Diplomarbeit

Effiziente Optimierung hochdimensionaler Probleme mit Anwendungen aus der Finanzmathematik

Angefertigt am
Institut für Numerische Simulation

Vorgelegt der
Mathematisch-Naturwissenschaftlichen Fakultät der
Rheinischen Friedrich-Wilhelms-Universität Bonn

Juni 2013

Von

Nina Merz

Aus
Hattert

Inhaltsverzeichnis

1 Überblick und Gliederung

„Most studies find that the universe of mutual funds does not outperform its benchmarks after expenses [...]“ [15]

1.1 Optimierung in der Finanzmathematik

Finanzmärkte durchdringen fast alle Bereiche unseres täglichen Lebens. Kaum eine Transaktion - sei es der Erwerb von Nahrung, der Erhalt eines Wohnsitzes, das Konsumieren von Transport und Energie - greift nicht auf das abstrakte Zahlungsmittel „Geld“ zurück und unterliegt damit in seiner Wertstellung den globalen Märkten. Auch eine Privatperson kann sich also der Teilnahme am globalen Markt nicht entziehen.

Während vor etwa 500 Jahren, zur Entstehungszeit der ersten regionalen Börsen, Angebot und Nachfrage von Gütern noch vergleichsweise überschaubar gewesen sein mögen, hat die globale Marktwirtschaft in den letzten Jahrzehnten auf beinahe erschreckende, exponentielle Weise an Komplexität gewonnen. Und nicht nur an Komplexität - auch an Geschwindigkeit. Investitionsentscheidungen müssen in immer kürzerer Zeit getroffen werden, bei steigender Zahl von zu beachtenden Variablen, um Gewinn zu bringen.

Insgesamt scheinen grundlegende Fragen, darunter „Wie investiere ich meine Mittel auf kluge Weise?“, „Wie wäge ich Anlagerisiko und Gewinnchance gegeneinander ab?“, für die Einzelperson ohne Computer-Unterstützung praktisch nicht beantwortbar.

Wie also den gewaltigen Datenmengen, den komplexen und zufällig wirkenden Prozessen, den Risiken Herr werden? Damit beschäftigt sich das Gebiet der Finanzmathematik. Die Komplexität des globalen Marktes spiegelt sich nicht zuletzt in den Ressourcen wider, die aufgewandt werden, um Entscheidungen an globalen Märkten zu treffen: Geschätzte *zwanzig* Prozent der weltweiten Rechenleistung stehen im Dienst der globalen Finanzen. Dieser gewaltige Aufwand liegt nicht zuletzt im „Fluch der Dimension“ begründet: Die meisten finanzmathematischen Probleme sind hochdimensional, oft schon allein weil die Zahl der betrachteten Güter so groß ist, andererseits aber auch durch die stochastische Natur vieler solcher Probleme - insbesondere bei Finanzderivaten. Die Kosten einer ϵ-Approximation an die optimale Lösung eines solches Problems steigen exponentiell mit dessen Dimension. Die Finanzmathematik beschäftigt sich mit der Entwicklung von Methoden und Algorithmen für konkrete Probleme aus dem Finanzbereich, um diesen Fluch der Dimension möglichst zu umgehen oder zumindest dessen Auswirkungen abzuschwächen. Ein wichtiges Konzept zur Lösung multivariater Probleme ist dabei die *Zerlegung* der zu optimierenden Funktion nach bestimmten Merkmalen.

In dieser Arbeit soll ein solches hochdimensionales Optimierungsproblem aus der Finanzmathematik vorgestellt werden, das *Tracking-Error-Minimization*-Problem. Wie das eingehende Zitat beschreibt, erkennen die Akteure der globalen Finanzmärkte immer mehr, dass es sich beim hochfrequenten Investmenthandel um ein Nullsummenspiel handelt. Es zeichnet sich ein Trend zu sogenannten *passiven Handesstrategien* ab. Zu diesen gehört auch das in dieser Arbeit behandelte *Index Tracking*. Es betrachtet die Frage, wie zu einem gegebenen Aktienindex ein möglichst kleines Portfolio aus Aktien dieses Index ausgewählt werden kann, sodass über einen gewissen Zeitraum der Wert des Portfolios möglichst exakt dem Wert des Aktienindex entspricht. Dabei soll dieses Tracking-Portfolio in seinem Wertverlauf über einen möglichst langen Zeitraum die Entwicklungen des entsprechenden Marktes nachbilden. Die Anlagestrategie besteht also darin, einen Markt auszuwählen, von dem man sich - als Gesamtes - ein Wachstum erhofft, und diesen dann mit möglichst geringem Aufwand - einem kleinen Portfolio aus wenig verschiedenen Aktien - zu „tracken“. Solche Tracking-Portfolios erfreuen sich in den letzten Jahren wachsender Beliebtheit. Für große Aktienindizes, etwa den amerikanischen Russel3000-Index, ergeben sich durch die große Zahl an Aktien im Index Optimierungsprobleme mit über tausend Variablen.

1.2 Gliederung

Die Arbeit ist wie folgt gegliedert:

Zunächst werden in Kapitel 2 Klassen von Optimierungsproblemen und Optimierungsansätze für die einzelnen Klassen vorgestellt.

Danach werden in Kapitel 3 Ansätze zur Zerlegung von Funktionen in Teilfunktionen mit niedrigerer Dimension eingeführt.

Zum allgemeinen Lösen solcher zerlegten Funktionen lassen sich dann die in Kapitel 4 beschriebenen iterativen Lösungsansätze verwenden.

Nachdem in Kapitel 5 die grundlegenden Begriffe und Modelle der Finanzmathematik eingeführt wurden, wird das *Tracking-Error-Minimization*-Problem selbst in Kapitel 6 in diesem Kontext formuliert. Im selben Kapitel wird ein Lösungsansatz des Tracking-Error-Minimization-Problems vorgestellt. Dieser Ansatz basiert auf dem Prinzip der *Graduated Non-Convexity*, der es erlaubt, unerwünschte Eigenschaften - hier nicht-Konvexität - einer zu optimierenden Funktion zu reduzieren.

Auf diesem Ansatz aufbauend werden nun die Zerlegungs-Techniken aus Kapitel 3 entsprechend der allgemeinen Formulierung in Kapitel 4 an das konkrete Problem angepasst. So existieren nun sogenannte *multiplikative* und *additive* Verfahren zur Minimierung des Tracking Error, beschrieben in Kapitel 7. Diese Verfahren wurden implementiert und getestet, Ergebnisse - auch Vergleiche mit ursprünglichen, bereits existierenden Verfahren - werden in Kapitel 8 vorgestellt.

Schließlich werden in Kapitel 9 mögliche Verbesserungen und weitere Anwendungsbereiche des Algorithmus diskutiert.

2 Globale Optimierung

In diesem Kapitel werden die für das später betrachtete Anwendungsproblem relevanten Teilgebiete der Globalen Optimierung beschrieben.

Zunächst sei ein *globales Optimierungsproblem* wie folgt definiert: Gegeben eine Funktion $f : \mathbb{R}^n \to \mathbb{R}$, eine Menge $S \subseteq \mathbb{R}^n$ und $m \in \mathbb{N}$ weitere Funktionen $g_i : \mathbb{R}^n \to \mathbb{R}$. Dann ist das zugehörige globale Optimierungsproblem definiert durch:

$$\boxed{\begin{aligned} &\min f(\mathbf{x}) \\ &\text{s.t.} \\ &\quad g_i(\mathbf{x}) = 0 \text{ für } i = 1, \ldots, m \\ &\quad \mathbf{x} \in S \end{aligned}} \tag{2.1}$$

Die zu optimierende Funktion f wird auch *Zielfunktion* genannt. Über ein globales Optimierungsproblem in dieser allgemeinen Formulierung können keine Aussagen bezüglich dessen Lösbarkeit getroffen werden. Gleichzeitig kann man auch zu Lösern allgemeiner Optimierungsprobleme keine Aussagen bezüglich Komplexität, Konvergenz und Qualität der Lösung machen.

Dass es keine numerische Methode geben kann, die für alle Optimierungsprobleme in optimaler Zeit eine optimale Lösung findet, zeigt bereits ein einfaches Gedankenexperiment aus [20]: Ein Algorithmus, der nichts weiter tut, außer als „Lösung“ $\mathbf{x} = 0$ zurückzugeben, hat eine optimale, da konstante, Laufzeit für eben jene Probleme, deren optimale Lösung bei $\mathbf{x} = 0$ liegt. Gleichzeitig ist dieser Algorithmus beliebig schlecht für alle übrigen Probleme.

Sinnvolle Aussagen über Lösungsmethoden lassen sich also nur für *Klassen* von Optimierungsproblemen angeben. Daher werden in den folgenden Abschnitten solche einschränkenden Problem-Klassen betrachtet.

Die einzelnen Klassen solcher Optimierungsprobleme unterscheiden sich oft nur in kleinen Details, die jedoch weitreichende Konsequenzen für Lösungsverfahren haben können.

Die wichtigsten Optimierungsklassen und deren Lösungsmethoden sind die folgenden:

- Linerare Optimierungsprobleme der Form

$$\boxed{\begin{aligned} &\min \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ &\text{s.t.} \\ &\mathbf{A}\mathbf{x} \leq \mathbf{b} \end{aligned}} \tag{2.2}$$

mit $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$ und $\mathbf{x} \in \mathbb{R}^n$. Lineare Optimierungsprobleme lassen sich in der Theorie mit der Ellipsoidmethode [14] in polynomieller Zeit optimal lösen. In der Praxis wird der Simplex-Algorithmus [11] verwendet, der zwar eine theoretisch exponentielle Laufzeit besitzt, aber deutlich leichter zu implementieren ist als die Ellipsoidmethode und praktische Probleme in den meisten Fällen schneller löst.

- Konvexe Optimierungsprobleme der Form

$$\boxed{\begin{aligned} &\min f(\mathbf{x}) \\ &\text{s.t.} \\ &\mathbf{x} \in \mathcal{Q} \end{aligned}} \tag{2.3}$$

  mit einer konvexen Funktion $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ und einer konvexen Menge $\mathcal{Q} \subseteq \mathbb{R}^n$. Da bei einer konvexen Funktion ein lokales Minimum stets auch ein globales Minimum ist, beschränkt sich das Minimierungsproblem darauf, *ein* Minimum zu finden. Zum Finden eines solchen Minimums werden Innere-Punkte-Methoden [22], aber auch Subgradienten-Verfahren [27] benutzt.

- Quadratische Optimierungsprobleme, eine Unterklasse der Konvexen Optimierungsprobleme, der Form

$$\boxed{\begin{aligned} &\min \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} + \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ &\text{s.t.} \\ &\mathbf{A}\mathbf{x} \leq \mathbf{b} \end{aligned}} \tag{2.4}$$

  mit einer symmetrischen Matrix $\mathbf{Q} \in \mathbb{R}^{n \times n}$, $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$ und $\mathbf{x} \in \mathbb{R}^n$, welche sich mit wieder Innere-Punkte-Methoden lösen lassen, aber auch mit einer passenden Variante des Simplex-Algorithmus [30]. Damit sind Quadratische Optimierungsprobleme in der Praxis schnell und effizient lösbar.

- Nichtlineare Optimierungsprobleme: Ist f nicht konvex, aber stetig differenzierbar, und sind ebenso die Nebenbedingungsfunktionen $g_i(\mathbf{x})$stetig differenzierbar, so kann man das zugehörige Nichtlineare Optimierungsproblem schreiben als:

$$\boxed{\begin{aligned} &\min f(\mathbf{x}) && \\ &\text{s.t.} && \\ &g_i(\mathbf{x}) \geq 0 \quad && \text{für } i = 1, 2, \ldots, m \end{aligned}} \tag{2.5}$$

  Bei solch allgemeinen Optimierungsproblemen muss ein Lösungsansatz nicht mehr zum globalen Minimum konvergieren. Lokale Minima lassen sich jedoch finden, etwa mit der Methode des Sequential Quadratic Programming, welche in Abschnitt 2.2 eingeführt wird.

In allen Vektor-Gleichungen sind Vergleiche und Gleichheit, sofern nicht anders angemerkt, komponentenweise gemeint.

Zwei dieser Teilgebiete der Globalen Optimierung verdienen in Bezug auf die spätere Anwendung besondere Beachtung: Das in Kapitel 6 vorgestellte Optimierungsproblem ist zunächst aufgrund seiner Nebenbedingungen weder differenzierbar noch stetig. Der in Abschnitt 6.6 beschriebene GNC-Ansatz ermöglicht jedoch eine Formulierung, die zunächst in die Klasse der *Nichtlinearen Optimierungsprobleme* eingeordnet werden kann: Ein nichtlineares, differenzierbares Funktional soll unter nichtlinearen, differenzierbaren Nebenbedingungen optimiert werden. Obwohl die in dieser Arbeit betrachtete Anwendung zunächst die Optimierung eines nicht-konvexen Funktionals beinhaltet, werden bei der Lösung des Problems Schritte unternommen, die zu optimierende Funktion konvex zu approximieren. Somit lassen sich auch Lösungsmethoden aus der *Konvexen Optimierung* verwenden. Für die Klasse der Konvexen Optimierungsprobleme sind globale Konvergenzbeweise von Lösungsalgorithmen möglich. Dies motiviert den Ansatz, auch nicht-konvexe Probleme zunächst durch konvexe oder annähernd konvexe Funktionen zu beschreiben.

2.1 Nichtlineare Optimierung

Für eine glatte Funktion f ist das *Nichtlineare Optimierungsproblem* (Non-Linear Program, NLP) wie in (2.5) definiert.

Die Glattheits-Eigenschaft ermöglicht nun Konvergenzbeweise von Lösungsmethoden. Dabei wird je nach Methode von der Funktion f nicht unbedingt Glattheit, sondern ein- oder zweimalige stetige Differenzierbarkeit gefordert. Ein Lösungsansatz dieses Optimierungsproblems ist das *Sequential Quadratic Programming* (SQP), welches im nächsten Abschnitt beschrieben wird. Das SQP-Verfahren arbeitet mit einer Approximation der Hesse-Matrix der Funktion f. Andere Verfahren, etwa *Trust-Region-Verfahren* [9], erfordern nur die Berechnung der ersten Ableitung von f.

2.2 Sequential Quadratic Programming

Das in Abschnitt 2.1 beschriebene Optimierungsproblem (2.5) wird nun zunächst auf ganz $\mathbb{R}^n$ betrachtet:

$$\boxed{\begin{array}{l}\min\limits_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \\ \text{s.t.} \\ g(\mathbf{x}) \geq 0\end{array}} \tag{2.6}$$

Die m Funktionen $g_i(\mathbf{x})$ wurden dabei zusammengefasst zu $g(\mathbf{x})$. Die obige Formulierung enthält nur Ungleichheitsbedingungen; Schranken und Gleichheitsbedingungen lassen sich jedoch leicht als Nebenbedingungsfunktionen $g_i(\mathbf{x})$ hinzufügen.

2.2.1 Lagrange-Multiplikatoren

Die etablierte Methode der Lagrange-Multiplikatoren (siehe etwa [25]) formuliert nun die Nebenbedingungsfunktionen um, sodass diese Teil der Zielfunktion werden.

Dazu sei zunächst das *zulässige Gebiet* P definiert als die Menge aller zulässigen Lösungen:

$$P := \{\mathbf{x} \in \mathbb{R}^n : g(\mathbf{x}) \geq 0\}$$

und die Menge der *aktiven Nebenbedingungen* bezüglich $\mathbf{x} \in P$ definiert durch

$$I(\mathbf{x}) := \{j : g_j(\mathbf{x}) = 0, 1 \leq j \leq m\} \ .$$

Die *Langrange-Funktion* von 2.6 ist dann definiert durch

$$L(\mathbf{x}, \mathbf{u}) \ := f(\mathbf{x}) - \mathbf{u}^{\mathrm{T}} g(\mathbf{x}) \tag{2.7}$$

für alle $\mathbf{x} \in \mathbb{R}^n$ und $\mathbf{u} = (u_1, \ \ldots, \ u_m)^{\mathrm{T}} \in \mathbb{R}^m, \ \mathbf{u} \geq 0$. Die Variablen u_j heißen *Lagrange-Multiplikatoren* des Problems, $\mathbf{x}$ heißt dessen *primale*, $\mathbf{u}$ dessen *duale* Variable. Mit dieser Formulierung kann die Bedingung für stationäre Punkte der Lagrange-Funktion sofort angegeben werden:

$$\nabla L(\mathbf{x}, \mathbf{u}) = 0 \ .$$

Dabei gilt wie üblich

$$\nabla f(\mathbf{x}) = \left(\frac{\partial}{\partial x_1} f(\mathbf{x}), \ldots, \frac{\partial}{\partial x_n} f(\mathbf{x}) \right)^{\mathrm{T}} \ .$$

Für vektorwertige Funktionen $g : \mathbb{R}^n \to \mathbb{R}^m$ wird der Operator ∇ auch verwendet, um die Jacobi-Matrix zu bezeichnen:

$$\nabla g(\mathbf{x}) = (\nabla g_1(\mathbf{x}), \ldots, \nabla g_m(\mathbf{x}))$$

2.2.2 Das SQP-Verfahren

Das SQP-Verfahren ist ein iteratives Verfahren für einfach stetig differenzierbare Optimierungsprobleme. Die Grundidee besteht darin, in jeder Iteration als Teilproblem ein *Quadratisches Programm* (QP) wie in (2.4) zu lösen. Dieses QP entsteht aus der Linearisierung der Nebenbedingungsfunktionen, womit dann die zugehörige Langrange-Funktion $L(\mathbf{x}, \ \mathbf{u})$ (2.7) quadratisch approximiert wird. Dieses QP ist dann wiederum schnell und effizient lösbar. Sei also $\mathbf{x}_0 \in \mathbb{R}^n$ gegeben, und $\mathbf{x}_k \in \mathbb{R}^n$ bereits eine Approximation der Lösung des Problems (2.6), $\mathbf{v}_k \in \mathbb{R}^m$ eine Approximation der Lagrange-Multiplikatoren

und $\mathbf{H}_k \in \mathbb{R}^{n \times n}$ eine Approximation der Hesse-Matrix der Lagrange-Funktion. Dann löst das SQP-Verfahren in jeder Iteration ein QP der Form:

$$\boxed{\begin{array}{l} \min\limits_{\mathbf{d} \in \mathbb{R}^n} \dfrac{1}{2}\mathbf{d}^{\mathrm{T}}\mathbf{H}_k\mathbf{d} + \nabla f(\mathbf{x}_k)^{\mathrm{T}}\mathbf{d} \\ \text{s.t.} \\ \nabla g(\mathbf{x}_k)^{\mathrm{T}}\mathbf{d} + g(\mathbf{x}_k) \geq 0 \end{array}} \tag{2.8}$$

Sei $\mathbf{d}_k$ die optimale Lösung von 2.8 (unter der Annahme, dass 2.8 stets lösbar ist), und sei $\mathbf{u}_k$ der zugehörige Lagrange-Multiplikator dieses Teilproblems. Der Iterationsschritt des SQP-Verfahrens besteht dann aus

$$\begin{aligned} \mathbf{x}_{k+1} &:= \mathbf{x}_k + \mathbf{d}_k \\ \mathbf{v}_{k+1} &:= \mathbf{u}_k \ . \end{aligned} \tag{2.9}$$

2.2.3 Optimalitätsbedingungen für Quadratische Programme mit Gleichheitsbedingungen

Enthält das Optimierungsproblem 2.6 nur Gleichheitsbedingungen der Form $g(\mathbf{x}) = 0$, so lassen sich die *Karush-Kuhn-Tucker-Optimalitätsbedingungen* (KKT) wie folgt aufschreiben:

$$F(\mathbf{x}, \mathbf{u}) := \begin{pmatrix} \nabla f(\mathbf{x}) - \nabla g(\mathbf{x})\mathbf{u} \\ g(\mathbf{x}) \end{pmatrix} = 0 \tag{2.10}$$

Die optimale Lösung und die zugehörigen Lagrange-Multiplikatoren sind also die Lösung eines Systems von $n+m$ nichtlinearen Gleichungen der Form $F(\mathbf{x}, \mathbf{u}) = 0$ mit $n+m$ Unbekannten, $\mathbf{x} \in \mathbb{R}^n$ und $\mathbf{u} \in \mathbb{R}^m$.

Mit der Newton-Methode erhält man für eine gegebene Approximation $(\mathbf{x}_k,\ \mathbf{v}_k)$ den nächsten Iterationsschritt $(\mathbf{d}_k,\ \mathbf{y}_k)$:

$$\nabla F(\mathbf{x}_k,\ \mathbf{v}_k) \begin{pmatrix} \mathbf{d}_k \\ \mathbf{y}_k \end{pmatrix} + F(\mathbf{x}_k,\ \mathbf{v}_k) = 0$$

Mit der abkürzenden Schreibweise $\mathbf{H}_k$ für die Hesse-Matrix der Lagrange-Funktion erhält man

$$\begin{pmatrix} \mathbf{H}_k & -\nabla g(\mathbf{x}_k) \\ \nabla g(\mathbf{x}_k)^{\mathrm{T}} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{d}_k \\ \mathbf{y}_k \end{pmatrix} + \begin{pmatrix} \nabla f(\mathbf{x}_k) - \nabla g(\mathbf{x}_k)\mathbf{v}_k \\ g(\mathbf{x}_k) \end{pmatrix} = 0 \ . \tag{2.11}$$

Mit der Definition $\mathbf{u}_k := \mathbf{y}_k + \mathbf{v}_k$ erhält man dann

$$\mathbf{H}_k\mathbf{d}_k - \nabla g(\mathbf{x}_k)\mathbf{u}_k + \nabla f(\mathbf{x}_k) = 0$$

und

$$\nabla g(\mathbf{x}_k)^{\mathrm{T}}\mathbf{d}_k + g(\mathbf{x}_k) = 0 \ ,$$

dies sind genau die Optimalitätskriterien für ein Quadratisches Programm mit Gleichheits-Nebenbedingungen. Ist $\mathbf{H}_k$ also die Hesse-Matrix der Lagrange-Funktion, und die Startlösung $\mathbf{x}_0$ nah genug an der optimalen Lösung, so ist die SQP-Methode nichts anderes als die Newton-Methode zum Lösen der KKT-Bedingungen.

2.2.4 Ungleichheitsbedingungen

Für Ungleichheits-Bedingungen erhält man folgendes Resultat: Wenn $\mathbf{d}_k = 0$ eine optimale Lösung des QPs (2.8) ist und $\mathbf{u}_k$ der entsprechende Vektor der Lagrange-Multiplikatoren, dann erfüllen $\mathbf{x}_k$ und $\mathbf{u}_k$ die notwendingen Optimalitätsbedingungen des Optimierungsproblems (2.6).

Da die Hesse-Matrix nach den Voraussetzungen an die Funktion und die Nebenbedingungen nicht existieren muss, wird sie durch $\mathbf{H}_k$ approximiert. $\mathbf{H}_k$ soll dabei positiv definit sein.

Um die Konvergenzeigenschaften zu verbessern, wird diese Approximation selbst in jedem Iterationsschritt wiederum durch eine quasi-Newton-Methode verfeinert, indem

$$\mathbf{H}_{k+1} := \mathbf{H}_k + \frac{\mathbf{y}_k\mathbf{y}_k^{\mathrm{T}}}{\mathbf{s}_k^{\mathrm{T}}\mathbf{y}_k} - \frac{\mathbf{H}_k\mathbf{s}_k\mathbf{s}_k^{\mathrm{T}}\mathbf{H}_k}{\mathbf{s}_k^{\mathrm{T}}\mathbf{H}_k\mathbf{s}_k} \tag{2.12}$$

gesetzt wird, wobei

$$\begin{aligned} \mathbf{y}_k &:= \nabla f(\mathbf{x}_{k+1}) - \nabla g(\mathbf{x}_{k+1})\mathbf{u} - (\nabla f(\mathbf{x}_k) - \nabla g(\mathbf{x}_k)\mathbf{u}) \\ \mathbf{s}_k &:= \mathbf{x}_{k+1} - \mathbf{x}_k \ . \end{aligned}$$

$\mathbf{H}_0$ kann als Einheitsmatrix gewählt werden. Die positive Definitheit der Matrizen $\mathbf{H}_k$ garantiert eindeutige Lösungen des Systems 2.11.

2.3 Startlösung und Stabilisierung des SQP-Verfahrens

In Abschnitt 2.2.2 war die Startlösung breits eine Approximation der optimalen Lösung. Im Fall einer beliebigen Startlösung wird das SQP-Verfahren mit verschiedenen Ansätzen „stabilisiert“, um möglichst Konvergenz herzustellen.

2.3.1 Reduzierte Hesse-SQP-Schritte

Bei vorhandener QR-Zerlegung

$$\nabla g(\mathbf{x}_k) = (\mathbf{Y}_k,\ \mathbf{Z}_k) \begin{pmatrix} \mathbf{R}_k \\ 0 \end{pmatrix} \ ,$$

wobei $\mathbf{p}_k = \mathbf{Y}_k^{\mathrm{T}}\mathbf{d}_k$ und $\mathbf{q}_k = \mathbf{Z}_k^{\mathrm{T}}\mathbf{d}_k$, kann das lineare Gleichungssystem 2.11 umgeschrieben werden als

$$\begin{pmatrix} \mathbf{Y}_k^T\mathbf{H}_k\mathbf{Y}_k & \mathbf{Y}_k^T\mathbf{H}_k\mathbf{Z}_k & -\mathbf{R}_k^T \\ \mathbf{Z}_k^T\mathbf{H}_k\mathbf{Y}_k & \mathbf{Z}_k^T\mathbf{H}_k\mathbf{Z}_k & 0 \\ -\mathbf{R}_k & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{p}_k \\ \mathbf{q}_k \\ \mathbf{y}_k \end{pmatrix} = \begin{pmatrix} -\mathbf{Y}_k^T\nabla f(\mathbf{x}_k) \\ -\mathbf{Z}_k^T\nabla f(\mathbf{x}_k) \\ g(\mathbf{x}_k) \end{pmatrix} .$$

Man sieht also leicht, dass $\mathbf{p}_k$ und $\mathbf{q}_k$ erhalten werden können durch Lösen von

$$\begin{pmatrix} \mathbf{Z}_k^{\mathrm{T}}\mathbf{H}_k\mathbf{Y}_k\mathbf{Z}_k^{\mathrm{T}}\mathbf{H}_k & \mathbf{Z}_k \\ -\mathbf{R}_k^{\mathrm{T}} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{p}_k \\ \mathbf{q}_k \end{pmatrix} = \begin{pmatrix} -\mathbf{Z}_k^T\nabla f(\mathbf{x}_k) \\ g(\mathbf{x}_k) \end{pmatrix} .$$

Dies zeigt, dass lediglich die reduzierte Matrix $\mathbf{Z}_k^T\mathbf{H}_k$ statt der vollen Matrix $\mathbf{H}_k$ notwendig ist, um den QP-Schritt $\mathbf{d}_k$ zu berechnen. Daher wird in [21] vorgeschlagen,

$\mathbf{Z}_k^{\mathrm{T}}\mathbf{H}_k$ durch eine quasi-Newton-Matrix $\overline{\mathbf{H}}_k$ zu ersetzen und so den QP-Schritt $\mathbf{d}_k$ durch Lösen von

$$\begin{pmatrix} \overline{\mathbf{H}}_k \\ -\nabla g(\mathbf{x}_k)^T \end{pmatrix} \mathbf{d}_k = \begin{pmatrix} -\mathbf{Z}_k^{\mathrm{T}}\nabla f(\mathbf{x}_k) \\ g(\mathbf{x}_k) \end{pmatrix}$$

zu vollziehen.

Die Matrix $\overline{\mathbf{H}}_k \in \mathbb{R}^{(n-m)\times n}$ ist eine quasi-Newton-Matrix, die $\mathbf{Z}_k^T\mathbf{H}_k$ approximiert, und kann mit dem Broyden-Verfahren [6] berechnet werden:

$$\overline{\mathbf{H}}_{k+1} := \overline{\mathbf{H}}_k + \frac{(\overline{\mathbf{y}}_k - \overline{\mathbf{H}}_k\mathbf{s}_k)\mathbf{s}_k^{\mathrm{T}}}{||\mathbf{s}_k||_2^2} ,$$

wobei $\mathbf{s}_k := \mathbf{x}_{k+1} - \mathbf{x}_k$ und $\overline{\mathbf{y}}_k := \mathbf{Z}_{k+1}^{\mathrm{T}}\nabla f(\mathbf{x}_{k+1}) - \mathbf{Z}_k^{\mathrm{T}}\nabla f(\mathbf{x}_k)$. Das Verwenden einer reduzierten Matrix mit Dimensionen $(n-m)\times n$ verringert den Speicherbedarf und die Komplexität der Berechnungen im SQP-Verfahren und kann das Verfahren so unter Umständen beschleunigen, besonders bei Problemen mit sehr vielen Variablen.

2.3.2 Liniensuche

Ein weiterer wichtiger Parameter für die Leistungsfähigkeit eines SQP-Verfahrens ist die Wahl der *Schrittlänge* in jedem Iterationsschritt. Mit der Definition $\mathbf{z}_k := (\mathbf{x}_k,\ \mathbf{v}_k)$, wobei $\mathbf{x}_k \in \mathbb{R}^n$ und $\mathbf{v}_k \in \mathbb{R}^m$ jeweils Approximationen eines KKT-Punktes im Schritt k sind, wird der nächste Iterationsschritt mit Schrittlänge $\alpha_k \in \mathbb{R}, \alpha_k > 0$ mit Hilfe der Vorschrift

$$\mathbf{z}_{k+1} := \mathbf{z}_k + \alpha_k\mathbf{p}_k$$

berechnet. Dabei gibt $\mathbf{p}_k := (\mathbf{d}_k,\ \mathbf{y}_k) = (\mathbf{d}_k,\ \mathbf{u}_k - \mathbf{v}_k)$ die Suchrichtung an, diese wird erhalten durch Lösen des QPs (2.11). Ziel ist, mit möglichst wenig Funktionsauswertungen eine Schrittlänge zu erhalten, die den Funktionswert einer sogenannten *Anpassungsfunktion (merit function)* ϕ_r verringert, um so, durch passende Wahl der Schrittlänge α, die Konvergenz des Algorithmus sicherzustellen:

$$\phi_r(\alpha) := \psi_r \left(\begin{pmatrix} \mathbf{x} \\ \mathbf{v} \end{pmatrix} + \alpha \begin{pmatrix} \mathbf{d} \\ \mathbf{u} - \mathbf{v} \end{pmatrix} \right) ,$$

mit einer passenden Straffunktion $\psi_r(\mathbf{x},\ \mathbf{v})$ und $\mathbf{r} = (r_1,\ \ldots,\ r_m)^T \in \mathbb{R}^m$, dem Vektor der m positiven Strafparamter. Eine solche Anpassungsfunktion soll Informationen sowohl über das zu minimierende Funktional enthalten als auch über verletzte Nebenbedingungen.

Üblicherweise werden *augmentierte Lagrange-Anpassungsfunktionen* verwendet, die in [5] definiert sind als:

$$\psi_r(\mathbf{x},\ \mathbf{v}) \ := f(\mathbf{x}) - \mathbf{v}^{\mathrm{T}} g(\mathbf{x}) + \frac{1}{2} \sum_{j=1}^{m} r_j g_j(\mathbf{x})^2$$

Weitere solcher Anpassungsfunktionen werden etwa in [26] vorgestellt. Allen Anpassungsfunktionen gemein ist, dass die Zielfunktion für verletzte Nebenbedinungen *bestraft* wird.

Je besser die Anpassungsfunktion auf das zu lösende Optimierungsproblem passt, umso besser konvergiert das entsprechende SQP-Verfahren.

2.4 Konvexe Optimierung

Das *Konvexe Optimierungsproblem* war definiert als

$$\boxed{\begin{array}{l} \min f(\mathbf{x}) \\ \text{s.t.} \\ \mathbf{x} \in \mathcal{Q} \ . \end{array}} \tag{2.13}$$

Dabei ist $\mathbf{x}$ ist ein n-dimensionaler reeller Vektor, das Gebiet $\mathcal{Q} \subseteq \mathbb{R}^n$ konvex und $f : \mathbb{R}^n \to \mathbb{R}$ eine konvexe, reellwertige Funktion. Ist $\mathbf{x} \in \mathcal{Q}$, so heißt $\mathbf{x}$ *zulässig*. Ein zulässiges $\mathbf{x}^*$ heißt *Globales Minimum* von f, wenn

$$f(\mathbf{x}^*) \leq f(\mathbf{x}) \quad \forall\ \mathbf{x} \in \mathcal{Q} \ .$$

Ein zulässiges $\mathbf{x}^*$ heißt *Lokales Minimum* von f, wenn eine Umgebung

$$B_\epsilon(\mathbf{x}) = \{\mathbf{z} \in \mathbb{R}^n : ||\mathbf{z} - \mathbf{x}|| < \epsilon\} \subseteq \mathcal{Q}$$

existiert mit

$$f(\mathbf{x}^*) \leq f(\mathbf{x}) \quad \forall\ \mathbf{x} \in B_\epsilon(\mathbf{x}^*) \ .$$

Entsprechend definiert sind ϵ-Approximationen dieser Minima.

Ist f eine *glatte* konvexe Funktion, so lassen sich Komplexitätsschranken solcher ϵ-Approximationen von 2.13 finden. Ist f etwa Lipschitz-stetig mit Lipschitz-Konstante L, so liegt das Problem 2.13 in der Komplexitätsklasse $O\left(\sqrt{\frac{L}{\epsilon}}\right)$, siehe etwa [24].

Für nicht glatte konvexe Funktionen ist die Komplexitätsklasse der ϵ-Approximation von 2.13 $O\left(\sqrt{\frac{1}{\epsilon^2}}\right)$. Für diese Laufzeitschranke muss nur ein *Orakel* für f existieren. Sie lässt sich in der Praxis unterbieten, wenn etwas über die Struktur des Optimierungsproblems bekannt ist.

Techniken zur Behandlung des Optimierungsproblems für nicht glatte, konvexe Funktionen finden sich in [24], etwa *Subgradienten-Verfahren.*

3 Subspace Correction

In diesem Kapitel wird die von [31] eingeführte Methode der *Subspace Correction* (Teilraumkorrektur) beschrieben. Es handelt sich dabei um eine Methode, die - zunächst zum Lösen großer lineare Gleichungssysteme - einen „Divide and Conquer"-Ansatz vorschlägt. Dieser Ansatz wird später verallgemeinert und lässt sich dann auch auf konvexe Optimierungsprobleme anwenden.

3.1 Lineare Gleichungssysteme

Der Teilraumkorrektur-Ansatz für lineare Gleichungssysteme betrachtet als Problemstellung ein System der Form

$$\mathbf{A}\mathbf{u} = \mathbf{f}\,. \tag{3.1}$$

dabei ist $\mathbf{A}$ ein positiv definiter Operator über einem endlichdimensionalen Vektorraum $\mathcal{V}$, und $\mathbf{u}, \mathbf{f} \in \mathcal{V}$. Der Teilraum-Korrektur-Ansatz bietet ein vereinheitlichtes Framework zur Lösung solcher Probleme, in welchem sich viele bereits bekannte Lösungsansätze wiederfinden, etwa Gauss-Seidel, Jacobi-Iteration und Multigrid-Ansätze.

Die Grundidee des Frameworks ist ein Einzelschrittverfahren, das aus der Lösung der vorhergehenden Iteration, $\mathbf{u}^{\text{alt}}$, eine neue Approximation $\mathbf{u}^{\text{neu}}$ berechnet:

Algorithmus 1 : Ein-Schritt-Iterationsverfahren

$\mathbf{r}^{\text{alt}} := \mathbf{f} - \mathbf{A}\mathbf{u}^{\text{alt}}$
$\hat{\mathbf{e}} := \mathbf{B}\mathbf{r}^{\text{alt}}$ mit $\mathbf{B} \approx \mathbf{A}^{-1}$ (Approximiere $\mathbf{A}\mathbf{e} = \mathbf{r}^{\text{alt}}$)
$\mathbf{u}^{\text{neu}} := \mathbf{u}^{\text{alt}} + \hat{\mathbf{e}}$

Der Vektor $\hat{\mathbf{e}}$ ist der dem Verfahren namensgebende „Korrekturterm". Der Kern des Verfahrens, die Berechnung einer approximierten Inversen $\mathbf{B}$ der Matrix $\mathbf{A}$, wird im Teilraum-Korrektur-Verfahren durch Lösen passender Probleme auf Teilräumen von $\mathcal{V}$ geleistet. Sei also

$$\mathcal{V} = \sum_{i=1}^{J} \mathcal{V}_i$$

eine Zerlegung von $\mathcal{V}$ in $1 \leq J \in \mathbb{N}$ Teilräume $\mathcal{V}_i$ von $\mathcal{V}$.

Jedes Element $\mathbf{v} \in \mathcal{V}$ besitzt also eine (nicht notwendigerweise eindeutige) Darstellung

$$\mathbf{v} = \sum_{i=1}^{J} \mathbf{v}_i \qquad \text{mit}$$
$$\mathbf{v}_i \in \mathcal{V}_i \qquad \text{für } 1 \leq i \leq J \ .$$

Mit $\mathbf{A}_i = \mathcal{V}_i \to \mathcal{V}_i$ sei die Einschränkung eines Operators $\mathbf{A}$ auf $\mathcal{V}_i$ bezeichnet. Dann können die Schritte in Algorithmus 1 in jedem Teilraum $\mathcal{V}_i$ ausgeführt werden, wobei $\mathbf{B} \approx \mathbf{A}_i^{-1}$ für $i = 1 \dots J$.

3.1.1 Definitionen

In [31] werden zunächst zwei Klassen von Lösungsansätzen des in 3.1 aufgestellten Problems betrachtet. Hierzu werden zusätzlich zum endlichdimensionalen Vektorraum $\mathcal{V}$ folgende Definitionen eingeführt:

- $L(\mathcal{V})$, der Raum der linearen Operatoren von $\mathcal{V} \to \mathcal{V}$
- Für einen gegebenen linearen Operator $\mathbf{A}$ auf $\mathcal{V}$
  - das Spektrum $\sigma(\mathbf{A})$ von $\mathbf{A}$
  - den Spektral-Radius $\rho(\mathbf{A})$ von $\mathbf{A}$
  - den minimalen und maximalen Eigenwert $\lambda_{\min}(\mathbf{A})$ und $\lambda_{\max}(\mathbf{A})$
- Innere Produkte $(\cdot, \cdot)$ mit induzierter Norm $\|\cdot, \cdot\|$

Weiterhin bedeuten $\lesssim$, $\gtrsim$ und $\cong$, dass Konstanten existieren, die die entsprechenden Vergleiche und Gleichheit wahrmachen.

Aus der linearen Algebra sind folgende Fakten und Definitionen bekannt: Wenn der Operator $\mathbf{A} \in L(\mathcal{V})$ *symmetrisch positiv definit (SPD)* bezüglich eines inneren Produktes $(\cdot, \cdot)$ ist, so sind alle dessen Eigenvektoren positiv, und es gilt

$$\lambda_{\min}(\mathbf{A}) = \min_{\mathbf{v} \in \mathcal{V} \setminus 0} \frac{(\mathbf{A}\mathbf{v}, \mathbf{v})}{\|\mathbf{v}\|^2} \ , \qquad \lambda_{\max}(\mathbf{A}) = \max_{\mathbf{v} \in \mathcal{V} \setminus 0} \frac{(\mathbf{A}\mathbf{v}, \mathbf{v})}{\|\mathbf{v}\|^2} \ .$$

Mit $\kappa(\mathbf{A}) = \lambda_{\max}(\mathbf{A}) / \lambda_{\min}(\mathbf{A})$ sei die Konditionszahl von $\mathbf{A}$ bezeichnet.

Es wird für jeden Operator der betrachteten Problemstellung 3.1 angenommen, dass ein inneres Produkt exisitert, bezüglich dem der Operator SPD ist.

Ein Operator $\mathbf{A} \in L(\mathcal{V})$ heißt *selbstadjungierend*, wenn gilt

$$(\mathbf{A}\mathbf{u}, \mathbf{v}) = (\mathbf{u}, \mathbf{A}^{\mathrm{T}}\mathbf{v}) \quad \forall \quad \mathbf{u}, \mathbf{v} \in \mathcal{V} \ .$$

Wenn $\mathbf{A}, \mathbf{B} \in \mathcal{V}$ beide SPD sind, so ist ihr Produkt $\mathbf{B}\mathbf{A}$ selbstadjungierend bezüglich der von $\mathbf{A}$ induzierten Metrik

$$(\cdot, \cdot)_{\mathbf{A}} := (\mathbf{A}\cdot, \cdot) \ .$$

3.1.2 Beispiel: Konjugierte Gradienten

Das Verfahren der *Konjugierten Gradienten (CG)* nutzt aus, dass Lösen von

$$\mathbf{A}\mathbf{u} = \mathbf{f}$$

äquivalent ist zum Lösen des Minimierungsproblems

$$\min E(\mathbf{x}) = \frac{1}{2}(\mathbf{A}\mathbf{x}, \mathbf{x}) - (\mathbf{f}, \mathbf{x}) \ .$$

Die quadratische Form $E(x)$ wird nun minimiert, indem nach Wahl eines Startwertes $\mathbf{u}_0$ sukzessive eine $\mathbf{A}$-konjugierte Basis $\boldsymbol{\alpha}^d, d = 1 \dots J$, des affinen Raums $\mathcal{V}_k$ berechnet wird. Für die mit $\mathbf{u}_k$ bezeichnete k-te Iterierte des CG-Verfahrens gilt dann

$$\|\mathbf{u} - \mathbf{u}_k\| \leq 2 \left(\frac{\sqrt{\kappa(\mathbf{A})} - 1}{\sqrt{\kappa(\mathbf{A})} + 1} \right)^k \|\mathbf{u} - \mathbf{u}_0\|_{\mathbf{A}} \ .$$

Auch weitere klassische iterative Lösungsmethoden können als Einzelschrittverfahren der Form 1 geschrieben werden:

- Richardson-Iteration: $b_i = \frac{1}{\omega} > 0$ mit einem $\omega \in \mathbb{R}$
- Jacobi-Methdode: $b_i = a_{ii}$, wobei a_{ii} der i-te Diagonaleintrag von $\mathbf{A}$ ist

3.2 Teilraumzerlegung und Projektionen

Zu einer gegebenen Zerlegung

$$\mathcal{V} = \sum_{i=1}^{J} \mathcal{V}_i$$

von $\mathcal{V}$ in J Teilräume $\mathcal{V}_i \subset \mathcal{V}$ lassen sich nun die passenden orthogonalen Projektionen auf die einzelnen Teilräume definieren durch

$$\begin{aligned} \mathbf{Q}_i &: \mathcal{V} \mapsto \mathcal{V}_i \\ (\mathbf{Q}_i\mathbf{u}, \mathbf{v}_i) &= (\mathbf{u}, \mathbf{v}_i) \ , \quad \mathbf{u} \in \mathcal{V}, \mathbf{v}_i \in \mathcal{V}_i \end{aligned} \tag{3.2}$$

und

$$\begin{aligned} \mathbf{P}_i &: \mathcal{V} \mapsto \mathcal{V}_i \\ (\mathbf{P}_i\mathbf{u}, \mathbf{v}_i)_{\mathbf{A}} &= (\mathbf{u}, \mathbf{v}_i)_{\mathbf{A}} \ , \quad \mathbf{u} \in \mathcal{V}, \mathbf{v}_i \in \mathcal{V}_i \ . \end{aligned}$$

$\mathbf{A}_i$ sei weiterhin die Einschränkung von $\mathbf{A}$ auf $\mathcal{V}_i$:

$$\begin{aligned} \mathbf{A}_i &: \mathcal{V}_i \mapsto \mathcal{V}_i \\ (\mathbf{A}_i\mathbf{u}_i, \ \mathbf{v}_i) &= (\mathbf{A}\mathbf{u}_i, \ \mathbf{v}_i), \quad \mathbf{u}_i, \ \mathbf{v}_i \in \mathcal{V}_i \ . \end{aligned}$$

$\mathbf{A}_i$ ist dabei wieder symmetrisch und positiv definit, genau wie $\mathbf{A}$.

3.3 Subspace-Correction-Methoden

Mit den im vorigen Abschnitt definierten Projektionen lässt sich nun das Prinzip der Subspace Correction beschreiben. Grundidee ist, auf jedem Teilraum eine approximierte Inverse der Einschränkung von $\mathbf{A}$ auf diesen Teilraum zu finden, um daraus eine Approximation der Inversen von $\mathbf{A}$ zu konstruieren. Aus der Definition der Projektionen folgt sofort die Gleichung

$$\mathbf{A}_i\mathbf{P}_i = \mathbf{Q}_i\mathbf{A} \ .$$

Sei nun für ein $\mathbf{u} \in \mathcal{V}$ definiert: $\mathbf{u}_i = \mathbf{P}_i\mathbf{u}$, und für die rechte Seite $\mathbf{f}$ des Gleichungssystems (3.1) analog $\mathbf{f}_i = \mathbf{Q}_i$. Damit hat jede optimale Lösung $\mathbf{u}$ von (3.1) die Eigenschaft, dass auf den einzelnen Teilräumen $\mathcal{V}_i$ gilt:

$$\mathbf{A}_i\mathbf{u}_i = \mathbf{f}_i \tag{3.3}$$

Umgekehrt sei für jeden Teilraum $\mathcal{V}_i$ der Operator $\mathbf{R}_i$ als approximierte Inverse von $\mathbf{A}_i$ definiert:

$$\mathbf{R}_i : \mathcal{V}_i \mapsto \mathcal{V}_i \tag{3.4}$$

Mit dieser Definition lässt sich eine approximierte Lösung der Gleichung (3.3) schreiben als

$$\hat{\mathbf{u}}_i = \mathbf{R}_i\mathbf{f}_i \ .$$

3.3.1 Parallele Subspace-Correction-Methoden

Das erste der beiden in [31] vorgestellten Verfahren ist die Parallele Subspace-Correction-Methode. Sei dazu $\mathbf{u}^{\text{alt}}$ als Startlösung eine bereits gegebene Approximation der Lösung $\mathbf{u}$ des Gleichungssystems (3.1).

Als Maß für die Güte der Approximation dient das *Residuum*, definiert als

$$\mathbf{r}^{\text{alt}} = \mathbf{f} - \mathbf{A}\mathbf{u}^{\text{alt}} \ .$$

In der optimalen Lösung gilt also $\mathbf{r} = 0$. Ansonsten wird die *Residualgleichung*

$$\mathbf{A}\mathbf{e} = \mathbf{r}^{\text{alt}}$$

approximativ gelöst, um eine bessere Lösung zu erhalten. Mit dieser Definition gilt dann auch, dass $\mathbf{u} = \mathbf{u}^{\text{alt}} + \mathbf{e}$ die exakte Lösung von (3.1) ist.

Im Subspace-Correction-Ansatz wird diese Residualgleichung aber nun auf jedem Teilraum $\mathcal{V}_i$ separat gelöst, als Gleichungssystem der Form

$$\mathbf{A}_i \mathbf{e}_i = \mathbf{Q}_i \mathbf{r}^{\text{alt}}. \tag{3.5}$$

Mit der zuvor in (3.4) definierten approximierten Inversen $\mathbf{R}_i$ von $\mathbf{A}_i$, und der in (3.2) definierten Projektion $\mathbf{Q}_i$ gilt dann, dass die approximierte Lösung der Teilraum-Gleichung (3.5) sich schreiben lässt als

$$\hat{\mathbf{e}}_i = \mathbf{R}_i \mathbf{Q}_i \mathbf{r}^{\text{alt}} \ .$$

Nachdem die Teilraum-Gleichungen auf jedem der J Teilräume gelöst wurden, wird die Gesamtlösung zusammengesetzt durch

$$\mathbf{u}^{\text{neu}} = \mathbf{u}^{\text{alt}} + \sum_{i=1}^{J} \hat{\mathbf{e}}_i \ .$$

Mit der Definition

$$\mathbf{B} = \sum_{i=1}^{J} \mathbf{R}_i \mathbf{Q}_i$$

ergibt sich dann die Kurzschreibweise

$$\mathbf{u}^{\text{neu}} = \mathbf{u}^{\text{alt}} + \mathbf{B}(\mathbf{f} - \mathbf{A}\mathbf{u}^{\text{alt}}) \ .$$

Der Parallele-Subspace-Correction-Algorithmus hat also die in Algorithmus 2 gezeigte Form.

3.3.2 Sukzessive Subspace-Correction-Methoden

Im Parallelen Subspace-Correction-Verfahren ist das Lösen der Residualgleichung auf einem Teilraum unabhängig von den Lösungen auf den übrigen Teilräumen, daher ist das Verfahren leicht parallelisierbar. Der Ansatz der *Sukzessiven Subspace-Correction* dagegen nutzt die Kenntnis der Teilraum-Lösungen aus den vorhergehenden Teilräumen, um die aktuelle Teilraum-Gleichung zu lösen: Sei $\mathbf{v}^i$ entweder die Startlösung ($i = 0$) oder eine bereits berechnete Lösung für Teilraum i, dann berechnet sich die Lösung für Teilraum $i + 1$ durch

$$\begin{aligned} \mathbf{v}^{i+1} &= \mathbf{v}^i + \mathbf{R}_i \mathbf{Q}_i \mathbf{r}_i \\ &= \mathbf{v}^i + \mathbf{R}_i \mathbf{Q}_i (\mathbf{f} - \mathbf{A}\mathbf{v}^i) \ . \end{aligned}$$

Der entsprechende Algorithmus für die Sukzessive Subspace-Correction ist Algorithmus 3.

Algorithmus 2 : Parallele Subspace-Correction (PSC)

Input : Lineares Gleichungssystem $\mathbf{Ax} = \mathbf{f}$, Startlösung $\mathbf{u}^{(0)}$.
Output : Vektor $\mathbf{u}_k$ mit $\mathbf{Au}_k \approx \mathbf{f}$.
for $k = 1, 2, \ldots$ **do**
  Setze $\mathbf{r}^{(0)} := \mathbf{f} - \mathbf{Au}^{(0)}$
  for $i = 1$ *to* J **do**
    Berechne parallel für alle i jeweils die approximative Inverse R_i
    Setze parallel für alle i:

$$\hat{\mathbf{e}}^{(k,i)} = \mathbf{R}_i \mathbf{Q}_i \mathbf{r}^k$$

  end
  Setze $\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \sum_{i=1}^{J} \hat{\mathbf{e}}^{(k,i)}$
end

Algorithmus 3 : Sukzessive Subspace-Correction (SSC)

Input : Lineares Gleichungssystem $\mathbf{Ax} = \mathbf{f}$, Startlösung $\mathbf{u}^{(0)}$.
Output : Vektor $\mathbf{u}_k$ mit $\mathbf{Au}_k \approx \mathbf{f}$.
for $k = 1, 2, \ldots$ **do**
  Setze $\mathbf{r}^{(0)} := \mathbf{f} - \mathbf{Au}^{(0)}$
  for $i = 1$ *to* J **do**
    Berechne:

$$\mathbf{u}^{(k+i/J)} = \mathbf{u}^{(k+(i-1)/J)} + \mathbf{R}_i \mathbf{Q}_i (\mathbf{f} - \mathbf{Au}^{(k+(i-1)/J)})$$

  end
  Setze $\mathbf{u}^{(k+1)} = \mathbf{u}^{(k)} + \sum_{i=1}^{J} \hat{\mathbf{e}}^{(k,i)}$
end

3.4 Beispiele

Für endliche reelle Vektorräume, $\mathcal{V} = \mathbb{R}^n$, ist die naheliegendeste Dekomposition die in die kanonischen Basisvektoren $\mathbf{e}^i$:

$$\mathbb{R}^n = \sum_{i=1}^{n} \operatorname{span}\left\{\mathbf{e}^i\right\} \tag{3.6}$$

Ist nun eine symmetrisch positiv definite Matrix $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{n\times n}$ gegeben, so lauten die Matrizen für die Teilraumgleichungen $(i = 1, \ldots, J)$

$$\mathbf{A}_i = a_{ii}$$

und die orthogonalen Projektionen für einen Vektor $\mathbf{x} \in \mathbb{R}^n, x = (x_1, \ldots, x_n)^{\mathrm{T}}$ jeweils

$$\mathbf{Q}_i\mathbf{x} = \mathbf{x}_i\mathbf{e}^i \ .$$

Der Algorithmus 2 für die Parallele Subspace-Correction entspricht dann genau dem Jacobi-Verfahren, der Algorithmus 3 dagegen dem Gauß-Seidel-Verfahren.

3.5 Vergallgemeinerung: Stable Space Splittings und Schwarz-Methoden

Einen allgemeineren Betrachtungsrahmen für Techniken für Teilraum-Zerlegung bietet das Konzept der *Stable Space Splittings*. Hier ist zunächst als zu lösendes Problem ein Variationsproblem über einem Hilbertraum definiert. Analog zur Subspace Correction wird dieser Raum in Teilräume zerlegt, dabei muss die Zerlegung gewisse Bedigungen erfüllen. Tatsächlich sind die Subspace-Correction-Methoden in der Sprache der Stable Space Splittings Spezialfälle solcher Stable Splittings; es lassen sich sogenannte Additive und Multiplikative Schwarz-Methoden formulieren, die jeweils als Sonderfall die Parallele bzw. Sukzessive Subspace-Correction enthalten.

4 Optimierung per Teilraumzerlegung

Während in Kapitel 3 Systeme linearer Gleichungen und Variationsprobleme über Bilinearformen betrachtet wurden, sollen nun mitdem selben Ansatz der Teilraumzerlegung *Optimierungsprobleme* - konkret Minimierung - über konvexen Funktionen approximativ gelöst werden.

Dieser Ansatz erlaubt es, Probleme aus praktischen Anwendungen wie der Tracking-Error-Minimierung aus Kapitel 6 effizient zu approximieren.

4.1 Ansatz

In [28] wird ein Ansatz zur Erweiterung der Subspace-Correction auf konvexe Optimierungsprobleme beschrieben, welcher im Folgenden vorgestellt werden soll.

Der Ansatz basiert wieder darauf, das ursprüngliche Optimierungsproblem in Teilprobleme zu zerlegen und gleichzeitig sicherzustellen, dass sich das Gesamtergebnis in jeder Iteration verbessert, der Zielfunktionswert der Folge der Lösungen also monoton fällt.

Anzumerken ist weiterhin, dass die folgenden Konvergenzbeweise aus [28] im Gegensatz zu vielen anderen Konvergenzergebnissen auf diesem Gebiet *nicht* davon abhängen, wie nah die Startlösung am Optimum des Problems liegt, sondern uniform gelten.

4.2 Problemstellung und Definitionen

Gegeben sei ein Hilbertraum $\mathcal{V}$ und ein konvexes Funktional

$$F : \mathcal{V} \to \mathbb{R} \ . \tag{4.1}$$

Das betrachtete unbeschränkte konvexe Optimierungsproblem lautet dann

$$\min_{v \in \mathcal{V}} F(v) \ . \tag{4.2}$$

An das Funktional F sind einige Zusatzanforderungen gestellt. Zunächst muss F Gâteaux-differenzierbar [4] sein.

Weiterhin müssen Konstanten $K, L > 0$, $p \geq q > 1$ existieren, sodass

$$\begin{aligned} \langle F'(w) - F'(v), w - v \rangle &\geq K \left\| w - q \right\|_V^p \quad \forall w,\ v \in \mathcal{V} \text{ und} \\ \left\| F'(w) - F'(v), w - v \right\|_{V'} &\leq L \left\| w - q \right\|_V^{q-1} \quad \forall w,\ v \in \mathcal{V} \ . \end{aligned} \tag{4.3}$$

$\langle \cdot, \cdot \rangle$ sei die Dualitäts-Paarung von $\mathcal{V}$ mit seinem Dualraum $\mathcal{V}'$.

Aus den obigen Ungleichungen und den Eigenschaften der Norm des Dualraums folgt sofort

$$K \left\| w - q \right\|_V^p \leq \langle F'(w) - F'(v), w - v \rangle \leq L \left\| w - q \right\|_V^q \ .$$

Aus den Bedingungen (4.3) folgt, wie etwa in [16] bewiesen, die eindeutige Lösbarkeit des Optimierungsproblems (4.2).

4.3 Subspace-Correction-Methoden

Nachdem nun grundlegende Eigenschaften des Optimierungsproblems definiert sind, werden zwei Lösungsmethoden dafür vorgestellt, die beide auf Teilraumzerlegung beruhen.

Seien also $\mathcal{V}_i \subset \mathcal{V}, i = 1, 2, \ldots, m$, $m \in \mathbb{N}$ abgeschlossene Teilräume, sodass gilt

$$\mathcal{V} = \sum_{i=1}^{m} \mathcal{V}_i \ . \tag{4.4}$$

Somit hat jedes Element $v \in \mathcal{V}$ eine nicht notwendigerweise eindeutige Darstellung als

$$v = \sum_{i=1}^{m} v_i \ . \tag{4.5}$$

4.3.1 Parallele Subspace-Correction (PSC)

Die erste vorgestellte iterative Lösungsmethode ist die *Parallele Subspace-Correction*, analog zu Methode 2 für lineare Gleichungssysteme. Der Pseudocode des Algorithmus ist in Algorithmus 4 abgebildet. Das PSC-Verfahren ist also ein additives Verfahren.

4.3.2 Sukzessive Subspace-Correction (SSC)

Die *Sukzessive Subspace-Correction* für konvexe Optimierungsprobleme ist das Analogon zu Algorithmus 3. Das Verfahren in Pseudocode ist in Algorithmus 5 abgebildet.

Startlösung für Teilraum $i + 1$ ist also die Lösung aus dem vorherigen Iterationsschritt für Teilraum i. Das SSC-Verfahren ist dementsprechend ein multiplikatives Verfahren.

Beide Teilraumprobleme (4.6) und (4.7) sind durch die Anforderungen, die an die Funktion F gestellt wurden, eindeutig lösbar.

Algorithmus 4 : Parallele Subspace Correction (PSC)

Wähle Startwert $u^0 \in \mathcal{V}$ und Relaxierungsparameter $\alpha_i > 0$ sodass $\sum_{i=1}^{m} \alpha_i \leq 1$.
Iteriere:

1. Wenn für $n \geq 0$ bereits $u^n \in \mathcal{V}$ bestimmt wurde, so finde parallel für alle $i = 1, \ldots, m$ Richtungen $e_i^n \in V_i$, sodass gilt

$$F(u^n + e_i^n) \leq F(u^n + v_i) \quad \forall v_i \in \mathcal{V}_i \ , \tag{4.6}$$

2. Setze die neue Lösung u^{n+1} auf

$$u^{n+1} = u^n + \sum_{i=1}^{m} \alpha_i e_i^n \ .$$

Algorithmus 5 : Sukzessive Subspace Correction (SSC)

Wähle einen Startwert $u^0 \in \mathcal{V}$. Iteriere:

1. Wenn für $n \geq 0$ bereits $u^n \in \mathcal{V}$ bestimmt wurde, so finde sequentiell für $i = 1, \ldots, m$

$$u^{n+i/m} = u^{n+(i-1)/m} + e_i^n \tag{4.7}$$

mit $e_i^n \in \mathcal{V}_i$, sodass gilt

$$F(u^{n+(i-1)/m} + e_i^n) \leq F(u^{n+(i-1)/m} + v_i) \quad \forall v_i \in \mathcal{V}_i \ .$$

4.3.3 Charakteristika der Teilraumzerlegung

Um die folgende Konvergenz-Analyse greifbarer zu machen, werden zunächst zwei Konstanten eingeführt, die die Teilraumzerlegung charakterisieren. Die erste Konstante, C_1, sei die kleinste Konstante, für die gilt: Für jedes $v \in \mathcal{V}$ existieren $v_i \in \mathcal{V}_i$ mit

$$v = \sum_{i=1}^{m} v_i \ ,$$
$$\left(\sum_{i=1}^{m} \|v_i\|_{\mathcal{V}}^{\sigma} \right)^{\frac{1}{\sigma}} \leq C_1 \|v\|_{\mathcal{V}} \ . \tag{4.8}$$

Die Existenz einer solchen Konstante in endlichdimensionalen Banachräumen, wie sie bei der Minimierung des Tracking Error in Kapitel 6 auftreten, ist leicht zu zeigen.

Die zweite Konstante, C_2, sei die kleinste Konstante, für die gilt: Für jedes $w_{ij} \in \mathcal{V}, u_i \in \mathcal{V}_i, v_j \in \mathcal{V}_j$ gilt die Ungleichung

$$\sum_{i,j=1}^{m} \langle F'(w_{ij} + u_i) - F'(w_{ij}), v_j \rangle \leq C_2 \cdot \left(\sum_{i=1}^{m} \|u_i\|_{\mathcal{V}}^{p} \right)^{\frac{q-1}{p}} \cdot \left(\sum_{j=1}^{m} \|v_j\|_{\mathcal{V}}^{\sigma} \right)^{\frac{1}{\sigma}} \ . \tag{4.9}$$

Die Existenz eines solchen C_2 folgt aus den Annahmen (4.3), insbesondere führt Anwenden der Hölder-Ungleichung zu

$$C_2 \leq Lm \ .$$

4.4 Konvergenz-Analyse

Sei u die exakte Lösung des konvexen Minimierungsproblems (4.2) und u^n die Lösung des n-ten Schrittes eines der beiden Verfahren PSC (Algorithmus 4) oder SSC (Algorithmus 5). Zur Fehlerabschätzung für die beiden Subspace-Correction-Verfahren diene

$$d_n = F(u^n) - F(u) \tag{4.10}$$

als Maß für den Fehler in der n-ten Iteration.

Satz über uniforme Konvergenz

Hauptergebnis in [28] ist nun folgender Satz:

Satz 4.1. *Für eine Funktion F, die die Bedingungen* (4.3) *zur eindeutigen Lösbarkeit des Minimierungsproblems 4.2 und die Bedingungen für die Konstanten C_1 und C_2,* (4.8) *und* (4.9)*, efüllt, gilt für die beiden Algorithmen PSC und SSC:*

1. *Wenn $p = q$, so existiert eine Konstante $\delta \in (0, 1)$, die von p, q, K, L, C_1 und C_2 abhängt, sodass*

$$d_n \leq \delta d_{n-1} \leq \delta^n d_0 \ \forall n \geq 1 \ . \tag{4.11}$$

2. *Wenn $p > q$, so existiert eine Konstante $c_0 > 0$, die von d_0, p, q, K, L, C_1 und C_2 abhängt, sodass*

$$d_n \leq \frac{d_{n-1}}{(1 + c_0 d_{n-1}^{r-1})^{\frac{1}{r-1}}} \leq \frac{d_0}{(1 + c_0 d_0^{r-1} n)^{\frac{1}{r-1}}} \ \forall n \geq 1 \ . \tag{4.12}$$

Der Satz besagt, dass für $p = q$ beide Algorithmen geometrisch konvergieren. Für den Fall $p > q$ kann die Konvergenz sehr langsam sein.

5 Mathematische Modelle in der Finanzmathematik

Das Gebiet der *Finanzmathematik* beschäftigt sich mit der mathematischen Modellierung von Finanzmärkten. Hauptanliegen sind Modelle, die Aktienkursentwicklungen beschreiben und auch, soweit möglich, die Vorhersagen solcher erlauben. Im nächsten Abschnitt sollen zunächst die grundlegenden Begriffe eingeführt werden. Darauf aufbauend wird in Abschnitt 5.2 beschrieben, wie ein einzelner Aktienkurs modelliert werden kann. Als angewandte Disziplin ist die Finanzmathematik problemorientiert; Ziele sind etwa die Berechnung einer möglichst - im Sinne eines Marktes - gewinnträchtigen Handelsstrategie oder Risikominimierung. In Abschnitt 5.3 wird ein Überblick über verschiedene Handelsstrategien und deren zugrundeliegende Paradigmen gegeben.

5.1 Grundlegende Begriffe

5.1.1 Aktien und Portfolios

Kernobjekt der Finanzmathematik ist die *Aktie*, ein Wertpapier, das den Inhaber als Eigentümer eines Anteils eines Unternehmens - einer Aktiengesellschaft - ausweist. Hierbei wird unterschieden zwischen *Nennwertaktien*, die einem festen Wert des Grundkapitals des Unternehmens entsprechen, und *Stückaktien*, die keinen Nennwert besitzen, sondern lediglich den Anteil am Grundkapital widerspiegeln. Ihr theoretischer Nennwert ergibt sich aus der Gesamtzahl der existierenden Stückaktien des Unternehmens und dessen Grundkapital.

Was im Alltag mit „Wert“ einer Aktie bezeichnet wird, bezieht sich jedoch im Allgemeinen nicht auf deren Nennwert, sondern auf den *Börsenkurs* der Aktie. Dieser wird an einer *Wertpapierbörse* bestimmt, etwa der Frankfurter Wertpapierbörse. Diese ist ein organisierter Markt, welcher - im Fall der Frankfurter Wertpapierbörse dem deutschen Wertpapierhandelsgesetz folgend - die Kursfeststellung einer Aktie wie folgt vornimmt: Sogenannte *Orders*, Kauf- und Verkaufswünsche, jeweils mit eigenen Mindest- und Höchstpreisen, werden innerhalb eines Zeitraums gesammelt. Jener Preis der Aktie, der durch Erfüllen solcher Orders für den höchsten Umsatz an Aktien sorgt, ist der *Börsenkurs* dieser Aktie. Diese Berechnung wird computergestützt durchgeführt, an der Frankfurter Wertpapierbörse durch das Xetra-System. Letztendlich bestimmen also Angebot und Nachfrage, ganz wie von Léon Walras [29] gefordert, den Börsenkurs. Nachfolgend werden Wert und Börsenkurs einer Aktie zu einem Zeitpunkt synonym verwendet.

Eine Menge von Aktien verschiedener Unternehmen wird als *Portfolio* bezeichnet. Das Finden einer optimalen Zusammensetzung eines solchen Portfolios unter verschiedenen Gesichtspunkten - Gewinnmaximierung, Risikominimierung, Diversität - ist eines der Hauptprobleme der Finanzmathematik. Auch in der in dieser Arbeit betrachteten Problemstellung der Tracking-Error-Minimierung geht es um die Erstellung eines Portfolios.

5.1.2 Aktienindizes

Ein *Aktienindex* ist eine Vergleichszahl, die die Entwicklung eines Marktes abbilden soll. Hierzu wird zu diesem Markt ein virtuelles Portfolio erstellt. Je nach Index werden alle Unternehmen eines Marktes in den Index aufgenommen, häufig aber nur eine Auswahl, etwa die größten und/oder umsatzstärksten. Die Werte der in diesem Portfolio enthaltenen Aktien werden zum Wert des Aktienindex summiert. Prominentes Beispiel eines Aktienindex ist der DAX, der Deutsche Aktien IndeX. Dessen virtuelles Portfolio setzt sich zusammen aus den dreißig umsatzstärksten deutschen Firmen, die an der Frankfurter Wertpapierbörse gehandelt werden. Die Zusammensetzung und Gewichtung des Portfolios ist in Tabelle 5.1 zu sehen. Der Verlauf des DAX für den Zeitraum vom 01.04.2012 bis 01.04.2013 ist in Abbildung 5.1 zu sehen. Der DAX ist ein sogenannter *Performance-Index*, dies bedeutet, dass Dividenden und ähnliche Einnahmen aus den Aktien des DAX wieder in den Index re-investiert werden.

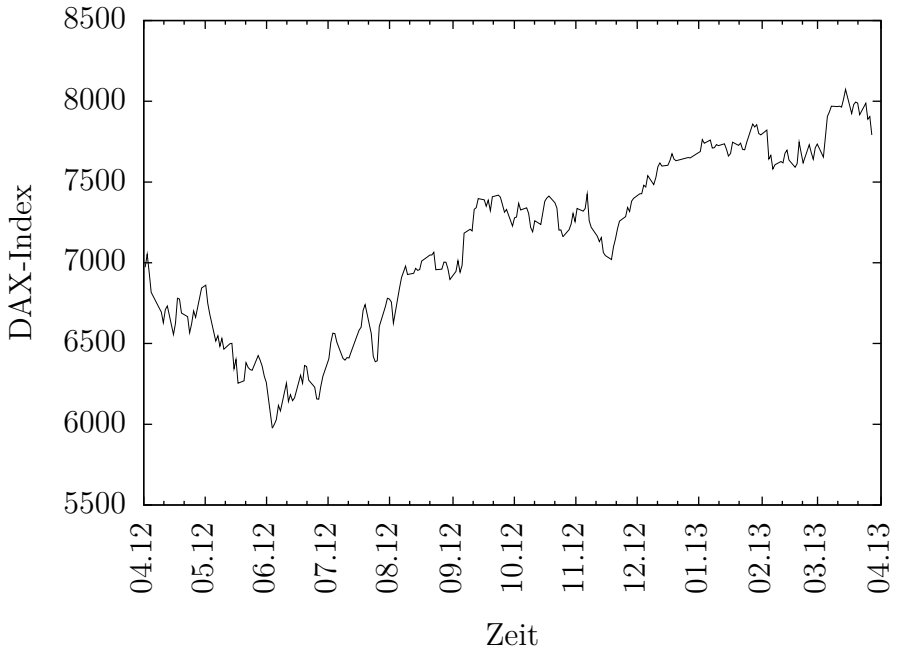

Abbildung 5.1: Verlauf des Dax vom 01.04.2012 bis zum 01.04.2013

5.1.3 Rendite

Zu einer gegebenen Aktie lässt sich der über einen Zeitraum erzielte Gewinn oder Verlust - die *Rendite* - wie folgt berechnen: Seien s und t Anfangs- und Endzeitpunkt des zu betrachtenden Zeitraums. Sei S_s der Börsenkurs der Aktie zum Zeitpunkt s, S_t der Börsenkurs der Aktie zum Zeitpunkt t. Dann ist die Rendite $R_{s,t}$ zwischen s und t definiert durch

$$R_{s,t} := \frac{S_t - S_s}{S_s} \ . \tag{5.1}$$

Die Rendite eines gesamten Marktes über einen Zeitraum wird *Marktrendite* genannt. Als Kennzahlen der Entwicklung von Märkten wurden die Aktienindizes eingeführt. Folglich wird die Marktrendite über die Rendite eines solchen Index - je nach betrachtetem Markt oder Teilmarkt - berechnet.

5.2 Modellierung von Aktienkursen

Jedes Modell, das Aktienkurse vorhersagen soll, muss mit Hilfe von unvollständigen Informationen versuchen, zukünftige Entwicklungen vorauszusagen. Vollständige Information über alle Faktoren, die den Börsenkurs einer Aktie beinflussen können, zu gewinnen, ist schlicht unmöglich - allein schon da Kauf- und Verkaufsentscheidungen letztlich von Menschen mit freiem Willen getroffen werden. Daher müssen zur Kursmodellierung zumindest einige einschränkende Annahmen getroffen werden. Im Folgenden sei davon ausgegangen, dass jegliche Investitionsentscheidungen ohne „Insiderwissen“, etwa über den bevorstehenden Kauf oder Verkauf großer Mengen einer Aktie durch Dritte, getroffen werden müssen. Das Fehlen von vollständiger Information wird dadurch beschrieben, dass ein Aktienkurs als *Zufallsvariable* modelliert wird.

5.2.1 Stochastische Prozesse und Wiener-Prozesse

Der nach dem Mathematiker Norbert Wiener benannte *Wiener-Prozess*, auch geometrische Brownsche Bewegung genannt, ist ein zeitstetiger *stochastischer Prozess*, der unabhängig normalverteilte Zuwächse hat.

Sei $\Omega = (\Omega, \Sigma, \mathcal{P})$ ein Wahrscheinlichkeitsraum und T eine Indexmenge. Ein reellwertiger *stochastischer Prozess* $\mathbf{X}$ ist eine Familie von Zufallsvariablen

$$\mathbf{X}_t : \Omega \to \mathbb{R}, \quad t \in T \ .$$

Die Indexmenge T soll einen Zeitverlauf widerspiegeln. Ist also T abzählbar, so spricht man von einem Zeit-diskreten Prozess, ist T überabzählbar, von einem zeitstetigen Prozess. Für jedes $\omega \in \Omega$ erzeugt der stochastische Prozess $\mathbf{X}$ eine Abbildung

$$\begin{aligned} T &\to \mathbb{R} \\ t \in T &\mapsto \mathbf{X}_t\omega \ , \end{aligned}$$

den zu ω gehörigen *Pfad* des Prozesses.

Ein Wiener-Prozess W ist nun ein zeitstetiger stochastischer Prozess mit $T = \mathbb{R}^+$ auf einem Wahrscheinlichkeitsraum Ω, für den gilt:

1. $\mathcal{P}(W_0 = 0) = 1$
2. Für gegebene Zeitpunkte $0 \leq t_1 \leq \cdots \leq t_n$ sind die Zufallsvariablen $W_{t_i+1} - W_{t_i}, i = 2, \ldots, n$, also die Zuwächse zwischen den Zeitpunkten t_{i+1} und t_i, stochastisch unabhängig.
3. Für die Verteilung der Zuwächse gilt $W_t - W_s \sim \mathcal{N}(0, t - s)$, sie sind also normalverteilt mit Erwartungswert 0 und Varianz $t - s$.
4. Die Pfade des Prozesses sind fast sicher stetig.

5.2.2 Das Black-Scholes-Modell

Das verbreiteteste Modell zur Modellierung eines Aktienkurses ist das *Black-Scholes-Modell*. Grundlegende Annahme ist, dass der natürliche Logarithmus des Kurs-Verlaufs die Form eines Wiener-Prozesses hat. In diesem Modell gilt für einen Aktienkurs S mit zum Zeitpunkt S_0 bekanntem Börsenkurs, mit erwarteter Rendite μ und *Volatilität* σ:

$$\mathrm{d}S_t = \mu S_t \mathrm{d}t + \sigma S_t \mathrm{d}W_t \, t \geq 0 \tag{5.2}$$

Dabei ist W_t der Standard-Wiener-Prozess, wie in 5.2.1 definiert.

Ein Beispiel für einen möglichen Verlauf eines solchen Prozesses ist in Abbildung 5.2 gezeigt. Für die Simulation wurden eine erwartete Rendite $\mu = 0$ angenommen und eine Volatilität $\sigma = 0.25$, der Pfad wurde 500 Zeitschritte lang simuliert, der Startwert der Aktie betrug 50.

5.3 Handelsstrategien

Als Maß für die Qualität einer Anlagestrategie innerhalb eines Marktes wird die Marktrendite dieses Marktes verwendet. Man unterscheidet zwischen zwei Paradigmen: *Aktiven* und *passiven* Anlagestrategien.

5.3.1 Aktive Handelsstrategien

Aktive Strategien haben das Ziel, ein Portfolio zu erstellen, dessen Rendite die eines gewählten Index - des Benchmarks - übertrifft. Dabei wird davon ausgegangen, dass menschliche Expertise und Intuition in der Lage sind, Anlagen zu finden, die vom Markt falsch bewertet wurden, und so den Markt zu „schlagen“. Jede Abweichung vom Benchmark in der Portfolio-Zusammensetzung birgt natürlich das Risiko, eine falsche Anlageentscheidung zu treffen, und so weniger Rendite als die Marktrendite zu erzielen.

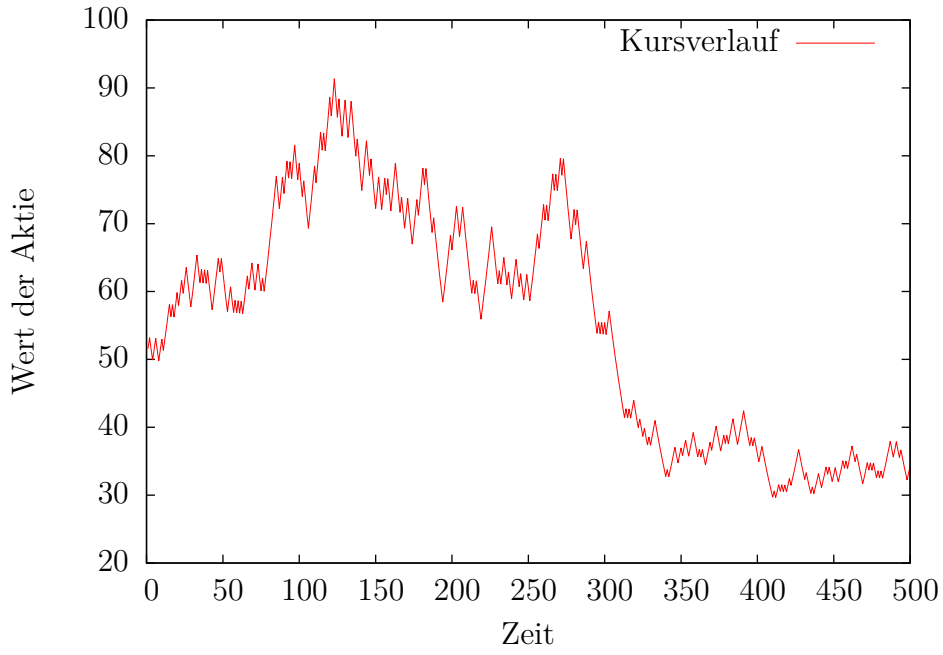

Abbildung 5.2: Beispiel für einen Aktienkursverlauf nach dem Black-Scholes-Modell mit Parametern $\mu = 0$, $\sigma = 0.25$

5.3.2 Passive Handelsstrategien

Das in dieser Arbeit behandelte Problem der Minimierung des Tracking Error zählt dagegen zu den passiven Anlage-Strategien. Eine solche Strategie geht davon aus, dass es ohne Vorteile außerhalb eines Marktes - etwa Insiderwissen - sehr unwahrscheinlich ist, eine Anlagestrategie zu entwickeln, die über einen langen Zeitraum stets bessere Gewinne einbringt als der Marktdurchschnitt. Weiterhin wird davon ausgegangen, dass die einzigen für eine Investitionsentscheidung relevanten Informationen der Wert und die Wert-Historie einer Anlage sind. Eine passive Anlagestrategie versucht nun, ein Portfolio zu erstellen, dessen Entwicklung der des betrachteten Marktes entspricht, und dessen Rendite damit ebenfalls der Marktrendite.

Unternehmen	Gewichtung im Index
Adidas	2,12
Allianz	7,20
BASF	9,86
Bayer	9,10
Beiersdorf	0,92
BMW	3,53
Commerzbank	0,95
Continental AG	1,05
Daimler	6,18
Deutsche Bank	4,57
Deutsche Börse	1,31
Deutsche Post	2,26
Deutsche Telekom	3,83
E.ON	4,03
Fresenius Medical Care	1,64
Fresenius SE	1,71
HeidelbergCement	0,97
Henkel	1,64
Infineon Technologies	1,01
K+S	0,92
Lanxess	0,85
Linde	3,70
Lufthansa	1,00
Merck	0,98
Munich Re	3,27
RWE	2,31
SAP	8,22
Siemens Elektrotechnik	10,11
ThyssenKrupp Stahl	1,04
Volkswagen	3,71

Tabelle 5.1: Zusammensetzung und Gewichtung das Aktienindex DAX am 01.04.2013

6 Index Tracking

In diesem Kapitel wird das in dieser Arbeit zentral behandelte Optimierungsproblem aus der Finanzmathematik, die kardinalitätsbeschränkte *Tracking Error Minimization*, vorgestellt. Ziel der Tracking Error Minimization ist das Erstellen eines Portfolios, das den Wertverlauf eines durch einen Index gegebenen Marktes möglichst genau abbildet, dabei jedoch aus möglichst wenigen verschiedenen Aktien besteht. Während die betrachteten Aktien im mathematischen Modell wie in Kapitel 5 durch Zufallsvariablen beschrieben werden, liegt dem Index Tracking letztlich auch eine weitere Fragestellung zugrunde: In welchem langfristigen mathematischen Zusammenhang stehen die Kursentwicklungen verschiedener Aktien *zueinander*? Dabei wird angenommen, dass dieses Verhältnis - im Gegensatz zu den Aktienkursen selbst - zumindest teilweise nicht zufällig ist. So erwartet man bei zwei Aktien eines Index, die zum selben Industrie-Sektor gehören, eher einen ähnlichen Kursverlauf als bei Aktien aus verschiedenen Sektoren.

6.1 Definition: Tracking Error

Gegeben seien ein Portfolio und ein Aktienindex, der *Benchmark*.

Der *Tracking Error* ist definiert als die Standardabweichung der Differenzrendite zwischen diesem Portfolio und dem Index über einen gewissen Zeitraum. Der Tracking Error ist also ein Maß für die „Ähnlichkeit“ der Entwicklung eines Portfolios verglichen mit einem Aktienindex.

Berechnen lässt sich der Tracking Error über die Formel

$$\mathrm{TE}(\mathbf{x}) := \sqrt{(\mathbf{x}-\mathbf{w})^{\mathrm{T}}\,\mathbf{Q}(\mathbf{x}-\mathbf{w})} \tag{6.1}$$

mit

- $\mathbf{x} \in \mathbb{R}^n, x_i \geq 0$ Prozentsatz des Portfolios, der in Anlage i investiert ist
- $\mathbf{w} \in \mathbb{R}^n, w_i \geq 0$ Prozent-Gewicht der Anlage i im Aktienindex
- $\mathbf{Q} \in \mathbb{R}^{n\times n}$ Kovarianzmatrix der Renditen des Aktienindex (symmetrisch, positiv semidefinit)

$\mathrm{TE}(\mathbf{x})$ wird auch als *Active Risk* bezeichnet.

6.2 Problemstellung: Minimierung des Tracking Error

Dass es nicht genügt, ein beliebiges Portfolio aus Aktien des jeweiligen Index zu erstellen, um den Index genügend genau nachzubilden, wird in den Grafiken 6.1 und 6.2 verdeutlicht. Während der Kursverlauf der beiden Aktien in Abbildung 6.1 dem Verlauf des Index recht ähnlich ist, ist dies bei den Aktien in Abbildung 6.2 nicht mehr der Fall.

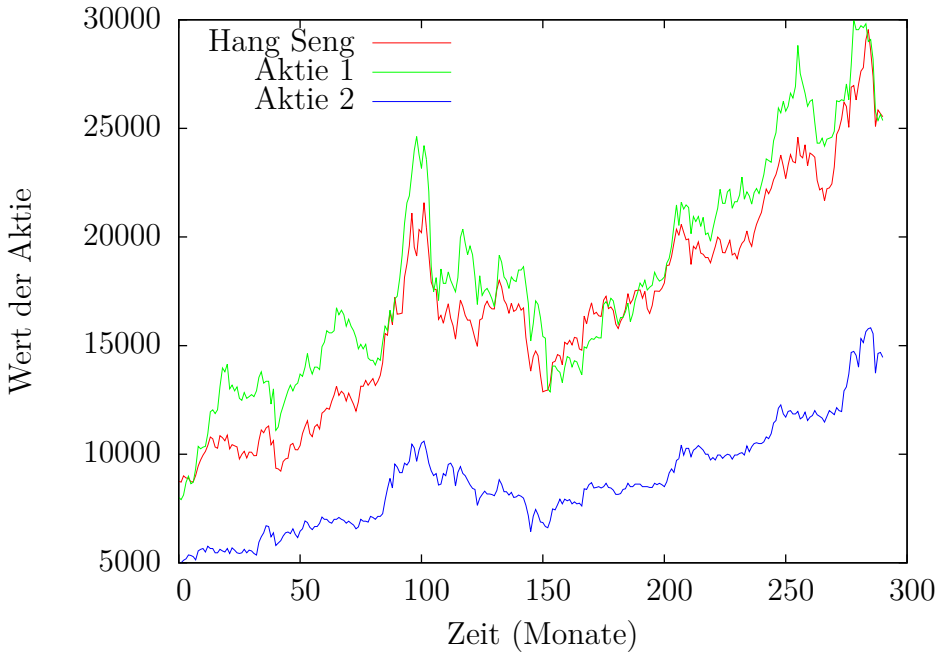

Abbildung 6.1: Verlauf des Hang Seng (rot) zusammen mit zwei darin enthaltenen Aktien (skaliert)

6.3 Kursverlauf und Industriesektoren

Die Abbildungen 6.3 und 6.4 zeigen beispielhaft, dass die Vermutung, dass Firmen aus demselben Marktsegment einen ähnlichen Wertverlauf ihrer Aktien haben, durchaus begründet ist: Die Aktienkurse der beiden Automobilhersteller BMW und Daimler zeigen einen ähnlichen Verlauf in Abbildung 6.3, während sich in Abbildung 6.4 der Kursverlauf der BMW-Aktie deutlich von dem des Bekleidungsherstellers Adidas und dem Dialyse-Dienstleister Fresenius Medical Care unterscheidet. Bei beiden Abbildungen reicht der betrachtete Zeitraum wieder vom 01.04.2012 bis zum 01.04.2013.

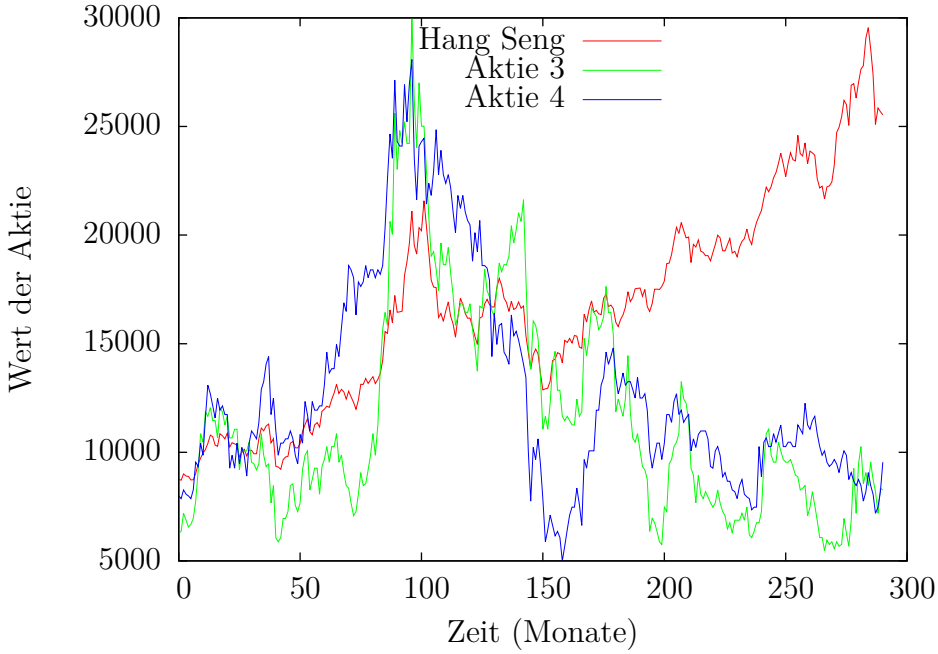

Abbildung 6.2: Verlauf des Hang Seng (rot) zusammen mit zwei darin enthaltenen Aktien (skaliert)

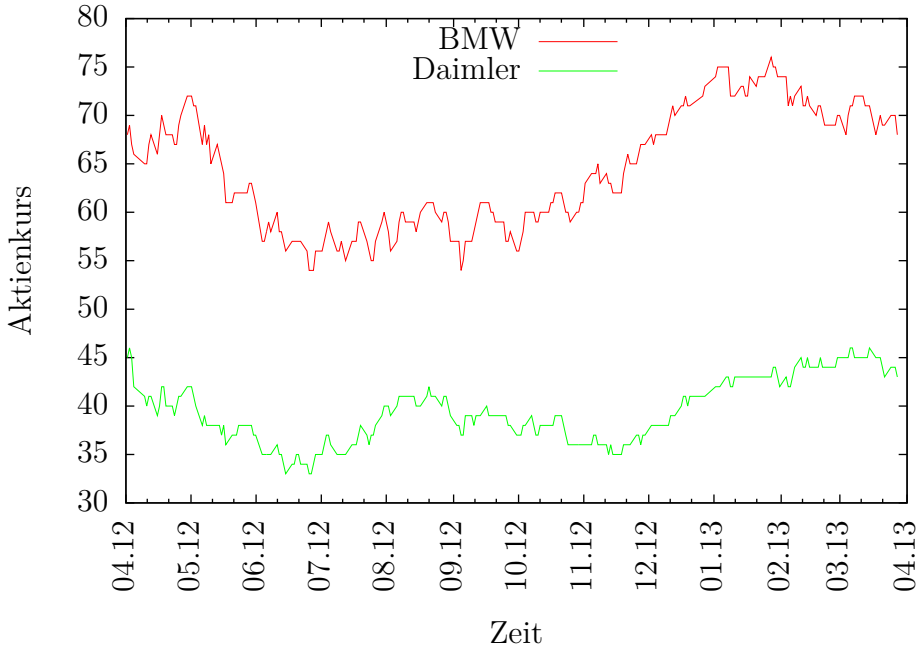

Abbildung 6.3: Verlauf der Aktienkurse von BMW und Daimler vom 01.04.2012 bis zum 01.04.2013

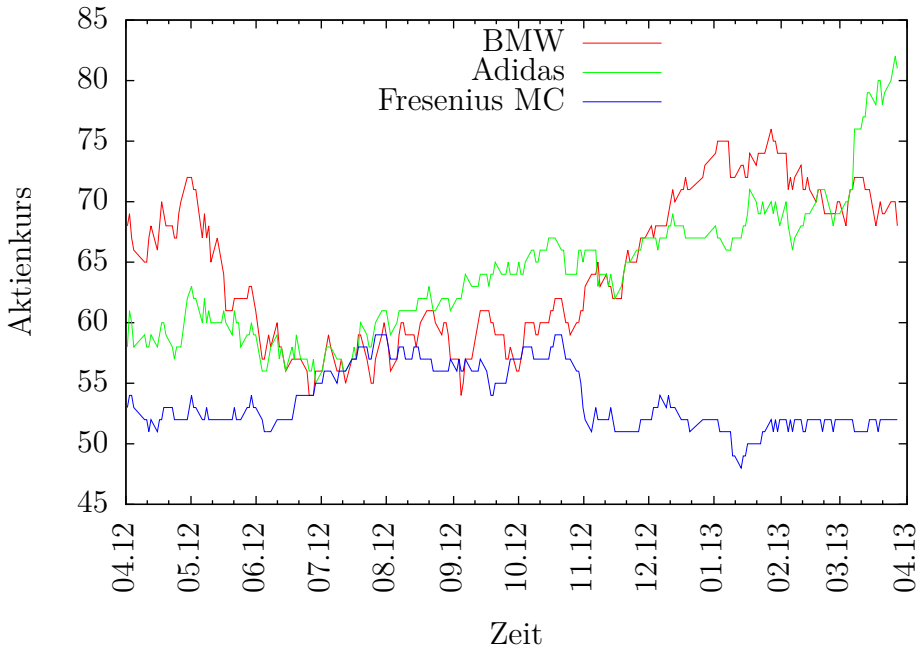

Abbildung 6.4: Verlauf der Aktienkurse von Adidas, BMW und Fresenius Medical Care vom 01.04.2012 bis zum 01.04.2013

6.4 Formulierung des Problems

Gegeben sei ein Aktienindex als Benchmark, der aus n Anlagen besteht, und eine Zahl K. Aufgabe ist nun, ein Portfolio zu erstellen, das aus nicht mehr als K verschiedenen Aktien besteht, sodass der *Tracking Error* (6.1) minimal ist. Aus Gründen der Übersichtlichkeit, und da es für ein Minimierungsproblem keine Rolle spielt, wird in den folgenden Formulierungen das Wurzelzeichen in der Berechnungsformel für den Tracking Error weggelassen.

Die Beschränkung des Portfolios auf K verschiedene Anlagen soll dazu dienen, die Kosten des Trackings gering zu halten.

Das kardinalitätsbeschränkte *Tracking-Error-Minimization*-Problem lautet also:

$$
\begin{aligned}
&\min_{\mathbf{x}\in\mathbb{R}^n} \mathrm{TE}(\mathbf{x}) && && \\
&\text{s.t.} && && \\
&\sum_{i=1}^{n} \Lambda(x_i) && \leq K \text{ mit} && \Lambda(x_i) = 1 \text{ für } x_i \neq 0 \\
& && && \Lambda(x_i) = 0 \text{ sonst} \\
&\sum_{i=1}^{n} x_i && = 1 && \\
&\mathbf{x} && \geq 0 &&
\end{aligned}
\tag{6.2}
$$

Dabei geben die Komponenten x_i des Vektors $\mathbf{x}$ den prozentualen Anteil der Aktie i im Portfolio an.

Das Optimierungsproblem (6.2) ist nicht stetig, da $\Lambda(x)$ nicht stetig ist. In [8] wird außerdem gezeigt, dass es für kardinalitätsbeschränkte Portfolios NP-schwer ist.

6.5 Kovarianzmatrix und Stichproben-Kovarianzmatrix

In der Definition des Tracking Error taucht die *Kovarianzmatrix* der Renditen der im Index enthaltenen Aktien auf.

Sei zunächst $\Omega = (\Omega, \Sigma, \mathcal{P})$ ein beliebiger Wahrscheinlichkeitsraum. Dann ist die Kovarianz zweier eindimensionaler, reellwertiger Zufallsvariablen $x, y : \Omega \to \mathbb{R}$ definiert als der Erwartungswert

$$\operatorname{cov}(x,\ y) = \mathbb{E}\left((x - \mathbb{E}(x)) \cdot (y - \mathbb{E}(y))\right).$$

Allgemein ist für eine d-dimensionale, reellwertige Zufallsvariable $\mathbf{x} : \Omega \to \mathbb{R}^d$, die *Kovarianzmatrix* von $\mathbf{x}$ gegeben durch

$$\mathbf{COV}(\mathbf{x}) = \mathbb{E}\left((\mathbf{x} - \mathbb{E}(\mathbf{x})) \cdot (\mathbf{x} - \mathbb{E}(\mathbf{x}))^{\mathrm{T}}\right).$$

Für die praktische Anwendung setzen beide Definitionen jedoch Kenntnis der Erwartungswerte der Zufallsvariablen „Renditen der Aktienkurse“ voraus. Diese sind jedoch *nicht* bekannt - lediglich deren historische Werte. Mit Hilfe dieser sogenannten Historie lässt sich die *Stichproben-Kovarianzmatrix* $\mathbf{Q}$ berechnen, die eine Approximation der Kovarianzmatrix der Renditen darstellt.

Die Einträge der Stichproben-Kovarianzmatrix berechnen sich wie folgt: Seien n Stichproben $\mathbf{x}_i$ von $\mathbf{x}$ gegeben. Sei

$$\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i$$

das arithmetische Mittel dieser Stichproben.

Dann ist $\mathbf{Q}$ definiert durch

$$\mathbf{Q} = \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}}) \cdot (\mathbf{x}_i - \bar{\mathbf{x}})^{\mathrm{T}}.$$

Wie bereits erwähnt handelt es sich hier nur um eine Approximation der tatsächlichen Kovarianzmatrix; Techniken, mit denen sich bestimmen lässt, inwieweit die Stichproben-Kovarianzmatrix der tatsächlichen Kovarianzmatrix entspricht und wie die zugrundeliegende Struktur der betrachteten Zufallsvariablen sich besser abbilden lässt, etwa Shrinking und PCA, werden in [3] und [19] vorgestellt.

Die Stichproben-Kovarianzmatrix für die Renditen der Aktienkurse lässt sich bei bekannter Historie sofort berechnen, damit auch die Zielfunktion des Tracking-Error-Minimierungsproblems.

6.6 GNC-Ansatz

Das eigentliche Hindernis bei der Minimierung des Tracking Error ist nicht die Zielfunktion TE($\mathbf{x}$) selbst, sondern es sind wie bereits erwähnt die Nebenbedingungen, die das Problem schwierig machen. Diese sind zunächst ganzzahlig, daher lassen sich generische Löser für stetig differenzierbare und/oder konvexe Probleme nicht sofort anwenden. Der bei [8] beschriebene Ansatz macht sich zunutze, dass durch Relaxieren gewisser Nebenbedingungen das betrachtete Optimierungsproblem (6.2) stark vereinfacht wird.

Ohne Kardinalitätsbeschränkung ist das Optimierungsproblem (6.2) ein stetiges Optimierungsproblem:

$$\boxed{\begin{aligned} &\min_{\mathbf{x}\in\mathbb{R}^n} \mathrm{TE}(\mathbf{x}) \\ &s.t. \\ &\sum_{i=1}^{n} x_i = 1 \\ &\mathbf{x} \geq 0 \end{aligned}} \tag{6.3}$$

6.6.1 Approximation durch stetige Optimierungsprobleme

Man erstellt nun eine Folge $\mathcal{P}_k$ von Minimierungsproblemen, ausgehend von $\mathcal{P}_0$ wie in (6.3).

Die nicht stetige reellwertige Funktion $\Lambda(x)$ wird approximiert durch

$$h_\lambda(x) = \begin{cases} \lambda x^2 & \text{für } |x| \leq \sqrt{\frac{1}{\lambda}} \\ 1 & \text{sonst.} \end{cases}$$

h_λ ist stetig, aber nicht differenzierbar.

Der Verlauf der Funktion $h_\lambda(x)$ ist, jeweils für verschiedene Werte von λ, in Abbildung 6.5 aufgetragen. Man erkennt, dass mit größerem λ die Funktion $h_\lambda(x)$ also wie gewünscht ähnlich zur ursprünglichen Funktion $\Lambda(x)$ verläuft, während für niedrige Werte von λ gilt, dass $h_\lambda(x)$ annähernd konvex ist.

Mit der Hilfsfunktion $h_\lambda(x)$ kann nun (6.2) wie folgt umformuliert werden:

$$\boxed{\begin{aligned} &\min_{\mathbf{x}\in\mathbb{R}^n} \mathrm{TE}(\mathbf{x}) + \mu \sum_{i=1}^{n} h_\lambda(x_i) \\ &\text{s.t.} \\ &\sum_{i=1}^{n} x_i = 1 \\ &\mathbf{x} \geq 0 \end{aligned}} \tag{6.4}$$

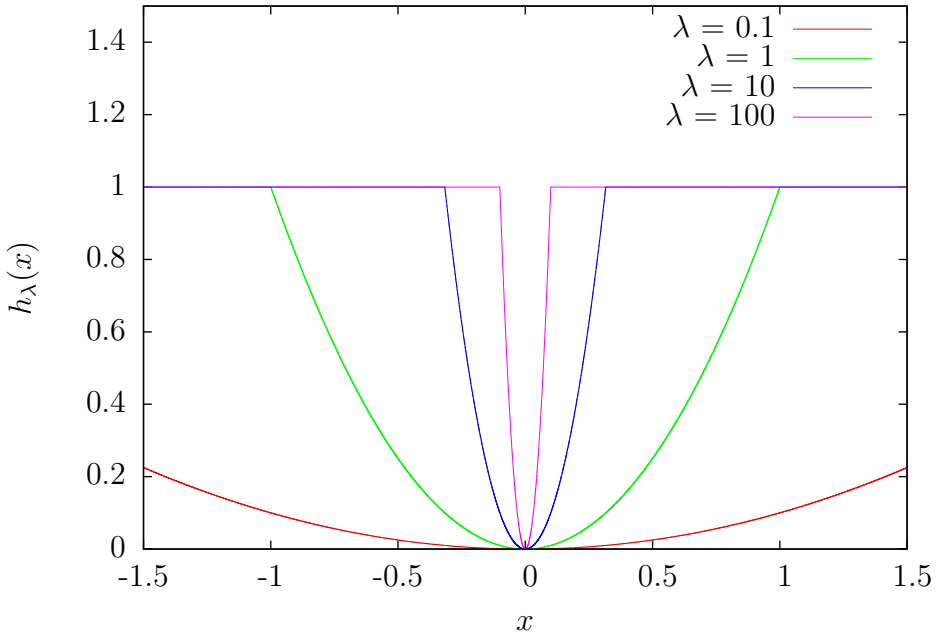

Abbildung 6.5: Die Funktion $h_\lambda(x)$ für verschiedene Werte von λ

6.6.2 Approximation durch stetig differenzierbare Optimierungsprobleme

Da wie erwähnt und auch an den Abbildungen zu erkennen die Funktion $h_\lambda(x)$ nicht differenzierbar ist, jedoch eine stetig differenzierbare Funktion gewünscht wird, um verschiedene globale Optimierungsverfahren anwenden, etwa das SQP-Verfahren, zu können, wird eine weitere, diesmal stetig differenzierbare Hilfsfunktion $g_\lambda(x)$ definiert:

$$g_\lambda(x,\rho) = \begin{cases} \lambda x^2 & \text{für } |x| \leq q \\ 1 - \frac{\rho}{2}(|x| - r)^2 & \text{für } q \leq |x| \leq r \\ 1 & \text{sonst} \end{cases}$$

mit

$$r = \sqrt{\frac{2}{\rho} + \frac{1}{\lambda}}$$
$$q = \frac{1}{\lambda r} \; .$$

In Abbildung 6.6 ist für den eindimensionalen Fall der Verlauf der Hilfsfunktion $g_\lambda(x,\rho)$ für den festen Wert $\lambda = 100$ und verschiedene Werte von ρ dargestellt. Man erkennt, dass für kleinere Werte von ρ die Funktion $g_\lambda(x,\rho)$ in einem größer werdenden Gebiet um den

Ursprung stetig differenzierbar und konvex ist. Für größer werdendes ρ nähert sich der Verlauf von $g_\lambda(x, \rho)$ dagegen immer mehr der ursprünglichen Nebenbedingungsfunktion $\Lambda(x)$ an, genau wie gewünscht.

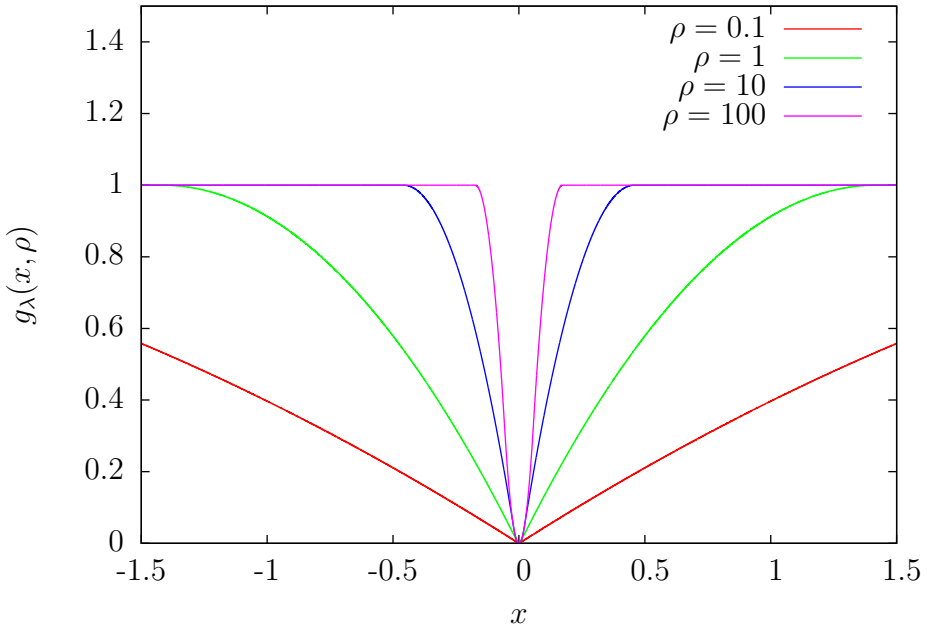

Abbildung 6.6: Die Funktionen $h_\lambda(x)$ und $g_\lambda(x, \rho)$für $\lambda = 100$ und $\rho = 100$

Es ergibt sich als Folge für den GNC-Prozess

$$\begin{aligned} &\min_{\mathbf{x}\in\mathbb{R}^n} \mathbf{gnc}_k(x) = \mathrm{TE}(\mathbf{x}) + \mu \sum_{i=1}^{n} g_\lambda(x_i, \rho_k) \\ &\text{s.t.} \\ &\sum_{i=1}^{n} x_i = 1 \\ &\mathbf{x} \geq 0 \ . \end{aligned} \tag{6.5}$$

Die GNC-Folge enthält in der Zielfunktion also jeweils einen „Straffaktor“, der umso größer ist, je mehr der x_i ungleich 0 sind.

6.7 GNC-Ansatz für eine vorgegebene Maximalzahl von Aktien im Portfolio

Ein Portfolio, in welchem nur maximal K Aktien enthalten sind, während der Tracking Error möglichst gering ist, lässt sich mit (6.5) bereits berechnen, indem so lange Werte für

den Parameter μ ausprobiert werden, bis das berechnete Portfolio aus der gewünschten Anzahl Aktien besteht.

Besser ist natürlich, die Zielfunktion $\mathbf{gnc}_k(\mathbf{x})$ so anzupassen, dass die Schranke K dort bereits auftritt. Es ergibt sich folgende Formulierung der GNC-Folge:

$$\begin{aligned} &\min_{\mathbf{x}\in\mathbb{R}^n} \mathbf{gnc(K)}_k(\mathbf{x}) = \mathrm{TE}(\mathbf{x}) + \mu \cdot \max\left(\sum_{i=1}^{n} g_\lambda(x_i, \rho_k) - K, 0\right) \\ &\text{s.t.} \\ &\sum_{i=1}^{n} x_i = 1 \\ &\mathbf{x} \geq 0 \end{aligned} \tag{6.6}$$

Die Zielfunktion wurde also wie folgt angepasst: Lösungen, für die der Wert der Summe $\sum_{i=1}^{n} g_\lambda(x_i, \rho_k)$, der für große Werte von ρ ja annähernd dem Wert von $\sum_{i=1}^{n} \Lambda(x)$ entspricht und damit annähernd der Zahl der nicht-Null-Variablen der aktuellen Lösung, größer ist als der vorgegebene Parameter K, werden mit einem „Straffaktor“ μ versehen. Im implementierten Algorithmus ist $\mu = 100$ gesetzt.

6.8 Weitere Ansätze zur Minimierung des Tracking Error

Für das *Tracking-Error-Minimization*-Problem finden sich in der Literatur eine Vielzahl verschiedener Ansätze, wobei heuristische Ansätze dominieren. In [8] wird der oben vorgestellte GNC-Ansatz verglichen mit einem Hybrid-Populations-Heuristik-Ansatz [2], der von zufälligen Start-Portfolios ausgehend versucht, diese zu einem optimalen Tracking-Portfolio zusammenzusetzen, und einem Ansatz, der ebenso wie der in diseser Arbeit vorgestellte GNC-Ansatz versucht, die unstetigen Stellen der Nebenbedingungen stetig zu approximieren [17].

Ein weiterer Ansatz, der Mixed Integer Programming verwendet, wird in [7] vorgestellt. Die dortige Formulierung umfasst Transaktionskosten, belässt aber die Entscheidungsvariablen, ob eine Aktie im Portfolio enthalten ist oder nicht, als binäre, ganzzahlige Variablen. Anders als im GNC-Algorithmus lässt sich die Anzahl der für das Portfolio gewählten Aktien nicht direkt beschränken, der Mixed-Integer-Algorithmus liefert jedoch für die auch in Kapitel 8 verwendeten Daten Portfolios mit einer geringen Zahl ausgewählter Aktien und geringem Tracking Error.

Im Folgenden werden die wichtigsten existierenden Ansätze kurz vorgestellt.

6.8.1 Genetic Programming mit Populationsheuristik

Genetic-Programming-Ansätze nutzen stochastische Optimierungs- und Suchverfahren, die sich an die Grundprinzipien der biologischen Evolution anlehnen.

Zunächst wird eine zufällige Population möglicher Lösungen des Optimierungsproblems erzeugt. Diese werden anhand einer sogenannten Fitnessfunktion bewertet, die das zu lösende Optimierungsproblem beschreibt. Die Lösungen, welche die besten Fitness-Werte aufweisen, werden zufällig verändert (durch die Operationen Mutation und ggf. Rekombination), während die restlichen verworfen werden (Selektion). Dieser Zyklus aus Bewertung, Mutation/Rekombination und Selektion wird iteriert, bis ein Abbruchkriterium erreicht wird.

Erstellen der Population ist denkbar leicht: Es wird zufällig eine Menge der möglichen Portfolios, die die gewünschte Anzahl Aktien enthalten, ausgewählt. Die Gewichtung der Aktien im Portfolio geschieht ebenfalls zufällig. Kreuzen zweier solcher Portfolios geschieht dann unter der Bedingung, dass die Zahl der Aktien im Portfolio gleich bleiben muss. Dabei unterscheidet man zwischen Ein-Punkt-Kreuzungen, wie in Tabelle 6.1, bei dem Portfolio 1 an einer zufälligen Stelle abgetrennt und mit Portfolio 2 ergänzt wird, und Zwei-Punkt-Kreuzungen, bei denen es zwei solcher Trennpunkte gibt, wie in Tabelle 6.2 dargestellt.

Eltern-Portfolio 1	**1110**100100
Eltern-Portfolio 2	0100**101011**
Trennpunkt	5
Kind-Portfolio 1	1110101011
Kind-Portfolio 2	0100100100

Tabelle 6.1: Ein-Punkt-Kreuzung

Eltern-Portfolio 1	**1000011011**
Eltern-Portfolio 2	0110**110**010
Trennpunkte	3,7
Kind-Portfolio 1	1000110011
Kind-Portfolio 2	0111001010

Tabelle 6.2: Zwei-Punkt-Kreuzung

6.8.2 Hybrid-Ansatz mit Quadratic Programming

In [2] wird der generische Genetic-Programming-Ansatz hybridisiert, indem als Fitness-Funktion zur Bewertung einzelner Mitglieder der Population jeweils das quadratische

Programm

$$\boxed{\begin{aligned} &\min \sqrt{\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x}} && (6.7)\\ &\text{s. t.} \\ &1 \cdot \mathbf{x} = 0 \\ &-\mathbf{w} \leq \mathbf{x} \leq -\mathbf{w} + 1 && (6.8) \end{aligned}}$$

gelöst wird. Zu beachten ist, dass im Gegensatz zur Definition des Tracking Error in 6.1 $\mathbf{x}$ hier für die Differenz des Portfolios zum Benchmark-Portfolio $\mathbf{w}$ steht.

6.8.3 Simulated Annealing und Threshold-Accepting

Wie das Genetic Programming ist auch Simulated Annealing ein Ansatz, der heursitisch nach einem Minimum sucht. Hierbei bedient sich der Ansatz, der auf Modellen der statistischen Thermodynamik basiert (siehe [18]), einer Analogie aus dem Hüttenwesen: Ein Energie-Minimum soll gefunden werden durch langsames *Kühlen* (Annealing) einer Substanz. Der Kühlvorgang findet dabei „langsam“ statt, um möglichst das globale und nicht ein lokales Minimum zu erreichen. Dieser Ansatz wird in [12] auf das Index-Tracking-Problem angewandt.

Das sogenannte Threshold-Accepting-Verfahren lässt sich als deterministisches Gegenstück zum Simulated Annealing betrachten. Es wird versucht, ein „Steckenbleiben“ in einem lokalen Minimum zu vermeiden, indem alle Lösungen betrachtet werden, die nicht schlechter als ein gegebener Schwellenwert sind. Für das Index-Tracking-Problem wurde dieser Ansatz in [13] implementiert.

6.8.4 Weitere Ansätze

In [32] wird ein Ansatz vorgestellt, der auf der Simulation von Partikel-Schwärmen basiert. Ein weiterer Ansatz, in [10] vorgestellt, versucht sogenannte Faktoren zu analysieren, die den Index beeinflussen. Neben der Marktrendite werden hier auch weitere Aspekte wie Liquidität der einzelnen Marktsektoren und Volatilität der (Modelle der) einzelnen Aktien betrachtet.

6.9 Der GNC-Algorithmus

Mit der in 6.6 definierten GNC-Folge ist damit bereits ein iteratives Verfahren gegeben: Ausgehend von einem sehr kleinen ρ-Wert, der zur Folge hat, dass das Optimierungsproblem 6.6 konvex ist, wird ρ schrittweise erhöht und das entstehende Optimierungsproblem erneut gelöst.

Der Iterationsprozess terminiert, wenn für alle Variablen x_i jeweils genau eine der folgenden Bedingungen erfüllt ist: Entweder gilt $x_i < q$, und damit ist die Aktie *nicht* im

Tracking-Portfolio enthalten, oder es gilt $x_i > r$, und die Aktie ist Teil des Tracking-Portfolios.

Der Pseudocode des Verfahrens ist in Algorithmus 6 abgebildet.

Algorithmus 6 : GNC-Algorithmus zur Minimierung des Tracking Error

Input : Konstanten λ μ, K, monoton steigende Folge $\{\rho_k\}$ mit $\lim_{k\to\infty} \rho_k = \infty$.

Output : Vektor x_k mit $\Lambda(x_k) < K$ und $\mathbf{gnc}_k(x_k)$ lokal minimal.

$k := 1$

$$x_k := \min_{x\in\mathbb{R}^n} TE(x)$$

$$s.t.$$

$$\sum_{i=1}^{n} x_i = 1$$

$$x \geq 0$$

while $!\,(\ (x_k)_i \leq q_k \,||\, (x_k)_i \geq r_k\ )\ \forall i$ **do**

$k := k + 1$

$$x_k := \min_{x\in\mathbb{R}} \mathbf{gnc}_k(x) = TE(x) + \mu \cdot \max\left(\sum_{i=1}^{n} g_\lambda(x_i, \rho_k) - K, 0\right)$$

$$s.t.$$

$$\sum_{i=1}^{n} x_i = 1$$

$$x \geq 0$$

end

7 Index Tracking per Subspace-Correction

In diesem Kapitel werden die iterativen Ansätze der Subspace Correction aus Kapitel 3 auf das Tracking-Error-Minimization-Problem angewandt. In Kapitel 4 wurde bereits gezeigt, wie Subspace Correction zum Lösen von Optimierungsproblemen angewandt werden kann. Im Folgenden werden die problemspezifisch notwendigen Modifikationen vorgestellt, nachdem noch einmal beschrieben wird, warum das aus der Stochastik bekannte Standardverfahren der Principal Component Analysis das Problem nicht löst.

7.1 Principal Component Analysis

Üblicherweise werden Kovarianzmatrizen, wie sie in der Formel (6.1) für die Berechnung des Tracking-Error auftreten, mit *Principal Component Analysis* (PCA) behandelt, um die „entscheidenden“ Variablen zu bestimmen. Wendet man aber diese Technik auf die Stichproben-Kovarianzmatrizen der Renditen im Tracking-Error-Minimization-Problem an, so sieht man schnell, dass das *kardinalitätsbeschränkte* Optimierungsproblem damit keineswegs gelöst ist.

Zunächst erhält man, wenn man die Eigenwerte der Stichproben-Kovarianzmatrix betrachtet (wie in Abbildung 7.1 für den S&P500-Index gezeigt), den erwarteten starken Abfall der Beträge der Eigenwerte. Für das Beispiel des S&P500-Index ergibt sich jedoch, dass die erste der Principal Components für lediglich $22,64\%$ der Varianz der Zielfunktion verantwortlich ist. Die ersten fünf Eigenwerte sind nur für $37,07\%$ der Varianz verantwortlich, wie in Abbildung 7.2 erkennbar. Erfolgreiches Anwenden der PCA hofft auf Varianzen jenseits der 90%, die von den ersten beiden, oder maximal den ersten drei der Principal Components hervorgerufen werden.

Doch selbst wenn dieser Zustand gegeben wäre und ein Großteil der Varianz der Zielfunktion durch wenige Eigenwerte verursacht würde, wäre damit das kardinalitätsbeschränkte Tracking-Error-Minimization-Problem nicht gelöst: Wie in Abbildung 7.3 zu sehen, sind in der von der Principal Component Analysis erstellten neuen Orthogonalbasis für die erste Principal Component im Basisvektor nur *fünf* Einträge nahe 0. Ein Tracking-Portfolio basierend auf dieser Principal Component würde also 452 Aktien enthalten.

Für ein Tracking-Portfolio, das aus wenigen Aktien bestehen soll, ist die Principal Component Analysis als einziges Auswahlkriterium daher ungeeignet.

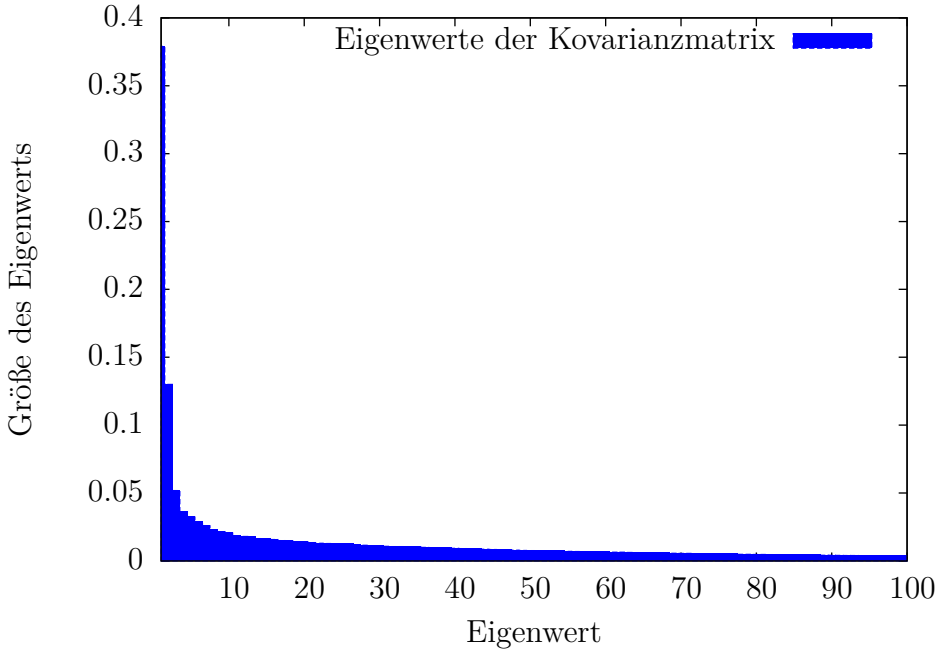

Abbildung 7.1: Die 100 größten Eigenwerte aus der Principal Component Analysis der Stichproben-Kovarianzmatrix des S&P500-Index

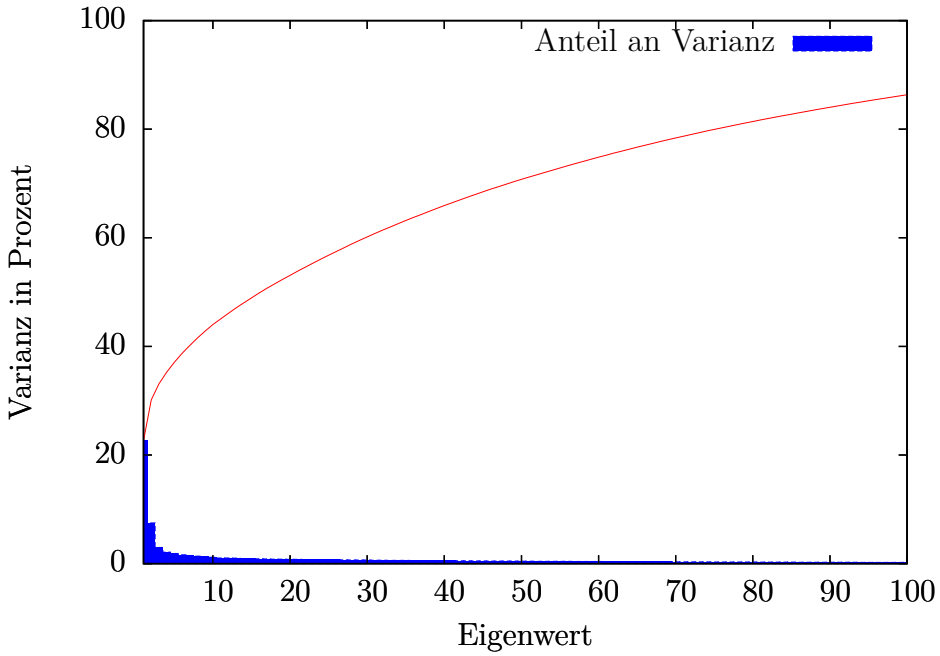

Abbildung 7.2: Anteile der Varianz je Eigenwert und Summation über diese für die Zielfunktion des S&P500-Index, nur die 100 größten Eigenwerte

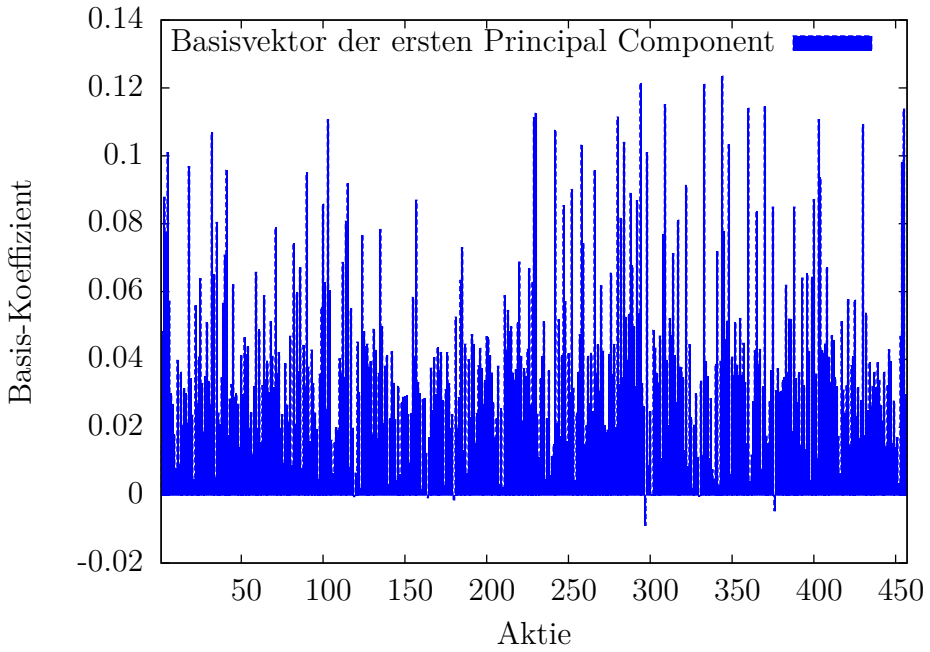

Abbildung 7.3: Basisvektor der ersten Principal Component für die PCA des S&P500-Index

7.2 Zerlegung in Teilräume

Der Raum, auf dem das Tracking-Error-Minimization-Problem definiert ist, ist für einen Index aus n Aktien der $\mathcal{V} = \mathbb{R}^n$. Jede Dimension entspricht einer Aktie aus dem Index. Die orthogonalen Teilräume $\mathcal{V}_i$ für die beiden Subspace-Correction-Methoden sind nun gerade die $d = \frac{n}{M}$-dimensionalen Unterräume des $\mathbb{R}^n$, die durch die kanonischen Einbettungen

$$\begin{aligned} \mathcal{V}_i &= \mathbb{R}^d \\ \mathbf{Q}_i : \mathcal{V} &\hookrightarrow \mathbb{R}^n \\ \mathbf{Q}_i : \mathbf{v}_i \in \mathcal{V}_i &\mapsto \left(\underbrace{0, \cdots, 0}_{(i-1)\cdot d}, \mathbf{v}_i, \underbrace{0, \cdots, 0}_{n-i\cdot d} \right)^{\mathrm{T}} \end{aligned}$$

gegeben sind. Der Vektor aus dem Teilraum wird also passend mit Nullen aufgefüllt.

7.3 Anpassung der Zielfunktion auf die Teilräume

Soll das Tracking-Portfolio aus K Aktien bestehen, so werden jedem Teilraum davon $\left\lfloor \frac{K}{M} \right\rfloor$ Aktien zugeordnet. Die Teilräume sind gleich gewichtet. Würde diese Gewichtung

beibehalten werden, ist anzunehmen, dass man das erhaltene Ergebnis durch Verändern der Gewichtung der Teilräume verbessern kann. Daher wird in Abschnitt 7.6 beschrieben, wie die Teilräume nach der Subspace-Correction neu gewichtet werden, um das Ergebnis zu verbessern.

In jeder Iteration und in jedem Teilraum wird das auf diesen Teilraum reduzierte Problem gelöst: Nur Aktien aus dem aktuellen Teilraum dürfen zum Tracking-Portfolio dieses Teilraums gehören. Dabei wird der GNC-Algorithmus aus Abschnitt 6.9 verwendet.

Pro Iteration ändert sich jedoch der Gewichtsvektor in der Zielfunktion: Nur in der ersten Iteration wird das Tracking-Error-Minimierungsproblem in der üblichen Formulierung (siehe (6.1)) gelöst, ab der zweiten Iteration ist die Lösung der vorigen Iterationen als Residuum im Gewichtsvektor enthalten: Für Iteration k lautet die Zielfunktion also mit den neuen Gewichten $\mathbf{w}^k$

$$\begin{aligned} \mathbf{w}^k &= \mathbf{w} - \sum_{j=0}^{k-1} \mathbf{x}^j \\ \mathrm{TE_k}(\mathbf{x}) &= \left(\mathbf{x} - \mathbf{w}^k\right)^{\mathrm{T}} \mathbf{Q} \left(\mathbf{x} - \mathbf{w}^k\right) \ , \end{aligned}$$

wobei die $\mathbf{x}^j$ jeweils die Lösungen aus dem j-ten Iterationsschritt sind.

7.4 Parallele Subspace-Correction

Der Algorithmus 7 beschreibt ein additives Verfahren zur Tracking-Error-Minimierung. Sei dazu $\mathbf{x}^0$ die gewählte Startlösung, $\mathbf{w}$ der Gewichtsvektor des Aktienindex und $\mathbf{Q}$ die Stichproben-Kovarianzmatrix des Index. Der Pseudocode für das additive Verfahren lautet dann wie in Algorithmus 7.

7.5 Sukzessive Subspace-Correction

Algorithmus 8 beschreibt ein multiplikatives Verfahren zur Tracking-Error-Minimierung, für eine gegebene oder geratene Startlösung $\mathbf{x}^0$, Gewichte $\mathbf{w}$ und Stichproben-Kovarianzmatrix $\mathbf{Q}$. Beim multiplikativen Verfahren fließt also in Schritt k die Lösung für Teilraum i bereits in das Residuum für Teilraum $i+1$ mit ein.

7.6 Zusammenfügen der Teilräume in der Tracking-Error-Minimierung

In diesem Abschnitt werden einige Verbesserungen der zuvor vorgestellten Subspace-Correction-Verfahren beschrieben.

Algorithmus 7 : Parallele Subspace-Correction

Iteriere:

1. Wenn für $k \geq 0$ bereits $\mathbf{x}^k \in \mathbb{R}^n$ bestimmt wurde, so finde parallel für alle $i = 1, \ldots, M$ Richtungen $\mathbf{e}_i^n \in V_i$, sodass gilt

$$\mathbf{x}^{k+1} = \text{argmin}(TE(\mathbf{x}^k + \mathbf{e}_i^k)) \text{ , also}$$
$$\text{TE}(\mathbf{x}^k + \mathbf{e}_i^k) \leq \text{TE}(\mathbf{x}^k + \mathbf{v}_i) \; \forall \mathbf{v}_i \in \mathcal{V}_i \text{ ,}$$

   der Tracking Error wird also durch eine GNC-Iteration auf jedem Teilraum minimiert.

2. Setze die neue Lösung $\mathbf{x}^{k+1}$ auf

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \sum_{i=1}^{M} \mathbf{e}_i^n \text{ .}$$

3. Setze $\mathbf{w} = \mathbf{w} - \mathbf{x}^{k+1}$.

Algorithmus 8 : Sukzessive Subspace-Correction

Iteriere:

1. Wenn für $k \geq 0$ bereits $\mathbf{x}^k \in \mathbb{R}^n$ bestimmt wurde, so finde sequentiell für $i = 1, \ldots, M$

$$\mathbf{x}^{n+i/m} = \mathbf{x}^{n+(i-1)/m} + \mathbf{e}_i^n \tag{7.1}$$

   mit $\mathbf{e}_i^n \in \mathcal{V}_i$, sodass gilt

$$TE(\mathbf{x}^{n+(i-1)/m} + \mathbf{e}_i^n) \leq \text{TE}(\mathbf{x}^{n+(i-1)/m} + \mathbf{v}_i) \; \forall \mathbf{v}_i \in \mathcal{V}_i \text{ .}$$

2. Setze $\mathbf{w} = \mathbf{w} - \mathbf{x}^{k+1}$.

7.6.1 Hybridverfahren mit Nachoptimierung

Bisher waren die einzelnen Teilräume gleich gewichtet. Als weiteren Schritt, um das Gesamtergebnis zu verbessern, wird nach der Subspace Correction eine Neugewichtung der einzelnen Aktien im Tracking-Portfolio durchgeführt. Analog zum Hybrid-Genetic-Programming-Ansatz (siehe Kapitel 6.8.2), werden die Teillösungen zusammengefasst, und dann das stark vereinfachte Optimierungsproblem

$$\boxed{\begin{aligned} &\min \sqrt{\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x}} \\ &\text{s.t.} \\ &\mathbf{1}^{\mathrm{T}} \cdot \mathbf{x} = 0 \\ &-\mathbf{w} \leq \mathbf{x} \leq -\mathbf{w} + \mathbf{1} \end{aligned}}$$

gelöst. Dabei ist $\mathbf{1}$ der Vektor, der aus Einsen besteht, und im Vektor $\mathbf{x}$ sind bereits die Gewichte $\mathbf{w}$ enthalten. Es werden also keine weiteren Aktien zum Tracking-Portfolio hinzugefügt, sondern lediglich die Gewichtung der einzelnen Aktien im Portfolio nachoptimiert. Dieses Optimierungsproblem ist ein Quadratisches Problem mit linearen Nebenbedingungen; es hat genau K Variablen - auch wenn die Zielfunktion in den Gesamtraum der Aktien eingebettet wird - und ist daher schnell lösbar.

7.6.2 Multi-Level-Verfahren, additiv

Eine Variante der klassischen Subspace-Correction-Methoden sind die sogenannten *Multi-Level-Verfahren*. In der additiven Multi-Level-Version des Subspace-Correction-Algorithmus wird zunächst wie gehabt eine natürliche Zahl $M < \frac{n}{2}$ gewählt, die Zahl der Teilräume, und der Aktienindex dann in M gleich große Teile geteilt. Die maximale Anzahl an verschiedenen Aktien im Tracking-Portfolio sei wieder durch K beschränkt. Dabei soll gelten, dass $K * M < n$.

Dann wird der GNC-Algorithmus auf die einzelnen Teilräume angewandt. Die Teil-Optimierungsprobleme werden unabhängig voneinander gelöst. Im Gegensatz zum bisherigen Subspace-Correction-Verfahren wird nun auch auf dem aktuellen Teilraum ein Portfolio aus K Aktien erstellt. Danach werden die Lösungen von jeweils zwei Teilproblemen im Produkt der beiden Teilräume zusammengefasst. Entsprechend werden dann die Lösungen der $\lceil \frac{M}{2} \rceil$ Teilräume auf die gleiche Weise in Paaren zusammengefasst, der Vorgang wird entsprechend weitergeführt, bis die Gesamtlösung berechnet wurde. Beim Zusammenfügen wird jeweils wieder der GNC-Algorithmus verwendet, um aus den $2K$ Aktien des zusammengefügten Tracking-Portfolios ein Portfolio aus K Aktien auszuwählen. Da also bis auf in der feinsten Ebene der Teilraumzerlegung nur Optimierungsprobleme mit jeweils $2K$ Variablen betrachtet werden, ist die Gesamtlaufzeit des Verfahrens geringer als die des globalen GNC-Verfahrens.

In Grafik 7.4 sind die Komponenten des Portfolio-Vektors $\mathbf{x}$, die an der Lösung des jeweiligen Teilproblems beteiligt sind, illustriert.

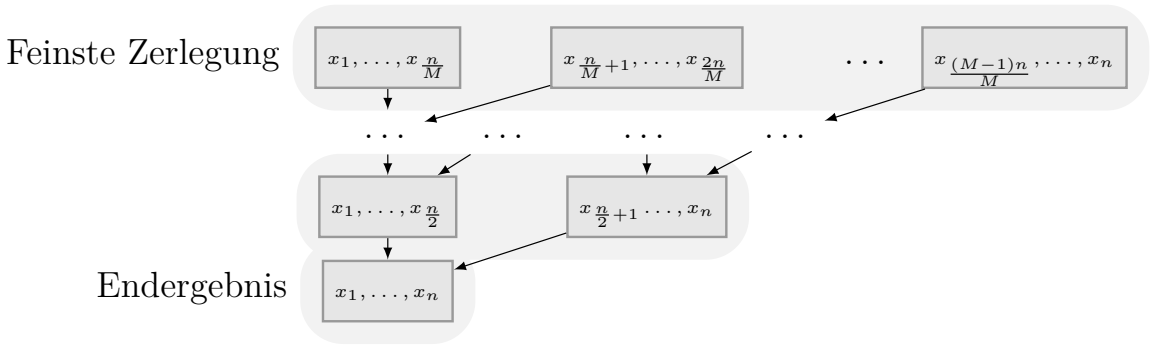

Abbildung 7.4: Zerlegung der Aktien im additiven Multi-Level Subspace-Correction-Algorithmus

7.6.3 Multi-Level-Verfahren, multiplikativ

Die multiplikative Variante des Algorithmus unterscheidet sich von der additiven wie folgt: Innerhalb einer Zerlegungsebene werden die Teilprobleme nicht parallel und separat gelöst, sondern wie folgt sequentiell: Die einzelnen Blöcke dürfen sich um eine natürliche Zahl O von Aktien überlappen. Zu jeder Zeit ist die Startlösung für den aktuellen Block die Lösung der vorigen Blöcke. Komponenten des Lösungsvektors, die aufgrund der Einbettung nicht zu den aktuell betrachteten Variablen gehören, gehen also dennoch in die Zielfunktion mit ein. Sind alle Blöcke abgearbeitet, werden die Teillösungen wie im additiven Verfahren zusammengefügt und die nächstgröbere Ebene berechnet. Das Zusammenfügen der Blöcke ist in Abbildung 7.5 gezeigt. Dabei sind Blöcke abgebildet, die sich jeweils um die Hälfte überlappen.

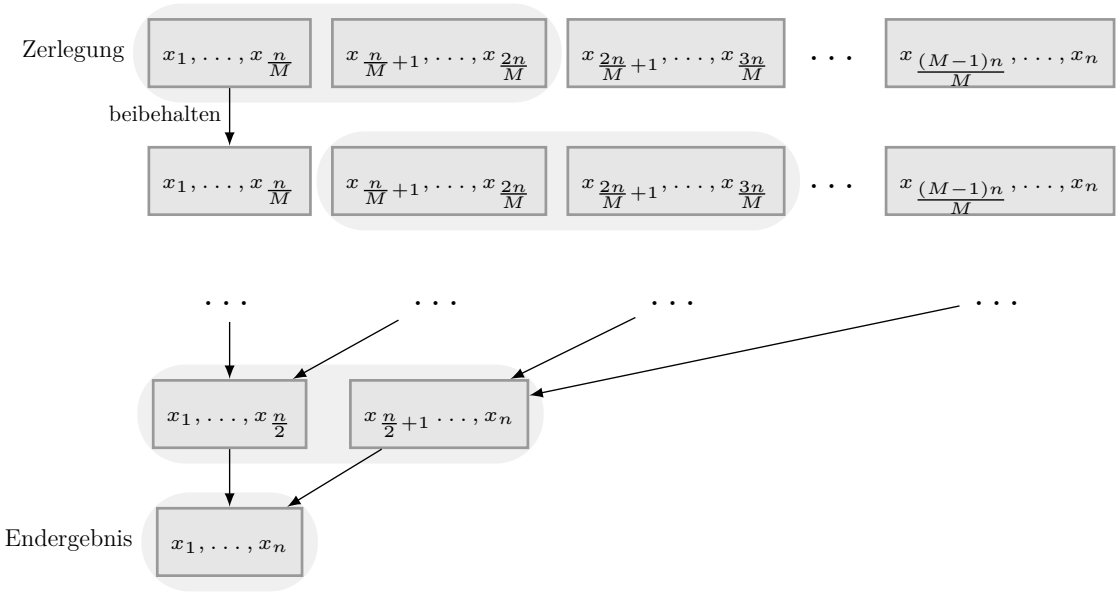

Abbildung 7.5: Zerlegung der Aktien im multiplikativen Hybrid-Subspace-Correction-Algorithmus

8 Testergebnisse

8.1 Vergleich der Algorithmen

Um die Güte des Subspace-Correction-Ansatzes zu überprüfen, wurden zunächst Testreihen mit den in [8] angegebenen Daten und Parametern durchgeführt.

Die Datensätze stammen aus der in [1] dokumentierten OR-Library und beinhalten wöchentliche Daten von März 1992 bis September 1997 für die folgenden Aktienindizes:

- Hang Seng, der wichtigste Index der Hongkonger Börse
- DAX100, dem heutigen HDAX, der die Aktien des DAX aus Tabelle 5.1 enthält und zusätzlich die 70 folgenden umsatzstärksten deutschen Aktien
- FTSE100, das Äquivalent des DAX100 der Londoner Börse, welches die 100 an der Londoner Börse gelisteten Unternehmen mit höchstem Börsenwert enthält
- S&P100, dem entsprechenden Index der 100 stärksten Unternehmen an der New Yorker Börse
- Nikkei225, dem wichtigsten asiatischen Index
- S&P500, einer Obermenge des S&P100, der die 500 Unternehmen der New Yorker Börse mit höchstem Börsenwert enthält
- Russel3000, der die 3000 Unternehmen der New Yorker Börse mit der höchsten Marktkapitalisierung enthält.

Alle Daten sind Split-bereinigt und Aktien, die in dem betrachteten Zeitraum aus dem Index ausgeschieden sind, wurden entfernt.

Die für die Algorithmen verwendeten Parameter wurden aus [8] übernommen, soweit angegeben:

- $\lambda = 10^8$
- $\mu = 100$
- $\rho_0 = 0.001$
- $K = 25$

Die monoton steigende Folge der ρ_k berechnet sich nach der Rechenvorschrift

$$\rho_{k+1} = 2.0 * \rho_k$$

Alle Algorithmen wurden in MATLAB implementiert, das verwendete Optimierungsverfahren für nichtlineare Probleme ist das SQP-Verfahren, wie in Abschnitt 2.2 beschrieben.

Die getesteten Verfahren sind

- Standard - das in Kapitel 6.9 beschriebene globale GNC-Verfahren
- Additiv - Paralleler Subspace-Correction-Algorithmus wie in Algorithmus 7 mit Nachoptimierung wie in Abschnitt 7.6.1 angegeben
- Multi-Level additiv - Paralleler Subspace-Correction-Algorithmus wie in Algorithmus 7 mit Zusammenfügen und Neuberechnung auf zusammengefügten Teilräumen wie in Abschnitt 7.6.2
- Multiplikativ - Sukzessiver Subspace-Correction-Algorithmus wie in Algorithmus 8 mit Nachoptimierung
- Multi-Level multiplikativ - Sukzessiver Subspace-Correction-Algorithmus wie in Algorithmus 7 mit Zusammenfügen und Neuberechnung auf zusammengefügten Teilräumen wie in Abschnitt 7.6.3

Für die Zahlenwerte des Tracking-Error wird die Berechnungsvorschrift 6.1 verwendet. Die Stichproben-Kovarianzmatrix wurde mit den Werten aus den ersten 260 Wochen der Datensätze berechnet. Der Tracking Error der Portfolios wurde dann für die übrigen 30 Wochen berechnet.

8.1.1 Hang Seng

Verfahren	Tracking Error ($\times 10^2$)
Standard	0.24
additiv	0.24
Multi-Level additiv	0.23
multiplikativ	0.23
Multi-Level multiplikativ	0.23

Tabelle 8.1: Ergebnisse Hang Seng

Der kleinste der Aktien-Indizes profitiert von der Teilraumzerlegung kaum. Dies ist nicht weiter verwunderlich, da der Index lediglich aus 31 Aktien besteht, das gewünschte Portfolio aber aus 25 Aktien bestehen soll.

8.1.2 Dax100

Verfahren	Tracking Error ($\times 10^2$)
Standard	0.58
additiv	0.52
Multi-Level additiv	0.41
multiplikativ	0.47
Multi-Level multiplikativ	0.38

Tabelle 8.2: Ergebnisse Dax100

Beim Dax100-Index lässt sich der Tracking Error im Vergleich zum GNC-Verfahren durch Nutzen eines Multiplikativen Subspace-Correction-Vefahrens um 34% verbessern.

8.1.3 FTSE100

Verfahren	Tracking Error ($\times 10^2$)
Standard	0.43
additiv	0.38
Multi-Level additiv	0.35
multiplikativ	0.45
Multi-Level multiplikativ	0.30

Tabelle 8.3: Ergebnisse FTSE100

Ähnlich wie beim Dax100-Index liefern die Subspace-Correction-Algorithmen in drei von vier Varianten ein besseres Ergebnis. Ein schlechterer Tracking Error entsteht nur beim multiplikativen Subspace-Correction-Verfahren.

8.1.4 S&P100

Verfahren	Tracking Error ($\times 10^2$)
Standard	0.50
additiv	0.43
Multi-Level additiv	0.45
multiplikativ	0.75
Multi-Level multiplikativ	0.44

Tabelle 8.4: Ergebnisse S&P100

Auch beim S&P100-Index liefern alle Varianten bis auf den multiplikativen Subspace-Correction-Algorithmus bessere Ergebnisse.

8.1.5 Nikkei225

Verfahren	Tracking Error ($\times 10^2$)
Standard	0.0058
additiv	0.0044
Multi-Level additiv	0.0038
multiplikativ	0.0040
Multi-Level multiplikativ	0.0037

Tabelle 8.5: Ergebnisse Nikkei225

Wieder liefern die Subspace-Correction-Verfahren einen um 34% besseren Tracking Error.

8.1.6 S&P500

Verfahren	Tracking Error ($\times 10^2$)
Standard	0.15
additiv	0.14
Multi-Level additiv	0.12
multiplikativ	0.13
Multi-Level multiplikativ	0.12

Tabelle 8.6: Ergebnisse S&P500

Beim ersten größeren Index, dem S&P500, der nach Bereinigung aller nicht durchgehend beteiligten Firmen aus 457 Aktien besteht, ist zum Einen der Tracking Error bei allen Verfahren sehr niedrig, zum Anderen sind die Subspace-Correction-Verfahren wieder um bis zu 20% besser.

8.1.7 Russel3000

Verfahren	Tracking Error ($\times 10^2$)
Standard	konvergiert nicht
additiv	0.93
Multi-Level additiv	0.85
multiplikativ	0.93
Multi-Level multiplikativ	0.83

Tabelle 8.7: Ergebnisse Russel3000

Beim Russel3000-Index, der bereinigt noch aus 2151 Aktien besteht, muss zunächst-festgestellt werden, dass der in [8] vorgestellte Algorithmus nicht zu einer gültigen Lösung

führt. Vergleicht man die beiden additiven Subspace-Correction-Varianten miteinander (Tabelle 8.7), sieht man das bereits bei den vorigen Indizes aufgetretene Verhalten: Die beiden Multilevel-Verfahren liefern bei längerer Laufzeit, bedingt durch das Lösen der zusätzlichen Optimierungsprobleme, ein etwas besseres Ergebnis.

8.2 Auswirkungen der Zahl an Aktien im Portfolio auf den Tracking Error

In diesem Abschnitt wird betrachtet, wieweit der Tracking Error eines Portfolios von der Zahl der in diesem Portfolio enthaltenen Aktien abhängt. In den meisten Fällen ergibt sich eine „Sättigung„ des Abfalls des Tracking Errors mit steigender Aktienzahl zwischen 10 und 25 Aktien. In allen Fällen wurde die komplette vorhandene Historie zur Approximation der Kovarianzmatrix bestimmt, die Historie dabei linear gewichtet. Die Teilräume für die Subspace-Correction-Methoden wurden dabei so gewählt, dass der kleinste Teilraum mindestens aus 15, höchstens aus $2K$ Aktien bestand.

8.2.1 Hang Seng

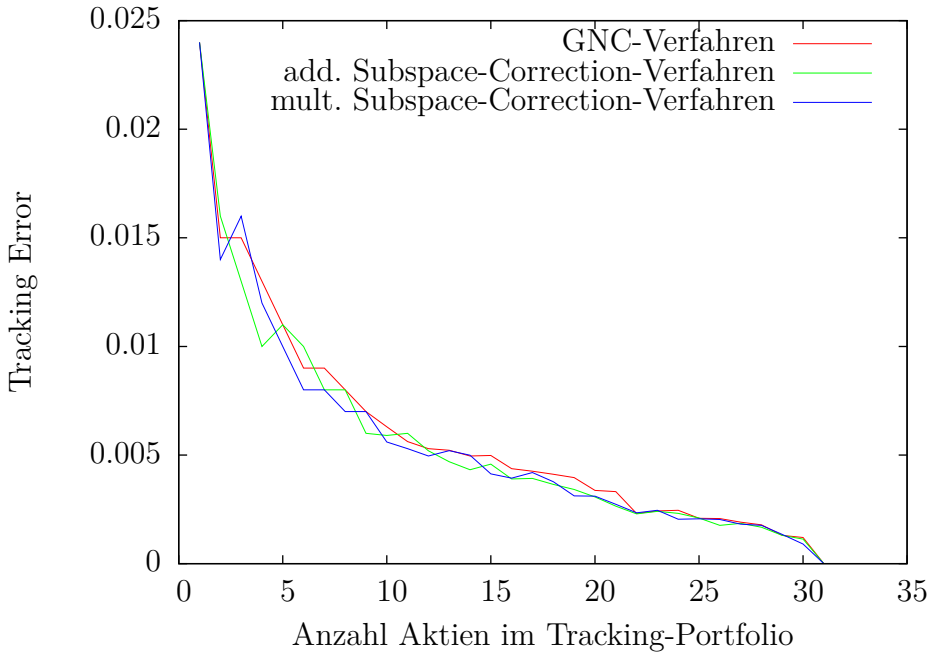

Abbildung 8.1: Tracking Error für Tracking-Portfolios des Hang Seng mit verschiedener Anzahl Aktien

Der Tacking Error in Abbildung 8.1 zeigt einen starken Abfall des Tracking Errors bei steigender Anzahl Aktien bis etwa $K = 12$, danach fällt der Tracking Error deutlich langsamer. Weder das additive noch das multiplikative Subspace-Correction-Verfahren ist dabei systematisch signifikant schlechter als das globale GNC-Verfahren.

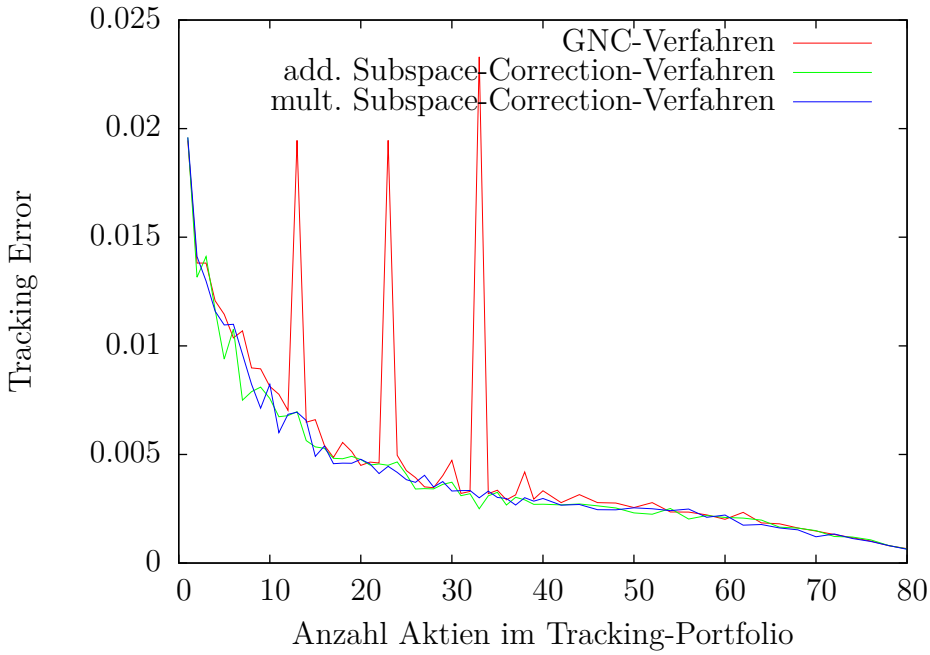

Abbildung 8.2: Tracking Error für Tracking-Portfolios des DAX100 mit verschiedener Anzahl Aktien

8.2.2 Dax100

Der Dax100 zeigt einen ähnlichen Abfall des Tracking Errors bei steigender Anzahl Aktien im Portfolio, unabhängig vom gewählten Algorithmus. Allerdings bleibt bei diesem Index das GNC-Verfahren vergleichsweise oft in einem lokalen Minimum „stecken“, erkennbar an den Ausschlägen des Graphen 8.2.

8.2.3 FTSE100

Beim FTSE100-Index in Abbildung 8.3 lässt sich der Tracking Error ab einer Aktienzahl von etwa 30 im Portfolio kaum noch verbessern. Wieder ist das GNC-Verfahren für einige Portfolios deutlich schlechter als die beiden Subspace-Correction-Verfahren.

8.2.4 S&P100

Für den S&P100-Index, zu sehen in Abbildung 8.4, zeigen die Tracking-Portfolios ein ähnliches Verhalten wie beim FTSE100-Index. Ein Portfolio aus 40 Aktien hat keinen signifikant besseren Tracking Error als eines aus 25 Aktien.

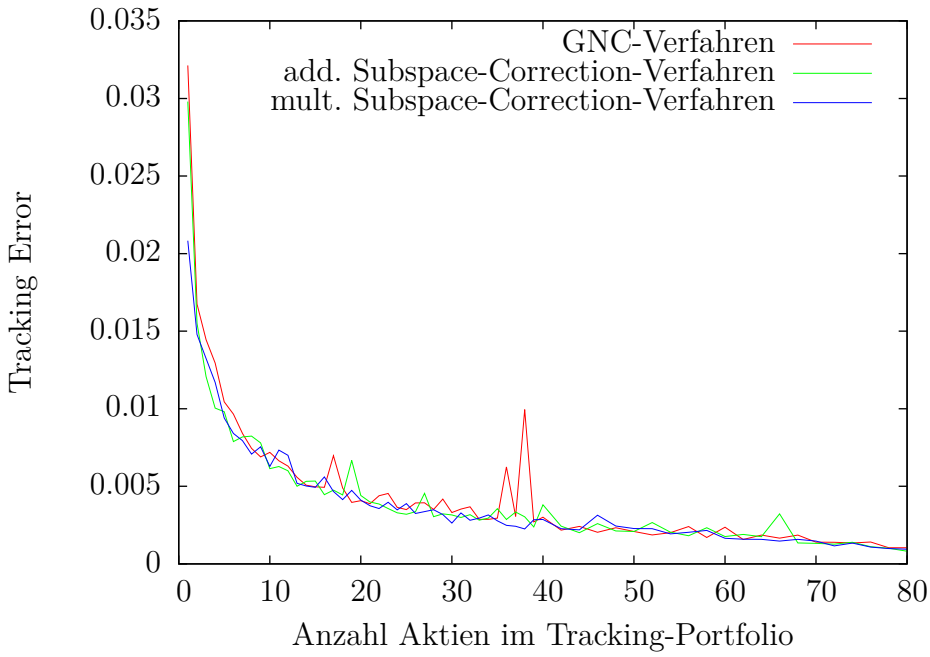

Abbildung 8.3: Tracking Error für Tracking-Portfolios des FTSE100 mit verschiedener Anzahl Aktien

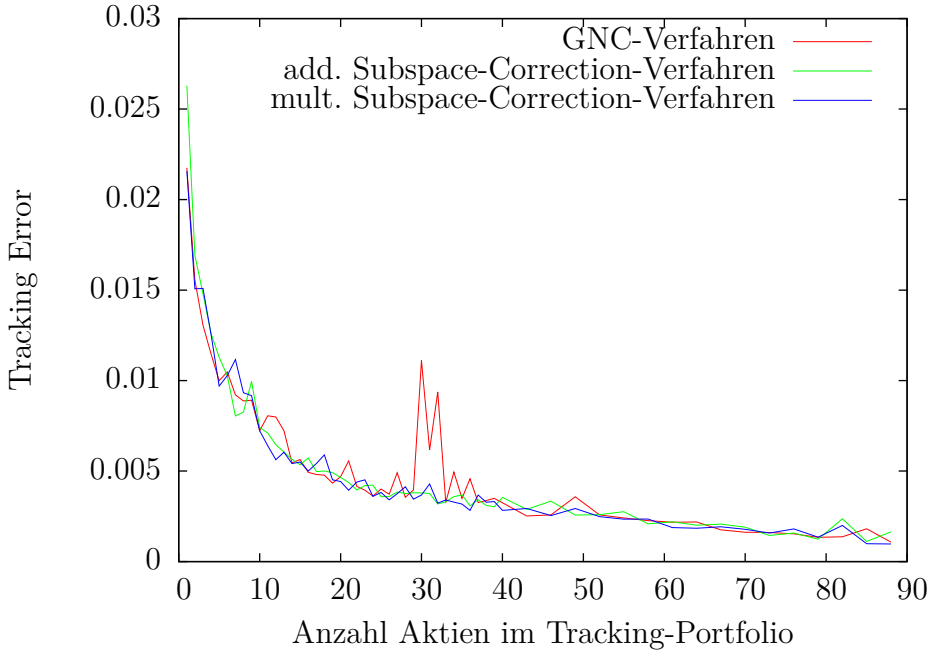

Abbildung 8.4: Tracking Error für Tracking-Portfolios des S&P100 mit verschiedener Anzahl Aktien

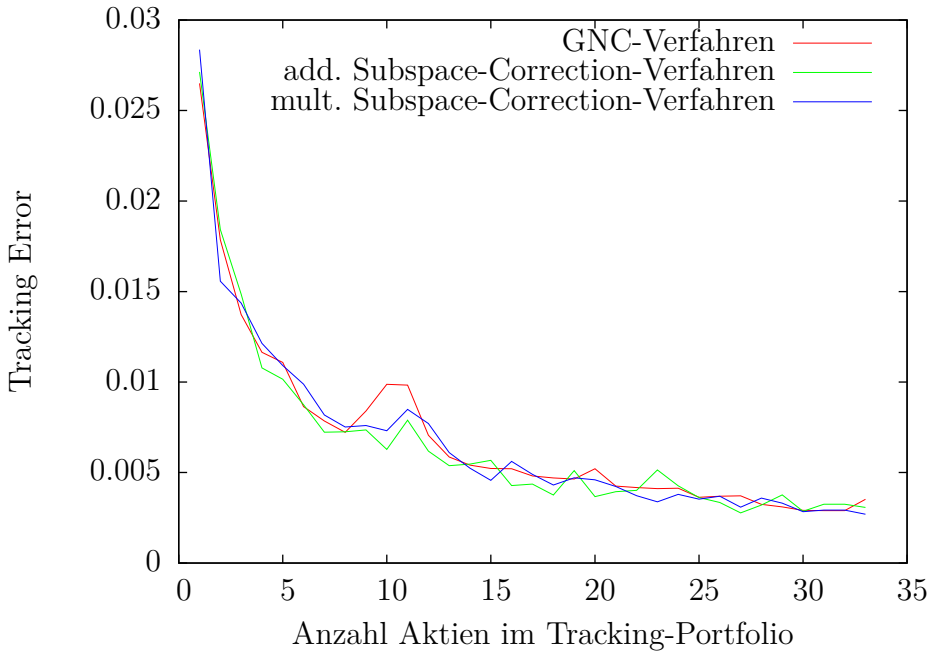

Abbildung 8.5: Tracking Error für Tracking-Portfolios des Nikkei225 mit verschiedener Anzahl Aktien

8.2.5 Nikkei225

Der Nikkei225-Index zeigt in Abbildung 8.5 ein ähnliches Verhalten im Abfall des Tracking Errors wie die vorangegangenen Indices. Ein Tracking Error unter 0.002 scheint sich auch mit mehr Aktien im Portfolio nicht erreichen zu lassen.

8.2.6 S&P500

Beim S&P500-Index ist der Tracking Error für Portfolios mit mehr als 20 Aktien sehr niedrig. Der Tracking Error eines Portofolios aus 10 Aktien beträgt das Fünffache dessen eines Portfolios aus 25 Aktien, wie in Abbildung 8.6 zu sehen.

8.2.7 Russel3000

Beim größten betrachteten Index, dem Russel3000-Index, sind etwa 50 Aktien nötig, um ein Tracking-Portfolio zu erstellen, für das das Hinzufügen weiterer Aktien den Tracking Error kaum noch verringert. In Abbildung 8.7 ist zur besseren Übersicht der Tracking Error logarithmisch aufgetragen, da für dieser für sehr kleine Portfolios sehr groß ist.

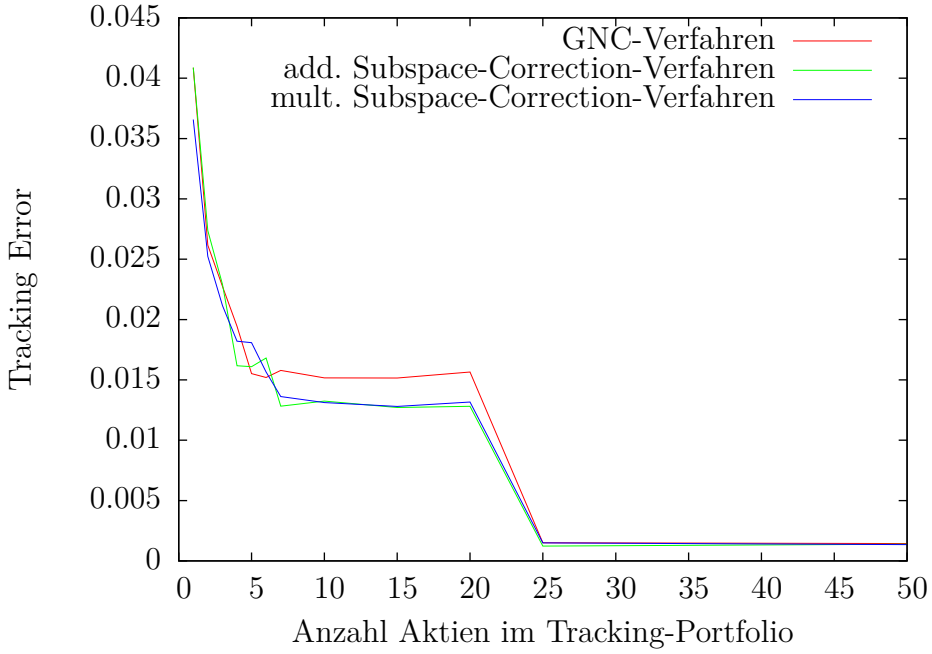

Abbildung 8.6: Tracking Error für Tracking-Portfolios des S&P500 mit verschiedener Anzahl Aktien

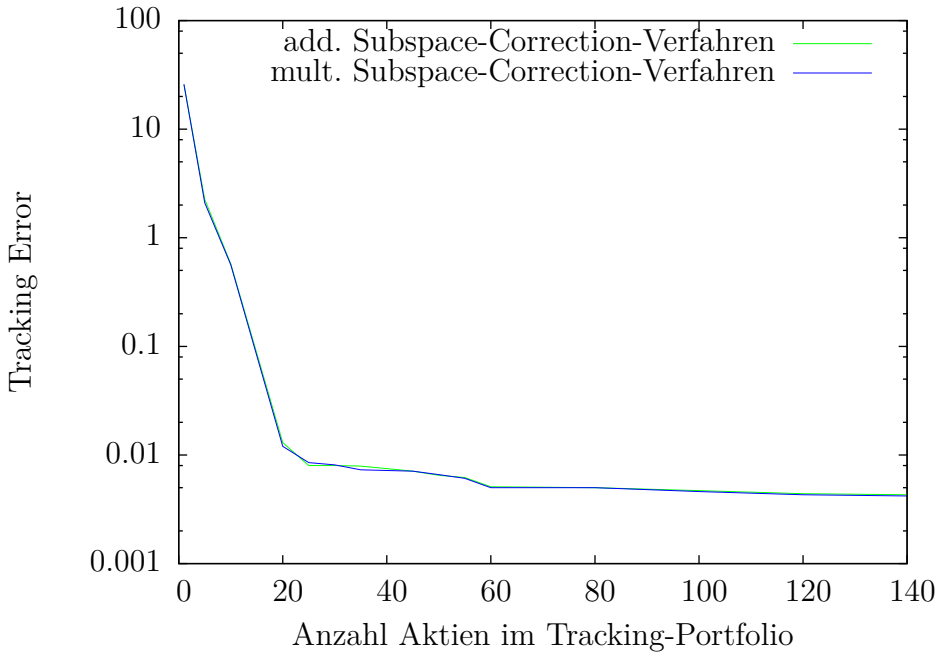

Abbildung 8.7: Tracking Error für Tracking-Portfolios des Russel3000 mit verschiedener Anzahl Aktien

8.3 Gewichtung der Historie

In der Praxis steht man vor der Entscheidung, wie groß die Stichproben-Kovarianzmatrix sein soll, die zur Ermittlung des Tracking-Portfolios benutzt wird - wie viel der bekannten Historie also in die Berechnung einfließen soll. Hat man ein Tracking-Portfolio ermittelt, so möchte man die Entwicklung des Tracking Errors bezüglich des Wertverlaufs des Index beobachten und so ein weiteres Maß für die Qualität des Portfolios erhalten.

Es stellen sich einige interessante Fragen:

- Wie groß sollte der betrachtete Zeitraum der Stichproben sein?
- Liefert ein größerer Beobachtungszeitraum, und damit eine möglicherweise überdefinierte Stichproben-Kovarianzmatrix, ein besseres Tracking-Portfolio?
- Oder liefert ein kleinerer Beobachtungszeitraum bessere Ergebnisse, da ein großer Zeitraum äußere Faktoren wie Firmenpleiten oder etwa Naturkatastrophen überbewertet?
- Wie gut ist der Tracking Error der Portfolios, die mit den vorgestellten Algorithmen bestimmt wurden, außerhalb des „Trainings“-Zeitraumes?
- Wie lange erhält man mit einem festen Tracking-Portfolio einen noch akzeptablen Tracking-Error, bevor man das Portfolio aktualisieren sollte?

Die letzten beiden Fragen werden in Abschnitt 8.5 noch einmal aufgegriffen.

Testreihen: Auswirkungen der Länge und Gewichtung der Historie auf den Tracking Error

Mithilfe der in Kapitel 7.6 vorgestellten Algorithmen wurden Portfolios erstellt, die auf einer kleineren Stichproben-Kovarianzmatrix basieren. Mehrere solche „Trainings“-Zeiträume wurden ausgewählt - von einem halben Jahr bis zu fünf Jahren - und die Qualität der Portfolios verglichen.

Ein weiterer Ansatz zur Verbesserung des Tracking Errors ist der, zeitlich weiter zurückliegende Historie der Aktien im Index geringer zu bewerten als die „aktuelle“ Historie. Dieser Vorgang ist als „exponentielle Glättung“ bekannt und wird etwa in [23] beschrieben.

8.3.1 Hang Seng

Beim Hang-Seng-Index liefert die Tracking-Error-Minimierung die besten Ergebnisse für eine Historienlänge zwischen zwei Jahren und einem Jahr. Bei einer Historienlänge von einem halben Jahr (26 Datenpunkten) steigt der Tracking Error unabhängig von der Gewichtung wieder stark an - dieses Intervall scheint zu klein, um die Kovarianzmatrix der Renditen der Aktienkurse gut zu approximieren. Dieser Effekt lässt sich sowohl bei

Länge der Historie, lineare Gewichtung	Tracking Error ($\times 10^2$)
5 Jahre	0.47
4 Jahre	0.43
3 Jahre	0.37
2 Jahre	0.30
1 Jahr	0.29
0.5 Jahre	0.47
Länge der Historie, exponentielle Gewichtung	Tracking Error ($\times 10^2$)
5 Jahre	0.40
4 Jahre	0.37
3 Jahre	0.34
2 Jahre	0.28
1 Jahr	0.27
0.5 Jahre	0.33

Tabelle 8.8: Ergebnisse Hang Seng, K = 15

linearer als auch bei exponentieller Gewichtung der Historie beobachten. Exponentielle Gewichtung der Historie verbessert den Tracking Error, insbesondere gerade in dem interessanten Intervall zwischen einem und zwei Jahren Historie, wo der Tracking Error noch einmal um 8% (zwei Jahre Historie) bzw. 10% (ein Jahr Historie) verbessert wird, wie in Tabelle 8.8 zu sehen.

8.3.2 DAX100

Beim DAX100-Index zeigt sich kein so einheitliches Bild. In der Tat erhält man das beste Tracking-Portfolio bei linearer Gewichtung der Historie und Historienlänge von einem Jahr, wie Tabelle 8.9 zeigt. Abgesehen davon liefert eine exponentiell abfallende Gewichtung der Historie in allen anderen Fällen bei gleicher Historienlänge einen besseren Tracking Error. Wieder ist jedoch zu sehen, dass ein halbes Jahr Historie zu wenig scheint, um ein gutes Tracking-Portfolio zu erstellen.

8.3.3 FTSE100

Beim FTSE100-Index ist ähnlich wie beim DAX100-Index der beste Tracking Error mit einer linear gewichteten Historie mit einer Länge von einem Jahr erreichbar. Wieder ist - bis auf die Historienlängen vier Jahre und ein Jahr - der Tracking Error bei exponentieller Gewichtung besser, wie Tabelle 8.10 zeigt.

Länge der Historie, lineare Gewichtung	Tracking Error ($\times 10^2$)
5 Jahre	0.36
4 Jahre	0.43
3 Jahre	0.36
2 Jahre	0.32
1 Jahr	0.23
0.5 Jahre	0.54
Länge der Historie, exponentielle Gewichtung	Tracking Error ($\times 10^2$)
5 Jahre	0.36
4 Jahre	0.33
3 Jahre	0.25
2 Jahre	0.27
1 Jahr	0.22
0.5 Jahre	0.37

Tabelle 8.9: Ergebnisse DAX100, K = 25

Länge der Historie, lineare Gewichtung	Tracking Error ($\times 10^2$)
5 Jahre	0.45
4 Jahre	0.32
3 Jahre	0.30
2 Jahre	0.28
1 Jahr	0.16
0.5 Jahre	0.41
Länge der Historie, exponentielle Gewichtung	Tracking Error ($\times 10^2$)
5 Jahre	0.38
4 Jahre	0.35
3 Jahre	0.26
2 Jahre	0.20
1 Jahr	0.16
0.5 Jahre	0.68

Tabelle 8.10: Ergebnisse FTSE100, K = 25

Länge der Historie, lineare Gewichtung	Tracking Error ($\times 10^2$)
5 Jahre	0.37
4 Jahre	0.36
3 Jahre	0.32
2 Jahre	0.29
1 Jahr	0.13
0.5 Jahre	0.51
Länge der Historie, exponentielle Gewichtung	Tracking Error ($\times 10^2$)
5 Jahre	0.45
4 Jahre	0.34
3 Jahre	0.30
2 Jahre	0.24
1 Jahr	0.21
0.5 Jahre	0.38

Tabelle 8.11: Ergebnisse S&P100, K = 25

8.3.4 S&P100

Wie Tabelle 8.11 zeigt, ist wieder die lineare Gewichtung für eine Historien-Länge von einem Jahr die beste Wahl.

8.3.5 Nikkei225

Auch beim Nikkei225-Index, siehe Tabelle 8.12, ist die 1-Jahres-Historie mit exponentieller Gewichtung die beste Wahl.

8.3.6 S&P500

Wie bereits bei den vorherigen Indizes ergibt sich auch beim S&P500-Index bei einer linear gewichteten 1-Jahres-Historie der beste Tracking Error. Dabei ist die Differenz zu längeren und kürzeren Historien besonders ausgeprägt, zu sehen in Tabelle 8.13.

8.3.7 Russel3000

Beim Russel3000-Index dagegen sind, wie in Tabelle 8.14 gezeigt, exponentielle und lineare Gewichtung bei kurzen Historien fast gleichwertig. Den besten Tracking Error erhält man bei einer exponentiell gewichteten Historie von einem Jahr Länge.

Länge der Historie, lineare Gewichtung	Tracking Error ($\times 10^2$)
5 Jahre	0.35
4 Jahre	0.36
3 Jahre	0.24
2 Jahre	0.18
1 Jahr	0.20
0.5 Jahre	0.77
Länge der Historie, exponentielle Gewichtung	Tracking Error ($\times 10^2$)
5 Jahre	0.31
4 Jahre	0.26
3 Jahre	0.26
2 Jahre	0.20
1 Jahr	0.18
0.5 Jahre	0.42

Tabelle 8.12: Ergebnisse Nikkei225, K = 30

Länge der Historie, lineare Gewichtung	Tracking Error ($\times 10^2$)
5 Jahre	0.59
4 Jahre	0.47
3 Jahre	0.42
2 Jahre	0.36
1 Jahr	0.17
0.5 Jahre	1.12
Länge der Historie, exponentielle Gewichtung	Tracking Error ($\times 10^2$)
5 Jahre	0.56
4 Jahre	0.48
3 Jahre	0.44
2 Jahre	0.38
1 Jahr	0.24
0.5 Jahre	0.67

Tabelle 8.13: Ergebnisse S&P500, K = 25

Länge der Historie, lineare Gewichtung	Tracking Error ($\times 10^2$)
5 Jahre	0.63
4 Jahre	0.61
3 Jahre	0.53
2 Jahre	0.41
1 Jahr	0.41
0.5 Jahre	0.43
Länge der Historie, exponentielle Gewichtung	Tracking Error ($\times 10^2$)
5 Jahre	0.64
4 Jahre	0.60
3 Jahre	0.55
2 Jahre	0.42
1 Jahr	0.40
0.5 Jahre	0.42

Tabelle 8.14: Ergebnisse Russel3000, K = 50

8.4 Laufzeittests und Vergleich mit anderen Verfahren

Alle Laufzeittests wurden mit MATLAB Version 8.0.0.783 unter Ubuntu 10.04 auf einem 2.4GHZ Intel E6600 Prozessor mit 2GB RAM durchgeführt. Random (1000) bedeutet dabei, dass 1000 zufällig bestimmte Portfolios berechnet wurden. Genetic Programming (1000) bedeutet, dass eine Population von 1000 zufällig gewählten Portfolios 1000 mal durch die Schritte Mutation, Rekombination und Selektion optimiert wurde. Bei Hybrid Genetic wurde dann das erhaltene Portfolio des Genetic-Programming-Ansatzes entsprechend Abschnitt 6.8.2 nachoptimiert.

8.4.1 Hang Seng

Verfahren	Tracking Error ($\times 10^2$)	Laufzeit (s)
Random (1000)	0.69	0.16
Genetic Programming (1000)	0.51	43.09
Hybrid Genetic	0.50	56.02
Graduated Non-Convexity	0.49	59.23
Subspace Correction additiv	0.50	17.60
Subspace Correction multiplikativ	0.29	35.89
Subspace Correction Multi-Level additiv	0.41	23.63
Subspace Correction Multi-Level multiplikativ	0.27	38.14

Tabelle 8.15: Hang Seng

Beim Hang-Seng-Index wurde der Wert K=15 für die Maximalgröße des Portfolios gewählt. Die Länge der Historie wurde auf ein Jahr gesetzt, ausgehend von den Ergebnissen in Abschnitt 8.3.1. In den Ergebnissen in Tabelle 8.15 sieht man das erwartete Ergebnis: Beide Subspace-Correction-Methoden sind schneller als der globale GNC-Algorithmus, das multiplikative Verfahren ist langsamer als das additive, liefert aber das beste Ergebnis. Das globale GNC-Verfahren liefert ein lokales Minimum mit deutlich schlechterem Tracking Error.

8.4.2 Dax100

Beim DAX100, Tabelle 8.16, wurde als Portfolio-Größe K=25, für die Länge der exponentiell gewichteten Historie ein Jahr gewählt. Ähnlich wie beim Hang-Seng-Index liefert das multiplikative Subspace-Correction-Verfahren das beste Ergebnis. Das additive Verfahren liefert schlechtere Ergebnisse als das globale GNC-Verfahren, ist dafür aber fast doppelt so schnell.

htpb

Verfahren	Tracking Error ($\times 10^2$)	Laufzeit (s)
Random (1000)	0.67	10.50
Genetic Programming (1000)	0.52	36.17
Hybrid Genetic	0.49	41.42
GNC	0.29	149.30
Subspace Correction additiv	0.37	78.96
Subspace Correction multiplikativ	0.22	136.95
Subspace Correction Multi-Level additiv	0.35	88.75
Subspace Correction Multi-Level multiplikativ	0.22	132.33

Tabelle 8.16: Ergebnisse DAX100

8.4.3 FTSE100

Verfahren	Tracking Error ($\times 10^2$)	Laufzeit (s)
Random (1000)	0.56	13.55
Genetic Programming (1000)	0.54	28.87
Hybrid Genetic	0.54	34.15
GNC	0.56	86.89
Subspace Correction additiv	0.36	58.43
Subspace Correction multiplikativ	0.19	95.98
Subspace Correction Multi-Level additiv	0.33	69.86
Subspace Correction Multi-Level multiplikativ	0.16	111.15

Tabelle 8.17: FTSE100

Beim FTSE100-Index fällt auf, dass die multiplikativen Subspace-Correction-Verfahren eine längere Laufzeit als das globale GNC-Verfahren haben. Dies zeigt Tabelle 8.17. Dafür ist jedoch auch der Tracking Error deutlich verbessert. Für das Portfolio wurde eine Maximalgröße von 25 Aktien festgelegt, die Historie wurde exponentiell gewichtet, mit Länge von einem Jahr.

8.4.4 S&P100

Verfahren	Tracking Error ($\times 10^2$)	Laufzeit (s)
Random (1000)	0.60	4.98
Genetic Programming (1000)	0.21	109.39
Hybrid Genetic	0.21	116.73
GNC	0.21	117.20
Subspace Correction additiv	0.23	69.86
Subspace Correction multiplikativ	0.21	106.78
Subspace Correction Multi-Level additiv	0.22	73.94
Subspace Correction Multi-Level multiplikativ	0.21	111.1

Tabelle 8.18: S&P100

Für den S&P100 wurde wieder $K = 25$ gewählt, Historienlänge ein Jahr. Die meisten betrachteten Algorithmen erreichen ein sehr ähnliches Ergebnis, wie Tabelle 8.18 zeigt. Das additive Verfahren ist wieder etwas schlechter als die übrigen, aber bis auf die Wahl eines zufälligen Portfolios auch das schnellste.

8.4.5 Nikkei225

Verfahren	Tracking Error ($\times 10^2$)	Laufzeit (s)
Random (1000)	0.52	5.82
Genetic Programming (1000)	0.41	244.53
Hybrid Genetic	0.40	248.33
GNC	0.35	610.49
Subspace Correction additiv	0.31	358.82
Subspace Correction multiplikativ	0.24	402.61
Subspace Correction Multi-Level additiv	0.30	389.31
Subspace Correction Multi-Level multiplikativ	0.18	428.71

Tabelle 8.19: Nikkei225

Auch für den Nikkei225-Index wurde eine Historienlänge von einem Jahr gewählt, und das Tracking-Portfolio sollte aus maximal 25 verschiedenen Aktien bestehen. Für diesen Index lieferte das multiplikative Multi-Level-Subspace-Correction-Verfahren das beste Ergebnis, wie in Tabelle 8.19 zu sehen. Dabei ist jedoch der Laufzeitgewinn verglichen mit dem Standard-GNC-Verfahren recht gering.

8.4.6 S&P500

Verfahren	Tracking Error ($\times 10^2$)	Laufzeit (s)
Random (1000)	0.92	1.00
Genetic Programming (1000)	0.47	318.65
Hybrid Genetic	0.32	325.41
GNC	0.27	6623.24
Subspace Correction additiv	0.28	1056.83
Subspace Correction multiplikativ	0.19	1298.51
Subspace Correction Multi-Level additiv	0.26	2501.76
Subspace Correction Multi-Level multiplikativ	0.17	2831.34

Tabelle 8.20: S&P500

In Tabelle 8.20 sind die Ergebnisse für den S&P-500-Index zu sehen. Wie bei den übrigen Indizes wurde ausgehend von den Testergebnissen in Abschnitt 8.2 die linear gewichtete Historie auf ein Jahr Länge festgesetzt, das Portfolio sollte aus 25 Aktien bestehen. Wieder liefern die Multi-Level-Verfahren das beste Ergebnis.

8.4.7 Russel3000

Verfahren	Tracking Error ($\times 10^2$)	Laufzeit (s)
Random (10000)	1.98	18.91
Genetic Programming (10000)	1.37	2135.13
Hybrid Genetic	1.27	2198.54
Subspace Correction add.	0.91	3668015.62
Subspace Correction mult.	0.80	4942046.78
Subspace Correction Multi-Level add.	0.90	98421315.78
Subspace Correction Multi-Level mult.	0.80	108199268.57

Tabelle 8.21: Russel3000

Beim größten betrachteten Index, dem Russel3000 in Tabelle 8.21, fällt zunächst die sehr hohe Laufzeit der Subspace-Correction-Verfahren auf. Dafür liefern auch alle betrachteten Subspace-Correction-Varianten einen gegenüber den anderen betrachteten Verfahren verbesserten Tracking Error.

8.5 On-line Tracking-Error-Minimierung

Wie bereits erwähnt ist die Zusammensetzung eines Aktienindex mit der Zeit keineswegs stets dieselbe. Firmen lösen sich auf oder erfüllen aus anderen Gründen die Auswahlkriterien des Index nicht mehr. Aber auch wenn die Zusammensetzung des Index gleich bleibt, unterliegt die Wertentwicklung der Firmen, deren Aktien den Index bilden, verschiedensten Faktoren. Deshalb ist es für den Investor von Interesse, von Zeit zu Zeit das Tracking-Portfolio zu prüfen und gegebenenfalls zu optimieren. Daher beschäftigt sich dieses Kapitel mit der Frage, wie lange ein berechnetes Tracking-Portfolio nach seiner Erstellung noch einen akzeptablen Tracking Error liefert.

Update-Intervalle für Tracking-Portfolios

In der folgenden Testreihe wurde jeweils ein Tracking-Portfolio mit einem Jahr Historie erstellt. Die Wahl des verwendeten Algorithmus wurde ausgehend von den Ergebnissen der vorangehenden Kapitel getroffen. Daraufhin wurde für die verbleibenden vier Jahre vorhandener Historie alle 52 Wochen ein neues Tracking-Portfolio erstellt und der Tracking Error des neues Portfolios mit dem des alten verglichen. In den Tabellen sind jeweils die Tracking-Error-Differenzen zwischen den einzelnen Portfolios zu sehen. Eine Zeile gibt dabei die aktuell betrachtete Woche an, eine Spalte die Differenz im Tracking Error zwischen dem aktuell erstellten Portfolio und dem Portfolio aus der Spaltenüberschrift. In den Graphen ist jeweils ein Vergleich zwischen Wertverlauf des Index und Wertverlauf der Tracking-Portfolios abgebildet. Allen berechnetemTracking-Portfolios ist gemeinsam, dass der Tracking Error sich zwar nach einem Jahr Tracking verschlechtert hat, dieser Prozess jedoch nicht linear wachsend zu sein scheint - im Gegenteil, der Tracking Error steigt in dem betrachteten Zeitraum nicht beliebig an. Dies bestätigt die mit dem Index Tracking verfolgte Strategie, einen Markt mit einem aus wenigen Aktien bestehenden Portfolio langfristig abbilden zu können.

8.5.1 Hang Seng

Tracking-Error-Differenzen ($\times 10^2$)					
Woche Portfolio	52	104	156	208	260
52	0.0				
104	0.05	0.0			
156	0.07	0.05	0.0		
208	0.07	0.04	0.01	0	
260	0.06	0.05	0.00	0.01	0.0

Tabelle 8.22: Tracking-Error-Differenzen Hang Seng

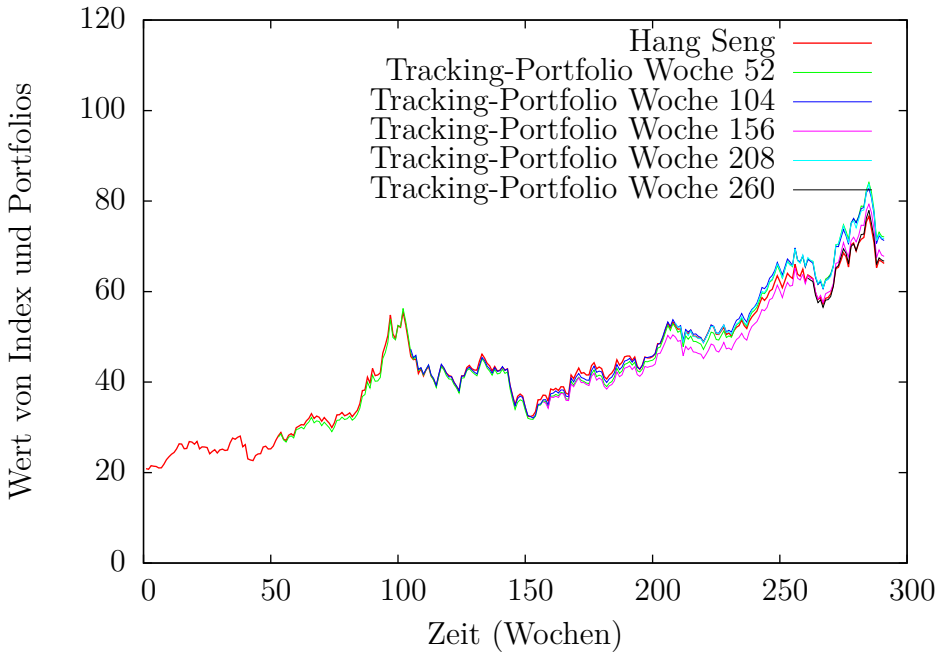

Abbildung 8.8: Verlauf des Hang Seng und der berechneten Tracking-Portfolios

Der Hang-Seng-Index lässt sich langfristig fast perfekt tracken, wie Tabelle 8.22 zeigt. Die Differenzen im Tracking Error zwischen den zu verschiedenen Zeitpunkten erstellten Portfolios sind gering. Auch nach über vier Jahren entspricht der Verlauf des mit einer Historie von 52 Wochen erstellten Tracking-Portfolios noch fast exakt dem Verlauf des Index, wie in Abbildung 8.8 zu sehen.

8.5.2 Dax100

Tracking-Error-Differenzen ($\times 10^2$)					
Woche	52	104	156	208	260
52	0.0				
104	0.01	0.0			
156	0.27	0.26	0.0		
208	0.22	0.20	0.01	0	
260	0.32	0.31	0.05	0.10	0.0

Tabelle 8.23: Tracking-Error-Differenzen Dax100

Beim Dax100 zeigen alle Portfolios bis Ende des betrachteten Zeitraums einen ähnlichen Verlauf wie der Index, wie in Abbildung 8.9 zu sehen. Tabelle 8.23 zeigt die Differenzen im Tracking Error zwischen den einzelnen Portfolios. Mit wachsendem zeitlichen Abstand

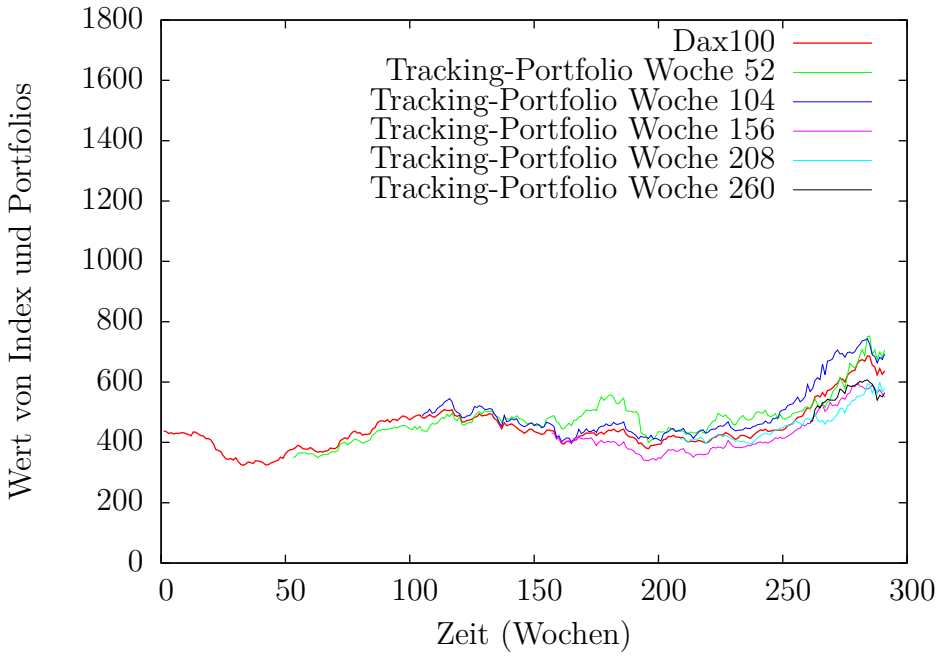

Abbildung 8.9: Verlauf des Dax100 und der berechneten Tracking-Portfolios

zwischen der Erstellung der Portfolios wächst dieser an.

8.5.3 FTSE100

Tracking-Error-Differenzen ($\times 10^2$)					
Woche	52	104	156	208	260
52	0.0				
104	0.01	0.0			
156	0.27	0.26	0.0		
208	0.22	0.20	0.01	0	
260	0.32	0.31	0.05	0.10	0.0

Tabelle 8.24: Tracking-Error-Differenzen FTSE100

Beim FTSE100 zeigen auch nach langer Zeit die früh erstellten Tracking-Portfolios in Abbildung 8.10 noch einen ähnlichen Verlauf, doch der Gesamtwert der Portfolios weicht nach etwa 2 Jahren recht stark ab, wie auch Tabelle 8.24 andeutet.

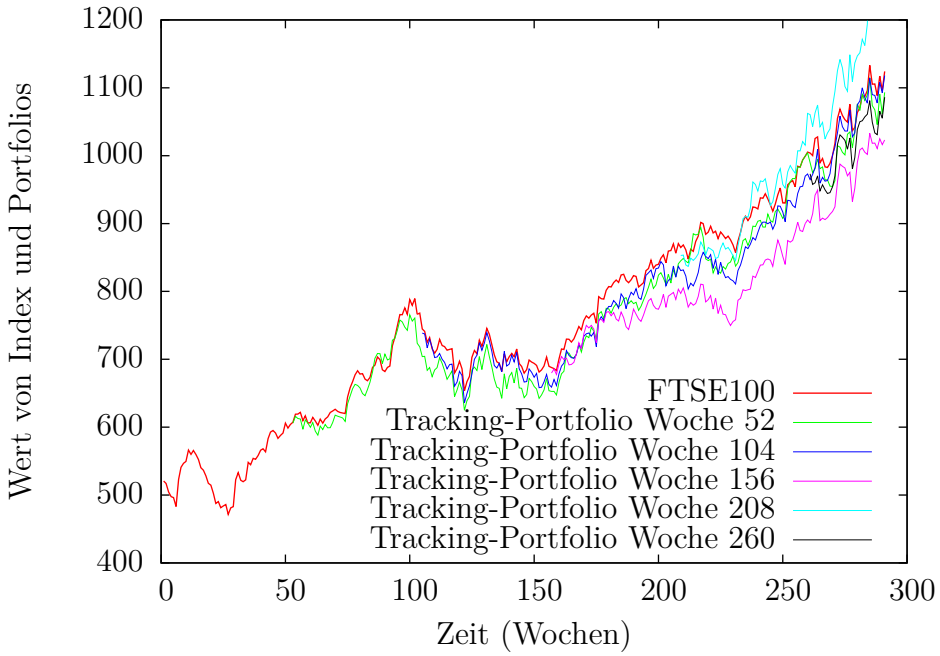

Abbildung 8.10: Verlauf des FTSE100 und der berechneten Tracking-Portfolios

Tracking-Error-Differenzen ($\times 10^2$)					
Woche	52	104	156	208	260
52	0.0				
104	0.03	0.0			
156	0.14	0.11	0.0		
208	0.24	0.21	0.10	0	
260	0.35	0.32	0.11	0.11	0.0

Tabelle 8.25: Tracking-Error-Differenzen S&P100

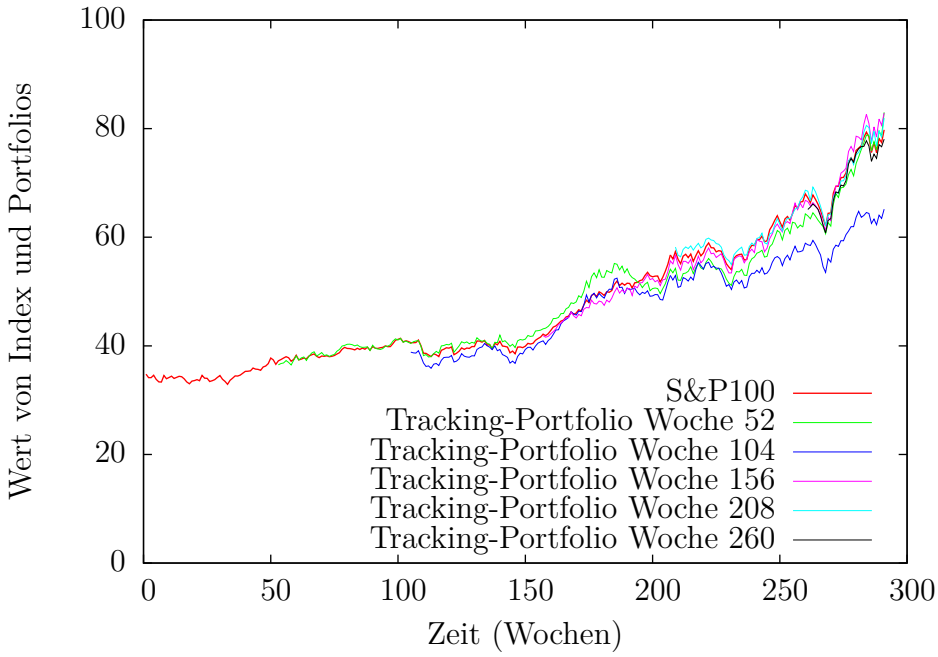

Abbildung 8.11: Verlauf des S&P100 und der berechneten Tracking-Portfolios

8.5.4 S&P100

Der S&P100-Index lässt sich je etwa ein Jahr lang gut tracken, danach zeigen die Tracking-Portfolios teilweise starke Abweichungen, wie in Abbildung 8.11 zu sehen. Tabelle 8.25 gibt die Tracking-Error-Abweichungen der Portfolios an.

8.5.5 Nikkei225

Wie auch beim S&P100-Index lässt sich beim Nikkei225 über ein Jahr ein gutes Tracking erreichen, bevor die Tracking-Portfolios in ihrem Wertverlauf stark abzuweichen beginnen. Abbildung 8.12 zeigt den Verlauf der Tracking-Portfolios, Tabelle 8.26 die Unterschiede im Tracking Error zwischen den Portoflios.

Tracking-Error-Differenzen ($\times 10^2$)					
Woche	52	104	156	208	260
52	0.0				
104	0.03	0.0			
156	0.14	0.11	0.0		
208	0.24	0.21	0.10	0	
260	0.35	0.32	0.11	0.11	0.0

Tabelle 8.26: Tracking-Error-Differenzen Nikkei225

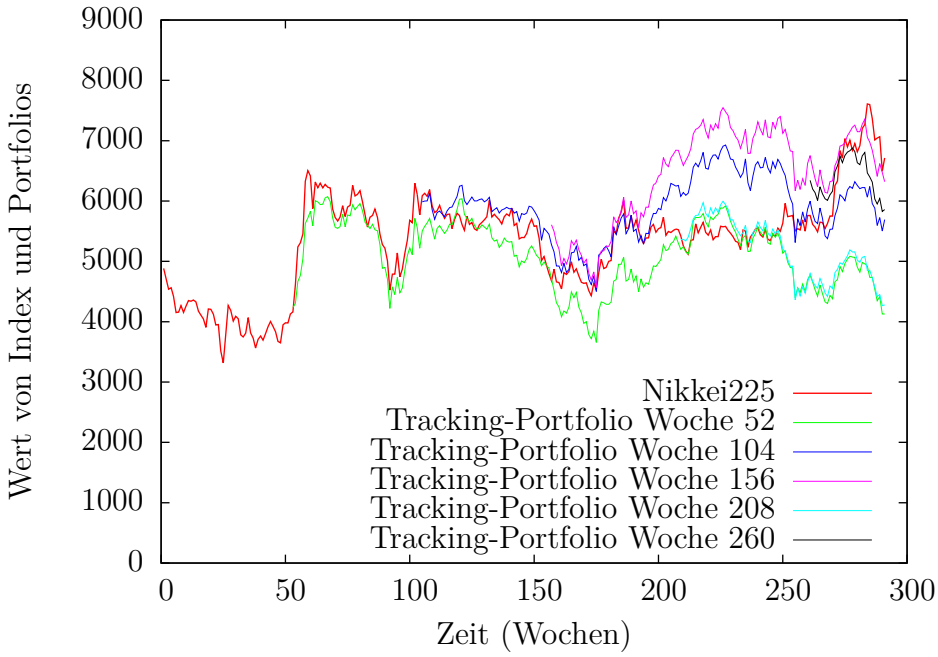

Abbildung 8.12: Verlauf des Nikkei225 und der berechneten Tracking-Portfolios

8.5.6 S&P500

Tracking-Error-Differenzen ($\times 10^2$)					
Woche	52	104	156	208	260
52	0.0				
104	0.03	0.0			
156	0.14	0.11	0.0		
208	0.24	0.21	0.10	0	
260	0.35	0.32	0.11	0.11	0.0

Tabelle 8.27: Tracking-Error-Differenzen S&P500

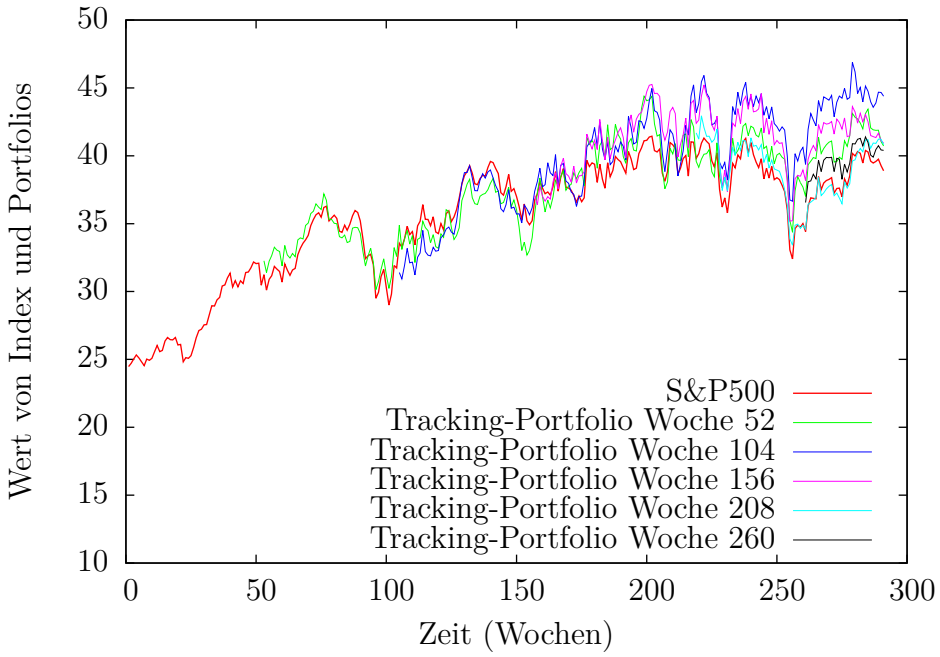

Abbildung 8.13: Verlauf des S&P500 und der berechneten Tracking-Portfolios

Für den S&P500-Index liefern die Algorithmen ein über 2 Jahre hinweg passables Tracking, wie in Abbildung 8.5.6 und Tabelle 8.27 gezeigt.

8.5.7 Russel3000

Tracking-Error-Differenzen ($\times 10^2$)					
Woche	52	104	156	208	260
52	0.0				
104	0.03	0.0			
156	0.11	0.13	0.0		
208	0.10	0.15	0.08	0	
260	0.14	0.18	0.09	0.03	0.0

Tabelle 8.28: Tracking-Error-Differenzen Russel2000

Auch für den Russel3000-Index zeigen Tabelle 8.28 und Abbildung 8.14 das bereits vertraute Bild eines guten Trackings über ein bis zwei Jahre, bevor die Tracking-Portfolios in ihrem Werteverlauf stärker abzuweichen beginnen.

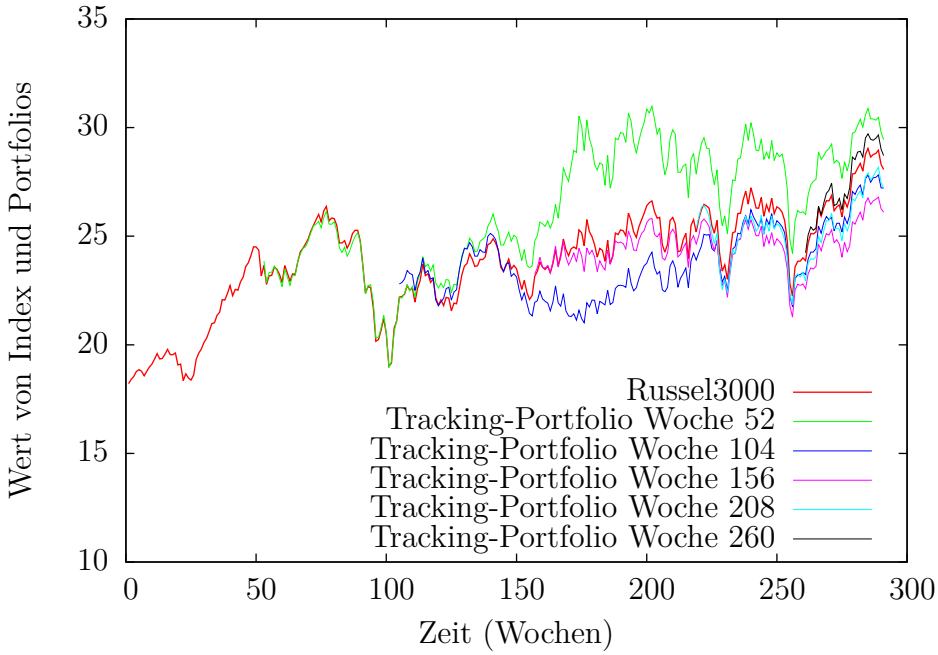

Abbildung 8.14: Verlauf des Russel3000 und der berechneten Tracking-Portfolios

9 Zusammenfassung und Ausblick

9.1 Zusammenfassung

Mit der Tracking-Error-Minimierung per Subspace-Correction existieren nun deterministische Algorithmen zum Lösen eines hochdimensionalen Optimierungsproblems. Die Algorithmen verbinden zwei Techniken, die zur Behandlung hochdimensionaler Probleme oft herangezogen werden: Einerseits, die Graduated Non-Convexity, andererseits die Subspace Correction. Die berechneten Portfolios sind geeignet, mit dem Erwerb weniger verschiedener Aktien einen Aktienindex langfristig abzubilden - in einigen Fällen sogar über mehrere Jahre. Dies entspricht den Anforderungen einer langfristigen, passiven Handelsstrategie. Die implementierte Teilraumzerlegung verbesserte dabei in viele Fällen sowohl die Laufzeit als auch das Tracking-Error-Ergebnis gegenüber einem globalen Verfahren, da der Divide-and-Conquer-Ansatz weniger anfällig war, nur ein wenig optimales lokales Minimum zu finden. Im Fall des größten betrachteten Index, dem Russel3000, kapitulierte das betrachtete globale Verfahren sogar gänzlich.

9.2 Ausblick

Bei den in dieser Arbeit betrachteten Indizes handelte es sich ausschließlich um Aktienindizes. Da mit Hilfe der Tracking-Error-Minimierung eine sehr langfristige Investitionsstrategie verfolgt werden kann, ist denkbar, dass auch andere, im Gegensatz zu Aktienindizes weniger liquide Indizes, getrackt werden können. Beispiele hierfür sind etwa Regionen-Indizes, Hedge-Fond-Indizes oder Immobilien-Fonds. Weiterhin denkbar ist eine direkte Modellierung von Transaktionskosten, hierfür existieren bereits einige Modelle. Die in dieser Arbeit vorgestellten additiven Methoden profitieren außerdem stark von Parallelisierung, dies würde die Laufzeit der Algorithmen bei Optimierungsproblemen mit sehr vielen Variablen stark verringern.

Literaturverzeichnis

[1] Beasley, J. E.: *Obtaining test problems via Internet.* Journal of Global Optimization, S. 429–433, 1996.

[2] Beasley, J. E., N. Meade und T.-J. Chang: *An evolutionary heuristic for the index tracking problem.* European Journal of Operational Research, 148(3):621 – 643, 2003.

[3] Bejan, A.: *Largest eigenvalues and sample covariance matrices.* Masterarbeit, Department of Statistics, University of Warwick, Great Britain, 2005.

[4] Berger, M.: *Nonlinearity & Functional Analysis: Lectures on Nonlinear Problems in Mathematical Analysis.* Pure and Applied Mathematics. Elsevier Science, 1977.

[5] Boggs, P. T. und J. W. Tolle: *Sequential Quadratic Programming.* Acta Numerica, 4:1–51, 0 1995.

[6] Broyden, C. G.: *A class of methods for solving nonlinear simultaneous equations.* Mathematics of Computation, 19:577–593, 1965.

[7] Canakgoz, N. A. und J. E. Beasley: *Mixed-integer programming approaches for index tracking and enhanced indexation.* European Journal of Operational Research, 196(1):384 – 399, 2009.

[8] Coleman, T. F., Y. Li und J. Henniger: *Minimizing Tracking Error While Restricting the Number of Assets*, 2004.

[9] Conn, A., N. Gould und P. Toint: *Trust Region Methods.* MPS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, 2000.

[10] Corielli, F. und M. Marcellino: *Factor Based Index Tracking.* CEPR Discussion Papers 3265, C.E.P.R. Discussion Papers, 2002.

[11] Dantzig, G. B.: *Maximization of a Linear Function of Variables Subject to Linear Inequalities, in Activity Analysis of Production and Allocation*, Kap. XXI. Wiley, New York, 1951.

[12] Derigs, U. und N. H. Nickel: *On a local-search heuristic for a class of tracking error minimization problems in portfolio management.* Annals of Operations Research, 131(1-4):45–77, Okt. 2004.

[13] Gilli, M. und E. Këllezi: *Threshold Accepting for Index Tracking*, 2001.

[14] Grötschel, M. und L. Lovász and A. Schrijver: *The ellipsoid method and its consequences in combinatorial optimization.* Combinatorica, 1(2):169–197, 1981.

[15] Haslem, J., H. Baker und D. Smith: *Performance and Characteristics of Actively Managed Retail Mutual Funds with Diverse Expense Ratios.* Financial services review, 2008.

[16] I.r Ekeland, R. T.: *Convex Analysis and Variational Problems.* SIAM, 1999.

[17] Jansen, R. und R. van Dijk: *Optimal Benchmark Tracking with Small Portfolios.* The Journal of Portfolio Management, 28(2):33–39, 2002.

[18] Kirkpatrick, S., C. D. Gelatt und M. P. Vecchi: *Optimization by simulated annealing.* science, 220(4598):671–680, 1983.

[19] Ledoit, O. und M. Wolf: *Improved Estimation of the Covariance Matrix of Stock Returns with an Application to Portfolio Selection.* Journal of Empirical Finance, 10:603–621, 2003.

[20] Nesterov, Y.: *Introductory Lectures on Convex Optimization: A Basic Course.* Kluwer, 2003.

[21] Nocedal, J. und M. Overton: *Projected Hessian updating algorithms for nonlinear constrained optimization.* SIAM Journal on Numerical Analysis, (95), 1985.

[22] Nocedal, J. und S. J. Wright: *Numerical Optimization.* Springer, Aug. 2000.

[23] Pozzi, F., T. Matteo, und T. Aste: *Exponential smoothing weighted correlations.* The European Physical Journal B, 85(6):1–21, 2012.

[24] Richtárik, P.: *Some Algorithms for Large-Scale Linear and Convex Minimization in Relative Scale.* Doktorarbeit, School of Operations Research and Information Engineering, Cornell University, 2007.

[25] Rockafellar, R.: *Convex Analysis.* Princeton landmarks in mathematics and physics series. Princeton University Press, 1997.

[26] Schittkowski, K.: *On the convergence of a sequential quadratic programming method with an augmented lagrangian line search function.* Mathematische Operationsforschung und Statistik, Series Optimization, 14:1–20, 1983.

[27] Shor, N.: *Minimization methods for non-differentiable functions.* Springer series in computational mathematics. Springer-Verlag, 1985.

[28] Tai, X.-C. und J. Xu: *Global and uniform convergence of subspace correction methods for some convex optimization problems.* Math. Comp, 71:105–124, 1999.

[29] Walras, L.: *Éléments d'économie politique pure, ou, Théorie de la richesse sociale.* 1896.

[30] Wolfe, P.: *The simplex method for quadratic programming.*

[31] Xu, J.: *Iterative methods by space decomposition and subspace correction.* SIAM Review, 34:581–613, 1992.

[32] Zhu, H., Y. Chen, und K. Wang: *A particle swarm optimization heuristic for the index tacking problem.* In: *Proceedings of the 7th international conference on Advances in Neural Networks - Volume Part I*, ISNN'10, S. 238–245, Berlin, Heidelberg, 2010. Springer-Verlag.