universität wien

DIPLOMARBEIT

Titel der Diplomarbeit

A Sparse Grid Stochastic Collocation FEM for Uncertainty Quantification in a Nanowire Senor Model

Verfasser

Claus Aichinger

angestrebter akademischer Grad

Magister der Naturwissenschaften (Mag. rer. nat.)

Wien, im April 2013

Studienkennzahl lt. Studienblatt:	A 405
Studienrichtung lt. Studienblatt:	Diplomstudium Mathematik
Betreuer:	Priv.-Doz. DI Dr. Clemens Heitzinger

Acknowledgements

First of all I want to thank my adviser Clemens Heitzinger for giving me the opportunity to write my thesis with the support of his WWTF (Wiener Wissenschafts-, Forschungs- und Technologiefonds) project *Mathematics and nanosensors*. I am very grateful for the discussions we had and all the insight and valuable experiences I could gather in course of my work in this highly interesting field of research. Accordingly, I want to thank all others, especially those at the AIT (Austrian Institute of Technology), I had the pleasure to work with, but cannot be mentioned here by name. Furthermore owe thanks to my parents for their patience and their support, that paved the way for my studies. Additionally I wish to address my girlfriend and say *'Thank you!'* for the encouragement she gave me.

Please note that the computational results presented have been achieved (in part) using the VSC (Vienna Scientific Cluster).

Abstract

In the present thesis, we are concerned with the modelling and simulation of nanowire field-effect bio- and gas sensors, with special focus on non-intrusive methods for uncertainty quantification (UQ).
So far most of the effort regarding the mathematical understanding of nanowire sensors was devoted to deterministic models and this branch of research is still actively pursued. However, due to the structure of such a sensor, random effects play an important role. By incorporating them, one is naturally led to stochastic formulations that need to be assessed in a quantitative fashion. To this end, we apply different methods for uncertainty quantification to help gain insight into the (physical) behaviour the system under the influence of stochastic drivers. More precisely, we investigate a semi-linear second order elliptic PDE with random input data.

The work is organised as follows:

- **Chapter 1** gives a rather general introduction to nanowire sensors, their structure and functional principle as well as to the stochastic effects we need to incorporate.

- **Chapter 2** presents the Poisson-Boltzmann equation as the proper model to capture the physical situation at hand. We then turn to the stochastic formulation, discuss methods for uncertainty quantification and identify the calculation of deterministic solutions as well as high-dimensional integrals as our main task.

- **Chapter 3** starts with a brief review of the finite element method and the damped Newton method. We also comment on the issue of generating a suitable unstructured mesh to resolve our computational domain. The focus of the remaining part is then on the numerics of high-dimensional integration. After summarizing some classical results regarding quadrature formulas, we take a closer look at sparse-grid quadrature based on Smolyak's construction, for which we also briefly discuss adaptive strategies, and (quasi) Monte Carlo methods.

- **Chapter 4** sketches the program developed for the numerical simulations.
- **Chapter 5** shows the numerical results, gives a comparison between the different approaches and discusses the findings with respect to the differences between the linear and the non-linear equation.
- **Chapter 6** contains some concluding remarks and states aspects considered to be of interest in a future work.

Zusammenfassung

In der vorliegenden Arbeit beschäftigen wir uns mit der Modellierung und Simulierung von Nanodraht Feldeffekt Bio- und Gassensoren, mit besonderer Aufmerksamkeit auf nicht-eindringende Verfahren zur *Uncertainty quantification* (*UQ*) (wörtlich: Unsicherheits-Quantifizierung).
Bisher lag der Fokus im Hinblick auf das mathematische und physikalische Verständnis von Nanodrahtsensoren bei deterministischen Modellen, wobei in diesem Forschungsgebiet auch weiterhin aktiv gearbeitet wird. In Folge des Aufbaus eines solchen Sensors spielen zufällige Einflüsse jedoch eine wichtige Rolle. Durch deren Einbeziehung gelangt man auf natürliche Weise zu einer stochastischen Formulierung, die einer quantitativen Untersuchung bedarf. Zu diesem Zweck wenden wir mehrere Methoden der UQ an, die helfen, Einblicke ins das (physikalische) Verhalten des System unter Einfluss stochastischer Treiber zu gewinnen. Mathematische ausgedrückt, untersuchen wir eine semi-lineare elliptische PDE zweiter Ordnung mit stochastischen Daten.

Konkret gliedert sich die Arbeit wie folgt:

- **Kapitel 1** gibt eine allgemein Einführung zu Nanodrahtsensoren, deren Aufbau und Funktionsweise sowie den auftretenden stochastischen Effekten, die einbezogen werden müssen.

- **Kapitel 2** präsentiert die Poisson-Boltzmann Gleichung als das geeignete Modell zur Abbildung des betrachteten physikalischen Problems. Wir wenden uns dann der stochastischen Formulierung zu und diskutieren Methoden zur *Uncertainty quantification* und indentifizieren die Berechnung von deterministischen Lösungen und hochdimensionalen Integralen als unsere Hauptaufgabe.

- **Kapitel 3** beginnt mit einer kurzen Darstellung der Finite-Elemente-Methode und eines gedämpften Newton-Verfahrens. Wir Befassen uns auch mit der Erzeugung einer geeigneten Triangulierung zur Auflösung des physikalischen Aufbaus. Der Schwerpunkt des nachfolgenden Teils liegt dann auf der

numerischen Berechnung hochdimensionaler Integrale. Nach einer Zusammenfassung klassischer Resultate ber Quadraturformeln, setzen wir uns genauer mit dünne-Gitter-Quadraturverfahren, die auf der Smolyak-Konstruktion basieren, auseinander, wo wir auch kurz adaptive Strategien diskutieren. (Quasi)-Monte Carlo Methoden werden ebenfalls vorgestellt.

- **Chapter 4** skizziert das für die numerischen Simulationen entwickelte Programm.

- **Kapitel 5** zeigt die numerischen Resulate, einen Vergleich der verschiedenen Methoden und diskutiert die Ergebnisse im Hinblick auf die Unterschiede zwischen der linearen und der nichtlinearen Gleichung.

- **Kapitel 6** beinhaltet abschließende Bemerkungen und nennt Aspekte, die in zukünftigen Arbeiten untersuchenswert erscheinen.

Contents

Chapter 1

General Introduction

In this chapter we briefly outline the working principle of nanowire-based field-effect bio- and gas-sensors and discuss the kind of uncertainties that arise when modelling such a real-world device. For a book chapter discussing state of the art quantitative approaches for the deterministic problem, see for example [6].

Nanowire Sensors
The core element of a nanowire sensor is a physical transducer combined with a recognition layer. The choice of the transducer determines the measuring principle and the layer, comprising receptors such as enzymes or binding proteins like antibodies, is responsible for the required sensitivity and selectivity. Being in close contact with the transducer, one aims to measure the physical or chemical change that occurs as a result of (bio)specific interactions between the targeted substance and the recognition layer. Optical devices for example try to exploit interferometric effects occurring, when targets bind to receptors (reflectometric interference spectroscopy).

- In the case of nanowire gas sensors, the nanowire is directly exposed to the targeted substance. Occurring surface reactions result in a charge transfer between gas molecules and the nanowire, thereby increasing or decreasing the number of free carriers in the nanowire. This change in the conductance can be measured and allows to draw conclusions regarding the type of gas molecules that are present. However, due to the absence of a selective recognition layer, the identification of a characteristic sensor behaviour for a certain gas is very challenging (it is difficult to distinguish between gases causing the same type of interaction). Since the physical properties of gas sensors are not yet well understood, we investigate nanowire biosensors as described in below. Note however, that the methodology developed in this work is applicable in both cases.

- The nanowire biosensor considered here, a so called *bioFET*, resembles a field-effect transistor by making use of the field effect due to the partial charges of target molecules such as DNA strands. A sketch of such a device is shown in Figure 1.1. We see that the gate structure is replaced by a bio-functionalized layer of immobilized receptor molecules such as ssDNA strands. When complementary target molecules hybridize to the receptors, the charge distribution near the surface changes, leading to an increase of conductance and thereby modulating the current through the transducer, that is a semiconductor. Due to the use of specific receptor molecules, biosensors are by design highly selective and, in contrast to conventional detection methods, do not require labeling of target molecules. This is their main advantage.
  See for example [9], [35] or [32] for experimental set-ups and findings. For a theoretical treatment of nanosensors see for example [30], [3], [5] and [17].

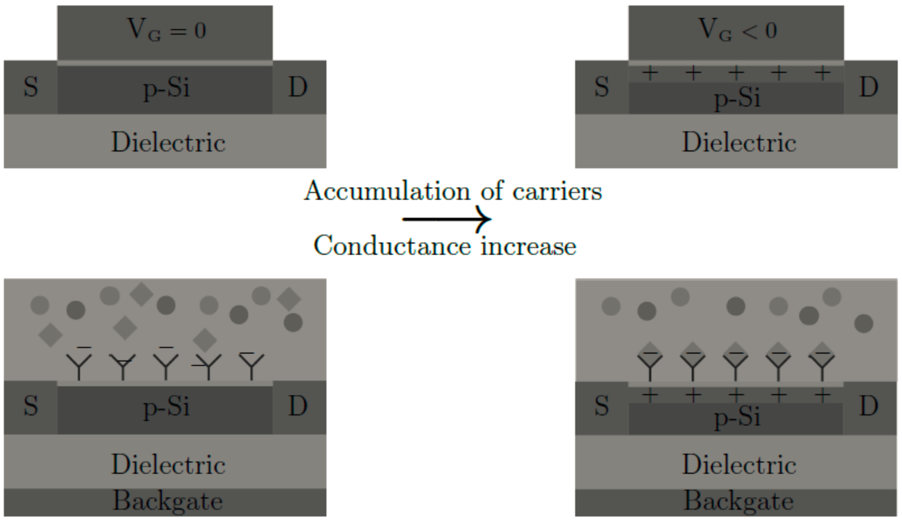

Figure 1.1: Top: a convential FET, bottom: a nanowire based sensor. We see that, as negative charges assemble at the gate contact, charge carriers accumulate near the surface increasing the conductivity. A current, related to the potential difference between the source (S) and the drain (D) contact as well as the transducers conductivity, can then be measured and allows to draw conclusions regarding the presence of target molecules. Figure taken from [6].

Uncertainty Quantification

In many applications, we observe the need for including uncertainties in mathematical models in order to quantify their effect on given outputs of interest. Such uncertainties my enter a model for different reasons. On one hand side, it might be our inability to exactly characterize all input parameters and on the other hand side, uncertainties might as well be an intrinsic property of the situation we model, as is the case here.

In a BioFET, several input parameters are indeed subject to natural fluctuations. This is easily seen when considering the situation when it is applied, since then, a lot of unknowns, whose determination is part of the job, enter the game:

- How many and what kind of target molecules are there?
- How do they distribute over the sensor surface?
- What about the ionic concentration in or a possible contamination of the analyte or other parts used in the sensor?

It is clear that such fluctuations give rise to fluctuations in the electrostatic potential and hence in the current through the transducer, the measured variable, from which the experimenter has to draw his conclusions regarding the substance under investigation. By means of random variables, we can incorporate such effects in a rigorous fashion, leading to an elliptic PDE with random input data, where we try to estimate mean and variance of the solution. In order to address this problem, we consider so called *non-intrusive* methods, i.e., methods that build on solvers for the deterministic problem. We will develop such a solver using the finite element method and are then able to generate an ensemble of solutions from which statistical information can be extracted. Usually, this process is closely related to integration and we therefore investigate the following two and a half approaches:

1. *(Quasi) Monte Carlo Sampling ((q)MCS)*: the method of choice for high-dimensional problems. We simply forward propagate the input randomness to obtain the statistical parameters. In its naive form, we mention that MC methods suffer from low convergence rates and only statistical error estimates. Therefore we also investigate the usefulness of quasi Monte Carlo methods, being the half approach.

2. *Stochastic Collocation (SC)*: based on the *Smolyak algorithm* for integration and interpolation, we construct *sparse grids* of collocation points, to efficiently exploit possible regularity of the solution and achieve better convergence rates.

For a discussion related to different approaches see for example [39].

Chapter 2

Mathematical Model of Nanowire Field-Effect Sensors

In this chapter we formulate the model equations describing the situation depicted in Figure 2.1. A doped silicon nanowire serving as the transducer (T) is embedded in a rather thick layer of silicon dioxide, the substrate (S), whose boundary layer is bio-functionalised. The substance to be investigated, the analyte, is an electrolyte (E) containing the target molecules is put/dripped onto of the sensor.

2.1 Formulation and Motivation

For reasons that will be explained in more detail later on, see Section 2.1.1, we are preliminarily interested in calculating the electric potential u in the cross section shown below. We assume the device to be symmetrical about the y-axis and having the following dimensions:

$$
\begin{array}{ll}
S_1 = (x_{S_1}, y_{S_1}) = (108, 0)\text{ nm}, & T_1 = (x_{T_1}, y_{T_1}) = (25, 145)\text{ nm}, \\
S_2 = (x_{S_2}, y_{S_2}) = (108, 145)\text{ nm}, & T_2 = (x_{T_2}, y_{T_2}) = (25, 195)\text{ nm}, \\
S_3 = (x_{S_3}, y_{S_3}) = (58, 145)\text{ nm}, & E_1 = (x_{E_1}, y_{E_1}) = (108, 253)\text{ nm}. \\
S_4 = (x_{S_4}, y_{S_4}) = (58, 203)\text{ nm}, &
\end{array}
$$

Since in an ionic system electrostatic interactions are screened and therefore limited in range, we only consider target molecules that are bound to a receptor, i.e., directly attached to the boundary layer, for we expect them to have the greatest influence on the transducer. For brevity, we refer to such pairs simply as molecules. Since we focus on the stochastic aspect, we avoid scaling problems by not resolving the true (and possibly unknown) structure of a molecule and instead

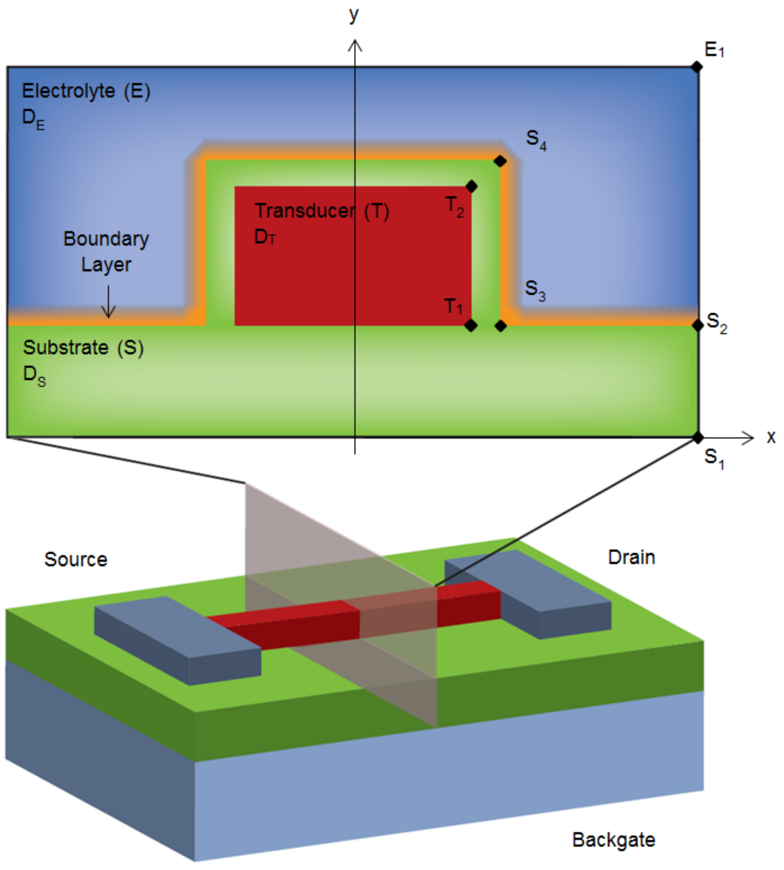

Figure 2.1: Schematic diagram of the BioFET investigated here. Figure taken from [4] (and modified).

consider a simplified geometric form as shown in Figure 2.2. In Section 2.1.2 we elaborate further on the physical properties of the molecules and their distribution. For now it is clear, that given radius r_M^i, length l_M^i, angle φ_M^i and position x_M^i, each molecule is uniquely described. As the molecules are negatively charged, the repulsive electrostatic forces cause non-intersecting configurations. Defining

- $D_M := \bigcup_{i=1}^n D_M^i$,
- $D_T := [-x_{T_1}, x_{T_1}] \times [y_{T_1}, y_{T_2}]$,
- $D_S := [-x_{S_1}, x_{S_1}] \times [y_{S_1}, y_{S_2}] \cup [-x_{S_3}, x_{S_3}] \times [y_{S_3}, y_{S_4}] \backslash D_T$,
- $D_E := [-x_{E_1}, x_{E_1}] \times (y_{S_4}, y_{S_4}] \cup [-x_{E_1}, -x_{S_3}] \times (y_{S_3}, y_{S_4}] \cup [x_{S_3}, x_{E_1}] \times (y_{S_3}, y_{S_4}]$,

we obtain a decomposition of the cross section, $D := D_M \cup D_T \cup D_S \cup D_E$, into non-intersecting domains.

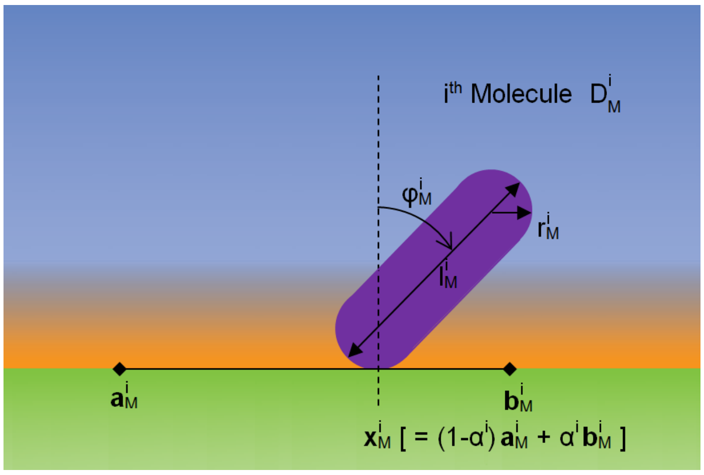

Figure 2.2: Schematic diagram of a molecule attached to the surface.

The electrostatic interactions between molecules in an ionic solution can be modelled using the *Poisson-Boltzmann equation (PBE)*, a *second order semi-linear elliptic PDE* (the difference to the ordinary *Poisson* equation lies in the additional term ρ_{free}, which is defined below):

$$-\nabla\cdot(A\nabla u) = \rho_{\text{fixed}} + \rho_{\text{free}} \qquad \text{in } D, \tag{2.1a}$$

$$u = u_D \qquad \text{on } \partial D_D, \tag{2.1b}$$

$$\nu\cdot\nabla u = u_N \qquad \text{on } \partial D_N. \tag{2.1c}$$

u_D and u_N are the Dirichlet and Neumann boundary conditions respectively, where ∂D_N is the left and right hand side boundary and ∂D_D the lower and upper boundary of D (such that $\partial D = \partial D_D \cup \partial D_N$ and $D_D \cap \partial D_N = \emptyset$) and ν the respective outward pointing unit vector. On the left hand side we have the permittivity A, describing how the electric field, given by $E = -\nabla u$, with u the electrostatic potential, is affected by a dielectric medium and vice versa. On the right hand side, we find the charge density, already written as a sum over the fixed and the free charge density. The fixed charge density ρ_{fixed} includes the contribution from immobile charges, such as molecules, the surface charge at the boundary between the substrate and the electrolyte and additionally the dopants in the transducer, see (2.10) and the comments at (2.5). The ions in the electrolyte enter via the Boltzmann-term, ρ_{free}, which is given by

$$\rho_{\text{free}}(x,u) := \sum_{j\in S} z_j \kappa_j \exp\left(-\frac{z_j q\,(u(x)-\varphi_F)}{k_B T}\right).$$

Here S is the set of charge species such as anions and cations in the ionic solution, $z_j \in \mathbb{Z}$ is the valence of the respective species, $\kappa_j(x) \geq 0$ is the corresponding bulk concentration, q the elementary charge, k_B the Boltzmann constant, T the temperature and φ_F the Fermi level. In our case, the above expression simplifies due to the fact that we only consider two different kind of species, allowing to set $S := \{-1, +1\}$, $z_j := j$ and $\kappa_{-1}(x) = \kappa_1(x) =: \kappa(x)$. Accordingly, we obtain

$$\rho_{\text{free}}(x,u) = -2\kappa(x) \sinh \frac{q\,(u(x) - \varphi_F))}{k_B T}. \tag{2.2}$$

As we see, without ρ_{free}, equation (2.1) is the Poisson equation, whose solution gives the electrostatic potential resulting from a given charge distribution. However, due to the free charges, this distribution is a priori not known. Indeed, one observes the following situation: depending on the sign of their charge, ions preferentially reside in regions with high or low electric potential, i.e., closer or farther apart from the molecules, and thereby in turn influence the electrostatic potential, explaining why we have to consider an non-linear equation. Since in such an ionic system electrostatic interactions are screened, i.e., molecules having already attracted ions become ‘invisible’ for the more distant ones, electrostatic effects due to the solute are limited to a small distance. Equation (2.1) incorporates these observations and the electrostatic potential obtained by solving (2.1) corresponds to the steady-state distribution. A brief discussion of the derivation of the PBE can be found for example in [12]. Note further that the main assumption in this continuum model is that the charges are points.

Calculating the linear term of the Taylor expansion around a suitable expansion point u_0 yields

$$\rho_{free}(x) = \alpha(x) - \gamma(x)u(x) + O\left((u(x) - u_0)^2\right), \tag{2.3a}$$

$$\alpha(x) := 2\kappa(x) \sinh \frac{q\,(u_0 - \varphi_F)}{k_B T} + \frac{2qu_0\kappa(x)}{k_B T} \cosh \frac{q\,(u_0 - \varphi_F)}{k_B T}, \tag{2.3b}$$

$$\gamma(x) := \frac{2q\kappa(x)}{k_B T} \cosh \frac{q\,(u_0 - \varphi_F)}{k_B T}, \tag{2.3c}$$

and we then obtain the so called *linearised Poisson-Boltzmann equation (LPBE)* as

$$-\nabla \cdot (A\nabla u) + \gamma u = \rho_{\text{fixed}} + \alpha \qquad \text{in } D, \tag{2.4a}$$

$$u = u_D \qquad \text{on } \partial D_D, \tag{2.4b}$$

$$\nu \cdot \nabla u = u_N \qquad \text{on } \partial D_N. \tag{2.4c}$$

Clearly, a more advanced model to capture the situation in the transducer would be the *drift-diffusion-Poisson system (DDP)* [6]:

$$\begin{aligned}
-\nabla\cdot(A\nabla u) &= q(C+p-n), &\quad (2.5a)\\
\nabla\cdot J_n &= R, &\quad (2.5b)\\
\nabla\cdot J_p &= -R, &\quad (2.5c)\\
J_n &= D_n\nabla n-\mu_n n\nabla u, &\quad (2.5d)\\
J_p &= -D_p\nabla p-\mu_p p\nabla u, &\quad (2.5e)
\end{aligned}$$

where n_i is the intrinsic charge concentration, C is the doping concentration, n and p are the concentration of the free carriers, J_n and J_p are the current densities and D_n and D_p the diffusion coefficients, μ_n and μ_p the mobilities of the electrons and holes respectively and R is the recombination rate, taken as the Shockley-Read-Hall term

$$R=\frac{np-n_i}{\tau_p(n+n_i)+\tau_n(p+n_i)}.$$

Here, τ_n and τ_p denote the lifetime of electrons and holes respectively. However, assuming thermal equilibrium, one can obtain, taking $\kappa=n_i=C+p-n$ and $\rho_{\text{fixed}}=qC$ for $x\in D_T$, the PBE already introduced above.

2.1.1 The Graded-Channel Approximation

Starting from the DDP system, we will now derive a first approximation of the current I, given as the integral over the current densities J_n and J_p, in the transducer. Neglecting diffusion and assuming that the electric field is constant with respect to the z-axis, i.e., given by $E=(u_\text{s}-u_\text{d})/z_{sd}$, with the potential difference $u_\text{s}-u_\text{d}$ between the source and the gate contact and z_{sd} the respective distance, we get, since $E=-\nabla u$,

$$J_n^{\text{drift}}+J_n^{\text{drift}}=-q\mu_n n\nabla u-q\mu_p p\nabla u=qE(\mu_n n+\mu_p p). \quad (2.6)$$

Approximating the carrier charge concentration by the Boltzmann distribution, we have

$$\begin{aligned}
p &= n_i\exp\left(\frac{q(u-\phi_F)}{k_BT}\right), &\quad (2.7a)\\
n &= n_i\exp\left(-\frac{q(u-\phi_F)}{k_BT}\right). &\quad (2.7b)
\end{aligned}$$

In total, we arrive at

$$I=\int_{D_T}J_n^{\text{drift}}+J_p^{\text{drift}}\,\mathrm{d}x\,\mathrm{d}y=qE\int_{D_T}(\mu_n n(x,y)+\mu_p p(x,y))\,\mathrm{d}x\,\mathrm{d}y, \quad (2.8)$$

where we only rely on a solution u of the PBE on a cross section of the sensor. Although simpler than a self-consistent transport model, this first approximation is sufficient for our purposes.

2.1.2 Properties and Distribution of Molecules

In this subsection we are concerned with the physical properties of the molecules and their distribution.

Physical Properties
We consider so called *oligomers* or *n-mers*, i.e., DNA strands comprising n base pairs. The structure, double helices coiled around the same axis, has typically a radius of $r_M = 1$nm, an increase per base of about 0.34nm and a charge per base of $-q$, where q is the (positive) elementary charge. Possible targets are *PNA*, *ssDNA (single stranded)* or *dsDNA (double stranded)*, which – for our purposes – only differ in their charge and otherwise share the same properties. PNA is uncharged and dsDNA has twice the charge of ssDNA. As an example, let $n = 20$, an ssDNA strand is then of length $l_M = 6.8$nm and total charge $q_M = -20$q. To obtain the charge density, one divides by the molecules A_M in the 2D case, see Figure 2.2, or by its volume in the 3D case. In the numerical simulations we use polygons to approximate the shape of a molecule.

Distribution
The simplest counting process in continuous time is the widely used *Poisson process*, that finds its applications in a variety of different fields. It models well phenomena, where one counts uniformly distributed events over a certain unit of time or space and it is thus suitable to adequately describe the spread of molecules in our sensor. For sake of brevity, we restrict ourselves to the basic definitions, for a detailed account on stochastic processes see the vast literature on this subject, for example [31].

For we will need it, recall that the *Poisson distribution* is given by $P_\lambda(n) = e^{\lambda}\frac{\lambda^n}{n!}$, with $n \in \mathbb{N}_0$ and parameter $\lambda > 0$. Note that λ equals both the mean and the variance of a Poisson distributed random variable. A stochastic process $(N_{\lambda,t})_{t\geq 0}$ with values in $\mathbb{N}_0$ is called *Poisson process* with parameter $\lambda > 0$, if the following conditions are met:

1. $N_{\lambda,0} = 0$, i.e., the counting of events begins at time $t = 0$.

2. $N_{\lambda,t+s} - N_{\lambda,t} \sim P_{\lambda s}\ \forall s,t > 0$, i.e., the increments on any interval of length s are Poisson-distributed with parameter λs.

3. For $n \in \mathbb{N}$ and given times $0 < t_1 < \cdots < t_n$, the family of increments $\left(N_{\lambda,t_i} - N_{\lambda,t_{i-1}}\right)_{2 \le i \le n}$, is *independent*.

The time between events is exponentially distributed, and the parameter λ is sometimes referred to as *intensity*, since we expect λ increments per time unit. Due to the third condition, the increments are *stationary*, i.e., the distribution of the number of events occurring in an arbitrary time interval depends only on the length of this interval. Returning to the our scenario, note that although the total number of molecules is given by a Poisson distribution, the occurrences are nevertheless distributed uniformly on any given interval. Having specified the position of the molecules, it now remains to discuss the distribution of angles. As a result of chemical reactions between the substrate and the electrolyte, the formation of a surface charge is observed. It is hence reasonable to imagine that different angles are assumed with different probabilities, as is indeed the case. We follow the approach of [16], in order to associate orientations (with respect to the surface) with probabilities. To this end, we consider the free energy E_i of a given configuration i, i.e., with angle φ_i, and assign a probability according to the Boltzmann distribution

$$p_i := \frac{\exp(-E_i/(k_B T))}{\sum_i \exp(-E_i/(k_B T))}.$$

We use the data in [16]. In order to obtain a reasonable probability distribution, it is beneficial to first fit polynomials (in a least-squares sense) of total degree of four to the energy data. Polynomials of lower degree are cannot reproduce all relevant features whereas polynomials of higher degree lead to over-fitting. Figure 2.3 shows two examples.

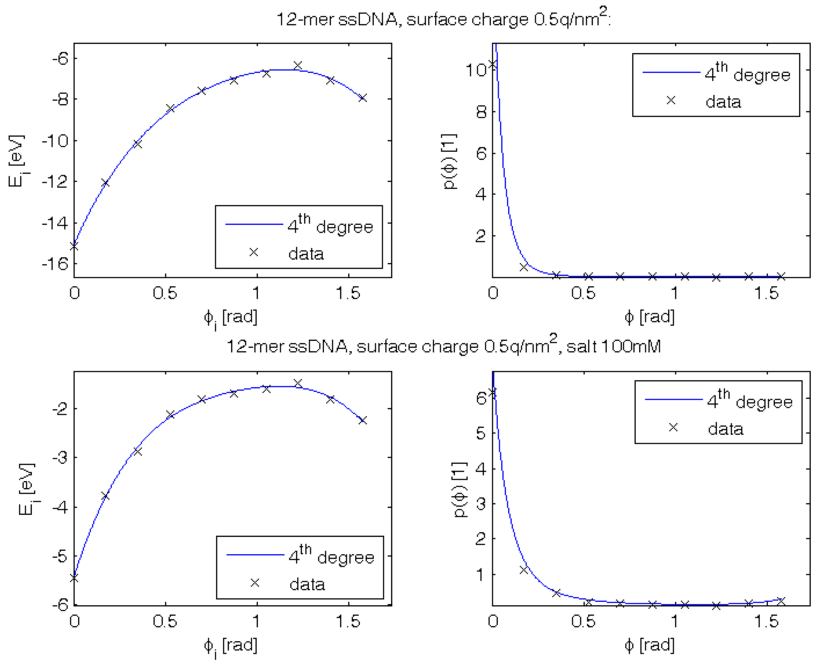

Figure 2.3: Energy levels and obtained probability distribution for two different scenarios are depicted. The energies are the electrostatic free energies of the molecules. Data taken from [16].

2.2 Variables and Units

For a better overview, we summarize all relevant physical units and numerical parameters in the Table 2.1 below and review the coefficients for the PBE (the situation in linearized case is similar). The given values are used in the simulations (if not stated otherwise).

- On the left hand side of (2.1), we have the position dependent permittivity A, given by

$$A(x) := \begin{cases} \varepsilon_0 \varepsilon_T & \text{if } x \in D_T, \\ \varepsilon_0 \varepsilon_S & \text{if } x \in D_S, \\ \varepsilon_0 \varepsilon_M & \text{if } x \in D_M, \\ \varepsilon_0 \varepsilon_E & \text{if } x \in D_E. \end{cases} \tag{2.9}$$

- On the left hand side, we find the two charge concentrations. Firstly, ρ_{fixed}, related to the fixed charges, namely

$$\rho_{\text{fixed}}(x) := \begin{cases} \rho_M & \text{if } x \in D_M, \\ \rho_C & \text{if } x \in \partial D_S \cap \partial D_E, \\ \rho_T = q \cdot C & \text{if } x \in D_T, \\ 0 & \text{otherwise.} \end{cases} \tag{2.10}$$

Recall that $\rho_M = q_M/A_M = -qn/(r_M^2\pi + 2(l_M - 2r_M))$, where n is the number of base pairs, $r_M = 1$ nm the radius and $l_M = n \cdot 0.34\text{nm}$ nm the length of the molecule.

- Secondly, ρ_{free}, incorporating the contribution of the free charges, according to the potential u and the respective bulk concentration κ, the latter given as

$$\kappa(x) := \begin{cases} \kappa_T = n_i & \text{if } x \in D_T, \\ \kappa_E & \text{if } x \in D_E, \\ 0 & \text{otherwise.} \end{cases} \tag{2.11}$$

Meaning	Variable	(Value·)Unit
Temperature	T	300 K
Elementary charge	q	$1.60218 \cdot 10^{-19}$ C
Electron Volt	eV = q·V	$1.60218 \cdot 10^{-19}$ J
Boltzmann constant	k_B	$1.38065 \cdot 10^{-23}$ J·K^{-1}
Mole per litre	M = mol/l	1000 mol m^{-3}
Vacuum permittivity	ε_0	8.85419·10^{-12} A·s·V^{-1}·m^{-1}
Relative permittivity of electrolyte (water)	ε_E	80.1
Relative permittivity of substrate (silicon dioxide)	ε_S	3.9
Relative permittivity of transducer (silicon)	ε_T	11.9
Relative permittivity of molecules (DNA)	ε_M	4
Surface charge density	ρ_C	0.5 q·nm^{-2}
Ion concentration	κ_E	10 mM
Doping concentration	C	1·10^{-5} nm^{-3}
Intrinsic concentration	n_i	1.059·10$^-$12 nm^{-3}
Electron mobility	μ_n	1500 cm^2· V^{-1}·s^{-1}
Hole mobility	μ_p	450 cm^2· V^{-1}·s^{-1}
Distance between source and drain contact	z_{sd}	1000 nm
Potential difference between source and drain contact	$u_s - u_s$	3 V

Table 2.1: Variables and units

2.3 Uncertainty Quantification

Introduction

Numerical simulations are devised to predict the behaviour of natural or engineered systems. In the field of numerical analysis, extensive efforts are devoted to the development of numerical algorithms, to ensure that numerical errors inevitably occurring in any calculation are understood and controlled. Much less attention has been paid to the question of estimating the impact of possible errors or uncertainties in the input data. By investigating the effect of such errors *uncertainty quantification (UQ)* aims to bridge this gap and provide more reliable predictions for practical applications.

In our context, we wish to incorporate the uncertainty introduced by both the positions x_M^i as well as the angles φ_M^i of the molecules, which are random parameters whose stochastic influence we denote by ω. Now let $D \subset \mathbb{R}^d$ be a bounded physical domain with boundary ∂D and let $(\Omega, \mathscr{A}, P)$ be a complete probability space. Here Ω is the event space, the set of possible outcomes ω, $\mathscr{A} \subset 2^\Omega$ is a σ-algebra of events and $P : \mathscr{A} \to [0,1]$ a probability measure. We then consider the *stochatic Poisson-Boltzmann equation (SPBE)*

$$-\nabla \cdot (A(x;\omega)\nabla u(x;\omega)) = \rho_{\text{fixed}}(x;\omega) + \rho_{\text{free}}(x,u;\omega) \qquad \text{in } D, \qquad (2.12a)$$
$$u(x;\omega) = u_D(x) \qquad \text{on } \partial D_D, \qquad (2.12b)$$
$$\nu \cdot \nabla u(x;\omega) = u_N(x) \qquad \text{on } \partial D_N, \qquad (2.12c)$$

and the problem of finding a random function $u : \bar{D} \times \Omega \to \mathbb{R}$, such that P-almost everywhere (a.e.) in Ω, or in other words, almost surely (a.s.), the above equation (2.12) is satisfied. Note that the boundary data is still deterministic. Because of the randomness in the parameters, the solution $u(x;\omega)$ now exhibits an implicit dependence on ω and we thus obtain a family of solutions, where one is in general not interested in any particular solution related to a particular choice of ω (except the case of investigating worst-case scenarios) and instead aims to characterize its statistics. Clearly, the average behaviour of the system, expressed by $\mathbb{E}[u(x)] = \bar{u}(x)$, as well as the variance $\mathbb{V}[u(x)]$ from that average are of most interest. Intuitively, one would address this problem by solving (2.12) for any possible value of ω and then weigh the result by is probability $P(\omega)$. Hence, considering the weak formulation of the problem by integrating over Ω yields

$$\int_\Omega \int_D A(x,\omega)\nabla u(x,\omega) \cdot \nabla \phi_i(x,\omega)\, \mathrm{d}x\, \mathrm{d}P(\omega) =$$
$$= \int_\Omega \int_D (\rho_{\text{fixed}}(x;\omega) + \rho_{\text{free}}(x,u;\omega))\, \phi_i(x,\omega)\, \mathrm{d}x\, \mathrm{d}P(\omega),$$

where the ϕ_i are suitable test functions. Interchanging the order of integration then gives the usual formulation encountered in the finite element method, resulting in a (non-linear) equation of the form $A(\bar{U})\bar{U} = F(\bar{U})$ which we can solve to obtain an approximation of $\bar{u}(x)$. Clearly, calculating A and F requires integration over the physical, as well as the stochastic domain, the latter being potentially high-dimensional.
In the linear case, a straightforward application of the *Lax-Milgram theorem* allows to proof existence and uniqueness of the weak solution, see for example [2]. In the nonlinear case, the situation is more involved, see for example [18]. After this introduction, we now discuss the general mathematical framework.

Mathematical Framework

Let D and $(\Omega, \mathscr{A}, P)$ be as above and be L a differential and B a boundary operator. We consider a system of PDEs

$$L(x,u;\omega) = 0 \quad x \text{ in } D \times \Omega, \tag{2.13a}$$

$$B(x,u;\omega) = 0 \quad x \text{ on } \partial D \times \Omega, \tag{2.13b}$$

where $\omega = (\omega_1, \dots, \omega_k) \in \Omega$ denotes the random input, which we assume can be parametrized by mutually independent random variables $y = (y_1, \dots, y_n)^{\mathrm{T}} \in \mathbb{R}^n$, $1 \leq n \leq k$. Let $\rho_i : \Gamma_i \to \mathbb{R}^+$ be the probability density function of the random variable $y_i(\omega)$, we can write the joint probability density of y as

$$\rho(y) = \prod_{i=1}^{n} \rho_i(y_i), \tag{2.14}$$

with support

$$\Gamma := \prod_{i=1}^{n} \Gamma_i \subset \mathbb{R}^n, \tag{2.15}$$

with images $\Gamma_i := y_i(\Omega) \subset \mathbb{R}$. We now reformulate (2.13) as

$$L(x,u;y) = 0 \quad x \text{ in } D \times \Gamma, \tag{2.16a}$$

$$B(x,u;y) = 0 \quad x \text{ on } \partial D \times \Gamma, \tag{2.16b}$$

allowing us to conduct our considerations for the finite dimensional random space Γ, in replacement of the infinite dimensional space Ω. Clearly, we wish to find a stochastic function $u = u(x,y) : \bar{D} \times \Gamma \to \mathbb{R}$ satisfying (2.13) for all $(x,y) \in \bar{D} \times \Gamma$ and therefore make the assumption that (2.13) is well-posed P-almost surely in Ω.

Karhunen-Loève Expansion

Since we have and will heavily exploit the fact, that the random input can be parametrized by a finite set of n independent random variables, we mention the *Karhunen-Loève expansion*, a widely used technique for dimension reduction and representation of random processes. In this connection, one seeks an expansion of the random input process Y_t with mean $\mu_Y(t)$ and covariance function $C(t,s) = \operatorname{cov}(Y_t, Y_s)$ as

$$Y_t(\omega) = \mu_Y(t) + \sum_{i=1}^{\infty} \sqrt{\lambda_i} \psi_i(t) Y_i(\omega),$$

where ψ_i are the orthogonal eigenfunctions with eigenvalues λ_i of the corresponding eigenvalue problem

$$\int_T C(t,s) \psi_i(s) \, \mathrm{d}s = \lambda_i \psi_i(t), \quad t \in T,$$

and $\{Y_i(\omega)\}_i$ are mutually independent uncorrelated random variables defined by

$$Y_i(\omega) = \frac{1}{\sqrt{\lambda_i}} \int_T (Y_t(\omega) - \mu_Y(t)\psi_i(t))$$

and satisfying $\mathbb{E}[Y_i] = 0$ (and $\mathbb{E}[Y_iY_j] = \delta_{ij}$, as they are independent). In any application, one relies on a truncation of the above series to obtain an approximation based on independent random variables. Its accuracy is related to the decay of the eigenvalues λ_i. From now on, assume the existence of a suitable parametrization. Further details can be found for example in [39].

Stochastic Collocation

We now seek an approximation to the exact solution of (2.16) by means of a *collocation method*. Generally speaking, collocation methods require the governing equation to be satisfied at a discrete set of nodes, the *collocation points*, in the computational domain. Hence, let $\Theta = \{y_i\}_i \subset \Gamma$ be such a finite set of prescribes nodes, we determine respective solutions $u_i = u(\cdot, y_i)$ of

$$L(x, u_i; y_i) = 0 \quad x \text{ in } D, \tag{2.17a}$$

$$B(x, u_i; y_i) = 0 \quad x \text{ on } \partial D. \tag{2.17b}$$

Clearly, this is a deterministic problem which can be solved given a corresponding solver. One then applies post-processing operations, such as interpolation and/or integration, on the ensemble $\{u_i\}_i$ to obtain an approximation and/or statistical information.

More precisely, we seek a numerical approximation $u_{h,p}$ to the exact solution u of (2.12) in the finite dimensional tensor product subspace $V_{h,p} := H_h(D) \otimes P_p(\Gamma)$, where:

- $H_h(D) \subset H_0^1(D)$ is a standard finite element space of dimension N_h, based on continuous piecewise polynomials associated with a regular triangulation with maximum mesh spacing parameter $h > 0$,

and

- $\mathscr{P}_{\mathbf{p}}(\Gamma) = L^2_\rho(\Gamma)$ denotes the space spanned by tensor product polynomials of degree at most $\mathbf{p} = (p_1, \dots, p_n)$, giving $\mathscr{P}_{\mathbf{p}}(\Gamma) = \bigotimes_{i=1}^n \mathscr{P}_{p_i}(\Gamma_i)$, with

  $$\mathscr{P}_{p_i}(\Gamma_i) = \operatorname{span}\left(y_i^k, k = 0, \dots, p_i\right), \quad i = 1, \dots, n,$$

  yielding that $\mathscr{P}_{\mathbf{p}}(\Gamma)$ is of dimension $N_p = \prod_{i=1}^n (p_i + 1)$.

Our collocation method now amounts to the following:

1. Calculate a semi-discrete approximation $u_h : \Gamma \to H_h(D)$ satisfying the deterministic weak form of (2.12)

$$\begin{aligned}&\int_D A(x;y)\nabla u_h(x;y)\cdot\nabla\phi_h(x)\,\mathrm{d}x = \qquad (2.18)\\ &= \int_D \left(\rho_{\text{fixed}}(x;y)+\rho_{\text{free}}(x,u;y)\right)\phi_h(x)\,\mathrm{d}x \quad \forall\phi_h\in H_h(D), \quad \text{for a.e. } y\in\Gamma,\end{aligned}$$

   to obtain a finite element approximation in the physical domain.

2. Collocate (2.18) on a set of prescribes nodes $y_i \in \Theta \subset \Gamma$ and construct the fully discrete approximation $u_{h,\mathbf{p}} \in \mathscr{C}^0(\Gamma; H_h(D))$ by interpolation in y in the collocated solutions,

$$u_{h,\mathbf{p}}(x;y) = \sum_i u_h(x,y_i)l_i^{\mathbf{p}}(y), \qquad (2.19)$$

   to obtain a polynomial approximation in the stochastic domain. The interpolating basis functions $l_i^{\mathbf{p}}$ can, for instance, be taken as the Lagrange polynomials.

3. Compute

$$\mathbb{E}[u] \approx \bar{u}_{h,\mathbf{p}} = \sum_i u_h(\cdot,y_i)\int_\Gamma l_i^{\mathbf{p}}(y)\rho(y)\,\mathrm{d}y, \qquad (2.20a)$$

$$\mathbb{V}[u] \approx \sum_i u_h(\cdot,y_i)^2\int_\Gamma l_i^{\mathbf{p}}(y)\rho(y)\,\mathrm{d}y - \left(\bar{u}_{h,\mathbf{p}}\right)^2, \qquad (2.20b)$$

   to obtain an approximation of the statistical parameters of interest.

A more detailed account, including error estimation for the (linear) Poisson equation with random input data, can be found in [24] and the references therein. See also [39] for a different exposition. Note that the nomenclature is not uniform, since this field of research is still under rapid development.

Summary, Observation and Outlook:

- For a given collocation point $y_i \in \Theta$, we need to calculate the corresponding solution u_h of (2.18). To this end we develop a FEM-based solver, the necessary prerequisites are presented in Chapter 3.1. Although other solution methods are equally possible, the FEM with its unstructured meshes is well suited to resolve the underlying geometry and produce results efficiently – for a lot of them are needed.

- Equation 2.20 is basically a quadrature formula and this motivates the use of quadrature points for the collocation. Indeed, since we are primarily interested in $\mathbb{E}[u]$ and $\mathbb{V}[u]$ and do not require an explicit representation of $u_{h,\mathbf{p}}$, we can leave out the second step above, i.e., (2.19). We instead sample at the nodes of a quadrature rule and estimate mean and variance directly. Chapter 3.2 is therefore denoted to numerical integration, where we present a sparse grid technique, namely the *Smolyak construction* to integrate smooth functions over high-dimensional tensor product domains. Additionally, we also consider (q)MC methods to carry out the integration and obtain reference values for comparison. Since every collocation point requires a solution of the deterministic system, it is particularly important to both have an efficient solver and an efficient integration method. In the interpolation framework, the approach is similar, the only difference being the additional post-processing step for the interpolation.

- Note that we are primarily interested in the nonlinear case whose treatment is more involved both theoretically and numerically. Although the exponential term in the charge distribution might cause problems, we avoid the error-prone choice of expansion points and the difficult question, in which regime the linear approximation is still valid. Therefore, by directly addressing the physically well founded PBE, we expect to obtain sound and realistic results.

Recall that we made the assumption that the uncertainty can by parametrized by a set of independent random variables y_i, allowing us to write Γ as a tensor product of intervals Γ_i. For this condition to be met, we devised the scenarios described in Chapters 5. The idea being that we assign each molecule an admissible region on the boundary layer, chosen in such that the molecules are expected not to interfere and this way guaranteeing independent inputs. The Poisson process is approximated by distributing the molecules uniformly in their respective region. Having n molecules, for the ith molecule we get, cf. Figure 2.2,

$$\begin{aligned} x_M^i &= (1-\alpha^i)a_M^i + \alpha^i b_M^i, \quad \alpha^i \in [0,1], \\ \varphi_M^i &= (1-\alpha^{n+i})(-\pi) + \alpha^{n+i}\pi, \quad \alpha^{n+i} \in [0,1], \end{aligned}$$

where α^j, with $1 \leq j \leq n$ is distributed uniformly and α^j, with $n+1 \leq j \leq 2n$ is distributed according to the experimental data. Clearly, parametrizing the random input is trivial and we obtain the unit hypercube $\Gamma = [0,1]^{2n}$ as stochastic domain. Further details can be found in Chapter 5.

Chapter 3

Numerical Methods

3.1 Finite-Element Method

In this section, we give a brief outline of the idea behind the finite-element method (FEM), assuming that the reader is already familiar with the basic concepts. We also comment on methods for solving non-linear systems of equations. We then have a closer look at an algorithm to create unstructured meshes for our purposes.

3.1.1 FEM in a Nutshell

Given a linear elliptic boundary value problem, we multiply with an arbitrary testfunction v and integrate over the domain to obtain the corresponding weak formulation:

$$\text{find } u \in V, \text{ such that } a(u,v) = l(v) \quad \forall v \in V. \tag{3.1}$$

Here V is the solution and testfunction space (usually a Sobolev space), $a(\cdot,\cdot)$ a bilinear functional on $V \times V$ and $l(\cdot)$ a linear functional on V. We now replace V by a finite-dimensional subspace V_h of dimension $N(h)$ and obtain the finite dimensional problem

$$\text{find } u_h \in V_h, \text{ such that } a(u_h,v_h) = l(v_h) \quad \forall v_h \in V_h, \tag{3.2}$$

the projection of the weak form of the differential equation onto this subspace. Denoting by $\phi_1, \dots, \phi_N(h)$ a basis, we can express the approximate solution u_h in terms of the basis functions

$$u_h(x) = \sum_{i=1}^{N(h)} U_i \phi_i(x), \tag{3.3}$$

with uniquely determined coefficients U_i. Clearly, since the ϕ_i form a basis, we can replace $v_h \in V_h$ in (3.2) by ϕ_i, $i = 1, \dots N(h)$, yielding

$$\text{find } (U_i)_{i=1}^{N(h)} \in \mathbb{R}^{N(h)}, \text{ such that } \sum_{i=1}^{N(h)} a(\phi_i, \phi_j) \cdot U_i = l(\phi_j) \quad \forall j = 1, \dots, N(h), \tag{3.4}$$

a system of linear equations. Introducing the shorthand notation $A_{i,j} = a(\phi_i, \phi_j)$ for $i, j = 1, \dots, N(h)$, $U = (U_1, \dots, U_{N(h)})^{\mathrm{T}}$ and $F = (l(\phi_1), \dots, l(\phi_{N(h)}))^{\mathrm{T}}$, we arrive at the linear problem

$$\text{find } U \in \mathbb{R}^{N(h)}, \text{ such that } AU = F.$$

A and F are usually called *stiffness matrix* and *load vector* respectively. Solving (3.4), (3.3) gives the approximation u_h to the exact solution u.

The subspace $V_h \subset V$ is usually chosen in such a way, that it is spanned by continuous piecewise polynomials of fixed degree and small support, which are associated with certain subdivisions of the computational domain. Polynomials are practical, in the sense, that the integrals that have to be evaluated in order to calculate A and F can be computed with little effort. The small support leads to a sparse structure, reducing the cost of solving the resulting system of equations. The process of calculating A and F is usually referred to as *assembling*. Note further, that if a is elliptic and self-adjoined, then A is both symmetric and positive definite allowing to use powerful iterative algorithms, such as the *conjugated gradient (CG)* method, to calculate the solution.

As already indicated, the starting point of every finite element method is choosing subdivision of the computational domain D. Usually, one considers a decomposition into triangles, the *finite elements*, giving a *triangulation* or *mesh*. Here we assume, that any pair of triangles intersect at a complete edge, at a vertex or not at all, such a triangulation is called *regular*. With each interior node, we then associate basis functions $\phi_i(x)$, that are equal to 1 at that node and equal to 0 at all others. In the simplest case, one chooses piecewise linear functions, which then look like a *hat* or *tent* and are illustrated in Figure 3.1. Denoting a basis function associated with interior node x_i by ϕ_i, we obviously have $\phi_i(x_j) = \delta_{ij}$, yielding the important property, that

$$u_h(x_j) = \sum_{i=1}^{N(h)} U_i \phi_i(x_j) = U_j.$$

Hence the expansion coefficients are exactly the nodal values of the approximate solution. Another immediate consequence of this choice of basis functions is, that

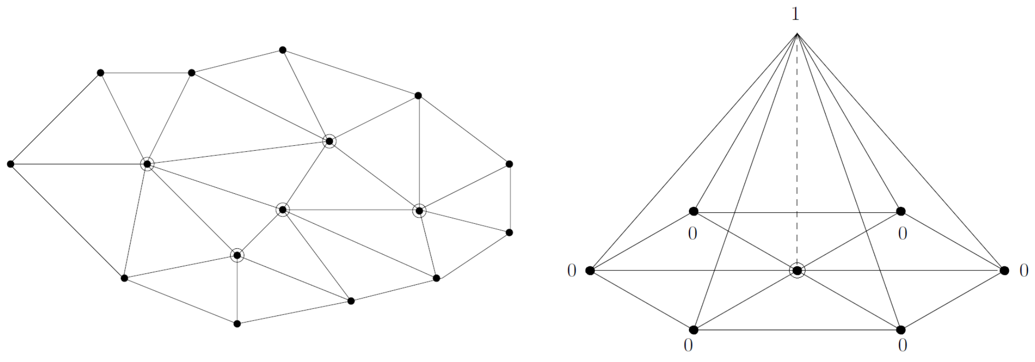

Figure 3.1: A subdivision (triangulation) of $\bar{D}$ and a typical finite element basis function ϕ. Figure taken from [34] (and modified).

A_{ij} is zero unless x_i and x_j belong to the same triangle, leading to sparse matrices. See [1] for a well commented FEM implementation in MATLAB, the issue of incorporating boundary conditions is also discussed there. However, for performance reasons, we rely on the MATLAB function *assempde*[1].

3.1.2 Solving Systems of Nonlinear Equations

When dealing with a non-linear PDE, the above procedure leads to a non-linear system of equations

$$A(U)U = F(U),$$

where the stiffness matrix $A(U)$ as well as the load vector $F(U)$ depend on U. We denote the residual vector by

$$r(U) := A(U)U - F(U)$$

and consider different methods to find a solution, i.e., U, such that $r(U) = 0$. Since in the following we are not restricted to our FEM-related problem, we instead write $f : \mathbb{R}^n \to \mathbb{R}^n$ (sufficiently smooth) instead of r and x instead of U. A well known approach to the problem

$$\text{find } x \in \mathbb{R}^n : \quad f(x) = 0 \tag{3.5}$$

is Newton's method. Starting with an initial guess x^k, one obtains an improved approximation by computing

$$x^{k+1} := x^k - D\left(f(x^k)\right)^{-1} f(x^k),$$

[1]http://www.mathworks.de/de/help/pde/ug/assempde.html

where D denotes the Jacobian. Although under suitable assumptions quadratically convergent, see for example [33], a bad initial guess can easily render Newton's method useless, for it is only locally convergent. One therefore turns to interpret (3.5) as an unconstrained optimization problem

$$\min_{x\in\mathbb{R}^n} h(x), \tag{3.6}$$

where $h(x) = \|f(x)\|_2^2$, and wishes to construct a descent sequence $\{x^k\}_k$ such that $\{h(x^k)\}_k$ decreased monotonically, i.e., $h(x^{k+1}) < h(x^k)$, and x^k thus converges to a minimum. Let $g^k = g(x^k)$ with $g(x) = Dh(x)$ be the gradient of our objective function h. To construct the descent sequence, we choose in the k^th^ step a *search direction* $d^k \neq 0$ satisfying the descent condition

$$g^{k^{\mathrm{T}}} d^k < 0, \tag{3.7}$$

and consider the associated *search path* given by $x^k + \alpha d^k$, $\alpha \geq 0$. We then calculate a suitable *step size parameter* α^k, such that

$$h(x^k + \alpha^k d^k) < h(x^k), \tag{3.8}$$

which is, due to the descent condition (3.7) and $h(x^k + \alpha^k d^k) = h(x^k) + \alpha^k g^{k^{\mathrm{T}}} d^k + o(\alpha^k)$, indeed possible for sufficiently small α^k. This process is referred to as *line search* and an obvious candidate for α^k is

$$\alpha^k := \mathrm{argmin}_{\alpha>0} h(x^k - \alpha d^k),$$

which is called *exact line search*. However, this choice is considered ineffective due to its large effort (in every step, a global optimization problem has to be solved). One therefore tries to minimize f inexactly, while taking care that the steps are neither too long nor too short. To achieve this, one can compute a measure for the progress of the line search, such as the *Goldstein quotient*

$$\mu^k(\alpha) := \frac{h(x^k + \alpha d^k) - h(x^k)}{\alpha g^{k^{\mathrm{T}}} d^k}, \quad \alpha > 0, \tag{3.9}$$

and introduces the Goldstein condition

$$\mu_1 \leq \mu^k(\alpha) \leq \mu_2, \quad 0 < \mu_1 < \mu_2. \tag{3.10}$$

The Goldstein condition tries to ensure sufficient descent and keep α away from 0, the situation is nicely illustrated in Figure 3.2. Usually one introduces a third parameter $1 \geq \mu_3$ as well, to prevent too small steps whenever g^k becomes too small, i.e., h flat.

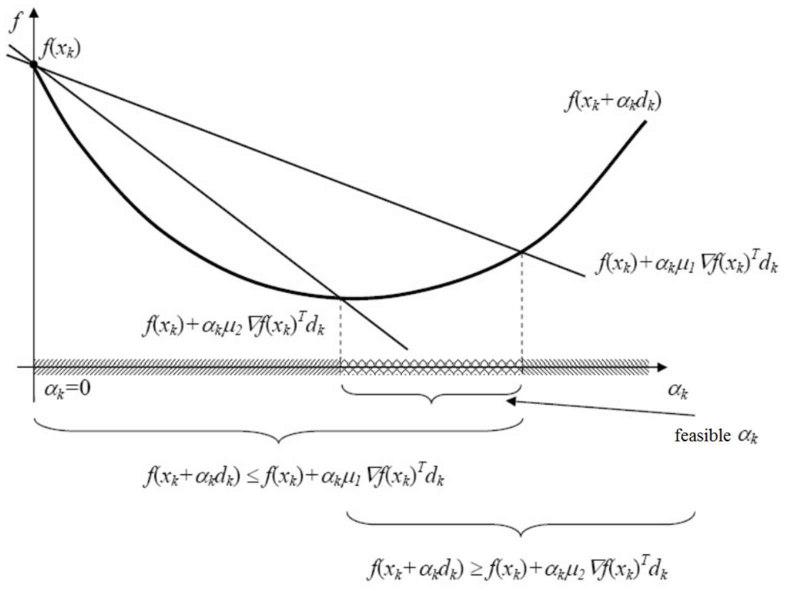
f
$f(x_k)$
$f(x_k+\alpha_k d_k)$
$f(x_k)+\alpha_k\mu_1\nabla f(x_k)^T d_k$
$f(x_k)+\alpha_k\mu_2\nabla f(x_k)^T d_k$
$\alpha_k=0$
α_k
feasible α_k
$f(x_k+\alpha_k d_k)\leq f(x_k)+\alpha_k\mu_1\nabla f(x_k)^T d_k$
$f(x_k+\alpha_k d_k)\geq f(x_k)+\alpha_k\mu_2\nabla f(x_k)^T d_k$

Figure 3.2: Geometric interpretation of the Goldstein condition (3.10), note the admissible set of step length parameters. Figure taken from [7] (and modified).

An efficient way to obtain a step length parameter satisfying the Goldstein condition, is the use of a *backtracking* strategy. In its simplest form, it looks as follows:

1. Start with initial value $\alpha_0 > 0$, e.g $\alpha_0 = 1$, set $\alpha^{(0)} = \alpha_0$ and $l = 0$.
2. Until $\mu^k(\alpha^{(l)})$ satisfies (3.10) (or alternatively, until descent is achieved):
   - set: $\alpha^{(l+1)} = \tau\alpha^{(l)}$, where $\tau \in (0,1)$ is fixed, e.g., $\tau = \frac{1}{2}$,
   - increment: $l = l+1$.
3. Choose $\alpha^k = \alpha^{(l)}$.

This way one reduces the line search to a finite process. By choosing the Newton direction $d^k = D(f(x^k))^{-1}f(x^k)$ as decent direction and $\alpha_0 = 1$ as initial step length, we recover the quadratic convergence of Newton's method as x^k advances towards the minimum, cf. [33]. Such an approach is called *damped Newton method* and the non-linear solver *pdenonlin*[2] implemented in MATLAB and used for our simulations is based on a similar methodology as described above. For the calculation of the Jacobian, we rely on a numerical approximation computed by finite differences. We mention that we presented only the very basic ideas, for a

[2]http://www.mathworks.de/help/pde/ug/pdenonlin.html

more in-depth discussion of optimization algorithms see the vast literature on the subject (note that the nomenclature is not uniform). An introduction can be found in [33].

3.1.3 Grid Generation

As we have outlined above, the triangulation of the computational domain is the very foundation of the finite element method and much of the quality of the numerical results relies on the quality of the mesh. We therefore consider it important to understand the basic working principle and devote this section to a short description of *distmesh*[3] by [29], on which *mesh2d*[4], the algorithm used for our simulations, is based. Note that the working principle of *distmesh* naturally extends to the 3D case.

The rationale behind unstructured meshes is to use elements of varying size to resolve fine features of the underlying geometry while at the same time balancing the total mesh size by having a coarse grid where possible. Using error indicators, a adaptive solvers can be developed to impose further constraints on the mesh size, in order to improve the approximate solution.

Distmesh

We consider the planar 2D case. To begin with, we note that for any given set of points (that do not lie on the same straight line), there exists a *Delaunay triangulation* [11]. Such a triangulation is characterised by the Delaunay condition, demanding that no point lies inside the circumcircle of any triangle. This kind of triangulation is widely used, since it tends to maximize the minimum angle of all triangles, thereby improving error estimates. One way, and probably least efficient, to obtain a Delaunay triangulation is to start with any triangulation and then flip edges until all triangles satisfy the Delaunay condition. Knowing that every non-degenerated point set can be triangulated, we need a to find a way to distribute these points to obtain in well-shaped mesh. Distmesh uses a *signed distance function* $d(x,y)$, that assumes negative values inside the region, to describe the shape of the region. The basic idea of the algorithm is to interpret a triangulation as a truss structure or a structure of springs, where an edge corresponds to a bar or spring and a node to a joint. In accordance with this interpretation, we discover a proper distribution of points by solving for a static force equilibrium. We use ordinary linear springs to implement this idea. Allowing only repulsive

[3]http://persson.berkeley.edu/distmesh/

[4]http://www.mathworks.com/matlabcentral/fileexchange/25555-mesh2d-automatic-mesh-generation

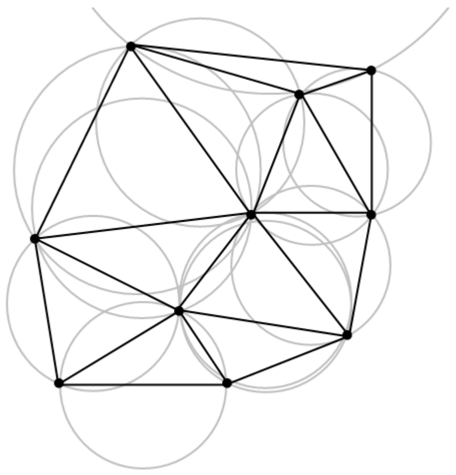

Figure 3.3: A Delaunay triangulation of a point set, note that indeed no point lies inside the circumcircle of any triangle. Figure taken from [38].

forces, we get a force-displacement relationship $f(l,l_0)$ given by

$$f(l,l_0) := \begin{cases} k(l_0 - l) & \text{if } l < l_0, \\ 0 & \text{if } l \geq l_0, \end{cases} \tag{3.11}$$

where k is the spring constant, l the current length of the bar, i.e., the euclidean distance between the joints, and l_0 its unextended length. An external reaction force acting normal to the boundary will be needed to prevent the nodes from moving outside the region (see below). Collecting (applying MATLAB notation) the x- and y-coordinates of all N meshpoints in p,

$$p = [x\,y],$$

the resulting force at each point is given by

$$F(p) = F_{\text{int}}(p) + F_{\text{ext}}(p). \tag{3.12}$$

Here F_{int} denotes the interior forces due to the point-to-point interaction via the bars and F_{ext} is the external (reaction) forces coming from the boundaries. Since the force-displacement relationship depends on the triangulation, i.e., how the points are connected might change as they move, $F(p)$ depends non-continuously on p. As a consequence, finding p, such that $F(p) = 0$, is not an easy task. We therefore introduce an the artificial time-dependence, to avoid a direct calculation of p and obtain an iterative scheme. Considering the system of ODEs,

$$\frac{dp}{dt} = F(p), \quad \text{for } t > 0 \text{ and } p(0) = p_0, \tag{3.13}$$

where p_0 is arbitrary initial distribution, we see that a stationary solution naturally satisfies $F(p) = 0$. Using a simple forward Euler scheme, we get

$$p_{n+1} = p_n + \Delta t F(p_n), \quad \text{with } t_n = n\Delta t. \tag{3.14}$$

The external reaction forces are implemented by moving every point that moved outside the boundary back to the closest boundary point. By doing so, points are allowed to move along the boundary but cannot go outside, conforming the requirement of a force acting normal to the boundary.

To facilitate the point spread and assure that they cover the whole region, it is important to have repulsive forces, i.e., $f > 0$, at most of the bars. This can be achieved by choosing l_0 slightly larger than the actually desired length. The desired edge length distribution enters the algorithm by means of an *element size function* $h(x,y)$, giving the *relative* distribution of element sizes over the domain. For a uniform mesh, l_0 is the same everywhere and thus h is constant. In the non-uniform case, we want higher h-values effect smaller desired lengths, i.e., relatively more points in the respective region. We therefore introduce a scaling factor s, being the ratio between the actual mesh area and the desired size,

$$s := \left(\frac{\sum l_i^2}{\sum h(x_i, y_i)} \right)^{1/2}, \tag{3.15}$$

where l_i is the actual edge length and $h(x_i, y_i)$ the value of h at the midpoints of the edges. The desired length is then given by $l_0 = 1.2hs$, where the additional factor 1.2 ensures slightly larger l_0 and thus $f > 0$ at most of the bars. Regarding the initial distribution, uniform distributions p_0 perform well for uniform meshes. For non-uniform meshes, it is advisable to start with a weighted distribution, for example by using a rejection method, where points are discarded with probability proportional to $1/h(x,y)^2$.

The overall procedure is depicted in Figure 3.4.

Now, some remarks regarding the size function h are in order, for there are a variety of different constituting factors that should be incorporated. First of all, the user may specify some regions, where a higher resolution is desirable. Additionally, there might be external information available, for example from error indicators, hinting at areas that need to be refined. Besides that, we certainly wish to resolve the underlying geometry and therefore need small elements to resemble curved boundaries and narrow features. To accomplish this, one has to identify parameters that specify the *local feature size*. For the mesh to be well-shaped, we further require neighbouring elements to be of not too different size. Last but not least, the overall number of points should be minimal. Clearly, the automatic construction of suitable size function is a non-trivial task and beyond the scope of this work. See [28] for an insightful discussion of this subject. As already men-

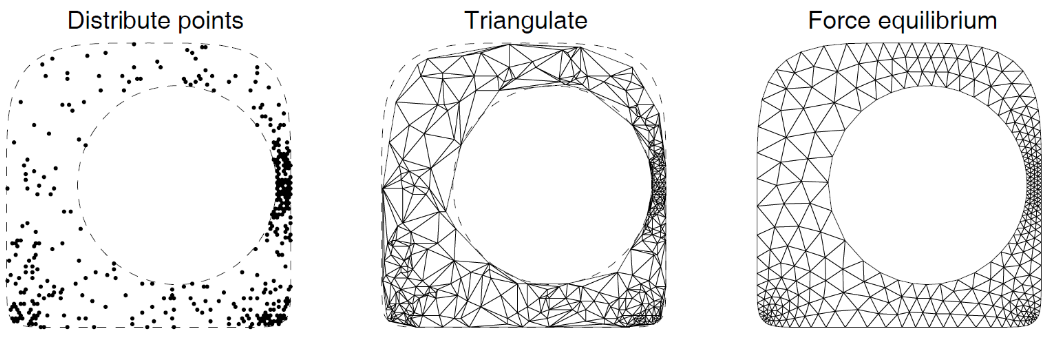

Figure 3.4: The generation of a non-uniform triangular mesh using *distmesh*, note the higher resolution in areas with finer geometric features. Figure taken from [29] (slightly modified).

tioned, we use (a slighty modified version of) *mesh2d* for the mesh generation. The main difference for the user being the possibility to define regions and subregions (faces) using polygons. See Figures 3.5 and 3.6 for exemplary meshes.

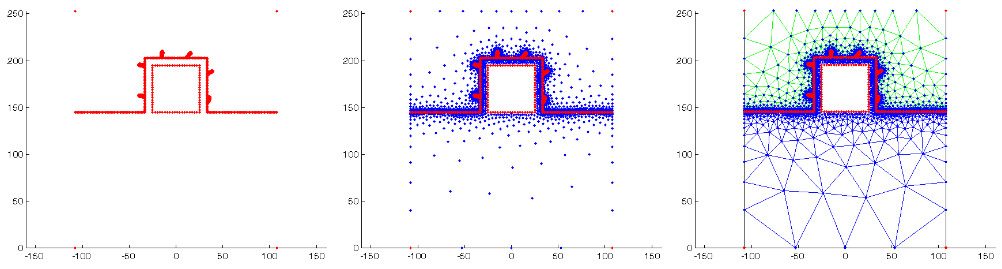

Figure 3.5: Left: a point set together with additional connectivity arrays defining subregions serves as input data for *mesh2d*. Middle: the result after the iterative procedure, note how local features are resolved by placing more points in narrower regions. Left: the mesh obtained after triangulation.

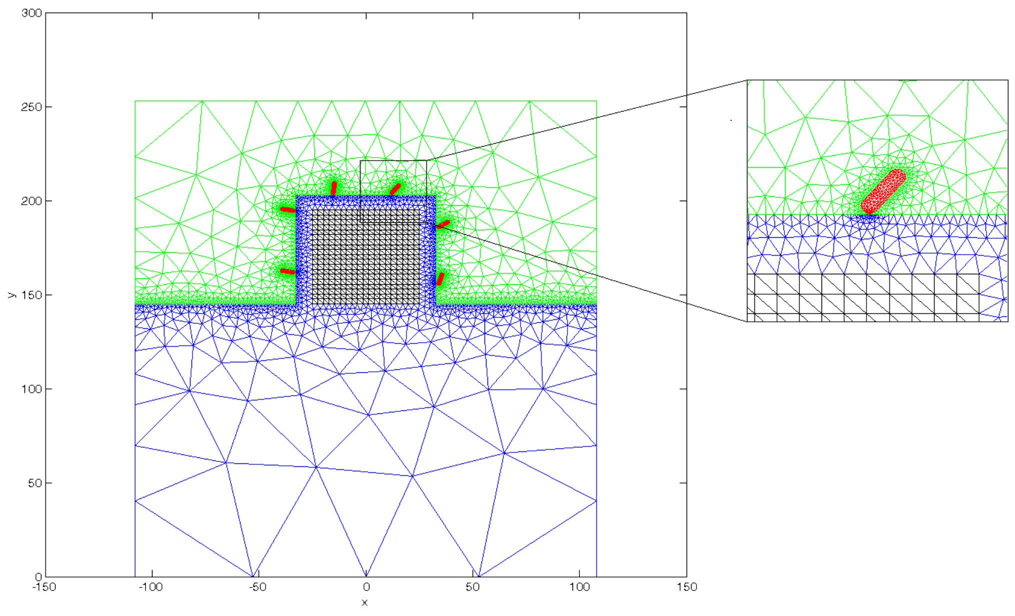

Figure 3.6: The mesh finally used in the FEM. Note that we use a uniform mesh (denser than depicted here) in the transducer, allowing us to compare solutions resulting from different configurations.

3.2 Numerical Integration

3.2.1 Problem Formulation

In this section we consider various methods for the numerical integration of (smooth) functions over high-dimensional domains. After formulating the problem and recalling some basic facts about 1D quadrature rules and their extension to higher dimensions, we take a closer look at sparse grid and stochastic integration methods. Most of the following is based on [15] and we thus adopt (almost) the same notation.

For $r \in \mathbb{N}$, let

$$\mathscr{C}^r := \left\{ f : \Omega \mapsto \mathbb{R}, \left\| \frac{\partial^s f}{\partial x^s} \right\|_\infty < \infty, s \leq r \right\},$$

$$\mathscr{W}_d^r := \left\{ f : \Omega \mapsto \mathbb{R}, \left\| \frac{\partial^{|\mathbf{s}|} f}{\partial x_1^{s_1} \cdots x_d^{s_d}} \right\|_\infty < \infty, s_i \leq r \right\},$$

be the function classes and $\Omega := [-1,1]^d$ the d-dimensional integration domain under consideration. Denoting the exact integral of a function f over Ω by

$$I^d f := \int_\Omega f(x)\,\mathrm{d}x,$$

we seek an approximation by means of *interpolatory quadrature formulas*

$$Q_l^d f := \sum_{i=1}^{n_l^d} w_{li} f(x_{li}),$$

where the *nodes* or *abscissas* x_{li} and *weights* w_{li} depend on the level $l \in \mathbb{N}$ but not on f. Requiring $n_l^d < n_{l+1}^d$, we obtain an ordered sequence. For the underlying set of nodes, we introduce

$$\Gamma_l^d := \left\{ x_{li}, i \in 1, 2, \dots, n_l^d \right\},$$

and say a quadrature formula is *nested*, if the respective *grids* satisfy $\Gamma_l^d \subset \Gamma_{l+1}^d$. To assess the performance of a given rule Q_l^d, we define the quadrature error

$$E_l^d f := \left| I^d f - Q_l^d f \right|,$$

and say a quadrature formula is of *order*, *exactness* or *precision* p, if it integrates exactly all polynomials having degree less then p. Clearly, the art of numerical integration lies in the advantageous selection of nodes x_{li} and weights w_{li}. In the following, we only consider nested rules, in order to keep the point count low by reusing points. As such rules tend to grow exponentially, we always start with

$$n_l^1 = 1 \quad \text{and} \quad Q_l^1 f = 2f(0).$$

Figure 3.7 depicts the nodes of one of the quadrature formulas considered in the next subsection.

3.2.2 Review of 1D Quadrature Formulas

Newton-Cotes formulas

Newton-Cotes formulas, such as the well known *trapezoidal* or *Simpson* rule, are based on the idea of substituting the integrand f by a suitable interpolation polynomial P and using $\int P(x)\,\mathrm{d}x$ as an approximation for $\int f(x)\,\mathrm{d}x$. The nodes are taken to be equally spaced and the weights are derived by integrating the interpolating Lagrange polynomial. For larger number of nodes, some of the weights become negative and render the approach numerically unstable. In general one

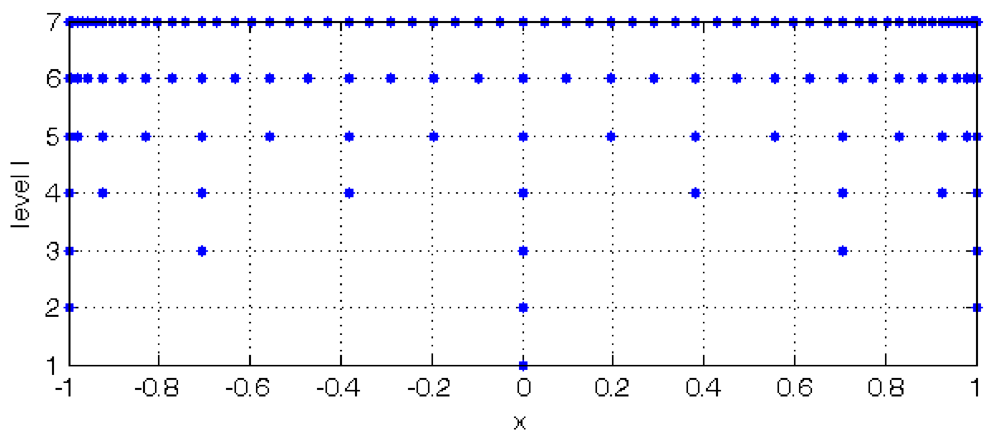

Figure 3.7: Clenshaw-Curtis quadrature points for different levels l, observe the nesting property and the exponential growth of the point count, $n_l^1 = 2^{l-1} + 1$ for $l \geq 2$.

thus uses iterated versions of low degree formulas to obtain a composite rule. We conclude with the properties of the trapeziodal rule [15]. For $l \geq 2$, we have

$$\begin{aligned} n_l^1 &= 2^{l-1} + 1, \\ p &= 2, \\ E_l^1 f &= O\left(2^{-2l}\right) \text{ for } f \in \mathscr{C}^2. \end{aligned}$$

For a thorough treatment of rules of different accuracy see for example [10]. Note however, that in general, using n_l^1 points allows to integrate exactly polynomials up to order n_l^1.

Gauss(-Patterson) formulas

Gauss formulas are defined by choosing the nodes and weights in such a way, that the resulting formula maximizes the degree of polynomials that it integrates exactly. With respect to their order, Gauss formulas are thus optimal quadrature rules. The integrands are of the more general form $f(x) = w(x)g(x)$, where $w(x)$ is a non-negative weight function. One then considers w-orthogonal polynomials, whose roots serve as the nodes and the weights can be derived by integrating the associated w-weighted Lagrange polynomials. The formulas obtained in this way are then of order $2n_l^1 + 1$. From a numerical perspective, it is important to note that the computation of nodes and abscissas can be related to an eigenvalue problem. Clearly, different weight functions lead to different orthogonal polynomials, some of the more common examples are given in Table 3.1.

$w(x)$	$[a,b]$	rule	prob. distr.
1	$[-1,1]$	Gauss-Legendre	Uniform
$\exp(-(x-\alpha)^2/\beta^2)$	$(-\infty,\infty)$	Gauss-Hermite	Gaussian
$\exp(-x\alpha)$	$[0,\infty)$	Gauss-Laguerre	Gamma
$(1-x)^{\alpha}(1+x)^{\beta}$	$[-1,1]$	Gauss-Jacobi	Beta

Table 3.1: Weight functions and corresponding Gaussian rule, the second term denotes the respective class of orthogonal polynomials. For stochastic integrals, it is recommended to use the appropriate type of quadrature rule.

Unfortunately, the roots of orthogonal polynomials are not nested. However, Kronrod [21] and Patterson [27] extended the original formulation to obtain sequences of nested quadrature formulas with maximal degree of exactness, which we will refer to as *Gauss-Patterson* rules. For more details we regarding the construction we refer to the above the literature and [10]. We conclude with the properties of the Gauss-Patterson rule [15]. For $l \geq 2$, we have

$$\begin{aligned} n_l^1 &= 2^l - 1, \\ p &= 3 \cdot 2^{l-1} - 1 \\ E_l^1 f &= O\left(2^{-lr}\right) \text{ for } f \in \mathscr{C}^r. \end{aligned}$$

Clenshaw-Curtis formula

The *Clenshaw-Curtis* formula [8] is based on integrating an expansion of the integrand in terms of Chebyshev polynomials, with the non-equidistant abscissas being the respective extreme points and leading to a nested rule. From a numerical perspective, we note that above procedure can be related to the discrete cosine transform, see [10] for a thorough treatment. We conclude with the properties of the Clenshaw-Curtis rule [15]. For $l \geq 2$, we have

$$\begin{aligned} n_l^1 &= 2^{l-1} + 1, \\ p &= n_l^1 + 1 \text{ and} \\ E_l^1 f &= O(2^{-lr}) \text{ for } f \in \mathscr{C}^r. \end{aligned}$$

Practical Considerations

We did not elaborate further on the details of the construction of the above quadrature rules, since we can rely on the MATLAB library *SANDIA_RULES*[5], providing a variety of different formulas.

Considering Figure 3.8, we observe that although comparable in the sense of polynomial exactness, the nodes of a Clenshaw-Curtis rule resemble more Gaussian quadrature points than equidistant Newton-Cotes points. Note further, that the pronounced factor-of-2 advantage of Gaussian rules (in the non-nested case) is rarely realized in numerical experiments, [36] and the insightful discussion therein. Recall that due to the different exponents, for a given level l, the number of nodes n_l^1 in a Gaussian and Clenshaw-Curtis rule can differ by almost a factor of 2.

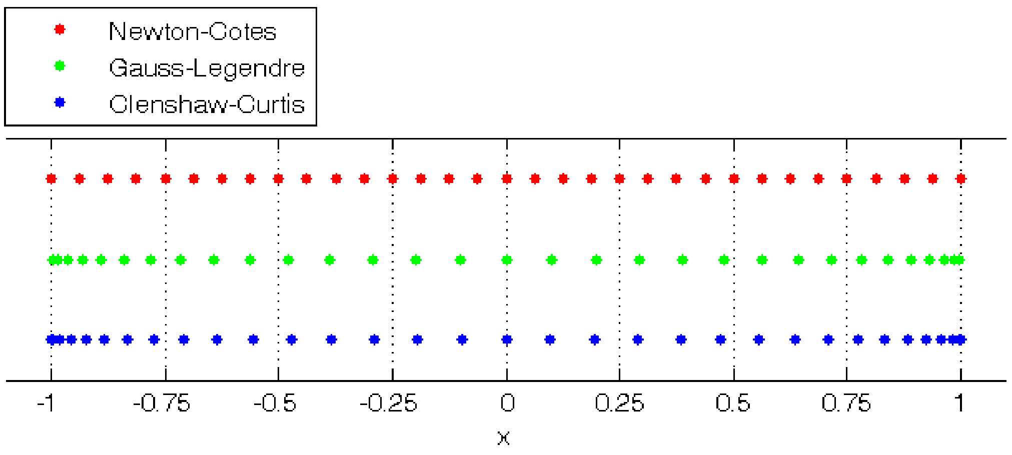

Figure 3.8: Newton-Cotes ($l = 6$, $n_l^1 = 33$), Gauss-Legendre ($l = 5$, $n_l^1 = 31$) and Clenshaw-Curtis ($l = 6$, $n_l^1 = 33$) quadrature points. The latter two showing a similar distribution, with more points shifted towards the end of the interval.

The Curse of Dimension

In the d-dimensional case, Fubini's theorem allows to compute a d-fold integral using iterated 1D integrals. This amounts to the so called *tensor product rule*

[5]http://people.sc.fsu.edu/~jburkardt/m_src/sandia_rules/sandia_rules.html and http://people.sc.fsu.edu/~jburkardt/m_src/sgmga/sgmga.html

$Q^1_{l_1} \otimes \cdots \otimes Q^1_{l_d}$, which is defined as the summation over all possible combinations

$$\left(Q^1_{l_1} \otimes \cdots \otimes Q^1_{l_d}\right) f := \sum_{i_1=1}^{n^1_{l_1}} \cdots \sum_{i_d=1}^{n^1_{l_d}} w_{l_1 i_1} \cdots w_{l_d i_d} f(x_{l_1 i_1}, \ldots, x_{l_d i_d}). \tag{3.16}$$

For simplicity, let the number of abscissas be the same in every dimension. For a given level l, we then need a total of $n = \left(n^1_l\right)^d$ function evaluations, whereas the accuracy remains the same, consider for example $f(x_1, \ldots, x_d) = g(x_1)$. Now, assuming our integration is based on a Gaussian rule, we observe the error bound

$$E^d_l = O(n^{-r/d}),$$

to deteriorate dramatically with increasing dimension, this behaviour is sometimes referred to as the *curse of dimension*. The remainder of this chapter will be devoted to methods that allow to overcome this problem, at least to a certain extent. We mention that the development of such methods is still an active area of research.

3.2.3 Numerical Integration using Sparse Grids

Before giving the exact definition of *Smolyaks's method* to handle tensor product problems, we sketch the basic idea. Let $\Omega = [0,1]^2$ and Q^1_l to be a family of nested quadrature rules, integrating exactly polynomials of degree up to l. The 2D product rule is then given by $Q^2_{l_1,l_2} = Q^x_{l_1} \otimes Q^y_{l_2}$, where adapted the notation regarding the upper subscripts in the 1D rules on the right hand side to indicate the dimension on which the rule acts. For example $Q^2_{1,1} = Q^x_1 \otimes Q^y_1$, uses four points, two in each dimension, to integrate exactly any terms in the linear combination of $1, x, y, xy$. Assume that the product rules take the following form:

$$\begin{aligned}
Q^x_1 &= af(0,0), \\
Q^x_2 &= bf(0,0) + cf(1,0), \\
Q^y_2 &= df(0,0) + ef(0,1).
\end{aligned}$$

We now consider the combination

$$\mathscr{A}^2_2 := Q^x_2 + Q^y_2 - Q^x_1 = (b+d-a)f(0,0) + cf(1,0) + ef(0,1)$$

and see, that $\mathscr{A}^2_2$ achieves the same as the product rule $Q^2_{1,1}$ using only three points. This is already the idea of Smolyak's construction: combine low-order product rules in a way to achieve a desired performance while at the same time avoiding the excessive function evaluations arising in the standard full product rule.

To begin with, we start with a sequence of 1D quadrature formulas

$$Q_l^1 f := \sum_{i=1}^{n_l^1} w_{li} f(x_{li}),$$

and define the difference quadrature formula

$$\Delta_k^1 := Q_k^1 - Q_{k-1}^1, \quad \text{setting } Q_0^1 := 0.$$

Δ_k^1, being a sum combining weights and function evaluations, is again a quadrature formula operating on the union of grids $\Gamma_k^1 \cup \Gamma_{k-1}^1$, which equals Γ_k^1 in the nested case.
Smolyak now devised the following rule to integrate a d-dimensional function f:

$$\mathscr{A}_l^d f := \sum_{|\mathbf{k}|_1 \leq l+d-1} \left(\Delta_{k_1}^1 \otimes \cdots \otimes \Delta_{k_d}^1\right) f, \tag{3.17}$$

where $\mathbf{k} = (k_1, \ldots, k_d) \in \mathbb{N}^d$ is a multi-index and $l \in \mathbb{N}$. Observe, that since

$$\left(Q_{l_1}^1 \otimes \cdots \otimes Q_{l_d}^1\right) f = \sum_{j=1}^{d} \sum_{1 \leq k_j \leq l} \left(\Delta_{k_1}^1 \otimes \cdots \otimes \Delta_{k_d}^1\right) f,$$

performing the summation in (3.17) over a different index set, namely the cube $|\mathbf{k}|_\infty \leq l$ instead of the simplex $|\mathbf{k}|_1 \leq l+d-1$, recovers the ordinary product rule. Expressing $\mathscr{A}_l^d$ in terms of the quadrature formulas $Q_{k_j}^1$ yields [37]

$$\mathscr{A}_l^d f = \sum_{l \leq |\mathbf{k}|_1 \leq l+d-1} (-1)^{l+d-|\mathbf{k}|_1-1} \binom{d-1}{|\mathbf{k}|_1 - l} \left(Q_{k_1}^1 \otimes \cdots \otimes Q_{k_d}^1\right) f. \tag{3.18}$$

We have not restricted ourselves to specific 1D rules and we will see later on, that we can further modify the index set in the summation to tailor the construction according to our needs. With the current choice, it is clear that the level k_j of any 1D rule never exceeds l.
We define the so called *sparse grid* as the set of points in $\mathscr{A}_l^d$, namely the union

$$\Theta_l^d := \biguplus_{|\mathbf{k}|_1 \leq l+d-1} \Theta_{k_1}^1 \times \cdots \times \Theta_{k_d}^1,$$

over the pairwise disjoint difference grids $\Theta_{k_1}^1 \times \cdots \times \Theta_{k_d}^1$, where

$$\Theta_l^1 := \Gamma_l^1 \setminus \Gamma_{l-1}^1, \text{ with } \Gamma_0^1 := \emptyset,$$

are the one-dimensional difference grids. Not surprisingly, for nested 1D quadrature formulas, the corresponding sparse grid formula is nested as well [26]. The number of points constituting the tensor product is calculated by

$$n_l^d = \sum_{\mathbf{k}|_1 \leq l+d-1} m_{k_1} \cdots m_{k_d},$$

and if the 1D rule is based on $n_l^1 = O(2^l)$ points, we have [37]

$$n_l^d = O\left(2^l l^{d-1}\right).$$

Here we observe the important difference to the full tensor product grid, where the point count is $O(2^{ld})$.
In a practical application, one computes

$$\mathscr{A}_l^d f := \sum_{|\mathbf{k}|_1 \leq l+d-1} \sum_{j_1=1}^{m_{k_1}} \cdots \sum_{j_d=1}^{m_{k_d}} w_{\mathbf{kv}} f(x_{\mathbf{kv}}),$$

to obtain an approximation of the exact integral. To compute the nodes an weights, we resort to the mentioned *SANDIA* library. For a discussion regarding the numerical aspects, see for example [15].

Finally, a short discussion of the integration error E_l^d is in order. We assume the underlying 1D formulas to satisfy the error bound

$$E_l^1 f = O\left(\left(n_l^1\right)^{-r}\right) \qquad \text{for } f \in \mathscr{C}_1^r,$$

which holds for all interpolatory quadrature formulas with positive weights, such as the above discussed Clenshaw-Curtis and Gauss-Patterson rules, cf. [10]. Such formulas with a point count of order $n_l^1 = O(2^l)$ taken as a 1D basis then lead to a sparse grid formula of order [37]

$$E_l^d f = O\left(\left(n_l^d\right)^{-r} \left(\log n_l^d\right)^{(d-1)(r+1)}\right) \tag{3.19a}$$

$$= O\left(2^{-lr} l^{(d-1)(r+1)}\right), \text{ for } f \in \mathscr{W}_d^r. \tag{3.19b}$$

Note the strong dependency on the smoothness and only the weak one on the dimension. Regarding its order, the multi-variate rule is exact for polynomials with total degree less or equal $l+d-1$ whenever the uni-variate rules are exact for polynomials with degree less or equal to l [26].

In our simulations, we use sparse grids based on Clenshaw-Curtis and Gauss-Patterson rules, which are depicted in Figure 3.9. An application to test problems

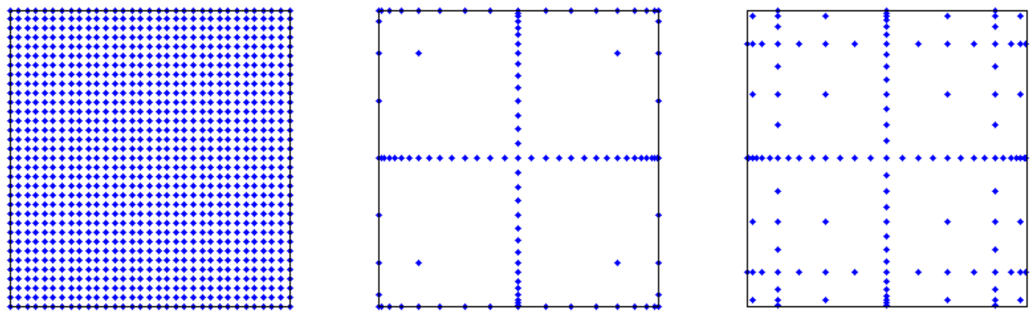

Figure 3.9: From left to right: Newton-Cotes full product grid based on trapezoidal rule ($l = 6$, $n_6^2 = 33^2 = 1089$), Clenshaw-Curtis sparse grid ($l = 6$, $n_6^2 = 113$), Gauss-Patterson sparse grid ($l = 5$, $n_6^2 = 129$).

is presented in 3.2.5. To create a more tangible picture of how the rules are actually combined, we consider a 2D example, having in mind for example Clenshaw-Curtis rule $Q_l^1 = CC_l$. The Smolyak rule then takes the form

$$\begin{aligned}
\mathscr{A}_1^2 &= CC_1 \otimes CC_1, \\
\mathscr{A}_2^2 &= CC_1 \otimes CC_1 + CC_1 \otimes CC_2 - CC_1 \otimes CC_1, \\
\mathscr{A}_3^2 &= CC_3 \otimes CC_1 + CC_2 \otimes CC_2 + CC_1 \otimes CC_3 - CC_2 \otimes CC_1 - CC_1 \otimes CC_2,
\end{aligned}$$

and Figure 3.10 shows, qualitatively, the situation for $\mathscr{A}_3^2$. For larger levels the coefficients take a more complicated form.

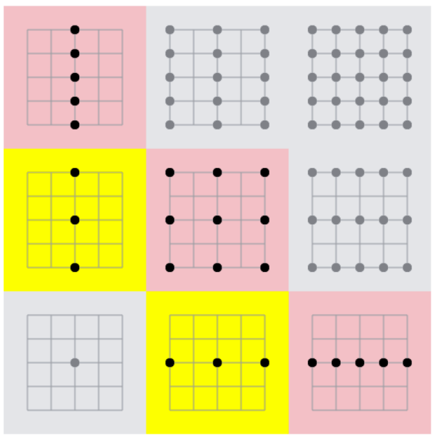

Figure 3.10: Only the five lower order grids contribute to $\mathscr{A}_2^2$, which are, because of nesting, contained in the diagonal entries. Figure taken from [20].

Generalized Sparse Grids

What we have discussed so far is often termed the *conventional* or *classical construction* and we only considered *isotropic* grids, i.e., grids where each dimension is treated the same. Clearly, one can base the construction on mixed rules to account for anisotropic effects, an example being shown in Figure 3.11.

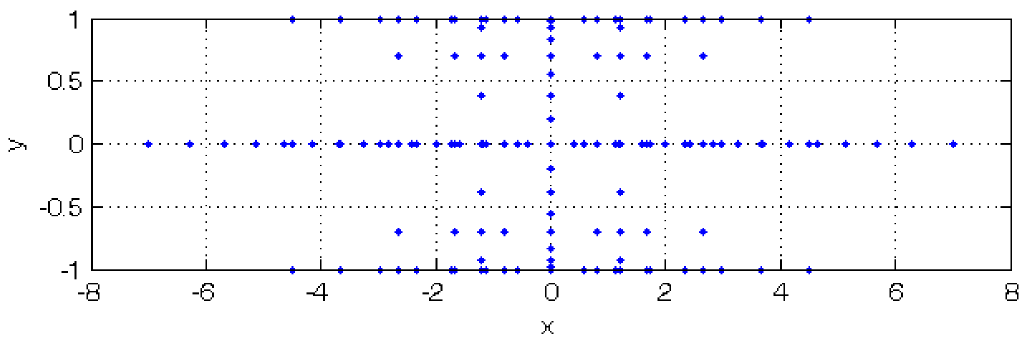

Figure 3.11: A Smolyak sparse grid based on a Gauss-Hermite rule in x- and Clenshaw-Curtis rule in y-direction ($l = 5$).

For some applications however, it is more appropriate to consider general index sets in (3.17). This allows to tackle *anisotropic* problems in a systematic fashion and the respective approach is called *general sparse grid construction* [15]. To begin with, we say an index set $\mathscr{I}$ is called *admissible* if the condition

$$\forall \mathbf{k} \in \mathscr{I} : \quad \mathbf{k} - e_j \in \mathscr{I} \text{ for (some) } 1 \leq j \leq d, k_j > 1,$$

is satisfied, where e_j denotes the j^{th} unit vector. According to the above condition, an index set is admissible, if for every index $\mathbf{k}$, $\mathscr{I}$ contains all indices having smaller entries than $\mathbf{k}$ in at least one dimension as well. The general sparse grid construction is then given by replacing the index set in (3.17) by an arbitrary admissible index set $\mathscr{I}$, i.e.

$$\mathscr{A}_l^d f := \sum_{\mathbf{k} \in \mathscr{I}} \left(\Delta_{k_1}^1 \otimes \cdots \otimes \Delta_{k_d}^1\right) f. \tag{3.20}$$

Note that by choosing $\mathscr{I} := \{\mathbf{k} : |\mathbf{k}|_1 \leq l + d - 1\}$ or $\mathscr{I} := \{\mathbf{k} : |\mathbf{k}|_\infty \leq l\}$ both the conventional and the full product formula are included as special cases. Since we aim to resolve the dimensions differently, a straightforward generalization is to consider a general class of simplices of the form $\mathbf{a} \cdot \mathbf{k} \leq l + d - 1$, where $\mathbf{a} \in \mathbb{R}^+$ is a weight vector directed at the different dimensions. This route is taken in [25] and an example is depicted in Figure 3.12. [14] proposes a more general setting, since in some cases, for example when more or less points in mixed directions are required, even such a simplex may be inadequate.

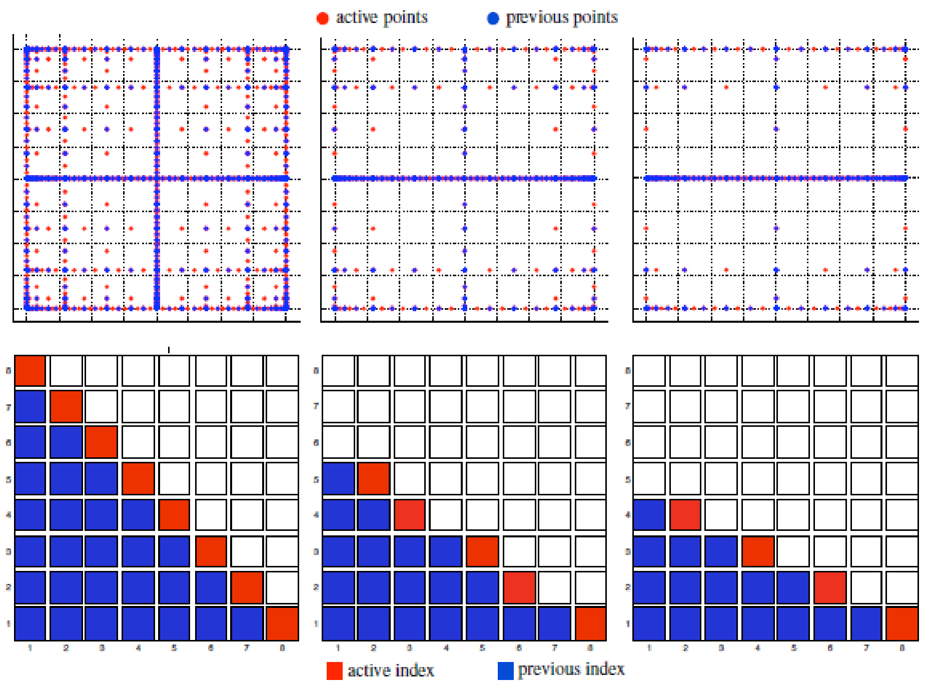

Figure 3.12: First line, from left to right: isotropic Clenshaw-Curtis sparse grid ($l = 8$), anisotropic Clenshaw-Curtis sparse grid $a_2/a_1 = 3/2$, anisotropic Clenshaw-Curtis sparse grid $a_2/a_1 = 2/1$. Second line, from left to right: the corresponding indices entering the rule. Figure taken from [25].

Depending on whether a priori information regarding the integrand is available, the weight vector **a** is either set beforehand or chosen iteratively by means of an adaptive algorithm throughout the integration process. We do not elaborate on this subject further and instead refer the interested reader to [25] and [14]. In our case, we expect the molecules farther away from the transducer to have a smaller impact on the overall result.

3.2.4 (Quasi) Monte Carlo Methods

Generally speaking, a *Monte Carlo method* is a computational algorithm based on random numbers devised to solve a problem whose solution can be obtained by means of a stochastic process. As is the case in numerical integration, the underlying problem must not be directly related to random events at all.

Now, let λ_d the Lebesgue-measure on $\mathbb{R}^d$ and B an integration domain with $0 < \lambda(B) < \infty$. Then, by defining the probability space $(B, \mathscr{L}(B), \mathrm{d}\mu)$ with probability measure $\mathrm{d}\mu = \lambda/\lambda(B)$, we have

$$\int_B f(x)\,\mathrm{d}x = \lambda(B) \int_B f(x)\,\mathrm{d}\mu(x) = \lambda(B)\mathbb{E}f(X)$$

for an integrable function f. Here X is a μ-distributed (i.e., uniformly) random variable and $\mathbb{E}$ denotes the expected value. We have thus transformed the problem of integrating f to estimating $\mathbb{E}f(X)$. Intuitively, we expect to obtain an approximation of the expected value by repeatedly sampling the random variable. According to the *strong law of large numbers*, which by *Kolmogorov's theorem* holds for *independent and identically distributed (iid)* random variables, we indeed obtain

$$\lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} f(x_i) = \mathbb{E}f(X),$$

almost surely. In any computation, N is clearly a finite number and one usually takes the standard deviation $\sigma(f) = \sqrt{\mathbb{V}f(X)}$ to obtain a measure for the deviation from the true expected value. For the variance $\mathbb{V}f(X)$, it is easy to show, that

$$\int_B \cdots \int_B \left(\frac{1}{N} \sum_{i=1}^{N} f(x_i) - \mathbb{E}f(X) \right)^2 \mathrm{d}\mu(x_1)\ldots\mu(x_N) = \frac{\mathbb{V}f(X)}{N},$$

holds for $f \in L^2(\mu)$ and iid random variables. This results in an average error of

$$E^d_{\mathrm{MC}} f = O(\sigma(f)N^{-1/2}), \tag{3.21}$$

which does not depend on the dimension at all. In the naive form, one generates a set of uniformly distributed point $x_i \in B$, $i = 1, \ldots, N$ to get

$$\frac{\lambda(B)}{N} \sum_{i=1}^{N} f(x_i) \approx \int_B f(x)\,\mathrm{d}x.$$

Although we overcame the curse of dimension, we now face some serious shortcomings. First and foremost, the error bound is only of stochastic nature, secondly, the above result heavily relies on the quality of the random variables (independence) and thirdly, the constant $\sigma(f)$ is unknown. *Quasi Monte Carlo methods (qMC)* try to remedy the disadvantages of the naive approach by resorting to quasi-random numbers, thereby leading to both better and deterministic error bounds. For a detailed account on the qMC method and the topic of random number generation, see for example [23]. In the following, we will only state the most important ingredients needed for our purpose.

Let $P = (x_1,\ldots,x_N)$ be a set of points in Ω, $\mathscr{J}^*$ the family of sub-cubes of Ω given by $\prod_{i=1}^d [0,u_i)$ and by $\mathscr{J}$ the family of sub-cubes of Ω given by $\prod_{i=1}^d [u_i,v_i)$. Then the *(star) discrepancy* of P with respect to $\mathscr{J}^{(*)}$ is defined as

$$D_N^{(*)}\left(\mathscr{J}^{(*)},P\right) = \sup_{J\in\mathscr{J}^{(*)}} \left|\frac{A(J;P)}{N} - \lambda_d(J)\right|,$$

where $A(J;P)$ is a *counting function* giving the number of points in P that also belong to J. We see from the definition, that a sequence has *low discrepancy*, if the proportion of points that belong to an arbitrary set B is roughly the same as the measure of B. This resembles the situation that occurs on average for uniformly distributed points (in contrast to a particular sample that can exhibit large holes or clusters). Hence, a low-discrepancy quasi-random sequence seeks to fill the space uniformly. The prefix 'quasi' emphasizes that such sequences are neither random nor pseudorandom, as they indeed fail many statistical tests for randomness. We require the integrand f to have bounded *variation* $V(f)$ on $\bar{\Omega}$ in the sense of *Hardy and Krause* and a point set $P = (x_1,\ldots,x_N)$ in Ω given. We then have the fundamental *Koksma–Hlawka inequality* [19], stating that

$$E^d_{\text{qMC}} f = \left|\frac{1}{N}\sum_{i=1}^N f(x_i) - \int_{\bar{\Omega}} f(u)\,du\right| \le V(f)\,D_N^*(P). \tag{3.22}$$

Accordingly, we aim to find sequences with lowest possible star discrepancy which are usually simply called *low-discrepancy sequences*. In contrast to (3.21), the above error bound (3.22) is deterministic and we indeed exploit the integrands regularity (in terms of its variation). Furthermore, the inequality is sharp in the sense, that for every such set P and $\varepsilon > 0$, there exists a $\mathscr{C}^\infty$ function f with $V(f) = 1$ such that

$$\left|\frac{1}{N}\sum_{i=1}^N f(x_i) - \int_{\bar{\Omega}} f(u)\,du\right| > D_N^*(x_1,\ldots,x_N) - \varepsilon.$$

The dimension enters to some extend via the discrepancy $D_N^*(P)$, which is observed to be bounded from below by a term depending on N and d. However, it is possible to obtain near optimal sequences whose construction is beyond the scope of this work but treated thoroughly for example in [23]. For our purpose, it is sufficient to note that MATLAB[6] provides generators for the well known *van der Corput sequences*, *Halton sequences* and *Sobol sequences*.

[6]http://www.mathworks.de/de/help/stats/generating-quasi-random-numbers.html

As a concluding remark, we state that in the one-dimensional case, one can show for $0 \leq x_1 < x_2 < \cdots < x_n \leq 1$, that

$$D_N^*(P) = \frac{1}{2N} + \max_{1 \leq i \leq N} \left| x_i - \frac{2i-1}{2N} \right|,$$

with equality for $x_i = \frac{2i-1}{2N}$, which corresponds to the classical midpoint rule.

In Figure 3.13, we see a comparison of the nodes of all types of quadrature rules considered in this work. An application to test problems is presented in the

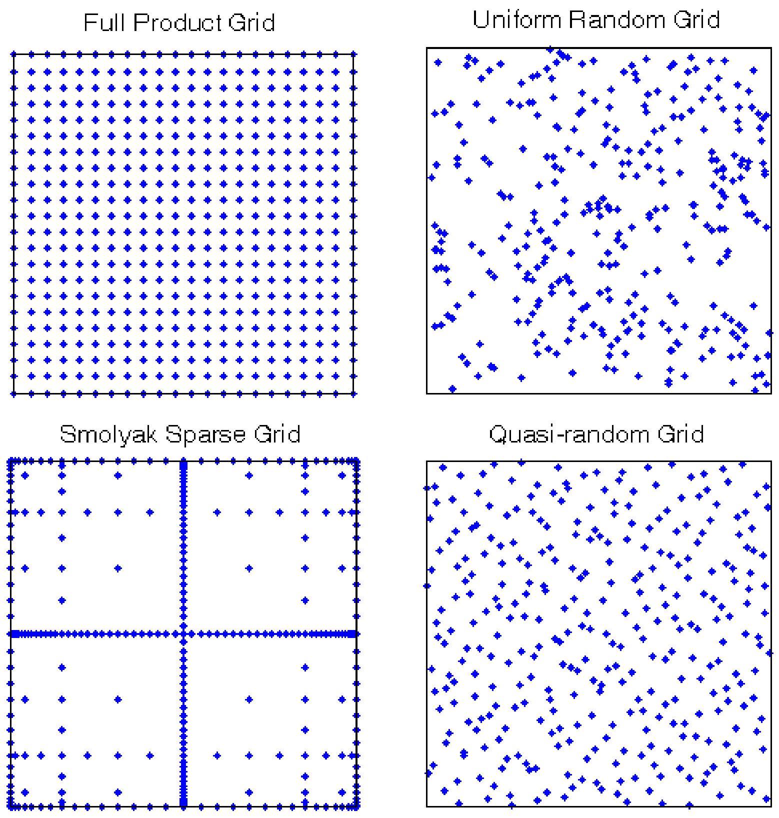

Figure 3.13: Upper left: a 22×22 full product grid, upper right: 500 uniformly distributed (pseudo)random points (note the holes and clusters), lower left: a Smolyak sparse grid based on Clenshaw-Curtis nodes ($l = 7$, $n_7^2 = 321$), lower right: 500 low-discrepancy quasi-random points (a Halton point set, observe the uniform filling).

next subsection.

3.2.5 Comparison and Adaptive Strategies

We will now assess the performance of the so far discussed methods by applying them to the set of well known Genz test functions [13]. Each of them belonging to a certain family with distinguished properties.

Integrand Family	Attribute
$f_1(x) = \cos\left(2\pi u_1 + \sum_{i=1}^{n} a_i x_i\right)$	*Oscillatory*
$f_2(x) = \prod_{i=1}^{n} \left(a_i^{-2} + (x_i - u_i)^2\right)^{-1}$	*Product Peak*
$f_3(x) = \left(1 + \sum_{i=1}^{n} a_i x_i\right)^{(n+1)}$	*Corner Peak*
$f_4(x) = \exp\left(-\sum_{i=1}^{n} a_i^2 (x_i - u_i)^2\right)$	*Gaussian*
$f_5(x) = \exp\left(-\sum_{i=1}^{n} a_i \lvert x_i - u_i \rvert\right)$	C^0 *Function*
$f_6(x) = \begin{cases} 0 & \text{if } x_{1,2} > u_{1,2}, \\ \exp\left(\sum_{i=1}^{n} a_i x_i\right) & \text{otherwise.} \end{cases}$	*Discontinuous*

The integration domain is the n-dimensional unit hypercube $\Omega = [0,1]^n$. The parameters $u = (u_1, \dots, u_n)^{\mathrm{T}} \in \Omega$ can be varied randomly to generate different samples, the difficulty of the corresponding integral should not be affected. The parameters $a = (a_1, \dots, a_n)^{\mathrm{T}} \in \Omega$ will, with increasing $\|a\|$, make the integrands more difficult. The difficulty level for a given series can be fixed by properly scaling a. Here, for the j^{th} integrand family one selects fixed numbers n, e_j and h_j and computes for each test in series a vector $a' \in \Omega$. a is then obtained by $a = ca'$, where c is chosen in such that a satisfies

$$n^{e_j} \sum_{i=1}^{n} a_i = h_j. \tag{3.23}$$

In order to use our integration methods effectively, we need a suitable termination criterion. Note that this is indeed a vital question, since stepping up the level l of our sparse grid quadrature rules doubles the size of the grid, i.e., the new level comes at the cost of all previous ones. The obvious candidates are clearly the absolute as well as the relative changes in the integral value for two consecutive levels, i.e.

$$\Delta_l = \mathscr{A}_l^d - \mathscr{A}_{l-1}^d, \tag{3.24}$$

$$\delta_l = \frac{\Delta_l}{\mathscr{A}_l^d}, \tag{3.25}$$

and we terminate when either $|\Delta_l|$ or $|\delta_l|$ falls below a given tolerance ε. For (q)MC integration a similar scheme can be applied. However, we also want to consider *Richardson extrapolation*, as proposed in [22].

The idea of Richardson extrapolation is the following: Suppose we want to approximate a quantity $\hat{A}$ and we have an approximation of order p with small parameter $h > 0$ satisfying

$$\hat{A} = A(h) + c \cdot h^p + O(h^{p+1}).$$

Now, we take two slightly different parameters h_1 and h_2, which are of similar order of magnitude, $O(h)$, but not too close together, i.e., $\left(h_2^p - h_1^p\right) = O(1/h^p)$. We then have two equations

$$\hat{A} = A(h_{1,2}) + c \cdot h_{1,2}^p + O\left(h_{1,2}^{p+1}\right)$$

from which we can eliminate the $c \cdot h_{1,2}^p$ term by multiplying the first equation by h_2^p, the second by h_1^p and then subtract to obtain

$$\left(h_2^p - h_1^p\right)\hat{A} = h_2^p A(h_1) - h_1^p A(h_2) + O\left(h^{2p+1}\right),$$

yielding

$$\hat{A} = \frac{h_2^p A(h_1) - h_1^p A(h_2)}{\left(h_2^p - h_1^p\right)} + O\left(h^{p+1}\right).$$

This motivates the definition of the Richardson extrapolation value for $\hat{A}$,

$$A_R = \frac{h_2^p A(h_1) - h_1^p A(h_2)}{\left(h_2^p - h_1^p\right)}, \tag{3.26}$$

whose error is of order $p+1$.

Although the assumptions regarding the error are not exactly met in the Smolyak construction, using (3.19), we nevertheless hope that

$$\mathscr{A}_{R_l}^d := \frac{a_{l-1}\mathscr{A}_l^d - a_l \mathscr{A}_{l-1}^d}{a_{l-1} - a_l}, \tag{3.27a}$$

$$a_l^d := 2^{-lr} l^{(d-1)(r+1)}, \tag{3.27b}$$

gives an improved estimate. For a given level l, we then compute

$$\Delta_{Rl} = \mathscr{A}_l^d - \mathscr{A}_{Rl}^d, \tag{3.28}$$

and terminate if $|\Delta_{Rl}|$ is below ε. Clearly, r is not known and we have to rely on guesses.

We apply our integration methods to the above stated test functions, using the parameters given in [26]. Hence we set $d = 10$ and the difficulty according to (3.23) with $n^{e_j} = 1$ and h_j as in Table 3.2, where j denotes the respective integrand family f_j. Additionally, we also consider the performance of the three different choices for the termination criterion.

j	1	2	3	4	5	6
h_j	9.0	7.25	1.85	7.03	2.04	4.3

Table 3.2: Level of difficulty for the different integrand families.

Conclusion

Considering Figure 3.14, we observe a very good performance of the sparse grid quadrature rule. In particular it seems to be superior to conventional (q)MC integration methods. It is to no surprise, that all methods fail in the discontinuous case $(f_6(x))$, however, the comparatively good result of the sparse grid rule in case of the non-smooth C^0 function is noteworthy. As the qMC method based on the Halton sequence delivers better results than the one based on the Sobol sequence, the former will be used in our simulations.

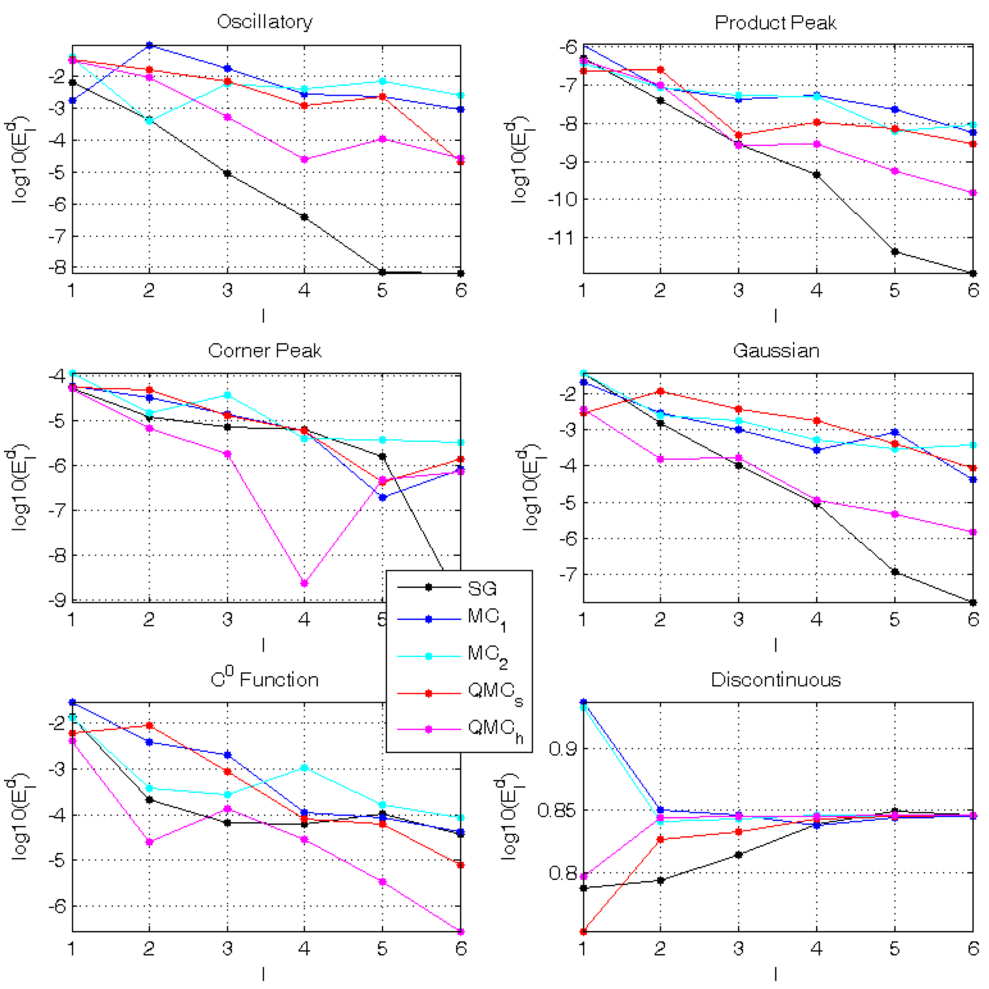

Figure 3.14: For $d = 10$ dimensions, we integrate the Genz test functions using: SG – Gauss-Patterson sparse grid, MC1 – Monte Carlo, MC2 – Monte Carlo, qMC$_s$ – quasi-Monte Carlo Sobol sequence, qMC$_h$ quasi-Monte Carlo Halton sequence, for level $l = [1, 2, 3, 4, 5, 6]$, which corresponds to a total number of points $n_l^d = [21, 221, 1581, 8801, 41265, 171425]$. The (q)MC results are based on the respective sparse grid point count. Note the very good performance of the sparse grid rule.

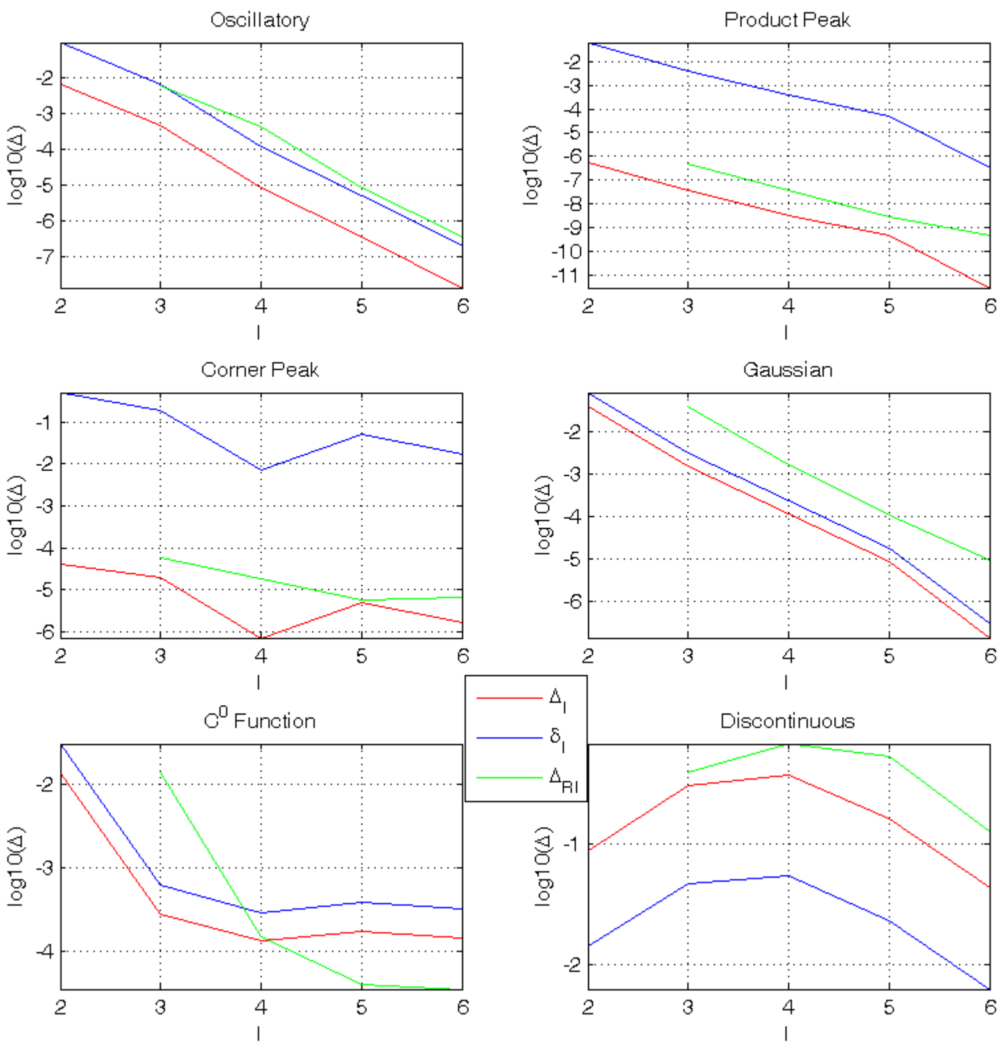

Figure 3.15: For the same set up as in Figure 3.14, we show the different candidates for the termination criterion. Note the similar behaviour.

Chapter 4

Algorithm

Below, the main steps performed in our program are summarized. The procedure is similar in the (q)MC-case.

Sparse Grid FE Stochastic Collocation:

```
• Define scenario (assign admissible regions to the
  molecules), select quadrature rule and termination
  criterion.
• Repeat until integration converges:
   1. Calculate sparse grid quadrature points and,
      if there are any, incorporate old results to
      exploit nesting.
      Then, collocate in the stochastic domain:
     (a) Place molecules according to quadrature
         point and generate mesh.
     (b) Compute FE-approximation in the spatial
         domain by solving the linear as well as the
         nonlinear Poisson-Boltzmann equation.
     (c) Calculate current in transducer.
   2. Use quadrature rule to estimate expected value
      and variance of the current.
   3. Increment level if termination criterion is not
      satisfied.
• Post-processing, i.e., analysing the impact of the
  number of molecules and their respective position
  and/or angle on the current.
```

Chapter 5

Numerical Results

Before we present the numerical results, some remarks regarding the investigated scenarios are in order. As already indicated in Section 2.1.2, we assign to each molecule a certain region on the boundary layer. These regions, intervals in our case, are chosen such that the molecules do not overlap after rotations or translations. We create scenarios by placing different numbers of molecules in each of the five regions, which we index from left to right, of the boundary layer (recall Figure 2.1). Using a row vector M, we can describe the situation by writing $M = (m_1, m_2, m_3, m_4, m_5)$, where the i^{th} entry denotes the number of molecules occupying the i^{th} face. Let d be the number of stochastic dimensions; we have $d = 2(m_1 + m_2 + m_3 + m_4 + m_5)$, since every molecule has two degrees of freedom. To keep the number of dimensions as low a possible, we restrict ourselves to ssDNA strands of equal size. A scenario is then defined by prescribing the molecule type, i.e., the number of base pairs n, the ion concentration κ_E as well as the m_i. The considered cases are as follows:

- **Scenario 1**: $M = m(0,1,1,1,0)$, with $m = 1,2,3(,4)$, $\kappa_E = 10\,\text{mM}, 30\,\text{mM}, 100\,\text{mM}$ and $n = 12, 20$.
- **Scenario 2**: $M = m(1,1,1,1,1)$, with $m = 1,2,3,4$ and $\kappa_E = 10\,\text{mM}$ and $n = 12$.

Each scenario is then investigated by means of the numerical methods developed in Chapter 3. Figure 5.1 illustrates scenario 2 for $m = 1,2,3$. To get a better understanding for cost of the stochastic collocation procedure, Table 5.1 gives the point count of the Gauss-Patterson sparse grid rule (which we will from now on abbreviate by SG) used in the simulations. We mention that scenario 2 is only considered in the beginning to obtain information regarding the anisotropy of the problem. Later on we focus on Scenario 1 to reduce the enormous computational effort.

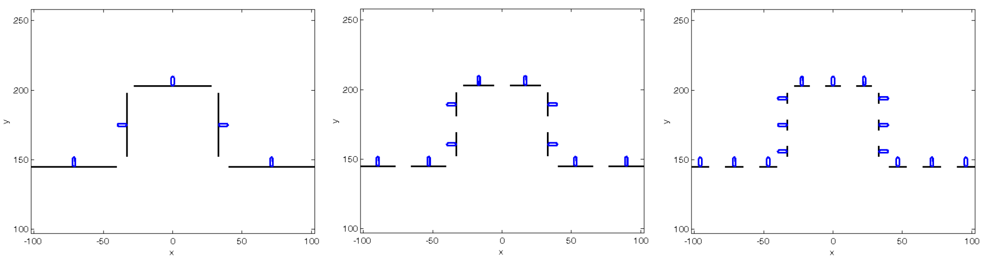

Figure 5.1: Exemplary distribution of molecules in scenario 2 (from left to right: $m = 1,2,3$). The black lines indicate the admissible regions in which the molecules reside.

		Level l					
m	d	1	2	3	4	5	6
1	6	13	97	545	2561	10625	40193
2	12	25	337	3249	25089	164865	956929
3	18	37	721	9841	105601	948289	
4	24	49	1249	22049	302849		

Table 5.1: The number of grid points for a given number of molecules m per admissible region and level l of the sparse grid rule (Gauss-Patterson) in Scenario 1. The number is of course far larger in scenario 2.

In Section 2.1.2, we outlined that due to the surface charge, the angles of the molecules are distributed according to certain probability density functions p that were determined experimentally. Following the arguments in Section 3.2, it is advantageous to use a quadrature rule which is defined with respect to this weight function p. However, the derivation of a nested quadrature rule for an arbitrary weight function is a non-trivial task and we therefore resort to the following strategy. First, we assume a uniform distribution for which our developed methods are directly applicable. Then we compare these results to (q)MC simulations, for which it is rather simple (at least in our case) to create random numbers distributed according to the weight function p. We shall denote these methods by w(q)MC. To generate the random numbers, we apply the well-known *inversion method.*

Inversion Method:
Let X be a continuous random variable with probability density function (pdf) f_X. The cumulative distribution function (cdf) F_X is given by $F_X(x) = \int_{-\infty}^{x} f(t)\,dt$. Clearly we have $F_X : \mathbb{R} \mapsto [0,1]$ and note, that $Y = F_X(X)$ is uniformly distributed on $[0,1]$. Therefore, if Y is uniformly distributed on $[0,1]$ and X as above, then the cdf of the random variable given by $F_X^{-1}(Y)$ is F_X. Since number generators

for uniformly distributed numbers are widely available, this idea can be used to obtain numbers for arbitrary distributions given that one can invert the cdf. To summarize, one performs the following steps:

1. Generate uniformly distributed (quasi) random number $y \in [0,1]$.
2. Compute x, such that $F_X(x) = y$. x is then distributed according to f_X.

In our case, we invert a piecewise linear approximation of the cdf F_X. Note that

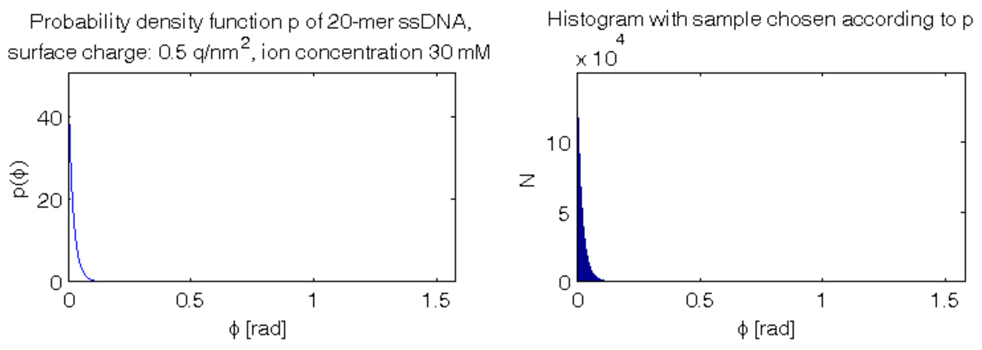

Figure 5.2: Probability density function of the angles of a ssDNA strand with 20 base pairs in a solute with an ion concentration of 30 mM, according to [16]. The histogram shows a random sample produced using the inversion method.

for numerical reasons we do not include the (constant) surface charge, which is no severe restriction, since we are primarily interested in the fluctuations.

Since stochastic collocation requires solving the underlying problem at a prescribed set of nodes, we included an example. In the first step, see Figure 5.3, the molecules are placed according to the collocation point and in the second step, see Figure 5.4, the respective solution u of the PBE equation is calculated to compute the current I through the transducer.

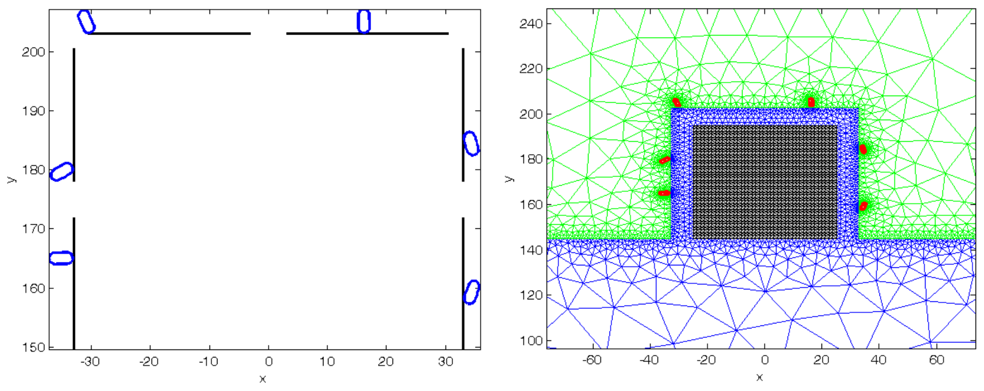

Figure 5.3: A certain molecule distribution corresponding to scenario 1 with $m = 2$ and $n = 12$. On the left hand side we see a sketch of illustrating how the molecules are placed and on the right hand side we see a part of the unstructured mesh generated for the finite-element method.

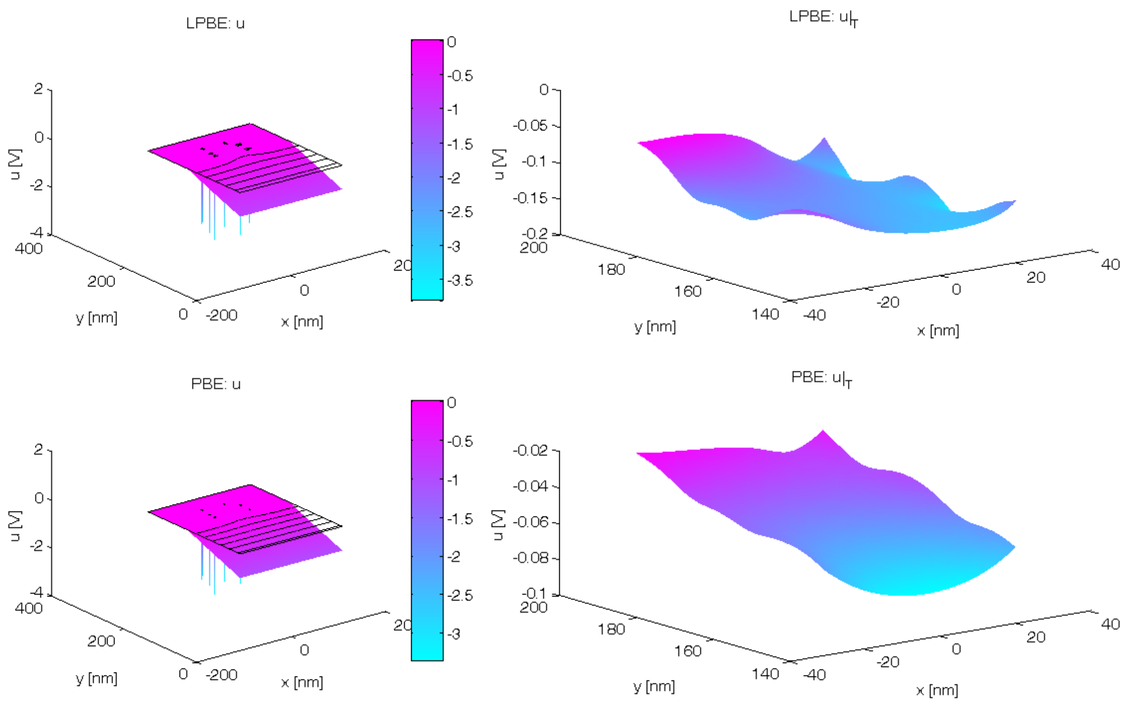

Figure 5.4: Left: The solution u of the LPBE (top) as well as of the PBE (bottom). Right: the solution in the cross section of the transducer only. The 'bumps' in the electrostatic potential close to the boundary stem from the molecules. Note the clear difference between the two solutions. In the linearised case, the the effect of the molecules is more pronounced, whereas in the non-linear case their electric field is screened to a larger extent. In this case we obtain $I_{LPBE} = 1.0807 \cdot 10^{-11}$ A and $I_{PBE} = 4.8226 \cdot 10^{-13}$ A.

5.1 Comparison of Scenario 1 and 2

We begin with a comparison of scenarios 1 and 2. The ion concentration in this case is 10 mM and we have molecule consisting of $n = 12$ base pairs. Figure 5.5 shows the current I in the transducer for both scenarios.

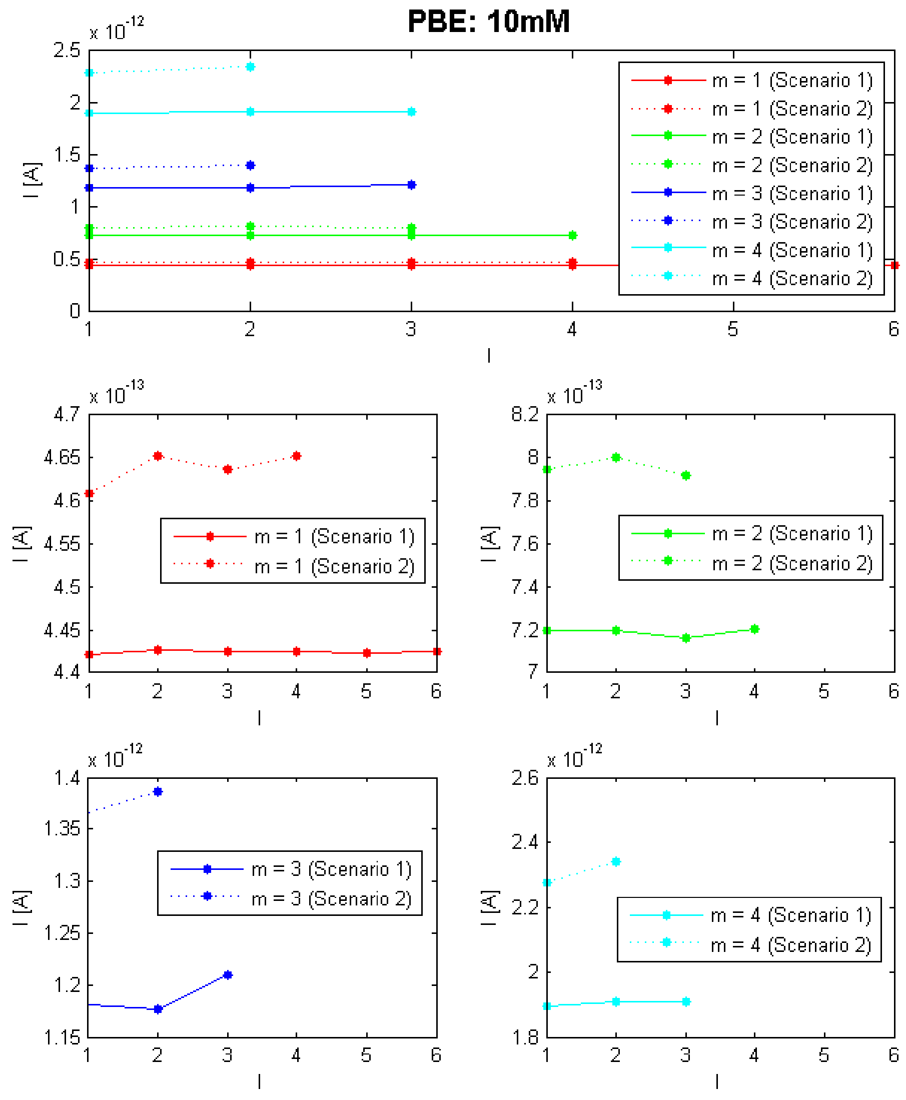

Figure 5.5: PBE, SG results: We observe fast convergence and similar results for small m. This indicates, that the molecules that reside to the left and right hand side of the transducer have a lesser impact on the current, implying that our problem is anisotropic. However, as m grows, more of these molecules are located closer to the transducer and the effect diminishes.

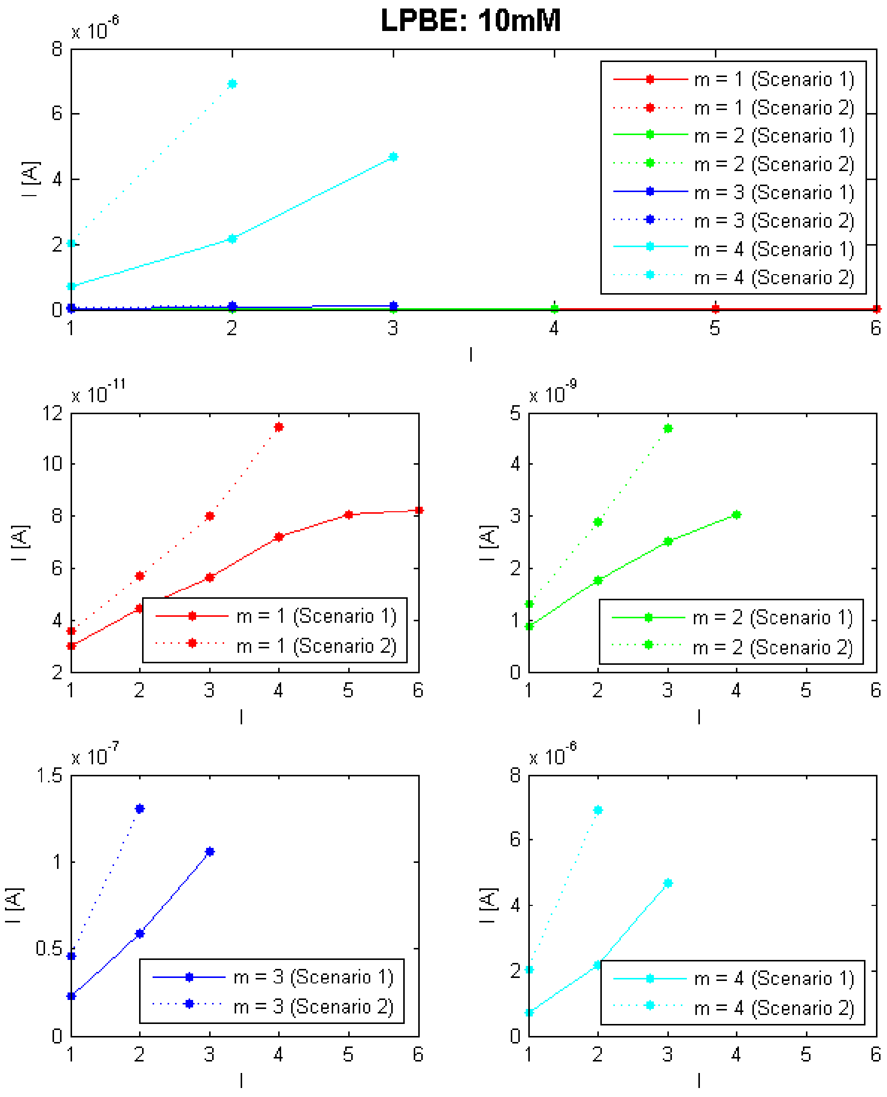

Figure 5.6: LPBE, SG results: The difference to the non-linearised case in Figure 5.5 is obvious, although the effect is exaggerated due to the far larger current for $m = 4$ which makes scaling difficult. We see that the results have not yet converged. Note that for increasing m, the quadrature rule places more points near the boundary of the integration domain, see Figure 3.8, and therefore more samples with molecules lying almost flat on the boundary surface are considered. Since in the linearised case the effect of screening is reduced, the current increases much faster than in the non-linear case and higher levels are necessary to resolve this behaviour. Compare the speed of convergence for $m = 1$ in both cases.

5.2 Scenario 1 – PBE: 10 mM and 30 mM

For the remainder of this chapter we only consider scenario 1 and start with the results for an ion concentration of 10 mM and ssDNA strands with 12 base pairs.

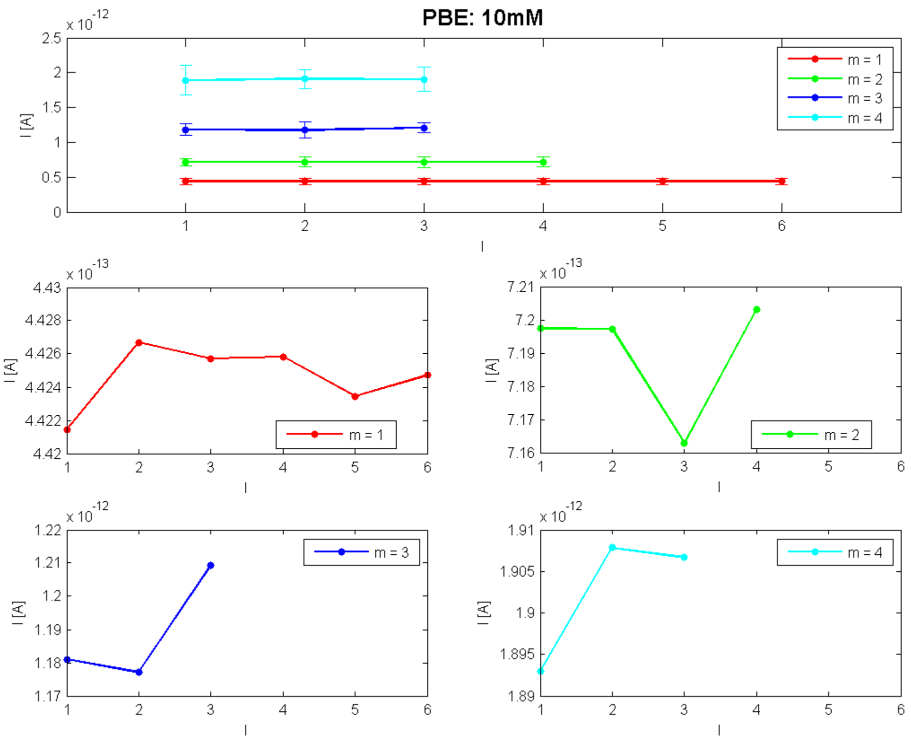

Figure 5.7: PBE, SG results: We observe fast convergence and have no difficulty in distinguishing the cases relating to different *m*, highlighting the high sensitivity of the device. The error bars showing the standard deviation indicate that the fluctuations are comparatively small, the likely reason being the screening of the molecules.

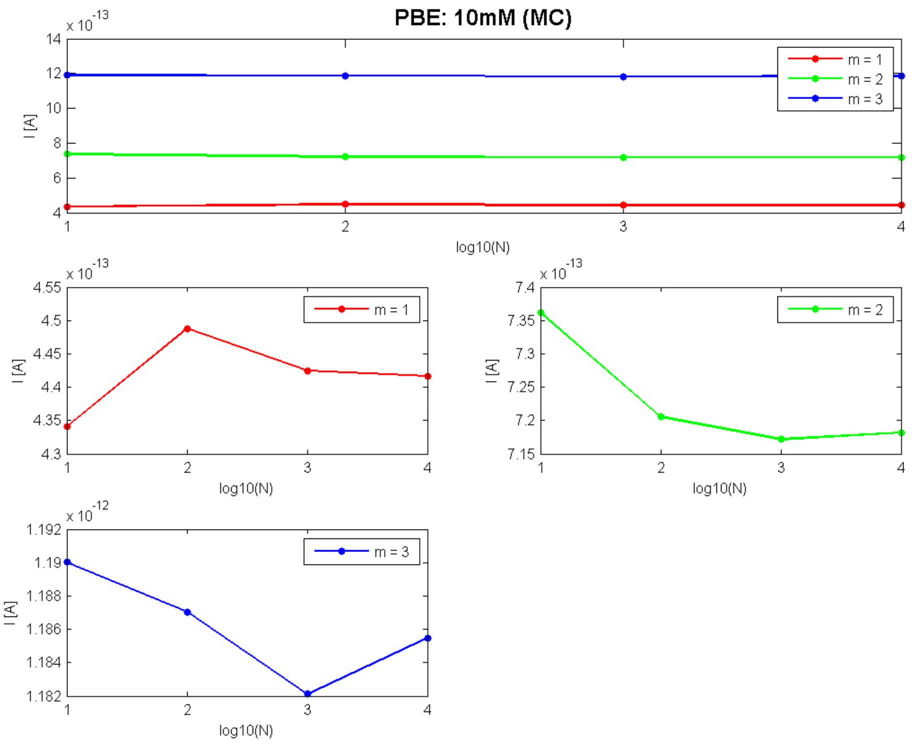

Figure 5.8: PBE, MC results: The situation is comparable to the one already described in Figure 5.7, note in particular the very similar values for the current. The convergence in the sparse grid case is faster, at least in lower dimensions.

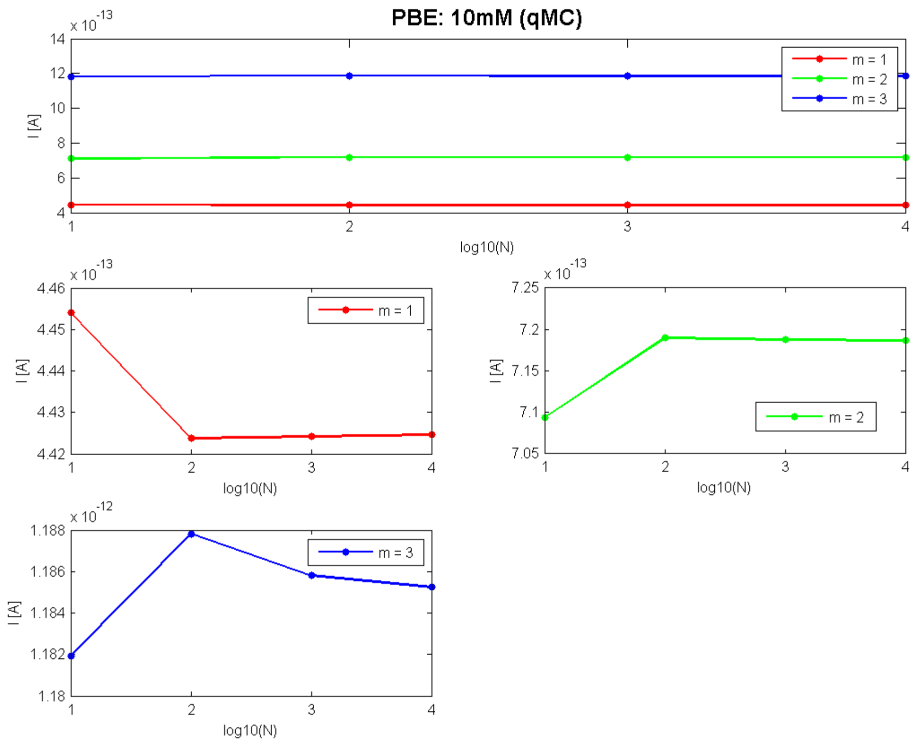

Figure 5.9: PBE, qMC results: The situation is comparable to the one already described in Figure 5.7, note in particular the very similar values for the current. Convergence is certainly faster than in the MC case, which was to be expected, and is even slightly faster than in the sparse grid case.

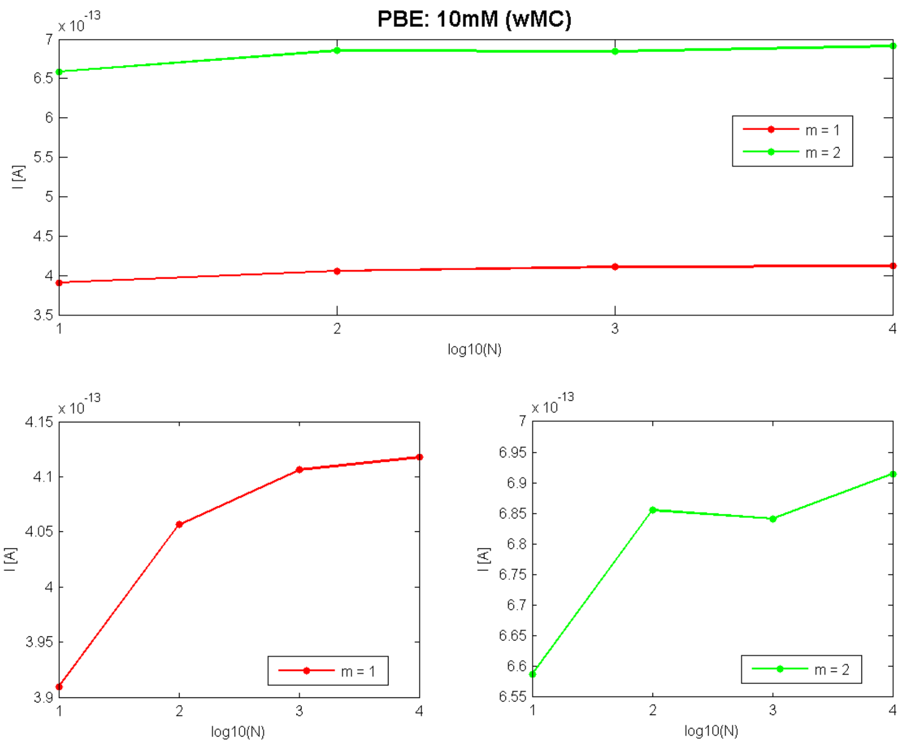

Figure 5.10: PBE, wMC results: As discussed above, we now observe regions which are sampled with very low probability. Using a sample chosen according to the calculated probability density, we observe a difference of about 5% compared with the MC case. This effect is not more pronounced, since molecules that stay away from the surface have already a lesser impact due to screening.

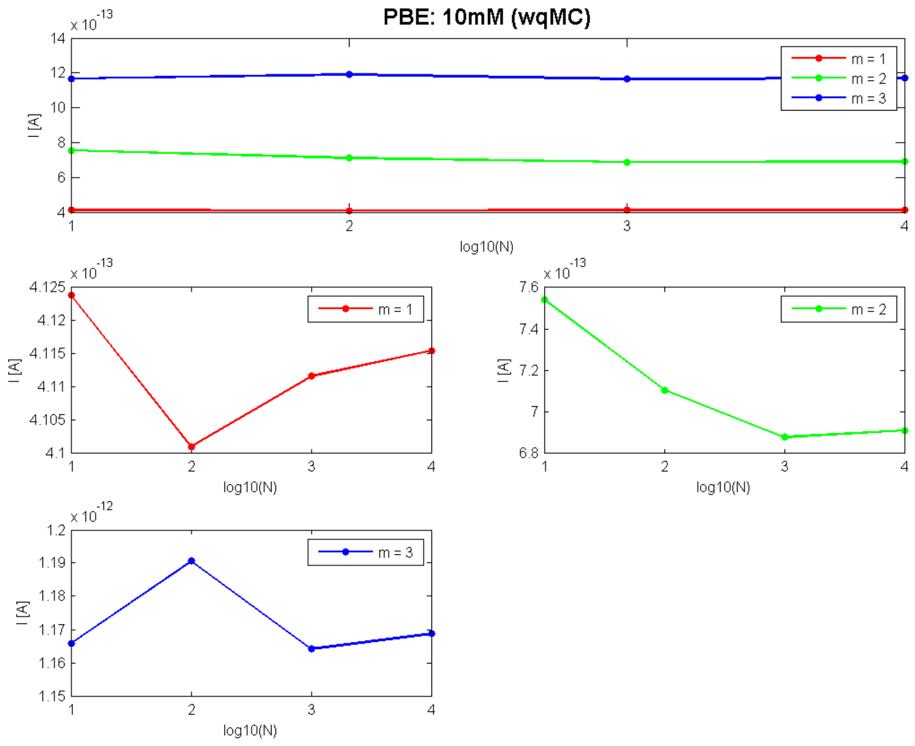

Figure 5.11: PBE, wqQM results: For the results are, besides the faster convergence, very similar and we refer to the discussion in Figure 5.10.

In the following, we step up the ion concentration to 30 mM and now consider ssDNA strands with 20 base pairs.

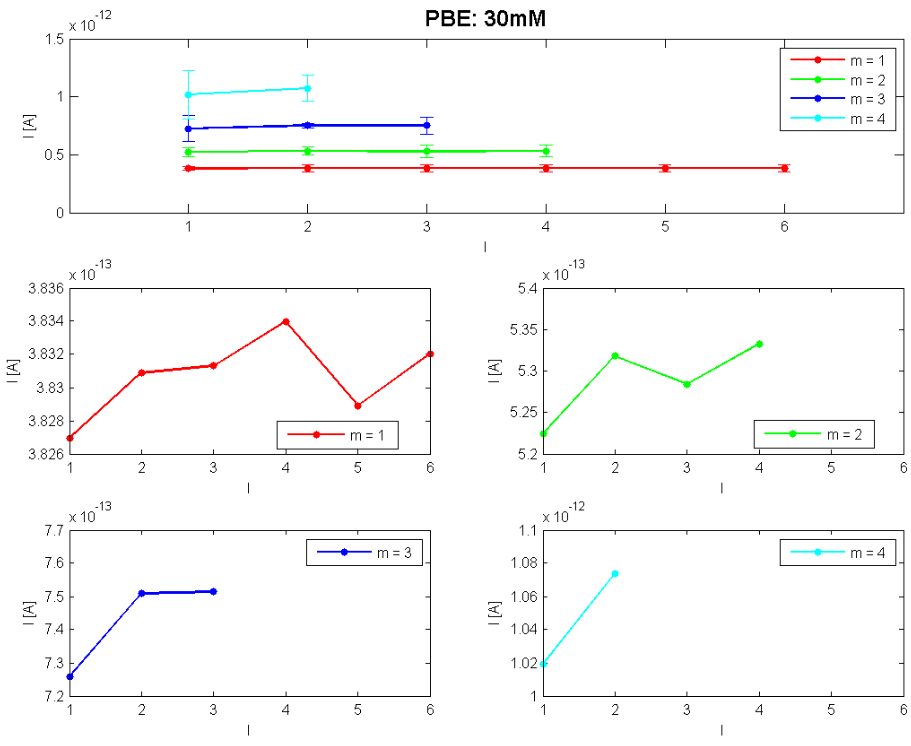

Figure 5.12: PBE, SG results: We again observe quick convergence and have no difficulty in distinguishing the cases relating to different *m*, highlighting the high sensitivity of the device also for higher ion concentrations. Note however, that due to the higher concentration, the current is now significantly smaller than in the 10 mM case depicted in Figure 5.7.

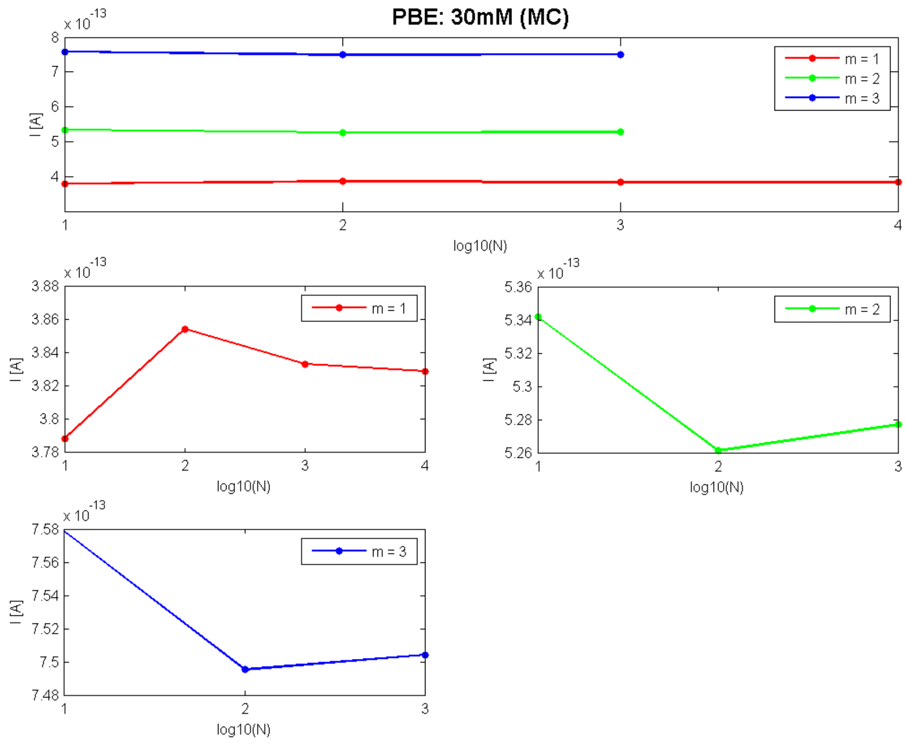

Figure 5.13: PBE, MC results: Again the results agree very well with the ones presented in Figure 5.12, note in particular the very similar values for the current.

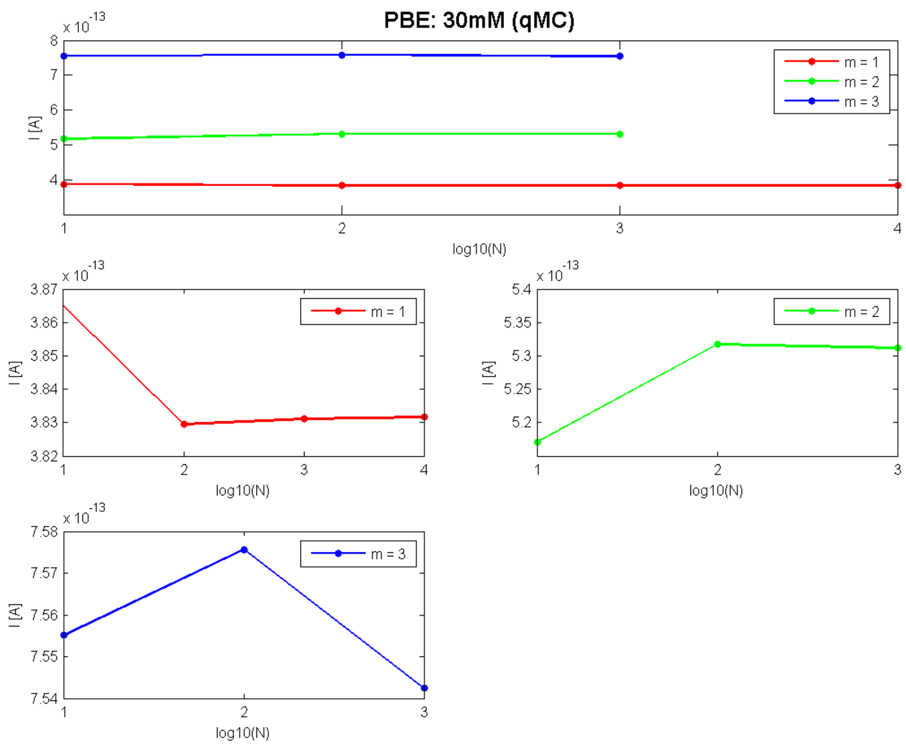

Figure 5.14: PBE, qMC results: The situation is comparable to the one already described in Figure 5.12, note in particular the very similar values for the current. Convergence is certainly faster than in the MC case, which can be expected, and comparable to the sparse grid case.

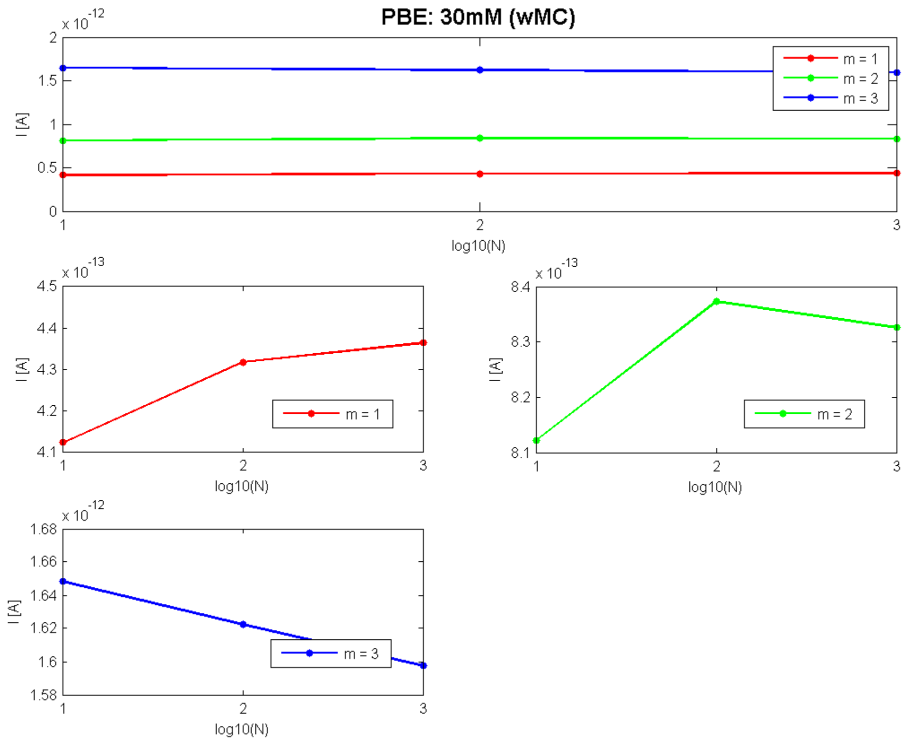

Figure 5.15: PBE, wMC results: As can be seen from Figure 5.2, there is a region where the probability density function is almost zero, restricting the set of admissible angels. Therefore, the difference in the current is now clearly visible. Note further, that the current is larger here than in Figure 5.13, which is reasonable since we only consider orientations, i.e., the molecules lie flat, which have a large impact on the current.

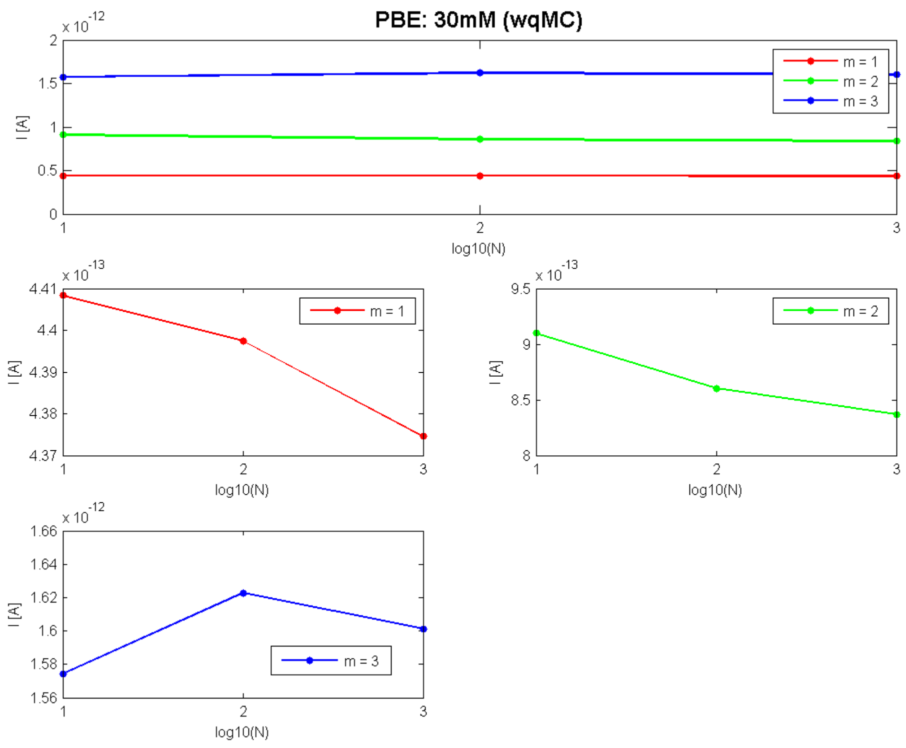

Figure 5.16: PBE, wqQM results: We refer to Figure 5.15 for the discussion therein.

5.3 Scenario 1 – LPBE: 100 mM

As already illustrated and discussed in Figure 5.6, the results in the linearised differ greatly from the more realistic non-linear case. However, the situation changes when we consider a much larger ion concentration of 100 mM. The molecules we are again ssDNA strands consisting of 12 base pairs. Figure 5.17 shows the results for the non-linear equation and the sparse grid approximation whereas Figure 5.18 shows the results for the linear equation.

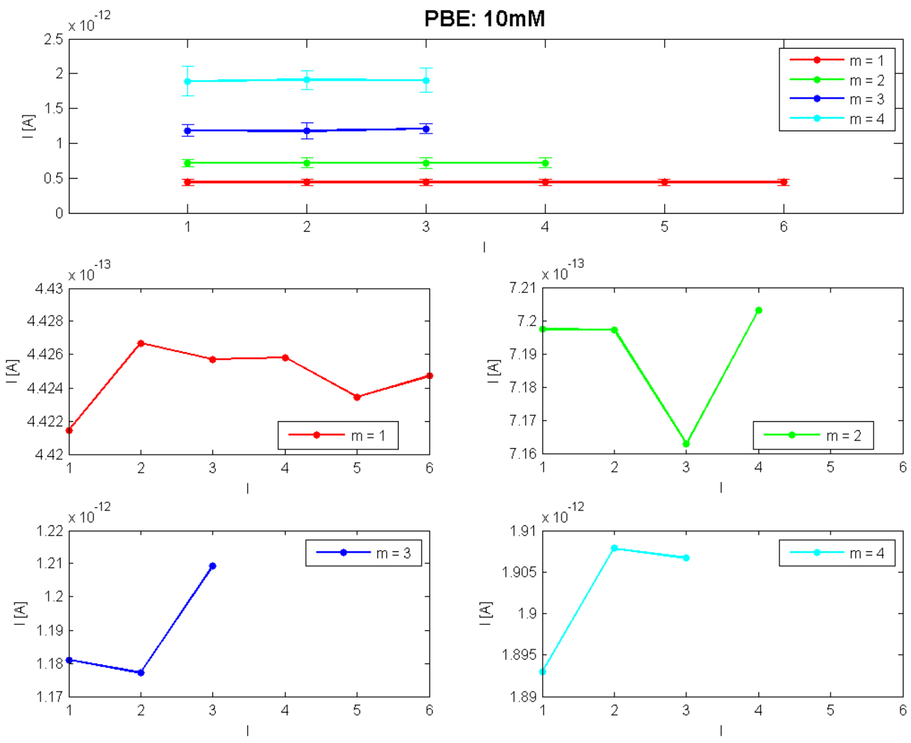

Figure 5.17: PBE, SG results: In the nonlinear case, we observe a behaviour similar to the one already discussed above in the other examples, see for example the Figures 5.7 or 5.12.

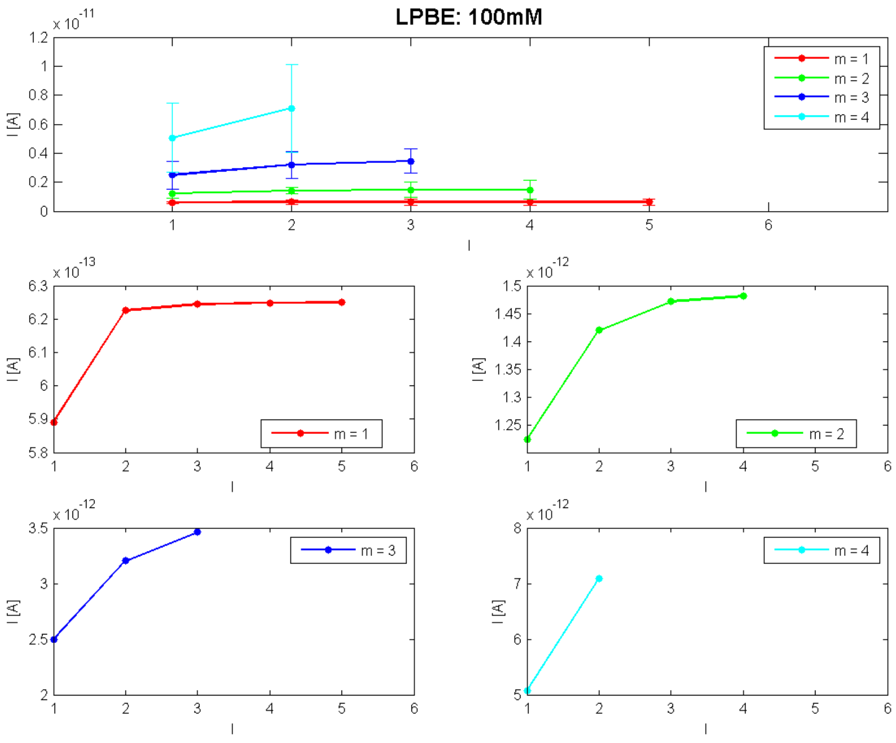

Figure 5.18: LPBE, SG results: In the linearised case, in contrast to the lower ion concentration of 10 mM depicted in Figure 5.6, we now indeed observe convergence, justifying the arguments stated there. For such a large ion concentration, the molecules are effectively screened also in the linearised case, mitigating the dependency on the angle and allowing faster convergence already for lower levels.

Chapter 6

Conclusion

In this thesis, we aimed to enhance numerical simulation methods for nanowire sensors by estimating the impact of uncertainties (or noise and fluctuations) in the model. To this end, we presented two different non-intrusive methods for uncertainty quantification: Besides the classical (q)MC approach, we also considered a stochastic collocation scheme based on the Smolyak algorithm for tensor product problems.
The numerical results show a very good agreement between the different methods and prove the Smolyak sparse grid quadrature to be superior to the classical sampling schemes, at least for a moderate number of dimensions. Since the method is non-intrusive, it can be parallelised without further modifications, as has been done here, and provides an effective framework to deal with the stochastic linear and nonlinear Poisson-Boltzmann equations. We paid particular attention to the more involved non-linear case and the very different behaviour of the solution highlights the importance of addressing the problem in a proper fashion.

Bibliography

[1] Jochen Alberty, Carsten Carstensen, and Stefan A. Funken. Remarks around 50 lines of Matlab: short finite element implementation. *Numer. Algorithms*, 20(2-3):117–137, 1999.

[2] Ivo Babuška, Fabio Nobile, and Raúl Tempone. A stochastic collocation method for elliptic partial differential equations with random input data. *SIAM Rev.*, 52(2):317–355, 2010.

[3] Stefan Baumgartner and Clemens Heitzinger. Existence and local uniqueness for 3d self-consistent multiscale models for field-effect sensors. *Commun. Math. Sci.*, 10(2):693–716, 2012.

[4] Stefan Baumgartner, Clemens Heitzinger, Aleksandar Vacic, and Mark A. Reed. Predictive simulations and optimization of nanowire field-effect PSA sensors including screening. *Nanotechnology*, pages 1–16. *In review*.

[5] Stefan Baumgartner, Martin Vasicek, Alena Bulyha, and Clemens Heitzinger. Optimization of nanowire DNA sensor sensitivity using self-consistent simulation. *Nanotechnology*, 22(42):425503/1–8, October 2011.

[6] Stefan Baumgartner, Martin Vasicek, and Clemens Heitzinger. Modeling and simulation of nanowire based field-effect biosensors. In G. Korotcenkov, editor, *Chemical Sensors: Simulation and Modeling*, pages 447–469. Momentum Press, 2012.

[7] Christopher Anand. Continuous Optimization Algorithms: http://www.cas.mcmaster.ca/∼cs4te3/notes/GA_principle.pdf, accessed March 27, 2013.

[8] Charles Clenshaw and Alan Curtis. A method for numerical integration on an automatic computer. *Numerische Mathematik*, 2(1):197–205, December 1960.

[9] Yi Cui, Qingqiao Wei, Hongkun Park, and Charles M. Lieber. Nanowire nanosensors for highly sensitive and selective detection of biological and chemical species. *Science*, 293(5533):1289–1292, 2001.

[10] Philip J. Davis and Philip Rabinowitz. *Methods of numerical integration. Corrected reprint of the 1984 2nd ed.* Mineola, NY: Dover Publications, 2007.

[11] Herbert Edelsbrunner. *Geometry and topology for mesh generation.* Cambridge: Cambridge University Press, 2001.

[12] Fogolari, F. and Brigo, A. and Molinari, H. The PoissonBoltzmann equation for biomolecular electrostatics: a tool for structural biology. *Journal of Molecular Recognition*, 15(6):377–392, 2002.

[13] Alan Genz. *A Package for Testing Multiple Integration Subroutines*, volume 203 of *NATO ASI Series*. Springer Netherlands, 1987.

[14] T. Gerstner and M. Griebel. Dimension-adaptive tensor-product quadrature. *Computing*, 71(1):65–87, 2003.

[15] Thomas Gerstner and Michael Griebel. Numerical integration using sparse grids. *Numer. Algorithms*, 18(3-4):209–232, 1998.

[16] Clemens Heitzinger, Yang Liu, Norbert Mauser, Christian Ringhofer, and Robert W. Dutton. Calculation of fluctuations in boundary layers of nanowire field-effect biosensors. *J. Comput. Theor. Nanosci.*, 7(12):2574–2580, 2010.

[17] Clemens Heitzinger, Norbert Mauser, and Christian Ringhofer. Multiscale modeling of planar and nanowire field-effect biosensors. *SIAM J. Appl. Math.*, 70(5):1634–1654, 2010.

[18] Clemens Heitzinger and Christian Ringhofer. Multiscale modeling of fluctuations in stochastic elliptic PDE models of nanosensors. *Commun. Math. Sci.*, pages 1–21, 2013. *At press*.

[19] Edmund Hlawka. Funktionen von beschränkter Variation in der Theorie der Gleichverteilung. *Ann. Mat. Pura Appl., IV. Ser.*, 54:325–333, 1961.

[20] John Burkard. Sparse Grid Collocation for Uncertainty Quantification: http://people.sc.fsu.edu/∼jburkardt/presentations/ornl_2012.pdf, accessed March 27, 2013.

[21] A.S. Kronrod. *Nodes and weights of quadrature formulas. Sixteen-place tables. Authorized translation from the Russian.* 1965.

[22] Meilin Liu, Zhen Gao, and Jan S. Hesthaven. Adaptive sparse grid algorithms with applications to electromagnetic scattering under uncertainty. *Applied Numerical Mathematics*, 61(1):24 – 37, 2011.

[23] Harold Niederreiter. *Random number generation and quasi-Monte Carlo methods.* Philadelphia, PA: SIAM, 1992.

[24] Fabio Nobile, Raul Tempone, and Clayton G. Webster. A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM J. Numer. Anal.*, 46(5):2309–2345, 2008.

[25] Fabio Nobile, Raul Tempone, and Clayton G. Webster. An anisotropic sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM J. Numer. Anal.*, 46(5):2411–2442, 2008.

[26] Erich Novak and Klaus Ritter. High dimensional integration of smooth functions over cubes. *Numer. Math.*, 75(1):79–97, 1996.

[27] Thomas Patterson. The optimal addition of points to quadrature formulae. *Mathematics of Computation*, 22(104):847–856, October 1968.

[28] Per-Olof Persson. Mesh Generation for Implicit Geometries, Ph.D. thesis, http://persson.berkeley.edu/thesis/persson-thesis-color.pdf, accessed March 09, 2013.

[29] Per-Olof Persson and Gilbert Strang. A simple mesh generator in MATLAB. *SIAM Rev.*, 46(2):329–345, 2004.

[30] Manuel Punzet, Dieter Baurecht, Franz Varga, Heidrun Karlic, and Clemens Heitzinger. Determination of surface concentrations of individual molecule-layers used in nanoscale biosensors by in-situ ATR-FTIR spectroscopy. *Nanoscale*, 4(7):2431–2438, 2012.

[31] Sheldon M. Ross. *Stochastic processes. 2nd ed.* New York, NY: John Wiley; Sons. xvi, 510, 1996.

[32] Eric Stern, Robin Wagner, Fred J. Sigworth, Ronald Breaker, Tarek M. Fahmy, and Mark A. Reed. Importance of the debye screening length on nanowire field effect transistor sensors. *Nano Letters*, 7(11):3405–3409, 2007.

[33] Josef Stoer and Roland Bulirsch. *Numerical mathematics 2. An introduction – under consideration of lectures by F. L. Bauer. (Numerische Mathematik 2. Eine Einführung – unter Berücksichtigung von Vorlesungen von F. L. Bauer.) 5th ed.* Berlin: Springer, 2005.

[34] Suli, Endre. Finite Element Methods for Partial Differential Equations. Lecture notes: http://people.maths.ox.ac.uk/suli/fem.pdf, accessed March 27, 2013.

[35] Bozhi Tian, Tzahi Cohen-Karni, Quan Qing, Xiaojie Duan, Ping Xie, and Charles M. Lieber. Three-dimensional, flexible nanoscale field-effect transistors as localized bioprobes. *Science*, 329(5993):830–834, 2010.

[36] Lloyd N. Trefethen. Is Gauss quadrature better than Clenshaw-Curtis? *SIAM Rev.*, 50(1):67–87, 2008.

[37] Grzegorz W. Wasilkowski and Henryk Woźniakowski. Explicit cost bounds of algorithms for multivariate tensor product problems. *J. Complexity*, 11(1):1–56, 1995.

[38] Wikipedia. Delaunay Triangulation:
http://en.wikipedia.org/wiki/Delaunay_triangulation, accessed March 27, 2013.

[39] Dongbin Xiu. *Numerical methods for stochastic computations. A spectral method approach.* Princeton, NJ: Princeton University Press., 2010.

LEBENSLAUF

PERSÖNLICHE DATEN

Name:	**Claus Aichinger**
Geburtsdaten:	08.10.1985, in Wien
Staatsbürgerschaft:	Österreich

AUSBILDUNG

2006 – dato	**Diplomstudium Mathematik** (Applied Mathematics and Scientific Computing) UNIVERSITÄT WIEN • Auslandsjahr an der Universidad Autónoma de Madrid
2006 – dato	**Diplomstudium Physik** (Computational Physics) UNIVERSITÄT WIEN
2006 – 2007	Sprachstudium Chinesisch UNIVERSITÄT WIEN
2000 – 2005	**HTL Mechatronik (Automatisierungstechnik) mit Matura** HTL WIEN ETTENREICHGASSE, 1100 WIEN