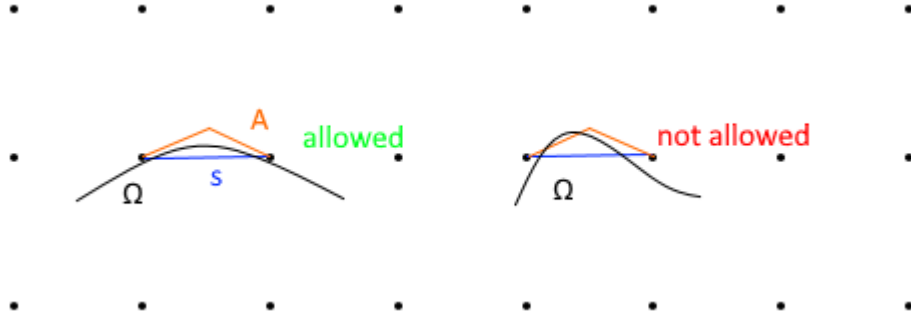
coordinates of  $zInd[0] = z_1$ . This means  $zxToCheck$  contains the coordinate  $zRound[0] = z_{r,1}$  and the neighboring coordinate of  $zRound[0]$  that is nearer to  $zInd[0]$ . This are the  $x$ -coordinates of  $Z_2$ . If the neighboring coordinate of  $zRound[0]$  that is further away from  $zInd[0]$  also is included decides the condition:

$$\frac{z_{\Delta,1}}{z_{\Delta,max}} < boundarySmoothnessConst. \quad (45)$$

The condition (45) tells us if the corresponding  $x$ -coordinate of  $Z_3$  also should be included. Obviously the checks are analog for  $zyToCheck$  and  $zzToCheck$  (only change the indices).

With the use of  $boundarySmoothnessConst$  and the grid  $G$  being cubical with grid lengths  $l_G$ , foreach grid cell where all corner points lie outside  $\Omega$ , the following holds: When the boundary of  $\Omega$  intersects one grid side  $s$  with corner points  $\mathbf{p}_1, \dots, \mathbf{p}_4$  (3-dim case) and normal vector  $\mathbf{n}_s$  of  $s$  pointing inside  $\Omega$ , then the same boundary part of  $\Omega$  is not allowed to intersect the set  $A := \{\mathbf{y} + \alpha(\mathbf{y}) \mathbf{n} \mid \mathbf{y} \in s, \alpha(\mathbf{y}) = \min\{\|\mathbf{p}_i - \mathbf{y}\|_2 \mid i = 1 \dots 4\} * boundarySmoothnessConst\}$ .

In the 2 dimensional setting and setting  $boundarySmoothnessConst := 1/4$ , the next illustration explains the previous statement:



### 2.3.2 Calculate the volume of $\Omega \cap (\Omega + \mathbf{z})$

To make the calculation of the volume of  $\Omega \cap (\Omega + \mathbf{z})$  as exact as possible, every grid point within this set is counted. Multiplied by the volume of one grid cell, this is an approximation of the real volume of  $\Omega \cap (\Omega + \mathbf{z})$ . Therefore, for every possible  $x$ - and  $y$ -coordinate with respect to the grid the following happens:

1. A *SortedList* with all possible  $z$ -coordinates of the boundary points of  $\Omega \cap (\Omega + \mathbf{z})$  is created.
2. These points (with  $z$ -coordinates in *SortedList*) and the spaces between those points are checked if they are inside  $\Omega \cap (\Omega + \mathbf{z})$ . The task here is to get the exact number of grid points lying in  $\Omega \cap (\Omega + \mathbf{z})$  for this  $x$ - and  $y$ -coordinate pair.

The first thing to check are the possible  $x$ - and  $y$ -coordinate ranges that all grid points that lie inside  $\Omega \cap (\Omega + \mathbf{z})$  can have. Therefore the possible  $x$ -coordinate ranges for  $\Omega$  and the possible  $x$ -coordinate ranges for  $(\Omega + \mathbf{z})$  are

attained. Then the two attained ranges are intersected to get the  $x$ -coordinate range of  $\Omega \cap (\Omega + \mathbf{z})$ . The analog thing is done to get the  $y$ -coordinate range (and also the  $z$ -coordinate range) of  $\Omega \cap (\Omega + \mathbf{z})$ .

For each of the possible  $x$ - and  $y$ -coordinates the next task is to get possible boundary points of  $\Omega \cap (\Omega + \mathbf{z})$  in  $z$ -direction. Candidates for this are the boundary points of  $\Omega$  in  $z$ -direction and the possible boundary points of  $\Omega + \mathbf{z}$  in  $z$ -direction. With the use of those points count the grid points with this  $x$ - and  $y$ -coordinate lying in  $\Omega \cap (\Omega + \mathbf{z})$ . For a fixed  $x$ -coordinate  $y_1$  and a fixed  $y$ -coordinate  $y_2$  the following is done:

1. use the values *thinned\_out\_boundary* $[y_1][y_2]$  to get the boundary of  $\Omega$  in  $z$ -direction and add them to the *SortedList zCoordsToCheck*.
2.  $y_1$  and  $y_2$  are fixed, so the task is to get all  $z$ -coordinates  $y_3$  that are candidates for the boundary points of  $\Omega + \mathbf{z}$ . Because each coordinate of  $\mathbf{z}$  does not have to be a multiple of the grid length, several  $\tilde{\mathbf{z}}, \tilde{\mathbf{z}} \in Z_4$  around  $\mathbf{z}$  have to be checked to get all boundary points of  $\Omega + \mathbf{z}$ . ( $Z_4$  is defined at (44) in the last chapter 2.3.1 and is described by the arrays *zxToCheck*, *zyToCheck* and *zzToCheck*.) So a  $z$ -coordinate  $y_3$  with the corresponding grid point  $\mathbf{y} = [y_1, y_2, y_3]$  has to be checked, if one of the points  $\mathbf{y} - \tilde{\mathbf{z}}, \tilde{\mathbf{z}} \in Z_4$  is a boundary point of  $\Omega$ . This means:  $y_3$  has to be checked if and only if there exists a  $\tilde{\mathbf{z}} \in Z_4$  so that  $y_3 - \tilde{z}_3 \in \text{boundary}[y_1 - \tilde{z}_1][y_2 - \tilde{z}_2]$ . So, for every  $\tilde{\mathbf{z}} \in Z_4$  the  $z$ -coordinates  $\text{boundary}[y_1 - \tilde{z}_1][y_2 - \tilde{z}_2] + \tilde{z}_3$  are added to the *SortedList zCoordsToCheck* (only add a  $z$ -coordinate if it is was not already added to *zCoordsToCheck*).
3. For each attained  $z$ -coordinate  $y_3$  in *zCoordsToCheck* with corresponding point  $\mathbf{y} = [y_1, y_2, y_3]$  then do the following:
  - If  $\mathbf{y}$  is inside  $\Omega \cap (\Omega + \mathbf{z})$ , the volume of 1 grid cell is added to the total volume.
  - And for each consecutive 2 points in the list that are inside  $\Omega \cap (\Omega + \mathbf{z})$  which have a grid point between them that is also inside  $\Omega \cap (\Omega + \mathbf{z})$ , the number of grid points between those 2 points are counted. The volume of that many grid cells are also added to the total volume. This is the volume in the "inside" of  $\Omega \cap (\Omega + \mathbf{z})$ .

## 2.4 Calculate $I_s(q)$

To calculate  $I_s(q)$ , we use the representation we derived in (6):

$$I_s(q) = \int_0^{\tilde{R}} \frac{r}{q} \sin(rq) \int_{\mathbf{p} \in S^2} \gamma(r\mathbf{p}) dS(\mathbf{p}) dr \quad (46)$$

with  $\tilde{R} \geq \sup_{\mathbf{x}, \mathbf{y} \in \Omega} \|\mathbf{x} - \mathbf{y}\|_2$ .

The first task is to find an  $\tilde{R} \in \mathbb{R}$  that satisfies the condition in (46), whereby  $\tilde{R}$  should not be too large to avoid integrating over too much space where we know the value of the integral is 0. We can use the variables *midPointOmega* and *max\_radius* we calculated in the post calculation of the boundary (section 2.2.4, page 31).

**Lemma 2.12.** *For  $\tilde{R} := \text{max\_radius}$  (*max\_radius* defined in section 2.2.4), it holds that*

$$\tilde{R} \geq \sup_{\mathbf{x}, \mathbf{y} \in \Omega} \|\mathbf{x} - \mathbf{y}\|_2 = \text{diam}(\Omega).$$

*Proof.* Let  $\mathbf{m}_\Omega := \text{midPointOmega}$  (also defined in section 2.2.4), then *max\_radius* is defined in a way (see section section 2.2.4) so that

$$\Omega \subset B_{(\tilde{R}/2)}(\mathbf{m}_\Omega).$$

So for all  $\mathbf{x}, \mathbf{y} \in \Omega$  it holds that

$$\tilde{R} = \tilde{R}/2 + \tilde{R}/2 > \|\mathbf{x} - \mathbf{m}_\Omega\|_2 + \|\mathbf{y} - \mathbf{m}_\Omega\|_2 \geq \|\mathbf{x} - \mathbf{y}\|_2,$$

and therefore

$$\tilde{R} \geq \sup_{\mathbf{x}, \mathbf{y} \in \Omega} \|\mathbf{x} - \mathbf{y}\|_2.$$

□

So a valid (and in practice good enough) choice for  $\tilde{R}$  is:

$$\tilde{R} := \text{max\_radius}. \quad (47)$$

Remark: The variable name *max\_radius* was chosen, because it is our choice for the upper bound for the integral (over  $dr$ ) given at (46).

To evaluate the integral (46), a numerical integration method is applied. The idea is to replace the integral by a finite sum. For  $\int_a^b f(x) dx$  the task is to find points  $x_1, \dots, x_n \in [a, b]$  and weights  $w_1, \dots, w_n \in \mathbb{R}$  so that  $\sum_{k=1}^n w_k f(x_k)$  with  $\sum_{k=1}^n w_k = b - a$  is a good approximation to  $\int_a^b f(x) dx$ . If  $f(x)$  can be approximated reasonably well by polynomials, the usual choice (and also our choice) is to take the points and weights received by the Gaussian quadrature rule. The Gaussian quadrature rule is well described in the book *Numerik-Algorithmen, Auflage 10*, section 14.7, p. 597 by Gisela Engeln-Müllges, Klaus Niederdrenk and Reinhard Wodicka [4]. Moreover, a table for the points and weights for different  $n \in \mathbb{N}$  is given at the *Handbook of Mathematical Functions* by Milton Abramowitz and Irene Stegun, p. 916 to p. 920, table 25.4 [11]. If the domain of integration is given as  $[-1, 1]$ , the following holds for the  $n$ -point Gaussian quadrature rule with  $n \in \mathbb{N}$ :

- There are Gaussian points  $x_1, \dots, x_n \in [-1, 1]$  with weights  $w_1, \dots, w_n \in \mathbb{R}$ , so that all polynomials  $p(x)$  of order  $(2n - 1)$  or less are integrated exactly by the formula  $\int_{-1}^1 p(x) dx = \sum_{k=1}^n w_k p(x_k)$ .
- The weights  $w_1, \dots, w_n$  are positive.
- Applying the Gaussian quadrature rule to an integral  $\int_a^b f(x) dx$  over  $[a, b]$  gives:  

$$\int_a^b f(x) dx \approx \frac{b-a}{2} \sum_{i=1}^n w_i f\left(\frac{b-a}{2} x_i + \frac{a+b}{2}\right)$$

To integrate over the unit sphere in  $\mathbb{R}^3$ , yet another numerical integration method also is applied. A good choice is the Lebedev quadrature which is only available for some  $n \in \mathbb{N}$  (e.g. 6, 14, 26, 38, 50, 74, 86, ...). The Lebedev quadrature rule is described in the book *Spherical harmonics and approximations on the unit sphere: an introduction* by Atkinson Kendal and Han Weimin [9]. The Lebedev quadrature of order  $n$  constructs  $n$  points  $\mathbf{p}_1, \dots, \mathbf{p}_n$  on the unit sphere with corresponding weights  $w_1, \dots, w_n \in \mathbb{R}$ . The weights are non-negative and add up to 1. Having found the quadrature points and weights, we get

$$\int_{\mathbf{p} \in S^2} f(\mathbf{p}) dS(\mathbf{p}) \approx \sum_{i=1}^n w_i f(\mathbf{p}_i).$$

For fixed  $n$ , this approximation is exact for all spherical harmonics up to some order  $\tilde{n}(n)$  [9].

The variable `nrOfGaussPoints` determines the order of the Gaussian quadrature rule for the outer integral of (46) and the variable `nrOfSpherePoints` determines the order of the Lebedev quadrature rule for the inner integral over the unit sphere in (46). Let  $n := \text{nrOfGaussPoints}$ ,  $m := \text{nrOfSpherePoints}$ , points and weights for the Gaussian quadrature be  $(x_1, w_1), \dots, (x_n, w_n)$ , points and weights for the Lebedev quadrature be  $(\mathbf{p}_1, v_1), \dots, (\mathbf{p}_m, v_m)$  and the argument of the outer integral be  $f(r)$ , then the chosen approximation for  $I_s(q)$  is given by

$$\begin{aligned}
I_s(q) &= \int_0^R \frac{r}{q} \sin(rq) \int_{\mathbf{p} \in S^2} \gamma(r\mathbf{p}) dS(\mathbf{p}) dr \approx \frac{R}{2} \sum_{i=1}^n w_i f\left(\frac{(x_i+1)R}{2}\right) = \\
&= \frac{R}{2} \sum_{i=1}^n w_i \frac{(x_i+1)R}{2q} \sin\left(\frac{(x_i+1)Rq}{2}\right) \int_{\mathbf{p} \in S^2} \gamma\left(\frac{(x_i+1)R}{2}\mathbf{p}\right) dS(\mathbf{p}) \approx \\
&\approx \frac{R}{2} \sum_{i=1}^n w_i \frac{(x_i+1)R}{2q} \sin\left(\frac{(x_i+1)Rq}{2}\right) \sum_{j=1}^m v_j \gamma\left(\frac{(x_i+1)R}{2}\mathbf{p}_j\right).
\end{aligned} \tag{48}$$

## 2.5 Calculate $F(\mathbf{x}, q)$ for $\mathbf{x} \in \partial\Omega$

For this section, let  $\mathbf{x} = [x_1, x_2, x_3]$  and  $q$  be fixed. Recall (see section 2.1) that the quantity

$$F(\mathbf{x}, q) := 2c_\rho^2 \int_{\Omega} \text{sinc}(q\|\mathbf{z} - \mathbf{x}\|) d\mathbf{z}$$

needs to be evaluated to determine the shape derivative of the intensity function.

For the numerical integration, we have to rearrange the integral

$$\begin{aligned}
\int_{\Omega} \text{sinc}(q\|\mathbf{z} - \mathbf{x}\|) d\mathbf{z} &= \int_{\mathbb{R}^3} \mathbb{1}_{\Omega}(\mathbf{z}) \text{sinc}(q\|\mathbf{z} - \mathbf{x}\|) d\mathbf{z} = \\
&= \int_{\mathbb{R}} \int_{\mathbb{R}} \int_{\mathbb{R}} \mathbb{1}_{\Omega}([z_1, z_2, z_3]) \text{sinc}(q\|[z_1, z_2, z_3] - \mathbf{x}\|) dz_3 dz_2 dz_1.
\end{aligned} \tag{49}$$

These are the approximations we will use to obtain an approximate result of the triple integral given at (49):

1. Instead of integrating over  $\Omega$  we integrate over cuboids, whereby the center points of those cuboids are grid points of  $G$ . The size of those cuboids have the same size as the grid cells of  $G$ . We only integrate over those cuboids where the center point is inside  $\Omega$ . We see (w.o. proof) that the union set of those cuboids is the set  $\Omega_G$  given in definition 2.11.
2. Then we define the inner most integral in (49) as the function  $\hat{F}(z_1, z_2)$  (see equation (52)). For  $z_1 = i l_{x,y} + x_{min}, z_2 = j l_{x,y} + y_{min}, i, j \in \mathbb{Z}$ , we continue to derive another representation of  $\hat{F}(z_1, z_2)$ . We shall see the result in equation (56).
3. Afterwards, we analyze the 1-dimensional integrals in equation (56) and derive a minimum number of Gauss integration points to numerically integrate  $\hat{F}(z_1, z_2)$  in  $z_3$ -direction.

4. When we only consider the  $z_1$ - and  $z_2$ -indices of the grid  $G$ , we get 2d-grid with the grid points

$$G_{x,y} := \{[z_1, z_2] | z_1 = i l_{x,y} + x_{min}, z_2 = j l_{x,y} + y_{min}, i, j \in \mathbb{Z}\}.$$

On the grid points of  $G_{x,y}$  we know the representation (??) of  $\hat{F}(z_1, z_2)$ . Inside every 2d grid cell of  $G_{x,y}$ , we use the bilinear interpolation  $\hat{P}(z_1, z_2)$  to approximate  $\hat{F}(z_1, z_2)$ .

5. Finally, we use 2d Gaussian interpolation to numerically integrate

$$\int_{\mathbb{R}} \int_{\mathbb{R}} \hat{P}(z_1, z_2) dz_2 dz_1$$

As mentioned previously, we first define an “approximation domain”  $\Omega_G$  of  $\Omega$ . With the help of  $\Omega_G$ , it will be possible to check if a point lies inside  $\Omega_G$  by only using the variable *boundary* (or *thinned\_out\_boundary*).

**Definition 2.11.** Let the approximation domain of  $\Omega$  with respect to  $G$  be

$$\Omega_G := \{\mathbf{x} \in \mathbb{R}^3 \mid \text{round}_G(\mathbf{x}) \in \Omega\}. \quad (50)$$

Instead of the representation given in (49), we rearrange the integral in the following way:

$$\begin{aligned} \int_{\Omega} \text{sinc}(q\|\mathbf{z} - \mathbf{x}\|) d\mathbf{z} &\approx \int_{\Omega_G} \text{sinc}(q\|\mathbf{z} - \mathbf{x}\|) d\mathbf{z} = \int_{\mathbb{R}^3} \mathbb{1}_{\Omega_G}(\mathbf{z}) \text{sinc}(q\|\mathbf{z} - \mathbf{x}\|) d\mathbf{z} = \\ &= \int_{\mathbb{R}} \int_{\mathbb{R}} \int_{\mathbb{R}} \mathbb{1}_{\Omega_G}([z_1, z_2, z_3]) \text{sinc}(q\|[z_1, z_2, z_3] - \mathbf{x}\|) dz_3 dz_2 dz_1. \end{aligned} \quad (51)$$

If we define  $\hat{F} : \mathbb{R}^2 \rightarrow \mathbb{R}$  as

$$\hat{F}(z_1, z_2) = \int_{\mathbb{R}} \mathbb{1}_{\Omega_G}([z_1, z_2, z_3]) \text{sinc}(q\|[z_1, z_2, z_3] - \mathbf{x}\|) dz_3, \quad (52)$$

then equation (51) can be rewritten as

$$\int_{\Omega_G} \text{sinc}(q\|\mathbf{z} - \mathbf{x}\|) d\mathbf{z} = \int_{\mathbb{R}} \int_{\mathbb{R}} \hat{F}(z_1, z_2) dz_2 dz_1.$$

Our next step is the transformation of the integral given in (52), but only for  $z_1$  and  $z_2$  being multiples of the corresponding grid lengths.

### 2.5.1 Transformation of $\hat{F}(z_1, z_2)$

**Theorem 2.13.** *Let  $z_1 = i l_{x,y} + x_{min}$ ,  $z_2 = j l_{x,y} + y_{min}$ ,  $i, j \in \mathbb{Z}$ . Then it holds that*

$$\{[z_1, z_2, z_3] \mid z_3 \in \mathbb{R}\} \cap \Omega_G = \bigcup_{k \in \mathbb{Z}, [i,j,k]_G \in \Omega} \{[i, j, k + q]_G \mid q \in [-\frac{1}{2}, \frac{1}{2})\}.$$

Moreover, for  $k_1, k_2 \in \mathbb{Z}$  so that  $k_1 \neq k_2$  it holds that

$$\{[i, j, k_1 + q]_G \mid q \in [-\frac{1}{2}, \frac{1}{2})\} \cap \{[i, j, k_2 + q]_G \mid q \in [-\frac{1}{2}, \frac{1}{2})\} = \emptyset.$$

*Proof.* Let  $x_{min} = i_0 l_{x,y}$ ,  $y_{min} = j_0 l_{x,y}$  and  $z_{min} = k_0 l_{x,y}$ ,  $i_0, j_0, k_0 \in \mathbb{Z}$ .

First, we show that

$$\{[z_1, z_2, z_3] \mid z_3 \in \mathbb{R}\} \cap \Omega_G \subset \bigcup_{k \in \mathbb{Z}, [i,j,k]_G \in \Omega} \{[i, j, k + q]_G \mid q \in [-\frac{1}{2}, \frac{1}{2})\}.$$

Let  $\mathbf{z} \in \{[z_1, z_2, z_3] \mid z_3 \in \mathbb{R}\} \cap \Omega_G$ , so  $z_1 = i l_{x,y} + x_{min}$ ,  $z_2 = j l_{x,y} + y_{min}$  and  $\mathbf{z}_r := \text{round}_G(\mathbf{z}) \in \Omega$ .

With  $k := \text{round}((z_3 - z_{min})/l_z)$  it holds that

$$\begin{aligned} \text{round}(z_3/l_z) l_z &= \text{round}(z_3/l_z) l_z - k_0 l_z + k_0 l_z = \\ &= \text{round}(z_3/l_z - k_0) l_z + z_{min} = \\ &= \text{round}((z_3 - k_0 l_z)/l_z) l_z + z_{min} = \\ &= \text{round}((z_3 - z_{min})/l_z) l_z + z_{min} = k l_z + z_{min}. \end{aligned}$$

Furthermore, it holds that

$$\text{round}(z_1/l_{x,y}) l_{x,y} = \text{round}((i + i_0) l_{x,y}/l_{x,y}) l_{x,y} = (i + i_0) l_{x,y} = i l_{x,y} + x_{min}$$

and analogously,

$$\text{round}(z_2/l_{x,y}) l_{x,y} = \text{round}((j + j_0) l_{x,y}/l_{x,y}) l_{x,y} = (j + j_0) l_{x,y} = j l_{x,y} + y_{min}.$$

Therefore,

$$\begin{aligned} \mathbf{z}_r &= [\text{round}(z_1/l_{x,y}) l_{x,y}, \text{round}(z_2/l_{x,y}) l_{x,y}, \text{round}(z_3/l_z) l_z] = \\ &= [i l_{x,y} + x_{min}, j l_{x,y} + y_{min}, k l_z + z_{min}] = [i, j, k]_G \in \Omega. \end{aligned}$$

Let  $q := (z_3 - z_{min})/l_z - k = (z_3 - z_{min})/l_z - \text{round}((z_3 - z_{min})/l_z)$ , then  $q \in [-\frac{1}{2}, \frac{1}{2})$  and

$$z_{min} + (k + q) l_z = z_{min} + (k + (z_3 - z_{min})/l_z - k) l_z = z_3.$$

It follows that

$$[i, j, k + q]_G = [x_{min} + i l_{x,y}, y_{min} + j l_{x,y}, z_{min} + (k + q) l_z] = [z_1, z_2, z_3]$$

and therefore

$$\mathbf{z} \in \bigcup_{k \in \mathbb{Z}, [i, j, k]_G \in \Omega} \{[i, j, k + q]_G \mid q \in [-\frac{1}{2}, \frac{1}{2})\}.$$

Now, we show that

$$\bigcup_{k \in \mathbb{Z}, [i, j, k]_G \in \Omega} \{[i, j, k + q]_G \mid q \in [-\frac{1}{2}, \frac{1}{2})\} \subset \{[z_1, z_2, z_3] \mid z_3 \in \mathbb{R}\} \cap \Omega_G.$$

Let  $\mathbf{y} \in \bigcup_{k \in \mathbb{Z}, [i, j, k]_G \in \Omega} \{[i, j, k + q]_G \mid q \in [-\frac{1}{2}, \frac{1}{2})\}$ . So there is a  $q \in [-\frac{1}{2}, \frac{1}{2})$  and a  $k \in \mathbb{Z}$  so that  $\mathbf{y} = [i, j, k + q]_G$  and  $[i, j, k]_G \in \Omega$ . It holds that

$$\text{round}((k + k_0 + q) l_z / l_z) l_z = (\text{round}(q) + k + k_0) l_z = (k + k_0) l_z$$

and therefore

$$\begin{aligned} \mathbf{y}_r &= \text{round}_G(\mathbf{y}) = \text{round}_G([x_{\min} + i l_{x,y}, y_{\min} + j l_{x,y}, z_{\min} + (k + q) l_z]) = \\ &= \text{round}_G([(i + i_0) l_{x,y}, (j + j_0) l_{x,y}, (k + k_0 + q) l_z]) = \\ &= [\text{round}((i + i_0) l_{x,y} / l_{x,y}) l_{x,y}, \text{round}((j + j_0) l_{x,y} / l_{x,y}) l_{x,y}, \text{round}((k + k_0 + q) l_z / l_z) l_z] = \\ &= [(i + i_0) l_{x,y}, (j + j_0) l_{x,y}, (k + k_0) l_z] = [x_{\min} + i l_{x,y}, y_{\min} + j l_{x,y}, z_{\min} + k l_z] = \\ &= [i, j, k]_G \in \Omega. \end{aligned}$$

Consequently, it follows that  $\mathbf{y} \in \Omega_G$ . Setting  $z_3 := z_{\min} + (k + q) l_z$  yields

$$\mathbf{y} = [x_{\min} + i l_{x,y}, y_{\min} + j l_{x,y}, z_{\min} + (k + q) l_z] = [z_1, z_2, z_3]$$

and therefore

$$\mathbf{y} \in \{[z_1, z_2, z_3] \mid z_3 \in \mathbb{R}\} \cap \Omega_G.$$

It is left to show that the right hand side is a union of disjoint sets (equation (2.13)). So let  $\mathbf{x} \in \{[i, j, k_1 + q]_G \mid q \in [-\frac{1}{2}, \frac{1}{2})\}$  and  $\mathbf{y} \in \{[i, j, k_1 + q]_G \mid q \in [-\frac{1}{2}, \frac{1}{2})\}$  and without loss of generality,  $k_1 < k_2$ . This yields  $k_1 \leq k_2 - 1$  and there exist  $q_1, q_2 \in [-\frac{1}{2}, \frac{1}{2})$  so that

$$\mathbf{x} = [x_1, x_2, x_3] = [i, j, k_1 + q_1]_G = [x_{\min} + i l_{x,y}, y_{\min} + j l_{x,y}, z_{\min} + (k_1 + q_1) l_z]$$

and

$$\mathbf{y} = [y_1, y_2, y_3] = [i, j, k_2 + q_2]_G = [x_{\min} + i l_{x,y}, y_{\min} + j l_{x,y}, z_{\min} + (k_2 + q_2) l_z].$$

It holds that

$$x_3 = z_{\min} + (k_1 + q_1) l_z < z_{\min} + (k_2 - 1) l_z + l_z / 2 = z_{\min} + (k_2 - 1/2) l_z \leq z_{\min} + (k_2 + q_2) l_z = y_3$$



and therefore  $\mathbf{x} \neq \mathbf{y}$ . So,

$$\{[i, j, k_1 + q]_G \mid q \in [-\frac{1}{2}, \frac{1}{2})\} \cap \{[i, j, k_2 + q]_G \mid q \in [-\frac{1}{2}, \frac{1}{2})\} = \emptyset.$$

□

For  $z_1 = i l_{x,y} + x_{min}$ ,  $z_2 = j l_{x,y} + y_{min}$ ,  $i, j \in \mathbb{Z}$ , we use Theorem 2.13 to transform the integral

$$\begin{aligned} \hat{F}(z_1, z_2) &= \int_{\mathbb{R}} \mathbb{1}_{\Omega_G}([z_1, z_2, z_3]) \operatorname{sinc}(q\|[z_1, z_2, z_3] - \mathbf{x}\|) dz_3 = \\ &= \int_{\mathbb{R}} \mathbb{1}_{\{[z_1, z_2, \tilde{z}_3] \mid \tilde{z}_3 \in \mathbb{R}\} \cap \Omega_G}([z_1, z_2, z_3]) \operatorname{sinc}(q\|[z_1, z_2, z_3] - \mathbf{x}\|) dz_3 = \\ &= \int_{\mathbb{R}} \mathbb{1}_{\bigcup_{k \in \mathbb{Z}, [i, j, k]_G \in \Omega} \{[i, j, k+q]_G \mid q \in [-\frac{1}{2}, \frac{1}{2})\}}([z_1, z_2, z_3]) \operatorname{sinc}(q\|[z_1, z_2, z_3] - \mathbf{x}\|) dz_3 = \\ &= \int_{\mathbb{R}} \sum_{k \in \mathbb{Z}, [i, j, k]_G \in \Omega} \mathbb{1}_{\{[i, j, k+q]_G \mid q \in [-\frac{1}{2}, \frac{1}{2})\}}([z_1, z_2, z_3]) \operatorname{sinc}(q\|[z_1, z_2, z_3] - \mathbf{x}\|) dz_3 = \\ &= \sum_{k \in \mathbb{Z}, [i, j, k]_G \in \Omega} \int_{\mathbb{R}} \mathbb{1}_{\{[z_1, z_2, z_{min} + (k+q) l_z] \mid q \in [-\frac{1}{2}, \frac{1}{2})\}}([z_1, z_2, z_3]) \operatorname{sinc}(q\|[z_1, z_2, z_3] - \mathbf{x}\|) dz_3 = \\ &= \sum_{k \in \mathbb{Z}, [i, j, k]_G \in \Omega} \int_{z_{min} + (k-1/2) l_z}^{z_{min} + (k+1/2) l_z} \operatorname{sinc}(q\|[z_1, z_2, z_3] - \mathbf{x}\|) dz_3. \end{aligned} \tag{53}$$

To evaluate the last sum, we have to check for which  $k \in \mathbb{Z}$  it holds that  $[i, j, k]_G \in \Omega$ . We therefore use the variable *thinned\_out\_boundary*, because *thinned\_out\_boundary*[*i*][*j*] is a sorted list (ascending values) with all  $k \in \mathbb{Z}$  so that  $[i, j, k]_G \in \Omega$  and  $([i, j, k-1]_G \notin \Omega$  or  $[i, j, k+1]_G \notin \Omega)$ . Let *thinned\_out\_boundary*[*i*][*j*] be the list

$$\text{thinned\_out\_boundary}[i][j] = [\tilde{k}_1, \dots, \tilde{k}_m], \quad \tilde{k}_l \in \mathbb{Z}, l = 1, \dots, m.$$

We now want to find disjoint intervals  $[k_{l,1}, k_{l,2}]$ ,  $k_{l,1}, k_{l,2} \in \mathbb{Z}$  for  $l = 1, \dots, n$  so that

$$\{k \in \mathbb{Z} \mid [i, j, k]_G \in \Omega\} = \mathbb{Z} \cap \bigcup_{l=1}^n [k_{l,1}, k_{l,2}]. \tag{54}$$

The following pseudo-code shows how we find those intervals:

```
intervals = []
is_outside = True
k1 = 0
for k in thinned_out_boundary:
```

```

if is_outside:
    if [i, j, k + 1]_G not inside Omega:
        intervals.add([k, k])
    else:
        k1 = k
        is_outside = False
else:
    intervals.add([k1, k])
    is_outside = True

```

**Lemma 2.14.** *Let  $[k_{l,1}, k_{l,2}]$ ,  $l \in \{1, \dots, n\}$  be an interval obtained by the previous pseudo-code. Then, for every  $\tilde{k} \in [k_{l,1}, k_{l,2}]$  it holds that  $[i, j, \tilde{k}]_G \in \Omega$ .*

*Proof.* The intervals are added to the variable *intervals* at 2 different locations. At the first location,  $k_{l,1} = k_{l,2} \in \text{thinned\_out\_boundary}[i][j]$  and therefore  $[i, j, \tilde{k}]_G = [i, j, k_{l,1}]_G \in \Omega$ .

At the second location,  $k = k_{l,2}$  is the index of the actual point in *thinned\\_out\\_boundary* and  $k_1 = k_{l,1}$  is the index of the previous point in *thinned\\_out\\_boundary*.  $k_1$  was only set to the previous point in *thinned\\_out\\_boundary*, if  $[i, j, k_1 + 1]_G \in \Omega$ . Starting at  $k_1$ ,  $k$  is the next index that is a boundary point in  $z$ -direction, therefore every point  $[i, j, \tilde{k}]_G$ ,  $\tilde{k} \in \{k_{l,1} + 1, \dots, k_{l,2} - 1\}$  also has to lie inside  $\Omega$ .

Because  $k_{l,1}$  and  $k_{l,2}$  are inside *thinned\\_out\\_boundary* $[i][j]$ , we get  $[i, j, k_{l,1}]_G \in \Omega$  and  $[i, j, k_{l,2}]_G \in \Omega$ . Altogether, for  $\tilde{k} \in \{k_{l,1}, \dots, k_{l,2}\}$  it holds that  $[i, j, \tilde{k}]_G \in \Omega$ . Therefore,  $[i, j, \tilde{k}]_G \in \Omega$ .  $\square$

**Lemma 2.15.** *For the intervals  $[k_{l,1}, k_{l,2}]$ ,  $l \in \{1, \dots, n\}$  obtained by the previous pseudo-code, it holds that*

$$[i, j, k_{l,2} + 1]_G \notin \Omega.$$

*Proof.* The intervals are added to the variable *intervals* at 2 different locations. At the first location,  $[i, j, k_{l,2} + 1]_G \notin \Omega$  is exactly the condition when the interval is added to *intervals*.

At the second location,  $k = k_{l,2}$  is the index of the actual point in *thinned\\_out\\_boundary* and  $k_1 = k_{l,1}$  is the index of the previous point in *thinned\\_out\\_boundary*. Lemma 2.14 states that for every  $\tilde{k} \in [k_{l,1}, k_{l,2}]$ ,  $[i, j, \tilde{k}]_G \in \Omega$ . In particular,  $[i, j, k - 1]_G \in \Omega$ . Because  $k$  is an element of *thinned\\_out\\_boundary*, either  $[i, j, k - 1]_G$  or  $[i, j, k + 1]_G$  have to lie outside  $\Omega$  and therefore,  $[i, j, k + 1]_G = [i, j, k_{l,2} + 1]_G \notin \Omega$ .  $\square$

**Theorem 2.16.** *For the intervals  $[k_{l,1}, k_{l,2}]$ ,  $l \in \{1, \dots, n\}$  obtained by the previous pseudo-code, the equation (54),*

$$\{k \in \mathbb{Z} \mid [i, j, k]_G \in \Omega\} = \mathbb{Z} \cap \bigcup_{l=1}^n [k_{l,1}, k_{l,2}],$$

*holds true. Moreover, the intervals  $[k_{l,1}, k_{l,2}]$  are pairwise disjoint.*

*Proof.* The case, if the left and the right set are both empty, is trivial. If there is no point  $k \in \mathbb{Z}$  so that  $[i, j, k]_G \in \Omega$ , the list *thinned\_out\_boundary* is empty, and vice versa.

$$\{k \in \mathbb{Z} \mid [i, j, k]_G \in \Omega\} \supset \mathbb{Z} \cap \bigcup_{l=1}^n [k_{l,1}, k_{l,2}]$$

follows directly from Lemma 2.15. So we only have to show that

$$\{k \in \mathbb{Z} \mid [i, j, k]_G \in \Omega\} \subset \mathbb{Z} \cap \bigcup_{l=1}^n [k_{l,1}, k_{l,2}].$$

Let  $k \in \mathbb{Z}$  so that  $[i, j, k]_G \in \Omega$ . Then  $k \geq k_{1,1}$  (checking *thinned\_out\_boundary* $[i][j]$  we see that  $k_{1,1} = \min\{\tilde{k} \mid [i, j, \tilde{k}]_G \in \Omega\}$ ). Let

$$l := \max\{p \in \{1, \dots, n\} \mid k_{p,1} \leq k\}. \quad (55)$$

We show that  $k_{l,2} \geq k$ :

Suppose that  $k_{l,2} < k$ , then Lemma 2.15 yields that  $[i, j, k_{l,2} + 1]_G \notin \Omega$ . Because  $[i, j, k]_G \in \Omega$  it holds that  $k \neq k_{l,2} + 1$ . Together with  $k_{l,2} < k$  it follows that  $k_{l,2} + 1 < k$ , so there has to be a  $\hat{k} \in \{k_{l,2} + 2, \dots, k\}$  so that  $[i, j, \hat{k}]_G \in \Omega$  and  $[i, j, \hat{k} - 1]_G \notin \Omega$ . It follows that  $\hat{k} \in \text{thinned\_out\_boundary}[i][j]$ . So there is an  $\hat{l} \in \{1, \dots, n\}$  so that  $\hat{k} = k_{\hat{l},1}$  or  $\hat{k} = k_{\hat{l},2}$ .

Lets analyze the case  $\hat{k} = k_{\hat{l},2}$ . Because of  $[i, j, k_{\hat{l},2} - 1]_G = [i, j, \hat{k} - 1]_G \notin \Omega$  and Lemma 2.14 it follows that  $k_{\hat{l},1} > k_{\hat{l},2} - 1$ . Moreover, for every obtained interval it holds that  $k_{\hat{l},1} \leq k_{\hat{l},2} - 1$ . Hence, it follows that  $k_{\hat{l},1} = k_{\hat{l},2} = \hat{k}$ . Therefore,  $\hat{k} = k_{\hat{l},1}$ . But  $k_{l,1} < k_{\hat{l},1} = \hat{k} \leq k$ , which is a contradiction to equation (55).

So,  $k_{l,2} \geq k$  has to hold. Together with  $k_{l,2} \leq k$  it holds that  $k \in [k_{l,1}, k_{l,2}]$  and therefore,  $k \in \mathbb{Z} \cap \bigcup_{l=1}^n [k_{l,1}, k_{l,2}]$ .

Left to show is that the intervals  $[k_{l,1}, k_{l,2}]$  are pairwise disjoint. The elements of *thinned\_out\_boundary* are strictly monotonically increasing, so for the intervals it always holds true that  $k_{l,2} < k_{l+1,1}, l \in \{1, \dots, n-1\}$ . Therefore the intervals are disjoint.  $\square$

We continue to rearrange equation (53) by using Theorem 2.16:

$$\begin{aligned}
\hat{F}(x_{min} + i l_{x,y}, y_{min} + j l_{x,y}) &= \hat{F}(z_1, z_2) = \\
&= \int_{\mathbb{R}} \mathbb{1}_{\Omega_G}([z_1, z_2, z_3]) \operatorname{sinc}(q\|[z_1, z_2, z_3] - \mathbf{x}\|) dz_3 = \\
&= \sum_{k \in \mathbb{Z}, [i,j,k]_G \in \Omega} \int_{z_{min} + (k-1/2) l_z}^{z_{min} + (k+1/2) l_z} \operatorname{sinc}(q\|[z_1, z_2, z_3] - \mathbf{x}\|) dz_3 = \\
&= \sum_{k \in \mathbb{Z} \cap \bigcup_{l=1}^n [k_{l,1}, k_{l,2}]_{z_{min} + (k-1/2) l_z}} \int_{z_{min} + (k-1/2) l_z}^{z_{min} + (k+1/2) l_z} \operatorname{sinc}(q\|[z_1, z_2, z_3] - \mathbf{x}\|) dz_3 = \\
&= \sum_{l=1}^n \sum_{k=k_{l,1}}^{k_{l,2}} \int_{z_{min} + (k-1/2) l_z}^{z_{min} + (k+1/2) l_z} \operatorname{sinc}(q\|[z_1, z_2, z_3] - \mathbf{x}\|) dz_3 = \\
&= \sum_{l=1}^n \int_{z_{min} + (k_{l,1}-1/2) l_z}^{z_{min} + (k_{l,2}+1/2) l_z} \operatorname{sinc}(q\|[z_1, z_2, z_3] - \mathbf{x}\|) dz_3.
\end{aligned} \tag{56}$$

### 2.5.2 Sufficient number of Gaussian points in $z$ -direction

We now have a sum of  $n$  integrals that have to be integrated numerically. The number of Gaussian points necessary for integrating one of the integrals to ensure some accuracy is very much dependent on how many oscillations the sinc-function has between the integral boundaries. Therefore, we analyze the sinc-function a little bit.

We tested the integral of  $\operatorname{sinc}(x) = \sin(x)/x$  from  $a = 0.0$  to  $b = 2\pi$ . The number of Gaussian points, so that a numerical integration over this integral has an relative error of  $10^{-5}$  or below, is 6. We tested it also with other interval boundaries and length  $2\pi$  and also with smaller intervals, but the relative error was never over  $10^{-5}$ . When integrating from  $a = 0.0$  to  $b = 4\pi$ , for an error of max  $10^{-5}$ , 9 Gaussian points were necessary. Integration  $a = 0.0$  to  $b = 6\pi$  needed 12 Gaussian points for the same relative error. So, for integrating  $\operatorname{sinc}(x) = \sin(x)/x$  from  $a$  to  $b$ , by empirical testing it should be sufficient to use

$$n_{P,emp} := \operatorname{ceil}(3 + 3(b - a)/(2\pi)) \tag{57}$$

Gaussian points.

We continue to analyze  $\text{sinc}(q\|[z_1, z_2, z_3] - \mathbf{x}\|)$  in equation (56). First we see that for fixed  $z_1, z_2 \in \mathbb{R}$ ,  $s := \{[z_1, z_2, z_3] \mid z_3 \in \mathbb{R}\}$  is a straight line and that  $\|[z_1, z_2, z_3] - \mathbf{x}\|$  is the distance from the point  $[z_1, z_2, z_3] \in s$  to the given point  $\mathbf{x} \in \mathbb{R}^3$ . Moreover, we know that there is a minimum distance from the line  $s$  to  $\mathbf{x}$  and that this minimum distance is obtained at some point on  $s$ , so let

$$\tilde{\mathbf{z}} = [z_1, z_2, \tilde{z}_3] \in s, \quad \tilde{z}_3 = \text{argmin}_{z_3 \in \mathbb{R}} \{ \|[z_1, z_2, z_3] - \mathbf{x}\| \}.$$

For  $z_3 < \tilde{z}_3$ ,  $f(z_3) := \|[z_1, z_2, z_3] - \mathbf{x}\|$  is strictly monotonically decreasing and for  $z_3 > \tilde{z}_3$ ,  $f(z_3) = \|[z_1, z_2, z_3] - \mathbf{x}\|$  is strictly monotonically increasing. This holds also true for  $\tilde{f}(z_3) := q\|[z_1, z_2, z_3] - \mathbf{x}\|$ .

We set

$$\begin{aligned} \tilde{a} &:= z_{\min} + (k_{l,1} - 1/2) l_z, \\ \tilde{b} &:= z_{\min} + (k_{l,2} + 1/2) l_z, \end{aligned}$$

and analyze the boundaries of the integral

$$I_1 := \int_{z_{\min} + (k_{l,1} - 1/2) l_z}^{z_{\min} + (k_{l,2} + 1/2) l_z} \text{sinc}(q\|[z_1, z_2, z_3] - \mathbf{x}\|) dz_3 = \int_{\tilde{a}}^{\tilde{b}} \text{sinc}(\tilde{f}(z_3)) dz_3.$$

Then we have to distinguish between the cases  $\tilde{z}_3 \in [\tilde{a}, \tilde{b}]$  and  $\tilde{z}_3 \notin [\tilde{a}, \tilde{b}]$ .

**Case 1**,  $\tilde{z}_3 \notin [\tilde{a}, \tilde{b}]$ : Then,  $\tilde{f}$  is strictly monotone on  $[\tilde{a}, \tilde{b}]$  and the largest and smallest argument for  $\text{sinc}(\cdot)$  are

$$\begin{aligned} \tilde{f}(\tilde{a}) &= q\|[z_1, z_2, z_{\min} + (k_{l,1} - 1/2) l_z] - \mathbf{x}\|, \\ \tilde{f}(\tilde{b}) &= q\|[z_1, z_2, z_{\min} + (k_{l,2} + 1/2) l_z] - \mathbf{x}\|, \end{aligned}$$

(it does not matter which one is the largest/smallest argument) and so, the length of the interval  $\text{sinc}(\cdot)$  has to be integrated over, is:

$$\begin{aligned} |\tilde{f}(\tilde{b}) - \tilde{f}(\tilde{a})| &= \\ &= q\|[z_1, z_2, z_{\min} + (k_{l,2} + 1/2) l_z] - \mathbf{x}\| - \|[z_1, z_2, z_{\min} + (k_{l,1} - 1/2) l_z] - \mathbf{x}\| \leq \\ &\leq q\|([z_1, z_2, z_{\min} + (k_{l,2} + 1/2) l_z] - \mathbf{x}) - ([z_1, z_2, z_{\min} + (k_{l,1} - 1/2) l_z] - \mathbf{x})\| = \\ &= q\|[0, 0, (k_{l,2} + 1/2) l_z - (k_{l,1} - 1/2) l_z]\| = q l_z (k_{l,2} - k_{l,1} + 1). \end{aligned}$$

When we compare the following 2 integrals

$$I_1 = \int_{\tilde{a}}^{\tilde{b}} \text{sinc}(\tilde{f}(x)) dx \quad \text{and} \quad I_2 := \int_{\tilde{f}(\tilde{a})}^{\tilde{f}(\tilde{b})} \text{sinc}(y) dy,$$

we see that as long as  $\tilde{f}$  is strictly monotone in  $[\tilde{a}, \tilde{b}]$ ,  $\text{sinc}$  has as many oscillations in  $I_1$  as in  $I_2$ . Therefore, the same number of Gaussian points should be sufficient

for both integrals to get about the same maximum relative error. When we use the empirical result in equation (57), we get the minimum number of Gaussian points:

$$n_P(k_{l,1}, k_{l,2}) := \text{ceil}(3 + 3 q l_z (k_{l,2} - k_{l,1} + 1)/(2\pi)) \geq \text{ceil}(3 + 3 |\tilde{f}(\tilde{b}) - \tilde{f}(\tilde{a})|/(2\pi)).$$

**Case 2,**  $\tilde{z}_3 \in [\tilde{a}, \tilde{b}]$ : Then,  $\tilde{f}$  is strictly monotonically decreasing on  $[\tilde{a}, \tilde{z}_3]$  and strictly monotonically increasing on  $[\tilde{z}_3, \tilde{b}]$ . Let  $k_{\tilde{z}_3} := (\tilde{z}_3 - z_{min})/l_z \in \mathbb{R}$ , then it holds true that

$$\tilde{z}_3 = z_{min} + k_{\tilde{z}_3} l_z \quad \text{and} \quad k_{l,1} - 1/2 \leq k_{\tilde{z}_3} \leq k_{l,2} + 1/2.$$

Analogously to Case 1 we can conclude that

$$\begin{aligned} |\tilde{f}(\tilde{b}) - \tilde{f}(\tilde{z}_3)| &\leq q l_z (k_{l,2} - k_{\tilde{z}_3} + 1/2) \\ |\tilde{f}(\tilde{a}) - \tilde{f}(\tilde{z}_3)| &\leq q l_z (k_{\tilde{z}_3} - k_{l,1} + 1/2). \end{aligned}$$

We also can conclude that for the integrals

$$\begin{aligned} I_{1,1} &= \int_{\tilde{a}}^{\tilde{z}_3} \text{sinc}(\tilde{f}(x)) dx, & I_{2,1} &:= \int_{\tilde{f}(\tilde{a})}^{\tilde{f}(\tilde{z}_3)} \text{sinc}(y) dy, \\ I_{1,2} &:= \int_{\tilde{z}_3}^{\tilde{b}} \text{sinc}(\tilde{f}(x)) dx, & I_{2,2} &:= \int_{\tilde{f}(\tilde{z}_3)}^{\tilde{f}(\tilde{b})} \text{sinc}(y) dy, \end{aligned}$$

it holds that  $I_{1,1}$  and  $I_{2,1}$  respectively  $I_{1,2}$  and  $I_{2,2}$  need about the same number of Gaussian points for the same numerical integration accuracy. Because  $I_1$  is split up into  $I_{1,1}$  and  $I_{1,2}$ , the number of necessary Gaussian points for  $(I_{2,1}$  and  $I_{2,2})$  should be about the number of necessary Gaussian points for  $I_1$ . Therefore,

$$\begin{aligned} n_P(k_{l,1}, k_{l,2}) &= \text{ceil}(3 + 3 q l_z (k_{l,2} - k_{l,1} + 1)/(2\pi)) = \\ &= \text{ceil}(3 + 3 q l_z (k_{l,2} - k_{\tilde{z}_3} + 1/2 + k_{\tilde{z}_3} - k_{l,1} + 1/2)/(2\pi)) \geq \\ &\geq \text{ceil}(3 + 3 (|\tilde{f}(\tilde{b}) - \tilde{f}(\tilde{z}_3)| + |\tilde{f}(\tilde{z}_3) - \tilde{f}(\tilde{a})|)/(2\pi)) \end{aligned}$$

Gaussian points should also be sufficient for Case 2 to ensure sufficient accuracy of the numerical integration.

Now (for both cases together) we want to find an  $\tilde{n}_P \in \mathbb{N}$  so that  $\tilde{n}_P \geq n_P(k_{l,1}, k_{l,2})$  for each choice of  $k_{l,1}$  and  $k_{l,2}$ . It holds that  $k_{l,1}$  and  $k_{l,2}$  are third coordinates for grid points inside  $\Omega$ . Therefore we can use Lemma 2.2 to see that with the use of  $\mathbf{m} = [m_1, m_2, m_3] = \text{mid}XYZ$  and  $\mathbf{r} = [r_1, r_2, r_3] = \text{range}XYZ$  for  $k_{l,1}$  and  $k_{l,2}$  it holds that

$$m_3 - r_3/2 \leq k_{l,1} \leq k_{l,2} \leq m_3 + r_3/2,$$

and therefore

$$n_P(k_{l,1}, k_{l,2}) := \text{ceil}(3 + 3 q l_z (k_{l,2} - k_{l,1} + 1)/(2\pi)) \leq \text{ceil}(3 + 3 q l_z (r_3 + 1)/(2\pi)).$$

So we set

$$\tilde{n}_P := \text{ceil}(3 + 3 q l_z (r_3 + 1)/(2\pi)). \quad (58)$$

We showed that using  $\tilde{n}_P$  Gaussian points should be sufficient to numerically integrate every integral at the end of equation (56).

### 2.5.3 Using bilinear interpolation to approximate $\hat{F}(z_1, z_2)$

In numerical integration,  $(z_1, z_2)$  will be Gaussian points and cannot be restricted to grid points only. Therefore, we use bilinear interpolation to determine  $\hat{F}(z_1, z_2)$  for arbitrary  $z_1, z_2$ . We recall equation (51), which is given as

$$\int_{\Omega_G} \text{sinc}(q\|\mathbf{z} - \mathbf{x}\|) d\mathbf{z} = \int_{\mathbb{R}} \int_{\mathbb{R}} \hat{F}(z_1, z_2) dz_2 dz_1,$$

with  $\hat{F}(z_1, z_2)$  given as

$$\hat{F}(z_1, z_2) = \int_{\mathbb{R}} \mathbb{1}_{\Omega_G}([z_1, z_2, z_3]) \text{sinc}(q\|[z_1, z_2, z_3] - \mathbf{x}\|) dz_3.$$

Furthermore, let the grid  $G_{x,y}$  be

$$G_{x,y} := \{[x_{min} + i l_{x,y}, y_{min} + j l_{x,y}] \mid i, j \in \mathbb{Z}\}, \quad (59)$$

which is the grid  $G$  projected onto the first and second coordinates. For  $i, j \in \mathbb{Z}$  let  $\hat{F}_{ij}$  be

$$\hat{F}_{i,j} := \hat{F}(x_{min} + i l_{x,y}, y_{min} + j l_{x,y}), \quad (60)$$

which are the values of  $\hat{F}$  on the grid points of  $G_{x,y}$ .

Now let  $\hat{P} : \mathbb{R}^2 \rightarrow \mathbb{R}$  be the bilinear interpolation of  $\hat{F}(z_1, z_2)$  on the grid  $G_{x,y}$  as given in the book *Numerical recipes in C: the art of scientific computing* by W. H. Press, B. P. Flannery, S. A. Teukolsky and W. T. Vetterling [8]. We know that for a point  $(x, y) \in \mathbb{R}^2$  there exists a rectangular 2d grid cell  $C$  of  $G_{x,y}$  with corner points  $(x_1, y_1), (x_1, y_2), (x_2, y_1), (x_2, y_2) \in \mathbb{R}^2$  so that  $x_1 \leq x < x_2$  and  $y_1 \leq y < y_2$ . On  $(x, y)$ ,  $\hat{P}(x, y)$  is given as follows:

$$\hat{P}(x, y) = \frac{1}{(x_2 - x_1)(y_2 - y_1)} \begin{bmatrix} (x_2 - x), & (x - x_1) \end{bmatrix} \hat{F} \begin{bmatrix} y_2 - y \\ y - y_1 \end{bmatrix} \quad (61)$$

with  $\hat{F} = \begin{bmatrix} \hat{F}(x_1, y_1) & \hat{F}(x_1, y_2) \\ \hat{F}(x_2, y_1) & \hat{F}(x_2, y_2) \end{bmatrix}.$

We now can approximate equation (51) with the help of the bilinear interpolation  $\hat{P}(z_1, z_2)$  as follows:

$$\int_{\Omega_G} \text{sinc}(q\|\mathbf{z} - \mathbf{x}\|) d\mathbf{z} = \int_{\mathbb{R}} \int_{\mathbb{R}} \hat{F}(z_1, z_2) dz_2 dz_1 \approx \int_{\mathbb{R}} \int_{\mathbb{R}} \hat{P}(z_1, z_2) dz_2 dz_1. \quad (62)$$

We want to integrate equation (62) numerically, so we first want to find boundaries for the integral, so that  $\hat{P}(z_1, z_2)$  can only be non-zero inside those boundaries.

**Lemma 2.17.** *Let  $\mathbf{m} = [m_1, m_2, m_3] := \text{mid}XYZ$  and  $\mathbf{r} = [r_1, r_2, r_3] = \text{range}XYZg$ . Then,  $\hat{F}_{i,j}$  can be non-zero only for indices  $i, j \in \mathbb{Z}$  so that  $[i, j]$  is inside the rectangle*

$$R_{\hat{F}} := [m_1 - r_1/2, m_1 + r_1/2] \times [m_2 - r_2/2, m_2 + r_2/2].$$

*Proof.* Let  $i, j \in \mathbb{Z}$  so that  $\hat{F}_{i,j} \neq 0$ . Equation (56) yields

$$\hat{F}_{i,j} = \hat{F}(x_{\min} + i l_{x,y}, y_{\min} + j l_{x,y}) = \sum_{l=1}^n \int_{z_{\min} + (k_{l,1}-1/2)l_z}^{z_{\min} + (k_{l,2}+1/2)l_z} \text{sinc}(q\|[z_1, z_2, z_3] - \mathbf{x}\|) dz_3,$$

where  $k_{l,1}, k_{l,2} \in \mathbb{Z}, l = 1, \dots, n$  are indices so that  $[i, j, k_{l,1}]_G$  and  $[i, j, k_{l,2}]_G$  are boundary grid points of  $\Omega$  with respect to  $G$ .

We have  $\hat{F}_{i,j} \neq 0$ , so the sum in equation (56) is not an empty sum. In particular, there exists a  $k \in \mathbb{Z}$  so that  $[i, j, k]_G \in \Omega$ . Lemma 2.2 yields that every grid point inside  $\Omega$  is contained in the cuboid

$$C = [m_1 - r_1/2, m_1 + r_1/2] \times [m_2 - r_2/2, m_2 + r_2/2] \times [m_3 - r_3/2, m_3 + r_3/2].$$

So,  $[i, j, k]_G \in C$  and therefore

$$[i, j] \in [m_1 - r_1/2, m_1 + r_1/2] \times [m_2 - r_2/2, m_2 + r_2/2] = R_{\hat{F}}.$$

□

**Theorem 2.18.** *Let  $\mathbf{m} = [m_1, m_2, m_3] := \text{mid}XYZ$  and  $\tilde{\mathbf{r}} = [\tilde{r}_1, \tilde{r}_2, \tilde{r}_3] = \text{range}XYZ$ . Then, for  $z_1 = x_{\min} + q_1 l_{x,y}, z_2 = y_{\min} + q_2 l_{x,y}, q_1, q_2 \in \mathbb{R}$ , the value of  $\hat{P}(z_1, z_2)$  can be non-zero only if  $[q_1, q_2]$  is inside the rectangle*

$$R = [m_1 - \tilde{r}_1/2, m_1 + \tilde{r}_1/2] \times [m_2 - \tilde{r}_2/2, m_2 + \tilde{r}_2/2].$$

*Proof.* Let  $z_1 = x_{\min} + q_1 l_{x,y}, z_2 = y_{\min} + q_2 l_{x,y}, q_1, q_2 \in \mathbb{R}$  and  $\hat{P}(z_1, z_2) \neq 0$ .

Furthermore, let  $i := \text{floor}(r_1)$  and  $j := \text{floor}(r_2)$ . Then we see that  $[z_1, z_2]$  is inside the (2-dimensional) grid cell  $C$  of  $G_{x,y}$  with the 4 corner points

$$c_{m,n} := [x_{\min} + m l_{x,y}, y_{\min} + n l_{x,y}], \quad m \in \{i, i+1\}, n \in \{j, j+1\}.$$



Because  $[z_1, z_2] \in C$  and because of equation (61), there are coefficients  $a_{i,j}, a_{i,j+1}, a_{i+1,j}, a_{i+1,j+1} \in \mathbb{R}$  so that

$$\hat{P}(z_1, z_2) = a_{i,j} \hat{F}_{i,j} + a_{i,j+1} \hat{F}_{i,j+1} + a_{i+1,j} \hat{F}_{i+1,j} + a_{i+1,j+1} \hat{F}_{i+1,j+1} \neq 0,$$

so at least one of the values  $\hat{F}_{i,j}, \hat{F}_{i,j+1}, \hat{F}_{i+1,j}$  or  $\hat{F}_{i+1,j+1}$  is not equal to 0.

With  $\mathbf{m} = [m_1, m_2, m_3] := \text{mid}XYZ$  and  $\mathbf{r} = [r_1, r_2, r_3] = \text{range}XYZg$ , Lemma 2.17 yields that one of the index pairs  $[i, j], [i, j+1], [i+1, j]$  or  $[i+1, j+1]$  has to be inside  $[m_1 - r_1/2, m_1 + r_1/2] \times [m_2 - r_2/2, m_2 + r_2/2] = R_{\hat{F}}$ .

We can follow that these 4 inequalities have to be met:

$$\begin{aligned} i+1 &\geq m_1 - r_1/2, \\ i &\leq m_1 + r_1/2, \\ j+1 &\geq m_2 - r_2/2, \\ j &\leq m_2 + r_2/2. \end{aligned}$$

In the section 2.2.4 we defined  $\tilde{\mathbf{r}}$  as

$$\tilde{\mathbf{r}} = \mathbf{r} + [2, 2, 2] \quad \text{with} \quad \mathbf{r} = [r_1, r_2, r_3] = \text{range}XYZg.$$

Together with  $i \leq q_1 \leq i+1$  and  $j \leq q_2 \leq j+1$  we conclude that

$$\begin{aligned} q_1 &\geq i = (i+1) - 1 \geq m_1 - r_1/2 - 1 = m_1 - (r_1 + 2)/2 = m_1 - \tilde{r}_1/2 \\ q_1 &\leq i+1 \leq m_1 + r_1/2 + 1 = m_1 + (r_1 + 2)/2 = m_1 + \tilde{r}_1/2 \\ q_2 &\geq j = (j+1) - 1 \geq m_2 - r_2/2 - 1 = m_2 - (r_2 + 2)/2 = m_2 - \tilde{r}_2/2 \\ q_2 &\leq j+1 \leq m_2 + r_2/2 + 1 = m_2 + (r_2 + 2)/2 = m_2 + \tilde{r}_2/2 \end{aligned}$$

and therefore  $[q_1, q_2] \in R$ .  $\square$

For equation (62), let  $q_1 := (z_1 - x_{\min})/l_{x,y}$  and  $q_2 := (z_1 - x_{\min})/l_{x,y}$ . Then  $z_1 = x_{\min} + q_1 l_{x,y}$ ,  $z_2 = y_{\min} + q_2 l_{x,y}$ ,  $dz_1 = l_{x,y} dq_1$  and  $dz_2 = l_{x,y} dq_2$ . We can transform equation (62) and apply Theorem 2.18 to get

$$\begin{aligned} \int_{\Omega_G} \text{sinc}(q\|\mathbf{z} - \mathbf{x}\|) d\mathbf{z} &\approx \int_{\mathbb{R}} \int_{\mathbb{R}} \hat{P}(z_1, z_2) dz_2 dz_1 = \\ &= l_{x,y}^2 \int_{\mathbb{R}} \int_{\mathbb{R}} \hat{P}(x_{\min} + q_1 l_{x,y}, y_{\min} + q_2 l_{x,y}) dq_2 dq_1 = \\ &= l_{x,y}^2 \int_{m_1 - \tilde{r}_1/2}^{m_1 + \tilde{r}_1/2} \int_{m_2 - \tilde{r}_2/2}^{m_2 + \tilde{r}_2/2} \hat{P}(x_{\min} + q_1 l_{x,y}, y_{\min} + q_2 l_{x,y}) dq_2 dq_1. \end{aligned} \tag{63}$$

### 2.5.4 Applying the Gaussian quadrature rule

We use the Gaussian quadrature rule to numerically integrate these two integrals. The number of Gaussian points we use for both integrals is determined by the input parameter  $n_G := nrOfGaussPoints$ . In section 2.4 we saw that if we apply the Gaussian quadrature rule with  $n_G$  Gaussian points  $x_1, \dots, x_{n_G}$  and Gaussian weights  $w_1, \dots, w_{n_G}$  on an integral  $\int_a^b f(x) dx$ , the following holds true:

$$\int_a^b f(x) dx \approx \frac{b-a}{2} \sum_{i=1}^{n_G} w_i f\left(\frac{b-a}{2}x_i + \frac{a+b}{2}\right). \quad (64)$$

We see that for the two integrals in equation (63), for  $k \in \{1, 2\}$  and for the boundaries of the integrals  $a_k$  and  $b_k$ , it holds that

$$\begin{aligned} \frac{b_k - a_k}{2} &= ((m_k + \tilde{r}_k/2) - (m_k - \tilde{r}_k/2))/2 = \frac{\tilde{r}_k}{2}, \\ \frac{a_k + b_k}{2} &= ((m_k - \tilde{r}_k/2) + (m_k + \tilde{r}_k/2))/2 = m_k \end{aligned}$$

and for each Gaussian point,

$$q_k(x_i) = \frac{b_k - a_k}{2}x_i + \frac{a_k + b_k}{2} = m_k + \frac{\tilde{r}_k}{2}x_i.$$

Therefore we can apply the Gaussian quadrature rule on equation (63) as follows:

$$\begin{aligned} \int_{\Omega} \text{sinc}(q\|\mathbf{z} - \mathbf{x}\|)d\mathbf{z} &\approx \int_{\Omega_G} \text{sinc}(q\|\mathbf{z} - \mathbf{x}\|)d\mathbf{z} \approx \int_{\mathbb{R}} \int_{\mathbb{R}} \hat{P}(z_1, z_2)dz_2 dz_1 = \\ &= l_{x,y}^2 \int_{m_1 - \tilde{r}_1/2}^{m_1 + \tilde{r}_1/2} \int_{m_2 - \tilde{r}_2/2}^{m_2 + \tilde{r}_2/2} \hat{P}(x_{min} + q_1 l_{x,y}, y_{min} + q_2 l_{x,y})dq_2 dq_1 \approx \\ &\approx l_{x,y}^2 \frac{\tilde{r}_1}{2} \frac{\tilde{r}_2}{2} \sum_{m=1}^{n_G} w_m \sum_{n=1}^{n_G} w_n \hat{P}\left(x_{min} + q_1(x_n)l_{x,y}, y_{min} + q_2(x_m)l_{x,y}\right) = \\ &= \frac{l_{x,y}^2 \tilde{r}_1 \tilde{r}_2}{4} \sum_{m=1}^{n_G} w_m \sum_{n=1}^{n_G} w_n \hat{P}\left(x_{min} + (m_1 + \frac{\tilde{r}_1 x_n}{2})l_{x,y}, y_{min} + (m_2 + \frac{\tilde{r}_2 x_m}{2})l_{x,y}\right). \end{aligned} \quad (65)$$

In equation (65) we now have  $n_G^2$  different summands

$$\hat{P}_{m,n} := \hat{P}(x_{min} + (m_1 + \frac{\tilde{r}_1 x_n}{2})l_{x,y}, y_{min} + (m_2 + \frac{\tilde{r}_2 x_m}{2})l_{x,y}).$$

Let us break down a single summand  $\hat{P}_{m,n}, m, n \in \{1, \dots, n_G\}$ . Let

$$\begin{aligned} z_1 &:= x_{min} + (m_1 + \frac{\tilde{r}_1 x_n}{2}) l_{x,y}, \\ z_2 &:= y_{min} + (m_2 + \frac{\tilde{r}_2 x_m}{2}) l_{x,y}, \\ \tilde{i} &= \tilde{i}(m, n) := \text{floor}(m_1 + \frac{\tilde{r}_1 x_n}{2}), \\ \tilde{j} &= \tilde{j}(m, n) := \text{floor}(m_2 + \frac{\tilde{r}_2 x_m}{2}), \end{aligned}$$

then we see that  $[z_1, z_2]$  is inside the (2-dimensional) grid cell  $C$  of  $G_{x,y}$  with the 4 corner points

$$c_{i,j} := [x_{min} + i l_{x,y}, y_{min} + j l_{x,y}], \quad i \in \{\tilde{i}, \tilde{i} + 1\}, j \in \{\tilde{j}, \tilde{j} + 1\}.$$

Because  $[z_1, z_2] \in C$  and because of equation (61), there are coefficients  $a_{i,j}(m, n), i \in \{\tilde{i}, \tilde{i} + 1\}, j \in \{\tilde{j}, \tilde{j} + 1\} \in \mathbb{R}$  so that

$$\hat{P}_{m,n} = \hat{P}(z_1, z_2) = \sum_{i \in \{\tilde{i}, \tilde{i} + 1\}, j \in \{\tilde{j}, \tilde{j} + 1\}} a_{i,j} \hat{F}_{i,j}.$$

Equation (60) and equation (56) yield that

$$\begin{aligned} \hat{F}_{i,j} &= \hat{F}(x_{min} + i l_{x,y}, y_{min} + j l_{x,y}) = \\ &= \sum_{l=1}^{n_{i,j}} \int_{z_{min} + (k_{l,1} - 1/2) l_z}^{z_{min} + (k_{l,2} + 1/2) l_z} \text{sinc}(q \| [x_{min} + i l_{x,y}, y_{min} + j l_{x,y}, z_3] - \mathbf{x} \|) dz_3. \end{aligned} \tag{66}$$

From page 60 to page 63 we showed that a sufficient number for the Gaussian points to numerically integrate every integral in (66) by the Gaussian quadrature rule is

$$\tilde{n}_P := \text{ceil}(3 + 3 q l_z (r_3 + 1) / (2\pi)). \tag{67}$$

So let  $y_1, \dots, y_{\tilde{n}_P}$  be the Gaussian points and  $v_1, \dots, v_{\tilde{n}_P}$  the Gaussian weights for the Gaussian quadrature rule for  $\tilde{n}_P$  points. We then apply (64) to each integral given at (66). This yields

$$\begin{aligned} \frac{b-a}{2} &= ((z_{min} + (k_{l,2} + 1/2) l_z) - (z_{min} + (k_{l,1} - 1/2) l_z)) / 2 = \frac{k_{l,2} - k_{l,1} + 1}{2} l_z, \\ \frac{a+b}{2} &= ((z_{min} + (k_{l,1} - 1/2) l_z) + (z_{min} + (k_{l,2} + 1/2) l_z)) / 2 = z_{min} + \frac{k_{l,1} + k_{l,2}}{2} l_z, \\ &\int_{z_{min} + (k_{l,1} - 1/2) l_z}^{z_{min} + (k_{l,2} + 1/2) l_z} \text{sinc}(q \| [x_{min} + i l_{x,y}, y_{min} + j l_{x,y}, z_3] - \mathbf{x} \|) dz_3 = \\ &= \frac{k_{l,2} - k_{l,1} + 1}{2} l_z \sum_{p=1}^{\tilde{n}_P} v_p S(i, j, l, p) \end{aligned}$$

with

$$\begin{aligned}
S(i, j, l, p) &= \text{sinc}(q \| [x_{\min} + i l_{x,y}, y_{\min} + j l_{x,y}, \frac{a+b}{2} + \frac{b-a}{2} y_p] \|) \\
&= \text{sinc}(q \| [x_{\min} + i l_{x,y}, y_{\min} + j l_{x,y}, z_{\min} + \frac{k_{l,1} + k_{l,2}}{2} l_z + \frac{k_{l,2} - k_{l,1} + 1}{2} l_z y_p] \|), \\
&\quad k_{l,1} = k_{l,1}(i, j), \quad k_{l,2} = k_{l,2}(i, j).
\end{aligned} \tag{68}$$

### 2.5.5 Result and considerations

Alltogether we get

$$\begin{aligned}
\int_{\Omega} \text{sinc}(q \| \mathbf{z} - \mathbf{x} \|) d\mathbf{z} &\approx \int_{\Omega_G} \text{sinc}(q \| \mathbf{z} - \mathbf{x} \|) d\mathbf{z} = \\
&= \int_{\mathbb{R}} \int_{\mathbb{R}} \hat{F}(z_1, z_2) dz_2 dz_1 \approx \int_{\mathbb{R}} \int_{\mathbb{R}} \hat{P}(z_1, z_2) dz_2 dz_1 \approx \\
&\approx \frac{l_{x,y}^2 \tilde{r}_1 \tilde{r}_2}{4} \sum_{m=1}^{n_G} w_m \sum_{n=1}^{n_G} w_n \hat{P}\left(x_{\min} + (m_1 + \frac{\tilde{r}_1 x_n}{2}) l_{x,y}, y_{\min} + (m_2 + \frac{\tilde{r}_2 x_m}{2}) l_{x,y}\right) = \\
&= \frac{l_{x,y}^2 \tilde{r}_1 \tilde{r}_2}{4} \sum_{m=1}^{n_G} w_m \sum_{n=1}^{n_G} w_n \sum_{i \in \{\tilde{i}, \tilde{i}+1\}, j \in \{\tilde{j}, \tilde{j}+1\}} a_{i,j}(m, n) \hat{F}_{i,j} \approx \\
&\approx \frac{l_{x,y}^2 \tilde{r}_1 \tilde{r}_2}{4} \sum_{m=1}^{n_G} w_m \sum_{n=1}^{n_G} w_n \sum_{i \in \{\tilde{i}, \tilde{i}+1\}, j \in \{\tilde{j}, \tilde{j}+1\}} a_{i,j}(m, n) \sum_{l=1}^{n_{i,j}} \frac{k_{l,2} - k_{l,1} + 1}{2} l_z \sum_{p=1}^{\tilde{n}_P} v_p S(i, j, l, p) = \\
&= \frac{l_{x,y}^2 l_z \tilde{r}_1 \tilde{r}_2}{8} \sum_{m=1}^{n_G} w_m \sum_{n=1}^{n_G} w_n \sum_{i \in \{\tilde{i}, \tilde{i}+1\}, j \in \{\tilde{j}, \tilde{j}+1\}} a_{i,j}(m, n) \sum_{l=1}^{n_{i,j}} (k_{l,2} - k_{l,1} + 1) \sum_{p=1}^{\tilde{n}_P} v_p S(i, j, l, p).
\end{aligned} \tag{69}$$

We obtain our final result for  $F(\mathbf{x}, q)$ , which is

$$\begin{aligned}
F(\mathbf{x}, q) &= 2 c_\rho^2 \int_{\Omega} \text{sinc}(q \| \mathbf{z} - \mathbf{x} \|) d\mathbf{z} \approx \\
&\approx \frac{c_\rho^2 l_{x,y}^2 l_z \tilde{r}_1 \tilde{r}_2}{4} \sum_{m=1}^{n_G} w_m \sum_{n=1}^{n_G} w_n \sum_{i \in \{\tilde{i}, \tilde{i}+1\}, j \in \{\tilde{j}, \tilde{j}+1\}} a_{i,j}(m, n) \sum_{l=1}^{n_{i,j}} (k_{l,2} - k_{l,1} + 1) \sum_{p=1}^{\tilde{n}_P} v_p S(i, j, l, p)
\end{aligned} \tag{70}$$

with  $S(i, j, l, p)$  as given in equation (68).

In equation (69) we have 4 approximations. These are the considerations we have to make for each approximation in order:

1.  $\Omega \approx \Omega_G$ : We suppose that the grid was chosen fine enough that integrating over unit cuboids around each grid point in  $G$  yields about the same result as integrating over  $\Omega$ .
2.  $\hat{P}(z_1, z_2)$ : We want to evaluate  $\hat{F}(z_1, z_2)$  only on  $[z_1, z_2] \in G_{x,y}$ . Therefore we use the bilinear interpolation  $\hat{P}(z_1, z_2)$  of  $\hat{F}(z_1, z_2)$ , which is only dependent on values of  $\hat{F}(z_1, z_2)$  for  $[z_1, z_2] \in G_{x,y}$ .
3. We apply the Gaussian quadrature rule to numerically integrate over  $\int \int_{\mathbb{R} \times \mathbb{R}} \hat{P}(z_1, z_2) dz_2 dz_1$ . The number of Gaussian points  $n_G$  is an input for the algorithm, so the value of  $n_G$  should be a tradeoff between approximation accuracy and algorithmic performance.
4. We apply the Gaussian quadrature rule again for  $\hat{F}_{i,j}$ . Here we have to resolve the oscillations of the sinc-function well enough. The chosen value  $\tilde{n}_P$  for the number of Gaussian points is given at (58). It is important to note that  $\tilde{n}_P$  is (linearly) dependent on  $q$ , so evaluating  $F(\mathbf{x}, q)$  for a larger  $q$  can take more time. Additionally,  $\tilde{n}_P$  can be a significantly higher number than  $n_G$ .

### 3 Finding a body of rotation that minimizes the error $J(\Omega)$

Recall, that the measured intensity data  $I_{s,meas}(q)$  is provided and was obtained by measuring the scattering intensity  $I_s(q)$  of a particle for  $q$  in a range  $[q_{min}, q_{max}]$  with SAXS. And the intensity data of a shape  $\Omega$  is determined by evaluating  $I_s(q)$  for sufficiently many  $q$  in the same range  $[q_{min}, q_{max}]$ . The goal is to find a shape that minimizes the error measure  $J(\Omega)$  between measured and simulated data, which is given by

$$J(\Omega) = \int_{q_{min}}^{q_{max}} \left( \ln \frac{I_s(q) + \epsilon}{I_{s,meas}(q) + \epsilon} \right)^2 dq.$$

To find the optimal matching shape, we use the shape derivative of  $J(\Omega)$ , which is given by

$$\begin{aligned} dJ(\Omega; V) &= \int_{\partial\Omega} G(\mathbf{x}) \langle V(\mathbf{x}), n(\mathbf{x}) \rangle dS(\mathbf{x}), \\ G(\mathbf{x}) &= 2 \int_{q_{min}}^{q_{max}} F(\mathbf{x}, q) \frac{1}{I_s(q) + \epsilon} \ln \frac{I_s(q) + \epsilon}{I_{s,meas}(q) + \epsilon} dq, \\ F(\mathbf{x}, q) &= 2 c_\rho^2 \int_{\Omega} \text{sinc}(q \|\mathbf{z} - \mathbf{x}\|) d\mathbf{z}. \end{aligned}$$

In the previous chapter, we showed how we calculate  $I_s(q)$  and  $F(\mathbf{x}, q)$ .

#### 3.1 Limitiations on $\Omega$

Feigin and Svergun showed that the map  $\Omega \rightarrow I_s(q; \Omega)$  is not injective, which is given in the book *Structure analysis by small-angle X-ray and neutron scattering* [2] at page 112. So, reconstruction is only possible for a restricted class of shapes.

In this chapter, we restrict the class of admissible shapes, that is, we limit ourselves to finding a simply connected body of rotation  $\Omega$  that minimizes the error  $J(\Omega)$ . So, the task is to find a shape  $\tilde{\Omega}_{opt}$ , so that

$$\tilde{\Omega}_{opt} = \text{argmin}\{J(\Omega) \mid \Omega \subset \mathbb{R}^3, \Omega \text{ is a body of rotation and simply connected}\}. \quad (71)$$

##### 3.1.1 Representation of the body of rotation $\Omega$

In the first chapter we showed that  $I_s(q) = I_s(q, \Omega)$  is rotational and translational invariant with respect to  $\Omega$ . Therefore, given a rotationally symmetric shape, the axis of rotation can be chosen arbitrarily. We fix the  $z$ -axis as the axis of rotation. Then,  $\Omega$  can be described by a curve  $c : [0, L] \rightarrow \mathbb{R}^2$ ,  $c(t) = [c_1(t), c_2(t)]$  that is the boundary of an area which is rotated around the  $z$ -axis. These are the requirements for the curve  $c$ :

- $c_1(0) = 0 = c_1(L)$ ,
- $c_1(t) > 0$  for  $t \in (0, L)$ .
- The curve does not intersect itself, this means for  $t_1, t_2 \in [0, L], t_1 \neq t_2$  it holds that  $c(t_1) \neq c(t_2)$ .

The coordinates of  $c(t)$  correspond to the  $x$ - and the  $z$ -coordinate in the standard 3d coordinate system.

The following image shows the area  $\hat{\Omega}$  that is rotated to form  $\Omega$ . Hereby, the curve  $c$  is the boundary between the inner (purple) and outer (yellow) area (but not the boundary on the  $z$ -axis).

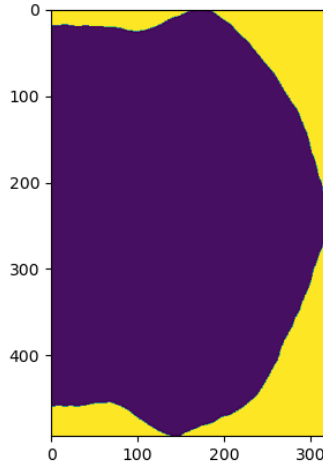


Figure 5: Area that is rotated to form  $\Omega$

We further restrict  $c$  to be a polygonal line  $\hat{P}$  with  $N_\Omega$  nodes. Let the nodes be  $(\hat{p}_1 = [\hat{x}_1, \hat{z}_1], \dots, \hat{p}_{N_\Omega} = [\hat{x}_{N_\Omega}, \hat{z}_{N_\Omega}])$ .  $\hat{P}$  has to fulfill the following conditions:

1.  $\hat{x}_1 = 0$  and  $\hat{x}_{N_\Omega} = 0$ .
2. For every  $j \in \{2, \dots, N_\Omega - 1\}$  it holds that  $\hat{x}_j > 0$ .
3. The polygonal line is not allowed to intersect itself.
4. With the help of a small constant  $l_p := \text{pixel\_length}$ , the following should hold:  $[\hat{x}_i, \hat{z}_i] = l_p [\tilde{x}_i, \tilde{z}_i]$ , whereby  $\tilde{x}_i$  and  $\tilde{z}_i$  are integers for all  $i = 1, \dots, N_\Omega$ . This means that admissible nodes lie on a grid with grid length  $l_p$ .
5.  $\hat{z}_{N_\Omega} > \hat{z}_1$

The variable  $l_p := \text{pixel\_length}$  is set in our algorithm so that the height of the rotational area is always about 500 pixels. The polygonal line

$$\tilde{P} = (\tilde{p}_1 = [\tilde{x}_1, \tilde{z}_1], \dots, \tilde{p}_N = [\tilde{x}_{N_\Omega}, \tilde{z}_{N_\Omega}]) \quad (72)$$

is again a polygonal line with nodes in  $\mathbb{Z} \times \mathbb{Z}$  that fulfills the conditions above (except condition 4).

Moreover, we define  $\hat{\Omega}$  as the area that is rotated to form  $\Omega$ , i.e. the planar region enclosed by  $\tilde{P}$  and the  $z$ -axis.

### 3.1.2 Construct $\Omega$ given a polygonal line

We show how we find grid points inside *Omega*, given a polygonal line  $\tilde{P}$  with nodes  $(\tilde{p}_1 = [\tilde{x}_1, \tilde{z}_1], \dots, \tilde{p}_{N_\Omega} = [\tilde{x}_{N_\Omega}, \tilde{z}_{N_\Omega}])$  that fulfills the previous properties. First, we check the maximum  $x$ - and  $z$ -coordinate of the polygonal line, so let

$$\tilde{x}_{max} = \max_{i=1, \dots, N_\Omega} \tilde{x}_i, \quad \tilde{z}_{max} = \max_{i=1, \dots, N_\Omega} \tilde{z}_i.$$

Then we initialize a 2-dimensional array *image* with height  $\tilde{z}_{max} - \tilde{z}_{min} + 1$  and width  $\tilde{x}_{max} + 1$ , which is our drawing area, and we set every value of this array (every pixel) to 255. Next, we set every pixel to 0 that lies on the line between 2 consecutive nodes of the polygonal line. Now that we have drawn the boundary of the rotational area, we use the *Python* function *skimage.segmentation.flood\_fill* to fill the inner area with the value 0 (*skimage* is short for *scikit-image* which is a collection of algorithms for image processing). So we get an image like in figure 5.

Now, we show how we construct  $\Omega$  by using  $\hat{\Omega}$ . This means, we have to show how we determine if a point  $\mathbf{x} \in \mathbb{R}^3$  is inside or outside  $\Omega$ . The following parameters are used to check, if a point  $\mathbf{x} = [x_1, x_2, x_3]$  is inside  $\Omega$ :

- *image*: This is the image we just obtained.
- *pixel\_length* =  $l_p$
- *topz*:  $[x = 0, z = \text{topz}]$  is the origin of our 2d coordinate system. Moreover, *pixel\_length* has to divide *topz*.

To check if a point  $\mathbf{x} = [x_1, x_2, x_3]$  lies inside  $\Omega$ , the distance

$$d = \frac{\sqrt{x_1^2 + x_2^2}}{l_p}$$

to the  $z$ -axis in “pixel units” is calculated. Furthermore,

$$h = \frac{\text{topz} - x_3}{l_p}$$



is the distance in “pixel units” of the  $z$ -coordinate of  $\mathbf{x}$  to  $topz$ , which is the  $z$ -coordinate of the origin of our 2-dimensional coordinate system.

Then,  $\mathbf{x}$  is considered to be inside  $\Omega$ , if  $0 \leq d \leq \tilde{x}_{max} + 0.5$ ,  $0 \leq h \leq \tilde{z}_{max} + 0.5$  and the following equation holds true:

$$image[round(h)][round(d)] = 0,$$

which means that the pixel in row  $round(h)$  and column  $round(d)$  is inside our 2d image (compare to figure 5).

### 3.2 Formulation of the actual inverse problem

In this section, we

1. define even more restrictions on  $\Omega$ , so that we get a practical choice for our algorithm,
2. formulate the actual inverse problem,
3. define the 3d polygonal line “counterpart” of the 2d polygonal line we defined in the last section. And we discuss the differences and similarities of those polygonal lines.
4. Finally, we briefly discuss our strategy to approach an optimal polygonal line for our inverse problem.

We limit  $\Omega$  not only to be a simply connected rotational body as described in the beginning of chapter 3.1, we furthermore limit the boundary of  $\Omega$  to be a rotated polygonal line  $P$  with vertices ( $\mathbf{p}_1 = [x_1, y_1, z_1], \dots, \mathbf{p}_{N_\Omega} = [x_{N_\Omega}, y_{N_\Omega}, z_{N_\Omega}]$ ) so that

1.  $y_i = 0$  and  $x_i \geq 0$  for all  $i \in \{1, \dots, N_\Omega\}$ , so the polygonal line lies in the half-plane with  $y = 0, x \geq 0$ .
2.  $x_1 = 0, x_{N_\Omega} = 0$  and for every  $i \in \{2, \dots, N_\Omega - 1\}$  it holds that  $x_i > 0$ .
3.  $z_{N_\Omega} < z_1$
4. The polygonal line is not allowed to intersect itself.
5. For all vertices, all coordinates are a multiple of a small constant  $l_p := pixel\_length$ . So, all vertices lie on a grid with grid size  $l_p$ .

Then the main task for this chapter and also this thesis is to find  $\Omega_{opt}$  so that

$$\Omega_{opt} = \min\{J(\Omega) \mid \Omega \subset \mathbb{R}^3, \partial\Omega \text{ is a rotated polygonal line like given above, } \Omega \text{ is bounded}\}. \quad (73)$$

Remarks:

- It is clear that such a rotated polygonal line divides the two-dimensional space in 2 parts, whereby  $\Omega$  has to be the inner (bounded) part.
- Because  $x_i > 0$  for  $i \in \{2, \dots, N_\Omega - 1\}$  and because the polygonal line is not allowed to intersect itself,  $\Omega$  has to be simply connected. (We do not proof this statement.)
- The restrictions on  $\Omega$  in equation (73) are stricter than the restrictions on  $\Omega$  in equation (71), but as  $N_\Omega \rightarrow \infty$  the restrictions in (71) (should, without proof) approach the restrictions in (73).
- $N_\Omega$  is an input parameter of our algorithm.

### 3.2.1 Similarities of the 2d and 3d polygonal lines

We want to discuss the differences and similarities of the 3d polygonal line in this section and the 2d polygonal line in section 3.1.1. When we consider the mappings

$$\begin{aligned}\Phi_1 : \mathbb{R}_{\geq 0} \times \mathbb{R} &\rightarrow \mathbb{R}_{\geq 0} \times \{0\} \times \mathbb{R}, & \Phi_1([x, z]) &= [x, 0, \text{top}z - z], \\ \Phi_2 : \mathbb{R}_{\geq 0} \times \{0\} \times \mathbb{R} &\rightarrow \mathbb{R}_{\geq 0} \times \mathbb{R}, & \Phi_2([x, 0, z]) &= [x, \text{top}z - z],\end{aligned}\tag{74}$$

we see that  $\Phi_1 = \Phi_2^{-1}$ . We decide to flip the  $z$ -coordinate, because the vertical coordinate in 2d images (in *Python*) usually points downwards, whereas the vertical coordinate in 3d graphics usually points upwards. And with this decision, when we show an image/graphic of the 2d or 3d polygonal line by a standard *Python* function, the first point of the polygonal line always lie “above” the last point of the polygonal line.

For a polygonal line  $\hat{P} = (\hat{p}_1 = [\hat{x}_1, \hat{z}_1], \dots, \hat{p}_{N_\Omega} = [\hat{x}_{N_\Omega}, \hat{z}_{N_\Omega}])$  that fulfills the 5 conditions given in section 3.1.1 let

$$\Phi_1(\hat{P}) := (\Phi_1(\hat{p}_1), \dots, \Phi_1(\hat{p}_{N_\Omega}))$$

and for a polygonal line  $P = (\mathbf{p}_1 = [x_1, y_1, z_1], \dots, \mathbf{p}_{N_\Omega} = [x_{N_\Omega}, y_{N_\Omega}, z_{N_\Omega}])$  that fulfills the 5 conditions given in this section let

$$\Phi_2(P) := (\Phi_2(\mathbf{p}_1), \dots, \Phi_2(\mathbf{p}_{N_\Omega})).$$

It is obvious that  $\Phi_1(\hat{P})$  and  $\Phi_2(P)$  are well defined (the vertices of the polygonal lines are in the domain of  $\Phi_1$  respectively  $\Phi_2$ ). Moreover it is also obvious that  $\Phi_1(\hat{P})$  is a polygonal line in 3d and  $\Phi_2(P)$  is a polygonal line in 2d. We want to proof that  $\Phi_1(\hat{P})$  fulfills all conditions of the polygonal line in this section, and  $\Phi_2(P)$  fulfills all conditions of the polygonal line in section 3.1.1.

**Lemma 3.1.**  $\Phi_1(\hat{P}) = ([\tilde{x}_1, \tilde{y}_1, \tilde{z}_1], \dots, [\tilde{x}_{N_\Omega}, \tilde{y}_{N_\Omega}, \tilde{z}_{N_\Omega}])$  fulfills all conditions on the polygonal line in the beginning of this section.

*Proof.* When we observe  $\Phi_1([x, z]) = [x, 0, \text{top}z - z]$ , we see that

- $\tilde{g}_i = 0$  for  $i \in \{1, \dots, N_\Omega\}$ .
- Because  $\hat{x}_i = 0 = \hat{x}_{N_\Omega}$  it also holds true that  $\tilde{x}_1 = 0 = \tilde{x}_{N_\Omega}$
- Because  $\hat{x}_i > 0$  for  $i \in \{2, \dots, N_\Omega - 1\}$ , it also hold true that  $\tilde{x}_i > 0$ .
- Because  $\hat{z}_1 < \hat{z}_{N_\Omega}$ ,  $\tilde{z}_1 = \text{topz} - \hat{z}_1 > \text{topz} - \hat{z}_{N_\Omega} = \tilde{z}_{N_\Omega}$ .

Let  $i \in \{1, \dots, N_\Omega\}$ . Because of the 4<sup>th</sup> condition in section 3.1.1,  $l_p = \text{pixel\_length}$  divides  $\hat{x}_i = \tilde{x}_i$  and  $\hat{z}_i$ . Because  $l_p$  also divides  $\text{topz}$ ,  $l_p$  divides  $\text{topz} - \hat{z}_i = \tilde{z}_i$ . Therefore, all coordinates are a multiple of  $l_p$ .

Furthermore,  $\Phi_1$  is injective, so  $\Phi_1(\hat{P})$  also does not intersect itself.  $\square$

**Lemma 3.2.**  $\Phi_2(P) = ([\tilde{x}_1, \tilde{z}_1], \dots, [\tilde{x}_{N_\Omega}, \tilde{z}_{N_\Omega}])$  fulfills all conditions on the polygonal line in section 3.1.1.

*Proof.* Analog to the proof in Lemma 3.1.  $\square$

**Lemma 3.3.**  $\Phi_1$  and  $\Phi_2$  are isometries with respect to the Euclidean norm in 2d and 3d.

*Proof.* Let  $[x_1, z_1], [x_2, z_2] \in \mathbb{R}_{\geq 0} \times \mathbb{R}$ , then

$$\begin{aligned} \|\Phi_1([x_1, z_1]) - \Phi_1([x_2, z_2])\|_2 &= \|[x_1, 0, \text{topz} - z_1] - [x_2, 0, \text{topz} - z_2]\|_2 = \\ &= \|[x_1 - x_2, 0, z_2 - z_1]\|_2 = \sqrt{(x_1 - x_2)^2 + (z_1 - z_2)^2} = \\ &= \|[x_1 - x_2, z_1 - z_2]\|_2 = \|[x_1, z_1] - [x_2, z_2]\|_2. \end{aligned}$$

Furthermore, let  $[x_3, y_3, z_3], [x_4, y_4, z_4] \in \mathbb{R}_{\geq 0} \times \{0\} \times \mathbb{R}$ , then  $y_3 = y_4 = 0$  and

$$\begin{aligned} \|\Phi_2([x_3, y_3, z_3]) - \Phi_2([x_4, y_4, z_4])\|_2 &= \|[x_3, \text{topz} - z_3] - [x_4, \text{topz} - z_4]\|_2 = \\ &= \|[x_3 - x_4, z_4 - z_3]\|_2 = \sqrt{(x_3 - x_4)^2 + (z_3 - z_4)^2} = \\ &= \|[x_3 - x_4, 0 - 0, z_3 - z_4]\|_2 = \|[x_3, y_3, z_3] - [x_4, y_4, z_4]\|_2. \end{aligned}$$

$\square$

When we have a 2d polygon  $\hat{P}$  that satisfies all conditions in section 3.1.1 and we change the coordinates of the nodes of  $\hat{P}$  so that we obtain a new polygonal line  $\hat{P}_2$  that also satisfies all conditions in section 3.1.1, we know because of the last 2 lemmas that:

- $\Phi_1(\hat{P})$  and  $\Phi_1(\hat{P}_2)$  are 3d polygonal lines that satisfy the conditions in the beginning of this section.
- Because  $\Phi_1$  is an isometry, the distances between the nodes and also the angles at every node of  $\hat{P}$  and  $\Phi_1(\hat{P})$  are all equal (the same holds true for  $\hat{P}_2$  and  $\Phi_1(\hat{P}_2)$ ).
- Because  $\Phi_1$  is an isometry, for every normal vector  $n(\mathbf{x})$  to  $\hat{P}$  on  $\mathbf{x} \in \hat{P} \setminus \{\hat{p}_1, \dots, \hat{p}_{N_\Omega}\}$ ,  $\Phi_1(n(\mathbf{x}))$  is also a normal vector on  $\Phi_1(\mathbf{x}) \in \Phi_1(\hat{P})$ .

When we want to change the boundary of  $\Omega$  which is described by the polygonal line  $P$ , we can therefore alter the 2d polygonal line  $\hat{P} := \Phi_2(P)$  to get  $\hat{P}_2$ . To get the changed boundary of  $\Omega$ , we then have to retrieve the 3d polygonal line  $P_2 := \Phi_1(\hat{P}_2)$ , which is the new polygonal line that is rotated to form  $\Omega$ .

### 3.2.2 Strategy to approach $\Omega_{opt}$

We start with an initial polygonal line  $\hat{P}$ . In the next figure there are 3 different starting options that were used in different test runs:

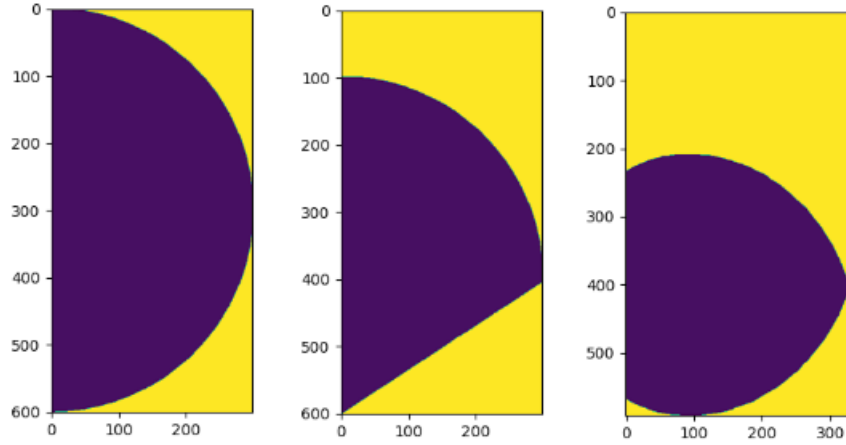


Figure 6: Different initial polygonal lines

We used different initial polygonal lines, because of 2 reasons:

- The algorithm has problems to break the symmetry between upper and lower half of the polygonal line.
- The algorithm sometimes converged to a local minimum that was not a global minimum.

After we choose an initial polygonal line, we iteratively change the boundary of  $\Omega$  by shifting the nodes of the polygonal lines. In one iteration (one change of the boundary), the following happens:

1. We have to calculate  $I_s(q, \Omega)$  for  $q$  in the range  $[q_{min}, q_{max}]$  and calculate the error  $J(\Omega)$ .
2. To decrease  $J(\Omega)$ , we want to obtain a field  $V(\mathbf{x})$  which provides us information, in which direction and how much we should deform  $\Omega$ . In section 3.4 we discuss how we obtain such a field  $V(\mathbf{x})$ .

3. In section 3.5 we discuss different step size choices for the deformation of  $\Omega$  in direction  $V(\mathbf{x})$ . Furthermore, we discuss how the best step size is chosen and how the update of  $\Omega$  works technically (update of the 2d polygonal line  $\hat{P}$ ).

### 3.3 Calculate $G(\mathbf{x})$ for $\mathbf{x} \in \partial\Omega$

Before calculating

$$G(\mathbf{x}) = 2 \int_{q_{min}}^{q_{max}} F(\mathbf{x}, q) \frac{1}{I_s(q) + \epsilon} \ln \frac{I_s(q) + \epsilon}{I_{s,meas}(q) + \epsilon} dq$$

for some  $\mathbf{x} \in \partial\Omega$ , the actual  $I_s(q)$ -values for  $q$  in the range  $[q_{min}, q_{max}]$  have to be calculated.

The input measurement data are  $N_q \in \mathbb{N}$  equidistant  $q$ -values ( $q_{min} = q_0, \dots, q_{N_q-1} = q_{max}$ ) in the range  $[q_{min}, q_{max}]$  and the  $N_q$  corresponding measured  $I_{s,meas}(q_i)$  values ( $i = 0, \dots, N_q - 1$ ), which were measured e.g. by SAXS. The  $N_q$  tuples of  $q_i$ - and  $I_s(q_i)$ -values are stored in the variable *isq\_meas*.

To calculate  $I_s(q)$ , we take the same  $N_q$  equidistant  $q$ -values ( $q_0, \dots, q_{N_q-1}$ ) and calculate  $I_s(q_i)$  for each of them like we described it in section 2.4. The representation of  $\Omega$  and how we determine if some point is inside or outside  $\Omega$  is described in section 3.1.1. The  $N_q$  tuples of  $q_i$ - and  $I_s(q_i)$ -values are then stored in the variable *isq\_act*.

The following parameters are the input parameters to calculate  $G(\mathbf{x})$  for some boundary point  $\mathbf{x} \in \partial\Omega$  along the ideas presented in Chapter 2:

- *boundary\_point* =  $\mathbf{x}$ : The coordinates of  $\mathbf{x}$  in an 3-dimensional array
- *isq\_meas*: The  $N_q$  tuples of  $q$ -values and corresponding measurement data  $I_{s,meas}(q)$
- *isq\_act*: The  $N_q$  tuples of  $q$ -values and corresponding calculated  $I_s(q)$ -data.
- *epsilon*: A small value greater than 0 that is important for numerical stability

Then, the following code prepares the  $(I_s(q) + \epsilon)$ - and the  $(I_{s,meas}(q) + \epsilon)$ -data we use to calculate  $G(\mathbf{x})$ :

```
qVals = [i[0] for i in isq_act]
isqActVals = [(np.max([i[1], 0.0]) + epsilon) for i in isq_act]
isqMeasVals = [(np.max([i[1], 0.0]) + epsilon) for i in isq_meas]
```

We describe some basic construction principles of the code:

- $np.max(i[1], 0.0)$  for  $i$  in  $isq\_act$ : Neither the measured nor the actual  $I_s(q)$ -data should have values smaller than 0. But because of numerical errors and because some concrete values are really small (e.g.  $I_s(q) < 10^{-12}$  for some  $q$ ) it can happen that the calculation of  $I_s(q)$  returns a negative number - therefore take the value 0 for every value of  $I_s(q)$  that is smaller than 0.
- $epsilon$  is added to every value so that every value in the lists  $isqActVals$  and  $isqMeasVals$  is greater than 0 - and so the division  $\frac{1}{I_s(q)+\epsilon}$  and the evaluation of  $\ln \frac{I_s(q)+\epsilon}{I_{s,meas}(q)+\epsilon}$  always works.

Now we have to numerically calculate the integral

$$\int_{q_{min}}^{q_{max}} F(\mathbf{x}, q) \frac{1}{I_s(q) + \epsilon} \ln \frac{I_s(q) + \epsilon}{I_{s,meas}(q) + \epsilon} dq. \quad (75)$$

Because the argument  $A(q) := F(\mathbf{x}, q) \frac{1}{I_s(q) + \epsilon} \ln \frac{I_s(q) + \epsilon}{I_{s,meas}(q) + \epsilon}$  of this integral is not very smooth (lots of bumps around 0), typical numerical integration methods like the Gaussian quadrature formula that approximate the argument by polynomials are not the way to go. We instead use (almost) equidistant points where we evaluate the argument  $A(q)$  and uniform weights (at least for the points that are not on the boundary), which is a variant of the simple rectangular rule.

We introduce the variable  $d_q := takeEveryNthQVal \in \mathbb{N}$  and evaluate  $A(q)$  only on  $q = q_i$  where  $d_q$  divides  $i$  and for the last point  $q_{N_q}$ . We set

$$\begin{aligned} N_G &:= \text{ceil}((N_q - 1)/d_q) + 1, \\ \tilde{q}_i &:= q_{i \cdot d_q} \quad \text{for } i = 0, \dots, N_G - 2, \\ \tilde{q}_{N_G-1} &:= q_{N_q-1}. \end{aligned} \quad (76)$$

Every two consecutive values in  $\{\tilde{q}_0, \dots, \tilde{q}_{N_G-2}\}$  are equidistant whereas the distance between  $\tilde{q}_{N_G-2}$  and  $\tilde{q}_{N_G-1}$  can be lower. In the case that all points (also the last 2) are equidistant, we use the following numerical integration to evaluate an approximation to the integral (75):

$$\int_{q_{min}}^{q_{max}} A(q) dq = \frac{q_{max} - q_{min}}{N_G - 1} \left( \frac{A(\tilde{q}_0)}{2} + \sum_{i=1}^{N_G-2} A(\tilde{q}_i) + \frac{A(\tilde{q}_{N_G-1})}{2} \right). \quad (77)$$

In the case that the last two points are not equidistant, the weight of the last two values is proportionally smaller and then the sum of all weights is also smaller than  $N_G - 1$  (some non-integer value in the range  $(N_G - 2, N_G - 1)$ ).

The question that arises is how accurate we want to numerically integrate expression (75) and therefore how we set the variable  $d_q = takeEveryNthQVal$ . These are the considerations:

- We do not use a higher order numerical integration technique like the Gaussian quadrature formula, so we need a larger  $N_G$  (therefore a smaller  $d_q$ ) to get good enough accuracy.
- Evaluating  $G(\mathbf{x})$  is a bottleneck for the algorithm, so we want  $N_G$  to be as small as possible ( $d_q$  as large as possible).
- In the next section we will conclude that the most important information about  $G(\mathbf{x})$  is whether  $G(\mathbf{x}) \leq 0$  or  $G(\mathbf{x}) > 0$ . So it is necessary that we are more exact if the value of  $G(\mathbf{x})$  is approximately 0.

Therefore, our strategy is as follows:

1. Set  $d_{q,1} := \text{floor}(N_q/20)$ ,  $d_{q,2} := \max(3, \text{floor}(N_q/80))$  and  $d_{q,3} := 1$ . Set  $d_q := d_{q,1}$ .
2. Evaluate  $N_G$ ,  $\tilde{q}_i$  and  $A(\tilde{q}_i)$  for  $i = 0, \dots, N_G$  like in equation (76).
3. Evaluate  $A_{max} := \max_{i=0, \dots, N_G-1} |A(\tilde{q}_i)|$ . Then use the numerical integration formula (77) to evaluate (75) and set this value to the variable  $G_{num}$ .
4. Calculate the quotient  $Q := \frac{G_{num}}{(q_{max} - q_{min})A_{max}}$ . This value tells us how close  $G_{num}$  is to 0 in relation to the largest argument.
5. If  $Q > 1/20$  or ( $Q > 1/200$  and  $d_q = d_{q,2}$ ) or  $d_q = 1$  we are done and return the value  $G(\mathbf{x}) \approx 2G_{num}$ .  
Else, if  $Q > 1/200$  and  $d_q = d_{q,1}$ , we set  $d_q := d_{q,2}$  and we go to (2.).  
Else (if  $Q \leq 1/200$  and  $d_q \neq 1$ ), we set  $d_q := 1$  and we go to (2.).

The constants we used (e.g. 20, 80 and 200) are chosen after testing different values. For the test we had 201 different  $q$ -values (this means  $N_q = 201$ ).

### 3.4 Choice of $V$ in $dJ(\Omega, V)$

The vector field  $V$  determines the length and the direction of the shape change of  $\Omega$  in one step. When we consider

$$dJ(\Omega; V) = \int_{\partial\Omega} G(\mathbf{x}) \langle V(\mathbf{x}), n(\mathbf{x}) \rangle dS(\mathbf{x}) \quad (78)$$

we see that it is sufficient to define  $V$  on  $\partial\Omega$ . Moreover, we want to achieve a balance between the following goals:

1. Make  $-dJ(\Omega, V)$  as large as possible to obtain the maximum descent of  $J(\Omega)$  when we change the boundary of  $\Omega$  in direction (and length)  $V$ .
2.  $\|V(\mathbf{x})\|$  should be bounded by some constant  $c_{V,1} \in \mathbb{R}$ , and also  $\int_{\partial\Omega} \|V(\mathbf{x})\| dS(\mathbf{x})$  should be bounded by some constant  $c_{V,2} \in \mathbb{R}$ , because in one step (one change of the boundary of  $\Omega$ ) we want to make only small changes locally on  $\partial\Omega$  and also only small changes on the boundary in total.

3.  $V$  should be smooth on most of the points, because we want to make smooth changes on the boundary. At least,  $V(\mathbf{x})$  should be smooth whenever  $n(\mathbf{x})$  is well-defined on  $\partial\Omega$ .

When we consider

$$V(\mathbf{x}) = l(\mathbf{x})\mathbf{d}(\mathbf{x}), \quad \text{for } x \in \partial\Omega,$$

whereby  $l(\mathbf{x})$  is the length of the vector  $V(\mathbf{x})$  (so  $l(\mathbf{x}) > 0$ ) and  $\mathbf{d}(\mathbf{x})$  is the direction of  $V(\mathbf{x})$  (so  $\|\mathbf{d}(\mathbf{x})\| = 1$ ), we see that:

- The second goal only depends only on  $l(\mathbf{x})$ .
- $l(\mathbf{x})$  has to be smooth on  $\partial\Omega$ .
- With given  $l(\mathbf{x})$  and without considering the third goal, we can obtain a positive value for  $-dJ(\Omega, V)$  by setting  $\mathbf{d}(\mathbf{x}) := -\text{sign}(G(\mathbf{x}))n(\mathbf{x})$ .
- When we also consider the third goal, we want that  $\mathbf{d}(\mathbf{x})$  is smooth whenever the sign of  $G(\mathbf{x})$  does not change. Because  $n(\mathbf{x})$  is not necessarily defined on all points  $\mathbf{x} \in \partial\Omega$ , we have to define some  $\bar{n}(\mathbf{x})$  so that  $\bar{n}$  is smooth on  $\partial\Omega$  and  $\bar{n}(\mathbf{x}) \approx n(\mathbf{x})$  for every  $\mathbf{x} \in \partial\Omega$  where  $n(\mathbf{x})$  is defined. Then, the choice  $\mathbf{d}(\mathbf{x}) := -\text{sign}(G(\mathbf{x}))\bar{n}(\mathbf{x})$  yields that  $\mathbf{d}(\mathbf{x})$  is smooth whenever the sign of  $G(\mathbf{x})$  does not change. (An exact definition of  $\bar{n}(\mathbf{x})$  follows later in this section.)
- And whenever the sign of  $G(\mathbf{x})$  changes, the length  $l(\mathbf{x})$  has to be 0 to ensure smoothness also in points where the sign of  $G(\mathbf{x})$  changes.
- Considering the first and second goal, we see that it is of advantage to define  $l(\mathbf{x})$  larger whenever  $|G(\mathbf{x})|$  is larger and smaller whenever  $|G(\mathbf{x})|$  is smaller, because whenever  $|G(\mathbf{x})|$  is larger, the point  $\mathbf{x}$  has a larger impact on  $-dJ(\Omega, V)$ .
- With all considerations above, we see that  $V(\mathbf{x}) = l(\mathbf{x})\mathbf{d}(\mathbf{x})$  is smooth (third goal), the directions  $\mathbf{d}(\mathbf{x})$  and the length  $l(\mathbf{x})$  are chosen so that  $-dJ(\Omega, V)$  is near the maximal obtainable value (first goal)

Alltogether, for  $x \in \partial\Omega$  a good choice for  $V(\mathbf{x})$  would be

$$\begin{aligned} V(\mathbf{x}) &= l(\mathbf{x})\mathbf{d}(\mathbf{x}), \\ \mathbf{d}(\mathbf{x}) &= -\text{sign}(G(\mathbf{x}))\bar{n}(\mathbf{x}), \\ l(\mathbf{x}) &= |G(\mathbf{x})|, \end{aligned}$$

but there are two remaining problems:

- We still have to define  $\bar{n}(\mathbf{x})$
- We do not want to evaluate  $G(\mathbf{x})$  on too many points  $\mathbf{x} \in \partial\Omega$ , because evaluating  $G(\mathbf{x})$  is a bottleneck of our algorithm.

These problems are addressed in the following sections.



### 3.4.1 Smooth approximation of the normal vector on $\partial\Omega$

We know that  $\Omega$  is a rotational body and the boundary is a rotated polygonal line  $P$  with vertices  $(\mathbf{p}_1 = (x_1, y_1, z_1), \dots, \mathbf{p}_{N_\Omega} = (x_{N_\Omega}, y_{N_\Omega}, z_{N_\Omega}))$  that fulfills all conditions at the beginning of section 3.2. First, we want to define the approximation  $\hat{n}$  of the normal vector only on  $P$ . In section 3.2.1 we saw that  $\hat{P} := \Phi_2(P)$  is a polygonal line in 2d and because  $\Phi_2$  is a isometry, it is possible to define the approximation  $\hat{n}$  of the normal vector in the 2d setting and transfer it back to the 3d setting with the isometry  $\Phi_1 = \Phi_2^{-1}$  ( $\Phi_1$  and  $\Phi_2$  are defined at (74)).

In the 2d setting, we first define the approximation  $\hat{n}$  of the normal vector on every node  $\hat{p}_i = [\hat{x}_i, \hat{z}_i]$  of  $\hat{P}$ ,  $i = 1, \dots, N_\Omega$ :

$$\begin{aligned} \hat{n}(\hat{p}_1) &:= [0, -1], \\ \hat{n}(\hat{p}_i) &:= \frac{[\hat{z}_{i+1} - \hat{z}_{i-1}, \hat{x}_{i-1} - \hat{x}_{i+1}]}{\|[\hat{z}_{i+1} - \hat{z}_{i-1}, \hat{x}_{i-1} - \hat{x}_{i+1}]\|} \quad \text{for } i \in \{2, \dots, N_\Omega - 1\}, \\ \hat{n}(\hat{p}_{N_\Omega}) &:= [0, 1]. \end{aligned}$$

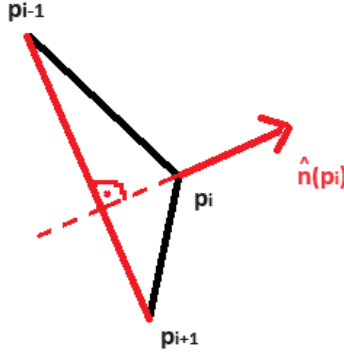


Figure 7: Approximation  $\hat{n}$  of the normal vector on  $p_i$

Every other point  $\mathbf{x} \in \mathbb{R}^2$  on  $\hat{P}$  lies on the connecting line between 2 nodes of  $\hat{P}$ , so there exist a  $j \in \{1, \dots, N_\Omega - 1\}$  and a  $\lambda \in [0, 1]$  so that  $\mathbf{x} = \lambda \hat{p}_j + (1 - \lambda) \hat{p}_{j+1}$ . For this  $\mathbf{x}$  we define  $\hat{n}(\mathbf{x})$  as follows:

$$\hat{n}(\mathbf{x}) = \hat{n}(\lambda \hat{p}_j + (1 - \lambda) \hat{p}_{j+1}) := \frac{\lambda \hat{n}(\hat{p}_j) + (1 - \lambda) \hat{n}(\hat{p}_{j+1})}{\|\lambda \hat{n}(\hat{p}_j) + (1 - \lambda) \hat{n}(\hat{p}_{j+1})\|} \quad (79)$$

We show that  $\hat{n}$  is well defined and continuous, and we also show some properties of  $\hat{n}$ .

**Lemma 3.4.**  *$\hat{n}$  is well defined on  $\hat{P}$  and  $\|\hat{n}(\mathbf{x})\| = 1$  for every  $\mathbf{x} \in \hat{P}$ .*

*Proof.* For  $i = 2, \dots, N_\Omega - 1$  we know that  $\hat{p}_{i-1}$  and  $\hat{p}_{i+1}$  are not the same point, because the polygonal line is not allowed to intersect itself. Therefore,  $\|[\hat{z}_{i+1} - \hat{z}_{i-1}, \hat{x}_{i-1} - \hat{x}_{i+1}]\| > 0$  and  $\hat{p}_i$  is well defined. Moreover it holds that  $\|\hat{n}(\hat{p}_i)\| = 1$  for all  $i = 1, \dots, N_\Omega$ .

We show that  $\lambda\hat{n}(\hat{p}_j) + (1-\lambda)\hat{n}(\hat{p}_{j+1}) \neq [0, 0]$  for  $\lambda \in [0, 1], j = 1, \dots, N_\Omega - 1$ . Suppose there exist a  $j \in \{1, \dots, N_\Omega - 1\}$  and a  $\lambda \in [0, 1]$  so that

$$\lambda\hat{n}(\hat{p}_j) + (1-\lambda)\hat{n}(\hat{p}_{j+1}) = [0, 0].$$

It follows that

$$\lambda\hat{n}(\hat{p}_j) = -(1-\lambda)\hat{n}(\hat{p}_{j+1})$$

and

$$\lambda = \lambda\|\hat{n}(\hat{p}_j)\| = \|(1-\lambda)\hat{n}(\hat{p}_{j+1})\| = (1-\lambda)\|\hat{n}(\hat{p}_{j+1})\| = (1-\lambda)$$

and therefore  $\lambda = 0.5$ . So,

$$\frac{[\hat{z}_{j+1} - \hat{z}_{j-1}, \hat{x}_{j-1} - \hat{x}_{j+1}]}{\|[\hat{z}_{j+1} - \hat{z}_{j-1}, \hat{x}_{j-1} - \hat{x}_{j+1}]\|} = \hat{n}(\hat{p}_j) = -\hat{n}(\hat{p}_{j+1}) = -\frac{[\hat{z}_{j+2} - \hat{z}_j, \hat{x}_j - \hat{x}_{j+2}]}{\|[\hat{z}_{j+2} - \hat{z}_j, \hat{x}_j - \hat{x}_{j+2}]\|}.$$

Let  $\alpha := \|[\hat{z}_{j+1} - \hat{z}_{j-1}, \hat{x}_{j-1} - \hat{x}_{j+1}]\|$  and  $\beta := \|[\hat{z}_{j+2} - \hat{z}_j, \hat{x}_j - \hat{x}_{j+2}]\|$ , then

$$\begin{aligned}\beta(\hat{z}_{j+1} - \hat{z}_{j-1}) + \alpha(\hat{z}_{j+2} - \hat{z}_j) &= 0, \\ \beta(\hat{x}_{j-1} - \hat{x}_{j+1}) + \alpha(\hat{x}_j - \hat{x}_{j+2}) &= 0.\end{aligned}$$

So we have

$$\begin{aligned}\hat{z}_j - \hat{z}_{j+2} &= \frac{\beta}{\alpha}(\hat{z}_{j+1} - \hat{z}_{j-1}), \\ \hat{x}_j - \hat{x}_{j+2} &= \frac{\beta}{\alpha}(\hat{x}_{j+1} - \hat{x}_{j-1}), \hat{p}_j - \hat{p}_{j+2} = \frac{\beta}{\alpha}(\hat{p}_{j+1} - \hat{p}_{j-1})\end{aligned}$$

and therefore, the vectors  $\mathbf{v}_1 := (\hat{p}_j - \hat{p}_{j+2})$  and  $\mathbf{v}_2 := (\hat{p}_{j+1} - \hat{p}_{j-1})$  are parallel and have the same direction ( $\beta/\alpha > 0$ ). We analyze the vector  $\mathbf{v}_3 := (\hat{p}_j - \hat{p}_{j+1})$ .

$\mathbf{v}_3$  is not parallel to  $\mathbf{v}_1$  (and  $\mathbf{v}_2$ ), because then all 4 points  $\hat{p}_l, l = j-1, \dots, j+2$  would lie on the same line  $\mu(k) = \hat{p}_j + k\mathbf{v}_2, k \in \mathbb{R}$ . For  $l = j-1, \dots, j+2$ , let  $k_l \in \mathbb{R}$  so that  $\mu(k_l) = \hat{p}_l$ . When we order the 4 points by the parameter  $k$ , we see that  $k_{j+2} < k_j$  and  $k_{j-1} < k_j$ . The only way the polygonal line does not intersect itself is, when all 4 points would lie in the same order (so either  $k_{j-2} < k_{j-1} < k_j < k_{j+1}$  or  $k_{j-2} > k_{j-1} > k_j > k_{j+1}$ ), which is not the case.

Therefore,  $\mathbf{v}_3$  is not parallel to  $\mathbf{v}_1$  and we can construct the following trapezoid with the help of the vectors  $\mathbf{v}_1, \mathbf{v}_2$  and  $\mathbf{v}_3$  that are connecting all 4 points  $\hat{p}_l, l = j-1, \dots, j+2$ :

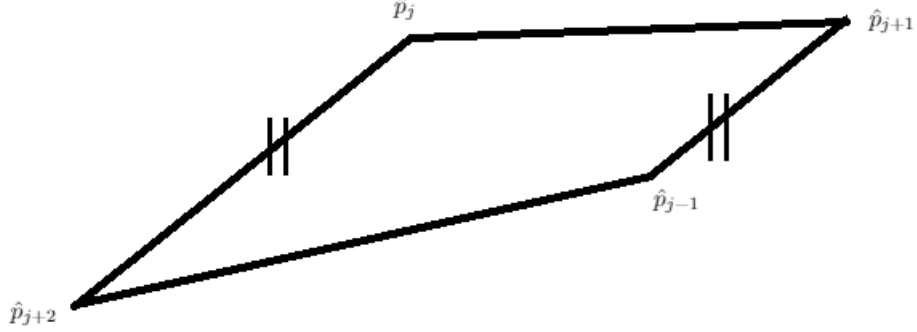


Figure 8: Sketch of a trapezoid

The 2 parallel sides of the trapezoid are the side from  $\hat{p}_{j+2}$  to  $\hat{p}_j$  and the side from  $\hat{p}_{j-1}$  to  $\hat{p}_{j+1}$  (because  $\mathbf{v}_1$  and  $\mathbf{v}_2$  are parallel). Because  $\mathbf{v}_1$  and  $\mathbf{v}_2$  also have the same direction, the remaining two sides are the side from  $\hat{p}_{j+2}$  to  $\hat{p}_{j-1}$  and the side from  $\hat{p}_j$  to  $\hat{p}_{j+1}$ .

The 2 diagonals of the trapezoid are both subsets of  $\hat{P}$ , because they are connection lines between 2 consecutive points of  $\hat{P}$ . Diagonals of a trapezoid always have an intersection point, and therefore  $\hat{P}$  intersects itself. So we found a contradiction to the assumption that

$$\lambda \hat{n}(\hat{p}_j) + (1 - \lambda) \hat{n}(\hat{p}_{j+1}) = [0, 0].$$

Therefore,  $\lambda \hat{n}(\hat{p}_j) + (1 - \lambda) \hat{n}(\hat{p}_{j+1}) \neq [0, 0]$  for  $\lambda \in [0, 1], j = 1, \dots, N_\Omega - 1$  and therefore,  $\hat{n}(\mathbf{x})$  is well defined for every point  $\mathbf{x} \in \hat{P}$ . Moreover, it is clear that  $\|\hat{n}(\mathbf{x})\| = 1$  for every  $\mathbf{x} \in \hat{P}$ . □

**Lemma 3.5.**  *$\hat{n}$  is continuous on  $\hat{P}$ .*

*Proof.* Let  $\mathbf{x} \in \hat{P}$ . We have to consider 2 cases, depending on whether  $\mathbf{x}$  is a node of  $\hat{P}$ .

Case 1,  $\mathbf{x} = \hat{p}_j$  for a  $j \in \{1, \dots, N_\Omega\}$ : If  $j = 1$  or  $j = N_\Omega$ , there is only one way on  $\hat{P}$  to approach  $\mathbf{x}$ . Otherwise, we can approach  $\mathbf{x} = \hat{p}_j$  on  $\hat{P}$  on the connection line

$$l_1 = \{\lambda \hat{p}_{j-1} + (1 - \lambda) \hat{p}_j \mid \lambda \in [0, 1]\}$$

between  $\hat{p}_{j-1}$  and  $\hat{p}_j$  and on the connection line

$$l_2 = \{\lambda \hat{p}_j + (1 - \lambda) \hat{p}_{j+1} \mid \lambda \in [0, 1]\}$$

between  $\hat{p}_j$  and  $\hat{p}_{j+1}$  in the following way:

$$\begin{aligned}\mathbf{x} &= \hat{p}_j = \lim_{\lambda \searrow 0} \lambda \hat{p}_{j-1} + (1 - \lambda) \hat{p}_j, \\ \mathbf{x} &= \hat{p}_j = \lim_{\lambda \nearrow 1} \lambda \hat{p}_j + (1 - \lambda) \hat{p}_{j+1}.\end{aligned}$$

When we apply the definition of  $\hat{n}(\mathbf{x})$  (Equation (79)) on each point on  $l_1$  and  $l_2$  we obtain

$$\begin{aligned}\lim_{\lambda \searrow 0} \hat{n}(\lambda \hat{p}_{j-1} + (1 - \lambda) \hat{p}_j) &= \lim_{\lambda \searrow 0} \frac{\lambda \hat{n}(\hat{p}_{j-1}) + (1 - \lambda) \hat{n}(\hat{p}_j)}{\|\lambda \hat{n}(\hat{p}_{j-1}) + (1 - \lambda) \hat{n}(\hat{p}_j)\|} = \frac{\hat{n}(\hat{p}_j)}{\|\hat{n}(\hat{p}_j)\|} = \hat{n}(\hat{p}_j), \\ \lim_{\lambda \nearrow 1} \hat{n}(\lambda \hat{p}_j + (1 - \lambda) \hat{p}_{j+1}) &= \lim_{\lambda \nearrow 1} \frac{\lambda \hat{n}(\hat{p}_j) + (1 - \lambda) \hat{n}(\hat{p}_{j+1})}{\|\lambda \hat{n}(\hat{p}_j) + (1 - \lambda) \hat{n}(\hat{p}_{j+1})\|} = \frac{\hat{n}(\hat{p}_j)}{\|\hat{n}(\hat{p}_j)\|} = \hat{n}(\hat{p}_j).\end{aligned}$$

Therefore,  $\hat{n}$  is continuous on  $\mathbf{x} = \hat{p}_j$ .

Case 2, there exists a  $j \in \{1, \dots, N_\Omega - 1\}$  so that  $\mathbf{x}$  lies on a connection line between  $\hat{p}_j$  and  $\hat{p}_{j+1}$  (and  $\mathbf{x} \neq p_j, \mathbf{x} \neq p_{j+1}$ ). So there exists a  $\lambda_0 \in (0, 1)$  so that  $\mathbf{x} = \lambda_0 \hat{p}_j + (1 - \lambda_0) \hat{p}_{j+1}$ . The only ways we can approach  $\mathbf{x}$  on  $\hat{P}$  are

$$\begin{aligned}\mathbf{x} &= \lim_{\lambda \searrow \lambda_0} \lambda \hat{p}_j + (1 - \lambda) \hat{p}_{j+1}, \\ \mathbf{x} &= \lim_{\lambda \nearrow \lambda_0} \lambda \hat{p}_j + (1 - \lambda) \hat{p}_{j+1}.\end{aligned}\tag{80}$$

Here, it holds true that

$$\begin{aligned}\lim_{\lambda \searrow \lambda_0} \hat{n}(\lambda \hat{p}_j + (1 - \lambda) \hat{p}_{j+1}) &= \lim_{\lambda \searrow \lambda_0} \frac{\lambda \hat{n}(\hat{p}_j) + (1 - \lambda) \hat{n}(\hat{p}_{j+1})}{\|\lambda \hat{n}(\hat{p}_j) + (1 - \lambda) \hat{n}(\hat{p}_{j+1})\|} = \\ &= \frac{\lambda_0 \hat{n}(\hat{p}_j) + (1 - \lambda_0) \hat{n}(\hat{p}_{j+1})}{\|\lambda_0 \hat{n}(\hat{p}_j) + (1 - \lambda_0) \hat{n}(\hat{p}_{j+1})\|} = \hat{n}(\lambda_0 \hat{p}_j + (1 - \lambda_0) \hat{p}_{j+1}) = \hat{n}(\mathbf{x})\end{aligned}$$

and the same holds true for the limit  $\lambda \nearrow \lambda_0$ . Therefore,  $\hat{n}$  is continuous on  $\mathbf{x}$ .  $\square$

**Lemma 3.6.** For  $j \in \{2, \dots, N_\Omega - 1\}$ ,  $\hat{n}(\hat{p}_j)$  is a normalized normal vector of the connecting line between  $\hat{p}_{j-1}$  and  $\hat{p}_{j+1}$ .

*Proof.* Because  $\hat{p}_{j-1} \neq \hat{p}_{j+1}$ , we know that for every normalized normal vector  $\mathbf{v}$  it holds true that

$$\langle \hat{p}_{j+1} - \hat{p}_{j-1}, \mathbf{v} \rangle = 0, \quad \|\mathbf{v}\| = 1.$$

We know that  $\|\hat{n}(\hat{p}_j)\| = 1$  and it holds true that

$$\begin{aligned}\langle \hat{p}_{j+1} - \hat{p}_{j-1}, \hat{n}(\hat{p}_j) \rangle &= \langle [\hat{x}_{j+1} - \hat{x}_{j-1}, \hat{z}_{j+1} - \hat{z}_{j-1}], \frac{[\hat{z}_{j+1} - \hat{z}_{j-1}, \hat{x}_{j-1} - \hat{x}_{j+1}]}{\|[\hat{z}_{j+1} - \hat{z}_{j-1}, \hat{x}_{j-1} - \hat{x}_{j+1}]\|} \rangle = \\ &= \frac{(\hat{x}_{j+1} - \hat{x}_{j-1})(\hat{z}_{j+1} - \hat{z}_{j-1}) + (\hat{z}_{j+1} - \hat{z}_{j-1})(\hat{x}_{j-1} - \hat{x}_{j+1})}{\|[\hat{z}_{j+1} - \hat{z}_{j-1}, \hat{x}_{j-1} - \hat{x}_{j+1}]\|} = 0,\end{aligned}$$

so  $\hat{n}(\hat{p}_j)$  is a normalized normal vector of the connecting line between  $\hat{p}_{j-1}$  and  $\hat{p}_{j+1}$ .  $\square$

Next, we show that  $\hat{n}(\mathbf{x})$  faces outward of  $\hat{\Omega}$  (it faces into  $\hat{\Omega}^c$ ). This can only be ensured if all connecting lines between two neighboring points of the polygonal line all have the same length. But this is not a practicable restriction on the polygonal line for our algorithm, so in practice we only ensure that the connecting lines between two neighboring points have “about” the same length.

**Lemma 3.7.** *For  $\mathbf{x} \in \partial\hat{\Omega} \setminus \{\hat{p}_1, \dots, \hat{p}_{N_\Omega}\}$ ,  $\hat{n}(\mathbf{x})$  is an approximation of the normalized normal vector, that faces outward of  $\hat{\Omega}$ , if all connecting lines between two neighboring points have (approximately) the same length. (In the following proof, we use “ $\approx$ ” instead of “ $=$ ” every time an argument holds only if all connecting lines have exactly the same length.)*

*Proof.* Considering the 2d coordinate system of the plots in `matplotlib.pyplot`, we see that the  $x$ -coordinate is the coordinate that faces to the right side and the  $z$ -coordinate is the coordinate that faces downwards.

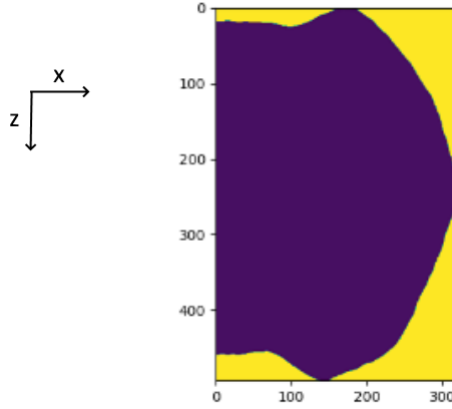


Figure 9:  $x$ - and  $z$ -coordinate for the area  $\hat{\Omega}$

Furthermore, we know that for the 2d polygonal line  $\hat{P} = (\hat{p}_1 = [\hat{x}_1, \hat{z}_1], \dots, \hat{p}_{N_\Omega} = [\hat{x}_{N_\Omega}, \hat{z}_{N_\Omega}])$  it holds that  $\hat{z}_1 < \hat{z}_{N_\Omega}$ . So when we walk along the polygonal line from node to node in this order, we see that  $\hat{\Omega}$  is located on the right hand side of the polygonal line and we see that for every segment of the polygonal line between the nodes  $\hat{p}_j$  and  $\hat{p}_{j+1}$ ,  $j \in \{1, \dots, N_\Omega - 1\}$ , the two normalized normal vectors are given by

$$n_{j,1} = \frac{[\hat{z}_{j+1} - \hat{z}_j, \hat{x}_j - \hat{x}_{j+1}]}{\|[\hat{z}_{j+1} - \hat{z}_j, \hat{x}_j - \hat{x}_{j+1}]\|}, \quad n_{j,2} = \frac{[\hat{z}_j - \hat{z}_{j+1}, \hat{x}_{j+1} - \hat{x}_j]}{\|[\hat{z}_j - \hat{z}_{j+1}, \hat{x}_{j+1} - \hat{x}_j]\|}.$$

(We can easily see that  $\langle n_{j,i}, p_{j+1} - p_j \rangle = 0$  and  $\|n_i\| = 1$ ,  $i \in \{1, 2\}$ , which are the 2 conditions on a vector to be a normalized normal vector on the connecting line from  $p_j$  to  $p_{j+1}$ ).

Now we have to check which one is the normal vector that faces outward  $\hat{\Omega}$ . Let  $[\hat{x}_{j+1} - \hat{x}_j, \hat{z}_{j+1} - \hat{z}_j] = \hat{p}_{j+1} - \hat{p}_j := [d\hat{x}_j, d\hat{z}_j]$ , then we see that the normal vector that faces outward  $\hat{\Omega}$  has to be the vector that rotates  $[d\hat{x}_j, d\hat{z}_j]$  counter clockwise. In our coordinate system this means, that a positive  $x$ -coordinate gets rotated to a negative  $z$ -coordinate, and a positive  $z$ -coordinate gets rotated to a positive  $x$ -coordinate. When we analyze

$$n_{j,1} = \frac{[d\hat{z}_j, -d\hat{x}_j]}{\|[d\hat{z}_j, -d\hat{x}_j]\|}, \quad n_{j,2} = \frac{[-d\hat{z}_j, d\hat{x}_j]}{\|[-d\hat{z}_j, d\hat{x}_j]\|},$$

we see that  $n_{j,1}$  is the normal vector that faces outward  $\hat{\Omega}$ .

It is clear that  $\hat{n}(\hat{p}_1) = [0, -1]$  and  $\hat{n}(\hat{p}_{N_\Omega}) = [0, 1]$  are facing outward  $\hat{\Omega}$ . Let  $i \in \{2, \dots, N_\Omega\}$  and let  $2\alpha := \|\hat{z}_{i+1} - \hat{z}_{i-1}, \hat{x}_{i-1} - \hat{x}_{i+1}\|$ . Then

$$\begin{aligned} \hat{n}(\hat{p}_i) &= \frac{[\hat{z}_{i+1} - \hat{z}_{i-1}, \hat{x}_{i-1} - \hat{x}_{i+1}]}{\|[\hat{z}_{i+1} - \hat{z}_{i-1}, \hat{x}_{i-1} - \hat{x}_{i+1}]\|} = \\ &= \frac{1}{2\alpha}([\hat{z}_{i+1} - \hat{z}_i, \hat{x}_i - \hat{x}_{i+1}] + [\hat{z}_i - \hat{z}_{i-1}, \hat{x}_{i-1} - \hat{x}_i]) = \frac{1}{2\alpha}([d\hat{z}_i, -d\hat{x}_i] + [d\hat{z}_{i-1}, -d\hat{x}_{i-1}]). \end{aligned}$$

Let us assume that every part of the polygonal line has about the same length, we obtain

$$\|[d\hat{z}_i, -d\hat{x}_i]\| = \|[d\hat{x}_i, d\hat{z}_i]\| \approx \|[d\hat{x}_{i-1}, d\hat{z}_{i-1}]\| = \|[d\hat{z}_{i-1}, -d\hat{x}_{i-1}]\|.$$

Let  $\beta := \|[d\hat{z}_i, -d\hat{x}_i]\|$ , then

$$\hat{n}(\hat{p}_i) = \frac{\beta}{2\alpha} \left( \frac{[d\hat{z}_i, -d\hat{x}_i]}{\beta} + \frac{[d\hat{z}_{i-1}, -d\hat{x}_{i-1}]}{\beta} \right) \approx \frac{\beta}{2\alpha} (n_{i,1} + n_{i-1,1}) = \frac{\beta}{\alpha} \frac{n_{i-1,1} + n_{i,1}}{2},$$

so we see that for every point  $\hat{p}_i$ , the approximation to the normal vector  $\hat{n}(\hat{p}_i)$  is about the normalized mean of the outward facing normal vector  $n_{i-1,1}$  of the line between  $\hat{p}_{i-1}$  and  $\hat{p}_i$  and the outward facing normal vector  $n_{i,1}$  of the line between  $\hat{p}_i$  and  $\hat{p}_{i+1}$ . (The mean of the 2 vectors gets normalized by the positive constant  $\frac{\beta}{\alpha}$  to obtain the normalized vector  $\hat{n}(\hat{p}_i)$ ).

For every connecting line segment  $l$  between  $\hat{p}_j$  and  $\hat{p}_{j+1}$ , we analyze equation (79) and we see, that the direction of  $\hat{n}(\mathbf{x})$  for  $\mathbf{x} \in l$  is the linear interpolation of the directions of  $\hat{n}(\hat{p}_j)$  (value on  $\hat{p}_j$ ) and  $\hat{n}(\hat{p}_{j+1})$  (value on  $\hat{p}_{j+1}$ ). So  $\hat{n}(\mathbf{x})$  is also an approximation of the outward facing normalized normal vector in  $\mathbf{x}$ .  $\square$

We have shown that  $\hat{n}$  is in fact smooth on  $\hat{P}$  and an approximation of the outward facing normal vector on  $\hat{P} \setminus \{\hat{p}_1, \dots, \hat{p}_{N_\Omega}\}$ . On the nodes  $\hat{p}_1, \dots, \hat{p}_{N_\Omega}$ , a normal vector on  $\hat{\Omega}$  is not defined, because the polygonal line and curve  $\hat{P}$  is not continuously differentiable in these nodes. But the normal vector is defined on almost every point on  $\hat{P}$  and this is all we need, because we only need to evaluate  $\hat{n}$  whenever we integrate over  $\hat{P}$ .

We know that the normal vector on  $\hat{P}$  is constant on every connecting line between two nodes of  $\hat{P}$ . But especially if the polygonal line  $\hat{P}$  is very “spiky” (this means, that two consecutive connecting lines  $l_1$  between  $\hat{p}_{j-1}$  and  $\hat{p}_j$ , and  $l_2$  between  $\hat{p}_j$  and  $\hat{p}_{j+1}$  point in very different directions),  $\hat{n}$  also can be very rough. So we want to define a smoother approximation of the normal vector on  $\hat{P}$ . The idea is to average  $\hat{n}$  over a broader range. Let

$$\begin{aligned}\tilde{n}(\hat{p}_1) &:= [0, -1], \\ \tilde{n}(\hat{p}_i) &:= \frac{\sum_{j \in \{-4, \dots, 4\}, i+j \in [0, N_\Omega]} (5 - |j|) \hat{n}(\hat{p}_{i+j})}{\left\| \sum_{j \in \{-4, \dots, 4\}, i+j \in [0, N_\Omega]} (5 - |j|) \hat{n}(\hat{p}_{i+j}) \right\|} \quad \text{for } i \in \{2, \dots, N_\Omega - 1\}, \\ \tilde{n}(\hat{p}_{N_\Omega}) &:= [0, 1].\end{aligned}\tag{81}$$

Every other point  $\mathbf{x} \in \mathbb{R}^2$  on  $\hat{P}$  lies on the connecting line between two nodes of  $\hat{P}$ , so there exist a  $j \in \{1, \dots, N_\Omega - 1\}$  and a  $\lambda \in [0, 1]$  so that  $\mathbf{x} = \lambda \hat{p}_j + (1 - \lambda) \hat{p}_{j+1}$ . For this  $\mathbf{x}$  we define  $\tilde{n}(\mathbf{x})$  as follows:

$$\tilde{n}(\mathbf{x}) = \tilde{n}(\lambda \hat{p}_j + (1 - \lambda) \hat{p}_{j+1}) := \frac{\lambda \tilde{n}(\hat{p}_j) + (1 - \lambda) \tilde{n}(\hat{p}_{j+1})}{\left\| \lambda \tilde{n}(\hat{p}_j) + (1 - \lambda) \tilde{n}(\hat{p}_{j+1}) \right\|}.\tag{82}$$

We defined  $\tilde{n}$  on every connecting line between two nodes on  $\hat{P}$  in a similar way as we defined  $\hat{n}$ . And we defined  $\tilde{n}(\hat{p}_i)$  as a weighted mean of the approximated normal vectors on the nodes  $\hat{n}(\hat{p}_j), j \in \{i - 4, i - 3, \dots, i + 4\}$ . Therefore,  $\tilde{n}$  is expected to be much smoother than  $\hat{n}$ .

A drawback is that one can construct a polygonal line so that  $\tilde{n}$  is not well defined. This is the case, if the denominator in the definition of  $\tilde{n}$  (Equation (81)) is zero. But usually, 10 neighboring weighted approximated normal vectors  $\hat{n}(\hat{p}_i)$  do not exactly cancel out. In this non-degenerate case,  $\tilde{n}$  is well defined. Then the following statements hold true:

- Obviously,  $\|\tilde{n}(\mathbf{x})\| = 1$  for every  $\mathbf{x} \in \hat{P}$ .
- $\tilde{n}$  is continuous on  $\hat{P}$  (similar proof as in Lemma 3.5).
- Lemma 3.6 is only needed to show that  $\hat{n}$  is a good and intuitive approximation to the normal vector of  $\hat{\Omega}$  on  $\hat{P}$ . Because  $\tilde{n}$  is a continuous approximation to  $\hat{n}$ ,  $\tilde{n}$  is also expected to be a reasonable approximation to the normal vector of  $\partial\hat{\Omega}$  on  $\hat{P}$ .
- $\tilde{n}$  is also an approximation of the outward facing normal vector of  $\partial\hat{\Omega}$  almost everywhere on  $\hat{P}$ .

The next step is to transmit  $\tilde{n}$  to a normal vector on the 3d polygonal line  $\Phi_1(\hat{P})$ . First, we introduce another two mappings  $\Psi_1$  and  $\Psi_2$ , which are very

similar to  $\Phi_1$  and  $\Phi_2$ . Let

$$\begin{aligned}\Psi_1 : \mathbb{R}_{\geq 0} \times \mathbb{R} &\rightarrow \mathbb{R}_{\geq 0} \times \{0\} \times \mathbb{R}, & \Psi_1([x, z]) &= [x, 0, -z], \\ \Psi_2 : \mathbb{R}_{\geq 0} \times \{0\} \times \mathbb{R} &\rightarrow \mathbb{R}_{\geq 0} \times \mathbb{R}, & \Psi_2([x, 0, z]) &= [x, -z].\end{aligned}\tag{83}$$

The only difference between  $\Psi_1$  and  $\Phi_1$  respectively  $\Psi_2$  and  $\Phi_2$  is, that for the last coordinate, the constant  $topz$  is not added. Analog to Lemma 3.3 we can show that  $\Psi_1$  and  $\Psi_2$  are isometries and it is obvious that  $\Psi_1 = \Psi_2^{-1}$ .

**Lemma 3.8.** *Let  $n(\mathbf{x})$  be a normal vector of  $\hat{\Omega}$  on  $\mathbf{x} = [x_1, x_2, x_3] \in \hat{P} \setminus \{\hat{p}_1, \dots, \hat{p}_{N_\Omega}\}$  (with the usual conditions on  $\hat{P}$ ). Then,  $\Psi_1(n(\mathbf{x}))$  is a normal vector of  $\partial\Omega$  on the 3d polygonal line  $\Phi_1(\hat{P})$ .*

*Proof.* Let  $\mathbf{x}$  lie on the connecting line between  $\hat{p}_j$  and  $\hat{p}_{j+1}$  for some  $j \in \{1, \dots, N_\Omega - 1\}$ . Then, for a normal vector  $n(\mathbf{x})$  it holds true that

$$\langle n(\mathbf{x}), \hat{p}_{j+1} - \hat{p}_j \rangle = 0.$$

Every tangential vector lies on the tangential surface which is spanned by arbitrary two linearly independent tangential vectors. And if  $n(\mathbf{x})$  is normal to those two tangential vectors, it follows that  $n(\mathbf{x})$  is normal to all tangential vectors. So we have to find two linearly independent tangential vectors of  $\partial\Omega$  on  $\Phi_1(\mathbf{x})$ . Because  $\Phi_1(\mathbf{x})$  lies on the polygonal line  $\Phi_1(\hat{P})$  on the connection line between the nodes  $\Phi_1(\hat{p}_j)$  and  $\Phi_1(\hat{p}_{j+1})$ , it holds true that

$$t_1 := \frac{\Phi_1(\hat{p}_{j+1}) - \Phi_1(\hat{p}_j)}{\|\Phi_1(\hat{p}_{j+1}) - \Phi_1(\hat{p}_j)\|}$$

is a tangential vector. Let  $[\tilde{x}, 0, \tilde{z}] := \Phi_1(\mathbf{x})$ . We know that  $\tilde{x} > 0$  (condition for a point on the 3d polygonal line) and we know that every point on  $\Phi_1(\hat{P})$  is rotated around the  $z$ -axis to form the boundary of  $\Omega$ . We therefore see that for the circle

$$C := \{[x, y, \tilde{z}] \mid \sqrt{x^2 + y^2} = \tilde{x}\}$$

it holds that  $C \subset \partial\Omega$  and  $[\tilde{x}, 0, \tilde{z}] \in C$ . Furthermore, it is obvious that

$$t_2 := [0, 1, 0]$$

is a tangential vector to  $C$  on  $[\tilde{x}, 0, \tilde{z}]$ , so it is also a tangential vector of  $\partial\Omega$  in  $[\tilde{x}, 0, \tilde{z}] = \Phi_1(\mathbf{x})$ . It is obvious that  $t_1$  and  $t_2$  are linearly independent and to show that  $\Psi_1(n(\mathbf{x}))$  is a normal vector of  $\partial\Omega$  on  $\Phi_1(\mathbf{x})$ , it is sufficient to show that  $\Psi_1(n(\mathbf{x}))$  is normal to  $t_1$  and  $t_2$ .

$$\begin{aligned}\langle \Psi_1(n(\mathbf{x})), t_1 \rangle &= \frac{\langle [n(\mathbf{x})_1, 0, -n(\mathbf{x})_2], [\hat{x}_{j+1}, 0, topz - \hat{z}_{j+1}] - [\hat{x}_j, 0, topz - \hat{z}_j] \rangle}{\|\Phi_1(\hat{p}_{j+1}) - \Phi_1(\hat{p}_j)\|} = \\ &= \frac{n(\mathbf{x})_1(\hat{x}_{j+1} - \hat{x}_j) + 0 + n(\mathbf{x})_2(\hat{z}_{j+1} - \hat{z}_j)}{\|\Phi_1(\hat{p}_{j+1}) - \Phi_1(\hat{p}_j)\|} = \frac{\langle n(\mathbf{x}), \hat{p}_{j+1} - \hat{p}_j \rangle}{\|\Phi_1(\hat{p}_{j+1}) - \Phi_1(\hat{p}_j)\|} = 0, \\ \langle \Psi_1(n(\mathbf{x})), t_2 \rangle &= \langle [n(\mathbf{x})_1, 0, -n(\mathbf{x})_2], [0, 1, 0] \rangle = 0.\end{aligned}$$

□



**Lemma 3.9.** *Let  $n(\mathbf{x})$  be the outward facing normalized normal vector of  $\partial\hat{\Omega}$  on  $\mathbf{x} \in \hat{P} \setminus \{\hat{p}_1, \dots, \hat{p}_{N_\Omega}\}$ . Then,  $\Psi_1(n(\mathbf{x}))$  is also the outward facing normalized normal vector of  $\partial\Omega$  on  $\Phi_1(\mathbf{x})$ . Equivalently, if  $n(\mathbf{x})$  is the inward facing normalized normal vector on  $\mathbf{x}$ ,  $\Psi_1(n(\mathbf{x}))$  is also the inward facing normalized normal vector on  $\Phi_1(\mathbf{x})$ .*

*Proof.* The proof is similar to the proof in Lemma 3.7. By Lemma 3.8 we know that  $\Psi_1(n(\mathbf{x}))$  is a normal vector on  $\Phi_1(\mathbf{x})$  and we know that  $\Psi_1$  is an isometry. Therefore,  $\Psi_1(n(\mathbf{x}))$  has the same length as  $n(\mathbf{x})$  and is also a normalized normal vector. Moreover, we know that the second normalized normal vector on  $\Phi_1(\mathbf{x})$  is  $-\Psi_1(n(\mathbf{x}))$ . We have to check, which vector is the vector that faces outwards  $\Omega$  (in  $\Phi_1(\mathbf{x})$ ).

We consider the plane with the  $y$ -coordinate being 0 in the standard 3d coordinate system. But differently to the proof in Lemma 3.7, we consider the  $z$ -coordinate to be the coordinate that faces upwards instead of downwards. For the polygonal line  $\Phi_1(\hat{P})$  with the nodes  $(p_1 = [x_1, 0, z_1], \dots, p_{N_\Omega} = [x_{N_\Omega}, 0, z_{N_\Omega}])$  it holds that  $z_{N_\Omega} < z_1$  (also different from Lemma 3.7). When we draw the polygonal line with the given coordinate system in the plane with the  $y$ -coordinate being zero we see that the polygonal line  $\Phi_1(\hat{P})$  starts again at the top and ends at the bottom (like in Lemma 3.7).

We know that  $\mathbf{x}$  lies on a connection line between two nodes  $\hat{p}_j$  and  $\hat{p}_{j+1}$  of  $\hat{P}$ . Therefore,  $\Phi_1(\mathbf{x})$  lies on the connection line between the two nodes  $\Phi_1(\hat{p}_j)$  and  $\Phi_1(\hat{p}_{j+1})$  of  $\Phi_1(\hat{P})$ . So, the outward facing normal vector is again the vector that rotates the vector

$$[\tilde{x}_j, 0, \tilde{z}_j] := \Phi_1(\hat{p}_{j+1}) - \Phi_1(\hat{p}_j) = [\hat{x}_{j+1} - \hat{x}_j, 0, -\hat{z}_{j+1} + \hat{z}_j]$$

counter clockwise. But differently to Lemma 3.7 this means that a positive  $x$ -coordinate gets rotated to a positive  $z$ -coordinate, and a positive  $z$ -coordinate gets rotated to a negative  $x$ -coordinate. So the normalized normal vector on the connecting line between  $\Phi_1(\hat{p}_j)$  and  $\Phi_1(\hat{p}_{j+1})$  is given by

$$n_j := \frac{[\hat{z}_{j+1} - \hat{z}_j, 0, \hat{x}_{j+1} - \hat{x}_j]}{\|[\hat{z}_{j+1} - \hat{z}_j, 0, \hat{x}_{j+1} - \hat{x}_j]\|}.$$

From the proof of Lemma 3.7 we know that the outward facing normalized normal vector on the connecting line between  $\hat{p}_j$  and  $\hat{p}_{j+1}$  is given by

$$n(\mathbf{x}) = n_{j,1} = \frac{[\hat{z}_{j+1} - \hat{z}_j, -\hat{x}_{j+1} + \hat{x}_j]}{\|[\hat{z}_{j+1} - \hat{z}_j, -\hat{x}_{j+1} + \hat{x}_j]\|},$$

so it is left to show that

$$\Psi_1(n(\mathbf{x})) = \Psi_1\left(\frac{[\hat{z}_{j+1} - \hat{z}_j, -\hat{x}_{j+1} + \hat{x}_j]}{\|[\hat{z}_{j+1} - \hat{z}_j, -\hat{x}_{j+1} + \hat{x}_j]\|}\right) = \frac{[\hat{z}_{j+1} - \hat{z}_j, 0, \hat{x}_{j+1} - \hat{x}_j]}{\sqrt{(\hat{z}_{j+1} - \hat{z}_j)^2 + (\hat{x}_{j+1} - \hat{x}_j)^2}} = n_j.$$

Therefore,  $\Psi_1(n_{j,1})$  is the normalized outward facing normal vector on  $\Phi_1(\mathbf{x})$ .

If  $n(\mathbf{x})$  is the normalized inward facing normal vector on  $\mathbf{x}$ , the proof is analog to show that  $\Psi_1(n(\mathbf{x}))$  is the normalized inward facing normal vector on  $\Phi_1(\mathbf{x})$ .  $\square$

**Corollary 3.9.1.** *Let  $n(\mathbf{y})$  be the outward facing normalized normal vector of  $\partial\Omega$  on  $\mathbf{y} \in P \setminus \{p_1, \dots, p_{N_\Omega}\}$ . Then,  $\Psi_2(n(\mathbf{y}))$  is also the outward facing normalized normal vector of  $\partial\Omega$  in  $\Phi_2(\mathbf{y})$ . Equivalently, if  $n(\mathbf{y})$  is the inward facing normalized normal vector on  $\mathbf{y}$ ,  $\Psi_2(n(\mathbf{y}))$  is also the inward facing normalized normal vector in  $\Phi_2(\mathbf{y})$ .*

*Proof.* Let the outward facing normalized normal vector on  $\Psi_2(n(\mathbf{y}))$  be  $n_0(\Phi_2(\mathbf{y}))$ . By Lemma 3.9 we know that  $\Psi_1(n_0(\Phi_2(\mathbf{y})))$  is the outward facing normalized normal vector on  $\Phi_1(\Phi_2(\mathbf{y})) = \mathbf{y}$ . But we know that the unique normalized normal vector with respect to  $\partial\Omega$  in  $\mathbf{y}$  is given by  $n(\mathbf{y})$ . Therefore it holds that

$$\Psi_1(n_0(\Phi_2(\mathbf{y}))) = n(\mathbf{y})$$

and because  $\Psi_1^{-1} = \Psi_2$  it immediately follows that

$$n_0(\Phi_2(\mathbf{y})) = \Psi_2(n(\mathbf{y}))$$

is the normalized outward facing normal vector on  $\Phi_2(\mathbf{y})$ .

If  $n(\mathbf{y})$  is the normalized inward facing normal vector on  $\mathbf{y}$ , the proof is analog to show that  $\Psi_2(n(\mathbf{y}))$  is the normalized inward facing normal vector on  $\Phi_2(\mathbf{y})$ .  $\square$

**Lemma 3.10.**  *$(\Psi_1 \circ \tilde{n} \circ \Phi_2)(\mathbf{y})$  is an approximation to the outward facing normalized normal vector  $n(\mathbf{y})$  with respect to  $\partial\Omega$  on  $\mathbf{y} \in P \setminus \{p_1, \dots, p_{N_\Omega}\}$ , whereby  $P$  is the polygonal line*

$$P := \Phi_1(\hat{P}) = (\Phi_1(\hat{p}_1), \dots, \Phi_1(\hat{p}_{N_\Omega})), \quad p_i := \Phi_1(\hat{p}_i), \quad i = 1, \dots, N_\Omega$$

*Proof.* For  $\mathbf{y} \in P \setminus \{p_1, \dots, p_{N_\Omega}\}$  there exists a  $\mathbf{x} \in \Phi_2(P \setminus \{p_1, \dots, p_{N_\Omega}\}) = \hat{P} \setminus \{\hat{p}_1, \dots, \hat{p}_{N_\Omega}\}$  so that  $\mathbf{y} = \Phi_1(\mathbf{x})$ , so  $\mathbf{x} = \Phi_2(\mathbf{y})$ . Moreover, let  $n(\mathbf{y})$  be the outward facing normalized normal vector on  $\mathbf{y}$ . Then, Corollary 3.9.1 yields that  $\Psi_2(n(\mathbf{y}))$  is the outward facing normalized normal vector of  $\partial\Omega$  on  $\Phi_2(\mathbf{y}) = \Phi_2(\Phi_1(\mathbf{x})) = \mathbf{x}$ .

By Lemma 3.7 we know that  $\tilde{n}(\mathbf{x}) = \tilde{n}(\Phi_2(\mathbf{y}))$  is an approximation to the outward facing normalized normal vector  $\Psi_2(n(\mathbf{y}))$  on  $\mathbf{x}$ . Because  $\Psi_1$  is an isometry, we can follow that  $\Psi_1(\tilde{n}(\Phi_2(\mathbf{y})))$  is an approximation to  $\Psi_1(\Psi_2(n(\mathbf{y}))) = n(\mathbf{y})$ , which is the outward facing normalized normal vector of  $\partial\Omega$  on  $\mathbf{y}$ .  $\square$

**Lemma 3.11.** *Let*

$$P = \Phi_1(\hat{P}) = (\Phi_1(\hat{p}_1), \dots, \Phi_1(\hat{p}_{N_\Omega})) = (\Phi_1([\hat{x}_1, \hat{z}_1]), \dots, \Phi_1([\hat{x}_{N_\Omega}, \hat{z}_{N_\Omega}]))$$

be the polygonal line that is rotated around the  $z$ -axis to form the boundary of  $\Omega$  and let  $N_\Omega \geq 3$ . Then, it holds true that

$$\partial\Omega \cap (\{0\} \times \{0\} \times \mathbb{R}) = \{[0, 0, \text{top}z - \hat{z}_1], [0, 0, \text{top}z - \hat{z}_{N_\Omega}]\}.$$

*Proof.* “ $\supset$ ” Because it holds true that  $\hat{x}_1 = 0 = \hat{x}_{N_\Omega}$  and that  $P \subset \partial\Omega$ , we can follow that

$$\begin{aligned} [0, 0, \text{top}z - \hat{z}_1] &= \Phi_1([0, \hat{z}_1]) = \Phi_1(\hat{p}_1) \in P \subset \partial\Omega, \\ [0, 0, \text{top}z - \hat{z}_{N_\Omega}] &= \Phi_1([0, \hat{z}_{N_\Omega}]) = \Phi_1(\hat{p}_{N_\Omega}) \in P \subset \partial\Omega. \end{aligned}$$

“ $\subset$ ” For  $\mathbf{z} = [0, 0, z_3] \in \partial\Omega \cap (\{0\} \times \{0\} \times \mathbb{R})$  we know that there exists a  $\mathbf{y} = [y_1, y_2, y_3] \in P$  and a rotation matrix  $R \in \mathbb{R}^{3 \times 3}$  that rotates a vector around the  $z$ -axis, so that  $\mathbf{z} = R\mathbf{y}$ . Because  $R$  rotates  $\mathbf{y}$  around the  $z$ -axis and does not change the  $z$ -coordinate of  $\mathbf{y}$ , we know that  $y_3 = z_3$ . Moreover,  $\|R\mathbf{y}\| = \|\mathbf{y}\|$ , so it holds true that

$$y_1^2 + y_2^2 + y_3^2 = \|\mathbf{y}\|^2 = \|R\mathbf{y}\|^2 = \|\mathbf{z}\|^2 = 0 + 0 + z_3^2 = y_3^2$$

and therefore  $y_1 = 0, y_2 = 0$  and

$$\mathbf{z} = [0, 0, z_3] = [0, 0, y_3] = \mathbf{y}$$

We show that for every  $\mathbf{x} = [x_1, x_2, x_3] \in P \setminus \{\Phi_1(\hat{p}_1), \Phi_1(\hat{p}_{N_\Omega})\}$  it holds true that  $x_1 > 0$ :

- Case 1: There exists an  $i \in \{2, \dots, N_\Omega - 1\}$  so that  $\mathbf{x} = \Phi_1(\hat{p}_i)$ . By the conditions on our 3d polygonal line we know that  $x_1 > 0$ .
- Case 2:  $\mathbf{x}$  is on the connecting line between  $[u_1, u_2, u_3] = \Phi_1(\hat{p}_1)$  and  $[v_1, v_2, v_3] = \Phi_1(\hat{p}_2)$ . By the conditions on our 3d polygonal line we know that  $u_1 = 0$  and that  $v_1 > 0$ . So for every point on the connecting line between  $\Phi_1(\hat{p}_1)$  and  $\Phi_1(\hat{p}_2)$  it has to hold that the first coordinate is greater than 0, and therefore,  $x_1 > 0$ . (We used that  $N_\Omega \geq 3$  and therefore,  $v_1 > 0$ .)
- Case 3:  $\mathbf{x}$  is on the connecting line between  $[u_1, u_2, u_3] = \Phi_1(\hat{p}_{N_\Omega})$  and  $[v_1, v_2, v_3] = \Phi_1(\hat{p}_{N_\Omega-1})$ . Here we know that  $u_1 = 0$  and that  $v_1 > 0$  and as in Case 2 we can conclude that  $x_1 > 0$ .
- Case 4: There exists a  $j \in \{2, \dots, N_\Omega - 2\}$  so that  $\mathbf{x}$  lies on the connecting line between  $[u_1, u_2, u_3] = \Phi_1(\hat{p}_j)$  and  $[v_1, v_2, v_3] = \Phi_1(\hat{p}_{j+1})$ . Here it holds that  $u_1 > 0$  and  $v_1 > 0$ . Therefore, for every point on the connecting line between  $\Phi_1(\hat{p}_j)$  and  $\Phi_1(\hat{p}_{j+1})$  it holds that the first coordinate is greater than 0, in particular  $x_1 > 0$ .

Therefore we can conclude that either  $\mathbf{y} = \Phi_1(\hat{p}_1)$  or  $\mathbf{y} = \Phi_1(\hat{p}_{N_\Omega})$ . Because  $\mathbf{z} = \mathbf{y}$  it holds that  $\mathbf{z} = \Phi_1(\hat{p}_1) = [0, 0, \text{top}z - \hat{z}_1]$  or  $\mathbf{z} = \Phi_1(\hat{p}_{N_\Omega}) = [0, 0, \text{top}z - \hat{z}_{N_\Omega}]$ .  $\square$

The next Lemma we are not going to proof, because the statement is obviously true but it can be tricky to proof (we would need the exact definition of tangent vectors, tangent planes and normal vectors and use them to construct a quite technical proof).

**Lemma 3.12.** *Let  $\partial\Omega$  be a rotational body so that the  $z$ -axis is the rotational axis. And let  $n(\mathbf{x})$  be a normal vector on  $\mathbf{x} \in \partial\Omega$  with respect to  $\partial\Omega$ . Then, for any rotation matrix  $R$  that rotates vectors around the  $z$ -axis it holds that  $Rn(\mathbf{x})$  is a normal vector on  $R\mathbf{x}$ .*

Now we have defined, proved or heuristically argued all components to define the continuous approximation  $\bar{n} : \partial\Omega \rightarrow \mathbb{R}^3$  of the normal vector on  $\partial\Omega$ . We define  $\bar{n}$  as follows:

Let  $\mathbf{y} \in \partial\Omega$ , then there exists a  $\mathbf{x} \in P$  and a rotation matrix  $R$  that rotates a vector around the  $z$ -axis so that  $R\mathbf{x} = \mathbf{y}$ . For this  $\mathbf{y}$  let

$$\bar{n}(\mathbf{y}) := R(\Psi_1 \circ \tilde{n} \circ \Phi_2)(\mathbf{x}) = R(\Psi_1 \circ \tilde{n} \circ \Phi_2)(R^{-1}\mathbf{y}). \quad (84)$$

With this,  $\bar{n}$  is well defined, because:

- For every point on  $\mathbf{y} \in \partial\Omega$  that does not lie directly on the  $z$ -axis there exists a unique  $\mathbf{x} \in P$  and a unique rotation matrix  $R \in \mathbb{R}^{3 \times 3}$  that rotates vectors around the  $z$ -axis so that  $\mathbf{y} = R\mathbf{x}$ . Therefore, for points that do not lie directly on the  $z$ -axis,  $\bar{n}$  is well defined
- Lemma (3.11) states that the only 2 points that lie directly on the  $z$ -axis are  $p_1 = [0, 0, \text{top}z - \hat{z}_1]$  and  $p_{N_\Omega} = [0, 0, \text{top}z - \hat{z}_{N_\Omega}]$ . Independent of the rotation matrix  $R$ , it holds that  $Rp_1 = p_1$  and  $Rp_{N_\Omega} = p_{N_\Omega}$ . At equation (81) we defined  $\tilde{n}(\hat{p}_1) = [0, -1]$  and  $\hat{n}(\hat{p}_{N_\Omega}) = [0, 1]$ , so

$$\begin{aligned} (\Psi_1 \circ \tilde{n} \circ \Phi_2)(p_1) &= (\Psi_1 \circ \tilde{n})(\Phi_2([0, 0, \text{top}z - \hat{z}_1])) = \\ &= (\Psi_1 \circ \tilde{n})([0, \hat{z}_1]) = \Psi_1(\tilde{n}(\hat{p}_1)) = \Psi_1([0, -1]) = [0, 0, 1], \\ (\Psi_1 \circ \tilde{n} \circ \Phi_2)(p_{N_\Omega}) &= (\Psi_1 \circ \tilde{n})(\Phi_2([0, 0, \text{top}z - \hat{z}_{N_\Omega}])) = \\ &= (\Psi_1 \circ \tilde{n})([0, \hat{z}_{N_\Omega}]) = \Psi_1(\tilde{n}(\hat{p}_{N_\Omega})) = \Psi_1([0, 1]) = [0, 0, -1]. \end{aligned}$$

And, independent of the rotation matrix  $R$ , it holds that  $R[0, 0, 1] = [0, 0, 1]$  and  $R[0, 0, -1] = [0, 0, -1]$ . So,  $\bar{n}$  is also well defined on  $p_1$  and  $p_{N_\Omega}$ .

$\bar{n}$  is normalized, because

- $\Phi_2(\mathbf{x}) \in \hat{P}$  for  $\mathbf{x} \in P$ ,
- $\|\tilde{n}(\hat{x})\| = 1$  for  $\hat{x} \in P$ ,
- $\|\Psi_1(\mathbf{v})\| = \|\mathbf{v}\|$  for all  $\mathbf{v} \in \mathbb{R}^2$ .
- $R$  is a unitary matrix, so  $\|R\mathbf{z}\| = \|\mathbf{z}\|$  for all  $\mathbf{z} \in \mathbb{R}^3$ .

- Therefore,

$$\|\bar{n}(\mathbf{y})\| = \|R(\Psi_1 \circ \tilde{n} \circ \Phi_2)(\mathbf{x})\| = \|(\Psi_1 \circ \tilde{n} \circ \Phi_2)(\mathbf{x})\| = \|\tilde{n}(\Phi_2(\mathbf{x}))\| = 1.$$

$\bar{n}$  is a good approximation of the outward facing normalized normal vector on  $\partial\Omega$  (wherever the normal vector is well defined), because:

- When we check the definition of  $\bar{n}$ , we see that for  $\mathbf{y} \in P$  it holds that  $R = I$  and therefore  $\bar{n}(\mathbf{y}) := (\Psi_1 \circ \tilde{n} \circ \Phi_2)(\mathbf{y})$ . Lemma (3.10) states that  $\bar{n}(\mathbf{y}) = (\Psi_1 \circ \tilde{n} \circ \Phi_2)(\mathbf{y})$  is a good approximation of the outward facing normalized normal vector on  $\mathbf{y} \in P \setminus \{p_1, \dots, p_{N_\Omega}\}$ . (This holds everywhere on  $P$ , where the normal vector is well defined.) Therefore,  $\bar{n}$  is a good approximation of the outward facing normalized normal vector on  $P$ .
- Let  $\mathbf{y} \in \partial\Omega \setminus \{p_1, p_{N_\Omega}\}$ . Then we know that there exists a unique  $\mathbf{x} \in P$  and a unique rotation matrix  $R$  that rotates a vector around the  $z$ -axis, so that  $\mathbf{y} = R\mathbf{x}$ . By Lemma (3.12) we know that the outward facing normalized normal vector on  $\mathbf{y}$  is then given by  $n(\mathbf{y}) = Rn(\mathbf{x})$ . Moreover,  $\bar{n}(\mathbf{y}) := R(\Psi_1 \circ \tilde{n} \circ \Phi_2)(\mathbf{x})$ . Because  $R$  is a unitary matrix, it holds that

$$\langle \bar{n}(\mathbf{y}), n(\mathbf{y}) \rangle = \langle R(\Psi_1 \circ \tilde{n} \circ \Phi_2)(\mathbf{x}), Rn(\mathbf{x}) \rangle = \langle (\Psi_1 \circ \tilde{n} \circ \Phi_2)(\mathbf{x}), n(\mathbf{x}) \rangle.$$

Let  $\alpha$  be the angle between the vectors  $\bar{n}(\mathbf{y})$  and  $n(\mathbf{y})$  and let  $\beta$  be the angle between the vectors  $(\Psi_1 \circ \tilde{n} \circ \Phi_2)(\mathbf{x})$  and  $n(\mathbf{x})$ ,  $\alpha, \beta \in [0, \pi]$ , then we know that it holds that

$$\cos(\alpha) = \frac{\langle \bar{n}(\mathbf{y}), n(\mathbf{y}) \rangle}{\|\bar{n}(\mathbf{y})\| \|n(\mathbf{y})\|}, \quad \cos(\beta) = \frac{\langle (\Psi_1 \circ \tilde{n} \circ \Phi_2)(\mathbf{x}), n(\mathbf{x}) \rangle}{\|(\Psi_1 \circ \tilde{n} \circ \Phi_2)(\mathbf{x})\| \|n(\mathbf{x})\|}.$$

All four vectors are normalized, so it holds that

$$\cos(\alpha) = \langle \bar{n}(\mathbf{y}), n(\mathbf{y}) \rangle = \langle (\Psi_1 \circ \tilde{n} \circ \Phi_2)(\mathbf{x}), n(\mathbf{x}) \rangle = \cos(\beta),$$

and because  $\alpha, \beta \in [0, \pi]$  it holds that  $\alpha = \beta$ . We know that  $\langle (\Psi_1 \circ \tilde{n} \circ \Phi_2)(\mathbf{x}), n(\mathbf{x}) \rangle$  is a good approximation to  $n(\mathbf{x})$ , so  $\bar{n}(\mathbf{y})$  is also a good approximation to the outward facing normalized normal vector  $n(\mathbf{y})$ .

$\bar{n}$  is also continuous on  $\partial\Omega$ , because:

- $\bar{n}$  is continuous on  $P$ , because here it holds that  $\bar{n}(\mathbf{y}) = (\Psi_1 \circ \tilde{n} \circ \Phi_2)(\mathbf{y})$ ,  $\tilde{n}$  is continuous on  $\hat{P}$  and  $\Psi_1$  is an isometry.
- For a rotation matrix  $R$  that rotates vectors around the  $z$ -axis,  $\bar{n}$  is continuous with respect to the “longitude”  $RP$ , because  $(\Psi_1 \circ \tilde{n} \circ \Phi_2)$  is continuous on  $P$ , so  $\bar{n} = R(\Psi_1 \circ \tilde{n} \circ \Phi_2)R^{-1}$  is continuous on  $RP$ .
- For a fixed point  $\mathbf{x} \in P \setminus \{p_1, p_{N_\Omega}\}$ ,  $\mathbf{y}$  we show that  $\bar{n}$  is continuous with respect to the “latitude”  $\mathbf{y}(\alpha) := R(\alpha)\mathbf{x}$ , whereby  $R(\alpha)$  is the rotation matrix, that rotates a vector around the  $z$ -axis by the angle  $\alpha \in [0, 2\pi]$ . Because  $\mathbf{x}$  does not lie on the  $z$ -axis, the latitude is a circle around the  $z$ -axis. On this latitude,  $\bar{n}$  has the values  $\bar{n}(\alpha) = R(\alpha)\bar{n}(\mathbf{y})(R(\alpha))^{-1}$ . Because  $R(\alpha)$  is continuous on this latitude,  $\bar{n}$  is also continuous there.

- By combining the last two statements ( $\bar{n}$  is continuous on every longitude and latitude), we can conclude that  $\bar{n}$  is continuous on  $\partial\Omega \setminus \{p_1, p_{N_\Omega}\}$ .
- It is left to show that  $\bar{n}$  is continuous in  $p_1$  and  $p_{N_\Omega}$ . Because the arguments for these two points are similar, we only show that  $\bar{n}$  is continuous in  $p_1$ . When we approach  $p_1$  on  $P$ , we know that  $\bar{n}$  is continuous there. So, for every  $\epsilon > 0$  there exists a  $\delta(\epsilon) > 0$  so that for  $\mathbf{x} \in P$ ,  $\|\mathbf{x} - p_1\| < \epsilon$ , it holds that
$$\|\bar{n}(\mathbf{x}) - [0, 0, 1]\| = \|\bar{n}(\mathbf{x}) - \bar{n}(p_1)\| < \delta(\epsilon).$$

Let  $\epsilon_1 > 0$  and  $\mathbf{y} \in \partial\Omega$  so that  $\|\mathbf{y} - p_1\| < \epsilon_1$ . Here, it exists a rotation matrix  $R$  that rotates vectors around the  $z$ -axis and a  $\mathbf{x} \in P$  so that  $R\mathbf{x} = \mathbf{y}$ . It is obvious that for every point  $\mathbf{z}$  on the  $z$ -axis, it holds true that  $\|\mathbf{x} - \mathbf{z}\| = \|\mathbf{y} - \mathbf{z}\|$ , so in particular,  $\|\mathbf{x} - p_1\| = \|\mathbf{y} - p_1\| < \epsilon_1$ . Moreover,

$$\begin{aligned} \|\bar{n}(\mathbf{y}) - \bar{n}(p_1)\| &= \|R\bar{n}(R^{-1}\mathbf{y}) - [0, 0, 1]\| = \\ &= \|R\bar{n}(\mathbf{x}) - R[0, 0, 1]\| = \|\bar{n}(\mathbf{x}) - [0, 0, 1]\| < \delta(\epsilon_1), \end{aligned}$$

and therefore,  $\bar{n}$  is continuous in  $p_1$ .

### 3.4.2 Interpolation of $G(\mathbf{x})$ on $\partial\Omega$

First, we show that if  $\Omega$  is a body of rotation with the  $z$ -axis being its rotational axis,  $G(\mathbf{x})$  as well as  $F(\mathbf{x}, q)$  are rotationally symmetric with respect to the  $z$ -axis.

**Lemma 3.13.** *Let  $\Omega$  be a body of rotation with the  $z$ -axis as the rotational axis. Then, for every rotation matrix  $R$  that rotates vectors around the  $z$ -axis and for every point  $\mathbf{x} = [x_1, x_2, x_3] \in \partial\Omega$ , it holds that  $R\mathbf{x} \in \partial\Omega$  and furthermore,  $G(R\mathbf{x}) = G(\mathbf{x})$ .*

*Proof.* Let  $\mathbf{x} = [x_1, x_2, x_3] \in \partial\Omega$ , so there are sequences  $(\mathbf{u}_n)_{n \in \mathbb{N}} \subset \Omega$  and  $(\mathbf{v}_n)_{n \in \mathbb{N}} \subset \Omega^c$  so that  $(\mathbf{u}_n) \rightarrow \mathbf{x}$  and  $(\mathbf{v}_n) \rightarrow \mathbf{x}$  for  $n \rightarrow \infty$ . Let  $R$  be a rotation matrix that rotates vectors around the  $z$ -axis, so  $R\Omega = \Omega$  and  $\|R\mathbf{u}\| = \|\mathbf{u}\|$  for all  $\mathbf{u} \in \mathbb{R}^3$ . Because  $R\Omega = \Omega$  it holds that  $(R\mathbf{u}_n)_{n \in \mathbb{N}} \subset \Omega$  and  $(R\mathbf{v}_n)_{n \in \mathbb{N}} \subset \Omega^c$ . Furthermore, it holds true that

$$\begin{aligned} \|R\mathbf{u}_n - R\mathbf{x}\| &= \|R(\mathbf{u}_n - \mathbf{x})\| = \|\mathbf{u}_n - \mathbf{x}\| \rightarrow 0, \\ \|R\mathbf{v}_n - R\mathbf{x}\| &= \|R(\mathbf{v}_n - \mathbf{x})\| = \|\mathbf{v}_n - \mathbf{x}\| \rightarrow 0 \end{aligned}$$

for  $n \rightarrow \infty$ . So, we found a sequence inside  $\Omega$  and a sequence outside  $\Omega$  that both converge to  $R\mathbf{x}$ . Therefore,  $R\mathbf{x} \in \partial\Omega$ .

It holds that  $\|R\mathbf{a}\| = \|\mathbf{a}\|$  for  $\mathbf{a} \in \mathbb{R}^3$ . Furthermore, let  $q \in \mathbb{R}_{\geq 0}$  and

$\mathbf{y}(\mathbf{z}) := R^{-1}\mathbf{z}$ , then it holds that  $d\mathbf{y} = d\mathbf{z}$  and therefore

$$\begin{aligned} F(R\mathbf{x}, q) &= 2c_\rho^2 \int_{\Omega} \text{sinc}(q\|\mathbf{z} - R\mathbf{x}\|) d\mathbf{z} = 2c_\rho^2 \int_{R\Omega} \text{sinc}(q\|\mathbf{z} - R\mathbf{x}\|) d\mathbf{z} = \\ &= 2c_\rho^2 \int_{\Omega} \text{sinc}(q\|R\mathbf{y} - R\mathbf{x}\|) d\mathbf{y} = 2c_\rho^2 \int_{\Omega} \text{sinc}(q\|\mathbf{y} - \mathbf{x}\|) d\mathbf{y} = F(\mathbf{x}, q). \end{aligned}$$

It follows that

$$G(R\mathbf{x}) = 2 \int_{q_{min}}^{q_{max}} F(R\mathbf{x}, q) \frac{1}{I_s(q) + \epsilon} \ln \frac{I_s(q) + \epsilon}{I_{s, meas}(q) + \epsilon} dq = G(\mathbf{x}).$$

□

**Corollary 3.13.1.** *Let  $\Omega$  be a body of rotation, whereby its boundary is a polygonal line  $P$  rotated around the  $z$ -axis like we described in section 3.2 and let  $\mathbf{y} \in \partial\Omega$ . Then, there exists a  $\mathbf{x} \in P$  and a rotation matrix  $R$  that rotates vectors around the  $z$ -axis, so that  $R\mathbf{x} = \mathbf{y}$  and  $G(\mathbf{y}) = G(\mathbf{x})$ .*

*Proof.* Because  $\partial\Omega$  is the polygonal line  $P$  rotated around the  $z$ -axis, there has to be an  $\mathbf{x} \in P$  and a rotation matrix  $R$  that rotates vectors around the  $z$ -axis, so that  $R\mathbf{x} = \mathbf{y}$ . Because of Lemma (3.13) it holds true for this particular  $\mathbf{x}$  that

$$G(\mathbf{x}) = G(R\mathbf{x}) = G(\mathbf{y}).$$

□

Because of Corollary (3.13.1) we know that it is sufficient to determine the values of  $G(\mathbf{x})$  only on  $\mathbf{x} \in P$ , when we want all values of  $G(\mathbf{y})$  for  $\mathbf{y} \in \partial\Omega$ . As we already mentioned in the last chapter, calculating  $G(\mathbf{x})$  is a bottleneck of the algorithm. Therefore, we use the interpolation  $\tilde{G}(\mathbf{x})$  of  $G(\mathbf{x})$ . We want to calculate  $G(\mathbf{x})$  only on some specific points on  $P$  (here we set  $\tilde{G}(\mathbf{x}) = G(\mathbf{x})$ ). For the other nodes of the polygonal line we determine  $G(\mathbf{x})$  by cubical interpolation. And for all points of the polygonal line that are not nodes, we use linear interpolation between the value of  $\tilde{G}(\mathbf{x})$  of the two neighboring nodes.

For this matter, we introduce the parameter *indicesToCheck* which is an array of indices, whereas  $G(p_{i+1})$  is calculated if and only if  $i \in \text{indicesToCheck}$ . For all other polygon nodes  $p_j$ ,  $G(p_j)$  is determined by cubical interpolation. (E.g.  $N_\Omega = 100$  and *indicesToCheck* =  $[0, 20, 40, 60, 80, 99]$ , then  $G(p_1)$ ,  $G(p_{21})$ ,  $G(p_{41})$ ,  $G(p_{61})$ ,  $G(p_{81})$  and  $G(p_{100})$  are calculated)

After we calculated  $G(p_{i+1})$  for all  $i \in \text{indicesToCheck}$ , we set  $\tilde{G}(p_{i+1}) := G(p_{i+1})$  for all  $i \in \text{indicesToCheck}$ , then, for all  $j \in \{0, \dots, N_\Omega - 1\} \setminus \text{indicesToCheck}$ , we

- determine

$$\begin{aligned} i_1 &= \max\{i \in \text{indicesToCheck} \mid i < j\}, \\ i_2 &= \min\{i \in \text{indicesToCheck} \mid i > j\}, \end{aligned}$$

- set  $G_1 := G(p_{i_1+1})$  and  $G_2 := G(p_{i_2+1})$ ,
- set  $l_1 := \|p_{i_1+1} - p_{j+1}\|$  and  $l_2 := \|p_{i_2+1} - p_{j+1}\|$ ,
- set  $\alpha = \frac{l_1}{l_1+l_2}$ . It holds true that
  - $\alpha \in [0, 1]$
  - If  $\alpha < 0.5$ ,  $p_{j+1}$  is nearer to  $p_{i_1+1}$  then to  $p_{i_2+1}$ . (The opposite is the case if  $\alpha > 0.5$ .)
  - $\alpha$  is the main parameter for the cubical interpolation.
- analyze the third order polynomial  $f : [0, 1] \rightarrow \mathbb{R}$ ,  $f(x) = ax^3 + bx^2 + cx + d$  with the boundary conditions  $f'(0) = f'(1) = 0$ ,  $f(0) = 0$ ,  $f(1) = 1$  and see that  $f(x) = -2x^3 + 3x^2$  is the only polynomial that fulfills the conditions and therefore we set

$$\lambda := f(\alpha) = -2\alpha^3 + 3\alpha^2.$$

- finally set  $\tilde{G}(p_{j+1}) := (1 - \lambda)G_1 + \lambda G_2$ .

Now we have determined  $\tilde{G}$  on every node of  $P$ . On the connecting line between two nodes,  $\tilde{G}(\mathbf{x})$  is linearly interpolated. But actually, we never have to evaluate the value of  $\tilde{G}(\mathbf{x})$  for points that are not nodes of  $P$ .

For every point  $\mathbf{y} \in \partial\Omega$ , we know that there exists a  $\mathbf{x} \in P$  and a rotation matrix  $R$  that rotates vectors around the  $z$ -axis, so that  $R\mathbf{x} = \mathbf{y}$ . We know that  $G(\mathbf{x}) = G(\mathbf{y})$  (Corollary 3.13.1). Therefore we set

$$\tilde{G}(\mathbf{y}) := \tilde{G}(R^{-1}\mathbf{y}) = \tilde{G}(\mathbf{x}).$$

### 3.4.3 Final setting of $V(\mathbf{x})$

In section 3.4.1 we defined  $\bar{n} : \partial\Omega \rightarrow \mathbb{R}^3$ , which is a normalized and continuous approximation of the outward facing normalized normal vector on  $\partial\Omega$ . In section 3.4.2 we defined the approximation  $\tilde{G} : \partial\Omega \rightarrow \mathbb{R}$  to  $G(\mathbf{x})$ , because we wanted to minimize the number of evaluations of  $G(\mathbf{x})$ . Therefore, for every  $\mathbf{x} \in \partial\Omega$  we set our change direction field  $V(\mathbf{x})$  to

$$\begin{aligned} V(\mathbf{x}) &= l(\mathbf{x})\mathbf{d}(\mathbf{x}), \\ \mathbf{d}(\mathbf{x}) &= -\text{sign}(\tilde{G}(\mathbf{x}))\bar{n}(\mathbf{x}), \\ l(\mathbf{x}) &= |\tilde{G}(\mathbf{x})|. \end{aligned} \tag{85}$$

## 3.5 Deformation of $\Omega$ in direction $V(\mathbf{x})$

In one optimization step, we deform  $\Omega$  in direction  $V(\mathbf{x})$ . But we still have to choose the step length. These are the advantages and disadvantages to choose a small or a large step length:



- Advantage for a large step length: Possibility to approach more quickly to a domain  $\Omega$  where the error  $J(\Omega)$  is as small as possible
- Disadvantage for a large step length: The possibility to over-shoot is high, so that  $J(\Omega)$  is increasing instead of decreasing. Also irregular geometries are more likely to occur.

Therefore we try different step sizes. Because obtaining  $V(\mathbf{x})$  (mainly the evaluation of  $G(\mathbf{x})$ ) takes a lot of time, trying out several different step sizes is not a huge performance drop for our algorithm.

To deform  $\Omega$ , all we have to do is to shift nodes of the 2d polygonal line  $\hat{P}$  we introduced in section 3.1.1, because the 3d polygonal line  $P$  and the rotational body  $\Omega$  are uniquely defined by the 2d polygonal line  $\hat{P}$ . We introduce the array  $stepLengths := [s_1, \dots, s_{N_s}]$ . One entry  $\lambda_i$  determines the maximum change in “pixel” dimension in direction  $V(\mathbf{x})$  (more precisely  $\Psi_2(V(\mathbf{x}))$ , because it is a change in 2d). The next figure shows an example of the directions  $\tilde{G}(\mathbf{x})$  on the actual 2d polygonal line (left picture). To obtain the right picture, the nodes of the polygonal line of the picture in the middle were shifted by 30 pixel in the direction  $\tilde{G}(\mathbf{x})$ .

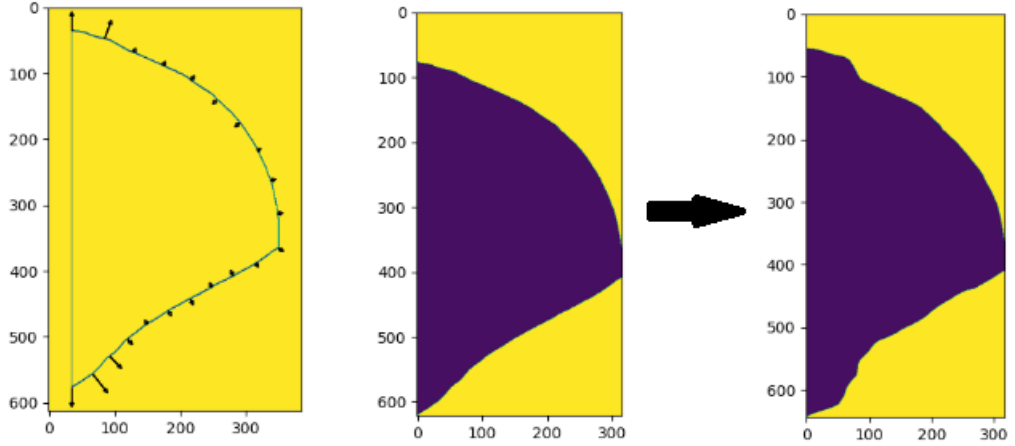


Figure 10: Shift actual picture (middle) max. 30px in given directions (left) to obtain the next picture (right)

For one step size  $s_i$ , the maximum change over all nodes of  $\hat{P}$  has to be  $s_i$ . We described the actual positions of the nodes of  $\hat{P}$  in section 3.1.1 as  $\tilde{p}_j = [\tilde{x}_j, \tilde{z}_j]$ ,  $j = 1, \dots, N_\Omega$ . We obtain the next polygon by shifting these nodes. Then we get a new body of rotation  $\Omega_{s_i}$ , where we can calculate  $I_s(q, \Omega_{s_i})$  and the error  $J(\Omega_{s_i})$ :

- First we set  $l_{max} := \max_{k \in \{1, \dots, N_\Omega\}} l(p_k)$

- For every  $j \in \{1, \dots, N_\Omega\}$ , set
  - $\hat{p}_j = l_p \tilde{p}_j$  with  $l_p = \text{pixel\_length}$
  - $p_j = \Phi_1(\hat{p}_j)$
  - $V_j = (1/l_{max})V(p_j)$  (for all vectors  $V(p_j)$  it holds that  $\|V(p_j)\| \leq 1$  and there exists a  $j_0 \in \{1, \dots, N_\Omega\}$  so that  $\|V(p_{j_0})\| = 1$ )
  - $d_j = s_i \Psi_2(V_j)$ . These are the changes in pixel. Because  $\Psi_2$  is an isometry, the maximum change in pixel is  $\max_{k=1, \dots, N_\Omega} d_k = s_i$ .
  - $\tilde{p}_j = \text{np.round}(\tilde{p}_j + d_j)$ . We round the change, because we want to work again with integers.
- Construct the image of the area that gets rotated to form  $\Omega_{s_i}$ .
- Calculate  $I_s(q, \Omega_{s_i})$ .
- Calculate the error  $J(\Omega_{s_i})$ .

A typical step size array is e.g.

$$\begin{aligned} \text{stepLengths} = [ & -60, -40, -30, -25, -22, -20, -19, -18, -17, -16, -15, -14, -13, \\ & -12, -11, -10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 1, 2, 3, 4, 5, 6, \\ & 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 22, 25, 30, 40, 60]. \end{aligned}$$

When we apply the strategy described above for every step size  $s_i \in \text{stepLengths}$  and obtain the errors  $E_i := J(\Omega_{s_i})$ , we can plot the step size vs. the error like in the following figure:

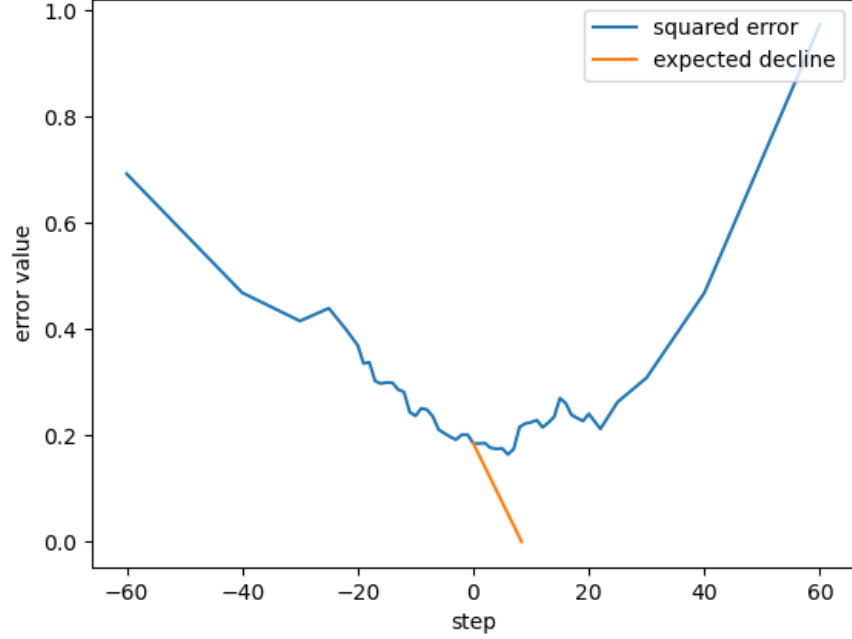


Figure 11: Comparison of the errors  $J(\Omega_{s_i})$  for different step sizes  $s_i$

Finally, we choose the step size  $s_i$  where the calculated error  $J(\Omega_{s_i})$  was the lowest and set

$$\begin{aligned} s_{min} &:= \operatorname{argmin}\{J(\Omega_{s_i}) \mid s_i \in \text{stepLengths}\}, \\ \Omega &:= \Omega_{s_{min}}. \end{aligned} \tag{86}$$

### 3.5.1 Approximation of the descent of $J(\Omega)$

In figure 11 we see an additional red line, which gives us information about the expected descent of  $J(\Omega)$  in direction  $V(\mathbf{x})$ . In more detail, the red line  $r(s)$  is an approximation to the tangent of the blue line and should approximately have the values

$$r(s) = J(\Omega) + s \frac{1}{l_p} dJ(\Omega, \frac{1}{l_{max}} V), \quad [s] = \text{“pixel”}.$$

Remark: For the vector field  $V_0 := \frac{1}{l_{max}} V$ ,  $dJ(\Omega, V_0)$  gives us the gradient when we move 1 step (unitless) in direction  $V_0$ . But we only want to move 1 pixel in direction  $V_0$ , therefore we have to divide by  $l_p = \text{pixel\_len}$ .

So we want to calculate an approximation to  $dJ(\Omega, \frac{1}{l_{max}} V)$ , where  $V(\mathbf{x})$  is chosen like in section 3.4.3. Beforehand, we have to split  $\partial\Omega$  into the  $(N_\Omega - 1)$  parts  $\partial\Omega_1, \dots, \partial\Omega_{N_\Omega-1}$  as follows:

**Definition 3.1** ( $\partial\Omega_i$ ). Let  $i \in \{1, \dots, N_\Omega - 1\}$  and let  $c_i$  be the connecting line between the nodes  $p_i$  and  $p_{i+1}$  of the polygonal line  $P$ . Then we define  $\partial\Omega_i$  as the line  $c_i$  that is rotated around the  $z$ -axis, in other words:

$$\partial\Omega_i = \{\mathbf{y} \in \mathbb{R}^3 \mid \text{there exist an } \mathbf{x} \in c_i \text{ and a rotation matrix } R \\ \text{that rotates vectors around the } z\text{-axis, so that } R\mathbf{x} = \mathbf{y}\}.$$

These are the approximations we use to calculate the approximation to the descent  $dJ(\Omega, \frac{1}{l_{max}}V)$ :

1.  $\tilde{G}(\mathbf{x}) \approx G(\mathbf{x})$
2.  $\bar{n}(\mathbf{x}) \approx n(\mathbf{x})$  and because both vectors are normalized, it holds true that  $\langle \bar{n}(\mathbf{x}), n(\mathbf{x}) \rangle \approx 1$

Remark: Probably a better approximation would be e.g.  $\langle \bar{n}(\mathbf{x}), n(\mathbf{x}) \rangle \approx 0.9$ , because  $\langle \bar{n}(\mathbf{x}), n(\mathbf{x}) \rangle$  is always smaller than 1, and only 1 if  $\bar{n}(\mathbf{x}) = n(\mathbf{x})$ . Keep this reasoning in mind, because it should be a main factor why the approximation of the descent is a little bit larger than the true descent.

3. For every boundary part  $\partial\Omega_i$  we use the trapezoidal rule

$$\int_{\partial\Omega_i} \tilde{G}^2(x) da(x) \approx \int_{\partial\Omega_i} \frac{\tilde{G}^2(p_i) + \tilde{G}^2(p_{i+1})}{2} da(x) = \frac{\tilde{G}^2(p_i) + \tilde{G}^2(p_{i+1})}{2} \int_{\partial\Omega_i} da(x)$$

Remarks:

- We defined  $\tilde{G}(\mathbf{x})$  so that it is invariant with respect to rotations around the  $z$ -axis. To know every value of  $\tilde{G}(\mathbf{x})$  on  $\partial\Omega_i$ , we only have to evaluate  $\tilde{G}$  on the points on the connecting line between  $p_i$  and  $p_{i+1}$ .
- When we consider the integral  $\int_a^b l^2(x) dx$  for  $a < b$  and a straight line  $l(x) = kx + d$  with  $k, d \in \mathbb{R}$ , we can compare

$$\begin{aligned} \int_a^b l^2(x) dx &= \int_a^b ((kx)^2 + 2(kx)d + d^2) dx = \left[ \frac{k^2 x^3}{3} + kdx^2 + d^2 x \right]_a^b = \\ &= k^2 \left( \frac{b^3}{3} - \frac{a^3}{3} \right) + kd(b^2 - a^2) + d^2(b - a) \end{aligned}$$

with

$$\begin{aligned}
\int_a^b \frac{l^2(a) + l^2(b)}{2} dx &= (b-a) \frac{1}{2} [(ka+d)^2 + (kb+d)^2] = \\
&= \frac{(b-a)}{2} [k^2 a^2 + 2kda + d^2 + k^2 b^2 + 2kdb + d^2] = \\
&= \frac{1}{2} [k^2(-a^3 + a^2b - ab^2 + b^3) + 2kd(-a^2 + ab - ab + b^2) + 2d^2(-a + b)] = \\
&= \frac{1}{2} k^2(-a^3 + a^2b - ab^2 + b^3) + kd(b^2 - a^2) + d^2(b-a)
\end{aligned}$$

and we see that

$$\begin{aligned}
\int_a^b \frac{l^2(a) + l^2(b)}{2} dx - \int_a^b l^2(x) dx &= \frac{k^2}{6} (-3a^3 + 3a^2b - 3ab^2 + 3b^3 - 2b^3 + 2a^3) = \\
&= \frac{k^2}{6} (b^3 - 3ab^2 + 3a^2b - a^3) = \frac{k^2}{6} (b-a)^3 > 0
\end{aligned}$$

- We can infer that it is very likely that for a function that is not too different to a linear function  $f : [a, b] \rightarrow \mathbb{R}$  it also holds true that

$$\int_a^b \frac{f^2(a) + f^2(b)}{2} dx > \int_a^b f^2(x) dx$$

and therefore it is also likely that

$$\int_{\partial\Omega_i} \frac{\tilde{G}^2(p_i) + \tilde{G}^2(p_{i+1})}{2} da(x) > \int_{\partial\Omega_i} \tilde{G}^2(x) da(x) \quad (87)$$

holds true. (This statement is very heuristic and does not hold in the general case.)

- The difference between the terms  $\int_a^b \frac{l^2(a) + l^2(b)}{2} dx$  and  $\int_a^b l^2(x) dx$  is  $\frac{k^2}{6} (b-a)^3$ , so the difference is small if
  - $a$  and  $b$  are close respectively  $p_i$  and  $p_{i+1}$  are close,
  - $k$  is small respectively  $f(a)$  and  $f(b)$  are close respectively  $\tilde{G}(p_i)$  and  $\tilde{G}(p_{i+1})$  are close.

In the book *Numerische Mathematik 1: Eine algorithmisch orientierte Einführung (De Gruyter Studium, Band 1)* of Deuffhard and Hohmann [12], the authors show on page 303 that this difference depends on  $|\tilde{G}^{2''}|$ .

**Theorem 3.14.** *Considering the three approximations above, an approximation of the expected descent of  $J(\Omega)$  in direction  $\frac{1}{l_{max}}V$  is given by*

$$dJ(\Omega, \frac{1}{l_{max}}V) \approx -\frac{\pi}{2} \sum_{i=1}^{N_{\Omega}-1} (G^2(p_{i+1}) + G^2(p_i)) \|p_{i+1} - p_i\| (\hat{x}_{i+1} + \hat{x}_i),$$

where  $\hat{x}_i$  is the first coordinate of the  $i^{th}$  node of the 2-dimensional polygonal line  $\hat{P}$ .

*Proof.* By equation (85) we know that  $V$  is given by

$$V(\mathbf{x}) = l(\mathbf{x})\mathbf{d}(\mathbf{x}) = |\tilde{G}(\mathbf{x})|[-\text{sign}(\tilde{G}(\mathbf{x}))\bar{n}(\mathbf{x})] = -\tilde{G}(\mathbf{x})\bar{n}(\mathbf{x}).$$

Therefore,

$$\begin{aligned} dJ(\Omega, \frac{1}{l_{max}}V) &= \int_{\partial\Omega} G(\mathbf{x}) \langle V(\mathbf{x}), n(\mathbf{x}) \rangle da = \int_{\partial\Omega} G(\mathbf{x}) \langle -\tilde{G}(\mathbf{x})\bar{n}(\mathbf{x}), n(\mathbf{x}) \rangle da \approx \\ &\approx - \int_{\partial\Omega} \tilde{G}^2(\mathbf{x}) \langle \bar{n}(\mathbf{x}), n(\mathbf{x}) \rangle da \approx - \int_{\partial\Omega} \tilde{G}^2(\mathbf{x}) \langle \bar{n}(\mathbf{x}), \bar{n}(\mathbf{x}) \rangle da = - \int_{\partial\Omega} \tilde{G}^2(\mathbf{x}) da. \end{aligned}$$

It is obvious that  $(\bigcup_{i=1}^{N_{\Omega}-1} \partial\Omega_i) \subset \partial\Omega$ .

Moreover, for  $\partial\Omega_i$  and  $\partial\Omega_j$ ,  $i < j$ , it holds true that

$$\partial\Omega_i \cap \partial\Omega_j = \begin{cases} \emptyset & \text{if } i+1 \neq j, \\ p_j \text{ rotated around the } z\text{-axis} & \text{else.} \end{cases}$$

So the only points  $\mathbf{x} \in \partial\Omega$ , that are inside at least 2 different parts  $\partial\Omega_k$  and  $\partial\Omega_l$  for  $k \neq l$  are inside the set

$$B = \{Rp_i \mid i \in \{1, \dots, N_{\Omega}\}, R \text{ is a rot. matrix that rotates vectors around the } z\text{-axis}\}.$$

The set  $B$  consists of points that are rotated, so it consists of several closed paths. Therefore, its 2-dimensional measure is 0 and we can split the integral  $\int_{\partial\Omega} \tilde{G}^2(\mathbf{x}) da$  up as follows:

$$dJ(\Omega, \frac{1}{l_{max}}V) \approx - \int_{\partial\Omega} \tilde{G}^2(\mathbf{x}) da = - \sum_{i=1}^{N_{\Omega}-1} \int_{\partial\Omega_i} \tilde{G}^2(\mathbf{x}) da.$$

We use the consideration we made at (87) to receive

$$\begin{aligned} dJ(\Omega, \frac{1}{l_{max}}V) &\approx - \sum_{i=1}^{N_{\Omega}-1} \int_{\partial\Omega_i} \tilde{G}^2(\mathbf{x}) da \approx - \sum_{i=1}^{N_{\Omega}-1} \int_{\partial\Omega_i} \frac{\tilde{G}^2(p_{i+1}) + \tilde{G}^2(p_i)}{2} da = \\ &= - \sum_{i=1}^{N_{\Omega}-1} \frac{\tilde{G}^2(p_{i+1}) + \tilde{G}^2(p_i)}{2} \int_{\partial\Omega_i} da \end{aligned}$$

Now we analyze  $\int_{\partial\Omega_i} da$  further for a fixed  $i \in \{1, \dots, N_\Omega - 1\}$ .

The first Pappus' centroid Theorem states that the surface area  $A$  of a surface of revolution generated by rotating a plane curve  $C$  about an axis external to  $C$  and on the same plane is equal to the product of the arc length  $s$  of  $C$  and the distance  $d$  traveled by the geometric centroid of  $C$  [13].

We can apply this theorem to  $\partial\Omega_i$ , because  $\partial\Omega_i$  is the connecting line  $c_i$  between  $p_i$  and  $p_{i+1}$  rotated around the  $z$ -axis. The length of the  $c_i$  is  $\|p_{i+1} - p_i\|$  and the geometric centroid of  $c_i$  is its midpoint  $m_i := \frac{1}{2}(p_{i+1} + p_i)$ , which has the distance  $\frac{1}{2}(x_{i+1} + x_i) = \frac{1}{2}(\hat{x}_{i+1} + \hat{x}_i)$  to the  $z$ -axis. So the distance the centroid travels around the  $z$ -axis is

$$d = 2\left(\frac{1}{2}(\hat{x}_{i+1} + \hat{x}_i)\right)\pi = \pi(\hat{x}_{i+1} + \hat{x}_i)$$

and the area  $\int_{\partial\Omega_i} da$  equals

$$\int_{\partial\Omega_i} da = \|p_{i+1} - p_i\|d = \pi\|p_{i+1} - p_i\|(\hat{x}_{i+1} + \hat{x}_i).$$

Alltogether, we get

$$\begin{aligned} dJ(\Omega, \frac{1}{l_{max}}V) &\approx - \sum_{i=1}^{N_\Omega-1} \frac{\tilde{G}^2(p_{i+1}) + \tilde{G}^2(p_i)}{2} \int_{\partial\Omega_i} da = \\ &= - \sum_{i=1}^{N_\Omega-1} \frac{\tilde{G}^2(p_{i+1}) + \tilde{G}^2(p_i)}{2} \pi\|p_{i+1} - p_i\|(\hat{x}_{i+1} + \hat{x}_i). \end{aligned}$$

□

Once again it is important to note that the first and the third approximations at the beginning of section 3.5.1 overestimate the actual value of  $dJ(\Omega, \frac{1}{l_{max}}V)$ . So, by multiplying the estimation recieved in Theorem 3.14 with a positive correction factor  $\alpha_{cor} \in (0, 1)$  (e.g.  $\alpha_{cor} := 0.7$ ), we probably get a more exact approximation to  $dJ(\Omega, \frac{1}{l_{max}}V)$ . Therefore,

$$dJ(\Omega, \frac{1}{l_{max}}V) \approx -\alpha_{cor} \frac{\pi}{2} \sum_{i=1}^{N_\Omega-1} (G^2(p_{i+1}) + G^2(p_i))\|p_{i+1} - p_i\|(\hat{x}_{i+1} + \hat{x}_i). \quad (88)$$

### 3.6 Importance of scaling $\Omega$ to a specific volume

The volume of  $\Omega$  has a large impact on the  $I_s(q)$  respectively  $\log(I_s(q))$  graph, and therefore also on the error  $J(\Omega)$ . When testing the algorithm it could be seen that small changes of the volume (for the same shaped  $\Omega$ , only scaled) made a bigger impact on  $I_s(q)$  than small non-volume-changing perturbations on the boundary.

Therefore, these problems occurred:

- At the first few steps of the algorithm (accepted changes of the boundary), the algorithm was more inclined to find an  $\Omega$  for which the volume was optimal, and made more or less “random” changes on the boundary to achieve that.
- When the algorithm found some  $\Omega$  with optimal volume, changes in a direction where the  $\log(I_s(q))$  curve should (theoretically) approach the measured  $\log(I_{s,meas}(q))$  curve were declined because the volume changed too much - and therefore the  $J(\Omega)$  increased instead of decreased.
- Larger (shape) changes on  $\Omega$  most likely changed the volume a lot, so they got declined more often.
- To represent  $\Omega$  we check grid points on a 3d mesh whether they are inside  $\Omega$ . When we make a theoretically non-volume-changing movement on the boundary, it can happen that the number of grid points inside  $\Omega$  changes nevertheless - and therefore the calculated(!) volume changes. Therefore, non-volume-changing perturbations where the  $\log(I_s(q))$  curve should theoretically approach the measured  $\log(I_{s,meas}(q))$  curve, also were declined sometimes.

Fortunately, there is an easy way to check the volume of  $\Omega$  by only knowing the  $I_s(q)$  (respectively  $\log(I_s(q))$ ) curve.

**Lemma 3.15.** *For the volume of  $\Omega$ , it holds true that*

$$\text{vol}(\Omega) = \frac{1}{c_\rho} \sqrt{I_s(0)}.$$

*Proof.* We see that

$$\begin{aligned} \frac{1}{c_\rho^2} I_s(0) &= \frac{1}{c_\rho^2} \int_{\mathbb{R}^3} \gamma(\mathbf{z}) \text{sinc}(0\|\mathbf{z}\|) d\mathbf{z} = \frac{1}{c_\rho^2} \int_{\mathbb{R}^3} \gamma(\mathbf{z}) d\mathbf{z} = \\ &= \frac{1}{c_\rho^2} \int_{\mathbb{R}^3} \left( c_\rho^2 \int_{\Omega \cap (\Omega + \mathbf{z})} d\mathbf{x} \right) d\mathbf{z} = \int_{\mathbb{R}^3} \int_{\Omega} \mathbb{1}_\Omega(\mathbf{x} - \mathbf{z}) d\mathbf{x} d\mathbf{z} = \int_{\Omega} \int_{\mathbb{R}^3} \mathbb{1}_\Omega(\mathbf{x} - \mathbf{z}) d\mathbf{z} d\mathbf{x}. \end{aligned}$$

Let  $\mathbf{y} := \mathbf{x} - \mathbf{z}$ . Then it holds true that  $d\mathbf{y} = d\mathbf{z}$  and

$$\begin{aligned} \frac{1}{c_\rho^2} I_s(0) &= \int_{\Omega} \int_{\mathbb{R}^3} \mathbb{1}_\Omega(\mathbf{x} - \mathbf{z}) d\mathbf{z} d\mathbf{x} = \int_{\Omega} \int_{\mathbb{R}^3} \mathbb{1}_\Omega(\mathbf{y}) d\mathbf{y} d\mathbf{x} = \\ &= \int_{\Omega} \text{vol}(\Omega) d\mathbf{x} = \text{vol}(\Omega) \int_{\Omega} d\mathbf{x} = (\text{vol}(\Omega))^2. \end{aligned}$$

Therefore,

$$\text{vol}(\Omega) = \frac{1}{c_\rho} \sqrt{I_s(0)}.$$

□



When  $I_{s,meas}(0)$  is given beforehand, we therefore know the optimal volume for the resulting  $\Omega$ . When  $I_{s,meas}(0)$  is not given - this can only happen if  $I_{s,meas}(q)$  is only given for  $q$  in the range  $[q_{min}, q_{max}]$  with  $q_{min} > 0$  -  $I_{s,meas}(q)$  has to be extrapolated to get an approximate value of  $I_{s,meas}(0)$ . Therefore, the optimal volume for the resulting  $\Omega$  should be

$$\text{vol}(\Omega_{res}) = \frac{1}{c_\rho} \sqrt{I_s(0)}. \quad (89)$$

So, whenever we want to determine the values  $I_s(q)$  of some domain  $\Omega$ , we scale  $\Omega$  to the volume  $\text{vol}(\Omega_{res})$  as follows:

1. We start with an initial 2d polygonal line  $\tilde{P}$  like we described in section 3.1.1, and calculate the volume of  $\Omega = \Omega(\tilde{P})$  by counting all grid points that lie inside  $\Omega$ .
2. We set  $\tilde{P}_{act} = \tilde{P}$ ,  $\tilde{P}_{last} := \tilde{P}$ .
3. If  $\text{vol}(\Omega) < \text{vol}(\Omega_{res})$ , set  $\tilde{P}_{last} := \tilde{P}_{act}$  and  $\tilde{P}_{act} = \text{round}(1.1\tilde{P}_{act})$  (so make  $\tilde{P}_{act}$  10% larger and round all coordinates of the polygonal line nodes to the nearest integer). Then calculate  $\text{vol}(\Omega)$  again and repeat until  $\text{vol}(\Omega) > \text{vol}(\Omega_{res})$ .
4. Else if  $\text{vol}(\Omega) > \text{vol}(\Omega_{res})$ , set  $\tilde{P}_{last} := \tilde{P}_{act}$  and  $\tilde{P}_{act} = \text{round}(0.9\tilde{P}_{act})$  (so make  $\tilde{P}_{act}$  10% smaller and round all coordinates of the polygonal line nodes to the nearest integer). Then calculate  $\text{vol}(\Omega)$  again and repeat until  $\text{vol}(\Omega) < \text{vol}(\Omega_{res})$ .
5. Now we retrieved 2 polygonal lines  $\tilde{P}_{last}$  and  $\tilde{P}_{act}$ . The volume of one of the 2 polygonal lines is larger and one volume is smaller than  $\text{vol}(\Omega_{res})$ . Their relative size difference is about 10%. We set  $\tilde{P}_{low}$  to the polygonal line with the smaller volume and  $\tilde{P}_{high}$  to the polygonal line with the higher volume.
6. We then apply a type of bisection method with the goal to retrieve a polygonal line so that  $I_s(0) \approx \text{vol}(\Omega_{res})$ . In every step we set  $\tilde{P}_{act} := \text{round}(\frac{1}{2}(\tilde{P}_{low} + \tilde{P}_{high}))$ , calculate the values of  $I_s(q)$  and set
  - $\tilde{P}_{low} := \tilde{P}_{act}$  if  $I_s(0) < \text{vol}(\Omega_{res})$ ,
  - $\tilde{P}_{high} := \tilde{P}_{act}$  else.

This bisection step gets repeated several times (10 times should be enough and is also our choice).

After we apply the algorithm above, we retrieve a polygon  $\tilde{P}_{scaled}$  whereby  $\text{vol}(\Omega(\tilde{P}_{scaled})) \approx \text{vol}(\Omega_{res})$ . We then can continue to calculate  $I_s(q)$ , the error  $J(\Omega)$  and so on.

The last thing we have to discuss is the performance of this algorithm in comparison to the performance of the calculation of  $I_s(q)$ . Therefore, a very exact parameter set was chosen so that the computation time for determining  $I_s(q)$  one time was about 7 minutes. One calculation of the volume (additionally, finding the boundary of  $\Omega$  every time) took about 2 seconds. The algorithm above never needed more than 15 volume calculations, so the time to find the polygonal line  $\tilde{P}_{scaled}$  never took longer than 30 seconds. Because of the advantages of scaling to the desired volume, the scaling should always be executed before calculating  $I_s(q)$ .

### 3.7 Parameter settings and their influence on $\log(I_s(q))$

Before we show one run to find  $\Omega_{opt}$ , we have to discuss to which values the most important parameters *lengthXY*, *nrOfGaussPoints* and *nrOfSpherePoints*, . . . were set and how it affected the accuracy and the computation time. We always set  $lengthZ = lengthXY$ .

We first checked the influences of these three parameters on the  $\log(I_s(q))$  curve for an ellipsoid with radii

- $\hat{r}_1 := 1.0$ ,
- $\hat{r}_2 := 1.0$ ,
- $\hat{r}_3 := 2.0$ .

First, we generated the  $\log(I_s(q))$  chart for the very exact parameters:

- $lengthXY = 0.05$
- $nrOfGaussPoints = 50$
- $nrOfSpherePoints = 770$

While *nrOfGaussPoints* and *nrOfSpherePoints* are (more or less) independent of the size respectively volume of  $\Omega$ , the mesh size *lengthXY* has to be chosen depending on the volume of  $\Omega$  to maintain a good balance between accuracy and performance. For our choice, the ellipsoid has an exact volume of

$$\text{vol(ellipsoid)} = \frac{4}{3}\pi \hat{r}_1 \hat{r}_2 \hat{r}_3 = \frac{8}{3}\pi,$$

the volume of one grid cell is  $0.05^3$ , so there are approximately

$$\frac{\text{vol(ellipsoid)}}{0.05^3} \approx 67020$$

grid points inside  $\Omega$ .

### 3.7.1 Influence of $nrOfGaussPoints$ on $\log(I_s(q))$

In figure 12, the influence of the number of interpolation points  $nrOfGaussPoints$  on the  $\log(I_s(q))$  curves for the given ellipsoid is shown.

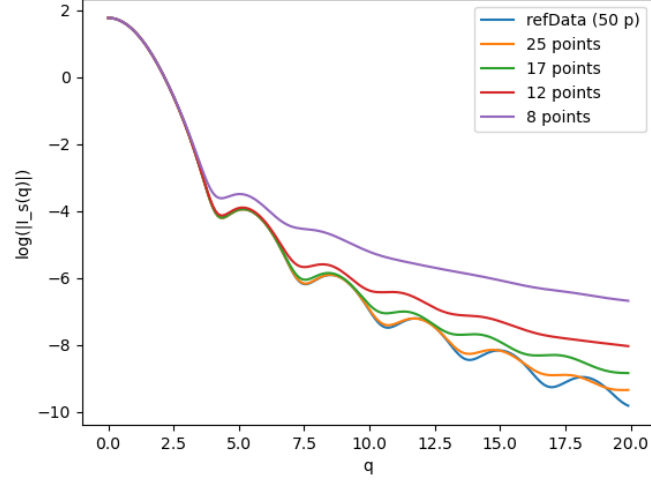


Figure 12: Comparison of different  $\log(I_s(q))$  curves for different choices of  $nrOfGaussPoints$

And in figure 13, the  $\log(I_s(q))$  differences to the  $\log(I_s(q))$ -values of the reference data are shown.

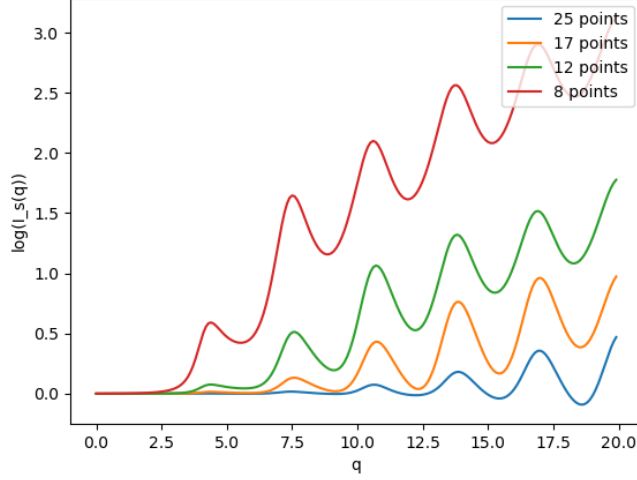


Figure 13: Difference to the  $\log(I_s(q))$ -values of the reference data for different choices of *nrOfGaussPoints*

We see that for higher  $q$ , more interpolation points are necessary to get a good approximation of the correct value of  $\log(I_s(q))$ . When we analyze

$$I_s(q) = \int_{\mathbb{R}^3} \gamma(\mathbf{z}) \operatorname{sinc}(q\|\mathbf{z}\|) d\mathbf{z},$$

we see the reason behind it. The parameter  $q$  occurs (only) in the term

$$\operatorname{sinc}(q\|\mathbf{z}\|) = \frac{\sin(q\|\mathbf{z}\|)}{q\|\mathbf{z}\|}$$

and the function  $\sin(q\|\mathbf{z}\|)$  has a higher frequency (more local maxima) for a higher  $q$  than for a lower  $q$ . In section 2.5.2 we discussed how many interpolation points are necessary to ensure a small enough error.

Before choosing the value of *nrOfGaussPoints*, it is recommended to analyze the given measurement data (a “.csv” file with  $[q, I_{s,meas}(q)]$  values) and plot  $q$  vs.  $\log I_{s,meas}(q)$ . This way, one can see how many local maxima the curve has. In figure 15, we obviously have about 6 local maxima - and we see that 50 interpolation points are sufficient to get a good approximation of  $\log(I_s(q))$  there. About 25 interpolation points are probably enough if there are only 3 local maxima.

To calculate the minimum number of interpolation points, we recall equation (58),

$$\tilde{n}_P := \operatorname{ceil}(3 + 3ql_z(r_3 + 1)/(2\pi)).$$

We can use this equation to check how many interpolation points we need for calculating  $\log(I_s(q))$  for  $q_{max} = 20.0$  our ellipsoid.  $l_z$  is the mesh size in  $z$ -direction and  $r_3$  is the difference between the highest  $z$ -grid-index and the lowest  $z$ -grid-index for the variable boundary. One can estimate that  $l_z(r_3 + 1)$  is about the range of  $\Omega$  in  $z$ -direction. Because we analyze an ellipsoid which third radius is given by  $\hat{r}_3 := 2.0$  and so, the length of the ellipsoid in  $z$ -direction is 4.0, we can conclude that

$$\tilde{n}_P = \text{ceil}(3 + 3 q_{max} l_z (r_3 + 1) / (2\pi)) = \text{ceil}(3 + 3 * 20.0 * 4.0 / (2\pi)) = 42.$$

Therefore, we know that we need at least 42 interpolation points for our highest value  $q = q_{max}$ .

Remark: Different from section 2.5.2, we do not only interpolate in  $z$ -direction, so we have to calculate the number of interpolation points also for the  $x$ - and  $y$ -direction, and set *nrOfGaussPoints* to the maximum of these three values.

### 3.7.2 Influence of *nrOfSpherePoints* on $\log(I_s(q))$

We analyze the influence of the parameter *nrOfSpherePoints* on the  $\log(I_s(q))$  graph. Because plotting all  $\log(I_s(q))$  curves for 6 different choices of *nrOfSpherePoints* was too confusing, the next plot shows the  $\log(I_s(q))$  difference to the reference  $\log(I_s(q))$  curve:

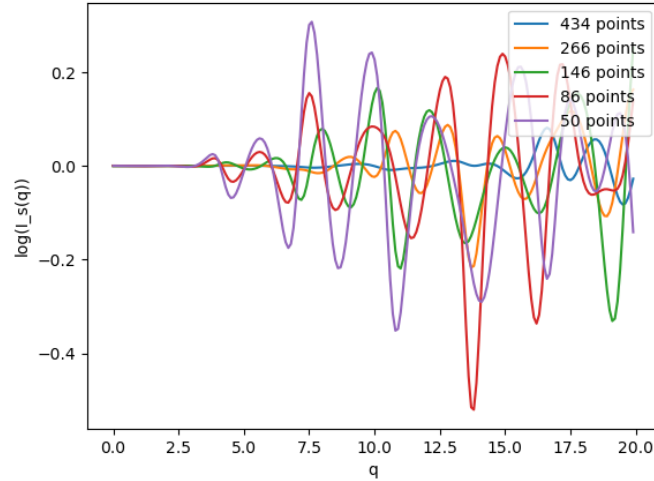


Figure 14: Difference to the  $\log(I_s(q))$ -values of the reference data for different choices of *nrOfGaussPoints*

Recall that the value of  $nrOfSpherePoints$  determines, how many Lebedev interpolation points we use to interpolate on a sphere. Because this is a 2-dimensional interpolation,  $nrOfSpherePoints$  should be considerably larger than  $nrOfGaussPoints$ . When we compare figure 14 to figure 13, we see that:

- Using less Gaussian interpolation points has a huge impact on the error. There is a  $\log(I_s(q))$  difference of about 0.5 already for the choice  $nrOfGaussPoints = 25$ !
- Using less Lebedev interpolation points has a smaller impact on the error. Even for the choice  $nrOfSpherePoints = 50$  the  $\log(I_s(q))$  difference did not exceed 0.5.
- For the choice  $nrOfSpherePoints = 434$  (the blue line in figure 14) the  $\log(I_s(q))$  difference did never exceed 0.1, whereas there was almost no difference for the first 4 local maxima.

Empirically, the choice  $nrOfSpherePoints = 434$  should be good enough to capture accurately the first four local maxima. For 6 local maxima, the choice  $nrOfSpherePoints = 770$  should be taken.

### 3.7.3 Influence of $lengthXY$ on $\log(I_s(q))$

We analyze the influence of the parameter  $lengthXY$  (the mesh size) on the  $\log(I_s(q))$  chart. Because plotting all  $\log(I_s(q))$  curves for 6 different choices of  $lengthXY$  was too unclear, the next plot shows the  $\log(I_s(q))$  difference to the reference  $\log(I_s(q))$  curve for three different choices of  $lengthXY$ :

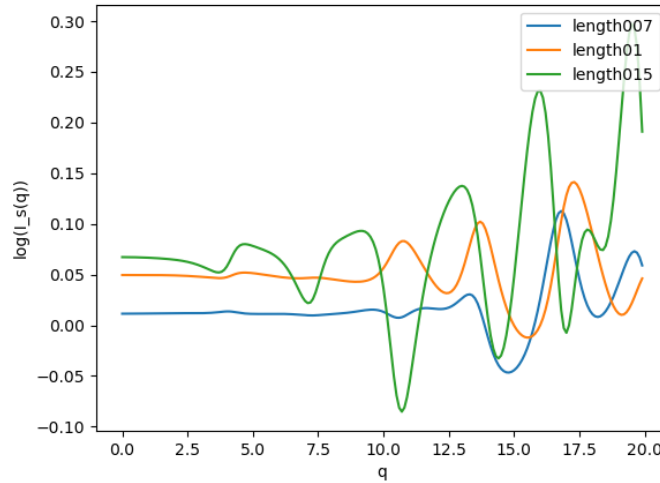


Figure 15: Difference to the  $\log(I_s(q))$ -values of the reference data for different choices of  $lengthXY$

Scaling to a specific volume (section 3.6) was not implemented for the ellipsoid! Therefore there are major differences in the  $\log(I_s(0))$ -values between the curves, because the calculated volumes (number of grid points inside  $\Omega$  times the volume of one grid cell) for different choices of  $lengthXY$  differ.

So, we took the polygonal lines we gathered from an optimization run (about 30 different polygonal lines). For each of these polygonal lines we calculated the  $I_s(q)$ -values for

$$lengthXY \in \{0.15, 0.13, 0.11, 0.10, 0.09, 0.08, 0.07, 0.06, 0.05\},$$

after we scaled  $\Omega$  (depending on the polygonal line) to the same volume. (The scaling factor for different choices of  $lengthXY$  can differ a little bit, because the calculated volume for different choices of  $lengthXY$  can differ.) Then, we plotted the  $\log(I_s(q))$ -differences of each choice of  $lengthXY$  to the most exact setup  $lengthXY = 0.05$ . The next 2 charts show the  $\log(I_s(q))$ -differences for the first polygonal line.

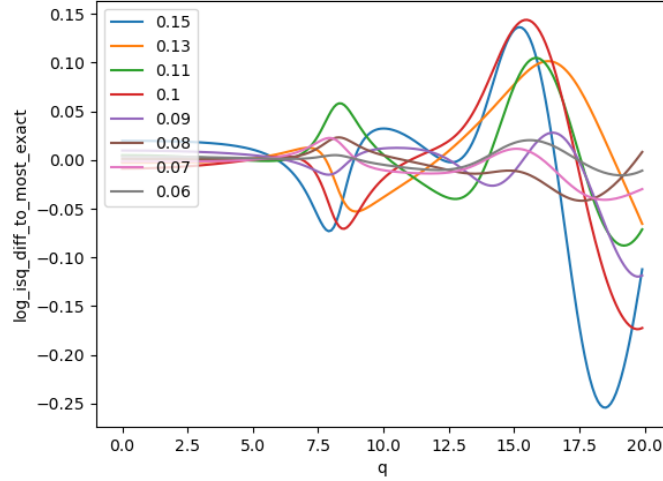


Figure 16: Difference to the  $\log(I_s(q))$ -values of  $lengthXY = 0.05$  for the first polygonal line

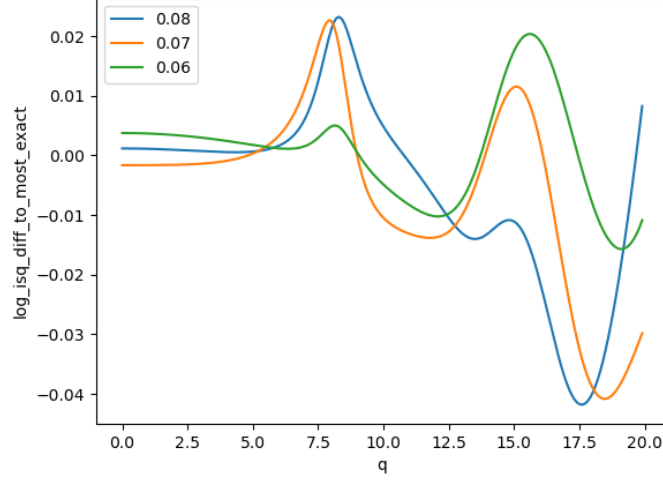


Figure 17: Difference to the  $\log(I_s(q))$ -values of  $lengthXY = 0.05$  for the first polygonal line (only  $lengthXY \in \{0.08, 0.07, 0.06\}$ )

One can see that for the more exact choices of  $lengthXY \in \{0.08, 0.07, 0.06\}$ , there were no huge differences in the  $\log(I_s(q))$ -values. The biggest difference was about 0.04. When we consider our error function

$$J(\Omega) = \int_{q_{min}}^{q_{max}} \left( \ln \frac{I_s(q) + \epsilon}{I_{s, meas}(q) + \epsilon} \right)^2 dq,$$

we see that the squared  $\log(I_s(q))$ -difference is integrated up. Because  $0.04^2 = 0.0016$ , the  $\log(I_s(q))$ -differences between the choices  $lengthXY \in \{0.08, 0.07, 0.06, 0.05\}$  are almost neglectable.

These differences were plotted for 15 different polygonal lines. The biggest  $\log(I_s(q))$ -difference for the last 3 choices of  $lengthXY \in \{0.08, 0.07, 0.06\}$  was observed for the 14<sup>th</sup> polygonal line:



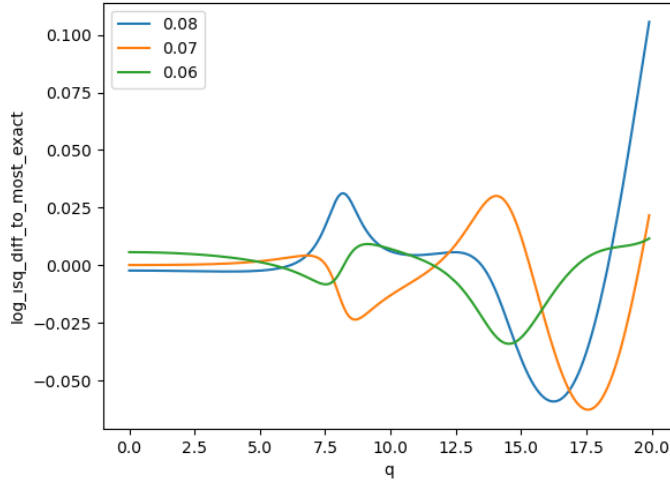


Figure 18: Difference to the  $\log(I_s(q))$ -values of  $lengthXY = 0.05$  for the 14<sup>th</sup> polygonal line (only  $lengthXY \in \{0.08, 0.07, 0.06\}$ )

We have to mention that for these polygonal lines, the  $\log(I_s(q))$ -curves only have 3 local maxima. When we compare it to the  $\log(I_s(q))$ -curve for the ellipsoid (see figure 12), we see that the  $\log(I_s(q))$ -curve for the ellipsoid has about 6 local maxima within the same range of  $q$  and the  $\log(I_s(q))$ -differences are larger at the last 3 local maxima than at the first 3 local maxima.

Therefore, we plotted the  $\log(I_s(q))$ -differences again, but now for  $q$  in the range  $[0, 40]$ .

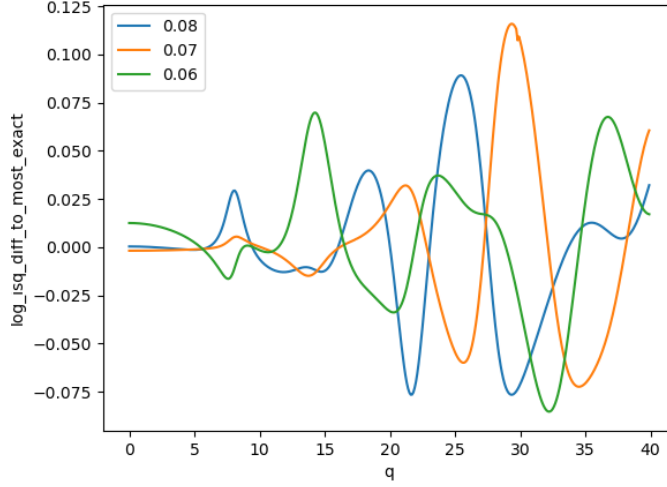


Figure 19: Difference to the  $\log(I_s(q))$ -values of  $lengthXY = 0.05$  for the 15<sup>th</sup> polygonal line (only  $lengthXY \in \{0.08, 0.07, 0.06\}$ )

We see that the  $\log(I_s(q))$ -differences between the different mesh sizes is sometimes larger than 0.1, so we can assume that the  $\log(I_s(q))$ -differences to the “true”  $\log(I_s(q))$ -values (assumed the mesh size is infinitesimal small) can also be in the ballpark of 0.1. We can calculate the squared error between the  $\log(I_s(q))$ -values of the mesh sizes  $lengthXY = 0.05$  and  $lengthXY = 0.06$  and we get

$$\int_0^{40} (\log(I_{s,lengthXY=0.06}(q)) - \log(I_{s,lengthXY=0.05}(q)))^2 dq \approx 0.05.$$

We can estimate and expect that the error to the “true”  $\log(I_s(q))$ -values can also be in this range, this means

$$\begin{aligned} \int_0^{40} (\log(I_{s,lengthXY=0.06}(q)) - \log(I_{s,true}(q)))^2 dq &= \\ &= \int_0^{40} \left( \log \frac{I_{s,lengthXY=0.06}(q)}{I_{s,true}(q)} \right)^2 dq \approx 0.05. \end{aligned} \tag{90}$$

Let us assume that we perform an optimization run with the parameter choice  $lengthXY = 0.06$  to get an optimal polygonal line, given some measurement

data - and at some step we were able to get a polygonal line so that the error

$$J_{lengthXY=0.06}(\Omega) = \int_0^{40} \left( \log \frac{I_{s,lengthXY=0.06}(q) + \epsilon}{I_{s,meas}(q) + \epsilon} \right)^2 dq = 0.05.$$

We set  $\epsilon = 0$ , then we can write  $J_{lengthXY=0.06}(\Omega)$  as

$$\begin{aligned} 0.05 &= J_{lengthXY=0.06}(\Omega) = \int_0^{40} \left( \log \frac{I_{s,lengthXY=0.06}(q)}{I_{s,meas}(q)} \right)^2 dq = \\ &= \int_0^{40} \left( \log \frac{I_{s,lengthXY=0.06}(q)}{I_{s,true}(q)} + \log \frac{I_{s,true}(q)}{I_{s,meas}(q)} \right)^2 dq. \end{aligned} \quad (91)$$

We analyze two possible extreme cases for equation (91):

1.  $I_{s,true}(q) = I_{s,meas}(q)$  for all  $q$ : In this case, we found an optimal shape  $\Omega$ , because the  $I_s(q)$ -values for an infinitesimal small mesh size are the same as the measured  $I_s(q)$ -values. We get

$$\begin{aligned} 0.05 &= \int_0^{40} \left( \log \frac{I_{s,lengthXY=0.06}(q)}{I_{s,true}(q)} + \log \frac{I_{s,true}(q)}{I_{s,meas}(q)} \right)^2 dq = \\ &= \int_0^{40} \left( \log \frac{I_{s,lengthXY=0.06}(q)}{I_{s,true}(q)} \right)^2 dq, \end{aligned}$$

which is a possible outcome because of the considerations for equation (90). Here, the true error here is obviously

$$J_{true}(\Omega) = \int_0^{40} \left( \log \frac{I_{s,true}(q)}{I_{s,meas}(q)} \right)^2 dq = 0.$$

2.  $\log \frac{I_{s,true}(q)}{I_{s,meas}(q)} = -2 \log \frac{I_{s,lengthXY=0.06}(q)}{I_{s,true}(q)}$  for all  $q$ : In this case we also get

$$\begin{aligned} 0.05 &= \int_0^{40} \left( \log \frac{I_{s,lengthXY=0.06}(q)}{I_{s,true}(q)} + \log \frac{I_{s,true}(q)}{I_{s,meas}(q)} \right)^2 dq = \\ &= \int_0^{40} \left( -\log \frac{I_{s,lengthXY=0.06}(q)}{I_{s,true}(q)} \right)^2 dq = \int_0^{40} \left( \log \frac{I_{s,lengthXY=0.06}(q)}{I_{s,true}(q)} \right)^2 dq, \end{aligned}$$

which is again a possible outcome because of equation (90). But here, the true error

$$\begin{aligned} J_{true}(\Omega) &= \int_0^{40} \left( \log \frac{I_{s,true}(q)}{I_{s,meas}(q)} \right)^2 dq = \int_0^{40} \left( -2 \log \frac{I_{s,lengthXY=0.06}(q)}{I_{s,true}(q)} \right)^2 dq = \\ &= 4 \int_0^{40} \left( \log \frac{I_{s,lengthXY=0.06}(q)}{I_{s,true}(q)} \right)^2 dq = 4 * 0.05 = 0.2. \end{aligned}$$

### 3.7.4 Summary

We see that if  $J_{lengthXY=0.06}(\Omega) = 0.05$ , there is a possibility that we found an optimal shape  $\Omega_{opt}$ , but it is also possible that the true error is still as large as 0.2 (which is not really a good approximation). For the choice  $lengthXY = 0.06$ , one calculation of  $I_s(q)$  takes about 30 minutes on the PC we ran the simulation (ASUS VivoBook 17, Processor: AMD Ryzen 7, OS: Microsoft Windows 10 Pro). The calculation of the vector field  $G(\mathbf{x})$  on the boundary of  $\Omega$  for sufficiently many points on  $\partial\Omega$  took about 12 hours.

So the performance is far too slow, because we need several steps (calculations of the vector field  $G(\mathbf{x})$ ) to approach an optimal  $\Omega$ . Moreover, the accuracy is also quite bad and therefore, a good optimization run is unfortunately not possible, at least:

1. on this PC,
2. with this ansatz.

Remember, the calculation of  $I_s(q)$  we used until now was developed for the general case, not only for rotational bodies  $\Omega$ . For practical applications, we therefore use another algorithm that is a lot faster, but it only works for rotational bodies that have the  $z$ -axis as a rotational axis.

Idea: Calculate  $I_s(q)$  by integrating areas of lens-shaped domains.

## 3.8 Algorithm to calculate $I_s(q)$ for rotational bodies

Contrary to the other algorithms, this algorithm was written in MATLAB and not in *Python*. The MATLAB file of this function is called *calc\_isq.m* and was written by Wolfgang Ring. Because we do not want to get too much into depth, we only discuss the input parameters, the output and good parameter choices.

The input parameters of this algorithm are:

- $x1\_vec, x3\_vec$ : These are the coordinates of the 2d polygonal line that is rotated around the  $z$ -axis to form the boundary of  $\Omega$ . Obviously,  $x1\_vec$  are the  $x$ -coordinates of the nodes of the polygonal line and  $x3\_vec$  are the  $z$ -coordinates.  $x1\_vec$  and  $x3\_vec$  have to have the same length  $N_\Omega$ . When we choose  $N_\Omega$  to be larger, it does not only affect the resolution of the polygonal line - the calculation of  $I_s(q)$  is also more exact.

- $r\_steps$ : Recall Theorem 1.4 as

$$I_s(q) = \int_0^{r_{max}} r^2 \text{sinc}(rq) \int_{\mathbf{p} \in S^2} \gamma(r\mathbf{p}) dS(\mathbf{p}) dr.$$

The integrals  $\int_{\mathbf{p} \in S^2} \gamma(r\mathbf{p}) dS(\mathbf{p}) dr$  are calculated for  $r\_steps$  equidistant radii  $r$  in the interval  $[0, r_{max}]$ , whereas  $r_{max}$  is calculated so that  $r_{max} \geq \sup_{\mathbf{x}, \mathbf{y} \in \Omega} \|\mathbf{x} - \mathbf{y}\|_2 = \text{diam}(\Omega)$ .

- $q\_vals$ : The array of  $q$ -values to calculate  $I_s(q)$ .
- $deg\_quad$ : To calculate  $\int_{\mathbf{p} \in S^2} \gamma(r\mathbf{p}) dS(\mathbf{p}) dr$  for a rotational body  $\Omega$ , it is only necessary to calculate  $\gamma(r\mathbf{p})$  for  $\mathbf{p} \in \{[\sin(\theta), 0, \cos(\theta)] \mid \theta \in [0, \pi/2]\}$ . This result was obtained in the paper *Model based estimation of small angle scattering data* of Wolfgang Ring and Karin Kornmüller. To integrate over  $\theta \in [0, \pi/2]$ , Gaussian integration of order  $deg\_quad$  is used.

The output of the algorithm is a list (respectively a vector) of  $I_s(q)$ -values for all  $q$ -values for the input parameter  $q\_vals$ .

To call this m-function within Python, we use “GNU Octave”, which is an open source software that can execute “.m” MATLAB-files. The 3 following Python commands are sufficient to call *calc\_isq.m*:

- `from oct2py import octave`  
After installing GNU Octave and oct2py, this is how we set up the “Python-to-Octave” interface.
- `octave.addpath('./mfiles')`  
The directory containing the m-files we want to call has to be added.
- `isqVals = octave.calc_isq(x1_vect, x3_vect, r_points, qVals, gauss_points_mfile)`  
This is how we call *calc\_isq.m*.

### 3.8.1 Parameter choices

For this algorithm, we are able to make the following 3 parameter choices:

- $N_\Omega$ : The number of nodes of the 2d polygonal line respectively the length of the vectors  $x1\_vec$  and  $x3\_vec$ .
- $r\_steps$
- $deg\_quad$

Very exact results could be obtained with the parameter choices

- $N_{\Omega} = 500$ ,
- $r\_steps = 500$ ,
- $deg\_quad = 200$ .

Because we want to test our algorithm on a practical example, we test it on the shape of different LDL particles obtained by the “cryogenic electron microscopy” technique, in short, “cryo-EM” technique. An overview over different “cryo-EM” methods is given in the book *CryoEM* by Tamir Gonen and Brent L. Nannenga [17]. By applying the “cryo-EM” technique, Karin Kornmüller obtained the following rotation symmetrically shapes of different LDL particles:

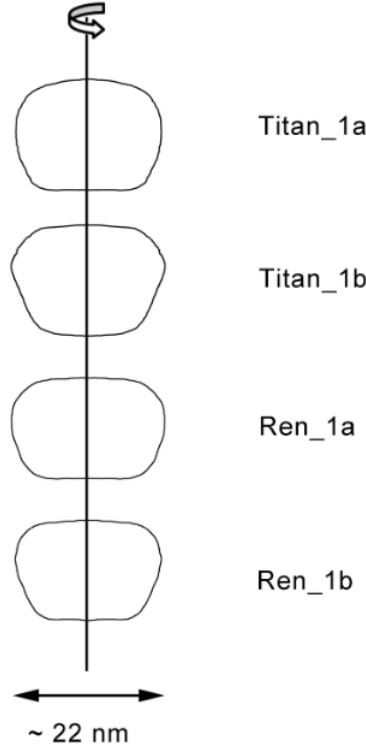


Figure 20: LDL particle shapes obtained by the cryo-EM technique

Of those 4 particles, we decide to reconstruct the “Titan\_1b”-particle. So, we first calculate the  $I_s(q)$ -data for the “Titan\_1b”-shape by executing the following steps:

- We retrieve a 2d polygonal line from an image which is the boundary of the area of rotation of the “Titan\_1b”-particle. The height is about 400 pixel.

- We set  $pixellen = 0.002$ , so the height of the “Titan\_1b”-shape is about  $0.002 * 400 = 0.8$ .
- We set  $q\_vals = \{0.0, 0.1, 0.2, \dots, 39.9, 40.0\}$  and retrieve the  $I_s(q)$ -data for the “Titan\_1b”-shape by executing *calc\_isq.m*.

These are the  $\log(I_s(q))$ -values for the “Titan\_1b”-shape:

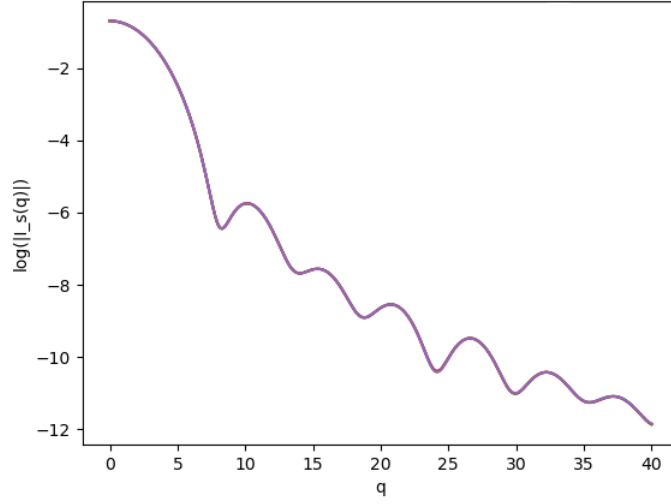


Figure 21:  $\log(I_s(q))$ -values of the shape Titan\_1b

We see that this graph has 7 local maxima - therefore the retrieved  $I_s(q)$ -data meets the requirement “at least 6 local maxima” for the measurement data to reconstruct the original shape (in this case, the “Titan\_1b”-shape).

These are the  $\log(I_s(q))$ -differences when we make the choice  $N_\Omega \in \{300, 200, 100\}$ :

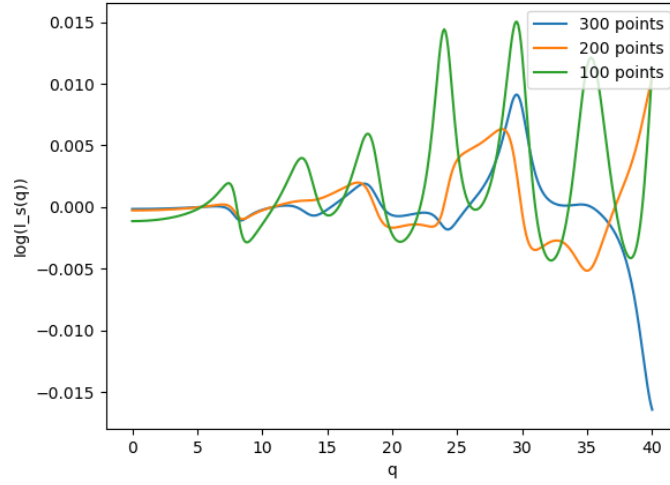


Figure 22: Difference to the  $\log(I_s(q))$ -values to  $N_\Omega = 500$  for the choices  $N_\Omega \in \{300, 200, 100\}$

These are the  $\log(I_s(q))$ -differences when we make the choice  $r\_steps \in \{300, 200, 100, 50\}$ :

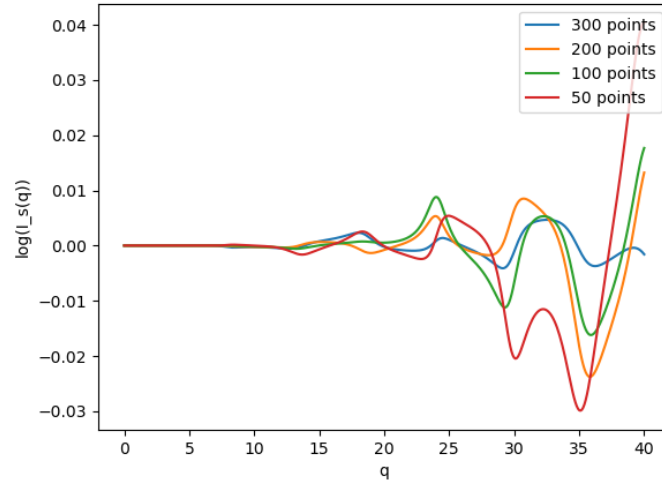


Figure 23: Difference to the  $\log(I_s(q))$ -values to  $r\_steps$  for the choices  $r\_steps \in \{300, 200, 100, 50\}$



These are the  $\log(I_s(q))$ -differences when we make the choice  $\text{deg\_quad} \in \{120, 80, 50, 30\}$ :

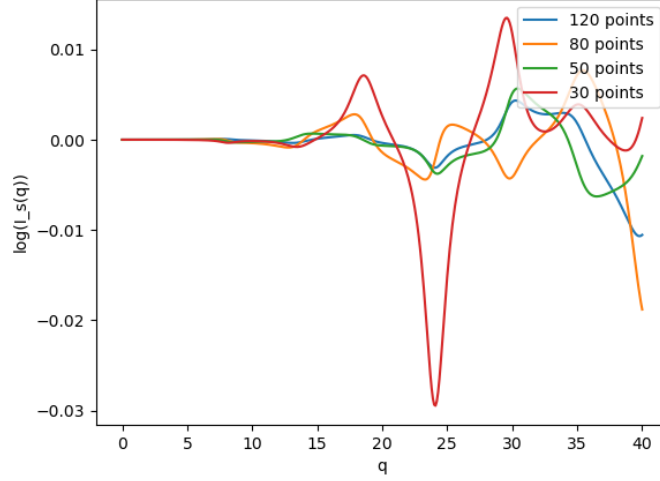


Figure 24: Difference to the  $\log(I_s(q))$ -values to  $\text{deg\_quad} = 200$  for the choices  $\text{deg\_quad} \in \{120, 80, 50, 30\}$

We see that the  $\log(I_s(q))$ -differences to the most exact calculation are a lot smaller than in section 3.7. Of all 3 parameters, we also see that the choice of  $r\_steps$  has the biggest impact on the  $\log(I_s(q))$ -difference. A  $\log(I_s(q))$ -difference of 0.02 does not have a huge impact on the overall squared error (because  $0.02^2$  is relatively small). Therefore, we set

- $N_\Omega = 200$ ,
- $r\_steps = 300$ ,
- $\text{deg\_quad} = 120$

and we expect to get about the same results for the  $\log(I_s(q))$ -data for different shapes as for the most exact parameter choice.

### 3.9 Information about the test runs for this thesis

For this thesis, more than 25 different simulation runs with different algorithm choices were made to reconstruct a particle. In the directory of the *main.py* Python file, all data of those simulation runs is stored in the directory *optimization*. The following data is stored:

- *measuredIsq.csv*: The  $I_{s,meas}(q)$  data of the particle we want to reconstruct

- Directory *polygonData*: The progression of the polygonal line data throughout the algorithm
- Directory *isqData*: The progression of the  $I_s(q)$  data
- Directory *isqData*: Images of the progression of the polygonal line
- Directory *logisqCharts*: Charts of the progression of the  $\log(I_s(q))$ -data compared to the given  $\log(I_{s,meas}(q))$  data.
- *errorInfo.csv*: log file with error information throughout the algorithm

Most of the runs used the old general ansatz which was either too computational expensive or too inaccurate (when choosing non-optimal parameters). And only 4 simulation runs used the ansatz described at section 3.8, whereby the chosen algorithm is a random search algorithm described in the next chapter. 3 of these 4 simulation runs did not yield good results, because of:

- bad parameter choices
- bad starting polygonal line: the algorithm was stuck in a local minimum which was obviously not a global minimum.

The one remaining successful simulation run is described in the end of the next chapter in section 4.6.

## 4 Random Search ansatz to find $\Omega_{opt}$

For the mathematical ansatz in chapter 3, calculating the change direction on the boundary of  $\Omega$  has taken a lot more time (in most tests about 100 times slower) than just calculating  $I_s(q)$  and the corresponding error. Therefore, another ansatz to approach  $\Omega_{opt}$  was chosen.

We chose a random search ansatz, where we make random changes on the 2d polygonal line  $\hat{P}$  in each iteration. The objective formula we want to minimize consists of 2 parts:

- our error term  $J(\Omega(\tilde{P}))$ ,
- the "misshape" penalty term  $\vartheta(\tilde{P})$ .

The smoother the polygonal line, the smaller is the "misshape" penalty (see sections 4.2 and 4.3). For one iteration, we decided to only make local changes (see section 4.4). And we use the Simulated Annealing method [10] to decide whether a change in an iteration is accepted or declined (section 4.5).

We then show an unsuccessful reconstruction of the "Titan\_1b" particle. The reconstruction was unsuccessful, because

- The algorithm was stuck in a local minimum that was not a global minimum. This problem could be solved by choosing another starting polygonal line.
- The parameters of the simulated annealing algorithm (e.g. the temperature) were not set optimally
- We did not use the improved change strategy given in 4.4.

Finally, we show a successful reconstruction of the "Titan\_1b" particle 4.7.

### 4.1 Limitations on $\Omega$ for the Random Search ansatz

For the Random Search ansatz, we also constrict  $\Omega$  to a body of rotation, whereas the boundary is a polygonal line that is rotated around the  $z$ -axis. But there are further restrictions for  $\Omega$ , which are discussed later.

To construct the 2d polygonal line  $\tilde{P}$  (equation (72)), we define a midpoint  $\mathbf{m} = [0, m_z]$  on the  $z$ -axis, and we define the array

$$dists\_to\_mid = \{d_1, \dots, d_{N_\Omega}\}, \quad d_i \in [d_{min}, d_{max}],$$

and set  $d_{min} := 20$  and  $d_{max} := 600$ . The polygon is then defined in polar coordinates w.r.t.  $\mathbf{m}$  with equidistant angles. So, the array  $dists\_to\_mid$  are the distances of the nodes of the polygonal line  $\tilde{P}$  to the midpoint  $\mathbf{m}$  in pixel in clockwise order. After creating the 2d polygonal line, we scale the polygonal line to our desired volume like described in section 3.6. So, the nodes of  $\tilde{P}$  are created like described by the following pseudo-code:

```

for i in range(len(dists_to_mid)):
    angle = np.pi * i / (len(dists_to_mid) - 1)
    # first point: angle = 0, second point: angle = pi/N_Omega,...,
    # last point: angle = pi
    polygon.append([m_z - dists_to_mid[i] * np.cos(angle),
                    dists_to_mid[i] * np.sin(angle)])
polygon = zoom_polygon_to_volume(polygon, measuredVolume)
polygon = np.round(np.array(polygon)).astype(int)
# coordinates for the nodes of the pol. line should be integers

```

We can see that for  $i \in \{1, \dots, N_\Omega\}$ , the angle of the connecting line  $c_i$  between the node  $\tilde{p}_i$  of  $\tilde{P}$  and the midpoint  $\mathbf{m}$ , and the  $z$ -axis is always

$$\alpha_i = \frac{\pi(i-1)}{N_\Omega - 1}.$$

The advantages of this ansatz are:

- It is impossible that the polygonal line  $\tilde{P}$  intersects itself. (This is easy to see, because  $\alpha_i$  is strictly monotonically increasing as  $i$  increases. Considering every angle between the connecting line from  $\mathbf{x} \in \tilde{P}$  to  $\mathbf{m}$  and the  $z$ -axis, we see that this angle is also strictly monotonically increasing as we walk through the polygonal line  $\tilde{P}$ .) Moreover, all  $d_i$  are bounded, so all restrictions for  $\tilde{P}$  (respectively  $\hat{P}$  and  $P$ ) in section 3.1.1 are fulfilled.
- It is easier to control that  $\Omega$  stays sufficiently regular in comparison to the ansatz from last chapter.

And the disadvantages of this ansatz are:

- $\Omega$  is further restricted to a star-shaped domain with midpoint  $\Phi_1(\mathbf{m})$  (this means: the connecting line of every point  $\mathbf{x} \in \Omega$  and  $\Phi_1(\mathbf{m})$  is a subset of  $\Omega$ ).
- The angles  $\alpha_i$  stay constant, which is a further restriction to the shape of  $\Omega$ .

Alltogether, we can write  $\Omega$  as

$$\Omega = \Omega(\tilde{P}) = \Omega(\tilde{P}(\text{dists\_to\_mid})).$$

In the next section we define the objective function we want to minimize.

Moreover, when we approach  $\Omega_{opt}$ , we do not change the polygonal line nodes directly. Instead, we change the array *dists\_to\_mid*. The strategy how we change this array is discussed in section 4.4.

## 4.2 The objective function to minimize

The objective function we want to minimize is given by

$$\tilde{J}(\text{dists\_to\_mid}) := J(\Omega(\tilde{P}(\text{dists\_to\_mid}))) + \lambda_{mis} \vartheta(\tilde{P}(\text{dists\_to\_mid})),$$

whereas  $J(\Omega)$  is the error term given in equation (19) and  $\vartheta(\tilde{P})$  is a “mis-shape” penalty term for the polygonal line. Consequently, the variable and factor

$$\lambda_{mis} := \text{pol\_misshape\_penalty}$$

determines the influence of the penalty term  $\vartheta(\tilde{P})$  on the overall error.

## 4.3 The penalty term $\vartheta(\tilde{P})$

The idea and the first approach to define the penalty term  $\vartheta(\tilde{P})$  was to add up the square of the angle differences of all 2 consecutive connecting lines of the polygonal line. Additionally, the connecting line between the first two points of the polygonal line should be close to perpendicular to the  $z$ -axis, as well as the connecting line between the last two points of the polygonal line. So, the first approach was to define the connecting lines  $w_1, \dots, w_{N_\Omega - 1}$  as

$$w_i = \tilde{p}_{i+1} - \tilde{p}_i, i = 1, \dots, N_\Omega - 1.$$

Moreover, because we want  $w_1$  to be as close to a (positive) multiple of  $[1, 0]$  and  $w_{N_\Omega - 1}$  to be as close to a (positive) multiple of  $[-1, 0]$ , we define

$$\begin{aligned} w_0 &= [1, 0], \\ w_{N_\Omega} &= [-1, 0], \end{aligned}$$

and the angle differences between the connecting lines of the polygonal line as

$$\alpha_j = \arccos \left\langle \frac{w_j}{\|w_j\|}, \frac{w_{j+1}}{\|w_{j+1}\|} \right\rangle, \quad \text{for } j = 0, \dots, N_\Omega - 1.$$

Then, we can define a penalty term  $\tilde{\vartheta}(\tilde{P})$  (not the one we used in the end) as

$$\tilde{\vartheta}(\tilde{P}) := \sum_{j=0}^{N_\Omega - 1} \alpha_j^2.$$

To minimize the penalty term  $\tilde{\vartheta}(\tilde{P})$ , the optimal shape is a (almost perfect) half circle (w.o. proof). So, the minimization of the objective function is a balance between minimizing the error term  $J(\Omega)$  and minimizing the square of the angle differences between all consecutive connecting lines of the polygonal line.

But, by trying out this penalty term, 2 problems occurred:

1. By setting  $\lambda_{mis}$  too high, almost all changes of the polygonal line were rejected, because the changes often generated small spikes in the polygonal line and the reduction of the error term was only minimal.

2. By setting  $\lambda_{mis}$  too low, the angle difference between several consecutive lines of the polygonal line often got too high (e.g.  $\alpha_{10}, \alpha_{11}, \dots, \alpha_{15}$  were all about 10 degrees, then the angle difference between all 2 consecutive lines was OK, but the angle difference between  $w_{10}$  and  $w_{15}$  was too high in this instance, and the polygonal line was not well shaped anymore).

Therefore, the idea is to consider the angle difference between the lines that connect every  $n$ -th node of the polygonal line. So for e.g.  $n = 10$ , we define  $w_1 = p_1 1 - p_1, w_2 = p_2 1 - p_1 1, w_3 = p_3 1 - p_1 1, \dots$ , and sum up the square of the angle difference between  $w_{i+1}$  and  $w_i$ . This way, we allow local spikes, but overall, the polygonal line stays well shaped. In the following, we give the exact definition and the variables used to define our penalty term  $\vartheta(\tilde{P})$ .

We introduce the variable

$$n_{mis} := \text{pol\_points\_for\_misshape},$$

which determines how many points of the polygonal line are at least considered to determine  $\vartheta(\tilde{P})$ . The variable

$$m_{div} := \text{check\_every\_nth\_polpoint} := \text{floor}\left(\frac{N_\Omega}{n_{mis}}\right)$$

determines that only the points of the polygon which zero based indices are divided by  $m_{div}$  are included to determine  $\vartheta(\tilde{P})$ , so

$$\tilde{p}_i \text{ included to determine } \vartheta(\tilde{P}) \Leftrightarrow m_{div} \text{ divides } (i - 1).$$

Let

$$\tilde{n}_{mis} := \text{floor}\left(\frac{N_\Omega}{m_{div}}\right),$$

then the exact number of included points to determine  $\vartheta(\tilde{P})$  is  $(\tilde{n}_{mis} + 1)$ . We define the vectors  $v_1, \dots, v_{\tilde{n}_{mis}}$  as

$$\begin{aligned} v_i &:= \tilde{p}_{i m_{div} + 1} - \tilde{p}_{(i-1) m_{div} + 1} \quad \text{for } i = 1, \dots, (\tilde{n}_{mis} - 1), \\ v_{\tilde{n}_{mis}} &:= \tilde{p}_{N_\Omega} - \tilde{p}_{(\tilde{n}_{mis}-1) m_{div} + 1}. \end{aligned}$$

These vectors are the connections between the points of the polygonal line which indices are multiples of  $m_{div}$ . Moreover, we define

$$\begin{aligned} v_0 &:= [0, 1] \quad \text{and} \\ v_{\tilde{n}_{mis}+1} &:= [0, -1]. \end{aligned}$$

The angle differences  $\beta_0, \dots, \beta_{\tilde{n}_{mis}}$  between two successive vectors are calculated as follows:

$$\beta_j := \arccos \left\langle \frac{v_j}{\|v_j\|}, \frac{v_{j+1}}{\|v_{j+1}\|} \right\rangle, \quad \text{for } j = 0, \dots, \tilde{n}_{mis}.$$

The penalty term  $\vartheta(\tilde{P})$  is then set to

$$\vartheta(\tilde{P}) := \sum_{j=0}^{\tilde{n}_{mis}} \beta_j^2$$

Our choices for the variables  $n_{mis}$  and  $\lambda_{mis}$  are

$$\begin{aligned} n_{mis} &:= 50, \\ \lambda_{mis} &:= 0.002. \end{aligned}$$

By defining  $\vartheta(\tilde{P})$  like we did, we do not necessarily enforce local smoothness of the polygonal line, but we ensure some sort of “global” smoothness. So the polygonal line can be “spiky” locally, but the overall optimal shape of the polygonal line is still a half circle. In the next chapter, the “Change 2” on  $\Omega$  is our way to ensure that the polygonal line does not get too “spiky” locally.

#### 4.4 Changes on $\Omega$ in one step

We want to make changes locally on  $\Omega$  by changing the array *dists\_to\_mid* locally. Therefore, we introduce the parameter *change\_range\_half*. For each change of the array *dists\_to\_mid*, there exists an index  $i_0 \in \{0, \dots, N_\Omega - 1\}$  so that *dists\_to\_mid* is only altered at the indices

$$i \in I := \{i_0 - \text{change\_range\_half}, \dots, i_0 + \text{change\_range\_half}\}.$$

Obviously, the number of indices in  $I$  are

$$\text{change\_range} := (2 * \text{change\_range\_half} + 1).$$

Because we want to make somewhat smooth changes on *dists\_to\_mid*, we introduce the array *change\_factor\_nearby* as

```
temp_ar = np.linspace(-np.pi, np.pi, num=(2 * change_range_half + 1))
change_nearby = 0.5 * (np.cos(temp_ar) + 1)
```

The temporary array *temp\_ar* is set to *change\_range* equidistant points from  $-\pi$  to  $\pi$  and the array *change\_nearby* has a shape of a cosine function from  $-\pi$  to  $\pi$  with values between 0 and 1. *change\_nearby* is also an array of length *change\_range*. Because we only want to make small changes to *dists\_to\_mid* at one update of  $\Omega$ , we also introduce the variable *max\_step\_length*. When we make a change on the array *dists\_to\_mid*, we add a multiple  $\alpha$  of *change\_nearby* to the indices  $I$  of *dists\_to\_mid*, whereas it holds true that  $-\text{max\_step\_length} \leq \alpha \leq \text{max\_step\_length}$ .

Moreover, we want that the polygonal line stays well-shaped - this means, successive values in the array *dists\_to\_mid* should not be too different. So we use 3 different ways to change *dists\_to\_mid*.

**Change 1:** First, we generate a random index  $i_0 \in \{0, \dots, N_\Omega - 1\}$  and we generate a random step length  $\alpha$  uniformly from the range

$$[-max\_step\_length, max\_step\_length].$$

Then, the following code is executed

```
for i in range(i_0 - change_range_half, i_0 + change_range_half):
    if i >= 0 and i < len(dists_to_mid):
        dists_to_mid[i] += alpha *
            change_nearby[i - (i_0 - change_range_half)]
```

So, the largest change to  $dists\_to\_mid$  is made at the index  $i_0$  and the only indices of  $dists\_to\_mid$  that get changed are the indices  $i \in I$ .

**Change 2:** First, a random index

$$i_0 \in \{change\_range\_half, \dots, N_\Omega - change\_range\_half - 1\}$$

is chosen and we set

```
d_1 = dists_to_mid[i_0 - change_range_half],
d_2 = dists_to_mid[i_0 + change_range_half].
```

The linear interpolation

$$\phi : [i_0 - change\_range\_half, i_0 + change\_range\_half] \rightarrow \mathbb{R}$$

with

$$\begin{aligned}\phi(i_0 - change\_range\_half) &= d_1, \\ \phi(i_0 + change\_range\_half) &= d_2\end{aligned}$$

is given by

$$\phi(i) = (i - (i_0 - change\_range\_half))d_2 + ((i_0 + change\_range\_half) - i)d_1.$$

To smooth the array  $dists\_to\_mid$  for the indices  $i \in I$ , we set every value of  $dists\_to\_mid[i]$  to the mean of its actual value and the value of  $\phi(i)$  as follows:

```
for i in range(i_0 - change_range_half + 1, i_0 + change_range_half):
    dists_to_mid[i] = 0.5 * (dists_to_mid[i] + phi(i))
```

**Change 3:** We set  $i_0$  randomly to one of the values  $\{0, N_\Omega - 1\}$  and we continue to apply Change 1. This was necessary, because  $dists\_to\_mid$  was not changed often enough at the first and the last indices.

For every step, one of these three changes is applied by some predefined probability. Good results were made by setting:

$$\begin{aligned}P(\text{Change 1}) &= 0.5, \\ P(\text{Change 2}) &= 0.4, \\ P(\text{Change 3}) &= 0.1.\end{aligned}$$



## 4.5 Accepting or declining the changes on $\Omega$

Before applying one of the changes in section 4.4, the values of the variable  $dists\_to\_mid$  are saved in the variable  $old\_dists\_to\_mid$ . These old values of  $dists\_to\_mid$  are only needed as a backup if the change was declined.

To accept or decline a change on  $\Omega$  respectively  $dists\_to\_mid$ , we use the well-known strategy called “Simulated Annealing”. In the book *Simulated Annealing: Theory and Applications* by J. M. van Laarhoven, he explains the algorithm and he gives examples where the algorithm is used. In short, whenever our objective function  $\tilde{J}(dists\_to\_mid)$  decreases, we accept the change. And whenever  $\tilde{J}(dists\_to\_mid)$  increases, we accept the change only to some probability, whereby the probability decreases the more  $\tilde{J}(dists\_to\_mid)$  increased. Therefore we introduce the variable  $T := temperature$ .

Whenever  $\tilde{J}(dists\_to\_mid)$  increases, we accept the change at probability

$$P_{accept} = \exp \left( \frac{\tilde{J}(old\_dists\_to\_mid) - \tilde{J}(dists\_to\_mid)}{T} \right).$$

In addition, we also introduce the constant  $\mu_T := temperature\_decline$ . At the end of every step, the temperature is set to

$$T := \mu_T T.$$

Obviously,  $\mu_T$  should be set to a value between 0 and 1. Frequently chosen values are between 0.9 and very close to 1 (e.g. 0.99999).

When the temperature  $T$  approaches 0, the probability that a change on  $dists\_to\_mid$  is accepted whenever  $\tilde{J}(dists\_to\_mid)$  increases, also approaches 0. Then, it is likely that  $\tilde{J}(dists\_to\_mid)$  is stuck at a local minimum that is not a global minimum. To counteract this behaviour, we set

$$T := 2T \tag{92}$$

every time when 10 consecutive changes on  $dists\_to\_mid$  are declined. Because then, changes on  $dists\_to\_mid$  that increase  $\tilde{J}(dists\_to\_mid)$  are much more likely to be accepted in the next steps.

The chosen value for  $T$  before the first step is

$$T := 0.002$$

and the chosen value for  $\mu_T$  is

$$\mu_T := 0.99.$$

The choice  $\mu_T = 0.99$  indicates that  $T$  declines relatively fast. Usual values for  $\mu_T$  are usually between 0.995 and 0.9999. The advantage for our choice is that there is a faster convergence to a local minimum. To not get stuck at a local minimum, we additionally implemented the strategy (92).

#### 4.6 Unsuccessful test run to reconstruct the polygonal line for the Titan\_1b particle and modifications for the algorithm

Beforehand, we calculated the  $\log(I_s(q))$  values for the following “Titan\_1b” particle using the parameter choices described in chapter 3.8.1:

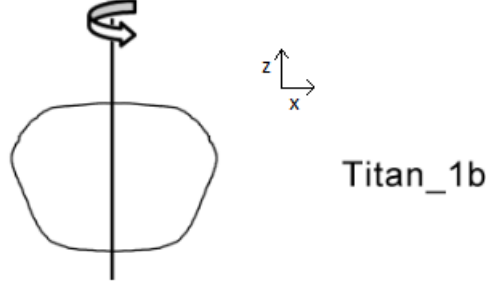


Figure 25: Shape of the Titan\_1b particle

We retrieved the following  $\log(I_s(q))$  values for  $q$  in the range  $[q_{min} = 0.0, q_{max} = 40.0]$ :

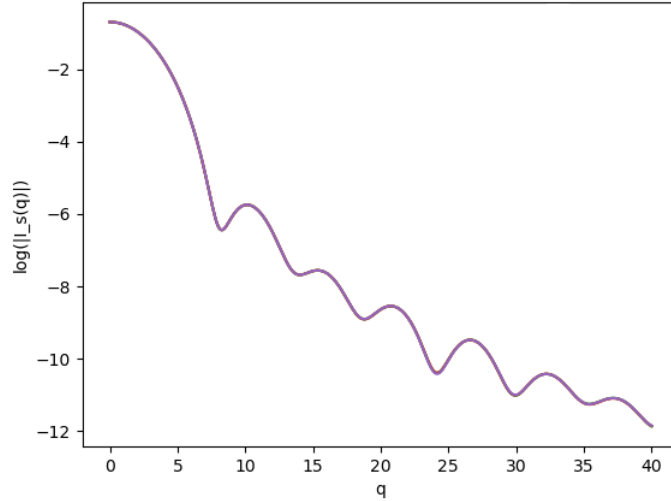


Figure 26:  $\log(I_s(q))$  values for the Titan\_1b particle

To reconstruct the Titan\_1b particle, these  $\log(I_s(q))$  values are given and we try to reconstruct the shape in figure 25.

We mentioned in section 3.9 that only 4 simulation runs were made with the ansatz described in section 3.8. One of those simulation runs (the data is found in the directory *simulated\_annealing\_1*) got stuck in a local minimum. The main reason herefor was our choice for a starting polygonal line - we made the straight-forward choice that the starting 2d polygonal line should be shaped like a half circle. These are the polygonal lines for the ansatz that got stuck in a local minimum:

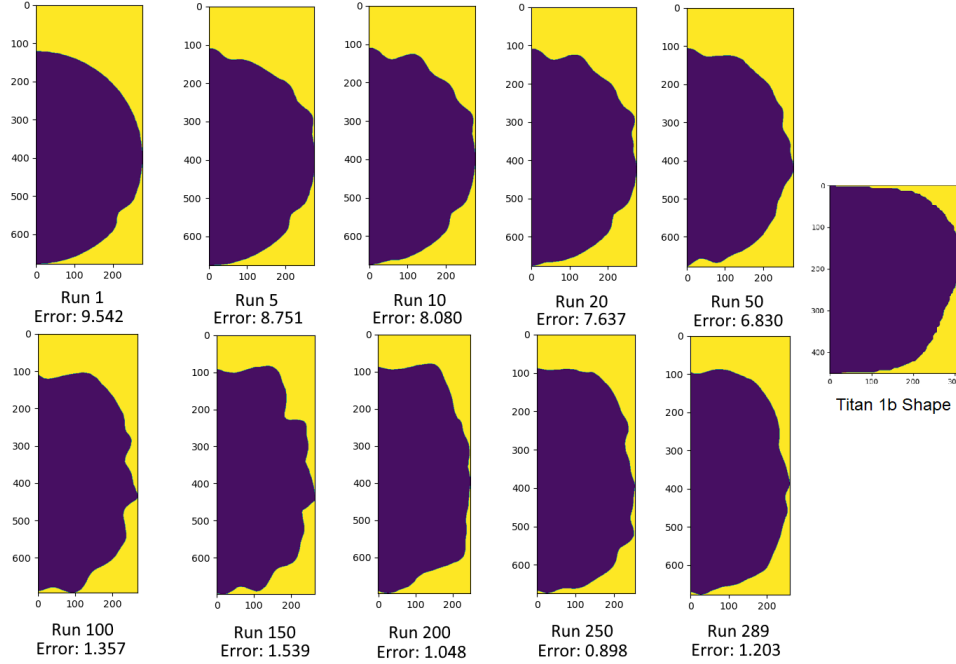


Figure 27: Ansatz that was stuck in a local minimum

One can see that the algorithm changed the initial shape towards the “wrong” local minimum - the vertical extension increased and the horizontal extension decreased - but the opposite should be the case when we want to approach the “Titan\_1b” shape. We see that the error increased between Run 100 and Run 150, as well as between Run 250 and Run 289 (the last run where we finished the simulation) - this is a usual behaviour for the simulated annealing algorithm to ensure to get out of a local minimum. There was no need to find the exact local minimum the shape is converging to, but we see from Run 100 to Run 289 that the shape did not change a lot. So it is very unlikely the algorithm manages to converge to another better local minimum later on.

It was implemented that the parameters for the algorithm could be changed between every 2 runs, e.g. the factor  $\lambda_{min}$  was increased around Run 150 so that the 2d polygonal line got smoother afterwards. Around Run 150, the temperature  $T$  was also set to a smaller value so that the algorithm converges

faster to a local minimum. Altogether, the temperature was way too high throughout the algorithm (bad changes were accepted very often), so we did not get a glance what the optimal shape for this local minimum would be.

The most important change for the next 3 simulations is the starting 2d polygonal line. We changed it to the following polygonal line:

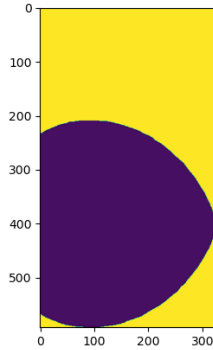


Figure 28: Starting polygonal line so that the algorithm can converge

The next 2 simulations (the data is found in the directories *simulated\_annealing\_2* and *simulated\_annealing\_3*) were mainly important to improve the algorithm and find good choices for the important parameters. These were the final polygonal lines for these 2 simulations:

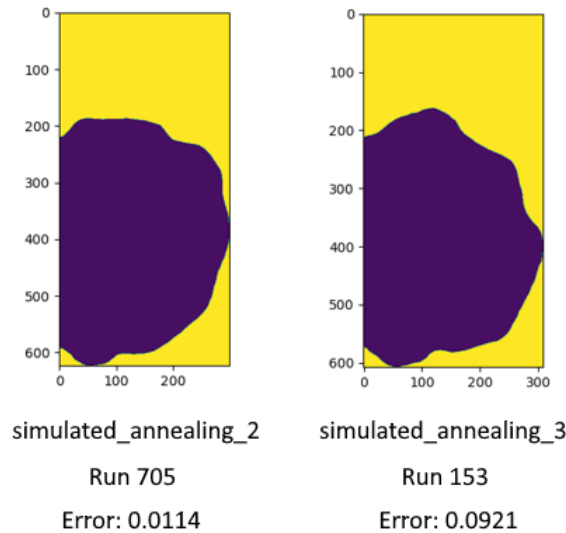


Figure 29: The 2 simulations to get good choices for the parameters

Then we compared different runs of the simulation and we saw that there were no major changes of the coordinates of the first and the last node of the polygonal line. Therefore, the “Change 3” explained in section 4.4 was implemented. Furthermore, we saw that the polygonal line converged fast for  $T$  in the range  $[0.001, 0.002]$ . So we initialized  $T$  as  $T = 0.002$  and set  $\mu_T := 0.99$ . And for every time no change was accepted 10 times in a row, we doubled  $T$ . These considerations are described in section 4.5. Furthermore, a good balance between the true error  $J(\Omega)$  and the shape penalty term  $\vartheta(\Omega)$  was found by setting  $\lambda_{mis} := 0.002$ .

#### 4.7 Successful testrun to reconstruct the polygonal line for the Titan\_1b particle

The final simulation run (the data is found in the directory *simulated\_annealing\_4*) also started with the polygonal line given at figure 28. The next figure shows the progression of the polygonal line:

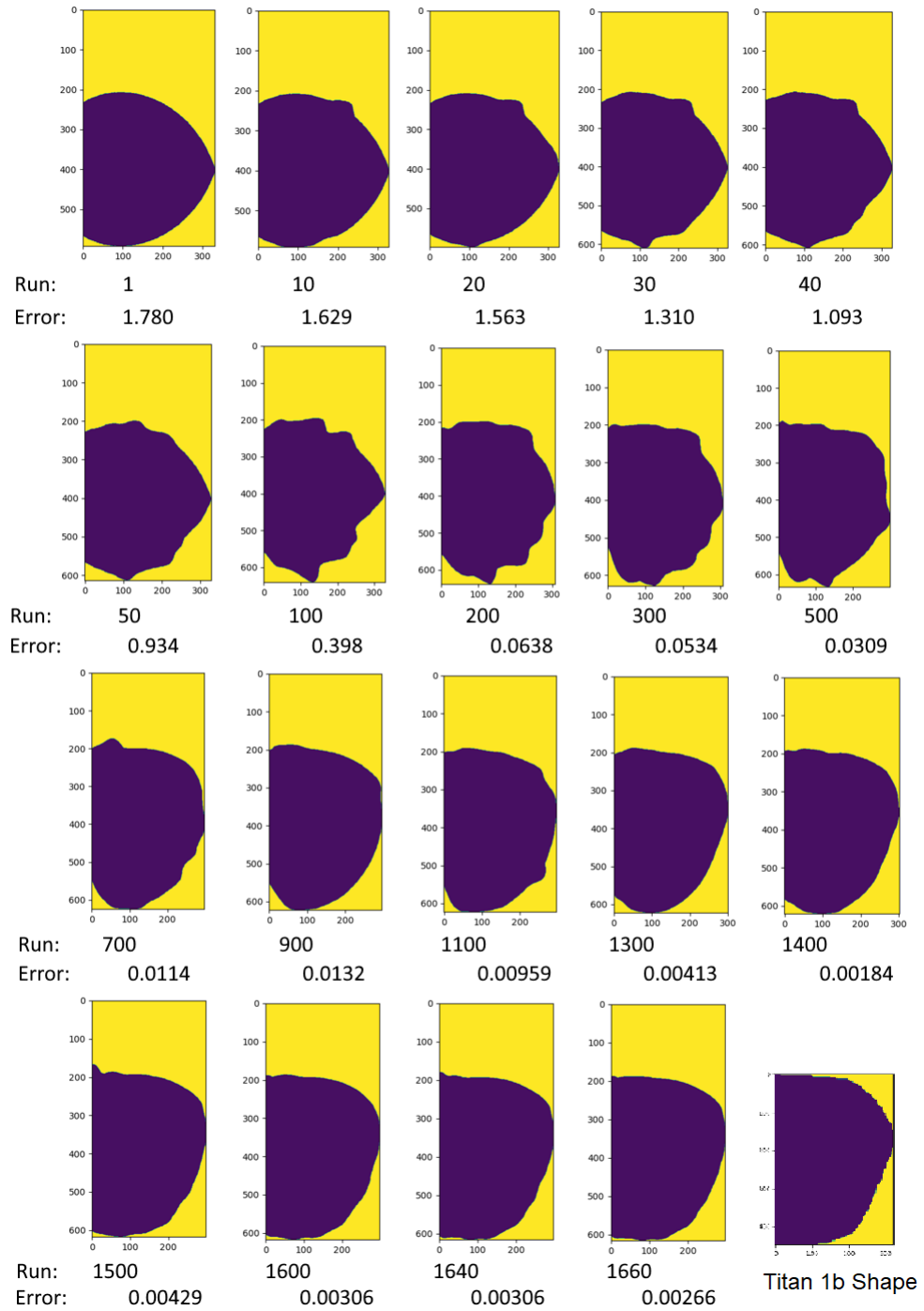


Figure 30: Polygonal line shapes of the runs of the final simulation

The given error values in figure 30 are the true error values  $J(\Omega)$  of the respective runs, disregarding the penalty term  $\vartheta(\tilde{P})$ . This is the main reason that of the given runs in figure 30, the Run 1400 had the smallest error. But the shape of the polygonal line got smoothed until Run 1660 and the overall error  $\tilde{J}(dists\_to\_mid)$  was smaller at Run 1660 (error = 0.004552) than at Run 1400 (error = 0.00517). Moreover, the polygonal line got less smooth in the first 100 runs, because the shape penalty term  $\vartheta(\tilde{P})$  did not play a big role as long as the true error  $J(\Omega)$  was relatively large.

Finally, we show the difference of the  $\log(I_s(q))$ -values to the  $\log(I_{s,meas}(q))$ -values of some runs:

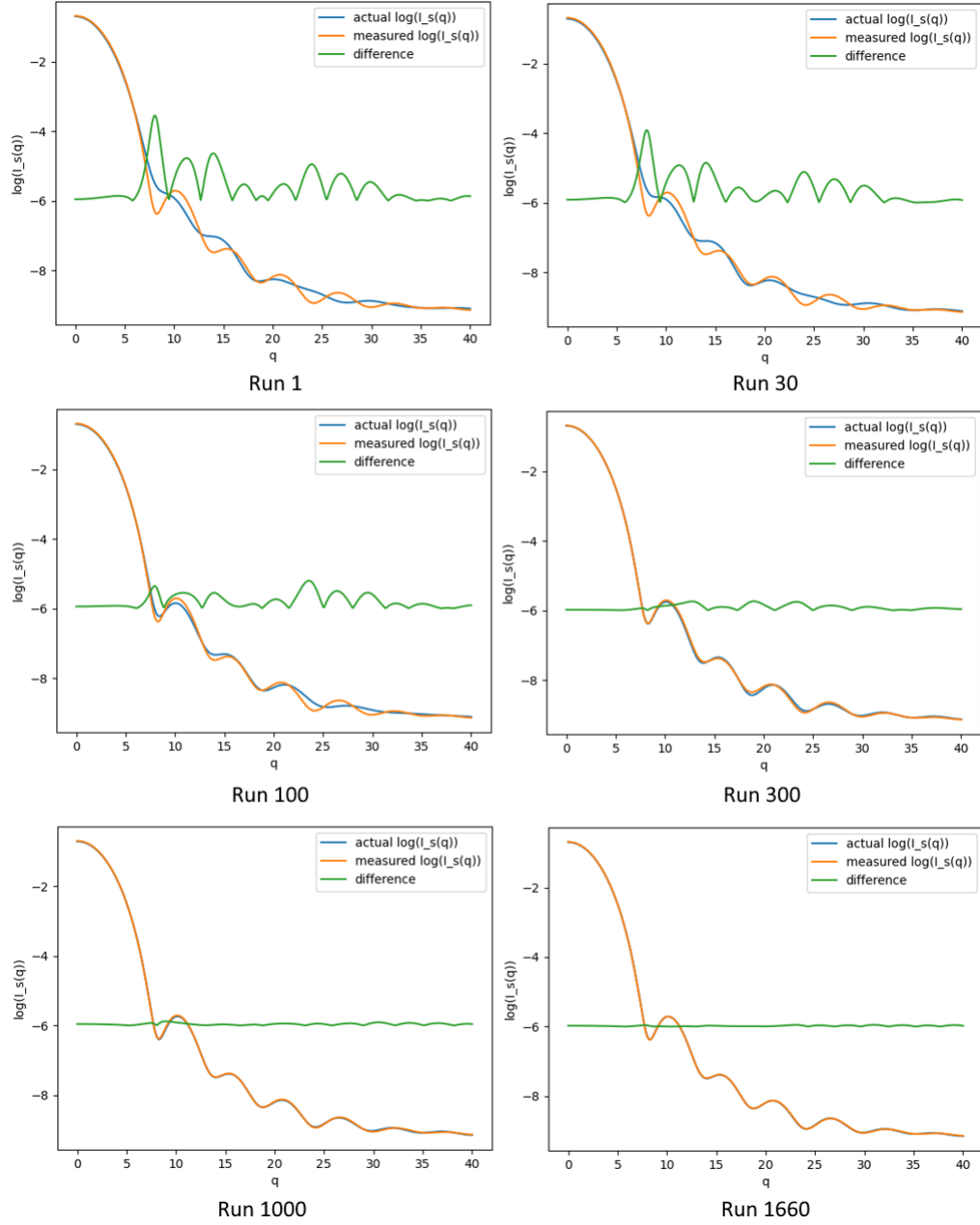


Figure 31:  $\log(I_s(q))$  difference to the measurement data of some runs

In figure 31, the green lines indicate the absolute difference between the actual  $\log(I_{s,i}(q))$  values of Run  $i$  and the measured  $\log(I_{s,meas}(q))$  values. To



show the difference in the same chart, the values  $\text{diff}_i(q)$  of the green lines at Run  $i$  are calculated as follows:

$$\text{diff}_i(q) = 4|\log(I_{s,i}(q)) - \log(I_{s,meas}(q))| - 6.$$

We can see that the algorithm successfully finds a shape so that the given  $I_{s,meas}(q)$  values are reasonably well approximated by the calculated  $I_s(q)$  values.

## 5 Conclusion

In this thesis, we:

- Derive formulas that give information about how the intensity data change when we alter the shape of the particle (chapter 1).
- Provide and describe 2 different types of algorithms that reconstruct the particle shape given some measured intensity data so that the calculated intensity data of the reconstructed shape gets as close as possible to the measured intensity data. We call these 2 reconstruction algorithms the “mathematical ansatz” (chapter 3, with the help of algorithms described in chapter 2) and the “random search ansatz” (chapter 4, with the help of algorithms described in chapter 2).

We consider only particles with constant electron density

$$\rho(\mathbf{x}) = \begin{cases} c_\rho & \mathbf{x} \in \Omega, \\ 0 & \text{else.} \end{cases}$$

Furthermore, these are the restrictions we make on  $\Omega$ :

- $\Omega$  is an open bounded set so that  $\partial\Omega$  is smooth enough so that the Gauss’ Theorem holds (stated at the beginning of section 1.5). Moreover, we state a rather technical (and practically not so important) restriction at the requirements of Theorem 1.15.
- For the reconstruction algorithms, we furthermore restrict  $\Omega$  to be a body of rotation and to be simply connected. The reason why this is necessary is given in section 3.1.
- Depending on the mesh size, the restriction so that every grid point can be classified to be inside or outside  $\Omega$  is given at the end of section 2.2.2. For internal calculations, further restrictions are the conditions (40) and (41).
- For the algorithm described in chapter 3,  $\Omega$  is restricted to be a 2d polygonal line that is rotated around the  $z$ -axis. For the algorithm described in chapter 4, further restrictions are given in section 4.1.

In the first chapter, we explain SAXS, we mention previous work on the topic and we state the intensity data SAXS would provide us (stated by Glatter and Kratky [6]):

$$I_s(q) = \int_{\mathbb{R}^3} \gamma(\mathbf{z}) \operatorname{sinc}(q\|\mathbf{z}\|) d\mathbf{z},$$

$$\gamma(\mathbf{z}) = \int_{\mathbb{R}^3} \rho(\mathbf{x}) \rho(\mathbf{x} - \mathbf{z}) d\mathbf{x}.$$

Then we explain and derive the shape derivative (Azegami, [1]) of  $I_s(q)$  in section 1.5 as

$$dI_s(q; \Omega, V) = \int_{\partial\Omega} F(\mathbf{x}, q) \langle V(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle dS(\mathbf{x}),$$

$$F(\mathbf{x}, q) = 2c_p^2 \int_{\Omega} \text{sinc}(q\|\mathbf{z} - \mathbf{x}\|) d\mathbf{z}.$$

Moreover, we define an error function between actual intensity data  $I_s(q)$  and measured intensity data  $I_{s,meas}(q)$  in section 1.6 as

$$J(\Omega) = \int_{q_{min}}^{q_{max}} |\ln(I_s(q) + \epsilon) - \ln(I_{s,meas}(q) + \epsilon)|^2 dq$$

and derive its shape derivative, which is given by

$$dJ(\Omega; V) = \int_{\partial\Omega} G(\mathbf{x}) \langle V(\mathbf{x}), \mathbf{n}(\mathbf{x}) \rangle dS(\mathbf{x}),$$

$$G(\mathbf{x}) = 2 \int_{q_{min}}^{q_{max}} F(\mathbf{x}, q) \frac{1}{I_s(q) + \epsilon} \ln \frac{I_s(q) + \epsilon}{I_{s,meas}(q) + \epsilon} dq.$$

In the second chapter:

- We describe how we find all boundary points of  $\Omega$  with respect to a rectangular grid  $G$ . (We defined boundary points w.r.t. a grid  $G$  as grid points in  $G \cap \Omega$  where a direct grid neighbor lies outside  $\Omega$ .)
- A very important aspect of this thesis is to find all boundary points of  $\Omega + \mathbf{z}$ ,  $\mathbf{z} \in \mathbb{R}$  with an effective algorithm. Such an algorithm is described in the section 2.3.1. This algorithm is central to calculate  $\gamma(\mathbf{z})$  and therefore  $I_s(q)$ .
- Calculating  $F(\mathbf{x}, q)$  in an effective way is described in section 2.5. The approximations that were used for this calculation are summarized in the end of section 2.5.5.

In the third chapter, we describe the “mathematical ansatz” for the algorithm to reconstruct the particle. The goal is to minimize the error  $J(\Omega)$ . The changes in every algorithm iteration are dependent on the shape derivative  $dJ(\Omega; V)$  of  $J(\Omega)$ . The choice of  $V$  is crucial to approach the minimum of  $J(\Omega)$ , this is described in section 3.4. Unfortunately, depending on the settings, the “mathematical ansatz” was too slow or too inaccurate to yield useful results. Therefore, we decided to try a “random search ansatz” instead.

In the fourth chapter, we describe the “random search ansatz” for the algorithm to reconstruct the particle. The goal is to minimize the error  $J(\Omega)$ . The changes in every algorithm iteration are made randomly locally on some

part of the boundary. We use the “Simulated Annealing” algorithm to decide, whether a change in one iteration is accepted or declined. Despite needing more iterations, this ansatz is a lot faster than the “mathematical ansatz” algorithm, even with exacter parameter choices. We explain parameter choices and explain why some test runs were not successful. Finally, explained at the end of the chapter, we were able to reconstruct a particle.

## References

- [1] H. Azegami. *Shape Optimization Problems*. Springer, 2020.
- [2] L. A. Feigin and D. I. Svergun. *Structure analysis by small-angle X-ray and neutron scattering*. New York: Plenum Press, 1987.
- [3] O. Forster. *Analysis 3*. Springer, 2012.
- [4] Engeln-Müllges G., Niederdrenk K., and Wodicka R. *Numerik-Algorithmen, 10. Auflage*. Springer, 2011.
- [5] O. Glatter. *Scattering methods and their application in Colloid and Interface Science*. Elsevier, 2018.
- [6] O. Glatter and O. Kratky. *Small angle x-ray scattering*. Academic Press, 1982.
- [7] A. Guinier and G. Fournet. *Small-angle scattering of X-rays*. Wiley, New York, 1955.
- [8] Press W. H. et al. *Numerical recipes in C: the art of scientific computing*. Cambridge University Press, 1992.
- [9] Atkinson K. and Han W. *Spherical Harmonics and Approximations on the Unit Sphere: An Introduction*. Springer, 2012.
- [10] van Laarhoven J. M. *Simulated Annealing: Theory and Applications*. Springer, 1987.
- [11] Abramowitz M. and Stegun I. A. *Handbook of Mathematical Functions, 10th printing*. Dover Publications, 1972.
- [12] Deuffhard P. and Hohmann A. *Numerische Mathematik 1: Eine algorithmisch orientierte Einführung (De Gruyter Studium, Band 1)*. De Gruyter, 2018.
- [13] Guldin P. *De centro gravitatis trium specierum quantitatis continuæ*. Viennæ : Formis Gregorii Gelbhaar Typographi Caesarei, 1640.
- [14] J. Sokolowski and J.-P. Zolesio. *Introduction to Shape Optimization*. Springer, 1992.
- [15] D. I. Svergun. *Small Angle X-Ray and Neutron Scattering from Solutions of Biological Macromolecules*. Oxford University Press, 2013.
- [16] D. I. Svergun. “Small-angle X-ray and neutron scattering as a tool for structural systems biology”. In: *Biological chemistry* (2010).
- [17] Gonen T. and Nannenga B. L. *cryoEM*. Springer, 2021.