*Entropy* **2013**, *15*, 3640-3687; doi:10.3390/e15093640

OPEN ACCESS

*entropy*

**ISSN 1099-4300**

www.mdpi.com/journal/entropy

*Article*

# SpaGrOW—A Derivative-Free Optimization Scheme for Intermolecular Force Field Parameters Based on Sparse Grid Methods

**Marco Hülsmann** [1,*] **and Dirk Reith** [2]

[1] Fraunhofer Institute for Algorithms and Scientific Computing (SCAI), Schloss Birlinghoven, Sankt Augustin 53757, Germany

[2] Hochschule-Bonn Rhein-Sieg (HBRS), Grantham-Allee 20, Sankt Augustin 53757, Germany; E-Mail: dirk.reith@h-brs.de

* Author to whom correspondence should be addressed; E-Mail: marco.huelsmann@scai.fraunhofer.de; Tel.: +49-2241-14-2053; Fax: +49-2241-14-2656.

*Received: 16 February 2013; in revised form: 15 July 2013 / Accepted: 28 August 2013 / Published: 6 September 2013*

---

**Abstract:** Molecular modeling is an important subdomain in the field of computational modeling, regarding both scientific and industrial applications. This is because computer simulations on a molecular level are a virtuous instrument to study the impact of microscopic on macroscopic phenomena. Accurate molecular models are indispensable for such simulations in order to predict physical target observables, like density, pressure, diffusion coefficients or energetic properties, quantitatively over a wide range of temperatures. Thereby, molecular interactions are described mathematically by force fields. The mathematical description includes parameters for both intramolecular and intermolecular interactions. While intramolecular force field parameters can be determined by quantum mechanics, the parameterization of the intermolecular part is often tedious. Recently, an empirical procedure, based on the minimization of a loss function between simulated and experimental physical properties, was published by the authors. Thereby, efficient gradient-based numerical optimization algorithms were used. However, empirical force field optimization is inhibited by the two following central issues appearing in molecular simulations: firstly, they are extremely time-consuming, even on modern and high-performance computer clusters, and secondly, simulation data is affected by statistical noise. The latter provokes the fact that an accurate computation of gradients or Hessians is nearly impossible close to a local or global minimum, mainly because the loss function is flat. Therefore, the question arises of whether to apply a derivative-free method approximating

the loss function by an appropriate model function. In this paper, a new Sparse Grid-based Optimization Workflow (SpaGrOW) is presented, which accomplishes this task robustly and, at the same time, keeps the number of time-consuming simulations relatively small. This is achieved by an efficient sampling procedure for the approximation based on sparse grids, which is described in full detail: in order to counteract the fact that sparse grids are fully occupied on their boundaries, a mathematical transformation is applied to generate homogeneous Dirichlet boundary conditions. As the main drawback of sparse grids methods is the assumption that the function to be modeled exhibits certain smoothness properties, it has to be approximated by smooth functions first. Radial basis functions turned out to be very suitable to solve this task. The smoothing procedure and the subsequent interpolation on sparse grids are performed within sufficiently large compact trust regions of the parameter space. It is shown and explained how the combination of the three ingredients leads to a new efficient derivative-free algorithm, which has the additional advantage that it is capable of reducing the overall number of simulations by a factor of about two in comparison to gradient-based optimization methods. At the same time, the robustness with respect to statistical noise is maintained. This assertion is proven by both theoretical considerations and practical evaluations for molecular simulations on chemical example substances.

**Keywords:** force field parameterization; molecular simulations; atomistic models; derivative-free optimization; sparse grids; smoothing procedures

**Classification: PACS** 34.20.Gj; 64.75.Gh

## 1. Introduction

In the last few decades, computer simulations have gained in importance for both science and industry, particularly due to the fast development of parallel high performance clusters. A denotative subarea of computer simulations are molecular simulations, which allow one to study the effects of modifications in microscopic states on macroscopic system properties. In contrast to simulations in process engineering, not the continuum, but the molecular level of a system is modeled. Thereby, the goal is to describe interatomic and intermolecular interactions, so that, on the one hand, certain accuracy demands are fulfilled, and on the other hand, the required computation time is as low as possible. The latter aspect is very important, because molecular simulations are numerically costly, also on modern computer clusters. The industrial relevance of molecular simulations originates from the fact that labor- and cost-intensive chemical experiments can be avoided. Hence, chemical systems can be simulated at temperatures and pressures that are very difficult to realize in a laboratory, the properties of toxic substances can be calculated without any risk and measure of precaution and processes on surfaces or within membranes are observable on a microscopic level. Another advantage of molecular simulations is that both the location and velocity of each particle are saved after certain time intervals, which results in a detailed observation of the behavior of the system. Altogether, molecular simulations have emerged as their own

scientific discipline, and because of the continuous growth of computer resources, they will still become much more important in the coming years [1,2].

In principle, it is possible to describe interactions within a chemical system by quantum mechanics. Thereby, a partial differential equation, the so-called *Schrödinger equation* [3], has to be solved. As this turns out to be extremely difficult, especially for multi-particle systems, the problem is simplified by classical mechanical methods. Then, the system is considered on an atomistic level, *i.e.*, the smallest unit is an atom. Molecular simulation techniques are based on statistical mechanics, a subarea of classical mechanics. The most important simulation methods are molecular dynamics (MD) and Monte Carlo (MC). Thereby, both intra- and inter-molecular interactions are described by the foundation of a simulation, the force field. A force field consists of an analytic term and parameters to be adjusted. It is given by the potential energy, which has the following typical form [1,4]:

$$\begin{aligned} U_{\text{pot}}(r^M) &:= \sum_{\text{Bonds}} \frac{k_r}{2}\left(r - r_0\right)^2 + \sum_{\text{Angles}} \frac{k_\phi}{2}\left(\phi - \phi_0\right)^2 + \sum_{\text{Dihedrals}} \sum_{n=1}^{m} V_n \cos(n\omega) \\ &+ \sum_{i<j}^{N} \left\{ 4\varepsilon_{ij} \left[ \left( \frac{\sigma_{ij}}{r_{ij}} \right)^{12} - \left( \frac{\sigma_{ij}}{r_{ij}} \right)^{6} \right] + \frac{q_i q_j}{4\pi\varepsilon_0 r_{ij}} \right\} \end{aligned} \tag{1}$$

Thereby, $r^M \in \mathbb{R}^{3M}$ is a vector containing all three-dimensional coordinates of the interaction sites, where $M$ is the number of particles in the system. The first row of Equation (1) describes the intramolecular and the second row the intermolecular part. The parameters of the intramolecular part, modeling the interactions caused by the modifications of bond lengths $r$, bond angles $\phi$ and dihedral angles $\omega$ can be computed by quantum mechanics. Please note that the index, $_0$, denotes the respective parameter in equilibrium and that $k_r$ and $k_\phi$ are force constants. The factors, $V_n,\ n = 1, ..., m,\ m \in \mathbb{N}$, describe the rotation barriers around the molecular axes. The parameterization of the intermolecular part, modeling the interactions caused by dispersion—described by the Lennard-Jones (LJ) parameters, $\sigma_{ij}$ and $\varepsilon_{ij},\ i, j = 1, ..., M,\ i < j$, and electrostatic effects—described by partial atomic charges, $q_i$ and $q_j,\ i, j = 1, ..., M,\ i < j$, is often tedious. The constant, $\varepsilon_0 = 8.854 \times 10^{-12}$ Fm$^{-1}$, is the dielectric constant.

### *1.1. Force Field Parameterization*

It is known from the literature that many force fields describe molecular interactions accurately, qualitatively and quantitatively [5]. Intramolecular parameters can be determined by quantum mechanics, *i.e.*, by the minimization of a potential hyperplane. Partial atomic charges can also be computed from the position of nuclei and electrons. However, quantum mechanical methods require a high computational effort, especially for large molecules. This is why some simplifications were carried out, and the adjustment of the parameters was performed by fitting them to spectroscopic data. Some of the most famous force fields based on such semiempirical methods have been developed, e.g., by [6–8]. Quantum mechanical calculations of partial atomic charges were realized, e.g., by [9]. In related work [10,11], the automized optimization workflow, *WOLF$_2$PACK*, was created, which combines quantum mechanical algorithms with atomistic models and is capable of calculating both

optimal intramolecular force field parameters and partial atomic charges. For the intermolecular part of Equation (1), the respective force field parameters were mostly adjusted to experimental target data, *i.e.*, physical properties resulting from a molecular simulation were compared with experimental reference data. In particular, this empirical approach was realized by [12–14]. However, the parameters obtained cannot be considered as optimal, because they were not fitted to a large number of experimental data. Furthermore, in most cases, they ware adjusted manually, which is always time-consuming. They are transferable to other substances, but a subsequent readjustment is indispensable [15]. Many users take standard force fields from the literature for their simulations, which may lead to satisfactory, but not to optimal, target properties.

In the last decade, a few approaches were published realizing an automated force field parameterization procedure [16–20]. Thereby, physical target properties, like density, enthalpy of vaporization and vapor pressure, were fitted to their respective experimental reference data at different temperatures and pressures simultaneously. This was done via the minimization of a quadratic loss function between simulated and experimental data, *i.e.*, by solving a mathematical optimization problem with numerical optimization algorithms. This approach is pursued in this paper, as well. The loss function to be minimized is given by:

$$F : \mathbb{R}^N \rightarrow \mathbb{R}_0^+ \tag{2}$$

$$x \mapsto \sum_{i=1}^{n} w_i^2 \left( \frac{f_i^{\mathrm{exp}} - f_i^{\mathrm{sim}}(x)}{f_i^{\mathrm{exp}}} \right)^2 \tag{3}$$

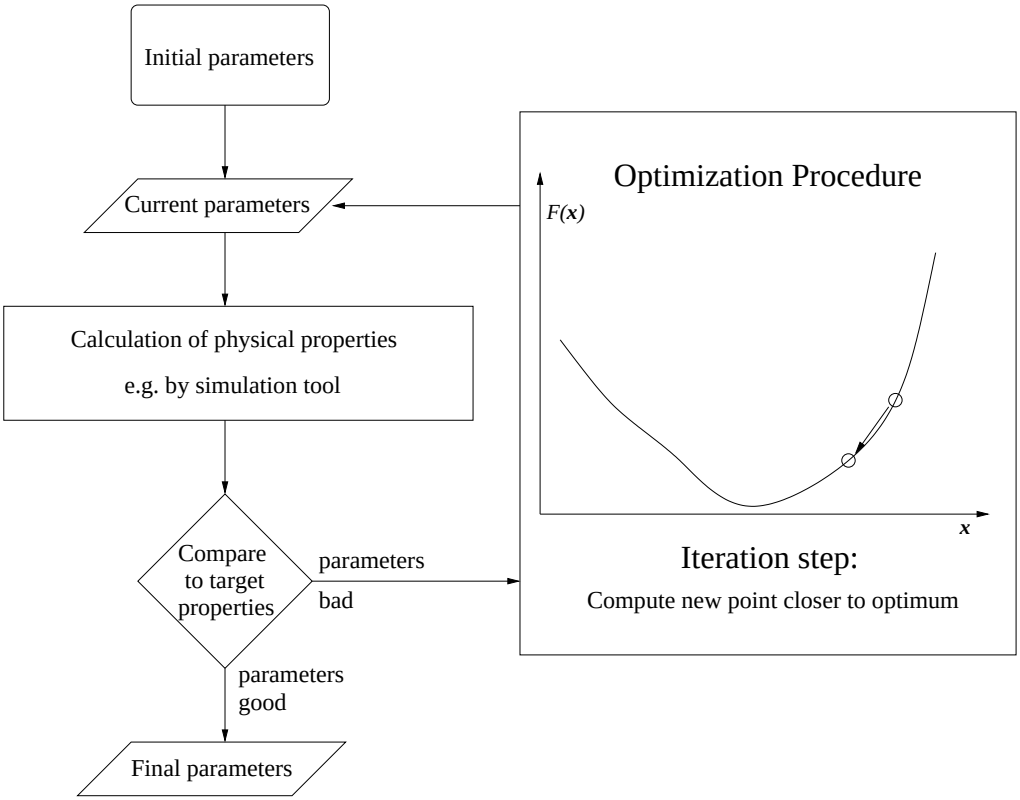
where $x = (x_1, ..., x_N)^T$ is a force field parameter vector, $N$ the dimension of the parameter space, $n$ the number of considered physical properties, maybe at different temperatures, $f_i^{\mathrm{sim}}(x),\ i = 1, ..., n$ the simulated physical target properties as functions of the parameter vector, $x$, and $f_i^{\mathrm{exp}},\ i = 1, ..., n$ the respective experimental reference data. The weights, $w_i^2,\ i = 1, ..., n$, account for the fact that some properties are easier to reproduce or measured more accurately than others. The loss function, $F$, is minimized within a compact domain, $\Omega \subset \mathbb{R}^N$.

The optimization workflow is shown in Figure 1: the initial guess has to be reasonably close to the minimum. The target properties computed by a simulation tool are inserted into loss function in Equation (3) and compared with the experimental target properties. If a specified stopping criterion is fulfilled, the parameters are final, and the workflow terminates. Otherwise, the current parameter vector is passed on to the optimization procedure searching for new parameters with a lower loss function value.

There are two main requirements for the numerical optimization algorithms solving the minimization problem: Firstly, they have to be efficient, *i.e.*, their convergency must be fast and the number of function evaluations has to be low, because for each function evaluation, time-consuming molecular simulations are needed, which can be parallelized for the different temperatures. Secondly, they have to be robust with respect to statistical noise, because simulation data is always affected by uncertainties. The reason for this is the fact that physical properties are computed by averaging over a certain time period in the case of MD simulations and over a certain number of system states in the case of MC simulations. In previous work [21–23], the software package, *Gradient-based Optimization Workflow (GROW)*, was developed. Thereby, efficient gradient-based numerical optimization algorithms were successfully applied to minimize loss function in Equation (3) in an efficient and robust way.

Thereby, simple methods, like steepest descent and conjugate gradient algorithms, turned out to be most suitable. Algorithms requiring a Hessian were too time-consuming or not reliable whenever the Hessian was assumed to be positive definite, which it was not in most cases. Gradients and Hessians were approximated using first-order finite differences. The lengths of the descent directions were computed by an Armijo step length control mechanism.

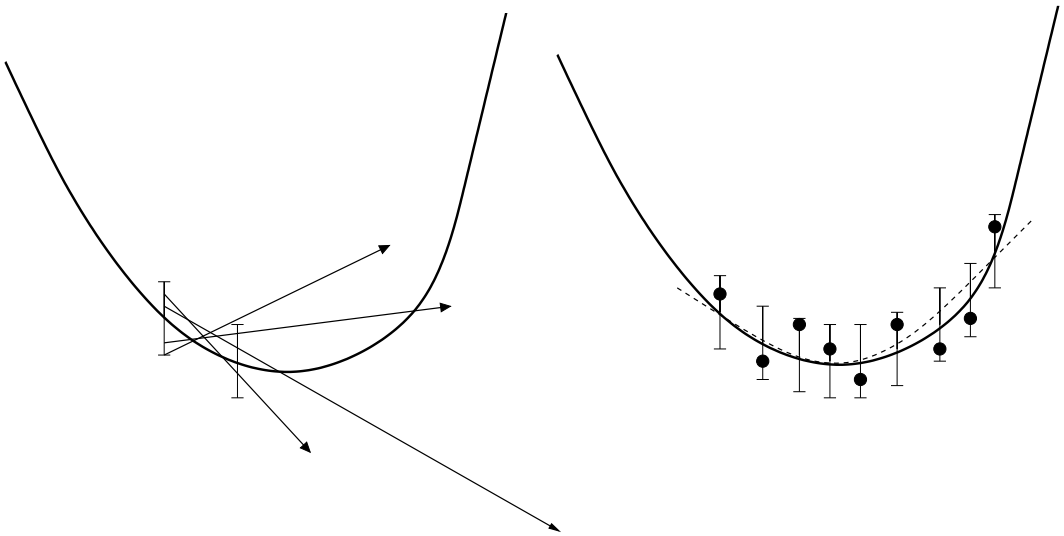
**Figure 1.** Optimization workflow: The target properties are computed for an initial guess for the force field parameters. If they do not agree sufficiently well with the experimental target properties, the optimization procedure is performed searching for new parameters with a lower loss function value.




However, the accurate computation of a gradient or a Hessian is problematic close to a global or local minimum of $F$, due to the presence of statistical noise, *cf.* Figure 2. For the finite difference approximation of the gradient, two adjacent points are necessary. If these two points are situated too close to each other, their loss function values cannot be distinguished anymore, due to the error bars surrounding them. Therefore, the direction of the approximated gradient can be completely wrong. This leads to the motivation to develop a new efficient derivative-free method counteracting this problem. The right side of Figure 2 shows that a possible solution is to approximate the loss function by an adequate model function and to determine the minimum of the model function. In this work, a new Sparse Grid-based Optimization Workflow (SpaGrOW) is presented in detail, implementing the aforesaid modeling approach. The main difficulty is to filter out the statistical noise during the approximation process, which is realized in SpaGrOW by regularization methods. As approximation is always based on sampling the parameter space and as molecular simulations are required for all selected points within the sampling process, sparse grids are involved in order to avoid high computational effort. The combination

technique developed by [24] is applied in order to perform a piecewise multilinear interpolation from a sparse grid to a full grid. As sparse grids are fully occupied at their boundaries, the loss function is artificially set to zero at the boundary by a mathematical transformation. Then, the minimum on the full grid is determined, and a back-transformation is performed. As the combination technique requires certain smoothness properties, and as the loss function cannot be assumed to be smooth, it has to be preprocessed. Hence, it is approximated by a smooth model function before. The selection of a suitable model function is done by both theoretical considerations and practical evaluations.

**Figure 2.** Motivation of a derivative-free method in the case of noisy loss function values close to the minimum; the direction of a gradient can be completely wrong. Hence, an approximation of the loss function is necessary. This regression procedure has to filter out the statistical noise.



Another very important goal of SpaGrOW is to outperform gradient-based optimization algorithms with respect to computational effort, *i.e.*, the number of molecular simulations to be performed should be reduced by about a factor of two. At the same time, it should be at least as robust as gradient-based methods. Hence, SpaGrOW should also be applicable in domains that are further away from the minimum. This can only be realized efficiently by a local consideration. The minimization of SpaGrOW is performed in a small compact trust region. As the minimum can be situated at the boundary of the trust region, the method becomes an iterative procedure. The actual minimum is accepted as a new iteration, if the piecewise multilinear model predicts the decreasing trend of the loss function in a reliable way. Hence, SpaGrOW is combined with the approach of another category of optimization algorithms, the Trust Region methods, *cf.* [25,26]. The speed of convergency can be controlled by the size of the trust region. Altogether, SpaGrOW is shown to be an efficient combination of numerical methods, which has the following three very important issues: it is efficient, robust and gets very close to the minimum, if certain assumptions are fulfilled. However, these assumptions cannot be proven *a priori*, because the shape of the loss function and the amount of statistical noise is unknown in most cases. Therefore, a detailed practical evaluation is indispensable.

The methodology of SpaGrOW is presented in detail in Section 2, including complexity and convergency considerations. Section 4 shows the results of a detailed practical evaluation of the methodology, and finally, Section 5 concludes the paper.

### *1.2. Drawbacks of Gradient-Based Methods*

Although gradient-based optimization methods turned out to be suitable and also efficient instruments for the parameterization of force fields in previous work [21–23], they unfortunately have the following disadvantages:

- Due to the statistical noise in the simulation data, the accuracy of a gradient is always limited. It can only predict the decreasing trend of the loss function. As motivated above, this becomes problematic close to the minimum. Theoretically, it is possible to counteract this problem by a suitable discretization for the calculation of the finite differences. However, optimizing the distance of two adjacent points would lead to much more computational effort. Moreover, even if the direction of the gradient can be calculated accurately, the step length control algorithm will only deliver insignificantly small improvements compared to the higher amount of computation time. This is a drawback of the Armijo step length control, but the application of more efficient step length control mechanisms would lead to more computational effort again.
- Whenever a descent direction is incorrect, a point with a smaller loss function value can only be found by chance, because there is no possibility to change the direction. In contrast to descent methods, Trust Region methods decrease the step length in order to find a new, maybe more suitable descent direction of the loss function. This is why these methods are able to get closer to the minimum. However, the Trust Region method applied in previous work requires a Hessian matrix, which leads to significantly more loss function evaluations.
- The amount of simulations is still quite high for gradient- and especially Hessian-based optimization algorithms. At each iteration, a gradient or, additionally, a Hessian have to be computed, which leads to $N$ or, even, $N + N(N + 1)/2$ loss function evaluations. Furthermore, the Armijo step length control algorithm requires additional functional evaluations.

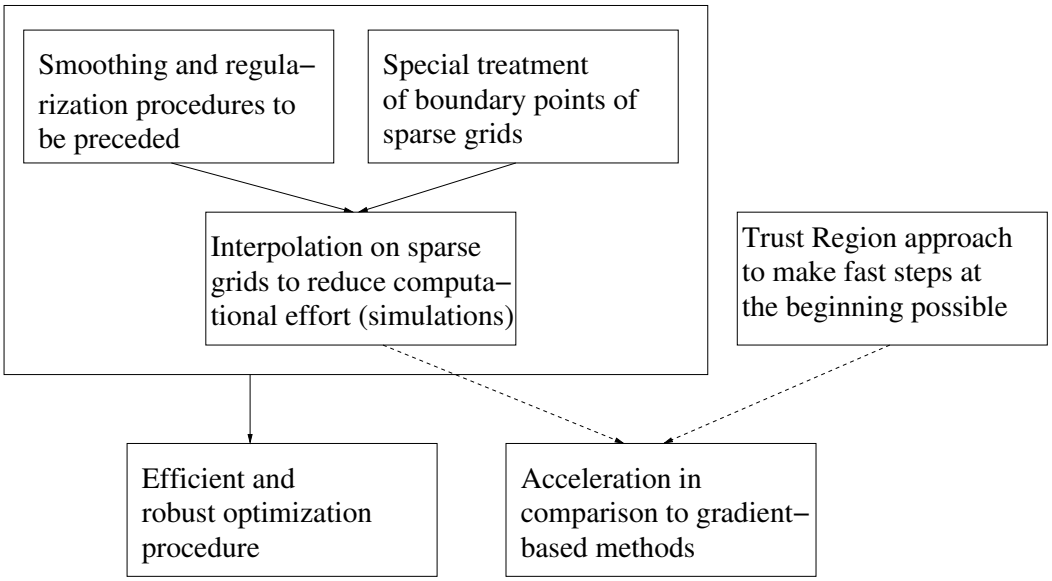
These drawbacks of gradient-based methods motivate the development of an algorithm that is capable of both getting robustly closer to the minimum and solving the optimization problem with significantly less simulations. It should not assume the loss function to be smooth. Instead, it should use smoothing and regularization procedures in order to handle the statistical noise and jagged functions.

## 2. SpaGrOW Methodology: The Main Elements

In this work, a new derivative-free algorithm based on sparse grids and smoothing methods is presented. Figure 3 visualizes the gain in both efficiency and robustness by the combination of the Trust Region approach with the interpolation on sparse grids and smoothing procedures: the interpolation from a sparse grid to a full grid realizes a significant reduction of function evaluations, *i.e.*, simulations, without increasing the interpolation error significantly, on the other hand. At each iteration, the loss function is smoothed before the interpolation is carried out. The quality of the interpolation model is

measured following the Trust Region idea. The actual minimum, which is determined on the full grid, is either accepted as a new iteration or the trust region is decreased. It is desisted from a continuous minimization within the trust region in order to avoid additional internal optimization iterations. There also exist approaches to determine the global minimum of a piecewise-linearly interpolated function via so-called subgradients [27]. However, only for the interpolation itself, at least $3^N$ function evaluations are required. In the present case, there are constraints, due to the minimization on a compact trust region. Hence, the sparse grid approach seems to be more reasonable. Another advantage of the Trust Region approach is the fact that it is able to be fast at the beginning of the optimization procedure and to jump over undesired intermediate local minima. This is due to the size of the step length, which is selected before a descent direction is calculated.

**Figure 3.** Overview of the Sparse Grid-based Optimization Workflow (SpaGrOW): the combination of the Trust Region approach with the interpolation on sparse grids requiring a smoothing procedure to be preceded leads to both increasing efficiency and robustness.




First, the two main elements of SpaGrOW are presented: In Section 2.1, the idea of sparse grids is introduced and the advantages with respect to reduction of computation time are presented. Suitable smoothing and regularization procedures are described in Section 2.2. The algorithm of SpaGrOW is introduced afterwards in Section 3.

### *2.1. Interpolation on Sparse Grids*

The interpolation on sparse grids is a highly efficient discretization method in the field of Finite Element Methodology (FEM). In contrast to full grids, sparse grids possess significantly less grid points, especially in high dimensional spaces. The computational effort decreases from $\mathcal{O}(h^{-N})$ to $\mathcal{O}(h^{-1} \times (\log h^{-1})^{N-1})$, where $h$ is the mesh size of the full grid. In the meantime, the interpolation error increases from $\mathcal{O}(h^2)$ to $\mathcal{O}(h^2 \times (\log h^{-1})^{N-1})$ only, with respect to the $L_\infty$-norm [28]. The interpolation is founded on a tensor product approach for high dimensions and a linear combination of basis functions, e.g., of hierarchical hat functions [29]. Thereby, the basis functions with small

coefficients within the linear combination are left out, and the remaining basis functions correspond to points of a sparse grid. Here, the combination technique by [24] is used, an efficient methodology interpolating a function on regular subgrids. The combination of these subgrids results in a sparse grid. The most important application of interpolation on sparse grids is the solution of partial differential equations, *cf.* [30].

#### 2.1.1. Idea of Sparse Grids

The main idea of sparse grids is to reduce the computational effort without obtaining an intolerable increase of the interpolation error. Especially in high dimensions, the reduction of computational effort becomes notable; Table 1 shows the number of grid points of full and sparse grids of different resolutions and dimensions. In the following, full and sparse grids are introduced with the aid of basis functions for piecewise-bilinear functions, *i.e.*, for the two-dimensional case:

**Table 1.** Comparison of the number of grid points on full and sparse grids of different levels and dimensions. Especially for high levels and dimensions, the computational effort decreases significantly using sparse grids.

| **Level** | $N = 1$ | $N = 2$ | $N = 3$ | $N = 4$ |
|---|---|---|---|---|
| 1 | $3 \rightarrow 3$ | $9 \rightarrow 9$ | $27 \rightarrow 27$ | $81 \rightarrow 81$ |
| 2 | $5 \rightarrow 5$ | $25 \rightarrow 21$ | $125 \rightarrow 81$ | $625 \rightarrow 393$ |
| 3 | $9 \rightarrow 9$ | $81 \rightarrow 49$ | $729 \rightarrow 225$ | $6561 \rightarrow 1329$ |
| 4 | $17 \rightarrow 17$ | $289 \rightarrow 113$ | $4913 \rightarrow 593$ | $83521 \rightarrow 3921$ |
| 10 | $1025 \rightarrow 1025$ | $1.05 \times 10^6 \rightarrow 9217$ | $1.07 \times 10^9 \rightarrow 47103$ | $1.1 \times 10^{12} \rightarrow 1.78 \times 10^5$ |

Let $\Omega_{i,j}$ be the equidistant rectangular grid on the unit square, $\Omega := [0,1]^2$, with mesh sizes $h_i := 2^{-i}$ in $x$- and $h_j := 2^{-j}$ in the $y$-direction. The vector, $\ell := (i,j) \in \mathbb{N}^2$, is called *level* of the grid, $\Omega_{i,j}$, with one-norm $|\ell|_1 := i + j$. Moreover, let $S_{i,j}$ be the space of piecewise-bilinear functions on the grid, $\Omega_{i,j}$. For reasons of simplicity, only those piecewise-bilinear functions are considered here, which fulfill homogeneous Dirichlet boundary conditions in $\Omega_{i,j}$. The respective space is denoted by $S^0_{i,j}$. It can be expressed as a tensor product of subspaces $T_{s,t},\ s = 1, ..., i,\ t = 1, ..., j$, whose functions vanish on all grid points corresponding to $S^0_{s-1,t}$ and $S^0_{s,t-1}$:

$$S^0_{i,j} = \bigotimes_{s=1}^{i} \bigotimes_{t=1}^{j} T_{s,t} \tag{4}$$

Uniquely determined piecewise-bilinear basis functions in $T_{s,t}$ with non-overlapping rectangular supports of size $1/2^{s-1} \times 1/2^{t-1}$ are introduced; the so-called *hierarchical basis* [29]. Thereby, each grid point corresponds to a specific hierarchical basis function. It is situated in the center of the support. Moreover, each grid point belongs to a grid of a certain level, $\ell$. The full grid results from the combination of these grids, and the direct sums of the hierarchical basis functions are the standard basis functions of the full grid.

Each function, $u \in S^0_{i,j}$, can be expressed by a linear combination of hierarchical basis functions:

$$u = \sum_{s=1}^{i} \sum_{t=1}^{j} u_{s,t},\ u_{s,t} \in T_{s,t},\ s = 1, ..., i,\ t = 1, ..., j. \tag{5}$$

In [29], the inequality:

$$\|u_{s,t}\|_\infty \leq 4^{-s-t-1}|u| \tag{6}$$

was proven, where the seminorm, $|u|$, is given by:

$$|u| := \left\| \frac{\partial^4 u}{\partial x^2 \partial y^2} \right\|_\infty \tag{7}$$

In order to obtain a sparse grid, only the functions whose coefficients are greater than or equal to a certain tolerance value are chosen. In [29], this tolerance value is set to $4^{-\hat{\ell}-1}|u|$, where $\hat{\ell}$ is the level of the sparse grid, which is defined in the following. All remaining basis functions are neglected, and a sparse grid space can be defined as follows:

**Definition 1** (Sparse grid space)**.** *The space, $\hat{S}^0_{\hat{\ell}}$, spanned by the subspaces, $T_{i,j}$, with $i + j = \ell \leq \hat{\ell} + 1$, i.e.,*

$$\hat{S}^0_{\hat{\ell}} := \bigotimes_{s=1}^{\hat{\ell}} \bigotimes_{t=1}^{\hat{\ell}-s+1} T_{s,t} = \bigotimes_{s+t \leq \hat{\ell}+1} T_{s,t} \tag{8}$$

*is called the* sparse grid space*. The grid resulting from the combination of the subgrids corresponding to the subspaces, $T_{s,t},\ s + t \leq \hat{\ell} + 1$, is called the* sparse grid *and has the level, $\hat{\ell}$.*

The condition, $|\ell|_1 \leq \hat{\ell} + 1$, leads to a triangular scheme of subspaces, $T_{i,j}$, which is depicted in Figure 4. Please note that $\ell_k = 0,\ k \in \{1, 2\}$, is feasible, but it must hold $\forall_{k\in\{1,2\}}\ \ell_k < \hat{\ell}$. The transition to $N$-dimensional sparse grids of the level, $\hat{\ell}$, is trivial: all subgrids of the level, $\ell$, with $|\ell|_1 \leq \hat{\ell} + N - 1\ \wedge\ \forall_{k=1,...,N}\ \ell_k < \hat{\ell}$ have to be combined, where $|\ell|_1 = \sum_{i=1}^{N} \ell_i$.

Figure 5 shows some examples of sparse grids of the levels 3, 4 and 5 in 2D and 3D, which were produced with an algorithm combining sparse grids of a given level from the respective subgrids.

The coefficients corresponding to grid points of a grid with a high level do not contribute considerably to the interpolant. Hence, they can be neglected. However, a drawback of the sparse grid interpolation is the fact that this is only possible for functions that are sufficiently smooth. Because of inequality in Equation (6), the function that is interpolated must be at least four times continuously differentiable. Otherwise, the estimate cannot be made. As differentiability cannot be assumed in the present application, the combination technique developed by [24] is used here. A second problem concerning the computational effort is that sparse grids are still fully occupied at the boundary, *cf.* Figure 5. This problem can be handled using a transformation that sets the loss function to zero at the boundary. The methodology applied for SpaGrOW is described in the following.

**Figure 4.** Triangular scheme for the combination of a sparse grid of level 3 from two-dimensional subgrids meeting the condition, $|\ell|_1 \leq 3+2-1 = 4 \wedge \forall_{k\in\{1,2\}}\, \ell_k < 3$. If all eight subgrids are combined, a full grid of level 3 is obtained. If only the subgrids of levels (0,0), (1,0), (2,0), (3,0), (0,1), (0,2), (0,3), (1,1), (2,1), (1,2), (2,2), (3,1) and (1,3) are taken, *i.e.*, if the small triangle consisting of three grids at the bottom right is left out, the corresponding sparse grid is obtained, *cf.* Figure 5 top left.

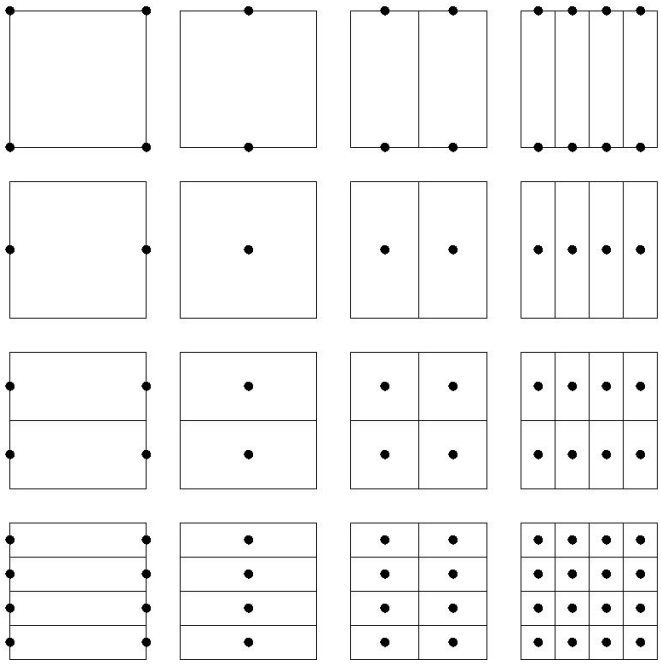
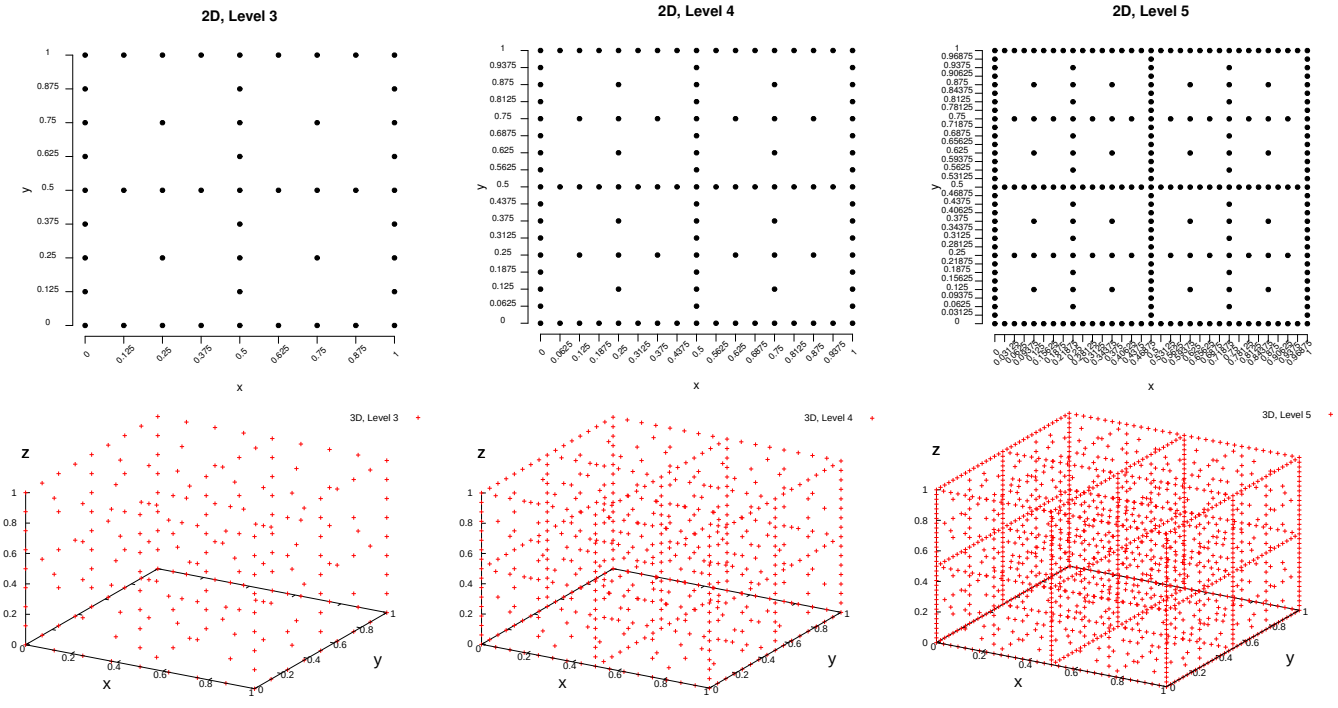


**Figure 5.** Sparse grids of the levels 3, 4, and 5 in 2D and 3D.



2.1.2. Combination Technique

The combination technique developed by [24] combines problems on regular grids of the level, $\ell$, with different mesh sizes, $\ell_1, ..., \ell_N$, to a sparse grid problem in a very efficient way. As this method has proven of value in practical applications and as it is also applicable to functions that are not necessarily smooth, it is taken as the sparse grid interpolation method for SpaGrOW.

As before, the two-dimensional case is considered first: Let $u : \mathbb{R}^2 \to \mathbb{R}$ be an arbitrary function. Every function defined on a sparse grid can be linearly combined from their interpolants on the regular subgrids, $\Omega_{i,j},\ i+j \in \{\hat{\ell}, \hat{\ell}+1\}$. If $u_{i,j} \in T_{i,j}$ is considered, a combination function, $u^c_{\hat{\ell}}$, can be defined as follows:

**Definition 2** (Combination function (2D))**.** *The* combination function *of solutions $u_{i,j} \in T_{i,j}$ of FEM problems on regular two-dimensional grids of the level, $(i,j)$, $i+j \in \{\hat{\ell}, \hat{\ell}+1\}$, is given by:*

$$u^c_{\hat{\ell}} := \sum_{i+j=\hat{\ell}+1} u_{i,j} - \sum_{i+j=\hat{\ell}} u_{i,j} \tag{9}$$

The error, $u - u^c_{\hat{\ell}}$, is in $\mathcal{O}\left(h^2_{\hat{\ell}} \log(h_{\hat{\ell}})^{-1}\right)$, *cf.* Section 3.3.1. Analogous to Definition 2, the following definition is given for the three-dimensional case:

**Definition 3** (Combination function (3D))**.** *The* combination function *of solutions $u_{i,j,k} \in T_{i,j,k}$ of FEM problems on regular three-dimensional grids of the level, $(i,j,k)$, $i+j+k \in \{\hat{\ell}, \hat{\ell}+1, \hat{\ell}+2\}$, is given by:*

$$u^c_{\hat{\ell}} := \sum_{i+j+k=\hat{\ell}+2} u_{i,j,k} - 2 \sum_{i+j+k=\hat{\ell}+1} u_{i,j,k} + \sum_{i+j+k=\hat{\ell}} u_{i,j,k} \tag{10}$$

Via complete induction, this can directly be transferred to the $N$-dimensional case:

**Definition 4** (Combination function (in general))**.** *The* combination function *of solutions $u_\ell \in T_\ell$ of FEM problems on regular $N$-dimensional grids of the level, $\ell$, with $|\ell|_1 = \hat{\ell} + N - 1 - i,\ i = 0, ..., N-1$, is given by:*

$$u^c_{\hat{\ell}} := \sum_{i=0}^{N-1} (-1)^i \binom{N-1}{i} \sum_{|\ell|_1 = \hat{\ell}+N-1-i} u_\ell \tag{11}$$

**Remark 5.** *Because of the condition, $|\ell|_1 = \hat{\ell} + N - 1 - i,\ i = 0, ..., N-1$, a multilinear interpolation on a full regular grid with mesh size $1/2^{\hat{\ell}}$ and level $\bar{\ell} = (\hat{\ell})$ can be executed as follows: All function values are computed on a sparse grid of the level, $(\hat{\ell}, ..., \hat{\ell}) \in \mathbb{N}^N$. Because of the hierarchical structure of the sparse grid, all function values of the regular subgrids of the level, $\ell$, with $|\ell|_1 \leq \hat{\ell} + N - 1$ are known, i.e., all function values required for the interpolation, cf. Equation (11).*

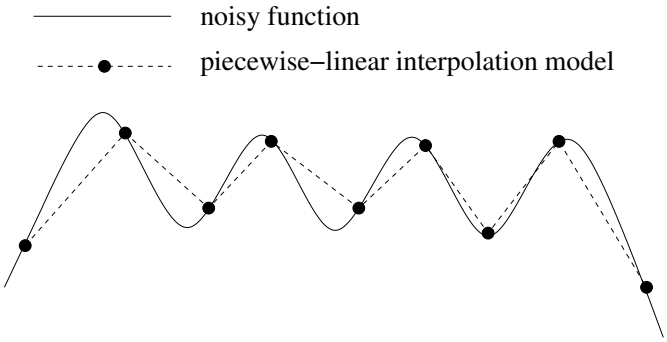
### *2.2. Smoothing Procedures*

As already mentioned, the combination technique is applicable to functions that are not necessarily differentiable. However, an assumption of this method is the existence of an asymptotic error expansion for the discrete solution, *cf.* Section 3.3.1. Hence, the function to be interpolated should possess certain smoothness properties. At least, the statistical noise has to be filtered out as effectively as possible, and the quality of the continuity of the loss function should be high enough. As noise can be unfavorable for the combination technique, smoothing procedures have to be applied before the piecewise-linear interpolation is performed. In SpaGrOW, radial basis functions (RBFs) have turned out to be very suitable in order to approximate the loss function. In some cases, especially close to the minimum, a simple quadratic model is sufficient, as well. Additionally, the noise is filtered out via regularization procedures. In the approximation process, the respective nonlinear regression problem is not solved via least squares, but estimators with a lower variance. The regularization methods used in SpaGrOW are elastic nets and multi-adaptive regression splines. The applied smoothing and regularization algorithms are described in brief, and a theoretical selection is presented and discussed in the following. The final decision can only be made after a detailed practical evaluation, which is performed in Section 4.1.

#### 2.2.1. Effects of Statistical Noise on Piecewise-linear Interpolation

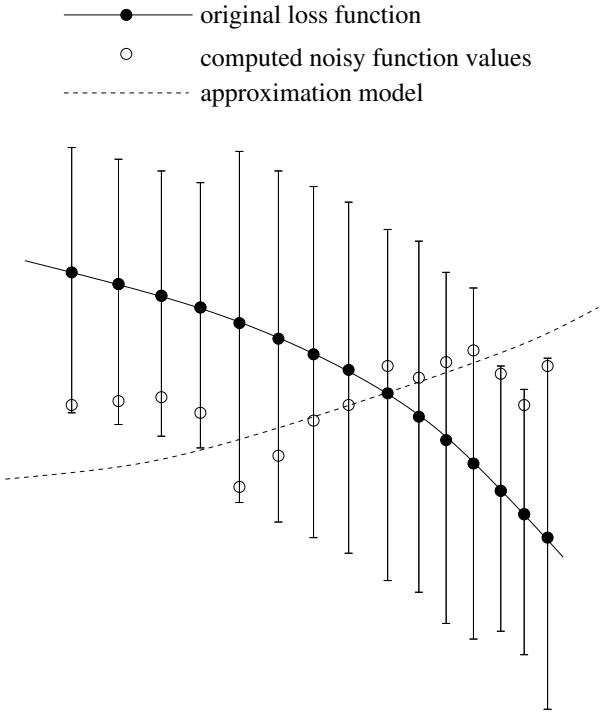
Suppose $u$ is linear and noisy. Then, the piecewise-linear interpolation should reproduce the function exactly. However, statistical noise can lead to a staggered function, which is shown in Figure 6. As this is not desired, a preprocessing, which approximates the function to be interpolated and eliminates the noise, is indispensable.

**Figure 6.** Problematic in the case of piecewise-linear interpolation of a noisy function, the interpolation leads to a staggered function reproducing the noise.




However, another problematic may appear whenever the sampled points used for the approximation are situated too close to each other: Figure 7 indicates that in this case, the trend of the function can be reproduced completely incorrectly. Therefore, SpaGrOW has to deal with this phenomenon, and the size of the trust region becomes too small has to be avoided.

**Figure 7.** Problematic in the case of the approximation of a noisy function, when the points are situated too close to each other, the function values cannot be differentiated anymore, and the trend of the function can be reproduced completely incorrectly by the smoothing procedure.




2.2.2. Smoothing Functions

In the following, the approximation of high-dimensional functions with radial basis functions (RBFs) is presented shortly.

RBFs are an efficient and common method for interpolation and approximation tasks [31]. The word *radial* indicates that they are functions of the distance of two adjacent points. Hence, points which are situated on a hyperball with radius $r$ around a reference point have the same function value. Here are some examples:

$$\phi(r) = r \text{ (linear)} \tag{12}$$

$$\phi(r) = \exp\left(-cr^2\right),\ c \in \mathbb{R}^+ \text{ (Gaussian)} \tag{13}$$

$$\phi(r) = \sqrt{r^2 + c^2},\ c \in \mathbb{R}^{\neq 0} \text{ (multiquadric)} \tag{14}$$

$$\phi(r) = \frac{1}{\sqrt{r^2 + c^2}},\ c \in \mathbb{R}^{\neq 0} \text{ (inverse multiquadric)} \tag{15}$$

$$\phi(r) = r^3 \text{ (cubic)} \tag{16}$$

$$\phi(r) = r^2 \log r \text{ (thin plate splines)} \tag{17}$$

A function, $f : \mathbb{R}^N \to \mathbb{R}$, is approximated by:

$$f(X) \approx \sum_{i=1}^{K} \beta_i h_i \left(\|X - X_i\|\right) \tag{18}$$

where $X \in \mathbb{R}^N$ is an arbitrary test point and $K \leq m$, where $m \in \mathbb{N}$ is the total number of points. The points, $X_i \in \mathbb{R}^N,\ i = 1, ..., K$, are the $K$ representative centers of the training set, on the basis of which the coefficients, $\beta_i \in \mathbb{R},\ i = 1, ..., K$, are determined. The functions, $h_i \in \mathcal{H},\ i = 1, ..., K$, are from the set of RBFs, $\mathcal{H}$. The computation of the coefficients is realized by so-called *RBF networks*, *cf.* Figure 8, which are directed neuronal networks [32] and proceed from the input to the output neurons. They possess one layer of hidden neurons only, which corresponds to the determination of the coefficients. In the case of an approximation, it holds $m \leq K$, and the following overdetermined linear equation system (LES) has to be solved:

$$H\beta = Y \tag{19}$$

where:

$$H = \begin{pmatrix} h_1(\|X_1 - X_1\|) & h_2(\|X_1 - X_2\|) & \cdots & h_K(\|X_1 - X_K\|) \\ h_1(\|X_2 - X_1\|) & h_2(\|X_2 - X_2\|) & \cdots & h_K(\|X_2 - X_K\|) \\ \vdots & \vdots & \vdots & \vdots \\ h_1(\|X_m - X_1\|) & h_2(\|X_m - X_2\|) & \cdots & h_K(\|X_m - X_K\|) \end{pmatrix} \in \mathbb{R}^{m \times K}$$

and:

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} f(X_1) \\ f(X_2) \\ \vdots \\ f(X_m) \end{pmatrix} \in \mathbb{R}^m$$

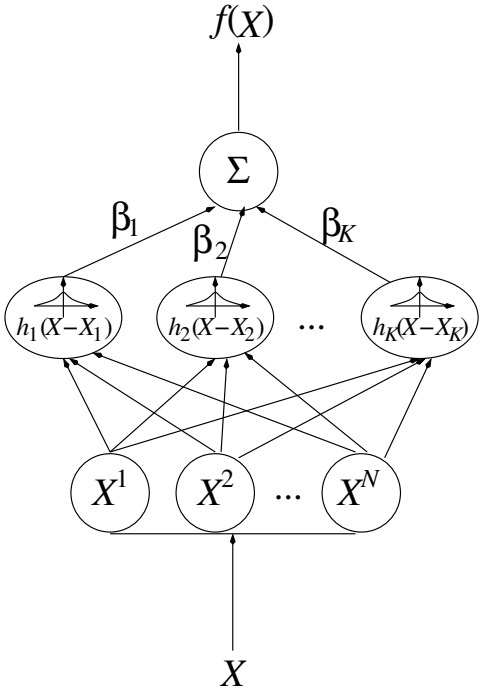
The solution vector, $\beta := (\beta_1, ..., \beta_K)^T \in \mathbb{R}^K$, can be computed e.g., by using least squares.

In order to select $K$ representative centers out of the $m$ training data in an efficient way, an unsupervised learning method is used, namely the automated classification (*clustering*) procedure, *k-means* [33]. The $K$ centers are also the centroids of the data classes obtained by the clustering algorithm.

A drawback of RBFs is the high computational effort required for the collocation of the matrix, $H$, in Equation (19). For $N = 5$, more than one thousand and for $N = 6$, over 72 million RBF evaluations are necessary. Hence, RBFs should not be used for $N > 4$.

For dimensions optimization problems, a simpler smoothing procedure is applied, whose complexity is only quadratic in the dimension. Instead of approximating the loss function itself, each of the physical target properties is approximated linearly, which results in a quadratic model of the loss function. This method is called *linear property approximation (Lipra)* and only requires the solution of an LES of the size $N$ on a sparse grid. The loss function values can easily be obtained by inserting the approximated physical properties into the loss function. The complexity of the Lipra algorithm depends on the number of properties $n$ and the dimension, $N$. It is $\mathcal{O}(nN^2)$.

**Figure 8.** Radial basis function (RBF) network: The network proceeds for a test point, $X$, from the input neurons, $X^j,\ j = 1, ..., N$, (components of $X$), over a layer of hidden neurons containing the RBF evaluations, $h_i(X - X_i),\ i = 1, ..., K$, with the coefficients, $\beta_i,\ i = 1, ..., K$, to the output neuron, where the summation takes place. The result is the function value, $f(X)$.




2.2.3. Regularization Algorithms

In order to avoid overfitting, the statistical noise the simulation data is affected with has to be eliminated. Therefore, the regression coefficients computed within the smoothing procedure may not be overestimated, so that not too much stress is laid on noisy data points. This can be achieved by a weighted regression, where a weighting factor is assigned to each data point that is small, whenever the noise on its function value is high, and *vice versa*. Another way is to constrain the coefficients in the optimization problem contained in the regression procedure. These so-called *regularization algorithms* do not solve the regression problem with the least square estimator, $\beta_{\mathrm{LS}}$, but with low-variance estimators. The former is given by:

$$\beta_{\mathrm{LS}} = \arg\min_{\beta} ||Y - H\beta||_2^2 \tag{20}$$

and has the following drawbacks:

(1) Due to its high variance, the probability of overfitting is always high.
(2) The euclidean norm, $||\beta_{\mathrm{LS}}||_2$, is, in general, high, as well.
(3) There is no variable selection, *i.e.*, none of the coefficients of $\beta_{\mathrm{LS}}$ can be zero. Hence, correlated variables have always an influence within the model, even if they do not have any impact on the loss function.
(4) If the dimension is greater than the number of data points, the least square method does not have a solution. However, due to $K << m$, this is not important in the present case.

In order to counteract these drawbacks, the regression coefficients have to be *shrinked* or set to zero in some cases (variable selection). This has the effect that especially correlated variables and outliers only have low influence within the model. Moreover, the variance of the estimator is reduced, so that the probability of overfitting decreases. In SpaGrOW, this is realized by a *Naive Elastic Net* [34]:

$$\beta_{\mathrm{NEN}} = \arg\min_{\beta} ||Y - H\beta||_2^2, \text{ where } \alpha||\beta||_2^2 + (1-\alpha)|\beta|_1 \leq t, \ \alpha \in [0,1], \ t > 0 \tag{21}$$

It contains two additional model parameters, $\alpha \in [0,1]$ and $t \in \mathbb{R}^{>0}$, which can be determined, e.g., by a ten-fold cross-validation. A naive elastic net possesses a so-called *grouping effect*, *i.e.*, correlated model variables have similar regression coefficients. A penalized formulation of Equation (21) is achieved by introducing Lagrangian multiplicators, $\lambda_1, \lambda_2 \geq 0$:

$$\beta_{\mathrm{NEN}} = \arg\min_{\beta} \mathcal{L}(\beta, \lambda_1, \lambda_2), \ \mathcal{L}(\beta, \lambda_1, \lambda_2) := ||Y - H\beta||_2^2 + \lambda_2||\beta||_2^2 + \lambda_1|\beta|_1 \tag{22}$$

It holds $\alpha = \frac{\lambda_2}{\lambda_1+\lambda_2}$. For $\alpha \in \{0,1\}$, the two most famous methods in the field of biased regression are obtained: for $\alpha = 1$, it is the *Ridge Regression* [35], and for $\alpha = 0$, it is the *Least Absolute Shrinkage and Selection Operator (LASSO)* [36]. Both methods contain shrinkage, but only in the case of the LASSO, a variable selection is possible. For more details about Elastic Nets, Ridge Regression and the LASSO, *cf.* [34].

Another regularization method is based on the application of *Multivariate Adaptive Regression Splines (MARS)* [37]. The basis functions within this algorithm are products of so-called *hinge functions* of the order $q \in \mathbb{N}$:

$$S_q(x) = S_q(x - \nu) := [\pm(x - \nu)]_+^q, \ q \in \mathbb{N} \tag{23}$$

where $\nu$ is a *node*. "+" means that for negative arguments, the functions becomes zero. Two hinge functions with opposite sign belong together and are mirrored at their common node. Hence, they divide the input space into two disjoint subsets. By the recursive selection of $p$ nodes, the input space is divided into $p+1$ disjoint subsets. A single hinge function or a product of hinge functions, which lead to even more divisions, is assigned to each subspace. This models the interaction of the input variables. The approximation of a function, $f : \mathbb{R}^N \to \mathbb{R}$, within the MARS algorithm is realized as follows:

$$f(x) \approx \sum_{i=1}^{p} (\beta_M)_i \prod_{j=1}^{s_i} S_q(x_{N(i,j)} - \nu_{i,j}) \tag{24}$$

Thereby, the $s_i$ are the numbers of hinge functions considered in the partitioning, $i \in \{1, ..., p\}$, $\nu_{i,j}$, $i = 1, ..., p$, $j = 1, ..., s_i$ are the nodes of the recursive division and $N(i,j)$ is the dimension of the input vector, $x$, to which the division defined by the node $\nu_{i,j}$ refers. In total, $f$ is approximated by a regression spline and, in the case of $q = 1$, by a piecewise-linear function. The lower probability of overfitting is reached by pruning the basis functions involved in the model. For more details about MARS, *cf.* [37]. A disadvantage of this regularization algorithm is the fact that it requires high computational effort for higher dimensions, due to the pruning procedure.

Please note that in the case of the Lipra algorithm, which is used for high-dimensional problems, no regularization procedure is applied, due to the high amount of computational effort.

2.2.4. Selection of Smoothing Procedures: Theoretical Considerations

The selection of the best smoothing procedure is made first from a theoretical perspective. A detailed practical evaluation is pointed out in Section 4.1. Theoretically, an approximation method can be selected, due to a reliable estimate of the approximation error within a domain, $\Omega \subset \mathbb{R}^N$. Such estimates exist for positive definite RBFs, *cf.* [38]. The Gaussian, the inverse multiquadric and so-called *Wendland functions* [39] are positive definite. The latter are RBFs with compact supports and are piecewise polynomial with a minimal degree dependent on differentiability and dimension.

An estimate of the approximation error can be realized via the introduction of *Native Spaces* [38]:

**Definition 6** (Native Spaces)**.** *The* Native Space, $\mathcal{N}_{\phi(\Omega)}$, *for a given positive definite RBF,* $\phi$, *within the domain,* $\Omega$, *is given by:*

$$\mathcal{N}_{\phi(\Omega)} := \overline{\{\phi(||\cdot - x||), x \in \Omega\}} \tag{25}$$

*where for* $f_1 = \sum_{k=1}^{n_1} \alpha_k \phi(||\cdot - x_k||) \in \mathcal{N}_{\phi(\Omega)}$ *and* $f_2 = \sum_{j=1}^{n_2} \beta_j \phi(||\cdot - x_j||) \in \mathcal{N}_{\phi(\Omega)}$*:*

$$\langle f_1, f_2 \rangle_{\phi(\Omega)} := \sum_{k=1}^{n_1} \sum_{j=1}^{n_2} \alpha_k \beta_j \phi(||x_k - x_j||) \tag{26}$$

*Thereby,* $n_1, n_2 \in \mathbb{N}$, $\alpha_k, \beta_j \in \mathbb{R}$, $k = 1, ..., n_1$, $j = 1, ..., n_2$, *and* $x_k, x_j \in \Omega$, $k = 1, ..., n_1$, $j = 1, ..., n_2$.

The Hilbert space $\mathcal{N}_{\phi(\Omega)}$ is the completing of the pre-Hilbert space, $\{\phi(||\cdot - x||), x \in \Omega\}$. As the approximation error depends on the size of the domain, $\Omega$, the so-called *fill distance* is defined next:

**Definition 7** (Fill distance)**.** *For a given discrete training set,* $\mathcal{X} \subset \Omega$, *with training data,* $x_j$, $j = 1, ..., m$, $m \in \mathbb{N}$, *the* fill distance, $\Delta_{\Omega,\mathcal{X}}$, *is defined by:*

$$\Delta_{\Omega,\mathcal{X}} := \sup_{x \in \Omega} \min_{j=1,...,m} ||x - x_j|| \tag{27}$$

From Definitions 6 and 7, the following theorem for the estimation of the approximation error can be formulated:

**Theorem 8** (Approximation error for positive definite RBFs)**.** *Let* $\Omega$, $\mathcal{X}$, $\phi$ *and* $f$ *be defined as above. Let, furthermore,* $g$ *be the approximating function for* $f \in \mathcal{N}_{\phi(\Omega)}$, *obtained by the positive definite RBF,* $\phi$*. Then, the following estimate for the approximation error holds:*

$$||f - g||_{L_\infty(\Omega)} \leq h(\Delta_{\Omega,\mathcal{X}}) ||f||_{\mathcal{N}_{\phi(\Omega)}} \tag{28}$$

*where* $\lim_{\Delta_{\Omega,\mathcal{X}} \to 0} h(\Delta_{\Omega,\mathcal{X}}) = 0$.

*Proof.* [38] □

This theorem is very important for the convergency of SpaGrOW, *cf.* Section 3.3.2. However, for SpaGrOW, another condition, $\lim_{\Delta_{\Omega,\mathcal{X}} \to 0} h(\Delta_{\Omega,\mathcal{X}}) = 0$, has to be fulfilled. The following definition introduces the term *stability* of an approximation in the sense of SpaGrOW:

**Definition 9** (Stable approximation)**.** *Let $\Omega$ and $\mathcal{X}$ be defined as above. Let be $f \in \mathcal{H}$, where $\mathcal{H}$ is a Hilbert space of functions on $\Omega$ with the scalar product, $\langle \cdot, \cdot \rangle_{\mathcal{H}}$, and norm $||f||_{\mathcal{H}} = \sqrt{\langle f, f \rangle_{\mathcal{H}}}$ for $f \in \mathcal{H}$. An approximation within the domain, $\Omega$, by a function, $g \in \mathcal{P}$, where $\mathcal{P}$ is a pre-Hilbert space with the same scalar product and $\bar{\mathcal{P}} = \mathcal{H}$, is called stable, if:*

$$||f - g||_{\mathcal{H}} \leq \kappa h(\Delta_{\Omega,\mathcal{X}}) \tag{29}$$

*where $\kappa > 0$, $\lim_{\Delta_{\Omega,\mathcal{X}} \to 0} h(\Delta_{\Omega,\mathcal{X}}) = 0$ and $\lim_{\Delta_{\Omega,\mathcal{X}} \to 0} \frac{h(\Delta_{\Omega,\mathcal{X}})}{\Delta_{\Omega,\mathcal{X}}} = 0$.*

The second important condition for the convergency of SpaGrOW is $\lim_{\Delta_{\Omega,\mathcal{X}} \to 0} \frac{h(\Delta_{\Omega,\mathcal{X}})}{\Delta_{\Omega,\mathcal{X}}} = 0$. The following corollary ascertains the stability of an approximation based on a Gaussian RBF:

**Corollary 10** (Stability of an approximation based on a Gaussian RBF)**.** *Let $\Omega, \mathcal{X}$ and $f$ be defined as in Theorem 8. Let, furthermore, $\phi(|| \cdot - x||) = \exp(-c|| \cdot - x||^2)$ for $x \in \Omega$ and $c \in \mathbb{R}^+$ a Gaussian RBF. Then, the approximation of $f$ by $g(x) := \sum_{k=1}^{\nu} \alpha_k \phi(||x_k - x||),\ \nu \in \mathbb{N},\ \alpha_k \in \mathbb{R},\ x_k \in \Omega,\ k = 1, ..., \nu$, is stable.*

*Proof.* For a Gaussian RBF, the following estimation holds (*cf.* [38]):

$$||f - g||_{L_\infty(\Omega)} \leq \exp\left(-d\left(\frac{\log(\Delta_{\Omega,\mathcal{X}})}{\Delta_{\Omega,\mathcal{X}}}\right)\right) ||f||_{\mathcal{N}_{\phi(\Omega)}} \tag{30}$$

where $d > 0$ is a constant. With $h(\Delta_{\Omega,\mathcal{X}}) := \exp\left(-d\left(\frac{\log(\Delta_{\Omega,\mathcal{X}})}{\Delta_{\Omega,\mathcal{X}}}\right)\right)$, it holds $\lim_{\Delta_{\Omega,\mathcal{X}} \to 0} h(\Delta_{\Omega,\mathcal{X}}) = 0$ and $\lim_{\Delta_{\Omega,\mathcal{X}} \to 0} \frac{h(\Delta_{\Omega,\mathcal{X}})}{\Delta_{\Omega,\mathcal{X}}} = 0$. With $\kappa := ||f||_{\mathcal{N}_{\phi(\Omega)}}$ and the fact that $\mathcal{N}_{\phi(\Omega)}$ is a Hilbert space and $g$ an element of the pre-Hilbert space, $\{\phi(|| \cdot - x||), x \in \Omega\}$, with $\overline{\{\phi(|| \cdot - x||), x \in \Omega\}} = \mathcal{N}_{\phi(\Omega)}$, it follows that the approximation based on a Gaussian RBF is stable with respect to Definition 9. □

Because of Corollary 10, the Gaussian RBF is selected for SpaGrOW for theoretical reasons. However, such estimates exist for other positive RBFs, as well. The final selection of the Gaussian RBF is made after a detailed practical evaluation, as mentioned above.

## 3. The SpaGrOW Algorithm

### *3.1. Ingredients of the Algorithm*

Within SpaGrOW, the minimization of the interpolation model on a full grid is performed discretely. The corresponding complexity is $\mathcal{O}\left((n+1)^{2N}\right)$, where $n+1$ $(n \in \mathbb{N})$ is the number of points of the full grid in one dimension. The first question arising is whether the algorithm should be applied locally or globally. Globally, this means that the smoothing procedure and the sparse grid interpolation are performed on the complete feasible domain for the force field parameters. However, it results that a local consideration is the better way, *i.e.*, the combination with the Trust Region approach is highly important. This makes SpaGrOW an iterative local optimization method. The algorithm is described in detail in the following.

3.1.1. Local and Global Consideration

The combination technique delivers a piecewise-multilinear function, $q : \mathbb{R}^N \rightarrow \mathbb{R}$, which either interpolates the loss function, $F$, itself or an approximating function, $G$, from a sparse grid of the level, $\hat{\ell} \in \mathbb{N}$, on a full grid, $G^N_{\bar{\ell}}$, with the level, $\bar{\ell} = (\hat{\ell}, ..., \hat{\ell}) \in \mathbb{N}^N$. The total error resulting from the approximation and interpolation error can be measured via the $L_2$- or $L_\infty$-distance between $F$ and the interpolation model, $q$, on the unit square, $\Omega := [0,1]^N$:

$$||e||_{L_2,[0,1]^N} := ||F-q||_{L_2,[0,1]^N} = \left( \int_{[0,1]^N} (F(x)-q(x))^2 \, dx \right)^{\frac{1}{2}} \tag{31}$$

$$||e||_{L_\infty,[0,1]^N} := ||F-q||_{L_\infty,[0,1]^N} = \max_{x \in [0,1]^N} |F(x)-q(x)| \tag{32}$$

If the function values, $F(x)$, are known for $x \in G^N_{\bar{\ell}}$, the error terms in Equations (31) and (32) can be approximated numerically.

The total error is important for the assessment of whether SpaGrOW can be applied globally or not. Hence, it has to be determined on a full grid for the local and the global variant. Moreover, the convergency behavior of SpaGrOW has to be analyzed. There are some arguments for a local consideration: Following the heuristics indicated in Section 2.2, it is indispensable to perform a smoothing procedure before interpolating. In the global case, this smoothing procedure would be executed on a sparse grid discretizing the complete feasible domain. As the loss function can be arbitrarily complex and jagged, it should be nearly impossible to reproduce them in an accurate way within a huge domain with only a small number of data points. Furthermore, the discretization error would be too high within a huge domain for the determination of the global minimum. However, it would be possible to apply the algorithm globally first and to make local refinements afterwards, but an inaccurate approximation of the loss function can lead far away from the minimum. Hence, a local consideration should be preferred, and SpaGrOW should be a local iterative procedure, like the gradient-based methods applied for the present task.

In order to verify these heuristic considerations, a practical analysis was performed, where the loss function was replaced by the simple two-dimensional function, $H(x_1, x_2) = -\exp\left(-(x_1-2)^2-(x_2+1)^2\right)$. The global minimum is situated at $(x_1^*, x_2^*) = (2, -1)$, with corresponding function value, $H(2,-1) = -1$. Artificial statistical uncertainties, *i.e.*, uniformly distributed random numbers from the interval, $[-0.03H(x_1,x_2), +0.03H(x_1,x_2)]$, were added on the function values, $H(x_1, x_2)$. For reasons of brevity, the analysis is not reported here. The results were the following:

- **Comparison of the total errors:**
  - In the global consideration, the total errors were significantly higher than in the local consideration with respect to both the $L_2$- and the $L_\infty$-norm.
  - The smoothing error is considerably higher in the global case than in the local one, *i.e.*, the function cannot be reproduced accurately. This was also the case when the function was not affected with statistical noise.

- **Convergency analysis:**
  - Without artificial uncertainties, the global variant of SpaGrOW only led to the minimum when the minimum was a grid point. Otherwise, the minimum could only be determined within the accuracy of the discretization.
  - With artificial uncertainties, both discretization and approximation error had a negative effect on the convergency. The resulting approximated minimum was far away from the real one.

To summarize, a local consideration is preferred to a global one.

#### 3.1.2. Combination with the Trust Region Approach

For each iteration, a compact neighborhood, where the minimization problem is solved discretely, is determined. In most cases, the minimum is situated at the boundary of the compact domain. If certain assumptions are fulfilled, this boundary minimum becomes the new iteration and the center of a new compact neighborhood.

Following the idea of Trust Region methods [25,26], the compact neighborhood is a trust region of the size, $\Delta_k > 0$. Due to the sparse grid interpolation, it is not a ball, $B_{\Delta_k}(x^k)$, but a hyperdice, $\mathcal{W}^k$, of the form:

$$\mathcal{W}^k := \bigotimes_{i=1}^{N} \left[x_i^k - \Delta_k, x_i^k + \Delta_k\right] \tag{33}$$

where $x^k$ is the $k$th iteration of SpaGrOW. On the one hand, the size, $\Delta_k$, has to be small enough, so that $\mathcal{W}^k$ is situated within the feasible domain of the force field parameters and the interpolation model is consistent with the original loss function, $F$, as already discussed in Section 3.1.1. On the other hand, $\Delta_k$ has to be large enough, so that the method converges as fast as possible to the global minimum within the feasible domain. Hence, the division of the domain into non-disjoint hyperdices should not be too fine.

The quality of the model, $q$, is estimated using the following ratio:

$$r^k := \frac{F(x^k) - F(x^*)}{q(x^k) - q(x^*)} = \frac{F(x^k) - F(x^*)}{G(x^k) - q(x^*)} \tag{34}$$

where $x^*$ is the discrete minimum on a full grid within the hyperdice, $\mathcal{W}^k$. As $x^k$ is a point of the sparse grid and the model, $q$, interpolates the approximating function, $G$, from the sparse grid on the full grid, it holds $q(x^k) = G(x^k)$.

In practice, two thresholds, $0 < \eta_1 < \eta_2$, and size parameters, $0 < \gamma_1 < 1 < \gamma_2$, are introduced, and the following three cases are considered:

- $\eta_2 > r^k \geq \eta_1$—in this case, the model is consistent with $F$, and the minimum, $x^*$, is taken as a new iteration: $x^{k+1} := x^*$.
- $r^k \geq \eta_2 > \eta_1$—then, $x^*$ is taken as a new iteration, as well, and at the same time, the trust region is increased in order to accelerate the convergency: $\Delta^{k+1} := \gamma_2 \Delta_k$.
- $r^k < \eta_1$—in this case, the model is not consistent with $F$, and a better model has to be determined by decreasing the trust region: $\Delta^{k+1} := \gamma_1 \Delta_k$.

As for the Trust Region methods, a minimal $\Delta^{\min} > 0$ is defined *a priori*. The algorithm is stopped as soon as $\exists\, k \;:\; \Delta_k < \Delta^{\min}$.

#### 3.1.3. Treatment of Boundary Points

After the application of a transformation, $\xi : \mathcal{W}^k \to [0,1]^N$, into the unit hyperdice, *cf.* Section 3.2.1, the loss function values are set to zero at its boundary by an appropriate modification. The reason of the realization of homogeneous Dirichlet boundary conditions is the fact that also sparse grids are fully occupied at their boundaries. In order to save a magnitude of simulations, the original loss function, $F$, is multiplied with a product of sine functions, *i.e.*, the modified function:

$$\begin{aligned} \bar{F} : [0,1]^N &\to \mathbb{R}_0^+ \\ y &\mapsto \left( \prod_{i=1}^{N} \sin \pi y_i \right) F(y) \end{aligned} \tag{35}$$

where $y = \xi(x)$ with $x \in \mathcal{W}^k$ is considered instead of $F$. Hence, the smoothing and interpolation procedures are applied for $\bar{F} \circ \xi : \mathcal{W}^k \to \mathbb{R}_0^+$. Due to Equation (35), it holds $\bar{F}|_{\partial[0,1]^N} = 0$. However, as $F$ and not $\bar{F}$ has to be minimized and as $F$ and $\bar{F}$ do not have the same minimum, the back-transformation:

$$F(x) = F(\xi^{-1}(y)) = \frac{\bar{F}(y)}{\prod_{i=1}^{N} \sin \pi y_i} \tag{36}$$

has to be applied before the discrete minimization is performed. Hence, the minimum of $F \circ \xi^{-1}$ has to be determined, which is not possible at the boundary of $\mathcal{W}^k$. As the minimum is expected to be situated at the boundary, only the grid:

$$\tilde{G}_{\hat{\ell}}^N := G_{\hat{\ell}}^N \backslash \partial G_{\hat{\ell}}^N \tag{37}$$

is considered for the minimization. This grid does not contain any boundary point of the original grid. However, this reduces the size of the trust region, but the loss in convergency speed is negligible, due to the number of simulations to be saved; for $N = 4$ and $\hat{\ell} = 2$, only nine instead of 393 simulations (*cf.* Table 1) per iteration are required.

### *3.2. The Full Algorithm*

#### 3.2.1. Structure

The algorithm of SpaGrOW, *i.e.*, the inner iteration of the optimization procedure shown in Figure 1, is visualized in Figure 9 and has the following structure:

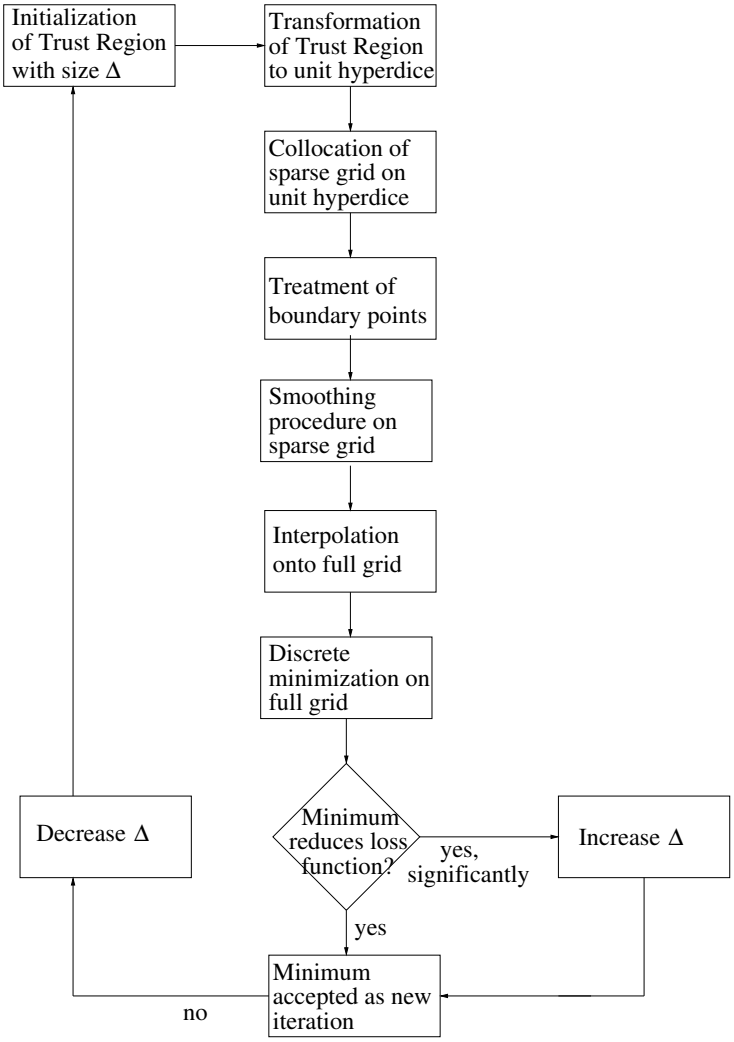
- **Initialization:** Choose an initial vector, $x^0$, and an initial step length, $D^0 > 0$, so that:

$$\forall_{i=1,\ldots,N}\; c_i^0 \leq x_i^0 \leq C_i^0,\; \Delta_0 < C_i^0 - x_i^0,\; \Delta_0 < x_i^0 - c_i^0 \tag{38}$$

  Thereby, $[c_i^0, C_i^0]$ is the feasible interval for the $i$th force field parameter. The maximal step length possible, $\Delta^{\max}$, is computed at the beginning, and $\Delta_0$, as well as a minimal step length, $\Delta^{\min}$, are set in relation to it. Please note that on the one hand, $\Delta_0$ must not be too small, due to the

noise, and that, on the other hand, it must not be too large, so that the problematic described in Section 3.1.1. is not faced.
Let $k := 0$.

**Figure 9.** Overview of the SpaGrOW algorithm, *i.e.*, the inner iteration of the optimization procedure visualized in Figure 1. The Trust Region size, $\Delta$, is increased or decreased, depending on the quality of the approximation model on the sparse grid.




- **Transformation:** Via the transformation:

$$\begin{aligned} \xi : \mathcal{W}^k &\rightarrow [0,1]^N \\ x &\mapsto \frac{x - \left(\min_{i=1,\ldots,N} x_i^k\right) \cdot e}{\left(\max_{i=1,\ldots,N} x_i^k\right) \cdot e - \left(\min_{i=1,\ldots,N} x_i^k\right) \cdot e} \\ &= \frac{1}{2\Delta_k}\left(x - x^k + \Delta_k \cdot e\right) \end{aligned} \tag{39}$$

the initial vector, $x^0$, is mapped from the hyperdice of the size, $\Delta_0$, into the unit hyperdice. Thereby, $e = (1, ..., 1)^T \in \mathbb{R}^N$ and $x_i^k,\ i = 1, ..., N$ are the components of the vector, $x^k$.
Please note that only the back-transformation obtained by the inverse function:

$$\begin{aligned} \xi^{-1} : [0,1]^N &\rightarrow \mathcal{W}^k \\ y &\mapsto 2\Delta_k y + x^k - \Delta_k \cdot e \end{aligned} \tag{40}$$

is important, because, first, a sparse grid is simply collocated in $[0, 1]^N$. Then, the grid points are back-transformed into the force field parameter space via $\xi^{-1}$, so that molecular simulations can be executed.

- **Sparse grid:** A sparse grid, $\hat{G}_{\hat{\ell}}^N$, is determined within $[0, 1]^N = \xi\left(\mathcal{W}^k\right)$. Note that the transformation, $\xi$, has not to be applied explicitly. As the combination technique is used, the grid points of the sparse grid are determined hierarchically from the required subgrids, *cf.* Equation (11). Let $y_{\ell,j}$ be a point of the subgrids of the level, $\ell = (\ell_1, ..., \ell_N)$, and position, $j = (j_1, ..., j_N)$, which is, at the same time, a non-boundary point of the sparse grid. For each of these points, the loss function value, $F(y_{\ell,j})$, is computed.
- **Boundary:** By multiplying $F$ with a sine term:

$$\begin{aligned} \bar{F} : [0,1]^N &\rightarrow \mathbb{R}_0^+ &(41)\\ y &\mapsto \left(\prod_{i=1}^{N} \sin \pi y_i\right) F(y) &(42) \end{aligned}$$

homogeneous Dirichlet boundary conditions are realized in order to reduce the computational effort significantly. The function, $\bar{F}$, is applied to each point, $y_{\ell,j}$, of the sparse grid.
- **Smoothing:** As the function to be minimized is affected with statistical noise, the function, $\bar{F}$, is smoothed and regularized by the methods indicated in Section 2.2.2. Hence, for each point, $y_{\ell,j}$, of the sparse grid, a value, $G(y_{\ell,j})$, of the approximating function is obtained.
- **Interpolation:** The function, $G$, is interpolated from the sparse grid, $\hat{G}_{\hat{\ell}}^N$, on the full grid, $G_{\bar{\ell}}^N$, by the combination technique. Hence, an interpolation model, $\bar{q}$, is obtained for each point, $y \in G_{\bar{\ell}}^N$. As the smoothing and interpolation procedures have been executed for $\bar{F}$, the following division has to be applied for each non-boundary point of the full grid:

$$\forall_{y \in \tilde{G}_{\bar{\ell}}^N}\ q(y) = \frac{\bar{q}(y)}{\prod_{i=1}^{N} \sin \pi y_i} \tag{43}$$

The interpolation model, $q$, is only valid for all $y \in \tilde{G}_{\bar{\ell}}^N$.
- **Discrete minimization:** Determine:

$$y^* := \arg\min_{y \in \tilde{G}_{\bar{\ell}}^N}\ q(y) \tag{44}$$

- **Iteration step $x^k \rightarrow x^{k+1}$:** The ratio:

$$r^k := \frac{F(y^k) - F(y^*)}{q(y^k) - q(y^*)} = \frac{F(y^k) - F(y^*)}{G(y^k) - q(y^*)},\ y^k = \xi(x^k) \tag{45}$$

is determined, and the following three cases are differentiated:

- $r^k \geq \eta_1 \Rightarrow x^{k+1} := \xi^{-1}(y^*) \wedge \Delta_{k+1} := \Delta_k$.
- $r^k \geq \eta_2 > \eta_1 \Rightarrow x^{k+1} := \xi^{-1}(y^*) \wedge \Delta_{k+1} := \gamma_2 \Delta_k$
- $r^k < \eta_1 \Rightarrow x^{k+1} := x^k \wedge \Delta_{k+1} := \gamma_1 \Delta_k$.

Thereby, $\eta_2 > \eta_1 > 0$ and $\gamma_2 > 1 > \gamma_1 > 0$ are global parameters.
Let $k := k + 1$, and go to step 2.

- **Stopping criteria:** The general stopping criterion is:

$$F(x^*) \leq \tau \tag{46}$$

where $\tau > 0$. However, an additional criterion is that the minimum is situated within the hyperdice and not at its boundary. In total, the following three stopping criteria are considered:

(i) $F(x^*) \leq \tau \wedge \xi(x^*) \notin U_0 \cup U_1$
(ii) $F(x^*) \leq \tau \wedge \xi(x^*) \notin U_0 \cup U_1 \wedge \Delta^* < \Delta^{\min}$
(iii) $\exists\, k \in \mathbb{N} \;:\; r^k < \eta_1 \wedge \Delta_k < \Delta^{\min}$.

Thereby:

$$\begin{aligned} U_0 &:= \{y \in [0,1]^N | \; \exists\, i \in 1, ..., N \;:\; y_i \in \{0, 1\}\}, \\ U_1 &:= \{y \in [0,1]^N | \; \exists\, i \in 1, ..., N \;:\; y_i \in \{2^{-\hat{\ell}}, 1 - 2^{-\hat{\ell}}\}\} \end{aligned}$$

Moreover, $\Delta^* := \Delta_k$, where $x^* = x^k$.
If the stopping criteria, (i) and (ii), are fulfilled, then SpaGrOW has converged successfully. Due to $\xi(x^*) \notin U_1$, it is excluded, as well, that the minimum is situated at the boundary of the grid, where the interpolation model, $q$, is valid. Stopping criterion (ii) contains the additional condition, $\Delta^* < \Delta^{\min}$, excluding that improvements can be achieved by local refinements. Hence, this is the ideal stopping criterion.
Stopping criterion (iii) means that SpaGrOW has not led to success, *i.e.*, even by decreasing the trust region, no accurate model, $q$, can be found. In particular, this is the case when the assumptions for the application of the combination technique are not fulfilled, which may be caused by an inaccurate smoothing procedure, wherein the noise has not been filtered out in a sufficient way.

### 3.2.2. Complexity

In the following, the complexity of SpaGrOW is discussed. The present section is organized like Section 3.2.1. , but here, SpaGrOW is discussed with respect to complexity:

- **Initialization:** The effort is only caused by the allocation of the initial variables, like the initial force field parameters, their upper and lower bounds, as well as the maximal, initial and minimal step length.
- **Transformation:** The transformation is performed in an imaginary way only. The sparse grid is constructed directly in step 3. Only the execution of the back-transformation, $\xi^{-1}$, requires a small computational effort.
- **Sparse grid:** The complexity to obtain a sparse grid of the level, $\hat{\ell}$, is $\mathcal{O}\left(2^{\hat{\ell}} \cdot \hat{\ell}^{N-1}\right)$, which is still exponential in the dimension, but especially in the case of higher dimensions, it is significantly

lower than for a full grid of the same level. The computational effort does not only concern the number of grid points to be computed, but in the first instance, the number of loss function evaluations, *i.e.*, simulations.

- **Boundary:** Setting the loss function values to zero at the boundary of the sparse grid does not mean any computational effort. For all other grid points, $2N+1$ ($\mathcal{O}(N)$) multiplications have to be performed, *cf.* Equation (35). Furthermore, there are $N$ sine evaluations. The number of boundary points of an $N$-dimensional sparse grid of the level, $\hat{\ell}$, is $\mathcal{O}\left(N2^{N-1}2^{\hat{\ell}}\right)$. Thereby, $N2^{N-1}$ is the number of edges of an $N$-dimensional hyperdice. Hence, the number of simulations to be executed is reduced to:
$$\mathcal{O}\left[2^{\hat{\ell}}\left(\hat{\ell}^{N-1}-N2^{N-1}\right)\right] \tag{47}$$
This number multiplied by $2N+1$ is the number of required multiplications and multiplied by $N$, the number of sine evaluations required. The reduction of molecular simulations is achieved at the expense of
$$\mathcal{O}\left[N2^{\hat{\ell}}\left(\hat{\ell}^{N-1}-N2^{N-1}\right)\right] \tag{48}$$
multiplications and sine evaluations, a computational effort that can be neglected, if the high amount of computation time for a simulation is opposed.
- **Smoothing:** In the case of most approximation methods, a multivariate linear regression has to be performed with complexity $\mathcal{O}(mM^2+M^3)$, where $m$ is the number of data points and $M$, the number of basis functions (e.g., $M=K$ for RBFs). However, this complexity can often be reduced to $\mathcal{O}(mM^2+M^2)$ or, even, to $\mathcal{O}(1)$ by smart numerical methods, e.g., by a Cholesky factorization, in the case of positive definiteness. In contrast to simulations, the computational effort required for a smoothing procedure is negligible, as well. However, one has to consider that $m$ and $M$ must be large enough, on the one hand, in order to achieve an approximation as accurate as possible, and also small enough, on the other hand, in order to keep the computational effort low and to avoid overfitting. Additional effort appears due to the selection of centroids by the $k$-*means* algorithm, whose convergency speed always depends on the random choice of the initial centroids. The evaluation of the model function is done by a summation of the centroids only.
  Furthermore, the regularization methods require some amount of computational effort, in particular, due to the application of Newton-Lagrange algorithms for the constrained optimization. Please note that most of them have been parameterized, as well, e.g., by cross validation.
- **Interpolation:** In the multilinear interpolation, all adjacent points have to be considered for each grid point. Hence, the interpolation is in $\mathcal{O}\left[(2N+1)\cdot 2^{\hat{\ell}}\cdot\hat{\ell}^{N-1}\right]$.
  After the multilinear interpolation, a division by a sine term has to be executed for each point situated inside the unit hyperdice. As the sine term has already been calculated for each point of the sparse grid, only:
$$\mathcal{O}\left[(2N+1)\left((2^{\hat{\ell}}-1)^N-2^{\hat{\ell}}\hat{\ell}^{N-1}\right)\right] \tag{49}$$
  multiplications and:
$$\mathcal{O}\left[N\left((2^{\hat{\ell}}-1)^N-2^{\hat{\ell}}\hat{\ell}^{N-1}\right)\right] \tag{50}$$

sine evaluations have to be performed. The number of required divisions is equal to $(2^{\hat{\ell}}-1)^N$. Please note that the divisions have to be performed for each inner point of the sparse grid, as well, because only the approximating function, $G$, coincides with the interpolation model, but not $\bar{F}$.

- **Discrete minimization** For the minimization of $q$, a maximal function value of $10^6$ is supposed. For each point on the full grid, a comparison has to be performed, in total, $(2^{\hat{\ell}}-1)^N$ comparisons.
- **Iteration step:** Only a few small operations are necessary in order to perform an iteration step, *i.e.*, simple subtractions, divisions, multiplications and comparisons.
- **Stopping criteria:** Only comparisons have to be performed in order to check the stopping criteria. The decision whether a vector is situated in $U_0 \cup U_1$ or not is made by comparing the components of $y$ with the values zero, one, $h$ and $1-h$. If one component coincides with one of these four values, the procedure is stopped, and $y \in U_0 \cup U_1$ is observed.

### *3.3. Convergency*

In the following, some convergency aspects the SpaGrOW algorithm are considered. In [40], it was proven that the algorithm converges under certain assumptions. Thereby, both smooth and noisy objective functions were considered. In the smooth case, the interpolation error is relevant, and in the noisy case, the approximation error with respect to the original function, *i.e.*, the function without noise, has to be taken into account. Moreover, it is examined to what extent SpaGrOW manages with less simulations than gradient-based methods. In the case of the latter, simulations have to be performed for the gradient components, the entries of the Hessian matrix and the Armijo steps. In the case of SpaGrOW, they have to be executed for the sparse grid points and the Trust Region steps. The steepest descent algorithm and the conjugate gradient methods required significantly less simulations for the gradient than SpaGrOW for the sparse grid: for $N = 4$, four simulations are required for the gradient and nine for a sparse grid of the level 2. As for the step length control, it can be observed that both gradient-based methods and SpaGrOW mostly need one Armijo or, respectively, one Trust Region step at the beginning of the optimization, but many more close to the minimum. The reason for the reduction of computational effort in the case of SpaGrOW lies in the lower number of iterations, not in the lower number of function evaluations per iteration. The advantage is that the step length is determined before, so that, especially at the beginning of the optimization, large steps can be realized. For smooth functions, the combination technique delivers small interpolation errors in most cases, also when the trust region is quite large, but always a descent direction, mostly leading to the boundary of the actual trust region.

Close to the minimum, both approaches have the drawback that after a high number of step length control iterations, *i.e.*, after a large computational effort, only marginal improvements in the loss function values are observed. However, due to the statistical noise, the minimum can never be predicted exactly. At some point, the minimization has to be stopped, and the actual result has to be evaluated. However SpaGrOW is capable of searching for a smaller loss function value in more than one direction, due to the grid approach. Furthermore, its modeling approach increases the probability to get close to the minimum than gradient-based methods, as motivated in Section 1.

As the interpolation and smoothing errors are essential for the convergency of SpaGrOW, they are introduced and discussed in the following.

3.3.1. Interpolation Error

Under certain assumptions [24], the interpolation error for a sparse grid of the level, $\hat{\ell}$, is of the order $\mathcal{O}\left(h_{\hat{\ell}}^2 \left(\log(h_{\hat{\ell}})^{-1}\right)^{\hat{\ell}-1}\right)$. First, the two-dimensional case is considered again. If the difference $u - u_{i,j}$, where $u_{i,j} \in T_{i,j}$, $i + j \in \{\hat{\ell}, \hat{\ell}+1\}$, meets point-wise an asymptotic error expansion, *i.e.*,

$$u - u_{i,j} = C_1(h_i)h_i^2 + C_2(h_j)h_j^2 + D(h_i, h_j)h_i^2 h_j^2 \tag{51}$$

where $\forall_i \, |C_1(h_i)| \leq \kappa$, $\forall_j \, |C_2(h_j)| \leq \kappa$ and $\forall_{i,j} \, |D(h_i, h_j)| \leq \kappa, \; \kappa > 0$, then the interpolation error can be estimated as follows:

$$|u - \hat{u}_{\hat{\ell}}^c| \leq \left(1 + \frac{5}{4}\log\left(h_{\hat{\ell}}^{-1}\right)\right)\kappa h_{\hat{\ell}}^2 = \mathcal{O}\left(h_{\hat{\ell}}^2 \log(h_{\hat{\ell}})^{-1}\right) \tag{52}$$

For the $N$-dimensional case, an analogous asymptotic error expansion delivers the following formula:

$$|u - \hat{u}_{\hat{\ell}}^c| \leq \left(\sum_{l=0}^{N-1} \chi_l \left(\log\left(h_{\hat{\ell}}^{-1}\right)\right)^l\right)\kappa h_{\hat{\ell}}^2 = \mathcal{O}\left(h_{\hat{\ell}}^2 \left(\log(h_{\hat{\ell}})^{-1}\right)^{\hat{\ell}-1}\right) \tag{53}$$

Thereby, $u \in S_{\bar{\ell}}^0, \; \bar{\ell} = (\hat{\ell}, ..., \hat{\ell}) \in \mathbb{N}^N$ and $\chi_l \in \mathbb{N}, \; l = 0, ..., \hat{\ell} - 1$.

Please note that in order to guarantee the existence of such an asymptotic error expansion, the exact solution, $u$, must fulfill certain continuity and smoothness conditions. As $u$ is supposed to reproduce $F$ exactly on the sparse grid and as accurately as possible between the sparse grid points, these assumptions have to be transferred to $F$. This motivates again the need for a smoothing procedure in the case of statistical noise. In most cases, the existence of an asymptotic error expansion cannot be proven *a priori*. However, the combination technique was shown to deliver very good results in practice [24].

The interpolation error can be estimated above as follows:

**Theorem 11** (Estimate for the interpolation error)**.** *Let $\Delta > 0$ be the size of the hyperdice, $\mathcal{W}^{\Delta}(x)$, with center $x \in \mathbb{R}^N$, where a sparse grid of the level, $\hat{\ell}$, is defined. Let the approximated function, $G : \mathbb{R}^N \to \mathbb{R}_0^+$, be given on the sparse grid and interpolated by the model function, $q : B_{\Delta}(0) \to \mathbb{R}$, on a full grid of the level, $\ell := (\hat{\ell}, ..., \hat{\ell}) \in \mathbb{N}^N$, using the combination method. Then, for the interpolation error, $\varepsilon = |u - \hat{u}_{\hat{\ell}}^c|$, in Inequality (53), it holds for $u := G$ and $\hat{u}_{\hat{\ell}}^c := q$:*

$$\forall_{y \in \mathcal{W}^{\Delta}(x)} \, |\varepsilon(y)| \leq \kappa_{\varepsilon}(\hat{\ell}) f_{\hat{\ell}}^{\varepsilon}(\Delta) \tag{54}$$

*where $\kappa_{\varepsilon} : \mathbb{N} \to \mathbb{R}^+$ and $f_{\hat{\ell}}^{\varepsilon} : \mathbb{R}^+ \to \mathbb{R}^+$ with $\lim_{\hat{\ell} \to \infty} \kappa_{\varepsilon}(\hat{\ell}) = 0$ and $\lim_{\Delta \to 0} f_{\hat{\ell}}^{\varepsilon}(\Delta) = 0$. The function, $f_{\hat{\ell}}^{\varepsilon}$, is continuous. Moreover, $\tilde{f}_{\hat{\ell}}^{\varepsilon} := \frac{f_{\hat{\ell}}^{\varepsilon}}{\Delta} : \mathbb{R}^+ \to \mathbb{R}^+$ is continuous, as well, with $\lim_{\Delta \to 0} \tilde{f}_{\hat{\ell}}^{\varepsilon}(\Delta) = 0$.*

*Proof.* Let $y \in \mathcal{W}^{\Delta}(x)$ be an arbitrary point in the hyperdice. It holds $h_{\hat{\ell}} = 2^{1-\hat{\ell}}\Delta$. Following Inequality (53), there exist constants $\chi_l > 0, \; l = 0, ..., N-1$ and a $\kappa > 0$, so that:

$$\begin{aligned} |\varepsilon(y)| &\leq \left(\sum_{l=0}^{N-1} \chi_l \left(\log\left(2^{\hat{\ell}-1}\Delta^{-1}\right)\right)^l\right)\kappa 2^{2-2\hat{\ell}}\Delta^2 \\ &= \left(\sum_{l=0}^{N-1} \chi_l \left(\hat{\ell} - 1 - \log\Delta\right)^l\right)\kappa 2^{2-2\hat{\ell}}\Delta^2 \end{aligned}$$

$$\leq \underbrace{N \cdot \max_{l=0}^{N-1} \chi_l \cdot \kappa 2^{2-2\hat{\ell}}}_{:=\kappa_\varepsilon(\hat{\ell})} \cdot \underbrace{\max_{l=0}^{N-1} \left(\hat{\ell} - 1 - \log \Delta\right)^l \cdot \Delta^2}_{:=f_{\hat{\ell}}^{\varepsilon}(\Delta)}$$

$$= \kappa_\varepsilon(\hat{\ell}) f_{\hat{\ell}}^{\varepsilon}(\Delta) \tag{55}$$

It holds $\lim_{\hat{\ell}\to\infty} \kappa_\varepsilon(\hat{\ell}) = 0$, and also, because of l'Hospital's rule, $\lim_{\Delta\to 0} f_{\hat{\ell}}^{\varepsilon}(\Delta) = 0$. Due to $\tilde{f}_{\hat{\ell}}^{\varepsilon}(\Delta) = \max_{l=0}^{N-1} \left(\hat{\ell} - 1 - \log \Delta\right)^l \cdot \Delta$, it follows also from l'Hospital's rule that $\lim_{\Delta\to 0} \tilde{f}_{\hat{\ell}}^{\varepsilon}(\Delta) = 0$. □

### 3.3.2. Smoothing Error

Let $||\cdot||$ be one of the two norms, $||\cdot||_{L_2}$ or $||\cdot||_{L_\infty}$. The smoothing error, $\mu$, is the error with respect to $||\cdot||$ between the original function, $F = \tilde{F} - \Delta F$, without noise and the approximating function, $G$. Thereby, $\tilde{F}$ is the noisy function. It holds:

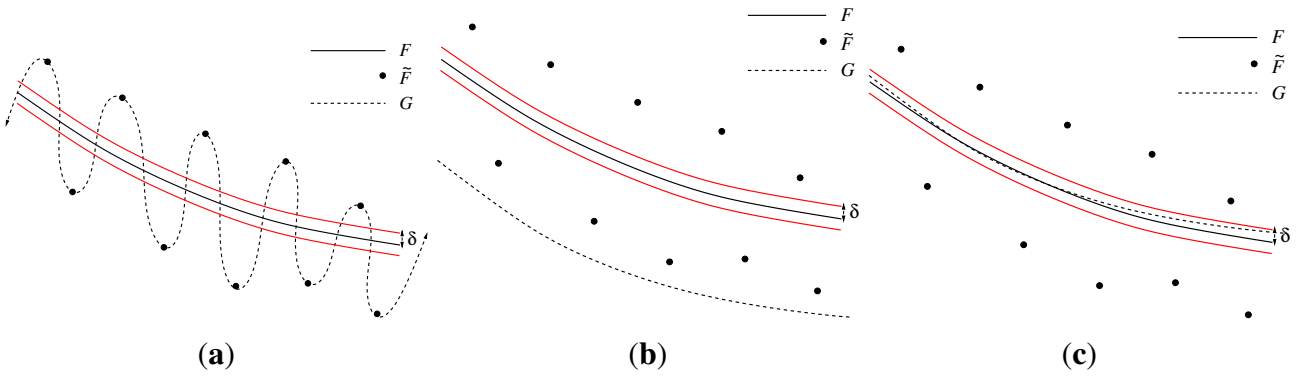
$$\mu := ||F - G|| = ||(F + \Delta F) - G - \Delta F|| = ||\tilde{F} - G - \Delta F|| \leq |\vartheta| + ||\Delta F|| \tag{56}$$

where $\vartheta$ denotes the training error. If $\Omega$ is a trust region, then define $\forall_{x\in\Omega}\, \vartheta(x) := \tilde{F}(x) - G(x)$. In the ideal case, $\vartheta(x) = \Delta F_x$, it holds $\mu(x) = 0$. Otherwise, for $0 < \delta << |\Delta F_x|$, the following inequation must hold:

$$|\mu(x)| = |\Delta F_x - \vartheta(x)| \leq \delta \tag{57}$$

This means that the training error, $\vartheta$, must not be too small. Figure 10a shows an overfitted model, $G$, which reproduces exactly the oscillations produced by the noise. In this case, $\vartheta = 0$, but $|\mu(x)| = |\Delta F_x| >> \delta$ for certain $x \in \Omega$. On the other hand, the training error must not be too high. In Figure 10b, it holds $\forall_{x\in\Omega}\, |\mu(x)| \approx |\Delta F_x| >> \delta$. Only Figure 10c depicts a feasible case; here, the training error is in the same order as the noise, and it holds $|\mu| \leq \delta$.

**Figure 10.** Approximation models with different training errors, $\vartheta$, where the function, $G$, approximates the noisy function, $\tilde{F} = F + \Delta F$, overfitted model with $\vartheta = 0$ (**a**); model, where $\vartheta$ is too high (**b**); and feasible model with $|\Delta F - \vartheta| \leq \delta$ (**c**). In the ideal case, it holds $\vartheta(x) = \Delta F_x$.




(**a**) (**b**) (**c**)

In order to keep the smoothing error low enough, a smoothing procedure has to be applied, which filters out the noise and reproduces the loss function, at least on the sparse grid, as exactly as possible.

As a smoothing procedure can only be evaluated on a sparse grid, due to the high amount of computation time for molecular simulations, the condition, $|\mu| \leq \delta$, is considered only on the sparse grid. It does not matter if $G$ has oscillations between the sparse grid points, because the piecewise-multilinear sparse grid interpolation is not capable of modeling them anyway.

The following theorem is essential for the convergency proof of SpaGrOW and gives an estimate for the smoothing error in the case of positive definite RBFs:

**Theorem 12** (Estimate for the smoothing error)**.** *Let $\Delta > 0$ be the size of the hyperdice, $\mathcal{W}^{\Delta}(x)$, with the center, $x \in \mathbb{R}^N$, in which a sparse grid of the level, $\hat{\ell}$, is constructed. Let the loss function, $F : \mathbb{R}^N \to \mathbb{R}_0^+$, be given on the sparse grid and approximated by the function, $G : \mathbb{R}^N \to \mathbb{R}$. Suppose that the approximation be stable. Then, the smoothing error $\mu$ from Equation (56) can be estimated as follows:*

$$\exists\, \kappa_\mu > 0 \;:\; \forall_{y \in \mathcal{W}^{\Delta}(x)} \; |\mu(y)| \leq \kappa_\mu f^\mu(\Delta) \tag{58}$$

*where $f^\mu : \mathbb{R}^+ \to \mathbb{R}^+$ is continuous with $\lim_{\Delta \to 0} f^\mu(\Delta) = 0$. Furthermore, $\tilde{f}(\mu) := \frac{f^\mu}{\Delta} : \mathbb{R}^+ \to \mathbb{R}^+$ is continuous, as well, with $\lim_{\Delta \to 0} \tilde{f}^\mu(\Delta) = 0$.*

*Proof.* Due to the stability of the smoothness, Inequality (29) holds, where $\Omega = \mathcal{W}^{\Delta}(x)$ and $\mathcal{X}$ is the sparse grid. The fill distance on the sparse grid and $\Delta$ only differ by a constant, $\omega > 0$, *i.e.*, $\Delta_{\Omega,\mathcal{X}} = \omega\Delta$. For $\kappa_\mu := \kappa$ and $f^\mu(\Delta) := h(\omega\Delta) = h(\Delta_{\Omega,\mathcal{X}})$, the estimate for the smoothing error in Equation (58) follows directly. □

**Remark 13.** *Following Corollary 10, estimate Equation (58) is given in the case of a smoothing procedure based on Gaussian RBFs.*

**Remark 14.** *For Theorem 12 and the full convergency proof, it is irrelevant whether the original function, $F : \mathbb{R}^N \to \mathbb{R}_0^+$, or the transformed function, $\bar{F} : [0,1]^N \to \mathbb{R}_0^+$, is approximated by $G : \mathbb{R}^N \to \mathbb{R}$. The function to be smoothed only has to be continuous within the trust region. Please note that in the case of $\bar{F}$, the function, $G$, can have negative values, as $\bar{F}$ is equal to zero at the boundary of the trust region. For the original function, $G(x) \geq 0$ can be assumed, due to Theorem 12, when $\Delta$ is small enough. For $\bar{F}$, this can be assumed, as well. Otherwise, consider a translation that does not have any impact on either the approximation or the minimization.*

The convergency proof executed for SpaGrOW was related to a general convergency proof for derivative-free Trust Region methods [41]. However, the Trust Region method used in that paper is based on an interpolation with Newtonian fundamental polynomials. Hence, the partial proofs cannot be transferred, but have to be developed anew. Another crucial difference consists in the assumption for the loss function to be at least two times continuously differentiable with a bounded Hessian norm, which cannot be made in the case of SpaGrOW. The detailed convergency proof was performed in [40].

### *3.4. Speed of Convergency Compared to Gradient-Based Methods*

The speed of convergency of SpaGrOW is discussed in the following, also with respect to statistical noise. As already mentioned, the trust region size, $\Delta$, may not be too small, due to the noise. However, the convergency proof is based on the choice of a small $\Delta$. This dilemma, which is also present in the case of gradient-based methods whenever two adjacent points are required for the computation of a partial derivative, leads to the need for an optimal parameterization of SpaGrOW. To achieve a high speed of convergency, primarily at the beginning of the optimization, the choice of a large $\Delta$ is required without hurting one of the assumptions for the convergency proof, *cf.* [40]. In the following, some heuristic considerations are made in this regard. Thereby, the index, $_g$, refers to gradient-based descent methods, the index, $_H$ to descent methods using a Hessian and the index, $_S$ to SpaGrOW. Furthermore, let $\bar{M}$ be the average number of function evaluations per iteration and $\bar{l}$, the average number of Armijo or Trust Region steps. Then, it holds:

$$\bar{M}_g = N + \bar{l}_g \tag{59}$$

$$\bar{M}_H = N + \frac{N(N+1)}{2} + \bar{l}_H \tag{60}$$

$$\bar{M}_S = 2N \cdot \bar{l}_S \tag{61}$$

The drawback of SpaGrOW lies in the multiplicative dependency of $\bar{M}_S$ on $\bar{l}_S$. This is due to the fact that a new sparse grid is used at each iteration step. However, at the beginning of the optimization, $\bar{l}_g = \bar{l}_H = \bar{l}_S = 1$ is assumed. Then, it holds $\bar{M}_g < \bar{M}_S < \bar{M}_H$. Hence, SpaGrOW requires less iterations on average than a method based on Hessians. However, in total, it has to manage with less iterations than a gradient-based method requiring less function evaluations per iteration: Let $k$ be the general number of iterations; then it must hold: $k_S < k_g$. If:

$$k_S < \frac{N+1}{2N} k_g \tag{62}$$

SpaGrOW needs less iterations and function evaluations than a gradient-based method. Now, the question arises, how this can be steered. By choosing an initial $\Delta_0$ (and also $\gamma_1$) that is large enough, a faster convergency can be achieved.

In the following, the initial phase of an optimization process is considered, and a short comparison between SpaGrOW and a gradient-based descent method is pointed out, *i.e.*, it is discussed under what conditions the speed of convergency is significantly higher in the case of SpaGrOW. Please note that *at the beginning of the optimization* means here that the number of trust region or Armijo steps is equal to one at each iterations and that $k_g$ is chosen, so that the number of Armijo steps for $x^{k_g}$ is still equal to one and for $x^{k_g+1}$, greater than one. Furthermore, choose $k_S$, so that $||x^{k_S} - x^0|| \leq ||x^{k_g} - x^0||$, the size of the trust region is equal to $\Delta_0$ and $x^{k_g} \in \Omega_k$. Hence, $x^{k_g} \notin \Omega_{k-1}$. This means that both $x^{k_S}$ and $x^{k_g}$ are reached by SpaGrOW with $k_S$ steps of length $\Delta_0$, which is depicted in Figure 11. The distance between $x^0$ and $x^{k_S}$ is equal to $k_S\Delta_0$. It holds:

$$k_S\Delta_0 \leq ||x^{k_g} - x^0|| \tag{63}$$

If:

$$\Delta_0 > \frac{||x^{k_g} - x^0||}{\underbrace{\frac{N+1}{2N} k_g}_{>1, \text{ if } k_g \geq 2}} \tag{64}$$

*i.e.*, if:

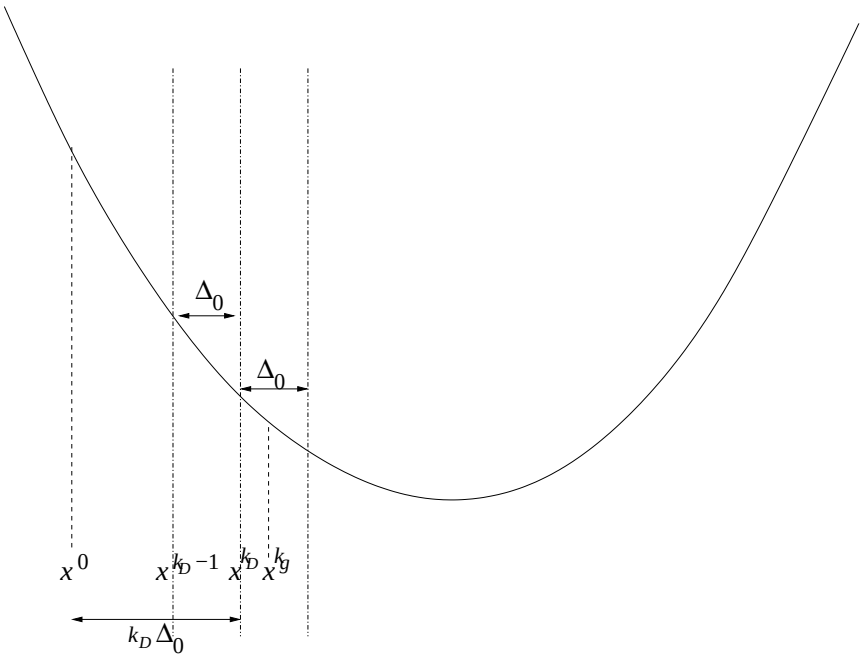
$$\Delta_0 > \zeta ||x^{k_g} - x^0|| \tag{65}$$

for a maximal $\zeta \in (0, 1)$; then, Inequality (62) follows. In practice, a realistic case is $k_g = 4$. If:

$$\zeta := \frac{4N}{(N+1)k_g} = \frac{N}{N+1} < 1 \tag{66}$$

is chosen, SpaGrOW requires less than 50% of the iterations and function evaluations required by the gradient-based descent method. Hence, a reduction of computation time by a factor of two is plausible at the beginning of the optimization process.

**Figure 11.** Speed of convergency of SpaGrOW at the beginning of the optimization. For an appropriate choice of the size of the initial trust region, $\Delta_0$, the number of iterations in the case of SpaGrOW ($k_S$) is significantly smaller than in the case of a gradient-based method ($k_g$). It is realistic that the number of iterations and function evaluations can be reduced by a factor of two.



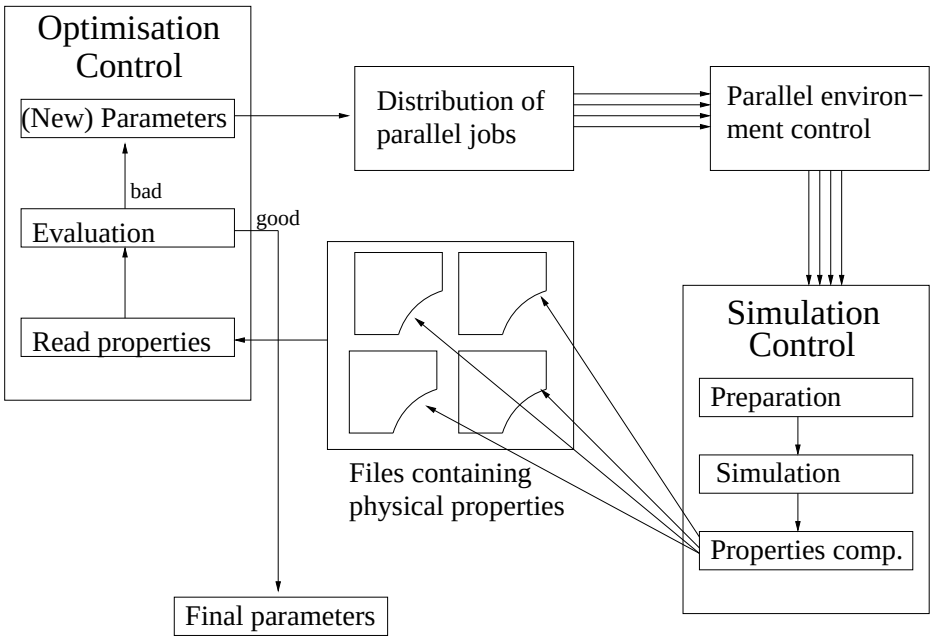

## 4. Practical Evaluation and Results

The methodology of SpaGrOW was implemented in *python* (version 2.4.3) and is modularly constructed. The software consists of a main control script and secondary control scripts for specific parts of the algorithm, e.g., for the sparse grid interpolations and the control of the smoothing procedures, which were implemented in *S+* (version 2.1.10), scripting language related to the *R Project for Statistical Computing* [42].

The implementation of SpaGrOW contains the full algorithm described in the previous section and acts as an interface between optimization and simulation, providing all necessary control routines for both tasks. On the optimization side, the method starts with an initial guess and evaluates the loss function based on the results of a simulation. If one of the stopping criteria of SpaGrOW is fulfilled, the optimization workflow terminates. Otherwise, the parameters are updated by SpaGrOW.

On the simulation side, a control script calls the simulation tool performing all preparation routines and computing the trajectory, as well as the desired physical target properties. The latter are passed on to the optimization workflow of SpaGrOW. In the case of a simultaneous optimization of properties at different temperatures, the respective simulations are executed in parallel. In this case, a script distributing $K$ jobs at $K$ temperatures is called. A script controlling the parallel environment and the simulation control script are called $K$ times. If $m = n/K$ physical properties are fitted, the result of each job consists of $m$ properties files. Figure 12 shows both the optimization and the simulation side and how they interact with each other in the case of parallel jobs at different temperatures.

**Figure 12.** Technical realization of optimization procedure for physical target properties at different temperatures. If the simulated properties do no coincide well with their experimental reference data, the optimization control script—depicted on the left—passes the current force field parameters on to a distribution control script, which submits parallel jobs at different temperatures. Then, a parallel environment control script is executed, and a simulation control script is called, which performs the following three tasks: preparation routines, the simulation itself and the computation of the simulated target properties. The properties are written into separate files, which are read by the optimization control script. Finally, the loss function is evaluated and the workflow continues.




The simulations were performed on a parallel computer cluster with 215 available nodes, where each node is provided with two Intel-Nehalem-EP-Quadcore processors (Xeon X5550) with 24 GB of

main storage, which are connected by a fast QDR infiniband interconnect with a 40-Gb/s Double Date Rate (DDR).

In this section, SpaGrOW is evaluated in practice and applied to molecular simulations as described above. The questions to be answered are the following:

- Which smoothing and regularization procedure are most suitable?
- Is SpaGrOW capable of saving simulations in comparison to gradient-based methods?
- How close does SpaGrOW get to the minimum?

### *4.1. Selection of Smoothing and Regularization Methods*

As molecular simulations are extremely time-consuming, an analytical model replacing them was used for the selection. In previous work [22], a similar assessment has already been performed for gradient-based methods. Vapor-Liquid Equilibrium (VLE) data, like the saturated liquid density, $\rho_l$, the enthalpy of vaporization, $\Delta_v H$, and the vapor pressure, $p_\sigma$, can be evaluated directly as functions of certain force field parameters. These functions were determined by [43] by nonlinear regression using a high amount of simulation data for two-centered Lennard-Jones (LJ) fluids with a quadrupolar moment. Here, nitrogen was considered as an example for this kind of fluid. The model parameters to be optimized were the elongation, $L$, of the two LJ sites, the quadrupolar moment, $Q^2$, and the two LJ parameters, $\sigma$ and $\varepsilon$. In order to mimic molecular simulations, uniformly distributed artificial uncertainties ($0.5\%$ for the density and $3\%$ for the pressure) were added for the simultaneous optimization of $\rho_l$ and $p_\sigma$ at six different temperatures, $T/\mathrm{K} \in \{65, 75, 85, 95, 105, 115\}$. As in [22], the weights of the properties in the loss function were all equal to one, because all properties were considered as homologous. As the simulation data were noisy, ten statistically independent random replicates were performed for each optimization run, whose results were averaged.

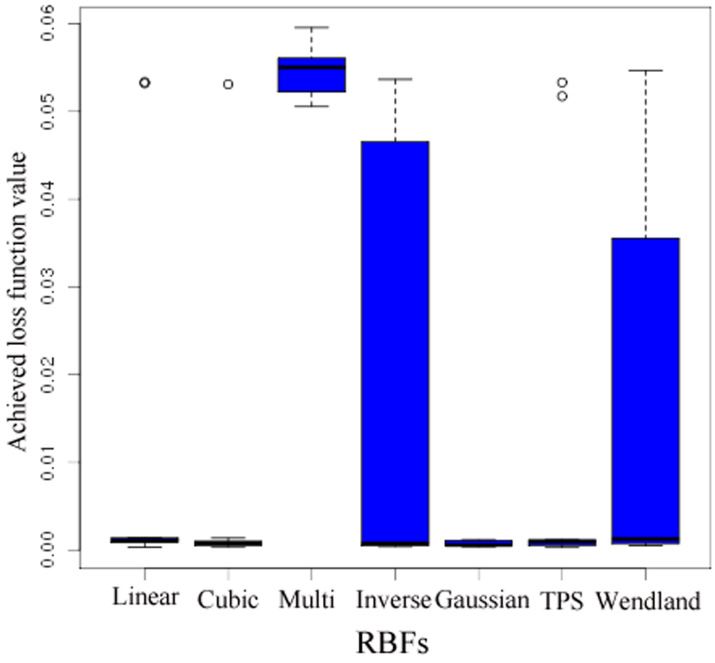
Due to theoretical considerations, the tendency consists in selecting positive definite RBFs, in particular, Gaussian RBFs. In the following, it is shown that this is also a good choice in practice.

In order to evaluate, whether a smoothing or regularization procedure is appropriate for SpaGrOW, the behavior of the algorithm combined with each preprocessing procedure is analyzed. Thereby, both efficiency and robustness with respect to noise are considered. Table 2 shows the candidates and their abbreviations.

**Table 2.** Candidates for smoothing and regularization procedures within SpaGrOW together with their abbreviations.

| **Smoothing procedures** | **Regularization procedures** |
|---|---|
| Radial Basis Functions | Least squares |
| | Naive Elastic Nets (NENs) |
| | $\alpha = 0$: Least Absolute Shrinkage and Selection Operator (LASSO) |
| | $\alpha = 1$: Ridge Regression |
| Linear property approximation (Lipra) | Multivariate Adaptive Regression Splines (MARS) |

**Figure 13.** Box plots of the loss function values achieved by SpaGrOW in combination with a smoothing procedure based on Radial Basis Functions (RBFs). The RBFs were the linear, cubic, multiquadric (Multi), inverse multiquadric (Invers), Gaussian, thin plate spline RBF (TPS) and a Wendland function. Suitable RBFs were only the linear, cubic, Gaussian and thin plate spline RBF.



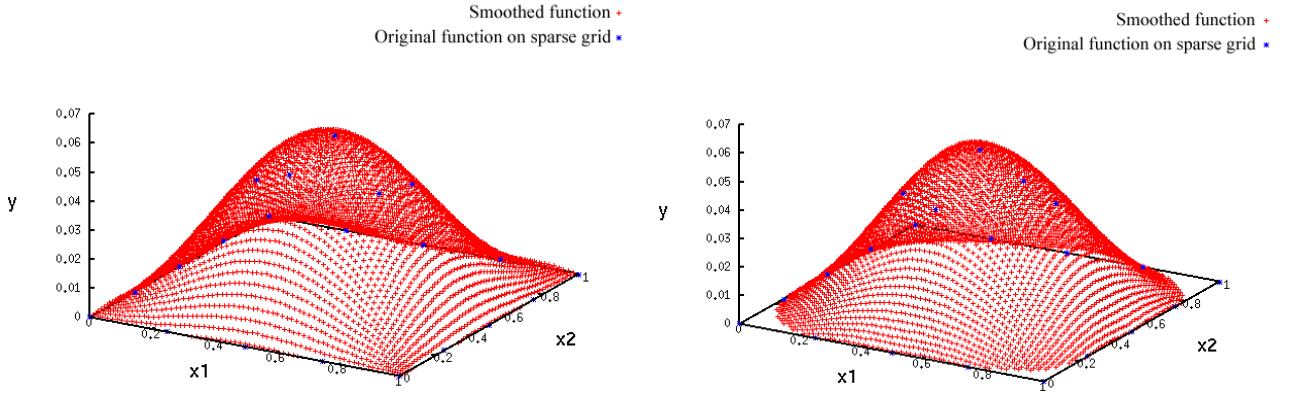

**Selection of the Best Smoothing Procedure** It was already motivated that a smoothing procedure is indispensable, whenever noisy loss function values are present. A detailed analysis has shown that certain RBFs deliver better results by far than others. Suitable RBFs are the linear, cubic, Gaussian and thin plate spline RBF, *cf.* Section 2.2.2. . The multiquadric functions were not reliable. Moreover, Wendland functions were considered, as well. Figure 13 shows box plots for the loss function values achieved by SpaGrOW combined with the different RBFs, which is a criterion for the quality of convergency. The results over ten statistically independent replicates are indicated. The lower the loss function achieved was, the closer got the algorithm to the minimum. The smallest loss function values were achieved robustly by the four RBFs mentioned above. A Gaussian RBF is selected here for the following reasons:

- No outliers were detected.
- A more accurate analysis of the approximation error has shown that the smoothing procedures based on cubic and thin plate spline RBFs reproduced the function, $\bar{F}$, *cf.* Section 3.2.1, worse than the one based on Gaussian RBFs. Figure 14 depicts the results for the Gaussian RBFs (Figure 14a) and the thin plate spline RBFs (Figure 14b) for the two-dimensional case: The approximating function and the original function values of $\bar{F}$, *cf.* Equation (35), on a sparse grid of the level 2 are plotted *versus* $\xi(Q^2)$ and $\xi(L)$, *cf.* Equation (39). The LJ parameters, $\sigma$ and $\varepsilon$, were fixed. As can be seen, a smoothing procedure based on thin plate spline RBFs reproduced $\bar{F}$ at the boundary of the unit square in a very bad fashion, whereas Gaussian RBFs delivered a good approximation on the complete unit square.

- The Gaussian RBF is the only of the four RBFs mentioned above that is positive definite, *i.e.*, the selection of Gaussian RBFs is also founded theoretically according to the considerations in Section 2.2.4.

Please note that linear RBFs delivered good approximations at the beginning of the optimization, as well, which was not surprising, because the steepest descent method was also very successful [22]. For higher dimensions, the Lipra method, *i.e.*, a quadratic approximation of the loss function, could be convincing.

**Figure 14.** Approximations on the unit square, $[0,1]^2$, based on Gaussian RBFs (**a**) and thin plate spline RBFs (**b**). The blue points mark the original (noisy) function values of $\bar{F}$ on the sparse grid. It holds $x1 = \xi(Q^2),\ x2 = \xi(L)$ and $y = \bar{F}(x1, x2)$. The smoothing procedure based on thin plate spline RBFs reproduces $\bar{F}$ at the boundary of the unit square in a very bad fashion.

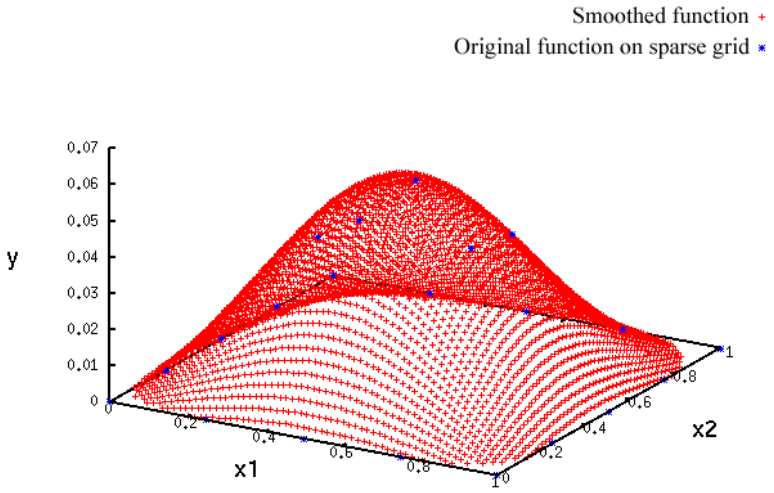



(**a**) (**b**)

**Selection of the Best Regularization Method** As already mentioned, the selection of the best regularization method can only be achieved by practical evaluations. Candidates are least squares, NENs (LASSO for $\alpha = 0$ and Ridge Regression for $\alpha = 1$), as well as MARS.

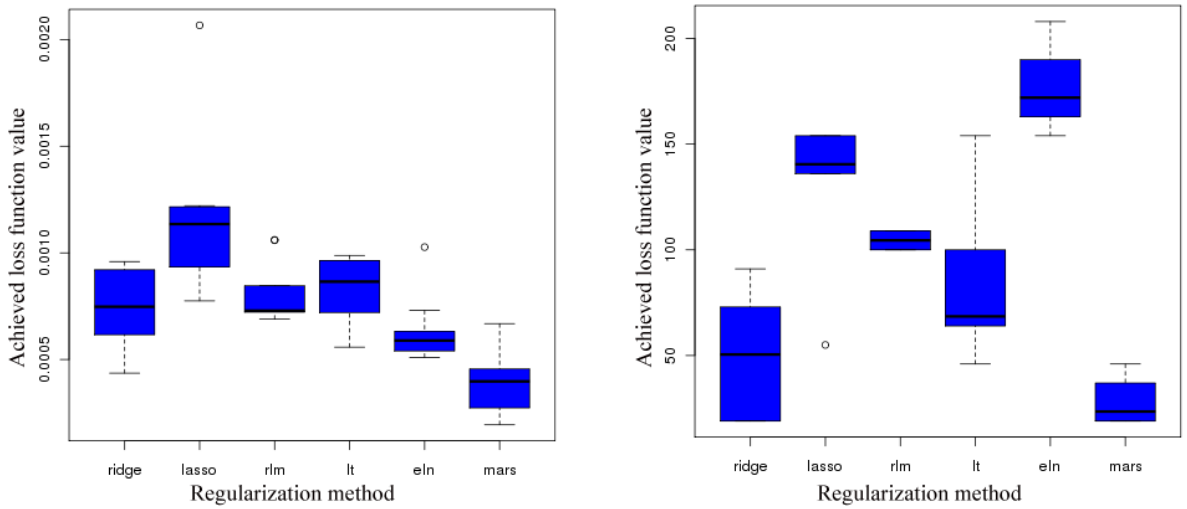
The regularization methods were evaluated in combination with the selected smoothing procedure based on Gaussian RBFs. The application of least squares is the same as not using any regularization method. All other regularization methods could improve the results achieved with least squares only. The LASSO algorithm performs a variable selection, *i.e.*, it tends to detect outliers by mistake. Hence, the model obtained by LASSO was often under-fitted. In contrast, the Ridge Regression estimator was more suitable for the present task, as it is a compromise between least squares, which often lead to overfitting, and the LASSO, which often leads to under-fitting. Figure 15 shows an approximation based on Gaussian RBFs in combination with LASSO. As can be seen, the function to be approximated was reproduced in a bad fashion at the boundary.

**Figure 15.** Approximations of $\bar{F}$ on the unit square, $[0,1]^2$, based on Gaussian RBFs, combined with a LASSO regularization. The blue points mark the original (noisy) function values of $\bar{F}$ on the sparse grid. It holds $x1 = \xi(Q^2),\ x2 = \xi(L)$ and $y = \bar{F}(x1, x2)$. At the boundary of the unit square, the function is reproduced in a bad fashion.




An NEN with $\alpha = 0.7$ delivered an even better quality of convergence; however, the computational effort to optimize $\alpha$ was too high compared to the benefit achieved. The application of an NEN with $\alpha \notin \{0, 1\}$ is not worthwhile for the present task.

**Figure 16.** Box plots of the loss function values (**a**) and of the number of function evaluations (**b**) resulting from the application of SpaGrOW combined with a smoothing procedure based on Gaussian RBFs and different regularization methods: Ridge Regression, LASSO, a weighted linear regression (rlm), an RBF approximation with an additional linear term (lt), an NEN (eln) with $\alpha = 0.7$ and MARS. It becomes clear that MARS is the algorithm to select for regularization. Ridge Regression is reliable, as well.



The best regularization method is the MARS algorithm, not only with respect to robustness and quality of convergency, but also with respect to the number of function evaluations: Figure 16 shows box plots of all regularization methods applied. Figure 16a shows the loss function values achieved and Figure 16b the number of function evaluations. Besides the methods considered here, two other ones were tried: a weighted linear regression based on $M$-estimators and an RBF approximation with an additional linear term. As can be seen, the MARS algorithm delivers the best results. Hence, it is selected as regularization method for SpaGrOW. However, Ridge Regression is reliable, as well, and suggested as the alternative. The NEN with $\alpha = 0.7$ achieved very low loss function values in a very robust way, but it always required more than a hundred function evaluations.

For Lipra, the least square estimator was the best regularization method. All other methods biased the quadratic approximation in a highly inappropriate way.

To summarize, Gaussian RBFs in combination with MARS and, in particular, for $N \geq 5$, the Lipra method together with the least square estimator have turned out to be most suitable for the present task and are implemented in SpaGrOW for this reason.

### *4.2. Application of SpaGROW to Molecular Simulations*

Finally, SpaGrOW is applied to molecular simulations. Thereby, it is compared to gradient-based methods with respect to computational effort. Additionally, for an eight-dimensional problem, the Lipra method is evaluated, and it is analyzed how close SpaGrOW can get to the minimum.

#### 4.2.1. Comparison to Gradient-Based Methods with Respect to Computation Time: Benzene and Ethylene Oxide

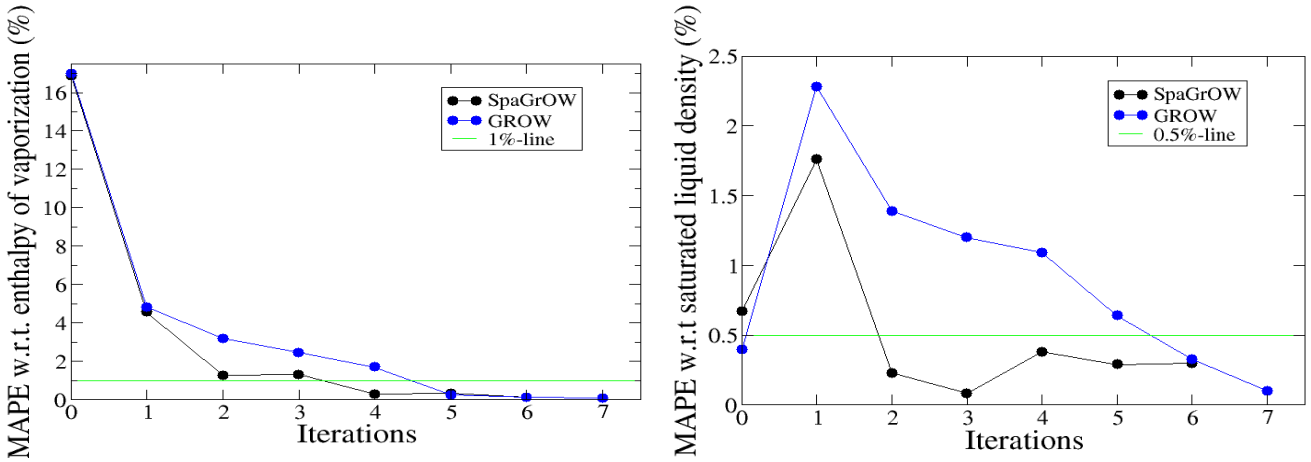
In the following, SpaGrOW is compared to GROW with respect to computational effort on the basis of two applications: benzene and ethylene oxide.

**Benzene** Benzene ($C_6H_6$) is a quite simple molecule, because of its symmetric structure and the fact that it does not possess a permanent dipolar moment. Furthermore, benzene has two chemically independent atom types only. Hence, the force field parameterization for benzene was deemed to be a relatively easy task. However, it is still challenging, because of the $\pi$ interactions.

Figure 17 shows the comparison between SpaGrOW and GROW with respect to the computational effort required within the respective optimization procedures. The target observables were the enthalpy of vaporization (Figure 17a) and the saturated liquid density (Figure 17b), considered at three different temperatures. The values indicated on the $y$-axis are the Mean Absolute Percentage Errors (MAPE), *i.e.*, the absolute deviations from the respective experimental reference data in %, averaged over the range of temperatures. The experimental saturated liquid density was taken from [44] and the enthalpy of vaporization from the NISTdatabase [45]. The simulations performed were molecular dynamics simulations in the $NpT$ ensemble executed with the software tool, *GROMACS* [46]. The non-bonded potential energy was computed by *Moscito* [47] using the trajectories collocated by GROMACS. Please note that the experimental target observables were VLE define data and that the simulated properties were only approximations, due to the lack of an explicit gas phase, which was assumed to be ideal. On

the computer cluster mentioned above, three to four hours were required for the simulation of 1,000 benzene molecules for 2 ns using a time step of 2 fs. For SpaGrOW, the following variables were chosen: $\eta_1 = 0.2,\ \eta_2 = 0.7,\ \gamma_1 = 0.5,\ \gamma_2 = 1.1$, and $\Delta_0 = 0.3 \times \Delta_{\max}$. The force field parameters were $\sigma(\mathrm{H}), \sigma(\mathrm{C}), \varepsilon(\mathrm{H})$ and $\varepsilon(\mathrm{C})$. The initial parameters were taken from previous work [23]. Thereby, the saturated liquid density and the self-diffusion coefficient were optimized at the vapor-liquid coexistence curve. The feasible domain was defined as follows: $\sigma$ was changed by no more than 30% and $\varepsilon$ by no more than 80%. As it was a four-dimensional optimization problem, Gaussian RBFs were chosen for the smoothing and the MARS algorithm for the regularization procedure.

**Figure 17.** Mean Absolute Percentage Errors (MAPE) values with respect to $\Delta_v H$ (**a**) and $\rho_l$ (**b**) for benzene during the SpaGrOW optimization in comparison to GROW. The smoothing procedure was based on Gaussian RBFs in combination with MARS. The force field parameters to be optimized were $\sigma(\mathrm{H}), \sigma(\mathrm{C}), \varepsilon(\mathrm{H})$ and $\varepsilon(\mathrm{C})$. A faster convergency of SpaGrOW could be confirmed.



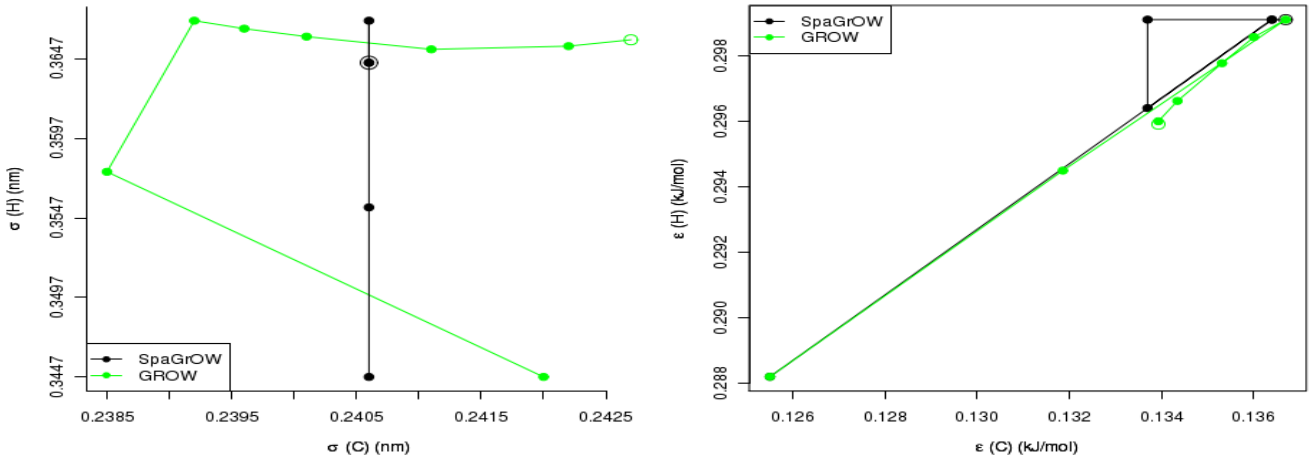

(**a**) (**b**)

GROW needed seven iterations in total: five steepest descent and two conjugate gradient iterations. In contrast, SpaGrOW required six iterations only. Please note that an optimal force field with respect to $\Delta_v H$ and $\rho_l$ was already achieved after four iterations, *i.e.*, both target observables were equal to their experimental reference data up to statistical uncertainties for all temperatures. Typical statistical uncertainties for $\Delta_v H$ and $\rho_l$ are 1% and 0.5%, respectively, *cf.* e.g., [19]. For GROW, this was the case after seven iterations. The number of function evaluations, *i.e.*, simulations, for SpaGrOW and GROW was 37 and 62, respectively. However, in the latter case, seven simulations have to be subtracted for the comparison, because the partial atomic charged were optimized, as well. Hence, in the case of GROW, it was a five-dimensional optimization problem, and for the gradient calculation, one simulation more was required at each iteration. However, SpaGrOW was significantly faster than GROW.

Figure 18 depicts the development of the LJ Parameters for GROW and SpaGrOW (Figure 18a refers to $\sigma$ and Figure 18b to $\varepsilon$). $\sigma(\mathrm{C})$ remained constant, and all other force field parameters were increased. At the fourth iteration of SpaGrOW, the parameters were very similar to the ones of GROW at the seventh iteration. However, the following interesting observation could be made. The parameter, $\sigma(\mathrm{C})$, remained constant, even during the whole optimization procedure. In the case of GROW, it was first decreased in

order to obtain nearly the same value as before. As it is gradient-based, GROW tends to make detours. As it is grid-based, SpaGrOW is capable of keeping one or more parameters constant during the whole optimization procedure, because it can converge along a certain grid line or hyperplane. This is another reason for the faster convergency of SpaGrOW. Only in the case of $\varepsilon$, some small detours were observed. The algorithm ran through the triangle indicated in Figure 18b. SpaGrOW delivered a set of force field parameters, which differed a little from the ones obtained by GROW. Hence, it achieved a different domain close to the minimum.

**Figure 18.** Development of the Lennard-Jones (LJ) parameters in the case of benzene ($\Delta_v H$,$\rho_l$) for GROW and SpaGrOW—$\sigma$(H) and $\sigma$(C) (**a**)—as well as $\varepsilon$(H) and $\varepsilon$(C) (**b**). The unfilled circles indicate the optimal parameters. SpaGrOW led in a more direct way to the minimum than GROW. Only in the case of $\varepsilon$, some detours could be observed, due to the triangle.

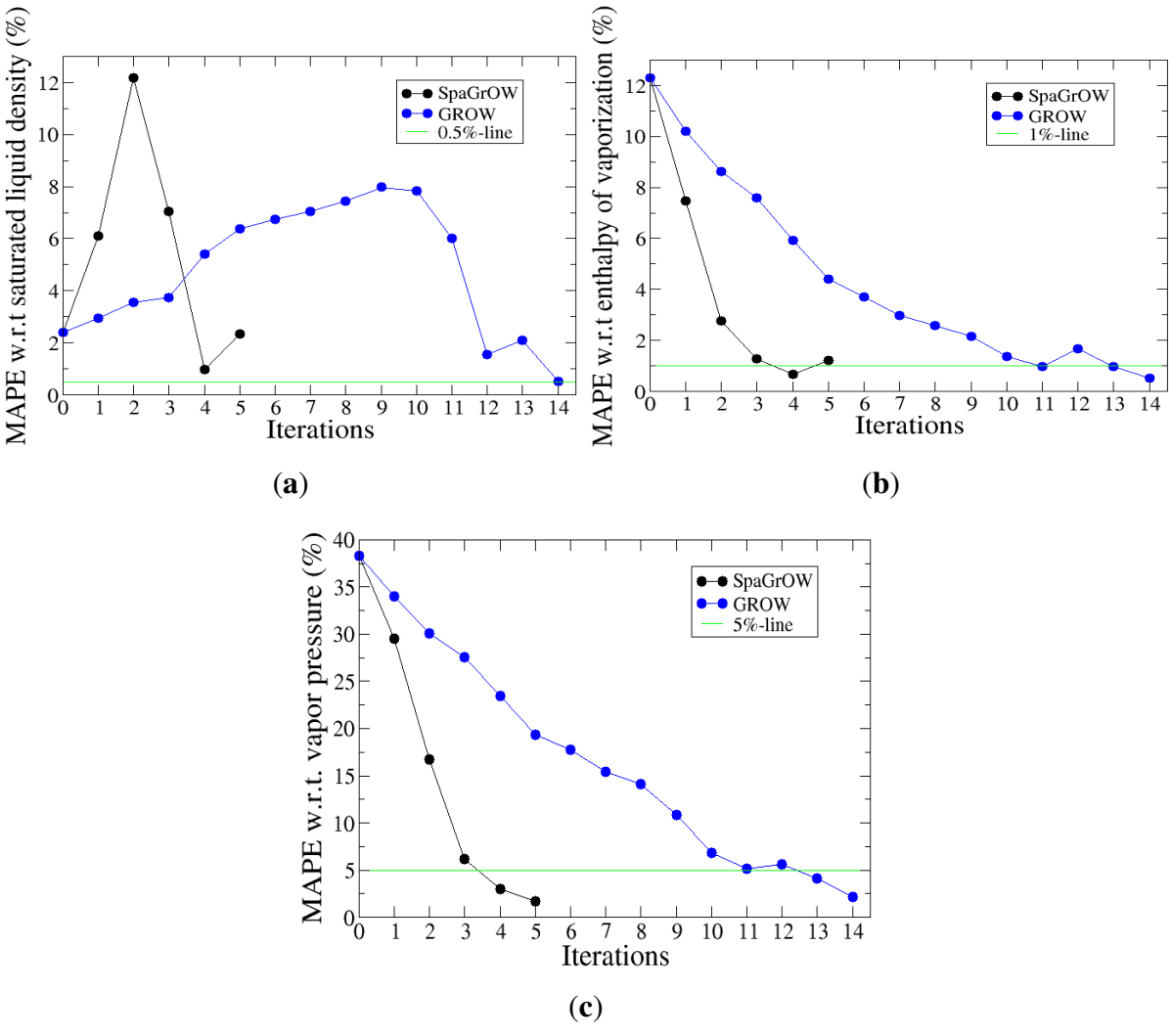



**Ethylene oxide** Ethylene oxide ($C_2H_4O$) is a highly toxic substance, but very relevant for industrial applications, because it is an educt for many industrially fabricated materials. It is very suitable for the evaluations of the optimization algorithms, because it has been characterized both experimentally [48,49] and by molecular simulations [14,50,51].

Figure 19 shows the MAPE values for ethylene oxide during the optimization procedures. The target observables were the saturated liquid density (Figure 19a), the enthalpy of vaporization (Figure 19b) and the vapor pressure (Figure 19c), considered at seven different temperatures. For pressures, the statistical uncertainty within molecular simulations is higher than for $\Delta_v H$ and $\rho_l$. Here, 5% were assumed. Experimental data was taken from [48]. The simulations performed were Grand-Equilibrium Monte-Carlo (GEMC) simulations based on a rigid united-atom model with a dipolar moment developed by [14]. They were executed with the software tool *ms2* [52]. As GEMC simulations simulate the liquid and gas phase successively, the VLE data could be calculated correctly. The liquid simulation was performed in the $NpT$ ensemble and the gas simulation in the $\mu$VT ensemble, where the chemical potential, $\mu$, was kept constant. On the computer cluster mentioned above, eight to ten hours were required for the simulation of 500 liquid and 500 gaseous ethylene oxide molecules. Thereby, the total number of 345,000 Monte-Carlo steps was distributed on eight processors. The force field parameters to

be optimized were $\varepsilon(\mathrm{CH_2}), \varepsilon(\mathrm{O}), \sigma(\mathrm{CH_2})$ and $\sigma(\mathrm{O})$. The initial parameters were the result of parameters obtained by a global pre-optimization based on random search [51]. Both parameters were changed by no more than 20%. The SpaGrOW variables were the same as for benzene. As the optimization problem was four-dimensional, Gaussian RBFs in combination with MARS were used again.

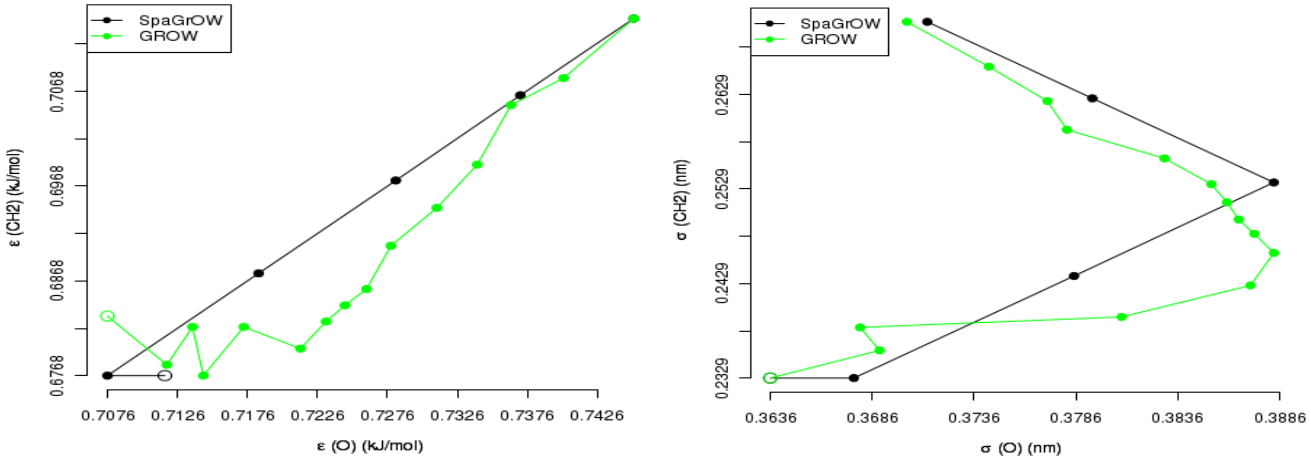
**Figure 19.** MAPE values with respect to $\rho_l$ (**a**), $\Delta_v H$ (**b**) and $p_\sigma$ (**c**) in the case of ethylene oxide during the Vapor-Liquid Equilibrium (VLE) optimization for SpaGrOW and GROW. The smoothing procedure was based on Gaussian RBFs combined with the MARS algorithm. The force field parameters were $\varepsilon(\mathrm{CH_2}), \varepsilon(\mathrm{O}), \sigma(\mathrm{CH_2})$ and $\sigma(\mathrm{O})$. The faster convergency of SpaGrOW compared to GROW could be confirmed again.




In total, GROW required 14 iterations for the optimization using the steepest descent method only. SpaGrOW only needed five to achieve a comparable loss function value in the same order of magnitude: for GROW, $F(x^{(14)}) = 2.4 \times 10^{-4}$ and, for SpaGrOW, $F(x^{(5)}) = 3.9 \times 10^{-4}$. Please note that the force field was not optimal and that other methods had to be applied in order to optimize it. For details, *cf.* [40]. SpaGrOW and GROW required 54 and 77 molecular simulations, respectively. Hence, SpaGrOW was significantly faster again.

Figure 20 shows the development of the LJ parameters during the optimization process in comparison to GROW (Figure 20a refers to $\varepsilon$ and Figure 20b to $\sigma$). All parameters were decreased, and SpaGrOW delivered approximately the same parameters as GROW. Undesired detours were avoided again, except for $\sigma$. However, the detours made by GROW could be linearized.

**Figure 20.** Development of the LJ parameters in the case of ethylene oxide (VLE) for GROW and SpaGrOW: $\varepsilon(CH_2)$ and $\varepsilon(O)$ (**a**), as well as $\sigma(CH_2)$ and $\sigma(O)$ (**b**). The unfilled circles show the final parameters. SpaGrOW led to a more direct way to the minimum again.




(**a**) (**b**)

To summarize, SpaGrOW exhibits a significantly higher speed of convergency than GROW.

#### 4.2.2. Comparison to Gradient-Based Methods Close to the Minimum: Dipropylene Glycol Dimethyl Ether

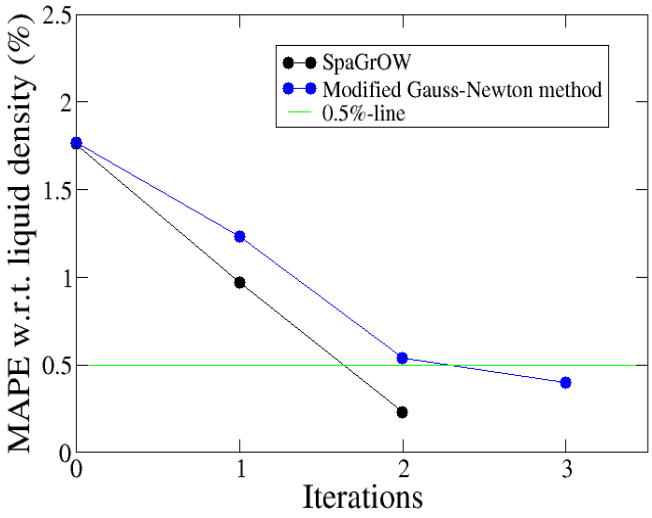
In the framework of the *Industrial Fluid Property Simulation Challenge (IFPSC) 2010* [53], a liquid-liquid equilibrium (LLE) between two liquid phases should be calculated with molecular models. One of the two component was water and the other, dipropylene glycol dimethyl ether ($C_8H_{18}O_3$), an inert, water-resistant, nontoxic and industrially highly relevant solvent.

The target observable was the liquid density, $\rho$, only, considered at four different temperatures and taken from [54]. The simulations performed were molecular dynamics simulations in the $NpT$ ensemble executed with the software tool, *GROMACS* [46]. On the computer cluster mentioned above, three to four hours were required for the simulation of 512 ether molecules for 0.5 ns using a time step of 2 fs.

The LJ sites were located at the $CH_3$, the $CH_2$, the CH group and the oxygen. Hence, it was an eight-dimensional optimization problem. The initial force field parameters from [55]. As GROW could not achieve an optimal force field reproducing the liquid density of the ether [55], another gradient-based method was applied, based on a Taylor series up to the first member for the target observables, delivering a quadratic model for the loss function. It is a modified Gauss-Newton method combined with the Trust Region approach, *i.e.*, the quadratic model is minimized within a compact domain, as well. The algorithm has also been developed by the authors, *cf.* [40] for more details. After three iterations, the modified Gauss-Newton method could achieve optimal liquid densities.

Figure 21 indicates that SpaGrOW (with $\Delta_0 = 0.3 \times \Delta^{\max}$) required two iterations only to do so. As the optimization problem was eight-dimensional, the Lipra method was applied as a smoothing procedure. However, SpaGrOW needed 38 simulations, three-times more than the modified Gauss–Newton method, which only had to execute twelve simulations. With an optimal $\Delta_0$, the number of simulations could have been reduced to 19 in the case of SpaGrOW. The modified Gauss-Newton method is more reliable than SpaGrOW close to the minimum, but it could be shown that SpaGrOW is capable of getting closer to the minimum than the standard gradient-based algorithms used in GROW.

**Figure 21.** MAPE values with respect to $\rho$ in the case of dipropylene glycol dimethyl ether during the optimization process of SpaGrOW in comparison to the modified Gauss-Newton method. The smoothing procedure was realized by the Lipra method. The optimization problem was eight-dimensional. SpaGrOW needed two iterations only, but significantly more simulations than the modified Gauss-Newton method in order to achieve optimal liquid densities.

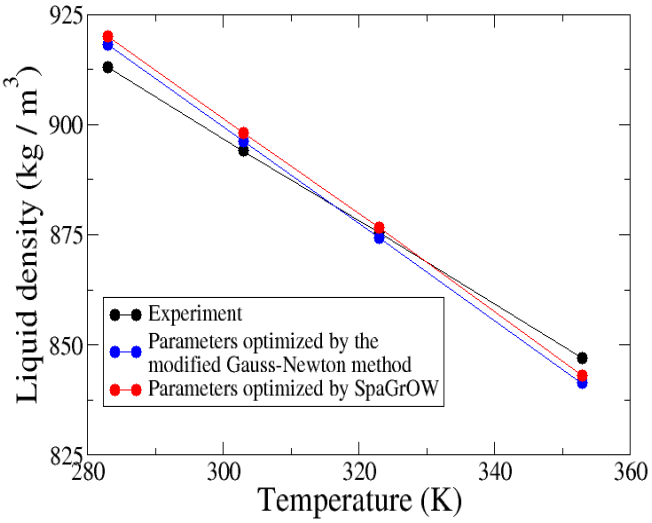



However, the Gauss-Newton method is only applicable close to the minimum, *i.e.*, when the loss function is nearly quadratic. In contrast, SpaGrOW is more generally applicable, and another drawback of the Gauss-Newton method is that it requires a gradient, which is not guaranteed to be computable, due to the reasons mentioned in Section 1. Please note that the density is not as noisy as other target properties, like the pressure or diffusion coefficient. Of course, the existence of an optimal $\Delta$ for SpaGrOW is not guaranteed either in practice.

Figure 22 shows the optimal liquid densities as a function of temperature for the modified Gauss-Newton method and SpaGrOW. Both methods could achieve an optimal force field with respect to $\rho$ in contrast to GROW. However, SpaGrOW could reproduce the trend of the curve better than the modified Gauss-Newton method.

To summarize, SpaGrOW is capable of getting closer to the minimum than GROW, when the smoothing procedure and $\Delta_0$ are chosen properly. However, the modified Gauss-Newton method needs significantly less simulations.

**Figure 22.** Optimization of the density $\rho$ in the case of dipropylene glycol dimethyl ether using the modified Gauss-Newton method and SpaGrOW. Optimal densities could be achieved by both methods, but SpaGrOW needed many more simulations. However, SpaGrOW could reproduce the trend of the curve better.




## 5. Conclusions

In this paper, the new derivative-free optimization method was presented in detail and applied to the parameterization of force fields in the field of molecular simulations. It is a combination of appropriate smoothing and regularization procedures, interpolation on sparse grids and the Trust Region approach. SpaGrOW turned out to be a highly efficient algorithm outperforming standard gradient-based methods with respect to the speed of convergency. Furthermore, it is capable of getting closer to the minimum. However, if a gradient can be calculated correctly, the gradient-based methods exhibit a slightly higher robustness than SpaGrOW. The new method is validated in the following with respect to the three criteria speed of convergency, local refinements and robustness:

- **Speed of convergency:** Whenever $\Delta_0$ is chosen properly, SpaGrOW often requires only half of the number of simulations than gradient-based methods. The speed of convergency was also higher, *i.e.*, the number of iterations was significantly lower.
  The choice, $\Delta_0 = 0.3 \times \Delta^{\max}$, was suitable in most cases, *i.e.*, the parameterization of SpaGrOW is not critical at the beginning of the optimization.
- **Local refinements:** SpaGrOW is capable of getting closer to the minimum than GROW. However, the choice of $\Delta$ becomes critical, which reduces the robustness of SpaGrOW: If $\Delta$ is too high, the smoothing and interpolation algorithms cannot deliver a reliable model for the loss functions. If it is too small, only noise can be reproduced. A modified Gauss-Newton method turned out to be more efficient and close to the minimum, if the associated gradients can be computed correctly. An important advantage of SpaGrOW is the fact that the step length can be modified by the Trust Region steps, leading to different descent directions, which is not possible for gradient-based methods first determining the descent direction and, then, searching for a reliable step length.

- **Robustness:** SpaGrOW exhibits a slightly lower robustness than the gradient-based methods. Due to an inappropriate choice of $\Delta$, the minimum of the model can be transferred under certain conditions, and the course of the algorithm can be modified. The variable must be chosen, so that a decreasing trend of the loss function is present within the actual trust region. However, this is not trivial here, because the shape of the loss function is not known *a priori*.

It is extremely difficult to find a method that is efficient and robust at the same time. These two properties often face each other: Stochastic global optimization methods, like simulated annealing or evolutionary algorithms, are very robust with respect to statistical noise, but require a high amount of computation time to determine the global minimum exactly. Gradient-based ones are fast-convergent, but are less robust and reliant on the differentiability of the function to be minimized. SpaGrOW is situated in between: The higher speed of convergency has to be compensated by a lower robustness. However, the robustness was not reduced significantly, and the assumption that the loss function is smooth does not have to be made. Furthermore, SpaGrOW may still be successful when gradient-based methods exist.
To summarize, SpaGrOW is a generic, efficient and, also, quite robust algorithm, which can be used for many optimization problems. For force field parameterization tasks, it is highly recommended and preferred to gradient-based algorithms.

## Acknowledgements

We are grateful to Janina Hemmersbach for the detailed analysis and selection of appropriate smoothing procedures, as well as to Anton Schüller for fruitful discussions and advice.

## Conflicts of Interest

The authors declare no conflict of interest.

## References

1. Allen, M.P.; Tildesley, D.J. *Computer Simulation of Liquids*; Clarendon Press: Oxford, UK, 1987.
2. Frenkel, D.; Smit, B. *Understanding Molecular Simulation: From Algorithms to Applications*; Academic Press: Waltham, MA, USA, 2006.
3. Schrödinger, E. Quantisierung als eigenwertproblem. *Ann. Phys.* **1926**, *79*, 361–367.
4. Jensen, F. *Introduction to Computational Chemistry*; Wiley: Hoboken, NJ, USA, 1999.
5. Guevara-Carrion, G.; Hasse, H.; Vrabec, J. *Multiscale Molecular Methods in Applied Chemistry*; Kirschner, B., Vrabec, J., Eds.; Series: Topics in Current Chemistry, Volume 307; Springer: Berlin/Heidelberg, Germany, 2012; pp 201–249.
6. Weiner, S.J.; Kollman, P.A.; Case, D.A.; Singh, U.C.; Ghio, C.; Alagona, G.; Profeta, S.; Weiner, P. A new force field for molecular mechanical simulation of nucleic acids and proteins. *J. Am. Chem. Soc.* **1984**, *106*, 765–784.
7. Berendsen, H.J.; van Gunsteren, W.F. *GROMOS87 Manual*, Library Manual; Biomos AG: Groningen, The Netherlands, 1987.

8. Jorgensen, W.L.; Maxwell, D.S.; Tirado-Rives, J. Development and testing of the OPLS all-atom force field on conformational energetics and properties of organic liquids. *J. Am. Chem. Soc.* **1996**, *118*, 11225–11236.
9. Duan, Y.; Wu, C.; Chowdhury, S.; Lee, M.C.; Xiong, G.; Zhang, W.; Yang, R.; Cieplak, P.; Luo, R.; Lee, T.; *et al*. A point-charge force field for molecular mechanics simulations of proteins based on condensed-phase quantum mechancical calculations. *J. Comput. Chem.* **2003**, *24*, 1999–2012.
10. Reith, D.; Kirschner, K.N. A modern workflow for force-field development—bridging quantum mechanics and atomistic computational models. *Comput. Phys. Commun.* **2011**, *182*, 2184–2191.
11. Kräemer-Fuhrmann, O.; Neisius, J.; Gehlen, N.; Reith, D.; Kirschner K.N. $Wolf_2Pack$—Portal based atomistic force-field development. *J. Chem. Inf. Mod.* **2013**, *53*, 802–808.
12. Jorgensen, W.L.; Madura, J.D.; Swensen, C.J. Optimized intermolecular potential functions for liquid hydrocarbons. *J. Am. Chem. Soc.* **1984**, *106*, 6638–6646.
13. Martin, M.G.; Siepmann, J.I. Transferable potentials for phase equilibria. 1. United-atom description of $n$-alkanes. *J. Phys. Chem. B* **1998**, *102*, 2567–2577.
14. Eckl, B.; Vrabec, J.; Hasse, H. On the application of force fields for predicting a wide variety of properties: Ethylene oxide as an example. *Fluid Phase Equilib.* **2008**, *274*, 16–26.
15. Peguin, R.P.S.; Kamath, G.; Potoff, J.J.; da Rocha, S.R.P. All-atom force field for the prediction of vapor–liquid equilibria and interfacial properties of HFA134a. *J. Phys. Chem. B* **2009**, *113*, 178–187.
16. Faller, R.; Schmitz, H.; Biermann, O.; Müller-Plathe, F. Automatic parameterization of force fields liquids by simplex optimization. *J. Comput. Chem.* **1999**, *20*, 1009–1017.
17. Ungerer, P.; Beauvais, C.; Delhommelle, J.; Boutin, A.; Rousseau, B.; Fuchs, A.H. Optimization of the anisotropic united atoms intermolecular potential for $n$-alkanes, *J. Comput. Phys.* **1999**, *112*, 5499–5510.
18. Bourasseau, E.; Haboudou, M.; Boutin, A.; Fuchs, A.H.; Ungerer, P. New optimization method for intermolecular potentials: Optimization of a new anisotropic united atoms potential for Olefins: Prediction of equilibrium properties. *J. Chem. Phys.* **2003**, *118*, 3020–3034.
19. Stoll, J.; Vrabec, J.; Hasse, H. A set of molecular models for carbon monoxide and halogenated hydrocarbons. *J. Chem. Phys.* **2003**, *119*, 11396–11407.
20. Sun, H. Prediction of fluid densities using automatically derived VDW parameters. *Fluid Phase Equilib.* **2004**, *217*, 59–76.
21. Hülsmann, M.; Köddermann, T.; Vrabec, J.; Reith, D. GROW: A gradient-based optimization workflow for the automated development of molecular models. *Comput. Phys. Commun.* **2010**, *181*, 499–513.
22. Hülsmann, M.; Vrabec, J.; Maaß, A.; Reith, D. Assessment of numerical optimization algorithms for the development of molecular models. *Comput. Phys. Commun.* **2010**, *181*, 887–905.
23. Hülsmann, M.; Müller, T.J.; Köddermann, T.; Reith, D. Automated force field optimisation of small molecules using a gradient-based workflow package. *Mol. Simul.* **2011**, *36*, 1182–1196.
24. Griebel, M.; Schneider, M.; Zenger, C. *A Combination Technique for the Solution of Sparse Grid Problems*; Technical Report; Institut für Informatik, Technische Universität München: Munich, Germany, 1990.

25. Moré, J.J. *Mathematical Programming: The State of the Art*; Bachem, A., Grötschel, M., Korte, B., Eds.; Springer: Berlin/Heidelberg, Germany, 1983; pp 258–287.
26. Nocedal, J.; Wright, S.J. *Numerical Optimization*; Springer: Berlin/Heidelberg, Germany, 1999.
27. Mangasarian, O.L.; Rosen, J.B.; Thompson, M.E. Global minimization via piecewise-linear underestimation. *J. Glob. Optim.* **2004**, *32*, 1–9.
28. Smolyak, S.A. Quadrature and interpolation formulas for tensor products of certain classes of functions. *Sov. Math. Doklady* **1963**, *4*, 240–243.
29. Zenger, C. *Parallel Algorithms for Partial Differential Equations, Notes on Numerical Fluid Mechanics*; Hackbusch, W., Ed.; Vieweg: Wiesbaden, Germany, 1991; Volume 31, pp. 241–251.
30. Bungartz, H.-J. Iterative Methods in Linear Algebra. In Proceedings of the IMACS International Symposium, Brüssel, Belgium, April 1991; de Groen, P., Beauwens, R., Eds.; North-Holland Publishing Co.: Amsterdam, The Netherlands, 1992; pp. 293–310.
31. Powell, M. *Algorithms for Approximation*; Mason, J.C., Cox, M.G., Eds.; Clarendon Press: Oxford, UK, 1987; pp. 143–167.
32. Bishop, C.M. *Pattern Recognition and Machine Learning* ; Series: Information Science and Statistics; Springer: Berlin/Heidelberg, Germany, 2007.
33. MacQueen, J.B. Some Methods for Classification and Analysis of Multivariate Observations. In Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, CA, USA, June/July 1965; University of California Press: Berkeley, CA, USA, 1967; pp. 281–297.
34. Zou, H.; Hastie, T. Regularization and variable selection via the elastic net. *J. Roy. Stat. Soc. Ser. B* **2005**, *67*, 301–320.
35. Hoerl, A.E.; Kennard, R.W. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics* **1970**, *12*, 55–67.
36. Tibshirani, R. Regression shrinkage and selection via the lasso. *J. Roy. Stat. Soc. Ser. B* **1996**, *58*, 267–288.
37. Friedman, J.H. Multivariate adaptive regression splines. *Ann. Stat.* **1991**, *19*, 1–67.
38. Wendland, H. *Scattered Data Approximation*; Cambridge University Press: Cambridge, UK, 2005.
39. Wendland, H. Konstruktion und Untersuchung radialer Basisfunktionen mit kompaktem Träger. Ph.D. Thesis, Universität Göttingen, Göttingen, Germany, 1996.
40. Hülsmann, M. Effiziente und neuartige Verfahren zur Optimierung von Kraftfeldparametern bei atomistischen Molekularen Simulationen kondensierter Materie. Ph.D. Thesis, Universität zu Köln, Cologne, Germany, 2012.
41. Conn, A.R.; Scheinberg, K.; Toint, P.L. *On the Convergence of Derivative-free Methods for Unconstrained Optimization*; Iserles, A., Buhmann, M., Eds.; Cambridge University Press: Cambridge, UK, 1997.
42. The R Project for Statistical Computing. Available online: http://www.r-project.org/ (accessed on 14 February 2013).
43. Stoll, J.; Vrabec, J.; Hasse, H.; Fischer, J. Comprehensive study of the vapour-liquid equilibria of the pure two-centre Lennard-Jones plus pointquadrupole fluid. *Fluid Phase Equilib.* **2001**, *179*, 339–362.

44. Yoshida, K.; Matubayasi, N.; Nakahara, M. Self-diffusion coefficients for water and organic solvents at high temperatures along the coexistence curve. *J. Chem. Phys.* **2008**, *129*, 214501–214509.
45. NIST Chemistry Webbook. Available online: http://webbook.nist.gov/chemistry/ (accessed on 14 February 2013).
46. GROMACS Molecular Simulation Tool. Available online: http://www.gromacs.org/ (accessed on 14 February 2013).
47. Moscito Molecular Simulation Tool. Available online: http://ganter.chemie.uni-dortmund.de/MOSCITO/ (accessed on 14 February 2013).
48. Buckles, C.; Chipman, P.; Cubillas, M.; Lakin, M.; Slezak, D.; Townsend, D.; Vogel, K.; Wagner, M. *Ethylene Oxide User's Guide*; Online Manual, Publisher: American Chemistry Council, 1999; Available online: http://www.ethyleneoxide.com (accessed on 14 February 2013).
49. Olson, J.D.; Wilson, L.C. Benchmarks for the fourth industrial fluid properties simulation Challenge. *Fluid Phase Equilib.* **2008**, *274*, 10–15.
50. Müller, T.J.; Roy, S.; Zhao, W.; Maaß, A.; Reith, D. Economic simplex optimization for broad range property prediction: Strengths and weaknesses of an automated approach for tailoring of parameters. *Fluid Phase Equilib.* **2008**, *274*, 27–35.
51. Maaß, A.; Nikitina, L.; Clees, T.; Kirschner, K.N.; Reith, D. Multiobjective optimisation on the basis of random models for ethylene oxide. *Mol. Simul.* **2010**, *36*, 1208–1218.
52. ms2 Molecular Simulation Tool. Available online: http://www.ms-2.de/ (accessed on 14 February 2013).
53. The Industrial Fluid Property Simulation Challenge, 2010. Available online: http://www.ifpsc.com (accessed on 14 February 2013).
54. Esteve, X.; Conesa, A.; Coronas, A. Liquid densities, kinematic viscosities, and heat capacities of some alkylene glycol dialkyl ethers. *J. Chem. Eng. Data* **2003**, *48*, 392–397.
55. Köddermann, T.; Kirschner, K.N.; Vrabec, J.; Hülsmann, M.; Reith, D. Liquid-liquid equilibria of dipropylene glycol dimethyl ether and water by molecular dynamics. *Fluid Phase Equilib.* **2011**, *310*, 25–31.