Technische Universität Berlin

Institut für Mathematik und Naturwissenschaften
Fachgebiet Modellierung, Simulation und Optimierung in Natur- und Ingenieurwissenschaften

Fakultät II
Straße des 17. Juni 136
10623 Berlin
http://www.math.tu-berlin.de

Master Thesis

On a Novel Approach for Global Optimization of Non-Convex Problems

Wilhelm Bender

Matriculation Number: 325935
13.05.2019

Supervised by
Priv.-Doz. Dr. Konstantin Fackeldey

Assistant Supervisor
Dr. Marcus Weber

Hereby I declare that I wrote this thesis myself with the help of no more than the mentioned literature and auxiliary means.

Berlin, 13.05.2019

.....

(Signature Wilhelm Bender)

Contents

List of Figures

List of Tables

Abstract

In dieser Arbeit werden neue Ansätze für die globale Optimierung entwickelt. Es wird von theoretischen Zusammenhängen mit der Kompläxitätstheorie auf praktische Ergebnisse aus dem Protein-Ligand-Docking überführt. Angesichts der Effizienz des vorgeschlagenen Optimierers werden Kontinuitätsmethoden motiviert, die einen verbessertes Optimierungsverfahren liefern.

]Bibliography

[author] Wilhelm Bender, Master's Thesis, Technische Univesität Berlin, Konstantin Fackeldey, Marcus Weber, 2020

Abstract

In this work new approaches for global optimization are developed. We move from theoretical connections in complexity theory to very practical results in protein-ligand-docking. In regard to the efficiency of the suggested optimizer, continuity methods are motivated, that yield an improved optimization method.

]Bibliography

[author] Wilhelm Bender, Master's Thesis, Technische Univesität Berlin, Konstantin Fackeldey, Marcus Weber, 2020

1 Introduction

Content wise the thesis consist of two parts that will be proceeded by two introductory chapters on some non-linear optimization basics and a consideration on two problems from complexity theory. In the first half, we will exhibit the **projected newton method**, an algorithm that is new in the field of numerical optimization. Even though its theory is very close to theorems, that the reader can find concerned with the convergence analysis of the newton method itself, we provide an incomparable approach to how one can find a global minimum using the method. The new approach is benchmarked for numerical test problems in comparison of global optimization algorithms from the MATLAB Global Optimization package. The performance of the algorithm applied to the protein-ligand docking problem will be tested and evaluated in comparison to current docking software packages. The algorithm has to be regarded as a heuristic method.
The second half is an advance on the problems, that emerged with the projected newton method based optimizer in the first half. Even though the principle of finding roots of a multivariate function with a numerical method is still applied, we exhibit another brand new root finder called the **dynamic embedding algorithm**. With this iterative method, we replace the projected newton method and suddenly are able to prove global convergence for a subroutine of the optimizer. In the course of this thesis, we will see that the global convergence property is very important to finally hope for an effective algorithm to solve for the global minimum of a multivariate, non-convex, coercive and continuous optimization problem. The algorithm has to be regarded as a deterministic method.
Finally we conclude on an answer to the thesis' question and describe further possibilities of research in the outlook.
The thesis aims to cause for thought of looking at the novel methods from the perspective of complexity theory.

1.1 In Regard of Complexity Theory

Global optimization of non-linear problems is a topic, which has a lot of open questions. Generally there is no solver that allows for efficient calculations of global optima but either has to be approximated with heuristic methods or computed with deterministic methods in impracticable amounts of time. All of these problems intrude the researcher to think about the complexity of the problems, they are dealing with. To raise the topic of complexity theory in theoretical computer science, means to talk about deterministic polynomial time languages (class $\mathcal{P}$) and non-deterministic polynomial time languages (class $\mathcal{NP}$) that get accepted by Turing machines. Cook [1] showed that if and only

if there is an algorithm solving the satisfiability problem in polynomial time, then the classes $\mathcal{P}$ and $\mathcal{NP}$ are equivalent. Among many problems listed in Karp [8] it is known, that problems like the travelling salesman problem do have polynomial deterministic time algorithms if and only if the classes $\mathcal{P}$ and $\mathcal{NP}$ are the same. Among all the research, that has been done so far, we are right to assume the classes are not to be the same, however we are missing a formal proof. In this thesis we apply a novel heuristic projected newton solver, that works similarly to a stochastic gradient descent, as first introduced by Robbins and Monro [9]. Similarly we will apply the idea of randomly selecting starting values instead of executing the minimization algorithm only once. With the new approach another problem emerges, that is very closely related to the decidability, namely the decidability we are defining in complexity theory. One interesting example in this relation is the satisfiability problem. It is contained in $\mathcal{NP}$ but still can be formulated as a diophantine problem on integers, that is even harder to solve and proven to be unsolvable or to stay with the terms of complexity theory, is undecidable. Davis [10] reviews the findings of Matiyasevich [13] that deal with Hilbert's tenth problem, concerned with the very matter of decidability of diophantine equations on integers. The bottom line is that Matiyasevich's theorem proves the impossibility of a solution to Hilbert's tenth problem. In deeper context diophantine equations are semi-decidable, which means that a Turing Machine can decide if a solution is found but can not decide if there is no solution. After assuming the projected newton method is not an optimal approach to overcome the undecidability of the problem, we open a new approach to solving a problem in effective time. The *dynamic embedding algorithm* provided in the second half is in need of mathematical investigations concerning runtime and convergence properties. Applied to various classes of problems, it remains unclear how the method influences the decidability of a diophantine equation on rational numbers. We provide the reader with constructions, methods and empirical data to motivate the research of the open questions. We can embrace these open questions with excitement, once we see the behaviour of the dynamic embedding algorithm in application of difficult test problems.

1.2 In Disregard of Critical Points

We construct an alternative approach to solving multivariate optimization problems. The most known theory of non-linear optimization is concerned with the solution of a problem where $\Omega \subset \mathbb{R}^n$, $F : \Omega \to \mathbb{R}^n$, F convex on Ω. This is due to the fact, that for multivariate problems, we solve for the roots of the gradient of the objective function f. In many cases it is a map $f : \Omega \to \mathbb{R}$. We can denote

$$\nabla f = F$$

A reader who is used to the common approach of choosing efficient search directions, will have to look at optimization from a different perspective. Even though we are optimizing the parameters for f, we are not concerned with the roots of the function

F. In other words we do not look for the extrema of the function f. Instead we are interested in roots of the function f itself and not in the ones of the gradient ∇f. In the course of the thesis we construct methods to solve for the roots of f, in order to reach the global optimum of a function by a so called Level-Set-Bisection algorithm. As the reader will see, this is a very effective approach to overcome the exponential barrier. The investigation on the computational complexity of the applied algorithms is not being followed to every aspect within this thesis. We make a runtime analysis for local cases, however provide the tools to apply the result to an advanced approach.
Most of this thesis is concerned with the mathematics behind finding a $y \in \Omega$ such that

$$f(y) = 0 \tag{1.2.1}$$

What makes this problem hard to solve is not its dimensionality only. The biggest and most interesting problems emerge as soon as f becomes non-convex on Ω. Mathematical modelling of interesting use cases, as for example in pharmaceutical research and also operational research, to just name a few, very often are non-convex when formulated as a non-linear program (NLP). Some formulations also allow for convex objective functions, however often result in a number of inequality constraints that cause the issue of choosing the right constraints to be active or not. To put the first relation to complexity theory, we will also empirically examine a few runtime properties of the suggested methods. Beginning with the projected newton method, we will only examine local runtime complexity mathematically but will also give a hint on how the solver will perform on non-convex problems with rising number of variables in chosen examples of test problems that can be found in [4].

1.3 Global Optimization

Lipschitz-Optimization is the basic concept of solving an optimization problem deterministically. Since variations of this method as described in Gablonsky [5], are open for paralellization, this field has been experiencing further research. It is known, that these approaches reach exponential runtime complexity, when computing complex problems. They are the only known way of solving problems without taking cutbacks in the precision of the calculation. In order to formally state the problem of interest, we define the global optimization problem.

Definition 1.3.1. Let $\Omega \subset \mathbb{R}^N$ the domain and $f : \Omega \to \mathbb{R}$, then $x_{opt} \in \Omega$ is called *global minimum of* f if

$$f(x_{opt}) \leq f(x) \quad \forall x \in \Omega \tag{1.3.1}$$

1.4 Global Optimality Criteria

In general we know a lot about local optimality criteria. However in this thesis we are putting our focus on methods to full fill global optimality criteria. Since numerical approaches often suffer of imprecision, we state the weaker formulation

Definition 1.4.1. Let $\Omega \subset \mathbb{R}^N$ the domain and $f : \Omega \to \mathbb{R}$ be lipschitz-continuous. Find $x_{opt} \in \Omega$ such that

$$f_{opt} = f(x_{opt}) \leq f^* + \epsilon, \tag{1.4.1}$$

where $\epsilon > 0$ is constant.

These criteria are not much to begin with but we will use them to think about the following idea. Lets take an one-dimensional example

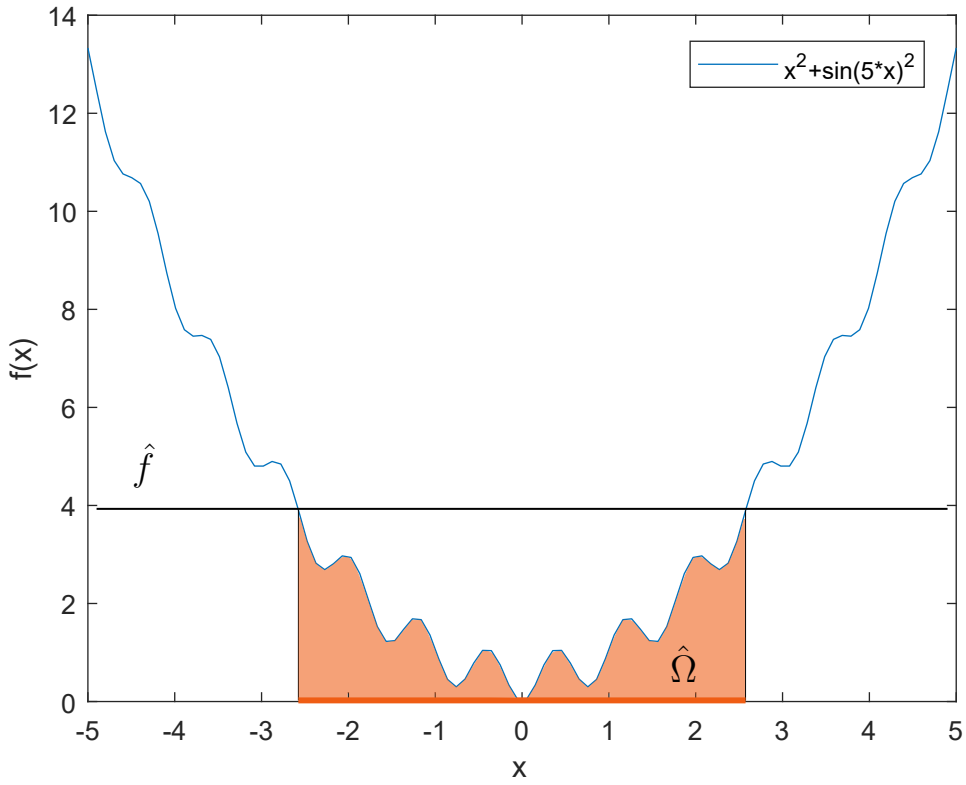


Figure 1.1: Example for suboptimal level and suboptimal points of non-convex problem in one dimension.

If we just pick a random $x \in \Omega$, we would be able to pick a subset of the domain, let us denote it $\hat{\Omega} \subset \Omega$, who's points have bigger value in the function than the complementary set. We have a set of suboptimal points.

Definition 1.4.2. Let $a, b \in \mathbb{R}^n$, $\Omega = \{x \in \mathbb{R}^n : a_i \leq x_i \leq b_i\}$, and $f : \Omega \to \mathbb{R}$. All points $\hat{x} \in \hat{\Omega} \subset \Omega$ are called **suboptimal points** if

$$f(\hat{x}) \leq \hat{f} \leq f(y), \quad \forall y \in \Omega \setminus \hat{\Omega} \tag{1.4.2}$$

$\hat{f}$ is called the **suboptimal level**.

As one can imagine, it is very easy to pick such a suboptimal level for any given continuous function by simply picking a random point from the domain. Given the function is coercive, i.e. $||x|| \to \infty : f(x) \to \infty$, we can pick a suboptimal level, where x^* is the global minimum of f, such that

$$\hat{f} \leq f(x^*) \tag{1.4.3}$$

In which case the set of suboptimal points is empty. Let us assume that we can always make a statement about the emptiness of the suboptimal set. We define the following **compact** series, that takes advantage of existence of such a statement

$$\hat{f}_n = \begin{cases} \hat{f}_{n-1} - \frac{|\hat{f}_{n-1} - \hat{f}_{n-2}|}{2}, & \hat{\Omega}_{n-1} \neq \emptyset \\ \hat{f}_{n-1} + \frac{|\hat{f}_{n-1} - \hat{f}_{n-2}|}{2}, & \hat{\Omega}_{n-1} = \emptyset \end{cases} \tag{1.4.4}$$

Notice that if $\hat{\Omega} = \emptyset$, the according suboptimal level holds $\hat{f} < f^*$. Vice versa if $\hat{\Omega} \neq \emptyset$, the according suboptimal level holds $\hat{f} > f^*$.

Theorem 1.4.3. *Let series $\hat{f}_n$ be defined as in* (1.4.4) *on the corresponding coercive function $f : \Omega \to \mathbb{R}$, $\Omega \in \mathbb{R}^N$, with global minimum at x^*. The series $\hat{f}_n$ holds*

$$\lim_{n \to \infty} \hat{f}_n = f(x^*) \tag{1.4.5}$$

Proof. We construct two series, that represent the upper and lower bound of the series $\hat{f}_n$. These look like this

$$u_n = \begin{cases} \hat{f}_n, & \hat{f}_n > f^* \\ u_{n-1}, & \hat{f}_n < f^* \end{cases} \tag{1.4.6}$$

$$l_n = \begin{cases} l_{n-1}, & \hat{f}_n > f^* \\ \hat{f}_n, & \hat{f}_n < f^* \end{cases} \tag{1.4.7}$$

where the initial values are:

$$u_0 = f(x), \text{ for some } x \in \Omega \qquad c \in \mathbb{R} \text{ s.t.: } l_0 = c < \inf f(x)$$

Because f is coercive, we can assume without loss of generality, that

$$f(x) > 0, \quad \forall x \in \Omega$$

With the definition of the global minimum it holds

$$f(x^*) \leq f(x), \quad \text{for all } x \in \Omega$$

So we can take $u_0 = f(z)$ for some $z \in \Omega$ and since $f(x) > 0$ we can assume $l_0 = 0$. We now have a bound on the global minimum.

$$0 = l_0 < f^* \leq f(x) = u_0$$

Also note that u_n is monotonically decreasing and l_n is monotonically increasing. Because the series $\hat{f}_n$ is compact, we can apply the theorem of Bolzano-Weierstraß and prove convergence of the series $\hat{f}_n$, i.e. $\lim_{n\to\infty} \hat{f}_n = \hat{f}^*$. Since the series u_n and l_n are defined as in (1.4.6) and (1.4.7), we can assume that

$$\lim_{n\to\infty} u_n = \lim_{n\to\infty} l_n = \hat{f}^*$$

All together we get

$$\begin{aligned} 0 < f^* \leq f(x) = u_0 \\ l_0 < l_1 \leq \ldots \leq l_n \leq f^* \leq u_n \leq \ldots \leq u_0 \\ N \to \infty : \hat{f}^* \leq f^* \leq \hat{f}^* \end{aligned}$$

Now we know that l_n, u_n and $\hat{f}_n$ have the same limit and that this limit is equal to the global minimal function value f^*, the assertion

$$\lim_{n\to\infty} \hat{f}_n = f^* = f(x^*)$$

holds □

If a series of suboptimal levels as in definition 1.4.2 exists and is unique, i.e. the series

$$\hat{f}_n = \begin{cases} \hat{f}_{n-1} - \frac{|\hat{f}_{n-1} - \hat{f}_{n-2}|}{2}, & \hat{\Omega}_{n-1} \neq \emptyset \\ \hat{f}_{n-1} + \frac{|\hat{f}_{n-1} - \hat{f}_{n-2}|}{2}, & \hat{\Omega}_{n-1} = \emptyset \end{cases}$$

is well-defined, then the limit value of the series is the global minimum of f, i.e.

$$\lim_{n\to\infty} \hat{f}_n = f^*$$

1.5 Fundamental Question of the Thesis

If a series like $\hat{f}_n$ could be easily established, it would not be hard to compute a global minimum for all coercive functions. We will take a look at coercive continuous non-convex functions. The biggest question, that no widely accepted theory has an answer to, is if the set of suboptimal points Ω is empty or not. We will attempt to bind the considered theories of this thesis to complexity theory. As we mentioned earlier, diophantine equations on integers are undecidable. The reader might wonder how that stands in

relation to an algorithm, that can find the global minimum of a coercive function, which then even might be non-convex. We strongly suspect, that the term of decidability in complexity theory relates very much to the idea of finding a global minimum of a coercive non-convex function. We will show that that we can transform the satisfiability problem to a diophantine equation on rational numbers. The problem then has also the form of a non-linear program or more specifically is a multivariate polynomial of degree n of the form

$$p(x_1, \ldots, x_m) = \sum_{i_1+\cdots+i_m \leq n} a_{i_1 \ldots i_m} x_1^{i_1} x_2^{i_2} \ldots x_m^{i_m} = 0 \tag{1.5.1}$$

The problems *structure* is therefore similar. What is so surprising, is that if the variables $x_1, \ldots, x_n$ are integers, the problem becomes **diophantine** and therefore undecidable. Based on the work of Nikolai Vorobe'ev on Fibonacci numbers, Matiyasevich [13] publishes the celebrated proof, that the set $P = \{(a, b) | a > 0, b = F_{2a}\}$, where F_n is the n-th Fibonacci number, is diophantine and exhibits exponential growth. This proves the hypothesis by Julia Robinson that every recursively enumerable set is exponential diophantine and combined with Matiyasevich's shows that diophantine sets are recursively enumerable. Reviewing this line of work, including many other celebrated mathematicians at the time, M.Davis publishes [10], stating that Hilbert's tenth problem on integers is unsolvable. Hilbert's tenth problem on rational numbers however remains unsolved to this date. When arithmetizing the satisfiability problem (see section 2), we get to the special case, where $x_1, \ldots, x_n \in [0, 1]$. Obviously equation 1.5.1 is diophantine and therefore undecidable. So what is the point in transforming the satisfiability problem into something, that is even harder to solve? We even know from Lagarias [11] simplified diophantine equations, namely Binary Quadratic Diophantine Equations (BQDE), are in the class of EXP (the class of languages that get accepted by Turing Machines in deterministic exponential time). Well, coming from the perspective of non-linear optimization we have a chance. Also here we have no possibility of evaluating the equations unsolvability but we are able to transform the problem further into something useful with something called a *homotopy*. As this thesis is intended to introduce new approaches of how to solve global optimization problems, we only provide the *dynamic embedding algorithm* in section 4 as a tool to exhibit the class of problems, that converge to a global minimum. We also will provide an additional show case in section 2 of how the integer linear program formulation of the travelling salesman problem can be transformed into a non-linear program. This enables us to start of with the *dynamic embedding algortihm's* ability to determine if the problem has no solution i.e. is undecidable i.e. the suboptimal set $\hat{\Omega} = \emptyset$. So our fundamental question is

> Is a there a transformation for diophantine equations into Non-Linear Problems such that non-existence of a solution can be defined?

In order to provide deterministic answers to the question $\hat{\Omega} = \emptyset$ or not, we need to take a look at the proceeding sections of this thesis. It is going to be exciting.

2 Transformation of $\mathcal{NP}$-hard\complete Problems

2.1 Penalty Method for Constrained Problems

This section shows how problems constrained by inequality, as well as equality conditions, can be formulated as an unconstrained problem. The cutback of this approach will be that the resulting unconstrained problem will be "more" non-convex, meaning the issue of choosing the right set of active inequality constraints is transformed to the problem of having many more local minima. We here only show what is all ready in use and how to reduce a constrained problem to an unconstrained one. This is technique is found in many good non-linear optimization books. For the purposes in this thesis we state a general problem. Let us denote $\Omega \subset \mathbb{R}^N$, $N \in \mathbb{N}$ the feasible domain and $f : \Omega \to \mathbb{R}$ the objective function. With $D_1, D_2 \subset \mathbb{R}^N$. Let $h_i : D_1 \to \mathbb{R}$ the i-th equality constraint and $g_j : D_2 \to \mathbb{R}$ the j-th inequality constraint.

$$\begin{aligned} \text{obj.:} \quad & \min_{x \in \Omega} \quad f(x) \\ \text{s.t.:} \quad & h_i = \quad 0, \quad i = 1, \ldots, M, \quad M \in \mathbb{N} \\ & g_j \geq \quad 0, \quad j = 1, \ldots, K, \quad K \in \mathbb{N} \end{aligned} \tag{2.1.1}$$

The according penalty function $\mathcal{L} : \mathbb{R}^N \supseteq D \mapsto \mathbb{R}$ with $\tau, \sigma \in \mathbb{R}$ the penalty parameters such that the unrestricted problem

$$\min_{x \in \Omega} \mathcal{L}(x) := f(x) + \tau \sum_{i=1}^{M} h_i(x)^2 + \sigma \sum_{j=1}^{K} [g_i(x)]_-^2 \tag{2.1.2}$$

has the same solution as the constrained problem. The penalty method usually operates by iteratively applying the optimizer to the problem in the unrestricted form. After the optimizer terminates, the penalty parameters τ and\or σ are increased by a rate of choice. There is no guarantee that the minimum which the penalty method computes, is also the global minimum when we are dealing with general non-linear non-convex problems.

2.2 Penalty Functions as a Homotopy

We present an advanced approach on how to slowly exciting inequality constraints into a given optimization problem. Regarding only the inequality constraints, the according penalty function looks like this

$$\mathcal{L}(x) := f(x) + \sigma \sum_{j=1}^{K} [g_i(x)]_-^2 \tag{2.2.1}$$

An alternative approach to how to excite the inequality constraints is by defining a homotopy function on the problem in the following. Let us define $\mathcal{H}_L : D \times [0,1] \to \mathbb{R}$

$$\mathcal{H}_{f,g}(x,\epsilon) := (1-\epsilon)f(x) + \epsilon \left(f(x) + \hat{\sigma} \sum_{j=1}^{K} [g(x)_i]_-^2 \right) \tag{2.2.2}$$

If we choose $\hat{\sigma}$ to be very high, i.e. $\hat{\sigma} >> 1$, we can highly penalize the objective function yielding good results as an alternative to the penalty function revisited in the previous section. Notice that $\mathcal{H}_L(x,0) = f(x)$ and $\mathcal{H}_L(x,1)$ is the objective function only with a large scaled penalty term. Without further analysis, we state the algorithm for the homotopy function on the penalty function as follows

Algorithm 1: Penalty Method

Data:
$x_0^* \in \mathbb{R}^N$, the initial value
integer $m > 0$ for the number of optimization runs
initial real number $\sigma_0 \geq 0$
penalty rate $\alpha > 0$
the homotopy on the penalty function $\mathcal{H}_L(x,0) = f(x)$
$\Delta\epsilon = 1/m$, $m \in \mathbb{N}$, m the number of steps
Result:
x^* minimum of $\mathcal{H}_L(x,1)$

1 **for** $k = 0, \ldots, m$ **do**
2 solve $x_k^* = \underset{y \in \Omega}{\operatorname{argmin}}\ \mathcal{H}_L(y, \epsilon_k)$
3 update homotopy parameter $\epsilon_{k+1} = \epsilon_k + \Delta\epsilon$
4 **return** $x^* = x_m$

2.3 Traveling Salesman Problem

A popular way to solve the $\mathcal{NP}$-hard travelling salesman problem is by integer linear programming. Let n selected cities in the salesman's tour be the set of vertices $\mathbf{V}$ of a graph. The set of edges $\mathbf{E}$ of the graph correspond to different connections between each city. The graph is complete, that is, there exists a connection between each city. To every connection we associate a binary variable.

$$x_{ij} = \begin{cases} 1, & \text{if edge } (i,j) \in \mathbf{E} \text{ is in tour} \\ 0, & \text{otherwise.} \end{cases} \tag{2.3.1}$$

Because the graph is undirected we have $x_{ij} = x_{ji}$ therefore it suffices to only regard $i < j$. If we define the distance between each city to be d_{ij}. The entire distance travelled by the salesman adds up with

$$\sum_{(i,j)\in E} d_{ij}x_{ij} \tag{2.3.2}$$

In order to establish exactly one incoming edge and one outgoing edge to every city, we write the constraint

$$\sum_{j\in\mathbf{V}} x_{ij} = 2 \quad \forall i \in \mathbf{V} \tag{2.3.3}$$

Feasible solutions of the minimal distance with the above constraint will allow for subtours. This we want to avoid by formulating the following constraint

$$\sum_{(i,j)\in\mathbf{S},i\neq j} x_{ij} \leq |\mathbf{S}|-1, \quad \forall \mathbf{S} \subset \mathbf{V}, \mathbf{S} \neq \emptyset \tag{2.3.4}$$

All in all the optimization problem looks like this.

$$\begin{aligned}
\text{obj.:} \quad & \min_{x\in\Omega} \sum_{(i,j)\in E} d_{ij}x_{ij} \\
\text{s.t.:} \quad & \sum_{j\in\mathbf{V}} x_{ij} = 2 \quad \forall i \in \mathbf{V} \\
& \sum_{(i,j)\in\mathbf{S},i\neq j} x_{ij} \leq |\mathbf{S}|-1, \quad \forall \mathbf{S} \subset \mathbf{V}, \mathbf{S} \neq \emptyset \\
& x_{ij} \in \{0,1\}
\end{aligned} \tag{2.3.5}$$

The constraint basically excludes solutions, that have a certain number of edges. In practice a solution of the problem is found without the subtour constraint and the resulting subtour solution is then taken to be the subtour in the optimization problem with the subtour constraint. The process consists of taking the solutions of the problem and then consecutively adding the constraints for the subtours. This approach makes sense for integer linear programming software packages today and constraints handled like that are called lazy constraints. However if we would like to solve the problem with one run, we would need to rewrite it without the subtours in the following way

$$\begin{aligned}
\text{obj.:} \quad & \min_{x\in\Omega} \sum_{(i,j)\in E} d_{ij}x_{ij} \\
\text{s.t.:} \quad & \sum_{j\in\mathbf{V}} x_{ij} = 2 \quad \forall i \in \mathbf{V} \\
& \sum_{(i,j)\in\mathbf{S},i\neq j} x_{ij} = |\mathbf{V}|-1 \\
& x_{ij} \in \{0,1\}
\end{aligned} \tag{2.3.6}$$

This problem will only need to be solved once but is harder to calculate in terms of integer linear programming (ILP). However we do not want to apply discrete optimization theory to this problem. We attempt to reformulate the problem into an non-linear programming problem (NLP). Applying the stated theory about penalty methods, which we review in section 2.1, we are able to write an unrestricted non-linear problem, that will only contain the equality constraints.

$$\tilde{L} = \sum_{(i,j)\in\mathbf{V}} (x_{ij}d_{ij})^2 + \sum_{i=1}^{|V|} \left(\left[\sum_{j=1}^{|V|} x_{ij} \right] - 2 \right)^2 + \left(\left[\sum_{(i,j)\in\mathbf{V}} x_{ij} \right] - (|V|-1) \right)^2 \tag{2.3.7}$$

Here we squared the objective function in order to make it coercive. In this case we only polish the problem for this purpose. We missed out the constraint to integer numbers of the solution and now model it with the following penalty terms in addition.

$$L = \tilde{L} + \sigma \left[\sum_{(i,j)\in\mathbf{V}} x_{ij}^2 \cdot (x_{ij} - 1)^2 \right] \tag{2.3.8}$$

2.4 Arithmetization of The Satisfiability Problem

As we mentioned in the introduction, the satisfiability problem (SAT) can be transformed into a diophantine equation. This is done by a technique known as arithmetization. That means that a boolean formula can be transformed into a diophantine multivariate polynomial such as (1.5.1). In order to overcome a modelling issue with the resulting arithmetic representation of a general boolean formula, we want to also mention, that any boolean formula can be transformed into a 3-conjunctive normal form (3-CNF), which has been proven by Stephen Cook 1971. See a good book on complexity theory for further reference. So if SAT is always translatable to 3-CNF, we might as well only deal with boolean formulas of the form

$$C_1 \wedge C_2 \wedge \ldots \wedge C_M \tag{2.4.1}$$

where each C_i is an $\vee$ of three or less literals. To make this more specific

$$C_m = (x_{i_m} \vee x_{j_m} \vee x_{k_m}) \tag{2.4.2}$$

where i_m, j_m and k_m are indizes of the first, second and third literal in the m-th clause. Let us introduce the transformations for the arithmetization of such a boolean formula.

1. $x \to x$
2. $\bar{x} \to (1 - x)$
3. $x \wedge y \to x \cdot y$
4. $x \vee y \vee z \to 1 - (x-1)(y-1)(z-1)$

Applying these transformations, we achieve a multidimensional polynomial formulation for any boolean formula. More so we can translate the resulting diophantine equation into a non-linear problem. To constrain the solution of the problem to be integers only, we simply apply

$$\sum_{i=n}^{N} x_i^2 \cdot (x_i - 1)^2 \tag{2.4.3}$$

as a penalty term in the unrestricted formulation. We refer to the introduced penalty method in section 2.1.

Every boolen formula in 3-CNF

$$\bigwedge_{m=1}^{M} (x_{i_m} \vee x_{j_m} \vee x_{k_m}) \tag{2.4.4}$$

can be transformed into a mixed integer non-linear problem of the form

$$\begin{aligned} &\text{obj.:} \quad \underset{x\in[0,1]}{\arg\,\text{zero}} \quad \prod_{m=1}^{M} (1 - (x_{i_m} - 1)(x_{j_m} - 1)(x_{k_m} - 1)) \\ &\text{s.t.:} \qquad\qquad x_{i_m}, x_{j_m}, x_{k_m} \in \{0, 1\} \end{aligned} \tag{2.4.5}$$

With the objective function denoted as $f(x_{i_m}, x_{j_m}, x_{k_m})$, where $i_m, j_m, k_m \in \mathcal{I} \subset \mathcal{P}(\{1, \ldots, N\})$ and N the number of literals, the homotopy on the penalty function is then

$$\begin{aligned} \mathcal{H}_{f,g}(\epsilon) = (1 - \epsilon) f(x_{i_m}, x_{j_m}, x_{k_m}) + \\ + \epsilon \left(f(x_{i_m}, x_{j_m}, x_{k_m}) + \hat{\sigma} \sum_{i}^{N} x_i^2 \cdot (x_i - 1)^2 \right) \end{aligned} \tag{2.4.6}$$

The satisfiability of the original 3-CNF formula is then equivalent to the solvability of

$$\mathcal{H}_{f,g}(1) = 1 \tag{2.4.7}$$

In the above transformations we neglected the negated literals. This can be repaired by defining augmented literals of the like $x_m \in \{y_m, \bar{y}_m\}$.

3 Projected Newton Method

We introduce a global optimization method, that solves for global minima of continuously differentiable functions. This approach has shown to be very effective in the application of protein-ligand docking as we will exhibit in section 3.5.3.

3.1 Construction

Definition 3.1.1. Let $\Omega \in \mathbb{R}$ be an open set and $F : \Omega \mapsto \mathbb{R}$, then $b \in \mathbb{R}$ is the lower bound of F if

$$\forall x \in \Omega : \qquad F(x) \geq b \tag{3.1.1}$$

and $a \in \mathbb{R}$ is upper bound of F if

$$\exists x \in \mathbb{R} : \qquad F(x) \leq a \tag{3.1.2}$$

We begin the construction with the Taylor series of F in at $\xi \in \Omega$.

$$F(x) = F(\xi) + \nabla F(\xi)(x - \xi) + \sum_{\alpha=2}^{n} (x - \xi)^{\alpha-1} \frac{D^{\alpha} F(\xi)}{\alpha!}(x - \xi) \tag{3.1.3}$$

Since F is continuously differentiable we have

$$F(x) = F(\xi) + \nabla F(\xi)(x - \xi) \tag{3.1.4}$$

We extend with $\nabla F(\xi)$, and solve for x

$$x = a - \frac{F(a)}{\|\nabla F(a)\|^2} \nabla F(a) \tag{3.1.5}$$

From here we formulate an iteration where $k \in \mathbb{N}_0$, $x = x^{(k+1)}$ and $a = x^{(k)}$. Also we define function $f : \Omega \mapsto \mathbb{R}$, natural number $s \in \mathbb{N}_0$ and bound $b_s \in \mathbb{R}$. From here we define F more specifically

$$F(x) = f(x) - b_s \tag{3.1.6}$$

In this way the projected newton method is constructed

$$x^{(k+1)} = x^{(k)} + \frac{b_s - f(x^{(k)})}{\left\|\nabla f(x^{(k)})\right\|^2} \nabla f(x^{(k)}) \tag{3.1.7}$$

We prove the local convergence and the convergence speed of the method when $k \mapsto \infty$ in the next section 3.3. $|x_{k+1} - x_k| < \epsilon$, where $\epsilon > 0$. To make this method become a global optimizer, we introduce an explicit operation from $s \mapsto s + 1$. That means for every iteration $k \mapsto k + 1$, $b_s \in \mathbb{R}$ is a constant. The intention behind that is to shift the function by a value of b_s up or down and investigate if it the function has a point $x \in \Omega$, such that $f(x) = b_s$ is fulfilled. Starting from definition (3.1.1) an interval $[a, b]$ is initialized. So this is a test

$$\forall x \in \Omega, b \in \mathbb{R}: \quad f(x) \geq b \tag{3.1.8}$$

$$a \in \mathbb{R}, \exists x \in \Omega: \quad f(x) \leq a \tag{3.1.9}$$

This interval is an estimation of the function's value in a global minimum of f We denote $b_0 = a$ and $b_1 = b$, initialize $s = 2$, iterate (3.1.7) and therefore solve for $x \in \Omega$, solving equation (3.1.6). We distinguish two cases. As we will prove in section 3.3, iteration (3.1.7) converges after a finite number of steps. In this case b_s is computed by (3.1.10). If (3.1.7)does not converge after a given number of steps K, we assume that (3.1.6) has no solution to a given b_s. In this case b_s is computed by (3.1.11). All in all we are just interested if iteration (3.1.7) converges after K steps or not. In other words, we are interested in the existence and non existence of $x \in \Omega$ full filling (3.1.6). We put this together in an algorithm for later reference.

Algorithm 2: Level-Set Bisection

Data: continuously differentiable map $f : \mathbb{R}^n \to \mathbb{R}$,
iterative non linear equation solver **S**,
a lower bound of f,
b upper bound of f,
K the maximum number of steps

Result: value/level of global minimum of f

$b_s = a$
$b_{s+1} = b$
$s = 2$
while solver **S** solves for solution $x \in \Omega$ of $f(x) - b_s = 0$ **do**
  if iterative solver converges before K steps **then**

$$b_s = b_{s-1} - \frac{|b_{s-1} - b_{s-2}|}{2} \tag{3.1.10}$$

$$s = s + 1$$

  if iterative solver does not converge after K steps **then**

$$b_s = b_{s-1} + \frac{|b_{s-1} - b_{s-2}|}{2} \tag{3.1.11}$$

$$s = s + 1$$

  if $|b_s + 1 - b_s| <$ TOL **then**
    STOP

In the fist section we went over theorem 1.4.3 that proves the Level-Set Bisection algorithm (provided $K \to \infty$) to be convergent to the global minimum of the objective function. This only holds true if the decision of a non-converging iterative solver is a definite one. Most of the effort will go in overcoming that decision problem but so far we can build a good heuristic solver for with the projected newton method in section 3 and the Level-Set Bisection method. In order to exploit the advantages of the combined algorithms, we first make some adjustments to some special cases.

3.2 Special Cases

If we look at (3.1.7) more closely, we see that the case of $\|\nabla f(x_k)\|^2 = 0$ has to be appropriately coped with. In this case x_k is a critical point. Also here we have a distinction between two cases.

1. $f(x_k) = b_s$: $F(x)$ has a root. Compute b_s with (3.1.10)

2. $f(x_k) \neq b_s$: x_k is a critical point. Shift x_k by $r \in \Omega$ a random number such that $x_k + r \in \Omega$

Algorithm 3: Global Minimizer

Data: f objective function,
x_0 the start vector,
a lower bound of f,
b upper bound of f,
K the maximum number of steps,
ϵ the desired precision (absolute)

Result: x^* the global minimum of f

$x_{k+1} = x_0$;
$b_s = a$;
$b_{s+1} = b$;
$s = 2$;
$k = 0$;
while $|b_{s+2} - b_{s+1}| > \epsilon$ **do**
  $k = k + 1$;
  $x_k = x_{k+1}$;
  $x_{k+1} = x_k - \frac{f(x_k) - b_s}{\|\nabla f(x_k)\|^2} \nabla f(x_k)$;
  if $\|\nabla f(x)\|^2 = 0$ **then**
    $x_0 = x_{k+1}$;
    $b_s = b_{s-1} + \frac{|b_{s-1} - b_{s-2}|}{2}$;
    $s = s + 1$;
  else if $k = K$ **then**
    $x_{k+1} = x_0$;
    $b_s = b_{s-1} - \frac{|b_{s-1} - b_{s-2}|}{2}$;
    $s = s + 1$;
  else if $|x_{k+1} - x_k| < \epsilon$ **then**
    $x_0 = x_{k+1}$;
    $b_s = b_{s-1} + \frac{|b_{s-1} - b_{s-2}|}{2}$;
    $s = s + 1$;
return $x^* = x_{k+1}$

3.2.1 Correction of Error

If the reader has not noticed yet, it is important to mention, that the convergence of the method (3.1.6) can only be assured for functions, that full fill the properties for convergence from section 3.3 globally. Generally it is not given to find a bound K for the number of steps in order to decide that the method converges or not. In most cases this finite number can be extremely large. In order to make the computation more efficient, we will describe a modification of the above method in order to have

precise computations made, that choose a large K but will not have to exploit too much computational resource. The error that we have to reduce, emerges when the method decides that $f - b_s$ has no root. This decision is made after K steps. However it may occur that the method converges after $K + 1$ steps. Basically what we choose to do, is to continue the computation at b_s and b_{s+1} in parallel. So the primary computation at b_{s+1} continues in the same way as described in algorithm 3 but at the same time the secondary computation at b_s is carried on further. If the secondary computation at b_s converges, the primary computation at b_{s+1} is interrupted and put to a new start at b_{s+1}, which is recomputed on the basis of b_s. The correcting routine, the one that carries on with the computation of the former decided convergence, is running as long as the first routine has decided another convergence or if it reaches a large number of Iterations M. We express the above in algorithm 4.

Algorithm 4: Global Minimizer with correction

Data: f objective function, x_0 the start vector, a lower bound of f, b upper bound of f, K the maximum number of steps for the primary computation, M the maximum number of steps for the secondary computation, ϵ the desired accuracy (absolut)

Result: x^* the global minimum of f

$x_{k+1}^{prim} = x_0$, $b_s^{prim} = a$, $b_{s+1}^{prim} = b$, $s = 2$, $t = 2$, $k = 0$, $m = 0$, $final = false$;
Inizialize stack;
while $|b_{s+2} - b_{s+1}| > \epsilon$ AND $m < M$ **do**
  if $final = false$ **then**
    $k = k + 1$, $x_k^{prim} = x_{k+1}^{prim}$, $x_{k+1}^{prim} = x_k^{prim} - \frac{f(x_k^{prim}) - b_s^{prim}}{\left\|\nabla f(x_k^{prim})\right\|^2} \nabla f(x_k^{prim})$;
    if $\left\|\nabla f(x_{k+1}^{prim})\right\|^2 = 0$ **then**
      $x_0 = x_{k+1}$, $b_s^{prim} = b_{s-1}^{prim} + \frac{|b_{s-1}^{prim} - b_{s-2}^{prim}|}{2}$, $s = s + 1$;
    else if $k = K$ **then**
      save all data that has been computed so far on the stack ;
      Initialize x_{k+1}^{sec}, x_k^{sec}, b_s^{sec}, b_{s-1}^{sec}, b_{s-2}^{sec} with previously computed data and set $m = k$;
      $x_{k+1}^{prim} = x_0^{prim}$, $b_s^{prim} = b_{s-1}^{prim} - \frac{|b_{s-1}^{prim} - b_{s-2}^{prim}|}{2}$, $s = s + 1$;
    else if $\left|x_{k+1}^{prim} - x_k^{prim}\right| < \epsilon$ **then**
      $x_0^{prim} = x_{k+1}^{prim}$, $b_s^{prim} = b_{s-1}^{prim} + \frac{|b_{s-1}^{prim} - b_{s-2}^{prim}|}{2}$, $s = s + 1$;
  if $final = true$ **then**
    $m = m + 1$, $x_m^{sec} = x_{m+1}^{sec}$, $x_{m+1}^{sec} = x_m^{sec} - \frac{f(x_m^{sec}) - b_t^{sec}}{\|\nabla f(x_m^{sec})\|^2} \nabla f(x_m^{sec})$;
    if $\left\|\nabla f(x_{m+1}^{sec})\right\|^2 = 0$ **then**
      perform Pop-operation on stack;
      Initialize x_{m+1}^{sec}, x_m^{sec}, b_s^{sec}, b_{t-1}^{sec}, b_{t-2}^{sec} with values from stack;
      $final = false$, $x_0^{prim} = x_{m+1}^{sec}$, $b_t^{prim} = b_{t-1}^{sec} + \frac{|b_{t-1}^{sec} - b_{s-2}^{sec}|}{2}$, $t = t + 1$;
    else if $\left|x_{m+1}^{sec} - x_m^{sec}\right| < \epsilon$ **then**
      $x_0^{prim} = x_{m+1}^{sec}$, $b_t^{prim} = b_{t-1}^{sec} + \frac{|b_{t-1}^{sec} - b_{t-2}^{sec}|}{2}$, $t = t + 1$;
return $x^* = x_{k+1}^{prim}$

3.3 Local Convergence

For the analysis we use a theorem but don't prove it.

Theorem 3.3.1. *Let $B : X \mapsto X$ a bounded linear operator in the Banach-Space X with $\|B\| < 1$. Then operator $I - B$ with the unit operator I is invertable, i.a. the equation*

$$x - Bx = y \tag{3.3.1}$$

has a solution $x \in X$ for every $y \in X$. The inverse operator $(I-B)^{-1}$ is bounded by

$$\left\|(I-B)^{-1}\right\| \leq \frac{1}{1-\|B\|} \tag{3.3.2}$$

We look at local convergence properties of the projected gradient method.

Theorem 3.3.2. *Let $G \subset \mathbb{R}^m$ an open and convex set. Let $f : G \mapsto \mathbb{R}$ be continuous and differentiable. For every $x^{(0)} \in G$ we assume that f satisfies every norm $\|\cdot\|$ on $\mathbb{R}^m$ with the following properties*

(i) There exists a number $\gamma > 0$ with
$\|f'(x) - f'(y)\| \leq \gamma \|x-y\|, \quad \forall x, y \in G$

(ii) For all $x \in G$ exists $\frac{1}{\|f'(x)\|}$ and a number $\beta > 0$ such that
$\frac{1}{\|f'(x)\|} \leq \beta$, $\forall x \in G$

(iii) With $\alpha := \frac{\|f'(x)f(x)\|}{\|f'(x)\|^2}$ it holds
$\rho := \alpha\beta\gamma < \frac{1}{2}$

(iv) With $r := 2\alpha$ it holds $B(x^{(0)}, r) := \left\{x : \left\|x - x^{(0)}\right\| \leq r\right\} \subset G$

Then the following is true

1) The projected gradient method

$$x^{(n+1)} = x^{(n)} - \frac{f(x^{(n)})}{\left\|f'(x^{(n)})\right\|^2} f'(x^{(n)})$$

with $n \in \mathbb{N}_0$ and start vector is well-defined.

2) the series $(x^{(n)})$ converges to the root x^* of f with
$\left\|x^{(n)} - x^*\right\| \leq 2\rho^{2^n - 1}$, $n \in \mathbb{N}_0$. *In the sphere $B(x^{(0)}, r)$ $x^*$is the only root of f*

Proof. a) **Preparational Step**

We begin with the mean value theorem in multiple dimensions. We expand with $f'(z)(y-x), \forall x, y, z \in G$ and choose an appropriate $\xi = y + \theta h$, $\theta \in [0,1], h = x - y$

$$f(y) - f(x) = f'(\xi)(y-x)$$
$$f(y) - f(x) - f'(z)(y-x) = f'(\xi)(y-x) - f'(z)(y-x)$$

We find a bound under the assumption (iv)

$$\begin{aligned}\left\|f(y) - f(x) - f'(z)(y-x)\right\| &= \left\|(f'(\xi) - f'(z))(y-x)\right\| \\ &\leq \gamma \|y + \theta h - z\| \|y - x\| \\ &= \gamma \|y + \theta x - \theta y - z\| \|y - x\|\end{aligned}$$

We choose $\theta = 1/2$

$$\leq \frac{1}{2}\gamma \|y-x\| (\|y-z\| + \|x-z\|)$$

with $z = x$ follows

$$\left\|f(y) - f(x) - f'(x)(y-x)\right\| \leq \frac{\gamma}{2}\left\|y-x\right\|^2 \forall x, y \in G \tag{3.3.3}$$

and with $z = x^{(0)}$

$$\left\|f(y) - f(x) - f'(x^{(0)})(y-x)\right\| \leq r\gamma\left\|y-x\right\| \forall x, y \in B(x^{(0)}, r) \tag{3.3.4}$$

b) **The Iteration Is Well-Defined**

We show that the solution series $(x^{(0)})$ holds:

$$\left\|x^{(n)} - x^{(0)}\right\| < r \tag{3.3.5}$$

$$\left\|x^{(n)} - x^{(n-1)}\right\| \leq \alpha\rho^{2^n-1} \tag{3.3.6}$$

We prove two inequalities via induction:

Initial step:

n = 1:
$$\left\|x^{(1)} - x^{(0)}\right\| = \left\|\frac{f'(x^{(0)})}{\left\|f'(x^{(0)})\right\|^2} f(x^{(0)})\right\| \overset{(iii)}{=} \alpha \overset{(iv)}{=} \frac{r}{2} < r$$

Induction step:

n = n+1:
$$\begin{aligned}\left\|x^{(n+1)} - x^{(n)}\right\| &= \left\|\frac{f'(x^{(n)})}{\left\|f'(x^{(n)})\right\|^2} f(x^{(n)})\right\| \\ &\overset{(ii)}{\leq} \beta\left\|f(x^{(n)})\right\| \\ &= \beta\left\|f(x^{(n)}) - f(x^{(n-1)}) - f'(x^{(n-1)})(x^{(n)} - x^{(n-1)})\right\| \\ &\overset{(3.3.3)}{\leq} \frac{1}{2}\beta\gamma\left\|x^{(n)} - x^{(n-1)}\right\|^2 \\ &\overset{IV}{\leq} \frac{1}{2}\beta\gamma\left[\alpha\rho^{2^{n-1}-1}\right]^2 \\ &\overset{(iii)}{=} \frac{1}{2}\alpha\rho^{2^n-1} < \alpha\rho^{2^n-1}\end{aligned}$$

Thereby we showed (3.3.6). We use this for (3.3.5). Under usage of the triangle inequality

$$\begin{aligned}\left\|x^{(n+1)}-x^{(0)}\right\| &\overset{D.-U.}{\leq} \left\|x^{(n+1)}-x^{(n)}\right\|+\cdots+\left\|x^{(1)}-x^{(0)}\right\| \\ &\overset{(3.3.6)}{\leq} \alpha(1+\rho+\rho^3+\rho^7+\cdots+\rho^{2^n-1}) \\ &< \frac{\alpha}{1-\rho} \leq 2\alpha = r\end{aligned}$$

Thereby (3.3.5) also holds.

c) **Existence Of The Limit And Error Approximation**

Sei $k \in \mathbb{N}$

$$\begin{aligned}\left\|x^{(n)}-x^{(n+k)}\right\| &\leq \left\|x^{(n)}-x^{(n+1)}\right\|+\cdots+\left\|x^{(n+k-1)}-x^{(n+k)}\right\| \\ &\overset{(3.3.6)}{\leq} \alpha(\rho^{2^n-1}+\rho^{2^{n+1}-1}+\cdots+\rho^{2^{n+k-1}-1}) \\ &= \alpha\rho^{2^n-1}(1+\rho^{2^n}+\cdots+(\rho^{2^n})^{2^{k-1}}) \\ &\overset{k\to\infty}{<} 2\alpha\rho^{2^n-1} \overset{n\to\infty}{=} 0\end{aligned}$$

$x^{(n)}$ is a Cauchy-Sequence. Since $\mathbb{R}^m$ is complete

$$x^* = \lim_{n\to\infty} x^{(n)}$$

exists.

d) **Proof of x^* being a root of f**

Starting from the definition of the method

$$\begin{aligned}x^{(n+1)} &= x^{(n)} - \frac{f(x^{(n)})}{\left\|f'(x^{(n)})\right\|^2} f'(x^{(n)}) \\ \Leftrightarrow f(x^{(n)})f'(x^{(n)}) &= \left\|f'(x^{(n)})\right\|^2 (x^{(n+1)}-x^{(n)}) \\ f(x^{(n)}) &= f'(x^{(n)})(x^{(n+1)}-x^{(n)}) \\ &\overset{(3.3.3)}{\leq} \left\|f'(x^{(n)})-f'(x^{(0)})+f'(x^{(0)})\right\| \left\|x^{(n+1)}-x^{(n)}\right\| \\ &\leq \left\{\gamma\left\|x^{(n)}-x^{(0)}\right\|+\left\|f'(x^{(0)})\right\|\right\} \underbrace{\left\|x^{(n+1)}-x^{(n)}\right\|}_{\overset{(3.3.6)}{\leq} \alpha\rho^{2^n-1}} \\ &\overset{n\to\infty}{=} 0\end{aligned}$$

Therfore $\lim_{n\to\infty} f(x^{(n)}) = 0$. Because of continouity of f, $f(x^*) = 0$ holds.

e) **Uniqueness of the root in B.**

With function $g : B(x^{(0)}, r) \mapsto \mathbb{R}^m$,

$$g(x) := x - \frac{f(x)}{\left\|f'(x^{(0)})\right\|^2} f'(x^{(0)})$$

Starting from

$$\begin{aligned} g(x) - g(y) &= \frac{f'(x^{(0)})f(y)}{\left\|f'(x^{(0)}))\right\|^2} - \frac{f'(x^{(0)})f(x)}{\left\|f'(x^{(0)}))\right\|^2} - (y - x) \\ \Leftrightarrow f'(x^{(0)})(g(x) - g(y)) &= f(y) - f(x) - f'(x^{(0)})(y - x) \end{aligned}$$

Estimation under the assumptions (ii) and (iii) of the theorem, the triangle inequality and (3.3.5) we get

$$\left\|f'(x^{(0)})(g(x) - g(y))\right\| = \left\|f(y) - f(x) - f'(x^{(0)})(y - x)\right\| \tag{3.3.7}$$

$$\Leftrightarrow \frac{1}{\left\|f'(x^{(0)})\right\|} \left\|f'(x^{(0)})(g(x) - g(y))\right\| \leq \|(g(x) - g(y))\| \tag{3.3.8}$$

$$\leq \frac{1}{\left\|f'(x^{(0)})\right\|} \left\|f(y) - f(x) - f'(x^{(0)})(y - x)\right\| \tag{3.3.9}$$

$$\leq \beta r\gamma \|y - x\| \tag{3.3.10}$$

$$\leq 2\rho \|y - x\|, \forall x, y \in B(x^{(0)}, r) \tag{3.3.11}$$

Because of $\rho < 1/2$, g the method, is contractive and has at most one fixed point according to Banach's theorem. g also has at most one fixed point in $B(x^{(0)}, r)$.

The uniqueness of f concludes from the equivalence of the fixed point equation $x = g(x)$ to $f(x) = 0$.

□

Theorem 3.3.3. *Let $G \subseteq \mathbb{R}^m$ be an open set, $f : G \mapsto \mathbb{R}$ two times continuously differentiable and x^* the root of f with $f'(x^*) \neq 0$. Then a $\delta > 0$ exists such that the projected gradient method with the start vector $x^{(0)}$ converges with $\left\|x^{(0)} - x^*\right\| < \delta$ to x^*.*

Proof. Let there be a closed sphere $B(x^*, R) := \{x : \|x - x^*\| \leq R\} \subseteq G$.

Because of the partial derivatives of the second order of f we can use the mean value theorem f' and write $\gamma > 0$

$$\left\|f'(x) - f'(y)\right\| \leq \gamma \|x - y\|, \qquad \forall x, y \in B(x^*, R) \tag{3.3.12}$$

We start from the identity

$$\begin{aligned}\left\|f'(x)\right\| &= \left\{1 + \frac{\|f'(x^*)\| \left[\|f'(x)\| - \|f'(x^*)\|\right]}{\|f'(x^*)\|^2}\right\} \left\|f'(x^*)\right\| \\ &:= (1 - B) \left\|f'(x^*)\right\|\end{aligned}$$

In order put theorem 3.3.1 to use, $\|B\| < 1$ has to be full filled. We use estimation (3.3.12) and we get for $\|B\|$

$$\begin{aligned}\|B\| &= \left\| \frac{\|f'(x^*)\| \left[\|f'(x)\| - \|f'(x^*)\|\right]}{\|f'(x^*)\|^2} \right\| \\ &\leq \left\| \frac{\|f'(x^*)\| \left[\|f'(x) - f'(x^*)\|\right]}{\|f'(x^*)\|^2} \right\| \\ &\leq \frac{1}{\|f'(x^*)\|} \gamma \|x - x^*\| \leq \frac{1}{\|f'(x^*)\|} \gamma R \overset{!}{<} 1\end{aligned}$$

For one $\beta > 0$ it holds that

$$\frac{1}{\|f'(x)\|} \leq \beta, \qquad \forall x \in B(x^*, R)$$

If we choose $R < \frac{1}{\beta\gamma}$ it follows that $\|B\| < 1$. Because of continuity f and $f(x^*) = 0$ we can find a number $\delta < \frac{1}{2}R$ with $\forall x^{(0)} : \left\|x^{(0)} - x^*\right\| < \delta$such that

$$\begin{aligned}\left\|f(x^{(0)}) - f(x^*)\right\| \leq & \quad \gamma \left\|x^{(0)} - x^*\right\| \\ \left\|f(x^{(0)})\right\| \leq & \quad \gamma \left\| \tfrac{f'(x^{(0)})}{\|f'(x^{(0)})\|^2} f(x^{(0)}) \right\| \\ \leq & \quad \gamma\beta \left\|f(x^{(0)})\right\| \qquad < \gamma\delta < \gamma\tfrac{1}{2}R \\ \Leftrightarrow & \quad \left\|f(x^{(0)})\right\| < \tfrac{1}{2\beta^2\gamma}\end{aligned}$$

If we define $\alpha := \frac{\|f(x^{(0)})\|}{\|f'(x^{(0)})\|}$ and $2\alpha < \frac{1}{2}R$

$$\left\|f(x^{(0)})\right\| < min\left\{\frac{R}{4\beta}, \frac{1}{2\beta^2\gamma}\right\}, \qquad \forall x^{(0)} \text{ with } \left\|x^{(0)} - x^*\right\| < \delta$$

$$\rho := \alpha\beta\gamma \leq \left\|f(x^{(0)})\right\| \beta^2\gamma < \frac{1}{2} \quad \text{und} \quad 2\alpha \leq 2\beta \left\|f(x^{(0)})\right\| < \frac{1}{2}R$$

For an open convex sphere $G := B(x^*, R)$ and every $x^{(0)}$ with $\left\|x^{(0)} - x^*\right\| < \delta$ the assumptions of theorem 3.3.2 hold true. Therefore the theorem holds true. □

Remark 3.3.4. $B : X \mapsto X$ is a linear operator where X is Banach-Space, defined by $X := \mathbb{R}$

Theorem 3.3.5. *Under assumption of theorem 3.3.2 there is a $c > 0$ such that the projected gradient method for every starting point $x^{(0)}$ has a convergent sequence $x^{(n)}$ with limit x^* where $n \in \mathbb{N}_0$, it holds*

$$\left\|x^{(n+1)} - x^*\right\| \leq c \left\|x^{(n)} - x^*\right\|^2$$

Proof. By using (3.3.3) we get from

$$\frac{f'(x^{(n)})}{\left\|f'(x^{(n)})\right\|}(x^{(n+1)} - x^*) = \frac{f'(x^{(n)})}{\left\|f'(x^{(n)})\right\|}\left[x^{(n)} - \frac{f'(x^{(n)})}{\left\|f'(x^{(n)})\right\|^2} f(x^{(n)}) + \frac{f'(x^{(n)})}{\left\|f'(x^{(n)})\right\|^2} f(x^*) - x^*\right]$$

the inequality

$$\begin{aligned}
\left\|x^{(n+1)} - x^*\right\| &\leq \frac{1}{\left\|f'(x^{(n)})\right\|} \left\|f(x^*) - f(x^{(n)}) - f'(x^{(n)})(x^* - x^{(n)})\right\| \\
&\leq \underbrace{\frac{\gamma}{2\left\|f'(x^{(n)})\right\|}}_{:=c>0} \left\|x^{(n)} - x^*\right\|^2 \\
&\leq \frac{\beta\gamma}{2} \left\|x^{(n)} - x^*\right\|^2
\end{aligned}$$

where γ is the lipschitz constant and it holds $\boxed{c = \frac{\gamma}{2\left\|f'(x^{(n)})\right\|}}$. □

Theorem 3.3.6. *Let $\Omega \subseteq \mathbb{R}^n$ open and $f : \Omega \mapsto \mathbb{R}$ continuously differentiable. Furthermore let $x^* \in \mathbb{R}^n$ root of f on Ω and the norm $\|\cdot\|$, which is induced by the euclidean scalar product $\langle \cdot, \cdot \rangle : \mathbb{R}^n \mapsto \mathbb{R}$, such that the following properties hold:*

(i) There exists a real number $L > 0$ with $\|f(x) - f(y)\| \leq L \|x - y\|, \qquad \forall x, y \in \Omega$
(ii) For all $x \in \Omega$ exist $\frac{1}{\|f'(x)\|}$ and a real number $\beta > 0$ such that $\frac{1}{\|f'(x)\|} \leq \beta$
(iii) With $\delta > 0$ holds $B(x^*, \delta) := \{x : \|x - x^*\| \leq \delta\} \subset \Omega$

Then it holds:
For every start value $x^{(0)} \in B(x^*, \delta)$ PNM terminates either at the root x^* or it produces a sequence, that converges super linearly to x^*.

Proof. Using $f(x^*) = 0$ and the mean value theorem we get for every $x^{(n)} \in B(x^*, \delta)$, where $\frac{1}{\|f'(x)\|}$ exists and $n \in N_0$

$$\begin{aligned}
x^{(n)} - x^* &= x^{(n)} - \frac{f(x^{(n)})}{\left\|f'(x^{(n)})\right\|^2} f'(x^{(n)}) - x^* \\
&= \frac{f(x^{(n)})}{\left\|f'(x^{(n)})\right\|^2}(f(x^{(n)}) - f(x^*) - f'(x^{(n)})(x^* - x^{(n)}))
\end{aligned}$$

therefore it follows similarly to inequality (3.3.5)

$$\left\|x^{(n+1)} - x^*\right\| \leq \beta\delta L \left\|x^{(n)} - x^*\right\|$$

For sufficiently large n it holds

$$\begin{aligned}\left\|x^{(n+1)} - x^*\right\| &\leq \left\|x^{(n+1)} - x^{(n)}\right\| + \left\|x^{(n+1)} - x^*\right\| \\ &\leq \left\|x^{(n+1)} - x^{(n)}\right\| + \frac{1}{2}\left\|x^{(n)} - x^*\right\|\end{aligned}$$

therefore it follows:

$$\frac{1}{2}\left\|x^{(n)} - x^*\right\| \leq \left\|x^{(n+1)} - x^{(n)}\right\|$$

$(x^{(n)})_{n\in\mathbb{N}_0}$ is a bounded sequence and there exists a closed sphere K, that includes all elements in the sequence and the limit value as well. With $f(x^*) = 0$ and by assumption there exists a $L > 0$ such that

$$\begin{aligned}\frac{\left\|f(x^{(n+1)})\right\|}{\left\|x^{(n+1)} - x^{(n)}\right\|} &= \frac{\left\|f(x^{(n+1)}) - f(x^*)\right\|}{\left\|x^{(n+1)} - x^{(n)}\right\|} \\ &\leq L\frac{\left\|x^{(n+1)} - x^*\right\|}{\left\|x^{(n+1)} - x^{(n)}\right\|} \\ &\leq \underbrace{2L}_{:=c>0}\frac{\left\|x^{(n+1)} - x^*\right\|}{\left\|x^{(n)} - x^*\right\|} \longrightarrow 0\end{aligned}$$

The sequence $(x^{(n)})_{n\in\mathbb{N}_0}$ converges super-linearly for all starting values $x^{(n)} \in B(x^*, \delta)$ specifically also for $n = 0$. Furthermore it holds $\boxed{c := 2L}$. □

As briefly introduced in the beginning of this thesis, we want to proof the polynomial runtime character of this algorithm.

Theorem 3.3.7. *Let the assumptions of Theorem 3.3.2 be given, $\|\cdot\|$ the 2-norm and the input values and variables as in algorithm 4. Let $M \in \mathbb{N}$ the dimension of the domain of f and two constants $a \in \mathbb{N}$ and $b > 0$. If a local step of PNM is executed in $a \cdot M^b \in O(M^b)$ many arithmetic operations, then PNM is an approximation to the minimum of f in polynomial time.*

Proof. Until a level is updated, there are at most K local steps made. Every level b_s is investigated for convergence and a level b_{s-1} checked for divergence. In the worst case

every level set N_{b_s} contains a root of f, but is never found after K steps by the primary computation routine. Since Algorithm 4 checks two levels at the same time, where one is the divergent and the other the convergent, it only need up to $2 \cdot K$ steps.
As soon as the distance between the two levels is smaller $\epsilon > 0$ the algorithm terminates.

$$\|b_{s+1} - b_s\| < \epsilon \tag{3.3.13}$$

in every global step $l \in \mathbb{N}$ level-set bisections take place, such that from

$$\frac{\|b_{sup} - b_{inf}\|}{2^l} \leq \epsilon \tag{3.3.14}$$

we obtain the number of global steps $l_{max} \in \mathbb{N}$ by computing

$$\left\lceil \log_2 \left[\frac{\|b_{sup} - b_{inf}\|}{\epsilon} \right] \right\rceil = l_{max} \tag{3.3.15}$$

Until now algorithm 4 declared every level as divergent and therefore reaches $b_s+1 = b_{sup}$. This makes a combined $2K \cdot l_{max}$ many steps that are executed.
Since the algorithm saved all divergent levels and information of its according last iterates, all x_{2K}^{sec} are tested until the number of iterations $n_i \in \mathbb{N}$, $i \in I := \{1, \ldots, m\}$ is reached, such that convergence can be established for the according level in the local step. Notice that $n_i > 2K, \forall i \in I$. Therefore $l_{sec} := \left\lceil \frac{\|b_{sup} - b_{inf}\|}{\epsilon} \right\rceil$ many levels are iterated until convergence is reached in the local step of the level set bisection. The number of steps is therefore bounded by

$$\sum_{i=1}^{l_{sec}} n_i \leq l_{sec} \cdot \max_{i \in I} \{n_i\} = l_{sec} \cdot n_{max}$$

By assumption the algorithm needs a polynomial number of arithmetic operations for every local step i.e. $a \cdot M^b \in O(M^b)$. Therefore in the worst case

$$\underbrace{(2K \cdot l_{max} + l_{sec} \cdot n_{max}) \cdot a}_{\alpha = konst} \cdot M^b \in O(M^b) \qquad \alpha \in \mathbb{R}_{>0} \tag{3.3.16}$$

The constant α is not dependent of the number of dimensions. Therefore the hypothesis holds

$$PNM \in O(M^b)$$

□

It might have come to the readers attention, that we were only able to make out local convergence properties of the projected newton method. The Level-Set Bisection algorithm 2 is defined for an iterative solver (in our case we use the projected newton

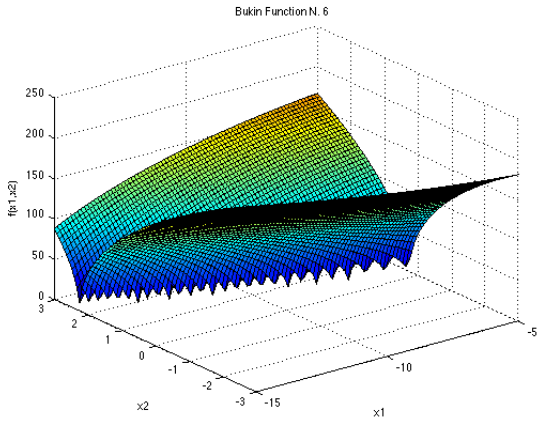
method) with a maximum number of steps K. Ideally we would like to make an indefinite amount of steps in order to be sure if a level-set exists or not. We remind of the idea presented in section 1.4, where we assume that the case of the empty set of suboptimal points, i.e. $\Omega = \emptyset$, is defined. Because we have no suitable definition for that case at hand, we need to apply optimization techniques as introduced in section 2.1 and 2.2.

3.4 Applications

3.4.1 Test Functions

In order to show the algorithm's advantages and disadvantages to a given problem, a selection of multidimensional test functions common to numerical optimization benchmarking is evaluated. We give an overview on what these functions look like, state their known global minima and the according point in the domain.

Bukin Function



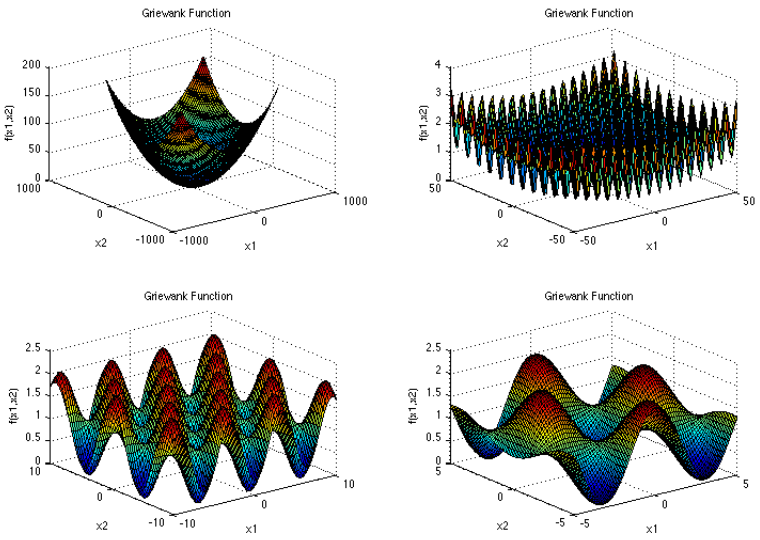
$$f(x_1, x_2) = 100\sqrt{|x_2 - 0.01x_1^2|} + 0.01|x_1 + 10| \tag{3.4.1}$$

We abreviate the function with B. It is defined on the set

$$\Omega = \left\{x \in \mathbb{R}^2 : -3 \leq x_i \leq 2, 1 \leq i \leq 2\right\} \tag{3.4.2}$$

The Bukin function has many local minima, all of which lie in a ridge. The global minimums' value is $B^* = 0$ at x.

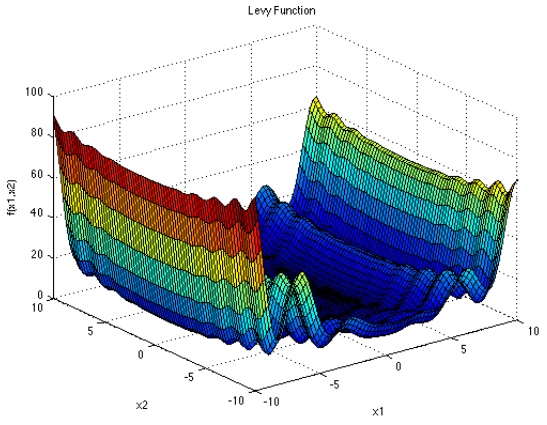
Griewank Function



$$f(x) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{3.4.3}$$

The Griewank function has many widespread local minima, which are regularly distributed. The comlexity is shown in the zoomed-in plots. The function is evaluated on the hypercube $x_i \in [-600, 600]$ for all $i \in \{1, \ldots, d\}$. The global minimum's value is $G^* = 0$ at $x^* = (0, \ldots, 0)$.

Levy

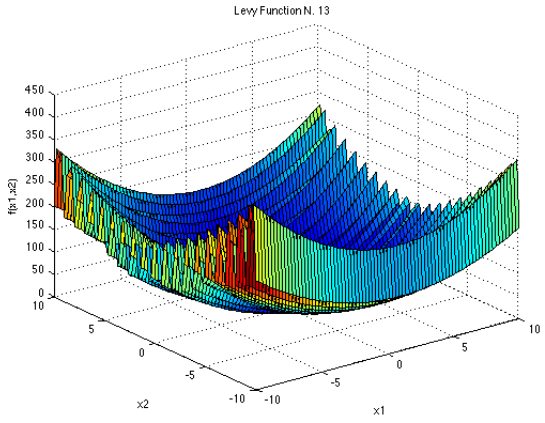


$$f(x) = \sin^2(\pi\omega_1) + \sum_{i=1}^{d-1}(\omega_i - 1)\left[1 + 10\sin^2(\pi\omega_i + 1)\right] + (\omega_d - 1)^2\left[1 + \sin^2(2\pi\omega_d)\right] \quad (3.4.4)$$

$$\text{where } \omega_i = 1 + \frac{x_i - 1}{4}$$

The function is evaluated on the hypercube $x_i \in [-10, 10]$, for all $i = 1, \ldots, d$. The global minimums value is $L^* = 0$, at $x^* = (1, \ldots, 1)$.

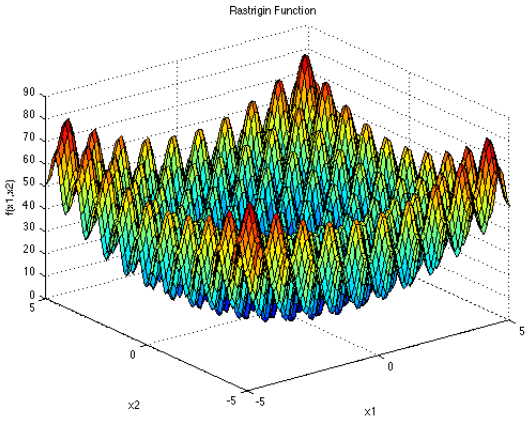
Levy N. 13



$$f(x) = \sin^2(3\pi x_1) + (x_1 - 1)^2\left[1 + \sin^2(3\pi x_2)\right] + (x_2 - 1)^2\left[1 + \sin^2(2\pi x_2)\right] \quad (3.4.5)$$

The function is evaluated on the square $x_i \in [-10, 10]$, for all $i = 1, 2$. The global minimums value is $L13^* = 0$, at $x^* = (1, 1)$.

Rastrigin



$$f(x) = 10d + \sum_{i=1}^{d} \left[x_i^2 - 10\cos(2\pi x_i)\right] \tag{3.4.6}$$

The Rastrigin function has several local minima. It is highly multimodal, but locations of the minima are regularly distributed. It is shown in the plot above in its two-dimensional form. It is evaluated on the hypercube $x_i \in [-5.12, 5.12]$, for all $i = 1, \ldots, d$. The global minimums value is $R^* = 0$, at $x^* = (0, \ldots, 0)$.

3.4.2 Protein-Ligand Docking

It is a fairly big topic to state as an instance for testing among other problems. However we decide to put it into account, since it is a very interesting real-world application. We introduce the topic by mentioning, that we only will deal with so called molecular mechanics models.

Classical Models For Molecular Potentials

Most of the physics in this field is not so well understood, meaning that the modelling suffers of substantial effort to store accurate molecular models in a computer or cluster of computers. To approach this problem of storage complexity, we will step away from energy potentials of quantum mechanics to describe the exact physics of molecules and go more into the direction of evaluating approximations of molecular potentials. As one example of these approximations, we state the OPLS-potential, such that the reader might gain an idea, about what kind of potential we are talking about.

$$E(r^N) = E_{bonds} + E_{angles} + E_{dihedrals} + E_{nonbonded} \tag{3.4.7}$$

$$E_{bonds} = \sum_{bonds} K_r(r - r_0)^2 \tag{3.4.8}$$

$$E_{angles} = \sum_{angles} k_\theta(\theta - \theta_0)^2 \tag{3.4.9}$$

$$\begin{aligned} E_{dihedral} = \sum_{dihedrals} \Bigg(& \frac{V_1}{2}\left[1 + \cos(\phi - \phi_1)\right] + \frac{V_2}{2}\left[1 + \cos(2\phi - \phi_2)\right] \\ & + \frac{V_3}{2}\left[1 + \cos(3\phi - \phi_3)\right] + \frac{V_4}{2}\left[1 + \cos(4\phi - \phi_4)\right] \Bigg) \end{aligned} \tag{3.4.10}$$

$$E_{nonbonded} = \sum_{i>j} f_{ij} \left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{C_{ij}}{r_{ij}^{6}} + \frac{q_i q_j e^2}{4\pi\epsilon_0 r_{ij}} \right) \tag{3.4.11}$$

This potential is to be thought of as a model of springs connecting atom to atom with the according force. If the reader researches literature on mechanics of springs, he

or she will learn that the equations for the bonds' energy and the angles' energy are surprisingly similar to the ones of the energy of springs towed between two points. The only exception is coming from electrodynamics with the non-bonded energy describing repulsion forces of the atom core and electrostatic forces being the attracting forces called Van-der-Waals forces. So far there are many approaches to model these molecules but recent developments go into the direction of modelling scoring functions. Scoring functions can be any sets of terms that are modelled in the form of

$$E(x) = \sum_{i=1}^{M} c_i T_i(x) \tag{3.4.12}$$

This would model a function, that has coefficients c_i used to weigh in arbitrary terms T_i and will model a molecule. The approach here is to take a set of known molecular structures, that are able to be found publicly like on the protein data bank [3] and use their experimentally evaluated energies to non-linearly regress a model that fits the experimental data. More recent approaches attempt to do the same with so called machine learning techniques [14]. In this thesis we have successfully established a benchmark, that shows a comparison between a well-known open source package named Autodock Vina, which has been refined to a more stable version called Autodock Smina. The package is also able to take user-defined scoring functions into account. As it was also possible to improve docking results with a scoring function named *Vinardo* [15], we also decided to use that function to run our benchmark. Because the code is open-source, we were able to take advantage of hardware optimizations like the math kernel library of Intel for Intel processors and software optimizations for readily modelled molecules with the openbabel package. We therefore had the opportunity to compare the well studied simulated annealing approach as in Goodsell and Olson [16] to our approach while staying in the absolute same, yet very complex model of protein-ligand docking. The programmatical evaluations of the scoring function were also left the same as they would be used by the simulated annealing optimizer build into Autodock Smina by default.

Quality measures

Especially because computer aided drug design is a very inexact science, one will need appropriate quality measures to ensure the prediction of a new drug. Usually drug-design software packages are tested on experimental data such as the DUD-database. The data provided, includes a set of target proteins and their according ligands as well as decoys, that are not supposed to bind to the protein. We refer to the definitions of these terms as described in [15].

Scoring Power The scoring power is a measure of how close the function, that models the molecular potential, is to the actual experimentally measured energy. For that purpose we can take the correlation coefficient, also referred to as the pearson coefficient,

between calculated experimental and computational data.

$$\rho_{X,Y} = \frac{cov\,(X,Y)}{\sigma_X \sigma_Y} \tag{3.4.13}$$

where $cov\,(X,Y)$ is the covariance matrix between two random variables and σ_X and σ_Y, the standard deviations of the random variables X and Y.

Docking Power The docking power is measured through determining the actual position of the compound, that is placed in the binding pocket of the protein. This measure has many ways to be defined. The key idea here is think of the measure by calculating the root-mean-square-distance of the atoms' coordinate positions the computed coordinates to the experimentally measured coordinates. Lets denote $x_e \in \mathbb{R}^{3N}$ the vector of N-many atoms in three dimensions and $x_c \in \mathbb{R}^{3N}$ the vector of N-many atoms, which were evaluated computationally. The RMSD is defined as

$$RMSD(x_e, x_c) = \frac{1}{N}\sum_{i=1}^{N} ||x_e - x_c||_2 \tag{3.4.14}$$

where $||\cdot||_2$ is the euclidean distance

Ranking Power The ranking power test assesses the ability of a scoring function to correctly rank the known ligands of the same target protein based on their predicted binding affinity given the poses from the experimentally found crystal structures. For each benchmark, there are a total number of target proteins and a specified number of known ligands for each protein. Two levels of success, namely high-level and low-level, are evaluated in datasets as CASF-2013. For the high-level, the three ligands for target protein should be ranked by predicted score as the best $>$ the median $>$ the poorest, while the low-level only needs to pick the best one out of three. The success rate is calculated by the number of the correctly ranked targets among all targets.

Screening Power Given set of compounds and the target, the screening power is the ability to select the true best binder, that can be confirmed in the experiment. For that purpose enhancement factors (EF) are computed.

$$EF_{1\%} = \frac{NTB_{1\%}}{NTB_{total} \times 1\%}$$
$$EF_{5\%} = \frac{NTB_{5\%}}{NTB_{total} \times 5\%}$$
$$EF_{10\%} = \frac{NTB_{10\%}}{NTB_{total} \times 10\%}$$

The numbers $NBT_{1\%}$,$NBT_{5\%}$ and $NBT_{10\%}$ are the numbers of true binders observed among the top 1%, 5% and 10% percent candidates selected by the given scoring function in a list of binding energies sorted from top to bottom by energy value. The number

NBT_{total} is the total number of true binders, that have been mixed to the decoy compounds. As an example let 103 compounds be docked with an optimizer and scoring function. Let 3 of those 103 compounds be experimentally determined true binders, then $NBT_{total} = 3$. After virtual docking computations took place and the number of true binders found in the top 1% of the resulting energy list is $NBT_{1\%} = 2$, the resulting enrichment factor would be $EF_{1\%} = \frac{2}{3\times 1} = 66.67\%$.

Receiver Operating Characteristic (ROC) This measure is applied for binary classifiers. Since we decide between binder and non-binders, we have applied the measure to our evaluations. Among two values, which we compute, are the True Positive Rate

$$TPR = \frac{TP}{TP + FN} \tag{3.4.15}$$

where TP is the number of true positives and FN is the number of false negative classifications and the False Positive Rate

$$FNR = \frac{FP}{TN + FP} \tag{3.4.16}$$

where TN is the number of true negatives and FP is the number of false positive classifications. We plot both values against each other and get the ROC-curve.

Experimental Setup

One of the difficult tasks in docking is the approximation of an objective function, that when globally optimized, will yield atomic configuration results, that match the experimental results. A good optimization function can be established by taking experimental data into account and regress the objective functions' hyper-parameters to that data. Here we have used the parameters of the vinardo function, which has been evaluated and proposed in [15]. Disregarding the objective function, which always can be faulty and only fitting to a certain set of proteins and binding compounds, the overall goal of such a simulated docking experiment is to find a substance that will act as an actual binder. Therefore instances for testing have been put up on the DUD-database [12], that contain a target protein, its according binders (ligands), that are known experimentally to bind and decoys specifically constructed to make up a mixture with the ligands such that optimizer and approximated physical potential function may be tested for its performance. Computations were carried out on Intel Xeon E5-2698 v3 processors with 128GB RAM. One docking computation was held in a single thread for the modified package Autodock Smina and the Autodock Smina package. With a target taken from the DUD-database, namely human glycinamide ribonucleotide synthetase (GART), we were docking 919 compounds, some of which were decoys and others were actual experimentally determined binders or *ligands*. We implement the projected newton method in combination with the Level-Set Bisection algorithm (see algorithm 3). Additionally we execute the algorithm multiple times with different starting configurations for the

compounds. Essentially we are making a multi-start approach combined with the established algorithms from chapter 3. This combination of techniques we refer to as the **G**lobal**N**ewton**M**ethod or short GNM.

3.5 Benchmark

3.5.1 Test Functions

We benchmark the test functions as exhibited in section 3.4.1 and see how the method performs compared to the optimizers used in the Global Optimization toolbox of MATLAB. Since no global minimum can be guaranteed, the results of the evaluations are the mean value of four-hundred executions of according optimizations. The initial values for all methods that need one, are chosen randomly from the domain, that we have specified for each function. We have aimed for user friendly application of the optimizers and tried to make as little modifications on the optimizers' default settings as possible. Without knowing what the global minimum is, the optimizers have the possibility to evaluate their best guess for the global optimum given the initial value. Optimizer fminunc was specified to use the quasi-newton algorithm, since we want to test the optimizers with as little information as the objective function itself. It needs to be mentioned that fminunc does not necessarily implement a global optimizer. The initial population parameter 'InitialPopulationMatrix' for the genetic algorithm and the particle swarm algorithm were ten populations.

	B	G	L	$L13$	R
Projected Newton M.	1610	1262	999	834	982
fminunc	89	32	38	70	33
pattern search	102	190	131	126	215
genetic algorithm	4475	5430	5778	5852	5621
particle swarm	2250	2053	1027	1195	1629
surrogateopt	200	200	200	200	200
Simmulated Anneal.	1737	2142	1809	1769	2071

Table 3.1: Dimensions of the functions are $d = 2$. The Projected Newton Method compared in **number of function evaluations** to the global optimization algorithms of the MATLAB software-package

	B	G	L	$L13$	R
Projected Newton M.	0.6785	34.4523	2.6469	43.5140	11.7170
fminunc	0.0268	61.8956	4.2507	27.0950	17.4763
pattern search	0.2	1.5273	1.4998e-32	1.3498e-31	2.9932e-09
genetic algorithm	6.4429	0.8769	0.0094	0.1598	1.1522
particle swarm	0.5328	0.8842	3.2954e-09	0.0014	0.3198
surrogateopt	0.9668	0.8690	1.4002e-06	0.0229	0
Simmulated Anneal.	0.2892	0.9919	0.0043	0.0110	0.8567

Table 3.2: Dimensions of the functions are $d = 2$. Projected Newton Method compared in **error** to the global optimization algorithms of the MATLAB software-package

Clearly the Projected Newton Method is not among the best performers. In certain cases it is better than the genetic algorithm. In most cases the surrogate optimizer outperforms all the others in this given set of two dimensional functions. Lets observe the performance in higher dimensions.

3.5.2 N-Dimensional Test Functions

An extension to higher dimensions of the mentioned test functions from subsection 3.4.1 is of interest. We want to see if there is an exponential relationship to the dimension of the test functions and the computation time. Since non of the methods, we compare here are deterministic global optimizers, it can be assumed that no such correlation will occure. However we want to emphasize, that this exponential correlation needs to be excluded and has to be at least some polynomial behaviour if we want to look out for an optimizer that can approximate global minima in polynomial time. We list the results here.

	Error			FEvals		
	$G10$	$L10$	$R10$	$G10$	$L10$	$R10$
PNM	0.16079	18.7854	87.3211	3737	4679	4263
Fminunc	0.3774	19.1439	94.023	791	451	183
PS	0	1.4998e-32	0.0000	756	401	756
GA	0.0514	0.02634	7.1059	59700	71960	50380
PA	0.066432	0.68082	13.0339	17240	11090	19300
Surr	0	0.10821	0.0000	500	500	500
SA	1.7127	10.3797	64.9722	11209	8776	10343
	$G20$	$L20$	$R20$	$G20$	$L20$	$R20$
PNM	0.0248	36.254	178.33	6874	7734	7703
Fminunc	4.7476e-12	35.988	179.390	1621	1386	336
PS	0	1.4998e-32	0.0000	2211	801	2211
GA	0.0598	0.2296	24.177	98500	87620	83600
PA	0.0167	6.7009	64.672	22180	18120	18720
Surr	0	0.16171	0.0000	1000	1000	1000
SA	4.0952	57.1941	153.435	25477	29230	22823
	$G40$	$L40$	$R40$	$G40$	$L40$	$R40$
PNM	0.0172	80.043	365.6517	13895	18149	19300
Fminunc	2.0394e-11	83.998	366.4410	2542	3562	652
PS	0	1.4998e-32	1.4998e-32	7221	1601	7221
GA	0.1533	0.8854	0	137840	162720	130280
PA	0.0280	46.0084	80.3702	41140	30120	31130
Surr	0	0.2152	208.8409	2000	2000	2000
SA	9.7802	109.7134	400.9163	57427	58505	46808

Table 3.3: Dimensions of the functions are $d = 10, 20, 40$. Projected Newton Method compared in error to the global optimization algorithms of the MATLAB software-package

We observe that with rising dimension all the optimizers loose their accuracy and speed. Except for the particle swarm algorithm, all errors rise with at least one order of magnitude in at least one of the instances. What is interesting to observe, is that even though PNM is not among the best optimizers, it outperforms the Simulated Annealing algorithm. Among many other tests done on other generic problems, the comparison of performance of PNM and Simulated Annealing has led to the conclusion, that PNM might be a good alternative, where Simulated Annealing was the only choice in the matter.

3.5.3 Protein-Ligand Docking

We now apply the algorithm 3 to a problem known as the Docking Problem. With the use of algorithm 3, we were able to directly test the simulated annealing algorithm of

Autodock Smina against our algorithm. Since the code is open-source, it was possible to work with exactly the same computational overhead of evaluating the objective function. Furthermore we had the opportunity to exploit the advantages of the boost library, which among other packages contains the math kernel library, and therefore were able to work with highly performant code. If a researcher using Autodock Smina or Autodock Vina was to think of improvements of his\her computational methods, the introduced GNM-algorithm should be on his\her list of choices.

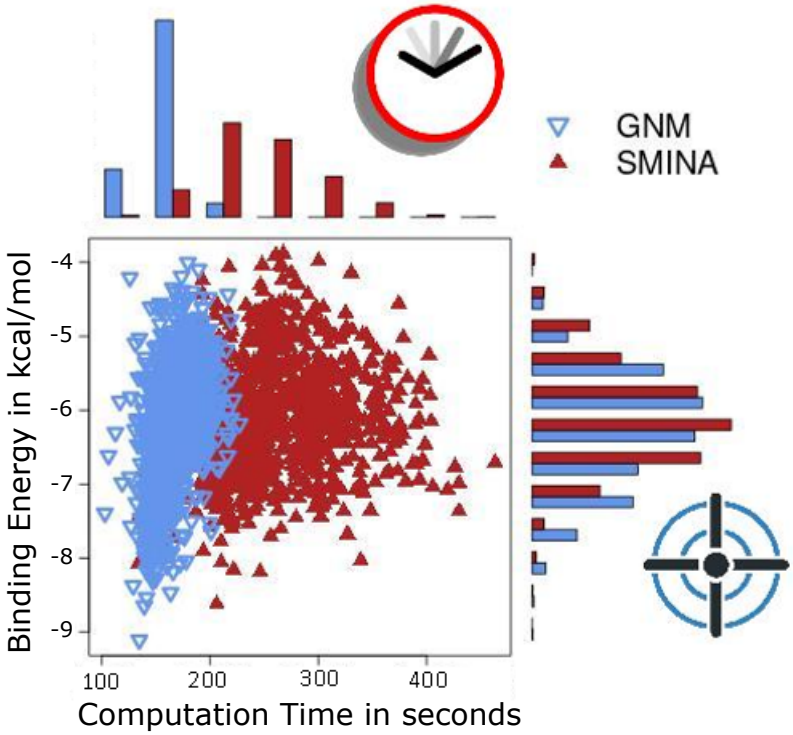


Figure 3.1: Comparison of computation time and calculated energy for human glycinamide ribonucleotide synthetase (PDB: 2QK4, DUD-Database 2006) with Autodock Smina (red) and the stochastic global newton method (blue)

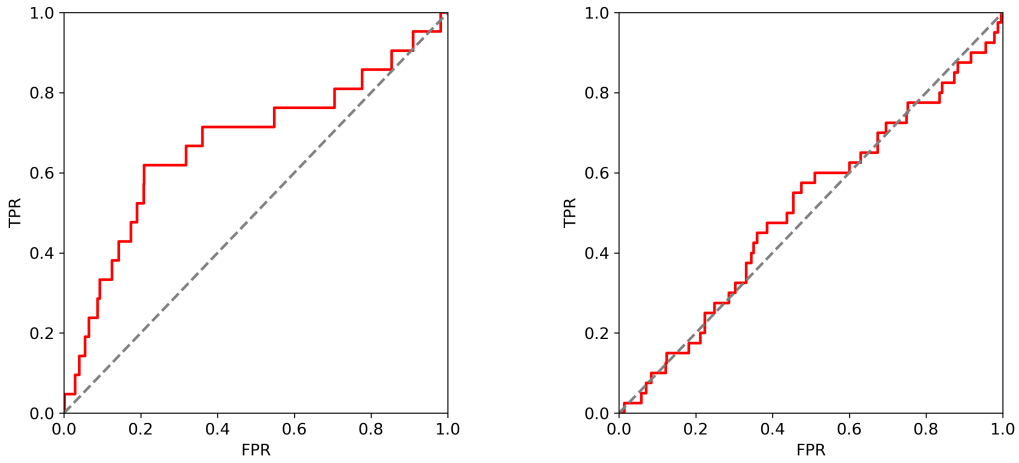


Figure 3.2: ROC curve for docking experiment on human glycinamide ribonucleotide synthetase (GART, PDB: 2QK4, DUD-Database 2006). **Left:** with stochastic PNM, **Right:** with Autodock SMINA

The results shown in figure 3.1 are suggesting, that the performance of the modified Autodock Smina software package (GNM) is superior. As we see in figure 3.2 the GNM algorithm, has a significant improving impact on the ROC curve. Even though we do not have a detailed mathematical analysis for the case of scoring functions as used in the Autodock Smina package, we were able to achieve a result that might motivate further research in GNM. An outlook of what these results imply in the environment of computational drug design, will be given in the last part of the thesis.

4 Dynamic Embedding Algorithm

4.1 Zangwill's Global Convergence Theory

The global convergence theory of Zangwill [7], will be a vehicle to proving global convergence of iterative methods. Here is a systematic deduction into the main result of Zangwill's global convergence theorem.

Definition 4.1.1. Given two sets, X and Y, a *set-valued mapping* defined on X with range in Y is a map Φ, which assigns to each $x \in X$ a subset $\Phi(x) \subset \mathcal{P}(Y)$

To put this definition to use, we define the following.

Definition 4.1.2. Let X be a set and $x \in X$ a given point. Then the *iterative algorithm* $\mathcal{A}$, with initial point x_0 is a set-valued mapping $\mathcal{A} : X \to \mathcal{P}(X)$ which generates a sequence $\{x_n\}_{n=1}^{\infty}$ according to

$$x_{n+1} \in \mathcal{A}(x_n), n = 0, 1, \ldots \tag{4.1.1}$$

We hereby have a general formulation of an algorithm, that may be for instance the steepest descent algorithm. This method for example produces a well-defined sequence given a particular starting point. Our algorithms later in this thesis, have the same character.

Definition 4.1.3. Given $\Gamma \subset X$ and an iterative algorithm $\mathcal{A}$ on X, a continuous real-valued function $Z : X \to \mathbb{R}$ is called a *descent function* provided
1.

$$\text{If } x \notin \Gamma \text{ and } y \in \mathcal{A},\ Z(y) < Z(x) \tag{4.1.2}$$

2.

$$\text{If } x \in \Gamma \text{ and } y \in \mathcal{A},\ Z(y) \leq Z(x) \tag{4.1.3}$$

Definition 4.1.4. A *set-valued mapping* $\Phi : X \to Y$ is said to be closed at $x_0 \in X$ provided
(i)

$$x_k \to x_0 \text{ as } k \to \infty, x_k \in X \tag{4.1.4}$$

(ii)

$$y_k \to y_0 \text{ as } k \to \infty, y_k, y_0 \in Y \tag{4.1.5}$$

implies $y_0 \in \Phi(x_0)$. The map Φ is called closed on $S \subset X$, provided it is closed at each $x \in S$.

One other important concept for supporting a global convergence statement is

Definition 4.1.5. Let $\{\mathcal{A}, \Gamma, Z\}$ be an iterative descent algorithm on a set X. This algorithm is said to be *globally convergent* provided, for any starting point $x_0 \in X$, the sequence generated by $\mathcal{A}$ has x_0 as an accumulation point.

I order to have a term for continuity of set-valued mappings, we define continuity in terms of sequences.

Definition 4.1.6. Given two metric spaces X and Y, a function $f : X \to Y$ is said to be *continuous* on X provided, given $x_0 \in X$ and a sequence $\{x_n\}_{n=1}^{\infty}$ such that $x_n \to x_0$ as $n \to \infty$, then the sequence $\{y_n\}_{n=0}^{\infty} = \{f(x_n)\}_{n=0}^{\infty}$ converges to $y_0 = f(x_0)$.

With the definition of a composite map

Definition 4.1.7. Let $\mathcal{A} : X \to Y$ and $\mathcal{B} : Y \to Z$ be two point set mappings. The composite map $\mathcal{C} = \mathcal{B} \circ \mathcal{A}$ which takes points $x \in X$ to sets $\mathcal{C}(x) \subset Z$ is defined by

$$\mathcal{C}(x) := \bigcup_{y \in \mathcal{A}(x)} \mathcal{B}(x) \tag{4.1.6}$$

the Lemma on composite maps will become useful.

Lemma 4.1.8. *Let $\mathcal{A} : X \to Y$ and $\mathcal{B} : Y \to Z$ be two point set mappings. Suppose*

(i) *$\mathcal{A}$ is closed at x_0*
(ii) *$\mathcal{B}$ is closed on $\mathcal{A}(x_0)$*
(iii) *If $x_k \to x_0$ and $y_k \in \mathcal{A}(x_k)$ then there exists a y such that, for some subsequence $\{y_{k_j}\}$, $y_{k_j} \to y$ as $j \to \infty$*

Then the composite map $\mathcal{C} = \mathcal{B} \circ \mathcal{A}$ is closed at x_0

Proof. The proof can be found in [7] □

Zangwill's main result is the following

Theorem 4.1.9. *Let $\mathcal{A}$ be an algorithm on X, and suppose that, given $x_0 \in X$, the sequence $\{x_k\}_{k=1}^{\infty}$ is generated and satisfies*

$$x_{k+1} \in \mathcal{A}(x_k) \tag{4.1.7}$$

Let a solution set $\Gamma \subset X$ be given, and suppose that

(i) *the sequence $\{x_k\}_{k=0}^{\infty} \subset S$ for $S \subset X$ a compact set*
(ii) *there is a continuous function Z on X such that*
(a) *if $x \notin \Gamma$, then $Z(y) < Z(x)$ for all $y \in \mathcal{A}(x)$*

(b) if $x \in \Gamma$, then $Z(y) \leq Z(x)$ for all $y \in \mathcal{A}(x)$
(iii) the mapping $\mathcal{A}$ is closed at all points $X \setminus \Gamma$

Proof. The proof can be found in [7] □

Then the limit of any convergent subsequence of $\{x_k\}_{k=0}^{\infty}$ is a solution.

In order to apply this theorem, it must only be established, that the definitions of Zangwill's global convergence theory apply to the algorithm constructed in section 4.3. But first some basics for the introduction to some known ideas for it.

4.2 Newton Continuation Method

For the sake of introducing the topic to the reader, we will recite theorems taken from the book [6]. The preliminaries we need in order to conclude convergence results to a global minimum, are the newton continuation methods. These methods are applied to systems of non-linear equations. We need to take a look at the known theory as we will proceed with it in section 4.3 regarding only a special case. This theory works with so called homotopies on functions. What they are? Here is a definition

Definition 4.2.1. A homotopy between two functions F and G from a topological space X to another topological space Y is a continuous map $H : X \times [0,1] \to Y$ such that $H(x,0) = F(x)$ and $H(x,1) = G(x)$.

With our original problem $F(x) = 0$, where $D \in \mathbb{R}^N$ and $F : D \to \mathbb{R}^N$ and another function $G(x_0) = 0$ with $x^0 \in \mathbb{R}^N$ and $G : D \to \mathbb{R}^N$, we will specify the homotopy function $H : D \times [0,1] \to \mathbb{R}$ sometimes also referred to as an *embedding*. At this point we mention three very common homotopy functions namely

- Fixed-Point Homotopy:

$$H(x,\lambda) := (1-\lambda)F(x) + \lambda(x - x_0) \tag{4.2.1}$$

- Convex Homotopy:

$$H(x,\lambda) := (1-\lambda)F(x) + \lambda E(x) \tag{4.2.2}$$

- Newton Homotopy:

$$H(x,\lambda) := F(x) - (1-\lambda)F(x_0) \tag{4.2.3}$$

In this context the variable λ is often referred to as the "homotopy-parameter". Continuation methods aim for the solution of **systems** of non-linear equations. It is the aim of this approach to find the so called homotopy path represented by the set

$$H^{-1}(0) := \{(x,t) \in \mathbb{R}^n \times \mathbb{R} | H(x,t) = 0\} \tag{4.2.4}$$

The existence of this path is established by the implicit function theorem, namely if $(x_0, 1)$ is a regular zero point of H, i.e. the Jacobian $H'(x_0, 1)$ has full rank N, then

a curve $c(\sigma) \in H^{-1}(0)$ with initial value $c(0) = (x_0, 1)$ and tangent $\dot{c}(0) \neq 0$ will exist at least locally i.e. on some open interval around zero. Additionally if zero is a regular value of H or all zero points of H are regular points, then this curve is diffeomorphic to a circle or the real line.

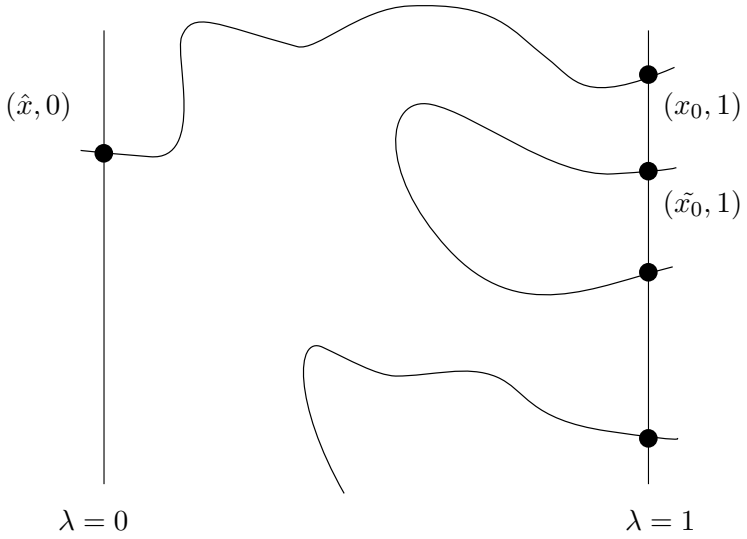


Figure 4.1: Possible paths for curve $c(\sigma)$

We are going to put thought into the question of how to compute such a path numerically and how to become sure, that the path is reaching the zero contour set of $H(x, 0) = F(x)$, as it is illustrated in figure 4.2. The homotopy paths for curve $c(s)$can lead from a point $(\tilde{x_0}, 1)$ forward and back again to another point at $\lambda = 1$ or may lead to infinity. In order to understand why these paths are regarded as parametrized curves, we must understand what it means to solve for the path without the parametrization through the arc-length s. For that purpose, we review the known Embedding Algorithm stated as follows

Algorithm 5: Embedding Algorithm

Data:
$x_0 \in \mathbb{R}^N$ such that $H(x_0, 1) = 0$
integer $m > 0$
Result:
x the solution

```
1 Set x := x_0, λ := (m − 1)/m, Δλ := 1/m
2 for i = 1, ..., m do
3 |   solve H(y, λ) = 0 iteratively for y using x as starting value.
4 |   x := y, λ := λ − Δλ
```

The embedding algorithm builds on the idea, that through increments of the homotopy-parameter λ the solution of $H(y, \lambda) = 0$ will always be found. This idea clearly fails,

when a λ is reached, that passes a turning point of the curve.

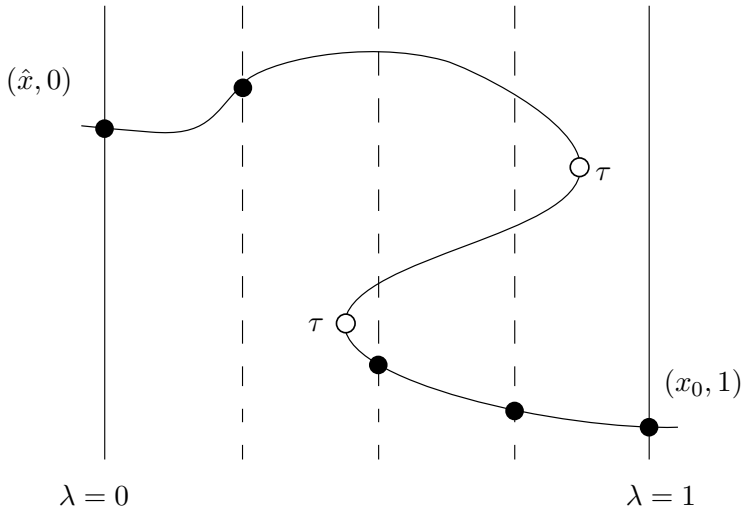


Figure 4.2: The embedding algorithm fails at turning points τ

Evidently we need to solve this problem differently. We decide to parametrize the curve by arc-length parameter s. From now on we regard the problem as an ODE by differentiation of

$$H(c(s)) = 0 \tag{4.2.5}$$

with respect to s such that

$$H'(c)\dot{c} = 0, \quad ||\dot{c}|| = 1, \quad c(0) = (x_0, 1) \tag{4.2.6}$$

Differentiating $H(c(s))$ results in the Davidenko differential equation. The name comes from Davidenko's original paper from 1953, who has been traced to be one of the first people, who formulated the equation in this context. So

$$\frac{dH}{d\sigma} = \frac{\partial H}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial H}{\partial \lambda}\frac{\partial \lambda}{\partial s} = 0 \tag{4.2.7}$$

where s is the arc-length of the homotopy path. For this ODE we state the initial value problem

Problem 4.2.2. The initial value problem for Davidenko's ODE is as follows

(i)

$$\left(\frac{\partial H}{\partial x}\ \frac{\partial H}{\partial \lambda}\right)\begin{pmatrix}\frac{\partial x}{\partial s}\\ \frac{\partial \lambda}{\partial s}\end{pmatrix} = 0 \tag{4.2.8}$$

(ii)

$$\begin{pmatrix}x(0)\\ \lambda(0)\end{pmatrix} = \begin{pmatrix}x^0\\ 0\end{pmatrix} \tag{4.2.9}$$

(iii)

$$||\frac{dx}{ds}||^2+\left(\frac{d\lambda}{ds}\right)^2=1 \tag{4.2.10}$$

One popular approach to solve the initial value problem 4.2.2, is by using a predictor-corrector method. The predictor exerts a superposition of the current iterate by the tangent vector of the curve $c(s)$ on the current iterate. The corrector usually drives the superposed iterate to a point, that is closer to the exact solution. Before we apply the idea the of the Predictor-Corrector method, we make some general statements, on which we pick up in section 4.3. These are important in order to achieve a higher level of mathematical precision in the advancing theory. For a system of non-linear equations $H(x,\lambda)=0$, we make the

Assumption 4.2.3. $H:\mathbb{R}^{N+1}\to\mathbb{R}^N$ is a smooth map i.e. $H\in\mathcal{C}^\infty(\mathbb{R}^{N+1})$.
There is a point $u_0\in\mathbb{R}^{N+1}$ such that:

(i) $H(u_0)=0$
(ii) the Jacobian matrix $H'(u_0)$ has maximum rank i.e. $\text{rank}(H'(u_0))=N$

With assumption 4.2.3, we can choose an index i, such that the sub-matrix of the Jacobian $H'(u_0)$, obtained by deleting column i, is non-singular. It follows from the implicit function theorem, that a local parametrization of the solution set $H^{-1}(0)$ with respect to the i^{th} coordinate exists. We re-parametrize and obtain

Corollary 4.2.4. *Under the assumption 4.2.3, there exists a smooth curve $\alpha\in J\to c(\alpha)\in\mathbb{R}^{N+1}$ for some open interval J containing zero such that for all $\alpha\in J$:*

(i) $c(0)=u_0$
(ii) $H(c(\alpha))=0$
(iii) $rank(H'(c(\alpha)))=N$
(iv) $c'(\alpha)\neq 0$

Differentiating equation (ii) in Corollary 4.2.4 we see that the tangent vector $c'(\alpha)$ satisfies equation

$$H'(c(\alpha))c'(\alpha)=0 \tag{4.2.11}$$

Obviously the tangent $c'(\alpha)$ spans the one-dimensional kernel $\ker(H'(c(\alpha)))$. Equivalently, and this is important to notice, the choice $\dot{c}(\alpha)$ is **orthogonal to all rows of** $H'(c(\alpha))$. Later in section 4.3, this will be the key thought of how to construct such a tangent vector for a smooth map assigning values from $\mathbb{R}^{N+1}$ to $\mathbb{R}$. In this section, we would like to explain the current known theory a little more, in order to be able to relate to it, such that the advancements in the continuation theory in section 4.3 don't seem too strange. So the aim of the current theory is to also add a consistent direction for the traversing of the curve. This yields the idea to augment the Jacobian of H and introduce the $(N+1)\times(N+1)$ **augmented Jacobian** matrix defined to be

$$\begin{pmatrix}H'(c(s))\\ \dot{c}(s)^T\end{pmatrix} \tag{4.2.12}$$

Since the tangent $\dot{c}(s)$ is orthogonal to the N linearly independent rows of the Jacobian $H'(c(s))$, it follows that the augmented Jacobian is non-singular for all $s \in J$. The advantage of defining this augmented Jacobian, is that now we are able compute the determinant of it and define, that the sign has to be constant on J. This approach adopts the conventions of differential geometry. If we apply the definition 4.2.6 of the tangent vector induced by A, we can state the following

Lemma 4.2.5. *Let $c(s)$ be the positively oriented solution curve parametrized with respect to arc-length s which satisfies $c(0) = u_0$ and $H(c(s)) = 0$ for s in some open interval J containing zero. Then for all $s \in J$, the tangent $\dot{c}(s)$ satisfies the following three conditions:*

(i)

$$H'(c(s))\dot{c}(s) = 0$$

(ii)

$$||\dot{c}(s)|| = 1$$

(iii)

$$\det \begin{pmatrix} H'(c(s)) \\ \dot{c}(s)^T \end{pmatrix} > 0$$

In classical methods property (iii) of Lemma 4.2.5 is used to specify the tangent of the curve. That motivates the following definition of the tangent vector t

Definition 4.2.6. Let A be a $N \times (N+1)$-matrix, with $rank(A) = N$. The unique vector $t(A) \in \mathbb{R}^{N+1}$ satisfying the three conditions

(i) $At = 0$
(ii) $||t|| = 1$
(iii) $\det \begin{pmatrix} A \\ t^T \end{pmatrix} > 0$

is called the **tangent vector induced by A**

It can be shown by the Implicit Function Theorem, that the tangent vector $\dot{c}(s)$ induced by A, is also a smooth map.

Lemma 4.2.7. *The set $\mathcal{M}$ of all $N \times (N+1)$- matrices A having maximal rank N is an open subset of $\mathbb{R}^{N\times(N+1)}$, and the map $A \in \mathcal{M} \to t(A)$ is smooth*

Proof. See [6] □

As Lemma 4.2.7 suggests, equation's (4.3.7) right hand side is a smooth map. By the Lemmas and definitions established so far, we are only able to formulate this problem specifically for the system of N-many non-linear equations. We would like to extend that to the general case

Definition 4.2.8. Let $f : \mathbb{R}^p \to \mathbb{R}^q$ be a smooth map. A point $x \in \mathbb{R}^p$ is called a regular point of f if the Jacobian $f'(x)$ has maximal rank $\min\{p, q\}$. A value $y \in \mathbb{R}^q$ is called a regular value of f if x is a regular point of f for all $x \in f^{-1}(y)$. Points and values are called singular if they are not regular.

For further use in the next section, the following Lemma will help us to establish existence of the curve $c(s)$ not only in the classic continuation theory but also for the case of a single non-linear equation.

Lemma 4.2.9. *Let $f : \mathbb{R}^p \to \mathbb{R}^q$ be a smooth map. Then the set*

$$\{x \in \mathbb{R}^p | x \text{ is a regular point of } f\}$$

is open

Proof. Consider the case $p \leq q$. Then x is regular if and only if

$$\det(f'(x)f'(x)^T) \neq 0$$

and the set of such x is open since the map $x \to f'(x)$ is continuous. The case $p < q$ is treated analogously by considering the determinant of $f'(x)^T f'(x)$ □

Now we are able to state the IVP in terms of the tangent vector.

Problem 4.2.10. The initial value problem for the Davidenko ordinary differential equation in terms of the tangent vector induced by Jacobian $H'(u)$, with $u \in \mathcal{C}^1$, is stated as follows

$$\dot{u} = t\left(H'(u)\right) \tag{4.2.13}$$

$$u(0) = u_0 \tag{4.2.14}$$

u is defined to be points such that $H'(u)$ has maximal rank.

The smoothness of the IVP for the single non-linear equation case is therefore given implicitly. We hereby conclude the theory for the classic continuation method and will proceed with new ideas in the next section.

4.3 Single Non-Linear Equation

We remind the reader, that we want to deal with one non-linear equation only and not with a system of equations. It would be possible to treat this equation as an underdetermined system, but this would also include a lot of terms and definitions, we actually don't need. With our original problem $F(x) = 0$, where $D \subset \mathbb{R}^N$ and $F : D \to \mathbb{R}$ and another function $E(x_0) = 0$ with $x^0 \in \mathbb{R}^N$ and $E : D \to \mathbb{R}$, we define the homotopy function $H : D \times [0, 1] \to \mathbb{R}$ sometimes also referred to as an *embedding*. At this point, we work with the Fixed-Point Homotopy:

$$H(x,t) := (1-t)F(x) + t(x - x_0) \tag{4.3.1}$$

As we have introduced the theory for existence and local uniqueness for approximating the homotopy path namely

$$H^{-1}(0) := \{x, t \in \mathbb{R}^n \times \mathbb{R} | H(x,t) = 0\} \tag{4.3.2}$$

The differential equation in this case, looks the same for the under-determined case

$$\frac{dH}{ds} = \frac{\partial H}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial H}{\partial t}\frac{\partial t}{\partial s} = 0 \tag{4.3.3}$$

However if we want to state the initial value problem for that case, we must reconsider our definition of what a tangent vector is. Definition 4.2.8 refers to a matrix inducing the desired tangent vector, we are only left with a vector. This vector should suffice the following

Definition 4.3.1. Let v be a $(N+1)$-vector. The vector $t(v) \in \mathbb{R}^{N+1}$ satisfying the three conditions
1.

$$< v, t > = 0 \tag{4.3.4}$$

2.

$$||t|| = 1 \tag{4.3.5}$$

3.

$$\{t(v)\}_{N+1} > 0 \tag{4.3.6}$$

is called the **tangent vector induced by v**

In the previous section 4.2, we generalized the ODE in problem 4.2.10 to maps $f : \mathbb{R}^p \to \mathbb{R}^q$. So according to Definition 4.2.8 we have maximal rank of the Jacobian of H, specifically $rank \min\{p = N, q = 1\} = 1$ for all regular points of H. Unlike in the previous section 4.2, we don't have a Lemma like 4.2.7 that guarantees for the smoothness of $t(H')$. We therefore assume the following problem to be defined.

Problem 4.3.2. The initial value problem for the Davidenko ordinary differential equation in terms of the tangent vector induced by Jacobian $H'(u)$, with $u \in \mathcal{C}^1$, is stated as follows

$$\dot{u} = t\left(H'(u)\right) \tag{4.3.7}$$

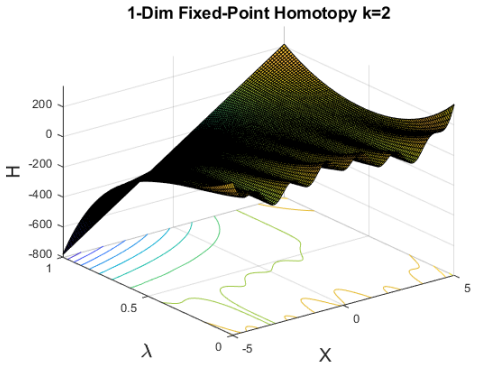
$$u(0) = u_0 \tag{4.3.8}$$

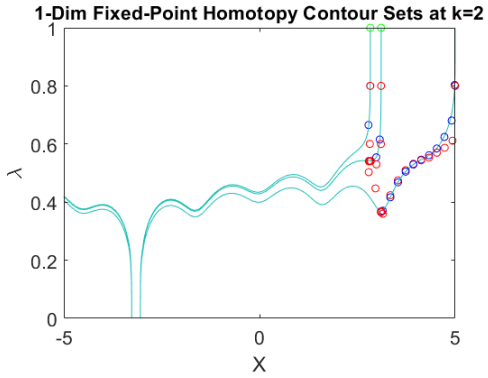
u is defined to be points such that $H'(u)$ has maximal rank 1.

What we have as a preliminary assumption for the smoothness of the right hand side is the Lemma's 4.2.9 implication, that the set of regular points of our single equation H is open. That being said, we proceed to the calculation of the tangent vector. The tangent vector defined in 4.3.1 is used in a PC-Approach (**P**redictor-**C**orrector). The reader might ask why we made the effort to restate the definition of the tangent vector from the general matrix induced tangent vector (definition 4.2.6) to the vector induced tangent vector (definition 4.3.1). In section 4.2 we introduced the idea of a directed tangent vector over the definition with its augmented Jacobian matrix. We however achieve the same thing by condition 4.3.6 in definition 4.3.1. The tangent vector has many degrees of freedom. We specifically choose the orthogonal projection of the gradient of H onto the e_{N+1}-direction, the penalty-parameter dimension. I.e.

$$t = \Pi_{[e_{N+1}]}(\nabla H) = e_{N+1} - \frac{\partial_\lambda \mathcal{H}}{||\nabla \mathcal{H}||^2} \nabla \mathcal{H} \tag{4.3.9}$$

Before we define the algorithm to find roots of the original problem F, we mention once again, that the homotopy applied, is the Fixed-Point Homotopy. Additionally we do not only vary the homotopy function H in the parameters x and λ but also in the initial choice of x_0. All in all this results in a *dynamic homotopy function* $\mathcal{H} : D \times D \times [0,1] \to \mathbb{R}$. Since homotopies on functions are also known as *embeddings*, we choose the algorithm's name to be *Dynamic Embedding Algorithm*. In order to prove global convergence in the context of the new algorithm, we will use Zangwill's global convergence theory. We have revised Zangwill's global convergence theory in section 4.1. Γ is supposed to be the solution set of our problem. In our case this is $\Gamma := \{(x,\lambda) \in D \times [0,1] \,|\partial_\lambda H(x,\lambda) = 0 \text{ and } H(x,\lambda) = 0\}$.





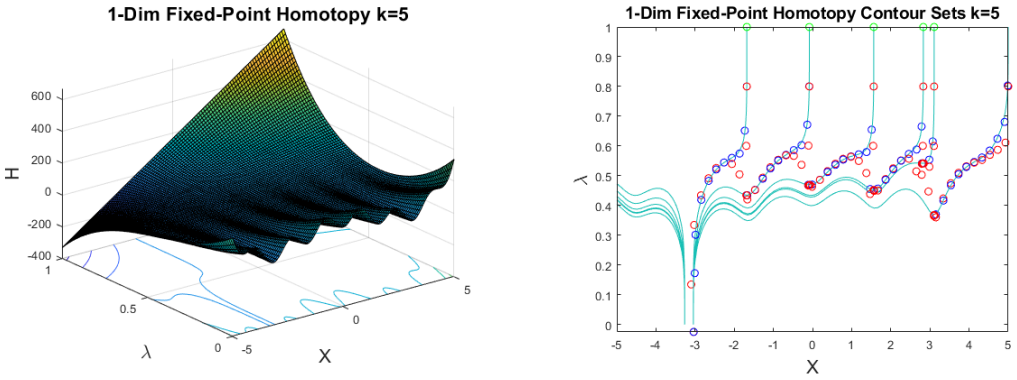


Figure 4.3: Top-Left: The Fixed-Point Homotopy on function $f(x)$. The initial root x_0 has shifted to the two former accumulation points contained in the solution set.**Top-Right:** Predictor (red) and Corrector (blue) steps solve for the curve $c(\alpha)$.**Bottom-Left:** The Fixed-Point Homotopy on function $f(x)$. The initial root x_0 has shifted to the accumulation in the solution set, such that zero-level-set of the homotopy function is monotonous in λ-direction.**Bottom-Right:** Predictor and Corrector steps on the homotopy's zero-level-set. Since the curve solving the Davidenko differential equation is monotonous in λ-direction, the traversing PC-steps reaches $\lambda = 0$.

So if Γ is constructed in that way, we define our *descend function* to be $Z(x, \lambda) := \lambda$, $\forall x \in D$. What we aim for with this definition is to construct something like a gradient descent on the zero-level-set of the homotopy function. We visualize the idea in figure 4.4 in one dimension first, regarding the Fixed-Point Homotopy on $f(x) = (x+4.5)(x+1)(x-1)(x-4)+100\sin(2x)^2$. One can think of the method, as a gradient descent on the zero-level-set of the homotopy function. Once a "local minimum" is found, the dynamic initial root of the homotopy function is shifted in the x-dimension to match the "local minimum's" x-position. In such a way iterates are able to jump over local minima.

A visualization in two dimensions for the function

$$f(x_1, x_2) = (x_1 + 4.5)(x_1 + 1)(x_1 - 1)(x_1 - 4) + 30\sin(2x_1)^2$$
$$+ (x_2 + 4.5)(x_2 + 1)(x_2 - 1)(x_2 - 4) + 30\sin(2x_2)^2$$

this looks as follows.

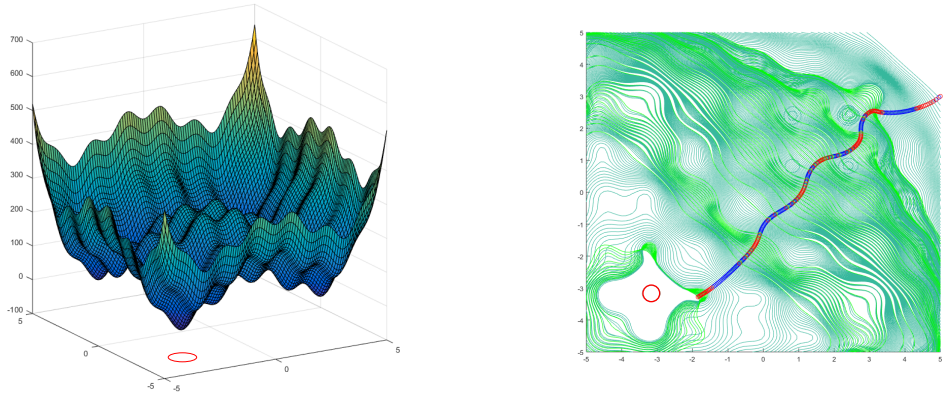


Figure 4.4: Left: Graph of non-convex coercive function $f(x_1, x_2)$. **Right:** Predictor (red) and Corrector (blue) steps solve for the curve $c(\alpha)$. Contour lines mark the zeros level-set of the homotopy function at the current λ. Blue contour lines mark the zero-level-sets of the PC-steps of the inner while loop, while green contour lines mark the zero-level-sets of the PC-steps, when the initial fixed-point x_0 is shifted. In this version of the algorithm the Excitation step in line 4 of the algorithm was made by gradual increase, i.e. $\lambda = \lambda + \Delta\lambda$, instead of setting it to $\lambda = 1$. The approach to the zero-level-set of the function f (red circle in lower left corner) is visible.

The algorithm is an inaccurate root finder, however does not suffer from the cutbacks of not knowing when to stop, as it is the case with the projected newton method in section 3. Certain instances of problems allow us to call the algorithm a root finder for certain classes of functions. So far no analytical result concerning what this class of functions might be, was established. The only thing we can do from here is to prove global convergence of the inner while loop of algorithm 6.

Algorithm 6: Dynamic Embedding Algorithm

Data:

· initial known fixed-point x_0 and initial point $x^{(0)} \in D$, s.t. $H(x^{(0)}, x_0, 1) = 0$

· predictor step length $0 < h \leq 1$ and tolerance $\epsilon > 0$

Result:

$x^* \in \Gamma \subset D$ such that $H(x^*, x_0, 0) = 0$ or there is no solution, i.e. $\Gamma = \emptyset$

Set $x := x^{(0)}$, $\lambda := 1$

while $\lambda > 0$ **do**

  if $\hat{\lambda} - \lambda < \epsilon$ **then**

    Excitation: $\lambda = 1$

    Compute new x_0 for the excited λ. Use fixed x.

$$x_0 = \underset{y \in D}{\arg \operatorname{zero}} \mathcal{H}(x, y, \lambda) \tag{4.3.10}$$

  else

    (Set-valued map $\mathcal{A}$):

    Set: $\hat{h} = h$

    Compute tangent vector w.r.t. x and λ. Use fixed x_0.

$$t = e_{N+1} - \frac{\mathcal{H}_\lambda}{||\nabla \mathcal{H}||^2} \nabla \mathcal{H} \tag{4.3.11}$$

    while $\hat{\lambda} > \lambda$ **do**

      Predictor step: (Set-valued map S_2)

$$\begin{pmatrix} \tilde{x} \\ \tilde{\lambda} \end{pmatrix} = \begin{pmatrix} x \\ \lambda \end{pmatrix} - \hat{h} \cdot t \tag{4.3.12}$$

      Corrector step: (Set-valued map S_1)

$$\begin{pmatrix} x \\ \lambda \end{pmatrix} = \begin{pmatrix} \tilde{x} \\ \tilde{\lambda} \end{pmatrix} - \frac{H(\tilde{x}, x_0, \tilde{\lambda})}{||\nabla H(\tilde{x}, x_0, \tilde{\lambda})||^2} \nabla H(\tilde{x}, x_0, \tilde{\lambda}) \tag{4.3.13}$$

      Set: $\hat{h} = \frac{\hat{h}}{2}$

    Set: $\hat{\lambda} = \lambda$

  if $(x, \lambda) \notin D$ **then**

    return $x^* = \emptyset$

  if $H(x, x_0, 0) = 0$ **then**

    return $x^* = x, x^* \in \mathcal{N}_0(H(x, x_0, 0))$

We open the following analysis with Zangwill's global convergence theory.

4.3.1 Global Convergence

We need to empathize, that most of the proving we do concerning global convergence in the context of zangwill's theory, has been done in his original paper [7]. Especially the proof the final global convergence theorem 4.1.9 is mainly held in its original general form. So in order to build up a global convergence proof, we need to put the projected newton method in conformity with the terms from Zangwill's convergence theory, since it is implicitly used in algorithm 6. At first we propose, that the projected newton method is a closed set-valued map, which satisfies the definition 4.1.4. We thereby state the

Lemma 4.3.3. *Let f have the properties stated in theorem 3.3.6. Furthermore let $D \subset \mathbb{R}^N$, D compact, $N \in \mathbb{N}$ and $B(x,\rho) \subset D \times [0,1]$ be compact sphere with radius $\rho > 0$. Then the iterative algorithm $S_1 : B(x,\rho) \to D$, defined as*

$$S_1(x) := \left\{ y \in D | y = x + d \text{ and } f(x) + \nabla f(x)^T d = 0 \text{ at which } d \neq 0 \right\} \tag{4.3.14}$$

is closed at any point $x \in B(x,\rho)$

Proof. We know by theorem 3.3.6 that there is a sequences $\{x_k\}_{k=1}^{\infty}$ such that $x_k \to x^*$ as $k \to \infty$ where $d \neq 0$. By definition of the projected newton method the iterates $\{y_k\}_{k=1}^{\infty}$ are defined to be $y_k \in S_1(x_k)$ for all k and that $y_k \to y^*$ as $k \to \infty$. We now need to show that $y^* \in S_1(x^*)$.
Since we know that d_k within the projected newton method holds

$$d_k = -\frac{f(x_k)}{||\nabla f(x_k)||^2} \nabla f(x_k) \tag{4.3.15}$$

$$k \to \infty : \qquad d^* = -\frac{f(x^*)}{||\nabla f(x^*)||^2} \nabla f(x^*) \tag{4.3.16}$$

We can see that $y^* = x^* + d^*$ solves the projected newton problem

$$f(x^*) + \nabla f(x^*)^T d = f(x^*) - \nabla f(x^*)^T \frac{f(x^*)}{||\nabla f(x^*)||^2} \nabla f(x^*) \tag{4.3.17}$$

$$= f(x^*) - f(x^*) \tag{4.3.18}$$

$$= 0 \tag{4.3.19}$$

which implies $y^* \in S_1(x^*)$ □

Even though we might look at the predictor step as something we could have put right into one set-valued mapping, we want to keep it organized and separate them. As we will see right after this, it comes in handy to prove

Lemma 4.3.4. *Let f have the same properties stated in theorem 3.3.6. Furthermore $S_1 : B(x,\rho) \to D \subset \mathbb{R}^N$ as defined in Lemma 4.3.3 and $S_2 : D \to Z$ be two point set mappings. Then S_2 defined*

$$S_2(x) := \left\{ y \in D | y = x + d \text{ and } \nabla f(x)^T d = 0 \text{ at which } d \neq 0 \right\} \tag{4.3.20}$$

is closed on $S_1(x_0)$

Proof. So similar as before we assume that $\{x_k\}_{k=1}^{\infty}$ exist such that $x_k \to x^*$ as $k \to \infty$ where $d \neq 0$. Since the prediction step is a single computation of an orthogonal projection, we can simplify the sequence $\{x_k\}_{k=1}^{\infty}$ to $\{x_1\} = \{x_0 + d\}$. We thereby see

$$\begin{aligned} x_1 &= x_0 + d \\ &= x^* \end{aligned}$$

All we need to show is then $y^* \in S_1(x^*) = \{x_0 + d\}$. If d is chosen to be an orthogonal projection of the unit vector e_N (the λ-direction of the homotopy function) on the vector $\nabla f(x^*)$, it is orthogonal to $\nabla f(x^*)^T$ by definition

$$\nabla f(x^*) d^* =< \nabla f(x^*), \Pi_{[e_N]}(\nabla f(x^*)) > \tag{4.3.21}$$

$$=< \nabla f(x^*), e_N - \frac{\partial_N f(x^*)}{||\nabla f(x^*)||^2} \nabla f(x^*) >= 0 \tag{4.3.22}$$

Thereby $y^* \in S_2(x^*)$ hold true. □

The continuation method is composed of two algorithms that depend on each others results. We therefore speak of a composite map in the sense of definition 4.1.7. In order to prove the composite map to be closed we need to state following result of Zangwill's theory

Remark 4.3.5. One might have noticed that if Y is compact, property (iii) in Lemma 4.1.8 hold true immediately. So if $\mathcal{A}$ is closed at x_0 and $\mathcal{B}$ is closed on $\mathcal{A}(x_0)$, the composite map $\mathcal{C} = \mathcal{B} \circ \mathcal{A}$ is closed at x_0.

We put together Lemmas 4.3.4 and 4.3.4 and therefore are able to use Lemma 4.1.8. This yields that $A = S_1 \circ S_2$ is closed on $D \times [0,1] \setminus \Gamma$. Since we have excluded directions $d = 0$ in both S_1 and S_2, this implied that the solution set is excluded. As this is not very obvious, we want to make the reader notice

$$f(x) \neq 0 \Leftrightarrow d_1 = -\frac{f(x)}{||\nabla f(x)||^2} \nabla f(x) \neq 0 \qquad \forall x \in D \setminus \Gamma$$

$$\partial_N f(x) \neq 0 \Leftrightarrow d_2 = -\frac{\partial_N f(x)}{||\nabla f(x)||^2} \nabla f(x) \neq 0 \qquad \forall x \in D \setminus \Gamma$$

Where d_1 would be the search direction of S_1 and d_2 the search direction of S_2. This concludes the third assumption for global convergence of theorem. 4.1.9.We are left to show that the descent function $Z(x,\lambda) = \lambda$ on $D \times [0,1]$ fits the properties we are looking for in theorem 4.1.9. Since it seems clear that the while-loop in algorithm 6 only allows for descending λ, we assume the second assumption of theorem 4.1.9 to be true. The application of theorem 4.1.9 is now imminent. We need to note that the dynamic embedding algorithm consist of the set-valued map $\mathcal{A} = S_1 \circ S_2$ and the computation of an initial value, see equation (4.3.10). We can assume that such an initial value will be chosen in a way that the assumption $x_0 \in S \subset X$ always holds. What is left to prove is the global convergence of the iterative algorithm A. We now formulate the proof in the formalism as it was introduced in section 4.1 such that it fits the language of the dynamic embedding algorithm.

Theorem 4.3.6. *Let $D \subset \mathbb{R}^N$ be a compact set, $\mathcal{A}$ be an iterative algorithm on $D \times [0,1]$ and the continuously differentiable map $H : D \times [0,1] \to \mathbb{R}$. Suppose that, given $(x_0, \lambda_k) \in D \times [0,1]$, the sequence $\{x_k, \lambda_k\}_{k=1}^{\infty}$ is generated and satisfies*

$$(x_{k+1}, \lambda_{k+1}) \in \mathcal{A}(x_k, \lambda_k) \tag{4.3.23}$$

Let the solution set

$$\Gamma := \left\{ (x,\lambda) \in D \times [0,1] \,\middle|\, \frac{\partial H(x,\lambda)}{\partial \lambda} = 0 \text{ and } H(x,\lambda) = 0 \right\} \tag{4.3.24}$$

be given and suppose that

(i) the sequence $\{x_k, \lambda_k\}_{k=0}^{\infty} \subset S$ for $S \subset D \times [0,1]$ is compact

(ii) there is a continuous function $Z(x,\lambda) := \lambda$ on $D \times [0,1]$ such that

(a) if $(x,\lambda) \notin \Gamma$, then $Z(y,\mu) < Z(x,\lambda)$ for all $(y,\mu) \in \mathcal{A}(x,\lambda)$.

(b) if $(x,\lambda) \in \Gamma$, then $Z(y,\mu) \leq Z(x,\lambda)$ for all $(y,\mu) \in \mathcal{A}(x,\lambda)$.

(iii) the mapping $\mathcal{A}$ is closed at all points of $D \times [0,1] \setminus \Gamma$

Then the limit of any convergent subsequence of $\{x_k, \lambda_k\}_{k=0}^{\infty}$ is in Γ.

Proof. The proof in a more general form can be found in [7]. Here we only made slight modifications.
Suppose that (x^*, λ^*) is a limit point of the sequence $\{x_{k_j}, \lambda_{k_j}\}_{j=0}^{\infty}$ such that $(x_{k_j}, \lambda_{k_j}) \to (x^*, \lambda^*)$ as $j \to \infty$. Since the descent function Z is continuous, we have $Z(x_{k_j}, \lambda_{k_j}) \to Z(x^*, \lambda^*)$ when $j \to \infty$. We will show that indeed $Z(x_k, \lambda_k) \to Z(x^*, \lambda^*)$ as $k \to \infty$. Because λ_k is monotonically decreasing on the sequence $\{x_k, \lambda_k\}_{j=0}^{\infty}$ as follows from the property that $(x_{k+1}, \lambda_{k+1}) \in \mathcal{A}(x_k, \lambda_k)$ and from the properties in (ii). We therefore must have $Z(x_k, \lambda_k) - Z(x^*, \lambda^*) \geq 0$ for all k. So given $\epsilon > 0$, there is a j_0 such that, for $j \geq j_0$, we have

$$Z(x_{k_j}, \lambda_{k+1}) - Z(x^*, \lambda^*) < \epsilon, \quad \forall j \geq j_0 \tag{4.3.25}$$

Which includes all $k \geq j_0$ such that

$$Z(x_k, \lambda_k) - Z(x^*, \lambda^*) = Z(x_k, \lambda_k) - Z(x_{k_{j_0}}, \lambda_{k_{j_0}}) + Z(x_{k_{j_0}}, \lambda_{k_{j_0}}) - Z(x^*, \lambda^*) < \epsilon \quad (4.3.26)$$

as $k \to \infty$ it follows $Z(x_k, \lambda_k) \to Z(x^*, \lambda^*)$. It is left to show that (x^*, λ^*) is a solution. We prove this by contradiction. Suppose that (x^*, λ^*) is not a solution. We consider the sequence $\left\{x_{k_j+1}, \lambda_{k_j+1}\right\}_{k=0}^{\infty}$ which has the property that, for each j, $(x_{k_j+1}, \lambda_{k_j+1}) \in \mathcal{A}(x^*, \lambda^*)$. This new sequence lies in the compact set S and hence contains a convergent subsequence $(x_{(k_j+1)_l}, \lambda_{(k_j+1)_l}) \to \hat{x}$ as $l \to \infty$. Since $\mathcal{A}$ is closed on $D \times [0,1] \setminus \Gamma$ and, by assumption $(x^*, \lambda^*) \in \Gamma$, we see that

$$(\hat{x}, \hat{\lambda}) \in \mathcal{A}(x^*, \lambda^*) \quad (4.3.27)$$

On the other hand if the original sequence is regarded it follows $Z(x_k, \lambda_k) \to Z(x^*, \lambda^*)$ implies that we must have $Z(\hat{x}, \hat{\lambda}) = Z(x^*, \lambda^*)$ and this contradicts property (ii) (a) of the assumption. □

We can now say that every initial root for the fixed-point homotopy will yield a convergent prediction and correction iteration in the inner while loop of the algorithm. Obviously this does not prove the global convergence to a global minimum. For this further experiments with the optimizer on various functions have to be made. Some of them we will exhibit in the next section. In order to proceed to further mathematical analysis of the algorithm, some more experience with the algorithms behaviour will need to be gathered. We will discuss further in the outlook of this thesis and come to the use cases we have established so far.

4.4 Applications

4.4.1 N-Dimensional Test Functions

We exhibit test functions, that can easily extend to high number of variables. The main goal here is to achieve a proof of concept for the established algorithm. As the evaluations of the test functions in section 3.5.1 have been done for certain instances of the dimension N, only the Griewank and the Rastrigin function will be of interest here. Table 4.1 contains the the errors and function evaluations when applying dynamic embedding as a root finder combined with the Level-Set Bisection algorithm. In the following we denote $\hat{f}$ to be the computed global minimum value and f^* the exact minimum value. After several experiments, it turns out that the accuracy of computed minima depends strongly on the step size parameter h, the parameter that determines the predictor step size or in other words the length of the tangent vector $t(H')$. If we choose h to be small, the resulting errors become smaller. However, the number of function evaluations rises with linear correlation too. We might demand the necessity for the step size parameter to be small enough, such that the local convergence properties of the implicitly iterated projected newton method are exploited.

	Error		FEvals	
Dimensions	*GN*	*RN*	*GN*	*RN*
2	0.0025	-0.3120	128284	95921
4	0.0033	-0.7009	220640	376754
6	0.0088	-0.6495	349938	459531
8	0.0346	-0.6293	435900	509630
10	0.0333	-0.6577	599681	604709
20	-0.0101	0.1105	1970788	1127664
40	-0.0202	7.5001	5424714	2363372
60	-0.0225	-0.6399	9261675	6739238
80	-0.0263	0.0124	14549670	6565356
100	-8.5229	-0.7044	13037734	14262802

Table 4.1: Projected Newton Method compared in error to the global optimization algorithms of the MATLAB software-package

On the other hand, table 4.1 shows, that in dimension $N = 100$ the error suddenly rises by two orders of magnitude. We suspect that the accuracy of the computed global minimum must also depend on the step size. Both figures 4.5 and 4.6 show unwanted error sizes in the varying dimensions. Especially in figure 4.5, we see that there must be a relationship between error and step size parameter h.

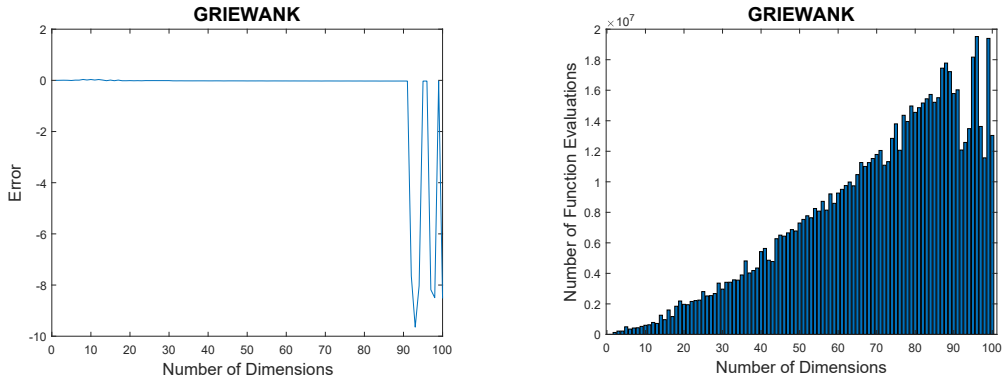


Figure 4.5: Left: Error measured with $(\hat{f} - f^*)$. Negative errors are possible due to precision of the optimizer. **Right:** Number of function evaluations quadratically correlates to the number of dimensions.

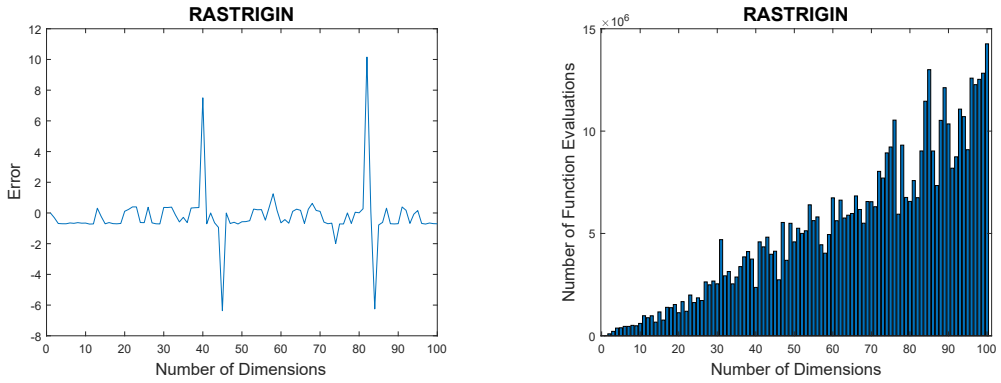
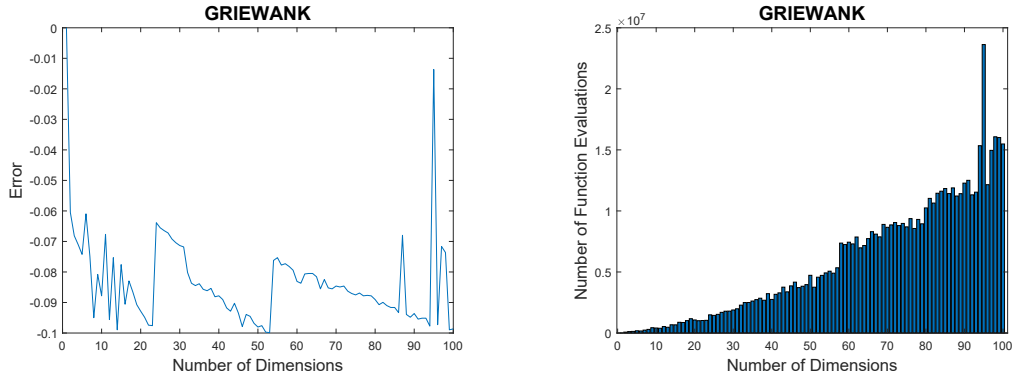


Figure 4.6: Left: Error measured with $(\hat{f} - f^*)$. Negative errors are possible due to precision of the optimizer. **Right:** Number of function evaluations linearly correlates to the number of dimensions.

An advanced experiment shows that the dimensionality of both problems does not correlate exponentially with the step size.



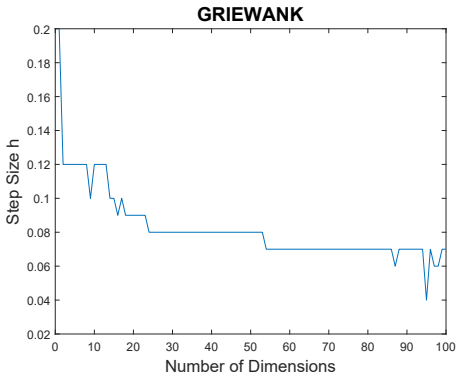


Figure 4.7: Top-Left: Error measured with $(\hat{f} - f^*)$. Negative errors are possible due to precision of the optimizer. **Top-Right:** Number of function evaluations correlates to the number of dimensions. The correlation is not exponential.**Bottom:** the step size h chosen, such that the error is below 0.1.

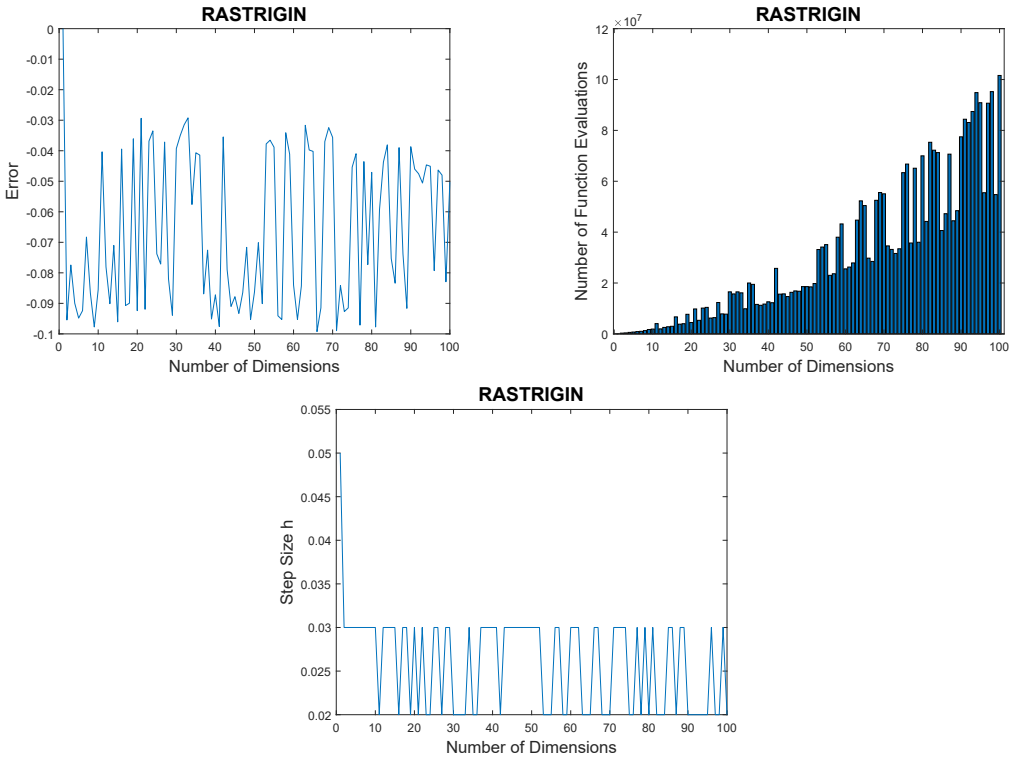


Figure 4.8: Left: Graph of function $f(x_1, x_2)$. **Right:** For certain initial values the algorithm passes through many local minima caused by the sin-terms.

If we compute the step size required to reduce the error to < 0.1, as the visualisations suggest, the step size parameter h converges to a number bigger zero at an asymptotic

rate. The computational complexity of the dynamic embedding algorithm in application on these test functions can therefore be expected to be non-exponential and less then exponential in runtime with increasing dimension. As of now, it goes without saying, that the investigations concerning runtime of this algorithm in the general case are not to be considered closed. These evaluations had the sole purpose of motivating further research towards the convergence properties for quadratic programs with non-linear constraints, since these are of a similar form.

5 Conclusion

5.1 The Fundamental Question

The problem of the decidability or the problem of an explicit decision of how to evaluate if the problem has an empty set of sub-optimal points, was motivated and is able to be solved for certain instances of non-linear problems. We have looked into transformations of np-hard and np-complete problems and found that these problems can be transformed into Homotopies on merit-functions of multivariate polynomials, penalized by continuous integer constraints. These transformations resulted in polynomials of at least 4-th order. In section 4.4.1 we evaluated however only quadratic functions with a sin-and cos-perturbations. We have managed to formulate algorithm 6 and can rightfully argument, that the algorithm always converges. As for now we can suspect the algorithm to solve for zero-level-sets and their non-existence, only for quadratic functions with a sin-and cos-perturbations. If we want to answer the question of thesis we need to prove

> If Algorithm 6, Dynamic Embedding, uniquely decides on the emptyness of the set of sub-optimal points (def. 1.4.2) for multivariate polynomials on $\mathbb{R}^N$, the satisfiability problem after arithmetization becomes decidable.

5.2 The Projected Newton Method

We have constructed the projected newton method and applied it to numerical test functions. We derived from the conclusions of the results, that it might be a good idea to apply the Level-Set Bisection algorithm combined with the projected newton method to a very interesting real world application, namely Protein-Ligand Docking. On the way to solving this problem, such that it might get into competition with a very known technique, called Simulated Annealing, we got to the point of using stochastically generated initial values for the global optimizer. That optimizer we called GNM-algorithm. Unluckily, the findings of chapter 3 lead to the following

> There is no global convergence to a global minimum of non-convex coercive functions with the projected newton method and Level-Set Bisection.

However, we seem to perform much better, than the widely used Simulated Annealing algorithm for applications as Docking. As Simulated Annealing is also applied in many

other applications, we strongly suspect, that other real-world problems can be successfully solved.

6 Outlook

6.1 Protein-Ligand Docking

The performance results of section 3.5.3 open many questions of how the application of the GNM algorithm might perform on larger datasets. So far we only have evaluated one single protein. A new update of the celebrated CASF dataset has come out as the most recent successor of the CASF-2007 and CASF-2013 datasets. The new CASF-2016 dataset has not yet been tested for all major docking packages, however there exist many for the two former ones. If we take a look at figure 6.1 and 6.2, taken from Wang and Zhang [17], all introduced measures from the subsection on docking quality measures 3.4.2 have been benchmarked for the CASF-2013 dataset. All prominent scoring functions have been evaluated as well as the Autodock Vina scoring function.

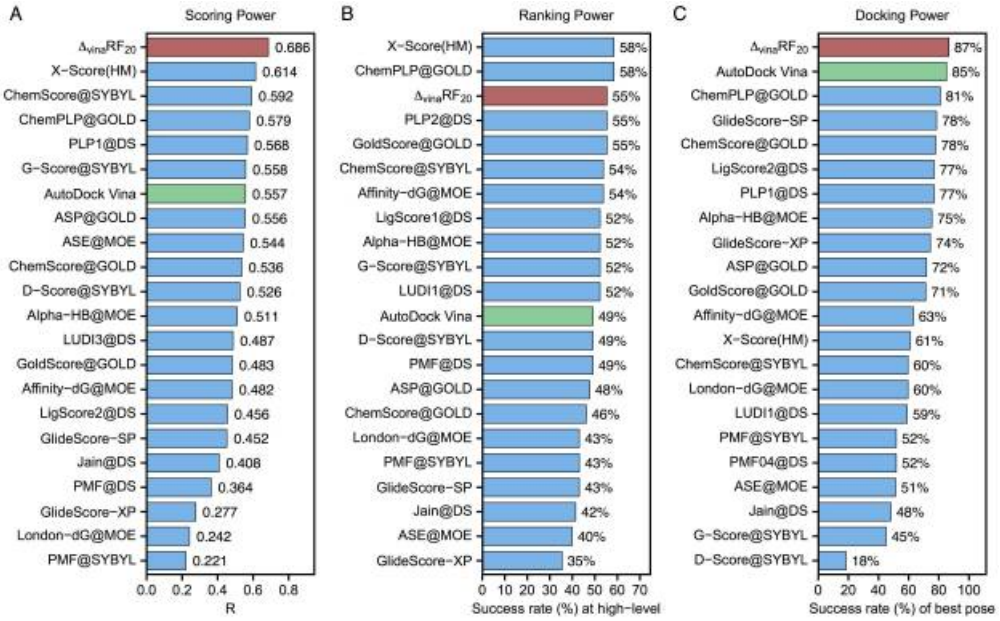


Figure 6.1: Performance of 22 scoring functions in (A) scoring power measured by Pearson's R, (B) ranking power in terms of high-level success rate and (C) docking power measured by the success rate, when the best-scored pose is considered to match the native pose in CASF-2013 benchmark. $\Delta_{vina}RF_{20}$ is colored in red and AutoDock Vina is colored in green. All results colored in blue are obtained from reference[18]

Wang and Zhang introduce a method to specifically use the Autodock Vina software

package and apply a machine learning approach by the name of *random forest*. This powerful boosting technique is applied to work out a scoring function, which seems to improve the underlying Autodock Vina function by far. The authors also visualize a benchmark on the CASF-2007 dataset where the applied boosted function is out performing the underlying function and a lot of the competitors. Since our modified software package is a more stable version on Autodock Vina, which allows for specifying different scoring functions, we wonder how the boosting approach might improve the benchmark in combination of our optimizer.

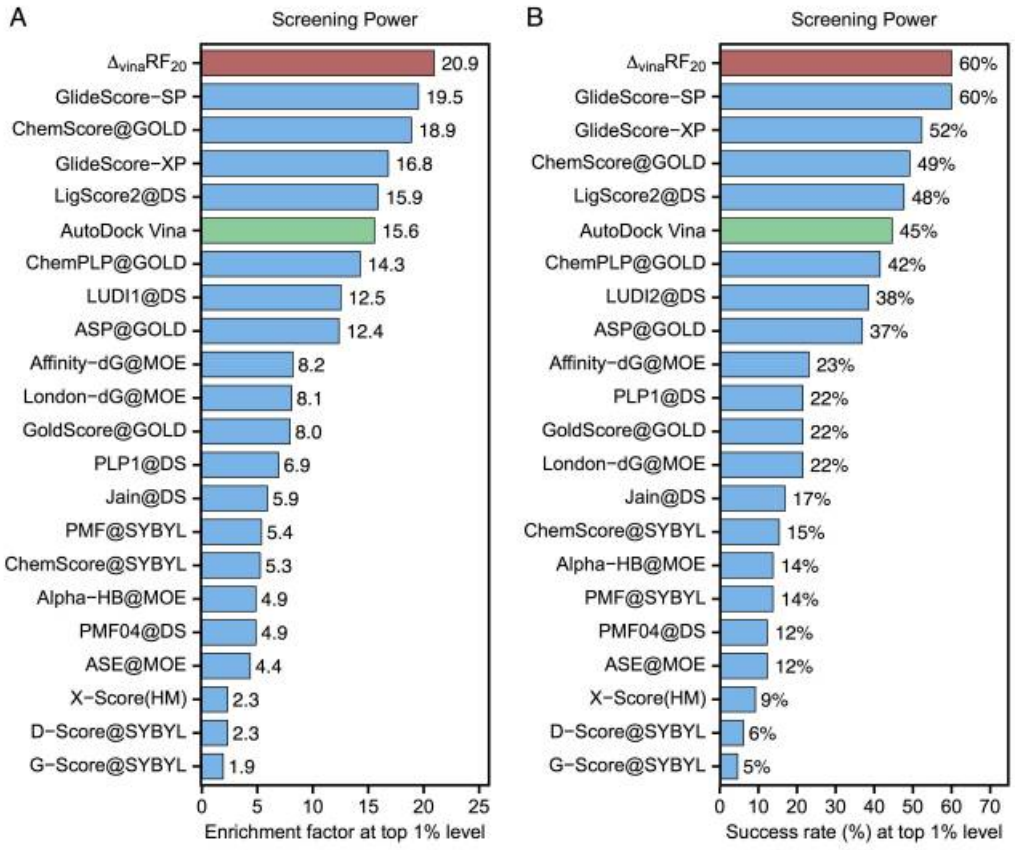


Figure 6.2: Performance of 22 scoring functions in screening power measured by (A) enrichment factor and (B) success rate at top 1% level in CASF-2013 benchmark. $\Delta_{vina}RF_{20}$ is colored in red and AutoDock Vina is colored in green. All results colored in blue are obtained from reference[18]

We suggest to evaluate the suggested optimizer from section 3 in combination with the random forest approach of Wang and Zhang.

6.2 Dynamic Embedding Algorithm

So far we have only proven global convergence of the inner while loop of algorithm 5. It may seem intuitive to go ahead and approach the global convergence proof not only to any accumulation point, but also to a point that is an element of the level-set of the original function. The examples we evaluated in section 4.4.1 were of a form that work well in the current formulation of the algorithm using the fixed-point homotopy. For every dynamic initial root $x_0^{(0)}$ of the dynamic homotopy function, i.e. $H(x, x_0^{(0)}, 0) = 0$, we have proven global convergence to an accumulation point in the solution set (4.3.24). Note however, that this solution set does not necessarily include the the solution set

$$\hat{\Gamma} := \{x \in D | H(x, 0) = 0\} = \mathcal{N}_0(f)$$

The behaviour of the solver in algorithm 6 is yet a little puzzling. On the one hand it is able to overcome local minima but on the other hand fails to hover over to the actual zero-level-set of the original problem. As figure 6.3 shows for the problem function

$$\begin{aligned} f(x_1, x_2) = &(x_1 + 4.5)(x_1 - 4) + 30\sin(2x_1)^2 \\ &+ (x_2 + 4.5)(x_2 + 1)(x_2 - 1)(x_2 - 4) + 30\sin(2x_2)^2 \end{aligned}$$

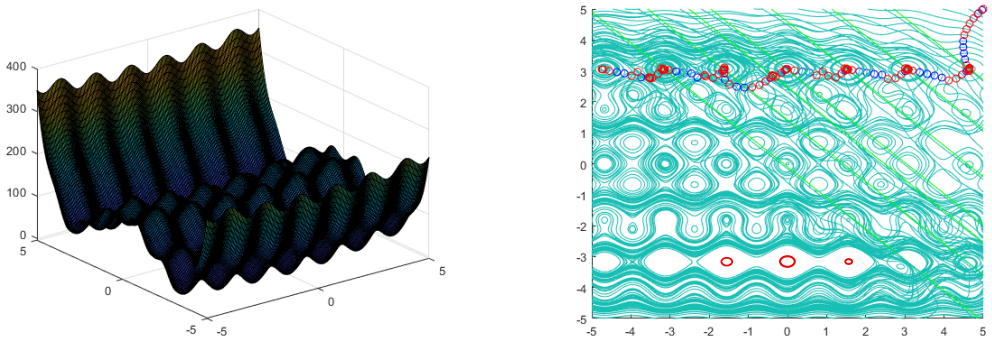


Figure 6.3: Left: Graph of function $f(x_1, x_2)$. The function is a quadratic function in x_1-direction and a polynome of 4-th order in x_2-direction **Right:** The PC-steps are converging to local minima of the contour set w.r.t to the homotopy-parameter, but fail to leave the a bigger surrounding local minimum resulting from the structure of the 4-th order polynome in x_2-direction

Further experimenting with homotopies and most likely dynamic homotopies is necessary to gain a better picture of the behaviour of the algorithm in various problems. Regarding the solution set that comes with the dynamic homotopy function, lets define it as

$$\Gamma^k := \left\{ \left(x, x_0^{(k)}, \lambda\right) \in D \times D \times [0,1] \,\middle|\, \frac{\partial H\left(x, x_0^{(k)}, \lambda\right)}{\partial \lambda} = 0 \text{ and } H\left(x, x_0^{(k)}, \lambda\right) = 0 \right\}$$

We would expect to find an algorithm that defines the series of solution sets, such that

$$\Gamma^k \overset{k\to\infty}{=} \Gamma^* \subset \mathcal{N}_0(f)$$

But this is for later.

Bibliography

[1] Cook, S. A. , The complexity of theorem proving procedures, Conference Record of Third ACM Symposium on Theory of Computing, 1971, pp. 151-158

[2] Shani, S. (1974), Computationally Related Problems, SIAM J. Comput. 3, 262-279

[3] https://www.rcsb.org/

[4] http://www.sfu.ca/~ssurjano/optimization.html

[5] Jones, D.R., Perttunen, C.D. Stuckman, B.E., Lipschitzian optimization without the Lipschitz constant, J Optim Theory Appl (1993) 79: 157. https://doi.org/10.1007/BF00941892

[6] E.L. Allgower, K. Georg, Numerical Continuation Methods , 2003, ISBN: 9780898715446

[7] Zangwill, W. I. , Nonlinear Programming, Prentice Hall, Englewood Cliffs, N. J., 1969.

[8] R. M. Karp, Reducibility among combinatorial problems, Complexity of Computer Computations, R. E. Miller and J. W, Thatcher, eds., Plenum Press, New York, 1972, pp. 85-104

[9] H. Robbins, S. Monro, A Stochastic Approximation Method,The Annals of Mathematical Statistics, Vol. 22, No. 3. (Sep., 1951), pp. 400-407

[10] Davis, M. "Hilbert's Tenth Problem is Unsolvable." Amer. Math. Monthly 80, 233-269, 1973

[11] Lagarias J. C. , Worst-case complexity bounds for algorithms in the theory of integral quadratic forms, Journal of Algorithms, ISSN: 0196-6774, Vol: 1, Issue: 2, Page: 142-186, 1980, https://doi.org/10.1016/0196-6774(80)90021-8

[12] http://dud.docking.org/

[13] Matiyasevich, Yu. V. Hilbert's Tenth Problem. Cambridge, MA: MIT Press, 1993

[14] Yu-Chen Lo, Stefano E. Rensi, Wen Torng, Russ B. Altman Machine learning in chemoinformatics and drug discovery Drug Discovery Today, Volume 23, Issue 8, 2018, pp. 1538-1546

[15] Quiroga R, Villarreal MA (2016) Vinardo: A Scoring Function Based on Autodock Vina Improves Scoring, Docking, and Virtual Screening. PLoS ONE 11(5): e0155183. https://doi.org/10.1371/journal.pone.0155183

[16] Goodsell, D. S. and Olson, A. J. , Automated docking of substrates to proteins by simulated annealing. Proteins, 1990, 8: 195-202. https://doi.org/10.1002/prot.340080302

[17] Wang,C. and Zhang, Yi. , Improving Scoring-Docking-Screening Powers of Protein-Ligand Scoring Functions using Random Forest, J Comput Chem. 2017 Jan 30; 38(3): 169–177. https://doi.org/10.1002/jcc.24667

[18] Li Y, Han L, Liu ZH, Wang RX. J Chem Inf Model. 2014;54:1717., https://doi.org/10.1021/ci500081m