Global optimization with unknown Lipschitz constant

Dissertation
zur Erlangung des akademischen Grades
doctor rerum naturalium (Dr. rer. nat.)

vorgelegt dem Rat
der Fakultät für Mathematik und Informatik
der Friedrich-Schiller-Universität Jena

von Dipl.-Math. Matthias Ulrich Horn
geboren am 11. Mai 1975 in Frankfurt am Main

Gutachter

1. Prof. Dr. Erich Novak
2. Prof. Dr. James M. Calvin

Tag der letzten Prüfung des Rigorosums: 07. Oktober 2005

Tag der öffentlichen Verteidigung: 13. Oktober 2005

Zusammenfassung

Wir betrachten das globale Optimierungsproblem, d.h. für eine reellwertige und beschränkte Funktion f suchen wir eine Stelle x des Definitionsbereichs, deren Funktionswert $f(x)$ nahe dem Infimum $\inf f$ liegt. Wir untersuchen den Fall, daß f eine d-variate, Lipschitz-stetige Funktion ist, die in einem gewissen Sinne in der Nähe der globalen Minimalstelle(n) nicht zu langsam wächst. Wir leisten zwei Beiträge:

- Wir zeigen, daß für eine optimale Methode Adaptivität notwendig ist und daß Randomisierung (Monte Carlo) keine weiteren Vorteile bringt.
- Wir stellen eine Methode vor, die universell ist im folgenden Sinne: Diese Methode hat die optimale Konvergenzrate auch in dem Fall, daß weder die Lipschitzkonstante noch die übrigen Klassenparameter bekannt sind.

Im folgenden werden wir detaillierter. Wir betrachten Funktionen

$$f : [0,1]^d \to \mathbb{R},$$

die die folgenden zwei Eigenschaften haben:

1. Die Funktion f ist Lipschitz-stetig mit Lipschitzkonstante $L > 0$:

   $$\forall x, y \in [0,1]^d \qquad |f(x) - f(y)| \leq L\|x - y\|_\infty.$$

   Hier bezeichnet $\| \cdot \|_\infty$ die Maximumsnorm.

2. Sei λ^d das Lebesgue-Maß auf $[0,1]^d$. Für die Niveaumengen

   $$A(f,\delta) := \{x \in [0,1]^d : f(x) \leq \inf f + \delta\}$$

   existieren Konstanten $\varrho, D > 0$, sodaß

   $$\forall\, 0 \leq \delta \leq \varrho \qquad \lambda^d(A(f,\delta)) \leq D\,\delta^{d/2}.$$

Wir sagen $f \in F^d_{L,D,\varrho}$. Wir erwähnen einige Merkmale dieser Problemklasse: Eine Funktion $f \in F^d_{L,D,\varrho}$ kann mehrere Minimalstellen haben. Als Glattheit wird nur Lipschitz-Stetigkeit gefordert. Weiterhin kann man zu jeder zweifach stetig differenzierbaren Funktion, die an jeder von endlich vielen Minimalstellen eine positiv definite Hesse-Matrix hat, Parameter L, D, ϱ finden, sodaß $f \in F^d_{L,D,\varrho}$. Wir diskutieren die Problemklasse $F^d_{L,D,\varrho}$ ausführlich in Chapter 1.

Für $F^d_{L,D,\varrho}$ schlagen wir zwei adaptive Optimierungsalgorithmen vor, die wir in Chapter 2 vorstellen. Der erste findet Anwendung in der Situation, daß die Lipschitzkonstante oder eine obere Schranke für sie bekannt ist. Die beiden anderen Klassenparameter D und ϱ fließen nicht in die Definition des Algorithmus ein und brauchen nicht bekannt sein. Sie sind allerdings wichtig für eine Abschätzung der Kosten.

Zusätzlich kann es vorkommen, daß auch für die Lipschitzkonstante keine Information vorliegt. Dann ist unklar, welche Methode benutzt werden soll. Von einem praktischen Standpunkt aus gesehen sind deshalb solche Algorithmen von Interesse, die für viele Problemklassen gute Ergebnisse liefern. Der zweite von uns vorgeschlagene Algorithmus kann auch dann angewendet werden, falls wir keine Information über L haben. Tatsächlich konvergiert er für alle stetigen Funktionen.

In Chapter 3 vergleichen wir unsere Algorithmen mit möglicherweise besseren Algorithmen. Die Methoden, die wir dabei betrachten, haben zwei wesentliche Merkmale:

1. Ein Algorithmus kann reelle Zahlen exakt verarbeiten und speichern.

2. Ein Algorithmus kann eine endliche Anzahl Funktionswerte benutzen, d.h. er hat nur partielle Information über die Zielfunktion.

Wir unterscheiden zwischen adaptiven und nichtadaptiven Methoden, je nachdem, wie der Algorithmus die Auswertungsstellen bestimmt.

Eine grundlegende Frage in der Numerik ist, ob adaptive Methoden wesentlich besser sind als nichtadaptive. Um diese Frage zu beantworten, führen wir zunächst die Begriffe Kosten und Fehler ein. Für einen Algorithmus A definieren wir $\mathrm{cost}(A, F^d_{L,D,\varrho})$ als die im worst case benötigte Anzahl Funktionswerte, um für $f \in F^d_{L,D,\varrho}$ eine Näherungslösung zu liefern. Gibt A für f die Näherungslösung x zurück, so ist der funktionsweise Fehler mit $\Delta(A, f) := f(x) - \inf f$ gegeben. Der Fehler $\Delta(A, F^d_{L,D,\varrho})$ ist der worst case Fehler für $f \in F^d_{L,D,\varrho}$. Die Fehlerzahlen

$$e^{ad}_n(F^d_{L,D,\varrho}) := \inf \{\Delta(A, F^d_{L,D,\varrho}) : A \text{ ad. Meth.}, \mathrm{cost}(A, F^d_{L,D,\varrho}) \le n\},$$

$$e_n^{non}(F^d_{L,D,\varrho}) := \inf\{\Delta(A, F^d_{L,D,\varrho}) : A \text{ nichtad. Meth., } \mathrm{cost}(A, F^d_{L,D,\varrho}) \le n\}$$

sind die Werkzeuge, um die obige Frage zu beantworten. Wir sagen, adaptive Methoden sind wesentlich besser als nichtadaptive, falls

$$\lim_{n\to\infty} \frac{e_n^{ad}(F^d_{L,D,\varrho})}{e_n^{non}(F^d_{L,D,\varrho})} = 0.$$

Wir zitieren einige bekannte Komplexitätsergebnisse in Section 3.2, insbesondere für Klassen, die Ähnlichkeiten zu unserer haben. Für $F^d_{L,D,\varrho}$ zeigen wir

$$e_n^{ad}(F^d_{L,D,\varrho}) \asymp n^{-2/d},$$
$$e_n^{non}(F^d_{L,D,\varrho}) \asymp n^{-1/d}.$$

Insbesondere liefern also adaptive Methoden einen quadratischen speed-up gegenüber nichtadaptiven.

Eine weitere grundlegende Frage in der Numerik ist, ob Randomisierung (Monte Carlo) eine zusätzliche Verbesserung der Konvergenzrate ermöglicht. Wir betrachten randomisierte Methoden und weisen nach, daß sie die Konvergenzgeschwindigkeit nicht wesentlich erhöhen können.

Ein Hauptbeitrag dieser Arbeit beschäftigt sich mit Universalität. Wir sagen, eine Methode A ist universell für eine Familie $\mathcal{F}$ von Problemklassen, falls A für jede Klasse aus $\mathcal{F}$ die optimale Konvergenzrate hat. Wir zeigen, daß unser zweiter Algorithmus universell ist für $(F^d_{L,D,\varrho})$. Weiterhin zeigen wir, daß unser erster Algorithmus die optimale Konvergenzrate hat, falls die Lipschitzkonstante bekannt ist.

Es scheint, daß das Thema Universalität in der globalen Optimierung zuvor nicht betrachtet wurde. Die Artikel, die wir finden konnten und die sich mit universellen Methoden beschäftigen, betrachten andere numerische Probleme wie beispielsweise Integration, siehe auch Section 3.2.

Für unseren universellen Algorithmus nehmen wir in Chapter 4 ein heuristisches fine-tuning vor. Dieses hat zwar keine Auswirkungen auf die theoretischen Ergebnisse, jedoch erhoffen wir uns Verbesserungen des nichtasymptotischen Verhaltens, das für eine Implementierung wichtig ist. Wir testen verschiedene Versionen unseres universalen Algorithmus an einigen populären Testfunktionen. Wir verwenden die Ergebnisse, um eine konkrete Version zu empfehlen. Wir illustrieren die Funktionsweise dieser Version, indem wir die Punktewahl für eine uni- und zwei bivariate Testfunktionen zu verschiedenen Stadien der Approximation abbilden.

Wir schließen mit zwei Anwendungen des universellen Algorithmus, die wir in Chapter 5 vorstellen. Wir überprüfen die Vermutung

$$\mathrm{diam}\ M_3 \ge \frac{9}{5}$$

für das Minkowski-Kompaktum M_3. Unsere Ergebnisse bestätigen diese Vermutung.

Weiterhin präsentieren wir ein Schema, wie der universelle Algorithmus zum Beweis oberer Schranken mittlerer Fehler eingesetzt werden kann. Für das Wiener-Maß ist das beste bekannte Ergebnis, daß der mittlere Fehler eine obere Schranke von $\mathcal{O}(n^{-1/2})$ hat, wobei n die Anzahl der benutzten Funktionwerte bezeichne. Wir benutzen den universellen Algorithmus, um numerisch eine obere Schranke zu bestimmen. Die Resultate lassen vermuten, daß der mittlere Fehler eine obere Schranke von $\mathcal{O}(n^{-1.93})$ hat.

Ich möchte allen danken, die mich bei dieser Arbeit unterstützt haben. Mein besonderer Dank gilt Herrn Prof. Dr. Erich Novak für großzügig gegebenen Rat und Hilfe. Herr Prof. Dr. James M. Calvin gab wertvollen Rat zum Thema mittlere Fehler in der globalen Optimierung. Herr Dr. habil. Aicke Hinrichs schlug vor, den universellen Algorithmus auf das Minkowski-Kompaktum anzuwenden.

Contents

Introduction

We study the global optimization problem, i.e., for a real-valued and bounded function f we are interested in a point x of the domain whose function value $f(x)$ is close to the infimum $\inf f$. We consider the case that f is d-variate, Lipschitz, and, in a certain sense, does not increase too slowly in a neighborhood of the global minimizer(s). We give two contributions:

- We show that for an optimal method adaptiveness is necessary and that randomization (Monte Carlo) yields no further advantage.
- We present a method that is universal in the following sense: This algorithm has the optimal rate of convergence even if neither the Lipschitz constant nor any other function parameter is known.

Now we give more details. We consider functions

$$f : [0,1]^d \to \mathbb{R}$$

that fulfill the two following properties:

1. The function f is Lipschitz with constant $L > 0$:

   $$\forall x, y \in [0,1]^d \qquad |f(x) - f(y)| \leq L\|x - y\|_\infty.$$

   Here, $\|\cdot\|_\infty$ denotes the maximum norm.

2. Let λ^d be the Lebesgue measure on $[0,1]^d$. For the level sets

   $$A(f,\delta) := \{x \in [0,1]^d : f(x) \leq \inf f + \delta\}$$

   there exist constants $\varrho, D > 0$ such that

   $$\forall\, 0 \leq \delta \leq \varrho \qquad \lambda^d(A(f,\delta)) \leq D\,\delta^{d/2}.$$

We say $f \in F^d_{L,D,\varrho}$. We mention some features of this problem class: A function $f \in F^d_{L,D,\varrho}$ can have several local and global minima. For smoothness we only require Lipschitz continuity. Furthermore, for every twice continuously differentiable function f with positive definite Hessian at each of finitely many minimizers we can always find parameters L, D, ϱ such that $f \in F^d_{L,D,\varrho}$. We discuss the problem class $F^d_{L,D,\varrho}$ in detail in Chapter 1.

For $F^d_{L,D,\varrho}$ we propose two adative optimization algorithms, which we present in Chapter 2. The first one fits to the situation that the Lipschitz constant L or an upper bound of it is known. The class parameters D and ϱ are unimportant for the design of the algorithm and need not to be known. They are important only for the cost estimation.

In practice, there is often no information about the Lipschitz constant, either. Then, it is not clear which method should be used. Therefore, from a practical point of view, an algorithm that yields good results for many problem classes is of special interest. Our second algorithm can be applied if neither L nor the parameters D and ϱ are known. In fact, it converges for every continuous function.

We compare our algorithms with possibly better algorithms in Chapter 3. The algorithms we consider have two main properties:

1. An algrithm can store real numbers exactly and calculate with them exactly.

2. An algorithm can use a finite number of function values, i.e., it has only partial information about the objective.

We differ between adaptive and non-adaptive methods, depending on how the evaluation points are determined.

A basic issue in numerics is to find out whether adaptive methods are essentially better than non-adaptive ones. To answer this question we introduce the notions of cost and error. For an algorithm A we define $\mathrm{cost}(A, F^d_{L,D,\varrho})$ to be the number of oracle calls a method A needs for $f \in F^d_{L,D,\varrho}$ in the worst case. Let x be the point that A returns for f. Then, the function-wise error is given by $\Delta(A, f) := f(x) - \inf f$. The error $\Delta(A, F^d_{L,D,\varrho})$ is the worst case error for $f \in F^d_{L,D,\varrho}$. The error numbers

$$e^{ad}_n(F^d_{L,D,\varrho}) := \inf \{\Delta(A, F^d_{L,D,\varrho}) : \ A \text{ ad. meth.}, \ \mathrm{cost}(A, F^d_{L,D,\varrho}) \le n\},$$

$$e^{non}_n(F^d_{L,D,\varrho}) := \inf \{\Delta(A, F^d_{L,D,\varrho}) : \ A \text{ non-ad. meth.}, \ \mathrm{cost}(A, F^d_{L,D,\varrho}) \le n\}$$

are the tools to decide the above question. We say that adaptive methods are essentially better than non-adaptive ones if

$$\lim_{n\to\infty} \frac{e_n^{ad}(F^d_{L,D,\varrho})}{e_n^{non}(F^d_{L,D,\varrho})} = 0.$$

We give some known complexity results in Section 3.2, in particular for classes related to ours. For $F^d_{L,D,\varrho}$ we show

$$e_n^{ad}(F^d_{L,D,\varrho}) \asymp n^{-2/d},$$

$$e_n^{non}(F^d_{L,D,\varrho}) \asymp n^{-1/d}.$$

Hence we have a quadratic speed-up for adaptive methods.

Another issue of numerics is the question whether randomization (Monte Carlo) can further improve the rate of convergence. We consider randomized methods and see that they cannot yield any further essential improvement.

A main result of this work concerns universality. We say a method A is universal for a family $\mathcal{F}$ of problem classes if for every $F \in \mathcal{F}$ the method has the optimal rate of convergence. We show that our second method is universal for $(F^d_{L,D,\varrho})$. We also show that our first algorithm has the optimal rate of convergence if the Lipschitz constant is known.

It seems that the issue of universality in global optimization is new. The only papers about universality we could find consider other numerical problems such as integration, see Section 3.2.

Once we have established the universality of our second method, we do some heuristic fine-tuning, see Chapter 4. This tuning cannot improve the theoretic results. However, we try to get a better non-asymptotic behavior, which is important for implementation. We test several modifications of the universal algorithm on some popular test functions. We use the results to recommend the use of a particular version. To illustrate its performance, we show how this version behaves for a one- and two bivariate test functions.

We close with two mathematical applications of our universal algorithm, which we present in Chapter 5. We test the conjecture

$$\text{diam } M_3 \geq \frac{9}{5}$$

for the Minkowski compactum M_3. Our results confirm this conjecture.

Furthermore, we present a scheme of how our algorithm may be applied to prove upper bounds for mean errors. For the Wiener measure the best known result is that the mean error has an upper bound of $\mathcal{O}(n^{-1/2})$ where

n denotes the number of used function calls. We use our universal method to determine an upper bound numerically. The result suggests that the error has an upper bound of $\mathcal{O}(n^{-1.93})$.

I want to thank all those who supported me on this work. I especially thank my supervisor Prof. Dr. Erich Novak for generous advice and support. Prof. Dr. James M. Calvin gave valuable advice on mean errors in global optimization. Dr. habil. Aicke Hinrichs suggested the application on the Minkowski compactum.

Chapter 1

The function class $F^d_{L,D,\varrho}$

We consider the global optimization problem for objectives $f : [0,1]^d \to \mathbb{R}$ that have two properties:

1. The function f is Lipschitz with constant $L > 0$:

$$\forall x, y \in [0,1]^d \qquad |f(x) - f(y)| \leq L\|x - y\|_\infty. \tag{1.1}$$

   Here, $\| \cdot \|_\infty$ denotes the maximum norm.

2. Let λ^d be the Lebesgue measure on $[0,1]^d$. For the level sets

$$A(f,\delta) := \{x \in [0,1]^d : \ f(x) \leq \inf f + \delta\} \tag{1.2}$$

   there exist constants $\varrho, D > 0$ such that

$$\forall\, 0 \leq \delta \leq \varrho \qquad \lambda^d(A(f,\delta)) \leq D\,\delta^{d/2}. \tag{1.3}$$

We say

$$f \in F^d_{L,D,\varrho}.$$

1.1 Properties of $F^d_{L,D,\varrho}$

We want to highlight some properties of $F^d_{L,D,\varrho}$ and give some examples.

- For smoothness only Lipschitz continuity is required.

  Example 1.1.1. The function $g_1 : [0,1] \to \mathbb{R}$ linearly interpolates the points

$$\left[0, \frac{3}{2}\right], \left[\frac{1}{6}, 0\right], \left[\frac{1}{3}, 1\right], \left[\frac{1}{2}, 0\right], \left[\frac{5}{6}, \frac{2}{3}\right], \left[\frac{11}{12}, -\frac{1}{3}\right], [1,1],$$

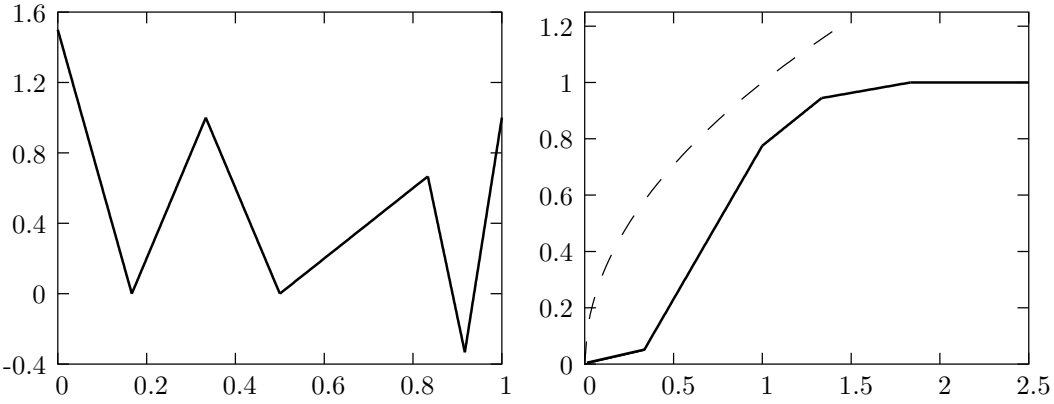


Figure 1.1: The function g_1 and its level set function.

see Figure 1.1. It is Lipschitz with constant

$$L := 16.$$

For the parameters D and ϱ there is no straightforward choice. We choose

$$D := 1.$$

Then, any $\varrho > 0$ is admissible. Since $A(g_1, 11/6)$ is already the whole domain, we set

$$\varrho := 11/6.$$

The dashed line in Figure 1.1 (right-hand side) is the square-root function coresponding to a bound with $D = 1$.
So, for this choice of parameters we have $g_1 \in F^d_{L,D,\varrho}$. We also have $g_1 \in F^d_{L',D',\varrho'}$ for any $L' \geq L$, $D' \geq D$, and $\varrho' \leq \varrho$.

- A function $f \in F^d_{L,D,\varrho}$ may have many global and local minimizers.

  Example 1.1.2. The function $g_2 : [0,1] \to \mathbb{R}$,

  $$g_2(x) := |\cos(3\pi x)|,$$

  has three global minimizers: 1/6, 1/2, 5/6, see Figure 1.2. It is Lipschitz with constant

  $$L := 3\pi.$$

  Again, there is no straightforward choice for D and ϱ. For D we may again choose 1 such that any $\varrho > 0$ is admissible. We can also set

  $$D := 1/2.$$

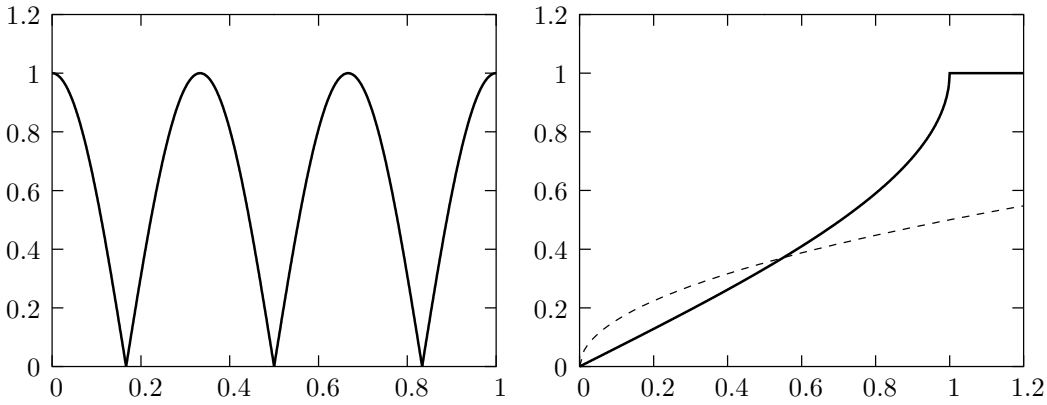


Figure 1.2: The function g_2 and its level set function.

Then,

$$\varrho := 0.4$$

is an admissible choice, as we see in Figure 1.2. The dashed line on the right-hand side is the function $\delta \mapsto 0.5 \cdot \delta^{1/2}$.

- We have an important subclass belonging to $F^d_{L,D,\varrho}$. Assumption (1.3) guarantees a minimum increase in a neighborhood of the global minimizer or, if there are several global minimizers, in a neighborhood of each of them. The upper bound $D\,\delta^{d/2}$ for the level sets $A(f,\delta)$ allows that for

$$f \in C^2[0,1]^d$$

with a finite number of global and local minimizers x^* and a positive Hessian $(\nabla^2 f)(x^*)$ for each of them, we can always find class parameters L, D, ϱ such that $f \in F^d_{L,D,\varrho}$. This is a consequence of the Taylor expansion.
Consider at first $f \in C^2[0,1]^d$ which has a unique minimizer x^* and for which

$$\forall x \in \mathbb{R}^d \qquad x^T(\nabla^2 f(x^*))x \geq M\|x\|_2^2$$

holds for a positive constant M. Here, $\|\cdot\|_2$ denotes the Euclidian norm. The case that f has several minimizers is then straightforward.
To guarantee $f \in F^d_{L,D,\varrho}$ we choose

$$L := \sup_{x\in[0,1]^d} \|\nabla f(x)\|_\infty.$$

A choice of D and ϱ needs some more consideration: For $x \in \mathbb{R}^d$, positive r, and the metric $\|\cdot\|$, we define the balls

$$B(x, r, \|\cdot\|) := \{y \in [0,1]^d : \|x - y\| < r\} \tag{1.4}$$

and

$$B(x,r) := B(x,r,\|\cdot\|_\infty). \tag{1.5}$$

Due to the continuity of the second order derivatives of f, there exists a neighborhood $B(x^*,r,\|\cdot\|_2)$ with

$$\forall y \in B(x^*,r,\|\cdot\|_2) \quad \forall x \in \mathbb{R}^d \qquad x^T(\nabla^2 f(y))x \geq \tfrac{1}{2}\, M \,\|x\|_2^2.$$

Using the estimate of the remainder of the Taylor polynomial, we show

$$\forall x \in B(x^*,r,\|\cdot\|_2) \qquad f(x) \geq g(x) := f(x^*) + \tfrac{1}{4}\, M \,\|x-x^*\|_2^2.$$

Let

$$v(d) := \begin{cases} \dfrac{\pi^n}{\frac{2}{2}\cdot\frac{4}{2}\cdot\ldots\cdot\frac{2n}{2}}, & d = 2n, \\ \dfrac{\pi^{n-1}}{\frac{1}{2}\cdot\frac{3}{2}\cdot\ldots\cdot\frac{2n-1}{2}}, & d = 2n-1, \end{cases}$$

denote the volume of the unit ball in $\mathbb{R}^d$ endowed with the metric induced by the Euclidian norm $\|\cdot\|_2$.

For $\delta \geq 0$ we have

$$A(g,\delta) = B(x^*, 2\sqrt{\delta M^{-1}}, \|\cdot\|_2).$$

So,

$$\lambda^d(A(g,\delta)) \leq \left(\frac{4\delta}{M}\right)^{d/2} v(d).$$

If $A(f,\delta) \subset B(x^*,r,\|\cdot\|_2)$ then

$$\lambda^d(A(f,\delta)) \leq v(d)\left(\frac{4}{M}\right)^{d/2} \delta^{d/2}. \tag{1.6}$$

We define

$$D := v(d)\left(\frac{4}{M}\right)^{d/2}$$

and determine $\varrho > 0$ such that (1.6) is true for all $\delta \leq \varrho$ as follows:
a) We guarantee $A(g,\varrho) \subset \overline{B(x^*,r,\|\cdot\|_2)}$:
It holds $\overline{B(x^*,r,\|\cdot\|_2)} = A(g,\frac{Mr^2}{4})$. We set

$$\varrho \leq \varrho_1 := \frac{Mr^2}{4}. \tag{1.7}$$

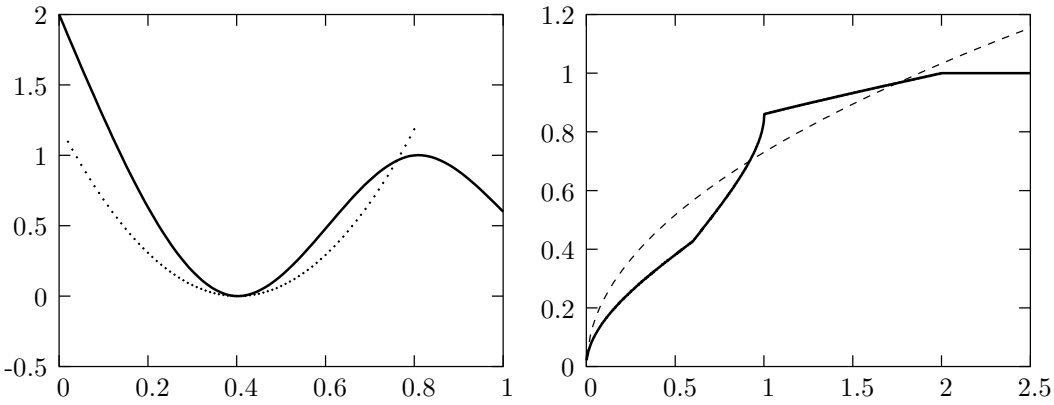


Figure 1.3: The function g_3 and its level set function. The dashed lines are g and its level set function.

b) We guarantee $A(f,\varrho) \subset B(x^*, r, \|\cdot\|_2)$: We set

$$\varrho < \varrho_2 := \inf\{|f(x) - f(x^*)| : x \notin \overline{B(x^*, \varrho_1, \|\cdot\|_2)}\}. \qquad (1.8)$$

For every ϱ fulfilling conditions (1.7) and (1.8)

$$\forall\, 0 \le \delta \le \varrho \qquad \lambda^d(A(f,\delta)) \le D\,\delta^{d/2}$$

holds.

Example 1.1.3. Let g_3 be the natural cubic spline interpolating the points

$$[0,2], [0.4,0], [0.8,1], [1,0.6],$$

see Figure 1.3. The global minimizer

$$x^* = \frac{106}{175} - \frac{2\sqrt{2846}}{525} \approx 0.4025$$

has the function value $g_3(x^*) \approx -0.00011269$. For $x \in [x^*-0.1, x^*+0.1]$ we have $g_3''(x) \ge 15$. So, for the quadratic function

$$g(x) := \frac{15}{2}(x - x^*)^2 + g_3(x^*)$$

we have $g|_{[x^*-0.1,x^*+0.1]} \le g_3|_{[x^*-0.1,x^*+0.1]}$. For g the level set function is easy to calculate:

$$\lambda^1(A(g,\delta)) \le 2 \cdot \sqrt{\frac{2}{15}}\,\delta^{1/2}.$$

We set

$$D := 2 \cdot \sqrt{\frac{2}{15}}.$$

In order to obtain a suitable ϱ we consider conditions (1.7) and (1.8). We have $\varrho_1 = 0.075$, and we can choose $\varrho_2 := 0.5$. We set

$$\varrho := 0.075.$$

We see in Figure 1.3 that this is a cautious choice.

1.2 Admissible parameters

Not all combinations of class parameters make sense. Indeed, for a bad combination $F^d_{L,D,\varrho}$ is empty. As a consequence of Lemma 1.2.2 below, we will always assume

$$\varrho < \min\{L, \tfrac{1}{4}L^2D^{2/d}\}.$$

We define

$$\mathbb{F}^d := \{F^d_{L,D,\varrho} : \varrho < \min\{L, \tfrac{1}{4}L^2D^{2/d}\}\}. \tag{1.9}$$

Proposition 1.2.1. *Let $g : [0, \varrho] \to [0, \infty)$ be piecewise linear with $n \in \mathbb{N}$ nodes $0 = \delta_1 < \delta_2 < \cdots < \delta_n = \varrho$ such that $g(\delta_i) \le D^{1/d}\delta_i^{1/2}$. Suppose for $f : [0,1]^d \to \mathbb{R}$ and $0 \le \delta \le \varrho$ that $\lambda^d(A(f,\delta)) \le g^d(\delta)$. Then*

$$\forall\, 0 < \delta \le \varrho \qquad \lambda^d(A(f,\delta)) \le D\delta^{d/2}.$$

Proof. From $g(\delta_i) \le D^{1/d}\delta_i^{1/2}$ for $i = 1, \ldots, n$ and the fact that the square-root function is concave we conclude that

$$\forall\, 0 \le \delta \le \varrho \qquad g(\delta) \le D^{1/d}\delta^{1/2}.$$

We know $\lambda^d(A(f,\delta)) \le g^d(\delta)$. Furthermore, $x \mapsto x^d$ is strictly monotone. So,

$$\forall\, 0 < \delta \le \varrho \qquad \lambda^d(A(f,\delta)) \le D\delta^{d/2}.$$

□

Lemma 1.2.2. *Let $\varrho < \min\{L, \frac{1}{4}L^2D^{2/d}\}$. Then*

1. *$F^d_{L,D,\varrho}$ is nonempty.*
2. *$F^d_{L,D,\varrho}$ is not symmetric.*

3. $F^d_{L,D,\varrho}$ *is not convex.*

4. *For every* $x \in [0,1]^d$ *the class* $F^d_{L,D,\varrho}$ *contains several functions* f *with* $f(x) = \min f$.

Proof. 1. Let

$$f(x) := L\,\|x\|_\infty.$$

Then for $\delta \le L$

$$A(f,\delta) = [0, \delta L^{-1}]^d,$$

i.e., $\lambda^d(A(f,\delta)) = (\delta L^{-1})^d$. For $\delta \le L^2 D^{2/d}$ it follows $\lambda^d(A(f,\delta)) \le D\,\delta^{d/2}$.
2. We choose $\gamma \in (\varrho, L)$ and define

$$f_\gamma(x) := \begin{cases} L\,\|x\|_\infty, & \|x\|_\infty \le \gamma L^{-1}, \\ \gamma, & \|x\|_\infty > \gamma L^{-1}. \end{cases}$$

Analogously to *1.*, we show $f_\gamma \in F^d_{L,D,\varrho}$. Furthermore,

$$\lambda^d(\{f_\gamma \ge \gamma\}) = \lambda^d(\{f_\gamma = \gamma\}) \ge (1 - \gamma L^{-1})^d > 0.$$

It holds $-f_\gamma \notin F^d_{L,D,\varrho}$ since

$$\lambda^d(A(-f_\gamma, 0)) \ge (1 - \gamma L^{-1})^d > 0.$$

3. We choose

$$s \in (\max\{1/2, \varrho/L\}, 1).$$

For

$$\alpha := D\,\varrho^{d/2} - \left(\frac{\varrho}{L}\right)^d$$

we have $\alpha > 0$. We set

$$r := \min\left\{\frac{1-s}{2}, \alpha^{1/d}, \varrho^{1/2} D^{1/d}\right\}, \qquad \delta_1 := \frac{r^2}{D^{2/d}}.$$

In particular,

$$\delta_1 \le \varrho.$$

For

$$f_{r,s}(x) := \begin{cases} \dfrac{\delta_1 \|x\|_\infty}{r}, & \|x\|_\infty \le r, \\ \delta_1 + L(\|x\|_\infty - r), & r < \|x\|_\infty \le \dfrac{s+1}{2}, \\ \delta_1 + L\left(\dfrac{s+1}{2} - r\right), & \dfrac{s+1}{2} < \|x\|_\infty \le 1, \end{cases}$$

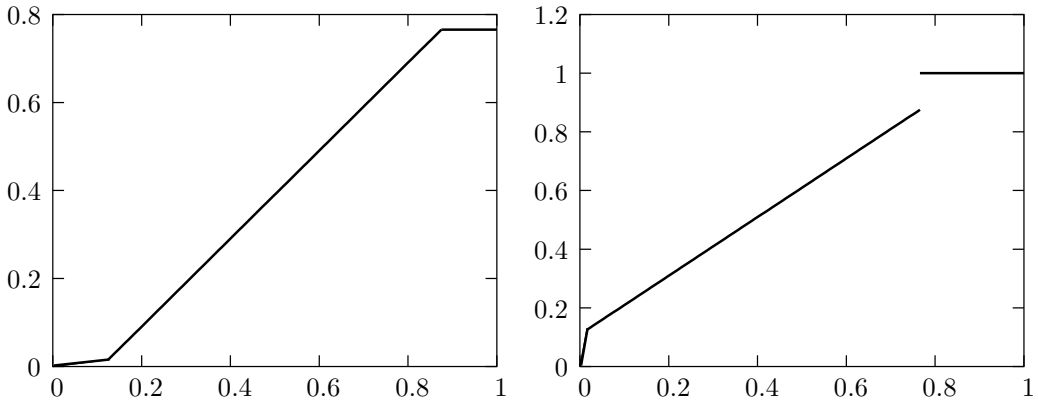


Figure 1.4: The function $f_{r,s}$ and its level set function for $d = 1$, $L = 1$, $D = 1$, $\varrho = 1/9$ and $s = 3/4$

we show $f_{r,s} \in F^d_{L,D,\varrho}$: Due to $\delta_1 \leq \varrho < \frac{1}{4} L^2 D^{2/d}$ it holds

$$\frac{\delta_1}{r} = \frac{\delta_1^{1/2}}{D^{1/d}} < \tfrac{1}{4} L.$$

Consequently, the Lipschitz condition is fulfilled. For the level sets we have

$$\lambda^d(A(f_{r,s}, \delta)) = \begin{cases} \left(\dfrac{\delta r}{\delta_1}\right)^d, & \delta \leq \delta_1, \\ \left(\dfrac{\delta - \delta_1}{L} + r\right)^d, & \delta_1 < \delta < \delta_1 + L\left(\dfrac{s+1}{2} - r\right), \\ 1, & \delta_1 + L\left(\dfrac{s+1}{2} - r\right) \leq \delta. \end{cases}$$

Consider the piece-wise linear function g with nodes and function values

$$g(0) = 0, \quad g(\delta_1) = r, \quad g(\varrho) = (\varrho - \delta_1)/L + r.$$

Application of Proposition 1.2.1 yields

$$\forall\, 0 < \delta \leq \varrho \qquad \lambda^d(A(f_{r,s}, \delta)) \leq D\, \delta^{d/2},$$

i.e., we showed $f_{r,s} \in F^d_{L,D,\varrho}$. In the same way, we prove

$$x \mapsto f_{r,s}(\mathbb{1} - x) \in F^d_{L,D,\varrho},$$

with

$$\mathbb{1} := (1, \ldots, 1)^T \in \mathbb{R}^d.$$

We consider the convex combination

$$g(x) := \tfrac{1}{2}(f_{r,s}(x) + f_{r,s}(\mathbb{1} - x)).$$

a) For $x \in [0,1]^d$ let $\widehat{x} := \|x\|_\infty \cdot \mathbb{1}$. For all $x \in [0,1]^d$ the point $\widehat{x}$ lies on the diagonal $\{t \cdot \mathbb{1}, t \in [0,1]\}$. Furthermore, we have $\|x\|_\infty = \|\widehat{x}\|_\infty$. So,

$$\begin{aligned} g(x) &= \tfrac{1}{2}(f_{r,s}(x) + f_{r,s}(\mathbb{1} - x)) &= \tfrac{1}{2}(f_{r,s}(\widehat{x}) + f_{r,s}(\mathbb{1} - x)) \\ &\geq \tfrac{1}{2}(f_{r,s}(\widehat{x}) + f_{r,s}(\mathbb{1} - \widehat{x})) &= g(\widehat{x}). \end{aligned}$$

b) The function g is point symmetric in $\mathbb{1}/2$.
c) From $\widehat{x} < \widehat{y} \leq \mathbb{1}/2$ we conclude $g(\widehat{x}) \leq g(\widehat{y})$.
d) We have

$$g(0) = \min_{y \in [0,1]^d} g(y) = \tfrac{1}{2}[\delta_1 + L((s+1)/2 - r)].$$

e) It holds

$$[0,r]^d \cup [1-r,1]^d \subset A(g, \delta_1/2).$$

In conclusion,

$$\lambda^d(A(g,\delta_1)) \geq \lambda^d(A(g,\delta_1/2)) \geq 2\,r^d = 2\,D\,\delta_1^{d/2} > D\,\delta_1^{d/2}.$$

Since $\delta_1 \leq \varrho$ we have $g \notin F^d_{L,D,\varrho}$.
4. We show that for every $x \in [0,1]^d$

$$f_x(z) := L\,\|z - x\|_\infty \in F^d_{L,D,\varrho}.$$

Obviously, the Lipschitz condition is fulfilled. For $\delta \leq \frac{1}{4}D^{2/d}L^2$ we have

$$\lambda^d(A(f_x,\delta)) \leq D\,\delta^{d/2}.$$

(We have equality for $x = \frac{1}{2} \cdot \mathbb{1}$ and $\delta = \frac{1}{4}D^{2/d}L^2$.) Let $\theta > 0$ be defined by $\varrho = \frac{\theta^2}{4}D^{2/d}L^2$. It holds $\theta \in (0,1)$. We define

$$\widetilde{f}_x(z) := \theta L \|z - x\|_\infty.$$

We conclude $\widetilde{f}_x \in F^d_{\theta L,D,\varrho} \subset F^d_{L,D,\varrho}$. □

Remark 1.2.3. We showed that for $\varrho < \min\{\frac{1}{4}L^2D^{2/d}, L\}$ the problem class $F^d_{L,D,\varrho}$ is neither convex nor symmetric. The problem class being not convex or not symmetric is a necessary condition for the situation that adaptive methods are essentially better than non-adaptive ones, see Section 3.2.

Chapter 2

Optimization algorithms

We propose two adaptive algorithms. The first one is applicable in a situation where the Lipschitz parameter L of the objective is known, or at least an upper bound of it. The second one is suitable for the case that the Lipschitz parameter is unknown. For both algorithms, the knowledge of the other parameters D and ϱ is unimportant for their definition and their success. Nevertheless, they are important for the cost estimation.

For both algorithms we give cost and error bounds according to the following definitions:

The error of a method A applied to a function f and returning $A(f) = x_* = x_*(f)$ is given by

$$\Delta(A, f) := f(x_*) - \inf f.$$

The (worst case) error of the method A is

$$\Delta(A, F^d_{L,D,\varrho}) := \sup \{\Delta(A, f) : \ f \in F^d_{L,D,\varrho}\}.$$

The function-wise cost $\mathrm{cost}(A, f)$ of the method A applied to f is the number of function calls the method A uses for f. The (worst case) cost of A is

$$\mathrm{cost}(A, F^d_{L,D,\varrho}) := \sup \{\mathrm{cost}(A, f) : \ f \in F^d_{L,D,\varrho}\}.$$

We come to some preliminaries for the two algorithms we propose. Let $e^{(i)}$ be the i-th unit vector in $\mathbb{R}^d$ having $(e^{(i)})_i = 1$ and $(e^{(i)})_j = 0$ for $j \neq i$. The algorithms use

$$Y(j) := \left\{ \sum_{i=1}^{d} a_j \cdot e^{(i)}, \ \ a_j \in \{-3^{-j+1}, 0, 3^{-j+1}\} \right\} \setminus \{0\}, \quad j \in \mathbb{N}.$$

Each of the sets $Y(j)$ consists of $3^d - 1$ points. Finally, let

$$\mathcal{M} := (\tfrac{1}{2}, \ldots, \tfrac{1}{2})^T$$

denote the midpoint of the unit cube in $\mathbb{R}^d$.

2.1 The case of a known Lipschitz constant

For the case of a known Lipschitz parameter L, we propose the following optimization algorithm $S(L, k)$ performing step$(L,\ 1)$, $\ldots$, step$(L,\ k)$ as described in Figure 2.1. After these k steps, $S(L, k)$ returns x_*. It is similar to the one of Perevozchikov (1990).

Lemma 2.1.1. *Let $f : [0, 1]^d \to \mathbb{R}$ be Lipschitz with constant $L > 0$. After step(L, j) and for each global minimizer x^* there exists a pair $(x_j, f(x_j)) \in N_{L,j}$ such that $\|x_j - x^*\|_\infty \leq 2^{-1}3^{-j+1}$.*

Proof. By induction.
$j = 1$: We have $(\mathcal{M}, f(\mathcal{M})) \in N_{L,1}$ and $\|x - \mathcal{M}\|_\infty \leq 2^{-1}$ for all $x \in [0, 1]^d$.
$j \to j + 1$: Let $(x_j, f(x_j)) \in N_{L,j}$ such that $\|x_j - x^*\|_\infty \leq 2^{-1}3^{-j+1}$. In step$(L, j + 1)$, we check whether $f(x_j) \leq f_* + L\, 2^{-1}3^{-j+1}$. This is true:

$$|f(x_j) - f_*| \leq |f(x_j) - f(x^*)| \leq L\, 2^{-1}3^{-j+1}. \tag{2.1}$$

So we choose the pairs

$$(x_j + y, f(x_j + y)), \quad y \in Y(j + 1),$$

to be in $N_{L,j+1}$. For (at least) one of these y we have

$$\|x_j + y - x^*\|_\infty \leq 2^{-1}3^{-j}.$$

Choose $x_{j+1} := x_j + y$ for such a y. □

The sets $N_{L,j}$ are subsets of the equidistant meshes

$$\text{mesh}(j) := \left\{ \sum_{i=1}^{d} \alpha_i \cdot e^{(i)},\ \alpha_i \in \{2^{-1}3^{-j+1} + l \cdot 3^{-j+1}, l = 0, \ldots, 3^{j-1} - 1\} \right\}.$$

Furthermore, $S(L, k)$ guarantees the same level of approximation as mesh(k) in the following sense: For every global minimizer x^* we have

$$\min_{x \in \text{mesh}(k)} \|x - x^*\|_\infty \leq 2^{-1}3^{-k+1}, \qquad \min_{x \in N_{L,k}} \|x - x^*\|_\infty \leq 2^{-1}3^{-k+1}.$$

step(L, 1):
get $f(\mathcal{M})$ (oracle call);
set $N_{L,1} := \{(\mathcal{M}, f(\mathcal{M}))\}$;
set $x_* := \mathcal{M}$; $f_* := f(\mathcal{M})$.

step(L, j), $2 \le j \le k$:
set $N_{L,j} := \emptyset$;
for $(x, f(x)) \in N_{L,j-1}$ do
  if $(f(x) \le f_* + L\,2^{-1}3^{-j+2}\,)$ then $\qquad (*)$
    set $N_{L,j} := N_{L,j} \cup \{(x, f(x))\}$;
    for $y \in Y(j)$ do
      get $f(x+y)$ (oracle call);
      set $N_{L,j} := N_{L,j} \cup \{(x+y, f(x+y))\}$;
      if $(f(x+y) < f_*)$ then
        set $x_* := x+y$; $f_* := f(x+y)$;
      end if;
    next y;
  end if;
next x;

Figure 2.1: $S(L,k)$ performs step$(L,\,1)$, …, step$(L,\,k)$ and then returns x_*

We are now ready to prove cost and error bounds of $S(L,k)$. We need the following constants:

$$\varepsilon_{L,k} := L2^{-1}3^{-k+1}, \tag{2.2}$$

$$j(L,\varrho) := \lceil \log_3(L/(2\varrho)) \rceil + 3, \tag{2.3}$$

$$j(L,D,\varrho,d) := \lceil \log_3(L/(2\varrho^2 D^{2/d})) \rceil + 6, \tag{2.4}$$

$$c(d) := \frac{3^{d/2}}{3^{d/2}-1}. \tag{2.5}$$

For $d \geq 1$, we have $c(d) \in (1, 2.37]$. We use an idea of Perevozchikov (1990) to prove the following result.

Theorem 2.1.2. *1. Error estimation: For $k \in \mathbb{N}$, we have*

$$\Delta(S(L,k), F^d_{L,D,\varrho}) \leq \varepsilon_{L,k}.$$

2. Cost estimation: We have

(a) for $k \in \mathbb{N}$

$$\mathrm{cost}(S(L,k), F^d_{L,D,\varrho}) \leq 3^{d(k-1)},$$

(b) for $k \geq j(L,\varrho)$

$$\mathrm{cost}(S(L,k), F^d_{L,D,\varrho}) \leq \left(\tfrac{27}{2}\, L/\varrho\right)^d + c(d)\,(3^d-1)\,D\,L^{d/2}2^{-d/2}\left(3^{(k-1)d/2} - \left(\tfrac{3}{2}L/\varrho\right)^{d/2}\right),$$

(c) for $k \geq \max\{j(L,\varrho), j(L,D,\varrho,d)\}$

$$\mathrm{cost}(S(L,k), F^d_{L,D,\varrho}) \leq DL^{d/2}2^{-d/2}3^{(k+1)d/2+1} = DL^d 2^{-d} 3^{d+1}\varepsilon_{L,k}^{-d/2}.$$

Proof. 1. Let $f \in F^d_{L,D,\varrho}$. From Lemma 2.1.1 we know that there exists a pair $(x_k, f(x_k)) \in N_{L,k}$ such that $\|x_k - x^*\|_\infty \leq 2^{-1}3^{-k+1}$. Then

$$|f(x_*) - f(x^*)| \leq |f(x_k) - f(x^*)| \leq L\,2^{-1}3^{-k+1}.$$

2. For *(a)*, we have that $S(L,k)$ chooses only points in $\mathrm{mesh}(k)$, which consists of $3^{d(k-1)}$ points.

(b). Let

$$N^*_{L,j-1} \tag{2.6}$$

be the set of pairs $(x, f(x)) \in N_{L,j-1}$ that in $\mathrm{step}(L,j)$ pass the test $f(x) \leq f_* + L2^{-1}3^{-j+2}$. Then, in $\mathrm{step}(L,j)$, the number of new function evaluations is bounded by

$$|N_{L,j} \setminus N_{L,j-1}| \leq (3^d - 1)N^*_{j-1}.$$

We can use this estimation for every $\mathrm{step}(L,j)$ with $N^*_{L,j-1} \subset A(f,\varrho)$. As in *1.*, we can show

$$\min\{f(y) : y \in N^*_{L,j-1}\} \leq \min f + L\,2^{-1}3^{-j+2}.$$

Furthermore,

$$\forall x \in N^*_{L,j-1} \quad f(x) \leq f_* + L\,2^{-1}3^{-j+2} \leq \min\{f(y) :\, y \in N^*_{L,j-1}\} + L\,2^{-1}3^{-j+2}.$$

So we get

$$\forall x \in N^*_{L,j-1} \qquad x \in A(L\,3^{-j+2}).$$

For $x, y \in N^*_{L,j-1}$ and $x \neq y$ we have

$$B(x, 2^{-1}3^{-j+2}) \subset A(L\,2^{-1}3^{-j+3}), \quad B(x, 2^{-1}3^{-j+2}) \cap B(y, 2^{-1}3^{-j+2}) = \emptyset.$$

For $L\,2^{-1}3^{-j+3} \leq \varrho$, i.e.,

$$j - 1 \geq \lceil \log_3(L/(2\varrho)) \rceil + 2 = j(L, \varrho) - 1,$$

we can estimate

$$\begin{aligned} |\, N^*_{L,j-1}\,| &\leq \frac{\lambda^d(A(L\,2^{-1}3^{-j+3}))}{\lambda^d(B(x, \frac{1}{2}\,3^{-j+2}))} \leq \frac{D\,(L\,2^{-1}3^{-j+3})^{d/2}}{(3^{-j+2})^d} \\ &= DL^{d/2}2^{-d/2}3^{(j-1)d/2}\,. \end{aligned} \tag{2.7}$$

So the number of new points in level j is bounded by

$$(3^d - 1)\,DL^{d/2}2^{-d/2}3^{(j-1)d/2}.$$

It follows immediately that

$$\begin{aligned} &\mathrm{cost}(S(L,k), F^d_{L,D,\varrho}) \\ \leq\ & |\mathrm{mesh}(j(L,\varrho) - 1)| + \sum_{j=j(L,\varrho)}^{k} (3^d - 1)\,D\,L^{d/2}2^{-d/2}3^{(j-1)d/2} \\ \leq\ & \left(\tfrac{27}{2}\,L/\varrho\right)^d + c(d)\,(3^d - 1)\,D\,L^{d/2}2^{-d/2}(3^{(k-1)d/2} - 3^{(k(L,\varrho)-2)d/2}) \\ \leq\ & \left(\tfrac{27}{2}\,L/\varrho\right)^d + c(d)\,(3^d - 1)\,D\,L^{d/2}2^{-d/2}\left(3^{(k-1)d/2} - \left(\tfrac{3}{2}L/\varrho\right)^{d/2}\right). \end{aligned}$$

So we proved *(b)*.
Under the assumptions of *(c)* we have for $d = 1$

$$\tfrac{27}{2}L/\varrho \leq c(d)DL^{1/2}2^{-1/2}3^{(k-1)/2}$$

and for $d \geq 2$

$$\left(\tfrac{27}{2}\,L/\varrho\right)^d \leq \tfrac{3}{2} \cdot 3^d DL^{d/2}2^{-d/2}3^{(k-1)d/2}.$$

In both cases we conclude from *(b)* that

$$\mathrm{cost}(S(L,k), F^d_{L,D,\varrho}) \leq D\,L^{d/2}2^{-d/2}3^{(k+1)d/2+1} = DL^d 2^{-d}3^{d+1}\varepsilon_{L,k}^{-d/2}.$$

□

2.2 The case of an unknown Lipschitz constant

We now turn to the case of an unknown Lipschitz parameter L. For this situation, we propose the algorithm Z as described in Figure 2.2. It uses the steps of $S(L, \cdot)$ for a sequence $(L(i))_{i\in\mathbb{N}}$ of increasing Lipschitz constants. An additional step(L, 1') is also used:

step(L, 1'):
oracle call: get $f(\mathcal{M})$;
set $N_{L,1} := \{(\mathcal{M}, f(\mathcal{M}))\}$.

The constants $L(i)$ and two controlling functions

$$\mathit{lastconst} : \mathbb{N} \to \mathbb{N}, \quad l \mapsto \mathit{lastconst}(l)$$

$$\mathit{laststep} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}, \quad (l, i) \mapsto \mathit{laststep}(l, i)$$

determine the behavior of the algorithm. In a first definition, we only require the following properties:

- $L(i+1) > L(i)$ for all $i \in \mathbb{N}$,
- *lastconst* increasing,
- $\mathit{laststep}(l, i)$ increasing in l, and decreasing in i.

$Z(k)$ is a diagonal scheme. The parameter k in $Z(k)$ is the number of performed diagonals. In diagonal l, the algorithm examines the objective assuming the Lipschitz constants $L(1), \ldots, L(\mathit{lastconst}(l))$, i.e., for constant $L(i)$, the algorithms performs

$$\text{step}(L(i), \mathit{laststep}(l-1, i)+1), \ldots, \text{step}(L(i), \mathit{laststep}(l, i)).$$

While the algorithm has only one instance of f_* and x_*, it uses separate sets $N_{L(i),j}$ for each constant $L(i)$.
Note that the objective f will be evaluated at certain points for several constants $L(i)$, i.e., several times, the midpoint $\mathcal{M}$ for example *lastconst(k)* times. We will discuss in Section 2.2.1 why this is reasonable.

```
for l from 1 to k do    # diagonal l
    for i from 1 to lastconst(l)    # constant L(i)
        for j from 1 to laststep(l, i) do    # step j
            if (i ≠ 1, j = 1) then
                apply step(L(i), 1');
            else
                apply step(L(i), j);
            end if;
        next j;
    next i;
next l;
return x*;
```

Figure 2.2: The algorithm $Z(k)$. It uses the steps defined in Figure 2.1.

Let $m, k \in \mathbb{N}$ such that $\mathit{lastconst}(k) \geq m$. The method $Z(k)$ performs step($L(m)$, 1') (or 1) to step($L(m)$, $\mathit{laststep}(k, m)$). This way, $Z(k)$ determines the sets

$$N_{L(m),1}, \ldots, N_{L(m),\mathit{laststep}(k,m)}.$$

We have the following analogue to Lemma 2.1.1:

Lemma 2.2.1. *Let $f \in F^d_{L,D,\varrho}$ with $L \leq L(m)$ for some $m \in \mathbb{N}$, and $k \in \mathbb{N}$ such that $\mathit{lastconst}(k) \geq m$. Let x^* be a global minimizer. For $1 \leq j \leq \mathit{laststep}(k, m)$ there exists a pair $(x_j, f(x_j)) \in N_{L(m),j}$ such that $\|x_j - x^*\|_\infty \leq 2^{-1}3^{-j+1}$.*

Proof. The proof is similar to that of Lemma 2.1.1. The only difference in the situation is that now, the least value found so far f_* is shared and altered by function evaluations applying different constants $L(i)$. However, f_* enters the proof only in (2.1). Here, $f(x^*) \leq f_* \leq f(x_j)$ is needed. This is also true for the new situation. □

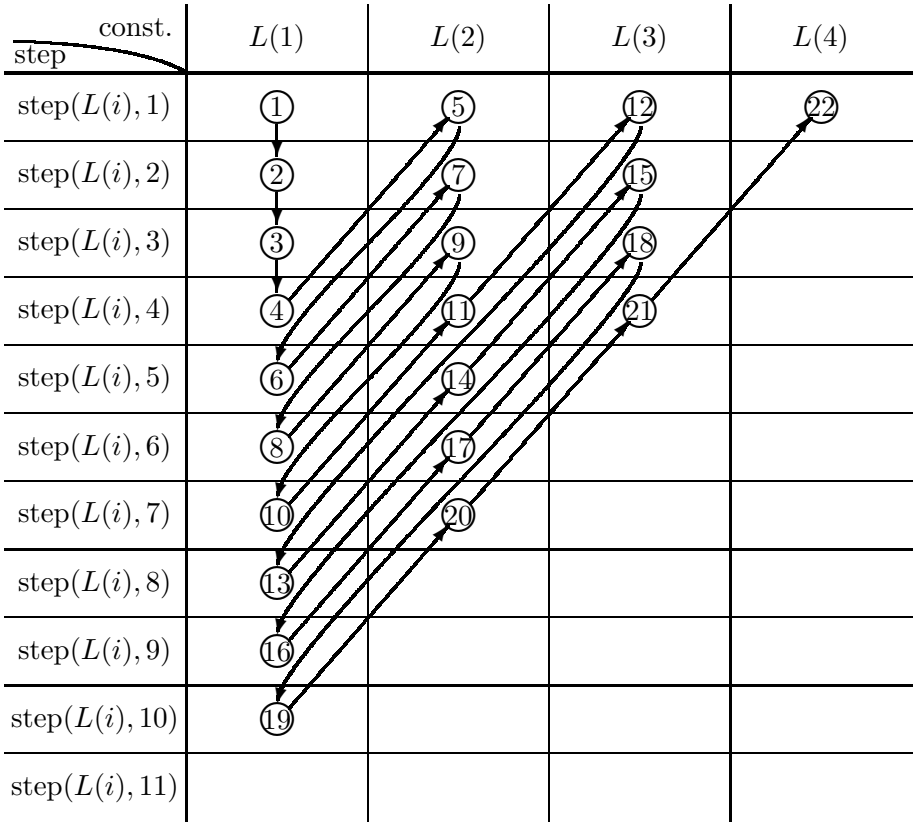


Figure 2.3: Scheme of $Z(h,k)$ for $h=3$ and $k=10$

We want to examine $Z(k)$ for the choice

$$L(i) := 3^{i-1}, \quad i \in \mathbb{N}, \tag{2.8}$$

$$\mathit{lastconst}(l) := \left\lceil \frac{l}{h} \right\rceil, \qquad \mathit{laststep}(l,i) := \begin{cases} l - h(i-1), & \text{if } i \le \mathit{lastconst}(l), \\ 0, & \text{else,} \end{cases}$$

with parameter $h \in \{3, 4, \dots\}$ and the ceiling function

$$\lceil \cdot \rceil : \mathbb{R} \to \mathbb{Z}, \qquad \lceil r \rceil := \inf\{z \in \mathbb{Z} : \, z \ge r\}. \tag{2.9}$$

We will discuss the choice of h in Remark 2.2.3 and in Chapter 4. Let

$$Z(h,k)$$

be the algorithm defined in this manner. We use the constants

$$\varepsilon_{L,h,k} := L^{h+1}2^{-1}3^{-k+1}, \tag{2.10}$$

$$c(d,h) := \frac{1}{1-3^{-hd/2}}, \qquad c'(d,h) := \frac{1}{1-3^{(1-h/2)d}}, \tag{2.11}$$

$$C(L,d,h) := c(d)\,\left[c(d,h)3^{-d/2} + c'(d,h)L^{-hd/2}\right], \tag{2.12}$$

with $c(d)$ defined as in (2.5). For $h \geq 3$, $d \geq 1$, and $L \geq 1$ we have

$$c(d,h) \in (1, 1.25], \quad c'(d,h) \in (1, 2.37], \quad C(L,d,h) \in (0, 7.37].$$

Theorem 2.2.2. *Let $m \in \mathbb{N}$ and $k \geq h(m-1)+1$.*

1. *Error estimation:*

$$\Delta(Z(h,k), F^d_{L(m),D,\varrho}) \leq \varepsilon_{L(m),h,k}.$$

2. *Cost estimation:*

$$\begin{aligned}
& \operatorname{cost}(Z(h,k), F^d_{L(m),D,\varrho}) \\
\leq\ & \left(3L^{h+1}(m)/\varrho\right)^d c(2d,h) + c(2d)\,3^{d(\lceil k/h\rceil - m)}\left(\tfrac{27}{2}\,L(m)/\varrho\right)^d + \\
& C(L(m),d,h)\,L(m)^{(h+1)d/2} D\,2^{-d/2}3^{(k-1)d/2} \\
=\ & \left(3L^{h+1}(m)/\varrho\right)^d c(2d,h) + \\
& \left(\tfrac{27}{2}\,L(m)/\varrho\right)^d c(2d)\,2^{-d/h}L(m)^{d/h}\,\varepsilon^{-d/h}_{L(m),h,k} + \\
& C(L(m),d,h)\,D\,L(m)^{(h+1)d}2^{-d}\,\varepsilon^{-d/2}_{L(m),h,k}.
\end{aligned}$$

Proof. 1. Let us assume that we apply $Z(h,k)$ to a function $f \in F^d_{L(m),D,\varrho}$. For the constant $L(m)$ our method performs step($L(m)$, 1) or step($L(m)$, 1') up to step($L(m)$, $k-h(m-1)$). From Lemma 2.2.1 we know that for every global minimizer x^* there exists $x_{k-h(m-1)} \in N_{L(m),k-h(m-1)}$ such that

$$\|x_{k-h(m-1)} - x^*\|_\infty \leq 2^{-1}3^{-k+h(m-1)+1}.$$

It follows immediately that

$$|f(x_*) - f(x^*)| \leq |f(x_{k-h(m-1)}) - f(x^*)| \leq L(m)\,2^{-1}3^{-k+h(m-1)+1} = \varepsilon_{L(m),h,k}.$$

2. The cost estimation is similar to the one in the proof of Theorem 2.1.2. Recall $N^*_{L,j-1}$ to be defined as in (2.6). We consider the steps for constants

$L \geq L(m)$ and those for constants $L < L(m)$ separately.
Let $m \leq i \leq \mathit{lastconst}(k)$. For the new points in step$(L(i), j)$, we have

$$|N_{L(i),j} \setminus N_{L(i),j-1}| \leq (3^d - 1)|N^*_{L(i),j-1}|.$$

We can use this estimation for all j with $N^*_{L(i),j-1} \subset A(f, \varrho)$. As in *1.*, we can show

$$\min\{f(y) : y \in N_{L(i),j-1}\} \leq \min f + L(m)\, 2^{-1} 3^{-j+2}.$$

Furthermore,

$$\forall\, x \in N^*_{L(i),j-1} \qquad f(x) < \min\{f(y) : y \in N_{L(i),j-1}\} + L(i)\, 2^{-1} 3^{-j+2},$$

so we get

$$\forall\, x \in N^*_{L(i),j-1} \qquad x \in A(f, (L(m) + L(i))2^{-1}3^{-j+2}).$$

For $x, y \in N^*_{L(i),j-1}$ with $x \neq y$ we have

$$B(x, 2^{-1}3^{-j+2}) \subset A(f, (2L(m) + L(i))\, 2^{-1}3^{-j+2}),$$

$$B(x, 2^{-1}3^{-j+2}) \cap B(y, 2^{-1}3^{-j+2}) = \emptyset.$$

For $(2L(m) + L(i))2^{-1}3^{-j+2} \leq \varrho$, i.e.,

$$j - 1 \geq \lceil \log_3((2L(m) + L(i))/(2\varrho)) \rceil + 1 =: j(m, i, \varrho) - 1, \tag{2.13}$$

we get

$$\begin{aligned} |N^*_{L(i),j-1}| &\leq \frac{\lambda^d(A(f, (2L(m) + L(i))2^{-1}3^{-j+2}))}{\lambda^d(B(x, 2^{-1}3^{-j+2}))} \\ &\leq D(L(m) + 2^{-1}L(i))^{d/2}\, 3^{(j-2)d/2}. \end{aligned} \tag{2.14}$$

It follows immediately that

$$|N_{L(i),j} \setminus N_{L(i),j-1}| \leq (3^d - 1)D(L(m) + 2^{-1}L(i))^{d/2}\, 3^{(j-2)d/2}.$$

For $k - h(i-1) \geq j(m,i,\varrho)$ we get the cost estimation

$$\begin{aligned}
& |N_{L(i),1}| + \sum_{j=2}^{k-h(i-1)} |N_{L(i),j} \setminus N_{L(i),j-1}| \\
\leq\ & |\operatorname{mesh}(j(m,i,\varrho)-1)| + \sum_{j=j(m,i,\varrho)}^{k-h(i-1)} (3^d-1) D (L(m) + 2^{-1}L(i))^{d/2}\, 3^{(j-2)d/2} \\
\leq\ & \left(\frac{9(2L(m)+L(i))}{2\varrho}\right)^d + \\
& (3^d-1) D (L(m) + 2^{-1}L(i))^{d/2} 3^{(j(m,i,\varrho)-2)d/2}\, \frac{3^{(k-h(i-1)-j(m,i,\varrho)+1)d/2}-1}{3^{d/2}-1} \\
\leq\ & \left(\frac{27L(i)}{2\varrho}\right)^d + (3^d-1) D L(i)^{d/2} c(d)\, 2^{-d/2} 3^{(k-h(i-1)-1)d/2}.
\end{aligned}$$

Now let $1 \leq i \leq m-1$. Again, we want to estimate $|N^*_{L(i),j-1}|$. Before the algorithm applies $\operatorname{step}(L(i),j)$, the last step concerning $L(m)$ was $\operatorname{step}(L(m), j-1-h(m-i))$. So we know that at the beginning of $\operatorname{step}(L(i),j)$

$$f_* \leq \min f + L(m) 2^{-1} 3^{-j+h(m-i)+2}.$$

Consequently,

$$\forall x \in N^*_{L(i),j-1} \qquad f(x) \leq \min f + 2^{-1} 3^{-j+2} [L(i) + L(m)\, 3^{h(m-i)}].$$

Define

$$j'(m,i,h,\varrho) := \lceil \log_3((L(i) + L(m)(1 + L^h(m)/L^h(i)))/(2\varrho)) \rceil + 2. \quad (2.15)$$

We have $j'(m,i,h,\varrho) \leq h(m-i) + m + 1 - \lfloor \log_3(\varrho) \rfloor$. If $k \geq hm+1$ then $k - h(i-1) \geq j'(m,i,h,\varrho)$. In the same manner as above we can show for $j \geq j'(m,i,h,\varrho)$ that

$$\begin{aligned}
|N^*_{L(i),j-1}| &\leq \frac{\lambda^d(A(f, 3^{-j+2}L(m)3^{h(m-i)}))}{\lambda^d(B(x, 2^{-1}3^{-j+2}))} \\
&\leq D\, L(m)^{d/2} 3^{(j+h(m-i)-2)d/2}.
\end{aligned}$$

We conclude that

$$
\begin{aligned}
& |N_{L(i),1}| + \sum_{j=2}^{k-h(i-1)} |N_{L(i),j} \setminus N_{L(i),j-1}| \\
\leq\ & |\mathrm{mesh}(j'(m,i,h,\varrho)-1)| + \sum_{j=j'(m,i,h,\varrho)}^{k-h(i-1)} (3^d-1)DL(m)^{d/2}3^{(j+h(m-i)-2)d/2} \\
\leq\ & \left(\frac{3L^{h+1}(m)}{L^h(i)\varrho}\right)^d + (3^d-1)DL(m)^{d/2}c(d)3^{(k+h(m-2i+1)-2)d/2}.
\end{aligned}
$$

In order to get an estimation for $\mathrm{cost}(Z(h,k), F^d_{L(m),D,\varrho})$, we sum up these numbers for the constants $L(1),\ldots,L(\lceil k/h\rceil)$:

$$
\begin{aligned}
& \mathrm{cost}(Z(h,k), F^d_{L(m),D,\varrho}) \leq \sum_{i=1}^{\lceil k/h\rceil} \left[|N_{L(i),1}| + \sum_{j=2}^{k-h(i-1)} |N_{L(i),j} \setminus N_{L(i),j-1}| \right] \\
\leq\ & \sum_{i=1}^{m-1} \left(\frac{3L^{h+1}(m)}{L^h(i)\varrho}\right)^d + \sum_{i=m}^{\lceil k/h\rceil} \left(\frac{9(3L(i))}{2\varrho}\right)^d + \\
& \sum_{i=1}^{m-1} (3^d-1)DL(m)^{d/2}c(d)3^{(k+h(m-2i+1)-2)d/2} + \\
& \sum_{i=m}^{\lceil k/h\rceil} (3^d-1)DL(i)^{d/2}c(d)2^{-d/2}3^{(k-h(i-1)-1)d/2} \\
\leq\ & \left(3L^{h+1}(m)/\varrho\right)^d \sum_{i=0}^{m-2} 3^{-hdi} + \left(\frac{27}{2}\frac{L(m)}{\varrho}\right)^d \sum_{i=0}^{\lceil k/h\rceil-m} 3^{di} + \\
& (3^d-1)DL(m)^{d/2}c(d)3^{(k+h(m-1)-2)d/2} \sum_{i=0}^{m-2} 3^{-dhi} + \\
& (3^d-1)DL(m)^{d/2}c(d)2^{-d/2}3^{(k-h(m-1)-1)d/2} \sum_{i=0}^{\lceil k/h\rceil-m} 3^{(1-h/2)di} \qquad (2.16) \\
\leq\ & \left(3L^{h+1}(m)/\varrho\right)^d c(2d,h) + c(2d)\, 3^{d(\lceil k/h\rceil-m)} \left(\tfrac{27}{2}\, L(m)/\varrho\right)^d + \\
& (3^d-1)DL(m)^{d/2}c(d)\, c(2d,h)\, 3^{(k+h(m-1)-2)d/2} + \\
& (3^d-1)DL(m)^{d/2}c(d)2^{-d/2}3^{(k-h(m-1)-1)d/2}c'(d,h) \\
\leq\ & \left(3L^{h+1}(m)/\varrho\right)^d c(2d,h) + c(2d)\, 3^{d(\lceil k/h\rceil-m)} \left(\tfrac{27}{2}\, L(m)/\varrho\right)^d + \\
& (3^d-1)DL(m)^{d/2}c(d)3^{(k-1+h(m-1))d/2} \left[c(2d,h)3^{-d/2} + c'(d,h)2^{-d/2}3^{-(m-1)hd}\right]
\end{aligned}
$$

$$\leq \left(3L^{h+1}(m)/\varrho\right)^d c(2d,h) + c(2d)\, 3^{d(\lceil k/h\rceil - m)} \left(\tfrac{27}{2}\, L(m)/\varrho\right)^d + C(L(m),d,h)\, L(m)^{(h+1)d/2} D\, 2^{-d/2} 3^{(k-1)d/2}.$$

With

$$\varepsilon_{L(m),h,k}^{-d/2} = L(m)^{-(h+1)d/2} 2^{d/2} 3^{(k-1)d/2}, \quad \varepsilon_{L(m),h,k}^{-d/h} = L(m)^{-d-d/h} 2^{d/h} 3^{d(k-1)/h},$$

we get

$$\begin{aligned} & \operatorname{cost}(Z(h,k), F^d_{L(m),D,\varrho}) \\ \leq\ & \left(3L^{h+1}(m)/\varrho\right)^d c(2d,h) + \left(\tfrac{27}{2}\, L(m)/\varrho\right)^d c(2d)\, 2^{-d/h} L(m)^{d/h}\, \varepsilon_{L(m),h,k}^{-d/h} + \\ & C(L(m),d,h)\, D\, L(m)^{(h+1)d} 2^{-d}\, \varepsilon_{L(m),h,k}^{-d/2}. \end{aligned}$$

□

Remark 2.2.3. We can now explain the restriction on the parameter h: In order for the sum in (2.16) to be bounded for all $k \in \mathbb{N}$, we need $h \geq 3$. The parameter h allows us to decide whether to focus on local or on global search in the following sense: The cost used by performing the steps for $L(i)$ with $i \geq m$ are approximately

$$(3^d - 1) D L(i)^{d/2} c(d)\, 2^{-d/2} 3^{(k-h(i-1)-1)d/2}.$$

For constant $L(i+1)$ we spend $3^{-(h-1)d/2}$ times as much as we spend for $L(i)$. Choosing a high value for h leads to focus on a precise approximation of found (local) minima. This is done in steps for small constants $L(i)$. On the other hand, a low value of h, of say, 3 or 4, will focus more on global search, performed by steps for big constants $L(i)$.

We will give an heuristic advice on how to choose h in Chapter 4.

We close the examinations in this section by showing that $(Z(k)(f))_{k\in\mathbb{N}}$ converges for every continuous f.

Lemma 2.2.4. *Let $f \in C[0,1]^d$. Then*

$$\lim_{k\to\infty} f((Z(k)(f))) = \inf f.$$

Proof. Since f is continuous, it is sufficient to show that

$$\forall j \in \mathbb{N} \quad \forall x \in \operatorname{mesh}(j) \quad \exists\, k \in \mathbb{N} \qquad Z(k) \text{ evaluates at } x.$$

Let $M := \sup|f|$ and $\widetilde{L} := 4M3^{j-2}$. For $L \geq \widetilde{L}$ we show by induction

$$\forall\, 1 \leq l \leq j \quad \forall x \in \operatorname{mesh}(l) \qquad (x, f(x)) \in N_{L,l}.$$

$l = 1$ is obvious.
$l - 1 \to l$ for $2 \leq l \leq j$: Let $x \in \text{mesh}(l-1)$. Under the assumption of the induction, we have $(x, f(x)) \in N_{L,l-1}$. Furthermore,

$$|f(x) - f_*| \leq 2M = \widetilde{L}2^{-1}3^{-j+2} \leq L2^{-1}3^{-l+2}.$$

So for all $y \in Y(l) \cup \{0\}$ we have

$$(x + y, f(x + y)) \in N_{L,l}.$$

Now let $i \in \mathbb{N}$ such that $L(i) \geq \widetilde{L}$, and $k \in \mathbb{N}$ such that *lastconst*$(k) \geq i$ and *laststep*$(k, i) \geq j$. Then Z(k) evaluates f at every $x \in \text{mesh}(j)$. □

2.2.1 A remark on computational cost and on storage requirements

The cost definition we use is limited to the information cost, i.e., the number of function calls the method uses. However, the development of arithmetic cost and storage requirements are important for implementation, too. We will sketch that for the first and the second algorithm our cost definition is a good measure also for these quantities. This is not obvious. In fact, the policy of the second algorithm not to store every function value is necessary to obtain this result.

We examine the arithmetic cost and storage requirements for the algorithm $Z(h, k)$, then ask how much the (information) cost bound could be improved if we stored all function values, and then reconsider the arithmetic cost for this case. A similar argumentation applies for the simpler situation of the first algorithm.

Arithmetic cost and storage requirements for $Z(h, k)$

We start with the arithmetic cost. We reconsider Figure 2.2. We neglect the arithmetic cost necessary for the three "for"-loops and the "if"-loop. So we must still consider the steps(L, j) as in Figure 2.1, and step$(L, 1')$ as on page 20. For step$(L, 1)$ and step$(L, 1')$ we have one oracle call, i.e., the information cost is 1. On the other hand, the arithmetic cost is bounded by a constant. In step(L, j) with $j \geq 2$, we have to run an outer "for"- and an "if"-loop for every point $(x, f(x)) \in N_{L,j-1}$. The effort to run these loops is bounded by a fixed constant times the information cost of step$(L, j-1)$. The cost of the inner "for"-loop is bounded by a constant times the information cost of step(L, j). Altogether, the arithmetic cost behaves linearly to the information cost.

We turn to the storage requirements. We neglect the storage of x_*, f_*, and the variables needed to run the "for"-loops and the "if"-loop. The storage requirements for the sets $N_{L,j}$ remain to be considered. The information cost is an upper bound for their storage requirements since for every $(x, f(x)) \in N_{L,j}$ there has been an oracle call at x, and if $(x, f(x))$ is stored several times, l times say, then x has also been evaluated (at least) l times.

Possible information cost reduction

We want to find out how much (information) cost we may save if we store all function evaluations and evaluate at the same point only once.

Proposition 2.2.5. *Let $f : [0,1]^d \to \mathbb{R}$ be Lipschitz with constant $L > 0$. For $\widetilde{L} \geq 3\,L$ we have*

$$\forall\, j \in \mathbb{N} \qquad N_{L,j} \subset N_{\widetilde{L},j}\,.$$

Proof. We show

$$\exists j \in \mathbb{N}\ \ N_{L,j} \not\subset N_{\widetilde{L},j} \ \ \Rightarrow\ \ \exists\, x_1, x_2 \in [0,1]^d\ \ |f(x_1) - f(x_2)| > L\|x_1 - x_2\|_\infty.$$

Let $x \in N_{L,j}$ but $x \notin N_{\widetilde{L},j}$. From $x \in N_{L,j}$ we conclude that there exists a $y \in N_{L,j-1}$ with

$$f(y) \leq \min\{f(z) :\, z \in N_{L,j-1}\} + L\,2^{-1}3^{-j+2}, \qquad \|\,y - x\,\|_\infty \leq 3^{-j+1}.$$

Since $x \notin N_{\widetilde{L},j}$ we conclude for this y that

$$f(y) > \min\{f(z) : z \in N_{\widetilde{L},j-1}\} + \widetilde{L}\,2^{-1}3^{-j+2}.$$

Consequently,

$$\min\{f(z) :\, z \in N_{\widetilde{L},j-1}\} < \min\{f(z) :\, z \in N_{L,j-1}\} - \underbrace{(\widetilde{L} - L)2^{-1}3^{-j+2}}_{=L\,3^{-j+2}}.$$

So,

$$\inf f < \min\{f(z) :\, z \in N_{L,j-1}\} - L\,3^{-j+2},$$

which is a contradiction to Lemma 2.1.1. □

It is easy to see that a similar result for a Lipschitz constant $\widetilde{L} < L$ cannot hold. However, we can use Proposition 2.2.5 to improve the cost bound of

Theorem 2.2.2. Again, we assume $f \in F^d_{L(m),D,\varrho}$. For Lipschitz constants $L(i)$ with $m \le i \le \mathit{lastconst}(k)$ we only have to consider the cost

$$\begin{aligned} & \sum_{j=k-hi+1}^{k-h(i-1)} |N_{L(i),j} \setminus N_{L(i),j-1}| \\ \le\ & (3^d-1)DL(i)^{d/2}c(d)2^{-d/2}3^{(k-h(i-1)-2)d/2} \end{aligned}$$

if $k - hi + 1 \ge j(m, i, \varrho)$. We compare this with the corresponding cost as in the proof of Theorem 2.2.2:

$$\begin{aligned} & |N_{L(i),1}| + \sum_{j=2}^{k-h(i-1)} |N_{L(i),j} \setminus N_{L(i),j-1}| \\ \le\ & \left(\frac{27L(i)}{2\varrho}\right)^d + (3^d-1)DL(i)^{d/2}c(d)\, 2^{-d/2}3^{(k-h(i-1)-1)d/2}. \end{aligned}$$

We save the cost of $\left(\frac{27L(i)}{2\varrho}\right)^d$, which belongs to the less important part of the cost estimation. For the important one, we reduce the cost by the factor $3^{-d/2}$. This is a good improvement, especially for higher dimensions. However, we cannot improve the rate of convergence. Furthermore, this improvement is restricted to only a part of the cost estimation. The cost bound for constants $L(i) < L(m)$ and $L(i) = L(\mathit{lastconst}(k))$ stays the same.

Arithmetic cost revisited

We still assume that we store all function evaluations. Evaluating only once at the same point means that for every function value that we need during a step$(L(i), j)$ with $i \ge 2$, we first check whether it was already determined and then, in case it was not, perform an oracle call.

Let us assume that we already evaluated at n points and that we must now ask whether a certain point has already been evaluated. If we do not sort the evaluation points then we need to assume a worst-case cost of $\mathcal{O}(n)$. We conclude for the whole checking procedure that after n function calls we have a total arithmetic cost of order $\mathcal{O}(n^2)$.

We are in a better situation if we sort. It is well-known that the best sorting routines can reduce the arithmetic cost for the complete sorting procedure to $\mathcal{O}(n \cdot \log n)$. In this case, an additonal logarithmic factor must still be considered, i.e., even when sorting, the information cost is no longer a good measure for the arithmetic cost.

For the above reason, we decided not to store all function evaluations.

Remark 2.2.6. We have only been vague about the definition of the arithmetic cost. It will be treated in detail in Section 3.1. However, the above argumentation, which uses an intuitive understanding, is consistent with the definition that follows.

Chapter 3

Optimality results

We prove lower error bounds for adaptive deterministic, non-adaptive deterministic and adaptive randomized methods showing that

- the algorithms $S(L, \cdot)$ and $Z(h, \cdot)$ have the optimal rate of convergence,
- adaptiveness is essential for optimality and yields a quadratic speed-up,
- up to constants, randomization (Monte Carlo) gives no further advantage.

We also show that our second algorithm is universal for $\mathbb{F}^d$ as defined in (1.9). Our first method has the optimal rate of convergence if the Lipschitz constant is known.

We start with some important concepts.

3.1 The concept of Information-Based Complexity

Let F be a problem class of real valued and bounded functions with a common domain $\mathcal{D} \subset \mathbb{R}^d$. For some classes F every function $f \in F$ can be described with a finite number of parameters. This is the case, e.g., for classes of polynomials $x \mapsto a_n x^n + \cdots + a_0$. However, for many interesting classes, and also for $F^d_{L,D,\varrho}$, this is not the case. In consequence, we cannot assume to have a closed formula for every $f \in F$. This means in particular that f cannot be involved completely into the computation.

Instead, we assume that we have access to a finite number of function values provided by a subroutine, the oracle. The computation itself is done with the Unlimited Register Machine with an Oracle, a model of an idealized

computer. In particular, we assume that we can calculate with real numbers and that we can store them.

3.1.1 The Unlimited Register Machine with an Oracle

Our model of computation is the Unlimited Register Machine (URM) with an Oracle. We follow the approach of NOVAK (1995), but add the ceiling function to the allowed artihmetic instructions. We do not mention what happens if an error occurs, e.g., if we divide by zero or if the program does not terminate. For those cases and to learn more about the URM we refer to NOVAK (1995).

For every $k \in \mathbb{Z}$ we have a register R_k with an entry r_k which is a real number. At the beginning of the computation we have $r_k = 0$ for all $k \in \mathbb{Z}$. The contents of the registers may be altered by the machine in response to certain instructions. An algorithm is a finite list of instructions. After the machine has worked off this list and if $d := r_0 \in \mathbb{N}$ then the output is the vector $(r_1, \ldots, r_d)$. For the global optimization problem, the output is interpreted as the coordinates of the approximation x_*. The following instructions can be used:

Arithmetic instructions

- Addition of a constant. For $i \in \mathbb{Z}$ and $s \in \mathbb{R}$ there is an instruction $\text{Add}(i, s)$. The response of the machine is $r_i := r_i + s$.

- Addition of two numbers. For $i, j, k \in \mathbb{Z}$ there is an instruction $\text{Add}(i, j, k)$. The response of the machine is $r_i := r_j + r_k$.

- Multiplication with a constant. For $i \in \mathbb{Z}$ and $s \in \mathbb{R}$ there is an instruction $\text{Mul}(i, s)$. The response of the machine is $r_i := r_i \cdot s$.

- Multiplication of two numbers. For $i, j, k \in \mathbb{Z}$ there is an instruction $\text{Mul}(i, j, k)$. The response of the machine is $r_i := r_j \cdot r_k$.

- Division of two numbers. For $i, j, k \in \mathbb{Z}$ there is an instruction $\text{Div}(i, j, k)$. The response of the machine is $r_i := r_j / r_k$.

- Ceiling operation. For $i \in \mathbb{Z}$ there is an operation $\text{Ceil}(i)$. The response of the machine is $r_i := \lceil r_i \rceil$, where $\lceil \cdot \rceil$ is defined as in (2.9).

Jump instruction

- For $i, j \in \mathbb{Z}$ and $l \in \mathbb{N}$ there is an instruction Jump(i, j, l). If $r_i \geq r_j$ then the machine proceeds with the next instruction of the list, otherwise it proceeds with the l-th instruction of the list.

Copy instructions

- Assignment of a constant. For $i \in \mathbb{Z}$ and $s \in \mathbb{R}$ there is an instruction Assign(i, s). The response of the machine is $r_i := s$.

- Direct and indirect copy assignments. For $i, l \in \mathbb{Z}$ there are four different kinds of direct and indirect copy instructions Copy(i, l), Copy(r_i, l), Copy(i, r_l), Copy(r_i, r_l). The response of the machine to Copy(r_i, r_l) is $r_{r_i} := r_{r_l}$ if $r_i, r_l \in \mathbb{Z}$. The response of the other instructions is defined analogously.

Oracle call

- For $i \in \mathbb{Z}$ there is an instruction Oracle(i). The response of the machine is as follows. Let $d := r_{-1}$ and $f \in F$ be the problem element. If $d \in \mathbb{N}$ and $f(r_{-2}, \ldots, r_{-d-1})$ is defined, then the response of the machine is $r_i := f(r_{-2}, \ldots, r_{-d-1})$.

This list of available instructions could be completed by other computations one may wish to perform. For example, one may wish to have instructions that calculate the functions $\sin, \cos, \exp$ eliminating the need to worry about approximation errors. In Section 5.2 we present a variant of our universal algorithm, which also uses the function $x \mapsto x^\alpha$ for some $\alpha \in (0, 1/2)$. Here, we wish to add an instruction to our list delivering this calculation.

Definition 3.1.1. A finite list of instructions for the URM with an Oracle is called an *algorithm* if for all $f \in F$ no error occurs, i.e., if all instructions are used only with admissible parameters, if the calculation terminates, and if in the final configuration r_0 is equal to the dimension d of the domain $\mathcal{D}$ and $(r_1, \ldots, r_d)^T \in \mathcal{D}$.

Synonymously, we also call an algorithm a method.

Example 3.1.2. We present a simple optimization algorithm for univaritate functions defined on $[0, 1]$. The program uses the oracle to get the function values for $1/6$, $1/2$, $5/6$ and then returns the (smallest) point with the least function value.

1. Assign$(-1, 1)$
2. Assign$(-2, 1/6)$
3. Oracle(2)
4. Assign$(-2, 1/2)$
5. Oracle(3)
6. Assign$(-2, 5/6)$
7. Oracle(4)
8. Assign$(1, 5/6)$
9. Jump$(4, 3, 12)$
10. Assign$(1, 1/2)$
11. Copy$(4, 3)$
12. Jump$(4, 2, 14)$
13. Assign$(1, 1/6)$
14. Assign$(0, 1)$

We see that even for this straightforward example, an algorithm written in the language of the URM is rather hard to read. We introduced the URM to define the class of algorithms we want to consider. It is also a necessary requirement to define the notion of cost, especially computational cost. However, we will also use other ways to describe algorithms – as we already did for the algorithms of Chapter 2. The algorithm of Example 3.1.2 can also be described by

$$f \mapsto \min\{\operatorname{argmin}\{f(1/6), f(1/2), f(5/6)\}\}.$$

More generally, we can say that each algorithm A, as defined for the URM, induces a mapping on F, the one that maps f to the output $A(f)$ of A applied to f. We call this mapping A, too.

Definition 3.1.1 allows an algorithm to use function values at adaptively chosen points $x^{(1)}, \ldots, x^{(n)}$. However, the first point $x^{(1)}$ is independent of the objective f and fixed for a particular algorithm. The points $x^{(j)}$ with $j \geq 2$ may depend on those previously chosen and their function values. Also, the number of oracle calls may depend on the observed data. Every such method A can be expressed by

$$A(f) = \phi \circ N(f).$$

The information operator

$$N : F \to \bigcup_{n=1}^{\infty} \mathbb{R}^n \tag{3.1}$$

gives the function values at the adaptively chosen points. They are obtained by applying functions $\psi_j : \mathbb{R}^{j-1} \to \mathcal{D}$ such that

$$x^{(j)} = \psi_j(f(x^{(1)}), \ldots, f(x^{(j-1)})).$$

It stops after n oracle calls according to a stopping rule $s : \bigcup_{j=1}^{\infty} \mathbb{R}^j \to \{0, 1\}$ iff

$$\forall j < n \quad s(f(x^{(1)}), \ldots, f(x^{(j)})) = 1, \qquad s(f(x^{(1)}), \ldots, f(x^{(n)})) = 0.$$

The mapping

$$\phi : \bigcup_{i=1}^{\infty} \mathbb{R}^n \to \mathcal{D}$$

constructs the output $x_* = x_*(f) = A(f)$ using the information vector $N(f)$.

This alternative notion of an algorithm is much more convenient for our purposes to prove cost and error bounds, as we will see in the next section.

3.1.2 Cost and error

We distinguish between computational cost and information cost. For the computational cost we consider all instructions in the list of Section 3.1.1, except for the oracle call. It is generally assumed that the URM always performs a certain instruction at the same cost, i.e., that the cost of an instruction does not depend on the arguments it is called with or the time when it is performed. In principle, it is possible to assign a different cost to every type of instruction. Novak (1995) suggests assigning each arithmetic operation and each jump unitary cost while a copy instruction is free. For the information cost, he prices every oracle call with $c > 0$. This constant is to be chosen in such a way that the cost of an arithmetic operation and that of an oracle call are in an appropriate ratio. This ratio may depend on the particular situation.

The cost approach we use is much simpler. We say that an oracle call has unitary cost while all other instructions are free, i.e., we concentrate on the information cost. So,

$$\mathrm{cost}(A, F)$$

is the number of oracle calls A needs to return an approximation $A(f)$ for $f \in F$ in the worst case. In Section 2.2.1 we showed that for our two algorithms this cost definition is also a good measure for arithmetic cost and storage requirements, i.e., this simplification of the cost definition means only a minor loss of information of how expensive an implementation is.

We come to the notion of the error. Let A be an optimization algorithm. For an element f of the problem class F we define the function-wise error

$$\Delta(A, f) := f(A(f)) - \inf f.$$

The worst-case error of A is given by

$$\Delta(A, F) := \sup_{f \in F} \Delta(A, f).$$

In Section 3.2 we will also give the notion of mean errors and errors of randomized methods.

Example 3.1.3. We consider the problem class

$$F_L := \{f : [0,1] \to \mathbb{R},\ |f(x) - f(y)| \le L|x-y| \text{ for all } x, y \in [0,1]\}$$

and the optimization algorithm A_n that evaluates f at the points

$$x^{(i)} := \frac{1}{2n} + \frac{i-1}{n}, \quad 1 \le i \le n,$$

and then returns $A(f)$ which is the (smallest) point $x^{(i)}$ with the least observed function value.

Let $f \in F_L$ and $x^* \in [0,1]$ be a global minimizer. Since the points $x^{(i)}$ are chosen equidistantly, we can say

$$\exists\, j \in \{1, \dots, n\} \qquad |\, x^* - x^{(j)}\,| \le \frac{1}{2n}.$$

Using the Lipschitz property we get

$$f(A(f)) - f(x^*) \le f(x^{(j)}) - f(x^*) \le L\,|\, x^{(j)} - x^*\,| \le \frac{L}{2n}.$$

So, $\Delta(A_n, f) \le L/(2n)$ and, since $f \in F$ was arbitrary,

$$\Delta(A_n, F) \le L/(2n).$$

3.1.3 Error numbers

We define error numbers for two classes of algorithms depending on the structure of the information vector N as defined in (3.1). We say that an algorithm is adaptive if it fulfills Definition 3.1.1. It is non-adaptive if it additionally fulfills

$$\exists\, n \in \mathbb{N} \quad \exists\, x_1, \dots, x_n \in [0,1]^d \quad \forall f \in F^d_{L,D,\varrho} \qquad N(f) = (f(x_1), \dots, f(x_n)).$$

Note that non-adaptive algorithms are a sub-class of adaptive algorithms.

From a practical point of view, non-adaptive methods are advantageous in that the function values $f(x^{(i)})$ can be determined parallelly. On the other hand, adaptive methods often need far less function calls.

We say

$$A \in \mathcal{A}_n^{ad}(F)$$

if A is adaptive and has $\text{cost}(A, F) \le n$, and

$$A \in \mathcal{A}_n^{non}(F)$$

if A is non-adaptive with $\text{cost}(A,F) \le n$. We have $\mathcal{A}_n^{non}(F) \subset \mathcal{A}_n^{ad}(F)$. We define the error numbers

$$e_n^{ad}(F) := \inf\{\Delta(A,F) : A \in \mathcal{A}_n^{ad}(F)\}, \tag{3.2}$$

$$e_n^{non}(F) := \inf\{\Delta(A,F) : A \in \mathcal{A}_n^{non}(F)\}. \tag{3.3}$$

The error numbers $e_n^{ad}(F)$ give information about the intrinsic difficulty of the optimization problem. Any method yielding an error of at most $e_n^{ad}(F)$ for all $f \in F$ uses at least n function calls for at least one function $f \in F$. These error numbers are the benchmark for our algorithms. Furthermore, we say that adaptive algorithms are essentially better than non-adaptive ones if

$$\lim_{n\to\infty} \frac{e_n^{ad}(F)}{e_n^{non}(F)} = 0.$$

Example 3.1.4. We reconsider the class F_L of Example 3.1.3. Since for the particular non-adaptive algorithm A_n we could prove $\Delta(A_n, F_L) \le L/(2n)$, we conclude that

$$e_n^{ad}(F_L) \le e_n^{non}(F_L) \le \frac{L}{2n}.$$

For a lower bound we consider an arbitrary (adaptive) algorithm A that uses at most n function values. We can assume that A uses exactly n function values. Let $x^{(1)}, \dots, x^{(n)}$ be the points chosen by the algorithm for the function $h(x) \equiv 0$ and $x^{(n+1)} := A(h)$. Without loss of generality, we assume $x^{(1)} \le \cdots \le x^{(n+1)}$. There exists at least one point $z \in [0,1]$ such that

$$\min_{1\le i\le n+1} |z - x^{(i)}| \ge \frac{1}{2(n+1)}.$$

We define the function

$$g_z(x) := \begin{cases} -\frac{L}{2(n+1)} + L|z-x|, & x \in [z - 1/(2(n+1)), z + 1/(2(n+1))], \\ 0, & \text{else.} \end{cases}$$

We easily check $g_z \in F_L$ and $A(g_z) = A(h) = x^{(n+1)}$. So,

$$g_z(A(g_z)) = 0, \qquad \min g_z = -\frac{L}{2(n+1)}.$$

We have $\Delta(A, g_z) = L/(2(n+1))$. Consequently, $\Delta(A, F_L) \ge L/(2(n+1))$, such that

$$\frac{L}{2(n+1)} \le e_n^{ad}(F_L) \le e_n^{non}(F_L) \le \frac{L}{2n}.$$

With some more effort, one can show that $L/(2n) \le e_n^{ad}(F_L)$, see NOVAK (1988), Proposition 1.3.6, for details. This means that for F_L non-adaptive methods are optimal. We will see in the next section that this is no coincidence.

We will also introduce error numbers for mean errors and for randomized methods in Section 3.2.

3.1.4 Universality and tractablilty

We introduce two important notions.

Definition 3.1.5. Let $\mathcal{F}$ be a family of problem classes and $(A_n)_{n\in\mathbb{N}}$ be a sequence of optimization algorithms with $\mathrm{cost}(A_n, F) \le n$ for every $F \in \mathcal{F}$. The methods A_n are called *universal* for $\mathcal{F}$ if

$$\forall F \in \mathcal{F} \quad \exists\, C_F < \infty \quad \forall n \in \mathbb{N} \qquad \Delta(A_n, F) \le C_F \cdot e_n^{ad}(F).$$

We also call a subsequence $(A_{n(i)})_{i\in\mathbb{N}}$ universal for $\mathcal{F}$ if $n(i) \uparrow \infty$.

Definition 3.1.6. Let $(F_d)_{d\in\mathbb{N}}$ be a sequence of function classes. The global optimization problem is *tractable* for $(F_d)_{d\in\mathbb{N}}$ if there exists an estimate of the sort

$$\forall d, n \in \mathbb{N} \qquad e_n^{ad}(F_d) \le C \cdot d^p \cdot n^{-q}$$

with a constant $C > 0$ and $p \ge 0$, $q > 0$, all independent of d and n. It is *strongly tractable* if we can choose $p = 0$.

The general case is that one considers tractabilty concerning the dimension of the domain.

3.2 Some known complexity results

A well examined optimization problem is convex programming. Here, we have a unique global minimizer. Local and global search coincide. For this situation methods are known whose costs behave polynomially in dimension d and error level ε. This means in particular that convex programming is tractable. The ellipsoid method assumes the existence of an oracle that delivers for a problem element f and a point x of the domain both the function value $f(x)$ and the derivative $(\nabla f)(x)$. It yields an approximation to the error level $\varepsilon > 0$ using $\mathcal{O}(d^2 \ln(1/\varepsilon))$ oracle calls. For details and further complexity results for convex functions we refer to the mini-course of Nemirovski (1995).

The problem class $F^d_{L,D,\varrho}$ contains functions with many global minimizers. In this property, it is closely related to Lipschitz classes. For

$$F_L^d := \{f : [0,1]^d \to \mathbb{R},\ |f(x) - f(y)| \le L \,\|x - y\|_\infty \text{ for all } x, y \in [0,1]^d\},$$

we have that a non-adaptive method using equidistant points delivers the optimal result. The class F_L^d is a special case of a convex and symmetric (i.e., $f \in F \Rightarrow -f \in F$) class. In this situation, adaptiveness can yield only a minor improvement, if at all, compared to the best non-adaptive algorithms. See Novak (1988), Proposition 1.3.2, for details. On the other hand, there are examples that adaptiveness helps essentially if the problem class is convex but not symmetric or vice versa.

Let $(a_n)_{n\in\mathbb{N}}$ and $(b_n)_{n\in\mathbb{N}}$ be non-negative sequences and $0 < c \le C < \infty$ such that

$$\exists N \in \mathbb{N} \quad \forall\, n \ge N \qquad c \cdot a_n \le b_n \le C \cdot a_n.$$

Then we write

$$a_n \asymp b_n.$$

For F_L^d, we have $e_n^{ad/non}(F_L^d) \asymp Ln^{-1/d}$. In contrast, we will see for $F_{L,D,\varrho}^d$ that adaption is essential for optimality and that it leads to a quadratic speed-up: $e_n^{ad}(F_{L,D,\varrho}^d) \asymp L^2 D^{2/d} n^{-2/d}$. In both cases, however, the error numbers depend exponentially on d, which means that for F_L^d and $F_{L,D,\varrho}^d$ the optimization problem is not tractable.

One may say that in many situations the worst case definition of error or cost is too pessimistic and that the average case may give a more realistic picture. Let F be a problem class endowed with a probability measure P defined on a suitable σ-algebra. The mean error of a method A is defined by

$$\Delta(A, P) := \int_F \Delta(A, f) P(df),$$

if the function-wise error $\Delta(A, f)$ is a measurable function. (If not, one may apply the upper integral instead.) The mean error numbers for adaptive and non-adaptive methods are given by

$$e_n^{ad}(P) := \inf\{\Delta(A, P) :\ A \in \mathcal{A}_n^{ad}(F)\}, \tag{3.4}$$

$$e_n^{non}(P) := \inf\{\Delta(A, P) :\ A \in \mathcal{A}_n^{non}(F)\}.$$

A prominent model for $d = 1$ is to assume the Wiener measure P^* on $C[0, 1]$. It is easy to see that for $F = C[0, 1]$ the worst-case error of any method A is unlimited. We have only little knowledge concerning the mean error numbers. Ritter (1990), Theorem 4, proves the upper bound

$$\exists\, C > 0 \qquad e_n^{non}(P^*) \le C \cdot n^{-1/2}.$$

For a lower bound, Calvin (2004) uses sophisticated methods to show that the error cannot decrease exponentially:

$$\forall C_1, C_2 > 0 \quad \exists n \in \mathbb{N} \qquad e_n^{ad}(P^*) > C_1 \cdot \exp(-C_2 \cdot n).$$

We will reconsider the Wiener measure in Section 5.2.

Instead of the mean error one may also consider the mean cost, which is defined analogously. The simplex algorithm is a prominent example where the average case cost is much better than that of the worst case. The simplex algorithm applies to the following situation: We want to maximize $v^T x$ under the constraints $a_1^T x \le b^1, \ldots, a_m^T x \le b^m$, with $v, x, \ldots, a_m \in \mathbb{R}^d$ and $b \in \mathbb{R}^m$. If the problem has a solution, the simplex algorithm needs a finite amount of time to deliver a point x^* with $v^T x^* = \min_{x \in X} v^T x$ that fulfills the constraints, i.e., it delivers an exact solution. A worst-case analysis of many variants of the simplex algorithm shows that it has a cost that grows exponentially in m and d.

Borgwardt (1987) shows that the mean cost grows only polynomially in m and d. He uses a particular version of the simplex algorithm and a stochastic model where the vectors $a_1, \ldots, a_m$ and v are independent, equally distributed, and symmetrically distributed under rotation. This result for the mean cost is confirmed by the favorable outcomes in practice, see Borgwardt (1987), pp. 15 and pp. 31, for details.

Randomized algorithms, also called Monte Carlo methods, are used in situations where the worst-case error numbers $e_n^{ad}(F)$ only decrease slowly with increasing n. This is the case for many high-dimensional non-tractable problems. For a definition of randomized algorithms for the Unlimited Register Machine we refer to Novak (1995). For our purposes it is sufficient that any such method can be described as follows:

An adaptive Monte Carlo method (MCM) Q for the problem class F that uses at most n function calls can formally be described as a random variable on a suitable probability space with values in the set $\mathcal{A}_n^{ad}(F)$. If one only wants to consider non-adaptive MCM then the values are in $\mathcal{A}_n^{non}(F)$. Let $MC(\mathcal{A}_n^{ad}(F))$ and $MC(\mathcal{A}_n^{non}(F))$ be the sets of all such methods. Let $(\Omega, \mathcal{C}, P)$ be the probability space concerning Q. The function-wise error of Q is defined by

$$\Delta(Q, f) := \int_\Omega \Delta(Q(\omega), f)\, P(d\omega)$$

if the function $\omega \mapsto \Delta(Q(\omega), f)$ is measurable. (If not, one may again use the upper integral.) The error of Q is defined by

$$\Delta(Q, F) := \sup_{f \in F} \Delta(Q, f). \tag{3.5}$$

Analogously to the error numbers $e_n^{ad}(F)$ and $e_n^{non}(F)$, we define

$$\sigma_n^{ad}(F) := \inf_{Q \in MC(\mathcal{A}_n^{ad}(F))} \Delta(Q, F), \tag{3.6}$$

$$\sigma_n^{non}(F) := \inf_{Q \in MC(\mathcal{A}_n^{non}(F))} \Delta(Q, F). \tag{3.7}$$

For the Lipschitz classes F_L^d as above we have the result that Monte Carlo methods yield no essential advantage towards deterministic algorithms. More exactly, we have the asymptotic result

$$\forall \delta > 0 \qquad (1/2 - \delta)\, 2^{-1/d}\, e_n^{non}(F_L^d) \le \sigma_n^{ad}(F_L^d),$$

see Novak (1988), Prop. 1 in 2.2.6. There are further results for Monte Carlo methods in Nemirovski, Yudin (1983), in particular Chapter 1.6, and in Wasilkowski (1989).

Remark 3.2.1. It seems that the issue of universality in global optimization has not yet been considered. Still, there are classes where universal algorithms are known. The algorithms $(A_n)_{n\in\mathbb{N}}$ of Example 3.1.3 are universal for the classes $(F_L)_{L>0}$ since the non-adaptively chosen points $x^{(i)}$ are independent of the Lipschitz parameter L of the problem class F_L. Similar results hold for other Lipschitz classes.

The issue of universality was addressed for other numerical problems. For multivariate integration and approximation, a recent paper of Griebel, Woźnakowski (2005) delivers positive and negative results for the question of whether universal algorithms exist. Further papers concerning universal algorithms for integration are Babuska (1968), Motornyj (1974), Brass (1988), Petras (1996), Novak, Ritter (1996), and Novak, Ritter (1998). The definition of universality in these papers may slightly vary from ours.

Remark 3.2.2. We compare our results with those of two recent papers. Perevozchikov (1990) defines a class similar to $F_{L,D,\varrho}^d$. For

$$\widetilde{F}_{L,r,\varrho}^d := \{f : [0,1]^d \to \mathbb{R}, |f(x) - f(y)| \le L\|x - y\|,\ \lambda^d(A(\delta)) \le \delta^r \text{ for } \delta \le \varrho\}$$

with a norm $\|\cdot\|$ which is not necessarily $\|\cdot\|_\infty$ he develops an algorithm that yields the upper bound

$$e_n(\widetilde{F}_{L,r,\varrho}^d) \le \begin{cases} \mathcal{O}(n^{-1/d(1-r)}), & r < 1, \\ \mathcal{O}(e^{-n}), & r = 1. \end{cases}$$

This algorithm is similar to our universal algorithm. Lower bounds for $e_n(\widetilde{F}_{L,r,\varrho}^d)$ are not considered. Furthermore, the method of Perevozchikov assumes a fixed, i.e., known Lipschitz constant.

Like our universal algorithm, the method described in Jones et al. (1993) applies for Lipschitz optimization when the Lipschitz constant is not

known. They also have in common some constructional elements. In contrast to most other algorithms, they mix global and local search and do not apply a two step scheme, first to search globally and then to search locally. The algorithm of Jones et al. yields good results on some popular test functions. Error bounds are not proven.

3.3 Error bounds for $F^d_{L,D,\varrho}$

3.3.1 A lower error bound for adaptive methods

We establish a lower bound for $e^{ad}_n(F^d_{L,D,\varrho})$. The basic idea of the proof is to construct a set of fooling functions. This refers back to Bakhvalov (1959). We will later use this principle for non-adaptive and Monte Carlo methods, too.

Theorem 3.3.1. *Let $m \in \mathbb{N}$ with*

$$m \geq \max\left\{\frac{1}{2}L^2D^{2/d}, \frac{D^{2/d}L^2}{4\varrho}, \frac{D^{1/d}L}{\varrho^{1/2}}\right\}$$

and $n = m^d - 2$. Then

$$e^{ad}_n(F^d_{L,D,\varrho}) \geq \frac{L^2D^{2/d}}{4(n+2)^{2/d}}.$$

Proof. Let

$$I := \{\boldsymbol{i} : \boldsymbol{i} = (i_1, \ldots, i_d),\ i_k \in \{1, \ldots, m\},\ k = 1, \ldots, d\}, \tag{3.8}$$

$$l := \frac{D^{2/d}L}{2m}, \tag{3.9}$$

$$y^{\boldsymbol{i}} := \frac{l}{m} \cdot (i_1 - 1/2, \ldots, i_d - 1/2)^T.$$

The condition $m \geq \frac{1}{2}L^2D^{2/d}$ guarantees that $l \leq 1$ and $y^{\boldsymbol{i}} \in [0,1]^d$. For $\boldsymbol{i} \in I$ we define

$$f_{\boldsymbol{i}}(x) := \begin{cases} L\|x - y^{\boldsymbol{i}}\|_\infty, & \|x - y^{\boldsymbol{i}}\|_\infty \leq l/(2m), \\ \dfrac{Ll}{2m}, & x \in [0,l]^d \setminus \overline{B(y^{\boldsymbol{i}}, l/(2m))}, \\ \dfrac{Ll}{2m} + L(\|x\|_\infty - l), & \|x\|_\infty > l. \end{cases} \tag{3.10}$$

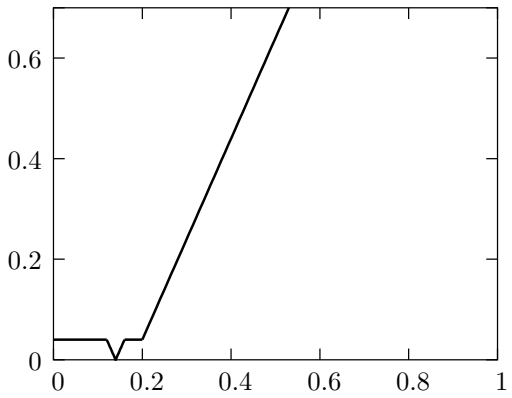


Figure 3.1: $f_{\boldsymbol{i}}$ for $d = 1$, $L = 2$, $D = 1$, $m = 5$, $\boldsymbol{i} = 4$.

We show $f_{\boldsymbol{i}} \in F^d_{L,D,\varrho}$. Obviously, $f_{\boldsymbol{i}}$ is Lipschitz with constant L. For the level sets we have

$$\lambda^d(A(f_{\boldsymbol{i}}, 0)) = 0, \qquad \lambda^d(A(f_{\boldsymbol{i}}, Ll/(2m))) = l^d = D(Ll/(2m))^{d/2}.$$

Since $m \geq D^{2/d}L^2/(4\varrho)$ we know that $\varrho \geq Ll/(2m)$ and consequently, that $\lambda^d(A(f_{\boldsymbol{i}}, \varrho)) \leq (l + \varrho/L - l/(2m))^d$. So,

$$\text{if } l + \varrho/L - l/(2m) \leq D^{1/d}\varrho^{1/2} \qquad \text{then } \lambda^d(A(f_{\boldsymbol{i}}, \varrho)) \leq D\varrho^{d/2}.$$

Since $\varrho^{1/2} < \frac{1}{2}D^{1/d}L$ and since $l - l/(2m) \leq \frac{1}{2}D^{1/d}\varrho^{1/2}$ is guaranteed by $m \geq D^{1/d}L/\varrho^{1/2}$ we conclude $\lambda^d(A(f_{\boldsymbol{i}}, \varrho)) \leq D\varrho^{d/2}$. Application of Lemma 1.2.1 delivers $f_{\boldsymbol{i}} \in F^d_{L,D,\varrho}$.

Let $A_n = \phi \circ N$ be an algorithm that uses at most n function calls. Then there exist (at least) two different $\boldsymbol{i}, \boldsymbol{j} \in I$ such that

$$N(f_{\boldsymbol{i}}) = N(f_{\boldsymbol{j}}).$$

No matter where the algorithm chooses $x_* = \phi \circ N(f_{\boldsymbol{i}}) = \phi \circ N(f_{\boldsymbol{j}})$, we will have $f_{\boldsymbol{i}}(x_*) \geq Ll/(2m)$ or $f_{\boldsymbol{j}}(x_*) \geq Ll/(2m)$, but $\min f_{\boldsymbol{i}} = \min f_{\boldsymbol{j}} = 0$. Consequently,

$$\Delta(A_n, F^d_{L,D,\varrho}) \geq \frac{l\,L}{2m} = \frac{L^2 D^{2/d}}{4(n+2)^{2/d}}.$$

Since A_n was arbitrary we have the same bound for $e^{ad}_n(F^d_{L,D,\varrho})$. □

3.3.2 Optimality and universality of the algorithms of Chapter 2

Corollary 3.3.2. *The algorithms $(S(L,k))_{k\in\mathbb{N}}$ and $(Z(h,k))_{k\in\mathbb{N}}$ have the optimal rate of convergence. In particular, $(Z(h,k))_{k\in\mathbb{N}}$ is universal for $\mathbb{F}^d$.*

Proof. We define $m_0 := \lceil L^2 D^{2/d}(LD^{2/d} - \varrho)^{-1} \rceil$ and $n(m) := m^d - 2$. Let

$$n \geq \max\{n(m_0), 3, ((9/8)^{d/2} - 1)^{-1}\}.$$

We choose m such that $n(m-1) + 1 \leq n \leq n(m)$. Then

$$e_n(F^d_{L,D,\varrho}) \geq e_{n(m)}(F^d_{L,D,\varrho}) \geq \frac{D^{2/d}L^2}{4m^2} \geq \frac{D^{2/d}L^2}{9\, n^{2/d}}.$$

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

$$n_k := \lfloor DL^{d/2} 2^{-d/2} 3^{(k+1)d/2+1} \rfloor.$$

From Theorem 2.1.2 we know that $S(L,k)$ uses at most n_k oracle calls and delivers an error level

$$\Delta(S(L,k), F^d_{L,D,\varrho}) \leq \varepsilon_{L,k} \leq D^{2/d} L^2 2^{-2} 3^{2+2/d} n_k^{-2/d}.$$

This means that the algorithms $(S(L,k))_{k\in\mathbb{N}}$ have the optimal rate of convergence.

Let $m \in \mathbb{N}$. For $k \geq h(m-1)+1$ we have $\Delta(Z(h,k), F^d_{L(m),D,\varrho}) \leq \varepsilon_{L(m),h,k}$ and

$$\begin{aligned} & \operatorname{cost}(Z(h,k), F^d_{L(m),D,\varrho}) \\ \leq\ & \alpha(L(m),d,h,\varrho) + \beta(L(m),d,h)\, \varepsilon^{-d/h}_{L(m),h,k} + \gamma(L(m),D,h,d)\, \varepsilon^{-2/d}_{L(m),h,k}, \end{aligned}$$

with constants α, β, γ that can be determined with Theorem 2.2.2. Recall $h \geq 3$. For $\varepsilon_{L(m),h,k} \leq 1$ we have

$$\operatorname{cost}(Z(h,k), F^d_{L(m),D,\varrho}) \leq (\alpha + \beta + \gamma)\, \varepsilon^{-2/d}_{L(m),h,k}.$$

Proceeding as for $S(L,k)_{k\in\mathbb{N}}$ we see that $(Z(h,k))_{k\in\mathbb{N}}$ have the optimal rate of convergence. This means in particular that $(Z(h,k))_{k\in\mathbb{N}}$ are universal for $\mathbb{F}^d$. □

3.3.3 A lower error bound for non-adaptive methods

We turn to non-adaptive methods.

Lemma 3.3.3. *Let* $m \geq \max\{(\frac{1}{4}D^{2/d}L)^{-1}, (2D^{1/d}\varrho^{1/2} - 4\varrho/L)^{-1}, L/(4\varrho)\}$. *For* $n = m^d - 1$ *we have*

$$e^{non}_n(F^d_{L,D,\varrho}) \geq \frac{L}{4(n+1)^{1/d}}.$$

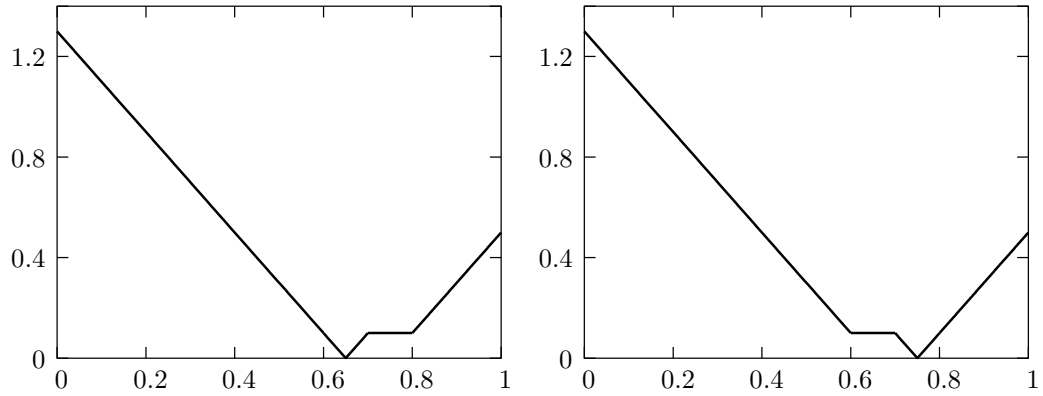


Figure 3.2: $f_{\boldsymbol{i},1}$ and $f_{\boldsymbol{i},2}$ for $d = 1$, $L = 2$, $m = 5$, $\boldsymbol{i} = 4$

Proof. Let I as in (3.8). For $\boldsymbol{i} \in I$ define

$$\begin{aligned} x^{\boldsymbol{i}} &:= \tfrac{1}{m}(i_1 - \tfrac{1}{2}, \ldots, i_d - \tfrac{1}{2})^T, \\ x^{\boldsymbol{i},1} &:= \tfrac{1}{m}(i_1 - \tfrac{3}{4}, i_2 - \tfrac{1}{2}, \ldots, i_d - \tfrac{1}{2})^T, \\ x^{\boldsymbol{i},2} &:= \tfrac{1}{2}(i_1 - \tfrac{1}{4}, i_2 - \tfrac{1}{2}, \ldots, i_d - \tfrac{1}{2})^T, \end{aligned} \tag{3.11}$$

$$f_{\boldsymbol{i},1}(x) := \begin{cases} L\,\|x - x^{\boldsymbol{i},1}\|_\infty, & x \in \overline{B(x^{\boldsymbol{i},1}, 1/(4m))}, \\ \dfrac{L}{4m}, & x \in \overline{B(x^{\boldsymbol{i}}, 1/(2m))} \setminus \overline{B(x^{\boldsymbol{i},1}, 1/(4m))}, \\ L\,\|x - x^{\boldsymbol{i}}\|_\infty - \dfrac{L}{4m}, & x \in [0,1]^d \setminus \overline{B(x^{\boldsymbol{i}}, 1/(2m))}, \end{cases}$$

$$f_{\boldsymbol{i},2}(x) := \begin{cases} L\,\|x - x^{\boldsymbol{i},2}\|_\infty, & x \in \overline{B(x^{\boldsymbol{i},2}, 1/(4m))}, \\ \dfrac{L}{4m}, & x \in \overline{B(x^{\boldsymbol{i}}, 1/(2m))} \setminus \overline{B(x^{\boldsymbol{i},2}, 1/(4m))}, \\ L\,\|x - x^{\boldsymbol{i}}\|_\infty - \dfrac{L}{4m}, & x \in [0,1]^d \setminus \overline{B(x^{\boldsymbol{i}}, 1/(2m))}. \end{cases}$$

We show

$$\forall \boldsymbol{i} \in I \qquad f_{\boldsymbol{i},1}, f_{\boldsymbol{i},2} \in F^d_{L,D,\varrho}.$$

With the above assumption we have $L/(4m) \le \varrho$. Since $m \ge (\frac{1}{4}D^{2/d}L)^{-1}$ it follows that

$$\lambda^d(A(f_{\boldsymbol{i},1}, L/(4m))) \le D(L/(4m))^{d/2},$$

and since $m \ge (2D^{1/d}\varrho^{1/2} - 4\varrho/L)^{-1}$ we have

$$\lambda^d(A(f_{\boldsymbol{i},1}, \varrho)) \le D\varrho^{d/2}.$$

Application of Proposition 1.2.1 on the piece-wise linear function g with nodes $\delta = 0, L/(4m), \varrho$ and values $g(\delta) = (\delta^d(A(f_{\boldsymbol{i},1}, \delta))^{1/d}$ yields

$$\forall\, 0 \le \delta \le \varrho \qquad \lambda^d(A(f_{\boldsymbol{i},1}, \delta)) \le D\,\delta^{d/2}.$$

Consequently,

$$f_{\boldsymbol{i},1} \in F^d_{L,D,\varrho}.$$

Analogously, we show

$$f_{\boldsymbol{i},2} \in F^d_{L,D,\varrho}.$$

For an arbitrary method A_n using n oracle calls, we have that in (at least) one cube

$$Q_{\boldsymbol{i}} := \{x \in \mathcal{D} :\, (\boldsymbol{i} - (1/m, \ldots, 1/m) < x < \boldsymbol{i}/m\}, \qquad \boldsymbol{i} \in I,$$

there is no evaluation. For such an index $\boldsymbol{i}$, we have

$$N(f_{\boldsymbol{i},1}) = N(f_{\boldsymbol{i},2}).$$

Consequently,

$$\Delta(A_n, F^d_{L,D,\varrho}) \ge \frac{L}{4m} = \frac{L}{4(n+1)^{1/d}}.$$

Since A_n was arbitrary we have the same bound for $e^{non}_n(F^d_{L,D,\varrho})$. □

3.3.4 A lower error bound for randomized methods

Finally, we turn to Monte Carlo methods (MCM). See Section 3.2 for the definition of a MCM, its error, and the error numbers $\sigma^{ad}_n(F)$.

We obtain a lower bound for $\sigma^{ad}_n(F^d_{L,D,\varrho})$ with a method similar to that in Novak (1988), Section 2.1.9. The following lemma uses the mean error numbers $e^{ad}_n(P)$ for a suitable probability measure P to prove a lower bound for $\sigma^{ad}_n(F)$:

Lemma 3.3.4. *Let P be a Borel measure on F of the sort $P = \sum_{i=1}^m c_i \delta_{f_i}$ with $c_i \ge 0$ and $\sum_{i=1}^m c_i = 1$. Furthermore, let $f_i \in F$ for $i = 1, \ldots, m$. Then we have*

$$e^{ad}_n(P) \le \sigma^{ad}_n(F).$$

Proof. See Novak (1988), Proposition 2.1.9. □

Lemma 3.3.5. *Let m be as in Theorem 3.3.1, even, and $n = m^d/2$. Then*

$$\sigma^{ad}_n(F^d_{L,D,\varrho}) \ge \frac{n-1}{n} \frac{L^2 D^{2/d}}{8(2n)^{2/d}}.$$

Proof. Let l as in (3.9), and I and $f_{\boldsymbol{i}}$ as in (3.8) and (3.10). Define

$$g(x) := \begin{cases} \dfrac{Ll}{2m}, & \|x\|_\infty \le l, \\ \dfrac{Ll}{2m} + L \max\limits_{1\le j\le d}(x_j - l), & \|x\|_\infty > l. \end{cases}$$

Let $F_I := \{f_{\boldsymbol{i}}, \boldsymbol{i} \in I\}$. We know from Theorem 3.3.1 that $F_I \subset F^d_{L,D,\varrho}$. Let $A = \phi \circ N$ be an (adaptive) deterministic algorithm using at most n oracle calls. For at least n different $\boldsymbol{i} \in I$ we have $N(f_{\boldsymbol{i}}) = N(g)$. Consequently,

$$\sum_{f\in F_I} (\inf f - f(A(f))) \ge \frac{(n-1)Ll}{2m} = \frac{(n-1)\,L^2 D^{2/d}}{4m^2}.$$

The proof is complete with Lemma 3.3.4 using the uniform distribution on F_I. □

3.3.5 Conclusion

We obtained several results in this chapter:

- Adaptive algorithms: We know from Theorem 3.3.1 that for n large enough, $n \ge n(L, D, \varrho, d)$,

  $$e^{ad}_n(F^d_{L,D,\varrho}) \ge \frac{L^2 D^{2/d}}{4(n+2)^{2/d}}.$$

  Furthermore, we conclude from Theorem 2.1.2 *(c)* that

  $$e^{ad}_{n_k}(F^d_{L,D,\varrho}) \le 2^{-2} 3^{2+2/d} L^2 D^{2/d} n^{-2/d},$$

  for $k \ge \max\{j(L,\varrho), j(L,D,\varrho,d)$ and $n_k := \lfloor DL^{d/2} 2^{-d/2} 3^{(k+1)d/2+1} \rfloor$. So,

  $$e^{ad}_n(F^d_{L,D,\varrho}) \asymp \frac{L^2 D^{2/d}}{n^{2/d}}. \tag{3.12}$$

- Non-adaptive algorithms: From Lemma 3.3.3 we know that for n large enough, $n \ge n(L, D, \varrho, d)$,

  $$e^{non}_n(F^d_{L,D,\varrho}) \ge L\,(4n+1)^{-1/d}.$$

On the other hand, $F^d_{L,D,\varrho} \subset F^d_L$ with F^d_L as defined in Section 3.2. With Novak (1988), Proposition 1.3.6, we have

$$e^{non}_n(F^d_{L,D,\varrho}) \leq e^{non}_n(F^d_L) = \mathcal{O}(L\, n^{-1/d})$$

and conclude that

$$e^{non}_n(F^d_{L,D,\varrho}) \asymp \frac{L}{n^{1/d}}. \tag{3.13}$$

- Randomized algorithms: From Lemma 3.3.5 we know that for n large enough, $n \geq n(L, D, \varrho, d)$,

$$\sigma^{ad}(F^d_{L,D,\varrho}) \geq \frac{n-1}{n} \frac{L^2 D^{/d}}{8(2n)^{2/d}}.$$

On the other hand, we have

$$\sigma^{ad}_n(F^d_{L,D,\varrho}) \leq e^{ad}_n(F^d_{L,D,\varrho})$$

since any deterministic algorithm can be seen as a MCM with Dirac measure. So,

$$\sigma^{ad}_n(F^d_{L,D,\varrho}) \asymp \frac{L^2 D^{2/d}}{n^{2/d}}. \tag{3.14}$$

Since non-adaptive MCM can never be better than adaptive ones we neglected to estimate the error numbers $\sigma^{non}_n(F^d_{L,D,\varrho})$.

So, for our problem class $F^d_{L,D,\varrho}$ adaptiveness is essential for optimal algorithms and yields a quadratic speed-up while randomization cannot yield any further essential advantage.

Our second method is universal for $\mathbb{F}^d$. Our first method has the optimal rate of convergence in case that the Lipschitz constant is known.

Chapter 4

Numerical experiments

We will now dicuss some questions that have no theoretical consequence but that are of practical interest.

- We give advice for a good choice of h. This parameter of the universal algorithm defines the steepness of the diagonal scheme and influences the ratio between global and local search.

- The definitions of the algorithms $S(L, \cdot)$ and $Z(\cdot)$ are not precise in one aspect. Consider once more step(L, j) for $j \geq 2$, as defined in Figure 2.1. Here, the question is left open in which order the algorithm considers the elements of $N_{L,j-1}$, and those of $Y(j)$. This question was unimportant for proving error bounds. However, a deliberately chosen order may yield better results in the nonasymptotic behavior.

- We again refer to the design of step(L, j), $j \geq 2$. The question whether an element of $N_{L,j-1}$ and its neighboring mesh points $(x+y, f(x+y))$, $y \in Y(j)$, will also belong to $N_{L,j}$ depends on the particular order in which the points in $N_{L,j-1}$ are considered. This is a consequence of the fact that f_* may vary during step(L, j). We introduce a version of $Z(h, k)$ such that the cost and error bounds of Chapter 2 still hold, but the set $N_{L,j}$ is independent of the order in which the points of $N_{L,j-1}$ are considered.

These considerations led us to develop different versions of the algorithm $Z(h, k)$. We will introduce them in the next section. We then apply them to a set of test functions presented in Section 4.2. The results are given in Section 4.3. We use them to recommend a particular version. For illustration, we plot the points chosen by this version for three of the test functions.

4.1 Algorithm versions

We want to apply different values of h. We test

$$h = 2, 3, 4, 5, 8, 12. \tag{4.1}$$

Note that $h = 2$ is not covered by the results of Theorem 2.2.2. Furthermore, we introduce three different design features the algorithm may or may not have and that all concern only the design of step(L, j):

- We say that the algorithm has the feature $F1$ if it performs step(L, j) in two sub-steps as follows: Divide the set $Y(j)$ into the 2 disjoint subsets

$$Y(j,1) := \bigcup_{i=1}^{d} \{e^{(i)}, -e^{(i)}\}, \quad Y(j,2) := Y(j) \setminus Y(j,1).$$

  Replace step(L, j) by step(L, j_1) and step(L, j_2). Step(L,j_1) is the same as step(L, j) with the difference that it uses $Y(j,1)$ instead of $Y(j)$. Analogously, we define step(L, j_2).
  This way, the new points evaluated in step(L, j) are scattered well not only after the end of step(L, j), i.e., after approximately $|N_{L,j-1}| \cdot 3^d$ points, but also after approximately $|N_{L,j-1}| \cdot 2d$ points.

- We say the algorithm has the feature $F2$ if it sorts the elements of $N_{L,j-1}$ as follows: Let $q \in \mathbb{N}$ be a fixed number. Without loss of generality assume $q \leq |N_{L,j-1}| =: p$. We write the elements of $N_{L,j-1}$ in a list

$$\widetilde{N}_{L,j-1} := [\,(\widetilde{x}_1, f(\widetilde{x}_1)), \ldots, (\widetilde{x}_p, f(\widetilde{x}_p))\,]$$

  such that

$$f(\widetilde{x}_i) \leq f(\widetilde{x}_{i+1}), \quad 1 \leq i \leq q-1, \qquad f(\widetilde{x}_q) \leq f(\widetilde{x}_j), \quad q < j \leq p.$$

  The algorithm shall examine the elements of $N_{L,j-1}$ in such an order. This way, we examine first those q points that, because of their function value, seem to be most promising. We only sort the first q best points instead of sorting them all, in order not to get an additional logarithmic factor for the computational cost. We choose $q := 2 \cdot d$.

- Recall the test $(*)$ of step(L, j), $j \geq 2$. We say the algorithm has the feature $F3$ if it uses an additional variable $\widetilde{f}_*$. This variable assumes the value of f_* at the very beginning of step(L, j) and will not be

changed during the whole step. The algorithm performs the test $(*)$ with $\widetilde{f}_*$ instead of f_*.
This way, a point $N_{L,j-1}$ passes the test $(*)$ independently of the order in which the points in $N_{L,j-1}$ are considered.

Note that two or all three of the features $F1$, $F2$ and $F3$ can be combined. If $F1$ and $F3$ are combined, we want to assume that $\widetilde{f}_*$ is not updated before step(L, j_2). If F_2 and F_3 are combined we will sort the elements of $N_{L,j-1}$ only before step(L, j_1). It is easily checked that all features and their combinations have no effect on neither the error nor the cost bound proved in Chapter 2.

We will test all 8 resulting versions, each of them with the 6 different values of h as stated in (4.1). Altogether, we test 48 versions.

4.2 Test functions

We want to test these 48 versions each on the following test functions, which can all be found in TÖRN, ŽILINSKAS (1989).

***d*=1**

Shekel g_6: $[0, 10] \to \mathbb{R}$,

$$x \mapsto -\sum_{i=1}^{6} \frac{1}{(k_i(x - a_i))^2 + c_i},$$

with coefficients a_i, k_i, c_i uniformly distributed: $a_i \sim U(0, 10)$, $k_i \sim U(1, 3)$, $c_i \sim U(0.1, 0.3)$. We choose the values given in TÖRN, ŽILINSKAS (1989), Table 8.2.

***d*=2**

C: $[-5, 5]^2 \to \mathbb{R}$,

$$x \mapsto 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4.$$

BR: $[-5, 10] \times [0, 15] \to \mathbb{R}$,

$$x \mapsto \left[x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right]^2 + 10\left[1 - \frac{1}{8\pi}\right]\cos x_1 + 10.$$

GP: $[-2,2]^2 \to \mathbb{R}$,

$$\begin{aligned} x \;\mapsto\; & \left[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)\right]\cdot \\ & \left[30+(2x_1-3x_2)^2(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)\right]. \end{aligned}$$

G2: $[-100,100/\sqrt{2}]^2 \to \mathbb{R}$,

$$x \mapsto \frac{1}{200}(x_1^2+x_2^2)-\cos(x_1)\cdot\cos(x_2 7\sqrt{2})+1.$$

The domain given for G2 in Törn, Žilinskas (1989) is $[-100,100]^2$. We changed it because otherwise, the first point evaluated by our algorithm would already be the global minimizer.

R: $[-1,1/\sqrt{2}]^2 \to \mathbb{R}$,

$$x \mapsto x_1^2+x_2^2-\cos(18x_1)-\cos(18x_2).$$

We changed the domain $[-1,1]^2$ given in Törn, Žilinskas (1989) for the same reason.

d=3

H3: $[0,1]^3 \to \mathbb{R}$,

$$x \mapsto -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{3}\alpha_{i,j}(x_j-p_{i,j})^2\right],$$

with coefficients $\alpha_{i,j}$ and $p_{i,j}$ that can be found in Törn, Žilinskas (1989), p. 185.

d=4

S5, S7, S10: $[0,10]^4 \to \mathbb{R}$,

$$x \mapsto -\sum_{i=1}^{m}\frac{1}{(x-A(i))(x-A(i))^T+c_i}.$$

with vectors $A(i)$ and coefficients c_i that can again be found in Törn, Žilinskas (1989), p. 184.

St: $[-1,1]^4 \to \mathbb{R}$,

$$x \mapsto \left[\left[\sum_{i=1}^{7}\sum_{j=1}^{7} A_{i,j}a_{i,j}(x) + B_{i,j}b_{i,j}(x)\right]^2 + \left[\sum_{i=1}^{7}\sum_{j=1}^{7} C_{i,j}c_{i,j}(x) + D_{i,j}d_{i,j}(x)\right]^2\right]^{1/2},$$

where

$$\begin{aligned} a_{i,j}(x) &:= \sin(i\pi x_1)\sin(j\pi x_2), \quad b_{i,j}(x) := \cos(i\pi x_3)\cos(j\pi x_4), \\ c_{i,j}(x) &:= \sin(i\pi x_3)\sin(j\pi x_4), \quad d_{i,j}(x) := \cos(i\pi x_1)\cos(j\pi x_2), \end{aligned}$$

and matrices A, B, C, D, whose entries are independently $U(-1,1)$ sampled. Since this is only of minor importance, we do not state these 196 random numbers. This function was originally defined for $d = 2$.

d=6

H6: $[0,1]^6 \to \mathbb{R}$,

$$x \mapsto -\sum_{i=1}^{4} c_i \exp\left[-\sum_{j=1}^{6} \alpha_{i,j}(x_j - p_{i,j})^2\right],$$

where the coefficients $\alpha_{i,j}$ and $p_{i,j}$ are as in Törn, Žilinskas (1989), p. 185.

d=10

G10: $[-600, 600/\sqrt{2}]^{10} \to \mathbb{R}$,

$$x \mapsto \sum_{i=1}^{10} \frac{1}{4000}x_i^2 - \prod_{i=1}^{10} \cos(x_i/\sqrt{i}) + 1.$$

For the same reason as above, we changed the domain $[-600, 600]^{10}$ given in Törn, Žilinskas (1989).

If necessary, we scale the functions such that the domain is $[0,1]^d$.

h / Feature	2	3	4	5	8	12
no feature	670	666	697	684	749	841
$F1$	701	640	784	767	849	977
$F2$	423	266	283	293	313	394
$F3$	663	742	744	667	740	784
$F1$, $F2$	609	572	568	620	698	807
$F1$, $F3$	658	553	631	743	674	748
$F2$, $F3$	495	407	442	475	483	470
$F1$, $F2$, $F3$	512	513	452	516	508	552

Table 4.1: Results for the summed up rank numbers

4.3 Results

For each function, we compare the values of f_* after 250, 1,500 and 10,000 oracle calls and rank the 48 versions. This results in 39 different rankings. For each algorithm version we sum up all its rank numbers. We state the results in Table 4.1.

The results make it easy to give advice: The version with $h = 3$ and $F2$ has the least value. Furthermore, the result for $F2$ is best for any value of h. Also, the column for $h = 3$ has 5 times the best and 2 times the second best value.

We suggest to use $Z(3, \cdot)$ with Feature $F2$.

We give the results for this version in Table 4.2 and plots of the points it chooses for the functions Shekel g_6, C, and BR.

d	fct.	$n = 250$	$n = 1,500$	$n = 10,000$	glob. min.
1	g_6	-10.05863092	-10.05869313	-10.05869313	
2	C	-0.990514469	-1.031480295	-1.031625022	-1.0316285
	BR	0.397935486	0.397887448	0.397887387	$\approx$ 0.398
	GP	3.064984070	3.000811378	3.000010033	3
	G2	0.0088591487	0.0006864435	0.0000009992	0
	R	-1.031206852	-1.871066950	-1.999970549	-2
3	H3	-3.862583278	-3.862698023	-3.862724551	-3.86
4	S5	-1.631190022	-10.15319696	-10.15319696	$\approx$ -10
	S7	-1.945297979	-10.40282047	-10.40293718	$\approx$ -10
	S10	-7.216953325	-10.53629007	-10.53640678	$\approx$ -10
6	H6	-2.035737075	-3.168159945	-3.203076797	-3.32
10	G10	20.30238203	20.30238203	20.30238203	0

Table 4.2: The results for the version with $h = 3$ and feature $F2$. The right hand side column gives the minimum value according to Törn, Žilinskas (1989). The minimum of g_6 given there is -13.9223449 and seems to be wrong. It should be attained at $x_0 := 4.8555654$, but $g_6(x_0) = -8.937$. For St, the minimum depends on the random numbers. In consequence, we have no value which we could compare to ours. For $G10$ the approximation of our algorithm is not too good even after 10,000 points. This is not surprising since here, $d = 10$. The first improvement is after $n = 59114$ points: $f_* = 9.730$. After $n = 67789$ points we have $f_* = 2.662$.

Plots for Shekel g_6

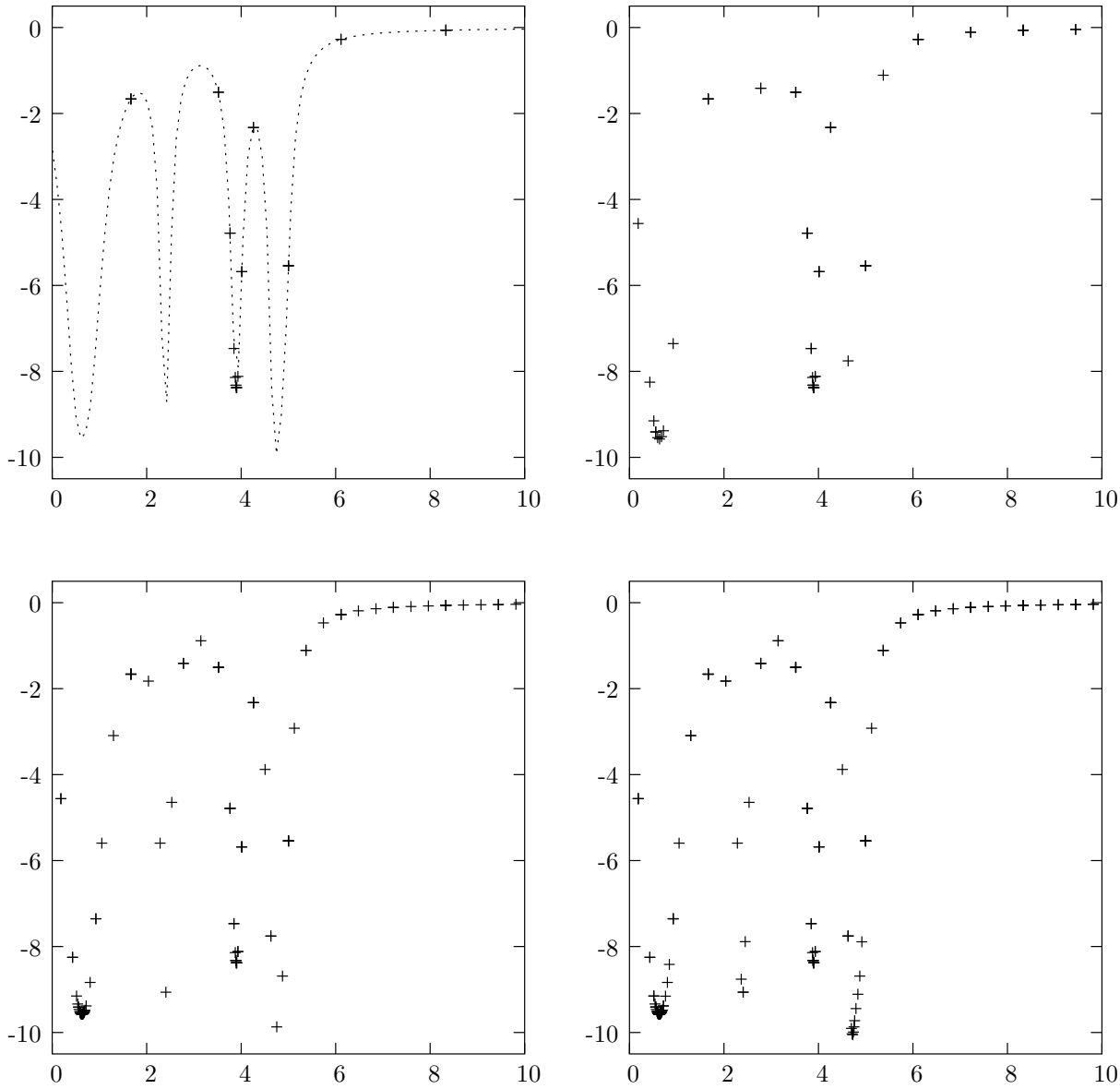


Figure 4.1: The points chosen by $Z(3,8)$, $Z(3,12)$, $Z(3,14)$, $Z(3,16)$ for g_6.

function	algorithm	# points	f_*
Shekel g_6	$Z(3,8)$	27	-8.383952678
	$Z(3,12)$	60	-9.575211370
	$Z(3,14)$	115	-9.871841283
	$Z(3,16)$	162	-10.05469879

Plots for C

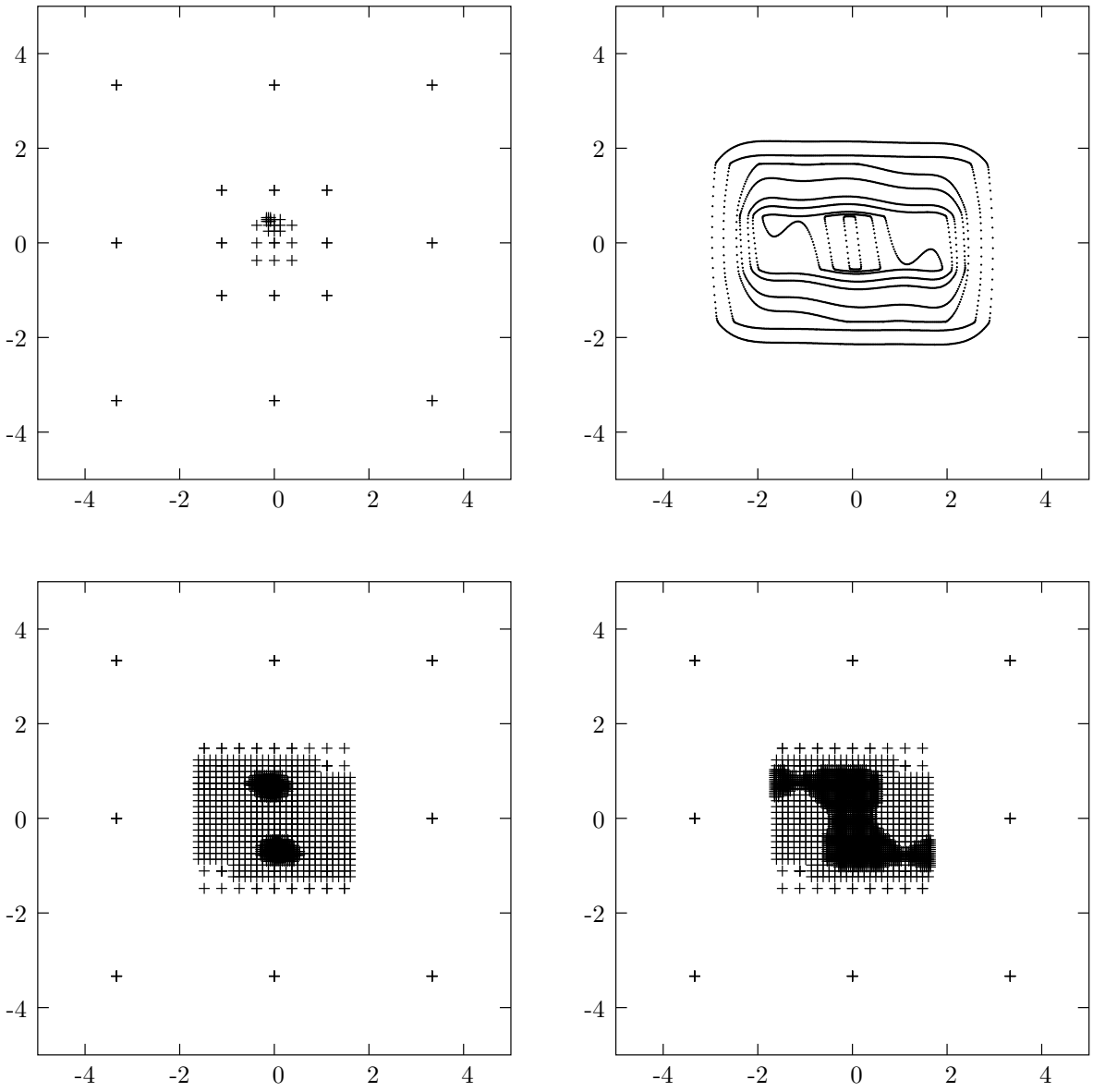


Figure 4.2: The points chosen by $Z(3,6)$, $Z(3,14)$ $Z(3,15)$ for the function C and a contour plot of C with altitudes $-0.8, 0, 1, 4, 7, 14, 20, 50, 100$.

function	algorithm	# points	f_*
C	$Z(3,6)$	58	-.8341946919
	$Z(3,14)$	5277	-1.031615969
	$Z(3,15)$	15741	-1.031625022

Plots for BR

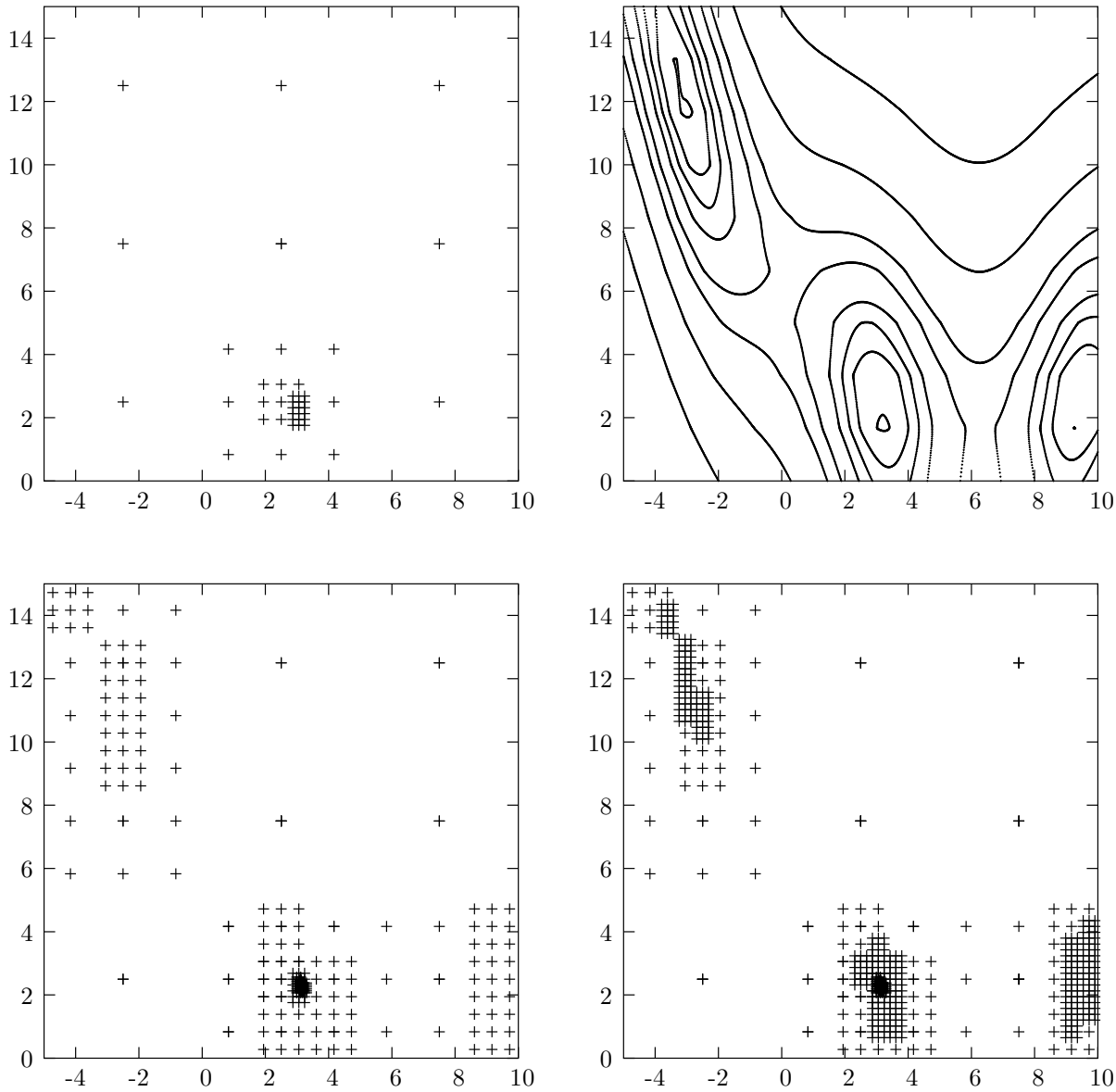


Figure 4.3: The points chosen by $Z(3,5)$, $Z(3,10)$, $Z(3,11)$ for BR, and a contour plot of BR with altitudes 1, 4, 7, 11, 19, 27, 50, 100.

function	algorithm	# points	f_*
BR	$Z(3,5)$	42	0.434202095
	$Z(3,10)$	3988	0.397887448
	$Z(3,11)$	11788	0.3978878368

Chapter 5

Applications

We present two applications of the fine-tuned version of our universal algorithm.

5.1 Banach-Mazur distance

Let E, F, G be n-dimensional normed vector spaces. The Banach-Mazur distance of E and F is defined by

$$d(E,F) := \inf\{\|T\| \cdot \|T^{-1}\|,\ T : E \to F \text{ linear isomorphism}\}. \tag{5.1}$$

It measures how far the unit ball of E is from an affine image of the unit ball of F. It also shows how much numerical parameters of E can differ from the corresponding ones of F. We have

$$d(E,G) \leq d(E,F) \cdot d(F,G).$$

Furthermore, $\log d(\cdot,\cdot)$ is a semi-metric on the set of all n-dimensional normed vector spaces with $\log d(E,F) = 0$ if E and F are isometric. We say F belongs to the class $[E]$ if $\log d(E,F) = 0$.

Let M_n be the set of all classes of n-dimensional normed vector spaces. M_n is compact for all $n \in \mathbb{N}$ and is called the Minkowski compactum, see Tomczak-Jaegermann (1989), Chapter 9, for this and also to learn more about the Banach-Mazur distance. For

$$\operatorname{diam} M_n := \sup_{[E],[F] \in M_n} d(E,F)$$

we know that for an unknown constant c independent of the dimension n

$$c \cdot n \leq \operatorname{diam} M_n \leq n.$$

The first inequality is due to JOHN (1948), the second one is a corollary of GLUSKIN (1981). For dimensions $n > 2$ the exact value of diam M_n is not known.

In our application we consider the case $n = 3$ and the particular spaces l_1^3 and l_∞^3 being $\mathbb{R}^3$ endowed with the norms

$$\| x \,|\, l_1^3 \| := \sum_{i=1}^{3} |x_i| \qquad \text{and} \qquad \| x \,|\, l_\infty^3 \| := \max_{1 \le i \le 3} |x_i|,$$

respectively. For the linear isomorphism

$$T : l_1^3 \to l_\infty^3, \qquad x \mapsto \begin{pmatrix} 1 & -1/3 & 1 \\ 1 & 1 & -1/3 \\ -1/3 & 1 & 1 \end{pmatrix} x,$$

we know $\|T\| \cdot \|T^{-1}\| = \frac{9}{5}$. The conjecture is that this value is already best possible, i.e., that $d(l_1^3, l_\infty^3) = \frac{9}{5}$. This would imply

$$\text{diam } M_3 \ge \frac{9}{5}.$$

We want to test this hypothesis by applying our universal algorithm. As a first step, we reformulate the problem:

$$\text{“Minimize } \|T\| \cdot \|T^{-1}\| \quad \text{for } T : l_1^3 \to l_\infty^3 \text{ linear isomorphism”} \tag{5.2}$$

to an optimization problem with a 6-variate objective defined on the domain $[-1, 1]^6$.

For the rest of this section, T will denote a linear isomorphism from l_1^3 to l_∞^3. Without loss of generality, we assume

$$\|T\| = 1.$$

Let $(t_{i,j})_{i,j=1}^3$ be the matrix associated to T in the usual way. In our special situation $T : l_1^3 \to l_\infty^3$, we know $\| T \| = \max_{i,j=1,\dots,3} |t_{i,j}|$. A simple argument shows that we can restrict our search to isomorphisms that can be identified with a matrix as follows:

$$T \sim \begin{pmatrix} 1 & t_{1,2} & t_{1,3} \\ t_{2,1} & 1 & t_{2,3} \\ t_{3,1} & t_{3,2} & 1 \end{pmatrix}, \qquad t_{i,j} \in [-1, 1]. \tag{5.3}$$

Let $e^{(1)}, e^{(2)}, e^{(3)}$ be the unit vectors in $\mathbb{R}^3$. Then

$$T \sim \left(Te^{(1)} \middle| Te^{(2)} \middle| Te^{(3)}\right) =: \left(y^{(1)} \middle| y^{(2)} \middle| y^{(3)}\right).$$

For $x \in \mathbb{R}^3$ we define $\lambda_1^T(x), \lambda_2^T(x), \lambda_3^T(x)$ to be the (unique) solutions of

$$\sum_{i=1}^{3} \lambda_i^T(x) y^{(i)} = x \quad \Leftrightarrow \quad \sum_{i=1}^{3} \lambda_i^T(x) e^{(i)} = T^{-1}x.$$

We conclude that

$$\| T^{-1}x \,|\, l_1^3 \| = \sum_{i=1}^{3} |\lambda_i^T(x)|.$$

One verifies easily that

$$\|T^{-1}\| = \sup_{x \neq 0} \frac{\|T^{-1}x \,|\, l_1^3\|}{\| x \,|\, l_\infty^3\|} = \max_{1 \leq j \leq 4} \sum_{i=1}^{3} |\lambda_i^T(v^{(j)})|,$$

with

$$v^{(1)} := \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, v^{(2)} := \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix}, v^{(3)} := \begin{pmatrix} 1 \\ -1 \\ 1 \end{pmatrix}, v^{(4)} := \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}. \quad (5.4)$$

Then,

$$d(l_1^3, l_\infty^3) = \inf_{\| T \|=1} \max_{1 \leq j \leq 4} \sum_{i=1}^{3} |\lambda_i^T(v^{(j)})|.$$

The coefficients $\lambda_i^T(x)$ can be calculated by

$$\lambda_1^T(x) = \frac{\det\left(x \,|\, y^{(2)} \,|\, y^{(3)}\right)}{\det\left(y^{(1)} \,|\, y^{(2)} \,|\, y^{(3)}\right)}, \qquad \lambda_2^T(x) = \frac{\det\left(y^{(1)} \,|\, x \,|\, y^{(3)}\right)}{\det\left(y^{(1)} \,|\, y^{(2)} \,|\, y^{(3)}\right)},$$

$$\lambda_3^T(x) = \frac{\det\left(y^{(1)} \,|\, y^{(2)} \,|\, x\right)}{\det\left(y^{(1)} \,|\, y^{(2)} \,|\, y^{(3)}\right)}.$$

Since every T we want to consider can be identified with one point in $[-1,1]^6$, as done in (5.3), we reformulate (5.2) to:

"Minimize the objective

$$x \mapsto \max_{1 \leq j \leq 4} \frac{\det T_1(x, v^{(j)}) + \det T_2(x, v^{(j)}) + \det T_3(x, v^{(j)})}{\det T(x)}, \quad x \in [-1,1]^6 \setminus \Omega_0,$$

with

$$T_1(x, v^{(j)}) := \begin{pmatrix} v_1^{(j)} & x_3 & x_5 \\ v_2^{(j)} & 1 & x_6 \\ v_3^{(j)} & x_4 & 1 \end{pmatrix}, \qquad T_2(x, v^{(j)}) := \begin{pmatrix} 1 & v_1^{(j)} & x_5 \\ x_1 & v_2^{(j)} & x_6 \\ x_2 & v_3^{(j)} & 1 \end{pmatrix},$$

$$T_3(x, v^{(j)}) := \begin{pmatrix} 1 & x_3 & v_1^{(j)} \\ x_1 & 1 & v_2^{(j)} \\ x_2 & x_4 & v_3^{(j)} \end{pmatrix}, \qquad T(x) := \begin{pmatrix} 1 & x_3 & x_5 \\ x_1 & 1 & x_6 \\ x_2 & x_4 & 1 \end{pmatrix},$$

# points	f_*	x_*
250	2.3846154	$\left(0,0,0,\frac{2}{3},0,\frac{-2}{3}\right)$
1500	2.1172414	$\left(0,0,0,\frac{8}{9},0,\frac{-9}{9}\right)$
10,000	1.8077816	$\left(\frac{242}{243},\frac{-82}{242},\frac{-82}{243},\frac{242}{243},\frac{242}{243},\frac{-80}{243}\right)$
100,000	1.8025930	$\left(\frac{728}{729},\frac{242}{729},\frac{-244}{729},\frac{-728}{729},\frac{-728}{729},\frac{244}{729}\right)$
1,000,000	1.8008642	$\left(\frac{2186}{2187},\frac{728}{2187},\frac{-730}{2187},\frac{-2186}{2187},\frac{-2186}{2187},\frac{730}{2187}\right)$
2,000,000	1.8000960	$\left(\frac{19682}{19683},\frac{6560}{19683},\frac{-6562}{19683},\frac{-19682}{19683},\frac{-19682}{19683},\frac{6562}{19693}\right)$

Table 5.1: Results

$v^{(j)}$ as in (5.4), and $\Omega_0 := \{x \in [-1,1]^6 : \det T(x) = 0\}$."

Applying our algorithm, we ignored the existence of Ω_0. This is possible because all singularities are on the border of the domain while all evaluation points are in its interior. The results are stated in Table 5.1.

Our results confirm the conjecture diam $M_3 \geq \frac{9}{5}$.

5.2 Mean errors in global optimization

We consider a function space F endowed with a probability measure P defined on a suitable σ-algebra. We present a scheme of how the universal algorithm, or a suitable modification, can be used to prove an upper bound for the mean error numbers $e_n^{ad}(P)$ as defined in (3.4). We try to apply this scheme to the particular case $F = C[0,1]$ and P the Wiener measure P^*.

5.2.1 Basic scheme

Let (Ω_i) be a sequence of subsets of F such that

$$\Omega_0 := \emptyset,$$

$$\Omega_i \subset \Omega_{i+1},$$

$$P(\bigcup_{i\in\mathbb{N}} \Omega_i) = 1.$$

Furthermore, for a sequence $p_i \downarrow 0$ with $p_0 := 1$

$$P(\Omega_i) \geq 1 - p_i$$

shall be fulfilled. Let $(Y_n)_{n\in\mathbb{N}}$ be a sequence of optimization algorithms with Y_n using at most n function values. Concerning the sets Ω_i, their worst-case error shall be estimated by a sequence $(\epsilon_{n,i})_{n,i\in\mathbb{N}}$:

$$\Delta(Y_n, \Omega_i) \leq \epsilon_{n,i}. \tag{5.5}$$

Without loss of generality, we assume the sequence $(\epsilon_{n,i})_{n\in\mathbb{N}}$ to be monotone decreasing for fixed i. We need the following technical result.

Proposition 5.2.1. *Let $I \in \mathbb{N}$. Then*

$$\sum_{i=1}^{I} \epsilon_{n,i} P(\Omega_i \setminus \Omega_{i-1}) \leq \sum_{i=1}^{I} \epsilon_{n,i}(p_{i-1} - p_i) + \varepsilon_{n,I}[P(\Omega_I) - (1 - p_I)].$$

Proof. Recall $\Omega_0 = \emptyset$ and $p_0 = 1$. For $I = 1$ we have

$$\begin{aligned} \epsilon_{n,1} P(\Omega_1 \setminus \Omega_0) &= \epsilon_{n,1}(1 - p_1) + \epsilon_{n,1}(p_1 - 1 + P(\Omega_1 \setminus \Omega_0)) \\ &= \epsilon_{n,1}(p_0 - p_1) + \epsilon_{n,1}[P(\Omega_1) - (1 - p_1)]. \end{aligned}$$

For $I + 1$ we have by induction

$$\begin{aligned} \sum_{i=1}^{I+1} \epsilon_{n,i} P(\Omega_i \setminus \Omega_{i-1}) &\leq \sum_{i=1}^{I} \epsilon_{n,i}(p_{i-1} - p_i) + \epsilon_{n,I}[P(\Omega_I) - (1 - p_I)] + \\ &\quad \varepsilon_{n,I+1} P(\Omega_{I+1} \setminus \Omega_I) \\ &= \sum_{i=1}^{I+1} \epsilon_{n,i}(p_{i-1} - p_i) + \epsilon_{n,I+1}[P(\Omega_{I+1}) - (1 - p_{I+1})] \\ &\quad + \underbrace{(\epsilon_{I+1} - \epsilon_I)}_{\geq 0} \underbrace{[(1 - p_I) - P(\Omega_I)]}_{\leq 0} \\ &\leq \sum_{i=1}^{I+1} \epsilon_{n,i}(p_{i-1} - p_i) + \epsilon_{n,I+1}[P(\Omega_{I+1}) - (1 - p_{I+1})]. \end{aligned}$$

□

Lemma 5.2.2. *Assume that the error $\Delta(Y_1, P)$ is finite and that*

$$\lim_{I\to\infty} \epsilon_{1,I}[P(\Omega_I) - (1 - p_I)] = 0. \tag{5.6}$$

Then the average error is bounded by

$$\Delta(Y_n, P) \leq \sum_{i=1}^{\infty} \epsilon_{n,i}\,(p_{i-1} - p_i). \tag{5.7}$$

Proof. For $I \in \mathbb{N}$ we have

$$\begin{aligned}\Delta(Y_n, P) &= \int_{C[0,1]} |f(Y_n(f)) - \inf f| \, dP(f) \\ &\leq \sum_{i=1}^{I} \epsilon_{n,i} P(\Omega_i \setminus \Omega_{i-1}) + \int_{C[0,1]\setminus\Omega_I} |f(Y_n(f)) - \inf f| \, dP(f).\end{aligned}$$

Application of Proposition 5.2.1 yields

$$\begin{aligned}\Delta(Y_n, P) \leq \sum_{i=1}^{I} \epsilon_{n,i}(p_{i-1} - p_i) + \varepsilon_{n,I}[P(\Omega_I) - (1 - p_I)] + \\ \int_{C[0,1]\setminus\Omega_I} |f(Y_n(f)) - \inf f| \, dP(f).\end{aligned}$$

Taking the limit $I \to \infty$ we get

$$\Delta(Y_n, P) \leq \sum_{i=1}^{\infty} \epsilon_{n,i}(p_{i-1} - p_i).$$

□

Remark 5.2.3. For the Wiener measure P^* on $C[0,1]$ it is known that $E(\inf f) = -\sqrt{2/\pi}$, so $\Delta(Y_n, P^*)$ is finite for any method Y_1 that returns a point $x_* \in \{0, x^{(1)}\}$.

5.2.2 The Wiener measure

For a definition of the Wiener measure P^* we refer to Partzsch (1984) or Revuz, Yor (1990).

For the mean error in global optimization the following results are known. Ritter (1990) shows that a best non-adaptive method has an error of $\mathcal{O}(n^{-1/2})$ where n is the number of used function calls. As a consequence, there exists a constant C_1 such that

$$e_n^{ad}(P^*) \leq C_1 \cdot n^{-1/2}.$$

Calvin (2004) shows that the mean error cannot decrease exponentially:

$$\forall C_2, C_3 > 0 \quad \exists n \in \mathrm{N} \qquad e_n^{ad}(P^*) > C_2 \cdot \exp(-C_3 \cdot n).$$

There is a huge gap between the lower and the upper bound. We try to apply the concept of the previous section to improve the upper bound. Let

$$\alpha \in (0, 1/2) \tag{5.8}$$

and $L(i)$ as in (2.8). For the sets

$$\Omega_i' := \{f \in C[0,1] : \forall x, y \in [0,1] \qquad |f(x) - f(y)| \leq L(i)\,|x-y|^\alpha\}$$

NOVAK ET AL. (1995), Lemma 3.1, show as a corollary to LEDOUX, TALAGRAND (1991), Lemma 3.1, that

$$\forall i \in \mathbb{N} \qquad P^*(\Omega_i') \geq 1 - c_1 \exp(-c_2 L(i)^2)$$

for some constants c_1, $c_2 > 0$. Note that we have Hölder- instead of Lipschitz continuity. We will adopt $Z(h,k)$ to this new situation. Reconsider Figure 2.1. By changing the test $(*)$ to

$$f(x) \leq f_* + L\left[2^{-1}3^{-j+2}\right]^\alpha$$

we obtain a version of $Z(h,k)$ that we call

$$Z(h,\alpha,k).$$

This one can be applied to the new situation. It is easy to check that

$$\Delta(Z(h,\alpha,k),\Omega_i') \leq L(i)\left[2^{-1}3^{-k+h(m-1)+1}\right]^\alpha.$$

From NOVAK (1988) we know that $e_n^{ad}(\Omega_i') = e_n^{non}(\Omega_i') = L(i)(2n)^{-\alpha}$. As a consequence, for $\epsilon_{n,i}$ as in (5.5) we have

$$\epsilon_{n,i} = \mathcal{O}(n^{-\alpha}).$$

Since $\alpha < 1/2$ it is impossible to obtain an estimate as in (5.7) that is better than the result obtained by Ritter.

It would be helpful in this situation to have an additional estimate of the following kind: Let $g : [0,\infty) \to [0,1]$ with $g(0) = 0$ be increasing and $h : [0,\infty) \to [0,1]$ be decreasing such that

$$P^*\left(\bigcap_{\delta>0}\{f;\, \lambda(A(f,\delta)) \leq D\, g(\delta)\}\right) \geq 1 - h(D). \tag{5.9}$$

Then, with a suitable sequence $(D(i))_{i\in\mathbb{N}}$ another sequence of sets $(\Omega_i'')_{i\in\mathbb{N}}$ could be constructed:

$$\Omega_i'' := \{f \in C[0,1] : \forall \delta > 0 \quad \lambda(A(f,\delta)) \leq D(i)\, g(\delta)\}.$$

We define

$$\Omega_i := \Omega_i' \cap \Omega_i''$$

and obtain

$$P^*(\Omega_i) \geq 1 - c_1 \exp(-c_2 L(i)^2) - h(D(i)).$$

If Condition (5.6) is fulfilled we can apply our basic scheme. Then, a cost estimation as that of Theorem 2.2.2 could be established. However, it seems that a result as in (5.9) is not known.

We refrain from theoretical speculations. Instead, we are interested in how the algorithms $Z(h, \alpha, \cdot)$ behave in a numerical experiment.

5.2.3 Numerical simulation

We use the algorithms $Z(h, \alpha, \cdot)$ to determine numerically an upper bound for the error numbers $e_n^{ad}(P^*)$. We found the diploma thesis of NESTEL (1988) very useful for this purpose. In Chapter 4, he treats problems occurring when the rate of convergence of mean errors is to be determined if the Wiener measure is assumed. We only sketch how we proceed and refer to NESTEL (1988) for details.

As a first step, we introduce the algorithms

$$Y_n, \quad n \in \mathbb{N},$$

which are basically the same methods as $(Z(h, \alpha, k))_{k \in \mathbb{N}}$ with $h = 3$, Feature $F2$ and $\alpha = 0.49$. The difference is that $Z(h, \alpha, k)$ stops after k diagonals while Y_n is defined to stop after n function calls. Furthermore, Y_n uses the information $P^*(\{f(0) = 0\}) = 1$.

We want to determine numerically the behavior of

$$\Delta(Y_n, P^*) = \int_{C[0,1]} [f(Y_n(f)) - \inf f] \, P^*(df)$$

for $n \to \infty$.

Simulation of $f(Y_n(f))$

For simulating $f(Y_n(f))$ we can use the fact that for applying Y_n to a path f we do not need to simulate the whole path but only its function values at the points $x^{(1)}, \ldots, x^{(n)}$ that are used by Y_n. The following method is an application of the Lévy-Cielsielski representation of Brownian motion, see PARTZSCH (1984), pp. 26, for details. Let

$$X_1, \ldots, X_n$$

be realizations of independent standard normal distributed random variables. The first point $x^{(1)}$ evaluated by Y_n is 0.5. We set

$$f(x^{(1)}) = \sqrt{1/2} \cdot X_1.$$

Now let $2 \leq i \leq n$. Without loss of generality we assume $x^{(1)} \leq \cdots \leq x^{(i-1)}$. We define $x^{(0)} := 0$ and set $f(x^{(0)}) := 0$. For $x^{(i)}$ we have to consider three cases:

1. $x^{(i)} \in (x^{(j-1)}, x^{(j)})$ for some $1 \leq j \leq i-1$. We set

$$f(x^{(i)}) := \frac{f(x^{(j-1)}) \cdot (x^{(j)} - x^{(i)}) + f(x^{(j)}) \cdot (x^{(i)} - x^{(j-1)})}{x^{(j)} - x^{(j-1)}} + X_i \cdot \sqrt{\frac{(x^{(i)} - x^{(j-1)}) \cdot (x^{(j)} - x^{(i)})}{x^{(j)} - x^{(j-1)}}}.$$

2. $x^{(i)} = x^{(j)}$ for some $1 \leq j \leq i-1$. We set

$$f(x^{(i)}) := f(x^{(j)}).$$

3. $x^{(i)} > x^{(i-1)}$. We set

$$f(x^{(i)}) := f(x^{(i-1)}) + X_i \cdot \sqrt{x^{(i)} - x^{(i-1)}}.$$

Finally, we set

$$f(Y_n(f)) := \min_{1 \leq i \leq n} f(x^{(i)}).$$

Now we know how to simulate $f(Y_n(f))$ for a path of the Brownian motion. We present two different methods of how to determine $\Delta(Y_n, P^*)$ numerically.

Direct simulation

Let $Y_n^{(1)}, \ldots, Y_n^{(l)}$ be the realizations of l independent copies of the random variable $f(Y_n(f))$. The direct simulation uses the discretization

$$\Delta(Y_n, P^*) \approx \frac{1}{l} \sum_{i=1}^{l} Y_n^{(i)} + \sqrt{\frac{2}{\pi}} =: \widetilde{\Delta}^{(l)}(Y_n, P^*). \tag{5.10}$$

Simulation using local errors

Again, let $x^{(1)}, \dots, x^{(n)}$ be the points chosen by Y_n. The information vector used by Y_n is given by

$$N_n : C[0,1] \to \mathbb{R}^n, \quad f \mapsto (f(x^{(1)}), \dots, f(x^{(n)})).$$

N_n is a random variable. Let $P^*_{N_n}$ denote its distribution. We use the desintegration

$$\begin{aligned} \Delta(Y_n, P^*) &= \int_{\mathbb{R}^n} \int_{C[0,1]} (f(Y_n(f)) - \inf f) P^*(df | N_n = y)\, P^*_{N_n}(dy) \\ &= \int_{\mathbb{R}^n} \underbrace{\left[\min\{0, y_1, \dots, y_n\} - \int_{C[0,1]} \inf f P^*(df | N_n = y) \right]}_{\text{local error } \Delta(Y_n, P^*, y)} dP^*_{N_n}(y). \end{aligned}$$

Let $Y_n^{(1)}, \dots, Y_n^{(l)}$ be the realizations of l independent copies of $f(Y_n(f))$ and $N_n^{(1)}, \dots, N_n^{(l)}$ be the according information vectors. The simulation with local errors uses the discretization

$$\Delta(Y_n, P^*) \approx \frac{1}{l} \sum_{i=1}^{l} \left[Y_n^{(i)} - \int_{C[0,1]} \inf f P^*(df \mid N_n = N_n^{(i)}) \right] =: \Delta^{(l)}(Y_n, P^*). \tag{5.11}$$

The simulation $\Delta^{(l)}(Y_n, P^*)$ is by far more complicated than $\widetilde{\Delta}^{(l)}(Y_n, P^*)$ since we have to compute the local errors $\Delta(Y_n, P^*, N_n^{(i)})$ numerically. On the other hand, $\Delta^{(l)}(Y_n, P^*)$ has a decisive advantage, as we will see in the next section.

Asymptotic cost

Both $\Delta^{(l)}(Y_n, P^*)$ and $\widetilde{\Delta}^{(l)}(Y_n, P^*)$ are realizations of random variables, say $\Delta_n^{(l)}$ and $\widetilde{\Delta}_n^{(l)}$. Let us consider $\widetilde{\Delta}_n^{(l)}$ first. In order to obtain a meaningful result, we require for the standard deviation, its mean value, and a constant $0 < c < 1$ that

$$\sigma(\widetilde{\Delta}_n^{(l)}) \le c \cdot E(\widetilde{\Delta}_n^{(l)}). \tag{5.12}$$

This has consequences for the choice of $l = l(n)$. Empirical results lead us to assume that

$$\forall\, n \in \mathbb{N} \qquad \sigma(f(Y_n(f))) \approx \sigma(\inf f).$$

Consequently,

$$\sigma(\widetilde{\Delta}_n^{(l)}) \approx n^{-1/2} \sigma(\inf f).$$

If we want to test the conjecture $\Delta(Y_n, P^*) = \mathcal{O}(n^{-p})$ for some $p > 0$ then we need

$$l(n) = \mathcal{O}(n^{2p})$$

runs, i.e. $\mathcal{O}(n^{2p+1})$ function calls / random numbers.

Example 5.2.4. Consider the algorithm A_n that chooses the points equidistantly: $N_n := (f(1/n), f(2/n), \ldots, f(1))$. We know from RITTER (1990) that it has a convergence rate of $n^{-1/2}$. For $n = 21$ knots we need $l = 100$ runs in order to get

$$\sigma(\widetilde{\Delta}^{(l)}(A_n, P^*)) \leq 0.5 \cdot E(\widetilde{\Delta}^{(l)}(A_n, P^*)).$$

For $n = 18.000$, we need $l = 90.000$ runs, see NESTEL (1988) for details. Now let us assume a rate of convergence of 1 instead of 0.5. Then, these 90.000 runs are already necessary for $n = 134 \approx \sqrt{18000}$. We see that the direct simulation is problematic in examining asymptotic behavior for "high" convergence rates.

The situation is better for $\Delta^{(l)}(Y_n, P^*)$ because the standard deviation of the local error is decreasing with l. NESTEL (1988) gives the upper bound

$$\sigma^2(f(Y_n(f))) \leq \frac{L}{2},$$

where L is the length of the longest subinterval that is induced by the points $x^{(1)}, \ldots, x^{(n)}$. The empirical standard deviations decreases much quicker. Let us assume that $\sigma(f(Y_n(f))) = \mathcal{O}(n^{-q})$. To fulfill condition (5.12) we need

$$l(n) = \mathcal{O}(n^{2(p-q)})$$

runs, i.e., $\mathcal{O}(n^{2(p-q)+1})$ function calls / random numbers. We want to use this advantage towards direct simulation.

Calculating the local error

We sketch how we calculate the local error numerically, i.e., for $y \in \mathbb{R}^n$ we want to approximate

$$E_y(\inf f) := \int_{C[0,1]} \inf f \, P^*(df \mid N_n = y).$$

We implement the calculation in Maple, using the standard accuracy of ten digits. We proceed in 4 steps.

1. Let $P_y^* := P^*(\,\cdot \mid N_k = y)$. For $y = (y_1, \ldots, y_k)$ let $y_0 := \min\{0, y_1, \ldots, y_k\}$. With partial integration we get

$$\begin{aligned} E_y(\inf f) &= \int_{-\infty}^{y_0} u P_y^*(\inf f = u)\, du = y_0 - \int_{-\infty}^{y_0} P_y^*(\inf f \le u)\, du \\ &= y_0 - \int_{-\infty}^{y_0} [1 - P_y^*(\inf f > u)]\, du. \end{aligned}$$

Assume $0 = x^{(0)} < \cdots < x^{(n)} < 1$. Then we know (see, e.g., Partzsch (1984), Behauptung 4.3)

$$\begin{aligned} & P^*(\inf_{x^{(j-1)} \le x \le x^{(j)}} f(x) > u \mid f(x^{(j-1)}) = y_{j-1}, f(x^{(j)}) = y_j) \\ = & \begin{cases} 1 - \exp(-2\frac{(y_{j-1}-u)(y_j-u)}{x^{(j)}-x^{(j-1)}}), & u < \min\{y_{j-1}, y_j\}, \\ 0, & \text{else.} \end{cases} \end{aligned}$$

For $x^{(n)} \le x \le 1$ we have

$$P^*(\inf_{x^{(n)} \le x \le 1} f(x) > u \mid f(x^{(n)}) = y_n) = \begin{cases} 2\Phi\left(\frac{y_n - u}{\sqrt{1-x^{(n)}}}\right) - 1, & u < y_n, \\ 0, & \text{else,} \end{cases}$$

with Φ the distribution function of the standard normal distribution. For k independent random variables Z_j we know

$$P(\min_{1 \le j \le k} Z_j > u) = \prod_{j=1}^{k} P(Z_j > u).$$

So,

$$E_y(\inf f) = \int_{-\infty}^{y_0} [1 - P_y^*(\inf f > u)]\, du \tag{5.13}$$

$$= y_0 - \int_{-\infty}^{y_0} \underbrace{\left[1 - \left[2\Phi\left(\frac{y_n - u}{\sqrt{1-x^{(n)}}}\right) - 1\right] \prod_{j=1}^{k} (1 - \exp(-2\frac{(y_{j-1}-u)(y_j-u)}{x^{(j)}-x^{(j-1)}}))\right]}_{h(u)} du.$$

2. To approximate (5.13), we cannot integrate over $(-\infty, y_0]$ but only over a finite interval. Let ϵ be the accuracy we want to determine the integral with. Let L be the length of the longest subinterval induced by the evaluation points, let

$$z_0 := y_0 - \sqrt{-\frac{L}{4} \ln(0.5\pi L \epsilon^2)}, \qquad z_{i+1} := (z_i + y_0)/2.$$

For j such that $h(z_{j+1}) > \epsilon$ and $h(z_j) \leq \epsilon$, choose $[z_j, y_0]$ as the integration interval. Then, the maximal error is less than ϵ. We choose $\epsilon := 10^{-1.5 \cdot \log(n)}$.

3. To approximate $h(u)$ we use logarithmic calculation. Define

$$q_j := \exp(-2\frac{(y_{j-1}-u)(y_j-u)}{x^{(j)}-x^{(j-1)}}), \qquad p_j := 1-q_j.$$

We use

$$\prod_{j=1}^{k} p_j = \exp(\textstyle\sum_{j=1}^{k} \ln p_j).$$

Because very often $q_j \ll 1$ we can calculate $\ln p_j$ with the Taylor expansion $\ln(1-q) = -\sum_{j=1}^{\infty} \frac{q^j}{j}$. A faster converging method is $\ln(\frac{1-u}{1+u}) = \ln(1-q)$ with $u = \frac{q}{2-q}$. We get the Taylor expansion

$$\ln(\frac{1-u}{1+u}) = -2\sum_{j=0}^{\infty} \frac{u^{2j+1}}{2j+1}. \tag{5.14}$$

We take the first 10 summands of (5.14). For the calculation of Φ we use the subroutine of Maple.

4. To calculate $\int_{z_j}^{y_0} h(u)\,du$ numerically, we take into consideration that $h(u)$ decreases quickly for u decreasing. We can exploit that h is smooth. We use Romberg integration, see, e.g., KINCAID, CHENEY (2002) for details. To approximate

$$I(f) = \int_a^b f(x)\,dx$$

we introduce

$$\begin{aligned}
R(0,0) &:= \frac{1}{2}(b-a)[f(a)+f(b)], \\
R(j,0) &:= \frac{1}{2}R(j-1,0) + h_j \sum_{i=1}^{2^{j-1}} f(a+(2i-1)h_j \\
R(j,m) &:= R(j,m-1) + \frac{1}{4^m-1}[R(j,m-1)-R(j-1,m-1)], \quad m \leq j,
\end{aligned}$$

with $h_0 := b-a$ and $h_j = h_{j-1}/2$ for $j \geq 1$. The Romberg integration $R(j,j)$ is an $\mathcal{O}(h_j^{2j})$ approximation of $I(f)$ if $f \in C^{2j}[a,b]$. We use $R(6,6)$.

n	$\Delta^{(l)}(Y_n, P^*)$	$\widetilde{\sigma}(f(Y_n(f)))$	$\widetilde{\sigma}(\Delta^{(l)}(Y_n, P^*))$
12	0.07438209115	0.02342877456	0.01171438728
25	0.02201084315	0.02010604556	0.006358089871
50	0.008856878920	0.01525149098	0.003410337058
100	0.003076658378	0.006514667583	0.001030059388
200	0.001130572636	0.004480232594	0.0005009052314
400	0.00003234345839	0.0001537732155	0.00001215684010
800	0.00003342611825	0.0004077644289	0.00002279472453
1600	0.000009293359809	0.0001084567565	0.000004287129726

Table 5.2: Results for the simulated mean error

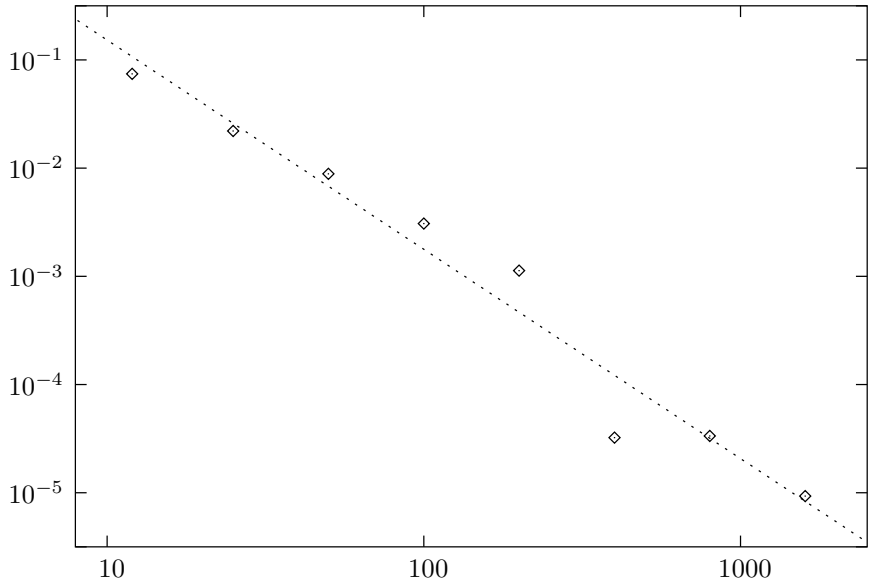


Figure 5.1: The simulated mean error on a logarithmic scale.

Results

We determined numerically the values of $\Delta^{(l)}(Y_n, P^*)$ for

$$n = 12, 25, 50, 100, 200, 400, 800, 1600.$$

We assumed a rate of convergence of the error of at least n^{-1} and a rate of $n^{-1/2}$ for the standard deviation of the local error. In order to fulfill (5.12) we set $l(n) := \lceil c \cdot n \rceil$ with $c := 0.4$. In addition to the result for $\Delta^{(l)}(Y_n, P^*)$,

we also recorded the empirical standard deviation of the local error, and the empirical standard deviation of $\Delta^{(l)}(Y_n, P^*)$, see Table 5.2 for the results. They show that the number $l(n)$ of iterations was chosen reasonably.

A least-square fit of the logarithmized data suggests to assume a dependence of

$$\Delta(Y_n, P^*) \approx 13.12 \cdot n^{-1.93},$$

which is represented by the dotted line in Figure 5.1.

Notations

$A(f,\delta)$	5	level set
A, $A(f)$	15	optimization algorithm
$\mathcal{A}_n^{ad}(F)$	38	class of adaptive methods
$\mathcal{A}_n^{non}(F)$	38	class of non-adaptive methods
$\asymp$	41	
$B(x,r,\|\cdot\|)$	7	ball
$B(x,r)$	8	ball
$\mathrm{cost}(A,f)$	15	function-wise error
$\mathrm{cost}(A,F)$	15	worst-case cost
$c(d,h)$	23	constant
$C(L,d,h)$	23	constant
$\lceil\cdot\rceil$	22	ceiling function
D	5	class parameter
$\mathcal{D}$	33	domain
$\Delta(A,f)$	37	function-wise error
$\Delta(A,F)$	37	worst-case error
$\Delta(A,P)$	41	mean error
$\Delta(Q,f)$	42	error
$\Delta(Q,F)$	42	error
e_i	15	unit vector in $\mathbb{R}^d$
$e_n^{ad}(F)$	39	error number
$e_n^{non}(F)$	39	error number
$e_n^{ad}(P)$	41	mean error numbers
$e_n^{non}(P)$	41	mean error numbers
$\varepsilon_{L,k}$	17	error level
$\varepsilon_{L,h,k}$	23	error level
$F^d_{L,D,\varrho}$	5	function class
f_*	17	approximation
F	33	problem class
$\mathbb{F}^d$	10	class of problem classes $F^d_{L,D,\varrho}$

$j(L, \varrho)$	17	constant
$j(L, D, \varrho, d)$	17	constant
$j(m, i, \varrho)$	24	constant
$j'(m, i, h, \varrho)$	25	constant
L	5	Lipschitz parameter
$L(i)$	22	Lipschitz parameter
lastconst	20	control function
laststep	20	control function
$\mathcal{M}$	16	midpoint $(1/2, \ldots, 1/2)$
mesh(j)	16	equidistant mesh
$MC(\mathcal{A}_n^{ad}(F))$	42	set of adaptive MCMs
$MC(\mathcal{A}_n^{non}(F))$	42	set of non-adaptive MCMs
$N_{L,j}$	17	
$N_{L,j}^*$	18	
N	36	information vector
$\mathbb{1}$	12	$\mathbb{1} := (1, \ldots, 1)^T$
P	41	probability measure
P^*	41	Wiener measure
ϕ	37	mapping
Q	42	Monte Carlo method
R_k	34	register
r_k	34	real number in register
ϱ	5	class parameter
$S(L, k)$	17	non-universal algorithm
step(L, j)	17	
step$(L, 1')$	20	
$\sigma_n^{ad}(F)$	42	error number
$\sigma_n^{non}(F)$	43	error number
x^*	7	global minimizer
x_*	15	approximation
$Y(j)$	15	set of scaled vectors
$Z(k)$	21	universal algorithm
$Z(h, k)$	23	universal algorithm

Bibliography

Babuska, I. (1968), Über universal optimale Quadraturformeln, in *Appl. Mat.* **13**, pp. 304-338, 338-404.

Bakhvalov, N. S. (1959), On approximate computation of multiple integrals (in Russian), in *Vestnik Moscow Univ. Ser. I Mat. Mekh.* **4**, pp. 3-18.

Borgwardt, K.H. (1987), *The Simplex Method. A Probabilistic Analysis,* Springer, Berlin.

Brass, H. (1988), Universal quadrature rules in the space of periodic functions, in *Numerical Integration III,* eds. H. Brass and G. Hämmerlin, ISNM **85**, pp. 16-24, Birkhäuser, Basel.

Calvin, J. M. (2004), Lower bounds on complexity of optimization of continuous functions, in *Journal of Complexity,* **20**, pp. 773-795.

Gluskin, E. D. (1981), The diameter of the Minkowski compactum is approximately equal to n, in *Functional Analysis and its Applications* **15** (1), pp. 57-58.

Griebel, M., Woźnakowski, H. (2005), On the optimal convergence rate of universal and non-universal algorithms for multivariate integration and approximation, preprint.

John, F. (1948), Extremum problems with inequalities as subsidiary conditions, in *Courant Anniversary Volume,* pp. 187-204. Interscience, New York.

Jones, D. R., Perttunen, C. D., and Stuckman, B. E. (1993), Lipschitzian optimization without the Lipschitz constant, in *Journal of Optimization Theory and Application* **79**, No. 1, pp. 157-181.

Kincaid, D., Cheney, W. (2002), *Numerical Analysis: Mathematics of Scientific Computing,* 3rd edition. Brooks/Cole, Pacific Grove.

LEDOUX, M., TALAGRAND, M. (1991), *Probability in Banach Spaces,* Springer, Berlin.

MOTORNYJ, V., On the best quadrature formula of the form $\sum_{k=1}^{n} p_k f(x_k)$ for some classes of periodic differentiable functions, in *Ivz. Adak. Nauk USSR ser. Mat.* **38** , pp. 538-614.

NEMIROVSKI, A. (1995), Polynomial time methods in convex programming, in *AMS-SIAM Summer Seminar in Applied Mathematics (1995: Park City, Utah),* pp. 543-589. American Mathematical Society, Providence.

NEMIROVSKI, A., YUDIN, D. (1983), *Problem Complexity and Method Efficiency in Optimization,* John Wiley, Chichester.

NESTEL, F. (1988), *Globale Optimierung auf Wienerpfaden,* Diploma thesis, Universität Erlangen.

NOVAK, E. (1988), *Deterministic and Stochastic Error Bounds in Numerical Analysis,* Springer Lecture Notes in Mathematics **1349**, Berlin.

NOVAK, E. (1995), The real number model in numerical analysis, in *Journal of Complexity* **11**, pp. 57-73.

NOVAK, E., RITTER, K. (1996), High dimensional integration of smooth functions over cubes, in *Numerische Mathematik* **75**, pp. 79-97.

NOVAK, E., RITTER, K. (1998), The curse of dimension and a universal method for numerical integration, in *Multivariate Approximation und Splines,* eds. G. Nürnberger, J. Schmidt and G. Walz, International Series in Numerical Mathematics, Birkhäuser, Basel.

NOVAK, E., RITTER, K., WOŹNIAKOWSKI, H. (1995), Average case optimality of a hybrid secant-bisection method, in *Mathematics of Computation* **64**, No. 212, pp. 1517-1539.

PARTZSCH, L. (1984), *Vorlesungen zum eindimensionalen Wienerschen Prozess,* Teubner-Texte zur Mathematik **66**, Leipzig.

PEREVOZCHIKOV, A. G. (1990), The complexity of the computation of the global extremum in a class of multi-extremum problems, in *U.S.S.R. Comp. Maths. Math. Phys.* **30**, No. 2, pp. 28-33.

PETRAS, K. (1996), On the universality of the Gaussian formula, *East J. Approx.* **2**, pp. 427-438.

Revuz, D., Yor, M. (1990), *Continuous Martingales and Brownian Motion,* Springer Grundlehren der mathematischen Wissenschaften **293**, 3rd edtion 1999, Berlin.

Ritter, K. (1990), Approximation and optimization on the Wiener space, in *Journal of Complexity,* **6**, pp. 337-364.

Tomczak-Jaegermann, N. (1989), *Banach-Mazur Distances and Finite-Dimensional Ideals,* Pitman Monographs and Surveys in Pure and Applied Mathematics **38**, Longman Scientific & Technical, Essex.

Törn, A., Žilinskas, A. (1989), *Global Optimization,* Springer Lecture Notes in Computer Science **350**, Berlin.

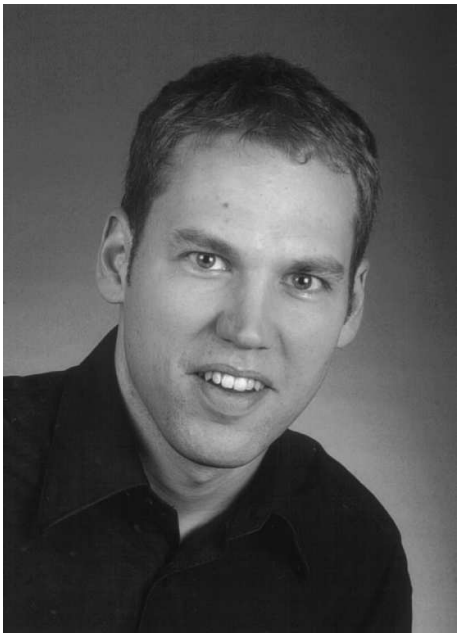
Wasilkowski, G.W. (1989), Randomization for continuous problems, in *Journal of Complexity* **5**, 2, pp. 195.

Selbständigkeitserklärung

Ich erkläre, daß ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Hilfsmittel und Literatur angefertigt habe.

Jena, den 29. Juli 2005

Lebenslauf



Geboren am 11. Mai 1975 in Frankfurt am Main

Oktober 2002 bis September 2005

Stipendiat im Graduiertenkolleg 'Approximation und algorithmische Verfahren', Universität Jena.

Oktober 2001 bis August 2002

Wissenschaftlicher Mitarbeiter in der Arbeitsgruppe Stochastik am Fachbereich Mathematik der Universität Oldenburg.

April 1996 bis Mai 2001

Studium Diplom-Mathematik mit Nebenfach Volkswirtschaftslehre an der TU Darmstadt. Diplomarbeit in der stochastischen Analysis bei Prof. Dr. K. Ritter.

Wintersemester 1995/96

Ein Semester Architektur an der TU Darmstadt.

Juli 1994 bis Juni 1995

Wehrdienst.

1981 bis 1994

Schule. Abitur am Immanuel-Kant-Gymnasium, Rüsselsheim.